UNIVERSITATEA TEHNICĂ A MOLDOVEI
Facultatea Calculatoare, Informatică și Microelectronică
Departamentul Informatică și Ingineria Sistemelor

GRAFICA PE CALCULATOR

TEMA 9. ФУНДАМЕНТАЛЬНЫЕ АЛГОРИТМЫ СИНТЕЗА ИЗОБРАЖЕНИЙ

l.u., dr. NASTAS Andrei

СОДЕРЖАНИЕ

9.1. Алгоритмы построения векторов в дискретном пространстве

Высокопроизводительные алгоритмы, используемые при реализации функций отображения графических систем.

Как известно, оборудование, воспроизводящие изображения, использует растровый метод и получает изображение в закодированном числовом виде. Изображение хранится в растровой памяти.

Область отображения рассматривается как массив дискретных ячеек, называемых ***точками*** или ***пикселями***.

Каждый пиксель имеет отдельный адрес (*xp*, *yp*) на поверхности дисплея, к которому прикреплена 2D-декартова система координат.

Между пиксельными адресами и ячейками памяти растра, в которых хранятся интенсивности отображения (цвета) пикселей, существует двустороннее соответствие.

9.1. Алгоритмы построения векторов в дискретном пространстве

Алгоритмы построения графических примитивов в дискретном пространстве определяют пиксельные адреса, которые лучше всего аппроксимируют примитиву.

Для того чтобы просмотреть примитив, необходимо вызвать подпрограмму, используемого оборудования, которая вписывает в ячейку, соответствующую заданному адресу пикселю, значение интенсивности (цвета), с которой желательно отобразить этот пиксель.

Одной из таких процедур является ***putpixel()*** из пакета функций языка C++.

Любая графическая библиотека содержит подпрограмму, которая может быть вызвана в прикладных программах с целью рисования отрезка линии; например, подпрограммы ***line()*** и ***lineto()***, которые используются в языке C++.

В рамках подпрограммы отображения отрезка линии необходимо определить те пиксели, которые лучше всего аппроксимируют теоретический отрезок, то есть тот отрезок, математически определяемый координатами его концов.

9.1. Алгоритмы построения векторов в дискретном пространстве

Чем лучше разрешение поверхности дисплея, тем лучше аппроксимируются линии. Однако для определенных наклонов дискретное приближение видно даже при хорошем разрешении. Большинство алгоритмов аппроксимации векторов в дискретном пространстве основаны на инкрементных методах.

Таким образом, отталкиваясь от уравнения прямой, определяемой точками (x_1, y_1) и (x_2, y_2),

$$y = m \cdot x + b, \tag{9.1}$$

где $m = \frac{y_2 - y_1}{x_2 - x_1}$

и $b = y_1 - m \cdot x_1$.

Если $x_1 < x_2$, последовательность точек на линии может быть определена путем изменения x от x_1 до x_2 и вычисления y из уравнения прямой.

В этом случае отображение отрезка (x_1, y_1) – (x_2, y_2) можно описать следующим образом: пиксель с адресом ($xp = x$, $yp =$ целое число ближайшее к y) отображается, в указанном цвете.

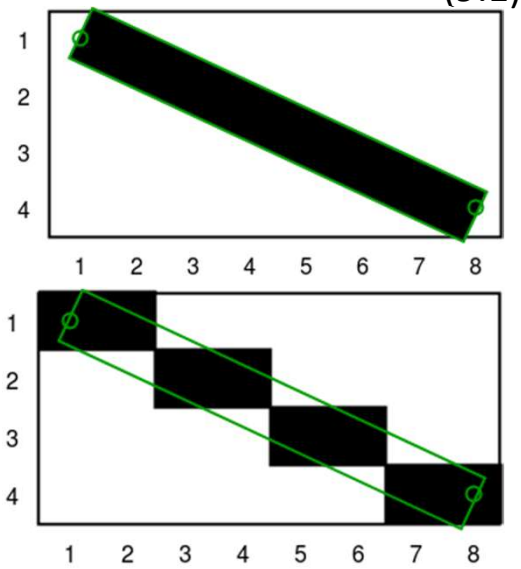


Рис. 9.1. Разложение прямой в растр

9.1. Алгоритмы построения векторов в дискретном пространстве

Это описание имеет два основных недостатка:

1) Уклон линии не учитывается; таким образом, чем больше наклон, тем меньше количество вычисляемых и отображаемых точек, а полученные точки будут все больше раздвигаться, так как изменение *y* между двумя последовательными значениями *x* увеличивается.

2) Вычисление каждой точки на отрезке предполагает выполнение умножения и сложения между двумя реальными числами.

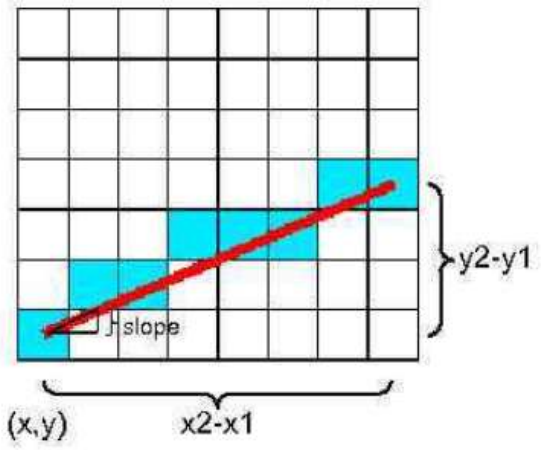
У большинства процессоров выполнение операции умножения или деления между двумя целыми числами занимает как минимум в 10 раз больше времени, чем выполнение операции сложения или вычитания.

Также не все процессоры, используемые для выполнения базовых подпрограмм графической системы, работают с реальными числами (операции моделируются).

Даже в случае сопроцессора, выполняющего операции с реальными числами, они медленнее, чем целочисленные операции.

9.1.1. Алгоритм цифрового дифференциального анализатора DDA (Digital Differential Analyser)

DDA используется для рисования прямой линии для формирования линии, треугольника или многоугольника в компьютерной графике. DDA анализирует образцы вдоль линии через равные промежутки времени по одной координате в целом, а для другой координаты округляются между целым, ближайшим к линии. Поэтому по мере продвижения по линии, сканируется первая целочисленная координата и округляется вторая до ближайшего целого числа. Следовательно, линия, проведенная с использованием DDA для координат *x*, будет x_0 до x_1, а для координат *y* будет $y = mx + b$, а для рисования функция будет *Fn*(*x*, *y* округленный).



9.1.1. Алгоритм DDA (Digital Differential Analyser)

Первый недостаток можно устранить следующим образом: Если $0 \leq \mathrm{abs}(m) \leq 1$, тогда последовательность точек будет получена путем приращения *x*, в противном случае путем увеличения *y*.

Алгоритмы генерации векторов в дискретном пространстве, которые будут представлены ниже, не содержат вычислений умножения или деления для определения значения точек на векторах, а некоторые из них работают с целыми числами.

Рассмотрим отрезок, (x_1, y_1) – (x_2, y_2) с точками (x', y'), (x'', y'') расположенных последовательно на нем.

Тогда: $m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{y" - y'}{x" - x'}$.

1) Для $\mathrm{abs}(m) \leq 1$ и $x_1 < x_2$ отрезок генерируется путем приращения *x*, поэтому:

$$x'' = x' + 1,\ x'' - x' = 1,\ y'' = y' + m. \qquad (9.2)$$

2) Для $\mathrm{abs}(m) > 1$ и $y_1 < y_2$ отрезок генерируется путем приращения y, поэтому:

$$y'' = y' + 1,\ y'' - y' = 1,\ x'' = x' + 1/m. \qquad (9.3)$$

Случаи: $\mathrm{abs}(m) \leq 1$ și $x_1 > x_2$ и $\mathrm{abs}(m) > 1$ și $y_1 > y_2$ сводятся к случаям в случаях (1) и (2) путем изменения значений переменных x_1 и x_2, соответственно y_1 и y_2.

Вычисление точек на отрезке содержит только операции сложения и вычитания, между реальными числами.

9.1.2. Алгоритм Брезенхама

Алгоритм Брезенхэма был разработан J. E. Bresenham в 1962 году и является гораздо более точным и гораздо более эффективным, чем DDA. Он сканирует координаты, но вместо того, чтобы округлять их, он учитывает добавочное значение путем сложения или вычитания и, следовательно, может использоваться для рисования кругов и кривых. Поэтому, если линия проводится между двумя точками *x* и *y*, то следующие координаты будут (*xa* + 1, *ya*) и (*xa* + 1, *ya* + 1), где *a* — добавочное значение следующих координат и разница между ними будет вычисляться путем вычитания или сложения образованных ими уравнений.

9.1.2. Алгоритм Брезенхама

Алгоритм Брезенхама также основан на инкрементальном методе, но содержит только целочисленные операции.

Алгоритм определен для векторов с наклоном от 0 до 1.

Для каждого значения *x* выбирается дискретная точка пространства, которая ближе к точке теоретического вектора.

Выбор основан на расстояниях от двух точек-кандидатов (точки над вектором и точки под вектором) до соответствующей точки вектора.

Пусть $m = \frac{y_2 - y_1}{x_2 - x_1}$ наклон вектора и (x_i, y_i) последняя точка дискретного пространства, выбранного в процессе генерации вектора.

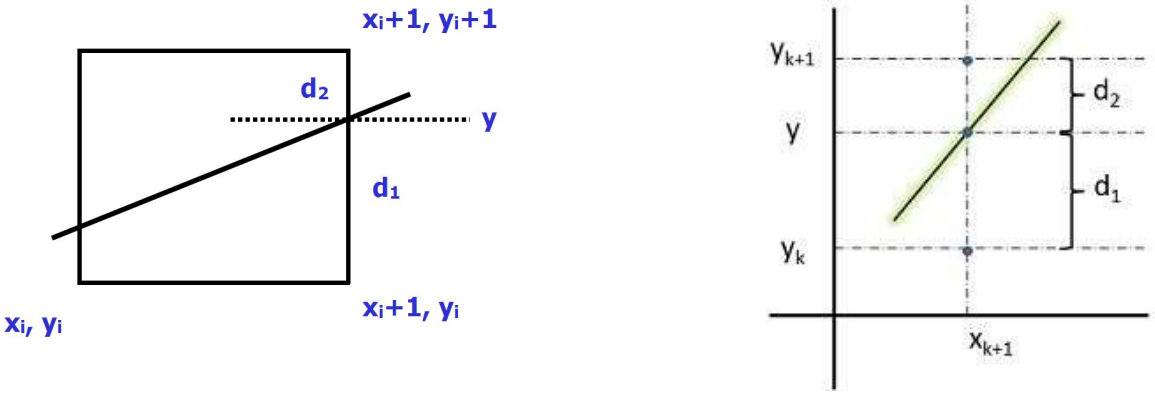


Рис. 9.2. Представление прямой по алгоритму Брезенхама

9.1.2. Алгоритм Брезенхама

Обозначаем через: d_1 расстояние от теоретического вектора до точки $O(x_i + 1, y_i)$ и d_2 расстояние от теоретического вектора до $D(x_i + 1, y_i + 1)$.

Следующим выбранным пунктом будет O, если $d_1 < d_2$, или пункт D в противном случае.

Если $d_1 = d_2$ можно выбрать любой из двух пунктов.

Мы выражаем разницу $d_1 - d_2$:

$$y = m \cdot (x_i + 1) + b, \tag{9.4}$$

это ордината точки с теоретического вектора

$$d_1 = y - y_i = m \cdot (x_i + 1) + b - y_i, \tag{9.5}$$

$$d_2 = y_i + 1 - y = y_i + 1 - m \cdot (x_i + 1) - b, \tag{9.6}$$

$$d_1 - d_2 = 2 \cdot m \cdot (x_i + 1) - 2 \cdot y_i + 2 \cdot b - 1. \tag{9.7}$$

Заменив m на dy/dx, затем умножаются обе стороны на dx.

В результате получаем:

$$t_i = (d_1 - d_2) \cdot d_x = 2 \cdot d_y \cdot (x_i + 1) - 2 \cdot d_x \cdot y_i + 2 \cdot b \cdot d_x - d_x, \tag{9.8}$$

t_i представляет *погрешность приближения на шаге i*.

9.1.2. Алгоритм Брезенхама

Значение $c = 2 \cdot b \cdot d_x - d_x + 2 \cdot d_y$ одинаково для каждого шага, поэтому:

$$t_i = 2 \cdot d_y \cdot x_i - 2 \cdot d_x \cdot y_i + c, \quad (9.9)$$

Обозначим через $(x_i + 1, y_i + 1)$ точку, которую необходимо выбрать на текущем шаге.

Тогда выражение погрешности аппроксимации для следующего шага выглядит следующим образом:

$$t_{i+1} = 2 \cdot d_y \cdot x_{i+1} - 2 \cdot d_x \cdot y_{i+1} + c. \quad (9.10)$$

1) Если $t_i \le 0$ выбираем точку O, таким образом $x_{i+1} = x_i + 1$ и $y_{i+1} = y_i$.

Следует:
$$t_{i+1} = 2 \cdot d_y \cdot (x_i + 1) - 2 \cdot d_x \cdot y_i + c, \quad (9.11)$$
или
$$t_{i+1} = t_i + 2 \cdot d_y.$$

2) Если $t_i > 0$ выбираем точку D, таким образом $x_{i+1} = x_i + 1$ и $y_{i+1} = y_i + 1$.

Следует:
$$t_{i+1} = 2 \cdot d_y \cdot (x_i + 1) - 2 \cdot d_x \cdot (y_i + 1) + c, \quad (9.12)$$
или
$$t_{i+1} = t_{i+2} \cdot d_y - 2 \cdot d_x.$$

Вычисляется погрешности приближения для первого шага путем замены в выражении (9.9) x_i на x_1 и y_i на y_1:

$$t_1 = 2 \cdot d_y \cdot x_1 - 2 \cdot d_x \cdot y_1 + 2 \cdot d_y - d_x + 2 \cdot d_x (y_1 - (d_y / d_x) \cdot x_1), \quad (9.13)$$
$$t_1 = 2 \cdot d_y - d_x.$$

9.1.3. Обобщение алгоритма Брезенхама

Векторы, определенные в 2D-пространстве, которые не являются горизонтальными или вертикальными, могут быть классифицированы на восемь геометрических классов, называемых октантами.

Имеем вектор $(x_1, y_1) – (x_2, y_2)$.

Октант, к которому он принадлежит, определяется в соответствии с $d_x = x_2 – x_1$ şi $d_y = y_2 – y_1$, таким образом:

Октант 1: $d_x > 0$, $d_y > 0$ и $d_x \geq d_y$;
Октант 2: $d_x > 0$, $d_y > 0$ и $d_x < d_y$;
Октант 3: $d_x < 0$, $d_y > 0$ и $\text{abs}(d_x) < d_y$;
Октант 4: $d_x > 0$, $d_y > 0$ и $\text{abs}(d_x) \geq d_y$;
Октант 5: $d_x < 0$, $d_y < 0$ и $\text{abs}(d_x) \geq \text{abs}(d_y)$;
Октант 6: $d_x < 0$, $d_y < 0$ и $\text{abs}(d_x) < \text{abs}(d_y)$;
Октант 7: $d_x > 0$, $d_y < 0$ и $d_x < \text{abs}(d_y)$;
Октант 8: $d_x > 0$, $d_y < 0$ и $d_x \geq \text{abs}(d_y)$.

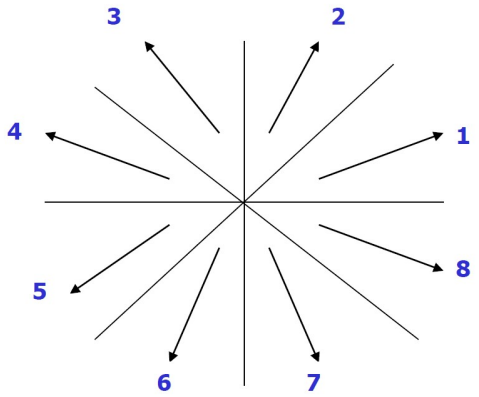


Рис. 9.3. Направления векторов в зависимости от октанта в котором он расположен

9.1.3. Обобщение алгоритма Брезенхама

Разница между алгоритмом DDA и Брезенхама

• DDA использует плавающие точки, в то время как алгоритм Брезенхама использует фиксированные точки.

• DDA округляет координаты до ближайшего целого числа, алгоритм Брезенхама этого не делает.

• Алгоритм Брезенхэма намного точнее и эффективнее, чем DDA.

• Алгоритм Брезенхэма может рисовать круги и кривые с гораздо большей точностью, чем DDA.

• DDA использует умножение и деление уравнения, но алгоритм Брезенхама использует только вычитание и сложение.

9.2. Алгоритмы построения кругов

Окружность определяется ее центром и радиусом, или периферийной точкой.
Его можно аналитически определить в:
– Декартовых координатах,
– или полярных (по параметрическим уравнениям).

9.2.1. Вычисление точек на окружности по декартовым координатам окружности

Инкрементное вычисление точек на окружности может быть основано на уравнении в декартовых координатах:

$$(x - xc)^2 + (y - yc)^2 = r^2,$$

где: (xc, yc) является центром круга, а r — радиус.

Увеличивая x от значения $xc - r$ до $xc + r$ с шагом 1 и мы определяем y с помощью выражения:

$$y = yc \pm \sqrt{r^2 - (x - xc)^2}, \quad (9.14)$$

или увеличиваем x из уравнения окружности.

Метод имеет два недостатка:

— арифметическое выражение, с помощью которого получается y (или x), содержит трудоемкие операции (повышение в степень, извлечение квадратного корня);

— полученные точки расположены неравномерно; таким образом, если x увеличивается, то точки в непосредственной близости от концов диапазона [$xc - r$, $xc + r$] будут очень редкими (точки, для которых при изменении x на 1 приведет к очень большому изменению y).

9.2.2. Вычисление точек на окружности с использованием параметрических уравнений окружности

Параметрические уравнения окружности:

$$\begin{cases} x = xc + r \cdot \cos(t), \\ y = yc + r \cdot \sin(t), \end{cases} \quad \text{где } 0 \le t \le 6{,}28\ (2 \cdot \pi), \tag{9.15}$$

позволяют получать равноудаленные точки от окружности путем изменения ее на t от 0 до 6,28 с постоянным шагом.

Окружность может быть аппроксимирована путем соединения полученных точек, отрезками линии (рис. 9.4).

Если мы хотим аппроксимировать окружность по точкам, то следует выбирать угловой шаг, равный 1/*r*.

Можно избежать расчета синусоидальных и косинусных функций для каждой вычисляемой точки, зная, что между каждой из двух последовательных точек существует одинаковое угловое расстояние.

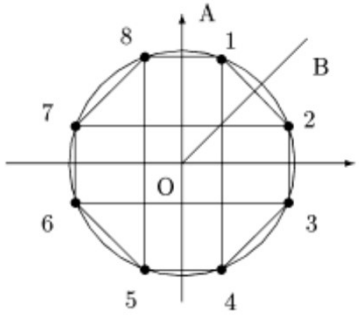


Рис. 9.4. Аппроксимация окружности

9.2.2. Вычисление точек на окружности с использованием параметрических уравнений окружности

Пусть (x_i, y_i) последняя вычисляемая точка и (x_{i+1}, y_{i+1}) следующая точка :

$$x_i = xc + \mathrm{dx}_i,\ dx_i = r \cdot \cos(t),$$
$$y_i = yc + dy_i,\ dy_i = r \cdot \sin(t), \tag{9.16}$$

$$x_{i+1} = xc + dx_{i+1},$$
$$dx_{i+1} = r \cdot \cos(t + \mathrm{pas}), \tag{9.17}$$

$$y_{i+1} = yc + dy_{i+1},$$
$$dy_{i+1} = r \cdot \sin(t + \mathrm{pas}). \tag{9.18}$$

Но мы знаем, что:

$$\cos(t + \mathrm{pas}) = \cos(t)\cdot\cos(\mathrm{pas}) - \sin(t)\cdot\sin(\mathrm{pas}),$$
$$\sin(t + \mathrm{pas}) = \cos(t)\cdot\sin(\mathrm{pas}) + \sin(t)\cdot\cos(\mathrm{pas}). \tag{9.19}$$

Если $c = \cos(\mathrm{pas})$ и $s = \sin(\mathrm{pas})$. Эти значения вычисляются только один раз при создании круга.
Следующие отношения повторения приводят к вычислению точек на окружности:

$$dx_{i+1} = dx_i \cdot c - dy_i \cdot s,$$
$$dy_{i+1} = \mathrm{dx}_i \cdot \mathrm{s} - dy_i \cdot c. \tag{9.20}$$

9.2.2. Расчет точек в окружности с использованием параметрических уравнений окружности

Время, необходимое для вычисления точек на окружности, может быть существенно уменьшено с учетом симметрии точек на окружности.

Таким образом, достаточно вычислить описанным выше методом координаты точек в октанте.

Каждая вычисляемая точка соответствует еще 7 симметриям.

– Например, в функции cerc_sim() вычисляются только точки первого октанта;

– для каждой вычисляемой точки вызывается функция punct_simetric(), которая отображает не только вычисляемую точку, но и ее симметрию (по отношению к углу *u*).

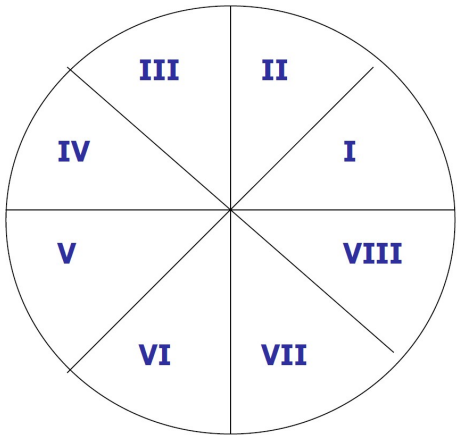


Рис. 9.5. Симметрия точек на окружности

9.2.3. Построение кругов в дискретном пространстве. Алгоритм Брезенхама

Алгоритм Брезенхэма использует инкрементальный метод и осевую погрешность в качестве меры приближения выбранной точки к теоретическому кругу.

Рассмотрим окружность с центром в начале системы координат и радиусом r.

В алгоритме вычисляются только точки второго октанта, остальные получаются симметрией.

Пусть (x_i, y_i) последняя точка дискретного пространства, выбранная для аппроксимации окружности.

Следующим пунктом будет один из (x_{i+1}, y_i) и (x_{i+1}, y_{i-1}).

Ордината точки на теоретической окружности получается из уравнения окружности в декартовых координатах:

$$y^2 = r^2 - (x_i + 1)^2 . \qquad (9.21)$$

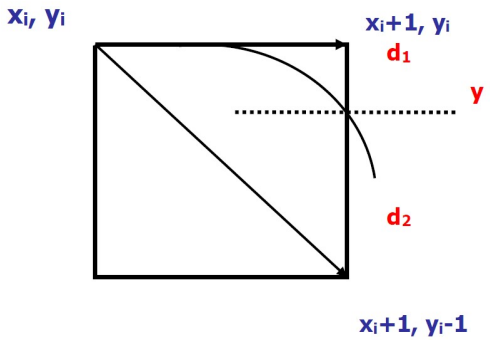


Рис. 9.6. Представление окружности по алгоритму Брезенхама

9.2.3. Построение кругов в дискретном пространстве. Алгоритм Брезенхама

Рассчитываются расстояния между точками на окружности и двумя точками-кандидатами:

$$d1 = y_i^2 - y^2 = y_i^2 - r^2 + (x_i + 1)^2, \quad (9.22)$$

$$d2 = y^2 - (y_i - 1)^2 = r^2 - (x_i + 1)^2 - (y_i - 1)^2.$$

Обозначаем через t_i погрешность приближения на текущем шаге:

$$t_i = d1 - d2 = y_i^2 + 2(x_i + 1)^2 + (y_i - 1)^2 - 2r^2. \quad (9.23)$$

Далее получается отношение для вычисления погрешности приближения на следующем шаге:

$$t_{i+1} = y_{i+1}^2 + 2(x_{i+1} + 1)^2 + (y_{i+1} - 1)^2 - 2r^2. \quad (9.24)$$

1) Если $t_i < 0$, тогда $x_{i+1} = x_i + 1$ и $y_{i+1} = y_i$.

Таким образом, $t_{i+1} = y_i^2 + 2\big((x_i + 1) + 1\big)^2 + (y_i - 1)^2 - 2r^2,$

или $t_{i+1} = t_i + 4 \cdot x_i + 6.$

2) Если $t_i \geq 0$, тогда $x_{i+1} = x_i + 1$ и $y_{i+1} = y_i - 1$.

Таким образом, $t_{i+1} = (y_i - 1)^2 + 2\big((x_i + 1) + 1\big)^2 + \big((y_i - 1) - 1\big)^2 - 2r^2,$

или $t_{i+1} = t_i + 4 \cdot (x_i - y_i) + 10.$

Значение погрешности на первом шаге t_i получается путем замены в его выражении x_i нас $x_1 = 0$ и y_i на $y_1 = r$.

Таким образом: $t_1 = 3 - 2 \cdot r.$

ВОПРОСЫ