Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №7 по курсу «Компьютерная графика»

Студент: Д. А. Ваньков
Преподаватель: Г. С. Филиппов
Группа: М8О-307Б

Дата:
Оценка:
Подпись:

Москва, 2019

Построение плоских полиномиальных кривых.

Задача: Написать программу, строящую полиномиальную кривую по заданным точкам. Обеспечить возможность изменения позиции точек и, при необходимости, значений касательных векторов и натяжения.

Вариант №18:
B-сплайн. $n = 6$, $k = 3$. Узловой вектор равномерный.

1 Описание

Необходимо построить B-spline по трем точкам. Кривая строится в соответсвие с формулой по трем точкам:
$B_{i,k+1}(x) = (x - t_i)/(t_{i+k} - i_i) * B_{i,k}(x) + (t_{i+k+1} - x)/(t_{i+k+1} - t_{i+1}) * B_{i+1,k}(x)$

2 Исходный код

Объявляем глобальные переменные. Отвечающие за вектор узлов, контрольные точки, радиус, видимые линии и цвета.

```
vector<double> knots;
vector<dvec2> control;
float cRadius = 0.013f;
int selected = -1;
bool movePoint = false;
bool showPoints = true;
bool showColours = true;
bool niceLines = true;
```

Также нужно завести порядок B-spline, в нашем случае 3, и апрокисмацию.

```
int k = 3;
double uinc = 0.001;
double vinc = PI/16;
```

Генерируем точку на сплайне алгоритмом Бура.

```
dvec2 bspline(double u, int d) {

```

```
  dvec2 *c = new dvec2[control.size()];
  for (int i = 0; i <= k - 1; ++i) {
    c[i] = control[d - i];
  }

  for (int r = k; r >= 2; --r) {
    int i = d;
    for (int s = 0; s <= r - 2; ++s) {
      double u_i = knots[i];
      double u_ir1 = knots[i + r - 1];
      double omega = (u - u_i) / (u_ir1 - u_i);
      c[s] = omega * c[s] + (1.0 - omega) * c[s + 1];
      i--;
    }
  }

  dvec2 result = c[0];
  delete[] c;
  return result;
}
```

Прописываем функцию отрисовки.

```
void render() {
  glEnable(GL_DEPTH_TEST);
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  if (niceLines && !sRevolution) {
    glEnable(GL_LINE_SMOOTH);
    glHint(GL_LINE_SMOOTH_HINT, GL_NICEST);
    glEnable(GL_BLEND);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
  }
  else {
    glDisable(GL_LINE_SMOOTH);
    glDisable(GL_BLEND);
  }
```

Осталось только сгенерировать B-spline и отрисовать его, запустив окно. В этом цикле получаем индекс дельта отрезка в равномерном векторе узлов, генерируем цвета и вершины,а также отрисовываем.

```
for (double u = knots[k-1] + uinc; u <= knots[control.size()]; u += uinc) {
    int d = delta(u);

    if (control.size() >= d) {
      float cr = 1;
      float cg = 1;
```

```
7       float cb = 1;
8       if (showColours) {
9         cr = 0.5 * (sin(101 * u) + 1);
10        cg = 0.5 + 0.25 * (cos(11 * u) + 1);
11        cb = 0.5 * (sin(71 * u) + 1);
12      }
13      glColor3f(cr, cg, cb);
14
15      dvec2 point = bspline(u, d);
16      /*if (!sRevolution)*/
17      glVertex3f(point.x, point.y, 0);
18    }
19  }
```

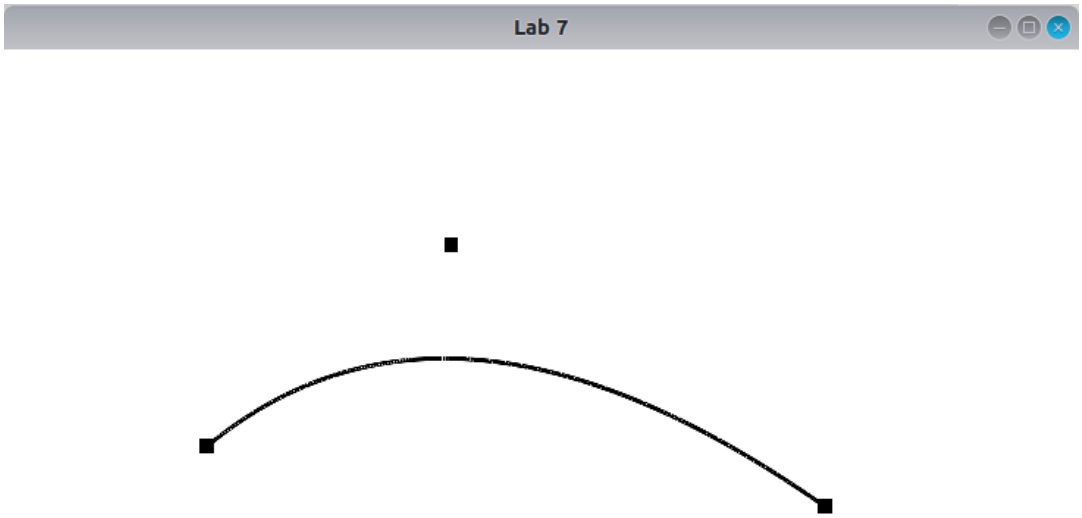
3 Консоль

В консоли необходимо скомпилировать исходный код и запустить. Затем поставить кликом мыши 3 вершины и получить кривую.

```
(base) chappybunny@chappybunny:~/CG/lab7$ g++ -std=c++11 B-splain.cpp -o main
-lglut -lGL -lGLU -lglfw
(base) chappybunny@chappybunny:~/CG/lab7$ ./main
```

После откроется черное окно.

Это окно можно изменять по размерам и перемещать по экрану без всяких побочных эффектов, кривая подстраивается под изменение размеров экрана и масштабируется соответствующим образом.



4 Выводы

Выполнив данную лабораторную работу по курсу «Компьютерная графика», я не столкнулся с определенными сложностями, однако узнал, что можно добавлять анимации, способные вращать фигуры по некоторым законам.