

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра автоматизированных систем управления

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
К ЛАБОРАТОРНЫМ РАБОТАМ

по дисциплине

«КОМПЬЮТЕРНАЯ ГРАФИКА»

для студентов направления подготовки

09.03.02 Информационные системы и технологии
заочной и заочной с сокращенным сроком форм обучения

Рассмотрено на заседании кафедры
«Автоматизированные системы
управления»

Протокол № 6 от 19 января 2017 г.

Утверждено на заседании
учебно – издательского совета ДонНТУ
Протокол № 3 от 6 апреля 2017 г.

Донецк
2017

Методические указания к лабораторным работам по дисциплине «Компьютерная графика» для студентов направления подготовки 09.03.02 «Информационные системы и технологии» заочной и заочной с сокращенным сроком форм обучения/ Составили: Васяева Т.А., Матях И.В. – Донецк: ДНТУ, 2017. – 28 с.

Методические указания содержат краткие теоретические сведения, методические рекомендации и задания для выполнения лабораторных работ по дисциплине «Компьютерная графика». Изложена методика выполнения каждой из лабораторных работ, требования к содержанию отчетов, список рекомендуемой литературы.

Методические указания к лабораторным работам предназначены для студентов направления подготовки 09.03.02 «Информационные системы и технологии» профиль «Информационные системы и технологии в технике и бизнесе» заочной и заочной с сокращенным сроком форм обучения.

Составители:

к.т.н., доц. Васяева Т.А.

ас. Матях И.В.

Рецензент: к.т.н., доц., доц. каф. АСУ

М.В. Привалов

Рецензент: к.т.н., доц., доц. каф. АТ

В.В. Червинский

Ответственный за выпуск:

зав. каф. АСУ Привалов М.В.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
ЛАБОРАТОРНАЯ РАБОТА №1	5
ЛАБОРАТОРНАЯ РАБОТА №2	19
ПРИЛОЖЕНИЕ А	25
СПИСОК ЛИТЕРАТУРЫ	275

ВВЕДЕНИЕ

Компьютерная графика представляет собой одно из направлений развития систем обработки информации, связанной с изображением объектов, которое возникло в связи с необходимостью широкого использования систем компьютерной графики в виртуальной реальности, в глобальной сети Internet и системах интерактивной графики. Понятие «компьютерная графика» очень часто трактуется по-разному. Компьютерная графика – это новая отрасль знаний, которая, с одной стороны, представляет комплекс аппаратных и программных средств, используемых для формирования, преобразования и выдачи информации в визуальной форме на средства отображения ЭВМ. С другой стороны, под компьютерной графикой понимают совокупность методов и приемов для преобразования при помощи ЭВМ данных в графическое представление. Системы компьютерной графики обеспечивают пользователю широкий набор услуг и позволяют создавать целый ряд различных способов диалога, типа человек – компьютер, позволяют создавать анимационные и реалистичные изображения и совершенствуют способы ввода-вывода информации.

Изучение данной дисциплины вносит необходимый вклад в достижение ожидаемых результатов в профессиональной части программы подготовки студентов специальностей «Информационные системы и технологии».

Курс основан на изучении математических основ предметной области, значительное внимание уделяется программной реализации.

Основная задача курса – дать в доступной форме математический аппарат для создания изображений геометрических объектов на экране и их манипуляций, алгоритмы растровой графики, представление пространственных форм.

Лабораторная работа № 1

Тема: растровые алгоритмы построения контура фигур

Алгоритм Брезенхема рисования линии

Пусть необходимо построить отрезок начало которого имеет координаты (x_1, y_1) , а конец (x_2, y_2) . Обозначим $dx = (x_2 - x_1)$, $dy = (y_2 - y_1)$. Не нарушая общности, будем считать, что начало отрезка совпадает с началом координат, и прямая имеет вид $y = \frac{dy}{dx}x$, где $\frac{dy}{dx} \in [0, 1]$, т.е. $dy < dx$. Считаем, что начальная точка находится слева. Пусть на $(i-1)$ -м шаге текущей точкой отрезка является $P_{i-1} = (r, q)$. Необходимо сделать выбор следующей точки $S_i(r+1, q+1)$ или $T_i(r+1, q)$.

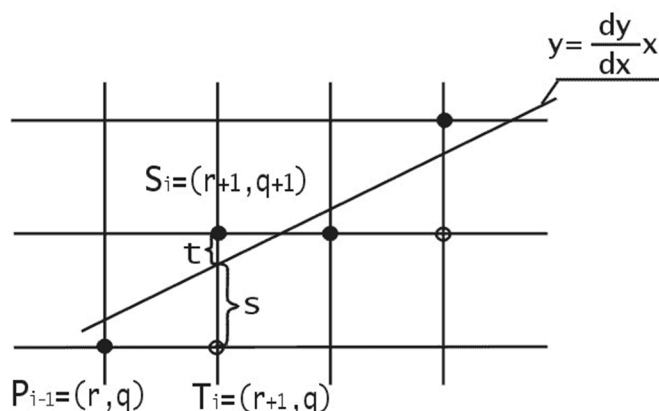


Рисунок 1.1 - Рисование отрезков прямых по методу Брезенхема

1. Рассчитаем $dx = (x_2 - x_1)$ и $dy = (y_2 - y_1)$.
2. $i = 1$, $d_i = 2dy - dx$, $y_i = y_1$, $x_i = x_1$.
3. Рисуем точку с координатами (x_i, y_i) .
4. Если $d_i < 0$, тогда $d_{i+1} = d_i + 2dy$ и $y_{i+1} = y_i$. Если $d_i \geq 0$, то $d_{i+1} = d_i + 2(dy - dx)$ и $y_{i+1} = y_i + 1$.
5. $x_{i+1} = x_i + 1$; $i = i + 1$.
6. Если $x_i < x_2$, то перейти на пункт 3, иначе — конец.

Если $dy > dx$, то необходимо будет использовать этот же алгоритм, но пошагово увеличивая y ($y_{i+1} = y_i + 1$) и на каждом шаге вычислять x ($x_i = x_i + 1$ или $x_i = x_i$), пока $y_i < y_2$.

Реализация этого алгоритма может выглядеть следующим образом:

```
void drawDot(GLint x, GLint y) // рисуем точку
{
    glBegin(GL_POINTS);
    glVertex2i(x,y);
    glEnd();
    glFlush();
}
```

```
void line(GLint x1, GLint y1, GLint x2, GLint y2)
// алгоритм Брезенхема для отрезка
{
    int dy=abs(y2-y1);
    int dx=abs(x2-x1);
    int tmp_x, tmp_y,x,y, flag;

    if (dx==0) { // вертикальная линия
        if (y2<y1) {tmp_y=y2;y2=y1; y1=tmp_y;}
        for (y=y1;y<=y2;y++)
            drawDot(x1,y);
        return;
    }

    if (dy==0) { // горизонтальная линия
        ...
    }
```

```

if (dy<dx) {                                     // наклонная линия
flag=0;
if (x2<x1)
{tmp_x=x2;    x2=x1;    x1=tmp_x;    tmp_y=y2;y2=y1;
y1=tmp_y;}
if (y2<y1){flag=1;}
int di;
di=2*dy-dx;
y=y1;
x=x1;
do {
    drawDot(x,y);
    if (di<0) {di=di+2*dy; x++;}
    else {di=di+2*(dy-dx); x++; if (flag==0) {y++;}
else {y--;}}
    } while (x<x2);
return;
    }

if (dy>dx){..... // наклонная линия
...
    }

if (dx==dy) // диагональ
{
    ...
}

}

```

Растровая развёртка окружности

Для упрощения алгоритма растровой развёртки стандартной окружности можно воспользоваться её симметрией относительно координатных осей и прямых $y = \pm x$; в случае, когда центр окружности не совпадает с началом координат, эти прямые необходимо сдвинуть параллельно так, чтобы они прошли через центр окружности. Тем самым достаточно построить растровое представление для 1/8 части окружности, а все оставшиеся точки получить симметрией (рис. 5.2).

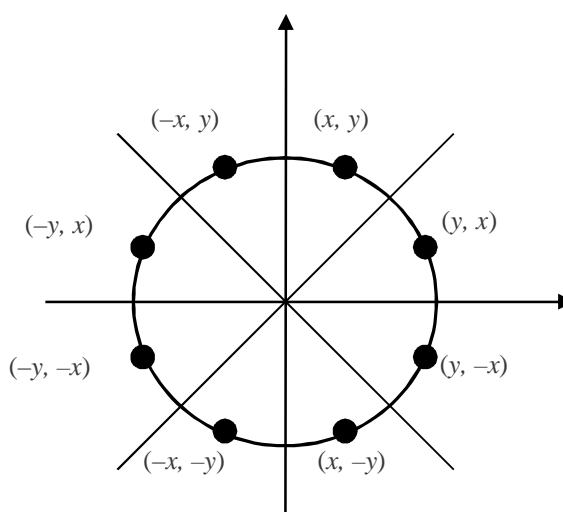


Рисунок 1.2 – Восьмисторонняя симметрия

Если точка (x, y) лежит на окружности, то легко вычислить семь точек, принадлежащих окружности, симметричных этой. То есть, имея функцию вычисления значения y по $x=0 \dots R/\sqrt{2}$ для построения дуги от 0° до 45° , можно реализовать функцию, которая будет по координате одной точки рисовать сразу восемь точек, учитывая симметрию окружности.

```
void cir_pix(int x0, int y0, int x, int y) // 8
точек на окружности
{
    drawDot(x0+x, y0+y);
    drawDot(x0+y, y0+x);
```



```

drawDot (x0+x, y0-y) ;
drawDot (x0+y, y0-x) ;
drawDot (x0-x, y0-y) ;
drawDot (x0-y, y0-x) ;
drawDot (x0-x, y0+y) ;
drawDot (x0-y, y0+x) ;
}

```

Алгоритм Брезенхейма для окружности

Рассмотрим участок окружности $x \in [0, R/\sqrt{2}]$ (рис. 1.2). На каждом шаге алгоритм выбирает точку $P_i(x_i, y_i)$, которая является ближайшей к истинной окружности. Идея алгоритма заключается в выборе ближайшей точки при помощи управляющих переменных, значения которых можно вычислить в пошаговом режиме с использованием небольшого числа сложений, вычитаний и сдвигов.

Рассмотрим небольшой участок сетки пикселей, а также возможные способы (от А до Е) прохождения истинной окружности через сетку (рис. 1.3).

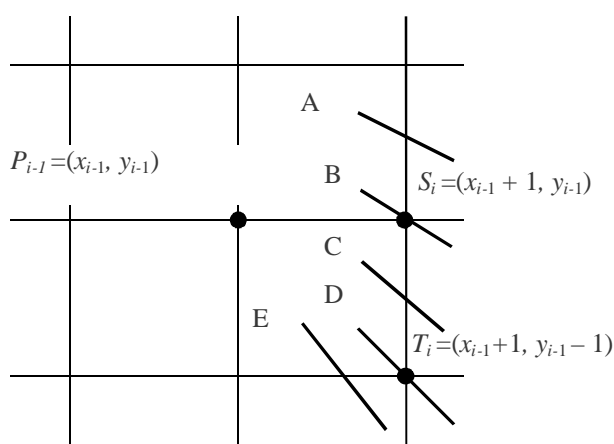


Рисунок 1.3 – Варианты прохождения окружности через растровую сетку

Предположим, что точка P_{i-1} была выбрана как ближайшая к окружности при $x = x_{i-1}$. Теперь найдем, какая из точек ($S_i(x_{i-1} + 1, y_{i-1})$ или $T_i(x_{i-1} + 1, y_{i-1} - 1)$) расположена ближе к окружности при $x = x_{i-1} + 1$.

Алгоритм формулируется следующим образом. Дано (x_0, y_0) – центр окружности, R – радиус.

1. Берем первую точку $i=1$, $x_i = 0$, $y_i = R$, рассчитываем $d_i = 3 - 2R$.
2. Вызов функции рисования 8-ми точек на окружности по координатам одной точки, учитывая смещение центра окружности от начала координат.
3. Если $d_i < 0$, то $d_{i+1} = d_i + 4x_i + 6$, $x_{i+1} = x_i + 1$, $y_{i+1} = y_i$.
4. Если $d_i \geq 0$, то $d_{i+1} = d_i + 4(x_{i-1} - y_{i-1}) + 10$, $x_{i+1} = x_i + 1$, $y_{i+1} = y_i - 1$.
5. $i = i + 1$.
6. Если $x_i > R/\sqrt{2}$, то перейти на пункт 2, иначе – конец.

Растровая развёртка эллипса

Простым методом растровой развертки эллипса является использование вычислений x и y по формулам:

$$x = R_x \cos \alpha, \quad (1.1)$$

$$y = R_y \sin \alpha \quad (1.2)$$

при пошаговом изменении угла α от 0° до 360° .

Можно также использовать симметрию эллипса. Существует модификация алгоритма Брезенхейма для эллипса.

Кривая Безье

Разработана математиком Пьером Безье. Кривые и поверхности Безье были использованы в 60-х годах компанией «Рено» для компьютерного проектирования формы кузовов автомобилей. В настоящее время они широко используются в компьютерной графике.

Кривые Безье описываются в параметрической форме:

$$x = P_x(t), \quad y = P_y(t). \quad (1.3)$$

Значение t выступает как параметр, которому соответствуют координаты отдельной точки линии. Параметрическая форма описания может быть удобнее для некоторых кривых, чем задание в виде функции $y = ?(x)$, поскольку функция $?(x)$ может быть намного сложнее, чем $P_x(t)$ и $P_y(t)$, кроме того, $?(x)$ может быть неоднозначной.

Многочлены Безье для P_x и P_y имеют такой вид:

$$P_x(t) = \sum_{i=0}^m C_m^i t^i (1-t)^{m-i} x_i, \quad P_y(t) = \sum_{i=0}^m C_m^i t^i (1-t)^{m-i} y_i, \quad (1.4)$$

где x_i и y_i — координаты точек-ориентиров P_i , а величины C_m^i — это известные из комбинаторики, так называемые сочетания (они также известны как коэффициенты бинома Ньютона):

$$C_m^i = \frac{m!}{i!(m-i)!}. \quad (1.5)$$

Значение m можно рассматривать и как степень полинома, и как значение, которое на единицу меньше количества точек-ориентиров.

Рассмотрим кривые Безье, классифицируя их по значениям m .

1. $m = 1$ (по двум точкам)

Кривая вырождается в отрезок прямой линии, которая определяется конечными точками P_0 и P_1 , как показано на рис. 5.4:

$$P(t) = (1-t)P_0 + tP_1 \quad (1.6)$$

2. $m = 2$ (по трем точкам), рис. 5.4:

$$P(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2. \quad (1.7)$$

3. $m = 3$ (по четырем точкам), кубическая (рис. 5.5), используется довольно часто, в особенности в сплайновых кривых:

$$P(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t) P_2 + t^3 P_3. \quad (1.8)$$

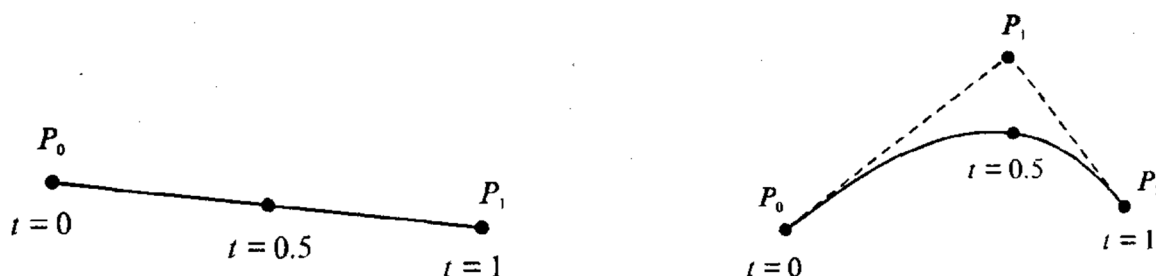


Рисунок 1.4 – Кривая Безье ($m=1$) и кривая Безье ($m=2$)

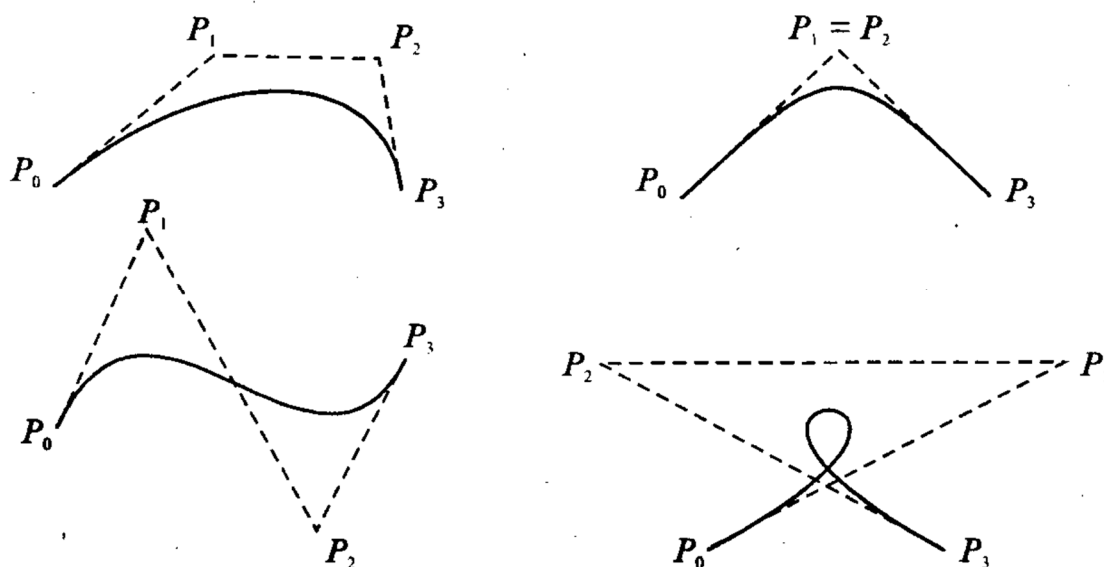


Рисунок 1.5 – Кубические кривые Безье ($m=3$)

Геометрический алгоритм для кривой Безье

Этот алгоритм позволяет вычислить координаты (x, y) точки кривой Безье по значению параметра t .

1. Каждая сторона контура многоугольника, который проходит по точкам-ориентирам, делится пропорционально значению t .
2. Точки деления соединяются отрезками прямых и образуют новый многоугольник. Количество узлов нового контура на единицу меньше, чем количество узлов предшествующего контура.
3. Стороны нового контура снова делятся пропорционально значению t . И так далее. Это продолжается до тех пор, пока не будет получена единственная точка деления. Эта точка и будет точкой кривой Безье (рис.5.6).

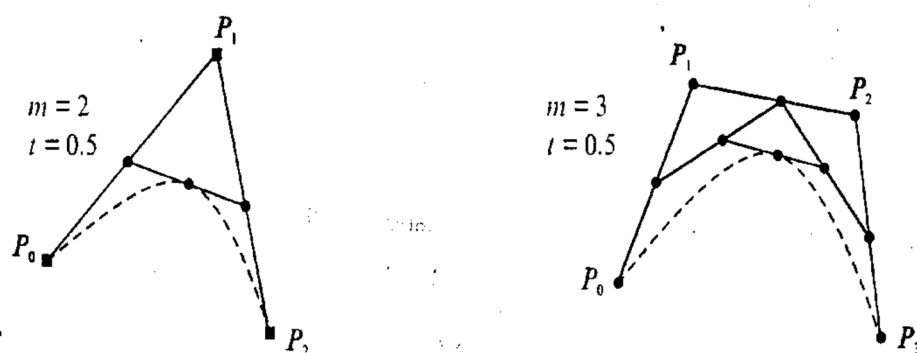


Рисунок 1.6 – Геометрический алгоритм для кривых Безье

Пример на языке C++:

```
int m=3; // степень полинома
int n=6; // количество точек на кривой
float t;
for (int l=0; l<=n; l++)
{
    t=(1.0/n)*l;
    for (int i=0; i<2; i++)
        for (int j=0; j<=m; j++)
            R[i][j]=(P[i][j]); // вспомогательный
//массив дублирует координаты точек – ориентиров.
    for (int i=0; i<2; i++) // координаты x и y
        for (int j=m; j>=0; j--)
            for (int k=0; k<j; k++)
                { R[i][k]+= t*(R[i][k+1]-R[i][k]);
                  T[i][l]=R[i][k] ;//координаты точек на
кривой
                }
}
```

Уравнения кривой Безье можно записать в матричном виде:

$$P(t) = [T][N][G] \quad (1.9)$$

где для $m = 3$:

$$P(t) = [T][N][G] = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}. \quad (1.10)$$

Матрица N представляет собой коэффициенты при соответствующих t (Первый столбец при P_0 , второй – при P_1 , третий – при P_2 и четвертый – при P_3 . Первая строка при t^3 , вторая – при t^2 , третья – при t и четвертая – свободный член). P_0, P_1, P_2 и P_3 – вершины многоугольника Безье.

Аналогично для $m = 4$:

$$P(t) = \begin{bmatrix} t_4 & t_3 & t_2 & t & 1 \end{bmatrix} \begin{bmatrix} 1 & -4 & 6 & -4 & 1 \\ -4 & 12 & -12 & 4 & 0 \\ 6 & -12 & 6 & 0 & 0 \\ -4 & 4 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix} \quad (1.11)$$

Сплайн Эрмита

Одним из способов задания параметрического кубического сплайна является указание координат начальной и конечной точек, а также векторов касательных в них. Такой способ задания называется формой Эрмита. Обозначим концевые точки P_1 и P_4 , а касательные векторы в них R_1 и R_4 .

$$M_h G_{hx} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_{1x} \\ P_{4x} \\ R_{1x} \\ R_{4x} \end{bmatrix} \quad (1.12)$$

Здесь M_h – Эрмитова матрица, G_h – геометрический вектор Эрмита. в матричном виде $x(t)$: $x(t) = TM_h G_{hx}$. Аналогично для остальных координат: $y(t) = TM_h G_{hy}$, $z(t) = TM_h G_{hz}$.

Форму кривой, заданной в форме Эрмита, легко изменять если учитывать, что направление вектора касательной задает начальное направление, а модуль вектора касательной задает степень вытянутости кривой в направлении этого вектора, как показано на рис. 5.7.

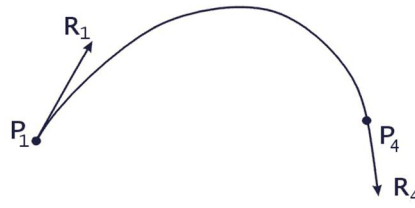


Рисунок 1.7 – Параметрический сплайн в форме Эрмита. Вытянутость кривой вправо обеспечивается тем, что $|R_1| > |R_4|$.

В-сплайны

Кривая, представленная в виде кубического В-сплайна, в общем случае может проходить через любые управляющие точки, однако она непрерывна и, кроме того, непрерывностью изменения обладают ее касательный вектор и кривизна (т. е. первая и вторая производные кривой непрерывны в конечных точках) в отличие от формы Безье, у которой в конечных точках непрерывны лишь первые производные (но которые проходят через управляющие точки). Таким образом, можно утверждать, что форма В-сплайнов «более гладкая», чем другие формы. В-сплайн описывается следующей формулой:

$$x(t) = TM_s G_{sx}, \quad (1.13)$$

где

$$M_s = \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}. \quad (1.14)$$

При аппроксимации управляющих точек P_1, P_2, \dots, P_n последовательно В-сплайнов необходимо применять между каждой парой соседних точек геометрические матрицы. Для аппроксимации в интервале, близком к точкам P_i и P_{i+1} , используется:

$$G_s^i = \begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix}, \quad 2 \leq i \leq n-2. \quad (1.15)$$

Пример построения сплайна

```
// построение сплайна
...
float Dx[4][1]={ {300}, // точки ориентиры для
                 {250}, // кривой Безье/ В-сплайна
                 {250}, // (координаты x)
                 {300}};
float Dy[4][1]={ {200}, // ---//----
                 {250}, // (координаты y)
                 {150},
                 {200}};
for(float i=0; i<=1;i+=0.01)
    poit_b(i,&Dx[0][0],&Dy[0][0]); // ваша функция для
рисования одной точки на сплайне
...
```

Пример 1. Построение дерева (рис. 1.8)

Точки – ориентиры для сегментов кривой Безье, сегменты нумеруются по часовой стрелке:

- 1) P= 4 0
 2 1
 2 3
 3.5 2.7
- 2) P= 3.5 2.7
 3 4.5
 3 7
 3.7 5
- 3) P= 3.7 5
 4 8
 4 8

4) $P =$

5	5
5	5
6	7
6	5
5	3

5) $P =$

5	3
6	3.5
6	1.5
5	0

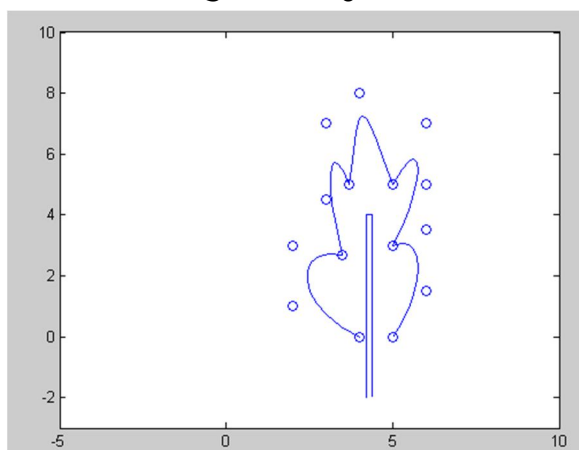


Рисунок 1.8 – Пример рисунка – дерево.

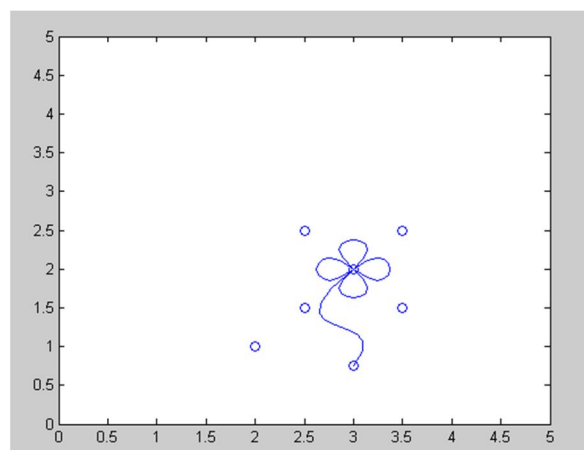


Рисунок 1.9 – Пример рисунка – цветок.

Пример 2. Построение цветка (рис. 5.9).

Точки – ориентиры для сегментов кривой Безье:

1) $P =$

3	2
2.5	2.5
2.5	1.5
3	2

2) $P =$

3	2
3.5	2.5
3.5	1.5
3	2

3) $P =$

3	2
2.5	2.5
3.5	2.5
3	2

4) $P =$

3	2
2.5	1.5
3.5	1.5
3	2

Стебель:

$$\begin{array}{rcl}
 5) \ P = & 3 & 0.75 \\
 & 3.5 & 1.5 \\
 & 2 & 1 \\
 & 3 & 2
 \end{array}$$

Для кривой Безье функция `void poit_b(float t, float *px, float *py)` может реализовывать один из трех алгоритмов: матричный, геометрический или параметрический. Для В-сплайнов – матричный алгоритм.

Задание

Написать и отладить программу на языке C/C++, используя библиотеки OpenGL (рекомендуется) или стандартную `graphics.h`, для вывода контура рисунка по варианту (Приложение А табл. А.1). Для этого:

1. Написать необходимые функции для реализации растровой развертки графических примитивов:
 - отрезок (алгоритмом Брезенхема);
 - окружность (алгоритмом Брезенхема);
 - эллипс (можно параметрический);
 - сплайновые кривые (Безье и / или В-сплайны (сплайн Эрмита по желанию)).
2. Нарисовать заданный по варианту рисунок (**только контур**).

Содержание отчета

1. Титульный лист.
2. Задание.
3. Краткие теоретические сведения.
4. Текст программы.

При защите лабораторной работы тестирование программы **обязательно!**

Лабораторная работа № 2

Тема: растровые алгоритмы заполнения фигур

Растровое заполнение круга и полигона

Основная идея – закрашивание фигуры отрезками прямых линий. Удобней использовать горизонтали. Алгоритм представляет собою цикл вдоль оси y , в ходе этого цикла выполняется поиск точек пересечения линии контура с соответствующими горизонталями.

Заполнение круга. Для заполнения круга можно использовать алгоритм вывода контура. В процессе выполнения этого алгоритма последовательно вычисляются координаты пикселей контура в границах одного октанта (например, алгоритмом Брезенхема). Для заполнения следует выводить горизонтали, которые соединяют пары точек на контуре, расположенные симметрично относительно оси y (рис. 2.1).

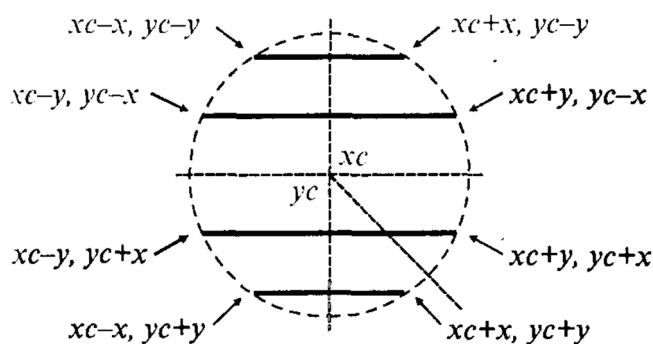


Рисунок 2.1. – Заполнение круга

Аналогично можно разработать алгоритм заполнения эллипса.

Заполнение полигонов. Контур полигона определяется вершинами, которые соединены отрезками прямых – ребрами (рис.2.2).

При нахождении точек пересечения горизонталь с контуром необходимо принимать во внимание особые точки. Если горизонталь имеет координату (y), совпадающую с координатой y_i вершины P_i тогда надлежит анализировать то, как горизонталь проходит через вершину. Если горизонталь при этом пересекает контур, как, например, в вершинах P_0 или P_4 , то в массив за-

писывается одна точка пересечения. Если горизонталь касается вершины контура (в этом случае вершина соответствует локальному минимуму или максимуму, как, например, в вершинах P_1 , P_2 , P_3 или P_5), тогда координата точки касания или не записывается, или записывается в массив два раза. Это является условием четного количества точек пересечения, хранящихся в массиве $\{x_j\}$.

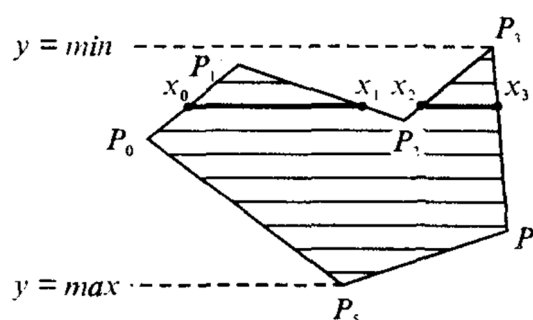


Рисунок 2.2 – Заполнение полигона

Процедура определения точек пересечения контура с горизонталью, учитывая анализ на локальный максимум, может быть достаточно сложной.

Рассмотрим, какие случаи могут возникнуть при делении многоугольника на области сканирующей строкой.

1. Простой случай. Например, на рис. 2.3 сканирующая строка $y = 4$ пересекает многоугольник при $x = 1$ и $x = 6$. Получается три области: $x < 1$; $1 \leq x \leq 6$; $x > 6$. Сканирующая строка $y = 6$ пересекает многоугольник при $x = 1$; $x = 2$; $x = 5$; $x = 6$. Получается пять областей: $x < 1$; $1 \leq x \leq 2$; $2 < x < 5$; $5 \leq x \leq 6$; $x > 6$. В этом случае x сортируется в порядке возрастания. Далее список иксов рассматривается попарно. Между парами точек пересечения закрашиваются все пиксели. Для $y = 4$ закрашиваются пиксели в интервале $(1, 6)$, для $y = 6$ закрашиваются пиксели в интервалах $(1, 2)$ и $(5, 6)$.

2. Сканирующая строка проходит через вершину (рис. 2.4). Например, по сканирующей строке $y = 3$ упорядоченный список x получится как $(2, 2, 4)$. Вершина многоугольника была учтена дважды, и поэтому закрашиваемый интервал получается неверным: $(2, 2)$. Следовательно, при пересечении вер-

шины сканирующей строкой она должна учитываться единожды. И список по x в приведенном примере будет (2, 4).

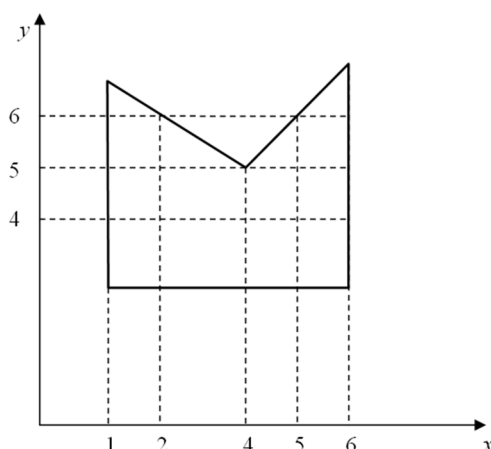


Рисунок 2.3 – Прохождение сканирующих строк по многоугольнику

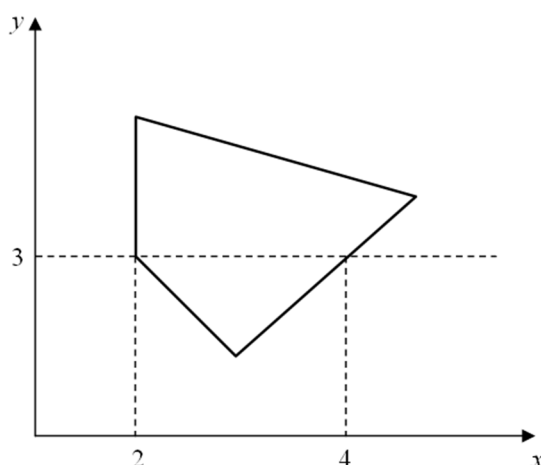


Рисунок 2.4. Прохождение сканирующей строки через вершину

3. Сканирующая строка проходит через локальный минимум или максимум (рис. 2.3 при $y = 5$). В этом случае учитываются все пересечения вершин сканирующей строкой. На рис. 2.4 при $y = 5$ формируется список x (1, 4, 4, 6). Закрашиваемые интервалы (1, 4) и (4, 6). Условие нахождения локального минимума или максимума определяется при рассмотрении концевых вершин для ребер, соединенных в вершине. Если у обоих концов координаты y больше, чем у вершины пересечения, то вершина – локальный минимум. Если меньше, то вершина пересечения – локальный максимум.

Алгоритмы закрашивания произвольных областей, заданных цветом границы

Простейший алгоритм закрашивания. Для такого алгоритма закрашивания нужно задавать начальную точку внутри контура с координатами x_0 , y_0 , от которой будет выполняться закрашка каждого соседнего пикселя рекурсивно.

Простейший рекурсивный алгоритм:

```
void PixelFill (int x, int y, unsigned char
*border_color, unsigned char *color) // рекурсивная за-
краска
{
    unsigned char p[4];
    int i,k=0,f=0;
    glColor4ub(*(color+0),*(color+1),*(color+2),*(color+3));
    glReadPixels(x, y, 1, 1, GL_RGBA, GL_UNSIGNED_BYTE,
    &p[0]);
    for(i=0;i<4;i++)
    {k++;
    if(p[i]!=*(border_color+i)) break;}
    for(i=0;i<4;i++)
    {f++;
    if(p[i]!=*(color+i)) break;}
    if((k<4)&&(f<4))
    { drawDot(x,y);
    PixelFill (x, y+1 , border_color, color);
    PixelFill (x+1, y , border_color, color);
    PixelFill (x, y-1 , border_color, color);
    PixelFill (x-1, y , border_color, color);
    }
}
```

Этот алгоритм является слишком неэффективным, так как для всякого уже отрисованного пикселя функция вызывается ещё 4 раза и, кроме того, данный алгоритм не пригоден для закрашивания контуров фигур площадью в тысячу и более пикселей, так как вложенные вызовы функций делаются для каждого пикселя, что приводит к переполнению стека в ходе выполнения программы.

Поэтому для решения задачи закрашки области предпочтительнее алгоритмы, способные обрабатывать сразу целые группы пикселей, т. е. использовать их «связность». Если данный пиксель принадлежит области, то, скорее всего, его ближайшие соседи также принадлежат данной области. Группой таких пикселей обычно выступает полоса, определяемая правым пикселем.

Алгоритм закрашивания линиями. Данный алгоритм получил широкое распространение в компьютерной графике. От простейшего алгоритма он отличается тем, что на каждом шаге закрашивания рисуется горизонтальная линия, которая размещается между пикселями контура. Алгоритм также рекурсивный, но поскольку вызов функции осуществляется для линии, а не для каждого отдельного пикселя, то количество вложенных вызовов уменьшается пропорционально длине линии. Это уменьшает нагрузку на стековую память компьютера и обеспечивает высокую скорость работы.

Задание

Написать и отладить программу на языке C/C++, используя библиотеки OpenGL (рекомендуется) или стандартную `graphics.h`, для вывода контура и закрашивания рисунка по варианту (Приложение А табл. А.1). Для этого:

1. Написать необходимые функции для реализации растровых алгоритмов:
 - закрашка окружности;
 - закрашка эллипса;
 - закрашка многоугольника;
 - закрашка произвольной области.

2. Нарисовать и закрасить объект по варианту согласно данным приложения А в табл. А.1. Не использовать рекурсивную закрашку произвольной области для закрашки круга, эллипса или полигона.



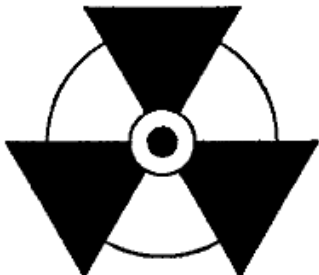
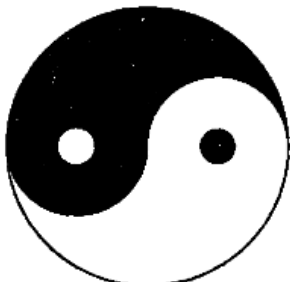






Содержание отчета






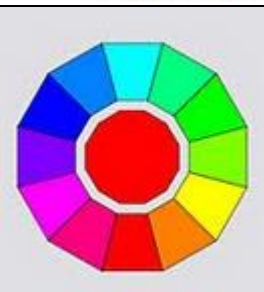
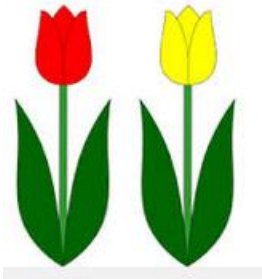

1. Титульный лист.
2. Задание.
3. Описание реализованных методов.
4. Текст программы.

При защите лабораторной работы тестирование программы **обязательно!**

ПРИЛОЖЕНИЕ А

Таблица А.1 – Варианты индивидуальных заданий

№ вар.	Рисунок	№ вар.	Рисунок
1		2	
3		4	
5		6	
7		8	
9		10	

№ вар.	Рисунок	№ вар.	Рисунок
11		12	
13		14	
15		16	
17		18	

СПИСОК ЛИТЕРАТУРЫ

1. Блинова Т.А. Компьютерная графика: учебник для вузов / Т. А. Блинова, В. Н. Порев ; Т.А. Блинова, В.Н. Порев; под ред. В.Н. Порева. - К. : Юниор; СПб. : Корона принт, 2006. - 520с. : ил. - (Учебное пособие). - ISBN 966-7323-48-X.
2. Роджерс Д.Ф. Алгоритмические основы машинной графики / Д. Ф. Роджерс; Д.Ф. Роджерс; пер. с англ.: С.А. Вичеса и др.; под ред.: Ю.М. Баяковского, В.А. Галактионова. - М. : Мир, 1989. - 503с. : ил. - Перевод изд.: Procedural elements for computer graphics/D. F. Rogers.
3. Дейтел, Х.М. Как программировать на C++ [Электронный ресурс] / Х. М. Дейтел, Дейтел П.Дж. ; Х.М. Дейтел, П.Дж. Дейтел ; пер. с англ. под ред. В.В. Тимофеева. - 5-е изд. - 19 Мб. - М. : Бином-Пресс, 2008. - 1 файл. - Перевод изд.: C++ How to Program/H.M. Deitel, P.J. Deitel.
4. OpenGL. Руководство по программированию / М. Ву [и др.] ; М.Ву, Т. Девис, Дж. Нейдер, Д. Шрайнер ; пер. с англ.: Е. Васильев, Е. Эрман. - 4-е изд. - СПб. : Питер, 2006. - 624с.: ил. - (Библиотека программиста). - Перевод изд.: OpenGL/ M. Woo, T. Davis, J. Neider, D. Shreiner. - ISBN 5-94723-827-6.
5. Шрайнер Д. OpenGL. Официальный справочник : перевод с английского / Д. Шрайнер; Д. Шрайнер ; под ред. Д. Шрейнера. - СПб.: ДиаСофтЮП, 2002. - 512с. - Перевод изд.: OpenGL Reference Manual, Third Edition/ ed. D. Shreiner. - ISBN 5-93772-048-2.
6. Тихомиров, Ю.В. OpenGL. Программирование трехмерной графики / Ю. В. Тихомиров ; Ю.В. Тихомиров. - 2-е изд. - СПб. : БХВ-Петербург, 2002. - 304с.: ил. - (Мастер программ). - ISBN 5-94157-174-7.
7. Боресков, А.В. Компьютерная графика: первое знакомство / А. В. Боресков, Г. Е. Шикин, Г. Е. Шикина ; А.В. Боресков, Е.В. Шикин, Г.Е. Шикина ; под ред. Е.В. Шикина . - М. : Финансы и статистика, 1996. - 176с. : ил. - (Диалог с компьютером).
8. Гайван А.В. Компьютерная графика и численные методы / А. В. Гайван ; А.В. Гайван. - М. : ВА Принт, 1994. - 114с. : ил.

9. Залогова Л.А. Компьютерная графика : практикум / Л. А. Залогова ; Л.А. Залогова ; науч. ред. С.В. Русаков. - 2-е изд. - М. : БИНОМ. Лаборатория знаний, 2007. - 245с.: ил. - (Элективный курс. Информатика). - ISBN 978-5-94774-656-3.

10. Корриган Д. Компьютерная графика : секреты и решения / Д. Корриган ; пер.с англ. Д.А. Куликова. - М. : Энтроп, 1995. - 352с. : ил. - Перевод изд.: Computer graphics/J. Corrigan.

11. Коцюбинский А.О. Компьютерная графика : практическое пособие / А. О. Коцюбинский, С. В. Грошев ; А.О.Коцюбинский, С.В.Грошев. - М. : ТЕХНОЛОДЖИ-3000, 2001. - 752с. : ил. - ISBN 5-94472-011-8.

12. Дегтярев, В.М. Компьютерная геометрия и графика: учебник для вузов / В. М. Дегтярев ; В.М. Дегтярев. - М. : ИЦ "Академия", 2010. - 192с. - (Высшее профессиональное образование. Информатика и вычислительная техника). - ISBN 978-5-7695-5888-7.