27.10.2019

Hazırlayan: Tayfun GÜRLEVİK

Öğrenci No: N19139647

ÖDEV NO: 3

1)

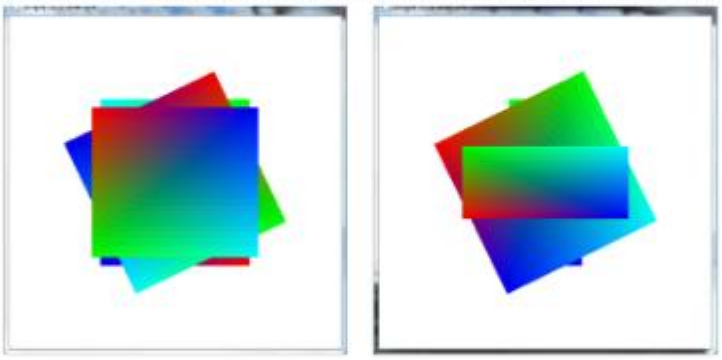
Aşağıdaki adımları takip ediniz:

Sahne içine sadece y-düzlemine göre dönebilen bir renkli dikdörtgen yerleştiriniz (bkz. Şekil 1).

Şekil 1

Sadece z-düzlemine göre dönebilen ikinci bir renkli dikdörtgen yerleştiriniz.

Sadece x-düzlemine göre dönebilen üçüncü bir renkli dikdörtgen yerleştiriniz (bkz. Şekil 2).



Şekil 2

Bu üç dikdörtgen birbirilerine yakın konumda olacak şekilde sahne içine yerleştirilmelidir.

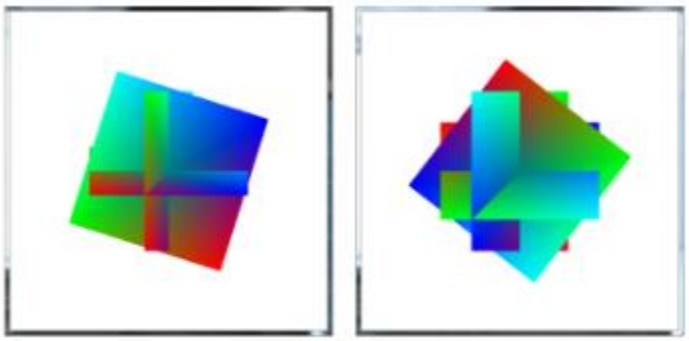
Her üç dikdörtgen de belirtilen eksen etrafında dönmelidir.

Dikdörtgenler genList ile oluşturulmalıdır.

Dönme esnasında düzlemler birbiri içine geçmelidir.

Programın ilk aşamasında derinlik testi yapmadan sonuç elde edilmeli (sonuç Şekil 2 'deki gibi olmalıdır).

Programın ikinci aşamasında derinlik testi uygulanmalıdır (sonuç Şekil 3 'deki gibi olmalıdır)



Şekil 3

YANIT:

İlk aşama(DEPTH_TEST kapalı):

a) Çözüm

İlk adım olarak dikdortgen isminde tek elemanlı bir glGenList olusturdum.

dikdortgen = glGenLists(1);

GL_POLYGON kullanarak dikdörtgeni olusturdum. Koordinatları sırasıyla (-40,-40,0), (40,-40,0),(40,40,0), (-40,40,0) vertexlerinden oluşmaktadır.

drawScene fonksiyonunda farklı eksenlerde dönecek olan 3 dikdörtgeni de `glCallList` fonksiyonu yardımıyla sahneye yerleştirdim. Her bir dörtgeni mümkün olduğunca orijin noktasına yakın yerleştirerek kendi eksenleri etrafında dönmelerini sağladım.

```
glPushMatrix();
glTranslatef(0, 0, 0);
glRotatef(angle, 0, 1, 0);
glCallList(dikdortgen); // Execute display list.
glPopMatrix();
glPushMatrix();
glTranslatef(0, 0, -0.0001f);
glRotatef(angle, 0, 0, 1);
glCallList(dikdortgen); // Execute display list.
glPopMatrix();
glPushMatrix();
glTranslatef(0, 0, 0.0001f);
glRotatef(angle, 1, 0, 0);
glCallList(dikdortgen); // Execute display list.
glPopMatrix();
```

Angle isminde global bir değişken tanımladım ve ilk değerini 0 olarak atadım. +/- tuşlarına basıldıkça angle değerini 1 birim arttırdım veya azalttım. Bu şekilde dönme animasyonunu gerçekleştirmiş oldum.

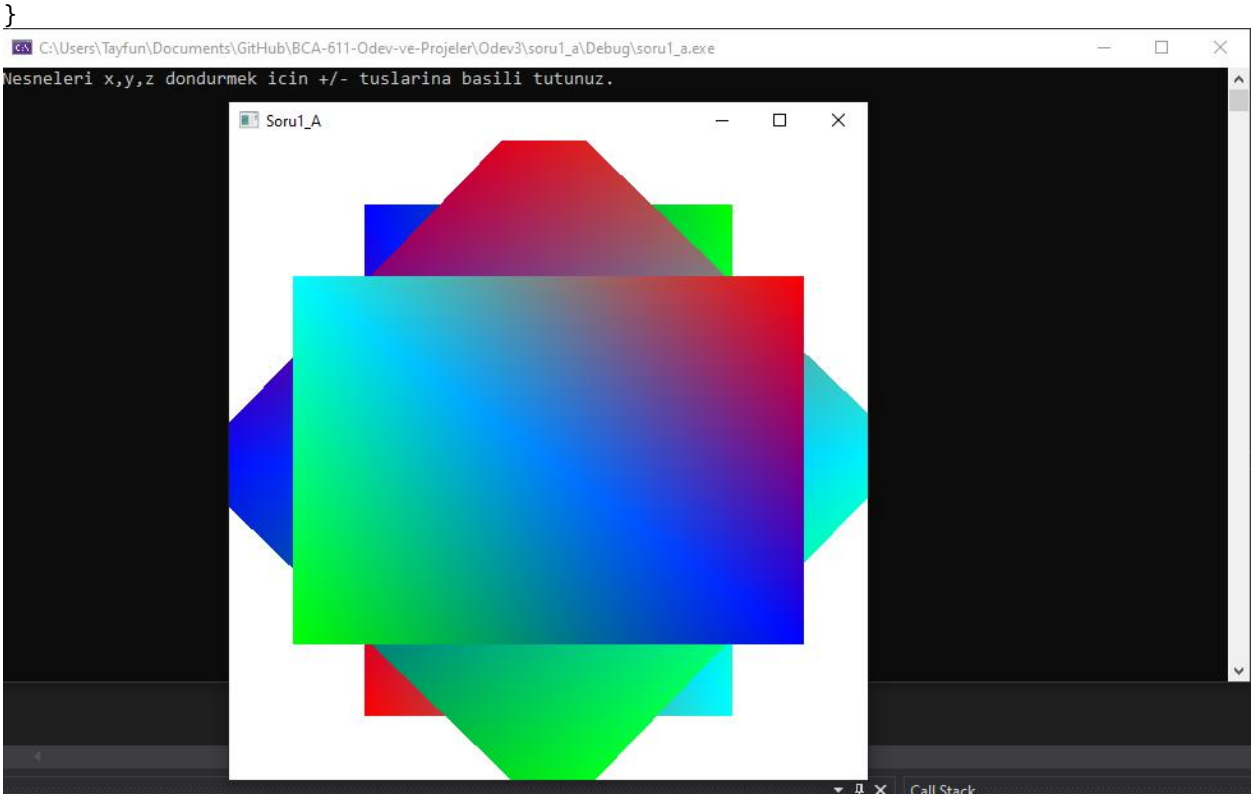
b) Kod

```
#include <cstdlib>
#include <cmath>
#include <iostream>

#include <GL/glew.h>
#include <GL/freeglut.h>

// Globals.
static float angle = 0.0f;
static unsigned int dikdortgen; // List index.

// Initialization routine.
void setup(void)
{


	dikdortgen = glGenLists(1); // Return a list index.

	// Begin create a display list.
	glNewList(dikdortgen, GL_COMPILE);

	// Draw a rectangle.

	glBegin(GL_POLYGON);
	glColor3f(0, 1.0, 1.0);
	glVertex3f(-40, -40, 0);
	glColor3f(1.0, 0, 0);
	glVertex3f(40, -40, 0);
	glColor3f(0, 0, 1.0);
	glVertex3f(40, 40, 0);
	glColor3f(0, 1.0, 0);
	glVertex3f(-40, 40, 0);
	glEnd();

	glEndList();



	// End create a display list.

	glClearColor(1.0, 1.0, 1.0, 0.0);
}

// Drawing routine.
void drawScene(void)
{

	glClear(GL_COLOR_BUFFER_BIT );// Clear window.

	glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
	glPushMatrix();
	glTranslatef(0, 0, 0);
	glRotatef(angle, 0, 1, 0);

	glCallList(dikdortgen); // Execute display list.
	glPopMatrix();
```

```
	glPushMatrix();
	glTranslatef(0, 0, -0.0001f);
	glRotatef(angle, 0, 0, 1);

	glCallList(dikdortgen); // Execute display list.
	glPopMatrix();
	glPushMatrix();
	glTranslatef(0, 0, 0.0001f);
	glRotatef(angle, 1, 0, 0);

	glCallList(dikdortgen); // Execute display list.
	glPopMatrix();


	glFlush();

}

// OpenGL window reshape routine.
void resize(int w, int h)
{
	glViewport(0, 0, w, h);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	glOrtho(-50, 50, -50, 50, -50, 100);

	/*glFrustum(-5.0, 5.0, -5.0, 5.0, 5, 100.0);
	gluLookAt(0, 0, 50, 0, 0, 0, 0, 1, 0);*/
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
}

// Keyboard input processing routine.
void keyInput(unsigned char key, int x, int y)
{
	switch (key)
	{
	case 27:
		exit(0);
		break;

	case '+':
		angle += 1;

		glPushMatrix();
		glTranslatef(0, 0, 0);
		glRotatef(angle, 0, 1, 0);

		glCallList(dikdortgen); // Execute display list.
		glPopMatrix();
		glPushMatrix();
		glTranslatef(0, 0, -0.0001f);
		glRotatef(angle, 0, 0, 1);

		glCallList(dikdortgen); // Execute display list.
		glPopMatrix();
		glPushMatrix();
		glTranslatef(0, 0, 0.0001f);
```

```
			glRotatef(angle, 0, 0, 0);

			glCallList(dikdortgen); // Execute display list.
			glPopMatrix();
			glutPostRedisplay();
			break;
		case '-':
			angle -= 1;

			glPushMatrix();
			glTranslatef(0, 0, 0);
			glRotatef(angle, 0, 1, 0);

			glCallList(dikdortgen); // Execute display list.
			glPopMatrix();
			glPushMatrix();
			glTranslatef(0, 0, -0.0001f);
			glRotatef(angle, 0, 0, 1);

			glCallList(dikdortgen); // Execute display list.
			glPopMatrix();
			glPushMatrix();
			glTranslatef(0, 0, 0.0001f);
			glRotatef(angle, 0, 0, 0);

			glCallList(dikdortgen); // Execute display list.
			glPopMatrix();
			glutPostRedisplay();
			break;
		default:
			break;
		}
}
void AciklamaYazdir()
{
		std::cout << "Nesneleri x,y,z dondurmek icin +/- tuslarina basili tutunuz." <<
std::endl;
}
// Main routine.
int main(int argc, char** argv)
{
		glutInit(&argc, argv);
		AciklamaYazdir();

		glutInitContextVersion(4, 3);
		glutInitContextProfile(GLUT_COMPATIBILITY_PROFILE);

		glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
		glutInitWindowSize(500, 500);
		glutInitWindowPosition(100, 100);
		glutCreateWindow("Soru1_A");
		glutDisplayFunc(drawScene);
		glutReshapeFunc(resize);
		glutKeyboardFunc(keyInput);

		glewExperimental = GL_TRUE;
		glewInit();
```

```
        setup();

        glutMainLoop();
}
```



c) Ekler

Soru1a.cpp,soru1_a.exe

İkinci Aşama aşama(DEPTH_TEST açık):

a)Çözüm:

Deph testi açmak için main() fonksiyonunda glewInıt(); satırından hemen sonra glEnable(GL_DEPTH_TEST); satırını ekledim.

glClear(GL_COLOR_BUFFER_BIT); satırını glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT); şeklinde güncelledim.

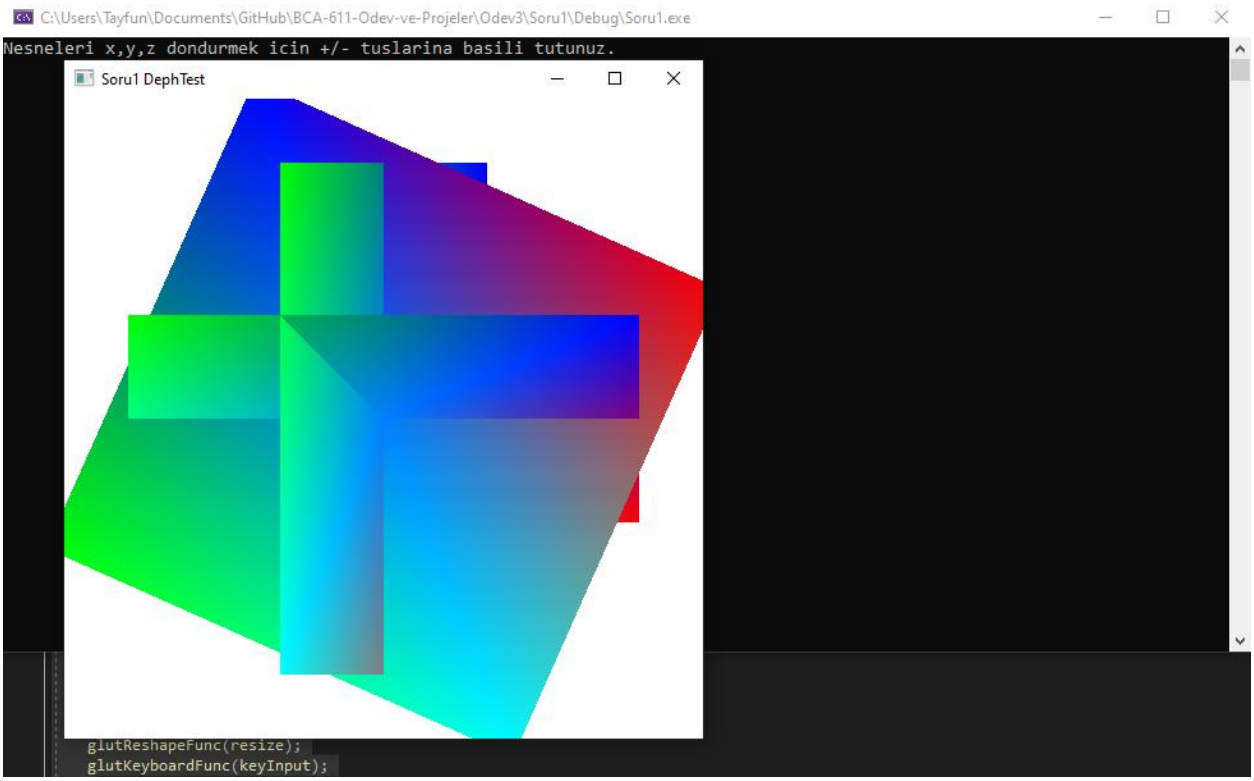
b)Kod:

```
#include <cstdlib>
#include <cmath>
#include <iostream>

#include <GL/glew.h>
#include <GL/freeglut.h>
```

```
// Globals.
static float angle = 0.0f;

static unsigned int dikdortgen; // List index.

// Initialization routine.
void setup(void)
{


	dikdortgen = glGenLists(1); // Return a list index.

	// Begin create a display list.
	glNewList(dikdortgen, GL_COMPILE);

	// Draw a rectangle.

	glBegin(GL_POLYGON);
	glColor3f(0, 1.0, 1.0);
	glVertex3f(-40, -40,0);
	glColor3f(1.0, 0, 0);
	glVertex3f(40, -40, 0);
	glColor3f(0, 0, 1.0);
	glVertex3f(40, 40, 0);
	glColor3f(0, 1.0, 0);
	glVertex3f(-40, 40, 0);
	glEnd();

	glEndList();



	// End create a display list.

	glClearColor(1.0, 1.0, 1.0, 0.0);
}

// Drawing routine.
void drawScene(void)
{

	glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);// Clear window.

	glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
	glPushMatrix();
	glTranslatef(0, 0,0);
	glRotatef(angle, 0, 1, 0);

	glCallList(dikdortgen); // Execute display list.
	glPopMatrix();
	glPushMatrix();
	glTranslatef(0, 0, -0.0001f);
	glRotatef(angle, 0, 0, 1);

	glCallList(dikdortgen); // Execute display list.
	glPopMatrix();
	glPushMatrix();
```

```
	glTranslatef(0, 0, 0.0001f);
	glRotatef(angle, 1, 0, 0);

	glCallList(dikdortgen); // Execute display list.
	glPopMatrix();


	glFlush();

}

// OpenGL window reshape routine.
void resize(int w, int h)
{
	glViewport(0, 0, w, h);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	glOrtho(-50, 50, -50, 50, -50, 100);

	/*glFrustum(-5.0, 5.0, -5.0, 5.0, 5, 100.0);
	gluLookAt(0, 0, 50, 0, 0, 0, 0, 1, 0);*/
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
}

// Keyboard input processing routine.
void keyInput(unsigned char key, int x, int y)
{
	switch (key)
	{
	case 27:
		exit(0);
		break;

	case '+':
		angle += 1;

		glPushMatrix();
		glTranslatef(0, 0, 0);
		glRotatef(angle, 0, 1, 0);

		glCallList(dikdortgen); // Execute display list.
		glPopMatrix();
		glPushMatrix();
		glTranslatef(0, 0, -0.0001f);
		glRotatef(angle,0, 0, 1);

		glCallList(dikdortgen); // Execute display list.
		glPopMatrix();
		glPushMatrix();
		glTranslatef(0, 0, 0.0001f);
		glRotatef(angle, 0,0, 0);

		glCallList(dikdortgen); // Execute display list.
		glPopMatrix();
		glutPostRedisplay();
		break;
	case '-':
```

```
			angle -= 1;

			glPushMatrix();
			glTranslatef(0, 0, 0);
			glRotatef(angle, 0, 1, 0);

			glCallList(dikdortgen); // Execute display list.
			glPopMatrix();
			glPushMatrix();
			glTranslatef(0, 0, -0.0001f);
			glRotatef(angle, 0, 0, 1);

			glCallList(dikdortgen); // Execute display list.
			glPopMatrix();
			glPushMatrix();
			glTranslatef(0, 0, 0.0001f);
			glRotatef(angle, 0, 0, 0);

			glCallList(dikdortgen); // Execute display list.
			glPopMatrix();
			glutPostRedisplay();
			break;
		default:
			break;
		}
}
void AciklamaYazdir()
{
		std::cout << "Nesneleri x,y,z dondurmek icin +/- tuslarina basili tutunuz." <<
std::endl;
}
// Main routine.
int main(int argc, char** argv)
{
		glutInit(&argc, argv);
		AciklamaYazdir();

		glutInitContextVersion(4, 3);
		glutInitContextProfile(GLUT_COMPATIBILITY_PROFILE);

		glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
		glutInitWindowSize(500, 500);
		glutInitWindowPosition(100, 100);
		glutCreateWindow("Soru1 DephTest");
		glutDisplayFunc(drawScene);
		glutReshapeFunc(resize);
		glutKeyboardFunc(keyInput);

		glewExperimental = GL_TRUE;
		glewInit();
		glEnable(GL_DEPTH_TEST);

		setup();

		glutMainLoop();
}
```



c)Ekler:

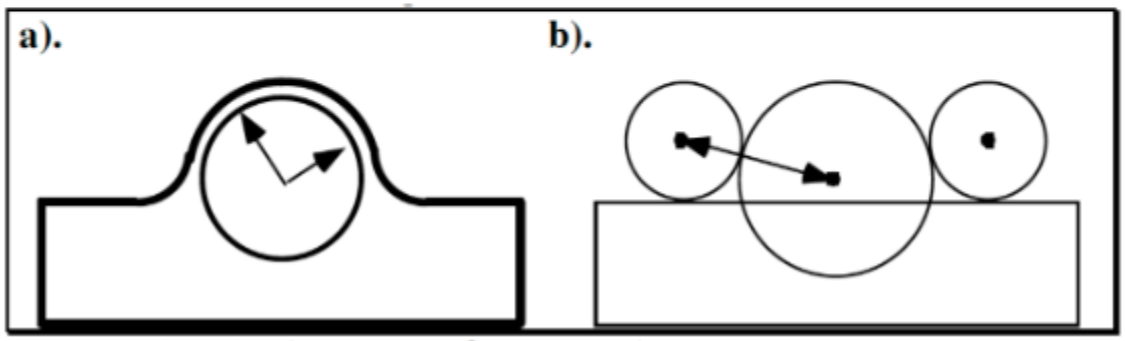
DephTest.cpp, Soru1.exe

2)

Aşağıdaki bağlantıda bir başlangıç açısı ile bir bitim açısı arasında kalan r yarıçaplı, (x,y) merkezli yay parçasını çizdiren OpenGl kodları yer almaktadır.

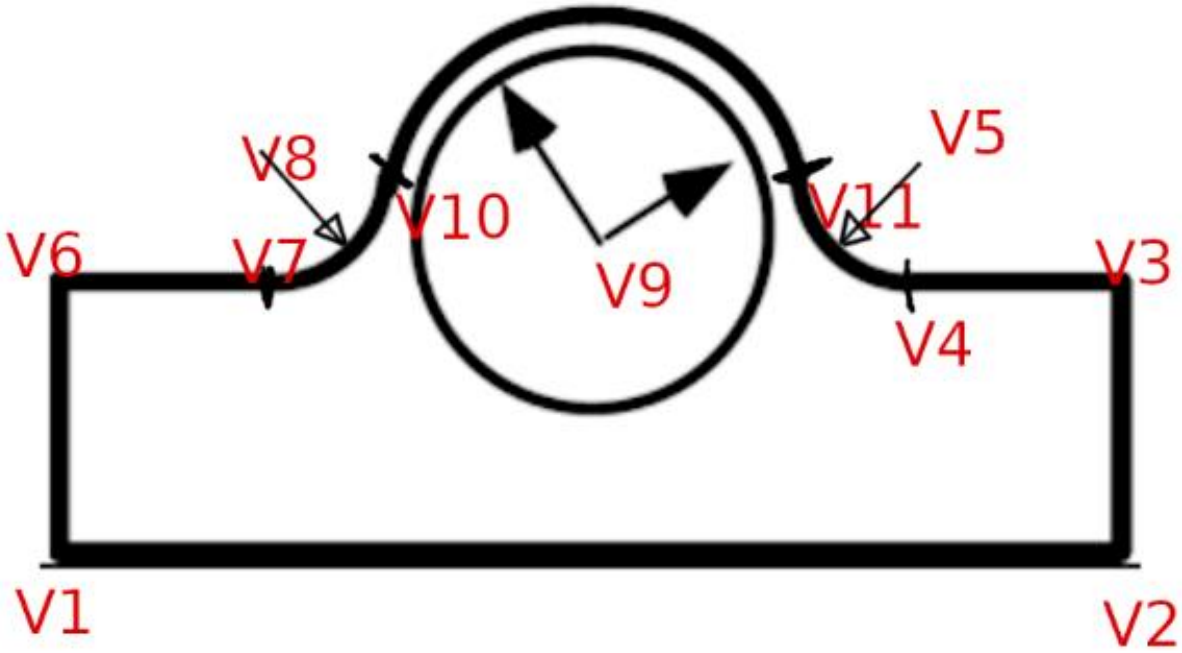
https://www.codeproject.com/Questions/1100935/How-to-draw-arc-in-opengl-using-x-y-position-radiu

Yay parçaları yardımıyla şekil (a) 'da görülen saati çizdiren programı yazınız.



YANIT:

a)Çözüm:



Sorunun çözümü için V9 noktasının x bileşeni V1V2 kenarının tam orta noktasında olacak şekilde tasarlanmıştır. Saatin uzun kenarı 80 birim uzunluğunda, kısa kenarı 40 birim uzunluğundadır. Küçük yay parçalarının yarıçapı 10 birim Kabul edilerek, merkez noktaları kenarlardan x ekseninde 20 birim, y ekseninde 10 birim olacak şekilde tasarlanmıştır. Buna göre vertexlerin koordinatları(x,y);

V1=10,10;

V2=90,10;

V3=90,50;

V4=70,50;

V5=70,60;

V6=10,50;

V7=30,50;

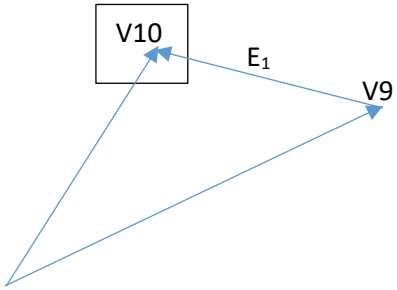
V8=30,60;

V9=50,55; olacak şekilde programa elle girilmiştir.

Küçük yayların yarıçapı=10 birim

V10,V11 ve saatin tepesindeki yayın yarıçapı programda çalışma zamanında hesaplanacaktır.

V10 ve V11 noktalarının bulunması;



E_1=V10-V9 olur,

Aynı şekilde saatin sağ tarafındaki yay için E_2=V11-V9 olur.

Cosinus teoreminden,

Θ_1=acos(dot(E_1,E_2)/|E_1|.|E_2|) ifadesiyle bulunabilir.

E_3=V7-V8 ve E_4=V9-V8 vektörleri yardımıyla küçük yayları oluşturan açı değeri de kosinüs teoremi vasıtasıyla hesaplanabilir.

Θ_2=acos(dot(E_3,E_4)/|E_3|.|E_4|)

V7,V8 ve V9 noktaları bilindiğinden ilk önce bu yayları gören açılar hesaplanarak, yayların çizimi programda yapılmıştır.

Yaylar çizilirken başlangıç ve bitiş noktaları koordinatları bilinmeyen V10 ve V11 vertekslerine atanmış, daha sonar bu değerler yardımıyla da Θ_1 açısı bulunarak saatin tepesindeki yay çizilmiştir.

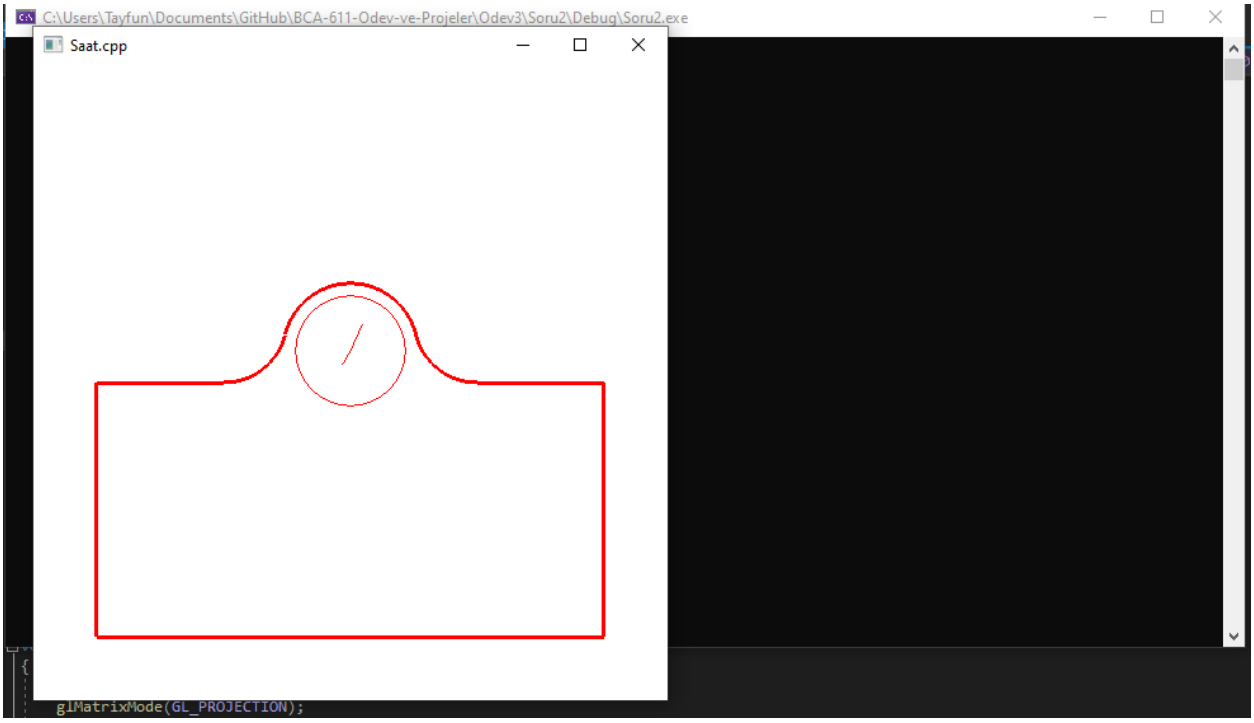
b)Kod:

```
#include <cmath>
#include <iostream>
#include <ctime>
#include <GL/glew.h>
#include <GL/freeglut.h>
#include <glm.hpp>
#define PI 3.14159265358979324


void ArcCizdir(float x,float y,float yaricap,float baslangicAcisi,float bitisAcisi,float
increment,glm::vec2* baslangicNoktasi,glm::vec2*bitisNoktasi)
{
	glBegin(GL_LINE_STRIP);
	for (float theta = baslangicAcisi; theta <= bitisAcisi; theta += increment)
	{

		//x = r * Math.Cos (theta);
```

```
			//y = r * Math.Sin (theta);
			if (theta==baslangicAcisi)
			{
				baslangicNoktasi->x = x + yaricap * cos(theta);
				baslangicNoktasi->y = y + yaricap * sin(theta);
			}
			if (theta== bitisAcisi)
			{
				bitisNoktasi->x = x + yaricap * cos(theta);
				bitisNoktasi->y = y + yaricap * sin(theta);
			}
			glVertex2f(x + yaricap * cos(theta), y + yaricap * sin(theta));
		}
		glEnd();
}
void ArcCizdir(float x, float y, float yaricap, float baslangicAcisi, float bitisAcisi,
float increment)
{
		glBegin(GL_LINE_STRIP);
		for (float theta = baslangicAcisi; theta <= bitisAcisi; theta += increment)
		{

			//x = r * Math.Cos (theta);
			//y = r * Math.Sin (theta);

			glVertex2f(x + yaricap * cos(theta), y + yaricap * sin(theta));
		}
		glEnd();
}


// Drawing routine.
void drawScene(void)
{
		glm::vec2  V4,V7, V10, V11;
		glm::vec2 V9;
		V9.x = 50; V9.y = 55;
		glm::vec2 V8;
		V8.x = 30; V8.y = 60;

		V7.x = 30; V7.y = 50;
		glm::vec2 V5;
		V5.x = 70; V5.y = 60;
		glm::vec2 E3 = V7 - V8;
		glm::vec2 E4 = V9 - V8;
		float theta2 = acos(glm::dot(E3, E4) / (glm::length(E3) * glm::length(E4)));

		glClear(GL_COLOR_BUFFER_BIT);
		//Saatin sol ust yayinin cizimi
		float r = 10;
		float start_angle = 3*PI/2;
		float end_angle = start_angle+theta2;
		glColor3f(1, 0, 0);
		glLineWidth(3.0f);
		ArcCizdir(V8.x, V8.y, r, start_angle, end_angle, PI / 1000,&V7,&V10);
		//Saatin duz kenarlarinin cizimi
		glBegin(GL_LINES);
```

```
	glVertex3f(10, 10, 0);//V1
	glVertex3f(90,10, 0);//V2
	glVertex3f(90, 10, 0);//V2
	glVertex3f(90, 50, 0);//V3
	glVertex3f(10, 10, 0);//V1
	glVertex3f(10, 50, 0);//V6
	glVertex3f(10, 50, 0);//V6
	glVertex3f(30, 50, 0);//V7
	glVertex3f(90, 50, 0);//V3
	glVertex3f(70, 50, 0);//V4
	glEnd();
	//Saatin sag ust yayinin cizimi
	ArcCizdir(V5.x,V5.y, r, 3 * PI / 2-theta2, 3*PI/2, PI / 1000,&V11,&V4);
	//saatin orta ust yayinin cizimi
	glm::vec2 E1 = V10 - V9;
	glm::vec2 E2 = V11 - V9;
	float theta1 = acos(glm::dot(E1, E2) / (glm::length(E1) * glm::length(E2)));
	float r2=glm::length(V8 - V9) - r;
	ArcCizdir(V9.x, V9.y, r2, PI / 2 - theta2, PI - (PI / 2 - theta2), PI / 1000,
&V11, &V10);
	//ic dairenin cizimi
	glLineWidth(1.0f);
	ArcCizdir(V9.x, V9.y, r2 - 2, 0, 2 * PI, PI / 1000);
	//Akrep ve yelkovanin cizimi
	time_t now = time(0);
	tm ltm;
	localtime_s(&ltm,&now);
	//std::cout << ltm.tm_hour << ltm.tm_min << std::endl;
	//Akrep ve yelkovanin cizimi
	glPushMatrix();
	glTranslatef(V9.x,V9.y,0);
	glBegin(GL_LINES);
	glVertex3f(0, 0, 0);
	float akrepUzunlugu = r2 - 8;
	glVertex3f(sin(glm::radians(((float)ltm.tm_hour * 30))) * akrepUzunlugu,
cos(glm::radians((float)ltm.tm_hour*30)) * akrepUzunlugu, 0);
	float yelkovanUzunlugu = r2 - 6;
	glVertex3f(0, 0, 0);
	glVertex3f(sin(glm::radians((float)ltm.tm_min * 6)) * yelkovanUzunlugu,
cos(glm::radians((float)ltm.tm_min)*6) * yelkovanUzunlugu, 0);
	glEnd();
	glFlush();
}

// Initialization routine.
void setup(void)
{
	glClearColor(1.0, 1.0, 1.0, 0.0);
}

// OpenGL window reshape routine.
void resize(int w, int h)
{
	glViewport(0, 0, w, h);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	glOrtho(0.0, 100.0, 0.0, 100.0, -1.0, 1.0);
	glMatrixMode(GL_MODELVIEW);
```

```
	glLoadIdentity();
}

// Keyboard input processing routine.
void keyInput(unsigned char key, int x, int y)
{
	switch (key)
	{
	case 27:
		exit(0);
		break;

	default:
		break;
	}
}



// Main routine.
int main(int argc, char** argv)
{

	glutInit(&argc, argv);

	glutInitContextVersion(4, 3);
	glutInitContextProfile(GLUT_COMPATIBILITY_PROFILE);

	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
	glutInitWindowSize(500, 500);
	glutInitWindowPosition(100, 100);
	glutCreateWindow("Saat.cpp");
	glutDisplayFunc(drawScene);
	glutReshapeFunc(resize);
	glutKeyboardFunc(keyInput);

	glewExperimental = GL_TRUE;
	glewInit();

	setup();

	glutMainLoop();
}
```



c)Ekler

Saat.cpp, Soru2.exe