Национальный исследовательский университет «МЭИ»

Институт автоматики и вычислительной техники

Кафедра вычислительной техники

КУРСОВАЯ РАБОТА
ПО КУРСАМ
ГРАФИЧЕСКИЕ СИСТЕМЫ
И
ГЕОМЕТРИЧЕСКОЕ МОДЕЛИРОВАНИЕ В САПР

Исполнитель

группа: А-06м-16

Мян Р. А.

Преподаватель

Пирогова М. А.

Лешихина И. Е.

Москва, 2016

Техническое задание

Геометрическое моделирование в САПР

Задание 1
Задать кодом Фримена треугольник (число отрезков больше 3). Осуществить аффинные преобразования треугольника: • отрицательный поворот относительно середины одной из сторон; • перенос вдоль оси X; • затем неоднородное (локальное) масштабирование (разные коэффициенты масштабирования по координатным осям); • отражение относительно прямой $y=6x+5$. Преобразования выполнять: a) в пошаговом режиме; b) как комплексное преобразование.

Задание 2
Построить: • Кривую из двух конических сечений. Изменяя веса, менять тип кривой. • Кривую на основе периодического B-сплайна 3-ей степени, число опорных точек не меньше 19. Полигоны обеих кривых совпадают по первой и последней точкам, графики совмещены.

Задание 3
Построить lofting поверхность, выполняя общее масштабирования. Профиль – кривая из двух конических сечений. Направляющая – отрезок прямой. Образующие – две кривые на основе периодического B-сплайна 3-ей степени, число опорных точек не меньше 19. Образующие связаны с профилем.

Техническое задание

Графические системы

Используя любой из известных Вам наборов управляющих графических объектов инструментальных средств графической оболочки операционной системы UNIX (ИПВУ) написать командный файл – скрипт, реализующий пример оконного пользовательского интерфейса работы в рамках задания на Курсовую работу по Геометрическому моделированию: интерфейс должен содержать меню задания или выбора из списков (возможно – в виде радиокнопок или с использованием других управляющих элементов) необходимых параметров, данных, цветов для визуализации кривых и поверхностей, опорных точек и т.д. для визуализации результата выполнения задания на КР по ГМ. Предусмотреть возможность переключения при выборе из списка цветов в режимы различного представления доступных в системе значений цветов: в виде перечисления текстовых наименований, или в виде цветовых индикаторов. Выполнять проверку введенное число опорных точек (не меньше 19), на совпадение цветов визуализации кривых (поверхностей) и опорных точек, например, удаление выбранного значения первого цвета из списка для выбора второго, или при независимо производимом выборе обоих цветов и выявленном после выбора такой пары равенства значений обоих цветов вывода на экран пользователя текстового сообщения-предупреждения о недопустимости такого выбора с последующим рестартом процедуры выбора цветов. Интерфейс можно сопроводить всплывающими окнами с подсказками и демонстрационными примерами: что такое коническое сечение, периодический B-сплайн 3-й степени – креатив приветствуется.

При разработке GUI использовать максимально возможное количество различных управляющих элементов выбранного языка программирования GUI ИПВУ (например TCL/Tk) для ввода/выбора различных величин. Полученный интерфейс должен быть дружественным, что означает: наличие достаточного количества подсказок и регламентирующих действия пользователя надписей, должен содержать механизм выхода из системы, прерывания работы, повторного выполнения задания с измененными параметрами.

Скрипт должен быть хорошо документирован. Пояснительная записка по КР должна содержать постановку задачи, словесное описание разработанного алгоритма, блок-схему скрипта и хорошо документированный листинг текста скрипта.

Выбор инструментов для реализации ТЗ

В качестве инструмента для реализации ТЗ, выбран язык программирования Python. Этот язык, подобно Tcl, является интерпретируемым и кроссплатформенным. Сильные его стороны – актуальность, большое количество разработчиков и, как следствие, множество библиотек.

Для реализации GUI, Python предлагает различные библиотеки. Например, Tkinter – встроенная библиотека, работа с которой сильно напоминает Tk для Tcl; PyQt – библиотека, являющаяся «оберткой» над одноименной библиотекой для C++. Tkinter прост и лаконичен. PyQt крупнее и перспективнее. В данной работе используется PyQt.

В задачах по геометрическому моделированию активно используется матричный аппарат. Для Python существует библиотека NumPy, которая предоставляет требуемый функционал.

Установка программы (Ubuntu 16.04)

Для загрузки используемых в скриптах библиотек, необходимо выполнить в терминале следующие команды:

```
sudo apt-get install python3-pyqt5
sudo pip3 install numpy
```

Если последняя команда не сработает, использовать:

```
sudo apt-get install python3-numpy
```

Структура кода программы

Код программы разбит на 2 части – Model и View.

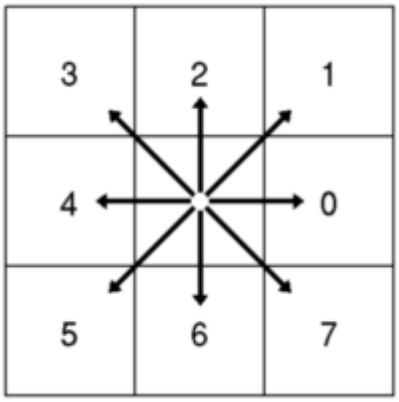
В Model помещены скрипты, реализующие логику программы (курс Геометрическое моделирование в САПР).

В View помещены скрипты, связанные с реализацией GUI (курс Графические системы).

Исходный код снабжен комментариями.

Часть 1. Модель

Код Фримена и аффинные преобразования



Код Фримена – метод компактного представления контуров объекта. С помощью цепного кода граница объекта представляется в виде последовательности соединенных отрезков, для которых указаны длина и направление. В качестве длины принят единичный отрезок системы координат. Направление задается числами от 0 до 7 в соответствии с рисунком справа. Методы для работы с кодом Фримена реализованы в скрипте **Model.Freeman**.

Аффинные преобразования основаны на матричном аппарате. Множество исходных точек представляется в виде матрицы. Преобразование исходных точек выполняется за счет умножения на соответствую матрицу преобразования. Для получения обратного эффекта, необходимо сформировать обратную матрицу преобразования. Матрицы преобразований реализованы в скрипте **Model.Transformation**. Выполняя различные манипуляции над матрицей исходных точек, получается матрица точек в новой конфигурации.

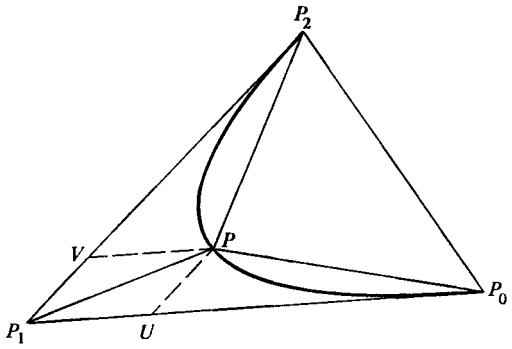
Методы для выполнения аффинных преобразований над фигурой, заданной кодом Фримена, реализованы в скрипте **Model.Freeman**.

Кривая из двух конических сечений

Коническое сечение получено на основе рациональной кривой Безье 2-ой степени, которая задается следующей формулой:

$$r(t)=\frac{(1-t)^2 p_0+2t(1-t)w(p)p_1+t^2 p_2}{(1-t)^2+2t(1-t)w(p)+t^2}\quad,\quad 0\le t\le 1$$

где $w=w(p)$ – вес вершины p_1, который определяется положением четвертой точки p относительно первых трех. Точки p_0, p_1, p_2 и p на рисунке справа обозначены через P_0, P_1, P_2 и P [2, стр. 99].



Последовательно построив обе кривые, получим кривую из двух конических сечений. Класс, реализующий рациональную кривую Безье второй степени представлен в скрипте **Model.Rbezier**.

Периодический B-сплайн 3-ей степени

Для построения периодического B-сплайна используется следующая формула:

$$P(t)=\sum_{i=1}^{n+1} B_i N_{i,k}(t) \quad , \quad t_{min}\le t\le t_{max} \quad , \quad 2\le k\le n+1$$

где B_i есть $n+1$ вершина многоугольника, а $N_{i,k}$ – нормализованные функции базиса B-сплайна.

Для i-й нормализованной функции базиса порядка k (степени $p=k-1$) функции базиса $N_{i,k}(t)$ определяются рекурсивными формулами Кокса-де Бура:

$$N_{i,1}(t)=\begin{cases}1, \textit{если } x_i\le t\le x_{i+1}\\ 0, \textit{иначе}\end{cases} \quad \text{и} \quad N_{i,k}(t)=\frac{(t-x_i)N_{i,k-1}(t)}{x_{i+k-1}-x_i}+\frac{(x_{i+k}-t)N_{i+1,k-1}(t)}{x_{i+k}-x_{i+1}}$$

Величины x_i – это элементы узлового вектора, удовлетворяющие отношению $x_i\le x_{i+1}$. Параметр t изменяется от t_{min} до t_{max} вдоль кривой $P(t)$. Считается, что $\frac{0}{0}=0$. [1, стр. 311]

Длина вектора параметризации m (узлового вектора) степень p и число опорных точек $(n+1)$ связаны следующим образом:

$$m=n+p+2$$

Узловой вектор для периодических базисных функций имеет вид $\{0,1,2,\ldots,n+p+1\}$.

Расчет кривой происходит только на так называемых *существенных интервалах* [3] (т.е. интервалах, на которых определена аппроксимирующая функция). Число существенных интервалов вычисляется по следующей формуле:

$$interval=n-(p-1)$$

Значения t_{min} и t_{max} принимают следующие значения:

$$t_{min}=p \quad , \quad t_{max}=t_{min}+interval$$

Отметим основное свойство данной кривой – периодический B-сплайн не проходит через первую и последнюю опорные точки.

Класс для работы с периодическим B-сплайном представлен в скрипте **Model.PBSpline**.

Lofting поверхность

Для построения lofting поверхности были решены приведенные ниже подзадачи.

1. Определить плоскость сечения.
2. Для каждой образующей кривой, определить точку пересечения с плоскостью сечения.
3. Определить расстояние между найденными точками (масштабный коэффициент)
4. Пересчитать точки кривой профиля для образовавшейся системы координат.

Определение плоскости сечения

Плоскость сечения расположена перпендикулярно направляющей прямой. Следовательно, направляющая прямая совпадает с нормалью к плоскости сечения. Зная точку и вектор нормали, можно определить плоскость сечения по уравнению, известному в математике как «Уравнение плоскости, проходящей через данную точку» [4, стр. 130]. Уравнение имеет следующий вид:

$$A(x-x_1)+B(y-y_1)+C(z-z_1)=0 \quad (*)$$

где:

A , B , C – координаты вектора нормали;

x_1 , y_1 , z_1 – координаты точки, лежащей в плоскости.

Определение точки пересечения кривой и плоскости

Для нахождения точки пересечения кривой с плоскостью, используется следующий подход. Кривая представляется в виде множества маленьких отрезов (принцип кусочно-линейной аппроксимации кривой) и для каждого отрезка осуществляется поиск точки пересечения с плоскостью.

Перед процедурой поиска точки пересечения, проводится предварительный анализ взаимного расположения границ отрезка с плоскостью. Используя левую часть уравнения (*), можно определить расположение пары точек следующим образом. Если для заданной пары точек, правая часть уравнения представляет ненулевые числа одного знака, то точки лежат по одну сторону от плоскости. Иначе, если знаки разные, то точки лежат по разные стороны от плоскости и, следовательно, отрезок, проходящий через эти точки, пересекает данную плоскость.

Убедившись, что отрезок пересекает плоскость, решается задача, известная в математике как «Поиск точки пересечения прямой с плоскостью» [4, стр. 145]. Для этого решается система уравнений:

$$\begin{cases} \frac{x-x_1}{m}=\frac{y-y_1}{n}=\frac{z-z_1}{p} \\ Ax+By+Cz+D=0 \end{cases} \quad (**)$$

где:

m , n , p – координаты направляющего вектора отрезка;

$D=-(A*x_1+B*y_1+C*z_1)$ – свободный член уравнения плоскости

Решение данной системы найдём с помощью параметрических уравнений прямой:

$$\begin{cases} x=x_1+mt \\ y=y_1+nt \\ z=z_1+pt \end{cases}$$

Подставив x, y, z из этих уравнений в (**) и выразив t , получим:

$$t=-(\frac{Ax_1+By_1+Cz_1+D}{Am+Bn+Cp})$$

Значение отрезка в найденном параметре является искомой точкой пересечения.

Расчет масштабного коэффициента

Масштабный коэффициент (расстояние между двумя точками) рассчитывается по формуле:

$$M=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2}$$

Общее масштабирование выполняется умножением точек профиля на M делённое на длину основания профиля.

Пересчет точек кривой профиля для произвольной системы координат

Для построения профиля в плоскости сечения, реализуем пересчет точек локальной системы координат, начинающейся в точке пересечения одной из образующих с плоскостью и имеющей следующий базис:

- Ось X – единичный направляющий вектор основания профиля;
- Ось Z – единичный направляющий вектор направляющей;
- Ось Y – результат векторного произведения векторов выше.

Результирующие точки для глобальной системы координат получаются по формуле [5]:

$$\vec{M}_{глоб.}=\vec{A}*\vec{M}_{лок.}+\vec{l}$$

где:

$\vec{A}$ – матрица перехода от базиса глобальной системы координат к базису локальной системы координат (элементы матрицы см. в коде);

$\vec{l}$ – координаты точки начала координат локальной системы координат;

$\vec{M}$ – матрица точек кривой.

Класс для реализации lofting поверхности расположен в скрипте **Model.LoftSurface**. Работа с прямыми (параметрическое задание, расчет длины, расчет направляющего вектора) реализована в отдельном классе, расположенном в скрипте **Model.Line**.

Часть 2. Представление

Общие сведения

Для PyQt существует утилита QtDesigner, позволяющая в графическом режиме перетаскивать и настраивать виджеты. Однако, в учебных целях, размещение виджетов на форме выполнено вручную.

Для размещения виджетов, в PyQt, так же как и для Tcl\Tk, имеются компоновщики виджетов. При реализации GUI активно использовались QVBoxLayout, QHBoxLayout, напоминающие по принципу действия компоновщик Pack для Tk. Также применялся QFormLayout, который позволяет компактно располагать виджеты и пояснения к ним (QLabel). Полезным свойством данного компоновщика является возможность автоматически определять оптимальные размеры виджетов.

В программе имеется главное меню, из которого осуществляется навигация по задачам. Виджеты главного меню расположены в скрипте **View.MainPage**.

Виджеты для каждой задачи расположены в скриптах **View.FirstTaskPage**, **View.SecondTaskPage**, **View.ThirdTaskPage**. Реализация каждой страницы разбита на фрагменты, что позволяет, при необходимости, повторно использовать размещенные виджеты. Это достигается при помощи ООП. Фрагмент страницы представляет собой класс, в котором расположены виджеты. Создавая экземпляр класса, создаются новые копии виджетов. Такой прием используется, например, при создании виджетов задачи 3. В задаче 3 используются созданные для задачи 2 виджеты.

Для экономии места окна используется виджет QToolBox. Он позволяет хранить в себе наборы виджетов и переключаться между ними.

Компонент для визуализация графиков реализован вручную на основе так называемой канвы (в PyQt функционал канвы реализует QPainter). Для хранения данных, изображенных на канве, используются характерные для python типы данных – словари и списки. Для обеспечения возможности отображать трехмерное изображение, используется диметрическая проекция. Разработанные компоненты расположены в скриптах **View.Plot**.

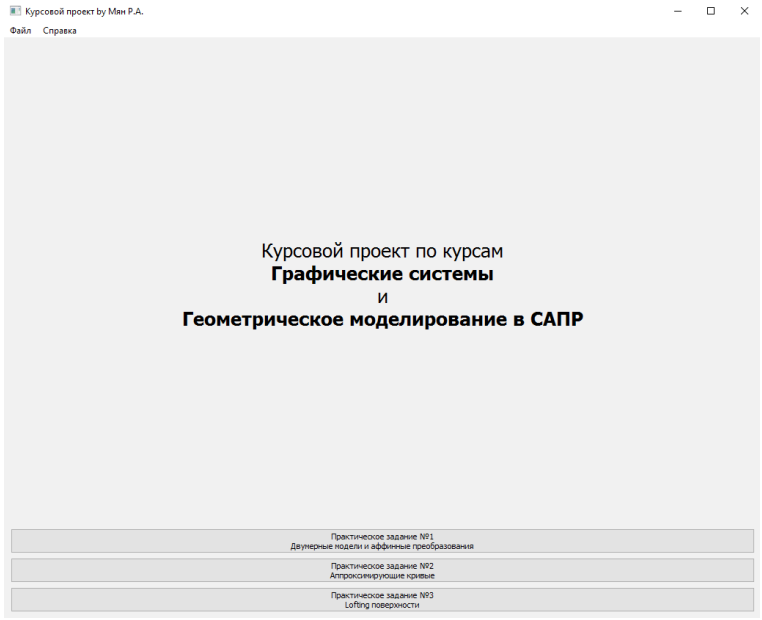
Осуществить выход из программы можно как стандартным способом (нажатием на крестик), так и воспользовавшись пунктом меню Файл → Выход.

Обработка исключений

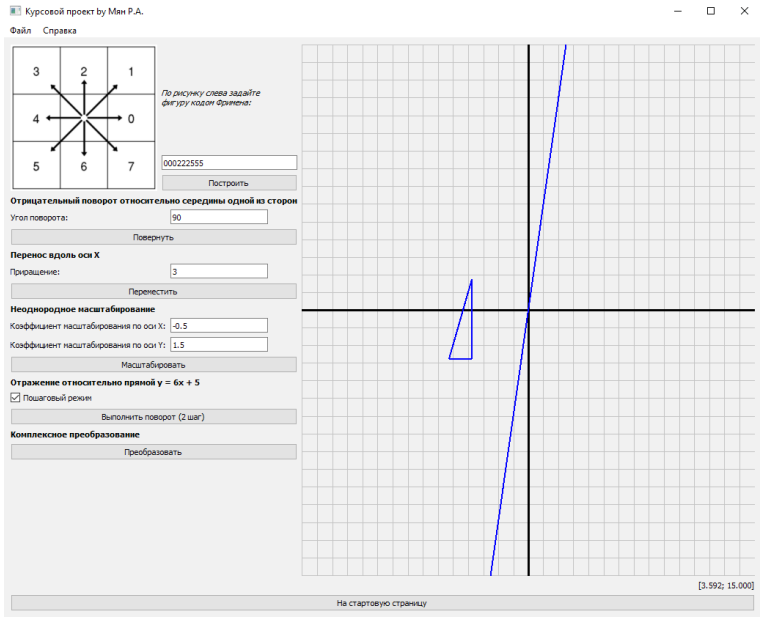
Чтобы не пугать пользователей всплывающими окнами с предупреждениями, виджеты подобраны таким образом, чтобы пользователь физически не смог ввести неправильные данные. Например, в поле, предназначенное для ввода чисел, невозможно ввести буквы.

Скриншоты

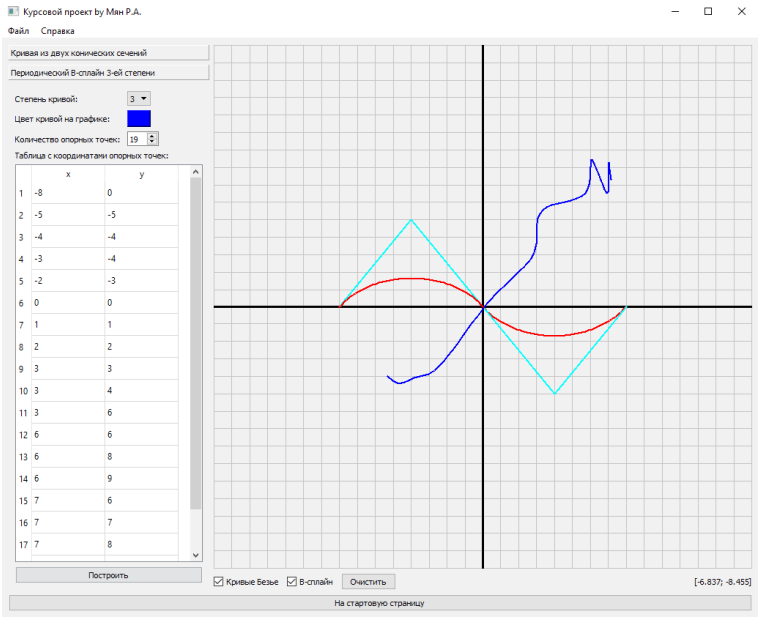
- Главное меню



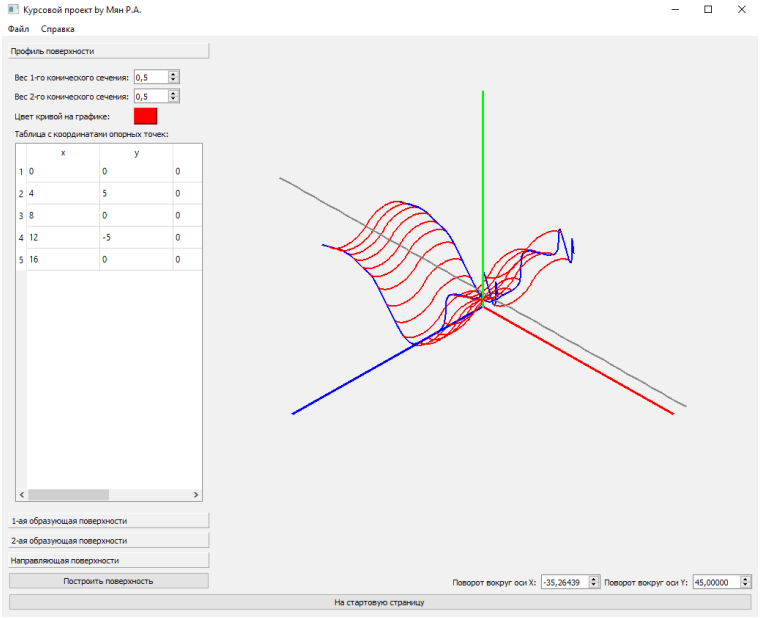
- Задача 1



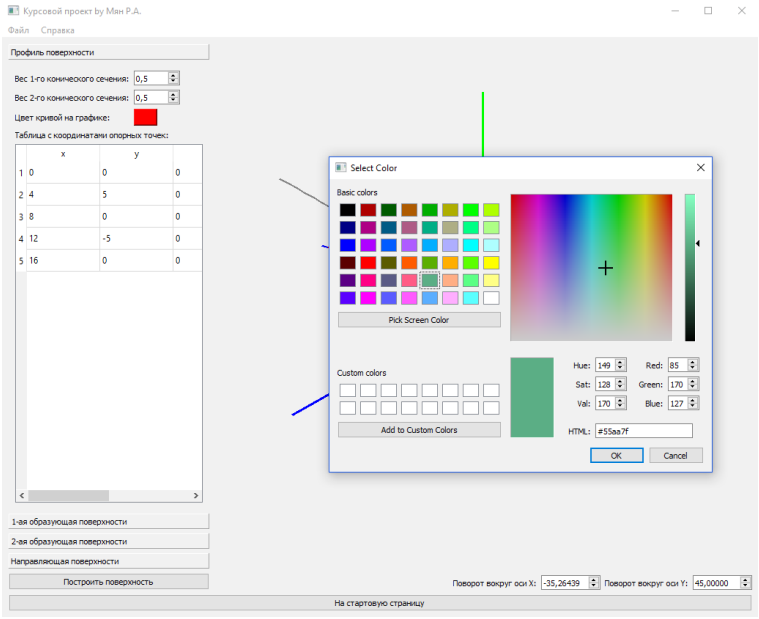
■ Задача 2



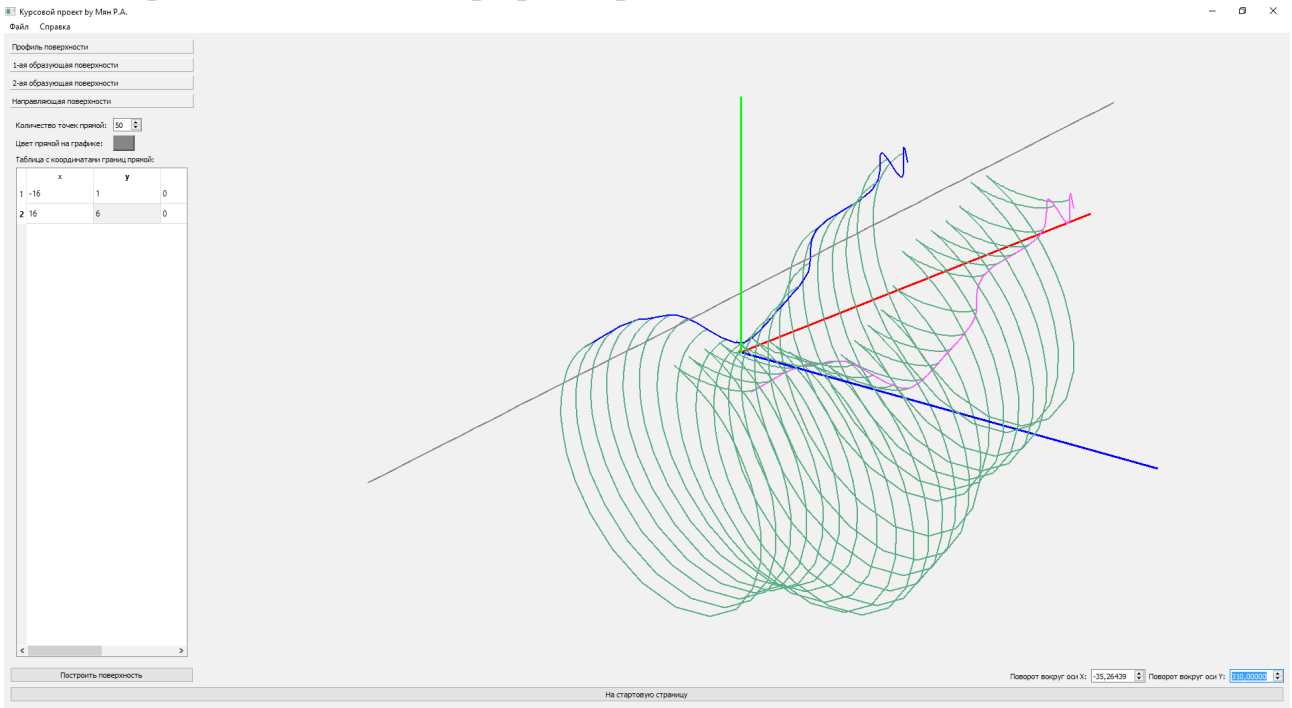
■ Задача 3



- Окно выбора цвета



- Поворот камеры и визуализация нестандартной поверхности (отрицательный вес в профиле кривой)



Используемые источники

1. Роджерс Д., Адамс Дж. – Математические основы машинной графики, 2001
2. Голованов Н. Н. – Геометрическое моделирование, 2002
3. http://ileshikhina.narod.ru/spl.htm
4. Шнейдер В.Е., Слуцкий А.И., Шумов А.С. – Краткий курс высшей математики, 1972
5. http://www.intuit.ru/studies/courses/70/70/lecture/2096?page=8