Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра программного обеспечения информационных технологий

А. Т. Пешков, А. С. Кобайло

КОМПЬЮТЕРНАЯ ГРАФИКА. АЛГОРИТМЫ ПОСТРОЕНИЯ И ОТСЕЧЕНИЯ ЛИНИЙ

*Рекомендовано УМО по образованию
в области информатики и радиоэлектроники
для специальности 1-40 01 01
«Программное обеспечение информационных технологий»
в качестве учебно-методического пособия*

Минск БГУИР 2014

УДК 004.925(076)
ББК 32.973.26-018.2я73
П31

Р е ц е н з е н т ы:

заведующий кафедрой информатики учреждения образования «Минский государственный высший радиотехнический колледж», кандидат технических наук, доцент Ю. А. Скудняков;

профессор кафедры менеджмента частного учреждения образования «Минский институт управления», доктор технических наук, профессор В. А. Вишняков;

доцент кафедры информатики учреждения образования «Белорусский государственный университет информатики и радиоэлектроники», кандидат физико-математических наук С. И. Сиротко;

профессор кафедры вычислительных методов и программирования учреждения образования «Белорусский государственный университет информатики и радиоэлектроники», доктор технических наук, профессор А. А. Иванюк

Пешков, А. Т.

П31 Компьютерная графика. Алгоритмы построения и отсечения линий : учеб.-метод. пособие / А. Т. Пешков, А. С. Кобайло. – Минск : БГУИР, 2014. – 48 с. : ил.

ISBN 978-985-488-943-6.

Пособие содержит материалы, связанные с классическими алгоритмами построения типовых линий и отсечения окнами.

Адресовано студентам специальности 1-40 01 01 «Программное обеспечение информационных технологий».

УДК 004.925(076)
ББК 32.973.26-018.2я73

ISBN 978-985-488-943-6

СОДЕРЖАНИЕ

Введение 4

1. Отображение пространства пользователя и машинного носителя 6

1.1. Расчет пользовательских и машинных координат 6

2. Алгоритмы генерирования линий 8

2.1. Генерирование отрезка прямой. Алгоритм Брезенхема 9

2.2. Формирование дуги окружности 17

2.3. Кривые Безье 22

3. Двумерные отсечения 29

3.1. Отсечение прямоугольным окном.

Алгоритм Сазерленда – Коуэна 29

3.2. Отсечение выпуклым многоугольником.

Алгоритм Кируса – Бэка 37

3.3. Определение выпуклости многоугольника 42

3.4. Отсечение невыпуклым многоугольником 45

Литература 47

Введение

Предметом компьютерной графики является изучение методов визуализации объектов различной природы с использованием средств компьютерной техники и методов обработки графической информации.

Компьютерная графика широко используется при решении задач автоматизации проектирования в машино- и приборостроении, при реализации проблем машинного зрения, при формировании и обработке кино- и видеоинформации и во многих других областях.

В компьютерной графике можно выделить два основных типа задач:

– ввод – вывод графической информации;
– преобразование графической информации.

Ввод – вывод графической информации осуществляется, с одной стороны, за счет восприятия с носителя графического изображения и формирования информации, представляющей собой первичное описание этого изображения, с другой стороны, – за счет фиксации на носителе оптических неоднородностей, соответствующих выходному описанию графических объектов. Задача ввода – вывода графической информации решается с помощью технических средств, представляющих собой специальные периферийные устройства, которыми оснащается компьютер.

Преобразование графической информации предполагает в общем случае обработку входного описания для извлечения из него необходимых сведений, представление этой информации в форме, удобной для ее передачи, хранения и отображения. В качестве составляющих этого преобразования можно выделить анализ и синтез графической информации.

Анализ графической информации предполагает:

– приведение ее первичного описания к форме, удобной для последующей обработки;
– представление графической информации в виде описания ее компонентов в соответствии с заданными критериями;
– подавление шумов;
– локализацию и реставрацию дефектных участков;
– преобразование масштаба и шкалы яркости;
– выделение контуров и областей;
– распознавание отдельных графических объектов и их комбинаций (сцен);
– формирование графических баз данных, позволяющих в компактной и удобной форме передавать графическую информацию и хранить ее в памяти.

К числу задач синтеза относятся:

– построение типовых линий;
– синтез фигур с использованием многоугольников;
– удаление невидимых линий и поверхностей;
– заливка областей;

– преобразование координат;
– проекции графических объектов;
– формирование теней;
– формирование выходного описания графического изображения.

Задачи преобразования графической информации решаются с помощью графических процессоров, которые могут быть реализованы или программно, или в виде специального процессора.

Последовательность обработки графического изображения иллюстрируется нижеприведенным рисунком (рис. В.1).

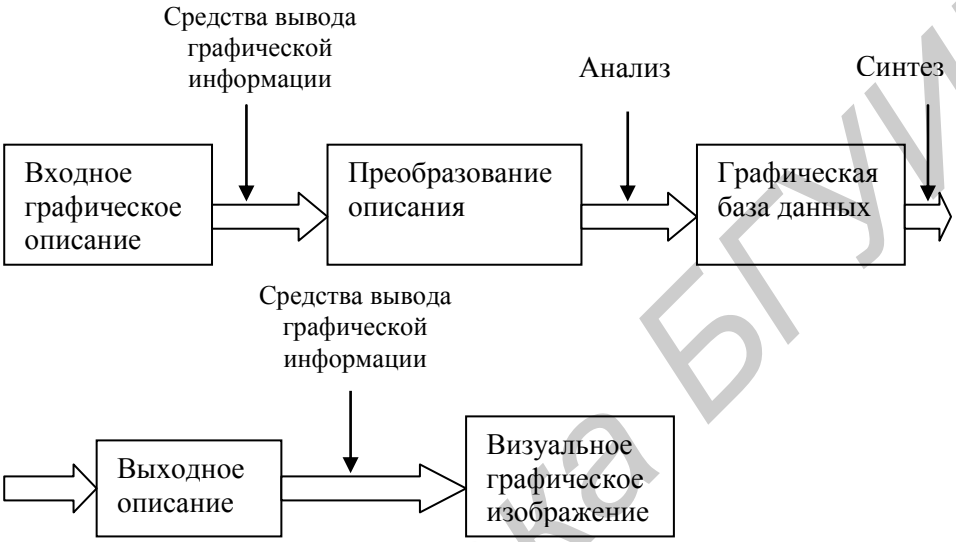


Рис. В.1

Средства ввода, формируя входное (первичное) описание, учитывают предметную область. Их разработка или выбор осуществляются с учетом особенностей считываемого графического изображения. При формировании первичного описания учитываются алгоритмы его последующей обработки. Одним из результатов анализа могут быть сведения, характеризующие объект. Иногда эти сведения являются конечным результатом обработки графической информации.

Реализация процедур синтеза, как правило, предполагает знание особенностей обрабатываемого графического изображения и особенностей технических средств, используемых для их отображения на носителе.

1. Отображение пространства пользователя и машинного носителя

При формировании изображения с использованием средств компьютерной техники весьма часто решаются две взаимосвязанные задачи:

– перенос графической информации с носителя пользователя на машинный носитель (при вводе);

– перенос графической информации с машинного носителя на носитель пользователя (при выводе).

В общем случае при решении этих задач необходимо установить соответствие между выделенным участком на носителе пользователя, с одной стороны, и заданным участком машинного носителя, с другой стороны.

Данные задачи в конечном счете сводятся к расчету координат точки на машинном носителе на основании ее координат на носителе пользователя и наоборот.

1.1. Расчет пользовательских и машинных координат

Отображаемый участок носителя пользователя может представлять собой некоторый участок общего пространства носителя, имеющий форму в виде прямоугольника (пользовательское окно) с параметрами, задающими координаты его левой нижней ($x_{пн}$, $y_{пн}$) и правой верхней ($x_{пв}$, $y_{пв}$) точками.

Машинное окно может быть также представлено через координаты левой нижней ($x_{мн}$, $y_{мн}$) и правой верхней ($x_{мв}$, $y_{мв}$) точек (рис. 1.1).

Отображение друг на друга точек двух приведенных пространств предполагает использование зависимости координат, соответствующей следующим уравнениям:

$$\frac{x_п - x_{пн}}{x_{пв} - x_{пн}} = \frac{x_м - x_{мн}}{x_{мв} - x_{мн}}; \qquad (1.1)$$

$$\frac{y_п - y_{пн}}{y_{пв} - y_{пн}} = \frac{y_м - y_{мн}}{y_{мв} - y_{мн}}. \qquad (1.2)$$

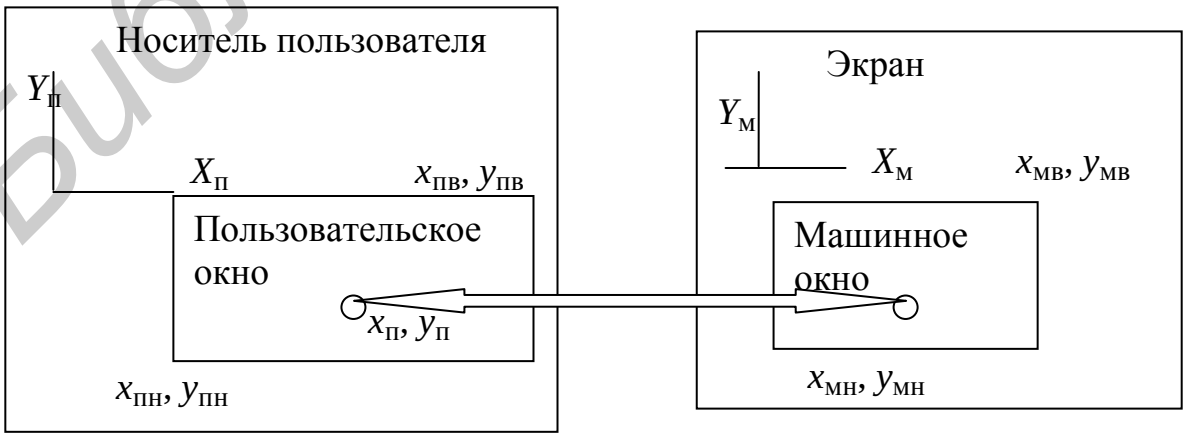


Рис. 1.1

Разрешая приведенные уравнения относительно $x_п$, $y_п$, $x_м$, $y_м$, можно получить следующие расчетные выражения.

Для случая перехода от носителя пользователя к машинному носителю имеем:

$$x_м = x_п A_х + B_х;$$

$$A_х = \frac{x_{мв} - x_{мн}}{x_{пв} - x_{пн}};\ B_x = x_{мн} - x_{пн}\frac{x_{мв} - x_{мн}}{x_{пв} - x_{пн}}; \quad (1.3)$$

$$y_п = y_м A_y + B_y;$$

$$A_y = \frac{y_{мв} - y_{мн}}{y_{пв} - y_{пн}};\ B_y = y_{мн} - y_{пн}\frac{y_{мв}\ y_{мн}}{y_{пв} - y_{пн}}. \quad (1.4)$$

A_x, A_y – масштабы по координатам x и y соответственно;

B_x, B_y – смещения по координатам x и y соответственно.

Для случая перехода от машинного носителя к носителю пользователя:

$$x_п = x_м C_х + D_х;$$

$$C_х = \frac{x_{пв} - x_{пн}}{x_{мв} - x_{мн}};\ D_x = x_{пн} - x_{мн}\frac{x_{пв} - x_{пн}}{x_{мв} - x_{мн}}; \quad (1.5)$$

$$y_п = y_м C_у + D_y;$$

$$C_y = \frac{y_{пв} - y_{пн}}{y_{мв} - y_{мн}};\ D_y = y_{пн} - y_{мн}\frac{y_{пв} - y_{пн}}{y_{мв} - y_{мн}}. \quad (1.6)$$

C_x, C_y – масштабы по координатам x и y соответственно;

D_x, D_y – смещения по координатам x и y соответственно.

Задача отображения координат может быть усложнена дополнительными требованиями. При отображении на машинный носитель такими требованиями могут быть:

- предусмотреть кромку по краям экрана (h);
- обеспечить максимальное использование выделенной площади экрана;
- использовать одинаковые масштабы по координатным осям X, Y;
- изображение поместить по центру выделенного окна.

Создание кромки по периферии экрана обеспечивается за счет коррекции параметров экранного окна:

$$x'_{мн} = x_{мн} + h;\ y'_{мн} = y_н + h;$$

$$x'_{мв} = x_{мв} - h;\ y'_{мв} = y_{мв} - h.$$

Эффективное использование выделенной площади окна на экране может быть обеспечено за счет того, что вместо использования габаритов окна в пользовательском пространстве следует использовать габариты имеющегося в этом окне графического изображения, т. е. его экстремальные значения по координа-

там x и y, соответственно x_{max}, x_{min} и y_{max}, y_{min}. В этом случае масштабы по осям координат рассчитываются следующим образом:

$$M_x = \frac{x'_{мв} - x'_{мн}}{x_{max} - x_{min}}; \quad M_y = \frac{y'_{мв} - y'_{мн}}{y_{max} - y_{min}}.$$

Для удовлетворения требования обеспечения одинакового масштаба по координатным осям необходимо этот масштаб M взять как

$M = \min(M_x, M_y)$.

Обеспечение центровки изображения на экране предполагает выполнение следующих вычислений:

– рассчитывается положение центра машинного окна $y_{мц}$, $x_{мц}$:

$$y_{мц} = \frac{y'_{мв} + y'_{мн}}{2}; \quad x_{мц} = \frac{x'_{мв} + x'_{мн}}{2};$$

– рассчитывается положение центров изображения $y_{иц}$, $x_{иц}$:

$$y_{иц} = \frac{y_{max} + y_{min}}{2}; \quad x_{иц} = \frac{x_{max} + x_{min}}{2};$$

– рассчитывается отклонение полученных центров:

$$\delta_{xц} = x_{мц} - Mx_{иц}; \quad \delta_{yц} = y_{мц} - My_{иц}.$$

С учетом полученных отклонений машинные координаты рассчитываются как

$$x_м = Mx_n + C_x + \delta_{xц};$$
$$y_м = My_п + C_y + \delta_{yц}.$$

2. Алгоритмы генерирования линий

При формировании любой линии предъявляются следующие очевидные требования:

– линия должна начинаться и заканчиваться в заданных точках;

– контрастность и яркость вдоль всей длины формируемой линии должны быть одинаковыми и не зависеть от крутизны и длины линии;

– необходимо отражать характер линии (отрезки прямой должны быть прямыми, формируемые дуги должны иметь форму дуги и т. д.);

– формирование отдельных точек должно выполняться с минимальными затратами времени, т. к. количество точек обычно достаточно велико.

Не все перечисленные требования могут быть выполнены точно.

Из-за принципиальных особенностей средств графического ввода – вывода носитель, на котором формируется или с которого считывается графическая информация, рассматривается в виде матрицы дискретных элементов, называемых пикселами, визуальные параметры (яркость, цветность и т. п.) ко-

торых могут задаваться. Поэтому формируемые линии не могут начинаться и заканчиваться точно в математически определенных точках носителя.

Ошибка положения формируемых точек определяется уровнем дискретизации или количеством дискретных точек, на которое разбивается носитель. Таким образом, максимальная погрешность положения точки определяется шагом между двумя соседними пикселами.

По этой же причине даже отображение отрезка прямой линии не является прямой, как это показано на рис. 2.1. Линия представляется набором горизонтальных (или вертикальных) цепочек пикселов, которые с учетом шага дискретизации наилучшим образом отражают характер генерируемой прямой линии.

В общем случае, если не принимать специальные меры, яркость линии на отдельных участках будет зависеть от крутизны этих участков.

Удовлетворение требования минимальных затрат времени на расчет координат одной точки формируемой линии обеспечивается за счет использования приближенных методов вычислений, применения целочисленной арифметики, использования специальных алгоритмов, реализуемых программно или на специальных графических процессорах.

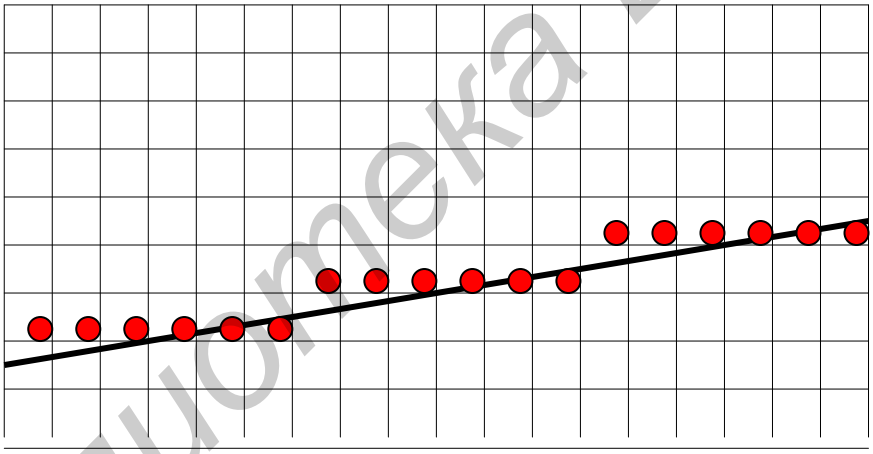


Рис. 2.1

2.1. Генерирование отрезка прямой. Алгоритм Брезенхема

Генерирование отрезка прямой осуществляется точка за точкой, начиная от заданной начальной точки. При формировании очередной точки используется характер изменения ее координат, определяемый положением конечной точки относительно начальной точки. Формируемые координаты очередной точки определяются за счет модификации координат предыдущей точки. Одна координата используется как аргумент, и ее модификация всегда одинакова и равна единице. Изменение второй координаты осуществляется таким образом, чтобы очередная точка имела положение, наиболее близкое по отношению к положению, рассчитанному на основании аналитической зависимости.

Пусть отрезок прямой задается начальной точкой Т1 с координатами x_1, y_1 и конечной точкой Т2 с координатами x_2, y_2.

Исходя из заданных концевых точек, рассчитывается крутизна отрезка:

$$m = \Delta y/\Delta x,$$

где $\Delta x = x_2 - x_1$; $\Delta y = y_2 - y_1$.

Рассмотрим простейший случай, когда $\Delta x >= 0$, $\Delta y >= 0$, $\Delta x >= \Delta y$, и для этого случая разработаем алгоритм формирования отрезка прямой.

Возьмем в качестве аргумента x, а в качестве функции y.

Учитывая характер рассматриваемого случая, можно утверждать, что претендентами на роль следующей за последней найденной точкой с координатами x_i, y_i будут точки-соседи, имеющие координаты x, отличающиеся от координаты x последней найденной точки на «+1», а координата y одного из двух претендентов равна координате y найденной точки, а другого – с приращением «+1» (рис. 2.2).

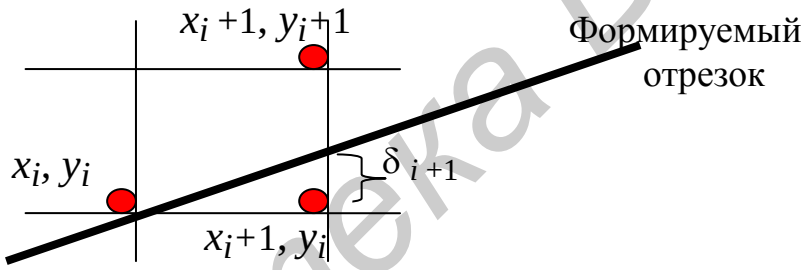


Рис. 2.2

В качестве следующей точки из двух претендентов выбирается тот, который ближе располагается к расчетному положению.

Для очередной формируемой (i +1)-й точки ее отклонение δ'_{i+1} от расчетной точки с учетом только наклона отрезка определяется следующим образом (см. рис. 2.1):

$\delta'_{i+1} = \delta_i + m$.

Отсюда можно записать:

$$x_{i+1} = x_i + 1;$$

$$y_{i+1} = \begin{cases} y_i + 1, & \text{если } \delta'_{i+1} \geq 1/2; \\ y_i, & \text{если } \delta'_{i+1} < 1/2. \end{cases}$$

Процедура выбора следующей точки иллюстрируется нижеприведенными рисунками.

На рис. 2.3 изображена последовательность пикселов, с помощью которых отображается формируемый отрезок.

На рис. 2.4 приведена кривая, соответствующая погрешности (отклонению от рассчетного положения) формирования отрезка на каждой формируемой точке. При этом расчет окончательной погрешности положения найденной (i+1)-й точки δ_{i+1} выполняется по следующему правилу:

если $y_{i+1} = y_i$, то $\delta_{i+1} = \delta'_{i+1}$;
если $y_{i+1} = y_i + 1$, то $\delta_{i+1} = \delta'_{i+1} - 1$.

Для того чтобы избавиться от сравнения отклонения с дробным числом 1/2, вводится постоянное смещение отклонения, равное –1/2. В этом случае вид кривой погрешности приведен на рис. 2.5, а процесс выбора следующей точки из двух претендентов происходит по правилу

$$y_{i+1} = \begin{cases} y_i + 1, & \text{если } \delta'_{i+1} \geq 0; \\ y_i, & \text{если } \delta'_{i+1} < 0. \end{cases}$$
$$x_{i+1} = x_i + 1.$$

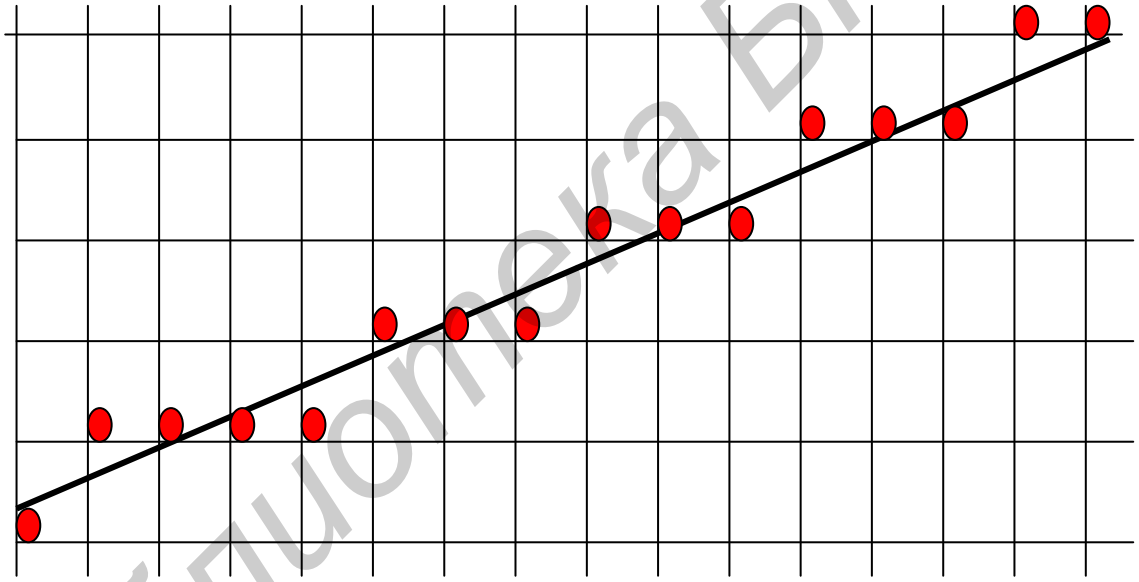


Рис. 2.3

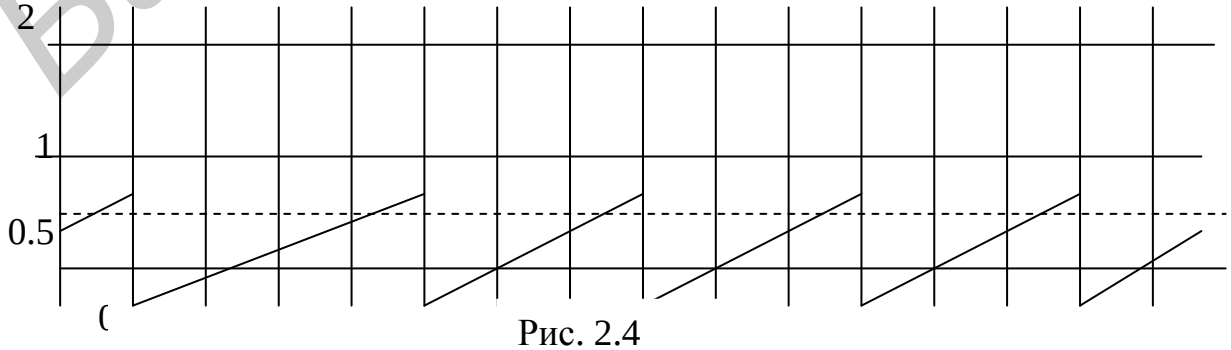


Рис. 2.4

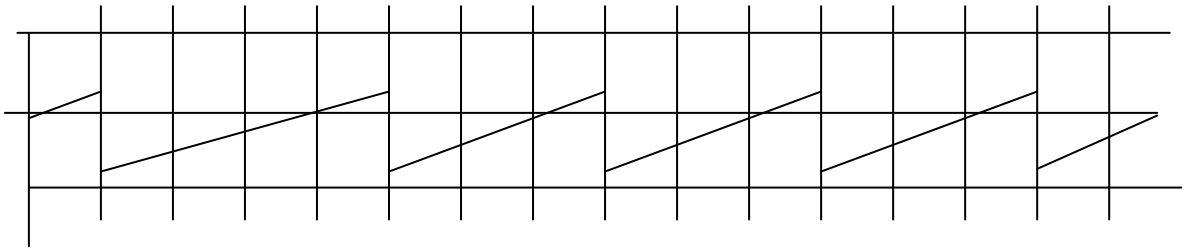


Рис. 2.5

Вышеизложенное позволяет представить алгоритм формирования отрезка прямой для рассматриваемого частного случая в виде граф-схемы, приведенной на рис. 2.6.

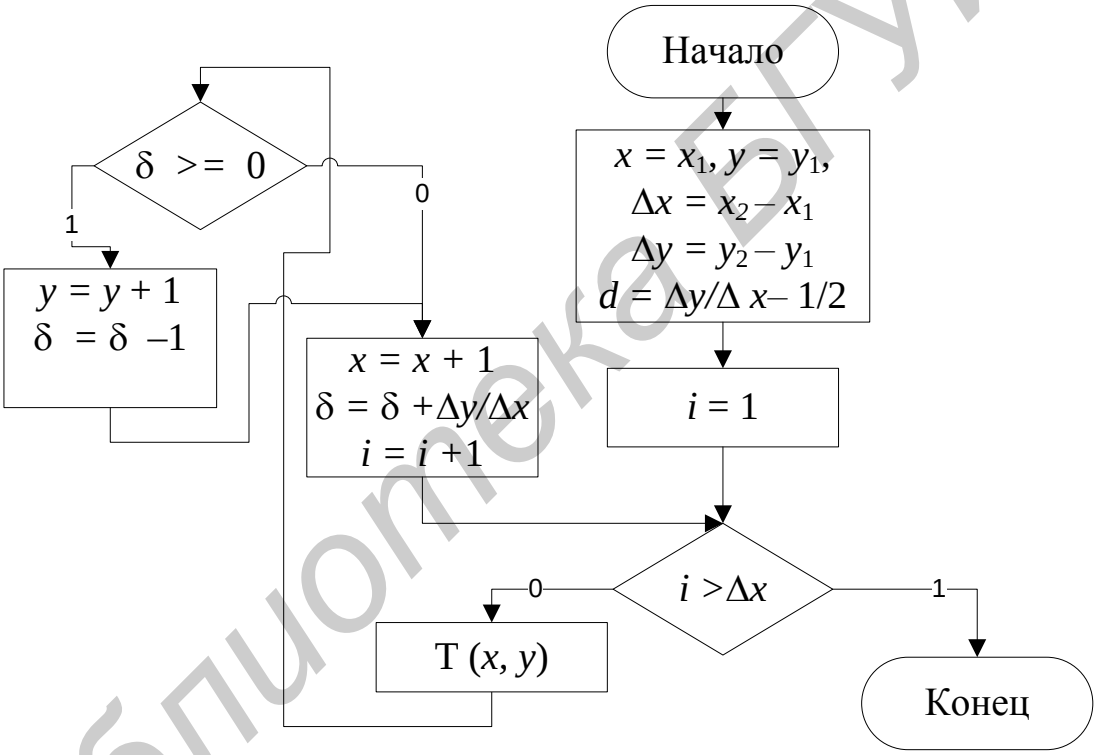


Рис. 2.6

Здесь T (*x, y*) обозначает процедуру отображения на носителе точки с координатами *x, y*.

Для того чтобы избавиться от длинных операций и действительных величин, введем замену: $e = 2\delta\Delta x$ (рис. 2.7).

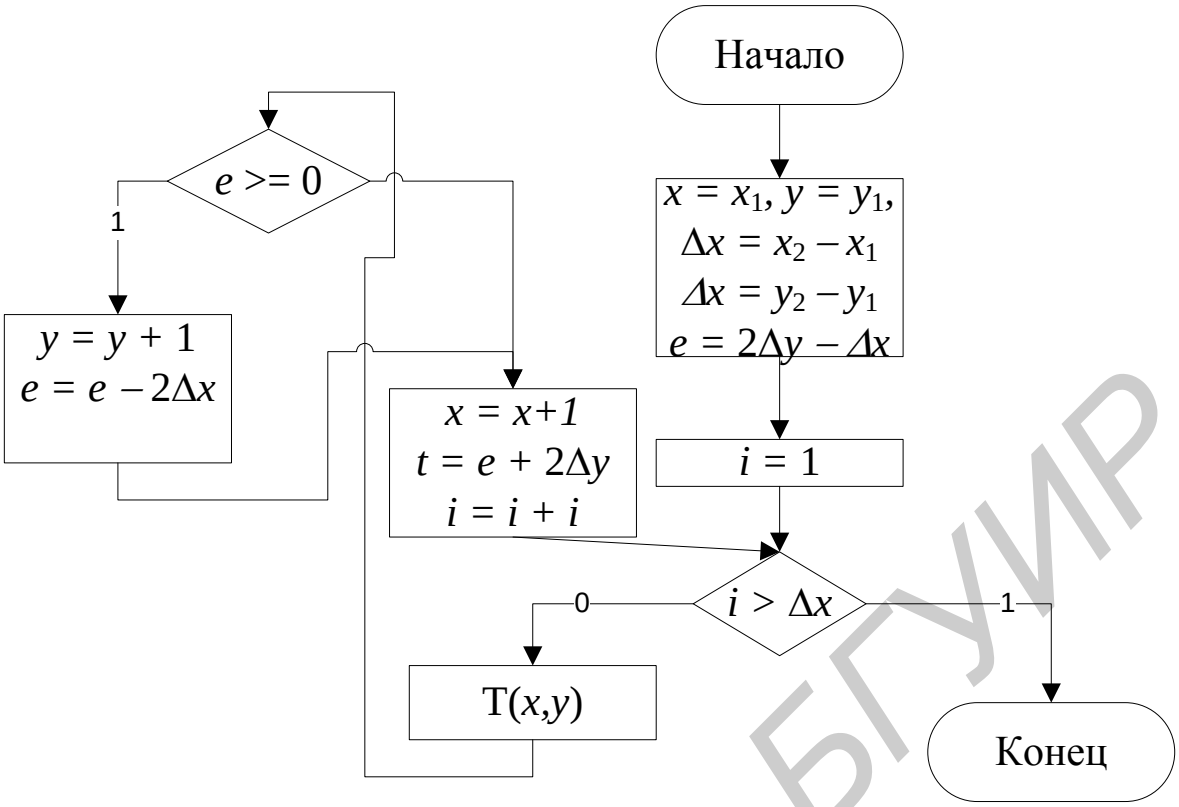


Рис. 2.7

В приведенном алгоритме анализируется не величина переменной δ, а только ее знак. Учитывая то, что в рассматриваемом частном случае положения формируемого отрезка приращение Δx всегда положительно, вводимая переменная *e* будет иметь тот же знак, что и δ. Поэтому в алгоритме на рис. 2.6 анализ переменной δ можно заменить анализом переменной *e*.

С учетом этой замены граф-схема алгоритма будет иметь вид, приведенный на рис. 2.7.

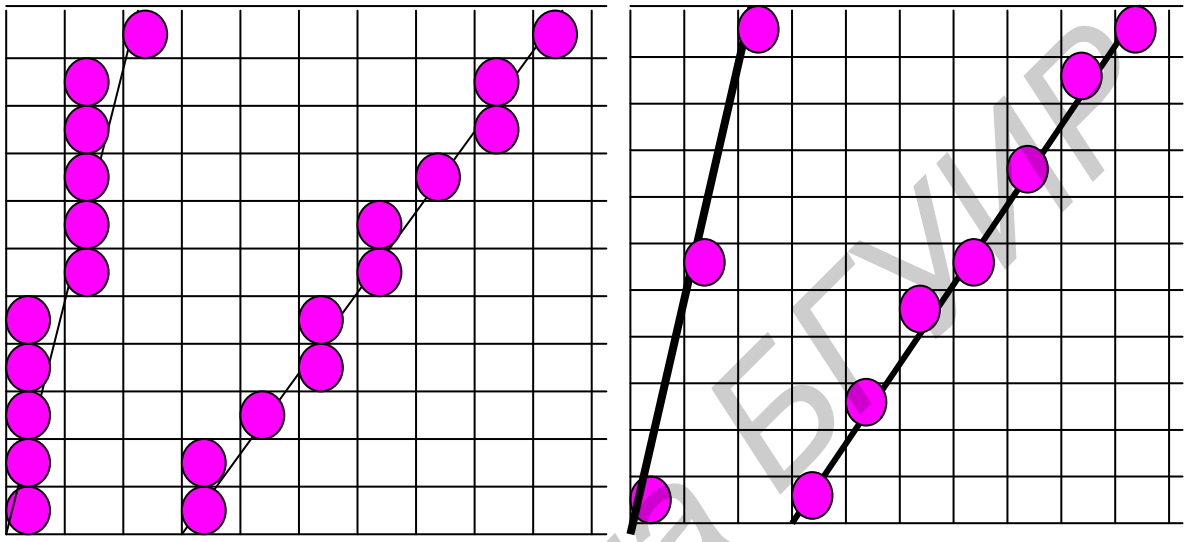
Введенная замена приводит к тому, что в приведенном на рис. 2.7 алгоритме отсутствуют длинные операции. Кроме того, все операнды являются целыми числами, а следовательно, вычисления будут выполняться с использованием чисел с фиксированной точкой, что потребует меньших затрат времени, чем в случае действительных операндов, которые должны представляться в форме с плавающей точкой.

Для того чтобы распространить приведенный алгоритм на общий случай положения отрезка, проанализируем особенности рассмотренного частного случая положения отрезка и разработаем такую оболочку для рассмотренного алгоритма, которая позволит адаптировать его ко всем возможным ситуациям общего случая положения отрезка.

Для общего положения формируемого отрезка на плоскости приращения аргумента и функции могут быть как на «+1», так и на «–1».

Кроме того, крутизна $m = \Delta y/\Delta x$ может быть больше или меньше единицы.

На рис. 2.8 приведено формирование из пикселов отрезков, крутизна которых больше единицы. На рис. 2.8, б в качестве аргумента используется координата *x*, на рис. 2.8, а – координата *y*. На рис. 2.8, а расстояния между всеми точками формируемого отрезка соответствует расстоянию между пикселами-соседями, и поэтому яркость (плотность пикселов на единицу длины отрезка) для всех отрезков, независимо от значения их крутизны, одинаковая.



а б

Рис. 2.8

Из рис. 2.8, б видно, что расстояния между пикселами-соседями тем больше, чем больше крутизна формируемого отрезка, и как следствие этого, яркость отображения отрезка будет тем меньше, чем больше крутизна отрезка, что крайне нежелательно. Поэтому в случаях, когда крутизна формируемого отрезка больше единицы, в качестве аргумента необходимо использовать координату *y*.

Все возможные значения крутизны и приращений координат приведены на рис. 2.9.

Исходя из приведенного рисунка, чтобы использовать рассмотренный алгоритм формирования отрезка прямой во всех случаях, необходимо снабдить его оболочкой, которая должна:

– выбирать в качестве аргумента координату *x* тогда, когда положение формируемого отрезка соответствует октантам 1, 4, 5, 8 (см. рис. 2.9);

– выбирать в качестве аргумента координату *y* тогда, когда положение формируемого отрезка соответствует октантам 2, 3, 6, 7 (см. рис. 2.9);

– модифицировать координату *y* на «+1» в октантах 1, 2, 3, 4;

– модифицировать координату *y* на «–1» в октантах 5, 6, 7, 8;

– модифицировать координату *x* на «+1» в октантах 1, 2, 7, 8;

– модифицировать координату *x* на «–1» в октантах 4, 5, 6, 7, 8.

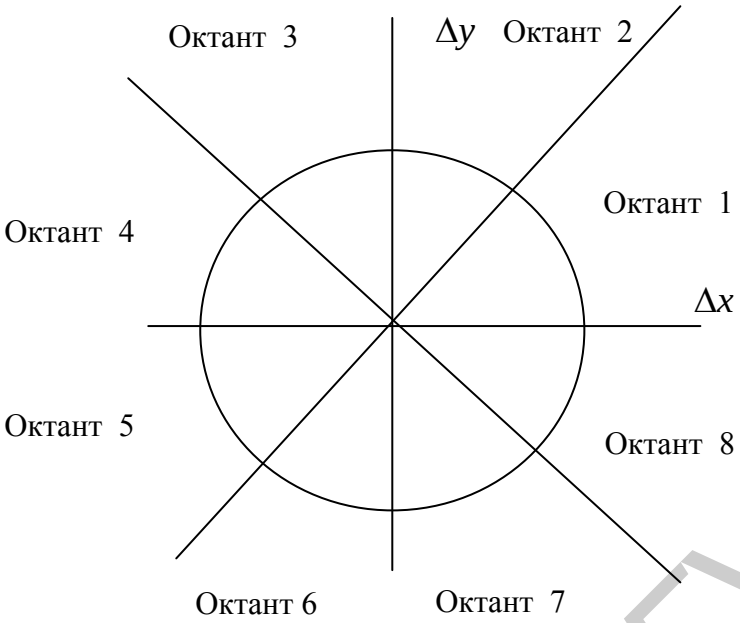


Рис. 2.9

Для построения алгоритма для общего случая положения отрезка введем переменную-флажок *C*, которой будем приписывать значение «0», если в качестве аргумента выбирается *x*, и «1», если в качестве аргумента выбирается *y*. Для обеспечения требуемой модификации аргумента и функции (на «+1» или на «–1») определим функцию знака sign(z), которая принимает следующие значения:

$$\text{sign}(z) = \begin{cases} 1, & \text{если } z \geq 0; \\ -1, & \text{если } z < 0. \end{cases}$$

В этом случае граф-схема алгоритма формирования отрезка прямой (алгоритма Брезенхема) будет иметь вид, приведенный на рис. 2.10. Здесь используются обозначения, принятые на рис. 2.7. Кроме того, введены дополнительные обозначения:

- S_x – знак приращения *x* от начальной к конечной точке отрезка;
- S_y – знак приращения *y* от начальной к конечной точке отрезка.

Отдельные фрагменты приведенного алгоритма выполняют следующие действия.

Блок операторов 1 обеспечивает установку начальных значений для переменных, используемых в рассматриваемом алгоритме.

Блок операторов 2 обеспечивает в зависимости от заданного положения отрезка выбор в качестве аргумента координаты *x* или координаты *y*. Кроме того, в этом блоке устанавливается соответствующее значение флага *C*.

Блок операторов 3 обеспечивает анализ завершения формирования отрезка. Отрезок считается сформированным, если значение параметра цикла *i* превысило число точек, составляющих отрезок. Количество точек равно Δ*x*, ко-

торое в данной точке алгоритма равно количеству аргументов (по координате *x* или *y* в зависимости от наклона отрезка).

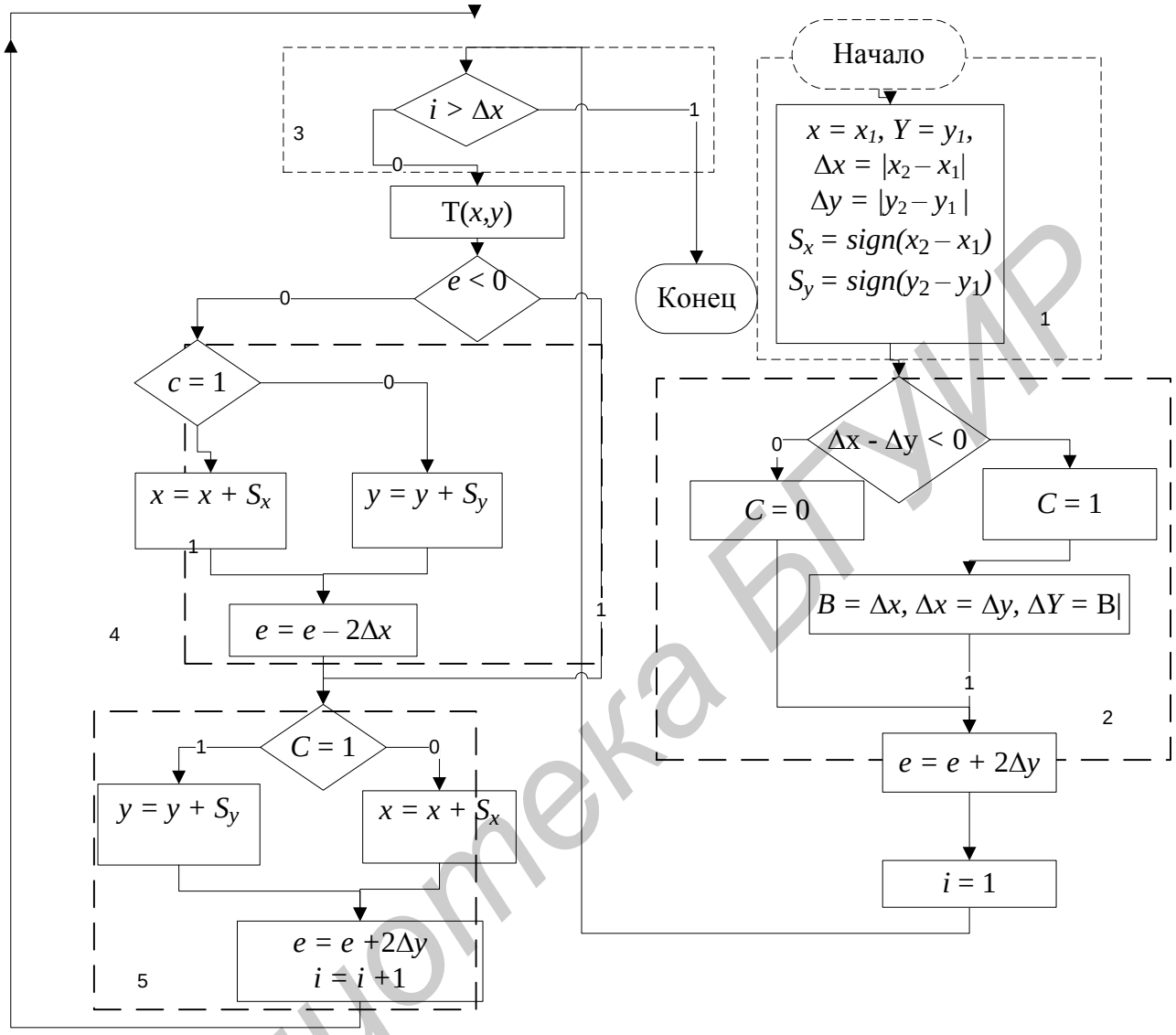


Рис. 2.10

Блок операторов 4 обеспечивает определение, какая координата (*x* или *y)* используется в качестве функции, и модифицирует эту координату.

Блок операторов 5 обеспечивает определение, какая координата (*x* или *y*) используется в качестве аргумента, и модифицирует эту координату.

Отличительной особенностью данного алгоритма является то, что при формировании координат отдельных точек, из которых образуется заданный отрезок, используются только простейшие операции (операции сложения и логические операции). Кроме того, действия выполняются над целочисленными операндами, что позволяет использовать арифметику с фиксированной точкой. Все это свидетельствует о том, что при реализации данного алгоритма имеют место минимальные затраты времени.

2.2. Формирование дуги окружности

Данный алгоритм обеспечивает формирование дуги окружности с центром в начале координат, расположенной в первом квадранте (рис. 2.11). Используя зеркальное отображение относительно координатной оси, можно на основании дуги, сформированной для первого квадранта, получить изображение окружности во всех остальных квадрантах.

При формировании дуги окружности пиксел, который должен быть использован в качестве следующей точки формируемой окружности, выбирается из возможных претендентов (соседних точек), и его координаты рассчитываются за счет единичного приращения координат предыдущей найденной точки.

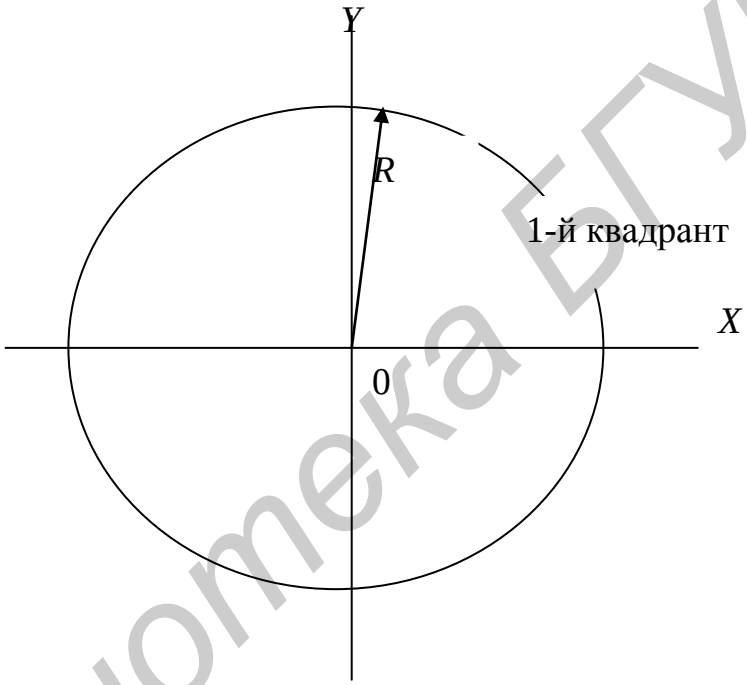


Рис. 2.11

При формировании дуги окружности, расположенной в первом квадранте, точками-претендентами являются три точки: горизонтальная, диагональная и вертикальная, из которых выбирается та, которая ближе всего находится от расчетной окружности.

На рис. 2.12 приведены последняя найденная точка T_i с координатами x_i, y_i , горизонтальный претендент $П_H$ с координатами x_i +1, y_i , диагональный претендент $П_D$ с координатами x_i +1, y_i –1 и вертикальный претендент $П_V$ с координатами x_i, y_i –1. Кроме того, на этом рисунке показано пять возможных положений расчетной окружности по отношению к последней найденной точке и трем претендентам.

.

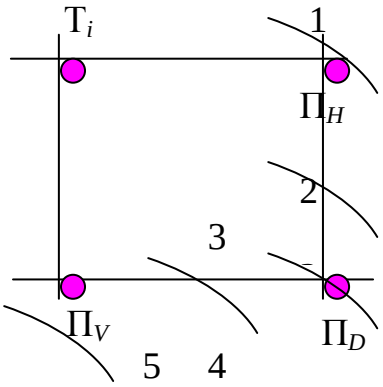


Рис. 2.12

В качестве критерия выбора одного из претендентов используется квадрат расстояния точек-претендентов $П_H$, $П_D$, и $П_V$ до окружности, соответственно m_H, m_D, m_v. Значения этих квадратов расстояний, которые называются отклонениями, рассчитываются по следующим формулам:

$m_H = |(x_i+1)^2 + y_i^2 - R^2|$;

$m_D = |(x_i+1)^2 + (y_i-1)^2 - R^2|$;

$m_V = |x_i^2 + (y_i^2 - 1) - R^2|$,

где через R обозначен радиус окружности.

Принятие решения о выборе одного из трех претендентов в качестве следующей точки формируемой дуги осуществляется следующим образом.

Для текущей i-й точки вводится понятие базовая погрешность Δ_i , в качестве которого используется отклонение от диагонального претендента, рассчитываемое как

$\Delta_i = (x_i+1)^2 + (y_i-1)^2 - R^2$.

Логика выбора одного из трех претендентов в качестве очередной точки формируемой дуги представлена на следующем рисунке (рис. 2.13), где приняты следующие обозначения:

Δ_ι – базовое отклонение;

δ_1 – разность расстояний горизонтального и диагонального претендентов от окружности;

δ_2 – разность расстояний диагонального и вертикального претендентов до окружности.

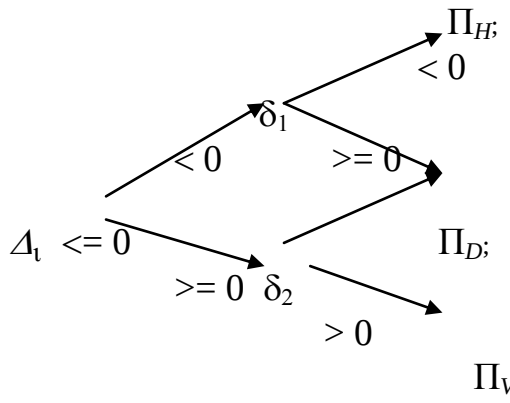


Рис. 2.13

Найдем компактные аналитические выражения для расчета δ_1 и δ_2, используя для их расчета базовое отклонение Δ_i.

Разность δ_1 расстояний горизонтального и диагонального претендентов до окружности рассчитывается только в случае, когда выполняется условие

$$\Delta_i < 0 , \qquad (2.1)$$

при выполнении которого может иметь место расчетное положение окружности 1, 2, 5.

В этом случае δ_1 формируется за счет вычитания из абсолютного значения расстояния h_1 горизонтального претендента до окружности абсолютного значения расстояния h_2 диагонального претендента до окружности, что можно записать следующим образом:

$$\delta_1 = h_1 - h_2 = |(x_i + 1)^2 + y_i^2 - R^2| - |(x_i + 1)^2 + (y_i - 1)^2 - R^2|. \qquad (2.2)$$

Для положения окружности между горизонтальным и диагональным претендентами (положение 1 или 5 на рис. 2.12) h_1 – положительное, а h_2 – отрицательное, что позволяет перейти от выражения (2.2) к выражению

$$\delta_1 = (x_i + 1)^2 + y_i^2 - R^2 + (x_i + 1)^2 + (y_i - 1)^2 - R^2.$$

Выразим δ_1 через базовую погрешность Δ_i:

$$\delta_1 = (x_i + 1)^2 + y_i^2 - R^2 + (x_i + 1)^2 + (y_i - 1)^2 - R^2 - \underline{2y_i + 1 + 2y_i - 1} =$$
$$= (x_i + 1)^2 + y_i^2 - \underline{2y_i + 1} + R^2 + (x_i + 1)^2 + (y_i - 1)^2 - R^2 \underline{+ 2y_i - 1} =$$
$$= 2\Delta_i + 2y_i - 1,$$

что позволяет использовать для расчета δ_1 сравнительно простое выражение:

$$\delta_1 = 2\Delta_i + 2y_i - 1. \qquad (2.3)$$

Таким образом, если при выполнении условия (2.1) расчетное положение окружности соответствует положению 1 или 5, то при отрицательном значении δ_1 в качестве очередной точки выбирается горизонтальный претендент. Заметим, что при расчетном положении окружности, соответствующем положению 2, δ_1 тем более будет отрицательным (т. к. и величина h_1 , и величина h_2 отрицательные), что и при расчетном положении 2 окружности также обусловит выбор горизонтального претендента в качестве следующей точки формируемой дуги.

Разность δ_2 расстояний диагонального и вертикального претендентов до окружности рассчитывается только в случае, когда выполняется условие $\Delta_i > 0$, при выполнении которого может иметь место расчетное положение окружности 5, 3, 4.

В этом случае δ_2 формируется за счет вычитания из абсолютного значения расстояния h_2 диагонального претендента до окружности абсолютного значения расстояния h_3 вертикального претендента до окружности, что можно записать как

$$\delta_2 = h_2 - h_3 = |(x_i + 1)^2 + (y_i - 1)^2 - R^2| - |x_i^2 + (y_i - 1)^2 - R^2|. \qquad (2.4)$$

Для положения окружности между вертикальным и диагональным претендентами (положение 3 или 5 на рис. 2.13), h_2 положительное, а h_3 отрицательное, что позволяет перейти от выражения (2.3) к выражению

$$\delta_2 = h_2 - h_3 = (x_i+1)^2 + (y_i - 1)^2 - R^2 + x_i^2 + (y_i - 1)^2 - R^2.$$

Выразим δ_2 через базовую погрешность Δ_i:

$$\delta_2 = (x_i+1)^2 + (y_i - 1)^2 - R^2 - x_i^2 + (y_i - 1)^2 - R^2 + \underline{2y_i + 1 - 2y_i} - \underline{1} =$$
$$= (x_i+1)^2 + (y_i - 1)^2 - R^2 - x_i^2 + (y_i - 1)^2 - R^2 - \underline{2x_i} - 1 = 2x_i - 2y_i - 1,$$

т. е. для расчета δ_2 можно использовать выражение

$$\delta_2 = 2\Delta_i - 2x_i - 1. \qquad (2.5)$$

Таким образом, если при невыполнении условия (2.1) расчетное положение окружности соответствует положению 3 или 5, то при положительном значении δ_2 в качестве очередной точки выбирается вертикальный претендент. Заметим, что при расчетном положении окружности, соответствующем положению 4, δ_2 тем более будет положительным, что также обусловит выбор вертикального претендента.

Начальной точкой формируемой дуги является точка с координатами $x_н = 0$, $y_н = R$. Для этой точки базовое отклонение $\Delta_н$ определяется как

$$\Delta_н = (x_н + 1)^2 + (y_н - 1)^2 - R^2 = 2(1 - R).$$

Расчет по результату выбора претендента координат очередной точки формируемой дуги и соответствующего ей базового отклонения выполняется следующим образом.

Если в качестве очередной точки выбран *горизонтальный претендент* $П_H$, тогда

$$x_{i+1} = x_i + 1;\ y_{i+1} = y_i;$$

$$\Delta_{i+1} = (x_{i+1} + 1)^2 + (y_{i+1} - 1)^2 - R^2 = x_{i+1}{}^2 + 2x_{i+1} + 1 + (y_i - 1)^2 - R^2 =$$
$$= (x_i + 1)^2 + (y_i - 1)^2 - R^2 + 2x_{i+1} + 1 = \Delta_i + 2x_{i+1} + 1.$$

Если в качестве очередной точки выбран *вертикальный претендент* $П_V$, тогда

$$x_{i+1} = x_i;\ y_{i+1} = y_i - 1;$$
$$\Delta_{i+1} = (x_{i+1} + 1)^2 + (y_{i+1} - 1)^2 - R^2 = (x_i + 1)^2 + y_{i+1}{}^2 - 2y_{i+1} + 1 - R^2 =$$
$$= (x_i + 1)^2 + (y_i - 1)^2 - R^2 - 2y_{i+1} + 1 = \Delta_i - 2y_{i+1} + 1.$$

Если в качестве очередной точки выбран *диагональный претендент* $П_D$, тогда

$$x_{i+1} = x_i + 1;\ y_{i+1} = y_i - 1;$$
$$\Delta_{i+1} = (x_{i+1} + 1)^2 + (y_{i+1} - 1)^2 - R^2 = x_{i+1}{}^2 + 2x_{i+1} + y_{i+1}{}^2 - 2y_{i+1} + 1 -$$
$$- R^2 = (x_i + 1)^2 + (y_i - 1)^2 - R^2 + 2x_{i+1} + 1 - 2y_{i+1} + 1 = \Delta_i + 2x_{i+1} - 2y_{i+1} + 2.$$

Учитывая изложенное, алгоритм формирования дуги окружности можно представит в виде граф-схемы, приведенной на следующем рисунке (рис. 2.14).

Так как при формировании очередной точки рассчитывается и используется или δ_1, или δ_2 и никогда оба вместе, для них в алгоритме используется одна и та же переменная δ.

Оператор 1 данной граф-схемы алгоритма обеспечивает установку исходных значений координат текущей точки формируемой в первом квадранте дуги и соответствующего ей базового отклонения для случая, когда начало дуги располагается на координатной оси Y ($x_н = 0$, $y_н = 1$).

Оператор 2 осуществляет фиксацию на точки с координатами x, y.

Операторы 4 и 5 осуществляют определение разности отклонений точек претендентов от расчетного положения точки окружности.

Операторы 8–10 рассчитывают координаты очередной выбранной точки формируемой окружности и соответствующее ей базовое отклонение.

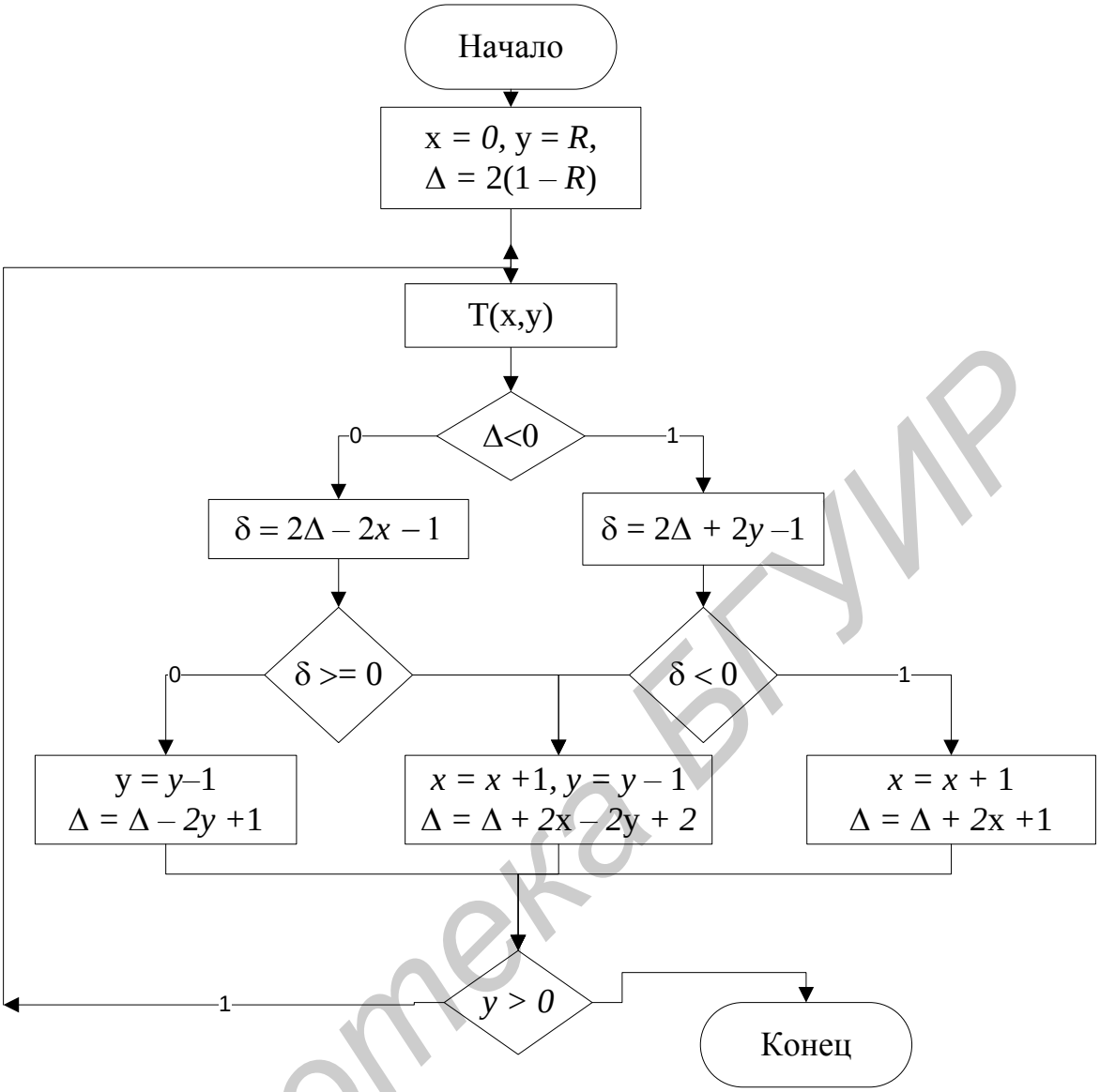


Рис. 2.14

Особенностью данного алгоритма является отсутствие при его выполнении длинных операций, что способствует уменьшению затрат времени на его реализацию.

2.3. Кривые Безье

Кривые Безье применяются для приближенного решения задачи отображения кривых, задаваемых с помощью множества точек. При этом вместо «жесткого» задания точек, определяющих формируемую кривую, пользователь в интерактивном режиме подбирает такой набор опорных точек-ориентиров, который при построении кривой Безье дает форму, наиболее соответствующую требуемой.

Кривые Безье строятся на основании так называемого многочлена Безье.

Многочлен Безье задается в параметрической форме через параметр. При этом связь параметра *t* с декартовыми координатами задается как

$x = P_x(t);\ y = P_y(t),$

где параметр *t* изменяется в диапазоне 0 < = *t*< = 1.

Если заданы точки-ориентиры

$x_0, y_0;\ x_1, y_1\ ; \ldots x_i, y_i;\ \ldots x_m, y_m,$

то многочлен Безье представляется в виде

$$\vec{P}_x(t) = \sum_{i=0}^{m} C_m^i t^i (1-t)^{m-i} \vec{P}_i,$$

$$\vec{P}_y(t) = \sum_{i=0}^{m} C_m^i t^i (1-t)^{m-i} y_i,$$

где 0<= *t* <= 1;

В векторной форме многочлен Безье задается как

$$\vec{P}_x(t) = \sum_{i=0}^{m} C_m^i t^i (1-t)^{m-i} \vec{P}_i, \qquad (2.6)$$

где $\vec{P}_i$ – *i*-я опорная точка с координатами x_i, y_i.

Для анализа особенностей этого выражения, вынесем из-под знака суммы слагаемые со значением $i = 0$ и $i = m$, представив многочлен Безье в виде

$$\vec{P}_x(t) = \sum_{i=0}^{m} C_m^i t^i (1-t)^{m-i} \vec{P}_i.$$

Если учесть, что 0! = 1, $0^0 = 1$ и $C_m^{\ 0} = C_m^{\ m} = 1$, то будем иметь

$$\vec{P}(t) = (1-t)^m \vec{P}_0 + \sum_{i=i}^{m-1} C_m^i t^i (1-t)^{m-i} \vec{P}_i + t^m \vec{P}_m.$$

Найдем значения для *P* при *t* = 0 и *t* = 1:
Подставив в полученное выражение значение *t* = 1 и *t* = 0, получим

$\vec{P}(0) = \overrightarrow{P_0}$ и $\vec{P}(1) = \overrightarrow{P_m}$.

Откуда следует, что кривая Безье начинается в начальной опорной точке P_0 и заканчивается в конечной опорной точке P_m.

Оценим поведение многочлена Безье в окрестностях точек $t = 0$ и $t = 1$. Для этого разложим функцию $f(x)$ в окрестности точки $x = a$ в ряд Тейлора:

$$f(x) = f(a) + ((x - a)/1!)f^{(1)}(a) + ((x - a)^{(2)}/2!)f^{(2)}(a) + ((x - a)/3!)f^{(3)}(a) + \\ + \ldots + R(x^{n+1})$$

где $f^{(i)}(a)$ – i-я производная функции $f(x)$ в точке $x = a$;

$R(x^{n+1})$ – остаточный член, определяющий погрешность порядка n-й степени x.

Ограничившись погрешностью второй степени, для многочлена Безье в окрестности точки $t = 0$ будем иметь

$$P(t = 0) = P(0) + tP^{(1)}(0) + R(t^2) \cong P(0) + P^{(1)}(0). \qquad (2.7)$$

Так как параметр t, оставаясь меньше единицы, стремится к нулю, достаточно остановиться на погрешности второго порядка.

При значении $t = 1$ имеет место

$$P(t = 1) = P(1) + (1 - t)P^{(1)}(1) + R(1-t)^2 \cong P(1) + P^{(1)}(1). \qquad (2.8)$$

Найдем общее выражение для производной многочлена Безье:

$$\vec{P}^{(1)}(t) = -m(1-t)^{m-1}\overrightarrow{P_0} + \sum_{i=1}^{m-1} C_m^i \left(it^{i-1}(1-t)^{m-1} - \right. \\ \left. -(m-1)t^i(1-t)^{m-i-1}\right)\vec{P_i} + mt^{m-1}\vec{P}_m.$$

При $t = 0$ имеем

$$\vec{P}^{(1)}(t) = -m(1-0)^{m-1}\overrightarrow{P_0}\sum_{i=1}^{m-1} C_m^i \left(i0^{i-1}(1-0)^{m-1} - \right. \\ \left. -(m-1)t^i(1-0)^{m-i-1}\right)\vec{P_i} + m0^{m-1}\vec{P}_m$$

или, если учесть, что выражение 0^{i-1} равно нулю для всех значений i, отличных от 1, а при $i = 1$ имеем $0^{i-1} = 1$, то при $i = 0$ производная в точке $t = 0$ определяется как

$$\vec{P}(0) = -m\vec{P}_0 + C_m^1(i0^0(1-0)^{m-i}\vec{P}_1 = m\left(\vec{P}_1 - \vec{P}_0\right).$$

При $t = 1$ будем иметь

$$\vec{P}^{(1)}(1) = -m(1-1)^{m-1}\overrightarrow{P_0} + \sum_{i=1}^{m-1} C_m^i \left(i1^{i-1}(1-1)^{m-1} - \right. \\ \left. -(m-1)1^i(1-1)^{m-i-1}\right)\vec{P_i} + m1^{m-1}\vec{P}_m$$

$$\vec{P}^{(1)}(1) = -m\vec{P}_{m-1} + m\vec{P}_m = m(\vec{P}_m - \vec{P}_{m-1}).$$

Полученные значения производных для $t = 0$ и $t = 1$ подставим в выражения (2.7), (2.8):

$$\vec{P}(t \to 0) = \vec{P}(0) + tm(\vec{P}_1 - \vec{P}_0);$$

$$\vec{P}\left(t \to 1\right) = \vec{P}(1) + (1-t)m(\vec{P}_m - \vec{P}_{m-1}).$$

Из полученных выражений видно, что кривая Безье в окрестности точки $t = 0$ стремится к прямой P_0P_1, отличаясь от нее тем меньше, чем t ближе к 0, а в окрестности точки $t = 1$ она стремится к прямой $P_{m-1}P_m$, отличаясь от нее тем меньше, чем t ближе к 1.

Выведем рекурсивные выражения для расчета функции, определяемой полиномом Безье.

Сначала выведем рекурсивное выражение для расчета C_m^i.

Покажем, что имеет место

$$C^i_m = C^i_{m-1} + C^{i-1}_{m-1}.$$

Для этого выполним следующие преобразования правой части доказываемого равенства:

$$C^i_{m-1} + C^{i-1}_{m-1} = \frac{m!}{i!\,(m-1)!} = \frac{(m-1)!}{i!\,\big((m-1)-i\big)} + \frac{(m-)!\,1}{i!\,\big((m-1)-(i-1)\big)!} =$$
$$= \frac{(m-1)!\,(m-i)}{i!\,(m-i)\big(m-(i-1)\big)!} + \frac{i(m-1)!}{i(i-1)!} = \frac{m!}{i!\,(m-i)!} = C^i_m.$$

В выражение (2.6) вместо C_m^i подставим $C_{m-1}^i + C_{m-1}^{i-1}$.

$$\vec{P}(t) = \sum_{i=0}^{m} C^i_m\, t^i\,(1-t)^{m-1}\,\vec{P}_i = \sum_i^m (C^i_{m-1} + C^{i-1}_{m-1})\, t^i\,(1-t)^{m-1}\vec{P}_i =$$
$$= \sum_{i=0}^{m-1} C^i_{m-1}\, t^i\,(1-t)^{m-i}\,\overrightarrow{P}_i + \sum_{i=1}^{m} C^{i-1}_{m-1}\, t^i\,(1-t)^{m-i}\,\vec{P}_i =$$
$$= (1-t)\sum_{i=0}^{m-1} C^i_{m-1}\, t^i\,(1-t)^{m-1}\vec{P}_i + t\sum_{i=1}^{m} C^{i-1}_{m-1}\, t^{i-1}\,(1-t)^{m-i}\,\vec{P}_i. \qquad (2.9)$$

Полином $\vec{P}(t)$, представленный записью (2.6), строится на $(m + 1)$ точках, начиная с 0-й точки P_0 и заканчивая точкой $\vec{P}_m$. Для простоты дальнейших рассуждений обозначим этот полином как $P_{0,m}$.

Фрагмент $\sum_{i=0}^{m-1} C^i_{m-1} t^i\,(1-t)^{m-i}\,\vec{P}_i$ из правой части уравнения (2.9) есть не что иное, как полином, построенный на m опорных точках, начиная с $\vec{P}_0$ и заканчивая $\vec{P}_{m-1}$, т. е. можно ввести обозначение

$$P_{0,m-1} = \sum_{i=0}^{m-1} C_{m-1}^{i}\, t^{i} (1-t)^{m-i}\, \vec{P}_i.$$

Фрагмент $\sum_{i=1}^{m} C_{m-1}^{i-1}\, t^{i-1} (1-t)^{m-i}\, \vec{P}_i$ из правой части уравнения (2.9) есть не что иное, как полином, построенный на m опорных точках, начиная с $\vec{P}_1$ и заканчивая $\vec{P}_m$, т. е. можно ввести обозначение

$$P_{1,\ m} = \sum_{i=1}^{m} C_{m-1}^{i-1}\, t^{i-1} (1-t)^{m-i}\, \vec{P}_i.$$

После введения этих обозначений уравнение (2.9) может быть представлено в виде

$$P_{0.\ m}(t) = (1-t) P_{0,\ m-1} - t P_{1,\ m}. \qquad (2.10)$$

Отсюда следует, что для определения значения полинома Безье $P_{0,m}(t)$, заданного на точках-ориентирах $P_0, P_1, \ldots P_i, \ldots P_{m-1}, P_m$, для некоторого конкретного значения параметра t, можно использовать следующее правило:

– для заданного значения t (например, $t = 0{,}5$) находится значение полинома Безье, заданного на точках-ориентирах $P_0, P_1, \ldots P_i, \ldots P_{m-1}, P_{1,m-1}$, и полинома Безье, заданного на точках-ориентирах $P_1, \ldots P_i, \ldots P_{m-1}, P_m$;

– определяется отрезок, соединяющий найденные точки;

– в качестве значения полинома Безье $P_{0,m}(t)$, заданного на точках-ориентирах $P_0, P_1, \ldots P_i, \ldots P_{m-1}, P_m$ берется точка, соответствующая середине (если $t = 0{,}5$) найденного отрезка.

Таким образом, в качестве итерационного алгоритма формирования значения полинома Безье для текущего значения t можно использовать следующий алгоритм.

На каждой итерации рассчитываются точки, расположенные на t-й части отрезков, соединяющих две соседние точки-ориентиры, полученные на предыдущей итерации. Эти рассчитанные точки берутся как точки-ориентиры для следующей итерации. Если после очередной итерации будет получена одна точка-ориентир, то эта точка и есть искомая точка, определяемая полиномом Безье для заданного значения t.

Граф-схема такого алгоритма приведена на рис. 2.15.

Здесь приняты следующие обозначения:

– $P_0, P_1, \ldots P_m$ – начальные опорные точки формируемой кривой;

– Q_i – значение полинома Безье на опорных точках i и $i + 1$;

– R_i – опорные точки для полинома следующей итерации.

На рис. 2.15 приведены примеры расчета значения полинома Безье для разного количества точек-ориентиров и разных значений параметра t.

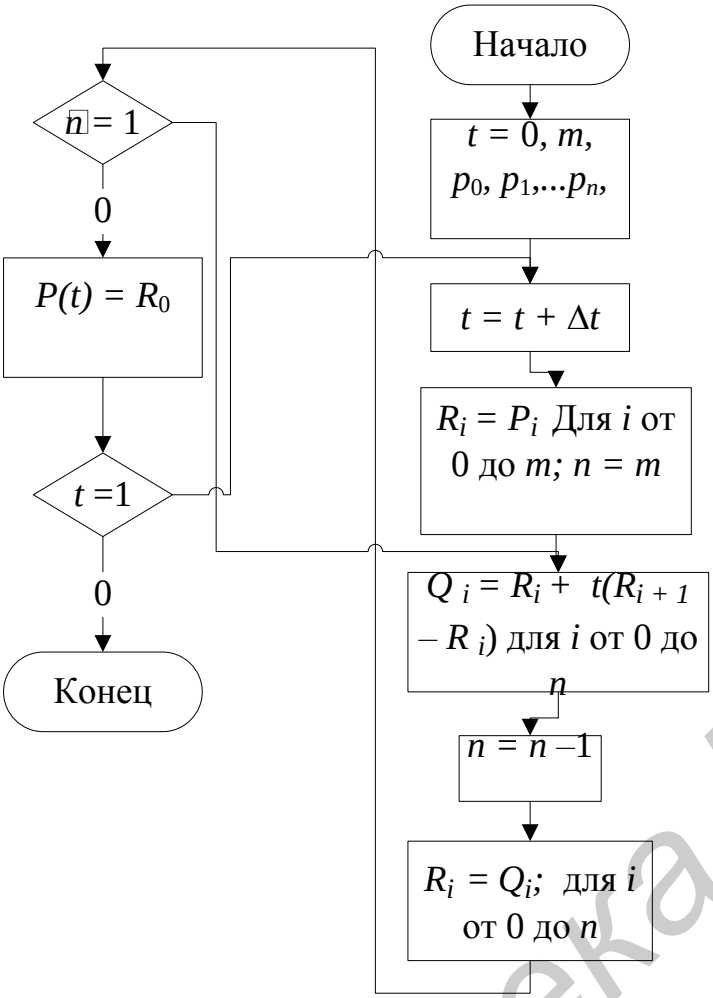


Рис. 2.15

На рис. 2.16 серыми кружочками обозначены вводимые промежуточные точки-ориентиры, получаемые в процессе расчета кривой Безье, и точка, которая соответствует значению полинома Безье при заданном значении параметра *t*.

Содержательно многочлен Безье можно представить в виде модели, в которой используется эластичная магнитная нить, закрепленная в точках P_0 и P_n, расположенная в магнитном поле, создаваемом одинаковыми магнитами, помещенными в точках P_1, …, P_{m-1}.

Для того чтобы обеспечивать большее «притяжение» к какой-либо точке-ориентиру, можно использовать механизм кратных точек (удвоение, утроение и т. д.), в соответствующее количество раз увеличивающий напряженность поля.

На рис. 2.17 механизм кратных точек используется для точек-ориентиров P_1, P_2, которые берутся с одинаковыми координатами. Поэтому при рассмотрении полинома, на точках-ориентирах P_1, P_2, в качестве его значения для любой заданной величины параметра *t* выбирается точка, соответствующая P_1 (или, что то же самое P_2). Поэтому на приведенном рисунке промежуточная точка-ориентир совпадает с начальными точками-ориентирами P_1, P_2.

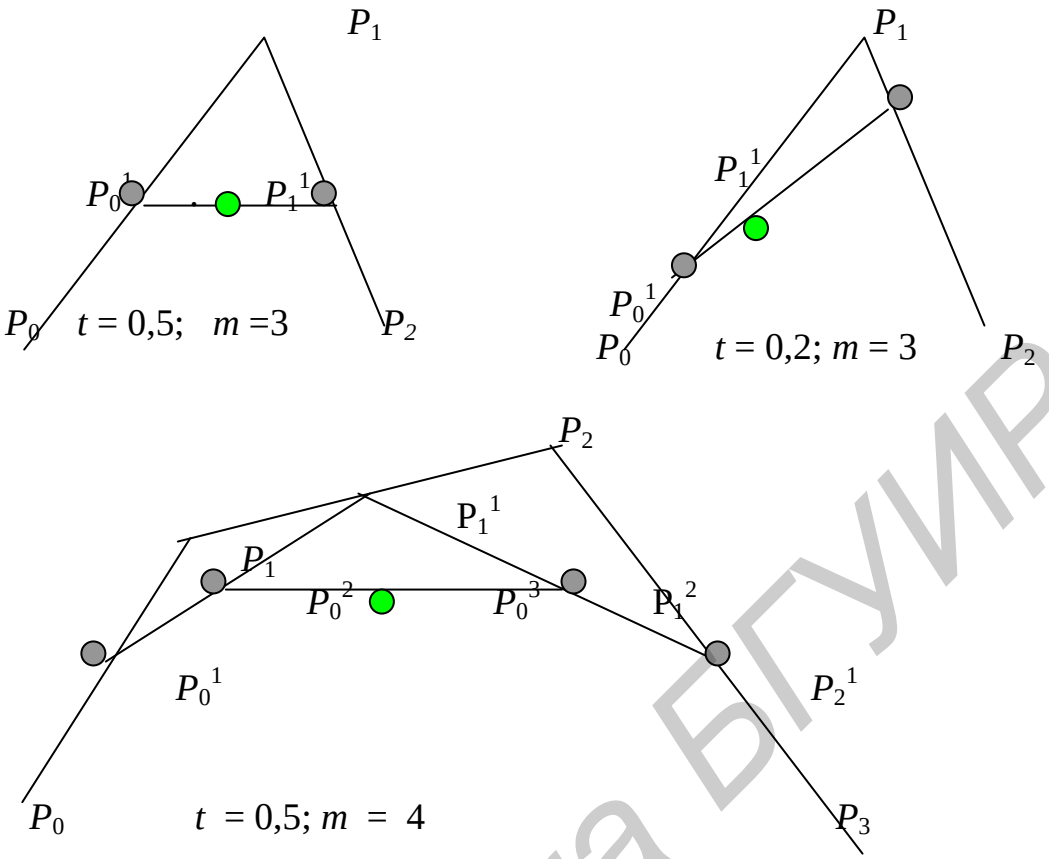


Рис. 2.16

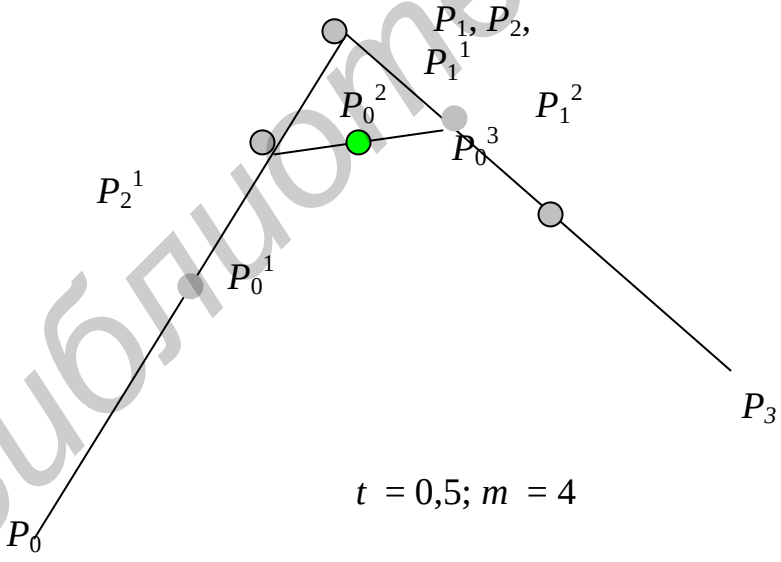


Рис. 2.17

Реализация данного алгоритма предполагает сравнительно больших временных затрат. Однако это не критично в большинстве случаев, учитывая, что кривые Безье строятся в интерактивном режиме.

3. Двумерные отсечения

Задача отсечения весьма часто встречается в машинной графике, например, при затенении объектов заднего плана объектами переднего плана, при работе с пользовательскими окнами и т. п.

3.1. Отсечение прямоугольным окном.

Алгоритм Сазерленда – Коуэна

В рассматриваемом отсечении окно представляет собой прямоугольник, стороны которого параллельны осям координат, как это показано на приведенном рисунке (рис. 3.1).

При *внутреннем отсечении* отображаются только те фрагменты отсекаемой фигуры, которые попадают во внутреннюю область окна, а все остальные отбрасываются. Для приведенного на рисунке окна у треугольника ABC отбрасывается часть BP_1P_2 и треугольник отображается своей частью AP_1P_2C.

При *внешнем отсечении* отображается лишь то, что оказывается вне окна.

Определить отображаемую часть отсекаемой фигуры можно, если будут известны точки пересечения ее ребер со сторонами заданного окна. Поэтому задача отсечения окном в итоге сводится к задаче отсечения отрезка.

Прямоугольное окно задается четырьмя параметрами $x_л$, $x_п$, $y_н$, $y_в$, соответственно координатами левой, правой, нижней и верхней границ окна.

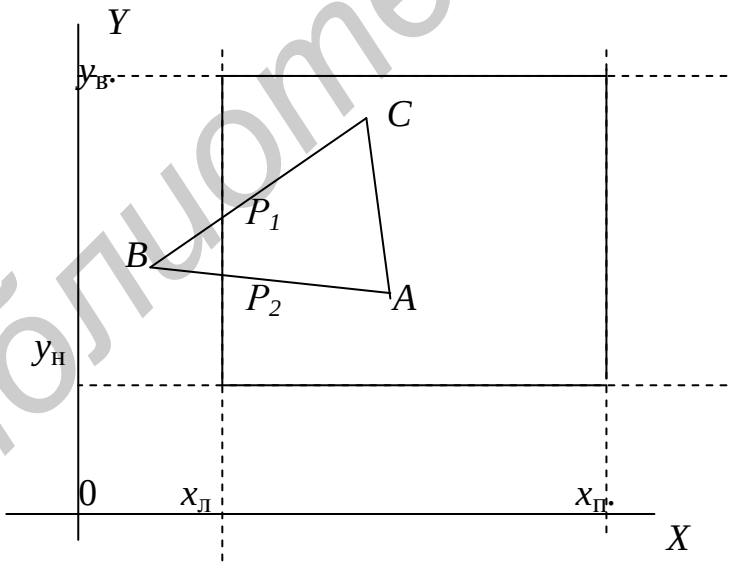


Рис. 3.1

Математически задача отсечения довольно проста. Она сводится к поиску точек пересечения отсекаемого отрезка со сторонами окна. Пересечение отрезка, заданного начальной Н($x_н$, $y_н$) и конечной К($x_к$, $y_к$) точками, с вертикальной (например левой) стороной ищется следующим образом (рис. 3.2, а), где точка пересечения обозначена как «П»:

$x_п = x_л$,

$(y_п - y_н)/(y_к - y_н) = (x_л - x_н)/(x_к - x_н)$,

откуда

$y_п = y_н + (x_л - x_н)\, m$,

где крутизна отрезка $m = (y_к - y_н)/(x_к - x_н)$.

Пересечение этого отрезка с горизонтальной (например верхней) стороной определяется как

$y_п = y_в$,

$(x_п - x_н)/(x_к - x_н) = (y_в - y_н)/(y_к - y_н)$, откуда:

$x_п = x_н + (y_в - y_н)(1/m)$.

Таким образом находятся точки пересечения прямых, несущих заданный отрезок, и текущей стороной прямоугольного окна. Далее необходимо определить принадлежность этой точки одновременно и заданному отрезку и соответствующей стороне окна. Эта процедура должна быть выполнена для всех четырех сторон окна, что требует больших затрат времени.

Количество отсекаемых отрезков в общем может быть достаточно велико, большинство из которых вообще может не иметь пересечение со сторонами окна. В этой связи интерес представляет алгоритм Сазерленда – Коуэна, обеспечивающий поиск точек пересечения только для ограниченного числа отрезков, видимость (или затенение) которых окном не очевидна.

Для быстрого анализа видимости отрезка в этом алгоритме используется кодирование концов отрезка. При этом пространство, в котором находится окно, разбивается на девять областей, как это показано на рис. 3.2.

Каждой области приписывается четырехразрядный код, причем первый разряд определяет положение области по отношению к линии, несущей левую сторону окна, второй разряд определяет положение области по отношению к линии, несущей правую сторону окна, третий разряд определяет положение области по отношению к линии, несущей нижнюю сторону окна, четвертый разряд определяет положение области по отношению к линии, несущей верхнюю сторону окна. При этом принимается, что каждая линия, несущая сторону окна, разбивает все пространство на две области: внутреннюю, в которой располагается окно, и противоположную – внешнюю.

Если точка лежит по отношению к некоторой стороне окна во внутренней области, то в соответствующий разряд четырехбитного кода устанавливается ноль, в противном случае, в этот разряд устанавливается единица. При кодировании концевым точкам отрезка приписываются коды областей, в которой они находятся. Например, на рис. 3.2, б точка *A* по отношению к левой стороне окна располагается во внутренней области, поэтому первый разряд четырехбитового кода равен нулю. Второй разряд этого кода равен нулю, так как точка *A* располагается во внутренней области по отношению к правой стороне окна. Третий разряд этого кода равен единице, так как точка *A* располагается во внешней области по отношению к нижней стороне окна. Четвертый разряд ко-

да равен нулю, так как точка *A* располагается во внутренней области по отношению к верхней стороне окна.

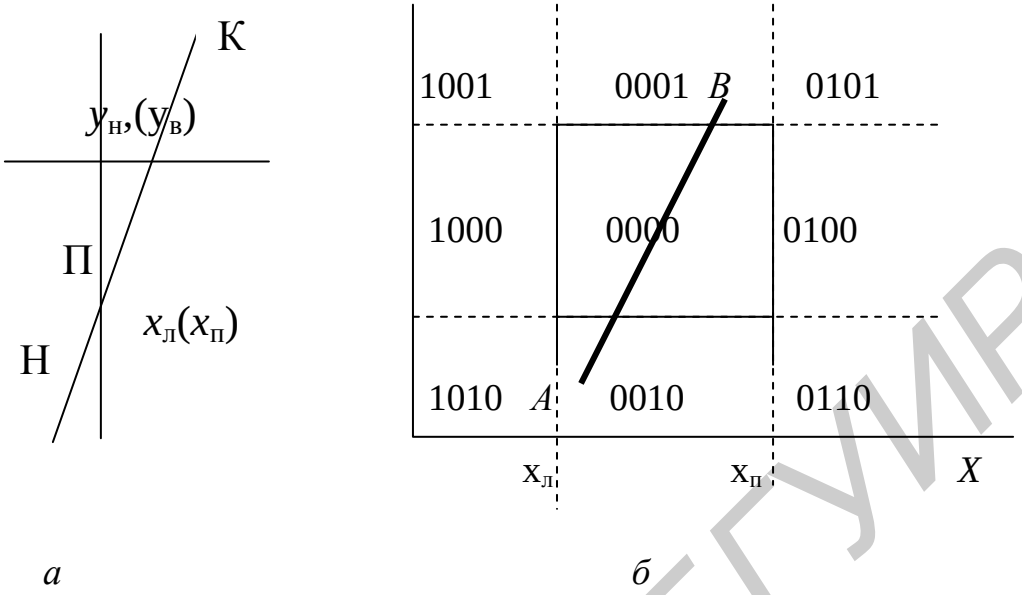


Рис. 3.2

Используя описанную кодировку, можно утверждать, что полностью видимый отрезок должен имеет нулевые коды для обоих своих концов. С другой стороны, если оба конца отрезка располагаются с внешней стороны по отношению хотя бы к одной стороне окна, то он полностью невидим. Следует учесть тот факт, что логическое произведение кодов концов отрезка в этом случае будет отлично от нуля.

На рис. 3.3 приведено несколько отрезков, расположенных в пространстве с окном, а на рис. 3.4 приведены в виде таблицы кодировка концов всех имеющихся отрезков и результаты анализа их видимости.

Идея алгоритма Сазерленда – Коуэна заключается в следующем.

Сначала кодируются концы отсекаемого отрезка и проверяется наличие его явной видимости (отрезок полностью видим или очевидно невидим). Если признаков явной видимости нет, то поочередно ищется пересечение отрезка с линией, несущей одно из ребер окна. При этом перед поиском пересечения отрезок ориентируется таким образом, чтобы его начало находилось с внешней стороны относительно линии, несущей текущую сторону (ребро) окна. Пересечение определяется только в том случае, если концы отрезка располагаются по разные стороны от текущего ребра окна.

Если оба конца отрезка находятся с внутренней стороны от рассматриваемой линии, то осуществляется переход к следующему ребру окна.

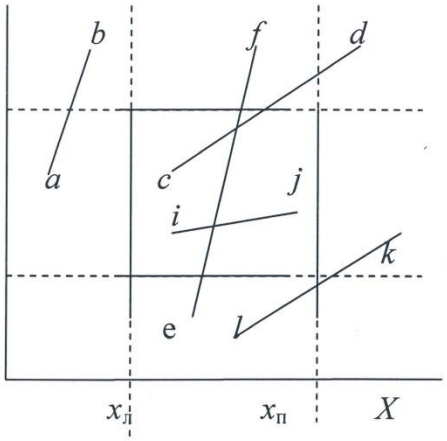


Рис. 3.3

Отре-зок	K1	K2	K1&K2	Видимость
ab	1001	1000	1000	Невидим
cd	0000	0101	0000	Неясно
ef	0010	0001	0000	Неясно
ij	0000	0000	0000	Видим
kl	0010	0100	0000	Неясно

Рис. 3.4

Если точка пересечения найдена, то начало отрезка перемещается в эту точку, точка нового начала отрезка кодируется и осуществляется анализ наличия очевидной видимости полученного нового отрезка. Если видимость отрезка не очевидна, то процедура поиска пересечения повторяется для очередного ребра окна и т. д.

В крайнем случае, после выполнения описанной процедуры для последнего ребра обязательно будет иметь место случай очевидной видимости. При обнаружении очевидной видимости отрезок будет или полностью невидим, или полностью видим, и задача считается решенной.

На рис. 3.5 приведена последовательность операций с отрезком. В исходном задании положение начальной $P_н$ и конечной $P_к$ точек таково, что принимается решение о неочевидной видимости отрезка. Поэтому запускается процедура поиска его пересечения с ребрами окна, начиная с левого.

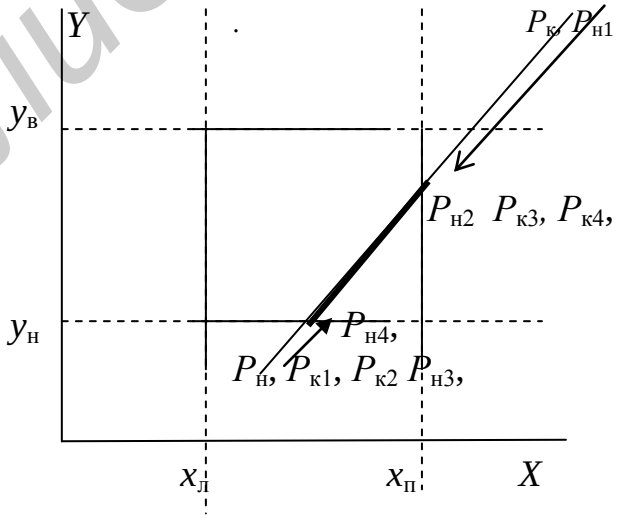


Рис. 3.5

С линией, несущей левое ребро, отрезок не пересекается, поэтому осуществляется переход к следующему правому ребру.

Отрезок пересекается с линией, несущей правое ребро, но его начало располагается с внутренней стороны от этого ребра, поэтому начало и конец отрезка меняются местами, и ищется пересечение линии, несущей отрезок $P_{н1}$, $P_{к1}$ и линии, несущей правое ребро окна, и начало отрезка переносится в найденную точку пересечения.

Вновь полученный отрезок $P_{н2}$, $P_{к2}$ имеет неочевидную видимость, поэтому осуществляется переход к определению пересечения отрезка с очередным (нижним) ребром. Начало и конец меняются местами и для вновь образованного отрезка $P_{н3}$, $P_{к3}$ находится точка его пересечения с нижним ребром. В эту точку переносится начало отрезка, и анализируется видимость вновь полученного отрезка $P_{н4}$, $P_{к4}$.

Отрезок $P_{н4}$, $P_{к4}$ будет полностью видим, задача считается решенной и часть $P_{н4}$, $P_{к4}$ исходного отрезка $P_{н}$, $P_{к}$ отображается на носителе. Граф-схема алгоритма приведена на рис. 3.6 и рис. 3.7.

На граф-схеме приняты следующие обозначения:

– $P_{н}$, $P_{к}$ – двухэлементная матрица координат начальной и конечной точек отрезка, нулевые элементы которых содержит координаты *x*, а первые элементы – координаты *y* точек;

– О – четырехэлементная матрица, задающая положение отсекающего окна, причем в ее нулевом, первом, втором и третьем элементах располагаются координаты, определяющие положение, соответственно, левого, правого, нижнего и верхнего ребер заданного окна;

– Кк, Кн – коды, соответственно, конца и начала отрезка, причем в их нулевом, первом, втором и третьем элементах располагаются биты, определяющие положение кодируемой точки по отношению к, соответственно, левому, правому, нижнему и верхнему ребрам заданного окна;

– *J* – номер текущего ребра заданного окна;

– *B* – функция очевидной видимости отрезка, которой может быть приписано одно из трех значений:

1 – если отрезок полностью видим;

0 – если отрезок полностью невидим;

2 – если видимость отрезка неочевидна;

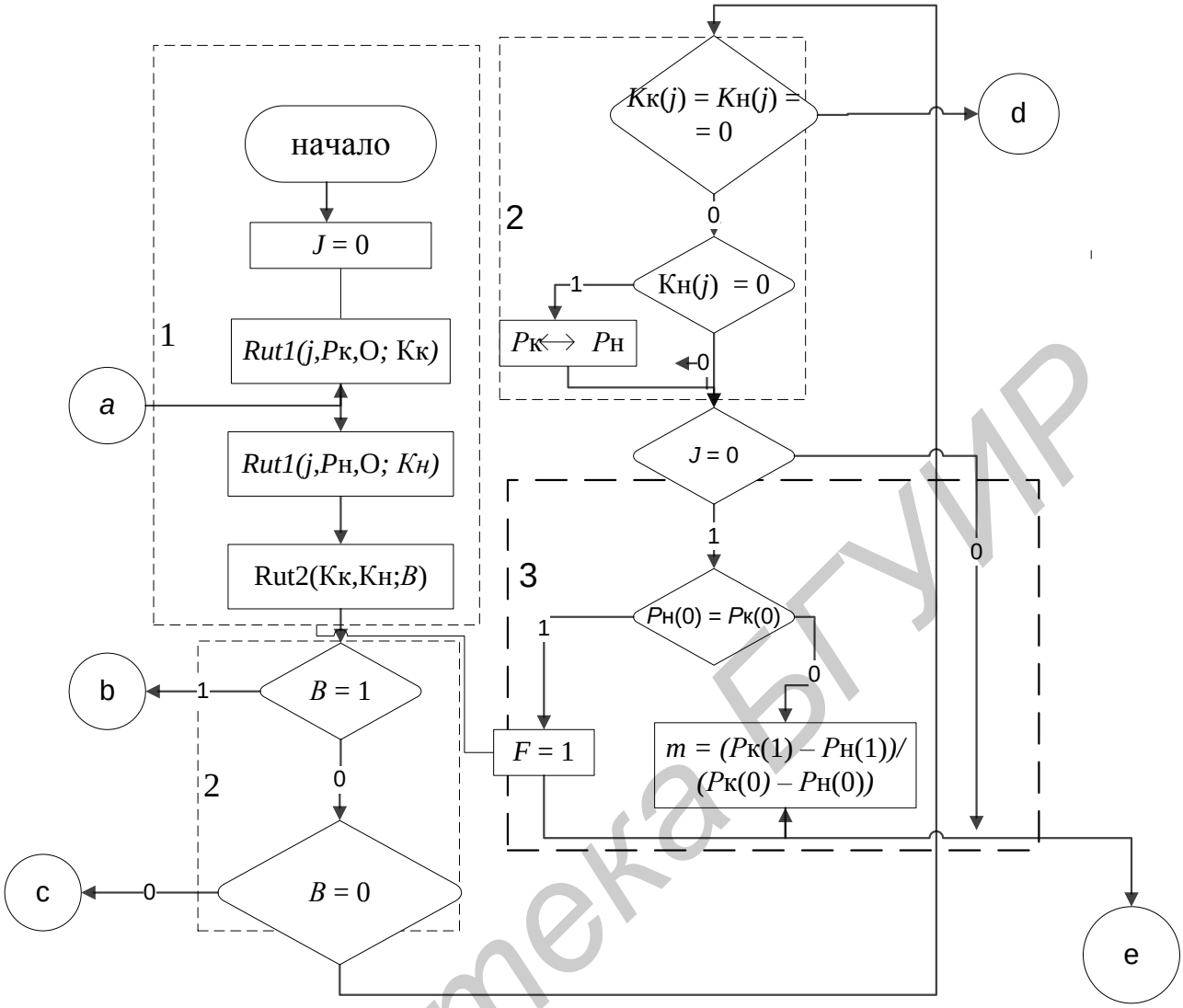


Рис. 3.6

– *F* – переменная, определяющая вид отрезка, которой приписывается значение «1», если отрезок вертикальный, и «0» во всех остальных случаях;

– *m* – крутизна отрезка.

Блок операторов 1 граф-схемы выполняет кодировку концов отрезка (подпрограммы *Rut*1) и определяет переменную очевидной видимости *B* (подпрограммы *Rut*2). Блок 2 на основании значения *B* или отображает отрезок, или заканчивает задачу (отрезок очевидно видим или очевидно не видим), или переходит к процессу поиска пересечения отрезка с очередным ребром *j* с предварительно сориентированным отрезком. Блок 3 реализует задачу определения крутизны отрезка перед определением пересечения с левым ребром.

Блок 4 осуществляет расчет точки пересечения для вертикальных ребер.

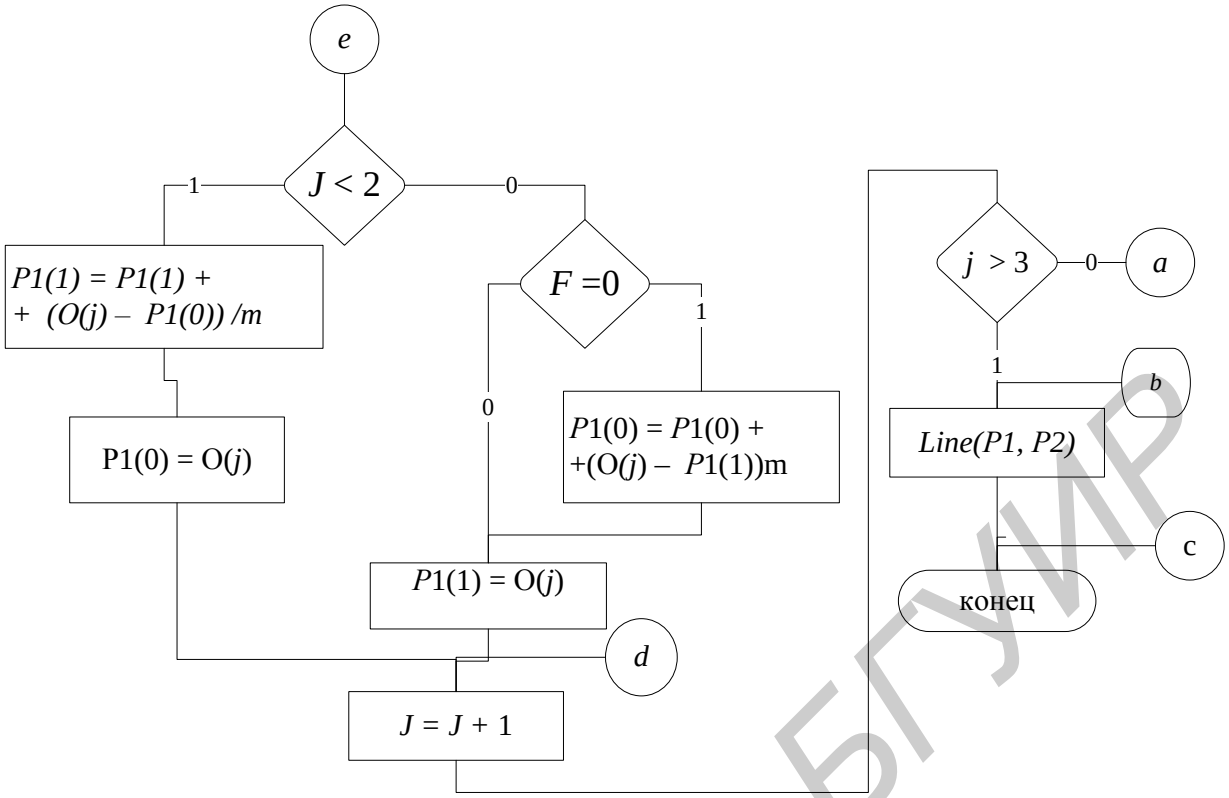


Рис. 3.7

На рис. 3.8 приведена ГСА кодирования точки, на рис. 3.9 представлена ГСА определения видимости отрезка.

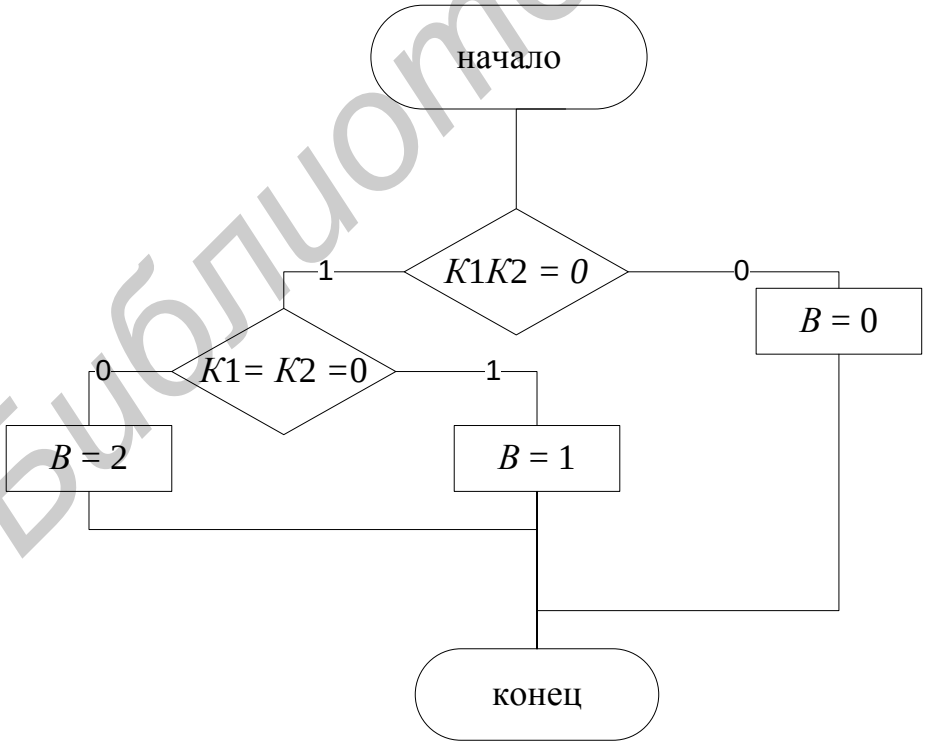


Рис. 3.8

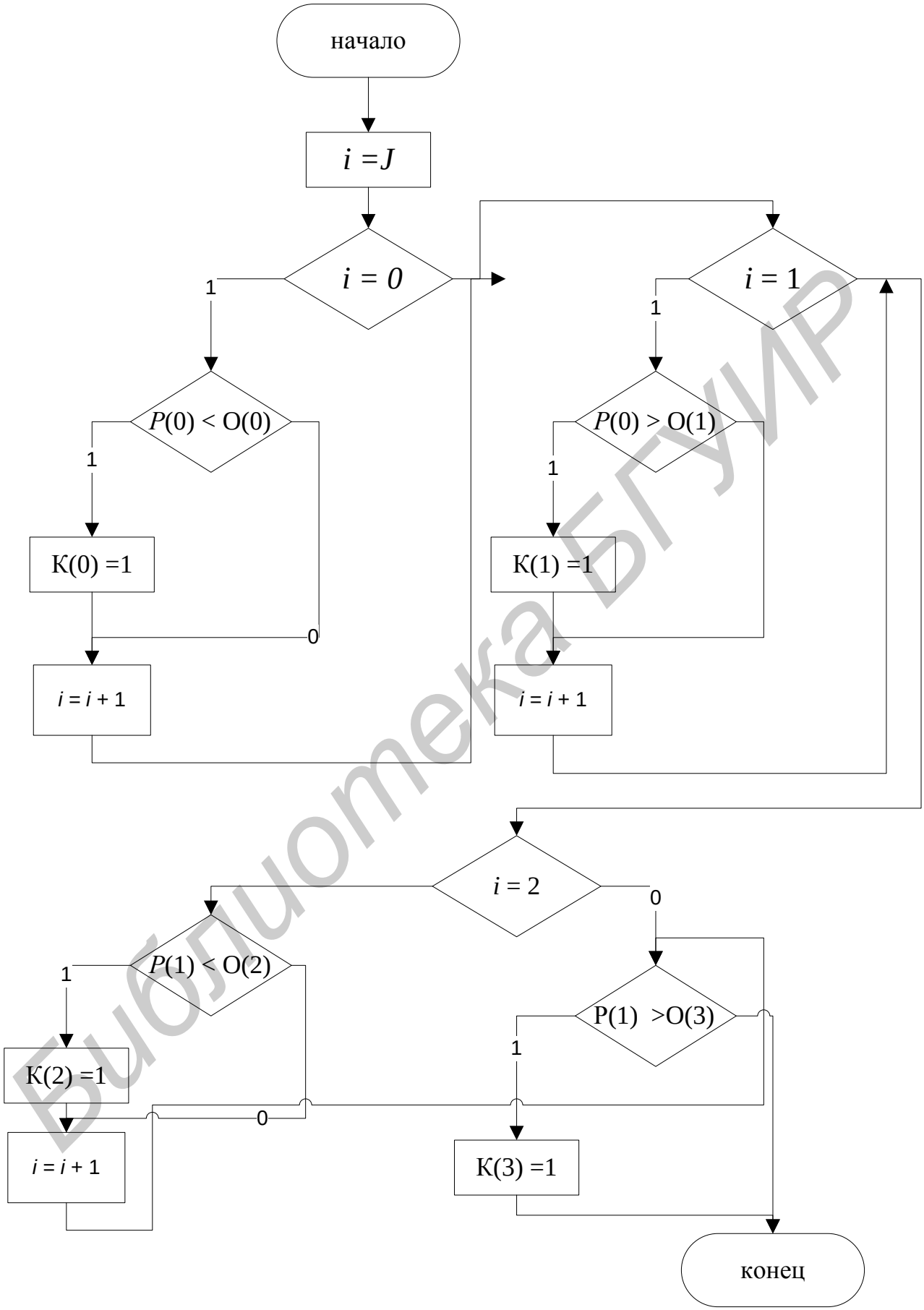


Рис. 3.9

3.2. Отсечение выпуклым многоугольником. Алгоритм Кируса – Бэка

Алгоритм Кируса – Бэка является одним из наиболее известных алгоритмов отсечения выпуклым многоугольником. Особенностью этого алгоритма является следующее.

Отрезок может пересекать выпуклый многоугольник не более, чем в двух точках, при этом, если рассматривать ориентированный отрезок, то он входит в тело многоугольника через одно ребро (назовем его фронтальным) и выходит через другое ребро (назовем его тыльным). Если отрезок рассматривать как вектор, то независимо от его положения в двумерном пространстве, для заданной направленности вектора все множество ребер многоугольника можно разбить на подмножество фронтальных и подмножество тыльных ребер.

На рис. 3.10 подмножество фронтальных ребер включает ребра $A_1\ A_2$, A_2A_3, а подмножество тыльных – ребра A_3A_4, A_4A_5, $A_5\ A_1$. При различных положениях отрезка заданной ориентации точка входа отрезка в тело многоугольника может быть только на фронтальных ребрах, а точка выхода – только на тыльных ребрах.

В качестве критерия отнесения отрезка к подмножеству тыльных или фронтальных ребер используется скалярное произведение вектора внутренней нормали $n_{вн}$ к соответствующему ребру и вектора директрисы отрезка D, при этом если скалярное произведение положительное, то ребро фронтальное, в противном случае – ребро тыльное.

Точкой входа отрезка в многоугольник может быть только одна из точек пересечения линий, несущих фронтальные ребра, и линии, несущей отрезок. На рис. 3.10 для отрезка PQ это точка вх. A_1 (точка пересечения линий, несущих отрезок и ребро A_1A_2) и точка вх. A_5 (точка пересечения линий, несущих отрезок и ребро A_2A_3), причем это должна быть наиболее удаленная от начала отрезка точка (в данном случае – это точка вх. A_1).

Точкой выхода отрезка из многоугольника может быть только одна из точек пересечения линий, несущих тыльные ребра, и линии, несущей отрезок. На рисунке для отрезка PQ это точка вых. A_3 (точка пересечения линий, несущих отрезок и ребро A_3A_4), точка вых. A_4 и точка вых. A_2, причем из всех точек выхода реальной точкой выхода отрезка из тела многоугольника может быть ближайшая из этих точек относительно начала отрезка (в данном случае – это точка вых. A_4).

Таким образом, точками пересечения многоугольника и отрезка PQ будет точка входа, соответствующая точке вх. A_1, и точка выхода, соответствующая точке вых. A_4.

Легко убедиться, что если не существует пересечения линии, несущей отрезок, и многоугольника, то точка входа, определенная вышеописанным способом, будет располагаться дальше от начала отрезка, чем точка выхода (см. отрезок P_2Q_2 на рис. 3.10).

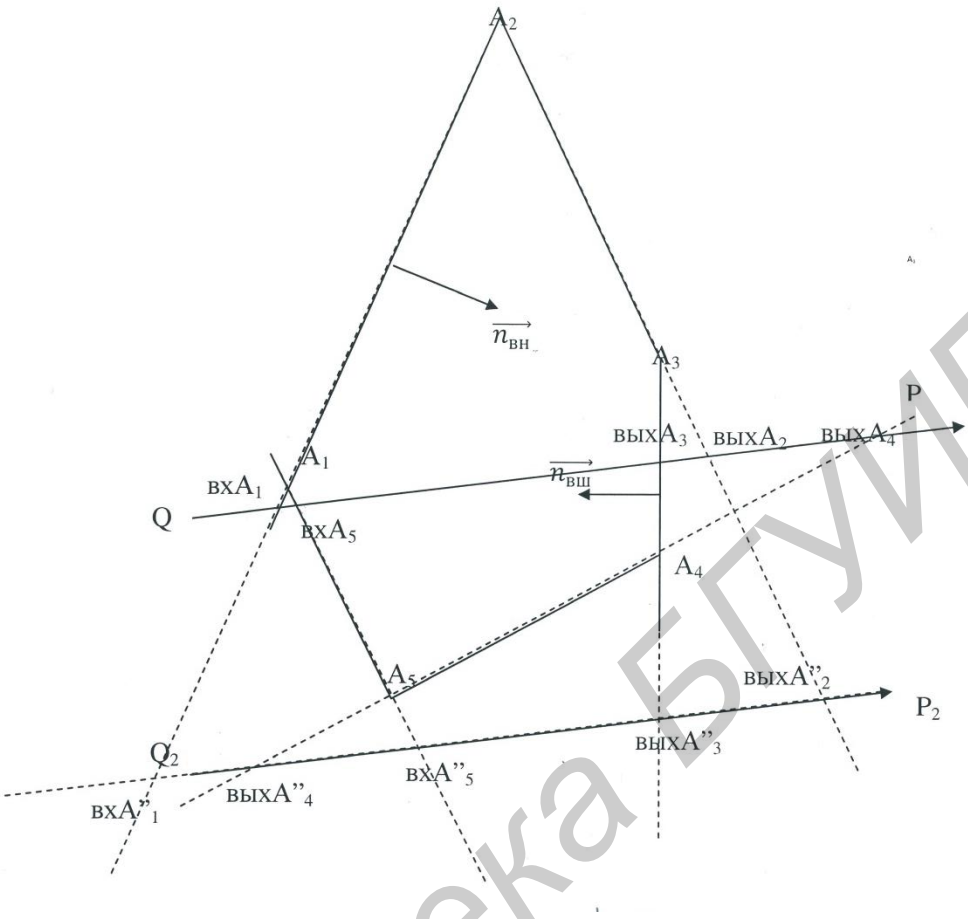


Рис. 3.10

Таким образом, поиск точек пересечения отрезка с многоугольником сводится к поиску пересечения линии, несущей отрезок, с линиями, несущими ребра многоугольника, определению типа каждого ребра (фронтальное или тыльное) и определению наиболее удаленной из них точек пересечения фронтальных ребер и ближайшей из точек пересечения для тыльных ребер. При этом выбранные две точки действительно являются искомыми точками пересечения, если экстремальная фронтальная точка пересечения располагается ближе к началу отрезка, чем экстремальная тыльная точка. В противном случае, отрезок не пересекается с заданным многоугольником.

Для решения проблемы определения точек пересечения линии, несущей отрезка, и линии, несущей соответствующее ребро многоугольника, выполним следующие действия.

Представим заданный отрезок *PQ* в виде вектора, как это показано на рис. 3.11, и запишем уравнение несущей его прямой в параметрической форме в виде:

$$\vec{P}(t) = \vec{P} + t(\overrightarrow{Q - P}),$$

где $P(t)$ – точка, лежащая на прямой, несущей отрезок *PQ*;

t – параметр задания прямой, несущей отрезок.

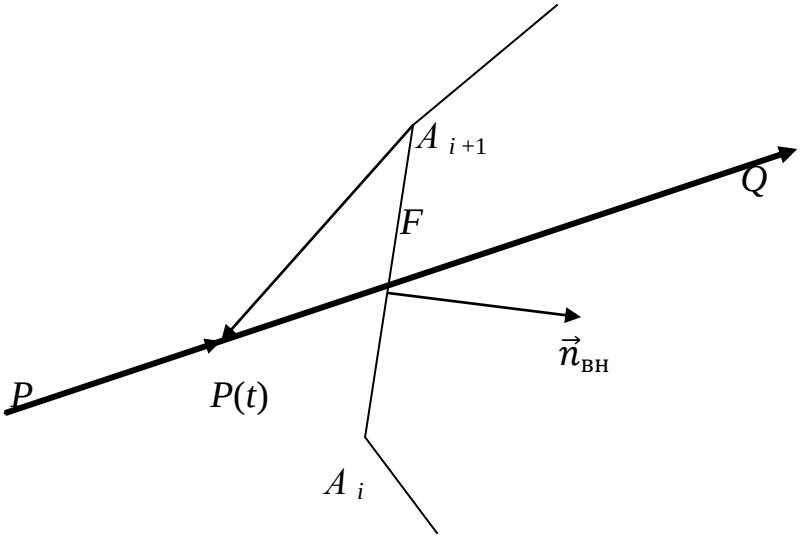


Рис. 3.11

Координаты точки $P(t)$, принадлежащей линии, несущей отрезок, для заданного значения параметра t рассчитываются следующим образом:

$z = x_P + t(x_Q - x_P)$;
$y = y_P + t(y_Q - y_P)$;
$z = z_P + t(z_Q - z_P)$.

Условием принадлежности точки заданному отрезку является выполнение условия

$0 \le t \le 1$.

Для поиска точки пересечения линии, несущей отрезок, и линии, несущей очередное ребро, возьмем некоторую произвольную точку F (см. рис. 3.11) на ли-нии, несущей это ребро, (в качестве этой точки можно использовать один из концов рассматриваемого ребра многоугольника) и введем вектор FP:

$$\overrightarrow{FP}(t) = \vec{P} - \vec{F} = \vec{P}_{\text{н}} + t(\vec{P}_{\text{к}} - \vec{P}_{\text{н}}) - \vec{F}.$$

Возьмем внутреннюю нормаль $n_{\text{вн}}$ для текущего ребра многоугольника и запишем скалярное произведение:

$$\vec{n}_{\text{вн}}\overrightarrow{FP}(t) = \vec{n}_{\text{вн}}(\vec{P}_{н} + t(\vec{P}_{к} - \vec{P}_{н}) - \vec{F}) =$$
$$= \vec{n}_{\text{вн}}\left(\vec{P}_{\text{н}} - \vec{F}\right) + t(\vec{P}_{\text{к}} - \vec{P}_{\text{н}})\vec{n}_{\text{вн}}. \qquad (3.1)$$

Обозначим вектор в первой скобке через $\vec{w}$ (вектор, определяющий положение ребра по отношению к начальной точке отрезка), а вектор во второй скобке через D (директрисса, определяющая направленность отрезка), и запишем уравнение (3.1) в виде

$$\vec{n}_{\text{вн}} \cdot \overrightarrow{FP} = \vec{n}_{\text{вн}} \cdot \vec{w} + t\vec{n}_{\text{вн}} \cdot \vec{D}.$$

Для положения точки $P(t)$, соответствующей искомой точке пересечения ребра и отрезка, вектор FP будет совпадать с линией, несущей анализируемое ребро многоугольника, а, следовательно, этот вектор будет перпендикулярен вектору внутренней нормали к этому ребру. Это позволяет для точки пересечения записать:

$$0 = \vec{n}_{\text{вн}} \cdot \vec{w} + t\vec{n}_{\text{вн}} \cdot \vec{D},$$

откуда:

$$t = -t\frac{\vec{n}_{\text{вн}} \cdot \vec{w}}{\vec{n}_{\text{вн}} \cdot \vec{D}}. \qquad (3.2)$$

Выражение (3.2) позволяет рассчитать значение параметра t для точки пересечения линий, несущих анализируемое ребро, и линии, несущей заданный отрезок.

Деление на ноль не допустимо, поэтому рассмотрим случаи, когда знаменатель рассматриваемого выражения равен нулю:

$$\vec{n}_{\text{вн}} \cdot \vec{D} = 0$$

1-й случай:

$D = 0$, что соответствует отрезку, вырожденному в точку.

В этом случае возможны три значения скалярного произведения, а следовательно, и три возможных положения точки по отношению к анализируемому ребру:

$$\vec{n}_{\text{вн}} \cdot \vec{w} = \begin{cases} < 0 - \text{точка лежит вне многоугольника;} \\ = 0 - \text{точка лежит на ребре;} \\ > - \text{точка лежит с внутренней стороны.} \end{cases}$$

2-й случай:

$D \neq 0$, что соответствует отрезку с не нулевой длиной.

Так как вектор внутренней нормали $\vec{n}_{\text{вн}}$ не равен нулю, равенство нулю рассматриваемого скалярного произведение может быть только из-за того, что вектор $\vec{n}_{\text{вн}}$ перпендикулярен директрисе D, т. е. рассматриваемый отрезок параллелен анализируемому ребру. В этом случае в зависимости от знака скалярного произведения вектора $\vec{n}_{\text{вн}}$ и вектора $\vec{w}$ будем иметь три различных варианта:

$$\vec{n}_{\text{вн}} \cdot \vec{w} = \begin{cases} < 0 - \text{отрезок лежит вне многоугольника;} \\ = 0 - \text{отрезок лежит на ребре;} \\ > - \text{отрезок лежит с внутренней стороны.} \end{cases}$$

На рис. 3.12 приведена граф-схема описанного алгоритма.

Оператор 1 осуществляет задание начальных значений для точки входа $t_{\text{вх}}$, точки выхода $t_{\text{вых}}$, начального номера очередного ребра i, количество ребер многоугольника «к» и рассчитывает директрису для заданного отрезка.

Оператор 2 осуществляет расчет внутренней нормали $n_{\text{вн}}$ и вектора ϖ для текущего ребра.

Оператор 5 определяет вид текущего ребра (тыльное или фронтальное).

Операторы 6 определяют текущее экстремальное значение для точки входа $t_{вх}$ и точки выхода $t_{вых}$.

В данном алгоритме для найденной точки пересечения линии, несущей отрезок, и линии, несущей текущее ребро многоугольника, непосредственно принадлежность отрезку не определяется. Эта задача решается за счет задания начального значения для $t_{вх}$, равного нулю, что соответствует началу отрезка, и начального значения для $t_{вых}$, равного единице, что соответствует концу отрезка.

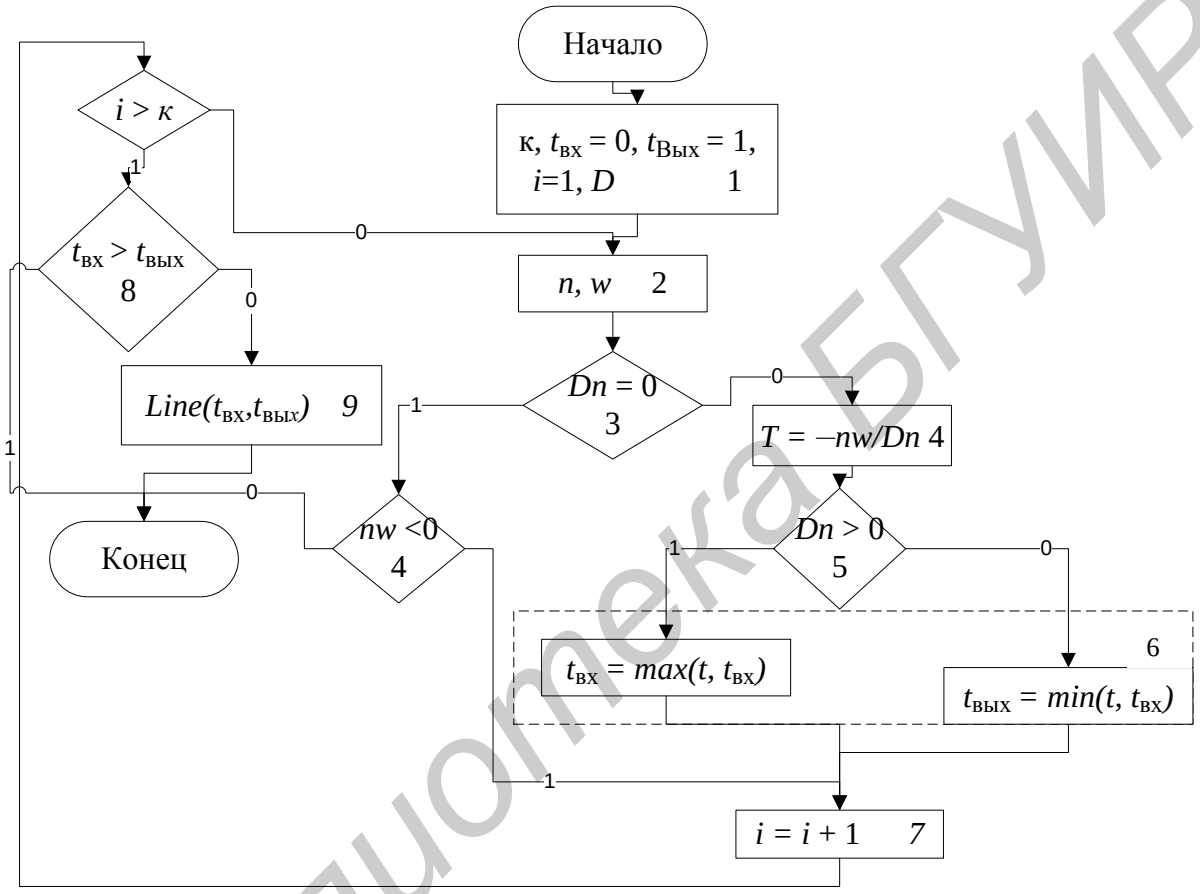


Рис. 3.12

Рассмотрим процедуры формирования скалярного произведения и внутренней нормали, которые используются в рассматриваемом алгоритме.

Скалярное произведение вектора V и W, заданных проекциями по осям координат V_x,V_y и W_x,W_y, в общем случае записывается следующим образом:

$$\vec{V} \cdot \vec{W} = (\vec{\imath} \cdot V_x + \vec{\jmath} \cdot V_y) \cdot (\vec{\imath} \cdot W_x + \vec{\jmath} \cdot W_y) = \vec{\imath} \cdot V_x \cdot \vec{\imath} \cdot W_x + \vec{\jmath} \cdot V_y \cdot \vec{\jmath} \cdot W_y +$$
$$+ \vec{\imath} \cdot V_x \cdot \vec{\jmath} \cdot W_y + \vec{\jmath} \cdot V_y \cdot \vec{\imath} \cdot W_x = V_x \cdot W_x + V_y \cdot W_y.$$

Расчет скалярного произведения вектора $\vec{V}$, соответствующего очередному ребру, и вектора $\vec{n}$, соответствующего нормали к этому ребру (рис. 3.13), осуществляется следующим образом:

$$A_i \cdot \vec{n}_i = A_{xi} \cdot n_{xi} + A_{yi} \cdot n_{yi},$$

где n_{xi}, n_{yi} – проекции вектора нормали к i-му ребру на координатные оси;

$\vec{A}_i$ – вектор, соответствующий i-му ребру многоугольника;

A_{xi}, A_{yi} – проекции вектора $\vec{A}_i$ на координатные оси .

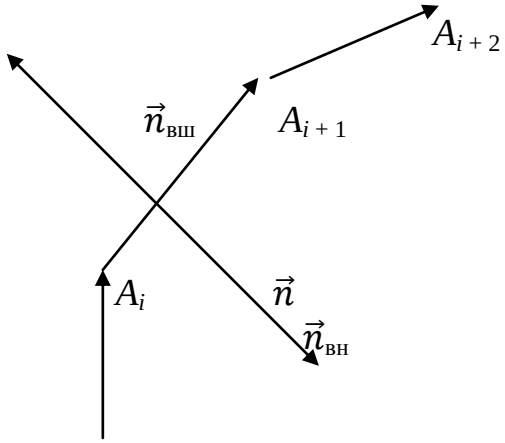


Рис. 3.13

Так как интерес представляет только направленность нормали, зададим n_x равной единице. Тогда n_y рассчитывается как

$$n_y = A_y/A_x \qquad (3.3)$$

Выражение (3.3) дает возможность рассчитать нормаль к ребру A_i, однако, это может быть или внутренняя нормаль или внешняя. Чтобы проверить, какая это нормаль, найдем скалярное произведение найденной нормали и вектора $A_{i+1}A_{i+2}$, соответствующего соседнему $(i+1)$-му ребру многоугольника, и проанализируем знак полученного результата:

$$\vec{A}_{i+1} \cdot \vec{n}_i = \begin{cases} \geq 0 - \text{нормаль внутренняя;} \\ < 0 - \text{нормаль внешняя.} \end{cases}$$

Если найденная нормаль внешняя, то после смены ее ориентации на противоположную будет получена искомая внутренняя нормаль.

3.3. Определение выпуклости многоугольника

Алгоритм Кируса – Бэка предполагает наличие выпуклого многоугольника, используемого в качестве окна.

Однако на практике весьма часто возникает задача отсечения многоугольником, а информация о том, является он выпуклым или нет, изначально не задается. В таком случае, прежде, чем начать процедуру отсечения, необходимо определить, какой задан многоугольник – выпуклый или нет.

Дадим некоторые определения выпуклости многоугольника.

Выпуклым считается многоугольник, для которого выполняется одно из нижеперечисленных условий (рис. 3.14):

– в выпуклом многоугольнике все вершины располагаются по одну сторону от линии, несущей любое ребро (по внутреннюю сторону относительно данного ребра);

– все внутренние углы многоугольника меньше 180^{o};

– все диагонали, связывающие вершины многоугольника, лежат внутри этого многоугольника.

Для выработки аналитического представление последнего критерия выпуклости, используем векторное произведение.

Векторное произведение W двух векторов a и b (рис. 3.15, а) определяется как

$$\overrightarrow{W} = \vec{a} \cdot \vec{b} = \left(\vec{\imath}a_x + \vec{\jmath}a_x + \vec{k}a_z\right)(\vec{\imath}b_x + \vec{\jmath}d_y + \vec{k}b_z),$$

где a_x, a_y, a_z и b_x, b_y, b_z являются проекциями на оси координат, соответственно, векторов – сомножителей a и b;

i, j, k – единичные векторы по координатным осям X, Y, Z.

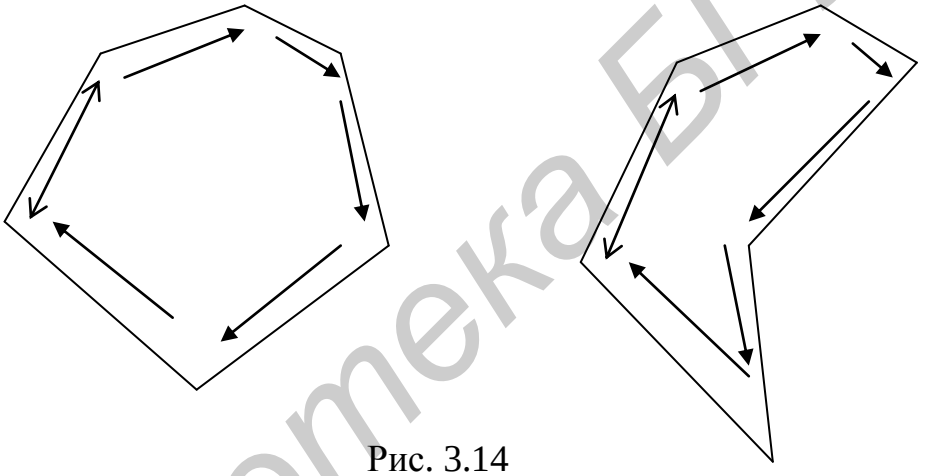


Рис. 3.14

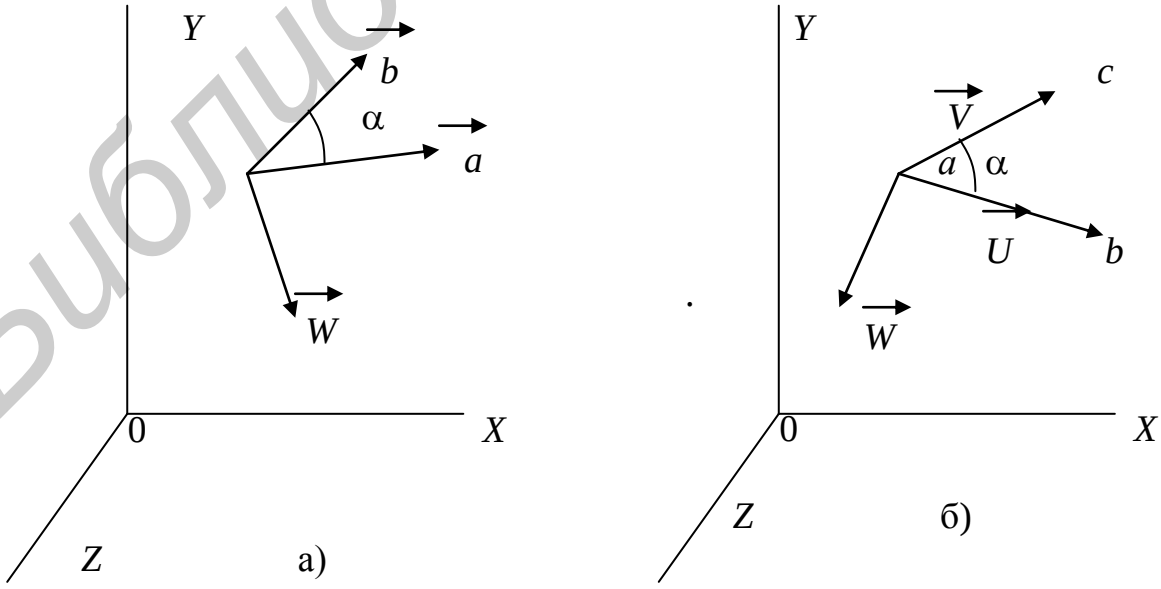


Рис. 3.15

Если рассматривать двумерное представление многоугольника как представление его в координатной плоскости *XY* трехмерной системе координат *X,Y,Z* (рис. 3.15, б), то выражение для формирования векторного произведения векторов *U* и *V*, являющихся соседними ребрами, образующими угол многоугольника, можно записать в виде определителя:

$$\vec{w} = \vec{U} \cdot \vec{V} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ U_x & U_y & U_z \\ V_x & V_y & V_z \end{vmatrix}.$$

Вектор векторного произведения перпендикулярен плоскости, в которой находятся вектора-сомножители. Направление вектора произведения определяется по правилу буравчика или по правилу винта с правой нарезкой.

Для случая, представленного на рис. 3.15, б), вектор *W*, соответствующий векторному произведению векторов *V,U*, будет иметь ту же направленность, что и направленность координатной оси *Z*.

Учитывая то, что проекции на ось *Z* векторов-сомножителей в этом случае равны нулю, векторное произведение можно представить в виде

$$\vec{v} = \vec{U} \cdot \vec{V} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ U_x & U_y & 0 \\ V_x & V_y & 0 \end{vmatrix} = \begin{vmatrix} U_x & U_y \\ V_x & V_y \end{vmatrix} \vec{k} = \Delta \vec{k} \qquad (3.4)$$

Единичный вектор *k* всегда положительный, следовательно, знак вектора ω векторного произведения будет определяться только знаком определителя Δ в вышеприведенном выражении. Отметим, что на основании свойства векторного произведения, при перестановке местами векторов-сомножителей *U* и *V* знак вектора ω будет меняться на противоположный.

Отсюда следует, что, если в качестве векторов *V* и *U* рассматривать два соседних ребра многоугольника, то порядок перечисления векторов в векторном произведении можно поставить в соответствие с обходом рассматриваемого угла многоугольника или ребер, образующих этот угол. Это позволяет использовать в качестве критерия определения выпуклости многоугольника правило: если для всех пар ребер многоугольника определители Δ имеют одинаковые знаки, то многоугольник выпуклый.

Так как ребра многоугольника задаются в виде координат их концевых точек, то для определения знака векторного произведения удобнее использовать определитель:

$$\Delta = \begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{vmatrix}.$$

Легко показать, что этот определитель эквивалентен определителю в выражении (3.4).

3.4. Отсечение невыпуклым многоугольником

Отсечение отрезка окном в форме невыпуклого многоугольника можно выполнить двумя способами.

1-й способ предполагает выполнение следующих действий:

1. Вводятся многоугольники, дополняющие заданный невыпуклый многоугольник до выпуклого.

2. Выполняется внутреннее отсечение полученным выпуклым многоугольником.

3. Для участка отрезка, полученного в результате этого внутреннего отсечения, осуществляется внешнее отсечение многоугольниками, дополняющими исходный до выпуклого.

Полученная часть отрезка и будет видимой частью отрезка при его внутреннем отсечении заданным окном в форме невыпуклого многоугольника.

Способ иллюстрируется ниже приведенным рисунком (рис. 3.16).

Для окна, представленного на рисунке, дополняющим многоугольником будет являться треугольник *CBD*. Поэтому сначала выполняется внутреннее отсечение выпуклым многоугольником *ABDE*, в результате чего получается отрезок t_1t_3. Далее выполняется внешнее отсечение отрезка t_1t_3 треугольником *CBD*. В результате чего будет отброшена часть t_1t_2, а оставшаяся часть t_2t_3 будет являться видимой частью исходного отрезка $T_нT_к$ в заданном окне *ABCDE*.

2-й способ предполагает выполнение следующих действий.

Заданный невыпуклый многоугольник разбивается на несколько выпуклых многоугольников.

1. Выполняется внутреннее отсечение заданного отреза всеми выпуклыми многоугольниками, составляющими исходный невыпуклый многоугольник.

2. Отображение всех частей отрезков, полученных при выполнении пункта 1, позволит получить часть исходного отрезка, являющуюся результатом внутреннего отсечения исходного отрезка заданным невыпуклым многоугольником.

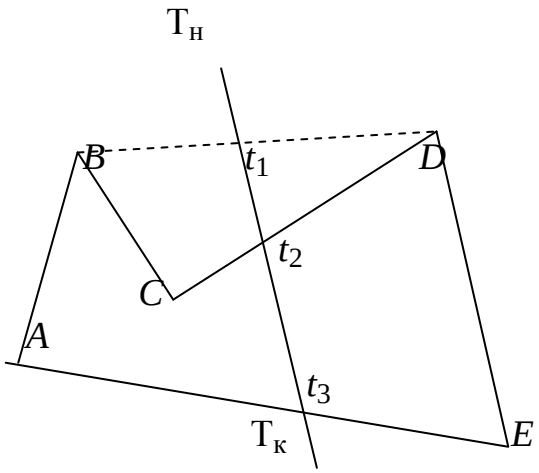


Рис. 3.16

Наиболее удобным является разбиение исходного многоугольника на треугольники, которые всегда выпуклые (рис. 3.17). Эта процедура может быть выполнена следующим образом:

– определяются диагонали исходного многоугольника, соединяющие стороны его внутренних углов, полностью находящиеся в теле многоугольника;

– из найденных диагоналей выбирается минимальная и соответствующий ей треугольник отбрасывается.

Процесс повторяется для оставшейся части многоугольника до тех пор, пока оставшаяся часть не станет треугольником.

Для многоугольников, приведенных на следующем рисунке (рис. 3.17), отсечение треугольников будет выполняться в последовательности, соответствующей приведенной на рисунке нумерации диагоналей.

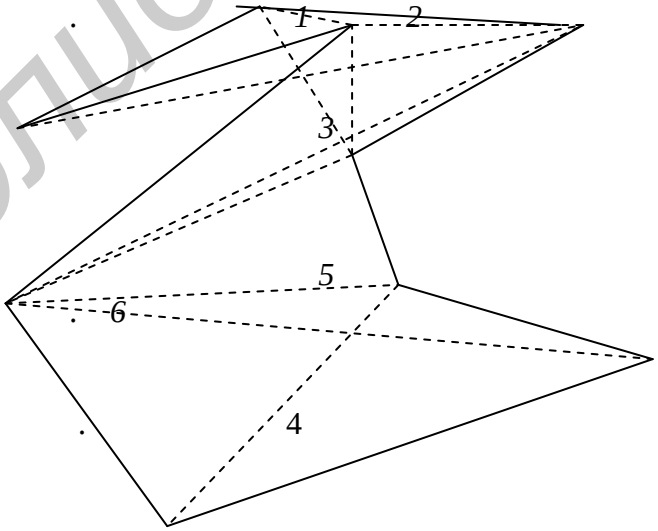


Рис. 3.17

Литература

1. Аммерал, Л. Принципы программирования в машинной графике / Л. Аммерал; пер. с англ. В. А. Львова. – М. : Сол. Систем, 1992. – 222 с.

2. Аммерал, Л. Машинная графика на персональных компьютерах / Л. Аммерал; пер. с англ. В. А. Львова. – М. : Сол. Систем, 1992. – 229 с.

3. Математика и САПР. В 2 кн. кн. 1: Основные методы. Теория полюсов / П. Шенен [и др.]; пер. с фр. С. Д. Чигиря; под ред. Н. Г. Волкова. – М. : Мир, 1988. – 206 с.

4. Сергеев, А. П. Основы компьютерной графики: Adobe Photoshop и CorelDRAW – два в одном: самоучитель / А. П. Сергеев, С. В. Кущенко. – М. : Диалектика, 2007. – 534 с.

5. Поляков, А. Ю. Методы и алгоритмы компьютерной графики в примерах на Visual C++ и Cj / А. Ю. Поляков, В. А. Бруснецев. – 2-е изд., перераб. и доп. – СПб. : БХВ-Петербург, 2003. – 547 с.

6. Никитин, А. П. Компьютерная графика и алгоритмы машинной графики / А. П. Никитин. – СПб. : БХВ-Петербург, 2003. – 164 с.

Св. план 2013, поз. 31

Учебное издание

Пешков Анатолий Тимофеевич
Кобайло Александр Серафимович

КОМПЬЮТЕРНАЯ ГРАФИКА. АЛГОРИТМЫ ПОСТРОЕНИЯ И ОТСЕЧЕНИЯ ЛИНИЙ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редакторы *И. В. Ничипор, М. А. Зайцева*
Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *В. М. Задоля*

Подписано в печать 03.01.2014. Формат 60x84 1/16. Бумага офсетная. Гарнитура «Таймс». Отпечатано на ризографе. Усл. печ. л. 2,91. Уч.-изд. л. 3,0. Тираж 100 экз. Заказ 633.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0494371 от 16.03.2009. ЛП №02330/0494175 от 03.04.2009.
220013, Минск, П. Бровки, 6