МИНИСТЕРСТВО НАУКИ И ВЫШЕГО ОБРАЗОВАНИЯ РФ
ФГБОУ ВО «ДАГЕСТАНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

КАФЕДРА «ДИЗАЙН»

УЧЕБНО-МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ по дисциплине «Технология графических преобразований» для студентов направления подготовки 09.03.03– ПИ, «Прикладная информатика в дизайне»

Махачкала – 2024

УДК 621.398.23(075.8)
ББК 32.811
Б78

Учебно-методические указания к выполнению лабораторных работ по дисциплине «Технология графических преобразований » для студентов направления подготовки 09.03.03– ПИ, «Прикладная информатика в дизайне» - Махачкала, ДГТУ, 2024. – __98__ с.

Для успешного решения задач по удовлетворению нужд потребителей, специалисту необходимо овладеть технологиями графических преобразований

Современный специалист должен знать, владеть и иметь представления об алгоритмах, методах и технологиях графических преобразований, умело использовать инструментальную базу современных пакетов, ориентированных в том числе и на языки программирования, для решения поставленных перед ним практических задач.

Учебно-методические указания состоят из краткого конспекта лекций и практических заданий, которые помогут закрепить полученные теоретические знания по дисциплине «Технология графических преобразований» предназначены для аудиторной и лабораторной работы студентов, а так же представляет интерес для тех, кто работает в сфере дизайна и искусства.

Составители:

© Парамазова А.Ш. заведующий кафедрой «Дизайн», ФГБОУ ВО «ДГТУ», член союза художников РФ, член ассоциации искусствоведов (АИС), заслуженный деятель искусств РД.

© Гаджиев А.М., кан.физ.-мат.наук, доцент кафедры «Дизайн» ФГБОУ ВО «ДГТУ»,

Рецензенты:

М.К. Курбанов ген. директор авторизованного учебного центра «Академия компьютерной графики»

А.З. Ахмедова кан.физ.-мат.наук, доцент кафедры ИТиБКС ФГБОУ ВО «ДГУ», заслуженный деятель науки РД

Печатается согласно Постановлению Ученого совета ФГБОУ ВО «ДГТУ»

Оглавление

Введение в технологии графических преобразований

80% информации человек получает посредством зрения. Поэтому необходимо активное развитие способов человеко-машинного взаимодействия. Основная функция компьютера – обработка информации (в том числе и графической).

Компьютерная графика - это область информатики, которая охватывает все стороны формирования изображений с помощью компьютера.

Появившись в 1950-х годах, она поначалу давала возможность выводить лишь несколько десятков отрезков на экране. В наши дни средства компьютерной графики позволяют создавать реалистические изображения, не уступающие фотографическим снимкам Трехмерные изображения используются в медицине (компьютерная томография), картографии, полиграфии, геофизике, ядерной физике и других областях. Телевидение и другие отрасли индустрии развлечений используют анимационные средства компьютерной графики (компьютерные игры, фильмы). Общепринятой практикой считается также использование компьютерного моделирования при обучении пилотов и представителей других профессий (тренажеры). Знание основ компьютерной графики сейчас необходимо и инженеру, и ученому.

Сферы применения компьютерной графики

Сферы применения компьютерной графики включает четыре основных области.

1. Отображение информации

Ни одна из областей современной науки не обходится без графического представления информации. Помимо визуализации результатов экспериментов и анализа данных натурных наблюдений существует обширная область математического моделирования процессов и явлений, которая просто немыслима без графического вывода. В медицине в настоящее время широко используются методы диагностики, использующие компьютерную визуализацию внутренних органов человека. Томография (в частности, ультразвуковое исследование) позволяет получить трехмерную информацию, которая затем подвергается математической обработке и выводится на экран.

2. Проектирование

В строительстве и технике чертежи давно представляют собой основу проектирования новых сооружений или изделий. Построение предварительных макетов - достаточно долгое и дорогое дело. Сегодня существуют развитые программные средства автоматизации проектно-конструкторских работ (САПР, CAD, CAM), позволяющие быстро создавать чертежи объектов, выполнять прочностные расчеты и т.п. Они дают возможность не только изобразить проекции изделия, но и рассмотреть его в объемном виде с различных сторон.

3. Моделирование

Под моделированием в данном случае понимается имитация различного рода ситуаций, возникающих, например, при полете самолета или

космического аппарата, движении автомобиля и т.п. В телевизионной рекламе, в научно-популярных и других фильмах теперь синтезируются движущиеся объекты, визуально мало уступающие тем, которые могут быть получены с помощью кинокамеры. Кроме того, компьютерная графика предоставила киноиндустрии возможности создания спецэффектов, которые в прежние годы были попросту невозможны. В последние годы широко распространилась еще одна сфера применения компьютерной графики - создание виртуальной реальности.

4. Графический пользовательский интерфейс

На раннем этапе использования дисплеев как одного из устройств компьютерного вывода информации диалог "человек-компьютер" в основном осуществлялся в алфавитно-цифровом виде. Теперь же практически все системы программирования применяют графический интерфейс. Особенно впечатляюще выглядят разработки в области сети Internet. Существует множество различных программ-браузеров, реализующих в том или ином виде средства общения в сети, без которых доступ к ней трудно себе представить. Эти программы работают в различных операционных средах, но реализуют, по существу, одни и те же функции, включающие окна, баннеры, анимацию и т.д.

Основные направления в компьютерной графике

В современной компьютерной графике можно выделить следующие основные направления:

изобразительная компьютерная графика, обработка и анализ изображений, анализ сцен (перцептивная компьютерная графика), компьютерная графика для научных абстракций (когнитивная компьютерная графика, т.е. графика, способствующая познанию).

Изобразительная компьютерная графика своим предметом имеет синтезированные изображения. Основные виды задач, которые она решает, сводятся к следующим:

- построение модели объекта и формирование изображения;
- преобразование модели и изображения;
- идентификация объекта и получение требуемой информации.

Обработка и анализ изображений касаются в основном дискретного (цифрового) представления фотографий и других изображений. Средства компьютерной графики здесь используются для:

- повышения качества изображения;
- оценки изображения - определения формы, местоположения, размеров и других параметров требуемых объектов;
- распознавания образов - выделения и классификации свойств объектов (при обработке аэрокосмических снимков, вводе чертежей, в системах навигации, обнаружения и наведения).

Анализ сцен (Компьютерное зрение) связан с исследованием абстрактных моделей графических объектов и взаимосвязей между ними. Объекты могут быть как синтезированными, так и выделенными на фотоснимках. К таким задачам относятся, например, моделирование

"машинного зрения" (роботы), анализ рентгеновских снимков с выделением и отслеживанием интересующего объекта (внутреннего органа), разработка систем видеонаблюдения.

Когнитивная компьютерная графика - только формирующееся новое направление, пока еще недостаточно четко очерченное. Это - компьютерная графика для научных абстракций, способствующая рождению нового научного знания.

Тема 1.Геометрические преобразования на плоскости

В компьютерной геометрии и графике широко используется матричный аппарат вычислений для геометрических преобразований.

Вспомним основные матричные операции:

Сложение и разность двух матриц. Результатом этой операции является матрица элементы которой определяются по следующей формуле:

$$\mathrm{A}_{i,j} \mp B_{i,j} = C_{i,j}$$

где каждый элемент матрицы $\boldsymbol{C_{i,j}}$ представляет собой сумму или разность соответствующих элементов матриц $\boldsymbol{A_{i,j}}$ и $\boldsymbol{B_{i,j}}$.

Умножение матрицы на скаляр осуществляется по тому же принципу:

$$\mathrm{A}_{i,j} * n = C_{i,j}$$

В результате каждый элемент матрицы $\boldsymbol{C_{i,j}}$ представляет собой произведение соответствующих элементов матриц $\boldsymbol{A_{i,j}}$ на скаляр n.

Иная ситуация возникает при перемножении матриц. Данная операция осуществляется по следующему правилу:

$$\mathrm{A}_{i,j} * B_{i,j} = C_{i,j} = \sum_{k=1}^{n} \mathrm{A}_{i,k} * B_{k,j}$$

где n- размерность матриц (количество строк и столбиков). Т.е. У первой матрицы берется строка, второй матрицы столбик, эти элементы попарно перемножаются и складываются. В результате получаем элемент первой строки и первого столбика результирующей матрицы $\boldsymbol{C_{1,1}}$. И т.д. Важным свойством данной операции является то, что произведение матриц не перестановочно!!! Т.е.

$$\mathrm{A}_{i,j} * B_{i,j} \neq B_{i,j} * A_{i,j}$$

Точка $\boldsymbol{P}$ на плоскости однозначно определяется двумя своими координатами $\boldsymbol{(x, y)}$. В соответствие ей можно поставить матрицу-строку размером 1×2 вида $\boldsymbol{[P]=[x \quad y]}$ (сначала будем использовать именно такое отображение точки). Следует заметить также, что точка может задаваться и соответствующей матрицей-столбцом размером 2×1. В любом случае матрицу, определяющую положение точки, часто называют ***координатным вектором*** или ***вектором положения***.

Большинство из перечисленных выше элементарных преобразований по отношению к точке можно реализовать путем умножения матрицы $[P]$ на матрицу общего преобразования размером 2×2 вида $[T]=\begin{bmatrix} a & b \\ c & d \end{bmatrix}$:

$$[P][T]=[x \quad y]\begin{bmatrix} a & b \\ c & d \end{bmatrix}=[(ax+cy) \quad (bx+dy)]=[x^* \quad y^*]=[P^*]$$

где $x^*=ax+cy$, $y^*=bx+dy$ – координаты точки P^*, являющейся результатом преобразования точки P, причем $[P^*]=[x^* \quad y^*]$.

Рассмотрим некоторые специальные случаи.

Умножение исходной матрицы на единичную 2×2 матрицу ($a = d = 1$, $b = c = 0$)

$$[P][T]=[x \quad y]\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}=[x \quad y]=[P]$$

не приводит к каким-либо изменениям; поэтому подобную единичную матрицу часто называют матрицей тождественного преобразования.

В случае $d = 1$, $b = c = 0$

$$[P][T]=[x \quad y]\begin{bmatrix} a & 0 \\ 0 & 1 \end{bmatrix}=[ax \quad y]=[x^* \quad y^*]=[P^*]$$

– происходит так называемое локальное масштабирование (растяжение при $|a| > 1$ или сжатие при $0 < |a| < 1$) координаты x ($x^*=ax$) без изменения координаты y; если при этом $a < 0$, кроме масштабирования происходит отражение относительно оси y.

В аналогичном случае, когда $a = 1$, $b = c = 0$,

$$[P][T]=[x \quad y]\begin{bmatrix} 1 & 0 \\ 0 & d \end{bmatrix}=[x \quad dy]=[x^* \quad y^*]=[P^*]$$

– происходит локальное масштабирование (растяжение при $|d| > 1$ или сжатие при $0 < |d| < 1$) координаты y ($y^*=dy$) без изменения координаты x; при $d < 0$ масштабирование сопровождается отражением относительно оси x.

В общем случае, когда $b = c = 0$,

$$[P][T]=[x \quad y]\begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix}=[ax \quad dy]=[x^* \quad y^*]=[P^*]$$

– наблюдается локальное масштабирование обеих координат; кроме того, при $a < 0$, $d > 0$ происходит отражение относительно оси y, при $a > 0$, $d < 0$ – отражение относительно оси x, при $a < 0$, $d < 0$ – отражение относительно начала координат.

Приведенные примеры позволяют сделать вывод о том, что диагональные члены матрицы преобразования обусловливают локальное

масштабирование и, при соответствующих условиях, отражение относительно координатных осей или точки начала координат.

Посмотрим теперь, на что влияют недиагональные члены этой матрицы.

Пусть $a = d = 1$, $c = 0$. Тогда

$$[P][T] = [x \quad y]\begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix} = [x \quad (bx + y)] = [x^* \quad y^*] = [P^*]$$

– координата $x^* = x$ осталась неизменной, а координата y^* стала линейно зависеть от исходной координаты x; произошел так называемый сдвиг вдоль оси y пропорционально координате x (на bx).

Аналогично, когда $a = d = 1$, $b = 0$,

$$[P][T] = [x \quad y]\begin{bmatrix} 1 & 0 \\ c & 1 \end{bmatrix} = [(x + cy) \quad y] = [x^* \quad y^*] = [P^*]$$

– происходит сдвиг вдоль оси x пропорционально координате y (на cy).

Таким образом установили, что недиагональные члены матрицы преобразования создают эффект сдвига координат исходной точки вдоль координатных осей.

С помощью 2×2 матрицы общего преобразования можно организовать и повороты точек, отрезков и многоугольников относительно начала координат.

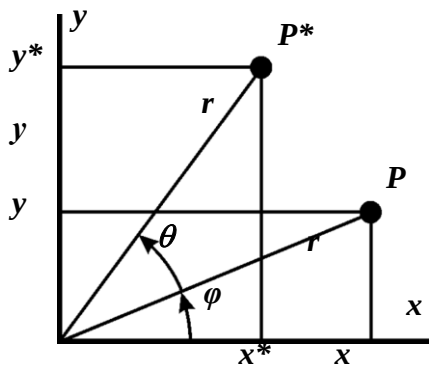


Рис 1.1.Поворот точки на произвольный угол

Рассмотрим теперь, как осуществить поворот на произвольный угол θ на примере преобразования точки $P(x, y)$. Вектор положения данной точки равен

$$[P] = [x \quad y] = [r\cos\varphi \quad r\sin\varphi],$$

где r – длина вектора, а φ – угол наклона вектора по отношению к оси x.

Вектор положения точки $P^*(x^*, y^*)$, полученной из исходной точки путем ее поворота относительно начала координат на положительный угол θ, равен соответственно

$$[P^*] = [x^* \quad y^*] = [r\cos(\varphi + \theta) \quad r\sin(\varphi + \theta)]$$

или, после преобразований –

$$[P^*] = [x^* \quad y^*] =$$
$$= [r(\cos\varphi\cos\theta - \sin\varphi\sin\theta) \quad r(\cos\varphi\sin\theta + \sin\varphi\cos\theta)] =$$
$$= [x\cos\theta - y\sin\theta \quad x\sin\theta + y\cos\theta].$$

Таким образом, преобразованная точка имеет координаты

$$x^* = x\cos\theta - y\sin\theta, \quad y^* = x\sin\theta + y\cos\theta.$$

В матричном виде операцию преобразования можно записать так:

$$[P^*] = [x^* \quad y^*] = [P][T] = [x \quad y]\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}.$$

Другими словами, поворот относительно начала координат на произвольный угол θ задается матрицей преобразования $[T] = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$.

Ранее было замечено, что использование двумерных координатных векторов, отображающих точки на плоскости, в совокупности с матрицей общего преобразования $[T]$ размером 2×2 накладывает ряд ограничений на модификацию объектов. В первую очередь, эти ограничения обусловлены невозможностью применения преобразования координат. Существенно расширить возможности модификации позволяет использование однородных координат для отображения точек и, соответственно, матрицы общего преобразования $[T]$ размером 3×3.

Однородные координаты точки $P\ (x, y)$ на физической плоскости xy представляют собой тройку чисел x', y', h; первые два из них связаны с реальными координатами точки соотношениями $x' = hx$ и $y' = hy$, а h – это некоторое вещественное число (отметим, что случай $h = 0$ является особым и будет рассмотрен ниже). Однородным координатам точки можно поставить в соответствие трехмерный координатный вектор (вектор положения) – матрицу размером 1×3 вида $[x' \quad y' \quad h]$. Очевидно, что при таком подходе каждую точку можно связать с бесконечным множеством наборов однородных координат и, соответственно, координатных векторов вида $[hx \quad hy \quad h]$. Вместе с тем, для точки имеется лишь один набор однородных координат со значением $h = 1$; ему соответствует вектор положения вида $[x \quad y \quad 1]$. В компьютерной графике для отображения точек (за исключением точек бесконечности, см. далее) используются координатные векторы именно такого вида.

Применяя к вектору положения исходной точки $[x \quad y \quad 1]$ матрицу общего преобразования размером 3×3 вида $[T] = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{bmatrix}$, получаем:

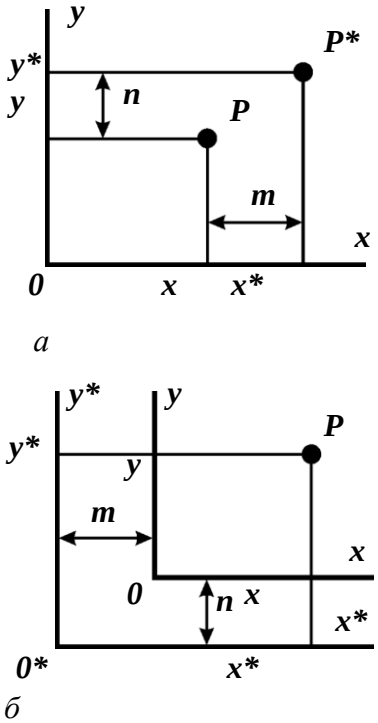
$$[x \quad y \quad 1]\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{bmatrix} = [ax + cy \quad bx + dy \quad 1] = [x^* \quad y^* \quad 1].$$

Данный результат практически идентичен тому, который был получен при умножении координатного вектора $[x \quad y]$ на матрицу общего преобразования размером 2×2 (см. выше). Выражения для координат x^* и

y^* преобразованной точки в обоих случаях полностью совпадают. Следовательно, использование однородных координат совместно с матрицей общего преобразования вида $[T]=\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{bmatrix}$ позволяет осуществить все те преобразования точек, отрезков и многоугольников, о которых шла речь выше (это и происходит при реальной обработке графических объектов). Задавая соответствующие значения a, b, c и d, можно реализовать тождественное преобразование, операции локального масштабирования, отражения, сдвига и поворота, аналогичные уже рассмотренным. Однако, все они, по-прежнему, будут осуществляться относительно точки начала координат.

Исследуем теперь дополнительные возможности, открывающиеся при новом подходе к преобразованиям. Проведем следующую матричную операцию:

$$[x \quad y \quad 1]\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & 1 \end{bmatrix}=[x+m \quad y+n \quad 1]=[x^* \quad y^* \quad 1]$$



а

б

Рис. 1.2. Перемещение

полученный результат можно трактовать двояко: с одной стороны, можно считать, что операция привела к перемещению исходной точки вдоль осей x и y соответственно на m и n в исходной системе координат $x0y$; с другой стороны, можно полагать, что точка осталась на месте, а произошло преобразование координат – новая система координат $x^*0^*y^*$ сдвинута относительно исходной на $-m$ вдоль оси x и на $-n$ вдоль оси y.

В любом случае, выяснилось, что элементы m и n матрицы преобразования размером 3×3 являются коэффициентами перемещения в направлениях x и y соответственно. И не менее важный вывод из приведенного примера – теперь каждая точка плоскости, в том числе начало координат, может быть преобразована.

Матрицу общего преобразования для трехмерных координатных векторов, используемых при двумерных преобразованиях, в общем виде можно представить так:

$$[T]=\begin{bmatrix} a & b & p \\ c & d & q \\ m & n & s \end{bmatrix}.$$

Ранее было установлено, как входящие в нее коэффициенты ***a, b, c, d, m*** и ***n*** влияют на соответствующие преобразования. Остальным трем коэффициентам в предыдущих разделах присваивались вполне определенные значения (***p = q = 0, s = 1***), и они, по сути дела, не принимали участия в преобразованиях. Координатные векторы преобразованных точек всегда имели вид $[x^* \quad y^* \quad 1]$, т.е. число ***h*** тождественно принимало единичное значение. Геометрически это можно трактовать как ограничение преобразований физической плоскостью ***h = 1*** в трехмерном пространстве ***xyh.*** Вместе с тем, при других значениях коэффициентов ***p, q*** и ***s*** они также могут участвовать в преобразованиях.

Рассмотрим сначала, к какому эффекту приведут ненулевые значения коэффициентов ***p*** и ***q***. Запишем следующее выражение:

$$[x \quad y \quad 1]\begin{bmatrix} 1 & 0 & p \\ 0 & 1 & q \\ 0 & 0 & 1 \end{bmatrix}=[x \quad y \quad px+qy+1]=[x' \quad y' \quad h];$$

данное преобразование привело к тому, что точка **(*x, y*)**, которой изначально ставился в соответствие координатный вектор вида $[x \quad y \quad 1]$, преобразована в точку, которой ставится в соответствие координатный вектор вида $[x' \quad y' \quad h]$, где $x' = x$, $y' = y$, $h = px+qy+1$; с геометрической точки зрения полученный результат интерпретируется следующим образом: в трехмерном пространстве ***xyh*** конец ***P*** (рис.2.12) координатного вектора исходной точки принадлежит плоскости ***h = 1***, а конец ***P′*** координатного вектора преобразованной точки – плоскости $h = px+qy+1$ (причем в данном конкретном случае две другие компоненты однородных координат и соответствующего координатного вектора остаются неизменными).

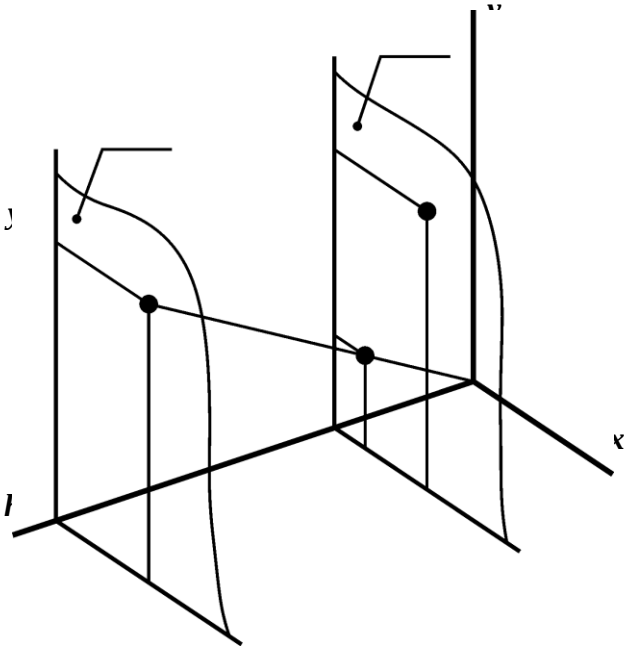


Рис 1.3.

Однако, как отмечалось ранее, в компьютерной графике используют векторы положения только вида $[x \quad y \quad 1]$. Поэтому,

когда какое-либо преобразование приводит к результату с $h \neq 1$ (и, кстати, с $h \neq 0$), этот результат нормализуют, т.е. приводят к требуемому виду путем деления всех трех составляющих однородных координат на величину h. В рассматриваемой задаче окончательный результат преобразования будет иметь вид

$$\begin{bmatrix} x^* & y^* & 1 \end{bmatrix} = \begin{bmatrix} \dfrac{x}{px+qy+1} & \dfrac{y}{px+qy+1} & 1 \end{bmatrix};$$

геометрически такой же результат, а именно точку P^*, можно получить путем проецирования точки P', принадлежащей плоскости $h \neq 1$, на плоскость $h = 1$ по лучу, соединяющему точку P' с началом координат.

Рассмотрим следующее преобразование:

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s \end{bmatrix} = \begin{bmatrix} x & y & s \end{bmatrix} = \begin{bmatrix} x' & y' & h \end{bmatrix},$$

где $x' = x, y' = y, h = s$; нормализуем полученный результат:

$$\begin{bmatrix} x^* & y^* & 1 \end{bmatrix} = \begin{bmatrix} \dfrac{x}{s} & \dfrac{y}{s} & 1 \end{bmatrix};$$

произошло пропорциональное масштабирование координат исходной точки: если $s > 1$ – равномерное сжатие, если $0 < s < 1$ – равномерное растяжение.

Рассмотрев действие всех коэффициентов 3×3 матрицы общего преобразования, сделаем следующее заключение. Условно ее можно разбить на четыре части (подматрицы) –

$$\begin{bmatrix} T \end{bmatrix} = \left[\begin{array}{cc:c} a & b & p \\ c & d & q \\ \hdashline m & n & s \end{array}\right];$$

коэффициенты каждой подматрицы связаны с преобразованиями определенного типа, а именно: коэффициенты левой верхней 2×2 подматрицы (a, b, c и d) – с операциями локального масштабирования, сдвига, отражения и поворота, коэффициенты левой нижней 1×2 подматрицы (m и n) – с перемещениями вдоль координатных осей, коэффициенты правой верхней 2×1 подматрицы (p и q) – с проецированием в однородных координатах, правая нижняя 1×1 подматрица (коэффициент s) – задает общее масштабирование.

Тема 2.Пространственные преобразования

Пространственные преобразования графических объектов так же, как и двумерные преобразования (см. предыдущую тему), реализуются с

использованием однородных координат. Каждая точка с конечными координатами $P\ (x,\ y,\ z)$ в пространстве xyz отображается содержащим однородные координаты этой точки четырехмерным координатным вектором (вектором положения) – матрицей размером 1×4 вида $[P]=[x \quad y \quad z \quad 1]$.

При пространственных преобразованиях применяют матрицу общего преобразования размером 4×4. Обобщенно ее можно представить так:

$$[T]=\left[\begin{array}{ccc:c} a & b & c & p \\ d & e & f & q \\ g & i & j & r \\ \hdashline l & m & n & s \end{array}\right];$$

причем (по аналогии с двумерными преобразованиями) коэффициенты левой верхней 3×3 подматрицы (a, b, c, d, e, f, g, i и j) связаны с операциями локального масштабирования, сдвига, отражения и поворота, коэффициенты левой нижней 1×3 подматрицы (l, m и n) – с перемещениями вдоль координатных осей, коэффициенты правой верхней 3×1 подматрицы (p, q и r) – с проецированием в однородных координатах, правая нижняя 1×1 подматрица (коэффициент s) – задает общее масштабирование.

Тождественное преобразование (не приводящее к изменению объекта) осуществляется единичной матрицей (матрицей тождественного преобразования). Оно сводится к следующему:

$$[P][T]=[x \quad y \quad z \quad 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}=[x \quad y \quad z \quad 1]=[P].$$

Масштабирование при пространственных преобразованиях (так же, как и при двумерных преобразованиях) задается диагональными элементами матрицы общего преобразования. Локальное масштабирование по осям x, y и z связано с коэффициентами соответственно a, e и j этой матрицы, что можно проиллюстрировать следующим примером преобразования точки:

$$[P][T]=[x \quad y \quad z \quad 1]\begin{bmatrix} a & 0 & 0 & 0 \\ 0 & e & 0 & 0 \\ 0 & 0 & j & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}=[x^* \quad y^* \quad z^* \quad 1]=[P^*],$$

где $x^*=ax$, $y^*=ey$, $z^*=jz$.

Симметричные отражения графических объектов относительно координатных плоскостей yz ($x = 0$), xz ($y = 0$) и xy ($z = 0$) можно осуществить матрицами преобразования соответственно

$$[T]=\begin{bmatrix}-1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix},\ [T]=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & -1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$ и

$$[T]=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}.$$

Симметричные отражения относительно координатных осей ***x***, ***y*** и ***z*** (повороты вокруг этих осей на 180°**)** реализуются с использованием соответственно матриц преобразования

$$[T]=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & -1 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix},\ [T]=\begin{bmatrix}-1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$ и

$$[T]=\begin{bmatrix}-1 & 0 & 0 & 0\\ 0 & -1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}.$$

Симметричному отражению относительно точки начала координат будет соответствовать матрица преобразования вида

$$[T]=\begin{bmatrix}-1 & 0 & 0 & 0\\ 0 & -1 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Общее масштабирование можно осуществить, воспользовавшись четвертым диагональным элементом матрицы общего преобразования, т.е. коэффициентом ***s*** (при условии, конечно, что $s \neq 1$ и $s \neq 0$). Приведем пример подобного преобразования точки:

$$[P][T]=[x \quad y \quad z \quad 1]\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & s\end{bmatrix}=[x \quad y \quad z \quad s];$$

окончательный результат после нормализации –

$$[P^*]=[x^* \quad y^* \quad z^* \quad 1]=\left[\frac{x}{s} \quad \frac{y}{s} \quad \frac{z}{s} \quad 1\right];$$

происходит пропорциональное (равномерное) масштабирование координат исходной точки: если $s > 1$ – сжатие, если $0 < s < 1$ – растяжение.

Сдвиги при пространственных преобразованиях обусловливают недиагональные элементы левой верхней 3×3 подматрицы матрицы общего преобразования. Рассмотрим следующее преобразование точки:

$$[P][T]=[x \quad y \quad z \quad 1]\begin{bmatrix} 1 & b & c & 0 \\ d & 1 & f & 0 \\ g & i & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}=$$

$$=[x+dy+gz \quad bx+y+iz \quad cx+fy+z \quad 1]=[x^* \quad y^* \quad z^* \quad 1];$$

произошли сдвиги: вдоль оси x пропорционально координатам y (на dy) и z (на gz), вдоль оси y пропорционально координатам x (на bx) и z (на iz), вдоль оси z пропорционально координатам x (на cx) и y (на fy).

Повороты вокруг координатных осей x, y и z в правосторонней системе координат на произвольные углы θ, ϕ и ψ осуществляются матрицами преобразования соответственно

$$[T]=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad [T]=\begin{bmatrix} \cos\phi & 0 & -\sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ и}$$

$$[T]=\begin{bmatrix} \cos\psi & \sin\psi & 0 & 0 \\ -\sin\psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Преобразование вида

$$[P][T]=[x \quad y \quad z \quad 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ l & m & n & 1 \end{bmatrix}=$$

$$=[x+l \quad y+m \quad z+n \quad 1]=[x^* \quad y^* \quad z^* \quad 1]$$

реализует перемещения: вдоль оси x на l, вдоль оси y на m, вдоль оси z на n.

При пространственных преобразованиях проецирование в однородных координатах связано с коэффициентами p, q и r матрицы общего

преобразования. Рассмотрим следующее преобразование при ненулевых значениях этих коэффициентов применительно к точке:

$$[P][T]=[x \quad y \quad z \quad 1]\begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{bmatrix}=$$

$$=[x \quad y \quad z \quad px+qy+rz+1]=[x \quad y \quad z \quad h],$$

где $h=px+qy+rz+1$; если $h\neq 1$ и $h\neq 0$, окончательный результат преобразования (после нормализации) будет следующим:

$$[P^*]=[x^* \quad y^* \quad z^* \quad 1]=$$

$$=\left[\frac{x}{px+qy+rz+1} \quad \frac{y}{px+qy+rz+1} \quad \frac{z}{px+qy+rz+1} \quad 1\right].$$

Тема 3.Растровое преобразование графических примитивов

Алгоритмы Брезенхема растровой дискретизации отрезка

Экран растрового дисплея можно рассматривать как матрицу дискретных элементов, или пикселей. Процесс определения пикселей, наилучшим образом аппроксимирующих некоторую геометрическую фигуру, называется разложением в растр, или построением растрового образа фигуры. Построчная визуализация растрового образа называется растровой разверткой данной фигуры.

В компьютерной графике необходимо иметь возможность генерировать отрезки, многоугольники и многогранники (представляющиеся как совокупности определенным образом ориентированных отрезков), окружности, эллипсы, параболы, гиперболы и т.д. Алгоритмы генерации таких фигур основаны на схожих принципах. Так, разложение фигур в растр базируется в большинстве случаев на пошаговом методе. Он предполагает построение фигуры или ее части путем поочередного перехода от текущего пикселя к одному из соседних пикселей, который, в свою очередь, становится текущим. При этом на каждом шаге вопрос о выборе конкретного перехода среди альтернативных вариантов решается с использованием, по сути дела, аналогичных с геометрической точки зрения критериев.

Алгоритмы Брезенхема растровой дискретизации отрезка

При построении растрового образа отрезка необходимо:

- Установить критерии "хорошей" аппроксимации (отрезок должен начинаться и кончаться в заданных точках и при этом выглядеть сплошным и прямым);
- Кроме того, яркость вдоль отрезка должна быть одинаковой и не зависеть от наклона отрезка и его длины (горизонтальные и вертикальные отрезки всегда будут ярче наклонных, а постоянная яркость вдоль отрезка опять же достигается на вертикальных, горизонтальных и наклоненных под углом в 45° линиях);
- Алгоритм должен работать быстро. Для этого необходимо по возможности исключить операции с вещественными числами;

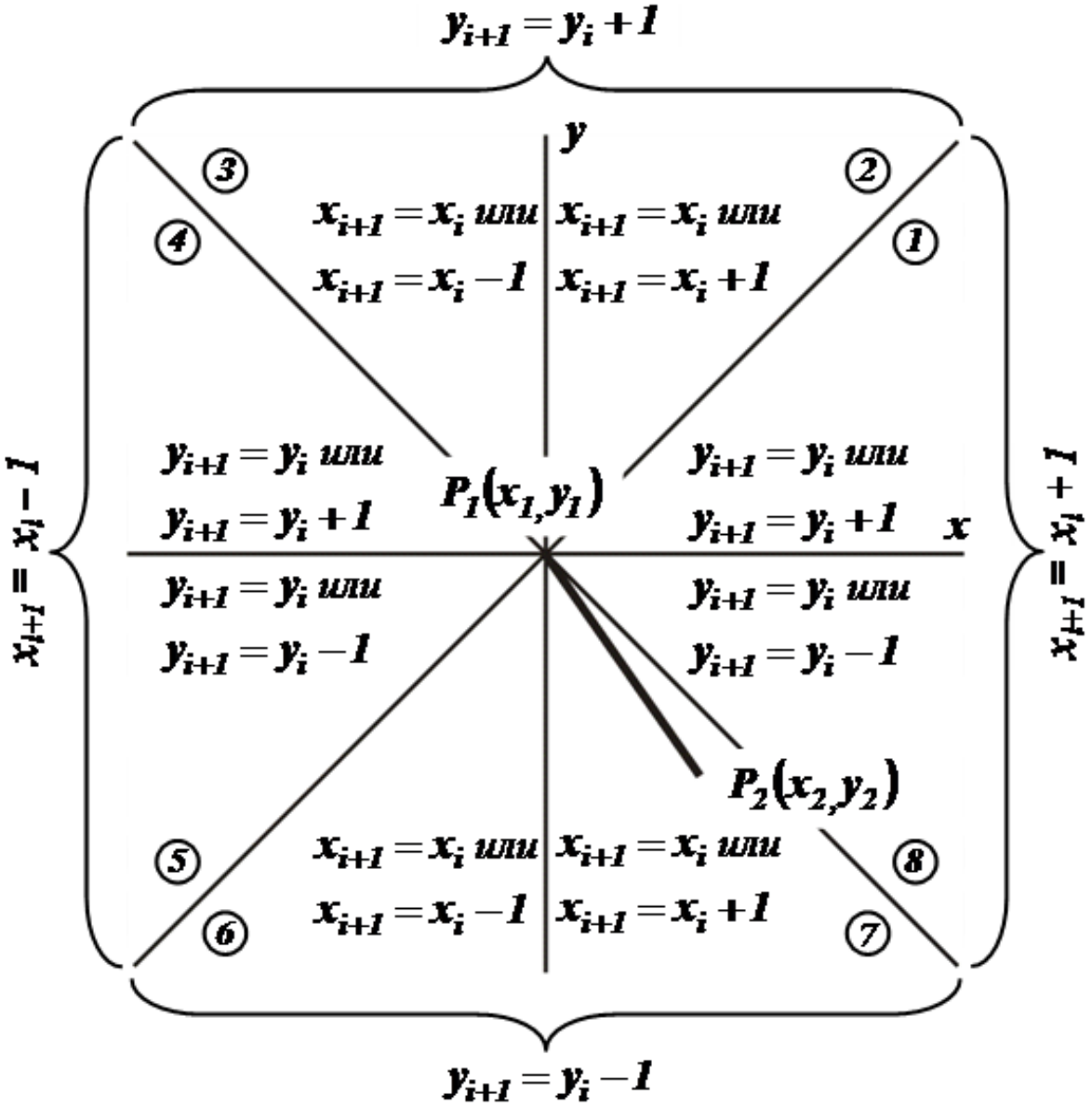


Рис.3.1. Изменения координат на каждом шаге в зависимости от ориентации отрезка

Более приемлемым для генерации отрезков считается ***алгоритм Брезенхема***. В соответствии с ним отрезок вычерчивается пошагово от начала – точки P_1 с координатами (x_1, y_1) – до конца – точки P_2 с координатами (x_2, y_2). Важную роль при этом имеет ориентация отрезка. Ее определяет положение точки P_2 относительно начала координат, которое условно совмещается с точкой P_1 (рис.6.1).

При попадании точки P_2 в какой-либо октант координатной плоскости говорят о том, что весь отрезок расположен в этом октанте (так, например, считается, что отрезок P_1P_2 на (рис.6.1) расположен в седьмом октанте). На каждом шаге в зависимости от величины тангенса угла наклона отрезка (назовем его угловым коэффициентов и обозначим как m) единичное приращение одной из координат задается безусловно. Необходимость единичного изменения другой координаты определяется по знаку рассчитываемой на каждом шаге величины e (ошибки), которая оценивает расстояние между действительным положением отрезка и ближайшими по данной координате адресуемыми точками растра. Если модуль углового коэффициента не больше единицы (1, 4, 5 и 8 октанты), на каждом шаге x изменяется на единицу, а y либо не изменяется, либо изменяется на единицу (рис.3.1), в зависимости от знака ошибки.

При модуле углового коэффициента больше единицы (2, 3, 6 и 7 октанты), наоборот, y безусловно изменяется на единицу, а знак ошибки используется для принятия решения об изменении величины x. Знак же приращений координат x и y зависит от конкретной ориентации отрезка (так, при генерации отрезка P_1P_2 , изображенного на рис.6.1, на каждом шаге y безусловно будет получать приращение -1, а x – либо изменяться на $+1$, либо оставаться неизменным).

Ознакомимся более подробно с принципами работы ***вещественного алгоритма Брезенхема для генерации отрезка в первом октанте*** (впоследствии он будет преобразован в целочисленный и обобщен на случай произвольной ориентации отрезка). Значение углового коэффициента отрезка, расположенного в первом октанте, лежит в диапазоне $0 \le m \le 1$, и на каждом шаге x безусловно получает приращение $+1$, а вопрос о приращении y на $+1$ решается с учетом знака ошибки e. Каким образом она формируется и используется для выбора решения, можно рассмотреть на следующем примере.

Допустим, необходимо построить отрезок в первом октанте с угловым коэффициентом $m = 3/8$. Точка начала отрезка с координатами (x_1, y_1) (впрочем, как и точка его конца) совпадает с каким-либо узлом растра. Активизируем пиксель с такими же координатами адресуемой точки и инициализируем начальное значение ошибки $e = -1/2$ (рис.3.2). Горизонтальная координата следующей адресуемой точки очевидна: $x = x_1 + 1$. Новое значение ошибки рассчитывается путем добавления к ней значения углового коэффициента: $e = e + m = -1/2 + 3/8 = -1/8$. Отрицательное значение ошибки свидетельствует о том, что пересечение реального отрезка с прямой $x = x_1 + 1$ расположено ближе к прямой $y = y_1$, чем к прямой $y = y_1 + 1$ (рис.6.2), поэтому точка растра $(x_1 + 1, y_1)$ лучше аппроксимирует ход отрезка, чем точка $(x_1 + 1, y_1 + 1)$. Активизировав пиксель с координатами адресуемой точки $(x_1 + 1, y_1)$, делаем следующий шаг: $x = x_1 + 2$, $e = e + m = -1/8 + 3/8 = 1/4$; т.к. ошибка положительна, задаем единичное приращение по вертикали $y = y_1 + 1$. Активизируем пиксель с адресуемой точкой $(x_1 + 2, y_1 + 1)$. Вместе с тем, как только ошибка стала положительной, ее значение необходимо скорректировать путем вычитания единицы: $e = e - 1 = 1/4 - 1 = -3/4$. Дальнейшие действия алгоритма аналогичны: $x = x_1 + 3$, $e = -3/4 + 3/8 = -3/8$, y не увеличивается, активизируется пиксель $(x_1 + 3, y_1 + 1)$ и т.д. Отметим здесь только еще один нюанс: если на каком-либо шаге ошибка принимает нулевое значение, нет предпочтительного варианта задания вертикальной координаты (т.к. отрезок пересекает соответствующую вертикаль строго посередине между двумя узлами растра); для определенности алгоритм Брезенхема в этом случае дает приращение y на единицу и корректирует значение ошибки. Поэтому при расчете следующей точки в рассмотренном только что примере результаты будут такими: $x = x_1 + 4$, $e = -3/8 + 3/8 = 0$,

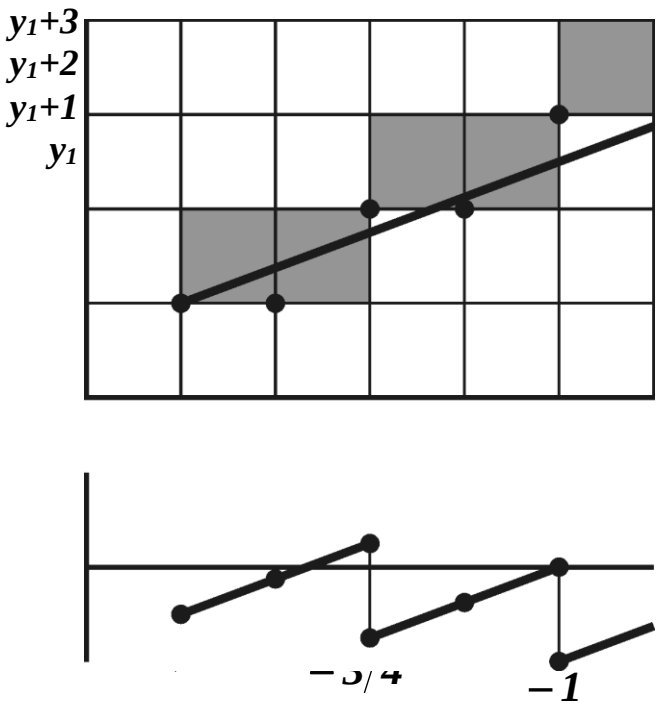


Рис.3.2. Пример разложения в растр отрезка в первом октанте

$y = y_1 + 2$, активизация пикселя с координатами адресуемой точки $(x_1 + 4, y_1 + 2)$, корректировка значения ошибки $e = 0 - 1 = -1$ (см. рис.3.2).

Алгоритм Брезенхема для генерации отрезка в первом октанте на блок-схеме выглядит так:

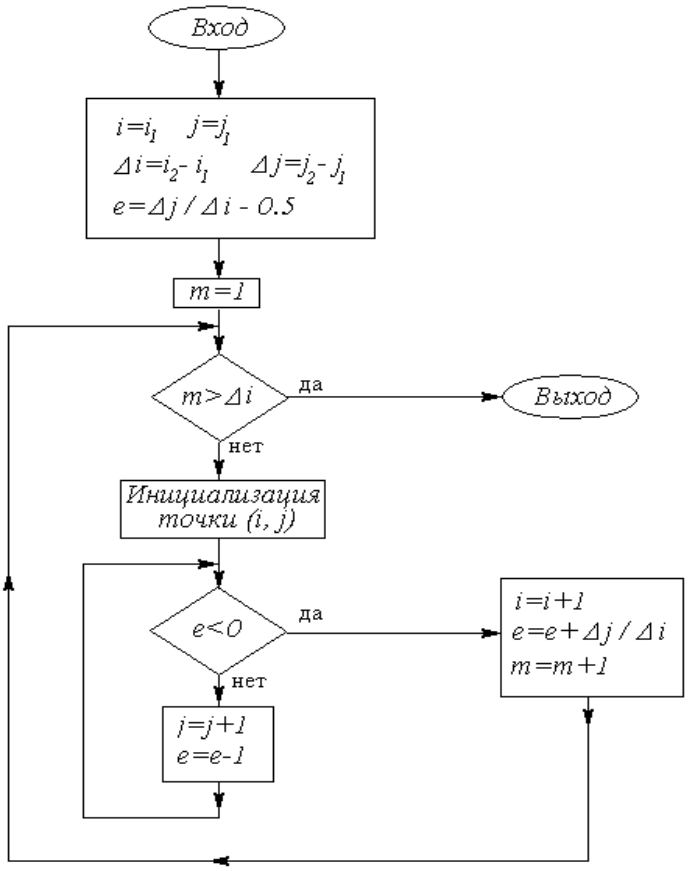


Рис.3.3. Блок-схема алгоритма Брезенхема для генерации отрезка в первом октанте

```
Program Lr_3;   { Алгоритмы Брезенхема растровой дискретизации отрезка }
Uses GraphABC, CRT;
Const i1=100;
        i2=300;
        j1=100;
        j2=300;
Var  i,j,di,dj,m:integer;
                e:real;
Begin
        i:=i1;
        j:=j1;
       di:=i2-i1;
       dj:=j2-j1;
       e:=dj/di-0.5;
       m:=1;
```

```
While m<=di do begin
                    Delay(10);
                    PutPixel(i,j,clBlack);
                    While e>0 do begin
                                        j:=j+1;
                                        e:=e-1;
                                end;
                    i:=i+1;
                    e:=e+dj/di;
                    m:=m+1;
            end;

end.
```

Алгоритм Брезенхема для генерации окружности

Существуют несколько версий алгоритма Брезенхема для генерации окружности. Они имеют ряд общих признаков. Так, во всех этих версиях изначально пошагово генерируется одна восьмая часть окружности, расположенная во втором октанте координатной плоскости, при условии, что центр окружности совмещен с точкой начала координат (рис.3.4); причем ее построение начинается в точке $(x = 0, y = R)$ и осуществляется в направлении по часовой стрелке. При этом y является монотонно убывающей функцией x. Остальные части окружности достраиваются последовательными симметричными отражениями: сгенерированной восьмой части во втором октанте – относительно прямой $y = x$ в первый октант, а полученной после этого четверти окружности в первом квадранте – относительно координатных осей x и y, а также точки начала координат в четвертый, второй и третий квадранты соответственно. Во всех случаях арифметика целочислена, при анализе альтернативных вариантов перехода (от пикселя к пикселю) используются не численные значения, а только знаки определенных величин (критериев), что упрощает и ускоряет вычисления. Отличаются версии, по сути дела, только несколько разными с математической, в том числе геометрической точки зрения подходами. Рассмотрим кратко первую, самую раннюю, версию алгоритма (сейчас она практически вышла из употребления) и

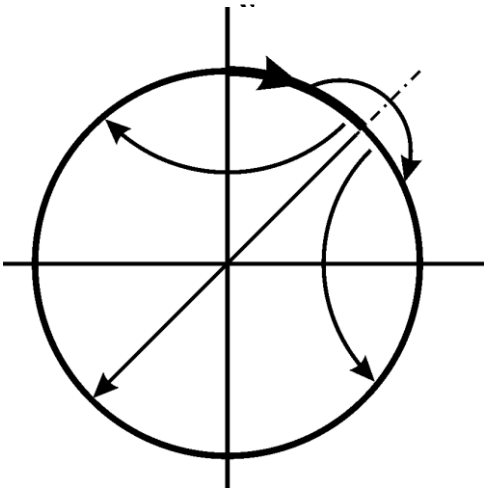


Рис.3.4. Генерация одной восьмой части окружности и ее симметричные отражения

одну из тех, которые появились позже и применяются в настоящее время.

Теоретические положения, на которых базируется первая версия алгоритма Брезенхема, пригодны для построения четверти (а не одной восьмой части) окружности в первом квадранте. При нахождении в этом квадранте в некоторой текущей точке растра с координатами (x_i, y_i) есть только три варианта выбора очередного пикселя (рис.3.5): по горизонтали вправо (направление ***H***), по диагонали вниз и вправо (направление ***D***) и по вертикали вниз (направление ***V***). Вместе с тем, в каждой такай ситуации возможны пять типов пересечений реальной окружности и сетки растра (рис.6.5).

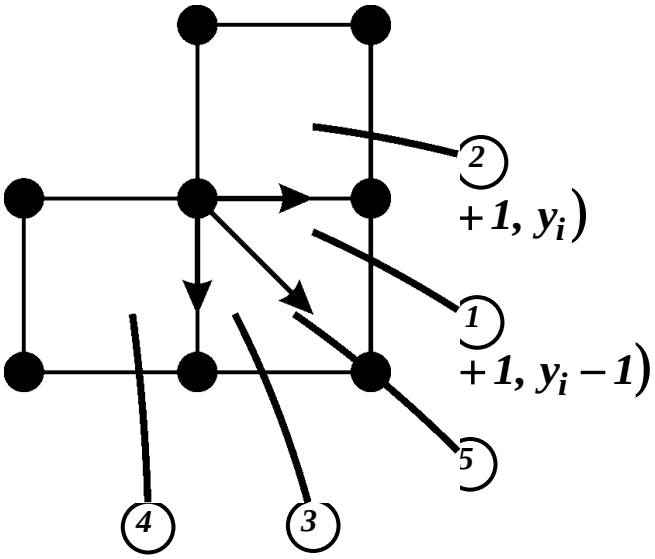


Рис.3.5. К выбору очередного пикселя по первой версии алгоритма Брезенхема

В качестве первого критерия для выбора очередного шага используется разность между квадратами расстояний от центра окружности до диагональной (по отношению к текущей) точки, т.е. точки с координатами $(x_i + 1, y_i - 1)$, и до реальной окружности:

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

При $\Delta_i < 0$ диагональная точка находится внутри окружности, т.е. это случаи пересечения ***1*** или ***2***. Ясно, что в данной ситуации следует выбрать либо точку $(x_i + 1, y_i)$, т.е. шаг ***H***, либо точку $(x_i + 1, y_i - 1)$, т.е. шаг ***D***.

Вторым критерием перехода при таких условиях служит величина

$$\delta = \left| (x_i + 1)^2 + y_i^2 - R^2 \right| - \left| (x_i + 1)^2 + (y_i - 1)^2 - R^2 \right| ;$$

она представляет собой разность двух модулей: под первым из них стоит разность между квадратами расстояний от центра окружности до горизонтальной (по отношению к текущей) точки и до реальной окружности; под вторым – разность между квадратами расстояний от центра окружности до диагональной (по отношению к текущей) точки и до реальной окружности.

При $\delta \le 0$ расстояние от окружности до горизонтальной точки не больше расстояния до диагональной точки, и следует выбирать горизонтальную точку, т.е. шаг ***H***. При $\delta > 0$ выбирается диагональная точка, т.е. шаг ***D***, т.к. расстояние от окружности до нее меньше.

Вместе с тем, при варианте пересечения ***1*** (см. рис.6.5)

$$(x_i + 1)^2 + y_i^2 - R^2 \ge 0 , \quad (x_i + 1)^2 + (y_i - 1)^2 - R^2 < 0 ,$$

т.к. горизонтальная точка лежит вне или на окружности, а диагональная – внутри нее, поэтому δ можно вычислить по формуле:

$$\delta = (x_i + 1)^2 + y_i^2 - R^2 + (x_i + 1)^2 + (y_i - 1)^2 - R^2;$$

или (после преобразования с учётом выражения для Δ_i):

$$\delta = 2(\Delta_i + y_i) - 1.$$

При варианте пересечения ***2*** (см. рис.6.5) расстояние от окружности до горизонтальной точки безусловно меньше расстояния от нее же до диагональной точки, и выбирать следует горизонтальную. При этом обе точки лежат внутри окружности, поэтому:

$$(x_i + 1)^2 + y_i^2 - R^2 < 0, \quad (x_i + 1)^2 + (y_i - 1)^2 - R^2 < 0,$$

и для расчета δ можно использовать ту же формулу, что и при варианте 1, т.к. она дает заведомо отрицательное значение.

Если $\Delta_i > 0$, то диагональная точка находится вне окружности, т.е. это случаи пересечения ***3*** или ***4*** (рис.6.5). Выбирать следует либо точку $(x_i + 1, y_i - 1)$, т.е. шаг ***D***, либо точку $(x_i, y_i - 1)$, т.е. шаг ***V***.

Можно показать (аналогично тому, как это было сделано для предыдущего случая), что критерием для выбора при этом является величина

$$\delta' = 2(\Delta_i - x_i) - 1;$$

при $\delta' \le 0$ выбирается диагональная точка, т.к. расстояние от окружности до нее не больше, чем до вертикальной (по отношению к текущей); при $\delta' > 0$ выбирается вертикальная точка, т.к. расстояние от окружности до нее меньше, чем до диагональной.

Очевидно, что при $\Delta_i = 0$ диагональная точка $(x_i + 1, y_i - 1)$ расположена непосредственно на окружности (вариант пересечения ***5*** на рис.7.5), и выбирается именно она.

Подведем итог. Согласно рассматриваемой версии алгоритма Брезенхема:

- если $\Delta_i < 0$:

  при $\delta \le 0$ выбираем направление ***H***, т.е. пиксель $(x_i + 1, y_i)$;

  при $\delta > 0$ выбираем направление ***D***, т.е. пиксель $(x_i + 1, y_i - 1)$;

- если $\Delta_i > 0$:

  при $\delta' \le 0$ выбираем направление ***D***, т.е. пиксель $(x_i + 1, y_i - 1)$;

  при $\delta' > 0$ выбираем направление ***V***, т.е. пиксель $(x_i, y_i - 1)$;

- если $\Delta_i = 0$ выбираем направление ***D***, т.е. пиксель $(x_i + 1, y_i - 1)$.

При переходе к очередному текущему пикселю можно использовать следующие выражения (для Δ_{i+1} – без вывода):

- при шаге в горизонтальном направлении ***H***:

$x_{i+1} = x_i + 1; \quad y_{i+1} = y_i; \quad \Delta_{i+1} = \Delta_i + 2x_{i+1} + 1;$

- при шаге в диагональном направлении D:

$x_{i+1} = x_i + 1; \quad y_{i+1} = y_i - 1; \quad \Delta_{i+1} = \Delta_i + 2x_{i+1} - 2y_{i+1} + 2;$

- при шаге в вертикальном направлении V:

$x_{i+1} = x_i; \quad y_{i+1} = y_i - 1; \quad \Delta_{i+1} = \Delta_i - 2y_{i+1} + 1.$

Если сгенерирована четверть окружности в первом квадранте, для построения полной окружности эту четверть следует симметрично отразить относительно осей x и y и точки начала координат, т.е. активизировать не только выбранные пиксели (x_i, y_i), но и симметричные им пиксели $(x_i, -y_i)$, $(-x_i, y_i)$ и $(-x_i, -y_i)$.

Ниже представлена блок-схема составленная с учетом изложенных соображений алгоритма для прорисовки окружности с исходной генерацией ее четверти:

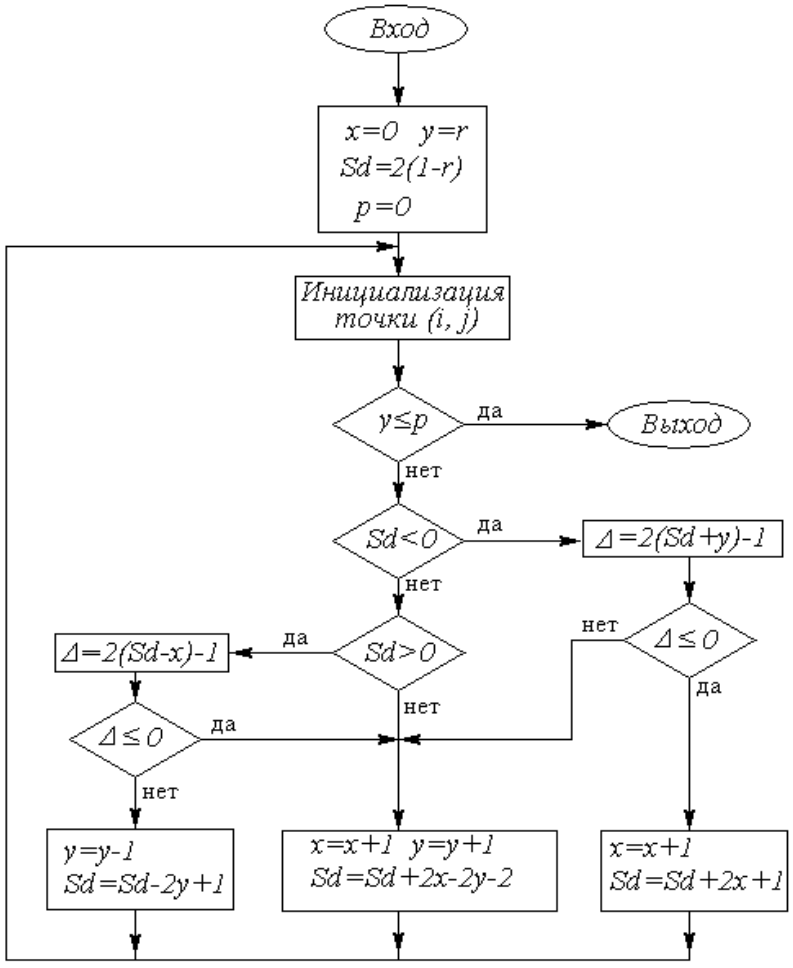


Рис.3.6. Блок-схема алгоритма для прорисовки окружности с исходной генерацией ее четверти

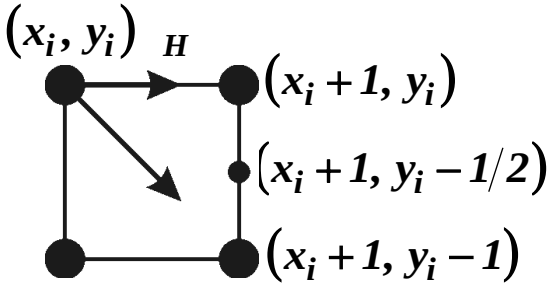


Рис.6.7. К выбору очередного пикселя по алгоритму средней точки

Рассмотрим теперь более позднюю и достаточно простую версию алгоритма Брезенхема (*алгоритм средней точки*). В ней учитывается то обстоятельство, что при построении одной восьмой части окружности во втором октанте на каждом шаге возможны только два варианта перехода: по горизонтали, т.е. шаг H, и по диагонали, т.е. шаг D (рис.3.7). Выбор между ними осуществляется с использованием единственного критерия – разности между квадратами расстояний от центра окружности до точки с координатами $(x_i+1, y_i-1/2)$ (так называемой средней точки, расположенной как раз посередине между альтернативными точками перехода, см. рис.7.7) и до реальной окружности:

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

Если $\Delta_i < 0$, средняя точка находится внутри окружности, и выбирается шаг H; если $\Delta_i \geq 0$, эта точка расположена вне окружности или непосредственно на ней, и выбирается шаг D.

При переходе к очередному текущему пикселю можно использовать следующие выражения (для Δ_{i+1} – без вывода):

- при шаге в горизонтальном направлении H:

$$x_{i+1} = x_i + 1; \quad y_{i+1} = y_i; \quad \Delta_{i+1} = \Delta_i + 2x_i + 3;$$

- при шаге в диагональном направлении D:

$$x_{i+1} = x_i + 1; \quad y_{i+1} = y_i - 1; \quad \Delta_{i+1} = \Delta_i + 2x_i - 2y_i + 5 .$$

Алгоритмы Брезенхема растровой дискретизации эллипса

Существуют несколько версий алгоритма генерации эллипса. Все они являются, по сути дела, обобщениями соответствующих версий алгоритма Брезенхема для генерации окружности. Основные принципы построения эллипса схожи во всех версиях. Центр эллипса условно совмещается с точкой начала координат (как и при построении окружности), а оси эллипса – с координатными осями. При этом уравнение эллипса имеет вид

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1;$$

или

$$b^2x^2 + a^2y^2 - a^2b^2 = 0 .$$

Генерируется одна четвертая часть эллипса в первом квадранте; затем эта четверть симметрично отражается во второй, третий и четвертый квадранты (рис.6.8). Пошаговое построение четверти эллипса начинается в точке $(x=0,\ y=b)$ и осуществляется в направлении по часовой стрелке. Данный процесс реализуется в два этапа: на первом прорисовывается участок, на котором модуль углового коэффициента принимает значения в диапазоне от нуля до единицы (на рис.6.8 – участок ***1***); на втором – участок, на котором угловой коэффициент по модулю больше единицы (на рис.6.8 – участок ***2***). При решении вопроса о том, на каком шаге следует перейти от построения первого участка к построению второго участка, можно использовать следующие соображения. Теоретически для первого участка справедливо неравенство $b^2x < a^2y$, для второго – неравенство $b^2x > a^2y$; в точке же, разделяющей эти участки, выполняется условие $b^2x = a^2y$. Подстановка этого условия в уравнение эллипса и решение последнего в общем виде относительно x дает выражение для расчета горизонтальной координаты в точке, разделяющей участки:

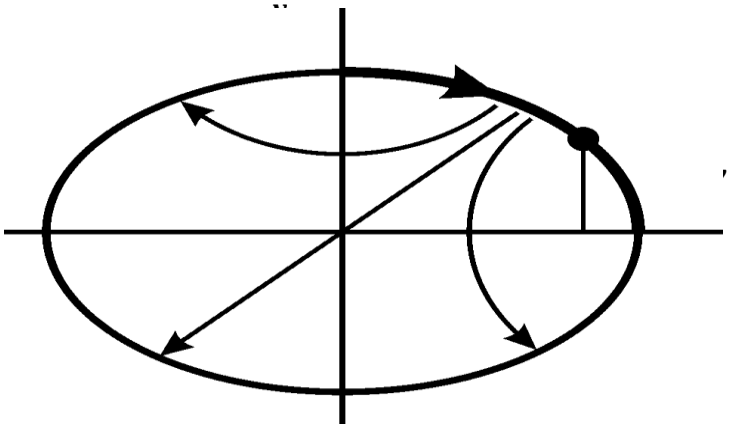


Рис.3.8. Генерация четверти эллипса и ее симметричные отражения

$$x_m = \frac{a^2}{\sqrt{a^2+b^2}}.$$

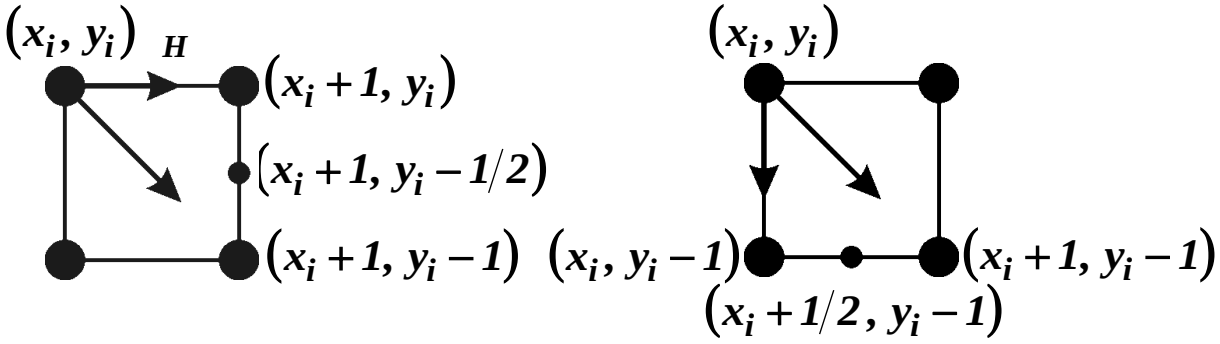
При нахождении в некоторой текущей точке с координатами (x_i, y_i) на первом участке есть только два варианта перехода к очередной точке (как и при построении одной восьмой части окружности во втором октанте): по горизонтали, т.е. шаг H, и по диагонали, т.е. шаг D (рис.6.9а). Критерий перехода формируется путем подстановки в левую часть уравнения эллипса (второго из двух, приведенных выше) координат точки $(x_i+1, y_i-1/2)$ (т.е. средней точки, расположенной посередине между альтернативными точками перехода):

$$\Delta_i = b^2(x_i+1)^2 + a^2(y_i-1/2)^2 - a^2b^2;$$

с геометрической точки зрения получаемая величина пропорциональна разности квадратов вертикальных координат средней точки и точки,

принадлежащей четверти эллипса, при $x = x_i + 1$ (коэффициентом пропорциональности служит a^2).

Если $\Delta_i < 0$, эллипс проходит выше средней точки, и выбирается шаг H; если $\Delta_i \geq 0$, эллипс проходит ниже этой точки или непосредственно через нее, и выбирается шаг D.



а *б*

Рис.3.9. К выбору очередного пикселя при генерации эллипса

При переходе к очередному текущему пикселю можно использовать следующие выражения (для Δ_{i+1} – без вывода):

- при шаге в горизонтальном направлении H:

$$x_{i+1} = x_i + 1; \quad y_{i+1} = y_i; \quad \Delta_{i+1} = \Delta_i + b^2(2x_i + 3);$$

- при шаге в диагональном направлении D:

$$x_{i+1} = x_i + 1; \quad y_{i+1} = y_i - 1; \quad \Delta_{i+1} = \Delta_i + b^2(2x_i + 3) + a^2(2 - 2y_i).$$

При нахождении текущей точке (x_i, y_i) на втором участке вариантов перехода к очередной точке тоже два: по диагонали, т.е. шаг D, или по вертикали, т.е. шаг V (рис.6.9б). Критерий перехода формируется по аналогии с предыдущим, но в данном случае используется средняя точка $(x_i + 1/2, y_i - 1)$:

$$\Delta_i = b^2(x_i + 1/2)^2 + a^2(y_i - 1)^2 - a^2b^2;$$

получаемая величина пропорциональна разности квадратов горизонтальных координат средней точки и точки, принадлежащей четверти эллипса, при $y = y_i - 1$ (коэффициент пропорциональности здесь равен b^2).

Если $\Delta_i < 0$, эллипс проходит правее средней точки, и выбирается шаг D; если $\Delta_i \geq 0$, эллипс проходит левее этой точки или непосредственно через нее, и выбирается шаг V.

Переход к очередному текущему пикселю осуществляется с использованием следующих выражений (для Δ_{i+1} – без вывода):

- при шаге в диагональном направлении D:

$$x_{i+1} = x_i + 1;\quad y_{i+1} = y_i - 1;\quad \Delta_{i+1} = \Delta_i + b^2(2x_i + 2) + a^2(3 - 2y_i);$$

- при шаге в вертикальном направлении ***V***:

$$x_{i+1} = x_i;\quad y_{i+1} = y_i - 1;\quad \Delta_{i+1} = \Delta_i + a^2(3 - 2y_i).$$

Вопрос о выборе критерия, на основании которого осуществляется переход от построения первого участка к построению второго участка, решается неоднозначно. Будем считать, что первый участок следует выстраивать до тех пор, пока горизонтальная координата точки, сдвинутой вправо на ***1/2*** относительно текущей, меньше чем x_m, т.е. пока выполняется условие $x_i + 1/2 < x_m$ или (что то же самое) $x_i < x_m - 1/2 = x_m'$.

Тема 4.Заполнение внутренних областей

Общие замечания

В компьютерной графике ***заполнением многоугольника*** (или ***заполнением контура***) называют процесс генерации сплошной области, т.е. заполнение этой области определенным цветом. Фигура, ограничивающая область, может быть непосредственно многоугольником или замкнутой кривой, которая при разложении в растр представляется многоугольником. Задать ограничивающий многоугольник можно координатами вершин или описаниями его ребер. Впрочем, граница области может быть определена как-либо иначе, например, совокупностью пикселей, образующих ограничивающий область контур.

Алгоритмы заполнения разделяют на две группы. Входящие в первую группу алгоритмы базируются на так называемых методах ***растровой развертки***, алгоритмы второй группы – на методах ***затравочного заполнения***. Часто алгоритмы этих групп называют просто ***алгоритмами растровой развертки*** и ***алгоритмами затравочного заполнения*** соответственно.

Общие принципы работы алгоритмов растровой развертки можно описать следующим образом. Для всех ребер многоугольника, за исключением горизонтальных, так или иначе, определяются точки их пересечения с осями сканирующих строк, сдвинутыми по оси ***y*** в координатной сетке растра на ***1/2*** вверх относительно координат адресуемых точек соответствующих рядов пикселей. Независимо от действий конкретного алгоритма результат получается следующим: на каждой сканирующей строке, имеющей такие пересечения, цветом многоугольника заполняются интервалы между парами пересечений, встречающимися на сканирующей строке при следовании вдоль нее от начала к концу; крайние интервалы (от начала строки до первого пересечения и от последнего пересечения до конца строки) и интервалы между этими парами пересечений заполняются фоновым цветом.

В алгоритмах затравочного заполнения один пиксель внутри заполняемого контура – ***затравка*** – должен быть заранее известен. Начиная с этого пикселя, алгоритмы находят и закрашивают все другие пиксели области, ограниченной контуром. В процессе работы в качестве новой затравки назначаются, по тем или иным соображениям, другие пиксели области. Таким образом, процесс поиска и заполнения осуществляется рекурсивно.

Алгоритм заполнения с упорядоченным списком ребер

Разберем принципы организации одного из алгоритмов растровой развертки, объединенных названием ***алгоритмов заполнения с упорядоченным списком ребер***. Исходными данными для его работы являются целочисленные координаты вершин заполняемого многоугольника и порядок их соединения, определяющий ребра. Последовательность действий алгоритма такова:

- в произвольной последовательности для каждого ребра многоугольника определить точки пересечений с осями сканирующих строк (горизонтальные ребра игнорировать); если ось сканирующей строки с вертикальной координатой $y+0.5$ пересекает ребро с вершинами $P_1(x_1, y_1)$ и $P_2(x_2, y_2)$, горизонтальная координата точки пересечения определяется по формуле $x=(x_2-x_1)(y+0.5-y_1)/(y_2-y_1)+x_1$;
- занести каждое пересечение $(x, y+0.5)$ в список;
- отсортировать список по строкам (сверху вниз) и по возрастанию x в строке – (x_1, y_1) предшествует (x_2, y_2), если $y_1 > y_2$ или $y_1 = y_2$ и $x_1 < x_2$;
- выделить из отсортированного списка пары элементов (x_1, y_1) и (x_2, y_2); при этом структура списка будет гарантировать, что в таких парах $y_1 = y_2$ и $x_1 < x_2$;
- заполнить на соответствующих сканирующих строках пиксели с горизонтальными координатами x адресуемых точек, удовлетворяющих условию $x_1 \le x+0.5 \le x_2$ на каждом интервале (т.е. пиксели, центры которых попадают в интервалы между выделенными парами точек пересечения).

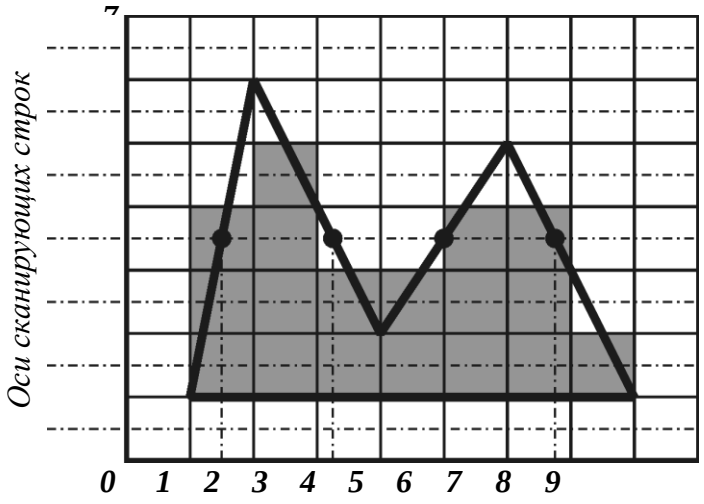


Рис.4.1. Пример работы алгоритма заполнения с упорядоченным списком ребер

При работе подобного алгоритма по заполнению, например, приведенного на рис.8.1 пятиугольника на оси сканирующей строки $y+0.5=3.5$ будут выделены две пары точек: с $x=1.5$ и $x=3.25$; с $x=5$ и $x=6.75$. На данной строке заполнятся пиксели с координатами адресуемых точек **(*1, 3*)**, **(*2, 3*)**, **(*5, 3*)** и **(*6, 3*)**, т.к. для первых двух пикселей выполняется условие: $1.5 \le x + 0.5 \le 3.25$, а для вторых двух: $5 \le x + 0.5 \le 6.75$. Рис.8.1 иллюстрирует также полный результат работы алгоритма.

Особенности алгоритмов затравочного заполнения

При организации работы конкретных алгоритмов затравочного заполнения учитывается специфика геометрии генерируемой области, а также способ задания ее границы. Различают ***гранично-определенные*** и ***внутренне-определенные*** области. Для гранично-определенной области граница задается контуром, содержащим пиксели одного цвета, причем ни один из пикселей самой области не может иметь этот цвет, а пиксели, внешние по отношению к границе, могут. Пиксели, расположенные на границе внутренне-определенной области, могут иметь разные цвета, за исключением цвета самой области. Алгоритмы, заполняющие гранично-определенные области, называются ***гранично-заполняющими***, а алгоритмы, работающие с внутренне-определенными областями – ***внутренне-заполняющими***. Гранично- и внутренне-определенные области могут быть ***четырехсвязными*** или ***восьмисвязными***. Четырехсвязной называется область, в которой от каждого пикселя можно «добраться» до любого другого, переходя от пикселя к пикселю в двух горизонтальных и двух вертикальных направлениях, т.е. вправо, влево, вверх и вниз. В восьмисвязной области это же можно сделать, добавив к указанным направлениям переходов еще четыре диагональных: вправо и вверх, вправо и вниз, влево и вверх, влево и вниз. Рис.4.2 иллюстрирует изложенные положения: на нем приведены примеры четырехсвязной гранично-определенной (рис.4.2а) и восьмисвязной внутренне-определенной (рис.4.2б) областей. Отметим также то, что граница четырехсвязной области является восьмисвязной, а граница восьмисвязной области – четырехсвязной.

Алгоритмы затравочного заполнения четырех- и восьмисвязных гранично- и внутренне-определенных областей строятся на базе схожих

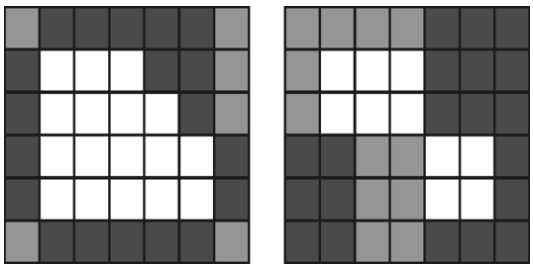
приемов. Далее подробно разбираются алгоритмы, реализующие заполнение только гранично-определенных областей.

Простой алгоритм заполнения с затравкой для четырехсвязной гранично-определенной области

Последовательность действий обозначенного в заголовке данного раздела алгоритма такова:

- поместить координаты адресуемой точки затравочного пикселя в стек;
- пока стек не пуст:
  извлечь данные о пикселе из стека; если ему уже присвоено требуемое значение цвета – проигнорировать, если нет, тогда:
  присвоить пикселю требуемое значение цвета;
  для каждого из соседних (в горизонтальном и в вертикальном направлениях) пикселей проверить, не является ли он граничным или не присвоено ли ему уже требуемое значение; проигнорировать пиксель в любом из этих двух случаев; в противном случае поместить данные о пикселе в стек.

Проследим действия такого алгоритма при заполнении области, представленной на рис.8.3. Выберем в качестве затравки пиксель с координатами адресуемой точки, например, **(*4, 4*)**. Данные об этом пикселе помещаются на ***1*** уровень стека, извлекаются оттуда, и пиксель заполняется, т.е. ему присваивается требуемое значение цвета. Проверка соседних пикселей приводит к помещению в стек на уровни ***1***, ***2*** и ***3*** пикселей, расположенных правее, выше и левее текущего, т.е. **(*5, 4*)**, **(*4, 5*)** и **(*3, 4*)** соответственно. Пиксель ниже текущего принадлежит границе, поэтому в стек не попадает. Далее с ***3*** уровня стека извлекается и заполняется пиксель **(*3, 4*)**. В результате проверки соседних пикселей на освободившийся ***3*** уровень стека попадает пиксель **(*3, 5*)**, а на ***4*** уровень – пиксель **(*2, 4*)**. Последующие шаги алгоритма очевидны. На рис.8.3 последовательность заполнения пикселей обозначена стрелками, а уровни стека, на которые попадают данные о пикселях – числами на изображениях пикселей. Поясним только некоторые



а *б*

Рис.4.2. Четырехсвязная гранично-определенная (а) и восьмисвязная внутренне-определенная (б) области

специфичные случаи в работе алгоритма. После извлечения с ***15*** уровня стека и заполнения пикселя **(*3, 1*)** выясняется, что все соседние с ним пиксели либо уже заполнены, либо принадлежат границе, поэтому ни один из них в стек не помещается. Из стека, с ***14*** его уровня, извлекается пиксель **(*7, 1*)**, после чего заполняется данный пиксель и еще три, расположенные выше него. Проверка пикселей, соседних с последним из заполненных, т.е. с пикселем **(*7, 4*)**, снова не приводит к пополнению стека, который на данный момент имеет тринадцать уровней данных. Последующее извлечение с ***13*** и ***12*** уровней стека данных о пикселях соответственно **(*7, 2*)** и **(*7, 3*)** не приводит к каким-либо последствиям. Последним в области заполняется пиксель **(*6, 5*)**, извлеченный перед этим с ***11*** уровня стека. После проверки соседних с ним пикселей стек, имеющий десять уровней, не пополняется. Далее происходит последовательное извлечение данных с десяти оставшихся уровней стека без какого-либо заполнения. После опустошения стека алгоритм завершает работу.

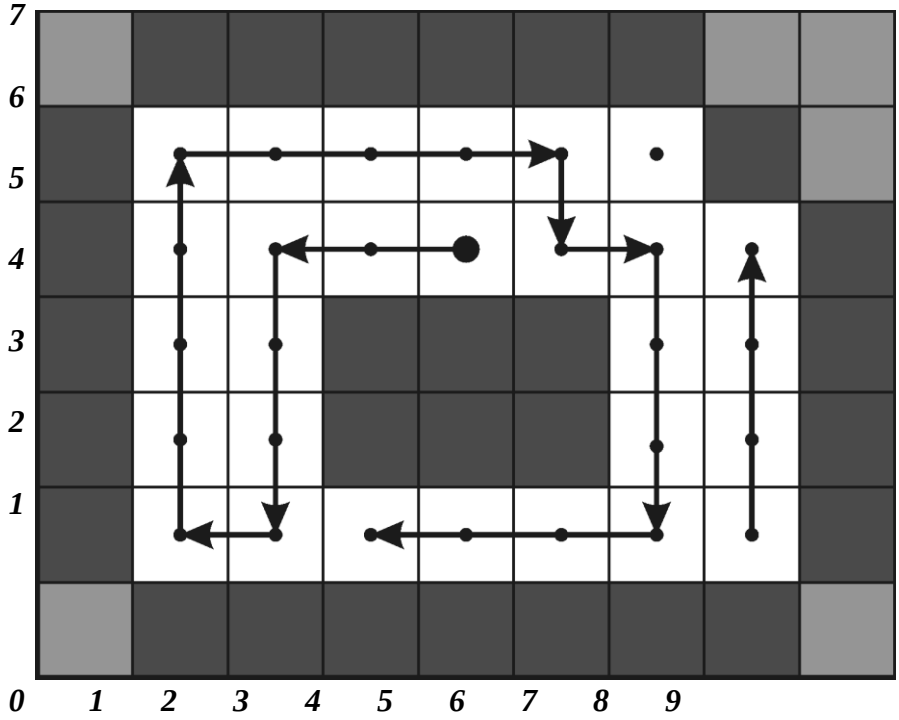


Рис.4.3. Пример работы простого алгоритма заполнения с затравкой для четырехсвязной гранично-определенной области

Рассмотренный алгоритм может быть легко обобщен на случай заполнения восьмисвязных областей. Для этого достаточно проверке на ранее осуществленное заполнение или принадлежность границе подвергать не четыре, а восемь пикселей, соседних с текущим, в том числе четыре, расположенные по отношению к нему диагонально. И, конечно же, при оговоренных выше условиях помещать эти пиксели в стек.

Простой алгоритм затравочного заполнения имеет существенный недостаток: данные о каждом пикселе области попадают в стек, и, зачастую, неоднократно (т.е. происходит дублирование данных), что приводит к необходимости использования стека большого объема и увеличению времени, затрачиваемого на обработку содержащейся в нем информации.

Построчный алгоритм заполнения с затравкой для четырехсвязной гранично-определенной области

Более рациональным для генерации четырехсвязных областей является построчный алгоритм затравочного заполнения. В нем размер стека минимизируется за счет хранения данных только об одном пикселе на любом непрерывном интервале сканирующей строки, принадлежащем внутренней области заполняемого многоугольника (наблюдается, правда, одно исключение из этого утверждения).

Работу такого алгоритма для случая гранично-определенной области схематично можно описать так:

- поместить координаты адресуемой точки затравочного пикселя в стек;
- пока стек не пуст:
  - извлечь данные о пикселе из стека; если ему уже присвоено требуемое значение цвета – проигнорировать, если нет, тогда:
  - присвоить пикселю требуемое значение цвета;
  - заполнить интервал с текущим пикселем вправо и влево от него вдоль сканирующей строки до обнаружения границ;
  - переменным $x_{лев}$ и $x_{пр}$ присвоить значения горизонтальных координат адресуемых точек соответственно крайнего левого и крайнего правого пикселей интервала;
  - в диапазоне $x_{лев} \le x \le x_{пр}$ проверить строки, расположение непосредственно над и под текущей строкой; если в них есть еще не заполненные интервалы (т.е. не все пиксели граничные или уже заполненные), то в указанном диапазоне данные о крайнем правом пикселе на каждом интервале поместить в стек.

Рис.8.4 иллюстрирует последовательность действий при работе такого алгоритма по заполнению конкретной четырехсвязной гранично-определенной области.

Тема 5.Двумерное, трехмерное отсечения

Отсечение – это процесс, связанный с выделением и визуализацией фрагмента плоской или пространственной сцены, расположенного внутри (***внутреннее отсечение***) или, наоборот, вне (***внешнее отсечение***) некоторой соответственно двумерной или трехмерной отсекающей фигуры (***отсекателя***). Оставшаяся часть сцены при этом игнорируется, т.е.

визуализации не подлежит. При реализации такого процесса используются алгоритмы ***двумерного*** или ***трехмерного отсечения***. Специфика конкретного алгоритма определяется, в первую очередь, формой отсекающей фигуры, которая может быть регулярной (стандартной, такой, например, какую имеют прямоугольник или прямоугольный параллелепипед) или нерегулярной (нестандартной). При регулярной форме отсекателя важным фактором является также его ориентация относительно системы координат.

Как уже отмечалось, изображение в компьютерной графике формируется с использованием, прежде всего, отрезков. Совокупностями определенным образом ориентированных отрезков представляются не только ломаные линии, многоугольники и многогранники, но также разомкнутые и замкнутые кривые. Трехмерные поверхности аппроксимируют наборами плоских полигонов, ограниченных многоугольниками (чаще всего, треугольниками), т.е. опять же наборами отрезков. Поэтому и алгоритмы отсечения работают, в первую очередь, с отрезками.

Алгоритм двумерного отсечения Сазерленда-Коэна

Данный алгоритм реализует отсечение отрезков координатно-ориентированным прямоугольником, границы которого: левая, правая, нижняя и верхняя – задаются координатами соответственно $x_л$, $x_п$, $y_н$ и $y_в$ (рис.8.1). При таком отсечении отсекатель часто называют ***отсекающим окном***. Рассмотрим сначала основные принципы построения алгоритма применительно к внутреннему отсечению. Точка $P(x, y)$ находится внутри отсекающего окна, если ее координаты удовлетворяют двум условиям:

$$x_л \le x \le x_п;$$

$$y_н \le y \le y_в$$

(здесь и далее принадлежность точки границе отсекателя приравнивается случаю ее расположения внутри него). Если для обоих концов отрезка эти условия выполняются, он целиком расположен внутри отсекающего окна, т.е. является, как говорят, полностью видимым (как отрезок ***ab*** на рис.8.1). Если оба конца отрезка находятся левее отсекающего окна (их горизонтальные координаты удовлетворяют неравенству $x < x_л$, как у отрезка ***cd***) отрезок безусловно невидим. То же самое можно сказать про отрезки, расположенные

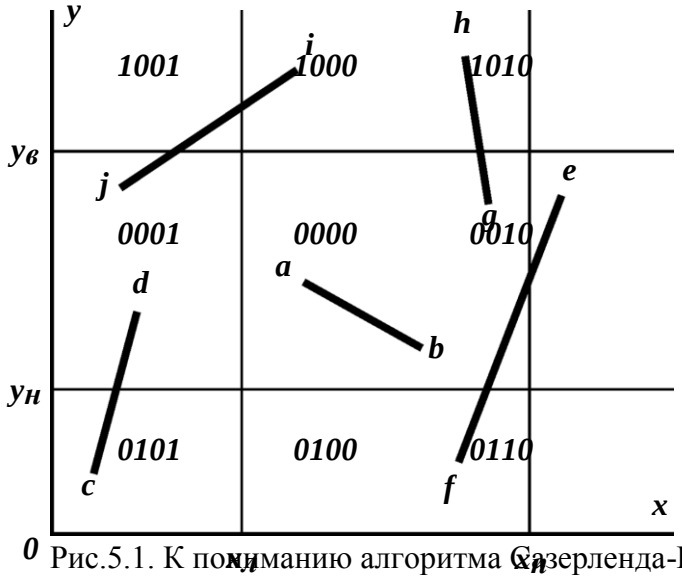


Рис.5.1. К пониманию алгоритма Сазерленда-Коэна

целиком правее, ниже или выше отсекателя (для обоих концов отрезков при этом справедливы неравенства соответственно $x > x_п$, $y < y_н$ или $y > y_в$). Сложности возникают при отсечении таких отрезков, как ***ef*** и ***ij***. Изначально они не идентифицируются как полностью видимые или безусловно невидимые. В действительности же каждый из таких отрезков может быть частично видимым (как отрезок ***ef***) или полностью невидимым (как отрезок ***ij***). Поэтому при анализе подобных отрезков необходимы какие-либо формальные методы, позволяющие определить реальные точки пересечения отрезков с границами отсекающего окна или установить, что они отсутствуют. Задача нахождения реальной точки пересечения отрезка с какой-либо одной границей отсекателя возникает и в случае, когда один из концов отрезка расположен внутри отсекающего окна, а второй – вне его (как у отрезка ***gh***).

В алгоритме Сазерленда-Коэна при обработке отрезков используются четырехразрядные (битовые) коды, определяющие расположение концов отрезков относительно границ отсекающего окна – концевые коды. При формировании кода конца отрезка с координатами **(*x, y*)** в первый (крайний правый) бит заносится ***1***, если $x < x_л$, во второй – если $x > x_п$, в третий – если $y < y_н$, в четвертый – если $y > y_в$. В остальных случаях в соответствующие биты заносится ***0***. Таким образом, вся координатная плоскость разбивается на девять областей, в каждой из которых концу отрезка присваивается уникальный концевой код (см. рис.8.1). Выявить целиком видимый отрезок при этом достаточно просто: оба концевых кода у него полностью нулевые (как у отрезка ***ab*** на рис.8.1). Если отрезок не идентифицирован как целиком видимый, для него определяется также побитовое логическое произведение концевых кодов. Очевидно, что у отрезков, целиком расположенных левее, правее, ниже или выше отсекающего окна, в какой-либо один разряд обоих концевых кодов при их формировании будет занесено по ***1***; это обусловит наличие ***1*** в соответствующем разряде побитового логического произведения этих концевых кодов (так, у отрезка ***cd***, расположенного целиком левее отсекающего окна, с кодами концов ***0101*** и ***0001*** побитовое логическое произведение равно ***0001***). Поэтому отличное от нуля побитовое логическое произведение является признаком безусловной невидимости отрезка. Вместе с тем, при полностью нулевом побитовом логическом произведении концевых кодов отрезок может оказаться частично видимым (как отрезки ***ef*** и ***gh***) или полностью невидимым (как отрезок ***ij***). Для подобных отрезков при выявлении их видимых частей или идентификации как полностью невидимых используются результаты расчета точек пересечения бесконечной прямой линии, проведенной через отрезок, с бесконечными прямыми линиями, проведенными через ребра отсекающего окна.

Для отрезка с началом в точке $P_1(x_1, y_1)$ и концом в точке $P_2(x_2, y_2)$, если он не вертикален ($m = (y_2 - y_1)/(x_2 - x_1) \neq \infty$), такие точки пересечения на прямых, проведенных через левое и правое ребра, будут иметь координаты соответственно:

$x = x_л$, $y = m(x_л - x_1) + y_1$; $x = x_п$, $y = m(x_п - x_1) + y_1$.

Если отрезок не горизонтален ($m \neq 0$), координаты аналогичных точек пересечения на прямых, проведенных через нижнее и верхнее ребра, будут равны соответственно:

$x = (y_н - y_1)/m + x_1$, $y = y_н$; $x = (y_в - y_1)/m + x_1$, $y = y_в$.

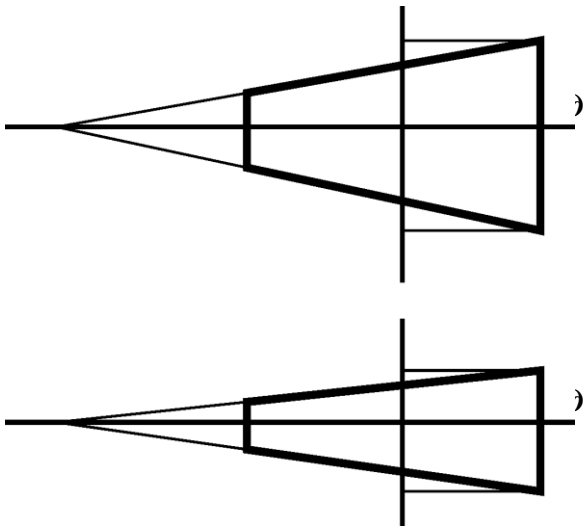
Последовательность действий в алгоритме двумерного внутреннего отсечения Сазерленда-Коэна следующая. С использованием концевых кодов отрезок проверяется на полную видимость и безусловную невидимость. Если проверка не дает очевидного результата, поочередно работают четыре бесконечные отсекающие прямые, проходящие соответственно через левое, правое, нижнее и верхнее ребра отсекающего окна. В каждом случае проверяется, не лежит ли отрезок полностью в той же стороне от отсекающей прямой, что и само окно (очевидно, например, что если в первом бите кодов концов отрезка содержится ***0***, отрезок полностью лежит правее левой отсекающей прямой). Если это так, переходят к следующей отсекающей прямой. В противном случае проверяют, не лежит ли начало отрезка в той же стороне, что и окно. Если это подтверждается, начало и конец отрезка меняются местами. Начало отрезка в любом случае оказывается по отношению к отсекающей прямой в стороне, противоположной той, в которой находится окно. После определения точки пересечения прямой, проведенной через отрезок, с текущей отсекающей прямой начало отрезка переносится в эту точку (удаляется часть отрезка). Полученный новый отрезок вновь проверяется на полную видимость и безусловную невидимость. В зависимости от результата переходят к следующей отсекающей прямой или алгоритм заканчивает работу.

Алгоритм трехмерного отсечения Сазерленда-Коэна

При помощи этого алгоритма можно осуществлять отсечение отрезков двумя наиболее распространенными объемными отсекателями регулярной формы: координатно-ориентированным прямоугольным параллелепипедом (рис.8.2а) и усеченной пирамидой (рис.8.2б), называемой также пирамидой видимости. Каждый из названных отсекателей имеет шесть граней: левую, правую, нижнюю, верхнюю, ближнюю и дальнюю. Так же, как и при двумерном отсечении, положение каждого конца отсекаемого отрезка относительно границ отсекателя (точнее – относительно бесконечных плоскостей, проходящих через его грани) можно связать с соответствующим концевым кодом, имеющим в данном случае шесть разрядов (бит).

Границы первого отсекателя – прямоугольного параллелепипеда – задаются шестью координатами: $x_л$, $x_п$, $y_н$, $y_в$, $z_б$ и $z_д$ (рис.6.2а). При этом процесс формирования кода конца отрезка с координатами **(*x*, *y*, *z*)** полностью аналогичен тому, который применяется при двумерном отсечении: ***1*** заносится в первый (крайний правый) бит, если $x < x_л$ (конец левее отсекателя), во второй – если $x > x_п$ (конец правее отсекателя), в третий – если $y < y_н$

(конец ниже отсекателя), в четвертый – если $y > y_в$ (конец выше отсекателя), в пятый – если $z > z_б$ (конец ближе отсекателя), в шестой – если $z < z_д$ (конец дальше отсекателя); в остальных случаях в соответствующие биты заносится 0.

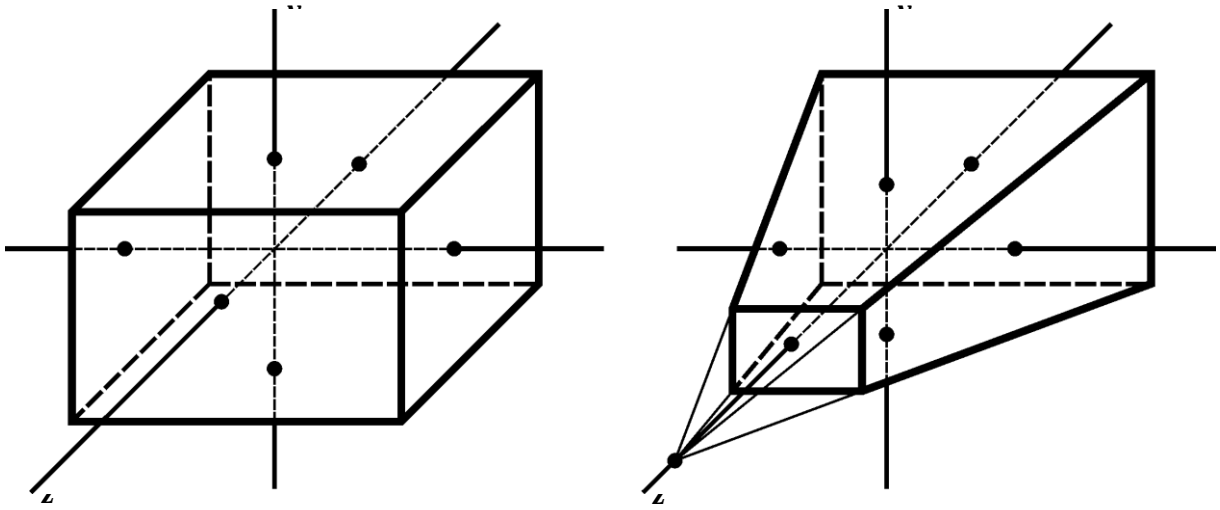


б

Рис.5.2. Пирамида видимости: виды сверху (а) и справа (б)

При использовании в качестве отсекателя усеченной пирамиды концевые коды формируются с учетом специфики геометрии данного объемного тела. Допустим, что она задана следующим образом (рис.8.4а и 8.4б): z_C – z-координата центра проекции, расположенного на оси z; $z_б$ и $z_д$ – координаты соответственно ближней и дальней граней по оси z; $x_л$ и $x_п$ – координаты соответственно левой и правой граней по оси x при $z = z_д$ (см. рис.8.4а); $y_н$ и $y_в$ – координаты соответственно нижней и верхней граней по оси y при $z = z_д$ (см. рис.8.4б).

Для анализа положения конца отрезка относительно левой, правой, нижней и верхней граней отсекающей пирамиды используются так называемые пробные функции. Идея их составления довольно проста. Уравнения плоскостей, несущих указанные грани, будут выглядеть соответственно так (см. рис.8.4):



а *б*

Рис.5.3. Объемные отсекатели регулярной формы: прямоугольный параллелепипед (а) и пирамида видимости (б)

$$x - x_л \frac{z_C - z}{z_C - z_д} = 0; \quad x - x_п \frac{z_C - z}{z_C - z_д} = 0;$$

$$y - y_н \frac{z_C - z}{z_C - z_д} = 0; \quad y - y_в \frac{z_C - z}{z_C - z_д} = 0.$$

Пробные функции применительно к этим граням совпадают с левыми частями приведенных уравнений:

$$f_л = x - x_л \frac{z_C - z}{z_C - z_д}; \quad f_п = x - x_п \frac{z_C - z}{z_C - z_д};$$

$$f_н = y - y_н \frac{z_C - z}{z_C - z_д}; \quad f_в = y - y_в \frac{z_C - z}{z_C - z_д}.$$

Значение пробной функции после подстановки в нее координат конца отрезка **(*x, y, z*)** позволяет определить положение этого конца относительно соответствующей грани и, в свою очередь, заполнить надлежащим образом соответствующий бит концевого кода. При внутреннем отсечении это осуществляется так.

Если $f_л < 0$, конец расположен левее плоскости левой грани, и в первый бит концевого кода заносится ***1***; если же $f_л = 0$ или $f_л > 0$, конец расположен соответственно на плоскости левой грани или правее нее, и в первый бит заносится ***0***.

Если $f_п > 0$, конец расположен правее плоскости правой грани, и во второй бит концевого кода заносится ***1***; если же $f_п = 0$ или $f_п < 0$, конец расположен соответственно на плоскости правой грани или левее нее, и во второй бит заносится ***0***.

Если $f_н < 0$, конец расположен ниже плоскости нижней грани, и в третий бит концевого кода заносится ***1***; если же $f_н = 0$ или $f_н > 0$, конец расположен соответственно на плоскости нижней грани или выше нее, и в третий бит заносится ***0***.

Если $f_в > 0$, конец расположен выше плоскости верхней грани, и в четвертый бит концевого кода заносится ***1***; если же $f_в = 0$ или $f_в < 0$, конец расположен соответственно на плоскости верхней грани или ниже нее, и в четвертый бит заносится ***0***.

Аналогичный подход применим и для заполнения пятого и шестого битов концевых кодов с учетом положения концов отрезка относительно ближней и

дальней граней. Уравнения плоскостей, проходящих через эти грани, выглядят соответственно так:

$$z - z_б = 0; \quad z - z_д = 0.$$

Вид соответствующих пробных функций очевиден:

$$f_б = z - z_б; \quad f_д = z - z_д.$$

Если после подстановки в первую из них ***z***-координаты конца отрезка $f_б > 0$, конец расположен перед плоскостью ближней грани, и в пятый бит концевого кода заносится ***1***; если же $f_б = 0$ или $f_б < 0$ конец расположен соответственно на плоскости ближней грани или за ней, и в пятый бит заносится ***0***.

Если после такой же подстановки во вторую пробную функцию оказывается, что $f_д < 0$, конец расположен за плоскостью дальней грани, и в шестой бит концевого кода заносится ***1***; если же $f_д = 0$ или $f_д > 0$ конец расположен соответственно на плоскости дальней грани или перед ней, и в шестой бит заносится ***0***.

Таким образом, если отсечение осуществляется прямоугольным параллелепипедом, то все координатное пространство разбивается на двадцать семь областей, в каждой из которых концу отрезка присваивается уникальный код.

Если отсекателем является усеченная пирамида, координатное пространство разбивается в общем случае на тридцать шесть областей с уникальным кодом конца отрезка в каждой из них: двадцать семь областей – спереди от центра проекции, т.е. перед точкой наблюдения, и девять – позади от центра проекции, т.е. за точкой наблюдения.

Вместе с тем, представляется целесообразным использовать иной подход к решению этой задачи. Все, что находится позади от центра проекции, наблюдатель видеть не может. Поэтому, независимо от того, какое именно отсечение реализуется – внутреннее или внешнее, начать обработку трехмерной сцены можно с удаления ее части, расположенной за центром проекции, т.е., иными словами, в полупространстве $z > z_C$ (здесь и далее принадлежность точки плоскости $z = z_C$ приравнивается случаю ее расположения перед этой плоскостью). Применительно к конкретному отрезку с началом в точке $P_1(x_1, y_1, z_1)$ и концом в точке $P_2(x_2, y_2, z_2)$ это можно сделать, используя данную плоскость в качестве отсекающей (по аналогии с работой бесконечной отсекающей прямой, проходящей через ребро отсекающего прямоугольника, в алгоритме двумерного отсечения Сазерленда-Коэна).

Последовательность действий при этом должна быть следующей. Сначала следует проверить, не принадлежит ли весь отрезок полупространству $z > z_C$. Если это подтверждается (***z***-координаты концов отрезка

удовлетворяют неравенствам $z_1 > z_C$, $z_2 > z_C$), отрезок идентифицируется как полностью невидимый, и обработка завершается. В противном случае проверяется, не расположен ли отрезок целиком в доступном наблюдению замкнутом полупространстве $z \le z_C$. Если это так ($z_1 \le z_C$, $z_2 \le z_C$), можно приступать к основной обработке (к отсечению отрезка пирамидой видимости), если нет (что на данном этапе проверки возможно в случаях, когда $z_1 > z_C$, $z_2 \le z_C$ или $z_1 \le z_C$, $z_2 > z_C$) – надо найти точку пересечения отрезка с плоскостью $z = z_C$ и перенести в нее тот конец отрезка, который оказался позади от центра проекции. Координаты точки пересечения определяются путем подстановки в уравнение прямой, проведенной через отрезок, т.е. в уравнение вида

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} = \frac{z - z_1}{z_2 - z_1},$$

значения $z = z_C$.

Введем обозначения (все они будут использоваться и далее в рамках настоящего раздела):

$$m_{xy} = (x_2 - x_1)/(y_2 - y_1); \quad m_{xz} = (x_2 - x_1)/(z_2 - z_1);$$
$$m_{yz} = (y_2 - y_1)/(z_2 - z_1).$$

Используя два из них, запишем решения приведенного выше уравнения относительно координат искомой точки пересечения в следующем виде:

$$x = m_{xz}(z_C - z_1) + x_1; \quad y = m_{yz}(z_C - z_1) + y_1; \quad z = z_C.$$

Если в результате описанных действий отрезок не идентифицирован как целиком невидимый, его следует подвергнуть дальнейшей обработке традиционным для алгоритма Сазерленда-Коэна образом. При этом концам отрезка, каждый из которых гарантированно попадает в одну из двадцати семи областей перед центром проекции, присваиваются вполне адекватные концевые коды (как и в случае отсечения отрезка прямоугольным параллелепипедом).

При изложенном подходе последующие действия алгоритма трехмерного отсечения Сазерленда-Коэна при обеих возможных формах отсекателя будут полностью аналогичны тем, которые осуществляются в соответствующем алгоритме двумерного отсечения. Приведем их краткое описание.

Если после определения концевых кодов выясняется, что оба они равны нулю, отрезок полагается видимым полностью, и обработка заканчивается. Если это не подтверждается, то вычисляется побитовое логическое произведение концевых кодов. При отличном от нуля значении этого произведения отрезок идентифицируется как безусловно невидимый, что также приводит к завершению обработки. При полностью нулевом побитовом логическом произведении отрезок может оказаться как частично видимым, так и полностью невидимым. Дальнейшая его обработка связана с поочередной работой шести бесконечных отсекающих плоскостей, несущих

соответственно левую, правую, нижнюю, верхнюю, ближнюю и дальнюю грани объемного отсекателя (по аналогии с работой четырех бесконечных отсекающих прямых, проходящих через ребра отсекающего прямоугольника а двумерном алгоритме отсечения). В каждом случае проверяется, не лежит ли отрезок полностью в той же стороне от отсекающей плоскости, что и сам отсекатель. Если это так, переходят к следующей отсекающей плоскости. В противном случае проверяют, не лежит ли начало отрезка в той же стороне, что и отсекатель. Если это подтверждается, начало и конец отрезка меняют местами. Начало отрезка в любом случае оказывается по отношению к отсекающей плоскости в стороне, противоположной той, в которой находится отсекатель. После определения точки пересечения прямой, проведенной через отрезок, с текущей отсекающей плоскостью начало отрезка переносится в эту точку (удаляется часть отрезка). Полученный новый отрезок вновь проверяется на полную видимость и безусловную невидимость. В зависимости от результата переходят к следующей отсекающей плоскости или алгоритм завершает работу.

Алгоритм двумерного отсечения Кируса-Бека

Данный алгоритм реализует отсечение отрезков произвольным выпуклым многоугольником. В его основу положено параметрическое представление отрезка.

Параметрическая форма записи уравнения отрезка с концами P_1 и P_2 имеет следующий вид:

$$P(t) = P_1 + (P_2 - P_1)t,$$

где t – параметр, принимающий значения в диапазоне $0 \le t \le 1$.

В декартовой системе координат приведенное выше уравнение сводится к двум одномерным параметрическим уравнениям:

$$x(t) = x_1 + (x_2 - x_1)t; \quad y(t) = y_1 + (y_2 - y_1)t;$$

здесь x_1, y_1 и x_2, y_2 – координаты соответственно начала и конца отрезка.

На координатной плоскости xy точкам начала и конца отрезка можно поставить в соответствие векторы (рис.8.5):

$$\overline{P_1} = x_1\bar{i} + y_1\bar{j}; \quad \overline{P_2} = x_2\bar{i} + y_2\bar{j};$$

в приведенных выражениях $\bar{i}$ и $\bar{j}$ – это единичные векторы (орты), направленные вдоль осей x и y соответственно.

Для вектора, связанного с произвольной точкой $P(t)$ отрезка, выражение в параметрическом виде запишется так:

$$\overline{P}(t) = \overline{P_1} + (\overline{P_2} - \overline{P_1})t = [x_1 + (x_2 - x_1)t]\bar{i} + [y_1 + (y_2 - y_1)t]\bar{j}.$$

Точке f (расположенной, например, на границе 1 отсекателя в одной из сопряженных с этой границей вершин, см. рис.8.5) с координатами f_x и f_y будет соответствовать вектор

$$\overline{f} = f_x \overline{i} + f_y \overline{j} .$$

Ориентацию же точки $P(t)$ относительно точки f можно характеризовать вектором

$$\overline{P}(t) - \overline{f} = \left[x_1 + (x_2 - x_1)t - f_x\right]\overline{i} + \left[y_1 + (y_2 - y_1)t - f_y\right]\overline{j} .$$

О расположении произвольной точки $P(t)$, принадлежащей отрезку P_1P_2, по отношению к какой-либо границе (например, опять к границе 1, см. рис.8.5), можно судить по величине скалярного произведения

$$\overline{n} \cdot \left[\overline{P}(t) - \overline{f}\right],$$

где $\overline{n}$ – вектор внутренней (т.е. направленной внутрь отсекателя) нормали к данной границе.

Если это произведение больше или меньше нуля, то точка $P(t)$ расположена относительно границы, точнее – относительно бесконечной прямой, проходящей через границу, соответственно в той же стороне, что и сам отсекатель, или в противоположной (рис.8.5). И только в том случае, когда это произведение равно нулю, точка $P(t)$ лежит непосредственно на указанной прямой и является возможной точкой пересечения (рис.8.5).

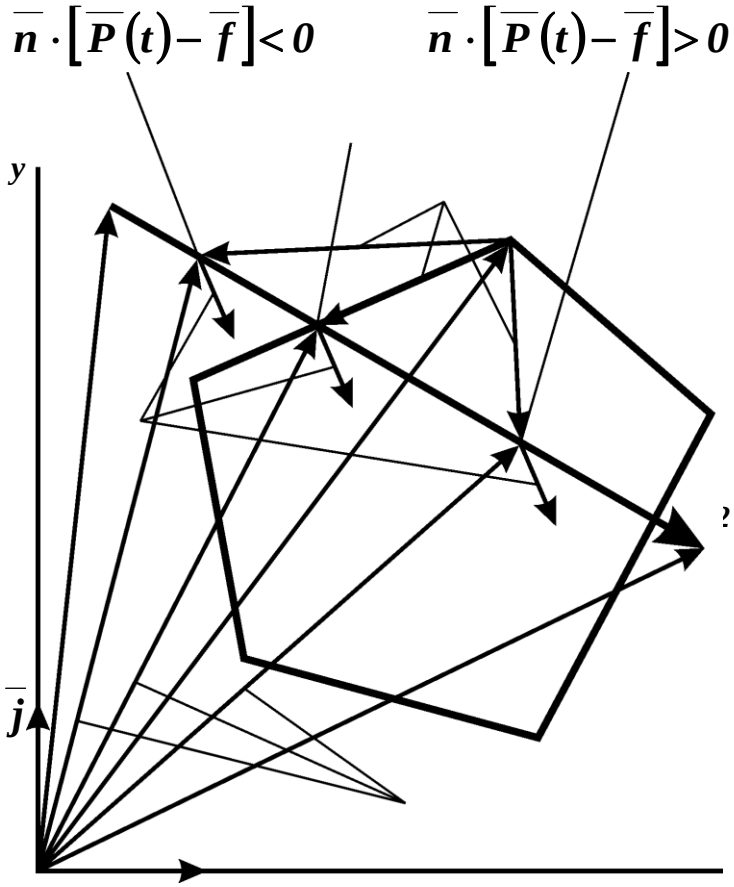


Рис.5.4. К пониманию основ алгоритма Кируса-Бека

Поэтому в алгоритме Кируса-Бека реальные точки пересечения отрезка с границами отсекающего многоугольника ищутся среди решений уравнений вида

$$\overline{n_i} \cdot \left[\overline{P}(t_i) - \overline{f_i}\right] = 0 ;$$

здесь $\overline{n_i}$ – внутренняя нормаль к i-ой к границе многоугольника, $\overline{f_i}$ – вектор, соответствующий принадлежащей этой границе

точке f_i, t_i – значение параметра t в возможной точке пересечения отрезка с i-ой границей.

После подстановки в это уравнение выражения для вектора $\overline{P}(t)$ в параметрическом виде оно будет выглядеть так:

$$\overline{n_i} \cdot \left[\overline{P_1} + \left(\overline{P_2} - \overline{P_1}\right) t_i - \overline{f_i}\right] = \overline{n_i} \cdot \left(\overline{P_2} - \overline{P_1}\right) t_i + \overline{n_i} \cdot \left(\overline{P_1} - \overline{f_i}\right) = 0.$$

Введем обозначения: $\overline{D} = \overline{P_2} - \overline{P_1}$ – вектор, называемый директрисой отрезка и соединяющий начало отрезка с его концом; $\overline{W_i} = \overline{P_1} - \overline{f_i}$ – вектор, соединяющий точку f_i с началом отрезка.

Используя вновь введенные векторы, преобразуем последнее уравнение и решим его в общем виде относительно t_i:

$$t_i = -\frac{\overline{W_i} \cdot \overline{n_i}}{\overline{D} \cdot \overline{n_i}}.$$

Полученное выражение позволяет рассчитывать значения параметра отрезка в возможных точках пересечения с границами отсекающего многоугольника при условии, что $\overline{D} \cdot \overline{n_i} \neq 0$ (случай $\overline{D} \cdot \overline{n_i} = 0$ является особым и будет рассмотрен ниже). Если при внутреннем отсечении найденное значение t_i выходит за пределы интервала $0 \le t_i \le 1$, то его следует проигнорировать (как значения $t_1 < 0$ и $t_4 > 1$ для отрезка P_1P_2 на рис.8.6), т.к. соответствующая точка пересечения, несмотря на то, что она лежит на прямой, проходящей через отрезок, самому отрезку не принадлежит. Вместе с тем, количество полученных значений t_i, принадлежащих указанному интервалу, может оказаться больше двух (t_2, t_3, t_5, t_6 и t_7 для отрезка P_1P_2 на рис.8.6). В этом случае среди группы точек с $\overline{D} \cdot \overline{n_i} > 0$ – группы нижнего

предела (t_2 и t_3 на рис.8.6), расположенных ближе к началу отрезка, следует выбрать точку с максимальным значением параметра, т.к. именно она является реальной точкой пересечения и определяет так называемый нижний предел $t_н$ ($t_н = max(t_2, t_3) = t_2$ на рис.8.6). Среди группы точек с $\overline{D} \cdot \overline{n_i} < 0$ – группы верхнего предела (t_5, t_6 и t_7 на рис.8.6), лежащих ближе к концу отрезка, надо выбрать точку с минимальным значением параметра, т.к. она является реальной точкой пересечения и определяет верхний предел ($t_в = min(t_5, t_6, t_7) = t_6$ на рис.8.6). В случае, когда $t_н$ и $t_в$ определены, т.е. выявлены две реальные точки пересечения, часть отрезка между этими точками будет видима, и ее следует визуализировать (здесь и далее полагается, что реализуется внутреннее отсечение).

Алгоритм трехмерного отсечения Кируса-Бека

Данный алгоритм осуществляет отсечение отрезков произвольным выпуклым многогранником. По сути, он является полным аналогом соответствующего двумерного алгоритма. В нем применяются абсолютно те же подходы, что и в двумерном варианте, но распространенные на трехмерный случай. Понятие «граница (или сторона, или ребро) отсекающего многоугольника», используемое в двумерном алгоритме, в трехмерном алгоритме трансформируется в понятие «граница (или грань) отсекающего многогранника». Бесконечные прямые, проведенные через ребра многоугольника, заменяются бесконечными плоскостями, несущими грани многогранника. Все векторы в трехмерной версии алгоритма имеют не две, а три компоненты: вдоль координатных осей x, y и z. В выражениях для векторов

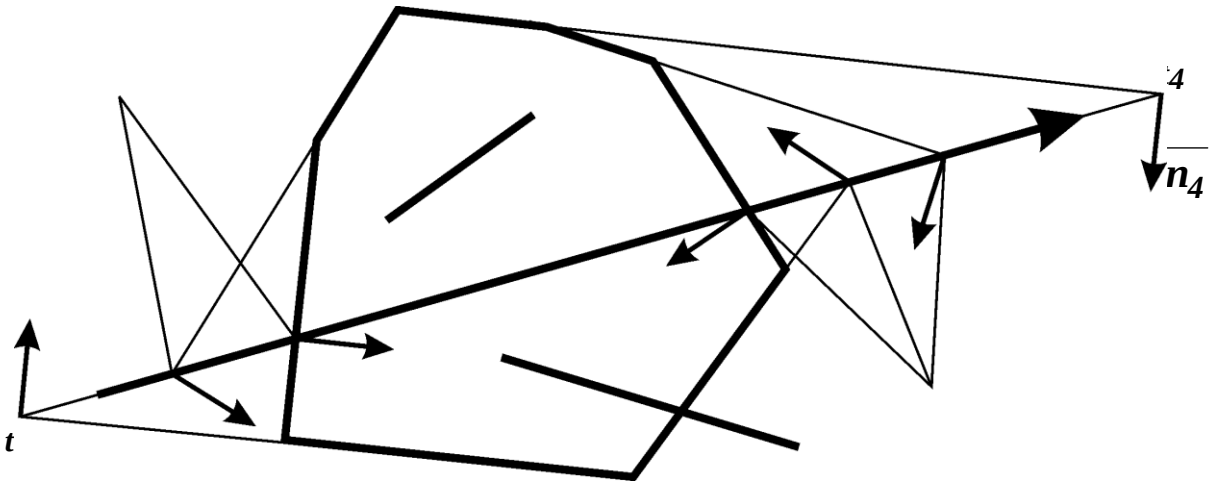


Рис.5.5. К определению нижнего и верхнего пределов параметра t

при записи третьей компоненты используется единичный вектор (орт) $\overline{k}$, направленный вдоль оси ***z***.

При этом, однако, надо учитывать следующее. В трехмерной версии ***значение M должно соответствовать*** числу граней отсекающего многогранника. Подпрограмма расчета скалярного произведения векторов реализуется по аналогии с тем, как это сделано в двумерном алгоритме, но векторы $\overline{Вект1}$ ***и*** $\overline{Вект2}$ ***з***адаются тремя компонентами (по осям ***x***, ***y*** и ***z*): *Вект1x, Вект1y, Вект1z и Вект2x, Вект2y, Вект2z соответственно.***

Рассмотрим пример работы трехмерного алгоритма Кируса-Бека при внутреннем отсечении отрезка P_1P_2 с координатами начала и конца соответственно **(*–2, –1, 1/2*)** и **(*3/2, 3/2, –1/2*)** кубом ($M=6$) с гранями, заданными координатами $x_л=-1$, $x_п=1$, $y_н=-1$, $y_в=1$, $z_б=1$, $z_д=-1$ (рис.8.7).

$t_н=0$

$t_в=1$

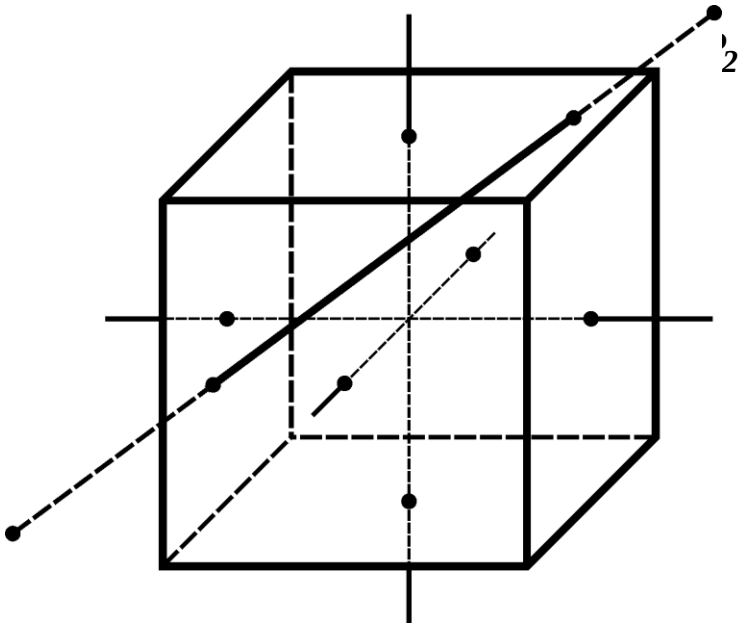


Рис.5.6. Пример работы трехмерного алгоритма Кируса-Бека

$$\overline{D}=\overline{P_2}-\overline{P_1}=\left(\frac{3}{2}\overline{i}+\frac{3}{2}\overline{j}-\frac{1}{2}\overline{k}\right)-\left(-2\overline{i}-\overline{j}+\frac{1}{2}\overline{k}\right)=\left(\frac{7}{2}\overline{i}+\frac{5}{2}\overline{j}-\overline{k}\right)$$

При проведении расчетов полагалось, что в главном цикле грани обрабатываются в следующей последовательности: левая ($i=1$), правая ($i=2$), нижняя ($i=3$), верхняя ($i=4$), ближняя ($i=5$), дальняя ($i=6$). Причем при обработке левой, нижней и дальней граней в качестве точки ***f*** использовалась вершина куба с координатами **(*–1, –1, –1*)**, а при обработке правой, верхней и ближней граней – вершина куба с координатами **(*1, 1, 1*)**. Векторы внутренних нормалей задавались по геометрическим соображениям.

Таким образом, отрезок частично видим в интервале изменения значений параметра от $t_н = 2/7$ *до* $t_в = 4/5$. ***Подстановка этих значений в одномерные*** параметрические уравнения отрезка дает координаты точек пересечения, соответствующих нижнему и верхнему пределам (см. также рис.8.7):

$$x(t_н) = -2 + (3/2 + 2) \cdot 2/7 = -1,$$

$$y(t_н) = -1 + (3/2 + 1) \cdot 2/7 = -2/7,$$

$$z(t_н) = 1/2 + (-1/2 - 1/2) \cdot 2/7 = 3/14;$$

$$x(t_в) = -2 + (3/2 + 2) \cdot 4/5 = 4/5,$$

$$y(t_в) = -1 + (3/2 + 1) \cdot 4/5 = 1,$$

$$z(t_в) = 1/2 + (-1/2 - 1/2) \cdot 4/5 = -3/10.$$

До начала работы трехмерного алгоритма Кируса-Бека необходимо убедиться в выпуклости отсекающего многогранника, а также определить внутренние нормали к его границам (граням).

Особенности реализации внешнего и комбинированного отсечений

Задачи внутреннего отсечения легко адаптировать к решению задач внешнего отсечения. Для этого достаточно учесть только то обстоятельство, что результаты внешнего отсечения могут быть получены путем инвертирования (обращения) результатов внутреннего отсечения: отрезки или их части, которые при внутреннем отсечении определяются как видимые, при внешнем отсечении на экран не выводятся, и, наоборот, отрезки или их части, которые алгоритмом внутреннего отсечения игнорируются, при внешнем отсечении идентифицируются как видимые и визуализируются.В ряде случаев может возникнуть необходимость осуществления комбинации внутреннего и внешнего отсечений, т.е. комбинированного отсечения. Характерный пример подобного отсечения наблюдается на экране компьютерного дисплея при многооконном режиме его работы под управлением операционной системы Windows (рис.8.8): содержимое рабочей области пассивного окна 3 подвергается внутреннему отсечению прямоугольником, образованным границами этой области; само окно 3 со всем содержимым подвергается внешнему отсечению окнами 1 и 2, имеющими приоритеты выше, чем окно 3. Комбинированное отсечение позволяет реализовать также внутрен
нее или внешнее отсечение графических объектов невыпуклыми (частично вогнутыми) отсекателями. При этом в качестве базового используется соответствующий алгоритм двумерного или трехмерного отсечения Кируса-Бека.

Тема 6.Удаление невидимых поверхностей и линий

Удаление невидимых поверхностей и линий

Задача удаления невидимых линий и поверхностей является одной из наиболее интересных и сложных в компьютерной графике. Алгоритмы удаления заключаются в определении линий ребер, поверхностей или объемов, которые видимы или невидимы для наблюдателя, находящегося в заданной точке пространства.

Изображение без удаления невидимых линий воспринимается неоднозначно.

Рис. 6.1. Неоднозначность восприятия изображения куба

Сложность задачи удаления невидимых линий и поверхностей привела к появлению большого числа различных способов ее решения. Многие из них ориентированы на специализированные приложения. Единого (общего) решения этой задачи, годного для различных случаев, естественно, не существует: для каждого случая выбирается наиболее подходящий метод. Например, для моделирования процессов в реальном времени требуются быстрые алгоритмы, в то время как для формирования сложного реалистического изображения, в котором представлены тени, прозрачность и фактура, учитывающие эффекты отражения и преломления цвета в мельчайших оттенках, фактор времени выполнения уже не так существенен. Подобные алгоритмы работают медленно. Существует тесная взаимосвязь между скоростью работы алгоритма и детальностью его результата. Ни один из алгоритмов не может достигнуть хороших оценок для этих двух показателей одновременно. По мере создания все более быстрых алгоритмов можно строить все более детальные изображения. Реальные задачи, однако, всегда будут требовать учета еще большего количества деталей.

Все алгоритмы такого рода так или иначе включают в себя сортировку, причем главная сортировка ведется по геометрическому расстоянию от тела, поверхности, ребра или точки до точки наблюдения или картинной плоскости. Основная идея, положенная в основу сортировки по расстоянию, заключается в том, что чем дальше расположен объект от точки наблюдения, тем больше вероятность, что он будет полностью или частично заслонен одним из объектов, более близких к точке наблюдения. После определения расстояний или приоритетов по глубине остается провести сортировку по горизонтали и

по вертикали, чтобы выяснить, будет ли рассматриваемый объект действительно заслонен объектом, расположенным ближе к точке наблюдения. Эффективность любого алгоритма удаления в значительной мере зависит от эффективности процесса сортировки.

Алгоритмы удаления невидимых линий или поверхностей можно классифицировать по способу выбора системы координат или пространства, в котором они работают. Алгоритмы, работающие в объектном пространстве, имеют дело с мировой системой координат, в которой описаны эти объекты. При этом получаются весьма точные результаты, ограниченные, вообще говоря, лишь погрешностью вычислений. Полученные изображения можно свободно масштабировать. Алгоритмы, работающие в объектном пространстве, особенно полезны в тех приложениях, где необходима высокая точность. Алгоритмы же, работающие в пространстве изображения, имеют дело с системой координат того экрана, на котором объекты визуализируются. При этом точность вычислений ограничена разрешающей способностью экрана.

Удаление нелицевых граней многогранника Алгоритм Робертса

Этот алгоритм, предложенный в 1963 г., является первой разработкой такого рода и предназначен для удаления невидимых линий при штриховом изображении объектов, составленных из выпуклых многогранников. Он относится к алгоритмам, работающим в объектном пространстве, и очень элегантен с математической точки зрения. В нем очень удачно сочетаются геометрические методы и методы линейного программирования.

Выпуклый многогранник однозначно определяется набором плоскостей, образующих его грани, поэтому исходными данными для алгоритма являются многогранники, заданные списком своих граней. Грани задаются в виде плоскостей, заданных в канонической форме в объектной системе координат: $ax + by + cz + d = 0$.

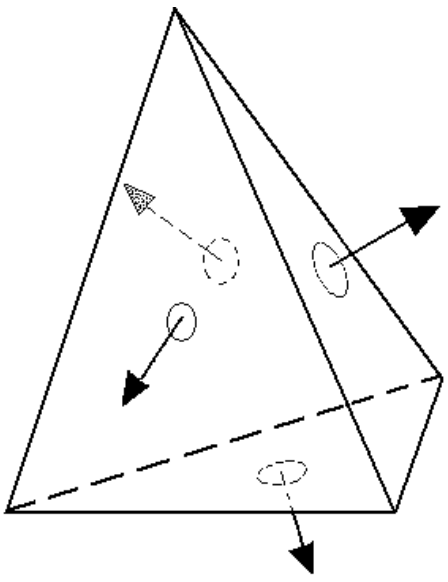


Рис. 6.2. Внешние нормали тетраэдра

Таким образом, каждая плоскость определяется четырехмерным вектором $\overrightarrow{P}$, а каждая точка $\overrightarrow{r}$, заданная в однородных координатах, также представляет собой четырехмерный вектор:

$$\overrightarrow{P} = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}, \overrightarrow{r} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}.$$

Принадлежность точки плоскости можно установить с помощью скалярного произведения, т.е. если $(\overrightarrow{P} \cdot \overrightarrow{r}) = 0$, то точка принадлежит плоскости, если же нет, то знак произведения показывает, по какую сторону от плоскости эта точка находится. Из векторов плоскостей строится прямоугольная матрица порядка $4 \times n$, которая называется **обобщенной матрицей описания многогранника**:

$$M = \begin{pmatrix} a_1 & a_2 & a_3 & \dots & a_n \\ b_1 & b_2 & b_3 & \dots & b_n \\ c_1 & c_2 & c_3 & \dots & c_n \\ d_1 & d_2 & d_3 & \dots & d_n \end{pmatrix}.$$

Умножая столбцы матрицы на вектор $\overrightarrow{r}$, получим *n*-мерный вектор, и если все его компоненты неотрицательны, то точка принадлежит многограннику. Это условие будем записывать в виде $(\overrightarrow{r} \cdot M) \geq 0$ (имеется в виду умножение вектор-строки на матрицу).

В своем алгоритме Робертс рассматривает только отрезки, являющиеся пересечением граней многогранника.

Из обобщенной матрицы можно получить информацию о том, какие грани многогранника пересекаются в вершинах. Действительно, если вершина $\overrightarrow{v} = (x, y, z, 1)$ принадлежит граням $\overrightarrow{P}_1, \overrightarrow{P}_2, \overrightarrow{P}_3$, то она удовлетворяет уравнениям

$$\left.\begin{aligned} (\overrightarrow{v} \cdot \overrightarrow{P}_1) &= 0 \\ (\overrightarrow{v} \cdot \overrightarrow{P}_2) &= 0 \\ (\overrightarrow{v} \cdot \overrightarrow{P}_3) &= 0 \\ (\overrightarrow{v} \cdot \overrightarrow{e}_4) &= 1 \end{aligned}\right\}, \quad \text{где} \quad \overrightarrow{e}_4 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Эту систему можно записать в матричном виде:

$$\overrightarrow{v} \cdot Q = \overrightarrow{e}_4,$$

где Q - матрица, составленная из вектор-столбцов $\overrightarrow{P}_1, \overrightarrow{P}_2, \overrightarrow{P}_3, \overrightarrow{e}_4$. Значит, координаты вершины определяются соотношением

$$\overrightarrow{v} = \overrightarrow{e}_4 \cdot Q^{(-1)},$$

т.е. они составляют последнюю строку обратной матрицы. А это означает, что если для каких-либо трех плоскостей обратная матрица существует, то плоскости имеют общую вершину.

Алгоритм прежде всего удаляет из каждого многогранника те ребра или грани, которые экранируются самим телом. Робертс использовал для этого простой тест: если одна или обе смежные грани обращены своей внешней поверхностью к наблюдателю, то ребро является видимым. Тест этот выполняется вычислением скалярного произведения координат наблюдателя на вектор внешней нормали грани: если результат отрицательный, то грань видима.

Алгоритм Варнока

В отличие от алгоритма Робертса, Варнок в 1968 г. предложил алгоритм, работающий не в объектном пространстве, а в пространстве образа. Он также нацелен на изображение многогранников, а главная идея его основана на гипотезе о способе обработки информации, содержащейся в сцене, глазом и мозгом человека. Эта гипотеза заключается в том, что тратится очень мало времени и усилий на обработку тех областей, которые содержат мало информации. Большая часть времени и труда затрачивается на области с высоким информационным содержимым. Так, например, рассматривая помещение, в котором имеется только картина на стене, мы быстро осматриваем стены, пол и потолок, а затем все внимание сосредоточиваем на картине. В свою очередь, на этой картине, если это портрет, мы бегло отмечаем фон, а затем более внимательно рассматриваем лицо изображенного персонажа, в особенности глаза, губы. Как правило, достаточно детально рассматриваются еще и руки и с чуть меньшим вниманием - одежда.

В алгоритме Варнока и его вариантах делается попытка воспользоваться тем, что большие области изображения однородны. Такое свойство называют **когерентностью**, имея в виду, что смежные области (пиксели) вдоль обеих осей x и y имеют тенденцию к однородности.

В пространстве изображения рассматривается окно и решается вопрос о том, пусто ли оно, или его содержимое достаточно просто для визуализации. Если это не так, то окно разбивается на фрагменты до тех пор, пока содержимое фрагмента не станет достаточно простым для визуализации или его размер не достигнет требуемого предела разрешения. В последнем случае информация, содержащаяся в окне, усредняется, и результат изображается с одинаковой интенсивностью или цветом.

Конкретная реализация алгоритма Варнока зависит от метода разбиения окна и от деталей критерия, используемого для того, чтобы решить, является ли содержимое окна достаточно простым. В оригинальной версии алгоритма каждое окно разбивалось на четыре одинаковых подокна. Многоугольник, входящий в изображаемую сцену, по отношению к окну будем называть

- **внешним**, если он целиком находится вне окна;
- **внутренним**, если он целиком расположен внутри окна;
- **пересекающим**, если он пересекает границу окна;
- **охватывающим**, если окно целиком расположено внутри него.

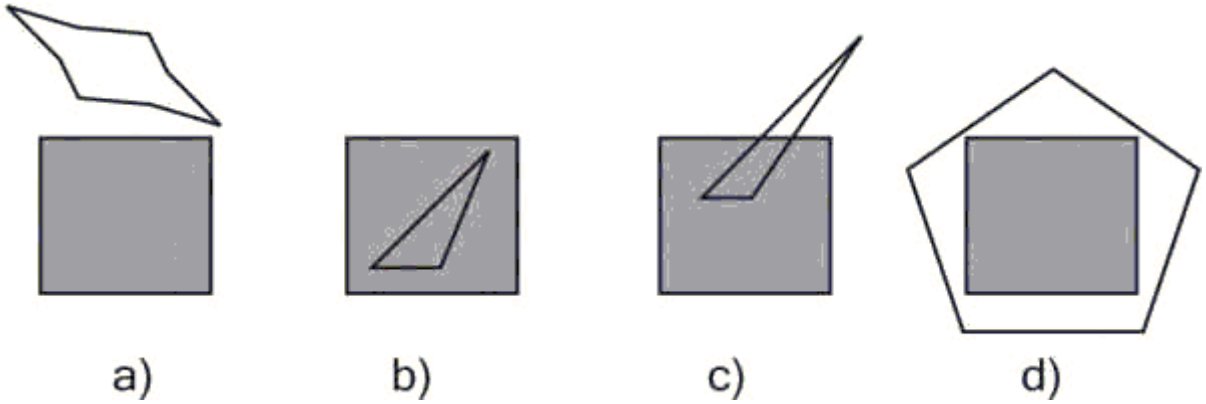


Рис. 6.3. Варианты расположения многоугольника по отношению к окну

Теперь можно в самом общем виде описать алгоритм.

Для каждого окна:

1. Если все многоугольники сцены являются внешними по отношению к окну, то оно пусто; изображается фоновым цветом и дальнейшему разбиению не подлежит.
2. Если только один многоугольник сцены имеет общие точки с окном и является по отношению к нему внутренним, то окно заполняется фоновым цветом, а сам многоугольник заполняется своим цветом.
3. Если только один многоугольник сцены имеет общие точки с окном и является по отношению к нему пересекающим, то окно заполняется фоновым цветом, а часть многоугольника, принадлежащая окну, заполняется цветом многоугольника.
4. Если только один многоугольник охватывает окно и нет других многоугольников, имеющих общие точки с окном, то окно заполняется цветом этого многоугольника.
5. Если существует хотя бы один многоугольник, охватывающий окно, то среди всех таких многоугольников выбирается тот, который расположен ближе всех многоугольников к точке наблюдения, и окно заполняется цветом этого многоугольника.
6. В противном случае производится новое разбиение окна.

Шаги 1–4 рассматривают ситуацию пересечения окна только с одним многоугольником. Они используются для сокращения числа подразбиений. Шаг 5 решает задачу удаления невидимых поверхностей. Многоугольник, находящийся ближе всех к точке наблюдения, экранирует все остальные.

Для реализации алгоритма необходимы функции, определяющие взаимное расположение окна и многоугольника, которые достаточно легко реализуются в случае прямоугольных окон и выпуклых многоугольников.

Алгоритм Вейлера-Азертона

Вейлер и Азертон попытались оптимизировать алгоритм Варнока в отношении числа выполняемых разбиений, перейдя от прямоугольных разбиений к разбиениям вдоль границ многоугольников (1977). Для этого они

использовали алгоритм отсечения многоугольников. Алгоритм работает в объектном пространстве, и результатом его работы являются многоугольники. В самом общем виде он состоит из четырех шагов.

1. Предварительная сортировка по глубине.
2. Отсечение по границе ближайшего к точке наблюдения многоугольника, называемое сортировкой многоугольников на плоскости.
3. Удаление многоугольников, экранируемых более близкими к точке наблюдения многоугольниками.
4. Если требуется, то рекурсивное разбиение и новая сортировка.

В процессе предварительной сортировки создается список приблизительных приоритетов, причем близость многоугольника к точке наблюдения определяется расстоянием до ближайшей к ней вершины. Затем выполняется отсечение по самому первому из многоугольников. Отсечению подвергаются все многоугольники из списка, причем эта операция выполняется над проекциями многоугольников на картинную плоскость. При этом создаются списки внешних и внутренних фигур. Все попавшие в список внешних не экранируются отсекающим многоугольником. Затем рассматривается список внутренних многоугольников и выполняется сортировка по расстоянию до отсекающего многоугольника. Если все вершины некоторого многоугольника оказываются дальше от наблюдателя, чем самая удаленная из вершин экранирующего, то они невидимы, и тогда они удаляются. После этого работа алгоритма продолжается с внешним списком.

Если какая-то из вершин внутреннего многоугольника оказывается ближе к наблюдателю, чем ближайшая из вершин экранирующего многоугольника, то такой многоугольник является частично видимым. В этом случае предварительный список приоритетов некорректен, и тогда в качестве нового отсекающего многоугольника выбирается именно этот "нарушитель порядка". При этом используется именно исходный многоугольник, а не тот, что получился в результате первого отсечения. Такой подход позволяет минимизировать число разбиений.

Метод Z-буфера

Это относительно простой алгоритм удаления невидимых поверхностей. Идея ***z***-буфера схожа с идеей о буфере кадра. Буфер кадра используется для запоминания атрибутов (в том числе цвета) каждого пикселя в пространстве изображения, ***z***-буфер – для запоминания координаты ***z*** (или глубины) каждого видимого пикселя в пространстве изображения. При создании объемной сцены сначала в ***z***-буфер для всех пикселей заносится некоторое минимальное значение ***z*** (оно определяет наиболее удаленную от точки наблюдения границу этой сцены), а в буфер кадра – атрибуты фонового цвета. Далее в произвольном порядке обрабатываются участвующие в сцене фигуры (это, как правило, плоские многоугольники, в том числе плоские полигоны, совокупностями которых аппроксимируются поверхности объемных тел). При обработке очередной фигуры глубина каждого пикселя, принадлежащего формируемому изображению этой фигуры, сравнивается с глубиной, которая

для данного пикселя уже занесена в ***z***-буфер. Если это сравнение показывает, что обрабатываемая фигура в данной точке имеет меньшую глубину (расположена ближе к наблюдателю), чем имеющаяся в ***z***-буфере, то в обоих буферах производят изменения: в буфер кадра для соответствующего пикселя заносятся атрибуты цвета фигуры, а в ***z***-буфер – его глубина. Если же сравнение дает противоположный результат, то никаких действий не производится. По сути, алгоритм реализует поиск для каждого пикселя наибольшего значения $z(x, y)$.

Приведем формальное описание последовательности действий алгоритма, использующего *z*-буфер:

заполнить буфер кадра фоновым значением цвета;
заполнить ***z***-буфер минимальным значением ***z***;
в произвольном порядке обработать каждый многоугольник:
преобразовать его в растровую форму;
для каждого пикселя – ***Пиксель*** (x, y), принадлежащего многоугольнику, вычислить глубину $z(x, y)$ и сравнить ее со значением ***Z-буфер*** (x, y), хранящимся в ***z***-буфере в этой же позиции; если $z(x, y) > Z\text{-буфер}(x, y)$, то записать атрибуты многоугольника (цвет) в буфер кадра и заменить ***Z-буфер*** (x, y) на $z(x, y)$; в противном случае никаких действий не производить. При реализации алгоритма следует использовать также следующие соображения. Каждый многоугольник лежит в плоскости, которую можно описать уравнением вида

$ax + by + cz + d = 0$ или (при $c \neq 0$) $z = -(ax + by + d) / c$. За исключением ряда особых случаев расположения такой плоскости, многоугольники целесообразно обрабатывать построчно, переходя по горизонтали от пикселя к пикселю пошагово. На сканирующей строке $y = const$ поэтому на каждом таком шаге приращение (на Δx) получает только горизонтальная координата. Если пиксель с координатами (x_i, y) уже обработан и определена его глубина z_i, при переходе к следующему пикселю с координатами (x_{i+1}, y), где $x_{i+1} = x_i + \Delta x$, приращение глубины определяется выражением

$z_{i+1} - z_i = -(ax_i + a\Delta x + by + d) / c + (ax_i + by + d) / c = -(a / c)\Delta x$. С учетом же того, что $\Delta x = 1$, глубину очередного пикселя можно рассчитать по очень простой формуле:

$z_{i+1} = z_i - (a / c)$. Ограничимся приведенными соображениями по построению алгоритма удаления невидимых поверхностей с использованием ***z***-буфера.

Методы приоритетов (художника, плавающего горизонта)

Здесь мы рассмотрим группу методов, учитывающих специфику изображаемой сцены для удаления невидимых линий и поверхностей.

При изображении сцен со сплошным закрашиванием поверхностей можно воспользоваться **методом художника**: элементы сцены изображаются в последовательности от наиболее удаленных от наблюдателя к более близким. При экранировании одних участков сцены другими невидимые участки просто закрашиваются. Если вычислительная трудоемкость

получения изображения для отдельных элементов достаточно высока, то такой алгоритм будет не самым лучшим по эффективности, но зато мы избежим анализа (и вполне возможно, тоже дорогостоящего), позволяющего установить, какие же из элементов изображать не надо в силу их невидимости.

Данный алгоритм относится к группе алгоритмов, реализующих удаление невидимых линий и поверхностей. Чаще всего он используется для визуализации трехмерных поверхностей, описываемых функциями вида $F(x, y, z) = 0$. Главная идея заключается в сведении трехмерной задачи к двумерной: поверхность сечется параллельными и равноотстоящими друг от друга плоскостями, и на экран выводятся линии пересечения (рис.10.1). Так, например, при задании секущих плоскостей постоянными значениями z поверхность представляется совокупностью кривых, которые описываются функциями $y = f(x, z)$ при $z = const$, т.е. $y = f(x)$. Плоскости $z = const$ сортируют по степени их удаленности от точки наблюдения. Затем поочередно для каждой плоскости, начиная с ближней, строится лежащая на ней кривая. При этом для каждого из значений x, задаваемых в пространстве изображения (в координатной плоскости экрана) с шагом, равным расстоянию между пикселями (т.е. единице), определяется значение y. Если на текущей кривой при конкретном значении x соответствующее значение y больше или, наоборот, меньше, чем значения y для всех предыдущих кривых при том же значении x, то текущая кривая полагается видимой в данной точке; в противном случае она считается невидимой (на рис.10.1 невидимые участки кривых показаны пунктиром).

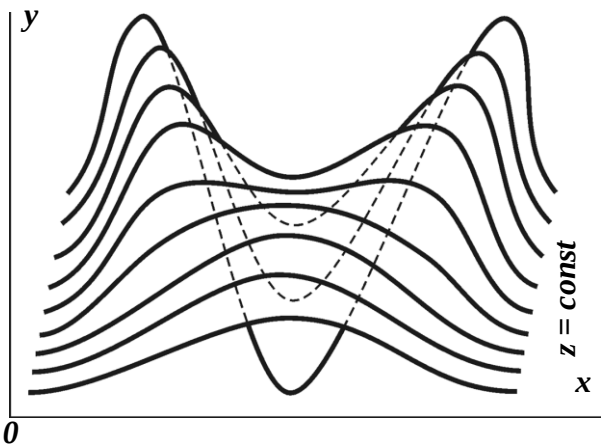


Рис.6.4 Кривые y = f (x) в секущих плоскостях z = const

Для хранения максимальных и минимальных значений y при каждом значении x используются два массива чисел: ***массив верхнего горизонта*** и ***массив нижнего горизонта*** (они отражают текущее состояние «верхнего горизонта» и «нижнего горизонта» соответственно). Очевидно, что по мере обработки кривых значения в первом массиве могут только увеличиваться («верхний горизонт» всплывает), а во втором – только уменьшаться («нижний горизонт» опускается). При использовании таких массивов текущая кривая в какой-либо точке полагается видимой, если при заданном значении x значение y либо больше соответствующего значения в массиве верхнего горизонта, либо меньше аналогичного значения в массиве нижнего горизонта. Если кривая в данной точке видима, значение y заносится в один из массивов на место прежнего элемента.

Приведем уравнение, соответствующее линейной интерполяции кривой между двумя известными точками с координатами (x_n, y_n) и (x_{n+k}, y_{n+k}) (здесь k – шаг задания точек по оси x):

$$y = (y_{n+k} - y_n)/(x_{n+k} - x_n) * (x - x_n) + y_n.$$

При изложенном подходе, когда обрабатываемая текущая кривая появляется слева или справа из-под множества ранее обработанных кривых, может появиться некорректность в изображении поверхности – ***эффект зазубренного ребра*** (на рис.10.2 приводящие к искажению фрагменты кривой показаны пунктиром). Избежать такого эффекта можно относительно простым приемом: при обработке каждой текущей кривой (за исключением первой) создаются ложные боковые ребра, соединяющие соответственно крайние левые и крайние правые точки текущей и предыдущей кривых (на рис.10.2 – штрихпунктирные линии). Ложные ребра на экран не выводятся, но ординаты точек на них при тех дискретных значениях ***x***, при которых ребра определены, участвуют в формировании массивов верхнего и нижнего горизонтов (при тех же условиях, что и сами кривые).

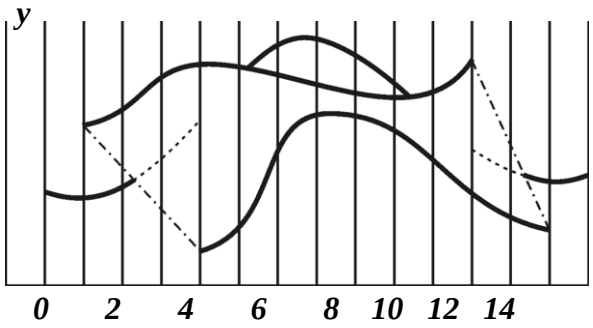


Рис.6.5.Устранение эффекта зазубренного ребра

С учетом приведенных соображений формальное описание последовательности действий алгоритма плавающего горизонта можно описать примерно так: поочередно для каждой из равноотстоящих друг от друга плоскостей ***z = const***, начиная с ближней от точки наблюдения:

- определить координаты первой (левой) точки P_n на лежащей в плоскости кривой $y = f(x)$;
- обработать левое боковое ребро: если точка P_n является первой точкой на первой кривой, то запомнить ее в качестве P_{n-1} и закончить обработку; в противном случае создать ребро, соединяющее P_{n-1} и P_n, занести, если это требуется, ординаты точек ребра в массивы верхнего и нижнего горизонтов и запомнить P_n в качестве P_{n-1};
- определить координаты остальных точек на кривой $y = f(x)$ при дискретном изменении x с единичным шагом, в том числе последней (правой) точки Q_n;
- если $y = f(x)$ является первой обрабатываемой кривой, занести значения y всех точек на ней для соответствующих x в массивы верхнего и нижнего горизонтов и закончить обработку, если нет, тогда: сравнить для каждой точки на кривой значение y с имеющимися в массивах верхнего и нижнего горизонтов (для соответствующего значения x); если y точки не меньше, чем в массиве верхнего горизонта, или не больше, чем в массиве нижнего горизонта, или значения в массивах отсутствуют, объявить точку видимой и занести y точки в соответствующий массив (или в оба массива); в противном случае объявить точку невидимой;

- обработать правое боковое ребро: если точка Q_n является последней точкой на первой кривой, то запомнить Q_n в качестве Q_{n-1} и закончить обработку; в противном случае создать ребро, соединяющее Q_{n-1} и Q_n, занести, если это требуется, ординаты точек ребра в массивы верхнего и нижнего горизонтов и запомнить Q_n в качестве Q_{n-1}.

Поясним, что ординаты боковых ребер заносятся в массив верхнего горизонта, только когда они больше, а в массив нижнего горизонта, только когда они меньше имеющихся там значений y для соответствующих значений x. Ординаты левого бокового ребра могут заноситься в массивы также в случае отсутствия в них каких-либо значений y для тех значений x, в которых определено ребро.

Алгоритмы построчного сканирования для криволинейных поверхностей

Идея построчного сканирования, предложенная в 1967 г. Уайли, Ромни, Эвансом и Эрдалом, заключалась в том, что сцена обрабатывается в порядке прохождения сканирующей прямой. В объектном пространстве это соответствует проведению секущей плоскости, перпендикулярной пространству изображения. Строго говоря, алгоритм работает именно в пространстве изображения, отыскивая точки пересечения сканирующей прямой с ребрами многоугольников, составляющих картину (для случая изображения многогранников). Но при пересечении очередного элемента рисунка выполняется анализ глубины полученной точки и сравнение ее с глубиной других точек на сканирующей плоскости. В некоторых случаях можно построить список ребер, упорядоченный по глубине, но зачастую это невозможно выполнить корректно. Поэтому сканирование сочетают с другими методами.

Метод двоичного разбиения пространства

Теперь разберем один способ использования метода художника при изображении пространственных сцен, содержащих несколько объектов или составные объекты. Это так называемый **метод двоичного разбиения пространства плоскостями**. Плоскости, как обычно, будут задаваться с помощью вектора нормали $\vec{n}$ и расстояния до начала координат d (с точностью до знака). Пусть изображаемая сцена состоит из набора непересекающихся граней $F_1, F_2, \ldots, F_n$ (они могут иметь общие прямолинейные участки границы). Проведем плоскость P_1, разбивающую все пространство на два полупространства, в одном из которых находится наблюдатель. Предположим, что плоскость при этом не пересекает ни одну из граней (но может содержать участок ее границы). Тогда грани, находящиеся в одном полупространстве с наблюдателем, могут заслонять от него часть граней из второго полупространства, но не наоборот. Это означает, что они должны изображаться позже. Разобьем плоскостью P_2 второе полупространство и снова определим, какая группа граней из него должна

изображаться раньше. Продолжая этот процесс до того уровня, когда все пространство будет разбито плоскостями на секции, в каждой из которых будет находиться только одна грань, мы получим упорядоченный набор граней. Этот порядок можно изобразить в виде двоичного дерева. В контексте рассматриваемого алгоритма это дерево представляет собой структуру данных T, элементами которой являются указатель на грань изображаемой сцены, плоскость, отделяющая эту грань, указатели на левое и правое поддерево TL и TR. Такой элемент называется узлом дерева.

В каждом узле дерева левое поддерево будет содержать грани, отделенные плоскостью, а правое - не отделенные. Рисование сцены осуществляется с помощью рекурсивного алгоритма следующего вида:

Метод трассировки лучей

Главная идея этого алгоритма была предложена в 1968 г. А.Аппелем, а первая реализация была выполнена в 1971 г.

Наблюдатель видит любой объект посредством испускаемого неким источником света, который падает на этот объект, отражается или преломляется согласно законам оптики и затем каким-то путем доходит до глаза наблюдателя. Из огромного множества лучей света, выпущенных источником, лишь небольшая часть дойдет до наблюдателя. Следовательно, отслеживать пути лучей в таком порядке неэффективно с точки зрения вычислений. Аппель предложил отслеживать (трассировать) лучи в обратном направлении, т.е. от наблюдателя к объекту. В первой реализации этого метода трассировка прекращалась, как только луч пересекал поверхность видимого непрозрачного объекта; т.е. луч использовался только для обработки скрытых или видимых поверхностей. Впоследствии были реализованы алгоритмы трассировки лучей с использованием более полных моделей освещения с учетом эффектов отражения одного объекта от поверхности другого, преломления, прозрачности и затенения.

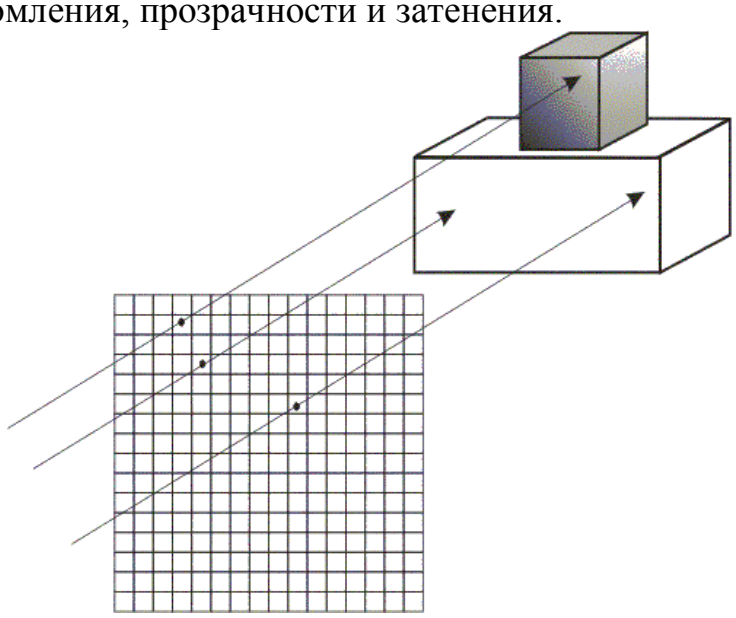


Рис. 6.6.Трассировка параллельными лучами

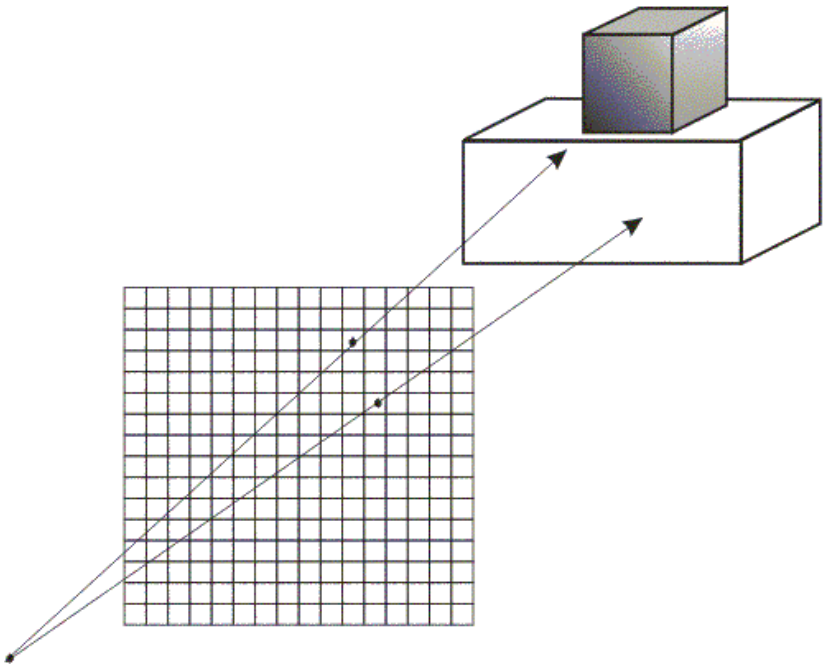


Рис. 6.7. Трассировка с центральной точкой

Необходимо проверить пересечение каждого объекта сцены с каждым лучом. Если луч пересекает объект, то определяются все возможные точки пересечения луча и объекта. Можно получить большое количество пересечений, если рассматривать много объектов. Эти пересечения упорядочиваются по глубине. Пересечение с максимальным значением z представляет видимую поверхность для данного пикселя. Атрибуты этого объекта используются для определения характеристик пикселя.

Наиболее важным и трудоемким элементом этого алгоритма является процедура определения пересечений, поскольку эта задача отнимает наибольшую часть времени всей работы алгоритма. Поэтому эффективность методов поиска особенно важна. Объекты сцены могут состоять из набора плоских многоугольников, многогранников или тел, ограниченных замкнутыми параметрическими поверхностями. Для ускорения поиска важно иметь эффективные критерии, позволяющие исключить из процесса заведомо лишние объекты.

Одним из способов сокращения числа пересекаемых объектов является погружение объектов в выпуклую оболочку - сферическую или в форме параллелепипеда. Поиск пересечения с такой оболочкой очень прост, и если луч не пересекает оболочки, то не нужно больше искать пересечений этого объекта с лучом.

Алгоритм трассировки лучей для простых непрозрачных поверхностей можно представить следующим образом.

Создать список объектов, содержащий:

- полное описание объекта: тип, поверхность, характеристики, тип оболочки и т.п.;

- описание оболочки: центр и радиус для сферы или шесть значений для параллелепипеда $(x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max})$.

Для каждого трассируемого луча:

Выполнить для каждого объекта трехмерный тест на пересечение с оболочкой. Если луч пересекает эту оболочку, то занести объект в список активных объектов.

Если список активных объектов пуст, то изобразить данный пиксель с фоновым значением цвета и продолжать работу. В противном случае для каждого объекта из списка активных объектов:

- Найти пересечения со всеми активными объектами.
- Если список пересечений пуст, то изобразить данный пиксель с фоновым значением цвета.
- В противном случае в списке пресечений найти ближайшее к наблюдателю (с максимальным значением z) и определить атрибуты точки.
- Изобразить данный пиксель, используя найденные атрибуты пересеченного объекта и соответствующую модель освещенности.

Тема 7.Построение реалистических изображений. Модели освещения

Эта группа состоит из наиболее сложных алгоритмов, реализующих закраску участвующих в сцене объектов с учетом их взаимного расположения и физических, в том числе оптических свойств, а также расположения и характеристик источников света.

Визуальное восприятие объектов окружающей действительности представляет собой сложный процесс, имеющий как физические, так и психологические аспекты.

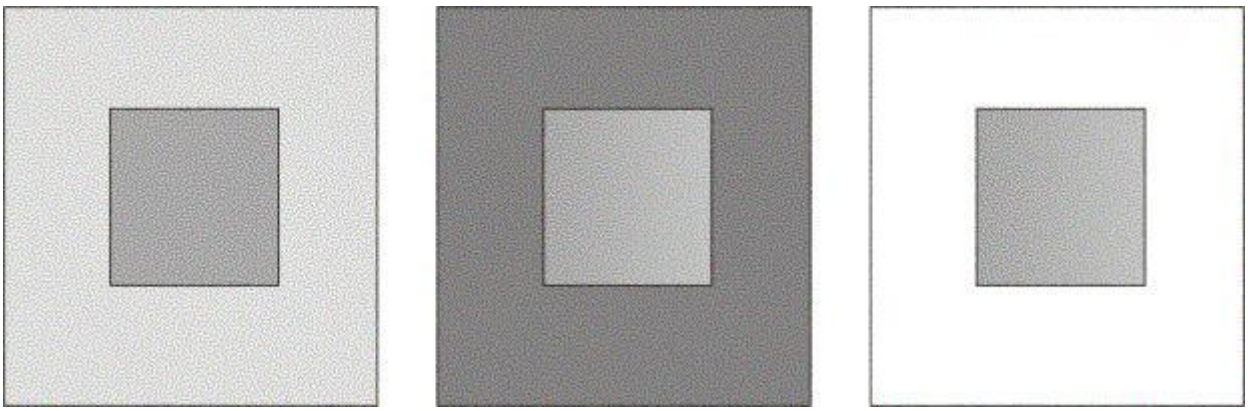


Рис. 7.1. Неоднозначность восприятия

Глаз адаптируется к средней яркости рассматриваемой сцены, поэтому при смене фона изменяется восприятие сцены. Например, однородно окрашенная область на более темном фоне будет казаться более яркой, чем на светлом.

Еще одна особенность восприятия заключается в том, что граница равномерно освещенной области кажется более яркой по сравнению с внутренними частями. Это явление было обнаружено Эрнстом Махом, поэтому оно получило название **эффекта полос Маха**. Такие особенности необходимо учитывать, если мы стремимся к созданию реалистических изображений сцен.

При формировании изображения сцен, содержащих зеркальные и полупрозрачные поверхности, следует использовать законы геометрической оптики, преломляющие свойства материалов, эффекты смешения цветов и т.д.

Закраска граней: плоское закрашивание (Ламберта)

Объекты окружающего пространства становятся видимыми для глаза благодаря световой энергии, которая может излучаться поверхностью предмета, отражаться или проходить сквозь нее. В свою очередь, отражение света от поверхности зависит от физических свойств материала, из которого она изготовлена, а также от характера и расположения источника света. Яркость (или интенсивность) освещения зависит от энергии светового потока, которая обуславливается, во-первых, мощностью источника света, а во-вторых, отражающими и пропускающими свойствами объекта.

Сначала мы рассмотрим модель освещения, учитывающую только отражение. Свойства отраженного света зависят главным образом от направления лучей и характеристик отражающей поверхности.

Отражение может быть двух видов: **диффузное** и **зеркальное**. Первое из них возникает в ситуации, когда свет как бы проникает под поверхность объекта, поглощается, а потом равномерно излучается во всех направлениях. Поверхность в этом случае рассматривается как идеальный рассеиватель. При этом возникает эффект матового света, а видимая освещенность того или иного участка поверхности не зависит от положения наблюдателя. Зеркальное отражение, наоборот, происходит от внешней поверхности, интенсивность его неоднородна, поэтому видимый максимум освещенности зависит от положения глаза наблюдателя.

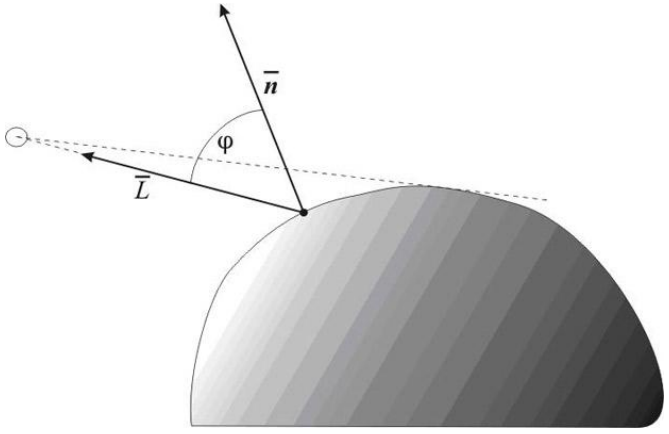


Рис. 7.2. Освещение точечным источником

Свет точечного источника отражается от поверхности рассеивателя по закону Ламберта: интенсивность отражения пропорциональна косинусу угла между внешней нормалью к поверхности и направлением к источнику света Если I_S - интенсивность источника света, φ - угол между вектором внешней нормали к поверхности и направлением к источнику света, то интенсивность отраженного света определяется формулой

$$I = \begin{cases} I_S \cos\varphi \text{ при } 0 \le \varphi \le \pi/2 \\ \\ 0 \text{ в противном случае} \end{cases}$$

При таком расчете интенсивности получится очень контрастная картина, т.к. участки поверхности, на которые лучи от источника не попадают напрямую, останутся абсолютно черными. Для повышения реалистичности необходимо учитывать рассеивание света в окружающем пространстве. Поэтому вводится **фоновая освещенность**, зависящая от интенсивности рассеянного света I_F, и интенсивность отраженного света определяется выражением

$$I = \begin{cases} I_F k_F + k_s I_S \cos\varphi \text{ при } 0 \le \varphi \le \pi/2 \\ \\ I_F k_F \text{ в противном случае} \end{cases}$$

где k_F - коэффициент диффузного отражения рассеянного света, k_S - коэффициент диффузного отражения падающего света, $0 \le k_S \le 1,\ 0 \le k_F \le 1$.

В описанной модели пока никак не учитывалась удаленность источника света от поверхности, поэтому по освещенности двух объектов нельзя судить об их взаимном расположении в пространстве. Если мы хотим получить перспективное изображение, то необходимо включить затухание интенсивности с расстоянием. Обычно интенсивность света обратно пропорциональна квадрату расстояния от источника. В качестве расстояния до источника в случае перспективного преобразования можно взять расстояние до центра проекции, и если он достаточно удален, то изображение будет достаточно адекватным. Но если этот центр расположен близко к объекту, то квадрат расстояния меняется очень быстро, и в этом случае лучше использовать линейное затухание. В этом случае интенсивность отраженного света от непосредственно освещенных участков поверхности будет задаваться формулой

$$I = I_F k_F + \frac{k_S I_S \cos\varphi}{d + C} \qquad (9.3)$$

где d - расстояние до центра проекции, а C - произвольная постоянная. Если центр проекции находится на бесконечности, т. е. при параллельном проецировании, то в качестве d можно взять расстояние до объекта, наиболее близкого к наблюдателю.

В отличие от диффузного, зеркальное отражение является направленным. Идеальное зеркало отражает лучи по принципу "отраженный и падающий лучи лежат в одной плоскости, причем угол падения равен углу отражения" (имеется в виду угол между направлением луча и нормалью к поверхности). Если поверхность не идеально зеркальная, то лучи отражаются в различных направлениях, но с разной интенсивностью, а функция изменения интенсивности имеет четко выраженный максимум.

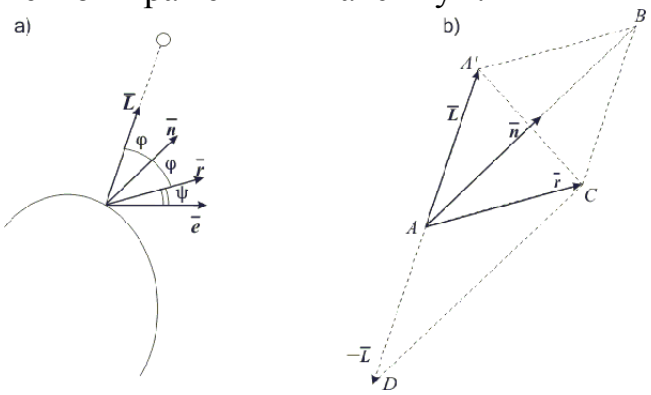


Рис. 7.3. Зеркальное отражение

Вместе с тем, выделяют три относительно простых метода затенения (закрашивания), предполагающие ту или иную интерполяцию освещенности: метод постоянного закрашивания (по Ламберту), метод Гуро и метод Фонга.

Независимо от применяемого в дальнейшем метода поверхности объектов аппроксимируются набором плоских выпуклых граней – ***полигонов***. Чаще всего, для этой цели используются треугольные полигоны. Это объясняется тем, что три вершины однозначно определяют положение плоскости в пространстве и, кроме того, из треугольников всегда можно получить любой другой многоугольник.

Суть метода постоянного закрашивания (по Ламберту) заключается в том, что на каждом полигоне определяется освещенность в произвольной точке, и полученное значение используется для всего полигона. Изображение при этом имеет ярко выраженный полигональный характер – видно, что поверхность состоит из отдельных граней, на границах между которыми освещенность претерпевает разрывы (фактически освещенность является кусочно-постоянной функцией).

Метод Гуро обеспечивает непрерывность освещенности за счет ее билинейной интерполяции: после определения значений освещенности в вершинах полигона применяют линейную интерполяцию вдоль сторон полигона, а потом – линейную интерполяцию между сторонами полигона вдоль каждой из сканирующих строк, пересекающих полигон (при этом освещенность рассчитывается для каждого пикселя соответствующего интервала сканирующей строки).

В методе Фонга также применяется билинейная интерполяция, но по отношению к вектору нормали к поверхности: рассчитывают векторы нормали в вершинах полигона, осуществляют линейную интерполяцию вектора

нормали вдоль сторон полигона, а затем – линейную интерполяцию вдоль каждой сканирующей строки, пересекающей полигон, между сторонами полигона (при этом вектор нормали рассчитывается для каждого пикселя соответствующего интервала сканирующей строки). Полученное поле распределения вектора нормали используется в дальнейшем при расчете освещенности для каждого пикселя полигона (отметим здесь, что ориентация друг относительно друга вектора нормали к поверхности и направления света от точечного источника учитывается во всех моделях освещенности).

Закраска методом Гуро

Один из способов устранения дискретности интенсивностей закрашивания был предложен Гуро. Его метод заключается в том, что используются не нормали к плоским граням, а нормали к аппроксимируемой поверхности, построенные в вершинах многогранника. После этого вычисляются интенсивности в вершинах, а затем во всех внутренних точках многоугольника выполняется билинейная интерполяция интенсивности.
Метод сочетается с алгоритмом построчного сканирования. После того как грань отображена на плоскость изображения, для каждой сканирующей строки определяются ее точки пересечения с ребрами. В этих точках интенсивность вычисляется с помощью линейной интерполяции интенсивностей в вершинах ребра. Затем для всех внутренних точек многоугольника, лежащих на сканирующей строке, также вычисляется интенсивность методом линейной интерполяции двух полученных значений.

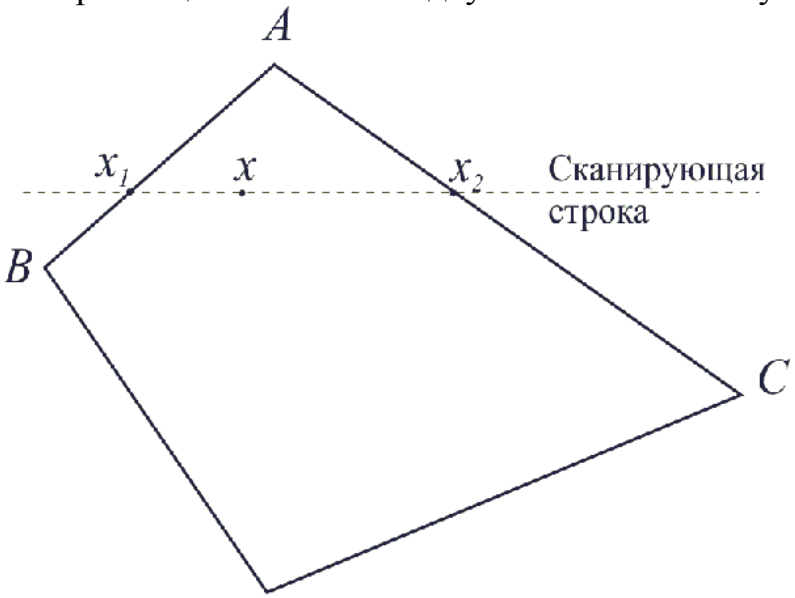


Рис. 7.4. Интерполяция интенсивности

Пусть I_A, I_B, I_C - интенсивности в вершинах A, B, C , x_A, x_B, x_C - горизонтальные координаты этих точек. Тогда в точках пересечения сканирующей строки с ребрами многоугольника интенсивности можно вычислить по формулам интерполяции:

$$I_1 = t_1 I_A + (1 - t_1) I_B, \quad t_1 = \frac{x_1 - x_B}{x_A - x_B},$$
$$I_2 = t_2 I_A + (1 - t_2) I_C, \quad t_2 = \frac{x_2 - x_C}{x_A - x_C}.$$ (9

После этого интенсивность в точке **x** получаем путем интерполяции значений на концах отрезка:

$$I = t I_1 + (1 - t) I_2, \quad t = \frac{x_2 - x}{x_2 - x_1}$$ (9.8)

К недостаткам метода Гуро следует отнести то, что он хорошо работает только с диффузной моделью отражения. Форма бликов на поверхности и их расположение не могут быть адекватно воспроизведены при интерполяции на многоугольниках. Кроме того, есть проблема построения нормалей к поверхности. В алгоритме Гуро нормаль в вершине многогранника вычисляется путем усреднения нормалей к граням, примыкающим к этой вершине. Такое построение сильно зависит от характера разбиения.

Модель Фонга

описывается соотношением

$$I_Z = \omega(\varphi, \lambda) \cdot I_S \cos^n \psi$$

где $\omega(\varphi, \lambda)$ - функция отражения, λ - длина волны. Степень, в которую возводится косинус угла, влияет на размеры светового блика, наблюдаемого зрителем. Графики этой функции являются характерными кривыми поведения функции изменения интенсивности в зависимости от свойств поверхности.

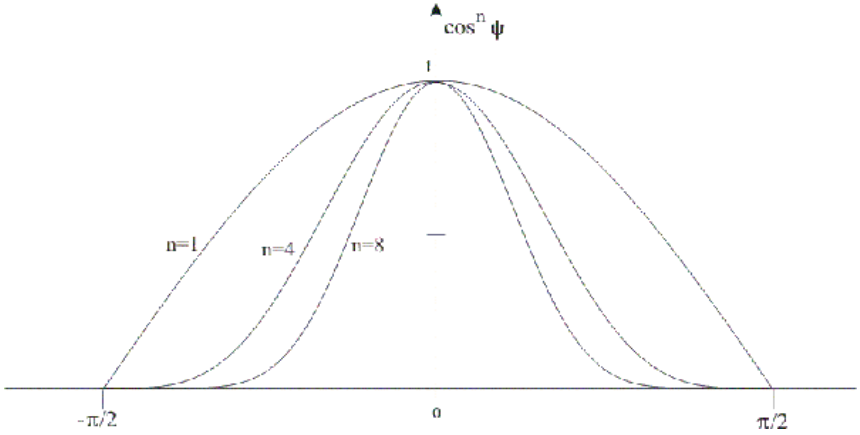


Рис. 7.5. Зеркальное отражение

Теперь модель освещенности, учитывающую зеркальное и диффузное отражения, можно описать формулой

$$I = I_F k_F + \frac{I_S (k_s \cos\varphi + \omega(\varphi, \lambda) \cdot \cos^n \psi)}{d + C}.$$

Используя единичные векторы $\vec{L}$ (направление к источнику) и $\vec{n}$ (внешняя нормаль), косинус угла φ можно вычислить через скалярное произведение: $\cos\varphi = (\vec{L} \cdot \vec{n})$. Для расчета интенсивности зеркального отражения сначала надо определить отраженный вектор $\vec{r}$. Из рис. 11.2

видно, что $\vec{r} = |AB| \cdot \vec{n} - \vec{L}$. С другой стороны AB является диагональю ромба $AA'BC$, поэтому $|AB| = 2 \cdot (\vec{L} \cdot \vec{n})$. Учитывая все эти соотношения, получаем формулу

$$I = I_F k_F + \frac{I_S\{k_s(\vec{L} \cdot \vec{n}) + \omega(\varphi, \lambda) \cdot [2 \cdot (\vec{L} \cdot \vec{n}) \cdot (\vec{e} \cdot \vec{n}) - (\vec{e} \cdot \vec{L})]^n\}}{d + C}.$$

В алгоритмах закрашивания с использованием цветовых моделей интенсивность рассчитывается для каждого из базовых цветов, поскольку изменение интенсивности при зеркальном отражении зависит от длины волны. Фонг предложил вместо интерполяции интенсивностей произвести интерполяцию вектора нормали к поверхности на сканирующей строке. Этот метод требует больших вычислительных затрат, поскольку формулы интерполяции применяются к трем компонентам вектора нормали, но зато дает лучшую аппроксимацию кривизны поверхности. Поэтому зеркальные свойства поверхности воспроизводятся гораздо лучше.

Нормали к поверхности в вершинах многогранника вычисляются так же, как и в методе Гуро. А затем выполняется билинейная интерполяция в сочетании с построчным сканированием. После построения вектора нормали в очередной точке вычисляется интенсивность.

Этот метод позволяет устранить ряд недостатков метода Гуро, но не все. В частности, эффект полос Маха в отдельных случаях в методе Фонга бывает даже сильнее, хотя в подавляющем большинстве случаев аппроксимация Фонга дает лучшие результаты.

Тема 8.Визуализация пространственных реалистических сцен

Свето-теневой анализ

Тени возникают как результат экранирования источника света предметами, составляющими сцену. При этом существуют полные тени и полутени, что связано с физической природой света: полная тень образуется в центральной части затенения, а полутень - вблизи ее границ.

Если считать, что наблюдатель находится в одной точке с источником света, то тени в наблюдаемой сцене не возникают: затеняемая область является невидимой. Во всех прочих случаях тени видны. Источник света может находиться на бесконечности или на конечном расстоянии, причем во втором случае он может оказаться в поле зрения наблюдателя.

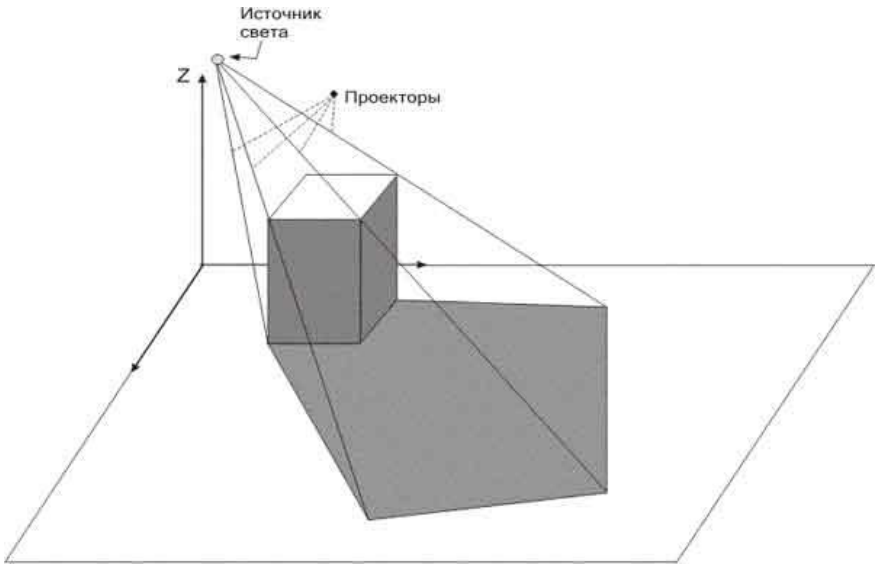


Рис. 8.1. Образование теней при конечном расстоянии до источника света

Для бесконечно удаленного источника света тени на картинной плоскости получаются в результате параллельной проекции объектов, а для близкого источника - центральной проекции. Объекты и их части становятся невидимыми, если они попадают в область тени, поэтому при построении изображения задача об удалении невидимых областей решается дважды: относительно наблюдателя в процессе проецирования и относительно источника света. При определении расположения теней строятся проекции невидимых с позиции источника света граней (неосвещенных) на картинную плоскость, в результате чего получаются теневые многоугольники. Эти многоугольники строятся для всех объектов сцены и заносятся в список. Отметим, что теневые многоугольники не зависят от положения наблюдателя, поэтому при осмотре сцен с различных точек зрения они строятся только один раз. Впервые идея совмещенного анализа видимости и затененности была предложена в 1968 г. Аппелем. В качестве примера рассмотрим один алгоритм на основе построчного сканирования, состоящий из двух основных этапов.

I. Анализ сцены по отношению к источнику света. Для всех многоугольников, полученных в результате проецирования сцены, определяются неосвещенные (затененные) участки и теневые многоугольники (проекционные тени), причем многоугольники образуют пронумерованный список. Для этих многоугольников формируется матрица $A = (a_{ij})$, позволяющая определить, отбрасывает ли многоугольник тень и какие из многоугольников он может закрывать. Если для некоторых значений $i, j \quad a_{ij} = 1$, то это означает, что многоугольник с номером i может отбрасывать тень на многоугольник с номером j.

Таким образом, на этом этапе основным является вопрос об эффективном алгоритме построения такой матрицы. Если проекция включает N многоугольников, то необходимо рассмотреть "взаимоотношения" $N \cdot (N-1)$ пар многоугольников. Сократить перебор

можно за счет погружения объектов в прямоугольные или сферические оболочки или путем использования сортировки по глубине.

II. Анализ сцены по отношению к наблюдателю. Выполняются два процесса сканирования. Первый - для определения отрезков, видимых с позиции наблюдателя. Второй - для определения пересечений отрезков с теневыми многоугольниками из списка. Рекурсивный алгоритм второго сканирования состоит из четырех основных шагов.

Для каждого видимого отрезка:

1. Если нет ни одного теневого многоугольника, то отрезок изображается с основной интенсивностью.
2. Если многоугольник, содержащий видимый отрезок, не пересекается с теневыми многоугольниками, то отрезок изображается с основной интенсивностью.
3. Если отрезок полностью закрывается некоторыми теневыми многоугольниками, то интенсивность его изображения определяется с учетом затенения всеми этими многоугольниками.
4. Если несколько теневых многоугольников частично закрывают отрезок, то он разбивается на ряд отрезков точками пересечения с теневыми многоугольниками.

Этот алгоритм предполагает, что затенение не абсолютное, т.е. затененные участки все-таки являются видимыми, только их освещенность падает в зависимости от количества и освещенности затеняющих многоугольников. При полном затенении в третьем пункте алгоритма отрезок становится полностью невидимым, а в четвертом дальнейшему анализу подвергаются только незатененные отрезки.

Способы расчета интенсивности при неполном затенении могут быть различны. В этом случае все затененные многоугольники имеют свою интенсивность в зависимости от выбранной модели освещенности. При этом можно учитывать расстояние затененного участка от поверхности, отбрасывающей тень.

Еще один алгоритм, часто применяемый при построении теней, носит название метода теневого буфера. Он строится на основе метода Z-буфера, описанного Теневой буфер - это тот же Z-буфер, только с точки зрения источника света. Таким образом, используются два буфера: один - для расстояния от картинной плоскости до точек изображаемой сцены, а другой - для расстояний от этих же точек до источника света. Алгоритм позволяет изображать сцены с полным затенением и сводится к двум основным этапам:

1. Сцена рассматривается из точки расположения источника света в соответствующей системе координат. Итогом построения является полностью заполненный теневой буфер.
2. Сцена рассматривается с точки зрения наблюдателя, применяется обычный метод Z-буфера с небольшим дополнением. Если точка (x, y, z) является видимой в этой системе координат, то вычисляются ее координаты в системе, связанной с источником света - (x', y', z') , затем проверяется,

является ли точка видимой с этой позиции. Для этого значение z' сравнивается со значением, содержащимся в теневом буфере для этой точки, и в случае видимости значение интенсивности заносится в буфер кадра в точке (x, y).

Оба приведенных алгоритма работают в пространстве изображения, т. е. имеют дело с проекциями на плоскость и некоторой дополнительной информацией о точках сцены, соответствующих этим проекциям. Существуют алгоритмы, работающие в трехмерном объектном пространстве. В частности, для построения теней используются модификации алгоритма Вейлера-Азертона. Модификация заключается в том, что, как и в случае теневого буфера, задача удаления невидимых граней решается сначала с позиции источника света, а затем полученная информация об объектах используется при построении изображения с позиции наблюдателя. В общих чертах шаги алгоритма можно описать так:

1. Определяются грани, видимые из точки расположения источника света. С целью повышения эффективности запоминается информация только о видимых гранях. Поскольку анализ выполняется в системе координат, связанной с источником света, то полученные видимые многоугольники затем заново приводятся к исходной системе координат. Многоугольники связываются с гранями, которым они принадлежат (в результате затенения одна грань может содержать несколько многоугольников).
2. Сцена обрабатывается из положения наблюдателя. При изображении видимой грани учитываются только те многоугольники, которые входят в список, полученный на первом этапе, т.е. грань рассматривается как совокупность таких многоугольников.

При наличии нескольких источников света количество освещенных участков естественным образом увеличивается.

Метод излучательности

Освещенность поверхности определяется собственным излучением тела и отраженными лучами, падающими от других тел (источников). Модель излучательности включает оба эти фактора и основана на уравнениях энергетического баланса. При этом выполняемые расчеты учитывают только взаимное расположение элементов сцены и не зависят от положения наблюдателя.

Представим сцену из N элементов (участков поверхностей). Освещенность будем моделировать как количество энергии, излучаемое поверхностью. Для каждого элемента это количество энергии складывается из собственной энергии (E_k) и отраженной доли энергии, полученной от других объектов. Предполагается, что для каждой пары элементов с номерами i, j можно определить, какая доля энергии одного попадает на другой (w_{ij}). Пусть α_i - коэффициент отражения энергии $i-$м элементом. Тогда полная энергия, излучаемая этим элементом, будет определяться уравнением $U_i = E_i + \alpha_i \sum_{j=1}^{N} w_{ij} U_j$.

Таким образом, мы получаем систему уравнений для нахождения значений U_i, которая в матричном виде выглядит следующим образом:

$(I - W) \cdot U = E$

где I - единичная матрица, U и T - векторы излучаемой и собственной энергий, а матрица W состоит из элементов $(\alpha_i w_{ij})$. Поскольку часть излучения элемента может не попадать ни на один из оставшихся, то

$$\sum_{i=1}^{N} w_{ij} \leq 1,$$

а это условие в сочетании с тем, что $\alpha_i < 1$ (отражение не является полным), приводит к тому, что матрица системы имеет так называемое **диагональное преобладание**, т.е. диагональный элемент по абсолютной величине больше, чем сумма остальных элементов строки. В таком случае система уравнений имеет решение, которое можно найти с помощью численных методов.

Итак, шаги алгоритма изображения сцены сводятся к следующим:

1. Сцена разбивается на отдельные участки, для каждого из которых определяются значения $E_i, \alpha_i, w_{ij}, \quad j = 1, 2, \ldots, N$.
2. Находятся значения U_i для каждой из трех основных компонент цвета.
3. Для выбранной точки наблюдения строится проекция с удалением невидимых граней и осуществляется закрашивание, использующее значения U_i для задания интенсивности. При этом могут использоваться какие-либо алгоритмы, позволяющие сгладить изображение.

Сложным моментом в модели излучательности является расчет коэффициентов w_{ij}.

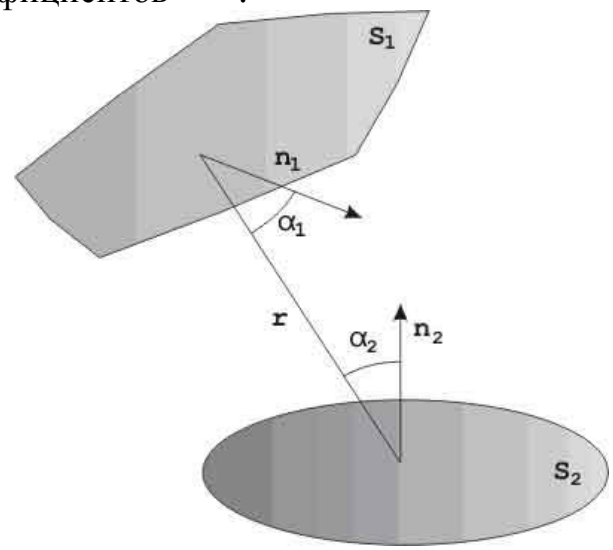


Рис. 8.2. Два элемента сцены

Рассмотрим один пример. Пусть имеется два элемента сцены S_1 и S_1 (рис. 11.2.). Поскольку используется диффузная модель освещения, то доля энергии малого участка dS_1 с нормалью $\vec{n}_1$, излучаемая под углом α_1 к этой нормали, пропорциональна косинусу угла.

Следовательно, в направлении элементарного участка dS_2 уходит доля энергии, пропорциональная косинусу угла между $\vec{n}_1$ и отрезком, который соединяет эти участки. Соответственно, получаемая вторым участком доля этой энергии будет пропорциональна косинусу угла между нормалью $\vec{n}_2$ и этим же отрезком. Итак, доля энергии, получаемая элементом dS_2 от элемента, dS_1 - $dw_{21} = \cos(\alpha_1) \cdot \cos(\alpha_2)/\pi r^2$, где r - расстояние между элементами. Кроме того, необходимо учесть, что излучаемая элементарным участком энергия равномерно распределена по всем направлениям. И, наконец, в каждой сцене одни объекты могут частично экранировать другие, поэтому надо ввести коэффициент, определяющий степень видимости объекта с позиции другого. Далее полученное выражение интегрируется по S_1 и S_2, что также может быть сложной задачей.

Отсюда видно, насколько трудоемкой может оказаться процедура вычисления коэффициентов w_{ij} . Поэтому, как правило, используются приближенные методы их вычисления. В частности, можно рассматривать поверхности объектов как многогранники, тогда элементами сцены будут плоские многоугольники, для которых формулы несколько упрощаются.

Глобальная модель освещения с трассировкой лучей

Мы уже касались ранее понятия трассировки лучей при описании алгоритмов удаления невидимых граней. Теперь рассмотрим аналогичную процедуру в применении к моделям освещения. В предыдущей главе были описаны модели освещенности от некоторого источника света без учета того, что сами объекты сцены освещают друг друга посредством отраженных лучей. Метод излучательности, разработанный для диффузной модели освещенности, уже учитывает этот фактор.

Глобальная модель освещенности способна воспроизводить эффекты зеркального отражения и преломления лучей (прозрачность и полупрозрачность), а также затенение. Она является составной частью алгоритма удаления невидимых поверхностей методом трассировки.

Если рассмотреть сцену, содержащую в числе прочих зеркальные и полупрозрачные поверхности (рис. 11.3.), то изображение будет включать, во-первых, проекции самих объектов, освещенных одним или несколькими источниками света. В некоторых своих частях эти объекты будут искажены за счет преломления лучей в прозрачных и полупрозрачных телах. Во- вторых, часть объектов будет отражаться зеркальными поверхностями, и эти отражения появятся на проекциях зеркальных объектов. В изображенной на рис. 11.3. сцене точки на поверхности призмы C, D видны на картинной плоскости дважды: один раз - сквозь полупрозрачный параллелепипед в виде точек $\tilde{C}$, $\tilde{D}$, а второй раз - как дважды отраженные невидимой поверхностью параллелепипеда и зеркалом C'', D'' . Параллелепипед в данном случае частично обладает зеркальными свойствами.

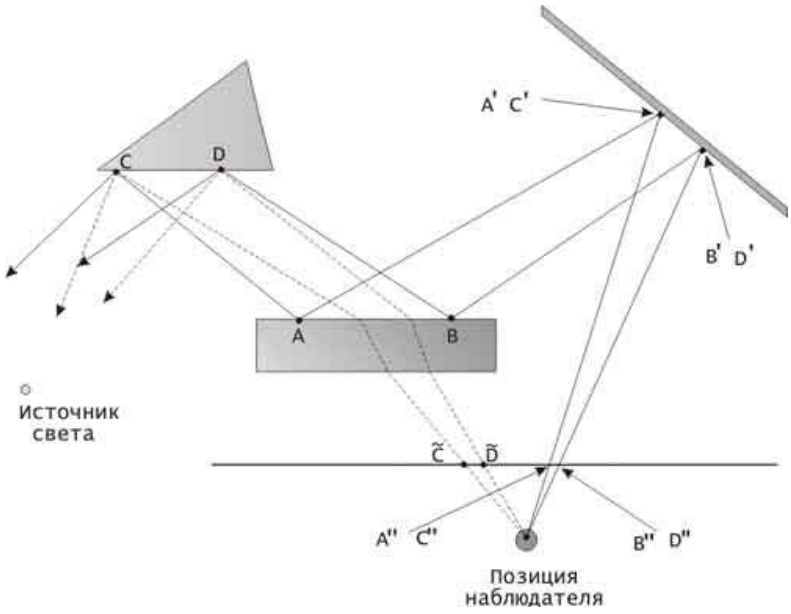


Рис. 8.3. Сцена, содержащая зеркальные и полупрозрачные поверхности

Глобальная модель освещения для каждого пикселя изображения определяет его интенсивность. Будем для простоты считать, что все источники света - точечные. Сначала определяется непосредственная освещенность источниками без учета отражений от других поверхностей (вторичная освещенность): отслеживаются лучи, направленные ко всем источникам. Тогда наблюдаемая интенсивность (или отраженная точкой энергия) выражается следующим соотношением:

$$I = k_0 I_0 + k_d \sum_j I_j(\vec{n} \cdot \vec{l}_j) + k_r \sum_j I_j(\vec{s} \cdot \vec{r}_j)^\beta + k_r I_r + k_t I_t,$$

где

k_0 - коэффициент фонового (рассеянного) освещения;

k_d - коэффициент диффузного отражения;

k_r - коэффициент зеркального отражения;

k_t - коэффициент пропускания;

$\vec{n}$ - единичный вектор нормали к поверхности в точке;

$\vec{l}_j$ - единичный вектор, направленный к j -му источнику света;

$\vec{s}$ - единичный локальный вектор, направленный в точку наблюдения;

$\vec{r}_j$ - отраженный вектор $\vec{l}_j$;

I_0 - интенсивность фонового освещения;

I_j - интенсивность j -го источника света;

I_r - интенсивность, приходящая по зеркально отраженному лучу;

I_t - интенсивность, приходящая по преломленному лучу.

В алгоритме удаления невидимых линий трассировка луча продолжалась до первого пересечения с поверхностью. В глобальной модели освещения этим дело не ограничивается: осуществляется дальнейшая

трассировка отраженного и преломленного лучей. Таким образом, происходит разветвление алгоритма в виде двоичного дерева. Процесс продолжается до тех пор, пока очередные лучи не останутся без пересечений. Отражение и преломление рассчитываются по законам геометрической оптики, которые уже рассматривались в предыдущей главе.

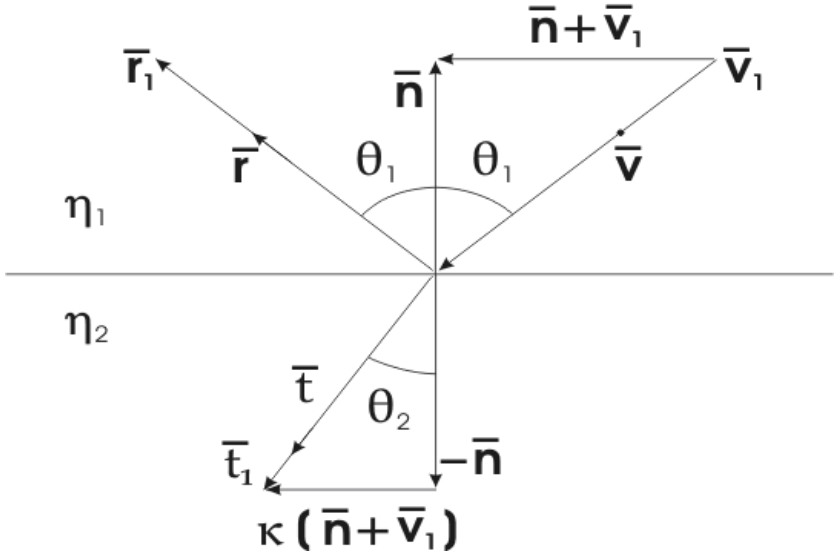


Рис. 8.4. Зеркальное отражение и преломление

Пусть $\vec{v}, \vec{r}, \vec{t}$ - направления падающего, отраженного и преломленного лучей (рис.11.4.), $\vec{v}_1 = \vec{v}/\cos(\theta_1)$, $\vec{n}$ - единичная внешняя нормаль, η_1, η_2 - коэффициенты преломления сред, разделенных поверхностью. Тогда можно показать, что

$$\vec{r}_1 = \vec{v}_1 + 2 \cdot \vec{n}, \quad \vec{t}_1 = \kappa(\vec{n} + v_1) - \vec{n},$$
$$\kappa = (k_\eta^2 \cdot |\vec{v}_1|^2 - |\vec{v}|_1 + \vec{n}^2)^{-1/2}, \quad k_\eta = \frac{\eta_2}{\eta_1}.$$

Соответствующие единичные векторы получить нетрудно.

Двоичное дерево лучей можно строить по принципу "левое поддерево соответствует отраженному лучу, а правое - преломленному". После того как оно построено, можно вычислить интенсивность в точке. Для этого осуществляется обратный проход от вершин к корню, и при прохождении узлов интенсивность убывает.

Теоретически дерево может оказаться бесконечным, поэтому при его построении желательно задать максимальную глубину, чтобы избежать переполнения памяти компьютера.

Поскольку значительная часть лучей, исходящая от источников света и других поверхностей, не попадает в поле зрения наблюдателя, то отслеживать их все не имеет смысла. Поэтому для формирования изображения используется обратная трассировка, т. е. лучи отслеживаются в обратном порядке: от положения наблюдателя через все точки картинной плоскости к объектам и далее - по отраженным и преломленным лучам.

Текстуры

Текстура поверхности - это детализация ее строения, учитывающая микрорельеф и особенности окраски. Во-первых, гладкая поверхность может быть покрыта каким-либо узором, и тогда при ее изображении решается задача отображения этого узора на проекции фрагментов поверхности (многоугольники). Во-вторых, поверхность может быть шероховатой, поэтому нужны специальные приемы имитации такого микрорельефа при закрашивании.

Сначала рассмотрим методы отображения узоров. Чаще всего узор задается в виде образца, заданного на прямоугольнике в декартовой системе координат η, ξ в пространстве текстуры. Фрагмент поверхности может быть задан в **параметрическом виде** в трехмерной декартовой системе координат:

$$x = f(u,v), \quad y = g(u,v), \quad z = h(u,v), \quad a \le u \le b, \quad c \le v \le d.$$

Теперь достаточно построить отображение области в пространстве текстуры в область параметров поверхности

$$u = \varphi(\eta,\xi), \quad v = \psi(\eta,\xi),$$

или

$$\eta = \chi(u,v), \quad \xi = \theta(u,v),$$

и тем самым каждой точке поверхности будет соответствовать точка образца текстуры. Пусть, например, поверхность представляет собой один октант сферы единичного радиуса, заданный формулами

$$x = \sin\alpha \cdot \sin\beta, \quad y = \cos\beta, \quad z = \cos\alpha \cdot \sin\beta,$$
$$0 \le \alpha \le \pi/2, \quad \pi/4 \le \beta \le \pi/2$$

а образец текстуры задан на квадрате $0 \le \eta \le 1, \quad 0 \le \xi \le 1$. Тогда можно воспользоваться линейным отображением вида

$$\alpha = a\eta + b, \quad \beta = c\eta + d.$$

Если положить $a = \pi/2, \quad b = 0, \quad c = -\pi/4, \quad d = \pi/2$, то углы образца отобразятся в углы криволинейного четырехугольника, как это показано на рис. 11.6.

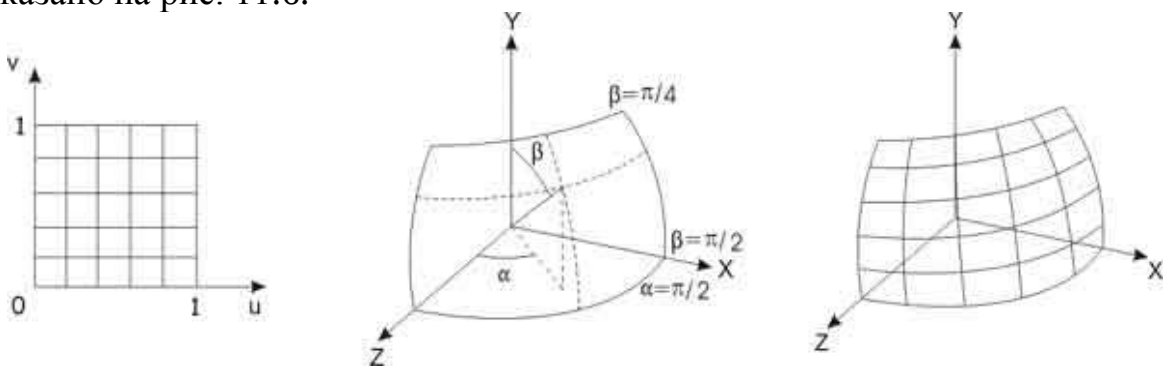


Рис. 8.5. Текстура на сферической поверхности

Обратное отображение имеет вид

$$\eta = \frac{\alpha}{\pi/2}, \quad \xi = \frac{\pi/2 - \beta}{\pi/4},$$

следовательно, вертикальные и горизонтальные линии образца отобразятся на окружности большого круга сферы.

Пусть теперь нужно нанести текстуру при перспективном проецировании произвольно ориентированной прямоугольной грани. Грань задана в пространстве набором своих вершин A, B, C, D . Построим векторы $\vec{e}_1 = B - A$ и $\vec{e}_2 = D - A$, направленные вдоль сторон прямоугольника. Любую точку прямоугольника можно единственным образом представить в виде

$$P = A + u\vec{e}_1 + v\vec{e}_2.$$

Будем считать, что используется простейший случай перспективного преобразования, задаваемый формулами

$$x' = \frac{x}{z}, \quad y' = yz.$$

Найдем образ точки P при таком преобразовании:

$$x' = \frac{A_x + u\vec{e}_{1x} + v\vec{e}_{2x}}{A_z + u\vec{e}_{1z} + v\vec{e}_{2z}}, \quad y' = \frac{A_y + u\vec{e}_{1y} + v\vec{e}_{2y}}{A_z + u\vec{e}_{1z} + v\vec{e}_{2x}},$$

или

$$\begin{cases} u(x'\vec{e}_{1z} - \vec{e}_{1x}) + v(x'\vec{e}_{2z} - \vec{e}_{2x}) = A_x - A_z x' \\ \\ u(y'\vec{e}_{1z} - \vec{e}_{1y}) + v(y'\vec{e}_{2z} - \vec{e}_{2y}) = A_y - A_z y' \end{cases}.$$

Если теперь рассматривать эти соотношения как систему уравнений для нахождения параметров u, v , то, решив ее, получим требуемое обратное преобразование. Для решения можно воспользоваться, например, правилом Крамера:

$$u = \Delta_u/\Delta, \quad v = \Delta_v/\Delta$$

где

$$\Delta = (x'\vec{e}_{1z} - \vec{e}_{1x}) \cdot (y'\vec{e}_{2z} - \vec{e}_{2y}) - (x'\vec{e}_{2z} - \vec{e}_{2x}) \cdot (y'\vec{e}_{1z} - \vec{e}_{1y});$$

$$\Delta_1 = (A_x - A_z x') \cdot (y'\vec{e}_{2z} - \vec{e}_{2y}) - (x'\vec{e}_{2z} - \vec{e}_{2x}) \cdot (A_y - A_z y');$$

$$\Delta_2 = (x'\vec{e}_{1z} - \vec{e}_{1x}) \cdot (A_y - A_z y') - (A_x - A_z x') \cdot (y'\vec{e}_{1z} - \vec{e}_{1y}).$$

Найденные параметры будут определять точку текстуры, соответствующую точке проекции.

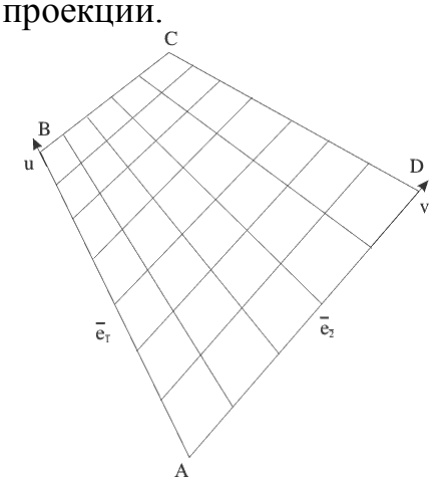


Рис. 8.6. Текстура при перспективной проекции

Методические указания к выполнению лабораторных работ

Общие методические указания к выполнению лабораторной работы №1

Исследование двумерных преобразований графических объектов

Рис.1. Панель управления

После загрузки приложения, реализующего работу лабораторно-исследовательского модуля, на экране дисплея отображается интерфейс программы, который обязательно включает ***панель управления*** (на рис.1 приведен вид такой панели при двумерных преобразованиях) и ***область построений***.

На панели управления имеется несколько блоков, предназначенных для введения данных и наблюдения результатов расчетов. Окна ввода блока ***«Однородные координаты вершин исходного …»*** заполняются однородными координатами вершин исходного графического объекта: при двумерных преобразованиях – треугольника, при пространственных преобразованиях – четырехгранника. При этом допускается введение только целочисленных значений координат ***x***, ***y*** и – при пространственных преобразованиях – ***z***. Отсутствие значения в окне ввода воспринимается программой как наличие в

нем нуля. Последняя однородная координата ***h*** любой вершины вводу и изменению не подлежит (т.к. заведомо равна единице). После нажатия кнопки ***«ОК»***, расположенной в рассматриваемом блоке, происходит следующее. Если введенные данные позволяют полностью визуализировать объект в области построений, в информационных окнах блока выводятся сообщения о принадлежности вершин (точек) этой области. Если какая-либо точка выходит за границы области построений, появляется соответствующее сообщение. В этом случае необходимо изменить координаты данной точки таким образом, чтобы она оказалась внутри области построений. Далее осуществляются проверки: при двумерных преобразованиях – лежат ли вершины треугольника на одной прямой; при пространственных преобразованиях – лежат ли первые три вершины четырехгранника на одной прямой, а также лежит ли четвертая вершина четырехгранника в той же плоскости, что и первые три. В подобных ситуациях никаких ограничений на проведение преобразований нет, но исходные объекты и, как правило, результаты их преобразований не отличаются наглядностью. Поэтому, если что-либо из перечисленного подтверждается, выводится соответствующее сообщение, и следует, учитывая геометрический смысл сообщения, изменить координаты хотя бы одной из вершин. Так или иначе, по окончании работы с блоком ***«Однородные координаты вершин исходного …»***, заканчивающейся нажатием кнопки ***«ОК»,*** исходный графический объект визуализируется в области построений. Одновременно с этим происходит блокировка окон ввода данных об этом объекте. В дальнейшем изменить эти данные возможно только после нажатия кнопки ***«Удалить все»*** (см. ниже).

Во втором блоке панели управления ***«Однородные координаты вершин преобразуемого …»*** при первом преобразовании графического объекта дублируются данные из первого блока. Вместе с тем, в лабораторно-исследовательских модулях предусмотрена возможность осуществления последовательных преобразований объектов. При таких преобразованиях в рассматриваемый блок можно автоматически вводить результат последнего преобразования из блока ***«Результат преобразования после нормализации»***. В любом случае, если второй блок заполнен, данные в нем соответствуют однородным координатам вершин того объекта, который будет подвергаться преобразованию; при этом в области построений можно наблюдать этот объект или его фрагмент, если, конечно, объект соответственно полностью или частично расположен внутри данной области.

Блок ***«Матрица преобразования»*** панели управления предназначен для ввода коэффициентов матрицы общего преобразования, которым можно присваивать вещественные значения (с точностью до пятого знака после запятой). При вводе вещественных значений коэффициентов целую и дробную части следует разделять точкой. Пустое окно равнозначно наличию в нем нуля. После нажатия в блоке кнопки ***«ОК»*** происходит умножение матрицы, содержащей однородные координаты вершин преобразуемого объекта, на матрицу общего преобразования.

Результат такого умножения выводится в окнах блока

«Промежуточный результат преобразования».

В то же время в блоке ***«Результат преобразования после нормализации»*** появляется окончательный результат преобразования, полученный путем нормализации результата умножения матриц и округления координат вершин преобразованного объекта до ближайших целых чисел. Если преобразованный графический объект полностью или частично расположен внутри области построений, он будет соответственно полностью или частично визуализирован в данной области. При этом в случае, когда какая-либо вершина преобразованного объекта имеет конечные координаты, это находит отражение в соответствующем информационном окне, даже если эта вершина находится вне области построений. Если все вершины имеют конечные координаты, полученный объект можно подвергнуть очередному преобразованию. Если же одна или более вершин преобразованы в точки бесконечности – о чем также выводится сообщение – дальнейшие преобразования осуществить не удастся.

Модуль предоставляет возможность выбора трех возможных вариантов продолжения исследований. Кнопка ***«Использовать результат для следующего преобразования»*** позволяет полученный в результате преобразования объект, имеющий конечные координаты всех вершин, подвергнуть очередному преобразованию. При ее нажатии однородные координаты вершин объекта автоматически заносятся в блок ***«Однородные координаты вершин преобразуемого …»*** (см. выше); одновременно с этим из области построений удаляются данные о том объекте, который подвергался предыдущему преобразованию, но сохраняется изображение исходного объекта. Кнопка ***«Удалить результаты преобразований»*** удаляет из области построений все, кроме исходного графического объекта, и вновь вводит однородные координаты его вершин в блок ***«Однородные координаты вершин преобразуемого …»***. Кнопка ***«Удалить все»*** полностью очищает область построений и открывает доступ к окнам ввода данных об исходном объекте в блоке ***«Однородные координаты вершин исходного …»***. Ее использование целесообразно, если начинаются исследования преобразований нового графического объекта.

Методические указания по выполнению работы

При двумерных преобразованиях графических объектов каждая точка $\boldsymbol{P}$ $\boldsymbol{(x, y)}$ на плоскости однозначно отображается содержащей однородные координаты этой точки матрицей (координатным вектором) размером 1×3 вида $\begin{bmatrix} x & y & 1 \end{bmatrix}$. Отрезку прямой между точками $\boldsymbol{(x_1, y_1)}$ и $\boldsymbol{(x_2, y_2)}$ ставится в соответствие 2×3 матрица вида $\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{bmatrix}$. Многоугольник может быть представлен $\boldsymbol{M}$×3 матрицей (где $\boldsymbol{M}$ – число вершин многоугольника),

содержащей однородные координаты его вершин: $\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \dots & \dots & \dots \\ x_M & y_M & 1 \end{bmatrix}$.

Преобразования осуществляются путем умножения таких матриц на матрицу общего преобразования размером 3×3 вида $[T]=\begin{bmatrix} a & b & p \\ c & d & q \\ m & n & s \end{bmatrix}$. Результат преобразования зависит от конкретного вида матрицы преобразования. Если координатный вектор преобразованной точки $[x' \quad y' \quad h]$ содержит $h \neq 1$ и $h \neq 0$, результат нормализуют путем деления всех трех составляющих однородных координат на h, т.е. приводят к виду $[x^* \quad y^* \quad 1]$, где $x^* = x'/h$, $y^* = y'/h$. Равенство нулю координаты h в результате матричного умножения (координатный вектор преобразованной точки при этом имеет вид $[x' \quad y' \quad 0]$) свидетельствует о том, что исходная точка преобразована в точку бесконечности, лежащую на луче, который идет из начала координат через точку (x', y').

Требования к оформлению результатов выполнения лабораторных работ

Результаты проведения лабораторных исследований оформляются в виде индивидуальных (для каждого студента) отчетов. Пример оформления ***титульного листа*** отчета приведен в приложении 1. Основная часть отчета должна содержать:

- формулировку ***цели проведения исследований***;
- ***исходные данные*** для проведения исследований – приводятся координаты вершин исходного графического объекта;
- ***результаты проведения исследований*** – для каждого пункта (подпункта) программы работ, связанного с каким-либо преобразованием графического объекта, следует пояснить суть преобразования, привести матрицу общего преобразования с конкретными численными значениями всех ее коэффициентов, проиллюстрировать преобразование соответствующим фрагментом области построений; если на то есть указание в программе работ, необходимо сформулировать промежуточные выводы по результатам выполнения пункта (или нескольких пунктов) исследований;
- ***общие выводы по результатам исследований*** – составляются с использованием промежуточных выводов по соответствующим пунктам исследований; в общих выводах обязательно должны содержаться сведения о назначении отдельных коэффициентов матрицы общего преобразования и четырех ее подматриц в целом, а также о правилах реализации комбинированных преобразований графических объектов.

Матрицы простых двумерных преобразований

Локальное масштабирование: в a раз по координатной оси x, в d раз по координатной оси y	$[T]=\begin{bmatrix} a & 0 & 0 \\ 0 & d & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Симметричное отражение относительно координатной оси x	$[T]=\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Симметричное отражение относительно координатной оси y	$[T]=\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Симметричное отражение относительно точки начала координат (поворот вокруг точки начала координат на 180°)	$[T]=\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Сдвиги: вдоль координатной оси x на cy, вдоль координатной оси y на bx	$[T]=\begin{bmatrix} 1 & b & 0 \\ c & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Поворот на произвольный угол θ относительно точки начала координат	$[T]=\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Отражение относительно прямой $y = x$	$[T]=\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Отражение относительно прямой $y = -x$	$[T]=\begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Перемещения: вдоль координатной оси x на m, вдоль координатной оси y на n	$[T]=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & 1 \end{bmatrix}$

Проецирование в однородных координатах (если $h = px + qy + 1 \neq 1$ и $h \neq 0$, результат необходимо нормализовать путем деления всех однородных координат на h)	$[T] = \begin{bmatrix} 1 & 0 & p \\ 0 & 1 & q \\ 0 & 0 & 1 \end{bmatrix}$
Общее масштабирование в $1/s$ раз (если $h = s \neq 1$, и $h \neq 0$ результат необходимо нормализовать путем деления всех однородных координат на h)	$[T] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s \end{bmatrix}$

Варианты заданий к лабораторной работе №1

Дисциплина: «Компьютерная геометрия и графика»	
Лабораторная работа №1 ***Вариант задания №1***	
Координаты вершин исходного треугольника	
$x_1 = 20$; $y_1 = 10$; $x_2 = 40$; $y_2 = 70$; $x_3 = 100$; $y_3 = 50$	
Пункт/подпункт программы работ	Параметры преобразования
2a	$a = 2$; $d = 1{,}5$ *
2d	$c = 2$ (или $b = 1{,}5$) *
2f	$\theta = 50°$
3	$m = 60$; $n = -80$ *
4	$m = 10$; $n = 60$; $\theta = -75°$
6	$k = 0{,}5$; $y_0 = 60$
8	$p = 0{,}005$; $q = 0{,}002$ *
9	$s = 0{,}6$ *
10	$p = -0{,}02$; $q = 0{,}02$ *
** Значения остальных коэффициентов матрицы общего преобразования следует принимать такими же, как в матрице тождественного преобразования (см. приложение 1)*	

Общие методические указания к выполнению лабораторной работы №2

Исследование пространственных преобразований графических объектов

Лабораторно-исследовательские модули, поддерживающие проведение исследований двумерных и пространственных преобразований, загружаются файлами-приложениями соответственно ***«3D-преобразования»*** (пути доступа к файлам узнайте у ведущего преподавателя или лаборанта).

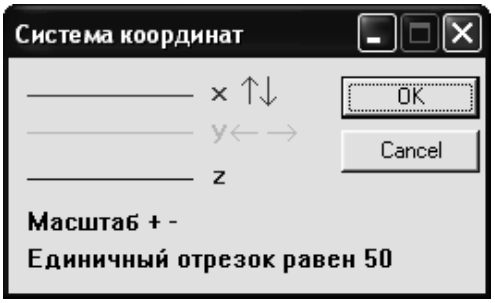


Рис.3. Окно «Система координат» модуля «3D-преобразования»

Интерфейс модуля ***«3D-преобразования»*** содержит одно полноэкранное окно, отведенное под ***область построений***, и два окна меньших размеров. Одно их них – ***«Панель управления»*** – помимо описанных выше возможностей предоставляет ряд дополнительных. Три кнопки расположенного на ней блока ***«Ортогональные проекции на координатные плоскости»*** (рис.2) предназначены для вывода в области построений ортогональных проекций пространственной сцены на координатные плоскости ***yz*** (кнопка ***«X = 0»***), ***xz*** (кнопка ***«Y = 0»***) и ***xy*** (кнопка ***«Z = 0»***), причем во всех этих случаях представляется вид сцены со стороны положительной координатной полуоси, перпендикулярной плоскости проекции. Кнопка ***«XYZ»*** панели ***«Аксонометрическая проекция»*** (рис.2) связана с выводом в области построений аксонометрической, причем изначально – изометрической проекции всей сцены, включая координатные оси. Окно интерфейса ***«Система координат»*** (рис.3) позволяет однозначно идентифицировать отображающиеся в области построений оси как координатные оси ***x, y*** и ***z*** пространственной правосторонней системы координат, определить масштабы по этим осям при текущем состоянии программы. Кроме того, в этом окне присутствуют подсказки по использованию клавиш клавиатуры ПК для осуществления поворотов всей сцены относительно координатных осей ***x*** (клавиши ***«↑» и «↓»***) и ***y*** (клавиши ***«←» и «→»***) и, соответственно, получения различного вида ее аксонометрических проекций, а также для увеличения (клавиша ***«+»***) или уменьшения (клавиша ***«–»***) масштаба выводимого в области построений изображения. Данные

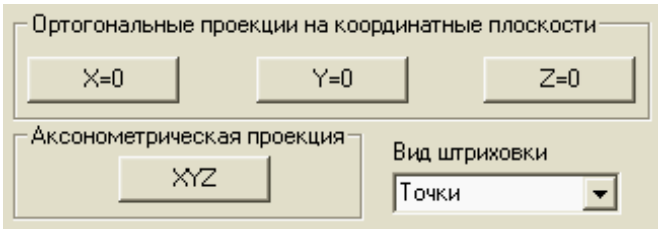


Рис.2. Дополнительные возможности панели управления модуля «3D-преобразования»

операции служат для обеспечения возможности наблюдать сцену под разными, в том числе и наиболее выгодными с точки зрения пользователя ракурсами, причем в требуемом ему масштабе.

Управление окнами, образующими интерфейсы модулей, осуществляется привычными для пользователей ОС Windows приемами.

Методические указания по выполнению работы

При пространственных преобразованиях графических объектов каждая точка $P\ (x,\ y,\ z)$ в пространстве однозначно отображается содержащей однородные координаты этой точки матрицей (координатным вектором) размером 1×4 вида $\begin{bmatrix} x & y & z & 1 \end{bmatrix}$. Отрезку прямой между точками (x_1, y_1, z_1) и (x_2, y_2, z_2) ставится в соответствие 2×4 матрица вида $\begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \end{bmatrix}$. Многогранник может быть представлен M×4 матрицей (где M – число вершин многогранника), содержащей однородные координаты его вершин: $\begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ \dots & \dots & \dots & \dots \\ x_M & y_M & z_M & 1 \end{bmatrix}$. Преобразования осуществляются путем умножения таких матриц на матрицу общего преобразования размером 4×4 вида $[T] = \begin{bmatrix} a & b & c & p \\ d & e & f & q \\ g & i & j & r \\ l & m & n & s \end{bmatrix}$. Результат преобразования зависит от конкретного вида матрицы преобразования. Если координатный вектор преобразованной точки $\begin{bmatrix} x' & y' & z' & h \end{bmatrix}$ содержит $h \neq 1$ и $h \neq 0$, результат нормализуют путем деления всех четырех составляющих однородных координат на h, т.е. приводят к виду $\begin{bmatrix} x^* & y^* & z^* & 1 \end{bmatrix}$, где $x^* = x'/h, y^* = y'/h, z^* = z'/h$. Равенство нулю координаты h в результате матричного умножения (координатный вектор преобразованной точки при этом имеет вид $\begin{bmatrix} x' & y' & z' & 0 \end{bmatrix}$) свидетельствует о том, что исходная точка преобразована в точку бесконечности, лежащую на луче, который идет из начала координат через точку (x', y', z').

Матрицы простых пространственных преобразований графических объектов представлены в приложении 3.

Ряд последовательных преобразований объекта можно комбинировать: предварительно рассчитав матрицу полного преобразования путем перемножения в строгой последовательности матриц отдельных

преобразований, применить ее для преобразования исходного объекта.

Требования к оформлению результатов выполнения лабораторных работ

Результаты проведения лабораторных исследований оформляются в виде индивидуальных (для каждого студента) отчетов. Пример оформления ***титульного листа*** отчета приведен в приложении 1. Основная часть отчета должна содержать:

- формулировку ***цели проведения исследований***;
- ***исходные данные*** для проведения исследований – приводятся координаты вершин исходного графического объекта;
- ***результаты проведения исследований*** – для каждого пункта (подпункта) программы работ, связанного с каким-либо преобразованием графического объекта, следует пояснить суть преобразования, привести матрицу общего преобразования с конкретными численными значениями всех ее коэффициентов, проиллюстрировать преобразование соответствующим фрагментом области построений; если на то есть указание в программе работ, необходимо сформулировать промежуточные выводы по результатам выполнения пункта (или нескольких пунктов) исследований;
- ***общие выводы по результатам исследований*** – составляются с использованием промежуточных выводов по соответствующим пунктам исследований; в общих выводах обязательно должны содержаться сведения о назначении отдельных коэффициентов матрицы общего преобразования и четырех ее подматриц в целом, а также о правилах реализации комбинированных преобразований графических объектов.

Матрицы простых пространственных преобразований

-

Локальное масштабирование: в ***a*** раз по координатной оси ***x***, в ***e*** раз по координатной оси ***y***, в ***j*** раз по координатной оси ***z***	$[T]=\begin{bmatrix} a & 0 & 0 & 0 \\ 0 & e & 0 & 0 \\ 0 & 0 & j & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Симметричное отражение относительно координатной плоскости ***yz*** (***x = 0***)	$[T]=\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Симметричное отражение относительно координатной плоскости ***xz*** (***y = 0***)	$[T]=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Симметричное отражение относительно координатной плоскости ***xy*** (***z = 0***)	$[T]=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Симметричное отражение относительно координатной оси ***x*** (поворот вокруг оси ***x*** на 180°)	$[T]=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Симметричное отражение относительно координатной оси ***y*** (поворот вокруг оси ***y*** на 180°)	$[T]=\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Симметричное отражение относительно координатной оси ***z*** (поворот вокруг оси ***z*** на 180°)	$[T]=\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Симметричное отражение относительно точки начала координат	$[T]=\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Сдвиги: вдоль оси ***x*** на ***dy + gz***, вдоль оси ***y*** на ***bx + iz***, вдоль оси ***z*** на ***cx + fy***	$[T]=\begin{bmatrix} 1 & b & c & 0 \\ d & 1 & f & 0 \\ g & i & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Поворот вокруг координатной оси ***x*** на произвольный угол θ	$[T]=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Поворот вокруг координатной оси ***y*** на произвольный угол ϕ	$[T]=\begin{bmatrix} \cos\phi & 0 & -\sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Поворот вокруг координатной оси ***z*** на произвольный угол ψ	$[T]=\begin{bmatrix} \cos\psi & \sin\psi & 0 & 0 \\ -\sin\psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Перемещения: вдоль координатной оси ***x*** на ***l***, вдоль координатной оси ***y*** на ***m***, вдоль координатной оси ***z*** на ***n***	$[T]=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ l & m & n & 1 \end{bmatrix}$
Проецирование в однородных координатах (если $h=px+qy+rz+1\neq 1$ и $h\neq 0$, результат необходимо нормализовать путем деления всех однородных координат на ***h***)	$[T]=\begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Общее масштабирование в $1/s$ раз (если $h=s\neq 1$, и $h\neq 0$ результат необходимо нормализовать путем деления всех однородных координат на ***h***)	$[T]=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & s \end{bmatrix}$
Параллельное ортографическое проецирование на координатную плоскость ***xy*** (***z*** = ***0***)	$[T]=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Варианты заданий к лабораторной работе №2

Дисциплина: «Компьютерная геометрия и графика»	
Лабораторная работа №2 ***Вариант задания №1***	
Координаты вершин исходного четырёхгранника	
$x_1 = 100$; $y_1 = 100$; $z_1 = 50$; $x_2 = 0$; $y_2 = 100$; $z_2 = 150$; $x_3 = 100$; $y_3 = 0$; $z_3 = 150$; $x_4 = 100$; $y_4 = 100$; $z_4 = 150$	
Пункт/подпункт программы работ	Параметры преобразования
2a	$a = 3$; $j = 1.5$ *
2d	$g = 1.5$ *
2f	$\phi = 30°$
3	$m = 100$; $n = 150$ *
4	$p = 0.03$; $q = 0.02$ *
5	$s = 2$*
6	$\phi = 50°$; $l = 100$; $n = 50$ *
8	$\chi = 105°$
9	$D = -50$
* *Значения остальных коэффициентов матрицы общего преобразования следует принимать такими же, как в матрице тождественного преобразования (см. приложение 1)*	

Примеры тестовых вопросов к экзамену

№Вопрос1

Какую часть окружности достаточно построить, чтобы затем путем отражений получить окружность целиком по первой версии алгоритма Брезенхема?

- 1/16
- 1/4
- 1/2
- 1/8

№Вопрос1

Какую часть окружности достаточно построить, чтобы затем путем отражений получить окружность целиком по второй версии алгоритма Брезенхема?

- 1/4
- 1/8
- 1/2
- 1/16

№Вопрос1

Какую часть эллипса достаточно построить, чтобы затем путем отражений получить эллипс целиком?

- 1/8
- 1/4
- 1/2
- 1/16

№Вопрос1

Выделите коэффициент, который осуществляет локальное масштабирование по координатной оси *x* в двумерных преобразованиях $\begin{bmatrix} a & b & p \\ c & d & q \\ m & n & s \end{bmatrix}$

- a
- b
- d
- c

№Вопрос1

Выделите коэффициент, который осуществляет локальное масштабирование по координатной оси *y* в двумерных преобразованиях $\begin{bmatrix} a & b & p \\ c & d & q \\ m & n & s \end{bmatrix}$

- d

- b
- a
- c

№Вопрос2

Выделите коэффициенты, которые осуществляют локальное масштабирование по координатной оси *x*и по координатной оси *y* в двумерных преобразованиях$\begin{bmatrix} a & b & p \\ c & d & q \\ m & n & s \end{bmatrix}$

- a
- d
- b
- s
- c

№Вопрос 2

Выделите коэффициенты, которые принимают значение 1 при симметричном отражении относительно координатной оси *x* двумерных преобразованиях$\begin{bmatrix} a & b & p \\ c & d & q \\ m & n & s \end{bmatrix}$

- a
- s
- b
- d
- c

№Вопрос 2

Выделите коэффициенты, которые принимают значение 1 при симметричном отражении относительно координатной оси *y* двумерных преобразованиях$\begin{bmatrix} a & b & p \\ c & d & q \\ m & n & s \end{bmatrix}$

- d
- s
- b
- a
- c

№Вопрос1 Что является основой воксельной модели?

- Все варианты
- Произвольная поверхность, обладающая свойствами гладкости, непрерывности и неразрывности

- Главным элементом описания является вершина, все остальные являются производными
- Возможность представлять внутренность объекта, а не только внешний слой; простая процедура отображения объемных сцен

№Вопрос1 Что является основой поверхностей свободных форм?

- Все варианты
- Возможность представлять внутренность объекта, а не только внешний слой; простая процедура отображения объемных сцен
- Произвольная поверхность, обладающая свойствами гладкости, непрерывности и неразрывности
- Главным элементом описания является вершина, все остальные являются производными

№Вопрос4

Установите соответствие достоинств:

1 полигональной модели,

2 воксельной модели,

3 поверхностей свободных форм

- Удобство масштабирования объектов
- Возможность представлять внутренность объекта
- Легкая процедура расчета координат каждой точки

№Вопрос4

Установите соответствие достоинств:

1 полигональной модели,

2 воксельной модели,

3 поверхностей свободных форм

- Небольшой объем данных для описания простых поверхностей
- Простое выполнение топологических операций
- Небольшой объем информации для описания достаточно сложных форм

№Вопрос4

Установите соответствие достоинств:

1 полигональной модели,

2 воксельной модели,

3 поверхностей свободных форм

- Аппаратная поддержка многих операций
- Возможность представлять внутренность объекта
- Возможность строить поверхности на основе скалярных данных без предварительной триангуляции

№Вопрос4

Установите соответствие недостатков:

1 полигональной модели,

2 воксельной модели,

3 поверхностей свободных форм

- Алгоритмы визуализации выполнения топологических операций довольно сложны

- Большое количество информации, необходимое для представления объемных данных
- Сложность при описании примитивов

№Вопрос4

Установите соответствие недостатков:

1 полигональной модели,

2 воксельной модели,

3 поверхностей свободных форм

- аппроксимация плоскими гранями приводит к значительной погрешности
- значительные затраты памяти, ограничивающие разрешающую способность, точность моделирования
- Сложность при описании примитивов

№Вопрос4

Установите соответствие недостатков:

1 полигональной модели,

2 воксельной модели,

3 поверхностей свободных форм

- моделирование поверхностей сложной формы
- Проблемы при увеличении или уменьшении изображения
- Сложность при описании примитивов

№Вопрос1

Алгоритм предназначенный для удаления невидимых линий содержит следующие этапы:1Предварительная сортировка по глубине, 2 Отсечение по границе ближайшего к точке наблюдения многоугольника, 3 Удаление многоугольников, экранируемых более близкими к точке наблюдения многоугольниками, 4 Если требуется, то рекурсивное разбиение и новая сортировка это:

- АлгоритмКэтмула
- алгоритм Вейлера-Азертона
- алгоритм Робертса
- алгоритмВарнока

№Вопрос1

Алгоритм предназначенный для удаления невидимых линий содержит следующие этапы: 1. Рекурсивно разбивается поверхность до тех пор, пока проекция элемента на плоскость изображения не будет покрывать не больше одного пикселя. 2. Определить атрибуты поверхности в этом пикселе и изобразить его. Это алгоритм:

- Алгоритм Вейлера-Азертона
- алгоритм Кэтмула
- алгоритм Робертса
- Алгоритм Варнока

№Вопрос1

Пространство изображения разбивается на 4, 16 или больше прямоугольников или полос. В предельном варианте можно использовать буфер размером в одну строку развертки. Это алгоритм:

- Интервальный алгоритм построчного сканирования
- Алгоритм построчного сканирования
- Алгоритм построчного сканирования криволинейных поверхностей

№Вопрос1

Сцена обрабатывается в порядке прохождения сканирующей прямой. В объектном пространстве это соответствует проведению секущей плоскости, перпендикулярной пространству изображения. Это алгоритм:

- Алгоритм построчного сканирования
- Алгоритм построчного сканирования криволинейных поверхностей
- Интервальный алгоритм построчного сканирования

№Вопрос1

Сканирующая строка разбивается проекциями точек пересечения ребер многоугольников на интервалы, затем в каждом из интервалов выбираются видимые отрезки. В этой ситуации их уже можно отсортировать по глубине. Это алгоритм:

- Алгоритм построчного сканирования
- Интервальный алгоритм построчного сканирования
- Алгоритм построчного сканирования криволинейных поверхностей

№Вопрос1

Элементы сцены изображаются в последовательности от наиболее удаленных от наблюдателя к более близким. При экранировании одних участков сцены другими невидимые участки просто закрашиваются. Это метод:

- Метод трассировки лучей
- Метод приоритетов: художника
- Метод приоритетов: плавающего горизонта
- Метод двоичного разбиения пространства
- Метод Z-буфера

№Вопрос1

Метод применим когда объект представляется в виде набора кривых или ломаных линий. Это метод:

- Метод трассировки лучей
- Метод приоритетов: плавающего горизонта
- Метод приоритетов: художника
- Метод двоичного разбиения пространства
- Метод Z-буфера

№Вопрос1

Некоторая область памяти предназначена для запоминания глубины (расстояния от картинной плоскости) каждого видимого пикселя в пространстве изображения. Это метод:

- Метод трассировки лучей

- Метод Z-буфера
- Метод приоритетов: художника
- Метод двоичного разбиения пространства
- Метод приоритетов: плавающего горизонта

№Вопрос1

Метод для простых непрозрачных поверхностей можно представить следующим образом. Создать список объектов, содержащий:

полное описание объекта: тип, поверхность, характеристики, тип оболочки;

описание оболочки: центр и радиус для сферы или шесть значений для параллелепипеда.

Выполнить для каждого объекта трехмерный тест на пересечение с оболочкой.

Это метод:

- Метод Z-буфера
- Метод трассировки лучей
- Метод приоритетов: художника
- Метод двоичного разбиения пространства
- Метод приоритетов: плавающего горизонта

№Вопрос1

Суть метода сводится к методу художника, для сцен содержащих несколько объектов с последующим разбиением пространствами… Это метод:

- Метод Z-буфера
- Метод двоичного разбиения пространства
- Метод приоритетов: художника
- Метод трассировки лучей
- Метод приоритетов: плавающего горизонта

№Вопрос2

Назовите два основных вида проекций, определяемых типом пучка лучей

- ортографическая проекция
- изометрическая проекция
- центральная проекция
- параллельная проекция

№Вопрос2

Назовите четыре вида параллельных проекций.

ортографическая проекция

- кабинетная проекция
- горизонтальная косоугольная
- центральная проекция
- изометрическая проекция
- триметрическая
- диметрическая

№Вопрос1

Что такое перспективное укорачивание?

- по мере увеличения расстояния от центра до объекта размер получаемой проекции уменьшается
- соотношение между центром проекции и проекционной плоскостью.
- расстояние вдоль каждой из главных координатных осей (в общем случае с различными масштабными коэффициентами)
- отрезки в кабинетной проекции перпендикулярные проекционной плоскости, после проецирования составляют 1/2 их действительной длины, что более соответствует визуальному опыту.

№Вопрос1

Какиецентральные проекциичаще используются и дают наиболее реалистическую картину с одной, с двумя или с тремя точками схода?

- без точек схода
- с двумя точками схода
- с тремя точками схода
- с одной точкой схода

№Вопрос2

Поверхности каких фигур называются развертывающимися?

- шар
- пирамида
- конус
- цилиндр
- многогранник

№Вопрос4

Сопоставить проекциям соответствие их свойствам:

1.стереографическая(конформная)

2. гномоническая.

3.ортографическая

4.Меркатора

5.поликоническая картографическая проекция

- карты, сохраняющие углы, называются…
- любая дуга большого круга на поверхности земного шара переходит в прямую на карте
- эта проекция получается при проецировании на плоскость, касательную к сфере в центре изображаемого явления, с помощью лучей, перпендикулярных этой плоскости
- достигается за счет растягивания цилиндра за полюсы, при этом в верхней и нижней части этого цилиндра масштаб становится очень искаженным
- проекция, строящаяся с помощью ряда конусов, касательных к земному эллипсоиду

№Вопрос1

По какому критерию инициализируется пиксель в алгоритме Брезенхема для генерации прямой?

- растровая развертка

- что анализируется лишь знак этого смещения
- по методу цифрового дифференциального анализатора
- затравочное заполнение

№Вопрос1
Что такое эффект полос Маха?
- Нет правильного ответа
- граница равномерно освещенной области кажется более яркой по сравнению с внутренними частями
- видимая освещенность того или иного участка поверхности не зависит от положения наблюдателя
- при расчете интенсивности получится очень контрастная картина, т.к. участки поверхности, на которые лучи от источника не попадают напрямую, останутся абсолютно черными.

№Вопрос4
Установить соответствие: Какой параметр интерполируется:
1.при плоском закрашивании
2.при закрашивании методом Гуро
3. при закрашивании методом Фонга?
- используются нормали к плоским граням
- нормали к аппроксимируемой поверхности, построенные в вершинах многогранника.
- интерполяция вектора нормали к поверхности на сканирующей строке.

№Вопрос4
Установить соответствие: Какой параметр интерполируется:
1.при плоском закрашивании
2.при закрашивании методом Гуро
3. при закрашивании методом Фонга?
- используются нормали к плоским граням
- нормали к аппроксимируемой поверхности, построенные в вершинах многогранника.
- интерполяция вектора нормали к поверхности на сканирующей строке.

№Вопрос4
Установить соответствие между вариантами и методами:
1.вариант дает изображение ребристой поверхности с очень контрастными переходами от одной грани к другой
2.вариант дает более гладкое изображение, но в районе бликов отчетливо наблюдаются линии ребер, хотя и сглаженные
3.вариант получился наиболее гладким, зеркальные блики имеют достаточно реалистичную форму
- плоское закрашивание
- закрашивание по методу Гуро
- закрашивание по методу Фонга

№Вопрос1

Какой метод закрашивания хорошо работает только с диффузной моделью отражения?

- все три
- закрашивание по методу Гуро
- плоское закрашивание
- закрашивание по методу Фонга

№Вопрос1

В каком методе закрашивания больше всего проявляется эффект полос Маха?

- Во всех трех
- плоское закрашивание
- закрашивание по методу Гуро
- закрашивание по методу Фонга

Список литературы

Основная литература

1. Боресков, А.В. Компьютерная графика: динамика, реалистические изображения / А.В. Боресков, Е.В. Шикин. - Москва : Диалог-МИФИ, 1995. - 280 с. : ил.,табл., схем. - Библиогр. в кн. - ISBN 5-86404-061-4 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=54731 (11.03.2021).

2. Шикин, Е.В. Компьютерная графика: полигональные модели / Е.В. Шикин, А.В. Боресков. - Москва : Диалог-МИФИ, 2005. - 462 с. : табл., граф., схем., ил. - Библиогр. в кн. - ISBN 5-86404-139-4 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=89300 (03.02.2022).

3. Гонсалес, Р. Цифровая обработка изображений : практические советы / Р. Гонсалес, Р. Вудс ; пер. П.А. Чочиа, Л.И. Рубанова. - 3-е изд., испр. и доп. - Москва : Техносфера, 2012. - 1104 с. : ил.,табл., схем. - (Мир цифровой обработки). - ISBN 978-5-94836-331-8 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=233465 (26.02.2022).

Дополнительная литература

1. Ковтанюк, Ю.С. CorelDRAW X3 на примерах / Ю.С. Ковтанюк. - Москва : Диалог-МИФИ, 2007. - 352 с. : табл., ил., схем. - ISBN 5-86404-211-0 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=54775 (21.02.2022)

2. Молочков, В.П. Основы работы в Adobe Photoshop CS5 / В.П. Молочков. - Москва : Интернет-Университет Информационных Технологий, 2011. - 236 с. ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=234169 (21.02.2022).

3. Олби, Т. Компьютерная графика в кинематографе. Создание фильма «Призрачный воин» / Т. Олби ; пер. И. Чумаченко. - Москва : СОЛОН-ПРЕСС, 2008. - 368 с. - ISBN 5-98003-254-1 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=227070 (21.02.2022)

4. Васильев С.А. Компьютерная графика и геометрическое моделирование в информационных системах [Электронный ресурс] : учебное пособие для бакалавров направлений подготовки 230100 «Информатика и вычислительная техника», 230400 «Информационные системы и технологии» очной формы обучения / С.А. Васильев, И.В. Милованов. — Электрон. текстовые данные. — Тамбов: Тамбовский государственный технический университет, ЭБС АСВ, 2015. — 81 с. — 978-5-8265-1432-0. — Режим доступа: http://www.iprbookshop.ru/64103.html

5. Компьютерная графика. Часть 1 [Электронный ресурс] : методическое пособие по выполнению домашних заданий и контрольных работ / В.Н. Смоляков [и др.]. — Электрон. текстовые данные. — Ростов-на-Дону: Северо-Кавказский филиал Московского технического университета

связи и информатики, 2010. — 134 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/61297.html

1. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины

1. *Образовательные фильмы на различные темы. Лекции в ведущих российских и зарубежных вузах. Научная конференция или научно -популярная лекция по интересующему вопросу Открытый образовательный видеопортал UniverTV.ru.* http://univertv.ru/video/matematika/
2. *Интернет-библиотека и пользовательские сервисы для полноценной работы с библиотечными фондами. Свободный доступ к электронным учебникам, справочным и учебным пособиям. Электронная библиотека IQlib образовательных и просветительских изданий.* http ://www. iqlib. ru/
3. *ГОСТ Р 7.0.5.-2008. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая ссылка. Общие требования и правила составления. Введ. 2009-01-01. - М.: Стандартинформ, 2008. - 22 с.* (http://gostexpert.ru/gost/gost-7.0.5-2008, *Российское образование" Федеральный портал. Каталог образовательных интернет-ресурсов. URL:* http://www.edu.ru/index.php*;*
4. *Федеральное агентство по образованию РФ. URL: http ://* www.ed.gov.ru */*
5. *Официальный сайт Министерства образования и науки Российской Федерации. URL: http:// mon.gov*
6. Электронный образовательные ресурсы [Электронный ресурс]: Курс лекций по компьютерной геометрии и графике Гаджиев А.М. /Дагестанский гос. ун-т. – Махачкала, 2016 – Режим доступа: http://elib.dgu.ru, свободный (дата обращения: 21.02.2023).