Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем

Кафедра системного программирования

Черников Антон Александрович

# Создание построителя и анализатора кривых и поверхностей

Отчёт по учебной практике

Научный руководитель:

к. т. н., доцент Т. А. Брыксин

Санкт-Петербург

2020

# Оглавление

# Введение

Обучение играет ключевую роль в жизни человека, так как оно позволяет людям добиваться невероятных высот, строить себе карьеру, быть востребованными и профессиональными специалистами в своей сфере деятельности.

Наглядность – одна из важных, если не самая важная часть обучения. Люди воспринимают информацию, переданную через изображения гораздо лучше, чем текстовую или словесную информацию. Поэтому, научиться чему-либо посредством образов куда проще, чем читая или разбираясь в этом без них. Очень известная цитата говорит: «одна картинка стоит тысячи слов».

Студентам, в частности, тоже зачастую очень тяжело даётся информация. Например, при понимании дифференциальной геометрии уходит много времени на её осознание и требуется множество опыта в решении практических задач для её понимания. Процесс обучения ускорился бы в разы, если бы студенты во время обучения своими руками задавали нужные им кривые и поверхности и изучали их свойства, получая от среды в понятном и удобном виде требуемые характеристики.

С целью ускорить и упростить обучение студентов дифференциальной геометрии, было принято решение разработать такое приложение.

# Постановка задачи

Цель работы: создание построителя и анализатора кривых и поверхностей. Данная работа является групповой, вовлекающей в себя двух участников. Для достижения поставленной цели от автора работы требуется:

- реализовать построение моделей поверхности, вектора и кривой;
- создать освещение;
- реализовать подсчёт кривизны, кручения, базиса Френе для кривой и подсчёт главных кривизн и направлений, касательной плоскости и соприкасающегося параболоида для поверхности.

# 1. Обзор существующих решений

## 1.1. Microsoft Mathematics

Microsoft Mathematics[1] – бесплатная обучающая программа, позволяющая пользователям решать математические и научные задачи. В первую очередь она предназначена для студентов как инструмент обучения. Может быть полезной при решении небольших задач по алгебре, геометрии, анализу, статистике, физике и другим дисциплинам, требующим математические расчеты.

Присутствует поддержка построения поверхностей и кривых, заданных явно, через параметр или через полярные координаты.

Также приложение позволяет решать уравнения, находить данные о треугольнике, а также имеет средство преобразования единиц измерений. Решение сопровождается пошаговым объяснением.

## 1.2. Advanced grapher

Advanced grapher[2] – программа для построения графиков и их анализа.

Позволяет строить двумерные и трёхмерные графики функций, которые могут быть заданы параметрически, неявно, в полярных координатах, через таблицы или через неравенства.

Также в приложении есть возможность находить точки экстремума, приближённые значения корней функции, выполнять дифференцирование и численное интегрирование.

## 1.3. Rhinoceros

Rhinoceros[3] является программным обеспечением для трехмерного моделирования.

В этом приложении реализовано высчитывание длины, угла между кривыми, отображение базиса Френе для кривых и высчитывание площади, главных кривизн и Гауссовой и средней кривизн для поверхностей.

## 1.4. FairCurveModeler

Основная цель этого приложения[4] заключается в том, чтобы строить по заданной кривой более плавную, устроенную проще и обладающую более

[1] https://www.microsoft.com/ru-ru/download/details.aspx?id=15702
[2] https://www.alentum.com/agrapher
[3] https://www.rhino-3d.ru

простыми свойствами, но проходящую через те же основные точки, что и заданная. Оно умеет считать кривизну, кручение и многие другие параметры кривой, но использует их для промежуточных вычислений при выполнении основной задачи и не направлена на обучение геометрии. Это прикладная программа для проектирования физических объектов, например поверхности плуга или дорожных трасс.

## 1.5. Выводы

Авторами данного проекта был проведён анализ существующих популярных приложений, возможности которых максимально приближены к тем, что указаны в поставленных авторами проекта задачах. От планируемого приложения требовалось, чтобы оно и подходило для обучения, служив наглядным пособием, и при этом ещё использовало достаточно углубленную геометрию. Результаты анализа показали, что существующие приложения или хорошо решают только первую задачу, но при этом обучение сводится к базовой математике, такие как Microsoft Mathematics и Advanced grapher, или вторую, но не подходят для обучения, так как направлены на решение специфических задач моделирования, такие как FairCurveModeler и Rhinoceros. Таким образом, было принято решение создать приложение, отвечающее всем заявленным требованиям.

[4] http://spliner.ru

# 2. Технологии и инструменты

Проект написан на языке программирования Java, для работы с отображением графических объектов была выбрана библиотека Lightweight Java Game Library.

Также требовалась библиотека, предоставляющая возможность создавать графический пользовательский интерфейс. Среди рассмотренных была выбрана LWJGUI (light weight java gui), так как эта библиотека предоставляет больший по сравнению с другими набор элементов интерфейса.

# 3. Общая архитектура

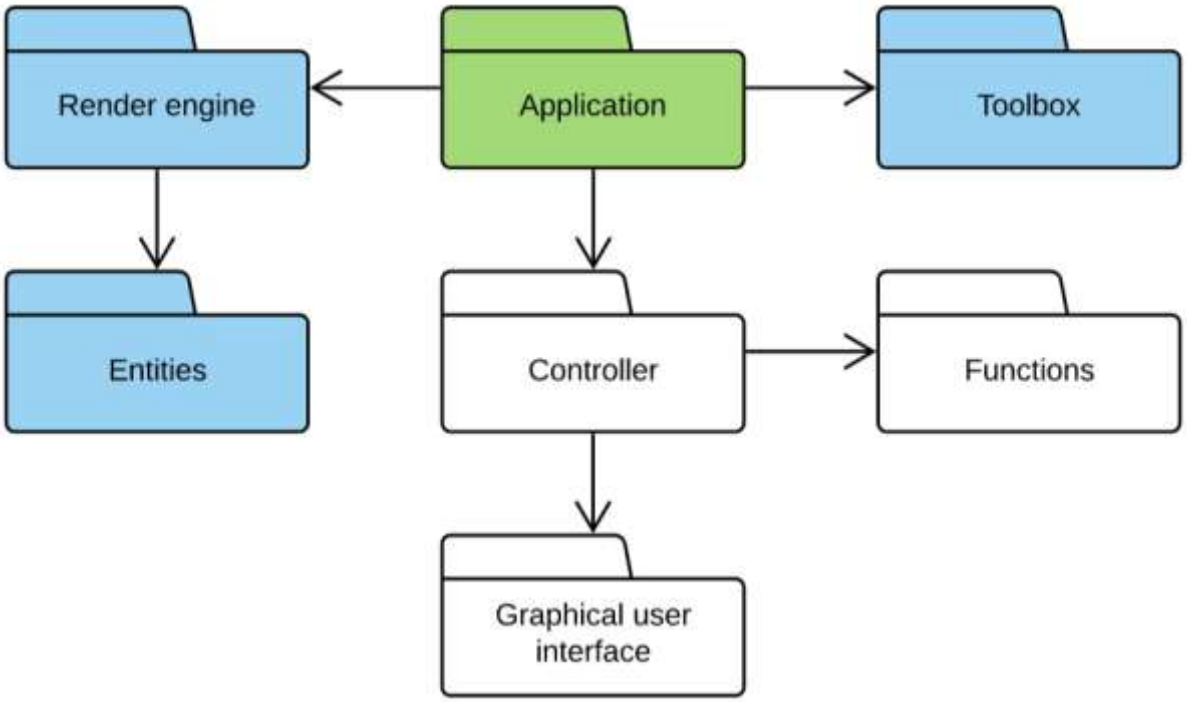


Рис. 1 Общая архитектура

На диаграмме (рис. 1) представлены пакеты классов и их взаимодействие друг с другом. Синим цветом выделены пакеты, с которыми работал автор работы, зелёным – реализованные совместно с участником проекта.

Render engine представляет собой инструменты, с помощью которых происходит отображение данных на экране.

Мощные существующие движки подключать было нецелесообразно, поэтому были использованы стандартные инструкции рендеринга в OpenGL.

Этот пакет использует объекты из пакета Entities и визуализирует их на экране.

В пакете Entities содержатся классы, представляющие плоскость, кривую, а также камеру и свет.

Toolbox содержит математические инструменты для работы с матрицами.

Controller обрабатывает нажатия клавиш мыши и кнопок клавиатуры.

Functions содержит инструменты для представления функций, составления функции по строковому представлению и проведения операций над функциями.

Пакет Graphical User Interface представляет собой набор утилит для представления элементов графического интерфейса. Данный пакет использует библиотеку LWJGUI.

# 4. Построение моделей

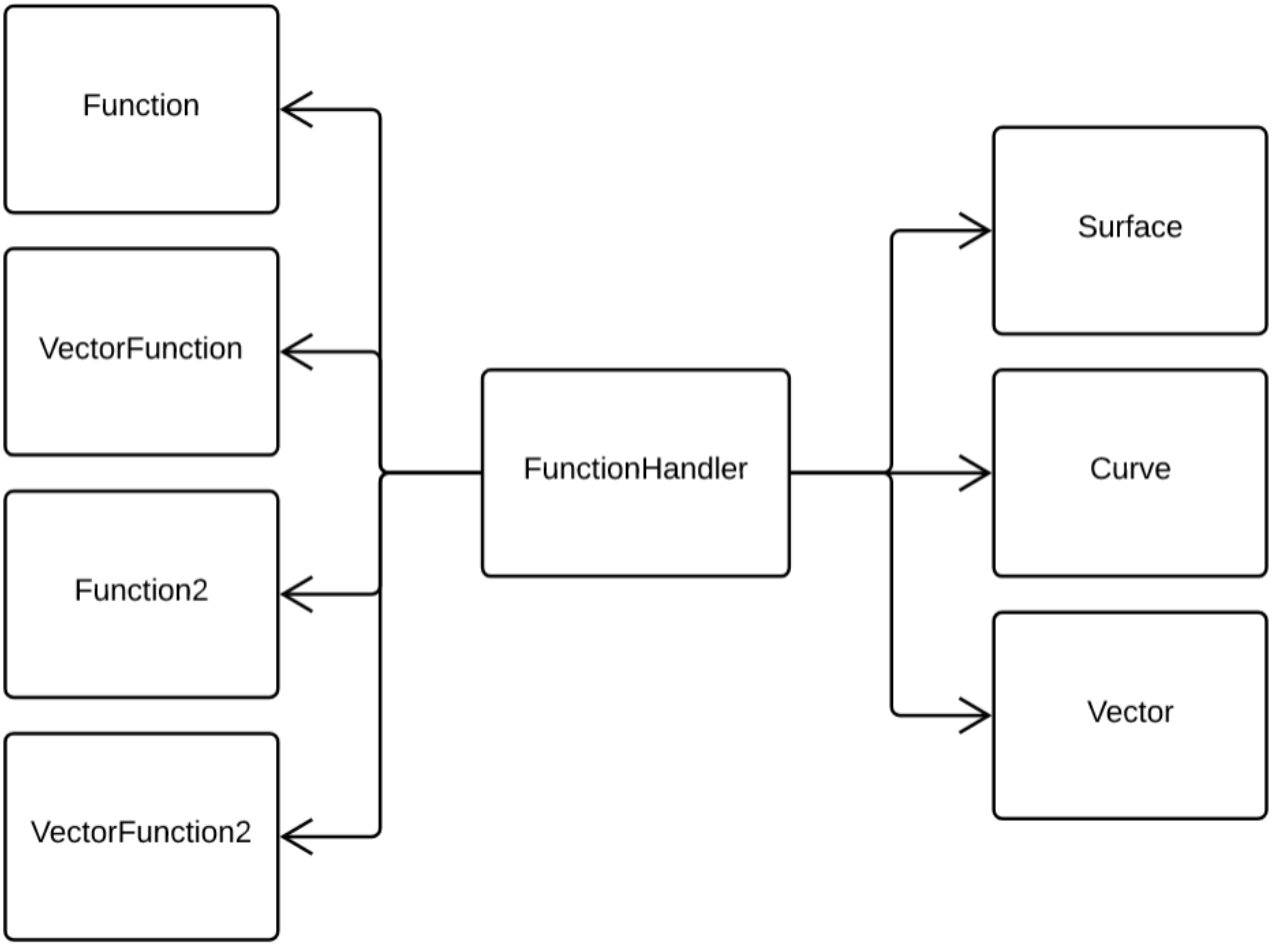


Рис. 2 Архитектура геометрических объектов

## 4.1. Кривая

Кривая определяется вектор-функцией от одного параметра. Если кривая задана явно функцией *f(x)*, то переменная *x* обозначается за переменную *t*, *y* – за *f(t)*, а *z* присваивается значение *0*. Таким образом, задание кривой явно сводится к заданию параметрически. Класс FunctionHandler содержит метод createCurve, который принимает функцию, промежуток задания параметра, шаг параметра и цвет кривой и возвращает объект класса Curve. Промежуток задания разбивается шагом на точки, потом они преобразуются в точки в пространстве xyz и соединяются линиями в ломаную, получая тем самым трёхмерную кривую.

## 4.2. Вектор

FunctionHandler имеет также метод createVector, который принимает позицию вектора, его направление и цвет и возвращает объект класса Vector. Вектор генерируется аналогично кривой, только разбивается лишь на одну линию, а её длина зависит от приближённости камеры к вектору. Это сделано так, потому что приложение должно акцентировать внимание на направлении вектора. Если бы

длина вектора была бы фиксированной, при большом отдалении камеры от него вектор переставал бы быть заметным.

## 4.3. Поверхность

Отображение поверхностей устроено немного иначе. Поверхность определяет вектор-функция от двух параметров. Аналогично кривой задание явно сводится к заданию параметрически (если $z=f(x,y)$, то $x$ принимается за $u$, $y$ – за $v$, а $z$ присваевается $f(u,v)$). В классе FunctionHandler есть также метод createSurface, принимающий на вход эту функцию, промежутки задания параметров и их шаги и цвет и возвращающий объект класса Surface. Этот метод разбивает uv плоскость на прямоугольники, далее каждый прямоугольник делится на два треугольника и полученные треугольники отображаются в xyz пространство, получая трёхмерную модель поверхности состоящую из треугольников.

# Обработка моделей

За загрузку моделей в OpenGL и отображение их на экран отвечают классы Loader и Renderer соответственно, которые реализованы в движке проекта. В них есть методы, использующие инструменты библиотеки LWJGL, которые принимают получившиеся по вышеописанным алгоритмам модели и загружают или отображают уже загруженные модели.

# 5. Освещение

Для того, чтобы был виден объём поверхности, её глубина, изгибы и выпуклости, было реализовано освещение объекта. У поверхности высчитываются вектора нормалей в заданных точках. Они нужны для создания освещения. Освещение в данном проекте было реализовано следующим образом: создаётся класс Light, который хранит в себе цвет света и точку, в которой он находится. Вершинный шейдер высчитывает вектор, направленный к свету, вектор нормали и вектор, направленный к камере и передаёт их фрагментному шейдеру. Фрагментный шейдер инвертирует вектор к свету, отражает полученный вектор относительно нормали и в результате получается вектор «отражённый свет». Далее высчитывается скалярное произведение вектора, направленного к камере и отражённого луча и в зависимости от полученного числа задаётся яркость и отражённость объекта. Чем больше произведение, тем больше яркость и отражённость.

Освещение с одним источником света позволяет увидеть структуру поверхности только с одной стороны, в то время как другая отрисовывается однотонным цветом. Поэтому было принято решение добавить возможность загружать сразу несколько источников света, тем самым решив эту проблему. Каждый источник света обрабатывается в массиве. Для того чтобы у поверхности сохранились тени, источники были наделены разной яркостью.

# 6. Функциональные возможности

## 6.1. Кривая

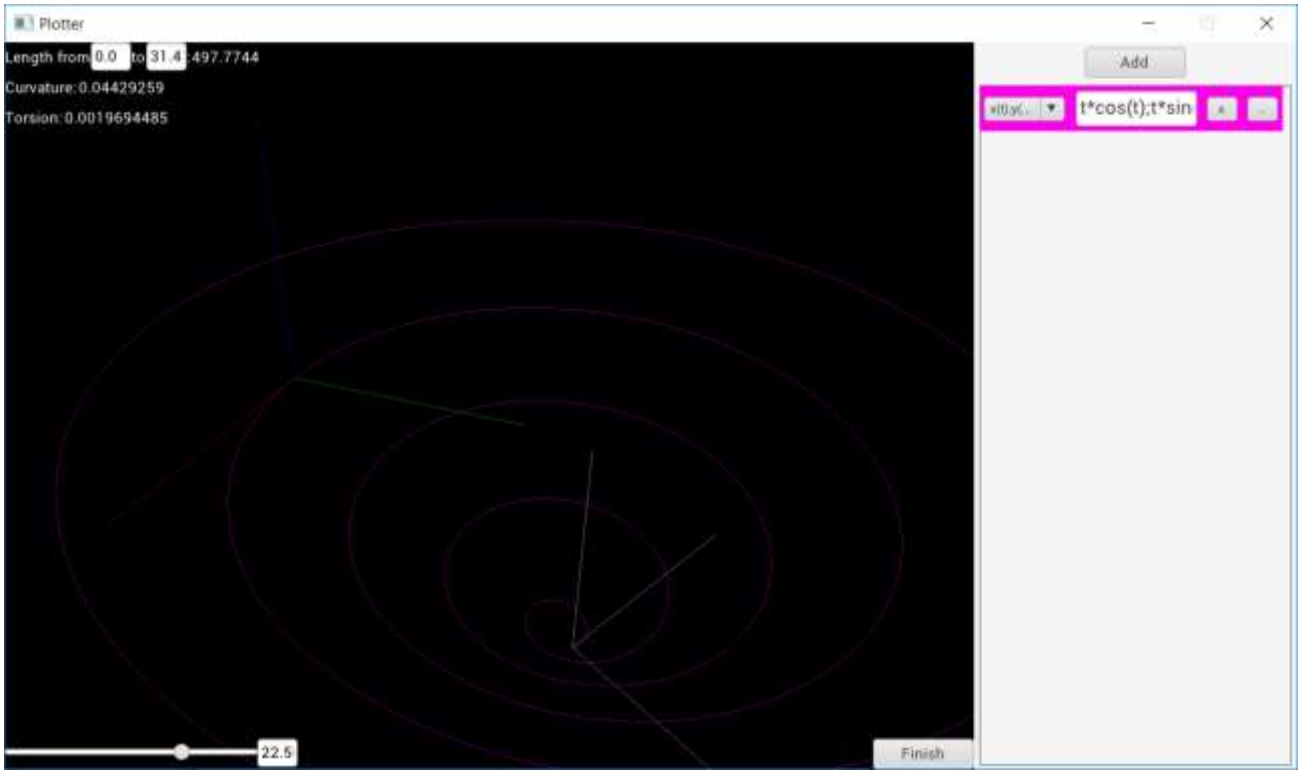


Рис. 3 Базис Френе

Одной из функциональных возможностей приложения является подсчёт кривизны, кручения и базиса Френе кривой в выбранной точке. Кривизна и кручение возвращаются методами класса Curve, которые считают их используя математические формулы. У каждой кривой есть поле, содержащее в себе список векторов, хранящий в себе базис Френе. Если от кривой требуется базис в выбранной точке, то у неё вызывается метод getFrenetFrame, считающий его по формулам и возвращающий список полученных векторов. В приложении он подсвечивается тремя векторами: красный – касательная, зелёный – нормаль, синий – бинормаль.

Для длины кривой есть тоже отдельный метод, который считает её с помощью взятия производной, модуля и интегрирования и возвращает результат.

## 6.2. Поверхность

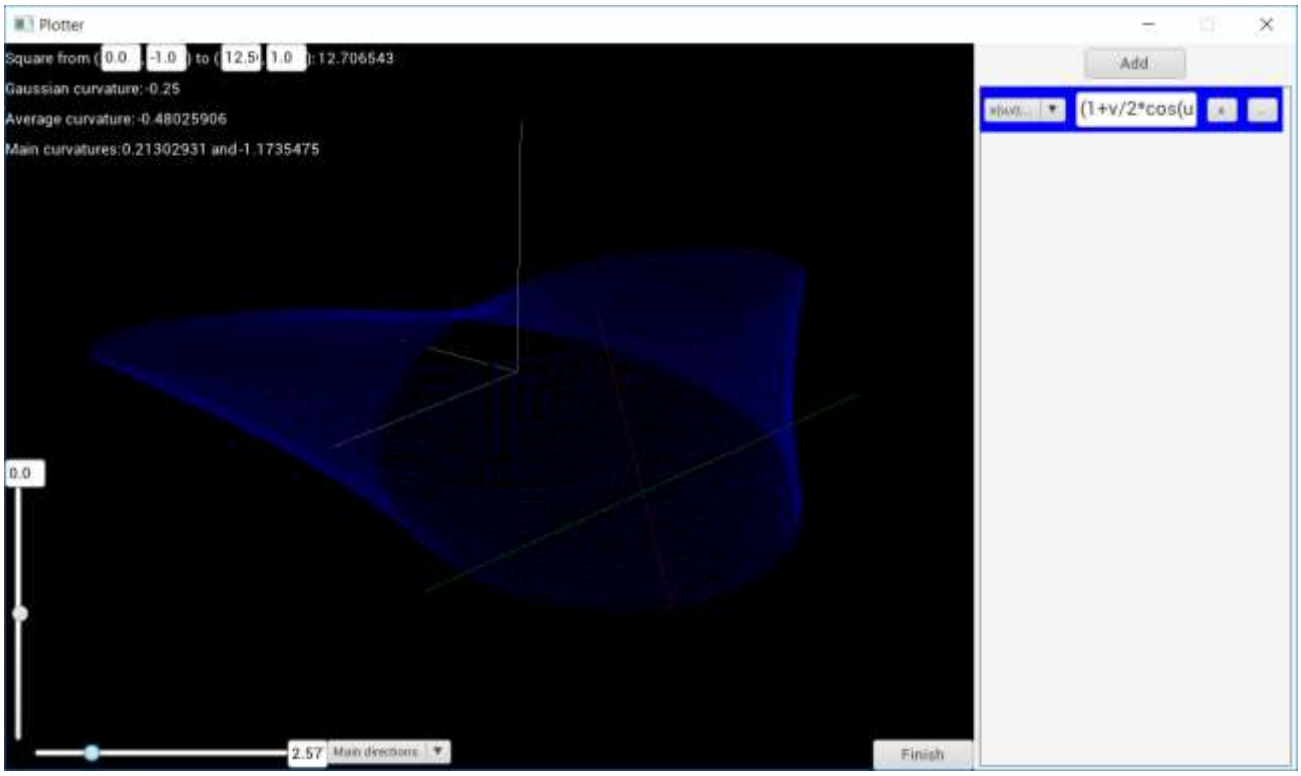


Рис. 4 Главные направления

Для поверхностей был реализован другой способ отображения. При представлении поверхности как сплошной фигуры видны её глубина и неровности, однако может быть непонятно, как она устроена в недоступных для невооружённого глаза местах. Для этого была сделана возможность представления поверхности как множества кривых. Построение происходит следующим образом: один из параметров поверхности берётся за константу и получается уравнение от одного параметра, то есть кривая. Константы для фиксированного параметра берутся из его промежутка задания и определяются его шагом. Далее проводятся аналогичные преобразования над другим параметром. В итоге у нас получается поверхность, состоящая из кривых. Оба представления просчитываются сразу во время создания поверхности, а какое из них отображать решает Renderer в зависимости от того, какой способ указал пользователь в окне.

У поверхностей есть метод getMainDirections, который аналогичен методу getFrenetFrame у кривых, только считающий главные направления. Направление с минимальной нормальной кривизной подсвечивается одним цветом, с максимальной — другим. Также у поверхностей реализован подсчёт главных кривизн (и средней и Гауссовой кривизны) методами, считающими их в точке и возвращающими результат.

Площадь поверхности высчитывается аналогично длине кривой только при подсчёте берётся двойной интеграл.

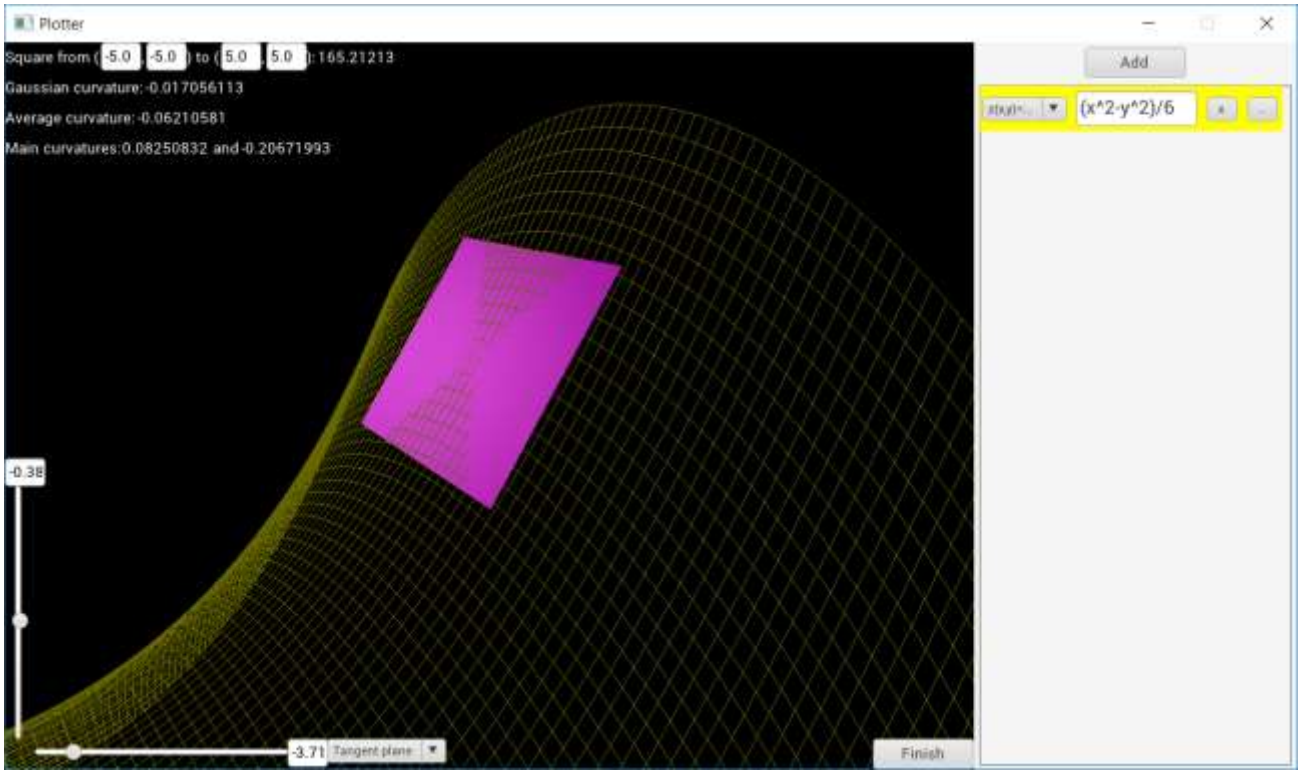


Рис. 5 Касательная плоскость

Другим инструментом приложения является решение задачи аппроксимации. Поверхность может быть аппроксимирована линейно и квадратично. Линейным приближением является касательная плоскость. Метод getTangentPlane класса Surface принимает на вход точку поверхности, заданную двумя параметрами, и возвращает плоскость (объект класса Surface). Прострoение происходит следующим образом: для начала просчитываются векторы, которые получаются дифференцированием вектор-функции по её параметрам в данной точке. Затем они ортонормируются по алгоритму ортогонализации Грама-Шмидта и по полученным векторам строится функция, описывающая касательную плоскость. Потом по этой функции строится объект класса Surface.

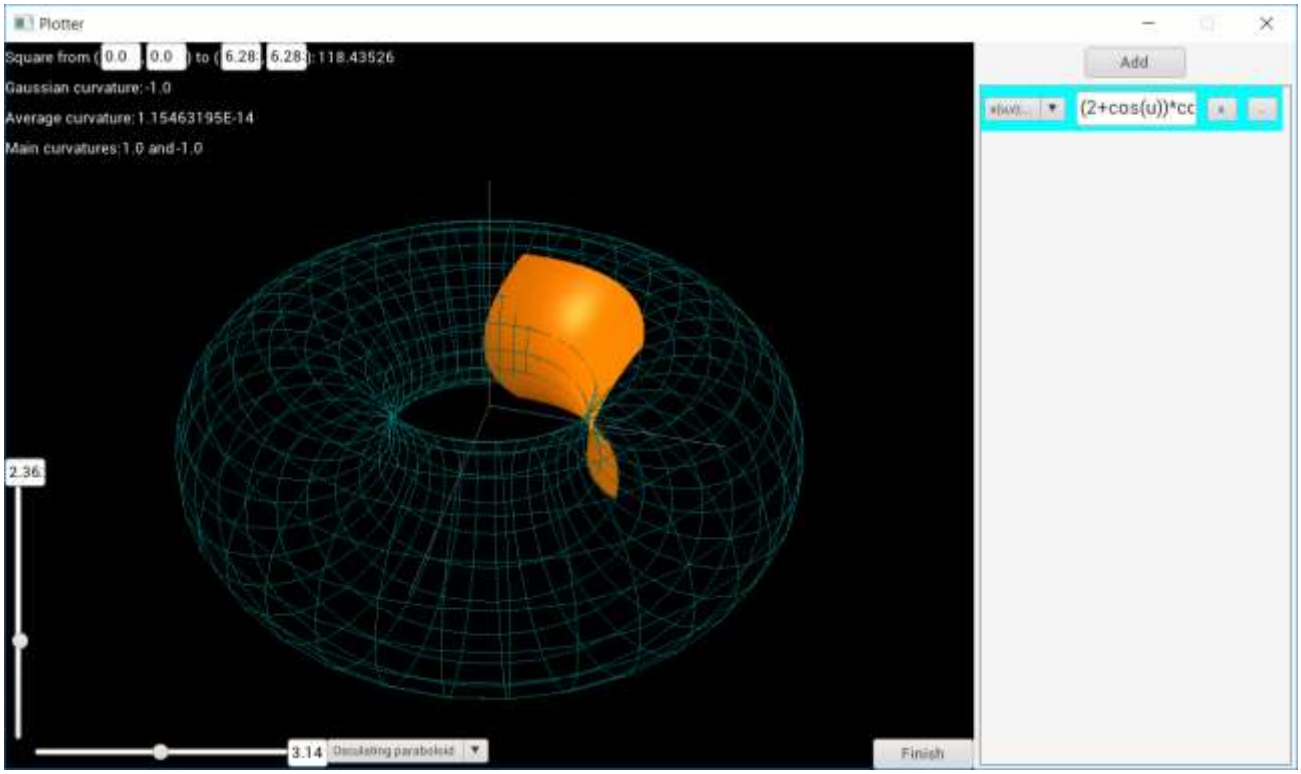


Рис. 6 Соприкасающийся параболоид

Квадратичная аппроксимация - это соприкасающийся параболоид. У класса Surface есть метод getOsculatingParaboloid, принимающий точку и возвращающий соприкасающийся параболоид в этой точке. Уравнение параболоида зависит только от главных кривизн и системы координат, в которой он построен. В качестве базиса для этой системы выбирается тройка из вектора нормали и двух перпендикулярных главных направлений в выбранной точке. Для построения параболоида существует математический алгоритм, однако функции, которые используются при его построении, имеют большой и сложный вид, а также они задают параболоид громоздким уравнением, которое трудно считать и которое занимает много памяти. Исходя из этого, теряется много времени на подсчёт значений в сложной формуле. Поэтому отображение параболоида было реализовано по-другому. Сначала он строится в стандартной координатной системе по простому уравнению, а потом перемещениями и поворотами самой модели, формулы которых имеют гораздо более простой вид, происходит переход к нужной системе координат. В результате получается соприкасающийся параболоид.

# Заключение

В ходе данной работы были достигнуты следующие результаты:

- разработано приложение, позволяющее строить и анализировать геометрические объекты;
- получен опыт работы с библиотекой LWJGL;
- получен опыт работы с проектом, имеющим научную направленность.

Приложение исправно отображает геометрические объекты и позволяет производить их анализ с помощью встроенных инструментов, поэтому можно уверенно сказать, что поставленные задачи были выполнены. Код проекта доступен на GitHub[5].

В результате переговоров с автором библиотеки LWJGUI, он пометил эту работу как проект, использующий его библиотеку, указав ссылку.

[5] https://github.com/artemgl/HarryPlotter

# Список литературы

[1] Цикл видеоуроков по OpenGL (плейлист) – URL: https://www.youtube.com/playlist?list=PLRIWtICgwaX0u7Rf9zkZhLoLuZVfUksDP

[2] Спецификация языка GLSL – URL: https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.4.40.pdf

[3] Спецификация OpenGL – URL: https://www.khronos.org/registry/OpenGL/specs/gl/glspec44.core.pdf

[4] Библиотека LWJGUI – URL: https://github.com/orange451/LWJGUI