ГОСУДАРСТВЕННЫЙ КОМИТЕТ РОССИЙСКОЙ ФЕДЕРАЦИИ ПО ВЫСШЕМУ ОБРАЗОВАНИЮ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

С.М.АВДЕЕВА, А.В.КУРОВ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ КУРСОВОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ "МАШИННАЯ ГРАФИКА"

Москва
1995

ОГЛАВЛЕНИЕ Стр.

1. СОДЕРЖАНИЕ КУРСОВОЙ РАБОТЫ

Курсовая работа по дисциплине "Машинная графика" является первой курсовой работой, выполняемой студентами, обучающимися по специальности "Программное обеспечение вычислительной техники и автоматизированных систем". При ее выполнении студент должен продемонстрировать умение применять теоретические знания и практические навыки при разработке законченного программного продукта.

Курсовая работа представляет собой комплексную работу и ее выполнение требует использования знаний, полученных не только в одной конкретной дисциплине, но и в ходе предшествующего изучения как фундаментальных и общеинженерных дисциплин ("Высшая математика","Физика", "Инженерная графика"), так и дисциплин специальности ("Основы информатики", "Типы и структуры данных", "Программирование на языке Си", "Системное программирование", "Об"ектно-ориентированное программирование").

Курсовая работа должна быть посвящена разработке законченного программного продукта, позволяющего моделировать трехмерные и/или реалистические изображения на экране дисплея. Такая направленность работы связана с тем, что алгоритмы нижнего уровня студенты достаточно глубоко и всесторонне изучают в ходе теоретических и практических занятий в течение предыдущего семестра. Алгоритмы верхнего уровня (предназначенные для изображения трехмерных и реалистических об"ектов) достаточно громоздки, программы, их реализующие, об"емны, что практически делает невозможным их разработку и отладку в ходе лабораторных работ.

В ходе выполнения курсовой работы студенты решают задачи, связанные с обоснованием и разработкой новых или модификацией
и использованием известных методов и алгоритмов представления
об"ектов, выбора и обоснования структуры данных, а также технологические и
эргономические вопросы.

На защиту должны быть представлены: комплекс программ, расчетно-пояснительная записка и графическая часть.

Комплекс программ представляет собой законченный программный продукт, который может настраиваться на конкретную программно-техническкую среду ЭВМ. Для взаимодействия пользователя с программной системой студент разрабатывает инткрфейс пользователя, включающий простое общепринятое меню, необходимые подсказки и помощь как по эксплуатации программы, так и для интерпретации получаемых результатов.

Расчетно-пояснительная записка должна иметь об"ем 30-40 листов рукописного (или машинописного) текста на листах формата А4. Записка должна содержать следующие разделы:

1)Введение
2)Конструкторский
3)Технологический
4)Экспериментально-исследовательский
5)Заключение

Кроме того, записка должна иметь содержание, список использованной литературы, а также в нее могут входить различные приложения.

Во введении дается обзор и анализ существующих программных систем в выбранном направлении, обосновывается необходимость разработки нового комплекса программ. Здесь же проводится анализ и краткое описание с указанием их характеристик известных алгоритмов решения стоящей задачи. Об"ем введения 3-5 листов.

В конструкторской части на основе сделанного во введении обзора проводится выбор и обоснование предлагаемого алгоритма. При использовании известного алгоритма следует более подробно продемонстрировать его преимущества в сравнении с другими известными алгоритмами, указать сложности, особенности его практической реализации, пути решения задач, возникающих в ходе программной реализации.

При разработке нового метода или алгоритма следует подробно изложить полученные самостоятельно (или недостаточно известные) математические соотношения, положенные в основу решения задачи, а также описать предлагаемый алгоритм. При этом следует четко выделить основные этапы работы алгоритма с указанием необходимых исходных данных для его работы и получаемых на каждом этапе результатов.

Как правило, выбор алгоритма тесно связан с используемой структурой данных. Поэтому конструкторская часть должна также содержать обоснование аппроксимации представляемых кривых, поверхностей тел, граней и т.д. Должны быть указаны исходные данные, с помощью которых задаются отображаемые об"екты, проведено сравнение с другими возможными способами их задания. При этом следует проанализировать избыточность

выбранного способа описания об"ектов, показать преимущества или удобства при оперировании этими данными.

На основе выбранных данных для представления об"ектов студент осуществляет последующий выбор и обоснование используемых типов и структур для их машинного представления. При их выборе следует исходить из возможностей языка программирования для их представления, затрат памяти на хранение, времени на обработку,
размерности данных.

Выбор исходных данных и формы их представления должен увязываться с такой характеристикой, как их об"ем и удобства пользователя при вводе. В частности, ввод большого количества данных утомляет пользователя и увеличивает вероятность ввода ошибочных данных. В данной части записки могут выполняться расчеты для определения об"емов памяти, необходимой для хранения исходных данных, промежуточных и окончательных результатов, а также расчеты, позволяющие оценить время решения задачи на ЭВМ. Результаты таких расчетов должны использоваться при сравнении альтернативных вариантов алгоритмов, а также оценки возможности практической реализации стоящей задачи на имеющейся технической базе. Об"ем конструкторской части должен составлять 35-55% всего об"ема записки.

Технологический раздел должен содержать обоснование технологии изготовления комплекса программ: модульная или об"ектно-ориентированная. Необходимо представить модульную структуру комплекса, обоснование выбранного принципа разбиения программ на модули, назначение, взаимосвязь с другими составными частями каждого модуля. В случае использования об"ектно-ориентированного программирования следует обосновать и описать введенные классы об"ектов. В этом же разделе решается вопрос с выбором и обоснованием языка программирования. Большое внимание здесь должно быть также уделено разработке интерфейса пользователя, выбору меню, которое бы в наилучшей степени отвечало характеру работы спроектированного комплекса программ и было удобным и понятным пользователю.

Данный раздел должен заканчиваться изложением руководства

программиста, в котором излагаются требования к аппаратным средствам и программному обеспечению ЭВМ, а также излагается порядок работы с комплексом программ. Эта часть оформляется в соответствии с ГОСТ 19.504-79 и она должна содержать следующие разделы:

-назначение и условия применения программы;
-характеристики программы;
-обращение к программе;
-входные и выходные данные;
-сообщения.

В разделе "Назначение и условия применения программы" должны быть указаны назначение и функции, выполняемые программой, условия, необходимые для выполнения программы (объем оперативной памяти,требования к составу и параметрам периферийных устройств, требования к программному обеспечению).

В разделе "Характеристики программы" приводится описание основных характеристик и особенностей программы (временные, режим работы, средства контроля правильности выполнения и самовосстановления программы).

В разделе "Обращение к программе" должно быть приведено описание процедур вызова программы (способы передачи управления и параметров данных).

В разделе " Выходные данные" должно быть приведено описание используемой входной и выходной информации.

В разделе "Сообщения" должны быть указаны тексты сообщений, выдаваемых программой в ходе ее выполнения, описаны их содержание и действия, которые необходимо предпринять по этим сообщениям.

В приложениях к руководству программиста могут приводиться дополнительные материалы (примеры, иллюстрации, таблицы, графики).

Технологический раздел должен содержать разработанные тесты для проверки правильности работы комплекса программ, результаты тестирования на тестовых примерах. Об"ем этой части работы составляет 35-40%.

Исследовательско-экспериментальный раздел является рекомендуемой частью курсовой работы. Он должен содержать результаты теоретического или экспериментального исследования в ходе выполнения курсовой работы. В первом случае это могут быть результаты, полученные при исследовании математического метода, положенного в основу алгоритма. Во втором случае экспериментально исследуется разработанный комплекс программ с целью получения значений временных, об"емных и иных характеристик комплекса программ (алгоритма) в зависимости от количества изображаемых об"ектов, их сложности, точности представления (вида аппроксимации, количества граней, аппроксимирующих криволинейную поверхность и т.д.).

В этой части работы должны быть представлены примеры использования комплекса программ с изложением постановки конкретной решаемой задачи, описанием конкретных вводимых исходных данных и полученных результатов с указанием значений характеристик требуемых ресурсов ЭВМ (затраты памяти, время счета и т. д.). Об"ем этой части записки составляет 10-15%.

В приложении даются листинги программ пакета или его наиболее интересных составляющих частей. Здесь же могут приводиться твердые копии изображений, получаемых на экране дисплея и
выведенные затем на принтер. При наличии аналитических результатов (в виде числовых величин) даются также и их распечатки, графики, диаграммы. Представленные результаты должны сопровождаться также распечатками исходных данных, для которых они были получены.

Все разделы работы должны быть увязаны тесным образом между собой и представлять собой единое законченное целое.

Материал записки должен излагаться грамотным техническим языком, быть оформлен в соответствии с требованиями ЕСКД, ГОСТ, ЕСПД.

За принятые решения, правильность выполненных расчетов и сделанных выводов ответственность несет автор курсовой работы-студент.

Графическая часть данной курсовой работы носит иллюстративный, вспомогательный характер. Об"ем ее должен составлять 2- 3 листа формата А1.

Основное назначение графической части - помочь студенту наиболее полно в наглядной форме продемонстрировать во время защиты работы существо разработанного программного продукта, изложить основные алгоритмы и математические методы, положенные в основу работы программ.

Графическая часть может выполняться карандашом, тушью или фломастером. При этом все листы должны выполняться однотипно.

На листах графической части должны быть представлены: постановка задачи (в словесной и (или) математической форме), функциональная схема системы, схемы алгоритмов, структура данных, интерфейс пользователя, сравнительные характеристики пакетов (алгоритмов)-аналогов.

В графической части могут быть также представлены иллюстрации (копии экранов) полученных в ходе работы комплекса программ результатов, а также выводы по работе.

Защита курсовой работы

Защита курсовой работы подводит итог всей работы студента в течение семестра. Защита курсовых работ проходит, как правило, в период зачетной сессии. Предварительно составляется график работы комиссий по приему курсовых работ. Обязанность студента записаться на защиту в соответствии с предлагаемым графиком и не допускать переноса срока защиты. Защита осуществляется публично, кроме членов комиссии (2-3 преподавателя) и защищающегося, могут присутствовать другие преподаватели, сотрудники и студенты.

Перед защитой работы студенты должны заблаговременно инсталлировать на ПЭВМ разработанные программные изделия. В начале защиты студент делает доклад с изложением сути проделанной работы, для иллюстрации основных положений он использует графический

материал. После этого, как правило, следуют вопросы со стороны членов комиссии, на которые студент обязан ответить. Вторая часть защиты заключается в демонстрации комплекса программ. При этом необходимо пояснить правила взаимодействия
пользователя с программой, проиллюстрировать на заранее подготовленных примерах характерные особенности реализованного метода (алгоритма). Затем могут быть заданы вопросы по практической части.

Доклад должен быть кратким (5-7 минут), четким и ясным. В докладе должны быть выделены основные задачи, стоявшие при вы-
полнении работы, указаны пути их решения и об"яснены полученные
результаты. Не следует впадать в излишнюю детализацию, останавливаться на второстепенных моментах. Все частности члены комиссии могут выяснить путем постановки соответствующих вопросов. Заканчиваться доклад должен выводами по проделанной работе.

Защита данной курсовой работы является практически первым публичным выступлением студента, поэтому долг руководителя - помочь студенту в составлении доклада. Нельзя строить доклад как некоторое описание или пояснение графической части. Наоборот, графическая часть должна пояснять и помогать в более наглядной форме доносить до слушателей мысли выступающего.

Студент должен перед защитой совместно с преподавателем продумать ответы на возможные вопросы, определить основные достоинства и недостатки курсовой работы, что поможет при ответах на вопросы.

Оценка курсовой работы складывается из ряда показателей, среди которых можно выделить 1)качество, глубину проработки темы, соответствие работы поставленному техническому заданию;
2)качество, об"ем программного продукта, удобство его эксплуатации; 3)качество доклада, правильность ответов на вопросы.

2. ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ КУРСОВОЙ РАБОТЫ

Оформление расчетно-пояснительной записки осуществляется в соответствии с ГОСТ 7.32-81 ЕСКД.

Название разделов и их возможное содержание уже рассмотрены, возможные названия подразделов, их об"ем приведены в разде-

ле 3.

Все листы записки, включая иллюстрации, расположенные на отдельных листах, имеют сквозную нумерацию. Иллюстрации, выполненные на листах, больших чем формат А4, размещаются в конце записки после заключения и учитываются как одна страница.Номер ставится в правом верхнем углу. Первым листом является титульный лист (он не нумеруется). Записка сшивается студентом, причем обложкой является стандартный лист формата А3, выдаваемый руководителем. При его отсутствии студент самостоятельно изготавливает обложку из ватмана или картона.

Вторым листом является задание на выполнение курсовой работы. Задание оформляется на стандартном бланке и подписывается руководителем. При отсутствии стандартного бланка задание оформляется на обычном листе и должно содержать все необходимые разделы, а также подпись руководителя.

После задания должен размещаться реферат, который должен содержать сведения об об"еме курсовой работы, количестве иллюстраций, таблиц, количестве использованных источников. Затем приводится перечень ключевых слов (от 5 до 15) в именительном падеже, перечисляемых в строку через запятую. В тексте реферата указываются основные сведения о разработанном комплексе программ (цель разработки, метод исследования, полученные результаты, их новизна, область применения, основные характеристики).

После реферата размещается содержание расчетно-пояснительной записки, которое включает наименования всех разделов, подразделов и пунктов (если они имеют названия) с указанием номеров страниц, где они начинаются.

Далее следует перечень условных обозначений, символов,

единиц и терминов. Они размещаются в алфавитном порядке, причем слева приводится сокращение, а справа его расшифровка. Если сокращение используется менее трех раз, то расшифровка может быть дана в тексте.

Вслед за перечнем располагаются введение, основная часть и заключение. В заключении должны содержаться краткие выводы по результатам выполненной работы и предложения по их использованию, дальнейшему развитию или модификации разработанного комплекса программ. Должно быть указано соответствие технических характеристик разработанного комплекса программ характеристикам, указанным в техническом задании. Об"ем заключения 1-2 листа.

После заключения должны располагаться список использованных литературных источников и приложения.

Сведения об использованных источниках располагаются в том же порядке, что и ссылки на них в тексте записки, список оформляется в соответствии с ГОСТ 7.1-76. В расчетно-пояснительной записке ссылки на использованные литературные источники (книги, статьи, стандарты, справочники) даются в местах, где были использованы сведения из этой литературы. Ссылки представляют собой порядковый номер по списку источника, заключенный в косые черточки, например, /3,5/.

Библиографическое описание источника составляется на языке текста этого источника . Бибилиографическое описание представляет собой совокупность сведений о документе, необходимых и достаточных для общей характеристики и идентификации документа.

Библиографическое описание состоит из элементов, об"единенных в области, и заголовка. Элементы и области приводятся в последовательности, установленной стандартом. В описании можно выделить следующие составные части: основное заглавие, область издания, область выходных данных, область количественной характеристики, т.е. должны указываться сведения об авторе, названии источника, издательстве, месте и дате издания, об"еме.

Каждой области описания (кроме первой) предшествует знак тире и точка. Внутри элемента пунктуация должна соответствовать нормам языка, на котором составлено описание.

В заголовке описания фамилии авторов приводятся в именительном падеже с указанием инициалов после фамилии, фамилии нескольких авторов разделяются запятыми. Если книга имеет более трех авторов, то сначала располагается название книги, а затем перечисляются авторы, причем в этом случае инициалы предшествуют фамилии. При наличии многих авторов перечисляют первых трех, а затем добавляют слово "и др.".

После заглавия через двоеточие даются сведения, поясняющие заглавие, уточняющие назначение, а также могут указываться сведения о дате и месте проведения мероприятия (например,для сборников материалов конференций). Помещаемые после заглавия сведения об ответственности отделяют от заглавия косой чертой.

Область выходных данных содержит сведения о том, где, кем и когда была опубликована книга. Название места издания приводят в именительном падеже. В качестве даты приводится год издания. Область количественной характеристики содержит об"ем книги в страницах или начальную и конечную страницы расположения для статей, тезисов докладов и т.д.

Сведения о документе, в котором помещена составная часть (например, статья из сборника), располагаются после сведений о составной части через знак две косые черты, причем до и после него делается по одному пробелу.

Примеры оформления пристатейного библиографического списка приведены в списке рекомендуемой литературы.

Каждое приложение следует начинать с нового листа с указанием в правом верхнем углу слова "ПРИЛОЖЕНИЕ". Приложение должно иметь содержательный заголовок. Каждое приложение имеет свой порядковый номер, для нумерации используются арабские цифры. Нумерация разделов, таблиц, рисунков, формул ведется в пределах каждого приложения. Располагаемые в приложениях распечатки программ должны быть сложены по формату А4.

Текст расчетно-пояснительной записки располагается на стандартных листах бумаги формата А4 с одной стороны, должны выдерживаться следующие размеры полей: левое - 30 мм, правое - 10 мм, верхнее - 15 мм, нижнее - 20 мм. Заголовки разделов располагаются симметрично тексту прописными буквами. Заголовки подразделов располагают с абзацным отступом строчными буквами (кроме первой прописной). Перенос слов в заголовках не допускается, точка в конце не ставится.

Каждый раздел должен начинаться с нового листа. Номера разделов обозначаются арабскими цифрами с точкой в конце, подразделы нумеруют арабскими цифрами в пределах каждого раздела (состоит из номера раздела и подраздела, разделенных точкой, в конце ставится точка), например, 2.3. Пункты нумеруют арабскими цифрами в пределах каждого подраздела, например, 2.3.1.

Иллюстрации обозначают словом "Рис." и нумеруют последовательно арабскими цифрами в пределах раздела, при этом номер рисунка состоит из номера раздела и номера рисунка, например,
рис. 2.3. Иллюстрации должны иметь наименование. При необходимости их снабжают поясняющими данными. Наименование иллюстрации помещают над ней, поясняющую надпись - под ней.Номер рисунка помещается ниже поясняющей надписи.

Таблицы нумеруют аналогично, при этом вверху таблицы справа пишут слово "Таблица" и указывают номер. Каждая таблица должна иметь заголовок. Заголовок и слово Таблица пачинают с прописной буквы. Заголовки граф пишут с прописных букв, подзаголовки - со строчных, если они составляют одно предложение с заголовком. Если подзаголовки имеют самостоятельное значение, то они пишутся с прописных букв. Графы таблиц делить по диагонали не допускается.

Иллюстрации и таблицы располагают в тексте после первой ссылки на них так, чтобы их можно было читать без поворота записки или с поворотом по часовой стрелке на 90 градусов.

Формулы нумеруют арабскими цифрами в пределах раздела, при этом номер состоит из номера раздела и порядкового номера формулы и помещается в круглых скобках у правого поля листа на строке самой формулы. Под формулой располагают пояснение значений символов в той же последовательности, что и в формуле. Значение каждого символа пишется с новой строки, первому символу предшествует слово "где" без двоеточия. Ссылка на формулу производится путем указания ее номера в круглых скобках.

При изображении схем следует руководствоваться правилами оформления, изложенными в действующих ГОСТ, ЕСКД, ЕСПД. Часть графического материала должна дублироваться в записке. Это требование является обязательным, так как расчетно-пояснительная записка является самостоятельным документом и ее содержание
должно быть понятно и без графической части.

Расчетно-пояснительная записка подписывается студентом, а затем преподавателем - руководителем курсовой работы. Подпись руководителя означает допуск студента к защите курсовой работы.

3. ПРИМЕР ЗАДАНИЯ НА ВЫПОЛНЕНИЕ КУРСОВОЙ РАБОТЫ

ЗАДАНИЕ НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ ПО КУРСУ "МАШИННАЯ ГРАФИКА"

СТУДЕНТА ГРУППЫ ИУ7 - 51 СИДОРОВА С.Н.

ТЕМА КУРСОВОЙ РАБОТЫ

"Разработка ППП, моделирующего движение группы динамических объектов в пространстве и синтезирующего их изображение на экране дисплея."

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Промоделировать движение и получить изображение на экране графического дисплея группы объектов (от 1 до 10), совершающих управляемые маневры в пространстве. Объекты описываются координатами вершин (x,y,z), ребрами и гранями. В качестве управляющих сигналов задаются значения векторов угловой и линейной скоростей:

$$W = F(t),\ t \in [t0,tk];$$

$$V = F(t),\ t \in [t0,tk],$$

где [t0,tk] -интервал времени моделирования.

Предполагается, что картинная плоскость изображения совпадает с экраном графического дисплея. Частота смены изображения не менее 25 Гц.

При работе с изображением реализовать процедуру " Быстрого перемещения изображения объекта".

Требования к процедуре "Быстрого перемещения изображения об"екта":

1. Изображение объекта задается битовой картой.
2. Смена номера изображения производится под управлением вызывающей программы в процессе настройки.
3. После переноса изображения управление передается вызывающей программе для расчета нового положения объекта.
4. В процедуру передаются следующие параметры:
   - координаты центра изображения (xc,yc);
   - номер объекта (номер группы битовой карты);
   - номер объекта в группе;
   - адреса всех битовых карт; при необходимости:
   - текущие координаты изображения (проекции (xvi, yvi) объектов на картинную плоскость);
5. Размер изображения:
   - max: 32 * 20 пикселов;
   - min: 8 * 5 пикселов.
6. Интерфейс процедуры должен соответствовать стандарту языка Паскаль.

СОСТАВ КУРСОВОЙ РАБОТЫ Расчетно-пояснительная записка. Графическая часть.

Пакет программ.

ПРИМЕРНОЕ СОДЕРЖАНИЕ СОСТАВНЫХ ЧАСТЕЙ РАБОТЫ:

5. СПИСОК ЛИТЕРАТУРЫ
6. ПРИЛОЖЕНИЯ

П.1. Листинг программы .....
П.2. Копии экрана .....
П.3. Распечатки результатов .....

ГРАФИЧЕСКАЯ ЧАСТЬ

1. Постановка задачи
2. Математические методы решения задачи.....
3. Функциональная схема системы .....
4. Схема алгоритма .....
5. Сравнительные характеристики аналогов .....
6. Листинг программы (фрагмент) .....
7. Интерфейс пользователя .....
8. Иллюстрация работы с примером задания исходных данных.....

На защиту должны быть представлены:
1. Пояснительная записка объемом 25 - 30 страниц.
2. Графическая часть - 3 листа формата А1.

4. СПИСОК РЕКОМЕНДУЕМЫХ ТЕМ КУРСОВЫХ РАБОТ

1. Реализация алгоритма Робертса для об"ектов, описываемых полигональными моделями.
2. Реализация алгоритма Варнока для об"ектов, описываемых полигональными моделями.
3. Реализация алгоритма с приоритетами для об"ектов, описываемых полигональными моделями.
4. Реализация алгоритма Z-буфера для об"ектов, описываемых полигональными моделями.
5. Реализация алгоритма построчного сканирования для об"ектов, описываемых полигональными моделями.
6. Реализация алгоритма трассировки лучей для об"ектов, описываемых полигональными моделями.
7. Реализация алгоритма трассировки лучей с учетом источников освещения и специальными эффектами (учет прозрачности, отражения, преломления).
8. Реализация простого алгоритма закраски.
9. Реализация алгоритма закраски по методу Гуро.
10. Реализация алгоритма закраски по методу Фонга.
11. Реализация и сравнительное исследование алгоритмов закраски - простой, по методу Гуро и по методу Фонга.
12. Построение реалистических изображений с учетом теней.
13. Реализация алгоритмов для построения изображений с учетом перспективы.
14. Пакет деловой графики (двух- и трехмерный варианты).
15. Пакет для изображения и манипуляции с трехмерным (об"емным) шрифтом.
16. Пакет для изображения рельефа местности на основе линий уровня.
17. Обучающий пакет для об"яснения происхождения конических и цилиндрических сечений.
18. Пакет для изображения поверхностей вращения по заданной образующей.
19. Графическая библиотека примитивов для построения трехмерных об"ектов.

5. СПИСОК ЛИТЕРАТУРЫ, ИСПОЛЬЗУЕМЫЙ ПРИ ВЫПОЛНЕНИИ КУРСОВОЙ РАБОТЫ

1. Аммерал Л. Машинная графика на языке Си.-М.:СолСистем,
1992.
Т.1:Принципы программирования в машинной графике.-224 с.
Т.2:Машинная графика на персональных компьютерах.-232 с.
Т.3:Интерактивная трехмерная машинная графика.-317 с.
Т.4:Программирование графики на Турбо Си.-221 с.
2. Булатов В., Дмитриев В. Увидеть невидимое // Компьютер- Пресс.-1993.-N4.-С.3-10.
3. Булатов В., Дмитриев В. Искусство преображения информации.
Ч.1 // КомпьютерПресс.-1993.-N4.-С.11-16.
4. Булатов В., Дмитриев В. Искусство преображения информации.
Ч.2 // КомпьютерПресс.-1993.-N5.-С.20-26.
5. Гардан И., Люка М. Машинная графика и автоматизация конструирования.-М.:Мир.-1987.-
272 с.
6. Геометрический процессор синтезирующей системы визуализации / В.А.Бурцев, С.В.Власов, С.И.Вяткин и др. // Автометрия.-1986.-N4.-С.3-8.
7. Гилой В. Интерактивная машинная графика.-М.:Мир,1981.-380 с.
8. Ковалев А.М., Талныкин Э.А. Машинный синтез визуальной обстановки // Автометрия.-
1984.-N4.-С.67-76.
9. Курковский С. Интервальные методы в компьютерной графике //
Монитор.-1993.-N7-8.-С.76-85.
10.Ньюмен У., Спрулл Р. Основы интерактивной машинной графики.-
М.: Мир,1976.-573 с.
11.Павлидиус Т. Алгоритмы машинной графики и обработки изображений.- М.:Радио и связь,1986.-400 с.

12.Роджерс Д., Адамс Дж. Математические основы машинной графики.
-М.:Мир, 1980.-240 с.
13.Роджерс Д. Алгоритмические основы машинной графики.-М.:Мир,
1989.-512 с.
14.Уокер Б.С., Гурд Дж., Дроник Е.А. Интерактивная машинная графика.-
М.:Машиностроение, 1980.-168с.
15.Фоли Дж., вэн Дэм А. Основы интерактивной машинной графики.
-М.:Мир,1985.-Т.1:375 с.,Т.2:368 с.
16.Фролов А.Г., Фролов Г.В. Программирование видеоадаптеров
CGA, EGA и VGA. - М.: Диалог - МИФИ, 1992.-288 с.
17. Хирн Д., Бейкер М. Микрокомпьютерная графика.-М.:Мир, 1987.
-352 с.
18. Шикин Е.В., Боресков А.В., Зайцев А.А. Начала компьютерной графики.-М.: Диалог-МИФИ,
1993.-138 с.
19. Эгрон Ж. Синтез изображений. Базовые алгоритмы.-М.:Радио и связь, 1993.-216 с.
20. Эйнджел Й. Практическое введение в машинную графику.-М.:Радио и связь, 1984.-135 с.
21. Эндерле Г., Кэнси К., Пфафф Г. Программные средства машинной графики. Международный стандарт GKS. - М.:Радио и связь,
1988.-479 с.

6. АЛГОРИТМЫ МАШИННОЙ ГРАФИКИ, ИСПОЛЬЗУЕМЫЕ ПРИ ВЫПОЛНЕНИИ КУРСОВОЙ РАБОТЫ

Алгоритмы машинной графики можно разделить на два уровня: нижний и верхний. Группа алгоритмов нижнего уровня предназначена для реализации графических примитивов. Они достаточно подробно изучаются во время лекционных занятий и реализуются студентами на практических занятиях. Эти алгоритмы или подобные им воспроизведены в графических библиотеках языков высокого уровня, реализованы аппаратно в графических процессорах и графических рабочих станциях.

Однако для конкретных случаев можно написать программу, существенно более эффективную по времени. Среди алгоритмов нижнего уровня можно выделить группу простейших в смысле используемых математических методов и отличающихся простотой реализации.

Как правило, такие алгоритмы не являются наилучшими по объему выполняемых вычислений или требуемым ресурсам памяти. Поэтому можно выделить вторую группу алгоритмов, использующих более сложные математические предпосылки и отличающихся большей эффективностью.

К третьей группе следует отнести алгоритмы, которые могут быть без больших затруднений реализованы аппаратно (допускающие распараллеливание, рекурсивные, реализуемые в простейших командах). В эту группу могут попасть и алгоритмы, представленные в первых двух группах.

Наконец, к четвертой группе можно отнести алгоритмы со специальными эффектами, например, с устранением лестничного эффекта.

К алгоритмам верхнего уровня относятся в первую очередь алгоритмы удаления невидимых линий и поверхностей. Задача удаления невидимых линий и поверхностей продолжает оставаться центральной в машинной графике. От эффективности алгоритмов, позволяющих решить эту задачу, зависят качество и скорость построения трехмерного изображения.

Однако при этом не следует забывать, что вывод объектов обеспечивается примитивами, реализующими алгоритмы нижнего уровня, поэтому нельзя игнорировать проблему выбора и разработки эффективных алгоритмов нижнего уровня.

Для разных областей применения машинной графики на первый план могут выдвигаться разные свойства алгоритмов. Для научной графики большое значение имеет универсальность алгоритма, быстродействие может отходить на второй план. Для систем моделирования, воспроизводящих движущиеся объекты, быстродействие становится главным критерием, поскольку требуется генерировать изображение практически в реальном масштабе времени.

В данном пособии рассматривается ряд алгоритмов верхнего уровня, рассчитанных на работу с примитивами, позволяющих получить хорошие результаты при небольших затратах памяти и процессорного времени.

К задаче удаления невидимых линий и поверхностей примыкает задача построения реалистических изображений, т.е. учета явлений, связанных с количеством и характером источников света, учета свойств поверхности тела (прозрачность, преломление, отражение света).

В пособии рассматриваются алгоритмы двух последних групп, поскольку усилия студентов во время выполнения курсовой работы сосредоточиваются именно на реализации и использовании данных алгоритмов.

Сложность задачи удаления невидимых линий и поверхностей привела к появлению большого числа различных способов ее решения, различных алгоритмов, но наилучшего решения поставленной задачи не существует. Главным недостатком всех алгоритмов является значительный объем вычислений, необходимых для определения удаляемых линий и поверхностей.

Вначале реализации любого алгоритма удаления невидимых линий и поверхностей для повышения эффективности его работы обычно проводится сортировка координат объектов синтезируемой сцены. Основная идея сортировки заключается в том, что, чем дальше расположен объект от точки визирования, тем больше вероятность того, что он будет полностью или частично экранироваться одним из объектов, более близких к точке наблюдения.

Алгоритмы удаления невидимых линий и поверхностей классифицируются по способу выбора систем координат или пространства, в котором они работают. Первый класс - это алгоритмы, работающие в объектном пространстве, имеющие дело с физической системой координат (мировые координаты), в которой они описаны. Второй класс алгоритмов работает в пространстве изображения и имеет дело с системой координат того устройства, на котором эти объекты синтезируются. Алгоритмы первого класса используются в тех случаях, когда требуется высокая точность изображения объектов. Синтезируемые в этом случае изображения можно свободно увеличивать (уменьшать) во много раз, сдвигать или поворачивать. Точность вычисле-

ний алгоритмов второго класса ограничивается разрешающей способностью экрана. Результаты, полученные в пространстве изображения, а затем увеличенные (уменьшенные) во много раз, не будут соответствовать исходной сцене.

Различны и объемы вычислений для различного рода алгоритмов. Так объем вычислений для алгоритмов, работающих в объектом
пространстве, и сравнивающих каждый объект сцены со всеми остальными объектами этой сцены, растет теоретически как квадрат числа объектов. Аналогично, объем вычислений алгоритмов, работающих в пространстве изображений и сравнивающих каждый объект с позициями пикселов в экранной системе координат, растет теоретически, как произведение числа объектов на число пикселов экрана.

На практике, сравнительный анализ существующих алгоритмов удаления невидимых линий крайне затруднителен. В различных случаях при работе с различными моделями синтезируемого пространства эффективны различные алгоритмы. Даже при работе с одной и той же моделью оказывается, что в зависимости от точки наблюдения следует использовать различные алгоритмы.

6.1. АЛГОРИТМ РОБЕРТСА

Алгоритм Робертса представляет собой первый из известных алгоритмов удаления невидимых линий. Этот математически элегантный метод работает в объектном пространстве. Модификации алгоритма, использующие предварительную пространственную сортировку вдоль оси z и простые габаритные или минимаксные тесты, позволяют добиться почти линейной зависимости роста объема вычислений от роста числа объектов.

Робертс решил проблему удаления невидимых линий для объектов, составленных из выпуклых многогранников. Любой замкнутый объект с плоскими гранями можно представить в виде набора выпуклых многогранников (рис.), то есть в виде наборов плоскостей, образующих грани данных многогранников. Поскольку объем вычислений в алгоритмах удаления невидимых линий и поверхностей
растет с увеличением числа многоугольников, для описания поверхностей объектов желательно использовать многоугольники с более чем тремя сторонами.

Таким образом, для реализации алгоритма необходимо вначале представить объекты визуализации в виде наборов многогранников, каждый из которых задан уравнениями плоскостей своих граней. Каждая плоскость многогранника описывается уравнением:

$$aX + bY + cZ + d = 0, \qquad (6.1)$$

где X,Y и Z - мировые координаты.

В матричной форме это уравнение запишется в виде:

$$[X\ Y\ Z\ 1]\,[P] = 0, \text{ где}$$

$$P = \begin{vmatrix} a \\ b \\ c \\ d \end{vmatrix}.$$

Любой выпуклый объект можно описать матрицей объекта, состоящей из коэффицентов уравнений плоскостей:

$$V = \begin{vmatrix} a1 & a2 & \dots & an \\ b1 & b2 & \dots & bn \\ c1 & c2 & \dots & cn \\ d1 & d2 & \dots & dn \end{vmatrix},$$

где каждый столбец содержит коэффициенты одной плоскости.

Известно, что любая точка на плоскости однозначно определяется двумя координатами; точка в пространстве в однородных коор-
динатах описывается вектором [S] = [X Y Z 1]. Известно также, если точка лежит на плоскости, то скалярное произведение [S][P] равно 0; если же точка не лежит на плоскости, то знак этого скалярного произведения показывает, по какую сторону от плоскости расположена точка. В

алгоритме Робертса предполагается, что точки, лежащие внутри тела дают положительное скалярное произведение.

Сформировать матрицу объекта, состоящую из коэффициентов уравнений плоскостей можно несколькими методами. Один из них использует общеизвестный из аналитической геометрии принцип, что плоскость можно определить по трем неколлинеарным точкам (хотя уравнение плоскости содержит четыре неизвестных коэффициента, его можно нормировать так, чтобы d = 1). Другой метод используется, если известен вектор нормали к плоскости , то есть :

$$\mathbf{n} = a\mathbf{i} + b\mathbf{j} + c\mathbf{k},$$

где **i,j,k** - единичные векторы осей X, Y, Z соответственно. Уравнение плоскости описывается формулой (6.1). Коэффицент d вычисляется с помощью произвольной точки на плоскости. Например, если эта точка с координатами (x1, y1, z1), то d = - (ax1 + by1 + cz1).

Метод, предложенный Мартином Ньюэлом, позволяет найти как точное решение для уравнений плоскостей многоугольников, так и "наилучшее" приближение для неплоских многоугольников. Этот метод эквивалентен определению нормали в каждой вершине многоугольника с помощью векторного произведения прилежащих ребер и усреднения результатов. Коэффициенты плоскости вычисляются из следующей стстемы уравнений:

$$a = (yi - yj)(zi + zj)$$

$$b = (zi - zj)(xi + xj)$$

$$c = (xi - xj)(yi + yj),$$

где j=i+1,если i=3,то j=1,

а коэффициент d вычисляется с помощью любой точки на плоскости.

Перед реализацией алгоритма удаления невидимых линий и поверхностей для получения нужного положения синтезируемой сцены

обычно проводится трехмерное видовое преобразование. Матрицы объектов преобразованной сцены можно получить или преобразованием исходных матриц объектов визуализации или вычислением новых матриц объектов, используя преобразованные вершины или другие точки объектов.

Предположим, что [B] - матрица однородных координат, задающая исходные вершины объекта, [T] - матрица видового преобразования размером 4*4. Тогда координаты преобразованных вершин опишуться с помощью соотношения:

$$[BT] = [B][T];$$

а уравнения исходных плоскостей, ограничивающих объект:

$$[B][V] = [D],$$

где [D] - ненулевая матрица. Аналогично, уравнения преобразованных плоскосстей задаются следующим образом:

$$[BT][VT] = [D],$$

где [VT] - преобразованная матрица объекта. Приравнивая левые части двух последних соотношений, получаем:

$$[BT][VT] = [B][V].$$

Затем преобразовываем к виду:

$$[VT] = [T]\,[V].$$

Таким образом, преобразованная матрица объекта получается умножением исходной матрицы объекта слева на обратную матрицу видового преобразования.

После формирования матрицы объекта и видового преобразования определяются невидимые грани и ребра объекта. Предполагается, что наблюдатель находится в бесконечности на положительной полуоси Z и смотрит в начало координат. В однородных координатах вектор такого направления равен :

$$[T] = [0\ 0\ -1\ 0].$$

Этот вектор также является образом точки, лежащей в бесконечности на отрицательной полуоси Z. Фактически [E] представляет собой любую точку, лежащую на плоскости Z= -. . , то есть любую точку типа (X,Y,- . .). Поэтому, если скалярное произведение [E] на столбец матрицы объекта, соответствующий какой-либо плоскости, отрицательно, то [E] лежит по отрицательную сторону от этой плоскости. И, следовательно, эта плоскость невидима из любой точки наблюдения, лежащей в плоскости Z =. . , а исходная точка на Z = . .- экранируется самим объектом (рис.). Такие плоскости называются нелицевыми.

Таким образом, если скалярное произведение [E][V] < 0, то плоскости являются нелицевыми (для аксонометрических проекций это эквивалентно поиску положительных значений в третьей строке матрицы объекта). После отбрасывания нелицевых плоскостей число граней для выпуклых многогранников сокращается примерно наполовину.

Вышеописанный метод удаления нелицевых плоскостей эквивалентен вычислению нормали к поверхности для каждого отдельного многоугольника. Если нормаль отрицательна, то она направлена в сторону от наблюдателя, и , следовательно, данный многоугольник не виден.

Затем определяются и удаляются из списка ребер нелицевые ребра, которые образуются в результате пересечения пары нелицевых
плоскостей (рис.).

После удаления нелицевых граней и ребер для каждого объекта выполняется самая трудоемкая часть алгоритма : проверка каждого оставщегося ребра на закрытие другими объектами. При этом использование Z - сортировки и простого минимаксного или габаритного с прямоугольной объемлющей оболочкой тестов позволяет удалить целые группы ребер и объектов. Например, все тела в сцене упорядочиваются в некоторый преоритетный список по значениям Z ближайших вершин и расстояния до наблюдателя. Тогда никакой объект из этого списка, у которого ближайшая точка находится дальше от наблюдателя, чем самая удаленная из концевых точек ребра, не может закрывать это ребро. Более того, ни одно из оставшихся тел, прямоугольная оболочка которого расположена полностью справа, слева, над или под ребром, не может экранировать это ребро. Использование этих приемов значительно сокращает число тел, с которыми нужно сравнивать каждый отрезок или ребро.

Для сравнения отрезка (R1,R2) с объектом используется параметрическое представление этого отрезка:

$$R(t) = R1 + (R2 - R1)t \ , \quad 0<=t<=1 \qquad (6.2)$$

или

$$V = S + dt,$$

где V - вектор точки на отрезке, S - начальная точка, a d
- направление отрезка.

Необходимо определить: существуют ли значения t , при которых данный отрезок или ребро невидимы. Для этого формируется другой отрезок от точки R(t) до точки наблюдения g :

$Q(f,t) = U = V + gf = S + dt + gf$, $0<=t<=1$, $f>=0$. Значение t указывает точку на отрезке R(t) , a f указывает точ-

ку на отрезке, проведенном от точки R(t) до точки наблюдения.
Фактически Q(f,t) представляет собой плоскость в трехмерном пространстве, а (f,t) определяет точку на этой плоскости. Значение f всегда положительно, ибо объекты, экранирующие R(t), могут находиться только в той части данной плоскости, которая заключена между исследуемым отрезком R(t) и точкой наблюдения.

Скалярное произведение любой точки, расположенной внутри объекта, на матрицу тела положительно. Если точка лежит внутри тела, то она невидима. Следовательно, для определения невидимой части ребра, которая экранируется объектом, достаточно определить те значения f и t , для которых скалярное произведение Q(f,t) = U
на матрицу объекта положительно:

$$H = U\,[VT] = S[VT] + dt[VT] + gf[VT] > 0. \qquad (6.3)$$

Те значения t и f , для которых все значения компоненты H положительны соответствуют невидимой части ребра.

Обозначим:

$$p = S[VT] \ , \quad q = d[VT] \ , \quad w = g[VT].$$

Тогда условие (6.3) запишется в виде:

$$Hj = pj + tqj + fwj > 0 \ , \qquad (6.4)$$

где j - номер столбца в матрице объекта.

Условия (6.4) должны выполняться для всех плоскостей, ограничивающих объект, то есть для всех j.

Случай, когда Hj = 0, является граничным между видимой и невидомой частями ребра. Полагая Hj = 0 для всех плоскостей, получают систему уравнений относительно t и f, которые должны удовлетворяться одновременно. Решая эту систему уравнений, находят все значения t и

f, при которых изменяется значение видимости ребра или части ребра (число возможных возможных решений при j
плоскостях равно j(j - 1)/2). Затем каждое решение в интервалах
0<= t <=1 , f >= 0, подставляется во все остальные неравенства системы (6.3) для проверки того, что условие Hj >= 0 выполнено. Поиск корректных решений производится для того, чтобы найти минимальное среди максимальных значений параметра t (t minmax) и максимальное среди минимальных значений t (t maxmin). Подставляя эти значения в уравнение (6.2) определяют видимые участки ребра на интервалах [0,t maxmin] и [t mimmax,1]. Условие экранирования ребер или отрезков ребер является простым следствием из классической задачи линейного программирования:

t maxmin < t , t minmax.

Решения, удовлетворяющие неравенствам Hj > 0, могут существовать и за пределами области, ограниченной условиями:

0 <= t <= 1 и f >= 0.

Поэтому к системе (6.4) необходимо добавить три уравнения, описывающие эти границы:

t = 0 , t-1 = 0 и f = 0.

Тогда число решений будет равно (j + 2)(j + 3)/2.

Для ускорения работы алгоритма перед определением t maxmin и t minmax удаляются не только нелицевые ребра, но и полностью видимые ребра. Видимые ребра определяются на основании того, что оба конца такого ребра должны лежать между точкой наблюдения и какой-либо видимой плоскостью. При f = 0 значение U задает сам отрезок. Далее, если f = 0, то при t = 0 и t = 1 получаются концевые точки отрезка. Из соотношений (6.4) видно, что при t = 0 pj является скалярным произведением концевой точки отрезка и j - ой плоскости. Аналогично, pj + qj является скалярным произведением другой концевой точки отрезка и j-ой плоскости. В свою очередь,

j-я плоскость, ограничивающая объект видима, если wj <= 0. Следовательно, если wj <=0 , pj <= 0 и pj + qj <=0, то отрезок полностью видим , а оба его конца лежат либо на видимой плоскости, либо между этой плоскостью и точкой наблюдения.

Для полностью невидимых ребер объекта отсутствуют простые тесты из-за бесконечности плоскостей. Поэтому полностью невидимые ребра определяют также как и частично невидимые (в этом случае невидимый участок будет простираться от t = 0 до t = 1).

После определения частично видимых или полностью невидимых ребер определяются пары объектов, связанных отношениями протыкания, (в случае протыкания объектов сцены возникают решения на границе f = 0) и вычисляются отрезки, которые образуются при протыкании объектами друг друга. Эти отрезки проверяются на экранирование всеми прочими объектами сцены. Видимые отрезки образуют структуру протыкания.

Таким образом, реализация алгоритма Робертса подразделяется на следующие этапы
(рис.):

1. Определение коэффициентов уравнения плоскости каждой грани, проверка правильности знака уравнения и формирование матрицы объекта визуализации.

2. Проведение видового преобразования матрицы объекта, вычисление прямоугольной охватывающей оболочки объекта.

3. Определение нелицевых граней, удаление их из списка граней и соответствующих ребер - из списка ребер.

4. Определение списка других объектов синтезируемой сцены, которые могут быть экранированы данным объектом визуализации на
основании сравнений охватывающих оболочек объектов.

5. Формирование списка протыканий также на основании сравне-

ний охватывающих оболочек объектов.

6. Определение невидимых ребер или участков ребер. Практическая реализация данного этапа основана на том, что скалярное произведение любой точки, лежащей внутри объекта на матрицу объекта положительно.

7.Формирования списка возможных ребер, соединяющих точки протыкания, для пар объектов, связанных отношением протыкания.

8.Проверка видимости полученных ребер по отношению ко всем объектам сцены в соответствии с действиями этапов 3 и 6.

9.Визуализация изображения.

6.2. АЛГОРИТМ ВАРНОКА

Алгоритм Варнока работает в пространстве изображений. В основу алгоритма положен принцип "разделяй и властвуй", состоящий в разбиении области рисунка на более мелкие подобласти (окна). Для каждой подобласти (окна) определяются связанные с ней многоугольники и те из них, видимость которых определить "легко", изображаются на экране. В противном же случае разбиение повторяется, и для каждой из вновь полученных подобластей рекурсивно применяется процедура принятия решения.

Предполагается, что с уменьшением размеров области ее перекрывает все меньшее и меньшее количество многоугольников. Считается, что в пределе будут получены области, содержащие не более одного многоугольника, и решение будет принято достаточно просто. Если же в процессе разбиения будут оставаться области, содержащие не один многоугольник, то следует продолжать процесс разбиения до тех пор, пока размер области не станет совпадать с одним пикселом. В этом случае для полученного пиксела необходимо вычислить глубину (значение координаты Z) каждого многоугольника и визуализировать тот из них, у которого максимальное значение этой координаты (считаем, что наблюдатель находится на оси Z в плюс бесконечности).

В процессе анализа взаимного расположения окна и многоугольника могут быть получены следующие варианты:

1) многоугольник внешний (целиком находится за пределами области);
2) многоугольник внутренний (целиком лежит внутри области);
3) пересекающий многоугольник (пересекает область, т. е. одна часть его лежит внутри области, другая - за пределами области);
4) охватывающий многоугольник (область целиком лежит внутри многоугольника).

Внешние многоугольники не оказывают влияния на принятие решения о визуализации окна. Внешняя часть пересекающего многоугольника может рассматриваться как внешний многоугольник. Внутренняя часть пересекающего многоугольника обрабатывается так же, как и внутренний многоугольник.

В следующих четырех случаях можно непосредственно принять решение относительно окна и не выполнять дальнейшего его разбиения:

1. Все многоугольники являются внешними по отношению к окну; область можно закрасить цветом фона.
2. Имеется только один внутренний или только один пересекающий многоугольник. В этом случае сначала окно закрашивается цветом фона, а затем многоугольник преобразуется в растровую форму. Для пересекающего многоугольника сначала

   следует выполнить отсечение и результат (внутренний многоугольник) преобразовать в растровую форму. Для преобразования в растровую форму может применяться один из алгоритмов нижнего уровня или примитив, имеющийся в составе графической библиотеки языка программирования.
3. Имеется единственный охватывающий многоугольник и нет никаких других многоугольников. Область закрашивается цветом этого охватывающего многоугольника.
4. Имеется несколько пересекающих, внутренних и охватывающих многоугольников. В этом случае делается попытка найти охватывающий многоугольник, расположенный впереди всех других многоугольников. При нахождении такого многоугольника область закрашивается цветом охватывающего многоугольника.

Для выявления такого охватывающего многоугольника применяется следующий тест: вычисляются Z-координаты плоскостей всех охватывающих, пересекающих и внутренних многоугольников в четырех угловых точках рассматриваемой области. Если существует охватывающий многоугольник, для которого все четыре вычисленные Z -координаты больше (ближе к наблюдателю), чем любые из других вычисленных Z-координат, то этот охватывающий многоугольник лежит перед всеми другими многоугольниками в

рассматриваемой области, следовательно, область закрашивается цветом охватывающего многоугольника. Проверяемое в этом случае условие является достаточным, но не необходимым, чтобы охватывающий многоугольник был расположен ближе других к наблюдателю.

Если исследуемая ситуация не приводится ни к одному из этих четырех случаев, то проводится разбиение области. При этом
следует проверять лишь внутренние и пересекающие многоугольники. Охватывающие и внешние многоугольники для исходной области останутся таковыми по отношению и к любому из получаемых подокон. Процесс разбиения завершается, когда размер области совпадает с одним пикселом (достигается разрешение экрана).

Если ни один из четырех случаев не обнаруживается и для области размером с пиксел, то вычисляется глубина в этой точке для всех многоугольников, связанных с областью (внутренних, пересекающих, охватывающих), и пиксел закрашивается цветом многоугольника с максимальной Z-координатой.

Для вычисления глубины плоскости в точке с известными координатами (x1,y1) можно воспользоваться уравнением плоскости Ax+By+Cz+D=0, откуда получить

$$z = -\frac{Ax1+By1+D}{C} \qquad (C<>0)$$

Если же C=0, то плоскость параллельна оси Z. В этом случае глубина находится из уравнений ребер многоугольника, которые могут пересекать прямую, параллельную оси Z и проходящую через точку (x1,y1). В качестве глубины берется глубина той точки пересечения, у которой координата Z оказалась максимальной.

Области удобнее брать квадратными с размерами сторон (в пикселах), представляющими степень двойки. Для выполнения этого условия можно взять область, несколько превышающую по размерам экран дисплея. При этом подобласти, лежащие за пределами экрана, анализировать не надо (чтобы не проводить ненужных вычислений). Ошибки не будет, если такие области будут проанализирова-
ны обычным порядком, так как при их изображении координаты не
будут соответствовать допустимым и изображаться ничего не будет.

Другим вариантом является разбиение относительно вершины многоугольника (если существует вершина, попадающая в область). Этим делается попытка избежать ненужных разбиений.

Следует отметить, что не существует единого алгоритма Варнока. Конкретные реализации этого алгоритма различаются в деталях. В простейшем варианте алгоритма Варнока любое окно размером больше пиксела всегда разбивается, если оно не пусто. В этой версии алгоритма для идентификации внешних многоугольников используется простой габаритный тест с прямоугольной оболочкой для областей, размер которых больше одного пиксела. Лишь для окон размером с пиксел выполняется более сложный тест на внешность.

Более сложные варианты алгоритма используют перечисленные тесты. При этом целесообразно иметь три списка многоугольников:
1)охватывающих; 2)внешних; 3)пересекающих и внутренних. При построении этих списков запоминается уровень (шаг разбиения), на котором многоугольник попал в тот или иной список.

Рассмотрим тесты, позволяющие распознать тип многоугольника. Простой габаритный тест выявляет факт внешности многоугольника по отношению к прямоугольному окну на основе сравнения координат окна с координатами прямоугольной оболочки многоугольника. Если Хл, Хп - абсциссы левой и правой границ, а Yн, Yв - ординаты нижней и верхней границ области, а Xmin, Xmax,
Ymin, Ymax - координаты аналогичных границ оболочки, то многоугольник внешний, если истинно следующее выражение:

(Xmin>Хп) (Xmax<Хл) (Ymin>Yв) (Ymax<Yн)

Многоугольник будет внутренним, если его об"емлющая оболочка лежит внутри области, т.е. должно быть истинным выражение

(Xmin> Хл)&(Xmax< Хп)&(Ymin> Yн)&(Ymax< Yв).

Для определения пересекающих многоугольников можно подставить координаты вершин окна в пробную функцию, представляющую собой уравнение прямой, несущей ребро многоугольника. Эта функция имеет вид

$$f=Ax+By+C,$$

где A, B, C - коэффициенты уравнения прямой.

Если знак этой функции не зависит от выбора вершины окна, то все его вершины лежат по одну сторону от несущей прямой и на ней нет точек пересечения с рассматриваемой областью. Если ни одно из ребер многоугольника не пересекает окна, то этот многоугольник является либо внешним, либо охватывающим окно. При проведении этого теста надо учитывать тот факт, что используется уравнение бесконечной прямой, а не конечного отрезка. Поэтому для ребра, которое может пересекаться с окном, надо дополнительно вычислить координаты точки пересечения со сторонами окна. Если полученная точка принадлежит ребру, то многоугольник пересекает окно. В противном случае он является либо внешним, либо охватывающим (внутренние многоугольники к этому моменту уже определены).

На основе теста с прямоугольной оболочкой и теста с пробной функцией выявляются внутренние, пересекающие и часть внешних многоугольников. Часть внешних многоугольников (например, огибающих угол окна) не может быть отделена от охватывающих многоугольников. Поэтому требуются дополнительные тесты для выявления внешних и охватывающих многоугольников (рис.).

В первом тесте проводится луч из любой точки окна (удобно из вершины) в бесконечность. Затем подсчитывается количество пересечений луча с многоугольником. При четном (или равным нулю) количестве пересечений многоугольник является внешним, при нечетном - охватывающим.

При прохождении луча через вершину многоугольника возникает неопределенность. Ее можно устранить, если считать касание за два пересечения, а протыкание - за одно.

Во втором тесте осуществляется подсчет угла. Из произвольной точки окна (лучше из центра) проводятся лучи в начало и конец каждого ребра многоугольника. При этом находится сумма углов, образованных такими лучами для каждого ребра. Обход по ребрам многоугольника совершается по или против часовой стрелки. Полученная сумма интерпретируется следующим образом:

ALFAi=0 - многоугольник внешний по отношению к окну.
(ALFAi - угол, образованный лучами, проведенными в конец i-го ребра).

ALFAi=360m - многоугольник охватывает окно m раз (многоугольник без самопересечений может охватить точку только один раз).

Процесс вычисления суммы можно упростить, так как нет нужды подсчитывать углы с высокой точностью (достаточно ограничиться приращениями по 45 , т.е. считать целые октанты, покрытые отдельными углами). Октанты нумеруются от 0 до 7. Они получаются, если считать стороны окна бесконечными прямыми).
Число целых октантов, покрытых углом, равно разности между номерами октантов, в которых лежат концы его ребер. При этом

ALFA= ALFA-8, если ALFA>4
ALFA= ALFA +8, если ALFA<-4

Если ALFA=+4, то сторона окна расщепляет ребро многоугольника, поэтому ребро в этом случае надо разбить на два стороной окна (в противном случае получаются одинаковые результаты для внешнего и охватывающего многоугольника).

На основе суммирования вкладов отдельных ребер получим

$$S=\ ALFAi = \begin{cases} 0 - \text{многоугольник внешний по отношению к окну} \\ +8m - \text{многоугольник охватывает окно} \end{cases}$$

Для выпуклых тел целесообразно предварительно удалить нелицевые грани, как это делается в алгоритме Робертса.

6.3. АЛГОРИТМ ВЕЙЛЕРА-АЗЕРТОНА

Алгоритм Вейлера-Азертона развивает идеи алгоритма Варнока в направлении минимизации количества шагов при разбиении окон на подокна. Основой алгоритма служит алгоритм отсечения многоугольников на плоскости тех же авторов. Для повышения точности алгоритм работает в об"ектном пространстве, выходными данными являются многоугольники, поэтому его можно использовать как для удаления невидимых линий, так и для удаления невидимых поверхностей.

Алгоритм удаления невидимых поверхностей состоит из четырех шагов:

1. Предварительная сортировка по глубине.

В результате выполнения этого шага многоугольники упорядочиваются в порядке уменьшения максимального значения координаты Z и образуют некоторый приоритетный список. Считается, что наблюдатель располагается в бесконечности на положительной полуоси Z, поэтому многоугольник с наивысшим приоритетом окажется ближайшим к наблюдателю. В простейшем случае (если ближайший многоугольник не пересекается другими многоугольниками или полностью лежит ближе всех других многоугольников) он заслоняет все остальные многоугольники или их части. Поэтому область экрана, на которую проецируется первый многоугольник, можно закрасить цветом этого многоугольника.

Поскольку самый приоритетный многоугольник может оказаться "маленьким" и не будет закрывать все остальные многоугольники, то необходимо выполнить второй шаг алгоритма.

2. Отсечение по границе ближайшего к наблюдателю многоугольника (сортировка многоугольников на плоскости).

В качестве отсекателя используется копия самого приоритетного многоугольника из списка, полученного на первом шаге. Отсекаются все остающиеся в приоритетном списке многоугольники (в том числе и первый). С помощью алгоритма отсечения Вейлера- Азертона [1, с.315] проводится отсечение по границам отсекателя, в результате чего формируются два списка - внутренних и внешних многоугольников.

Фактически отсечение проводится с проекциями на плоскость XOY отсекателя и отсекаемого многоугольника (двумерная операция отсечения). Часть отсекаемого многоугольника (если она есть), попавшая внутрь отсекателя, добавляется к списку внутренних
многоугольников. Оставшаяся часть (находящаяся за пределами отсекателя) добавляется к списку внешних многоугольников.

Этот шаг представляет собой сортировку на плоскости или XY -сортировку.

3. Удаление многоульников, экранированных многоульником, ближайшим к точке наблюдения.

На этом шаге алгоритма сравниваются глубины каждого многоугольника из внутреннего списка с глубиной отсекающего многоугольника. Сначала для каждого многоугольника определяются коэффициенты уравнения несущей плоскости. Для каждой вершины каждого многоугольника с помощью полученных уравнений плоскостей вычисляются глубины (значения координаты Z). Полученные значения сравниваются с минимальным значением координаты Z (Zотс.min) отсекающего многоугольника. Если глубина ни одной из
вершин многоугольников из внутреннего списка не больше Zотс.min,
то все эти многоугольники экранируются отсекающим многоугольником. И в итоге должен быть изображен отсекающий многоугольник.

Затем аналогично рассматривается внешний список многоугольников.

Если же найдутся многоугольники, частично экранирующие наиболее приоритетный многоугольник в списке, то необходимо выполнить четвертый шаг алгоритма.

4. Рекурсивное подразбиение и окончательная сортировка для устранения неопределенностей.

Если координата Z какого-либо отсекаемого многоугольника окажется больше, чем Zотс.min, то такой многоугольник частично
экранирует отсекающий многоугольник. В таком случае результат
предварительной сортировки ошибочен, и алгоритм рекурсивно под-

разделяет плоскость (X,Y), используя многоугольник, нарушивший порядок, в качестве нового отсекающего многоугольника. Отсечению подвергаются многоугольники из внутреннего списка, причем старый отсекающий многоугольник сам подвергается отсечению.

Новый отсекающий многоугольник является копией исходного многоугольника, а не его остатка после предыдущего отсечения.

Дополнительно следует рассмотреть случай циклического перекрывания многоугольника и отсекающего многоугольника. Все экранируемое циклическим многоугольником удаляется на первом шаге отсечения. необходимо лишь произвести отсечение исходного многоугольника по границам циклического многоугольника, а затем изобразить полученный результат.

Для пересекающихся многоугольников с целью избежания зацикливания при построении приоритетного списка в качестве отсекателя следует брать часть многоугольника, ограниченную линией пересечения многоугольников. В этом случае получим два списка - внутренних и внешних многоугольников, для которых зацикливание не происходит.

6.4. АЛГОРИТМ, ИСПОЛЬЗУЮЩИЙ СПИСОК ПРИОРИТЕТОВ

Основная идея этого алгоритма заключается в упорядочении многоугольников в соответствии с их удаленностью от точки зрения и получении списка многоугольников, в котором бы никакие два элемента не перекрывали бы друг друга.

В этом случае все многоугольники можно последовательно разложить в растр, начиная просмотр списка с наиболее удаленных многоугольников. Ближайшие к наблюдателю многоугольники преобразуются в растровую форму в последнюю очередь и закрывают более удаленные многоугольники, поскольку записываются в буфер кадра (или изображаются на экране) поверх старых.

Рассматриваемый алгоритм называют еще алгоритмом художника, поскольку он аналогичен тому способу, которым художник создает картину. Сначала художник рисует фон, затем предметы, лежащие на среднем расстоянии и в последнюю очередь - передний план. Таким образом художник решает задачу об удалении невидимых поверхностей путем построения картины в порядке обратного приоритета. Алгоритм состоит из трех основных шагов.

1. Упорядочение всех многоугольников в соответствии с их наибольшим (или наименьшим) значением Z координаты.

На этом шаге алгоритма формируется приоритетный список многоугольников. Упорядочение можно проводить либо в соответствии с возрастанием максимального значения координаты Z многоугольника, либо по возрастанию минимального значения координаты Z. Предполагается, что такой многоугольник лежит дальше всех от точки наблюдения.

2. Разрешение неопределенностей, вызванных перекрытием Z-оболочек.

Будем считать для определенности, что плоскости упорядочены по возрастанию минимального значения координаты Z. Обозначим через P плоскость, стоящую в приоритетном списке на первом месте (она имеет min Zmini), а через Q - плоскость, стоящую на втором месте.

Тогда возможна ситуация, при которой плоскость P полностью или частично экранирует Q. Неопределенность возникает при циклическом перекрывании плоскостей, например, P находится впереди Q, причем Q лежит впереди R, а R в свою очередь находится перед P. Неопределенность возникает также и при протыкании многоугольников.

В отмеченных случаях окончательный список приоритетов не удается установить сразу. Для проверки правильности сформированного списка следует для каждого многоугольника P проверить его отношение с Q. Многоугольник P не может экранировать многоугольник Q, если его ближайшая к наблюдателю вершина (Pzmax) лежит дальше, чем самая удаленная вершина Q(Qzmin), т.е. Qzmin>Pzmax.

Если же Qzmin<Pzmax, то многоугольник P потенциально может экранировать многоугольник Q и любой другой многоугольник, подобный Q, для которого выполняется приведенное соотношение. Если же P фактически не экранирует Q, то P можно заносить в буфер кадра.

Для выяснения фактического взаимного расположения многоугольников P и Q необходимо выполнить проверку, представляющую собой тест из пяти шагов. Эти тестовые проверки упорядочены по возрастанию сложности их выполнения, причем при получении положительного ответа на очередном шаге теста многоугольник P можно сразу преобразовать в растровую форму и дальнейшие проверки не проводить.

Пятью тестами являются следующие:

1. X-оболочки многоугольников не перекрываются, поэтому сами многоугольники также не перекрываются. положительный ответ здесь получается, если истинно следующее выражение:

(Pxmax<Qxmin) (Qxmax<Pxmin).

2. Y-оболочки многоугольников не перекрываются, поэтому

сами многоугольники также не перекрываются. Положительный ответ в этом тесте получается, если истинно соотношение

(Pymax<Qymin) (Qymax<Pymin)

3. P целиком лежит с той стороны от плоскости Q, которая дальше от точки расположения наблюдателя. Для выполнения этого теста необходимо определить коэффициенты в уравнении плоскости Q:

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

Затем в полученное уравнение подставить поочередно координаты всех вершин многоугольника P. Если знаки результатов для всех вершин совпадают и совпадают со знаком результата при подстановке координат пробной точки, заведомо лежащей за плоскостью Q, то ответ на тест дается положительный. Если при подстановке координат точек получаемые значения равны нулю, то многоугольник P лежит на плоскости Q.

4. Q целиком находится с той стороны от плоскости P, которая ближе к точке расположения наблюдателя. Для реализации этого теста необходимо выполнить действия, аналогичные тем, которые выполняются в предыдущем тесте. Надо, во-первых, определить коэффициенты A, B, C, D уравнения плоскости, проходящей через многоугольник P. Во-вторых, подставить координаты всех вершин многоугольника Q в полученное уравнение и определить знаки полученных результатов. Если знаки всех результатов одинаковы и совпадают со знаком результата при подстановке пробной точки, заведомо лежащей перед плоскостью P, то ответ на этот тест дается утвердительный. Если получаемые значения равны нулю, то многоугольник Q лежит на плоскости P.

5. Проекции многоугольников на плоскость XOY (экран) не

перекрываются. Выполнение этого теста фактически означает проведение теста на внешность и может выполняться, как в алгоритме Варнока (например, с бесконечным лучом).

Все указанные тесты должны применяться к каждой плоскости типа Q. Если во всех пяти тестах ролучен отрицательный ответ, то предполагается, что P действительно закрывает Q, поэтому P и Q меняют местами в списке. При этом необходимо отметить, что многоугольник Q был перемещен на новое место.

Для измененного списка повторяются указанные тесты. В некоторых случаях изменение порядка плоскостей в списке может привести к правильному результату. Если же имеется взаимное экранирование нескольких плоскостей или их протыкание, то перестановка к успеху не приводит. Именно в этом случае придем к необходимости новой перестановки многоугольника Q, что и будет означать факт перекрытия или протыкания (рис.).

В этом случае необходимо многоугольник P разрезать плоскостью, несущей Q, на две части. При этом исходный многоугольник P удаляется из списка, а две его новые части включаются в список на соответствующие места, и тесты повторяются для нового списка.

Для разбиения многоугольника вдоль линии пересечения несущих эти многоугольники плоскостей можно использовать алгоритм Сазерленда-Ходжмена (если отсекатель выпуклый) или алгоритм Вейлера-Азертона в общем случае. Многоугольник Q используется как

отсекатель. Многоугольник P разбивается на два новых. Для нахождения точек пересечения ребер многоугольника P с отсекателем Q можно использовать алгоритм Кируса-Бека [ ].

Алгоритм, использующий список приоритетов, может работать

как в об"ектном пространстве, так и в пространстве изображения. Формирование списка приоритетов ведется в об"ектном пространстве, а результат заносится в буфер кадра в терминах пространства изображения. При этом необходимо произвести масштабирование (переход из пространства мировых координат в пространство экранных координат).

Данный алгоритм может применяться и для удаления невидимых линий (каркасная модель изображения). При этом в буфер кадра заносятся атрибуты, соответствующие цвету фона. При разложении многоугольника в растр его ребрам присваиваются атрибуты, отличающиеся от цвета фона, а все внутренние пикселы получают цвет фона. В этом случае новый многоугольник, закрывающий ранее рассмотренный многоугольник, при разложении в растр "сотрет" ребра предыдущего многоугольника (они будут закрашены цветом фона).

6.5. АЛГОРИТМ, ИСПОЛЬЗУЮЩИЙ Z-БУФЕР

Данный алгоритм удаления невидимых поверхностей является одним из простейших. Этот алгоритм работает в пространстве изображения. Здесь обобщается идея о буфере кадра. Буфер кадра используется для заполнения атрибутов (интенсивности) каждого пиксела в пространстве изображения. Наряду с буфером кадра вводится Z-буфер, представляющий собой специальный буфер глубины, в котором запоминаются координаты Z (глубина) каждого видимого пиксела в пространстве изображения.

В процессе работы глубина (значение координаты Z) каждого нового пиксела, который надо занести в буфер кадра, сравнивается с глубиной того пиксела, который уже занесен в Z-буфер. Если
это сравнение показывает, что новый пиксел расположен ближе к
наблюдателю, чем пиксел, уже находящийся в буфере кадра, то новый пиксел заносится в буфер кадра. Помимо этого производится корректировка Z-буфера: в него заносится глубина нового пиксела. Если же глубина (значение координаты Z) нового пиксела меньше, чем хранящегося в буфере, то никаких действий производить не надо. В сущности алгоритм для каждой точки (x,y) находит наибольшее значение функции Z(x,y).

Этот алгоритм несмотря на свою простоту позволяет удалять сложные поверхности и позволяет визуализировать пересечения таких поверхностей. Сцены могут быть произвольной сложности, а поскольку размеры изображения ограничены размером экрана дисплея, то трудоемкость алгоритма имеет линейную зависимость от числа рассматриваемых поверхностей. Элементы сцены заносятся в буфер кадра в произвольном порядке, поэтому в данном алгоритме не тратится время на выполнение сортировок, необходимых в других алгоритмах.

Главный недостаток алгоритма - большой об"ем требуемой памяти. Буфер кадра совместно с Z-буфером требует около 1,5 мегабайт памяти (при разрешающей способности 640*480 пикселов, 8 битах для хранения цвета пиксела и 32 битах для хранения глубины).
Один из путей решения этой проблемы - использование внешней памяти, но это может существенно замедлить работу. Другой путь экономии памяти - использование Z-буфера размером в одну строку (алгоритм построчного сканирования).

Еще два недостатка данного алгоритма - трудоемкость устранения лестничного эффекта и невозможность реализации эффектов
прозрачности.

Формально описать алгоритм можно следующим образом:

1. Заполнение буфера кадра фоновым значением интенсивности (цвета).

2. Заполнение Z-буфера минимальным значением Z.

3. Преобразование каждого многоугольника в растровую форму в произвольном
порядке.

4. Вычисление для каждого пиксела с координатами (x,y), принадлежащего многоугольнику, его глубины Z(x,y).

5. Сравнение глубины Z(x,y) со значением Zбуф(x,y), хранящимся в Z-буфере для
пиксела с теми же координатами (x,y):

если Z(x,y)>Zбуф(x,y), то записать атрибут очередного многоугольника в буфер кадра и Zбуф(x,y) заменить на значение Z(x,
y).

Предварительно для выпуклых многогранников целесообразно удалить нелицевые грани (выполнить первый этап алгоритма Робертса).

Для вычисления глубины каждого пиксела на сканирующей строке можно поступить следующим образом: уравнение плоскости, несущей многоугольник, имет вид

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

Отсюда при C<>0 Z=-(Ax+By+D)/C. Для сканирующей строки y=const, глубина пиксела, для которого x1=x+ x, равна:

$$z1=-\frac{Ax+By+D}{C}-\frac{A\,x}{C}=z-\frac{A}{C}\,x$$

Поскольку x=1, то z1=z-A/C. Если же C=0, то плоскость многоугльника параллельна оси Z. (Для наблюдателя такой много-
угольник вырождается в линию). Глубина пиксела, являющегося пересечением сканирующей строки с ребром многоугольника, вычисляется следующим образом. Сначала определяются ребра грани, вершины которых лежат по разные стороны от сканирующей строки (одна из вершин ребра может в крайнем случае лежать на сканирующей строке), так как только в этом случае сканирующая строка пересекает ребро. Затем из найденных точек пересечения выбирается ближайшая к наблюдателю. Глубина точки пересечения определяется из соотношения

$$z3=z2+\frac{y3-y2}{y2-y1}(z2-z1)$$

где (y1,z1) и (y2,z2) - координаты вершин проекции ребра на плоскость YOZ;
(y3,z3) - координаты проекции точки пересечения на ту же
плоскость.

Уравнение проекции ребра

$$\frac{z-z2}{z2-z1}=\frac{y-y2}{y2-y1}$$

Уравнение проекции плоскости y=const (y=y3).

Алгоритм, использующий Z-буфер, можно использовать для построения разрезов поверхностей. В этом случае изменяется только сравнение глубины пиксела со значением, занесенным в буфер:

$$(Z(x,y)>Zбуф(x,y))\&(Z(x,y)<Zразр),$$

где Zразр - глубина искомого разреза. В этом случае оста-

ются только элементы поверхности, которые лежат на плоскости
разреза или позади нее.

Для построения прозрачных поверхностей может применяться модифицированный вариант Z-буфера - ALFA-буфер. Если рассматриваются монохромные почти прозрачные поверхности, то в этом случае достаточно наряду с Z-буфером иметь еще один буфер с информацией о текущей интенсивности пиксела. При построении непрозрачной поверхности (как и в Z-буфере) используется Z-буфер, при этом обновляется информация и в ALFA-буфере. При построении прозрачной поверхности расстояние до очередного пиксела изображения сравнивается с элементом Z-буфера, и он строится, если находится ближе к наблюдателю, при этом элемент Z-буфера не модифицируется, а в элемент ALFA-буфера заносится информация о новом цвете пиксела по правилу:

$$I=T*Io$$

$$T=(1-(4piD+M))exp(-S*l),$$

где I - интенсивность пиксела,
T - коэффициент прозрачности,
D,M - коэффициенты диффузного и зеркального отражения,
S - коэффициент поглощения,

l - путь луча в рассматриваемом слое,

Io - текущее содержимое ALFA-буфера.

(коэффициент прозрачности должен быть близок к 1).

Ограничение на величину коэффициента прозрачности T позволяет не учитывать относительное расположение прозрачных поверхностей и вычислять результирующую интенсивность пиксела в линейном приближении.

В заключение можно отметить, что эксплуатационные характеристики алгоритма с Z-буфером остаются практически постоянными, т.к. с ростом числа многоугольников в видимом об"еме уменьшается число пикселов, покрываемых одним многоугольником.

6.6. АЛГОРИТМ ПОСТРОЧНОГО СКАНИРОВАНИЯ

Данный алгоритм работает в пространстве изображения. Он обобщает идеи растровой развертки многоугольников. Алгоритм сводит трехмерную задачу удаления невидимых линий и поверхностей к двумерной. Сканирующая плоскость определяется точкой наблюдения, расположенной в бесконечности на положительной полуоси Z, и сканирующей строкой, соответствующей очередной строке экрана дисплея.

Пересечение сканирующей плоскости и трехмерной сцены определяет окно размером в одну сканирующую строку. Задача удаления невидимых поверхностей решается в пределах этого окна, образованного сканирующей плоскостью. Сложность при использовании идей растровой развертки многоугольников связана с тем, что здесь нельзя непосредственно использовать алгоритм с упорядоченным списком ребер, так как это приведет к некорректным результатам там, где строка пересекает два многоугольника и более.

Основная идея алгоритма достаточна проста. Как и в предыдущем алгоритме, создается Z-буфер, но меньший по об"ему. Об"ем буфера соответствует количеству пикселов одной строки экрана. Первоначально этот буфер заполняется минимальным значением Z, а в буфер кадра заносится фоновое значение интенсивности.

Затем определяется пересечение сканирующей строки с проекцией каждого многоугольника на плоскость XOY (если эти пересечения существуют). Пересечения образуют пары, поэтому в интервале между концами пар глубина многоугольника сравнивается с глубиной, содержащейся в Z-буфере. Если глубина рассматриваемого пиксела многоугольника больше значения из Z-буфера, то рассматриваемый многоугольник будет текущим видимым. Атрибуты этого многоугольника заносятся в буфер кадра в текущую позицию, одновременно корректируется и Z-буфер. После обработки всех многоугольников сцены буфер размером в одну строку содержит решение задачи об удалении невидимых поверхностей для данной сканирующей строки. Содержимое буфера кадра может выводиться на экран.

Однако сравнение каждого многоугольника с каждой сканирующей строкой является неэффективным. Целесообразнее использовать список активных ребер [1, с.99]. Но в отличие от растровой развертки одного многоугольника здесь осуществляется работа со всеми многоугольниками об"екта.

Алгоритм может быть сформулирован следующим образом:

1. Подготовка исходных данных.

а) Создание списка активных многоугольников. Для этого определяется самая верхняя сканирующая строка, которую пересекает многоугольник, и заносится многоугольник в группу Y, соответствующую этой сканирующей строке. Для каждого многоугольника создается таблица (список), содержащая следующую информацию: коэффициенты A, B, C, D уравнения плоскости многоугольника; Yмн - число сканирующих строк, пересекаемых многоугольником; атрибуты (цвет, интенсивность) многоугольника; Zл - глубину многоугольника для пиксела, соответствующего левому ребру; Zx - приращение по Z вдоль сканирующей строки (Z=-A/C, C<>0); Zy - приращение по Z между сканирующими строками (Zy=- B /C, C<>0).

Если C=0, то плоскость параллельна оси Z, и линия пересечения сканирующей плоскости и рассматриваемой плоскости проецируется в точку. В этом случае значение координаты Z надо вычислить для точки пересечения сканирующей строки с ребром многоугольника, ближайшим к наблюдателю (так же, как в алгоритме Варнока и Z-буфера).

б) Создание списка активных ребер для всех негоризонтальных ребер. Элементы списка должны быть отсортированы по группам на основе меньшей Y-координаты (если считать, что верхней строке экрана соответствует минимальное значение Y-координаты).

в) Запоминание по каждому ребру в виде таблицы (списка) следующей информации: X-координаты вершины с наименьшей Y-координатой; X -приращения координаты X ребра при переходе к соседней сканирующей строке; Y - количества сканирующих строк, пересекаемых ребром; идентификатора многоугольника, показывающего принадлежность ребра многоугольнику.

2. Удаление невидимых поверхностей.

а) Инициализация буфера кадра размером с одну сканирующую строку дисплея для всех пикселов цветом фона.

б) Инициализация Z-буфера размером с одну сканирующую строку путем занесения в него значения Zmin.

в) Проверка для очередной сканирующей строки появления новых многоугольников (соответствующая Y-группа должна быть не пуста). Добавление новых многоугольников к списку активных многоугольников.

г) Проверка появления новых многоугольников в списке активных многоугольников. Добавление всех пар ребер новых
многоугольников к списку активных ребер.

д) Проверка удаления ребра из списка активных ребер. Проверка сохранения многоугольника в списке активных многоугольников. Укомплектование пары активных ребер многоугольника в списке активных ребер при сохранении многоугольника в списке активных многоугольников.

е) Упорядочивание пересечений слева направо (по возрастанию абсциссы). (Пары активных ребер многоугольников заносятся в список в произвольном порядке.) Для невыпуклых многоугольников может оказаться более одной пары активных ребер.

ж) Выполнение для каждой пары ребер многоугольника из списка активных ребер следующих действий:

- извлечение пары ребер многоугольника из списка активных ребер;
- инициализация Z со значением Zл;
- вычисление глубины для каждого пиксела, лежащего в интервале Xл<X<Xп (Xл, Xп - абсциссы точек пересечения левого и правого ребер пары со сканирующей строкой): для очередной точки текущей сканирующей строки Zx+ x=Zx- Zx;
- сравнение глубины Z(x) с величиной Zбуф(x) для те-
кущей строки. При Z(x)>Zбуф(x) занесение атрибутов многоугольника в буфер кадра для сканирующей строки и замена Zбуф(x) на Z(x);
- запись буфера кадра для сканирующей строки в буфер кадра дисплея.
- коррекция списка активных ребер:

Yл= Yл-1; Yп= Yп-1. При Yл<0 или Yп<0 удаление соответствующего ребра из списка; индикация обоих ребер и соот-
ветствующего многоугольника;

- вычисление новых абсцисс точек пересечения:

Xл=Xл+ Xл; Xп=Xп+ Xп;

- вычисление глубины многоугольника на левом ребре с помощью уравнения плоскости многоугольника:

Zл=Zл- Zx* X- Zy;

- удаление многоугольника из списка активных многоугольников при Yмн<0.

Перед началом работы алгоритма целесообразно удалить нелицевые грани.

6.7.АЛГОРИТМ ОПРЕДЕЛЕНИЯ ВИДИМЫХ ПОВЕРХНОСТЕЙ ПУТЕМ ТРАССИРОВКИ ЛУЧЕЙ

Данный алгоритм в отличие от уже рассмотренных не учитывает специфику обрабатываемого об"екта. Данный метод получил право на жизнь только в условиях современных высокопроизводительных технических средств, его эффективность особенно высока при наличии многопроцессорных вычислительных комплексов.

Основная идея этого метода заключается в том, что наблюдатель видит об"ект благодаря световым лучам, испускаемым некоторым источником, которые падают на об"ект. Свет достигает наблюдателя, если он отражается от поверхности, преломляется или проходит через нее.

Если наблюдать лучи, выпущенные источником, то можно убедиться, что лишь малое их количество дойдет до наблюдателя, поэтому такой подход в вычислительном плане весьма неэффективен. Более эффективным является подход, при котором лучи отслеживаются (трассируются) в обратном направлении, т.е. от наблюдателя к об"екту.

Здесь рассмотрим использование трассировки лучей только для определения видимых поверхностей. В целом же с помощью трассировки лучей можно учитывать эффекты отражения света от об"ектов, преломления, прозрачности, затенения.

Алгоритм работает в пространстве изображения, наблюдатель находится в бесконечности на положительной полуоси Z. В более простом варианте перспективное преобразование не учитывается.

Лучи, идущие от наблюдателя, параллельны в этом случае оси Z. Необходимо проследить траекторию луча, чтобы определить об"екты сцены, с которыми этот луч пересекается, и вычислить глубину пересечения с каждым об"ектом для каждого пиксела (лучей будет столько, сколько пикселей на экране). Из всех об"ектов, с которыми пересекается луч, видимым для данного пиксела будет тот, пересечение с которым имеет наибольшую глубину (максимальное значение координаты Z). И данный пиксел надо закрасить цветом об"екта с этой максимальной глубиной.

Если наблюдатель находится не в бесконечности, алгоритм усложняется несущественно. Задача сводится к построению одноточечной центральной проекции.

Центральным пунктом алгоритма является процедура определения пересечений луча с поверхностью об"екта. В сцену можно включать любые об"екты, для которых можно реализовать процедуру построения пересечений: плоские многоугольники, многогранники, тела, ограниченные квадратичными или биполиномиальными параметрическими поверхностями. Эффективность данной процедуры оказывает самое большое влияние на эффективность всего алгоритма, так как более 75% всего времени затрачивается на определение пересечений.

Для исключения ненужного поиска пересечений в алгоритме предлагается искать сначала пересечение луча с об"емлющей оболочкой об"екта. Если луч не пересекает оболочку, то он не пересекает и сам об"ект.

В качестве оболочек удобнее всего использовать сферу или прямоугольный параллелепипед.

Использование сферы может оказаться неэффективным, так как об"ем описанной сферы может существенно превышать об"ем об"екта. Но тест со сферической оболочкой чрезвычайно прост: достаточно определить расстояние от центра сферы до луча. При использовании параметрического представления прямой, проходящей через точки P1(x1,y1,z1) и P2(x2,y2,z2), получим

$$P(t)=P1+(P2-P1)t \text{ или } \quad x=x1+(x2-x1)t=x1+at$$
$$y=y1+(y2-y1)t=y1+bt$$
$$z=z1+(z2-z1)t=z1+ct$$

Если центр сферы лежит в точке с координатами Po(Xo,Yo,Zo), то квадрат расстояния от него до прямой:

$$d =(X1+at-Xo) + (Y1+bt-Yo) + (Z1+ct-Zo).$$

Значение параметра t, при котором это расстояние минимально, находится из условия

$$\frac{d\,d(t)}{dt} = 0$$

$$2(X1-Xo+at)a+2b(Y1-Yo+bt)+2c(Z1-Zo+ct)=0 \quad a(X1-Xo)+b(Y1-Yo)+c(Z1-Zo)$$

$$t = -\frac{a(X1-Xo)+b(Y1-Yo)+c(Z1-Zo)}{a+b+c}$$

Если d>R , где R-радиус сферической оболочки, то луч не пересекает оболочку.

Тест с прямоугольной оболочкой в трехмерном пространстве требует проверки как минимум с тремя бесконечными плоскостями, ограничивающими прямоугольную оболочку. Кроме того, точки пересечения могут оказаться за пределами граней параллелепипеда, поэтому надо провести еще дополнительную проверку на попадание точки пересечения внутрь параллеллепипеда или за его пределы. В силу этого тест со сферической оболочкой оказывается быстрее.

Упростить вычисление пересечений луча с об"ектом и осуществить сортировку по глубине можно путем совмещения луча с осью Z. Для этого используются переносы и повороты вокруг координатных осей.

В результате пересечение луча с оболочкой будет иметь место, если знаки максимальных и минимальных координат оболочки противоположны. Поверхность второго порядка в декартовой системе координат описывается уравнением

P(x,y,z)=A1x +A2y +A3z +B1xy+B2yz+B3xz+C1x+c2y+c3z=0 ( ) После выполнения преобразований для совмещения луча с

осью Z возможные пересечения луча с поверхностью будут иметь место при x=y=0. Поэтому координаты точек пересечения будут находиться из уравнения

$$z = \frac{-C3' + \sqrt{C3'^2 - 4A3'D'}}{2A3'} \qquad A3'z^2 + C3'z + D' = 0 \qquad (\ )$$

Коэффициенты со штрихом означают коэффициенты общего уравнения поверхности ( ) после преобразования. Если подкоренное выражение в ( ) отрицательно, то луч не пересекает поверхность. Для нахождения точки пересечения в исходной системе координат необходимо выполнить обратное преобразование.

Если после нахождения точки пересечения с бесконечной поверхностью требуется проверить попадание точки внутрь ограничивающей поверхности, то эту проверку можно произвести на двумерной проекции фигуры.

Формально алгоритм трассировки лучей можно записать следующим образом:

1. Подготовка исходных данных.

Создание списка об"ектов со следующей информацией:

- тип об"екта; характеристика ограничивающей поверхности; атрибуты об"екта (цвет, интенсивность);
- параметры сферической оболочки (координаты ее центра и радиус);
- признак необходимости выполнения теста с прямоугольной оболочкой;
- параметры прямоугольной оболочки Xmin, Xmax, Ymin, Ymax, Zmin, Zmax.

2. Выполнение следующих действий для каждого трассируемого луча:

- выполнение теста со сферической оболочкой для каждого об"екта сцены в исходной системе координат. Занесение об"екта в список активных об"ектов при пересечении луча с оболочкой;
- закраска пиксела фоновым цветом при пустом списке активных об"ектов и переход к следующему лучу;
- проведение преобразования для совмещения луча с осью Z.

3. Выполнение следующих операций для каждого об"екта из списка активных об"ектов:

- преобразование прямоугольной оболочки в новую систему координат и выполнение теста на пересечение луча с этой оболочкой (при наличии признака проведения теста);

- переход к следующему об"екту при отсутствии пересечения;
- преобразование об"екта к новой системе координат и вычисление пересечения с лучом при наличии пересечения;
- занесение точки пересечения в список пересечений;
- изображение пиксела фоновым цветом при отсутствии пересечений;
- определение Zmax для найденных точек пересечения;
- вычисление обратного преобразования;
- определение точки пересечения в исходной системе координат с использованием обратного преобразования;
- закраска пиксела с учетом атрибутов пересеченного об"екта и модели освещенности.

Алгоритм упрощается, если не учитывается ориентация поверхности в пространстве и ее свойства. В этом случае нет необходимости определять точку пересечения в исходной системе координат и искать преобразование.

Эффективность алгоритма можно повысить, если ввести кроме

оболочек для каждого об"екта, оболочки для групп об"ектов и даже для всех об"ектов сцены. Если луч не пересекает оболочку всей сцены, то пиксел, соответствующий этому лучу, может быть сразу изображен фоновым цветом.

Если луч пересекается с оболочкой сцены, то исследуются пересечения его с оболочками для каждой группы об"ектов. При пересечении луча с некоторой оболочкой алгоритм рекурсивно повторяется, пока не будут рассмотрены все об"екты.

Другая модификация алгоритма, повышающая его эффективность, состоит в том, что точки пересечения не ищутся сразу для каждого об"екта, с которым пересекается луч. Предварительно список об"ектов, с которым пересекается луч, упорядочивается по приоритету глубины. После этого ищутся точки пересечения только с потенциально видимыми об"ектами (их число обычно намного меньше общего числа об"ектов в списке пересеченных лучом). Необходимость поиска пересечения луча не только с самым приоритетным об"ектом об"ясняется тем, что точка пересечения конкретного луча с об"ектом будет не обязательно видимой.

6.8. Принципы построения полутоновых изображений. Разработано множество способов закрашивания [ ]: гранением, пропорциональное закрашивание, закрашивание по способу Гуро, закрашивание по способу Фонга, закрашивание способом трассировки лучей (лучей зондирования). Эти способы требуют различного количества процессорного времени и, следовательно, обеспечивают различное качество изображения.

Самый простой способ закрашивания назывется гранением (faceting). Он требует сравнительно небольших ресурсов компью-

тера, поскольку предполагает лишь заполнение каждого из многоугольников одним цветом или оттенком. Однако способ гранения слишком примитивен; закрашенные этим способом объекты выглядят не плавными, а так как если бы они были покрыты рыбными чешуйками.

Линейное, или пропорциональное закрашивание, для которого требуется больше вычислительной мощности, чем для способа гранения, предполагает изменение освещенности в пределах каждого многоугольника и благодаря этому позволяет воспроизводить более реалистичные изображения. Здесь яркость и цветовая насыщенность элементов каждого многоугольника плавно меняются в интервале между значениями, вычисленными для его вершин. При этом поверхности воспроизводимых предметов преобретают идеальную сюрреалистическую гладкость, как будто мы видим их при непрямом освещении.

Более реалистические изображения получаются в случае,если яркость и цветовая насыщенность каждого многоугольника плавно меняется не только от угла к углу, но и вдоль его ребер. Такое закрашивание носит название способа Гуро и осуществляется в че-

тыре этапа:
-вычисление нормалей к поверхности;
-определение нормалей в вершинах путем усреднения нормалей по граням, которым принадлежит данная вершина;
-вычисление интенсивности в вершинах;
-закрашивание многоугольника путем линейной интерполяции значений интенсивности вдоль ребер и между ребрами.
Основной недостаток - эффект полосы Маха : на ребрах смежных многоугольников возникает полоса разрыва непрерывности.

Закрашивание по способу Фонга решает проблемы полосы Маха, поскольку предполагает плавное изменение яркости и насыщенности не только вдоль ребер каждого многоугольника, но и по самой поверхности (вдоль сканирующей строки интерполируется значение вектора нормали, который затем используется в модели освещения для вычисления интенсивности пиксела).

Наиболее реалистичные изображения в трехмерной машинной графики обеспечивает закрашивание способом трассировочных лучей.

Л И Т Е Р А Т У Р А

1. ГОСТ 7.1-84 Библиографическое описание документа. Общие требования и правила составления.-Взамен ГОСТ 7.1-76; Введ. 01.01.86.-М.:Изд-во стандартов,1984.-78с.
2.Роджерс Д. Алгоритмические основы машинной графики.-М.:Мир, 1989.-512 с.
3.Фоли Дж., вэн Дэм А. Основы интерактивной машинной графики. -М.:Мир,1985.-Т.1:375 с.,Т.2:368 с.
4. Хомяков К.С., Сурков Л.В., Петрова Г.Б. Методические указания по дипломному проектированию.-М.:МВТУ,1985.-28 с. _