GRAFIKA KOMPUTER

Dr. Murinto, S.Si., M.Kom

2022

UAD Universitas Ahmad Dahlan

Pertemuan 2

Pengantar OpenGL

Why OpenGL?

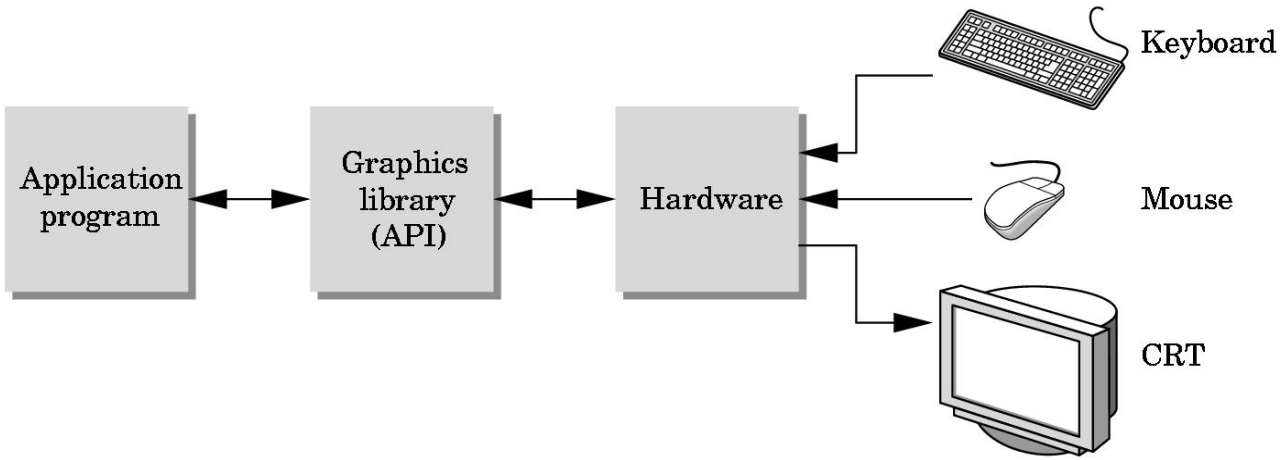
- *Device independence*
- *Platform independence (SGI irix, Linux, Windows)*
- *Abstraction (GL, GLU, GLUT)*
- *Open source*
- *Hardware-independent software interface*
- *Support of client-server protocol*
- *Other API's (OpenInventor->object-oriented toolkit, DirectX(Microsoft), Jave3D(Sun)*

Pengertian OpenGL

- OpenGL merupakan sebuah utiliti yang menjembatani (*interfacing*) antara peralatan grafis (*graphic hardware*) dengan bahasa pemrograman untuk membangun citra grafis 2D dan 3D
- OpenGL merupakan *library* standar yang dikembangkan oleh Silicon Graphics, Inc.
- OpenGL Library berisi fungsi-fungsi dan prosedur yang dapat dipanggil oleh bahasa pemrograman untuk mengakses peralatan grafis sehingga memudahkan dalam membuat aplikasi yang memanfaatkan peralatan grafis (Segel, et.al., 2003).

Interface Programmer

- Programmer melihat sistem grafis melalui suatu suatu interface software : the *Application Programmer Interface (API)*



Konten API

- Fungsi-fungsi yang secara spesifik apa yang dibutuhkan untuk suatu bentuk image
  - Objects
  - Viewer
  - Light Source(s)
  - Materials
- Informasi yang lain
  - Input dari *devices* seperti mouse and keyboard
  - *Capabilities of system*

Fitur dalam OpenGL

- Transformasi 3D (Rotasi, scaling, translasi, prepektif)
- Model Colour (Warna) (Nilai-nilai: R,G,B)
- Lighting (Flat shading, Gouraud shading, Phong shading)
- Rendering (Texture mapping)
- Modeling (non-uniform rational B-spline (NURB), curves, surfaces)
- Others: atmospheric fog, alpha blending, motion blur

Sejarah OpenGL

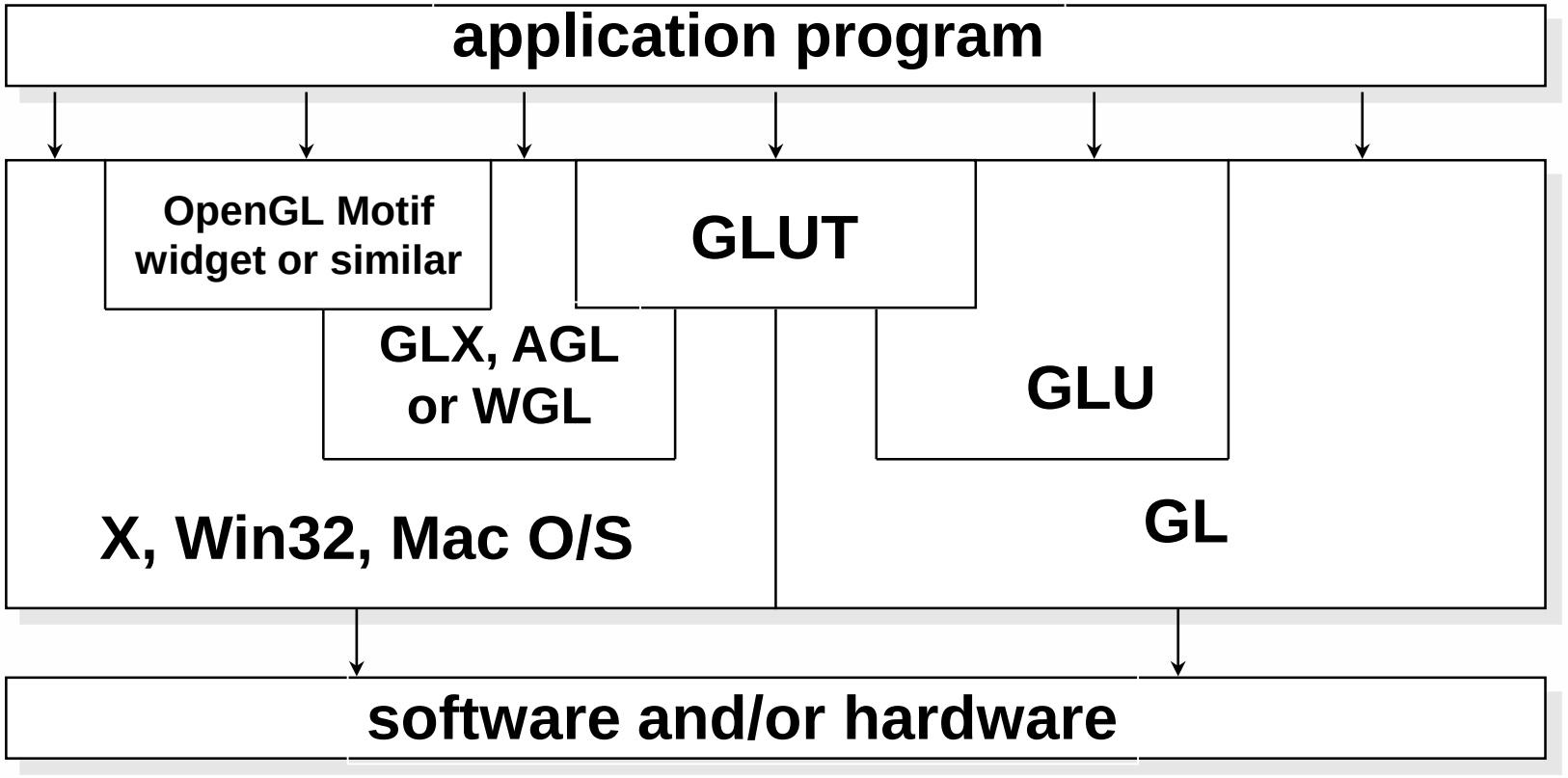
- Silicon Graphics (SGI) revolutionized the graphics workstation by implementing the pipeline in hardware (1982)
- To access the system, application programmers used a library called GL
- With GL, it was relatively simple to program three dimensional interactive applications

OpenGL Libraries

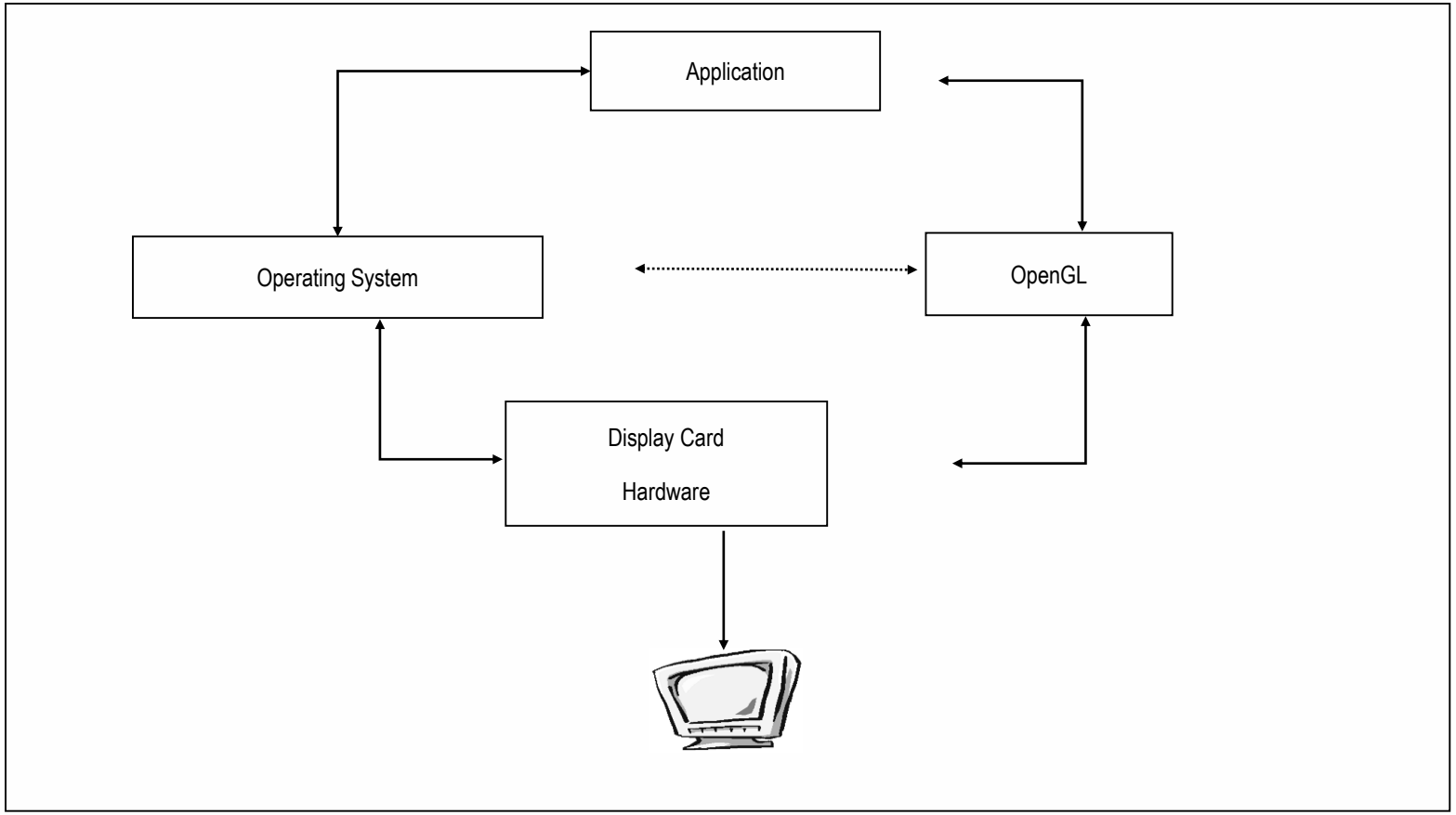
- **GL** (**G**raphics **L**ibrary):
  - Primitif (titik, garis, polygon)
  - Shading dan colour
  - translasi, rotasi, scaling
  - Viewing, Clipping, Texture
  - Hidden surface removal
- **GLU** (**GL U**tilities): Jenis-jenis functions dealing with camera set-up and higher-level shape descriptions (Viewing→prespektif, sphere)
- **GLUT** (**GL U**tility **T**oolkit): Window-system independent toolkit with numerous utility functions, mostly dealing with user interface (key, mouse, handler, window events)

Organisasi Software

- Software/hardware

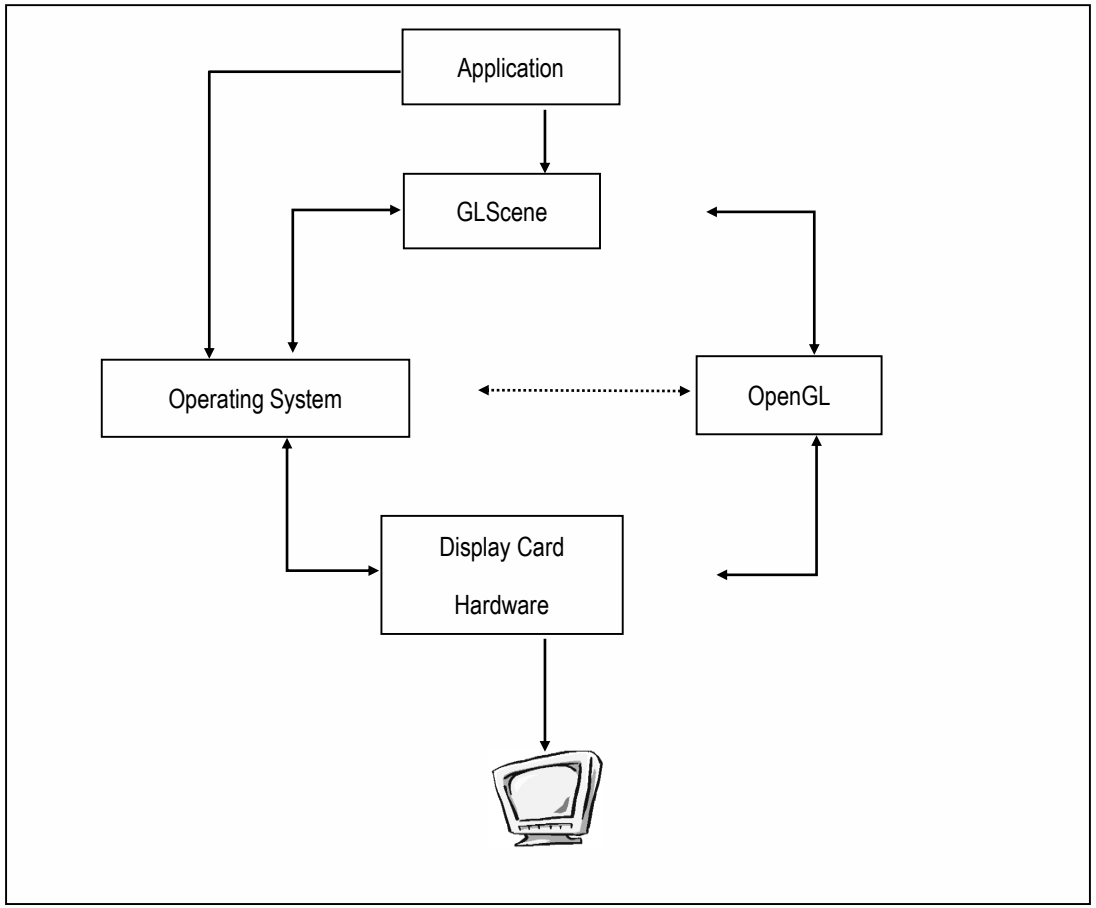


OpenGL System



OpenGL System

GLScene



Gambar 2.22 GLScene

Overview

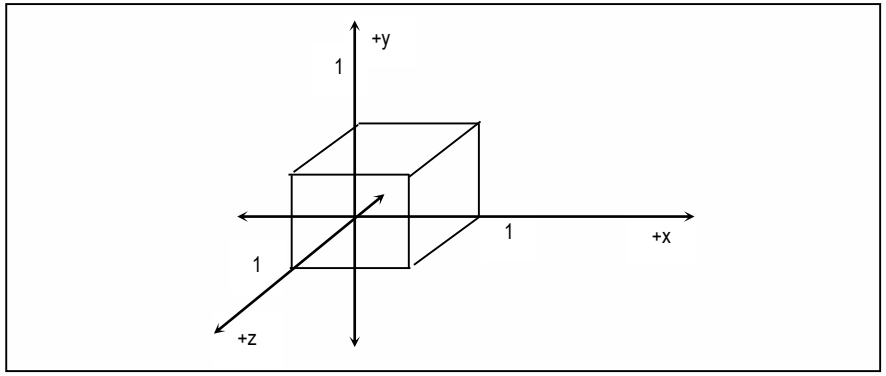
GLScene merupakan sekumpulan prosedur dan fungsi-fungsi yang disusun dalam bentuk komponen VCL (*visual components library*) yang merupakan suatu GLU (*Graphics Library Utility*) yang menjembatani antara bahasa pemograman dengan OpenGL Library. GLScene khusus digunakan untuk mengakses fungsi-fungsi dan prosedur grafis 3D yang ada pada OpenGL Library *(GL command)* untuk memudahkan dalam membuat suatu aplikasi 3D dalam bahasa pemograman visual Delphi (Grange, 2003).

Grafika Komputer 3D

Teknik grafika komputer 3D adalah cara membuat dan menampilkan objek grafis dalam bentuk 3-dimensi pada peralatan display (Hearn, et.al., 1994). Objek grafis yang paling sederhana adalah titik dan garis. Setiap objek grafis mempunyai atribut geometri dan atribut karakteristik lainnya seperti warna tekstur, tingkat transparansi dan lain-lain. Atribut geometri merupakan atribut dasar yang berisi informasi tentang keberadaan objek pada suatu sistem koordinat tertentu.

Pemodelan Objek 3D.

- Secara umum setiap objek grafis dibentuk dari gabungan beberapa objek grafis sederhana, misalnya 2 buah titik yang saling dihubungkan melalui suatu algoritma tertentu dapat membentuk suatu pola garis ataupun kurva tertentu.
- Objek grafis 3D juga dibentuk dari objek-objek grafis sederhana seperti titik dan garis. Objek grafis 3D dirancang berdasarkan sistem koordinat 3-dimensi.



Gambar 2.2 Pemodelan objek 3D

Representasi Objek 3D

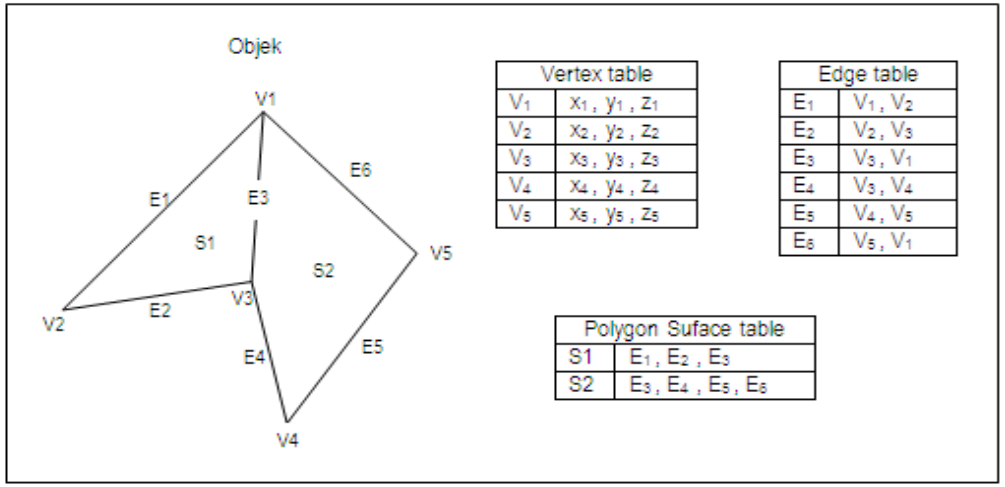
- Representasi objek adalah cara menjelaskan atau menggambarkan karakteristik suatu objek. Setiap objek grafis mempunyai karakteristik tertentu berupa atribut geometri dan berbagai atribut parameter karakteristik lainnya yang digunakan untuk menjelaskan tentang keberadaan serta ciri yang dimiliki objek tersebut. Setiap objek grafis akan ditampilkan berdasarkan informasi deskripsinya.
- Ada berbagai skema representasi objek 3D. Objek yang berbeda akan mempunyai skema representasi yang berbeda pula, misalnya representasi dengan polygon dan quadric surface sangat baik untuk merepresentasikan objek-objek sederhana seperti polyhedrons dan ellipsoids. Representasi spline surfaces baik digunakan untuk objek-objek berbentuk kurva seperti sayap pesawat terbang dan lain-lain.

Representasi Objek 3D

- Skema representasi untuk objek-objek solid umumnya dibagi dalam dua kategori. B-reps *(boundary representations)* menjelaskan objek 3D sebagai kumpulan dari suatu permukaan *(surface)* yang membatasi antara bagian objek sebelah dalam dengan lingkungannya.
- Sedangkan Space-partitioning representations digunakan untuk menjelaskan properti bagian sebelah dalam (bagian-bagian dari objek itu sendiri).
- Representasi untuk skema boundary yang umum digunakan adalah *polygon surfaces representation*. Polygon adalah objek grafis yang merupakan gabungan dari beberapa garis sehingga membentuk suatu pola atau shape tertutup. Representasi polygon surfaces menjelaskan Objek 3D sebagai kumpulan polygon yang menutupi bagian di permukaan objek.

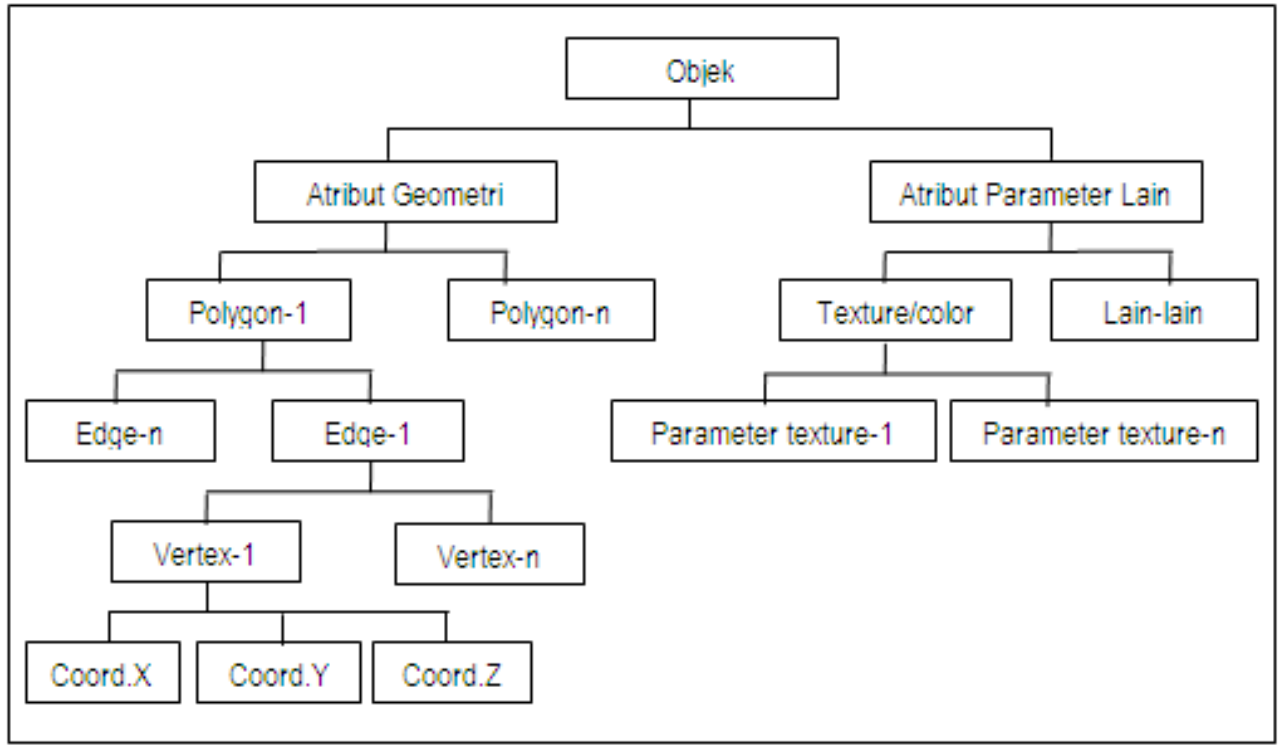
Representasi Objek 3D

- Dalam representasi polygon surfaces, setiap objek mempunyai karakteristik tertentu yang dinyatakan dalam satu atau lebih polygon. Setiap polygon mempunyai informasi karakteristik tertentu yang dinyatakan dalam sekumpulan garis *(edge).* Setiap garis mempunyai karakteristik tertentu yang dinyatakan dalam sekumpulan titik *(vertex).* Dan setiap titik dinyatakan dalam bentuk sekumpulan nilai koordinat (*x,y,z*). Semua informasi ini disimpan dalam masing-masing daftar tertentu seperti yang terlihat pada gambar 2.3 berikut ini :



Gambar 2.3 Representasi Polygon surfaces

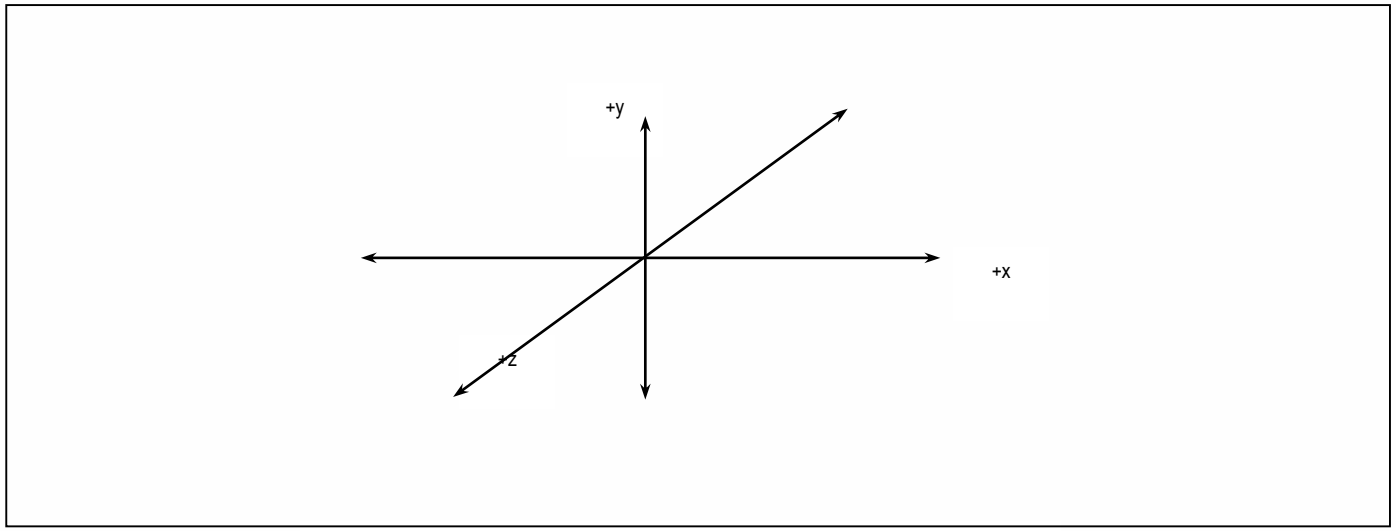
- Secara hirarki, struktur penyimpanan informasi deskripsi objek dalam polygon surfaces representation dalam gambar 2.4:



Gambar 2.4 Struktur Hirarki Representasi Polygon Surfaces

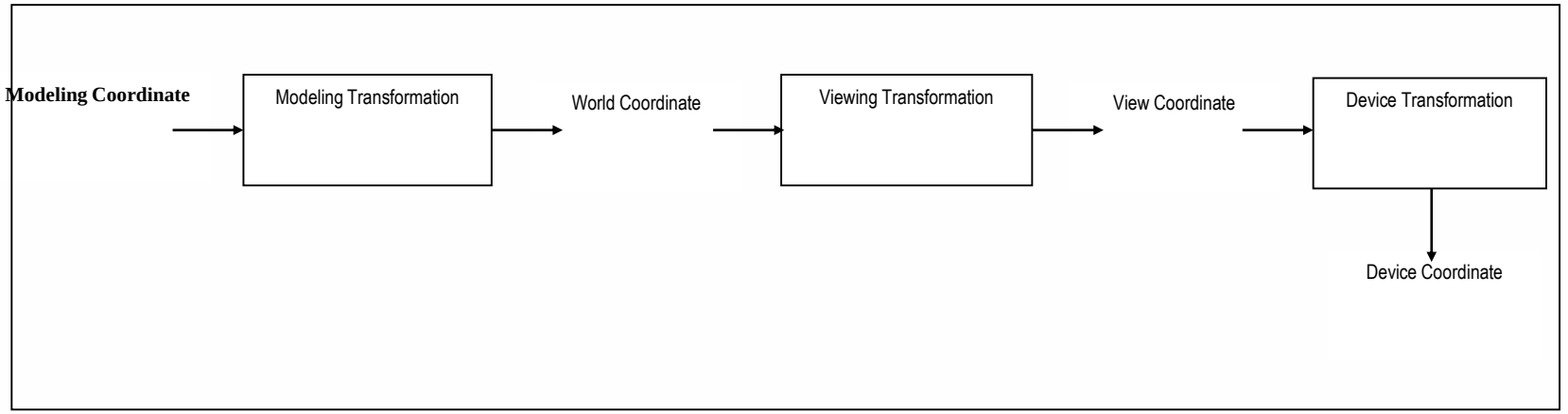
Sistem Koordinat 3D

- Pada dasarnya sistem koordinat 3-dimensi tidak berbeda dengan sistem koordinat 2-dimensi, yaitu sumbu-x secara horisontal, sumbu-y secara vertikal, dan menambahkan sumbu-z untuk menyatakan kedalaman.



Gambar 2.5 Sistem koordinat 3D

- Dalam grafika komputer 3D, untuk dapat menampilkan objek dalam tampilan 3-dimensi pada peralatan display, perlu dilakukan beberapa proses transformasi geometri dari sistem koordinat. Proses transformasi ini dilakukan dalam beberapa tahap seperti yang terlihat pada gambar.2.6 berikut :



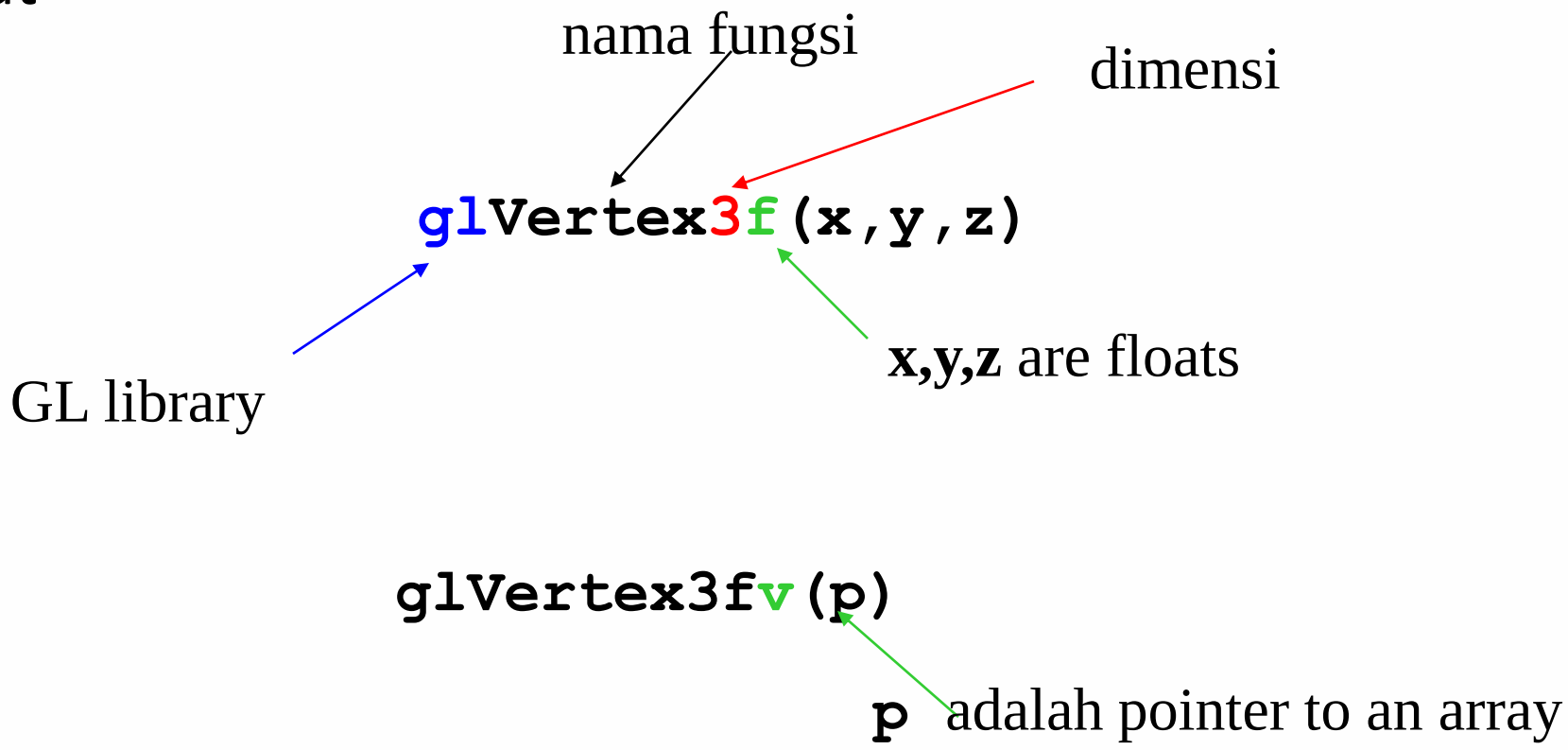
Gambar 2.6 Proses Transformasi Koordinat

Keterangan :

- Koordinat model *(modeling coordinates)* adalah sistem koordinat yang digunakan dalam pembentukan model.
- Koordinat lingkungan *(world coordinates)* adalah sistem koordinat yang digunakan pada lingkungan tempat objek diletakkan.
- Koordinat proyeksi atau penampakan *(viewing coordinate)* adalah sistem koordinat yang digunakan dalam memproyeksikan kenampakan objek berdasarkan posisi dan sudut pandang (*orientation*).
- Koordinat display *(device coordinate)* adalah sistem koordinat yang digunakan pada peralatan display.

OpenGL function format

- format



Contoh: simple.c

```
#include <GL/glut.h>
void mydisplay(){
  glClear(GL_COLOR_BUFFER_BIT);
  glBegin(GL_POLYGON);
        glVertex2f(-0.5, -0.5);
        glVertex2f(-0.5, 0.5);
        glVertex2f(0.5, 0.5);
        glVertex2f(0.5, -0.5);
  glEnd();
  glFlush();
}
int main(int argc, char** argv){
  glutCreateWindow("simple");
  glutDisplayFunc(mydisplay);
  glutMainLoop();
}
```

Event Loop

- program mendefinisikan *display callback* function dinamakan **`mydisplay`**
  - Setiap glut program harus punya *display callback*
  - *display callback* dijalankan kapan saja OpenGL memutuskan *display* harus di-*refresh*, contoh: ketika *window* dibuka
  - **`main`** *function* berakhir saat program *entering* suatu kejadian *loop*

Tutorial OpenGL

OpenGL adalah suatu API

- OpenGL **adalah** tidak lebih daru satu himpunan fungsi yang dipanggil dari program (kumpulan dari.h file(s)).
- Terdiri dari beberapa library dengan berbagai tingkat abstraksi: GL, GLU, and GLUT
- GL
  - Lowest level: vertex, matrix manipulation
  - glVertex3f(point.x, point.y, point.z)
- GLU
  - Helper functions for shapes, transformations
  - gluPerspective(fovy, aspect, near, far)
- GLUT
  - Highest level: Window and interface management
  - glutSwapBuffers()

OpenGL Implementations

- OpenGL IS an API (think of as collection of .h files):
  - #include <GL/gl.h>
  - #include <GL/glu.h>
  - #include <GL/glut.h>
- Windows, Linux, UNIX, etc. all provide a **platform specific** implementation.
- Windows: opengl32.lib glu32.lib glut32.lib
- Linux: -l GL -l GLU –l GLUT

OpenGL API

- As a programmer, you need to do the following things:
  - Specify the location/parameters of camera.
  - Specify the geometry (and appearance).
  - Specify the lights (optional).
- OpenGL will compute the resulting 2D image!

OpenGL Conventions

- Many functions have multiple forms:
  - glVertex2f, glVertex3i, glVertex4dv, etc.
- Number indicates number of arguments
- Letters indicate type
  - f: float, d: double, ub: unsigned byte, etc.
- ‘v’ (if present) indicates a single pointer argument
  - glVertex3f(point.x, point.y, point.z)
  - glVertex3fv(point)

OpenGL: Camera

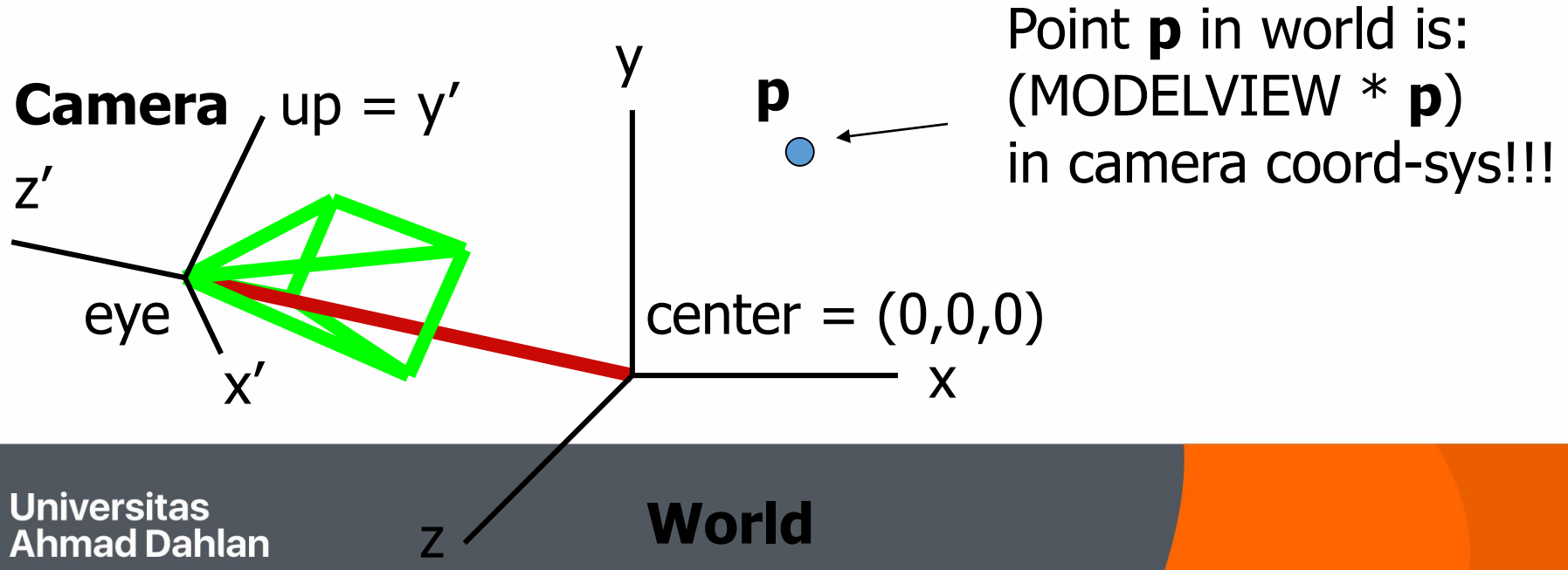
- Two things to specify:
  - Physical location of camera in the scene (MODELVIEW matrix in OpenGL).
    - Where is the camera?
    - Which direction is it pointing?
    - What is the orientation of the camera?
  - Projection properties of the camera (PROJECTION matrix in OpenGL):
    - Depth of field?
    - Field of view in the x and y directions?

OpenGL: Camera

- Sistem Koordinat direpresentasikan sebagai matriks dalam OpenGL.
- Kamera diimplikasikan sebagai suatu sistem koordinat dipusatkan pada bidang gambar.
- Oleh karena itu, menentukan matriks ini berarti menentukan sifat fisik kamera.

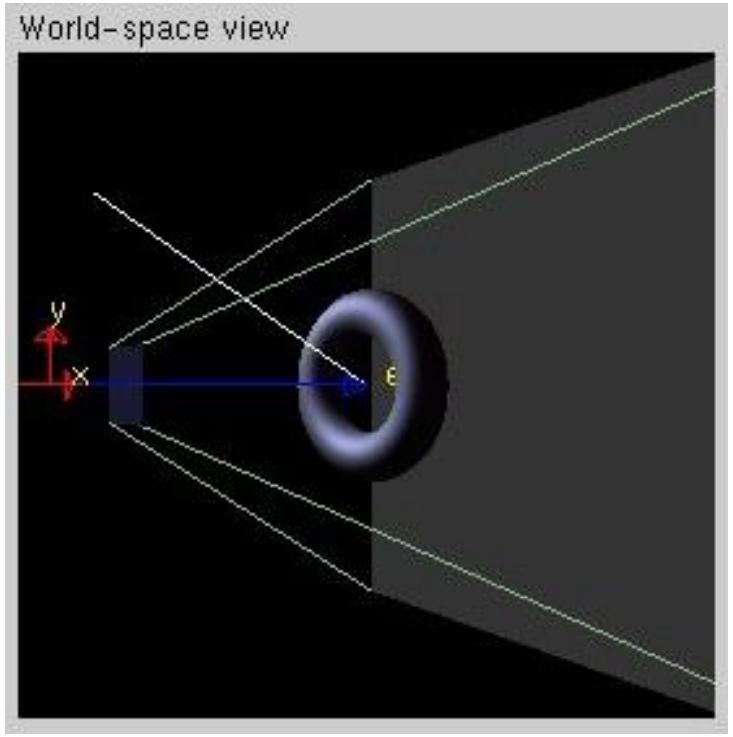
OpenGL: MODELVIEW

```
glMatrixMode(GL_MODELVIEW);  // Specify matrix mode
glLoadIdentity();                    // Clear modelview matrix
//Utility function provided by the GLU API (included with OpenGL)
gluLookAt(eyex,eyey,eyez,centerx,centery,centerz,upx,upy,upz);
```

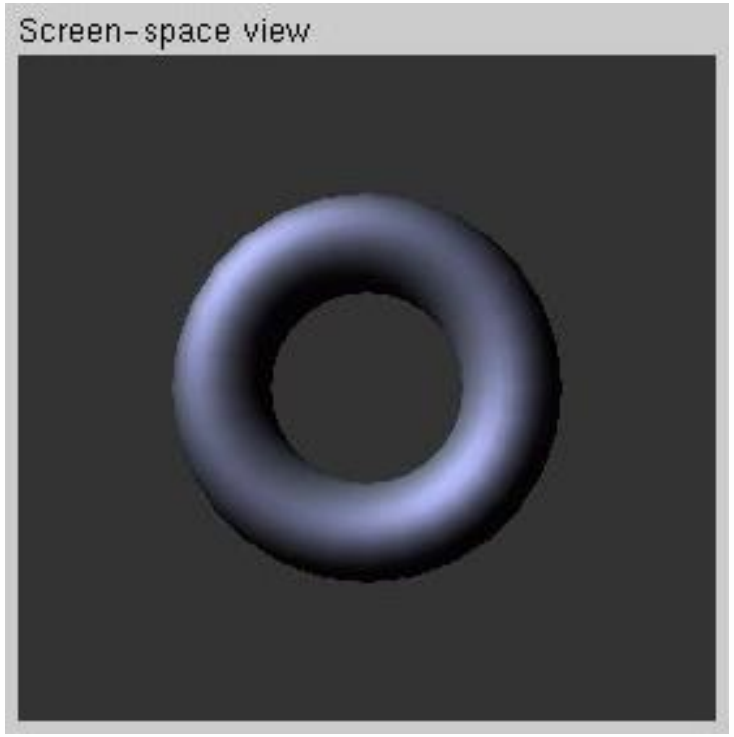


OpenGL: MODELVIEW

World coord-sys:

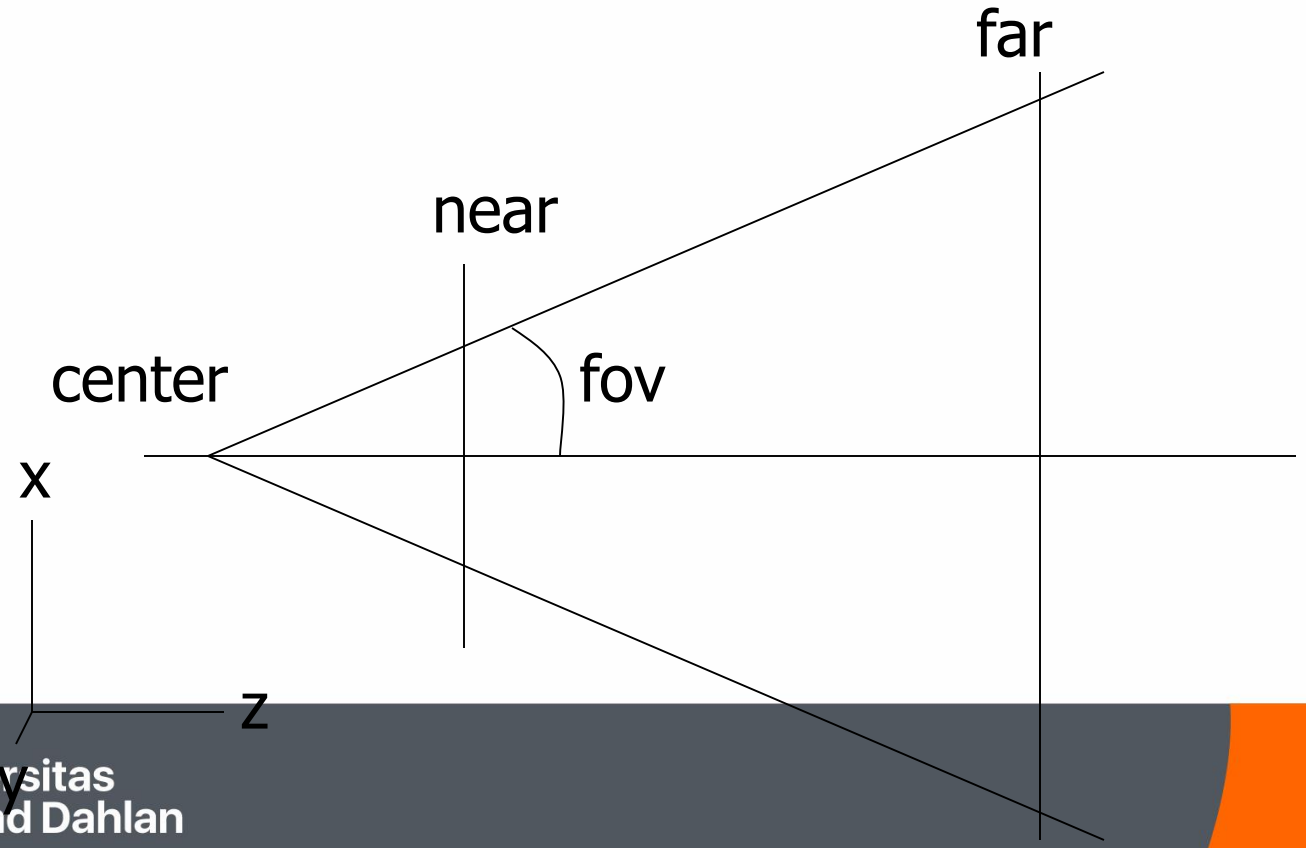


Camera coord-sys:



OpenGL: PROJECTION

- Intrinsic (optical) properties of camera:



OpenGL: PROJECTION

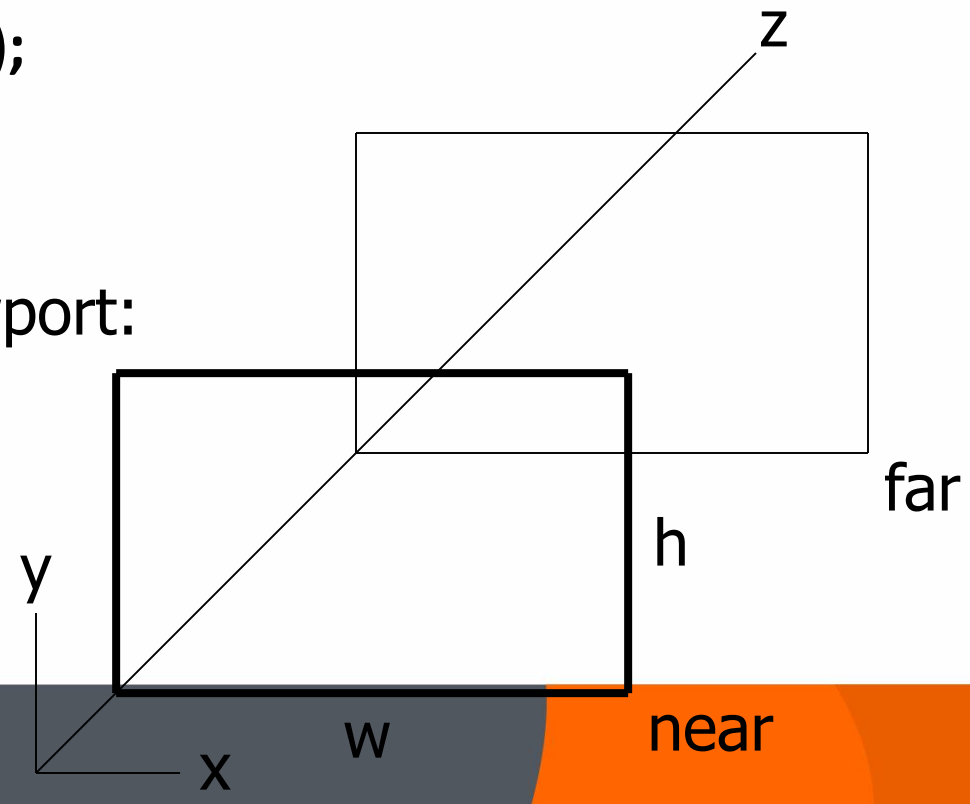
glMatrixMode(GL_PROJECTION);

glLoadIdentity();

gluPerspective(fovy, aspect, near, far);

Camera viewport:

aspect = w/h



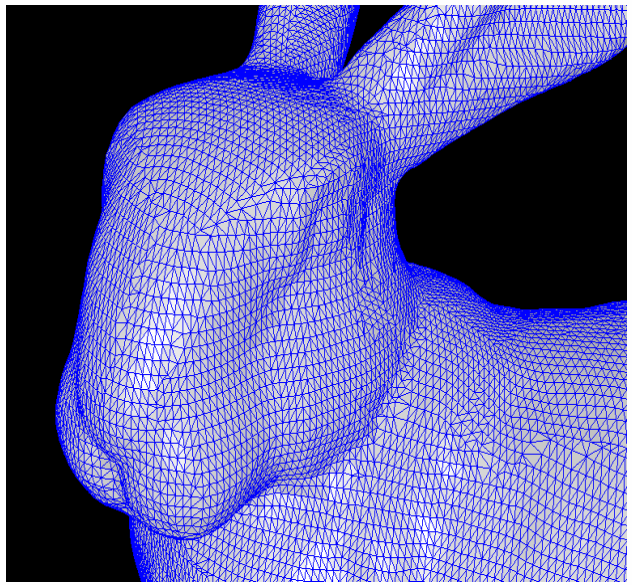
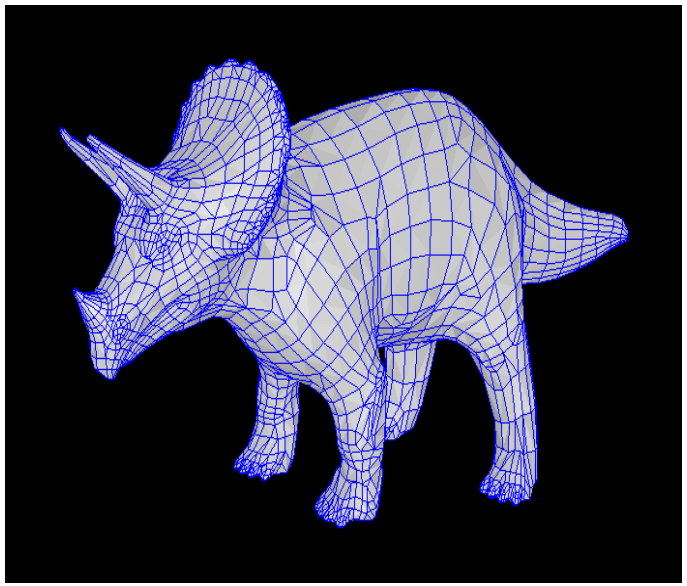
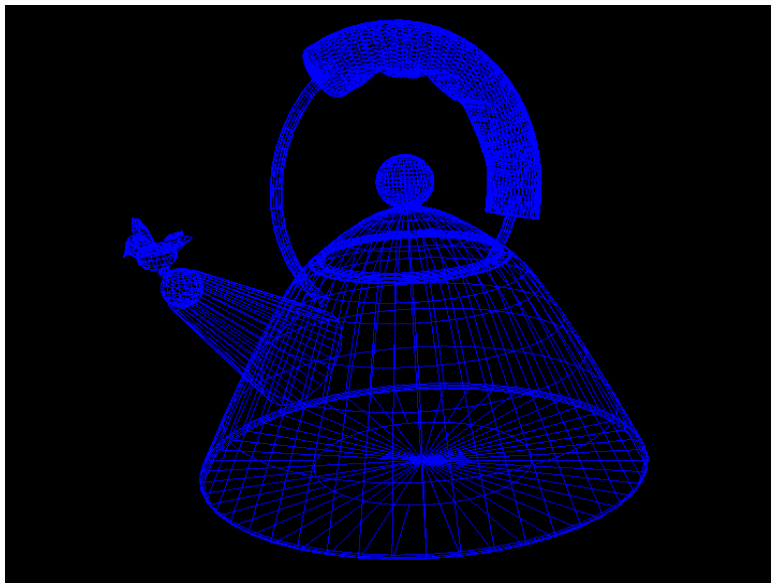
OpenGL: Setting Camera

- Assume window is **widthxheight**:

```
void SetCamera()
{
  glViewport(0, 0, width, height);
  /* Set camera position */
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  gluLookAt(m_vEye[0], m_vEye[1], m_vEye[2],
        m_vRef[0], m_vRef[1], m_vRef[2],
        m_vUp[0],  m_vUp[1],  m_vUp[2]);
  /* Set projection frustrum */
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluPerspective(m_fYFOV, width / height, m_fNear, m_fFar);
}
```

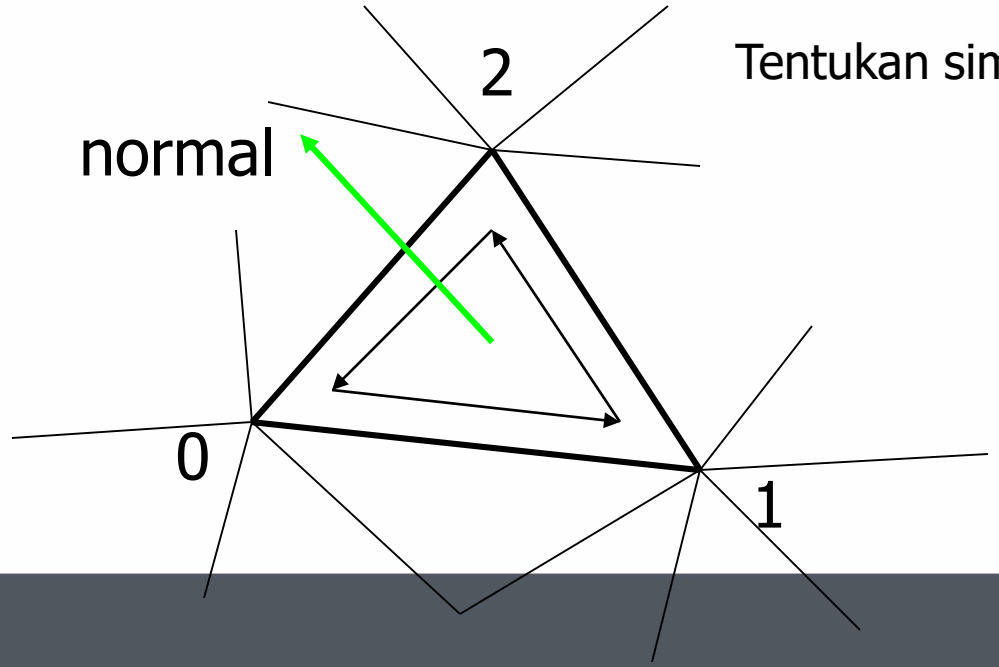
OpenGL: Geometry

- Tentukan geometri menggunakan primitif: segitiga, segi empat, garis, dll...

OpenGL: Triangle Mesh

- Merepresentasikan permukaan objek melalui sekumpulan dari **orientasi segitiga**:



Tentukan simpul sekitar cc normal

OpenGL: glBegin()...glEnd()

- Geometry passed btwn glBegin(.), glEnd(.)!!!

```
glBegin(GL_TRIANGLES);
for (int i=0; i<ntris; i++)
{
    glColor3f(tri[i].r0,tri[i].g0,tri[i].b0);          // Color of vertex
    glNormal3f(tri[i].nx0,tri[i].ny0,tri[i].nz0);    // Normal of vertex
    glVertex3f(tri[i].x0,tri[i].y0,tri[i].z0);         // Position of vertex
    …
    glColor3f(tri[i].r2,tri[i].g2,tri[i].b2);
    glNormal3f(tri[i].nx2,tri[i].ny2,tri[i].nz2);
    glVertex3f(tri[i].x2,tri[i].y2,tri[i].z2);
}
glEnd();   // Sends all the vertices/normals to the OpenGL library
```

OpenGL: glBegin()…glEnd()

- OpenGL mendukung beberapa primitif:

  glBegin(GL_LINES);

  glBegin(GL_QUADS);

  glBegin(GL_POLYGON);
- Gunakan GL_LINES untuk menentukan garis dalam ruang 3D (seperti sinar/ray).
- Semua vertek dapat punya suatu normal.

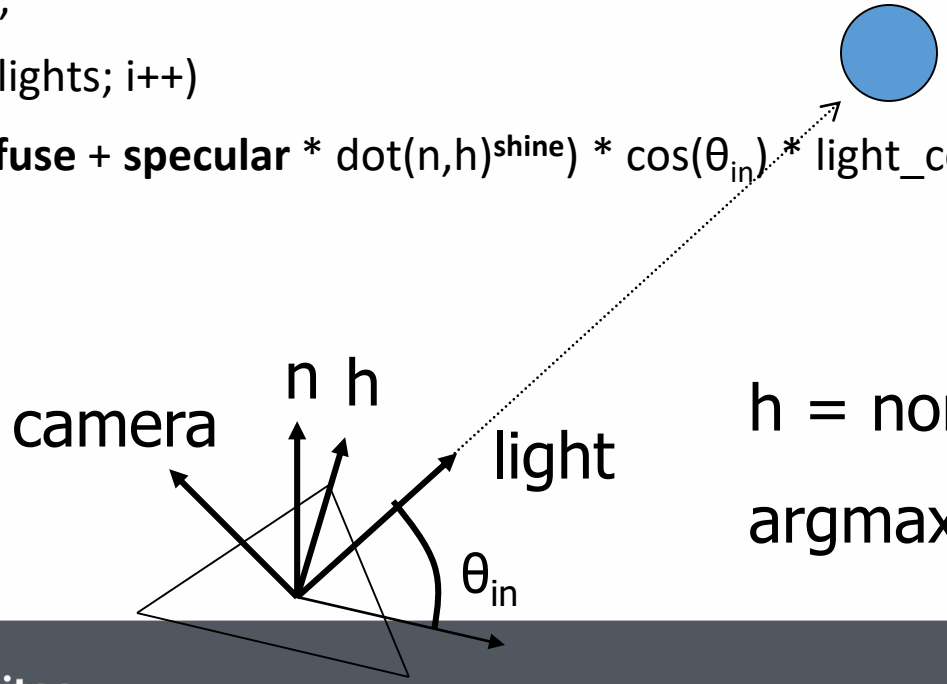
OpenGL: Lighting

- OpenGL menghitung warna pada setiap vertex dengan suatu penghitungan **shading**

color=**ambient**;

for (int i=0; i<nlights; i++)

color += (**diffuse** + **specular** * dot(n,h)$^{\textbf{shine}}$) * cos(θ_{in}) * light_color[i];



h = norm(light+camera)

argmax(dot(n,h)): h=n!

OpenGL: Lighting

```
GLfloat mat_diffuse[4] = {0.75, 0.75, 0.75, 1.0};
GLfloat mat_specular[4] = {0.55, 0.55, 0.55, 1.0};
GLfloat mat_shininess[1] = {80};

glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE,mat_diffuse);
glMaterialfv(GL_FRONT,GL_SPECULAR,mat_specular);
glMaterialfv(GL_FRONT,GL_SHININESS,mat_shininess);

GLfloat light0_ambient[4] = { 0.0, 0.0, 0.0, 1.0};
GLfloat light0_color[4] = { 0.4, 0.4, 0.4, 1.0 };
glLightfv(GL_LIGHT0, GL_AMBIENT, light0_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light0_color);
glLightfv(GL_LIGHT0, GL_SPECULAR, light0_color);

glEnable(GL_LIGHT0);
glEnable(GL_LIGHTING);
```

OpenGL: Lighting

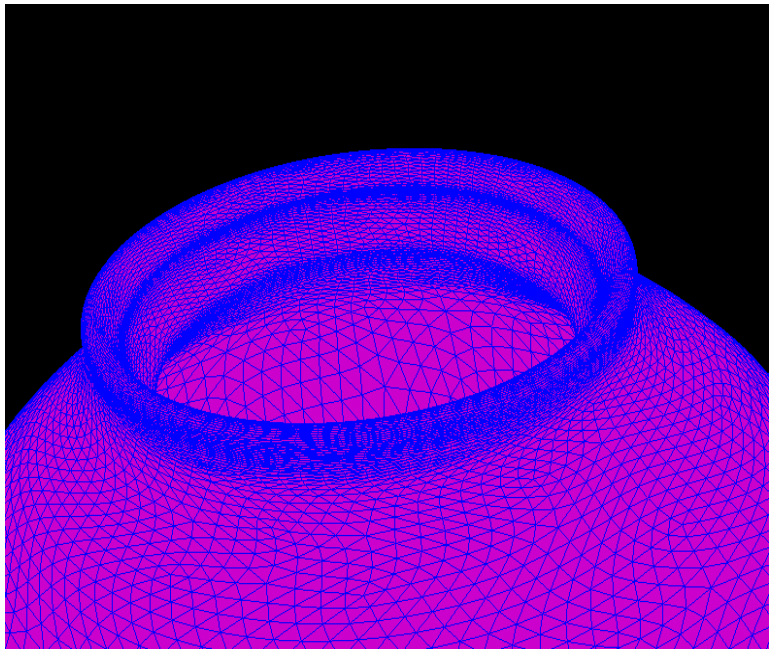
```
GLfloat mat_diffuse[4] = {0.75, 0.75, 0.75, 1.0};
GLfloat mat_specular[4] = {0.55, 0.55, 0.55, 1.0};
GLfloat mat_shininess[1] = {80};

glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT_AND_DIFFUSE,mat_diffuse);
glMaterialfv(GL_FRONT,GL_SPECULAR,mat_specular);
glMaterialfv(GL_FRONT,GL_SHININESS,mat_shininess);

GLfloat light0_ambient[4] = { 0.0, 0.0, 0.0, 1.0};
GLfloat light0_color[4] = { 0.4, 0.4, 0.4, 1.0 };
glLightfv(GL_LIGHT0, GL_AMBIENT, light0_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, light0_color);
glLightfv(GL_LIGHT0, GL_SPECULAR, light0_color);

glEnable(GL_LIGHT0);
glEnable(GL_LIGHTING);
```

OpenGL: Lighting

- There are two “shading modes” in OpenGL:
  - Flat shading: entire face is the color computed at the 0^{th} vertex.
  - Gouraud (“smooth”) shading: color computed at each vertex and interpolated across polygon.

OpenGL: Flat Shading

glShadeModel(GL_FLAT);



glShadeModel(GL_SMOOTH);

OpenGL: Lighting

- Lighting **merupakan suatu opsi**
  - glEnable(GL_LIGHTING);
  - glDisable(GL_LIGHTING);
- Saat disabled, warna dari tiap vertex secara langsung glColor3f(r,g,b).
- Tidak dibutuhkan *lighting* saat d*ebugging* menelusuri sinar (*ray tracer*).

GLUT

- GLUT is a **very simple** API that supports creating GUI+OpenGL apps.

  http://www.xmission.com/~nate/glut.html

GLUT

- Call these functions in your from main:

```
/* Initialize glut engine */
 glutInit(&argc,argv);
 glutInitWindowPosition(100,100);
 glutInitWindowSize(window_width,window_height);
 glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
 main_window = glutCreateWindow(“Ray Viewer");

 /* Register callbacks */
 glutDisplayFunc(MainRedraw);   // Main redraw function

 /* Run the interactive interface */
 glutMainLoop();
```

GLUT: MainRedraw

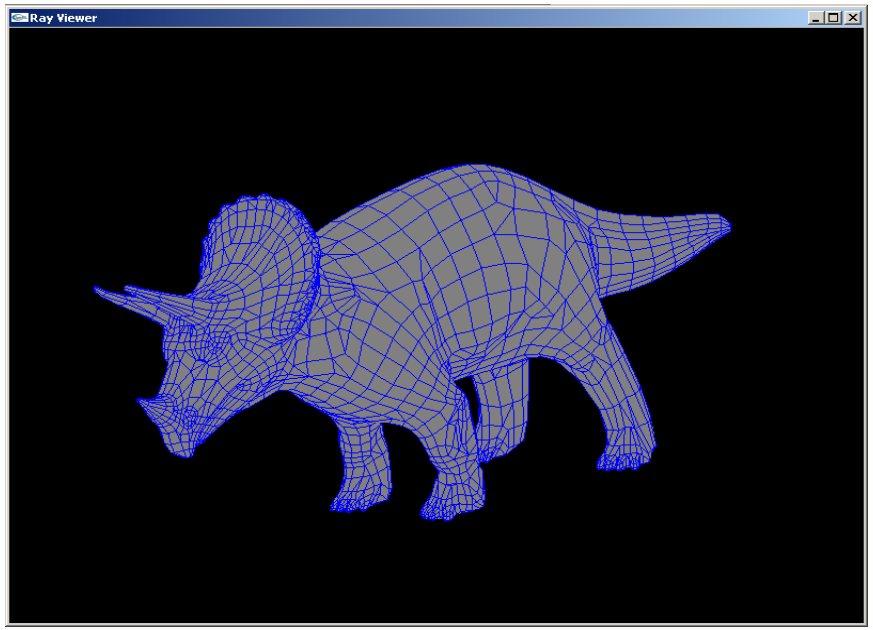
```
void MainRedraw()
{
   glClearColor(0,0,0,1);
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   SetCamera();
   DrawGeometry();
   glutSwapBuffers();  // Update the screen
}
```

GLUT: Example

- Skeleton code for a simple mesh viewer can be found at:

  http://www.cs.princeton.edu/~jlawrenc/cos426/mview.zip

- The program reads a “simple mesh” file format. The zip file includes several examples.



Referensi :

- Hills, Francis S Jr,*Computer Graphics Using OpenGl 2nd Edition*. New Jersey: Prentice Hall, 2000.
- Foley, James D, Andries van Dam, Steven K. Feiner and John F. Hughes, *Computer Graphics: Principles and Practice, Second Edition in C.* Addison-Wesley ,1994.
- Edward Angel ,*Interactive Computer Graphics: A Top-Down Approach with OpenGL*. 2nd , Addison Wesley, 2005
- Suyoto, *Teori dan Pemrograman Grafika Komputer dengan Visual C++ V.6 dan OpenGl*, Gava Media, Yogyakarta, 2003.
- https://sekolahinf.blogspot.com/2017/05/setting-codeblock-opengl-pemrograman.html
- http://www.xmission.com/~nate/tutors.html