ОГИ

Выполнить конспект лекции

ТРЕХМЕРНАЯ ГРАФИКА

Основные понятия трехмерной графики

Трехмерная графика нашла широкое применение в таких областях, как научные расчеты, инженерное проектирование, компьютерное моделирование физических объектов.

Для создания реалистичной модели объекта используются геометрические примитивы (куб, шар, конус и пр.) и гладкие, так называемые **сплайновые поверхности**. Вид поверхности определяется расположенной в пространстве сеткой опорных точек. Каждой точке присваивается коэффициент, величина которого определяет степень ее влияния на часть поверхности, проходящей вблизи точки. От взаимного расположения точек и величины коэффициентов зависит форма и гладкость поверхности в целом.

Деформация объекта обеспечивается перемещением контрольных точек, расположенных вблизи. Каждая контрольная точка связана с ближайшими опорными точками, степень ее влияния на них определяется удаленностью. Другой метод называют сеткой деформации. Вокруг объекта или его части размещается трехмерная сетка, перемещение любой точки которой вызывает упругую деформацию как самой сетки, так и окруженного объекта.

Еще одним способом построения объектов из примитивов служит твердотельное моделирование. Объекты представлены твердыми телами, которые при взаимодействии с другими телами различными способами (объединение, вычитание, слияние и др.) претерпевают необходимую трансформацию.

Все многообразие свойств в компьютерном моделировании сводится к визуализации поверхности, то есть к расчету коэффициента прозрачности поверхности и угла преломления лучей света на границе материала и окружающего пространства. Свойства поверхности описываются в создаваемых массивах текстур, в которых содержатся данные о степени прозрачности материала, коэффициенте преломления, цвете в каждой точке, цвете блика, его ширине и резкости и др.

После завершения конструирования и визуализации объекта приступают его "оживлению", т.е. заданию параметров движения. Компьютерная анимация базируется на ключевых кадрах.

Применение сложных математических моделей позволяет имитировать различные физические эффекты: взрывы, дождь, снег, огонь, дым, туман и др.

Основную долю рынка программных средств обработки трехмерной графики занимают три пакета: 3D Studio Max фирмы Kinetix; Softimage 3D компании Microsoft; Maya, разработанная консоциумом известных компаний (Alias, Wavefront, TDI). На сегодняшний день Maya является наиболее передовым пакетом в классе средств создания и обработки трехмерной графики для персональных компьютеров.

Трехмерная графика нашла широкое применение в таких областях, как научные расчеты, инженерное проектирование, компьютерное моделирование физических объектов. В качестве примера рассмотрим наиболее сложный вариант трехмерного моделирования – создание подвижного изображения реального физического тела.

В упрощенном виде для пространственного моделирования объекта требуется:

- спроектировать и создать виртуальный каркас (“скелет”) объекта, наиболее полно соответствующий его реальной форме;
- спроектировать и создать виртуальные материалы, по физическим свойствам визуализации похожие на реальные;
- присвоить материалы различным частям поверхности объекта (на профессиональном жаргоне – “спроектировать текстуры на объект”);
- настроить физические параметры пространства, в котором будет действовать объект, – задать освещение, гравитацию, свойства атмосферы, свойства взаимодействующих объектов и поверхностей;
- задать траектории движения объектов;

- рассчитать результирующую последовательность кадров;
- наложить поверхностные эффекты на итоговый анимационный ролик.

Для создания реалистичной модели объекта используют геометрические примитивы (прямоугольник, куб, шар, конус и прочие) и гладкие, так называемые *сплайновые поверхности.* В последнем случае применяют чаще всего метод *бикубических рациональных B-сплайнов на неравномерной сетке (NURBS).* Вид поверхности при этом определяется расположенной в пространстве сеткой опорных точек. Каждой точке присваивается коэффициент, величина которого определяет степень ее влияния на часть поверхности, проходящей вблизи точки. От взаимного расположения точек и величины коэффициентов зависит форма и "гладкость" поверхности в целом.

После формирования "скелета" объекта необходимо покрыть его поверхность материалами. Все многообразие свойств в компьютерном моделировании сводится к визуализации поверхности, то есть к расчету коэффициента прозрачности поверхности и угла преломления лучей света на границе материала и окружающего пространства.

Закраска поверхностей осуществляется методами Гуро *(Gouraud)* или Фонга *(Phong).* В первом случае цвет примитива рассчитывается лишь в его вершинах, а затем линейно интерполируется по поверхности. Во втором случае строится нормаль к объекту в целом, ее вектор интерполируется по поверхности составляющих примитивов и освещение рассчитывается для каждой точки.

Свет, уходящий с поверхности в конкретной точке в сторону наблюдателя, представляет собой сумму компонентов, умноженных на коэффициент, связанный с материалом и цветом поверхности в данной точке. К таковым компонентам относятся:

- свет, пришедший с обратной стороны поверхности, то есть преломленный свет (Refracted);
- свет, равномерно рассеиваемый поверхностью (Diffuse);
- зеркально отраженный свет (Reflected);
- блики, то есть отраженный свет источников (Specular);
- собственное свечение поверхности (Self Illumination).

Следующим этапом является наложение ("проектирование") текстур на определенные участки каркаса объекта. При этом необходимо учитывать их взаимное влияние на границах примитивов. Проектирование материалов на объект – задача трудно формализуемая, она сродни художественному процессу и требует от исполнителя хотя бы минимальных творческих способностей.

После завершения конструирования и визуализации объекта приступают к его "оживлению", то есть заданию параметров движения. Компьютерная анимация базируется на ключевых кадрах. В первом кадре объект выставляется в исходное положение. Через определенный промежуток (например, в восьмом кадре) задается новое положение объекта и так далее до конечного положения. Промежуточные значения вычисляет программа по специальному алгоритму. При этом происходит не просто линейная аппроксимация, а плавное изменение положения опорных точек объекта в соответствии с заданными условиями.

Эти условия определяются иерархией объектов (то есть законами их взаимодействия между собой), разрешенными плоскостями движения, предельными углами поворотов, величинами ускорений и скоростей. Такой подход называют методом *инверсной кинематики движения.* Он хорошо работает при моделировании механических устройств. В случае с имитацией живых объектов используют так называемые *скелетные модели.* То есть, создается некий каркас, подвижный в точках, характерных для моделируемого объекта. Движения точек просчитываются предыдущим методом. Затем на каркас накладывается оболочка, состоящая из смоделированных поверхностей, для которых каркас является набором контрольных точек, то есть создается *каркасная модель.* Каркасная модель визуализуется наложением поверхностных текстур с учетом условий освещения. В ходе перемещения объекта получается весьма правдоподобная имитация движений живых существ.

Наиболее совершенный метод анимации заключается в фиксации реальных движений физического объекта. Например, на человеке закрепляют в контрольных точках яркие источники света и снимают заданное движение на видео- или кинопленку. Затем координаты точек по кадрам переводят с пленки в компьютер и присваивают соответствующим опорным точкам каркасной модели. В результате движения имитируемого объекта практически неотличимы от живого прототипа.

Процесс расчета реалистичных изображений называют *рендерингом* (визуализацией).

Большинство современных программ рендеринга основаны на *методе обратной трассировки лучей (Backway Ray Tracing).* Применение сложных математических моделей позволяет имитировать такие физические эффекты, как взрывы, дождь, огонь, дым, туман. По завершении рендеринга компьютерную трехмерную анимацию используют либо как самостоятельный продукт, либо в качестве отдельных частей или кадров готового продукта.

Особую область трёхмерного моделирования в режиме реального времени составляют тренажеры технических средств – автомобилей, судов, летательных и космических аппаратов. В них необходимо очень точно реализовывать технические параметры объектов и свойства окружающей физической среды. В более простых вариантах, например при обучении вождению наземных транспортных средств, тренажеры реализуют на персональных компьютерах.

Самые совершенные на сегодняшний день устройства созданы для обучения пилотированию космических кораблей и военных летательных аппаратов. Моделированием и визуализацией объектов в таких тренажерах заняты несколько специализированных графических станций, построенных на мощных *RISC*-процессорах и скоростных видеоадаптерах с аппаратными ускорителями трехмерной графики. Общее управление системой и просчет сценариев взаимодействия возложены на суперкомпьютер, состоящий из десятков и сотен процессоров. Стоимость таких комплексов выражается девятизначными цифрами, но их применение окупается достаточно быстро, так как обучение на реальных аппаратах в десятки раз дороже.

ЭЛЕМЕНТЫ МАТЕМАТИЧЕСКОЙ ЛОГИКИ

1.Даны множества: A = {3,5,7}и B = {0,3,5,7,8}
Найдите пересечение множеств A и B. Найдите объединение множеств A и B.
2. Даны множества: A = {5,4,3}и B = {6,7,8,9,10}.
Найдите пересечение множеств A и B. Найдите объединение множеств A и B.
3. Изобразите с помощью кругов Эйлера множества, которые не пересекаются и объединение множеств.

ОПЕРАЦИОННЫЕ СИСТЕМЫ

Записать конспект лекции

7. Лекция: Тупики

Введение

Определение. Множество процессов находится в тупиковой ситуации, если каждый процесс из множества ожидает события, которое может вызвать только другой процесс данного множества. Так как все процессы чего-то ожидают, то ни один из них не сможет инициировать событие, которое разбудило бы другого члена множества и, следовательно, все процессы будут спать вместе.

Тупики также могут быть вызваны ошибками программирования. Например, процесс может напрасно ждать открытия семафора, потому что в некорректно написанном приложении эту операцию забыли предусмотреть. Другой причиной бесконечного ожидания может быть дискриминационная политика по отношению к некоторым процессам. Однако чаще всего событие, которого ждет процесс в тупиковой ситуации, – освобождение ресурса, поэтому в дальнейшем будут рассмотрены методы борьбы с тупиками ресурсного типа.

Ресурсами могут быть как устройства, так и данные. Некоторые ресурсы допускают разделение между процессами, то есть являются разделяемыми ресурсами. Например, память, процессор, диски коллективно используются процессами. Другие не допускают разделения, то есть являются выделенными, например лентопротяжное устройство. К взаимоблокировке может привести использование как выделенных, так и разделяемых ресурсов. Например, чтение с разделяемого диска может одновременно осуществляться несколькими процессами, тогда как запись предполагает исключительный доступ к данным на диске. Можно считать, что часть диска, куда происходит запись, выделена конкретному процессу. Поэтому в дальнейшем мы будем исходить из предположения, что тупики связаны с выделенными ресурсами , то есть тупики возникают, когда процессу предоставляется эксклюзивный доступ к устройствам, файлам и другим ресурсам.

Традиционная последовательность событий при работе с ресурсом состоит из запроса, использования и освобождения ресурса. Тип запроса зависит от природы ресурса и от ОС. Запрос может быть явным, например специальный вызов request, или неявным – open для открытия файла. Обычно, если ресурс занят и запрос отклонен, запрашивающий процесс переходит в состояние ожидания.

Условия возникновения тупиков

Условия возникновения тупиков были сформулированы Коффманом, Элфиком и Шошани в 1970 г.

1. Условие взаимоисключения (Mutual exclusion). Одновременно использовать ресурс может только

один процесс.

2. Условие ожидания ресурсов (Hold and wait). Процессы удерживают ресурсы, уже выделенные им,

и могут запрашивать другие ресурсы.

3. Условие неперераспределяемости (No preemtion). Ресурс, выделенный ранее, не может быть при-

нудительно забран у процесса. Освобождены они могут быть только процессом, который их

удерживает.

4. Условие кругового ожидания (Circular wait). Существует кольцевая цепь процессов, в которой ка-

ждый процесс ждет доступа к ресурсу, удерживаемому другим процессом цепи.

Для образования тупика необходимым и достаточным является выполнение всех четырех условий.

Обычно тупик моделируется циклом в графе, состоящем из узлов двух видов: прямоугольников – процессов и эллипсов – ресурсов, наподобие того, что изображен на рис. 7.1. Стрелки, направленные от ресурса к процессу, показывают, что ресурс выделен данному процессу. Стрелки, направленные от процесса к ресурсу, означают, что процесс запрашивает данный ресурс.

Основные направления борьбы с тупиками

Проблема тупиков инициировала много интересных исследований в области информатики. Очевидно, что условие циклического ожидания отличается от остальных. Первые три условия формируют правила, существующие в системе, тогда как четвертое условие описывает ситуацию,

которая может сложиться при определенной неблагоприятной последовательности событий. Поэтому методы предотвращения взаимоблокировок ориентированы главным образом на нарушение первых трех условий путем введения ряда ограничений на поведение процессов и способы распределения ресурсов. Методы обнаружения и устранения менее консервативны и сводятся к поиску и разрыву цикла ожидания ресурсов.

Итак, основные направления борьбы с тупиками:

• Игнорирование проблемы в целом

• Предотвращение тупиков

• Обнаружение тупиков

• Восстановление после тупиков

Игнорирование проблемы тупиков

Простейший подход – не замечать проблему тупиков. Для того чтобы принять такое решение, необходимо оценить вероятность возникновения взаимоблокировки и сравнить ее с вероятностью ущерба от других отказов аппаратного и программного обеспечения. Проектировщики обычно не желают жертвовать производительностью системы или удобством пользователей для внедрения сложных и дорогостоящих средств борьбы с тупиками.

Любая ОС, имеющая в ядре ряд массивов фиксированной размерности, потенциально страдает от тупиков, даже если они не обнаружены. Таблица открытых файлов, таблица процессов, фактически каждая таблица являются ограниченными ресурсами. Заполнение всех записей таблицы процессов может привести к тому, что очередной запрос на создание процесса может быть отклонен. При неблагоприятном стечении обстоятельств несколько процессов могут выдать такой запрос одновременно и оказаться в тупике. Следует ли отказываться от вызова CreateProcess, чтобы решить эту проблему?

Подход большинства популярных ОС (Unix, Windows и др.) состоит в том, чтобы игнорировать данную проблему в предположении, что маловероятный случайный тупик предпочтительнее, чем нелепые правила, заставляющие пользователей ограничивать число процессов, открытых файлов и т. п. Сталкиваясь с нежелательным выбором между строгостью и удобством, трудно найти решение, которое устраивало бы всех.

Способы предотвращения тупиков

Цель предотвращения тупиков – обеспечить условия, исключающие возможность возникновения тупиковых ситуаций. Большинство методов связано с предотвращением одного из условий возникновения взаимоблокировки.

Система, предоставляя ресурс в распоряжение процесса, должна принять решение, безопасно это или нет. Возникает вопрос: есть ли такой алгоритм, который помогает всегда избегать тупиков и делать правильный выбор. Ответ – да, мы можем избегать тупиков, но только если определенная информация известна заранее.

Суть алгоритма состоит в следующем.

• Предположим, что у системы в наличии n устройств, например лент.

• ОС принимает запрос от пользовательского процесса, если его максимальная потребность не превышает n.

• Пользователь гарантирует, что если ОС в состоянии удовлетворить его запрос, то все устройства

будут возвращены системе в течение конечного времени.

• Текущее состояние системы называется надежным, если ОС может обеспечить всем процессам

их выполнение в течение конечного времени.

Нарушение условия взаимоисключения

В общем случае избежать взаимоисключений невозможно. Доступ к некоторым ресурсам должен быть исключительным. Тем не менее некоторые устройства удается обобществить. В качестве примера рассмотрим принтер. Известно, что пытаться осуществлять вывод на принтер могут несколько процессов. Во избежание хаоса организуют промежуточное формирование всех выходных данных процесса на диске, то есть разделяемом устройстве. Лишь один системный процесс, называемый сервисом или демоном принтера, отвечающий за вывод документов на печать по мере освобождения принтера, реально с ним взаимодействует. Эта схема называется спулингом (spooling). Таким образом, принтер становится разделяемым устройством, и тупик для него устранен.

К сожалению, не для всех устройств и не для всех данных можно организовать спулинг. Неприятным побочным следствием такой модели может быть потенциальная тупиковая ситуация из-за конкуренции за дисковое пространство для буфера спулинга. Тем не менее в той или иной форме эта идея применяется часто.

Нарушение условия ожидания дополнительных ресурсов

Условия ожидания ресурсов можно избежать, потребовав выполнения стратегии двухфазного захвата.

• В первой фазе процесс должен запрашивать все необходимые ему ресурсы сразу. До тех пор пока

они не предоставлены, процесс не может продолжать выполнение.

• Если в первой фазе некоторые ресурсы, которые были нужны данному процессу, уже заняты другими процессами, он освобождает все ресурсы, которые были ему выделены, и пытается повторить первую фазу.

Таким образом, один из способов – заставить все процессы затребовать нужные им ресурсы перед выполнением ("все или ничего"). Если система в состоянии выделить процессу все необходимое, он может работать до завершения. Если хотя бы один из ресурсов занят, процесс будет ждать.

Данное решение применяется в пакетных мэйнфреймах (mainframe), которые требуют от пользователей перечислить все необходимые его программе ресурсы. Другим примером может служить механизм двухфазной локализации записей в СУБД. Однако в целом подобный подход не слишком привлекателен и приводит к неэффективному использованию компьютера. Как уже отмечалось, перечень будущих запросов к ресурсам редко удается спрогнозировать. Если такая информация есть, то можно воспользоваться алгоритмом банкира. Заметим также, что описываемый подход противоречит парадигме модульности в программировании, поскольку приложение должно знать о предполагаемых запросах к ресурсам во всех модулях.

Нарушение принципа отсутствия перераспределения

Если бы можно было отбирать ресурсы у удерживающих их процессов до завершения этих процессов, то удалось бы добиться невыполнения третьего условия возникновения тупиков. Перечислим минусы данного подхода.

Во-первых, отбирать у процессов можно только те ресурсы, состояние которых легко сохранить, а позже восстановить, например состояние процессора. Во-вторых, если процесс в течение некоторого времени использует определенные ресурсы, а затем освобождает эти ресурсы, он может потерять результаты работы, проделанной до настоящего момента. Наконец, следствием данной схемы может быть дискриминация отдельных процессов, у которых постоянно отбирают ресурсы.

Весь вопрос в цене подобного решения, которая может быть слишком высокой, если необходимость отбирать ресурсы возникает часто.

Трудно предложить разумную стратегию, чтобы избежать последнего условия из раздела "Условия возникновения тупиков" – циклического ожидания.

Один из способов – упорядочить ресурсы. Например, можно присвоить всем ресурсам уникальные номера и потребовать, чтобы процессы запрашивали ресурсы в порядке их возрастания. Тогда круговое ожидание возникнуть не может. После последнего запроса и освобождения всех ресурсов можно разрешить процессу опять осуществить первый запрос. Очевидно, что практически невозможно найти порядок, который удовлетворит всех.

Один из немногих примеров упорядочивания ресурсов – создание иерархии спин-блокировок в Windows 2000. Спин-блокировка – простейший способ синхронизации (вопросы синхронизации процессов рассмотрены в соответствующей лекции). Спин-блокировка может быть захвачена и освобождена процессом. Классическая тупиковая ситуация возникает, когда процесс P1 захватывает спин-блокировку S1 и претендует на спин-блокировку S2, а процесс P2, захватывает спин-блокировку S2 и хочет дополнительно захватить спин-блокировку S1. Чтобы этого избежать, все спин-блокировки помещаются в упорядоченный список. Захват может осуществляться только в порядке, указанном в списке.

Другой способ атаки условия кругового ожидания – действовать в соответствии с правилом, согласно которому каждый процесс может иметь только один ресурс в каждый момент времени. Если нужен второй ресурс – освободи первый. Очевидно, что для многих процессов это

неприемлемо.

Таким образом, технология предотвращения циклического ожидания, как правило, неэффективна и может без необходимости закрывать доступ к ресурсам.

Заключение

Возникновение тупиков является потенциальной проблемой любой операционной системы. Они возникают, когда имеется группа процессов, каждый из которых пытается получить исключительный доступ к некоторым ресурсам и претендует на ресурсы, принадлежащие другому процессу. В итоге все они оказываются в состоянии бесконечного ожидания.

С тупиками можно бороться, можно их обнаруживать, избегать и восстанавливать систему после тупиков. Однако цена подобных действий высока и соответствующие усилия должны предприниматься только в системах, где игнорирование тупиковых ситуаций приводит к катастрофическим последствиям.

Элементы высшей математики.

Самостоятельная работа № 8

1. Написать уравнение окружности, диаметром которой служит отрезок, отсекаемый на оси Ox параболой $y = 3 - 2x - x^2$.

2. Составить уравнение гиперболы, имеющей общие фокусы с эллипсом $\frac{x^2}{49} + \frac{y^2}{24} = 1$, при условии, что эксцентриситет ее равен 5/4.

3. Найти точки пересечения асимптот гиперболы $x^2 - 3y^2 = 12$ с окружностью,

4. имеющей центр в правом фокусе гиперболы и проходящей через начало координат

5. Найти длину и уравнение перпендикуляра, опущенного из фокуса параболы $y = -x^2/8$ на прямую, отсекающую на осях координат отрезки $a = 2, b = 2$.