

ПРОГРАММИСТУ

Д. Конгер

ФИЗИКА

ДЛЯ РАЗРАБОТЧИКОВ
КОМПЬЮТЕРНЫХ ИГР



БИНОМ

ФИЗИКА

**ДЛЯ РАЗРАБОТЧИКОВ
КОМПЬЮТЕРНЫХ ИГР**

PHYSICS MODELING FOR GAME PROGRAMMERS

David Conger



THOMSON



™

COURSE TECHNOLOGY

Professional ■ Trade ■ Reference

Д. Конгер

ФИЗИКА

ДЛЯ РАЗРАБОТЧИКОВ КОМПЬЮТЕРНЫХ ИГР

Перевод с английского
А. С. Молякко



Москва
БИНОМ. Лаборатория знаний
2007

УДК 004.7
ББК 32.973.202
К64

Серия основана в 2005 г.

Конгер Д.

К64 Физика для разработчиков компьютерных игр /
Д. Конгер ; пер. с англ. — М. : БИНОМ. Лаборатория
знаний, 2007. — 520 с. : ил. — (Программисту).

ISBN 978-5-94774-317-3 (русск.)

ISBN 1-59200-093-2 (англ.)

Книга посвящена физическим и математическим основам построения компьютерных игр. Рассматриваются законы существования реального и виртуального миров, следование которым позволяет разработчику игры создавать реалистичную графику. Подробно рассматриваются приемы моделирования точечных частиц, твердых тел и жидкостей. Изложение богато иллюстрировано примерами моделирования на C++ для DirectX.

Для начинающих разработчиков игр, обладающих знаниями школьного курса физики и математики.

УДК 004.7
ББК 32.973.202

По вопросам приобретения обращаться:
«БИНОМ. Лаборатория знаний»
Телефон: (499) 157-5272
e-mail: binom@Lbz.ru, <http://www.Lbz.ru>

© 2004 by Thomson Course Technology PTR. All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system without written permission from Thomson Course Technology PTR, except for the inclusion of brief quotations in a review.

© Перевод на русский язык, оформление, БИНОМ. Лаборатория знаний, 2007

ISBN 978-5-94774-317-3 (русск.)
ISBN 1-59200-093-2 (англ.)

Оглавление

Введение	8
Часть I. Физика, математика и программирование игр	11
Глава 1. Физика в играх	12
Что я должен знать из физики, чтобы писать игры?	12
3D-объекты	13
3D-сцены	13
Движение	13
Твердые объекты	14
Вращение	14
Трение	14
Сопротивление воздуха и воды	15
Сила тяжести	15
Столкновения и взрывы	16
Гибкие вещи	16
Волны	16
Что я должен знать из математики, чтобы писать игры?	17
Основы геометрии треугольников	17
Векторы	17
Матрицы	17
Производные	18
Что я должен знать из программирования?	18
Глава 2. Имитация 3D-графики с помощью DirectX	19
Что такое DirectX?	19
Два представления DirectX	20
Низкоуровневое представление: HAL и HEL	20
Высокоуровневое представление: компоненты DirectX	21
COM-объекты	22
Использование DirectX	24
Инициализация DirectX лобовым способом	24
Инициализация DirectX с помощью мастера DirectX AppWizard	37
Инициализация DirectX с помощью платформы физического моделирования	39
Глава 3. Математические инструменты	51
Геометрия треугольников	51
Двумерные системы координат	53
Трехмерные и четырехмерные системы координат	54
Единицы измерения	56
Векторы	59
Реализация векторов в коде программ	62

Умножение и деление вектора на скаляр	68
Скалярное произведение векторов	70
Векторное произведение векторов	73
Единичные векторы	74
Проецирование	76
Векторы в Direct3D	80
Матрицы	81
Единичная матрица	83
Сложение и вычитание матриц	83
Умножение и деление матрицы на скаляр	84
Перемножение матриц	85
Транспонирование матриц	89
Определители	90
Обращение матриц	91
Глава 4. 2D-преобразования и рендеринг	96
2D-преобразования	96
Активные и пассивные трансформации	98
Перемещение	99
Поворот	100
Масштабирование	104
Сочетание преобразований	107
Применение преобразований – вращающийся треугольник	109
Применение платформы физического моделирования	109
Настройка геометрии	111
Обновление кадров	115
Рендеринг кадров	117
Запуск программы	118
Глава 5. 3D-преобразования и рендеринг	119
3D-преобразования	119
Однородные координаты	119
Перемещение	120
Масштабирование	121
Вращение	122
3D-конвейер	125
Локальные и глобальные координаты	126
Глобальные координаты и координаты отображения	127
Координаты отображения и координаты проецирования	128
Координаты проецирования и экранные координаты	129
Рендеринг в 3D	129
Пример 1: Вращающийся треугольник в 3D	129
Пример 2: Вращающаяся пирамида	133
Глава 6. Сетчатые модели и X-файлы	138
Текстуры	139
Создание текстур из файлов	140
Задание текстур	141
Материалы	142
Загрузка сетчатой модели	142
Извлечение текстур и материалов	143
Рендеринг сетчатой модели	145

Очистка сетчатой модели	145
Класс d3d_mesh	146
Загрузка сетчатой модели в классе d3d_mesh	147
Рендеринг сетчатой модели в классе d3d_mesh	150
Подсчет ссылок в классе d3d_mesh	151
Часть II. 3D-объекты, движение и столкновения	159
Глава 7. Динамика материальных точек	160
Материальные точки	160
Одномерная кинематика	162
Скорость	162
Скорость как производная	163
Ускорение	166
Силы	167
Двумерная и трехмерная кинематика	169
Моделирование материальных точек	172
Класс d3d_point_mass	172
Применение класса d3d_point_mass	177
Материальные точки в играх	186
Глава 8. Столкновения материальных точек	187
Обнаружение столкновений	187
Ограничивающие сферы	188
Ограничивающие цилиндры	192
Ограничивающие блоки	193
Оптимизация с помощью пространственного разделения	195
Реакция на столкновения	197
Закон сохранения импульса	198
Энергия	200
Упругие столкновения	201
Неупругие столкновения	202
Коэффициент восстановления	203
Столкновения материальных точек в двумерных и трехмерных системах координат	205
Столкновения сфер	205
Реализация	207
Глава 9. Динамика твердых тел	217
Твердые тела	217
Центр масс	218
Вращение двумерных твердых тел	220
Материальные точки в двумерном твердом теле	221
Вращающий момент и момент инерции	224
Твердые тела в 3D	229
Вращающий момент в 3D	232
Теорема Гюйгенса в 3D	234
Выбор осей	235
Ориентация	235
Реализация твердых тел в 3D	238

Класс <code>d3d_rigid_body</code>	238
Инициализация объекта класса <code>d3d_rigid_body</code>	240
Обновление объектов класса <code>d3d_rigid_body</code>	243
Рендеринг объекта класса <code>d3d_rigid_body</code>	247
Глава 10. Столкновения твердых тел	249
Обнаружение столкновений	249
Грубые приближения	249
Улучшенные методы обнаружения столкновений	251
Реакция на столкновения	255
Линейная реакция на столкновения	255
Угловая реакция на столкновение	256
Совмещение линейной и угловой реакции на столкновение	257
Обновление платформы физического моделирования	260
Глава 11. Сила тяжести и метательные снаряды	282
Закон всемирного тяготения Ньютона	282
Траектории метательных снарядов	284
Моделирование движения метательных снарядов	287
Разделение импульсных и постоянно действующих сил	288
Качение	308
Глава 12. Системы масс и пружин	310
Что можно делать с помощью пружин?	310
Волосы и прически	310
Ткань	312
Основа: гармонические колебания	313
Закон Гука	315
Затухающие гармонические колебания	315
Реализация ткани	316
Усовершенствование материальных точек	316
Пружины	322
Класс <code>cloth</code>	327
Инициализация фрагмента ткани	337
Обновление и рендеринг фрагмента ткани	338
Подстройка ткани	343
Возможные улучшения для моделирования ткани	343
Глава 13. Вода и волны	345
Вода и плавучесть	345
Свойства воды	346
Почему объекты плавают	347
Давление и плотность	349
Сопrotивление движению	351
Краткий обзор трения	351
Вязкость	354
Течения в воде	355
Волны	357
Вода в программах	360
Простые способы имитации	360
Трехмерная вода	361

Объекты в воде	363
Добавление плавучести в класс твердых тел	363
Пример программы	369
Часть III. Практические примеры	375
Глава 14. Готовимся создавать игры	376
Переработка платформы физического моделирования	376
Упрощение инициализации программы	377
Добавление класса <code>game</code>	380
Эффективное задание матриц преобразований	384
Восстановление потерянных объектов устройств	388
Переопределение твердых тел с помощью материальных точек	389
Центр масс и точка начала координат сетчатой модели	395
Введение в <code>DirectInput</code>	398
Инициализация <code>DirectInput</code>	399
Получение данных от клавиатуры и мыши	406
Завершение работы с <code>DirectInput</code>	410
Перемещение камеры в <code>DirectX</code>	410
Глава 15. Автомобили, корабли и лодки	415
Автомобили	415
Мощность, сила, ускорение и трение	415
Сопrotивление воздуха движущимся автомобилям	418
Торможение	419
Повороты автомобилей	420
Реализация простой модели автомобиля	423
Транспорт на воздушной подушке и антигравитационные транспортные средства	429
Как работает воздушная подушка	429
Сопrotивление воздуха транспортным средствам на воздушной подушке	430
Повороты транспортных средств на воздушной подушке	431
Корабли и лодки	431
Плавучесть кораблей и лодок	431
Вычисление объема корпуса	432
Остойчивость кораблей и лодок	433
Масса и виртуальная масса	435
Корабли и сопротивление движению	437
Сопrotивление воздуха	438
Течения и волны	438
Глава 16. Авиация и космические корабли	440
Простой подход к авиасимуляторам	440
Полет без моделирования физики	442
Использование класса <code>basic_flyer</code>	446
Физика самолетов	451
Основные части самолета	451
Основные силы	454
Моделирование летательных аппаратов: правильное приложение сил	457
Физика космических кораблей	457
Перестрелки в космосе	457

Ракеты	458
Высадка на Луну	463
Путешествия на другие планеты с помощью известных методов	464
Межпланетные путешествия согласно предполагаемым возможностям	468
Эпилог	470
Часть IV. Приложения	471
Приложение А. Глоссарий	472
Приложение В. Краткий обзор языка C++	475
Все начинается с функций	475
Функция main() и функции, вызываемые из нее	475
Параметры	476
Возвращаемые значения	476
Встраиваемые функции	477
Классы и объектно-ориентированное программирование	477
Пространства имен	480
Наследование	481
Переопределение функций	483
Виртуальные методы	484
Исключения	485
Другие способы создания новых типов	487
Структуры	487
Перечисляемые типы	487
Операторы typedef	488
Приложение С. Основы программирования для Windows	489
Добро пожаловать в WinMain()	489
Написание функции WinMain()	490
Определение класса окна	491
Регистрация класса окна	493
Создание окна	493
Отображение окна	495
Обработка сообщений Windows	495
Процедура обработки сообщений	496
Предметный указатель	498
Оглавление	515

Введение

Добро пожаловать в *мир физического моделирования!*

Моделирование физических законов реального мира все шире используется в играх. Оно позволяет создавать великолепно выглядящие игры, и оно является практически единственным инструментом, позволяющим создавать игры реалистичные. Компании, занимающиеся созданием компьютерных игр, постоянно ищут программистов, разбирающихся в физике. Хорошее знание физики может превратить человека с незначительными познаниями в программировании в ценного специалиста.

Кроме того, моделирование физики – само по себе интересная задача. Простая физическая модель позволяет создавать эффекты, которых практически невозможно добиться без моделирования физики. Например, хорошая модель пламени будет прекрасно выглядеть, независимо от того, изображает ли она огонь в камине или горящий двигатель падающего самолета.

Современные компьютерные игры в большинстве своем изображают целые виртуальные миры. Эти виртуальные миры могут выглядеть и функционировать так, как считают нужным их создатели. Однако если мы – создатели – хотим, чтобы игра была понятна игрокам и привлекала их, мы должны создавать миры, более или менее соответствующие реальности. А поведение и внешний вид реального мира – это и есть предмет физики.

Однако не только понятность делает реальный мир хорошим примером для наследования. Реальный мир – удивительное место, и ни один выдуманный, виртуальный мир не может быть таким же замысловатым, богатым и прекрасным, как вселенная вокруг нас.

Книга

Книга разделена на три части.

Часть первая. Физика, математика и программирование игр

В первой части рассматривается математический аппарат, который потребуется нам для моделирования физики. В частности, рассматривается евклидова геометрия. Эта геометрия понадобится нам для работы с DirectX, которую мы тоже рассмотрим в первой части книги. Собственно говоря, графические возможности DirectX – просто удобный инструмент для изображения евклидовой геометрии в Windows.

Часть вторая: Трехмерные объекты, движение и столкновения

Во второй части вы познакомитесь с динамикой материальных точек и твердых тел. Проще всего воспринимать динамику как науку о движении объектов. В этой части книги излагаются принципы и законы динамики, с помощью которых можно добиться реалистичного движения практически любых объектов в играх. Кроме того, мы разберемся, как изображать столкновения объектов – похоже, именно возможность врезаться во что-то пользуется особым спросом в играх.

Физика, используемая в компьютерных играх, не ограничивается динамикой. В этой части вы также узнаете о силе тяжести, пружинах и жидкостях. Эти элементы часто используются в играх, чтобы добиться реалистичности, и чем мощнее становятся наши компьютеры, тем большей реалистичности можно добиться с их помощью.

Часть третья: Практические трехмерные имитации

Невозможно смоделировать что бы то ни было абсолютно точно. Вычислительная мощь компьютеров небесноконечна, и это особенно заметно в играх, где любые вычисления нужно повторять не меньше 30 раз в секунду. Даже если вам хватает быстродействия компьютера, вы рано или поздно столкнетесь с чем-то, что невозможно точно смоделировать. И тогда приходится использовать приближения. Именно в этом и заключается суть имитаций. Третья часть посвящена самым распространенным типам имитаций в играх.

Компакт-диск

Компакт-диск, прилагаемый к книге, содержит множество полезных вещей. Прежде всего, на нем есть весь исходный код примеров к книге – вам не придется разбивать клавиатуру, набирая его. Весь исходный код находится в папке **Source**.

В папке **Tools** вы найдете полезные для создания игр инструменты. Во-первых, в этой папке есть подпапка **Microsoft DirectX SDK**. В ней вы найдете копию набора инструментов, который Microsoft предлагает программистам для создания игр под DirectX. Если вы хотите использовать примеры кода из книги, вам понадобится установить этот набор.

Кроме того, в папке **Tools** есть папка с замечательной маленькой программой **MilkShape3D**. Эта программа позволяет легко и быстро создавать трехмерные сетчатые модели. На компакт-диске находится пробная версия этой программы. Полнофункциональную версию стоимостью

20 долларов можно скачать с сайта разработчика – **chUmbaLum sOft** – по адресу <http://www.swissquake.ch/chumbalum-soft/>.

Далее, в папке **Tools** есть папка **Torque Game Engine** – в этой папке находится демонстрационная версия полнофункционального игрового движка Torque. Этот движок создан компанией Garage Games. Веб-сайт этой компании можно найти по адресу www.garagegames.com.

Если вы не можете себе позволить купить коммерческий игровой движок вроде Torque, попробуйте бесплатно распространяемый движок CrystalSpace 3D. Этот движок представляет собой проект с открытым исходным кодом. Он есть на компакт-диске в папке **Tools\CrystalSpace3D**. Одно из самых удобных свойств этого движка – то, что вы можете изменять его исходный код так, как вам заблагорассудится. Проект CrystalSpace 3D в Интернете доступен по адресу: <https://sourceforge.net/projects/crystal>.

Что вам понадобится

Чтобы воспользоваться большей частью изложенного в этой книге, вам понадобится компьютер с Windows 98 или более новой операционной системой и среда разработки программ на C++. Книга предполагает, что у вас есть доступ к Visual C++ 6.0 или более новой версии, но можно воспользоваться и другой средой. Кроме того, вам понадобится видеокарта, поддерживающая выходное разрешение 640 × 480 с 32-битным цветом.

Я полагаю, что вы немного знаете C или C++. В книге используются базовые возможности C++. Многие концепции из физики и игр превосходно реализуются в виде объектов – так зачем же гробить себя, вбивая их в прокрустово ложе структур C? Если вы знаете только C или ваши навыки программирования на C++ немного заржавели, обратитесь к приложению В, «Краткий обзор языка C++».

Вам не потребуются особые знания о программировании для Windows или DirectX. В первой части книги содержится достаточно информации, чтобы вы смогли приступить к написанию программ, использующих DirectX. Возможно, это введение в программирование покажется вам скуповатым, но изучите его, и вы сможете писать программы для Windows. Чтобы немного облегчить вашу задачу, в книге есть приложение С, «Основы программирования для Windows».

Книга в основном посвящена физике и программированию игр, поэтому в ней немало математики. Многих людей математика отпугивает. Однако попробуйте почитать эту книгу, и вы увидите, что она не слишком сложна. На случай, если ваши школьные учителя математики до сих пор являются вам в кошмарных снах, я скажу вам, что очень часто математику представляют более сложной, чем она есть на самом деле. Все, что вам потребуется, чтобы понять эту книгу – быть готовым воспринять несколько новых идей. Вот, пожалуй, и все – можно приступать к первой главе, «Физика в играх».

Глава 1

Физика в играх

Что делает игры привлекательными?

Люди, пишущие игры, тратят немало времени на поиски ответа на этот вопрос. Ответы, которые они дают, в основном зависят от аспектов игр, с которыми они работают. Писатели, маркетологи, производители и дизайнеры уровней ответят на этот вопрос по-разному. Однако подавляющее большинство игр связано с имитацией движения. Когда вы играете в игру, двигаются персонажи, объекты и сцены. Если вы хотите, чтобы игрок мог погрузиться в игровой мир, все должно двигаться реалистично. Это фундаментальный принцип игр. Чтобы добиться реалистичного движения персонажей и объектов на экране, нужно моделировать или имитировать физические законы реального мира. Современные игры обычно трехмерные.

Так как же моделировать физический мир в 3D?

Ответить на этот вопрос и призвана данная книга. Она позволит познакомиться с основами физики, математики и программирования трехмерных игр.

Реакцией многих людей на предыдущее предложение будет паника. Люди часто приходят в ужас при мысли о физике и математике. Да, безусловно, физика и математика могут быть сложными. Однако есть важное правило, которое должны всегда помнить программисты, моделируя физику в играх: если все выглядит нормально, значит, все нормально.

Это утверждение сильно облегчает программистам жизнь. Нам не нужно моделировать все аспекты физики, чтобы игры были реалистичными. Достаточно, если все выглядит правильным на экране. Такой подход освобождает нас, программистов, от необходимости работать с действительно замысловатыми областями физики и математики. Если мы знаем основы физики и обладаем некоторыми математическими навыками, этого почти наверняка хватит для программирования игр.

Что я должен знать из физики, чтобы писать игры?

Вспомните игры, в которые вы играли. Что происходит в этих играх? Ваш персонаж бежит и стреляет? Он лазит по лестницам, плавает, прыгает и так далее? Бросает ли он какие-то предметы? А взрывы? Похоже, что в наши дни игры просто не могут жить без взрывов.

Встраивание физики в игры означает моделирование нескольких основных вещей:

- 3D-объектов;
- 3D-сцен;
- движения;
- твердых объектов;
- вращения;
- трения;
- сопротивления воздуха и воды;
- силы тяжести;
- столкновений и взрывов;
- гибких вещей;
- волн.

3D-объекты

Создать программную модель 3D-объекта непросто. Собственно говоря, чтобы понять, как это делать, потребовались десятилетия работы множества умных людей. Однако теперь, благодаря этим первопроходцам, мы знаем, как это делается. Мы можем использовать для имитации 3D-объектов широко доступные инструменты. Это очень сильно облегчает нам жизнь.

Например, в большинстве современных компьютеров есть видеокарты, обладающие мощными возможностями по обработке 3D-графики. Это одновременно упрощает написание программ и ускоряет их работу.

Графические библиотеки, например, DirectX и Open Graphics Library (OpenGL), дополняют аппаратную поддержку 3D-графики. Они добавляют еще один слой, выполняющий за нас часть работы. В главе 2 «Имитация 3D-графики с помощью DirectX» вы кратко познакомитесь с DirectX, поэтому, если вы не знаете, что это такое, не волнуйтесь. Вы увидите, что заставить DirectX работать несложно.

3D-сцены

Моделирование целых сцен в 3D – это продолжение темы моделирования 3D-объектов. Вы начнете моделировать 3D-сцены в главе 8 «Столкновения материальных точек».

Движение

В играх много движений. Части тел персонажей двигаются, когда персонажи ходят, прыгают, бегают или подбирают предметы. В сценах двигаются и персонажи, и предметы. Как сделать так, чтобы их движение выглядело реалистичным – эта тема затрагивается почти в каждой главе книги.

Твердые объекты

Представьте себе, что вы пишете игру, в которой персонаж передвигается по внешней поверхности вращающейся космической станции. По мере перемещения персонажа от центра станции к ее краю силы, действующие на персонаж, растут. Чем ближе он к краю, тем больше у него вероятность сорваться и улететь в космос. Если он сорвется, игра окончена.

Вращающаяся космическая станция – это пример движущегося твердого объекта. Твердые объекты кажутся обманчиво простыми. В действительности нужно сделать намного больше, чем кажется на первый взгляд. В главе 9 «Динамика твердых тел» и главе 10 «Столкновения твердых тел» вы познакомитесь с основами физики твердых тел.

Вращение

3D-объекты могут двигаться вперед или назад, влево или вправо, вверх или вниз. Однако, двигаясь, они могут еще и вращаться. Моделирование вращения увеличивает количество сил, которые игра должна прикладывать к объекту. Вращение может стабилизировать или дестабилизировать движущиеся объекты.

Например, когда игрок в футбол (я говорю об американском футболе) бросает мяч, он произвольно бросает его так, чтобы мяч вращался. Вращение стабилизирует полет мяча, и его легче поймать. Если вы пишете игру, в которой моделируется игра в футбол, моделирование вращения будет важным моментом.

Трение

В реальном мире большинство движущихся объектов, в конце концов, останавливается из-за трения. Моделирование трения часто бывает необходимым и в играх. Я играл в игры, где персонажу приходится двигаться по обледенелым или просто скользким поверхностям. Игрок должен добиться, чтобы персонаж двигался в нужном направлении по этим поверхностям, а чтобы сделать жизнь интереснее, по персонажу обычно стреляют со всех сторон.

Мне часто приходилось сталкиваться со случаями, когда программисты теряли массу времени, пытаясь смоделировать трение. Они пытались свести все к правилу: «Если все выглядит нормально, значит, все нормально». Они писали программу и проверяли, правильным ли выглядело движение персонажа по скользкой поверхности. Если нет, они изменяли программу и снова проверяли. Поскольку они не опирались на реальную физику, им приходилось множество раз переписывать программу и проверять ее в работе. Они бы сэкономили массу времени, если бы просто использовали в программах формулы из физики.

Сопротивление воздуха и воды

Во многих играх сопротивление воздуха игнорируется вообще, однако никто этого не замечает. В прошлом игры выглядели правдоподобными и без моделирования сопротивления воздуха. Однако, похоже, это скоро отойдет в прошлое. Игры становятся все реалистичнее, и важность моделирования сопротивления воздуха растет.

Игнорировать сопротивление воды разработчики игр не могут. В любой игре, где персонажи и объекты двигаются в воде, возникает необходимость реалистично моделировать сопротивление воды.

Моделировать сопротивление воды значит не только замедлять движение в воде. Ведь вода может двигаться сама по себе. Возникающие при этом течения увеличивают сопротивление, если персонажи или объекты двигаются против этих течений. Течения увлекают за собой все в них падающее.

В некоторых играх движения в воде моделируются достаточно эффективно. Пример – серия игр Legend of Zelda. Главный персонаж часто движется в воде. При этом способ передвижения персонажа зависит от того, какими средствами на данный момент он располагает. Если у него есть волшебная маска, превращающая его в водное существо, он движется быстро. В противном случае его движения будут довольно медленными. Течения увеличивают или уменьшают сопротивление, когда персонаж плавает.

Если в вашей игре встречается движение в воде, то нужно смоделировать сопротивление воды хотя бы на том же уровне, что и в этой серии игр.

Сила тяжести

Сила тяжести влияет на все. От нее нельзя избавиться, даже в космосе. Не важно – бросает ли ваш персонаж гранату или ведет космический корабль к Марсу, на результат его действий влияет сила тяжести. Игра должна моделировать силу тяжести во всех ситуациях, поэтому моделированию силы тяжести я посвятил целую главу: это глава 11, «Сила тяжести и метательные снаряды». В ней рассказано достаточно, чтобы можно было смоделировать силу тяжести почти во всех ситуациях.

Замечание

Практически единственная ситуация, для которой я не описываю моделирование силы тяжести – это сила тяжести внутри черной дыры. Физические законы, которые нужны для моделирования этого случая, слишком сложны для этой книги. Если ваш персонаж в игре должен попадать в черные дыры, можете без угрызений совести обманывать игрока и программировать такое поведение, какое сочтете нужным. Игроку никогда не доводилось бывать в черной дыре, и он, скорее всего, поверит в то, что вы ему покажете.

Столкновения и взрывы

Что это за игра без взрывов? Даже в миролюбивых играх вроде The Sims есть взрывы. Я не знаю, почему это так, но большинству игроков нравится видеть, как предметы врезаются друг в друга и взрываются. Именно поэтому мы так часто видим сталкивающиеся автомобили в фильмах. Похоже, что Голливуд потребляет солидную часть продукции автопромышленности.

Невозможно смоделировать все аспекты столкновений и взрывов. Физические соотношения, работающие здесь, слишком сложны. К счастью, это не так уж важно. Если мы сможем смоделировать основные силы и взаимодействия объектов в столкновениях и взрывах, все будет выглядеть нормально. А если все выглядит нормально, значит, все нормально.

Гибкие вещи

Хотя обычно мы этого не замечаем, вокруг нас множество гибких вещей. Если я говорю «гибкие вещи», вы, вероятно, представляете себе палки для прыжков и тому подобное. В физике «гибкими вещами» будут, например, волосы и одежда. Представляли ли вы себе, как сложно смоделировать движение прически идущей девушки? Смоделировать движение платья ничуть не проще.

Долгое время моделирование одежды, волос и других гибких вещей было слишком сложным, чтобы они могли присутствовать в играх. Более того, это моделирование было настолько сложным, что его избегали в компьютерной графике и анимации. 3D-моделирование тех времен было достаточно хорошим, чтобы моделировать все, кроме одежды и волос. В результате появилось множество 3D-мультфильмов о насекомых. В этих мультфильмах нет ни одежды, ни волос.

Однако недавние достижения позволяют моделировать в играх гибкие вещи, включая одежду и волосы. Их движение можно сделать достаточно реалистичным.

Волны

Работая с водой, приходится иметь дело не только с сопротивлением и течениями: на воде должны быть волны. В старых играх волны имитировались медленным перемещением персонажа или камеры вверх и вниз. Но в современных 3D-играх такое не проходит. Нужен более реалистичный подход.

Например, предположим, что вы пишете игру о гонках на моторных лодках, вроде Hydro Thunder (аркадная игра). Если волна ударит лодку в лоб, лодку может подбросить вверх или даже перевернуть. Если волна ударит лодку в борт, лодка вполне может перевернуться. Результат ударов зависит от размеров волн, угла столкновения лодки с ними, веса и формы лодки и так далее. Все эти факторы нужно правильно смоделировать в игре.

Что я должен знать из математики, чтобы писать игры?

Физика требует математики. Если вы не математический гений, не волнуйтесь, я расскажу вам обо всех математических понятиях, которые вам потребуются, чтобы разобраться в этой книге. Вы познакомитесь с:

- основами геометрии треугольников;
- векторами;
- матрицами;
- производными.

Основы геометрии треугольников

Компьютерная 3D-графика основана на треугольниках. Если вы собираетесь моделировать 3D-сцены и объекты, вы должны знать основные свойства треугольников. Например, нужно уметь найти длину стороны треугольника, зная длину двух других сторон и величину угла между ними.

Векторы

Физика занимается вопросами взаимодействия сил и объектов. Силы очень удобно представлять с помощью векторов. Векторы дают удобный способ анализа сочетаний сил и определения сил, действующих на объекты.

Матрицы

Программисты, работающие с 3D-объектами, обычно преобразуют векторы сил в матрицы. Матрицы предоставляют изящный способ упрощенного представления проблем. Они делают многие задачи 3D-графики более простыми для понимания и выполнения.

Например, предположим, что нужно смоделировать поведение ящика, которое должно зависеть от того, куда приложено усилие – к ребру или к середине грани. Если усилие приложено к верхнему ребру, он должен перевернуться. Если оно приложено к середине боковой грани, он должен скользить по полу.

Чтобы правильно смоделировать поведение ящика, нужно начать с анализа сил с помощью векторов. Затем нужно преобразовать векторы в матрицы и воспользоваться правилами умножения матриц, чтобы определить величины сил, действующие на вершины ящика. В результате можно определить, как будет двигаться ящик.

Описанная только что методика применяется при решении многих задач. Умение обращаться с матрицами крайне важно для программиста, пишущего игры.

Производные

Производные – это часть математического анализа. Да, математический анализ – не самая простая область математики, и он сложен для понимания. Однако сам процесс использования производных можно упростить. Если вы не изучали математический анализ, я думаю, вы удивитесь тому, насколько простыми могут быть производные.

Что я должен знать из программирования?

Краткий ответ на этот вопрос: «Не слишком много». Если вы можете написать программу на C++ для Windows, то знаете достаточно, чтобы освоить эту книгу. Если вы изучали программирование на C++ в школе или институте, то поймете все, что мы будем рассматривать в этой книге.

Если вы изучали программирование самостоятельно и занимались написанием программ на C++ для Windows около года или больше, все будет в порядке.

Для программирования графики мы будем использовать библиотеки Microsoft DirectX. О DirectX написаны целые книги. Вам не обязательно иметь опыт работы с ним. В этой книге о DirectX будет рассказано достаточно, чтобы вы смогли выполнять физическое 3D-моделирование, которому посвящена данная книга. Если вы хотите глубже изучить DirectX, попробуйте почитать, например, книгу Wendy Jones «*Beginning DirectX 9*» (издательство *Premier Press*).

Если вы приверженец OpenGL или какой-то другой графической библиотеки, не пугайтесь. Хотя в примерах этой книги используется DirectX, собственно физическое моделирование выполняется в коде, который можно использовать практически с любой графической библиотекой. Можно использовать этот код с OpenGL или чем-то еще, не внося в него больших изменений.

Итоги

Чтобы создать реалистичную 3D-игру, программисты должны моделировать физические силы, действующие в природе. Это требует знания физики, математики и программирования 3D-графики. Обо всем этом и рассказывается в этой книге. Прежде всего, мы изучим основы 3D-программирования с помощью DirectX. Этому посвящена следующая глава.

[. . .]

Только в совершенстве овладев мастерством моделирования физических законов можно добиться исключительно высокого уровня реалистичности компьютерных игр. В этой книге помимо теоретических основ и практических методов вы найдете полезные рекомендации по приложению обретенного мастерства к реальным задачам, познакомитесь с математическими основами физического моделирования и узнаете, как получить реалистичную картину движения и столкновения объектов в играх. Завершают книгу примеры моделирования движения различных транспортных средств, в том числе автомобилей и летательных аппаратов.

Рассматриваются:

3D-моделирование с помощью DirectX®
Математический инструментарий физики и 3D-программирования
2D- и 3D-преобразования и рендеринг
3D-объекты, движения и столкновения
Сила тяжести и метательные снаряды
Автомобили, транспорт на воздушной подушке, корабли и лодки

На компакт-диске

Microsoft® DirectX 9 SDK
MilkShape 3D
CrystalSpace™ 3D
Инсталляция Torque Engine
Исходный код примеров из книги

Об авторе

Дэвид Конгер – программист с 20-летним стажем. Он участвовал в написании игр для DOS, многопользовательских Интернет-игр, встроенных программ для графических контроллеров в военной авиации и коммерческих приложений. В течение четырех лет читал курсы по computer science и программированию в колледже. Д. Конгер – автор документации Microsoft, ряда книг по C, C++, C# и программированию для платформы .NET, а также учебника по микрокомпьютерам.