MEMBANGUN APLIKASI ANIMASI 3D

BASKETMAN

Dikerjakan untuk memenuhi
Tugas Akhir Mata Kuliah Grafika Komputer

Dikerjakan Oleh :

Muhammad Firyanul Rizky (1708561006)

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN ILMU KOMPUTER

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS UDAYANA

2019

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Bola basket adalah salah satu cabang olahraga yang termasuk populer dan banyak digemari oleh masyarakat Indonesia. Permainan bola basket memiliki karakteristik tersendiri, antara lain kategori permainan yang mempergunakan bola besar, lapangan yang luas dan mempunyai papan pantul serta ring untuk memasukkan bola, yang terdiri atas dua tim yang beranggotakan masing - masing lima orang yang saling bertanding mencetak poin dengan memasukkan bola ke dalam keranjang lawan dan mencegah lawan untuk memasukkan bola ke keranjang sendiri. Bola basket sangat menarik untuk ditonton karena bisa dimainkan di ruang olahraga tertutup dan terbuka (indoor atau outdoor), serta hanya memerlukan lapangan yang relatif kecil. Bola basket banyak digemari oleh masyarakat, terutama kalangan pelajar dan mahasiswa. Melalui kegiatan olahraga bola basket ini para remaja banyak memperoleh manfaat khususnya dalam pertumbuhan fisik, mental, dan sosial, selain itu bola basket mudah dipelajari karena bentuk bolanya yang besar, sehingga tidak menyulitkan pemain ketika memantulkan atau melempar bola tersebut.

Maka daripada itu kelompok kami membangun sebuah game 3D sederhana terkait mensimulasikan permainan basket tersebut dengan mnegmplementasikan OPENGL.

1.2 Rumusan Masalah

Aplikasi apa yang perlu dibuat untuk mensimulasikan gerakan seorang atlet basket dalam melambungkan bola ke tiang net lawan ?

1.3 Maksud dan Tujuan

Maksud dari pembuatan tugas besar ini adalah Membuat simulasi 3D sederhana terkait permainan bola basket.

Tujuan dari tugas besar ini adalah :

- Mensimulasikan lambung bola ke arah net/jaring lawan

1.4 Batasan Masalah

Berikut ini dijelaskan batasan masalah dari pembuatan aplikasi 3D Sederhana Basketman.

1. Aplikasi dibangun menggunakan bahasa pemrograman C++ dengan tool Dev C++ dan library OpenGL GLUT.
2. Objek yang terdapat dalam aplikasi antara lain :
   a. Lokasi berada di sebuah lapangan basket mungil.
   b. Jumlah pemain hanya seorang.

BAB 2 LANDASAN TEORI

2.1 Story Board

Seorang pemuda cupu yang sangat terinspirasi menjadi pemain basket, kami bernama basketman sedang melatih dirinya seorang diri melambungkan bola basket ke arah tiang net, tujuannya yang tak lain dan tak bukan adalah melatih dirinya guna membuktikan ke teman-teman sepergaulannya bahwa ia pantas diterima di lingkungan mereka sebagai pemain basket.

2.2 Objek

Lapangan Basket

Berikut ini ada beberapa objek-objek pembentuk Lapangan Bola Basket, akan dijelaskan di bawah ini.

	Objek Pembentuk	Keterangan
	Sudut Lapangan	Menggunakan fungsi Lfloat courtVertices[][3] yang berfungsi untuk membentuk lapangan dan glColor3f untuk memberi warna

Tiang Keranjang Basket

Berikut ini adalah objek-objek yang membangun Selokan:

	Objek Pembentuk	Keterangan
	tiang keranjang pertama	Menggunakan fungsi GLfloat firstPoleVertices[][3] = untuk membentuk tiang. Dimana dalam hal ini pembentukannnya dibagi menjadi 3 (bagian base (paling bawah), bagian middle

		(tiang tengah), dan top (tiang atas))
	tiang keranjang kedua	Menggunakan fungsi GLfloat secondPoleVertices[][3] = untuk membentuk tiang. Dimana dalam hal ini pembentukannnya dibagi menjadi 3 (bagian base (paling bawah), bagian middle (tiang tengah), dan top (tiang atas))

Papan Keranjang Basket

Berikut ini adalah objek-objek yang membangun Orang Buang Sampah Organik :

	Objek Pembentuk	Keterangan
	papan keranjang pertama	Menggunakan fungsi GLfloat firstBoardVertices[][3] = untuk membentuk papan keranjang dasar. Dimana dalam hal ini pembentukannnya dibagi menjadi 2 (bagian base (paling tengah berwarna hitam), dan top (papan atas))
	papan keranjang kedua	Menggunakan fungsi GLfloat secondBoardVertices[][3] = untuk membentuk papan keranjang kedua. Dimana dalam hal ini pembentukannnya dibagi menjadi 2 (bagian base (paling tengah), dan top (papan atas)).

Pembentukan Karakter Orang

Berikut ini adalah objek-objek yang membangun karakter Orang :

	Objek Pembentuk	Keterangan
	Kepala	Menggunakan glutSolidSphere
	Badan	glutSolidCube di-glScale-kan sehingga membentuk balok, lalu ditumpuk dengan glutSolidSphere.
	Topi	glutWireTorus dilakukan glRotate dan di glTranslate sehingga masuk ke kepala
	Mata dan Bola Mata	Mata menggunakan glutWireTorus dilakukan glTranslate dan glScale sehingga membentuk mata, bola mata menggunakan glutSolidSphere
	Hidung	Menggunakan glutSolidSphere
	Mulut	Menggunakan glutWireTorus lalu di-glScale dan glRotate
	Tangan Kanan	Untuk tangan menggunakan glutSolidCube dilakukan glRotate supaya melentang. Untuk telapak tangannya menggunakan glutWireTorus
	Tangan Kiri	Sama seperti tangan kanan, namun perbedaannya di glRotate agar posisi tangan ke atas (sedang melempar bola)

	Kaki	Kedua kaki menggunakan glutSolidCube lalu di-glScale supaya membentuk kaki.
	Telapak Kaki	Kedua telapak kaki menggunakan glutSolidCube dilakukan glScale.

2.3. Animasi

Karakter Melakukan Lambungan Bola ke Keranjang

Berikut ini adalah proses yang membangun animasi karakter melambungkan bola ke keranjang :

	Objek Pembentuk	Keterangan
	Kepala	Menggunakan glutSolidSphere
	Badan	glutSolidCube di-glScale-kan sehingga membentuk balok, lalu ditumpuk dengan glutSolidSphere.
	Rambut	glutSolidSphere dilakukan glRotate dan di glTranslate sehingga pas dengan posisi kepala
	Mata	Mata kiri dan kanan menggunakan glutSolidSphere
	Leher	Menggunakan glutSolidCube
	Mulut	Menggunakan glutSolidSphere lalu di-glScale dan glRotate
	Tangan Kiri	Untuk tangan glutSolidCone, glutSolidCube dan untuk telapak tangan menggunakan glutSolidSphere

	Tangan Kanan	Sama seperti tangan kiri, namun dilakukan glRotate supaya membentuk tangan yang sedang melambungkan bole
	Kaki	Kedua kaki menggunakan glutSolidCube lalu di-glScale supaya membentuk kaki

2.4. Pembagian Tugas

Pembagian pekerjaan personal

1. I Made Suastika
   - Objek Lapangan
   - Objek Tiang net
2. Angie Safira Indah
   - Objek kotak net
   - Pewarnaan Objek
3. Muhammad Firyanul Rizky
   - Animasi Lambung Bola
   - Laporan
4. Gusti Ayu Purnami Indryaswari
   - Ide Skenario
   - Objek karakter

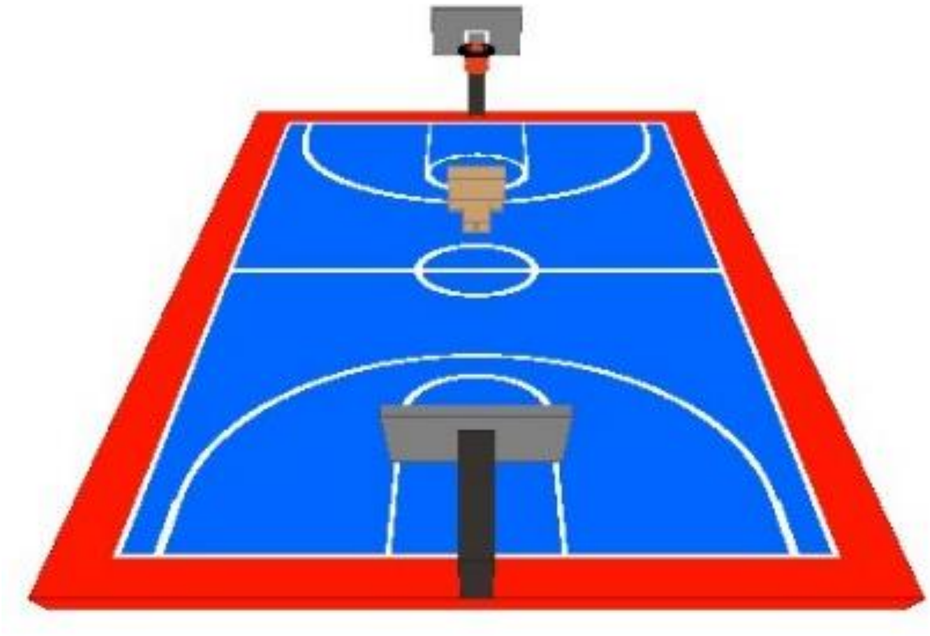
BAB 4

KESIMPULAN DAN SARAN

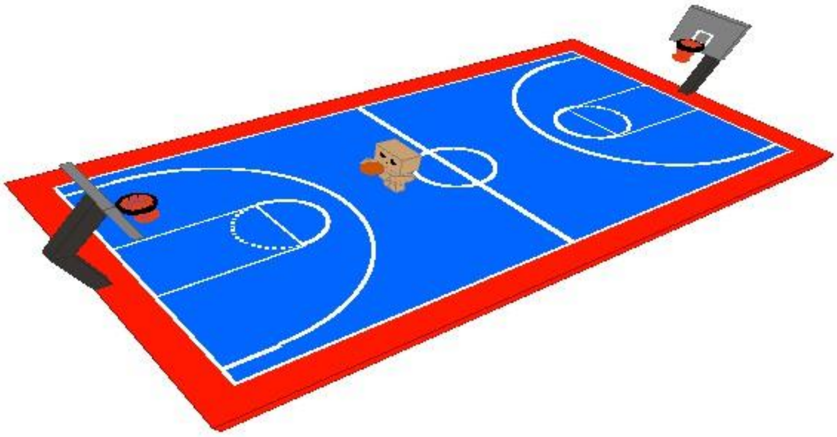
Tampilan

Berikut ini adalah implementasi Aplikasi yang dibuat.

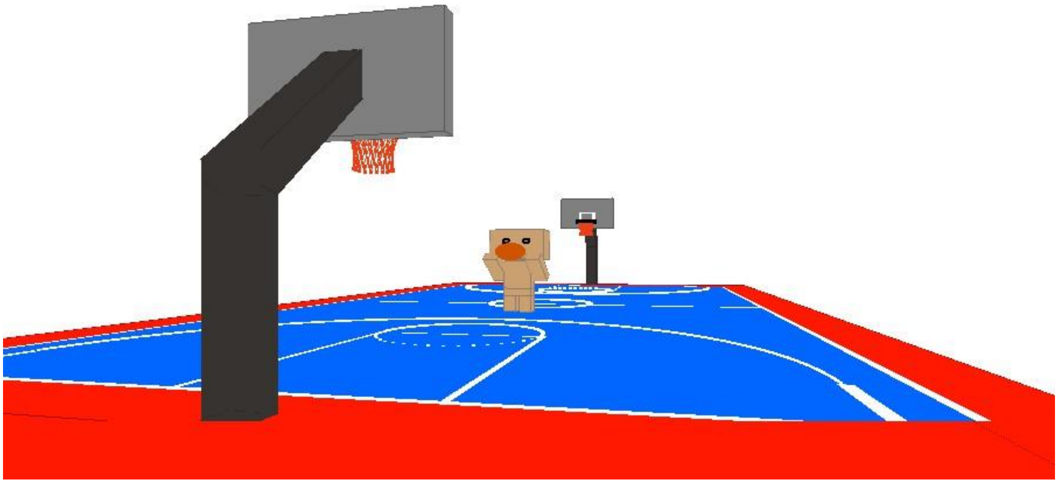
1. Tampak Atas



2. Tampak Samping



3. Tampak Depan



Source Code

```
#include<stdio.h>
#include<GL/glut.h>
#include<math.h>
#include<time.h>

int firsttime = 0;
float x = 0 , y = 0, z = 0.0;
GLfloat oldy = 0, oldz = 0, tempz, dy = 0, dz = 0;
int triggered = 0;
GLfloat courtVertices[][3] = {
      //basket ball court vertices
      {-2.5, -1.0, -4.7}, {2.5, -1.0, -4.7},
      {2.5, -1.0, 4.7}, {-2.5, -1.0, 4.7}
};

GLfloat firstPoleVertices[][3] = {
      //basket pole vertuces
      //base
      {-0.1, -1.0, -5.2}, {0.1, -1.0, -5.2},
      {-0.1, -1.0, -5.0}, {0.1, -1.0, -5.0},
      //middle
      {-0.1, 0.5, -5.2}, {0.1, 0.5, -5.2},
      {-0.1, 0.4, -5.0}, {0.1, 0.4, -5.0},
      //top
      {-0.1, 1.3, -4.4 }, { 0.1, 1.3 , -4.4},
      {-0.1, 1.7, -4.4 }, { 0.1, 1.7, -4.4 }
};

GLfloat secondPoleVertices[][3] = {
      //basket pole vertuces
      //base
      {-0.1, -1.0, 5.2}, {0.1, -1.0, 5.2},
      {-0.1, -1.0, 5.0}, {0.1, -1.0, 5.0},
      //middle
      {-0.1, 0.5, 5.2}, {0.1, 0.5, 5.2},
      {-0.1, 0.4, 5.0}, {0.1, 0.4, 5.0},
      //top
      {-0.1, 1.3, 4.4 }, { 0.1, 1.3 , 4.4},
      {-0.1, 1.7, 4.4 }, { 0.1, 1.7, 4.4}
};

GLfloat firstBoardVertices[][3] = {
      //basket board vertuces
      //base
      {-0.5, 1.0, -4.3}, { 0.5, 1.0, -4.3},
      {-0.5, 1.0, -4.4}, { 0.5, 1.0, -4.4},

      //top
      {-0.5, 2.0, -4.3}, { 0.5, 2.0, -4.3},
      {-0.5, 2.0, -4.4}, { 0.5, 2.0, -4.4},
};

GLfloat secondBoardVertices[][3] = {
      //basket board vertuces
      //base
      {-0.5, 1.0, 4.3}, { 0.5, 1.0, 4.3},
      {-0.5, 1.0, 4.4}, { 0.5, 1.0, 4.4},

      //top
      {-0.5, 2.0, 4.3}, { 0.5, 2.0, 4.3},
      {-0.5, 2.0, 4.4}, { 0.5, 2.0, 4.4},
};

GLfloat baseVertices[][3] = {
      //top
      {-3.0, -1.0001, -5.2}, {3.0, -1.0001, -5.2},
      {-3.0, -1.0001, 5.2}, {3.0, -1.0001, 5.2},
      //bottom
      {-3.0, -1.5, -5.2}, {3.0, -1.5, -5.2},
      {-3.0, -1.5, 5.2}, {3.0, -1.5, 5.2},

};
void poles(int a, int b, int c, int d)
{
      glBegin(GL_POLYGON);
      //glColor3f(0.0, 0.0, 0.0);
      glColor3f(55.0 / 255.0, 51.0/ 255.0, 49.0/ 255.0);
      glVertex3fv(firstPoleVertices[a]);
      glVertex3fv(firstPoleVertices[b]);
      glVertex3fv(firstPoleVertices[c]);
      glVertex3fv(firstPoleVertices[d]);
      glEnd();
      //border for the poles
      glBegin(GL_LINE_LOOP);
      glColor3f(43.0 / 255.0, 39.0/ 255.0, 37.0/ 255.0);
      glVertex3fv(firstPoleVertices[a]);
      glVertex3fv(firstPoleVertices[b]);
      glVertex3fv(firstPoleVertices[c]);
      glVertex3fv(firstPoleVertices[d]);
      glEnd();
      //second pole
      glBegin(GL_POLYGON);
      glColor3f(55.0 / 255.0, 51.0/ 255.0, 49.0/ 255.0);
      glVertex3fv(secondPoleVertices[a]);
      glVertex3fv(secondPoleVertices[b]);
      glVertex3fv(secondPoleVertices[c]);
      glVertex3fv(secondPoleVertices[d]);
      glEnd();

      glBegin(GL_LINE_LOOP);
      glColor3f(43.0 / 255.0, 39.0/ 255.0, 37.0/ 255.0);
      glVertex3fv(secondPoleVertices[a]);
      glVertex3fv(secondPoleVertices[b]);
      glVertex3fv(secondPoleVertices[c]);
      glVertex3fv(secondPoleVertices[d]);
      glEnd();
}
void lines(float a, float b, float c, float d)
{
      //a = -2.5, b = -0.05 c = 0.05 d = 2.5
      glBegin(GL_POLYGON);
      glColor3f(1.0, 1.0, 1.0);
      glVertex3f(a , -0.9999 , b);
      glVertex3f(d , -0.9999 , b);
      glVertex3f(d , -0.9999 , c);
      glVertex3f(a , -0.9999 , c);
      glEnd();

      glBegin(GL_POLYGON);
      glColor3f(1.0, 1.0, 1.0);
      glVertex3f(a , -0.9999 , b);
      glVertex3f(d , -0.9999 , b);
      glVertex3f(d , -0.9999 , c);
      glVertex3f(a , -0.9999 , c);
      glEnd();
}

void onBoardLines(float a, float b, float c, float d)
{
      glBegin(GL_POLYGON);
      glColor3f(1.0, 1.0, 1.0);
      glVertex3f(a, b, -4.29);
      glVertex3f(d, b, -4.29);
      glVertex3f(d, c, -4.29);
      glVertex3f(a, c, -4.29);
      glEnd();

      glBegin(GL_POLYGON);
      glColor3f(1.0, 1.0, 1.0);
      glVertex3f(a, b, 4.29);
      glVertex3f(d, b, 4.29);
      glVertex3f(d, c, 4.29);
      glVertex3f(a, c, 4.29);
      glEnd();
}

void board(int a, int b, int c, int d)
{
      glBegin(GL_POLYGON);
      glColor3f(0.5, 0.5, 0.5);
      glVertex3fv(firstBoardVertices[a]);
      glVertex3fv(firstBoardVertices[b]);
      glVertex3fv(firstBoardVertices[c]);
      glVertex3fv(firstBoardVertices[d]);
      glEnd();

      glBegin(GL_LINE_LOOP);
      glColor3f(86.0/255.0, 86.0/255.0, 86.0/255.0);
      glVertex3fv(firstBoardVertices[a]);
      glVertex3fv(firstBoardVertices[b]);
      glVertex3fv(firstBoardVertices[c]);
      glVertex3fv(firstBoardVertices[d]);
      glEnd();

      glBegin(GL_POLYGON);
      glColor3f(0.5, 0.5, 0.5);
      glVertex3fv(secondBoardVertices[a]);
      glVertex3fv(secondBoardVertices[b]);
      glVertex3fv(secondBoardVertices[c]);
      glVertex3fv(secondBoardVertices[d]);
      glEnd();

      glBegin(GL_LINE_LOOP);
      glColor3f(86.0/255.0, 86.0/255.0, 86.0/255.0);
      glVertex3fv(secondBoardVertices[a]);
      glVertex3fv(secondBoardVertices[b]);
      glVertex3fv(secondBoardVertices[c]);
      glVertex3fv(secondBoardVertices[d]);
      glEnd();
}

void base(int a, int b, int c, int d)
{
      glBegin(GL_POLYGON);
      glColor3f(1.0, 0.1, 0.0);
      glVertex3fv(baseVertices[a]);
      glVertex3fv(baseVertices[b]);
      glVertex3fv(baseVertices[c]);
      glVertex3fv(baseVertices[d]);
      glEnd();

      glBegin(GL_LINE_LOOP);
      glColor3f(165.0/255.0, 0.0/255.0, 3.0/255.0);
      glVertex3fv(baseVertices[a]);
      glVertex3fv(baseVertices[b]);
      glVertex3fv(baseVertices[c]);
      glVertex3fv(baseVertices[d]);
      glEnd();
}
void polygon(int a, int b, int c, int d)
{
      base(0, 1, 3, 2);
      base(4, 5, 7, 6);
      base(2, 3, 7, 6);
      base(0, 1, 5, 4);
      base(0, 2, 6, 4);
      base(1, 3, 7, 5);
      //court color
      glBegin(GL_POLYGON);
      glColor3f(0.0, 0.4, 1.0);
```

```
	glVertex3fv(courtVertices[a]);
	glColor3f(0.0, 0.4, 1.0);
	glVertex3fv(courtVertices[b]);
	glColor3f(0.0, 0.4, 1.0);
	glVertex3fv(courtVertices[c]);
	glColor3f(0.0, 0.4, 1.0);
	glVertex3fv(courtVertices[d]);
	glEnd();
	// pole from bast to the top
	poles(0, 1, 3, 2);
	poles(4, 5, 7, 6);
	poles(2, 3, 7, 6);
	poles(4, 5, 1, 0);
	poles(3, 1, 5, 7);
	poles(0, 2, 6, 4);

	//poles from center to board
	poles(6, 7, 9, 8);
	poles(4, 5, 11, 10);
	poles(6, 4, 10 , 8);
	poles(7, 5, 11, 9);

	//drawing board
	board(0, 1, 3, 2);
	board(4, 5, 7, 6);
	board(0, 2, 6, 4);
	board(1, 3, 7, 5);
	board(0, 1, 5, 4);
	board(3, 7, 6, 2);// remove the comments to draw the board
faces

	//center line
	lines(-2.5, -0.05, 0.05, 2.5);

	//side lines
	lines(-2.50, 4.7, -4.7, -2.55);
	lines(2.50, 4.7, -4.7, 2.55);

	//base lines
	lines(-2.55, 4.70, 4.75, 2.55);
	lines(-2.55, -4.70, -4.75, 2.55);


	//three pointer lines
	lines(-2.2, 4.7, 4.05, -2.27);
	lines( 2.2, 4.7, 4.05, 2.27);

	lines(-2.2, -4.7, -4.05, -2.27);
	lines( 2.2, -4.7, -4.05, 2.27);

	//two pointer lines
	lines(-0.6, 4.7, 2.8, -0.64);
	lines(0.6, 4.7, 2.8, 0.64);

	lines(-0.6, -4.7, -2.8, -0.64);
	lines(0.6, -4.7, -2.8, 0.64);

	//lines joining the above two lines
	lines(-0.6, 2.8, 2.84, 0.6);
```

```
      lines(-0.6, -2.8, -2.84, 0.6);

      //vertical lines on the board
      onBoardLines(-0.15, 1.2, 1.24, 0.15); //bottom
      onBoardLines(-0.15, 1.5, 1.54, 0.15); //down

      onBoardLines(-0.15, 1.2, 1.5, -0.10);
      onBoardLines(0.15, 1.2, 1.5, 0.10);

}

void circle(float r)
{
      int i;
      glColor3f(1.0, 1.0, 1.0);
      glPointSize(3.0);
      glBegin(GL_POINTS);
      for(i = 0; i < 1000; i++)
      {
      //x and y defines the radius
      glVertex3f( (r * cos(2*3.14159 * i/1000.0)), -0.9999, (r * 
sin(2*3.14159 * i/1000.0)));
      }
      glEnd();
}

void Dcircle(float r)
{
      int i;
      glColor3f(1.0, 1.0, 1.0);
      glBegin(GL_POINTS);
      for(i = 0; i < 1000; i++)
      {
      //x and y defines the radius
      glVertex3f( (r * cos(1*3.14159 * i/1000.0)), -0.9999, 2.8
- (r * sin(1*3.14159 * i/1000.0)));
      glVertex3f( (r * cos(1*3.14159 * i/1000.0)), -0.9999, -2.8
+ (r * sin(1*3.14159 * i/1000.0)));

      }

      for(i = 0; i < 20; i++)
      {
      //x and y defines the radius
      glVertex3f( (r * cos(1*3.14159 * i/20.0)), -0.9999, 2.8 +
(r * sin(1*3.14159 * i/20.0)));
      glVertex3f( (r * cos(1*3.14159 * i/20.0)), -0.9999, -2.8 -
(r * sin(1*3.14159 * i/20.0)));
      }
      glEnd();
}

void ring(float r)
{
      int i;
      glColor3f(0.0, 0.0, 0.0);
      glBegin(GL_POINTS);
      for(i = 0; i< 1000; i++)
      {
```

```
      //x and y defines the radius
      glVertex3f((r * cos(2*3.14159 * i/1000.0)), 1.2, 4.3 -
0.19 + (r * sin(2*3.14159 * i/1000.0)));
      glVertex3f((r * cos(2*3.14159 * i/1000.0)), 1.2, -4.3 +
0.19 + (r * sin(2*3.14159 * i/1000.0)));
      }
      glEnd();
}


void semicircle(float r)
{
      int i;
      glColor3f(1.0, 1.0, 1.0);
      glPointSize(3.0);
      glBegin(GL_POINTS);
      for(i = 0; i < 1000; i++)
      {
      glVertex3f((r * cos(1*3.14159 * i/1000.0)), -0.9999, 4.05
- (r * sin(1*3.14159 * i/1000.0)));
      glVertex3f((r * cos(1*3.14159 * i/1000.0)), -0.9999, -4.05
+ (r * sin(1*3.14159 * i/1000.0)));
      }
      glEnd();
}

void ball()
{
      if(firsttime)
      {
      glTranslatef(0.0, 1.2, -1.5); //calculated using the last
vertex of parabola
      }
      else
      {
      glTranslatef(0.0, 0.8, -2.8); //calculated using the last
vertex of parabola
      }
      glColor3f(0.81176, 0.3254, 0.0);
      glutSolidSphere(0.15, 1000, 20);
}

void net(int poleChooser)
{
      float r = 0.15;
      int i;
      float poleDecider = 0;
      GLfloat topVertices[10][200];
      GLfloat middleVertices[10][200];
      GLfloat bottomVertices[10][200];

      //choosing the pole
      if(poleChooser == 1)
      {
      poleDecider = 4.3 - 0.19;
      }
      else
      {
      poleDecider = -4.3 + 0.19;
      }

      //top vertices
      glColor3f(235.0/255.0, 63.0/255.0, 23.0/255.0);
      for(i = 0; i < 20; i++)
      {
      topVertices[0][i] = ((r) * cos(2 * 3.14159 * i/20.0)); //x
values
      topVertices[1][i] = (poleDecider + (r) * sin(2 * 3.14159 *
i/20.0)); //y values
      glBegin(GL_POINTS);
      glVertex3f(topVertices[0][i], 1.2, topVertices[1][i]);
      glEnd();
      }

      //middle vertices
      for(i = 0; i < 20; i++)
      {
      middleVertices[0][i] = ((r - 0.05)* cos(2 * 3.14159 *
i/20.0)); //x values
      middleVertices[1][i] = ( poleDecider + (r - 0.05)* sin(2 *
3.14159 * i/20.0)); //y values
      glBegin(GL_POINTS);
      glVertex3f(middleVertices[0][i], 1.0,
middleVertices[1][i]);
      glEnd();
      }

      //bottom vertices
      for(i = 0; i < 20; i++)
      {
      bottomVertices[0][i] = ((r - 0.05)* cos(2 * 3.14159 *
i/20.0)); //x values
      bottomVertices[1][i] = ( poleDecider + (r - 0.05)* sin(2 *
3.14159 * i/20.0)); //y values
      glBegin(GL_POINTS);
      glVertex3f(bottomVertices[0][i], 0.8,
bottomVertices[1][i]);
      glEnd();
      }


      //drawing lines using vertices to get the rhombus pattern
      for(i = 0; i < 20; i++)
      {
      //from top vertices to the middle vertices
      glBegin(GL_LINES);
      if(i == 19)
      {
            glVertex3f(topVertices[0][i], 1.2,
topVertices[1][i]);
            glVertex3f(middleVertices[0][0], 1.0,
middleVertices[1][0]);
      }
      else
      {
            glVertex3f(topVertices[0][i], 1.2,
topVertices[1][i]);
            glVertex3f(middleVertices[0][i + 1], 1.0,
middleVertices[1][i + 1]);
      }
      glEnd();

      glBegin(GL_LINES);
      if(i == 0)
      {
            glVertex3f(topVertices[0][i], 1.2,
topVertices[1][i]);
            glVertex3f(middleVertices[0][19], 1.0,
middleVertices[1][19]);
      }
      else
      {
            glVertex3f(topVertices[0][i], 1.2,
topVertices[1][i]);
            glVertex3f(middleVertices[0][i - 1], 1.0,
middleVertices[1][i - 1]);
      }
      glEnd();


      //from middle vertices to bottom vertices
      glBegin(GL_LINES);
      if(i == 19)
      {
            glVertex3f(middleVertices[0][i], 1.0,
middleVertices[1][i]);
            glVertex3f(bottomVertices[0][0], 0.8,
bottomVertices[1][0]);
      }
      else
      {
            glVertex3f(middleVertices[0][i], 1.0,
middleVertices[1][i]);
            glVertex3f(bottomVertices[0][i + 1], 0.8,
bottomVertices[1][i + 1]);
      }
      glEnd();

      glBegin(GL_LINES);
      if(i == 0)
      {
            glVertex3f(middleVertices[0][i], 1.0,
middleVertices[1][i]);
            glVertex3f(bottomVertices[0][19], 0.8,
bottomVertices[1][19]);
      }
      else
      {
            glVertex3f(middleVertices[0][i], 1.0,
middleVertices[1][i]);
            glVertex3f(bottomVertices[0][i - 1], 0.8,
bottomVertices[1][i - 1]);
      }
      glEnd();

      }
}
```

```
//shoes
float rightLeg[][3] = {
      {0.03, -0.9999, -0.1}, {0.03, -0.9999, 0.1}, {0.2, -
0.9999, 0.1}, {0.2, -0.9999, -0.1},
      {0.03, -0.6, -0.1}, {0.03, -0.6, 0.1}, {0.2, -0.6, 0.1},
{0.2, -0.6, -0.1}
};
float leftLeg[][3] = {
      {-0.2, -0.9999, -0.1}, {-0.2, -0.9999, 0.1}, {-0.03, -
0.9999, 0.1}, {-0.03, -0.9999, -0.1},
      {-0.2, -0.6, -0.1}, {-0.2, -0.6, 0.1}, {-0.03, -0.6, 0.1},
{-0.03, -0.6, -0.1}
};

float trunk[][3] = {
      {-0.2, -0.6, -0.1}, {-0.2, -0.6, 0.1}, {0.2, -0.6, 0.1},
{0.2, -0.6, -0.1},
      {-0.2, -0.4, -0.1}, {-0.2, -0.4, 0.1}, {0.2, -0.4, 0.1},
{0.2, -0.4, -0.1}
};
float body[][3] = {
      {-0.2, -0.4, -0.1}, {-0.2, -0.4, 0.1}, {0.2, -0.4, 0.1},
{0.2, -0.4, -0.1},
      {-0.2, 0.3, -0.1}, {-0.2, 0.3, 0.1}, {0.2, 0.3, 0.1},
{0.2, 0.3, -0.1}
};

float head[][3] = {
      {-0.4, 0.35, -0.2}, {-0.4, 0.35, 0.2}, {0.4, 0.35, 0.2},
{0.4, 0.35, -0.2},
      {-0.4, 1.1, -0.2}, {-0.4, 1.1, 0.2}, {0.4, 1.1, 0.2},
{0.4, 1.1, -0.2}
};

float leftHand[][3] = {
      {-0.2, 0.2, -0.1}, {-0.2, 0.2, 0.1}, {-0.4, 0.2, 0.1}, {-
0.4, 0.2, -0.1},
      {-0.2, -0.6, -0.1}, {-0.2, -0.6, 0.1}, {-0.4, -0.6, 0.1},
{-0.4, -0.6, -0.1}
};

float rightHand[][3] = {
      {0.2, 0.2, -0.1}, {0.2, 0.2, 0.1}, {0.4, 0.2, 0.1}, {0.4,
0.2, -0.1},
      {0.2, -0.6, -0.1}, {0.2, -0.6, 0.1}, {0.4, -0.6, 0.1},
{0.4, -0.6, -0.1}
};
void characterDesign(int a, int b, int c, int d)
{
      float R = 143.0/255.0, G = 125.0 / 255.0, B = 100.0 /
225.0;

      glColor3f(202.0 / 255.0, 160.0/255.0, 100.0/225.0);
      glBegin(GL_POLYGON);
      glVertex3fv(trunk[a]);
      glVertex3fv(trunk[b]);
      glVertex3fv(trunk[c]);
      glVertex3fv(trunk[d]);
      glEnd();


      glColor3f(202.0 / 255.0, 160.0/255.0, 100.0/225.0);
      glBegin(GL_POLYGON);
      glVertex3fv(body[a]);
      glVertex3fv(body[b]);
      glVertex3fv(body[c]);
      glVertex3fv(body[d]);
      glEnd();
      //head. rotated by 3

      glPushMatrix();
      glRotatef(3.0, 1.0, 0.0, 0.0);
      glColor3f(202.0 / 255.0, 160.0/255.0, 100.0/225.0);
      glBegin(GL_POLYGON);
      glVertex3fv(head[a]);
      glVertex3fv(head[b]);
      glVertex3fv(head[c]);
      glVertex3fv(head[d]);
      glEnd();

      glColor3f(R, G, B);
      glBegin(GL_LINE_LOOP);
      glVertex3fv(head[a]);
      glVertex3fv(head[b]);
      glVertex3fv(head[c]);
      glVertex3fv(head[d]);
      glEnd();

      //eyes
      int i;
      float r = 0.05;
      glColor3f(0.0, 0.0, 0.0);
      glPointSize(2.0);
      glBegin(GL_POINTS);
      for(i = 0; i< 1000; i++)
      {
      //x and y defines the radius
      glVertex3f( 0.15 - (r * cos(2*3.14159 * i/1000.0)), 0.8 +
(r * sin(2*3.14159 * i/1000.0)), -0.2);
      glVertex3f( -0.15 - (r * cos(2*3.14159 * i/1000.0)), 0.8 +
(r * sin(2*3.14159 * i/1000.0)), -0.2);
      }

      glEnd();

      //mouth
      glColor3f(0.0, 0.0, 0.0);
      glBegin(GL_POLYGON);
      glVertex3f(0.0 , 0.55 , -0.21);
      glVertex3f(0.05 , 0.6, -0.21);
      glVertex3f(-0.05 , 0.6 , -0.21);
      glEnd();
      glPopMatrix();

      //legs
      glColor3f(202.0 / 255.0, 160.0/255.0, 100.0/225.0);
      glBegin(GL_POLYGON);
```

```
	glVertex3fv(rightLeg[a]);
	glVertex3fv(rightLeg[b]);
	glVertex3fv(rightLeg[c]);
	glVertex3fv(rightLeg[d]);
	glEnd();

	glColor3f(202.0 / 255.0, 160.0/255.0, 100.0/225.0);
	glBegin(GL_POLYGON);
	glVertex3fv(leftLeg[a]);
	glVertex3fv(leftLeg[b]);
	glVertex3fv(leftLeg[c]);
	glVertex3fv(leftLeg[d]);
	glEnd();

	glPushMatrix();
	glRotatef(135.0, 1.0, 0.0, 0.0);
	glColor3f(202.0 / 255.0, 160.0/255.0, 100.0/225.0);
	glBegin(GL_POLYGON);
	glVertex3fv(leftHand[a]);
	glVertex3fv(leftHand[b]);
	glVertex3fv(leftHand[c]);
	glVertex3fv(leftHand[d]);
	glEnd();

	glColor3f(R, G, B);
	glBegin(GL_LINE_LOOP);
	glVertex3fv(leftHand[a]);
	glVertex3fv(leftHand[b]);
	glVertex3fv(leftHand[c]);
	glVertex3fv(leftHand[d]);
	glEnd();
	glPopMatrix();

	glPushMatrix();
	glRotatef(135.0, 1.0, 0.0, 0.0);
	glColor3f(202.0 / 255.0, 160.0/255.0, 100.0/225.0);
	glBegin(GL_POLYGON);
	glVertex3fv(rightHand[a]);
	glVertex3fv(rightHand[b]);
	glVertex3fv(rightHand[c]);
	glVertex3fv(rightHand[d]);
	glEnd();

	glColor3f(R, G, B);
	glBegin(GL_LINE_LOOP);
	glVertex3fv(rightHand[a]);
	glVertex3fv(rightHand[b]);
	glVertex3fv(rightHand[c]);
	glVertex3fv(rightHand[d]);
	glEnd();
	glPopMatrix();


	//borders

	glColor3f(R, G, B);
	glBegin(GL_LINE_LOOP);
	glVertex3fv(trunk[a]);
	glVertex3fv(trunk[b]);
```

```
      glVertex3fv(trunk[c]);
      glVertex3fv(trunk[d]);
      glEnd();

      glColor3f(R, G, B);
      glBegin(GL_LINE_LOOP);
      glVertex3fv(rightLeg[a]);
      glVertex3fv(rightLeg[b]);
      glVertex3fv(rightLeg[c]);
      glVertex3fv(rightLeg[d]);
      glEnd();

      glColor3f(R, G, B);
      glBegin(GL_LINE_LOOP);
      glVertex3fv(leftLeg[a]);
      glVertex3fv(leftLeg[b]);
      glVertex3fv(leftLeg[c]);
      glVertex3fv(leftLeg[d]);
      glEnd();

      glColor3f(R, G, B);
      glBegin(GL_LINE_LOOP);
      glVertex3fv(body[a]);
      glVertex3fv(body[b]);
      glVertex3fv(body[c]);
      glVertex3fv(body[d]);
      glEnd();


}
void character()
{
      glScalef(0.7, 0.7, 0.7);
      glTranslatef(0.0, -0.4, -1.5);
      characterDesign(0, 1, 2, 3);
      characterDesign(4, 5, 6, 7);
      characterDesign(0, 1, 5, 4);
      characterDesign(2, 3, 7, 6);
      characterDesign(1, 2, 6, 5);
      characterDesign(0, 3, 7, 4);
}

void draw(void)
{
      glPushMatrix();
      if(firsttime == 0)
      {
      glTranslatef(0, y, dz);
      }
      ball();
      glPopMatrix();

      polygon(0, 1, 2, 3);
      //center circle
      circle(0.60);

      Dcircle(0.6);

      net(1);
      net(2);

      ring(0.17);

      ////three pointer semi cirlce
      semicircle(2.22);

      //character
      character();
}

static GLfloat theta[] = {0.0, 0.0, 0.0};
static GLint axis = 2;
static GLdouble viewer[] =  {0.0, 7.0, 7.0};

void display(void)
{
      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
      glLoadIdentity();
      gluLookAt(viewer[0], viewer[1], viewer[2], 0.0, 0.0, 0.0,
0.0, 1.0, 0.0);
      glRotatef(theta[0], 1.0, 0.0, 0.0);
      glRotatef(theta[1], 0.0, 1.0, 0.0);
      glRotatef(theta[2], 0.0, 0.0, 1.0);
      draw();
      glFlush();
      glutSwapBuffers();
}
void update(int value)
{
      if(firsttime)
      {
      y = 0.5;
      dz = 1.2;
      firsttime = 0;
      }
      if(triggered && y >= -2 && z >= -1.0)
      {
      y = -(2 * dz * dz) + 3.6;
      dz -= 0.05;
      }
      else
      {
      y = -0.7;
      dz = 1.4;
      triggered = 0;
      }
      glutPostRedisplay();
      glutTimerFunc(25,update,0);
}

void mouse(int btn, int state, int x, int y)
{
      if(btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
      axis = 1;
      if(btn == GLUT_MIDDLE_BUTTON && state == GLUT_DOWN)
      axis = 0;
      if(btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
      axis = 2;
      theta[axis] += 2.0;
      if(theta[axis] > 360.0)
      theta[axis] -= 360.0;
      display();
}

void myReshape(int w, int h)
{
      glViewport(0, 0, w , h);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      if(w <= h)
      glFrustum(-2.0, 2.0, -2.0 * (GLfloat)h / (GLfloat)w , 2.0
* (GLfloat)h / (GLfloat)w, 2.0, 20.0);
      else
      glFrustum(-2.0, 2.0, -2.0 * (GLfloat)w / (GLfloat)h , 2.0
* (GLfloat)w / (GLfloat)h,  2.0, 20.0);
      glMatrixMode(GL_MODELVIEW);
}

void keys(unsigned char key, int x, int y)
{
      //test conditions to ensure that the camera always capture
the obejct and does move too far from the object
      if(key == 'x' && viewer[0] != -6) viewer[0] -= 1.0;
      if(key == 'X' && viewer[0] != 6) viewer[0] += 1.0;
      if(key == 'y' && viewer[1] != 0) viewer[1] -= 1.0;
      if(key == 'Y' && viewer[1] != 9) viewer[1] += 1.0;
      if(key == 'z'  && viewer[2] != 4) viewer[2] -= 1.0;
      if(key == 'Z'  && viewer[2] != 10) viewer[2] += 1.0;
      if(key == 's' || key == 'S')
      {
      triggered = 1;
      firsttime = 1;
      }
      display();
}

int main(int argc, char **argv)
{
      glutInit(&argc , argv);
      glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
      glutInitWindowSize(1000, 1000);
      glutCreateWindow("Colourable viewer");
      glutReshapeFunc(myReshape);
      glutDisplayFunc(display);
      glutMouseFunc(mouse);
      glutKeyboardFunc(keys);
      glEnable(GL_DEPTH_TEST);
      glutTimerFunc(1, update, 0);
      glClearColor(1.0, 1.0, 1.0, 0.0);
      glutMainLoop();
}
```