Modul Praktikum

Grafika Komputer

Disusun oleh: Riri Safitri, S.Si, MT

2019

Teknik Informatika

Universitas Al Azhar Indonesia

1/1/2019

Daftar Isi

[MODUL 1: Instalasi Open GL 1](#_Toc18334080)

[MODUL 2: Hello World 2](#_Toc18334081)

[MODUL 3: Open GL Primitives 5](#_Toc18334082)

[MODUL 4: Ploting Curva 10](#_Toc18334083)

[MODUL 5: Plot Fungsi Parametrik 14](#_Toc18334084)

[MODUL 6: 3D Vertex 17](#_Toc18334085)

[MODUL 7: Transformasi 20](#_Toc18334086)

[MODUL 8: Lighting 25](#_Toc18334087)

[MODUL 9: Texture Mapping 28](#_Toc18334088)

[MODUL 10: Games Sederhana 33](#_Toc18334089)

[Referensi 37](#_Toc18334090)

[Format Laporan Praktikum 38](#_Toc18334091)

# MODUL 1: Instalasi Open GL

**TUJUAN**

1. Mahasiswa mengetahui tools yang digunakan dalam Grafika Komputer
2. Mahasiswa mampu melakukan instalasi Open GL

**TOOLS**:

1. Python : <http://www.codeblocks.org/downloads/26>
2. Python OpenGL : sourceforge.net

**Cara Install :**

1. Install python python-3.7.3-amd64 untuk Windows 10 64 bit, sesuaikan dengan PC/ Laptop masing-masing
2. Kemudian buka Command Prompt untuk menambahkan library OpenGL
   1. curl [https://bootstrap.pypa.io/get-pip.py -o get-pip.py](https://bootstrap.pypa.io/get-pip.py%20-o%20get-pip.py)
   2. Lanjutkan untuk mendapatkan pip.py dengan C:\blabla>python get-pip.py
   3. Lanjutkan dengan install library openGL dengan C:\blabla>pip install pyopengl
   4. Lanjutkan dengan install pyGame dengan C:\blabla>pip install pygame
3. Kemudian mulai membuat program OpenGL dengan notepad atau IDE yang sudah diinstal di PC atau laptop masing-masing.

**Testing Program**

1. Buka Notepad.
2. Ketikkan program di Latihan Modul 2.
3. Run program di cmd dengan memanggil nama file program.

# MODUL 2: Hello World

**TUJUAN**

1. Membuat OpenGL-based window
2. Menampilkan “Hello World” dengan OpenGL
3. Mengenal fungsi-fungsi untuk mengatur vertex data

**TEORI**

Tujuan dari praktikum modul ini adalah untuk membuat OpenGL-based window. Ada banyak cara untuk membuat dan memperlihatkan suatu window dalam berbagai sistem window. Salah satunya adalah dengan menggunakan OpenGL Utility Toolkit (GLUT). GLUT menyediakan banyak fungsi yang dapat membuat *window* dengan cara yang independen terhadap sistem operasinya. Hal ini berarti program yang dibuat dengan GLUT dapat beroperasi pada sistem *windowing* yang berbeda tanpa merubah code secara manual.

Siapkan new folder 🡪 **Praktikum\_Grafikom**/**Modul2**. Ketiklah kode program berikut menggunakan Notepad kemudian simpan dengan nama **Latihan2\_1.py**

**Latihan2\_1. GLUT “Hello World” & Gambar Segiempat**

import pygame

from pygame.locals import \*

from OpenGL.GL import \*

from OpenGL.GLU import \*

def Cube():

glBegin(GL\_QUADS)

glVertex2f(-0.5, -0.5);

glVertex2f(-0.5, 0.5);

glVertex2f(0.5, 0.5);

glVertex2f(0.5, -0.5);

glEnd()

def main():

pygame.init()

display = (800,600)

pygame.display.set\_caption('Hello World..!!')

pygame.display.set\_mode(display, DOUBLEBUF|OPENGL)

gluPerspective(45, (display[0]/display[1]), 0.1, 50.0)

glTranslatef(0.0,0.0, -5)

while True:

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

quit()

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT)

Cube()

pygame.display.flip()

pygame.time.wait(10)

main()

**Tugas**: Coba periksa dimensi window seperti koordinat titik tengah, titik kiri atas, titik kiri bawah, titik kanan atas dan titik kanan bawah dengan merubah-rubah parameter fungsi **glVertex2f()**.

Diperoleh Koordinat:

* Titik Tengah ( ……,…..)
* Titik Kiri Atas ( ……,…..), Titik Kiri Bawah ( ……,…..)
* Titik Kanan Atas ( ……,…..), Titik Kanan Bawah ( ……,…..)

ScreenShoot Output Program:

Buat *project* baru dengan nama **Latihan2\_2**, Tambahkan fungsi init pada program diatas, simpan dengan nama **latihan2\_2**

**Latihan2\_2: fungsi init()**

def init():

glClearColor (1.0, 0.0, 0.0, 1.0); ## background colour

glColor3f(0.0, 1.0, 1.0); ## object colour

glMatrixMode (GL\_PROJECTION);

glLoadIdentity (); ##load identity

glOrtho(-1.0,1.0,-1.0,1.0,-1.0,1.0); ##projection

def main():

pygame.init()

display = (800,600)

pygame.display.set\_caption('Hello World..!!')

pygame.display.set\_mode(display, DOUBLEBUF|OPENGL)

gluPerspective(45, (display[0]/display[1]), 0.1, 50.0)

glTranslatef(0.0,0.0, -5)

while True:

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

quit()

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT)

Cube()

pygame.display.flip()

pygame.time.wait(10)

init()

main()

Tugas:

1. Ganti warna background dengan mengganti nilai glClearColor:
   * hiijau 🡪 glClearColor (……, …..., ……)
   * Biru 🡪 glClearColor (….., ….., ……)
   * Kuning 🡪 glClearColor (….., ….., ……)
2. Ganti warna object (segi empat) dengan mengganti nilai dalam glColor3f
   * Merah 🡪 glColor3f (……, …..., ……)
   * Putih 🡪 glColor3f (….., ….., ……)
   * Abu-Abu 🡪 glColor3f (….., ….., ……)
3. Perbesar ukuran window dengan mengganti nilai pada glutInitWindowSize
4. Ganti nilai pada glutInitWindowPosition, lihat dan perhatikan perubahan yang terjadi pada window, dan jelaskan:
5. Buatlah program untuk membuat segitiga sederhana, simpan dengan nama **tugas2\_1**:
6. Kesimpulan dari latihan pada Modul 2

# MODUL 3: Open GL Primitives

**TUJUAN**

1. Mengenal komponen dasar geometric
2. Membuat obyek 2D menggunakan geometric primitives

**TEORI**

Pada tutorial 01 telah diberikan contoh program untuk merepresentasikan model obyek segiempat 2D. OpenGL memiliki beberapa komponen dasar untuk merepresentasikan suatu obyek. Komponen dasar tersebut disebut **OpenGL Geometric primitives**. Gambar1 berikut menggambarkan semua OpenGL Geometric primitives yang mungkin.

**GL\_QUAD\_STRIP**

**GL\_POLYGON**

**GL\_TRIANGLE\_STRIP**

**GL\_TRIANGLE\_FAN**

**GL\_POINTS**

**GL\_LINES**

**GL\_LINE\_LOOP**

**GL\_LINE\_STRIP**

**GL\_TRIANGLES**

**GL\_QUADS**

Gambar 3.1. OpenGL Primitives

Setiap obyek harus dimodelkan sebagai kombinasi dari komponen-komponen dasar tersebut. Sebagai contoh, obyek segiempat pada latihan01 dimodelkan dengan menggunakan komponen dasar **GL\_POLYGON**. Obyek tersebut dapat pula dimodelkan dengan komponen dasar **GL\_TRIANGLES** atau **GL\_QUAD**.

**glVertex3fv( *v* )**

***Number of***

***components***

**2 - (x,y)**

**3 - (x,y,z)**

**4 - (x,y,z,w)**

***Data Type***

**b - byte**

**ub - unsigned byte**

**s - short**

**us - unsigned short**

**i - int**

**ui - unsigned int**

**f - float**

**d - double**

***Vector***

**omit “v” for**

**scalar form**

**glVertex2f( x, y )**

Gambar 3.2 Struktur Sintaks koordinat Open GL

Dalam OpenGL, menggambar geometric primitives selalu dilakukan di antara fungsi

**glBegin(PRIMITIVES)**

**// Fungsi Menggambar Primitives di sini**

**glEnd()**

Ketiklah kode program berikut menggunakan notepad, simpan dengan nama **Latihan3\_1.py**

**Latihan 3\_1. Segitiga**

import pygame

from pygame.locals import \*

from OpenGL.GL import \*

from OpenGL.GLU import \*

def Triangle():

glBegin(GL\_TRIANGLES)

glVertex2f(-0.5, -0.5);

glVertex2f(-0.5, 0.5);

glVertex2f(0.5, 0.5);

glEnd()

def init():

glClearColor (1.0, 0.0, 0.0, 1.0); ## background colour

glColor3f(0.0, 1.0, 1.0); ## object colour

def main():

pygame.init()

display = (800,600)

pygame.display.set\_caption(Triangles')

pygame.display.set\_mode(display, DOUBLEBUF|OPENGL)

gluPerspective(45, (display[0]/display[1]), 0.1, 50.0)

glTranslatef(0.0,0.0, -5)

while True:

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

quit()

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT)

Triangle()

pygame.display.flip()

pygame.time.wait(10)

init()

main()

**Tugas**: Perhatikan urutan dari vertex untuk setiap jenis OpenGL Geometric Primitive, lampirkan output program.

Buat *project* baru dengan nama **Latihan3\_2**, Tambahkan fungsi init pada program diatas, simpan dengan nama **latihan3\_2.c**

**Latihan3\_2: Segi Delapan**

import pygame

from pygame.locals import \*

from OpenGL.GL import \*

from OpenGL.GLU import \*

def Octagonal():

glBegin(GL\_POLYGON)

glColor3f(0, 1, 0);

glVertex2f(-0.5, -0.5);

glVertex2f(-0.75, 0);

glVertex2f(-0.5, 0.5);

glVertex2f(0, 0.75);

glVertex2f(0.5, 0.5);

glVertex2f(0.75, 0);

glVertex2f(0.5, -0.5);

glVertex2f(0,-0.75);

glEnd()

def init():

glClearColor (0.0, 0.0, 1.0, 1.0); ## background colour

glColor3f(1.0, 1.0, 0.0); ## object colour

def main():

pygame.init()

display = (800,600)

pygame.display.set\_caption('Triangles')

pygame.display.set\_mode(display, DOUBLEBUF|OPENGL)

gluPerspective(45, (display[0]/display[1]), 0.1, 50.0)

glTranslatef(0.0,0.0, -5)

while True:

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

quit()

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT)

Octagonal()

pygame.display.flip()

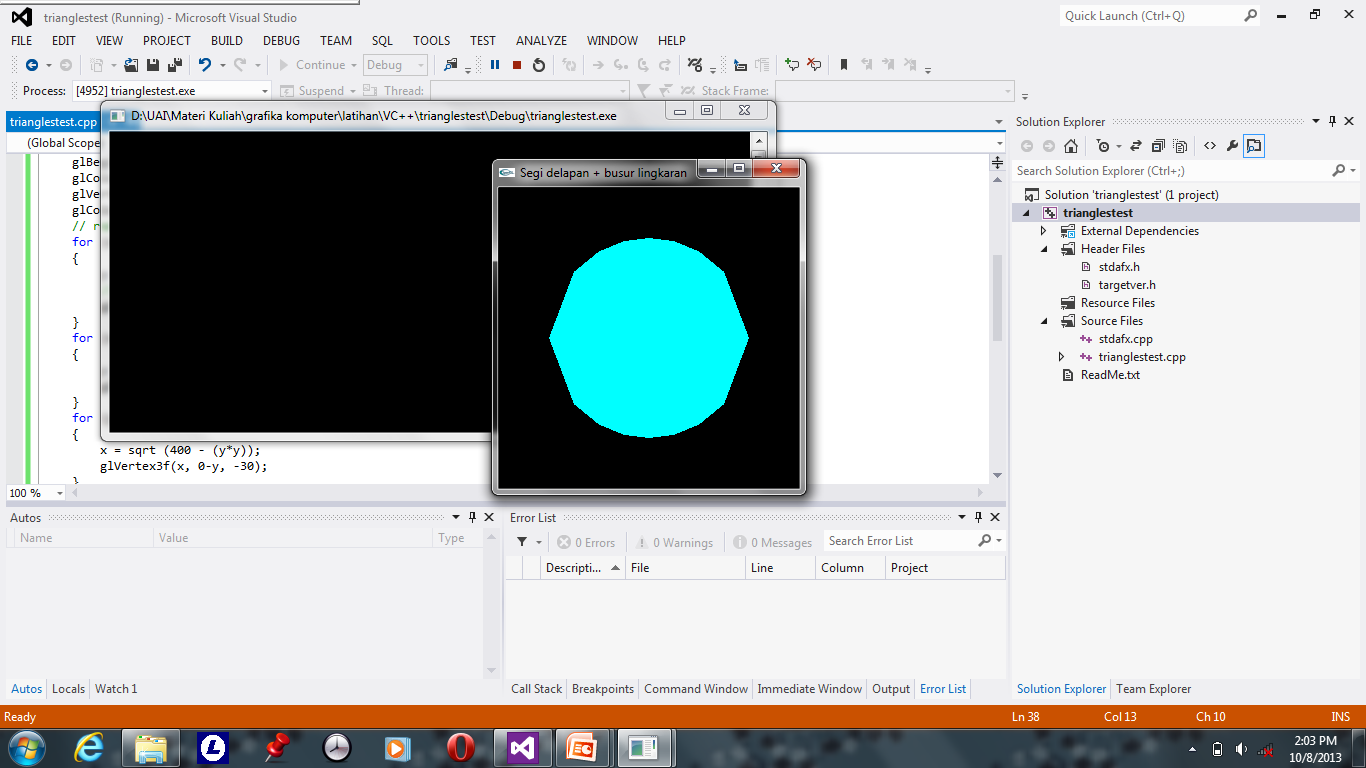
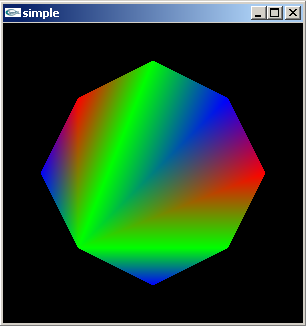
pygame.time.wait(10)

init()

main()

**Tugas:**

1. Buatlah program untuk membuat jenis openGL primitives yang lain, dengan mengganti PRIMITIVES (GL\_TRIANGLES) dengan primitives yang lain,yaitu:
   * GL\_LINES
   * GL\_TRIANGLESTRIP
   * GL\_QUADS
   * Jelaskan perbedaan masing-masing nya.
2. Buatlah program untuk membaut lingkaran, dengan output sebagai berikut:
3. Buatlah program untuk menghasilkan segi delapan berwarna, dengan menambahkan fungsi *glcolor3f()* simpan dengan nama Tugas2.c. Output sebagai berikut:

Gambar 3.3 Output Program Lingkaran dan Segi Delapan Berwarna

1. Kesimpulan dari latihan pada Modul 3:

# MODUL 4: Ploting Curva

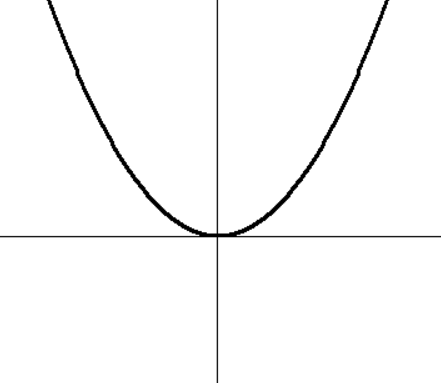
**TUJUAN**

1. Membuat kurva 2 dimensi menggunakan openGL
2. Mengarsir area tertentu pada kurva menggunakan openGL

**TEORI**

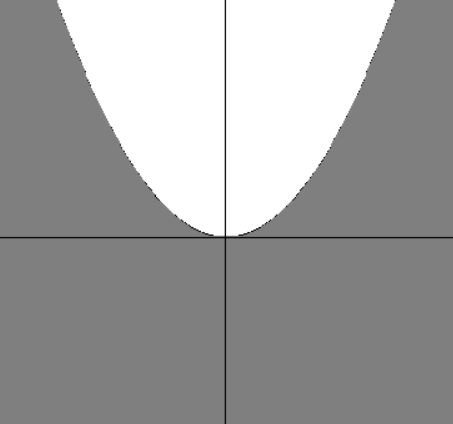
Tujuan dari modul ini adalah untuk membuat kurva dua dimensi dan mengarsir area tertentu pada kurva tersebut. Pada matematika, sebuah kurva adalah objek menyerupai garis, tetapi tidak harus lurus. Karena itu sebuah kurva adalah generalisasi dari sebuah garis yang dapat memiliki lengkungan.

Pembuatan kurva dua dimensi menggunakan openGL pada dasarnya meliputi pembuatan bidang kartesian dan pemetaan persamaan garis dari kurva yang diinginkan. Sebagai contoh, berikut ini adalah grafik kartesian untuk persamaan kurva y = x2.



Gambar 1- Kurva

Selain dari pembuatan grafik kartesian, pada modul ini juga akan dilakukan pengarsiran area tertentu dari grafik tersebut. Sebagai contoh, gambar berikut ini menunjukkan perngarsiran daerah y < x2.



Gambar 2- Kurva dengan arsiran daerah

**Latihan 4\_1.py**

import pygame

from pygame.locals import \*

from OpenGL.GL import \*

from OpenGL.GLU import \*

import numpy as np

import math

def plotfunc():

glClear(GL\_COLOR\_BUFFER\_BIT)

glColor3f(0.0, 0.0, 0.0)

glPointSize(3.0)

glBegin(GL\_LINES)

glVertex2f(-5.0, 0.0)

glVertex2f(5.0, 0.0)

glVertex2f(0.0, 5.0)

glVertex2f(0.0, -5.0)

glEnd()

for x in np.arange(-5.0, 5.0, 0.01):

y = x\*x

glBegin(GL\_POINTS)

glVertex2f(x,y)

# pygame.time.wait(50)

glEnd()

glFlush()

def init():

glClearColor (1.0, 0.0, 0.0, 1.0); ## background colour

def main():

pygame.init()

display = (400,400)

pygame.display.set\_caption('Function Plot')

pygame.display.set\_mode(display, DOUBLEBUF|OPENGL)

gluPerspective(45, (display[0]/display[1]), 0.1, 50.0)

glTranslatef(0.0,0.0, -5)

glClearColor(1.0, 0.0, 0.0, 1.0)

while True:

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

quit()

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT)

plotfunc()

pygame.display.flip()

pygame.time.wait(10)

main()

**Latihan 4\_2.py**

def plotfunc():

glClear(GL\_COLOR\_BUFFER\_BIT)

glColor3f(0.0, 0.0, 0.0)

glPointSize(1.0)

for x in np.arange(-5.0, 5.0, 0.01):

y = x\*x

glColor3f(0.0, 0.0, 0.0)

glBegin(GL\_POINTS)

glVertex2f(x,y)

glEnd()

for a in np.arange(-5.0, 5.0, 0.01):

if a < x\*x:

glColor3f(0.50,0.50,0.50)

glBegin(GL\_POINTS)

glVertex2f(x,a)

glEnd()

glColor3f(0.0, 0.0, 0.0)

glBegin(GL\_LINES)

glVertex2f(-5.0, 0.0)

glVertex2f(5.0, 0.0)

glVertex2f(0.0, 5.0)

glVertex2f(0.0, -5.0)

glEnd()

glFlush()

\*Pengarsiran membutuhkan memory cukup besar. Jika memory yang digunakan kurang, step pada looping kedua bisa diperbesar.

def plotfunc():

glClear(GL\_COLOR\_BUFFER\_BIT)

glColor3f(0.0, 0.0, 0.0)

glPointSize(1.0)

for x in np.arange(-5.0, 5.0, 0.01):

y = x\*x

glColor3f(0.0, 0.0, 0.0)

glBegin(GL\_POINTS)

glVertex2f(x,y)

glEnd()

for a in np.arange(-5.0, 5.0, 0.01):

if a < x\*x:

glColor3f(0.50,0.50,0.50)

glBegin(GL\_POINTS)

glVertex2f(x,a)

glEnd()

glColor3f(0.0, 0.0, 0.0)

glBegin(GL\_LINES)

glVertex2f(-5.0, 0.0)

glVertex2f(5.0, 0.0)

glVertex2f(0.0, 5.0)

glVertex2f(0.0, -5.0)

glEnd()

glFlush()

**TUGAS**:

1. Buatlah program untuk fungsi berikut:
2. Kesimpulan pada modul 4

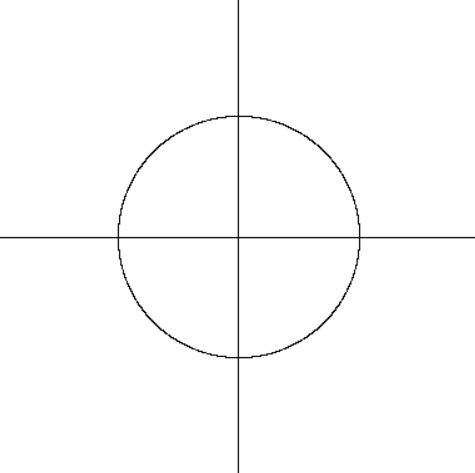
# MODUL 5: Plot Fungsi Parametrik

**TUJUAN**

1. Pemetaan fungsi parametrik dengan openGL
2. Pembuatan gambar abstrak menggunakan fungsi parametrik dengan openGL

**TEORI**

Fungsi atau persamaan parametrik adalah himpunan persamaan yang mengekspresikan sejumlah objek sebagai fungsi eksplisit dari sejumlah variabel independen yang disebut sebagai “parameter”.



Gambar 3- Kurva Lingkaran

Gambar di atas menunjukkan sebuah lingkaran yang didapatkan dari fungsi parametrik:

x = sin(t) dan y = cos(t)

**Latihan5\_1.py**

import pygame

from pygame.locals import \*

from OpenGL.GL import \*

from OpenGL.GLU import \*

import numpy as np

import math

def plotfunc():

glClear(GL\_COLOR\_BUFFER\_BIT)

glColor3f(0.0, 0.0, 0.0)

glPointSize(1.0)

glBegin(GL\_LINES)

glVertex2f(-2.0, 0.0)

glVertex2f(2.0, 0.0)

glVertex2f(0.0, 2.0)

glVertex2f(0.0, -2.0)

glEnd()

for t in np.arange(-5.0,6.28, 0.001):

x = math.sin(t)

y = math.cos(t)

glBegin(GL\_POINTS)

glVertex2f(x,y)

glVertex2f(t,z)

glEnd()

glFlush()

def main():

pygame.init()

display = (400,400)

pygame.display.set\_caption('Function Plot')

pygame.display.set\_mode(display, DOUBLEBUF|OPENGL)

gluPerspective(45, (display[0]/display[1]), 0.1, 50.0)

glTranslatef(0.0,0.0, -5)

glClearColor(1.0, 1.0, 1.0, 1.0)

while True:

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

quit()

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT)

plotfunc()

pygame.display.flip()

pygame.time.wait(10)

main()

**Latihan 5\_2.py**

Tambahkan fungsi garis y=x (memotong pusat lingkaran), dengan menambahkan sintaks:

def plotfunc():

glClear(GL\_COLOR\_BUFFER\_BIT)

glColor3f(0.0, 0.0, 0.0)

glPointSize(1.0)

# Plot the coordinate axes

glBegin(GL\_LINES)

glVertex2f(-2.0, 0.0)

glVertex2f(2.0, 0.0)

glVertex2f(0.0, 2.0)

glVertex2f(0.0, -2.0)

glEnd()

# Plot the parametric equations

for t in np.arange(0.0,6.28, 0.001):

x = math.sin(t)

y = math.cos(t)

z = t

glBegin(GL\_POINTS)

glVertex2f(x, y)

glVertex2f(t, z)

glEnd()

glFlush()

**Latihan5\_3.py**

def plotfunc():

glClear(GL\_COLOR\_BUFFER\_BIT)

glColor3f(0.0, 0.0, 0.0)

glPointSize(1.0)

# Plot the coordinate axes

glBegin(GL\_LINES)

glVertex2f(-2.0, 0.0)

glVertex2f(2.0, 0.0)

glVertex2f(0.0, 2.0)

glVertex2f(0.0, -2.0)

glEnd()

# Plot the parametric equations

for t in np.arange(0.0,6.28, 0.001):

x = math.sin(3\*t)

y = math.cos(5\*t)

glBegin(GL\_POINTS)

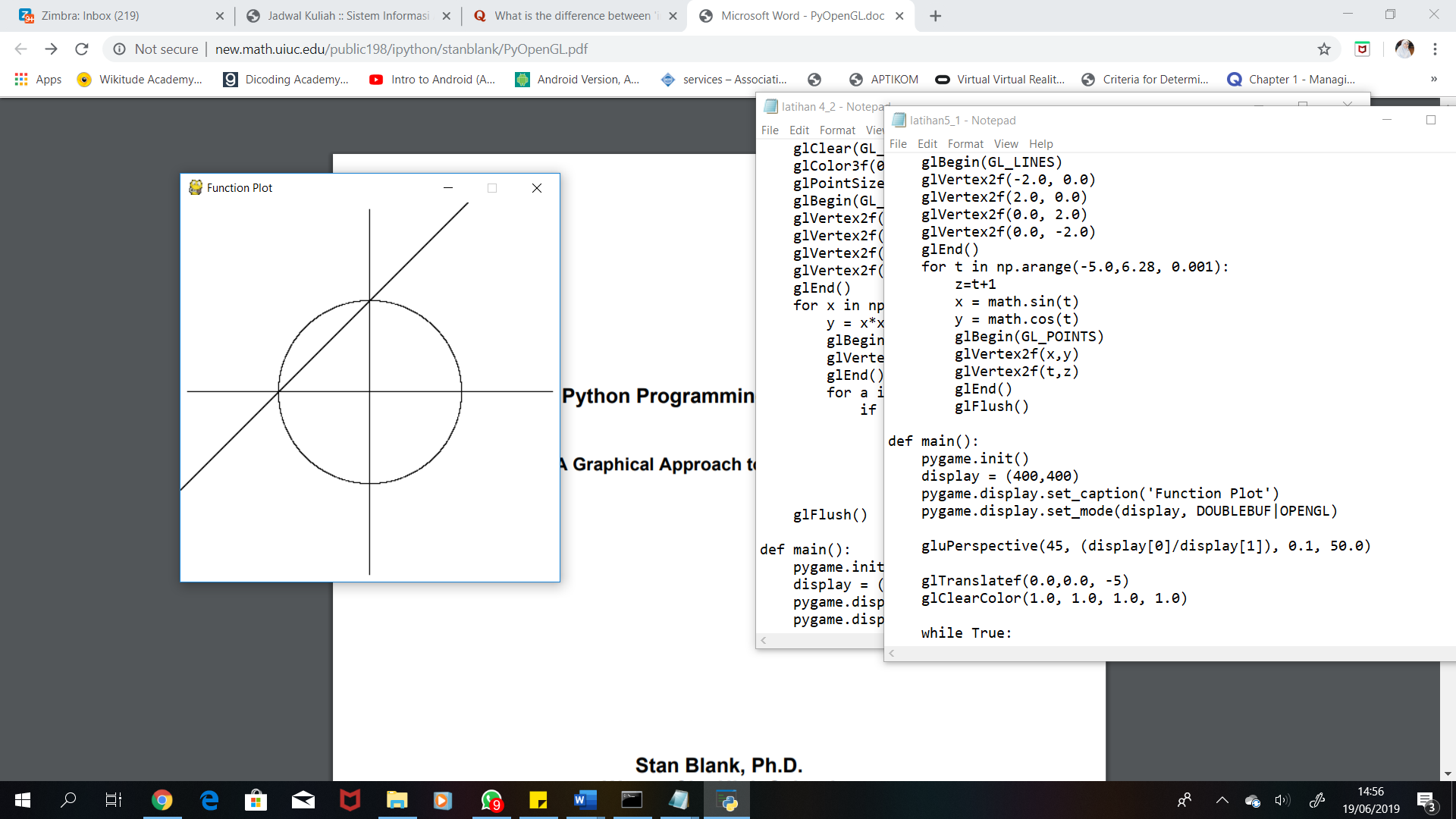
glVertex2f(x, y)

glEnd()

glFlush()

**TUGAS**

1. Tentukan persamaan garis yang baru, dengan menggeser pada sumbu y yang menghasilkan output sbb:



1. Kesimpulan dari latihan pada Modul 5:

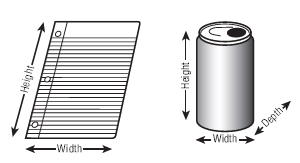
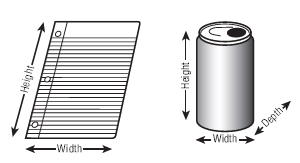
# MODUL 6: 3D Vertex

**TUJUAN**

1. Mengenal obyek 3 Dimensional vertex
2. Membuat obyek 3D menggunakan glVertex2f
3. Membuat obyek 3D menggunakan glVertex3f

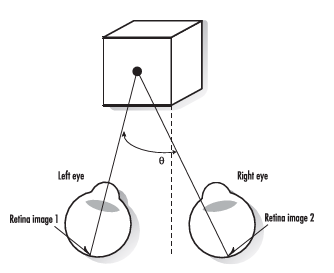
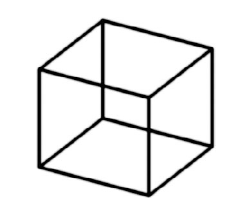
**TEORI**

Semua vertex dari obyek berupa model 2D menggunakan vertex properti koordinat x dan y dengan menggunakan fungsi **glVertex2f()**. Untuk memodelkan obyek dalam 3D kita perlu memberi properti koordinat z dengan menggunakan fungsi **glVertex3f()**. Salah satu contoh objek 2D dapat digambar/dilihat paa kertas. Kertas hanya dapat diisi atau ditulis dari kiri ke kanan/ dari atas ke bawah. Hal ini berbeda dengan sebuah botol soda. Minuman yang terdapat dalam botol soda dapat dapat bergerak, kiri-kanan, atas-bawah dan depan-belakang.



Gambar 6.1 Koordinat Objek 2D dan 3D

3D computer graphics gabungan objek 2D, dengan penambahan bayangan depth (kedalaman). Salah satu contoh 3D sederhana, sebuah kotak yang merupakan gabungan dari 12 segmen garis. Yang menjadikan kubus terlihat 3 dimensi adalah perspektiv (sudut antara garis dan dasar). Masing2 mata melihat objek 2D dari sudut pandang yang berbeda, karena kedua mata terpisah pada jarak tertentu. Otak kita menyatukan keduanya dan terlihat seperti satu objek yang utuh yaitu kubus 3D.

****

Gambar 6.2 Retinal viewing

**Latihan 6\_1: Kubus 3D**

def plotfunc():

squares = [

[[-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]]

]

glClear(GL\_COLOR\_BUFFER\_BIT)

glColor3f(0.0, 0.0, 0.0)

glPointSize(1.0)

for square in squares:

glColor3f(1,0,0)

glBegin(GL\_POLYGON)

for point in square:

glVertex3f(point[0], point[1], point[2])

glEnd()

glColor3f(0,0,0)

glBegin(GL\_LINE\_LOOP)

for point in square:

glVertex3f(point[0], point[1], point[2])

glEnd()

glFlush()

Latihan6\_2:Bangunan\_Sederhana

1. Gantilah objek kubus 3D dengan objek 3D menggunakan fungsi yang telah disediakan oleh GLUT:
   1. SolidTeapot
   2. WireTeapot
   3. SolidCube
   4. WireCube
2. Kesimpulan dari latihan pada Modul 6

**TUGAS**

Buatlah sebuah bangunan yang menggunakan dua atau lebih bangun dasar yang ada (contoh: balok, limas, atau prisma)

# MODUL 7: Transformasi

**TUJUAN**

1. Mengenal transformasi pada OpenGL
2. Membuat transformasi proyeksi perspective
3. Memahami perbedaan transformasi proyeksi orthogonal dan perspective

**TEORI**

Secara substansi, Grafika Komputer adalah proses transformasi dari model 3D obyek berupa informasi geometri bentuk, informasi pose, warna, texture, dan pencahayaan menjadi citra 2D

Model 3D Obyek

* Bentuk
* Pose (Posisi & Orientasi)
* Warna
* Texture
* Pencahayaan

Citra 2D Obyek

**T**

Jika dilihat secara analogi, hal di atas mirip dengan cara kerja kamera dalam mengambil foto dalam bidang fotografi Model ini disebut model sintesis kamera.

**camera**

**tripod**

**model**



**Viewing volume**

Gambar 5.1 ANalogi viewing volume

Untuk menghasilkan gambar dari obyek dengan skenario tertentu kita harus melakukan beberapa proses, yaitu:

* melakukan pengesetan kamera dalam bentuk setting lensa kamera (**Transformasi Proyeksi**),
* mengarah kamera dengan mengatur letak tripod (**Transformasi Viewing**),
* mengatur letak obyek (**Transformasi Modeling**), dan
* mengatur skala dan layout dari foto (**Transformasi Viewport**)

Kita telah mempelajari Transformasi Viewport pada tutorial sebelumnya dengan menggunakan perintah **glViewport()**.

Tipe transformasi proyeksi ada dua macam, bergantung pada parameter dan bentuk piramidanya. Dua tipe transformasi tersebut adalah **Transformasi Ortogonal/Paralel** (Orthogonal Transformation) dan **Transformasi Perspektif**(Perspective Transformation)

**Transformasi Viewing**

Untuk menghasilkan gambar, kamera perlu diletakkan pada posisi yang tepat didepan pemandangan yang diinginkan. Secara default, dalam OpenGL kemera akan berada pada posisi (0,0,0) dengan menghadap ke arah *z* = -1 dengan sumbu *y* mengarah ke atas kamera. Hal ini dapat dilakukan dengan menggunakan perintah **gluLookAt()** dengan didahului proses merubah status OpenGL ke mode proyeksi dengan perintah **glMatrixMode(GL\_MODELVIEW)**.

**Transformasi Modeling**

Selain posisi dan orientasi kamera yang dapat dirubah-rubah, secara natural obyek juga dapat berpindah posisi dan orientasi relatif terhadap yang lain.Transformasi obyek dapat direpresentasikan dengan dua cara, yaitu:

* menggunakan matriks transformasi (**glLoadMatrix**)
* menggunakan operasi transformasi (**glRotate**, **glTranslate**)

dengan didahului proses merubah status OpenGL ke mode proyeksi dengan perintah **glMatrixMode(GL\_MODELVIEW)**.

**Latihan7\_1: Rotasi**

import pygame

from pygame.locals import \*

from OpenGL.GL import \*

from OpenGL.GLU import \*

def plotfunc():

squares = [

[[-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]]

]

glClear(GL\_COLOR\_BUFFER\_BIT)

glColor3f(0.0, 0.0, 0.0)

glPointSize(1.0)

for square in squares:

lColor3f(1,0,0)

glBegin(GL\_POLYGON)

for point in square:

glVertex3f(point[0], point[1], point[2])

glEnd()

glColor3f(0,0,0)

glBegin(GL\_LINE\_LOOP)

for point in square:

glVertex3f(point[0], point[1], point[2])

glEnd()

glFlush()

def main():

pygame.init()

display = (400,400)

pygame.display.set\_caption('Function Plot')

pygame.display.set\_mode(display, DOUBLEBUF|OPENGL)

gluPerspective(45, (display[0]/display[1]), 0.1, 50.0)

glTranslatef(0.0,0.0, -5)

glClearColor(1.0, 1.0, 1.0, 1.0)

rotate\_angle = 1

x\_rotate = 3

y\_rotate = 2

z\_rotate = 1

while True:

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

quit()

glRotatef(rotate\_angle, x\_rotate, y\_rotate, z\_rotate)

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT)

plotfunc()

pygame.display.flip()

pygame.time.wait(10)

main()

**Latihan7\_2:Translasi**

def plotfunc(xt=0, yt=0, zt=0):

squares = [

[[-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]]

]

glClear(GL\_COLOR\_BUFFER\_BIT)

glColor3f(0.0, 0.0, 0.0)

glPointSize(1.0)

for square in squares:

glColor3f(1,0,0)

glBegin(GL\_POLYGON)

for point in square:

glVertex3f(point[0]+xt, point[1]+yt, point[2]+zt)

glEnd()

glColor3f(0,0,0)

glBegin(GL\_LINE\_LOOP)

for point in square:

glVertex3f(point[0]+xt, point[1]+yt, point[2]+zt)

glEnd()

glFlush()

Bagaimana parameter yang ditambahkan memengaruhi bangun yang dibuat?

**Latihan7\_3: Scaling**

def plotfunc(scale\_size=1):

squares = [

[[-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]]

]

glClear(GL\_COLOR\_BUFFER\_BIT)

glColor3f(0.0, 0.0, 0.0)

glPointSize(1.0)

for square in squares:

glColor3f(1,0,0)

glBegin(GL\_POLYGON)

for point in square:

glVertex3f((point[0]\*scale\_size), (point[1]\*scale\_size), (point[2]\*scale\_size))

glEnd()

glColor3f(0,0,0)

glBegin(GL\_LINE\_LOOP)

for point in square:

glVertex3f((point[0]\*scale\_size), (point[1]\*scale\_size), (point[2]\*scale\_size))

glEnd()

glFlush()

Latihan7\_4: Penggabungan\_Fungsi\_Rotasi\_Translasi\_dan\_Scaling

Gabungkanlah code pada Latihan7\_1, Latihan7\_2, dan Latihan7\_3 sehingga translasi dan scaling hanya memerlukan satu fungsi plotfunc() saja

**Latihan:**

1. Lakukan perubahan rotasi, translasi dan scaling dengan mengganti parameter translate, rotate angle dan scale size.
2. Bagaimana parameter yang ditambahkan memengaruhi bangun yang dibuat?

**TUGAS**

Buatlah miniatur sebuah kota yang di dalamnya terdapat beberapa bangunan, jalan, dan kendaraan yang bergerak. Bangunan dapat menggunakan hasil dari latihan sebelumnya. Kendaraan yang dibuat tidak perlu detail.

# MODUL 8: Lighting

**TUJUAN**

1. Mengenal proses lighting dalam OpenGL
2. Memahami pengaturan cahaya/ Lighting dalam merepresentasikan obyek 2D/3D

**TEORI**

Visualisasi tentu saja tidak akan terjadi bila tidak ada cahaya. Pencahayaan merupakan esensi dari visualisasi dan merupakan topik yang sangat kompleks. Hingga tahap ini lingkungan diberi pencahayaan default/standar dengan cahaya lingkungan (ambient) yang sama pada setiap titik. Kondisi default/standar dapat dicapai kapan saja dengan mematikan status Lighting menjadi disabled dengan **glDisable(GL\_LIGHT0)**.

Dalam pencahayaan, ada dua hal yang menentukan tampilan suatu obyek, yaitu:

* Sumber cahaya dan pengaruh lingkungan terhadap cahaya
  + Lokasi sumber cahaya
  + Arah pencahayaan dari sumber cahaya (omni, spot)
  + Komponen pengaruh lingkungan terhadap cahaya (ambient, diffuse, specular)
* Material dari obyek, yang memodelkan bagaimana material bereaksi terhadap sumber cahaya, yaitu:
  + Material reflektan terhadap komponen cahaya ambient
  + Material reflektan terhadap komponen cahaya diffuse
  + Material reflektan terhadap komponen cahaya specular
  + Material sebagai sumber cahaya (emitance)

Komponen ambient adalah cahaya yang arahnya tidak dapat ditentukan karena datang secara merata dari segala arah. Biasanya merupakan cahaya yang dihasilkan dari pemantulan berkali-kali sumber cahaya yang berarah. Poligon dalam openGL selalu diiluminasi secara seragam oleh komponen ambient tanpa memperdulikan orientasi dan posisinya. Komponen diffuse adalah cahaya yang bersumber dari satu arah dan mempengaruhi poligon secara uniform bergantung pada sudut datang terhadap permukaan poligon. Komponen specular adalah cahaya yang memantul dari obyek yang bergantung pada derajat inklinasi dari poligon terhadap cahaya dan posisi\_observer.

**LATIHAN**

**Latihan 8\_1. Lighting1**

import pygame

from pygame.locals import \*

import sys

from OpenGL.GL import \*

from OpenGL.GLU import \*

from OpenGL.GLUT import \*

import math

name = "Latihan8\_1"

def main():

glutInit(sys.argv)

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH)

glutInitWindowSize(800, 800)

glutInitWindowPosition(350, 200)

glutCreateWindow(name)

glClearColor(0., 0., 0., 1.)

glShadeModel(GL\_SMOOTH)

glEnable(GL\_CULL\_FACE)

glEnable(GL\_DEPTH\_TEST)

glEnable(GL\_LIGHTING)

lightZeroPosition = [10., 4., 10., 1.]

lightZeroColor = [0.8, 1.0, 0.8, 1.0]

glLightfv(GL\_LIGHT0, GL\_POSITION, lightZeroPosition)

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, lightZeroColor)

glLightf(GL\_LIGHT0, GL\_CONSTANT\_ATTENUATION, 0.1)

glLightf(GL\_LIGHT0, GL\_LINEAR\_ATTENUATION, 0.05)

glEnable(GL\_LIGHT0)

glutDisplayFunc(display\_scene)

glMatrixMode(GL\_PROJECTION)

gluPerspective(40., 1., 1., 40.)

glMatrixMode(GL\_MODELVIEW)

gluLookAt(0, 0, 10,

0, 0, 0,

0, 1, 0)

glPushMatrix()

glutMainLoop()

return

def display\_scene():

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

glPushMatrix()

# Left sphere

color = [1.0, 0.0, 0.0, 1.0]

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, color)

glTranslatef(-2, 0, 0)

glutSolidSphere(1, 100, 20)

# Right sphere

color = [0.0, 1.0, 0.0, 1.0]

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, color)

glTranslatef(4, 0, 0)

glutSolidSphere(1, 100, 20)

glPopMatrix()

glutSwapBuffers()

return

main()

**Latihan8\_2: Lighting2**

Buat *project* baru dengan nama **Latihan8\_2**. Ubahlah bangun dasar (bola solid) yang ada pada Latihan8\_1. Simpan dengan **latihan8\_2.py**

**Latihan8\_3: Lighting3**

Buat *project* baru dengan nama **Latihan8\_3**. Ber eksperimen lah dengan mengubah nilai-nilai yang ada pada variabel lightZeroPosition dan lightZeroColor. Simpan dengan nama **latihan8\_3.py**

**TUGAS**

1. Lampirkan output program 8\_1, 8\_2 dan 8\_3

……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

1. Jelaskan parameter-parameter yang dibutuhkan oleh fungsi glLightfv dan perbedaan mode-mode yang digunakan pada glLightfv!

……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

1. Kesimpulan dari latihan pada Modul 8:

……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

# MODUL 9: Texture Mapping

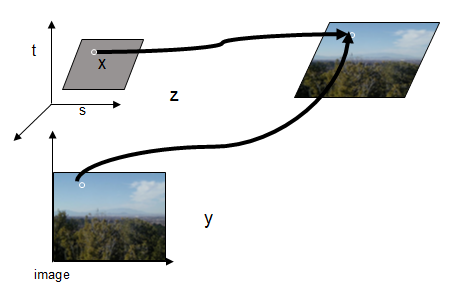
**TUJUAN**

1. Mengenal proses texturing obyek
2. Menambahkan texture mapping pada suatu obyek
3. Mengatur fungsi Blending OpenGL

**TEORI**

Texture mapping memungkinkan untuk menaruh gambar pada geometric primitive tersebut dan sekaligus mengikuti transformasi yang diterapkan kepada polygon tersebut. Texture merupakan data segi-empat sederhana yang berada pada bidang texture. Bidang texture diwakili oleh dua sumbu koordinat yaitu sumbu s dan sumbu t.

Setiap texture akan memenuhi bidang koordinat (0.0,0.0) sd. (1.0,1.0). Nilai individual dari array texture biasanya dikenal dengan istilah texels (texture pixels). Yang membuat texture mapping sedikit rumit adalah bagaimana proses pemetaan antara bentuk segi-empat texture ke polygon menginngat secara umum bentuk poligon biasanya .



Gambar 8.1 Texture Mapping

Salah satu keuntungan dari texture mapping adalah bahwa detail visual itu berada di citra bukan di geometri. Dan sekompleks apapun citra, selama tidak merubah ukuran citra, tidak berpengaruh pada kinerja keseluruhan, yaitu kompleksitas cari citra tidak berpengaruh kepada pipeline geometric (transformasi, clipping) dari OpenGL.

**LATIHAN**

**Latihan9\_1.TextureMapping** : Texture Mapping Bola dengan GLUT

from OpenGL.GLUT import \*

from OpenGL.GLU import \*

from OpenGL.GL import \*

import sys

from PIL import Image as Image

import numpy

name = 'Texturing'

def main():

glutInit(sys.argv)

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH)

glutInitWindowSize(800, 800)

glutInitWindowPosition(350, 200)

glutCreateWindow(name)

glutDisplayFunc(display\_scene)

glMatrixMode(GL\_PROJECTION)

gluPerspective(40., 1., 1., 40.)

glMatrixMode(GL\_MODELVIEW)

gluLookAt(0, 0, 10,

0, 0, 0,

0, 1, 0)

glPushMatrix()

glutMainLoop()

return

def display\_scene():

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

glPushMatrix()

# Textured thing

tex = read\_texture('brick.jpg')

qobj = gluNewQuadric()

gluQuadricTexture(qobj, GL\_TRUE)

glEnable(GL\_TEXTURE\_2D)

glBindTexture(GL\_TEXTURE\_2D, tex)

glBegin(GL\_TRIANGLES)

gluSphere(qobj, 1, 50, 50)

gluDeleteQuadric(qobj)

glDisable(GL\_TEXTURE\_2D)

glutSwapBuffers()

return

def read\_texture(filename):

img = Image.open(filename)

img\_data = numpy.array(list(img.getdata()), numpy.int8)

textID = glGenTextures(1)

glBindTexture(GL\_TEXTURE\_2D, textID)

glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_CLAMP)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_CLAMP)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_NEAREST)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_NEAREST)

glTexEnvf(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_DECAL)

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, img.size[0], img.size[1], 0, GL\_RGB, GL\_UNSIGNED\_BYTE, img\_data)

return textID

if \_\_name\_\_ == '\_\_main\_\_':

main()

**Latihan9\_2.TextureMapping** : Texture Mapping Bangun Kubus tanpa GLUT

import pygame

import sys

from OpenGL.GL import \*

from OpenGL.GLU import \*

vertices = (

# x y z

(1, -1, -1),

(1, 1, -1),

(-1, 1, -1),

(-1, -1, -1),

(1, -1, 1),

(1, 1, 1),

(-1, -1, 1),

(-1, 1, 1)

)

edges = (

(0, 1),

(0, 3),

(0, 4),

(2, 1),

(2, 3),

(2, 7),

(6, 3),

(6, 4),

(6, 7),

(5, 1),

(5, 4),

(5, 7)

)

def loadTexture():

textureSurface = pygame.image.load('test\_image.png')

textureData = pygame.image.tostring(textureSurface, "RGBA", 1)

width = textureSurface.get\_width()

height = textureSurface.get\_height()

glEnable(GL\_TEXTURE\_2D)

texid = glGenTextures(1)

glBindTexture(GL\_TEXTURE\_2D, texid)

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, width, height,

0, GL\_RGBA, GL\_UNSIGNED\_BYTE, textureData)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_NEAREST)

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_NEAREST)

return texid

def draw\_cube(lines=False):

if lines:

glBegin(GL\_LINES)

for edge in edges:

glColor3fv((1, 1, 1))

for vertex in edge:

glVertex3fv(vertices[vertex])

glEnd()

else:

glBegin(GL\_QUADS)

glTexCoord2f(0.0, 0.0)

glVertex3f(-1.0, -1.0, 1.0)

glTexCoord2f(1.0, 0.0)

glVertex3f(1.0, -1.0, 1.0)

glTexCoord2f(1.0, 1.0)

glVertex3f(1.0, 1.0, 1.0)

glTexCoord2f(0.0, 1.0)

glVertex3f(-1.0, 1.0, 1.0)

glTexCoord2f(1.0, 0.0)

glVertex3f(-1.0, -1.0, -1.0)

glTexCoord2f(1.0, 1.0)

glVertex3f(-1.0, 1.0, -1.0)

glTexCoord2f(0.0, 1.0)

glVertex3f(1.0, 1.0, -1.0)

glTexCoord2f(0.0, 0.0)

glVertex3f(1.0, -1.0, -1.0)

glTexCoord2f(0.0, 1.0)

glVertex3f(-1.0, 1.0, -1.0)

glTexCoord2f(0.0, 0.0)

glVertex3f(-1.0, 1.0, 1.0)

glTexCoord2f(1.0, 0.0)

glVertex3f(1.0, 1.0, 1.0)

glTexCoord2f(1.0, 1.0)

glVertex3f(1.0, 1.0, -1.0)

glTexCoord2f(1.0, 1.0)

glVertex3f(-1.0, -1.0, -1.0)

glTexCoord2f(0.0, 1.0)

glVertex3f(1.0, -1.0, -1.0)

glTexCoord2f(0.0, 0.0)

glVertex3f(1.0, -1.0, 1.0)

glTexCoord2f(1.0, 0.0)

glVertex3f(-1.0, -1.0, 1.0)

glTexCoord2f(1.0, 0.0)

glVertex3f(1.0, -1.0, -1.0)

glTexCoord2f(1.0, 1.0)

glVertex3f(1.0, 1.0, -1.0)

glTexCoord2f(0.0, 1.0)

glVertex3f(1.0, 1.0, 1.0)

glTexCoord2f(0.0, 0.0)

glVertex3f(1.0, -1.0, 1.0)

glTexCoord2f(0.0, 0.0)

glVertex3f(-1.0, -1.0, -1.0)

glTexCoord2f(1.0, 0.0)

glVertex3f(-1.0, -1.0, 1.0)

glTexCoord2f(1.0, 1.0)

glVertex3f(-1.0, 1.0, 1.0)

glTexCoord2f(0.0, 1.0)

glVertex3f(-1.0, 1.0, -1.0)

glEnd()

pygame.init()

display = (800, 600)

screen = pygame.display.set\_mode(

display, pygame.DOUBLEBUF | pygame.OPENGL | pygame.OPENGLBLIT)

loadTexture()

gluPerspective(45, display[0] / display[1], 0.1, 50.0)

glTranslatef(0.0, 0.0, -5)

while True:

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

draw\_cube(lines=False)

pygame.display.flip()

**Latihan9\_3.TextureMapping** : Texture Mapping Bangun Kubus Lanjutan

Buat *project* baru dengan nama **Latihan9\_3**. Buatlah sebuah kubus dengan texture yang berbeda pada tiap sisinya. Agar bisa terlihat lebih jelas, ubahlah sudut pandang kubus.

**Latihan\_Tambahan9\_4.TextureMapping** : Texture Mapping Bola Lanjutan

Buat *project* baru dengan nama **Latihan9\_4**. Dari program latihan 9\_1, proseslah gambar yang digunakan sehingga gambar tersebut dapat terlihat jelas pada bola yang dibuat.

**TUGAS**

1. Lampirkan output program 9\_1, 9\_2, 9\_3
2. Jelaskan texture mapping yang terjadi pada latihan 9\_1
3. Kesimpulan dari latihan pada Modul 9

# MODUL 10: Games Sederhana

**TUJUAN**

1. Bereksperimen dengan contoh game yang diberikan
2. Membuat sistem scoring pada game
3. Mengaplikasikan texture pada contoh game

**LATIHAN**

**Latihan10\_1.ContohGame** : snake.py

"""

Simple snake example.

Sample Python/Pygame Programs

Simpson College Computer Science

http://programarcadegames.com/

http://simpson.edu/computer-science/

"""

import pygame

# --- Globals ---

# Colors

BLACK = (0, 0, 0)

WHITE = (255, 255, 255)

# Set the width and height of each snake segment

segment\_width = 15

segment\_height = 15

# Margin between each segment

segment\_margin = 3

# Set initial speed

x\_change = segment\_width + segment\_margin

y\_change = 0

class Segment(pygame.sprite.Sprite):

""" Class to represent one segment of the snake. """

# -- Methods

# Constructor function

def \_\_init\_\_(self, x, y):

# Call the parent's constructor

super().\_\_init\_\_()

# Set height, width

self.image = pygame.Surface([segment\_width, segment\_height])

self.image.fill(WHITE)

# Make our top-left corner the passed-in location.

self.rect = self.image.get\_rect()

self.rect.x = x

self.rect.y = y

class Obstacle(pygame.sprite.Sprite):

def \_\_init\_\_ (self, x, y):

super().\_\_init\_\_()

self.image = pygame.Surface([segment\_width, segment\_height])

self.image.fill(WHITE)

self.rect = self.image.get\_rect()

self.rect.x = x

self.rect.y = y

# Call this function so the Pygame library can initialize itself

pygame.init()

# Create an 800x600 sized screen

screen = pygame.display.set\_mode([800, 600])

# Set the title of the window

pygame.display.set\_caption('Snake Example')

allspriteslist = pygame.sprite.Group()

#bordersprite = pygame.sprite.Group()

# Create border

borders\_objects = []

borders\_coor = []

for i in range(45):

x = 800 - (segment\_width + segment\_margin) \* i

y1 = 0

y2 = 576

obstacle = Obstacle(x, y1)

borders\_objects.append(obstacle)

borders\_coor.append([x, y1])

allspriteslist.add(obstacle)

obstacle = Obstacle(x, y2)

borders\_objects.append(obstacle)

borders\_coor.append([x, y2])

allspriteslist.add(obstacle)

for i in range(32):

y = 576 - (segment\_width + segment\_margin) \* i

x1 = 782

x2 = 8

obstacle = Obstacle(x1, y)

borders\_objects.append(obstacle)

borders\_coor.append([x1, y])

allspriteslist.add(obstacle)

obstacle = Obstacle(x2, y)

borders\_objects.append(obstacle)

borders\_coor.append([x2, y])

allspriteslist.add(obstacle)

#print(borders)

# Create an initial snake

snake\_segments = []

for i in range(15):

x = 296 - (segment\_width + segment\_margin) \* i

y = 36

segment = Segment(x, y)

snake\_segments.append(segment)

allspriteslist.add(segment)

clock = pygame.time.Clock()

done = False

colission = False

while not done and not colission:

for event in pygame.event.get():

if event.type == pygame.QUIT:

done = True

# Set the speed based on the key pressed

# We want the speed to be enough that we move a full

# segment, plus the margin.

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_LEFT:

x\_change = (segment\_width + segment\_margin) \* -1

y\_change = 0

if event.key == pygame.K\_RIGHT:

x\_change = (segment\_width + segment\_margin)

y\_change = 0

if event.key == pygame.K\_UP:

x\_change = 0

y\_change = (segment\_height + segment\_margin) \* -1

if event.key == pygame.K\_DOWN:

x\_change = 0

y\_change = (segment\_height + segment\_margin)

# Get rid of last segment of the snake

# .pop() command removes last item in list

old\_segment = snake\_segments.pop()

allspriteslist.remove(old\_segment)

# Figure out where new segment will be

x = snake\_segments[0].rect.x + x\_change

y = snake\_segments[0].rect.y + y\_change

segment = Segment(x, y)

if [x, y] in borders\_coor:

pygame.mixer.music.load('ayaya.wav')

pygame.mixer.music.play(0)

pygame.time.wait(2000)

print("You hit a wall")

colission = True

# pygame.quit()

# Insert new segment into the list

snake\_segments.insert(0, segment)

allspriteslist.add(segment)

# -- Draw everything

# Clear screen

screen.fill(BLACK)

allspriteslist.draw(screen)

#bordersprite(screen)

# Flip screen

pygame.display.flip()

# Pause

clock.tick(5)

pygame.quit()

**TUGAS**

1. **Tugas10\_2.ContohGame** : Adding Texture

…………………………………………………………………………………………………………………………………………………………

1. Tambahkan texture untuk background dari contoh latihan 10\_1

…………………………………………………………………………………………………………………………………………………………

1. **Tugas10\_3.ContohGame** : Obstacle and scoring system

…………………………………………………………………………………………………………………………………………………………

1. Tambahkan target makanan dan rintangan dari objek ular pada game. Tambahkan juga system scoring untuk game ini.

…………………………………………………………………………………………………………………………………………………………

1. **Tugas\_Tambahan10\_4.ContohGame** : Audio

…………………………………………………………………………………………………………………………………………………………

1. Berikanlah tambahan background music pada contoh game ini.

…………………………………………………………………………………………………………………………………………………………

1. Lampirkan output program contoh game yang dihasilkan beserta file-file yang dibutuhkan.

…………………………………………………………………………………………………………………………………………………………

1. Kesimpulan dari latihan pada Modul 10:

……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

## Referensi

F.S.Hill, Jr., *Computer Graphics – Using Open GL*, Second Edition, Prentice Hall, 2001.

Neider,Jackie.*OpenGL Programming Guide: the official guide to learning OpenGL release1*. Silicon Graphics,Inc. 1993

Setijadi.P.Ary,*OpenGL Tutorial dengan GLUT: FixedPipeline,*STEI ITB, 2007.

# Format Laporan Praktikum

**LAPORAN PRAKTIKUM**

**GRAFIKA KOMPUTER**

**MODUL xx**

**xxxxxxxxxxxxxxxxx**

**Nama:  
Xxxxxxxxxxxxxxxxxxxxx (NIM)**

**Dosen Pengampu:**

**Riri Safitri, S.Si, MT**

**Asisten:**

**Bintang Nafsul Muthmainnah, S.Kom**

**INFORMATIKA  
FAKULTAS SAINS DAN TEKNOLOGI**

**UNIVERSITAS AL AZHAR INDONESIA**

**2019**

**TUJUAN**

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

**TEORI**

Ringkasan teori

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

**LATIHAN**

1. Source code

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

1. Screenshoot

**TUGAS**

1. Source code

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

1. Screenshoot