

# Modul Praktikum Grafika Komputer

## PRAKTIKUM 01

### PENGANTAR GRAFIKA KOMPUTER DAN PENGENALAN PROCESSING 1.0

#### Tujuan Instruksional

Setelah mengikuti praktikum ini, mahasiswa diharapkan dapat untuk :

1. Memahami konsep Grafika Komputer
2. Mengoperasikan perangkat Processing 1.0 untuk pemrograman Grafika Komputer

#### Apakah Grafika Komputer Itu?

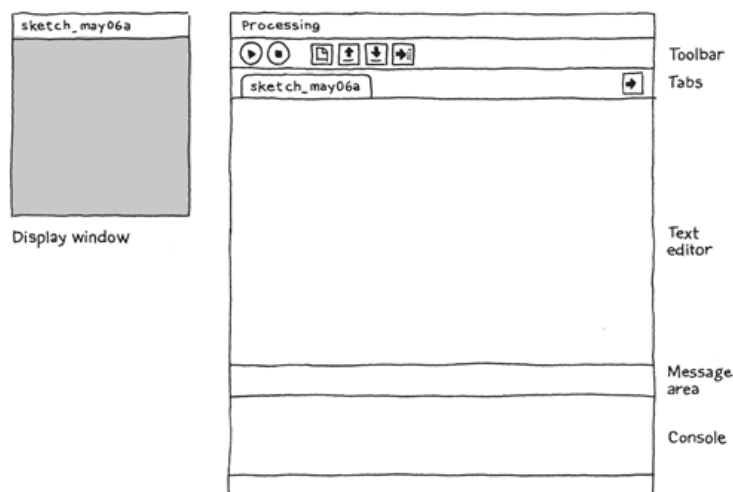
Grafika Komputer (Computer Graphic) adalah seperangkat alat yang terdiri dari hardware dan software untuk memproduksi suatu gambar, grafik atau citra realistic untuk seni, game computer, foto dan film animasi. Grafika Komputer merupakan bagian yang paling sulit di bidang computer, karena selain harus mengerti bahasa dan logika pemrograman juga dibutuhkan kemampuan analisis serta pemahaman matematik.

Banyak cara dan variasi untuk menghasilkan gambar dengan computer.

Seringkali kita akan dihadapkan pada masalah algoritma dan pemrograman untuk menghasilkan citra yang realistic. Hal ini adalah sesuatu yang lumrah, sehingga diperlukan pendekatan sesederhana mungkin dan pemahaman algoritma untuk mendapatkan citra realistic.

#### Processing untuk percobaan Grafika Komputer

Processing merupakan perangkat lunak open source yang menyediakan bahasa pemrograman dan lingkungan pemrograman bagi orang-orang yang ingin membuat program pengolahan citra, animasi dan suara. Processing digunakan dalam berbagai tahap seperti: belajar, membuat prototype sampai pasda tahap produksi. Processing dibuat dengan tujuan untuk memberikan fasilitas belajar pemrograman computer dalam konteks visual. Processing dapat diunduh di <http://www.processing.org>. Ada dua versi processing yang tersedia, yaitu versi tanpa java dan versi dengan java. Dengan sedikit keberuntungan, model Lingkungan kerja Processing adalah seperti ditunjukkan pada gambar 1.1.



Gambar 1.1, Lingkungan kerja Processing 1.0

# Modul Praktikum Grafika Komputer

---

Processing menggunakan istilah Sketch untuk proyek yang ditulis menggunakan processing. Setiap sketch disimpan dalam direktori terpisah dan program utama disimpan di file nama yang sama dengan nama direktori. Sebagai contoh, sketch dengan nama 'veronica' akan disimpan di direktori 'veronica' dengan file program 'veronica.pde'.

Anda sekarang menjalankan Pengolahan Development Environment (atau PDE). Tak banyak untuk itu, wilayah yang besar Editor Teks, dan ada deretan tombol di bagian atas, hal ini toolbar. Editor Berikut adalah Pesan Area, dan di bawah yaitu Console. Pesan Area digunakan untuk satu pesan baris, dan Konsol digunakan untuk rincian lebih teknis.

Toolbar



Gambar 1.2. Toolbar pada Processing

Pada gambar 1.2 ditunjukkan tombol untuk melakukan Running, Stop, New Window, Open, Save dan Export.

Program Pertama Anda

Mempersiapkan layar

Layar yang dimaksud adalah area untuk menampilkan objek. Perintah yang digunakan adalah : `size(width,height)`. Sebagai contoh `size(200,200)`, hal ini berarti ukuran layar adalah 200 x 200.

Pewarnaan Latar

Warna yang digunakan dalam processing adalah penggabungan dari elemen Red, Green dan Blue (RGB) atau warna Grayscale (abu-abu). Untuk memberikan warna latar (background) digunakan perintah `background(x)` atau `background(r,g,b)`. kisaran nilai warna yang digunakan adalah antara 0 sampai dengan 255. sebagai contoh `background(128)`, hal ini berarti warna latar adalah grayscale dengan nilai 128. Jika ditulis `background(255,0,0)` akan mengakibatkan warna latar menjadi merah, karena nilai merah di nilai 255 dan hijau serta biru adalah 0.

Nah sekarang coba anda menuliskan program seperti berikut dan analisa hasilnya.

Program 1.1

```
// Sets the screen to be 200, 200, so the width of the window is 200 pixels
// and the height of the window is 200 pixels
size(200, 200);
background(0);
```

Program 1.2

```
// Sets the screen to be 200, 200, so the width of the window is 200 pixels
// and the height of the window is 200 pixels
size(200, 200);
background(128);
```

## Modul Praktikum Grafika Komputer

---

### Program 1.3

```
// Sets the screen to be 200, 200, so the width of the window is 200 pixels
// and the height of the window is 200 pixels
size(200, 200);
background(255);
```

### Program 1.4

```
// Sets the screen to be 200, 200, so the width of the window is 200 pixels
// and the height of the window is 200 pixels
size(200, 200);
background(255,128,0);
```

Catatan : supaya lebih aman, silahkan anda simpan program-program diatas pada folder anda sendiri

### Tugas

- Lakukan analisa pada listing diatas !
- Setelah melakukan percobaan diatas, buat layar output dengan ukuran bebas tetapi dengan bentuk bujur sangkar, kemudian beri warna kuning, kemudian ungu, dan coklat.

# Modul Praktikum Grafika Komputer

---

Tgl.	TUGAS PRAKTIKUM 01	Ttd. Dosen :
Teknik Informatika		Nilai :

# Modul Praktikum Grafika Komputer

## PRAKTIKUM 02

### OUTPUT PRIMITIVES

#### Tujuan Instruksional

Setelah mengikuti praktikum ini, mahasiswa diharapkan dapat untuk :

1. Memahami bentuk output primitif / bentuk dasar
2. Membuat objek baru dengan mengkombinasi bentuk dasar

#### Elemen Dasar Gambar pada Grafika Komputer

Citra pada grafika computer menggunakan elemen dasar grafik. Elemen-elemen ini memudahkan untuk menggambar bentuk objek pada layar monitor. Dalam grafika computer terdapat 4 elemen dasar grafik yaitu :

1. Titik (point)
2. Garis (line)
3. Bentuk segi
4. Bentuk Bundar

#### Points and Lines

Untuk menggambar titik (point) digunakan perintah `point(x,y)` dimana nilai `x` dan `y` adalah koordinat pada layar. Sedangkan untuk membuat garis digunakan perintah `lines(x1,y1,x2,y2)`. Sebagai percobaan pertama silahkan ketik program berikut ini :

#### Program 2.1

```
int d = 40;
int p1 = d;
int p2 = p1+d;
int p3 = p2+d;
int p4 = p3+d;
size(200, 200);
background(0);
//Draw line from location (50,50) until (100,150)
stroke(255);
line(50, 50, 100, 150);

// Draw gray box
stroke(255);
line(p3, p3, p2, p3);
line(p2, p3, p2, p2);
line(p2, p2, p3, p2);
line(p3, p2, p3, p3);

// Draw white points
stroke(255);
point(p1, p1);
point(p1, p3);
point(p2, p4);
point(p3, p1);
point(p4, p2);
point(p4, p4);
```

## Modul Praktikum Grafika Komputer

Untuk mengubah warna garis, dapat digunakan perintah `stroke(x)` atau `stroke(r,g,b)`. Selain itu ketebalan garis dapat kita atur dengan menggunakan perintah `strokeWeight(x)` dengan nilai `x` adalah jumlah ketebalan pixel. Jika ingin merubah tampilan ujung garis, dapat juga kita tambahkan perintah `strokeCap(mode)`. Mode yang digunakan adalah `SQUARE`, `ROUND` dan `PROJECT`. Untuk lebih jelas, silahkan ketik program berikut ini:

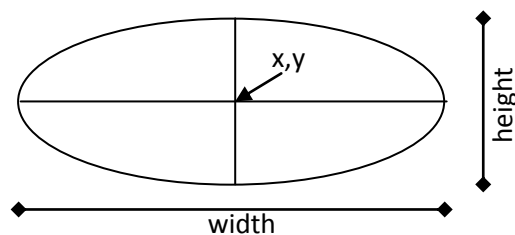
### Program 2.2

```
size(200, 200);
background(0);

// Draw white line standart
stroke(255);
line(25, 5, 175, 5);
// Draw red line
stroke(255,0,0);
line(25, 25, 175, 25);
// Draw Green line with 5 points thicknes
stroke(0,255,0);
strokeWeight(5);
line(25, 50, 175, 50);
//Draw Blue line with 10 thickness and square tip line
stroke(0,0,255);
strokeWeight(10);
strokeCap(SQUARE);
line(25, 75, 175, 75);
```

### Objek Bundar

Untuk membuat objek bundar dapat menggunakan beberapa cara bentuk, yaitu : ellipse dan busur. Untuk bentuk ellipse dapat digunakan perintah `ellipse(x,y,width,height)` dengan nilai `x` dan `y` adalah sebagai pusat ellipse seperti ditunjukkan pada gambar 2.1 berikut :



Gambar 2.2 Model Penggambaran Bentuk Ellipse

Sedikit berbeda dengan penggunaan busur (`arc`). Pada ellipse kita dapat membuat bentuk bundar secara utuh, namun pada busur kita dapat membuat bentuk bundar hanya sebagian dalam arti kurva terbuka. Perintah yang digunakan adalah `arc(x,y,width,height,start,stop)`, `x` dan `y` adalah posisi pusat busur, `width` adalah lebar dan `height` adalah tinggi. Penggunaan `start` pada `arc` adalah posisi awal penggambaran dan `stop` adalah posisi akhir penggambaran busur.

## Modul Praktikum Grafika Komputer

Nilai awal dan akhir pada arc, adalah menggunakan satuan nilai PI ( $\pi$ ). Untuk  $180^\circ$  sama dengan  $PI = 3,14$  atau  $(22/7)$  jika dibuat dengan persamaan lain yaitu :  $(\text{sudut} * PI) / 180^\circ$ . Untuk lebih yakin, silahkan tulis program berikut ini :

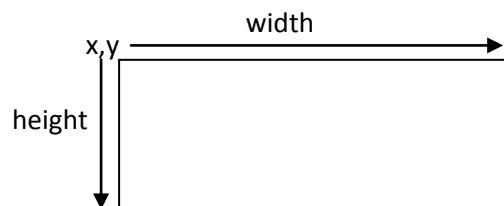
### Program 2.3

```
size(400,150);
background(255);
//draw ellipse
fill(255,0,0);
ellipse(50,50,75,100);
// draw arc 90 degree clockwise
fill(0,0,255);
arc(100,50,100,100,0,1.57);
// draw arc 90 degree
fill(0,255,0);
arc(175,50,100,100,(0*PI)/180,(90*PI)/180);
// draw arc 90 degree
noFill();
stroke(255,0,0);
arc(250,50,100,100,(0*PI)/180,(90*PI)/180);
```

Perintah `fill(r,g,b)` digunakan untuk member warna area didalam ellipse atau arch, dan `noFill()` digunakan untuk menghilangkan warna didalam area. Untuk perintah `stroke()`, `noStroke()`, `strokeCap()` dan `strokeWeight()` juga dapat diaplikasikan pada objek.

### Bentuk Segi

Bentuk segi yang dimaksud adalah berupa segi empat dan segi tiga atau segi lainnya. Untuk yang pertama kita mencoba untuk membuat segi empat standar dengan menggunakan perintah `rect(x,y,width,height)`, dimana  $x$  dan  $y$  adalah posisi awal sudut seperti ditunjukkan pada gambar 2.3 berikut:



Gambar 2.3. Model penggambaran perintah `rect()`

Perintah lain yang dapat digunakan adalah `quad(x1,y1,x2,y2,x3,y3,x4,y4)`.  $x_1$  dan  $y_1$  menyatakan koordinat  $xy$  titik sudut pertama,  $x_2$  dan  $y_2$  menyatakan koordinat  $xy$  titik sudut kedua,  $x_3$  dan  $y_3$  menyatakan koordinat  $xy$  titik sudut ketiga,  $x_4$  dan  $y_4$  menyatakan koordinat  $xy$  titik sudut keempat. Bentuk selanjutnya adalah segitiga yang dapat dibuat dengan perintah `triangle(x1,y1,x2,y2,x3,y3)`. Untuk lebih jelas silahkan coba program berikut ini:

### Program 2.4

```
size(200, 200);
smooth();
background(0);
```

# Modul Praktikum Grafika Komputer

```
noStroke();
fill(226);
triangle(10, 10, 10, 200, 45, 200);
rect(45, 45, 35, 35);
quad(105, 10, 120, 10, 120, 200, 80, 200);
triangle(160, 10, 195, 200, 160, 200);
```

Bentuk yang lain dalam penggunaan bentuk bebas yaitu dengan menggunakan perintah :

```
beginShape (mode);
```






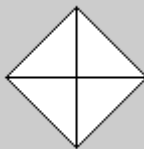

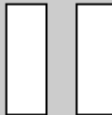

```
vertex(x,y);
```

```
vertex(x,y);
```

.....

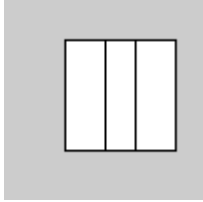
```
endShape();
```

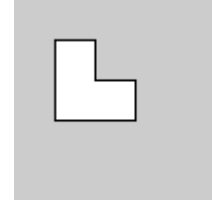
dengan menggunakan mode perintah ini, setiap vertex akan terhubung menjadi sebuah objek pada satu macam shape. Parameter mode shape dapat diisi dengan POINTS, LINES, TRIANGLES, TRIANGLE\_FAN, TRIANGLE\_STRIP, QUADS, QUAD\_STRIP.

	<pre>beginShape(); vertex(30, 20); vertex(85, 20); vertex(85, 75); vertex(30, 75); endShape(CLOSE);</pre>		<pre>beginShape(TRIANGLES); vertex(30, 75); vertex(40, 20); vertex(50, 75); vertex(60, 20); vertex(70, 75); vertex(80, 20); endShape();</pre>
	<pre>beginShape(POINTS); vertex(30, 20); vertex(85, 20); vertex(85, 75); vertex(30, 75); endShape();</pre>		<pre>beginShape(TRIANGLE_S TRIP); vertex(30, 75); vertex(40, 20); vertex(50, 75); vertex(60, 20); vertex(70, 75); vertex(80, 20); vertex(90, 75); endShape();</pre>
	<pre>beginShape(LINES); vertex(30, 20); vertex(85, 20); vertex(85, 75); vertex(30, 75); endShape();</pre>		<pre>beginShape(TRIANGLE_F AN); vertex(57.5, 50); vertex(57.5, 15); vertex(92, 50); vertex(57.5, 85); vertex(22, 50); vertex(57.5, 15); endShape();</pre>
	<pre>noFill(); beginShape(); vertex(30, 20); vertex(85, 20); vertex(85, 75); vertex(30, 75); endShape();</pre>		<pre>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();</pre>
	<pre>noFill(); beginShape(); vertex(30, 20); vertex(85, 20); vertex(85, 75); vertex(30, 75); endShape(CLOSE);</pre>		



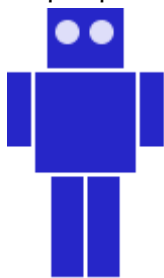
# Modul Praktikum Grafika Komputer

	<pre>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();</pre>
---	--

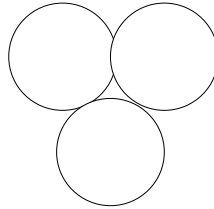
	<pre>beginShape(); vertex(20, 20); vertex(40, 20); vertex(40, 40); vertex(60, 40); vertex(60, 60); vertex(20, 60); endShape(CLOSE);</pre>
--	---

## Tugas

1. Buatlah sebuah objek lingkaran
2. Buatlah objek berikut dengan menggunakan kombinasi dari bentuk output primitive



dan



## Modul Praktikum Grafika Komputer

---

Tgl.	TUGAS PRAKTIKUM 02	Ttd. Dosen :
Teknik Informatika		Nilai :

#### Tujuan Instruksional

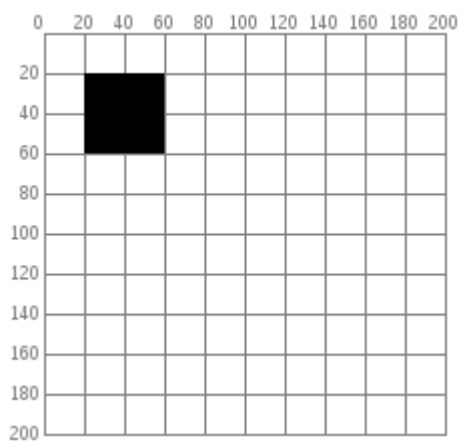
Setelah mengikuti praktikum ini, mahasiswa diharapkan dapat untuk :

1. Memahami model transformasi objek
2. Membuat animasi tingkat dasar

Processing memiliki fungsi built-in yang membuatnya mudah bagi Anda untuk memiliki objek dalam sketsa gerak, memutar, dan tumbuh atau menyusut. Tutorial ini akan memperkenalkan Anda ke proses Translation, Rotate, dan scalling sehingga Anda dapat menggunakannya dalam sketsa Anda. Secara mendasar teknik animasi adalah Translation (pergeseran), Rotation (perputaran) dan Scalling (penskalaan), dengan mengkombinasi ketiga proses tersebut nantinya anda akan mendapatkan sebuah objek yang dianimasi.

#### Translation

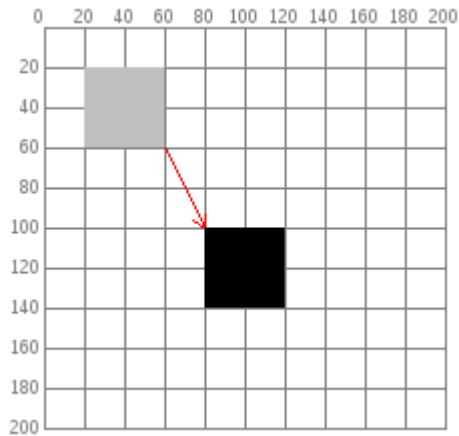
Seperti yang anda ketahui, layar yang disediakan processing adalah selayaknya sepotong kertas grafik. Jika kita ingin melakukan proses translasi sebenarnya terdapat dua pemikiran. Sebagai contoh adalah sebuah persegi sederhana dengan kode `rect(20,20,40,40)` seperti terlihat pada gambar 3.1.



Gambar 3.1. Objek persegi sederhana

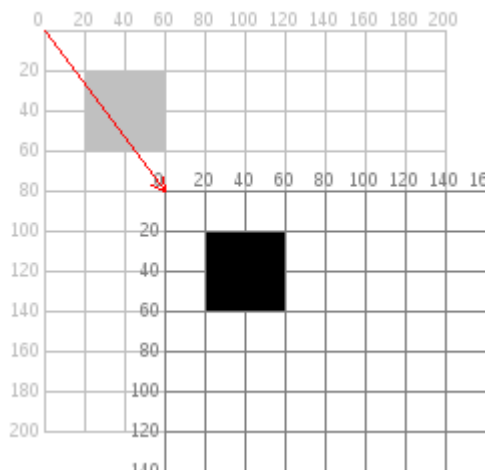
Jika anda ingin memindahkan persegi tersebut ke koordinat 60 unit ke kanan dan 80 unit ke bawah, pada pemikiran pertama adalah bahwa anda langsung memindahkan objek ke area tersebut dengan cara menambahkan nilai x dengan 60 dan nilai y dengan 80, sehingga kodenya adalah `rect(20+60,20+80,40,40)` seperti terlihat pada gambar 3.2 dan program 3.1.

## Modul Praktikum Grafika Komputer



Gambar 3.2. Proses pemindahan objek dengan translasi

Tapi ada cara yang lebih menarik untuk melakukannya, yaitu memindahkan kertas grafik sebagai gantinya, perhatikan gambar 3.3. Jika Anda memindahkan kertas grafik 60 unit kanan dan 80 unit ke bawah, Anda akan mendapatkan hasil persis visual yang sama. Memindahkan sistem koordinat seperti ini disebut translation.



Gambar 3.3. Proses translasi sistem koordinat

Hal penting untuk diperhatikan dalam diagram sebelumnya adalah bahwa, persegi panjang yang bersangkutan, itu tidak bergerak sama sekali tetap sudut kiri atas adalah masih di (20,20). Bila Anda menggunakan transformasi, hal yang menarik tidak pernah mengubah posisi sistem koordinat. Berikut ini adalah kode yang digunakan untuk melakukan transformasi objek dan transformasi system koordinat.

### Program 3.1

```
void setup()
{
  size(200, 200);
  background(255);
  noStroke();
  // draw the original position in gray
  fill(192);
  rect(20, 20, 40, 40);
```

## Modul Praktikum Grafika Komputer

---

```
// draw a translucent red rectangle by changing the object coordinates
fill(255, 0, 0, 128);
rect(20 + 60, 20 + 80, 40, 40);
}
```

### Program 3.2

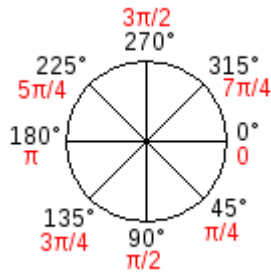
```
void setup()
{
  size(200, 200);
  background(255);
  noStroke();
  // draw the original position in gray
  fill(192);
  rect(20, 20, 40, 40);
  // draw a translucent blue rectangle by translating the grid
  fill(0, 0, 255, 128);
  pushMatrix();
  translate(60, 80);
  rect(20, 20, 40, 40);
  popMatrix();
}
```

Mari kita lihat kode terjemahan lebih terinci. `pushMatrix ()` adalah fungsi built-in yang menyelamatkan posisi sistem koordinat. Translasi (60, 80) bergerak pada sistem koordinat 60 unit dan 80 unit kanan bawah. `Rect (20, 20, 40, 40)` memposisikan persegi panjang di tempat yang sama awalnya. Ingat, objek tidak bergerak namun grid bergerak sebagai gantinya. Akhirnya, `popMatrix ()` mengembalikan sistem koordinat awal sebelum Anda melakukan translasi.

Anda bisa melakukan translasi (-60, -80) untuk memindahkan grid kembali ke posisi semula. Namun, ketika Anda mulai melakukan operasi yang lebih canggih dengan sistem koordinat, akan lebih mudah untuk menggunakan `pushMatrix ()` dan `popMatrix ()` untuk menyimpan dan mengembalikan status daripada harus membatalkan semua operasi Anda.

### Rotasi

Selain bergerak pada grid, Anda juga dapat memutar itu dengan fungsi `rotate()`. Fungsi ini memerlukan satu argumen, yang merupakan jumlah radian yang Anda ingin putar. Dalam Processing, semua fungsi harus dilakukan dengan mengukur sudut rotasi dalam radian (rad), bukan derajat. Ketika Anda berbicara tentang sudut dalam derajat, Anda mengatakan bahwa lingkaran penuh memiliki 360°. Ketika Anda berbicara tentang sudut dalam radian, Anda mengatakan bahwa lingkaran penuh telah  $2\pi$  radian. Berikut ini adalah diagram tentang bagaimana langkah-langkah Pengolahan sudut dalam derajat (hitam) dan radian (merah) gambar 3.4.

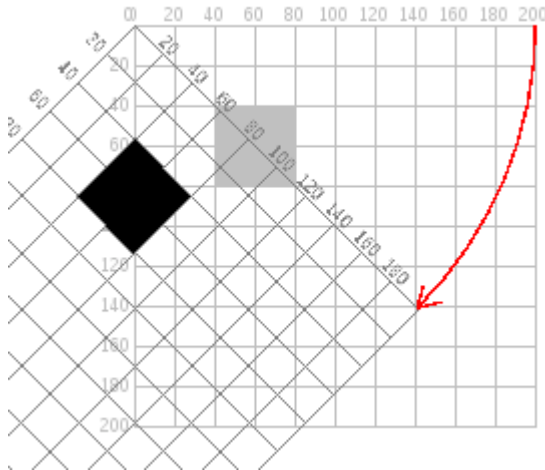


Gambar 3.4. Perbandingan sudut Radian dan Derajat

Karena kebanyakan orang berpikir dalam derajat, Processing memiliki built-in fungsi `radian()` yang mengambil sejumlah derajat sebagai argumen dan mengkonversi untuk Anda. Ini juga memiliki fungsi `derajat()` yang mengubah radian ke derajat. Sebagai percobaan pertama kita coba lakukan rotasi 45 derajat clockwise.

Program 3.3	
<pre>void setup() {   size(200, 200);   background(255);   smooth();   fill(192);   noStroke();   rect(40, 40, 40, 40);   pushMatrix();   rotate(radians(45));   fill(0);   rect(40, 40, 40, 40);   popMatrix(); }</pre>	

Dari program 3.3, apa yang terjadi? Kenapa objek persegi yang diputar menjadi terpotong?. Jawabannya adalah, sebenarnya objek persegi diputar dengan titik pusat rotasi adalah (0,0) atau dengan kata lain dapat diartikan bahwa objek persegi tidak di-rotasi, namun yang dirotasi adalah system koordinatnya, seperti diperlihatkan pada gambar 3.5.



Gambar 3.5. Proses rotasi grid

Cara yang lain adalah dengan menggunakan lokasi lain untuk dijadikan sebagai pusat rotasi. Sebagai contoh, objek persegi yang diatas akan diputar 45 derajat dengan menggunakan titik pusat rotasi adalah pojok kiri atas. Cara yang digunakan adalah :

1. Translasi lokasi koordinat (0,0) kearah pusat rotasi, dalam hal ini adalah (40,40)
  2. Gunakan fungsi rotate() dengan parameter 45 derajat untuk memutar grid
  3. Gambarkan kembali objek dengan posisi awal adalah lokasi original grid
  4. Kembalikan posisi grid ke awal dengan menggunakan fungsi popMatrix()
- berikut adalah kode yang dimaksudkan :

Program 3.4

```
void setup()
{
  size(200, 200);
  background(255);
  smooth();
  fill(192);
  noStroke();
  rect(40, 40, 40, 40);

  pushMatrix();
  // move the origin to the pivot point
  translate(40, 40);

  // then pivot the grid
  rotate(radians(45));

  // and draw the square at the origin
  fill(0);
  rect(0, 0, 40, 40);
  popMatrix();
}
```



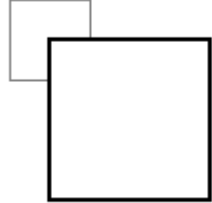
## Scaling

Transformasi yang terakhir adalah scaling, yang mana prosesnya adalah mengubah ukuran objek baik membesara atau pun mengecil. Lihat contoh

## Modul Praktikum Grafika Komputer

---

program berikut yang mengubah ukuran objek menjadi dua kali lebih besar dari originalnya.

Program 3.5	
<pre>void setup() {   size(200,200);   background(255);    stroke(128);   rect(20, 20, 40, 40);   stroke(0);   pushMatrix();   scale(2.0);   rect(20, 20, 40, 40);   popMatrix(); }</pre>	

Dari program diatas, maka seolah olah proses scalling memiliki titik prespektif dari pusat (0,0).

Tugas

1. Buatlah sebuah objek seperti berikut :



2. Buatlah program untuk melakukan proses scalling dari sembarang titik pusat skala



## Modul Praktikum Grafika Komputer

---

Tgl.	TUGAS PRAKTIKUM 03	Ttd. Dosen :
Teknik Informatika		Nilai :

# Modul Praktikum Grafika Komputer

## PRAKTIKUM 04

### PEMROGRAMAN MODE CONTINUOUS dan KENDALI PROGRAM

#### Tujuan Instruksional

1. Memahami pemrograman mode continuous
2. Memahami model animasi sederhana
3. Memahami kendali program dengan Seleksi Kondisi dan Perulangan

#### LANDASAN TEORI

Struktur program dalam mode continuous adalah sebagai berikut :

Struktur Program Mode Continuous
<pre>&lt;data global&gt; Void setup() {     pernyataan; } Void draw() {     pernyataan; }</pre>

Keterangan:

1. Setiap program hanya mempunyai satu fungsi setup() dan draw().
2. Setup () akan dipanggil saat pertama kali program berjalan, kemudian secara berkala fungsi draw() dijalankan.
3. Jika perintah noLoop () dipanggil pada fungsi setup () maka fungsi draw() hanya akan dipanggil sekali saat program berjalan pertama kali.

#### PERCOBAAN

1. Membedakan penggunaan fungsi noLoop() dan dan tanpa noLoop()

Lakukan percobaan menggunakan listing berikut ini, abaikan arti dari perintah-perintah yang ada di dalam program tersebut, tetapi konsentrasikan perhatian terhadap apa yang dilihat saat program dijalankan.

Percobaan 1
<pre>//sketch menggunakan noLoop() void setup() {     noLoop(); } void draw () {     background (200);     int s = second ();     int m = minute();     int h = hour();</pre>

## Modul Praktikum Grafika Komputer

---

```
line(s,0,s,33);  
line(m,33,m,66):  
line(h,66,h,100);  
}
```

Jalankan sketch diatas, perhatikan garis akan berubah lokasi sesuai dengan waktu computer anda. Hentikan sketch tersebut dan kemudian letakkan tanda // di depan noLoop() pada fungsi setup(). Kemudian jalankan kembali sketch tersebut, perhatikan meskipun garis digambar tetapi garis tersebut tidak pernah diubah lokasinya, hal ini terjadi karena perintah noLoop() menyebabkan fungsi draw() hanya dipanggil sekali saat program berjalan dan setelah itu tidak pernah dipanggil kembali.

### 2. Membuat animasi garis

```
void setup()  
{  
    size ( 400,400); // layar output  
    background (255); // warna layar putih  
}  
void draw ()  
{  
    int d;  
    d=second();  
    strokeWeight(4);  
    stroke(200,0,0);  
    line(10,30,10+5*d,30); // garis merah bergerak ke kanan  
}
```

### TUGAS LATIHAN

1. Amati listing program animasi garis sederhana diatas!
2. Buat program dengan mode continous untuk membuat 3 garis dengan warna dan ukuran yang berbeda dengan gerakan garis dimulai dari sisi kiri kemudian bergerak ke kanan, koordinat garis bebas!
3. Buat program dengan mode continous untuk membuat 2 garis dengan warna dan ukuran yang berbeda dengan gerakan garis dimulai dari atas ke bawah dan dari bawah ke atas koordinat garis bebas!

## Modul Praktikum Grafika Komputer

---

### 3. Membuat animasi kotak

```
void setup()
{
    size ( 400,400); // layar output
    background (255); // warna layar putih
}
void draw ()
{
    int d;
    d=second();
    strokeWeight(4);
    fill(200,0,200);
    rect(10,10,50+5*d,50+d*5);
    // kotak dari pojok kiri membesar ke kanan
}
```

#### TUGAS LATIHAN

1. Buat program dengan menggunakan mode continuous untuk membuat sebuah kotak dengan ukuran dan warna bebas:
  - a. posisi awal di pojok kanan atas layar kemudian membesar dan bergerak ke arah kiri bawah atau diagonal.
  - b. posisi awal di pojok kanan bawah layar kemudian membesar dan bergerak ke arah kiri bawah atau diagonal.
  - c. posisi awal di pojok kiri bawah layar kemudian membesar dan bergerak ke arah kanan atas atau diagonal.

#### TUGAS TAKE-HOME

1. Buat animasi dari gambar dengan topik berikut ini :
  - a. Pemandangan desa
  - b. Kepadatan lalu lintas
  - c. Pemandangan pantai

# Modul Praktikum Grafika Komputer

## LANDASAN TEORI

### Kendali Program

Kendali program adalah perintah-perintah yang mempengaruhi jalannya program. Hal ini disebabkan karena terpenuhinya suatu kondisi tertentu. Program mengetahui apakah kondisi tersebut terpenuhi atau tidak dengan melakukan pengujian terhadap data. Pengujian dilakukan dengan melihat apakah data yang diuji memenuhi hubungan yang diisyaratkan. Hubungan antar data dinyatakan melalui operator relasi dan operator logika.

#### 1. Operator Relasi

Operator relasi menjelaskan hubungan antar 2 data. Operator ini menghasilkan bilangan true atau false.

Operator	Keterangan	Contoh
$A \neq B$	A tidak sama dengan B	$A \neq 10$
$A > B$	A lebih besar dari B	$A > 10$
$A \geq B$	A lebih besar sama dengan B	$A \geq 10$
$A < B$	A lebih kecil B	$A < 10$
$A \leq B$	A lebih kecil sama dengan B	$A \leq 10$
$A == B$	A sama dengan B	$A == 10$

#### 2. Operator Logik

Operator ini membandingkan dua buah ekspresi dan menghasilkan nilai berdasarkan jenis perbandingan yang digunakan.

Operator	Keterangan	Contoh
!	Not : kebalikan dari nilai yang ada	$A = \text{false};$ $B = !A;$ // b berisi true
&&	And: bernilai true apabila kedua relasi bernilai true	$A = 10;$ $B = 20;$ $C = 30;$ $D = 40;$ $(A < B) \&\& (C < D)$
	Or: bernilai relasi true apabila salah satu relasi bernilai true atau kedua relasi bernilai true	$A = 10;$ $B = 20;$ $C = 30;$ $D = 40;$ $(A > B)    (C > D)$

# Modul Praktikum Grafika Komputer

---

## A. Percabangan

Percabangan adalah keadaan dimana program menjalankan sekelompok perintah apabila sejumlah kondisi dipenuhi atau tidak dipenuhi.

Perintah yang digunakan:

```
if (kondisi)
{ pernyataan jika kondisi dipenuhi;
}
else
{ pernyataan jika kondisi tidak dipenuhi;
}
```

## B. Pengulangan

Pengulangan merupakan situasi dimana program mengulang sekelompok perintah selama sebuah kondisi masih dipenuhi. Processing menyediakan dua perintah pengulangan yaitu:

### 1. For

Menggunakan nilai integer atau float untuk melakukan perulangan.

```
For (init;test;update)
{ pernyataan;}
```

### 2. While

Menggunakan sembarang data untuk melakukan pengulangan.

```
nilai awal;
while (kondisi)
{pernyataan;}
```

## PERCOBAAN

### 1. Membuat garis dengan percabangan

Listing:

```
int a=20;
int b=30;
void setup()
{
    size(300,300);
    background(255,255,0);
}
void draw()
{
    If (a<30)
    {
        stroke(255,0,0);
        line (a,b,a+30,b);
    }
    else
    {
        line(a,b+30,a+30,b+30);
    }
}
```

2. Membuat bujursangkar dengan percabangan

Listing:

```
int x=200;
int y=200;
void setup()
{
  size (400,400);
  background(255);
}
void draw ()
{
  If (x==200)&&(y==200)
  {
    rect(x,y,10,10);
  }
  else
  { fill (255,0,0);
    rect(x-100,y-100,50,50);
  }
}
```

### TUGAS LATIHAN

Buatlah gambar sebuah papan catur dengan menggunakan percabangan.

3. Membuat garis dengan pengulangan for

Listing:

```
void setup()
{ setup (400,400);}
void draw()
{ for ( int i=0; i<390;i=i+10)
  line ( 10,i,390,i);
}
```

4. Membuat bujursangkar dengan pengulangan for

Listing:

```
void setup()
{ setup (400,400);}
void draw()
{ for ( int i=0; i<100;i=i+10)
  rect (i,i,50,50);
}
```

### TUGAS LATIHAN

1. Buat program untuk membuat garis-garis vertical dengan menggunakan pengulangan for, ukuran dan warna garis bebas.
2. Buatlah program untuk membuat objek-objek lingkaran dengan menggunakan pengulangan for.

## Modul Praktikum Grafika Komputer

---

Tgl.	TUGAS PRAKTIKUM 04	Ttd. Dosen :
Teknik Informatika		Nilai :



## PRAKTIKUM 05

### KURVA : BEIZER DAN B-SPLINE

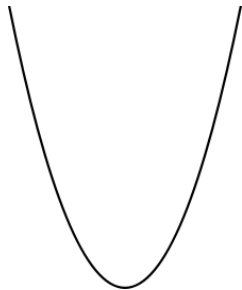
#### Tujuan Instruksional

Setelah mengikuti praktikum ini, mahasiswa diharapkan dapat untuk :

1. Mengetahui proses pembentukan kurva
2. Membuat objek

#### Kurva

Dalam matematika, sebuah kurva adalah suatu objek geometri yang merukanan *satu-dimensi* dan *kontinyu*. Banyak kurva khusus telah dipelajari dalam geometri, mulai dari lingkaran. Dalam pembahasan ini lingkaran bukan yang akan dipraktikkan, namun kurva yang dimaksud adalah bentuk objek parabola seperti terlihat pada gambar 5.1 berikut:

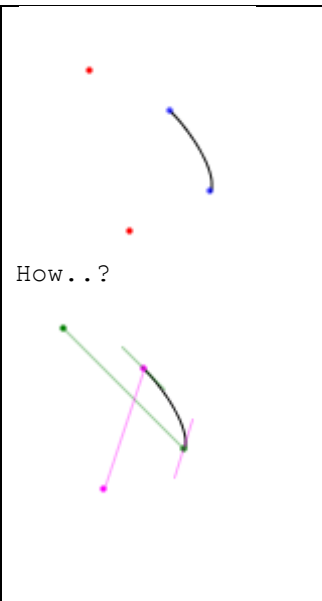


Gambar 5.1. Kurva Objek Parabola

Untuk menggambarkan objek parabola/kurva terbuka, processing memiliki fungsi `curve()` yang dapat digunakan untuk membentuk kurva standar. Sebagai parameter digunakan  $x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4$ . Parameter  $x_1, y_1$  dan  $x_4, y_4$  digunakan sebagai penanda awal dan akhir kurva, sedangkan  $x_2, y_2$  dan  $x_3, y_3$  digunakan sebagai control point kurva. Sebagai contoh coba ketik program berikut ini :

#### Program 5.1

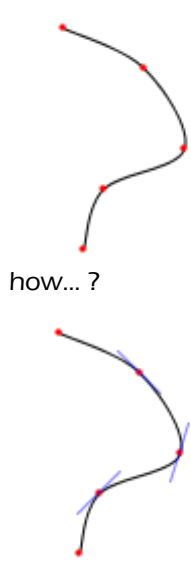
```
void setup()
{
  size(200, 200);
  background(255);
  smooth();
  //Draw curve
  stroke(0);
  curve(40, 40, 80, 60, 100, 100, 60,
  120);
  //Draw start point, end point and
  control point
  noStroke();
  fill(255, 0, 0);
  ellipse(40, 40, 3, 3);
  fill(0, 0, 255, 192);
  ellipse(100, 100, 3, 3);
  ellipse(80, 60, 3, 3);
  fill(255, 0, 0);
  ellipse(60, 120, 3, 3);
}
```



# Modul Praktikum Grafika Komputer

## Continuous Spline Curves

Cara yang lain untuk membuat kurva adalah dengan menggunakan fungsi `curveVertex(x,y)`. Fungsi ini hanya dapat digunakan dengan menggunakan `beginShape()` dan `endShape()` tanpa disertai `MODE`. Coba praktikkan program 5.2 berikut :

Program 5.2	
<pre>void setup() {   int[ ] coords = {     40, 40, 80, 60, 100, 100, 60, 120, 50, 150   };   int i;   size(200, 200);   background(255);   smooth();    noFill();   stroke(0);   beginShape();   // the first control point   curveVertex(40, 40);   // is also the start point of curve   curveVertex(40, 40);   curveVertex(80, 60);   curveVertex(100, 100);   curveVertex(60, 120);   // the last point of curve   curveVertex(50, 150);   // is also the last control point   curveVertex(50, 150);   endShape();    // use the array to keep the code shorter;   // you already know how to draw ellipses!   fill(255, 0, 0);   noStroke();   for (i = 0; i &lt; coords.length; i += 2)   {     ellipse(coords[i], coords[i + 1], 3, 3);   } }</pre>	

## Kurva Beizer

Kurva ini ditentukan oleh serangkaian jangkar dan kontrol poin. Dua parameter pertama menentukan titik jangkar pertama dan dua terakhir parameter menentukan titik anchor lainnya. Parameter tengah menentukan titik kontrol yang menentukan bentuk kurva. kurva Bezier dikembangkan oleh insinyur Prancis Pierre Bezier. Kode yang digunakan adalah `bezier(x1, y1, cx1, cy1, cx2, cy2, x2, y2)`; atau jika pada bidang 3 dimensi adalah `bezier(x1, y1, z1, cx1, cy1, cz1, cx2, cy2, cz2, x2, y2, z2)`. Untuk lebih jelas silahkan coba program berikut :

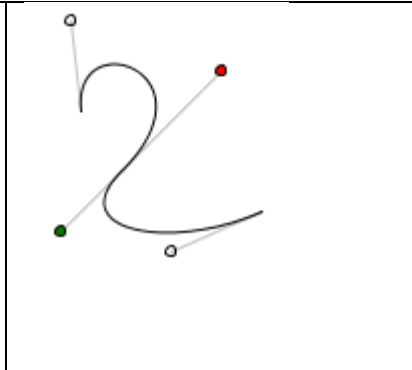
Program 5.3	
<pre>noFill(); stroke(255, 102, 0); line(85, 20, 10, 10); line(90, 90, 15, 80); stroke(0, 0, 0); bezier(85, 20, 10, 10, 90, 90, 15, 80);</pre>	
<pre>noFill(); stroke(255, 102, 0); line(30, 20, 80, 5); line(80, 75, 30, 75); stroke(0, 0, 0); bezier(30, 20, 80, 5, 80, 75, 30, 75);</pre>	
<pre>void setup( ) {   size(150, 150);   background(255);   smooth();   //Draw start,end point and control   point   // endpoints of curve   ellipse(50, 75, 5, 5);   ellipse(100, 75, 5, 5);   fill(255, 0, 0);   // control points   ellipse(25, 25, 5, 5);   ellipse(125, 25, 5, 5);   noFill();   //Draw Beizer curve   stroke(0);   bezier(50, 75, 25, 25, 125, 25, 100, 75); }</pre>	<p>How...?</p>

## Continuous Beizer Curve

Sebenarnya sama seperti bezier biasa, namun hal ini dibuat secara kontinyu. Anda dapat menggunakan perintah `bezierVertex()` dengan parameter `cpx1`, `cpy1`, `cpx2`, `cpy2`, `x`, `y` dengan `cpx1`, `cpy1` Koordinat titik control, `cpx2`, `cpy2` Koordinat titik kontrol kedua, `x`, `y` titik berikutnya pada kurva. Berikut adalah contoh penggunaan mode kontinyu bezier.

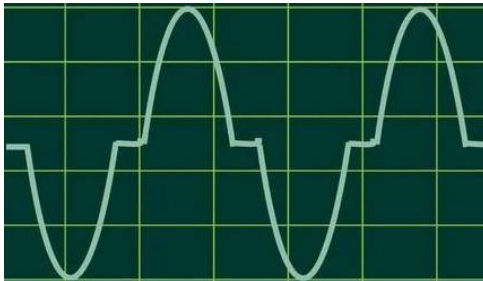
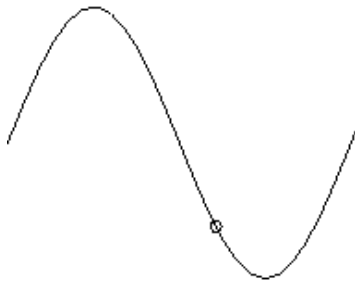
Program 5.4	
<pre>void setup( ) {   size(150, 150);   background(255);   smooth();   beginShape();   vertex(30, 70); // first point   bezierVertex(25, 25, 100, 50, 50, 100);   bezierVertex(50, 140, 75, 140, 120, 120);   endShape(); }</pre>	

```
void setup( )  
{  
  size(150, 150);  
  background(255);  
  smooth();  
  beginShape();  
  vertex(30, 70); // first point  
  bezierVertex(25, 25, 100, 50, 50, 100);  
  bezierVertex(20, 130, 75, 140, 120, 120);  
  endShape();  
}
```



## Tugas

1. Buatlah sebuah kurva gelombang sinus dengan menggunakan fungsi `curve()`, `curveVertex()` dan `beizer()` dan bandingkan hasilnya.



## Modul Praktikum Grafika Komputer

---

Tgl.	TUGAS PRAKTIKUM 05	Ttd. Dosen :
Teknik Informatika		Nilai :

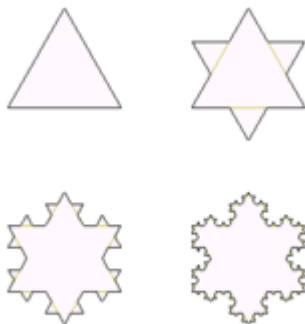
#### Tujuan Instruksional

Setelah mengikuti praktikum ini, mahasiswa diharapkan dapat untuk :

1. Mengetahui proses pembentukan objek
2. Membuat objek

#### Objek Fraktal

Fraktal adalah benda geometris yang kasar pada segala skala, dan terlihat dapat "dibagi-bagi" dengan cara yang radikal. Beberapa fraktal bisa dipecah menjadi beberapa bagian yang semuanya mirip dengan fraktal aslinya. Fraktal dikatakan memiliki detil yang tak hingga dan dapat memiliki struktur serupa diri pada tingkat perbesaran yang berbeda. Pada banyak kasus, sebuah fraktal bisa dihasilkan dengan cara mengulang suatu pola, biasanya dalam proses rekursif atau iteratif. Sebagai contoh adalah ditunjukkan pada gambar berikut ini :



Gambar 6.1. Model fraktal bunga salju

Pada processing telah disediakan contoh program fraktal, anda dapat melihat dengan memilih menu File → Example → Topic → Fractal and L-Elements.

#### Hidden Object

Pada processing, proses penampilan objek berdasar urutan layer. Dalam artian layer yang pertama di buat akan menjadi layer yang paling bawah jika ditambah objek lainnya lagi, sehingga jika kita melakukan penumpukan objek, yang terlihat adalah objek yang terakhir yang dibuat. Sebagai contoh, anda dapat membuka menu File → Example → Effect → unlimitedSprites.

Sebagai praktik, silahkan anda menuliskan program berikut ini :

#### Program 6.1

```
float frame = 0, speed = 1 / .0125, range = .125;  
float rx = PI / 6, ry = -PI / 6, rmax = PI / 240;  
  
bone root;
```

```
void setup () {
  size (256, 256, P3D);
  sphereDetail (5);

  root = new bone (0, -4, 0) // waist
  .link (new bone (0, -44, -6) // back
  .link (new bone (0, -24, 6)) // head
  .link (new bone (24, -4, -2) // left shoulder
  .link (new bone (4, 36, -4) // left elbow
  .link (new bone (-2, 32, 8)))) // left hand
  .link (new bone (-24, -4, -2) // right shoulder
  .link (new bone (-4, 36, -4) // right elbow
  .link (new bone (2, 32, 8)))) // right hand
  .link (new bone (12, 16, -2) // left hip
  .link (new bone (0, 56, 2) // left knee
  .link (new bone (0, 60, 0)))) // left foot
  .link (new bone (-12, 16, -2) // right hip
  .link (new bone (0, 56, 2) // right knee
  .link (new bone (0, 60, 0)))) // right foot
}

void draw () {
  if (!mousePressed) {
    speed = constrain (dist (128, 128, mouseX, mouseY) /
128, 0, 1) * .02;
    range = pow (constrain (dist (128, 128, mouseX, mouseY)
/ 128, 0, 1), 2) * .65;
  } else {
    rx = rx + (mouseX - pmouseX) * PI / width;
    ry = constrain (ry - (mouseY - pmouseY) * PI / height, -
HALF_PI, HALF_PI);
  }

  background (255);
  translate (128, 128, -192); rotateX (ry); rotateY (rx);
  noFill (); stroke (0, 0, 0, 31); box (256, 256, 256);
  root.rotate (0, 0, cycle (1, 0) * PI / 16);
  root.link (0).rotate (range * PI / 8 * (cycle (2, 0) - 1),
0, -cycle (1, 0) * PI / 14); // back
  root.link (0).link (0).rotate (-range * PI / 4 * cycle (2,
0), 0, 0); // head
  root.link (0).link (1).rotate (0, range * cycle (1, 0) *
PI / 4, 0); // left shoulder
  root.link (0).link (1).link (0).rotate (range * PI * cycle
(1, .5), 0, 0); // left elbow
  root.link (0).link (1).link (0).link (0).rotate (range *
HALF_PI * (cycle (1, .5) + .5), 0, 0); // left hand
  root.link (0).link (2).rotate (0, range * cycle (1, 0) *
PI / 4, 0); // right shoulder
  root.link (0).link (2).link (0).rotate (range * PI * cycle
(1, 0), 0, 0); // right elbow
  root.link (0).link (2).link (0).link (0).rotate (range *
HALF_PI * (cycle (1, 0) + .5), 0, 0); // right hand
  root.link (1).rotate (range * HALF_PI * cycle (1, 0), 0,
0); // left hip
  root.link (1).link (0).rotate (range * PI * cycle (1, 0),
0, -cycle (1, 0) * PI / 16); // left upper leg
  root.link (1).link (0).link (0).rotate ((-.5 - cycle (1,
.25)) * PI * range, 0, 0); // left lower leg
  root.link (2).rotate (range * HALF_PI * cycle (1, .5), 0,
0); // right hip
}
```

## Modul Praktikum Grafika Komputer

---

```
    root.link (2).link (0).rotate (range * PI * cycle (1, .5),
0, -cycle (1, 0) * PI / 16); // right upper leg
    root.link (2).link (0).link (0).rotate ((-.5 - cycle (1,
.75)) * PI * range, 0, 0); // right lower leg

    root.draw (null);

    frame += speed;
}

float cycle (float modifier, float displacement) { return
cos ((frame * modifier + displacement) * TWO_PI) / 2; }

class bone {
    // constants
    int maxchildren = 3;

    // variables
    float x, y, z, a, b ,c;
    int len; bone[] children;

    // methods
    bone (float x, float y, float z) { this.x = x; this.y = y;
this.z = z; a = b = c; len = 0; children = new
bone[maxchildren]; }
    bone rotate (float a, float b, float c) { this.a = a;
this.b = b; this.c = c; return this; }
    bone link (bone child) { children[len++] = child; return
this; }
    bone link (int which) { return children[which]; }

    void draw (bone parent) {
        pushMatrix ();

        rotateX (a); rotateY (b); rotateZ (c); translate (x, y,
z);

        if (parent != null) { stroke (0); line (0, 0, 0, -x, -y,
-z); }
        fill (159, 0, 191, 191); noStroke (); sphere (4);

        for (int i = 0; i < len; ++i) children[i].draw (this);

        popMatrix ();
    }
}
```

### Tugas

Buatlah sebuah objek bola yang berada di dalam kubus



## Modul Praktikum Grafika Komputer

---

Tgl.	TUGAS PRAKTIKUM 06	Ttd. Dosen :
Teknik Informatika		Nilai :

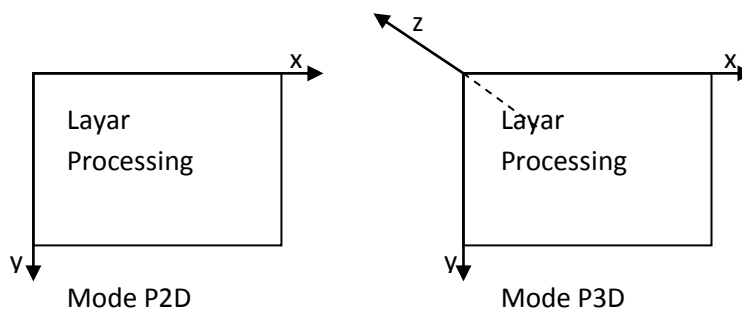
#### Tujuan Instruksional

Setelah mengikuti praktikum ini, mahasiswa diharapkan dapat untuk :

1. Mengetahui proses pemodelan objek
2. Membuat objek 2 dimensi dan 3 dimensi

#### Konsep Koordinat 3 Dimensi

Processing dapat melakukan pengolahan grafik baik 2 dimensi maupun 3 dimensi. Untuk melakukan pemrosesan 3 dimensi mode yang digunakan adalah P3D, misal saja `size(200,200,P3D)`; hal ini berarti ukuran layar 200 x 200 dengan mode proses 3 dimensi. Koordinat 2 dimensi berbeda dengan 3 dimensi, hal ini karena terdapat penambahan satu sumbu axis z pada lokasi origin (0,0) seperti tergambar pada gambar 6.1.



Gambar 6.1. Orientasi sumbu 2 dimensi dan 3 dimensi

Dari gambar 6.1, dijelaskan bahwa layar pada processing adalah bidang 2 dimensi dengan ujung pojok kiri atas sebagai titik origin (0,0). Sekarang dipikirkan untuk menampilkan model 3 dimensi dengan media 2 dimensi tersebut. Sehingga ditambahkan satu axis z untuk ordo ketiganya. Untuk mencobanya silahkan tuliskan program 6.1 berikut :

#### Program 6.1

```
void setup() {  
  size(300, 300, P3D);  
}  
void draw() {  
  background(200);  
  stroke(50);  
  box(100);  
}
```

## Modul Praktikum Grafika Komputer

---

### Program 6.2

```
void setup() {
  size(300, 300, P3D);
}

void draw() {
  background(200);
  stroke(50);
  translate(158, 148, 0);
  rotateX(mouseY * 0.05);
  rotateY(mouseX * 0.05);
  box(100);
}
```

Apa yang anda lihat pada hasil program 6.1 dan 6.2? coba anda analisis.

Pada program 6.2, origin axis (0,0,0) pada sumbu 3 dimensi dilakukan pergeseran sejauh 158 dan 148, sehingga bentuk objek `box()` dapat dilihat secara utuh.

Model objek yang lain adalah `sphere()`, silahkan anda coba untuk objek `sphere(100)`, bagaimana hasilnya.

### Model Prespektif

Menetapkan proyeksi perspektif foreshortening menerapkan, membuat obyek yang jauh tampak lebih kecil dibandingkan yang dekat. Parameter yang menentukan volume melihat dengan bentuk piramida terpotong. Objek dekat ke depan volume muncul ukuran sebenarnya mereka, sedangkan objek jauh terlihat lebih kecil. Proyeksi ini mensimulasikan perspektif dunia lebih akurat dibandingkan proyeksi ortografi. Versi perspektif tanpa parameter set default perspektif dan versi dengan empat parameter yang memungkinkan programmer untuk mengatur daerah tersebut tepat. Nilai default adalah: perspektif (PI/3.0, lebar / tinggi, cameraZ/10.0, cameraZ \* 10.0) di mana cameraZ adalah  $((\text{height}/2.0) / \tan(\text{PI} * 60.0/360.0))$ ; Sekarang silahkan coba program 6.3 berikut ini:

### Program 6.3

```
size(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);
```

### Tugas

Buatlah sebuah objek tata surya dengan 9 planet dan 1 matahari-nya

## Modul Praktikum Grafika Komputer

---

Tgl.	TUGAS PRAKTIKUM 07	Ttd. Dosen :
Teknik Informatika		Nilai :

#### Tujuan Instruksional

Setelah mengikuti praktikum ini, mahasiswa diharapkan dapat untuk :

1. Mengetahui proses pembentukan pencahayaan
2. Membuat objek 3 dimensi dengan pencahayaan

#### Light

Mengatur cahaya default ambient, cahaya directional, falloff, dan nilai-nilai specular. Secara default nilainya adalah `ambientLight (128, 128, 128)` dan `directionalLight (128, 128, 128, 0, 0, -1)`, `lightFalloff () (1, 0, 0)`, dan `lightSpecular (0, 0, 0)`. Mode pencahayaan perlu dimasukkan dalam fungsi `draw()` untuk tetap persisten dalam program perulangan. Menempatkan mereka di fungsi `setup ()` dari program looping akan menyebabkan mereka hanya memiliki efek pertama kalinya melalui loop. Sebagai contoh, silahkan anda praktekkan beberapa program berikut ini :

Kode pencahayan standar adalh `light()`.

#### Program 8.1

```
float spin = 0.0;

void setup()
{
  size(640, 360, P3D);
  noStroke();
}

void draw()
{
  background(51);
  lights();

  spin += 0.01;

  pushMatrix();
  translate(width/2, height/2, 0);
  rotateX(PI/9);
  rotateY(PI/5 + spin);
  box(150);
  popMatrix();
}
```

#### Directional Light

Menambahkan cahaya directional. Arah cahaya datang dari satu arah dan kuat ketika mengenai permukaan dan lebih lemah jika di sudut. Setelah mencapai permukaan, sebuah lampu arah memantulkan ke segala arah. Pengaruh dri parameter `v1`, `v2`, dan `v3` ditentukan oleh mode warna saat ini. Parameter `nx`, `ny`

## Modul Praktikum Grafika Komputer

---

dan  $n_z$  menentukan arah cahaya yang dihadapi. Misalnya, pengaturan  $n_y$  ke -1 akan menyebabkan geometri yang akan menyala dari bawah (terang menghadap langsung ke atas). Sintaksis untuk `directionalLight` adalah : `directionalLight (v1, v2, v3, nx, ny, nz)`.

Program 8.2

```
void setup() {
  size(640, 360, P3D);
  noStroke();
  fill(204);
}

void draw() {
  noStroke();
  background(0);
  float dirY = (mouseY / float(height) - 0.5) * 2;
  float dirX = (mouseX / float(width) - 0.5) * 2;
  directionalLight(204, 204, 204, -dirX, -dirY, -1);
  translate(width/2 - 100, height/2, 0);
  sphere(80);
  translate(200, 0, 0);
  sphere(80);
}
```

### Point Light

Menambahkan titik cahaya. Pengaruh dari parameter  $v_1$ ,  $v_2$ , dan  $v_3$  ditentukan oleh mode warna saat ini. Parameter  $x$ ,  $y$ , dan  $z$  mengatur posisi cahaya. Sintaksis yang digunakan adalah : `pointLight (v1, v2, v3, x, y, z)`

### Spot Light

Menambahkan spot light. Pengaruh dari parameter  $v_1$ ,  $v_2$ , dan  $v_3$  ditentukan oleh mode warna saat ini. Parameter  $x$ ,  $y$ , dan  $z$  parameter menentukan posisi cahaya dan  $n_x$ ,  $n_y$ ,  $n_z$  menentukan arah atau cahaya. Parameter sudut mempengaruhi sudut kerucut sorotan. Sintaksis yang digunakan adalah : `spotlight (v1, v2, v3, x, y, z, nx, ny, nz, sudut, konsentrasi)`

Program 8.3

```
void setup()
{
  size(640, 360, P3D);
  noStroke();
}

void draw()
{
  background(0);
  translate(width / 2, height / 2);

  // Orange point light on the right
  pointLight(150, 100, 0, // Color
            200, -150, 0); // Position
}
```

```
// Blue directional light from the left
directionalLight(0, 102, 255, // Color
                1, 0, 0); // The x-, y-, z-axis direction

// Yellow spotlight from the front
spotLight(255, 255, 109, // Color
          0, 40, 200, // Position
          0, -0.5, -0.5, // Direction
          PI / 2, 2); // Angle, concentration

rotateY(map(mouseX, 0, width, 0, PI));
rotateX(map(mouseY, 0, height, 0, PI));
box(150);
}
```

### Program 8.4

```
int concentration = 600; // Try values 1 -> 10000

void setup()
{
  //size(200, 200, P3D);
  size(640, 360, P3D);
  noStroke();
  fill(204);
  sphereDetail(60);
}

void draw()
{
  background(0);

  // Light the bottom of the sphere
  directionalLight(51, 102, 126, 0, -1, 0);

  // Orange light on the upper-right of the sphere
  spotLight(204, 153, 0, 360, 160, 600, 0, 0, -1, PI/2, 600);

  // Moving spotlight that follows the mouse
  spotLight(102, 153, 204, 360, mouseY, 600, 0, 0, -1, PI/2, 600);

  translate(width/2, height/2, 0);
  sphere(120);
}
```

### Tugas

Buatlah sebuah objek bola yang berputar dengan penggunaan lighting

## Modul Praktikum Grafika Komputer

---

Tgl.	TUGAS PRAKTIKUM 08	Ttd. Dosen :
Teknik Informatika		Nilai :



# Modul Praktikum Grafika Komputer

## PRAKTIKUM 09

### GRAPHICAL USER INTERFACE (GUI)

#### Tujuan Instruksional

Setelah mengikuti praktikum ini, mahasiswa diharapkan dapat untuk :

1. Mampu membuat perangkat antar muka
2. Mampu membuat animasi interatif

Model antarmuka untuk user seringkali dikaitkan dengan interaktif antara aplikasi dan user itu sendiri. Pada praktikum ini, akan dicoba untuk membuat antarmuka interaktif berupa pemberian masukan dari sebuah objek tombol dengan mouse atau keyboard.

#### Event Mouse

Processing secara otomatis melacak jika tombol mouse ditekan dan tombol yang ditekan. Nilai dari variabel `mouseButton` sistem baik KIRI, KANAN, atau PUSAT tergantung tombol mana yang ditekan. Untuk membuat `mouseButton`, kita dapat menggunakan fungsi `mousePressed()`, `mouseReleased()`, `mouseMoved()`, `mouseDragged()`. Sedangkan fungsi `mouseX()` dan `mouseY()` adalah digunakan untuk mengetahui lokasi mouse pada layar. Perhatikan program berikut ini:

##### Program 9.1

```
void draw()
{
  background(204);
  line(mouseX, 20, mouseX, 80);
}
```

##### Program 9.2

```
void draw()
{
  background(204);
  line(20, mouseY, 80, mouseY);
}
```

#### Fungsi `mousePressed()`

Fungsi ini bekerja jika tombol mouse ditekan. Nilai dari variabel sistem `mousePressed` adalah `true` jika tombol mouse ditekan dan `false` jika tidak ada tombol yang ditekan. Berikut adalah contoh programnya :

## Modul Praktikum Grafika Komputer

---

### Program 9.3

```
void draw() {
  if (mousePressed == true) {
    fill(0);
  } else {
    fill(255);
  }
  rect(25, 25, 50, 50);
}
```

### Fungsi mouseReleased()

Fungsi mouseReleased() akan dipanggil sewaktu tombol mouse dilepaskan. Berikut adalah contoh programnya.

### Program 9.4

```
// Click within the image to change
// the value of the rectangle

int value = 0;

void draw() {
  fill(value);
  rect(25, 25, 50, 50);
}

void mouseReleased() {
  if(value == 0) {
    value = 255;
  } else {
    value = 0;
  }
}
```

### Fungsi mouseMoved()

Fungsi mouseMoved() dipanggil setiap kali menggerakkan mouse dan tombol mouse tidak ditekan. Berikut adalah contoh programnya :

### Program 9.5

```
int value = 0;
void draw() {
  fill(value);
  rect(25, 25, 50, 50);
}
void mouseMoved() {
  value = value + 5;
  if (value > 255) {
    value = 0;
  }
}
```

## Modul Praktikum Grafika Komputer

---

### Fungsi mouseDragged()

Fungsi mouseDragged() disebut sekali setiap kali menggerakkan mouse dan tombol mouse ditekan. Sintak yang digunakan adalah :

```
void mouseDragged() {  
  statements  
}
```

Berikut adalah contoh programnya :

#### Program 9.6

```
int value = 0;  
  
void draw() {  
  fill(value);  
  rect(25, 25, 50, 50);  
}  
  
void mouseDragged()  
{  
  value = value + 5;  
  if (value > 255) {  
    value = 0;  
  }  
}
```

### Event keyboard

Selain menanggapi event mouse, Processing juga menanggapi event yang berasal dari keyboard serta menyimpan informasi dari keyboard ke dalam beberapa system variable. Event dari system variable dari keyboard adalah:

1. Tombol keyboard ditekan, direpson melalui fungsi keyPressed()
2. Tombol keyboard dilepas, ditanggapi melalui fungsi keyReleased()
3. keyPressed, bernilai true apabila ada tombol keyboar ditekan.
4. keyCode, berisi nilai yang menyatakan tombol fungsi yang ditekan, seperti: tombol ALT, tombol SHIFT dan sebagainya. Sebelum menggunakan variable ini ada baiknya diuji apakah variable key berisi CODED.
5. key, berisi nilai ASCII dari tombol yang ditekan.

## Modul Praktikum Grafika Komputer

---

Berikut contoh penggunaan Event Keyboard :

Program 9.7

```
void setup()
{
  size(400,300);
  background(255,0,0);
}
void draw()
{
  if(keyPressed)
  {
    if(key=='b' || key=='a')
    {
      fill(0);
    }
  }
  else
  {
    fill(255);
  }
  rect (25,25,50,50);
}
```

Program 9.8

```
//event keyboard_2 dengan tombol anak panah
void setup()
{
  size(400,200);
  background(0);
}
color warna=color(255);
void draw ()
{
  fill(warna);
  rect(25,25,50,50);
}

void keyPressed()
{
  if(key==CODED)
  {
    if(keyCode==LEFT)
    {
      warna=color(255,255,0);
    }
    else if(keyCode==RIGHT)
    {
      warna=color(0,255,0);
    }
  }
}
```

## Modul Praktikum Grafika Komputer

---

### Program 9.9

```
//event keyboard dengan sembarang tombol
void setup()
{
  size(400,200);
  background(255,0,0);
}
void draw ()
{
  if(keyPressed==true)
  {
    fill(0);
  }
  else
  {
    fill(255);
  }
  rect(25,25,50,50);
}
```

### TUGAS LATIHAN

Buat program jika menekan tombol keyboard D maka akan muncul animasi objek lingkaran dan jika menekan tombol keyboard K maka akan muncul animasi kotak, jika tidak menekan tombol keyboard apapun maka layar output dalam keadaan kosong.

Nah, sekarang anda lebih mengerti bagaimana bentuk bentuk grafik dalam computer disajikan.... Sulit juga bukan.... Maka dari itu, silahkan anda belajar lebih giat dan kerjakan tugas final berikut ini :

### TUGAS TAKE HOME

Buatlah sebuah animasi game watch dengan menggunakan tombol tombol sebagai pemicunya. Selamat berusaha.... ^\_^

## Modul Praktikum Grafika Komputer

---

Tgl.	TUGAS PRAKTIKUM 09	Ttd. Dosen :
Teknik Informatika		Nilai :