**Лабораторная работа №4**

**Тема**: Растровые и векторные примитивы. Алгоритмы генерации векторных примитивов

**Цель работы:** Анализ различных алгоритмов генерации отрезков на прямоугольном растре.

**Теоретический материал.**

Одним из графических примитивов в двумерной графике является отрезок прямой линии, задаваемый координатами своих концевых точек. Отображение таких отрезков в растровой графике отнюдь не просто. Связано это с тем, что изображение отрезка на прямоугольном растре является совокупностью горизонтальных, либо вертикальных отрезков и отдельных точек рис. 4.1.

Задачей алгоритма генерации отрезков прямых линий является определение координат пикселей, которые необходимо подсветить для получения изображения заданного отрезка. Существует несколько алгоритмов генерации отрезков. Эти алгоритмы выбирают пиксели так, чтобы их координаты были наиболее близки к координатам точек, принадлежащих реальному отрезку. Наибольшее распространение в компьютерной графике получили инкрементные алгоритмы генерации отрезков. В этих алгоритмах, отрезок рисуется последовательно пиксель за пикселем. Координаты

следующего пикселя определяются исходя из координат текущего, и некоторой дополнительной информации. Рассмотрим два инкрементных алгоритма.

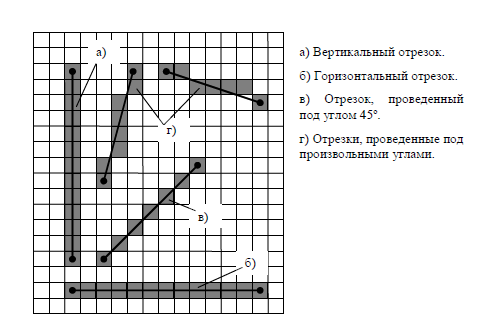


Рис. 4. Растровое представление отрезков прямых линий

***Алгоритм 1 Цифровой дифференциальный анализатор***

Алгоритм **Цифровой дифференциальный анализатор** (англ. DDA - Digital Differential Analyzer) строит 8-связную линию.

Для начала, пусть P1 = (1, 0) ; P2 = (1, 1). Для определения того, какой из пикселей, - P1 или P2, - следует закрасить, сравним расстояния до них. В силу подобия треугольников, образованных пересечением рисуемого отрезка, прямой x = 1 и перпендикулярами из P1 и P2 на отрезок ([см. рис.4.2](http://www.intuit.ru/studies/courses/993/163/lecture/4495?page=1#image.3.4)), достаточно сравнить e (ординату пересечения отрезка c прямой x = 1 ) с \frac{1}{2}. Далее, для следующего шага алгоритм работает аналогично с учетом изменения e - ординаты пересечения отрезка со следующей вертикальной прямой x = k, k \in \mathbb{N}.

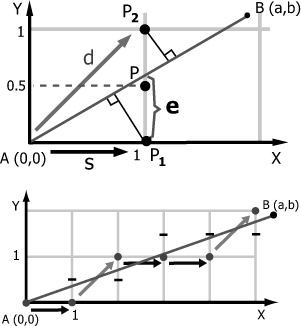


Рис. 4. Цифровой дифференциальный анализатор.

// Координаты концов отрезка - (0,0) и (a,b)

e = b/a; // Текущая ордината

delta\_e = b/a; // Приращение ординаты

// (x,y) - Координаты текущей точки

x = 0; y = 0;

while( x < a )

{

рисуем пиксель с координатами(x, y);

if( e > 1/2 )

{

// d : диагональное смещение

x++; y++;

// т.к. произошло смещение по y на 1 вверх

e += delta\_e - 1;

}

else

{

// s : горизонтальное смещение

x++;

e += delta\_e;

}

}

Листинг4.1. Цифровой дифференциальный анализатор

Недостатком данного алгоритма является то, что он работает с числами с плавающей точкой.

***Алгоритм 2 Несимметричный цифровой дифференциальный анализатор (несимметричный ЦДА)***

В основу работы этого алгоритма положен тот факт, что производная функции, представляющей прямую линию на плоскости, является величиной постоянной, т.е. . Если у нас имеется отрезок прямой, заданный координатами своих концевых точек (x1, y1) и (x2, y2 ), то величину этой производной можно рассчитать следующим образом: .

Алгоритм «несимметричный ЦДА» использует эту величину, чтобы рассчитать значение смещения по одной из координат, при смещении по другой на 1. Этот алгоритм можно описать следующей последовательностью действий (подразумевается, что хотя бы одно из значений Δx или Δy не равно нулю).

1. Рассчитаем величину d и определим, ось, вдоль которой будем продвигаться, следующим образом:

Если Δx = 0 , то наш отрезок является вертикальным, тогда d = 0, продвигаться будем вдоль оси y .

Если Δy = 0, то наш отрезок является горизонтальным, тогда d = 0 , и продвигаться будем вдоль оси x .

Если Δx ≥Δy , тогда, d=продвигаться будем вдоль оси x .

В противном случае, d=и продвигаться будем вдоль оси y .

1. Подсветим пиксель с координатами (x1, y1).
2. Увеличим координату, соответствующую оси, вдоль которой продвигаемся, на 1  
   (-1), а другую на d , и подсветим пиксель, соответствующий этим координатам.
3. Повторять пункт 3, до тех пор, пока не нарисуем весь отрезок.

Рассмотрим работу алгоритма на следующих примерах рис. 4.3.

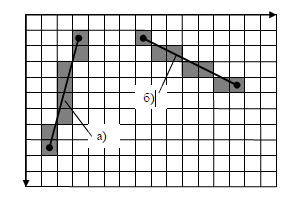


Рис. 4. Генерация отрезков несимметричным ЦДА.

***Отрезок а): x1=3, y1=1, x2=1, y2=8***

1. Поскольку Δy >Δx, то d=, продвигаться будем вдоль оси y.
2. Подсветим пиксель с координатами (3, 1).
3. Увеличим y1 на 1, а x1 на . Получим y1=2, x1=. Подсветим пиксель с координатами (3 (),2).
4. Далее повторяем пункт 3, пока не нарисуем весь отрезок.

На следующем шаге получим y1=3, x1=. Подсветим пиксель с координатами (3, 2). Далее получим y1=4, x1=. Подсветим пиксель с координатами (4, 2) и т.д.

***Отрезок б): x1=7, y1=1, x2=13, y2=4.***

1. Поскольку Δx >Δy, то d= , продвигаться будем вдоль оси x.
2. Подсветим пиксель с координатами (7, 1).
3. Увеличим x1 на 1, а y1 на 0,5. Получим y1=1,5, x1=8 . Подсветим пиксель с координатами (8, 2(1,52)).
4. Далее повторяем пункт 3, пока не нарисуем весь отрезок.

На следующем шаге получим y1=2, x1=9. Подсветим пиксель с координатами (9, 2). Далее получим y1=2,5, x1=10. Подсветим пиксель с координатами (10, 3) и т.д.

***Алгоритм 3 Алгоритм Брезенхема***

Рассмотренный выше алгоритм неудобен тем, что в общем случае нам приходится использовать вещественные переменные и операцию деления, что отрицательно сказывается на его быстродействии. Однако можно разработать такой алгоритм генерации отрезков, в котором будет использоваться только целочисленная арифметика, и не будет операций деления и умножения. Быстродействие такого алгоритма также возрастет. Данный алгоритм получил название «алгоритм Брезенхема» по фамилии его автора.

Будем разрабатывать алгоритм нахождения координат точек между A и B использующий вещественные числа, но таким образом, что это ограничение потом можно будет легко устранить, и использовать только целочисленные переменные. Для этого выполним следующие операции.

Начнем рисовать отрезок с точки А( ***Ax, Ay*** ). В цикле будем задавать приращение 1 для переменной ***Ay*** и будем оставлять ***Ax*** либо неизменным, либо также увеличивать на 1. При этом последний выбор будем осуществлять так, чтобы новая точка сетки ( ***Ax, Ay*** ) располагалась как можно ближе к прямой линии, проходящей через точки А и B рис. 4.4.

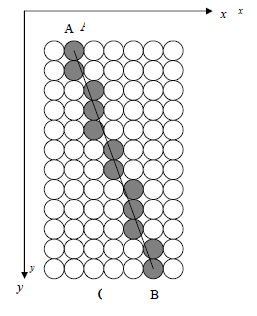


Рис. 4. Пример отрезка

Это означает, что расстояние по горизонтали между новой выбранной точкой и этой линией не должно превышать значения 0,5.

Введем переменную d для обозначения этого расстояния. Потребуем, чтобы .

Последнее неравенство обеспечивает условие для определения необходимости давать приращение переменной ***Ax*** .

Теперь можно написать первый вариант реализации алгоритма нахождения координат точек между A и В:

1. Установим значения для переменных d и t: d = 0, t=.
2. Подсветим пиксель с координатами ( ***Ax, Ay*** ).
3. Увеличим d на величину t, а ***Ay*** на 1 (-1).
4. Если d > 0,5 , то увеличим ***Ax*** на 1, и вычтем 1 из d.
5. Будем повторять пункты 2-4, пока не нарисуем весь отрезок.

Отклонение, вначале устанавливается равным нулю и изменяется на каждом шаге. Поскольку оно показывает, насколько левее точной прямой линии лежит вычисленная точка, то значение ***d*** увеличивается на значение ***t***, если y увеличивается на 1 и x остается без изменения. Это условие не выполняется, если значение ***d*** превышает 0,5. В этот момент нужно увеличить значение ***x*** на 1. Соответственно, и отклонение ***d*** должно быть уменьшено на единицу.

Теперь посмотрим, как можно избавиться от вещественных значений переменных t и d и постоянной 0,5.

Значение переменной t вычисляется следующим образом:



где числитель и знаменатель представляют собой целые числа.

Величина ***d*** вычисляется как конечная сумма элементов, каждый из которых равен либо t, либо -1. Поэтому ***d*** также можно записать в виде частного со знаменателем равным ***Ay - By*** .

Значит, можно перейти к целочисленным переменным ***t*** и ***d***, путем умножения на значение знаменателя. Также просто избавиться от константы 0,5. Для этого нужно дополнительно умножить значение знаменателя на 2. Эта операция может быть заменена простым поразрядным сдвигом на 1 бит влево. Таким образом t = 2Δx , где Δx = Ax - Bx .

Напишем новый вариант реализации нашего алгоритма.

1. Установим значения для переменных ***d, t*** ***и Δ***: ***d*** = 0 , ***t*** = 2Δx = 2(***Ax*** - ***Bx***),   
   ***Δ =*** 2Δy = 2(***Ay*** ***- By***) .
2. Подсветим пиксель с координатами ( ***Ax, Ay*** ).
3. Увеличим d на величину ***t***, а ***Ay*** на 1.
4. Если ***d > Δy*** , то увеличим Ax на 1, и вычтем ***Δ*** из ***d***.
5. Будем повторять пункты 2-4, пока не нарисуем весь отрезок.

Однако наша функция будет правильно работать только для того случая, который изображен на рис. 4.4. Для того, чтобы алгоритм правильно работал для любого отрезка, его необходимо дополнить. В результате получим следующий алгоритм:

1. Отсортируем вершины А и В так, чтобы выполнялось условие ***Ay ≤ By*** .
2. Рассчитаем значения ***Δx = Bx - Ax*** , и ***Δy = By - Ay*** .
3. Если ***Δx ≥ 0*** , то установим значение dx =1, иначе: dx = -1, и изменим знак у Δx на противоположный.
4. Установим ***d = 0*** . Если ***Δy ≥ Δx*** , то установим ***t = 2Δx*** , ***Δ= 2Δy*** . Иначе ***t = 2Δy***, ***Δ= 2Δx*** . (операцию умножения на 2 везде заменяем поразрядным сдвигом влево на 1 бит).
5. Если ***Δy ≥Δx*** , то перейти к пункту 6, иначе к пункту 11.
6. Подсветим пиксель с координатами ( ***Ax, Ay*** ).
7. Увеличим ***Ay*** на 1, а d на величину t.
8. Если ***d > Δy*** , то увеличить ***Ax*** на величину ***dx*** , и вычесть величину ***Δ*** из ***d*** .
9. Повторять пункты 6-8, пока не нарисуем весь отрезок.
10. Завершить выполнение алгоритма.
11. Подсветим пиксель с координатами ( ***Ax, Ay*** ).
12. Увеличим ***Ax*** на 1, а ***d*** на величину ***t*** .
13. Если ***d > Δx*** , то увеличить ***Ay*** на 1, и вычесть величину ***Δ*** из ***d*** .
14. Повторять пункты 6-8, пока не нарисуем весь отрезок.
15. Завершить выполнение алгоритма.

При построении растрового образа отрезка необходимо, прежде всего, установить критерии "хорошей" аппроксимации. Первое требование состоит в том, что отрезок должен начинаться и кончаться в заданных точках и при этом выглядеть сплошным и прямым (при достаточно высоком разрешении дисплея этого можно добиться). Кроме того, яркость вдоль отрезка должна быть одинаковой и не зависеть от наклона отрезка и его длины. Это требование выполнить сложнее, поскольку горизонтальные и вертикальные отрезки всегда будут ярче наклонных, а постоянная яркость вдоль отрезка опять же достигается на вертикальных, горизонтальных и наклоненных под углом в 45 \degлиниях. И, наконец, алгоритм должен работать быстро. Для этого необходимо по возможности исключить операции с вещественными числами. С целью ускорения работы алгоритма можно также реализовать его на аппаратном уровне.

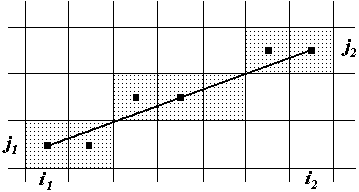


Рис. 4. Растровый образ отрезка

В большинстве алгоритмов используется пошаговый метод изображения, т.е. для нахождения координат очередной точки растрового образа наращивается значение одной из координат на единицу растра и вычисляется приращение другой координаты.

Задача состоит в построении отрезка, соединяющего на экране точки с координатами (i_1,j_1),(i_2,j_2)(будем считать, что i_1\ne i_2). Для построения отрезка прямой на плоскости с вещественными координатами можно воспользоваться уравнением прямой, проходящей через две заданные точки, которое имеет вид

j=j_1+k(i-i_1), \quad k=(j_2-j_1)/(i_2-i_1).

Теперь, считая, что 0\le k\le 1, \; (i,j)- координаты текущей точки растрового образа, а y- точное значение координаты точки отрезка, можно построить следующую точку:

i'=i+1, \quad j'=y+k

Следует заметить, что целочисленная координата jизменится только в том случае, если y превысит величину j+0.5( j'есть ближайшее к yцелое число, полученное в результате операции округления). Приведенный пример включает операции с вещественными числами, которые выполняются существенно медленнее, чем соответствующие целочисленные операции, а при построении растрового образа отрезка желателен алгоритм, по возможности обращающийся только к целочисленной арифметике. Кроме того, алгоритм должен работать при любом взаимном расположении концов отрезка.

Алгоритм Брезенхема построения растрового образа отрезка был изначально разработан для графопостроителей, но он полностью подходит и для растровых дисплеев. В процессе работы в зависимости от углового коэффициента отрезка наращивается на единицу либо i, либо j, а изменение другой координаты зависит от расстояния между действительным положением точки и ближайшей точкой растра (смещения). Алгоритм построен так, что анализируется лишь знак этого смещения.

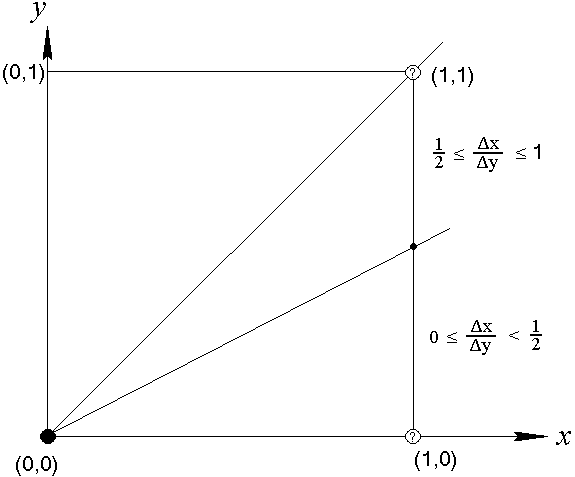


Рис. 4. Связь углового коэффициента с выбором пикселя

На [рис. 4.](http://www.intuit.ru/studies/courses/70/70/lecture/2106?page=1#image.8.2)4 это иллюстрируется для отрезка с угловым коэффициентом, лежащим в диапазоне от нуля до единицы. Из рисунка можно заметить, что если угловой коэффициент k\ge\frac12, то при выходе из точки (0,0)пересечение с прямой x=1будет ближе к прямой y=1, чем к прямой y=0. Следовательно, точка растра (1,1)лучше аппроксимирует прохождение отрезка, чем точка (1,0). При k<\frac12верно обратное.

На [рис. 4.5](http://www.intuit.ru/studies/courses/70/70/lecture/2106?page=1#image.8.3) показано, каким образом строятся точки растра для отрезка с тангенсом угла наклона 3/8, а на [рис. 4.6](http://www.intuit.ru/studies/courses/70/70/lecture/2106?page=1#image.8.4) - график смещения. В начале построения смещение полагается равным k-1/2, а затем на каждом шаге оно наращивается на величину k, и если при этом вертикальная координата точки растра увеличивается на единицу, то смещение в свою очередь уменьшается на единицу.

На [рис. 4.7](http://www.intuit.ru/studies/courses/70/70/lecture/2106?page=1#image.8.5) приведена блок-схема алгоритма для случая i_2>i_1, \quad j_2> j_1, \quad k>0. Нетрудно понять, как от этого алгоритма перейти к целочисленному: достаточно вместо величины смещения eперейти к величине \widetilde{e}=2\Delta i\cdot e.

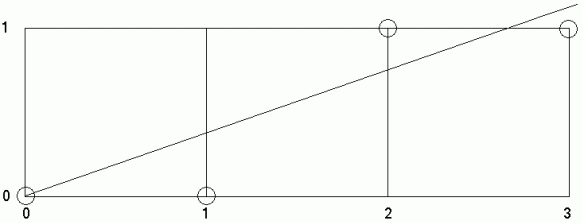


Рис. 4. Пиксели, принадлежащие развертке отрезка

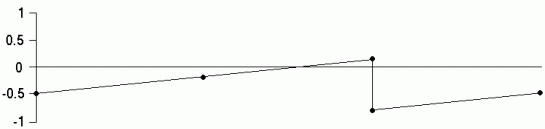


Рис. 4. График изменения отклонения

Приведем общий алгоритм Брезенхема, который учитывает все возможные случаи направления отрезка, рассматриваемого как вектор на координатной плоскости (на [рис. 4.9](http://www.intuit.ru/studies/courses/70/70/lecture/2106?page=1#image.8.6) выделены четыре области и указаны особенности алгоритма в каждой из них).

В описании алгоритма используются следующие функции:

swap (a, b): обмен значений переменных a, b;

abs (a): абсолютное значение a;

sign (a): 0, если a= 0, 1, если a>0, –1, если a<0;

point (i, j) - инициализация точки (i, j).

Предполагается, что концы отрезка (i_1,j_1),(i_2,j_2)не совпадают и что все используемые переменные являются целыми.

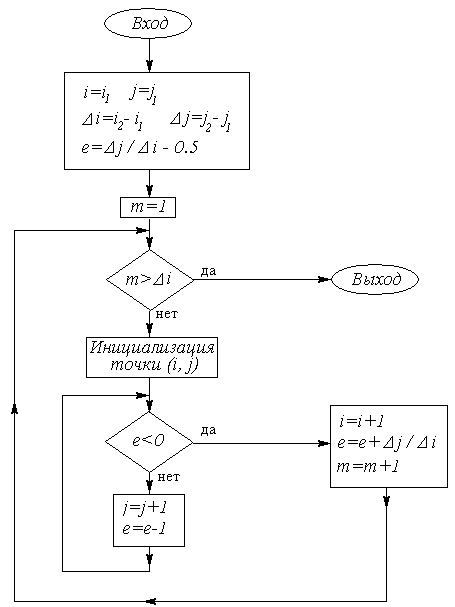


Рис. 4. Блок-схема одной ветви алгоритма Брезенхема

i=i1;

j=j1;

di=i2-i1;

dj=j2-j1;

s1=sign(i2-i1);

s2=sign(j2-j1);

di=abs(di);

dj=abs(dj);

if (dj>di)

{

swap(di,dj); c=1;

}

else c=0;

e=2\*dj-di; // Инициализация смещения

// Основной цикл

for (l=0; l<di; l++)

{

point(i,j);

while (e>=0)

{

if (c==1) i=i+s1;

else j=j+s2;

e=e-2\*di;

}

if (c==1) j=j+s2; else i=i+s1;

e=e+2\*dj;

}

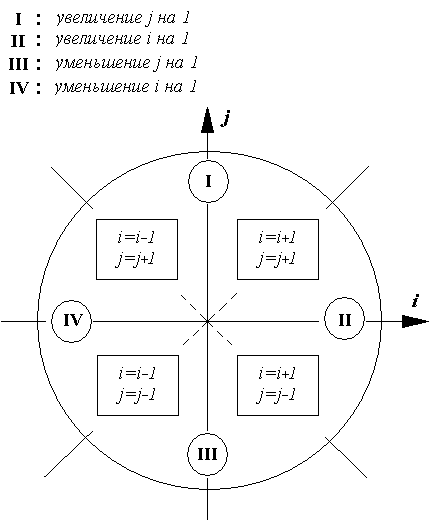


Рис. 4. 10 Четыре возможных направления отрезка

**Задания.**

Все задания к лабораторной работе №4 необходимо выполнять, используя псевдопиксель, в виде закрашенного сплошным цветом квадрата размером n х n пикселей. Для выполнения заданий необходимо в первую очередь реализовать соответствующие алгоритмы генерации отрезков на используемом языке программирования. Затем, используя эти алгоритмы, выполнить соответствующий вариант задания.

***При выполнении вариантов заданий ЗАПРЕЩАЕТСЯ пользоваться встроенными функциями рисования отрезков!***

|  |  |
| --- | --- |
| Вариант | Рисунок |
|  | Нарисовать красным цветом наибольшее изображение икосаэдра, которое поместится в области вывода рисунка. Ребра икосаэдра, наиболее близкие к наблюдателю, (жирные линии) нарисовать, используя алгоритм  несимметричный ЦДА и псевдопиксель размером 20х20 пикселей. Ребра дальние необходимо нарисовать, используя алгоритм Брезенхема и псевдопиксель размером 10х10 пикселей.  Дополнительное задание: Добейтесь, чтобы размеры рисунка изменялись при изменении размеров области вывода (окна). |
|  | На базе алгоритма ЦДА разработать алгоритм рисования пунктирной линии. Пунктирная линия состоит из отрезков, разделенных промежутками. Длина отрезка не может быть больше 10 псевдопикселей. Длина промежутка составляет половину длины отрезка. Линия начинается и заканчивается отрезком. Все отрезки и промежутки должны иметь примерно одинаковую длину. Используя алгоритм Брезенхема и псевдопиксель 20х20, нарисовать зеленым цветом наибольший прямоугольник, который поместится в области вывода. Диагонали прямоугольника провести пунктирной линией красного цвета, используя псевдопиксель 5х5.  Дополнительное задание: Добейтесь, чтобы размеры рисунка изменялись при изменении размеров области вывода (окна). |
|  | Нарисовать красным цветом наибольшее изображение куба, которое поместится в области вывода рисунка. Ребра куба наиболее близкие к наблюдателю (жирные линии) нарисовать, используя алгоритм Брезенхема и  псевдопиксель размером 20х20 пикселей. Стороны дальние необходимо нарисовать, используя алгоритм несимметричный ЦДА и псевдопиксель размером 10х10 пикселей.  Дополнительное задание: Добейтесь, чтобы размеры рисунка изменялись при изменении размеров области вывода (окна). |
|  | Нарисовать зеленым цветом наибольшее изображение октаэдра, которое поместится в области вывода рисун-ка. Стороны октаэдра наиболее близкие к наблюдате-лю (жирные линии) нарисовать, используя алгоритм ЦДА и псевдопиксель размером 20х20 пикселей. Стороны дальние необходимо нарисо-вать, используя алгоритм Брезенхема и псевдопиксель размером 10х10 пикселей.  Дополнительное задание: Добейтесь, чтобы размеры рисунка изменялись при изменении размеров области вывода (окна). |
|  | Нарисовать наибольшую шестилучевую звезду, которая поместится в области вывода рисунка. Стороны верхнего треугольника необходимо нарисовать зеленым цветом, используя алгоритм несимметричный ЦДА и псевдопиксель размером 10х10 пикселей. Стороны нижнего треугольника необходимо нарисовать красным цветом, используя алгоритм Брезенхема и  псевдопиксель размером 20х20 пикселей.  Дополнительное задание: Добейтесь, чтобы размеры рисунка изменялись при изменении размеров области вывода (окна). |
|  | Нарисовать наибольшую пятиконечную звезду, которая поместится в области вывода и пятиугольник внутри нее.. Звезду необходимо нарисовать красным цветом, используя алгоритм Брезенхема и псевдопиксель размером 20х20 пикселей. Пятиугольник нарисовать синим цветом алгоритмом ЦДА псевдопикселями размером 10х10 пикселей.  Дополнительное задание Добейтесь, чтобы размеры рисунка изменялись при изменении размеров области вывода (окна). |
|  | Нарисовать наибольший прямоугольник с соотношением сторон 2/3, который поместится в области вывода и построить все его диагонали. Прямоугольник расположить по центру экрана. Короткие стороны необходимо расположить параллельно горизонтальной стороне экрана. Прямоугольник необходимо нарисовать красным цветом, используя алгоритм несимметричный  ЦДА и псевдопиксель размером 20х20 пикселей. Диагонали изобразить двумя различными цветами (зеленым и синим) алгоритмом Брезенхема псевдопикселями размером 10х10 пикселей.  Дополнительное задание Добейтесь, чтобы размеры рисунка изменялись при изменении размеров области вывода (окна). |
|  | Нарисовать наибольший правильный шестиугольник, который поместится в области вывода и построить все его диагонали. Стороны шестиугольника необходимо нарисовать оранжевым цветом, используя алгоритм ЦДА и псевдопиксель размером 20х20 пикселей. Диагонали изобразить тремя различными цветами (красным зеленым и синим) алгоритмом Брезенхема псевдопикселями размером 10х10 пикселей.  Дополнительное задание Добейтесь, чтобы размеры рисунка изменялись при изменении размеров области вывода (окна). |
|  | Нарисовать наибольший равносторонний треугольник, который поместится в области вывода. Одна из сторон треугольника должна идти вдоль нижней границы области вывода. Боковые стороны необходимо нарисовать красными линиями, используя алгоритм Брезенхема. Основание нарисовать зеленой линией алгоритмом несимметричного ЦДА. Размер псевдопикселя для всех отрезков составляет 20х20 пикселей.  Дополнительное задание Добейтесь, чтобы размеры рисунка изменялись при изменении размеров области вывода (окна). |
|  | Нарисовать красным цветом наибольшее изображение икосаэдра, которое поместится в области вывода ри-сунка. Ребра икосаэдра, наиболее близкие к наблюда-телю, (жирные линии) нарисовать, используя алгоритм  ЦДА и псевдопиксель размером 20х20 пикселей. Ребра дальние необходимо нарисо-вать, используя алгоритм Брезенхема и псевдопиксель размером 10х10 пикселей.  Дополнительное задание Добейтесь, чтобы размеры рисунка изменялись при изменении размеров области вывода (окна). |
|  | На базе алгоритма несимметричный ЦДА разработать алгоритм рисования пунктирной линии. Пунктирная линия состоит из отрезков, разделенных промежутка-ми. Длина отрезка не может быть больше 10 псев-допикселей. Длина промежутка составляет половину длины отрезка. Линия начинается и заканчивается от-резком. Все отрезки и промежутки должны иметь при-мерно одинаковую длину. Используя алгоритм Бре-зенхема и псевдопиксель 20х20, нарисовать зеленым цветом наибольший прямоугольник, который поме-стится в области вывода. Диагонали прямоугольника провести пунктирной линией красного цвета, исполь-зуя псевдопиксель 5х5.  Дополнительное задание: Добейтесь, чтобы размеры рисунка изменялись при изменении размеров области вывода (окна). |
|  | Нарисовать красным цветом наибольшее изображение куба, которое поместится в области вывода рисунка. Ребра куба наиболее близкие к наблюдателю (жирные линии) нарисовать, используя алгоритм Брезенхема и  псевдопиксель размером 20х20 пикселей. Стороны дальние необходимо нарисовать, используя алгоритм ЦДА и псевдопиксель размером 10х10 пикселей.  Дополнительное задание: Добейтесь, чтобы размеры рисунка изменялись при изменении размеров области вывода (окна). |
|  | Нарисовать зеленым цветом наибольшее изображение октаэдра, которое поместится в области вывода рисунка. Стороны октаэдра наиболее близкие к наблюдателю (жирные линии) нарисовать, используя алгоритм несимметричный ЦДА и псевдопиксель размером 20х20 пикселей. Стороны дальние необходимо нарисовать, используя алгоритм Брезенхема и псевдопиксель размером 10х10 пикселей.  Дополнительное задание: Добейтесь, чтобы размеры рисунка изменялись при изменении размеров области вывода (окна). |
|  | Нарисовать наибольшую шестилучевую звезду, кото-рая поместится в области вывода рисунка. Стороны верхнего треугольника необходимо нарисовать зеле-ным цветом, используя алгоритм ЦДА и псевдопиксель размером 10х10 пикселей. Стороны нижнего треугольника необходимо нарисовать красным цветом, используя алгоритм Брезенхема и  псевдопиксель размером 20х20 пикселей.  Дополнительное задание: Добейтесь, чтобы размеры рисунка изменялись при изменении размеров области вывода (окна). |
|  | Нарисовать наибольшую пятиконечную звезду, кото-рая поместится в области вывода и пятиугольник внутри нее.. Звезду необходимо нарисовать красным цветом, используя алгоритм Брезенхема и псевдопик-сель размером 20х20 пикселей. Пятиугольник нарисо-вать синим цветом алгоритмом несимметричный ЦДА псевдопикселями размером 10х10 пикселей.  Дополнительное задание Добейтесь, чтобы размеры рисунка изменялись при изменении размеров области вывода (окна). |
|  | Нарисовать наибольший прямоугольник с соотноше-нием сторон 2/3, который поместится в области вывода и построить все его диагонали. Прямоугольник расположить по центру экрана. Короткие стороны необходимо расположить параллельно горизонтальной стороне экрана. Прямоугольник необходимо нарисовать красным цветом, используя алгоритм  ЦДА и псевдопиксель размером 20х20 пикселей. Диа-гонали изобразить двумя различными цветами (зеле-ным и синим) алгоритмом Брезенхема псевдопикселя-ми размером 10х10 пикселей.  Дополнительное задание Добейтесь, чтобы размеры рисунка изменялись при изменении размеров области вывода (окна). |
|  | Нарисовать наибольший правильный шестиугольник, который поместится в области вывода и построить все его диагонали. Стороны шестиугольника необходимо нарисовать оранжевым цветом, используя алгоритм несимметричный ЦДА и псевдопиксель размером 20х20 пикселей. Диагонали изобразить тремя различ-ными цветами (красным зеленым и синим) алгоритмом Брезенхема псевдопикселями размером 10х10 пикселей.  Дополнительное задание Добейтесь, чтобы размеры рисунка изменялись при изменении размеров области вывода (окна). |
|  | Нарисовать наибольший равносторонний треугольник, который поместится в области вывода. Одна из сторон треугольника должна идти вдоль нижней границы области вывода. Боковые стороны необходимо нарисовать красными линиями, используя алгоритм Брезенхема. Основание нарисовать зеленой линией алгоритмом ЦДА. Размер псевдопикселя для всех отрезков составляет 20х20 пикселей.  Дополнительное задание Добейтесь, чтобы размеры рисунка изменялись при изменении размеров области вывода (окна). |
|  | Нарисовать красным цветом наибольшее изображение икосаэдра, которое поместится в области вывода ри-сунка. Ребра икосаэдра, наиболее близкие к наблюда-телю, (жирные линии) нарисовать, используя алгоритм  несимметричный ЦДА и псевдопиксель размером 20х20 пикселей. Ребра дальние необходимо нарисо-вать, используя алгоритм Брезенхема и псевдопиксель размером 10х10 пикселей.  Дополнительное задание: Добейтесь, чтобы размеры рисунка изменялись при изменении размеров области вывода (окна). |
|  | На базе алгоритма ЦДА разработать алгоритм рисования пунктирной линии. Пунктирная линия состоит из отрезков, разделенных промежутка-ми. Длина отрезка не может быть больше 10 псев-допикселей. Длина промежутка составляет половину длины отрезка. Линия начинается и заканчивается от-резком. Все отрезки и промежутки должны иметь при-мерно одинаковую длину. Используя алгоритм Бре-зенхема и псевдопиксель 20х20, нарисовать зеленым цветом наибольший прямоугольник, который поме-стится в области вывода. Диагонали прямоугольника провести пунктирной линией красного цвета, исполь-зуя псевдопиксель 5х5.  Дополнительное задание: Добейтесь, чтобы размеры рисунка изменялись при изменении размеров области вывода (окна). |
|  | Нарисовать красным цветом наибольшее изображение куба, которое поместится в области вывода рисунка. Ребра куба наиболее близкие к наблюдателю (жирные линии) нарисовать, используя алгоритм Брезенхема и  псевдопиксель размером 20х20 пикселей. Стороны дальние необходимо нарисовать, используя алгоритм несимметричный ЦДА и псевдопиксель размером 10х10 пикселей.  Дополнительное задание: Добейтесь, чтобы размеры рисунка изменялись при изменении размеров области вывода (окна). |
|  | Нарисовать зеленым цветом наибольшее изображение октаэдра, которое поместится в области вывода рисунка. Стороны октаэдра наиболее близкие к наблюдателю (жирные линии) нарисовать, используя алгоритм ЦДА и псевдопиксель размером 20х20 пикселей. Стороны дальние необходимо нарисовать, используя алгоритм Брезенхема и псевдопиксель размером 10х10 пикселей.  Дополнительное задание: Добейтесь, чтобы размеры рисунка изменялись при изменении размеров области вывода (окна). |