Московский авиационный институт
(Национальный исследовательский университет)

Институт: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»
Дисциплина: «Компьютерная графика»

Лабораторная работа № 3

Тема: Основы построения фотореалистичных изображений

Студент: Махмудов Орхан
Группа: О8-305
Преподаватель: Чернышов Л.Н.
Дата:
Оценка:

Москва, 2020

1. Постановка задачи

.**Тема:** Основы построения фотореалистичных изображений
Задание: Используя результаты Л.Р.№2, аппроксимировать заданное тело выпуклым многогранником. Точность аппроксимации задается пользователем. Обеспечить возможность вращения и масштабирования многогранника и удаление невидимых линий и поверхностей. Реализовать простую модель закраски для случая одного источника света

2. Описание программы

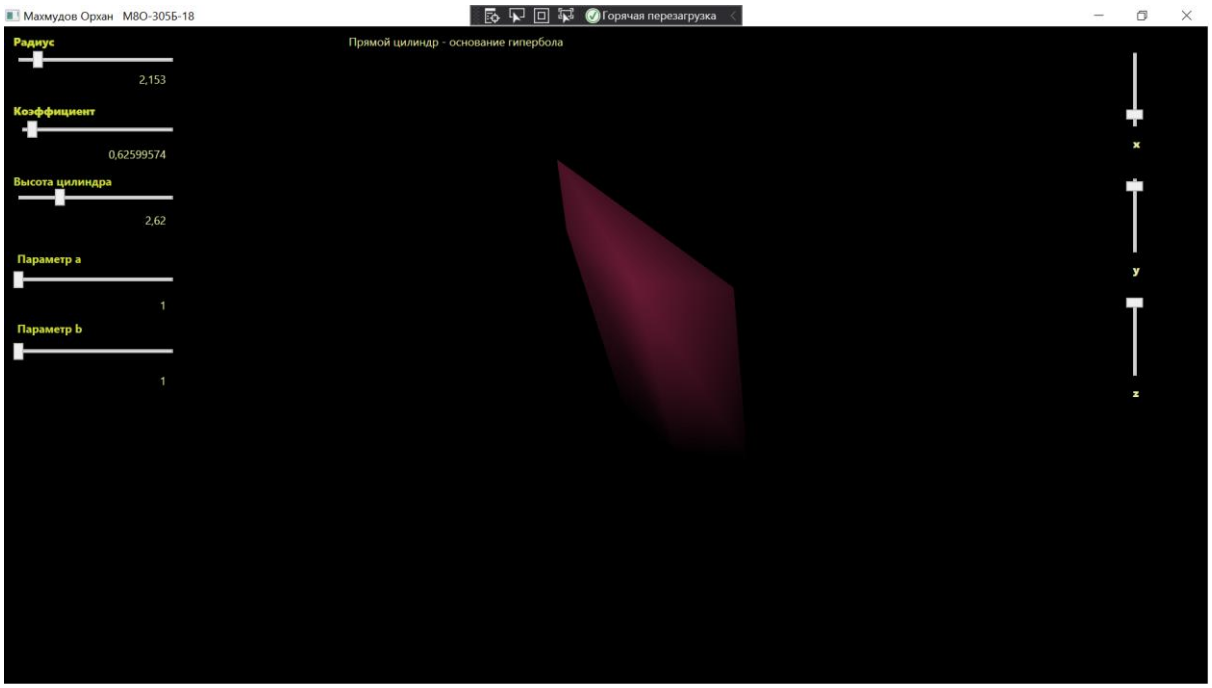
Язык программирования: C#
Используемые библиотеки (пакеты): System.Windows.Media.Media3D.
Используемая среда: Visual Studio 2019
Ввод: Ввод с консоли не требуется, все параметры задаются через окно, где и производится отображение фигуры.
Вывод: 3d-фигура, прямой цилиндр с основанием - гипербола.
Используемые структуры данных: массивы, двумерные массивы.
Краткая инструкция для пользователя: Поворот фигуры осуществляется тремя вертикальными ползунками по осям x,y,z. Можно изменять радиус цилиндра с помощью горизонтального ползунка, изменять аппроксимацию - через горизонтальный ползунок с подписью “коэффициент аппроксимации”. Задача аппроксимации в моем случае - определить, сколько сторон минимально должен иметь правильный многоугольник, чтобы соответствовать коэффициенту аппроксимации. По коэффициенту аппроксимации и задаваемому радиусу цилиндра можно найти радиус окружности, в которую можно вписать аппроксимированные многоугольниками основания цилиндра. Далее, через формулы описанной r = и вписанной окружностей (где a 2tg(π/n) R = a 2sin(π/n) a - длина стороны правильного многоугольника) можно найти искомое количество сторон многоугольника n = , которым аппроксимируются основания цилиндра.

3. Набор тестов

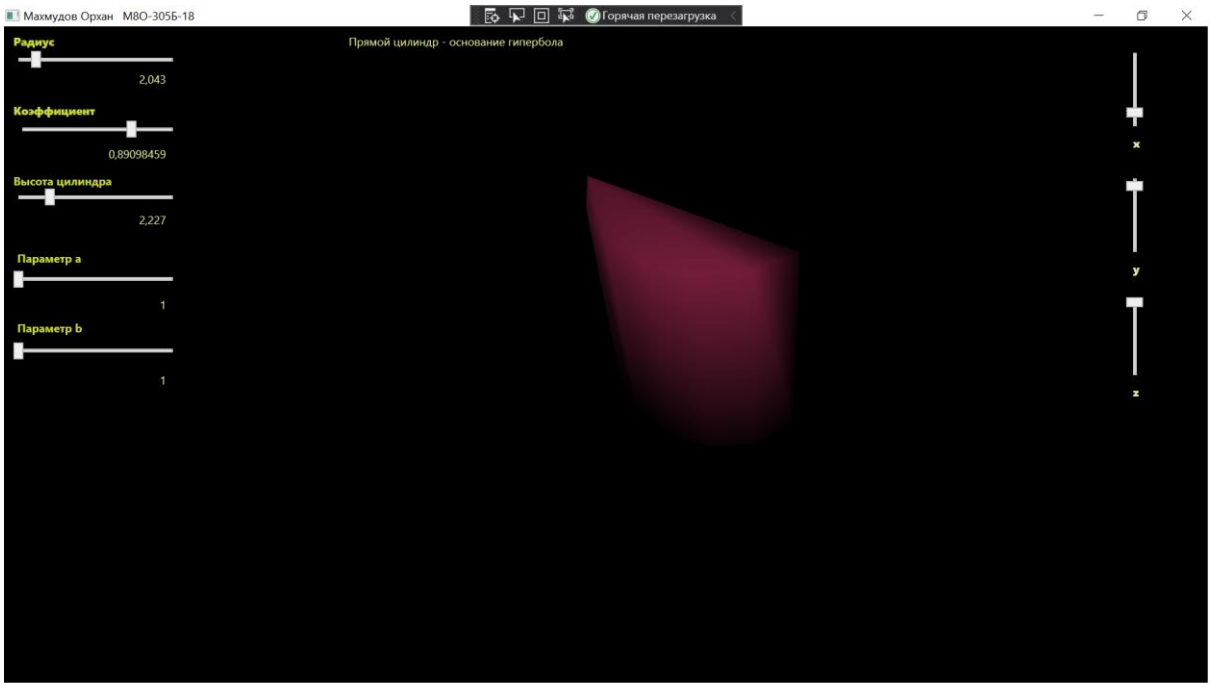
№ теста	коэффициент аппроксимации	радиус цилиндра	высота цилиндра	ось поворота
1	0,62599574	2,153	2,62	ось x
2	0,89098459	2,043	2,227	ось x
3	0,99999999	2,558	2,497	ось z

4. Результаты выполнения тестов

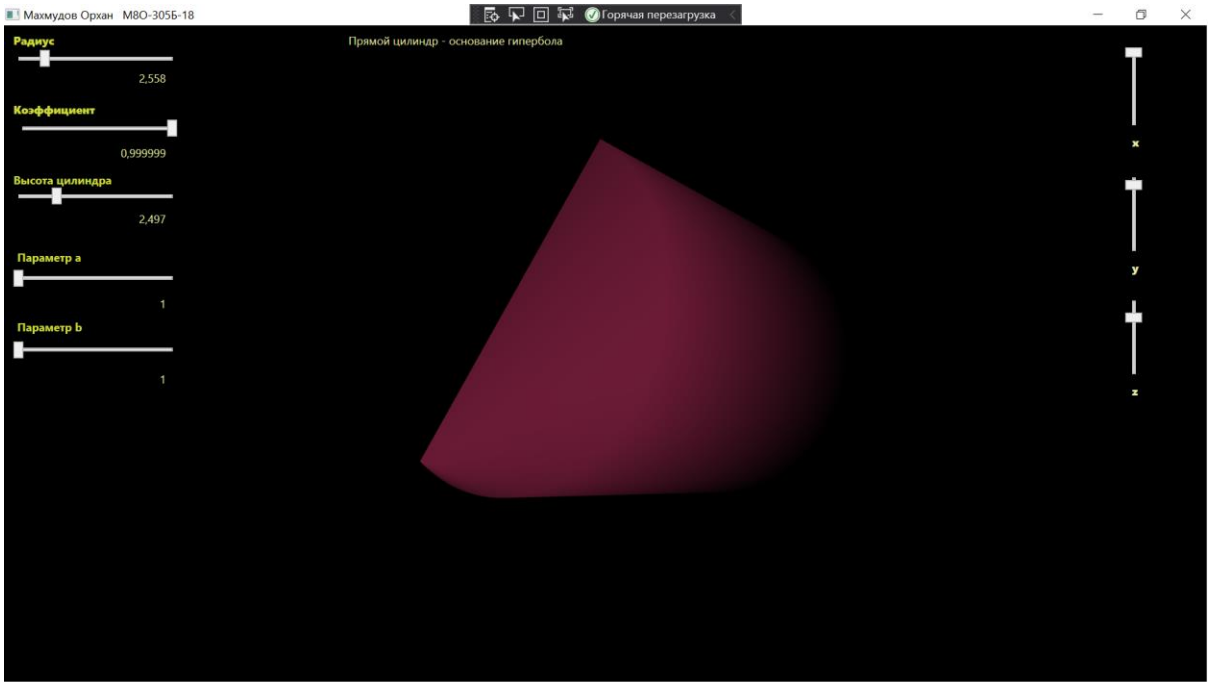
№ теста	коэффициент аппроксимации	радиус цилиндра	высота цилиндра	ось поворота
1	0,62599574	2,153	2,62	ось x



№ теста	коэффициент аппроксимации	радиус цилиндра	высота цилиндра	ось поворота
2	0,89098459	2,043	2,227	ось x



№ теста	коэффициент аппроксимации	радиус цилиндра	высота цилиндра	ось поворота
3	0,99999999	2,558	2,497	ось z



5. Листинг программы

TruncatedPyramid.cs

```
// Махмудов Орхан М80-305Б-18
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Media.Media3D;

namespace CG_Lab_3
{
   class TruncatedPyramid
   {
      private Point3D[] bottom_points;
      private Point3D[] top_points;
      public TruncatedPyramid(Point3D[] bottom_points, Point3D[] top_points)
      {
         this.bottom_points = bottom_points;
         this.top_points = top_points;
      }
      public void Draw(GeometryModel3D geometry_model_3d)
      {
         MeshGeometry3D mesh = new MeshGeometry3D();
         // Заносим вершины в массивы
         foreach (Point3D point in bottom_points) mesh.Positions.Add(point);
         foreach (Point3D point in top_points) mesh.Positions.Add(point);
         int n = bottom_points.Length;
         // Рисуем основания
         for (int i = 2; i < n; i++)
         {
            AddTriangle(mesh, 0, i, i-1);
            AddTriangle(mesh, i - 1 + n, i + n, n);
```

```
            }
            // Рисуем боковые грани
            for (int i = 0; i < n-1; i++)
            {
                AddTriangle(mesh, i, i+1, n+i);
                AddTriangle(mesh, i+1, n+i+1, n+i);
            }
            AddTriangle(mesh, 0, n, 2*n-1);
            AddTriangle(mesh, 0, 2*n-1, n-1);
            geometry_model_3d.Geometry = mesh;
        }
        // прорисовка треугольника по трем точкам
        private void AddTriangle(MeshGeometry3D mesh, int A, int B, int C)
        {
            mesh.TriangleIndices.Add(A);
            mesh.TriangleIndices.Add(B);
            mesh.TriangleIndices.Add(C);
        }
    }
}
```

MainWindow.xaml.cs

```
using System;
using System.Windows;
using System.Windows.Media;
using System.Windows.Media.Media3D;
namespace CG_Lab_3
{
    /// <summary>Логика взаимодействия для MainWindow.xaml</summary>
    public partial class MainWindow : Window
    {
        private int angle_number=3;       // кол-во углов пирамиды
        private int level_number = 1;   // общее кол-во пирамид
        private double r = 1;           // радиус цилиндра
        private double Approx = 0.6;    // Число аппорксимации, r/R
        private double Height = 1;      // Высота цилиндра
        private double tmpApprox;
        private double r_a = 1;
        private double r_b = 1;
        /// <summary> массив вершин нижнего основания</summary>
        private Point3D[] bottom_points;
        /// <summary> массив вершин верхнего основания</summary>
        private Point3D[] top_points;
        public MainWindow()
        {
            InitializeComponent();
        }
        /// <summary> Основная функция, осуществляющая аппроксимацию </summary>
        public void Approximate()
        {
            bottom_points = new Point3D[angle_number];
            top_points = new Point3D[angle_number];
            group.Children.Clear();
            SolidColorBrush brush = new SolidColorBrush(Color.FromRgb(120, 30, 60));
```

```
    DiffuseMaterial material = new DiffuseMaterial(brush);
    CountBottomPoints();
    GeometryModel3D geonetry_model_3d = new GeometryModel3D();
    geonetry_model_3d.Material = material;
    CountTopPoints(level_number);
    // Создание и отрисовка одной пирамиды
    TruncatedPyramid pyramid = new TruncatedPyramid(bottom_points, top_points);
    pyramid.Draw(geonetry_model_3d);
    group.Children.Add(geonetry_model_3d); // присоединение пирамиды к многограннику
    for (int i = 0; i < angle_number; i++) bottom_points[i] = top_points[i];
}
/// <summary>Просчитывание координат точек нижнего основания </summary>
private void CountBottomPoints()
{
    double step = Math.PI / (angle_number + 1); // точно
    double alpha = Math.PI ; // начало, пофиг
    for (int i = 0; i < angle_number; i++)
    {
        double x = r*r_a * Math.Cos(alpha);
        double y = r*r_b * Math.Sin(alpha);
        bottom_points[i] = new Point3D(x, y, 0);
        alpha += step;
    }
}
/// <summary>Просчитывание координат точек верхнего основания</summary>
private void CountTopPoints(int current_level)
{
    double alpha = Math.PI / (4 * level_number + 1);
    for (int i = 0; i < angle_number; i++)
    {
        Point3D A = bottom_points[i];
        double z = Height *r  * Math.Sin((current_level + 1) * alpha);
        double x = A.X;
        double y = A.Y;
        top_points[i] = new Point3D(x, y, z);
    }
}
/// <summary>считывание параметров</summary>
private void ReadParams()
{
    try
    {
        if (Slider_Height != null && TextBlock_Height != null)
        {
            Height = Math.Round(Convert.ToDouble(Slider_Height.Value), 3);
            TextBlock_Height.Text = Convert.ToString(Height);
        }
    }
    catch { }
    try
    {
        Approx = Convert.ToDouble(slider_Approx.Value);
```

```
            tmpApprox = Approx;
            angle_number = Convert.ToInt32(Math.PI / Math.Acos(Approx));
            if (angle_number < 3)
                angle_number = 3;
            if (textBlock_Approx != null)
                textBlock_Approx.Text = Convert.ToString(Approx);
        }
        catch { }
        try
        {
            if (Slider_Radius != null && TextBlock_Radius != null)
            {
                r = Math.Round(Convert.ToDouble(Slider_Radius.Value), 3);
                TextBlock_Radius.Text = Convert.ToString(r);
            }
        }
        catch { }
        try
        {
            if (slider_a != null && TextBlock_a!= null)
            {
                r_a = Math.Round(Convert.ToDouble(slider_a.Value), 3);
                TextBlock_a.Text = Convert.ToString(r_a);
            }
        }
        catch { }
        try
        {
            if (slider_b != null && TextBlock_b != null)
            {
                r_b = Math.Round(Convert.ToDouble(slider_b.Value), 3);
                TextBlock_b.Text = Convert.ToString(r_b);
            }
        }
        catch { }
    }
    /// <summary>
    /// В случае изменения числа аппроксимации перерисовать фигуру с новыми параметрами
    /// </summary>
    private void slider_Approx_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)
    {
        ReadParams();
        Approximate();
    }
    /// <summary>
    /// В случае изменения радиуса перерисовать фигуру с новыми параметрами
    /// </summary>
    private void Slider_Radius_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)
    {
        ReadParams();
        Approximate();
    }
```

```
        /// <summary>
        /// В случае изменения высоты перерисовать фигуру с новыми параметрами
        /// </summary>
        private void Slider_Height_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)
        {
            ReadParams();
            Approximate();
        }
        private void slider_x_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)
        {
            // try
            // {
            if (slider_Approx != null)
            Approx = Convert.ToDouble(slider_Approx.Value);

            //if (angle_number < 3)
            //    angle_number = 3;
            /*              if (textBlock_Approx != null && slider_Approx != null)
                        {
                            Approx = Convert.ToDouble(slider_Approx.Value);
                            angle_number = Convert.ToInt32(Math.PI / Math.Acos(Approx));
                            textBlock_Approx.Text = Convert.ToString(Approx);
                        }*/
            //}
            //catch { }
        }
        private void Slider_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)
        {
            ReadParams();
            Approximate();
        }
        private void Slider_ValueChanged_1(object sender, RoutedPropertyChangedEventArgs<double> e)
        {
            ReadParams();
            Approximate();
        }
    }
}
```

6. Вывод

В ходе данной лабораторной работы были приобретены навыки по аппроксимации тела выпуклым многогранником, была обеспечена возможность вращения и масштабирования многогранника, удаление невидимых линий, а также была разобрана и запрограммирована простая модель закраски для случая одного источника света.

Список литературы

1. Статья по введению в xaml [Электронный ресурс]. URL: https://metanit.com/sharp/wpf/2.php.
2. Документация Microsoft по xaml (C#) [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/dotnet/desktop/wpf/fundamentals/xaml?view=netdesktop-5.0.
3. Статья по xaml “Понимание xaml” [Электронный ресурс]. URL: https://habr.com/ru/post/141069/