Московский авиационный институт
(Национальный исследовательский университет)
Факультет прикладной математики и физики
Кафедра вычислительной математики и программирования

Курсовой проект

по курсу «Компьютерная графика».
Тема: «Каркасная визуализация поверхности».

Студентка: Довженко А.А.
Группа: 80-307Б
Преподаватель: Чернышов Л.Н.
Оценка:

Москва, 2018

Содержание:

Постановка задачи

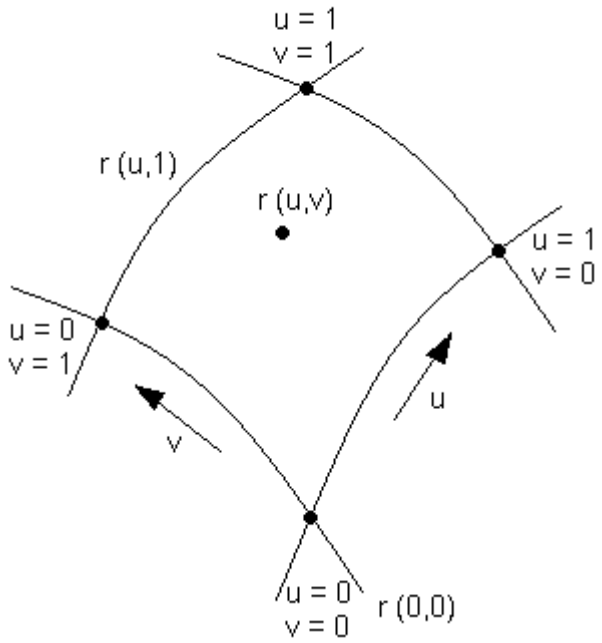
Составить и отладить программу, обеспечивающую каркасную визуализацию порции поверхности заданного типа. Исходные данные готовятся самостоятельно и вводятся из файла или в панели ввода данных. Должна быть обеспечена возможность тестирования программы на различных наборах исходных данных. Программа должна обеспечивать выполнение аффинных преобразований для заданной порции поверхности, а также возможность управлять количеством изображаемых параметрических линий. Для визуализации параметрических линий поверхности разрешается использовать только функции отрисовки отрезков в экранных координатах.

Вариант 7 Линейная поверхность Кунса (границы – Cardinal Spline 3D)

Решение задачи

Поверхность Кунса:

Поверхность Кунса - это бикубическая поверхность, ограниченная четырьмя пространственными кривыми.



Сетка кривых разбивает поверхность на совокупность ячеек, каждая из которых ограничена, параметрически представлена парой u – кривых и v – кривых.

Заданная ячейка поверхности находится в пределах:

$0 \le u \le 1$

$0 \le v \le 1$

и представляет собой исходную часть поверхности, ограниченную четырьмя исходными границами. Форрест предложил наглядную трактовку поверхности Кунса.

Кардинальные сплайны.

Сплайн — функция, область определения которой разбита на конечное число отрезков, на каждом из которых она совпадает с некоторым алгебраическим многочленом (полиномом). Максимальная из степеней использованных полиномов называется степенью сплайна.

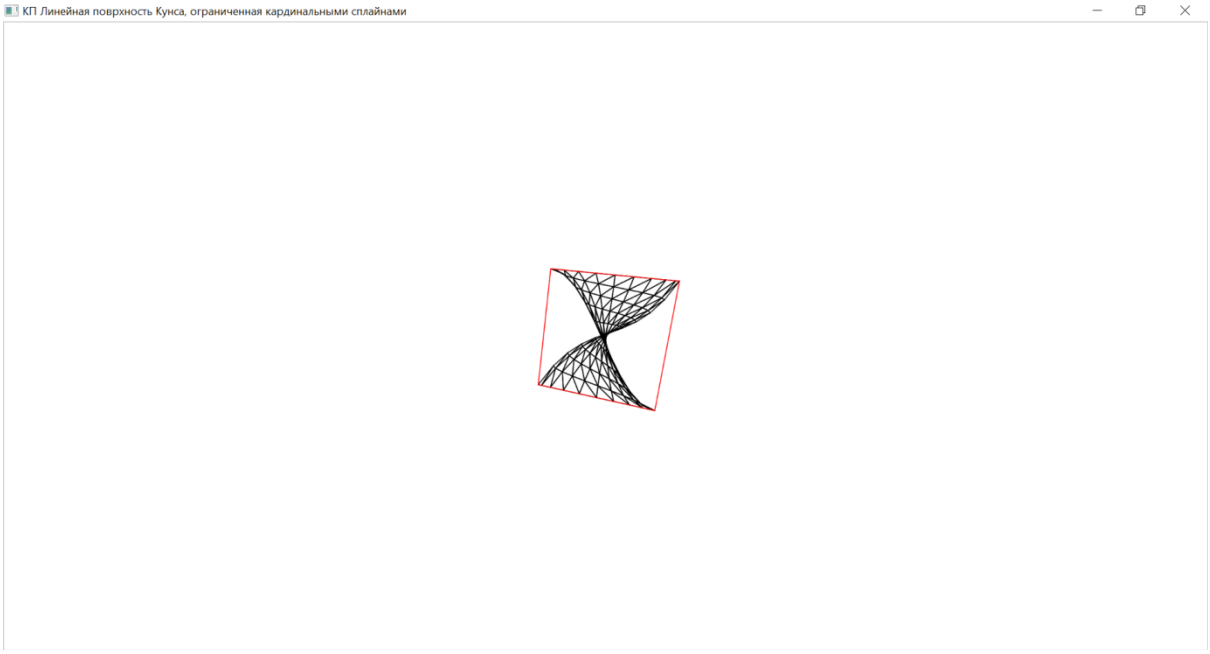
Число фрагментов. Очевидно, что минимальное число фрагментов — один. Классическое определение сплайна ограничивает число фрагментов определённым числом на конечном отрезке. Однако можно строить сплайны и с бесконечным числом фрагментов, а реально

эти методы и алгоритмы, которые не нуждаются в информации об определённом количестве фрагментов. Представителями этих сплайнов являются **кардинальные** сплайны, исследованные Шенбергом. Для построения сплайнов с неограниченным числом фрагментов лучше подходят локальные сплайны.

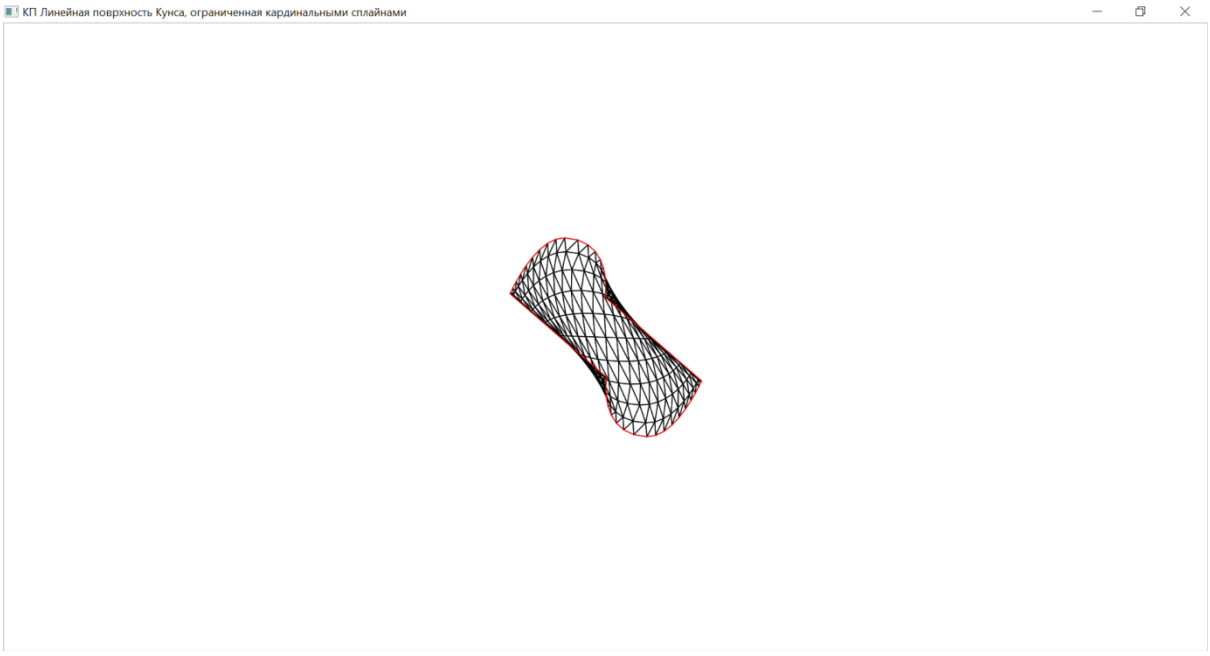
Руководство по использованию программы и пример работы

Запустить IntelliJ, открыть в нём проект с программой и запустить его.

Пример 1.



Пример 2.



Листинг программы

```
// Довженко А.А. М8О-307Б
// КП
// Линейная поврхность Кунса, ограниченная кардинальными сплайнами
// Главный класс
package main;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));
        primaryStage.setTitle("КП Линейная поврхность Кунса, ограниченная
кардинальными сплайнами");
        primaryStage.setScene(new Scene(root, 300, 275));
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}

package main;


import javafx.event.EventHandler;
import javafx.event.EventType;
import javafx.fxml.FXML;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.scene.text.Text;
```

```
public class Controller {

    private ResizableCanvas canvas;

    double xStart;
    double yStart;
    double xEnd;
    double yEnd;

    @FXML
    public AnchorPane canvasHolder;

    @FXML
    void initialize() {
        canvas = new ResizableCanvas(canvasHolder.getWidth(),
canvasHolder.getWidth());

        AnchorPane.setTopAnchor(canvas, 0.);
        AnchorPane.setBottomAnchor(canvas, 0.);
        AnchorPane.setLeftAnchor(canvas, 0.);
        AnchorPane.setRightAnchor(canvas, 0.);
        canvasHolder.getChildren().add(canvas);

        canvas.resize(canvasHolder.getWidth(), canvasHolder.getHeight());

        canvas.addEventHandler(MouseEvent.MOUSE_PRESSED, event -> {
            xStart = event.getScreenX();
            yStart = event.getScreenY();
            xEnd = event.getScreenX();
            yEnd = event.getScreenY();
        });

        canvas.addEventHandler(MouseEvent.MOUSE_DRAGGED, event -> {

            xEnd = event.getScreenX();
            yEnd = event.getScreenY();
```

```
            double dX = xEnd - xStart;
            double dY = yEnd - yStart;

            canvas.matrix.mult(TransformMatrix.rotationY(dX));
            canvas.matrix.mult(TransformMatrix.rotationX(dY));

            xStart = xEnd;
            yStart = yEnd;

            canvas.resize(canvas.width, canvas.height);
        });
    }
}

package main;

import javafx.scene.canvas.GraphicsContext;
import javafx.scene.paint.Color;

import java.util.Vector;

public class Pencil {
    Pencil() {
    }

    static void drawVector(Vector<Point> points, TransformMatrix matrix,
ResizableCanvas canvas, Color color) {
        Vector<Point> transformed = new Vector<>();
        for (int i = 0; i < points.size(); i++) {
            transformed.add( matrix.transform(points.get(i)).scale() );
        }

        GraphicsContext gc = canvas.getGraphicsContext2D();
        gc.setStroke(color);

        double xCenter = canvas.width / 2;
```

```
        double yCenter = canvas.height / 2;

        for (int i = 0; i < transformed.size() - 1; i++) {
            Point p1 = transformed.get(i);
            Point p2 = transformed.get(i + 1);

            gc.strokeLine(xCenter + p1.x, yCenter - p1.y, xCenter + p2.x, yCenter - p2.y);

        }
    }

    static void drawSurface(Vector<Vector<Point>> surface, TransformMatrix matrix, ResizableCanvas canvas) {
        Vector<Vector<Point>> points = new Vector<>();

        for (int i = 0; i < surface.size(); i++) {
            Vector<Point> tmp = new Vector<>();

            Vector<Point> current = surface.get(i);
            for (int j = 0; j < surface.get(i).size(); j++) {
                tmp.add( matrix.transform(current.get(j)).scale() );
            }
            points.add(tmp);
        }

        GraphicsContext gc = canvas.getGraphicsContext2D();
        gc.setStroke(Color.BLACK);

        double xCenter = canvas.width / 2;
        double yCenter = canvas.height / 2;

        for (int i = 0; i < points.size() - 1; i++) {
            Vector<Point> cur = points.get(i);
            Vector<Point> next = points.get(i + 1);
            for (int j = 0; j < cur.size() - 1; j++) {
```

```
            drawTriangle(cur.get(j), cur.get(j + 1), next.get(j), gc, xCenter,
yCenter);
            drawTriangle(cur.get(j + 1), next.get(j), next.get(j + 1), gc, xCenter,
yCenter);

          }
       }
    }

    private static void drawTriangle(Point p1, Point p2, Point p3,
GraphicsContext gc, double xCenter, double yCenter) {
       gc.strokeLine(xCenter + p1.x, yCenter - p1.y, xCenter + p2.x, yCenter -
p2.y);
       gc.strokeLine(xCenter + p1.x, yCenter - p1.y, xCenter + p3.x, yCenter -
p3.y);
       gc.strokeLine(xCenter + p3.x, yCenter - p3.y, xCenter + p2.x, yCenter -
p2.y);
    }

}

package main;

import java.util.Vector;

public class Point {
    public double x;
    public double y;
    public double z;
    public double w;


    Point(double x, double y, double z, double w) {
       this.x = x;
       this.y = y;
       this.z = z;
       this.w = w;
```

```
    }

    Point(Point p) {
        this.x = p.x;
        this.y = p.y;
    }

    public static Point sum(Point v1, Point v2) {
        return new Point(v1.x + v2.x, v1.y + v2.y, v1.z + v2.z, v2.w);
    }

    public static Point sub(Point v1, Point v2) {
        return new Point(v2.x - v1.x, v2.y - v1.y, v2.z - v1.z, v2.w);
    }

    public static Point mul(Point v1, double m) {
        return new Point(v1.x * m, v1.y * m, v1.z * m, v1.w);
    }


    public Point scale() {
        return new Point(x * w, y * w, z * w, 1);
    }
}

//Довженко А.А. М8О-307Б
//Холст

package main;

import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.paint.Color;



import java.io.*;
import java.util.Vector;
```

```
import static main.Pencil.*;

public class ResizableCanvas extends Canvas{
    int N;
    public double height,width;

    TransformMatrix matrix;

    Spline spline1;
    Spline spline2;
    Spline spline3;
    Spline spline4;

    Surface surface;

    @Override
    public boolean isResizable() {
        return true;
    }

    public ResizableCanvas(double width, double height) {
        super(width,height);
        this.width = width;
        this.height = height;
        matrix = new TransformMatrix();

        getPoints();
        surface = new Surface();
        surface.countSurface(spline1, spline2, spline3, spline4);
    }

    public void getPoints() {
        BufferedReader file = new BufferedReader(new
InputStreamReader(System.in));
        String filename = getLine(file);
        FileReader fr = null;
```

```
try {
   fr = new FileReader(filename);
} catch (FileNotFoundException e) {
   e.printStackTrace();
}

BufferedReader in = new BufferedReader(fr);

String input;
String[] parsed;
input = getLine(in);
N = Integer.parseInt(input);

int n,m;
parsed = getLines(in);
n = Integer.parseInt(parsed[0]);
m = Integer.parseInt(parsed[1]);

Vector<Vector<Point>> points = new Vector<>();
for (int i = 0; i < 2; i++) {
   Vector<Point> v1 = new Vector<>();
   Vector<Point> v2 = new Vector<>();

   parsed = getLines(in);
   for (int j = 0; j < n; j++) {
      v1.add(new Point(
            Double.parseDouble(parsed[j * 3]),
            Double.parseDouble(parsed[j * 3 + 1]),
            Double.parseDouble(parsed[j * 3 + 2]), 1
      ));
   }

   parsed = getLines(in);
   for (int j = 0; j < m; j++) {
      v2.add(new Point(
            Double.parseDouble(parsed[j * 3]),
            Double.parseDouble(parsed[j * 3 + 1]),
```

```
                Double.parseDouble(parsed[j * 3 + 2]), 1
            ));
        }
        points.add(v1);
        points.add(v2);
    }

    System.out.println(N + "\n" + n + " " + m);
    for (int i = 0; i < points.size(); i++) {
        Vector<Point> pts = points.get(i);
        for (int j = 0; j < pts.size(); j++) {
            System.out.println(pts.get(j).x + " " + pts.get(j).y + " " + pts.get(j).z);
        }
        System.out.print("\n");
    }

    spline1 = new Spline(points.get(0), N);
    spline2 = new Spline(points.get(1), N);
    spline3 = new Spline(points.get(2), N);
    spline4 = new Spline(points.get(3), N);
}

private String getLine(BufferedReader in) {
    String input = null;
    try {
        input = in.readLine();

    } catch (IOException e) {
        e.printStackTrace();
    }
    return input;
}

private String[] getLines(BufferedReader in) {
    String[] input = null;
    try {
        input = in.readLine().split("\\s");
```

```
        } catch (IOException e) {
            e.printStackTrace();
        }
        return input;
    }


    @Override
    public void resize(double width,double height) {
        //во время изменения размера происходит отрисовка
        setWidth(width);
        setHeight(height);

        GraphicsContext gc = this.getGraphicsContext2D();
        gc.setFill(Color.WHITE);
        gc.fillRect(0,0,getWidth(),getHeight());

        this.height = height;
        this.width = width;

        gc.setFill(Color.rgb( 72, 61,139));
        gc.setFill(Color.GOLD);
        gc.setStroke(Color.DARKTURQUOISE);


        matrix.matrix[3][3] = (int)(Math.min(width,height) / 10);
        drawSurface(surface.surface, matrix,this);

        drawVector(spline1.spline(true), matrix, this, Color.rgb(255, 0,0));
        drawVector(spline2.spline(true), matrix, this, Color.rgb(255, 0,0));
        drawVector(spline3.spline(true), matrix, this, Color.rgb(255, 0,0));
        drawVector(spline4.spline(true), matrix, this, Color.rgb(255, 0,0));
    }
}

package main;
```

```
import java.util.Vector;

import static main.Point.*;

public class Spline {

    double step;
    Point p1;
    Point p2;
    Point p3;

    Vector<Point> points;
    int n;

    Spline(Vector<Point> pts, int n) {

        this.step = 1. / (double) n;
        this.n = n;

        points = new Vector<>();
        points.addAll(pts);
    }

    public Vector<Point> spline(boolean flag) {

        Vector<Point> ret = new Vector<>();

        for (int i = 0; i < points.size() - 1; i++) {

            ret.addAll(count(i, flag));
        }
        ret.add(points.get(points.size() - 1));
        return ret;
    }

    private Vector<Point> count(int k, boolean flag) {
```

```
Point v1 = points.get(k);
Point v2 = points.get(k + 1);

Vector<Point> ret = new Vector<>();
Point p1 = points.get(k);
Point p2;

if(k == 0) {
   p2 = new Point(0,0,0,1);
}
else {
   p2 = m(points.get(k - 1), points.get(k + 1), 2 * step);
}

Point p3 = points.get(k + 1);
Point p4;
if(k == points.size() - 2 || k == points.size() - 1) {
   p4 = new Point(0,0,0,1);
}
else {
   p4 = m(points.get(k), points.get(k + 2), 2 * step);
}

double t = 0;
ret.add(v1);
for (int i = 0; i < n - 1; i++) {
   t += step;
   Point tmp1 = mul(p1,  2 * Math.pow(t,3) - 3 * Math.pow(t,2) + 1);
   Point tmp2 = mul(p2, (Math.pow(t,3) - 2 * Math.pow(t,2) + t) * 0.25);
   if(flag) {
      tmp2 = mul(tmp2, v2.x - v1.x);
   } else {
      tmp2 = mul(tmp2, v2.y - v1.y);
   }

   Point tmp3 = mul(p3, -2 * Math.pow(t,3) + 3 * Math.pow(t,2));
   Point tmp4 = mul(p4, (Math.pow(t,3) - Math.pow(t,2)) * 0.25);
```

```
            if(flag) {
                tmp4 = mul(tmp4, v2.x - v1.x);
            } else {
                tmp4 = mul(tmp4, v2.y - v1.y);
            }

            ret.add( sum( sum(tmp1, tmp2), sum(tmp3, tmp4)  ));
        }

        return ret;
    }

    private Point m(Point v1, Point v2, double dt) {
        return mul(sub(v1,v2), 1. / dt);
    }
}

package main;
import java.util.Vector;

import static main.Point.*;

public class Surface {
    public Vector<Vector<Point>> surface;

    Surface() {
        surface = new Vector<>();
    }


    void countSurface(Spline border1, Spline border2, Spline border3, Spline
border4) {
        if(border1.n != border3.n || border2.n != border4.n) {
            System.exit(1);
        }

        double step1 = border1.step;
```

```
        double step2 = border2.step;

        double w = 0;
        double u;

        Vector<Point> a = border1.spline(true); // 1 || 3
        Vector<Point> d = border2.spline(true); // 2 || 4
        Vector<Point> b = border3.spline(true);
        Vector<Point> c = border4.spline(true);

        int n = a.size();
        int m = c.size();

        for (int j = 0; j < n; j++) {
            Vector<Point> tmp = new Vector<>();
            u = j * border1.step;
            w = 0;
            for (int i = 0; i < m; i++) {
                w = i * border2.step;

                Point tmp1 = sum( mul( a.get(j), 1 - w), mul(b.get(j), w) );
                Point tmp2 = sum( mul( c.get(i), 1 - u), mul(d.get(i), u) );

                Point tmp3 = mul( sum( mul(a.get(0), (1-u) * (1-w)) , mul(b.get(0),
(1-u) * w) ) , -1);
                Point tmp4 = mul( sum( mul(a.get(n - 1), u * (1-w)) , mul(b.get(n - 1),
u * w) ) , -1);
                tmp.add( sum( sum(tmp1, tmp2) , sum(tmp3, tmp4) ));
            }

            surface.add(tmp);
        }
    }
}

package main;
```

```
public class TransformMatrix {

    public double[][] matrix;

    public TransformMatrix(double[][] matrix) {
        this.matrix = matrix;
    }

    public TransformMatrix() {
        this(new double[][]{
                {1, 0, 0, 0},
                {0, 1, 0, 0},
                {0, 0, 1, 0},
                {0, 0, 0, 1}
        });
    }

    public Point transform(Point vector) {
        // All matrix elements from 00 to 33
        double _00 = this.matrix[0][0];
        double _01 = this.matrix[0][1];
        double _02 = this.matrix[0][2];
        double _03 = this.matrix[0][3];
        double _10 = this.matrix[1][0];
        double _11 = this.matrix[1][1];
        double _12 = this.matrix[1][2];
        double _13 = this.matrix[1][3];
        double _20 = this.matrix[2][0];
        double _21 = this.matrix[2][1];
        double _22 = this.matrix[2][2];
        double _23 = this.matrix[2][3];
        double _30 = this.matrix[3][0];
        double _31 = this.matrix[3][1];
        double _32 = this.matrix[3][2];
        double _33 = this.matrix[3][3];
```

```
        double x = vector.x * _00 + vector.y * _10 + vector.z * _20 + vector.w *
_30;
        double y = vector.x * _01 + vector.y * _11 + vector.z * _21 + vector.w *
_31;
        double z = vector.x * _02 + vector.y * _12 + vector.z * _22 + vector.w *
_32;
        double h = vector.x * _03 + vector.y * _13 + vector.z * _23 + vector.w *
_33;

        return new Point(x, y, z, h);
    }

    public void mult(TransformMatrix m) {
        double[][] t = {
                {0, 0, 0, 0},
                {0, 0, 0, 0},
                {0, 0, 0, 0},
                {0, 0, 0, 0}
        };
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                for (int k = 0; k < 4; k++) {
                    t[i][j] += this.matrix[i][k] * m.getMatrix()[j][k];
                }
            }
        }
        this.matrix = t;
    }

    public static TransformMatrix rotationY(double angle){
        double rad = (angle%360)*Math.PI/180;
        double COS = Math.cos(rad);
        double SIN = Math.sin(rad);
        double[][] t = new double[][]{
                {COS,  0, -SIN, 0},
                {0,    1,    0, 0},
                {SIN,  0,  COS, 0},
```

```
            {0,    0,    0, 1}};
    return new TransformMatrix(t);
}

public static TransformMatrix rotationX(double angle){
    double rad = (angle%360)*Math.PI/180;
    double COS = Math.cos(rad);
    double SIN = Math.sin(rad);
    double[][] t = new double[][]{
            {1,    0,    0, 0},
            {0,  COS,  SIN, 0},
            {0, -SIN,  COS, 0},
            {0,    0,    0, 1}};
    return new TransformMatrix(t);
}

public static TransformMatrix screen(double width, double height) {
    final double A = (width - 1.0) / 2.0;
    final double B = (height - 1.0) / 2.0;
    double[][] t = new double[][]{
            {A, 0, 0, A},
            {0, -B, 0, B},
            {0, 0, 1, 0},
            {0, 0, 0, 1}};
    return new TransformMatrix(t);
}

public static TransformMatrix view(double x, double y, double z) {
    double[][] t = new double[][]{
            {1, 0, 0, 0},
            {0, 1, 0, 0},
            {0, 0, 1, 0},
            {x, y, z, 1}};
    return new TransformMatrix(t);
}
```

```
    public double[][] getMatrix() {
        return matrix;
    }
}
```

Выводы

Выполнив курсовой проект, я научилась реализовывать поверхность Кунса и кардинальные сплайны. Выполнять работу было в меру сложно и довольно-таки интересно, ведь эти поверхности и кривые широко применяются на практике.

Основным недостатком поверхности Кунса при простоте расчетных соотношений является невозможность ее экспорта для передачи в другие системы посредством известных стандартов передачи данных.

Образование линейчатых поверхностей находит широкое применение при создании новых форм в архитектуре.

Список литературы

1. Кривая Безье [Электронный ресурс] URL:https://ru.wikipedia.org/wiki/Кривая_Безье (дата обращения: 26.12.2018)
2. Кардинальные сплайны [Электронный ресурс] URL: http://www.thefullwiki.org/Cardinal_spline (дата обращения: 26.12.2018)