МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет математики и компьютерных наук

Кафедра математических и компьютерных методов

Доклад

по дисциплине

ОСНОВЫ КОМПЬЮТЕРНЫХ НАУК

Тема: «Реализация программного обеспечения для графической визуализации простейших графиков»

Работу выполнил
студент 1 курса
группы 13А
Акимов Вадим

Краснодар 2019

Введение

Графическая визуализация графиков, математических и физических процессов является интересной и быстро развивающееся отраслью не только моделирования, но и программирования. Одним из таких ответвлений является построение графиков. Для решения многих теоретических и прикладных задач очень полезно иметь наглядное представление о том, как ведет себя та или иная функция. Эта информация поможет сберечь не только много сил и нервов, но и возможно укажет на ваши ошибки.

Программное обеспечение „GraphViewer“

При реализации своего ПО я использовал графическую библиотеку OpenGL. Данная библиотека обладает огромным спектром возможностей, в том числе удобной отрисовки 2D и 3D объектов.

Архитектура программы основана на ООП, и включает в себя все 4 столпа ООП: наследование, полиморфизм, инкапсуляция и абстракция. Программный код реализовывался на языке программирования C++11. Выбор среды разработки пал на Qt Creator, потому что она (среда разработки) позволяет писать кроссплатформенные приложения под Unix подобные системы и Windows. Так же она позволяет легко разработать GUI (Graphic User Interface) с последующим внедрением в код.

Описание проекта

Реализована отрисовка следующих функций: sin(x), cos(x), tan(x), ln(x), 1/x. Так же мы можем самостоятельно с помощью GUI выбирать шаг отрисовки и отрезки, на которых график будет строиться. Для показа/обновления графика необходимо нажать на кнопку Draw. После этого данные из формы занесутся в переменные класса MainWindow, которые в свою очередь с помощью встроенного функционала Qt Creator signal/slot (альтернатива CallBack функций с MVS, только более безопасно) изменяет поля класса OpenglWidget, который в свою очередь наследуется от QOpenglWidget, и выводит график на экран.

Листинг программы

1. Project.pro

```
QT     += core gui opengl
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

LIBS += -lGLU
TARGET = Project
TEMPLATE = app

DEFINES += QT_DEPRECATED_WARNINGS

CONFIG += c++11

SOURCES += \
        main.cpp \
        mainwindow.cpp \
    openglwidget.cpp

HEADERS += \
        mainwindow.h \
    openglwidget.h

FORMS += \
        mainwindow.ui

qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target
```

2. main.cpp

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

3. mainwindow.h

```
#ifndef MAINWINDOW_H

#define MAINWINDOW_H

#include "openglwidget.h"

#include <QMainWindow>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void on_button_clicked();

signals:
    void signal_set_params_openglwidget(int func,
                                        double step,
                                        int interval_left_x,
                                        int interval_right_x,
                                        int interval_up_y,
                                        int interval_down_y);


private:
    Ui::MainWindow *ui;
    OpenglWidget *simulation;
};

#endif // MAINWINDOW_H
```

4. mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    this->setWindowTitle("GraphViewer");
    simulation = new OpenglWidget;
    ui->layout->addWidget(simulation);
    ui->comboBox->addItem("sin x");
    ui->comboBox->addItem("cos x");
    ui->comboBox->addItem("tan x");
    ui->comboBox->addItem("ln x");
    ui->comboBox->addItem("1 / x");
    ui->comboBox->currentIndex();

    connect(this, &MainWindow::signal_set_params_openglwidget,
            simulation, &OpenglWidget::set_painter);
}

MainWindow::~MainWindow() {
    delete ui;
}

void MainWindow::on_button_clicked() {
    int func             = ui->comboBox->currentIndex();
    double step          = ui->line_step->text().toDouble();
    int interval_left_x  = ui->line_interval_left_x-
>text().toInt();
    int interval_right_x = ui->line_interval_right_x-
>text().toInt();
    int interval_up_y    = ui->line_interval_up_y-
>text().toInt();
    int interval_down_y  = ui->line_interval_down_y-
>text().toInt();
    emit signal_set_params_openglwidget(func + 1,
                                        step,
                                        interval_left_x,
                                        interval_right_x,
                                        interval_up_y,
                                        interval_down_y);
}
```

5. openglwindget.h

```
#ifndef OPENGLWIDGET_H
#define OPENGLWIDGET_H

#include <QOpenGLWidget>
#include <GL/glu.h>
#include <QtMath>
#include <QDebug>

class OpenglWidget : public QOpenGLWidget
{
    Q_OBJECT
public:
    OpenglWidget();

    void initializeGL() override;
    void paintGL() override;

public slots:
    void set_painter(int func,
                     double step,
                     int interval_left_x,
                     int interval_right_x,
                     int interval_up_y,
                     int interval_down_y);

private:
    double cur_step;
    int status;
    int interval_left_x;
    int interval_right_x;
    int interval_up_y;
    int interval_down_y;

    void sin_x();
    void cos_x();
    void tan_x();
    void ln_x();
    void x_negative_1();

    void draw_grid();
};

#endif // OPENGLWIDGET_H
```

6. openglwindget.cpp

```
#include "openglwidget.h"
OpenglWidget::OpenglWidget() {
    cur_step = 0.01;
    status = 0;
    interval_left_x = -2;
    interval_right_x = 2;
    interval_down_y = -2;
    interval_up_y = 2;
}

void OpenglWidget::initializeGL() {
    glClearColor(1, 1, 1, 1);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(interval_left_x, interval_right_x,
interval_down_y, interval_up_y);
    glEnable(GL_SMOOTH);
}

void OpenglWidget::paintGL() {
    switch (status) {
    case 0:
        draw_grid();
    break;
    case 1:
        sin_x();
    break;
    case 2:
        cos_x();
    break;
    case 3:
        tan_x();
    break;
    case 4:
        ln_x();
    break;
    case 5:
        x_negative_1();
    break;
    }
}

void OpenglWidget::draw_grid() {
    glClearColor(1, 1, 1, 1);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
```

```
    gluOrtho2D(interval_left_x, interval_right_x,
interval_down_y, interval_up_y);
    glLineWidth(2);
    glBegin(GL_LINES);
        glColor3d(0, 0, 0);
        glVertex2d(interval_left_x, 0);
        glVertex2d(interval_right_x, 0);
        glVertex2d(0, interval_down_y);
        glVertex2d(0, interval_up_y);
    glEnd();
    glPointSize(4);
    glBegin(GL_POINTS);
        glVertex2d(1, 0);
        glVertex2d(0, 1);
    glEnd();
}

void OpenglWidget::set_painter(int func,
                               double step,
                               int interval_left_x,
                               int interval_right_x,
                               int interval_up_y,
                               int interval_down_y) {
    this->interval_left_x  = interval_left_x;
    this->interval_right_x = interval_right_x;
    this->interval_down_y  = interval_down_y;
    this->interval_up_y    = interval_up_y;
    status = func;
    cur_step = step;
    update();
}

void OpenglWidget::sin_x() {
    draw_grid();
    glLineWidth(2.5);
    glBegin(GL_LINES);
        glColor3d(0, 1, 0);
        for (double x = 0; x < interval_right_x; x += cur_step)
{
            glVertex2d(x, qSin(x));
            glVertex2d(x + cur_step, qSin(x + cur_step));
        }
        for (double x = 0; x > interval_left_x; x -= cur_step) {
            glVertex2d(x, qSin(x));
            glVertex2d(x - cur_step, qSin(x - cur_step));
        }
    glEnd();
    update();
}
```

```
void OpenglWidget::cos_x() {
    draw_grid();
    glLineWidth(2.5);
    glBegin(GL_LINES);
        glColor3d(0, 1, 0);
        for (double x = 0; x < interval_right_x; x += cur_step)
{
            glVertex2d(x, qCos(x));
            glVertex2d(x + cur_step, qCos(x + cur_step));
        }
        for (double x = 0; x > interval_left_x; x -= cur_step) {
            glVertex2d(x, qCos(x));
            glVertex2d(x - cur_step, qCos(x - cur_step));
        }
    glEnd();
    update();
}

void OpenglWidget::tan_x() {
    draw_grid();
    glLineWidth(2.5);
    glBegin(GL_LINES);
        glColor3d(0, 1, 0);
        for (double x = 0; x < interval_right_x; x += cur_step)
{
            glVertex2d(x, qTan(x));
            glVertex2d(x + cur_step, qTan(x + cur_step));
        }
        for (double x = 0; x > interval_left_x; x -= cur_step) {
            glVertex2d(x, qTan(x));
            glVertex2d(x - cur_step, qTan(x - cur_step));
        }
    glEnd();
    update();
}

void OpenglWidget::ln_x() {
    draw_grid();
    glLineWidth(2.5);
    glBegin(GL_LINES);
        glColor3d(0, 1, 0);
        for (double x = 0; x < interval_right_x; x += cur_step)
{
            glVertex2d(x, qLn(x));
            glVertex2d(x + cur_step, qLn(x + cur_step));
        }
        for (double x = 0; x > interval_left_x; x -= cur_step) {
            glVertex2d(x, qLn(x));
            glVertex2d(x - cur_step, qLn(x - cur_step));
```

```
        }
    glEnd();
    update();
}

void OpenglWidget::x_negative_1() {
    draw_grid();
    glLineWidth(2.5);
    glBegin(GL_LINES);
        glColor3d(0, 1, 0);
        for (double x = cur_step; x < interval_right_x; x +=
cur_step) {
            glVertex2d(x, 1 / (x));
            glVertex2d(x + cur_step, 1 / (x + cur_step));
        }
        for (double x = -cur_step; x > interval_left_x; x -=
cur_step) {
            glVertex2d(x, 1 / (x));
            glVertex2d(x - cur_step, 1 / (x - cur_step));
        }
    glEnd();
    update();
}
```

Для более подробного ознакомления с исходным кодом ПО перейдите по ссылке на мой git: https://github.com/vadim8960/semester-work-1/tree/master/Project