**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Компьютерная графика»

**Лабораторная работа № 4**

Тема: Ознакомление с технологией OpenGL

Студент: Квапель Александр Дмитриевич

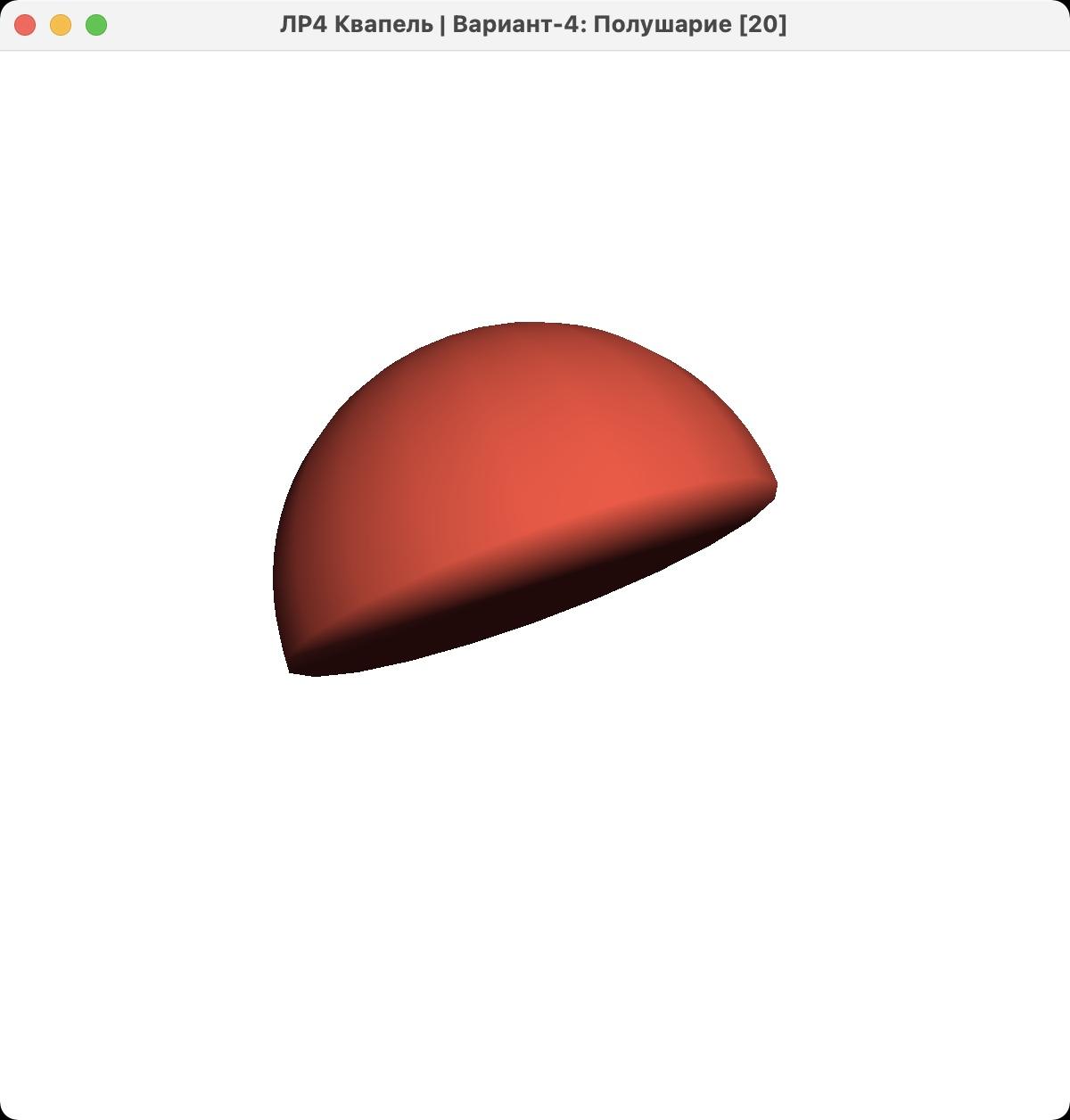
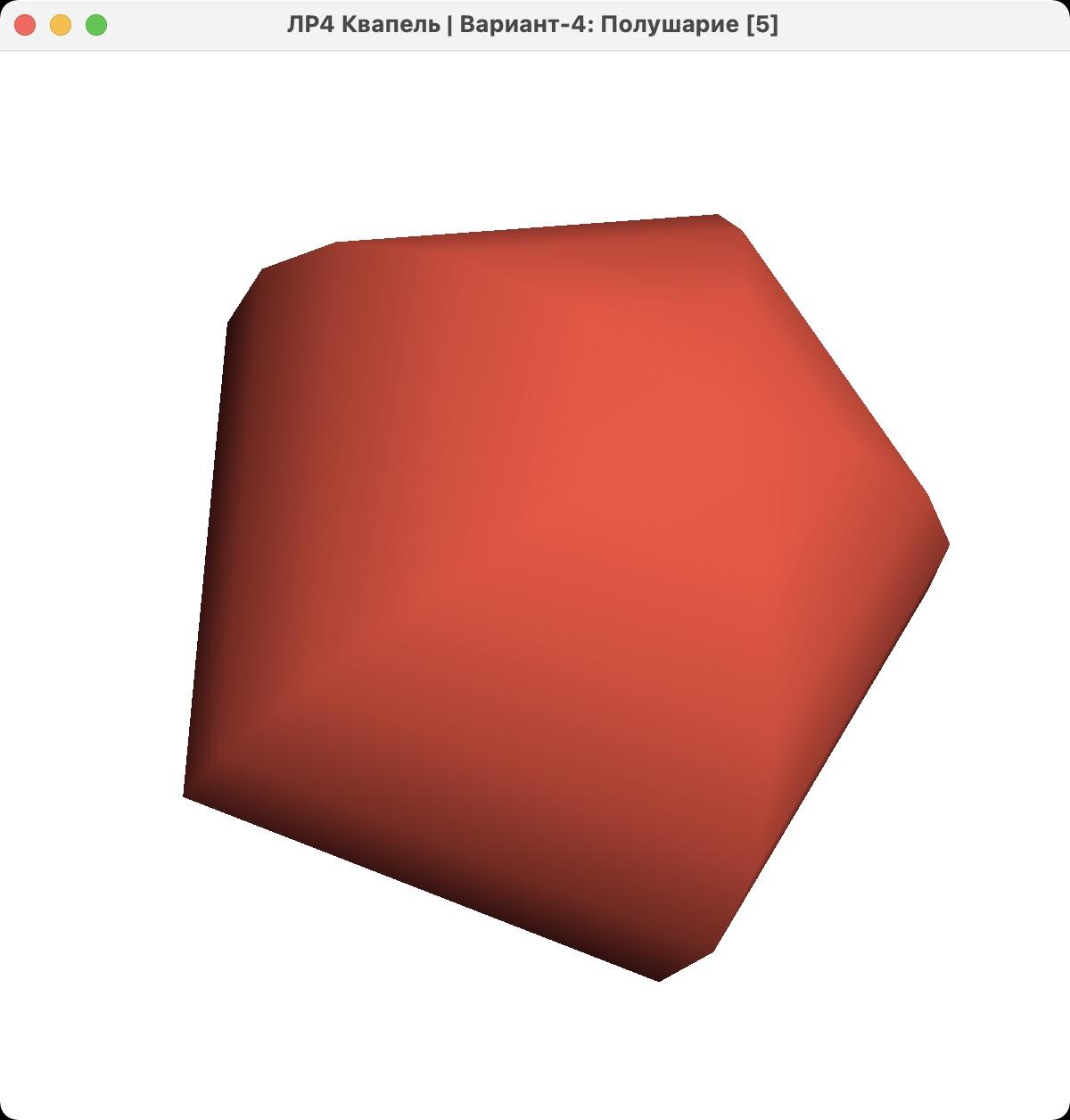
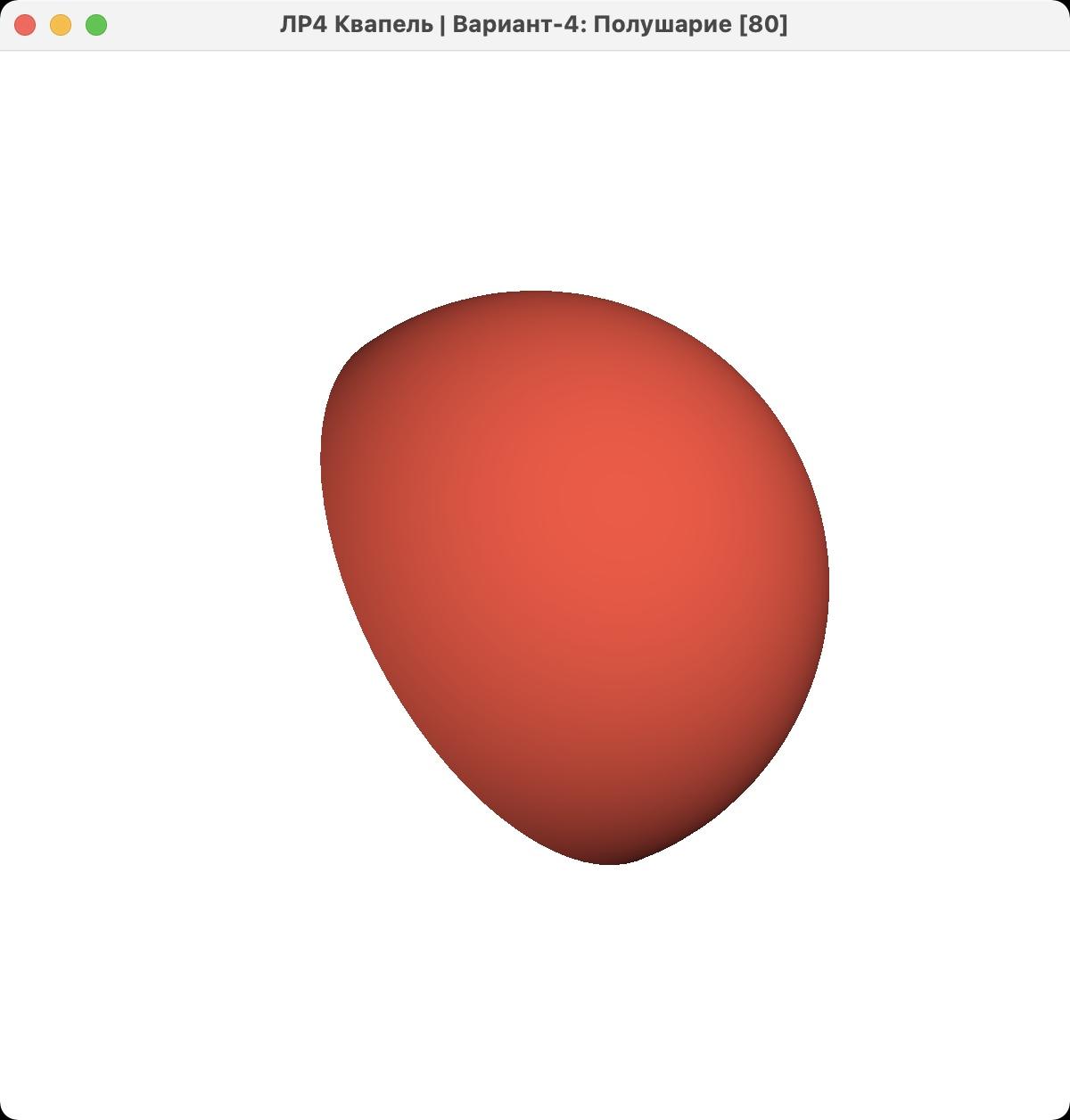
Группа: 08-303

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2021

1. Постановка задачи. Вариант 4  
   Создать графическое приложение с использованием OpenGL. Используя результаты Л.Р.№3, изобразить полушарие с использованием средств OpenGL 2.1. Использовать буфер вершин. Точность аппроксимации тела задается пользователем. Обеспечить возможность вращения и масштабирования многогранника и удаление невидимых линий и поверхностей. Реализовать простую модель освещения на GLSL. Параметры освещения и отражающие свойства материала задаются пользователем в диалоговом режиме.
2. Описание программы   
    Для решения задачи используется язык программирования C#. Используется низкоуровневая библиотека OpenTK, реализующая графическую библиотеку OpenGL, которая используется для построения геометрического тела, задания освещения и реализации вращения и масштабирования изображенного полушария.  
    Программа создает окно указанного размера, в котором будет строиться фигура. Благодаря указанной точности построения (которое пользователь может указывать самостоятельно), строится сетка углов наклона и азимута, где 0<φ<2π, 0<Ѳ<π. Полученные значения программа подставляет в параметрическое уравнение сферы с центром в точке (0, 0, 0) и указанным радиусом. Вычисленные точки передаются в параметр функций построения треугольников в трехмерном пространстве, которые в совокупности изображают полушарие.
3. Руководство по использованию программы  
    Файл Program.cs является “отправной точкой” программы, в которой создается экземпляр окна и, благодаря функции Start, оно отображается пользователю с изображенным полушарием и всем доступным функционалом.   
    Для выполнения пространственных поворотов необходимо использовать клавиши на клавиатуре: Up, Down, Left, Right. Для масштабирования полушария используются клавиши на клавиатуре: +, -. Для увеличения точности построения с шагом 1 следует нажать клавишу “M”, а для уменьшения “N”, значение которой отображается в заголовке окна в квадратных скобках.
4. Результаты выполнения тестов  
   1) Точность 20  
     
   2) Точность 5  
     
   3) Точность 80  
   
5. Листинг программы

Program.cs - точка запуска

// Квапель, М8О-303Б-19

// ЛР4: Создать графическое приложение с использованием OpenGL.

// Используя результаты Л.Р.№3, изобразить заданное тело (то же, что и в л.р. №3) с использованием средств OpenGL 2.1.

// Использовать буфер вершин. Точность аппроксимации тела задается пользователем.

// Обеспечить возможность вращения и масштабирования многогранника и удаление невидимых линий и поверхностей.

// Реализовать простую модель освещения на GLSL.

// Параметры освещения и отражающие свойства материала задаются пользователем в диалоговом режиме.

// Вариант-4: Полушарие.

namespace kvapel\_lab4

{

class MainClass

{

public static void Main(string[] args)

{

Plot plot = new Plot(600, 600);

plot.Start();

}

}

}

Plot.cs - класс окна вывода данных  
using System;

using OpenTK;

using OpenTK.Graphics;

using OpenTK.Graphics.OpenGL;

using OpenTK.Input;

namespace kvapel\_lab4

{

public class Plot

{

protected GameWindow window;

private HalfSphere halfsphere;

private float scaling = 10.0f;

private float xAngle = 0.0f;

private float yAngle = 0.0f;

public Plot(int width, int height)

{

window = new GameWindow(width, height, GraphicsMode.Default, "ЛР4 Квапель | Вариант-4: Полушарие");

window.Load += Window\_Load;

window.Resize += Window\_Resize;

window.RenderFrame += Window\_RenderFrame;

window.UpdateFrame += Window\_UpdateFrame;

window.KeyDown += Window\_KeyDown;

}

public void Start()

{

halfsphere = new HalfSphere(2, 20);

window.Run(1.0 / 60.0);

}

private void Window\_Load(object sender, EventArgs e)

{

GL.ClearColor(1.0f, 1.0f, 1.0f, 1.0f);

GL.Enable(EnableCap.DepthTest);

halfsphere.EnableLighting(1.0f, 0.4f, 0.3f);

}

private void Window\_Resize(object sender, EventArgs e)

{

GL.Viewport(0, 0, window.Width, window.Height);

GL.MatrixMode(MatrixMode.Projection);

GL.LoadIdentity();

Matrix4 matrix = Matrix4.CreatePerspectiveFieldOfView((float)Math.PI / 4, window.Width / window.Height, 1.0f, 100.0f);

GL.LoadMatrix(ref matrix);

GL.MatrixMode(MatrixMode.Modelview);

}

private void Window\_KeyDown(object sender, KeyboardKeyEventArgs e)

{

if (e.Key == Key.M)

{

halfsphere.Precision += 1;

}

else if (e.Key == Key.N)

{

halfsphere.Precision -= 1;

}

if (e.Key == Key.Plus)

{

scaling -= 0.5f;

}

else if (e.Key == Key.Minus)

{

scaling += 0.5f;

}

if (e.Key == Key.Down)

{

xAngle += 10.0f;

if (xAngle > 360.0f)

xAngle = 0.0f;

}

else if (e.Key == Key.Up)

{

xAngle -= 10.0f;

if (xAngle < 0.0f)

xAngle = 360.0f;

}

else if (e.Key == Key.Right)

{

yAngle += 10.0f;

if (yAngle > 360.0f)

yAngle = 0.0f;

}

else if (e.Key == Key.Left)

{

yAngle -= 10.0f;

if (yAngle < 0.0f)

yAngle = 360.0f;

}

}

private void Window\_UpdateFrame(object sender, FrameEventArgs e)

{

window.Title = $"ЛР4 Квапель | Вариант-4: Полушарие [{halfsphere.Precision}]";

}

private void Window\_RenderFrame(object sender, FrameEventArgs e)

{

GL.LoadIdentity();

GL.Clear(ClearBufferMask.ColorBufferBit | ClearBufferMask.DepthBufferBit);

GL.Translate(0.0, 0.0, -scaling);

GL.Rotate(xAngle, 1.0, 0.0, 0.0);

GL.Rotate(yAngle, 0.0, 1.0, 0.0);

halfsphere.Draw();

window.SwapBuffers();

}

}

}

Halfsphere.cs - класс полушария

using System;

using OpenTK.Graphics.OpenGL;

namespace kvapel\_lab4

{

public class HalfSphere

{

private float radius;

private int precision;

public HalfSphere(float radius, int precision)

{

this.radius = radius;

this.precision = precision;

}

public float Radius

{

get { return radius; }

set

{

if (radius <= 0)

radius = 1;

else

radius = value;

}

}

public int Precision

{

get { return precision; }

set

{

if (value <= 2)

precision = 2;

else

precision = value;

}

}

public void Draw()

{

float endPhi = (float)Math.PI \* 2.0f;

float endTheta = (float)Math.PI \* 0.5f;

float dPhi = endPhi / precision;

float dTheta = endTheta / precision;

for (var pointPhi = 0; pointPhi < precision; pointPhi++)

{

for (var pointTheta = 0; pointTheta < precision; pointTheta++)

{

float phi = pointPhi \* dPhi;

float theta = pointTheta \* dTheta;

float phit = (pointPhi + 1 == precision) ? endPhi : (pointPhi + 1) \* dPhi;

float thetat = (pointTheta + 1 == precision) ? endTheta : (pointTheta + 1) \* dTheta;

float[] p0 = { radius \* (float)Math.Sin(theta) \* (float)Math.Cos(phi), radius \* (float)Math.Sin(theta) \* (float)Math.Sin(phi), radius \* (float)Math.Cos(theta) };

float[] p1 = { radius \* (float)Math.Sin(thetat) \* (float)Math.Cos(phi), radius \* (float)Math.Sin(thetat) \* (float)Math.Sin(phi), radius \* (float)Math.Cos(thetat) };

float[] p2 = { radius \* (float)Math.Sin(theta) \* (float)Math.Cos(phit), radius \* (float)Math.Sin(theta) \* (float)Math.Sin(phit), radius \* (float)Math.Cos(theta) };

float[] p3 = { radius \* (float)Math.Sin(thetat) \* (float)Math.Cos(phit), radius \* (float)Math.Sin(thetat) \* (float)Math.Sin(phit), radius \* (float)Math.Cos(thetat) };

GL.Begin(PrimitiveType.Triangles);

GL.Normal3(p0[0] / radius, p0[1] / radius, p0[2] / radius);

GL.Vertex3(p0[0], p0[1], p0[2]);

GL.Normal3(p2[0] / radius, p2[1] / radius, p2[2] / radius);

GL.Vertex3(p2[0], p2[1], p2[2]);

GL.Normal3(p1[0] / radius, p1[1] / radius, p1[2] / radius);

GL.Vertex3(p1[0], p1[1], p1[2]);

GL.Normal3(p3[0] / radius, p3[1] / radius, p3[2] / radius);

GL.Vertex3(p3[0], p3[1], p3[2]);

GL.Normal3(p1[0] / radius, p1[1] / radius, p1[2] / radius);

GL.Vertex3(p1[0], p1[1], p1[2]);

GL.Normal3(p2[0] / radius, p2[1] / radius, p2[2] / radius);

GL.Vertex3(p2[0], p2[1], p2[2]);

GL.Normal3(p0[0] / radius, p0[1] / radius, 0);

GL.Vertex3(p0[0], p0[1], 0);

GL.Normal3(p2[0] / radius, p2[1] / radius, 0);

GL.Vertex3(p2[0], p2[1], 0);

GL.Normal3(p1[0] / radius, p1[1] / radius, 0);

GL.Vertex3(p1[0], p1[1], 0);

GL.Normal3(p3[0] / radius, p3[1] / radius, 0);

GL.Vertex3(p3[0], p3[1], 0);

GL.Normal3(p1[0] / radius, p1[1] / radius, 0);

GL.Vertex3(p1[0], p1[1], 0);

GL.Normal3(p2[0] / radius, p2[1] / radius, 0);

GL.Vertex3(p2[0], p2[1], 0);

GL.End();

}

}

}

public void EnableLighting(float r, float g, float b)

{

float[] lightPosition = { 20, 20, 80 };

float[] lightDiffuse = { r, g, b };

float[] lightAmbient = { 0.4f, 0.0f, 0.0f };

GL.Enable(EnableCap.Lighting);

GL.Light(LightName.Light0, LightParameter.Position, lightPosition);

GL.Light(LightName.Light0, LightParameter.Diffuse, lightDiffuse);

GL.Light(LightName.Light0, LightParameter.Ambient, lightAmbient);

GL.Enable(EnableCap.Light0);

}

public void DisableLighting()

{

GL.Disable(EnableCap.Light0);

GL.Disable(EnableCap.Lighting);

}

}

}

1. Выводы

В данной лабораторной работе была разработана процедура аппроксимации Полушария с заданной пользователем точностью с использованием технологии OpenGL 2.1, обеспечена возможность пространственных поворотов и масштабирования тела с помощью клавиатуры. Реализована простая модель освещения тела с одним источником света.

Список литературы

1. Справочник по OpenGL 2.1 [Электронный ресурс]. URL:<https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/> (дата обращения 04.11.2021).
2. Справочник по OpenTK [Электронный ресурс]. URL:<https://opentk.net/learn/index.html> (дата обращения 04.11.2021).