

**Modul Praktikum**

**Grafika Komputer**

Dosen Pengampu:

*Febi Eka Febriansyah. M.T.*

Penyusun:

*Nuha Hanifah Azmi*

*Devi Ranita*

*Fadhli Munadi Iman*

*Syifa Trianingsih*

*Tri Lestari*

Edisi 1 (2016)

Laboratorium Komputasi Dasar

Jurusan Ilmu Komputer

FMIPA Universitas Lampung

### **Deskripsi Mata Kuliah**

Grafika komputer adalah bagian dari ilmu komputer yang berkaitan dengan pembuatan dan manipulasi gambar (visual) secara digital. Dalam kuliah ini akan dibahas mengenai konsep grafika komputer yang menjadi dasar acuan konsep pembentukan model grafis pada komputer. Dengan konsep yang dikembangkan berdasarkan rumusan matematis dan gabungan metode-metode lainnya yang relevan dalam pembentukan grafis.

### **Tujuan Perkuliahan**

Agar mahasiswa dapat memahami konsep-konsep dan dasar grafika komputer beserta implementasinya.

### **Deskripsi Isi Perkuliahan**

Bahasan dalam praktikum ini mencakup konsep dasar sistem grafika komputer, pengantar web GL, objek grafik 2D, transformasi objek 2 dimensi, obyek grafik 3D, transformasi objek 3 dimensi, proyeksi, visualisasi 3 dimensi hingga animasi.

## Daftar Isi

<b>Daftar Isi</b> .....	iii
<b>Sistem Grafika Komputer</b> .....	5
1.1 Grafika Komputer .....	5
1.2 Sistem Grafika Komputer .....	5
<b>Sistem Grafika Komputer</b> .....	7
2.1 Dasar Teori.....	7
2.2 Program yang dibutuhkan .....	7
2.3 Langkah-Langkah Pengerjaan .....	7
<b>Pengantar Web GL</b> .....	9
3.1 Dasar Teori.....	9
3.2 Keuntungan .....	10
<b>Pengantar Web GL</b> .....	11
4.1 Dasar Teori.....	11
4.2 Program yang dibutuhkan .....	12
4.3 Langkah-Langkah Pengerjaan .....	12
<b>Output Primitif</b> .....	15
5.1 Dasar Teori.....	15
5.2 Program yang dibutuhkan .....	16
5.3 Langkah-Langkah Pengerjaan .....	16
<b>Atribut Output Primitif</b> .....	20
6.1 Dasar Teori .....	22
6.2 Program yang dibutuhkan .....	21
6.3 Langkah-Langkah Pengerjaan .....	21
<b>Transformasi 2 Dimensi</b> .....	22
7.1 Dasar Teori .....	22
7.2 Program yang dibutuhkan .....	22
7.3 Langkah-Langkah Pengerjaan .....	22
<b>3 Dimensi</b> .....	26
8.1. Dasar Teori.....	26
8.2 Program yang dibutuhkan .....	26
8.3 Langkah-Langkah Pengerjaan .....	27

<b>Proyeksi</b> .....	30
9.1    Dasar Teori.....	30
9.2    Program yang dibutuhkan .....	30
9.3    Langkah-Langkah Pengerjaan .....	31
<b>Transformasi 3 Dimensi</b> .....	32
10.1   Dasar Teori.....	32
10.2   Program yang dibutuhkan .....	32
10.3   Langkah-Langkah Pengerjaan .....	32
<b>Animasi</b> .....	38
11.1   Dasar Teori.....	38
11.2   Program yang dibutuhkan .....	38
11.3   Langkah-Langkah Pengerjaan .....	38

## Pertemuan1

### Sistem Grafika Komputer

<b>Tujuan Instruksional</b>	:Pengantar
Tujuan dari materi ini adalah mahasiswa dapat mengetahui apa itu sistem grafika komputer dan konsep dasarnya.	
<b>Kompetensi yang Diharapkan</b>	:
Mahasiswadiharapkan dapat memahami konsep dasar sistem grafika komputer.	
<b>Waktu Pertemuan</b>	: 100 Menit

Praktikum #1 berisi materi tentang pengertian dan konsep dasar dari sistem grafika komputer.

#### 1.1 Grafika Komputer

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.

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 pada 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.

#### 1.2 Sistem Grafika Komputer

Data yang banyak dapat diwakili dengan bagan. Dalam hal ini, user memasukkan sejumlah data, dan dengan prosedur tertentu komputer akan menampilkan bagan

yang diinginkan. Bagan yang dihasilkan ini merupakan gambar statis karena user tidak dapat berinteraksi dengan gambar yang ada pada layar komputer.

Cara ini sangat tidak memadai, karena ada kalanya user ingin berinteraksi langsung dengan gambar pada layar komputer. Sistem yang memungkinkan user berdialog dengan apa yang terlihat pada layar komputer disebut dengan sistem grafika komputer interaktif (*interactive komputer graphic*).

Dalam sistem interaktif, user bisa mengendalikan segala aspek gambar yang terlihat secara dinamis. Aspek-aspek tersebut terdiri dari isi gambar, format gambar, bentuk gambar, ukuran gambar dan warna gambar. Pengendalian secara dinamis dapat dilakukan dengan piranti-piranti seperti : keyboard, mouse, joystick, light pen, dll.

Keuntungan yang kita peroleh dari sistem interaktif adalah bahwa dengan mudah kita bisa menirukan atau mensimulasikan sesuatu kejadian dalam dunia nyata (*real world*) pada layar komputer.

### **Motion Dynamic dan Update Dynamic**

Berdasarkan cara pandang kita terhadap gambar yang ada pada layar, kita bisa membedakan apa yang dimaksud dengan *motion dynamic* dan *update dynamic*.

#### **a. Motion dynamic**

Pada dasarnya adalah cara pandang kita terhadap suatu obyek yang bergerak dan kita sebagai pengamat dalam keadaan diam atau obyek yang kita amati diam dan kita sebagai pengamat bisa bergerak bebas di sekeliling obyek tersebut. Contoh : *flight simulator*.

#### **b. Update dynamic**

Pada dasarnya berhubungan erat dengan perubahan sifat dari obyek yang sedang diamati. Sifat-sifat tersebut bisa berupa bentuk, warna atau sifat-sifat yang lain. Dengan menggunakan komputer, dengan mudah kita mensimulasikan tabrakan antara dua buah mobil dengan kecepatan yang bisa dibuat bervariasi. Dengan cara ini, pabrik mobil bisa mempelajari hasil simulasi dan menambahkan unsur-unsur yang diperlukan untuk lebih menjamin keselamatan pengemudi dan penumpang.

## Pertemuan 2

### Sistem Grafika Komputer

<b>Tujuan Instruksional</b>	:
Pokok bahasan ini melanjutkan materi sebelumnya mengenai sistem grafika komputer. Tujuan dari materi ini adalah mahasiswa dapat mengoperasikan text editor untuk pemrograman grafika komputer	
<b>Kompetensi yang Diharapkan</b>	:
Mahasiswa diharapkan dapat mengoperasikan text editor untuk pemrograman grafika komputer.	
<b>Waktu Pertemuan</b>	:100 menit

Praktikum #2 berisi materi tentang implementasi pemrograman grafika komputer.

#### 2.1 Dasar Teori

Citra pada grafika computer menggunakan elemen dasar grafik. Elemen-elemen ini memudahkan untuk menggambar bentuk objek pada layar monitor. Dalam grafika komputer terdapat 4 elemen dasar grafik yaitu titik, garis, bentuk segi, dan bentuk bundar.

#### 2.2 Program yang dibutuhkan

Program yang dibutuhkan dalam praktikum ini adalah text editor dan browser.

#### 2.3 Langkah-Langkah Pengerjaan

##### 1. Mempersiapkan layar

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

##### 2. Pewarnaan Latar

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 tuliskan program seperti berikut dan analisa outputnya

#### Program 2.1

```
// 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 2.2

```
// 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);
```

#### Program 2.3

```
// 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 2.4

```
// 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);
```



## Pertemuan 3

### Pengantar Web GL

<b>Tujuan Intruksional</b>	:	Pokok Bahasan ini mengenalkan tentang pengertian dan konsep dasar dari Web GL.
<b>Kompetensi Yang Diharapkan</b>	:	Mahasiswa diharapkan memahami konsep dasar dari Web GL.
<b>Waktu Pertemuan</b>	:	100 Menit

Praktikum #3 berisi materi tentang pengertian, konsep, dan dasar-dasar Web GL.

#### 3.1 Dasar Teori

WebGL (Web Graphics Library) adalah API JavaScript untuk rendering grafis 3D komputer interaktif dan grafis 2D dalam setiap web browser yang kompatibel tanpa menggunakan plug-in. Ini merupakan salah satu produk yang memanfaatkan elemen HTML5 yang didukung banyak pengembang browser, seperti Google (Chrome), Mozilla (Firefox), Apple (Safari).

Komponen aplikasi WebGL antara lain :

- Canvas : adalah tempat dimana komponen object akan diletakkan. Canvas merupakan elemen standar HTML5, dengan demikian dapat diakses dengan menggunakan Document Object Model (DOM) melalui JavaScript.
- Object : adalah entitas 3D yang membentuk bagian dari adegan itu.
- Lights : pencahayaan dibutuhkan dalam 3D
- Camera : untuk melihat dan mengeksplorasi adegan 3D dalam canvas yang berfungsi sebagai viewport.

#### Browser

Untuk mengakses konten WebGL, browser yang digunakan harus kompatibel. Tabel berikut menampilkan daftar browser yang mendukung WebGL :

## Browser web

Browser Nama	Versi	Mendukung
Saya nternet E Xplorer	11 dan di atas	dukungan lengkap
Google Chrome	39 dan di atas	dukungan lengkap
Safari	8	dukungan lengkap
Firefox	36 dan di atas	dukungan parsial
Opera	27 dan di atas	dukungan parsial

## Browser ponsel

Browser Nama	Versi	Mendukung
Chrome untuk Android	42	dukungan parsial
Browser Android	40	dukungan parsial
IOS Safari	8.3	dukungan lengkap
Opera Mini	8	Tidak mendukung
Blackberry Browser	10	dukungan lengkap
IE seluler	10	dukungan parsial

## 3.2 Keuntungan

Berikut adalah keuntungan dari menggunakan WebGL :

- JavaScript pemrograman - aplikasi WebGL ditulis dalam JavaScript
- Meningkatkan dukungan dengan browser mobile - WebGL juga mendukung browser mobile seperti iOS safari, Android Browser, dan Chrome untuk Android
- Open source - WebGL merupakan open source
- Tidak perlu untuk kompilasi - JavaScript adalah setengah-pemrograman dan setengah HTML komponen
- Manajemen memori otomatis - JavaScript mendukung manajemen memori otomatis
- Mudah untuk mengatur - Sejak WebGL terintegrasi dalam HTML 5, tidak ada kebutuhan untuk tambahan set up

## Petemuan 4

### Pengantar Web GL

**Tujuan Instruksional** :  
Pokok bahasan ini menjelaskan tentang struktur aplikasi Web GL dan implementasinya.

**Kompetensi yang Diharapkan** :  
Mahasiswa diharapkan dapat memahami struktur aplikasi Web GL serta mengimplementasikannya kedalam source code.

**Waktu Pertemuan** : 100 menit

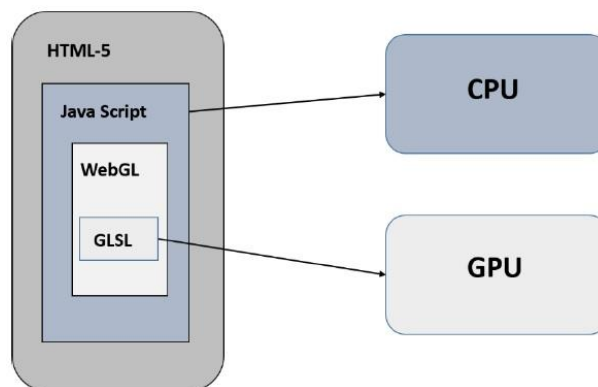
Praktikum #4 berisi materi tentang struktur aplikasi Web GL beserta implementasinya.

#### 4.1 Dasar Teori

##### Struktur Aplikasi WebGL

Kode aplikasi WebGL adalah kombinasi dari JavaScript dan OpenGL Shader Language.

- JavaScript diperlukan untuk berkomunikasi dengan CPU
- OpenGL Shader Bahasa diperlukan untuk berkomunikasi dengan GPU.



## 4.2 Program yang dibutuhkan

Program yang dibutuhkan dalam praktikum ini adalah text editor dan browser.

## 4.3 Langkah-Langkah Pengerjaan

Menggambar sebuah segitiga sederhana dengan koordinat 2D

```
<!doctype html>
<html>
  <body>
    <canvas width = "300" height = "300" id =
"my_Canvas"></canvas>

    <script>

      /* Step1: Prepare the canvas and get WebGL context
*/

      var canvas = document.getElementById('my_Canvas');
      var gl = canvas.getContext('experimental-webgl');

      /* Step2: Define the geometry and store it in
buffer objects */

      var vertices = [-0.5, 0.5, -0.5, -0.5, 0.0, -0.5,];

      // Create a new buffer object
      var vertex_buffer = gl.createBuffer();

      // Bind an empty array buffer to it
      gl.bindBuffer(gl.ARRAY_BUFFER, vertex_buffer);

      // Pass the vertices data to the buffer
      gl.bufferData(gl.ARRAY_BUFFER, new
Float32Array(vertices), gl.STATIC_DRAW);

      // Unbind the buffer
      gl.bindBuffer(gl.ARRAY_BUFFER, null);

      /* Step3: Create and compile Shader programs */

      // Vertex shader source code
      var vertCode =
```

```

        'attribute vec2 coordinates;' +
        'void main(void) {' + ' gl_Position =
vec4(coordinates,0.0, 1.0);' + '}';

//Create a vertex shader object
var vertShader = gl.createShader(gl.VERTEX_SHADER);

//Attach vertex shader source code
gl.shaderSource(vertShader, vertCode);

//Compile the vertex shader
gl.compileShader(vertShader);

//Fragment shader source code
var fragCode = 'void main(void) {' + 'gl_FragColor
= vec4(0.0, 0.0, 0.0, 0.1);' + '}';

// Create fragment shader object
var fragShader =
gl.createShader(gl.FRAGMENT_SHADER);

// Attach fragment shader source code
gl.shaderSource(fragShader, fragCode);

// Compile the fragment shader
gl.compileShader(fragShader);

// Create a shader program object to store combined
shader program
var shaderProgram = gl.createProgram();

// Attach a vertex shader
gl.attachShader(shaderProgram, vertShader);

// Attach a fragment shader
gl.attachShader(shaderProgram, fragShader);

// Link both programs
gl.linkProgram(shaderProgram);

// Use the combined shader program object
gl.useProgram(shaderProgram);

/* Step 4: Associate the shader programs to buffer
objects */

//Bind vertex buffer object
gl.bindBuffer(gl.ARRAY_BUFFER, vertex_buffer);

//Get the attribute location

```

```
        var coord = gl.getAttribLocation(shaderProgram,
"coordinates");

        //point an attribute to the currently bound VBO
gl.vertexAttribPointer(coord, 2, gl.FLOAT, false,
0, 0);

        //Enable the attribute
gl.enableVertexAttribArray(coord);

        /* Step5: Drawing the required object (triangle) */

        // Clear the canvas
gl.clearColor(0.5, 0.5, 0.5, 0.9);

        // Enable the depth test
gl.enable(gl.DEPTH_TEST);

        // Clear the color buffer bit
gl.clear(gl.COLOR_BUFFER_BIT);

        // Set the view port
gl.viewport(0,0,canvas.width,canvas.height);

        // Draw the triangle
gl.drawArrays(gl.TRIANGLES, 0, 3);

    </script>

</body>
</html>
```

## Petemuan 5

### Output Primitif

<p><b>TujuanInstruksional</b> :</p> <p>Pokok bahasan ini menjelaskan tentang bentuk output primitif / bentuk dasar.</p> <p><b>Kompetensi yang Diharapkan</b> :</p> <p>Mahasiswadiharapkan dapat memahami bentuk output primitif / bentuk dasar dan membuat objek baru dengan mengkombinasi bentuk dasar serta mengimplementasikan algoritma yang telah dipelajari.</p> <p><b>WaktuPertemuan</b> :100menit</p>
---

Praktikum #5 berisi materi tentang bentuk output primitif/bentuk dasar dalam grafika komputer serta implementasinya.

#### 5.1 Dasar Teori

##### Primitif Grafis

Secara umum algoritma grafis memiliki persamaan yaitu bagaimana menampilkan hasil. Primitif grafis yang umum dijelaskan pada tabel berikut :

OBJEK GRAFIS	PRIMITIF
Pixel (Dot)	Posisi (x,y), Warna
Garis (Line)	Posisi (x1,y1,x2,y2), Warna, Thickness, Pattern
Lingkaran (Circle)	Pusat(x,y), Radius, Warna, Thickness, Pattern
Ellipse	Pusat(x,y), Radius Horisonal/Vertikal, Warna, Thickness, Pattern
Kurva	Teratur/Tidak teratur (Bezier)
Character	Type, Slanted, Thickness, Color, dll

##### Algoritma Pembentukan Garis

Garis dibuat dengan menentukan dua endpoint atau posisi titik awal dan akhir dari suatu garis.

**Algoritma DDA** (Digital Differential Analyzer) adalah algoritma pembentukan

garis berdasarkan perhitungan dx maupun dy dengan menggunakan rumus  $dy = m \cdot dx$ .

Algoritma pembentukan garis DDA adalah sebagai berikut :

```

void lineDDA (int x0, int y0, int xEnd, int yEnd)
{
    int dx = xEnd - x0, dy = yEnd - y0, steps, k;
    float xIncrement, yIncrement, x = x0, y = y0;
    if (fabs (dx) > fabs (dy))
        steps = fabs (dx);
    else
        steps = fabs (dy);
    xIncrement = float (dx) / float (steps);
    yIncrement = float (dy) / float (steps);
    setPixel (round (x), round (y));
    for (k = 0; k < steps; k++) {
        x += xIncrement;
        y += yIncrement;
        setPixel (round (x), round (y));
    }
}

```

## 5.2 Program yang dibutuhkan

Perangkat yang dibutuhkan dalam praktikum ini adalah text editor dan browser.

## 5.3 Langkah-Langkah Pekerjaan

- 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 5.1

```

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);

```

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. Untuk lebih jelas, silahkan ketik program berikut ini:

#### Program 5.2

```

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. 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. Penggunaan `start` pada `arc` adalah posisi awal penggambaran dan `stop` adalah posisi akhir penggambaran busur.

#### Program 5.3

```
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 memberi warna area didalam 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 membuat segi empat standar dengan menggunakan perintah `rect(x,y,width,height)`, dimana `x` dan `y` adalah posisi awal sudut. Bentuk selanjutnya adalah segitiga yang dapat dibuat dengan perintah `triangle(x1,y1,x2,y2,x3,y3)`.

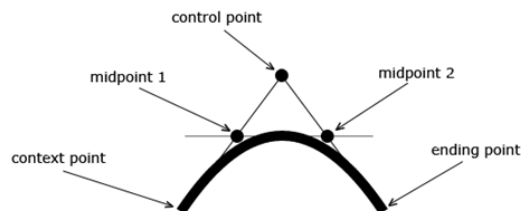
Program 5.4

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

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);
```

**Tugas:**

1. Buatlah kurva berikut:



2. Buatlah sebuah objek dengan mengombinasikan semua bentuk primitif yang telah dipelajari. Semenaarik mungkin dan selengkap-lengkapnya☺

## Petemuan 6

### Atribut Output Primitif

<p><b>Tujuan Instruksional</b> :</p> <p>Pokok bahasan ini adalah mempelajari tentang atribut output primitif.</p> <p><b>Kompetensi yang Diharapkan</b> :</p> <p>Mahasiswa diharapkan dapat memahami fungsi-fungsi dan atribut output primitif.</p> <p><b>Waktu Pertemuan</b> : 100 menit</p>
--

Praktikum #6 berisi materi tentang atribut output primitif, fungsi dan atribut titik, garis dan kurva.

#### 6.1 Dasar Teori

**Atribut Grafis** Atribut adalah semua parameter yang mempengaruhi bagaimana primitive grafis ditampilkan. Atribut dasar untuk titik adalah ukuran dan warna. Ukuran titik direpresentasikan sebagai beberapa piksel.

Atribut dasar pada garis adalah tebal, warna, dan tipe. Pada algoritma penggambaran garis, atribut tebal (width) dapat ditentukan seperti juga panjang dan spasi antar titik. Algoritma Raster menampilkan atribut ketebalan garis dengan mem-plot beberapa piksel sepanjang garis baik horizontal ( $m < 1$ ) maupun vertikal ( $m > 1$ ).

Atribut garis adalah tipe, warna dan ukuran garis. Garis memiliki 3 tipe standar yaitu solid, dashed dan dotted, tipe ini dapat dihasilkan dengan memodifikasi algoritma pembuatan garis standar yaitu DDA dan bresenham. Ukuran garis dihasilkan dengan memplot beberapa garis sepanjang jalur.

Pada system raster warna dapat dihasilkan dengan dua cara yaitu menyimpan langsung di frame buffer atau menyimpan di tabel warna. Pengisian area dibagi

menjadi 3 yaitu solid, hollow dan pattern

## 6.2 Program yang dibutuhkan

Perangkat yang dibutuhkan dalam praktikum ini adalah text editor dan browser.

## 6.3 Langkah-Langkah Pengerjaan

### Boundary Fill Algorithm

Pendekatan lain untuk area filling adalah memulai dari suatu titik didalam region dan mewarnai interior keluar batas. Jika boundary disepesifikasikan dengan satu warna, fill algorithm diproses per piksel sampai warna boundary didapat.

Algoritma ini menerima input koordinat titik interior (x,y), warna isi dan warna boundary.



Gambar 4.x ilustrasi boundary fill algorithm

```

Procedure BoundaryFill (x,y,fill,boundary : Integer); Var
    Current : integer;
Begin
    Current = getpixel(x,y);

    If (Current<>boundary) and (Current<>fill) then Begin

        setpixel (x,y,fill);
        Boundaryfill4(x+1,y,fill,boundary);
        Boundaryfill4 (x-1,y,fill, boundary);
        Boundaryfill4 (x,y+1,fill, boundary);
        Boundaryfill4 (x,y-1,fill, boundary);

    End;
End;

```

## Petemuan 7

### Transformasi 2 Dimensi

<b>Tujuan Instruksional</b>	:	Pokok bahasan ini untuk mengenalkan tentang transformasi 2 Dimensi dan memahami modelnya.
<b>Kompetensi yang Diharapkan</b>	:	Mahasiswa diharapkan dapat memahami model transformasi, pengertian dan konsep transformasi 2 Dimensi.
<b>Waktu Pertemuan</b>	:	100menit

Praktikum #7 berisi materi tentang pengenalan model transformasi 2 Dimensi.

#### 7.1 Dasar Teori

Transformasi adalah perubahan bentuk. Transformasi diperlukan untuk mengubah ( *transform* ) posisi suatu objek dari tempat asal ke posisi elemen grafik, transformasi juga diperlukan untuk memutar posisi suatu objek pada titik pusat, mengubah ukuran objek dan menarik sebagian objek sehingga tampak terdistorsi.

Bentuk-bentuk transformasi tersebut secara umum adalah sebagai berikut :

1. *Translation* ( mengeser )
2. *Scale* ( merubah ukuran )
3. *Rotation* ( memutar )

#### 7.2 Program yang dibutuhkan

Perangkat yang dibutuhkan dalam praktikum ini adalah text editor dan browser.

#### 7.3 Langkah-Langkah Pengerjaan

**Transformasi objek dan transformasi system koordinat.**

Program 7.1

```

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

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

```

Program 7.2

```

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();
}

```

**Rotasi 45 derajat clockwise.**

Program 7.3

```

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


```



<pre> pushMatrix();   rotate(radians(45));   fill(0);   rect(40, 40, 40, 40);   popMatrix(); } </pre>	
---	--

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 :

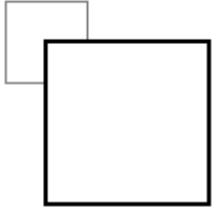
Program 7.4	
<pre> void setup() {   size(200, 200);   background(255);   smooth();   fill(192);   noStroke();   rect(40, 40, 40, 40);    pushMatrix();   // move the origin to the pivot point   translate(40, 40);    // then pivot the grid   rotate(radians(45));    // and draw the square at the origin </pre>	



<pre>fill(0); rect(0, 0, 40, 40); popMatrix(); }</pre>	
--	--

### Scaling

Transformasi yang terakhir adalah scaling, yang mana prosesnya adalah mengubah ukuran objek baik membesar atau pun mengecil. Lihat contoh program berikut yang mengubah ukuran objek menjadi dua kali lebih besar dari originalnya.

Program 7.5	
<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>	

### Tugas

1. Buatlah sebuah objek seperti berikut :



2. Implementasikan salah satu model transformasi yang telah kalian pelajari pada objek diatas☺

## Petemuan 8

### 3 Dimensi

<b>Tujuan Instruksional</b>	:	Pokok bahasan ini untuk mengenalkan tentang konsep 3 Dimensi.
<b>Kompetensi yang Diharapkan</b>	:	Mahasiswa diharapkan dapat memahami konsep 3 Dimensi dan mengimplementasikannya sebagai animasi sederhana.
<b>Waktu Pertemuan</b>	:	100 menit

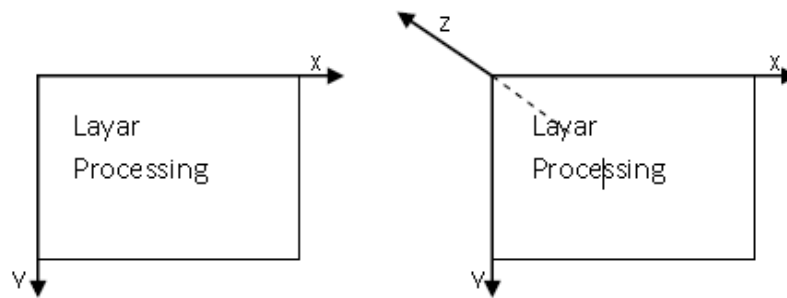
Praktikum #8 berisi materi tentang pengenalan model 3 Dimensi.

#### 8.1. Dasar Teori

Grafik 3Dimensi merupakan perkembangan dari grafik 2Dimensi. Didalam grafika komputer, 3D merupakan bentuk grafik yang menggunakan representasi data geometri tiga dimensi. Proses pembuatan grafik komputer 3D dapat dibagi ke dalam tiga fase, yaitu 3D modeling yang mendeskripsikan bentuk dari sebuah objek, layout dan animation yang mendeskripsikan gerakan dan tata letak sebuah objek, dan 3D rendering yang memproduksi image dari objek tersebut. Istilah atau Pengertian Grafik 3D adalah sebuah gambar, garis, lengkungan, dan sebagainya yang memiliki titik-titik yang menghubungkan menjadi sebuah bentuk 3D.

Dalam membuat objek 3D dapat menggunakan tipe file html. Scriptnya dapat dibuat di berbagai tools seperti Geany, WordPad, dll. Untuk melihat hasil 3D, file dapat dibuka dengan menggunakan browser (drag/double click file html).

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 8.1 Orientasi sumbu 2 dimensi dan 3 dimensi.



gambar 8.1 Orientasi sumbu 2 dimensi dan 3 dimensi.

## 8.2 Program yang dibutuhkan

Perangkat yang dibutuhkan dalam praktikum ini adalah text editor dan browser.

## 8.3 Langkah-Langkah Pengerjaan

### Contoh script objek 3D

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">

<html>

<head>

<title>MEMBUAT KUBUS</title>

<style type="text/css">

body{

padding:30px 0 0 50px;

}

.box1, .box2, .box3, .box4, .box5, .box6{

opacity:.8;

background:#000;
```

```
position:absolute;

margin-left:31px;

-moz-transform: scale(1) rotate(0deg) translate(0px, 0px)
skew(40deg, 0deg);

-webkit-transform: scale(1) rotate(0deg) translate(0px,
0px) skew(40deg, 0deg);

-o-transform: scale(1) rotate(0deg) translate(0px, 0px)
skew(40deg, 0deg);

-ms-transform: scale(1) rotate(0deg) translate(0px, 0px)
skew(40deg, 0deg);

transform: scale(1) rotate(0deg) translate(0px, 0px)
skew(40deg, 0deg);
}

.box2, .box5{

width:62px;

height:199px;

position:absolute;

margin-top:38px;

-moz-transform: scale(1) rotate(0deg) translate(0px, 0px)
skew(0deg, 50deg);

-webkit-transform: scale(1) rotate(0deg) translate(0px,
0px) skew(0deg, 50deg);

-o-transform: scale(1) rotate(0deg) translate(0px, 0px)
skew(0deg, 50deg);

-ms-transform: scale(1) rotate(0deg) translate(0px, 0px)
skew(0deg, 50deg);

transform: scale(1) rotate(0deg) translate(0px, 0px)
skew(0deg, 50deg);
}

.box3, .box4{

width:199px;
```

```
.box4{
margin:75px 0 0 63px;
border:0;
}

.box5{
margin-left:200px;
border:0;
}

.box6{
margin-top:200px;
border:0;
}

</style>
</head>
<body>
<div class="box1"></div>
<div class="box2"></div>
<div class="box3"></div>
<div class="box4"></div>
<div class="box5"></div>
<div class="box6"></div>

</body>
</html>
```

## Petemuan 9

### Proyeksi

<b>Tujuan Instruksional</b>	:	Pokok bahasan ini mempelajari tentang proyeksi.		
<b>Kompetensi yang Diharapkan</b>	:	Mahasiswadiharapkan dapat memahami proyeksi dan mengimplementasikannya.		
<b>Waktu Pertemuan</b>	:	100menit		

Praktikum #9 berisi materi tentang proyeksi dan implementasinya.

#### 9.1 Dasar Teori

Proyeksi merupakan salah satu jenis transformasi, yaitu transformasi koordinat. Proyeksi merupakan proses dimana informasi tentang titik di sebuah sistem koordinat berdimensi  $n$  dipindahkan ke system koordinat berdimensi kurang dari  $n$ . sebagai contoh, titik  $(x,y,z)$  yang berada di sistem koordinat berdimensi 3 dipindahkan ke sistem koordinat yang berdimensi 2 sehingga menjadi  $(x,y)$ , transformasi tersebut tentunya harus memperhitungkan pengaruh  $z$  terhadap titik  $(x,y)$ . Proyeksi dapat dilakukan terhadap bidang datar (*planar*) atau kebidang kurva. Bab ini hanya akan membahas proyeksi ke bidang *planar* atau disebut sebagai *planar geometric projections*.

#### 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.

## 9.2 Program yang dibutuhkan

Perangkat yang dibutuhkan dalam praktikum ini adalah text editor dan browser.

## 9.3 Langkah-Langkah Pekerjaan

### Model Prespektif

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 9.1

```
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);
```

## Petemuan 10

### Transformasi 3 Dimensi

<p><b>Tujuan Instruksional</b> :</p> <p>Pokok bahasan ini mengenalkan mengenai materi transformasi 3Dimensi.</p> <p><b>Kompetensi yang Diharapkan</b> :</p> <p>Mahasiswadiharapkan dapat memahami model transformasi 3 Dimensi.</p> <p><b>WaktuPertemuan</b> :100menit</p>
--

Praktikum #10 berisi materi tentang transformasi 3 Dimensi dimana melanjutkan materi pada bab sebelum-sebelumnya telah dibahas mengenai transformasi 2 Dimensi.

#### 10.1 Dasar Teori

Transformasi merupakan suatu metode untuk mengubah lokasi suatu titik pembentuk objek, sehingga objek tersebut mengalami perubahan. Transformasi 3 Dimensi merupakan transformasi yang terjadi pada objek 3 dimensi.

#### 10.2 Program yang dibutuhkan

Perangkat yang dibutuhkan dalam praktikum ini adalah text editor dan browser.

#### 10.3 Langkah-Langkah Pekerjaan

##### Menggambar Rotasi Kubus 3D

```

<!doctype html>
<html>
  <body>
    <canvas width = "570" height = "570" id =
"my_Canvas"></canvas>

    <script>

      /*===== Creating a canvas
=====*/
      var canvas = document.getElementById('my_Canvas');

```



```

    gl = canvas.getContext('experimental-webgl');

    /*===== Defining and storing the geometry =====*/

    var vertices = [
        -1,-1,-1, 1,-1,-1, 1, 1,-1, -1, 1,-1,
        -1,-1, 1, 1,-1, 1, 1, 1, 1, -1, 1, 1,
        -1,-1,-1, -1, 1,-1, -1, 1, 1, -1,-1, 1,
        1,-1,-1, 1, 1,-1, 1, 1, 1, 1,-1, 1,
        -1,-1,-1, -1,-1, 1, 1,-1, 1, 1,-1,-1,
        -1, 1,-1, -1, 1, 1, 1, 1, 1, 1, 1,-1,
    ];

    var colors = [
        5,3,7, 5,3,7, 5,3,7, 5,3,7,
        1,1,3, 1,1,3, 1,1,3, 1,1,3,
        0,0,1, 0,0,1, 0,0,1, 0,0,1,
        1,0,0, 1,0,0, 1,0,0, 1,0,0,
        1,1,0, 1,1,0, 1,1,0, 1,1,0,
        0,1,0, 0,1,0, 0,1,0, 0,1,0
    ];

    var indices = [
        0,1,2, 0,2,3, 4,5,6, 4,6,7,
        8,9,10, 8,10,11, 12,13,14, 12,14,15,
        16,17,18, 16,18,19, 20,21,22, 20,22,23
    ];

    // Create and store data into vertex buffer
    var vertex_buffer = gl.createBuffer ();
    gl.bindBuffer(gl.ARRAY_BUFFER, vertex_buffer);
    gl.bufferData(gl.ARRAY_BUFFER, new
Float32Array(vertices), gl.STATIC_DRAW);

    // Create and store data into color buffer
    var color_buffer = gl.createBuffer ();
    gl.bindBuffer(gl.ARRAY_BUFFER, color_buffer);
    gl.bufferData(gl.ARRAY_BUFFER, new
Float32Array(colors), gl.STATIC_DRAW);

    // Create and store data into index buffer
    var index_buffer = gl.createBuffer ();
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER,
index_buffer);

```

```

        gl.bufferData(gl.ELEMENT_ARRAY_BUFFER, new
        Uint16Array(indices), gl.STATIC_DRAW);

        /*===== Shaders =====*/

        var vertCode = 'attribute vec3 position;'+
            'uniform mat4 Pmatrix;'+
            'uniform mat4 Vmatrix;'+
            'uniform mat4 Mmatrix;'+
            'attribute vec3 color;'+//the color of the point
            'varying vec3 vColor;'+

            'void main(void) { '+//pre-built function
                'gl_Position =
                Pmatrix*Vmatrix*Mmatrix*vec4(position, 1.);'+
                'vColor = color;'+
            '}';

        var fragCode = 'precision mediump float;'+
            'varying vec3 vColor;'+
            'void main(void) {'+
                'gl_FragColor = vec4(vColor, 1.);'+
            '}';

        var vertShader = gl.createShader(gl.VERTEX_SHADER);
        gl.shaderSource(vertShader, vertCode);
        gl.compileShader(vertShader);

        var fragShader =
        gl.createShader(gl.FRAGMENT_SHADER);
        gl.shaderSource(fragShader, fragCode);
        gl.compileShader(fragShader);

        var shaderProgram = gl.createProgram();
        gl.attachShader(shaderProgram, vertShader);
        gl.attachShader(shaderProgram, fragShader);
        gl.linkProgram(shaderProgram);

        /* ===== Associating attributes to vertex shader
        =====*/
        var Pmatrix = gl.getUniformLocation(shaderProgram,
        "Pmatrix");
        var Vmatrix = gl.getUniformLocation(shaderProgram,
        "Vmatrix");

```

```

        var Mmatrix = gl.getUniformLocation(shaderProgram,
"Matrix");

        gl.bindBuffer(gl.ARRAY_BUFFER, vertex_buffer);
        var position = gl.getAttribLocation(shaderProgram,
"position");
        gl.vertexAttribPointer(position, 3, gl.FLOAT,
false,0,0) ;

        // Position
        gl.enableVertexAttribArray(position);
        gl.bindBuffer(gl.ARRAY_BUFFER, color_buffer);
        var color = gl.getAttribLocation(shaderProgram,
"color");
        gl.vertexAttribPointer(color, 3, gl.FLOAT,
false,0,0) ;

        // Color
        gl.enableVertexAttribArray(color);
        gl.useProgram(shaderProgram);

/*===== MATRIX =====*/

        function get_projection(angle, a, zMin, zMax) {
            var ang =
Math.tan((angle*.5)*Math.PI/180); //angle*.5
            return [
                0.5/ang, 0 , 0, 0,
                0, 0.5*a/ang, 0, 0,
                0, 0, -(zMax+zMin)/(zMax-zMin), -1,
                0, 0, (-2*zMax*zMin)/(zMax-zMin), 0
            ];
        }

        var proj_matrix = get_projection(40,
canvas.width/canvas.height, 1, 100);

        var mov_matrix = [1,0,0,0, 0,1,0,0, 0,0,1,0,
0,0,0,1];
        var view_matrix = [1,0,0,0, 0,1,0,0, 0,0,1,0,
0,0,0,1];

        // translating z
        view_matrix[14] = view_matrix[14]-6;//zoom

```

```
/*===== Rotation =====*/

function rotateZ(m, angle) {
    var c = Math.cos(angle);
    var s = Math.sin(angle);
    var mv0 = m[0], mv4 = m[4], mv8 = m[8];

    m[0] = c*m[0]-s*m[1];
    m[4] = c*m[4]-s*m[5];
    m[8] = c*m[8]-s*m[9];

    m[1]=c*m[1]+s*mv0;
    m[5]=c*m[5]+s*mv4;
    m[9]=c*m[9]+s*mv8;
}

function rotateX(m, angle) {
    var c = Math.cos(angle);
    var s = Math.sin(angle);
    var mv1 = m[1], mv5 = m[5], mv9 = m[9];

    m[1] = m[1]*c-m[2]*s;
    m[5] = m[5]*c-m[6]*s;
    m[9] = m[9]*c-m[10]*s;

    m[2] = m[2]*c+mv1*s;
    m[6] = m[6]*c+mv5*s;
    m[10] = m[10]*c+mv9*s;
}

function rotateY(m, angle) {
    var c = Math.cos(angle);
    var s = Math.sin(angle);
    var mv0 = m[0], mv4 = m[4], mv8 = m[8];

    m[0] = c*m[0]+s*m[2];
    m[4] = c*m[4]+s*m[6];
    m[8] = c*m[8]+s*m[10];

    m[2] = c*m[2]-s*mv0;
    m[6] = c*m[6]-s*mv4;
    m[10] = c*m[10]-s*mv8;
}
```

```
/*===== Drawing =====*/
var time_old = 0;

var animate = function(time) {

    var dt = time-time_old;
    rotateZ(mov_matrix, dt*0.005); //time
    rotateY(mov_matrix, dt*0.002);
    rotateX(mov_matrix, dt*0.003);
    time_old = time;

    gl.enable(gl.DEPTH_TEST);
    gl.depthFunc(gl.LEQUAL);
    gl.clearColor(0.5, 0.5, 0.5, 0.9);
    gl.clearDepth(1.0);

    gl.viewport(0.0, 0.0, canvas.width,
canvas.height);
    gl.clear(gl.COLOR_BUFFER_BIT |
gl.DEPTH_BUFFER_BIT);
    gl.uniformMatrix4fv(Pmatrix, false,
proj_matrix);
    gl.uniformMatrix4fv(Vmatrix, false,
view_matrix);
    gl.uniformMatrix4fv(Mmatrix, false, mov_matrix);
    gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER,
index_buffer);
    gl.drawElements(gl.TRIANGLES, indices.length,
gl.UNSIGNED_SHORT, 0);

    window.requestAnimationFrame(animate);
}
animate(0);

</script>

</body>
</html>
```

## Petemuan 11

### Animasi

<p><b>Tujuan Instruksional</b> :</p> <p>Pokok bahasan ini menjelaskan tentang pengenalan animasi, konsepnya dan contoh implementasinya.</p> <p><b>Kompetensi yang Diharapkan</b> :</p> <p>Mahasiswa diharapkan dapat memahami konsep animasi beserta implementasinya.</p> <p><b>WaktuPertemuan</b> :100menit</p>
--

Praktikum #11 berisi materi tentang pengenalan dan implementasi konsep dari animasi.

#### 11.1 Dasar Teori

**Animasi** adalah gambar bergerak berbentuk dari sekumpulan objek (gambar) yang disusun secara beraturan mengikuti alur pergerakan yang telah ditentukan pada setiap pertambahan hitungan waktu yang terjadi.

- Animasi 2 Dimensi adalah animasi yang paling sering kita jumpai, animasi ini biasa disebut dengan kartun. Animasi ini berbasis pada konsep gambar 2 dimensi.
- Animasi 3 Dimensi merupakan perkembangan dari animasi 2 dimensi. Dengan berdasarkan pada konsep gambar 3 dimensi sehingga menghasilkan animasi yang lebih realistis, detail dan nyata karena hampir menyerupai bentuk aslinya.

#### 11.2 Program yang dibutuhkan

Perangkat yang dibutuhkan dalam praktikum ini adalah text editor dan browser.

#### 11.3 Langkah-Langkah Pengerjaan

Membuat animasi garis
<pre>void setup() {</pre>

```

        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
    }

```

#### 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
}

```

Menggunakan HTML5 canvas:

Buat sebuah kotak yang akan dibuat animasi. Agar kotak tersebut bergeser ke kanan adalah dengan menghapus kanvas tersebut, membuat ulang kotak tapi kali ini kotaknya sedikit ke kanan, menghapus kanvas lagi, membuat kotak lagi, dst. Untuk mendapatkan efek seperti itu kita harus memasukkan dalam function dan memanggil function itu terus-menerus. Ketika fungsi animate() dijalankan, canvas dihapus, dan membuat timeout untuk menjalankan fungsi animate() tersebut setelah 30 millisecond, jadi karena nilai x bertambah, posisi kotak akan terus ke kanan.

```

window.onload=function() {
  c=document.getElementById("canvas");
  a=c.getContext('2d');
  x=0;
  y=100;

  animate();
  function animate() {
    a.clearRect(0,0,c.width,c.height);
    a.fillRect(x,y,50,50);
    x+=2;
    setTimeout(animate,30);
  }
};

```

Source code :

```

55c = document.getElementById("canvas");a = c.getContext('2d');
kotak=[]; //membuat array yang berisi kotak-kotak yang ada

function animate(sikotak) {
  x=sikotak.x; //memasukkan posisi x si kotak ke variable "x"
  y=sikotak.y; //sama dengan yang x
  a.clearRect(x,y-2,10,10); //menghapus "kotak sebelumnya"
  a.fillRect(x,y,10,10); //membuat kotak dengan posisi x dan ynya
  sikotak.y+=2; //mengubah posisi kotaknya agar turun
  setTimeout(function(){animate(sikotak);},30); //memanggil fungsi ini lagi
}

c.onclick=function(e) { //ketika canvas diklik
  kotak.push({x:e.pageX-45,y:e.pageY-45}); //masukkan daftar baru kedalam
array kotak yang berisi posisi x dan y kotak
  animate(kotak[kotak.length-1]); //memanggil fungsi animate dengan parameter
isi dari array "kotak" yang terbaru
};

```