Московский авиационный институт
(Национальный исследовательский университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: «Вычислительная математика и программирование»
Дисциплина: «Компьютерная графика»

Курсовой проект
по курсу «Компьютерная графика»
Тема: «Каркасная визуализация поверхности»

Студент: Тимофеев А. В.
Преподаватель: Морозов А. В.
Группа: М80-307Б
Дата:
Оценка:
Подпись:

Москва, 2022

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 21: Линейчатая поверхность Кунса (границы – кубические кривые Безье 3D)

Решение задачи

Поверхность Кунса.

Поверхность Кунса задается четырьмя контурными линиями $c_0(s)$, $c_1(s)$, $d_0(t)$, $d_1(t)$, имеющими точки пересечения $c_0(0) = d_0(0)$, $c_0(1) = d_1(0)$, $c_1(0) = d_0(1)$, $c_1(1) = d_1(1)$ (угловые точки). Сама же поверхность Кунса образуется как сумма двух линейчатых поверхностей L_c , Ld за вычетом поверхности B, построенной по точкам пересечения контурных линий. Дополнительная поверхность вычитается для компенсации того, что при сложении поверхностей граничные точки учитываются дважды, нарушая таким образом требование “натянутости” поверхности на граничные кривые.

Линейчатые поверхности Lc , Ld представляют собой поверхности, строящиеся движением прямой линии по двум противолежащим контурным линиям. Т.о. для вычисления линейчатых поверхностей пользуемся линейной интерполяцией:

$$L_c(s,t) = (1 - t)c_0(s) + tc_1(s)$$

$$L_d(s,t) = (1 - s)d_0(t) + sd_1(t)$$

Результирующая поверхность должна проходить через граничные кривые и угловые точки, а т.к. при сложении поверхностей значения там были учтены дважды, то дополнительная вычитаемая поверхность должна совпадать с значениями с о значениями на границе и в угловых точках. Этого можно добиться применив билинейную интерполяцию:

$B(s,t) = c_0(0)(1 - s)(1 - t) + c_0(1)s(1 - t) + c_1(0)(1 - s)t + c_1(1)st$

Результирующую поверхность получаем следующим образом:

$C(s,t) = L_c(s,t) + L_d(s,t) - B(s,t)$

Кривые Безье

Кривая Безье в общем виде задается многоугольником и имеет математическое параметрическое представление вида:

$\sum_{i=0}^{n} BiJn,i(t)n, 0 \le t \le 1,$

где базис Безье или Бернштейна, или функция аппроксимации

$$J_{n,i} = \binom{n}{i} t^i (1-t)^{n-i}$$

$$\binom{n}{i} = \frac{n!}{i!\,(n-i)!}$$

$J_{n,i}$ - это i-тая функция базиса Бернштейна порядка n. Здесь i — порядковый номер опорной вершины, n — порядок определяющей функции базиса Бернштейна — и, следовательно, сегмента полиномиальной кривой, на единицу меньше количества точек определяющего многоугольника.

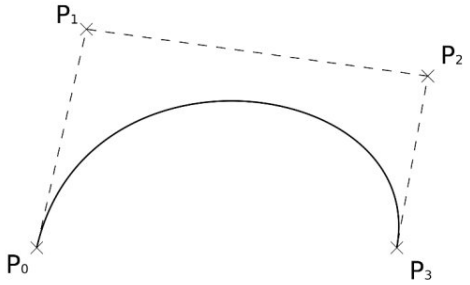


Рис. 2.1. Кривая Безье и, образующий ее, многоугольник

$\binom{n}{i}$ - биномиальные коэффициенты.

B_i - функция компонент векторов опорных вершин (координаты вершин многоугольника Безье).

В данной работе был использован частный случай кривых Безье – кубические кривые.

В параметрической форме они задаются следующим образом:

$B(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3, t \in [0,1]$,

где P_0, P_1, P_2, P_3 – опорные точки (вектора в 3-х мерном пространстве), задающие многоугольник, определяющий форму кривой.

Линия стартует в точке P_0, двигается в направлении P_1,отклоняясь к P_2, и, наконец, заканчивается в точке P_3. Кривая не проходит через точки P_1, P_2. Они используются для указания направления кривой.

Описание программы

В программе реализованы два основных класса BezierSpline, CoonsSurface, предоставляющие методы для вычисления координаты точки на поверхности, отрисовки как самих поверхностей и кривых, так и каркаса их задающего. На основе функционала этих классов и происходит расчет сетки поверхности для отображения. Так же для взаимодействия с пользователем реализовано управление мышью.

Описание работы программы

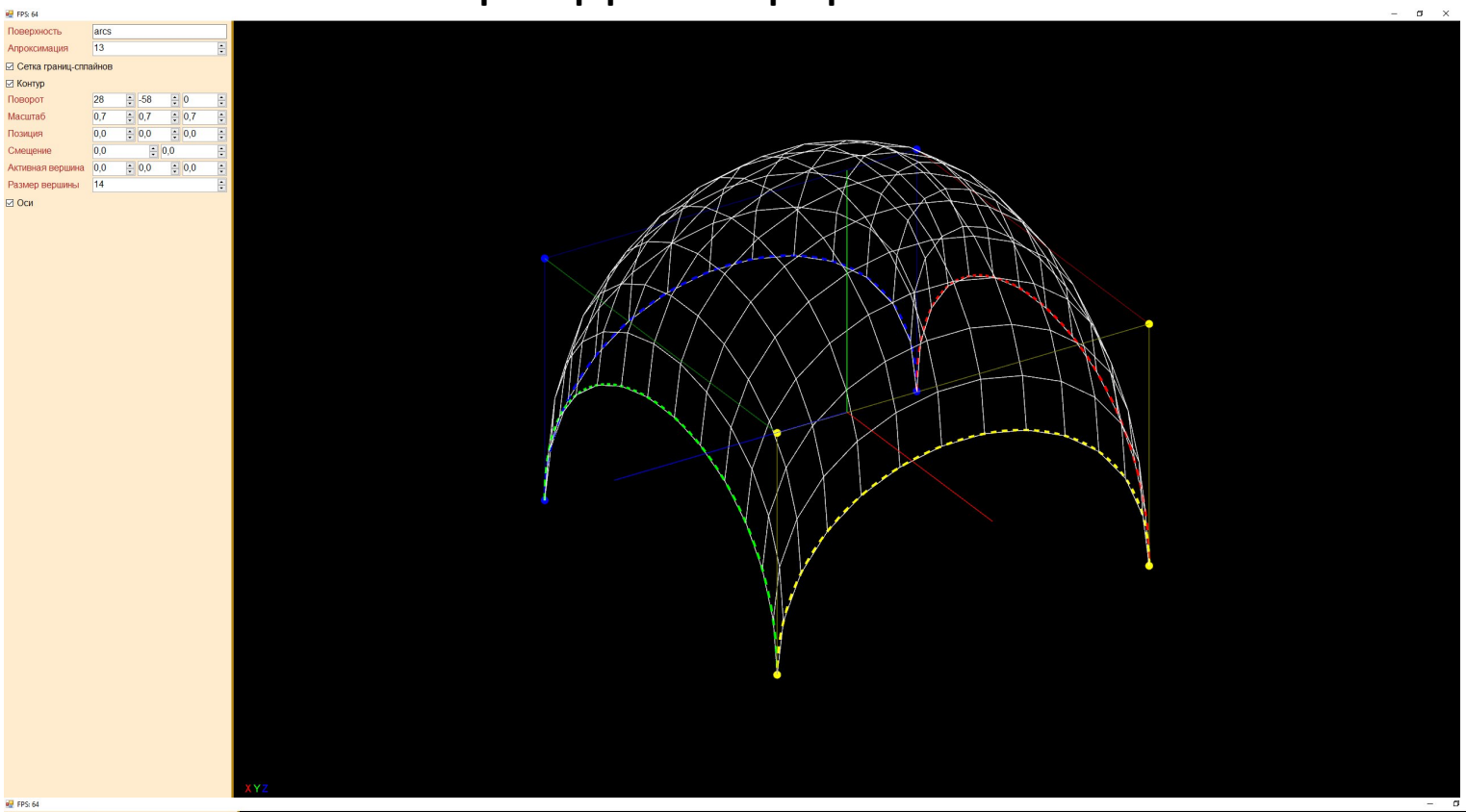
Программа позволяет визуализировать линейчатую поверхность Кунса, ограниченную кубическими кривыми Безье.

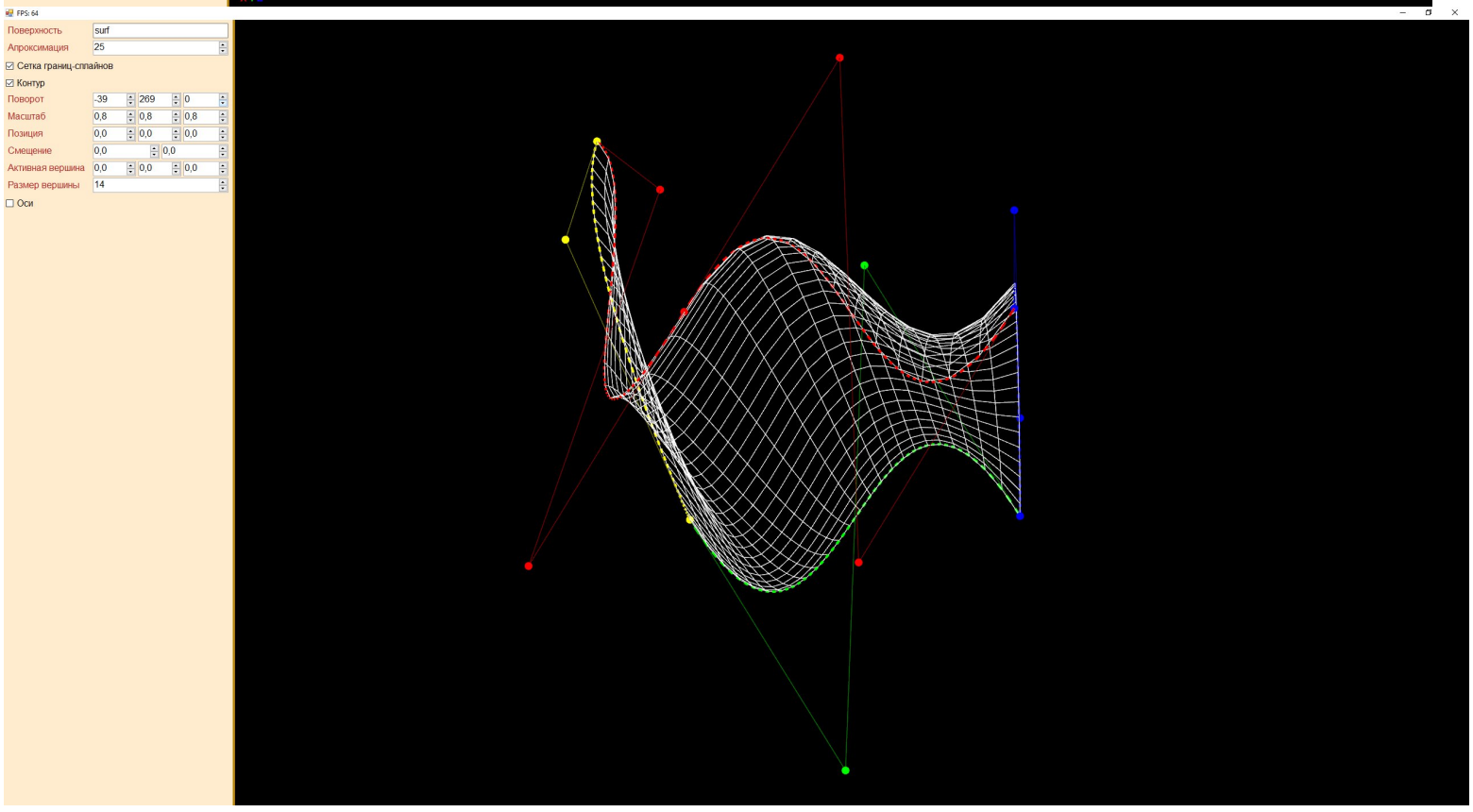
Чтобы изменить направления кривых, и саму поверхность соответственно, нужно кликом мышки по желаемой вершине выделить ее (подсветится розовым) и передвинуть используя поле “Активная вершина” на панели управления.

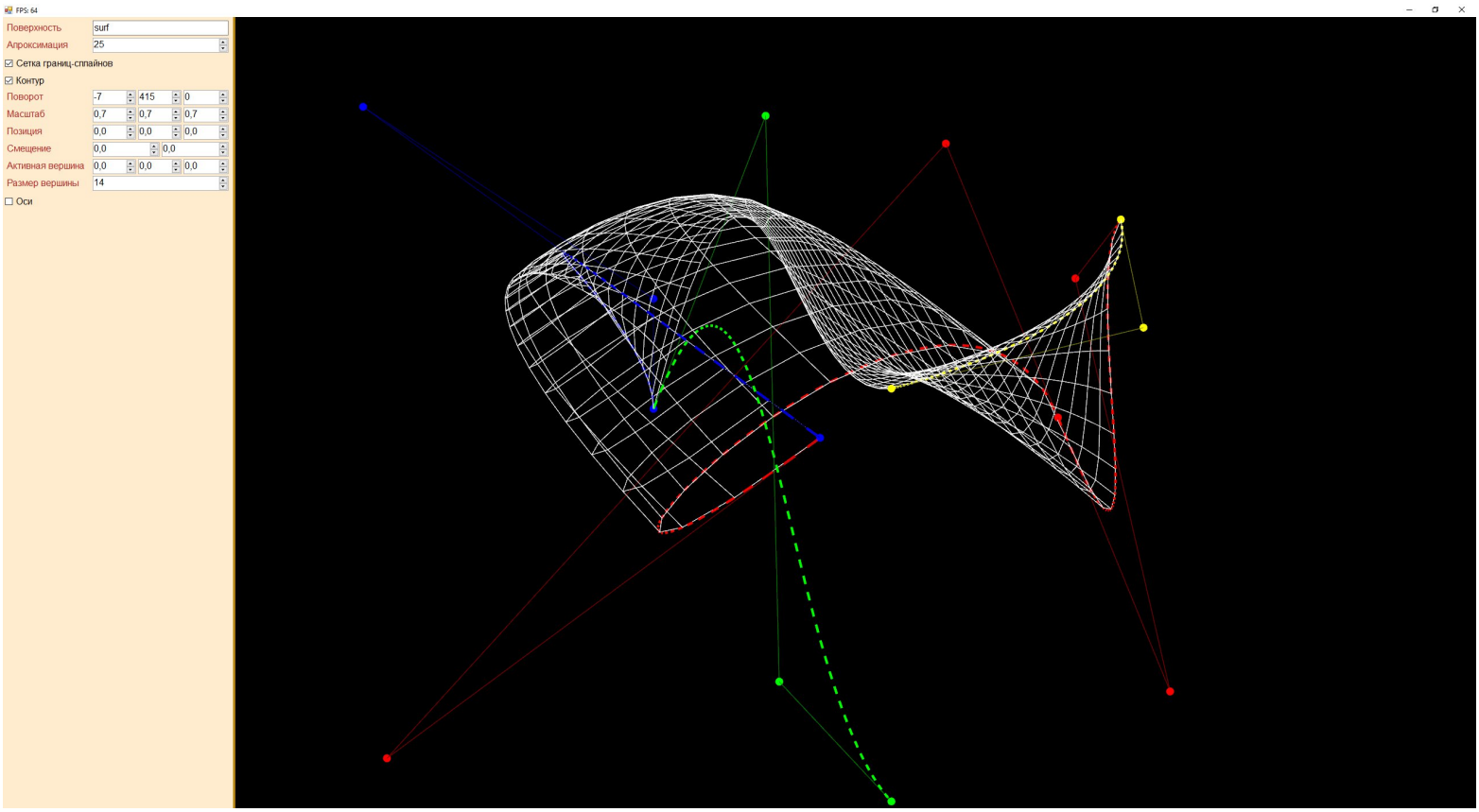
Так же программа позволяет изменить точность аппроксимации поверхности при помощи поле “Апроксимация”. Она отражается в количестве и плотности параметрических линий, образующих поверхность.

Программа предоставляет возможность аффинных преобразований поверхности: поворот, масштабирование, перемещение в пространстве при помощи полей “Поворот”, “Масштаб”, “Позиция” соответственно.

Пример работы программы







Листинг программы

(основная функция)

```
protected override unsafe void OnDeviceUpdate(object s, OGLDeviceUpdateArgs e){
    var gl = e.gl;
    // Очищаем буфер экрана и буфер глубины (иначе рисоваться все будет поверх старого)
    gl.Clear(OpenGL.GL_COLOR_BUFFER_BIT | OpenGL.GL_DEPTH_BUFFER_BIT |
OpenGL.GL_STENCIL_BUFFER_BIT);
    var modelMat = new float[16];
    gl.GetFloat(OpenGL.GL_MODELVIEW_MATRIX, modelMat);
    var projectionMat = new float[16];
    gl.GetFloat(OpenGL.GL_PROJECTION_MATRIX, projectionMat);
    var mM = new DMatrix4(modelMat.Select(val => (double) val).ToArray());
    var pM = new DMatrix4(projectionMat.Select(val => (double) val).ToArray());
    mM.Transpose();
    pM.Transpose();
    transformationMatrix = pM * mM;
    if (Wireframe || Border){
        var b0Color = new DVector3(1.0, 0.0, 0.0);
        var b1Color = new DVector3(0.0, 1.0, 0.0);
        var b2Color = new DVector3(0.0, 0.0, 1.0);
        var b3Color = new DVector3(1.0, 1.0, 0.0);
        if (Border){
            surface.border0.Draw(gl, b0Color);
            surface.border1.Draw(gl, b1Color);
            surface.border2.Draw(gl, b2Color);
```

```
        surface.border3.Draw(gl, b3Color);
    }
    if (Wireframe){
        surface.border0.DrawWireframe(gl, b0Color);
        surface.border1.DrawWireframe(gl, b1Color);
        surface.border2.DrawWireframe(gl, b2Color);
        surface.border3.DrawWireframe(gl, b3Color);
    }
}
gl.Color(1.0, 1.0, 1.0, 1.0);
surface.DrawByLines(gl);
if (Axis){
    DrawAxis(gl);
    gl.DrawText(20, 20, 1.0f, 0.0f, 0.0f, "Arial", 16, "X");
    gl.DrawText(35, 20, 0.0f, 1.0f, 0.0f, "Arial", 16, "Y");
    gl.DrawText(50, 20, 0.0f, 0.0f, 1.0f, "Arial", 16, "Z");
}
lock (ActiveVertexLocker){
    if (ActiveVertex != null){
        gl.Color(1.0, 0.0, 1.0, 1.0);
        gl.PointSize(15f);
        gl.Begin(OpenGL.GL_POINTS);
        gl.Vertex(ActiveVertex.Point.X, ActiveVertex.Point.Y, ActiveVertex.Point.Z);
        gl.End();
        gl.Color(1.0, 1.0, 1.0, 1.0);
        gl.PointSize(1f);
    }
}
return;}
```

Выводы

Выполнив курсовой проект, я научился реализовывать поверхность Кунса и кубические кривые Безье. Поверхности Кунса и кривые Безье, несмотря на относительную простоту реализации, повидимому, являются довольно мощными инструментами. Эти виды поверхностей и кривые широко применяются на практике, например, в САПРах и архитектуре.

Данный курс и в частности эта работа очень сильно помог мне разобраться во многих вещах, связанных с машинной графикой.