MODUL 6
OBJEK 3D

A. KOMPETENSI DASAR

- Memahami Inisialisasi dunia 3D
- Memahami Object 3D (Wired).
- Memahami dasar menampilkan susunan objek 3D.

B. ALOKASI WAKTU

4 js (4x50 menit)

C. PETUNJUK

- Awali setiap aktivitas dengan do'a, semoga berkah dan mendapat kemudahan.
- Pahami Tujuan, dasar teori, dan latihan-latihan praktikum dengan baik dan benar.
- Kerjakan tugas-tugas dengan baik, sabar, dan jujur.
- Tanyakan kepada asisten/dosen apabila ada hal-hal yang kurang jelas.

D. DASAR TEORI

1. Bentuk *Wire*

Fungsi Wire merupakan implementasi dari object 3Dberupa kerangka benda yang berpusat pada asal pemodelan sistem koordinat. Utara dan kutub selatan bola berada di Z positif dan negatif sumbu-masing-masing dan meridian utama persimpangan sumbu X positif.

Berikut adalah list untuk bangun kerangka pada 3D:

a. Ukuran adalah panjang sisi

```
void glutWireCube(GLdouble size);
```

b. Fungsi glutWireSphere dan glutSolidSphere membuat bola berpusat pada asal pemodelan sistem koordinat. Utara dan kutub selatan bola berada di Z positif dan negatif sumbu-masing-masing dan meridian utama persimpangan sumbu X positif.

```
void glutWireSphere(GLdouble radius,GLint slices, GLint stacks);
```

c. Ukuran benda ditentukan dari dasar jari-jari alasnya.

```
void glutWireCone(GLdouble base, GLdouble height,GLint slices, GLint
stacks);
```

d. Render ditentukan melalui 12 sisi. Berpusat pada asal,dan dengan radius sama dengan kuadrat dari 3.

```
void glutWireTorus(GLdouble innerRadius, GLdouble outerRadius, GLint
nsides, GLint rings);
```

e. Renders padat atau wireframe 12-sisi biasa padat. Berpusat di sekitar asal dan dengan radius sama dengan akar kuadrat dari 3

```
void glutWireDodecahedron(void);
```

f. Renders padat atau wireframe 4-sisi biasa padat. Berpusat di sekitar asal dan dengan radius sama dengan akar kuadrat dari 3.

```
glutWiredTetrahedron void (void);
```

g. Renders padat atau wireframe 8-sisi biasa padat. Berpusat di sekitar asal dan dengan radius sebesar 1.0.

```
void glutWireOctahedr5(void);
```

h. Renders padat atau wireframe 20-sisi biasa padat. Berpusat di sekitar asal dan dengan radius sebesar 1.0.

```
void glutWireIcosahedron(void);
```

i. Render dengan membuat membuat poci teh

```
void glutWireTeapot(GLdouble size);
```

2. Gunakan beberapa fungsi tambahan untuk memanggil fungsi reshape dengan cara menambahkan fungsi `glutReshapeFunc (reshape);`. Fungs-fungsi di bawah adalah inisialisasi 3dimensi grafika komputer menggunakan openGl.

```
void display (void) {
glClearColor (0.0,0.0,0.0,0.0);
glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glLoadIdentity();
glTranslatef(X,Y,Z);
cube();
glutSwapBuffers();
}
```

Kemudian tambahkan fungsi

```
void reshape (int w, int h) {
glViewport (0, 0, (GLsizei)w, (GLsizei)h);
glMatrixMode (GL_PROJECTION);
glLoadIdentity ();
gluPerspective (60, (GLfloat)w / (GLfloat)h, 1.0, 100.0);
glMatrixMode (GL_MODELVIEW);
}
```

```
void init(){
    glShadeModel(GL_SMOOTH);
    glClearColor(0.0f,0.0f,0.0f,0.5f);
    glClearDepth(1.0f);
    glEnable(GL_DEPTH_TEST);
    glDepthFunc(GL_LEQUAL);
    glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);

    return;
}
```

E. AKTIFITAS KELAS PRAKTIKUM

1. Buatlah sebuah fungsi yang dapat membuat objek 2D. tampilkan objek 2 D tersebut pada layar dan berikan interaksi menggunakan keyboard untuk memindahkan posisi objek pada KOORDINAT Z (mendekat dan menjauh). Tampilkan 2 *screenshot* kondisi ketika objek 2D jauh dan dekat. Dan Tulis program (display function).

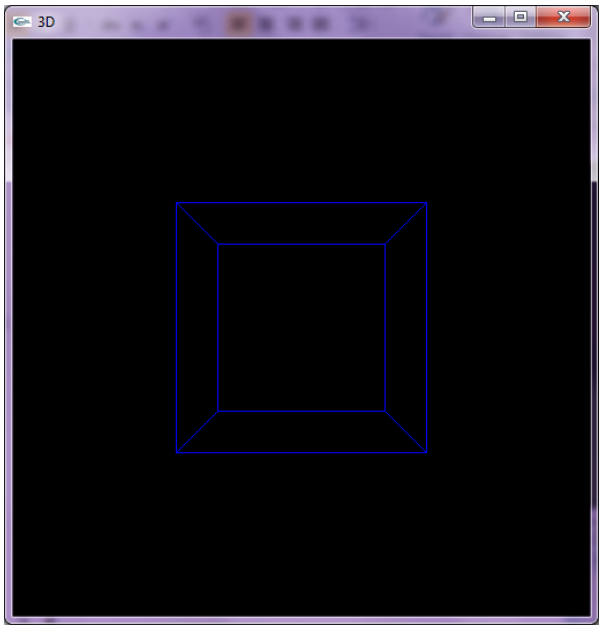
```
#include "stdlib.h"
#include "gl/glut.h"
int w=400, h=400, z=0;
int x1=0, y1=0, sudut=0, z1=0;
void renderScene(void){
 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
 glClearColor(1,1,1,1);
 glLoadIdentity();
 glTranslatef(0,0,z);
 glRotatef(sudut,x1,y1,z1);
 glColor3f(1,0,0);
 glutWireCube(3);
 glutSwapBuffers();
}
void resize(int w1, int h1){
 glViewport(0,0,w1,h1);
 glMatrixMode(GL_PROJECTION);
 glLoadIdentity();
 gluPerspective(45.0,(float) w1/(float) h1, 1.0,300.0);
 glMatrixMode(GL_MODELVIEW);
 glLoadIdentity();
}
void myKeyboard(unsigned char key, int x, int y){
 if (key =='a') z+=5;
 else if (key == 'd') z-=5;
 else if (key == 'x') {
  x1=1;
  y1=0;
  z1=0;
  sudut+=10;
 }
 else if (key == 'y') {
  y1=1;
  x1=0;
  z1=0;
```

```
  sudut+=-10;
 }
 else if (key == 'z') {
  y1=0;
  x1=0;
  z1=1;
  sudut+=-10;
 }
}
void init(){
 glClearColor(0,0,0,1);
 glEnable(GL_DEPTH_TEST);
 glMatrixMode(GL_PROJECTION);
 glLoadIdentity();
 gluPerspective(45.0,(GLdouble) w/(GLdouble) h,
1.0,300.0);
 glMatrixMode(GL_MODELVIEW);
}
void timer(int value){
 glutPostRedisplay();
 glutTimerFunc(50,timer,0);
}
void main (int argc, char **argv){
 glutInit(&argc, argv);
 glutInitDisplayMode(GLUT_DOUBLE | GLUT_DEPTH |
GLUT_RGBA);
 glutInitWindowPosition(100,100);
 glutInitWindowSize(w,h);
 glutCreateWindow("3D");
 gluOrtho2D(-w/2,w/2,-h/2,h/2);
 glutDisplayFunc(renderScene);
 glutReshapeFunc(resize);
 glutKeyboardFunc(myKeyboard);
 glutTimerFunc(1,timer,0);
 init();
 glutMainLoop();
}
```

2. Buat juga sebuah program yang dapat menganimasikan objek 2D tersebut menggunakan tombol (‘x’ = rotasi pada sumbu x, ‘y’ = rotasi pada sumbu y). Sajikan *screenshot* hasil interaksi yang menunjukkan animasi rotasi-rotasi tersebut dan tulis program pada fungsi *display*.
3. Buatlah sebuah program dengan menggunakan fungsi WireCube
   Tambahkan fungsi berikut

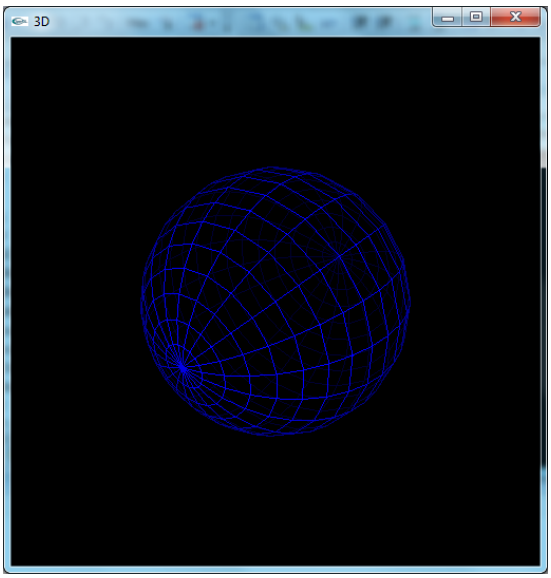
```
void cube (void) {
    glColor3f(r,g,b);
    glutWireCube(GLdouble Size);
}
```

Tampilkan ScreenShoot Seperti di bawah ini



4. Buatlah sebuah program dengan menggunakan fungsi WireSphere

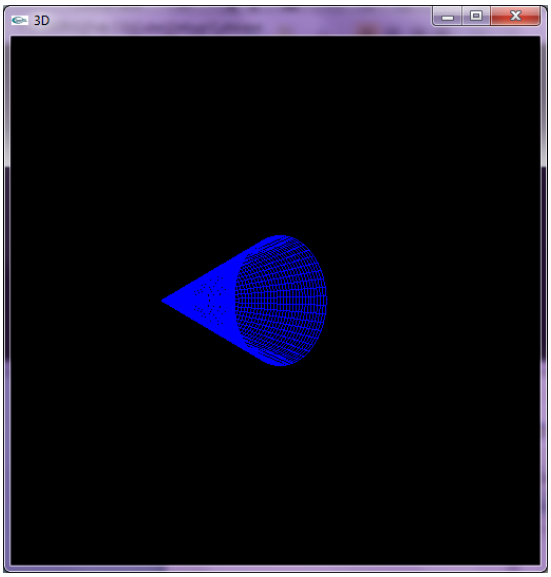
```
void renderScene(void){
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glClearColor(1,1,1,1);
glLoadIdentity();
glTranslatef(0,0,z);
glRotatef(sudut,x1,y1,z1);
glColor3f(0,0,1);
glutWireSphere(2,20,10);
glutSwapBuffers();
```




```
void sphere (void) {
     glColor(r,g,b);
     void glutWireSphere(GLdouble radius,
                         GLint slices, GLint stacks);
 }
```

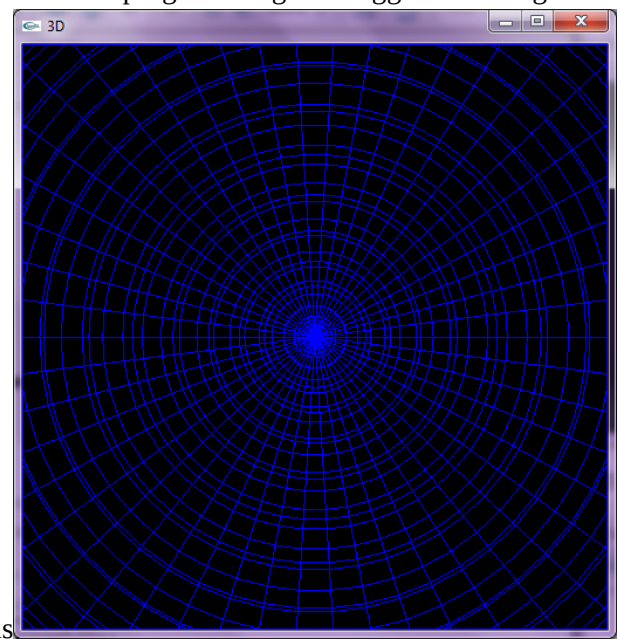
3. Buatlah sebuah program dengan menggunakan fungsi WireCone

```
void WireCone (void) {
     glColor3f(r,g,b);
     void glutWireCone(GLdouble base, GLdouble height,
                       GLint slices, GLint stacks);
}
```

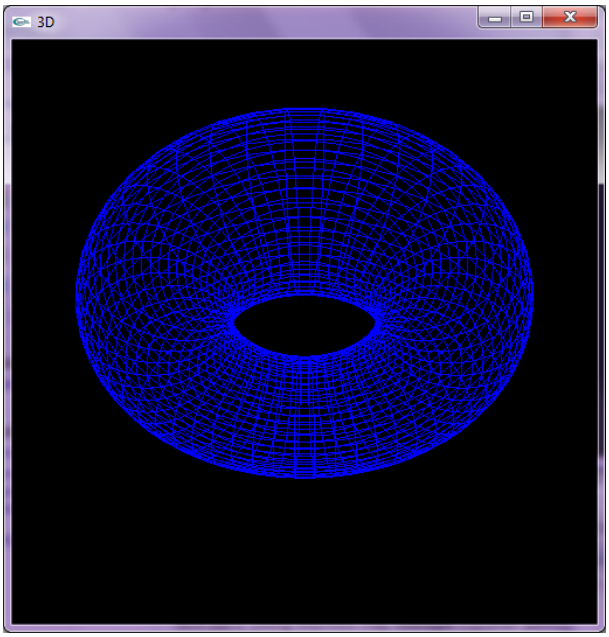


```
void renderScene(void){
 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
 glClearColor(1,1,1,1);
 glLoadIdentity();
 glTranslatef(0,0,z);
 glRotatef(sudut,x1,y1,z1);
 glColor3f(1,0,0);
 glutWireCone(2,4,25,25);
 glutSwapBuffers();
}
```

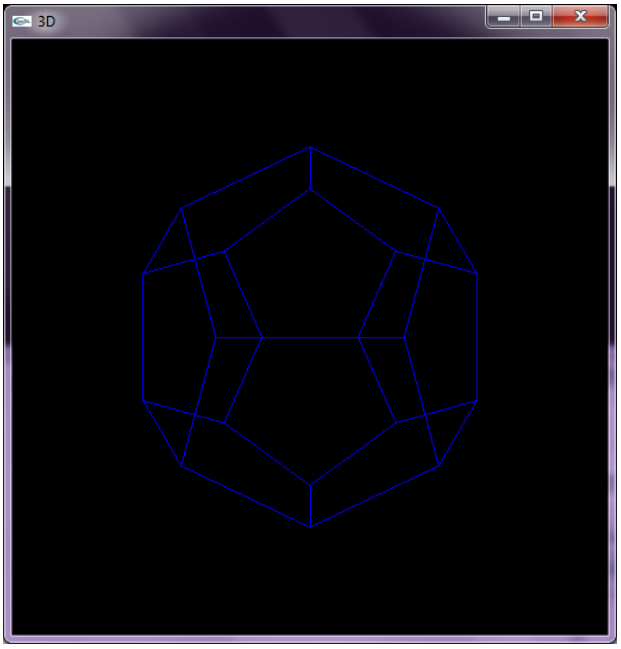
4. Buatlah sebuah program dengan menggunakan fungsi WireTorus



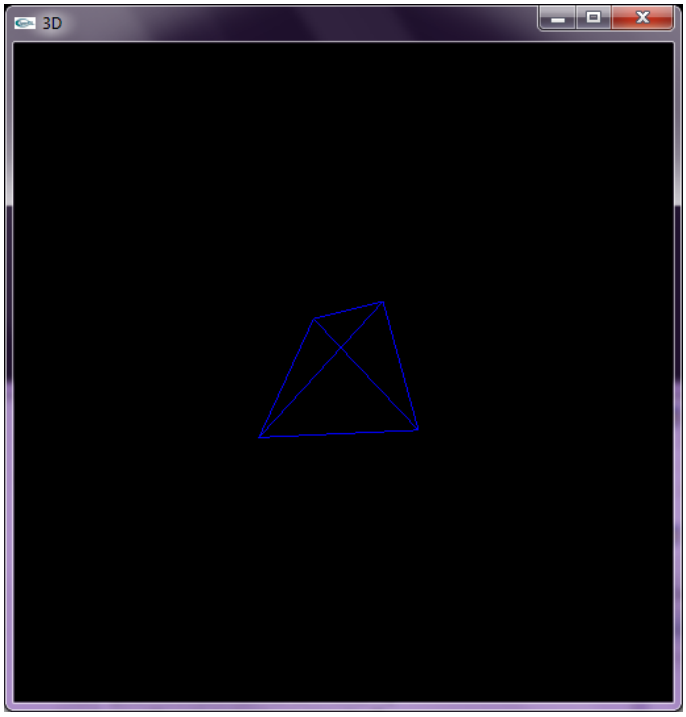

```
 void renderScene(void){
 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
 glClearColor(1,1,1,1);
 glLoadIdentity();
 glTranslatef(0,0,z);
 glRotatef(sudut,x1,y1,z1);
 glColor3f(0,0,1);
 glutWireTorus(1,3,20,40);
 glutSwapBuffers();
}
```



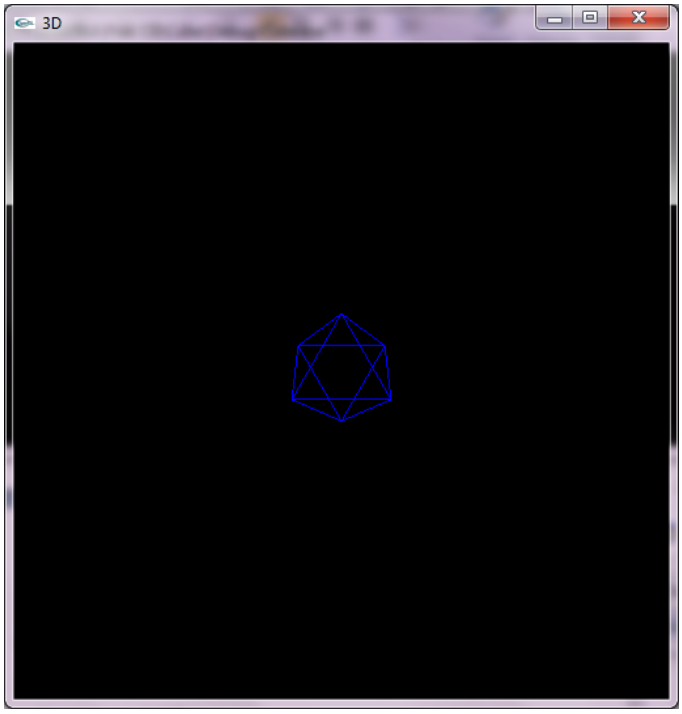
5. Buatlah sebuah program dengan menggunakan fungsi WireDodecahedron



6. Buatlah sebuah program dengan menggunakan fungsi WireTetrahedron

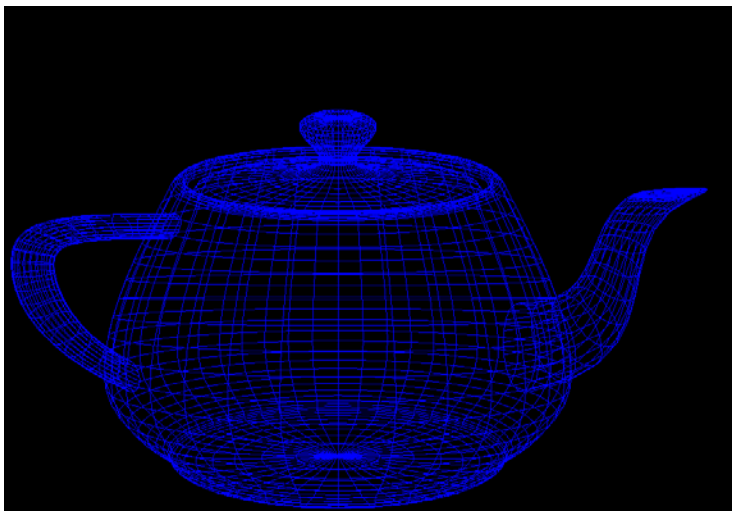

3D

7. Buatlah sebuah program dengan menggunakan fungsi WireOctahedron



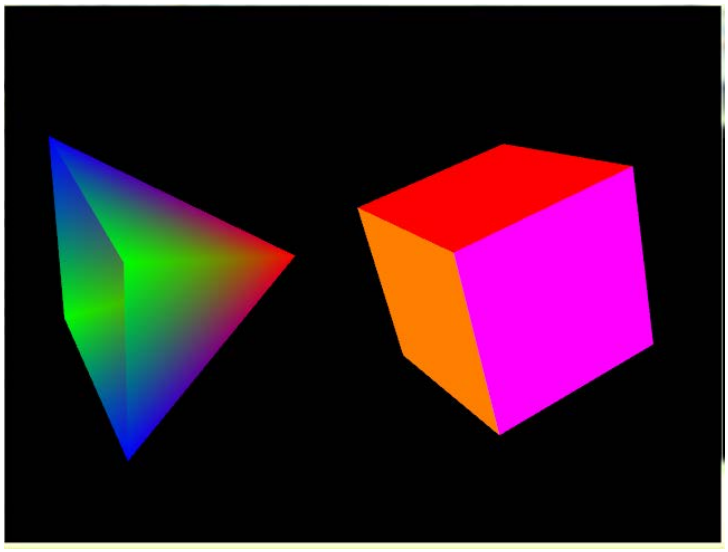
8. Buatlah sebuah program dengan menggunakan fungsi WireTeapot

```
void teapot (void) {
    glColor(r,g,b);
    glutWireTeapot(GLDouble Size);
}
```



```
void renderScene(void){
 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
 glClearColor(1,1,1,1);
 glLoadIdentity();
 glTranslatef(0,0,z);
 glRotatef(sudut,x1,y1,z1);
 glColor3f(1,0,0);
 glutWireTeapot(4);
 glutSwapBuffers();
}
```

Object 3 Dimensi Menggunakan Animasi



```
/*
 * OGL02Animation.cpp: 3D Shapes with animation
 */
#include <windows.h>  // for MS Windows
#include <GL/glut.h>  // GLUT, include glu.h and gl.h

/* Global variables */
char title[] = "3D Shapes with animation";
GLfloat anglePyramid = 0.0f;  // Rotational angle for pyramid [NEW]
GLfloat angleCube = 0.0f;     // Rotational angle for cube [NEW]
int refreshMills = 15;        // refresh interval in milliseconds [NEW]

/* Initialize OpenGL Graphics */
```

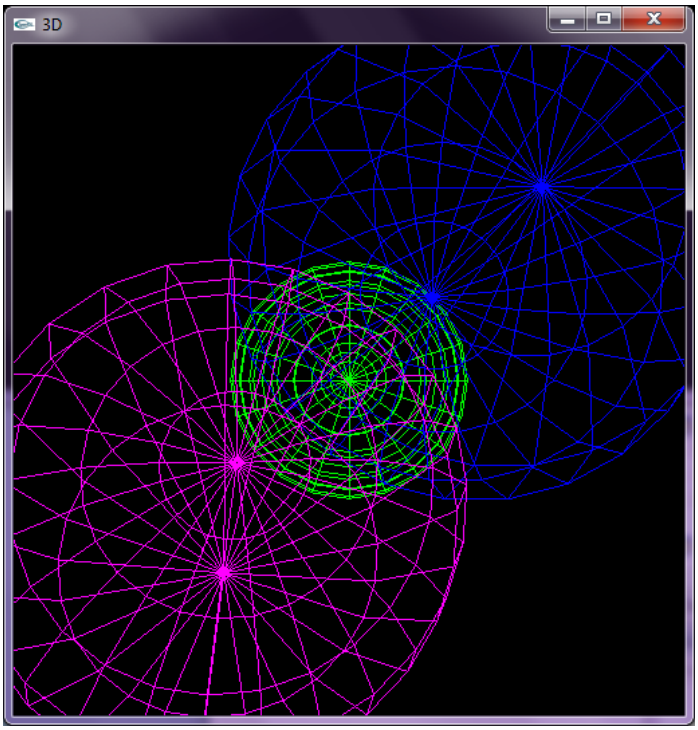
```
void initGL() {
   glClearColor(0.0f, 0.0f, 0.0f, 1.0f); // Set background color to black and opaque
   glClearDepth(1.0f);                   // Set background depth to farthest
   glEnable(GL_DEPTH_TEST);   // Enable depth testing for z-culling
   glDepthFunc(GL_LEQUAL);    // Set the type of depth-test
   glShadeModel(GL_SMOOTH);   // Enable smooth shading
   glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);  // Nice perspective corrections
}

/* Handler for window-repaint event. Called back when the window first appears and
   whenever the window needs to be re-painted. */
void display() {
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // Clear color and depth buffers
   glMatrixMode(GL_MODELVIEW);     // To operate on model-view matrix

   // Render a color-cube consisting of 6 quads with different colors
   glLoadIdentity();                 // Reset the model-view matrix
   glTranslatef(1.5f, 0.0f, -7.0f);  // Move right and into the screen
   glRotatef(angleCube, 1.0f, 1.0f, 1.0f);  // Rotate about (1,1,1)-axis [NEW]

   glBegin(GL_QUADS);                // Begin drawing the color cube with 6 quads
      // Top face (y = 1.0f)
      // Define vertices in counter-clockwise (CCW) order with normal pointing out
      glColor3f(0.0f, 1.0f, 0.0f);     // Green
      glVertex3f( 1.0f, 1.0f, -1.0f);
      glVertex3f(-1.0f, 1.0f, -1.0f);
      glVertex3f(-1.0f, 1.0f,  1.0f);
      glVertex3f( 1.0f, 1.0f,  1.0f);

      // Bottom face (y = -1.0f)
      glColor3f(1.0f, 0.5f, 0.0f);     // Orange
      glVertex3f( 1.0f, -1.0f,  1.0f);
      glVertex3f(-1.0f, -1.0f,  1.0f);
      glVertex3f(-1.0f, -1.0f, -1.0f);
      glVertex3f( 1.0f, -1.0f, -1.0f);

      // Front face  (z = 1.0f)
      glColor3f(1.0f, 0.0f, 0.0f);     // Red
      glVertex3f( 1.0f,  1.0f, 1.0f);
      glVertex3f(-1.0f,  1.0f, 1.0f);
      glVertex3f(-1.0f, -1.0f, 1.0f);
      glVertex3f( 1.0f, -1.0f, 1.0f);

      // Back face (z = -1.0f)
      glColor3f(1.0f, 1.0f, 0.0f);     // Yellow
      glVertex3f( 1.0f, -1.0f, -1.0f);
      glVertex3f(-1.0f, -1.0f, -1.0f);
      glVertex3f(-1.0f,  1.0f, -1.0f);
      glVertex3f( 1.0f,  1.0f, -1.0f);

      // Left face (x = -1.0f)
      glColor3f(0.0f, 0.0f, 1.0f);     // Blue
      glVertex3f(-1.0f,  1.0f,  1.0f);
      glVertex3f(-1.0f,  1.0f, -1.0f);
```

```
      glVertex3f(-1.0f, -1.0f, -1.0f);
      glVertex3f(-1.0f, -1.0f,  1.0f);

      // Right face (x = 1.0f)
      glColor3f(1.0f, 0.0f, 1.0f);     // Magenta
      glVertex3f(1.0f,  1.0f, -1.0f);
      glVertex3f(1.0f,  1.0f,  1.0f);
      glVertex3f(1.0f, -1.0f,  1.0f);
      glVertex3f(1.0f, -1.0f, -1.0f);
   glEnd();  // End of drawing color-cube

   // Render a pyramid consists of 4 triangles
   glLoadIdentity();                  // Reset the model-view matrix
   glTranslatef(-1.5f, 0.0f, -6.0f);  // Move left and into the screen
   glRotatef(anglePyramid, 1.0f, 1.0f, 0.0f);  // Rotate about the (1,1,0)-axis [NEW]

   glBegin(GL_TRIANGLES);           // Begin drawing the pyramid with 4 triangles
      // Front
      glColor3f(1.0f, 0.0f, 0.0f);     // Red
      glVertex3f( 0.0f, 1.0f, 0.0f);
      glColor3f(0.0f, 1.0f, 0.0f);     // Green
      glVertex3f(-1.0f, -1.0f, 1.0f);
      glColor3f(0.0f, 0.0f, 1.0f);     // Blue
      glVertex3f(1.0f, -1.0f, 1.0f);

      // Right
      glColor3f(1.0f, 0.0f, 0.0f);     // Red
      glVertex3f(0.0f, 1.0f, 0.0f);
      glColor3f(0.0f, 0.0f, 1.0f);     // Blue
      glVertex3f(1.0f, -1.0f, 1.0f);
      glColor3f(0.0f, 1.0f, 0.0f);     // Green
      glVertex3f(1.0f, -1.0f, -1.0f);

      // Back
      glColor3f(1.0f, 0.0f, 0.0f);     // Red
      glVertex3f(0.0f, 1.0f, 0.0f);
      glColor3f(0.0f, 1.0f, 0.0f);     // Green
      glVertex3f(1.0f, -1.0f, -1.0f);
      glColor3f(0.0f, 0.0f, 1.0f);     // Blue
      glVertex3f(-1.0f, -1.0f, -1.0f);

      // Left
      glColor3f(1.0f,0.0f,0.0f);       // Red
      glVertex3f( 0.0f, 1.0f, 0.0f);
      glColor3f(0.0f,0.0f,1.0f);       // Blue
      glVertex3f(-1.0f,-1.0f,-1.0f);
      glColor3f(0.0f,1.0f,0.0f);       // Green
      glVertex3f(-1.0f,-1.0f, 1.0f);
   glEnd();   // Done drawing the pyramid

   glutSwapBuffers();  // Swap the front and back frame buffers (double buffering)

   // Update the rotational angle after each refresh [NEW]
   anglePyramid += 0.2f;
```

```
   angleCube -= 0.15f;
}

/* Called back when timer expired [NEW] */
void timer(int value) {
   glutPostRedisplay();      // Post re-paint request to activate display()
   glutTimerFunc(refreshMills, timer, 0); // next timer call milliseconds later
}

/* Handler for window re-size event. Called back when the window first appears and
   whenever the window is re-sized with its new width and height */
void reshape(GLsizei width, GLsizei height) {  // GLsizei for non-negative integer
   // Compute aspect ratio of the new window
   if (height == 0) height = 1;                // To prevent divide by 0
   GLfloat aspect = (GLfloat)width / (GLfloat)height;

   // Set the viewport to cover the new window
   glViewport(0, 0, width, height);

   // Set the aspect ratio of the clipping volume to match the viewport
   glMatrixMode(GL_PROJECTION);  // To operate on the Projection matrix
   glLoadIdentity();             // Reset
   // Enable perspective projection with fovy, aspect, zNear and zFar
   gluPerspective(45.0f, aspect, 0.1f, 100.0f);
}

/* Main function: GLUT runs as a console application starting at main() */
int main(int argc, char** argv) {
   glutInit(&argc, argv);            // Initialize GLUT
   glutInitDisplayMode(GLUT_DOUBLE); // Enable double buffered mode
   glutInitWindowSize(640, 480);   // Set the window's initial width & height
   glutInitWindowPosition(50, 50); // Position the window's initial top-left corner
   glutCreateWindow(title);          // Create window with the given title
   glutDisplayFunc(display);       // Register callback handler for window re-paint event
   glutReshapeFunc(reshape);       // Register callback handler for window re-size event
   initGL();                       // Our own OpenGL initialization
   glutTimerFunc(0, timer, 0);     // First timer call immediately [NEW]
   glutMainLoop();                 // Enter the infinite event-processing loop
   return 0;
}
```

F. TUGAS ASISTENSI

1. Buatlah kesimpulan dari masing masing fungsi wire, mulai dari `glutWireCone`, `glutWireCube`, `glutWireTorus`, `glutWireDodecahedron`, `glutWireTeapot`, `glutWireOctahedron`, `glutWireTetrahedron`, dan `glutWireIcosahedron`.
2. Buatlah bentuk bangun sebagai berikut:



3. Buatlah Manusia Salju 3 dimensi (menggunakan wire). Sajikan source code dan screenshotnya.