МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Государственное образовательное учреждение высшего профессионального образования
«Амурский государственный университет»
(ГОУВПО «АмГУ»)

Компьютерная графика

Учебно-методический комплекс дисциплины

по направлению подготовки

010600.68 – «Прикладные математика и физика»

Утвержден на заседании кафедры теоретической и экспериментальной физики
«__» _____ 20__г.,
(протокол № __ от «__»_____20__г.)
Зав. кафедрой _____Е.А. Ванина

Печатается по решению
редакционно-издательского совета
инженерно-физического факультета
Амурского государственного университета

Рокосей В.А.

Учебно-методический комплекс по дисциплине «Компьютерная графика» для направления подготовки 010600.68 «Прикладные математика и физика». – Благовещенск: Амурский гос. ун-т, 2010, 92 с.

Учебно-методические рекомендации ориентированы на оказание помощи магистрантам очной формы обучения по направлению подготовки 010600.68 «Прикладные математика и физика». В комплексе отражены основные направления научной деятельности преподавателей факультета.

Содержание

1. Рабочая учебная программа

1.1 Пояснительная записка

Целью дисциплины «Компьютерная графика» является изучение базовых понятий начертательной геометрии и машинной графики, математических методов и часто применяемых алгоритмов построения изображений. Уделяется внимание работе в графических редакторах. УМКД содержит рабочую учебную программу, теоретический материал, методические рекомендации для студентов по изучению курса, дидактические материалы для контроля усвоения учебного материала, глоссарий, список литературы.

Курс «Компьютерная графика» объемом 150 часов читается на 6 курсе. В том числе: лекции – 36 часов, лабораторные занятия – 36 часов, самостоятельная работа – 78 часов. Итоговый контроль – экзамен.

Целью дисциплины «Компьютерная графика» является изучение базовых понятий начертательной геометрии и машинной графики, математических методов и алгоритмов. Уделяется внимание работе в графических редакторах.

В результате изучения программы курса студенты должны:

- знать основные понятия начертательной геометрии и машинной графики, математические основы компьютерной графики, технические основы построения изображений на экране компьютера, особенности представления растровых и векторных изображений в памяти компьютера, структуру и основные этапы разработки алгоритмов машинной генерации графических примитивов;
- уметь работать в графических редакторах Adobe Photoshop, GIMP, Inkscape, Blender, 3DS MAX, по предоставленному алгоритму построения фрактала написать программу его построения на самостоятельно выбранном языке программирования.
- иметь представление об алгоритмах сжатия изображений, алгоритмах фрактальной графики, о возможностях других графических редакторов.

Воспитательные цели преподавания дисциплины:

- формирование у студентов ценностного отношения к собственной жизни, потребности в ее проектировании и реализации;
- развитие природных задатков и творческого потенциала личности, формирование у студентов желания и потребности реализовать свои способности;
- формирование общечеловеческих норм гуманистической морали, культуры общения, культивирование интеллигентности как высокой меры воспитанности;
- воспитание положительного отношения к труду, развитие потребности в творческом труде, воспитание социально значимой целеустремленности;

- приобщение студентов к системе культурных ценностей, отражающих богатство общечеловеческой культуры;
- развитие у студентов внутренней свободы, способности к самоопределению, самореализации, саморазвитию.

При изучении дисциплины предполагается знание студентами математики, геометрии, информатики, владение навыками работы на компьютере.

1.2.Учебно-тематический план

Наименование разделов и темы	Всего часов	Виды учебных занятий			
		Лекции	Практи-ческие	Лабора-торные	Самостоятельные
Компьютерная графика					
1. Введение. Отображение геометрического объекта на плоскости.	8	4	-	-	4
2. Аппарат проецирования. Математические основы компьютерной графики.	16	8	-	-	8
3. Аппаратная база машинной графики.	12	4	-	-	8
4. Представление объектов и их машинная генерация. Программные средства компьютерной графики. Графические редакторы. Графические языки.	66	6	-	30	30
5. Интерактивная машинная графика как подсистема систем автоматического проектирования.	12	4	-		8
Фрактальная графика					
6. Основы фракталов.	6	2	-	-	4
7. Классические фракталы и самоподобие. Фрактальная графика.	18	4	-	6	8
8. Кодирование изображений с помощью простых преобразований.	12	4	-		8

Наименование разделов и темы	Всего часов	Виды учебных занятий			
		Лекции	Практи-ческие	Лабора-торные	Самостоятельные
ВСЕГО:	150	36	-	36	78

1.3. Содержание тем

1. Введение. Отображение геометрического объекта на плоскости.

   Краткая историческая справка. Область применения компьютерной графики. Классификация графических изображений. Виды компьютерной графики.

2. Аппарат проецирования. Математические основы компьютерной графики.

   Точка, прямая, плоскость, линия, поверхность, их пересечения, развертки; центральное и параллельное проецирование; теорема Польке; способ замены плоскостей проекций; метрические задачи; позиционные задачи; аксонометрические проекции. Преобразования в двухмерном пространстве. Преобразования в трехмерном пространстве. Аффинное проецирование. Перспективное проецирование. Стереографическая и специальные перспективные проекции. Масштабирование в окне. Нахождение параметров плоскости.

3. Аппаратная база машинной графики.

   Видеосистема персонального компьютера. Цвет. Цветовые модели. Форматы графических файлов.

4. Представление объектов и их машинная генерация.

   Программные средства компьютерной графики: базовые средства (графические объекты, примитивы и их атрибуты), графические возможности языков высокого уровня, графические редакторы; графические языки: основные конструкции, представление алгоритмов изображения объектов; графические библиотеки и их использование.

5. Интерактивная машинная графика как подсистема систем автоматического проектирования.

   Понятие системы автоматизированного проектирования. Аппаратные средства. Программные средства. Системы геометрического моделирования.

6. Основы фракталов.

   Обратная связь и итерация; принцип обратной связи; основные типы процессов обратной связи; побочный эффект малых возмущений; устойчивость вычислений.

7. Классические фракталы и самоподобие: множество Кантора; фракталы Серпинского; кривая Коха; кривые, заполняющие плоскость; фракталы и проблемы размерности; фрактальные кривые и рекурсии. Множества Жюлиа и Мандельброта и их компьютерное построение. Динамические процессы. Фрактальная графика.
8. Кодирование изображений с помощью простых преобразований. Фрактальное сжатие изображений. IFS-фракталы. Декодирование сжатых изображений. Необратимое сжатие, Обратимое сжатие, Общие положения алгоритмов сжатия изображений, Алгоритмы архивации без потерь, Алгоритмы архивации с потерями.

2. Теоретический материал

Теоретический материал, представленный в данном УМКД в своем содержании, в основном, опирается на курс лекций Поповой Е.Ф. «Компьютерная графика» и на учебник Сиденко Л.А. «Компьютерная графика, геометрическое моделирование», а также на электронный ресурс «Компьютерная графика», разработанный Мухиным О.И.

1. Введение. Отображение геометрического объекта на плоскости.

Первые системы машинной графики появились вместе с цифровыми компьютерами. Первым шагом к развитию компьютерной графики можно считать проект WHIRLWIND Массачусетского технологического института. К середине 1960-х годов наступил период плодотворной работы в промышленных приложениях машинной графики. В конце семидесятых появилась возможность создания растровых дисплеев, впервые стало возможным получение цветовой гаммы. Растровая технология в конце семидесятых стала доминирующей.

В конце восьмидесятых возникло новое направление рынка на развитие аппаратных и программных систем сканирования, автоматической оцифровки. Оригинальный толчок в таких системах должна была создать магическая машина Ozalid, которая бы сканировала и автоматически векторизовала чертеж на бумаге, преобразуя его в стандартные форматы CAD/CAM.

Основными достоинствами графического интерфейса являются наглядность и простота использования. Но графические же возможности не ограничиваются интерфейсом, их

применение обширно - от простейших иллюстраций и анимированных изображений до красочных презентаций и сложнейших спецэффектов.

Область применения компьютерной графики (КГ), можно сказать, безгранична, будь то инженерная и научная информация, бизнес и искусство, развлечения. Сформулируем точнее основные области применения компьютерной графики:

- Графический интерфейс пользователя.
- Создание иллюстративного материала.
- Автоматизированное проектирование двумерных и условно двумерных объектов.
- Создание плоских анимационных роликов.
- Трехмерная (3D) графика.
- Автоматизированное проектирование 3D объектов.

В настоящее время вся графика с точки зрения отображения ее на экране монитора является растровой, даже та, которую называют векторной. Различие состоит только в способе хранения и обработки файлов.

Несмотря на то, что для работы с компьютерной графикой существует множество классов программного обеспечения, различают всего три вида компьютерной графики.

Растровую графику применяют при разработке электронных и полиграфических изданий. Иллюстрации, выполненные средствами растровой графики, редко создают вручную с помощью компьютерных программ. Чаще для этой цели сканируют рисунки, подготовленные на бумаге, или фотографии. В последнее время для ввода растровых изображений в компьютер нашли широкое применение цифровые фото- и видеокамеры. Соответственно, большинство графических редакторов, предназначенных для работы с растровыми иллюстрациями, ориентированы не столько на создание изображений, сколько на их обработку. Растром называют прямоугольную матрицу пикселей экрана. Термин растровая графика достаточно очевиден, если усвоить понятия, относящиеся к растровым изображениям.

Растровые изображения напоминают лист клетчатой бумаги, на котором любая клетка (пиксель) закрашена каким-либо цветом, образуя в совокупности рисунок. Пиксель - основной элемент растровых изображений. Именно из таких элементов состоит растровое изображение.

В зависимости от того, на какое графическое разрешение экрана настроена операционная система компьютера, на экране могут размещаться изображения, имеющие 640x480, 800x600, 1024x768 и более пикселей.

Наибольшее влияние на количество памяти занимаемой растровым изображением оказывают три фактора:

- Размер изображения;

- Битовая глубина цвета;
- Формат файла, используемого для хранения изображения.

Программные средства для работы с **векторной графикой** наоборот предназначены, в первую очередь, для создания иллюстраций и в меньшей степени для их обработки. Оформительские работы, основанные на применении шрифтов и простейших геометрических элементов, решаются средствами векторной графики намного проще. В отличие от растровой графики в векторной графике изображение строится с помощью математических описаний объектов, окружностей и линий. В векторной графике объем памяти, занимаемый линией, не зависит от размеров линии, поскольку линия представляется в виде формулы, а точнее говоря, в виде нескольких параметров. Что бы мы ни делали с этой линией, меняются только ее параметры, хранящиеся в ячейках памяти. Количество же ячеек остается неизменным для любой линии.

Основными элементами векторной графики являются графические примитивы. В основе векторной графики лежат математические представления о свойствах геометрических фигур.

Какую бы цветовую модель не применял векторный формат, на размер файла он не влияет, кроме тех случаев, когда файл содержит растровые образы. В обычных векторных объектах значение цвета относится ко всему объекту в целом. Цвет объекта хранится в виде части его векторного описания.

Программные средства для работы с **фрактальной графикой** предназначены для автоматической генерации изображений путем математических расчетов. Создание фрактальной художественной композиции состоит не в рисовании или оформлении, а в программировании. Фрактальную графику чаще применяют для генерации ландшафтов, облаков, деревьев и т.п.

Роль фракталов в машинной графике сегодня достаточно велика. Они приходят на помощь, например, когда требуется, с помощью нескольких коэффициентов, задать линии и поверхности очень сложной формы. С точки зрения машинной графики, фрактальная геометрия незаменима при генерации искусственных облаков, гор, поверхности моря. Фактически найден способ легкого представления сложных неевклидовых объектов, образы которых весьма похожи на природные.

2. Аппарат проецирования. Математические основы компьютерной графики

Аппарат проецирования

Основными графическими объектами, которыми оперирует начертательная геометрия, являются точка, прямая, плоскость, поверхность.

Идеальная точка - объект, не имеющий измерений. Поэтому её считают **нульмерной**. Точки на чертеже обозначают прописными буквами латинского алфавита: **A, B, C, D, E, F...**

Графически на плоскости точка задается декартовыми координатами – числами (x,y), однозначно определяющими местонахождение объекта. В трехмерном пространстве положение точки устанавливают с помощью прямоугольных декартовых координат x , y и z (абсцисса, ордината и аппликата).

В современной математике точкой называют элементы весьма различной природы, из которых состоят различные пространства (например, в n-мерном евклидовом пространстве точкой называют упорядоченную совокупность из n-чисел).

Прямая линия. Если основой построения в геометрии служит понятие расстояния между двумя точками пространства, то прямую линию можно определить как линию, вдоль которой расстояние между двумя точками является кратчайшим.

Прямая линия в линейной алгебре - линия первого порядка на плоскости. Общее уравнение прямой:

$$Ax+By+C=0,$$

где A, B и C - любые постоянные.

Линии на чертежах обозначают строчными буквами латинского алфавита: **a, b, c, d, e, f ...**

Для определения положения прямой в пространстве существуют следующие методы: двумя точками**,** двумя проекциями, точкой и углами наклона к плоскостям проекций.

Геометрические образы, заданные неподвижными объектами, создают статичные геометрические образы. Кинематически прямая задаётся одной точкой и единственным направлением движения. Кинематическую линию в быту называют траекторией.

Плоскость – одно из основных понятий геометрии.

Некоторые характеристические свойства плоскости:

1. Плоскость есть поверхность, содержащая полностью каждую прямую, соединяющую любые ее точки;
2. Плоскость есть множество точек, равноотстоящих от двух заданных точек.

Плоскость в линейной алгебре - поверхность первого порядка: в декартовой системе координат плоскость может быть задана уравнением 1-ой степени. Общее уравнение плоскости:

$$Ax+By+Cz+D=0,$$

где ***A, B, C***, и ***D*** - постоянные, причем ***A, B*** и ***C*** одновременно не равны нулю.

Положение плоскости в пространстве можно определить:

1. Тремя точками, не лежащими на одной прямой линии
2. Прямой линией и точкой, не принадлежащей этой прямой
3. Двумя пересекающимися прямыми
4. Двумя параллельными прямыми

Поверхность - двумерный геометрический образ (можно измерить площадь данного рассматриваемого объекта), представляет непрерывное множество точек, расположение которых подчинено определённому закону, определяемому алгоритмом формирования задаваемого геометрического образа.

Как и линии, поверхности могут быть представлены статично и кинематически.

Поверхность можно рассматривать, как совокупность последовательных положений l_1, l_2… линии l, перемещающейся в пространстве по определенному закону. В процессе образования поверхности линия l может оставаться неизменной или менять свою форму - изгибаться или деформироваться.

Подвижную линию принято называть ***образующей***, неподвижные - ***направляющими***. Такой способ образования поверхности принято называть ***кинематическим***.

По виду образующей различают поверхности ***линейчатые*** и ***нелинейчатые***, образующая первых – прямая линия, вторых – кривая.

Линейчатые поверхности в свою очередь разделяют на так называемые ***развертывающие***, которые можно без складок и разрывов развернуть на плоскость и ***неразвертывающиеся***.

Значительный класс поверхностей формируется движением окружности постоянного или переменного радиуса. Это так называемые ***циклические*** поверхности

Множество линий, заполняющих поверхность так, что через каждую точку поверхности проходит в общем случае одна линия этого множества, называется ***каркасом*** поверхности.

Поверхность может быть задана и конечным множеством точек, которое принято называть ***точечным каркасом***.

Проекции каркаса могут быть построены, если задан ***определитель*** поверхности – совокупность условий, задающих поверхность в пространстве и на чертеже.

Различают две части определителя: геометрическую и алгоритмическую.

Геометрическая часть определителя представляет собой набор постоянных геометрических элементов (точек, прямых, плоскостей и т.п.), которые могут и не входить в состав поверхности.

Вторая часть – алгоритмическая (описательная) – содержит перечень операций, позволяющий реализовать переход от фигуры постоянных элементов к непрерывному каркасу.

Поверхности вращения – это поверхности, созданные при вращении образующей ***m*** вокруг оси ***i.***

Геометрическая часть определителя состоит из двух линий: образующей ***m*** и оси ***i***.

Алгоритмическая часть включает две операции:

1. На образующей ***m*** выделяют ряд точек ***A, B, C, …F***;

2. Каждую точку вращают вокруг оси ***i***.

Из закона образования поверхности вращения вытекают два основных свойства:

1. Плоскость перпендикулярная оси вращения, пересекает поверхность по окружности – *параллели*.

2. Плоскость, проходящая через ось вращения, пересекает поверхность по двум симметричным относительно оси линиям – *меридианам*.

Плоскость проходящая через ось параллельно фронтальной плоскости проекций называется *плоскостью главного меридиана.*

Рассмотрим наиболее распространенные поверхности вращения с криволинейными образующими.

Сфера – образуется вращением окружности вокруг её диаметра.

Тор – поверхность тора формируется при вращении окружности вокруг оси, не проходящей через центр окружности.

Параболоид вращения – образуется при вращении параболы вокруг своей оси.

Гиперболоид вращения – различают одно и двух полостной гиперболоиды вращения.

Поверхность с плоскостью параллелизма представляет собой множество прямых линий ***l*** (образующих), параллельных некоторой плоскости ***α*** (плоскости параллелизма) и пересекающих две данные направляющие ***m***, ***n***.

В зависимости от формы направляющих образуются три частных вида поверхностей.

Цилиндроид. Цилиндроидом называется поверхность, образованная движением прямолинейной образующей по двум направляющим кривым линиям, при этом образующая во всех положениях параллельна плоскости параллелизма.

Коноид. Коноидом называется поверхность, образованная движением прямолинейной образующей по двум направляющим, одна из которых кривая линия, а другая прямая, при этом образующая во всех положениях параллельна плоскости параллелизма.

Гиперболический параболоид. Гиперболическим параболоидом или косой плоскостью называется поверхность, образованная движением прямолинейной образующей, параллельной плоскости параллелизма, по двум направляющим линиям – скрещивающимся прямым.

Аппарат проецирования

Для построения изображения оригинала - его проекции, необходимо определить положение плоскости чертежа П1 относительно оригинала и направления проецирующих лучей, исходящих из центра проецирования S.

При центральном проецировании происходит искажение формы, размеров и некоторых других свойств предмета. Вместе с тем, нетрудно заметить, что часть свойств сохраняется, например, проекция точки является точкой; проекция прямой - тоже прямая линия; если точка принадлежит прямой, то проекция точки принадлежит проекции той же прямой; точка пересечения прямых проецируется в точку пересечения их проекций. Проекция предмета, построенная методом центрального проецирования, называется **перспективой.**

Если отнести центр проецирования в бесконечность, то он станет "несобственной" точкой. В обозримом пространстве все проецирующие лучи, исходящие из несобственной точки, параллельны. Такое проецирование называется параллельным.

Если при параллельном проецировании проецирующие лучи образуют с плоскостью проекций острый угол, то проецирование называется косоугольным. Если параллельные проецирующие лучи перпендикулярны плоскости проекций, такое проецирование называется прямоугольным или

ортогональным. В машиностроении применяется в большей мере ортогональное проецирование.

Свойства ортогонального проецирования

1. Проекция точки на плоскость есть точка: $\mathbf{A \rightarrow A_1}$**.**
2. Проекция прямой в общем случае прямая: $l \rightarrow l_1$; она вырождается в точку, если прямая параллельна направлению проецирования;
3. Если точка принадлежит линии, то проекция точки принадлежит проекции линии:

4. Точка пересечения линий проецируется в точку пересечения их проекций:

5. Проекции параллельных прямых параллельны

   Следствия:

   a. отношение длин отрезков параллельных прямых равно отношению длин их проекций;
   b. если точка, принадлежащая отрезку прямой, делит его в некотором отношении, то проекция точки делит проекцию отрезка в том же отношении.
2. Если геометрическая фигура принадлежит плоскости П, параллельной плоскости проекций (например, П1), то проекция этой фигуры на плоскость П1 конгруэнтна самой фигуре.
3. Проекция геометрической фигуры не изменяется при параллельном переносе плоскости проекций.

Аксонометрические проекции

Слово «аксонометрия» в переводе с греческого означает измерение по осям.

Сущность метода параллельного аксонометрического проецирования заключается в том, что предмет относят к некоторой системе координат и затем проецируют параллельными лучами на плоскость вместе с координатной системой.

Искажение отрезков осей координат при их проецировании на ***П'*** характеризуется так называемым коэффициентом искажения.

Коэффициентом искажения называется отношение длины проекции отрезка оси на картине к его истинной длине.

Так по оси x* коэффициент искажения составляет u=0*x*/0x, а по оси y* и z* соответственно υ=0*y*/0y и ω=0*z*/0z.

В зависимости от отношения коэффициентов искажения аксонометрические проекции могут быть:

Изометрическими, если коэффициенты искажения по всем трем осям равны между собой; в этом случае ***u=υ=ω*;**

Диметрическими, если коэффициенты искажения по двум любым осям равны между собой, а по третьей – отличается от первых двух;

Триметрическими, если все три коэффициента искажения по осям различны.

Аксонометрические проекции различаются также и по тому углу **φ**, который образуется проецирующим лучом с плоскостью проекций. Если **φ≠ 90^0**, то аксонометрическая проекция называется ***косоугольной***, а если **φ= 90^0** – прямоугольной.

ОСНОВНАЯ ТЕОРЕМА АКСОНОМЕТРИИ (теорема ПОЛЬКЕ)

Немецкий ученый Карл Польке (1810 -1876) сформулировал основную теорему аксонометрии:

Три отрезка прямых произвольной длины, лежащих в одной плоскости и выходящих из одной точки под произвольными углами друг к другу, представляют параллельную проекцию трех равных отрезков, отложенных на координатных осях от начала.

Согласно ГОСТ 2.317-69, из прямоугольных аксонометрических проекций рекомендуется применять прямоугольные ***изометрию*** и ***диметрию***.

Способы преобразования комплексного чертежа

Во многих случаях трудоемкость решения задачи зависит не столько от сложности ее условия, сколько от положения заданных геометрических фигур относительно плоскостей проекций.

Во всех случаях, когда заданные геометрические фигуры являются проецирующими, решение задачи, как правило, упрощается,

Такое положение геометрических фигур относительно плоскостей проекций, при котором мы непосредственно по чертежу получаем ответ на поставленный в задаче вопрос, называется наивыгоднейшим.

Таким образом, при решении той или иной задачи бывает целесообразно преобразовать чертеж так, чтобы заданные геометрические фигуры оказались бы в наивыгоднейшем положении относительно плоскостей проекций. Для

этого существуют различные способы преобразования комплексного чертежа. Каждый из них основан на одном из следующих принципов:

1) на изменении положения плоскостей проекций относительно неподвижных геометрических фигур;

2) на изменении положения заданных геометрических фигур относительно неподвижных плоскостей проекций;

3) на изменении направления проецирования, т. е. на замене ортогонального проецирования косоугольным или центральным на одну из старых плоскостей проекций или на какую-нибудь новую.

СПОСОБ ЗАМЕНЫ ПЛОСКОСТЕЙ ПРОЕКЦИЙ

Сущность способа состоит в том, что одну из заданных плоскостей проекций (П1 или П2) заменяют новой плоскостью П4. Новая плоскость проекций П4 выбирается с таким расчетом, чтобы она занимала частное положение по отношению к рассматриваемой геометрической фигуре и была при этом перпендикулярной к незаменяемой плоскости проекций.

ПОЗИЦИОННЫЕ ЗАДАЧИ

Задачи, в которых определяется относительное положение или общие элементы геометрических фигур, называются позиционными.

К ним относятся задачи на принадлежность точки и линии поверхности, задачи, выражающие отношения между геометрическими фигурами, задачи на определение общих элементов геометрических фигур.

ПЕРВАЯ ПОЗИЦИОННАЯ ЗАДАЧА (ПОСТРОЕНИЕ ТОЧЕК ПЕРЕСЕЧЕНИЯ ЛИНИИ И ПОВЕРХНОСТИ)

В зависимости от вида и взаимного расположения линии и поверхности точек их пересечения может быть одна или несколько. Например, прямая линия с алгебраической поверхностью n-го порядка пересекается в n точках. В основу их построения положен способ вспомогательных поверхностей, сущность которого состоит в том, что каждая из искомых точек рассматривается как результат пересечения двух линий, принадлежащих вспомогательной поверхности.

Одна из них является заданной линией, а вторая - линией пересечения вспомогательной и заданной поверхностей.
В соответствии с этим построение точек пересечения линии l и поверхности Ф (независимо от их вида) осуществляется по следующей общей:

1. Через данную линию l проводим вспомогательную поверхность.

2. Определяем линию m пересечения вспомогательной и заданной Ф поверхностей.

3. Отмечаем точку А пересечения линий l и m, которая и является искомой.

В символической записи схема имеет вид:
1) проводим Θ ⊃ l;
2) определяем m = Θ∩ Ф;
3) отмечаем А = l ∩ m = l ∩ Ф.

ВТОРАЯ ПОЗИЦИОННАЯ ЗАДАЧА (ПОСТРОЕНИЕ ЛИНИИ ПЕРЕСЕЧЕНИЯ ДВУХ ПОВЕРХНОСТЕЙ)

Две поверхности пересекаются по линии (совокупности линий), которая одновременно принадлежит каждой из них.

В зависимости от вида и взаимного положения поверхностей линия их пересечения может быть прямой, плоской или пространственной ломаной, плоской или пространственной кривой.

Построение этой линии (независимо от ее формы) сводится к построению ряда точек, одновременно принадлежащих каждой из пересекающихся поверхностей).

Основным способом построения точек, принадлежащих искомой линии пересечения, является способ вспомогательных поверхностей. Сущность его заключается в том, что каждая из искомых точек рассматривается как результат пересечения двух линий, одна из которых является линией пересечения вспомогательной поверхности с одной из заданных, а вторая - линией пересечения той же вспомогательной поверхности с другой из заданных поверхностей.

В соответствии с этим построение произвольных точек 1 и 2, принадлежащих линии l пересечения поверхностей Ф и (независимо от их вида), осуществляется по следующей общей схем):

1. Проводится вспомогательная поверхность Σ, пересекающая заданные поверхности Ф и Ψ.
2. Определяются линии m и n пересечения вспомогательной поверхности с каждой из заданных.
3. Отмечаются точки 1 и 2 пересечения построенных линий m и n, которые и являются искомыми, так как одновременно принадлежат данным поверхностям Ф и Ψ, следовательно, линии *l* их пересечения.

В символической записи схема имеет вид:
1) Σ∩Ф∧Ψ ;

2) m = Σ∩Φ∧ n = Σ∩Ψ;
3) 1 = m ∩ n ∧ 2 = m ∩ n.

МЕТРИЧЕСКИЕ ЗАДАЧИ

Метрическими называются задачи, связанные с измерением расстояний и углов. В них определяются действительные величины и форма геометрических фигур, расстояния между ними и другие характеристики по их метрически искаженным проекциям.

Решение метрических задач основано на том, что геометрическая фигура, принадлежащая плоскости, параллельной плоскости проекций, проецируется на нее в конгруэнтную ей фигуру.

Аффинные преобразования - масштабирование, перенос, отражение или поворот геометрического объекта на плоскости или в пространстве. В аффинных преобразованиях на плоскости особую роль играют несколько важных частных случаев, имеющих хорошо прослеживаемые геометрические характеристики. При исследовании геометрического смысла числовых коэффициентов в приведенных формулах для этих случаев нам удобно считать, что заданная система координат является прямоугольной декартовой. Любое преобразование всегда можно представить как последовательное исполнение (суперпозицию) простейших преобразований вида А, Б, В и Г (или части этих преобразований). Любое аффинное преобразование в трехмерном пространстве может быть представлено в виде суперпозиции вращений, растяжений, отражений и переносов.

Математические основы компьютерной графики

Преобразования в двухмерном пространстве

Преобразования в двухмерном пространстве используются в разнообразных случаях: чтобы отдельные части объекта можно было описывать в различных координатных системах; чтобы типовые и повторяющиеся части можно было располагать в произвольных положениях на чертеже и в пространстве, в том числе с использованием циклов; чтобы без повторной кодировки можно было получать симметричные части объекта; для направленной деформации фигур, тел и их частей; для изменения масштаба чертежа, построения проекций пространственных образов... С аналитической точки зрения преобразования — это пересчет значений координат.

Преобразование точки

Точка на плоскости представляется двумя координатами: |x y|. Матрица преобразования точки выглядит так:

Ниже показано преобразование точки через квадратную матрицу; здесь $x_n = xa + yc$ и $y_n = xb + yd$ — новые координаты точки после преобразования:

Преобразование фигуры

Если представить фигуру как совокупность точек, то можно провести и ее преобразование. В следующем примере задано четыре точки: A(0, 0), B(1, 0), C(1, 1), D(0, 1), каждая из которых после преобразования переходит соответственно в $A^*(0, 0)$, $B^*(a, b)$, $C^*(a + c, b + d)$, $D^*(c, d)$:

Геометрически это соответствует деформации фигуры:

При этом площадь новой фигуры равна площади старой фигуры, умноженной на детерминант матрицы преобразования: $S_2 = S_1 * |ad - bc|$.

Однородные координаты. Операции в них

Любая система координат, в которой представление точки в двухмерном (трехмерном) пространстве задается при помощи трех (четырех) координат (P_1, P_2, P_3(, P_4)), называется системой однородных координат. Вообще, для n-мерного пространства число однородных координат должно быть на единицу больше: n + 1.

Применение однородных координат в общем случае позволяет устранять аномалии, возникающие при работе в декартовых координатах, и представлять сложные преобразования в виде произведения нескольких матриц.

Геометрическая интерпретация на случай двухмерного пространства: введение третьей координаты, равной единице, можно трактовать как переход в трехмерное пространство, в котором разрешено работать только в плоскости z = 1. Следует представлять себе, что экран компьютера (картинная плоскость, плоскость изображения) находится в плоскости z = 1:

В случае выхода рисунка за сечение z = 1 рисунок возвращается принудительно в данное сечение — для того, чтобы были возможны последующие операции:

Такая операция называется нормализацией однородных координат:

Общий вид преобразования

Операция смещения

Матрица преобразования содержит в себе константы *m* и *n*, под действием которых точка смещается на *m* единиц вдоль оси *x* и на *n* единиц — вдоль оси *y*:

Операция масштабирования

За счет коэффициентов *a* и *d* матрицы преобразования происходит увеличение (или уменьшение) значения координат точки (x, y) в *a* и *d* раз по осям *x* и *y* соответственно:

Общее полное масштабирование

В данном случае при s < 1 будет происходить увеличение значения координат точки (x, y) в *s* раз; при s > 1 мы получим обратный эффект — уменьшение значения координат (x, y) в *s* раз.

Поворот на угол q

Здесь q — угол, на который требуется повернуть точку (x, y). Обратите внимание: поворот происходит относительно точки (0, 0) декартовой системы координат против часовой стрелки!

Отображение или зеркалирование

- Зеркалирование относительно прямой y = x (рис. 1.6a):
- Зеркалирование относительно прямой x = 0 (рис. 1.6b):
- Зеркалирование относительно прямой y = 0 (рис. 1.6c):
- Зеркалирование относительно начала координат (рис. 1.6d):

Поворот фигуры вокруг произвольной точки (m, n) на произвольный угол a.

Чтобы провести любое сложное преобразование, необходимо разложить его на базовые операции. Поворот фигуры вокруг произвольной точки (m, n) на произвольный угол a состоит из трех базовых операций: 1) перенос фигуры на вектор A(-m, -n) для совмещения точки (m, n) с началом координат; 2) поворот фигуры на угол a; 3) перенос фигуры на вектор A'(m, n) для возвращения ее в исходное положение. Так как фигуру можно представить набором точек, то операции 1) - 3) можно выполнять последовательно для каждой точки. Покажем это на примере.

Пусть мы хотим повернуть треугольник с координатами A(x, y), B(x_1, y_1), C(x_2, y_2) вокруг точки D(m, n) на угол a. Пусть P_{-s} — матрица переноса точки на вектор A(-m, -n), V_a — матрица поворота на угол a, P_s — матрица переноса точки на вектор A'(m, n).

Итак, мы имеем все данные, необходимые для проведения сложного преобразования первой точки A(x, y):

Точно такие же преобразования необходимо провести для оставшихся двух точек треугольника, подставляя соответствующие их координаты взамен *x* и *y* (последовательность операций см. на рис. 1.7). Таким образом, сложная операция разбивается на простейшие и задается произведением соответствующих матриц преобразования, причем порядок, в котором перемножаются матрицы, существенно определяет результат.

Центральное проецирование (перспектива)

px + qy + 1 = H — плоскость.

Примечания

1. В общем случае от перемены матриц местами результат меняется.
2. Матрицы операций, идущие подряд, можно перемножать раздельно, главное — не менять их порядок следования (см. примечание 1).
3. Линии при описанных выше (аффинных) преобразованиях переходят в линии. Поэтому обычно производится пересчет только координат вершин фигуры, а после этого соответствующие вершины в результирующей фигуре соединяются, как и в исходной фигуре.

Нахождение точки пересечения двух линий (пример)

Пусть имеются две линии: x + y = 1, 2x - 3y = 0, необходимо найти точку их пересечения. Решение может быть найдено с использованием матриц.

Перенесем все члены уравнений в левую часть: x + y - 1 = 0, 2x - 3y - 0 = 0; запишем коэффициенты первого уравнения в первый столбец матрицы, второго уравнения — во второй:

Условие, при котором пересекаются две прямые, выглядит следующим образом: |x y 1| * M = |0 0 1|

Для нахождения ответа необходимо обе части предыдущего уравнения домножить *справа* на обратную матрицу M^{-1} (при перемножении M и M^{-1} получается единичная матрица E): |x y 1| * E = |0 0 1| * M^{-1}

|x y 1| = |3/5 2/5 1|

Ответ: точка пересечения прямых: x = 3/5, y = 2/5.

3. Аппаратная база машинной графики.

Видеосистема персонального компьютера состоит из монитора (дисплея) и видеоадаптера (видеокарты).

Экран дисплея представляет собой прямоугольную матрицу пикселей (pixel от англ. picture element - элемент изображения), обладающих благодаря люминофору, которым покрыт экран, свойством светимости, при попадании на них электронного луча, который построчно слева направо и сверху вниз пробегает по экрану, изменяя свою яркость и цвет.

Потерю яркости светимости в люминофоре устраняют регенерацией с частотой 50 Гц.

Размеры пикселей и расстояния между ними подбираются таким образом, что если светится группа соседних пикселей, то они воспринимаются, как сплошной светящийся участок.

Качество изображения определяется разрешающей способностью экрана – количеством пикселей на единицу длины по вертикали и горизонтали. Каждому пикселю соответствует некоторое число битов в оперативной памяти компьютера.

Конструктивно видеоадаптер представляет собой отдельную плату, связанную с центральным процессором через общую шину. На плате видеоадаптера также размещена микросхема специальной памяти - видеобуфера, куда заносится информация о выводимом изображении.

Цвет - свойство света вызывать определенное зрительное ощущение в соответствии с его спектральным составом.

Распознавание цвета человеком зависит от освещения, объекта, отражающего свет, от глаз и мозга наблюдателя. Свет, попадая в глаз, преобразуется в сигналы нейронов, находящихся в сетчатке глаза, и по оптическому нерву пересылается в мозг.

Человеческий глаз может воспринимать огромное количество цветов, в то время как монитор и принтер в состоянии воспроизводить лишь ограниченную часть этого диапазона. Причем диапазон воспроизводимых цветов и способ их отображения для монитора и принтера тоже различны.

Существует две схемы воспроизведения цветов: аддитивная и субтрактивная.

Аддитивный цвет получается при соединении света разных цветов. В этой схеме отсутствие всех цветов представляет собой чёрный цвет, а присутствие всех цветов - белый. Схема аддитивных цветов работает с излучаемым светом, например, монитор компьютера.

В схеме **субтрактивных** цветов происходит обратный процесс. Здесь получается какой-либо цвет при вычитании других цветов из общего луча света. В этой схеме белый цвет появляется в результате отсутствия всех цветов, тогда как их присутствие даёт чёрный цвет. Схема субтрактивных цветов работает с отражённым светом, например, принтер.

Для описания различных физических процессов воспроизведения цвета, были разработаны различные ***цветовые модели***.

RGB. В основе лежит воспроизведение любого цвета путем сложения трех основных цветов: красного (Red), зеленого (Green) и синего (Blue). Именно на такой модели построено воспроизведение цвета современными мониторами.

CMYK. В качестве основных цветов здесь также принимаются три: Cyan (голубой), Magenta (розовый или его еще называют пурпурный), Yellow (желтый). K (черный от слова blacK) используется для повышения контрастности напечатанных изображений.

HSB и HSL. Системы цветов HSB и HSL базируются на ограничениях, накладываемых аппаратным обеспечением. В системе HSB описание цвета представляется в виде тона, насыщенности и яркости. В другой системе HSL задаётся тон, насыщенность и освещённость. Тон представляет собой

конкретный оттенок цвета. Насыщенность цвета характеризует его относительную интенсивность или частоту. Яркость или освещённость показывают величину чёрного оттенка добавленного к цвету, что делает его более тёмным.

Между этими моделями нет однозначного соответствия, то есть некоторые оттенки цвета одной модели не могут быть в принципе воспроизведены в другой модели и наоборот. Именно этим вызвана необходимость калибровки оборудования (сканера, монитора и принтера) на предмет соответствия цветов.

Все графические данные в компьютере можно разделить на две большие ветви: растровую и векторную. Векторы представляют собой математическое описание объектов относительно точки начала координат. Большинство векторных форматов могут так же содержать внедренные в файл растровые объекты или ссылку на растровый файл (технология OPI). Сложность при передаче данных из одного векторного формата в другой заключается в использовании программами различных алгоритмов, разной математики при построении векторных и описании растровых объектов.

Растровый файл устроен проще. Он представляет собой прямоугольную матрицу (bitmap), разделенную на пиксели. Растровые файлы можно разделить на два типа: предназначенные для вывода на экран и для печати.

Растровые форматы, предназначенные исключительно для вывода на экран имеют только экранное разрешение, то есть один пиксель в файле соответствует одному экранному пикселю. На печать они выводятся так же с экранным разрешением.

Растровые файлы, предназначенные для допечатной подготовки изданий имеют, подобно большинству векторных форматов, параметр Print Size - печатный размер. С ним связано понятие печатного разрешения, которое представляет из себя соотношение количества пикселей на один квадратный дюйм страницы (ppi, pixels per inch или dpi - dots per inch). Печатное разрешение может быть от 130 dpi (для газеты) до 300 (высококачественная печать).

Растровые форматы, так же отличаются друг от друга способностью нести дополнительную информацию: различные цветовые модели, вектора, альфа-каналы, слои различных типов, Interlaced, анимация, возможности сжатия и т.п.

BMP (Windows Device Independent Bitmap). Формат BMP является родным форматом Windows. Применяется для хранения растровых изображений, предназначенных для использования в Windows. Способен хранить как индексированный (до 256 цветов), так и RGB-цвет (16 млн. оттенков).

WMF (Windows Metafile). Еще один родной формат Windows. Служит для передачи векторов через буфер обмена (Clipboard). Понимается практически всеми программами Windows, так или иначе связанными с векторной графикой.

GIF (CompuServe Graphics Interchange Format). Разработан для передачи растровых изображений по сетям. Он использует LZW-компрессию, что позволяет хорошо сжимать файлы, в которых много однородных заливок (логотипы, надписи, схемы). GIF-формат позволяет записывать изображение "через строчку" (Interlaced), благодаря чему, имея только часть файла, можно увидеть изображение целиком, но с меньшим разрешением. Кроме того, файл GIF может содержать не одну, а несколько растровых картинок, которые браузеры могут подгружать одну за другой с указанной в файле частотой. Это называется GIF-анимация. Основное ограничение формата GIF состоит в том, что цветное изображение может быть записано только в режиме 256 цветов.

PNG (Portable Network Graphics). PNG - формат для Сети, призванный заменить собой GIF. Использует сжатие без потерь. Глубина цвета может быть любой, вплоть до 48 бит (RGB, для сравнения, - 24), используется Interlacing, причем не только строк, но и столбцов, поддерживается плавно переходящая прозрачность.

JPEG (Joint Photographic Experts Group). Строго говоря, JPEG - не формат, а алгоритм сжатия, основанный не на поиске одинаковых элементов, а на разнице между пикселями. JPEG ищет плавные цветовые переходы. Лишнюю, цветовую информацию он отбрасывает, усредняя некоторые значения. Вместо действительных значений JPEG хранит скорость изменения от пикселя к пикселю. Можно задать уровень компрессии. Чем выше уровень компрессии, тем больше данных отбрасывается и тем ниже качество. Используя JPEG, можно получить файл в 10-500 раз меньше, чем BMP. Формат аппаратно независим. В JPEG следует сохранять только конечный вариант работы, потому что каждое пересохранение приводит к новым потерям (отбрасыванию) данных**.**

TIFF (Tagged Image File Format). Аппаратно независимый формат TIFF на сегодняшний день является одним из самых распространенных и надежных, его поддерживают практически все программы на PC и Macintosh так или иначе связанные с графикой. Ему доступен весь диапазон цветовых моделей от монохромной до RGB, CMYK и дополнительных цветов Pantone. TIFF может сохранять векторы Photoshop и массу других дополнительных данных.

PSD (Adobe Photoshop Document). PSD - родной формат растрового редактора Photoshop. Он позволяет записывать изображение со многими слоями, их масками, дополнительными каналами, контурами и другой информацией - все, что может сделать Photoshop. Однослойный Photoshop Document понимают ряд программ, многослойные - могут импортировать Illustrator и InDesidn. Fractal Design Painter и Corel PHOTO-PAINT открывают на редакцию многослойные документы Photoshop, причем лишь PHOTO-PAINT 8 открывает файл Photoshop 100-процентно корректно.

CDR (CorelDraw Document). Формат известен в прошлом низкой устойчивостью и плохой совместимостью файлов, тем не менее, пользоваться CorelDraw чрезвычайно удобно, он имеет неоспоримое лидерство на платформе PC.

Многие программы на PC (FreeHand, Illustrator, PageMaker, ...) могут импортировать файлы CDR. 7-ю и 8-ю версии CorelDraw можно без натяжек назвать профессиональными. В файлах этих версий применяется компрессия для векторов и растра отдельно, могут внедряться шрифты, файлы CDR имеют огромное рабочее поле 45x45 метров (этот параметр важен для наружной рекламы); начиная с 4-й версии, поддерживается многостраничность.

4. Представление объектов и их машинная генерация.

Так как в подавляющем большинстве графические устройства являются растровыми, то есть представляют изображение в виде прямоугольной матрицы (сетки, целочисленной решетки) пикселей (растра), то, естественно, возникает необходимость в растровых алгоритмах.

Достаточно важным понятием для растровой сетки является связность - возможность соединения двух пикселей растровой линией, то есть последовательным набором пикселей. При этом возникает вопрос, когда пиксели (X_1,y_1) и (X_2,y_2) можно считать соседними. Вводится два понятия связности:

- 4-связность, когда пиксели считаются соседними, если либо их x-координаты, либо y-координаты отличаются на единицу, то есть |x1-x2| + |y1-y2|≤1;
- 8-связность, когда пиксели считаются соседними, если их x- и y-координаты отличаются не более чем на единицу, то есть |x1 – x2| ≤ 1, |y1 – У2| ≤ 1

В качестве линии на растровой сетке выступает набор пикселей P1, P2, …, Pn, где любые два пикселя Pi,Pi+1 являются соседними.

Простейший алгоритм растрового представления отрезка имеет вид:

```
//File Linel.cpp

void Line(int xl, int yl, int x2, int y2, int color)

{

double k=(y2-yl)/(x2-xl);

double b=yl-k*xl;

for (int x=xl; x<=x2; x++)
```

```
putpixel (x, round (k*x+b), color);

}
```

Алгоритм Брезенхейма

При построении растрового изображения отрезка всегда выбирается ближайший по вертикали пиксель. При этом из двух точек A и B выбирается та, которая ближе к исходной прямой (в данном случае выбирается точка A, так как a<Ь). Для этого вводится число d, равное (x2 – x1)(b — a).

В случае d>0 значение y от предыдущей точки увеличивается на 1, а d -на 2(Δy — Δx)

В противном случае значение y не изменяется, а значение d заменяется на 2Δy.

```
//File Line2.cpp

void Line{int xl, int yl, int x2, int y2, int color)

{

int dx=x2-xl;

int dy=y2-yl;

int d=(dy<<l)-dx;

int dl=dy << l;

int d2=(dy-dx) << l;

putpixel (xl, yl, color);

for (int x=x1+1, y=yl; x<=x2; x++)

{

if (d>0)

{

d+=d2;

y+=1;

}
```

```
else

d+=d1;

putpixel (x, y, color);

}

}
```

Алгоритм вывода окружности

Для вывода контура окружности можно использовать соотношение между координатами X и Y для точек окружности $X^2 + Y^2 = R^2$ и построить алгоритм прямого вычисления координат. Однако тогда необходимо вычислять квадратный корень, а это в цифровом компьютере выполняется медленно.

Кривая Безье

Кривые Безье описываются в параметрической форме: $x = P_x(t)$, $y = P_y(t)$.

Значение t выступает как параметр, которому соответствуют координаты отдельной точки линии. Параметрическая форма описания может быть удобнее для некоторых кривых, чем задание в виде функции y=f(x), поскольку функция f(x) может быть намного сложнее, чем $P_x(t)$ и $P_y(t)$, кроме того,f(x) может быть неоднозначной.

Многочлены Безье для P_x *и* P_y имеют такой вид:

где x_i и y_i - координаты точек-ориентиров P_i, а величины - это известные из *комбинаторики,* так называемые *сочетания* (они также известны как коэффициенты *бинома Ньютона):*

Значение *m* можно рассматривать и как степень полинома, и как значение, которое на I единицу меньше количества точек-ориентиров.

Геометрический алгоритм для кривой Безье

Этот алгоритм позволяет вычислить координаты *(x, y)* точки кривой Безье по значению параметра *t*.

1. Каждая сторона контура многоугольника, который проходит по точкам-ориентирам, делится пропорционально значению *t.*

2. Точки деления соединяются отрезками прямых и образуют новый многоугольник. Количество узлов нового контура на единицу меньше, чем количество узлов предшествующего контура.

3. Стороны нового контура снова делятся пропорционально значению t. И так далее. Это продолжается до тех пор, пока не будет получена единственная точка деления. Эта точка и будет точкой кривой Безье.

Отсечение отрезка

Необходимость отсечь выводимое изображение по границам некоторой области встречается довольно часто. В простейших ситуациях в качестве такой области, как правило, выступает прямоугольник.

Алгоритм заключается в разбиении всей плоскости на 9 областей прямыми, образующими прямоугольник. В каждой из этих областей все точки по отношению к прямоугольнику расположены одинаково. Определив, в какие области попали концы рассматриваемого отрезка, легко понять, где именно необходимо отсечение. Для этого каждой области сообщается 4-битовый код.

4-битовый код:

- бит 0 означает, что точка лежит левее прямоугольника,
- бит 1 означает, что точка лежит выше прямоугольника,
- бит 2 означает, что точка лежит правее прямоугольника,
- бит 3 означает, что точка лежит ниже прямоугольника.

Закраска области, заданной цветом границы

Рассмотрим область, ограниченную набором пикселей заданного цвета, и точку (x, y), лежащую внутри этой области.

```
//File fi111.cpp

void Pixe1Fi11(int x, int y, int BorderColor, int color)

{

int c=getpixel(x, y);

if ((c!=BorderColor) && {c!=color))

{

putpixel (x,y,color);

PixelFill (x -1,y, BorderColor,color);
```

```
PixelFill (x+1,y, BorderColor,color);

PixelFill (x, y-1, BorderColor,color);

PixelFill (x,y+1, BorderColor, color) ;

}

}
```

Простейший алгоритм, показанный выше, хотя и абсолютно корректно заполняющий даже самые сложные области, является слишком неэффективным, так как уже для отрисованного пикселя функция вызывается еще три раза, и, кроме того, требует слишком большого стека из-за большой глубины рекурсии.

Поэтому для решения задачи закраски области предпочтительнее алгоритмы, способные обрабатывать сразу целые группы пикселей.

Удаление невидимых линий и поверхностей

Первые публикации алгоритмов удаления невидимых элементов изображения появились сравнительно недавно, в начале 70-х годов. Первый из опубликованных алгоритмов имел номер 420. Это говорит о том, что существует много различных подходов к решению этой задачи. Необходимость удаления невидимых частей изображения связана с тем, что по каркасному изображению объектов невозможно определить, как выглядит сцена, как расположены объекты.

Известные алгоритмы удаления невидимых линий и поверхностей можно разделить на три группы по типу пространства, в котором производится анализ видимости элементов изображения:

1)объектные,

2)картинные,

3)смешанные.

Кроме того, различные алгоритмы ориентированы на определенный способ задания поверхностей. Поверхность может быть задана аналитически, в виде совокупности многоугольных граней, в виде совокупности частей, каждая из которых задается аналитически.

Adobe PhotoShop

Создание фотореалистических изображений, работа с цветными сканированными изображениями, ретуширование, цветокоррекция, коллажирование, трансформации и цветоделение и другое. Adobe PhotoShop

располагает всеми традиционными методами работы с растровыми изображениями, при этом имеет возможность работы со слоями и использует контуры. Программа позволяет легко изменять цветовое представление документов (битовое, в градациях серого, дуплекс, индексированные цвета, RGB или CMYK).

Adobe PhotoShop - это программа растровой графики, то есть любой элемент изображения строится по точкам (пикселям).

7.2. GIMP

GNU Image Manipulation Program или GIMP (Гимп) — растровый графический редактор, программа для создания и обработки растровой графики. Частично поддерживается векторная графика. Проект основан в 1995 году Спенсером Кимбеллом и Питером Маттисом как дипломный проект, в настоящий момент поддерживается группой добровольцев. Распространяется на условиях GNU General Public License. Типичные задачи, которые можно решать при помощи GIMP, включают в себя создание графики и логотипов, масштабирование и кадрирование фотографий, раскраска, комбинирование изображений с использованием слоёв, ретуширование и преобразования изображений в различные форматы.

- **GIMP является свободным ПО;**
- **GIMP является высококачественным приложением для фоторетуши и позволяет создание оригинальных изображений;**
- **GIMP является высококачественным приложением для создания экранной и веб-графики;**
- **GIMP является платформой для создания мощных и современных алгоритмов обработки графики учёными и дизайнерами;**
- **GIMP позволяет автоматизировать выполнение повторяющихся действий;**
- **GIMP легко расширяем за счёт простой установки дополнений.**

Эти тезисы определяют дальнейшее развитие GIMP.

Adobe Illustrator

Adobe Illustrator - программа для работы с векторной графикой. Эта программа разработана той же фирмой, что и программы PhotoShop и PageMaker. Поэтому существует тесная связь с этими программами. Illustrator позволяет редактировать и экспортировать файлы PDF, имеет улучшенные средства управления цветами и поддерживает ColorSync для Mac и Kodak Digital Science CMS для Windows.

Corel Draw

В состав пакета входит Photopaint. Инструменты Corel Draw! Позволяют к объектам применять напрямую процедуры коррекции и эффекты.

Сейчас можно редактировать объекты по базовым узлам, или трансформировать их при помощи любого инструмента создания объекта.

Новые растровые возможности делают размещение объекта и его изображение более плавным.

Панели инструментов и горячие клавиши можно перенастроить, при этом пакет позволяет сохранять несколько конфигураций рабочего стола.

Имеется возможность обзора архивных файлов в хронологическом порядке.

Когда на экране слишком много окон, навигацию по ним можно упростить, для этого можно сложить их в виде стопки страниц, у которых будут ярлычки.

3D Studio Max

Со времен DOS программы 3D Studio довольно конкурентоспособны. Версия 3D Studio VIZ – ориентирована в основном на проектные задачи. Часть возможностей существующих в 3D Studio MAX 1.x были исключены, но включены новые средства, существенно увеличивающие производительность, объектные привязки в стиле «AutoCAD», CAD-подобные средства работы со сплайнами, удобная работа с текстурами, совмещение объектов сцены с растровой фоновой подложкой, условий солнечного освещения, средства контроля над масштабом присваиваемых объекту текстур и материалов, поддержка файлов в формате DWG. 3D Studio Max R2 содержит все средства, что и 3D Studio VIZ плюс ряд дополнительных возможностей (всего добавлено около 1000 новых функций).

Blender

Blender - пакет для создания трёхмерной компьютерной графики, включающий в себя средства моделирования, анимации, рендеринга, постобработки видео, а также создания интерактивных игр. Программа является свободным программным обеспечением и распространяется под лицензией GNU GPL.В настоящее время Blender является проектом с открытым исходным кодом и развивается при активной поддержке Blender Foundation.

Характерной особенностью пакета Blender является его небольшой размер. Установленный пакет занимает около 10 МБ. В дистрибутив не входит развёрнутая документация и большое количество демонстрационных сцен.

5. Интерактивная машинная графика как подсистема систем автоматического проектирования.

Автоматизированное, проектирование (computer-aided design - CAD) представляет собой технологию, состоящую в использовании компьютерных систем для облегчения создания, изменения, анализа и оптимизации проектов. Таким образом, любая программа, работающая с компьютерной графикой, так же как и любое приложение, используемое в инженерных расчетах, относится к системам автоматизированного проектирования. Другими словами, множество средств CAD простирается от геометрических программ для работы с формами до специализированных приложений для анализа и оптимизации. Между этими крайностями умещаются программы для анализа допусков, моделирования методом конечных элементов и визуализации результатов анализа. Самая основная функция CAD - определение геометрии конструкции (детали механизма, архитектурные элементы, электронные схемы, планы зданий и т. п.), поскольку геометрия определяет все последующие этапы жизненного цикла продукта. Для этой цели обычно используются системы разработки рабочих чертежей и геометрического моделирования. Вот почему эти системы обычно и считаются системами автоматизированного проектирования. Более того, геометрия, определенная в этих системах, может использоваться в качестве основы для дальнейших операций в системах CAE и CAM. Это одно из наиболее значительных преимуществ CAD, позволяющее экономить время и сокращать количество ошибок, связанных с необходимостью определять геометрию конструкции с нуля каждый раз, когда она требуется в расчетах. Можно, следовательно, утверждать, что системы автоматизированной разработки рабочих чертежей и системы геометрического моделирования являются наиболее важными компонентами автоматизированного проектирования.

Графическое устройство состоит из дисплейного процессора и одного или нескольких устройств ввода. Дисплей (монитор) представляет собой экран, на который выводится графическое изображение, однако вывод конкретного изображения на экран выполняется дисплейным процессором.

В состав графического устройства обычно входит одно или несколько устройств ввода. Помимо клавиатуры к ним относятся мышь, трекбол, спейсбол и цифровой планшет с пером и роликом.

Каждое графическое устройство обычно подключается к устройствам вывода, например к плоттеру или цветному лазерному принтеру. Эти устройства могут использоваться несколькими графическими устройствами совместно. Устройства вывода позволяют вывести любое изображение на бумагу, пленку, ткань и т.п.

Векторные графические устройства, появившиеся в середине 60-х годах XX века, состоят из дисплейного процессора, дисплейного буфера памяти и

электронно-лучевой трубки. Основные принципы их функционирования вкратце можно описать следующим образом.

Дисплейный процессор считывает *дисплейный файл (display list),* который представляет собой последовательность передаваемых приложением кодов, соответствующих графическим командам. Дисплейный файл хранится в разделе памяти, который называется *дисплейным буфером (display buffer)*. Дисплейный процессор осуществляет также загрузку дисплейного файла в дисплейный буфер. После этого дисплейный процессор формирует необходимые напряжения на вертикальных и горизонтальных парах отклоняющих пластин таким образом, что электрон, вылетающий с катода, попадает в нужное место внутренней поверхности передней стенки электронно-лучевой трубки.

Однако люминофор излучает свет очень недолго, а затей гаснет. Поэтому изображение требует постоянного перерисовывания с очень высокой скоростью, чтобы пользователю не досаждало мерцание. Изображение сохраняется в мозге около 1/30 с, поэтому пользователь не будет замечать исчезновения изображения, если его перерисовывать с периодом менее 1/30 с. Процесс перерисовки называется *обновлением (refresh),* которое обеспечивается повторяющимся считыванием дисплейного буфера сверху вниз и направлением электронного пучка в соответствии с командами из буфера. Дисплейный буфер используется исключительно для реализации обновления. В зависимости от сложности изображения перерисовка может занимать и более 1/30 с. Тогда часть изображения, построенная в начале цикла обновления, успеет исчезнуть, в то время как другая часть изображения еще будет прорисовываться. Возникнет эффект мерцания картинки в целом. Мерцание изображения и высокая стоимость - основные недостатки устройств векторной графики.

Векторные устройства позволяют отображать динамическую анимацию. Динамика обеспечивается благодаря изменению содержимого дисплейного буфера по мере того, как дисплейный процессор занимается обновлением. Содержимое буфера изменяется графическими командами, передаваемыми управляющим приложением, в данном случае - программой анимации.

Основные принципы функционирования растровых графических устройств можно описать следующим образом. Дисплейный процессор принимает графические команды от приложения, преобразует их в растр, после чего сохраняет растр в разделе памяти, который называется *буфером кадра (frame buffer)*. Размеры точек определяются установленным разрешением. Растровые графические устройства должны хранить в своей памяти изображение в виде растра, в отличие от векторных, хранящих лишь дисплейные файлы. Поэтому требования к памяти у этих двух видов устройств отличаются, как и методы обновления изображения на экране.

Когда дисплейный процессор генерирует изображение и сохраняет его в буфере кадра, он параллельно считывает содержимое этого буфера и направляет электронные пучки на дисплей, воспроизводя на нем картинку, хранящуюся в

буфере. На внутренней поверхности электронно-лучевой трубки может быть столько точек люминофора, сколько точек описывается буфером кадра. Электронный пучок направляется на точки, соответствующие точкам растрового изображения. Время свечения люминофора в растровых устройствах так же коротко, как и в векторных, поэтому необходимо регулярное обновление изображения. Единственное отличие - порядок движения электронного пучка при обновлении. Пучок пробегает по экрану слева направо, переходя со строки на строку в направлении сверху вниз. Когда электронный пучок направляется на точку люминофора, соответствующую точке изображения, он включается, возбуждая свечение люминофора. Время обновления остается постоянным независимо от сложности воспроизводимого изображения. Время обновления определяется, таким образом, временем сканирования всех строк развертки от верхней до нижней, и составляет 1/60. Однако буфер кадра в растровых устройствах требует гораздо больше памяти, чем дисплейный буфер в векторных графических устройствах.

Растровое изображение в буфере кадра может содержать сведения о цвете, если каждой точке (пикселю) будет соответствовать не один бит, а несколько.

Рассмотрим пример с тремя битами на каждый пиксель. Буфер кадра может быть представлен тремя плоскостями, каждая из которых содержит по одному биту для каждого пикселя. В случае трехбитового представления цвета первый разряд может использоваться для включения или выключения красного, второй - зеленого, а третий - синего цвета. Так получается восемь цветов, которые могут быть одновременно выведены на экран графического устройства. ЦАПы, показанные на рисунке, - это цифроаналоговые преобразователи, выдающие аналоговый сигнал, управляющий электронной пушкой определенного цвета на основании значений битов соответствующей плоскости.

Трехбитовые плоскости

В настоящее время доминируют графические устройства с 24-битовыми плоскостями (по восемь битов на каждый из основных цветов). В таких устройствах каждый цвет может иметь 256 градаций (2^8), а всего возможно одновременно отобразить 16 777 216 (2^{24}) цветов. ЦАПы на рисунке работают с 8 разрядами вместо одного.

Двадцатичетырехбитовые плоскости

Программные компоненты

Любая программа, используемая в жизненном цикле продукта для сокращения времени и стоимости разработки продукта, а также для повышения его качества, может быть отнесена к классу CAD/CAM/CAE. В основе лежат программы

CAD, позволяющие конструктору создавать формы и манипулировать ими на мониторе в интерактивном режиме, сохраняя результаты в базе данных.

Типичные системы CAD/CAM/CAE

Область применения	Программы	Интегрированные системы
CAD: двумерные чертежи	CADAM, AutoCAD, MicroCADAM, VersaCAD	Pro/ENGINEER Unigraphics CATIA I-DEAS I/EMS
CAD: твердотельное моделирование	Solid Edge, SolidWorks, SolidDesigner, Mechanical Desktop	
CAM	BravoNCG, VERICUT, DUCT, Camand, Mastercam,	
CAE	MSC/NASTRAN, ANSYS, PATRANADAMS, C-MOLD,	

6. Основы фракталов.

Обратная связь и итерация; принцип обратной связи; основные типы процессов обратной связи; побочный эффект малых возмущений; устойчивость вычислений. Фрактал — это объект, отдельные элементы которого наследуют свойства родительских структур. Самыми известными фрактальными объектами являются деревья: от каждой ветки ответвляются меньшие, похожие на нее, от тех — еще меньшие и так далее. По отдельной ветке математическими методами можно проследить свойства всего дерева. Фрактальными свойствами обладают многие природные объекты: снежинка при увеличении оказывается фракталом; по фрактальным алгоритмам растут кристаллы и растения. Если посмотреть на береговую линию моря на картах все более крупного масштаба, то становятся видны все новые изгибы и изломы, похожие на более крупные.

Появление новых элементов меньшего масштаба происходит по простому алгоритму. Очевидно, что описать такой объект можно всего лишь несколькими математическими уравнениями!

Посмотрим, как строится простейший фрактал — «фрактальный треугольник» (его еще называют «снежинка Коха»).

Треугольники можно достраивать аналогичным образом до бесконечности. Мы можем получить объект любого уровня сложности, используя простейший

алгоритм. При этом, в отличие от векторной графики, ничего, кроме самих уравнений, занимающих несколько байт, в памяти компьютера хранить не надо! Вся информация, необходимая для воспроизведения этого фрактала, занимает всего лишь десятки байт! Естественно, возник вопрос — а можно ли сжать *любую* информацию, подобрав подходящий фрактальный алгоритм? Принципиально можно, и в западных странах активно ведутся работы в этом направлении. По непроверенным данным из интернета, разработанные сейчас фрактальные алгоритмы позволяют сжимать определенные типы файлов в 30 раз. Таким образом, фракталы являются интереснейшим объектом для изучения по двум основным причинам:

- фракталы являются одной из лучших моделей живой природы;
- их исследование открывает невиданные перспективы для сжатия информации.

Мало того, сейчас научились строить фракталы по простейшим формулам, например, $y = x^2 + C$.

Правда, простота формулы компенсируется сложностью алгоритма. Чтобы дальше наш разговор шел на одном языке, предлагаю небольшой экскурс в математику.

Отображением называется операция, ставящая в соответствие каждому элементу одного множества элемент другого множества. Например, отображение $x \rightarrow x^2$ ставит в соответствие числу 2 из множества действительных чисел число 4 из того же множества; числу 3 - число 9 и т. д.

Мнимая единица (**i**) — число, квадрат которого равен –1.
Комплексное число — число вида $a + b \cdot \mathbf{i}$, где a и b — действительные числа.
Комплексная плоскость — плоскость, на которой введена декартова система координат, в которой по оси абсцисс откладывается параметр a, по оси ординат — параметр b. Каждая точка комплексной плоскости представляет собой определенное комплексное число.

Алгоритм построения фрактала такой: выбираем точку на комплексной плоскости. Действуем на нее отображением $x \rightarrow x^2 + C$, в результате чего точка «перемещается» на плоскости. На полученную точку повторно действуем отображением, и так несколько раз (несколько *итераций*). Если в результате этого точка «убегает» на бесконечность — красим ее в белый цвет, если «прыгает» вокруг своего исходного положения — в черный. Повторяем описанные действия для всех точек плоскости.

Таким образом мы получим двуцветную картину, где черная фигура представляет собой фрактал, называемый множеством Жюлиа. Его форма может меняться в зависимости от коэффициента C.

Можно получить также многоцветный фрактал: точки, не «убегающие» на бесконечность, красим в цвет номер 1, убегающие за одну итерацию — в цвет 2, за две — в цвет 3 и так далее. Количество цветов выбираем сами.

7. Классические фракталы и самоподобие. Фрактальная графика.

Фракталы встречаются везде, где заканчиваются правильные формы евклидовой геометрии. Все, что создано человеком, ограничено плоскостями. Если встречается природный объект, то с первого взгляда видно, что осознать, описать его форму со всеми шероховатостями можно только приблизительно. Здесь на помощь приходят фракталы.

Определение фрактала, данное Мандельбротом, звучит так: *"Фракталом называется структура, состоящая из частей, которые в каком-то смысле подобны целому"*

Одним из основных свойств фракталов является самоподобие.

В самом простом случае небольшая часть фрактала содержит информацию обо всем фрактале.

Фрактал — это объект, отдельные элементы которого наследуют свойства родительских структур.

Самыми известными фрактальными объектами являются деревья: от каждой ветки ответвляются меньшие, похожие на нее, от тех — еще меньшие и так далее.

Фрактальными свойствами обладают многие природные объекты: снежинка при увеличении оказывается фракталом; по фрактальным алгоритмам растут кристаллы и растения. Если посмотреть на береговую линию моря на картах все более крупного масштаба, то становятся видны все новые изгибы и изломы, похожие на более крупные.

Появление новых элементов меньшего масштаба происходит по простому алгоритму. Очевидно, что описать такой объект можно всего лишь несколькими математическими уравнениями!

По отдельной ветке математическими методами можно проследить свойства всего дерева.

- **Отображением** называется операция, ставящая в соответствие каждому элементу одного множества элемент другого множества. Например, отображение $x \rightarrow x2$ ставит в соответствие числу 2 из множества действительных чисел число 4 из того же множества; числу 3 - число 9 и т. д.

- **Мнимая единица** (**i**) — число, квадрат которого равен –1.
- **Комплексное число** — число вида a + b•**i**, где a и b — действительные числа.
- **Комплексная плоскость** — плоскость, на которой введена декартова система координат, в которой по оси абсцисс откладывается параметр a, по оси ординат — параметр b. Каждая точка комплексной плоскости представляет собой определенное комплексное число.

Алгоритм построения фрактала

1. Выбираем точку на комплексной плоскости.
2. Действуем на нее отображением x → x2 + C, в результате чего точка «перемещается» на плоскости.
3. На полученную точку повторно действуем отображением, и так несколько раз (несколько *итераций*).
4. Если в результате этого точка «убегает» на бесконечность — красим ее в белый цвет, если «прыгает» вокруг своего исходного положения — в черный.
5. Повторяем описанные действия для всех точек плоскости.

Природные фракталы

Очевидные примеры – дерево, куст, колония кораллов. Еще более наглядным примером может служить соцветие «сложный зонтик» – «зонтик», состоящий, в свою очередь, из маленьких зонтиков.

Одним из известнейших природных фракталов является «Папоротник». Приведем алгоритм создания этого фрактала.

Строится этот фрактал по следующему алгоритму.

```
program fern;

uses crt,graph;

var

gd,gm:integer;

procedure draw;

const

iterations=50000;

var

t ,x,y,p:real;
```

```
κ:longint;
mid_x,mid_y,radius:integer;
begin
mid_x:=getmaxX div 2;
mid_y:=getmaxY;
radius:=trunc(0.l*mid_y);
randomize;
x:=l.0;
y:=0.0;
for k:=l to iterations do begin
p:=random;
t:=x;
if p<=0.85 then begin
x:=0.85 * x + 0.04 * y;
y:=-0.04 * t + 0.85 * y + 1.6
end else if p<=0.92 then begin
x:=0.2 * x - 0.26 * y;
y:=0.23 * t + 0.22 * y + 1.6
end else if p<=0.99 then begin
x:=-0.15 * x + 0.28 * y;
y:=0.26 * t + 0.24 * y + 0.44
end else begin
x:=0.0;
y: =0. 16 * y
end;
```

```
putpixel(midx+round(radius*x),mid_y-round(radius*y), light green)
end
end;
begin
gd:=detect;
initgraph(gd,gm,'c:\bp\bgi');
draw;
readkey;
closegraph
end.


program FracTree;
uses Graph, CRT;
procedure Tree(x, y: Integer; a: Real; 1: Integer);
var
```

Геометрические фракталы

Именно с них началась история фракталов. Это и есть те функции-монстры, которых так называли за не дифференцируемость в каждой точке. Геометрические фракталы являются также самыми наглядными, т.к. сразу видна самоподобность. Вообще все геометрические фракталы обладают т.н. жесткой самоподобностью, не изменяющейся при изменении масштаба. Для построения геометрических фракталов характерно задание "основы" и "фрагмента", повторяющегося при каждом уменьшении масштаба. Поэтому эти фракталы иногда называют конструктивными или автомодельными.

Примерами таких фракталов являются Треугольник Серпинского, Снежинка Коха, Кривая Коха, Кривые Леви и Минковского и многие другие. В графике геометрические фракталы применяются для получения изображений деревьев, кустов, береговых линий и т.д.

Построение триадной кривой Коха

1. Построение кривой начинается с единичного отрезка, который называется инициатором и является предфракталом 0-го порядка.
2. Далее инициатор заменяется на образующий элемент - кривую из четырех прямолинейных звеньев, каждое из которых имеет длину 1/3. Так образуется предфрактал 1-го порядка. Его длина равна 4/3.
3. Для построения предфрактала следующего порядка каждое звено заменяется на уменьшенный образующий элемент.
4. В результате получаем кривую, состоящую из 4 x 4 = 16 звеньев, каждое из которых имеет длину (1/3) / 3 = 1/9, общая длина равна 16/9. Длина предфрактала n-го порядка равна (4/3) в степени n.

Очевидно, что предел длины кривой при **n**, стремящийся к бесконечности, равен бесконечности. В итоге получили кривую бесконечной длины, заполняющую ограниченное множество на плоскости, что само по себе очень любопытно.

Следующая программа реализует данный метод.

```
program Koh;

uses CRT, Graph;

var

gd,gm : Integer;

const

iter = 50000;

procedure draw;

var

t, x, y, p : Real;

к : Longlnt;

mx, my, rad : Integer;

begin

mx := 10;

my := 250;

rad :=600;

Randomize;
```

```
x := 0.0;

y := 0.0;

for к := 1 To iter do begin

p := Random;

t := x;

if p <= 1/2 then begin

x := 1/2 * x + 1/(2*sqrt(3)) * y;

y := 1/(2*sqrt(3)) * t - 1/2 * y;

end

else

begin

x := 1/2 * x - l/(2*sqrt(3)) * y +1/2;

y := -l/(2*sqrt(3)) * t - 1/2 * y + 1/(2*sqrt(3))

end;

PutPixeKmx + Round(rad * x) , my - Round(rad * y) , 2);

end;

end;

begin

gd := Detect;

InitGraph(gd,gm,");

draw; ReadKey;

CloseGraph;

end.
```

Если построение кривой начинать не с отрезка, а с треугольника, и применить вышеперечисленные построения к каждой его стороне, то получим «снежинку» Коха.

Существуют два основных способа построения геометрических фракталов.

Первый способ - использование L-систем (от имени Lindenmayer), **второй способ** - применение системы IFS (iterated function systems).

L-система - это грамматика некоторого языка, которая описывает инициатор и преобразование, выполняемое над ним, при помощи средств, аналогичных средствам языка Лого (аксиоматическое описание простейших геометрических фигур и допустимых преобразований на плоскости и в пространстве).

L – системы (рис.77)

Рис. 83

Подобные L-системы применяются в пакете Autodesk 3D Studio для описания цветов и других растений.

Системы итерирующих функций (IFS)

Система итерирующих функций - это совокупность сжимающих аффинных преобразований.

Рассмотрим подробнее построение кривой Коха с использованием аффинных преобразований. Каждый новый элемент кривой содержит четыре звена, полученных из образующего элемента использованием масштабирования, поворота и переноса.

1. Для получения первого звена достаточно сжать исходный отрезок в три раза. Следует отметить, что то же масштабирование применяется для всех звеньев.
2. Следующее звено строится с использованием всех возможных преобразований, а именно: сжатие в три раза, поворот на - 60^o и параллельный перенос на 1/3 по оси X.
3. Третье звено строится аналогично второму: сжатие в три раза, поворот на 60^o, параллельный перенос на 2/3 по оси X.
4. Последнее звено: сжатие в три раза, параллельный перенос на 2/3 по оси X.

Теперь мы можем найти систему итерирующих функций для описания кривой Коха. Осталось только произвести суперпозицию аффинных преобразований - масштабирования, поворота и параллельного переноса.

Из курса линейной алгебры известна формула вычисления новых координат x',y' при аффинных преобразованиях:

x` = x * a - y * b + e

y` = x * c + y * d + f

Здесь

a = cos (alpha) * scale_x,

b = sin (alpha) * scale_x,

c = sin (alpha) * scale_y,

d = cos (alpha) * scale_y,

e = move_x,

f = move_y,

где

scale_x - масштабирование по оси X;

scale_y - масштабирование по оси Y;

alpha - угол поворота;

move_x - параллельный перенос по оси X;

move_y - параллельный перенос по оси Y.

Полученные коэффициенты a, b, c, d, e, f для каждого звена и составят требуемую систему итерирующих функций.

Вычислим коэффициенты аффинного преобразования IFS для кривой Коха.

1. Для первого звена коэффициенты аффинного преобразования будут следующими: a = 0.3333, b = 0.0000, c = 0.0000, d = 0.3333, e = 0.0000, f = 0.0000.

Полученное аффинное преобразование является масштабированием на коэффициент 1/3=0,3333.

2. Вычислим коэффициенты преобразования для второго звена: a = 0.1667, b = -0.2887, c = 0.2887, d = 0.1667, e = 0.3333, f = 0.0000.
3. Коэффициенты для третьего звена будут такими: a = -0.1667, b = 0.2887, c = 0.2887, d = 0.1667, e = 0.6666, f = 0.
4. 4. И наконец, коэффициенты преобразования для последнего звена: a = 0.3333, b = 0.0000, c = 0.0000, d = 0.3333, e = 0.6666, f = 0.0000.

Коэффициенты для первого и последнего звеньев кривой Коха практически идентичны и отличаются только параллельным переносом по оси X

(коэффициент **e**). Второе и третье преобразования включают в себя не только масштабирование и перенос, но и поворот на 60^o и -60^o.

Здесь коэффициенты вычисляются так:

0.1667 = cos (60) * 1/3,

-0.2887 = -sin (60) * 1/3

Значение последнего (седьмого) параметра каждого преобразования пропорционально площади, занимаемой звеном. Сумма последних параметров для всех преобразований равна единице. В нашем примере размеры звеньев равны, поэтому седьмой параметр равен 1/4=0.25 для всех звеньев

В том случае, если оценить приблизительно размеры не удается, можно использовать формулу вычисления площади p = abs (a * d - b * c). Следует учитывать то, что эта формула дает не нормализованный результат. Поэтому нам придется еще приводить сумму к единице.

Алгебраические фракталы

Вторая большая группа фракталов – алгебраические. Свое название они получили, за то, что их строят, используя простые алгебраические формулы.

Получают их с помощью нелинейных процессов в n-мерных пространствах. Известно, что нелинейные динамические системы обладают несколькими устойчивыми состояниями. Состояние, в котором окажется динамическая система после некоторого числа итераций, зависит от начальных условий. Поэтому каждое устойчивое состояние (аттрактор) обладает некоторой областью начальных состояний, при которых система обязательно перейдет в рассматриваемые конечные состояния. Таким образом, фазовое пространство разбивается на области притяжения аттракторов.

Самыми известными из них являются множества Мандельброта и Жюлиа, Бассейны Ньютона и т.д.

Фрактал Мандельброта назван по имени французского математика польского происхождения Бенуа Мандельброта, известного исследователя фракталов. Множество Мандельброта получают следующим образом: строится отображение $z'=f_c(z)$, где числа z и c - комплексные.

Алгоритм построения множества Мандельброта выглядит следующим образом:

1. выбираем значение c
2. z:=0 n:=0
3. z:=f(z)
4. inc(n)

Если не достигнуто максимальное значение итераций или значение z не превысило максимального значения, то повторять шаги 3 и 4.

Наиболее часто используется формула $z = z^2 + c$, так как она наиболее проста в применении. Раскладывая уравнение на действительную и мнимую часть, получаем следующее:

Re: $x^2 - y^2 + a$
Im: $2xy + b$,
приняв, что z = x + yi и c = a + bi.
Цвет обычно выбирают по числу итераций, но есть и другие способы.

Для всех точек на комплексной плоскости в некотором интервале вычисляем достаточно большое количество раз $z = z^2 + c$, каждый раз проверяя абсолютное значение z. Если это значение больше максимального, то рисуем точку с цветом, вычисляемым по количеству итераций, иначе рисуем точку черного цвета.

Черный цвет в середине показывает, что в этих точках функция стремится к нулю - это и есть множество Мандельброта. За пределами этого множества функция стремится к бесконечности. А самое интересное - это границы множества. Они то и являются фрактальными. Если рассмотреть увеличение, то можно увидеть самоподобие фигуры.

Стохастические фракталы

Кривая Коха как бы не была похожа на границу берега не может выступать в качестве ее модели из-за того, что она всюду одинакова, самоподобна, а в действительности это не так. Все природные объекты создаются по капризу природы, и есть случайность в этом процессе.

Фракталы при построении которых в итеративной системе случайным образом изменяются какие-либо параметры называются стохастичными. Термин "стохастичность" происходит от греческого слова, обозначающего "предположение".

Также примером случайности в природе является броуновское движение. С помощью компьютера такие процессы строить достаточно просто, т.к. он позволяет генерировать последовательности случайных чисел. Эти фракталы используются при моделировании рельефов местности и поверхности морей, процесса электролиза.

Построение ландшафта

Возьмем карту размером 512*512 точек, каждая из которых представляет собой 8-ми битное целое значение высоты и цвета. Первоначально изображение выглядит таким образом, что при вызове функции высоты в точках с координатами (u,v) она возвращает одинаковые значения в углах условной

сетки точек. Т. е. w(0,0)=w(512,0)=w(0, 512)=w(512, 512). w(1,1)=w(512, 512) и т. д.

Координаты поверхности:

Координаты (u,v) описывают положение на поверхности. Всю поверхность можно представить как множество дискретных значений функции высоты h=w(u,v).

Координаты экрана:

Положение точек на экране будем описывать координатами (x,y).

Построение поверхности:

Генерация фрактальных поверхностей, по другому, называется плазма или рекурсивное разбиение и выглядит следующим образом. В начале вычисляем произвольные высоты в углах и середине карты поверхности т. е. в точках: (0,0), (512,0), (0, 512), (512, 512). Затем запускаем подпрограмму вычисления высот, для которой в качестве параметров передаются размер и положение квадратного участка поверхности, в первом случае этим участком является вся карта. Подпрограмма считывает значения высот из углов квадрата полученного в качестве параметра. Вдоль каждого ребра квадрата вычисляется среднее значение двух высот по краям ребра и к полученному значению прибавляется некоторое произвольное значение, пропорциональное длине ребра, результат записывается в точку, делящую ребро пополам. Затем вычисляется высота центра квадрата путем нахождения средней высоты четырех высот углов квадрата, и к полученному значению добавляют произвольное число.

В 3-х мерном изображении этот процесс будет выглядеть следующим образом.

Для получения более гладких поверхностей используют метод «смазывания», который предполагает пересчет вершин поверхности по следующей формуле:

w(u,v)=k1*w(u,v)+k2*w(u+3,v-2)+k3*w(u2,v+4).

Причем k1, k2, k3 подбирают таким образом, чтобы k1+k2+k3=1.

Все вычисления производятся в фиксированных точках целочисленной арифметики.

8. Кодирование изображений с помощью простых преобразований.

Работая с компьютерной графикой, мы заинтересованы в том, чтобы уменьшить размер блока графических данных и таким образом поместить в заданное

физическое пространство больше информации. Сжатие можно применять и для того, чтобы помещать большие изображения в блок памяти заданного размера.

Сжатие – это процесс, применяемый для уменьшения физического размера блока информации.

Основными техническими характеристиками процессов сжатия и результатов их работы являются:

- степень сжатия (compress rating) или отношение (ratio) объемов исходного и результирующего потоков;

- скорость сжатия - время, затрачиваемое на сжатие некоторого объема информации входного потока, до получения из него эквивалентного выходного потока;

- качество сжатия - величина, показывающая на сколько сильно упакован выходной поток, при помощи применения к нему повторного сжатия по этому же или иному алгоритму.

Любой алгоритм реализующий сжатие или компрессию данных предназначен для снижения объема выходного потока информации в битах при помощи ее обратимого или необратимого преобразования. Поэтому, прежде всего, по критерию, связанному с характером или форматом данных, все способы сжатия можно разделить на две категории: обратимое и необратимое сжатие (с потерями и без потерь).

Необратимое сжатие

Под необратимым сжатием подразумевают такое преобразование входного потока данных, при котором выходной поток, основанный на определенном формате информации, является достаточно похожим по внешним характеристикам на входной поток, однако отличается от него объемом.

Обратимое сжатие

В обратимых алгоритмах кодирование, как процесс, можно рассматривать со статистической точки зрения, что еще более полезно не только для построения алгоритмов сжатия, но и для оценки их эффективности. Для всех обратимых алгоритмов существует понятие стоимости кодирования. Под стоимостью кодирования понимается средняя длина кодового слова в битах. Избыточность кодирования равна разности между стоимостью и энтропией кодирования, а хороший алгоритм сжатия всегда должен минимизировать избыточность (напомним, что под энтропией информации понимают меру ее неупорядоченности.). Фундаментальная теорема Шеннона о кодировании информации говорит о том, что "стоимость кодирования всегда не меньше энтропии источника, хотя может быть сколь угодно близка к ней". Поэтому, для

любого алгоритма, всегда имеется некоторый предел степени сжатия, определяемый энтропией входного потока.

Общие положения алгоритмов сжатия изображений

Изображения (как и видео) занимают намного больше места в памяти, чем текст. Так, скромная, не очень качественная иллюстрация на обложке книги размером 500x800 точек, занимает 1.2 Мб - столько же, сколько художественная книга из 400 страниц (60 знаков в строке, 42 строки на странице). Эта особенность изображений определяет актуальность алгоритмов архивации графики.

Второй особенностью изображений является то, что человеческое зрение при анализе изображения оперирует контурами, общим переходом цветов и сравнительно нечувствительно к малым изменениям в изображении.

Таким образом, мы можем создать эффективные алгоритмы архивации изображений, в которых декомпрессированное изображение не будет совпадать с оригиналом, однако человек этого не заметит.

Данная особенность человеческого зрения позволила создать специальные алгоритмы сжатия, ориентированные только на изображения.

Характер использования изображений задает степень важности следующих ниже противоречивых требований к алгоритму:

- **Высокая степень компрессии.** Заметим, что далеко не для всех приложений актуальна высокая степень компрессии. Кроме того, некоторые алгоритмы дают лучшее соотношение качества к размеру файла при высоких степенях компрессии, однако проигрывают другим алгоритмам при низких степенях.
- **Высокое качество изображений.** Выполнение этого требования напрямую противоречит выполнению предыдущего...
- **Высокая скорость компрессии**. Это требование для некоторых алгоритмов с потерей информации является взаимоисключающим с первыми двумя. Интуитивно понятно, что чем больше времени мы будем анализировать изображение, пытаясь получить наивысшую степень компрессии, тем лучше будет результат.
- **Высокая скорость декомпрессии**. Достаточно универсальное требование, актуальное для многих приложений. Однако можно привести примеры приложений, где время декомпрессии далеко не критично.
- **Масштабирование изображений**. Данное требование подразумевает легкость изменения размеров изображения до размеров окна активного приложения.

- **Возможность показать огрубленное изображение (низкого разрешения), использовав только начало файла.** Данная возможность

актуальна для различного рода сетевых приложений, где перекачивание изображений может занять достаточно большое время, и желательно, получив начало файла, корректно показать preview.

- **Устойчивость к ошибкам**. Данное требование означает локальность нарушений в изображении при порче или потере фрагмента передаваемого файла. Данная возможность используется при широковещании (broadcasting — передача по многим адресам) изображений по сети, то есть в тех случаях, когда невозможно использовать протокол передачи, повторно запрашивающий данные у сервера при ошибках.
- **Учет специфики изображения**. Более высокая степень архивации для класса изображений, которые статистически чаще будут применяться в нашем приложении.
- **Редактируемость.** Под редактируемостью понимается минимальная степень ухудшения качества изображения при его повторном сохранении после редактирования. Многие алгоритмы с потерей информации могут существенно испортить изображение за несколько итераций редактирования.
- **Небольшая стоимость аппаратной реализации**. Эффективность программной реализации. Данные требования к алгоритму реально предъявляют не только производители игровых приставок, но и производители многих информационных систем.

Представление изображений конечным объемом данных

Компьютерное изображение в его цифровом представлении является набором значений интенсивностей светового потока, распределенных по конечной площади, имеющей обычно прямоугольную форму. Поток данных об изображении имеет существенное количество излишней информации, которая может быть устранена практически без заметных для глаза искажений.

Существует два типа избыточности:

- Статистическая избыточность, связанная с корреляцией и предсказуемостью данных. Эта избыточность может быть устранена без потери информации, исходные данные при этом могут быть полностью восстановлены.
- Визуальная (субъективная) избыточность, которую можно устранить с частичной потерей данных, мало влияющих на качество воспроизводимых изображений; это - информация, которую можно изъять из изображения, не нарушая визуально воспринимаемое качество изображений.

Статистическая избыточность изображений. Пусть имеется дискретизированное M x N пикселей и квантованное с точностью K бит на пиксель монохромное изображение. Следовательно, для хранения этого изображения необходимо M x N x K бит информации. Если предположить, что квантованные значения

яркости не равновероятны, то уменьшение информации возможно путем изменения количества бит информации для кодирования пикселей: более вероятные кодируются словами с меньшим количеством бит, менее вероятные - с большим.

Этот метод называется кодированием словами переменной длины или энтропийным кодированием.

Пусть квантованный уровень яркости z имеет вероятность *P(z)* и ему присваивается слово - код длины *L(z)* бит. Тогда средняя длина кода для всего изображения составит

бит на пиксель. Нижняя граница для определяется информационной теоремой и называется энтропией случайной величины:

Таким образом, энтропия - это мера количества информации, которую несет случайная величина уровня яркости z.

$H(f) \geq 0$, поскольку $P(z) \in [0,1]$.

Из формулы *H(f)* вытекает, что чем более неравномерно распределение *P(z)*, тем меньше энтропия и тем эффективнее может быть энтропийное кодирование.

Устранение визуальной избыточности изображений является основным резервом сокращения передаваемой информации. Для оптимизации процесса кодирования с точки зрения обеспечения передачи наименьшего объема информации необходимо, с одной стороны, не передавать избыточную информацию, а, с другой, - не допустить чрезмерной потери качества изображения.

До сих пор не существует простой и адекватной модели визуального восприятия изображений, пригодной для оптимизации их кодирования.

Наиболее известные методы эффективного кодирования символов основаны на знании частоты каждого символа присутствующего в сообщении. Зная эти частоты, строят таблицу кодов, обладающую следующими свойствами:

- различные коды могут иметь различное количество бит;
- коды символов с большей частотой встречаемости, имеют больше бит, чем коды символов с меньшей частотой;
- хотя коды имеют различную битовую длину, они могут быть восстановлены единственным образом.

Этими свойствами обладает известный алгоритм Хаффмана.

Алгоритмы архивации без потерь

Алгоритм RLE

Алгоритм рассчитан на деловую графику — изображения с большими областями повторяющегося цвета. Ситуация, когда файл увеличивается, для этого простого алгоритма не так уж редка. Ее можно легко получить, применяя групповое кодирование к обработанным цветным фотографиям. Для того, чтобы увеличить изображение в два раза, его надо применить к изображению, в котором значения всех пикселей больше двоичного 11000000 и подряд попарно не повторяются.

Алгоритм LZW

Название алгоритм получил по первым буквам фамилий его разработчиков — Lempel, Ziv и Welch. Сжатие в нем, в отличие от RLE, осуществляется уже за счет одинаковых цепочек байт.

Процесс сжатия выглядит достаточно просто. Мы считываем последовательно символы входного потока и проверяем, есть ли в созданной нами таблице строк такая строка. Если строка есть, то мы считываем следующий символ, а если строки нет, то мы заносим в поток код для предыдущей найденной строки, заносим строку в таблицу и начинаем поиск снова. Особенность LZW заключается в том, что для декомпрессии нам не надо сохранять таблицу строк в файл для распаковки. Алгоритм построен таким образом, что мы в состоянии восстановить таблицу строк, пользуясь только потоком кодов.

Алгоритм Хаффмана

Каждая строка изображения сжимается независимо. Считается, что в изображении существенно преобладает белый цвет, и все строки изображения начинаются с белой точки. Если строка начинается с черной точки, то считаем, что строка начинается белой серией с длиной 0. На практике в тех случаях, когда в изображении преобладает черный цвет, инвертируется изображение перед компрессией и записываем информацию об этом в заголовок файла.

Близкая модификация алгоритма используется при сжатии черно-белых изображений (один бит на пиксель). Полное название данного алгоритма CCITT Group 3. Это означает, что данный алгоритм был предложен третьей группой по стандартизации Международного Консультационного Комитета по Телеграфии и Телефонии (Consultative Committee International Telegraph and Telephone). Последовательности подряд идущих черных и белых точек в нем заменяются числом, равным их количеству. А этот ряд, уже в свою очередь, сжимается по Хаффману с фиксированной таблицей.

JBIG

Алгоритм разработан группой экспертов ISO (Joint Bi-level Experts Group) специально для сжатия однобитных черно-белых изображений. Например,

факсов или отсканированных документов. В принципе, может применяться и к 2-х, и к 4-х битовым картинкам. При этом алгоритм разбивает их на отдельные битовые плоскости. JBIG позволяет управлять такими параметрами, как порядок разбиения изображения на битовые плоскости, ширина полос в изображении, уровни масштабирования. Последняя возможность позволяет легко ориентироваться в базе больших по размерам изображений, просматривая сначала их уменьшенные копии.

Lossless JPEG

Этот алгоритм разработан группой экспертов в области фотографии (Joint Photographic Expert Group). В отличие от JBIG, Lossless JPEG ориентирован на полноцветные 24-битные или 8-битные в градациях серого изображения без палитры. Он представляет собой специальную реализацию JPEG без потерь. Степени сжатия от 1 до 20. Lossless JPEG рекомендуется применять в тех приложениях, где необходимо побитовое соответствие исходного и декомпрессированного изображений.

Алгоритмы архивации с потерями

Алгоритм JPEG

JPEG— один из самых достаточно мощных алгоритмов. Практически он является стандартом де-факто для полноцветных изображений. Оперирует алгоритм областями 8x8, на которых яркость и цвет меняются сравнительно плавно. Вследствие этого, при разложении матрицы такой области в двойной ряд по косинусам значимыми оказываются только первые коэффициенты. Таким образом, сжатие в JPEG осуществляется за счет плавности изменения цветов в изображении.

Фрактальный алгоритм

Фрактальная архивация основана на том, что мы представляем изображение в более компактной форме — с помощью коэффициентов системы итерируемых функций (Iterated Function System— далее по тексту как IFS). Строго говоря, IFS представляет собой набор трехмерных аффинных преобразований, в нашем случае переводящих одно изображение в другое. Преобразованию подвергаются точки в трехмерном пространстве (x_координата, y_координата, яркость).В худшем случае, если не будет применяться оптимизирующий алгоритм, потребуется перебор и сравнение всех возможных фрагментов изображения разного размера. Даже для небольших изображений при учете дискретности мы получим астрономическое число перебираемых вариантов. Причем, даже резкое сужение классов преобразований, например, за счет масштабирования только в определенное количество раз, не дает заметного выигрыша во времени. Кроме того, при этом теряется качество изображения. Подавляющее большинство исследований в области фрактальной компрессии сейчас направлены на

уменьшение времени архивации, необходимого для получения качественного изображения.

3. Методические рекомендации для студентов по изучению курса

В методических указаниях излагается порядок выполнения лабораторных работ.

Курс включает 36 часов лабораторных занятий. Перед выполнением работы преподаватель знакомит студентов с интерфейсом того редактора, в котором они будут работать, если же занятие не первое, то с теми кнопками и меню, которые чаще всего будут задействованы в работе. Для получения зачета по лабораторной работе студент должен предоставить отчет по лабораторной работе в виде файла, сохраненного в своей папке, а также ответить на вопросы преподавателя, связанные с ходом выполнения работы и получаемыми результатами. Все лабораторные работы должны выполняться во время аудиторных занятий в компьютерном классе. Результаты работы нужно сохранять в собственной папке «ФИО», которая должна находиться в папке «Компьютерная графика» на сетевом диске, имеющемся в классе.

Перечень лабораторных работ:

1. Photoshop: 6 часов.
2. GIMP: 6 часов
3. Inkscape: 6 часов.
4. 3ds MAX: 6 часов.
5. Blender: 6 часов.
6. Фрактальная графика: 6 часа.

Примеры лабораторных работ:

Лабораторная работа № 7

Тема: **Цветовая коррекция в GIMP**

Цель: Овладение навыками тоновой и цветовой коррекции.

Методические указания к лабораторной работе взяты с сайта www.progimp.ru.

Ход выполнения работы:

1.Открываем исходное изображение «Рисунок 1»:

2. Откройте **Цвет->Яркость-Контраст...** и сделаейте приблизительно такие настройки:

3. Далее откройте диалог **Цвет->Кривые...** и с помощью s-образной кривой уменьшайте яркость в тенях и немного увеличивайте в светах:

4. Теперь продублируйте получившийся слой. Перейдите в **Цвет->Тонировать...** и передвиньте ползунок **Насыщенность** до 0. Верхний слой получится черно-белым. Понизьте его непрозрачность. Эта установка делается по вкусу. Здесь выбрано значение 60%:

5. Второй вариант конечной обработки может быть таким. Непрозрачность оставьте около 75%. Затем перейдите в **Цвет->Тонировать...** и измените настройки по желанию.

Сравните исходное изображение с полученным результатом.

Контрольные вопросы

1. Как и для чего вызывается команда Цвет->Яркость-Контраст?
2. Как и для чего вызывается команда Цвет->Кривые?
3. Как и для чего вызывается команда Цвет->Тонировать?

Лабораторная работа № 7

Тема: **Цветовая коррекция в GIMP**

Цель: Овладение умениями работы с базовыми инструментами Inkscape: «Перо», «Эллипс», «Прямоугольник», «Градиент», «Трансформация», «Сумма» и «Разность контуров», а также умениями работы с цветовой палитрой Inkscape.

Методические указания к лабораторной работе взяты с сайта www.openarts.ru.

Ход выполнения работы:

1. Используйте инструмент Inscape *Каллиграфическое перо (Ctrl+F6)* для рисования веток.

2. Выделите все ветки и при помощи операции *Контур / Сумма (Ctrl++)* объедините их в единый объект. Теперь нарисуйте небольшой круг и поместите его в левом верхнем углу по отношению к дереву. Для того,

чтобы круг получился правильным, можете удерживать клавишу *Ctrl* во время его создания.

3. Выделите круг, выберите в меню Inscape *Правка / Клонирование / Создать узор из клонов*. Здесь советую поэкспериментировать со значениями, пока Вы не достигните необходимого результата. Например, задайте следующие параметры для вкладки *Смещение*: значение экспоненты *0.70* для обоих осей, а в столбце *Случайно* в обоих полях введите значение 50 процентов. Остальные поля также заполните на свое усмотрение. Вы в любой момент можете нажать на кнопку *Создать* и посмотреть на результат, а если он Вас не устроит - не закрывая диалогового окна нажать на комбинацию *Ctrl+Z* для отмены и продолжить экспериментировать.
   Кстати говоря, уже после создания узора из клонов Вы можете выделить первоначальный круг (который послужил основой) и изменить его параметры. В итоге все созданные клоны будут также изменены.

4.Выделите дерево и нажмите комбинацию *Ctrl+L* несколько раз. В результате упрощения наше дерево приобретет более округлые очертания. Также можно добавить градиент как на рисунке ниже.

Пример создания 3D объектов

Чтобы сделать изображение с 3D объектом в Inkscape, потребуется совсем немного инструментов - трансформация, сумма и разность контуров, и, что очень важно, градиент. В этом уроке техники применены к базовым контурам, тем не менее, полученный результат выглядить весьма эффектно.

Урок включает в себя 8 простых шагов, использовалась версия Inkscape 0.46.

Всего лишь 3 объекта понадобится, чтобы сделать в Inkscape это 3D кольцо.

При выполнении этого урока можно легко запутаться, поскольку множество раз будет выполняться дублирование объектов и различные операции над ними. Поэтому с самого начала будьте очень внимательны. Если описание покажется запутанным - смотрите на картинки - в них разобраться достаточно просто.

1. Рисуем в Inkscape эллипс *(F5)*, для удобства задаем значение непрозрачности для него в 75 процентов. Дублируем полученный эллипс *(Ctrl+D)* и уменьшаем его размер при помощи инструмента трансформации *(F1)*,

удерживая клавиши *Shift+Ctrl*.

2. Дублируем полученную группу из двух эллипсов *(Ctrl+D)* и на время перемещаем ее немного в сторону. Возвращаемся к первоначальной группе и снова дублируем ее. Зажав клавишу *Ctrl*, перемещаем дубликат немного вверх (как на рисунке).

3. Возвращаемся к группе из двух эллипсов, которую мы переместили в сторону, снова дублируем ее (Ctrl+D) и опять перемещаем в сторону. Затем выполняем операцию *Контур / Разность (Ctrl+-)*. В результате имеем объект, изображенный ниже (условно назовем его *Объект 1*).

4. Выделяем два маленьких эллипса, полученных во 2-м шаге урока, дублируем их и в который раз перемещаем немного в сторону. С ними также выполняем операцию *Контур / Разность (Ctrl+-)*. В результате имеем объект, изображенный ниже (условно назовем его *Объект 2*).

5. Выделяем два больших эллипса, полученных во 2-м шаге урока, дублируем их и перемещаем в сторону. Теперь нам нужно нарисовать прямоугольник. Его ширина будет равна ширине больших эллипсов, а высота - расстоянию между двумя эллипсами. Выделяем нижний эллипс и прямоугольник, после чего выбираем в меню Inkscape *Контур / Сумма (Ctrl++)*. Перемещаем этот объект вниз.
   Выбираем оба этих объекта и выполняем *Контур / Разность (Ctrl+-)*. В результате имеем объект, изображенный ниже (условно назовем его *Объект 3*).

6. Следующий этап урока, который нам нужно выполнить - это собрать все 3 полученных объекта вместе. Сделать это несложно, увеличив масштаб отображения и используя диалог выравнивания *(Shift-Ctrl-A)*.

7. Следующее, что нужно сделать в этом уроке - немного раскрасить наши объекты при помощи инструмента *Градиент* в Inkscape.
   Для первого объекта можно оставить плоскую заливку.
   Для второго объекта нам нужно назначить линейный градиент *темный-светлый-темный* (как на рисунке ниже).

Для третьего объекта нам нужно назначить линейный градиент *светлый-темный-светлый* (как на рисунке ниже). Значение непрозрачности для всех трех объектов - 100 процентов.

После всех вышеприведенных действий между объектами может наблюдаться небольшая щель. Чтобы исправить это, назначаем обводке объекта те же самые значения цвета и градиента, что и для заливки.

Точно таким же образом можно создавать трехугольные и четырехугольные, да и какие угодно другие формы.

4. Дидактические материалы

4.1. Фрактальное сжатие (материал для самостоятельного изучения)

Построение алгоритма фрактального сжатия

Основные определения, применяемые во фрактальном алгоритме сжатия

Определение: Преобразование , представимое в виде

(5)

где a, b, c, d, e, f - действительные числа и - называется двумерным аффинным преобразованием.

Определение: Преобразование представимое в виде

(6)

где a, b, c, d, e, f, p, q, r, s, t, u - действительные числа и - называется трехмерным аффинным преобразованием.

Определение: Пусть - преобразование в пространстве . Точка такая, что называется неподвижной точкой (аттрактором) преобразования.

Определение: Преобразование в метрическом пространстве называется сжимающим, если существует число , такое, что , где .

Замечание: Формально мы можем использовать любое сжимающее отображение при фрактальной компрессии, но реально используются лишь трехмерные аффинные преобразования с достаточно сильными ограничениями на коэффициенты.

Теорема. (О сжимающем преобразовании)

Пусть - сжимающее преобразование в полном метрическом пространстве . Тогда существует в точности одна неподвижная точка этого преобразования, и для любой точки последовательность сходится к .

Определение: Изображением называется функция , определенная на единичном квадрате и принимающая значения от 0 до 1 или .

Пусть трехмерное аффинное преобразование записано в виде

(7)

и определено на компактном подмножестве декартова квадрата (мы пользуемся особым видом матрицы преобразования, чтобы уменьшить размерность области определения с до). Тогда оно переведет часть поверхности в область ., расположенную со сдвигом (e, f) и поворотом, заданным матрицей

(8)

При этом, если интерпретировать значения функции как яркость соответствующих точек, она уменьшится в p раз (преобразование обязано быть сжимающим) и изменится на сдвиг q.

Определение: Конечная совокупность W сжимающих трехмерных аффинных преобразований , определенных на областях , таких, что и , называется системой итерируемых функций (IFS).

Алгоритм фрактального сжатия

Системе итерируемых функций однозначно сопоставляется неподвижная точка - изображение. Таким образом, процесс компрессии заключается в поиске коэффициентов системы, а процесс декомпрессии - в проведении итераций системы до стабилизации полученного изображения (неподвижной точки IFS). На практике бывает достаточно 7-16 итераций. Области в дальнейшем будут именоваться ранговыми, а области - доменными.

Итак, пусть изображение разбито на N ранговых областей , для каждой из которых найден соответствующий домен и преобразование , задаваемое коэффициентами , такое что для каждого существует такое что . Причём преобразования должны являться сжимающими, т.е. такими, что для всех выполняется

(9)

где . Из преобразований сформируем отображение , переводящее изображения в изображение

(10)

Следует учесть, что преобразования действуют только на соответствующие домены изображения . Доказано, что если преобразования являются сжимающими, то и отображение также является сжимающим

Как уже стало очевидным из изложенного выше, основной задачей при компрессии фрактальным алгоритмом является нахождение соответствующих аффинных преобразований. В самом общем случае мы можем переводить любые по размеру и форме области изображения, однако в этом случае получается астрономическое число перебираемых вариантов разных фрагментов, которое невозможно обработать на текущий момент даже на суперкомпьютере.

В алгоритме сделаны следующие ограничения на области:

Все области являются квадратами со сторонами, параллельными сторонам изображения. Это ограничение достаточно жесткое. Фактически мы собираемся аппроксимировать все многообразие геометрических фигур лишь квадратами.

При переводе доменной области в ранговую уменьшение размеров производится ровно в два раза. Это существенно упрощает как компрессор, так и декомпрессор, т.к. задача масштабирования небольших областей является нетривиальной.

Все доменные блоки — квадраты и имеют фиксированный размер. Изображение равномерной сеткой разбивается на набор доменных блоков.

Доменные области берутся «через точку» и по X, и по Y, что сразу уменьшает перебор в 4 раза.

При переводе доменной области в ранговую поворот куба возможен только на 0°, 90°, 180° или 270°. Также допускается зеркальное отражение. Общее число возможных преобразований (считая пустое) — 8.

Эти ограничения позволяют:

Построить алгоритм, для которого требуется сравнительно малое число операций даже на достаточно больших изображениях.

Очень компактно представить данные для записи в файл. Нам требуется на каждое аффинное преобразование в IFS:

три бита для того, чтобы задать преобразование симметрии при переводе доменного блока в ранговый;

7-9 бит для того, чтобы задать сдвиг по яркости при переводе.

Информацию о размере блоков можно хранить в заголовке файла. Таким образом, мы затратили менее 4 байт на одно аффинное преобразование. В зависимости от того, каков размер блока, можно высчитать, сколько блоков будет в изображении. Таким образом, мы можем получить оценку степени компрессии.

Отрицательные стороны предложенных ограничений:

Поскольку все области являются квадратами, невозможно воспользоваться подобием объектов, по форме далеких от квадратов (которые встречаются в реальных изображениях достаточно часто.)

Аналогично мы не сможем воспользоваться подобием объектов в изображении, коэффициент подобия между которыми сильно отличается от 2.

Алгоритм не сможет воспользоваться подобием объектов в изображении, угол между которыми не кратен 90°.

Такова плата за скорость компрессии и за простоту упаковки коэффициентов в файл. Сам алгоритм упаковки сводится к перебору всех доменных блоков и подбору для каждого соответствующего ему рангового блока.

Для каждого рангового блока делаем его проверку со всеми возможными доменными блоками (в том числе с прошедшими преобразование симметрии). Находим вариант с наименьшей мерой L2 (наименьшим среднеквадратичным отклонением) и сохраняем коэффициенты этого преобразования в файл. Коэффициенты — (1) это координаты найденного блока, (2) число от 0 до 7, характеризующее преобразование симметрии (поворот, отражение блока), и (3) сдвиг по яркости для этой пары блоков. Сдвиг по яркости вычисляется как:

(11)

где - значения пикселей рангового блока (R), а - значения пикселей доменного блока (D). При этом мера считается как:

(12)

Мы не вычисляем квадратного корня из L2 меры и не делим ее на n, поскольку данные преобразования монотонны и не помешают нам найти экстремум, однако мы сможем выполнять на две операции меньше для каждого блока. Таким образом, нам удалось уменьшить число операций алгоритма компрессии до вполне вычисляемых величин.

Схема алгоритма декомпрессии

Декомпрессия алгоритма фрактального сжатия чрезвычайно проста. Необходимо провести несколько итераций трехмерных аффинных преобразований, коэффициенты которых были получены на этапе компрессии.

В качестве начального может быть взято абсолютно любое изображение (например, абсолютно черное), поскольку соответствующий математический аппарат гарантирует нам сходимость последовательности изображений, получаемых в ходе итераций IFS, к неподвижному изображению (близкому к исходному). Обычно для этого достаточно 16 итераций.

Декомпрессия изображения

Поскольку мы записывали коэффициенты для блоков (которые, как мы оговорили, в нашем частном случае являются квадратами одинакового размера) последовательно, то получается, что мы последовательно заполняем изображение по квадратам сетки разбиения использованием аффинного преобразования.

Как можно подсчитать, количество операций на один пиксель изображения в градациях серого при восстановлении необычайно мало (N операций сложения «+» и N операций умножения «•», где N — количество итераций, т.е. 7-16). Благодаря этому, декомпрессия изображений для фрактального алгоритма проходит быстрее декомпрессии, например, для алгоритма JPEG. В простой реализации JPEG на точку приходится 64 операции сложения «+» и 64 операции умножения «•». При реализации быстрого ДКП можно получить, 7 сложений и 5 умножений на точку, но это без учета шагов RLE, квантования и кодирования по Хаффману. При этом для фрактального алгоритма умножение происходит на рациональное число, одно для каждого блока. Это означает, что мы можем, во-первых, использовать целочисленную рациональную арифметику, которая быстрее арифметики с плавающей точкой. Во-вторых, можно использовать умножение вектора на число — более простую и быструю операцию, часто закладываемую в архитектуру процессора (процессоры SGI, Intel MMX, векторные операции Athlon и т.д.). Для полноцветного изображения ситуация качественно не изменяется, поскольку перевод в другое цветовое пространство используют оба алгоритма.

Оценка потерь и способы их регулирования

При кратком изложении упрощенного варианта алгоритма были пропущены многие важные вопросы. Например, что делать, если алгоритм не может подобрать для какого-либо фрагмента изображения подобный ему? Достаточно очевидное решение — разбить этот фрагмент на более мелкие, и попытаться поискать для них. В то же время понятно, что эту процедуру нельзя повторять до бесконечности, иначе количество необходимых преобразований станет так велико, что алгоритм перестанет быть алгоритмом компрессии. Следовательно, мы допускаем потери в какой-то части изображения.

Для фрактального алгоритма компрессии, как и для других алгоритмов сжатия с потерями, очень важны механизмы, с помощью которых можно будет регулировать степень сжатия и степень потерь. К настоящему времени разработан достаточно большой набор таких методов. Во-первых, можно ограничить количество аффинных преобразований, заведомо обеспечив степень сжатия не ниже фиксированной величины. Во-вторых, можно потребовать, чтобы в ситуации, когда разница между обрабатываемым фрагментом и наилучшим его приближением будет выше определенного порогового значения, этот фрагмент дробился обязательно (для него обязательно заводится несколько «линз»). В-третьих, можно запретить дробить фрагменты размером меньше, допустим, четырех точек. Изменяя пороговые значения и приоритет этих условий, мы будем очень гибко управлять коэффициентом компрессии изображения в диапазоне от побитового соответствия до любой степени сжатия. Заметим, что эта гибкость будет гораздо выше, чем у ближайшего «конкурента»— алгоритма JPEG.

4.2. Вопросы к экзамену

1. Основные виды проецирования.
2. Свойства ортогонального проецирования.
3. Точка. Способы задания точки на плоскости и в пространстве.
4. Линия. Способы задания линии на плоскости и в пространстве.
5. Плоскость. Способы графического задания плоскостей
6. Образование и задание поверхности на чертеже.
7. Образование и задание поверхностей вращения.
8. Образование и задание линейчатых поверхностей.
9. Аксонометрические проекции. Виды.
10. Аксонометрические проекции 3-мерных тел.
11. Представление данных. Преобразования в двухмерном пространстве.
12. Представление данных. Преобразования в трехмерном пространстве.
13. Аффинное проецирование.
14. Перспективное проецирование.
15. Стереографическая и специальные перспективные проекции.
16. Математические тесты.
17. Математические отношения объектов.
18. Масштабирование в окне.
19. Нахождение параметров плоскости.
20. Состав видеосистемы ПК.
21. Форматы графических файлов.
22. Цвет. Цветовые модели.
23. Виды компьютерной графики. Растровая графика.
24. Виды компьютерной графики. Векторная графика.
25. Виды компьютерной графики. Фрактальная графика.
26. Графические редакторы.
27. Аффинные преобразования на плоскости.
28. Однородные координаты точки.
29. Аффинные преобразования в пространстве.

30. Растровая развертка отрезка.
31. Алгоритм Брезенхейма.
32. Алгоритм Сазерленда - Коэна
33. Закраска области, заданной цветом границы.
34. Сжатие изображений.
35. Общие положения алгоритмов сжатия изображений
36. Представление изображений конечным объемом данных
37. Алгоритмы архивации без потерь
38. Алгоритмы архивации с потерями
39. История развития компьютерной графики

6. Глоссарий

Adobe

Пакет программ компьютерной графики широкого применения. Наиболее известными программами являются A Photoshop, A Illustrator, Premiere, After Effects, FrameMaker, PageMaker, PageMill. Программы пакета поддерживают многослойную структуру изображений, содержат инструменты для создания и редактирования векторных контуров.

Adobe Photoshop

Одна из лучших программ компьютерной графики. Назначение программы:

- Рисование с помощью различных инструментов;
- Управление режимами изображений;
- Корректировка цветов;
- Деформирование, вращение, перемещение;
- Работа со слоями (напоминает работу с векторной графикой);
- Выделение фрагментов и работа с ними;
- Изменение изображений за счет использования фильтров;
- Ввод и редактирование текста.

Adobe Illustrator

Векторный пакет фирмы Adobe разработан для Macintosh, PowerMacintosh и Windows. Предназначен для создания иллюстраций и разработки общего дизайна страниц и ориентирован на вывод готовых изображений с высоким разрешением. Позволяет создавать фигуры и символы произвольной формы, а затем масштабировать, вращать и деформировать их. Кроме того, AI содержит широкий спектр инструментов для работы с текстом и многостраничными документами.

http://www.mgopu.ru/PVU/2.1/graphics/glossary.htm - Начало **Альфа буфер**

Дополнительный буфер, в котором содержится информация о прозрачности. С учетом A система RGB преобразуется к RGBA: Red, Green, Blue, Alpha. В 32-разрядном буфере содержится 24 бита информации о цвете: 8 бит на каждый из цветов (красный, зеленый и синий), и 8 бит на значение alpha.

Векторная графика

Метод построения изображений на экране монитора, при котором в максимальной степени используется математическое описание. Например, для о писания отрезка прямой достаточно указать координаты его концов, а окружность можно описать, задав координаты центра и радиус.

Визуализация

Выполняемый компьютером процесс преобразования математического описания объектов сцены (компьютерной модели) в форму, пригодную для непосредственного отображения этой сцены видеосистемой компьютера.

Видимый свет

Представляет собой спектральное распределение электромагнитной энергии с длинами волн в диапазоне 400--700 нм.

Видеопамять

Буферное запоминающее устройство, в котором хранится информация о текущем состоянии экрана дисплея.

Дисплей

То же, что Монитор (см.)

Интерфейс программы

Программа, работающая с операционной системой класса Windows, обычно имеет следующие элементы интерфейса: главное меню, панели инструментов, кнопки, флажки, переключатели, списки ввода, поля ввода. Эти и другие стандартные элементы можно изучить, используя любой учебник по основам информатики. Графические программы имеют следующие элементы интерфейса (инструменты): выделение объектов, стирание и закраска, изменение масштаба, инструменты для рисования, ввод и редактирование текста, примитивы.

Компьютерная графика

Область компьютерной технологии, которая изучает методы получения изображений в случае, когда исходной является информация, создаваемая пользователем, или получаемая на основе невизуальных данных.

Колбочки

Фоторецепторы глаза, сосредоточенные главным образом в центральной ямке и чувствительные к спектральному составу света. От К. зависит в первую очередь восприятие цвета.

Кратковременная память

Аналогом кратковременной памяти человека является ОЗУ. Ее назначение состоит в получении и обработке информации извне, и в извлечении информации из долговременной памяти и выработке адекватных реакций. Объем кратковременной памяти составляет 7 элементов. Для обеспечения лучшего запоминания, необходимо группировать информацию так, чтобы получалась иерархическая структура с объемом не более семи элементов на каждом уровне.

Монитор

Монитор - устройство отображения текстовой и графической информации. Различают М на основе электронно-лучевой трубки, жидкокристаллические и плазменные.

Насыщенность

Насыщенность - параметр, определяющий степень чистоты цвета. Чем ближе цвет к монохроматическому, тем более он насыщен.

Обработка изображений

Область компьютерной технологии, где рассматриваются задачи, в которых и входные и выходные данные являются изображениями.

Палочки

Фоторецепторы глаза, не обладающие преимущественной чувствительностью к какому-либо спектральному цвету и играющие главную роль в создании ахроматических зрительных образов.

Порог

Уровень энергии стимула, при котором мозг начинает реагировать, если энергия сигнала меньше, то это событие остается незамеченным мозгом.

Пиксель

Наименьший адресуемый элемент растрового изображения.

Примитив

Элемент векторного изображения, например, линия, окружность, кривая, куб, сфера. Для изображение П. достаточно выбрать соответствующую пиктограмму и задать параметры (координаты центра, радиус, количество граней на поверхности и т.п.). Параметры обычно определяются протягиванием указателя мыши. Более точный способ - путем ввода чисел с клавиатуры.

Параметры цвета

В качестве параметров используются три субъективных атрибута цвета: цветов ой тон, насыщенность и светлота.

Распознавание образов

Совокупность методов, позволяющих получить описание изображения, поданного на вход системы, либо отнести заданное изображение к некоторому классу.

Разрешающая способность монитора

Определяется количеством отображаемых им пикселей по горизонтали и по вертикали, например, 800x600 .

Растровая графика

Способ построения изображений на экране монитора, при котором графический файл представляет собой отображение текущего состояния экрана в такой форме, что каждый пиксель соответствует некоторому фиксированному объему памяти. Такие файлы получаются при использовании программ Paint, Adobe Photoshop и других.

Рендеринг

То же, что Визуализация (см.)

Светлота

Параметр цвета, который определяет степень ослабленности данного цвета белым цветом

Сжатие информации

Совокупность способов уменьшения объема хранимых данных путем сокращения избыточности. Различают С. без потерь, при котором гарантируется точное восстановление сжатой информации и сжатие с потерями, при котором часть информации может быть утрачена.

Фоторецепторы

Существуют два типа светочувствительных фоторецепторов: Колбочки, сосредоточенные главным образом в центральной ямке и расположенные в основном по периферии сетчатки палочки, не обладающие преимущественной чувствительностью к какому-либо спектральному цвету и играющие главную роль в создании ахроматических зрительных образов. Три типа колбочек называют либо как B, G и R, либо как S, M и L. Пики их чувствительности приходятся примерно на 440 нм, 545 нм и 580 нм (для "усредненного" наблюдателя). В каждом глазу 6 млн. колбочек и 120 млн. палочек (т.е. примерно 250 млн. рецепторов на два глаза).

Цвет

Представление человека о видимой части спектра электромагнитного излучения.

Цветовые модели

Методы математического описания способов получения цветовых оттенков путем смешивания нескольких основных цветов. Основными являются модели RGB, CMYK, HSB, HLS

Цветовая модель RGB

Описывает воспроизведение любого цвета путем сложения трех основных цветов: красного (Red), зеленого (Green) и синего (Blue). Такая модель называется аддитивной (additive).

Цветовая модель CMYK

В качестве основных цветов принимаются три: Cyan (голубой), Magenta (пурпурный), Yellow (желтый), K (черный от слова blacK). Черный цвет используется для повышения контрастности напечатанных изображений, поскольку при смешении трех перечисленных цветов получается коричневатый оттенок. Эта модель используется для описания отраженных от поверхности бумаги цветов, поэтому она называется субтрактивной (subtract - вычитать).

Цветовой тон

Характеристика цвета, которая определяет различие цветов и связан с длиной волны

Цветовой круг

Описание спектральной характеристики видимого света. В основе - красный, желтый и синий цвета. Является традиционным в области искусства.

Цвета основные

Красный, желтый и синий. В традиционной цветовой теории считается, что эти цвета не могут быть получены из остальных, а все остальные получаются как комбинация основных.

Цвета дополнительные

Зеленый, оранжевый и фиолетовый. Получаются путем попарного смешения основных цветов

Цвета производные

Желто-оранжевый, красно-оранжевый, красно-фиолетовый, сине-фиолетовый, сине-зеленый и желто-зеленый. Образуются путем смешения основного и рядом стоящего дополнительного цветов.

Цветовой тон

Параметр, определяющий различие цветов и связан с длиной волны

Список литературы

ОСНОВНАЯ:

1. Сиденко, Л.А. Компьютерная графика, геометрическое моделирование:уч. Пособие/ Л.А. Сиденко – М:Питерпресс, 2009, 224с.

ДОПОЛНИТЕЛЬНАЯ:

1. Артюхин, В.В. Методическое пособие по дисциплине «Компьютерная графика» «Математическое обеспечение и администрирование информационных систем» /В.В.Артюхин; Московский государственный университет экономики, статистики и информатики. – М., 2001. - 56 с.
2. Мухин, О.И. Компьютерная графика [Электронный ресурс] – Режим доступа: http://stratum.ac.ru.
3. Уроки по GIMP [Электронный ресурс]. – Режим доступа: http://www.progimp.ru.
4. Уроки по Inkscape [Электронный ресурс]. – Режим доступа: http://www.openart.ru.
5. Блинова, Т.А. Компьютерная графика / Т.А.Блинова, В.Н.Порев – К.: Издательство Юниор, СПб.: КОРОНА принт. К.:Век+, 2006. – 520с

6. Борковский, А. Б. Англо-русский словарь по программированию и информатике (с толкованиями) / А. Б. Борковский — М.: Рус. яз., 1989. — 335 с.
7. Ватолин Д., Ратушняк А., Смирнов М., Юкин В. Методы сжатия данных. — http://www.comression.ru.
8. Вернер, А. В. Геометрия. Ч. I. Учебное пособие / А. В.Вернер, Б. Е.Кантор, С. А.Франгулов — СПб.: Специальная литература, 1997. — 352 с.
9. Вернер, А. В. Геометрия. Ч. II. Учебное пособие / А. В.Вернер, Б. Е.Кантор, С. А.Франгулов - СПб.: Специальная литература, 1997. — 320 с.
10. Википедия - свободная энциклопедия - http://ru.wikipedia.org/wiki/
11. Глоссарий.ru – Словарь по естественным наукам -www.glossary.ru
12. Залогова, Л.А. Компьютерная графика. Практикум/ Л.А.Залогова. – 2-е изд. – М.:Лаборатория Базовых Знаний, 2005. – 320с.
13. Кишик, А.Н. Adobe Photoshop 7.0. Эффективный самоучитель / А.Н. Кишик - СПб.: ООО «ДиаСофтЮП», 2003. -368с.
14. Клецель, А. Форматы графических файлов - http://www.list.soft.ru.
15. Корн, Г. Справочник по математике (для научных работников и инженеров) / Г. Корн, Т. Корн - М.: Наука, 1984. - 832 с.
16. Миронов, Д.Ф.Компьютерная графика в дизайне: Учебник для вузов/ Д.Ф.Миронов – СПб.:Питер, 2004. – 224с.
17. Майкл Вычислительная геометрия и компьютерная графика на С++: Пер. с англ. / Майкл, Ласло - М.: Бином, 1997. - 304 с.
18. Медицинская энциклопедия [Электронный ресурс] – Малая медицинская энциклопедия, Популярная медицинская энциклопедия «Первая медицинская помощь», Энциклопедический словарь медицинских терминов. (2CD). – Екатеринбург.: Золотой фонд российских энциклопедий, 2004. – 2 электронных оптических диска (CD ROM) – Системные требования: Windows 98/Me/2000/XP; Pentium II 266 МГц; 64 Мб ОЗУ; SVGA 16 бит; 800x600; 4-x CD ROM или DVD дисковод; 250 Мб своб. Места на ЖД; мышь; модем 14,4 Кб/с для доступа в Интернет (желательно).
19. Мусхелишвили, Н. И. Курс аналитической геометрии / Н. И. Мусхелишвили - М.: Высшая школа, 1967. - 656 с.
20. Никулин, Е.А. Компьютерная геометрия и алгоритмы машинной графики / Е.А. Никулин – СПб.: БХВ-Петербург, 2005. – 576 с.
21. Симонович, С.В. Специальная информатика: Учебное пособие / С.В.Симонович, Г.А. Евсеев, А.Г. Алексеев – М.: АСТ-ПРЕСС: Инфорком-Пресс, 2001. – 480с.
22. Страуструп, Б. Язык программирования С++. В 2-х частях/ Б. Страуструп - Киев: ДиаСофт, 1993.
23. Федер, Е. Фракталы / Е.Федер - М.: Мир, 1991. - 254 с.
24. Шикин, А.В. Компьютерная графика. Полигональные модели / А.В.Шикин, А.В.Боресков – М.: ДИАЛОГ-МИФИ, 2005. – 464 с.