**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Компьютерная графика»

**Курсовой проект**

**по курсу «Компьютерная графика»**

Студент: Кондратьев Егор Алексеевич

Группа: 80-306

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2021

1. **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.

Типы порций поверхностей

**Линейчатая поверхность**

1. **Описание программы**

Задание курсовой состояло в том, чтобы визуализировать линейчатую поверхность, образованную при помощи основных сплайнов. Программа позволяет динамически изменять параметры поверхности, доступна возможность задания параметров отрисовки поверхности: выбор сброса проецирования, повороты вокруг осей, скрытие невидимых линий, настройка освещения, задание точности отрисовки.

**Cardinal spline 3D**

Для начала поговорим о Cardinal spline 3D. Это один из видов гладких кривых, строится изначально по массиву точек pk, а также mk, что является тангенсом наклона кривой при прохождении данной точки. Для каждых двух точек из заданного массива имеем аппроксимацию:

где для k = 1, …, n - интерполяционная сетка, t ∈ [0, 1], а

или

где c ∈ [0, 1], а при c = 0 называется сплайном Катмалла-Рома.

Но в данной работе для вычисления я использовал функцию Kochanek-Bartels Spline, выглядят они примерно так:

где t - регулирует длину вектора тангенса, b - направление вектора тангенса, c - так называемый continuity, задающий остроту угла между векторами.

1. **Результаты выполнения**

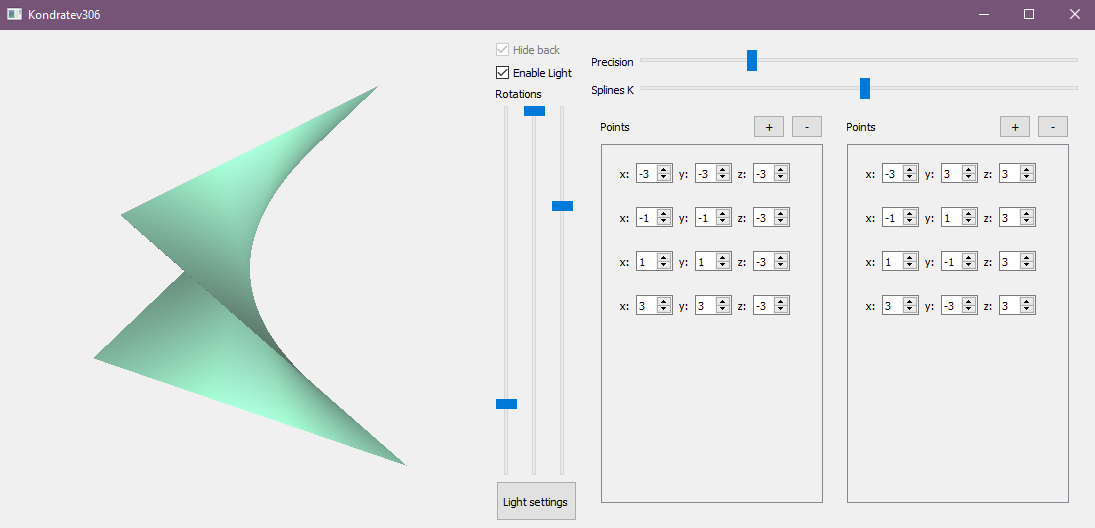
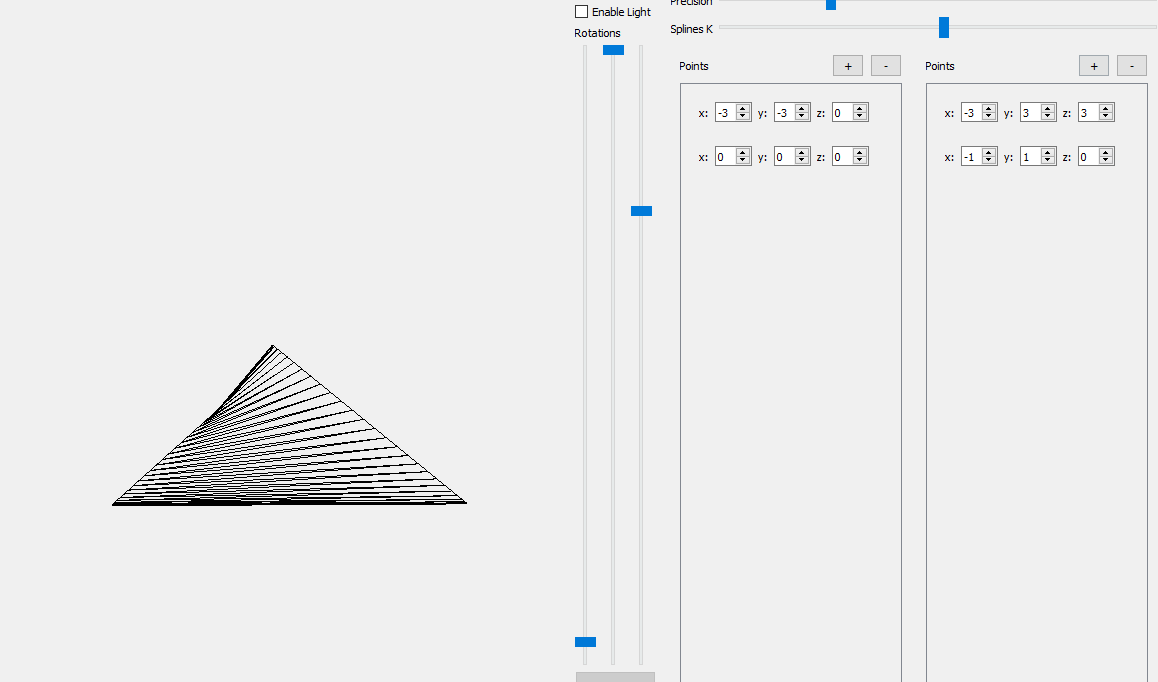


Рис 1. Билинейная поверхность

Рис 2. поверхность по 4 точкам

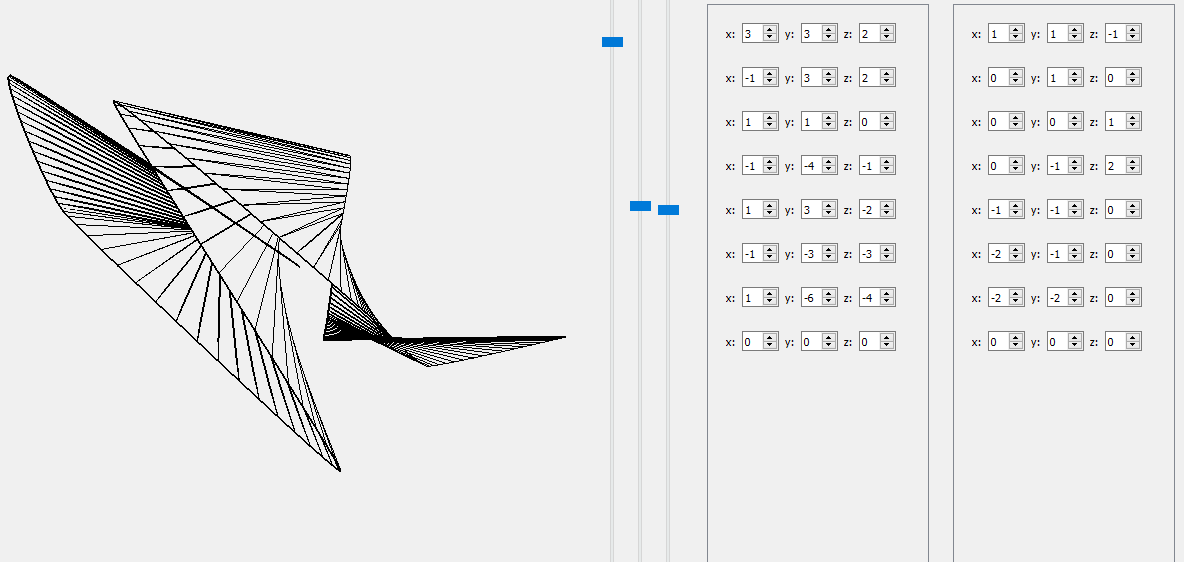


Рис 3. полет фантазии

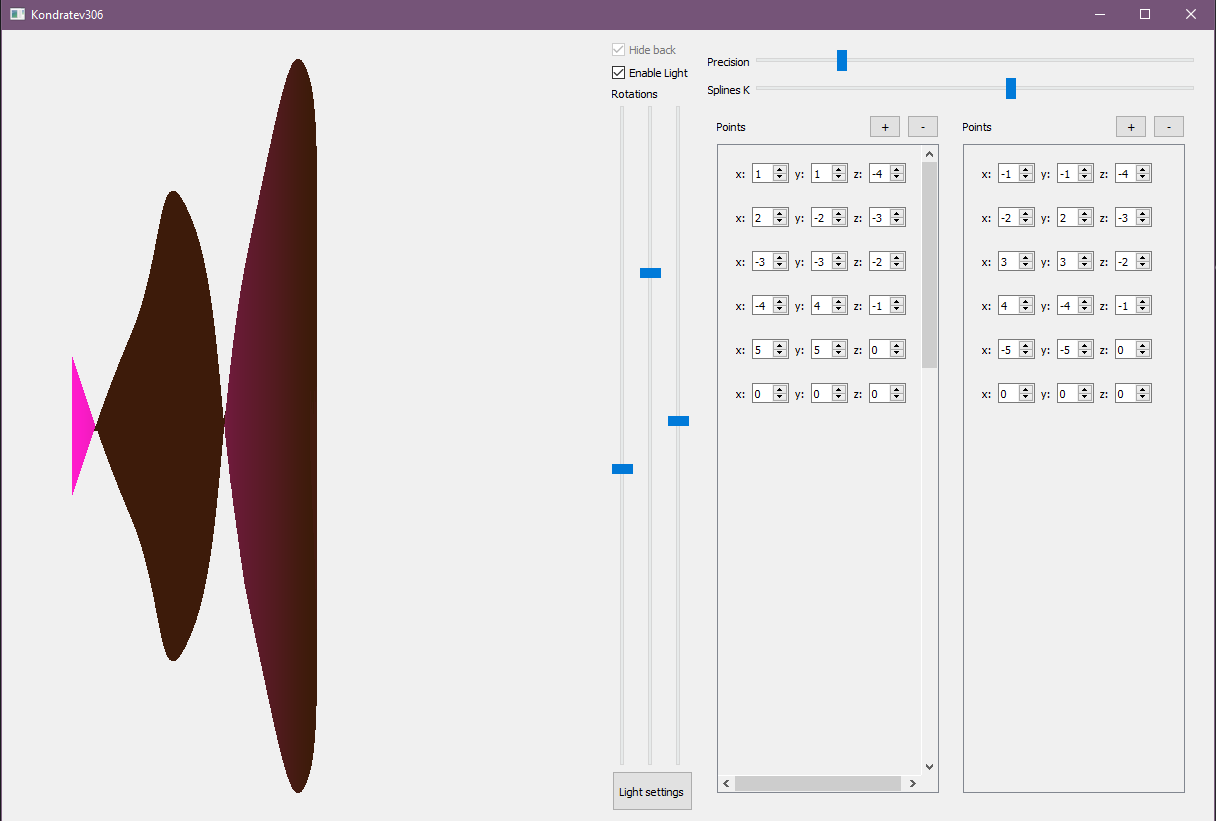


Рис 4-9.поверхность напоминающая сверло

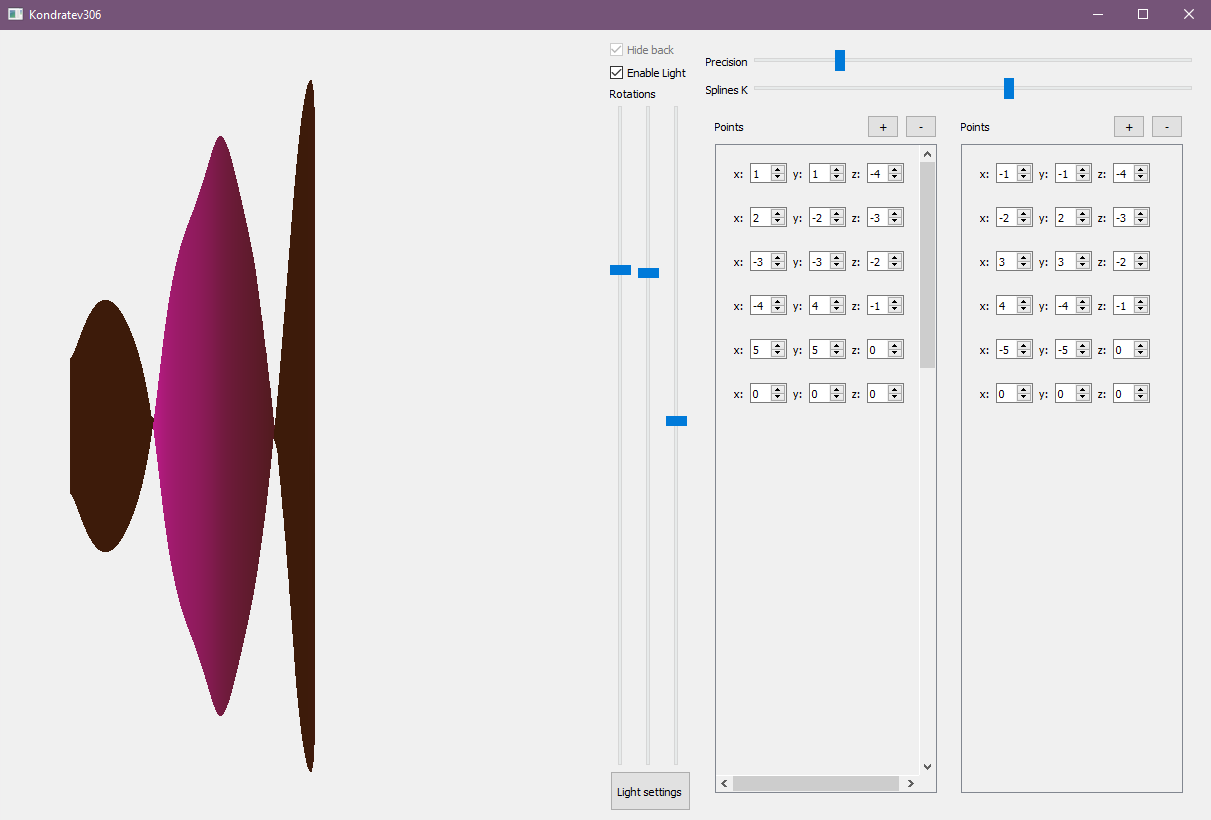


Рис 4-9.поверхность напоминающая сверло

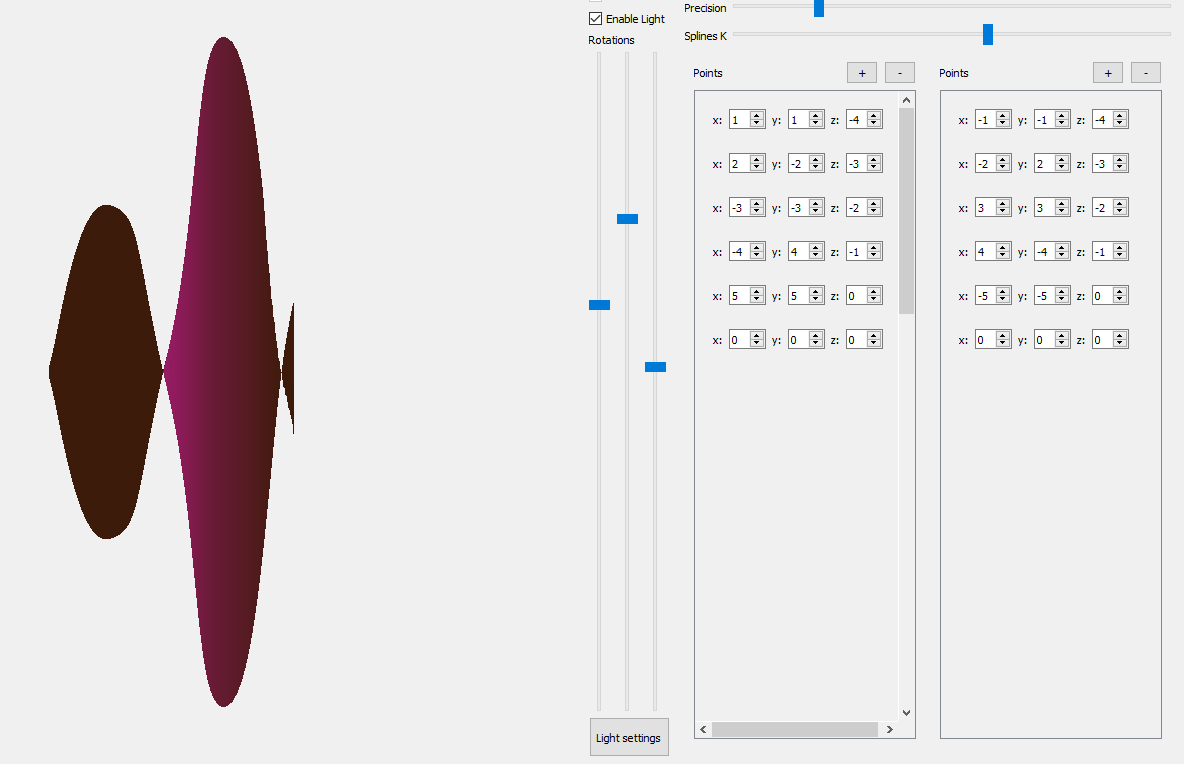


Рис 4-9.поверхность напоминающая сверло

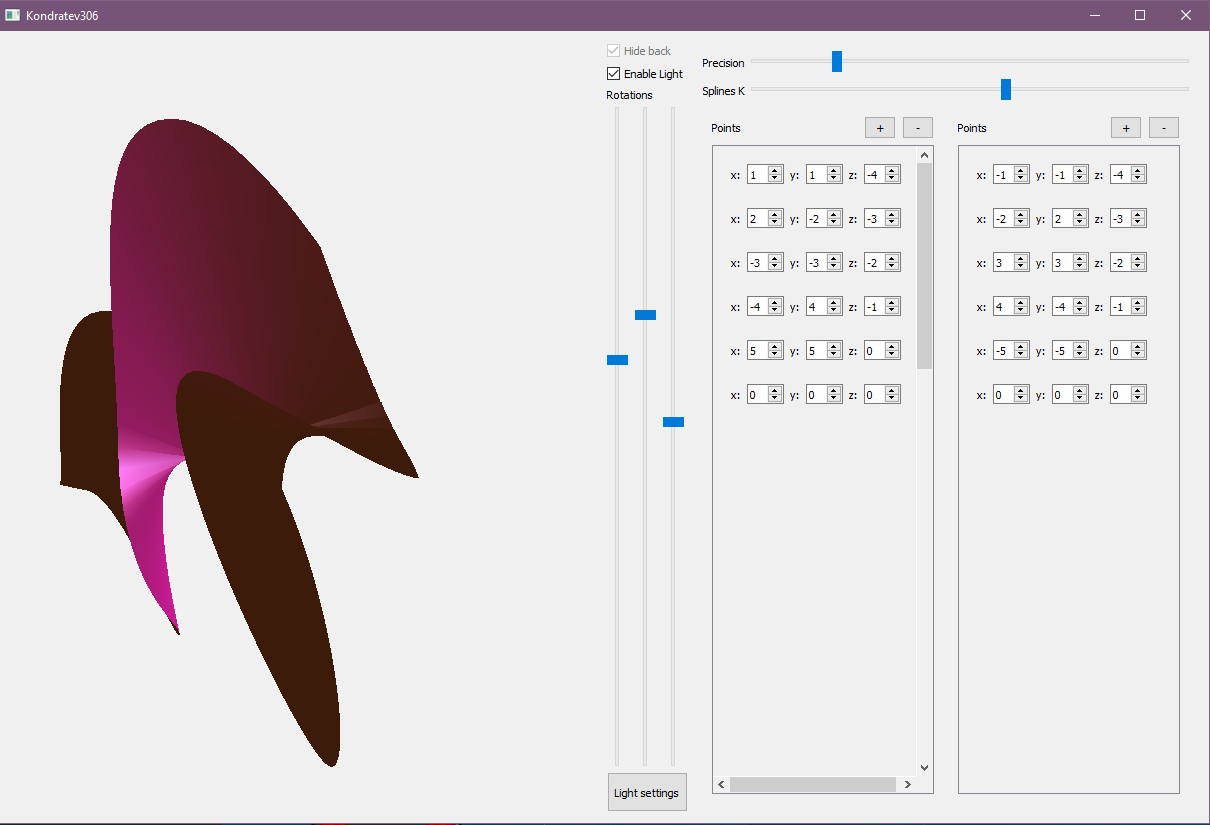


Рис 4-9.поверхность напоминающая сверло

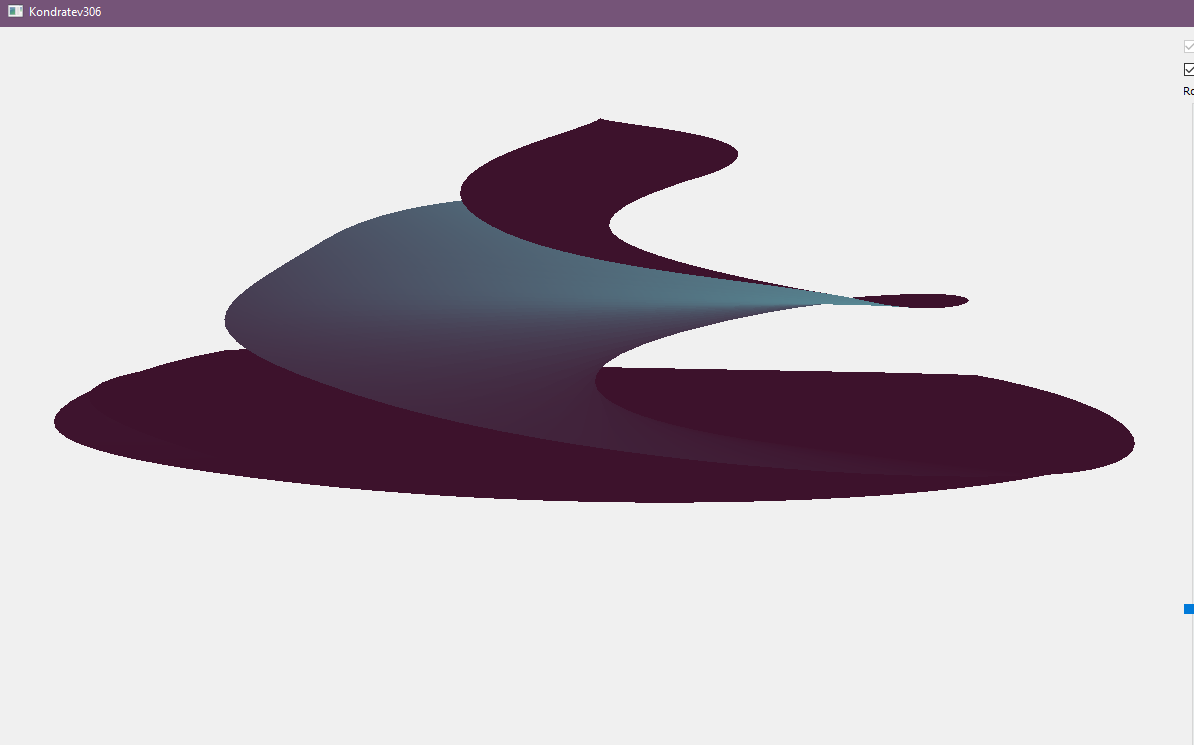


Рис 4-9.поверхность напоминающая сверло



Рис 4-9.поверхность напоминающая сверло

Комментарий: моя программа многогранна, в ней можно очень надолго засесть, чтобы сначала разобраться в работе, а потом сильно залипнуть на много часов, придумывая поверхности из головы( когда я все дописал, я засел тут часов на 5, у меня так же получились очень красивые поверхности, но, к сожалению, они не были сохранены)

1. **Листинг программы**

#include "splinesettingswidget.h"

#include "pointfieldwidget.h"

#include<QLabel>

SplineSettingsWidget::SplineSettingsWidget(CardinalSplineClass\* s): spline(s)

{

setFixedWidth(240);

plusButton->setMinimumWidth(32);

plusButton->setMaximumWidth(32);

minusButton->setMinimumWidth(32);

minusButton->setMaximumWidth(32);

vLayout = new QVBoxLayout(this);

auto hPointsLayout = new QHBoxLayout;

vLayout->addLayout(hPointsLayout);

hPointsLayout->addWidget(new QLabel("Points"));

hPointsLayout->addWidget(plusButton);

plusButton->setText("+");

hPointsLayout->addWidget(minusButton);

minusButton->setText("-");

vLayout->addWidget(saPoints);

saPoints->setWidget(pw);

QObject::connect(plusButton, SIGNAL(clicked()), this, SLOT(onClickPlusButton()));

QObject::connect(minusButton, SIGNAL(clicked()), this, SLOT(onClickMinusButton()));

}

void SplineSettingsWidget::onClickPlusButton(){

pw->addField();

update();

}

void SplineSettingsWidget::onClickMinusButton(){

pw->removeFiled(pw->count()-1);

update();

}

#ifndef SPLINESETTINGSWIDGET\_H

#define SPLINESETTINGSWIDGET\_H

#include "pointfieldwidget.h"

#include <QWidget>

#include <QHBoxLayout>

#include <QVBoxLayout>

#include <QSpinBox>

#include <QPushButton>

#include <QScrollArea>

#include "cardinalsplineclass.h"

class SplineSettingsWidget : public QWidget

{

Q\_OBJECT

public:

CardinalSplineClass\* spline = nullptr;

QVBoxLayout\* vLayout = nullptr;

QScrollArea\* saPoints = new QScrollArea;

QVBoxLayout\* vLayoutPoints = new QVBoxLayout;

QPushButton\* plusButton = new QPushButton;

QPushButton\* minusButton = new QPushButton;

PointsWidget\* pw = new PointsWidget(spline);

SplineSettingsWidget(CardinalSplineClass\* s);

public slots:

void onClickPlusButton();

void onClickMinusButton();

};

#endif // SPLINESETTINGSWIDGET\_H

#include "surfacesettingswidget.h"

#include<QLabel>

SurfaceSettingsWidget::SurfaceSettingsWidget(RuledSurfaceByCardinalSpline\* s) :

vLayout(this),

sbK(Qt::Horizontal),

sbPrecision(Qt::Horizontal),

fLineWidget(&s->fLine),

sLineWidget(&s->sLine)

{

sbPrecision.setValue(s->getPrecision());

sbK.setValue(s->fLine.getK());

sbK.setRange(-100, 100);

vLayout.addLayout(&hLayoutPrecision);

hLayoutPrecision.setAlignment(Qt::AlignLeft);

hLayoutPrecision.addWidget(new QLabel("Precision"));

hLayoutPrecision.addWidget(&sbPrecision);

sbPrecision.setSingleStep(1);

sbPrecision.setRange(0,100);

vLayout.addLayout(&hLayoutK);

hLayoutK.setAlignment(Qt::AlignLeft);

hLayoutK.addWidget(new QLabel("Splines K"));

hLayoutK.addWidget(&sbK);

vLayout.addLayout(&hLayout);

hLayout.addWidget(&fLineWidget);

hLayout.addWidget(&sLineWidget);

QObject::connect(&sbPrecision, SIGNAL(valueChanged(int)), s, SLOT(recievePrecision(int)));

QObject::connect(&sbK, SIGNAL(valueChanged(int)), &s->sLine, SLOT(acceptNewK(int)));

QObject::connect(&sbK, SIGNAL(valueChanged(int)), &s->fLine, SLOT(acceptNewK(int)));

}

#ifndef SURFACESETTINGSWIDGET\_H

#define SURFACESETTINGSWIDGET\_H

#include"splinesettingswidget.h"

#include<QSpinBox>

#include<QHBoxLayout>

#include<QVBoxLayout>

#include<QWidget>

#include<QSlider>

class SurfaceSettingsWidget : public QWidget

{

Q\_OBJECT

public:

RuledSurfaceByCardinalSpline\* surf;

QVBoxLayout vLayout;

QHBoxLayout hLayout;

QHBoxLayout hLayoutPrecision;

QHBoxLayout hLayoutK;

QSlider sbK;

QSlider sbPrecision;

SplineSettingsWidget fLineWidget;

SplineSettingsWidget sLineWidget;

SurfaceSettingsWidget(RuledSurfaceByCardinalSpline\* s);

};

#endif // SURFACESETTINGSWIDGET\_H

#include "viewwidget.h"

#include "glcuboid.h"

#include <QLabel>

ViewWidget::ViewWidget() :

ssw(&surface)

{

glw->Engine.addFigure(&surface);

hLayout->addWidget(glw,1);

hLayout->addLayout(vLayout);

vLayout->addWidget(isHidingBack);

vLayout->addWidget(isLigthing);

vLayout->addWidget(new QLabel("Rotations"));

vLayout->addLayout(hLayoutRotations);

hLayoutRotations->addWidget(sliderRotX);

hLayoutRotations->addWidget(sliderRotY);

hLayoutRotations->addWidget(sliderRotZ);

vLayout->addWidget(ligthSettings);

hLayout->addWidget(&ssw);

isHidingBack->setText("Hide back");

isLigthing->setText("Enable Light");

ligthSettings->setText("Light settings");

sliderRotX->setRange(0,MAX\_VALUE\_SLIDER);

sliderRotY->setRange(0,MAX\_VALUE\_SLIDER);

sliderRotZ->setRange(0,MAX\_VALUE\_SLIDER);

sliderRotX->setSingleStep(1);

sliderRotY->setSingleStep(1);

sliderRotZ->setSingleStep(1);

sliderRotX->setInvertedAppearance(true);

sliderRotY->setInvertedAppearance(true);

sliderRotZ->setInvertedAppearance(true);

ligthSettings->setEnabled(false);

ligthSettings->setFixedHeight(40);

QObject::connect(sliderRotX, SIGNAL(valueChanged(int)),

this, SLOT(configChanged(int)));

QObject::connect(sliderRotY, SIGNAL(valueChanged(int)),

this, SLOT(configChanged(int)));

QObject::connect(sliderRotZ, SIGNAL(valueChanged(int)),

this, SLOT(configChanged(int)));

QObject::connect(isHidingBack, SIGNAL(stateChanged(int)),

this, SLOT(configChanged(int)));

QObject::connect(isLigthing, SIGNAL(stateChanged(int)),

this, SLOT(configChanged(int)));

QObject::connect(ligthSettings, SIGNAL(clicked()),

this, SLOT(showLightSettings()));

QObject::connect(lsw->sliderPositionX, SIGNAL(valueChanged(int)),

this, SLOT(configChanged(int)));

QObject::connect(lsw->sliderPositionY, SIGNAL(valueChanged(int)),

this, SLOT(configChanged(int)));

QObject::connect(lsw->sliderPositionZ, SIGNAL(valueChanged(int)),

this, SLOT(configChanged(int)));

QObject::connect(lsw, SIGNAL(lightColorChanged()),

this, SLOT(configChanged()));

QObject::connect(&ssw.sbPrecision, SIGNAL(valueChanged(int)), &surface, SLOT(recievePrecision(int)));

QObject::connect(&surface, SIGNAL(surfaceChanged()), this, SLOT(configChanged()));

}

float f1(int v){

return (float)v\*360/MAX\_VALUE\_SLIDER;

}

float f2(int v){

return (float)v;

}

QVector4D f3(float r ,float angleX, float angleZ){

QVector4D res;

res.setX(r \* sinf(angleZ) \* cosf(angleX));

res.setY(r \* sinf(angleZ) \* sinf(angleX));

res.setZ(r \* cosf(angleZ));

res.setW(1.0);

return res;

}

void ViewWidget::configChanged(){

glw->Engine.setRotation(f1(sliderRotX->value()),

f1(sliderRotY->value()),

f1(sliderRotZ->value()));

glw->Engine.setHidingBack(isHidingBack->isChecked());

//isIzomteric->isChecked() ? glw->Engine.setIzomtericProjection() : glw->Engine.setOrthoProjection();

ligthSettings->setEnabled(isLigthing->isChecked());

if(isLigthing->isChecked()){

surface.setMode(MATERIAL);

glw->Engine.setHidingBack(true);

isHidingBack->setEnabled(false);

isHidingBack->setChecked(true);

glw->Engine.setLightEnabled(true);

} else{

surface.setMode(FRAME);

glw->Engine.setLightEnabled(false);

isHidingBack->setEnabled(true);

}

glw->Engine.light.direction = f3((float)lsw->sliderPositionX->value()/100, (float)lsw->sliderPositionY->value()/100, (float)lsw->sliderPositionZ->value()/100);

glw->Engine.light.ambient = lsw->ambient;

glw->Engine.light.specular = lsw->specular;

glw->Engine.light.diffuse = lsw->diffuse;

glw->updateGL();

}

void ViewWidget::configChanged(int){

configChanged();

}

ViewWidget::~ViewWidget(){

}

void ViewWidget::showLightSettings(){

lsw->show();

}

#ifndef VIEWIDGET\_H

#define VIEWIDGET\_H

#include <QWidget>

#include <QHBoxLayout>

#include <QVBoxLayout>

#include <QCheckBox>

#include <QSlider>

#include <QPushButton>

#include "glwidget.h"

#include "glcuboid.h"

#include "glsphere.h"

#include "lightsettingswidget.h"

#include "ruledsurfacebycardinalspline.h"

#include "surfacesettingswidget.h"

const int MAX\_VALUE\_SLIDER = 1000;

class ViewWidget : public QWidget

{

Q\_OBJECT

private:

RuledSurfaceByCardinalSpline surface;

QHBoxLayout\* hLayout = new QHBoxLayout(this);

QVBoxLayout\* vLayout = new QVBoxLayout();

QHBoxLayout\* hLayoutRotations = new QHBoxLayout();

SurfaceSettingsWidget ssw;

public slots:

void configChanged(int);

void configChanged();

void showLightSettings();

public:

LightSettingsWidget\* lsw = new LightSettingsWidget();

ViewWidget();

~ViewWidget();

GlWidget\* glw = new GlWidget;

QCheckBox\* isHidingBack = new QCheckBox();

QCheckBox\* isLigthing = new QCheckBox();

QSlider\* sliderRotX = new QSlider(Qt::Vertical);

QSlider\* sliderRotY = new QSlider(Qt::Vertical);

QSlider\* sliderRotZ = new QSlider(Qt::Vertical);

QPushButton\* ligthSettings = new QPushButton();

};

#endif // VIEWIDGET\_H

**Выводы**

Выполнив курсовой проект, я научился реализовывать поверхность 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 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.

**ЛИТЕРАТУРА**

1. QT documentation [Электронный ресурс]. URL: <https://doc.qt.io/>

(дата обращения: 28.11.2021).

1. Голованов Н. Н. Геометрическое моделирование.—М.: Издательство Физико-математической литературы, 2002

[Электронный ресурс]. URL: <https://scask.ru/q_book_g_mod.php>

(дата обращения: 28.11.2021).

1. Кривые и поверхности на экране компьютера. Руководство по сплайнам для пользователя. (Шикин Е.В., Плис А.И.)

[Электронный ресурс]. URL: <https://scask.ru/g_book_cpsp.php>

(дата обращения: 28.11.2021).