Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Оренбургский государственный университет»

Кафедра прикладной информатики в экономике и управлении

ПРИМЕНЕНИЕ КОМПЬЮТЕРНОЙ ГРАФИКИ ДЛЯ РЕШЕНИЯ ЭКОНОМИЧЕСКИХ И ИНЖЕНЕРНЫХ ЗАДАЧ

Учебное пособие

Рекомендовано ученым советом федерального государственного бюджетного образовательного учреждения высшего образования «Оренбургский государственный университет» для обучающихся по образовательным программам высшего образования по направлениям подготовки 09.03.03 Прикладная информатика и 38.03.05 Бизнес-информатика

Оренбург
2018

УДК 004.4:657.1 (075.8)
ББК 32.972.5я73+65.052я73
П76

Рецензент – профессор, доктор экономических наук Н.В. Спешилова
Авторы: А.М. Горюнова, Т.В. Омельченко, П.Н. Омельченко, Д.А. Муслимов

П76 Применение компьютерной графики для решения экономических и инженерных задач : учебное пособие / А.М. Горюнова, Т.В. Омельченко, П.Н. Омельченко, Д.А. Муслимов; Оренбургский гос. ун-т. – Оренбург : ОГУ, 2018.

ISBN 978-5-7410-2019-7

В учебном пособии представлен материал по основам компьютерной графики. Рассмотрены теоретические вопросы по фрактальной, растровой и векторной графике. Кроме того, описаны особенности построения трёхмерных моделей и 3D анимации. Приведены примеры практического применения инструментов компьютерной графики для решения экономических и инженерных задач.

Учебное пособие предназначено для обучающихся по направлениям подготовки 09.03.03 Прикладная информатика и 38.03.05 Бизнес-информатика, очной и заочной форм обучения, а также может быть полезно студентам других направлений подготовки при изучении возможностей компьютерной графики и особенностей ее практического применения.

УДК 004.4:657.1 (075.8)
ББК 32.972.5я73+65.052я73

ISBN 978-5-7410-2019-7

Содержание

Введение

Компьютерная графика (КГ) - это отдельная область информационных технологий, которая включает все аспекты автоматизированного формирования и обработки изображений. Под компьютерной графикой следует понимать и новые эффективные технические средства для проектирования, инструменты для конструкторов и исследователей, программные продукты и машинные языки, новые дисциплины в науке и образовании, родившиеся на базе синтеза аналитической, прикладной и начертательной геометрии, программирования, методов вычислительной математики и тому подобное.

Зародившись в середине XX века, КГ в начале предоставляла возможность вывода на экране лишь нескольких десятков отрезков. На сегодняшний момент средства компьютерной графики позволяют создавать реалистические изображения, не уступающие фотографическим снимкам. Современное состояние КГ характеризуется созданием различного рода аппаратного и программного обеспечения для получения изображений самого разнообразного вида и практического назначения, начиная от простых чертежей и заканчивая реалистичными образами натуральных объектов.

Преимущество компьютерной графики заключается в наглядном отображении таких сложных геометрических объектов, которые ранее не изображались математиками.

Современная компьютерная графика - это достаточно непростая и трудоемкая, основательно проработанная и разнообразная по наполнению и сферам приложения научно-техническая дисциплина. Такие ее разделы, как геометрические преобразования, способы описания кривых и поверхностей, к настоящему времени уже исследованы достаточно полно, тогда как ряд областей продолжает и сегодня активно развиваться, например, методы растрового сканирования, удаление невидимых линий и

поверхностей, моделирование цвета и освещенности, текстурирование, создание эффекта прозрачности и полупрозрачности и другие.

В связи с этим, основными задачами автоматизированного геометрического моделирования, конструирования и компьютерной графики являются синтез с помощью компьютера и анализ геометрических объектов, решение задач геометрического характера.

Графика все шире проникает в экономику – сегодня практически отсутствуют документы, созданные без применения какого–либо графического объекта. Специально разработано различное программное обеспечение для сосредоточения пользователей при работе не на графике, а на содержании документов.

В учебном пособии изложены теоретические сведения о базовых алгоритмах различных видов компьютерной графики для генерации простых фигур (отрезка и окружности), заполнения многоугольников, отсечения невидимых линий, двумерные и пространственные преобразованиях фигур.

Учебное пособие предназначено для студентов, обучающихся по направлениям подготовки 09.03.03 Прикладная информатика и 38.03.05 Бизнес-информатика, очной и заочной форм обучения, а также может быть полезно студентам других направлений подготовки при изучении возможностей компьютерной графики и особенностей ее практического применения.

1 История развития компьютерной графики

Становление компьютерной графики началось в 1940-х годах и продолжает развиваться до сих пор. Не секрет, что именно графика способствовала быстрому росту быстродействия компьютеров. Её развитие можно разделить на следующие исторические этапы:

1. **1940-1970 гг.**

Период принято называть эрой до персональных компьютеров. В силу того, что у пользователей раньше отсутствовал доступ к экрану компьютера, КГ развивалась на математическом уровне и выводилась на принтере в виде текста, и лишь на значительном расстоянии напоминала изображение.

2. **1950-е годы.**

Период векторных дисплеев. Качество изображений на векторных дисплеях было довольно низким, кроме того, достаточно примитивным. В отличие от распространенных растровых дисплеев, где луч каждый раз проходит слева-направо и сверху-вниз для отрисовки каждой строки, в векторных дисплеях луч двигается по линиям, определяющим изображение.

3. **1960-е г.**

В эти годы появились цифровые дисплеи, что открыло новые возможности для компьютерной графики в сфере выполнения задач цифровой обработки изображений.

В 1968 г. появилась разработанная компьютерная математическая модель, способная имитировать передвижения кошки. Машина БЭСМ-4 реализовывала алгоритм работы программы решения дифференциальных уравнений, тем самым формируя мультипликационный фильм «Кошечка». Этот фильм для того периода времени стал настоящим прорывом. Для визуализации процесса использовался алфавитно-цифровой принтер.

В этот же период появились графопостроители.

4. 1971-1985 гг.

После появления первых персональных компьютеров у пользователей появился доступ к дисплеям. Роль графики резко возросла, но наблюдалось очень низкое быстродействие компьютера. Несмотря на это, именно этот период можно охарактеризовать как зарождение реальной графики и появление цветного изображения.

5. Конец 80-х

Возникло новое направление рынка в сфере развития аппаратных и программных систем сканирования, автоматической оцифровки.

Кроме того, появились технологии мультимедиа, анимации и возможности выводить цветное изображение. К графике добавились обработка звука и видеоизображения, что позволило расширить общение пользователя с компьютером.

6. 1991-2008 гг.

Период характеризуется формированием графики в современном виде - Virtual Reality. Появились датчики перемещения, благодаря которым компьютер меняет изображения при помощи сигналов посылаемых на него. Новшеством можно считать изобретение стереоочков. На основе их высокого быстродействия реализуется имитация реального мира. Тем не менее, из-за предостережений представителей медицины происходит замедление развития данной технологии. Это объясняется тем, что виртуальная реальность способна очень сильно нарушить психику человека, благодаря мощному воздействию цвета на неё [1].

Сегодня в компьютерной графике выделяют 4 принципа формирования изображения:

- фрактальная графика, основанная на геометрической фигуре, которая является частью точной её копии в разы большей, чем одна часть;

- растровая (точечная) графика, предназначенная для создания фотореалистичных изображений;

- векторная графика, способная формировать сложные геометрические объекты;

- трёхмерная графика – позволяет создавать объёмные зрительно-подобные изображения и объекты.

2 Представление цвета в компьютере

Основополагающими понятиями в компьютерной графике принято считать свет и цвет.

Свет можно рассматривать двояко: как поток частиц различной энергии или как поток электромагнитных волн.

Понятие же цвета напрямую связано с восприятием человеческим глазом света. Принцип восприятия представлен на рисунке 2.1.

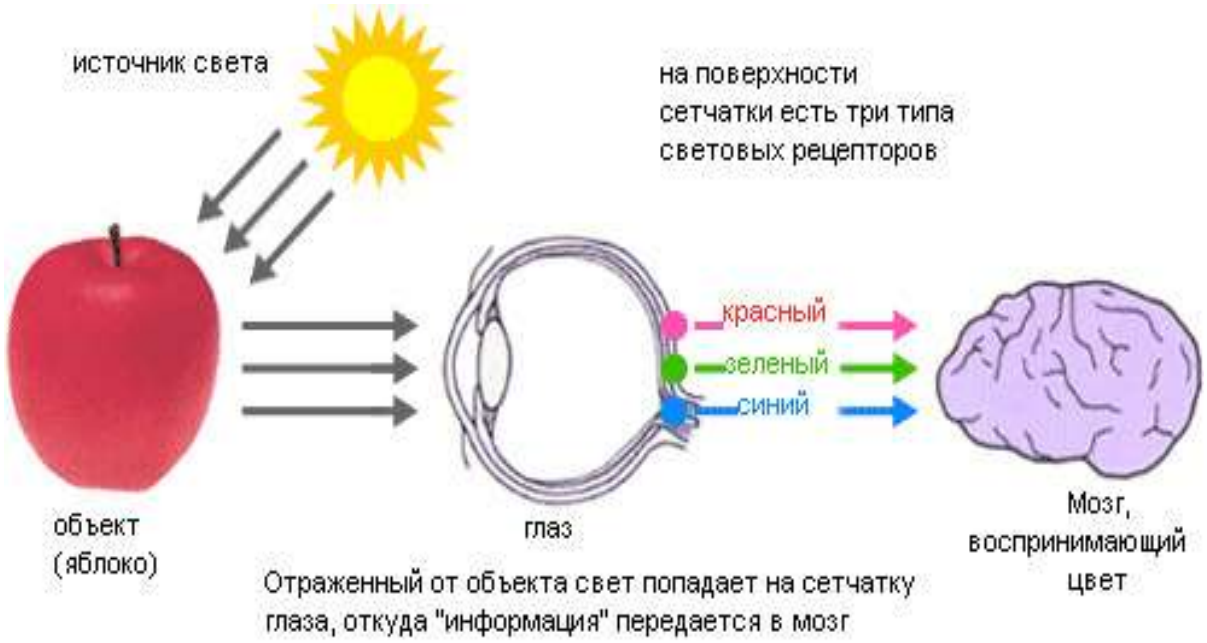


Рисунок 2.1 – Принцип восприятия человеческим глазом света

Как видно из рисунка ощущение света формируется мозгом в результате анализа светового потока, попадающего на сетчатку глаз.

Источник или объект является **ахроматическим,** если наблюдаемый свет содержит все видимые длины волн в приблизительно равных количествах. К ахроматическим цветам относят белый, черный и различные градации серого. Ахроматические цвета и их градация представлены на рисунках 2.2 и 2.3 соответственно.

Так, белыми выглядят те объекты, которые ахроматически отражающие более 80 % света белого источника, черными же, отражающие менее 3 %.

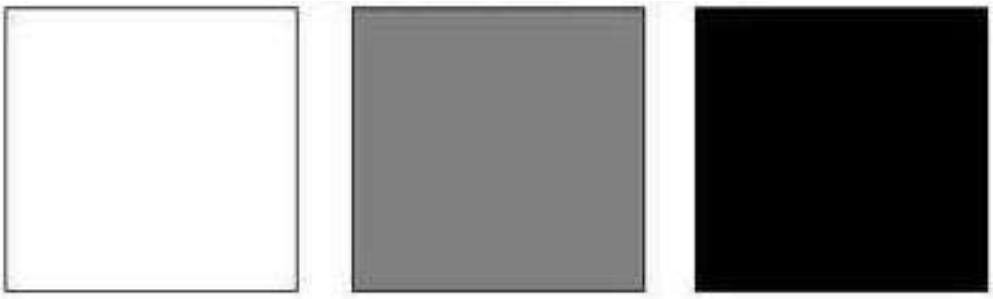


Рисунок 2.2 – Ахроматические цвета

Рисунок 2.3 – Изменение цвета по светлоте

Если воспринимаемый свет содержит длины волн в неравных количествах, то он называется **хроматическим.**

Принято считать, что в глазе человека существует три группы цветовых рецепторов, каждая из которых чувствительна к определенной длине световой волны. В каждой группе формируется один из трех основных цветов - красный, зеленый и синий.

Спектр хроматических цветов отражен на рисунке 2.4.

Рисунок 2.4 – Спектр хроматический цветов

Если длины волн светового потока сосредоточены у верхнего края видимого спектра, что около 700 Нм, то свет будет восприниматься красным. При концентрации длины волн у нижнего края видимого спектра - 400 Нм, то свет представляется синим. В случае длины волн в середине видимого спектра, что соответствует 550 Нм, то свет человеком воспринимается зеленым.

На основе ряда экспериментов, проведенных на базе этой гипотезы, были построены кривые реакции глаза, отраженные на рисунке 2.5.

Физические характеристики светового потока характеризуются параметрами **яркости, мощности и освещенности**. Визуальные параметры ощущения цвета определяются **светлотой**, **насыщенностью** и **цветовым тоном** [2].

Приведем значение данных понятий.

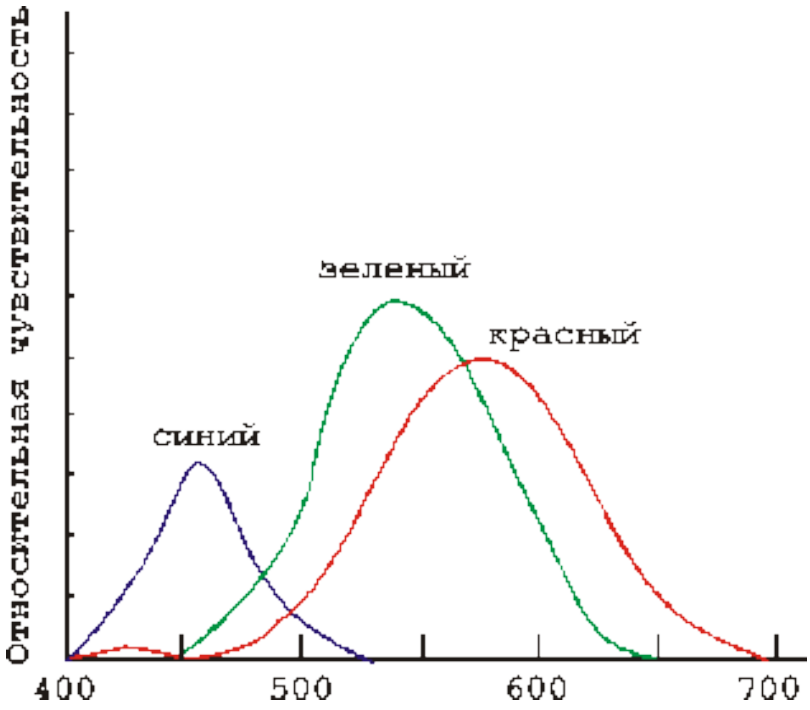


Рисунок 2.5 – Кривые реакции глаза

Светлота – это различимость участков с различной степенью силы отражения света. Порогом считается – минимальное отличие между яркостью различимых по светлоте объектов.

Насыщенность цвета – это степень отличия рассматриваемого цвета от монохроматического, иначе говоря «чистого» излучения того же светового тона. Проще говоря, насыщенность определяет силу разбавления данного цвета белым и позволяет отличать оттенки: например, розовый от красного, а голубой от синего. Цветовой тон позволяет различать основные цвета, такие, как красный, зеленый, синий.

Цвет в компьютере также выражается в виде комбинации нулей и единиц.

Можно выделить различные варианты описания цвета. Рассмотрим основные из них:

- **режим Bitmap** является самым элементарным побитовым представлением цвета. Цвет пикселя или векторного объекта кодируется одним битом. При этом, закодировать можно только черный и белый цвет

(или любой другой набор их двух цветов, например, красный и зеленый). В режиме Bitmap обычно представляется текст и штриховая графика – черные рисунки на белом фоне.

- **режим Grayscale.** В случае использования данного режима кодирование яркости тоновых черно-белых иллюстраций формируется в один байт, что позволяет получить для каждой точки $2^8 = 256$ градаций серого цвета. Этого вполне достаточно для черно-белой тоновой графики, в большей же детализации нет необходимости.

- **индексный режим.** Цвет кодируется одним байтом, всего может получиться те же 256 цветов. Безусловно, малое количество цветовых кодов снижает качество конечного изображения.

Если же при формировании изображений применять именно индексный режим, то в данном случае будет вызвана имеющаяся в программе индексная палитра, а затем и определяется нужный цвет. В случае, если отсутствует надобность проводить более детальный выбор, то на этом раскраска заканчивается.

Индексный режим чаще всего применяется в Интернете, поскольку существенную роль играет время загрузки веб-страницы и уже потом качество и, следовательно, размер страницы. Экономия на описании цвета тоже дает выигрыш в размере Интернет-баннеров.

- режим **True Color** или цветовая модель **RGB.** В основе этой модели заложены три цвета: красный, зеленый и синий - Red, Green, Blue (рисунок 2.6).

Зрение человека устроено так, что любой видимый цвет можно получить путем смешения этих трех основных цветов.

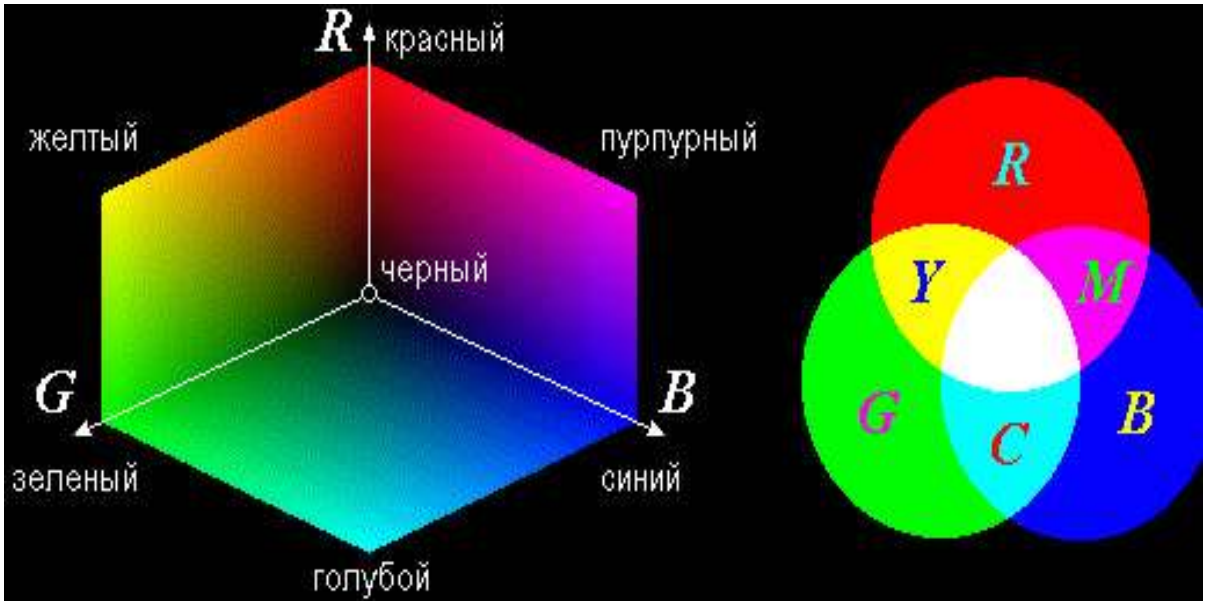


Рисунок 2.6 – Цветовая модель RGB

Модель хорошо подходит для объектов, испускающих свет, таких как экраны мониторов. Сканеры, цифровые камеры и прочие устройства ввода графики в компьютер также работают в модели RGB, в силу того, что итоговое изображение представляется все равно на экране монитора.

Для кодирования яркости каждого из основных цветов используется по 256 значений, то есть один байт или 8 разрядов. Всего на кодирование цвета одной точки необходимо использовать 24 разряда. Таким образом, система кодирования в целом обеспечивает однозначное определение $2^{24} \approx 16{,}8$ миллионов различных цветов.

В действительности такого большого числа цветов на экране не требуется, поскольку человек различает около 200 000 цветовых оттенков. Но принцип системы кодирования заключается в том, что на каждый канал отводится не меньше, чем по одному байту, а при обработке файлов бывает, что избыток оттенков может оказаться полезным и даже необходимым.

- цветовая модель CMYK. Основными цветами модели являются голубой (Cyan), пурпурный (Magenta), желтый (Yellow) и черный (Black) (рисунок 2.7).

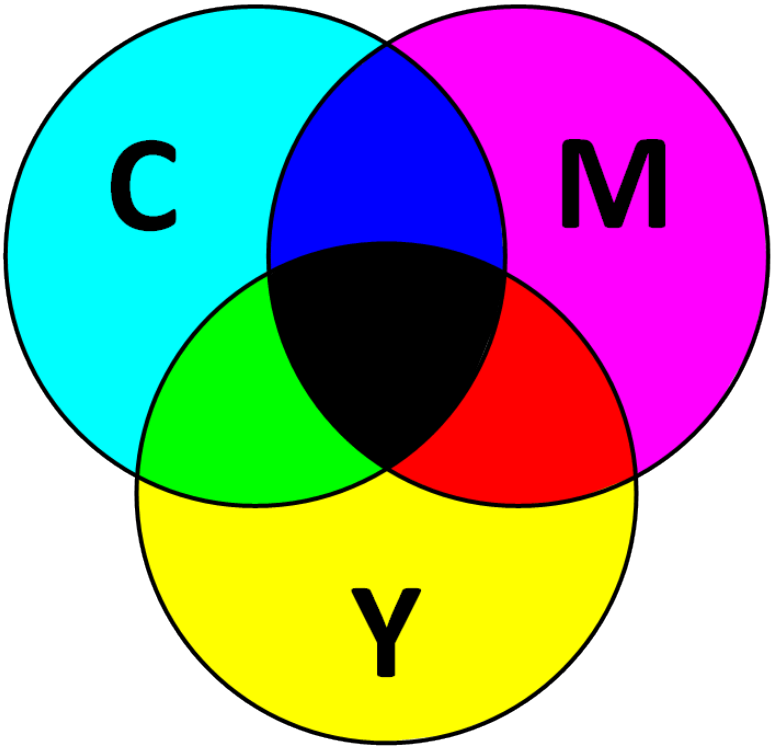


Рисунок 2.7 – Цветовая модель CMYK

CMYK-модель используется когда требуется описать отраженный цвет, преимущественно в полиграфии. Большинство печатной продукции в цвете реализует модель CMYK. Стоит отметить, что печать цветного изображения, даже на офисном принтере, выполняет автоматическую конвертацию RGB в CMYK.

При отражении света от поверхности часть света поглощается, и цвет определяется теми световыми волнами, которые поверхностью не поглотились.Чем больше было положено различных цветов, тем выше поглощение, ниже отражение и следовательно темнее выглядит поверхность. Смешение всех имеющихся красок будет выдавать черный цвет, а удаление же какого-либо поглощения позволит получать полное отражение, например, как в зеркале. Если на зеркало падает белый цвет, то окрашивание равно 0.

При испускании света иная ситуация, то есть чем больше испускается световых волн, тем ярче изображение. Равномерное испускание всех

световых волн соответствует белому цвету, а отсутствие испускания (если пренебречь отражением) характеризуется черным цветом.

Таким образом, как было сказано выше, модели RGB и CMYK описывают противоположные процессы. Поэтому в модели RGB черный цвет определяется по нулевым индексам, а всем единицам соответствует белый. В модели CMYK действует иной принцип: все 0 - это белый цвет, а 1 – черный.

В теории модели RGB и CMY (без K) зеркально противоположны: основные цвета одной модели являются дополнительными для другой и наоборот. Данный принцип представлен на рисунках 2.6 и 2.7.

Кроме того, при переходе к реально используемым при печати краскам теория не действительна. Смешение голубого, пурпурного и желтого цветов дает не черный, а темно-бурый цвет. Более того, черный является основным в полиграфии, потому что обычный текст преимущественно печатается черным, кроме того много выпускается именно черно-белой продукции, а не цветной. В силу выше сказанного, возникает необходимость во введении отдельной, черной координате в цветовой модели.

3 Фракталы

3.1 Понятие «фрактал». Историческая справка

На сегодняшний день **фрактальная графика** является второй по росту популярности из четырёх видов компьютерной графики.

Термин «фрактал» был введен Бенуа Мандельбротом в 1970-х гг. В латыни оно буквально означает «ломанный» или «дробленный». Сегодня под словом «**фрактал**» принято понимать графическое изображение структуры, которая в более крупном масштабе подобна сама себе.

Не смотря на то, что математическая база теории фракталов была заложена достаточно давно, однако развиться она смогла лишь с появлением вычислительных устройств.

Исследования Бенуа были связаны с решением задачи передачи данных на расстояние. В ходе исследований ученым были выявлены проблемы больших потерь вследствие шумовых помех. В связи с этим была поставлена задача понимания того как спрогнозировать возникновение шумовых помех в электронных схемах, в случае неэффективности статистического метода [3].

В результате измерений шума Мандельбротом была выявлена закономерность - графики шумов в разном масштабе выглядели одинаково. Идентичная картина наблюдалась вне зависимости от того за какой временной промежуток были построены графики: один день, неделя или час.

Фрактальный рисунок не имеет идентичных элементов, но обладает подобностью в любом масштабе. Построить рисунок с высокой степенью детализации вручную было невозможно раньше, потому как требовалось большое количество вычислений.

Одим из первых изображений фрактала является графическая интерпретация множества Мандельброта, которая появилось в качестве результата исследованиям Гастона Мориса Жюлиа.

Научная работа заключалась в графическом представлении множества, если построить его на основе простой формулы, проитерированной циклом

обратной связи. Проще говоря, для конкретного числа необходимо найти по формуле новое значение, после чего снова его подставить в формулу и получить еще одно значение. Результатом является большая последовательность чисел.

Мандельброт был первым, кто использовал компьютер для просчета классического фрактала. Обработав последовательность, состоящую из большого количества значений, Бенуа перенес результаты на график.

Теория фракталов достаточно быстро нашла свое практическое приложение, в силу того, что тесно связана с визуализацией самоподобных образов. Неудивительно, что первыми, кто взял на вооружение алгоритмы и принципы построения необычных форм, стали художники.

3.2 Классификация фракталов

3.2.1 Геометрические фракталы

Геометрические фракталы наиболее наглядные. Фракталы такого класса получают с помощью некоторой ломаной (в двухмерном случае) или поверхности (в трёхмерном простанстве), называемой **генератором.** За один шаг алгоритма каждый из отрезков, составляющих ломаную, заменяется на ломаную-генератор, в соответствующем масштабе. В результате бесконечного числа повторения этой процедуры, получается **геометрический фрактал.**

Приведем в качестве примера один из таких фрактальных объектов – триадная кривая Коха (рисунок 3.1).

Построение кривой Коха начинается с отрезка единичной длины - это нулевое поколение кривой. Затем каждое звено (в нулевом поколении один отрезок) заменяется на образующий элемент – на рисунке 3.1 обозначено через n=1. В результате чего формируется новое поколение кривой Коха.

В 1-ом поколении кривая состоит из 4-х прямолинейных звеньев, каждое из которых длиной по $\frac{1}{3}$. Для получения 3-го поколения аналогичные шаги повторяются. Следовательно, создавая новые последующие поколения, все звенья предыдущего требуется заменить уменьшенным образующим элементом. Кривая n-го поколения при любом конечном n называется **предфракталом**. Рисунок 3.1 отражает 5 поколений кривой. При n стремящемся к бесконечности кривая Коха становится фрактальным объектом.

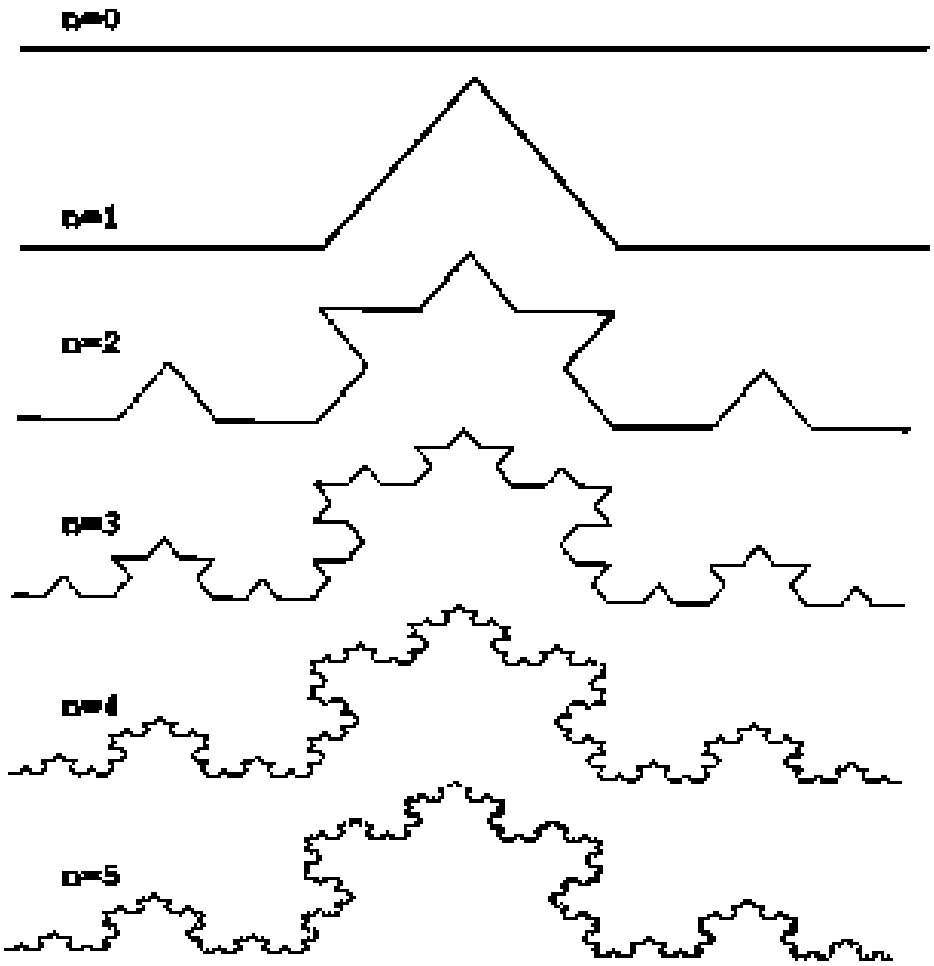


Рисунок 3.1 - Построение триадной кривой Коха

Для того, чтобы получить другой фрактальный объект требуется поменять правила, на основании которых проводится построение. Пусть

образующим элементом будут 2 одинаковых отрезка, соединенных под прямым углом. В нулевом поколении заменим единичный отрезок на этот образующий элемент, учитывая необходимость помещения угла сверху. Следовательно, замена провоцирует смещение середины звена. При формировании новых поколений соблюдается правило: самое 1-ое слева звено заменяется на образующий элемент так, чтобы середина звена смещалась влево от направления движения, а при замене последующих элементов, направления смещения середин отрезков должны чередоваться. На рисунке 3.2 показаны несколько первых поколений и 11-е поколение кривой, построенной по разобранному выше алгоритму [4]. Предельная фрактальная кривая (при n стремящемся к бесконечности) называется **драконом Хартера-Хейтуэя.**

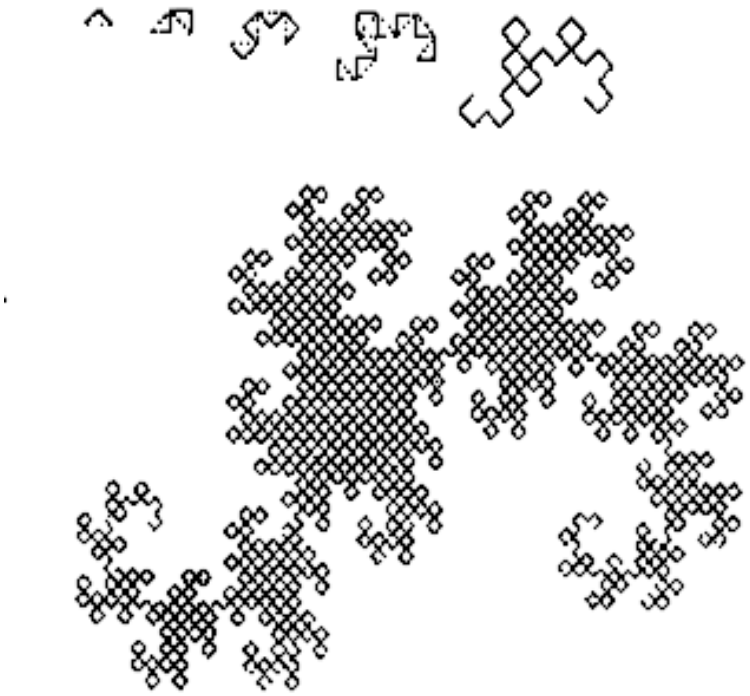


Рисунок 3.2 - Построение «дракона» Хартера-Хейтуэя

Применение геометрических фракталов в компьютерной графике позволяет получить изображение деревьев, кустов, береговой линии.

Двухмерные геометрические фракталы используются при создания объемных текстур (рисунка на поверхности объекта).

3.2.1.1 Кривая дракона. Построение кривой с помощью L-системы

L-системой (точнее, её простейшей разновидностью, детерминированной контекстонезависимой L-системой) называют набор, состоящий из алфавита, аксиомы, и множества правил.

Алфавитом называется конечное множество, а его элементы - символами. Природа символов не важна, их единственная функция - отличаться друг от друга. Строкой над алфавитом назовём конечную последовательность символов алфавита.

Аксиома - это некоторая строка над алфавитом.

Каждое **правило** - это пара, состоящая из предшественника и последователя.

Предшественник - это символ алфавита, а **последователь** – строка над алфавитом.

Например, вид пары: A→→FBFA+HFA+FB-FA (стрелка отделяет предшественника от последователя). В списке правил символы-предшественники должны быть уникальными.

Как только L-система определена, она начинает развиваться в соответствии с её правилами. Начальным состоянием L-системы является её аксиома. При дальнейшем развитии эта строка, описывающая состояние, будет меняться. Развитие L-системы происходит циклически. В каждом цикле развития строка просматривается от начала к концу, символ за символом. Для каждого символа ищется правило, для которого этот символ служит предшественником. Если такого правила не нашлось, символ оставляется без изменений. Иными словами, для тех символов X, для которых нет явного правила, действует неявное: X→→X. Если же соответствующее правило найдено, символ-предшественник заменяется на строку-последователь из этого правила.

Для отрисовки кривой дракона с помощью L-системы, необходимо ввести

переменные: X Y

константы: F + −

старт: FX

правила: $(X \rightarrow X + YF+), (Y \rightarrow -FX - Y)$

угол: 90°,

где F означает рисование отрезка, «-» обозначает поворот на 90 влево, а «+» - поворот вправо. X и Y не соответствуют какому-либо действию при рисовании, а применяются исключительно для построения кривой [5]. Результат представлен на рисунке 3.3.

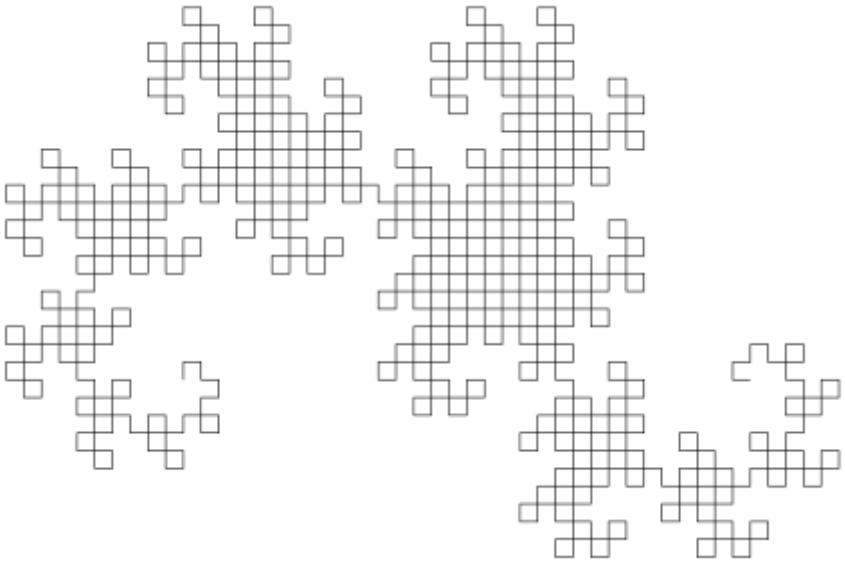


Рисунок 3.3 - Кривая дракона для n = 10

3.2.1.2 Салфетка и ковер Серпинского

В. Серписким были разработаны 2 вида фракталов на основе общей их математической постановки: **салфетка** и **ковер Серпинского**.

Салфетка Серпинского является регулярным фракталом и строится путем последовательного вырезания центральных равносторонних треугольников таким образом, как отражено на рисунке 3.4.

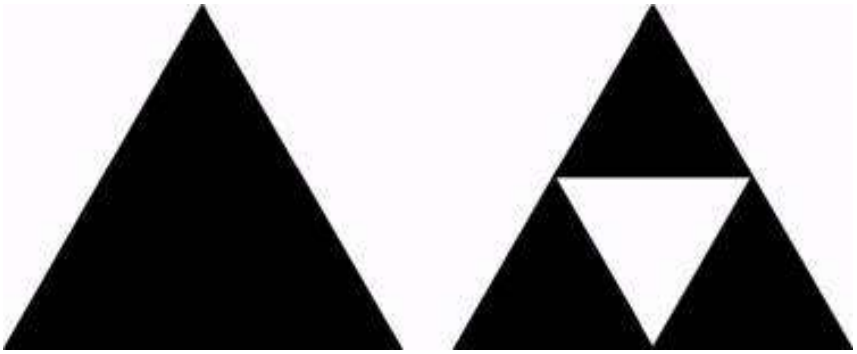


Рисунок 3.4 - Построение салфетки Серпинского

В результате получается фигура с вырезами (рисунок 3.5), состоящая из бесконечного числа изолированных точек. Фрактальная размерность салфетки Серпинского подсчитывается по формуле (3.1).

$$D = \frac{\ln\left(\frac{N(l)}{N(l')}\right)}{\ln\left(\frac{1}{l'}\right)}, \tag{3.1}$$

$$D = \frac{\ln 3}{\ln 2} = 1{,}5849. \tag{3.2}$$

На старте алгоритма имеется один равносторонний треугольник с длиной стороны равной единице, а на следующем - 3 равносторонних треугольника со сторонами $a` = \frac{1}{2}$. Поэтому, $a = 1$, $N(a) = 1$, $a` = 1/2$, $N(a`) = 3$.

Салфетка имеет нулевую площадь, в силу того, что в процессе ее формирования исключается площадь, абсолютно равная площади исходного треугольника. Это же описывает и значение фрактальной размерности D < 2, которая меньше размерности плоскости, в рамках которой расположен графический объект.

Вычислим периметр исключенных областей. Если сторона исходного треугольника равнялась единице, то на 1- ой итерации построения периметр

центрального треугольника равен $\frac{3}{2}$. На 2-ом шаге к нему добавляются 3 новых треугольника с общим периметром, равным $\frac{9}{4}$ и так далее. Очевидно, что на n-м шаге периметр P определяется суммой геометрической прогрессии [6].

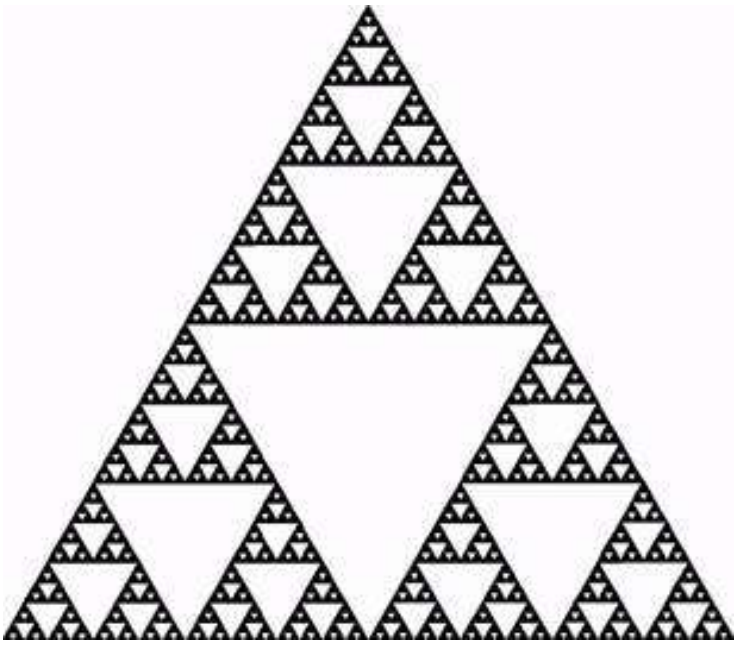


Рисунок 3.5 - Салфетка Серпинского

$$P = \sum_{k=1}^{n} \left(\frac{3}{2}\right)^{k} = 3\left[\left(\frac{3}{2}\right)^{n} - 1\right] \approx 3\left(\frac{3}{2}\right)^{n}. \quad (3.3)$$

С другой стороны, масштаб длины на n - м шаге равен $l = 1/2^{n}$. Поэтому формула для периметра, выраженного через этот масштаб, приобретает вид, схожий с формулой (3.4) для длины береговой линии.

$$L = l\left(\frac{R}{l}\right)^{D}, \quad (3.4)$$

$$P \approx l\left(\frac{2}{l}\right)^{D}. \quad (3.5)$$

где D определяется формулой (3.2).

Итак, построим непрерывную линию с такой величиной фрактальной размерности и геометрически эквивалентную салфетке Серпинского. Инициирующим элементом данного вида построения является отрезок длины равной одному. В последующем отрезок заменяется на конструкцию, которая называется генератором, состоящую из 3-х отрезков длиной по $\frac{1}{2}$, расположенных под углом 120 друг к другу (ситуации представлена на рисунке 3.6). После каждый из этих 3-х отрезков аналогично заменяется на генератор в 2 раза меньшего размера так, как показано на рисунке 3.7 слева. Правая часть того же рисунка иллюстрирует новую итерацию этой процедуры [6].

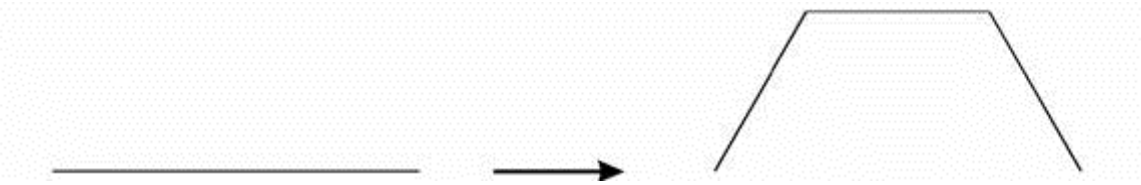


Рисунок 3.6 - Инициирующий элемент и генератор для кривой Серпинского

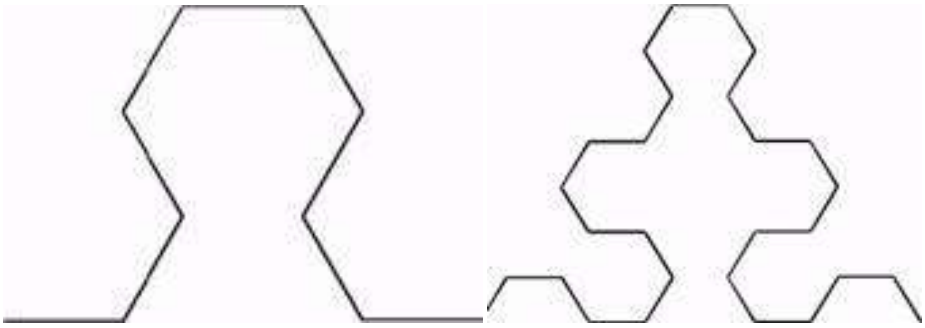


Рисунок 3.7 - Второй и третий шаги в построении кривой Серпинского

Контуры будущей салфетки Серпинского отчетливо проступают на следующих двух итерациях (рисунок 3.8). Данная процедура повторяется

бесконечное число раз. Новый рисунок может быть получен из предыдущего посредством склеивания трёх уменьшенных в 2 раза его копий, 2 из которых повернуты на угол в 120^o и -120^o относительно исходника.

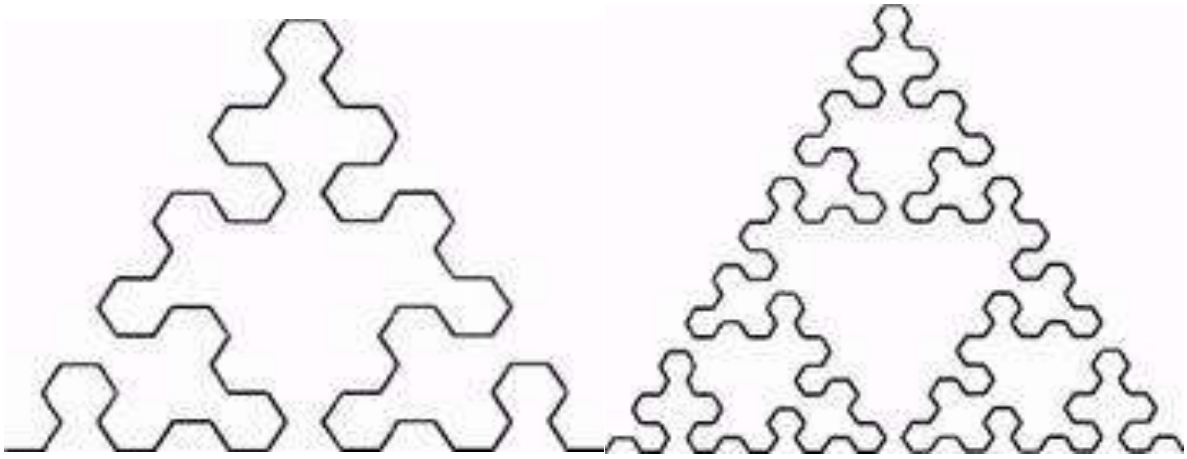


Рисунок 3.8 - Следующие два шага в построении кривой Серпинского

Аналогично салфетке Серпинского можно также построить квадратный ковер Серпинского. Ковер является 2-мерным аналогом канторовского множества исключенных средних третей.

Для его построения первоначально берется квадрат с длиной стороны, которая равна одному. После этого каждая из сторон квадрата делится на 3 равные части, а весь квадрат - на 9 одинаковых квадратов меньшего размера со стороной, равной $\frac{1}{3}$. Из полученной в результате преобразований фигуры вырезается центральный квадрат. Аналогичной процедуре после подвергается каждый из 8 оставшихся квадратиков и так далее [7].

Все это представлено на рисунках 3.9 и 3.10.

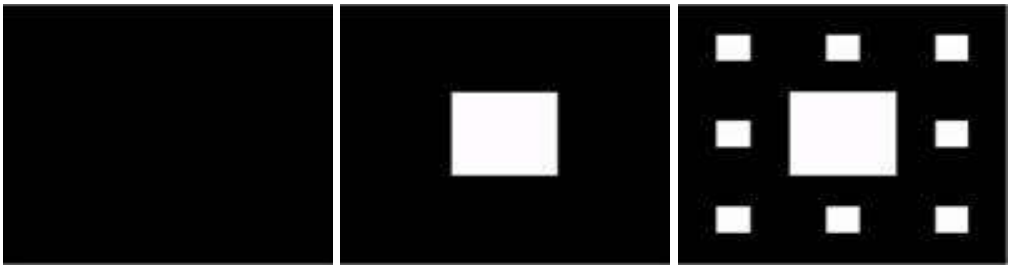


Рисунок 3.9 - Построение квадратного ковра Серпинского

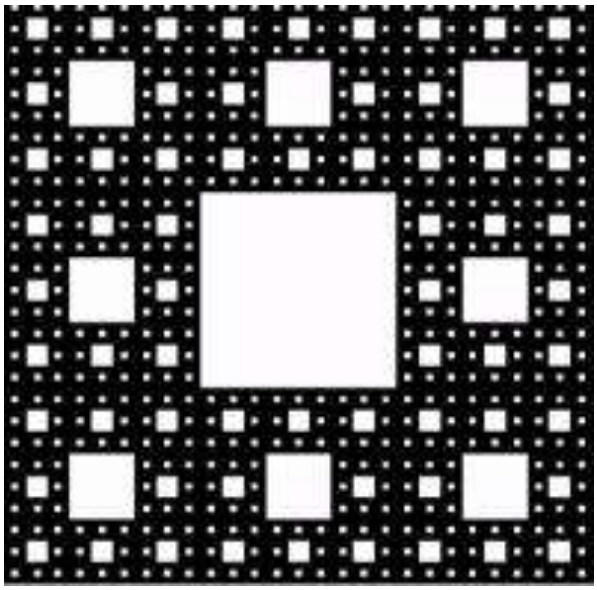


Рисунок 3.10 - Квадратный ковер Серпинского

В результате получается дырявый квадратный ковер Серпинского со значением фрактальной размерности:

$$D = \frac{\ln 8}{\ln 3} = 1{,}8928. \qquad (3.6)$$

Он также представляет собой пример идеального самоподобного фрактала. Его фрактальная размерность, однако, больше, чем у салфетки Серпинского, то есть он является в каком-то смысле менее дырявым.

3.2.2 Алгебраические фракталы

Самая крупная группа фракталов – это **алгебраические фракталы**. Данный класс фракталов получают с помощью нелинейных процессов в n-мерных пространствах. Наиболее изучены и распространены процессы двухмерного случая. Если интерпретировать нелинейный итерационный процесс как дискретную динамическую систему, то возможно использование терминологического аппарата теории этих систем: фазовый портрет, установившийся процесс, аттрактор и так далее.

Нелинейные динамические системы характеризуются конечным числом устойчивых состояний. Текущее состояние системы после определенного количества шагов алгоритма напрямую связано от ее первоначального состояния. Именно поэтому каждое устойчивое состояние, иначе называемое **аттрактором**, включает некоторую область начальных состояний, из которых система обязательно попадет в рассматриваемые конечные состояния. Следовательно, **фазовое пространство** системы делится на области притяжения аттракторов. Если фазовым является 2-мерное пространство, то в конечном итоге окрашивания области различными цветами, можно получить цветовой **фазовый портрет** данной системы (итерационного процесса). Изменяя алгоритм выбора цвета, можно получить сложные фрактальные картины с замысловатым разноцветным узором.

В качестве примера необходимо рассмотреть **множество Мандельброта**, представленное на рисунке 3.11. Алгоритм его построения достаточно прост и основан на простом итеративном выражении:

$$A[i+1] = A[i] * A[i] + C, \qquad (3.7)$$

где Ai и C - комплексные переменные.

Каждый новый шаг выполняется для всех начальных точек C прямоугольной или квадратной области - подмножестве комплексной

плоскости. Весь цикл продолжается пока $A[i]$ не покинет пределы окружности радиуса два, центр которой лежит в точке начала координат сойдется к какой-нибудь точке окружности. В зависимости от числа повторений, в течение которых оставалась внутри окружности, можно установить цвет точки С (если $A[i]$ остается внутри окружности в течение большого количества шагов, цикл завершается и эта точка растра окрашивается в черный цвет) [8].

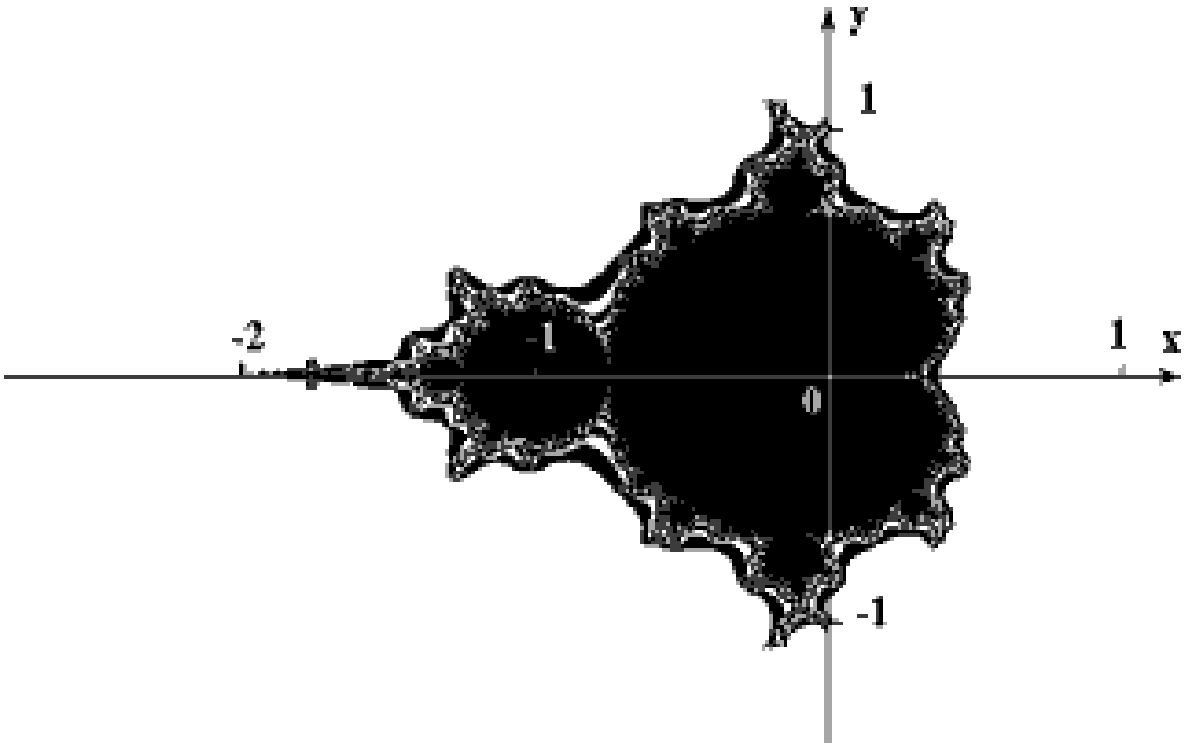


Рисунок 3.11 - Множество Мандельброта

Выше приведённый алгоритм даёт приближение к **множеству Мандельброта**, которому принадлежат точки, не уходящие в бесконечность на протяжении большого количества повторений (точки имеющие черный цвет). Те точки, которые лежат на границе множества (именно там возникают сложные структуры) уходят в бесконечность за конечное число шагов, точки же лежащие за пределами множества, уходят в бесконечность через несколько итераций, образуя белый фон.

3.2.3 Стохастические фракталы

Стохастические фракталы – это еще один из классов фракталов, которые получаются в случае случайного изменения различных параметров в итерационном процессе. В этом случае получаются объекты в значительной степени похожие на природные - асимметричные растения, изрезанные береговые линии, горы и тому подобное. Двумерные стохастические фракталы применяются для моделирования рельефа местности и поверхности моря.

3.3 Системы итерируемых функций

Метод систем итерируемых функций (Iterated Functions System - IFS) появился в середине 1980-х гг. в качестве простого средства получения фрактальных структур.

Метод представляет собой систему функций из некоторого фиксированного класса функций, отображающих одно многомерное множество на другое. Наиболее простая IFS состоит из аффинных преобразований плоскости:

$$X' = A * X + B * Y + C, \quad (3.8)$$

$$Y' = D * X + E * Y + F. \quad (3.9)$$

В 1988 году американские специалисты в теории динамических систем и эргодической теории Барнсли и Слоан предложили некоторые идеи, основанные на соображениях теории динамических систем, для сжатия и хранения различного рода графической информации. Метод получил название - **метод фрактального сжатия информации**. Это объясняется тем, что геометрические образы, возникающие в этом методе, обычно имеют фрактальную природу на подобии теории Мандельброта.

Основываясь на выше сказанном алгоритм Барнсли и Слоан, по их утверждению, должен был позволить сжимать информацию в 500-1000 раз.

Суть алгоритма состоит из следующем: изображение кодируется несколькими простыми преобразованиями (в данном случае аффинными), то есть коэффициентами этих преобразований (в нашем случае A, B, C, D, E, F).

В случае кодировки какого-либо изображения двумя аффинными преобразованиями, можно вполне однозначно определять его с помощью двенадцати коэффициентов. Если теперь задаться какой-либо начальной точкой (например, X=0 Y=0) и запустить алгоритм, то после 1-ой итерации получим две точки, после 2-ой - четыре, после 3-ей - восемь и так далее. После некоторого количества итераций совокупность полученных точек будет описывать закодированное изображение. Задача лишь заключается в том, что весьма сложно найти коэффициенты системы, которая кодировала бы произвольное изображение [9].

Для построения системы итерируемых функций используют кроме аффинных и другие классы простых геометрических преобразований, которые позволяют задавать IFS небольшим числом параметров. Например, проективные (3.10) и (3.11):

$$X' = (A1 * X + B1 * Y + C1) / (D1 * X + E1 * Y + F1), \quad (3.10)$$

$$Y' = (A2 * X + B2 * Y + C2) / (D2 * X + E2 * Y + F2), \quad (3.11)$$

или квадратичные (3.12), (3.13)

$$X' = A1 * X * X + B1 * X * Y + C1 * Y * Y + D1 * X + E1 * Y + F1, \quad (3.12)$$

$$Y' = A2 * X * X + B2 * X * Y + C2 * Y * Y + D2 * X + E2 * Y + F2. \quad (3.13)$$

преобразования на плоскости.

В качестве примера использования IFS для построения фрактальных структур, рассмотрим кривую Коха и «дракон» Хартера-Хейтуэя (рисунок 3.12). Для этого необходимо выделить в этих структурах идентичные части и, для каждой из них вычислить коэффициенты аффинного преобразования. В аффинный коллаж будет включено столько аффинных преобразований, сколько существует частей подобных целому изображению.

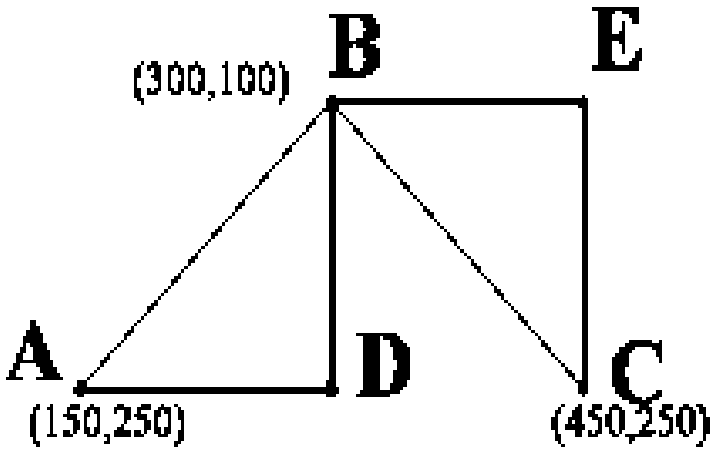


Рисунок 3.12 - Заготовка для построения IFS «дракона» Хартера-Хейтуэя

Для построения IFS «дракона» Хартера-Хейтуэя требуется расположить 1-ое поколение фрактала на сетке координат дисплея 640 x 350. Через A, B и C обозначим точки получившейся ломаной. По правилам построения у данного вида фрактала две части, подобные целому - это ломаные ADB и BEC. Зная координаты концов этих отрезков, можно вычислить коэффициенты двух аффинных преобразований, переводящих ломаную ABC в ADB и BEC:

$$X' = -0.5 * X - 0.5 * Y + 490$$
$$Y' = 0.5 * X - 0.5 * Y + 120$$

$$X' = 0.5 * X - 0.5 * Y + 340$$
$$Y' = 0.5 * X + 0.5 * Y - 110$$

В качестве начальной стартовой точки зададим, например, X=0 Y=0 и итерационно действуя на точку этой системы, после десятой итерации на экране получается фрактальная структура, изображенная на рисунке 3.13. Представленный вид структуры и называется «дракон» Хартера-Хейтуэя. Его кодом (сжатым описанием) является набор коэффициентов двух аффинных преобразований [10].

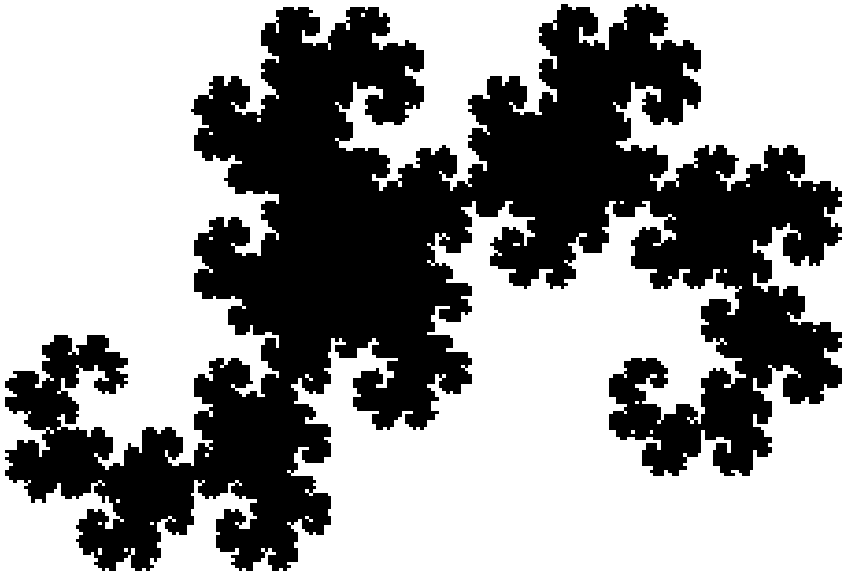


Рисунок 3.13 – «Дракон» Хартера-Хейтуэя, построенный с помощью IFS в прямоугольнике 640x350

Аналогично можно построить IFS для кривой Коха (рисунок 3.14). Эта кривая имеет четыре части, подобные целой кривой. Для нахождения системы итерируемых функций снова представим 1-ое поколение фрактала на сетке координат дисплея 640x350.

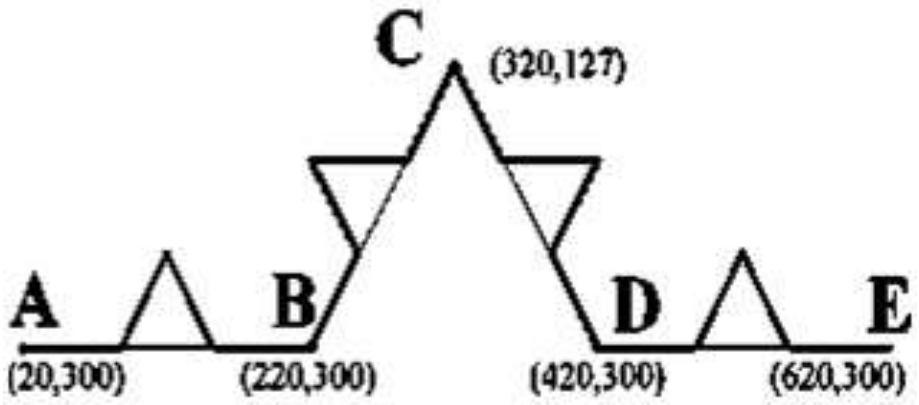


Рисунок 3.14- Заготовка для построения IFS кривой Коха

Для ее построения требуется набор аффинных преобразований, состоящий из четырех преобразований:

$$X' = 0.333 * X + 13.333$$
$$Y' = 0.333 * Y + 200$$

$$X' = 0.333 * X + 413.333$$
$$Y' = 0.333 * Y + 200$$

$$X' = 0.167 * X + 0.289 * Y + 130$$
$$Y' = -0.289 * X + 0.167 * Y + 256$$

$$X' = 0.167 * X - 0.289 * Y + 403$$
$$Y' = 0.289 * X + 0.167 * Y + 71$$

Результат применения данного аффинного коллажа после десятой итерации представлено на рисунке 3.15.

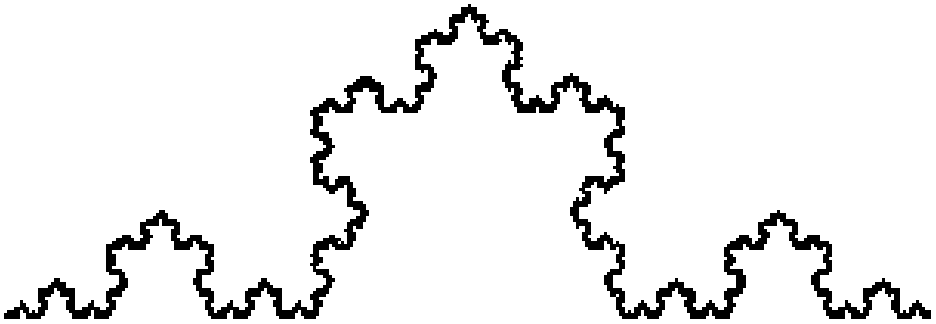


Рисунок 3.15 - Кривая Кох, построенная с помощью IFS в прямоугольнике 640x350

Применение IFS для сжатия обычных файлов, таких как фото, основывается на выявлении локального самоподобия, по сравнению с фракталами, где наблюдается глобальное самоподобие и нахождение системы итерируемых функций относительно просто. По алгоритму Барнсли выделяются в рисунке пары областей, наименьшая из которых подобна большей, и сохранение нескольких коэффициентов, кодирующих преобразование, переводящее большую область в меньшую. Необходимое условие это, когда множество меньших областей покрывает все изображение. К тому же в файл, кодирующий изображения будут записаны не только коэффициенты, характеризующие найденные преобразования, но и местоположение и линейные размеры «больших» областей, которые вместе с коэффициентами будут описывать локальное самоподобие кодируемой картинки. Восстанавливающий алгоритм, в данном случае, должен использовать каждое преобразование не ко всему множеству точек, получившихся на ранних итерациях алгоритма, а к некоторому их подмножеству, принадлежащему области, соответствующей применяемому преобразованию [10].

Одним из способов построения стохастических фракталов является **рандомизированный метод** на основе систем итерируемых функций.

Построение фракталов на основе IFS происходит следующим образом:

Пусть для некоторого фрактала требуются N аффинных преобразований.

На начальном шаге рандомизированного алгоритма выбирается одна точка.

Общий шаг алгоритма:

1) Случайным образом выбирается одно из N аффинных преобразований;

2) Образуется новая точка путем применения к предыдущей («старой») выбранного аффинного преобразования;

3) Новая точка изображается;

4) Выполняется присваивание: новая точка будет рассматриваться в качестве «старой» для следующего шага алгоритма.

Общий шаг можно повторять заданное число раз либо до тех пор, пока детали изображения не станут мельче заданной величины.

Таким образом, в отличие от детерминированного алгоритма на основе IFS на каждом шаге рандомизированного алгоритма обрабатывается только одна точка и применяется только одно аффинное преобразование [6].

Случайный выбор преобразования выполняется следующим образом:

1) Пусть каждому из N аффинных преобразований приписана некоторая вероятность p_i (число от 0 до 1), $i = 1, \ldots, N$, причём сумма вероятностей всех преобразований равна единице: $\text{api}_1^N = 1, i = 1$.

2) Разобьем интервал [0;1) на числовой оси точками с координатами $p1,\ p1 + p2, p1 + p2 + p3, \ldots, p1 + \ + pN + 1$ на N частичных интервалов $1,2, \ldots,\ N$: $1 - [0; p1)$, $2 - (p1;\ p1 + p2), \ldots,\ N - (p1 + \ + pN - 1; 1)$.

3) С помощью генератора случайных чисел выбирается R – случайное вещественное число от 0 до единицы.

4) Если R попало в частичный интервал i, то выбирается i-ое преобразование [11].

На рисунке 3.16 – 3.17 показаны результаты выполнения программы рисования фрактала «Лист папоротника».

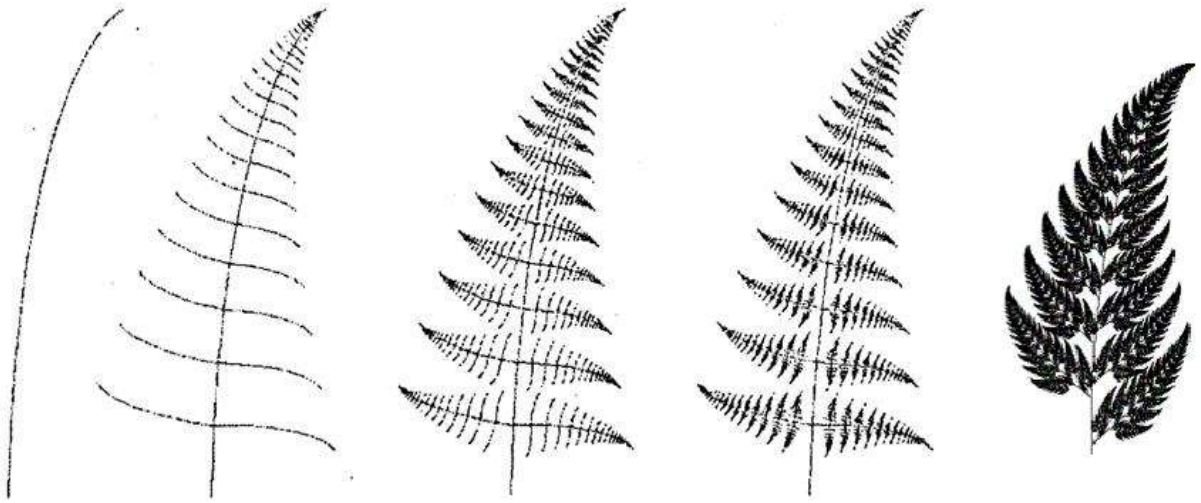


Рисунок 3.16 – Вид фрактала для разного количества циклов итераций

Применение системы итерируемых функций является простым средством получения фрактальных структур. Выше описанные алгоритмы имеют широкий спектр применения фракталов на практике, о чем речь пойдет в следующем пункте учебного пособия.

3.4 Практическое приложение теории фракталов

Одним из **первых примеров применения теории фракталов** является **рисование гор.** Принцип, разработанный Лореном Карпентером, состоял в разделении более крупной геометрической фигуры на мелкие элементы, а те, в свою очередь, делить на аналогичные фигуры меньшего размера. Визуализация представлена на рисунке 3.17.

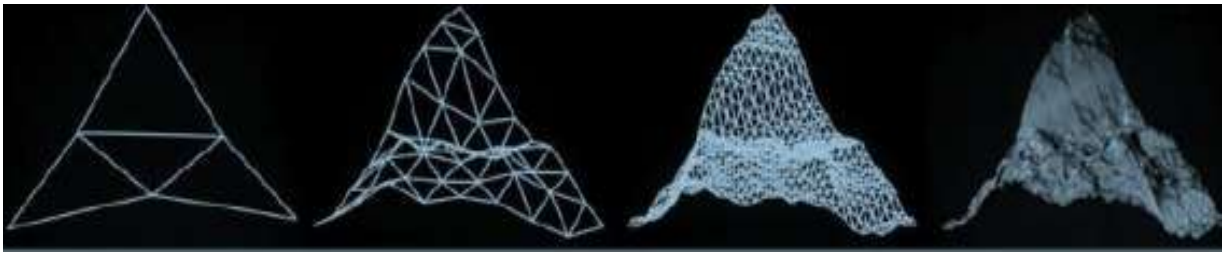


Рисунок 3.17 – Применение фрактальной теории для рисования гор

В случае использования более крупных элементов - треугольников, Карпентер предложил дробить их на четыре части, еще мельче размером. При чем, процедуру необходимо было повторять до тех пор, пока в результате не получается максимально реалистичный горный ландшафт. Таким образом, Карпентера принято считать первым художником, применившим фрактальный алгоритм компьютерной графики для построения изображений.

На сегодняшний момент все самые распространенные и популярные программные приложения, направленные на создание трёхмерных ландшафтов основываются на подобный принципе генерации объектов природы. Такие пакеты трёхмерной графики как Terragen, Bryce, Vue и так далее основываются на фрактальном алгоритме моделирования поверхностей и текстур.

Еще одним из практических приложений теории является создание Натаном Коэном антенны, обладающей как можно более высокой чувствительностью.

Единственный из известных на тот период времени способ улучшения параметров антенны - это увеличении ее геометрических размеров. Коэнс делал из проволоки один из известнейших фракталов - «снежинку Коха» (рисунок 3.18).

Существуют также другие разновидности кривой Коха, но примерная форма кривой остается аналогичной.

При подключении антенны Натана к устройству приема радиосигналов чувствительность резко увеличивается. В данном случае антенна обладает высоким КПД, а также покрывает гораздо более широкий частотный диапазон относительно классического решения проблемы. Более того, форма антенны в виде кривой фрактала, которая существенно сокращает ее геометрические размеры. Ученым была выведена теорема, позволяющая доказать достаточность самоподобной фрактальной формы кривой для создания широкополосной антенны.

Автор запатентовал свое открытие и основал фирму по разработке и проектированию фрактальных антенн FractalAntennaSystems. В настоящее время благодаря открытию Натана сотовые телефоны смогли избавиться от громоздких антенн, что позволило им стать значительно меньше.

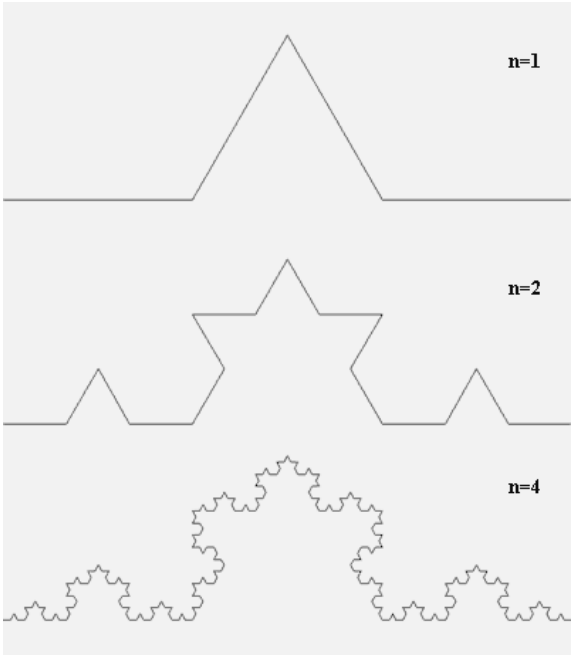


Рисунок 3.18 – «Снежинка Коха»

На сегодняшний момент ученые находят все больше областей практического применения теории фракталов, в частности экономическая сфера.

Современная экономическая теория давно доказала несостоятельность и неадекватность традиционных линейных моделей поведения рынков. Практика показывает, что динамика экономических процессов и явлений носит нелинейный и, зачастую, хаотичный (непредсказуемый) характер. Это обуславливает акутуальность применения фрактальных сруктур в экономике.

На данный момент фрактальный анализ рынка наиболее распространён на рынке Форекс.

Важнейшим классом природных фракталов являются **хаотические временные ряды,** или упорядоченные во времени наблюдения характеристик различных экономических процессов, в первую очередь финансовые временные ряды, а именно цены акций и курсы валют. Как известно, базовой моделью финансовых временных рядов является модель случайного блуждания, впервые полученная Луисом Башелье для описания наблюдений за ценами акций на Парижской фондовой бирже. В результате переосмысления этой модели, которая иногда наблюдается в поведении цен, возникла концепция эффективного рынка, на котором цена в полной мере отражает всю доступную информацию. Для существования такого рынка достаточно предположить, что на нем действует большое число полностью информированных рациональных агентов, которые мгновенно реагируют на поступающую информацию и корректируют цены, приводя их в состояние равновесия. Все основные результаты классической теории финансов (портфельная теория, модель CAPM, модель Блэка-Шоулза и другое) были получены в рамках именно такого подхода. В настоящее время концепция эффективного рынка продолжает играть доминирующую роль и в финансовой теории, и в финансовом бизнесе.

К началу 60-х годов прошлого века эмпирические исследования показали, что сильные изменения цен на рынке происходят значительно чаще, чем предсказывала основная модель эффективного рынка (модель случайного блуждания). Одним из первых, кто подверг концепцию эффективного рынка всесторонней критике, был **Мандельброт.**

Действительно, если корректно вычислить значение показателя H для какой-либо акции, то оно, вероятнее всего, будет отлично от H = 0,5, которое соответствует модели случайного блуждания. Мандельброт нашел все возможные обобщения этой модели, которые могут иметь отношение к реальному поведению цен. Как оказалось, это, с одной стороны, процессы, названные им полетом Леви, а с другой — процессы, которые он назвал обобщенным броуновским движением. Поведение временного ряда, для которого (достаточно часто наблюдается на реальном рынке), можно обозначить с помощью любого из этих процессов.

Для описания поведения цен обычно используют **концепцию фрактального рынка**, которую принято рассматривать в качестве альтернативы выше описанной. Концепция предполагает, что на рынке есть широкий спектр агентов с разными инвестиционными горизонтами и, следовательно, разными предпочтениями. Эти горизонты меняются от нескольких минут для внутридневных трейдеров до нескольких лет для крупных банков и инвестиционных фондов. Устойчивым положением на таком рынке является режим, при котором средняя доходность не зависит от масштаба, если не считать умножения на соответствующий масштабный коэффициент. Фактически речь идет о целом классе режимов, каждый из которых определяется своим значением показателя H. При этом значение H = 0,5 оказывается одним из многих возможных и, следовательно, равноправным с любым другим значением.

Эффективность цены

Исследование фрактальных свойств цен российских (в индексе ММВБ) и американских (входящих в Dow Jones Internet Index) компаний вместе с соответствующими индексами за последние десять лет подчеркивает особое положение значения H = 0,5. Для этого, однако, необходимо использовать новый фрактальный показатель (**индекс фрактальности**). Он связан с показателем H соотношением, однако для его определения с приемлемой точностью требуется на два порядка меньше данных, чем для

показателя H, поэтому его можно рассматривать в качестве локальной фрактальной характеристики. Оказывается, что с помощью индекса фрактальности можно дать обоснование современной теории финансов, а также предсказать сильные колебания на фондовом рынке.

В первом приближении общая картина, наблюдаемая во всех рядах, оказывается следующей. Индекс фрактальности (и фрактальная размерность финансовых рядов) совершает квазипериодические колебания около положения = 0,5 (этот режим соответствует случайному блужданию). При этом временной ряд непрерывно изменяет свой режим, переходя из тренда (< 0,5) через состояние случайного блуждания во флэт (> 0,5) и обратно. Время от времени для каждого ряда появляются и исчезают состояния с относительно стабильными значениями, отличными от 0,5. При этом режим с = 0,5 занимает явно привилегированное положение. Для каждого временного ряда он является самым длительным на всех интервалах, содержащих восемь точек и более.

Стоит отметить, что интерпретация колебаний цен, базирующаяся на описании поведения рыночных агентов, может сильно отличаться в разных масштабах. К примеру, в течение дня, где более половины сделок совершается торговыми роботами (на рынках США), поведение агентов, скорее всего, является очень близким к рациональному. На масштабах же от нескольких дней до нескольких месяцев существенную роль играет социальная психология, которая всегда содержит иррациональный элемент. Между тем неизменный характер колебаний с наиболее часто встречающимся режимом случайного блуждания воспроизводится на всех масштабах, начиная от самых малых. Это наводит на мысль о том, что в основе природы этих колебаний лежит, видимо, общий механизм запаздывания, сопутствующий самому способу принятия решений агентами на фондовом рынке. При этом основным состоянием цен является все же именно случайное блуждание, которое остается главным режимом притяжения на всех масштабах. Другими словами, несмотря на часто

возникающие длительные локальные отклонения, цены стремятся вернуться к эффективному поведению, которое описывает модель случайного блуждания [12].

Метод прогнозирования

Наличие описанных фрактальных свойств ценового ряда, наблюдаемых в широком диапазоне масштабов, позволяет по-новому взглянуть на возможность прогнозирования фондового рынка. В общем, задача прогноза - определить качественные или количественные параметры будущего поведения временного ряда на основе всего массива исторических данных. При этом особый интерес представляет определение ранних предвестников критического поведения ряда.

Рассмотрим один из новых подходов к решению этой задачи, основанный на фрактальных свойствах цен. Было доказано, что если ввести среднюю амплитуду колебаний как среднюю разность между максимальным и минимальным значениями цены, усредненными по сегментам размера t, то средняя амплитуда колебаний будет связана с масштабом наблюдения степенной зависимостью:

$$A(t) \sim t^{H_\mu}, \tag{3.14}$$

где

$$H_\mu \equiv 1 - \mu. \tag{3.15}$$

Индекс (который, как и индекс, для своего определения требует на два порядка меньше данных, чем показатель H) совпадает с H на тех участках, где H можно вычислить с приемлемой точностью. Зависимость средней амплитуды колебаний от масштаба наблюдений для разных значений H представлена на рисунке 3.19.

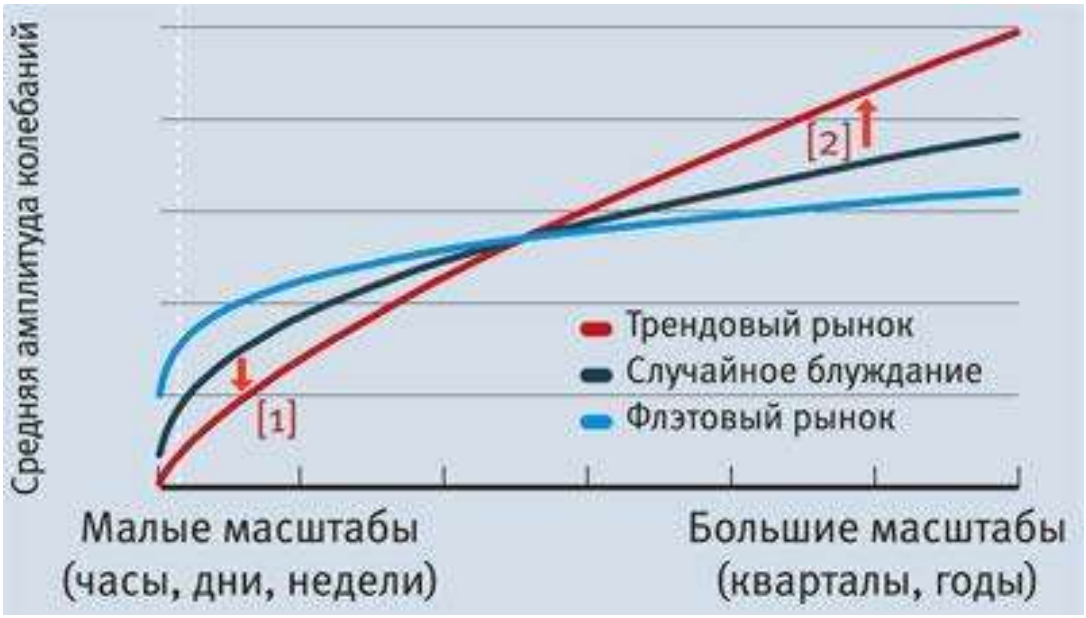


Рисунок 3.19 - Зависимость средней амплитуды колебаний цен от масштаба наблюдения

Оказывается, что знание закона зависимости амплитуды колебаний от времени в разных режимах позволяет обосновать весьма любопытный эффект, который может стать ключом к прогнозу возникновения на рынке сильного движения. Действительно, предположим, что в данный момент рынок находится в переходном режиме от случайного блуждания к сильному тренду. Значит, через определенное время амплитуда колебаний на больших масштабах (например, несколько месяцев) станет существенно больше, чем текущая амплитуда (стрелка 2 на рисунке 3.19 показывает переход от случайного блуждания к тренду на больших временных масштабах). Это одновременно означает (в силу свойства степенной функции), что на малых временных масштабах (часы, дни недели) должно наблюдаться уменьшение амплитуды колебаний по сравнению с предыдущим периодом (стрелка 1 рисунка показывает такой переход на малых масштабах). Таким образом, наблюдая за поведением амплитуды на малых масштабах, в некоторых случаях можно прогнозировать существенное увеличение амплитуды колебаний цен в будущем.

Состояния рынка с увеличенной амплитудой колебаний обычно наблюдаются в корнерах (резких взлетах цен на рынке) или крахах (резких обвалах). Эффект увеличения крупномасштабных колебаний при уменьшении мелкомасштабных был теоретически обоснован авторами. Как показало тестирование по всей указанной выше базе финансовых данных, этот эффект проявляется с вероятностью 70 – 80 %. В тех же случаях, когда удается свести до минимума влияние внешних факторов, этот процент оказывается еще выше.

4 Растровая графика

4.1 Общие сведения. Понятие растра и его математическая постановка

Растровая, или **точечная графика** - это изображения, составленные из пикселов размещенных в прямоугольной сетке.

Суть данного вида графики заключается в мозаичном сборе картинки, при этом, чем больше составляющих фрагментов мозаики, тем подробнее получается изображение, особенно если смотреть на него с расстояния.

Иначе говоря, **растр** - это описание рисунка на плоскости посредством деления всей плоскости или ее отдельной части на одинаковые квадраты и последующее присвоение каждому квадрату своего цветового (или иного, например, прозрачности, для последующего наложения изображений друг на друга) параметра. Если выделенных квадратов конечное число, то в результате получается, что непрерывная цветовая функция изображения приближенно представлена конечной совокупностью значений атрибутов. В некоторых случаях термин «растр» определяется несколько шире - разбиение плоскости (или ее участка) на равные элементы (то есть «замощение»). Если замощение происходит, например шестиугольниками, то поученный результат называется гексагональный растр.

Составляющие растрового изображения - это пикселы. **Пикселом** следует считать - элементарную единицу картинки в растровой графике, как правило, имеющая квадратную форму. Размеры пиксела зависят от разрешения изображения (количества пикселов на единицу длины) [13].

На рисунке 4.1 представлено растровое изображение и его увеличенный фрагмент, при этом достаточно отчетливо видны элементы квадратной формы, из которых складывается изображение.

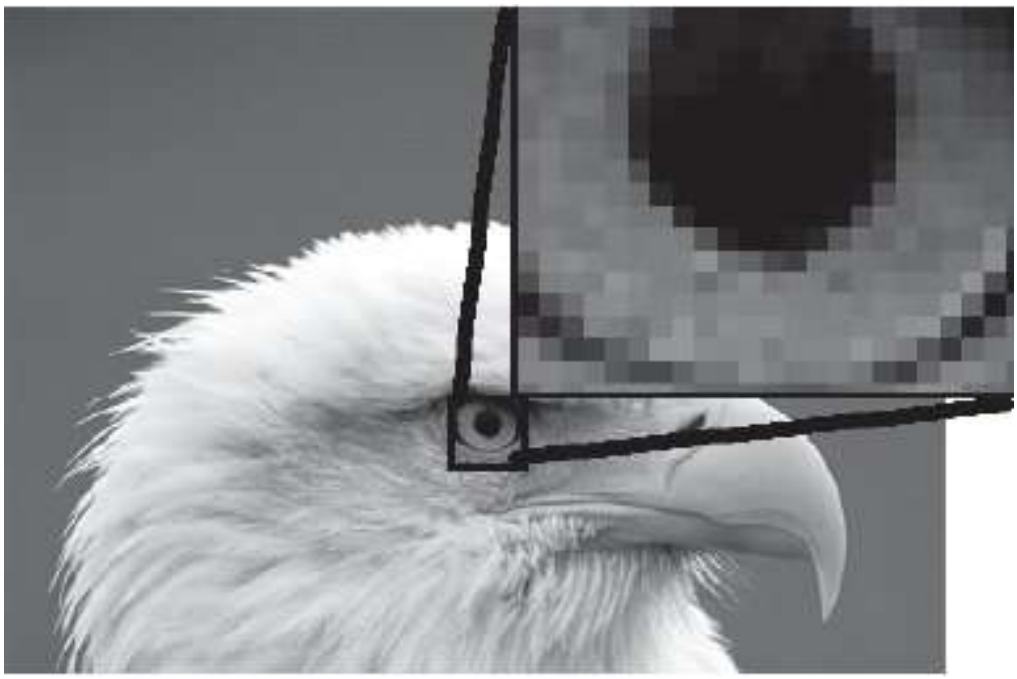


Рисунок 4.1 - Растровое изображение и его увеличенный фрагмент

С учетом всех особенностей построения, растровая графика имеет следующие отличительные черты:

- растровое изображение всегда должно быть прямоугольной формы;

- применение инструмента масштабирования позволит существенно понизить качество растрового рисунка;

- растровый документ не может содержать объекты в разных цветовых режимах.

Разрешение - это число точек на единицу длины изображения (дюйм, сантиметр). Разрешение является одним из основных параметров изображения и чем он выше, тем выше качество рисунка, но при этом увеличивается размер его файла. Чаще всего применяется 72 пиксела на дюйм (так называемое экранное разрешение), но для получения качественного результата в полиграфии необходимо значительно больше.

Повышение качества путем увеличения разрешения провоцирует пропорциональное увеличение размеров файла.

Интерполяция - это математический способ повышения числа пикселов в растровом изображении, который применяется в графических пакетах. Но не стоит путать масштабирование и изменение размеров.

Масштабирование - это изменение изображения на экране. Данная модификация, как правило, связана с инструментом Zoom (Масштаб).

Изменение размеров - это внесение изменений непосредственно в графический файл (рисунок 4.2).

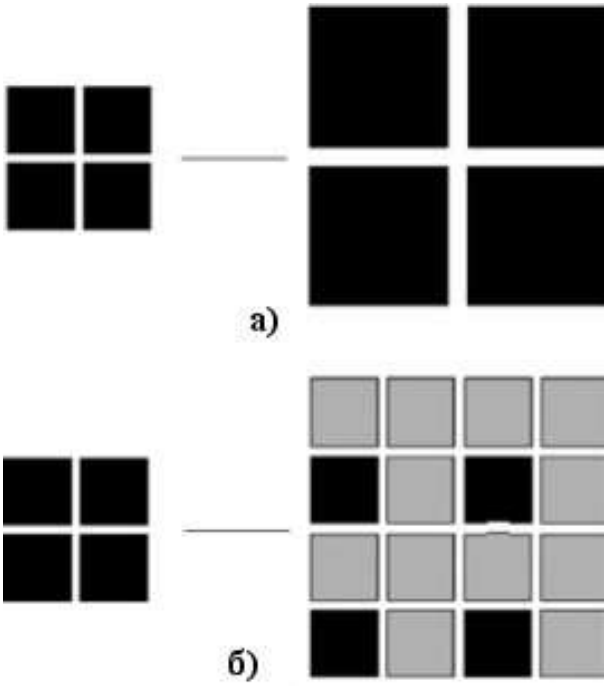


Рисунок 4.2 - Масштабирование изображения из четырех пикселов в два раза (а) и интерполяционное увеличение (б)

Под **связностью** необходимо понимать возможность соединения двух пикселей растровой линией, то есть последовательным набором пикселей. Но в данном случае необходимо определять какие пиксели можно считать соседними. Для этого вводятся два понятия связности:

1) **Четырёхсвязность**: пиксели считаются соседними, если либо их x-координаты, либо их y – координаты отличаются на единицу:

$$|x_1 - x_2| + |y_1 - y_2| \leq 1 \quad (4.1)$$

2) **Восьмисвязность**: пиксели считаются соседними, если их x-координаты и y-координаты отличаются не более чем на единицу:

$$|x_1 - x_2|, |y_1 - y_2| \leq 1| \quad (4.2)$$

На рисунке 4.3 изображены четырехсвязная и восьмисвязная линии.

Рисунок 4.3 - Четырехсвязная и восьмисвязная линии

Существуют алгоритмы перевода объектов в растровое изображение, которые используют как четырехсвязность, так и восьмисвязность.

Тем не менее, растр можно рассматривать как кусочно-постоянную аппроксимацию изображения, заданного как цветовая функция на плоскости. Такое понимание позволяет применять математический аппарат теории аппроксимации при работы с растровыми рисунками.

Формально, введем следующие определения:

Растр - это отображение вида

$$\mathrm{f: X \times Y \rightarrow 2^{R^2} \times C}, \quad (4.3)$$

где $X \subset Z, Y \subset Z$;

2^{R^2} - множество всех подмножеств R^2;

C - множество значений атрибутов (как правило, цвет).

f(i, j) - элемент растра, называемый пикселем.

$$f(i,j) = \big(A(i,j), C(i,j)\big), \tag{4.4}$$

где $A(i,j) \subset R^2$ - область пикселя;

$C(i,j) \in C$ - атрибут пикселя (как правило, цвет). Чаще всего будем пользоваться следующими двумя видами атрибутов:

$C(i,j) = I(i,j)$ - интенсивность (или яркость) пикселя;

$C(i,j) = \{R(i,j), G(i,j), B(i,j)\}$ - цветовые атрибуты в цветовой модели RGB (см. раздел 2).

Также иногда используются матричные обозначения:

$$Mij = (Aij, Cij) \tag{4.5}$$

Aij может определяться двояко, в зависимости от того, с какой цветовой моделью работают:

— $Aij := (i,j)$ - одна точка. Пример такой модели растра представлен на рисунке 4.4;

— $A_{ij} := (i, i+1) \times (j, j+1)$ квадрат. Пример такой модели растра показан на рисунке 4.5. На реальных графических устройствах физически пиксели могут быть прямоугольниками, что иногда порождает дополнительные трудности [14].

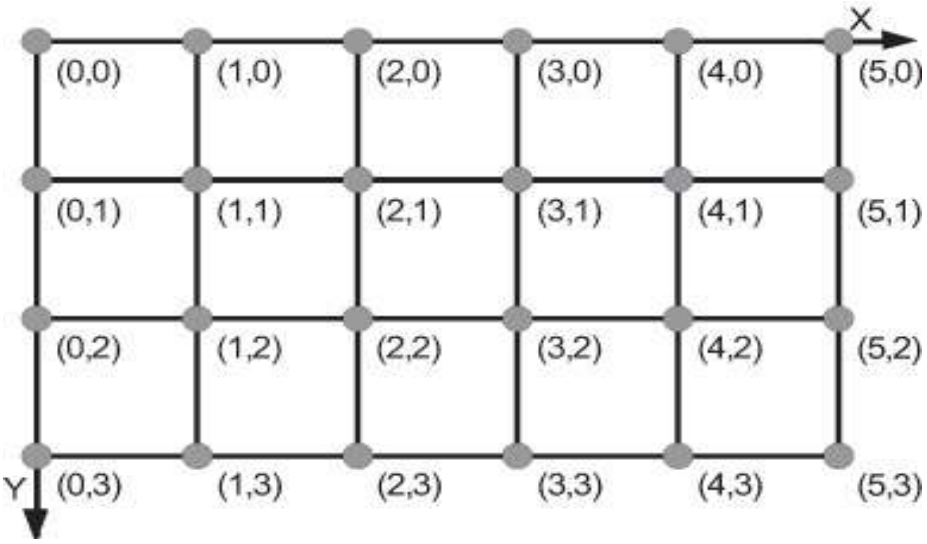


Рисунок 4.4 - Модель растра первого типа

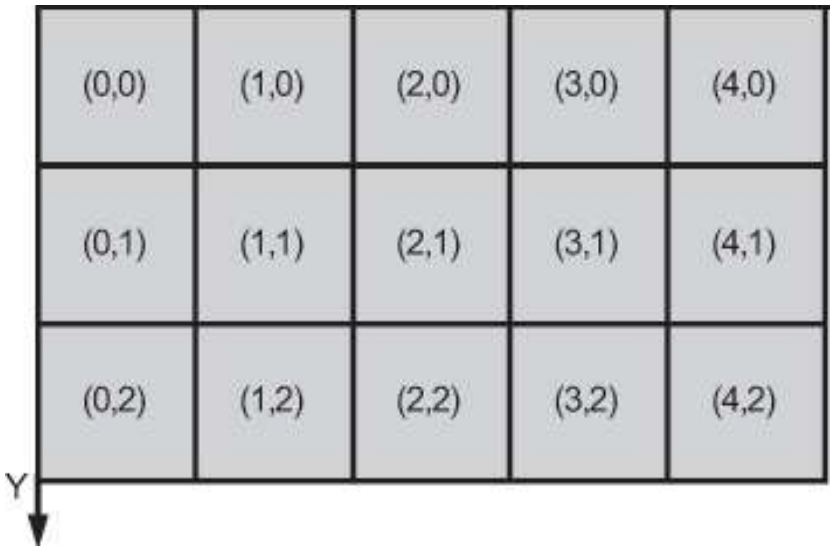


Рисунок 4.5 - Модель растра второго типа

В действительности преимущественно X и Y - это ограниченные наборы неотрицательных целых чисел. В данном случае растр называется прямоугольным. Для такого типа растра применим термин аспектовое отношение.

Аспектовым отношением называют отношение ширины к высоте растра $\frac{|X|}{|Y|}$. Как правило, данный термин применяется по отношению к

физическим растрам, таким как дисплеям, ПЗС-матрицами фотоаппаратов и так далее, и записывается в виде простой дроби с «:», например «4:3».

В случае, когда X и Y неограниченны, мы имеем дело с **бесконечными растрами**. Их применение бывает полезным для описания алгоритмов, что позволяет избежать особых ситуаций. Однако, самой сутью некоторых алгоритмов является непосредственно работа с граничными случаями.

Растровое представление изображений является обоснованным в том случае, если отсутствует дополнительная информация об изображаемых объектах (например, цифровым фотоаппаратом можно снимать изображения произвольного содержания). Если же объекты описываются на основе векторных, то имеем дело с более сложными объектами (линии и области, ограниченные линиями), что предполагает заранее обладать информацией о структуре изображения. В последнее время наиболее ярко выражена тенденция преобладания устройств ввода-вывода двумерной графической информации, основанных на растровом принципе как более универсальном. Возникающая при выводе задача отображения геометрических объектов, заданных их математическим описанием (например, координатами концевых точек и цветом для отрезка), на растре, называемая растеризацией.

При построении алгоритмов, которые работают с изображениями, также можно воспользоваться данными как непосредственно атрибутов пикселей, так и работать с примитивами более высокого порядка. В данном пособии в основном рассматриваются алгоритмы первого типа, про которые говорят, что они работают в пространстве изображения, тогда как вторые работают в объектном пространстве (эти термины чаще употребляются в трёхмерной графике).

4.2 Основные форматы растровых файлов

В основе растровой графики лежит одна и та же матрица пикселов, тем не менее форматов для записи растровых изображений существует большое

множество. Ниже приведены основные из них, которые наиболее часто используются в практической деятельности.

CPT (Corel Photo-Paint Image)

Стандартный формат, предназначенный для хранения растровых рисунков, созданных в программах Corel PHOTO-PAINT и CorelDRAW. Формат позволяет сохранять все, что возможно создать в этих программных пакетах при работе с точечной графикой. Тем не менее, для CPT характерна плохая поддержка другими редакторами, например, Photoshop.

PSD (Photoshop Document)

Собственный формат Photoshop. Также как и CPT - может сохранять любые объекты, созданные в «своей» программе Photoshop (слои, дополнительные каналы, комментарии, пути). На сегодняшний момент формат поддерживают все наиболее востребованные программные продукты редактирования растровой графики.

В PSD применяется стандарт компрессии RLE, что позволяет делать размеры файла несколько меньше, сохраняя качество изображения.

RLE (Run Length Encoding) - является одним из методов сжатия графических файлов. Принцип базируется на поиске совпадений в строках представлений. Например, если подряд встречается сорок черных пикселов, то они будут записаны не как 40 раз слово «черный», а как «40 черных». RLE эффективен только в случае большого количества повторяемости, то есть одноцветности областей изображения. В отличие от JPEG (описано ниже), при использовании RLE качество изображения не падает.

BMP (Windows Device Independent Bitmap)

Собственный формат операционной системы Windows. Поддерживает индексированный (256 цветов) и RGB-цвета. Этот формат воспринимаем абсолютно всеми (графические и не только) редакторами Windows.

Тем не менее, формат BMP имеет больше недостатков, чем достоинств:

- не подходит для работы в интернете;
- в случае распечатки изображений сильно теряется качество;

- является аппаратно зависимым форматом;
- некорректно использует RLE-сжатие;
- занимает большой объем места на диске.

Использование BMP будет оправдано лишь в случае применения в программах, которые не могут работать ни с какими другими форматами.

GIF (CompuServe Graphics Interchange Format)

Наиболее распространенный файловый формат для работы в Интернете. GIF был разработан в 1987 году для быстрой передачи изображений в сети. Изначально это был формат, поддерживающий только 256 цветов. Позднее была обнаружена возможность формата - записывать в один файл несколько изображений и проигрывать их с определенной задержкой. GIF поддерживает прозрачность, причем прозрачными можно назначить несколько цветов.

В GIF используется LZW-компрессия, что в сочетании с индексированными цветами делает этот формат идеальным для хранения и передачи изображений с малым количеством цветов.

LZW-компрессия - один из видов сжатия изображений, похожий на RLE. При такой компрессии происходит поиск повторяющихся комбинаций различных цветов, которые записываются в виде ключей. В последующем для кодирования изображения используются уже созданные ключи. Такой подход значительно лучше RLE при работе с областями, имеющими переходы цветов, однако кодировка в LZW требует больше системных ресурсов.

JPEG (Joint Photographic Experts Group)

Формат является, прежде всего, наиболее распространенной технологией сжатия файлов современности.

Технология JPEG реализуется так: для начала рисунок делится на блоки размером 8x8 пикселов, после записывается 2 типа данных - усредненная информация о блоке и сведения о его деталях. Затем, в зависимости от выбранной степени сжатия, удаляется те или иные

дополнительные данные, при чем, чем меньше будет размер файла, тем ниже будет его качество.

При каждом открытии и закрытии файла в растровом редакторе - он записывается снова, что влечет падение качества изображения. В этой связи большое количество перезаписей может привести к абсолютной непригодности изображения.

JPEG лучше применять для картинок, допускающих нечеткость краев и минимальную размытость. Тем не менее, для хранения, например, графиков лучше отдать предпочтение другим форматам.

JPEG 2000

Подтип обычного формата JPEG. В настоящее время он становится более распространен относительно обычного JPEG. Принципиальным их отличием способность JPEG 2000 сжимать файлы практически без потери качества (отсутствует заметное визуальное ухудшение качества).

PICT (Macintosh QuickDraw Picture Format)

Применяется преимущественнее на компьютерах Macintosh. Поддерживается всеми программами на этих компьютерах и даже служит стандартом буфера обмена. PICT позволяет хранить векторную информацию, текст, а также звук. Данный формат характеризуется большими возможностями сжатия изображений при условии наличия больших областей одного цвета (принцип RLE-компрессия).

PNG (Portable Network Graphics)

Самый перспективный формат Интернет. PNG был разработан в роли замены формата GIF. PNG поддерживает любое количество цветов. Сжатие реализуется по принципу, сходному с LZW, но немного усовершенствованному, что позволяет получать неплохие результаты. Кроме того, PNG поддерживает чересстрочную развертку, причем двойную - по горизонтали и вертикали одновременно.

Встроенная гамма-коррекция делает возможным корректно правильный вывод цветов на всех компьютерах, вне зависимости от платформы.

RAW

Является самым гибким графическим растровым форматом обмена файлами между компьютерами разных платформ. RAW позволяет сохранять изображения в любой цветовой модели, включая Lab и многоканальный, кроме того, поддерживает альфа-каналы.

В настоящее время RAW можно считать основным форматом для многих цифровых фотоаппаратов.

TIFF (Tagged Image File Format)

Самый популярный в последнее время растровый графический формат в издательском деле в силу наличия ряда достоинств:

- аппаратно независим, поэтому многие специально используют его для переноса изображений с PC на Macintosh и наоборот;
- практически все программы читают файлы данного формата без проблем;
- поддерживает все цветовые модели, включая CMYK и PANTONE;
- может нести дополнительную информацию, например, о контурах или альфа-каналах;
- поддерживает LZW-компрессию.

Формат TIFF, прежде всего, предназначен для печати результирующего изображения.

4.3 Растровые алгоритмы

Подавляющая часть современных графических устройств являются растровыми, что позволяет отображать рисунки в виде прямоугольной матрицы (сетки, целочисленной решетки) пикселей (растра). Большинство графических библиотек содержат внутри себя значительное количество

простейших растровых алгоритмов. На рисунке 4.6 приведена схема классификации растровых алгоритмов.

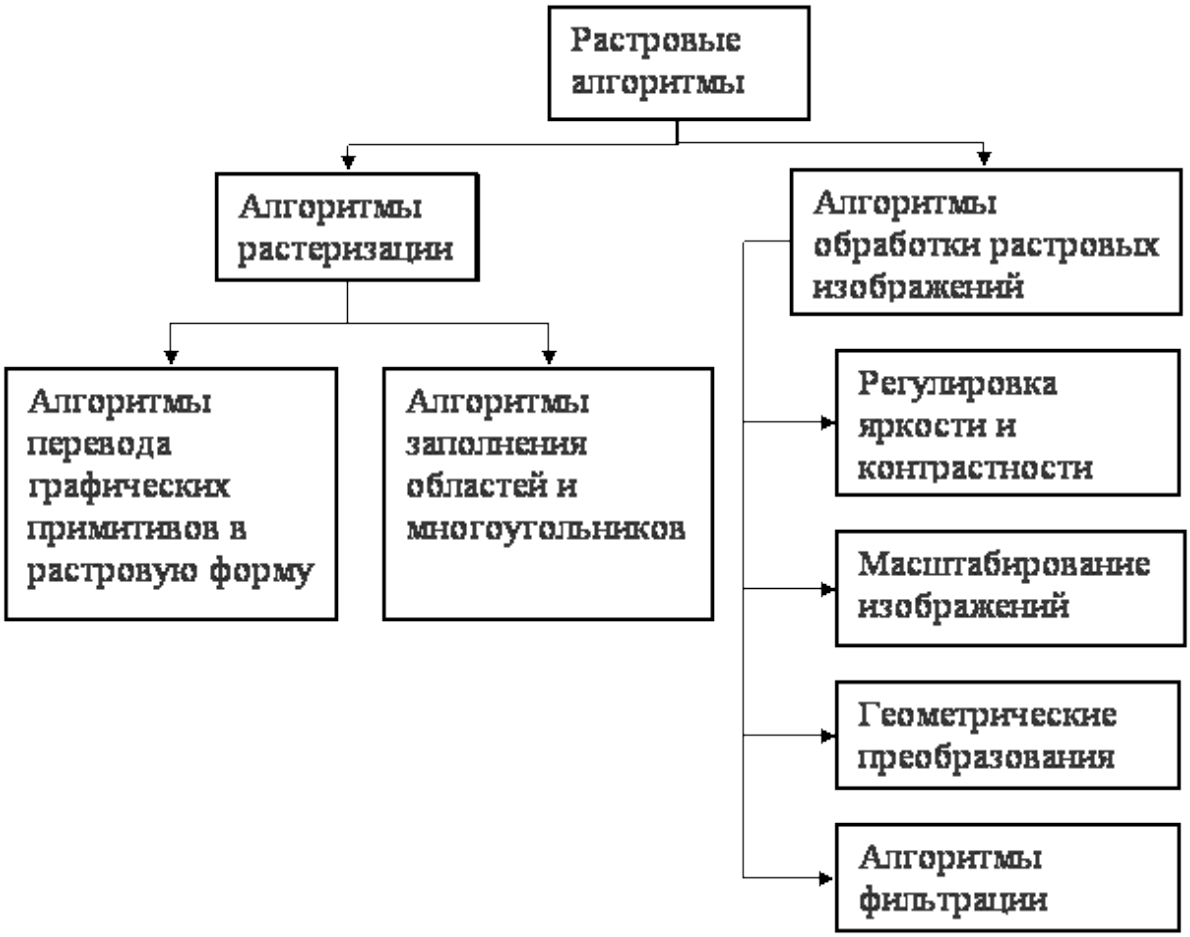


Рисунок 4.6 - Классификация растровых алгоритмов

4.3.1 Растровое представление отрезка. Алгоритм Брезенхейма

Рассмотрим задачу построения растрового изображения отрезка, соединяющего точки $A(x_a, y_a)$ и $B(x_b, y_b)$. Для простоты будем считать, что $0 \leq y_b - y_a \leq y_a - x_a$.

Тогда отрезок описывается уравнением:

$$y = y_a + \frac{y_b - y_a}{x_b - x_a}(x - x_a), \qquad (4.6)$$

где $(x–x_a)$, $x \in [x_a, x_b]$ или $y = cx + b$.

Таким образом, получаем простейший алгоритм растрового представления отрезка:

1) $c = \frac{y_b - y_a}{x_b - x_a}$;

2) $b = y_a - c \times x_a$

3) рассчитываем и увеличиваем значение $x = x_a$ до тех пор, пока оно не превысит значение x_b;

4) $y = c \times x + b$. Вычислений значений функции $y = c \times x + b$ можно избежать, используя в цикле рекуррентные соотношения, так как при изменении x на 1 значение y меняется на c [15].

Не смотря на то, что описанный алгоритм построения отрезка является простейшим, он обладает рядом существенных минусов:

1) выполняет операции над числами с плавающей точкой, хотя лучше работать с целочисленной арифметикой;

2) на каждой новой итерации выполняется операция округления, что также снижает быстродействие.

Эти недостатки устранены в алгоритме Брезенхейма, приведенном ниже.

Как и в предыдущем случае, будем считать, что тангенс угла наклона отрезка принимает значение в интервале [0;1]. Рассмотрим i-й шаг алгоритма (рисунок 4.7). На этом этапе пиксель P_{i-1} уже найден как ближайший к реальному отрезку. Требуется определить, какой из пикселов (T_i или S_i) будет установлен следующим [11].

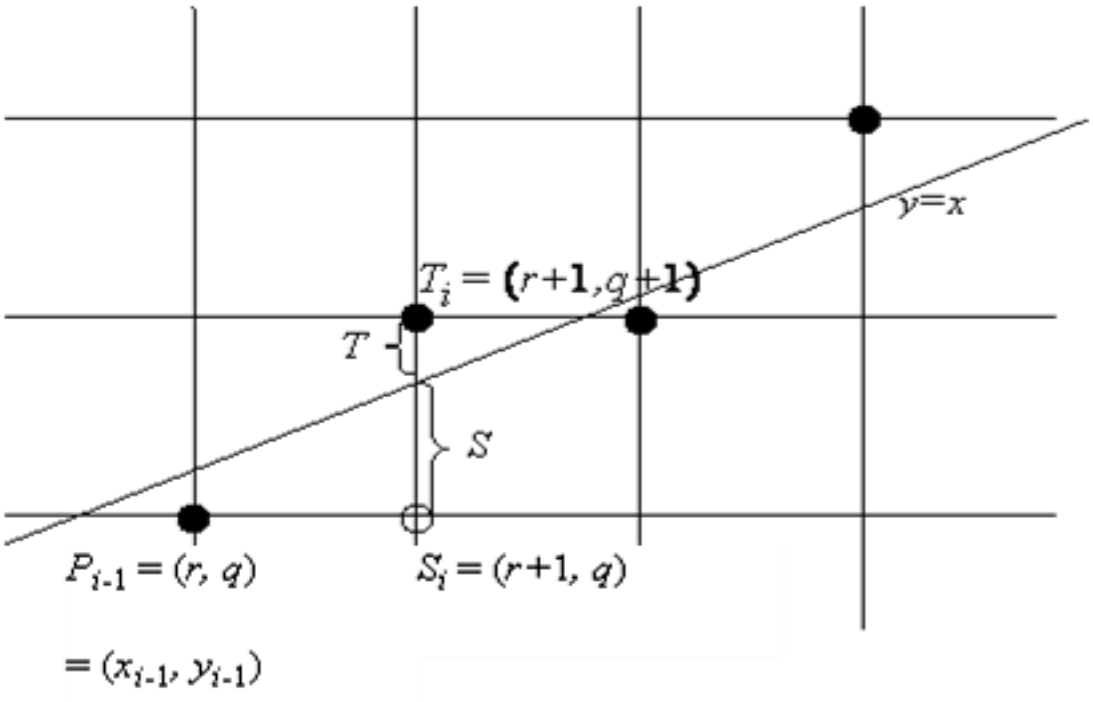


Рисунок 4.7 - i-й шаг алгоритма Брезенхейма

В рассматриваемом алгоритме Брезенхейма пользуются управляющей переменной d_i, которая на каждой итерации пропорциональна разности между S и T. Если S < T, то S_i ближе к отрезку, в противном случае T_i.

Предположим, что представляемый отрезок проходит из точки (x_1, y_1) в точку (x_2, y_2). Исходя из начальных условий, точка (x_1, y_1) ближе к началу координат. Тогда перенесем оба конца отрезка с помощью преобразования $T(-x_1, -y_1)$, так чтобы первый конец отрезка совпал с началом координат. Начальной точкой отрезка стала точка (0, 0), конечной точкой – (dx, dy), где $dx = x_2 - x_1$, $dy = y_2 - y_1$ (рисунок 4.8).

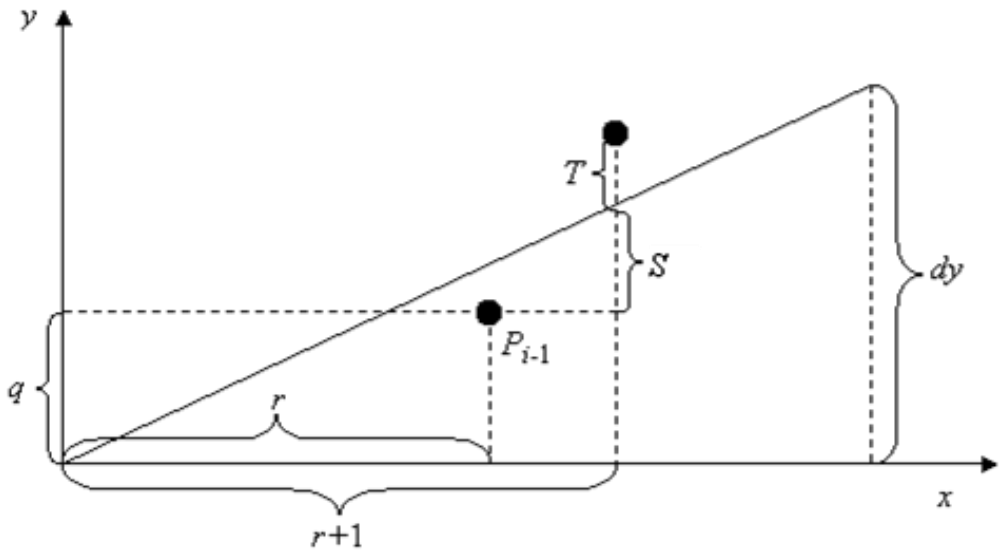


Рисунок 4.8 - Вид отрезка после переноса в начало координат

Уравнение прямой в этом случае будет иметь вид:

$$y = x\frac{dx}{dy}. \quad (4.7)$$

Обозначим координаты точки P_{i-1} после переноса через (r, q). Тогда $S_i = (r + 1, q)$ и $T_i = (r + 1, q + 1)$.

Из подобия треугольников на рисунке 4.8 можно записать, что

$$\frac{dy}{dx} = \frac{S+q}{r+1}. \quad (4.8)$$

Выразим S:

$$S = \frac{dy}{dx}(r + 1) - q. \quad (4.9)$$

T можно представить как T = 1 – S. Используем предыдущую формулу

$$T = 1 - S = 1 - \frac{dy}{dx}(r + 1) - q. \quad (4.10)$$

Найдем разницу S – T:

$$S - T = \frac{dy}{dx}(r + 1) - q - 1 + \frac{dy}{dx}(r + 1) - q = 2\frac{dy}{dx}(r + 1) - 2q - 1. \quad (4.11)$$

Помножим левую и правую часть на dx:

$$dx\,(S - T) = 2\,dy\,(r + 1) - 2\,q\,dx - dx = 2\,(r\,dy - q\,dx) + 2\,dy - dx.$$

Величина dx положительная, поэтому неравенство $dx\,(S - T) < 0$ можно использовать в качестве проверки при выборе S_i. Обозначим $d_i = dx\,(S - T)$, тогда

$$d_i = 2\,(r\,dy - q\,dx) + 2\,dy - dx.$$

Поскольку $r = x_{i-1}$ и $q = y_{i-1}$, то

$$d_i = 2\,x_{i-1}dy - 2\,y_{i-1}\,dx + 2\,dy - dx.$$

Прибавляя 1 к каждому индексу найдем d_{i+1}:

$$d_i + 1 = 2\,x_i\,dy - 2\,y_i\,dx + 2\,dy - dx.$$

Вычитая d_i из d_{i+1} получим

$$d_{i+1} - d_i = 2\,dy\,(x_i - x_{i-1}) - 2\,dx\,(y_i - y_{i-1}).$$

Известно, что $x_i - x_{i-1} = 1$, тогда

$$d_{i+1} - d_i = 2\,dy - 2\,dx\,(y_i - y_{i-1}).$$

Отсюда выразим d_{i+1}:

$$d_i + 1 = d_i + 2\,dy - 2\,dx\,(y_i - y_{i-1}).$$

Таким образом, получили итеративную формулу вычисления управляющего коэффициента d_{i+1} по предыдущему значению d_i. С помощью управляющего коэффициента выбирается следующий пиксель – S_i или T_i.

Если $d_i \geq 0$, тогда выбирается T_i и $y_i = y_{i-1} + 1$, $d_i + 1 = d_i + 2\,(dy - dx)$. Если $d_i < 0$, тогда выбирается S_i и $y_i = y_{i-1}$ и $d_{i+1} = d_i + 2\,dy$.

Начальные значения d_1 с учетом того, что $(x_0, y_0) = (0, 0)$, $d_1 = 2\,dy - dx$ [16].

Преимуществом алгоритма является его минимальные требования к арифметическим возможностям: сложение, вычитание и сдвиг влево для умножения на два.

Если dy > dx, то необходимо будет использовать этот же алгоритм, но пошагово увеличивая y и на каждом шаге вычислять x.

4.3.2 Закраска области, заданным цветом границ

Рассмотрим область, ограниченную группой пикселей определенного цвета и точку (x, y), расположенную внутри данной области.

Задача заливки области заранее заданным цветом является относительно трудной, если эта область не является выпуклой.

Рекурсивный алгоритм закраски области, заданной цветом границы является достаточно неэффективным, потому как для любого уже отрисованного пикселя функция вызывается ещё 4 раза, также алгоритм требует слишком большого объёма стека из-за большой глубины рекурсии. В силу выше сказанного, для решения задачи закраски области заданным цветом предпочтительнее алгоритмы, способные обрабатывать сразу целые группы пикселей, то есть применять их «связность». Если данный пиксель принадлежит определенной области, то, скорее всего, его ближайшие соседи также принадлежат данной области.

Массивом соседних пикселов чаще всего выступает полоса, определяемая правым пикселем. Для хранения правых определяющих пикселов используется стек. Алгоритм будет состоять из следующих шагов:

1) сначала заполняется горизонтальная полоса пикселов, содержащих начальную точку;

2) справа налево проверяется строка, предыдущая по отношению к только что заполненной полосе для определения самого правого пикселя каждой строки;

3) адреса найденных пикселов заносятся в стек;

4) все выше описанное повторяется и для строки, следующей и за последней заполненной полосой;

5) когда строка обработана таким способом, в качестве новой начальной точки используется пиксель, адрес которого берется из стека.

Для него повторяется вся описанная процедура. Алгоритм заканчивает свою работу, если стек пуст.

4.3.3 Растровая развёртка окружности

В настоящее время существуют множество относительно не сложных, но при этом достаточно не эффективных способов преобразования окружностей в растровую форму. Мы в качестве примера приведем окружность с центром в начале координат. Ее уравнение записывается как $x^2 + y^2 = R^2$. Решая это уравнение относительно y, получим

$$y = \pm\sqrt{R^2 - x^2}. \quad (4.12)$$

Для графического представления четвертой части окружности, необходимо изменять x с единичным шагом от 0 до R и на каждом шаге вычислять y. Другим же простым способом растровой развертки окружности является использование вычислений x и y по формулам $x = R\cos\alpha$, $y = R\sin\alpha$ при пошаговом изменении угла α от 0° до 90°.

Для упрощения алгоритма растровой развёртки стандартной окружности будет лучше использовать её симметрию относительно координатных осей и прямых $y = \pm x$; в случае, когда центр окружности не совпадает с началом координат, эти прямые необходимо сдвинуть параллельно так, чтобы они прошли через центр окружности. Тем самым достаточно построить растровое представление для $\frac{1}{8}$ части окружности, а все оставшиеся точки получить симметрией [17]. Принцип представлен на рисунке 4.9.

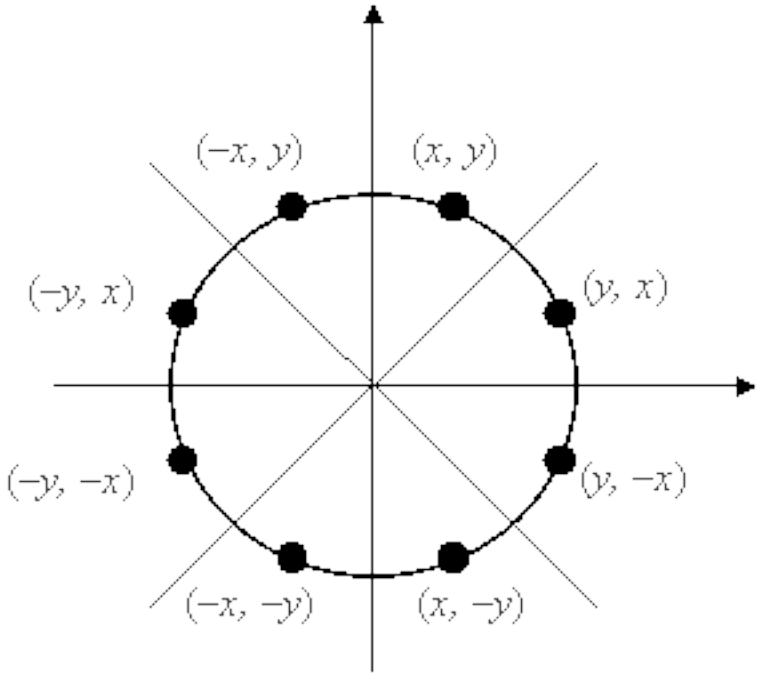


Рисунок 4.9 - Восьмисторонняя симметрия

Рассмотрим часть окружности из второго октанта $x \in \left[0, \frac{R}{\sqrt{2}}\right]$ по алгоритму Брезенхейма.

Каждая новая итерация алгоритма определяет ближайшую к истинной окружности точку P_i (x_i, y_i). Суть метода состоит в выборе ближайшей точки, используя управляющие переменные, значения которых можно вычислить на каждом шаге с помощью небольшого числа операций: сложений, вычитаний и сдвигов.

Рассмотрим небольшой участок сетки пикселов и всевозможные способы (от A до E) прохождения истинной окружности через сетку (рисунок 4.10). Допустим, точка P_{i-1} была выбрана как ближайшая к окружности при $x = x_{i-1}$. Теперь найдем, какая из точек (S_i или T_i) расположена ближе к окружности при $x = x_{i-1} + 1$.

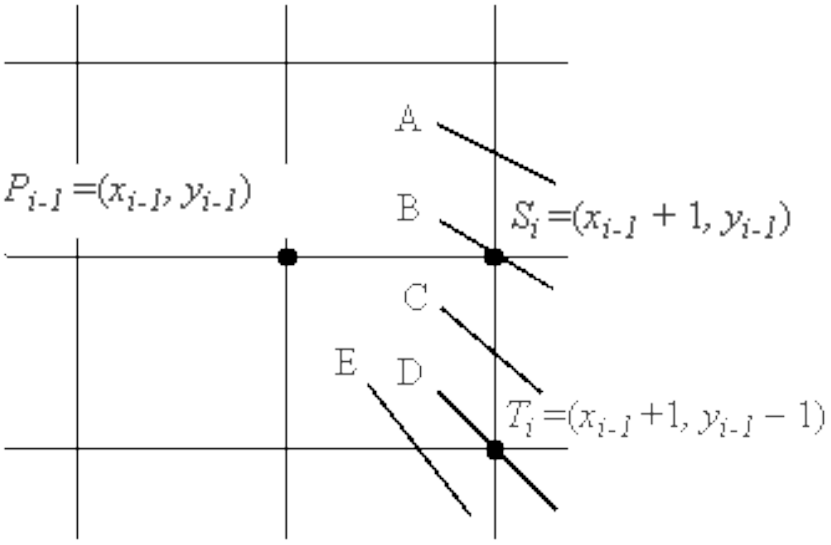


Рисунок 4.10 - Варианты прохождения окружности через растровую сетку

Заметим, что ошибка при выборе точки P_i (x_i, y_i) была равна $D(P_i) = (x_i^2 + y_i^2) - R^2$. Запишем выражение для ошибок, получаемых при выборе точки S_i или T_i:

$$D(S_i) = [(x_{i-1} + 1)^2 + (y_{i-1})^2] - R^2$$
$$D(T_i) = [(x_{i-1} + 1)^2 + (y_{i-1})^2] - R^2$$

Если $|D(S_i)| \geq |D(T_i)|$, то T_i ближе к реальной окружности, иначе выбирается S_i.

Введем $di = |D(S_i)| - |D(T_i)|$.

T_i будет выбираться при $d_i \geq 0$, в противном случае будет устанавливаться S_i.

Опуская алгебраические преобразования, запишем d_i и d_{i+1} для разных вариантов выбора точки S_i или T_i. $D_1 = 3 - 2R$.

Если выбирается S_i (когда $d_i < 0$), то $d_i + 1 = d_i + 4x_i - 1 + 6$.

Если выбирается T_i (когда $d_i \geq 0$), то $d_i + 1 = d_i + 4(x_i - 1 - y_i - -1) + 10$.

Существует модификация алгоритма Брезенхейма для эллипса [14].

4.3.4 Заполнение многоугольника

Одной из распространенных задач графики является - заполнение многоугольников, заданных набором вершин, которая решается в 2этапа:

1) проводится операция отсечения многоугольника;

2) заполнение полученных областей-многоугольников.

Первый этап требуется для определения реальных областей многоугольника, которые будут выведены на экран, что потребуется в случае если многоугольник больше или выходит за пределы экрана или окна вывода. Тем не менее отсечение может проводиться относительно и другого многоугольника. При этом порождается новый многоугольник или несколько новых многоугольников.

Рассмотрим алгоритм Сазерленда-Ходгмана, суть которого заключается в сведении решения общей задачи к решению ряда простых и похожих подзадач. Таковым примером является отсечение многоугольника относительно одной отсекающей границы. Последовательное решение 4-х подобных задач позволяет провести отсечение относительно прямоугольной области. Суть алгоритма графически представлена на рисунке 4.11.

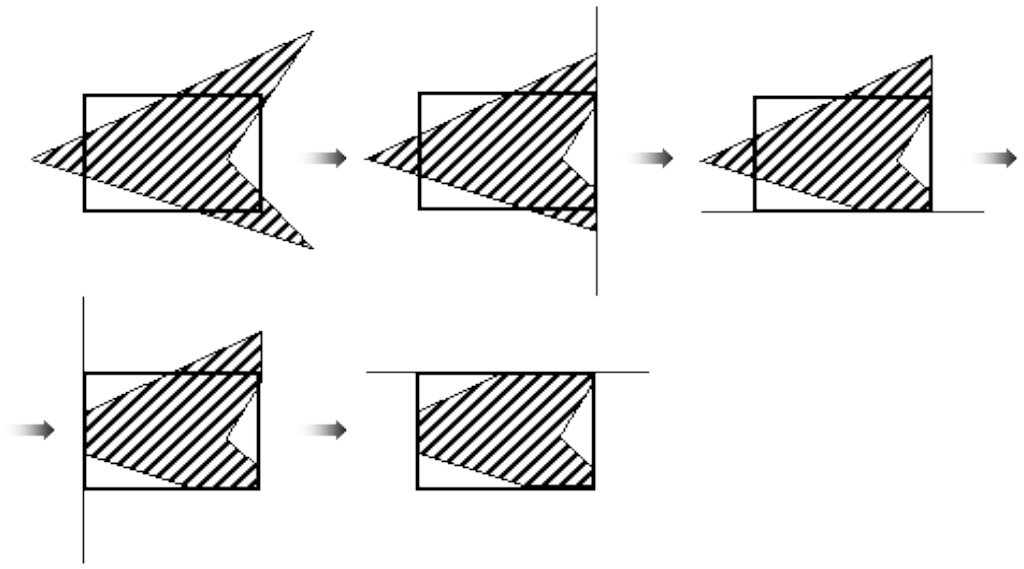


Рисунок 4.11 - Последовательное отсечение многоугольника

На вход алгоритма поступает последовательность вершин многоугольника, который подвергается отсечению - V_1, V_2, …, V_n. Эти ребра проходят от V_i к V_{i+1} от V_n к V_1. Используя разработанный алгоритм проводится отсечение относительно ребра и выводится другая новая последовательность его вершин, описывающая усеченный многоугольник.

Алгоритм проходит вокруг многоугольника от V_n к V_1 и обратно к V_n, проверяя на каждой итерации соотношение между последовательными вершинами и отсекающей границей. В данной задаче существуют четыре случая, представленных на рисунке 4.12.

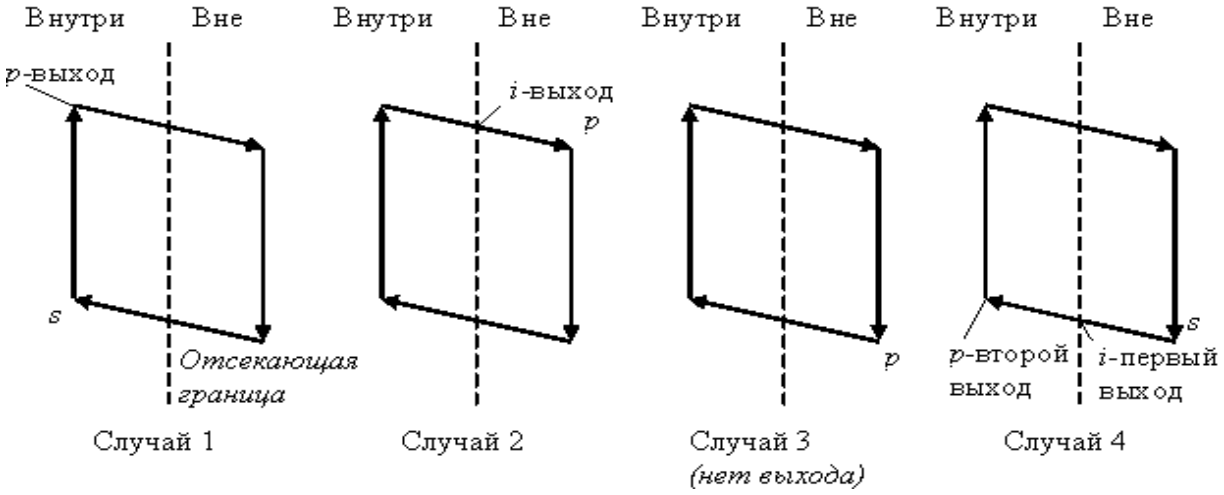


Рисунок 4.12 - Случаи, возникающие при отсечении многоугольников

В первом случае ребро полностью находится внутри отсекающей границы, а к результирующему списку приписывают вершину p. Вторая ситуация отражает пример, когда в качестве вершин выводится точка пересечения i, потому что ребро пересекает границу, тогда как начальная точка выведена была еще при анализе первого случая. В третьем случае обе вершины лежат вне отсекающих границ и ни одна из них не выводится. В четвертом примере к выходному списку добавляется точка пересечения i, а также вершина p. При отсечении многоугольника по алгоритму Сазерленда-Ходгмана возникает проблема, связанная с появлением ребер, частично

совпадающих с границей окон. Лишние ребра можно устранить, воспользовавшись дополнительной обработкой или же более сложным алгоритмом Вейлера – Азертона.

Заполнение многоугольников

Приведем алгоритм заполнения многоугольников, заданных замкнутой ломаной линией без самопересечений.

Самый простой метод заключается в проверке принадлежности каждой точки этому многоугольнику. Более эффективные алгоритмы используют тот факт, что соседние пикселы, как правило, имеют одинаковые характеристики (кроме пикселов граничных ребер). Свойство получило название пространственной когерентности.

В случае с многоугольником когерентность пикселей определяется вдоль сканирующей строки. Сканирующие строки обычно изменяются от верхнего края многоугольника до нижнего. Характеристики пикселов изменяются только там, где ребро многоугольника пересекает строку. Эти пересечения делят сканирующую строку на области закрашенных и не закрашенных пикселов.

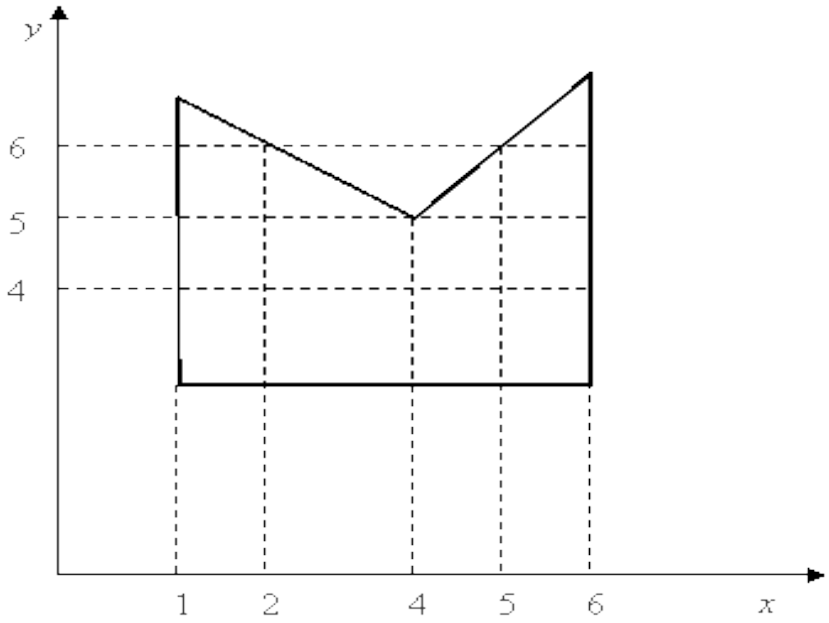


Рисунок 4.13 - Прохождение сканирующих строк по многоугольнику

Приведем примеры случаев, котореы могут возникнуть при делении многоугольника на области сканирующей строкой.

1) Первый случай является элементарным примером деления. На рисунке 4.13 сканирующая строка при y = 4 пересекает многоугольник при x = 1 и x = 6, формируя три области: x< 1; 1 ≤ x ≤ 6; x > 8. Сканирующая строка y = 6 пересекает многоугольник при x = 1; x = 2; x = 5;x = 6, тем самым выделяя 5 областей: x< 1; 1 ≤ x ≤ 2; 2 < x < 5; 5 ≤ x ≤ 6; x > 6. В данном примере x сортируется по возрастанию, затем список X рассматривается попарно. Между парами точек пересечения закрашиваются все пикселы. Для y = 4 закрашиваются пикселы в интервале (1, 6), для y = 6 закрашиваются пикселы в интервалах (1, 2) и (5, 6).

2) Во второй ситуации деление характеризуется прохождением сканирующей строки через вершину. Случай отражен на рисунке 4.14.

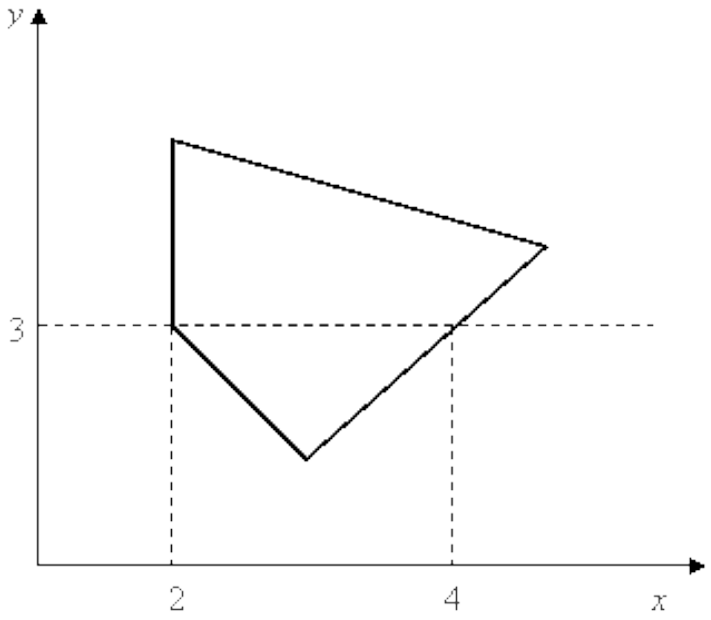


Рисунок 4.14 - Прохождение сканирующий строки через вершину

К примеру, по сканирующей строке y = 3 упорядоченный список x получится как (2, 2, 4). В силу того, что вершина многоугольника учитывается дважды, поэтому закрашиваемый интервал получается

неверным: (2, 2). Таким образом, когда сканирующая строка пересекает вершины - она должна учитываться один раз. В таком случае список по x в описанном примере будет (2, 4).

3) Третьим случаем деления многоугольника на области сканирующей строкой является ситуация, когда строка проходит через локальный минимум или максимум (рисунок 4.13 при y = 5). Для этого примера требуется учитывать все пересечения вершин сканирующей строкой. На рисунке 4.13 при y = 5 формируется список x (1, 4, 4, 6). Закрашиваемые интервалы (1, 4) и (4, 6). При рассмотрении концевых вершин для ребер, соединенных в ней, находится локальный минимум или максимум. Если у обоих концов координаты Y больше, чем у вершины пересечения, то вершина определяется как локальный минимум, в противном случае вершина является локальным максимумом.

Для повышения скорости выдачи результата алгоритмом можно воспользоваться списком активных ребер (САР). САР включает ребра многоугольника, которые пересекают сканирующую строку. В момент пересечения очередной сканирующей строки вершины многоугольника, из списка исключаются ребра, которые находятся выше, и добавляются концы, которые пересекает сканирующая строка. Запуск алгоритма позволяет выявлять пересечения сканирующей строки только с ребрами из списка активных [18].

В данной главе были рассмотрены наиболее распространенные алгоритмы растеризации. Несмотря на относительно сложное их математическое обеспечение, они довольно просты в реализации, а, кроме того, имеют широкий спектр практического применения.

4.4 Практическое применение растровой графики в экономике

Наиболее простым примером практического применения растровой графики для всех сфер деятельности, в частности для экономистов-

управленцев, является интернет-сайты. Вся всемирная паутина на сегодняшний момент работает с растровыми изображениями, а значит при формировании рекламного сайта или сайта-визитки компании не избежать работы с рисунками в растровом формате. Кроме того, растровые изображения удобны для передачи по факсу, что является одним из важнейших требований ведения документации в коммерческой деятельности у территориально распределенных филиалов.

Кроме того, логотипы компаний на сегодняшний момент создаются в растровой графике. Как показывает проект **Ecobranding Сильвена Бойера,** качественный, современный, а главное простой логотип крупных корпораций экономят огромное количество чернил, а это в свою очередь экономит вложение денежных средств. Всё связано: печатные станки начнут тратить меньше электроэнергии, значит, электростанции будут делать меньше вредных выбросов и далее по цепочке. Упрощая логотип экономия выходит в пределах от 10 до 39 процентов. Таким образом, внесены изменения в логотип «Макдоналдса», как наиболее показательный пример. Это позволило сэкономить 35 % чернил при использовании офсетной печати, то есть миллионы галлонов в год. Чернила для принтеров действительно являются одной из самых дорогих в мире жидкостей.

В 2013 году компания Consumer Reports рассчитала, что их стоимость составляет от 700 до 4500 рублей за 30 мл, что делает чернила более дорогими, чем шампанское или духи.

В качестве экономичного примера следует привести яблоко компании Apple (рисунок 4.15).

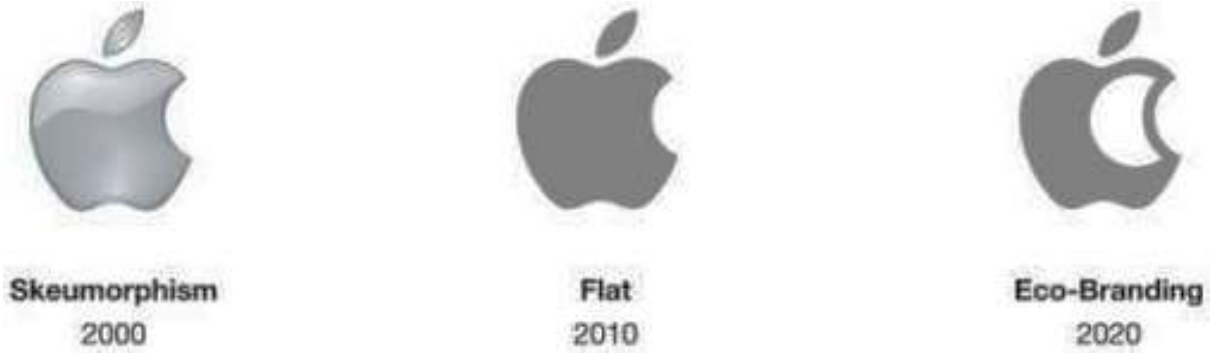


Рисунок 4.15 – Логотип компании Apple

Кстати, похожим образом мыслит голландская компания SPRANQ, которая в 2008 году начала создавать «экологически чистые» шрифты, а потом выпустила приложение, способное адаптировать чужие шрифты для экономии чернил.

Аналогичный ход сделало лондонское рекламное агентство Gray совместно с канцелярским брендом Ryman. Его представители разработали бесплатный шрифт Ryman Eco, который при печати использует на 30 процентов меньше чернил, чем популярные Arial, Times New Roman.

Актуально применением растра и для печати рекламных листовок.

Кроме того, растровая графика полезна при фомрмировании файлов отчетов на несколько десятков или даже сотен страниц с большим количеством изображений (графики, схемы и так далее) в силу того, что растровые файлы не занимают много дискового пространсва.

Алгоритмы растровой развертки окружностей применяются для построения экономических кластеров, сегментирования рынка, потребителей и формирование собственных номенклатурных групп. В методах затравочного заполнения предполагается, что известна некоторая точка (затравка) внутри замкнутого контура. В алгоритмах ищут точки, соседние с затравочной и расположенные внутри контура. Если соседняя точка расположена не внутри, значит, обнаружена граница контура. Если же точка оказалась внутри

контура, то она становится новой затравочной точкой и поиск продолжается рекурсивно.

Иной подход используется в алгоритмах заполнения с затравкой. В них предполагается, что известен хотя бы один элемент, классификация которого не поддается сомнению. Алгоритм пытается найти и закрасить все другие элементы, принадлежащие внутренней области. Области могут быть либо внутренние, либо гранично-определенные. Если область относится к внутренне-определенным, то все элементы, принадлежащие внутренней части, имеют одну и ту же принадлежность (алгоритм их окрашивает в один цвет), а все элементы, внешние по отношению к области, имеют другой цвет. Если область относится к гранично-определенным, то все элементы на границе области имеют выделенное значение или цвет. Алгоритмы, заполняющие внутренне-определенные области, называются внутренне-заполняющими, а алгоритмы для гранично-определенных областей - гранично-заполняющими.

5 Векторная графика

5.1 Общие сведения о векторной графике

Относительно выше рассмотренной растровой графики векторная позволяет стоить объекты, используя их математическое описание. Такой подход первоначально может показаться более трудоемким, нежели применение растровых массивов. Однако, для ряда объектов опираться на математическое описание является значительно проще.

Принципиальным отличием **векторной графики** следует считать применение комбинации компьютерных команд и математических формул для картинок. Все это способствует автоматизированному вычислению для расположения в правильном месте реальной точки во время отрисовки этих изображений. Данный принцип дает векторной графике значительные преимущества в сравнении с растровой. При этом, векторная графика характеризуется рядом недостатков.

Применительно к векторной графике еще используют термин **объектно-ориентированная графика** или чертежная графика.

Примитивами векторной графики называются элементарные, неделимые объекты, к которым относят окружности, линии, сферы, кубы и тому подобное. Эти простейшие объекты применяют для более сложных, комбинированных графических картин. Для того, чтобы создать объекты примитивов пользуются простыми описаниями. Скажем, прямая линия, дуги, окружности, эллипсы и области однотонного или изменяющегося света - это двухмерные изображения, применяемые для отрисовки детализированных картинок. В 3D графике при формировании сложных рисунков в большинстве своем используют объемные фигуры: сферы, кубы. Команды, описывающие векторные объекты простому пользователю скорее всего никогда не будут доступны. Каким образом описываются объекты определяет компьютерная программа, которая применяется для подготовки

векторных объектов. Для создания векторных рисунков необходимо использовать один из многочисленных пакетов графики.

Ключевым логическим объектом векторной графики считается геометрический объект.

Под **объектами** понимают простые геометрические фигуры, иначе называемые примитивами. Кроме того, важнейшим составляющим элементом векторной графики считается сплайн. Сплайном называется кривая, используя которую возможно описать различные геометрические объекты. Некоторые современные шрифты, такие как TrueType и PostScript, в своей основе имеют сплайн.

Все распространенные форматы векторной графики характеризуются большим спектром цветовых возможностей. Самые простые форматы всегда пользуются цветом своего устройства вывода и могут, при этом, не включать никаких данных о цвете в принципе. Однако, более сложные форматы способны хранить информацию о полном 32-битном цвете. Выбор цветовой модели, используемой векторным форматом, никаким образом не влияет на размер изображения, за исключением, когда файл содержит растровые образы. В обычных векторных картинах значение цвета относится ко всему объекту в целом, а его цвет сохраняется в виде части его векторного описания.

Различные векторные файлы ориентированы на формирование растровых эскизов рисунков, которые хранятся в нем. Данные растровые картинки порой называют кратким описанием изображений. Краткое описание фактически является эскизом векторных рисунков в целом. Данные эскизы бывают необходимы в случаях, когда нет необходимости открывать весь файл, чтобы просмотреть его содержимое, или же когда нет возможности увидеть векторный рисунок в момент его использования. Первый случай имеет место, когда требуется найти файл посредством программных пакетов. Чтобы упростить процесс поиска нужного векторного файла эти программы могут считывать растровый эскиз изображения и

другие параметры. К таковым параметрам относят формат, время создания, битовую глубину изображения и тому подобное. Второй случай бывает, когда в каком-либо графическом пакете издательского дела помещается на страницу векторный рисунок. Видимое изображение будет являться растровым эскизом настоящего векторного рисунка. При этом, у рисунка нельзя менять размер, обрезать его или как-то иначе модифицировать. Стоит отметить, что эскизы - это растровая версия рисунков, а растровые данные используют много памяти компьютера [19].

Векторная графика обладает рядом полезных достоинств. К примеру, данный вид компьютерной графики существенно экономит дисковое пространство, достаточное для хранения изображений: это объясняется тем, что сохраняется не сама картинка, а лишь некоторая информация. Применяя в последующем основные сведения о файле, графическая программа каждый раз воссоздает рисунок заново. Более того, описание цветовых характеристик не существенно увеличивает размер файла.

Объекты векторной графики не сложно трансформируются и модифицируются. Все преобразования объектов практически ни коем образом не влияют на качество изображения. Масштабирование, поворот, искривление могут быть сведены к минимальному количеству элементарных преобразований над векторами. Векторные программы незаменимы в тех практических областях применения векторной графики, где определяющим фактором является сохранение ясных и четких контуров, например, в шрифтовых композициях, в создании логотипов и тому подобное.

Основными составляющими векторных рисунков являются прямые линии, тогда как естественные объекты далеко не всегда можно назвать прямыми. До недавнего времени этот факт крайне ограничивал сферу практического применения векторной графики, предоставляя только изображения где естественность не обязательна, например, 2-мерные чертежи и круговые диаграммы, формируемые специальными программами автоматизированного проектирования, 2-х или 3-х мерные технические

изображения, стилизованные рисунки и значки, состоящие из прямых линий и областей, закрашенных однотонным цветом. Векторные картинки включают в себя всевозможные команды от ПК к устройствам вывода (принтеру). Графические пакеты векторной графики способны создавать растровые изображения в качестве одного из типов объектов. Это возможно потому, что растровый рисунок просто набор инструкций для компьютера, и так как инструкции эти весьма элементарны, то векторная графика может воспринимать растровые изображения наравне с остальными объектами, хотя можно поместить растровые изображения в виде объекта векторном формате, но не удается отредактировать и изменить в нем отдельные пикселы.

Недостатки векторного представления изображений:

- трудно получить фотореалистичное изображение;
- при прорисовке необходимо каждый раз растеризовать изображение;
- программная зависимость вследствие отсутствия единого формата.

5.2 Основные векторные форматы

HPGL - графический векторный формат разработанный для группы плоттеров компании Hewlett-Packard. Плоттером называют графопостроитель, иначе говоря аппарат, предназначенный для нанесения различных изображений на бумагу. Особенность движения пера плоттера заключается в том, некоторый заданный момент времени оно может перемещаться только по прямой. Именно с этим связано разбиение объектов в HPGL на отрезки (в таком виде изображения и записывают в файл). Несмотря на то, что метод относительно громоздкий, его можно считать достаточно простым и эффективным. HPGL в некоторой степени модифицированный и усложненный применяется до сих пор.

CDR (CorelDRAW)

Это основной формат программы CorelDRAW. CDR предназначен для хранения различного рода объектов как векторной графики, так и точечных изображений, текста, эффектов и многого другого.

Adobe PostScript

Настоящий формат векторной графики на сегодняшний момент считается стандартом в среде настольных издательских систем. Изначально разработчиками предполагалось применение формата в качестве языка отображения страниц на высококачественных выводных устройствах (лазерных принтеров и фотонаборных автоматов), однако, на сегодняшний момент время спектр применения Adobe PostScript несколько расширился: программы верстки и графические пакеты. Формат применяет в работе векторный и растровый способы записи информации.

Существует несколько разновидностей PostScript, в частности PDF (Adobe Portable Document Format) - формат программы Adobe Acrobat.

DXF (Drawing eXchange Format)

Формат ориентирован, прежде всего, на работу в системах автоматизированного проектирования (САПР), в первую очередь AutoCAD. В DXF разработчиками реализованы большинство принципиально новых возможностей, отсутствующие в большинстве других форматов, в частности, хранение объектов трёхмерной графики.

5.3 Векторная графика. Построение кривых

5.3.1 Общие сведения об особенностях построения кривых

Сплайн – это кривая, удовлетворяющая некоторым критериям гладкости.

Базовые (опорные) точки – набор точек, на основе которых выполняется построение кривой.

Интерполяция – построение кривой, точно проходящей через набор базовых точек.

Аппроксимация – сглаживание, приближение, то есть построение гладкой кривой, проходящей не через набор базовых точек, а вблизи них.

Экстраполяция – построение линии за пределами интервала, заданного набором базовых точек.

В простейшем случае интерполяция может быть реализована путем соединения базовых точек отрезками прямых линий. Этот способ называется линейной интерполяцией.

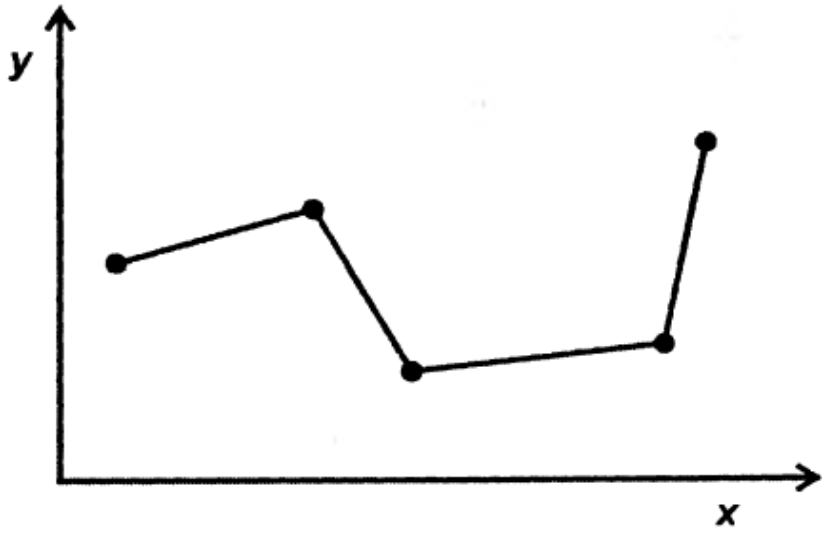


Рисунок 5.1 – Линейная интерполяция

Такая кривая точно проходит через набор базовых точек. Однако линейная интерполяция может не очень точно отражать поведение объекта эксперимента на интервалах между базовыми точками. Поэтому более приемлемой является интерполяция с помощью некоторой гладкой кривой.

Критерием гладкости является существование производных функции, описывающей кривую. Какого порядка существует производная такого порядка и гладкость. Обычно достаточно гладкой считается функция, если она имеет производную первого или второго порядка. Составная кривая

называется геометрически непрерывной, если вдоль этой кривой единичный вектор ее касательной изменяется непрерывно, и дважды геометрически непрерывной, если и вектор кривизны также меняется непрерывно.

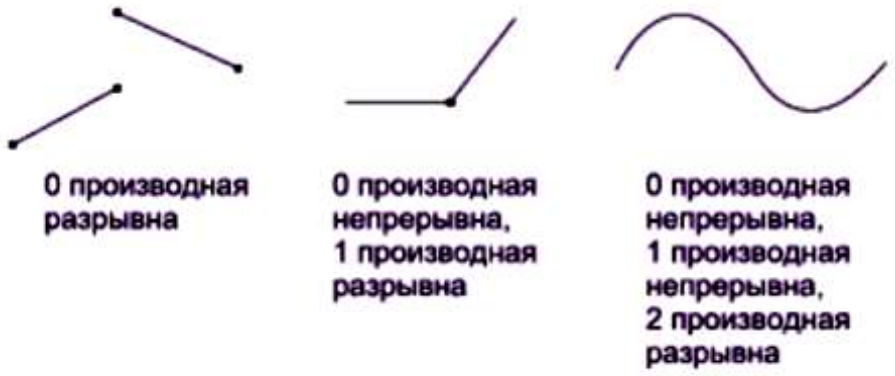


Рисунок 5.2 - Геометрически непрерывная кривая

Гладкая интерполяционная кривая может быть построена на основе базовых n+1 точек с помощью полинома степени n.

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + .. + a_1 x + a_0 = y. \quad (5.1)$$

Полиномом называется функция вида (5.1). В этом выражении неизвестными являются n+1 коэффициентов a_i. Для того, чтобы из найти, подставляем в уравнение n+1 раз координаты из набора базовых точек. В результате получаем систему из n+1 линейных уравнений.

Например, если набор точек состоит из трех, то степень полинома n=2, а коэффициенты a_i можно получить из следующей системы уравнений:

$$\begin{cases} a_2 x_1^2 + a_1 x_1 + a_0 = y_1 \\ a_2 x_2^2 + a_1 x_2 + a_0 = y_2, \\ a_2 x_3^2 + a_1 x_3 + a_0 = y_3 \end{cases} \quad (5.2)$$

Недостаток такого подхода (когда весь набор точек описывается одной функцией). Графики полиномов высоких степеней сильно меняются в промежутках между базовыми функциями. За пределами интервалов базовых точек полиномы имеют тенденцию сильно возрастать или убывать. Чем больше точек в наборе, тем больше уравнений необходимо решать. Чтобы избежать всех этих сложностей, при построении гладких кривых используют подход, заключающийся в формировании составной кривой из отдельных частей (сегментов). Составную кривую второго порядка гладкости можно образовать из дуг обыкновенных полиномов третьей степени. Для расчета коэффициентов такого полинома требуется четыре точки. При этом необходимо обеспечить гладкость в местах стыковки. На практике для построения сплайновой кривой обычно используют метод составления линий из отдельных сегментов, описываемые элементарными уравнениями, как правило, третьей степени [20].

Достоинства такого подхода:

- упрощение расчетов;

- использование уравнений невысоких степеней;

- при добавлении точки в базовый набор необходимо пересчитать только четыре сегмента кривой.

Сложные кривые удобно описывать в виде параметрических уравнений. Уравнение вида $x = x(t)$, $y = y(t)$, $z = z(t)$, $\alpha \leq t \leq \beta$ называют **параметрическим заданием кривой**, при этом переменная t называется **параметром**.

Кривая может быть представлена совокупностью точек. Если точки расположены близко друг от друга, то, соединяя их отрезками прямой, мы получим изображение кривой. На рисунке 5.3 показаны два разных точечных представления одной кривой. Точки на кривой 5.3 (a) расположены равномерно по ее длине. В результате получается довольно плохое представление кривой, особенно если мал радиус кривизны. Улучшить вид можно, увеличивая плотность точек в таких участках (рисунок 5.3 (b)).

Хотя, как показано выше, кривые могут быть достаточно хорошо представлены совокупностью точек, аналитическое представление во многих отношениях предпочтительнее. Его преимущества - это точность, компактность записи и простота вычисления промежуточных точек. Аналитическое представление позволяет без труда определить наклон и радиус кривизны, а при точечном представлении для этого требуется численное дифференцирование - чрезвычайно неточная процедура.

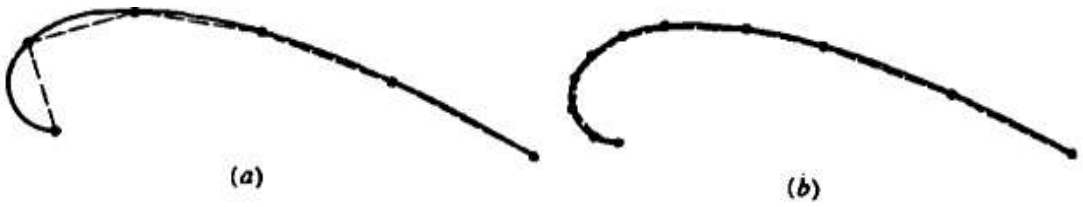


Рисунок 5.3 – Точечное представление кривых. Равномерная плотность точек вдоль кривой (а); плотность точек возрастает с уменьшением радиуса кривизны

5.3.2 Существует несколько **способов представления кривых на плоскости**:

- явный способ (explicit curves)

$$y = \sin x$$

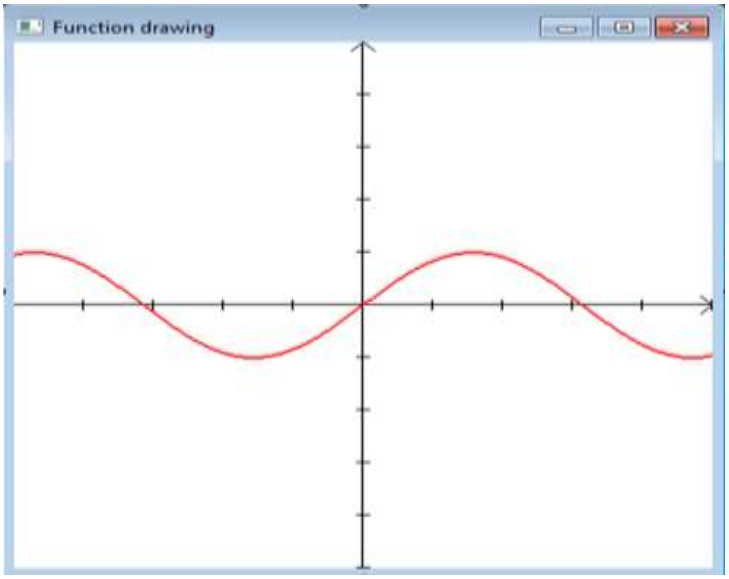


Рисунок 5.4 – Явное представление кривой на плоскости

- неявный способ (implicit)

$$f(x, y) = 0$$

$$x^2 + y^2 - R^2 = 0$$

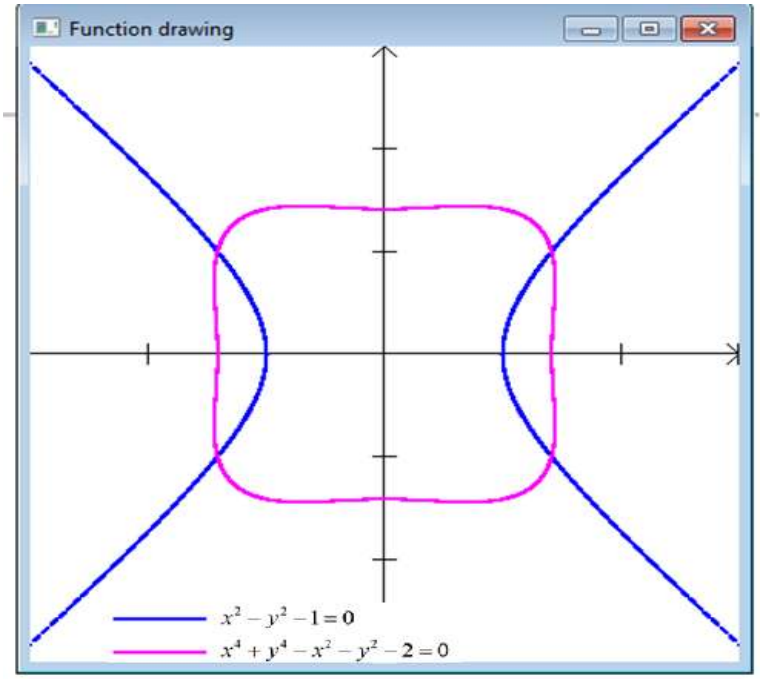


Рисунок 5.5 – Неявное представление кривой на плоскости

- параметрический способ (parametric curves) [21]

$$\begin{cases} x = f_x(t); \\ y = f_y(t). \end{cases} t \in [a, b] \qquad \begin{cases} x = \cos(t); \\ y = \sin(t). \end{cases} t \in [0, 2\pi)$$

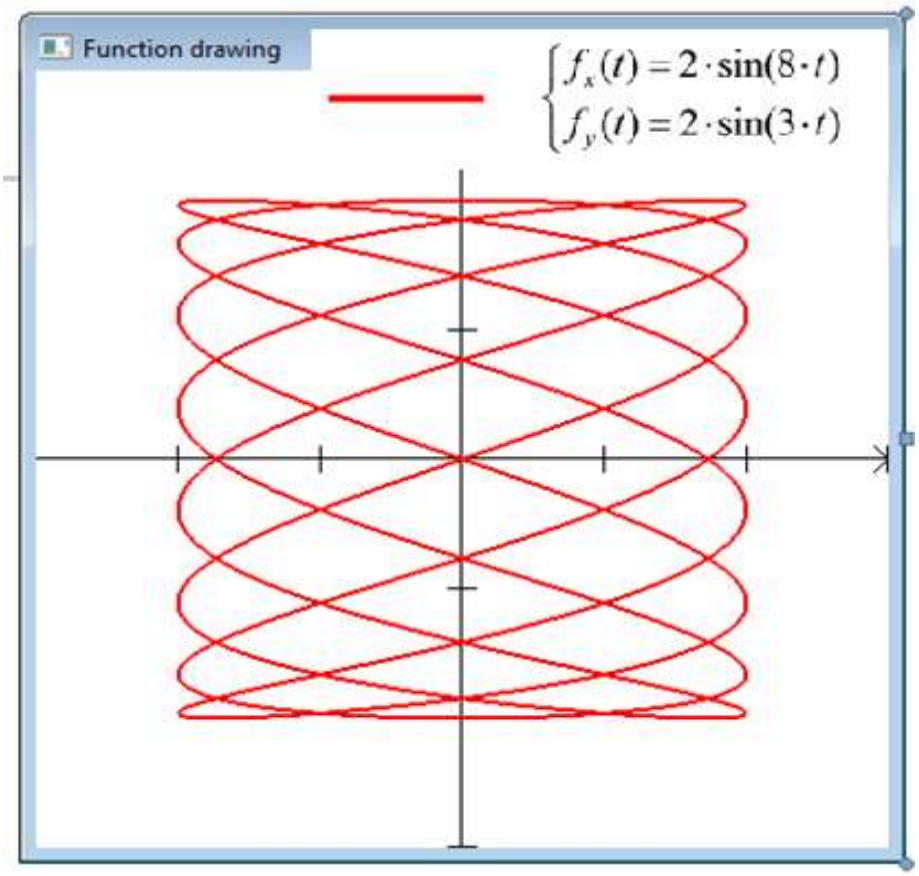


Рисунок 5.6 – Параметрический способ представления кривой на плоскости

5.3.3 **Кривые Безье** или **кривые Бернштейна-Безье** были разработаны в 60-х годах XX века независимо друг от друга Пьером Безье и Полем де Кастельжо, где применялись для проектирования кузовов автомобилей.

Кривая Безье является частным случаем многочленов Бернштейна, описанных С.Н. Бернштейном в 1912 г. Им найдены условия аналитичности решений уравнений второго порядка эллиптических и параболических типов, развиты новые методы решения граничных задач для нелинейных уравнений

эллиптического типа; совместно с учениками создана новая ветвь теории функций конструктивная теория функций. При доказательстве аппроксимационной теоремы Вейерштрасса Бернштейном были построены полиномы, оказавшиеся полезными в самых разных областях математики. Теперь их называют полиномами Бернштейна.

Кривая Безье является частным случаем многочленов Бернштейна, представляет собой параметрическую кривую и задается выражением:

$$B(t) = \sum_{i=0}^{n} P_i b_{i,n}(t), 0 \le t \le 1, \quad (5.3)$$

где n – количество опорных точек;

i – номер опорной точки;

t – шаг (например, при построении по 100 точкам шаг равен 0.01);

P – координата опорной точки;

b(t) – базисная функция кривой Безье (полином Бернштейна). Этот полином определяет вес опорной точки:

$$b_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-1}, \quad (5.4)$$

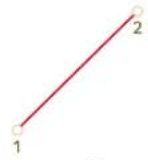
где $\binom{n}{i} = \frac{n!}{i!(n-i)!}$ число сочетаний из n по i

Впервые кривые были опубликованы в 1962 г. Пьером Безье. Он применил их для компьютерного проектирования кузова машины. Кривые были названы именем Безье, а именем де Кастельжо назван разработанный им **рекурсивный алгоритм**, по которому определяют кривые. Сегодня данный приём стал одним из важнейших средств САПР и систем компьютерной графики. Кривые Безье используются в компьютерной графике для рисования плавных изгибов, в CSS-анимации и много где ещё. В принципе, можно создавать анимацию и без знания кривых Безье, но стоит

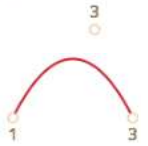
один раз изучить эту тему хотя бы для того, чтобы в дальнейшем с комфортом пользоваться этим инструментом [22].

Координаты кривой описываются в зависимости от параметра t∈[0,1].

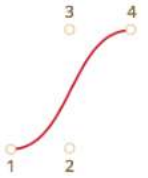
Для двух точек $P = (1 - t)P_1 + t\,P_2$



Для трех точек $P = (1 - t)^2 P_1 + 2(1 - t)tP_2 + t^2 P_3$



Для четырех точек $P = (1 - t)^3 P_1 + 3(1 - t)^2 tP_2 + 3(1 - t)t^2 P_3 + t^3 P_4$



Нужно отметить, что

1) Точки не всегда лежат на кривой.

2) Степень кривой равна числу точек минус один. Для двух точек это линейная кривая (т.е. прямая), для трёх точек квадратическая кривая (парабола), для четырёх кубическая.

3) Кривая всегда находится внутри выпуклой оболочки, образованной опорными точками:

4) Благодаря последнему свойству в компьютерной графике можно оптимизировать проверку пересечений двух кривых. Если их выпуклые оболочки не пересекаются, то и кривые тоже не пересекутся.

Метод де Кастельжо идентичен математическому определению кривой и наглядно показывает, как она строится. Метод заключается в том, что любую кривую Безье любого порядка можно очень простым способом разбить на две кривые того же порядка, и они будут в точности совпадать с исходной кривой. Посмотрим его на примере трёх точек.

1) Рисуем опорные точки.

2) Строятся отрезки между опорными точками.

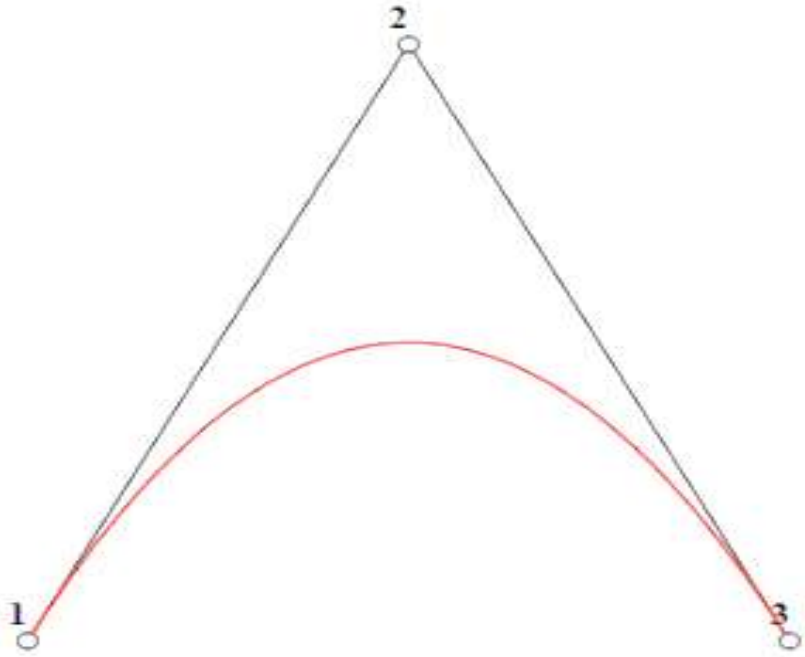


Рисунок 5.7 – Кривая, построенная по методу де Кастельжо

1) Параметр t пробегает значения от 0 до 1 с некоторым шагом;

2) На каждом из этих отрезков берется точка, находящаяся от начала на расстоянии от 0 до t пропорционально длине.

Так как черных отрезков – два. То и точек выходит две. Таким образом, при t = 0 точки будут в начале, при t = 0,25 на расстоянии 25 % от начала отрезка, при t = 0,5 – 50 %, при t = 1 – конец отрезка.

1) Эти точки соединяются. На рисунке 5.8 соединяющий отрезок показан синим цветом.

2) На получившимся отрезке берется точка на расстоянии t [20].

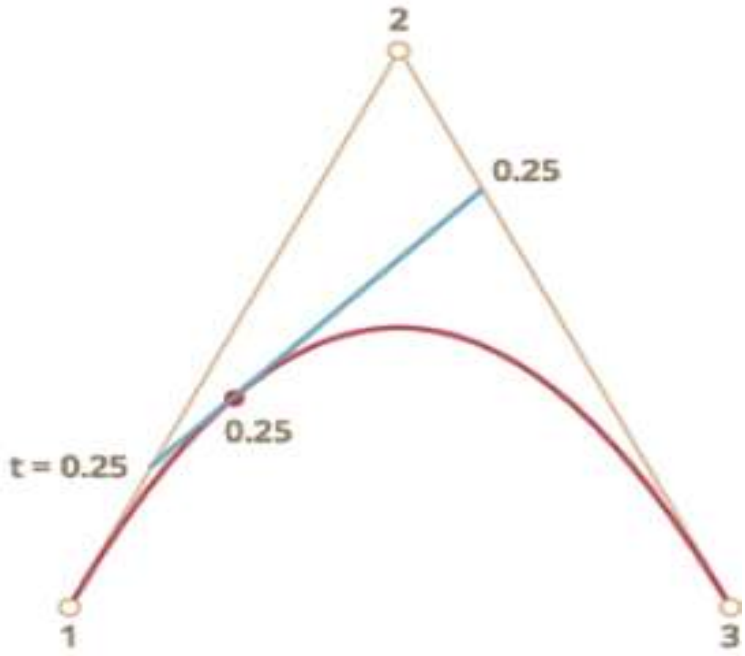


Рисунок 5.8 - Кривая, построенная по методу де Кастельжо при t = 0,25

Аналогичным образом могут быть построены кривые Безье и более высокого порядка (рисунок 5.9).

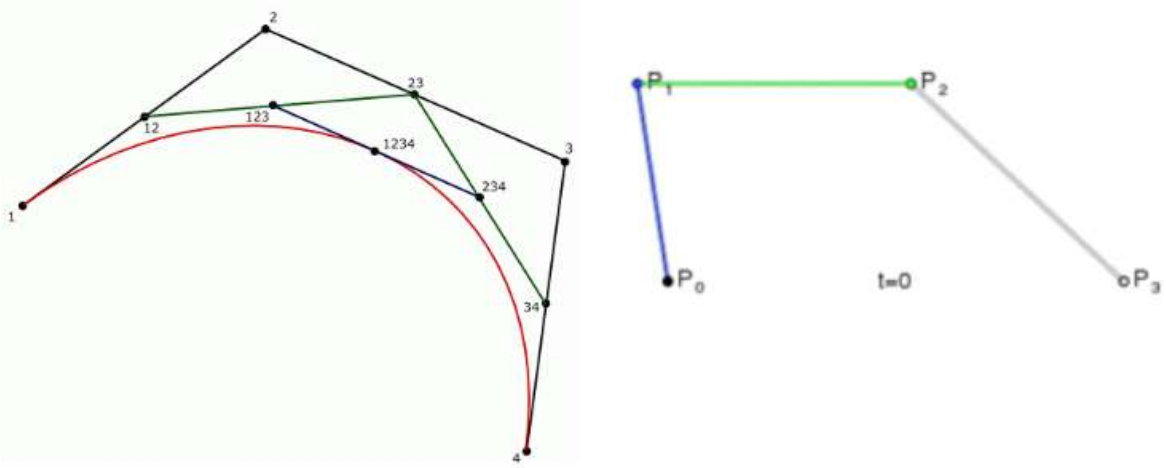


Рисунок 5.9 – Кривые Безье высокого порядка, построенные по методу де Кастельжо

Кривую Безье можно построить для любого числа точек. Но вычислять полиномы десятитысячной степени довольно сложно. Для решения этой

проблемы необходимо разбивать точки на группы по 4 штуки и уже для каждой из них строят кривую Безье и соединяют полученные сегменты в одну кривую. Единственная трудность – полученная кривая не достаточно «гладкая» на границах сегментов [23] (рисунок 5.10).

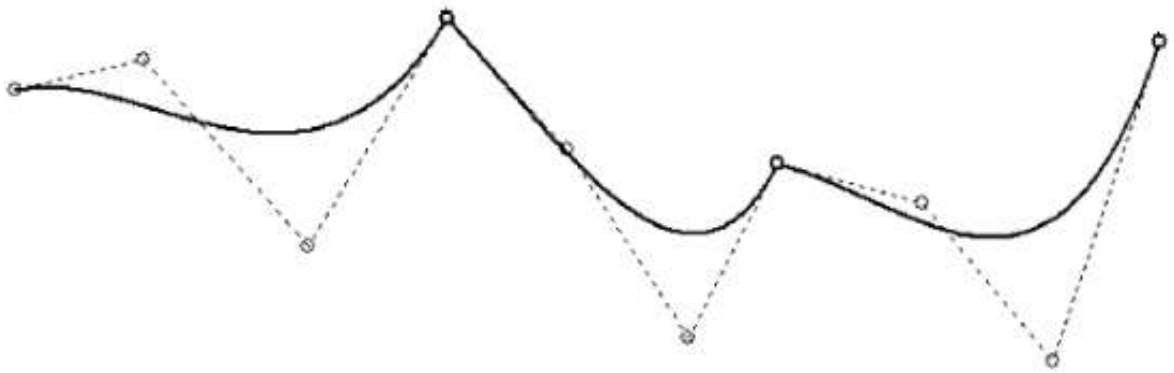


Рисунок 5.10 – Кривая Безье, построенная по принципу соединения кривых малого порядка

Для интерполяции, как правило, в компьютерной графике используют кубические кривые, соединенные гладким образом. Вместе они выглядят как одна кривая – называется принцип **интерполяция сплайнами.** Для того, чтобы составная кривая была геометрически непрерывной, необходимо каждые три точки в месте соединения лежали на одной прямой. Составную кривую можно построить из наборов элементарных кривых Безье для четверок вершин.

Допустим, у нас есть шесть точек x_1, x_2, x_3, x_4, x_5, x_6. Пусть (x_3; y_3) и (x_4; y_4) - координаты третьей и четвертой точки соответственно. Вставляем между ними дополнительную точку с координатами $x'=(x_3+x_4)/2$ и $y'=(y_3+y_4)/2$, после чего проводим одну кривую через точки x_1, x_2, x_3, x' а вторую — через точки x', x_4 , x_5 , x_6 . В результате получим одну гладкую кривую для шести точек.

Если учесть поведение кривой Безье и то, что точки (x_3; y_3), (x’; y’) и (x_4; y_4) лежат на одной прямой, это становится вполне очевидно [23].

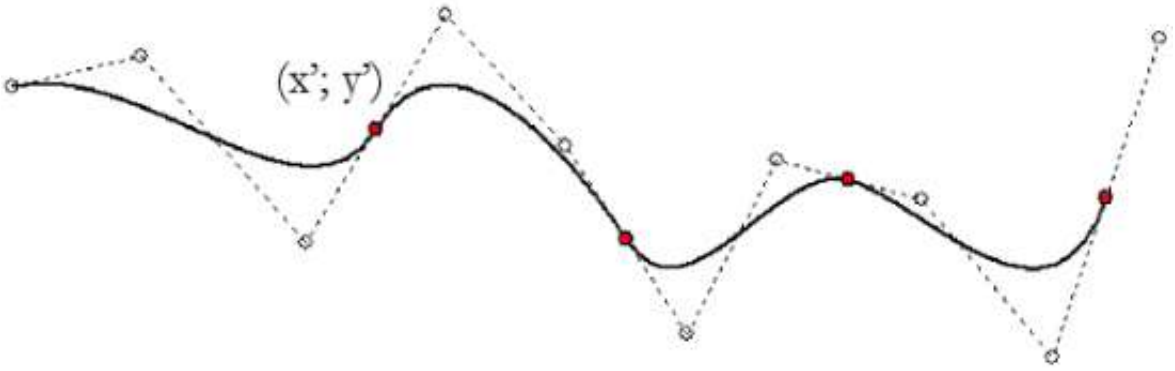


Рисунок 5.11 – Плавная кривая Безье

Если точек больше шести, их нужно разбить по такой же схеме и связать полученные группы с помощью дополнительных точек, как описано выше. Последнюю точку можно повторить несколько раз, если множество точек не делится на целое число групп, чтобы кривая доходила до последней точки.

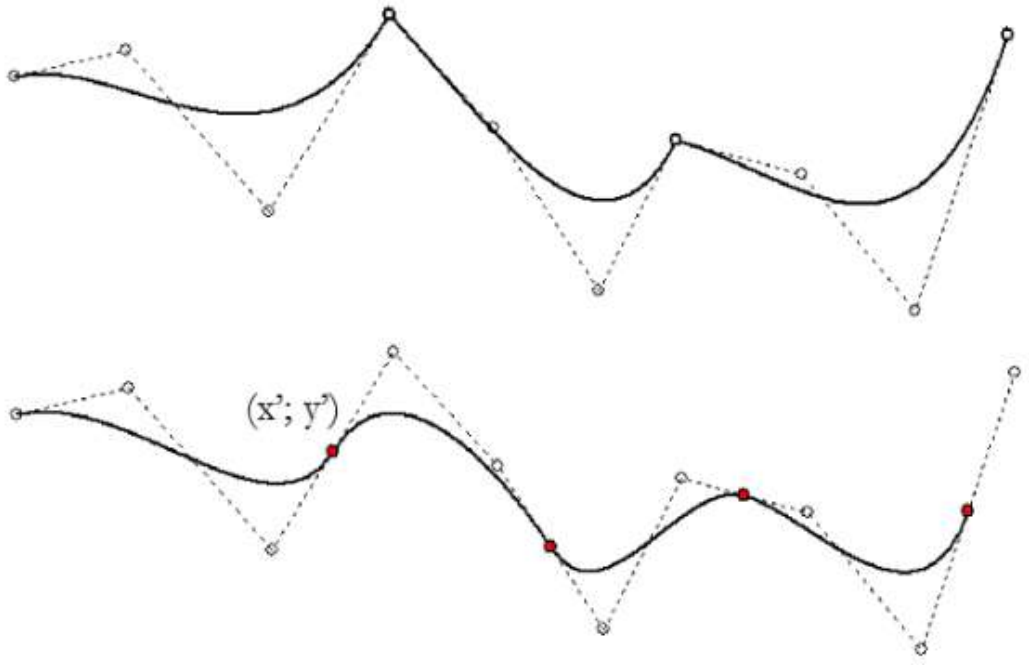


Рисунок 5.12 – Аппроксимация кривой Безье

5.3.4 Интерполяционная кривая Catmull - Rom

По заданному массиву точек P_0, P_1, P_2, P_3 **сплайновая кривая Catmull-Rom** определяется при помощи уравнения, имеющего следующий вид:

$$R(t)= 1/2(-t(1 - t)^2P_0 + (2 – 5t^2 + 3t^3)P_1 + t(1 + 4t + 3t^2)P_2 – t^2(1 - t)P_3), \qquad (5.5)$$

$$0 \le t \le 1.$$

Свойства составной сплайновая кривой Catmull-Rom:

- проходит точно через опорные точки;
- является геометрически непрерывной;
- набор базовых функций однозначно определяет кривую, т.е. нет возможности регулировать ее форму [24].

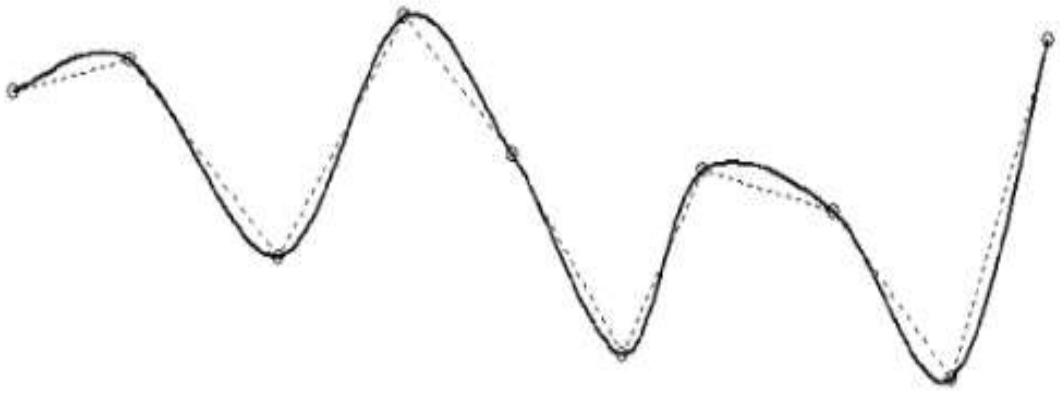


Рисунок 5.13 – Интерполяционная кривая Catmull-Rom

Для построения составной кривой нужно дополнить набор копиями первой и последней точек. Построить замкнутую интерполяционную сплайновую кривую можно, дополнив набор базовых точек из n штук точками $P_{n+1} = P_0$, $P_{n+2} = P_1$, $P_{n+3} = P_2$.

Для того, чтобы кривую плавной, необходимо каждый сегмент кривой строится на основе четырех точек. Построение кривой осуществляется

только между двумя внутренними точками каждой четверки. Четверки выбираются с перекрытием, то есть первой точкой очередной четверки выбирается вторая точка предыдущей четверки. Например, сегмент 1 – строится на основе точек 0, 1, 2, 3, а сегмент 2 – на основе точек 1, 2, 3, 4. Концевые точки дублируются.

5.4 Векторная графика на рассеянных кривых

Группа ученых из франции под руководством Александрины Орзан из Технологического института в Гренобле в 2008 году создали инструменты генерации векторной графики на рассеянных кривых, в том числе программу для автоматического конвертирования растровых изображений в векторные.

Рассеянные кривые - это обычные кривые Безье, однако, они хранят данные о цвете в основных точках, между которыми градиент меняется с помощью многосеточного метода.

Рассеянные кривые обеспечивают в небольшом объёме сведений закодировать относительно сложное и большое изображение. Именно это и требуется во время конвертирования растра в вектор.

Автоматический конвертер, разработанный Орзан с коллегами из института, позволяет генерировать результат, порой крайне близкий к оригиналу. К тому же возможна существенная экономия в размере файла, при сохранении всех преимуществ векторной графики. Минусом всего выше описанного является отсутствие рассеянных кривых в стандарте SVG. Именно поэтому алгоритм возможно реализовать исключительно в отдельном формате файлов. Результаты автоматической векторизации представлены на рисунках 5.14 и 5.15.

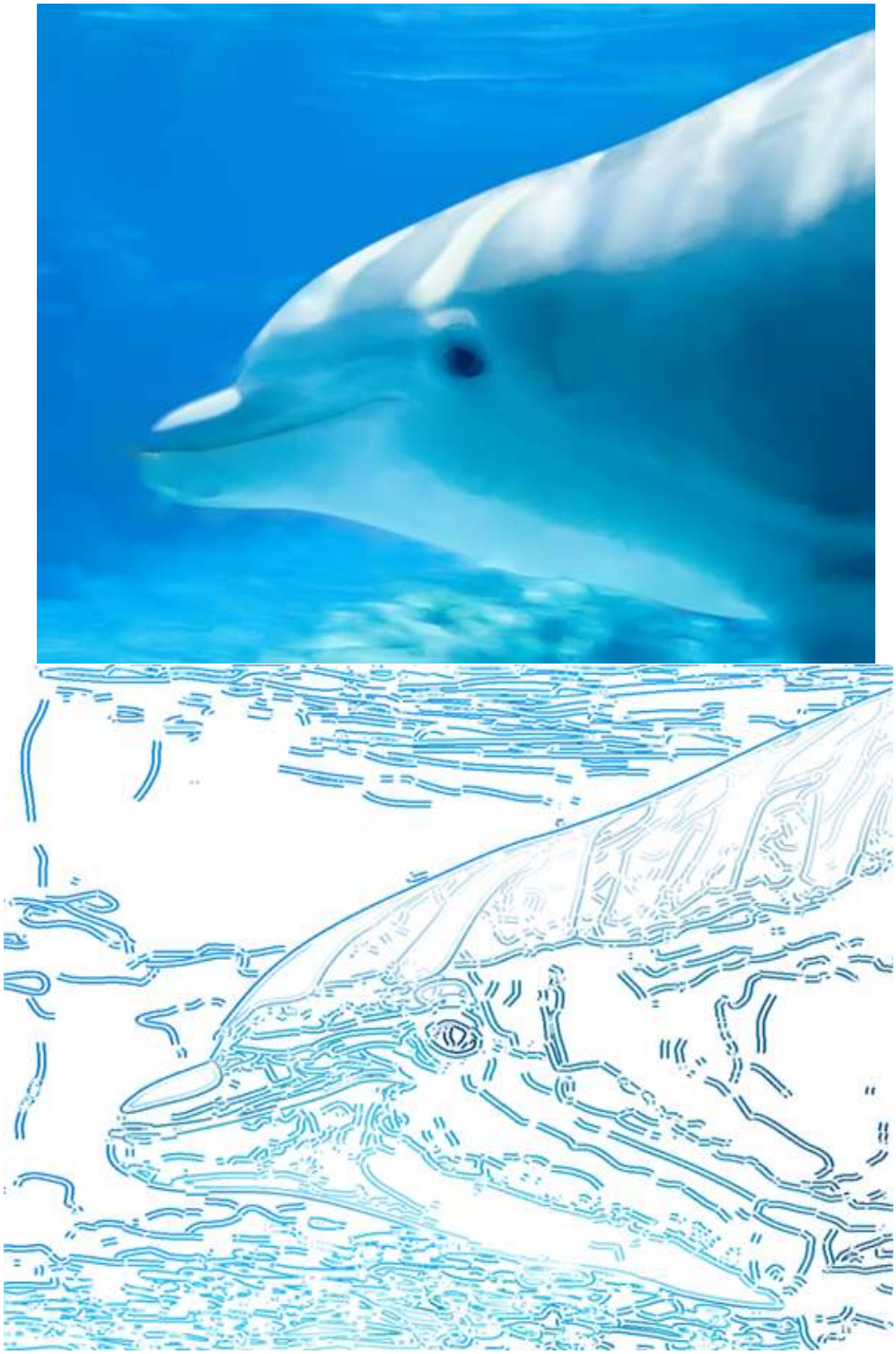


Рисунок 5.14 - Результат автоматической векторизации

Рисунок 5.15 - Результат автоматической векторизации

5.5 Алгоритмы заливки плоских фигур

В качестве основных примитивов векторной графики выделяют линию и образованный с ее помощью контур (открытый или замкнутый). Параметры обводки контура определяют его вид при отображении, а именно:

- толщина линии обводки;

- цвет линии;

- тип линии (сплошная, пунктирная и прочее);

- форма концов (со стрелкой, закругленные и тому подобное).

Для замкнутых контуров, в свою очередь, имеет место особое свойство: заливка, то есть параметры заполнения охватываемой контуром области.

Выделяют некоторые виды заливок:

- заливка одним цветом, то есть заполнение внутренней области определенным цветом;

- текстурная заливка - заполнение узором с регулярной структурой;

- заливка изображением-картой (узором);

- заполнение готовым растровым рисунком, иначе говоря, картой;

- градиентная заливка - заполнение двумя цветами с плавным переходом между ними.

Для всех из перечисленных типов заливок разработаны свои алгоритмы.

5.5.1 Заливка основным цветом

Для того, чтобы реализовать заливку графического объекта одним основным цветом, требуется определить непосредственно область заливки. Разработаны несколько приемов задания областей, ограниченных контурами:

- имеется группа точек расположенных как внутри контура, так и за его пределами, имеющих различные значения кода цвета или яркости внутри и вне заданного контура. Такое положение называется внутренне-определенной областью;

- существуют некоторые «затравочные» точки, в соответствии со свойствами которых производится закраска всей области, что получило название - гранично-определенные области;

- области, заданные списком вершин, скажем для закраски гранично-определенных областей существует несколько алгоритмов, к примеру рекурсивный метод.

Рекурсивный метод относительно легок для программирования, тем не менее считается низкоэффективным. От затравочной точки просматривают все точки. Если они не граничные, их закрашивают и код соответствующего цвета записывают в стековую память. Далее производится считывание данных из стека, и если очередная точка не является граничной и закрашенной, ей присваивают тот же код.

5.5.2 Текстурная заливка

Еще одним из видов заливки графических объектов с замкнутым контуром являются текстурные заливки. Данный вид заливки заполняет внутреннее пространство объекта единым рисунком, повторяющим текстуру таких поверхностей, как металл, облака, дерево, цемент и тому подобное, с использованием специально созданных цветных растровых рисунков. Теоретическая база текстурных заливок включает широкий спектр математических моделей, у каждой из которых имеется неповторимый набор управляющих параметров. Данные модели распределены по нескольким библиотекам, в которых хранятся несколько десятков описаний текстур и у каждой есть имя, описывающее ее внешний вид. Основным параметром всех математических моделей, предназначенного для построения изображения, является номер текстуры [25].

5.5.3 Заливка изображением-узором (картой)

Заливка изображений картами представляет собой узоры заливки в виде набора заранее подготовленных однотипных прямоугольных ячеек заполненных уникальным рисунком. После применения такого типа заливки в итоге внутреннее пространство объекта заполняется совокупностью этих ячеек, что образует непрерывный рисунок. Простейший вариант заливки узором - это двухцветный узор, например черно-белый. Полноцветный узор относится к векторной заливке и соответствует полноцветному узору с использованием как линий, так и заливок. Посредством узорных заливок возможно создавать мозаичные фоны или повышать реалистичность создаваемых изображений (при использовании растровых заливок). Все

заливки состоят из небольших квадратных плиток, которые складываются в правильную мозаику без искажений. Стоит отметить, что заливка картой предоставляет возможности интерактивной манипуляции элементарными частями, что позволяет создавать новый, совершенно не похожий на исходный, узор.

5.5.4 Градиентная заливка

Градиент определяется плавным переходом от одного оттенка или цвета к другому оттенку или цвету. Градиентные заливки в отличие от однородных типов заливок представляют сложные цветовые переходы между различными цветами. Если в градиент добавить некоторое количество промежуточных цветов, то в результате можно получить многоцветные градиентные заливки. В этом случае градиентную заливку можно трактовать как карту или узор, формируемый постепенным переходом между двумя или несколькими цветами или оттенками одного итого же цвета. В зависимости от направления перехода цветов выделяют 4 вида градиентных заливок: линейная, радиальная, коническая и квадратная. Цветовая гамма градиента определяется группой опорных точек, которые фиксируются на шкале цветового перехода специальными маркерами: начальный маркер, конечный маркер цвета и промежуточные маркеры. Опорные точки градиента - это точки, в которых тот или иной цвет присутствует в количестве 100 % и не смешивается с другими цветами. Для настройки цветового рисунка градиента требуется сдвигать маркеры опорных точек относительно друг друга.

5.6 Экономическое приложение векторной графики

По результатам проведенных исследований, вплоть до начала 1990-х гг. доходы от применения векторной графики в научно–инженерных приложениях были значительно выше доходов в других, ненаучных областях, в частности в коммерческой сфере. Тем не менее, в 1991 году доходы были поделены уже пополам, а баланс устойчиво начал сдвигается в

сторону нетехнического применения векторной КГ. На сегодняшний момент порядка 2/3 всех доходов от КГ поступает именно из нетехнических областей практического приложения.

«Классическая» векторная графика и сегодня часто используется в различных отраслях бизнеса, включая разработку концепции, тестирование и создание новых продуктов, но бизнес также стал лидирующим потребителем систем мультимедиа, например, в обучении или маркетинговых презентациях.

Векторная графика не заменима и для составления визитных карточек и рекламных вывесок.

6 Преобразования плоскости

6.1 Поворот плоскости и его матричное преобразование

Преобразование плоскости – это взаимно-однозначное отображение этой плоскости на себя. Наиболее важными геометрическими преобразованиями являются движения, то есть преобразования, сохраняющие расстояние.

Поворот плоскости является одной из разновидностью ее преобразования. Преобразования плоскости относительно несложные. Если поворот происходит относительно начала координат, то для задания вращения используется один угол φ. Пример показан на рисунке 6.1.

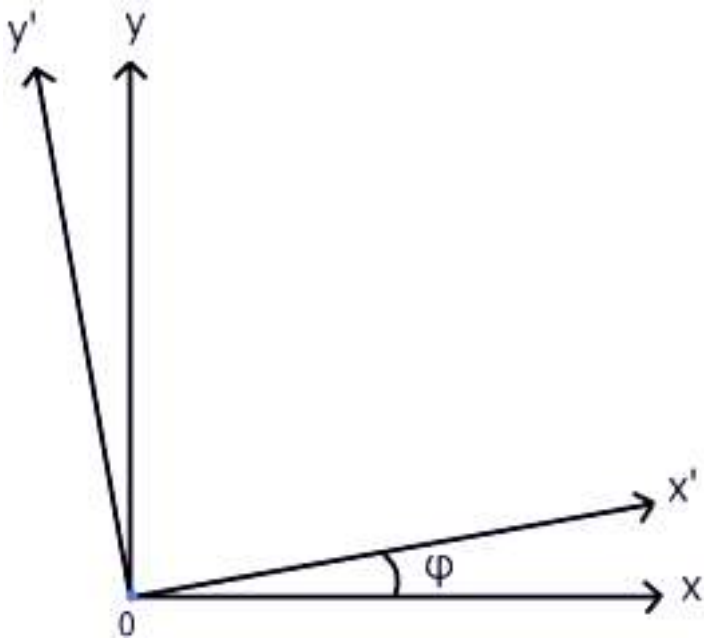


Рисунок 6.1 – Угол поворота плоскости относительно начала координат

Принято считать направление вращения против часовой стрелки положительным. При этом удобно считать, что угол φ находится в интервале $[-\pi; \pi]$.

Чтобы получить преобразование координат при повороте, возьмем произвольный вектор $\vec{r}$, задающий некоторую точку.

Его координаты (6.1), (6.2):

$$x = |r| \cos \alpha, \tag{6.1}$$

$$y = |r| \sin \alpha. \tag{6.2}$$

При повороте на угол φ:

$$x' = |r| \cos(\alpha + \varphi) = |r| (\cos(\alpha) \cos(\varphi) - \sin(\alpha) \sin(\varphi)) = x \cos \varphi - y \sin \varphi, \tag{6.3}$$

$$y' = |r| \sin(\alpha + \varphi) = |r| (\sin(\alpha) \cos(\varphi) + \cos(\alpha) \sin(\varphi)) = x \sin \varphi + y \cos \varphi. \tag{6.4}$$

Таким образом, при повороте на угол φ координаты x и y подвергаются преобразованию, написанному выше [26].

Написанное выше преобразование координат удобно представить в виде матрицы:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}, \tag{6.5}$$

Умножение матриц производится по принципу строка на столбец. Поэтому количество столбцов (элементов в строке) в матрице слева должно совпадать с количеством строк (элементов в столбце) в матрице справа.

Если умножить две матрицы, задающие повороты на углы α и β, то получится матрица поворота на угол α + β. Это легко проверить, перемножив

соответствующие матрицы и использовав формулы для косинуса и синуса суммы.

6.2 Поворот плоскости с помощью мыши

Для того, чтобы осуществить поворот плоскости с помощью мыши, сформулируем задачу:

Имеется пара точек $M_0(x_0, y_0)$, $M(x, y)$ – начальное и конечное положения мыши. По этим точкам нужно определить угол поворота плоскости. В случае формульного представления необходимо учитывать не только относительные изменения координат Δx, Δy, но и их абсолютные значения. На рисунке 6.2 приведены изменения по x для передвижений мыши M_0M и $M_0'M'$. Преобразования совпадают, но одно из этих передвижений вращает треугольник по часовой стрелке, а другое - против.

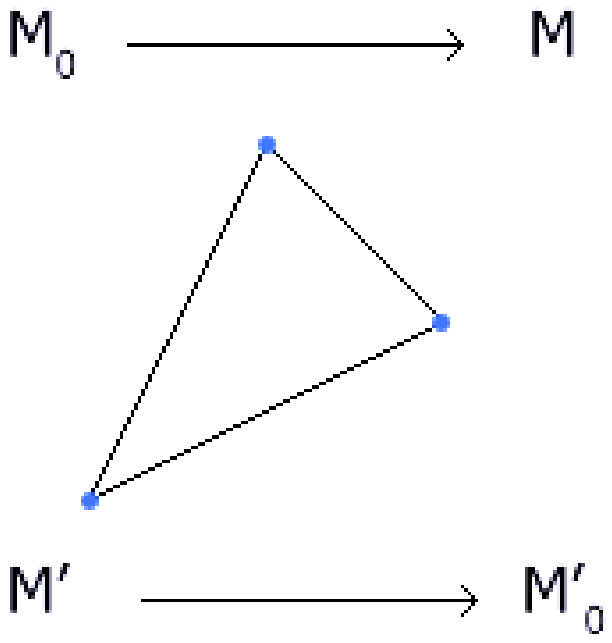


Рисунок 6.2 - Изменения по x для передвижений мыши M_0M и $M_0'M'$

Суть приема заключается в том, что строится воображаемая единичная окружность с центром в начале координат и на ней отмеряются начальный и конечный углы. Потом производится поворот на разницу **направленных углов** (рисунок 6.3)**.**

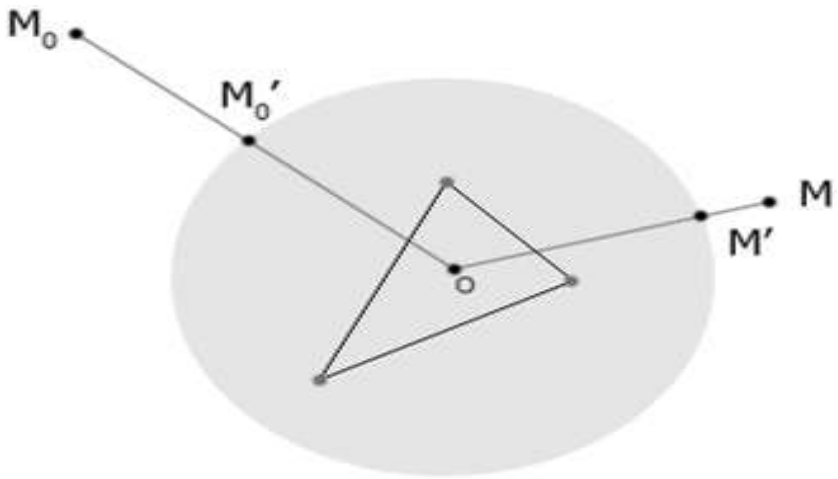


Рисунок 6.3 – Поворот плоскости

В данном случае, чтобы получить точки M_0’ и M’ надо просто нормализовать векторы OM_0 и OM. Теперь повернем плоскость, чтобы вектор OM_0’ перешел в OM'. Для расчета угла между векторами в декартовой системе координат, необходимо воспользоваться скалярным произведением [27].

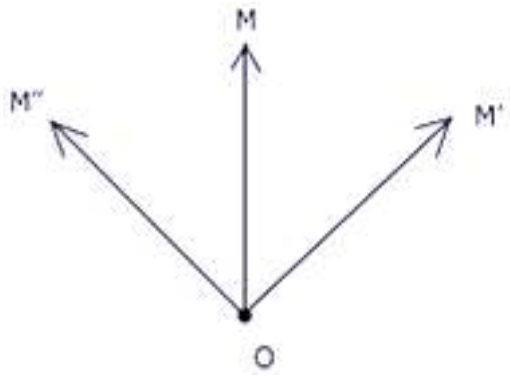


Рисунок 6.4 – Угол между векторами

Проблема заключается в том, что для скалярного произведения вектора ОМ" и ОМ ' будут неразличимы. Можно получить косинус угла между ними. Но если вращать плоскость от ОМ к ОМ', то вращение идет по часовой стрелке. В случае ОМ" - против часовой. Таким образом, кроме значения угла необходимо знать его знак. Для этого следует применить векторное произведение.

Так как, мы имеем дело с двухмерным пространством, то строго говорить о векторном произведении нельзя. Но можно использовать тот результат, что эта операция различает порядок векторов.

Рассмотрим трёхмерное пространство, причем наше двухмерное сечение это плоскость z = 0. Это означает, что все вектора в этой плоскости имеют третью координату равной 0. Векторно перемножим вектора ОМ(x, y) и ОМ'(x',y'):

$$\begin{pmatrix} x \\ y \\ 0 \end{pmatrix} \times \begin{pmatrix} x' \\ y' \\ 0 \end{pmatrix} = \begin{pmatrix} \vec{i} & \vec{j} & \vec{k} \\ x & y & 0 \\ x' & y' & 0 \end{pmatrix} = (xy' - yx')\vec{k}. \qquad (6.6)$$

Функция от векторов f(a, b) = (xy' – yx') будет равна синусу направленного угла от вектора a к вектору b, умноженному на длины векторов a и b (|a||b|sin(α)).

Учитывая тот факт, что вектора единичные (M_0' и M' лежат на единичной сфере), с помощью скалярного произведения и функции f(a, b) получаем косинус и синус нужного угла с учетом его направления. Остается подставить полученные значения в матрицу поворота и требуемый эффект будет получен.

6.3 Аффинное преобразование и его матричное представление

В предыдущих разделах учебного пособия была приведена математическая база для различных классов компьютерной графики, а также информация о геометрических преобразованиях в двумерном пространстве. Обобщая сказанное выше, рассмотрим особенности задания аффинного преобразования в матричной форме.

Преобразование плоскости называется **аффинным**, если

- оно взаимно однозначно;

- образом любой прямой является прямая.

Преобразование называется **взаимно однозначным**, если

- разные точки переходят в разные;

- в каждую точку переходит какая-то точка.

Если рассматривать параллельный перенос, то становится очевидным, что для его задания матрицы размера 2x2 уже недостаточно. Однако, его можно задавать с помощью матрицы размерности 3x3. Но, в таком случае, необходимо иметь третью координату у двумерной точки.

Однородные координаты - это особый тип координат, которые характеризуются свойством сохранения определяемого ими объекта в случае умножения всех координат на одно и то же число.

Однородными координатами вектора (x, y) является тройка чисел (x', y', h), где $x = x' / h$, $y = y'/h$, а h — некоторое вещественное число (случай, когда $h = 0$ является особым).

Данные координаты не позволяют однозначно задать точку плоскости. Например, $(1, 1, 1)$ и $(2, 2, 2)$ задают одну и ту же точку (1, 1). Предлагается взять набор (x, y, 1), который будет описывать все точки плоскости.

Матрица преобразования для однородных координат имеет размер 3x3 [28]. Рассмотрим некоторые преобразования в однородных координатах.

6.3.1 Сжатие/ растяжение

Это преобразование умножает соответствующие координаты точек на коэффициенты масштабирования по осям: (x, y) -> ($a_x \times x$, $a_y \times y$). Матрица преобразования запишется следующим образом:

$$\begin{pmatrix} a_x & 0 & 0 \\ 0 & a_y & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad (6.7)$$

где a_x – растяжение по оси x,

a_y – растяжение по оси y.

Преобразование представлено на рисунке 6.5.

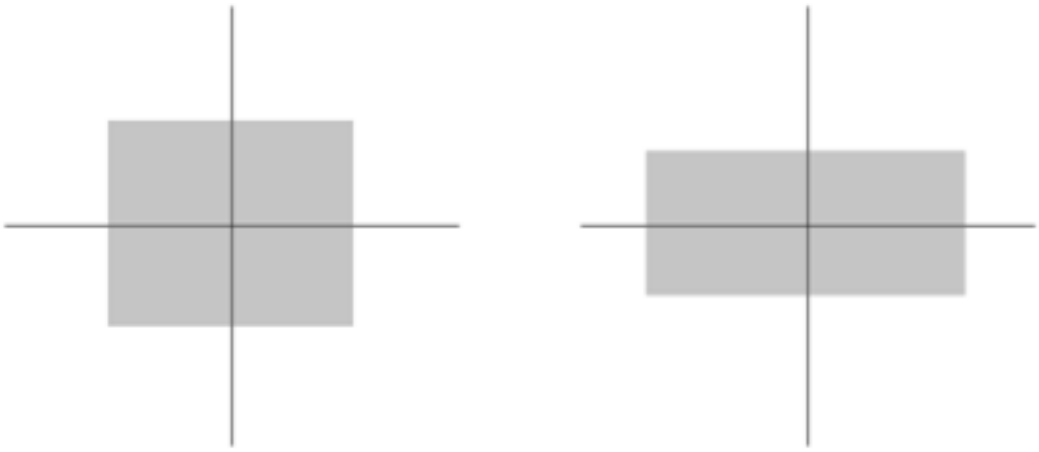


Рисунок 6.5 – Аффинное преобразование сжатие/ растяжение

При отрицательных значениях коэффициентов сжатия/растяжения происходит отражение относительно соответствующих осей. Этот случай можно включить в данное преобразование, а можно вынести в отдельное, сказав, что коэффициенты масштабирования принимают только положительные значения.

6.3.2 Поворот

Ранее разобранная матрица поворота 2x2 теперь дополняется строкой и столбцом:

$$\begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

При угле $\varphi = \pi$ эта матрица задает центральную симметрию относительно начала координат, которая является частным случаем поворота. Стоит отметить, что такую симметрию можно задать с помощью преобразования сжатия/растяжения (допуская отрицательные коэффициенты масштабирования).

7 Трёхмерная графика

Трёхмерная графика - раздел компьютерной графики, посвящённый методам создания изображений или видео путём моделирования объёмных объектов в трёхмерном пространстве.

3D-моделирование - это процесс создания трёхмерной модели объекта. Задача 3D-моделирования - разработать визуальный объёмный образ желаемого объекта.

С помощью трёхмерной графики можно создать точную копию конкретного предмета или разработать новый. Созданная модель может соответствовать объектам окружающей действительности, например люди, дома, ног также могут быть полной абстракцией реального мира (проекция четырёхмерного фрактала).

Первые компьютерные программы, формирующие простые трехмерные модели на основе эскизов, были созданы в 1960-х годах в университете города Юты (США) Иваном Сазерлендом.

Трёхмерные объекты широко применяются в здравоохранении (компьютерная томография), экономике (визуализация котировок, выручек и прочее) картографии, полиграфии и многом другом. Телевидение и другие отрасли индустрии развлечений применяют анимационные инструменты КГ (компьютерные игры, фильмы). Распространенной практикой является также использование компьютерного моделирования при обучении представителей различных профессий – тренажеры (например, пилотов). Знание основ компьютерной графики сейчас необходимо как инженеру, так и ученому.

7.1 История развития трёхмерной графики

Одним из основателей 3D графики считают Ивана Сазерленда, который, написал программу **Sketchpad**, позволявшую создавать простые

трёхмерные объекты. В последующем открывает в университете города Юты первую кафедру компьютерной графики. Его студенты:

Эд Катмулл, ныне технический директор корпорации Pixar, впервые смоделировал объект. В качестве предмета для моделирования выступила кисть его собственной руки.

Джим Блинн - создатель **bump mapping** и **environment mapping**, первых компьютерных анимаций для NASA и знаменитого материала **blinn.**

Би Тюн Фонгом - разработчик техники **Phong shading** (модель расчёта освещения трёхмерных объектов).

Анри Гюр – создатель принципа **Gouraud shading** (алгоритм затенения).

Изначально компьютерная графика и анимация использовалась преимущественно в рекламе и на телевидении. К примеру, компьютерной компании **MAGI** принадлежит заслуга в создании первой в истории коммерческой компьютерной анимации: вращающийся логотип IBM появился в начале 70-х годов.

Система моделирования, разработанная компанией MAGI являлась процедурной – модели создавались путем комбинирования 25 геометрических фигур, имевшихся в библиотеке программы. Из простейших фигур, вроде пирамиды, сферы и цилиндра создавались более сложные, которые впоследствии становились основой для конечной 3D-модели. Программа Synthavision разрабатывалась в течение пяти лет и была использована при создании знаменитого киношедевра «Трон» (1982).

Компания **Triple–I** была открыта в 1962 году и внесла огромный вклад в развитие 3D технологий. В 1975 году руководство компании открывает отделение компьютерной графики и анимации. В отличие от компании MAGI, использовавшей геометрические фигуры, Triple-I задействовала в качестве простейших единиц треугольники и квадраты. Такой метод моделирования получил название «полигонального». Компания Triple-I также принимала участие в работе над фильмом «Трон».

Первые персональные компьютеры отличались малой мощностью, что немало препятствовало работе с 3D графикой.

С распространением 3D технологий и их внедрением в развлекательную индустрию наибольшего успеха добиваются графические станции SGI серии **IRIS 4D**, которые оснащаются мощными системами визуализации Onyx, способными уместить до 64 процессоров. Графические станции оснащаются 64 битными микропроцессорами **MIPS**. Этими машинами оснащаются крупнейшие голливудские 3D студии: **ILM** и **Digital Domain**. Графические станции SGI обладали большой производительностью и невероятно высокой стоимостью. Одним словом, рабочие станции SGI были ориентированы на небольшую целевую аудиторию, состоящую из одних профессионалов.

Сегодняшний рынок 3D пакетов отличается большим разнообразием. А изначально не было комплексных программных решений, которые бы соединяли в себе функции по моделированию, анимации и рендерингу. Одним из первых комплексных пакетов для моделирования и рендеринга был **Lightwave 3D**, выпущенный в 1990 году для компьютеров серии **Amiga.**

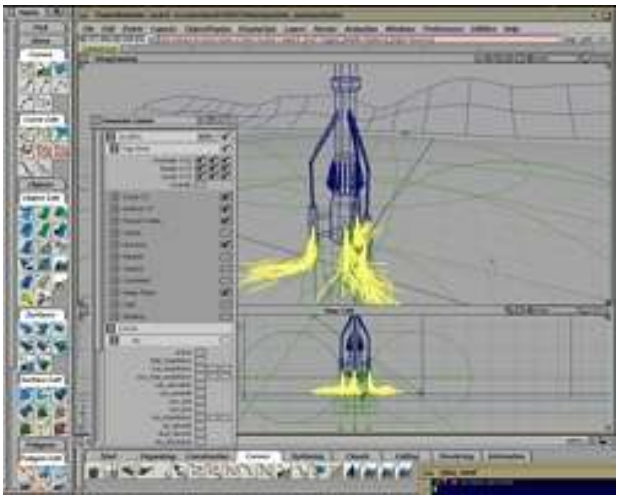


Рисунок 7.1 – Экранная форма Lightwave 3D

Специалисты компании соединили два программных пакета: **Aegis Modeler** (моделирование) и **Videoscape** (рендеринг и анимация). Изначально Lightwave 3D был приложением программного комплекса **Video Toaster**. Полноценная и независимая версия была выпущена в 1994 году.

PowerAnimator является одним из предков Maya, программы, с которой знаком, хотя бы по названию, любой современный тридэшник. PowerAnimator представлял собою дорогостоящий программный комплекс, работавший на графических станциях SGI под операционной системой **Irix**. PowerAnimator впервые был использован на проекте «Бездна»**.** Программа Maya 1.0 появилась в феврале 1998 году и соединила в себе достоинства трех следующих программных пакетов: The **Advanced Visualizer** (визуализация), **Thomson Digital Image** (моделирование) и Power Animator (анимация). Объединение **Alias** и **Wavefront** в 1995 году стало возможным благодаря покупке обеих компаний концерном SGI. После выхода программного пакета **Maya**, руководство приняло решение прекратить выпуск PowerAnimator и сконцентрировать свои силы на производстве и раскрутке нового бренда. Раскрутка удалась на славу – на сегодняшний день Maya является наиболее востребованным программным решением в сфере развлекательной индустрии. За восемь лет компания Alias|Wavefront выпустила семь версий своего ведущего программного решения.

3Ds Max компании Autodesk появился в 1996 году, и являлась развитием программы **3D Studio для DOS**. Новая разработка отличалась интерактивным 32-разрядным интерфейсом, программа лишилась характерных для досовских версий ограничений, кроме того, максимально расширились возможности пакета.

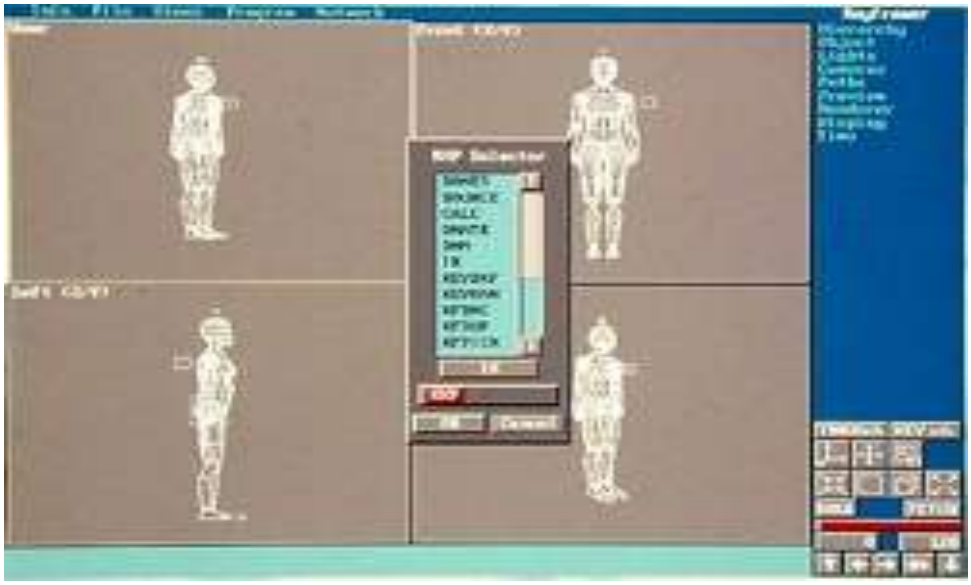


Рисунок 7.2 – Экранная форма Alias

В последние несколько лет устойчивыми лидерами в области моделирования и анимации являются всем известные коммерческие продукты, такие как **Maya**, **Lightwave**, **3dMAX**, **XSI** и **Houdini**. Тем не менее, уверенно набирают популярность и открытые продукты, распространяемые бесплатно, например, полнофункциональный пакет **Blender3D** (позволяет производить модели с последующим рендерингом) и **Wings3D**. Производители прекрасно понимают, что совместимость с максимально большим количеством программ и приложений способствует росту продаж, поэтому появляющиеся программные решения не имеют проблем с интеграцией.

Следует отметить также усилившуюся тенденцию к выпуску узконаправленных программ, вроде **Renderman** и **Mental Ray** в качестве плагинов для универсальных 3D пакетов. В любом случае место под солнцем найдется для всех, потому что мир окончательно и бесповоротно попал в 3D сеть технологического прогресса.

7.2 Общие сведения о 3D графике

Плоская 2D-графика была востребована вплоть до начала 90-х гг., пока не стали создавать 3D картинки. В основу трёхмерной графики легли две оси координат - X и Y – широта и высота. Для создания спецэффектов в кинофильмах и компьютерных играх, стали применять еще одну ось – Z (глубина). Позже объемную графику стали использовать и при разработке мультфильмов. Развитие сети Интернет только способствовало стремительному охвату большого количества областей, в том числе веб-ресурсы.

В настоящее время для формирования трёхмерного рисунка достаточно лишь грамотно владеть одним из программных продуктов разработки 3D-графики. Благодаря развитию программного обеспечения самостоятельно изготовить видеоролик или какую-нибудь объемную картинку может даже новичок в этом деле. Выбирая нужную программу, стоит обратить внимание на следующие моменты:

- наличие не сложных для понимания и высокопроизводительных по конечному результату средств моделирования трёхмерных объектов. Простота инструмента должна заключаться в интуитивно-понятном интерфейсе и поле для работы даже новичку;

- выход последней версии программы. Чем современнее пакет моделирования, тем шире функционал;

- наличие различных приложений. Во многих программах заложено огромное множество приложений, что предоставляет возможность разнообразить процесс создания модели и добавить только нужные эффекты. В некоторых пакетах можно, кроме того, создавать собственные приложения – и это также является положительным качеством.

Несмотря на все плюсы трёхмерных объектов, они обладают и рядом существенных недостатков:

- высокие требования к аппаратной составляющей компьютера: к его оперативной памяти, быстроте работы процессора и так далее;

- необходимость больших временных затрат на создание моделей всех объектов сцены, могущих оказаться в поле зрения камеры. Конечно, такая работа стократно окупается результатом;

- меньшую свободу в создании изображения, чем в двухмерной графике. Создавая объект карандашом на бумаге или средствами 2D-графике на экране, можно совершенно свободно искажать пропорции объектов, пренебрегать законами перспективы и пр. В 3D-формате это возможно только в наиболее мощных пакетах, но даже в них это требует дополнительных усилий и изобретательности;

- необходимость постоянно отслеживать взаимное расположение объектов в пространстве. Например, при создании 3D-анимации. Так как объекты 3D-графики «бестелесны», они легко проникают друг в друга и важно контролировать отсутствие ненужного контакта между ними. С этой же причиной связана необходимость использования приемов для деформации объектов при их столкновении между собой или разрешении. Если упустить этот момент, то, например, два сталкивающихся между собой персонажа просто пройдут друг сквозь друга.

7.3 Моделирование трёхмерных объектов

7.3.1 Общая постановка 3D моделирования

3D моделирование - это процесс создания 3D модели.

В настоящем разделе пособия мы опишем традиционные методы моделирования трёхмерных объектов, не беря во внимание пакеты скульптурной трёхмерной графики. Наиболее распространенный технический прием - это создание модели из примитива, то есть на основе элементарной геометрической фигуры, плоской или объёмной, которая после некоторого количества преобразований приобретает нужные формы.

Порой 3D модель создается из нескольких примитивов сразу. Не смотря на это, в случаи текстурирования необходимо , чтобы у трёхмерной модели не было невидимых, «внутренних» граней (или даже их фрагментов), а при стыковке нескольких примитивов подобное встречается достаточно часто.

Чтобы избежать описанной проблемы применяют булевы операции. Операция объединения позволяет составить из двух соприкасающихся или пересекающихся объектов один, чья поверхность состоит из суммы поверхностей исходных объектов, за вычетом тех областей, где происходит пересечение.

Трансформация во время 3D моделирования формируется в основном за счет экструдирования отдельных частей- вершин, рёбер и/или граней, разделение, при котором ребро или грань делится на несколько одинаковых частей, и перемещение и вращение отдельных элементов. Имеет место такой прием как «разрезание» одной или нескольких граней (или одного или нескольких рёбер) в произвольных местах.

Форму исходного примитива следует выбирать, основываясь на представлениях о конечном облике планируемой модели. Более того, при моделировании нужно использовать NURBS и/или кривые и поверхности Безье. Проявлением профессионализма в трёхмерном моделировании является применение кривых и направляющих.

Например, создание из одной или нескольких кривых контура, вдоль которого потом выстраиваются другие геометрические фигуры (шланг или изогнутый ствол дерева удобнее всего моделировать из множества окружностей, «нанизанных» на направляющую кривую).

Использование кривых и сплайнов позволяет добиться особой гладкости 3D модели, минимизируя заметность полигонов.

Очень полезно также при 3D моделировании использование всевозможных средств дубликации. Например, формирование винтовой

лестницы. Вручную её собирать сложно, тогда как различные 3D-пакеты предоставляют средства автоматизации процесса.

Традиционно рисуют в 2D (по осям X и Y). При этом отображают какую-то одну из сторон предмета. Но если мы хотим получить представление обо всех сторонах предмета, то необходимо нарисовать несколько рисунков. Так поступают в традиционной рисованной анимации. Но, вместе с тем, существует, кукольная анимация. Один раз изготовленную куклу снимают в необходимых позах и ракурсах, получая серию «плоских картинок». 3D (к X и Y добавляется координата глубины Z) визуализация - это те же «куклы», только существующие в цифровом виде. Другими словами, в специальных программах (Blender, 3ds Max, Maya, Cinema 4D и т.п.) создается объемное изображение, например авто (рисунок 7.3).

Рисунок 7.3 – 3D модель автомобиля

Ключевым достоинством этого способа является возможность свободного помещения модели в кадр и анимирования специалистом, а

графическое представление авто в финальной картинке полностью возлагается на программу визуализатор. Кроме этого, модель требуется изобразить 1 раз, чтобы в последующем использовать в других проектах. Для плоского рисунка такая ситуация невозможна. Также данный метод позволяет разрабатывать бесконечно детализированные объекты, например, смоделировать даже винтики на часах и прочие мелкие детали.

На сегодняшний момент разработаны множество способов трёхмерного моделирования. Однако, самым распространенным и часто встречающимся на практике является полигональное моделирование. Достаточно часто можно встретить в роликах о 3D или фантастических фильмах как тот или иной объект представляется в виде так называемой сетки (рисунок 7.3), что и есть пример полигонального моделирования. Суть такого метода заключается в том, что поверхности представляются в виде простых геометрических двумерных примитивов. В компьютерных играх это треугольники, для других целей обычно используют четырехугольники и фигуры с большим количеством углов. Такие примитивы и называют полигонами. Но при формировании трёхмерного объекта лучше будет использовать четырехугольники. Если требуется четырехугольные полигоны без труда трансформируются в треугольники при экспорте в игровой движок, а при необходимости сглаживания или тесселяции модель из четырехугольников получается, чаще всего, без артефактов.

Если какой-то объект формируется в виде полигонов (особенно дело касается органических объектов), то очевидна закономерность: чем меньше размер полигонов, чем их больше, тем более близкой может быть модель к реальному представлению.

Суть способа тесселяции состоит в первоначальном создании грубой фигуры из малого количества полигоновоперация тесселяции. Для тисселяции требуется разделить полигон на четыре части. В случае, если применяются четырехугольные полигоны, то алгоритмы тесселяции дают более качественный и предсказуемый результат.

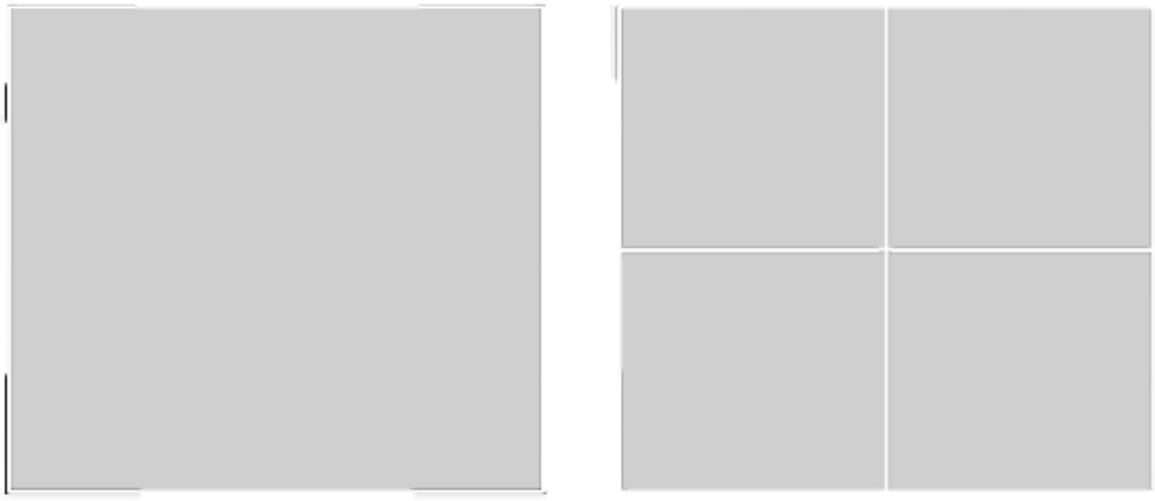


Рисунок 7.4 – Полигон и тисселяция

Рисунок 7.5 – Сглаживание

Тем не менее, чем больше используется полигонов, тем больше точек по которым они строятся, а значит больше информации приходится

обрабатывать процессору. В этой связи трёхмерная графика - это всегда баланс между подробной и тщательной детализацией модели и производительностью. Именно поэтому появились понятия высоко полигональной модели и низко полигональной модели. В играх применяются низко полигональные модели, так как в них выполняется отображение изображения в режиме реального времен.

Как правило, полигональное моделирование относится к пустотелому моделированию, где объект имеет только объем внешней оболочки. По сути это говорит о том, что если смоделируем объект, а потом уберем одну из стенок, то увидим внутри пустоту. Существуют программные средства и для твёрдотельного моделирования. В таких программах (например, Autodesk Inventor) используют математические модели отличные от тех, что в полигональном моделировании. Алгоритмы твёрдотельного моделирования целесообразнее применять для моделирования механизмов при разработке техники. Программы наподобие Autodesk Inventor имеют инструменты для моделирования, учитывая при этом особенности имитируемого технологического процесса. Готовые модели можно сразу отправить на подходящий станок для получения изделия в том материале, в котором они должны быть.

Кроме того, существуют программы трёхмерной лепки (ZBrush, Autodesk Mudbox), в которых моделирование сводится фактически к созданию углублений или выпуклостей. Такая техника похожа на то, как скульпторы лепят из глины - убирая ненужное и добавляя необходимое. С помощью таких программ можно добиться реалистичного рельефа поверхности, например морщин на коже или складок ткани. Сегодня высокополигональные реалистичные модели животного мира реализуются, как правило, с применение программы лепки. Распространена практика когда заготовка модели создается с помощью полигонального моделирования, а затем в программе лепки тесселируется и добавляются мелкие детали.

Рисунок 7.5 – Модель, созданная с помощью 3D лепки

Рисунок 7.6 – Модель без текстур и текстурированная модель танка

В общем понимании текстура представляет собой двумерный рисунок, накладываемый трёхмерную модель. Текстура бывает процедурной, то есть

алгоритмически сгенерированной, или же просто отрисованная в графическом редакторе, кроме того, может быть фото реального объекта. На основе текстуры задается рисунок и цвет модели. Однако, все реальные поверхности обладают большим количеством параметров: отражающей способностью, преломлением, рельефом, позрачностью и так далее. Все названные характеристики определяются в свойствах материала. Таким образом, любая поверхность или материал для трёхмерной графики представляется как математическая модель описывающая ее параметры. Для такого объекта как вода - обязательно требуется указать прозрачность и преломляющую, отражающую способности.

Перед помещением поверхности в модель нужно сформировать ее развертку. Это означает представление всех поверхностей в виде проекции на плоскость. Все это полезно для верного наложения поверхности.

Таким образом разработка трёхмерной модели чаще всего состоит из 5 основных стадий:

1) получение изображений референса (то есть того, с чего будет моделироваться) или самого референса или отрисовка экскиза;

2) моделирование геометрии на основе референса;

3) создание развертки;

4) отрисовка текстур или получение их другим способом в виде файлов;

5) настройка параметров материала (текстуры, преломление, отражение, прозрачность).

1-й и 4-й шаги могут быть опущены в случаи простой модели. Однако, качественной модели не получить, если не выполнить все 5 шагов.

2D изображение, чаще всего, формируется в 2 стадии: эскиз и раскрашивание. В 3D графике после изготовления модели ее необходимо поместить в сцену к другим объектам, добавить освещение, камеру и лишь затем можно надеяться получить финальную картинку. Изображение в 3D графике просчитывается на основе физической модели, как правило, это

модель распространения луча света с учетом отражения, преломления, рассеивания и тому подобное. Рисуя красками мы сами обозначаем тени, блики и так далее, а в трёхмерной графике мы подготавливаем сцену с учетом освещения, материалов, геометрии, свойств камеры, программа самостоятельно рассчитывает итоговое изображение.

7.3.2 Алгоритмы моделирования

На сегодня все алгоритмы можно разделить на четыре категории.

1) **Сплайновое моделирование** (термин «сплайн» означает кривые, которые бывают разных типов)

NURBS – поверхности, определяемые кривыми, на которые влияют «тяжелые» контрольные точки. Кривая следует по точкам (но не обязательно прикасается к ним). Увеличение веса точки привлечет кривую ближе к ней. NURBS являются на самом деле гладкими поверхностями, а не их имитациями с помощью маленьких плоских поверхностей, поэтому этот метод часто применяют для моделирования органических форм. Часто термин NURBS используется для обозначения всех методов сплайнов моделирования:

- патчи и кривые Безье - примитивный тип NURBS;
- би-сплайны (англ. Bi-spline) - это специальный тип сплайнов, которые могут быть быстро вычислены, как сумма базовых функций;
- Rational;
- Non-uniform (неравномерные) - дает возможность неравномерной параметризации вдоль поверхности;

2) **Полигональное моделирование**

Точки в 3D-пространстве, вершины, соединенные между собой линией - ребром, образуют поверхность по законам создания геометрических плоскостей. Набор объединенных плоскостей называют полигональной сеткой. Большая часть 3D-моделей сегодня строится как текстурированные многоугольные модели, поскольку они достаточно гибкие и компьютер может отрендерить их достаточно быстро. Однако, многоугольники являются

плоскими и могут только примерно передать изогнутые поверхности, используя много многоугольников.

3) **Моделирование с помощью сабдивов** - один из современных алгоритмов, который прогрессивно развивается и все более увеличивает конкуренцию двум предыдущим.

4) **Процедурное моделирование**

Такое моделирование позволяет оперировать масштабными проектами, требует в большинстве пайплайн, поэтому используется крупными студиями компьютерной графики.

Современное программное обеспечение позволяет использовать независимо от алгоритма моделирования различные подходы для построения модели [1].

Все современные программные продукты, предназначенные для оценки финансово-экономического состояния компаний и анализа инвестиционных проектов, требует предварительного сбора и обработки большого объема данных, необходимых в качестве исходной информации для работы компьютерной модели. В результате расчетов с помощью всех компьютерных аналитических систем для подготовки бизнес-плана инвестиционного проекта входная числовая информация преобразуется заданным образом в выходные форматы в виде расчетных таблиц, содержащих информацию о реализующем проект предприятии отчеты о прибылях и убытках, движении денежных средств, прогнозные балансы, таблицы финансовых коэффициентов, критериальных показателей непосредственно проекта и разнообразных графиков. Дальнейшая работа с такой выходной информацией позволяет аналитику проекта итеративным путем сделать проект наиболее привлекательным, улучшить условия его реализации и так далее.

7.4 Особенности 3D анимации

3D анимация - это автоматизация перемещения и трансформаций 3D модели в пространстве с течением времени.

В большинстве случаев на практике используется 3 способа анимирования трёхмерных объектов:

1) перемещение и вращение целого объекта, без изменения его формы;

2) динамические деформации (наиболее подходящим примером является бьющееся сердце или порхание крыльев бабочки);

3) скелетная анимация - самый трудоемкий вид анимации и используется чаще всего для анимации персонажей.

Кроме того, преимущественно применяют 3 метода анимации трёхмерных объектов: анимация по ключевым кадрам, анимация по кривым движения и анимация по траекториям.

Первый метод в своей основе содержит работу стандартных аниматоров, когда художник рисует основные позы персонажа, а позировщики заполняют промежуточные кадры, отрисовывая надлежащие изменения фигуры. В случае автоматизации функции позировщика выполняет система, то есть конкретные алгоритмы в программе-редакторе. Аниматору достаточно лишь обозначить несколько основных положений фигуры, интерполяция осуществится автоматически.

Следовательно, алгоритм заключается в следующем: выбираем первый ключевой кадр (А), фиксируем в нём первую позу объекта; затем выбираем новый ключевой кадр (Б), трансформируем объект (перемещаем, вращаем, изменяем размеры), фиксируем. Автоматизированная система после вычисляет промежуточный процесс - траекторию движения и вращения между кадрами А в Б, а также - изменения размеров (но не формы!), отображая эти изменения в виде кривых движения.Таким образом, обозначается четыре ключевых кадра, тогда как всего их в ролике - 260.

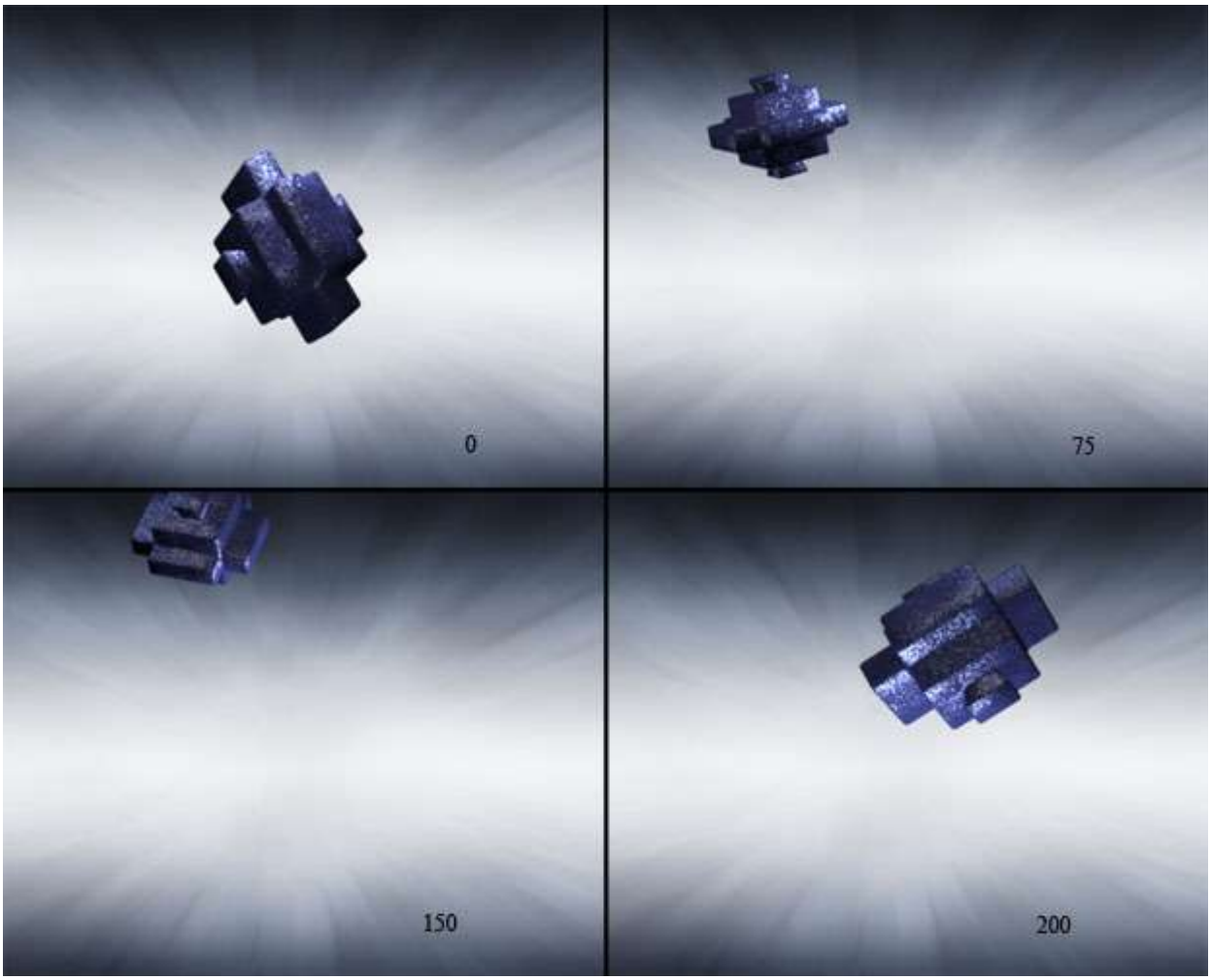


Рисунок 7.7 – Формирование анимации на основе четырех ключевых кадров

Кривые движения - это, собственно, представление перемещения или трансформации объекта в виде графиков для каждой из его координат XYZ.

Кривые движения (Loc), вращения (Rot) и изменения размеров (Scale), представленные на одном графике.

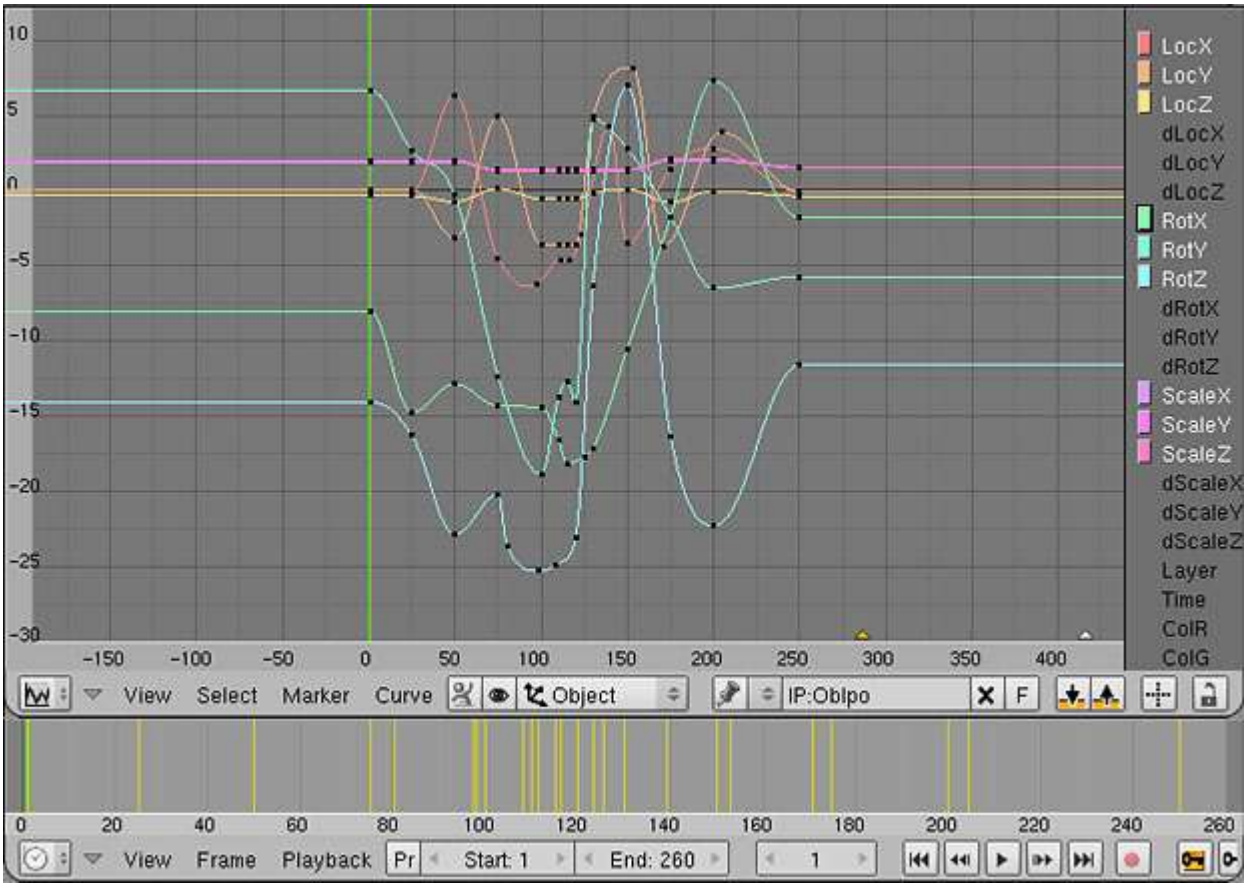


Рисунок 7.8 – Экранная форма рабочего окна программы Blender 3D

Кривые весьма удобно применять в случае, когда необходим точный контроль знаний каждого из параметров. Управление параметрами на сегодняшний день давно автоматизировано посредством программных пакетов. Однако, без регулярного учёта большого количества разнообразных характеристик, качественной трёхмерной анимации сделать не получится.

В одних пакетах - к примеру, Blender - функции работы с основными кадрами и кривыми объединены. Но имеют место и другие программные средства, в которых помимо кривых отдельно существует ещё и временная таблица, где все изменяемые при анимации параметры отображаются как события на временной оси.

Траектории - это отдельно определяемый путь передвижения объекта. Обязательно необходимо указать направление движения. Траекториями задаются скорости перемещения и возможные изменения ориентации

объекта в пространстве, которая регулируется обычно вышеупомянутыми кривыми.

Описанные методы и способы чаще всего применяются в комбинациях, особенно, когда речь идёт об относительно сложной, а главное реалистичной анимации [22].

7.5 Векторы в пространстве. Однородные координаты. Матрицы преобразований

Вектор в трёхмерном пространстве определяется тремя координатами p(x, y, z). При рассмотрении двумерных преобразований были введены однородные координаты. Удобно их будет ввести и при рассмотрении трёхмерного пространства. Их использование позволяет задавать аффинные и нелинейные преобразования (например перспективное) в виде матриц, что позволяет создавать весьма эффективные и простые реализации.

Однородные координаты - координаты, обладающие тем свойством, что определяемый ими объект не меняется при умножении всех координат на одно и то же число.

Однородными координатами вектора (x, y, z) является тройка чисел (x', y', z', w), где x = x' / w, y = y' / w, z = z' / w, а w - некоторое вещественное число (случай, когда w = 0 является особым).

Данные координаты не позволяют однозначно задать точку пространства. Например,(1, 1, 1, 1) и (2, 2, 2, 2) задают одну и ту же точку (1, 1, 1). При переходе к однородным координатам для точки с координатами (x, y, z)предлагается взять набор (x, y, z, 1). В процессе преобразований координата w может меняться. Обратный переход к декартовым координатам осуществляется посредством деления на w-координату.

Матрица преобразования для однородных координат имеет размер 4x4.

$$M = \begin{pmatrix} 2 & 0 & 0 & 1 \\ 0 & 3 & 0 & 2 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad (7.1)$$

x' = 2 * x + 1

y' = 3 * y + 2

z' = 1 * z – 1

w' = 1

Данная матрица определяет следующее преобразование: растяжение по оси x в два раза, по оси y в три раза и перенос на вектор (1, 2, -1). При этом матрица M применяется к **вектору-столбцу** слева:

$$p' = M \times p = \begin{pmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{pmatrix}, \quad (7.2)$$

Заметим, что вектор p' также представляет собой вектор-столбец. На самом деле возможны две, равноправные с точки зрения математики, записи:

1) Вектор записывается как столбец:

$$\begin{pmatrix} 2 & 0 & 0 & 1 \\ 0 & 3 & 0 & 2 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} 2x + 1 \\ 3y + 2 \\ 1z - 1 \\ 1 \end{pmatrix}$$

2) Вектор записывается как строка, матрица транспонируется:

$$(x\ y\ z\ 1) \times \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 2 & -1 & 0 \end{pmatrix} = (2x + 1 \quad 3y + 2 \quad 1z - 1 \quad 1)$$

С точки зрения самого преобразования эти записи эквивалентны. Какую из них использовать – это вопрос договоренности. На практике, и в OpenGL и в DirectX используется первый вариант, с учетом того,

что матрицы укладываются в памяти по столбцам. Т.е. матрица в линейной памяти будет выглядеть следующим образом:

$M_{11}\ M_{21}\ M_{31}\ M_{41}\ M_{12}\ M_{22}\ M_{32}\ M_{42}\ M_{13}\ M_{23}\ M_{33}\ M_{43}\ M_{14}\ M_{24}\ M_{34}M_{44}$

В трёхмерной математике также принято использовать первый вариант записи [20].

Матрица не аффинного преобразования

Примером может служить простейшая матрица перспективного преобразования.

$$M_{proj} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix}, \tag{7.3}$$

$$p_{proj} = M_{proj} \times p = M_{proj} \times \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ -z \end{pmatrix}, \tag{7.4}$$

$$p_{proj} = \left(\frac{-x}{z}, \frac{-y}{z}, -1, 1\right), \tag{7.5}$$

Последняя строка как раз и определяет вид изменений w-координаты исходного вектора.

7.6 Аффинные преобразования пространства

Как и при обработке плоскостей, объекты трёхмерного пространства также требуют различного рода модификаций, а именно движение, повороты, сжатие и растяжение, скосы и тому подобное. Однако, как правило, возникает необходимость сохранения после преобразований определенных свойств:

- аффинное преобразование трёхмерного пространства отображает n-мерный объект в n-мерный, фактически точку в точку, линию в линию, а поверхность в поверхность;

- после бесконечного множества преобразований объект сохраняет параллельность линий и плоскостей;

- аффинные преобразования позволяют оставлять неизменными пропорции параллельных объектов – длин отрезков на параллельных прямых и площадей на параллельных плоскостях, что требуется при работе с объектами реального мира.

Любое аффинное преобразование задается матрицей размера 3x3 с ненулевым определителем и вектором переноса:

$$\overrightarrow{p'} = R\vec{p} + \vec{t}\,, \tag{7.6}$$

$$\begin{vmatrix} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \mathrm{R}_{11} & \mathrm{R}_{12} & \mathrm{R}_{13} \\ \mathrm{R}_{21} & \mathrm{R}_{22} & \mathrm{R}_{23} \\ \mathrm{R}_{31} & \mathrm{R}_{32} & \mathrm{R}_{33} \end{pmatrix} \times \begin{pmatrix} \mathrm{x} \\ \mathrm{y} \\ \mathrm{z} \end{pmatrix}. \end{vmatrix} \tag{7.7}$$

R представляет собой матрицу линейного оператора над пространством трёхмерных векторов. Вектор T требуется для осуществления параллельного переноса: если помножить (0 0 0) на любую матрицу 3x3, опять получим (0 0 0) – начало системы координат, относительно преобразования R, является неподвижно точкой. Требование, чтобы определитель был ненулевой, диктуется определением. То есть, если определитель матрицы R равен 0, то и рассматриваемое пространство переходит в плоскость, прямую или точку. Этим и объяснятеся не соблюдение взаимной однозначности.

На практике бывает проще задавать аффинное преобразование одной матрицей 4x4. В этом случае используются однородные координаты:

$$\begin{pmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (7.8)$$

Обратим внимание на первые 3 значения последней строки - они равны нулю. Это необходимое условие того, что преобразование будет аффинным. В общем случае произвольная матрица размера 4x4 определяет проективное преобразование. Данный вид преобразования применяется для проецирования трёхмерной сцены.

Ниже приведем наиболее распространенные частные случаи аффинных преобразований. Для этого будет использоваться система координат, введенная следующим образом:

- система координат правая;
- ось z направлена на наблюдателя, перпендикулярно плоскости экрана;
- ось y находится в плоскости экрана и направлена вверх;
- ось x находится в плоскости экрана и направлена вправо.

Параллельный перенос

Матрица параллельного переноса выглядит представлена ниже:

$$\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (7.9)$$

В данном случае матрица R = E, единичной матрице.

Преобразования, рассматриваемые ниже, затрагивают только матрицу R, поэтому будет указываться только она.

Поворот (вращение)

В случаях поворотов плоскости - преобразование осуществляется вокруг некоторой точки. В трёхмерном же пространстве аффинные повороты реализуются вокруг произвольного вектора. Прежде чем перейти к

построению матрицы поворота вокруг некоторого вектора, необходимо привести частные случаи поворотов вокруг координатных осей.

Поворот вокруг произвольного вектора не равно повороту вокруг произвольной направленной прямой.

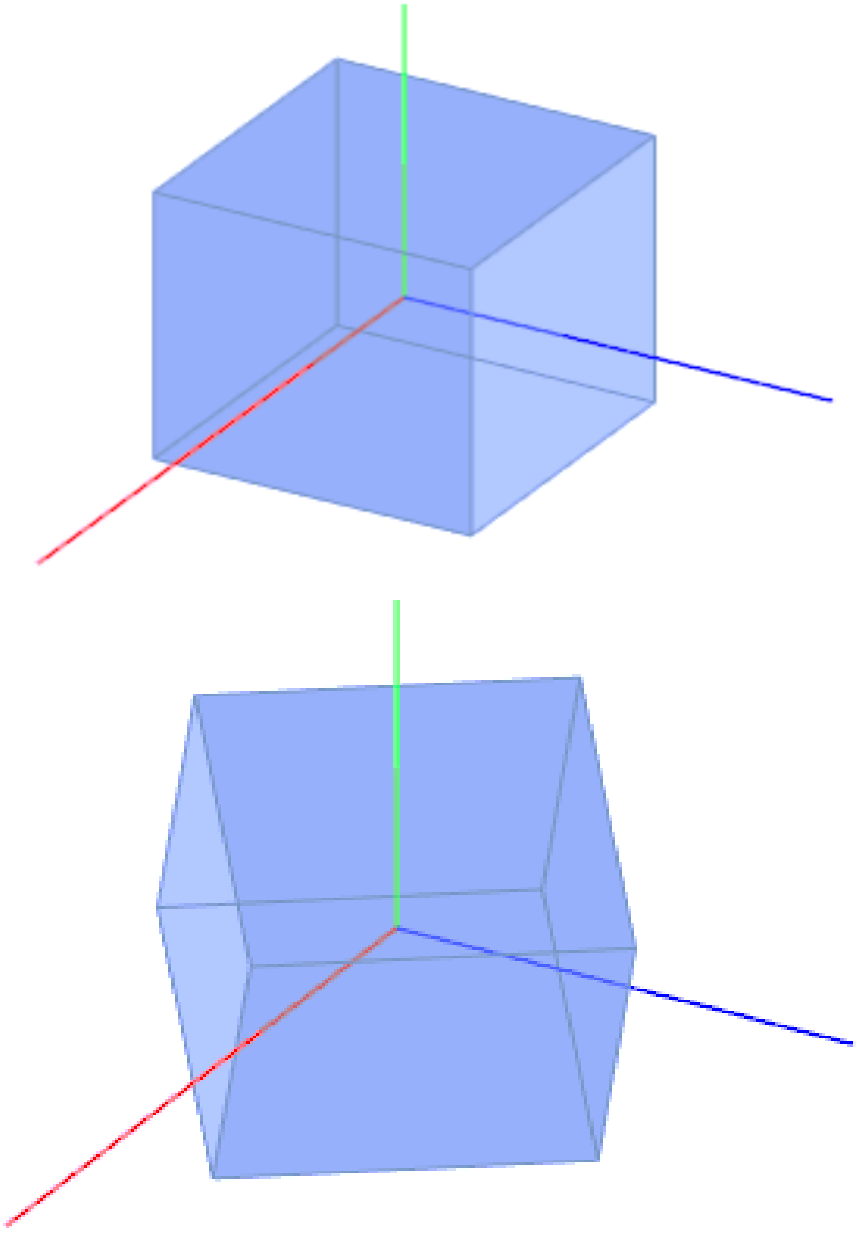


Рисунок 7.9 – Вращение

Поворот вокруг оси y

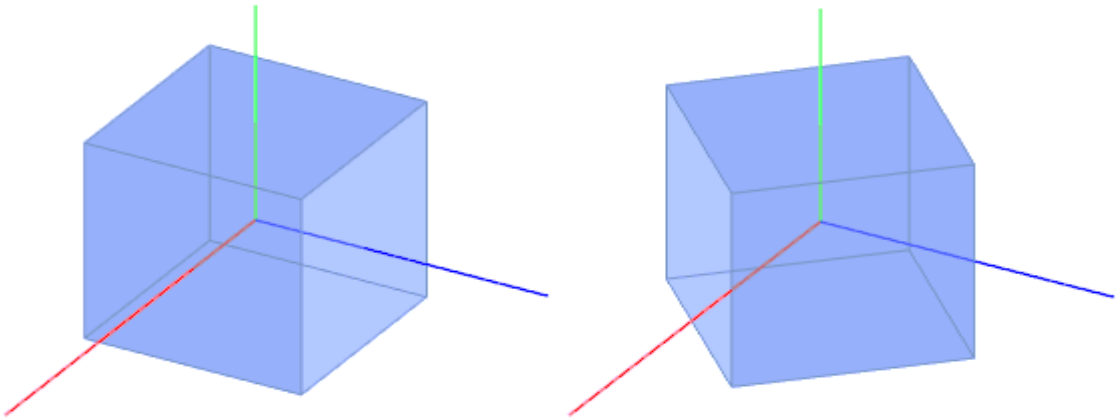


Рисунок 7.10 - Поворот вокруг оси y

Поворачивая y-координаты вокруг оси y - они не меняются. Кроме того, также стоит отметить, что координаты x и z точки преобразуются независимо от ординаты точек, то есть любая точка $p(x, y, z)$ перейдет в точку $p'(x'(x, z), y, z'(x, y))$. Координаты x и z в свою очередь преобразуются следующим образом: в плоскости Oxz это будет поворот вокруг начала координат по часовой стрелке (так как x z y - левая тройка), а именно в отрицательном направлении. Матрица данного аффинного преобразования приведена ниже:

$$\begin{pmatrix} \cos -\varphi_y & -\sin -\varphi_y \\ \sin -\varphi_y & \cos -\varphi_y \end{pmatrix} \tag{7.10}$$

В итоге:

$$x' = x\cos(\varphi_y) + z\sin(\varphi_y)$$

$$y' = y$$

$$z' = -x\sin(\varphi_y) + z\cos(\varphi_y)$$

Матрица преобразования $R_y(\varphi_y)$:

$$\begin{pmatrix} \cos(-\varphi_y) & 0 & -\sin(-\varphi_y) \\ 0 & 1 & 0 \\ \sin(-\varphi_y) & 0 & \cos(-\varphi_y) \end{pmatrix} \quad (7.11)$$

Поворот вокруг осей x и z

Аналогичными преобразованиями можно получить матрицы поворотов $R_x(\varphi_x)$ и $R_z(\varphi_z)$ вокруг осей x и z, соответственно. В формулах (7.12) и (7.13) приведены окончательные матрицы.

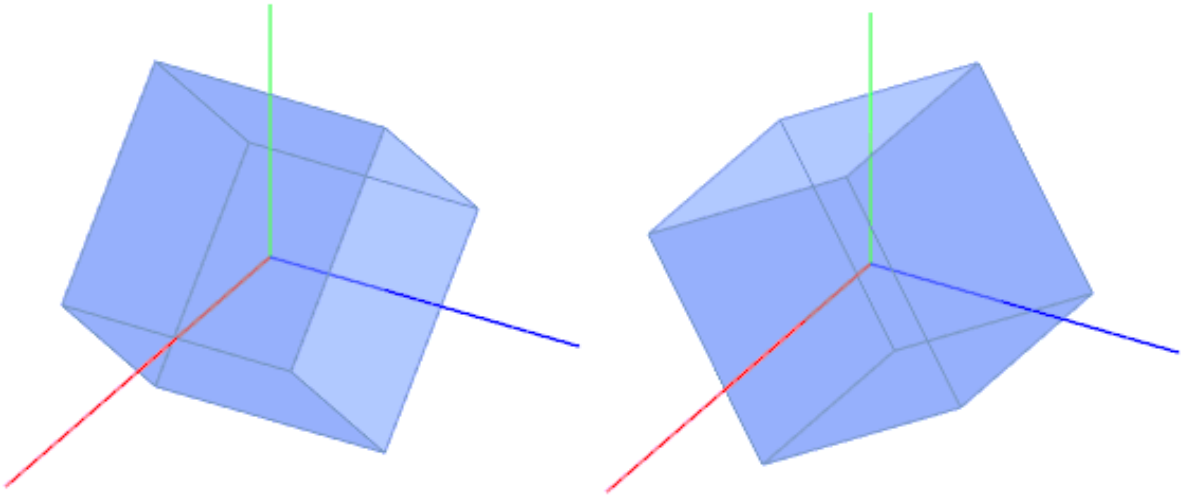


Рисунок 7.11 - Поворот вокруг осей x и z

$$R_x(\varphi_x) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi_y) & -\sin(-\varphi_y) \\ 0 & \sin(-\varphi_y) & \cos(\varphi_y) \end{pmatrix} \quad (7.12)$$

$$R_z(\varphi_z) = \begin{pmatrix} \cos(\varphi_z) & -\sin(\varphi_z) & 0 \\ \sin(\varphi_z) & \cos(\varphi_z) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (7.13)$$

Определители матриц R_x, R_y, R_z равны 1. Также матрицы вращений R_{rot} обладают свойством ортогональности: $R^TR = RR^T = E$. Следовательно, если выше названные условия выполняются, то обращение матрицы поворота можно заменить транспонированием: $R^{-1}(\varphi) = R^T(\varphi)$.

Масштабирование (сжатие/растяжение, отражение)

Коэффициенты сжатия/растяжения, по аналогии с двухмерным пространством, определяются диагональными членами матрицы R:

$$\begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{pmatrix} \quad (7.14)$$

В результате получаем:

$x' = s_x x$

$y' = s_y y$

$z' = s_z z$

Комбинация коэффициентов $s_x = -1$, $s_y = 1$, $s_z = 1$ будет задавать отражение от плоскости Oyz ($x = 0$). При $s_x = s_y = s_z = -1$ получим центральную симметрию относительно начала координат.

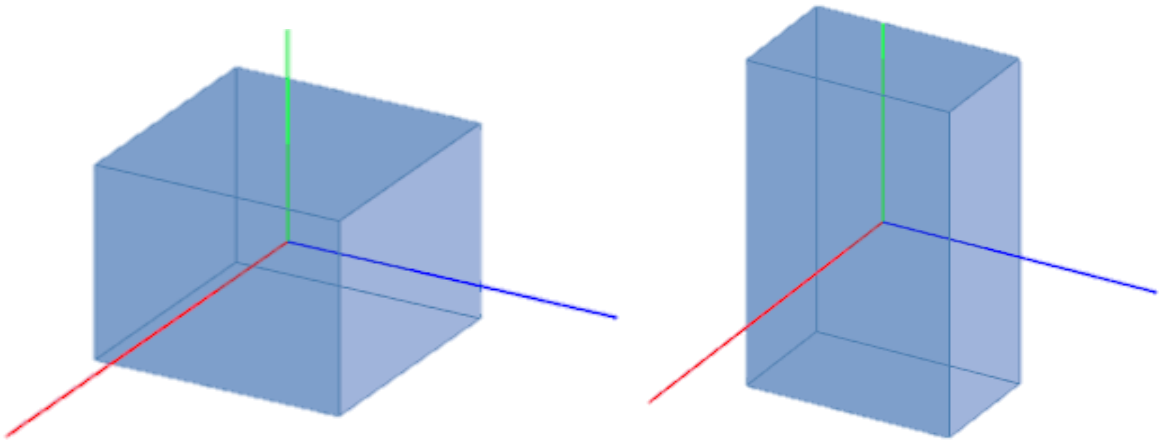


Рисунок 7.12 - Масштабирование (сжатие/растяжение, отражение)

Интерпретация матрицы P

Рассмотрим, что представляет собой матрица P с точки зрения линейной алгебры. Оказывается, что матрица R содержит базис новой системы координат.

Матрица $\begin{pmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{pmatrix}$ переводит вектора декартова базиса:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{pmatrix}$$

Скос

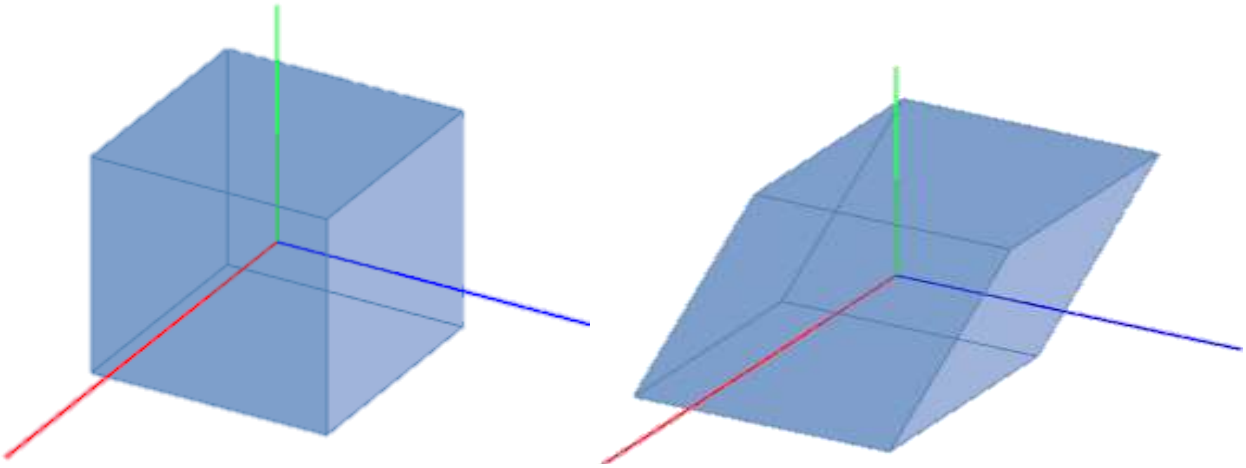


Рисунок 7.13 – Скос

Теперь несложно получить преобразование скоса. Например:

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

В рамках общепринятой терминологической базы математически описанное аффинное преобразование называют сдвигом. При этом, сдвигом является любое преобразование, если в результирующей матрице R главная диагональ единичная. Однако, если определитель матрицы R равен 0, то преобразование не является аффинным [5].

Сложные аффинные преобразования

Сложные аффинные преобразования формируются путем комбинации простых (элементарных) выше приведенных преобразований. Выбор простых преобразований для комбинации можно осуществлять различными способами в зависимости от требований к конечному результат и фантазии разработчика.

Скажем, поворот можно представить как комбинацию масштабирования и сдвига. Однако, для удобства, поворот также считается элементарным преобразованием. Поворот вокруг произвольного вектора представляется как комбинация поворотов вокруг координатных осей.

8 Практическое приложение инструментов компьютерной графики

Моментом рождения компьютерной графики в том современном понимании, что было описано в предыдущих разделах пособия, принято считать период возникновения мониторов, которые могли выводить графические объекты принято.

Первой сферой применения и полем для экспериментов с возможностями компьютерной графики стали компьютерные игры, и уже после этого получила более широкое распространение.

В некотором общем смысле понятие компьютерная графика - это все, для отображения чего на мониторе используется визуальная, образная среда.

Тем не менее, это слишком абстрактное определение, поэтому сузим понятие до практического применения.

Мы же под компьютерной графикой будем понимать процесс создания, обработки и вывода изображений разного рода с помощью средств автоматизации.

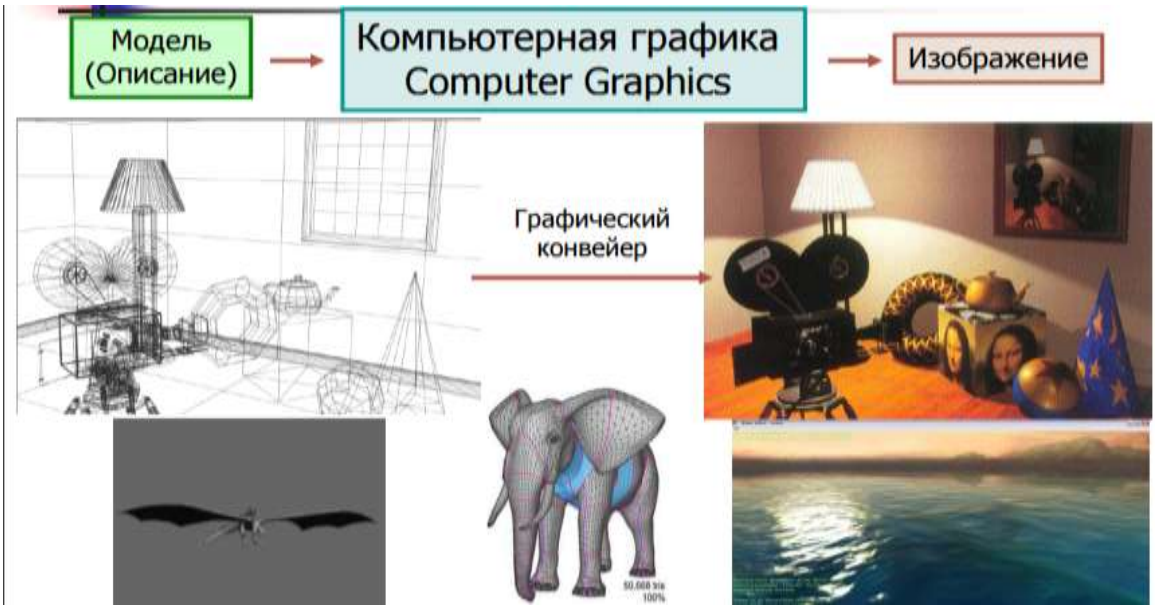


Рисунок 8.1 – Принцип формирования изображения

Важную роль в становлении компьютерной графики в конце 1970-х - начале 1980-х гг. сыграл выпуск компанией Apple компьютеров Macintosh, которые в свое время произвели настоящий прорыв по ряду причин:

- Macintosh серийно поставлялся с цветным монитором;

- операционная система этого компьютера обладала наглядным, визуальным интерфейсом (это было своего рода прообраз Windows);

- их мощности было достаточно для обработки компьютерной графики.

На рынке программных продуктов появилось несколько достаточно функциональных для своего времени графических редакторов.

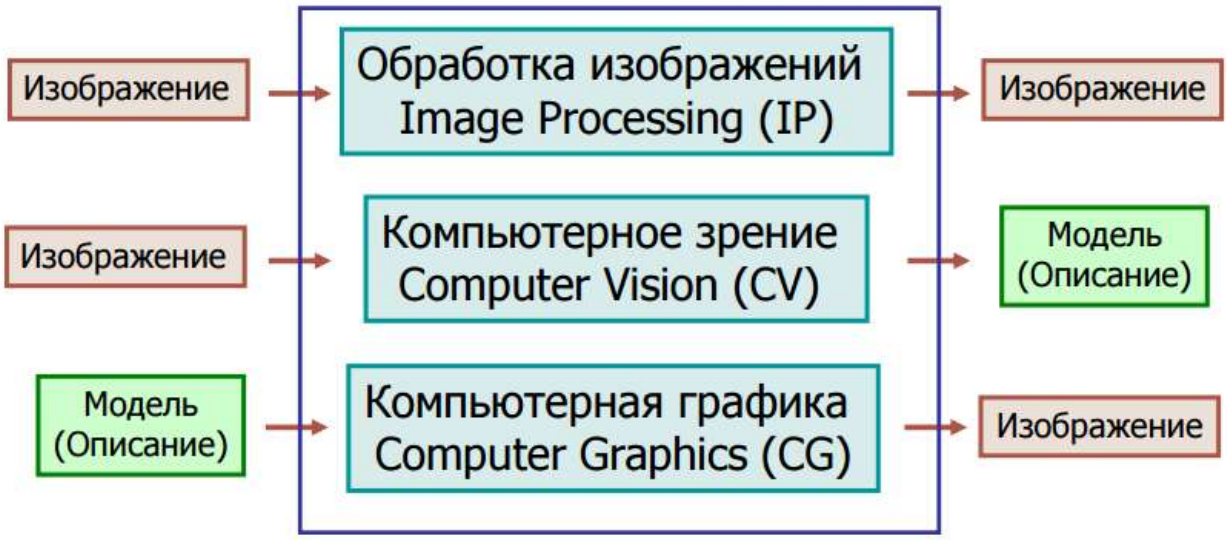


Рисунок 8.2 – Основные направления работы с изображениями

8.1 Области применения компьютерной графики в деятельности инженера

Сфера применения компьютерной графики включает четыре основных области.

1) Отображение информации

В случае отображения информации всегда существует проблема представления накопленных данных, например, статистика о выручке от

продаж, сведения о динамике производства продукции и тому подобное. В данном случае решение лучше всего будет реализовано с использованием средств графического отображения.

Большинство сфер научной и практической деятельности не возможно без графического представления данных. Кроме визуализации результатов экспериментов и анализа данных натурных наблюдений существует обширная область математического моделирования процессов и явлений, которая полностью основывается на графическом выводе. Например, динамические экономические процессы целесообразнее визуализировать, также как и описании маркетинговой деятельности. В геологии в результате обработки трёхмерных натурных данных можно получить геометрию пластов, залегающих на большой глубине.

В медицине сегодня распространены методы диагностирования диагнозов на основе визуального представления состояния внутренних органов. Томография, в том числе ультразвуковое исследование, может создавать трёхмерные данные для последующего их обработки.

2) Проектирование

В технической сфере, а особенно в строительстве чертежи давно являются базой проектирования новых сооружений или изделий. Процесс проектирования, если это требуется, может быть итеративным, то есть конструктор перебирает множество вариантов для последующего выбора наилучшего по ряду параметров. Не мало важное влияние на этот процесс оказывают требования заказчика, который не всегда четко представляет себе конечную цель и технические возможности. Построение предварительных эскизов - относительно продолжительный и дорогостоящий процесс. В последнее время разработаны современные мощные программные средства автоматизации проектно-конструкторских работ (САПР), позволяющие оперативно создавать чертежи объектов, выполнять прочностные расчеты и тому подобное. САПР позволяют как рисовать проекции изделий, так и просматривать их в объемном виде с различных ракурсов. Подобные

программные средства актуальны и полезны для дизайнеров интерьера, ландшафта.

3) Моделирование

Моделирование в данном случае должно восприниматься как имитация различного рода ситуаций, скажем походка человека, полет птицы и так далее. В английском языке термин лучше передается понятием simulation. Но, стоит отметить, что применение данного инструмента не ограничено созданием различного рода симуляторов-тренажеров. В рекламе, различного формата кинофильмах сегодня синтезируются движущиеся объекты, визуально практически не уступающие тем, которые могут быть получены посредством съемок. Более того, КГ позволяет киноиндустрии создавать такие спецэффекты, представление которых было просто невозможно. В последние время повсеместно распространилась еще одна сфера практического использования достижений компьютерной графики - создание виртуальной реальности.

4) Графический пользовательский интерфейс

Первоначально, когда мониторы использовались в качестве устройства вывода информации диалог «человек-компьютер» в основном осуществлялся в алфавитно-цифровом виде. Сегодня же все современные системы программирования, за редким исключением, применяют графический интерфейс. Особенно значительны разработки в области сети Internet. Разработано множество различных программ-браузеров, реализующих в том или ином виде средства общения в сети, без которых доступ к ней невозможен. Такие программы могут работать в любых операционных средах, однако, реализуют, фактически, одни и те же функции, включающие окна, баннеры, анимацию и тому подобное.

В современной компьютерной графике можно выделить следующие основные направления: изобразительная компьютерная графика, обработка и анализ изображений, анализ сцен (перцептивная компьютерная графика), КГ

для научных абстракций (когнитивная компьютерная графика, то есть графика, способствующая познанию).

Изобразительная компьютерная графика в качестве своего предмета имеет синтезированные изображения. Приведем основные классы задач, решение которых выходит в сферу изобразительной КГ:

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

Обработка и анализ изображений касаются как правило дискретного (цифрового) представления фотографий и других изображений. Средства компьютерной графики в данном случае применяются 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 связан с изучением абстрактных моделей графических объектов и связей между ними. Объекты могут быть как искусственно сформированы, так и выделенными на фотографиях. К подобным задачам относятся, моделирование «машинного зрения» (роботы), анализ рентгеновских снимков с выделением и отслеживанием интересующего объекта (внутреннего органа), разработка систем видеонаблюдения.

Когнитивная компьютерная графика - современное научное направление в сфере графики. Когнитивной называют графику, ориентированную прежде всего на научные абстракции, позволяющая сформировать новое научное знание. Технической основой для нее являются высокопроизводительные компьютеры и мощные средства визуализации.

Одним из наиболее ранних примеров практического применения когнитивной КГ является работа Ч.Страуса «Неожиданное применение ЭВМ

в чистой математике». В ней отражено каким образом используется «n-мерная» доска на основе графического терминала для анализа сложных алгебраических кривых. Пользуясь устройствами ввода, математик может быстро получать геометрические представления результатов направленного изменения параметров исследуемой зависимости. Он может также легко управлять текущими значениями параметров, «углубляя тем самым свое понимание роли вариаций этих параметров». В результате получено «несколько новых теорем и определены направления дальнейших исследований».

8.2 Применение компьютерной графики в экономике

Как показывает практика, применение компьютерной графики ведется не только профессиональными дизайнерами и художниками, но и представителями других профессий, в частности экономической сферы. Объясняется это возможностями графики во многом облегчить проведение анализа результатов принятия управленческих решений, анализ экономических показателей развития предприятия. управления производством, представляя их в графическом виде. Также графика применяется в управленческой и коммерческой деятельности с целью создания диаграмм, графиков, схем, которые предназначены для наглядного отображения разнообразной информации.

При использовании графических изображений проще анализировать ситуацию и в целом воспринимать информацию. Графическая визуализация бизнес-процессов позволяет решать плановые задачи на диаграмме. Для решения психологических и аналитических задач прибегают к средствам коммерческой графики. Психологическая задача заключается в том, что надо представить документ так, чтобы он казался солидным и убедительным, что способствовало бы его скорейшему согласованию и утверждению. А аналитическая задача выражается в поиске надежных и выгодных решений.

Цель бизнес-документов заключается в задаче убедить вкладчиков, инвесторов, спонсоров и других лиц стимулировать капитальные вложения, одобрить коммерческую политику, и так далее [29].

1) Научная графика Первоначально вычислительные машины применялись исключительно для решения научных и производственных проблем. С целью улучшения понимания выводимых результирующих данных использовалась их графическое представление: графики, диаграммы и чертежи. Первые графики на компьютере получали в режиме символьной печати. Позднее возникли специальные устройства - плоттеры, ориентированные на получение чертежей и графиков чернильным пером на холсте. Научная КГ сегодня предоставляет возможность реализовать вычислительные эксперименты с визуальным отображением их результатов.

Ярким примером практической реализации научной графики является – **имитационное моделирование**. В экономике посредством имитации можно решить задачи оптимизации бизнес-процессов, что позволит значительно облегчить деятельность топ-менеджера. Имитация позволяет определить, как преобразования повлияют на компанию, ставя эксперименты не на «живой» организации, а на ее модели. Учитывая, что в качестве объекта оптимизации может быть абсолютно любой параметр: операторы, принимающие звонки, количество обслуживаемых клиентов или производимый продукт, возникающие идеи по улучшению могут быть абсолютно разные: от ликвидации отдела до внедрения новых технологий. Понять, какая идея наиболее выигрышная — сложно. А проводить эксперименты на реальной компании — слишком дорого. Кроме того, не рационально проводить натурные эксперименты над сотрудниками.

2) Деловая графика - это сфера применения инструментов компьютерной графики, разработанная для визуального отображения различных параметров работы учреждений. С помощью деловой графики наглядно представляют финансовую составляющую фирмы, а также создают квартальные отчеты. Плановые показатели экономического развития,

отчетная документация бухгалтеров или сводные налоговые отчеты, статистические сводки об объемах произведенной продукции или реализованной все это примеры объектов деловой графики. Наиболее распространенными средами реализации возможностей деловой графики являются электронные таблицы, СУБД и механизм анализа данных в 1С:Предприятие. Деловую графику можно применять также и для выявления ассоциаций в покупках клиентов, что позволяет решать задачу мерчендайзинга.

Деловая графика широко применяется в проектах по реинжинирингу деятельности компаний, когда необходимо заранее спрогнозировать результаты.

Показатели, которые подвергаются оптимизации в первую очередь, это:

- затраты процесса;
- продолжительность процесса;
- количество обслуженных клиентов или количество произведенного продукта.

3) Конструкторская графика применяется в профессиональной деятельности инженеров-конструкторов, архитекторов, изобретателей новой техники. Данная область КГ считается обязательным элементом САПР. Инструменты конструкторской графики позволяют представлять как плоские изображения (проекции, сечения), так и пространственные 3D рисунки.

Одним из примеров конструкторской графики в экономике является BIM - информационное моделирование. Подход, при котором проектировщик использует BIM, позволяет более эффективно принимать бизнес-решения на основе комплексных данных, содержащихся в информационной модели.

Кроме того, конструкторские модели, построенные на основе САПР позволяют планово-экономическому отделу рассчитать нормо-часы сотрудников, с последующей выплатой заработной платы и стоимости самой проектируемой детали.

4) Иллюстративная графика - это свободное рисование и черчение на мониторе компьютера. Программы такого вида КГ классифицируются как прикладное программное обеспечение (ППП) общего назначения. Наиболее простые пакеты иллюстративной графики называются графическими редакторами.

К ППП общего назначения относят текстовые процессоры, которые являются незаменимыми инструментами при формировании различного рода сводных экономических отчетов о деятельности предприятия. Годовой бухгалтерский отчет сводится в текстовом процессоре и содержит элементы деловой графики. Также, для презентирования проектов для инвесторов используют еще один вид ППП – графические редакторы, что обусловлено необходимостью иллюстрирования возможных перспектив или уже достигнутых финансовых результатов.

Незаменимым средством построения иллюстративной графики являются возможности MS Exel. Данный пакет – это табличный процессор, который позволяет создавать приложения пользователя. Они предназначены для автоматизации коммерческих расчетов и позволяют обрабатывать большие таблицы с выводом их на экран и печать. Практически любые экономические расчеты, например по начислению денежных средств на определенные нужды или выплаты по контракту, выполняемые в процессе управления, могут моделироваться с помощью табличных процессоров. Кроме того, в рамках иллюстративной графики возможно использование к экономическим задачам сценарный подход, например, при решении задачи определения план расходов фирмы на будущий год, имея данные о текущих расходах за период.

5) Художественная и рекламная графика - вид КГ, который стал популярен и распространен в основном благодаря телевидению. Сегодня автоматизированно разрабатываются рекламные ролики, мультфильмы, компьютерные игры, видеоуроки, видеопрезентации. Программные продукты этого класса графики требовательны к быстродействию и памяти ПК.

Уникальной характеристикой этих графических пакетов является возможность создания реалистических изображений и анимационных рисунков. Получение картин трёхмерных объектов, их повороты, приближения, удаления, деформации производится с большим объемом вычислений. Передача освещенности объекта в зависимости от положения источника света, от расположения теней, от фактуры поверхности, требует расчетов в области оптики.

Для экономиста-управленца такой вид графики полезен при формировании маркетинговой стратегии по продвижению товара на рынке. Одним из направлений развития рекламной графики является становление частной предпринимательской деятельности в России. В то время как появлению модернисткого направления в рекламной графике способствовало бурное развитие рыночной экономики, так и реклама стала двигателем торговли.

6) Компьютерная анимация - это создание передвигающихся элементов на мониторе компьютера. Разработчик формирует 2 рисунка, которые соответствуют первоначальному и конечному состояниям анимированного объекта. Промежуточные же положения рассчитывает и отображает автоматически программа. Расчеты выполняются на основе мат. описания определенного вида движения. Результирующие изображения, выводимые последовательно на дисплей с некоторой частотой, создают иллюзию движения.

Применение анимации в экономической сфере для России – это новое направление и оно только начинает формироваться в виде технопарков. Кроме того, рекламные анимации – это средство продвижения бизнеса. Рекламный анимированный видеоряд приковывает внимание потребителя к экрану значительно чаще, чем привычные ролики на ТВ. Причем 3D анимация имеет больший успех по сравнению с плоскими изображениями.

7) Мультимедиа - это комбинация на мониторе высококачественного изображения и звукового сопровождения. Наибольшее

распространение системы мультимедиа получили в области обучения, рекламы, а также развлечений.

Преимущество мультимедиа заключается в минимальном времени на выдачу информации пользователю в удобном для него виде или объеме. При этом фактор времени, местоположение информации и расстояние практически не имеют значения — это существенно повышает организационную и **экономическую эффективность управленческого решения**. В качестве большой сети мультимедиа используется Интернет — гигантская мировая система компьютерных сетей, программного обеспечения и информации.

В качестве одного из важнейших достоинств средств мультимедиа следует выделить возможность создания с их помощью интерактивных презентаций, получивших широкое распространение в экономике и бизнесе. Использование мультимедиа презентаций позволяет повышать эффективность рекламы компаний, работы с персоналом и клиентами, а также предоставляет возможности анимационного сопровождения изделий.

Сегодня достаточно распространен термин «бизнес-презентация», которая ориентирована на внешнюю сторону работы фирмы. Торговые презентации используются торговыми агентами при заключении сделок. Торговые презентации позволяют за короткое время представить всю необходимую информацию о товаре, его преимуществе и тем самым значительно сэкономить время. Назначение маркетинговых презентаций (сообщающих основные сведения о приоритетных направлениях деятельности фирмы и продукции, которую она выпускает) состоит в подготовке почвы для будущих торговых презентаций. Маркетинговые презентации применяются в рекламной деятельности. Финансовые презентации нацелены на получение средств на фондовом рынке, так как освещают финансовую деятельность фирмы. Корпоративные презентации предназначены для ознакомления новых работников фирмы и кандидатов в

сотрудники со структурой фирмы, с ее культурными традициями и творческой атмосферой и рассказывают о преимуществах работы в ней.

Список использованных источников

1 Козик Е.В. Компьютерная графика [Электронный ресурс] : учеб. пособие / Е. В. Козик, С.В. Хазова, Н.А. Северюхина. - М. : ПРОМЕДИА, 2012. - 109 с. - ISBN 978-3-8473-9245-3.

2 Горельская, Л. В. Инженерная и компьютерная графика [Электронный ресурс] : электронное гиперссылочное учебное пособие / Л. В. Горельская, С. И. Павлов, Ю. В. Семагина. - М-во образования и науки Рос. Федерации, Федер. гос. бюджет. образоват. учреждение высш. проф. образования «Оренбург. гос. ун-т». - Оренбург : ОГУ, 2015.

3 Аверин, В.Н. Компьютерная инженерная графика : учеб. пособие / В. Н. Аверин.- 6-е изд. - Москва : Академия, 2014. - 224 с. - ISBN 978-5-4468-1152-6.

4 Селезнева, С. А. Компьютерная графика [Электронный ресурс] / С.А. Селезнева – Пенза : РИО ПГСХА, 2014.

5 Конакова, И. П. Инженерная и компьютерная графика [Электронный ресурс]: учеб. пособие / И.П. Конакова, И.И. Пирогова – Екатеринбирг : издательство Уральского университета, 2014.

6 Куликов, А.И. Алгоритмические основы современной компьютерной графики : учеб. пособие / А.И. Куликов, Т.Э. Овчинникова. - Москва : Интуит НОУ, 2016. - 231 с. - ISBN 978-5-94774-654-9.

7 Бейбалаев, В. Д. Математические модели неравновесных процессов в средах с фрактальной структурой : диссертация кандидата физико-математических наук : 05.13.18 / В.Д. Бейбалаев; [Место защиты: Юж. федер. ун-т]. - Таганрог, 2009. - 132 с. : ил.

8 Дёмин, А.Ю. Компьютерная графика : учеб. пособие / А.Ю. Дёмин, А.В. Кудинов / Том. политехн. ун-т. – Томск, 2005. – 160с.

9 Беклемишев, Д.В. Курс аналитической геометрии и линейной алгебры : учеб. для вузов / Д. В. Беклемишев.- 12-е изд., испр. - М.: Физматлит, 2008. - 312 с.

10 Перемитина, Т.О. Компьютерная графика: учеб. пособие / Т.О. Переметина. – Томск: Эль Контент, 2012, 144 с. - Режим доступа: http://biblioclub.ru/index.php?page=book_view&book_id=208688

11 Иванов, Д. В. Алгоритмические основы растровой машинной графики: учеб. пособие / Иванов Д.В., Карпов А.С., Кузьмин Е.П., Лемпицкий В.С., Хропов А.А. – М.: Интернет- Университет Информационных Технологий, 2007, 256 с. Режим доступа: http://biblioclub.ru/index.php?page=book_ view& book_id=233998

12 Лукинова, С.Г. Основы финасовой математики : учеб. пособие / С.Г. Лукинова, Н.А. Лозовая. - Красноярск: КФ МЭСИ, 2005, - 120с.

13 Никулин, Е. А. Компьютерная геометрия и алгоритмы машинной графики [Текст] : учеб. пособие / Е. А. Никулин. - Санкт Петербург : БВХ-Петербург, 2005. - 576 с. : ил. - Библиогр.: с. 549-550. - Предм. указ.: с. 551-560. - ISBN 5-94157-264-6.

14 Кормен, Т. Алгоритмы: построение и анализ: учеб. для вузов: пер. с англ / Т. Кормен, Ч. Лейзерсон, Р. Ривест. -М. : МЦНМО, 2002. - 960 с. : ил.- (Классические учеб.: computer scince). - Библиогр.: с. 901-955. - ISBN 5-900916-37-5.

15 Кормен, Т. Вычислительная геометрия / Т. Кормен, Х. Лейзерсон, И. Чарльз, Р. Ривест, К. Штайн - М.: «Вильямс», 2005. 1296 с.

16 Кормен, Т. Алгоритмы: построение и анализ / Т. Кормен, Х. Лейзерсон, И. Чарльз, Р. Ривест, К. Штайн - 2-е издание. - М.: «Вильямс», 2005. 1047 с. - ISBN 5-8459-0857-4.

17 Препарата, Ф. Вычислительная геометрия: введение / Ф. Препарата. - М.: Мир, 1989. - 478 с.

18 Ласло, М. Вычислительная геометрия и компьютерная графика на С++ / М Ласло. - М.: БИНОМ, 1997. - 304 с.

19 Сидоренко, С.М. Вычислительная геометрия / С.М. Сидоренко - М.: Луч, 1995. - 205 с.

20 Роджерс, Д. Математические основы машинной графики / Д.Роджерс, Дж. Адамс - М.: Мир, 2001. - 604с.

21 Роджерс, Д. Алгоритмические основы машинной графики / Д. Роджерс - М.: Мир, 1989. - 512с.

22 Чекмарев, А.А. Инженерная графика: учеб. для вузов / А.А. Чекмарев. - Москва: Высш. шк., 2007- 382 с.

23 Порев, В.Н. Компьютерная графика / В.Н. Порев – СПб: БХВ - Петербург, 2005. – 432с.

24 Горельская, Л.В. Компьютерная графика /Л.В. Горельская, А.В. Кострюков, С.И. Павлов - Оренбург: ИПК ОГУ, 2003. - 149 с.

25 Петров, М.Н. Компьютерная графика: учеб. пособие для вузов./ М.Н. Петров, В.П. Молочков. - СПб.: Питер, 2003. - 736 с.

26 Никулин, Е.А. Компьютерная геометрия и алгоритмы машинной графики./ Е.А. Никулин - СПб.: БХВ-Петербург, 2003. – 560 с.

27 Павлидис, А. Алгоритмы машинной графики и обработки изображений /А. Павлидис - М.: Радио и связь, 1986. – 400 с.

28 Шикин, Е.В. Компьютерная графика. Полигональные модели. [Электронный ресурс]: учебно-справочное издание / Е.В. Шикин, А.В. Боресков. - М.: Диалог-МИФИ, 2005. – 462с. - ISBN: 5-86404-139-4. – Режим доступа: http://biblioclub.ru/index.php?page=book&id=89300

29 Ширшов, Е.В. Финансовая математика : учеб. пособие / Е.В. Ширшов, Н.И. Петрик, А.Г. Тутыгин, Т.В. Меньшикова / 5-е изд., перераб. и доп. - М. : КНОРУС, 2010. - 144 с. ISBN 978-5-406-00823-2

Учебное пособие

Анна Михайловна Горюнова
Татьяна Валентиновна Омельченко
Петр Николаевич Омельченко
Дмитрий Алексеевич Муслимов

ПРИМЕНЕНИЕ КОМПЬЮТЕРНОЙ ГРАФИКИ ДЛЯ РЕШЕНИЯ ЭКОНОМИЧЕСКИХ И ИНЖЕНЕРНЫХ ЗАДАЧ

ISBN 978-5-7410-2019-7