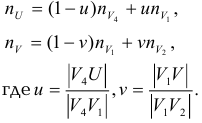
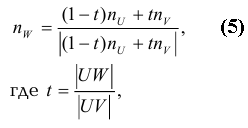
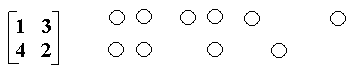
30. Интерполяционные алгоритмы закраски 3D полигональных объектов Гуро и Фонга.

Закраска методом Фонга здесь интерполируется не значение интенсивности по уже известным ее значениям в опорных точках, а значение вектора внешней нормали, которое затем используется для вычисления интенсивности пиксела. Метод Фонга заключается в построении для каждой точки вектора, играющего роль вектора внешней нормали, и использовании этого вектора для вычисления освещенности в рассматриваемой точке по формуле (5). При этом схема интерполяции, используемая при закраске Фонга, аналогична интерполяции в закраске Гуро. Для определения вектора "нормали" nw в точке W проводим через эту точку горизонтальную прямую и, используя значения векторов "нормалей" nU и nv в точках ее пересечения U и V с ребрами грани, получаем

а векторы внешних нормалей в точках U и V находятся, в свою очередь (также линейной интерполяцией), по векторам нормалей в концевых точках соответствующих ребер рассматриваемой многоугольной грани: Нормирование вектора nw необходимо в следствие того, что в формулах (1))(5) используется единичный вектор нормали.Достоинством предложенных моделей закраски (Гуро и Фонга) является их сравнительная простота. Однако вследствие значительных упрощений получаемый результат не всегда оказывается удовлетворительным. Преодолевать этот барьер качества лучше всего путем использования более совершенных моделей и методов

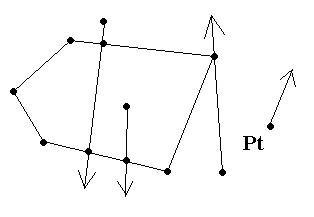
9. Цвет в машинной графике. Аппроксимация полутонами.

Для формирования тонов графики на устройствах одного цвета используют метод нарастающей интенсивности

Успех метода полутонов зависит от свойства зрительной системы человека быть интегратором, т. е. объединять или сглаживать дискретную информацию. 

Для каждой клетки используется четыре пиксела. При такой организации получается пять возможных уровней или тонов серого . В общем случае для двухуровнего дисплея число возможных интенсивностей на единицу больше числа пикселов в клетке. Если интенсивность изображения превышает некоторую пороговую величину, то пиксел считается белым, в противном случае он черный.

10. Тесты принадлежности точки к многоугольнику.

Pt вне многоугольника? Из Pt проводим вектор в х\з куда.

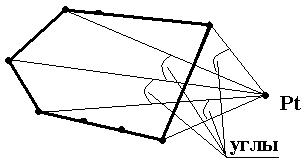
Если чето пересекли и четное пересечение – то вне контура.

Если не пересекали - то вне контура.

Если через точку (угол) – то Вылет.

Если нечетное количество пересечений – то внутри.

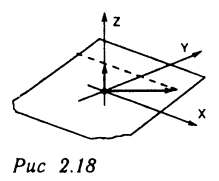
Еще один способ:

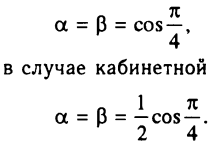
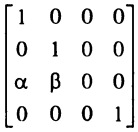
Точку Pt надо соединить с вершинами фигуры. Сумма углов равна 360, и значит Pt внутри фигуры, а иначе точка вне фигуры.

Еще как вариант – считать по площадям.

17. Косоугольные пpекции (военная и кабинетная перспектива) версия 1.

Наконец, косоугольная проекция (пучок прямых не перпендикулярен плоскости экрана). Выделяют два вида косоугольных проекций: свободную проекцию (угол наклона проектирующих прямых к плоскости экрана равен половине прямого) и кабинетную проекцию (частный случай свободной проекции; масштаб по третьей оси вдвое меньше).

При косоугольном проектировании орта оси Z на плоскость XOY (рис. 2.18) имеем (0 0 1 1)🡪(а β 0 1). Матрица соответствующего преобразования имеет следующий вид:

В случае свободной проекции

x' = x + z \* (L \* Math.Cos(alpha))

y' = y + z \* (L \* Math.Sin(alpha))

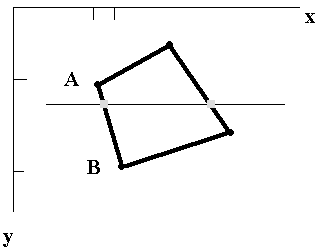
\* Военник=L=1

\* Кабинет=L=0.5

\* alpha=30\_OR\_45

13. Растровая развертка многоугольников.

значит смысл такой:

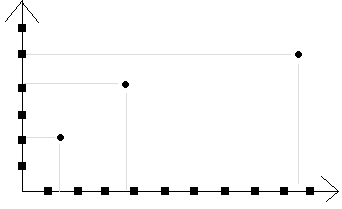
В функцию передаю координаты [x;y] начала и конца отрезков составляющих фигуру (у меня трапеция), прохожу в цикле по Y, в каждой строке по X нахожу пересечения (если есть) с линиями (через уравнение прямой), если нашел пересечение - заношу в стек, потом закрашиваю линию между пересечениями.

6. Окна в машинной графике (удаление, стирание, отсечение).

Назначения окна: 1)Выделение (цвет) 2)Отсечение (вне окна) 3)Удаление (в окне)

Обычно для реализации основных функций для работы с окнами требуется графический пакет, поддерживающий работу с областями сложной формы. С каждым окном связываются две такие области - область отсечения, представляющая собой видимую часть окна, и область, требующая перерисовки. Менеджер окон сам определяет окна, у которых область, требующая перерисовки, не пуста, и автоматически генерирует для таких окон запрос на перерисовку соответствующей области. В случае, когда мы работаем только прямоугольными окнами, все области, возникающие при выполнении над окнами основных операций, являются объединением нескольких прямоугольников, так что для простейшей реализации оконного интерфейса достаточно иметь графическую библиотеку с возможностью отсечения только по прямоугольным областям. В случае, когда область состоит из нескольких прямоугольников, каждый из них по очереди становится областью отсечения и для него выполняется функция перерисовки соответствующего окна.

1. Преобразование отрезков из векторной формы в pастpовую.

y = k \* x + b (уравнение прямой)

Делаем шаг по X, из вещественного числа округляем до целочисленого.

Недостатки формулы: 1)Вещественые числа 2)Операция умножения 3)Округление.

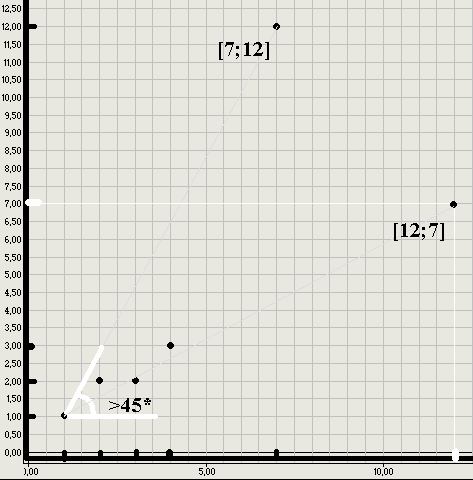
Простой пошаговый алгоритм:

Для X шаг = 1, для Y шаг = Y+dY,

где dY = (Y2-Y1)/(X2-X1).

Недостатки: 1)Вещественые числа 2)Округление

2. Алгоритм Бpезенхэма для построения отрезков прямых.

Алгоритм сводится к задаче построения линии по двум заданным точкам.

Разберем пример:

Даны начальные координаты точек [1;1] и [7;12].

Находим dx = 7 – 1 = 6 и dy = 12 – 1 = 11.

if( dx < dy ), то угол больше 45\*;

Меняем координаты местами

Находим dx = 12 – 1 = 11 и dy = 7 – 1 = 6.

(dx >= dy) следовательно d1 = 2\*dy – dx = 2\*6 – 11 = 1 > 0;

if(di >= 0) {

x2 = x1 + 1 = 2;

y2 = y1 + 1 = 2;

d2 = d1 + 2\*(dy - dx) = 1 + (-10) = -9 < 0;

};

if(d2 < 0) {

x3 = x2 + 1 = 3;

y3 = y2 = 2;

d3 = d2 + 2\*dy = -9 + 12 = 3 > 0;

}

x4 = x3 + 1 = 4;

y4 = y3 + 1 = 3;

и так до конечной точки.

Решающим являются два уравнения:

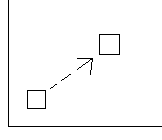
di + 2\*dy – dx, di = плюс.

di + 2\*dy, di = минус.

4. Геометрические двумерные пpеобpазования.ТУТ ЖЕ И

5. Однородные координаты и матричные представления двумерных преобразований.

1)Перемещение:

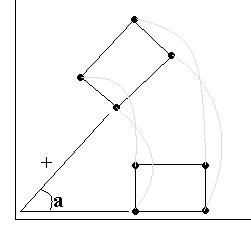
X’ = X + dX

Y’ = Y + dY, где dX и dY – значения сдвига точки.

2)Масштабирование:

X’ = X \* SX

Y’ = Y \* SY, где S – это множительный коэффициент масштабирования.

3)Повороты:

X’ = X \* Cos(a) – Y \* Sin(a)

Y’ = X \* Sin(a) – Y \* Cos(a)

В векторной форме преобразования имеют вид:

1)Перемещение:

[x’;y’]=[x;y]+[dx;dy]

p’ = p+T(dx;dy)

2)Масштабирование:

[x’;y’]=[x;y]\*

p’ = p\*S(Sx;Sy)

3)Поворот:

[x’;y’]=[x;y]\*

p’ = p\*R(a)

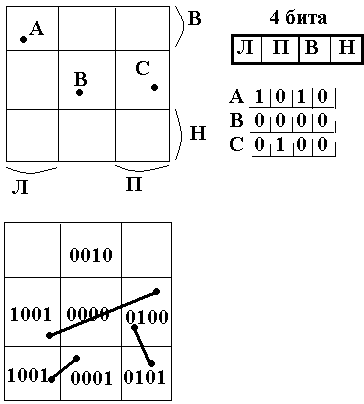
Проблема в том что в разных преобразованиях есть разные арифметические действия. Для унификации преобразований действия выполняют в 3D плоскости с Z=1.

1)Перемещение: 2)Mасштабирование: 3)Поворот:

[x’;y’;z’]=[x;y;1]\* [x’;y’;z’]=[x;y;1]\* [x’;y’;z’]=[x;y;1]\*

7. Метод Джона Коуэна для работы с окном.

Это модификация алгоритма работы с окнами от Дж.Коуэна и Анд.Сэзерленда.

Вводится понятие «паспорта» - начальные и конечные точки отрезка прямых.

Если оба паспорта отрезка имеют ноль-значения, то отрезок в окне.

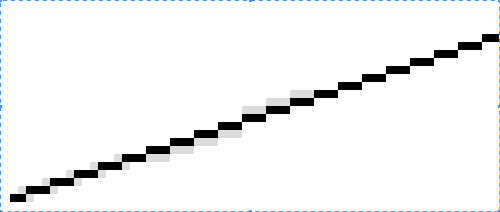
При логическом перемножении двух паспортов, если итог не равен нулю, то отрезок вне окна. Иначе возможно отрезок пересекается окном.

Все отрезки прямых, паспорт которых после умножения дает ноль, обрабатывают алгоритмом отсечения окон (типо вопрос 6).

Данный метод эфективен при большом/малом окне и насыщеном изображении.

3. Выравнивание отрезков прямых. Линии постоянной яркости.

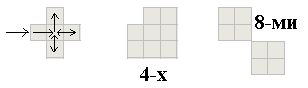
Скорее всего имеются ввиду дефекты, которые возникают при построениях линии алгоритмом Брезенхема, дефект ступенчатости. Для сглаживания таких участков используют дополнительные поддерживающие пиксеылы или целые отрезки прямых с цветом, в несколько порядков светлее того отрезка, для которого рисуем поддержку.

11. Заполнение областей. Типы областей. ТУДА ЖЕ И

12. Простые pекуpсивные алгоритмы заполнения областей.

Область – примыкающие друг к другу пикселы. Есть 1)Внутрене определеные и 2)Гранично заданые области.

Есть четырёх и восьми связаные области

(типо по количеству направлений).

Простой рекурсивный алгоритм заполнения областей:

///x, y - с какой точки начинать заполнение

///clr\_old, clr\_new - старый и новый цвета точки

void Fill\_4**(**x**,** y**,** clr\_old**,** clr\_new**)**

**{**

**if(**read\_pxl**(**x**,** y**)** **==** clr\_old**)**

**{**

write\_pxl**(**x**,** y**,** clr\_new**);**

//пытаемся пройти по 4-м направлениям:

Fill\_4**(**x**+**1**,** y**,** clr\_old**,** clr\_new**);**//вправо

Fill\_4**(**x**-**1**,** y**,** clr\_old**,** clr\_new**);**//влево

Fill\_4**(**x**,** y**+**1**,** clr\_old**,** clr\_new**);**//вниз

Fill\_4**(**x**,** y**-**1**,** clr\_old**,** clr\_new**);**//вврех

**}**

**}**

Минусы – большие методы заполнения невозможны.

Плюсы – для мылых областей самое то.

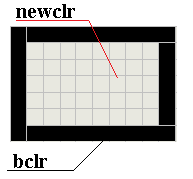
Для 8-ми связанных областей вводят дополнительный направления:

x+1, y+1

x+1, y-1

x-1, y+1

x-1, y-1

Рекурсивный алгоритм заполнения гранично-заданных областей:

///x, y - с какой точки начинать заполнение

///bclr, newclr - граничный и внутрений цвета

void FB\_4**(**x**,** y**,** bclr**,** newclr**)**

**{**

**if(**read\_pxl**(**x**,** y**)** **!=** bclr **&&** read\_pxl**(**x**,** y**)** **!=** newclr**)**

**{**

write\_pxl**(**x**,** y**,** newclr**);**

//пытаемся пройти по 4-м направлениям:

FB\_4**(**x**+**1**,** y**,** bclr**,** newclr**);**//вправо

FB\_4**(**x**-**1**,** y**,** bclr**,** newclr**);**//влево

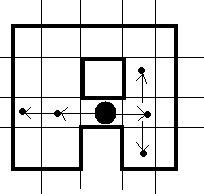
FB\_4**(**x**,** y**+**1**,** bclr**,** newclr**);**//вниз

FB\_4**(**x**,** y**-**1**,** bclr**,** newclr**);**//вврех

**}**

**}**

Алгоритм заполнения области полосами:

1)По заданной точке заполним полосу в лево и в право до границ.

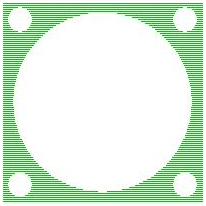
2)Берем выше полосу. Ищем правые точки начала полосы, пишем точку в стек.

3)Берем ниже полосу. Ищем правые точки начала полосы, пишем точку в стек.

4)Берем последнее значение из стека (типо стартовая точа) и переходим к пункту 1.

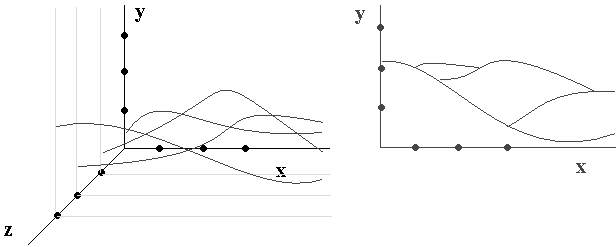
5)Если стек пуст, то область заполнениа.

Плюсы: быстрый, для любой области, малый стек.

Так же завязан на рекурсии.

26. Алгоритм плавающего горизонта для удаления невидимых участков поверхности.

Задана трех мерная функция F(x,y,z) = 0, её перегоняем в y=f(x,z) при z=const.

Суть сводится к перегону в xy-плоскость обьемной (xyz) картинки-графика.

Берем два масива – верхний\_горизонт(размерность) и нижний\_горизонт(размерность), размерность = ширинаЭкранаПК.

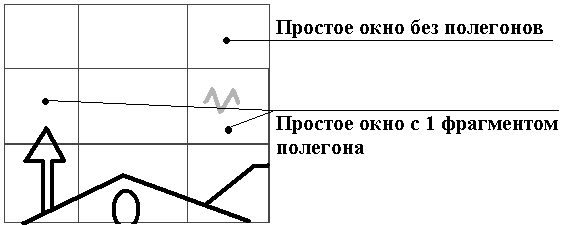
Двигаемся от Zначальн. до Zконечн. с шагом h по Z. Для фиксированого Z строим график функции y=f(x).

Если на текущей Z-плоскости при некотором X соответствующее значение Y на графике БОЛЬШЕ значения Y для всех предыдущих построенных кривых при этомже X, то текущая кривая видна в этой точке.

Иначе – невидна. Также анализируется и нижний горизонт – это min значения для Y.

В начале рекомендуют первую кривую графика вносить в оба массива горизонтов.

23. Удаление скрытых поверхностей алгоритмом Варнока.

Че глазеть на всю картинку(история про то как чел брал свернутую в трубку газетку и глядел через неё на картинку), чтоб ее обработать в разы быстрее разбиваем участки на маленькие области – простые окна.

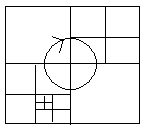
1)Когда все полегоны внешние – то окно пишем в растр, там нечего разбивать. 2)Когда 1 полегон внутри окна или 3)когда 1 окно охвачено полегоном (если за полегоном есть еще полегон, то они не рассматриваются). Остальные вариации окон не рассматриваются, их делят и получают более простые окна (3 вида выше).

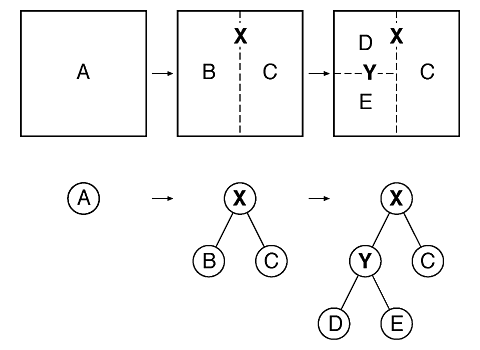
Рисунок разделяют на 4 части, по часовой стрелке анализируют. Если окно простое, то выводят в растр, если окно сложное – то опять это сложное окно дробят на 4 части и рассматриваем их на простоту. Глубина предела дробления окна для получения простого = 1 пиксел(дальше не прыгнешь).

25. Удаление скрытых поверхностей алгоритмом BSP-дерева.

BSP – binary space partition, используемое для сортировки и поиска многоугольников и многогранников в n-мерном пространстве. Взятое в целом BSP-дерево представляет собой все пространство, а каждый его узел ограничивает выпуклое подпространство. Каждый из узлов дерева хранит информацию о "гиперплоскости", делящей пространство узла на 2 части (переднюю и заднюю, что определяется направлением вектора нормали этой плоскости), а также ссылки на два новых узла, представляющих эти части. Заодно в узлах может храниться информация об одном или нескольких полигонах, лежащих в этой "гиперплоскости". Обычно BSP-деревья используются для представления двух- и трехмерных пространств, однако по определению эти структуры не ограничены тремя измерениями.

Имеет ряд требований – 1)Дерево должно быть симметричным 2)При построении дерева минимизируем разбитие полегонов плоскостями других полегонов 3)Процесс работы с деревом ассимитричен, много времени уходит на построение дерева.

Несложно понять BSP-дерево, если ограничится двумя измерениями. Еще проще нам будет, если предположим, что используются только линии, параллельные осям X и Y, и что будем делить пространство на равные части для каждого узла.

Такой пример: есть квадрат ~~затеряный~~ где-то в квадрате ~~альфа-центавра~~ плоскости XY.  
1.Производим первый раздел (который будет узлом нашего дерева). Разобьем квадрат пополам в направлении оси X.  
2. Для каждого рассечения мы будем изменять направление линии раздела на противоположное, т.о. второй "разрез" будет произведен над оставшимися секторами в направлении оси Y.  
3. Этот процесс можно продолжать рекурсивно, пока не будет достигнута "точка останова".  
Результат можно изобразить таким образом:

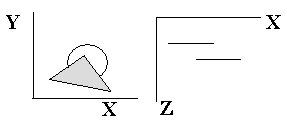
Логично предположить, что имея построенное дерево с плоскостями, а их глубина зависит от фантазии прораба и наличия хорошего трактора, удаление невидимых или ненужных полигонов для каждой отдельной сцены сводится к следующему: находясь в комнате Е, вы еще и находитесь в определенном узле дерева, и нам ненужно показывать комнаты D и C, их мы и неувидим ведь в дереве только направление к корню Y.

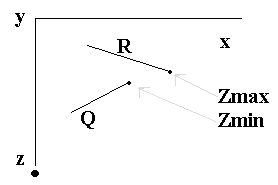
Определение положения точки (где находится персонаж игры) относительно плоскости производится простой подстановкой координат точки (x,y,z) в уравнение плоскости Ax+By+Cz+D = 0. Результатом этой операции будет расстояние от точки до плоскости. **Оно будет положительным**, если точка находится по ту сторону плоскости, в которую указывает вектор нормали **и отрицательным**, если по другую. Если же результат равен 0, то точка лежит на плоскости. *Если вы ноль в математике и "уравнение плоскости" для вас - непроходимый лес, я поясню, что A, B, и C - это координаты вектора нормали, а D может быть заранее рассчитано подстановкой точки с координатами (x,y,z), лежащей в этой плоскости.*  
Вообще выпуклые полигоны гораздо проще делить чем невыпуклые, так как первые всегда делятся на две выпуклых же части, а с последними иметь дело очень сложно, особенно если они самопересекающиеся.

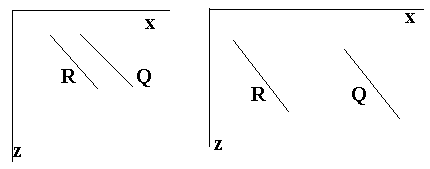
При построении BSP-дерева для удаления невидимых поверхностей делящие плоскости почти всегда выбираются из входящих в сцену полигонов. Однако, любая плоскость может быть использована как делящая, что наверняка приведет к положительному скачку в производительности генератора BSP-дерева и программы в целом. Если камера находится на делящей плоскости, то теоретически порядок прорисовки ветвей неясен. В этом случае мы не выводим на экран полигоны, принадлежащие плоскости, потому что их все равно не будет видно.

24. Удаление скрытых поверхностей алгоритмом соpтиpовки по глубине.

На примере алгоритма Художника. Сначало берется дальний рисунок, потом на него накладывается еще полегоны в порядке возрастания Z-глубины. Ну и сортируем кто как глубже расположен.

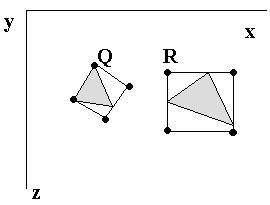
Есть ситуации когда становятся видимы те участки, которые должны быть скрыты. Для этой ситуации используют список приоритетов.

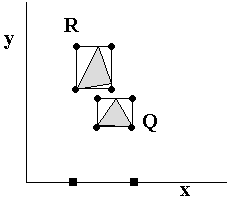
R – первый полигон в списке, Q – второй полигон в списке. Если то R не затеняет Q и R вывести в растр. После этого полегон Q становится R, а следующий полегон Q и опять повтор сравнений.

Иначе R возможно затеняет Q.

Для определения незатеняемости используют 5 тестов. Если хоть на один тест есть положительный результ, то R не затеняет Q.

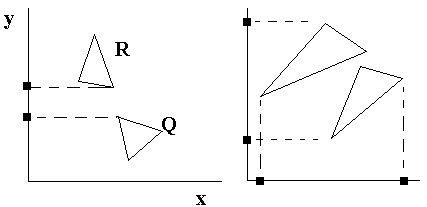
Тест 1.

Верно ли что прямоугольные оболочки полигонов R и Q не пересекаются по X, если да – то R выводят в растр.

Когда первый тест пройден положительно, начинают второй тест.

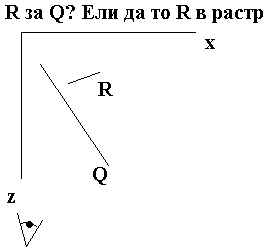
Тест 2.

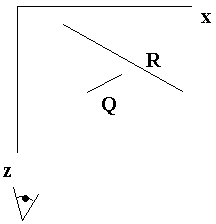
Верно ли что прямоугольные оболочки R и Q не пересекаются по оси Y?



Тест 3.

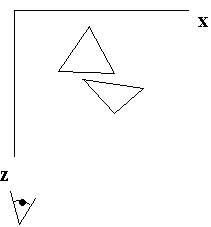
Верно ли что R целиком лежит по ту сторону плоскости несущей Q, которая расположена дальше от точки наблюдения

Тест 4

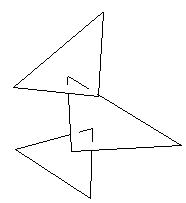
Верно ли что полигон Q целиком лежит по ту сторону плоскости, несущей полегон R котор. ближе к наблюдателю.

Тест 5:

Верно ли что проекции R и Q полегонов не пересекаются

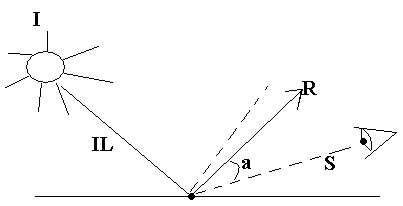


Если все пять тестов провалены, то R и Q меняют местами.

Если опять нет результатов, то полегон R делится плоскостью несущ. Q на 2 полегона и опять заново сортируем.

28. Модель зеркального отражения (Фонга).

Есть три вида освещения – зеркальное, диффузное, преломление.

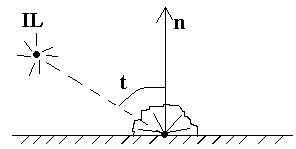


n – коэфф пространственного распределения зеркального отражения луча.

n = 1…200

Это модель Фонга.

29. Модель диффузного отражения (Ламберта).



Это модель Ламберта.

Если обработать 3D сцену моделью Ламберта то изображение будет напоминать фото сделаное со вспышкой. Все то что не освещается это будет черным цветом. Для устранения этого недостатка используют рассеивание:

Для предания фотографии реалистичности модель Фонга обьединяют с моделью рассеивания:

Еще добавим расстояние от камеры до освещаемой точки (S).

20) Матричное представление трехмерных преобразований с помощью однородных координат

Трехмерная точка (x,y,z) записывается в однородных координатах как, для получения трехмерных декартовых координат точки (x,y,z) первые три однородные координаты делятся на W.

Трехмерный **перенос** является простым расширением двумерного:

T(dx,dy,dz) = , т.е. [xyz1]T(dx,dy,dz) = [x+dxy+dyz+dz 1].

**Масштабирование** расширяется аналогичным образом:

S(Sx,Sy,Sz) = , [xyz1]S(Sx,Sy,Sz) = [Sxx Syy Szz 1].

В трехмерном пространстве **поворот** **вокруг оси z** описывается выражением (напомним, это выражение для правосторонней системы, для левостронней знаки синусов поменялись бы местами):

Rz(A) = .

Матрица **поворота вокруг оси x** имеет вид

Rx(A) = .

Матрица **поворота вокруг оси у** записывается в виде

Ry(A) = .

Столбцы (и строки) верхней левой подматрицы размером 3х3 матриц Rz(A), Rx(A), Ry(A) представляют собой взаимно ортогональные единичные векторы.

Все эти матрицы преобразований имеют обратные матрицы. Матрица, обратная Т, получается подстановкой знака минус перед dx, dy и dz; обратная S – заменой Sx,Sy и Sz на обратные им значения, а для каждой из трех матриц поворота – выбором отрицательного угла поворота.

21) Пример композиции трехмерных преобразований

Путем **объединения** элементарных трехмерных преобразований можно получить другие преобразования.



Начальная позиция Конечная позиция

Рис. 11. Преобразование точек Р1, Р2 и Р3 из начальной позиции в конечную

**Шаг 1. Перенос Р1 в начало координат**:

Т(-x1, -y1, -z1) = 

Применение Т к Р1, Р2 и Р3 дает:

Р1' = P1T(-x1, -y1, -z1) = [0 0 0 1],

Р2' = P2T(-x1, -y1, -z1) = [x2-x1y2-y1z2-z1 1],

Р3' = P3T(-x1, -y1, -z1) = [x3-x1 y3-y1 z3-z1 1].

**Шаг 2. Поворот вокруг оси у.**

На рис. 12 показаны отрезки  и  после шага 1 и проекция  на плоскость xz.



Рис. 12. Поворот вокруг оси у; проекция  поворачивается до совмещения с отрицательной полуосью z

Поворот производится на положительный угол А, для которого

,

,

где .

В результате подстановки этих значений в матрицу Ry(A) получаем:

.

**Шаг 3. Поворот вокруг оси х.**

На рис. 13 показан отрезок  после шага 2.



Рис. 13.

Поворот производится на отрицательный угол В, для которого

cos(-B) = cosB = ,

sin(-B) = -sinB = ,

где . (длина отрезка ).

Результатом поворота на шаге 3 является:

P2''' = P2''Rx(B) = P2'Ry(A)Rx(B) = P2TRy(A)Rx(B) = ,

т.е.  теперь совпадает с отрицательной полуосью z.

**Шаг 4. Поворот вокруг оси z.**

На рис. 14 показаны  и  после шага 3, когда Р2'' лежит на отрицательной полуоси z, а Р3''' - в точке

P3''' = [x3''' y3''' z3''' 1] = P3T(-x1, -y1, -z1)Ry(A)Rx(B).



Рис. 14. Поворот вокруг оси z; проекция  поворачивается до совмещения с осью у

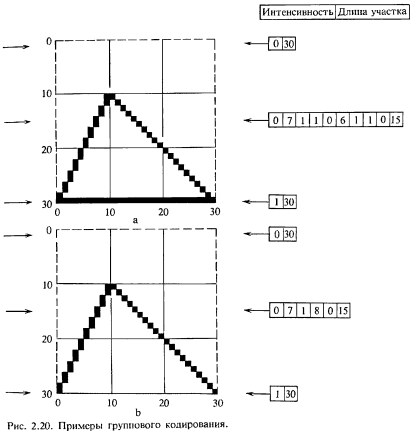
Поворот производится на положительный угол С, для которого cosC = y3'''/D2, sinC = x3'''/D2, D2 = .

После шага 4 получается конечный результат.

Результирующая матрица T(-x1,-y1,-z1)Ry(A)Rx(B)Rz(C) = TR описывает искомое преобразование.

8. Способы хранения pастpовой информации (групповое кодирование).

В методе группового кодирования сделана попытка воспользоваться тем, что большие области изображения имеют одинаковую интенсивность или цвет. При простейшем групповом кодировании определяется только интенсивность и количество последовательных пикселов с этой интенсивностью на данной сканирующей строке.

На рис. 2.20,а показан простой черно-белый чертеж на растре 30 х 30 и соответствующие кодирующие последовательности для сканирующих строк с номерами 1, 15 и 30. Кодирующие данные следует рассматривать группами по два. Первое число — интенсив- интенсивность, второе — число последовательных пикселов на сканирующей строке с этой интенсивностью:



Таким образом, в строке 1 на рис. 2.20,а имеется 30 пикселов нулевой интенсивности, т. е. черных или фоновых. Все изображение можно закодировать с помощью 208 чисел. Попиксельное хранение, т. е. одна порция информации на каждый пиксел (битовая карта), потребовало бы 900 значений интенсивности для растра 30 х 30. В этом случае сжатие данных с помощью группового кодирования составляет 4.33 : 1.

Легко обрабатываются с помощью данного метода сплошные фигуры, как это продемонстрировано на рис. 2.20,b с кодированием 1, 15 и 30 строк. Особый интерес представляет 15-я сканирующая строка. Все изображенные на рис. 2.20,b может быть закодировано с использованием 136 чисел при сжатии данных 6.62 : 1.

Большая степень сжатия изображений со сплошными фигурами по сравнению с сеточными рисунками объясняется тем, что два ребра покрываются одной парой «интенсивность — длина».

14. Текстура. Фильтрация текстур.

Текстура представляет собой двумерное растровое изображение, которое накладывается (натягивается) на поверхность объекта, например на плоский треугольник.Текстуры, как правило, хранятся в графических файлах форматов bmp, jpeg, tiff, tga, gif

Фильтрация текстур – это механизм, с помощью которого происходит наложение текстуры на полигоны отличающегося размера. Наиболее распространенными по использованию являются следующие типы фильтрации текстур:

1)точечная фильтрация (используется по умолчанию) – самая быстрая по скорости, но самая низкая по качеству;

2)линейная фильтрация – приемлемое качество и скорость;

3)анизотропная – самая медленная, но самая качественная.

**Билинейная фильтрация** — процесс извлечения нескольких пикселей исходной текстуры с последующим усреднением их значений для получения окончательного значения пикселя. Понятие «**билинейная фильтрация**», точно так же, как и сходное понятие «**трилинейная фильтрация**», применимо только к двумерным текстурам. Для трехмерных, например, данное понятие неприменимо, а понятие трилинейной фильтрации имеет совершенно другое значение.

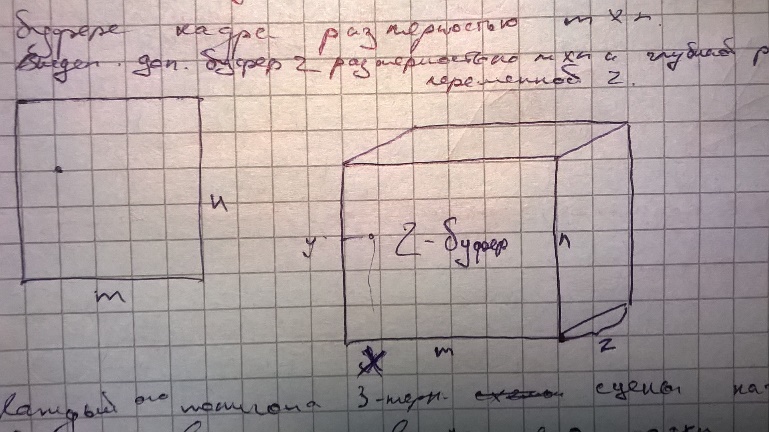
**Mip-mapping** - наложение текстур, имеющих разную степень или уровень детализации, когда в зависимости от расстояния до точки наблюдения выбирается текстура с необходимой детализацией.

В изображении, связанном с уровнем mip-map, пиксель представляется в виде среднего четырех пикселей из предыдущего уровня с более высоким разрешением. Отсюда, изображение связанное с каждым уровнем mip-текстуры в четыре раза меньше по размеру предыдущего mip-map уровня.

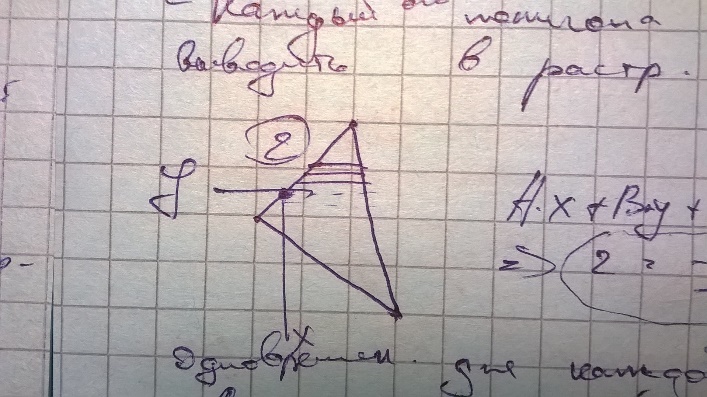
22.Удаление скрытых поверхностей алгоритмом, использующего Z-буфеp.

Пусть требуется построить изображение в буфере кадра размерностью m\*n.

Выделим дополнительный буфер Z размерности m\*n и глубиной равной типу переменной Z.



С каждого полигона 3-мерной сцены начинаем выводить в растр внутренние его точки.

 Если используется построчный алгоритм заливки, то легко сделать пошаговое вычисление Z-координаты очередного пиксела, дополнительно храня Z-координаты его вершин и вычисляя приращение dz Z-координаты при перемещении вдоль X на dx, равное 1. Если известно уравнение плоскости, в которой лежит обрабатываемый многоугольник, то можно обойтись без хранения Z-координат вершин. Пусть уравнение плоскости имеет вид:

Ax+By+Cz+D = 0.

Тогда при C не равном нулю Z = - (Ax+By+D)/C.

Найдем приращение Z-координаты пиксела при шаге по X на dx, помня, что Y очередной обрабатываемой строки - константа.

dZ= - (A(x+dx)+D)/C+(Ax+D)= - AdX/C.

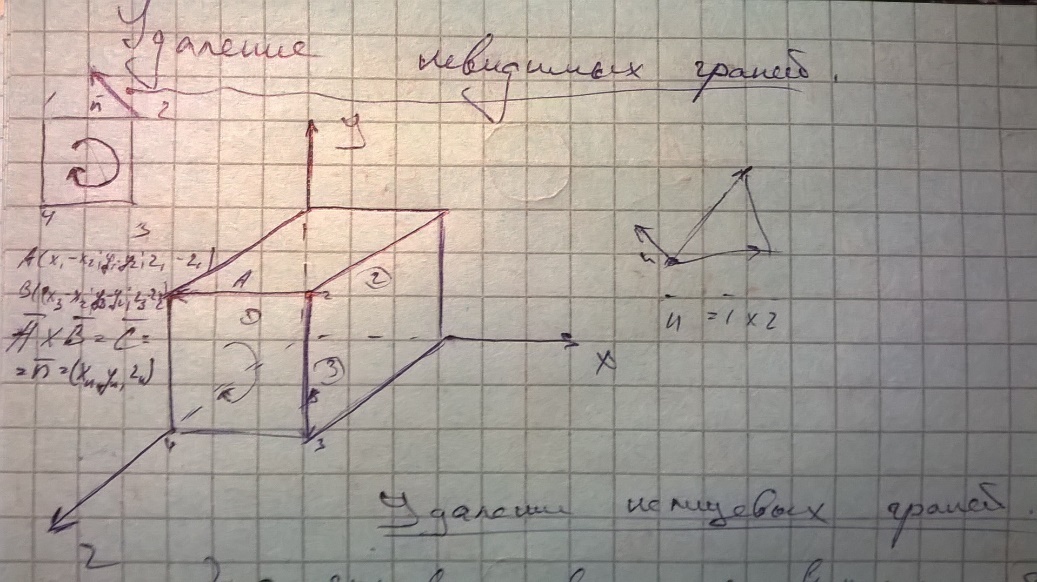
но dx = 1, поэтомуdZ= - A/C.

Одновременно для каждой внутренней точки полигона вычисляем глубину Z. Если вычисленная глубина Z по координате x,y больше чем значение z по индексам в массиве z-буфера, то точка видна и мы выводим её в растр, а значение Z заносим в z-буфер.

В начале работы алгоритма в z-буфер заносим min возможные значения.

+:1). Простота в реализации 2). Можно обработать любую по сложности 3-мерную сцену.

-: 1). Хранение большого z-буфера.

27.Удаление нелицевых граней выпуклых 3D объектов.

Данный метод справедлив для выпуклых объектов, если объект невыпуклый, разбиваем его на таковые.

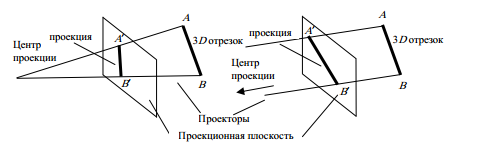
*Удаление нелицевый граней-обязательный тест*(желательный) перед работой основного алгоритма удаления невидимых граней.

Каждый полигон 3D объекта записывается своими вершинами по часовой стрелки (или против часовой). Для этого мы должны всегда смотреть на полигон с лицевой стороны. Выбираем любую вершину в полигоне и рассматриваем векторное произведение векторов, выходящих из этой вершины. Под векторами будем понимать ребра полигона. В результате векторного произведения получится нормаль.

Если параметр Z нормали отрицателен, то данный полигон невидим (нелицевой) – его рисовать не надо. В противном случаи, полигон выводится в растр.

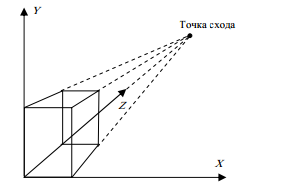
16. Проекции тpехмеpных объектов (центральная и параллельная).

Проекции строятся с помощью проецирующих лучей или проекторов, которые выходят из точки, называемой центром проекции. Проекторы проходят через плоскость проекционную или картинную плоскость и затем проходят через каждую точку трёхмерного объекта и образуют тем самым проекцию.

Геометрические проекции делятся на два вида: центральные и параллельные. Если центр проекции находится на конечном расстоянии от проекционной плоскости, то проекция – центральная. Если же центр проекции находится в бесконечности, то проекция – параллельная (рис. 5.1).

**Рис. 5.1. Центральная и параллельная проекции**

Точкой схода называется точка пересечения центральных проекций любой совокупности параллельный прямых, которые не параллельны проекционной плоскости. Существует бесконечное множество точек схода. Точка схода называется главной, если совокупность прямых параллельна одной из координатных осей. В зависимости от того, сколько координатных осей пересекает проекционную плоскость различают одно-, двух- и трёхточечные проекции (рис. 5.2).

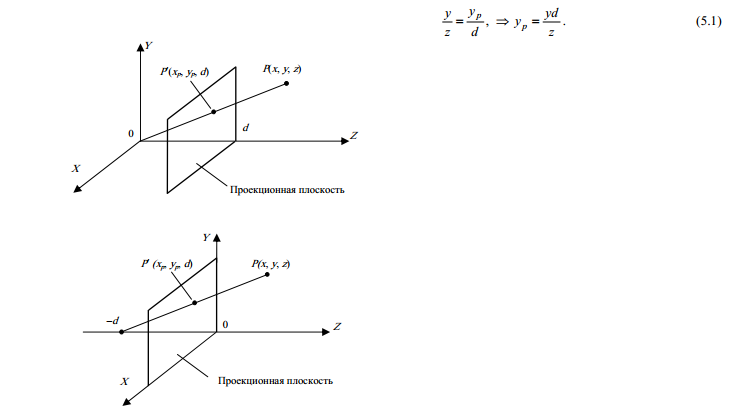
Простейшей является параллельная прямоугольная проекция. В ней совместно изображаются виды сверху, спереди и сбоку. Эти проекции часто используются в черчении. В зависимости от соотношения между направлениями проецирования и нормалью к проекционной плоскости параллельные проекции разделяются на ортографические или ортогональные, в которых эти направления совпадают, и косоугольные, в которых они не совпадают. В зависимости от положения осей системы координат объекта относительно проекционной плоскости ортографические проекции делятся на аксонометрические и изометрические. В изометрических проекциях оси системы координат составляют одинаковые углы с проекционной плоскостью. В аксонометрических проекциях эти углы разные. Центральная проекция приводит к визуальному эффекту, подобному тому, который даёт зрительная система человека. При этом наблюдается эффект перспективного укорачивания, когда размер проекции объекта изменяется обратно пропорционально расстоянию от центра проекции до объекта. В параллельных проекциях отсутствует перспективное укорачивание, за счёт чего изображение получается менее реалистичным и параллельные прямые всегда остаются параллельными.

**Рис. 5.2. Одноточечная проекция**

В компьютерной графике различают две модели центральной проекции:

− проекционная плоскость перпендикулярна оси Z и совпадает с плоскостью Z = d (рис. 5.3, вариант 1);

− проекционная плоскость перпендикулярна оси Z и совпадает с плоскостью Z = 0 (рис. 5.3, вариант 2).

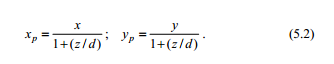
В первом случае 3D точка P проецируется на экран как P′. Расстояние от наблюдателя до проекционной плоскости равно d. Необходимо определить координаты точки P′ на экране. Обозначим их x′ и y′. Из подобия треугольников 0 PyPz0 и ypdO находим, что

**Рис. 5.3. Центральная проекция: варианты 1 и 2**

Аналогично для x: 5,31.png

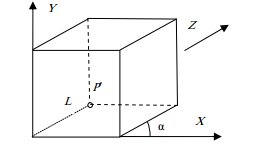
Напомним, что d – это расстояние наблюдателя до проекционной плоскости, а наблюдатель находится в точке (0, 0, 0).

Если поместить проекционную плоскость в начало координат, а точку наблюдения на расстояние –d, как показано на рис. 5.3 (вариант 2), то формулы для Хp и Уp примут вид:

Формулы (5.2) более удобны при необходимости

простым образом приближать или удалять наблюдателя от проекционной плоскости. Формулы (5.1) требуют меньше времени для вычислений за счёт отсутствия операции сложения.

17. Косоугольные пpоекции (военная и кабинетная перспектива) версия 2.

Рис. 5.4. Косоугольная параллельная проекция единичного куба

Для косоугольных проекций (рис. 5.4) проекторы пресекают проекционную плоскость под непрямым углом. Для военной косоугольной проекции этот угол составляет 45 градусов, а для кабинетной – 63,4 градуса. Математические соотношения формул косоугольной проекции произвольного 3D объекта базируются на формулах косоугольной проекции единичного куба (ребро длиной 1).

Из рисунка 5.4 видно, что проекцией точки P(0, 0, 1), находящейся на задней стороне единичного куба, является точка

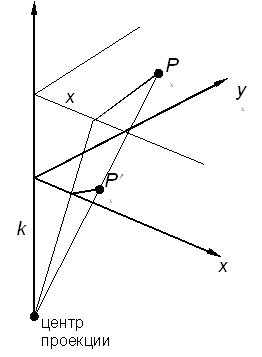
P′(L cos(α), Lsin(α)), принадлежащая, плоскости XY.

Математические соотношения для косоугольной параллельной проекции произвольного 3D объекта имеют вид

2015-01-13_222148.png

Для военной проекции L = 1, а для кабинетной проекции L = 0,5. Угол α обычно выбирают 30° или 45°. Не следует путать этот угол с углом между проекторами и проекционной плоскостью.

18. Математическое описание плоских геометрических проекций (центральная).

**Центральная** (перспективная) проекция получается путем перспективного преобразования и проецирования на некоторую двухмерную плоскость «наблюдения». Перспективная проекция на плоскость Z = 0 обеспечивается преобразованием

**Рис. 3.14. Вычисление одноточечной перспективы**

|  |  |  |
| --- | --- | --- |
| *[X Y Z H] = [x y z*1*]\** | http://compgraph.tpu.ru/matrixproections.files/image002.gif | = *[x y 0 (rz+*1*)].* |

или

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *x\** = | http://compgraph.tpu.ru/matrixproections.files/image006.gif | = | http://compgraph.tpu.ru/matrixproections.files/image008.gif | ; |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *y\** = | http://compgraph.tpu.ru/matrixproections.files/image010.gif | = | http://compgraph.tpu.ru/matrixproections.files/image012.gif | ; |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *z\** = | http://compgraph.tpu.ru/matrixproections.files/image014.gif | = | http://compgraph.tpu.ru/matrixproections.files/image016.gif | , |

где

 Центр проекции находится в точке с координатами *(*0,0*,-k)*(рис.3.14.), плоскость проецирования *Z = 0*. Соотношения между *x, y* и *x\*, y\** остаются теми же самыми. Рассматривая подобные треугольники, получим, что

аналогично

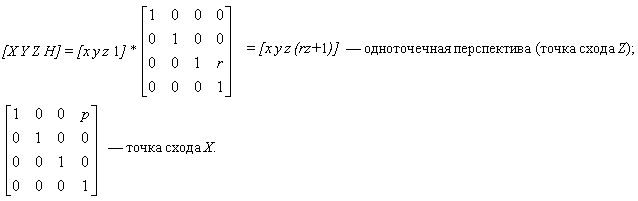
Координаты *x\*, y\** являются преобразованными координатами. В перспективном проектировании преобразованное пространство не является евклидовым, так как ортогональность осей не сохраняется. При *k*= ¥ получим аксонометрическое преобразование.

Аффинное преобразование есть комбинация линейных преобразований, сопровождаемых переносом.

Последний столбец в обобщенной матрице 4´4 должен быть равенhttp://compgraph.tpu.ru/matrixproections.files/image028.gif, в этом случае *H =*1.

Перспективному преобразованию может предшествовать произвольная последовательность аффинных преобразований. Таким образом, чтобы получить перспективные изображения из произвольной точки наблюдения вначале используют аффинные преобразования, позволяющие сформировать систему координат с осью *Z* вдоль желаемой линии визирования. Затем применяется перспективное преобразование.

Аналогично перспективное преобразование, когда картинная плоскость перпендикулярна оси *Z* и совпадает с плоскостью *Z* = 1/*r.*Центр проекции находится в центре координат:



19. Математическое описание плоских геометрических проекций (параллельная).

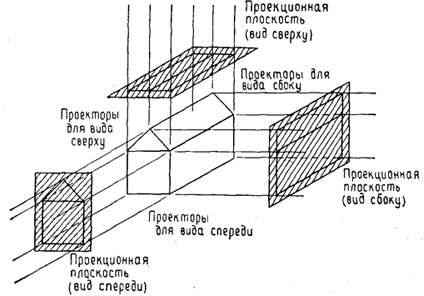
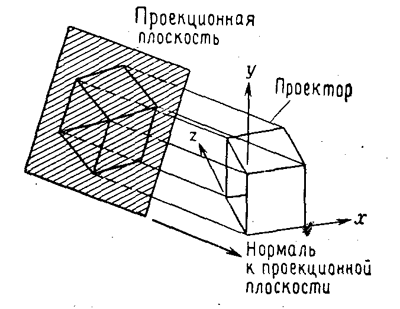
Параллельные проекции разделяются на два типа в зависимости от соотношения между направлением проецирования и нормалью к проекционной плоскости. В ортографических параллельных проекциях эти направления совпадают, а в косоугольных параллельных проекциях они не совпадают. То есть в ортографических проекциях направление проецирования является нормалью к проекционной плоскости.

Рис. 4.8. Построение трех ортографических проекций

Наиболее широко используемыми видами ортографических проекций являются вид спереди, вид сверху (план) и вид сбоку, в которых картинная плоскость перпендикулярна главным координатным осям, совпадающим вследствие этого с направлением проецирования. На рис. 4.8 показан процесс построения каждой из этих трех проекций, которые часто применяются в инженерной графике для описания машиностроительных деталей, агрегатов и сооружений, так как по ним можно измерять расстояния и углы. Поскольку каждая проекция отображает лишь одну сторону объекта, часто совсем непросто представить себе пространственную структуру проецируемого объекта, даже если рассматривать сразу несколько проекций одного и того же объекта.

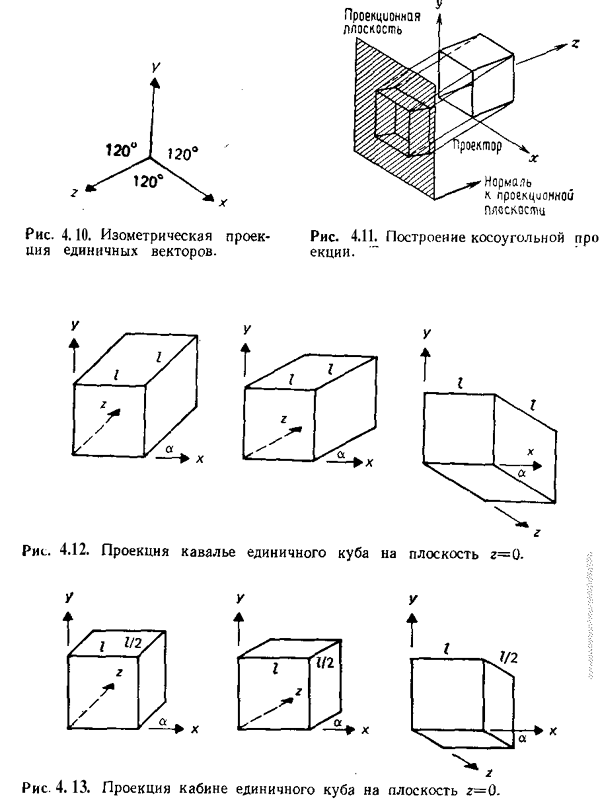
В случае **аксонометрических ортографических проекций** используются проекционные плоскости, не перпендикулярные главным координатным осям, поэтому на них изображаются сразу несколько сторон объекта, так же как и при центральном проецировании, однако в аксонометрии укорачивание постоянно, тогда как в случае центральной проекции оно связано с расстоянием от центра проекции. При аксонометрическом проецировании сохраняется параллельность прямых, а углы изменяются; расстояния же можно измерить вдоль каждой из главных координатных осей (в общем случае с различными масштабными коэффициентами).

Широко используемым видом *аксонометрической проекции является изометрическая проекция*. В этом случае нормаль к проекционной плоскости (а следовательно, и направление проецирования) составляет равные углы с каждой из главных координатных осей. Если нормаль к проекционной плоскости имеет координаты (а, b, с), то потребуем, чтобы |a|=|b|=|c|) или +/-a=+/-b=+/-c. Имеются ровно восемь направлений (по одному в каждом из октантов), которые удовлетворяют этому условию, однако существуют лишь четыре различные изометрические проекции (если не рассматривать удаление скрытых линий), поскольку векторы (а, а, а) и (-а, -а, -а) определяют нормали к одной и той же проекционной плоскости. Единичными нормалями этих направлений являются векторы (а, а, а), (-а, а, а), и (а, -а, а). На рис. 4.9 показан процесс построения изометрической проекции с направлением (1, -1, 1.)

Рис. 4.9. Построение изометрической проекции единичного куба

**Изометрическая проекция** обладает следующим свойством: все три главные координатные оси одинаково укорачиваются. Поэтому можно проводить измерения вдоль направления осей с одним и тем же масштабом (отсюда название: изо, что означает "равно", и метрия - "измерение"). Кроме того, главные координатные оси проецируются так, что их проекции составляют равные углы друг с другом (рис. 4.10).

**Косоугольные проекции** (второй тип параллельных проекций) сочетают в себе свойства ортографических проекций (видов спереди, сверху и сбоку) со свойствами аксонометрии. В этом случае проекционная плоскость перпендикулярна главной координатной оси, поэтому сторона объекта, параллельная этой плоскости, проецируется так, что можно измерять углы и расстояния. Проецирование других сторон объекта также допускает проведение линейных измерений (но не угловых) вдоль главных осей. Благодаря этим свойствам, а также простоте построения косоугольные проекции широко (хотя и не слишком) используются в этой книге. На рис. 4.11 показан процесс построения косоугольной проекции. Отметим, что нормаль к проекционной плоскости и направление проецирования не совпадают.

Двумя важными видами **косоугольных проекций** являются **проекции кавалье** (cavalier) (горизонтальная косоугольная изометрия) и **кабине** (cabinet)(фронтальная косоугольная диметрия).

**В проекции** кавалье направление проецирования составляет с плоскостью угол 45°. В результате проекция отрезка, перпендикулярного проекционной плоскости, имеет ту же длину, что и сам отрезок, т. е. укорачивание отсутствует. На рис. 4.12 приведено несколько проекций кавалье единичного куба на плоскость ху. Здесь уходящие вглубь линии являются проекциями тех ребер куба, которые перпендикулярны плоскости xу; они расположены под углом к к горизонтали. Этот угол обычно составляет 30 или 45°.

**Проекция кабине**, показанная на рис. 4.13, имеет направление проецирования, которое составляет с проекционной плоскостью угол агсс^ (1/2). При этом отрезки, перпендикулярные проекционной плоскости, после проецирования составляют 1/2 их действительной длины. Проекции кабине являются более реалистичными, чем проекции кавалье, поскольку укорачивание с коэффициентом 1/2 больше согласуется с нашим визуальным опытом.

15. Обработка 2D изображений (трафарет, сглаживание, выделение контуров, улучшение фокусировки).

Cглаживание вроде понятно, выделение контyров тоже. Да и про фокусировку можно что то сказать. Фокусировка-по сути улучшение четкоски изображения. А по трафарету мы курсач делали.

Вообще: сиди и сочиняй)

Буфер трафарета (stencil buffer, далее S-буфер) – многофункциональный буфер OpenGL - подобен буферам глубины и цвета, но, в отличие от них, его значения имеют специфичный для конкретного приложения смысл. Работу S-буфера определяют функция трафарета и операция трафарета. Функция задает условие прохождения теста трафарета, а операция – процедуру обновления значений в буфере в зависимости от результата этого теста. Примеры применения S-буфера – отсечение теней и зеркальных отражений, композиция изображений, вычисление глубинной сложности сцены и другие.