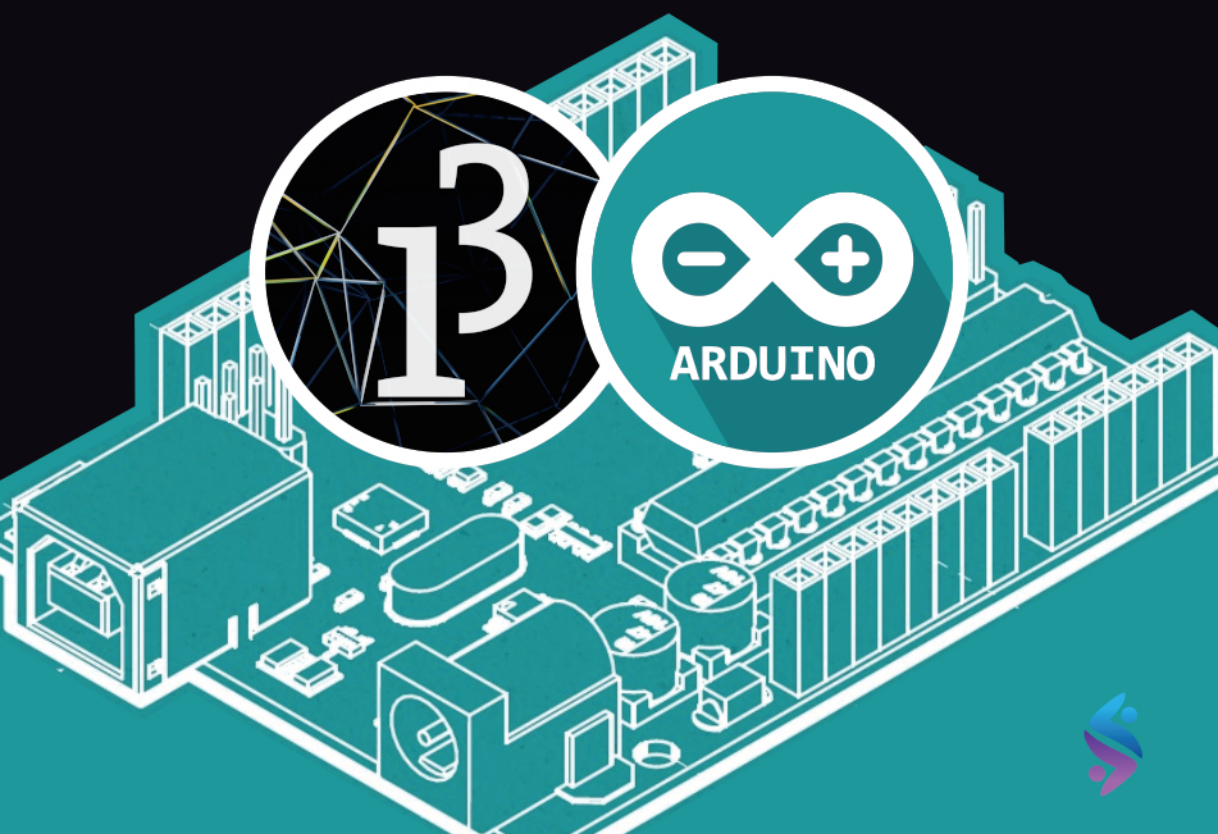
DESAIN INTERFACE GRAFIS ARDUINO DENGAN BAHASA PEMROGRAMAN PROCESSING

Slamet Winardi, ST., MT
Dr. Sri Wiwoho Mudjanarko, ST., MT



DESAIN INTERFACE GRAFIS **ARDUINO** DENGAN BAHASA PEMROGRAMAN **PROCESSING**

Slamet Winardi, ST, MT
Dr. Sri Wiwoho Mudjanarko ST., MT

Program Studi Sistem Komputer
Universitas Narotama Surabaya
2019

SCOPINDO
MEDIA PUSTAKA

DESAIN INTERFACE GRAFIS **ARDUINO** DENGAN BAHASA PEMROGRAMAN **PROCESSING**

Author :
Slamet Winardi, ST., MT.
Dr. Sri Wiwoho Mudjanarko, ST., MT.

Layouter :
Natasha AI

Editor :
Nur Azizah

Design Cover :
Azizur Rachman

Penerbit

Scopindo Media Pustaka
Jl. Menanggal III No. 45, Menanggal, Gayungan, Surabaya 60245
Telp. 0811300229
scopindomedia@gmail.com

ISBN : 978-623-92163-4-4

KATA PENGANTAR

Dengan menyebut nama Tuhan Yang Maha Esa yang selalu memberikan kesehatan dan keberkahan sehingga dapat menyelesaikan buku dari hasil hibah penelitian Ristekdikti. Terimakasih juga disampaikan kepada Rektor Universitas Narotama dan Lembaga Penelitian dan Pengabdian kepada Masyarakat yang selalu memberikan suportnya kepada penulis.

Semoga Buku ini dapat bermanfaat bagi orang-orang pendidikan yang membutuhkan ilmu tentang Arduino dan User Interface grafis dengan memanfaatkan software Processing. Buku ini akan digunakan pada mata kuliah Desain Hardware sebagai bentuk integrasi antara Penelitian dan Pengajaran. Berikutnya akan dikembangkan untuk tampilan grafis di Internet of Things (IoT) agar tampilan lebih cantik dan menarik.

Tidak ada yang sempurna di dunia ini, begitu juga tulisan ini, masih butuh masukan dan kritik dari pembaca agar buku yang dipersembahkan ini menjadi lebih baik lagi. Setelah buku ini akan dibuat lagi buku-buku tentang pemrograman Processing untuk mendukung kinerja Mikrokontroler Arduino, Wemos, maupun ESP.

Surabaya, Oktober 2019
Penulis

DAFTAR ISI

BAB I
OVERVIEW

Preview

Processing adalah sketsa perangkat lunak yang fleksibel dan bahasa untuk mempelajari cara membuat kode dalam konteks seni visual. Sejak tahun 2001, Processing telah mempromosikan literasi perangkat lunak dalam seni visual dan literasi visual dalam teknologi. Ada puluhan ribu siswa, seniman, perancang, peneliti, dan penggemar yang menggunakan Pemrosesan untuk belajar dan membuat prototipe.

Selama enam belas tahun terakhir, Processing telah mempromosikan literasi perangkat lunak, khususnya dalam seni visual, dan literasi visual dalam teknologi. Awalnya dibuat untuk berfungsi sebagai buku sketsa perangkat lunak dan untuk mengajarkan dasar-dasar pemrograman dalam konteks visual, Pemrosesan juga telah berkembang menjadi alat pengembangan bagi para profesional. Perangkat lunak pengolah gratis dan open source, dan berjalan pada platform Mac, Windows, dan GNU / Linux.

Processing terus menjadi alternatif untuk perangkat lunak berpemilik dengan lisensi terbatas dan mahal, membuatnya dapat diakses oleh sekolah dan siswa secara perorangan. Status sumber terbukanya mendorong partisipasi dan kolaborasi masyarakat yang penting bagi pertumbuhan Pemrosesan. Kontributor membagikan program, berkontribusi kode, dan membangun library, alat, dan mode untuk memperluas kemungkinan perangkat lunak. Komunitas Processing telah menulis lebih dari seratus library untuk memfasilitasi penglihatan komputer, visualisasi data, komposisi musik, jaringan, ekspor file 3D, dan pemrograman elektronik.

Pendidikan

Sejak awal, Processing dirancang sebagai bahasa pemrograman pertama. Itu terinspirasi oleh bahasa sebelumnya seperti BASIC dan Logo, serta pengalaman kami sebagai siswa dan pengajaran kurikulum yayasan seni visual. Elemen yang sama diajarkan di kelas sains komputer sekolah menengah atau universitas diajarkan melalui Pengolahan, tetapi dengan penekanan yang berbeda. Processing diarahkan untuk menciptakan media visual dan interaktif, sehingga program pertama dimulai dengan menggambar. Siswa yang baru dalam pemrograman merasa sangat memuaskan untuk membuat sesuatu muncul di layar mereka dalam beberapa saat setelah menggunakan perangkat lunak. Kurikulum yang memotivasi ini telah terbukti berhasil untuk mengarahkan siswa desain, seni, dan arsitektur ke dalam pemrograman dan untuk melibatkan badan siswa yang lebih luas di kelas ilmu komputer umum.

Processing digunakan di ruang kelas di seluruh dunia, sering di sekolah seni dan program seni visual di universitas, tetapi juga sering ditemukan di sekolah menengah, program ilmu komputer, dan kurikulum humaniora. Museum seperti Exploratorium di San Francisco menggunakan Processing untuk mengembangkan pameran mereka. Dalam survei yang disponsori oleh National Science Foundation, siswa dalam kursus komputasi pengantar tingkat perguruan tinggi yang diajarkan dengan Processing di Bryn Mawr College mengatakan bahwa mereka akan dua kali lebih mungkin untuk mengambil kelas ilmu komputer yang lain dibandingkan dengan siswa di kelas dengan kurikulum yang lebih tradisional.

Inovasi dalam pengajaran melalui Processing telah disesuaikan untuk tutorial ilmu komputer Khan Academy, yang ditawarkan secara online secara gratis. Tutorial dimulai dengan menggambar, sebagian besar fungsi Processing digunakan

untuk menggambar. Pendekatan Processing juga telah diterapkan pada elektronik melalui proyek Arduino dan Pengkabelan. Arduino menggunakan sintaks yang diilhami oleh yang digunakan dengan Processing, dan terus menggunakan versi modifikasi dari lingkungan pemrograman Processing untuk membuatnya lebih mudah bagi siswa untuk belajar bagaimana memprogram robot dan banyak proyek elektronik lainnya.

Budaya

Perangkat lunak Processing digunakan oleh ribuan desainer visual, seniman, dan arsitek untuk membuat karya mereka. Proyek yang dibuat dengan Processing telah ditampilkan di Museum Seni Modern di New York, Museum Victoria dan Albert di London, Centre Georges Pompidou di Paris, dan banyak tempat terkenal lainnya. Processing digunakan untuk membuat desain panggung yang diproyeksikan untuk pertunjukan tari dan musik; untuk menghasilkan gambar untuk video musik dan film; untuk mengekspor gambar untuk poster, majalah, dan buku; dan untuk membuat instalasi interaktif di galeri, di museum, dan di jalan. Beberapa proyek terkemuka termasuk video House of Cards untuk Radiohead, logo generatif MIT Media Lab, dan mural perangkat lunak yang diproyeksikan Chronograph untuk New World Center yang dirancang oleh Frank Gehry di Miami. Tetapi hal yang paling penting tentang Processing dan budaya bukanlah hasil profil tinggi - ini adalah bagaimana perangkat lunak telah melibatkan generasi baru seniman visual untuk mempertimbangkan pemrograman sebagai bagian penting dari praktik kreatif mereka.

Penelitian

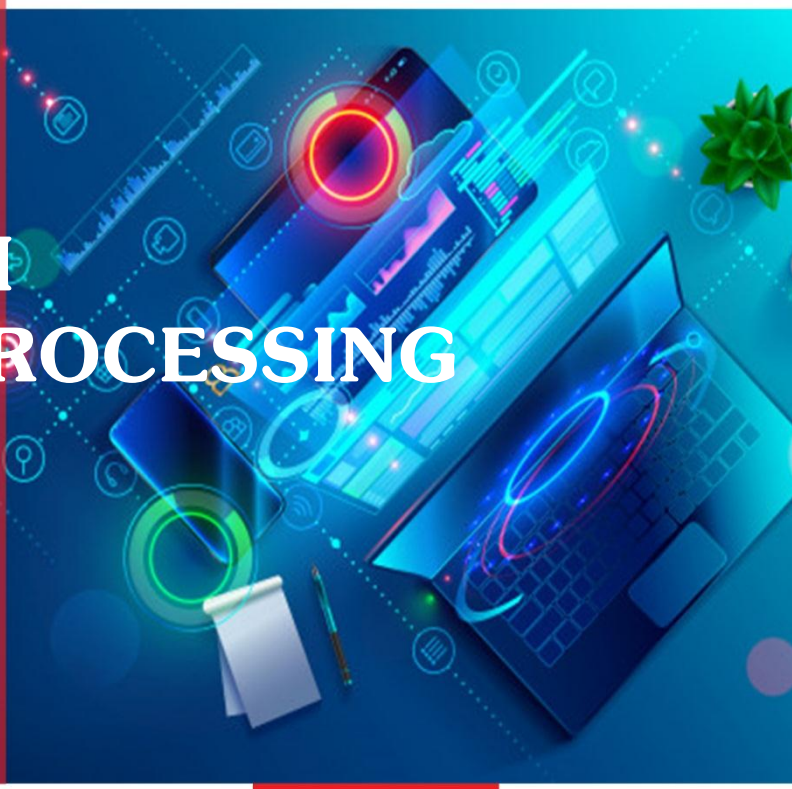
Prototipe perangkat lunak dan visualisasi data adalah dua bidang terpenting bagi pengembang Processing. Laboratorium penelitian di dalam perusahaan teknologi seperti Google dan Intel telah menggunakan Processing untuk membuat prototipe antarmuka dan layanan baru. Perusahaan termasuk General Electric, Nokia, dan Yahoo! telah menggunakan Processing untuk memvisualisasikan data internal mereka. Misalnya, Laboratorium R&D Perusahaan New York Times menggunakan Processing untuk memvisualisasikan cara perjalanan berita mereka melalui media sosial. NSF dan NOAA mendukung penelitian yang mengeksplorasi keanekaragaman fitoplankton dan zooplankton yang diwujudkan di University of Washington sebagai simulasi ekologi yang dinamis. Para peneliti di Texas Advanced Computer Center di UT Austin telah menggunakan Processing untuk menampilkan visualisasi data besar di seluruh kisi layar dalam layanan penelitian humaniora.

Sejarah

Processing dimulai oleh Ben Fry dan Casey Reas pada musim semi 2001, sementara keduanya adalah mahasiswa pascasarjana di MIT Media Lab dalam kelompok riset Estetika dan Komputasi John Maeda. Pengembangan berlanjut di waktu luang mereka sementara Casey mengejar karir seni dan mengajar dan Ben mengejar gelar Ph.D. dan mendirikan Desain Informasi Fathom. Banyak ide dalam Processing kembali ke Lokakarya Bahasa Visual Muriel Cooper, dan itu tumbuh langsung dari proyek Design By Numbers Maeda, yang dikembangkan di Media Lab dan dirilis pada 1999. Proyek Pengkabelan dan Arduino, pada gilirannya, tumbuh dari

Processing sementara Casey mengajar di Institut Desain Interaksi Ivrea di Italia.

BAB II
IDE PROCESSING



Pendahuluan

Processing Development Environment (PDE) membuatnya mudah untuk menulis program Processing. Program ditulis dalam Editor Teks dan dimulai dengan menekan tombol Run. Processing adalah sebuat program komputer disebut sketsa (sketch). Sketsa disimpan di Sketchbook, yang merupakan folder di komputer.

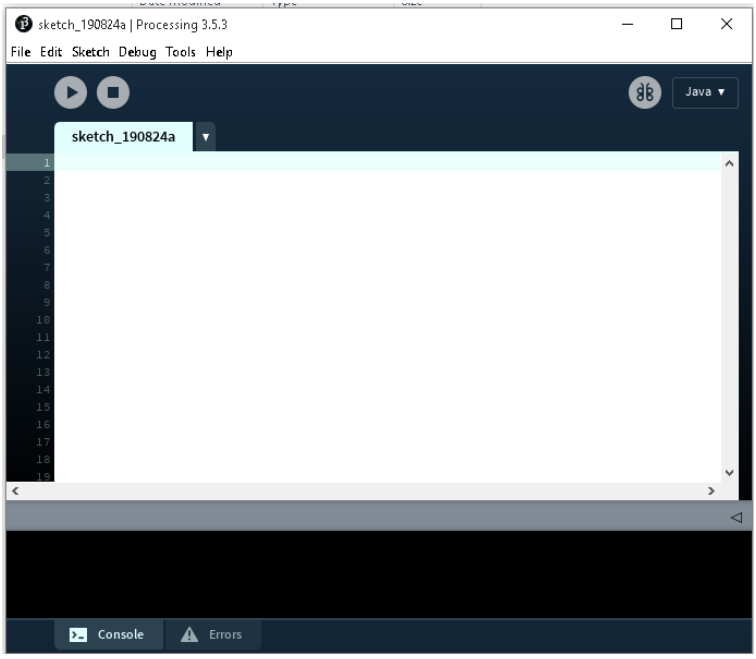
Sketsa dapat menggambar grafik dua dan tiga dimensi. Renderer default adalah untuk menggambar grafik dua dimensi. P3D renderer memungkinkan untuk menggambar grafik tiga dimensi, yang meliputi mengendalikan kamera, pencahayaan, dan material. Renderer P2D adalah renderer yang cepat, tetapi kurang akurat untuk menggambar grafik dua dimensi. Kedua penyaji P2D dan P3D dipercepat jika komputer memiliki kartu grafis yang kompatibel dengan OpenGL.

Kemampuan Processing diperluas dengan Library dan Tools. Library memungkinkan sketsa untuk melakukan hal-hal di luar kode Processing inti. Ada ratusan library yang dikontribusikan oleh komunitas Pemrosesan yang dapat ditambahkan ke sketsa untuk mengaktifkan hal-hal baru seperti memutar suara, melakukan penglihatan komputer, dan bekerja dengan geometri 3D canggih. Tools memperluas PDE untuk membantu membuat membuat sketsa lebih mudah dengan menyediakan antarmuka untuk tugas-tugas seperti memilih warna.

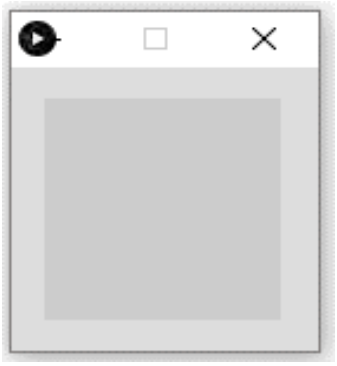
Processing memiliki mode pemrograman yang berbeda untuk memungkinkan pengerahan sketsa pada platform dan program yang berbeda dengan cara yang berbeda. Mode Java adalah default. Mode pemrograman lain dapat diunduh dengan memilih " Add Mode... " dari menu di sudut kanan atas PDE.

Processing Development Environment (PDE)

Processing Development Environment (PDE) terdiri dari editor teks sederhana untuk menulis kode, area pesan, konsol teks, tab untuk mengelola file, toolsbar dengan tombol untuk tindakan bersama, dan serangkaian menu. Opsi menu berubah dari mode ke mode. Mode default adalah Java Mode jadi jika belum ada Java Runtime Enviroment maka harus download dahulu.



Gambar 1. Processing Development Environment



Gambar 2. New Display Window

Program yang ditulis menggunakan Processing disebut sketsa. Sketsa ini ditulis dalam editor teks. Ini memiliki fitur untuk memotong / menempel dan untuk mencari / mengganti teks. Area pesan memberikan umpan balik saat menyimpan dan mengekspor dan juga menampilkan kesalahan. Konsol menampilkan output teks dengan Memproses sketsa termasuk pesan kesalahan lengkap dan output teks dari sketsa dengan fungsi print () dan println (). (Perhatikan bahwa konsol berfungsi dengan baik untuk pesan sesekali, tetapi tidak dimaksudkan untuk output real-time berkecepatan tinggi.)
Tombol-tombol pada toolbars yang dapat menjalankan dan menghentikan program:

Run : Jalankan sketsa. Dalam mode Java, ia mengkompilasi kode dan membuka jendela tampilan baru.

Stop : Menghentikan sketsa program yang sedang jalan.

Perintah tambahan ditemukan dalam enam menu: File, Edit, Sketch, Debug, Tools, Help. Menu sensitif konteks yang

berarti hanya item yang relevan dengan pekerjaan yang sedang dilakukan yang tersedia.

File

- **New**
  Membuat sketsa baru di window baru, dinamai sebagai tanggal saat ini dengan format "sketch_YYMMDDa".
- **Open...**
  Buka sketsa di window baru.
- **Open Recent**
  Pilih sketsa untuk dibuka dari daftar sketsa yang baru saja ditutup.
- **Sketchbook...**
  Buka window baru untuk menampilkan daftar sketsa di sketchbook.
- **Examples...**
  Buka window baru untuk menampilkan daftar examples.
- **Close**
  Tutup sketsa di window paling depan. Jika ini adalah sketsa terakhir yang terbuka, akan ditanya apakah ingin berhenti. Untuk menghindari konfirmasi, gunakan Quit dari perintah Close saat ingin keluar dari aplikasi.
- **Save**
  Menyimpan sketsa terbuka di kondisi saat ini.
- **Save as...**
  Menyimpan sketsa yang sedang terbuka, dengan opsi memberikan nama yang berbeda. Tidak mengganti sketsa versi sebelumnya.
- **Export**
  Mengekspor aplikasi Java sebagai file yang dapat dieksekusi dan membuka folder yang berisi file yang diekspor.
- **Page Setup**
  Tentukan pengaturan halaman untuk dicetak.
- **Print (Ctrl+P)**

Mencetak kode di dalam editor teks.

- **Preferences**
  Cara mengubah beberapa kerja Processing.
- **Quit**
  Keluar dari PDE dan tutup semua window Processing.

Edit

- **Undo**
  Mengembalikan perintah terakhir atau entri terakhir yang diketik. Cancel perintah Batalkan dengan memilih Edit »Redo.
- **Redo**
  Membalikkan tindakan perintah Undo terakhir. Opsi ini hanya tersedia jika sudah ada tindakan Undo.
- **Cut**
  Menghapus dan menyalin teks yang dipilih ke clipboard (buffer teks di luar layar).
- **Copy**
  Menyalin teks yang dipilih ke clipboard.
- **Copy as HTML**
  Memformat kode sebagai HTML dengan cara yang sama muncul di lingkungan Processing dan menyalinnya ke clipboard sehingga dapat ditempelkan di tempat lain.
- **Paste**
  Menyisipkan konten clipboard di lokasi kursor, dan mengganti teks yang dipilih.
- **Select All**
  Memilih semua teks dalam file yang saat ini terbuka di editor teks.
- **Auto Format**
  Mencoba memformat kode menjadi tata letak yang lebih dapat dibaca oleh manusia. Format Otomatis sebelumnya disebut Mempercantik.

- **Comment/Uncomment**
  Komentar teks yang dipilih. Jika teks yang dipilih sudah dikomentari, itu menghapus komentar itu.
- **Increase Indent**
  Indentasi teks yang dipilih dua spasi.
- **Decrease Indent (Ctrl+[)**
  Jika teks berlekuk, hapus dua spasi dari indentasi.
- **Find...**
  Menemukan kemunculan string teks di dalam file yang terbuka di editor teks dan memberikan opsi untuk menggantinya dengan teks yang berbeda.
- **Find Next**
  Menemukan kemunculan string teks berikutnya di dalam file yang terbuka di editor teks.
- **Find Previous**
  Menemukan kemunculan string teks sebelumnya di dalam file yang terbuka di editor teks.
- **Use Selection for Find**
  Set teks yang saat ini dipilih sebagai item untuk ditemukan dengan Find Next dan Find Previous.

Sketch

- **Run**
  Menjalankan kode (mengkompilasi kode, membuka window tampilan, dan menjalankan sketsa di dalamnya)
- **Present**
  Menjalankan kode di tengah layar dengan latar belakang warna solid. Klik tombol "stop" di kiri bawah untuk keluar dari presentasi atau tekan tombol Escape. Ubah warna latar belakang dalam Preferensi.
- **Tweak**
  Menjalankan kode dengan cara di mana beberapa warna dan nilai variabel dapat diubah saat kode berjalan. Sketsa

harus disimpan sebelum dapat dijalankan sebagai sketsa ke Tweak.

- **Stop**
  Jika kode sedang berjalan, hentikan eksekusi. Program yang ditulis tanpa menggunakan fungsi draw() dihentikan secara otomatis setelah mereka menggambar.
- **Import Library**
  Menambahkan pernyataan impor yang diperlukan ke bagian atas sketsa saat ini. Misalnya, memilih Sketch »Impor Library» pdf menambahkan pernyataan "import processing.pdf. *;" ke bagian atas file. Pernyataan impor ini diperlukan untuk menggunakan Library. Pilih Add Library ... untuk membuka Manajer Library untuk menelusuri dan menginstal library baru.
- **Show Sketch Folder**
  Buka folder untuk sketsa saat ini.
- **Add File...**
  Membuka window navigator file. Pilih gambar, font, atau file media lainnya untuk menambahkannya ke folder "data" sketsa.

Debug

- **Enable Debugger**
  Mengaktifkan debugger. Perhatikan bahwa tombol Jalankan akan berubah menjadi Debug. Tombol Lanjutkan dan Langkah baru akan muncul, bersama dengan jendela terpisah untuk melihat nilai variabel.
- **Continue**
  Tingkatkan kode hingga breakpoint berikutnya.
- **Step**
  Memajukan kode satu baris setiap kali. (Perhatikan bahwa begitu kode mencapai akhir panggilan fungsi saat ini, debugger akan kembali ke " continue. ")
- **Step Into**

Memajukan debugger ke bagian dalam panggilan fungsi. Ini hanya berfungsi untuk fungsi yang ditentukan pengguna dalam sketsa.

- **Step Out**
  Memajukan debugger di luar fungsi ke area panggilan. Ini hanya berfungsi untuk fungsi yang ditentukan pengguna dalam sketsa.
- **Toggle Breakpoint**
  Tambah atau hapus breakpoint. Ketika breakpoint ditambahkan, nomor baris diganti dengan simbol: <>.

Tools

- **Create Font...**
  Ubah font menjadi format Processing font (VLW) dan tambahkan ke sketsa saat ini. Membuka kotak dialog yang memberikan opsi untuk mengatur font, ukurannya, jika anti-alias (halus), dan karakter mana yang akan dihasilkan. Jumlah memori yang diperlukan untuk font ditentukan oleh ukuran yang dipilih dan jumlah karakter yang dipilih melalui menu " Characters ..."; Memproses font adalah tekstur, jadi font yang lebih besar membutuhkan lebih banyak data gambar. Font juga dapat dibuat dalam kode dengan fungsi createFont ().
- **Color Selector...**
  Antarmuka untuk memilih warna. Untuk setiap warna, nilai HSB, RBG, dan Hex ditampilkan. Nilai Hex dapat disalin ke clipboard dengan tombol Salin.
- **Archive Sketch**
  Mengarsipkan salinan sketsa saat ini dalam format .zip. Arsip diletakkan di folder yang sama dengan sketsa.
- **Install "processing-java"**
  Instal program processing-java untuk memungkinkan untuk membangun dan menjalankan sketsa mode Java dari baris perintah.

- **Movie Maker**
  Membuat film QuickTime dari urutan gambar. Opsi mencakup pengaturan ukuran, kecepatan bingkai, dan kompresi, serta file audio.
- **Add Tool...**
  Buka Pengelola Tool untuk menelusuri dan menginstal Tools baru.

Help

- **Environment**
  Membuka referensi untuk Processing Development Environment (halaman ini) di browser web default.
- **Reference**
  Membuka referensi di browser web default. Termasuk referensi untuk bahasa, lingkungan pemrograman, dan library inti.
- **Find in Reference**
  Pilih elemen bahasa Processing dalam editor teks dan pilih Find in Reference untuk membuka halaman itu di browser web default.
- **Libraries Reference**
  Pilih dari daftar untuk membuka referensi untuk Library yang kompatibel.
- **Tools Reference**
  Pilih dari daftar untuk membuka referensi untuk Tools yang dapat kompatibel.
- **Getting Started**
  Membuka tutorial Memulai online di browser default.
- **Troubleshooting**
  Membuka halaman wiki pemecahan masalah online di browser default.
- **Frequently Asked Questions**
  Buka halaman wiki FAQ online di browser default.

- **The Processing Foundation**
  Membuka situs web Foundation di browser default.
- **Visit Processing.org**
  Opens Processing website in the default browser.
- **Preferences**
  Processing Development Environment (PDE) sangat dapat dikonfigurasi. Preferensi yang paling umum dapat dimodifikasi di jendela Preferensi, yang terletak di menu File di Windows dan Linux dan di menu Processing pada Mac Os X. Daftar lengkap preferensi disimpan dalam file "preferences.txt". File ini dapat dibuka dan diedit secara langsung hanya ketika Processing tidak berjalan. Anda dapat menemukan lokasi file ini di komputer Anda dengan membaca sudut kiri bawah window Preferensi.
- **Sketchbook location**
  Folder apa pun dapat digunakan sebagai Buku Sketsa. Masukkan lokasi baru atau pilih "Browse" untuk mengatur folder yang ingin Anda gunakan.
- **Language**
  Pilih bahasa yang akan digunakan untuk menu. Processing perlu dimulai kembali setelah membuat pilihan baru.
- **Editor and Console font**
  Pilih font lain untuk digunakan untuk teks di Editor dan Konsol. Catatan: font yang dipilih harus sesuai dengan bahasa yang digunakan dalam Editor Teks. Lihat preferensi " Enable complex text input " di bawah ini.
- **Editor font size**
  Atur ukuran font kode dalam editor teks.
- **Console font size**
  Atur ukuran font teks di konsol.
- **Background color when Presenting**
  Menentukan warna latar belakang yang digunakan saat sketsa dijalankan dengan Present.

- **Use smooth text in editor window**
  Secara default, teks dalam editor adalah alias. Saat dicentang, editor beralih ke font anti-alias (dihaluskan). Mulai ulang Processing setelah melakukan perubahan ini.
- **Enable complex text input**
  Mengaktifkan Editor Teks untuk menampilkan font non-Latin seperti Jepang. Processing perlu dimulai kembali setelah membuat pilihan ini.
- **Continuously check for errors and Show warnings**
  Nyalakan dan matikan fitur yang terus-menerus memeriksa dan melaporkan potensi kesalahan kode.
- **Code completion with Ctrl-space**
  Nyalakan dan matikan penyelesaian kode. Tekan Ctrl-spasi untuk mengaktifkan penyelesaian kode saat mengetik.
- **Suggest import statements**
  Saat dicentang, Processing akan mencoba menyarankan pustaka untuk mengimpor ketika kode dari pustaka itu terdeteksi.
- **Increase maximum available memory**
  Alokasikan lebih banyak RAM untuk sketsa Processing saat dijalankan. Sketsa yang menggunakan file media (gambar, audio, dll.) Terkadang membutuhkan lebih banyak RAM. Tambah jumlah RAM jika sketsa membuang Kesalahan Memori.
- **Delete previous folder on export**
  Saat dicentang (perilaku default), Processing menghapus folder ekspor lengkap sebelum membuatnya kembali dan menambahkan media baru.
- **Check for updates on startup**
  Ketika dicentang (perilaku default), Anda akan diberi tahu tentang Processing peranti lunak yang baru dirilis saat tersedia melalui kotak dialog kecil yang terbuka saat Processing dimulai.

- **Run sketches on display**
  Jika lebih dari satu monitor terpasang, pilih monitor tempat sketsa ditampilkan.
- **Sketches dan Sketchbook**
  Semua proyek Processing disebut sketsa. Setiap sketsa memiliki foldernya sendiri. File utama untuk setiap sketsa memiliki nama yang sama dengan folder dan ditemukan di dalamnya. Misalnya, jika sketsa tersebut dinamai "Sketch_123", folder untuk sketsa tersebut akan disebut "Sketch_123" dan file utama akan disebut "Sketch_123.pde". Ekstensi file PDE adalah singkatan dari Processing Development Environment.
  Sketsa Processing dapat disimpan di mana saja di komputer, tetapi secara default sketsa disimpan di sketsa, yang akan berada di tempat yang berbeda di komputer atau jaringan tergantung apakah menggunakan PC, Mac, atau Linux dan bagaimana preferensi diatur. Untuk menemukan folder ini, pilih opsi "Preference" dari menu File (atau dari menu " Processing " pada Mac) dan cari "Lokasi sketsa."
  Folder sketsa terkadang berisi folder lain untuk file media dan kode lainnya. Ketika font atau gambar ditambahkan ke sketsa dengan memilih "Add File ..." dari menu Sketsa, folder "data" dibuat. File juga dapat ditambahkan ke sketsa pemrosesan Anda dengan menyeretnya ke editor teks. File gambar dan suara yang diseret ke dalam jendela aplikasi akan secara otomatis ditambahkan ke folder "data" sketsa saat ini. Semua gambar, font, suara, dan file data lainnya yang dimuat dalam sketsa harus ada di folder ini.

Renderers

Processing memiliki empat penyaji layar bawaan. Renderer default adalah untuk menggambar bentuk dua dimensi. P2D adalah renderer yang lebih cepat, tetapi kurang akurat untuk menggambar bentuk dua dimensi. P3D adalah untuk geometri tiga dimensi; itu juga dapat mengontrol kamera, pencahayaan, dan material. Penyaji P2D dan P3D dipercepat jika komputer Anda memiliki kartu grafis yang kompatibel dengan OpenGL. Fungsi smooth () memengaruhi jumlah antialiasing untuk setiap penyaji. Periksa referensi untuk smooth () untuk informasi lebih lanjut.

Dengan rilis Processing 3.0, renderer FX2D disertakan. Gunakan untuk grafis 2D cepat pada tampilan resolusi besar dan tinggi untuk kecepatan lebih dari penyaji standar. Penyaji ini masih eksperimental, tetapi bermanfaat untuk kondisi tertentu.

Penyaji yang digunakan untuk setiap sketsa ditentukan melalui fungsi size (). Jika penyaji tidak didefinisikan secara eksplisit dalam size (), ia menggunakan penyaji default seperti yang ditunjukkan dalam program berikut:

```
void setup() {
  size(300, 300);
}

void draw() {
  background(210);
  line(width/2, height/2, mouseX, mouseY);
}
```

Untuk mengubah penyaji, tambahkan parameter ketiga ke size (). Sebagai contoh:

```
void setup() {
```

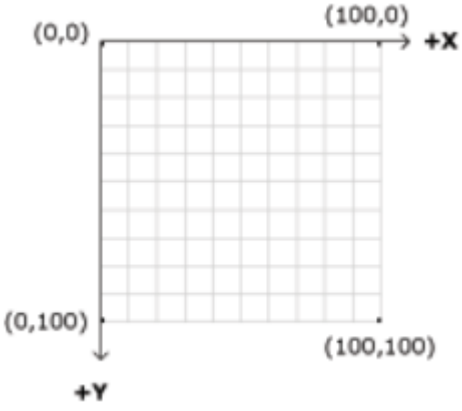
```
  size(300, 300, P2D);
}

void draw() {
  background(210);
  line(width/2, height/2, mouseX, mouseY);
}
```

Upaya besar telah dilakukan untuk membuat kode Processing berperilaku serupa di seluruh penyaji yang berbeda, tetapi saat ini ada beberapa inkonsistensi yang dijelaskan dalam referensi.

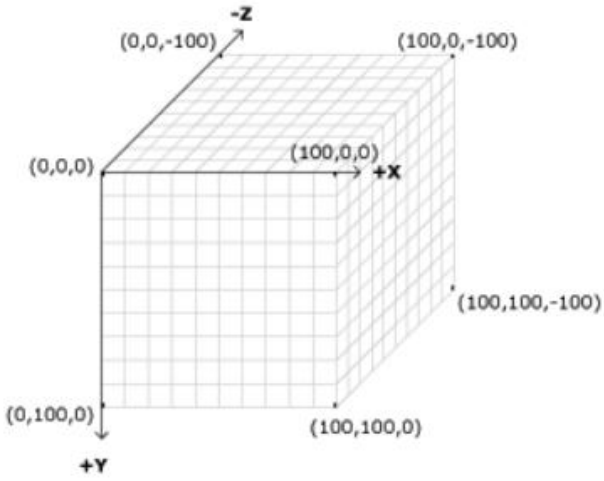
Untuk informasi lebih lanjut, lihat entri referensi size ().

Coordinat

Processing menggunakan sistem koordinat Cartesian dengan koordinat awal di sudut kiri atas. Jika sketsa memiliki lebar 320 pixel dan tinggi 240 pixel, koordinat (0, 0) adalah pixel kiri atas dan koordinat (320, 240) ada di kanan bawah. Pixel yang terlihat terakhir di sudut kanan bawah layar berada pada posisi (319, 239) karena pixel digambar ke kanan dan di bawah koordinat.



Gambar 3. Koordinat 2 Deminsi



Gambar 4. Koordinat 3 Dimensi

Dengan menggunakan sistem koordinat tiga dimensi P3D, koordinat z adalah nol pada permukaan gambar, dengan nilai-z negatif bergerak mundur dalam ruang. Saat menggambar dalam 3D, kamera diposisikan di tengah layar.

Tabs, Multiple Files, dan Classes

Mungkin merepotkan untuk menulis program panjang dalam satu file. Saat Mengolah sketsa tumbuh hingga ratusan atau ribuan garis, memecahnya menjadi unit modular membantu mengelola bagian yang berbeda. Memproses mengelola file dengan Sketchbook dan setiap sketsa dapat memiliki banyak file yang dikelola dengan tab.

Tombol panah di sebelah kanan tab di Processing Development Enviroment digunakan untuk mengelola file-file ini. Klik tombol ini untuk membuka opsi untuk membuat tab baru, mengganti nama tab saat ini, dan menghapus tab saat ini. Tab ditujukan untuk pengguna yang lebih mahir, dan karena alasan ini, menu yang mengontrol tab sengaja dibuat kurang menonjol.

Keuntungan

Ketika sebuah program dengan banyak tab dijalankan, kode dikelompokkan bersama dan kelas-kelas di tab lain menjadi inner classes. Karena inner classes, mereka tidak dapat memiliki variabel statis. Cukup tempatkan variabel "static" di luar kelas itu sendiri untuk melakukan hal yang sama (tidak perlu secara eksplisit dinamai "static" setelah mendaftar dengan cara ini). Jika tidak ingin kode menjadi inner classes, bisa membuat tab dengan akhiran ".java", yang berarti kode itu akan ditafsirkan sebagai kode java langsung. Juga tidak mungkin menggunakan kelas statis di tab terpisah. Namun, jika melakukan ini, harus meneruskan objek PApplet ke objek itu di tab itu untuk mendapatkan fungsi PApplet seperti line (), loadStrings () atau saveFrame () untuk bekerja.

Debug

Debugger Processing adalah alat untuk mendiagnosis masalah dengan sketsa. Aktifkan untuk menghentikan sketsa saat menjalankan dan memajukan kode satu baris setiap kali. Debugger diaktifkan melalui menu File (Debug> Enable Debugger) atau dengan mengklik ikon Debugger, kupu-kupu di sudut kanan atas PDE.

Ketika Debugger diaktifkan, program berjalan seperti biasa, tetapi berhenti di "breakpoints." Untuk membuat breakpoint, atur kursor pada baris yang ingin jeda sketsa dan pilih Debug> Toggle Breakpoint. Pintasan keyboard adalah Command-B. Untuk menghapus breakpoint, pilih Toggle Breakpoint lagi. Ketika breakpoint ditambahkan, nomor baris diganti dengan simbol: <>.

Menjalankan sketsa dalam mode Debug menyebabkan sketsa berhenti di setiap breakpoints. Saat dijeda, nilai variabel

saat ini terlihat di panel terpisah. Anda dapat maju ke breakpoint berikutnya dengan memilih " Continue " atau memajukan baris demi baris melalui kode dengan "Step". Melangkah hanya berfungsi dalam lingkup fungsi saat ini sedang dijalankan.

Mode Programming

Processing memiliki mode pemrograman yang berbeda untuk memungkinkan pengerahan sketsa pada platform dan program yang berbeda dengan cara yang berbeda. Mode pemrograman default saat ini adalah mode Java. Mode pemrograman lain seperti Android Mode dan Python ditambahkan dengan memilih "Add Mode ..." dari menu di sudut kanan atas PDE.

Java Mode

Mode ini memungkinkan untuk menulis program pendek untuk menarik ke layar, tetapi juga memungkinkan program Java yang kompleks juga. Ini dapat digunakan hanya oleh pemula, tetapi skala untuk pengembangan perangkat lunak Java profesional. Sketsa yang ditulis dalam mode ini dapat diekspor sebagai Aplikasi Java untuk dijalankan di Linux, Mac OS X, dan sistem operasi Windows.

Keuntungan

File Java dengan ekstensi .java dapat dimasukkan dengan sketsa mode Java. Mereka dapat dibuat langsung di PDE atau disalin ke folder sketsa melalui item "Add File ..." di menu Sketsa atau diseret ke editor teks. Dimungkinkan untuk menulis

kode Java apa pun dalam file dengan ekstensi .java. Bahkan, kode Java lengkap dapat ditulis dari dalam Lingkungan Processing dengan mensubklasifikasikan PApplet seperti ini:

public class MyDemo extends PApplet {

Ini hanya untuk pengembang tingkat lanjut dan tidak benar-benar direkomendasikan. Menggunakan teknik ini berarti bahwa setiap tab tambahan tidak akan lagi menjadi kelas dalam, artinya Anda harus melakukan pekerjaan ekstra untuk membuatnya berkomunikasi dengan benar dengan host **PApplet**. Tidak perlu menggunakan teknik ini hanya untuk mendapatkan fitur dari bahasa Java. Pengembang tingkat lanjut juga dapat memprogram dengan Processing dalam Java Editor lain jika diperlukan pengeditan dan alat kode tingkat tinggi. Core.jar pemrosesan dapat digunakan sebagai bagian dari proyek Java apa pun.

Penambahan Library, Tools, dan Mode

Processing 3.0 mencakup serangkaian fitur untuk membuatnya lebih mudah untuk menginstal, memperbarui, dan menghapus Libraries, Tools, Mode, dan Contoh.

Tambahkan pustaka kontribusi dengan memilih "Add Library ..." dari submenu "Impor Library ..." di dalam menu Sketsa. Ini membuka Manajer Library. Selanjutnya, pilih Library dan kemudian klik Instal untuk mengunduhnya.

Tambahkan alat bantu dengan memilih "Add Tools ..." dari menu Tools, lalu pilih Tools untuk diunduh dari Pengelola Tools.

Tambahkan mode kontribusi dengan memilih "Add Mode ..." dari menu Mode di sudut kanan atas PDE, lalu pilih Mode untuk menginstal.

Export

Halaman Expor informasi dan Tips pada Wiki Processing mencakup rincian mengekspor Aplikasi dari mode Java.

BAB III
STRUKTUR

() (parentheses)

Examples

```
int b;
b = (14 + 6) * 3;       // Pengelompokan ekspresi
if (b > 50) {           // berisi ekspresi
  line(b, 0, b, 200);  // Berisi daftar parameter
}
```

Deskripsi

Pengelompokan dan berisi ekspresi dan parameter. Kurung memiliki beberapa fungsi yang berkaitan dengan fungsi dan struktur. Mereka digunakan untuk berisi daftar parameter yang dilewatkan ke fungsi dan struktur kontrol dan mereka digunakan untuk ekspresi grup untuk mengontrol urutan eksekusi. Beberapa fungsi tidak memiliki parameter dan dalam hal ini, ruang antara tanda kurung kosong.

Sintak

function()
function(p1, ..., pN)
structure(expression)

Parameter-parameter

Fungsi beberapa fungsi

p1, ..., pN	daftar parameter khusus untuk fungsi
structure	Kontrol struktur seperti if, for, while
expressions	setiap ekspresi atau grup ekspresi yang valid

, (comma)

Examples

// Koma digunakan untuk memisahkan daftar deklarasi variabel
int x=20, y=30, z=80;

// Koma digunakan untuk memisahkan daftar nilai yang ditetapkan ke array
int[]w = { 20, 60, 80 };

// Koma digunakan untuk memisahkan daftar parameter yang dilewatkan ke suatu fungsi
line(x, y, z, y);
line(w[0], w[1], w[2], w[1]);

Diskripsi

Memisahkan parameter dalam panggilan fungsi dan elemen selama penugasan.

Sintak

value1, ..., valueN

Parameter-parameter

value1, ..., valueN pemisahan paramater int, float, byte, boolean, color, char, String.

. (dot)

Examples

// Deklarasikan dan bangun dua objek (h1 dan h2) dari kelas HLine
HLine h1 = new HLine(30, 1.0);

```
HLine h2 = new HLine(60, 5.0);

void setup() {
  size(300, 300);
}

void draw() {
  if (h2.speed > 1.0)
{  //Sintaks dot dapat digunakan untuk mendapatkan nilai
    h2.speed -= 0.01;    // atau atur nilainya.
  }
  h1.update();  //panggil objek h1  fungsiupdate()
  h2.update();  //panggil objek h2  fungsiupdate()
}

class HLine {  //difinisi Class
  float ypos, speed;  // Data
  HLine (float y, float s) {  //konstruksi Objek
    ypos = y;
    speed = s;
  }

  void update() {  //metode Update
    ypos += speed;
    if (ypos > width) {
      ypos = 0;
    }
    line(0, ypos, width, ypos);
  }
}
```

Deskripsi

Memberikan akses ke metode dan data objek. Objek adalah satu instance dari kelas dan dapat berisi metode (fungsi objek)

dan data (variabel objek dan konstanta), seperti yang ditentukan dalam definisi kelas. Operator dot mengarahkan program ke informasi yang dikemas dalam suatu objek.

Sintak
object.method()
object.data

Parameter-parameter
object objek yang akan diakses
method() metode yang dienkapsulasi dalam objek
data variabel atau konstanta yang dienkapsulasi dalam objek.

/* */ (multiline comment)

Examples
```
/*
Menarik dua garis yang membagi window menjadi empat kuadran. Pertama menggambar horizontal garis dan kemudian garis vertikal.
*/
line(0, 60, 120, 60);
line(60, 0, 60, 120);
```

Deskripsi
Catatan penjelasan yang tertanam dalam kode. Komentar digunakan untuk mengingatkan diri sendiri dan untuk memberi tahu orang lain tentang fungsi program Anda. Komentar multiline digunakan untuk deskripsi teks besar kode atau untuk mengomentari potongan kode saat debug aplikasi. Komentar diabaikan oleh kompiler.

Sintak

```
/ *
  komentar
* /
```

Parameter

komentar setiap urutan karakter

/** */ (doc comment)

Examples

```
/ **
  Menarik dua garis yang membagi jendela
  menjadi empat kuadran. Pertama menggambar horizontal
  garis dan kemudian garis vertikal
* /
line (0, 50, 100, 50);
line (50, 0, 50, 100);
```

Diskripsi

Catatan penjelasan tertanam dalam kode dan ditulis ke file "index.html" yang dibuat ketika kode diekspor. Doc Comments (komentar dokumentasi) digunakan untuk berbagi deskripsi sketsa Anda ketika program diekspor. Ekspor kode dengan menekan tombol "Ekspor" pada Toolbar.

Sintak

```
/**
 komen
*/
```


Parameter-parameter
Komentar untuk karakter apapun

// comment

Examples
// Menggambar dua garis membagi jendela
// Menjadi empat kuadran
line (0, 50, 200, 150); // gambar garis horizontal
line (50, 0, 250, 150); // gambar garis vertikal

Diskripsi
Catatan penjelasan tertanam dalam kode. Komentar digunakan untuk mengingatkan diri sendiri dan untuk memberi tahu orang lain tentang detail kode. Komentar baris tunggal ditandai dengan dua karakter garis miring. Komentar diabaikan oleh kompiler.
Sintak
// komentar

Parameter-parameter
Komentar urutan karakter apa pun

; (semicolon)

Examples
int b; // Pernyataan deklarasi
b = 30; // Pernyataan penugasan
println (b); // Pernyataan fungsi

Diskripsi
Terminator pernyataan yang memisahkan elemen program. Pernyataan adalah instruksi lengkap ke komputer dan titik koma digunakan untuk memisahkan instruksi (ini mirip dengan periode "." Dalam bahasa Inggris tertulis). Titik koma juga digunakan untuk memisahkan berbagai elemen struktur for .

Sintak
statement;

Parameter-parameter
Statement satu pernyataan untuk di eksekusi

= (assign)

Examples
int c;
c = 30; // Tetapkan nilai 30 ke variabel 'c'
c = c + 40; // Tetapkan nilai 70 ke variabel 'c'

Diskripsi
Menetapkan nilai ke variabel. Tanda "=" tidak berarti "sama dengan", tetapi digunakan untuk menempatkan data dalam suatu variabel. Operator "=" secara resmi disebut operator penugasan. Ada banyak jenis variabel (int, float, string, dll.) Dan operator penugasan hanya dapat menetapkan nilai yang jenisnya sama dengan variabel yang ditetapkannya. Misalnya, jika variabel bertipe int , nilainya juga harus berupa int .

Sintak
Var=value

Paramaters
Var Nama variabel yang valid
Value Nilai apapun dari jenis yang sama dengan variabel. Misalnya, jika variabel bertipe "int", nilainya juga harus berupa int.

[] (array access)

Examples
```
int [] angka = int baru [3];
angka [0] = 90;
angka [1] = 150;
angka [2] = 30;
int x = angka [0] + angka [1]; // Tetapkan variabel 'x' ke 240
int y = angka [1] + angka [2]; // Tetapkan variabel 'y' ke 180
```

Diskripsi
Operator akses array digunakan untuk menentukan lokasi dalam array. Data di lokasi ini dapat didefinisikan dengan Sintak [element]=value dan dibaca dengan Sintak value=array [element] seperti ditunjukan dalam contoh di atas.

Sintak
```
datatype[]
array[element]
```

Parameter-parameter
Tipe data setiap tipe data primitif atau majemuk, termasuk kelas yang ditentukan pengguna
Array Nama variabel apapun yang valid

Element int: tidak boleh melebihi panjang array minus 1

{} (CURLY BRACES)

Examples

```
int [] m = {5, 20, 25, 45, 70};
void setup () {
  size (100, 100);
}
void draw () {
  for (int i = 0; i <m.length; i ++) {
    line (0, m [i], 50, m [i]);
  }
}
```

Diskripsi

Tentukan blok fungsi awal dan akhir dan blok pernyataan seperti struktur for and if . Kurung kurawal juga digunakan untuk menentukan nilai awal dalam deklarasi array.

Sintak

{ statements }
{ ele0, ..., eleN }

Parameter-parameter

Element	setiap urutan pernyataan yang valid
ele0, ..., eleN	daftar elemen yang dipisahkan oleh koma

CATCH

Examples

```
BufferedReader Pembaca;
String line;
```


```
Void setup  () {
  // Buka file dari contoh createWriter ()
  Pembaca = createReader ("posisi.txt");
}
void draw () {
  try {
    line = reader.readLine ();
  } catch (IOException e) {
    e.printStackTrace ();
    line = null;
  }
  if (line == null) {
    // Berhenti membaca karena kesalahan atau file kosong
    noLoop ();
  } else {
    String[] pieces = split(line, TAB);
    int x = int(pieces[0]);
    int y = int(pieces[1]);
    point(x, y);
  }
}
```

Diskripsi

Kata kunci catch digunakan dengan mencoba menangani pengecualian. Dokumentasi Sun Java mendefinisikan pengecualian sebagai "suatu peristiwa, yang terjadi selama pelaksanaan suatu program, yang mengganggu aliran normal instruksi program." Ini bisa berupa, misalnya, kesalahan saat file dibaca.

Sintak

```
try {
  tryStatements
} catch (exception) {
  catchStatements
```

```
}
```

Parameter-parameter

tryStatements jika kode ini memunculkan eksepsi, maka kode dalam "catch" dijalankan.

catch (exception) pengecualian Java yang dilempar.

catchStatements kode yang menangani pengecualian.

CLASS

Examples

```
// Deklarasikan dan buat dua objek (h1, h2) dari kelas HLine
HLine h1 = new HLine (25, 2.0);
HLine h2 = new HLine (55, 2.5);

Void setup ()
{
  size (300, 300);
  frameRate (30);
}
void draw () {
  background (204);
  h1.update ();
  h2.update ();
}
class HLine {
  float ypos, speed;
  HLine (float y, float s) {
    ypos = y;
    speed = s;
  }
```


```
 Void update () {
   ypos + = speed;
   if (ypos> height) {
     ypos = 0;
   }
   line (0, ypos, width, ypos);
 }
}
```

Diskripsi

Kata kunci yang digunakan untuk menunjukkan deklarasi Class. Class adalah gabungan bidang (data) dan metode (fungsi yang merupakan bagian dari Class) yang dapat dipakai sebagai objek. Huruf pertama dari nama kelas biasanya huruf besar untuk memisahkannya dari jenis variabel lainnya.

Sintak

```
class ClassName {
  statements
}
```

Parameter-parameter

ClassName	Nama variabel apapun yang valid.
Statement	Pernyataan yang valid.

DRAW ()

Examples

```
float yPos = 0,0;
void setup () {// setup () berjalan sekali
  ukuran (200, 200);
```

```
  frameRate (30);
}
void draw () {// draw () loop selamanya, sampai berhenti
  latar belakang (204);
  yPos = yPos - 1.0;
  if (yPos <0) {
    yPos = tinggi;
  }
  baris (0, yPos, lebar, yPos);
}

Void setup() {
  size (200, 200);
}

// Meskipun kosong di sini, draw () diperlukan juga
// sketsa dapat memproses input acara pengguna
// (mouse menekan dalam hal ini).


void draw () {}
batal mouseTekan () {
  line (mouseX, 10, mouseX, 90);
}
void mousePressed() {
  line(mouseX, 10, mouseX, 90);
}
```

Diskripsi

Dipanggil langsung setelah setup () , fungsi draw () terus menerus mengeksekusi baris kode yang terkandung di dalam bloknya sampai program dihentikan atau noLoop () dipanggil. draw () dipanggil secara otomatis dan tidak boleh dipanggil

secara eksplisit. Semua program Pemrosesan memperbarui layar pada akhir pengundian (), tidak pernah lebih awal.
Untuk menghentikan kode di dalam draw () dari berjalan terus menerus, gunakan noLoop () , redraw () dan loop () . Jika noLoop () digunakan untuk menghentikan kode draw () agar tidak berjalan, maka redraw () akan menyebabkan kode di dalam draw ()untuk menjalankan waktu tunggal, dan loop () akan menyebabkan kode di dalam draw () untuk terus berjalan.
Jumlah kali draw () dieksekusi di setiap detik dapat dikontrol dengan fungsi frameRate (). Biasanya memanggil background () di dekat awal draw () untuk menghapus isi jendela, seperti yang ditunjukkan pada contoh pertama di atas. Karena piksel yang ditarik ke jendela bersifat kumulatif, menghilangkan background () dapat menghasilkan hasil yang tidak diinginkan. Hanya ada satu fungsi draw () untuk setiap sketsa, dan draw () harus ada jika Anda ingin kode dijalankan terus menerus, atau untuk memproses peristiwa seperti mousePressed () . Terkadang, Anda mungkin memiliki panggilan kosong untuk draw () di program Anda, seperti yang ditunjukkan pada contoh kedua di atas.

Sintak
Draw()

EXIT()

Examples

```
void draw () {
  line (mouseX, mouseY, 50, 50);
}
void mousePress () {
  exit();
```

}

Diskripsi

Berhenti / berhenti / keluar dari program. Program tanpa fungsi draw () berhenti secara otomatis setelah baris terakhir dijalankan, tetapi program dengan draw () berjalan terus menerus hingga program dihentikan secara manual atau exit ()dijalankan.

Daripada menghentikan segera, exit () akan menyebabkan sketsa keluar setelah draw () telah selesai (atau setelah setup () selesai jika dipanggil selama fungsi setup ()).

Untuk Jawa programmer, ini tidak sama dengan System exit (). Lebih lanjut, System exit () tidak boleh digunakan karena menutup aplikasi saat draw () berjalan dapat menyebabkan crash (terutama dengan P3D).

Sintak

exit()

Extends

Examples

```
DrawDot dd1 = DrawDot new (50, 80);
void setup () {
  size (200, 200);
}
void draw () {
  dd1.display ();
}
class Dot {
  int xpos, ypos;
```


```
}

class DrawDot extends Dot {
  DrawDot (int x, int y) {
    xpos = x;
    ypos = y;
  }
  Void display () {
    ellipse (xpos, ypos, 200, 200);
  }
}
```

Diskripsi

Mengizinkan kelas baru mewarisi bidang metode dan data (variabel dan konstanta) dari kelas yang ada. Dalam kode, sebutkan nama kelas baru, diikuti dengan kata kunci extends dan nama kelas dasar . Konsep pewarisan adalah salah satu prinsip dasar pemrograman berorientasi objek.

Perhatikan bahwa di Java, dan karenanya juga Memproses, Anda tidak dapat memperluas kelas lebih dari sekali. Sebagai gantinya, lihat implementasinya .

FALSE

Examples

```
rect (30, 20, 50, 50);
boolean b = false;
if (b == false) {
  baris (20, 10, 90, 80); // Baris ini ditarik
} lain {
  baris (20, 80, 90, 10); // Baris ini tidak ditarik
}
```

Diskripsi
Kata yang dicadangkan mewakili nilai logis "salah". Hanya variabel tipe boolean yang dapat diberi nilai false .

FINAL

Examples
constant float akhir = 12.84753;
println (konstan); // Mencetak "12.84753" ke konsol
konstan + = 12,84; // GALAT! Tidak mungkin mengubah nilai "akhir"

Diskripsi
Kata kunci yang digunakan untuk menyatakan bahwa nilai, kelas, atau metode tidak dapat diubah. Jika kata kunci final digunakan untuk mendefinisikan variabel, variabel tidak dapat diubah dalam program. Saat digunakan untuk mendefinisikan kelas, kelas final tidak dapat disubklasifikasikan. Ketika digunakan untuk mendefinisikan suatu fungsi atau metode, metode final tidak dapat ditimpa oleh subclass.
Kata kunci ini adalah bagian penting dari pemrograman Java dan biasanya tidak digunakan dengan Memproses. Konsultasikan referensi atau tutorial bahasa Jawa untuk informasi lebih lanjut.

IMPLEMENTS

Examples
interface Dot {
  void move();

```
  void display();
}
class CircleDot implements Dot {
  float x = 50;
  float y = 50;

  void move() {
    x = x + random(-1, 1);
  }

  void display() {
    ellipse(x, y, 16, 16);
  }
}

class SquareDot implements Dot {
  float x = 50;
  float y = 50;
  void move() {
    y = y + random(-1, 1);
  }

  void display() {
    rect(x, y, 16, 16);
  }
}
```

Diskripsi

Menerapkan antarmuka atau grup antarmuka. Antarmuka digunakan untuk membuat protokol antar kelas; mereka membuat formulir untuk kelas (nama metode, tipe pengembalian, dll.) tetapi tidak ada implementasi. Setelah implementasi, sebuah antarmuka dapat digunakan dan diperluas seperti kelas lainnya.

Karena Java tidak mengizinkan perluasan lebih dari satu kelas pada satu waktu, Anda dapat membuat antarmuka sebagai gantinya, sehingga metode dan bidang tertentu dapat ditemukan di kelas yang mengimplementasikannya. Thread adalah contoh; itu mengimplementasikan antarmuka "Runnable", yang berarti kelas memiliki metode yang disebut "public void run ()" di dalamnya.

IMPORT

Examples

```
import processing.pdf.*
void setup() {
  size(1024, 768, PDF);
}
void draw() {
  line(0, 0, width, height);
  line(0, height, width, 0);
}
```

Diskripsi

Impor kata kunci digunakan untuk memuat pustaka ke dalam sketsa Pemrosesan. Perpustakaan adalah satu atau beberapa kelas yang dikelompokkan bersama untuk memperluas kemampuan Pemrosesan. The * karakter sering digunakan pada akhir baris impor (lihat contoh kode di atas) untuk memuat semua kelas terkait sekaligus, tanpa harus referensi mereka secara individu.

Pernyataan impor akan dibuat untuk Anda dengan memilih perpustakaan dari item "Impor Perpustakaan ..." di menu Sketsa.

Sintak
import libraryName
Parameter-parameter
libraryName nama library untuk memuat (mis. "processing.pdf. *")

loop()

Examples

```
void setup() {
size (200, 200);
  noLoop (); // draw () tidak akan mengulang
}
float x = 0;
void draw() {
  background(204);
  x = x + .1;
  if (x > width) {
    x = 0;
  }
  line(x, 0, x, height);
}
void mousePressed() {
  loop();  // Menekan mouse akan mengaktifkan looping
}
void mouseReleased() {
  noLoop();    // Melepaskan mouse berhenti menggambar berulang ()
}
```

Diskripsi
Secara default, Memproses loop melalui draw () terus menerus, mengeksekusi kode di dalamnya. Namun, loop draw() dapat dihentikan dengan memanggil noLoop(). Jika demikian, loop draw() dapat dilanjutkan dengan loop().

Sintak
Loop()

new

Examples

```
HLine h1 = new HLine();
float[] speeds = new float[3];
float ypos;

void setup() {
  size(200, 200);
  speeds[0] = 0.1;
  speeds[1] = 2.0;
  speeds[2] = 0.5;
}
void draw() {
  ypos += speeds[int(random(3))];
  if (ypos > width) {
    ypos = 0;
  }
  h1.update(ypos);
}
class HLine {
  void update(float y) {
```

```
    line(0, y, width, y);
  }
}
```

Diskripsi
Membuat objek "new". Kata kunci new biasanya digunakan mirip dengan aplikasi dalam contoh di atas. Dalam contoh ini, objek baru "h1" dari tipe data "HLine" dibuat. Pada baris berikut, array baru pelampung yang disebut "speeds" dibuat.

noLoop()

Examples

```
void setup() {
  size(200, 200);
  noLoop();
}

void draw() {
  line(10, 10, 190, 190);
}
void setup() {
  size(200, 200);
}
float x = 0.0;
void draw() {
  background(204);
  x = x + 0.1;
  if (x > width) {
    x = 0;
  }
  line(x, 0, x, height);
```

```
}
void mousePressed() {
  noLoop();
}
void mouseReleased() {
  loop();
}

boolean someMode = false;

void setup() {
  noLoop();
}

void draw() {
  if (someMode) {
    // melakukan sesuatu
  }
}

void mousePressed() {
  someMode = true;
  redraw();  // atau loop()
}
```

Diskripsi

Menghentikan Memproses dari terus mengeksekusi kode dalam draw () . Jika loop () dipanggil, kode dalam draw () mulai berjalan terus menerus. Jika menggunakan noLoop () di setup () , itu harus menjadi baris terakhir di dalam blok.

Ketika noLoop () digunakan, itu tidak mungkin untuk memanipulasi atau mengakses layar di dalam fungsi-fungsi penanganan acara seperti mousePressed() atau keyPressed () . Sebagai gantinya, gunakan fungsi-fungsi itu untuk memanggil

redraw() atau loop() , yang akan menjalankan draw() , yang dapat memperbarui layar dengan benar. Ini berarti bahwa ketika noLoop() telah dipanggil, tidak ada gambar yang dapat terjadi, dan fungsi seperti saveFrame () atau loadPixels() tidak dapat digunakan.
Perhatikan bahwa jika sketsa diubah ukurannya, redraw () akan dipanggil untuk memperbarui sketsa, bahkan setelah noLoop () telah ditentukan. Kalau tidak, sketsa akan memasuki kondisi ganjil sampai loop () dipanggil.

Sintak

noLoop ()

null

Examples

```
String content = "Ini hari yang indah.";
String [] hasil; // Nyatakan array String kosong

hasil = cocok (konten, "oranye");
// Pernyataan pertandingan di atas akan gagal ditemukan
// kata "oranye" di String 'konten', jadi
// itu akan mengembalikan nilai nol ke 'hasil'.
if (hasil == null) {
  println ("Nilai 'hasil' adalah nol."); // Baris ini dicetak
} else {
  println ("Nilai 'hasil' bukan nol!"); // Baris ini tidak dicetak
}
```

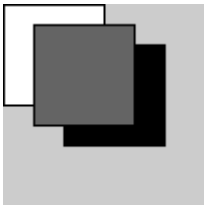
Diskripsi

Nilai khusus yang digunakan untuk menandakan target bukanlah elemen data yang valid. Dalam Memproses, Anda

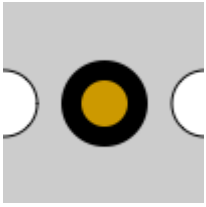
dapat menjalankan seluruh kata kunci null ketika mencoba mengakses data yang tidak ada.

pop()

Examples



```
fill(255);
rect(0, 0, 50, 50);//persegi panjang putih
push();
translate(30, 20);
fill(0);
rect(0, 0, 50, 50);  //Persegi panjang hitam
pop();  // kembali ke pengaturan asli
fill(100);
rect(15, 10, 50, 50);  // persegi panjang abu-abu
```



```
ellipse(0, 50, 33, 33);  // Lingkaran kiri
push();
strokeWeight(10);
fill(204, 153, 0);
ellipse(50, 50, 33, 33);  // lingkaran tengah
pop();  // kembali pengaturan asli
ellipse(100, 50, 33, 33);  // lingkaran kanan
```

Diskripsi

Fungsi pop () mengembalikan pengaturan dan transformasi gaya gambar sebelumnya setelah push () mengubahnya. Perhatikan bahwa fungsi-fungsi ini selalu digunakan bersama. Mereka memungkinkan Anda untuk mengubah pengaturan gaya dan transformasi dan kemudian kembali ke apa yang

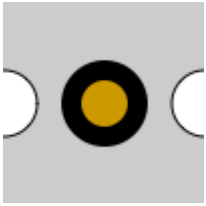
Anda miliki. Ketika negara baru dimulai dengan push (), itu dibangun pada gaya saat ini dan mengubah informasi.
push ()menyimpan informasi yang terkait dengan keadaan transformasi saat ini dan pengaturan gaya yang dikendalikan oleh fungsi berikut: rotate () , translate () , scale () , fill () , stroke () , tint () , tint () , strokeWeight () ,strokeCap () , strokeJoin () , imageMode () , rectMode () , ellipseMode () , colorMode () , textAlign () , textFont () , textMode () , textSize () , textLeading () , textLeading () .
Fungsi push () dan pop () ditambahkan dengan Memproses 3.5. Mereka dapat digunakan di tempat pushMatrix () , popMatrix () , pushStyles () , dan popStyles (). Perbedaannya adalah bahwa push () dan pop () mengontrol kedua transformasi (memutar, skala, menerjemahkan) dan gaya gambar pada saat yang sama.

Sintak

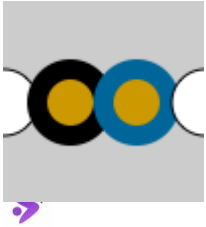
Pop()

popStyle ()

Examples



```
ellipse (0, 50, 33, 33); // Lingkaran kiri
pushStyle (); // Mulai gaya baru
strokeWeight (10);
isi (204, 153, 0);
ellipse (50, 50, 33, 33); // Lingkaran tengah
popStyle (); // Kembalikan gaya asli
ellipse (100, 50, 33, 33); // Lingkaran kanan
```



```
ellipse (0, 50, 33, 33); // Lingkaran kiri
pushStyle (); // Mulai gaya baru
strokeWeight (10);
isi (204, 153, 0);
```

```
ellipse (33, 50, 33, 33); // Lingkaran kiri-tengah
pushStyle (); // Mulai gaya baru lainnya
stroke (0, 102, 153);
ellipse (66, 50, 33, 33); // Lingkaran tengah-kanan
popStyle (); // Kembalikan gaya sebelumnya

popStyle (); // Kembalikan gaya asli
ellipse (100, 50, 33, 33); // Lingkaran kanan
```

Diskripsi

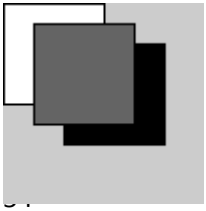
Fungsi pushStyle () menyimpan pengaturan gaya saat ini dan popStyle () mengembalikan pengaturan sebelumnya; fungsi-fungsi ini selalu digunakan bersama. Mereka memungkinkan Anda untuk mengubah pengaturan gaya dan kemudian kembali ke apa yang Anda miliki. Ketika gaya baru dimulai dengan pushStyle () , itu dibangun di atas informasi gaya saat ini. Fungsi pushStyle () dan popStyle () dapat disematkan untuk memberikan lebih banyak kontrol (lihat contoh kedua di atas untuk demonstrasi.)

Sintak

popStyle ()

push()

Examples



```
fill(255);
rect(0, 0, 50, 50);  // persegi panjang putih

push();
```

```
translate(30, 20);
fill(0);
rect(0, 0, 50, 50);  // persegi panjang hitam
pop();  // kembali pada pengaturan asli

fill(100);
rect(15, 10, 50, 50);  // persegi panjang abu-abu
```

Diskripsi

Fungsi push () menyimpan pengaturan dan transformasi gaya gambar saat ini, sementara pop () mengembalikan pengaturan ini. Perhatikan bahwa fungsi-fungsi ini selalu digunakan bersama. Mereka memungkinkan Anda untuk mengubah pengaturan gaya dan transformasi dan kemudian kembali ke apa yang Anda miliki. Ketika negara baru dimulai dengan push (), itu dibangun pada gaya saat ini dan mengubah informasi.
push () menyimpan informasi yang terkait dengan keadaan transformasi saat ini dan pengaturan gaya yang dikendalikan oleh fungsi berikut: rotate() , translate() , scale() , fill() ,stroke() , tint() , tint() , strokeWeight() ,strokeCap() , strokeJoin() , imageMode() , rectMode() , ellipseMode() , colorMode() , textAlign() , textFont() , textMode() , textSize() , textLeading() , textLeading() .

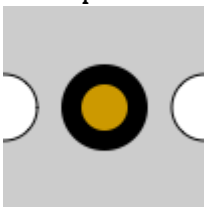
Fungsi push() dan pop() ditambahkan dengan Processing 3.5. Mereka dapat digunakan di tempat pushMatrix () , popMatrix() , pushStyles() ,dan popStyles(). Perbedaannya adalah bahwa push () dan pop () mengontrol kedua transformasi (memutar, skala, menerjemahkan) dan gaya gambar pada saat yang sama.

Sintak

Push ()

pushStyle()

Examples



```
ellipse(0, 50, 33, 33);  // lingkara kiri
pushStyle();  // mulai gaya baru
strokeWeight(10);
fill(204, 153, 0);
ellipse(50, 50, 33, 33);  // lingkaran tengah
popStyle();  // kembali pada gaya aslinya
ellipse(100, 50, 33, 33);  // lingkaran kanan
```

```
ellipse(0, 50, 33, 33);  // lingkaran kiri
pushStyle();  // mulai gaya baru
strokeWeight(10);
fill(204, 153, 0);

ellipse(33, 50, 33, 33);  // lingkaran kiri-tengah
pushStyle();  // memulai gaya baru lainnya
stroke(0, 102, 153);
ellipse(66, 50, 33, 33);  // lingkaran kanan-tengah
popStyle();  // kembali pada gaya sebelumnya
popStyle();  // kembali pada gaya aslinya
ellipse(100, 50, 33, 33);  // lingkaran kanan
```

Desccription

Fungsi pushStyle() menyimpan pengaturan gaya saat ini dan popStyle() untuk mengembalikan pengaturan sebelumnya. Perhatikan bahwa fungsi-fungsi ini selalu digunakan bersama. Mereka memungkinkan Anda untuk mengubah pengaturan gaya dan kemudian kembali ke apa yang Anda miliki. Ketika gaya baru dimulai dengan pushStyle () , itu dibangun di atas informasi gaya saat ini. Fungsi pushStyle () dan popStyle() dapat disematkan untuk memberikan lebih banyak kontrol. Informasi gaya yang dikendalikan oleh fungsi-fungsi berikut termasuk dalam gaya: fill (), stroke (), tint (), strokeWeight (), strokeCap (), strokeJoin (), imageMode (), rectMode (), rectMode (), ellipseMode (), shapeMode (), colorMode (), textAlign (), textFont (), textMode (), textSize (), textLeading (), emissive (), specular (), shininess (), ambient ()

Sintak

pushStyle ()

redraw()

Examples

```
float x = 0;
void setup() {
  size(200, 200);
  noLoop();
}
void draw() {
  background(204);
  line(x, 0, x, height);
}
void mousePressed() {
```

```
  x += 1;
  redraw();
}
```

Diskripsi

Menjalankan kode dalam draw() satu kali. Fungsi ini memungkinkan program untuk memperbarui jendela tampilan hanya jika diperlukan, misalnya ketika suatu peristiwa terdaftar oleh mousePressed () atau keyPressed () terjadi.

Dalam menyusun program, masuk akal untuk memanggil redraw () dalam acara seperti mousePressed () . Ini karena redraw () tidak menjalankan draw () dengan segera (hanya menetapkan flag yang menunjukkan pembaruan diperlukan).

Fungsi redraw() tidak berfungsi dengan benar saat dipanggil inside draw.Untuk mengaktifkan/menonaktifkan animasi, gunakan loop () dan noLoop () .

Sintak

Redraw ()

Return

Examples

```
int val = 30;
void draw() {
  int t = timestwo(val);
  println(t);
}
// 'int' pertama dalam deklarasi fungsi.
// menentukan tipe data yang akan dikembalikan.
int timestwo(int dVal) {
  dVal = dVal * 2;
```

```
  return dVal;  // dalam hal ini mengembalikan int 60
}

int[] vals = {10, 20, 30};

void draw() {
  int[] t = timestwo(vals);
  println(t);
  noLoop();
}

int[] timestwo(int[] dVals) {
  for (int i = 0; i < dVals.length; i++) {
    dVals[i] = dVals[i] * 2;
  }
  return dVals;  // mengembalikan 3 array int : 20,40,60
}
void draw() {
  background(204);
  line(0, 0, width, height);
  if (mousePressed) {
    return;  // keluar dari perintah draw(), melompati pernyataan
line dibawah ini.
  }
  line(0, height, width, 0);   // dapat di eksekusi jika tidak
menekan mouse
}
```

Diskripsi

Kata kunci yang digunakan untuk menunjukkan nilai untuk kembali dari suatu fungsi. Nilai yang dikembalikan harus datatype yang sama seperti yang didefinisikan dalam deklarasi fungsi. Fungsi yang dinyatakan dengan void tidak dapat mengembalikan nilai dan tidak boleh menyertakan nilai balik.

Kata kunci kembali juga dapat digunakan untuk keluar dari fungsi, sehingga tidak memungkinkan program untuk laporan yang tersisa.

Sintak

```
type function {
  statements
  return value
}
```

Parameter-parameter

Type boolean, byte, char, int, float, String, boolean [], byte [], char [], int [], float [], atau String []

Function	Fungsi yang sedang didefinisikan.
Statments	pernyataan yang falid.
Value	harus datatype yang sama dengan parameter "type"

setup()

Examples

```
int x = 0;
void setup() {
  size(200, 200);
  background(0);
  noStroke();
  fill(102);
}
void draw() {
  rect(x, 10, 2, 80);
  x = x + 1;
```

```
}
int x = 0;
void setup() {
  fullScreen();
  background(0);
  noStroke();
  fill(102);
}
void draw() {
  rect(x, height*0.2, 1, height*0.6);
  x = x + 2;
}
```

Diskripsi

Fungsi setup () dijalankan sekali, ketika program dimulai. Ini digunakan untuk mendefinisikan properti lingkungan awal seperti ukuran layar dan memuat media seperti gambar dan font saat program dimulai. Hanya ada satu fungsi setup () untuk setiap program dan tidak boleh dipanggil lagi setelah eksekusi awal.

Jika sketsa memiliki dimensi yang berbeda dari standar, fungsi size () atau fungsi fullscreen () harus menjadi baris pertama dalam setup () .

Catatan: Variabel yang dideklarasikan dalam setup () tidak dapat diakses dalam fungsi lain, termasuk draw () .

Sintak

setup()

static

Examples

```
void setup() {
  MiniClass mc1 = new MiniClass();
  MiniClass mc2 = new MiniClass();
  println( mc1.y );   // mencetak ‘10’ ke konsol
  MiniClass.y += 10;  // Variabel ‘y’ dibagikan oleh ‘mc1’ dan ‘mc2’
  println( mc1.y );   // mencetak "20" ke konsol
  println( mc2.y );   // mencetak "20" ke konsol
}
static class MiniClass {
  static int y = 10;  // Class variable
}
```

```
void setup() {
  println(MiniClass.add(3, 4));  // mencetak "7" ke konsol

static class MiniClass {
  static int add(int x, int y) {
    return(x + y);
  }
}
```

Diskripsi

Kata kunci yang digunakan untuk mendefinisikan variabel sebagai " class variable" dan metode sebagai " class method." Ketika suatu variabel dideklarasikan dengan kata kunci statis , semua instance dari kelas itu berbagi variabel yang sama. Ketika sebuah kelas didefinisikan dengan kata kunci static , metode itu dapat digunakan tanpa membuat turunan dari kelas. Contoh di atas menunjukkan masing-masing kegunaan ini.

Kata kunci ini adalah bagian penting dari pemrograman Java dan biasanya tidak digunakan dengan memproses. Konsultasikan referensi atau tutorial bahasa Jawa untuk informasi lebih lanjut.

SUPER

Examples

```
// Contoh ini adalah fragmen kode;
// tidak akan dikompilasi sendiri.

// Buat subclass DragDrop dari
// kelas Button. Jadi button
// superclass dari DragDrop.
class DragDrop extends Button {
  int xoff, yoff;
  DragDrop(int x, int y) {
// Menjalankan super konstruktor superclass ' (x, y);
  }
  void press(int mx, int my) {
    // Menjalankan metode superclass 'press ()
    super.press();
    xoff = mx;
    yoff = my;
  }
}
```

Diskripsi

Kata kunci yang digunakan untuk referensi superclass dari subclass.

thread()

Examples

```
String time = "";
void setup() {
  size(100, 100);
}
void draw() {
  background(0);
  // Setiap 30 frame meminta data baru
  if (frameCount % 30 == 0) {
    thread("requestData");
  }
  text(time, 10, 50);
}
// Ini terjadi sebagai utas terpisah dan dapat berlangsung selama ia ingin.
void requestData() {
  JSONObject json = loadJSONObject("http://time.jsontest.com/");
  time = json.getString("time");
}
```

Diskripsi

Pemrosesan sketsa mengikuti urutan langkah-langkah spesifik: setup () pertama, diikuti oleh draw () berulang-ulang dalam satu lingkaran. Utas juga merupakan serangkaian langkah dengan awal, tengah, dan akhir. Sketsa pemrosesan adalah utas tunggal, sering disebut sebagai utas "Animasi". Namun, urutan utas lainnya dapat berjalan secara independen dari loop animasi utama. Bahkan, Anda dapat meluncurkan sejumlah utas sekaligus, dan semuanya akan berjalan bersamaan.

Anda tidak dapat menggambar ke layar dari fungsi yang dipanggil oleh thread(). Karena berjalan secara independen, kode tidak akan disinkronkan ke utas animasi, menyebabkan hasil yang aneh atau setidaknya tidak konsisten. Gunakan thread()untuk memuat file atau melakukan tugas lain yang membutuhkan waktu. Ketika tugas selesai, setel variabel yang menunjukkan tugas selesai, dan periksa dari dalam metode draw () Anda .

Pemrosesan menggunakan utas cukup sering, seperti dengan fungsi perpustakaan seperti captureEvent () dan movieEvent () . Fungsi-fungsi ini dipicu oleh utas berbeda yang berjalan di belakang layar, dan mereka mengingatkan Pemrosesan kapan pun mereka memiliki sesuatu untuk dilaporkan. Ini berguna ketika Anda perlu melakukan tugas yang terlalu lama dan akan memperlambat laju bingkai animasi utama, seperti mengambil data dari jaringan. Jika utas terpisah macet atau memiliki kesalahan, seluruh program tidak akan berhenti, karena kesalahan hanya menghentikan utas individu tersebut.
Menulis utas Anda sendiri bisa menjadi upaya kompleks yang melibatkan perluasan kelas Java Thread . Namun, metode thread () adalah cara cepat dan kotor untuk mengimplementasikan utas sederhana dalam Pemrosesan. Dengan melewatkan String yang cocok dengan nama fungsi yang dideklarasikan di tempat lain dalam sketsa, Pemrosesan akan menjalankan fungsi itu di utas terpisah.

Sintak
thread(name)

Parameter-parameter
Name String: nama fungsi yang akan dieksekusi di thread terpisah.

True

Examples

```
rect(30, 20, 50, 50);
boolean b = true;
if (b == true) {
  line(20, 10, 90, 80);  // baris ini ditarik
} else {
  line(20, 80, 90, 10);  // baris ini tidak ditarik
}
```

Diskripsi

Kata yang dicadangkan mewakili nilai logis "true". Hanya variabel tipe boolean yang dapat diberi nilai true .

try

Examples

```
BufferedReader reader;
String line;
void setup() {
  // buka fle dari contoh createWriter()
  reader = createReader("positions.txt");
}
void draw() {
  try {
    line = reader.readLine();
  } catch (IOException e) {
    e.printStackTrace();
    line = null;
```


```
  }
  if (line == null) {
    // berhenti membaca karena kesalahan atau file kosong
    noLoop();
  } else {
    String[] pieces = split(line, TAB);
    int x = int(pieces[0]);
    int y = int(pieces[1]);
    point(x, y);
  }
}
```

Diskripsi

Kata kunci try digunakan dengan catch untuk menangani pengecualian. Dokumentasi Sun Java mendefinisikan pengecualian sebagai "suatu peristiwa, yang terjadi selama pelaksanaan suatu program, yang mengganggu aliran normal instruksi program." Ini bisa berupa, misalnya, kesalahan saat file dibaca.

Sintak

```
try {
  tryStatements
} catch (exception) {
  catchStatements
}
```

Parameter-parameter

tryStatements jika kode ini memunculkan eksepsi, maka kode dalam "catch" dijalankan.

exception pengecualian Java yang dilempar

catchStatements kode yang menangani pengecualian

VOID

Examples

```
void setup() {  // setup() tidak megembalikan nilai
  size(200, 200);
}
void draw() {  // draw() tidak mengembalikan nilai
  line(10, 100, 190, 100);
  drawCircle();
}
void drawCircle() {  // Fungsi ini juga tidak mengembalikan nilai
  ellipse(30, 30, 50, 50);
}
```

Diskripsi

Kata kunci yang digunakan menunjukkan bahwa suatu fungsi tidak mengembalikan nilai. Setiap fungsi harus mengembalikan nilai tipe data tertentu atau menggunakan kata kunci void untuk menentukan itu tidak menghasilkan apa-apa.

Sintak

```
void function {
  statements
}
```

Parameter-parameter

Function Fungsi apaun yang sedang dedefinisikan atau diimplementasikan.

Statements Pernyataan yang valid.

BAB IV
ENVIRONMENT

CURSOR()

Examples

```
// Gerakkan mouse ke kiri dan kanan melintasi gambar
// untuk melihat perubahan kursor dari tangan
void setup() {
  size(100, 100);
}
void draw() {
  if (mouseX < 50) {
    cursor(CROSS);
  } else {
    cursor(HAND);
  }
}
```

Diskripsi

Menyetel kursor ke simbol atau gambar yang telah ditentukan, atau membuatnya terlihat jika sudah tersembunyi. Jika Anda mencoba menetapkan gambar sebagai kursor, ukuran yang disarankan adalah 16x16 atau 32x32 piksel. Nilai untuk parameter x dan y harus lebih kecil dari dimensi gambar.

Mengatur atau menyembunyikan kursor umumnya tidak berfungsi dengan mode "Present" (saat menjalankan layar penuh).

Dengan renderer P2D dan P3D, seperangkat kursor generik digunakan karena renderer OpenGL tidak memiliki akses ke gambar kursor default untuk setiap platform .

Sintak

```
cursor(kind)
cursor(img)
cursor(img, x, y)
```


cursor()

Parameter-parameter

Kind int: either ARROW, CROSS, HAND, MOVE, TEXT, atau WAIT
img PImage: variabel apa pun dari tipe Pimage
x int: titik aktif horisontal kursor
y int: titik aktif vertikal kursor

delay()

Examples

```
import processing.serial.*;
Serial myPort;  // port serial
void setup() {
  printArray(Serial.list());
  myPort = new Serial(this, Serial.list()[0], 9600);
}

void draw() {
  while (myPort.available() > 0) {
    int inByte = myPort.read();
    println(inByte);
  }
  delay(100);
}
```

Diskripsi

Fungsi delay () berhenti untuk waktu yang ditentukan. Waktu tunda ditentukan dalam seperseribu detik. Misalnya, menjalankan delay (3000) akan menghentikan program selama

tiga detik dan delay (500) akan menghentikan program selama setengah detik.
Layar hanya diperbarui ketika akhir draw () tercapai, jadi delay () tidak dapat digunakan untuk memperlambat draw. Misalnya, Anda tidak dapat menggunakan delay () untuk mengontrol waktu animasi.
delay () fungsi hanya boleh digunakan untuk menjeda skrip (yaitu skrip yang perlu menjeda beberapa detik sebelum mencoba mengunduh, atau sketsa yang perlu menunggu beberapa milidetik sebelum membaca dari port serial.)

Sintak
delay(napTime)

Parameter-parameter
napTime int: milidetik untuk berhenti sebelum menjalankan draw () lagi.

displayDensity()

Examples

```
void setup() {
  size(100, 100);
  pixelDensity(displayDensity());
  noStroke();
}
void draw() {
  background(0);
  ellipse(30, 48, 36, 36);
  ellipse(70, 48, 36, 36);
}
```


Diskripsi
Fungsi ini mengembalikan angka "2" jika layar adalah layar kepadatan tinggi (disebut tampilan Retina pada OS X atau dpi tinggi pada Windows dan Linux) dan "1" jika tidak. Informasi ini berguna untuk sebuah program untuk beradaptasi agar berjalan dua kali lipat kerapatan piksel pada layar yang mendukungnya.

Sintak
displayDensity()
displayDensity(display)

Parameter-parameter
display int: nomor tampilan untuk diperiksa

focused

Examples

```
if (focused) {  // atau "if (focused == true)"
  ellipse(25, 25, 50, 50);
} else {
  line(0, 0, 100, 100);
  line(100, 0, 0, 100);
}
```

Diskripsi
Mengonfirmasi jika program Pemrosesan "focused," yang berarti program itu aktif dan akan menerima input mouse atau keyboard. Variabel ini "true" jika fokus dan "false" jika tidak.

frameCount

Examples

```
void setup() {
  frameRate(30);
}

void draw() {
  line(0, 0, width, height);
  println(frameCount);
}
```

Diskripsi

FrameCount variabel sistem berisi jumlah frame yang telah ditampilkan sejak program dimulai. Di dalam setup () value 0, setelah iterasi pertama draw adalah 1, dll.

frameRate()

Examples

```
void setup() {
  frameRate(4);
}
int pos = 0;
void draw() {
  background(204);
  pos++;
  line(pos, 20, pos, 80);
  if (pos > width) {
```


```
    pos = 0;
  }
}
```

Diskripsi
Menentukan jumlah bingkai yang akan ditampilkan setiap detik. Misalnya, frameRate panggilan fungsi (30) akan mencoba untuk menyegarkan 30 kali per detik. Jika prosesor tidak cukup cepat untuk mempertahankan laju yang ditentukan, laju bingkai tidak akan tercapai. Menyetel kecepatan bingkai dalam setup() disarankan. Tingkat default adalah 60 frame per detik.

Sintak
frameRate(fps)

Parameter-parameter
Fps float: jumlah bingkai yang diinginkan per detik

frameRate

Examples

```
void setup() {
  frameRate(30);
}
void draw() {
  line(0, 0, width, height);
  println(frameRate);
}
```

Diskripsi

FrameRate variabel sistem berisi perkiraan frame rate dari sketsa yang sedang berjalan. Nilai awal adalah 10 fps dan diperbarui dengan setiap frame. Nilainya dirata-rata pada beberapa frame, dan hanya akan akurat setelah fungsi draw berjalan 5-10 kali.

fullScreen()

Examples

```
// Jalankan kode pada dimensi penuh layar saat ini
// dipilih di dalam jendela Preferensi

int x = 0;
void setup() {
  fullScreen();
  background(0);
  noStroke();
  fill(102);
}
void draw() {
  rect(x, height*0.2, 1, height*0.6);
  x = x + 2;
}

// Jika lebih dari satu layar terpasang ke komputer, jalankan kode
// pada dimensi penuh pada layar yang ditentukan oleh Parameter
 // ke fullScreen ()

int x = 0;
```


```
void setup() {
  fullScreen(2);
  background(0);
  noStroke ();
  fill(102);
}

void draw() {
  rect(x, height*0.2, 1, height*0.6);
  x = x + 2;
}

// Jalankan layar penuh menggunakan renderer P2D pada layar
2

int x = 0;
void setup() {
  fullScreen(P2D, 2);
  background(0);
  noStroke();
  fill(102);
}

void draw() {
  rect(x, height*0.2, 1, height*0.6);
  x = x + 2;
}
// Jika lebih dari satu layar terpasang ke komputer, jalankan
kode
// pada dimensi penuh di semua layar yang terpasang
int x = 0;
void setup() {
  fullScreen(P2D, SPAN);
  background(0);
```

```
  noStroke();
  fill(102);
}
void draw() {
  rect(x, height*0.2, 1, height*0.6);
  x = x + 2;
}
```

Diskripsi

Fungsi ini baru untuk Memproses 3.0. Ini membuka sketsa menggunakan ukuran penuh layar komputer. Fungsi ini harus menjadi baris pertama dalam setup () . Fungsi size () dan fullScreen () tidak bisa digunakan dalam program yang sama, cukup pilih satu.

Ketika fullScreen () digunakan tanpa parameter, itu menggambar sketsa ke layar yang saat ini dipilih di dalam jendela Preferensi. Ketika digunakan dengan satu parameter, angka ini menentukan layar untuk ditampilkan ke program aktif (misalnya 1, 2, 3 ...). Ketika digunakan dengan dua parameter, yang pertama mendefinisikan renderer untuk digunakan (misalnya P2D) dan yang kedua mendefinisikan layar. The SPANparameter dapat digunakan sebagai pengganti nomor layar untuk menggambar sketsa sebagai jendela layar penuh di semua tampilan yang terpasang jika ada lebih dari satu. Sebelum Memproses 3.0, program layar penuh didefinisikan dengan ukuran (displayWidth, displayHeight) .

Sintak

fullScreen()
fullScreen(display)
fullScreen(renderer)
fullScreen(renderer, display)

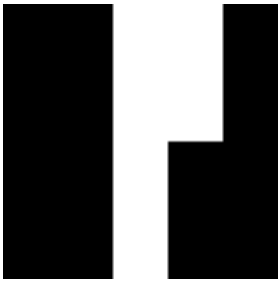
Parameter-parameter

Renderer String: renderer untuk digunakan, misalnya P2D, P3D, JAVA2D (default)

Display int: layar untuk menjalankan sketsa (1, 2, 3, dll. atau beberapa layar menggunakan SPAN)

height

Examples



```
noStroke();
background(0);
rect(40, 0, 20, height);
rect(60, 0, 20, height/2);
```

Diskripsi

Variabel sistem yang menyimpan ketinggian jendela tampilan. Nilai ini ditentukan oleh parameter kedua dari fungsi size () . Misalnya, ukuran panggilan fungsi (320, 240) mengatur variabel tinggi ke nilai 240. Nilai tinggi standar menjadi 100 jika size () tidak digunakan dalam suatu program.

noCursor()

Examples

```
// Tekan mouse untuk menyembunyikan kursor
void draw()
{
  if (mousePressed == true) {
    noCursor();
  } else {
    cursor(HAND);
  }
}
```

Diskripsi
Menyembunyikan kursor dari pandangan. Tidak akan berfungsi saat menjalankan program di browser web atau dalam mode layar penuh (Sekarang).

Sintak
noCursor ()

noSmooth()

Examples

```
size(100, 100);
noSmooth();
noStroke();
background(0);
ellipse(30, 48, 36, 36);
```


```
ellipse(70, 48, 36, 36);
void setup() {
  size(100, 100, P2D);
  noSmooth();
  noStroke();
}

void draw() {
  background(0);
  ellipse(30, 48, 36, 36);
  ellipse(70, 48, 36, 36);
}
```

Diskripsi
Gambar semua geometri dan font dengan tepi dan gambar bergerigi (alias) ketika tepi yang keras di antara piksel saat diperbesar alih-alih piksel yang disatukan. Perhatikan bahwa smooth () aktif secara default, sehingga perlu memanggil noSmooth () untuk menonaktifkan smoothing of geometry, font, dan gambar. Sejak rilis Processing 3.0, fungsi noSmooth () hanya dapat dijalankan satu kali untuk setiap sketsa, baik di bagian atas sketsa tanpa setup () , atau setelah fungsi size () bila digunakan dalam sketsa dengan setup () . Lihat contoh di atas untuk kedua skenario.

Sintak
noSmooth ()

pixelDensity()

Examples

```
size(100, 100);
```

```
pixelDensity(2);
noStroke();
background(0);
ellipse(30, 48, 36, 36);
ellipse(70, 48, 36, 36);
void setup() {
  size(100, 100);
  pixelDensity(2);
  noStroke();
}

void draw() {
  background(0);
  ellipse(30, 48, 36, 36);
  ellipse(70, 48, 36, 36);
}
void setup() {
  size(100, 100);
  // Pulling the display's density dynamically
  pixelDensity(displayDensity());
  noStroke();
}

void draw() {
  background(0);
  ellipse(30, 48, 36, 36);
  ellipse(70, 48, 36, 36);
}
```

Diskripsi

Fungsi ini baru dengan Memproses 3.0. Itu memungkinkan pemrosesan untuk merender menggunakan semua piksel pada layar resolusi tinggi seperti Apple Retina display dan Windows High-DPI display. Fungsi ini hanya dapat dijalankan satu kali dalam suatu program dan harus digunakan tepat setelah size ()

dalam suatu program tanpa setup () dan digunakan dalam setup () ketika sebuah program memiliki satu. The pixelDensity () hanya boleh digunakan dengan nomor hardcoded (dalam hampir semua kasus nomor ini akan 2) atau dalam kombinasi dengan displayDensity () seperti dalam contoh ketiga di atas. Untuk menggunakan variabel sebagai argumen ke fungsi pixelDensity () , tempatkan fungsi pixelDensity () di dalam setup () fungsi. Ada informasi lebih lanjut tentang ini di halaman referensi setup () .

Sintak
pixelDensity(density)

Parameter-parameter
Density int 1 atau 2

pixelHeight

Examples

```
size(600, 400);
pixelDensity(2);
println(width, height);
println(pixelWidth, pixelHeight);
```

Diskripsi
Ketika pixelDensity (2) digunakan untuk menggunakan tampilan resolusi tinggi (disebut tampilan Retina pada OS X atau dpi tinggi pada Windows dan Linux), lebar dan tinggi sketsa tidak berubah, tetapi jumlah piksel adalah dua kali lipat. Akibatnya, semua operasi yang menggunakan piksel (seperti loadPixels () , get () , set () , dll.) Terjadi di ruang berlipat ganda

ini. Sebagai kemudahan, variabel pixelWidth dan pixelHeight menampung lebar aktual dan tinggi sketsa dalam piksel. Ini berguna untuk setiap sketsa yang menggunakan array piksel [] , misalnya, karena jumlah elemen dalam array adalah pixelWidth * pixelHeight , bukan lebar * tinggi.

pixelWidth

Examples

```
size(600, 400);
pixelDensity(2);
println(width, height);
println(pixelWidth, pixelHeight);
```

Diskripsi

Ketika pixelDensity (2) digunakan untuk menggunakan tampilan resolusi tinggi (disebut tampilan Retina pada OS X atau dpi tinggi pada Windows dan Linux), lebar dan tinggi sketsa tidak berubah, tetapi jumlah piksel adalah dua kali lipat. Akibatnya, semua operasi yang menggunakan piksel (seperti loadPixels () , get () , set () , dll.) Terjadi di ruang berlipat ganda ini. Sebagai kemudahan, variabel pixelWidth dan pixelHeight menampung lebar aktual dan tinggi sketsa dalam piksel. Ini berguna untuk setiap sketsa yang menggunakan array piksel [] , misalnya, karena jumlah elemen dalam array adalah pixelWidth * pixelHeight , bukan lebar * tinggi.

settings()

Examples

```
// Jalankan kode pada layar penuh menggunakan renderer default
int x = 0;
void settings() {
  fullScreen();
}
void setup() {
  background(0);
  noStroke();
  fill(102);
}
void draw() {
  rect(x, height*0.2, 1, height*0.6);
  x = x + 2;
}
// Jalankan kode pada layar penuh menggunakan renderer P2D
// pada layar 2 dari beberapa pengaturan perangkat keras monitor
int x = 0;
void settings() {
  fullScreen(P2D, 2);
}
void setup() {
  background(0);
  noStroke();
  fill(102);
}
void draw() {
  rect(x, height*0.2, 1, height*0.6);
```

```
  x = x + 2;
}
// Jalankan kode pada layar penuh menggunakan renderer P2D
// di semua layar pada pengaturan beberapa monitor

int x = 0;
void settings() {
  fullScreen(P2D, SPAN);
}
void setup() {
  background(0);
  noStroke();
  fill(102);
}
void draw() {
  rect(x, height*0.2, 1, height*0.6);
  x = x + 2;
}
int w = 200;
int h = 200;
int x = 0;

void settings() {
  size(w, h);
}
void setup() {
  background(0);
  noStroke();
  fill(102);
}
void draw() {
  rect(x, 10, 1, 180);
  x = x + 2;
}
```


Diskripsi

Fungsi settings () baru dengan Processing 3.0. Itu tidak diperlukan di sebagian besar sketsa. Ini hanya berguna ketika benar-benar diperlukan untuk menentukan parameter to size () dengan variabel. Sebagai alternatif, fungsi settings () diperlukan ketika menggunakan kode Pemrosesan di luar Lingkungan Pengembangan Pemrosesan (PDE). Misalnya, ketika menggunakan code editor Eclipse, itu perlu untuk menggunakan settings () untuk menentukan size () dan smooth () nilai untuk sketsa ..

Settings () metode berjalan sebelum sketsa telah diatur, sehingga fungsi Pemrosesan lainnya tidak dapat digunakan pada saat itu. Misalnya, jangan gunakan loadImage () di dalam settings () . Metode settings () menjalankan "pasif" untuk mengatur beberapa variabel, dibandingkan dengan perintah setup () yang memanggil perintah dalam API Pemrosesan.

Sintak

settings()

size()

Examples

```
size(200, 100);
background(153);
line(0, 0, width, height);
void setup() {
  size(320, 240);
}
```

```
void draw() {
  background(153);
  line(0, 0, width, height);
}
size(150, 200, P3D);  // Specify P3D renderer
background(153);

// // Dengan P3D, kita dapat menggunakan nilai z (kedalaman)
...line(0, 0, 0, width, height, -100);
line(width, 0, 0, width, height, -100);
line(0, height, 0, width, height, -100);

//...dan fungsi khusus 3D, seperti kotak ()
translate(width/2, height/2);
rotateX(PI/6);
rotateY(PI/6);
box(35);
```

Diskripsi

Menentukan dimensi lebar dan tinggi jendela tampilan dalam satuan piksel. Dalam program yang memiliki fungsi settings () , fungsi size () harus menjadi baris pertama kode di dalam settings () , dan fungsi settings () harus muncul di tab kode dengan nama yang sama dengan folder sketsa Anda. Lebar dan tinggi.

Variabel bawaan ditetapkan oleh parameter yang diteruskan ke fungsi ini. Misalnya, ukuran lari (640, 480) akan menetapkan 640 untuk variabel lebar dan 480 untuk variabel tinggi . Jika size () tidak digunakan, jendela akan diberikan ukuran standar 100 x 100 piksel.

Fungsi size () hanya dapat digunakan sekali di dalam sketsa, dan tidak dapat digunakan untuk mengubah ukuran.

Pada Processing 3, untuk menjalankan sketsa pada dimensi penuh layar, gunakan fungsi fullScreen () , daripada cara yang lebih lama menggunakan ukuran (displayWidth, displayHeight).

Lebar dan tinggi maksimum dibatasi oleh sistem operasi Anda, dan biasanya lebar dan tinggi layar Anda yang sebenarnya. Pada beberapa mesin mungkin hanya jumlah piksel pada layar Anda saat ini, yang berarti bahwa layar 800 x 600 dapat mendukung ukuran (1600, 300), karena jumlah pikselnya sama. Ini sangat bervariasi, jadi Anda harus mencoba berbagai mode dan ukuran rendering hingga Anda mendapatkan yang Anda cari. Jika Anda membutuhkan sesuatu yang lebih besar, gunakan createGraphics untuk membuat permukaan gambar yang tidak terlihat.
P2D(Memproses 2D): renderer grafis 2D yang memanfaatkan perangkat keras grafis yang kompatibel dengan OpenGL.
P3D (Memproses 3D): renderer grafik 3D yang memanfaatkan perangkat keras grafis yang kompatibel dengan OpenGL.
FX2D (JavaFX 2D): Penyaji 2D yang menggunakan JavaFX, yang mungkin lebih cepat untuk beberapa aplikasi, tetapi memiliki beberapa kebiasaan kompatibilitas.
PDF : Penyaji PDF menggambar grafik 2D langsung ke file Acrobat PDF. Ini menghasilkan hasil yang sangat baik ketika Anda membutuhkan bentuk vektor untuk output atau pencetakan resolusi tinggi. Anda harus terlebih dahulu menggunakan Impor Perpustakaan → PDF untuk menggunakan perpustakaan. Informasi lebih lanjut dapat ditemukan di referensi perpustakaan PDF.
SVG: Perender SVG menggambar grafik 2D langsung ke file SVG. Ini bagus untuk mengimpor ke program vektor lain atau menggunakan untuk fabrikasi digital. Anda harus terlebih dahulu menggunakan Impor Perpustakaan → Ekspor SVG untuk menggunakan perpustakaan.

Pada Processing 3.0, untuk menggunakan variabel sebagai parameter ke size () fungsi, tempatkan fungsi size () dalam fungsi settings () (alih-alih settings ()). Ada informasi lebih lanjut tentang ini di halaman referensi settings () .

Sintak
size(width, height)
size(width, height, renderer)

Parameter-parameter
Width int: lebar jendela tampilan dalam satuan piksel.
Height int: tinggi jendela tampilan dalam satuan piksel

smooth()

Examples

```
void setup() {
  size(100, 100);
  smooth(2);
  noStroke();
}

void draw() {
  background(0);
  ellipse(30, 48, 36, 36);
  ellipse(70, 48, 36, 36);
}
void setup() {
  fullScreen(P2D, SPAN);
  smooth(4);
}
```

```
void draw() {
  background(0);
  ellipse(x, height/2, height/4, height/4);
  x += 1;
}
```

Diskripsi

Gambar semua geometri dengan tepi halus (anti-alias). Perilaku ini adalah default, jadi smooth () hanya perlu digunakan ketika sebuah program perlu mengatur smoothing dengan cara yang berbeda. The tingkat parameter meningkatkan tingkat kehalusan. Ini adalah tingkat pengambilan sampel berlebih yang diterapkan pada buffer grafis.

Dengan penyaji P2D dan P3D, smooth (2) adalah default, ini disebut "2x anti-aliasing." Kode smooth (4) digunakan untuk 4x anti-aliasing dan smooth (8) ditentukan untuk 8x anti-aliasing. Level anti-aliasing maksimum ditentukan oleh perangkat keras mesin yang menjalankan perangkat lunak, sangat smooth (4) dan smooth (8)tidak akan bekerja dengan setiap komputer.

Renderer default menggunakan smooth (3) secara default. Ini adalah pemulusan bikubik. Opsi lain untuk perender default adalah smooth (2) , yang merupakan perataan bilinear.

Dengan Processing 3.0, smooth () berbeda dari sebelumnya. Itu umum untuk menggunakan smooth () dan noSmooth () untuk menghidupkan dan mematikan antialiasing dalam sketsa. Sekarang, karena perubahan perangkat lunak, smooth () hanya dapat diatur satu kali dalam sketsa. Ini dapat digunakan baik di bagian atas sketsa tanpa fungsi setup () , atau setelah size () ketika digunakan dalam sketsa dengan setup () . ItuFungsi noSmooth () juga mengikuti aturan yang sama.

Sintak

smooth(level)

Parameter-parameter

Level int: baik 2, 3, 4, atau 8 tergantung pada penyaji

width

Examples

```
noStroke();
background(0);
rect(0, 40, width, 20);
rect(0, 60, width/2, 20);
```

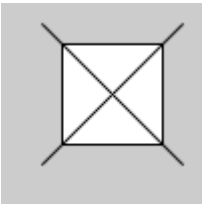
Diskripsi

Variabel sistem yang menyimpan lebar jendela tampilan. Nilai ini ditentukan oleh parameter pertama dari fungsi size () . Misalnya, ukuran panggilan fungsi (320, 240) menetapkan variabel lebar ke nilai 320. Nilai lebar standar menjadi 100 jika size () tidak digunakan dalam suatu program.

BAB V
DATA

boolean

Examples



```
boolean a = false;
if (!a) {
  rect(30, 20, 50, 50);
}
a = true;
if (a) {
  line(20, 10, 90, 80);
  line(20, 80, 90, 10);
}
```

Diskripsi

Datatype untuk nilai-nilai Boolean benar dan salah . Adalah umum untuk menggunakan nilai boolean dengan pernyataan kontrol untuk menentukan aliran suatu program. Pertama kali sebuah variabel ditulis, itu harus dideklarasikan dengan pernyataan yang mengekspresikan datatype-nya.

Sintak

boolean var
boolean var = booleanvalue

Parameter-parameter

Var boolean var boolean var = booleanvalue
Booleanvalue benar atau salah

byte

Examples

```
// Nyatakan variabel 'a' dari byte tipe
byte a;

// Tetapkan 23 untuk 'a'
a = 23;

// Deklarasikan variabel 'b' dan berikan nilai -128
byte b = -128;

// Nyatakan variabel 'c' dan tetapkan jumlahnya sebagai 'a' dan 'b'.
// Secara default, ketika dua byte ditambahkan, mereka dikonversi
// ke integer. Untuk menyimpan jawaban sebagai byte, masukkan mereka
// ke byte dengan fungsi konversi byte ()
byte c = byte (a + b);
```

Diskripsi

Datatype untuk byte, 8 bit informasi yang menyimpan nilai numerik dari 127 ke -128. Bytes adalah tipe data yang mudah untuk mengirim informasi ke dan dari port serial dan untuk mewakili huruf dalam format yang lebih sederhana daripada tipe data char . Pertama kali sebuah variabel ditulis, itu harus dideklarasikan dengan pernyataan yang mengekspresikan datatype-nya. Penggunaan variabel ini selanjutnya tidak boleh merujuk pada tipe data karena Pemrosesan akan berpikir variabel tersebut dideklarasikan lagi.

Sintak
byte var
byte var = value

Parameter-parameter

Var	nama variabel yang merujuk nilai
Value	angka antara 127 hingga -128

Char

Examples
char m; // Nyatakan variabel 'm' dari tipe char
m = 'A'; // Tetapkan nilai "A"
int n = '&'; // Nyatakan variabel 'n' dan berikan nilai "&"

Diskripsi
Tipe data untuk karakter, simbol tipografi seperti A, d, dan $. Sebuah Char toko huruf dan simbol dalam format Unicode, sistem pengkodean dikembangkan untuk mendukung berbagai bahasa dunia. Setiap karakter memiliki panjang dua byte (16 bit) dan dibedakan dengan mengelilinginya dengan tanda kutip tunggal. Karakter lolos juga dapat disimpan sebagai char . Misalnya, representasi untuk kunci "hapus" adalah 127. Pertama kali sebuah variabel ditulis, ia harus dideklarasikan dengan pernyataan yang mengekspresikan datatype-nya. Penggunaan variabel ini selanjutnya tidak boleh merujuk pada tipe data karena Pemrosesan akan berpikir variabel tersebut dideklarasikan lagi.

Sintak
char var
char var = value

Data

Parameter-parameter

Var	nama variabel yang merujuk nilai
Value	karakter apapun

Color

Examples

```
color c1 = color(204, 153, 0);
color c2 = #FFCC00;
noStroke();
fill(c1);
rect(0, 0, 25, 100);
fill(c2);
rect(25, 0, 25, 100);
color c3 = get(10, 50);
fill(c3);
rect(50, 0, 50, 100);
```

Diskripsi

Datatype untuk menyimpan nilai warna. Warna dapat ditetapkan dengan get () dan warna () atau mereka dapat ditentukan secara langsung menggunakan notasi heksadesimal seperti # FFCC00 atau 0xFFFFCCOO .

Menggunakan print () atau println () pada warna akan menghasilkan hasil yang aneh (biasanya angka negatif) karena cara warna disimpan dalam memori. Teknik yang lebih baik adalah dengan menggunakan fungsi hex () untuk memformat data warna, atau menggunakan fungsi merah () , hijau () , dan biru () untuk mendapatkan nilai individual dan mencetaknya. The hue () , saturasi () , danbrightness () berfungsi dengan cara

yang sama. Untuk mengekstraksi nilai merah, hijau, dan biru lebih cepat (misalnya saat menganalisis gambar atau bingkai video), gunakan bit shifting .

Nilai juga dapat dibuat menggunakan notasi warna web. Sebagai contoh: color c = # 006699
Notasi warna web hanya berfungsi untuk warna buram. Untuk menentukan warna dengan nilai alfa, Anda bisa menggunakan warna () atau notasi heksadesimal. Untuk notasi heksa, awali nilai dengan 0x , misalnya warna c = 0xCC006699 . Dalam contoh itu, CC(nilai hex 204) adalah nilai alpha, dan sisanya identik dengan warna web. Perhatikan bahwa dalam notasi heksadesimal, nilai alfa muncul di posisi pertama. (Nilai alfa muncul terakhir ketika digunakan dalam warna () , isian () , dan guratan () .) Berikut ini adalah cara yang setara untuk menulis contoh sebelumnya, tetapi menggunakan warna () dan menetapkan nilai alfa sebagai parameter kedua: color c = color (# 006699, 191)
Dari sudut pandang teknis, warna adalah 32 bit informasi yang dipesan sebagaI AAAAAAARARRRRRRRGGGGGGGGBBBBBBBB di mana A berisi nilai alfa, R adalah nilai merah, G adalah hijau, dan B adalah biru. Setiap komponen adalah 8 bit (angka antara 0 dan 255). Nilai-nilai ini dapat dimanipulasi dengan pergeseran bit .

double

Examples

```
double a;   // Nyatakan variabel 'a' dari tipe double
a = 1.5387D // Tetapkan 'a' nilai 1,5387
```

double b = -2.984D; // Deklarasikan variabel 'b' dan berikan nilai -2,984,
double c = a + b; // Nyatakan variabel 'c' dan tetapkan jumlah dari 'a' dan 'b'
float f = (float)c // Mengkonversi nilai 'c' dari dobel menjadi pelampung

Diskripsi
Datatype untuk angka floating-point lebih besar dari pada yang dapat disimpan dalam float . Sebuah mengambang adalah 32-bit nilai-nilai yang dapat sebagai besar sebagai 3.40282347E + 38 dan serendah -3.40282347E + 38. Sebuah ganda dapat digunakan mirip dengan pelampung, tetapi dapat memiliki besarnya lebih besar karena sejumlah 64-bit. Fungsi pemrosesan tidak menggunakan datatype ini, jadi saat mereka bekerja dalam bahasa, Anda biasanya harus mengonversi ke float menggunakan sintaks (float) sebelum beralih ke fungsi

Sintak
double var
double var = value

Parameter-parameter

Var	nama variabel referensi pelampung
Value	nilai floating-point

float

Examples

float a; // Nyatakan variabel 'a' dari tipe float
a = 1.5387; // Tetapkan 'a' nilai 1,5387

```
float b = -2.984; // Deklarasikan variabel 'b' dan berikan nilai -2.984
float c = a + b; // Nyatakan variabel 'c' dan tetapkan jumlah 'a' dan 'b'
float f1 = 0,0;
for (int i = 0; i <100000; i ++) {
  f1 = f1 + 0,0001; // Ide buruk! Lihat di bawah.
}
println (f1);

float f2 = 0,0;
for (int i = 0; i <100000; i ++) {
  // Variabel 'f2' akan bekerja lebih baik di sini, lebih sedikit terpengaruh oleh pembulatan
  f2 = i / 1000.0; // Hitung dengan seperseribu
}
println (f2)
```

Diskripsi

Jenis data untuk angka floating-point, misalnya angka yang memiliki titik desimal.

Mengapung tidak tepat, jadi menambahkan nilai-nilai kecil (seperti 0,0001) mungkin tidak selalu bertambah tepat karena kesalahan pembulatan. Jika Anda ingin menambah nilai dalam interval kecil, gunakan int , dan bagi dengan nilai float sebelum menggunakannya.

Angka floating-point bisa sebesar 3,40282347E + 38 dan serendah -3,40282347E + 38. Mereka disimpan sebagai 32 bit (4 byte) informasi. The mengapung tipe data diturunkan dari Jawa.

Pemrosesan mendukung gandadatatype dari Java juga. Namun, tidak ada fungsi Pemrosesan yang menggunakan nilai ganda , yang menggunakan lebih banyak memori dan biasanya

berlebihan untuk sebagian besar pekerjaan yang dibuat dalam Pemrosesan. Kami tidak berencana untuk menambahkan dukungan untuk nilai ganda , karena hal itu akan membutuhkan peningkatan jumlah fungsi API secara signifikan.

Sintak
float var
float var = value

Parameter-parameter

Var	nama variabel referensi float
Value	nilai floating-point

int

Examples

```
int a; // Nyatakan variabel 'a' dari tipe int
a = 23; // Tetapkan 'a' nilai 23
int b = -256; // Deklarasikan variabel 'b' dan berikan nilai -256
int c = a + b; // Nyatakan variabel 'c' dan tetapkan jumlah 'a' dan 'b'
```

Diskripsi
Datatype untuk bilangan bulat, angka tanpa titik desimal. Bilangan bulat bisa sebesar 2.147.483.647 dan serendah -2.147.483.648. Mereka disimpan sebagai 32 bit informasi. Pertama kali sebuah variabel ditulis, itu harus dideklarasikan dengan pernyataan yang mengekspresikan datatype-nya. Penggunaan variabel ini selanjutnya tidak boleh merujuk pada tipe data karena Pemrosesan akan berpikir variabel tersebut dideklarasikan lagi.

Sintak
int var
int var = value

Parameter-parameter

Var	nama variabel yang merujuk nilai
Value	nilai integer apapun

Long

Examples

long a; // Deklarasikan variabel 'a' dengan tipe long dan berikan nilai besar:
// a = 2147483648; // Kesalahan: Literal dari tipe int berada di luar jangkauan
a = 2147483648L; // Sebagai gantinya, tambahkan "L" ke nomor untuk menandainya sebagai panjang

long b = -256; // Deklarasikan variabel 'b' dan berikan nilai -256
long c = a + b; // Nyatakan variabel 'c' dan tetapkan jumlah 'a' dan 'b'
int i = (int) c; // Mengkonversi nilai 'c' dari panjang ke int

Diskripsi
Datatype untuk bilangan bulat besar. Sementara bilangan bulat dapat sebesar 2.147.483.647 dan serendah -2.147.483.648 (disimpan sebagai 32 bit), bilangan bulat panjang memiliki nilai minimum -9.223.372.036.854.775.808 dan nilai maksimum 9.223.372.036.854.775.807 (disimpan sebagai 64 bit).

Gunakan tipe data ini saat Anda membutuhkan nomor untuk memiliki besaran lebih besar daripada yang dapat disimpan dalam sebuah int . Saat menetapkan nilai literal yang lebih besar dari besarnya ini, perlu juga menambahkan kualifikasi "L" ke angka, seperti yang ditunjukkan pada contoh di atas. Fungsi pemrosesan tidak menggunakan tipe data ini, jadi saat mereka bekerja dalam bahasa tersebut, Anda biasanya harus mengonversi ke int menggunakan sintaks (int) sebelum beralih ke fungsi.

Sintak

long var
long var = value

Parameter-parameter

Var	nama variabel yang merujuk nilai
Value	nilai integer apa pun

BAB VI
COMPOSITE

Array

Example

```
int[] angka = new int[3];
angka[0] = 90;  // Tetapkan nilai pertama dalam array
angka[1] = 150; // Tetapkan nilai kedua dalam array
angka[2] = 30;  // Tetapkan nilai ketiga dalam array
int a = angka[0] + angka[1]; // Tetapkan variabel 'a' ke 240
int b = angka[1] + angka[2]; // Tetapkan variabel 'b' ke 180

int [] angka = {90, 150, 30}; // Sintaks alternatif
int a = angka [0] + angka [1]; // Tetapkan variabel 'a' ke 240
int b = angka [1] + angka [2]; // Tetapkan variabel 'b' ke 180

int degrees = 360;
float[] cos_vals = new float[degrees];
// Gunakan loop for () untuk dengan cepat beralih
// melalui semua nilai dalam array.

for (int i=0; i < degrees; i++) {
  cos_vals[i] = cos(TWO_PI/degrees * i);
```

Diskripsi

Array adalah daftar data. Dimungkinkan untuk memiliki berbagai jenis data. Setiap data dalam array diidentifikasi oleh nomor indeks yang mewakili posisinya dalam array. Elemen pertama dalam array adalah [0] , elemen kedua adalah [1] , dan seterusnya. Array mirip dengan objek, sehingga harus dibuat dengan kata kunci baru .

Setiap array memiliki panjang variabel , yang merupakan nilai integer untuk jumlah total elemen dalam array. Perhatikan bahwa sejak penomoran indeks dimulai dari nol (bukan 1), nilai terakhir dalam array dengan panjang 5 harus dirujuk sebagai

array [4] (yaitu, panjang minus 1), bukan array [5], yang akan memicu kesalahan.
Sumber kebingungan umum lainnya adalah perbedaan antara menggunakan panjang untuk mendapatkan ukuran array dan panjang () untuk mendapatkan ukuran String. Perhatikan keberadaan tanda kurung ketika bekerja dengan Strings. (array.length adalah variabel, sedangkan String.length () adalah metode khusus untuk kelas String.)

Sintak
datatype[] var
var[element] = value
var.length

Parameter-parameter
Datatype setiap tipe data primitif atau majemuk, termasuk kelas yang ditentukan pengguna
Var nama variabel apa pun yang valid
Element int: tidak boleh melebihi panjang array minus 1
Value data untuk ditetapkan ke elemen array; harus datatype yang sama dengan array

ArrayList

Examples
// Ini adalah fragmen kode yang menunjukkan cara menggunakan ArrayList.
// Mereka tidak akan mengkompilasi karena mereka menganggap keberadaan kelas Partikel.

// Deklarasikan ArrayList, catat penggunaan sintaks "<Particle>" untuk mengindikasikan

```
// niat kami untuk mengisi ArrayList ini dengan objek Partikel
ArrayList <Partikel> partikel = ArrayList baru <Partikel> ();

// Objek dapat ditambahkan ke ArrayList dengan add ()
partikel.add (Partikel baru ());

// Partikel dapat ditarik dari ArrayList dengan get ()
Particle part =icles.get (0);
part.display ();

// Metode size () mengembalikan jumlah item saat ini dalam daftar
int total = particles.size ();
println ("Jumlah total partikel adalah:" + total);

// Anda dapat beralih menggunakan ArrayList dengan dua cara.
// Yang pertama adalah dengan menghitung melalui elemen:
untuk (int i = 0; i <partic.size (); i ++) {
  Particle part =icles.get (i);
  part.display ();
}

// Yang kedua menggunakan loop yang disempurnakan:
untuk (Bagian partikel: partikel) {
  part.display ();
}
// Anda dapat menghapus partikel dari ArrayList dengan remove ()
partikel.hapus (0);
println (partikel. ukuran ()); // Sekarang kurang satu!

// Jika Anda memodifikasi ArrayList selama loop,
// maka Anda tidak dapat menggunakan sintaks loop ditingkatkan.
```

```
// Selain itu, saat menghapus untuk mengenai semua elemen,
// Anda harus mengulanginya, seperti yang ditunjukkan di sini:
untuk (int i = particles.size () - 1; i> = 0; i--) {
  Particle part =icles.get (i);
  if (part.finished ()) {
    partikel.hapus (i);
  }
}
```

Diskripsi

Sebuah ArrayList menyimpan sejumlah variabel obyek. Ini mirip dengan membuat array objek, tetapi dengan ArrayList , item dapat dengan mudah ditambahkan dan dihapus dari ArrayList dan diubah ukurannya secara dinamis. Ini bisa sangat nyaman, tetapi lebih lambat daripada membuat array objek saat menggunakan banyak elemen. Perhatikan bahwa untuk daftar bilangan bulat, float, dan String yang dapat diubah ukurannya, Anda dapat menggunakan kelas pemrosesan IntList, FloatList, dan StringList.

ArrayList adalah implementasi array-resizable dari antarmuka Daftar Java. Ini memiliki banyak metode yang digunakan untuk mengontrol dan mencari isinya. Misalnya, panjang ArrayList dikembalikan oleh ukurannya ()metode, yang merupakan nilai integer untuk jumlah total elemen dalam daftar. Elemen ditambahkan ke ArrayList dengan metode add () dan dihapus dengan metode remove () . Metode get () mengembalikan elemen pada posisi yang ditentukan dalam daftar.

Constructor

ArrayList<Type>()
ArrayList<Type>(initialCapacity)

Parameter-parameter

Type Nama Kelas: tipe data untuk objek yang akan ditempatkan di ArrayList.

initialCapacity int: mendefinisikan kapasitas awal daftar; itu kosong secara default

FloatList

Examples

```
FloatList inventory;
void setup() {
  size(200, 200);
  inventory = new FloatList();
  inventory.append(108.6);
  inventory.append(5.8);
  inventory.append(8.2);
  println(inventory);
  noLoop();
  fill(0);
  textAlign(CENTER);
}

void draw() {
  float nums = inventory.get(2);
  text(nums, width/2, height/2);
}
```

Diskripsi

Kelas pembantu untuk daftar pelampung. Daftar dirancang untuk memiliki beberapa fitur ArrayLists, tetapi untuk menjaga kesederhanaan dan efisiensi bekerja dengan array.

Fungsi seperti sort () dan shuffle () selalu bertindak pada daftar itu sendiri. Untuk mendapatkan salinan yang diurutkan, gunakan list.copy (). Sort ().

Method

size() Untuk mendapatkan daftar panjang
clear() Menghapus semua data dari list
get() Dapatkan entri di indeks tertentu
set() Atur entri pada indeks tertentu
remove() Hapus elemen dari indeks yang ditentukan
append() Tambahkan entri baru ke daftar
hasValue() Periksa apakah nomor merupakan bagian dari daftar
add() Menambahkan nilai
sub() Kurangi dari suatu nilai
mult() Lipat gandakan nilainya
div() Bagi nilai
min() Mengembalikan nilai terkecil
max() Mengembalikan nilai terbesar
sort() Mengurutkan array, terendah ke tertinggi

Constructor
FloatList ()
FloatList (items)

HashMap

Examples
import java.util.Map;

```
// Perhatikan "kunci" HashMap adalah sebuah String dan "nilai" adalah Integer
 HashMap<String,Integer> hm = new HashMap<String,Integer>();
// Memasukkan pasangan nilai kunci di HashMap
 hm.put("Ava", 1);
hm.put("Cait", 35);
hm.put("Casey", 36);
// Menggunakan loop yang ditingkatkan untuk beralih setiap entri
 for (Map.Entry me : hm.entrySet()) {
  print(me.getKey() + " is ");
  println(me.getValue());
}
// Kita juga bisa mengakses nilai dengan kunci mereka
 int val = hm.get("Casey");
println("Casey is " + val);
```

Diskripsi

Sebuah HashMap menyimpan koleksi benda-benda, masing-masing direferensikan oleh kunci. Ini mirip dengan Array , hanya alih-alih mengakses elemen dengan indeks numerik, sebuah String digunakan. (Jika Anda terbiasa dengan array asosiatif dari bahasa lain, ini adalah ide yang sama.) Contoh di atas mencakup penggunaan dasar, tetapi ada contoh yang lebih luas yang disertakan dengan contoh Pemrosesan. Selain itu, untuk pemasangan String dan integer sederhana, String dan float, atau String dan String, Anda sekarang dapat menggunakan kelas IntDict, FloatDict, dan StringDict yang lebih sederhana.

Constructor

HashMap<Key, Value>()
HashMap<Key, Value>(initialCapacity)

HashMap<Key, Value>(initialCapacity, loadFactor)
HashMap<Key, Value>(m)

Parameter-parameter
Key Nama Kelas: tipe data untuk kunci HashMap
Value Nama Kelas: tipe data untuk nilai-nilai HashMap

Initialcapacity int: mendefinisikan kapasitas awal peta; standarnya adalah 16

Loadfactor float: faktor muatan untuk peta; standarnya adalah 0,75
M Peta: memberikan pemetaan HashMap baru yang sama dengan Peta ini

IntDict

Examples

```
ntDict inventory;
void setup() {
  size(200, 200);
  inventory = new IntDict();
  inventory.set("cd", 84);
  inventory.set("tapes", 15);
  inventory.set("records", 102);
  println(inventory);
  noLoop();
  fill(0);
  textAlign(CENTER);
}

void draw() {
  int numRecords = inventory.get("records");
```

```
  text(numRecords, width/2, height/2);
}
```

Diskripsi

Kelas sederhana untuk menggunakan String sebagai pencarian untuk nilai int. String "kunci" dikaitkan dengan nilai integer.

Method

size() Mengembalikan jumlah pasangan kunci / nilai
clear() Mengembalikan jumlah pasangan kunci / nilai
keyArray() Kembalikan salinan array kunci internal
values() Kembalikan array internal yang digunakan untuk menyimpan nilai
valueArray() Buat array baru dan salin setiap nilai ke dalamnya

get() Kembalikan nilai untuk kunci yang ditentukan
set() Buat pasangan kunci / nilai baru atau ubah nilainya
hasKey() Periksa apakah kunci adalah bagian dari struktur data
increment() Tambah nilai nilai kunci tertentu sebesar 1
add() Tambahkan ke nilai
sub() Kurangi dari suatu nilai
mult() Lipat gandakan nilainya
div() Bagi nilai
remove() Hapus pasangan kunci / nilai
sortKeys() Sortir kunci menurut abjad
sortKeysReverse() Sortir kunci secara abjad secara terbalik
sortValues() Urutkan berdasarkan nilai dalam urutan menaik
sortValuesReverse() Urutkan berdasarkan nilai dalam urutan

menurun

Constructor
IntDict()
IntDict(pairs)

IntList

Examples

```
IntList inventory;
void setup() {
  size(200, 200);
  inventory = new IntList();
  inventory.append(84);
  inventory.append(15);
  inventory.append(102);
  println(inventory);
  noLoop();
  fill(0);
  textAlign(CENTER);
}

void draw() {
  int nums = inventory.get(2);
  text(nums, width/2, height/2);
}
```

Diskripsi
Kelas pembantu untuk daftar int. Daftar dirancang untuk memiliki beberapa fitur ArrayLists, tetapi untuk menjaga kesederhanaan dan efisiensi bekerja dengan array.

Fungsi seperti sort () dan shuffle () selalu bertindak pada daftar itu sendiri. Untuk mendapatkan salinan yang diurutkan, gunakan list.copy (). Sort ().

Method

size()	Dapatkan panjang daftar
clear()	Hapus semua entri dari daftar
get()	Dapatkan entri di indeks tertentu
set()	Atur entri pada indeks tertentu
remove()	Hapus elemen dari indeks yang ditentukan
append()	Tambahkan entri baru ke daftar
hasValue()	Periksa apakah nomor merupakan bagian dari daftar
increment()	Tambahkan satu ke nilai
add()	Tambahkan ke nilai
sub()	Kurangi dari suatu nilai
mult()	Lipat gandakan nilainya
div()	Bagi nilai
min()	Kembalikan nilai terkecil
max()	Kembalikan nilai terbesar
sort()	Urutkan array, terendah ke tertinggi
sortReverse()	Urut terbalik, nilai pesanan dari tertinggi ke terendah
reverse()	Membalik urutan elemen daftar
shuffle()	Acak urutan elemen daftar
array()	Buat array baru dengan salinan semua nilai

JSONArray

Examples

String[] species = { "Capra hircus", "Panthera pardus", "Equus zebra" };

```
String[]names = { "Goat", "Leopard", "Zebra" };

JSONArray values;

void setup() {

  values = new JSONArray();

  for (int i = 0; i < species.length; i++) {

    JSONObject animal = new JSONObject();

    animal.setInt("id", i);
    animal.setString("species", species[i]);
    animal.setString("name", names[i]);

    values.setJSONObject(i, animal);
  }

  saveJSONArray(values, "data/new.json");
}
// Sketch menyimpan yang berikut ke file yang disebut "new.json":
// [
// {
// "id": 0,
// "spesies": "Capra hircus",
// "nama": "Kambing"
//},
// {
// "id": 1,
// "spesies": "Panthera pardus",
// "nama": "Leopard"
//},
```

```
// {
// "id": 2,
// "spesies": "Equus zebra",
// "nama": "Zebra"
//}
//]
```

Diskripsi

Sebuah JSONArray menyimpan array obyek JSON. JSONArray s dapat dihasilkan dari awal, secara dinamis, atau menggunakan data dari file yang ada. JSON juga dapat di-output dan disimpan ke disk, seperti pada contoh di atas.

Method	
getString ()	Mendapat nilai String yang terkait dengan indeks
getInt ()	Mendapat nilai int yang terkait dengan indeks
getFloat ()	Mendapat nilai float yang terkait dengan indeks
getBoolean ()	Mendapat nilai boolean yang terkait dengan indeks
getJSONArray ()	Mendapat JSONArray yang terkait dengan nilai indeks
getJSONObject ()	Mendapat objek JSONOb terkait dengan nilai indeks
getStringArray ()	Mendapat seluruh array sebagai array dari Strings
getIntArray ()	Mendapat seluruh array sebagai array int
append()	Menambahkan nilai, menambah panjang array dengan satu
setString ()	Masukkan nilai String di JSONArray
setInt ()	Masukkan nilai int di JSONArray
setFloat ()	Masukkan nilai float di JSONArray

setBoolean ()	Masukkan nilai boolean di JSONArray
setJSONArray ()	Menetapkan nilai JSONArray yang terkait dengan nilai indeks
setJSONObject ()	Menetapkan nilai JSONObject yang terkait dengan nilai indeks
size()	Mendapat jumlah elemen dalam JSONArray
isNull ()	Tentukan apakah nilainya nol
remove()	Menghapus elemen

Constructor

JSONArray()

Object

Examples

```
// Deklarasikan dan buat dua objek (h1, h2) dari kelas HLine
 the class HLine
HLine h1 = new HLine(20, 2.0);
HLine h2 = new HLine(50, 2.5);

void setup()
{
  size(200, 200);
  frameRate(30);
}

void draw() {
  background(204);
  h1.update();
  h2.update();
}

class HLine {
  float ypos, speed;
  HLine (float y, float s) {
    ypos = y;
    speed = s;
  }
  void update() {
    ypos += speed;
    if (ypos > width) {
      ypos = 0;
    }
```


```
    line(0, ypos, width, ypos);
  }
}
```

Diskripsi
Objek adalah instance dari kelas. Kelas adalah pengelompokan metode terkait (fungsi) dan bidang (variabel dan konstanta).

Sintak
ClassName instanceName

Parameter-parameter
ClassName kelas untuk membuat objek baru
instanceName nama untuk objek baru

String

Examples

```
String str1 = "CCCP";
char data[] = {'C', 'C', 'C', 'P'};
String str2 = new String(data);
println(str1);  // Prints "CCCP" ke konsol
println(str2);  // Prints "CCCP" ke konsol
// Membandingkan objek String, lihat referensi di bawah.
String p = "potato";
if (p == "potato") {
  println("p == potato, yep.");  // This will not print
}

if (p.equals("potato")) {
  println("Yes, the contents of p and potato are the same.");
```

```
}
// Gunakan backslash untuk memasukkan tanda kutip ke dalam String.
String quoted = "This one has \"quotes\"";
println(quoted);  // This one has "quotes"
```

Diskripsi

String adalah urutan karakter. String kelas mencakup metode untuk memeriksa masing-masing karakter, membandingkan string, mencari string, mengekstraksi bagian string, dan untuk mengkonversi seluruh string huruf besar dan kecil. String selalu didefinisikan di dalam tanda kutip ganda ("Abc"), dan karakter selalu didefinisikan di dalam tanda kutip tunggal ('A').

Untuk membandingkan konten dua String, gunakan metode equals () , seperti pada if (a.equals (b)) , alih-alih if (a == b) . String adalah Object, jadi membandingkannya dengan operator == hanya membandingkan apakah kedua String disimpan di lokasi memori yang sama. Menggunakan equals ()Metode akan memastikan bahwa konten yang sebenarnya dibandingkan. (Referensi pemecahan masalah memiliki penjelasan yang lebih panjang.)

Karena String didefinisikan antara tanda kutip ganda, untuk memasukkan tanda tersebut dalam String itu sendiri, Anda harus menggunakan karakter \ (backslash). (Lihat contoh ketiga di atas.) Ini dikenal sebagai urutan pelarian . Urutan melarikan diri lainnya termasuk \ t untuk karakter tab dan \ n untuk baris baru. Karena backslash adalah karakter pelarian, untuk menyertakan backslash tunggal dalam sebuah String, Anda harus menggunakan dua backslash berturut-turut, seperti pada: \\

Ada lebih banyak metode string daripada yang ditautkan dari

halaman ini. Dokumentasi tambahan terletak online di Internetdokumentasi resmi Java .

Method
Character ()Mengembalikan karakter pada indeks yang ditentukan
Equal ()Membandingkan string ke objek tertentu
Indeks()Mengembalikan nilai indeks kejadian pertama substring dalam string input
length()Mengembalikan jumlah karakter dalam string input
substring ()Mengembalikan string baru yang merupakan bagian dari string input
toLowerCase ()Mengubah semua karakter menjadi huruf kecil
toUpperCase ()Mengubah semua karakter menjadi huruf besar

Constructor
String(data)
String(data, offset, length)

Parameter-parameter
data byte [] atau char []: array byte yang akan diterjemahkan ke dalam karakter, atau array karakter yang akan digabungkan menjadi string
offset int: indeks karakter pertama
length int: jumlah karakter

StringDict
Examples

StringDict inventory;

```
void setup() {
  size(200, 200);
  inventory = new StringDict();
  inventory.set("coffee","black");
  inventory.set("flour","white");
  inventory.set("tea","green");
  println(inventory);
  noLoop();
  fill(0);
  textAlign(CENTER);
}

void draw() {
  String s = inventory.get("tea");
  text(s, width/2, height/2);
}
```

Diskripsi

Kelas sederhana untuk menggunakan String sebagai pencarian untuk nilai String. String "kunci" dikaitkan dengan nilai-nilai String.

Method

size() Mengembalikan jumlah pasangan kunci / nilai

clear() Hapus semua entri

keyArray () Kembalikan salinan array kunci internal

value () Kembalikan array internal yang digunakan untuk menyimpan nilai

valueArray () Buat array baru dan salin setiap nilai ke dalamnya

get()	Kembalikan nilai untuk kunci yang ditentukan
set()	Buat pasangan kunci / nilai baru atau ubah nilainya
hasKey ()	Periksa apakah kunci adalah bagian dari struktur data
remove()	Hapus pasangan kunci / nilai
sortKeys ()	Sortir kunci menurut abjad
sortKeysReverse ()	Sortir kunci secara abjad secara terbalik
sortValues ()	Urutkan berdasarkan nilai dalam urutan menaik
sortValuesReverse ()	Urutkan berdasarkan nilai dalam urutan menurun

Constructor

StringDict()
StringDict(pairs)
StringDict(row)

StringList

Examples

```
StringList inventory;
void setup() {
  size(200, 200);
  inventory = new StringList();
  inventory.append("coffee");
  inventory.append("flour");
  inventory.append("tea");
  println(inventory);
  noLoop();
  fill(0);
```

```
  textAlign(CENTER);
}
void draw() {
  String item = inventory.get(2);
  text(item, width/2, height/2);
}
```

Diskripsi

Kelas pembantu untuk daftar Strings. Daftar dirancang untuk memiliki beberapa fitur ArrayLists, tetapi untuk menjaga kesederhanaan dan efisiensi bekerja dengan array.

Fungsi seperti sort () dan shuffle () selalu bertindak pada daftar itu sendiri. Untuk mendapatkan salinan yang diurutkan, gunakan list.copy (). Sort ().

Method

size()	Dapatkan panjang daftar
clear()	Hapus semua entri dari daftar
append()	Dapatkan entri di indeks tertentu
set()	Atur entri pada indeks tertentu
remove()	Hapus elemen dari indeks yang ditentukan
append()	Tambahkan entri baru ke daftar
hasValue ()	Periksa apakah suatu nilai adalah bagian dari daftar
sort()	Urutkan array pada tempatnya
sortReverse ()	Urut terbalik, nilai pesanan dari tertinggi ke terendah
Reverse()	Membalik urutan elemen daftar
shuffle ()	Acak urutan elemen daftar
lower()	Jadikan seluruh daftar huruf kecil
upper()	Jadikan seluruh daftar huruf besar
array ()	Buat array baru dengan salinan semua nilai

Constructor
StringList()
StringList(items)

Table

Examples

```
Table table;
void setup() {
  table = new Table();

  table.addColumn("id");
  table.addColumn("species");
  table.addColumn("name");

  TableRow newRow = table.addRow();
  newRow.setInt("id", table.lastRowIndex());
  newRow.setString("species", "Panthera leo");
  newRow.setString("name", "Lion");

  saveTable(table, "data/new.csv");
}

// Sketch saves the following to a file called "new.csv":
// id,species,name
// 0,Panthera leo,Lion
```

Diskripsi
Objek tabel menyimpan data dengan banyak baris dan kolom, seperti di spreadsheet tradisional. Tabel dapat dihasilkan dari awal, secara dinamis, atau menggunakan data dari file yang

ada. Tabel juga dapat di-output dan disimpan ke disk, seperti pada contoh di atas. Metode Tabel

Method

addColumn ()Menambahkan kolom baru ke table
removeColumn ()Menghapus kolom dari table
getColumnCount ()Mendapat jumlah kolom dalam sebuah table
getRowCount ()Mendapat jumlah baris dalam sebuah table
clearRows ()Menghapus semua baris dari sebuah table
AddRows() Menambahkan baris ke table
removeRow () Menghapus baris dari table
getRow () Mendapat baris dari table
line () Mendapat banyak baris dari sebuah table
getInt () Dapatkan nilai integer dari baris dan kolom yang ditentukan
setInt () Menyimpan nilai integer di baris dan kolom yang ditentukan
getFloat () Dapatkan nilai float dari baris dan kolom yang ditentukan
setFloat () Menyimpan nilai float di baris dan kolom yang ditentukan
getString () Dapatkan nilai String dari baris dan kolom yang ditentukan
setString () Menyimpan nilai String di baris dan kolom yang ditentukan
getStringColumn () Mendapat semua nilai di kolom yang ditentukan
findRow () Menemukan baris yang berisi nilai yang diberikan
findRows () Menemukan beberapa baris yang berisi nilai yang diberikan
matchRow () Menemukan baris yang cocok dengan ekspresi yang diberikan

matchRows ()	Menemukan beberapa baris yang cocok dengan ekspresi yang diberikan
removeTokens ()	Menghapus karakter dari table
trim()	Potong spasi putih dari nilai
sort()	Memesan tabel berdasarkan nilai dalam kolom

Constructor
Table()
Table(rows)

TableRow

Examples

```
Table table;
void setup() {
  table = new Table();

  table.addColumn("number", Table.INT);
  table.addColumn("mass", Table.FLOAT);
  table.addColumn("name", Table.STRING);
  TableRow row = table.addRow();
  row.setInt("number", 8);
  row.setFloat("mass", 15.9994);
  row.setString("name", "Oxygen");
  println(row.getInt("number"));   // Prints 8
  println(row.getFloat("mass"));   // Prints 15.9994
  println(row.getString("name"));  // Prints "Oxygen
}
```

Diskripsi
Sebuah TableRow objek mewakili satu baris dari nilai-nilai data, yang disimpan dalam kolom, dari meja. Metode TableRow.

Method

getString ()	Dapatkan nilai String dari kolom yang ditentukan
getInt ()	Dapatkan nilai integer dari kolom yang ditentukan
getFloat ()	Dapatkan nilai float dari kolom yang ditentukan
setString ()	Menyimpan nilai String di kolom yang ditentukan
setInt ()	Menyimpan nilai integer di kolom yang ditentukan
setFloat ()	Menyimpan nilai float di kolom yang ditentukan
getColumnCount ()	Dapatkan jumlah kolom
getColumnTitle ()	Dapatkan judul kolom

XML

Examples

```
// File XML pendek berikut yang disebut "mammals.xml" diuraikan
// dalam kode di bawah ini. Itu harus ada di folder "data" proyek.

// <? xml version = "1.0"?>
// <mamalia>
```

```
// <animal id = "0" species = "Capra hircus"> Kambing </animal>
// <animal id = "1" spesies = "Panthera pardus"> Leopard </animal>
// <animal id = "2" species = "Equus zebra"> Zebra </animal>
// </mammals>

XML xml;
void setup() {
  xml = loadXML("mammals.xml");
  XML[] children = xml.getChildren("animal");

  for (int i = 0; i < children.length; i++) {
    int id = children[i].getInt("id");
    String coloring = children[i].getString("species");
    String name = children[i].getContent();
    println(id + ", " + coloring + ", " + name);
  }
}
// Cetakan sketsa:
// 0, Capra hircus, Kambing
// 1, Panthera pardus, Leopard
// 2, Equus zebra, Zebra
```

Diskripsi

XML adalah representasi dari objek XML, mampu mem-parsing kode XML. Gunakan loadXML () untuk memuat file XML eksternal dan membuat objek XML .

Hanya file yang disandikan sebagai UTF-8 (atau ASCII biasa) yang diuraikan dengan benar; parameter penyandian di dalam file XML diabaikan.

Method

getParent ()	Mendapat salinan dari elemen induk
getName ()	Mendapat nama lengkap elemen
setName ()	Setel nama elemen
hasChildren ()	Periksa apakah suatu elemen memiliki anak atau tidak
listChildren ()	Mengembalikan nama semua anak sebagai array
getChildren ()	Mengembalikan array yang mengandung semua elemen anak
getChild ()	Mengembalikan elemen anak dengan nilai indeks atau jalur yang ditentukan
addChild ()	Menambahkan anak baru ke elemen
removeChild()	Menghapus anak yang ditentukan
getAttributeCount ()	Menghitung jumlah atribut elemen yang ditentukan
listAttributes ()	Mengembalikan daftar nama semua atribut sebagai array
hasAttribute ()	Cek apakah suatu elemen memiliki atribut yang ditentukan atau tidak
getString ()	Mendapat konten atribut sebagai String
setString ()	Menetapkan konten atribut sebagai String
getInt ()	Mendapat konten atribut sebagai int
setInt ()	Setel konten atribut sebagai int
getFloat ()	Mendapat konten atribut sebagai pelampung
setFloat ()	Setel konten atribut sebagai float
getContent ()	Mendapat konten elemen
getIntContent ()	Mendapat konten elemen sebagai int
getFloatContent ()	Mendapat konten elemen sebagai pelampung
setContent ()	Setel konten suatu elemen
format()	Memformat data XML sebagai String

toString ()	Mendapat data XML sebagai String menggunakan pemformatan default

Constructor
XML(name)

BAB VII
CONVERSION

Binary

Examples

```
color c = color(255, 204, 0);
println(c);              // Prints "-13312"
println(binary(c));              //Prints
"11111111111111111100110000000000"
println(binary(c, 16)); // Prints "1100110000000000"
```

Diskripsi

Mengonversi int , byte , char , atau warna ke String yang berisi notasi biner yang setara. Sebagai contoh, warna nilai yang dihasilkan oleh warna (0, 102, 153, 255) akan dikonversi ke String value "11111111000000000110011010011001" . Fungsi ini dapat membantu membuat sesi debug Anda yang culun jauh lebih bahagia.

Perhatikan bahwa jumlah digit maksimum adalah 32, karena nilai int hanya dapat mewakili hingga 32 bit. Menentukan lebih dari 32 digit tidak akan berpengaruh.

Sintak

biner (value)
binary (value, digits)

Parameter-parameter

Value	nilai char, byte, atau int: untuk mengkonversi
Digit	int: jumlah digit yang akan dikembalikan

boolean()

Examples

```
String s = "true";
boolean b = boolean(s);
if (b) {
  println("The boolean is true");
} else {
  println("The boolean is false");
}
```

Diskripsi

Mengonversi int atau String ke representasi booleannya. Untuk int , nilai bukan nol (positif atau negatif) bernilai true, sedangkan nol bernilai false. Untuk sebuah String , nilai "true" dievaluasi menjadi true, sementara nilai lain (termasuk "false" atau "hippopotamus") dievaluasi menjadi false.

Ketika array nilai int atau String dilewatkan, maka array boolean dengan panjang yang sama dikembalikan.

byte ()

Examples

```
char c = 'E';
byte b = byte (c);
println (c + ":" + b); // Mencetak "E: 69"

int i = 130;
b = byte (i);
println (i + ":" + b); // Mencetak "130: -126"
```

Diskripsi
Mengubah nilai apa pun dari tipe data primitif (boolean , byte , char , color , double , float , int , atau long) ke representasi byte-nya. Byte hanya bisa berupa bilangan bulat antara -128 dan 127 , jadi ketika nilai di luar rentang ini dikonversi, ia membungkus ke representasi byte yang sesuai. (Misalnya, byte (128) mengevaluasi ke -128 .)

Ketika array nilai dilewatkan, maka array byte dengan panjang yang sama dikembalikan.

Float()

Examples
```
int i = 65;
float f = float(i);
println(i + " : " + f);  // Prints "65 : 65.0"
```

Diskripsi
Mengubah int atau String ke representasi floating point-nya. Sebuah int mudah dikonversi ke mengapung , tapi isi dari String harus menyerupai angka, atau NaN (bukan angka) akan dikembalikan. Misalnya, float ("1234.56") bernilai 1234.56 , tetapi float ("jerapah") akan mengembalikan NaN .
Ketika array nilai int atau String dilewatkan, maka array floating point dengan panjang yang sama dikembalikan.

hex ()

Example

```
warna c = # ffcc00;
println (c); // Mencetak "-13312"
println (hex (c)); // Mencetak "FFFFCC00"
println (hex (c, 6)); // Mencetak "FFCC00"

warna c = warna (255, 204, 0);
println (c); // Mencetak "-13312"
println (hex (c)); // Mencetak "FFFFCC00"
println (hex (c, 6)); // Mencetak "FFCC00"
```

Diskripsi

Mengubah int , byte , char , atau warna ke String yang berisi notasi heksadesimal setara. Sebagai contoh, warna nilai yang dihasilkan oleh warna (0, 102, 153) akan dikonversi ke String value "FF006699" . Fungsi ini dapat membantu membuat sesi debug Anda yang culun jauh lebih bahagia.

Perhatikan bahwa jumlah digit maksimum adalah 8, karena nilai int hanya dapat mewakili hingga 32 bit. Menentukan lebih dari 8 digit tidak akan menambah panjang String lebih jauh.

Sintak

```
hex ( value)
hex ( value, digits)
```

Parameter-parameter

Value	int, char, atau byte: nilai untuk dikonversi
Digit	int: jumlah digit (maksimum 8)

int ()

Examples

```
float f = 65.0;
int i = int (f);
println (f + ":" + i); // Mencetak "65.0: 65"

char c = 'E';
i = int (c);
println (c + ":" + i); // Mencetak "E: 69"
```

Diskripsi

Mengubah nilai apa pun dari tipe data primitif (boolean , byte , char , warna , float , int , atau panjang) ke representasi integernya.

Ketika array nilai dilewatkan, maka array int dengan panjang yang sama dikembalikan.

str ()

Examples

```
boolean b = false;
byte y = -28;
char c = 'R';
float f = -32.6;
int i = 1024;
String sb = str (b);
String sy = str (y);
String sc = str (c);
```

```
String sf = str (f);
String si = str (i);

sb = sb + sy + sc + sf + si;
println (sb); // Mencetak 'false-28R-32.61024'
```

Diskripsi

Mengonversi nilai tipe data primitif (boolean , byte , char , int , atau float) ke representasi String -nya . Sebagai contoh, mengkonversi integer dengan str (3) akan mengembalikan String nilai "3" , mengkonversi float dengan str (-12,6) akan kembali "-12,6" , dan mengkonversi boolean dengan str (benar) akan kembali "true " .

Ketika array nilai dilewatkan, maka array String dengan panjang yang sama dikembalikan.

unbinary ()

Examples

```
String s1 = "00010000";
String s2 = "00001000";
String s3 = "00000100";
println (unbinary (s1)); // Mencetak "16"
println (unbinary (s2)); // Mencetak "8"
println (unbinary (s3)); // Mencetak "4"

int i1 = 10000;
int i2 = 1000;
int i3 = 100
println (unbinary (i1)); // Mencetak "16"
println (unbinary (i2)); // Mencetak "8"
println (unbinary (i3)); // Mencetak "4"
```

Diskripsi
Mengubah representasi String dari angka biner ke nilai integer yang setara. Misalnya, unbinary ("00001000") akan mengembalikan 8 . Sebagai alternatif, konversi representasi int dari angka biner ke nilai integer yang setara. Misalnya, unbinary (1000) akan mengembalikan 8 .

Sintak
unbinary (value)

Parameter-parameter
Value String: String untuk mengonversi bilangan bulat

unhex ()

Examples
String hs = "FF006699";
int hi = unhex (hs);
isi (hai);
rect (30, 20, 55, 55);

Diskripsi
Mengonversi representasi String dari angka heksadesimal ke nilai integer yang setara.
Sintak
unhex (value)

Parameter-parameter
Value tring: String untuk mengonversi bilangan bulat

BAB VII
STRING FUNCTIONS

join()

Examples

```
String[] animals = new String[3];
animals[0] = "cat";
animals[1] = "seal";
animals[2] = "bear";
String joinedAnimals = join(animals, " : ");
println(joinedAnimals);  // Prints "cat : seal : bear"

// Bergabung dengan array int perlu terlebih dahulu
// mengonversi ke array dari Strings
int[] numbers = new int[3];
numbers[0] = 8;
numbers[1] = 67;
numbers[2] = 5;
String joinedNumbers = join(nf(numbers, 0), ", ");
println(joinedNumbers);  // Prints "8, 67, 5"
```

Diskripsi

Menggabungkan array dari Strings menjadi satu String, masing-masing dipisahkan oleh karakter yang digunakan untuk parameter separator . Untuk bergabung dengan array int atau float, pertama-tama perlu untuk mengonversinya menjadi String menggunakan nf () atau nfs () .

Sintak

join(list, separator)

Parameter-parameter

List String []: array of Strings

Separator char: char atau String yang akan ditempatkan di antara setiap item

match()

Examples

```
tring s = "Di dalam tag, Anda akan menemukan <tag> konten </tag>.";
String [] m = match (s, "<tag> (. *?) </tag>");
println ("Ditemukan '" + m [1] + "' di dalam tag.");
// Mencetak ke konsol:
// "Ditemukan 'konten' di dalam tag."

String s1 = "Apakah Anda pernah mendengar tentang sesuatu yang disebut fluoridasi.";
      s1 + = "Fluoridasi air?";
String s2 = "Uh? Ya, aku pernah mendengarnya, Jack, ya. Ya.";

String [] m1 = cocok (s1, "fluoridasi");
if (m1! = null) {// Jika bukan null, maka kecocokan ditemukan
  // Ini akan dicetak ke konsol, karena kecocokan ditemukan.
  println ("Menemukan kecocokan di '" + s1 + "'");
} lain {
  println ("Tidak ditemukan kecocokan di '" + s1 + "'");
}
String [] m2 = cocok (s2, "fluoridasi");
if (m2! = null) {
  println ("Menemukan kecocokan di '" + s2 + "'");
} lain {
  // Ini akan dicetak ke konsol, karena tidak ditemukan kecocokan.
```

```
println ("Tidak ditemukan kecocokan di '" + s2 + "'");
}
```

Diskripsi

Fungsi ini digunakan untuk menerapkan ekspresi reguler ke selembar teks, dan mengembalikan grup yang cocok (elemen yang ditemukan di dalam tanda kurung) sebagai array String. Jika tidak ada kecocokan, nilai nol akan dikembalikan. Jika tidak ada grup yang ditentukan dalam ekspresi reguler, tetapi urutannya cocok, array dengan panjang 1 (dengan teks yang cocok sebagai elemen pertama dari array) akan dikembalikan.

Untuk menggunakan fungsi ini, periksa terlebih dahulu untuk melihat apakah hasilnya nol. Jika hasilnya nol, maka urutannya tidak cocok sama sekali. Jika urutannya cocok, sebuah array dikembalikan.

Jika ada grup (ditentukan oleh set kurung) dalam ekspresi reguler, maka konten masing-masing akan dikembalikan dalam array. Elemen [0] dari kecocokan ekspresi reguler mengembalikan seluruh string yang cocok, dan grup yang cocok mulai pada elemen [1] (grup pertama adalah [1], yang kedua [2], dan seterusnya).

Sintak

match(str, regexp)

Parameter-parameter

Str String	:	String yang akan dicari
Regexp String	:	regexp yang akan digunakan untuk pencocokan

matchAll()

Examples

```
String s = "Di dalam tag, Anda akan menemukan <tag> beberapa </tag>";
    s + = "<tag> bagian </tag> dari <tag> konten </tag>.";
String [] [] m = matchAll (s, "<tag> (. *?) </tag>");
untuk (int i = 0; i <m.length; i ++) {
  println ("Ditemukan '" + m [i] [1] + "' di dalam tag.");
}
// Mencetak ke konsol:
// "Ditemukan 'banyak' di dalam sebuah tag."
// "Ditemukan 'potongan' di dalam tag."
// "Ditemukan 'konten' di dalam tag."
```

Diskripsi

Fungsi ini digunakan untuk menerapkan ekspresi reguler ke selembar teks, dan mengembalikan daftar grup yang cocok (elemen yang ditemukan di dalam tanda kurung) sebagai array String dua dimensi. Jika tidak ada kecocokan, nilai nol akan dikembalikan. Jika tidak ada grup yang ditentukan dalam ekspresi reguler, tetapi urutannya cocok, array dua dimensi masih dikembalikan, tetapi dimensi kedua hanya panjang satu.
Fungsi ini digunakan untuk menerapkan ekspresi reguler ke selembar teks, dan mengembalikan daftar grup yang cocok (elemen yang ditemukan di dalam tanda kurung) sebagai array String dua dimensi. Jika tidak ada kecocokan, nilai nol akan dikembalikan. Jika tidak ada grup yang ditentukan dalam ekspresi reguler, tetapi urutannya cocok, array dua dimensi masih dikembalikan, tetapi dimensi kedua hanya panjang satu.

Untuk menggunakan fungsi ini, periksa terlebih dahulu untuk

melihat apakah hasilnya nol. Jika hasilnya nol, maka urutannya tidak cocok sama sekali. Jika urutannya cocok, array 2D dikembalikan.
Jika ada grup (ditentukan oleh set kurung) dalam ekspresi reguler, maka konten masing-masing akan dikembalikan dalam array. Dengan asumsi loop dengan variabel counter i, elemen [i] [0] dari pencocokan ekspresi reguler mengembalikan seluruh string yang cocok, dan grup pencocokan mulai dari elemen [i] [1] (grup pertama adalah [i] [1] , yang kedua [i] [2], dan seterusnya).

Sintak
matchAll (str, regexp)

Parameter-parameter
Str String: String yang akan dicari
Regexp String: regexp yang akan digunakan untuk pencocokan

nf()

Examples

```
int a=200, b=40, c=90;
String sa = nf(a, 10);
println(sa);  // Prints "0000000200"
String sb = nf(b, 5);
println(sb);  // Prints "00040"
String sc = nf(c, 3);
println(sc);  // Prints "090"

float d = 200.94, e = 40.2, f = 9.012;
```


```
String sd = nf(d, 10, 4);
println(sd);  // Prints "0000000200.9400"
String se = nf(e, 5, 3);
println(se);  // Prints "00040.200"
String sf = nf(f, 3, 5);
println(sf);  // Prints "009.01200"
```

Diskripsi

Fungsi utilitas untuk memformat angka menjadi string. Ada dua versi: satu untuk format float, dan satu untuk format ints. Nilai untuk parameter digit , kiri , dan kanan harus selalu berupa bilangan bulat positif.

Seperti yang ditunjukkan pada contoh di atas, nf () digunakan untuk menambahkan nol ke kiri dan / atau kanan angka. Ini biasanya untuk menyelaraskan daftar angka. Untuk menghapus angka dari angka titik-mengambang, gunakan fungsi int () , ceil () , floor () , atau round () .

Sintak

nf (num)
nf (nums)
nf (nums, digits)
nf (num, digits)
nf (nums, left, right)
nf (num, left, right)

Parameter-parameter

Nums	float [], atau int []: angka untuk diformat
Digit	int: jumlah digit untuk diisi dengan nol
Num	float, atau int: angka untuk diformat
Left	int: jumlah digit di sebelah kiri titik desimal
Right	int: jumlah digit di sebelah kanan titik desimal

nfp()

Examples

```
int a=200, b=-40, c=90;
String sa = nfp(a, 10);
println(sa);  // Prints "+0000000200"
String sb = nfp(b, 5);
println(sb);  // Prints "-00040"
String sc = nfp(c, 3);
println(sc);  // Prints "+090"

float d = -200.94, e = 40.2, f = -9.012;
String sd = nfp(d, 10, 4);
println(sd);  // Prints "-0000000200.9400"
String se = nfp(e, 5, 3);
println(se);  // Prints "+00040.200"
String sf = nfp(f, 3, 5);
println(sf);  // Prints "-009.01200"
```

Diskripsi

Fungsi utilitas untuk memformat angka menjadi string. Mirip dengan nf () tetapi menempatkan "+" di depan angka positif dan "-" di depan angka negatif. Ada dua versi: satu untuk format float, dan satu untuk format ints. Nilai untuk parameter digit , kiri , dan kanan harus selalu berupa bilangan bulat positif.

Sintak

```
nfp ( num, digits)
nfp ( nums, digits)
NFP ( nums, left, right)
NFP ( num, left, right)
```

Parameter-parameter

num float, atau int: angka untuk diformat
digit int: jumlah digit untuk diisi dengan nol
nums float [], atau int []: angka untuk diformat
left int: jumlah digit di sebelah kiri titik decimal
Right int: jumlah digit di sebelah kanan titik decimal

nfs ()

Examples

```
int a = 200, b = -40, c = 90;
String sa = nfs (a, 10);
println (sa); // Mencetak "0000000200"
String sb = nfs (b, 5);
println (sb); // Mencetak "-00040"
String sc = nfs (c, 3);
println (sc); // Mencetak "090"

float d = -200.94, e = 40.2, f = -9.012;
String sd = nfs (d, 10, 4);
println (sd); // Mencetak "-0000000200.9400"
String se = nfs (e, 5, 3);
println (se); // Mencetak "00040.200"
String sf = nfs (f, 3, 5);
println (sf); // Mencetak "-009.01200"
```

Diskripsi

Fungsi utilitas untuk memformat angka menjadi string. Mirip dengan nf () , tetapi meninggalkan ruang kosong di depan angka positif sehingga mereka sejajar dengan angka negatif terlepas dari simbol minus. Ada dua versi: satu untuk format

float, dan satu untuk format ints. Nilai untuk parameter digit , kiri , dan kanan harus selalu berupa bilangan bulat positif.

Sintak

nfs (num, digits)
nfs (nums, digits)
nfs (nums, left, right)
nfs (num, left, right)

Parameter-parameter

num	float, atau int: angka untuk diformat
digit	int: jumlah digit untuk diisi dengan nol
nums	float [], atau int []: angka untuk diformat
left	int: jumlah digit di sebelah kiri titik decimal
Right	int: jumlah digit di sebelah kanan titik decimal

Split()

Examples

String pria = "Chernenko, Andropov, Brezhnev";
String [] list = split (men, ',');
// daftar [0] sekarang "Chernenko", daftar [1] adalah "Andropov" ...
Nomor string = "8 67 5 309";
int [] nums = int (split (angka, "));
// nums [0] sekarang 8, nums [1] sekarang 67 ...
String pria = "Chernenko] Andropov] Brezhnev";
String [] list = split (men, "]");
// daftar [0] sekarang "Chernenko", daftar [1] adalah "Andropov" ...

Diskripsi
Fungsi split () memecah String menjadi beberapa bagian menggunakan karakter atau string sebagai pembatas. The delim parameter menentukan karakter atau karakter yang menandai batas-batas antara masing-masing bagian. Array String [] dikembalikan yang berisi masing-masing bagian.

Jika hasilnya adalah serangkaian angka, Anda bisa mengonversi array String [] menjadi array float [] atau int [] menggunakan fungsi konversi datatype int () dan float () . (Lihat contoh kedua di atas.)
Fungsi splitTokens () bekerja dengan cara yang serupa, kecuali bahwa fungsi itu terpecah menggunakan rentang karakter alih-alih karakter atau urutan tertentu

Sintak.
Split (value, delim)

Parameter-parameter

Value String: String yang akan dibagi

Delim char: karakter atau String yang digunakan untuk memisahkan data

splitTokens ()

Examples

```
String t = "a b";
String [] q = splitTokens (t);
println (q [0]); // Mencetak "a"
println (q [1]); // Mencetak "b"
// Meskipun pemformatan buruk, data diurai dengan benar.
// Tanda "," sebagai pembatas berarti terputus setiap kali koma * atau *
// spasi ditemukan di String. Berbeda dengan fungsi split (),
// beberapa pembatas yang berdekatan diperlakukan sebagai satu break.
String s = "a, bc ,, d";
String [] q = splitTokens (s, ",");
println (q.length + "values found"); // Mencetak "4 nilai ditemukan"
println (q [0]); // Mencetak "a"
println (q [1]); // Mencetak "b"
println (q [2]); // Mencetak "c"
println (q [3]); // Mencetak "d"
```

Diskripsi

Fungsi splitTokens () membagi sebuah String pada satu atau banyak pembatas karakter atau "token." The delim parameter

menentukan karakter atau karakter untuk digunakan sebagai batas.
Jika tidak ada karakter delim yang ditentukan, karakter spasi apa pun digunakan untuk membelah. Karakter spasi putih meliputi tab (\ t), line feed (\ n), carriage return (\ r), form feed (\ f), dan spasi.
Setelah menggunakan fungsi ini untuk mem-parsing data yang masuk, biasanya mengkonversi data dari Strings ke integer atau float dengan menggunakan fungsi konversi datatype int () dan float () .

Sintak
splitTokens (value)
splitTokens (value, delim)

Parameter-parameter
Value String: String yang akan dibagi
Delim String: daftar karakter individu yang akan digunakan sebagai pemisah

trim()

Examples
```
String s1 = "Somerville MA";
println (s1); // Mencetak "Somerville MA"
String s2 = trim (s1);
println (s2); // Mencetak "Somerville MA"

String [] a1 = {"inconsistent", "spacing"}; // Catat spasi
String [] a2 = trim (a1);
printArray (a2);
// Mencetak isi array berikut ke konsol:
```

```
// [0] "tidak konsisten"
// [1] "spasi"
```

Diskripsi

Menghapus karakter spasi putih dari awal dan akhir sebuah String. Selain karakter spasi putih standar seperti spasi, carriage return, dan tab, fungsi ini juga menghilangkan karakter Unicode "nbsp".

Sintak

```
trim ( str)
trim ( array)
```

Parameter-parameter

Str	String: string apa saja
Array	String []: sebuah array String

BAB VIII
ARRAY FUNCTIONS

append()

```
String [] sa1 = {"OH", "NY", "CA"};
String [] sa2 = append (sa1, "MA");
println (sa2);
// Mencetak konten array yang diperbarui ke konsol:
// [0] "OH"
// [1] "NY"
// [2] "CA"
// [3] "MA"
```

Diskripsi

Memperluas array dengan satu elemen dan menambahkan data ke posisi baru. Tipe data dari parameter elemen harus sama dengan tipe data array.

Saat menggunakan array objek, data yang dikembalikan dari fungsi harus dilemparkan ke tipe data array objek. Sebagai contoh: SomeClass [] items = (SomeClass []) append (originalArray, elemen)

Sintak

append(array, value)

Parameter-parameter

Array Objek, String [], float [], int [], char [], atau byte []: array untuk ditambahkan

Value Objek, String, float, int, char, atau byte: data baru untuk array

arrayCopy()

Examples

```
String[] north = { "OH", "IN", "MI" };
String[] south = { "GA", "FL", "NC" };
arrayCopy(north, south);
println(south);
// Prints updated array contents to the console:
// [0] "OH"
// [1] "IN"
// [2] "MI"

String[] north = { "OH", "IN", "MI"};
String[] south = { "GA", "FL", "NC"};
arrayCopy(north, 1, south, 0, 2);
println(south);
// Prints updated array contents to the console:
// [0] "IN"
// [1] "MI"
// [2] "NC"
```

Diskripsi

Menyalin array (atau bagian dari array) ke array lain. The src array disalin ke dst array, dimulai pada posisi yang ditentukan oleh srcPosition dan ke posisi yang ditentukan oleh dstPosition . Jumlah elemen yang akan disalin ditentukan oleh panjangnya . Perhatikan bahwa nilai penyalinan menimpa nilai yang ada di larik tujuan. Untuk menambahkan nilai alih-alih menimpanya, gunakan concat () .

Versi yang disederhanakan dengan hanya dua argumen - arrayCopy (src, dst) - menyalin seluruh array ke yang lain dengan ukuran yang sama. Ini sama dengan arrayCopy (src, 0, dst, 0, src.length) .

Menggunakan fungsi ini jauh lebih efisien untuk menyalin data array daripada mengulang melalui for () loop dan menyalin masing-masing elemen secara individual. Fungsi ini hanya menyalin referensi, yang berarti bahwa untuk sebagian besar tujuan hanya menyalin array satu dimensi (satu set kurung). Jika digunakan dengan array dua (atau tiga atau lebih), itu hanya akan menyalin referensi di tingkat pertama, karena array dua dimensi hanyalah sebuah "array array". Namun, ini tidak menghasilkan kesalahan, karena ini sering merupakan perilaku yang diinginkan. Secara internal, fungsi ini memanggil metode Java System.arraycopy () , sehingga sebagian besar hal yang berlaku di sana diwariskan.

Sintak

arrayCopy(src, srcPosition, dst, dstPosition, length)
arrayCopy(src, dst, length)
arrayCopy(src, dst)

Parameter-parameter

src : Obyek array sumber
srcPositionint: int posisi awal di larik sumber
dstObject: Objek array tujuan dengan tipe data yang sama dengan array sumber
dstPositionint: int posisi awal di array tujuan
lengthint: int jumlah elemen array yang akan disalin

concat()

Examples

```
String[] sa1 = { "OH", "NY", "CA"};
String[] sa2 = { "KY", "IN", "MA"};
String[] sa3 = concat(sa1, sa2);
```

```
println(sa3);
// Prints updated array contents to the console:
// [0] "OH"
// [1] "NY"
// [2] "CA"
// [3] "KY"
// [4] "IN"
// [5] "MA"
```

Diskripsi

Menggabungkan dua array. Misalnya, menggabungkan array {1, 2, 3} dan array {4, 5, 6} menghasilkan {1, 2, 3, 4, 5, 6}. Kedua parameter harus berupa array dengan tipe data yang sama.

Saat menggunakan array objek, data yang dikembalikan dari fungsi harus dilemparkan ke tipe data array objek. Sebagai contoh: SomeClass [] items = (SomeClass []) concat (array1, array2) .

Sintak

concat (a, b)

Parameter-parameter

a Objek, String [], float [], int [], char [], byte [], atau boolean []: array pertama yang akan digabungkan

b Objek, String [], float [], int [], char [], byte [], atau boolean []: larik kedua untuk menggabungkan

expand()

Examples

```
int[] data = {0, 1, 3, 4};
println(data.length);  // Prints "4"
data = expand(data);
println(data.length);  // Prints "8"
data = expand(data, 512);
println(data.length);  // Prints "512"
PImage[] imgs = new PImage[32];
println(imgs.length);  // Prints "32"
imgs = (PImage[]) expand(imgs);
println(imgs.length);  // Prints "64"
```

Diskripsi

Meningkatkan ukuran array. Secara default, fungsi ini menggandakan ukuran array, tetapi parameter newSize opsional memberikan kontrol yang tepat atas peningkatan ukuran.

Saat menggunakan array objek, data yang dikembalikan dari fungsi harus dilemparkan ke tipe data array objek. Sebagai contoh: SomeClass [] items = (SomeClass []) perluas (originalArray)

Sintak

expand(list)
expand(list, newSize)

Parameter-parameter

List Object, String [], double [], float [], long [], int [], char [], byte [], atau boolean []: array untuk memperluas
newSize int: ukuran baru untuk array

Reverse()

Examples

```
String sa [] = {"OH", "NY", "MA", "CA"};
sa = mundur (sa);
println (sa);
// Mencetak isi array yang diperbarui ke konsol:
// [0] "CA"
// [1] "MA"
// [2] "NY"
// [3] "OH"
```

Diskripsi

Membalik urutan array.

Sintak

reverse(list)

Parameter-parameter

List Object, String[], float[], int[], char[], byte[], or boolean[]: booleans[], bytes[], chars[], ints[], floats[], or Strings[]

shorten()

Examples

```
String[] sa1 = { "OH ", "NY ", "CA "};
String[] sa2 = shorten(sa1);
println(sa1);  // 'sa1' still contains OH, NY, CA
println(sa2);  // 'sa2' now contains OH, NY
```

Diskripsi

Mengurangi array dengan satu elemen dan mengembalikan array yang dipersingkat.
Saat menggunakan array objek, data yang dikembalikan dari fungsi harus dilemparkan ke tipe data array objek. Sebagai contoh: SomeClass [] items = (SomeClass []) mempersingkat (originalArray)

Sintak
shorten(list)

Parameter-parameter
List Object, String[], float[], int[], char[], byte[], or boolean[]: array untuk mempersingkat.

sort()

Example

```
float[] a = { 3.4, 3.6, 2, 0, 7.1 };
a = sort(a);
println(a);
// Prints the contents of the sorted array:
// [0] 0.0
// [1] 2.0
// [2] 3.4
// [3] 3.6
// [4] 7.1
String[] s = { "deer", "elephant", "bear", "aardvark", "cat" };
s = sort(s);
println(s);
// Prints the contents of the sorted array:
// [0] "aardvark"
```


```
// [1] "bear"
// [2] "cat"
// [3] "deer"
// [4] "elephant"
String[] s = { "deer", "elephant", "bear", "aardvark", "cat" };
s = sort(s, 3);
println(s);
// Prints the contents of the array, with the first 3 elements sorted:
// [0] "bear"
// [1] "deer"
// [2] "elephant"
// [3] "aardvark"
// [4] "cat"
```

Diskripsi

Mengurutkan susunan angka dari yang terkecil hingga yang terbesar, atau menempatkan susunan kata dalam urutan abjad. Array asli tidak dimodifikasi; array yang dipesan ulang dikembalikan. The count parameter menyatakan jumlah elemen untuk menyortir. Misalnya, jika ada 12 elemen dalam array dan hitungan diatur ke 5, hanya 5 elemen pertama dalam array yang akan diurutkan.

Sintak

Sort(list)
sort(list, count)

Parameter-parameter

List String [], float [], int [], char [], atau byte []: array untuk disortir
Count int: jumlah elemen untuk disortir, mulai dari 0

splice()

Examples

```
String[] a = { "OH", "NY", "CA" };
a = splice(a, "KY", 1);  // Splice one value into an array
println(a);
// Prints the following array contents to the console:
// [0] "OH"
// [1] "KY"
// [2] "NY"
// [3] "CA"

println();  // Prints a blank line

String[] b = { "VA", "CO", "IL" };
a = splice(a, b, 2);  // Splice one array of values into another
println(a);
// Prints the following array contents to the console:
// [0] "OH"
// [1] "KY"
// [2] "VA"
// [3] "CO"
// [4] "IL"
// [5] "NY"
// [6] "CA"
```

Diskripsi

Menyisipkan nilai atau array nilai ke dalam array yang ada. Dua parameter pertama harus berupa array dengan tipe data yang sama. Parameter pertama menentukan larik awal yang akan dimodifikasi, dan parameter kedua menentukan data yang akan dimasukkan.

Parameter ketiga adalah nilai indeks yang menentukan posisi array dari mana untuk memasukkan data. (Ingat bahwa penomoran indeks array dimulai dari nol, sehingga posisi pertama adalah 0, posisi kedua adalah 1, dan seterusnya.) Saat menyambungkan array objek, data yang dikembalikan dari fungsi harus dilemparkan ke tipe data array objek. . Sebagai contoh: SomeClass [] items = (SomeClass []) sambatan (array1, array2, indeks).

Sintak
splice(list, value, index)

Parameter-parameter
List Objek, String [], float [], int [], char [], byte [], atau boolean []: array untuk dipecah menjadi
Value Objek, String [], String, float [], float, int [], int, char [], char, byte [], byte, boolean [], atau boolean: nilai yang akan disambungkan dalam
Indeks int: posisi dalam array yang digunakan untuk memasukkan data

subset()

Examples

```
tring[] sa1 = { "OH", "NY", "CA", "VA", "CO", "IL" };
String[] sa2 = subset(sa1, 1);
println(sa2);
// Prints the following array contents to the console:
// [0] "NY"
// [1] "CA"
// [2] "VA"
// [3] "CO"
```

```
// [4] "IL"
println();
String[] sa3 = subset(sa1, 2, 3);
println(sa3);
// Prints the following array contents to the console:
// [0] "CA"
// [1] "VA"
// [2] "CO"
```

Diskripsi

Ekstrak array elemen dari array yang ada. The daftar parameter mendefinisikan array dari mana unsur-unsur akan disalin, dan mulai dan jumlah parameter menentukan unsur-unsur untuk mengekstrak. Jika tidak ada hitungan yang diberikan, elemen akan diekstraksi dari awal hingga akhir array. Saat menentukan awal , ingat bahwa elemen array pertama adalah 0. Fungsi ini tidak mengubah array sumber.

Saat menggunakan array objek, data yang dikembalikan dari fungsi harus dilemparkan ke tipe data array objek. Sebagai contoh: SomeClass [] items = (SomeClass []) subset (originalArray, 0, 4)

Sintak

subset(list, start)
subset(list, start, count)

Parameter-parameter

List Object, String [], double [], float [], long [], int [], char [], byte [], atau boolean []: array untuk mengekstrak dari
Start int: posisi untuk memulai
Count int: jumlah nilai untuk diekstraksi

BAB IX
CONTROL

Relational Operators

!= (inequality)
Examples

```
int a = 22;
int b = 23;
if (a != b) {
  println("variable a is not equal to variable b");
}
```

Diskripsi
Menentukan apakah satu ekspresi tidak setara dengan yang lain

Sintak
value1! = value2

Parameter-parameter
Value1int, float, char, byte, boolean
Value2int, float, char, byte, boolean

< (less than)

Examples

```
int a = 22;
int b = 23;
if (a < b) {
  println("variable a is less then variable b ");
}
```


Diskripsi
Menguji apakah nilai di sebelah kiri lebih kecil dari nilai di sebelah kanan.
Sintak
value1 <value2

Parameter-parameter
Value1int, float, char, atau byte
Value2int, float, char, atau byte

<= (less than atau equal to)

Examples
```
int a = 22;
int b = 23;
jika (a <= b) {
  println ("variabel a kurang atau sama dengan variabel b");
}
```

Diskripsi
Menguji apakah nilai di sebelah kiri kurang dari nilai di sebelah kanan atau jika nilainya setara.

Sintak
value1 <= value2

Parameter-parameter
Value1 int, float, char, atau byte
Value2 int, float, char, atau byte

== (equality)

Examples

```
int a = 23;
int b = 23;
if (a == b) {
  println ("variabel a dan b sama");
}
```

Diskripsi

Menentukan apakah dua nilai setara. Operator kesetaraan berbeda dari operator penugasan.
Perhatikan bahwa ketika membandingkan objek String, Anda harus menggunakan metode equals () alih-alih == untuk membandingkan kontennya

Sintak

value1 == value2

Parameter-parameter

Value1	int, float, char, byte,boolean
Value2	int, float, char, byte,boolean

> (greater than)

Examples

```
int a = 5;
int b = 13;
if (b> a) {
  println ("variabel b lebih besar variabel a");
```


}

Diskripsi
Menguji apakah nilai di sebelah kiri lebih besar dari nilai di sebelah kanan.

Sintak
value1> value2

Parameter-parameter

Value1	int, float, char, atau byte
Value2	int, float, char, atau byte

>= (greater than atau equal to)

Examples

```
int a = 23;
int b = 23;
if (a >= b) {
  println("variable a is greater or equal to variable b ")
}
```

Diskripsi
Menguji apakah nilai di sebelah kiri lebih besar dari nilai di sebelah kanan atau jika nilainya setara.

Sintak
value1> = value2

Parameter-parameter

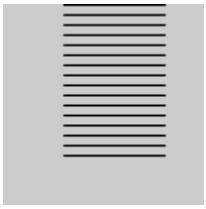
Value1	int, float, char, atau byte
Value2	int, float, char, atau byte

BAB X
ITERATION

for

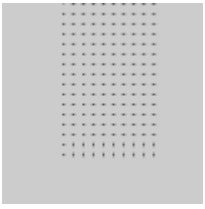
Examples

```
for (int i = 0; i < 40; i = i+1) {
  line(30, i, 80, i);
}
```



```
for (int i = 0; i < 80; i = i+5) {
  line(30, i, 80, i);
}
```

```
for (int i = 40; i < 80; i = i+5) {
  line(30, i, 80, i);
}
```



```
// Nested for () loop dapat digunakan untuk
// hasilkan pola dua dimensi

for (int i = 30; i < 80; i = i+5) {
  for (int j = 0; j < 80; j = j+5) {
    point(i, j);
  }
}
// Contoh ini tidak memiliki output visual,
// tetapi mencetak nilai ke konsol.
```

```
int[] nums = { 5, 4, 3, 2, 1 };
for (int i : nums) {
  println(i);
}
```

Diskripsi

Mengontrol urutan pengulangan. Dasar untuk struktur memiliki tiga bagian: init , test , dan update . Setiap bagian harus dipisahkan dengan tanda titik koma (;). Pengulangan berlanjut sampai pengujian bernilai false . Ketika struktur for dieksekusi, urutan peristiwa berikut terjadi:

Pernyataan init dijalankan.

Tes dievaluasi benar atau salah.

Jika tes ini benar , lompat ke langkah 4.Jika tes ini salah , lompat ke langkah 6.

Jalankan pernyataan di dalam blok.

Jalankan pernyataan pembaruan dan lompat ke langkah 2.

Keluar dari loop.

Pada contoh pertama di atas, struktur for dieksekusi 40 kali. Dalam pernyataan init, nilai i dibuat dan set ke nol. i kurang dari 40, sehingga tes mengevaluasi sebagai benar . Pada akhir setiap loop, saya bertambah satu. Pada eksekusi ke-41, tes dievaluasi sebagai false , karena i kemudian sama dengan 40, jadi i <40 tidak lagi benar. Dengan demikian, loop keluar.

Tipe kedua untukstruktur membuatnya lebih mudah untuk beralih ke setiap elemen dari sebuah array. Contoh terakhir di atas menunjukkan cara kerjanya. Di dalam tanda kurung, pertama-tama tentukan tipe data array, kemudian tentukan nama variabel. Nama variabel ini akan ditugaskan untuk setiap elemen array pada gilirannya sebagai untuk bergerak melalui seluruh array. Akhirnya, setelah titik dua, tentukan nama array yang akan digunakan.

Sintak

```
for (init; test; update) {
  statements
}
for (datatype element : array) {
  statements
```

```
}
```

Parameter-parameter

Init	pernyataan dieksekusi satu kali saat memulai loop
Test	jika tes bernilai true , pernyataan dieksekusi
Update	Dieksekusi pada akhir setiap iterasi
Statements	kumpulan pernyataan yang dieksekusi setiap kali melalui loop
Datatype	Tipe data elemen dalam array
Element	nama sementara untuk digunakan untuk setiap elemen array
Array	nama array yang akan diulangi

While

Examples

```
int i = 0;
while (i < 80) {
  line(30, i, 80, i);
  i = i + 5;
}
```

Diskripsi

Mengontrol urutan pengulangan. The sementara struktur mengeksekusi serangkaian pernyataan terus menerus sementara ekspresi adalah benar . Ekspresi harus diperbarui selama pengulangan atau program tidak akan pernah "keluar" dari sementara .

Fungsi ini bisa berbahaya karena kode di dalam sementara lingkaran tidak akan selesai sampai ekspresi dalam saat menjadi palsu. Ini akan mengunci semua kode lain dari menjalankan (mis., Acara mouse dan keyboard tidak akan diperbarui). Hati-hati - jika digunakan secara tidak benar, ini dapat mengunci kode Anda (dan kadang-kadang bahkan lingkungan Pemrosesan itu sendiri).

Synntax

```
while (expression) {
  statements
}
```

Parameter-parameter

Expression	ekspresi yang valid
Statements	satu atau lebih pernyataan

BAB XI
CONDITIONALS

?: (conditional)

Examples

```
int s = 0;
for (int i = 5; i < 100; i += 5) {
  s = (i < 50) ? 0 : 255;
  stroke(s);
  line(30, i, 80, i);
}
```

Diskripsi

Pintasan untuk menulis struktur if and else . Operator bersyarat, ?: Kadang-kadang disebut operator ternary, operator yang mengambil tiga argumen. Jika tes bernilai true , ekspresi1 dievaluasi dan dikembalikan. Jika kondisi dievaluasi salah , ekspresi2 dievaluasi dan dikembalikan.

Ekspresi bersyarat berikut:

```
hasil = tes? Ekspresi1: Ekspresi2
```

setara dengan struktur ini:

```
if (test) {
  result = expression1
} else {
  result = expression2
}
```

Sintak

```
test ? expression1 : expression2
```

Parameter-parameter

test	Setiap ekspresi valid yang mengevaluasi benar atau salah

expression1	Setiap ekspresi yang valid
expression2	Setiap ekspresi yang valid

break

Examples

```
char letter = 'B';

switch(letter) {
  case 'A':
    println("Alpha");  // tidak mengeksekusi
    break;
  case 'B':
    println("Bravo");  // mencetak "Bravo"
    break;
  default:
    println("Zulu");   // tidak mengeksekusi
    break;
}
```

Diskripsi

Mengakhiri pelaksanaan struktur seperti beralih , untuk , atau sementara dan melompat ke pernyataan berikutnya setelah.

case

Examples

```
char letter = 'B';

switch(letter) {
  case 'A':
    println("Alpha");  // tidak mengeksekusi
    break;
  case 'B':
    println("Bravo");  // mencetak "Bravo"
    break;
  default:
    println("Zulu");   // tidak menegeksekusi
    break;
}
```

Diskripsi

Menunjukkan label yang berbeda untuk dievaluasi dengan parameter dalam struktur sakelar .

Sintak

case label: statements

Parameter-parameter

label byte, char, or int

Statement satu atau lebih pernyataan yang valid

continue

Examples

```
for (int i = 0; i < 100; i += 10) {
  if (i == 70) {  // jika “i” adalah 70,
    continue;     // lompat ke iterasi berkutnya,
  }               // karena itu jangan menggambar garis.
  line(i, 0, i, height);
}
```

Diskripsi
Saat dijalankan di dalam untuk atau sementara , itu akan melompati sisa blok dan memulai iterasi berikutnya.

Sintak
Continue

Default

Examples

```
char letter = 'F';
switch(letter) {
  case 'A':
    println("Alpha");  // tidak mengeksekusi
    break;
  case 'B':
    println("Bravo");  // tidak mengeksekusi
    break;
  default:
    println("Zulu");   // mencetak "Zulu"
```

```
  break;
}
```

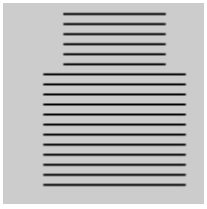
Diskripsi
Kata kunci untuk menentukan kondisi default sakelar . Jika tidak ada label case yang cocok dengan parameter switch , pernyataan setelah sintaks default dieksekusi. Beralih struktur tidak memerlukan default .

Sintak
default: statements

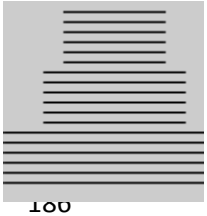
Parameter-parameter
Statement satu atau lebih pernyataan yang valid untuk dieksekusi

else

Examples



```
for (int i = 5; i < 95; i += 5) {
  if (i < 35) {
    line(30, i, 80, i);
  } else {
    line(20, i, 90, i);
  }
 }
```



```
for (int i = 5; i < 95; i += 5) {
 if (i < 35) {
   line(30, i, 80, i);
 } else if (i < 65) {
```

```
  line(20, i, 90, i);
 } else {
                    line(0, i, 100, i);
                  }
                }
```

Diskripsi

Memperpanjang struktur if yang memungkinkan program memilih antara dua atau lebih blok kode. Ini menentukan blok kode untuk mengeksekusi ketika ekspresi dalam jika adalah palsu.

Sintak

```
if (expression) {
  statements
} else {
  statements
}

if (expression) {
  statements
} else if (expression) {
  statements
} else {
  statements
}
```

Parameter-parameter

Expresion	setiap ekspresi valid yang mengevaluasi benar atau salah
Statement	satu atau lebih pernyataan yang akan dieksekusi

if

Examples

```
for (int i = 5; i <height; i + = 5) {
  stroke (255); // Atur warnanya menjadi putih
  if (i <35) {// Ketika 'i' kurang dari 35 ...
    stroke (0); //...set warnanya menjadi hitam
                        }
                        line (30, i, 80, i);}
```

Diskripsi

Mengizinkan program membuat keputusan tentang kode mana yang akan dieksekusi. Jika tes bernilai true , pernyataan yang disertakan dalam blok dieksekusi dan jika tes bernilai false , pernyataan tidak dieksekus.

Sintak

```
if (test) {
  statement
}
```

Parmeters

Test setiap ekspresi valid yang mengevaluasi benar atau salah
Satement satu atau lebih pernyataan yang akan dieksekusi

switch

Examples

```
int num = 1;

switch(num) {
```

```
  case 0:
    println("Zero");  // Does not execute
    break;
  case 1:
    println("One");  // Prints "One"
    break;
}
char letter = 'N';

switch(letter) {
  case 'A':
    println("Alpha");  // tidak mengeksekusi
    break;
  case 'B':
    println("Bravo");  // tidak mengeksekusi
    break;
  default:        // Default dijalankan jika label case
 println("None");   // tidak cocok dengan parameter sakelar
    istirahat;
    break;
}
// Menghapus "istirahat" memungkinkan pengujian
// untuk lebih dari satu nilai sekaligus
char letter = 'b';

switch(letter) {
  case 'a':
  case 'A':
    println("Alpha");  // Does not execute
    break;
  case 'b':
  case 'B':
    println("Bravo");  // Prints "Bravo"
    break;
```

```
}
```

Diskripsi

Bekerja seperti struktur if else , tetapi switch lebih nyaman ketika Anda harus memilih di antara tiga alternatif atau lebih. Kontrol program melompat ke case dengan nilai yang sama dengan ekspresi. Semua pernyataan yang tersisa di switch dieksekusi kecuali diarahkan oleh istirahat . Hanya tipe data primitif yang dapat dikonversi ke integer (byte, char, dan int) yang dapat digunakan sebagai parameter ekspresi . Standarnya adalah opsional.

Sintak

```
switch (ekspresi)
{
  label kasus:
     pernyataan
  label kasus: //
     Pernyataan opsional // "
  default: //"
     pernyataan // "
}
```

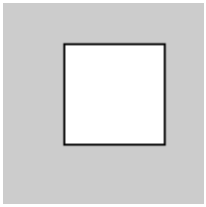
Parameter-parameter

Expression	byte, char, atau int
Label	byte, char, atau int
Statements	Satu atau lebih pernyataan yang di eksekusi

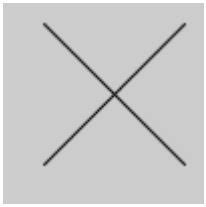
BAB XII
LOGICAL OPERATORS

! (logical NOT)

Examples



```
boolean a = false;
if (!a) {
  rect(30, 20, 50, 50);
}
if (a) {
  line(20, 10, 90, 80);
  line(20, 80, 90, 10);
}
```



```
boolean a = true;
if (!a) {
  rect(30, 20, 50, 50);
}
if (a) {
  line(20, 10, 90, 80);
  line(20, 80, 90, 10);
}
```

Diskripsi

Menolak nilai Boolean dari sebuah ekspresi. Mengembalikan nilai true jika ekspresi itu salah dan mengembalikan salah jika ekspresi itu benar . Jika ekspresi (a> b) bernilai true, maka ! (A> b) bernilai false.

Sintak

!expression

Parameter-parameter

Expressi	setiap ekspresi yang valid

&& (logical AND)

Examples

```
for (int i = 5; i <= 95; i += 5) {
  if ((i > 35) && (i < 60)) {
    stroke(0);  // Set color to black
  } else {
    stroke(255);  // Set color to white
  }
  line(30, i, 80, i);
}
```

Diskripsi

Membandingkan dua ekspresi dan mengembalikan true hanya jika keduanya bernilai true . Mengembalikan nilai false jika salah satu atau keduanya bernilai false . Daftar berikut menunjukkan semua kemungkinan kombinasi:

true && false // Mengevaluasi salah karena yang kedua adalah false
false && true // Mengevaluasi salah karena yang pertama adalah false
benar && benar // Mengevaluasi benar karena keduanya benar
salah && salah // Mengevaluasi salah karena keduanya salah

Sintak

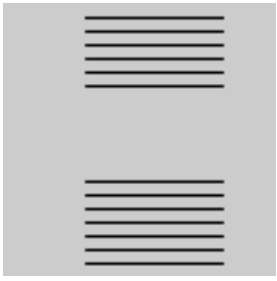
ekspresi1 && ekspresi2

Parameter-parameter

ekspresi1	setiap ekspresi yang valid
ekspresi2	setiap ekspresi yang valid

|| (logical OR)

Examples



```
for (int i = 5 ; i <= 95; i += 5) {
  if ((i < 35) || (i > 60)) {
    line(30, i, 80, i);
  }
}
```

Diskripsi

Membandingkan dua ekspresi dan mengembalikan true jika satu atau keduanya bernilai true . Mengembalikan nilai false hanya jika kedua ekspresi salah . Daftar berikut menunjukkan semua kemungkinan kombinasi:

true || false // Mengevaluasi benar karena yang pertama adalah false salah || true // Mengevaluasi benar karena yang kedua adalah benar
benar || true // Mengevaluasi benar karena keduanya benar salah || false // Mengevaluasi salah karena keduanya salah.

Sintak

expression1 || expression2

Parameter-parameter

ekspresi1	setiap ekspresi yang valid
ekspresi2	setiap ekspresi yang valid

IMPORT

Examples

```
import processing.pdf. *;

pengaturan batal () {
  ukuran (1024, 768, PDF);
}

batal draw () {
  garis (0, 0, lebar, tinggi);
  garis (0, tinggi, lebar, 0);
}
```

Diskripsi

Impor kata kunci digunakan untuk memuat pustaka ke dalam sketsa Pemrosesan. Perpustakaan adalah satu atau beberapa kelas yang dikelompokkan bersama untuk memperluas kemampuan Pemrosesan. The * karakter sering digunakan pada akhir baris impor (lihat contoh kode di atas) untuk memuat semua kelas terkait sekaligus, tanpa harus referensi mereka secara individu.

Pernyataan impor akan dibuat untuk Anda dengan memilih perpustakaan dari item "Impor Perpustakaan ..." di menu Sketsa.

Sintak

import libraryName

Parameter-parameter

libraryName nama perpustakaan untuk memuat (mis. "processing.pdf. *")

BAB XIII
SHAPE

createShape()

Examples

```
PShape square;  // The PShape object

void setup() {
  size(100, 100);
  // Creating the PShape as a square. The
  // numeric arguments are similar to rect().
  square = createShape(RECT, 0, 0, 50, 50);
  square.setFill(color(0, 0, 255));
  square.setStroke(false);
}

void draw() {
  shape(square, 25, 25);
}
```

```
PShape s;  // The PShape object

void setup() {
  size(100, 100);
  // Creating a custom PShape as a square, by
  // specifying a series of vertices.
  s = createShape();
  s.beginShape();
  s.fill(0, 0, 255);
  s.noStroke();
  s.vertex(0, 0);
  s.vertex(0, 50);
  s.vertex(50, 50);
  s.vertex(50, 0);
  s.endShape(CLOSE);
}
```


```
void draw() {
  shape(s, 25, 25);
}

PShape s;

void setup() {
  size(100, 100, P2D);
  s = createShape();
  s.beginShape(TRIANGLE_STRIP);
  s.vertex(30, 75);
  s.vertex(40, 20);
  s.vertex(50, 75);
  s.vertex(60, 20);
  s.vertex(70, 75);
  s.vertex(80, 20);
  s.vertex(90, 75);
  s.endShape();
}

void draw() {
  shape(s, 0, 0);
}
PShape alien, head, body;

void setup() {
  size(100, 100);

  // Create the shape group
  alien = createShape(GROUP);

  // Make two shapes
  ellipseMode(CORNER);
```

```
  head = createShape(ELLIPSE, -25, 0, 50, 50);
  head.setFill(color(255));
  body = createShape(RECT, -25, 45, 50, 40);
  body.setFill(color(0));

  // Add the two "child" shapes to the parent group
  alien.addChild(body);
  alien.addChild(head);
}

void draw() {
  background(204);
  translate(50, 15);
  shape(alien); // Draw the group
}
```

Diskripsi

Fungsi createShape () digunakan untuk mendefinisikan bentuk baru. Setelah dibuat, bentuk ini dapat digambar dengan fungsi Shape () . Cara dasar untuk menggunakan fungsi ini mendefinisikan bentuk primitif baru. Salah satu dari parameter berikut digunakan sebagai parameter pertama : ELLIPSE, RECT, ARC, TRIANGLE, SPHERE, BOX, QUAD, atau LINE. Parameter untuk masing-masing bentuk berbeda ini sama dengan fungsi yang sesuai : ellipse (), rect (), arc (), triangle (), sphere (), box (), quad (), dan line (). Contoh pertama di atas menjelaskan cara kerjanya.

Bentuk kustom dan unik dapat dibuat dengan menggunakan createShape () tanpa parameter. Setelah bentuk dimulai, atribut gambar dan geometri dapat diatur langsung ke bentuk dalam metode beginShape () dan endShape () . Lihat contoh kedua di atas untuk spesifik, dan referensi untuk beginShape () untuk semua opsi.

Fungsi createShape () juga dapat digunakan untuk membuat bentuk kompleks yang terbuat dari bentuk lain. Ini disebut "grup" dan dibuat dengan menggunakan parameter GROUP sebagai parameter pertama. Lihat contoh keempat diatas untuk melihat cara kerjanya.
Setelah menggunakan createShape () , stroke dan fill color dapat diatur dengan memanggil metode seperti setFill () dan setStroke () , seperti yang terlihat pada contoh di atas. Daftar lengkap metode dan bidang untuk kelas PShape ada di Memproses Javadoc .

Sintak
createShape()
createShape(type)
createShape(kind, p)

Parameter-parameter
Jenis int: POINT, LINE, SEGITIGA, QUAD, RECT, ELLIPSE, ARC, BOX, SPHERE
Hal float []: parameter yang cocok dengan jenis bentuk

loadShape()

Examples

```
PShape s;

void setup() {
  size(100, 100);
  // The file "bot.svg" must be in the data folder
```

```
  // of the current sketch to load successfully
  s = loadShape("bot.svg");
}

void draw() {
  shape(s, 10, 10, 80, 80);
}
PShape s;

void setup() {
  size(100, 100, P3D);
  // The file "bot.obj" must be in the data folder
  // of the current sketch to load successfully
  s = loadShape("bot.obj");
}

void draw() {
  background(204);
  translate(width/2, height/2);
  shape(s, 0, 0);
}
```

Diskripsi

Memuat geometri ke dalam variabel tipe PShape . File SVG dan OBJ dapat dimuat. Untuk memuat dengan benar, file harus berada di direktori data sketsa saat ini. Dalam kebanyakan kasus, loadShape () harus digunakan di dalam setup () karena memuat bentuk di dalam draw () akan mengurangi kecepatan sketsa.

Atau, file mungkin dimuat dari mana saja di komputer lokal menggunakan jalur absolut (sesuatu yang dimulai dengan / pada Unix dan Linux, atau huruf drive pada Windows), atau parameter nama file dapat menjadi URL untuk file yang ditemukan pada file jaringan.

Jika file tidak tersedia atau kesalahan terjadi, nullakan dikembalikan dan pesan kesalahan akan dicetak ke konsol. Pesan kesalahan tidak menghentikan program, namun nilai nol dapat menyebabkan NullPointerException jika kode Anda tidak memeriksa apakah nilai yang dikembalikan adalah nol.

Sintak
loadShape (filename)

Parameter-parameter
nama file String: nama file yang akan dimuat, bisa .svg atau .obj

PShape

Examples

```
PShape s;

void setup() {
  size(100, 100);
  // The file "bot.svg" must be in the data folder
  // of the current sketch to load successfully
  s = loadShape("bot.svg");
}

void draw() {
  shape(s, 10, 10, 80, 80);
}
PShape square;  // The PShape object

void setup() {
  size(100, 100);
```

```
  // Creating the PShape as a square. The corner
  // is 0,0 so that the center is at 40,40
  square = createShape(RECT, 0, 0, 80, 80);
}

void draw() {
  shape(square, 10, 10);
}
```

Diskripsi

Datatype untuk menyimpan bentuk. Sebelum bentuk digunakan, harus dimuat dengan loadShape () atau dibuat dengan createShape () . Fungsi shape () digunakan untuk menggambar shape ke jendela tampilan. Pemrosesan saat ini dapat memuat dan menampilkan bentuk SVG (Scalable Vector Graphics) dan OBJ. File OBJ hanya dapat dibuka menggunakan renderer P3D . Fungsi loadShape () mendukung file SVG yang dibuat dengan Inkscape dan Adobe Illustrator. Ini bukan implementasi SVG lengkap, tetapi menawarkan beberapa dukungan langsung untuk penanganan data vektor.

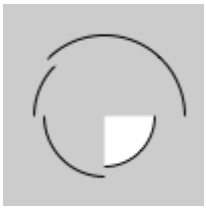
The PShapeobjek berisi sekelompok metode yang dapat beroperasi pada bentuk data. Beberapa metode tercantum di bawah ini, tetapi daftar lengkap yang digunakan untuk membuat dan memodifikasi bentuk tersedia di sini di Memproses Javadoc .

Untuk membuat bentuk baru, gunakan fungsi createShape () . Jangan gunakan sintaks PShape baru () .

BAB XIV
2D Primitives

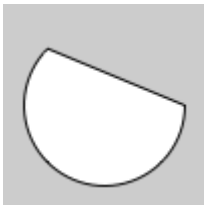
arc()

Examples

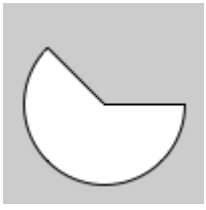


```
arc(50, 55, 50, 50, 0, HALF_PI);
noFill();
arc(50, 55, 60, 60, HALF_PI, PI);
arc(50, 55, 70, 70, PI, PI+QUARTER_PI);
arc(50, 55, 80, 80, PI+QUARTER_PI, TWO_PI);
```

```
arc(50, 50, 80, 80, 0, PI+QUARTER_PI, OPEN);
```



```
arc(50, 50, 80, 80, 0, PI+QUARTER_PI, CHORD);
```



```
arc(50, 50, 80, 80, 0, PI+QUARTER_PI, PIE);
```

Diskripsi

Menarik busur ke layar. Arcs digambar di sepanjang tepi luar elips yang didefinisikan oleh parameter a , b , c , dan d. Asal

Asal usul ellipse arc dapat diubah dengan fungsi ellipseMode (). Gunakan parameter start stop untuk menentukan sudut (dalam radian) untuk menggambar busur.
Ada tiga cara menggambar busur; teknik rendering yang digunakan ditentukan oleh parameter ketujuh opsional. Tiga opsi, yang digambarkan dalam contoh di atas, adalah PIE, OPEN, dan CHORD. Mode standar adalah stroke OPEN dengan isi PIE.
Dalam beberapa kasus, arc () fungsi tidak cukup akurat untuk menggambar halus. Sebagai contoh, bentuknya bisa naik-turun di layar saat berputar perlahan. Jika Anda memiliki masalah dengan bagaimana arc diberikan, Anda harus menggambar sendiri arc dengan beginShape () / endShape () atau PShape .

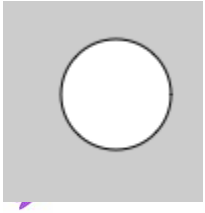
Sintak
arc(a, b, c, d, start, stop)
arc(a, b, c, d, start, stop, mode)

Parameter-parameter

A	float: x-coordinate of the arc's ellipse
B	float: y-coordinate of the arc's ellipse
C	float: width of the arc's ellipse by default
D	float: height of the arc's ellipse by default
start	float: angle to start the arc, specified in radians
stop	float: angle to stop the arc, specified in radians

circle()

Examples



circle(56, 46, 55);

Diskripsi

Menarik lingkaran ke layar. Secara default, dua parameter pertama mengatur lokasi pusat, dan yang ketiga menetapkan lebar dan tinggi bentuk. Asal dapat diubah dengan fungsi ellipseMode () .

Sintak

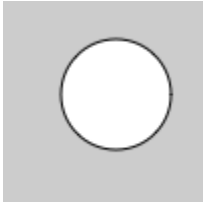
circle (x, y, extent)

Parameter-parameter

x float: x-coordinate of the ellipse
y float: y-coordinate of the ellipse
extent float: width and height of the ellipse by default

ellipse()

Examples



ellipse(56, 46, 55, 55);

Diskripsi

Menarik elips (oval) ke layar. Elips dengan lebar dan tinggi yang sama adalah sebuah lingkaran. Secara default, dua parameter pertama mengatur lokasi, dan parameter ketiga dan keempat mengatur lebar dan tinggi bentuk. Asal dapat diubah dengan fungsi ellipseMode ().

Sintak

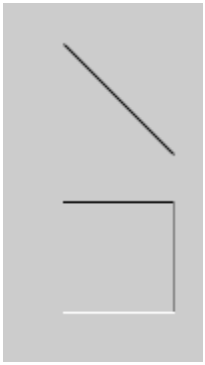
ellips (a, b, c, d)

Parameter-parameter

A float: x-coordinate of the arc's ellipse
B float: y-coordinate of the arc's ellipse
C float: width of the arc's ellipse by default
D float: height of the arc's ellipse by default

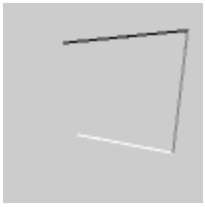
line()

Examples



```
line(30, 20, 85, 75);
```

```
line(30, 20, 85, 20);
stroke(126);
line(85, 20, 85, 75);
stroke(255);
line(85, 75, 30, 75);
```



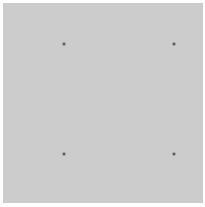
```
// Drawing lines in 3D requires P3D
// as a parameter to size()
size(100, 100, P3D);
line(30, 20, 0, 85, 20, 15);
stroke(126);
line(85, 20, 15, 85, 75, 0);
stroke(255);
line(85, 75, 0, 30, 75, -50);
```

Diskripsi

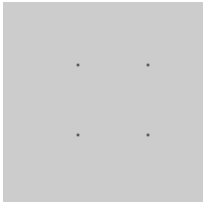
Menarik garis (jalur langsung antara dua titik) ke layar. Versi line () dengan empat parameter menarik garis dalam 2D. Untuk mewarnai garis, gunakan fungsi stroke () . Garis tidak dapat diisi, oleh karena itu fungsi fill () tidak akan mempengaruhi warna garis. Garis 2D digambar dengan lebar satu piksel secara default, tetapi ini dapat diubah dengan fungsi strokeWeight () . Versi dengan enam parameter memungkinkan garis ditempatkan di mana saja dalam ruang XYZ. Menggambar bentuk ini dalam 3D dengan parameter z membutuhkan parameter P3D dalam kombinasi dengan size () seperti yang ditunjukkan pada contoh di atas.

point()

Examples



```
noSmooth();
point(30, 20);
point(85, 20);
point(85, 75);
point(30, 75);
```



```
size(100, 100, P3D);
  noSmooth();
  point(30, 20, -50);
  point(85, 20, -50);
  point(85, 75, -50);
  point(30, 75, -50);
```

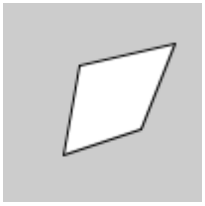
Diskripsi

Menarik titik, koordinat dalam ruang pada dimensi satu piksel. Parameter pertama adalah nilai horizontal untuk titik tersebut,

nilai kedua adalah nilai vertikal untuk titik tersebut, dan nilai ketiga opsional adalah nilai kedalaman. Menggambar bentuk ini dalam 3D dengan parameter z membutuhkan parameter P3D dalam kombinasi dengan size () seperti yang ditunjukkan pada contoh di atas. Gunakan stroke () untuk mengatur warna titik ().

quad()

Examples



quad(38, 31, 86, 20, 69, 63, 30, 76);

Diskripsi

Quad adalah segiempat, poligon empat sisi. Ini mirip dengan persegi panjang, tetapi sudut antara tepinya tidak dibatasi hingga sembilan puluh derajat. Pasangan parameter pertama (x1, y1) menetapkan titik pertama dan pasangan berikutnya harus berjalan searah atau berlawanan arah jarum jam di sekitar bentuk yang ditentukan.

Sintak

quad (x1, y1, x2, y2, x3, y3, x4, y4)

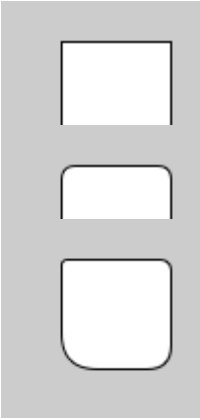
Parameter-parameter

X1 float: koordinat-x dari sudut pertama
Y1 float: koordinat-y dari sudut pertama
X2 float: koordinat-x dari sudut kedua
Y2 float: koordinat-y dari sudut kedua
X3 float: koordinat-x dari sudut ketiga

Y3 float: koordinat-y dari sudut ketiga
X4 float: koordinat-x dari sudut keempat
Y4 float: koordinat-y dari sudut keempat

rect()

Examples



rect(30, 20, 55, 55);

rect(30, 20, 55, 55, 7);

rect(30, 20, 55, 55, 3, 6, 12, 18);

Diskripsi

Menarik persegi panjang ke layar. Persegi panjang adalah bentuk empat sisi dengan setiap sudut pada Sembilan puluh derajat. Secara default, dua parameter pertama mengatur lokasi sudut kiri atas, parameter ketiga menetapkan lebar, dan parameter keempat menentukan ketinggian. Namun, cara parameter ini ditafsirkan, dapat diubah dengan funsi rectMode ().

Untuk menggambar persegi panjang bulat, tambahkan parameter kelima, yang digunakan sebagai nilai radius untuk keempat sudut.

Untuk menggunakan nilai radius berbeda untuk setiap sudut, sertakan delapan parameter. Saat menggunakan delapan parameter, empat yang terakhir mengatur jari-jari busur di

setiap sudut secara terpisah, dimulai dengan sudut kiri atas dan bergerak searah jarum jam di sekitar persegi panjang.

Sintak

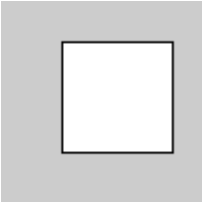
rect (a, b, c, d)
rect (a, b, c, d, r)
rect (a, b, c, d, tl, tr, br, bl)

Parameter-parameter

a	float: x-koordinat persegi panjang secara default
b	float: koordinat-y dari persegi panjang secara default
c	float: lebar persegi panjang secara default
d	float: tinggi persegi panjang secara default
r	float: jari-jari untuk keempat sudut
tl	float: radius untuk sudut kiri atas
tr	float: radius untuk sudut kanan atas
br	float: radius untuk sudut kanan bawah
bl	float: radius untuk sudut kiri bawah

square()

Examples



square(30, 20, 55);

Diskripsi

Menarik persegi ke layar. Kotak adalah bentuk empat sisi dengan setiap sudut pada sembilan puluh derajat dan setiap sisi

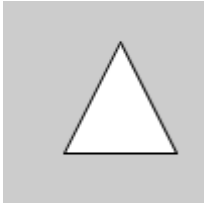
memiliki panjang yang sama. Secara default, dua parameter pertama mengatur lokasi sudut kiri atas, yang ketiga menetapkan lebar dan tinggi. Namun, cara parameter ini ditafsirkan, dapat diubah dengan fungsi rectMode ().

Sintak
square (x, y, extent)

Parameter-parameter
x float: x-koordinat persegi panjang secara default
y float: koordinat-y dari persegi panjang secara default
extent float: lebar dan tinggi persegi panjang secara default

triangle()

Examples



triangle(30, 75, 58, 20, 86, 75);

Diskripsi
Segitiga adalah bidang yang dibuat dengan menghubungkan tiga titik. Dua argumen pertama menentukan titik pertama, dua argumen tengah menentukan titik kedua, dan dua argumen terakhir menentukan titik ketiga.

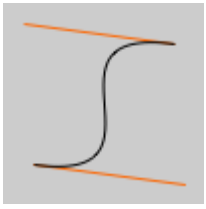
Sintak
triangle(x1, y1, x2, y2, x3, y3)

Parameter-parameter

X1 float: koordinat-x dari titik pertama
Y1 float: koordinat-y dari titik pertama
X2 float: koordinat-x dari titik kedua
Y2 float: koordinat-y dari titik kedua
X3 float: koordinat-x dari titik ketiga
Y3 float: koordinat-y dari titik ketiga

bezier()

Examples



```
noFill();
stroke(255, 102, 0);
line(85, 20, 10, 10);
line(90, 90, 15, 80);
stroke(0, 0, 0);
bezier(85, 20, 10, 10, 90, 90, 15, 80);
```

```
noFill();
stroke(255, 102, 0);
line(30, 20, 80, 5);
line(80, 75, 30, 75);
stroke(0, 0, 0);
bezier(30, 20,  80, 5,  80, 75,  30, 75);
```

Deskripsi

Menggambar kurva Bezier di layar. Kurva ini didefinisikan oleh serangkaian jangkar dan titik kontrol. Dua parameter pertama menentukan titik jangkar pertama dan dua parameter terakhir menentukan titik jangkar lainnya. Parameter tengah menentukan titik kontrol yang menentukan bentuk kurva. Kurva Bezier dikembangkan oleh insinyur Prancis Pierre Bezier.

Menggunakan versi 3D memerlukan rendering dengan P3D (lihat referensi Lingkungan untuk informasi lebih lanjut).

Sintak

bezier(x1, y1, x2, y2, x3, y3, x4, y4)
bezier(x1, y1, z1, x2, y2, z2, x3, y3, z3, x4, y4, z4)

Parameter-parameter

x1	float: koordinat untuk anchor point pertama
y1	float: koordinat untuk anchor point pertama
z1	float: koordinat untuk anchor point pertama
x2	float: koordinat untuk control point pertama
y2	float: koordinat untuk control point pertama
z2	float: koordinat untuk control point pertama
x3	float: koordinat untuk control point kedua
y3	float: koordinat untuk control point kedua
z3	float: koordinat untuk control point kedua
x4	float: koordinat untuk anchor point kedua
y4	float: koordinat untuk anchor point kedua
z4	float: koordinat untuk anchor point kedua

bezierDetail()

Example

// Gerakkan mouse ke kiri dan ke kanan untuk melihat perubahan detail

```
void setup() {
  size(100, 100, P3D);
  noFill();
}
```

```
void draw() {
  background(204);
  int d = int(map(mouseX, 0, 100, 1, 20));
  bezierDetail(d);
  bezier(85, 20, 10, 10, 90, 90, 15, 80);
}
```

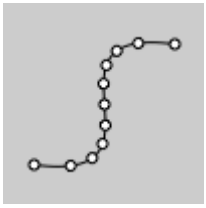
Diskripsi
Mengatur resolusi tempat Beziers ditampilkan. Nilai standarnya adalah 20. Fungsi ini hanya berguna saat menggunakan renderer P3D; penyaji P2D default tidak menggunakan informasi ini.

Sintak
bezierDetail(detail)

Parameter
Detail int: resolusi kurva

bezierPoint()

Examples



```
noFill();
bezier(85, 20, 10, 10, 90, 90, 15, 80);
fill(255);
int steps = 10;
for (int i = 0; i <= steps; i++) {
  float t = i / float(steps);
float x = bezierPoint(85, 10, 90, 15, t);
float y = bezierPoint(20, 10, 90, 80, t);
ellipse(x, y, 5, 5);
}
```

Diskripsi

Mengevaluasi Bezier pada titik t untuk poin a, b, c, d. Parameter t bervariasi antara 0 dan 1, a dan d adalah titik pada kurva, dan b dan c adalah titik kontrol. Ini dapat dilakukan sekali dengan koordinat x dan kedua kalinya dengan koordinat y untuk mendapatkan lokasi kurva bezier pada t.

Sintak

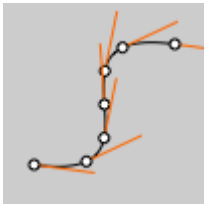
bezierPoint(a, b, c, d, t)

Parameter-parameter

a	float: koordinat titik pertama pada kurva
b	float: koordinat titik kontrol pertama
c	float: koordinat titik kontrol kedua
d	float: koordinat titik kedua pada kurva
t	float: nilai antara 0 dan 1

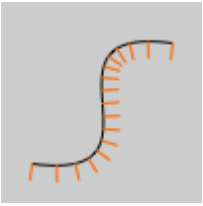
bezierTangent()

Examples



```
noFill();
bezier(85, 20, 10, 10, 90, 90, 15, 80);
int steps = 6;
fill(255);
for (int i = 0; i <= steps; i++) {
  float t = i / float(steps);
// Dapatkan lokasi intinya
float x = bezierPoint(85, 10, 90, 15, t);
float y = bezierPoint(20, 10, 90, 80, t);
// Dapatkan poin singgung
float tx = bezierTangent(85, 10, 90, 15, t);
```

```
  float ty = bezierTangent(20, 10, 90, 80, t);
  // Hitung sudut dari titik singgung
  float a = atan2(ty, tx);
  a += PI;
  stroke(255, 102, 0);
  line(x, y, cos(a)*30 + x, sin(a)*30 + y);
  // Baris kode berikut membuat garis
  // kebalikan dari baris di atas
  //line(x, y, cos(a)*-30 + x, sin(a)*-30 + y);
  stroke(0);
  ellipse(x, y, 5, 5);
}
```



```
noFill();
bezier(85, 20, 10, 10, 90, 90, 15, 80);
stroke(255, 102, 0);
int steps = 16;
for (int i = 0; i <= steps; i++) {
  float t = i / float(steps);
  float x = bezierPoint(85, 10, 90, 15, t);
  float y = bezierPoint(20, 10, 90, 80, t);
  float tx = bezierTangent(85, 10, 90, 15, t);
  float ty = bezierTangent(20, 10, 90, 80, t);
  float a = atan2(ty, tx);
  a -= HALF_PI;
  line(x, y, cos(a)*8 + x, sin(a)*8 + y);
}
```

Diskripsi

Menghitung garis singgung titik pada kurva Bezier. Ada definisi yang baik tentang garis singgung di Wikipedia.

Sintak

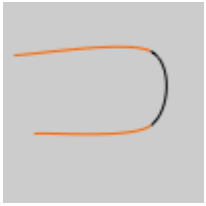
bezierTangent(a, b, c, d, t)

Parameter-parameter

a float: koordinat titik pertama pada kurva
b float: koordinat titik kontrol pertama
c float: koordinat titik kontrol kedua
d float: koordinat titik kedua pada kurva
t float: nilai antara 0 dan 1

curve()

Examples



```
noFill();
stroke(255, 102, 0);
curve(5, 26, 5, 26, 73, 24, 73, 61);
stroke(0);
curve(5, 26, 73, 24, 73, 61, 15, 65);
stroke(255, 102, 0);
curve(73, 24, 73, 61, 15, 65, 15, 65);
```

Diskripsi

Menarik garis melengkung di layar. Parameter pertama dan kedua menentukan titik kontrol awal dan dua parameter terakhir menentukan titik kontrol akhir. Parameter tengah menentukan awal dan berhenti kurva. Kurva yang lebih panjang dapat dibuat dengan menempatkan serangkaian fungsi curve () bersamaan atau menggunakan curveVertex (). Fungsi tambahan yang disebut curveTightness () menyediakan kontrol untuk kualitas visual kurva. Fungsi curve () adalah implementasi dari splines Catmull-Rom. Menggunakan versi 3D memerlukan rendering dengan P3D (lihat referensi Lingkungan untuk informasi lebih lanjut).

Sintak

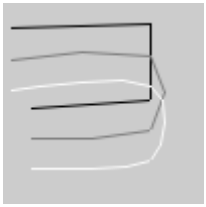
curve(x1, y1, x2, y2, x3, y3, x4, y4)
curve(x1, y1, z1, x2, y2, z2, x3, y3, z3, x4, y4, z4)

Parameter-parameter

x1	float: koordinat untuk titik kontrol awal
y1	float: koordinat untuk titik kontrol awal
x2	float: koordinat untuk titik pertama
y2	float: koordinat untuk titik pertama
x3	float: koordinat untuk titik kedua
y3	float: koordinat untuk titik kedua
x4	float: koordinat untuk titik kontrol akhir
y4	float: koordinat untuk titik kontrol akhir
z1	float: koordinat untuk titik pertama
z2	float: koordinat untuk titik pertama
z3	float: koordinat untuk titik kedua
z4	float: koordinat untuk titik kontrol akhir

curveDetail()

Examples



```
void setup() {
  size(100, 100, P3D);
  noFill();
  noLoop();
}

void draw() {
  curveDetail(1);
  drawCurves(-15);
  stroke(126);
  curveDetail(2);
```

```
  drawCurves(0);
  stroke(255);
  curveDetail(4);
  drawCurves(15);
}

void drawCurves(float y) {
  curve( 5, 28+y,  5, 28+y, 73, 26+y, 73, 63+y);
  curve( 5, 28+y, 73, 26+y, 73, 63+y, 15, 67+y);
  curve(73, 26+y, 73, 63+y, 15, 67+y, 15, 67+y);
}
```

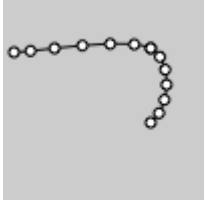
Diskripsi
Mengatur resolusi tampilan kurva. Nilai default adalah 20. Fungsi ini hanya berguna ketika menggunakan renderer P3D karena renderer P2D default tidak menggunakan informasi ini.

Sintak
curveDetail(detail)

Parameter-parameter
detailint: resolusi kurva

curvePoint()

Examples



```
noFill();
curve(5, 26, 5, 26, 73, 24, 73, 61);
curve(5, 26, 73, 24, 73, 61, 15, 65);
fill(255);
ellipseMode(CENTER);
int steps = 6;
```

```
for (int i = 0; i <= steps; i++) {
  float t = i / float(steps);
  float x = curvePoint(5, 5, 73, 73, t);
  float y = curvePoint(26, 26, 24, 61, t);
  ellipse(x, y, 5, 5);
  x = curvePoint(5, 73, 73, 15, t);
  y = curvePoint(26, 24, 61, 65, t);
  ellipse(x, y, 5, 5);
}
```

Diskripsi

Mengevaluasi kurva pada titik t untuk titik a, b, c, d. Parameter t dapat berkisar dari 0 (awal kurva) dan 1 (ujung kurva). a dan d adalah titik pada kurva, dan b dan c adalah titik kontrol. Ini dapat digunakan sekali dengan koordinat x dan kedua kalinya dengan koordinat y untuk mendapatkan lokasi kurva pada t.

Sintak

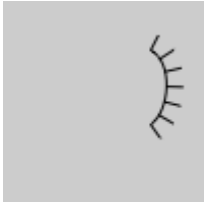
curvePoint(a, b, c, d, t)

Parameter-parameter

a	float: koordinat titik pertama pada kurva
b	float: koordinat titik kedua pada kurva
c	float: koordinat titik ketiga pada kurva
d	float: koordinat titik keempat pada kurva
t	float: nilai antara 0 dan 1

curveTangent()

Examples



```
noFill();
curve(5, 26, 73, 24, 73, 61, 15, 65);
int steps = 6;
for (int i = 0; i <= steps; i++) {
  float t = i / float(steps);
  float x = curvePoint(5, 73, 73, 15, t);
float y = curvePoint(26, 24, 61, 65, t);
//ellipse(x, y, 5, 5);
float tx = curveTangent(5, 73, 73, 15, t);
float ty = curveTangent(26, 24, 61, 65, t);
float a = atan2(ty, tx);
a -= PI/2.0;
line(x, y, cos(a)*8 + x, sin(a)*8 + y);
}
```

Diskripsi

Menghitung garis singgung titik pada kurva. Ada definisi yang baik tentang garis singgung di Wikipedia.

Sintak

curveTangent(a, b, c, d, t)

Parameter-parameter

a	float: koordinat titik pertama pada kurva
b	float: koordinat titik kedua pada kurva
c	float: koordinat titik ketiga pada kurva
d	float: koordinat titik keempat pada kurva
t	float: nilai antara 0 dan 1

curveTightness()

Examples

```
// Gerakkan mouse ke kiri dan ke kanan untuk melihat perubahan kurva

void setup() {
  size(100, 100);
  noFill();
}

void draw() {
  background(204);
  float t = map(mouseX, 0, width, -5, 5);
  curveTightness(t);
  beginShape();
  curveVertex(10, 26);
  curveVertex(10, 26);
  curveVertex(83, 24);
  curveVertex(83, 61);
  curveVertex(25, 65);
  curveVertex(25, 65);
  endShape();
```

Diskripsi

Mengubah kualitas formulir yang dibuat dengan curve () dan curveVertex (). Parameter tightness menentukan bagaimana kurva cocok dengan titik titik. Nilai 0,0 adalah nilai default untuk tightness (nilai ini mendefinisikan kurva sebagai splines Catmull-Rom) dan nilai 1.0 menghubungkan semua titik dengan garis lurus. Nilai dalam rentang -5.0 dan 5.0 akan merusak kurva tetapi akan membuat mereka dikenali dan

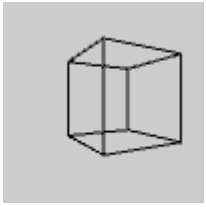
ketika nilai-nilai meningkat dalam besarnya, mereka akan terus berubah bentuk.

Sintak
curveTightness(tightness)

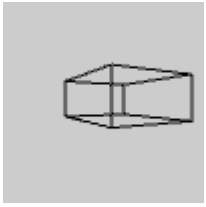
Parameter-parameter
tightness float: jumlah deformasi dari simpul asli

box()

Examples



```
size(100, 100, P3D);
translate(58, 48, 0);
rotateY(0.5);
noFill();
box(40);
```



```
size(100, 100, P3D);
translate(58, 48, 0);
rotateY(0.5);
noFill();
box(40, 20, 50);
```

Diskripsi
Sebuah kotak adalah kotak yang diekstrusi. Sebuah kotak dengan dimensi yang sama di semua sisi adalah sebuah kubus.

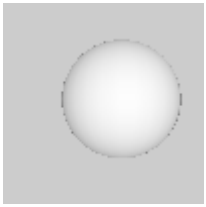
Sintak
box(size)
box(w, h, d)

Parameter-parameter

size	float: dimensi kotak di semua dimensi (membuat kubus)
w	float: dimensi kotak dalam dimensi x
h	float: dimensi kotak dalam dimensi y
d	float: dimensi kotak dalam dimensi z

sphere()

Examples



```
noStroke();
lights();
translate(58, 48, 0);
sphere(28);
```

Diskripsi

Bola adalah bola berlubang yang terbuat dari segitiga tessellated.

Sintak

sphere(r)

Parameter-parameter

r float: jari-jari bola

sphereDetail()

Examples

```
void setup() {
  size(100, 100, P3D);
```

```
}

void draw() {
  background(200);
  stroke(255, 50);
  translate(50, 50, 0);
  rotateX(mouseY * 0.05);
  rotateY(mouseX * 0.05);
  fill(mouseX * 2, 0, 160);
  sphereDetail(mouseX / 4);
  sphere(40);
}
```

Diskripsi

Kontrol detail yang digunakan untuk membuat bola dengan menyesuaikan jumlah simpul dari jaring bola. Resolusi default adalah 30, yang menciptakan definisi lingkup yang cukup rinci dengan simpul setiap 360/30 = 12 derajat. Jika Anda akan membuat banyak bola per frame, disarankan untuk mengurangi tingkat detail menggunakan fungsi ini. Pengaturan tetap aktif hingga sphereDetail () dipanggil lagi dengan parameter baru dan karenanya tidak boleh dipanggil sebelum setiap pernyataan sphere (), kecuali jika Anda ingin membuat bola dengan pengaturan yang berbeda, mis. menggunakan lebih sedikit detail untuk bola yang lebih kecil atau yang jauh dari kamera. Untuk mengontrol detail resolusi horizontal dan vertikal secara independen, gunakan versi fungsi dengan dua parameter.

Sintak

sphereDetail(res)
sphereDetail(ures, vres)

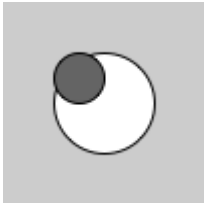
Parameter-parameter

res int: jumlah segmen (minimal 3) yang digunakan per revolusi lingkaran penuh
ures int: jumlah segmen yang digunakan secara longitudinal per putaran lingkaran penuh
vres int: jumlah segmen yang digunakan secara latitudinal dari atas ke bawah

ellipseMode()

Examples

```
ellipseMode(RADIUS);  // Set ellipseMode ke RADIUS
fill(255);  // Atur isi menjadi putih
ellipse(50, 50, 30, 30);  // Gambar elips putih menggunakan mode RADIUS

ellipseMode(CENTER);  // Set ellipseMode ke CENTER
fill(100);  // Atur isi menjadi abu-abu
ellipse(50, 50, 30, 30);  // Gambar elips abu-abu menggunakan mode CENTER
```



```
ellipseMode(CORNER);  // Set ellipseMode adalah CORNER
fill(255);  // Atur isi menjadi putih
ellipse(25, 25, 50, 50);  // Gambar elips putih menggunakan mode CORNER

ellipseMode(CORNERS);  // Set ellipseMode ke CORNERS
fill(100);  // Atur isi menjadi abu-abu
```

```
ellipse(25, 25, 50, 50);  // Gambar ellipse abu-abu menggunakan mode CORNERS
```

Diskripsi

Mengubah lokasi dari mana elips ditarik dengan mengubah cara parameter yang diberikan kepada ellipse () dimasukkan.
Mode default adalah ellipseMode (CENTER), yang mengartikan dua parameter pertama ellipse () sebagai titik pusat bentuk, sedangkan parameter ketiga dan keempat adalah lebar dan tinggi.
ellipseMode (RADIUS) juga menggunakan dua parameter pertama ellipse () sebagai titik pusat bentuk, tetapi menggunakan parameter ketiga dan keempat untuk menentukan setengah dari lebar dan tinggi bentuk.

ellipseMode (CORNER) mengartikan dua parameter pertama ellipse () sebagai sudut kiri atas bentuk, sedangkan parameter ketiga dan keempat adalah lebar dan tinggi.
ellipseMode (CORNERS) mengartikan dua parameter pertama dari ellipse () sebagai lokasi satu sudut kotak elips yang terikat, dan parameter ketiga dan keempat sebagai lokasi sudut yang berlawanan.
Parameter harus ditulis dalam ALL CAPS karena Pemrosesan adalah bahasa yang case-sensitive.

Sintak

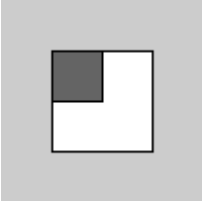
ellipseMode(mode)

Parameter-parameter

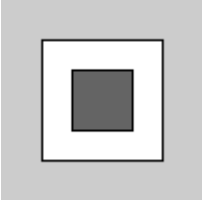
mode int: CENTER, RADIUS, CORNER, atau CORNERS

rectMode()

Examples



```
rectMode(CORNER);  // Rectmode default adalah CORNER
fill(255);  // Atur isi menjadi putih
rect(25, 25, 50, 50);  // Gambar kotak putih menggunakan mode CORNER

rectMode(CORNERS);  // Atur rectMode ke CORNERS
fill(100);  // Atur isi menjadi abu-abu
rect(25, 25, 50, 50);  // Gambar kotak abu-abu menggunakan mode CORNERS
```



```
rectMode(RADIUS);  // Atur rectMode ke RADIUS
fill(255);  // Atur isi menjadi putih
rect(50, 50, 30, 30);  // Gambar kotak putih menggunakan mode RADIUS

rectMode(CENTER);  // Atur rectMode ke CENTER
fill(100);  // Atur isi menjadi abu-abu
rect(50, 50, 30, 30);  // Gambar abu-abu persegi menggunakan mode PUSAT
```

Diskripsi

Memodifikasi lokasi dari mana persegi panjang diambil dengan mengubah cara parameter yang given to rect () dimasukkan.Mode default adalah rectMode (CORNER), yang mengartikan dua parameter pertama dari rect () sebagai sudut

kiri atas bentuk, sedangkan parameter ketiga dan keempat adalah lebar dan tinggi.
rectMode (CORNERS) mengartikan dua parameter pertama dari rect () sebagai lokasi satu sudut, dan parameter ketiga dan keempat sebagai lokasi sudut yang berlawanan.
rectMode (CENTER) menginterpretasikan dua parameter pertama dari rect () sebagai titik pusat bentuk, sedangkan parameter ketiga dan keempat adalah lebar dan tinggi.
rectMode (RADIUS) juga menggunakan dua parameter pertama dari rect () sebagai titik pusat bentuk, tetapi menggunakan parameter ketiga dan keempat untuk menentukan setengah dari lebar dan tinggi bentuk.
Parameter harus ditulis dalam SALL CAPS karena Pemrosesan adalah bahasa yang case-sensitive.

Sintak
rectMode(mode)

Parameter-parameter
mode int: CORNER, CORNERS, CENTER, atau RADIUS

strokeCap()

Examples

```
strokeWeight(12.0);
strokeCap(ROUND);
line(20, 30, 80, 30);
strokeCap(SQUARE);
line(20, 50, 80, 50);
strokeCap(PROJECT);
line(20, 70, 80, 70);
```

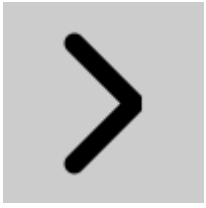
Diskripsi
Setel gaya untuk merender akhir baris. Ujung-ujung ini dapat dikuadratkan, diperpanjang, atau dibulatkan, masing-masing ditentukan dengan parameter yang sesuai: SQUARE, PROJECT, dan ROUND. Batas default adalah ROUND.

Sintak
strokeCap(cap)

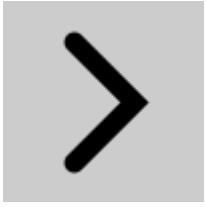
Parameter-parameter
cap int: SQUARE, PROJECT, atau ROUND

strokeJoin()

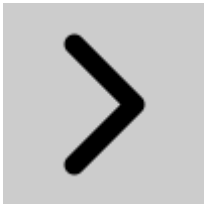
Examples



```
noFill();
strokeWeight(10.0);
strokeJoin(MITER);
beginShape();
vertex(35, 20);
vertex(65, 50);
vertex(35, 80);
endShape();
```



```
noFill();
strokeWeight(10.0);
strokeJoin(BEVEL);
beginShape();
vertex(35, 20);
vertex(65, 50);
vertex(35, 80);
endShape();
```



```
noFill();
strokeWeight(10.0);
strokeJoin(ROUND);
beginShape();
vertex(35, 20);
vertex(65, 50);
vertex(35, 80);
endShape();
```

Diskripsi

Mengatur gaya sambungan yang menghubungkan segmen garis. Sambungan-sambungan ini disatukan, miring, atau dibulatkan dan ditentukan dengan parameter yang sesuai MITER, BEVEL, dan ROUND. Sambungan default adalah MITER.

Sintak

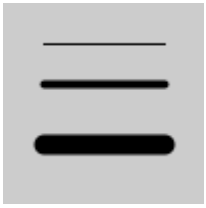
strokeJoin(join)

Parameter-parameter

join int: MITER, BEVEL,atau ROUND

strokeWeight()

Examples



```
strokeWeight(1);  // Default
line(20, 20, 80, 20);
strokeWeight(4);  // Thicker
line(20, 40, 80, 40);
strokeWeight(10);  // Beastly
line(20, 70, 80, 70);
```

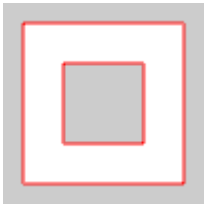
Diskripsi
Mengatur lebar goresan yang digunakan untuk garis, titik, dan batas di sekitar bentuk. Semua lebar diatur dalam satuan piksel.

Sintak
strokeWeight(weight)

Parameter-parameter
weight float: berat (dalam piksel) dari pukulan

beginContour()

Examples



```
size(100, 100);
translate(50, 50);
stroke(255, 0, 0);
beginShape();
// Bagian luar bentuk, searah jarum jam
vertex(-40, -40);
vertex(40, -40);
vertex(40, 40);
vertex(-40, 40);
// Bagian dalam bentuk, berliku berlawanan arah jarum jam
beginContour();
vertex(-20, -20);
vertex(-20, 20);
vertex(20, 20);
vertex(20, -20);
endContour();
endShape(CLOSE);
```

Diskripsi

Gunakan fungsi beginContour () dan endContour () untuk membuat bentuk negatif di dalam bentuk seperti pusat huruf 'O'. beginContour () mulai merekam simpul untuk bentuk dan endContour () berhenti merekam. Verteks yang menentukan bentuk negatif harus "berputar" ke arah yang berlawanan dari bentuk eksterior. Pertama menggambar simpul untuk bentuk eksterior dalam urutan searah jarum jam, kemudian untuk bentuk internal, menggambar simpul berlawanan arah jarum jam.

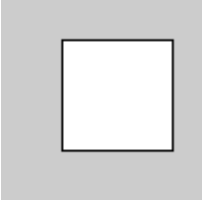
Fungsi-fungsi ini hanya dapat digunakan dalam pasangan beginShape () / endShape () dan transformasi seperti menerjemahkan (), memutar (), dan skala () tidak berfungsi dalam pasangan beginContour () / endContour (). Juga tidak mungkin untuk menggunakan bentuk lain, seperti ellipse () atau rect () di dalamnya.

Sintak

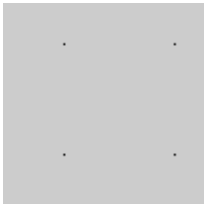
beginContour()

beginShape()

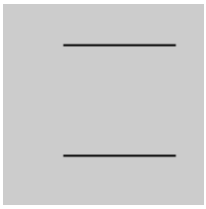
Examples



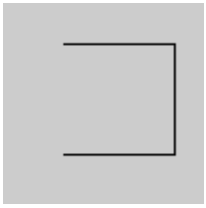
```
beginShape();
vertex(30, 20);
vertex(85, 20);
vertex(85, 75);
vertex(30, 75);
endShape(CLOSE);
```



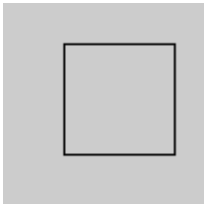
```
beginShape(POINTS);
vertex(30, 20);
vertex(85, 20);
vertex(85, 75);
vertex(30, 75);
endShape();
```



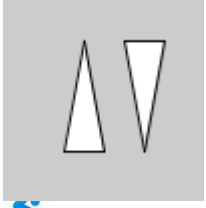
```
beginShape(LINES);
vertex(30, 20);
vertex(85, 20);
vertex(85, 75);
vertex(30, 75);
endShape();
```



```
noFill();
beginShape();
vertex(30, 20);
vertex(85, 20);
vertex(85, 75);
vertex(30, 75);
endShape();
```



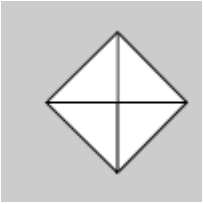
```
noFill();
beginShape();
vertex(30, 20);
vertex(85, 20);
vertex(85, 75);
vertex(30, 75);
endShape(CLOSE);
```



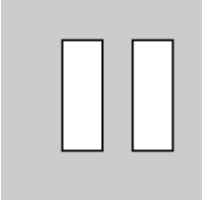
```
beginShape(TRIANGLES);
vertex(30, 75);
vertex(40, 20);
vertex(50, 75);
```

```
vertex(60, 20);
vertex(70, 75);
vertex(80, 20);
endShape();
```

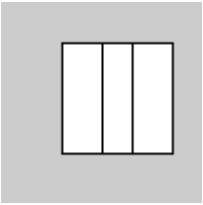
```
beginShape(TRIANGLE_STRIP);
vertex(30, 75);
vertex(40, 20);
vertex(50, 75);
vertex(60, 20);
vertex(70, 75);
vertex(80, 20);
vertex(90, 75);
endShape();
```



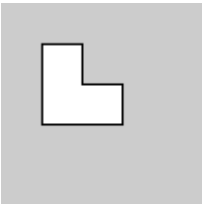
```
beginShape(TRIANGLE_FAN);
vertex(57.5, 50);
vertex(57.5, 15);
vertex(92, 50);
vertex(57.5, 85);
vertex(22, 50);
vertex(57.5, 15);
endShape();
```



```
beginShape(QUADS);
vertex(30, 20);
vertex(30, 75);
vertex(50, 75);
vertex(50, 20);
vertex(65, 20);
vertex(65, 75);
vertex(85, 75);
vertex(85, 20);
endShape();
```



```
beginShape(QUAD_STRIP);
vertex(30, 20);
vertex(30, 75);
vertex(50, 20);
vertex(50, 75);
vertex(65, 20);
vertex(65, 75);
vertex(85, 20);
vertex(85, 75);
endShape();
```



```
beginShape();
vertex(20, 20);
vertex(40, 20);
vertex(40, 40);
vertex(60, 40);
vertex(60, 60);
vertex(20, 60);
endShape(CLOSE);
```

Diskripsi

Menggunakan fungsi beginShape () dan endShape () memungkinkan membuat formulir yang lebih kompleks. beginShape () mulai merekam simpul untuk bentuk dan endShape () berhenti merekam. Nilai dari parameter jenis memberitahu jenis bentuk yang dibuat dari simpul yang disediakan. Tanpa mode yang ditentukan, bentuknya bisa berupa poligon beraturan. Parameter yang tersedia untuk beginShape () adalah POINTS, LINES, TRIANGLES, TRIANGLE_FAN, TRIANGLE_STRIP, QUADS, dan QUAD_STRIP. Setelah memanggil fungsi beginShape (), serangkaian perintah vertex () harus mengikuti. Untuk berhenti menggambar bentuk, panggil endShape (). Fungsi vertex ()

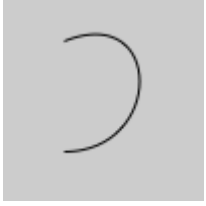
dengan dua parameter menentukan posisi dalam 2D dan fungsi vertex () dengan tiga parameter menentukan posisi dalam 3D. Setiap bentuk akan diuraikan dengan warna goresan saat ini dan diisi dengan warna isian.
Transformasi seperti translate (), rotate (), dan scale () tidak berfungsi dalam beginShape (). Juga tidak mungkin menggunakan bentuk lain, seperti ellipse () atau rect () dalam beginShape ().
Penyaji P2D dan P3D memungkinkan stroke () dan fill () diubah pada basis per-simpul, tetapi penyaji default tidak. Pengaturan seperti strokeWeight (), strokeCap (), dan strokeJoin () tidak dapat diubah saat berada di dalam blok beginShape () / endShape () dengan renderer apa pun

Sintak
beginShape()
beginShape(kind)

Parameter-parameter
kind int:POINTS, LINES, TRIANGLES, TRIANGLE_FAN, TRIANGLE_STRIP, QUADS, atau QUAD_STRIP

bezierVertex()

Examples



```
noFill();
beginShape();
vertex(30, 20);
bezierVertex(80, 0, 80, 75, 30, 75);
endShape();
```

```
beginShape();
vertex (30, 20);
bezierVertex(80, 0, 80, 75, 30, 75);
bezierVertex(50, 80, 60, 25, 30, 20);
endShape();
```

Diskripsi

Menentukan koordinat titik untuk kurva Bezier. Setiap panggilan ke bezierVertex () mendefinisikan posisi dua titik kontrol dan satu titik jangkar dari kurva Bezier, menambahkan segmen baru ke garis atau bentuk. bezierVertex () pertama kali digunakan dalam panggilan beginShape (), harus diawali dengan panggilan ke vertex () untuk mengatur titik jangkar pertama. Fungsi ini harus digunakan antara beginShape () dan endShape () dan hanya ketika tidak ada parameter MODE yang ditentukan untuk beginShape (). Menggunakan versi 3D memerlukan rendering dengan P3D (lihat referensi Lingkungan untuk informasi lebih lanjut).

Sintak

bezierVertex(x2, y2, x3, y3, x4, y4)
bezierVertex(x2, y2, z2, x3, y3, z3, x4, y4, z4)

Parameter-parameter

x2	float: koordinat x dari titik kontrol pertama
y2	float: koordinat y dari titik kontrol pertama
z2	float: koordinat-z dari titik kontrol pertama
x3	float: koordinat x dari titik kontrol kedua
y3	float: koordinat-y dari titik kontrol kedua
z3	float: koordinat-z dari titik kontrol kedua
x4	float: koordinat x dari titik jangkar
y4	float: koordinat-y dari titik jangkar
z4	float: koordinat-z dari titik jangkar

curveVertex()

Examples

```
noFill();
beginShape();
curveVertex(84,  91);
curveVertex(84,  91);
curveVertex(68,  19);
curveVertex(21,  17);
curveVertex(32, 100);
curveVertex(32, 100);
endShape();
```

Diskripsi

Menentukan koordinat titik untuk kurva. Fungsi ini hanya dapat digunakan antara beginShape () dan endShape () dan hanya ketika tidak ada parameter MODE yang ditentukan untuk beginShape (). Poin pertama dan terakhir dalam serangkaian garis curveVertex () akan digunakan untuk memandu awal dan akhir kurva. Minimal empat poin diperlukan untuk menggambar kurva kecil antara titik kedua dan ketiga. Menambahkan titik kelima dengan curveVertex () akan menggambar kurva antara titik kedua, ketiga, dan keempat. Fungsi curveVertex () adalah implementasi dari splines Catmull-Rom. Menggunakan versi 3D memerlukan rendering dengan P3D (lihat referensi Lingkungan untuk informasi lebih lanjut).

Sintak

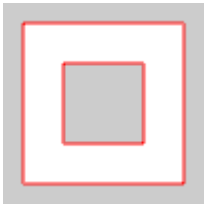
curveVertex(x, y)
curveVertex(x, y, z)

Parameter-parameter

x float: koordinat-x dari vertex
y float: koordinat-y dari vertex
z float: koordinat-z dari simpul

endContour()

Examples



```
size(100, 100);
translate(50, 50);
stroke(255, 0, 0);
beginShape();
// Bagian luar bentuk, berliku searah jarum jam
vertex(-40, -40);
vertex(40, -40);
vertex(40, 40);
vertex(-40, 40);
// Bagian dalam bentuk, berliku berlawanan arah jarum jam
beginContour();
vertex(-20, -20);
vertex(-20, 20);
vertex(20, 20);
vertex(20, -20);
endContour();
endShape(CLOSE);
```

Diskripsi

Gunakan fungsi beginContour () dan endContour () untuk membuat bentuk negatif di dalam bentuk seperti pusat huruf 'O'. beginContour () mulai merekam simpul untuk bentuk dan endContour () berhenti merekam. Verteks yang menentukan

bentuk negatif harus "berputar" ke arah yang berlawanan dari bentuk eksterior. Pertama menggambar simpul untuk bentuk eksterior dalam urutan searah jarum jam, kemudian untuk bentuk internal, menggambar simpul berlawanan arah jarum jam.

Fungsi-fungsi ini hanya dapat digunakan dalam pasangan beginShape () / endShape () dan transformasi seperti menerjemahkan (), memutar (), dan skala () tidak berfungsi dalam pasangan beginContour () / endContour (). Juga tidak mungkin untuk menggunakan bentuk lain, seperti ellipse () atau rect () di dalamnya.

Sintak

endContour()

endShape()

Examples

```
noFill();
beginShape();
vertex(20, 20);
vertex(45, 20);
vertex(45, 80);
endShape(CLOSE);

beginShape();
vertex(50, 20);
vertex(75, 20);
vertex(75, 80);
endShape();
```

Diskripsi
Fungsi endShape () adalah pendamping untuk beginShape () dan hanya dapat dipanggil setelah beginShape (). Ketika endshape () dipanggil, semua data gambar yang ditentukan sejak panggilan sebelumnya ke beginShape () ditulis ke dalam buffer gambar. Konstanta CLOSE sebagai nilai untuk parameter MODE untuk menutup bentuk (untuk menghubungkan awal dan akhir).

Sintak
endShape()
endShape(mode)

Parameter-parameter
mode int: gunakan CLOSE untuk menutup bentuk

quadraticVertex()

Examples

```
noFill();
strokeWeight(4);
beginShape();
vertex(20, 20);
quadraticVertex(80, 20, 50, 50);
endShape();
```

```
noFill();
strokeWeight(4);
beginShape();
vertex(20, 20);
quadraticVertex(80, 20, 50, 50);
quadraticVertex(20, 80, 80, 80);
```

```
vertex(80, 60);
endShape();
```

Diskripsi

Menentukan koordinat titik untuk kurva Bezier kuadratik. Setiap panggilan ke quadraticVertex () mendefinisikan posisi satu titik kontrol dan satu titik jangkar dari kurva Bezier, menambahkan segmen baru ke garis atau bentuk. quadraticVertex () pertama kali digunakan dalam panggilan beginShape (), harus diawali dengan panggilan ke vertex () untuk mengatur titik jangkar pertama. Fungsi ini harus digunakan antara beginShape () dan endShape () dan hanya ketika tidak ada parameter MODE yang ditentukan untuk beginShape (). Menggunakan versi 3D memerlukan rendering dengan P3D (lihat referensi Lingkungan untuk informasi lebih lanjut).

Sintak

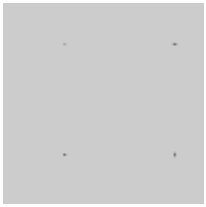
quadraticVertex(cx, cy, x3, y3)
quadraticVertex(cx, cy, cz, x3, y3, z3)

Parameter-parameter

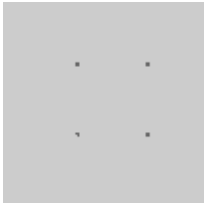
cx	float: koordinat x dari titik kontrol
cy	float: koordinat-y dari titik kontrol
x3	float: koordinat x dari titik jangkar
y3	float: koordinat-y dari titik jangka
cz	float: koordinat-z dari titik kontro
z3	float: koordinat-z dari titik jangkar

vertex()

Examples



```
beginShape(POINTS);
vertex(30, 20);
vertex(85, 20);
vertex(85, 75);
vertex(30, 75);
endShape();
```



```
// Menggambar simpul dalam 3D membutuhkan P3D
// sebagai parameter untuk ukuran ()
size(100, 100, P3D);
beginShape(POINTS);
vertex(30, 20, -50);
vertex(85, 20, -50);
vertex(85, 75, -50);
vertex(30, 75, -50);
endShape();
size(100, 100, P3D);
PImage img = loadImage("laDefense.jpg");
noStroke();
beginShape();
texture(img);
// "laDefense.jpg" berukuran 100x100 piksel
// nilai 0 dan 100 digunakan untuk
// parameter "u" dan "v" untuk memetakannya secara langsung
// ke titik titik
vertex(10, 20, 0, 0);
vertex(80, 5, 100, 0);
vertex(95, 90, 100, 100);
```

```
vertex(40, 95, 0, 100);
endShape();
```

Diskripsi

Semua bentuk dibangun dengan menghubungkan serangkaian simpul. vertex () digunakan untuk menentukan koordinat titik untuk titik, garis, segitiga, paha depan, dan poligon. Ini digunakan secara eksklusif dalam fungsi beginShape () dan endShape ().

Menggambar simpul dalam 3D menggunakan parameter z membutuhkan parameter P3D dalam kombinasi dengan ukuran, seperti yang ditunjukkan pada contoh di atas.

Fungsi ini juga digunakan untuk memetakan tekstur ke geometri. Fungsi tekstur () menyatakan tekstur untuk diterapkan pada geometri dan set koordinat u dan v menentukan pemetaan tekstur ini ke formulir. Secara default, koordinat yang digunakan untuk u dan v ditentukan dalam kaitannya dengan ukuran gambar dalam piksel, tetapi hubungan ini dapat diubah dengan teksturMode ().

Sintak

```
vertex(x, y)
vertex(x, y, z)
vertex(v)
vertex(x, y, u, v)
vertex(x, y, z, u, v)
```

Parameter-parameter

v	float[]:parameter vertex, sebagai array float dengan panjang VERTEX_FIELD_COUNT
x	float: koordinat-x dari vertex
y	float: koordinat-y dari vertex
z	float: koordinat-z dari vertex
u	float: koordinat horisontal untuk pemetaan tekstur

v float: koordinat vertikal untuk pemetaan tekstur

shape()

Examples

```
PShape s;
void setup() {
  s = loadShape("bot.svg");
}

void draw() {
  shape(s, 10, 10, 80, 80);
}
```

Diskripsi

Menarik bentuk ke jendela tampilan. Bentuk harus ada di direktori "data" sketsa untuk memuat dengan benar. Pilih "Tambah file ..." dari menu "Sketsa" untuk menambahkan bentuk. Pemrosesan saat ini berfungsi dengan SVG, OBJ, dan bentuk yang dibuat khusus. Parameter bentuk menentukan bentuk untuk ditampilkan dan parameter koordinat menentukan lokasi bentuk dari sudut kiri atas. Bentuk ditampilkan pada ukuran aslinya kecuali parameter c dan d menentukan ukuran yang berbeda. Fungsi shapeMode () dapat digunakan untuk mengubah cara parameter ini ditafsirkan.

Sintak

shape(shape)
shape(shape, x, y)
shape(shape, a, b, c, d)

Parameter-parameter

Shape PShape: bentuk untuk ditampilkan
x float: koordinat-x dari bentuk
y float: koordinat-y dari bentuk
a float: koordinat-x dari bentuk
b float: koordinat-y dari bentuk
c float: lebar untuk menampilkan bentuk
d float: tinggi untuk menampilkan bentuk

shapeMode()

Examples

```
PShape bot
void setup() {
  size(100, 100);
  bot = loadShape("bot.svg");
}

void draw() {
  shapeMode(CENTER);
  shape(bot, 35, 35, 50, 50);
  shapeMode(CORNER);
  shape(bot, 35, 35, 50, 50);
}
```

Diskripsi

Memodifikasi lokasi dari mana bentuk menggambar. Mode default adalah shapeMode (CORNER), yang menentukan lokasi sebagai sudut kiri atas bentuk dan menggunakan parameter ketiga dan keempat dari shape () untuk menentukan lebar dan tinggi. Sintaks shapeMode (CORNERS) menggunakan parameter pertama dan kedua shape () untuk mengatur lokasi

satu sudut dan menggunakan parameter ketiga dan keempat untuk mengatur sudut yang berlawanan. Sintak shapeMode (CENTER) menggambar bentuk dari titik pusatnya dan menggunakan parameter ketiga dan keempat dari shape () untuk menentukan lebar dan tinggi. Parameter harus ditulis dalam "ALL CAPS" karena Memproses adalah bahasa yang sensitif huruf.

Sintak
shapeMode(mode)

Parameter-parameter
mode int: baik CORNER, CORNERS, CENTER

BAB XV
mouseButton

mouseButton

Examples

```
// Klik di dalam gambar dan tekan
// tombol kiri dan kanan mouse untuk
// ubah nilai dari persegi panjang
void draw() {
  if (mousePressed && (mouseButton == LEFT)) {
    fill(0);
  } else if (mousePressed && (mouseButton == RIGHT)) {
    fill(255);
  } else {
    fill(126);
  }
  rect(25, 25, 50, 50);
}
```

```
// Klik di dalam gambar dan tekan
// tombol kiri dan kanan mouse untuk
// ubah nilai dari persegi panjang
void draw() {
  rect(25, 25, 50, 50);
}

void mousePressed() {
  if (mouseButton == LEFT) {
    fill(0);
  } else if (mouseButton == RIGHT) {
    fill(255);
  } else {
    fill(126);
  }
```


}

Diskripsi
Saat tombol mouse ditekan, nilai variabel sistem mouseButton diatur ke LEFT, RIGHT, atau CENTER, tergantung pada tombol mana yang ditekan. (Jika tidak ada tombol yang ditekan, mouseButton dapat diatur ulang ke 0. Untuk alasan itu, yang terbaik adalah menggunakan mousePressed terlebih dahulu untuk menguji apakah ada tombol yang ditekan, dan hanya kemudian menguji nilai mouseButton, seperti yang ditunjukkan pada contoh di atas.)

mouseClicked()

Examples

```
// Klik di dalam gambar untuk berubah
// nilai dari persegi panjang setelah
// setelah mouse diklik

int value = 0;

void draw() {
  fill(value);
  rect(25, 25, 50, 50);
}

void mouseClicked() {
  if (value == 0) {
    value = 255;
  } else {
    value = 0;
  }
```

```
}
```

Diskripsi

Fungsi mouseClicked () dipanggil setelah tombol mouse ditekan dan kemudian dilepaskan.

Acara mouse dan keyboard hanya berfungsi ketika suatu program memiliki draw (). Tanpa draw (), kode hanya dijalankan sekali dan kemudian berhenti mendengarkan acara.

Sintak

mouseClicked()
mouseClicked(event)

mouseDragged()

Examples

```
// Seret (klik dan tahan) mouse Anda melintasi
// gambar untuk mengubah nilai persegi panjang

int value = 0;

void draw() {
  fill(value);
  rect(25, 25, 50, 50);
}

void mouseDragged()
{
  value = value + 5;
  if (value > 255) {
    value = 0;
  }
```


MouseButton

}

Diskripsi

Fungsi mouseDragged () dipanggil sekali setiap kali mouse bergerak saat tombol mouse ditekan. (Jika tombol tidak ditekan, mouseMoved () dipanggil sebagai gantinya.)
Acara mouse dan keyboard hanya berfungsi ketika suatu program memiliki draw (). Tanpa draw (), kode hanya dijalankan sekali dan kemudian berhenti mendengarkan acara.

Sintak

mouseDragged()
mouseDragged(event)

mouseMoved()

Examples

```
// Gerakkan mouse melintasi gambar
// untuk mengubah nilainya

int value = 0;

void draw() {
  fill(value);
  rect(25, 25, 50, 50);
}

void mouseMoved() {
  value = value + 5;
  if (value > 255) {
    value = 0;
  }
```

```
}
```

Diskripsi

Fungsi mouseMoved () dipanggil setiap kali mouse bergerak dan tombol mouse tidak ditekan. (Jika tombol sedang ditekan, mouseDragged () dipanggil sebagai gantinya.)
Acara mouse dan keyboard hanya berfungsi ketika suatu program memiliki draw (). Tanpa draw (), kode hanya dijalankan sekali dan kemudian berhenti mendengarkan acara.

Sintak

mouseMoved()
mouseMoved(event)

mousePressed()

Examples

```
// Klik di dalam gambar untuk berubah
// nilai dari persegi panjang
int value = 0;

void draw() {
  fill(value);
  rect(25, 25, 50, 50);
}

void mousePressed() {
  if (value == 0) {
    value = 255;
  } else {
    value = 0;
  }
```


MouseButton

}

Diskripsi

Fungsi mousePressed () dipanggil satu kali setelah setiap kali tombol mouse ditekan. Variabel mouseButton (lihat entri referensi terkait) dapat digunakan untuk menentukan tombol mana yang telah ditekan.

Acara mouse dan keyboard hanya berfungsi ketika suatu program memiliki draw (). Tanpa draw (), kode hanya dijalankan sekali dan kemudian berhenti mendengarkan acara.

Sintak

mousePressed()

mousePressed(event)

mousePressed

Examples

```
// Klik di dalam gambar untuk berubah
// nilai dari persegi panjang
void draw() {
  if (mousePressed == true) {
    fill(0);
  } else {
    fill(255);
  }
  rect(25, 25, 50, 50);
}
```

Diskripsi

Variabel mousePressed menyimpan apakah tombol mouse sedang ditekan atau tidak. Nilainya benar ketika tombol mouse

ditekan, dan false jika tidak ada tombol yang ditekan. Variabel mouseButton (lihat entri referensi terkait) dapat digunakan untuk menentukan tombol mana yang telah ditekan.

mouseReleased()

Examples

```
// Klik di dalam gambar untuk berubah
// nilai dari persegi panjang
int value = 0;

void draw() {
  fill(value);
  rect(25, 25, 50, 50);
}
void mouseReleased() {
  if (value == 0) {
    value = 255;
  } else {
    value = 0;
  }
}
```

Diskripsi

Fungsi mouseReleased () dipanggil setiap kali tombol mouse dilepaskan.

Acara mouse dan keyboard hanya berfungsi ketika suatu program memiliki draw (). Tanpa draw (), kode hanya dijalankan sekali dan kemudian berhenti mendengarkan acara.

Sintak

mouseReleased()

mouseReleased(event)

mouseWheel()

Examples

```
void setup() {
  size(100, 100);
}

void draw() {}

void mouseWheel(MouseEvent event) {
  float e = event.getCount();
  println(e);
}
```

Diskripsi

Fungsi mouseWheel () mengembalikan nilai positif ketika roda mouse diputar ke bawah (ke arah pengguna), dan nilai negatif untuk arah lain (ke atas atau menjauh dari pengguna). Pada OS X dengan pengguliran "alami" diaktifkan, nilainya berlawanan.

Acara mouse dan keyboard hanya berfungsi ketika suatu program memiliki draw (). Tanpa draw (), kode hanya dijalankan sekali dan kemudian berhenti mendengarkan acara.

Sintak

mouseWheel(event)

mouseX

Examples

```
void draw() {
  background(204);
  line(mouseX, 20, mouseX, 80);
}
```

Diskripsi

Variabel sistem mouseX selalu berisi koordinat horizontal mouse saat ini.

Perhatikan bahwa Pemrosesan hanya dapat melacak posisi mouse ketika pointer berada di atas jendela saat ini. Nilai default mouseX adalah 0, jadi 0 akan dikembalikan sampai mouse bergerak di depan jendela sketsa. (Ini biasanya terjadi ketika sketsa dijalankan pertama kali.) Setelah mouse menjauh dari jendela, mouseX akan terus melaporkan posisi terakhirnya.

mouseY

Examples

```
void draw() {
  background(204);
  line(20, mouseY, 80, mouseY);
}
```

Diskripsi

Variabel sistem mouseY selalu berisi koordinat vertikal mouse saat ini.

Perhatikan bahwa Pemrosesan hanya dapat melacak posisi mouse ketika pointer berada di atas jendela saat ini. Nilai default mouseY adalah 0, jadi 0 akan dikembalikan sampai mouse bergerak di depan jendela sketsa. (Ini biasanya terjadi ketika sketsa dijalankan pertama kali.) Setelah mouse menjauh dari jendela, mouseY akan terus melaporkan posisi terakhirnya.

pmouseX

Examples

```
// Gerakkan mouse dengan cepat untuk melihat perbedaannya
// antara posisi saat ini dan sebelumnya
void draw() {
  background(204);
  line(mouseX, 20, pmouseX, 80);
  println(mouseX + " : " + pmouseX);
}
```

Diskripsi

Variabel sistem pmouseX selalu berisi posisi horisontal mouse dalam bingkai sebelum bingkai saat ini.

Anda mungkin menemukan bahwa pmouseX dan pmouseY memiliki nilai yang berbeda ketika direferensikan di dalam draw () dan di dalam acara mouse seperti mousePressed () dan mouseMoved (). Di dalam draw (), pmouseX dan pmouseY memperbarui hanya sekali per frame (sekali per perjalanan melalui loop draw ()). Namun di dalam acara mouse, mereka memperbarui setiap kali acara dipanggil. Jika nilai-nilai ini tidak segera diperbarui selama acara mouse, maka posisi mouse akan dibaca hanya sekali per frame, menghasilkan sedikit keterlambatan dan interaksi berombak. Jika variabel mouse selalu diperbarui beberapa kali per frame, maka sesuatu seperti baris (pmouseX, pmouseY, mouseX, mouseY) di dalam draw ()

akan memiliki banyak celah, karena pmouseX mungkin telah berubah beberapa kali di antara panggilan ke line ().

BAB XVI
Tombol Keyboard

key

Examples

```
// Klik pada jendela untuk fokus,
// dan tekan tombol 'B'.

void draw() {
  if (keyPressed) {
    if (key == 'b' || key == 'B') {
      fill(0);
    }
  } else {
    fill(255);
  }
  rect(25, 25, 50, 50);
}
```

Diskripsi

Kunci variabel sistem selalu berisi nilai kunci terbaru pada keyboard yang digunakan (baik ditekan atau dilepaskan).
Untuk kunci non-ASCII, gunakan variabel keyCode. Kunci yang termasuk dalam spesifikasi ASCII (BACKSPACE, TAB, ENTER, RETURN, ESC, dan DELETE) tidak perlu memeriksa untuk melihat apakah kunci dikodekan, dan Anda cukup menggunakan variabel kunci alih-alih keyCode Jika Anda membuat tanda silang proyek -platform, perhatikan bahwa kunci ENTER biasanya digunakan pada PC dan Unix dan kunci RETURN digunakan sebagai gantinya pada Macintosh. Periksa ENTER dan RETURN untuk memastikan program Anda akan bekerja untuk semua platform.

Ada masalah dengan bagaimana keyCode berperilaku di berbagai penyaji dan sistem operasi. Waspadai perilaku tak terduga saat Anda mengganti perender dan sistem operasi.

keyCode

Examples

```
color fillVal = color(126);

void draw() {
  fill(fillVal);
  rect(25, 25, 50, 50);
}

void keyPressed() {
  if (key == CODED) {
    if (keyCode == UP) {
      fillVal = 255;
    } else if (keyCode == DOWN) {
      fillVal = 0;
    }
  } else {
    fillVal = 126;
  }
}
```

Diskripsi

keyCode variabel digunakan untuk mendeteksi kunci khusus seperti tombol panah (UP, DOWN, LEFT,dan RIGHT) serta ALT, CONTROL, dan SHIFT.

Saat memeriksa kunci-kunci ini, akan berguna untuk memeriksa dulu apakah kunci tersebut dikodekan. Ini dilakukan dengan kondisional if (key == CODED), seperti yang ditunjukkan pada contoh di atas.

Kunci yang termasuk dalam spesifikasi ASCII (BACKSPACE, TAB, ENTER, RETURN, ESC, dan DELETE) tidak perlu memeriksa untuk melihat apakah kunci dikodekan; untuk kunci-kunci itu, Anda cukup menggunakan variabel kunci secara langsung (dan bukan keyCode). Jika Anda membuat proyek lintas platform, perhatikan bahwa kunci ENTER biasanya digunakan pada PC dan Unix, sedangkan kunci RETURN digunakan pada Mac. Pastikan program Anda akan bekerja pada semua platform dengan memeriksa ENTER dan RETURN.

Bagi mereka yang terbiasa dengan Java, nilai-nilai untuk UP dan BAWAH adalah versi yang lebih pendek dari Java's KeyEvent.VK_UP dan KeyEvent.VK_DOWN. Nilai keyCode lainnya dapat ditemukan di referensi Java KeyEvent.

Ada masalah dengan bagaimana keyCode berperilaku di berbagai penyaji dan sistem operasi. Berhati-hatilah terhadap perilaku yang tidak terduga saat Anda mengganti perender dan sistem operasi dan Anda menggunakan kunci tidak disebutkan dalam entri referensi ini.

keyPressed()

Examples

```
// Klik pada gambar untuk fokus,
// lalu tekan tombol apa saja.

int value = 0;
```


```
void draw() {
  fill(value);
  rect(25, 25, 50, 50);
}

void keyPressed() {
  if (value == 0) {
    value = 255;
  } else {
    value = 0;
  }
}
```

Diskripsi

Deskripsi Fungsi keyPressed () dipanggil satu kali setiap kali tombol ditekan. Kunci yang ditekan disimpan dalam variabel kunci.

Untuk kunci non-ASCII, gunakan variabel keyCode. Kunci yang termasuk dalam spesifikasi ASCII (BACKSPACE, TAB, ENTER, RETURN, ESC, dan DELETE) tidak perlu memeriksa untuk melihat apakah kunci dikodekan; untuk kunci-kunci itu, Anda cukup menggunakan variabel kunci secara langsung (dan bukan keyCode). Jika Anda membuat proyek lintas platform, perhatikan bahwa kunci ENTER biasanya digunakan pada PC dan Unix, sedangkan kunci RETURN digunakan pada Mac. Pastikan program Anda akan bekerja pada semua platform dengan memeriksa ENTER dan RETURN.

Karena cara sistem operasi menangani pengulangan tombol, menekan satu tombol dapat menyebabkan beberapa panggilan ke keyPressed (). Tingkat pengulangan ditentukan oleh sistem operasi, dan dapat dikonfigurasi secara berbeda pada setiap komputer.

Perhatikan bahwa ada variabel boolean bernama sama yang disebut keyPressed. Lihat halaman referensi untuk informasi lebih lanjut.
Acara mouse dan keyboard hanya berfungsi ketika suatu program memiliki draw (). Tanpa draw (), kode hanya dijalankan sekali dan kemudian berhenti mendengarkan acara.
Dengan rilis macOS Sierra, Apple mengubah cara kerja pengulangan tombol, jadi keyPressed mungkin tidak berfungsi seperti yang diharapkan. Lihat di sini untuk perincian masalah dan cara memperbaikinya.

Sintak
keyPressed()
keyPressed(event)

keyPressed

Examples

```
//Klik pada gambar untuk fokus,
//lalu tekan tombol apa saja.

//Catatan: Kotak dalam contoh ini mungkin
//berkedip seperti sistem operasi mungkin
//daftarkan tekan tombol panjang sebagai pengulangan
//dari penekanan tombol.
void draw() {
  if (keyPressed == true) {
    fill(0);
  } else {
    fill(255);
```


```
  }
  rect(25, 25, 50, 50);
}
```

Diskripsi

Tombol variabel sistem boolean Ditekan adalah benar jika ada tombol yang ditekan dan salah jika tidak ada tombol yang ditekan.

Perhatikan bahwa ada fungsi yang bernama serupa yang disebut keyPressed (). Lihat halaman referensi untuk informasi lebih lanjut.

keyReleased()

Examples

```
//Klik pada gambar untuk fokus,
//lalu tekan tombol apa saja.
int value = 0;

void draw() {
  fill(value);
  rect(25, 25, 50, 50);
}

void keyReleased() {
  if (value == 0) {
    value = 255;
  } else {
    value = 0;
  }
}
```

Diskripsi

Fungsi keyReleased () dipanggil sekali setiap kali kunci dilepaskan. Kunci yang dirilis akan disimpan dalam variabel kunci. Lihat key dan keyCode untuk informasi lebih lanjut.
Acara mouse dan keyboard hanya berfungsi ketika suatu program memiliki draw (). Tanpa draw (), kode hanya dijalankan sekali dan kemudian berhenti mendengarkan acara.

Sintak

keyReleased()
keyReleased(event)

keyTyped()

Examples

```
// Jalankan program ini untuk mempelajari bagaimana masing-masing fungsi ini
// berhubungan dengan yang lain.
void draw() { } // Empty draw() needed to keep the program running

void keyPressed() {
  println("pressed " + int(key) + " " + keyCode);
}

void keyTyped() {
  println("typed " + int(key) + " " + keyCode);
}

void keyReleased() {
  println("released " + int(key) + " " + keyCode);
```


}

Diskripsi

The keyTyped() function is called once every time a key is pressed, but action keys such as Ctrl, Shift, and Alt are ignored. Because of how operating systems handle key repeats, holding down a key may cause multiple calls to keyTyped(). The rate of repeat is set by the operating system, and may be configured differently on each computer.

Mouse and keyboard events only work when a program has draw(). Without draw(), the code is only run once and then stops listening for events.

Sintak

keyTyped()
keyTyped(event)

BAB XVII
File

BufferedReader

Examples

```
BufferedReader reader;
String line;

void setup() {
  // Buka file dari contoh createWriter ()
  reader = createReader("positions.txt");
}

void draw() {
  try {
    line = reader.readLine();
  } catch (IOException e) {
    e.printStackTrace();
    line = null;
  }
  if (line == null) {
    // Berhenti membaca karena kesalahan atau file kosong
    noLoop();
  } else {
    String[] pieces = split(line, TAB);
    int x = int(pieces[0]);
    int y = int(pieces[1]);
    point(x, y);
  }
}
```

Diskripsi

Objek BufferedReader digunakan untuk membaca file baris demi baris sebagai objek String individual.

Dimulai dengan Memproses rilis 0134, semua file dimuat dan

disimpan oleh API Pemrosesan menggunakan pengkodean UTF-8. Dalam rilis sebelumnya, penyandian default untuk platform Anda digunakan, yang menyebabkan masalah ketika file dipindahkan ke platform lain..

createInput()

Examples

```
// Muat file lokal 'data.txt' dan inisialisasi InputStream baru
InputStream input = createInput("data.txt");

String content = "";

try {
  int data = input.read();
  while (data != -1) {
    content += data;
    data = input.read();
  }
}
catch (IOException e) {
  e.printStackTrace();
}
finally {
  try {
    input.close();
  }
  catch (IOException e) {
    e.printStackTrace();
  }
}
```

println(content);

Diskripsi

Ini adalah fungsi singkatan untuk pemrogram tingkat lanjut untuk menginisialisasi dan membuka Java InputStream. Ini berguna jika Anda ingin menggunakan fasilitas yang disediakan oleh PApplet untuk dengan mudah membuka file dari folder data atau dari URL, tetapi Anda memerlukan objek InputStream sehingga Anda dapat menggunakan bagian lain dari Jawa untuk mengambil kendali lebih besar tentang bagaimana aliran dibaca .

Nama file yang diteruskan dapat berupa:

- URL, seperti pada: createInput ("http://processing.org/")
- Nama file dalam folder data sketsa
- Jalur lengkap ke file yang akan dibuka secara lokal (saat dijalankan sebagai aplikasi)

Jika item yang diminta tidak ada, null dikembalikan. Jika tidak online, ini juga akan memeriksa untuk melihat apakah pengguna meminta file yang namanya tidak ditulis dengan huruf besar. Jika kapitalisasi berbeda, kesalahan akan dicetak ke konsol. Ini membantu mencegah masalah yang muncul ketika sketsa diekspor ke web, di mana sensitivitas case penting, sebagai lawan berjalan dari dalam Lingkungan Pengembangan Pemrosesan pada Windows atau Mac OS, di mana sensitivitas case dipertahankan tetapi diabaikan.

Sintak

createInput(filename)

createReader()

Examples

```
void setup() {
  size(100, 100);
  parseFile();
}

void parseFile() {
  // Buka file dari contoh createWriter ()
  BufferedReader reader = createReader("positions.txt");
  String line = null;
  try {
    while ((line = reader.readLine()) != null) {
      String[] pieces = split(line, TAB);
      int x = int(pieces[0]);
      int y = int(pieces[1]);
      point(x, y);
    }
    reader.close();
  } catch (IOException e) {
    e.printStackTrace();
  }
}
```

Diskripsi

Membuat objek BufferedReader yang dapat digunakan untuk membaca file baris demi baris sebagai objek String individual. Ini adalah pelengkap fungsi createWriter (). Untuk informasi lebih lanjut tentang kelas BufferedReader dan metode-metodenya seperti readLine () dan tutup yang digunakan dalam contoh di atas, silakan baca referensi Java.

Dimulai dengan Memproses rilis 0134, semua file dimuat dan disimpan oleh API Pemrosesan menggunakan pengkodean UTF-8. Dalam rilis sebelumnya, penyandian default untuk platform Anda digunakan, yang menyebabkan masalah ketika file dipindahkan ke platform lain.

Sintak
createReader(filename)

launch()

Examples

```
void setup() {
  size(200, 200);
}

void draw() {
  // draw () harus ada agar mousePressed () berfungsi
}

void mousePressed() {
  println("Opening Process_4");
  launch("/Applications/Process_4.app");
}
```

Diskripsi
Mencoba membuka aplikasi atau file menggunakan peluncur platform Anda. Parameter nama file adalah String yang menentukan nama file dan lokasi. Parameter lokasi harus berupa nama jalur lengkap, atau nama yang dapat dieksekusi di PATH sistem. Dalam kebanyakan kasus, menggunakan path

lengkap adalah pilihan terbaik, daripada mengandalkan sistem PATH. Pastikan untuk membuat file tersebut dapat dieksekusi sebelum mencoba membukanya (chmod + x).
Fungsi ini (kira-kira) mengemulasi apa yang terjadi ketika Anda mengklik dua kali aplikasi atau dokumen di MacOS Finder, Windows Explorer, atau manajer file Linux favorit Anda. Jika Anda mencoba menjalankan fungsi baris perintah secara langsung, gunakan fungsi exec () sebagai gantinya (lihat di bawah).
Fungsi ini berperilaku berbeda pada setiap platform. Pada Windows, parameter dikirim ke shell Windows melalui "cmd / c". Pada Mac OS X, perintah "buka" digunakan (ketik "man open" di Terminal.app untuk dokumentasi). Di Linux, ia pertama-tama mencoba gnome-open, kemudian kde-open, tetapi jika tidak ada, ia mengirimkan perintah ke shell dan berdoa bahwa sesuatu yang bermanfaat terjadi.
Untuk pengguna yang terbiasa dengan Java, ini tidak sama dengan Runtime.exec (), karena perintah launcher didahulukan. Sebagai gantinya, fungsi exec (String []) adalah jalan pintas untuk Runtime.getRuntime.exec (String []). Fungsi exec () didokumentasikan dalam JavaDoc di kelas PApplet.

Sintak
launch(args)

Parameter-parameter
args String[]:argumen ke peluncur, mis. nama file

loadBytes()

Examples
// Buka file dan baca data binernyadata

```
byte b[] = loadBytes("something.dat");

// Cetak setiap nilai, dari 0 hingga 255for (int i = 0; i < b.length; i++) {
  // Setiap angka kesepuluh, mulailah baris baru
  if ((i % 10) == 0) {
    println();
  }
  // byte dari -128 hingga 127, ini dikonversi menjadi 0 hingga 255
  int a = b[i] & 0xff;
  print(a + " ");
}
// Cetak baris kosong di bagian akhir
println();
```

Diskripsi

Membaca konten file dan menempatkannya dalam array byte. Jika nama file digunakan sebagai parameter, seperti dalam contoh di atas, file tersebut harus dimuat dalam direktori / folder "data" sketsa.

Atau, file mungkin dimuat dari mana saja di komputer lokal menggunakan jalur absolut (sesuatu yang dimulai dengan / pada Unix dan Linux, atau huruf drive pada Windows), atau parameter nama file dapat menjadi URL untuk file yang ditemukan pada file jaringan.

Jika file tidak tersedia atau kesalahan terjadi, null akan dikembalikan dan pesan kesalahan akan dicetak ke konsol. Pesan kesalahan tidak menghentikan program, namun nilai nol dapat menyebabkan NullPointerException jika kode Anda tidak memeriksa apakah nilai yang dikembalikan adalah nol.

Sintak

loadBytes(filename)

filename String: nama file di folder data atau URL.

loadJSONArray()

Examples

```
// File JSON pendek berikut yang disebut "data.json" diuraikan
// dalam kode di bawah ini. Itu harus ada di folder "data" proyek.
//
// [
//   {
//     "id": 0,
//     "spesies": "Capra hircus",
//     "nama": "Kambing"
//   },
//   {
//     "id": 1,
//     "spesies": "Panthera pardus",
//     "nama": "Leopard"
//   },
//   {
//     "id": 2,
//     "spesies": "Equus zebra",
//     "nama": "Zebra"
//   }
// ]

JSONArray values;

void setup() {
```

```
  values = loadJSONArray("data.json");

  for (int i = 0; i < values.size(); i++) {

    JSONObject animal = values.getJSONObject(i);

    int id = animal.getInt("id");
    String species = animal.getString("species");
    String name = animal.getString("name");

    println(id + ", " + species + ", " + name);
  }
}

// Cetak sketsa:
// 0, Capra hircus, Goat
// 1, Panthera pardus, Leopard
// 2, Equus zebra, Zebra
```

Diskripsi

Memuat array objek JSON dari folder data atau URL, dan mengembalikan JSONArray. Per sintaks JSON standar, array harus dilampirkan dalam sepasang tanda kurung keras [], dan setiap objek dalam array harus dipisahkan oleh koma.

Semua file dimuat dan disimpan oleh API Pemrosesan menggunakan pengkodean UTF-8.

Sintak

loadJSONArray(filename)
loadJSONArray(file)

Parameter-parameter

Filename String: nama file di folder data atau URL

loadJSONObject()

Examples

```
// File JSON pendek berikut yang disebut "data.json" diuraikan
// dalam kode di bawah ini. Itu harus ada di folder "data"
proyek.
//
// {
//   "id": 0,
//   "spesies": "Panthera leo",
//   "nama": "Singa"
// }

JSONObject json;

void setup() {

  json = loadJSONObject("data.json");

  int id = json.getInt("id");
  String species = json.getString("species");
  String name = json.getString("name");

  println(id + ", " + species + ", " + name);
}

// Cetak sketsa:
// 0, Panthera leo, Lion
```

Diskripsi

Memuat JSON dari folder data atau URL, dan mengembalikan JSONObject.

Semua file dimuat dan disimpan oleh API Pemrosesan menggunakan pengkodean UTF-8.

Sintak
loadJSONObject(filename)

Parameter-parameter
Filename String: nama file di folder data atau URL

loadStrings()

Examples

```
String[] lines = loadStrings("list.txt");
println("there are " + lines.length + " lines");
for (int i = 0 ; i < lines.length; i++) {
  println(lines[i]);
}

String[] lines =
loadStrings("http://processing.org/about/index.html");
println("there are " + lines.length + " lines");
for (int i = 0 ; i < lines.length; i++) {
  println(lines[i]);
}
```

Diskripsi
Membaca isi file dan membuat array String dari masing-masing baris. Jika nama file digunakan sebagai parameter, seperti dalam contoh di atas, file tersebut harus dimuat dalam direktori / folder "data" sketsa.

Atau, file mungkin dimuat dari mana saja di komputer lokal menggunakan jalur absolut (sesuatu yang dimulai dengan / pada Unix dan Linux, atau huruf drive pada Windows), atau parameter nama file dapat menjadi URL untuk file yang ditemukan pada file jaringan.
Jika file tidak tersedia atau kesalahan terjadi, null akan dikembalikan dan pesan kesalahan akan dicetak ke konsol. Pesan kesalahan tidak menghentikan program, namun nilai nol dapat menyebabkan NullPointerException jika kode Anda tidak memeriksa apakah nilai yang dikembalikan adalah nol.
Dimulai dengan Memproses rilis 0134, semua file dimuat dan disimpan oleh API Pemrosesan menggunakan pengkodean UTF-8. Dalam rilis sebelumnya, penyandian default untuk platform Anda digunakan, yang menyebabkan masalah ketika file dipindahkan ke platform lain.

Sintak
loadStrings(filename)
loadStrings(reader)

Parameter-parameter
Filename String: nama file atau url yang akan dimuat

loadTable()

Examples
// File CSV pendek berikut yang disebut "mammals.csv" diuraikan
// dalam kode di bawah ini. Itu harus ada di folder "data" proyek.
//
// id, spesies, nama

```
// 0, Capra hircus, Goat
// 1, Panthera pardus, Leopard
// 2, Equus zebra, Zebra

Table table;

void setup() {

  table = loadTable("mammals.csv", "header");

  println(table.getRowCount() + " total rows in table");

  for (TableRow row : table.rows()) {

    int id = row.getInt("id");
    String species = row.getString("species");
    String name = row.getString("name");

    println(name + " (" + species + ") has an ID of " + id);
  }

}

// Cetak sketsa:
// 3 total baris dalam table
// Kambing (Capra hircus) memiliki ID 0
// Leopard (Panthera pardus) memiliki ID 1
// Zebra (Equus zebra) memiliki ID 2
```

Diskripsi

Membaca konten file atau URL dan membuat objek Tabel dengan nilainya. Jika file ditentukan, itu harus terletak di folder "data" sketsa. Parameter nama file juga bisa menjadi URL ke file yang ditemukan online. Nama file harus diakhiri dengan

ekstensi atau ekstensi harus ditentukan dalam parameter opsi. Misalnya, untuk menggunakan data yang dipisahkan dengan tab, sertakan "tsv" dalam parameter opsi jika nama file atau URL tidak berakhir dengan .tsv. Catatan: Jika ekstensi ada di kedua tempat, ekstensi di opsi digunakan.
Jika file berisi baris tajuk, masukkan "tajuk" di parameter opsi. Jika file tidak memiliki baris tajuk, maka cukup abaikan opsi "tajuk".
Saat menentukan header dan tipe file sebagai parameter opsi, pisahkan opsi dengan koma, seperti pada: loadTable ("data.csv", "header, tsv")
Semua file dimuat dan disimpan oleh API Pemrosesan menggunakan pengkodean UTF-8.

Sintak
loadTable(filename)
loadTable(filename, options)

Parameter-parameter

filename	String: nama file di folder data atau URL.
options	String: dapat berisi "header", "tsv", "csv", atau "bin" dipisahkan oleh koma

loadXML()

Examples
// File XML pendek berikut yang disebut "mammals.xml" diuraikan
// dalam kode di bawah ini. Itu harus ada di folder "data" proyek.
//
// <? xml version = "1.0"?>

```
//                                                    <mamalia>
// <animal id = "0" species = "Capra hircus"> Kambing </animal>
// <animal id = "1" species = "Panthera pardus"> Leopard </animal>
// <animal id = "2" species = "Equus zebra"> Zebra </animal>
// </mammals>

XML xml;

void setup() {
  xml = loadXML("mammals.xml");
  XML[] children = xml.getChildren("animal");

  for (int i = 0; i < children.length; i++) {
    int id = children[i].getInt("id");
    String coloring = children[i].getString("species");
    String name = children[i].getContent();
    println(id + ", " + coloring + ", " + name);
  }
}

// Cetak sketsa:
// 0, Capra hircus, Goat
// 1, Panthera pardus, Leopard
// 2, Equus zebra, Zebra
```

Diskripsi

Membaca konten file atau URL dan membuat objek XML dengan nilainya. Jika file ditentukan, itu harus terletak di folder "data" sketsa. Parameter nama file juga bisa menjadi URL ke file yang ditemukan online.

File

Semua file dimuat dan disimpan oleh API Pemrosesan menggunakan pengkodean UTF-8. Jika Anda perlu memuat file XML yang tidak dalam format UTF-8, lihat referensi pengembang untuk objek XML.

Sintak
loadXML(filename)

Parameter-parameter
filename String: nama file di folder data atau URL.

parseJSONArray()

Examples

```
String data = "[ \"Capra hircus\", \"Panthera pardus\", \"Equus zebra\" ]";

void setup() {
  JSONArray json = parseJSONArray(data);
  if (json == null) {
    println("JSONArray could not be parsed");
  } else {
    String species = json.getString(1);
    println(species);
  }
}

// Cetak sketsa:
// Panthera pardus
```

Diskripsi
Mengambil String, mem-parsing isinya, dan mengembalikan JSONArray. Jika String tidak berisi data JSONArray atau tidak dapat diuraikan, nilai nol dikembalikan.

parseJSONArray () paling berguna saat menarik data secara dinamis, seperti dari API pihak ketiga. Biasanya, hasil API akan disimpan ke String, dan kemudian dapat dikonversi ke JSONArray terstruktur menggunakan parseJSONArray (). Pastikan untuk memeriksa apakah null dikembalikan sebelum melakukan operasi pada JSONArray baru jika konten String tidak dapat diuraikan.

Jika data Anda sudah ada sebagai file JSON di folder data, lebih mudah menggunakan loadJSONArray ().

Sintak
parseJSONArray(input)

Parameter-parameter
input String: String untuk diurai sebagai JSONArray

parseJSONObject()

Examples
String data = "{ \"id\": 0, \"species\": \"Panthera leo\", \"name\": \"Lion\"}";

```
void setup() {
  JSONObject json = parseJSONObject(data);
  if (json == null) {
    println("JSONObject could not be parsed");
```


File

```
  } else {
    String species = json.getString("species");
    println(species);
  }
}

// Cetak sketsa:
// Panthera leo
```

Diskripsi
Mengambil String, mem-parsing isinya, dan mengembalikan JSONObject. Jika String tidak berisi data JSONObject atau tidak dapat diuraikan, nilai null dikembalikan.

parseJSONObject () paling berguna saat menarik data secara dinamis, seperti dari API pihak ketiga. Biasanya, hasil API akan disimpan ke sebuah String, dan kemudian dapat dikonversi ke JSONObject terstruktur menggunakan parseJSONObject (). Pastikan untuk memeriksa apakah null dikembalikan sebelum melakukan operasi pada JSONObject baru jika konten String tidak dapat diuraikan.
Jika data Anda sudah ada sebagai file JSON di folder data, lebih mudah menggunakan loadJSONObject ().

Sintak
parseJSONObject(input)

Parameter-parameter
Input String: String untuk diurai sebagai objek JSONOb

parseXML()

Examples

```
String data = "<mammals><animal>Goat</animal></mammals>";

void setup() {
  XML xml = parseXML(data);
  if (xml == null) {
    println("XML could not be parsed.");
  } else {
    XML firstChild = xml.getChild("animal");
    println(firstChild.getContent());
  }
}

// Cetak Sketsa:
// Kambing
```

Diskripsi

Mengambil String, mem-parsing isinya, dan mengembalikan objek XML. Jika String tidak berisi data XML atau tidak dapat diuraikan, nilai nol dikembalikan.

parseXML () paling berguna saat menarik data secara dinamis, seperti dari API pihak ketiga. Biasanya, hasil API akan disimpan ke String, dan kemudian dapat dikonversi ke objek XML terstruktur menggunakan parseXML (). Pastikan untuk memeriksa apakah null dikembalikan sebelum melakukan operasi pada objek XML baru, jika konten String tidak dapat diuraikan.

Jika data Anda sudah ada sebagai file XML di folder data, lebih mudah menggunakan loadXML ().

Sintak
parseXML(xmlString)
parseXML(xmlString, options)

Parameter-parameter
xmlString String: konten yang akan diuraikan sebagai XML

selectFolder()

Examples
```
void setup() {
  selectFolder("Select a folder to process:", "folderSelected");
}
void folderSelected(File selection) {
  if (selection == null) {
    println("Window was closed or the user hit cancel.");
  } else {
    println("User selected " + selection.getAbsolutePath());
  }
}
```

Diskripsi
Membuka dialog pemilih file platform khusus untuk memilih folder. Setelah pemilihan dilakukan, pemilihan akan diteruskan ke fungsi 'panggilan balik'. Jika dialog ditutup atau dibatalkan, null akan dikirim ke fungsi, sehingga program tidak menunggu input tambahan. Callback diperlukan karena cara kerja threading.

Sintak

selectFolder(prompt, callback)
selectFolder(prompt, callback, file)
selectFolder(prompt, callback, file, callbackObject)
selectFolder(prompt, callbackMethod, defaultSelection, callbackObject, parentFrame)
selectFolder(prompt, callbackMethod, defaultSelection, callbackObject, parentFrame, sketch)

Parameter-parameter

prompt	String: pesan ke pengguna
callback	String: nama metode yang akan dipanggil saat pemilihan dilakukan

selectInput()

Examples

```
void setup() {
  selectInput("Select a file to process:", "fileSelected");
}

void fileSelected(File selection) {
  if (selection == null) {
    println("Window was closed or the user hit cancel.");
  } else {
    println("User selected " + selection.getAbsolutePath());
  }
}
```


Diskripsi

Membuka dialog pemilih file platform khusus untuk memilih file untuk input. Setelah pemilihan dilakukan, File yang dipilih akan diteruskan ke fungsi 'panggilan balik'. Jika dialog ditutup atau dibatalkan, null akan dikirim ke fungsi, sehingga program tidak menunggu input tambahan. Callback diperlukan karena cara kerja threading.

Sintak

selectInput(prompt, callback)
selectInput(prompt, callback, file)
selectInput(prompt, callback, file, callbackObject)
selectInput(prompt, callbackMethod, file, callbackObject, parent, sketch)
selectInput(prompt, callbackMethod, file, callbackObject, parent)

Parameter-parameter

prompt String: pesan ke pengguna
callback String: nama metode yang akan dipanggil saat pemilihan dilakukan

BAB XVIII
WAKTU DAN TANGGAL

day()

Examples

```
int d = day();    // Nilai dari 1 - 31
int m = month();  // Nilai dari 1 - 12
int y = year();   // 2003, 2004, 2005, dll.

String s = String.valueOf(d);
text(s, 10, 28);
s = String.valueOf(m);
text(s, 10, 56);
s = String.valueOf(y);
text(s, 10, 84);
```

Diskripsi

Pemrosesan berkomunikasi dengan jam di komputer Anda. Fungsi day () mengembalikan hari ini sebagai nilai dari 1 – 31

Sintak

day()

hour()

Examples

```
void draw() {
  background(204);
  int s = second();  // Nilai dari 0 - 59
  int m = minute();  // Nilai dari 0 - 59
  int h = hour();    // Nilai dari 0 - 23
  line(s, 0, s, 33);
  line(m, 33, m, 66);
```

```
  line(h, 66, h, 100);
}
```

Diskripsi
Pemrosesan berkomunikasi dengan jam di komputer Anda. Fungsi hour () mengembalikan jam saat ini sebagai nilai dari 0 - 23.

Sintak
hour()

millis()

Examples

```
void draw() {
  int m = millis();
  noStroke();
  fill(m % 255);
  rect(25, 25, 50, 50);
}
```

Diskripsi
Mengembalikan jumlah milidetik (seperseribu detik) sejak memulai program. Informasi ini sering digunakan untuk peristiwa waktu dan urutan animasi.

Sintak
millis()

minute()

Examples

```
void draw() {
  background(204);
  int s = second();  // Nilai dari 0 - 59
  int m = minute();  // Nilai dari 0 - 59
  int h = hour();    // Nilai dari 0 - 23
  line(s, 0, s, 33);
  line(m, 33, m, 66);
  line(h, 66, h, 100);
}
```

Diskripsi

Pemrosesan berkomunikasi dengan jam di komputer Anda. Fungsi minute () mengembalikan menit saat ini sebagai nilai dari 0 - 59.

Sintak

minute()

month()

Examples

```
int d = day();    // Nilai dari 1 - 31
int m = month();  // Nilai dari 1 - 12
int y = year();   // 2003, 2004, 2005, dll.

String s = String.valueOf(d);
text(s, 10, 28);
```


```
s = String.valueOf(m);
text(s, 10, 56);
s = String.valueOf(y);
text(s, 10, 84);
```

Diskripsi
Pemrosesan berkomunikasi dengan jam di komputer Anda. Fungsi month () mengembalikan bulan saat ini sebagai nilai dari 1 - 12.

Sintak
month()

second()

Examples

```
void draw() {
  background(204);
  int s = second();  // Nilai dari 0 - 59
  int m = minute();  // Nilai dari from 0 - 59
  int h = hour();    // Nilai dari 0 - 23
  line(s, 0, s, 33);
  line(m, 33, m, 66);
  line(h, 66, h, 100);
}
```

Diskripsi
Pemrosesan berkomunikasi dengan jam di komputer Anda. Fungsi kedua () mengembalikan detik saat ini sebagai nilai dari 0 - 59.

Sintak

second()

year()

Examples

```
int d = day();    // Nilai dari 1 - 31
int m = month();  // Nilai dari 1 - 12
int y = year();   // 2003, 2004, 2005, dll.

String s = String.valueOf(d);
text(s, 10, 28);
s = String.valueOf(m);
text(s, 10, 56);
s = String.valueOf(y);
text(s, 10, 84);
```

Diskripsi

Pemrosesan berkomunikasi dengan jam di komputer Anda. Fungsi year () mengembalikan tahun berjalan sebagai integer (2003, 2004, 2005, dll).

Sintak

year()

BAB XIX
AREA OUTPUT TEXT

print()

Examples

```
String s = "The size is ";
int w = 1920;
int h = 1080;
print(s);
print(w, "x", h);

// Program ini menulis ke konsol:
// Ukurannya 1920 x 1080

print("begin- ");
float f = 0.3;
int i = 1024;
print("f is " + f + " and i is " + 1024);
String s = " -end";
println(s);
// Program ini menulis ke konsol:
// "begin- f adalah 0.3 dan i adalah 1024 -end"
```

Diskripsi

Fungsi print () menulis ke area konsol, persegi panjang hitam di bagian bawah lingkungan Pemrosesan. Fungsi ini sering membantu untuk melihat data yang dihasilkan suatu program. Println fungsi pendamping berfungsi seperti print (), tetapi membuat baris teks baru untuk setiap panggilan ke fungsi. Lebih dari satu parameter dapat diteruskan ke fungsi dengan memisahkannya dengan koma. Atau, elemen individual dapat dipisahkan dengan tanda kutip ("") dan bergabung dengan operator tambahan (+).

Menggunakan print () pada objek akan menghasilkan null, lokasi memori yang mungkin terlihat seperti "@ 10be08," atau

hasil dari metode toString () dari objek yang sedang dicetak. Pengguna mahir yang menginginkan hasil yang lebih berguna saat memanggil print () di kelas mereka sendiri dapat menambahkan metode toString () ke kelas yang mengembalikan sebuah String.

Perhatikan bahwa konsol ini relatif lambat. Ini berfungsi baik untuk pesan sesekali, tetapi tidak mendukung kecepatan tinggi, output real-time (seperti pada 60 frame per detik). Juga harus dicatat, bahwa print () dalam for loop kadang-kadang dapat mengunci program, dan menyebabkan sketsa membeku.

Sintak
print(what)
print(variables)

Parameter-parameter

what	String, float, char, boolean, atau byte: data untuk dicetak ke konsol
variables	Object []: daftar data, dipisahkan dengan koma

printArray()

Examples
```
float[] f = { 0.3, 0.4, 0.5 };
printArray(f);

// Kode di atas mencetak:
// [0] 0,3
// [1] 0,4
// [2] 0,5
```

Diskripsi

Fungsi printArray () menulis data larik ke area teks pada konsol Lingkungan pemrosesan. Baris baru diletakkan di antara setiap elemen array. Fungsi ini hanya dapat mencetak array satu dimensi.

Perhatikan bahwa konsol ini relatif lambat. Ini berfungsi baik untuk pesan sesekali, tetapi tidak mendukung kecepatan tinggi, output real-time (seperti pada 60 frame per detik).

Sintak

printArray(what)

Parameter-parameter

what Object: array satu dimensi

println()

Examples

```
String s = "The size is ";
int w = 1920;
int h = 1080;
println(s);
println(w, "x", h);

// Program ini menulis ke konsol:
// Ukurannya
// 1920 x 1080

print("begin- ");
float f = 0.3;
int i = 1024;
print("f is " + f + " and i is " + 1024);
```

```
String s = " -end";
println(s);
// Program ini menulis ke konsol:
// "begin- f adalah 0.3 dan i adalah 1024 -end"
```

Diskripsi

Fungsi println () menulis ke area konsol, persegi panjang hitam di bagian bawah lingkungan Pemrosesan. Fungsi ini sering membantu untuk melihat data yang dihasilkan suatu program. Setiap panggilan ke fungsi ini menciptakan jalur output baru. Lebih dari satu parameter dapat diteruskan ke fungsi dengan memisahkannya dengan koma. Atau, elemen individual dapat dipisahkan dengan tanda kutip ("") dan bergabung dengan operator tambahan (+).

Sebelum Memproses 2.1, println () digunakan untuk menulis data array ke konsol. Sekarang, gunakan printArray () untuk menulis data array ke konsol.

Perhatikan bahwa konsol ini relatif lambat. Ini berfungsi baik untuk pesan sesekali, tetapi tidak mendukung kecepatan tinggi, output real-time (seperti pada 60 frame per detik). Juga harus dicatat, bahwa println () di dalam for for terkadang dapat mengunci program, dan menyebabkan sketsa membeku.

Sintak

```
println()
println(what)
println(variables)
```

Parameter-parameter

what	Object, String, float, char, boolean, atau byte: data untuk dicetak ke konsol
variables	Object []: daftar data, dipisahkan dengan koma

save()

Examples

```
line(20, 20, 80, 80);
// Menyimpan file TIFF bernama "diagonal.tif""
save("diagonal.tif");
// Menyimpan file TARGA bernama "cross.tga"
line(80, 20, 20, 80);
save("cross.tga");
```

Diskripsi

Menyimpan gambar dari jendela tampilan. Tambahkan ekstensi file ke nama file, untuk menunjukkan format file yang akan digunakan: baik TIFF (.tif), TARGA (.tga), JPEG (.jpg), atau PNG (.png). Jika tidak ada ekstensi yang disertakan dalam nama file, gambar akan disimpan dalam format TIFF dan .tif akan ditambahkan ke nama. File-file ini disimpan ke folder sketsa, yang dapat dibuka dengan memilih "Tampilkan folder sketsa" dari menu "Sketsa". Atau, file dapat disimpan ke lokasi mana pun di komputer dengan menggunakan jalur absolut (sesuatu yang dimulai dengan / pada Unix dan Linux, atau huruf drive pada Windows).

Semua gambar yang disimpan dari jendela gambar utama akan menjadi buram. Untuk menyimpan gambar tanpa latar belakang, gunakan createGraphics ().

Sintak

save(filename)

Parameter-parameter

filename String: urutan huruf dan angka

saveFrame()

Examples

```
int x = 0;
void draw() {
  background(204);
  if (x < 100) {
    line(x, 0, x, 100);
    x = x + 1;
  } else {
    noLoop();
  }
  // Menyimpan setiap frame sebagai layar-0001.tif, layar-0002.tif, dll.
  saveFrame();
}
```

```
int x = 0;
void draw() {
  background(204);
  if (x < 100) {
    line(x, 0, x, 100);
    x = x + 1;
  } else {
    noLoop();
  }
    // Simpan setiap frame sebagai line-000001.png, line-000002.png, dll.
  saveFrame("line-######.png");
}
```

Diskripsi

Menyimpan urutan gambar bernomor, satu gambar setiap kali fungsi dijalankan. Untuk menyimpan gambar yang identik dengan jendela tampilan, jalankan fungsi di akhir draw () atau di dalam mouse dan acara utama seperti mousePressed () dan keyPressed (). Gunakan program Movie Maker di menu Tools untuk menggabungkan gambar-gambar ini ke film.

Jika saveFrame () digunakan tanpa parameter, itu akan menyimpan file sebagai screen-0000.tif, screen-0001.tif, dan sebagainya. Anda dapat menentukan nama urutan dengan parameter nama file, termasuk tanda pagar (####), yang akan diganti dengan nilai frameCount saat ini. (Jumlah tanda pagar digunakan untuk menentukan berapa banyak digit untuk disertakan dalam nama file.) Menambahkan ekstensi file, untuk menunjukkan format file yang akan digunakan: TIFF (.tif), TARGA (.tga), JPEG (.jpg), atau PNG (.png). File gambar disimpan ke folder sketsa, yang dapat dibuka dengan memilih "Tampilkan Folder Sketsa" dari menu "Sketsa".

Atau, file dapat disimpan ke lokasi mana pun di komputer dengan menggunakan jalur absolut (sesuatu yang dimulai dengan / pada Unix dan Linux, atau huruf drive pada Windows).

Semua gambar yang disimpan dari jendela gambar utama akan menjadi buram. Untuk menyimpan gambar tanpa latar belakang, gunakan createGraphics ().

Sintak

saveFrame()
saveFrame(filename)

Parameter-parameter

filename	String: setiap urutan huruf atau angka yang diakhiri dengan ".tif", ".tga", ".jpg", atau ".png"

beginRaw()

Examples

```
import processing.pdf.*;

void setup() {
  size(400, 400);
  beginRaw(PDF, "raw.pdf");
}

void draw() {
  line(pmouseX, pmouseY, mouseX, mouseY);
}

void keyPressed() {
  if (key == ' ') {
    endRaw();
    exit();
  }
}
```

Diskripsi

Untuk membuat vektor dari data 3D, gunakan perintah beginRaw () dan endRaw (). Perintah ini akan mengambil data bentuk sesaat sebelum ditampilkan ke layar. Pada tahap ini, seluruh adegan Anda hanyalah daftar panjang garis dan segitiga individu. Ini berarti bahwa bentuk yang dibuat dengan fungsi sphere () akan terdiri dari ratusan segitiga, bukan satu objek. Atau bahwa bentuk garis multi-segmen (seperti kurva) akan diberikan sebagai segmen individual.

Saat menggunakan beginRaw () dan endRaw (), dimungkinkan untuk menulis ke renderer 2D atau 3D. Sebagai contoh,

beginRaw () dengan pustaka PDF akan menulis geometri sebagai segitiga dan garis pipih, bahkan jika merekam dari renderer P3D.
Jika Anda ingin latar belakang muncul di file Anda, gunakan rect (0, 0, lebar, tinggi) setelah mengatur fill () ke warna latar belakang. Kalau tidak, latar belakang tidak akan diberikan ke file karena latar belakang tidak berbentuk.

Menggunakan petunjuk (ENABLE_DEPTH_SORT) dapat meningkatkan tampilan geometri 3D yang ditarik ke format file 2D.
Lihat contoh dalam referensi untuk pustaka PDF dan DXF untuk informasi lebih lanjut.

Sintak

beginRaw(renderer, filename)

Parameter-parameter

renderer	String: misalnya, PDF atau DXF
filename	String: nama file untuk keluaran

beginRecord()

Examples

```
import processing.pdf.*;

void setup() {
  size(400, 400);
  beginRecord(PDF, "everything.pdf");
}

void draw() {
  ellipse(mouseX, mouseY, 10, 10);
```

```
}

void mousePressed() {
  endRecord();
  exit();
}
```

Diskripsi

Membuka file baru dan semua fungsi menggambar selanjutnya digemakan ke file ini serta jendela tampilan. Fungsi beginRecord () membutuhkan dua parameter, yang pertama adalah renderer dan yang kedua adalah nama file. Fungsi ini selalu digunakan dengan endRecord () untuk menghentikan proses perekaman dan menutup file.

Perhatikan bahwa beginRecord () hanya akan mengambil pengaturan apa pun yang terjadi setelah dipanggil. Misalnya, jika Anda memanggil textFont () sebelum beginRecord (), font itu tidak akan ditetapkan untuk file yang Anda rekam.

beginRecord () hanya berfungsi dengan renderer PDF dan SVG.

Sintak

beginRecord(renderer, filename)

Parameter-parameter

renderer	String: PDF atau SVG
filename	String: nama file untuk keluaran

createOutput()

Diskripsi

Mirip dengan createInput (), ini membuat Java OutputStream untuk nama file atau jalur yang diberikan. File akan dibuat di folder sketsa, atau di folder yang sama dengan aplikasi yang diekspor.

Jika jalur tidak ada, folder perantara akan dibuat. Jika pengecualian terjadi, itu akan dicetak ke konsol, dan null akan dikembalikan.

Fungsi ini adalah kenyamanan atas pendekatan Java yang mengharuskan Anda untuk 1) membuat objek FileOutputStream, 2) menentukan lokasi file yang tepat, dan 3) menangani pengecualian. Pengecualian ditangani secara internal oleh fungsi, yang lebih sesuai untuk proyek "sketsa".

Jika nama file keluaran berakhir dengan .gz, output akan secara otomatis dikompres GZIP seperti yang tertulis.

Sintak

createOutput(filename)

Parameter-parameter

filename String: nama file yang akan dibuka

createWriter()

Examples

```
PrintWriter output;

void setup() {
  // Buat file baru di direktori sketsa
```

```
  output = createWriter("positions.txt");
}

void draw() {
  point(mouseX, mouseY);
  output.println(mouseX + "t" + mouseY); // Write the
coordinate to the file
}

void keyPressed() {
  output.flush(); // Menulis data yang tersisa ke file
  output.close(); // Menyelesaikan file
  exit(); // Menghentikan program
}
}
```

Diskripsi

Membuat file baru di folder sketsa, dan objek PrintWriter untuk menulisnya. Agar file dibuat dengan benar, ia harus dibilas dan harus ditutup dengan metode flush () dan close () (lihat contoh di atas).

Dimulai dengan Memproses rilis 0134, semua file dimuat dan disimpan oleh API Pemrosesan menggunakan pengkodean UTF-8. Dalam rilis sebelumnya, penyandian default untuk platform Anda digunakan, yang menyebabkan masalah ketika file dipindahkan ke platform lain.

Sintak

createWriter(filename)

Parameter-parameter

filename String: nama file yang akan dibuat

endRaw()

Examples

```
import processing.pdf.*;

void setup() {
  size(400, 400);
  beginRaw(PDF, "raw.pdf");
}

void draw() {
  line(pmouseX, pmouseY, mouseX, mouseY);
}

void keyPressed() {
  if (key == ' ') {
    endRaw();
    exit();
  }
}
```

Diskripsi

Untuk melengkapi beginRaw (); mereka harus selalu digunakan bersama. Lihat referensi beginRaw () untuk detailnya.

Sintak

endRaw()

endRecord()

Examples

```
import processing.pdf.*;
```


```
void setup() {
  size(400, 400);
  beginRecord(PDF, "everything.pdf");
}

void draw() {
  ellipse(mouseX, mouseY, 10, 10);
}

void mousePressed() {
  endRecord();
  exit();
}
```

Diskripsi
Menghentikan proses perekaman dimulai dengan beginRecord () dan menutup file.

Sintak
endRecord()

PrintWriter

Examples

```
PrintWriter output;

void setup() {
  // Buat file baru di direktori sketsa
  output = createWriter("positions.txt");
}
```

```
void draw() {
  point(mouseX, mouseY);
  output.println(mouseX);  // Tuliskan koordinat ke file
}

void keyPressed() {
  output.flush();  // Menulis data yang tersisa ke file
  output.close();  // Menyelesaikan file
  exit();  // Menghentikan program
}
```

Diskripsi

Mengizinkan karakter mencetak ke aliran keluaran teks. Objek PrintWriter baru dibuat dengan fungsi createWriter (). Agar file dibuat dengan benar, ia harus dibilas dan harus ditutup dengan metode flush () dan close () (lihat contoh di atas)

Methods

print()	Menambahkan data ke aliran
println()	Menambahkan data ke aliran dan memulai baris baru
flush()	Membersihkan aliran
close()	Menutup aliran

saveBytes()

Examples

```
byte[] nums = { 0, 34, 5, 127, 52};

// Menulis byte ke file
saveBytes("numbers.dat", nums);
```


Diskripsi
Sebagai kebalikan dari loadBytes (), fungsi ini akan menulis seluruh array byte ke file. Data disimpan dalam format biner. File ini disimpan ke folder sketsa, yang dibuka dengan memilih "Tampilkan Folder Sketsa" dari menu "Sketsa". Atau, file dapat disimpan ke lokasi mana pun di komputer dengan menggunakan jalur absolut (sesuatu yang dimulai dengan / pada Unix dan Linux, atau huruf drive pada Windows).

Sintak
saveBytes(filename, data)

Parameter-parameter

filename	String: nama file yang akan ditulis
data	data []: array byte yang akan ditulis

saveJSONArray()

Examples

```
String[] species = { "Capra hircus", "Panthera pardus", "Equus zebra" };
String[] names = { "Goat", "Leopard", "Zebra" };

JSONArray values;

void setup() {

  values = new JSONArray();

  for (int i = 0; i < species.length; i++) {
```

```
    JSONObject animal = new JSONObject();

    animal.setInt("id", i);
    animal.setString("species", species[i]);
    animal.setString("name", names[i]);

    values.setJSONObject(i, animal);
  }

  saveJSONArray(values, "data/new.json");
}

// Sketch menyimpan yang berikut ke file yang disebut
"new.json":
// [
// {
// "id": 0,
// "spesies": "Capra hircus",
// "nama": "Kambing"
//},
// {
// "id": 1,
// "spesies": "Panthera pardus",
// "nama": "Leopard"
//},
// {
// "id": 2,
// "spesies": "Equus zebra",
// "nama": "Zebra"
//}
//]
```

Diskripsi

Menulis konten objek JSONArray ke file. Secara default, file ini disimpan ke folder sketsa. Folder ini dibuka dengan memilih "Tampilkan Folder Sketsa" dari menu "Sketsa".

Atau, file dapat disimpan ke lokasi mana pun di komputer dengan menggunakan jalur absolut (sesuatu yang dimulai dengan / pada Unix dan Linux, atau huruf drive pada Windows).

Semua file dimuat dan disimpan oleh API Pemrosesan menggunakan pengkodean UTF-8.

Sintak

saveJSONArray(json, filename)
saveJSONArray(json, filename, options)

Parameter-parameter

json	JSONArray: JSONArray untuk disimpan
filename	String: nama file yang akan disimpan
options	String: "compact" dan "indent = N", ganti N dengan jumlah spasi

saveJSONObject()

Examples

```
JSONObject json;

void setup() {

  json = new JSONObject();

  json.setInt("id", 0);
```

```
  json.setString("species", "Panthera leo");
  json.setString("name", "Lion");

  saveJSONObject(json, "data/new.json");
}

// Sketch menyimpan yang berikut ke file yang disebut
"new.json":
// {
// "id": 0,
// "spesies": "Panthera leo",
// "nama": "Singa"
//}
```

Diskripsi

Menulis konten objek JSONObject ke file. Secara default, file ini disimpan ke folder sketsa. Folder ini dibuka dengan memilih "Tampilkan Folder Sketsa" dari menu "Sketsa".

Atau, file dapat disimpan ke lokasi mana pun di komputer dengan menggunakan jalur absolut (sesuatu yang dimulai dengan / pada Unix dan Linux, atau huruf drive pada Windows).

Semua file dimuat dan disimpan oleh API Pemrosesan menggunakan pengkodean UTF-8.

Sintak

saveJSONObject(json, filename)
saveJSONObject(json, filename, options)

Parameter-parameter

json	JSONObject: JSONObject untuk disimpan
filename	String: nama file yang akan disimpan
options	String: "compact" dan "indent = N", ganti N dengan jumlah spasi

saveStream()

Diskripsi
Simpan konten stream ke file di folder sketsa. Ini pada dasarnya adalah saveBytes (bla, loadBytes ()), tetapi dilakukan lebih efisien (dan dengan sintaks yang kurang membingungkan).
Parameter target dapat berupa String yang menentukan nama file, atau, untuk kontrol yang lebih besar atas lokasi file, objek File. (Perhatikan bahwa, tidak seperti beberapa fungsi lain, ini tidak akan secara otomatis memampatkan atau mengompres file gzip.)

Sintak
saveStream(target, source)

Parameter-parameter

target	File, atau String: nama file untuk menulis
source	String: lokasi untuk membaca (nama file, jalur, atau URL)

saveStrings()

Examples

```
String words = "apple bear cat dog";
String[] list = split(words, ' ');

// Menulis string ke file, masing-masing pada baris terpisah
saveStrings("nouns.txt", list);
```

Diskripsi
Menulis array String ke file, satu baris per String. Secara default, file ini disimpan ke folder sketsa. Folder ini dibuka dengan memilih "Tampilkan Folder Sketsa" dari menu "Sketsa".
Atau, file dapat disimpan ke lokasi mana pun di komputer dengan menggunakan jalur absolut (sesuatu yang dimulai dengan / pada Unix dan Linux, atau huruf drive pada Windows).
Dimulai dengan Memproses 1.0, semua file dimuat dan disimpan oleh API Pemrosesan menggunakan pengkodean UTF-8. Dalam rilis sebelumnya, penyandian default untuk platform Anda digunakan, yang menyebabkan masalah ketika file dipindahkan ke platform lain.

Sintak
saveStrings(filename, data)

Parameter-parameter

filename	String: nama file untuk keluaran
data	String []: array string yang akan ditulis

saveTable()

Examples

```
Table table;

void setup() {

  table = new Table();

  table.addColumn("id");
  table.addColumn("species");
```


```
  table.addColumn ("name");

  TableRow newRow = table.addRow();
  newRow.setInt("id", table.getRowCount() - 1);
  newRow.setString("species", "Panthera leo");
  newRow.setString("name", "Lion");

  saveTable(table, "data/new.csv");
}

// Sketch menyimpan yang berikut ke file yang disebut
"new.csv":
// id, spesies, nama
// 0, Panthera leo, Lion
```

Diskripsi
Menulis konten objek Table ke file. Secara default, file ini disimpan ke folder sketsa. Folder ini dibuka dengan memilih "Tampilkan Folder Sketsa" dari menu "Sketsa".

Atau, file dapat disimpan ke lokasi mana pun di komputer dengan menggunakan jalur absolut (sesuatu yang dimulai dengan / pada Unix dan Linux, atau huruf drive pada Windows).
Semua file dimuat dan disimpan oleh API Pemrosesan menggunakan pengkodean UTF-8.

Sintak
saveTable(table, filename)
saveTable(table, filename, options)

Parameter-parameter

table	Tabel: objek Tabel untuk menyimpan ke file
filename	String: nama file yang harus disimpan oleh Tabel

options dapat berupa "tsv", "csv", "bin", atau "html"

saveXML()

Examples

```
// File XML pendek berikut yang disebut "mammals.xml"
diuraikan
// dalam kode di bawah ini. Itu harus ada di folder "data"
proyek.
//
// <? xml version = "1.0"?>
// <mamalia>
// <animal id = "0" species = "Capra hircus"> Kambing
</animal>
// <animal id = "1" species = "Panthera pardus"> Leopard
</animal>
// <animal id = "2" species = "Equus zebra"> Zebra
</animal>
// </mammals>

XML xml;

void setup() {
  xml = loadXML("mammals.xml");
  XML firstChild = xml.getChild("animal");
  xml.removeChild(firstChild);
  saveXML(xml, "subset.xml");
}

// Sketsa menyimpan yang berikut ke file yang disebut
"subset.xml":
// <? xml version = "1.0"?>
```


```
// <mamalia>
// <animal id = "1" species = "Panthera pardus"> Leopard
</animal>
// <animal id = "2" species = "Equus zebra"> Zebra
</animal>
// </mammals>
```

Diskripsi
Menulis konten objek XML ke file. Secara default, file ini disimpan ke folder sketsa. Folder ini dibuka dengan memilih "Tampilkan Folder Sketsa" dari menu "Sketsa".
Atau, file dapat disimpan ke lokasi mana pun di komputer dengan menggunakan jalur absolut (sesuatu yang dimulai dengan / pada Unix dan Linux, atau huruf drive pada Windows).

Sintak
saveXML(xml, filename)

Parameter-parameter

xml	objek XML yang akan disimpan ke String: nama file yang akan ditulis disk
filename	String: name of the file to write to

selectOutput()

Examples

```
void setup() {
  selectOutput("Select a file to write to:", "fileSelected");
}

void fileSelected(File selection) {
  if (selection == null) {
```

```
    println("Window was closed or the user hit cancel.");
  } else {
    println("User selected " + selection.getAbsolutePath());
  }
}
```

Diskripsi

Membuka dialog pemilih file platform-spesifik untuk memilih file untuk output. Setelah pemilihan dilakukan, File yang dipilih akan diteruskan ke fungsi 'panggilan balik'. Jika dialog ditutup atau dibatalkan, null akan dikirim ke fungsi, sehingga program tidak menunggu input tambahan. Callback diperlukan karena cara kerja threading.

Sintak

selectOutput(prompt, callback)
selectOutput(prompt, callback, file)
selectOutput(prompt, callback, file, callbackObject)
selectOutput(prompt, callbackMethod, file, callbackObject, parent)
selectOutput(prompt, callbackMethod, file, callbackObject, parent, sketch)

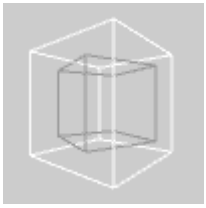
Parameter-parameter

prompt	String: pesan cepat ke pengguna
callback	String: nama metode yang akan dipanggil saat pemilihan dilakukan

BAB XX
Transform

applyMatrix()

Examples



```
size(100, 100, P3D);
noFill();
translate(50, 50, 0);
rotateY(PI/6);
stroke(153);
box(35);
// Atur sudut rotasi
float ct = cos(PI/9.0);
float st = sin(PI/9.0);
// Matriks untuk rotasi di sekitar sumbu Y
applyMatrix(  ct, 0.0,  st,  0.0,
              0.0, 1.0, 0.0,  0.0,
              -st, 0.0,  ct,  0.0,
              0.0, 0.0, 0.0,  1.0);
stroke(255);
box(50);
```

Diskripsi

Mengalikan matriks saat ini dengan yang ditentukan melalui parameter. Ini sangat lambat karena akan mencoba menghitung kebalikan dari transformasi, jadi hindarilah sebisa mungkin. Fungsi yang setara di OpenGL adalah glMultMatrix ().

Sintak

applyMatrix(source)
applyMatrix(n00, n01, n02, n10, n11, n12)
applyMatrix(n00, n01, n02, n03, n10, n11, n12, n13, n20, n21, n22, n23, n30, n31, n32, n33)

Parameter-parameter

n00 float: angka yang menentukan matriks 4x4 yang akan dikalikan

n01 float: angka yang menentukan matriks 4x4 yang akan dikalikan

n02 float: angka yang menentukan matriks 4x4 yang akan dikalikan

n10 float: angka yang menentukan matriks 4x4 yang akan dikalikan

n11 float: angka yang menentukan matriks 4x4 yang akan dikalikan

n12 float: angka yang menentukan matriks 4x4 yang akan dikalikan

n03 float: angka yang menentukan matriks 4x4 yang akan dikalikan

n13 float: angka yang menentukan matriks 4x4 yang akan dikalikan

n20 float: angka yang menentukan matriks 4x4 yang akan dikalikan

n21 float: angka yang menentukan matriks 4x4 yang akan dikalikan

n22 float: angka yang menentukan matriks 4x4 yang akan dikalikan

n23 float: angka yang menentukan matriks 4x4 yang akan dikalikan

n30 float: angka yang menentukan matriks 4x4 yang akan dikalikan

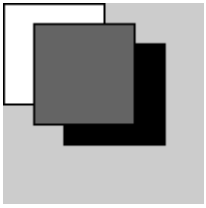
n31 float: angka yang menentukan matriks 4x4 yang akan dikalikan

n32 float: angka yang menentukan matriks 4x4 yang akan dikalikan

n33 float: angka yang menentukan matriks 4x4 yang akan dikalikan

popMatrix()

Examples



```
fill(255);
rect(0, 0, 50, 50);  // Kotak putih

pushMatrix();
translate(30, 20);
fill(0);
rect(0, 0, 50, 50);  // Kotak hitam
popMatrix();

fill(100);
rect(15, 10, 50, 50);  // Kotak abu-abu
```

Diskripsi

Pops matriks transformasi saat ini dari tumpukan matriks. Memahami mendorong dan muncul membutuhkan pemahaman konsep tumpukan matriks. Fungsi pushMatrix () menyimpan sistem koordinat saat ini ke stack dan popMatrix () mengembalikan sistem koordinat sebelumnya. pushMatrix () dan popMatrix () digunakan bersama dengan fungsi transformasi lainnya dan dapat tertanam untuk mengontrol ruang lingkup transformasi.

Sintak

popMatrix()

printMatrix()

Examples

```
size(100, 100, P3D);
printMatrix();
// Prints:
// 01.0000  00.0000  00.0000 -50.0000
// 00.0000  01.0000  00.0000 -50.0000
// 00.0000  00.0000  01.0000 -86.6025
// 00.0000  00.0000  00.0000  01.0000

resetMatrix();
printMatrix();
// Prints:
// 1.0000  0.0000  0.0000  0.0000
// 0.0000  1.0000  0.0000  0.0000
// 0.0000  0.0000  1.0000  0.0000
// 0.0000  0.0000  0.0000  1.0000
```

Diskripsi

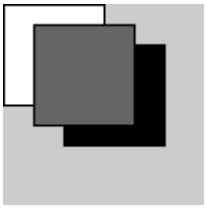
Mencetak matriks saat ini ke Konsol (jendela teks di bagian bawah Pemrosesan).

Sintak

```
printMatrix()
```

pushMatrix()

Examples



```
fill(255);
rect(0, 0, 50, 50);  // Kotak putih

pushMatrix();
translate(30, 20);
fill(0);
rect(0, 0, 50, 50);  // Kotak hitam
popMatrix();

fill(100);  Kotak abu-abu
```

Diskripsi

Dorong matriks transformasi saat ini ke tumpukan matriks. Memahami pushMatrix () dan popMatrix () membutuhkan pemahaman konsep tumpukan matriks. Fungsi pushMatrix () menyimpan sistem koordinat saat ini ke stack dan popMatrix () mengembalikan sistem koordinat sebelumnya. pushMatrix () dan popMatrix () digunakan bersama dengan fungsi transformasi lainnya dan dapat tertanam untuk mengontrol ruang lingkup transformasi.

Sintak

pushMatrix()

resetMatrix()

Examples

```
size(100, 100, P3D);
```

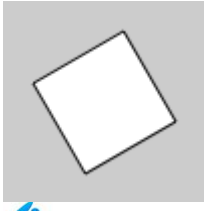
```
noFill();
box(80);
printMatrix();
// Prints:
// 01.0000  00.0000  00.0000 -50.0000
// 00.0000  01.0000  00.0000 -50.0000
// 00.0000  00.0000  01.0000 -86.6025
// 00.0000  00.0000  00.0000  01.0000

resetMatrix();
box(80);
printMatrix();
// Prints:
// 1.0000  0.0000  0.0000  0.0000
// 0.0000  1.0000  0.0000  0.0000
// 0.0000  0.0000  1.0000  0.0000
// 0.0000  0.0000  0.0000  1.0000
```

Diskripsi
Mengganti matriks saat ini dengan matriks identitas. Fungsi yang setara di OpenGL adalah glLoadIdentity ().

Sintak
resetMatrix()

rotate()

Examples



```
translate(width/2, height/2);
rotate(PI/3.0);
rect(-26, -26, 52, 52);
```

Diskripsi

Memutar jumlah yang ditentukan oleh parameter sudut. Sudut harus ditentukan dalam radian (nilai dari 0 hingga TWO_PI), atau mereka dapat dikonversi dari derajat ke radian dengan fungsi radians ().

Koordinat selalu diputar di sekitar posisi relatifnya ke titik asal. Bilangan positif memutar objek dalam arah searah jarum jam dan angka negatif berputar dalam arah berlawanan jarum jam. Transformasi berlaku untuk semua yang terjadi sesudahnya, dan panggilan selanjutnya ke fungsi menambah efek. Misalnya, memanggil rotate (PI / 2.0) satu kali dan kemudian memanggil rotate (PI / 2.0) kedua kalinya sama dengan single rotate (PI). Semua tranformasi diatur ulang saat draw () dimulai lagi.

Secara teknis, rotate () mengalikan matriks transformasi saat ini dengan matriks rotasi. Fungsi ini dapat lebih dikendalikan oleh pushMatrix () dan popMatrix ().

Sintak

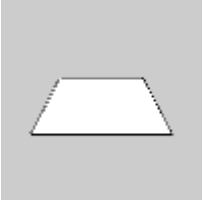
rotate(angle)

Parameter-parameter

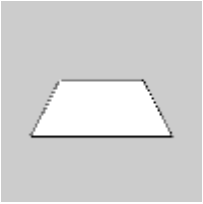
angle float: sudut rotasi yang ditentukan dalam radian

rotateX()

Examples



```
size(100, 100, P3D);
translate(width/2, height/2);
rotateX(PI/3.0);
rect(-26, -26, 52, 52);
```



```
size(100, 100, P3D);
translate(width/2, height/2);
rotateX(radians(60));
rect(-26, -26, 52, 52);
```

Diskripsi

Berputar di sekitar sumbu x jumlah yang ditentukan oleh parameter sudut. Sudut harus ditentukan dalam radian (nilai dari 0 hingga TWO_PI) atau dikonversi dari derajat ke radian dengan fungsi radians (). Koordinat selalu diputar di sekitar posisi relatifnya ke titik asal. Bilangan positif berputar searah jarum jam dan bilangan negatif berputar searah jarum jam. Transformasi berlaku untuk semua yang terjadi setelah dan panggilan selanjutnya ke fungsi mengakumulasi efek. Misalnya, memanggil rotateX (PI / 2) dan kemudian rotateX (PI / 2) sama dengan rotateX (PI). Jika rotateX () dijalankan di dalam draw (), transformasi diatur ulang ketika loop dimulai lagi. Fungsi ini mengharuskan menggunakan P3D sebagai parameter ketiga untuk ukuran () seperti yang ditunjukkan pada contoh di atas.

Sintak

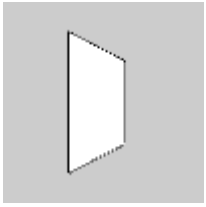
rotateX(angle)

Parameter-parameter

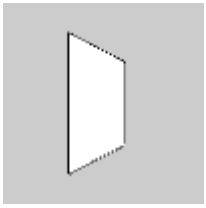
angle sudut rotasi yang ditentukan dalam radian

rotateY()

Examples



```
size(100, 100, P3D);
translate(width/2, height/2);
rotateY(PI/3.0);
rect(-26, -26, 52, 52);
```



```
size(100, 100, P3D);
translate(width/2, height/2);
rotateY(radians(60));
rect(-26, -26, 52, 52);
```

Diskripsi

Berputar di sekitar sumbu y jumlah yang ditentukan oleh parameter sudut. Sudut harus ditentukan dalam radian (nilai dari 0 hingga TWO_PI) atau dikonversi dari derajat ke radian dengan fungsi radians (). Koordinat selalu diputar di sekitar posisi relatifnya ke titik asal. Bilangan positif berputar searah jarum jam dan bilangan negatif berputar searah jarum jam. Transformasi berlaku untuk semua yang terjadi setelah dan panggilan selanjutnya ke fungsi mengakumulasi efek. Misalnya, memanggil rotateY (PI / 2) dan kemudian rotateY (PI / 2) sama dengan rotateY (PI). Jika rotateY () dijalankan di dalam draw (), transformasi diatur ulang ketika loop dimulai lagi. Fungsi ini membutuhkan penggunaan P3D.

Sintak

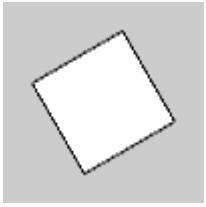
rotateY(angle)

Parameter-parameter

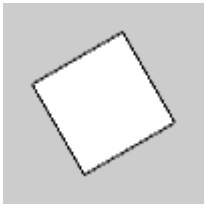
angle float: sudut rotasi yang ditentukan dalam radian

rotateZ()

Examples

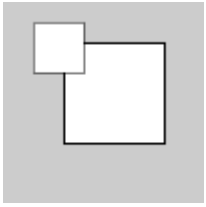


```
size(100, 100, P3D);
translate(width/2, height/2);
rotateZ(PI/3.0);
rect(-26, -26, 52, 52);
```



```
size(100, 100, P3D);
translate(width/2, height/2);
rotateZ(radians(60));
rect(-26, -26, 52, 52);
```

Diskripsi



Berputar di sekitar sumbu z jumlah yang ditentukan oleh parameter sudut. Sudut harus ditentukan dalam radian (nilai dari 0 hingga TWO_PI) atau dikonversi dari derajat ke radian dengan fungsi radians (). Koordinat selalu diputar di sekitar posisi relatifnya ke titik asal. Bilangan positif berputar searah jarum jam dan bilangan negatif berputar searah jarum jam. Transformasi berlaku untuk semua yang terjadi setelah dan panggilan selanjutnya ke fungsi mengakumulasi efek. Misalnya, memanggil rotateZ (PI / 2) dan kemudian rotateZ (PI / 2) sama dengan rotateZ (PI). Jika rotateZ () dijalankan di dalam draw (), transformasi diatur ulang

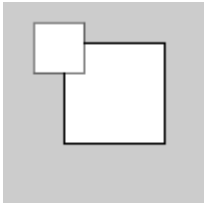
ketika loop dimulai lagi. Fungsi ini mengharuskan menggunakan P3D sebagai parameter ketiga untuk ukuran () seperti yang ditunjukkan pada contoh di atas.

Sintak
rotateZ(angle)

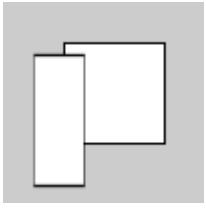
Parameter-parameter
angle float: sudut rotasi yang ditentukan dalam radian

scale()

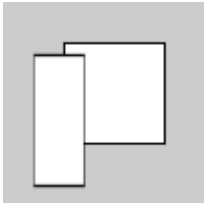
Examples



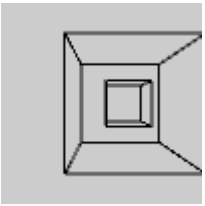
```
rect(30, 20, 50, 50);
scale(0.5);
rect(30, 20, 50, 50);
```



```
rect(30, 20, 50, 50);
scale(0.5);
rect(30, 20, 50, 50);
```



```
rect(30, 20, 50, 50);
scale(0.5, 1.3);
rect(30, 20, 50, 50);
```



```
// Penskalaan dalam 3D membutuhkan P3D
// sebagai parameter untuk ukuran ()
size(100, 100, P3D);
noFill();
translate(width/2+12, height/2);
box(20, 20, 20);
scale(2.5, 2.5, 2.5);
box(20, 20, 20);
```

Diskripsi

Menambah atau mengurangi ukuran bentuk dengan memperluas dan mengontrak simpul. Objek selalu skala dari asal relatifnya ke sistem koordinat. Nilai skala ditentukan sebagai persentase desimal. Misalnya, skala panggilan fungsi (2.0) meningkatkan dimensi bentuk sebesar 200%.

Transformasi berlaku untuk semua yang terjadi setelah dan panggilan selanjutnya ke fungsi, gandakan efeknya. Misalnya, skala panggilan (2.0) dan kemudian skala (1.5) sama dengan skala (3.0). Jika skala () disebut di dalam draw (), transformasi diatur ulang ketika loop dimulai lagi. Menggunakan fungsi ini dengan parameter z mengharuskan menggunakan P3D sebagai parameter untuk ukuran (), seperti yang ditunjukkan pada contoh ketiga di atas. Fungsi ini dapat dikontrol lebih lanjut dengan pushMatrix () dan popMatrix ().

Sintak

scale(s)
scale(x, y)
scale(x, y, z)

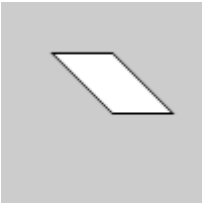
Parameter-parameter

s float: persentase untuk skala objek
x float: persentase untuk skala objek dalam sumbu x
y float: persentase untuk skala objek di sumbu y

z float: persentase untuk menskalakan objek pada sumbu-z

shearX()

Examples



```
size(100, 100);
translate(width/4, height/4);
shearX(PI/4.0);
rect(0, 0, 30, 30);
```

Diskripsi

Gunting bentuk di sekitar sumbu x jumlah yang ditentukan oleh parameter sudut. Sudut harus ditentukan dalam radian (nilai dari 0 hingga PI * 2) atau dikonversi ke radian dengan fungsi radian (). Objek selalu dicukur di sekitar posisi relatifnya ke titik asal dan bilangan positif menggeser objek dalam arah searah jarum jam. Transformasi berlaku untuk semua yang terjadi setelah dan panggilan selanjutnya ke fungsi mengakumulasi efek. Misalnya, memanggil shearX (PI / 2) dan kemudian shearX (PI / 2) sama dengan shearX (PI). Jika shearX () dipanggil di dalam draw (), transformasi diatur ulang ketika loop dimulai lagi.

Secara teknis, shearX () mengalikan matriks transformasi saat ini dengan matriks rotasi. Fungsi ini dapat lebih dikontrol oleh fungsi pushMatrix () dan popMatrix ().

Sintak

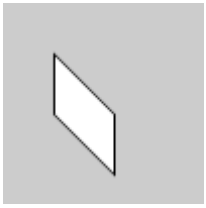
shearX(angle)

Parameter-parameter

angle float: sudut geser yang ditentukan dalam radian

shearY()

Examples



```
size(100, 100);
translate(width/4, height/4);
shearY(PI/4.0);
rect(0, 0, 30, 30);
```

Diskripsi

Gunting bentuk di sekitar sumbu y jumlah yang ditentukan oleh parameter sudut. Sudut harus ditentukan dalam radian (nilai dari 0 hingga PI * 2) atau dikonversi ke radian dengan fungsi radian (). Objek selalu dicukur di sekitar posisi relatifnya ke titik asal dan bilangan positif menggeser objek dalam arah searah jarum jam. Transformasi berlaku untuk semua yang terjadi setelah dan panggilan selanjutnya ke fungsi mengakumulasi efek. Misalnya, memanggil shearY (PI / 2) dan kemudian shearY (PI / 2) sama dengan shearY (PI). Jika shearY () dipanggil di dalam draw (), transformasi diatur ulang ketika loop dimulai lagi.

Secara teknis, shearY () mengalikan matriks transformasi saat ini dengan matriks rotasi. Fungsi ini dapat lebih dikontrol oleh fungsi pushMatrix () dan popMatrix ().

Sintak

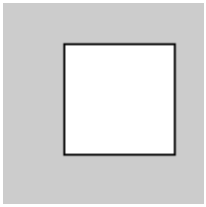
shearY(angle)

Parameter-parameter

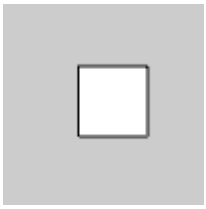
angle float: sudut geser yang ditentukan dalam radian

translate()

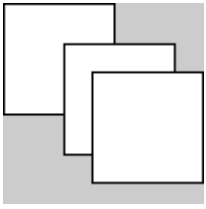
Examples



```
translate(30, 20);
rect(0, 0, 55, 55);
```



```
// Menerjemahkan dalam 3D membutuhkan P3D
// sebagai parameter untuk ukuran ()size(100, 100, P3D);
// Terjemahkan 30 ke seberang, 20 ke bawah, dan
// 50 kembali, atau "jauh" dari layar.
translate(30, 20, -50);
rect(0, 0, 55, 55);
```



```
rect(0, 0, 55, 55);  // Menggambar persegi pada aslinya 0,0
translate(30, 20);
rect(0, 0, 55, 55);  // Menggambar persegi di 0,0 barutranslate(14, 14);
rect(0, 0, 55, 55);  // Menggambar persegi di 0,0 baru
```

Diskripsi

Menentukan jumlah yang akan dipindahkan objek dalam jendela tampilan. Parameter x menentukan terjemahan kiri / kanan, parameter y menentukan terjemahan atas / bawah, dan parameter z menentukan terjemahan menuju / menjauh dari layar. Menggunakan fungsi ini dengan parameter z

membutuhkan menggunakan P3D sebagai parameter dalam kombinasi dengan ukuran seperti yang ditunjukkan pada contoh di atas.
Transformasi bersifat kumulatif dan berlaku untuk semua yang terjadi setelah dan panggilan selanjutnya ke fungsi mengakumulasi efek. Misalnya, memanggil terjemahan (50, 0) dan kemudian menerjemahkan (20, 0) sama dengan menerjemahkan (70, 0). Jika translate () dipanggil di dalam draw (), transformasi diatur ulang ketika loop dimulai lagi. Fungsi ini dapat dikontrol lebih lanjut dengan menggunakan pushMatrix () dan popMatrix ().

Sintak
translate(x, y)
translate(x, y, z)

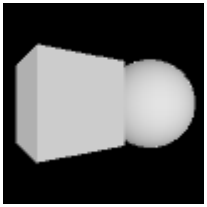
Parameter-parameter
x float: terjemahan kiri / kanan
y float: terjemahan atas bawah
z float: terjemahan depan belakang

BAB XXI
Cahaya dan Camera

ambientLight()

Examples

```
size(100, 100, P3D);
background(0);
noStroke();
// Bola-bola itu berwarna putih secara default
// lampu sekitar mengubah warnanya

ambientLight(51, 102, 126);
translate(20, 50, 0);
sphere(30);
translate(60, 0, 0);
sphere(30);
```



```
size(100, 100, P3D);
background(0);
noStroke();
directionalLight(126, 126, 126, 0, 0, -1);
ambientLight(102, 102, 102);
translate(32, 50, 0);
rotateY(PI/5);
box(40);
translate(60, 0, 0);
sphere(30);
```

Diskripsi

Menambahkan cahaya sekitar. Cahaya sekitar tidak datang dari arah tertentu, sinar cahaya telah memantul ke sana-sini sehingga benda-benda menyala secara merata dari semua sisi. Lampu ambient hampir selalu digunakan dalam kombinasi dengan jenis lampu lainnya. Lampu harus dimasukkan dalam

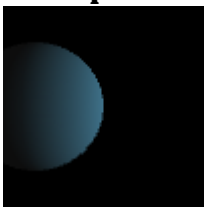
draw () untuk tetap bertahan dalam program looping. Menempatkan mereka di setup () dari program looping akan menyebabkan mereka hanya memiliki efek pertama kali melalui loop. Parameter v1, v2, dan v3 ditafsirkan sebagai nilai RGB atau HSB, tergantung pada mode warna saat ini.

Sintak
ambientLight(v1, v2, v3)
ambientLight(v1, v2, v3, x, y, z)

Parameter-parameter
v1 float: nilai merah atau rona (tergantung pada mode warna saat ini)
v2 float: nilai hijau atau saturasi (tergantung pada mode warna saat ini)
v3 float: nilai biru atau kecerahan (tergantung pada mode warna saat ini)
x loat: koordinat-x cahaya
y float: koordinat-y cahaya
z float: koordinat-z cahaya

directionalLight()

Examples



```
size(100, 100, P3D);
background(0);
noStroke();
directionalLight(51, 102, 126, -1, 0, 0);
translate(20, 50, 0);
sphere(30);
```

```
size(100, 100, P3D);
background(0);
noStroke();
directionalLight(51, 102, 126, 0, -1, 0);
translate(80, 50, 0);
sphere(30);
```

Diskripsi

Menambahkan cahaya arah. Cahaya directional datang dari satu arah: itu lebih kuat ketika mengenai permukaan secara tepat, dan lebih lemah jika mengenai sudut yang lembut. Setelah membentur permukaan, cahaya directional menyebar ke segala arah. Lampu harus dimasukkan dalam draw () untuk tetap bertahan dalam program looping. Menempatkan mereka di setup () dari program looping akan menyebabkan mereka hanya memiliki efek pertama kali melalui loop. Parameter v1, v2, dan v3 ditafsirkan sebagai nilai RGB atau HSB, tergantung pada mode warna saat ini. Parameter nx, ny, dan nz menentukan arah yang dihadapi cahaya. Misalnya, pengaturan ny ke -1 akan menyebabkan geometri menyala dari bawah (karena cahaya akan menghadap langsung ke atas).

Sintak

directionalLight(v1, v2, v3, nx, ny, nz)

Parameter-parameter

v1 float: nilai merah atau rona (tergantung pada mode warna saat ini)

v2 float: nilai hijau atau saturasi (tergantung pada mode warna saat ini)

v3 float: nilai biru atau kecerahan (tergantung pada mode warna saat ini)

nx float: arah sepanjang sumbu x

ny float: arah sepanjang sumbu y

nz float: arah sepanjang sumbu z

lightFalloff()

Examples

```
size(100, 100, P3D);
noStroke();
background(0);
lightFalloff(1.0, 0.001, 0.0);
pointLight(150, 250, 150, 50, 50, 50);
beginShape();
vertex(0, 0, 0);
vertex(100, 0, -100);
vertex(100, 100, -100);
vertex(0, 100, 0);
endShape(CLOSE);
```

Diskripsi

Mengatur tingkat kejatuhan untuk lampu titik, lampu sorot, dan lampu sekitar. Seperti fill (), itu hanya mempengaruhi elemen yang dibuat setelah itu dalam kode. Nilai defaultnya adalah lightFalloff (1.0, 0.0, 0.0), dan parameter digunakan untuk menghitung falloff dengan persamaan berikut:

d = jarak dari posisi cahaya ke posisi puncak
falloff = 1 / (cONSTAN + d * LINEAR + (d * d) * QUADRATIC)
Memikirkan cahaya sekitar dengan falloff bisa jadi rumit. Jika Anda ingin bagian dari adegan Anda menyala terang dengan satu warna dan wilayah lain menjadi terang dengan warna lain, Anda bisa menggunakan cahaya sekitar dengan lokasi dan

jatuh. Anda bisa menganggapnya sebagai titik cahaya yang tidak peduli ke arah mana permukaan menghadap.

Sintak

lightFalloff(constant, linear, quadratic)

Parameter-parameter

constant	float: nilai konstan atau menentukan falloff
linear	float: nilai linear untuk menentukan falloff
quadratic	float: nilai kuadrat untuk menentukan falloff

lights()

Examples

```
size(100, 100, P3D);
background(0);
noStroke();
// Mengatur ambient default
// dan cahaya directional
lights();
translate(20, 50, 0);
sphere(30);
translate(60, 0, 0);
sphere(30);
```

```
void setup() {
  size(100, 100, P3D);
  background(0);
  noStroke();
}
```

```
void draw() {
 // Sertakan lampu () di awal
  // of draw () untuk menjaga mereka tetap
ada lights();
 translate(20, 50, 0);
 sphere(30);
 translate(60, 0, 0);
 sphere(30);
}
```

Diskripsi

Sets the default ambient light, directional light, falloff, and specular values. The defaults are ambientLight(128, 128, 128) and directionalLight(128, 128, 128, 0, 0, -1), lightFalloff(1, 0, 0), and lightSpecular(0, 0, 0). Lights need to be included in the draw() to remain persistent in a looping program. Placing them in the setup() of a looping program will cause them to only have an effect the first time through the loop.

Sintak

lights()

lightSpecular()

Examples

```
size(100, 100, P3D);
background(0);
noStroke();
directionalLight(102, 102, 102, 0, 0, -1);
lightSpecular(204, 204, 204);
directionalLight(102, 102, 102, 0, 1, -1);
lightSpecular(102, 102, 102);
translate(20, 50, 0);
```

```
specular(51, 51, 51);
sphere(30);
translate(60, 0, 0);
specular(102, 102, 102);
sphere(30);
```

Diskripsi

Mengatur warna specular untuk lampu. Seperti fill (), itu hanya mempengaruhi elemen yang dibuat setelah itu dalam kode. Specular mengacu pada cahaya yang memantul dari permukaan ke arah yang lebih disukai (daripada memantul ke segala arah seperti cahaya difus) dan digunakan untuk membuat highlight. Kualitas specular cahaya berinteraksi dengan kualitas material specular yang ditetapkan melalui fungsi specular () dan shininess ().

Sintak

lightSpecular(v1, v2, v3)

Parameter-parameter

v1 float: nilai merah atau rona (tergantung pada mode warna saat ini)

v2 float: nilai hijau atau saturasi (tergantung pada mode warna saat ini)

v3 float: nilai biru atau kecerahan (tergantung pada mode warna saat ini)

noLights()

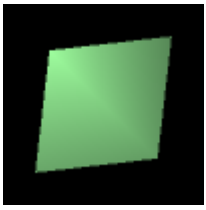
Diskripsi

Nonaktifkan semua pencahayaan. Pencahayaan dimatikan secara default dan diaktifkan dengan fungsi lampu (). Fungsi ini dapat digunakan untuk menonaktifkan pencahayaan sehingga geometri 2D (yang tidak memerlukan pencahayaan) dapat ditarik setelah satu set geometri 3D yang menyala.

Sintak
noLights()

normal()

Examples



```
size(100, 100, P3D);
noStroke();
background(0);
pointLight(150, 250, 150, 10, 30, 50);
beginShape();
normal(0, 0, 1);
vertex(20, 20, -10);
vertex(80, 20, 10);
vertex(80, 80, -10);
vertex(20, 80, 10);
endShape(CLOSE);
```

Diskripsi
Menetapkan vektor normal saat ini. Digunakan untuk menggambar bentuk dan permukaan tiga dimensi, normal () menentukan vektor tegak lurus terhadap permukaan bentuk yang, pada gilirannya, menentukan bagaimana pencahayaan memengaruhinya. Memproses upaya untuk secara otomatis menetapkan normals ke bentuk, tetapi karena itu tidak sempurna, ini adalah pilihan yang lebih baik ketika Anda ingin lebih banyak kontrol. Fungsi ini identik dengan glNormal3f () di OpenGL

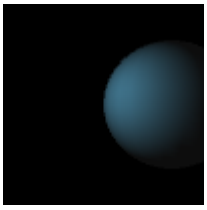
Sintak
normal(nx, ny, nz)

Parameter-parameter

nx float: arah x
ny float: arah y
nz float: arah z

pointLight()

Examples



```
size(100, 100, P3D);
background(0);
noStroke();
pointLight(51, 102, 126, 35, 40, 36);
translate(80, 50, 0);
sphere(30);
```

Diskripsi

Menambahkan titik cahaya. Lampu harus dimasukkan dalam draw () untuk tetap bertahan dalam program looping. Menempatkan mereka di setup () dari program looping akan menyebabkan mereka hanya memiliki efek pertama kali melalui loop. Parameter v1, v2, dan v3 ditafsirkan sebagai nilai RGB atau HSB, tergantung pada mode warna saat ini. Parameter x, y, dan z mengatur posisi cahaya.

Sintak

pointLight(v1, v2, v3, x, y, z)

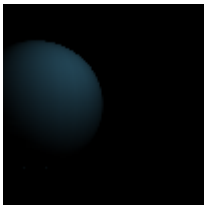
Parameter-parameter

v1 float: nilai merah atau rona (tergantung pada mode warna saat ini)
v2 float: nilai hijau atau saturasi (tergantung pada mode warna saat ini)
v3 float: nilai biru atau kecerahan (tergantung pada mode warna saat ini)
x float: koordinat-x cahaya

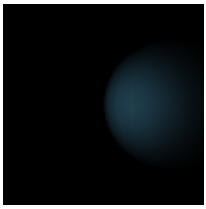
y float: koordinat-y cahaya
z float: z-koordinat cahaya

spotLight()

Examples



```
size(100, 100, P3D);
background(0);
noStroke();
spotLight(51, 102, 126, 80, 20, 40, -1, 0, 0,
PI/2, 2);
translate(20, 50, 0);
sphere(30);
```



```
size(100, 100, P3D);
int concentration = 600;  // Coba 1 -> 10000
background(0);
noStroke();
spotLight(51, 102, 126, 50, 50, 400,
        0, 0, -1, PI/16, concentration);
translate(80, 50, 0);
sphere(30);
```

Diskripsi

Menambahkan lampu sorot. Lampu harus dimasukkan dalam draw () untuk tetap bertahan dalam program looping. Menempatkan mereka di setup () dari program looping akan menyebabkan mereka hanya memiliki efek pertama kali melalui loop. Parameter v1, v2, dan v3 ditafsirkan sebagai nilai RGB atau HSB, tergantung pada mode warna saat ini. Parameter x, y, dan z menentukan posisi cahaya dan nx, ny, nz menentukan arah cahaya. Parameter sudut mempengaruhi sudut kerucut

sorotan, sementara konsentrasi mengatur bias cahaya yang fokus ke pusat kerucut itu.

Sintak

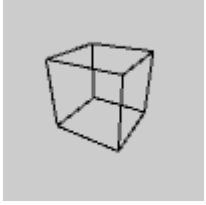
spotLight(v1, v2, v3, x, y, z, nx, ny, nz, angle, concentration)

Parameter-parameter

v1 float	:	nilai merah atau rona (tergantung pada mode warna saat ini)
v2 float	:	nilai hijau atau saturasi (tergantung pada mode warna saat ini)
v3 float	:	nilai biru atau kecerahan (tergantung pada mode warna saat ini)
x float	:	koordinat-x cahaya
y float	:	koordinat-y cahaya
z float	:	z-koordinat cahaya
nx float	:	arah sepanjang sumbu x
ny float	:	arah sepanjang sumbu y
nz float	:	arah sepanjang sumbu z
angle float	:	sudut kerucut sorotan
concentration float	:	eksponen menentukan bias pusat kerucut

beginCamera()

Examples



```
size(100, 100, P3D);
noFill();

beginCamera();
camera();
rotateX(-PI/6);
endCamera();
```

```
translate(50, 50, 0);
rotateY(PI/3);
box(45);
```

Diskripsi

Fungsi beginCamera () dan endCamera () memungkinkan penyesuaian lanjutan ruang kamera. Fungsi-fungsi ini berguna jika Anda ingin lebih mengontrol pergerakan kamera, namun bagi sebagian besar pengguna, fungsi kamera () akan cukup.

Fungsi-fungsi kamera akan menggantikan setiap transformasi (seperti rotate () atau translate ()) yang terjadi sebelum mereka di draw (), tetapi mereka tidak akan secara otomatis mengganti transformasi kamera itu sendiri. Karena alasan ini, fungsi kamera harus ditempatkan di awal draw () (sehingga transformasi terjadi sesudahnya), dan fungsi kamera () dapat digunakan setelah beginCamera () jika Anda ingin mengatur ulang kamera sebelum menerapkan transformasi.

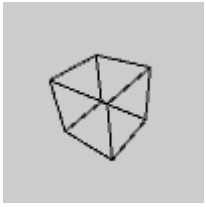
Fungsi ini mengatur mode matriks ke matriks kamera sehingga panggilan seperti translate (), rotate (), applyMatrix () dan resetMatrix () memengaruhi kamera. beginCamera () harus selalu digunakan dengan endCamera () dan pasangan beginCamera () dan endCamera () tidak dapat disarangkan.

Sintak

beginCamera()

camera()

Examples



```
size(100, 100, P3D);
noFill();
background(204);
camera(70.0, 35.0, 120.0, 50.0, 50.0, 0.0,
       0.0, 1.0, 0.0);
translate(50, 50, 0);
rotateX(-PI/6);
rotateY(PI/3);
box(45);
```

Diskripsi

Mengatur posisi kamera melalui pengaturan posisi mata, pusat adegan, dan sumbu mana yang menghadap ke atas. Menggerakkan posisi mata dan arah yang ditunjukkannya (bagian tengah adegan) memungkinkan gambar dilihat dari sudut yang berbeda. Versi tanpa parameter apa pun mengatur kamera ke posisi default, menunjuk ke tengah jendela tampilan dengan sumbu Y sebagai atas. Nilai standarnya adalah kamera (lebar / 2.0, tinggi / 2.0, (tinggi / 2.0) / tan (PI * 30.0 / 180.0), lebar / 2.0, tinggi / 2.0, 0, 0, 1, 0). Fungsi ini mirip dengan gluLookAt () di OpenGL, tetapi pertama-tama menghapus pengaturan kamera saat ini.

Sintak

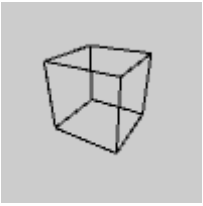
camera()
camera(*eye*X, *eye*Y, *eye*Z, centerX, centerY, centerZ, upX, upY, upZ)

Parameter-parameter

eyeX	float: koordinat-x untuk mata
eyeY	float: koordinat-y untuk mata
eyeZ	float: z-koordinat untuk mata
centerX	float: koordinat-x untuk pusat adegan
centerY	float: koordinat y untuk pusat adegan
centerZ	float: z-koordinat untuk pusat adegan
upX	float: biasanya 0,0, 1.0, atau -1.0
upY	float: biasanya 0.0, 1.0, or -1.0
upZ	float: biasanya 0.0, 1.0, or -1.0

endCamera ()

Examples



```
size(100, 100, P3D);
noFill();
beginCamera();
camera();
rotateX(-PI/6);
endCamera();
translate(50, 50, 0);
rotateY(PI/3);
box(45);
```

Diskripsi

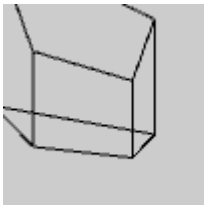
Fungsi beginCamera () dan endCamera () memungkinkan penyesuaian lanjutan ruang kamera. Silakan lihat referensi untuk beginCamera () untuk deskripsi tentang bagaimana fungsi digunakan.

Sintak

endCamera()

frustum()

Examples



```
size(100, 100, P3D);
noFill();
background(204);
frustum(-10, 0, 0, 10, 10, 200);
rotateY(PI/6);
box(45);
```

Diskripsi

Menetapkan matriks perspektif seperti yang didefinisikan oleh parameter.

Frum adalah bentuk geometris: piramida dengan bagian atasnya terpotong. Dengan mata penonton di puncak imajiner piramida, keenam bidang frustum bertindak sebagai pesawat kliping saat menampilkan tampilan 3D. Dengan demikian, segala bentuk di dalam pesawat kliping diberikan dan terlihat; apa pun di luar pesawat itu tidak terlihat.

Mengatur frustum memiliki efek mengubah perspektif yang digunakan untuk adegan itu. Ini dapat dicapai secara lebih sederhana dalam banyak kasus dengan menggunakan perspektif ().

Perhatikan bahwa nilai dekat harus lebih besar dari nol (karena titik "piramida" frustum tidak dapat bertemu "di belakang" pemirsa). Demikian pula, nilai jauh harus lebih besar dari nilai dekat (karena bidang "jauh" dari frustum harus "lebih jauh" dari pemirsa daripada bidang dekat).

Bekerja seperti glFrustum, kecuali menghapus matriks perspektif saat ini daripada mengalikannya dengan itu.

Sintak

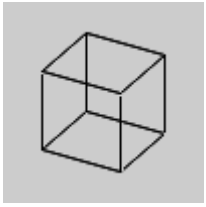
frustum(left, right, bottom, top, near, far)

Parameter-parameter

Left float	:	koordinat kiri dari pesawat kliping
Right float	:	koordinat kanan dari pesawat kliping
Bottom float	:	koordinat bawah dari pesawat kliping
Top float	:	koordinat atas dari pesawat kliping
Near float	:	dekat komponen bidang kliping; harus lebih besar dari nol
Far float	:	komponen jauh dari bidang kliping; harus lebih besar dari nilai dekat

ortho()

Examples



```
size(100, 100, P3D);
noFill();
ortho(-width/2, width/2, -height/2, height/2);
// Same as ortho()
translate(width/2, height/2, 0);
rotateX(-PI/6);
rotateY(PI/3);
box(45);
```

Diskripsi

Menetapkan proyeksi ortografis dan menentukan volume kliping paralel. Semua objek dengan dimensi yang sama muncul dengan ukuran yang sama, terlepas dari apakah mereka dekat atau jauh dari kamera. Parameter untuk fungsi ini menentukan volume kliping di mana kiri dan kanan adalah nilai x minimum dan maksimum, atas dan bawah adalah nilai y minimum dan maksimum, dan dekat dan jauh adalah nilai z minimum dan maksimum. Jika tidak ada parameter yang

diberikan, standar yang digunakan: ortho (-width / 2, lebar / 2, -tinggi / 2, tinggi / 2).

Sintak

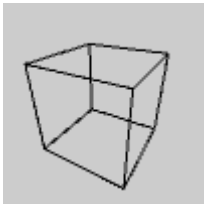
ortho()
ortho(left, right, bottom, top)
ortho(left, right, bottom, top, near, far)

Parameter-parameter

left	float: bidang kiri volume kliping
right	float: bidang kanan volume kliping
bottom	float: bidang bawah volume kliping
top	float: bidang atas volume kliping
near	float: jarak maksimum dari titik asal ke pengunjung
far	float: jarak maksimum dari asal jauh dari penampil

perspective()

Examples



```
// Menciptakan kembali perspektif
defaultsize(100, 100, P3D);
noFill();
float fov = PI/3.0;
float cameraZ = (height/2.0) / tan(fov/2.0);
perspective(fov, float(width)/float(height),
            cameraZ/10.0, cameraZ*10.0);
translate(50, 50, 0);
rotateX(-PI/6);
rotateY(PI/3);
box(45);
```

Diskripsi
Menetapkan proyeksi perspektif yang menerapkan foreshortening, membuat objek yang jauh tampak lebih kecil daripada yang lebih dekat. Parameter menentukan volume tampilan dengan bentuk piramida terpotong. Objek di dekat bagian depan volume muncul ukuran sebenarnya, sedangkan objek yang lebih jauh tampak lebih kecil. Proyeksi ini mensimulasikan perspektif dunia lebih akurat daripada proyeksi ortografis. Versi perspektif tanpa parameter menetapkan perspektif default dan versi dengan empat parameter memungkinkan programmer untuk mengatur area secara tepat. Nilai default adalah: perspektif (PI / 3.0, lebar / tinggi, cameraZ / 10.0, cameraZ * 10.0) di mana cameraZ berada ((tinggi / 2.0) / tan (PI * 60.0 / 360.0));

Sintak
perspective()
perspective(fovy, aspect, zNear, zFar)

Parameter-parameter
fovy float: sudut pandang bidang (dalam radian) untuk arah vertikal
aspect float: rasio lebar ke tinggi
zNear float: z-posisi bidang kliping terdekat
zFar float: z-posisi bidang kliping terjauh

printCamera()

Examples
size(100, 100, P3D);
printCamera();

// Program di atas mencetak data ini:

```
// 01.0000  00.0000  00.0000 -50.0000
// 00.0000  01.0000  00.0000 -50.0000
// 00.0000  00.0000  01.0000 -86.6025
// 00.0000  00.0000  00.0000  01.0000
```

Diskripsi
Mencetak matriks kamera saat ini ke Konsol (jendela teks di bagian bawah Pemrosesan).

Sintak
printCamera()

printProjection()

Examples

```
size(100, 100, P3D);
printProjection();
// Program di atas mencetak data ini:
// 01.7321  00.0000  00.0000  00.0000
// 00.0000 -01.7321  00.0000  00.0000
// 00.0000  00.0000 -01.0202 -17.4955
// 00.0000  00.0000 -01.0000  00.0000
```

Diskripsi
Mencetak matriks proyeksi saat ini ke Konsol (jendela teks di bagian bawah Pemrosesan).

Sintak
printProjection()

modelX()

Examples

```
void setup() {
  size(500, 500, P3D);
  noFill();
}

void draw() {
  background(0);

  pushMatrix();
  // mulai dari tengah layar
  translate(width/2, height/2, -200);
  // beberapa rotasi acak untuk membuat hal-hal menarik
  rotateY(1.0); //yrot);
  rotateZ(2.0); //zrot);
  // putar di X sedikit lebih banyak setiap frame
  rotateX(frameCount / 100.0);
  // diimbangi dari pusat
  translate(0, 150, 0);

  // gambar garis kotak putih di (0, 0, 0)
  stroke(255);
  box(50);

  // kotak itu digambar di (0, 0, 0), simpan lokasi itu
  float x = modelX(0, 0, 0);
  float y = modelY(0, 0, 0);
  float z = modelZ(0, 0, 0);
  // bersihkan semua transformasi
  popMatrix();
```

```
  // gambar kotak lain pada koordinat yang sama (x, y, z) dengan
  pushMatrix();
  translate(x, y, z);
  stroke(255, 0, 0);
  box(50);
  popMatrix();
}
```

Diskripsi

Mengembalikan posisi X, Y, Z tiga dimensi dalam ruang model. Ini mengembalikan nilai X untuk koordinat yang diberikan berdasarkan set transformasi saat ini (skala, memutar, menerjemahkan, dll.) Nilai X dapat digunakan untuk menempatkan objek di ruang relatif terhadap lokasi titik asli setelah transformasi dilakukan tidak lagi digunakan.

Dalam contoh ini, fungsi modelX (), modelY (), dan modelZ () merekam lokasi kotak di ruang angkasa setelah ditempatkan menggunakan serangkaian perintah terjemahkan dan putar. Setelah popMatrix () dipanggil, transformasi tersebut tidak lagi berlaku, tetapi koordinat (x, y, z) yang dikembalikan oleh fungsi model digunakan untuk menempatkan kotak lain di lokasi yang sama.

Sintak

modelX(x, y, z)

Parameter-parameter

x float: 3D x-coordinate yang akan dipetakan
y float: 3D y-coordinate yang akan dipetakan
z float: 3D z-coordinate yang akan dipetakan

modelY()

Examples

```
void setup() {
  size(500, 500, P3D);
  noFill();
}

void draw() {
  background(0);

  pushMatrix();
  // mulai dari tengah layar
  translate(width/2, height/2, -200);
  // beberapa rotasi acak untuk membuat hal-hal menarik
  rotateY(1.0); //yrot);
  rotateZ(2.0); //zrot);
  // putar di X sedikit lebih banyak setiap frame
  rotateX(frameCount / 100.0);
  // diimbangi dari pusat
  translate(0, 150, 0);

  // gambar garis kotak putih di (0, 0, 0)
  stroke(255);
  box(50);

  // kotak itu digambar di (0, 0, 0), simpan lokasi itu
  float x = modelX(0, 0, 0);
  float y = modelY(0, 0, 0);
  float z = modelZ(0, 0, 0);
  // bersihkan semua transformasi
  popMatrix();
```

```
 // gambar kotak lain pada koordinat yang sama (x, y, z) dengan yang lainnya
  pushMatrix ();
  translate(x, y, z);
  stroke(255, 0, 0);
  box(50);
  popMatrix();
}
```

Diskripsi

Mengembalikan posisi X, Y, Z tiga dimensi dalam ruang model. Ini mengembalikan nilai Y untuk koordinat yang diberikan berdasarkan set transformasi saat ini (skala, memutar, menerjemahkan, dll.) Nilai Y dapat digunakan untuk menempatkan objek di ruang relatif terhadap lokasi titik asli setelah transformasi dilakukan tidak lagi digunakan.

Dalam contoh ini, fungsi modelX (), modelY (), dan modelZ () merekam lokasi kotak di ruang angkasa setelah ditempatkan menggunakan serangkaian perintah terjemahkan dan putar. Setelah popMatrix () dipanggil, transformasi tersebut tidak lagi berlaku, tetapi koordinat (x, y, z) yang dikembalikan oleh fungsi model digunakan untuk menempatkan kotak lain di lokasi yang sama.

Sintak

modelY(x, y, z)

Parameter-parameter

x float: 3D x-coordinate yang akan dipetakan
y float: 3D y-coordinate yang akan dipetakan
z float: 3D z-coordinate yang akan dipetakan

modelZ()

Examples

```
void setup() {
  size(500, 500, P3D);
  noFill();
}

void draw() {
  background(0);

  pushMatrix();
  // mulai dari tengah layar
  translate(width/2, height/2, -200);
  // beberapa rotasi acak untuk membuat hal-hal menarik
  rotateY(1.0); //yrot);
  rotateZ(2.0); //zrot);
  // putar di X sedikit lebih banyak setiap frame
  rotateX(frameCount / 100.0);
  // diimbangi dari pusat
  translate(0, 150, 0);

  // gambar garis kotak putih di (0, 0, 0)
  stroke(255);
  box(50);

  // kotak itu digambar di (0, 0, 0), simpan lokasi itu
  float x = modelX(0, 0, 0);
  float y = modelY(0, 0, 0);
  float z = modelZ(0, 0, 0);
  // bersihkan semua transformasi
  popMatrix();
```

```
  // gambar kotak lain pada koordinat yang sama (x, y, z) dengan yang lainnya
  pushMatrix();
  translate(x, y, z);
  stroke(255, 0, 0);
  box(50);
  popMatrix();
}
```

Diskripsi

Mengembalikan posisi X, Y, Z tiga dimensi dalam ruang model. Ini mengembalikan nilai Z untuk koordinat yang diberikan berdasarkan set transformasi saat ini (skala, memutar, menerjemahkan, dll.) Nilai Z dapat digunakan untuk menempatkan objek di ruang relatif terhadap lokasi titik asli setelah transformasi dilakukan tidak lagi digunakan.

Dalam contoh ini, fungsi modelX (), modelY (), dan modelZ () merekam lokasi kotak di ruang angkasa setelah ditempatkan menggunakan serangkaian perintah terjemahkan dan putar. Setelah popMatrix () dipanggil, transformasi tersebut tidak lagi berlaku, tetapi koordinat (x, y, z) yang dikembalikan oleh fungsi model digunakan untuk menempatkan kotak lain di lokasi yang sama.

Sintak

modelZ(x, y, z)

Parameter-parameter

x float: 3D x-coordinate yang akan dipetakan
y float: 3D y-coordinate yang akan dipetakan
z float: 3D z-coordinate yang akan dipetakan

screenX()

Examples

```
void setup() {
  size(100, 100, P3D);
}

void draw() {
  background(204);

  float x = mouseX;
  float y = mouseY;
  float z = -100;

  // Gambar "X" di z = -100
  stroke(255);
  line(x-10, y-10, z, x+10, y+10, z);
  line(x+10, y-10, z, x-10, y+10, z);

  // Gambar garis abu-abu pada z = 0 dan sama
  // nilai x. Perhatikan paralaks stroke(102);
  line(x, 0, 0, x, height, 0);

  // Gambar garis hitam pada z = 0 untuk mencocokkan
  // elemen nilai x ditarik pada z = -100
  stroke(0);
  float theX = screenX(x, y, z);
  line(theX, 0, 0, theX, height, 0);
}
```

Diskripsi

Mengambil posisi X, Y, Z tiga dimensi dan mengembalikan nilai X di mana ia akan muncul pada layar (dua dimensi).

Sintak
screenX(x, y)
screenX(x, y, z)

Parameter-parameter
x float: 3D x-coordinate yang akan dipetakan
y float: 3D y-coordinate yang akan dipetakan
z float: 3D z-coordinate yang akan dipetakan

screenY()

Examples
```
void setup() {
  size(100, 100, P3D);
}

void draw() {
  background(204);

  float x = mouseX;
  float y = mouseY;
  float z = -100;

  // Gambar "X" di z = -100
  stroke(255);
  line(x-10, y-10, z, x+10, y+10, z);
  line(x+10, y-10, z, x-10, y+10, z);

  // Gambar garis abu-abu pada z = 0 dan sama
  // nilai y. Perhatikan paralaks
  stroke(102);
```


```
  line(0, y, 0, width, y, 0);

  // Gambar garis hitam pada z = 0 untuk mencocokkan
  // elemen nilai y ditarik pada z = -100
  stroke(0);
  float theY = screenY(x, y, z);
  line(0, theY, 0, width, theY, 0);
}
```

Diskripsi
Mengambil posisi X, Y, Z tiga dimensi dan mengembalikan nilai Y untuk kemunculannya pada layar (dua dimensi).

Sintak
screenY(x, y)
screenY(x, y, z)

Parameter-parameter
x float: 3D x-coordinate yang akan dipetakan
y float: 3D y-coordinate yang akan dipetakan
z float: 3D z-coordinate yang akan dipetakan

screenZ()

Diskripsi
Mengambil posisi X, Y, Z tiga dimensi dan mengembalikan nilai Z di mana ia akan muncul pada layar (dua dimensi).

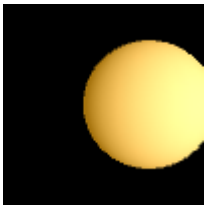
Sintak
screenZ(x, y, z)

Parameter-parameter

x float: 3D x-coordinate yang akan dipetakan
y float: 3D y-coordinate yang akan dipetakan
z float: 3D z-coordinate yang akan dipetakan

ambient()

Examples



```
size(100, 100, P3D);
background(0);
noStroke();
directionalLight(153, 153, 153, .5, 0, -1);
ambientLight(153, 102, 0);
ambient(51, 26, 0);
translate(70, 50, 0);
sphere(30);
```

Diskripsi

Mengatur pemantulan ambient untuk bentuk yang digambar ke layar. Ini dikombinasikan dengan komponen lingkungan sekitar. Komponen warna yang ditetapkan melalui parameter menentukan pantulan. Misalnya dalam mode warna default, pengaturan v1 = 255, v2 = 127, v3 = 0, akan menyebabkan semua lampu merah memantul dan setengah dari lampu hijau memantul. Digunakan dalam kombinasi dengan emissive (), specular (), dan shininess () dalam mengatur sifat material dari bentuk.

Sintak

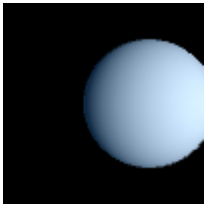
ambient(rgb)
ambient(gray)
ambient(v1, v2, v3)

Parameter-parameter

rgb int: nilai apa pun dari tipe data warna
gray float: angka yang menentukan nilai antara putih dan hitam
v1 float: nilai merah atau rona (tergantung pada mode warna saat ini)
v2 float: nilai hijau atau saturasi (tergantung pada mode warna saat ini)
v3 float: nilai biru atau kecerahan (tergantung pada mode warna saat ini)

emissive()

Examples



```
size(100, 100, P3D);
background(0);
noStroke();
background(0);
directionalLight(204, 204, 204, .5, 0, -1);
emissive(0, 26, 51);
translate(70, 50, 0);
sphere(30);
```

Diskripsi

Mengatur warna emisi bahan yang digunakan untuk menggambar bentuk yang digambar ke layar. Digunakan dalam kombinasi dengan ambient (), specular (), dan shininess () dalam mengatur sifat material dari bentuk.

Sintak

emissive(rgb)
emissive(gray)
emissive(v1, v2, v3)

Parameter-parameter

rgb int: warna untuk diatur

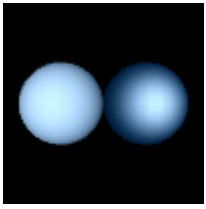
v1 float: nilai merah atau rona (tergantung pada mode warna saat ini))

v2 float: nilai hijau atau saturasi (tergantung pada mode warna saat ini)

v3 float: nilai biru atau kecerahan (tergantung pada mode warna saat ini)

shininess()

Examples



```
size(100, 100, P3D);
background(0);
noStroke();
background(0);
fill(0, 51, 102);
ambientLight(102, 102, 102);
lightSpecular(204, 204, 204);
directionalLight(102, 102, 102, 0, 0, -1);
specular(255, 255, 255);
translate(30, 50, 0);
shininess(1.0);
sphere(20);  // Left sphere
translate(40, 0, 0);
shininess(5.0);
sphere(20);  // Right sphere
```

Diskripsi

Mengatur jumlah gloss di permukaan bentuk. Digunakan dalam kombinasi dengan ambient (), specular (), dan emissive () dalam mengatur sifat material dari bentuk.

Sintak

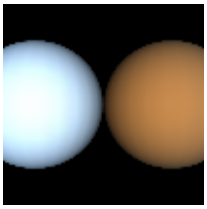
shininess(shine)

Parameter-parameter

shine float: tingkat kemilau

specular()

Examples



```
size(100, 100, P3D);
background(0);
noStroke();
background(0);
fill(0, 51, 102);
lightSpecular(255, 255, 255);
directionalLight(204, 204, 204, 0, 0, -1);
translate(20, 50, 0);
specular(255, 255, 255);
sphere(30);
translate(60, 0, 0);
specular(204, 102, 0);
sphere(30);
```

Diskripsi

Mengatur warna specular dari bahan yang digunakan untuk bentuk yang ditarik ke layar, yang mengatur warna highlight. Specular mengacu pada cahaya yang memantul dari permukaan ke arah yang lebih disukai (daripada memantul ke segala arah seperti cahaya difus). Digunakan dalam kombinasi dengan emissive (), ambient (), dan shininess () dalam mengatur sifat material dari bentuk.

Sintak

specular(rgb)
specular(gray)
specular(v1, v2, v3)

Parameter-parameter

rgb int : warna untuk diatur

v1 float : nilai merah atau rona (tergantung pada mode warna saat ini)

v2 float : nilai hijau atau saturasi (tergantung pada mode warna saat ini)

v3 float : nilai biru atau kecerahan (tergantung mod warna saat ini)

BAB XXII
COLOR

background()

Examples

background(51);

background(255, 204, 0)

```
PImage img;
img = loadImage("laDefense.jpg");
background(img);
```

Diskripsi

Fungsi background () mengatur warna yang digunakan untuk latar belakang jendela Pemrosesan. Latar belakang default adalah abu-abu terang. Fungsi ini biasanya digunakan di dalam draw () untuk menghapus jendela tampilan di awal setiap frame, tetapi dapat digunakan di dalam setup () untuk mengatur latar belakang pada frame animasi pertama atau jika backgound hanya perlu diatur sekali.

Gambar juga dapat digunakan sebagai latar belakang sketsa, meskipun lebar dan tinggi gambar harus sesuai dengan jendela

sketsa. Gambar yang digunakan dengan latar belakang () akan mengabaikan pengaturan tint () saat ini . Untuk mengubah ukuran gambar ke ukuran jendela sketsa, gunakan image.resize (lebar, tinggi).
Tidak mungkin untuk menggunakan parameter alpha transparansi dengan warna latar belakang pada permukaan gambar utama. Ini hanya dapat digunakan bersama dengan PGraphics objek dan CreateGraphics () .

Sintak

background(rgb)
background(rgb, alpha)
background(gray)
background(gray, alpha)
background(v1, v2, v3)
background(v1, v2, v3, alpha)
background(image)

Parameter-parameter

Rgb int	:	nilai apa pun dari datatype warna
Alpha float	:	opacity dari latar belakang
Gray float	:	menetapkan nilai antara putih dan hitam
v1 float	:	nilai merah atau rona (tergantung pada mode warna saat ini)
v2 float	:	nilai hijau atau saturasi (tergantung pada mode warna saat ini)
v3 float	:	nilai biru atau kecerahan (tergantung pada mode warna saat ini)
Image PImage	:	PImage untuk ditetapkan sebagai latar belakang (harus berukuran sama dengan jendela sketsa)

clear()

Examples

```
PGraphics pg;

void setup() {
  size(200, 200);
  pg = createGraphics(100, 100);
}

void draw() {
  background(204);
  pg.beginDraw();
  pg.stroke(0, 102, 153);
  pg.line(0, 0, mouseX, mouseY);
  pg.endDraw();
  image(pg, 50, 50);
}

// klik untuk menghapus objek
void mousePressed() {
  pg.beginDraw();
  pg.clear();
  pg.endDraw();
}
```

Diskripsi

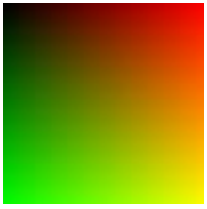
Membersihkan piksel dalam buffer. Fungsi ini hanya berfungsi pada objek PGraphics yang dibuat dengan fungsi createGraphics () . Berbeda dengan konteks grafik utama (jendela tampilan), piksel dalam area grafis tambahan yang dibuat dengan createGraphics () dapat seluruhnya atau sebagian transparan. Fungsi ini membersihkan semua yang ada

di objek PGraphics untuk membuat semua piksel 100% transparan.

Sintak
Clear()

colorMode ()

Examples



```
noStroke ();
colorMode (RGB, 100);
for (int i = 0; i <100; i ++) {

 for (int j = 0; j <100; j ++) {
 stroke (i, j, 0);
 point (i, j);
 }
}
```

```
noStroke ();
colorMode (HSB, 100);
for (int i = 0; i <100; i ++) {

 for (int j = 0; j <100; j ++) {
 stroke (i, j, 100);
 point (i, j);
  }
}
```

```
// Jika warnanya didefinisikan di sini, itu tidak akan
// terpengaruh oleh colorMode () di setup ().
// Sebaliknya, cukup deklarasikan variabel di sini dan
// tetapkan nilai setelah colorMode () di setup ()
// warna bg = warna (180, 50, 50); // Tidak
warna bg; // Ya, tapi tetapkan dalam pengaturan ()

void setup () {
  size (100, 100);
  colorMode (HSB, 360, 100, 100);
  bg = warna (180, 50, 50);
}

void draw () {
  background (bg);
}
```

Diskripsi

Mengubah cara Memproses data warna. Secara default, parameter untuk fill () , stroke () , background () , dan color () ditentukan oleh nilai antara 0 dan 255 menggunakan model warna RGB.

Fungsi colorMode () digunakan untuk mengubah rentang numerik yang digunakan untuk menentukan warna dan untuk beralih sistem warna.

Misalnya, memanggil colorMode (RGB, 1.0) akan menentukan bahwa nilai ditentukan antara 0 dan 1. Batas untuk mendefinisikan warna diubah dengan mengatur parameter max , max1 , max2 , max3 , dan maxA .

Setelah mengubah rentang nilai untuk warna dengan kode seperti colorMode (HSB, 360, 100, 100) , rentang itu tetap digunakan sampai mereka secara eksplisit diubah lagi. Misalnya, setelah menjalankan colorMode (HSB, 360, 100,

100) dan kemudian berubah kembali ke colorMode (RGB) , rentang untuk R akan menjadi 0 hingga 360 dan kisaran untuk G dan B akan menjadi 0 hingga 100. Untuk menghindari hal ini , secara eksplisit tentang rentang saat mengubah mode warna. Misalnya, alih-alih colorMode (RGB) , tulis colorMode (RGB, 255, 255, 255) .

Sintak

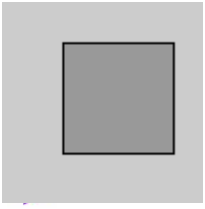
colorMode(mode)
colorMode(mode, max)
colorMode(mode, max1, max2, max3)
colorMode(mode, max1, max2, max3, maxA)

Parameter-parameter

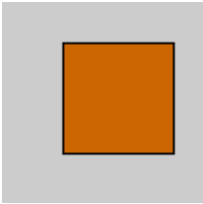
Mode int: Baik RGB atau HSB, sesuai dengan Merah / Hijau / Biru dan Hue / Saturasi / Kecerahan
Maks float: range untuk semua elemen warna
maks1 float: kisaran untuk warna merah atau rona tergantung pada mode warna saat ini
maks2 float: kisaran untuk warna hijau atau saturasi tergantung pada mode warna saat ini
max3 float: kisaran untuk biru atau kecerahan tergantung pada mode warna saat ini
maks float: range untuk alpha

fill()

Examples



```
fill(153);
rect(30, 20, 55, 55);
```



```
fill(204, 102, 0);
rect(30, 20, 55, 55);
```

Diskripsi

Mengatur warna yang digunakan untuk mengisi bentuk. Misalnya, jika Anda menjalankan isian (204, 102, 0) , semua bentuk berikutnya akan diisi dengan oranye. Warna ini ditentukan dalam warna RGB atau HSB tergantung pada colorMode saat ini () . Ruang warna default adalah RGB, dengan setiap nilai dalam rentang dari 0 hingga 255.

Saat menggunakan notasi heksadesimal untuk menentukan warna, gunakan " # " atau " 0x " sebelum nilai (misalnya, #CCFFAA atau 0xFFCCFFAA). The # sintaks menggunakan enam digit untuk menentukan warna (seperti warna biasanya ditentukan dalam HTML dan CSS).

Saat menggunakan notasi heksadesimal dimulai dengan " 0x", nilai heksadesimal harus ditentukan dengan delapan karakter; dua karakter pertama menentukan komponen alfa, dan sisanya menentukan komponen merah, hijau, dan biru.

Nilai untuk parameter" abu-abu "harus kurang dari atau sama dengan nilai maksimum saat ini sebagaimana ditentukan oleh colorMode () . Nilai maksimum default adalah 255.

Untuk mengubah warna gambar atau tekstur, gunakan tint ().

Sintak

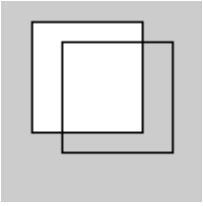
```
fill(rgb)
fill(rgb, alpha)
fill(gray)
fill(gray, alpha)
fill(v1, v2, v3)
fill(v1, v2, v3, alpha)
```

Parameter-parameter

rgb int: variabel warna atau nilai hex
alfa float: opacity dari fill
grey float: angka yang menentukan nilai antara putih dan hitam
v1 float: nilai merah atau rona (tergantung pada mode warna saat ini)
v2 float: nilai hijau atau saturasi (tergantung pada mode warna saat ini)
v3 float: nilai biru atau kecerahan (tergantung pada mode warna saat ini)

noFill()

Examples



```
rect(15, 10, 55, 55);
noFill();
rect(30, 20, 55, 55);
```

Diskripsi
Menonaktifkan pengisian geometri. Jika keduanya noStroke () dan noFill () dipanggil, tidak ada yang akan ditarik ke layar.

Sintak
noFill ()

noStroke()

Examples

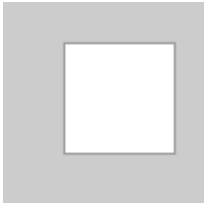

```
noStroke();
rect(30, 20, 55, 55);
```

Diskripsi
Menonaktifkan menggambar garis (garis besar). Jika keduanya noStroke () dan noFill () dipanggil, tidak ada yang akan ditarik ke layar.

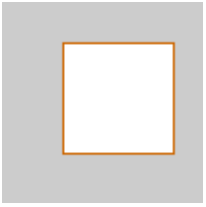
Sintak
noStroke ()

stroke()

Examples



```
stroke(153);
rect(30, 20, 55, 55);
```



```
stroke(204, 102, 0);
rect(30, 20, 55, 55);
```

Diskripsi

Mengatur warna yang digunakan untuk menggambar garis dan batas di sekitar bentuk. Warna ini ditentukan dalam warna RGB atau HSB tergantung pada colorMode saat ini (). Ruang warna default adalah RGB, dengan setiap nilai dalam rentang dari 0 hingga 255.

Saat menggunakan notasi heksadesimal untuk menentukan warna, gunakan " # " atau " 0x " sebelum nilai (misalnya, #CCFFAA atau 0xFFCCFFAA). The # sintaks menggunakan enam digit untuk menentukan warna (seperti warna biasanya ditentukan dalam HTML dan CSS). Saat menggunakan notasi heksadesimal dimulai dengan " 0x", nilai heksadesimal harus ditentukan dengan delapan karakter; dua karakter pertama menentukan komponen alfa, dan sisanya menentukan komponen merah, hijau, dan biru.

Nilai untuk parameter abu-abu harus kurang dari atau sama dengan maksimum saat ini nilai seperti yang ditentukan oleh colorMode () . Nilai maksimum default adalah 255.

Ketika menggambar dalam 2D dengan renderer default, Anda mungkin perlu petunjuk (ENABLE_STROKE_PURE) untuk meningkatkan kualitas gambar (dengan mengorbankan kinerja). Lihat petunjuk () dokumentasi untuk keterangan lebih lanjut.

Sintak

stroke (rgb)
stroke (rgb, alpha)
stroke (gray)
stroke (gray, alpha)
stroke (v1, v2, v3)
stroke (v1, v2, v3, alpha)

Parameter-parameter

Rgb	int: nilai warna dalam notasi heksadesimal
Alpha	float: opacity dari stroke

Grey float: menetapkan nilai antara putih dan hitam
V1 float: nilai merah atau rona (tergantung pada mode warna saat ini)
V2 float: nilai hijau atau saturasi (tergantung pada mode warna saat ini)
V3 float: nilai biru atau kecerahan (tergantung pada mode warna saat ini)

Creating & Reading

alpha()

Examples

```
noStroke();
color c = color(0, 126, 255, 102);
fill(c);
rect(15, 15, 35, 70);
float value = alpha(c);  // Sets 'value' to 102
fill(value);
rect(50, 15, 35, 70);
```

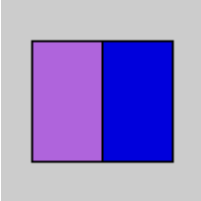
Diskripsi
Ekstrak nilai alpha dari suatu warna.

Sintak
alpha (rgb)

Parameter-parameter
Rgb int: nilai apa pun dari datatype warna

blue()

Examples



```
color c = color(175, 100, 220);  // tentukan warna 'c'
fill(c);  // Gunakan variabel warna 'c' sebagai warna isi
rect(15, 20, 35, 60);  // Gambarlah persegi panjang kiri
float blueValue = blue(c);  // Dapatkan biru di 'c'
println(blueValue);  // mencetak "220.0"
fill(0, 0, blueValue);  // Gunakan 'blueValue' dalam isian baru
rect(50, 20, 35, 60);  / Menggambar persegi panjang kanan
```

Diskripsi

Ekstrak nilai biru dari warna, diskalakan agar sesuai dengan colorMode saat ini () . Nilai selalu dikembalikan sebagai float, jadi berhati-hatilah untuk tidak menetapkannya ke nilai int.

Fungsi biru () mudah digunakan dan dipahami, tetapi lebih lambat dari teknik yang disebut bit masking. Saat bekerja dalam colorMode (RGB, 255) , Anda dapat memperoleh hasil yang sama seperti biru () tetapi dengan kecepatan yang lebih besar dengan menggunakan bit mask untuk menghapus komponen warna lainnya. Sebagai contoh, dua baris kode berikut adalah cara yang setara untuk mendapatkan nilai biru dari nilai warna c :

```
float b1 = biru (c); // Lebih sederhana, tetapi lebih lambat untuk menghitung
float b2 = c & 0xFF; // Sangat cepat untuk menghitung
```

Sintak
blue (rgb)

Parameter-parameter
Rgb int: nilai apa pun dari datatype warna

brightness()

Examples

```
noStroke ();
colorMode (HSB, 255);
warna c = warna (0, 126, 255);
isi (c);
rect (15, 20, 35, 60);
nilai float = kecerahan (c); // Tetapkan 'value' ke 255
isi (nilai);
rect (50, 20, 35, 60);
```

Diskripsi
Ekstrak nilai kecerahan dari suatu warna.

Sintak
brightness(rgb)

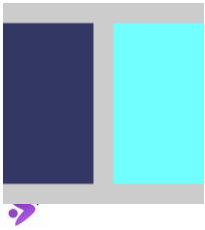
Parameter-parameter
rgb int: nilai apa pun dari datatype warna

color()

Examples

```
warna c = warna (255, 204, 0); // Tentukan warna 'c'
fill (c); // Gunakan variabel warna 'c' sebagai warna
noStroke (); // Jangan menggambar garis di sekitar bentuk
persegi panjang (30, 20, 55, 55); // Menggambar persegi panjang
```

```
warna c = warna (255, 204, 0); // Tentukan warna 'c'
fill (c); // Gunakan variabel warna 'c' sebagai warna
noStroke (); // Jangan menggambar garis bentuk
elips (25, 25, 80, 80); // Menggambar lingkaran kiri

// Menggunakan hanya satu nilai dengan warna ()
// menghasilkan nilai skala abu-abu.
c = warna (65); // Perbarui 'c' dengan
mengisi nilai skala abu-abu (c); // Gunakan 'c' yang diperbarui sebagai isian
ellipse warna (75, 75, 80, 80); // Gambarkan lingkaran kanan.
```



```
warna c; // Nyatakan warna 'c'
```

```
noStroke (); // Jangan menggambar garis di sekitar bentuk

// Jika tidak ada colorMode yang ditentukan, maka
// default RGB dengan skala 0-255 digunakan.
c = warna (50, 55, 100); // Buat warna untuk 'c'
fill (c); // Gunakan variabel warna 'c' sebagai warna mengisi
persegi (0, 10, 45, 80); // Menggambar

colorMode rect rectal (HSB, 100); // Gunakan HSB dengan skala 0-100
c = warna (50, 55, 100); // Perbarui 'c' dengan
mengisi warna baru (c); // Gunakan 'c' yang diperbarui sebagai kotak warna isi
(55, 10, 45, 80); // Menggambar kotak kanan
```

Diskripsi

Membuat warna untuk disimpan dalam variabel tipe data warna . Parameter diinterpretasikan sebagai nilai RGB atau HSB tergantung pada colorMode saat ini () . Mode standar adalah nilai RGB dari 0 hingga 255 dan, oleh karena itu, warna (255, 204, 0) akan mengembalikan warna kuning cerah (lihat contoh pertama di atas).

Perhatikan bahwa jika hanya satu nilai yang diberikan ke warna () , itu akan ditafsirkan sebagai nilai skala abu-abu. Tambahkan nilai kedua, dan itu akan digunakan untuk transparansi alfa. Ketika tiga nilai ditentukan, mereka ditafsirkan sebagai nilai

RGB atau HSB. Menambahkan nilai keempat berlaku transparansi alfa.
Perhatikan bahwa saat menggunakan notasi heksadesimal, tidak perlu menggunakan warna () , seperti pada:color c = # 006699.

Sintak

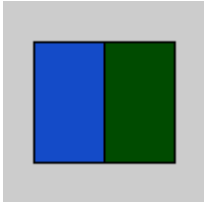
color(gray)
color(gray, alpha)
color(v1, v2, v3)
color(v1, v2, v3, alpha)

Parameter-parameter

grey int: angka yang menentukan nilai antara putih dan hitam
alpha float, atau int: relatif terhadap rentang warna saat ini
v1 float, atau int: nilai merah atau rona relatif terhadap rentang warna saat ini
v2 float, atau int: nilai hijau atau saturasi relatif terhadap rentang warna saat ini
v3 float, atau int: nilai-nilai biru atau kecerahan relatif terhadap rentang warna saat ini

green()

Examples



```
color c = color (20, 75, 200); // Tentukan warna 'c'
fill (c); // Gunakan variabel warna 'c' sebagai warna isi
rect (15, 20, 35, 60); // Gambarlah persegi panjang kiri
float greenValue = green (c); // Dapatkan hijau di 'c'
```

println (greenValue); // Cetak "75.0"
fill (0, greenValue, 0); // Gunakan 'greenValue' dalam isian baru
rect (50, 20, 35, 60); // Menggambar persegi panjang kanan

Diskripsi
Ekstrak nilai hijau dari warna, diskalakan agar sesuai dengan colorMode saat ini () . Nilai selalu dikembalikan sebagai float, jadi berhati-hatilah untuk tidak menetapkannya ke nilai int.
Fungsi hijau () mudah digunakan dan dipahami, tetapi lebih lambat dari teknik yang disebut bit shifting. Saat bekerja dalam colorMode (RGB, 255) , Anda dapat memperoleh hasil yang sama seperti hijau () tetapi dengan kecepatan yang lebih besar dengan menggunakan operator shift kanan (>>) dengan bit mask. Sebagai contoh, dua baris kode berikut ini adalah cara yang setara untuk mendapatkan nilai hijau dari nilai warna c :
float r1 = hijau (c); // Lebih sederhana, tetapi lebih lambat untuk menghitung
float r2 = c >> 8 & 0xFF; // Sangat cepat untuk menghitung

Sintak
green(rgb)

Parameter-parameter
Rgb int: nilai apa pun dari datatype warna

hue()

Examples

noStroke ();
colorMode (HSB, 255);
warna c = warna (0, 126, 255);
isi (c);

```
rect (15, 20, 35, 60);
nilai float = rona (c); // Tetapkan 'nilai' ke "0"
isi (nilai);
rect (50, 20, 35, 60);
```

Diskripsi
Ekstrak nilai rona dari warna.

Sintak
hue (rgb)

Parameter-parameter
Rgb int: nilai apa pun dari datatype warna

lerpColor()

Examples

```
stroke(255);
  background(51);
  color from = color(204, 102, 0);
  color to = color(0, 102, 153);
  color interA = lerpColor(from, to, .33);
  color interB = lerpColor(from, to, .66);
  fill(from);
  rect(10, 20, 20, 60);
  fill(interA);
  rect(30, 20, 20, 60);
  fill(interB);
  rect(50, 20, 20, 60);
  fill(to);
  rect(70, 20, 20, 60);
```

Diskripsi
Menghitung warna antara dua warna dengan kenaikan tertentu. The amt parameter jumlah yang interpolasi antara dua nilai di mana 0.0 sama dengan poin pertama, 0,1 sangat dekat titik pertama, 0,5 setengah jalan di antara, dll
Sebuah jumlah di bawah 0 akan diperlakukan sebagai 0. Demikian juga, jumlah di atas 1 akan dibatasi pada 1. Ini berbeda dari perilaku lerp (), tetapi perlu karena jika angka di luar kisaran akan menghasilkan warna yang aneh dan tidak terduga.

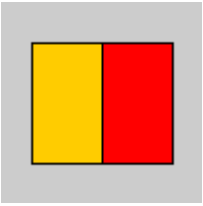
Sintak
lerpColor(c1, c2, amt)

Parameter-parameter

C1	int: interpolasi dari warna ini
C2	int: interpolasi dengan warna ini
Amt	float: antara 0.0 dan 1.0

red()

Examples



```
color c = color (255, 204, 0); // Tentukan warna 'c'
fill (c); // Gunakan variabel warna 'c' sebagai warna isi
rect (15, 20, 35, 60); // Gambarlah persegi panjang kiri

float redValue = red (c); // Dapatkan merah di 'c'
```

```
println (redValue); // Cetak "255.0"
fill (redValue, 0, 0); // Gunakan 'redValue' dalam isian baru
rect (50, 20, 35, 60); // Menggambar persegi panjang kanan
```

Diskripsi

Ekstrak nilai merah dari warna, diskalakan agar sesuai dengan colorMode saat ini () . Nilai selalu dikembalikan sebagai float, jadi berhati-hatilah untuk tidak menetapkannya ke nilai int.

Ekstrak nilai merah dari warna, diskalakan agar sesuai dengan colorMode saat ini () . Nilai selalu dikembalikan sebagai float, jadi berhati-hatilah untuk tidak menetapkannya ke nilai int.

Fungsi merah () mudah digunakan dan dipahami, tetapi lebih lambat dari teknik yang disebut bit shifting. Saat bekerja dalam colorMode (RGB, 255) , Anda dapat memperoleh hasil yang sama seperti merah () tetapi dengan kecepatan yang lebih besar dengan menggunakan operator shift kanan (>>) dengan bit mask. Sebagai contoh, dua baris kode berikut ini adalah cara yang setara untuk mendapatkan nilai merah dari nilai warna c :

```
float r1 = red (c); // Lebih sederhana, tetapi lebih lambat untuk menghitung
float r2 = c >> 16 & 0xFF; // Sangat cepat untuk menghitung
```

Sintak

```
red(rgb)
```

Parameter-parameter

Rgb int: nilai apa pun dari datatype warna

saturation()

Examples

```
noStroke ();
colorMode (HSB, 255);
warna c = warna (0, 126, 255);
isi (c);
rect (15, 20, 35, 60);
nilai float = saturasi (c); // Tetapkan 'nilai' ke
126
isi (nilai);
rect (50, 20, 35, 60);
```

Diskripsi

Ekstrak nilai saturasi dari suatu warna.

Sintak

saturation(rgb)

Parameter-parameter

Rgb int: nilai apa pun dari datatype warna

BAB XXIII
IMAGE

createImage()

Examples

```
PImage img = createImage(66, 66, RGB);
img.loadPixels();
for (int i = 0; i < img.pixels.length; i++) {
  img.pixels[i] = color(0, 90, 102);
}
img.updatePixels();
image(img, 17, 17);
```

```
PImage img = createImage(66, 66, ARGB);
img.loadPixels();
for (int i = 0; i < img.pixels.length; i++) {
  img.pixels[i] = color(0, 90, 102, i % img.width * 2);
}
img.updatePixels();
image(img, 17, 17);
image(img, 34, 34);
```

Diskripsi

Membuat PImage baru (tipe data untuk menyimpan gambar). Ini menyediakan buffer piksel baru untuk dimainkan. Atur ukuran buffer dengan parameter lebar dan tinggi . The Format parameter mendefinisikan bagaimana pixel disimpan. Lihat referensi PImage untuk informasi lebih lanjut.

Pastikan untuk memasukkan ketiga parameter, hanya menentukan lebar dan tinggi (tetapi tidak ada format) akan menghasilkan kesalahan aneh.

Pengguna mahir harap dicatat bahwa createImage () harus digunakan alih-alih sintaks PImage baru () .

Sintak

createImage (w, h, format)

Parameter-parameter

W	int: lebar dalam piksel
H	int: tinggi dalam piksel
Format	int: Baik RGB, ARGB, ALPHA (saluran alpha skala abu-abu)

PImage

Examples

```
PImage photo;

void setup() {
  size(100, 100);
  photo = loadImage("laDefense.jpg");
}

void draw() {
  image(photo, 0, 0);
}
```

Diskripsi

Datatype untuk menyimpan gambar. Pemrosesan dapat menampilkan gambar .gif , .jpg , .tga , dan .png . Gambar dapat ditampilkan dalam ruang 2D dan 3D. Sebelum gambar digunakan, gambar harus dimuat dengan fungsi loadImage () . Kelas PImage berisi bidang untuk lebar dan tinggi gambar, serta array yang disebut piksel [] yang berisi nilai untuk setiap piksel

dalam gambar. Metode yang dijelaskan di bawah ini memungkinkan akses mudah ke piksel gambar dan saluran alpha dan menyederhanakan proses pengomposisian.
Sebelum menggunakan array piksel [] , pastikan untuk menggunakan loadPixels ()metode pada gambar untuk memastikan bahwa data piksel dimuat dengan benar.
Untuk membuat gambar baru, gunakan fungsi createImage () . Jangan gunakan sintaks PImage baru () .

Loading & Displaying

image()

Examples

```
image img;

void setup () {
  // Gambar harus ada di direktori "data" untuk memuat dengan benar
  img = loadImage ("laDefense.jpg");
}

void draw () {
  gambar (img, 0, 0);
}
```

```
image img;
void setup () {
  // Gambar harus ada di direktori "data" untuk memuat dengan benar
  img = loadImage ("laDefense.jpg");
}
```

```
setup draw () {
  image (img, 0, 0);
  image (img, 0, 0, lebar / 2, tinggi / 2);
}
```

Diskripsi

Fungsi gambar () menarik gambar ke jendela tampilan. Gambar harus berada di direktori "data" sketsa agar dimuat dengan benar. Pilih "Tambah file ..." dari menu "Sketsa" untuk menambahkan gambar ke direktori data, atau cukup seret file gambar ke jendela sketsa. Pemrosesan saat ini berfungsi dengan gambar GIF, JPEG, dan PNG.

The img parameter menentukan gambar untuk menampilkan dan secara default suatu dan b parameter menentukan lokasi pojok kiri. Gambar ditampilkan pada ukuran aslinya kecuali parameter c dan d menentukan ukuran yang berbeda. Fungsi imageMode () dapat digunakan untuk mengubah cara parameter ini menggambar gambar.

Warna gambar dapat dimodifikasi dengan fungsi tint () . Fungsi ini akan menjaga transparansi untuk gambar GIF dan PNG.

Sintak

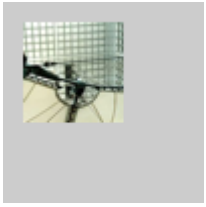
image (img, a, b)
image (img, a, b, c, d)

Parameter-parameter

Img	PImage: gambar yang akan ditampilkan
a	float: koordinat-x gambar secara default
b	float: koordinat-y gambar secara default
c	float: width untuk menampilkan gambar secara default
d	float: height untuk menampilkan gambar secara default

imageMode()

Examples



```
PImage img;

void setup() {
img = loadImage("laDefense.jpg");
}

void draw() {
  imageMode(CORNER);
  image(img, 10, 10, 50, 50);  // Draw image
using CORNER mode
}
```

```
PImage img;
void setup() {
img = loadImage("laDefense.jpg");
}

void draw() {
  imageMode(CORNERS);
  image(img, 10, 10, 90, 40);  //
}
```

```
PImage img;

void setup() {
  img = loadImage("laDefense.jpg");
}

void draw() {
```

```
  imageMode(CENTER);
  image(img, 50, 50, 80, 80);  // Draw image
using CENTER mode
}
```

Diskripsi

Memodifikasi lokasi dari mana gambar diambil dengan mengubah cara parameter yang diberikan kepada gambar () diintepretasikan.

Mode default adalah imageMode (CORNER) , yang mengartikan parameter gambar kedua dan ketiga () sebagai sudut kiri atas gambar. Jika dua parameter tambahan ditentukan, mereka digunakan untuk mengatur lebar dan tinggi gambar.

imageMode (CORNERS) mengartikan parameter gambar kedua dan ketiga () sebagai lokasi satu sudut, dan parameter keempat dan kelima sebagai sudut yang berlawanan.

imageMode (CENTER) mengartikan parameter gambar kedua dan ketiga ()sebagai titik tengah gambar. Jika dua parameter tambahan ditentukan, mereka digunakan untuk mengatur lebar dan tinggi gambar.

Parameter harus ditulis dalam SEMUA CAPS karena Pemrosesan adalah bahasa yang case-sensitive.

Sintak

imageMode (mode)

Parameter-parameter

Mode int: CORNER, CORNERS, atau CENTER

loadImage()

Examples

```
PImage img;
img = loadImage("laDefense.jpg");
image(img, 0, 0);
```

```
PImage img;
void setup() {
  img = loadImage("laDefense.jpg");
}
void draw() {
  image(img, 0, 0);
}
```

```
PImage webImg;
void setup() {
String url =
https://processing.org/img/processing-
web.png";
  // Load image from a web server
  webImg = loadImage(url, "png");
}
void draw() {
  background(0);
  image(webImg, 0, 0);
}
```

Diskripsi

Memuat gambar ke dalam variabel tipe PImage . Empat jenis gambar (.gif , .jpg , .tga , .png) dapat dimuat. Untuk memuat dengan benar, gambar harus berada di direktori data sketsa saat ini.

Dalam sebagian besar kasus, muat semua gambar dalam pengaturan () untuk memuatnya di awal program. Memuat gambar di dalam draw () akan mengurangi kecepatan suatu program. Gambar tidak dapat dimuat di luar pengaturan () kecuali mereka berada di dalam fungsi yang dipanggil setelah pengaturan () telah berjalan.

Atau, file mungkin dimuat dari mana saja di komputer lokal menggunakan jalur absolut (sesuatu yang dimulai dengan / pada Unix dan Linux, atau huruf drive pada Windows), atau parameter nama file dapat menjadi URL untuk file yang ditemukan pada file jaringan.

Jika file tidak tersedia atau kesalahan terjadi, null akan dikembalikan dan pesan kesalahan akan dicetak ke konsol. Pesan kesalahan tidak menghentikan program, namun nilai nol dapat menyebabkan NullPointerException jika kode Anda tidak memeriksa apakah nilai yang dikembalikan adalah nol.

The ekstensi parameter digunakan untuk menentukan jenis gambar dalam kasus di mana nama file gambar tidak berakhir dengan ekstensi yang tepat. Tentukan ekstensi sebagai parameter kedua untuk memuatImage (), seperti yang ditunjukkan pada contoh ketiga di halaman ini. Perhatikan bahwa gambar CMYK tidak didukung.

Bergantung pada jenis kesalahan, objek PImage mungkin masih dikembalikan, tetapi lebar dan tinggi gambar akan diatur ke -1. Ini terjadi jika data gambar yang buruk dikembalikan atau tidak dapat diterjemahkan dengan benar. Kadang-kadang ini terjadi dengan URL gambar yang menghasilkan kesalahan 403 atau yang mengarahkan ulang ke prompt kata sandi, karena

loadImage () akan berusaha menafsirkan HTML sebagai data gambar.

Sintak
loadImage (filename)
loadImage (filename, extension)

Parameter-parameter
File name String: nama file yang akan dimuat, bisa .gif, .jpg, .tga, atau beberapa jenis gambar lain tergantung pada platform Anda
Extension String: jenis gambar untuk memuat, misalnya "png", "gif", "jpg"

noTint()

Examples

```
image img;
img = loadImage ("laDefense.jpg");
tint (0, 153, 204); // Warna biru
image (img, 0, 0);
noTint (); // Nonaktifkan warna
image (img, 50, 0);
```

Diskripsi
Menghapus nilai isian saat ini untuk menampilkan gambar dan kembali ke menampilkan gambar dengan rona aslinya.

Sintak
noTint ()

requestImage ()

Examples

```
PImage bigImage;

void setup () {
  bigImage = requestImage ("something.jpg");
}

void draw () {
  if (bigImage.width == 0) {
    // Gambar belum dimuat
  } else if (bigImage.width == -1) {
    // Ini berarti kesalahan terjadi selama pemuatan gambar
  } else {
    // Gambar siap digunakan, gambarlah
    image (bigImage, 0, 0);
  }
}
```

Diskripsi

Fungsi ini memuat gambar pada utas terpisah sehingga sketsa Anda tidak membeku saat gambar dimuat selama penyiapan () . Saat gambar dimuat, lebar dan tingginya akan menjadi 0. Jika terjadi kesalahan saat memuat gambar, lebar dan tingginya akan ditetapkan ke -1. Anda akan tahu kapan gambar telah dimuat dengan benar karena lebarnya dan tinggi akan lebih besar dari 0. Memuat gambar yang tidak sinkron (terutama ketika mengunduh dari server) dapat secara dramatis meningkatkan kinerja.

The ekstensi parameter digunakan untuk menentukan jenis gambar dalam kasus di mana nama file gambar tidak berakhir

dengan ekstensi yang tepat. Tentukan ekstensi sebagai parameter kedua untuk requestImage () .

Sintak

requestImage (filename)
requestImage (filename, extension)

Parameter-parameter

filename String: nama file yang akan dimuat, bisa .gif, .jpg, .tga, atau beberapa jenis gambar lain tergantung pada platform Anda

extension String: jenis gambar yang akan dimuat, misalnya "png", "gif", "jpg"

tint()

Examples

```
image img;
img = loadImage ("laDefense.jpg");
image (img, 0, 0);
tint (0, 153, 204); // Mewarnai
image blue (img, 50, 0);
```

contoh foto

```
image img;
img = loadImage ("laDefense.jpg");
image (img, 0, 0);
tint (0, 153, 204, 126); // Mewarnai biru dan mengatur
image transparant (img, 50, 0);
```

contoh foto

```
image img;
img = loadImage ("laDefense.jpg");
image (img, 0, 0);
tint (255, 126); // Terapkan transparansi tanpa
mengubah
image color (img, 50, 0);
```

Diskripsi

Menetapkan nilai isi untuk menampilkan gambar. Gambar dapat diwarnai dengan warna yang ditentukan atau dibuat transparan dengan memasukkan nilai alfa.

Untuk menerapkan transparansi pada gambar tanpa mempengaruhi warnanya, gunakan putih sebagai warna warna dan tentukan nilai alfa. Misalnya, tint (255, 128) akan membuat gambar 50% transparan (dengan asumsi rentang alpha default 0-255, yang dapat diubah dengan colorMode ()).

Saat menggunakan notasi heksadesimal untuk menentukan warna, gunakan " # " atau " 0x " sebelum nilai (misalnya, #CCFFAA atau 0xFFCCFFAA). The #sintaks menggunakan enam digit untuk menentukan warna (seperti warna biasanya ditentukan dalam HTML dan CSS). Saat menggunakan notasi heksadesimal dimulai dengan " 0x ", nilai heksadesimal harus ditentukan dengan delapan karakter; dua karakter pertama menentukan komponen alfa, dan sisanya menentukan komponen merah, hijau, dan biru.

Nilai untuk parameter abu-abu harus kurang dari atau sama dengan nilai maksimum saat ini sebagaimana ditentukan oleh colorMode () . Nilai maksimum default adalah 255.

Fungsi tint () juga digunakan untuk mengontrol pewarnaan tekstur dalam 3D.

Sintak

tint(rgb)
tint(rgb, alpha)

tint(gray)
tint(gray, alpha)
tint(v1, v2, v3)
tint(v1, v2, v3, alpha)

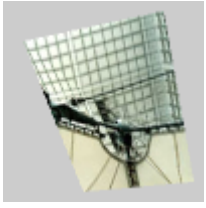
Parameter-parameter

Rgb int : nilai warna dalam notasi heksadesimal
Alpha float: opacity dari gambar
Gray float : menetapkan nilai antara putih dan hitam
v1 float : nilai merah atau rona (tergantung pada mode warna saat ini)
v2 float : nilai hijau atau saturasi (tergantung pada mode warna saat ini)
v3 float : nilai biru atau kecerahan (tergantung pada mode warna saat ini)

Textures

texture()

Examples



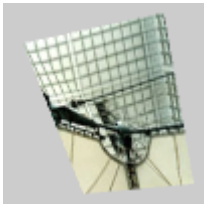
```
size(100, 100, P3D);
noStroke();
PImage img = loadImage("laDefense.jpg");
beginShape();
texture(img);
vertex(10, 20, 0, 0);
vertex(80, 5, 100, 0);
vertex(95, 90, 100, 100);
vertex(40, 95, 0, 100);
endShape();
```

Diskripsi
Setel tekstur yang akan diterapkan pada titik titik. Fungsi tekstur () harus dipanggil antara beginShape () dan endShape () dan sebelum panggilan ke vertex () . Fungsi ini hanya berfungsi dengan penyaji P2D dan P3D.
Saat tekstur sedang digunakan, warna isian diabaikan. Sebagai gantinya, gunakan tint () untuk menentukan warna tekstur saat diterapkan pada bentuk.

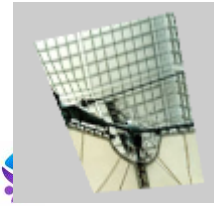
Sintak
texture(image)

Parameter-parameter
Image PImage: referensi ke objek Pimage

textureMode()



```
size(100, 100, P3D);
noStroke();
PImage img = loadImage("laDefense.jpg");
textureMode(IMAGE);
beginShape();
texture(img);
vertex(10, 20, 0, 0);
vertex(80, 5, 100, 0);
vertex(95, 90, 100, 100);
vertex(40, 95, 0, 100);
endShape();
```



example pic
size(100, 100, P3D);
noStroke();

```
PImage img = loadImage("laDefense.jpg");
textureMode(NORMAL);
beginShape();
texture(img);
vertex(10, 20, 0, 0);
vertex(80, 5, 1, 0);
vertex(95, 90, 1, 1);
vertex(40, 95, 0, 1);
endShape();
```

Diskripsi

Mengatur ruang koordinat untuk pemetaan tekstur. Mode standar adalah IMAGE , yang mengacu pada koordinat gambar yang sebenarnya. NORMAL mengacu pada ruang nilai yang dinormalisasi mulai dari 0 hingga 1. Fungsi ini hanya berfungsi dengan penyaji P2D dan P3D.

Dengan IMAGE , jika suatu gambar adalah 100 x 200 piksel, memetakan gambar ke seluruh ukuran quad akan membutuhkan poin (0,0) (100, 0) (100.200) (0,200). Pemetaan yang sama dalam NORMAL adalah (0,0) (1,0) (1,1) (0,1).

Sintak

textureMode(mode)

Parameter-parameter

Mode int : baik IMAGE atau NORMAL

textureWrap()

Examples

PImage img;

```
void setup() {
  size(300, 300, P2D);
  img = loadImage("berlin-1.jpg");
  textureMode(NORMAL);
}
void draw() {
  background(0);
  translate(width/2, height/2);
  rotate(map(mouseX, 0, width, -PI, PI));
  if (mousePressed) {
    textureWrap(REPEAT);
  } else {
    textureWrap(CLAMP);
  }
  beginShape();
  texture(img);
  vertex(-100, -100, 0, 0);
  vertex(100, -100, 2, 0);
  vertex(100, 100, 2, 2);
  vertex(-100, 100, 0, 2);
  endShape();
}
```

Diskripsi

Menentukan apakah tekstur mengulang atau menggambar sekali dalam peta tekstur. Dua parameter tersebut adalah CLAMP (perilaku default) dan REPEAT. Fungsi ini hanya berfungsi dengan penyaji P2D dan P3D.

Sintak

teksturWrap (wrap)

Parameter-parameter

Wrap int : CLAMP (default) atau REPEAT

BAB XXIV
PIXELS

blend()

Examples

```
background(loadImage("rockies.jpg"));
PImage img = loadImage("bricks.jpg");
image(img, 0, 0);
blend(img, 0, 0, 33, 100, 67, 0, 33, 100, ADD);
```

example pic

```
background(loadImage("rockies.jpg"));
PImage img = loadImage("bricks.jpg");
image(img, 0, 0);
blend(img, 0, 0, 33, 100, 67, 0, 33, 100, SUBTRACT);
```

example pic

```
background(loadImage("rockies.jpg"));
PImage img = loadImage("bricks.jpg");
image(img, 0, 0);
blend(img, 0, 0, 33, 100, 67, 0, 33, 100, DARKEST);
```

example pic

```
background(loadImage("rockies.jpg"));
PImage img = loadImage("bricks.jpg");
image(img, 0, 0);
blend(img, 0, 0, 33, 100, 67, 0, 33, 100, LIGHTEST);
```

Diskripsi

Memadukan wilayah piksel dari satu gambar ke gambar lain (atau dengan sendirinya lagi) dengan dukungan saluran alpha

penuh. Ada pilihan mode berikut untuk memadukan piksel sumber (A) dengan piksel dalam gambar tujuan (B):
BLEND - interpolasi warna linier: C = A * faktor + B
ADD - pencampuran aditif dengan klip putih : C = min (faktor A * + B, 255)
SUBTRACT - pencampuran subtraktif dengan klip hitam: C = maks (faktor B - A *, 0)
GELAP - hanya warna paling gelap yang berhasil: C = min (faktor A *, B)
LIGHTEST - hanya warna paling terang yang berhasil: C = maks (faktor A *, B)
PERBEDAAN - kurangi warna dari gambar yang mendasarinya.
PENGECUALIAN - mirip dengan PERBEDAAN, tetapi kurang ekstrim.
LEBIH BANYAK - Lipat gandakan warna, hasilnya akan selalu lebih gelap.
LAYAR - Berlipat ganda, menggunakan nilai kebalikan dari warna.
OVERLAY - Campuran BERBAGAI dan LAYAR. Mengalikan nilai gelap, dan menyaring nilai cahaya.

HARD_LIGHT - LAYAR bila lebih besar dari 50% abu-abu, LEBIH BANYAK saat lebih rendah.
SOFT_LIGHT - Campuran DARKEST dan LIGHTEST. Bekerja seperti OVERLAY, tetapi tidak sekeras itu.
DODGE - Mencerahkan nada cahaya dan meningkatkan kontras, mengabaikan kegelapan. Disebut "Color Dodge" di Illustrator dan Photoshop.
BURN - Area yang lebih gelap diterapkan, menambah kontras, mengabaikan cahaya. Disebut "Color Burn" di Illustrator dan Photoshop.
Semua mode menggunakan informasi alfa (byte tertinggi) dari piksel gambar sumber sebagai faktor campuran. Jika daerah sumber dan tujuan memiliki ukuran yang berbeda, gambar

akan secara otomatis diubah ukurannya agar sesuai dengan ukuran tujuan. Jika parameter src tidak digunakan, jendela tampilan digunakan sebagai gambar sumber.
Pada rilis 0149, fungsi ini mengabaikan imageMode () .

Sintak

blend(sx, sy, sw, sh, dx, dy, dw, dh, mode)
blend(src, sx, sy, sw, sh, dx, dy, dw, dh, mode)

Parameter-parameter

Src PImage: variabel gambar yang merujuk ke gambar sumber
sx int: X mengoordinasikan sudut kiri atas sumber
sy int: Y mengoordinasikan sudut kiri atas sumber
sw int: lebar gambar sumber
SH int: sumber tinggi gambar
dx int: X mengoordinasikan sudut kiri atas tujuan
dy int: Y mengoordinasikan sudut kiri atas tujuan
dw int: lebar gambar tujuan
dh int: tinggi gambar tujuan
mode int: Baik BLEND, ADD, SUBTRACT, LIGHTEST, DARKEST, PERBEDAAN, PENGECUALIAN, MULTIPLY, SCREEN, OVERLAY, HARD_LIGHT, SOFT_LIGHT, DODGE, BURN

copy()

Examples

```
PImage img = loadImage ("eames.jpg");
gambar (img, 0, 0, lebar, tinggi);
salinan (7, 22, 10, 10, 35, 25, 50, 50);
stroke (255);
noFill ();
```

```
// Rectangle menunjukkan area yang sedang disalin
rect (7, 22, 10, 10);
```

Diskripsi

Menyalin wilayah piksel dari jendela tampilan ke area lain dari jendela tampilan dan menyalin wilayah piksel dari gambar yang digunakan sebagai parameter srcImg ke dalam jendela tampilan. Jika daerah sumber dan tujuan tidak memiliki ukuran yang sama, secara otomatis akan mengubah ukuran piksel sumber agar sesuai dengan wilayah target yang ditentukan. Tidak ada informasi alfa yang digunakan dalam proses, namun jika gambar sumber memiliki set saluran alpha, itu akan disalin juga.

Pada rilis 0149, fungsi ini mengabaikan imageMode () .

Sintak

copy()
copy(sx, sy, sw, sh, dx, dy, dw, dh)
copy(src, sx, sy, sw, sh, dx, dy, dw, dh)

Paramaters

Sx int	:	X mengoordinasikan sudut kiri atas sumber
Sy int	:	Y mengoordinasikan sudut kiri atas sumber
Sw int	:	lebar gambar sumber
SH int	:	sumber tinggi gambar
Dx int	:	X mengoordinasikan sudut kiri atas tujuan
Dy int	:	Y mengoordinasikan sudut kiri atas tujuan
Dw int	:	lebar gambar tujuan
Dh int	:	tinggi gambar tujuan
Src PImage	:	variabel gambar yang merujuk ke gambar sumber.

filter()

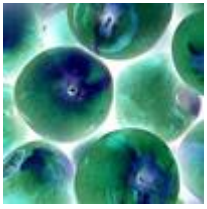
Examples

```
PImage img;
img = loadImage("apples.jpg");
image(img, 0, 0);
filter(THRESHOLD);
```

example pic

```
PImage img;
img = loadImage("apples.jpg");
image(img, 0, 0);
filter(GRAY);
```

example pic

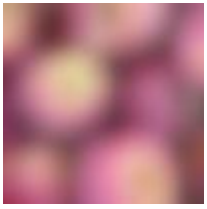


```
PImage img;
img = loadImage("apples.jpg");
image(img, 0, 0);
filter(INVERT);
```

example pic

```
PImage img;
img = loadImage("apples.jpg");
image(img, 0, 0);
filter(POSTERIZE, 4);
```

example pic



```
PImage img;
img = loadImage("apples.jpg");
image(img, 0, 0);
```

```
filter(BLUR, 6);
PShader blur;
PImage img;
void setup() {
size(100, 100, P2D);
blur = loadShader("blur.glsl");
img = loadImage("apples.jpg");
image(img, 0, 0);
}

void draw() {
  filter(blur); // Blurs more each time through
draw()
}
```

Diskripsi

Memfilter jendela tampilan menggunakan filter preset atau dengan shader khusus. Menggunakan shader dengan filter () jauh lebih cepat daripada tanpa. Shaders membutuhkan renderer P2D atau P3D dalam ukuran ().

Opsi preset adalah:

THRESHOLD

Mengubah gambar menjadi piksel hitam dan putih tergantung apakah mereka berada di atas atau di bawah ambang batas yang ditentukan oleh parameter level. Parameter harus antara 0,0 (hitam) dan 1,0 (putih). Jika tidak ada level yang ditentukan, 0,5 digunakan.

GREY

Mengubah warna apa pun pada gambar menjadi setara dengan skala abu-abu. Tidak ada parameter yang digunakan.

OPAQUE

Mengatur saluran alfa menjadi sepenuhnya buram. Tidak ada parameter yang digunakan.

INVERT

Mengatur setiap piksel ke nilai kebalikannya. Tidak ada parameter yang digunakan.
POSTERIZE
Membatasi setiap saluran gambar dengan jumlah warna yang ditentukan sebagai parameter. Parameter dapat diatur ke nilai antara 2 dan 255, tetapi hasilnya paling terlihat di rentang yang lebih rendah.
Blur
Melaksanakan kabur Guassian dengan parameter tingkat menentukan sejauh mana kabur tersebut. Jika tidak ada parameter yang digunakan, blur setara dengan Guassian blur radius 1. Nilai yang lebih besar meningkatkan blur.
ERODE
Mengurangi area cahaya. Tidak ada parameter yang digunakan.
DILATE
Meningkatkan area cahaya. Tidak ada parameter yang digunakan.

Sintak
filter (shader)
filter (kind)
filter (kind, param)

Parameter-parameter

Shader PShader	:	shader fragmen untuk diterapkan
Kind int	:	Entah THRESHOLD, GREY, OPAQUE, INVERT, POSTERIZE, BLUR, ERODE, atau DILATE
Param float	:	unik untuk masing-masing, lihat di atas

get()

Examples

```
PImage myImage = loadImage("apples.jpg");
image(myImage, 0, 0);
PImage c = get();
image(c, width/2, 0);
```

example pic

```
PImage myImage = loadImage("apples.jpg");
image(myImage, 0, 0);
color c = get(25, 25);
fill(c);
noStroke();
rect(25, 25, 50, 50);
```

Diskripsi

Membaca warna piksel apa pun atau mengambil bagian gambar. Jika tidak ada parameter yang ditentukan, seluruh gambar dikembalikan. Gunakan parameter x dan y untuk mendapatkan nilai satu piksel. Dapatkan bagian dari jendela tampilan dengan menentukan parameter w dan h tambahan . Saat mendapatkan gambar, parameter x dan y menentukan koordinat untuk sudut kiri atas gambar, terlepas dari imageMode saat ini () .

Jika piksel yang diminta berada di luar jendela gambar, hitam dikembalikan. Angka yang dikembalikan diskalakan sesuai dengan rentang warna saat ini, tetapi hanya nilai RGB yang dikembalikan oleh fungsi ini. Misalnya, meskipun Anda mungkin telah menggambar bentuk dengan colorMode (HSB) , angka yang dikembalikan akan dalam format RGB.

Jika lebar dan tinggi ditentukan, dapatkan (x, y, w, h) mengembalikan PImage yang sesuai dengan bagian dari PImage asli di mana piksel kiri atas berada pada posisi (x, y) dengan lebar w a ketinggian h .

Mendapatkan warna satu piksel dengan get (x, y) itu mudah, tetapi tidak secepat mengambil data langsung dari piksel []. Pernyataan setara untuk mendapatkan (x, y) menggunakan piksel [] adalah piksel [y * lebar + x] . Lihat referensi untuk piksel [] untuk informasi lebih lanjut.

Sintak

get(x, y)
get(x, y, w, h)
get()

Parameter-parameter

x	int: koordinat x piksel
y	int: koordinat-y dari piksel
w	int: lebar persegi panjang piksel untuk mendapatkan
h	int: tinggi persegi panjang piksel untuk mendapatkan

loadPixels()

Examples

```
int halfImage = width*height/2;
PImage myImage = loadImage("apples.jpg");
image(myImage, 0, 0);

loadPixels();
for (int i = 0; i < halfImage; i++) {
```

```
  pixels[i+halfImage] = pixels[i];
}
updatePixels();
```

Diskripsi

Memuat snapshot dari jendela tampilan saat ini ke dalam array piksel [] . Fungsi ini harus selalu dipanggil sebelum membaca dari atau menulis ke piksel [] . Perubahan selanjutnya pada jendela tampilan tidak akan tercermin dalam piksel sampai loadPixels () dipanggil lagi.

Penyaji tertentu mungkin atau mungkin tidak membutuhkan loadPixels () atau updatePixels () . Namun, aturannya adalah setiap kali Anda ingin memanipulasi array piksel [] , Anda harus sebelumnya memanggil loadPixels () , dan setelah perubahan dibuat, panggil updatePixels (). Bahkan jika penyaji sepertinya tidak menggunakan fungsi ini dalam rilis Pemrosesan saat ini, ini akan selalu dapat berubah.

Sintak

loadPixels ()

pixels[]

Examples

```
color pink = color(255, 102, 204);
loadPixels();
for (int i = 0; i < (width*height/2)-width/2;
i++) {
pixels[i] = pink;
}
updatePixels();
```

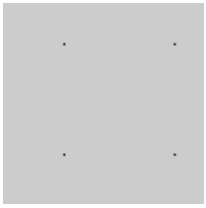
Diskripsi

Array berisi nilai untuk semua piksel di jendela tampilan. Nilai-nilai ini dari tipe data warna. Array ini adalah ukuran jendela tampilan. Misalnya, jika gambar adalah 100x100 piksel, akan ada nilai 10.000 dan jika jendela adalah 200x300 piksel, akan ada nilai 60000.

Sebelum mengakses array ini, data harus dimuat dengan fungsi loadPixels () . Kegagalan untuk melakukannya dapat menghasilkan NullPointerException. Perubahan selanjutnya pada jendela tampilan tidak akan tercermin dalam piksel sampai loadPixels () dipanggil lagi. Setelah piksel diubah, fungsi updatePixels () harus dijalankan untuk memperbarui konten jendela tampilan.

set()

Examples



```
color black = color (0);
set (30, 20, black);
set (85, 20, black);
set (85, 75, black);
set (30, 75, black);
```

contoh foto

```
for (int i = 30; i <width-15; i ++) {
  for (int j = 20; j <height-25; j ++) {
color c = color (204-j, 153-i, 0);
  set (i, j, c);
  }
}
```

contoh foto

```
ukuran (100, 100);
PImage myImage = loadImage ("apples.jpg");
set (0, 0, myImage);
garis (0, 0, lebar, tinggi);
garis (0, tinggi, lebar, 0);
```

Diskripsi

Mengubah warna piksel apa pun, atau menulis gambar langsung ke jendela tampilan.

The x dan y parameter menentukan pixel untuk mengubah dan c parameter menentukan nilai warna. The c parameter ditafsirkan sesuai dengan mode warna saat ini. (Mode warna default adalah nilai RGB dari 0 hingga 255.) Saat mengatur gambar, parameter x dan y menentukan koordinat untuk sudut kiri atas gambar, terlepas dari imageMode saat ini () .

Mengatur warna piksel tunggal dengan set (x, y) itu mudah, tetapi tidak secepat memasukkan data langsung ke piksel []. Pernyataan ekivalen untuk menetapkan (x, y, # 000000) menggunakan piksel [] adalah piksel [y * lebar + x] = # 000000 . Lihat referensi untuk piksel [] untuk informasi lebih lanjut.

Sintak

set (x, y, c)
set (x, y, img)

Parameter-parameter

x	int: koordinat x piksel
y	int: koordinat-y dari piksel
c	int: nilai apa pun dari datatype warna
img	PImage: gambar untuk disalin ke gambar asli

updatePixels()

Examples

```
PImage img = loadImage("rockies.jpg");
image(img, 0, 0);
int halfImage = img.width * img.height/2;
loadPixels();
for (int i = 0; i < halfImage; i++) {
  pixels[i+halfImage] = pixels[i];
}
updatePixels();
```

Diskripsi

Memperbarui jendela tampilan dengan data dalam larik piksel [] . Gunakan bersama dengan loadPixels () . Jika Anda hanya membaca piksel dari array, tidak perlu memanggil updatePixels () - memperbarui hanya diperlukan untuk menerapkan perubahan.

Penyaji tertentu mungkin atau mungkin tidak membutuhkan loadPixels () atau updatePixels () . Namun, aturannya adalah kapan pun Anda ingin memanipulasi array piksel [] , Anda harus memanggil loadPixels () terlebih dahulu , dan setelah perubahan dibuat, panggil updatePixels () . Bahkan jika penyaji sepertinya tidak menggunakan fungsi ini dalam rilis Pemrosesan saat ini, ini akan selalu dapat berubah.

Saat ini, sementara tidak ada penyaji menggunakan parameter tambahan untuk memperbaruiPixels () , ini dapat diimplementasikan di masa depan.

Sintak

updatePixels ()

BAB XXV
RENDERING

blendMode()

Examples

```
size(100, 100);
background(0);
blendMode(ADD);
stroke(102);
strokeWeight(30);
line(25, 25, 75, 75);
line(75, 25, 25, 75);
size(100, 100, P2D);
blendMode(MULTIPLY);
stroke(51);
strokeWeight(30);
line(25, 25, 75, 75);
line(75, 25, 25, 75);
```

Diskripsi

Memadukan piksel di jendela tampilan sesuai dengan mode yang ditentukan. Ada pilihan mode berikut untuk memadukan piksel sumber (A) dengan piksel yang sudah ada di jendela tampilan (B). Warna akhir piksel adalah hasil penerapan salah satu mode campuran di atas dengan setiap saluran (A) dan (B) secara independen. Misalnya, merah dibandingkan dengan merah, hijau dengan hijau, dan biru dengan biru.

BLEND - interpolasi linier warna: C = A * factor + B. Ini adalah mode blending default.

ADD - campuran aditif dengan klip putih: C = min (A * factor + B, 255)

SUBTRACT - campuran subtraktif dengan klip hitam: C = maks (faktor B - A *, 0)

GELAS - hanya warna paling gelap yang berhasil: C = min (faktor A *, B)

TERANG - hanya warna paling terang yang berhasil: C = maks (faktor *, B)
PERBEDAAN - kurangi warna dari gambar yang mendasarinya.
PENGECUALIAN - mirip dengan PERBEDAAN, tetapi kurang ekstrim.
LEBIH BANYAK - gandakan warna, hasilnya akan selalu lebih gelap.
LAYAR - multiply berlawanan, menggunakan nilai kebalikan dari warna.
REPLACE - piksel sepenuhnya menggantikan yang lain dan tidak menggunakan nilai alfa (transparansi).
Kami merekomendasikan menggunakan fungsi blendMode () dan bukan fungsi blend () sebelumnya . Namun, tidak seperti blend (), fungsi blendMode () tidak mendukung yang berikut: HARD_LIGHT, SOFT_LIGHT, OVERLAY, DODGE, BURN. Pada perangkat keras yang lebih lama, mode LIGHTEST, DARKEST, dan DIFERENSI mungkin tidak tersedia juga.

Sintak
blendMode (mode)

Parameter-parameter
Mode int: blending mode untuk digunakan

clip()

Examples
```
void setup() {
  size(200, 200);
  imageMode(CENTER);
}
void draw() {
```

```
  background(204);
  if (mousePressed) {
    clip(mouseX, mouseY, 100, 100);
  } else {
    noClip();
  }
  line(0, 0, width, height);
  line(0, height, width, 0);
}
```

Diskripsi

Membatasi rendering pada batas-batas persegi panjang yang ditentukan oleh parameter. Batas digambar berdasarkan keadaan fungsi imageMode () , baik CORNER, CORNERS, atau CENTER.

Sintak

clip (a, b, c, d)

Parameter-parameter

A	float: koordinat x persegi panjang, secara default
B	float: koordinat-y dari persegi panjang, secara default
C	float: lebar persegi panjang, secara default
D	float: tinggi persegi panjang, secara default

createGraphics ()

Examples

```
PGraphics pg;

void setup() {
  size(200, 200);
```

```
  pg = createGraphics(100, 100);
}

void draw() {
  pg.beginDraw();
  pg.background(102);
  pg.stroke(255);
  pg.line(pg.width*0.5, pg.height*0.5, mouseX, mouseY);
  pg.endDraw();
  image(pg, 50, 50);
}
```

Diskripsi

Membuat dan mengembalikan objek PGraphics baru . Gunakan kelas ini jika Anda perlu menggambar ke buffer grafis di luar layar. Dua parameter pertama menentukan lebar dan tinggi dalam piksel. Parameter ketiga, opsional menentukan penyaji. Ini dapat didefinisikan sebagai P2D, P3D, PDF, atau SVG. Jika parameter ketiga tidak digunakan, renderer default diatur. Penyaji PDF dan SVG memerlukan parameter nama file. Sangat penting untuk mempertimbangkan renderer yang digunakan dengan createGraphics () sehubungan dengan renderer utama yang ditentukan dalam ukuran () . Misalnya, hanya mungkin menggunakan P2D atau P3D dengan createGraphics () ketika salah satu dari mereka didefinisikan dalam ukuran (). Tidak seperti Memproses 1.0, P2D dan P3D menggunakan OpenGL untuk menggambar, dan ketika menggunakan renderer OpenGL, permukaan gambar utama harus berbasis OpenGL. Jika P2D atau P3D digunakan sebagai renderer dalam ukuran () , maka salah satu opsi dapat digunakan dengan createGraphics () . Jika renderer default digunakan dalam ukuran () , maka hanya default, PDF, atau SVG yang dapat digunakan dengan createGraphics () .

Sangat penting untuk memanggil fungsi menggambar antara pernyataan beginDraw () dan endDraw () . Ini juga berlaku untuk semua fungsi yang mempengaruhi gambar, seperti smooth () atau colorMode () .
The createGraphics ()fungsi seharusnya hampir tidak pernah digunakan di dalam draw () karena memori dan waktu yang diperlukan untuk mengatur grafik. Penggunaan satu kali atau sesekali selama draw () dapat diterima, tetapi kode yang memanggil createGraphics () pada 60 frame per detik akan kehabisan memori atau membekukan sketsa Anda.
Berbeda dengan permukaan gambar utama yang benar-benar buram, permukaan yang dibuat dengan createGraphics () dapat memiliki transparansi. Ini memungkinkan untuk menggambar ke dalam grafik dan memelihara saluran alpha. Dengan menggunakan save () untuk menulis file PNG atau TGA, transparansi objek grafis akan dihormati.

Sintak
createGraphics (w, h)
CreateGraphics (w, h, renderer)
CreateGraphics (w, h, renderer, path)

Parameter-parameter
w int: width in pixels
h int: height in pixels
renderer String: Either P2D, P3D, or PDF
path String: the name of the file (can be an absolute or relative path)

noClip()

Examples

```
void setup() {
  size(200, 200);
  imageMode(CENTER);
}

void draw() {
  background(204);
  if (mousePressed) {
    clip(mouseX, mouseY, 100, 100);
  } else {
    noClip();
  }
  line(0, 0, width, height);
  line(0, height, width, 0);
}
```

Diskripsi

Menonaktifkan kliping yang sebelumnya dimulai oleh fungsi clip ().

Sintak

noClip ()

PGraphics

Examples

```
PGraphics pg;

void setup() {
  size(100, 100);
```

```
  pg = createGraphics(40, 40);
}

void draw() {
  pg.beginDraw();
  pg.background(100);
  pg.stroke(255);
  pg.line(20, 20, mouseX, mouseY);
  pg.endDraw();
  image(pg, 9, 30);
  image(pg, 51, 30);
}
```

Diskripsi
Grafik utama dan konteks rendering, serta implementasi API dasar untuk memproses "inti". Gunakan kelas ini jika Anda perlu menggambar ke buffer grafis di luar layar. Objek PGraphics dapat dibangun dengan fungsi createGraphics (). Metode beginDraw () dan endDraw () (lihat Examples di atas) diperlukan untuk mengatur buffer dan menyelesaikannya.
Untuk membuat konteks grafik baru, gunakan fungsi createGraphics () . Jangan gunakan sintaks PGraphics baru () .

Shaders

loadShader()

Examples

```
PShader blur;

void setup() {
```

```
  size(640, 360, P2D);
  // Shaders files must be in the "data" folder to load correctly
  blur = loadShader("blur.glsl");
  stroke(0, 102, 153);
  rectMode(CENTER);
}

void draw() {
  filter(blur);
  rect(mouseX-75, mouseY, 150, 150);
  ellipse(mouseX+75, mouseY, 150, 150);
}
```

Diskripsi

Memuat shader ke objek PShader. File shader harus dimuat di folder / direktori "data" sketsa agar dimuat dengan benar. Shader kompatibel dengan renderer P2D dan P3D, tetapi tidak dengan renderer default.
Atau, file mungkin dimuat dari mana saja di komputer lokal menggunakan jalur absolut (sesuatu yang dimulai dengan / pada Unix dan Linux, atau huruf drive pada Windows), atau parameter nama file dapat menjadi URL untuk file yang ditemukan pada file jaringan.
Jika file tidak tersedia atau kesalahan terjadi, null akan dikembalikan dan pesan kesalahan akan dicetak ke konsol. Pesan kesalahan tidak menghentikan program, namun nilai nol dapat menyebabkan NullPointerException jika kode Anda tidak memeriksa apakah nilai yang dikembalikan adalah nol.

Sintak
loadShader (fragFilename)
loadShader (fragFilename, vertFilename)

Parameter-parameter
fragFilename String: name of fragment shader file
vertFilename String: name of vertex shader file

PShader

Examples

```
PShader blur;

void setup() {
  size(640, 360, P2D);
  // Shaders files must be in the "data" folder to load correctly
  blur = loadShader("blur.glsl");
  stroke(0, 102, 153);
  rectMode(CENTER);
}

void draw() {
  filter(blur);
  rect(mouseX-75, mouseY, 150, 150);
  ellipse(mouseX+75, mouseY, 150, 150);
}
```

Diskripsi
Kelas ini merangkum program shader GLSL, termasuk vertex dan shader fragmen. Ini kompatibel dengan renderer P2D dan P3D, tetapi tidak dengan renderer default. Gunakan fungsi loadShader () untuk memuat kode shader Anda. [Catatan: Sangat disarankan untuk menggunakan loadShader () untuk membuat objek PShader, daripada memanggil konstruktor PShader secara manual.]

resetShader()

Examples

```
PShader edges;
PImage img;

void setup() {
  size(640, 360, P2D);
  img = loadImage("leaves.jpg");
  edges = loadShader("edges.glsl");
}

void draw() {
  shader(edges);
  image(img, 0, 0);
  resetShader();
  image(img, width/2, 0);
}
```

Diskripsi

Kembalikan shader default. Kode yang berjalan setelah resetShader () tidak akan terpengaruh oleh shader yang didefinisikan sebelumnya.

Sintak

resetShader ()
resetShader (kind)

Parameter-parameter

jenis int: jenis shader, baik POIN, LINES, atau TRIANGLES

shader ()

Examples

```
PShader edges;
PImage img;

void setup() {
  size(640, 360, P2D);
  img = loadImage("leaves.jpg");
  edges = loadShader("edges.glsl");
}

void draw() {
  shader(edges);
  image(img, 0, 0);
}
```

Diskripsi

Menerapkan shader yang ditentukan oleh parameter. Ini kompatibel dengan renderer P2D dan P3D, tetapi tidak dengan renderer default.

Sintak

shader(shader)
shader(shader, kind)

Parameter-parameter

shader PShader: name of shader file
kind int: type of shader, either POINTS, LINES, or TRIANGLES

BAB XXVI
Typography

PFont

Examples


```
PFont font;
// The font must be located in the sketch's
// "data" directory to load successfully
font = createFont("LetterGothicStd.ttf", 32);
textFont(font);
text("word", 10, 50);
```

Diskripsi

PFont adalah kelas font untuk Pemrosesan. Untuk membuat font untuk digunakan dengan Memproses, pilih "Buat Font ..." dari menu Tools. Ini akan membuat font dalam format yang dibutuhkan Pemrosesan dan juga menambahkannya ke direktori data sketsa saat ini. Memproses menampilkan font menggunakan format font .vlw, yang menggunakan gambar untuk setiap huruf, daripada mendefinisikannya melalui data vektor. Fungsi loadFont () membuat font baru dan textFont () membuat font aktif. Metode list () membuat daftar font yang diinstal pada komputer, yang merupakan informasi berguna untuk digunakan dengan fungsi createFont () untuk secara dinamis mengubah font menjadi format untuk digunakan dengan Memproses.

Untuk membuat font baru secara dinamis, gunakanfungsi createFont () . Jangan gunakan sintaks PFont baru () .

Loading & Displaying

createFont()

Examples

```
PFont myFont;

void setup() {
  size(200, 200);
  // Uncomment the following two lines to see the available fonts
  //String[] fontList = PFont.list();
  //printArray(fontList);
  myFont = createFont("Georgia", 32);
  textFont(myFont);
  textAlign(CENTER, CENTER);
  text("!@#$%", width/2, height/2);
}
```

Diskripsi

Konversi font secara dinamis ke format yang digunakan oleh Memproses dari file. ttf atau .otf di dalam folder "data" sketsa atau font yang diinstal di tempat lain di komputer. Jika Anda ingin menggunakan font yang diinstal di komputer Anda, gunakan metode PFont.list () untuk menentukan nama font yang dikenali oleh komputer dan kompatibel dengan fungsi ini. Tidak semua font dapat digunakan dan beberapa mungkin bekerja dengan satu sistem operasi dan bukan yang lain. Saat membagikan sketsa dengan orang lain atau mempostingnya di web, Anda mungkin perlu menyertakan versi .ttf atau .otf dari font Anda di direktori data sketsa karena orang lain mungkin tidak memasang font di komputer mereka. Hanya font yang

dapat didistribusikan secara legal yang harus disertakan dengan sketsa.
The ukuranparameter menyatakan ukuran font yang ingin Anda hasilkan. The mulus parameter menentukan jika font harus antialiased atau tidak. The charset parameter array dari karakter yang menentukan karakter untuk menghasilkan.
Fungsi ini memungkinkan Pemrosesan untuk bekerja dengan font secara asli di renderer default, sehingga huruf-hurufnya didefinisikan oleh geometri vektor dan diterjemahkan dengan cepat. Dalam renderers P2D dan P3D , fungsi mengatur proyek untuk membuat font sebagai serangkaian tekstur kecil. Misalnya, ketika menggunakan renderer default, versi asli font yang sebenarnya akan digunakan oleh sketsa, meningkatkan kualitas gambar dan kinerja. Dengan P2D dan P3D render, versi bitmap akan digunakan untuk meningkatkan kecepatan dan penampilan, tetapi hasilnya buruk ketika mengekspor jika sketsa tidak menyertakan file .otf atau .ttf, dan font yang diminta tidak tersedia pada mesin yang menjalankan sketsa.

Sintak
createFont (name, size)
createFont (name, size, smooth)
createFont (name, size, smooth, charset)

Parameter-parameter

Name String	:	name of the font to load
Size float	:	point size of the font
Smooth Boolean	:	true for an antialiased font, false for aliased
Charset char[]	:	array containing characters to be generated

loadFont()

Examples


```
Font PFont;
// Font harus terletak di sketsa
// direktori "data" untuk memuat dengan sukses
font = loadFont ("LetterGothicStd-32.vlw");
textFont (font, 32);
teks ("kata", 10, 50);
```

Diskripsi

Memuat font berformat .vlw ke objek PFont . Buat font .vlw dengan memilih "Buat Font ..." dari menu Tools. Alat ini membuat tekstur untuk setiap karakter alfanumerik dan kemudian menambahkannya sebagai file .vlw ke folder data sketsa saat ini. Karena huruf didefinisikan sebagai tekstur (dan bukan data vektor) ukuran di mana font dibuat harus dipertimbangkan dalam kaitannya dengan ukuran di mana mereka dibuat. Misalnya, muat font 32pt jika sketsa menampilkan font berukuran 32 piksel atau lebih kecil. Sebaliknya, jika font 12pt dimuat dan ditampilkan pada 48pts, huruf-huruf akan terdistorsi karena program akan meregangkan grafik kecil ke ukuran besar.

Seperti loadImage () dan fungsi lain yang memuat data, loadFont ()fungsi tidak boleh digunakan di dalam draw () , karena itu akan sangat memperlambat sketsa, karena font akan dimuat ulang dari disk (atau jaringan) pada setiap frame. Dianjurkan untuk memuat file di dalam pengaturan ()

Untuk memuat dengan benar, font harus berada di folder "data" sketsa saat ini. Atau, file mungkin dimuat dari mana saja di komputer lokal menggunakan jalur absolut (sesuatu yang dimulai dengan / pada Unix dan Linux, atau huruf drive pada Windows), atau parameter nama file dapat menjadi URL untuk file yang ditemukan pada file jaringan.
Jika file tidak tersedia atau kesalahan terjadi, nullakan dikembalikan dan pesan kesalahan akan dicetak ke konsol. Pesan kesalahan tidak menghentikan program, namun nilai nol dapat menyebabkan NullPointerException jika kode Anda tidak memeriksa apakah nilai yang dikembalikan adalah nol.
Gunakan createFont () (bukan loadFont ()) untuk mengaktifkan data vektor untuk digunakan dengan pengaturan renderer default. Ini bisa membantu ketika banyak ukuran font dibutuhkan, atau saat menggunakan renderer apa pun berdasarkan renderer default, seperti perpustakaan PDF.

Sintak
loadFont (filename)

Parameter-parameter
Filename String: nama font untuk memuat

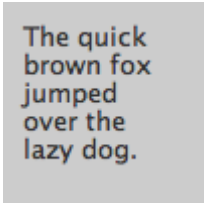
text()

Examples

Contohnya
contoh foto
textSize (32);
teks ("kata", 10, 30);
isi (0, 102, 153);
teks ("kata", 10, 60);


```
isi (0, 102, 153, 51);
teks ("kata", 10, 90);
```

contoh foto




```
size (100, 100, P3D);
textSize (32);
fill (0, 102, 153, 204);
teks ("kata", 12, 45, -30); // Tentukan nilai sumbu-z
teks ("kata", 12, 60); // Kedalaman default, tidak ada nilai z yang ditentukan
contoh foto
String s = "Rubah cokelat cepat melompati anjing malas.";
fill (50);
teks (s, 10, 10, 70, 80); // Teks membungkus dalam kotak teks
```

Diskripsi

Menarik teks ke layar. Menampilkan informasi yang ditentukan dalam parameter pertama pada layar pada posisi yang ditentukan oleh parameter tambahan. Font default akan digunakan kecuali font diatur dengan fungsi textFont () dan ukuran default akan digunakan kecuali font diatur dengan textSize () . Ubah warna teks dengan fungsi fill () . Teks ditampilkan dalam kaitannya dengan fungsi textAlign () , yang memberikan opsi untuk menggambar ke kiri, kanan, dan tengah koordinat.

The x2 dan y2parameter menentukan area persegi panjang untuk ditampilkan di dalam dan hanya dapat digunakan dengan data string. Ketika parameter ini ditentukan, mereka

ditafsirkan berdasarkan pengaturan rectMode () saat ini . Teks yang tidak sepenuhnya pas dalam persegi panjang yang ditentukan tidak akan ditarik ke layar.

Perhatikan bahwa Memproses sekarang memungkinkan Anda memanggil teks () tanpa terlebih dahulu menentukan PFont dengan textFont () . Dalam hal ini, font sans-serif generik akan digunakan sebagai gantinya. (Lihat contoh ketiga di atas.)

Sintak

text(c, x, y)
text(c, x, y, z)
text(str, x, y)
text(chars, start, stop, x, y)
text(str, x, y, z)
text(chars, start, stop, x, y, z)
text(str, x1, y1, x2, y2)
text(num, x, y)
text(num, x, y, z)

Parameter-parameter

C char	:	karakter alfanumerik yang akan ditampilkan
X float	:	koordinat x teks
Y float	:	koordinat-y teks
Z float	:	z-koordinat teks
Char char []	:	simbol alfanumerik yang akan ditampilkan
Start int	:	array index untuk mulai menulis karakter
Stop int	:	array index untuk menghentikan penulisan karakter
x1 float	:	secara default, koordinat x teks, lihat rectMode () untuk info lebih lanjut

y1 float	:	secara default, koordinat y teks, lihat rectMode () untuk info lebih lanjut
x2 float	:	secara default, lebar kotak teks, lihat rectMode () untuk info lebih lanjut
y2 float	:	secara default, ketinggian kotak teks, lihat rectMode () untuk info lebih lanjut
num int, atau float	:	nilai numerik yang akan ditampilkan

textFont()

Examples


```
PFont mono;
// Fon "andalemo.ttf" harus berada di
// direktori "data" sketsa saat ini untuk memuat dengan sukses
mono = loadFont ("andalemo.ttf", 32);
background (0);
textFont (mono);
teks ("kata", 12, 60);
```

Diskripsi

Mengatur font saat ini yang akan digambar dengan fungsi text () . Font harus dibuat untuk Diproses dengan createFont () atau dimuat dengan loadFont () sebelum dapat digunakan. Font yang diatur melalui textFont () akan digunakan dalam semua panggilan selanjutnya ke fungsi text () . Jika tidak ada parameter ukuran yang ditentukan, ukuran font default ke ukuran asli (ukuran di mana ia dibuat dengan alat "Buat Font ...") mengesampingkan panggilan sebelumnya ke textFont () atau textSize () .

Ketika font dirender sebagai tekstur gambar (seperti halnya dengan penyaji P2D dan P3D serta dengan loadFont ()dan vlw

file), Anda harus membuat font dengan ukuran yang paling sering digunakan. Menggunakan textFont () tanpa parameter ukuran akan menghasilkan tipe terbersih.

Sintak
textFont (which)
textFont (which, size)

Parameter-parameter

Which	variabel apa pun dari tipe PFont
Size	ukuran huruf dalam satuan piksel

Atribut

textAlign ()

Examples
variabel apa pun dari tipe PFont


```
background (0);
textSize (16);
textAlign (KANAN);
teks ("ABCD", 50, 30);
textAlign (PUSAT);
teks ("EFGH", 50, 50);
textAlign (KIRI);
teks ("IJKL", 50, 70);
```


```
background (0);
stroke (153);
textSize (11);
textAlign (CENTER, BOTTOM);
```

```
line (0, 30, lebar, 30);
teks ("CENTER, BOTTOM", 50, 30);
textAlign (PUSAT, PUSAT);
line (0, 50, lebar, 50);
teks ("CENTER, CENTER", 50, 50);
textAlign (PUSAT, TOP);
line (0, 70, lebar, 70);
teks("CENTER,TOP",50,70);
```

Diskripsi

Setel perataan saat ini untuk menggambar teks. Parameter LEFT, CENTER, dan RIGHT mengatur karakteristik tampilan huruf terkait dengan nilai untuk parameter x dan y fungsi text () .

Parameter kedua opsional dapat digunakan untuk meluruskan teks secara vertikal. BASELINE adalah default, dan perataan vertikal akan diatur ulang ke BASELINE jika parameter kedua tidak digunakan. Parameter TOP dan CENTER sangat mudah. Parameter BOTTOM mengimbangi garis berdasarkan pada textDescent saat ini () . Untuk beberapa baris, baris terakhir akan disejajarkan ke bawah, dengan garis sebelumnya muncul di atasnya.

Saat menggunakan teks ()dengan parameter lebar dan tinggi, BASELINE diabaikan, dan diperlakukan sebagai TOP. (Kalau tidak, teks akan secara default menggambar di luar kotak, karena BASELINE adalah pengaturan default. BASELINE bukan mode gambar yang berguna untuk teks yang ditarik dalam persegi panjang.)

Penyelarasan vertikal didasarkan pada nilai textAscent () , yang banyak font jangan menentukan dengan benar. Mungkin perlu menggunakan retasan dan offset dengan beberapa piksel dengan tangan agar offset terlihat benar. Untuk melakukan ini sebagai sedikit peretasan, gunakan beberapa persentase

textAscent () atau textDescent () sehingga peretasan bekerja bahkan jika Anda mengubah ukuran font.

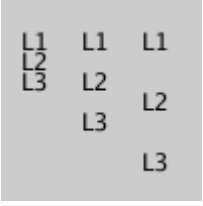
Sintak
textAlign (alignX)
textAlign(alignX,alignY)

Parameter-parameter

alignX int	:	perataan horizontal, baik KIRI, PUSAT, atau KANAN
alignY int	:	penyelarasan vertikal, baik TOP, BOTTOM, CENTER, atau BASELINE

textLeading()

Examples




```
// Teks untuk ditampilkan. "\ N" adalah karakter "baris baru"
line string = "L1 \ nL2 \ nL3";
textSize (12);
fill (0); // Atur isi menjadi hitam

textLeading (10); // Atur mengarah ke 10
teks (line, 10, 25);

textLeading (20); // Atur mengarah ke 20
teks (line, 40, 25);

textLeading (30); // Atur mengarah ke 30
teks (line, 70, 25);
```

Diskripsi
Mengatur jarak antar baris teks dalam satuan piksel. Pengaturan ini akan digunakan dalam semua panggilan berikutnya ke fungsi text () . Perhatikan, bagaimanapun, bahwa pimpinan diatur ulang oleh textSize () . Sebagai contoh, jika memimpin diatur ke 20 dengan textLeading (20) , maka jika textSize (48) dijalankan pada titik berikutnya, yang memimpin akan diatur ulang ke default untuk ukuran teks 48.

Sintak
textLeading (leading)

Parameter-parameter
Leading float : ukuran dalam piksel untuk spasi antar baris

textMode()

Examples

```
import processing.pdf.*;

void setup() {
  size(500, 500, PDF, "TypeDemo.pdf");
  textMode(SHAPE);
  textSize(180);
}

void draw() {
  text("ABC", 75, 350);
  exit();  // keluar program
}
```

Diskripsi

Mengatur cara teks menarik ke layar, baik sebagai peta tekstur atau sebagai geometri vektor. Default textMode (MODEL) , menggunakan tekstur untuk merender font. The TextMode (SHAPE) modus menarik teks menggunakan mesin terbang itu menguraikan karakter individu daripada sebagai tekstur. Mode ini hanya didukung dengan pengaturan renderer PDF dan P3D. Dengan renderer PDF , Anda harus memanggil textMode (SHAPE) sebelum gambar lainnya muncul. Jika outline tidak tersedia, maka textMode (SHAPE) akan diabaikan dan textMode (MODEL) akan digunakan sebagai gantinya.

Opsi textMode (SHAPE) di P3Ddapat dikombinasikan dengan beginRaw () untuk menulis teks yang akurat vektor ke file output 2D dan 3D, misalnya DXF atau PDF . The SHAPE modus saat ini tidak dioptimalkan untuk P3D , jadi jika merekam data bentuk, menggunakan mode teks (MODEL) sampai Anda siap untuk menangkap geometri dengan beginRaw () .

Sintak

textMode (mode)

Parameter-parameter

Mode int : MODEL atau SHAPE

textSize()

Examples


```
 background(0);
fill(255);
textSize(26);
```

```
text("WORD", 10, 50);
textSize(14);
text("WORD", 10, 70);
```

Diskripsi
Setel ukuran font saat ini. Ukuran ini akan digunakan dalam semua panggilan selanjutnya ke fungsi text () . Ukuran font diukur dalam satuan piksel.

Sintak
textSize (size)

Parameter-parameter
Size float: ukuran huruf dalam satuan piksel

textWidth()

Examples


```
textSize(28);

char c = 'T';
float cw = textWidth(c);
text(c, 0, 40);
line(cw, 0, cw, 50);

String s = "Tokyo";
float sw = textWidth(s);
text(s, 0, 85);
line(sw, 50, sw, 100);
```

Diskripsi

Menghitung dan mengembalikan lebar karakter atau string teks apa pun.

Sintak
textWidth (c)
textWidth (str)

Parameter-parameter
c char: karakter untuk diukur
str String: String karakter untuk diukur

textAscent()

Examples


```
float base = tinggi * 0,75;
float scalar = 0,8; // Berbeda untuk setiap font
textSize (32); // Tetapkan ukuran teks awal
float a = textAscent () * skalar; // Kenaikan pendakian
line (0, base-a, lebar, base-a);
teks ("dp", 0, base); // Gambar teks pada baseline
textSize (64); // Tambah ukuran teks
a = textAscent () * skalar; // Kalkulasi ulang pendakian
line (40, base-a, lebar, base-a);
teks ("dp", 40, base); // Gambar teks pada baseline
```

Diskripsi

Mengembalikan pendakian font saat ini pada ukuran saat ini. Informasi ini berguna untuk menentukan ketinggian font di atas garis dasar. Misalnya, menambahkan nilai textAscent () dan textDescent () akan memberi Anda ketinggian total baris.

Sintak
textAscent ()

BAB XXVII
Math

PVector

Examples

```
PVector v1, v2;
Void setup () {
  noLoop ();
  v1 = PVector baru (40, 20);
  v2 = PVector baru (25, 50);
}

void draw () {
  ellipse (v1.x, v1.y, 12, 12);
  ellipse (v2.x, v2.y, 12, 12);
  v2.add (v1);
  ellipse (v2.x, v2.y, 24, 24);
}
```

Diskripsi

Kelas untuk menggambarkan vektor dua atau tiga dimensi, khususnya vektor Euclidean (juga dikenal sebagai geometris). Vektor adalah entitas yang memiliki besaran dan arah. Tipe data, bagaimanapun, menyimpan komponen-komponen dari vektor (x, y untuk 2D, dan x, y, z untuk 3D). Besarnya dan arahnya dapat diakses melalui metode mag () dan heading () .

Dalam banyak contoh Pemrosesan, Anda akan melihat PVectordigunakan untuk menggambarkan posisi, kecepatan, atau percepatan. Misalnya, jika Anda mempertimbangkan sebuah persegi panjang yang bergerak melintasi layar, pada saat tertentu ia memiliki posisi (vektor yang menunjuk dari titik asal ke lokasi), kecepatan (laju perubahan posisi objek per unit waktu, dinyatakan sebagai vektor), dan akselerasi (laju perubahan kecepatan objek per unit waktu, dinyatakan sebagai vektor). Karena vektor mewakili pengelompokan nilai, kita tidak

bisa begitu saja menggunakan penjumlahan / perkalian tradisional / dll. Sebagai gantinya, kita perlu melakukan beberapa "vektor" matematika, yang dipermudah dengan metode di dalam kelas PVector .

Constructor
PVector ()
PVector (x, y, z)
PVector (x, y)

Parameter-parameter
x float: koordinat x.
y float: koordinat y.
z float: koordinat z.

Operators

% (modulo)

Examples
```
int a = 5 % 4;             // Sets 'a' to 1
int b = 125 % 100;         // Sets 'b' to 25
float c = 285.5 % 140.0;  // Sets 'c' to 5.5
float d = 30.0 % 33.0;    // Sets 'd' to 30.0
int a = 0;
void draw() {
  background(200);
  a = (a + 1) % width;  // 'a' increases between 0 and width
  line(a, 0, a, height);
}
```

Diskripsi
Menghitung sisanya ketika satu angka dibagi dengan yang lain. Misalnya, ketika 52.1 dibagi 10, pembagi (10) masuk ke dividen (52.1) lima kali (5 * 10 == 50), dan ada sisa dari 2.1 (52.1 - 50 == 2.1). Dengan demikian, 52,1% 10 menghasilkan 2.1
Perhatikan bahwa ketika pembagi lebih besar dari dividen, sisanya merupakan nilai seluruh dividen. Artinya, angka tidak dapat dibagi dengan angka yang lebih besar dari dirinya sendiri. Misalnya, ketika 9 dibagi 10, hasilnya nol dengan sisa 9. Jadi, 9% 10 menghasilkan 9 .
Modulo sangat berguna untuk memastikan nilai tetap dalam batas, seperti ketika menjaga bentuk di layar. (Lihat contoh kedua di atas.)

Sintak
value1% value2

Parameter-parameter

value 1	int atau float
value 2	int atau float

* (multiply)

Examples

```
int e = 50 * 5;  // Sets 'e' to 250
int f = e * 5;   // Sets 'f' to 1250
```

Diskripsi
Mengalikan nilai dari dua parameter. Perkalian setara dengan

urutan penambahan. Misalnya 5 * 4 setara dengan 5 + 5 + 5 + 5.

Sintak
value1 * value2

Parameter-parameter
Value1int, float, byte, atau char
Value2int, float, byte, atau char

*= (multiply assign)

Examples

```
int a = 5;
int b = 2;
a *= b;  // Sets 'a' to 10
```

Diskripsi
Menggabungkan multiplikasi dengan tugas. Ekspresi a * = b setara dengan a = a * b

Sintak
value1 * = value2

Parameter-parameter
value1 int atau float
value2 nilai numerik apa pun dengan tipe data yang sama dengan value1

+ (addition)

Examples

```
int a = 50 + 5;  // Sets 'a' to 55
int b = a + 5;   // Sets 'b' to 60
String s1 = "Chernenko";
String s2 = "Brezhnev";
String sc1 = s1 + s2;
String sc2 = s1 + ", Andropov," + s2;
println (sc1); // Mencetak "ChernenkoBrezhnev"
println (sc2); // Mencetak "Chernenko, Andropov, Brezhnev"
String s1 = "Gorbachev";
int i = 1987;
String sc1 = s1 + i;
println (sc1); // Mencetak "Gorbachev1987"
```

Diskripsi

Menambahkan dua nilai atau menggabungkan nilai string. Sebagai operator matematika, ia menghitung jumlah dari dua nilai. Sebagai operator string, ini menggabungkan dua string menjadi satu dan mengkonversi dari tipe data primitif ke dalam tipe data String jika perlu.

Sintak

value1 + value2

Parameter-parameter

value1 String, int, float, char, byte, boolean
value2 String, int, float, char, byte, boolean

++ (increment)

Examples
int a = 1; // Atur 'a' ke 1
int b = a ++; // Tetapkan 'b' ke 1, lalu tambahkan 'a' ke 2
int c = a; // Set 'c' ke 2

Diskripsi
Meningkatkan nilai variabel integer sebesar 1. Setara dengan operasi i = i + 1 . Jika nilai variabel i adalah lima, maka ekspresi i ++ meningkatkan nilai i menjadi 6.

Sintak
nilai ++

Parameter-parameter
Nilai int

+= (add assign)

Examples

int a = 50;
int b = 23;
a += b; // Sets 'a' to 73

Diskripsi
Menggabungkan penambahan dengan tugas. Ekspresi a = = b sama dengan a = a + b

Sintak
value1 + = value2

Parameter-parameter
Value1int atau float
Value2nilai numerik apa pun dengan tipe data yang sama dengan value1

- (minus)

Examples
int c = 50 - 5; // Atur 'c' ke 45
int d = c - 5; // Atur 'd' ke 40
int e = d - 60; // Atur 'e' ke -20
int a = 5; // Atur 'a' ke 5
int b = -a; // Set 'b' ke -5
int c = - (5 + 3); // Set 'c' ke -8

Diskripsi
Mengurangi satu nilai dari yang lain dan juga dapat digunakan untuk meniadakan nilai. Sebagai operator pengurangan, nilai parameter kedua dikurangi dari yang pertama. Misalnya, 5 - 3 menghasilkan angka 2. Sebagai operator negasi, itu sama dengan mengalikan angka dengan -1. Misalnya, -5 sama dengan 5 * -1.

Sintak
-value1
value1 - value2

Parameter-parameter
value1 int atau float
value2 int atau float

-- (decrement)

Examples

```
int a = 5;    // Sets 'a' to 5
int b = a--;  // Sets 'b' to 5, then decrements 'a' to 4
int c = a;    // Sets 'c' to 4
```

Diskripsi
Mengurangi nilai variabel integer dengan 1. Setara dengan operasi i = i - 1 . Jika nilai variabel i adalah lima, maka ekspresi i-- menurunkan nilai i menjadi 4.

Sintak
var—

Parameter-parameter

Var	int

-= (subtract assign)

Examples

```
int a = 50;
int b = 23;
a -= b;  // Sets 'a' to 27
```

Diskripsi
Menggabungkan pengurangan dengan penugasan. Ekspresi a -= b setara dengan a = a – b

Sintak
value1 - = value2

Parameter-parameter
value1 int atau float
value2 int atau float

/ (divide)

Examples
int g = 50/5; // Tetapkan 10 untuk 'g'
int h = g / 5; // Tetapkan 2 untuk 'h'

Diskripsi
Membagi nilai parameter pertama dengan nilai parameter kedua. Jawaban untuk persamaan 20/4 adalah 5. Angka 20 adalah jumlah dari empat kejadian dari angka 5. Sebagai persamaan kita melihat bahwa 5 + 5 + 5 + 5 = 20.

Sintak
value1 / value2

Parameter-parameter
value1 int atau float
value2 int atau float, tetapi bukan nol (tidak mungkin bagi dengan nol)

/= (divide assign)

Examples
int a = 12;
int b = 3;
a /= b; // Sets 'a' to 4

Diskripsi
Menggabungkan divisi dengan tugas. Ekspresi a / = b setara dengan a = a / b .

Sintak
value1 / = value2

Parameter-parameter
value1 int atau float
value2 nilai numerik apa pun dengan tipe data yang sama dengan value1

BAB XXVIII
Bitwise Operators

& (bitwise AND)

Examples

```
int a = 207;    // Dalam biner: 11001111
int b = 61;     // Dalam binner: 00111101
int c = a & b; // Dalam binner: 00001101
println(c);     // Mencetak
```

"13", desimal yang setara dengan 0000110

```
color argb = color(204, 204, 51, 255);
// Sytax "& 0xFF" membandingkan biner
// representasi dari dua nilai dan
// buat semuanya kecuali 8 bit terakhir menjadi 0.
// "0xFF" adalah 00000000000000000000000011111111
int a = argb >> 24 & 0xFF;
int r = argb >> 16 & 0xFF;
int g = argb >> 8 & 0xFF;
int b = argb & 0xFF;
fill(r, g, b, a);
rect(30, 20, 55, 55);
```

Diskripsi

Membandingkan masing-masing bit yang sesuai dalam representasi biner dari nilai-nilai. Untuk setiap perbandingan, dua hasil 1, 1, 1 dan 0 menghasilkan 0, dan dua menghasilkan 0, ini mudah dilihat ketika kita melihat representasi biner dari angka

```
  11010110 // 214
& 01011100 // 92
  --------
  01010100 // 84
```

Untuk melihat representasi biner dari suatu angka, gunakan fungsi binary () dengan println ().

Sintak
value & value2

Parameter-parameter

value1	int, char, byte
value2	int, char, byte

<< (left shift)

Examples

```
int m = 1 << 3;   // Dalam biner: 1 to 1000
println(m);  // Cetak "8"
int n = 1 << 8;   // Dalam biner: 1 to 100000000
println(n);  // Cetak "256"
int o = 2 << 3;   // Dalam biner: 10 to 10000
println(o);  // Cetak "16"
int p = 13 << 1;  // Dalam biner: 1101 to 11010
println(p);  // Cetak "26"

// Paket empat nomor 8 bit menjadi satu nomor 32 bit
int a = 255;  // Biner: 00000000000000000000000011111111
int r = 204;  // Biner: 00000000000000000000000011001100
int g = 204;  // Biner: 00000000000000000000000011001100
int b = 51;   // Biner: 00000000000000000000000000110011
a = a << 24;  // Biner:
11111111000000000000000000000000
r = r << 16;  // Biner:
00000000110011000000000000000000
```

```
g = g << 8;   // Biner:
00000000000000001100110000000000

// Setara dengan "color argb = color (r, g, b, a)" tetapi lebih
cepat
color argb = a | r | g | b;
fill(argb);
rect(30, 20, 55, 55);
```

Diskripsi

Menggeser bit ke kiri. Nomor di sebelah kiri operator digeser jumlah tempat yang ditentukan oleh nomor di sebelah kanan. Setiap shift ke kiri menggandakan nomor, oleh karena itu setiap shift kiri mengalikan angka asli dengan 2. Gunakan shift kiri untuk perkalian cepat atau untuk mengemas sekelompok angka bersama menjadi satu nomor yang lebih besar. Pergeseran kiri hanya berfungsi dengan bilangan bulat atau angka yang secara otomatis dikonversi ke bilangan bulat seperti pada byte dan char.

Sintak

value << n

Parameter-parameter

value	int: nilai untuk bergeser
n	int: jumlah tempat untuk bergeser ke kiri

>> (right shift)

Examples

```
int m = 8 >> 3;    // Dalam biner: 1000 to 1
```

```
println(m);  // Cetak "1"
int n = 256 >> 6;  // Dalam biner: 100000000 to 100
println(n);  // Cetak "4"
int o = 16 >> 3;   // Dalam biner: 10000 to 10
println(o);  // Cetak "2"
int p = 26 >> 1;   // Dalam biner: 11010 to 1101
println(p);  // Cetak "13"

// Menggunakan "shift kanan" sebagai teknik yang lebih cepat
daripada merah (), hijau (), dan biru ()
color argb = color(204, 204, 51, 255);
int a = (argb >> 24) & 0xFF;
int r = (argb >> 16) & 0xFF;  // Cara lebih cepat menjadi
merah (argb)
int g = (argb >> 8) & 0xFF;   // Cara lebih cepat untuk
menjadi hijau (argb)
int b = argb & 0xFF;          // Cara yang lebih cepat untuk
menjadi biru (argb)
fill(r, g, b, a);
rect(30, 20, 55, 55);
```

Diskripsi

Menggeser bit ke kanan. Nomor di sebelah kiri operator digeser jumlah tempat yang ditentukan oleh nomor di sebelah kanan. Setiap shift ke kanan membagi dua angka, oleh karena itu setiap shift kanan membagi angka asli dengan 2. Gunakan shift kanan untuk divisi cepat atau untuk mengekstraksi nomor individu dari nomor yang dikemas. Pergeseran kanan hanya berfungsi dengan bilangan bulat atau angka yang secara otomatis dikonversi ke bilangan bulat seperti pada byte dan char.

Pergeseran bit bermanfaat saat menggunakan tipe data warna. Pergeseran kanan dapat mengekstraksi nilai merah, hijau, biru, dan alfa dari suatu warna. Pergeseran kiri dapat digunakan

untuk dengan cepat merakit kembali nilai warna (lebih cepat dari fungsi color ()).

Sintak
value >> n

Parameter-parameter

value	int: nilai untuk bergeser
n	int: jumlah tempat untuk bergeser ke kanan

| (bitwise OR)

Examples

```
int a = 205;   // Dalam biner: 11001101
int b = 45;    // Dalam biner: 00101101
int c = a | b; // Dalam biner: 11101101
println(c);    // Mencetak "237", desimal yang setara dengan
11101101
int a = 255 << 24; // Biner:
11111111000000000000000000000000
int r = 204 << 16; // Biner:
00000000110011000000000000000000
int g = 204 << 8;  // Biner
00000000000000001100110000000000

int b = 51;        // Biner:
00000000000000000000000000110011
// ATAU nilainya bersama:
11111111111001100110000110011
color argb = a | r | g | b;
```

```
fill(argb);
rect(30, 20, 55, 55);
```

Diskripsi
Membandingkan masing-masing bit yang sesuai dalam representasi biner dari nilai-nilai. Untuk setiap perbandingan, dua hasil 1, 1, 1 dan 0 menghasilkan 1, dan dua 0 menghasilkan 0. Ini mudah dilihat ketika kita melihat representasi biner dari angka

```
11010110 // 214
| 01011100 // 92
--------
11011110 // 222
```

Untuk melihat representasi biner dari suatu angka, gunakan fungsi binary () dengan println ().

Sintak
value | value2

Parameter-parameter

value1	int, char, byte
value2	int, char, byte

abs()

Examples

```
int a = abs(153);      // Atur 'a' to 153
```

```
int b = abs(-15);      // Atur 'b' to 15
float c = abs(12.234); // Atur 'c' to 12.234
float d = abs(-9.23);  // Atur 'd' to 9.23
```

Diskripsi
Menghitung nilai absolut (besarnya) dari suatu angka. Nilai absolut dari angka selalu positif.

Sintak
abs(n)

Parameter-parameter
n int, atau float: angka untuk dihitung

ceil()

Examples

```
float x = 8.22;
int a = ceil(x);  // Atur 'a' to 9
```

Diskripsi
Menghitung nilai int terdekat yang lebih besar dari atau sama dengan nilai parameter. Sebagai contoh, ceil (9.03) mengembalikan nilai 10.

Sintak
ceil(n)

Parameter-parameter
n float: angka untuk dibulatkan

constrain()

Examples

```
void draw()
{
  background(204);
  float mx = constrain(mouseX, 30, 70);
  rect(mx-10, 40, 20, 20);
}
```

Diskripsi

Membatasi nilai agar tidak melebihi nilai maksimum dan minimum.

Sintak

constrain(amt, low, high)

Parameter-parameter

amt	int, atau float: nilai untuk membatasi
low	int, atau float: batas minimum
high	int, atau float: batas maksimum

dist()

Examples

```
// Tetapkan nilai abu-abu latar belakang berdasarkan jarak
// dari mouse dari tengah layar
void draw() {
  noStroke();
  float d = dist(width/2, height/2, mouseX, mouseY);
  float maxDist = dist(0, 0, width/2, height/2);
```

```
  float gray = map(d, 0, maxDist, 0, 255);
  fill(gray);
  rect(0, 0, width, height);
}
```

Diskripsi
Menghitung jarak antara dua titik.

Sintak
dist(x1, y1, x2, y2)
dist(x1, y1, z1, x2, y2, z2)

Parameter-parameter

x1	float: koordinat-x dari titik pertama
y1	float: koordinat-y dari titik pertama
z1	float: z-koordinat titik pertama
x2	float: koordinat-x dari titik kedua
y2	float: koordinat-y dari titik kedua
z2	float: z-koordinat titik kedua

Exp()

Examples

```
float v1 = exp(1.0);
println(v1);  // Cetak "2.7182817"
```

Diskripsi
Mengembalikan angka Euler *e* (2,71828 ...) dinaikkan ke kekuatan parameter n.

Sintak
exp(n)

Parameter-parameter
n float: eksponen untuk menaikkan

Floor()

Examples
```
float x = 2.88;
int a = floor(x);  // Atur 'a' to 2
```

Diskripsi
Menghitung nilai int terdekat yang kurang dari atau sama dengan nilai parameter.

Sintak
floor(n)

Parameter-parameter
n float: angka untuk dibulatkan ke bawah

lerp()

Examples


```
float a = 20;
float b = 80;
float c = lerp(a, b, .2);
float d = lerp(a, b, .5);
float e = lerp(a, b, .8);
beginShape(POINTS);
vertex(a, 50);
vertex(b, 50);
```

```
vertex(c, 50);
vertex(d, 50);
vertex(e, 50);
endShape();
```

```
int x1 = 15;
int y1 = 10;
int x2 = 80;
int y2 = 90;
line(x1, y1, x2, y2);
for (int i = 0; i <= 10; i++) {
  float x = lerp(x1, x2, i/10.0) + 10;
  float y = lerp(y1, y2, i/10.0);
  point(x, y);
}
```

Diskripsi

Menghitung angka antara dua angka dengan kenaikan tertentu. Parameter amt adalah jumlah yang diinterpolasi antara dua nilai di mana 0,0 sama dengan titik pertama, 0,1 sangat dekat dengan titik pertama, 0,5 berada di tengah-tengah di antaranya, dll. Fungsi lerp nyaman untuk membuat gerakan di sepanjang jalur lurus dan untuk menggambar garis putus-putus.

Sintak

lerp(start, stop, amt)

Parameter-parameter

start	float: nilai pertama
stop	float: nilai kedua
amt	float: mengambang antara 0,0 dan 1,0

log()

Examples

```
void setup() {
  int i = 12;
  println(log(i));
  println(log10(i));
}

// Menghitung basis-10 logaritma angka
float log10 (int x) {
  return (log(x) / log(10));
}
```

Diskripsi

Menghitung logaritma natural (logaritma base-e) dari suatu bilangan. Fungsi ini mengharapkan parameter n menjadi nilai yang lebih besar dari 0,0.

Sintak

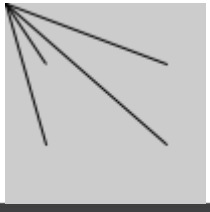
log(n)

Parameter-parameter

n float: angka lebih besar dari 0,0

mag()

Examples



```
float x1 = 20;
float x2 = 80;
float y1 = 30;
```

```
float y2 = 70;

line(0, 0, x1, y1);
println(mag(x1, y1));  // Cetak "36.05551"
line(0, 0, x2, y1);
println(mag(x2, y1));  // Cetak "85.44004"
line(0, 0, x1, y2);
println(mag(x1, y2));  // Cetak "72.8011"
line(0, 0, x2, y2);
println(mag(x2, y2));  // Cetak "106.30146"
```

Diskripsi

Menghitung besarnya (atau panjang) vektor. Vektor adalah arah dalam ruang yang biasa digunakan dalam grafik komputer dan aljabar linier. Karena tidak memiliki posisi "mulai", besarnya vektor dapat dianggap sebagai jarak dari koordinat 0,0 ke nilai x, y. Oleh karena itu, mag () adalah jalan pintas untuk menulis dist (0, 0, x, y).

Sintak

mag(a, b)
mag(a, b, c)

Parameter-parameter

a float: nilai pertama
b float: nilai kedua
c float: nilai ketiga

map()

Examples

```
size(200, 200);
float value = 25;
```

```
float m = map(value, 0, 100, 0, width);
ellipse(m, 200, 10, 10);

float value = 110;
float m = map(value, 0, 100, -20, -10);
println(m);  // Cetak "-9.0"

void setup() {
  size(200, 200);
  noStroke();
}

void draw() {
  background(204);
  float  x1 = map(mouseX, 0, width, 50, 150);
  ellipse(x1, 75, 50, 50);
  float x2 = map(mouseX, 0, width, 0, 200);
  ellipse(x2, 125, 50, 50);
}
```

Diskripsi

Memetakan kembali suatu nomor dari satu rentang ke yang lain.

Pada contoh pertama di atas, angka 25 dikonversi dari nilai dalam kisaran 0 hingga 100 menjadi nilai yang berkisar dari tepi kiri jendela (0) ke tepi kanan (lebar).

Seperti ditunjukkan dalam contoh kedua, angka di luar rentang tidak dijepit ke nilai parameter minimum dan maksimum, karena nilai di luar kisaran sering disengaja dan berguna.

Sintak

map(value, start1, stop1, start2, stop2)

Parameter-parameter

value	float: nilai yang masuk untuk dikonversi
start1	float: batas bawah dari kisaran nilai saat ini
stop1	float: batas atas dari kisaran nilai saat ini
start2	float: batas bawah kisaran target nilai
stop2	float: batas atas kisaran target nilai

max()

Examples

```
int a = max(5, 9);            // Atur 'a' to 9
int b = max(-4, -12);         // Atur 'b' to -4
float c = max(12.3, 230.24);  // Atur 'c' to 230.24

int[] values = {9,-4,362,21 }; // Buat array int

int d = max(values);       // Atur 'd' to 362
```

Diskripsi

Menentukan nilai terbesar dalam urutan angka, dan kemudian mengembalikan nilai itu. max () menerima dua atau tiga nilai float atau int sebagai parameter, atau array dengan panjang berapa pun.

Sintak
max(a, b)
max(a, b, c)
max(list)

Parameter-parameter
a float, atau int: angka pertama untuk dibandingkan
b float, atau int: angka kedua untuk dibandingkan
c float, atau int: angka ketiga untuk dibandingkan
list float [], atau int []: array angka untuk dibandingkan

min()

Examples
```
int d = min(5, 9);            // Atur 'd' to 5
int e = min(-4, -12);         // Atur 'e' to -12
float f = min(12.3, 230.24);  // Atur 'f' to 12.3

int[] values = { 5, 1, 2, -3 };  // Buat array int
int h = min(values);             // Atur 'h' to -3
```

Diskripsi
Menentukan nilai terkecil dalam urutan angka, dan kemudian mengembalikan nilai itu. min () menerima dua atau tiga nilai float atau int sebagai parameter, atau array dengan panjang berapa pun.

Sintak
min(a, b)
min(a, b, c)
min(list)

Parameter-parameter

a int, or float: first number
b int, or float: second number
c int, or float: third number
list float[], or int[]: array of numbers to compare

norm()

Examples

```
float value = 20;
float n = norm(value, 0, 50);
println(n);  // Cetak "0.4"

float value = -10;
float n = norm(value, 0, 100);
println(n);  // Cetak "-0.1"
```

Diskripsi

Menormalkan angka dari rentang lain menjadi nilai antara 0 dan 1. Identik ke peta (nilai, rendah, tinggi, 0, 1).
Angka di luar rentang tidak dijepit ke 0 dan 1, karena nilai di luar kisaran sering disengaja dan berguna. (Lihat contoh kedua di atas.)

Sintak

norm(value, start, stop)

Parameter-parameter

value float: nilai yang masuk untuk dikonversi
start float: batas bawah dari kisaran nilai saat ini
stop float: batas atas dari kisaran nilai saat ini

pow()

Examples

```
float a = pow( 1, 3);  // Atur 'a' to 1*1*1 = 1
float b = pow( 3, 5);  // Atur 'b' to 3*3*3*3*3 = 243
float c = pow( 3,- 5);  // Atur 'c' to 1 / 3*3*3*3*3 = 1 / 243 = .0041
float d = pow(-3, 5);  // Atur 'd' to -3*-3*-3*-3*-3 = -243
```

Diskripsi

Memfasilitasi ekspresi eksponensial. Fungsi pow () adalah cara yang efisien untuk mengalikan angka sendiri (atau kebalikannya) dalam jumlah besar. Misalnya, pow (3, 5) setara dengan ekspresi 3 * 3 * 3 * 3 * 3 dan pow (3, -5) setara dengan 1/3 * 3 * 3 * 3 * 3 * 3.

Sintak

pow(n, *e*)

Parameter-parameter

n — float: basis ekspresi eksponensial

e — float: kekuatan untuk menaikkan pangkalan

round ()

Examples

```
float x = 9.2;
int rx = round(x);  // Atur 'rx' to 9
```

```
float y = 9.5;
int ry = round(y);  // Atur 'ry' to 10

float z = 9.9;
int rz = round(z);  // Atur 'rz' to 10
```

Diskripsi
Menghitung bilangan bulat terdekat dengan parameter n. Misalnya, putaran (133,8) mengembalikan nilai 134.

Sintak
round(n)

Parameter-parameter
n float: angka untuk dibulatkan

sq()

Examples


```
noStroke();
float a = sq(1);   // Atur 'a' to 1
float b = sq(-5)  ;  // Atur 'b' to 25
float c = sq(9);   // Atur 'c' to 81
rect(0, 25, a, 10);
rect(0, 45, b, 10);
rect(0, 65, c, 10);
```

Diskripsi
Kuadratkan angka (mengalikan angka dengan sendirinya). Hasilnya selalu berupa angka positif, karena mengalikan dua angka negatif selalu menghasilkan hasil yang positif. Misalnya, -1 * -1 = 1.

Sintak
sq(n)

Parameter-parameter

n float: angka ke kotak

sqrt()

Examples


```
noStroke();
float a = sqrt(6561);  // Atur 'a' to 81
float b = sqrt(625);   // Atur 'b' to 25
float c = sqrt(1);     // Atur 'c' to 1
rect(0, 25, a, 10);
rect(0, 45, b, 10);
rect(0, 65, c, 10);
```

Diskripsi
Menghitung akar kuadrat dari angka. Akar kuadrat dari angka selalu positif, meskipun mungkin ada akar negatif yang valid. Akar kuadrat s dari angka a sedemikian rupa sehingga s * s = a. Ini kebalikan dari kuadrat.

Sintak
sqrt(n)

Parameter-parameter
n float: non-negative number

acos()

Examples

```
float a = PI;
float c = cos(a);
float ac = acos(c);
// Cetak "3.1415927 : -1.0 : 3.1415927"
println(a + " : " + c + " : " +  ac);

float a = PI + PI/4.0;
float c = cos(a);
float ac = acos(c);
// Cetak "3.926991 : -0.70710665 : 2.3561943"
println(a + " : " + c + " : " +  ac);
```

Diskripsi

Inversi cos (), mengembalikan arc cosine dari suatu nilai. Fungsi ini mengharapkan nilai dalam kisaran -1 hingga 1 dan nilai dikembalikan dalam kisaran 0 hingga PI (3.1415927).

Sintak

```
acos(value)
```

Parameter

value float: nilai arc cosine yang harus dikembalikan

asin()

Examples

```
float a = PI/3;
float s = sin(a);
```

```
float as = asin(s);
// Cetak "1.0471976 : 0.86602545 : 1.0471976"
println(a + " : " + s + " : " +  as);

float a = PI + PI/3.0;
float s = sin(a);
float as = asin(s);
// Cetak "4.1887903 : -0.86602545 : -1.0471976"
println(a + " : " + s + " : " +  as);
```

Diskripsi

Invers of sin (), mengembalikan arc dari nilai. Fungsi ini mengharapkan nilai dalam kisaran -1 hingga 1 dan nilai dikembalikan dalam rentang -PI / 2 hingga PI / 2.

Sintak

asin(value)

Parameter-parameter

value float: float: nilai yang harus dikembalikan arc

atan()

Examples

```
float a = PI/3;
float t = tan(a);
float at = atan(t);
// Cetak "1.0471976 : 1.7320509 : 1.0471976"
println(a + " : " + t + " : " +  at);

float a = PI + PI/3.0;
```

```
float t = tan(a);
float at = atan(t);
// Cetak "4.1887903 : 1.7320513 : 1.0471977"
println(a + " : " + t + " : " +  at);
```

Diskripsi

Invers dari tan (), mengembalikan arc tangent dari suatu nilai. Fungsi ini mengharapkan nilai dalam kisaran -Infinity ke Infinity (eksklusif) dan nilai dikembalikan dalam rentang -PI / 2 ke PI / 2.

Sintak

atan(value)

Parameter-parameter

value float: -Infinity to Infinity (eksklusif)

atan2()

Examples

```
void draw() {
  background(204);
  translate(width/2, height/2);
  float a = atan2(mouseY-height/2, mouseX-width/2);
  rotate(a);
  rect(-30, -5, 60, 10);
}
```

Diskripsi

Menghitung sudut (dalam radian) dari titik yang ditentukan ke titik koordinat yang diukur dari sumbu x positif. Nilai dikembalikan sebagai float dalam kisaran dari PI ke -PI. Fungsi atan2 () paling sering digunakan untuk mengarahkan geometri

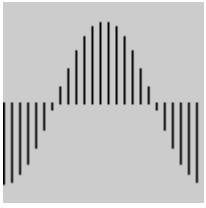
ke posisi kursor. Catatan: Koordinat-y titik adalah parameter pertama, dan koordinat-x adalah parameter kedua, karena struktur penghitungan tangen.

Sintak
atan2(y, x)

Parameter-parameter

y	float: koordinat y titik
x	float: koordinat x titik

cos()



```
float a = 0.0;
float inc = TWO_PI/25.0;
for (int i = 0; i < 25; i++) {
  line(i*4, 50, i*4, 50+cos(a)*40.0);
  a = a + inc;
}
```

Diskripsi
Menghitung kosinus dari suatu sudut. Fungsi ini mengharapkan nilai parameter sudut disediakan dalam radian (nilai dari 0 hingga PI * 2). Nilai dikembalikan dalam kisaran -1 hingga 1.

Sintak
cos(angle)

Parameter-parameter

angle	float: sudut dalam radian

degrees()

Examples

```
float rad = PI/4;
float deg = degrees(rad);
println(rad + " radians is " + deg + " degrees");
```

Diskripsi

Mengonversi pengukuran radian ke nilai terkait dalam derajat. Radian dan derajat adalah dua cara untuk mengukur hal yang sama. Ada 360 derajat dalam satu lingkaran dan 2 * PI radian dalam satu lingkaran. Misalnya, 90 ° = PI / 2 = 1.5707964. Semua fungsi trigonometri dalam Pemrosesan memerlukan parameternya untuk ditentukan dalam radian.

Sintak

degrees(radians)

Parameter-parameter

radians float: nilai radian untuk mengkonversi ke derajat

radians()

Examples

```
float deg = 45.0;
float rad = radians(deg);
println(deg + " degrees is " + rad + " radians");
```

Diskripsi

Mengubah pengukuran derajat ke nilai yang sesuai di radian. Radian dan derajat adalah dua cara untuk mengukur hal yang

sama. Ada 360 derajat dalam satu lingkaran dan 2 * PI radian dalam satu lingkaran. Misalnya, 90 ° = PI / 2 = 1.5707964. Semua fungsi trigonometri dalam Pemrosesan memerlukan parameternya untuk ditentukan dalam radian.

Sintak
radians(degrees)

Parameter-parameter

degrees float: nilai derajat untuk dikonversi ke radian

sin()

Examples

```
float a = 0.0;
float inc = TWO_PI/25.0;

for (int i = 0; i < 100; i=i+4) {
  line(i, 50, i, 50+sin(a)*40.0);
  a = a + inc;
}
```

Diskripsi
Menghitung sinus sudut. Fungsi ini mengharapkan nilai parameter sudut disediakan dalam radian (nilai dari 0 hingga 6.28). Nilai dikembalikan dalam kisaran -1 hingga 1.

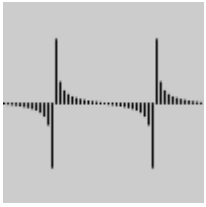
Sintak
sin(angle)

Parameter-parameter

angle float: sudut dalam radian

tan()

Examples



```
float a = 0.0;
float inc = TWO_PI/50.0;

for (int i = 0; i < 100; i = i+2) {
  line(i, 50, i, 50+tan(a)*2.0);
  a = a + inc;
}
```

Diskripsi

Menghitung rasio sinus dan kosinus sudut. Fungsi ini mengharapkan nilai parameter sudut disediakan dalam radian (nilai dari 0 hingga PI * 2). Nilai dikembalikan dalam kisaran tak terhingga hingga-tak terhingga.

Sintak

tan(angle)

Parameter-parameter

angle float: sudut dalam radian

noise()

Examples

```
float xoff = 0.0;

void draw() {
  background(204);
  xoff = xoff + .01;
  float n = noise(xoff) * width;
  line(n, 0, n, height);
}

float noiseScale = 0.02;

void draw() {
  background(0);
  for (int x=0; x < width; x++) {
    float noiseVal = noise((mouseX+x)*noiseScale,
mouseY*noiseScale);
    stroke(noiseVal*255);
    line(x, mouseY+noiseVal*80, x, height);
  }
}
```

Diskripsi

Mengembalikan nilai kebisingan Perlin pada koordinat yang ditentukan. Perlin noise adalah generator urutan acak yang menghasilkan suksesi harmonik angka yang lebih alami daripada fungsi standar random (). Ini dikembangkan oleh Ken Perlin pada 1980-an dan telah digunakan dalam aplikasi grafis untuk menghasilkan tekstur prosedural, bentuk, medan, dan bentuk organik lainnya.

Berbeda dengan fungsi random (), noise Perlin didefinisikan dalam ruang n-dimensi yang tak terbatas, di mana setiap pasangan koordinat sesuai dengan nilai semi-acak tetap (hanya ditetapkan untuk umur program). Nilai yang dihasilkan akan selalu antara 0,0 dan 1,0. Pemrosesan dapat menghitung noise 1D, 2D dan 3D, tergantung pada jumlah koordinat yang diberikan. Nilai noise dapat dianimasikan dengan bergerak melalui ruang noise, seperti yang ditunjukkan pada contoh pertama di atas. Dimensi ke-2 dan ke-3 juga dapat diartikan sebagai waktu.

Struktur noise sebenarnya mirip dengan sinyal audio, sehubungan dengan penggunaan frekuensi fungsi. Mirip dengan konsep harmonik dalam fisika, kebisingan Perlin dihitung selama beberapa oktaf yang ditambahkan bersama untuk hasil akhir.

Cara lain untuk menyesuaikan karakter dari urutan yang dihasilkan adalah skala koordinat input. Karena fungsi ini bekerja di dalam ruang tanpa batas, nilai koordinat tidak menjadi masalah; hanya jarak antara koordinat berturut-turut yang penting (seperti saat menggunakan noise () dalam satu lingkaran). Sebagai aturan umum, semakin kecil perbedaan antara koordinat, semakin halus urutan kebisingan yang dihasilkan. Langkah-langkah 0,005-0,03 bekerja paling baik untuk sebagian besar aplikasi, tetapi ini akan berbeda tergantung pada penggunaan.

Telah ada perdebatan tentang keakuratan implementasi kebisingan dalam Pemrosesan. Untuk klarifikasi, ini merupakan implementasi "klasik Perlin noise" dari tahun 1983, dan bukan metode "simplex noise" yang lebih baru dari tahun 2001.

Sintak

noise(x)
noise(x, y)
noise(x, y, z)

Parameter-parameter

x float: koordinat x dalam ruang kebisingan
y float: koordinat y dalam ruang kebisingan
z float: koordinat z dalam ruang kebisingan

noiseDetail()

Examples

```
float noiseVal;
float noiseScale=0.02;

void draw() {
  for (int y = 0; y < height; y++) {
    for (int x = 0; x < width/2; x++) {
      noiseDetail(3,0.5);
      noiseVal = noise((mouseX+x) * noiseScale, (mouseY+y) *
noiseScale);
      stroke(noiseVal*255);
      point(x,y);
      noiseDetail(8,0.65);
      noiseVal = noise((mouseX + x + width/2) * noiseScale,
                       (mouseY + y) * noiseScale);
      stroke(noiseVal * 255);
      point(x + width/2, y);
    }
  }
}
```

Diskripsi

Menyesuaikan karakter dan tingkat detail yang dihasilkan oleh fungsi noise Perlin. Mirip dengan harmonik dalam fisika,

kebisingan dihitung selama beberapa oktaf. Oktaf yang lebih rendah berkontribusi lebih banyak pada sinyal keluaran dan dengan demikian menentukan intensitas keseluruhan kebisingan, sedangkan oktaf yang lebih tinggi menciptakan detail yang lebih halus dalam urutan kebisingan.
Secara default, noise dihitung lebih dari 4 oktaf dengan masing-masing oktaf menyumbang tepat setengah dari pendahulunya, mulai dari kekuatan 50% untuk oktaf pertama. Jumlah falloff ini dapat diubah dengan menambahkan parameter fungsi tambahan. Misalnya, faktor falloff 0,75 berarti setiap oktaf sekarang akan memiliki dampak 75% (25% lebih sedikit) dari oktaf sebelumnya yang lebih rendah. Meskipun angka antara 0,0 dan 1,0 valid, perhatikan bahwa nilai yang lebih besar dari 0,5 dapat menghasilkan noise () yang mengembalikan nilai lebih dari 1,0.
Dengan mengubah parameter ini, sinyal yang dibuat oleh fungsi noise () dapat disesuaikan agar sesuai dengan kebutuhan dan karakteristik yang sangat spesifik.

Sintak
noiseDetail(lod)
noiseDetail(lod, falloff)

Parameter-parameter

lod	int: jumlah oktaf yang akan digunakan oleh kebisingan
falloff	float: faktor falloff untuk setiap oktaf

noiseSeed()

Examples
float xoff = 0.0;

```
void setup() {
  noiseSeed(0) ;
  stroke(0, 10);
}

void draw() {
  xoff = xoff + .01;
  float n = noise(xoff) * width;
  line(n, 0, n, height);
}
```

Diskripsi
Tetapkan nilai seed untuk noise (). Secara default, noise () menghasilkan hasil yang berbeda setiap kali program dijalankan. Atur parameter seed ke konstanta untuk mengembalikan nomor pseudo-acak yang sama setiap kali perangkat lunak dijalankan.

Sintak
noiseSeed(seed)

Parameter-parameter

seed	int: nilai benih

random()

Examples

```
for (int i = 0; i < 100; i++) {
  float r = random(50);
  stroke(r*5);
  line(50, i, 50+r, i);
}
```

```
for (int i = 0; i < 100; i++) {
  float r = random(-50, 50);
  println(r);
}

// Dapatkan elemen acak dari array
String[] words = { "apple", "bear", "cat", "dog" };
int index = int(random(words.length));  // Same as
int(random(4))
println(words[index]);  // Mencetak satu dari empat kata
```

Diskripsi

Menghasilkan angka acak. Setiap kali fungsi random () dipanggil, ia mengembalikan nilai tak terduga dalam rentang yang ditentukan. Jika hanya satu parameter yang diteruskan ke fungsi, itu akan mengembalikan float antara nol dan nilai parameter tinggi. Misalnya, acak (5) mengembalikan nilai antara 0 dan 5 (mulai dari nol, dan hingga, tetapi tidak termasuk, 5).

Jika dua parameter ditentukan, fungsi akan mengembalikan float dengan nilai di antara kedua nilai. Misalnya, acak (-5, 10.2) mengembalikan nilai mulai dari -5 dan hingga (tetapi tidak termasuk) 10.2. Untuk mengonversi angka acak floating-point ke integer, gunakan fungsi int ().

Sintak

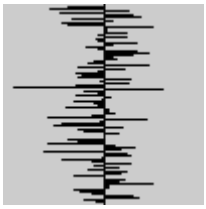
random(high)
random(low, high)

Parameter-parameter

low	float: batas bawah
high	float: batas atas

randomGaussian()

Examples



```
for (int y = 0; y < 100; y++) {
  float x = randomGaussian() * 15;
  line(50, y, 50 + x, y);
}
```

```
float[] distribution = new float[360];
void setup() {
  size(100, 100);
  for (int i = 0; i < distribution.length; i++)
{
    distribution[i] = int(randomGaussian() *
15);
  }
}

void draw() {
  background(204);

  translate(width/2, width/2);

  for (int i = 0; i < distribution.length; i++)
{
    rotate(TWO_PI/distribution.length);
    stroke(0);
    float dist = abs(distribution[i]);
    line(0, 0, dist, 0);
  }
}
```

Diskripsi
Mengembalikan float dari serangkaian angka acak yang memiliki rata-rata 0 dan standar deviasi 1. Setiap kali fungsi randomGaussian () dipanggil, ia mengembalikan angka yang pas dengan distribusi Gaussian, atau normal,. Secara teoritis tidak ada nilai minimum atau maksimum yang dapat dikembalikan randomGaussian (). Sebaliknya, hanya ada probabilitas yang sangat rendah bahwa nilai yang jauh dari nilai rata-rata akan dikembalikan; dan probabilitas yang lebih tinggi bahwa angka-angka di dekat rata-rata akan dikembalikan.

Sintak
randomGaussian()

randomSeed()

Examples
```
randomSeed(0);
for (int i=0; i < 100; i++) {
  float r = random(0, 255);
  stroke(r);
  line(i, 0, i, 100);
}
```

Diskripsi
Tetapkan nilai seed untuk random (). Secara default, random () menghasilkan hasil yang berbeda setiap kali program dijalankan. Atur parameter seed ke konstanta untuk mengembalikan nomor pseudo-acak yang sama setiap kali perangkat lunak dijalankan.

Sintak
randomSeed(seed)

Parameter-parameter
seed int: nilai benih

BAB XXIX
KONSTANTA

HALF_PI

Examples

```
float x = width/2;
float y = height/2;
float d = width * 0.8;
arc(x, y, d, d, 0, QUARTER_PI);
arc(x, y, d-20, d-20, 0, HALF_PI);
arc(x, y, d-40, d-40, 0, PI);
arc(x, y, d-60, d-60, 0, TWO_PI);
```

Diskripsi

HALF_PI adalah konstanta matematika dengan nilai 1.5707964. Ini adalah setengah rasio keliling lingkaran dengan diameternya. Berguna dalam kombinasi dengan fungsi trigonometri sin () dan cos ().

PI

Examples

```
float x = width/2;
float y = height/2;
float d = width * 0.8;
arc(x, y, d, d, 0, QUARTER_PI);
arc(x, y, d-20, d-20, 0, HALF_PI);
arc(x, y, d-40, d-40, 0, PI);
arc(x, y, d-60, d-60, 0, TWO_PI);
```

Diskripsi

PI adalah konstanta matematika dengan nilai 3.1415927. Ini adalah rasio keliling lingkaran dengan diameternya. Berguna dalam kombinasi dengan fungsi trigonometri sin () dan cos ().

QUARTER_PI

Examples

```
float x = width/2;
float y = height/2;
float d = width * 0.8;
arc(x, y, d, d, 0, QUARTER_PI);
arc(x, y, d-20, d-20, 0, HALF_PI);
arc(x, y, d-40, d-40, 0, PI);
arc(x, y, d-60, d-60, 0, TWO_PI);
```

Diskripsi

QUARTER_PI adalah konstanta matematika dengan nilai 0.7853982. Ini adalah seperempat rasio keliling lingkaran dengan diameternya. Berguna dalam kombinasi dengan fungsi trigonometri sin () dan cos ().

TAU

Examples

```
float x = width/2;
float y = height/2;
float d = width * 0.8;
arc(x, y, d, d, 0, QUARTER_PI);
arc(x, y, d-20, d-20, 0, HALF_PI);
arc(x, y, d-40, d-40, 0, PI);
arc(x, y, d-60, d-60, 0, TAU);
```

Diskripsi

TAU adalah konstanta matematika dengan nilai 6.2831855. Ini adalah konstanta lingkaran yang menghubungkan keliling lingkaran dengan dimensi liniernya, rasio keliling lingkaran dengan jari-jarinya. Ini berguna dalam kombinasi dengan fungsi trigonometri seperti sin () dan cos ().

TWO_PI

Examples

```
float x = width/2;
float y = height/2;
float d = width * 0.8;
arc(x, y, d, d, 0, QUARTER_PI);
arc(x, y, d-20, d-20, 0, HALF_PI);
arc(x, y, d-40, d-40, 0, PI);
arc(x, y, d-60, d-60, 0, TWO_PI);
```

Diskripsi

TWO_PI adalah konstanta matematika dengan nilai 6.2831855. Ini adalah dua kali rasio keliling lingkaran dengan diameternya. Berguna dalam kombinasi dengan fungsi trigonometri sin () dan cos ().

DAFTAR PUSTAKA

Andrés Colubri, **Processing for Android: Create Mobile, Sensor-Aware, and VR Applications Using Processing,** Published 2017, Apress

Ben Fry, **Visualizing Data,** Published December 2007, O'Reilly

Casey Reas and Ben Fry, **Make: Getting Started with Processing, Second Edition,** Published September 2015, Maker Media

Casey Reas and Ben Fry, **Processing: A Programming Handbook for Visual Designers, Second Edition,** Published December 2014, The MIT Press

Daniel Shiffman, **Learning Processing, Second Edition: A Beginner's Guide to Programming Images, Animation, and Interaction,** Published August 2015, Morgan Kaufmann

Daniel Shiffman, **The Nature of Code: Simulating Natural Systems with Processing,** Published December 2012. PDF, Web, Paperback

Derek Runberg, **The SparkFun Guide to Processing,** 2015, No Starch Press

Hartmut Bohnacker, Benedikt Gross, Julia Laub, and Claudius Lazzeroni, **Generative Design,** August 2012, Princeton Architectural Press

Ira Greenberg, Dianna Xu, Deepak Kumar, **Processing: Creative Coding and Generative Art in Processing 2,** Published April 2013, friends of ED

Jeffrey L. Nyhoff, Larry R. Nyhoff, **Processing: An Introduction to Programming,** Published May 2017, CRC Press

https://processing.org/books/

TENTANG PENULIS

Slamet Winardi, ST, MT terlahir di kota Semarang 3 Agutus 1971, mengenyam pendidikan terakhir di Sistem Kontrol ITS lulus tahun 2003. Sekarang Menekuni bidang Internet of Things khususnya IoT transportasi. Prestasi terakhir yang dihasilkan 3 publikasi Jurnal nasional dan internasional, 5 buku tentang IoT transportasi, 8 Presenter di tingkat Internasional maupun nasional, Pembicara di Kuliah Tamu UTHM Johor Bahru Malaysia, 15 HaKI tentang IoT transportasi, Publikasi di media masa cetak maupun internet, 2 Paten Merek, dan Sebuah Paten sederhana tentang IoT transportasi. Konsen dibidangnya dengan membentuk komunitas pengembang IoT yang diberi nama Indonesia Internet of Things yang disingkat ID-IoT.

Dr. Sri Wiwoho Mudjanarko, ST., MT, lahir di Surabaya, 24 Juni 1966. Penulis bekerja sebagai Dosen Teknik Sipil di Universitas Narotama dan sebagai Dosen LB S2 T. Sipil di Universitas 17 Agustus (Untag) Surabaya. Penulis menyelesaikan pendidikan Diploma III Teknik Sipil di Universitas Kristen Petra (UK PETRA), Surabaya, Sarjana Teknik Sipil di Universitas Narotama,Surabaya, Magister Teknik Sipil di Institut Teknologi Sepuluh Nopember (ITS) Surabaya, Pendidikan Program Doktor Teknik Sipil di Universitas Brawijaya, Malang, Program Profesi Insinyur Universitas Gadjah Mada (UGM). Penulis menjadi pemenang Hibah Penelitian sejak tahun 2009 hingga saat ini. Selain itu, hasil karya tulis ilmiah penulis banyak yang dimuat di berbagai Proceeding-Jurnal Nasional/Internasional. Penulis sering mengikuti Seminar Nasional, Intenational, Visiting Dan Kuliah Tamu di berbagai Universitas di dalam negeri maupun di -

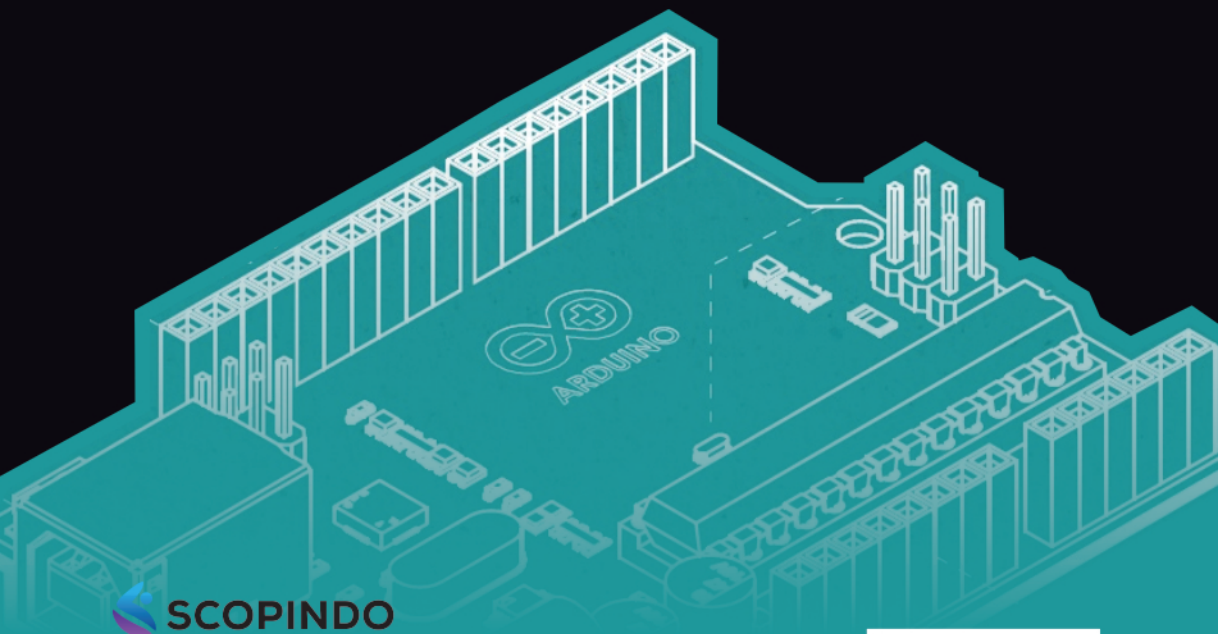
luar negeri. Penulis juga aktif mengikuti Pelatihan di berbagai Universitas.

DESAIN INTERFACE GRAFIS ARDUINO DENGAN BAHASA PEMROGRAMAN PROCESSING

Processing adalah perangkat lunak sketsa yang fleksibel dan mudah untuk dipelajari yaitu cara membuat program dalam konteks seni visual. Sejak tahun 2001, Processing telah mempromosikan literasi perangkat lunak dalam seni visual dan literasi visual dalam teknologi. Ada puluhan ribu siswa, seniman, perancang, peneliti, dan penggemar yang menggunakan Processing untuk belajar dan membuat prototype program aplikasi berbasis grafis.

Processing Development Environment (PDE) membuatnya mudah untuk menulis program dalam Processing. Program ditulis dalam Editor Teks dan untuk menjalankan dengan menekan tombol Run. Dalam Processing, program komputer disebut sketsa. Sketsa disimpan di Sketchbook, yang merupakan folder di komputer Anda.

Pada buku ini disajikan kamus pemrograman Processing yang dapat dimanfaatkan sebagai pegangan saat memulai belajar pemrograman berbasis grafis yang fleksibel dan interaktif. Kamus ini akan memberikan ulasan setiap perintah yang digunakan oleh software Processing mulai dari contoh program, diskripsi, sintak penulisan dan parameter-parameter yang digunakan oeh setiap perintah program.



SCOPINDO
MEDIA PUSTAKA

08813223878
www.scopindo.com
Scopindomedia@gmail.com
Jl. Menanggal III No.45, Surabaya

ISBN 978-623-92343-4-4
9 786239 216344