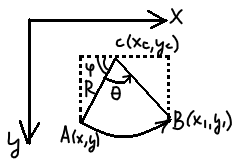
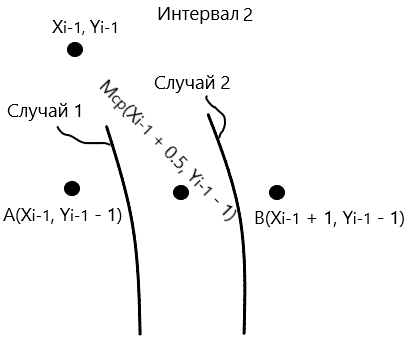
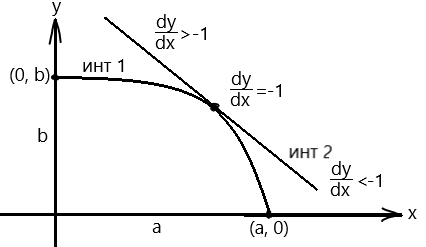
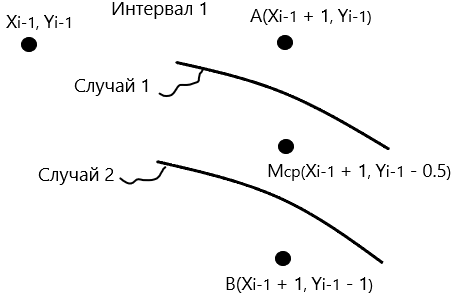
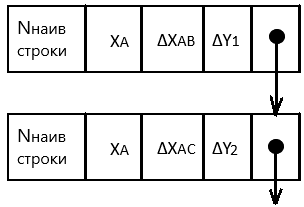
1.Задача синтеза сложного динамического изображения. Этапы синтеза изображения. Последовательность и основное содержание.  
В общем случае задача синтеза изображения трехмерных объектов представляет собой задачу имитации визуальной обстановки, т.е. искусственного построения изображений окружающей среды с такой степенью достоверности, которая достаточна для выработки и поддержания у пользователя программы навыков управления подвижными объектами. Таким образом, синтезируемое изображение должно быть динамическим и как можно более реалистичным. Оптические характеристики: цвет, коэффициенты диффузного отражения, прозрачности, зеркального отражения. Интенсивность света I = kr \* Ir + kg \* Ig + kb \* Ib. Изображение строим в картинной плоскости (расположена перпендикулярно взгляду), на ней задаётся окно обзора (ось z через центр окна обзора). f = 30-50 Гц – частота генерации изображения. Задача синтеза изображения состоит в том, чтобы для каждого временного интервала сформировать изображение объектов (или их частей) окружающей среды в плоскости обзора с учетом приемов имитации трехмерного пространства.  
Этапы: 1)Разработка трёхмерной математической модели объектов визуальной обстановки. 2)Определение: направления линии визирования, положения картинной плоскости, размеров окна обзора, значений управляющих сигналов. 3)Формирование управляющих операторов, определяющих пространственное перемещение объектов обстановки. 4)Преобразование модели синтезируемой обстановки к системе координат, связанной с наблюдателем. 5)Отсечение объектов обстановки в пределах пирамиды видимости. 6)Вычисление двумерных перспективных проекций объектов визуализации на картинную плоскость. 7)Исключение невидимых элементов синтезируемой обстановки, закрашивание, затенение видимых участков. 8)Вывод полученного изображения на экран растрового дисплея.

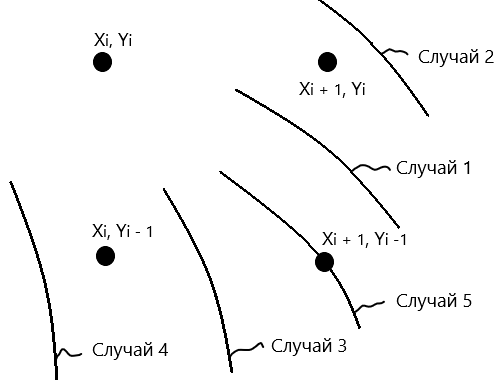
2. Преобразования на плоскости. Вывод расчетных соотношений. Матрицы преобразований.  
т.А(x, y) -> т.В(x1, y1), x1 = Ay + By + C, y1 = Dx + Ey + F;  
Однородные координаты: (x, y, w) где w – масштабный коэффициент, в плоском случае w = 1, декартовы координаты: Y = y/w, X = x/w;   
(x1 y1 1) = (x y 1)Mпр, . Операция преобразования в общем случает не коммутативна. Преобразования, при которых плоскость не вырождается в прямую или точку, сохраняется параллельность прямых и существует обратное преобразование, называются аффинными. Если определитель матрицы преобразования отличен от нуля, то такое преобразование будет являться аффинным.  
Перенос: 2 параметра: dx, dy; , x1 = x + dx, y1 = y + dy.  
Масштабирование: 4 параметра: M(xm, ym) – центр масштабирования, kx, ky – коэффициенты масштабирования. Если kx == ky, то масштабирование однородное. x1 – xm = kx(x – xm), x1 = kx \* x + (1 – kx) \* xm, y1 = ky \* y + (1 – ky) \* ym. . 1) Перенос, совмещающий центр масштабирования с началом координат. 2) Масштабирование. 3) Обратный перенос. kx > 1, ky > 1 – удаляется. 0 < kx < 1, 0 <ky < 1 – приближается. ky = 1, kx = -1 – осевая симметрия, ky = -1, kx = 1 – отражение, kx = -1, ky = -1 – центральная симметрия.  
Поворот: 3 параметра: xc, yc – центр поворота, θ – угол поворота. **x1** = xc + Rcos(180 – (φ + θ)) = xc – Rcos(φ + θ) = xc – R\*cosφ\*cosθ + R\*sinφ\*sinθ = xc – (xc – x)cosθ + (y – yc)sinθ = xc + (x – xc)cosθ + (y – yc)sinθ; **y1** = yc + Rsin(180 – (φ + θ)) = yc + Rsin(φ + θ) = yc + R\*sinφ\*cosθ + R\*cosφ\*sinθ = yc + (y - yc)cosθ + (xc – x)sinθ = yc - (x – xc)sinθ + (y – yc)cosθ. . 1) Перенос, совмещающий центр с началом координат. 2) Поворот. 3) Обратный перенос.  
Коммутативные операции: перенос|перенос, поворот|поворот, масштабирование|масштабирование, однородное масштабирование|поворот. Аддитивные: перенос, поворот. . . Мультипликативная: масштабирование: .

3. Построение плоских кривых. Выбор шага изменения аргумента. Алгоритм построения эллипса и окружности по методу средней точки.  
Построение окружности: (X-Xo)^2 + (Y-Yo)^2 = R^2 или X=Xo+Rcost, Y=Yo+Rsint, 0<=t<=2π . Шаг изменения аргумента должен составлять величину t=1/R, рассчитываем для каждого значения параметра t значения координат соответствующих точек окружности и соединяем их затем отрезками прямых. Такой путь требует довольно большого количества вычислений. Уравнение эллипса: (X-Xc)^2/a^2 +(Y-Yc)^2/b^2 =1, параметрическое: X(t)=Xc +a\*cos(t), Y(t)= Yc +b\*sin(t), 0<=t<=2π   
Пробная функция: b^2 \* X^2 + a^2 \* Y^2 – a^2 \* b^2 = 0. На каждом шаге работы алгоритм выбирает ближайший к эллипсу пиксел из двух возможных, анализируя, находится ли средняя точка между этими пикселами внутри или вне эллипса.   
Интервал 1: Fпр.ф. = = 1) < 0 – т.М внутри, 2) =0 – т.М на дуге, 3) <0 - т.М вне эллипса. Fпр.фi = . ∆Fпр.фi = Fпр.ф. – Fпр.фi-1 =[b2(Xi-1 +1)2 + a2 (Y i-1 –0,5)2  - a2 b2 ] – [b2(Xi-1 )2 + a2 (Y i-1 –0,5)2  - a2 b2 ] = b2 (2Xi-1 +1)=2b2 Xi-1 + b2. dx=dx+bd; df=df+b2+dx, где b2=b\*b, bd=2b2. Если выбран пиксел В, то полученное значение пробной функции надо скорректировать: ∆F = [b2(Xi-1 )2 + a2 (Yi-1 –0,5)2  - a2 b2 ] – [b2(Xi-1 )2 + a2 (Yi-1 +0,5)2  - a2b2] = - 2a2 Yi-1.   
Интервал 2: Fпр.ф. = b2(Xi-1 + 0,5)2 + a2 (Y i-1 –1)2  - a2 b2 = = 1) < 0 – т.М внутри, 2) =0 – т.М на дуге, 3) <0 - т.М вне эллипса. ∆Fпр.фi=Fпр.фi – Fпр.фi-1 = [b2(Xi-1 +0,5)2 + a2 (Y i-1 – 1)2  - a2 b2 ] - [b2(Xi-1 +0,5)2 + a2 (Y i-1 )2  - a2b2 ] = - а2 (2Yi-1 -1)= -2a2 Yi-1 + a2. dy=dy-ad; df=df+a2-dy, где a2=a\*a, ad=2a2. Если выбран пиксел В, то полученное значение пробной функции надо скорректировать: ∆F = [b2(Xi-1 +0,5)2 + a2 (Yi-1)2  - a2b2] - [b2(Xi-1 –0,5)2 + a2 (Yi-1 )2  - a2b2] = 2b2i-1. Переход от первого случая выбора пикселов ко второму случаю: dY/dX= - b2X/a2Y. x, . Значение пробной функции необходимо скорректировать: ∆Fпр.фi=Fпрi – Fпрi-1 = [b2(Xi-1 +0,5)2 + a2 (Y i-1 – 1)2  - a2 b2] - [b2(Xi-1 +1)2 + a2 (Y i-1 –0,5)2  - a2 b2] = b2 (-X i-1 –0,75) + а2 (-Yi-1 +0,75) = 3(a2 - b2)/4 – (b2 Xi-1 + a2 Yi-1). Fпр = b2 + a2(b-0,5)2 - a2 b2 = b2 - a2 b + 0,25a2

4. Требования, предъявляемые к алгоритмам вычерчивания отрезков. Пошаговый алгоритм разложения отрезка в растр. Разложение в растр по методу цифрового дифференциального анализатора.  
Процесс нахождения пикселей, наилучшим образом аппроксимирующих заданный отрезок, называется разложением отрезка в растр.  
Требования: 1) Заданный отрезок должен начинаться и заканчиваться в заданных точках и должен выглядеть как отрезок прямой. 2) Яркость отрезка не должна зависеть от его длины и угла наклона. 3) Алгоритмы должны работать быстро. Первое требование в силу дискретной природы растрового дисплея выполнено всегда быть не может. Можно лишь добиться того, что визуально отрезок будет восприниматься прямым. Решение этой задачи может достигаться путем увеличения разрешающей способности экрана дисплея и применения методов устранения ступенчатости. Второму требованию удовлетворяют также только горизонтальные, вертикальные и наклоненные под углом в 45 отрезки. Однако вертикальные и горизонтальные отрезки по сравнению с отрезками, расположенными под 45, будут выглядеть ярче, так как расстояние между соседними пикселами у них меньше, чем у наклонных отрезков. Обеспечение постоянной яркости вдоль отрезка требует высвечивания очередного пиксела яркостью, зависящей от расстояния между пикселами, вычисление которого производится с использованием операций извлечения квадратного корня и умножения. Использование этих операций существенно замедляет работу алгоритма, поэтому второе требование остается, как правило, невыполненным. Удовлетворение третьего требования достигается путем сведения к минимуму вычислительных операций, использования операций над целочисленными данными, а также реализацией алгоритмов на аппаратном или микропрограммном уровне. В пошаговых методах на каждом шаге вычисляется направление элементарного движения от текущей точки к следующей. При этом удается использовать целочисленную арифметику и избежать выполнения операций умножения и деления.  
ЦДА: уравнение прямой: AX + BY + C = 0, dy/dx = const, ,Xi+1 = Xi + ∆X, Yi+1 = Yi + ∆Y. ∆Y = dy/dx \* ∆X, Yi+1 = Yi + (Yk – Yn)/(Xk – Xn) \* ∆X. ∆X = |Xk – Xn|, ∆Y = |Yk – Yn|. Алгоритм: 1) Ввод Xn, Yn, Xk, Yk. 2) Анализ отрезка на вырожденность. 3) Вычислить dx = Xk – Xn, dy = Yk – Yn. 4) Если |dx| > |dy|, то l = |dx|, иначе l = |dy|. 5)Вычислить sx = dx / l, sy = dy / l. 6) X = Xn, Y = Yn. 7) Цикл построения отрезка(по i от 1 до l + 1). 7.1) Высветить Т(E(X), E(Y)), E –округление. 7.2) x += sx, y += sy. 7.3) Конец цикла. 8) Конец. Алгоритм работает медленнее других из-за округления.



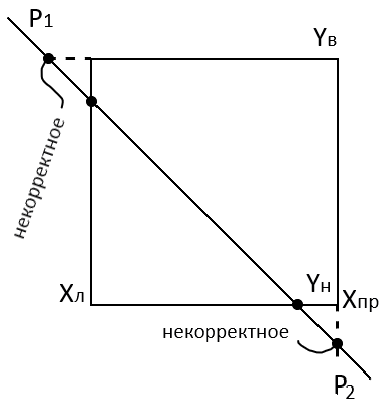
5. Алгоритмы Брезенхема разложения отрезков в растр. Простой алгоритм Брезенхема. Целочисленный алгоритм Брезенхема. Общий алгоритм Брезенхема.  
Работа алгоритма Брезенхема основывается на использовании понятия ошибка. Ошибка(e) – расстояние от действительного(идеального) отрезка до ближайшего пикселя(точки растра). Простой алгоритм Брезенхема предназначен для построения отрезков, расположенных в первом квадранте. В этом случае на каждом шаге координата X пиксела всегда получает единичное приращение, а координата Y изменяется либо на ноль, либо на единицу в зависимости от расстояния между действительным положением отрезка и ближайшей точкой растра, аппроксимирующей на данном шаге отрезок.В общем алгоритме Брезенхема большее по модулю из приращений принимается равным 1. ΔX=sign (Xк -Xн), если ⎥Хк -Xн ≥⎥Yк -Yн|, ΔY=sign(Yк -Yн), если ⎥Yк -Yн >⎥ Xк -Xн|  
ei = Yui – Yi, ei+1 = Yui+1 – Yi+1 = Yui – Yi + m = ei + m(если Yi+1 = Yi). Если же Yi+1 = Yi + 1, то, ei+1 = Yui+1 – Yi+1 = Yui – Yi + m – 1= ei + m – 1. Если e >= 1/2, то yi+1 = yi + 1, иначе yi+1 = yi. Начальное e = m – 1/2, так как нам важен знак ошибки, а не значение. Алгоритм: 1) Ввод (xn, yn), (xk, yk). 2) x = xn, y = yn. 3) dx = xk – xn, dy = yk – yn. 4) sx = sign(dx), sy = sign(dy). 5) dx = |dx|, dy = |dy|. 6) Если dx > dy, то obmen = false, иначе obmen = true, tmp = dx, dx = dy, dy = tmp. 7) m = dy / dx. 8) e = m – 1/2 // e = 2dy – dx. 9)Цикл вычерчивания отрезка(по i от 0 до dx). 9.1) Высвечивание T(x, y). 9.2) Если e >= 0 , то 1) если obmen == true, то y += sy, иначе x += sx; 2) e -= 1// e -= 2dx. 9.3) Если obmen = false, то x+= sx, иначе y += sy. 9.4) e += m // e += 2dy. 9.5) Конец цикла. 10) Конец

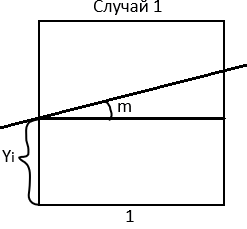
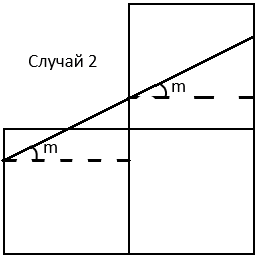
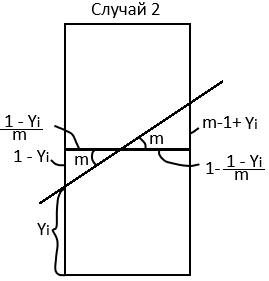
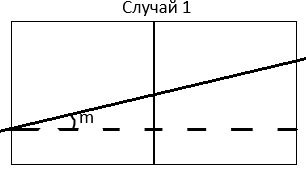
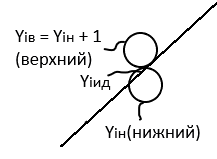
6. Основные расчетные соотношения и алгоритм Брезенхема для генерации окружности.  
Уравнение окружности: (X-Xo)2 + (Y-Yo)2 = R2. Cуществует три варианта выбора следующего пиксела: горизонтальный, диагональный, вертикальный. Расстояние от пиксела до окружности: Lг=⏐(Xi+1)2+ Yi 2- R2⏐, Lд=⏐(Xi+1)2+ (Yi-1)2 - R2⏐, Lв=⏐Xi2+ (Yi -1) 2- R2|. В окрестности текущей точки возможны 5 вариантов прохождения окружности: 1) между горизонтальным (Xi+1,Yi) и диагональным (Xi+1,Yi-1); 2)между горизонтальным (Xi+1,Yi) и другим диагональным (Xi+1,Yi+1); 3)между вертикальным (Xi,Yi-1) и диагональным (Xi+1,Yi-1) ; 4)между вертикальным (Xi,Yi-1) и третьим диагональным (Xi-1,Yi-1); 5)точно через диагональный (Xi+1,Yi-1). Di=(Xi+1)2 + (Yi-1) 2 - R2 = 1) >0 – диагональный вне окружности, 2) <0 – диагональный внутри окружности. Для случаев 1 и 2: D1=Lг-Lд=⏐(Xi+1)2 + Yi 2- R 2⏐-⏐(Xi+1) 2+ (Yi-1)2 - R 2⏐= 1) >0 - выбор диагонального, 2) <0 – выбор горизонтального, 3) =0 – выбор любого. Для случая 1 Lг>0, а Lд<0: D1=(Xi+1) 2+ Yi 2- R2+(Xi+1)2 + (Yi-1) 2- R2 = 2[(Xi+1) 2+ (Yi-1) 2- R2]+2Yi-1 = 2(Di+Yi)-1. Для случая 2 Lг<0 и Lд<0: D1=R 2- (Xi+1)2 – Yi 2+ (Xi+1)2 + (Yi-1) 2- R 2= 1 -2Yi , Yi >= 1, значит D1 < 0. Для случаев 3 и 4: D2=Lд-Lв=|(Xi+1) 2+ (Yi-1)2 - R2|-|Xi 2+ (Yi-1) 2- R2| = 1) >0 – выбор вертикального, 2) <0 – выбор диагонального, 3) = 0 – выбор любого. Для случая 3 Lд>0, а Lв<0 : D2=(Xi+1)2 + (Yi-1)2 - R2 +Xi 2+ (Yi-1)2 - R2= 2[(Xi+1)2 + (Yi-1) 2- R2-2Xi]-1=2(Di-Xi)-1. Для случая 4 Lд>0 и Lв>0: D2=(Xi+1)2 + (Yi-1)2- R2 – Xi 2- (Yi-1) 2+ R2=2Xi+1, Xi > 0, значит D2>0. Для случая 5 нужно только проверить знак D1 и D2: D1=(Xi+1) 2+ Yi2 - R 2> 0, D2=Xi 2- (Yi-1) 2- R2< 0. Горизонтальный шаг: Xi+1=Xi+1, Yi+1=Yi, Di+1=(Xi+1+1)2 + (Yi+1-1)2 - R2 =(Xi+1)2 +(Yi-1)2 - R 2+2Xi+3 = Di+2Xi+1+1.Диагональный шаг: Xi+1=Xi+1, Yi+1=Yi-1, Di+1=(Xi+2)2 + (Yi-2)2 - R2 =Di+2Xi+3-2Yi+3 = Di+2Xi+1-2Yi+1+2. Вертикальный шаг: Xi+1=Xi, Yi+1=Yi-1, Di+1=(Xi+1) 2+ (Yi-2)2 - R2 =Di-2Yi+3=Di-2Yi+1+1. Начальное значение для X=0, Y=R: D=1+(R-1)2 - R2 = 2(1-R).

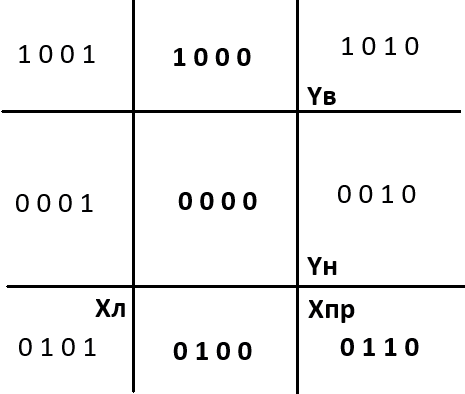
7. Растровая развертка сплошных областей. Алгоритм с упорядоченным списком ребер.  
Растровая развертка сплошных областей - генерация сплошных областей на основе простых описаний ребер или вершин. Две категории методов: растровое заполнение и затравочное заполнение. Идея: Делить сканирующую строку на области с границами в точках пересечения с ребрами многоугольника. Проблемы: 1) Каждая точка горизонтального ребра может рассматриваться как точка пересечения со сканирующей строкой. Это приведет к неправильному результату. Поэтому горизонтальные ребра игнорируются. 2) При пересечении сканирующей строки с ребрами многоугольника получим нечетное количество пересечений, и разбиение пикселов на пары даст неверный результат. Точку пересечения в вершине многоугольника следует учитывать два раза, если она является точкой локального экстремума, и учитывать один раз, если она не является точкой локального экстремума. Простой алгоритм: 1) Найти все точки пересечения сканирующих строк с ребрами многоугольника и сохранить их в массиве. 2) Упорядочить элементы массива по невозрастанию координаты Y. 3) Упорядочить все элементы в массиве с одинаковым значением координаты Y по неубыванию значения координаты x. 4) Точки пересечения, расположенные на одной сканирующей строке (имеющие одинаковое значение координаты Y) разбить на пары и закрасить все пиксели, расположенные внутри интервала, ограниченного очередной парой пикселей. Y-группы распределяющая сортировка: 1) Найти все точки пересечения и занести координаты X этих точек в соответствующие Y-группы. 2) Отсортировать точки пересечения, расположенные в пределах каждой Y-группы по неубыванию. 3) Разбить содержимое каждой Y-группы на пары. 4) Закрасить пиксели, расположенные внутри каждого интервала, ограниченного очередной парой точек пересечения. Алгоритм со списком активных ребер: Ребра в списке должны быть упорядочены по невозрастанию наибольшего значения координаты Y ребра. Y > Nнаив –ребро не активно, Y <= Nнаив – ребро активно. ∆X = (XB – XA) / | YB – YA|, ∆Y = YB – YA. 1) Сформировать список с информацией о ребрах. 2) Просматривать элементы списка, пока выполняется условие y <= Nнаив. Копировать абсциссы точек пересечения в другой список. Упорядочить полученный список по возрастанию. Элементы упорядоченного списка разбить на пары. Закрасить пиксели, расположенные внутри интервала, ограниченного парой пикселей. При просмотре элементов списка корректировать их содержимое: XA = XA + ∆X=, ∆Y = ∆Y – 1. Проверять ∆Y<0: при выполнении условия удалить элемент из списка. Изменять цвет пикселя нужно 1 раз, информация о цвете не получается, пиксели рассматриваются только внутри области.

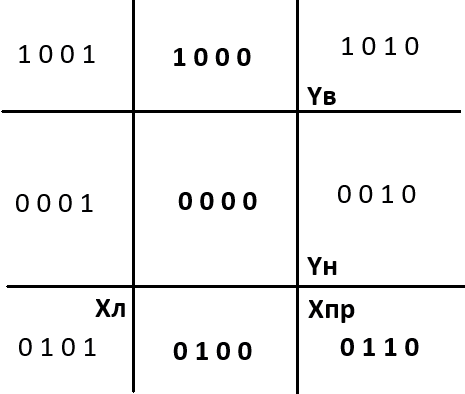
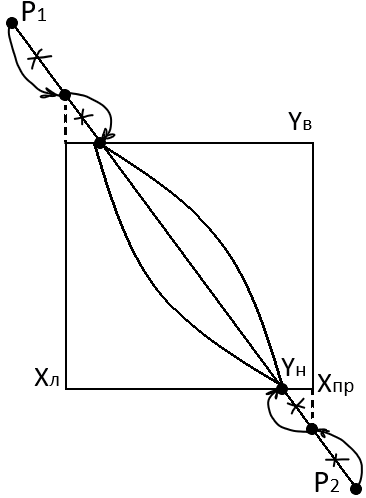
8. Заполнение многоугольников. Алгоритмы заполнения по ребрам, с перегородкой, со списком ребер и флагом.  
Растровая развертка сплошных областей - генерация сплошных областей на основе простых описаний ребер или вершин. Две категории методов: растровое заполнение и затравочное заполнение. Алгоритм заполнения по ребрам: Дополнить все пиксели сканирующей строки, расположенные правее точки пересечения ребра многоугольника с этой сканирующей строкой. Ребра можно обрабатывать в произвольном порядке. Каждый пиксель может изменять цвет многократно, количество ребер слева от пикселя равно количеству изменений цвета. Информация о цвете получается при каждом изменении цвета. Обрабатываются все пиксели от точки пересечения до правой максимальной точки многоугольника. Алгоритм заполнения с перегородкой: Ребра можно обрабатывать в произвольном порядке. Для каждой сканирующей строки, пересекающей ребро многоугольника, дополнить пиксели, расположенные правее точки пересечения, но левее перегородки, если пересечение расположено левее перегородки. Дополнить пиксели, расположенные левее точки пересечения, но правее перегородки, если пересечение расположено правее перегородки. Строки проходящие через вершины обрабатывать особым образом. Алгоритм со списком ребер и флагом: Пересечение сканирующей строки с ребрами многоугольника находится на основе анализа цвета пикселя. 1) Ввод исходных данных: цвет границы, цвет заполнения, кол-во вершин, координаты вершин многоугольника. 2) Высвечивание границ многоугольника. 3) Цикл по всем сканирующим строкам(по y от ymin до ymax). 3.1) flag = false. 3.2) Цикл анализа пикселей текущей строки(по x от xmin до xmax). 3.2.1) Если цвет(x,y) == цвет границ, то flag = -flag. 3.2.2) Если flag == true, то цвет(x,y) = цвет закраски, иначе цвет(x,y) = цвет фона. 3.3) Конец цикла по х. 4) Конец цикла по y.Каждый пиксель меняет цвет 1 раз. Информация о цвете получается 1 раз.

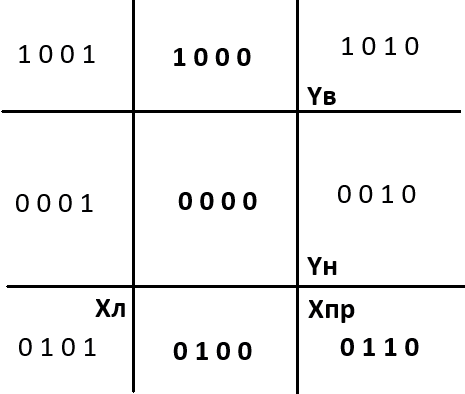
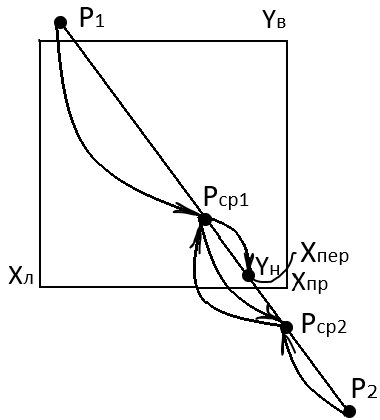
9. Алгоритм заполнения с затравкой, простой алгоритм заполнения с затравкой.  
Затравочный пиксель – точка внутри закрашиваемой области. Области бывают гранично определенными (область, все внутренние пикселы которой имеют один и тот же цвет) и внутренне определенными (область, граничные пикселы которой имеют вполне определенный цвет.). Области бывают 4-связными(4 направления движения) и 8-связными(8 направлений движения). Используем гранично-определённые 4-связные области. Алгоритм: 1) Ввод исходных данных: цвет границы, цвет заполнения, координаты затравки(x,y), информация о границе. 2) Очерчивание границ области. 3) Занесение затравочного пикселя в стек. 4) Пока стек не пуст, выполнять следующие действия. 4.1) Извлечь затравочный пиксель из стека. 4.2) Закрасить извлеченный пиксель: цвет(x,y) = цвет закраски(если цвет(x,y) != цвет закраски). 4.3) Анализ цвета 4 соседних пикселей. Анализ цвета: Если цвет(x,y) != цвет закраски и цвет(x,y) != цвет границ, то поместить пиксель(x,y) в стек. Минусы алгоритма: Некоторые пиксели могут помещаться в стек несколько раз. Большой расход памяти, т.к. каждый затравочный пиксель помещается в стек. Плюс: Алгоритм прост.

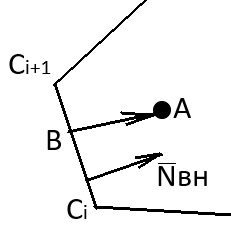
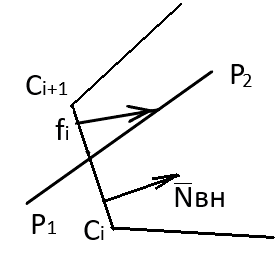
10. Алгоритмы заполнения с затравкой. Построчный алгоритм заполнения с затравкой.   
Затравочный пиксель – точка внутри закрашиваемой области. Области бывают гранично определенными (область, все внутренние пикселы которой имеют один и тот же цвет) и внутренне определенными (область, граничные пикселы которой имеют вполне определенный цвет.). Области бывают 4-связными (4 направления движения) и 8-связными (8 направлений движения). Используем гранично-определённые 4-связные области. Повышение эффективности по сравнению с простым алгоритмом связано с уменьшением объёма расходуемой памяти. В стек помещается 1 пиксель для непрерывного интервала пикселей. Непрерывный интервал – группа примыкающих друг к другу не закрашенных и не граничных пикселей, ограниченных закрашенными или граничными пикселями в пределах 1 строки. 2 этапа: 1) закраска пикселей, расположенных в 1 строке с затравочным. 2) Поиск новых затравочных пикселей в 2 смежных строках. Алгоритм: 1) Ввод исходных данных: информация о внешней и внутренней границах, цвет границы, цвет заполнения, затравочный пиксель(x, y). 2) Очерчивание границ области. 3) Занесение затравочного пикселя в стек. 4) Пока стек не пуст, выполнить. 4.1) Извлечение затравочного пикселя из стека. 4.2) Закрасить затравочный пиксель и пиксели текущей строки, двигаясь вправо и влево от затравочного до встречи с граничными пикселями. 4.3) Запомнить левую и правую границы интервала(Xл и Xпр). 4.4) Поиск новых затравочных пикселей в интервале Xл<= x <= Xпр на двух соседних строках (y +1, y – 1) ( в качестве затравочного в стек помещается самый правый еще не закрашенный и не граничный пиксель из интервала таких пикселей). 4.5) Занесение в стек найденных затравочных пикселей.

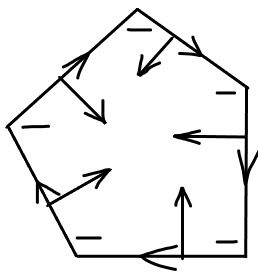
11. Основы методов устранения ступенчатости. Алгоритм Брезенхема с устранением ступенчатости. Алгоритм Ву. Принципиально устранить ступенчатость (лестничный эффект)невозможно. Однако применением специальных методов можно добиться того, что визуально ступеньки будут слабо заметны или практически незаметны. Способ Брезенхэма устранения ступенчатости рассматривает пиксел не как математическую точку, а как некоторую конечную область. Интенсивность пиксела устанавливается пропорционально площади части пиксела, находящейся под отрезком. Случай 1: Si = Yi \* 1 + 1 \* m / 2 = Yi + m/2. Случай 2: Sниж. = 1 – (1 – Yi)2 / 2m. Sверх = (m – 1 + Yi)2 / 2m. Sобщее = Sниж + Sверх = = 1 – (1 – Yi)2 / 2m + (m – 1 + Yi)2 / 2m = Yi + m/2. Вычисление площади для (i + 1) шага. Случай 1:Si+1 = Si + Sпряи = Si + m. Случай 2; Si+1 = Si + Sпряи - 1= Si + m - 1 Поскольку доля площади не может быть отрицательной величиной, то необходимо скорректировать величину ошибки, прибавив к ней величину W=1-m. S = S + W = 1/2. Iнач = Imax / 2. В качестве ошибки Брезенхэм предложил использовать площадь. I ~ e. e = Imax / 2. m = Imax m. W = Imax – m.   
Алгоритм Ву: ∆Yiн = Yiид – Yi, ∆Yiв = Yi+1 - Yiид, ∆Yiв = 1 - Yiид + ]Yiид[, ∆Yiн = Yiид  - ]Yiид[, Iiн ~ 1 - Yiид + ]Yiид[, Iiв ~ Yiид -]Yiид[, Iiн + Iiв = I.

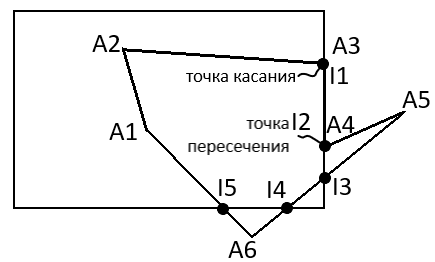
12. Двумерное отсечение. Простой алгоритм отсечения отрезка.  
Отсечение - это операция удаления части изображения за пределами выделенной области, называемой отсекателем. Отсечение бывает двумерное и трехмерное. Отсекатели бывают регулярные (прямоугольник со сторонами параллельными координатным осям) и нерегулярные (выпуклые и невыпуклые многоугольники). Видимый объект – объект, целиком располагающийся внутри отсекателя. Невидимый объект – объект, целиком располагающийся вне отсекателя. Частично видимый объект – объект, часть которого расположена внутри отсекателя, часть – вне отсекателя. S = = 1) =0 - точка видима, 2) != 0 – точка невидима. P = = 1) != 0 – тривиально невидим, 2) = 0 – видим, частично видим, нетривиально невидим. Пересечения: Yл= m(Xл-X1)+Y1 , m≠∞; Yпр= m(Xп-X1)+Y1 , m≠∞; Xн=(1/m)(Yн-Y1)+X1 , m≠0; Xв=(1/m)(Yв-Y1)+X1 , m≠0. Алгоритм: 1) Ввод данных: Xл, Xпр, Yн, Yв, P1, P2. 2) Инициализация признака видимости Fl = 1; 3) m = 1020(отрезок вертикален). 4) Вычисление кодов концов отрезка T1, T2 и их сумм S1, S2. 5) Проверка полной видимости отрезка: если S1 = 0 и S2 = 0, то R1 = P1, R2 = P2, переход к пункту 20. 6) Проверка полной невидимости отрезка: вычисление P = , если P != 0, то Fl = -1, переход к пункту 20. 7) Проверка видимости первой вершины: если S1 = 0, то R1 = P1, Pтек = P2, i = 2, переход к пункту 13. 8) Проверка видимости второй вершины: если S2 = 0, то R1 = P2, Pтек = P1, i = 2, переход к пункту 13. 9) Инициализация счетчика числа пересечений i = 0. 10) i = i + 1. 11) Если i > 2, переход к пункту 19. 12) Pтек = Pi. 13) Если P2.x – P1.x = 0, переход к пункту 17. 13) Вычисление тангенса угла наклона m = (P2.Y-P1.Y)/(P2.X-P1.X). 14)Если Pтек.X < Xл, то Yпер = m(Xл – Pтек.X) + Pтек.Y, проверка корректности пересечения: если Yн <= Yпер <= Yв, то Ri.X = Xл, Ri.Y = Yпер, переход к пункту 10. 15)Если Pтек.X > Xпр, то Yпер = m(Xпр – Pтек.X) + Pтек.Y, проверка корректности пересечения: если Yн <= Yпер <= Yв, то Ri.X = Xпр, Ri.Y = Yпер, переход к пункту 10. 17)Если Pтек.Y < Yн, то Xпер = (Yн – Pтек.Y)/m + Pтек.X, проверка корректности пересечения: если Xл <= Xпер <= Xпр, то Ri.Y = Yн, Ri.X = Xпер, переход к пункту 10. 18)Если Pтек.Y > Yв, то Xпер = (Yв – Pтек.Y)/m + Pтек.X, проверка корректности пересечения: если Xл <= Xпер <= Xпр, то Ri.Y = Yв, Ri.X = Xпер, переход к пункту 10. 19) Fl = -1(отрезок невидимый). 20) Если Fl =1, то визуализация отрезка R1R2.20) Конец

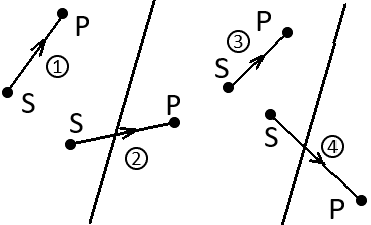
13. Отсечение. Алгоритм Сазерленда-Коэна отсечения отрезка.   
Отсечение - это операция удаления части изображения за пределами выделенной области, называемой отсекателем. Отсечение бывает двумерное и трехмерное. Отсекатели бывают регулярные (прямоугольник со сторонами параллельными координатным осям) и нерегулярные (выпуклые и невыпуклые многоугольники). Видимый объект – объект, целиком располагающийся внутри отсекателя. Невидимый объект – объект, целиком располагающийся вне отсекателя. Частично видимый объект – объект, часть которого расположена внутри отсекателя, часть – вне отсекателя. S = = 1) =0 - точка видима, 2) != 0 – точка невидима. P = = 1) != 0 – тривиально невидим, 2) = 0 – видим, частично видим, нетривиально невидим. Пересечения: Yл= m(Xл-X1)+Y1 , m≠∞; Yпр= m(Xп-X1)+Y1 , m≠∞; Xн=(1/m)(Yн-Y1)+X1 , m≠0; Xв=(1/m)(Yв-Y1)+X1 , m≠0. Алгоритм: 1) Ввод массива О(Xл, Xпр, Yн, Yв), P1, P1. 2) Fl = -1. 3) Если P2.X – P1.X != 0, то m = (P2.Y – P1.Y) / (P2.X – P1.X), если m = 0, то Fl = 1, иначе Fl = 0. 4) Цикл отсечения отрезка по всем границам отсекателя (по i от 1 до 4). 4.1) Определения признака видимости wid, кодов концов отрезка T1 и T2 с помощью подпрограммы(O, P1, P2)// на основе T1, T=, S1, S2, P. 4.2) Если wid = -1, то переход к пункту 6, иначе если wid = 1, то переход к пункту 5. 4.3) Если T1i = T2i, то переход на следующий шаг цикла. 4.4) Если T1i = 0, то R = P1, P1 = P2, P2 = R. 4.5) Если Fl = -1, то переход к п 4.8. 4.6) Если i <= 2, то P1.Y = m(Oi – P1.X) + P1.Y, P1.X = Oi, переход к пункту 4.9. 4.7) P1.X = (Oi – P1.Y)/m + P1.X. 4.8) P1.Y = Oi. 4.9) Конец цикла. 5) Визуализация отрезка P1P2. 6) Конец

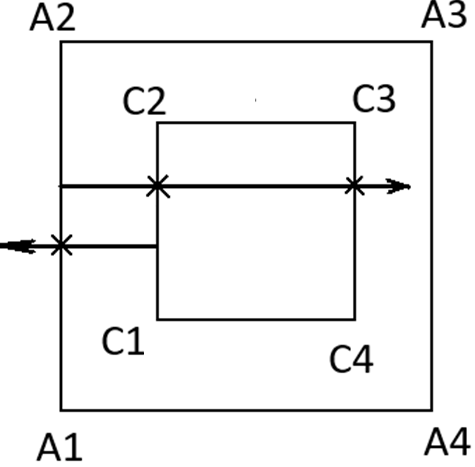
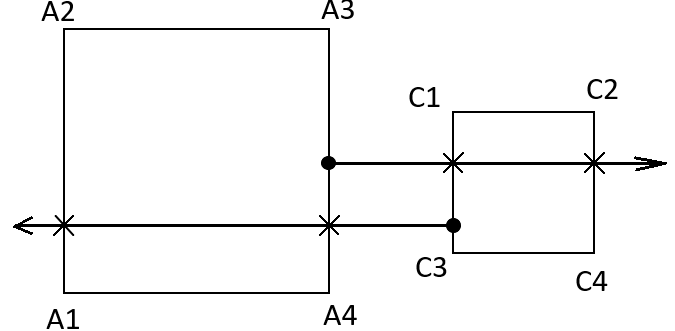
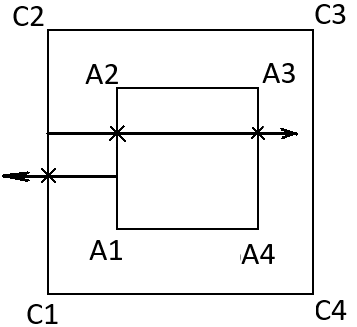
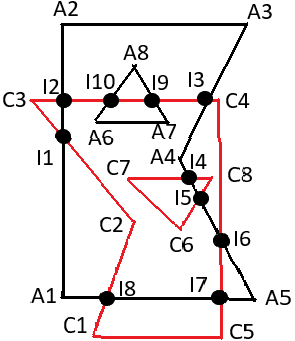
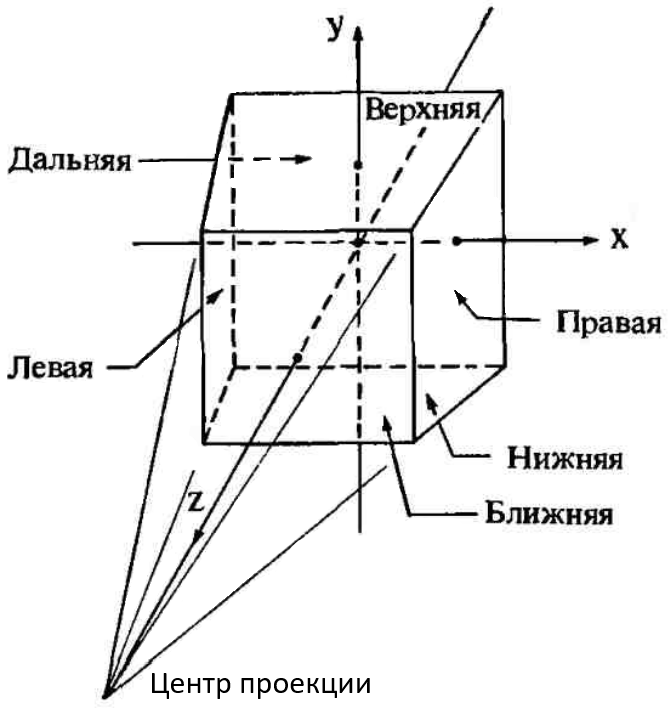
14. Отсечение Алгоритм разбиения средней точкой при отсечении отрезка.  
Отсечение - это операция удаления части изображения за пределами выделенной области, называемой отсекателем. Отсечение бывает двумерное и трехмерное. Отсекатели бывают регулярные (прямоугольник со сторонами параллельными координатным осям) и нерегулярные (выпуклые и невыпуклые многоугольники). Видимый объект – объект, целиком располагающийся внутри отсекателя. Невидимый объект – объект, целиком располагающийся вне отсекателя. Частично видимый объект – объект, часть которого расположена внутри отсекателя, часть – вне отсекателя. S = = 1) =0 - точка видима, 2) != 0 – точка невидима. P = = 1) != 0 – тривиально невидим, 2) = 0 – видим, частично видим, нетривиально невидим. Pср = (P1 + P2) /2. |P1 – P2| <= ε. Подход оправдан т.к. происходит деление на 2(сдвиг) на каждой итерации. Задачу нахождения пересечения отрезка с границей отсекателя можно рассматривать как задачу поиска наиболее удаленной, но еще видимой точки. Алгоритм: 1) Ввод данных: P1, P2, Xл, Xпр, Yн, Yв, ε. 2)i = 1. 3) Вычисление кодов концов отрезков, их сумм S1, S2. 4) Проверка полной видимости: если S1 = 0 и S2 = 0, то R1 = P1, R2 = P2, переход к пункту 17. 5)Выисление произведений кодов концов отрезка, проверка полной невидимости: если P != 0, то переход к пункту 18. 6) T = P1. 7) Если i > 2, то вычислить произведение кодов концов, если P != 0, то переход к пункту 18, иначе переход к пункту 17. 8) Если S2 = 0, то Ri = P2, P1 = P2, P2 = T, i = i + 1, переход к пункту 3. 9) Если |P1 – P2| <= ε, то Ri = P1, P1 = P2, P2 = T, i = i + 1, переход к пункту 3. 10) Pср = (P1 + P2) / 2. 11) Rтек = P1. 12) P1 = Pср. 13) Вычисление кодов для вершины P1. 14) Вычисление логического произведения кодов концов отрезка. 14) Если P != 0, то P1 = Rтек, P2 = Pср. 16) Переход к пункту 9. 17) Визуализация отрезка R1, R2. 18)Конец

15. Отсечение. Алгоритм Кируса-Бека отсечения отрезка.  
Отсечение - это операция удаления части изображения за пределами выделенной области, называемой отсекателем. Отсечение бывает двумерное и трехмерное. Отсекатели бывают регулярные (прямоугольник со сторонами параллельными координатным осям) и нерегулярные (выпуклые и невыпуклые многоугольники). Видимый объект – объект, целиком располагающийся внутри отсекателя. Невидимый объект – объект, целиком располагающийся вне отсекателя. Частично видимый объект – объект, часть которого расположена внутри отсекателя, часть – вне отсекателя. Параметрическое уравнение отрезка P1P2 имеет вид:P(t)= P1 + (P2- P1)t; 0≤ t ≤1, где t - параметр. внi \* = 1) >0 – т.А видима, 2) <0 – т.А невидима, 3) =0 – т.А лежит на границе отсекателя. внi \* = 1) <0 – точка отрезка расположена по невидимую сторону от i-ой границы, 2) =0 – точка отрезка расположена на границе отсекателя, она является точкой пересечения, 3) >0 – точка расположена по невидимую сторону от границы отсекателя. =– директриса отрезка, i = , внi \*  i + внi\*\*t = 0, t = - (внi \* i) / (внi \* ). Случай 1: отрезок параллелен стороне отсекателя. Проверяем знак внi \* i, определяем видимость отрезка относительно текущей стороны. Случай 2: D = 0, отрезок вырожденный, с помощью внi \* i проверяем видимость точки. Wскi = внi \* i, Dскi = внi \* , ti = - Wскi / Dскi. внi \* = 1) >0 – точка расположена ближе у началу отрезка, 2) <0 – точка расположена ближе к концу отрезка, 3) – точка расположена в середине отрезка. tн = max(ti), i = – номера точек, определяющих нижнюю границу. tв = min(tj), j = – номера точек, определяющих верхнюю границу. Полная невидимость отрезка: Случай 1: 1) t > 1. 2) Dск > 0 – точка определяет начало видимой части. 3) tн = max(tн, t) – отрезок полностью невидимый. Случай 2: 1) t <0. 2) Dск <0 – точка определяет конец видимой части. 3) tв = min(tв, t) – отрезок полностью невидимый. Случай 3: В конце работы алгоритма tн > tв. Алгоритм: 1) Ввод данных: P1, P2, N, C(N). 2) Проверка отсекателя на выпуклость. 3) tн = 0, tв = 1, = . 4) Цикл отсечения отрезка по всем границам отсекателя (по i от 1 до N). 4.1) Нахождение вектора внутренне нормали вн. 4.2)Нахождение вектора , = .(В качестве fi удобно взять одну из вершин текущего ребра отсекателя). 4.3) Вычисление скалярных произведений Dск = вн \* , Wск = вн \* . 4.4) Если Dск = 0, то если Wск>= 0, то переход на следующий шаг выполнения цикла, иначе переход на пункт 6. 4.5) t = - Wск / Dск. 4.6) Если Dск> 0, то если t> 1, то переход к пункту 6, иначе tн = max(tн, t), иначе если t < 0, то переход к пункту 6, иначе tв = min(tв, t). 4.7) Конец цикла. 5) Если tн <= tв, то изображение отрезка P(tн)P(tв). 6) Конец

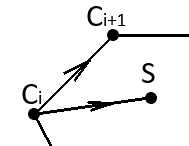
16. Внутреннее и внешнее отсечение. Определение выпуклости многоугольника; определение нормали; разбиение невыпуклых многоугольников. Триангуляция многоугольников.  
Отсечение(внутреннее) - это операция удаления части изображения за пределами выделенной области, называемой отсекателем. Стирание(внешнее) - это операция удаления части изображения в пределах выделенной области. Отсечение бывает двумерное и трехмерное. Отсекатели бывают регулярные (прямоугольник со сторонами параллельными координатным осям) и нерегулярные (выпуклые и невыпуклые многоугольники). Видимый объект – объект, целиком располагающийся внутри отсекателя. Невидимый объект – объект, целиком располагающийся вне отсекателя. Частично видимый объект – объект, часть которого расположена внутри отсекателя, часть – вне отсекателя. Определение выпуклости многоугольника: 1) Рассмотрим стороны как векторы. 2) Вычислим векторное произведение векторов, смежных для вершины. 3) Если знаки всех векторных произведений 0, то вырожденный. Если неположительны, то выпуклый, а внутренние нормали ориентированы вправо от направления обхода. Если неотрицательны, то выпуклый и нормали влево от обхода. Иначе – невыпуклый. Алгоритм разбиения: алгоритм трудоемкий, так как основан на выполнении преобразований переноса и поворота. 1) Перенос i-ой вершины в начало координат. 2) Поворот так, чтобы i+1-ая вершина была в положительной стороне Х. 3) Если знаки ординат у всех (i+2)-ых вершин неотрицательные, то многоугольник выпуклый относительно очередной стороны. Иначе многоугольник невыпуклый, ищется ближайшая к началу координат точка пересечения с осью Х, абсцисса которой больше значения xi+1. 1 многоугольник образован вершинами с (i+1)-ой и до точки пересечения, 2 многоугольник – оставшиеся точки и точка пересечения. Нахождение нормалей: ax \* nx + ay \* ny = 0, nx = - ay \* ny / ax, ny = 1. Если ax = 0, то nx = 1, ny = 0, если ay = 0, то nx = 0, ny = 1. i \* = 1) >0 – нормаль внутрення, 2) <0 – нормаль внешняя, N нужно умножить на -1.(j != 1)



17. Отсечение многоугольников. Алгоритм Сазерленда-Ходжмена.   
Многоугольник можно рассматривать как совокупность отрезков и попытаться применить ранее рассмотренные алгоритмы отсечения отрезков для отсечения ребер многоугольника. Однако это может привести к тому, что первоначально замкнутая фигура (многоугольник) превратится в незамкнутую. Случаи расположения сторон многоугольника: 1) Невидимое ребро, в результат ничего не заносить. 2) Ребро входит внутрь отсекателя(частично видимое), найти точку пересечения ребра с ганицей отсекателя, в результат занеснти точки пересечения и конечную вершину. 3)Ребро полностью видимо, в результат занести конечную точку (начальная уже была занесена как конечная на предыдущем этапе). 4)Ребро выходит из отсекателя наружу(частично видимое), найти точку пересечения и занести ее в результат (начальная точка уже была занесена). Определение видимости: 1)Использование скалярного произведения вектора внутренней нормали и вектора от точки на ребре отсекателя до данной точки. 2) Использование пробной функции (ур-ие прямой Fпр.ф. = ax + by +c). Сравниваем значение пробной функции в данной точке и в точке, положение которой известно. 3) Использование скалярного произведения × = 1) >0 – т.S видима, 2) <0 – т.S невидима, 3) =0 – т.S лежит на ребре отсекателя. Нахождение точки пересечения ребра многоугольника с прямой, проходящей через ребро отсекателя: P(t) = P1 + (P2- P1)t, 0<=t<=1. Q(s) = Q1 + (Q2 – Q1)s. P1.X + (P2.X – P1.X)\*t = Q1.X + (Q2.X – Q1.X)\*s, P1.Y + (P2.Y – P1.Y)\*t = Q1.Y + (Q2.Y – Q1.Y)\*s. Алгоритм: 1) Ввод данных: NA, A(NA), NC, C(NC) (продублировать точку С1 и записать ее в конец) 2) Проверить на выпкулость. 3) Отсечение многоугольника по всех границам отсекателя(по i от 1 до NC-1). 3.1) NB = 0, B = 0. 3.2) Цикл по всем вершинам(ребрам) отсекаемого многоугольника(по j от 1 до NA). 3.2.1) Если j = 1, то F = Aj, переход к пункту 3.2.4. 3.2.2) Определение наличия пересечения реьра SAj с прямой CiCi+1. 3.2.3) Если наличие пересечения установлено, то нахождение точки пересечения ребра SAj и прямой CiCi+1. Занесение т.I в результат: NB = NB + +1, B(NB) = I. 3.2.4) S = Aj. 3.2.5) Определение видимости т.S относительно приой CiCi+1.. 3.2.6) Если т.S видима, то занесение ее в результат: NB = NB + 1, B(NB) = S. 3.2.7) Конец цикла по j. 3.3) Если NB = 0, то переход к пункту 5. 3.4) Определение факта пересечения ребра SF с прямой CiCi+1. 3.5) Если наличие пересечения установлено, то нахождение точки пересечения I и занесение ее в результат: NB = NB +1, B(NB) = I. 3.6)NA = NB; A = B.(копирование промежуточного полученного результата в исходный массив). 3.7) Конец цикла по i. 4) Вузализация многоугольника А. 5)Конец. Недостаток алгоритма – в определенных ситуациях появляются ложне ребра, когда в результате отсечения появляется 2 или более многоугольников(ложные ребра соединяют эти многоугольники)

18. Отсечение многоугольников невыпуклыми областями. Алгоритм Вейлера-Азертона.  
Для реализации отсечения следует описать каждую границу отсекаемого многоугольника и отсекателя в виде циклического списка их вершин. При этом внешние границы многоугольников должны обходиться по часовой стрелке, а внутренние - против часовой. Таким образом внутренняя область будет расположена справа от границы. Точки пересечения нужно разделить на точки входа и точки выхода. Точка входа – точка, в которой ребро отсекаемого многоугольника входит внутрь отсекателя, точка выхода – выходит наружу. Определение типа точки: × = 1)>0 – точка входа, 2) <0 – точка выхода. Количество точек пересечения должно быть четным. Точки пересечения целесообразно соединять двусторонними связями, иначе нужно будет осуществлять одноименные точки пересечения в другом многоугольнике. При поиске внутренних многоугольников все списки просматриваются от начала к его концу. Поиск очередного многоугольника начинается с очередной точки входа. Нужно найти ее в списке границ отсекаемого многоугольника и начать просмотр списка с этой точки пока не встретим точку пересечения, затем нужно найти ее в другом списке и начать просматривать его. Эти действия повторять пока не встретим исходную точку пересечения. Поиск внешних многоугольников осуществляется просмотром списка отсекателя от конца к началу, начиная от очередной точки выхода в списке отсекаемого многоугольника. Пример: Внешние многоугольники: A1A2A3A4A5A1, C1C2C3C4C5C1. Внутренние многоугольники: A6A7A8A6, С6C7C8C6. Точки пересечения: I1,I2,I3,I4,I5,I6,I7,I8,I9,I10. Получившиеся многоугольники отсекаемого многоугольника: A1I1I2A2A3I3A4I4I5I6A5I7I8A1, A6A7I9A8I10A6. Получившиеся многоугольнки отсекателя: C1I8C2I1C3I2I10I9I3C4I6I7C5C1, C6I5C7I4C8C6. Точки входа:I1,I3,I5,I7,I10. Точки выхода: I2,I4,I6,I8,I9. Внутренние многоугольнки: I1I2I10A6A7I9I3A4I4C8C6I5I6I7I8C2I1. Внешние многоугольники: 1) I2A2A3I3I9A8I10I2, 2) I4I5C6C8I4, 3) I6A5I7I6, 4) I8A1I1C2I8. Случаи без пересечений: 1) Внешние многоугольники, 2) Отсекатель внтри отсекаемого, 3) Отсекаемый внутри отсекателя. Надо четко различать пересечение ребер от их касания.

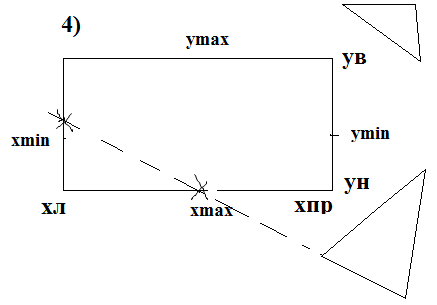
19. Модели трехмерных объектов. Требования, предъявляемые к моделям.  
Виды моделей: 1) Каркасная(проволочная), 2) Поверхностная, 3)Объёмная. Требования: 1) Модель не должны противоречить объекту, она должна правильно его отображать. 2) Модель должна допускать моделирования тела целиком. 3) Модель должна позволять выполнять математические и геометрические расчеты. 4) Модель должна позволять четко сказать, принадлежит ли точка телу или нет. 5) Модель должны быть мощной, открытой, позволять создавать подобные модели на своей основе и допускать редактирование. 6) Модель не должна деформироваться при преобразованиях.

20. Операции преобразования в трехмерном пространстве. Матрицы преобразований.  
Координаты точки (X, Y, Z) заменяются четверткой (WX, WY, WZ,W), W!=0. Каждая точка пространства может быть задана четверкой одновременно не равных нулю чисел, эта четверка определена однозначно до общего множителя. Такой подход позволяет в матричной форме описать операции преобразования. Любое аффинное преобразование в трехмерном пространстве может быть представлено в виде суперпозиции переносов, масштабирований, поворотов.Перенос: матрица преобразования: Mпр = , X1=X+dX; Y1=Y+dY; Z1=Z+dZ. Задавая одни и те же значения dX, dY, dZ для всех точек рисунка и повторяя многократно перенос, получим движение рисунка по экрану без изменения формы изображаемого объекта. Если же значения dX, dY, dZ не совпадают для всех точек, то движение будет происходить с изменением начальной формы. При движении объекта вдоль оси Z необходимо использовать при его изображении уравнение перспективы, поэтому при отрицательных значениях DZ объект, приближаясь к наблюдателю, увеличивается в размерах, при положительных значениях, удаляясь, уменьшается в размерах. Масштабирование: Мпр = , X1=X\*KX+(1-KX)\*Xм; Y1=Y\*KY+(1-KY)\*Yм; Z1= Z\*KZ+(1-KZ)\*Zм.1) Поворот совмещающий центр масштабирования с началом координат. 2) Масштабирование. 3) Обратный перенос. При KX= -1, KY=1, KZ=1 происходит отображение относительно плоскости YOZ; при KX=1, KY= -1, KZ=1 происходит отображение относительно плоскости XOZ; при KX=1, KY=1, KZ= -1 происходит отображение относительно плоскости XOY. Поворот: наблюдатель смотрит с конца положительной полуоси. Мпрz = , Мпрx = , Мпрy = , вокруг Z: X1=Xc+(X-Xc)\*cosθ-(Y-Yc)\*sinθ; Y1=Yc+(X-Xc)\*sinθ+(Y-Yc)\*cosθ, вокруг X: Y1=Yc+(Y-Yc)\*cosθ-(Z-Zc)\*sinθ; Z1=Zc+(Z-Zc)\*cosθ+(Y-Yc)\*sinθ , вокруг Y: X1=Xc+(X-Xc)\*cosθ+(Z-Zc)\*sinθ, Z1=Zc+(Z-Zc)\* cosθ -(X-Xc)\*sinθ. 1) Перенос, совмещающий центр поворота с началом координат. 2) Поворот. 3) Обратный перенос.

21.Трехмерное отсечение. Виды отсекателей. Вычисление кодов концов отрезка для каждого типа отсекателей. Алгоритм отсечения отрезков средней точкой.  
Стандартным отсекателем в трехмерном пространстве является прямоугольный параллелепипед с гранями, параллельными координатным плоскостям. Используется шестибитовый код для видимости: 1 бит – левая грань, 2 бит – правая грань, 3 бит – нижняя грань, 4 бит – верхняя грань, 5 бит – передняя грань, 6 бит – дальняя грань. В качестве отсекателя используется также усеченная пирамида. Объекты, попадающие в поле видимости наблюдателя, располагаются в пределах конуса, вершина которого совпадает с глазом человека. Наблюдатель не видит предметов, расположенных на очень близком расстоянии и на большом расстоянии, получается конус видимости. Удобнее пользоваться усеченной пирамидой (пирамидой видимости), которая аппроксимирует конус. Уравнение прямой на плоскости XZ, несущей проекцию правой грани отсекателя: X = (Z – ZЦП) \* Xпр / (ZД – ZЦП) = Z\*A + B, A = Xпр / (ZД – ZЦП), B = -A\*ZЦП. Пробная функция для правой грани: Fпр = X – Z\*A – B = 1)>0 – точка справа от плоскости, 2) <0 – точка слева от плоскости, 3) =0 – точка на плоскости. Пробная функция для левой грани: Fпр = X - Z\*C – D = 1)>0 – точка справа от плоскости, 2) <0 – точка слева от плоскости, 3) =0 – точка на плоскости, C = XЛ / (ZД – ZЦП), D = -C \* ZЦП. Пробная функция для верхней грани: Fпр = Y – Z\*E – G = 1)>0 – точка выше плоскости, 2) <0 – точка ниже плоскости, 3) =0 – точка на плоскости, E = YВ/(ZД – ZЦП), G = -E \* ZЦП. Пробная функция для нижней грани: Y – Z\*H – K = 1)>0 – точка выше плоскости, 2) <0 – точка ниже плоскости, 3) =0 – точка на плоскости, H = YН/(ZД – ZЦП), K = - H \* ZЦП. Пробная функция для ближней грани: Fпр = Z – ZБ = 1)>0 – точка ближе плоскости, 2) <0 – точка дальше плоскости, 3) =0 точка на плоскости. Пробная функция для дальней грани: Fпр = Z – ZД = 1)>0 – точка ближе плоскости, 2) <0 – точка дальше плоскости, 3) =0 точка на плоскости. При вычислении кодов концевых точек отрезков можно получить некорректный результат, если точки лежат за центром проекции. Это связано с тем, что левая, правая, нижняя, верхняя грани пересекаются в точке центра проекции. Поэтому существуют точки, лежащие одновременно левее левой и правее правой граней, выше верхней и ниже нижней граней. Если Z> ZП, нужно обратить значения первых 4 битов. Изменение алгоритма средней точки: вместо 4-битных кодов используются 6-битные, вычисление кодов концов отрезка ведется в соответствии с приведенными формулами, а при вычислении координат точек следует учитывать также третью координату.

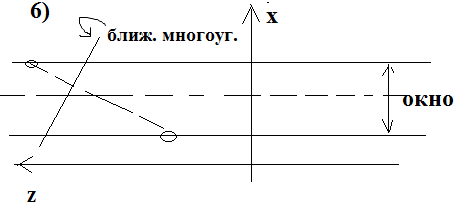
22. Отсечение отрезков в трехмерном пространстве. Трехмерный алгоритм Кируса-Бека.  
Стандартным отсекателем в трехмерном пространстве является прямоугольный параллелепипед с гранями, параллельными координатным плоскостям. В качестве отсекателя используется также усеченная пирамида. Объекты, попадающие в поле видимости наблюдателя, располагаются в пределах конуса, вершина которого совпадает с глазом человека. Наблюдатель не видит предметов, расположенных на очень близком расстоянии и на большом расстоянии, получается конус видимости. Удобнее пользоваться усеченной пирамидой (пирамидой видимости), которая аппроксимирует конус  
В алгоритме Кируса-Бека отсекатель в трехмерном случае может представлять собой произвольный выпуклый объем. Координаты точек и векторов должны иметь три компоненты в соответствии с количеством измерений. В качестве точек, лежащих на грани отсекателя, удобно выбирать точки, лежащие на концах главной диагонали. Для отсекателя - параллелепипеда легко определяются и внутренние нормали. В случае усеченной пирамиды в качестве пробных точек удобно выбирать вершины пирамиды, в которых сходятся три грани пирамиды. Определение нормалей к ближней и дальней граням очевидно, а определение нормалей к остальным четырем граням возможно на основании вычисления векторного произведения векторов, построенных на ребрах граней, начинающихся в одной из вершин грани.

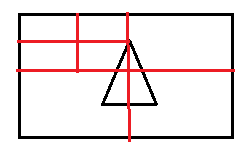
23. Определение факта выпуклости трехмерных тел. Разбиение тела на выпуклые многогранники.  
Для каждой грани тела выполнить: 1) Перенести тело так, чтобы одна из вершин грани оказалась в начале координат. 2) Повернуть тело вокруг начала координат так, чтобы одно из двух смежных выбранной вершине ребер грани совпало с одной из координатных осей.3) Повернуть тело вокруг выбранной оси координат так, чтобы выбранная грань легла на координатную плоскость.4) Для всех вершин тела, не принадлежащих выбранной грани, определить знаки координаты, которая перпендикулярна этой грани.5) Если знаки для всех вершин совпадают или равны нулю, то тело является выпуклым относительно выбранной грани; если знаки для всех вершин не совпадают, то тело невыпукло относительно очередной грани, если для всех вершин знаки, равны нулю, то тело вырождено, т.е. это плоский многоугольник. 6) Если тело оказалось невыпуклым, то его следует разрезать плоскостью, несущей выбранную грань. 7) Для каждого из вновь полученных тел повторить описанную процедуру до тех пор, пока все тела не станут выпуклыми

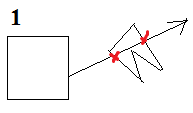
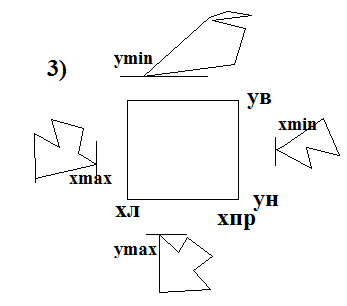
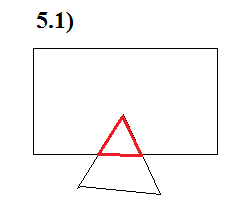
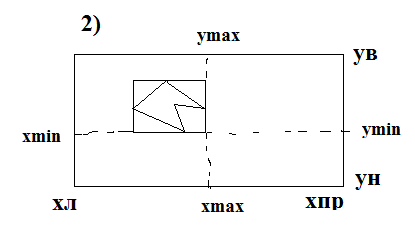
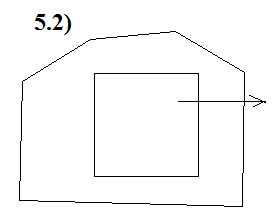
24. Алгоритм плавающего горизонта.  
F(x,y,z) = 0 – уравнение поверхности. Главная идея алгоритма заключается в сведении трехмерной задачи к двумерной путем пересечения исходной поверхности последовательностью параллельных секущих плоскостей, имеющих постоянные значения координат x,y или z. Зафиксируем z: y = φ(x, z = cons), . Алгоритм упорядочивает кривые пересечения плоскостей и поверхности по увеличению расстояния от наблюдателя, просматривает точки текущей кривой, если y текущей точки больше максимума (верхнего горизонта) или меньше минимума (нижнего горизонта) по y для всех предыдущих кривых при этом x, то текущая точка видима. Если при переходе к следующей точке видимость меняется, нужно найти точку пересечения. Точка пересечения ищется как пересечение двух отрезков: X = Xi + ∆X(Yтекi – Yпредi) / (∆Yтек - ∆Yпред), Y = m(X – Xi) + Yтекi., где ∆X = Xi+1 – Xi, ∆Yтек = Yтек(i+1) - Yтекi , ∆Yпред = Yпред(i+1) – Yпредi, m = (Yтек(i+1) – Yтекi) / ∆X. Для того, чтобы набор кривых выглядел как поверхность и устранения зазубренных боковых ребер создаются ложные боковые ребра, которые соединяют концы кривых. Алгоритм: 1) Для каждой секущей плоскости(z = const) выполнит ьдействия.1.1) Обработать левое боковое ребро: Если очередная точка кривой является первой и кривая первая, то запомнить эту точку(S). Если очередная точка (P), является первой и кривая не первая, то соединить т.P с т.S. Запомнить т.P в качестве т.S. 1.2) Для каждой точки, расположенной на кривой, полученной в текущей секущей плоскости выполнить действия. 1.2.1) Определить видимость точки: точка видима, если она расположена выше верхнего горизонта(Yтек(Xi) >Ymax(Xi)) или ниже нижнего горизонта(Yтек(Xi) >Ymin(Xi)). 1.2.2) Проверить изменение видимости текущей кривой на рассматриваемом интервале. В случае изменения видимости кривой найти точку пересечения кривой с соответствующим горизонтом. 1.2.3) Изобразить полностью участок(сегмент) текущей кривой, если очередная точка видима и видимость не изменилась. 1.2.4) Если очередная точка невидима и видимость не изменилась, ничего не изображать. 1.2.5) Если текущая точка видима и видимость изменилась, то изобразить участок кривой от точки пересечения до текущей точки. 1.2.6) Если текущая точка невидима и видимость изменилась, то изобразить участок кривой от предыдущей точки до точки пересечения. 1.2.7) Скорректировать в случае необходимости значения горизонтов. 1.3) Отобразить правое боковое ребро.

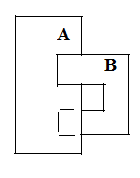
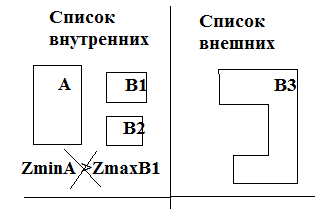
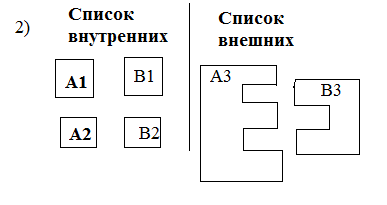
25. Задача удаления невидимых линий и поверхностей. Ее значение в машинной графике. Классификация алгоритмов по способу выбора системы координат (объектное пространство, пространство изображений).  
Задача удаления невидимых линий и поверхностей является одной̆ из наиболее сложных в машинной графике. Осложняется тем, что необходимо учитывать положение наблюдателя. Алгоритмы удаления невидимых линий и поверхностей служат для определения линий ребер, поверхностей или объемов, которые видимы или невидимы для наблюдателя, находящегося в заданной точке пространства. Единого и наилучшего способа решения задачи нет. Возникло множество алгоритмов, многие ориентированы на специализированные приложения. Все алгоритмы включают в себя сортировку. Главная – сортировка по расстоянию от точки наблюдения до тела-поверхности-ребра-точки. Так же есть сортировка по загораживанию тел друг другом. Эффективность алгоритмов в большой степени зависит от эффективности алгоритма сортировки. Существует связь между скоростью работы алгоритма и детальностью его результата – ни один алгоритм не может достигнуть хороших показателей одновременно. Алгоритмы делятся на: 1.Работающие в объектном пространстве (мировая ск, высокая точность). Сложность ~N2 2.Работающие в пространстве изображений (ск связана с дисплеем, точность ограничена разрешающей способностью дисплея). Сложность ~N\*M N – количество объектов, M – количество пикселей. N2<N\*M, но алгоритмы, работающие в пространстве изображений могут быть эффективнее в силу свойства когерентности при растровой реализации (близко расположенные пиксели чаще обладают одинаковыми свойствами).

26. Алгоритм Робертса. Основные этапы и математические основы каждого этапа.  
Алгоритм работает в объектном пространстве. В Алгоритме требуется, чтобы все тела и объекты были выпуклыми, невыпуклые разбить на выпуклые. Этапы: 1.Этап подготовки исходных данных.2.Этап удаления невидимых ребер, экранируемых самим телом. Если тело единственное – работу можно завершить.3. Этап удаления невидимых ребер, экранируемых другими телами. 4. Этап удаления невидимых ребер в случае взаимного протыкания тел. 1) Каждый объект может быть представлен в виде набора многогранников, которые могут быть описаны уравнениями плоскости. Каждая плоскость описывается так.x, y, z – мировые координаты. Векторы плоскостей составляют матрицу объекта где каждый столбец содержит коэффициенты одной плоскости. Корректность: каждая точка, расположенная внутри тела, располагается по положительную сторону от каждой грани. Получается положительное скалярное произведение , все компоненты P должны быть положительными. 2) Для определения невидимых граней нужно вектор наблюдения умножить на матрицу тела. В полученном векторе найти отриц. Компоненты. Это невидимые грани. Найдя невидимые грани, мы найдем невидимые ребра. Невидимые ребра образуют переход невидимых граней. Предполагается, что наблюдатель находится в бесконечности на положительной полуоси Z: [V]=[0 0-1 0] – направление взгляда. Скалярное произведение вектора наблюдения на столбец матрицы отрицательно – значит плоскость не видима (нелицевая сторона), иначе видима. Матрица преобразованного тела есть 3) Пусть задан отрезок в параметрической форме Требуется определить, существуют ли такие t, при которых данный отрезок невидим. Для этого соединим произв. точку с точкой наблюдения. Если прямая не пересекает ни одного многогранника (объекта), то точка видима. Прямую от точки R(t) до точки набл. представим в парам. Форме S – вектор координат начальной точки, f – точка на отрезке от точки наблюдения до R(t). Точка, находящаяся внутри объекта – невидима, относительно др. объектов. Следовательно, для проверки на экранирование вектор текущей точки умножают поочередно на матрицу каждого объекта: , Значения t и f, для которых все значения вектора Н положительны, соответствуют невидимой части отрезка. После замены где j - номер столбца в матрице объекта; w - вектор третьей строки матрицы объекта. Граница между видимой и невидимой частями отрезка определяется условием. Решая эту систему находят все значения *t* и *f,* для проверки того, что условие Hj > 0 выполнено. Так определяют минимальное значение параметра *t* среди максимальных () и максимальное среди минимальных ().Условие экранирования рёбер или отрезков ребер является простым следствием из классической задачи линейного программирования: < t <. К системе необходимо добавить условия, так как решения не могут существовать за пределами границы. Из соотношений видно, что является скалярным произведением концевой точки отрезка и j-й плоскости при t = 0, а ( ) является cкалярным произведением другой концевой точки отрезка и j - й плоскости. В свою очередь, j - я плоскость, ограничивающая объект, видима, если . Следовательно, если то отрезок целиком видим, а оба его конца лежат либо на видимой плоскости, либо между этой плоскостью и точкой наблюдения. Полностью невидимые отрезки определяют так же, как и частично невидимые (в этом случае невидимый отрезок будет простираться от *t = O* и *t = 1*). 4)В случае протыкания объектов сцены ищутся решения на границе f = 0), и вычисляют отрезки, которые образуются при протыкании объектами друг друга. Эти отрезки проверяют на экранирование всеми прочими объектами сцены. Видимые отрезки образуют структуру протыкания.

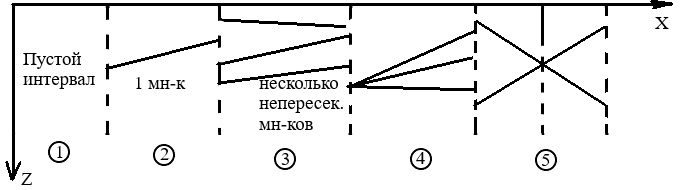


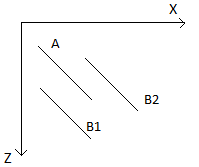
29.. Удаление невидимых линий и поверхностей в пространстве изображений. Алгоритм Варнока (разбиение окнами): последовательность действий и основные принципы.  
Одной версии алгоритма Варнока не существует. Выделяют простую версию алгоритма и более сложные. Окно делится на подокна всякий раз, если окно не пусто. Окно пусто, когда все многоугольники сцены являются внешними по отношению к этому окну. Пределом деления (в простейшем) является размер окна в один пиксель. По этому пикселю можно определить максимальную глубину и отобразить пиксель соответствующим цветом. Для двух пикселей глубину определить нельзя. В более сложных версиях алгоритма, когда окно размера>1 пикселя, ставится вопрос, о том, что изображается в этом окне. Рекомендуемая последовательность 1) Проводится простейший габаритный тест с прямоугольной оболочкой, определяется как можно большее количество пустых окон и окон с единственным внутренним многоугольником. 2)Выполнение теста с целью определения окон, пересекаемых единственным многоугольником. 3) Тест с целью распознавания внешних и охватывающих многоугольников. Получаем новые пустые окна и окна, охватываемые пустым многоугольником 4) Можно проводить разбиение окна на подокна или проводить ещё тест на обнаружение охватывающего многоугольника, лежащего ближе к наблюдателю

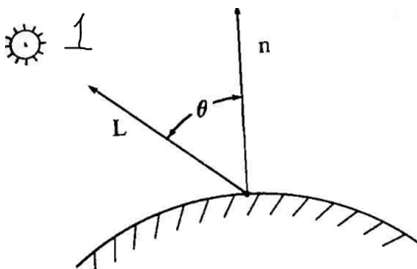
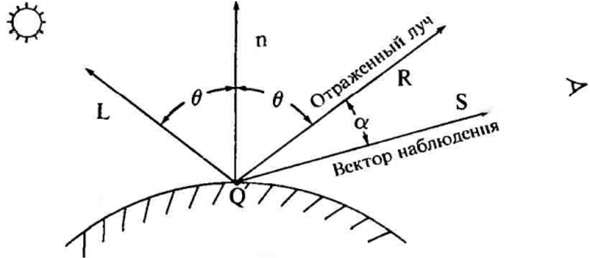
30. Типы многоугольников, анализируемых в алгоритме Варнока. Методы их идентификации.  
1) Все многоугольники внешние -> окно закрасить цветом фона  
2) Внутренний многоугольник (. Закрасить окно цветом фона и выполнить растровую развертку единого многоугольника.3) Внешние многоугольники.. Если все многоугольники – внешние закрасить цветом фона, если есть внутренние, продолжить деление. 4) Пересечения многоугольников Если для всех вершин окна знак пробной функции – одинаков, то эта сторона мно-ка пересекать окно не может. (Если пересекает, то ищем пересечение параметрической прямой, проходящей через пересекающее ребро с границами окна и проверяем корректность параметра t). 5)Охватывающие многоугольники. Выполнить отсечение мн-ка по сторонам отсекателя, Закрасить окно цветом фона, выполнить растровую развертку внутреннего многоугольника рис 5.1. Если пустить луч от окна через многоугольник, то если кол-во пересечений нечетно, то многоугольник охватывает отсекатель, необходимо окрасить цветом многоугольника рис 5.2. 6) Случай, когда с окном связано несколько мн-ков, но есть один охватывающий, расположенный ближе других к наблюдателю Для этого проводится тест глубины, однако он является достаточным, но никак не необходимым условием на рис 6 показана данная ситуация, нужно делить окно до тех пор пока не будет точно известно.

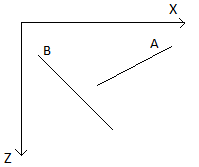
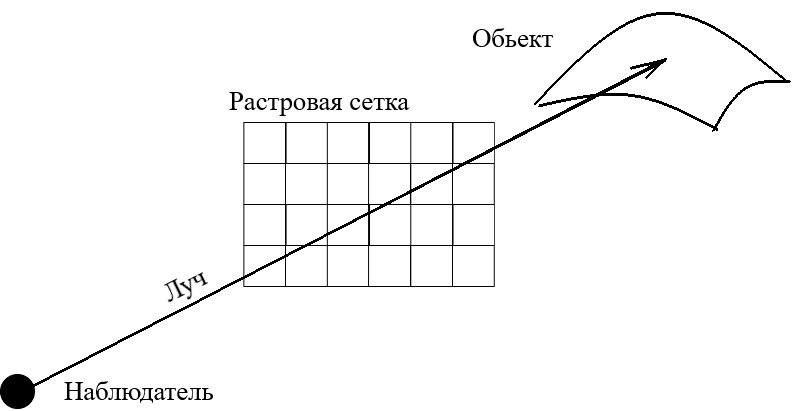
31. Алгоритм Вейлера-Азертона удаления невидимых линий и поверхностей.  
Алгоритм работает в объектном пространстве. В алгоритме Варнока необходимо выполнить много делений окна на подокна. В качестве отсекателя в этом алгоритме будет выбираться один из многоугольников сцены. Алгоритм базируется на алгоритме отсечения авторов, который позволяет находить как внутренние, так и внешние многоугольники. Алгоритм: 1. Сортировка многоугольников по глубине zmax. 2.Отсечение многоугольников сцены по границам ближайшего к наблюдателю многоугольника. 3. Удаление многоугольников, экранируемых ближайшим к наблюдателю многоугольником. 4.Устранение всех неопределенностей с помощью рекурсивного подразбиения многоугольника. После образования списка внутренних и списках внешних многоугольников, в списке внутренних проверяем, чтобы zmin Отсекаемого (для всех вершин) был больше zmax отсеченного, если это не так, то меняем их роли (отсекатель –> отсекаемый, отсекаемый -> отсекатель) и продолжаем проверку. На рис ниже представлена работа с циклически перекрывающимися многоугольниками. На 1 рисунке условие не выполнилось, на 2 Изобразить B1 и A2. Многоугольники А3 и B3 – внешние по отношению друг к другу, изобразить оба. Если многоугольники протыкания, то придется один из мн-ков разбить на два линией пересечения плоскостей, несущей многоугольники и продолжить работу алгоритма для полученных многоугольников.

32. Алгоритм, использующий Z-буфер.  
Этот алгоритм работает в пространстве изображения. Здесь используется идея о буфере кадра. Буфер кадра используется для заполнения атрибутов каждого пикселя в пространстве изображения. Z-буфер — это отдельный буфер глубины, используемый для запоминания координаты z каждого видимого пикселя в пространстве изображения. Вначале в Z-буфер заносятся минимально возможные значения z, а буфер кадра заполняется значениями пикселя, описывающими фон. Затем каждый многоугольник преобразуется в растровую форму и записывается в буфер кадра, при этом, однако, не производится начального упорядочения. Преимущества: простота, отсутствие сортировки (возможность обрабатывать многоугольники в произвольном порядке). Недостатки: требует большого объема памяти, трудоемкость устранения ступенчатости, трудоемкость реализации эффектов прозрачности и просвечивания. Алгоритм: 1)Всем элементам буфера кадра присвоить фоновое значение (цвет фона).2) Инициализировать z-буфер минимальным значением глубины.3) Преобразовать каждый многоугольник в растровую форму в произвольном порядке. (Выполнить растровую развертку каждого многоугольника) 4)Вычислить для каждого пикселя (x, y), принадлежащего многоугольнику, глубину z(x, y). 5)Сравнить глубину многоугольника со значением, которое хранится в z-буфере для соответствующего пикселя: Если z(x, y) > zбуф(x, y), то zбуф(x, y) = z(x, y) (корректировка z-буфера), цвет(x, y) = цвет текущего многоугольника (занесение информации в буфер кадра). 6)Вывести содержимое буфера кадра на экран. Вычисление z: Уравнение плоскости, несущей многоугольник: ax+by+cz+d = 0; z = - (ax+by+d) / c. Для сканирующей строки (y = const) глубина пикселя (x2, y), для которого x2 = x1+Δx = x1 + 1, равна z2 = z1 + Δz, где Δz = - (ax1+by+d) / c - (ax2+by+d) / c = (-x2 + x1) \* (a / c) = -a / c. Если c = 0, то многоугольник параллелен оси z, поэтому для наблюдатель увидит только ребро. Необходимо вычислить глубину ребра. Для горизонтальных и вертикальных ребер уравнения прямых, несущие эти ребра. (z – z1) / (z2 – z1) = (x – x1) / (x2 – x1), (x1, z1) и (x2, z2) - вершины видимого ребра, (z – z1) / (z2 – z1) = (y – y1) / (y2 – y1), (y1, z1) и (y2, z2) - вершины видимого ребра. Если и x2 – x1 = 0 и y2 – y2 = 0, то со стороны наблюдателя многоугольник вырождается в точку, и значение глубины будет соответствовать координате z в этой точке (вершине многоугольника).



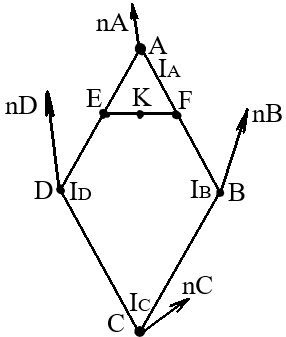
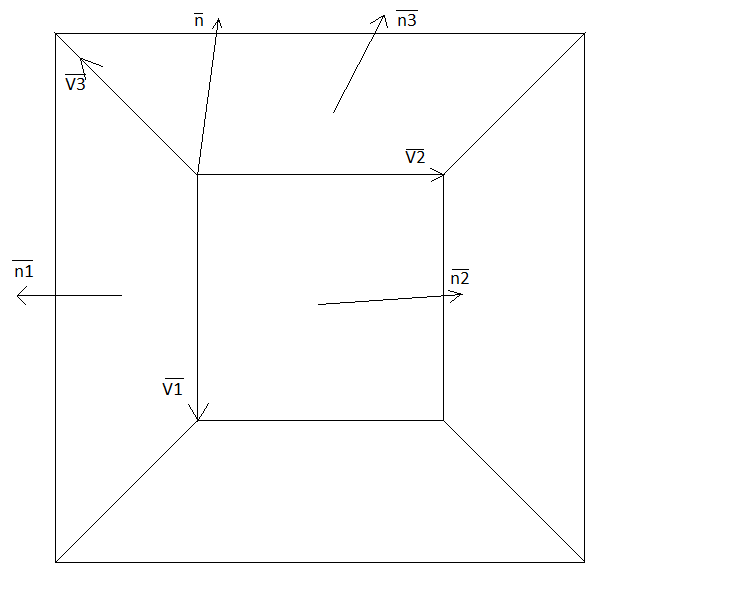
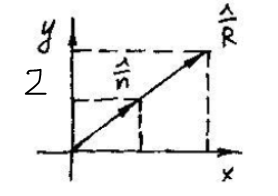
33. Алгоритм, использующий список приоритетов.  
(алгоритм художника)Способ изображения сцены от наиболее удаленных от наблюдателя объектов до ближайших. Формируется список приоритетов по глубине, используя в качестве ключа значение zmin для каждого многоугольника. Прежде чем изображать очередной многоугольник, необходимо проверить, не будет ли в дальнейшем этот многоугольник экранировать другие. Если вторым после многоугольника А будет многоугольник B1, то изображать A можно, если B2, то нельзя, так как тогда B2 будет экранировать A, а на самом деле это не так. Для проверки необходимо будет провести ряд тестов для текущего многоугольника А и последующего в списке приоритетов B, сформулированных в виде вопросов. Алгоритм: 1)Сортировка многоугольников по глубине (по zmin) в порядке возрастания. 2) Проверить, если zmaxA< zminB, то А не экранирует B и изображается А. 3)Иначе проверить, экранирует ли многоугольник А многоугольник B при zmaxA> zminB. Если получен хотя бы один положительный ответ на тесты, то изображается многоугольник A (A не экранирует B). Если ни на один из тестов не был дан положительный ответ, то многоугольники А и В необходимо поменять местами в приоритетном списке, зафиксировать факт обмена и провести все тесты для нового многоугольника. Если и для этого многоугольника не было положительных ответов, то многоугольники циклически перекрываются, в таком случае необходимо разбить один из многоугольников на 2 части плоскостью, несущей другой многоугольник. Исходный удалить из списка, а его части заносятся в список и тесты повторяются для нового списка. Тесты: 1)Верно ли, что прямоугольные объемлющие оболочки многоугольников А и В не пересекаются по оси x? 2) Верно ли, что прямоугольные объемлющие оболочки многоугольников не перекрываются по оси y? 3)Верно ли, что многоугольник А целиком расположен по ту сторону плоскости, несущей многоугольник B, которая расположена дальше от наблюдателя? 4) Верно ли, что многоугольник В целиком расположен по ту сторону плоскости, несущей многоугольник А, которая расположена ближе к наблюдателю? 5) Верно ли, что проекции многоугольников А и В не пересекаются? Для пунктов 1 и 2 проверяются координаты объемлющих оболочек. Для пунктов 3 и 5 можно воспользоваться пробной функцией f = Ax + By + Cz + D, где A, B, C, D - коэффициенты уравнения плоскости, несущей многоугольник P. Координаты каждой вершины многоугольника Q подставляются в пробную функцию. Если знаки пробной функции для всех вершин совпадают, то Q целиком лежит по одну сторону от P. Для того, чтобы определить по какую сторону лежит многоугольник, необходимо сравнить получившийся знак со знаком пробной функции точки, положение которой уже известно. Для пункта 5 можно использовать алгоритм Варнока.

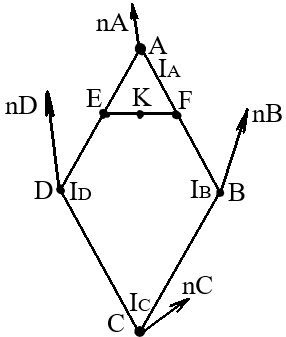
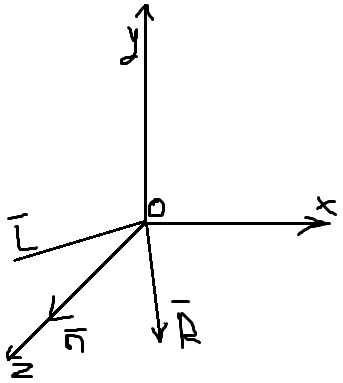
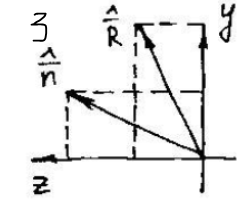
34. Алгоритм построчного сканирования, использующий Z-буфер. Интервальные методы построчного сканирования (основные предпосылки).  
Этот алгоритм работает в пространстве изображения. Для него требуется буфер кадра, в котором запоминаются значения цвета, а также Z-буфер, куда помещается информация о координате z для каждого пикселя. Вначале в Z-буфер заносятся минимально возможные значения z, а буфер кадра заполняется цветом фона. По сравнению с обычным алгоритмом, алгоритм построчного сканирования использует значительно меньше памяти(ширину экрана для каждого буфера, а не ширину \* высоту)Вычисление глубины каждого пикселя на сканирующей строке: Ax + By + Cz + D = 0, z = -(Ax + By + D)/C при C!=0. Для сканирующей строки y = const, xA = xB + 1, ∆z = zB – zA = (AxB + By + D) / C - (AxA + By + D) /C = A \* (xB - xA) / C = A / C. Если вершины ребра лежат по разные стороны от сканирующей строки, то ребро ее пересекает. Точка пересечения находится из соотношения (z – z1) / (z2 – z1) = (y – y1) / (y2 – y1). С помощью САР(список активных ребер) и САМ(список активных многоугольников) можно избежать операции умножения и деления. В САМ многоугольники делятся по Y-группам, для каждого многоугольника хранится кол-во пересекаемых строк, список рёбер многоугольника, коэффициенты уравнения плоскости многоугольника, визуальные атрибуты многоугольника. В списке САР хранятся пары ребер, для каждой пары хранятся: xЛ – пересечение левого ребра с текущей сканирующей строкой, ΔxЛ – приращение xЛ в интервале между соседними сканирующими строками, ΔyЛ – число сканирующих строк, пересекаемых левым ребром идентично для правого ребра xП, ΔxП, ΔyП, ΔzX =-A/C– приращение z при изменении x, ΔzY = -B/C – приращение z при изменении y. Если текущий y <= Y группы, то многоугольники группы активны. Для каждого активного многоугольника сделать нужные пары ребер активными. Для каждой активной пары рёбер многоугольника из САР выполнить: Для каждого пикселя xЛ<=x<=xП вычислить z(x,y=const). z1=zЛ, ..., zK+1 = zK+ ΔzX, если z >zбуф(x), то zбуф(x) = z и занести атрибуты многоугольника в буфер кадра; ΔxЛ--, ΔyП--, xЛ=xЛ+ΔxЛ, xП=xП+ΔxП, zЛ=zЛ+ΔzXΔx+ΔzY. Если у одного из ребер Δy<= 0, то удалить это ребро из пары и если многоугольник еще активен, найти ему новую пару. Уменьшить Δy многоугольника и в случае Δy<=0 удалить многоугольник из САМ. Перейти на следующую сканирующую строку. В этом алгоритме глубина многоугольника вычисляется для каждого пикселя на сканирующей строке. Количество вычислений можно сократить, если использовать понятие интервалов. Решение задачи удаления невидимых поверхностей сводится к выбору видимых отрезков в каждом интервале, полученном путём деления сканирующей̆ строки проекциями точек пересечения рёбер. Случаи: 1) Занести в буфер кадра цвет фона, 2) занести в буфер кадра параметры многоугольника, 3) Занести в буфер кадры параметры максимального по z многоугольника, 4) занести в буфер кадра параметры многоугольника максимального по z посередине интервала, 5) Пересечение есть, если sign(zн1 - zн2) != sign(zк1 - zк2), в этом случае разделить интервал на 2 по точке пересечения и перейти к случаю 4.

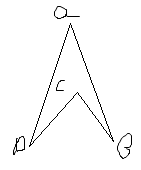
35. Алгоритм определения видимых поверхностей путем трассировки лучей.  
Решается в пространстве изображения. Идея заключается в том, что наблюдатель видит объект благодаря лучам, доходящим к нему от объекта. В алгоритме луч направлен от наблюдателя к объекту. Проверяется пересекает ли луч объемлющую оболочку объекта. В качестве объемлющей оболочку удобно взять сферу или прямоугольный параллелепипед. Для сферы: P(T) = P1 + (P2 – P1)t, t>= 0; x(T) = x1 + (x2 – x1)t; y(T) = y1 + (y2 – y1)t; z(T) = z1 + (z2 – z1)t; l2= (x1 + (x2 – x1)t – x0)2 + (y1 + (y2 – y1)t – y0)2 + (z1 + (z2 – z1)t – z0)2, dl2/dt = 2(a + (x2 – x1)t)(x2-x1) + 2(b + (y2 – y1)t)(y2-y1) + 2(c + (z2 – z1)t)(z2-z1) = 0, tmin = -(a(x2-x1) + b(y2-y1)+c(z2-z1)) / ((x2-x1)2 + (y2 – y1)2 + (z2 – z1)2). Для прямоугольного параллелепипеда: выполнит преобразование, совмещающие луч с осью z: (xmin < 0) ^ (xmax > 0) ^ (ymin < 0) ^ (ymax > 0). Нахождение пересечения с поверхностью второго порядка: Q(x, y, z) = a1x2 + a2y2 + a3z2 + b1xy + b2xz + b3yz + c1x + c2y + c3z + d = 0. Применяя преобразование: x = y = 0, QПР(z) = a1’z2 + c3’z + d = 0, z1,2 = -c3’ ± sqrt(c3’2 – 4 a3’d) / 2a3’. Если под корнем – то пересечения нет. Алгоритм: 1) Подготовка исходных данных: Для каждой поверхности задать коэффициенты и тип. 2) Вычислить параметры объемлющей оболочки. 3) Для каждого трассирующего луча выполнить: Выполнить тест в исходной системе координат со сферической оболочкой для каждой поверхности. Если пересечение найдено, то соответствующее тело занести в список активных тел. Если список пуст, высветить текущих пиксель цветом фона. Иначе найти преобразование, совмещающее луч с осью z(поворот и перенос). Для каждого объекта списка активных тел: Если используется прямоугольная объемлющая оболочка, то выполнить преобразование оболочки в новую систему координат и выполнить тест. Если пересечение луча с оболочкой установлено, то выполнить преобразование тела и найти пересечение, занести его в список пересечений. Иначе перейти к следующему объекту. Если список пересечений пуст, высветить цвет фона. Иначе выбрать пересечение с максимальной глубиной и закрасить цветом ближайшей поверхности, либо учитывая модель освещения. Если учитывается освещение, то либо преобразовать источник освещения, либо выполнить обратное преобразование для самой точки.

36. Построение реалистических изображений. Физические и психологические факторы, учитываемые при создании реалистичных изображений. Простая модель освещения.  
При построении реалистичного изображения необходимо: 1) Учитывать оптические свойства поверхностей 2) Воспроизводить рисунок на поверхности 3) Воспроизводить неровности 4) учитывать, что поверхности отбрасывают тени 5) Учитывать восприятие окружающего мира человеческим глазом. Простая модель освещения. Объект можно увидеть, если он отражает или пропускает свет, если же объект поглощает весь падающий свет, то он будет невидим. Виды отражения: 1) Диффузное рис 1. 2)Зеркальное рис 2. При диффузном считается, что тело поглощает падающую энергию. Диффузно отраженная энергия распространяется равномерно в пространстве (во всех направлениях). Диффузное отражение происходит, когда свет как бы проникает под поверхность объекта, поглощается, а затем вновь испускается. I=Iи kдcosθ 0≤θ≤π/2 где I – интенсивность отраженного света; Iи – интенсивность точечного источника; kд – коэффициент диффузного отражения (0≤ kд ≤1); θ - угол между направлением света и нормалью к поверхности. Но помимо обычного источника есть еще и рассеивающие, следовательно формула будет выглядеть так: . Ip kp – рассеивания. Так как интенсивность зависит от расстояния пройденного лучем можно формулу записать в следующем виде, учитывая что интенсивность пропорциональна не квадрату расстояние, а линейному., d – расстояние, К – константа, чтобы избежать деления на ноль(подбирается в зависимости от эксперимента). Зеркальное – присуще гладким поверхностям (не проникают внутрь) Цвет зеркально отраженного луча совпадает с цветом падения. Благодаря зеркальному отражению на блестящих предметах появляются световые блики. Интенсивность зеркально отраженного света зависит от угла падения, длины волны и свойств вещества: Iз= Iиw(θ,λ)cosnα, где w(θ,λ) – функция отражения, представляющая отношение зеркально отраженного света к падающему а n – степень гладкости поверхности/ Учитывая рассеивание и диффузное отражение I= Iр kр + Iu(kдcosθ+ w(θ,λ)cosnα)/(d + K). Функцию w(θ,λ) заменяют константой kз, которая выбирается из эстетических соображений или определяется экспериментально. В итоге I= Iр kр + (kдcosθ+ kзcosnα). Если имеется несколько источников света, то интенсивности от отдельных источников суммируются I= Iр kр + (kдcosθj+ kзcosnαj), где m – количество источников света.

37. Построение реалистических изображений. Метод Гуро закраски поверхностей (получение сглаженного изображения).

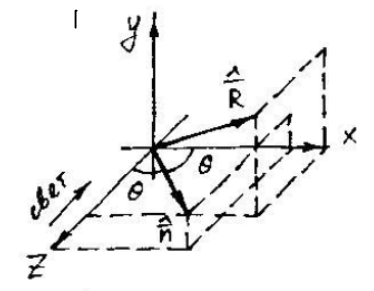
Существуют три основных способа закраски объектов: простая закраска, закраска по Гуро(интерполяцией интенсивности), закраска по Фонгу(интерполяция векторов нормалей).В простой закраске каждая грань закрашивается одним уровнем интенсивности. При ее использовании предполагается что: 1)Источник света расположен в бесконечности, поэтому угол между падающим лучом и нормалью постоянен на всей грани. 2) Наблюдатель находится в бесконечности, поэтому угол между вектором наблюдения и отражения постоянен на всей полигональной грани. 3) Закрашиваемая грань является реально существующей, а не получена в результате аппроксимации. Использование этого типа закраски приводит к тому, что переход от одной грани к другой очень заметен. Для сглаживания используют метод Гуро. 1) Вычисляются нормали к поверхности. 2) Определяются нормали в вершинах путем усреднения нормалей по всем полигональным граням, которым принадлежит рассматриваемая вершина.3) Вычисляются значения интенсивностей в вершинах многоугольника. 4) Осуществляется закраска многоугольника с использованием билинейной интерполяции интенсивностей. Сначала вычисляются значения интенсивностей вдоль каждого ребра (первая линейная интерполяция), а затем вычисляются значения интенсивностей вдоль сканирующей строки между ребрами многоугольника (вторая линейная интерполяция). IE = (1-v) IA + v ID, где v = AE/AD, IF = (1-w) IA + w IB , где w=AF/AB, IK = (1-t) IE + t IF , где t=EK/EF. Инкрементальное вычисление для сканирующей строки: IK1 = (1-t1) IE + t1 IF, IK2 = (1-t2) IE + t2 IF , IK2= IK1 + IF(v2 - v1) + IE(v1-v2 ) = IK1 +(IF – IE) (t2 - t1) = IK1 +ΔIΔt . Этот тип закраски хорошо сочетается с диффузным отражением. Недостаток: усреднение нормальей вследствие чего поверхность может казаться плоской. Если нужно сохранить переход то усреднение не делается. Иначе вводятся дополнительные многоугольники.

38. Построение реалистических изображений. Закраска Фонга (улучшение аппроксимации кривизны поверхности).  
Для построения реалистичности изображений необходимо учитывать закраску поверхностей. В зависимости от положения объекта наблюдателя и источника освещения, объект может закрашиваться разными уровнями интенсивности. Каждая грань тела закрашивается одним уровнем интенсивности, если: 1) Источник света находится в бесконечности. 2) Наблюдатель находится в бесконечности. 3) Закрашиваемый многоугольник реально существует, а не получен в результате аппроксимации поверхности. Если сферическая поверхность аппроксимирована полигонами, то каждый будет закрашен разными цветами, и мы увидим ребра, а их быть не должно. Необходимо улучшать изображение. Фонг предложил сглаживание изображения путем билинейной интерполяции нормали. Метод Фонга требует больших вычислительных затрат в отличии от метода Гуро. Сначала вычисляются нормали к поверхностям, затем на основе этих нормалей вычисляются значения нормалей в вершинах многоугольников. Далее выполняется аппроксимация вдоль ребер.



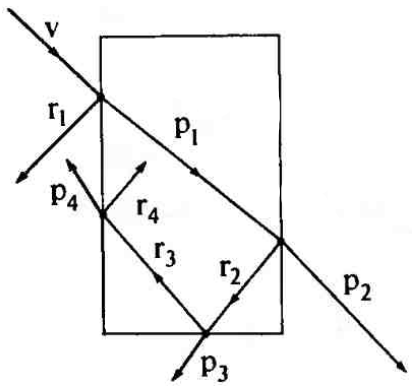
Значение нормали, вычисленное описанным способом, используется для вычисления интенсивности пиксела. Закраска по методу Фонга достигается лучшая локальная аппроксимация кривизны поверхности и, следовательно, получается более реалистическое изображение, в частности, правдоподобнее выглядят зеркальные блики. Так же есть ситуация при которой может быть резкий скачек интенсивности в точке с при интерполяции

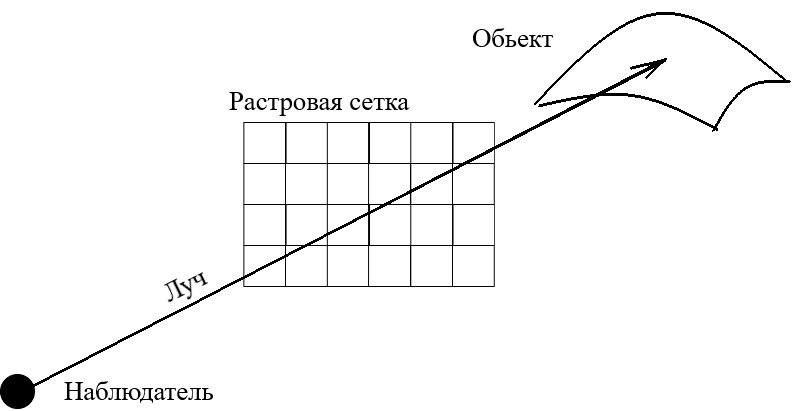
39. Определение нормали к поверхности и вектора отражения (4 способа) в алгоритмах построения реалистических изображений.  
Определение нормали к поверхности. (нужны только внешние нормали) 1)Для того, чтобы вычислить нормаль к поверхности, можно векторно умножить 2 вектора, являющихся касательными к поверхности. Эти векторы можно определить через частные производные функции, задающей исходную поверхность, по направлениям: ∂Q(x, y)/∂x, ∂Q(x, y)/∂y. Во многих алгоритмах удаления невидимых линий и поверхностей используются только ребра и вершины, поэтому при объединении их с моделью освещения необходимо знать приближенное значение нормали на ребрах и в вершинах. Пусть заданы уравнения плоскостей, несущих грани модели, тогда нормаль к их общей вершине равна среднему значению нормалей ко всем этим плоскостям: n = n1 + n2 + n3 = (a1 + a1 + a3) i + (b1 + b2 + b3) j + (c1 + c2 + c3) k, где a1-a3, b1-b3, c1-c2 - коэффициенты уравнений плоскостей. (если требуется найти только направление нормали, то делить результат на количество граней не нужно). Нормаль к вершине также можно определить, усредняя векторные произведения всех ребер, пересекающихся в вершине: n = [V1, V2] + [V2, V3] + [V3, V1]. Определение вектора отражения (вектор нормали, вектор падения и вектор отражения лежат в одной плоскости, угол падения равен углу отражения): 1)Этот случай удобен, когда источник света расположен на оси z и когда он единственный. Начало системы координат переносится в точку поверхности, тогда проекции нормали и вектора отражения на плоскость xy лежат на одной прямой. (Все действия с единичными векторами) , . Из второго рисунка по подобию треугольников:, .Из третьего рисунка: , , , , , ,   
2) Этот способ удобен в том случае, если в сцене несколько источников освещения. Совмещение вектора нормали с осью z (единичные векторы): , , , где вектор L – вектор падения. 3) Условие того, что векторы падения, нормали и отражения лежат в одной плоскости, можно записать так: . Составляем определитель для первого и второго векторных произведений через знак равенства. Раскрываем определитель: =   
Получаем систему из 3х уравнений с 3мя неизвестными. Система ЛЗ, поэтому нужно добавить еще одно уравнение, используя скалярное произведение:, . 4)Аналитический метод  
- скалярное произведение,   
Представим вектор R как линейную комбинацию векторов L и n: , , , , , , , , , , ,   
Векторы n и R сонаправлены, поэтому ,

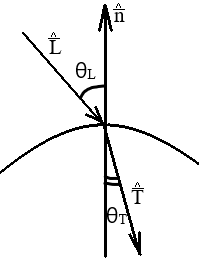
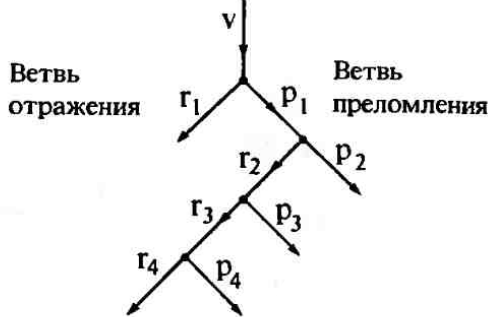
40. Построение теней при создании реалистических изображений. Учет теней в алгоритмах удаления невидимых поверхностей.  
Тень не зависит от положения наблюдателя. Если направление вектора наблюдения совпадает с вектором падения света, то тень не видна. Тень состоит из двух частей: полная тень — центральная, темная, резко очерченная часть; полутень — окружает полную, более светлая часть тени. Различают собственные тени — когда объект препятствует попаданию света на некоторые свои грани, и проекционные тени — когда один объект препятствует попаданию света на поверхность другого. Видимые тени — те, которые видны со стороны наблюдателя. Процесс построения тени: удалить невидимые поверхности для положения каждого источника; удалить невидимые поверхности для положения наблюдателя. Построение собственной тени: удаление нелицевых граней, если точку наблюдения совместить с источником света (алгоритм Робертса).Построение проекционной тени: построить проекции всех нелицевых граней на сцену, если точку наблюдения совместить с источником света. Точки пересечения проецируемой грани со всеми другими плоскостями образуют многоугольники, которые помечают как теневые и заносят в список.  
Интервальный метод построчного сканирования:1)Для каждого источника и многоугольника определить собственные тени и проекционные тени. Они записываются в виде двоичной матрицы – строки – многоугольники, отбрасывающие тень, столбцы – затеняемые многоугольники 2)Обработка сцены относительно положения наблюдателя:1.Определить видимые отрезки на интервале 2.С помощью списка теневых многоугольников определить, падает ли тень на многоугольник, которые создаёт видимый отрезок на данном интервале a)нет теневых многоугольников – изобразить отрезок b) есть теневые многоугольники, но они не пересекают и не покрывают данный интервал – изобразить видимый отрезок c)интервал полностью покрывается хотя бы одним многоугольником – интенсивность изображаемого отрезка определяется с учётом интенсивностей этих многоугольников и самого отрезка d)хотя бы один теневой многоугольник частично покрывает интервал – рекурсивно разбивать интервал и работать с подыинтервалами. Алгоритм, использующий Z-буфер: 1)Строится сцена, где точка наблюдения совпадает с источником. Запомнить значения Z в отдельном теневом Z-буфере. 2)Строится сцена из точки наблюдателя. При обработке каждой поверхности глубина каждого пикселя сравнивается с глубиной Zбуф наблюдателя. Если > (видима), то xyz из вида наблюдателя преобразуются в x' y' z' на виде из источника. Для того, чтобы проверить, видимо ли значение z' из положения источника, она сравнивается со значением теневого z-буфера при x' y'. Если оно видимо, то оно отображается в буфер кадра x, y без изменений. Если нет, то точка находится в тени и изображается согласно правилу расчета интенсивности с учетом затемнения, а значение в z-буфере наблюдателя заменяется на z'. Алгоритм Вейлера-Азертона (этапы): 1)С помощью алгоритма удаления невидимых поверхностей выделяются освещенные, видимые из положения источника, грани (хранятся именно они). 2)Освещенные многоугольники помечаются и преобразуются к исходной ориентации, где приписываются к своим прототипам (путем присвоения своего номера каждому многоугольнику или его части). 3)Объединенные данные о многоугольниках обрабатываются со стороны наблюдателя. Если какая-либо область не освещена, то применяется соответствующее правило расчета интенсивности с учетом затемнения.  
 (для нескольких источников света будет добавляться несколько наборов освещенных граней) Алгоритм трассировки лучей (этапы): 1) Трассировкой лучей от точки наблюдения через плоскость проекции определяются видимые точки сцены. 2)Трассировка лучей от видимой точки до источника света. Если на пути луча в сцене встретился какой-то объект, то свет от источника не попадает в данную точку и она оказывается в тени.

41. Учет прозрачности в модели освещения. Учет прозрачности в алгоритмах удаления невидимых поверхностей.  
Зеркальное (направленное) – свойственно прозрачным телам. Если смотреть на объект сквозь такое вещество, то за исключением контурных линий криволинейных поверхностей, искажения происходить не будет. Диффузное (рассеянное) – вещества кажутся полупрозрачными или матовыми. Чтобы включить преломление в модель освещения нужно при построении видимых поверхностей учитывать не только падающий, но и отражённый и пропущенный свет. Эффективнее это делается с помощью глобальной модели освещения в сочетании с алгоритмом трассировки лучей для выделения видимых поверхностей. В более простых реализациях эффект преломления не рассматривается, не принимается во внимание как путь, пройденным лучом в среде, влияет на его интенсивность. Простое пропускание света можно встроить в любой алгоритм, кроме алгоритма с Z-буфером. Прозрачные поверхности помечаются и если видимая грань прозрачна, то в буфер кадра записывается линейная комбинация двух ближайших поверхностей. При этом интенсивность: , 0<=t<=1. Где - видимая поверхность, - поверхность, расположенная непосредственно за ней, t – коэффициент прозрачности первой поверхности. Если вторая поверхность тоже прозрачна, то алгоритм применяется рекурсивно, пока не встретиться непрозрачная поверхность или фон.   
Алгоритм с z-буфером (нужны отдельные буферы прозрачности, интенсивности, весовых коэффициентов):1) Для каждого многоугольника 2)Если прозрачен – занести в список прозрачных 3)Если нет и Z>Zбуф, то записать в буфер кадра для непрозрачных многоугольников и скорректировать Zбуф 4)Для каждого многоугольника из списка прозрачных 5)Если Z≥Zбуф, то прибавить его коэффициент прозрачности к значению в буфере весовых коэфициентов прозрачности, прибавить его интенсивность к значению в буфере интенсивности прозрачности по правилу: , (bn– новое, bo – старое, с – для многоугольника). Таким образом получается взвешенная сумма интенсивностей всех прозрачных многоугольников, находящихся перед ближайшим непрозрачным. 6)Объединим буферы интенсивности для прозрачных и непрозрачных многоугольников : (Ifb – окончательная интенсивность в буфере кадра, Ifbo – старое значение интенсивности в этом буфере).

42. Учет фактуры при создании реалистических изображений.  
Под фактурой в компьютерной графике понимается детализация строения поверхности. Рассматривают 2 вида детализации: 1) На гладкую поверхность наносят рисунок, в этом случае поверхность остается гладкой. 2) Во втором случае ставится задача создания неровностей на поверхности, т.е создание шероховатой поверхности. Существующие способы моделирования текстуры можно подразделить на: 1) Проективные текстуры 2)Процедурные текстуры. В первом случае задается рисунок, в некоторой системе координат и ведется поиск функции отображения координат из исходной системы координат в нужную. Находим точки отображения, анализируем, В первом случае для точек плоского рисунка находим соответствующие им координаты точек на поверхности, во втором случае для точек поверхности находим соответствующие им точки на плоскости и анализируем, принадлежит ли полученная точка рисунку или нет. Если принадлежит, то исходную точку поверхности закрашиваем, в противном случае – нет.  
Нанесение на поверхность рисунка можно рассматривать как детализацию цветом. Для ее реализации используются многоугольники детализации. Они лежат в той же плоскости что и основные, но помечаются так, чтобы в алгоритмах удаления невидимых поверхностей могли присвоить им более высокие приоритеты. Во втором случае необходимо найти функцию C(X,Y,Z), определяющую для каждой точки поверхности цвет таким образом, чтобы он соответствовал цвету моделируемого материала (наносимого рисунка). Такие текстуры называют процедурными. Например для дерева можно использовать следующую функцию Q(X,Y)= C1 +(C2 – C1)f()где C1,C2 – некоторые цвета, соответствующие светлым и темным кольцам структуры дерева; f(t) – некоторая неотрицательная периодическая функция. Пусть Q(X,Y) – уравнение поверхности, т.е. функция Q позволяет для каждой точки поверхности определить ее третью координату Z. Нормаль в точке поверхности может определяться векторным произведением этих производных Q’x, Q `y: ; Пусть P(X,Y) – функция возмущения, тогда радиус-вектор точки на новой поверхности будет определяться , тогда , а

43. Глобальная модель освещения с трассировкой лучей.  
Модель освещения предназначена для расчета интенсивности отраженного к наблюдателю света в каждой точке изображения. Разделяют локальную и глобальную модели освещения. Локальная модель учитывает только свет, падающий от источника, и ориентацию поверхности: I = Iрkp + Iиkдcosθ + Iиkзcosnα. Глобальная модель учитывает также свет, отраженный от других объектов или пропущенный сквозь них: I = Iрkp + Iиkдcosθ + Iиkзcosnα +kзIзcosφ + kпрIпрcosψ. При трассировке отслеживают траекторию луча, который после столкновения с поверхностью частично поглощается, частично диффузно рассеивается, а остальная энергия луча идёт на зеркальное отражение и преломление. Если источник света видим из точки пересечения, то сначала вычисляют вклад этого источника в освещённость рассматриваемой точки, используя модель отражения. Затем трассируемый луч разделяют на составляющие отражения и преломления. Эти две составляющие в дальнейшем будут анализироваться, как и начальные приведённые лучи. При столкновении такого вторичного луча с поверхностью могут возникнуть новые вторичные лучи. Трассировку организуют как рекурсивную процедуру, которая сама себя вызывает, как только выясняется, что анализируемый луч отражается или преломляется. Процедура выполняется до тех пор, пока не останется пересечений либо пока интенсивность не станет слишком мала, чтобы ее учитывать. При таком алгоритме строится дерево, правые ветви которого это преломленные лучи, левые – отраженные, а узлы соответствуют точкам пересечения с поверхностями. Для расчёта интенсивности в точке нужно пройти дерево в обратном порядке. Таким образом, в отличие от простой модели освещения, глобальная модель дополнительно учитывает зеркальное отражение от других поверхностей, попадающее в рассматриваемую точку, а также преломление, т.е. интенсивность, приносимую преломленным лучом. Кроме того, в расчетах надо учитывать ослабление пропущенного и зеркально отраженного света с расстоянием между точками пересечения.

44. Алгоритм трассировки лучей с использованием глобальной модели освещения.  
Решается в пространстве изображения. Идея заключается в том, что наблюдатель видит объект благодаря лучам, доходящим к нему от объекта. В алгоритме луч направлен от наблюдателя к объекту. Проверяется пересекает ли луч объемлющую оболочку объекта. В качестве объемлющей оболочку удобно взять сферу или прямоугольный параллелепипед. Для сферы: P(T) = P1 + (P2 – P1)t, t>= 0; x(T) = x1 + (x2 – x1)t; y(T) = y1 + (y2 – y1)t; z(T) = z1 + (z2 – z1)t; l2= (x1 + (x2 – x1)t – x0)2 + (y1 + (y2 – y1)t – y0)2 + (z1 + (z2 – z1)t – z0)2, dl2/dt = 2(a + (x2 – x1)t)(x2-x1) + 2(b + (y2 – y1)t)(y2-y1) + 2(c + (z2 – z1)t)(z2-z1) = 0, tmin = -(a(x2-x1) + b(y2-y1)+c(z2-z1)) / ((x2-x1)2 + (y2 – y1)2 + (z2 – z1)2). Для прямоугольного параллелепипеда: выполнит преобразование, совмещающие луч с осью z: (xmin < 0) ^ (xmax > 0) ^ (ymin < 0) ^ (ymax > 0). Нахождение пересечения с поверхностью второго порядка: Q(x, y, z) = a1x2 + a2y2 + a3z2 + b1xy + b2xz + b3yz + c1x + c2y + c3z + d = 0. Применяя преобразование: x = y = 0, QПР(z) = a1’z2 + c3’z + d = 0, z1,2 = -c3’ ± sqrt(c3’2 – 4 a3’d) / 2a3’. Если под корнем – то пересечения нет. Алгоритм: 1) Подготовка исходных данных: Для каждой поверхности задать коэффициенты и тип. 2) Вычислить параметры объемлющей оболочки. 3) Для каждого трассирующего луча выполнить: Выполнить тест в исходной системе координат со сферической оболочкой для каждой поверхности. Если пересечение найдено, то соответствующее тело занести в список активных тел. Если список пуст, высветить текущих пиксель цветом фона. Иначе найти преобразование, совмещающее луч с осью z(поворот и перенос). Для каждого объекта списка активных тел: Если используется прямоугольная объемлющая оболочка, то выполнить преобразование оболочки в новую систему координат и выполнить тест. Если пересечение луча с оболочкой установлено, то выполнить преобразование тела и найти пересечение, занести его в список пересечений. Иначе перейти к следующему объекту. Если список пересечений пуст, высветить цвет фона. Иначе выбрать пересечение с максимальной глубиной и выполнить обратное преобразование для самой точки. Затем вычислить интенсивность точки с учетом глобальной модели освещения и высветить пиксель с полученной интенсивностью. . Глобальная модель учитывает свет, падающий от источника, ориентацию поверхности, свет, отраженный от других объектов или пропущенный сквозь них: I = Iрkp + Iиkдcosθ + Iиkзcosnα +kзIзcosφ + kпрIпрcosψ. При трассировке отслеживают траекторию луча, который после столкновения с поверхностью частично поглощается, частично диффузно рассеивается, а остальная энергия луча идёт на зеркальное отражение и преломление.

45. Определение направления преломленного луча.  
В более сложных моделях освещения учитываются пропускающие свойства поверхностей, т.е. прозрачность этих поверхностей, что приводит к наличию преломленного луча. Закон преломления: падающий и преломленной лучи, а также вектор нормали лежат в одной плоскости, а углы падения и преломления связаны следующей формулой: η1sinθL= η2 sinθT . Вектор преломления представляет собой линейную комбинацию вектора падающего света и вектора нормали: =α+β. (-;)=cosθL, (-;)= cosθT. (α+β;-)=cosθT, α(;-)+β(;-)=cosθT, α cosθL-β=cosθT, β = α cosθL- cosθT. Вычислим скалярное произведение единичного вектора преломления самого на себя: 2=(α+β)2= α2 ) +2 αβ() + β2() = 1, α2 + 2 α β()+ β2 =1, α2 - 2 α β cosθL + β2 =1. Подставим β: α2 - 2 α (αcosθL- cosθT) cosθL + (α cosθL- cosθT) 2 =1, α2 - (αcosθL- cosθT)(2α cosθL- cosθL + cosθT) 2 =1, α2 - (α2cos2θL- cos2θT)=1; Выразим угол преломления: η12(1 – cos2θL) = η22(1 – cos2θT), η2(1 – cos2θL) = 1 – cos2θT, cos2θT = 1 - η2(1 – cos2θL). Подставим в уравнение: α2 - (α2cos2θL - 1 + η2(1 – cos2θL))=1, α2 = η2, получаем α = η (второй корень α = -η не подходит по физическому смыслу). β= η cosθL - . Под корнем может быть отрицательное значение, это будет означать полное внутреннее отражение, т.е. отсутствие преломленного луча при переходе из оптически более плотной среды в оптически менее плотную среду.