

# Оглавление

Предисловие .....	26
От издательства .....	27
Новое в третьем издании.....	28
Благодарности.....	30

## Часть I. Основы

<b>Глава 1.</b> Введение.....	34
1.1. Структура типичной игровой команды.....	35
1.1.1. Разработчики .....	36
1.1.2. Специалисты творческих профессий .....	36
1.1.3. Геймдизайнеры .....	37
1.1.4. Продюсеры.....	38
1.1.5. Другой персонал .....	38
1.1.6. Издатели и студии.....	38
1.2. Что такое игра.....	39
1.2.1. Видеоигры как мягкая симуляция реального времени.....	39
1.3. Что такое игровой движок .....	41
1.4. Различия движков для разных жанров .....	43
1.4.1. Шутеры от первого лица.....	44
1.4.2. Платформеры и другие игры от третьего лица .....	46
1.4.3. Файтинги.....	47

1.4.4.	Гонки .....	49
1.4.5.	Игры-стратегии .....	51
1.4.6.	Многопользовательские онлайн-игры.....	52
1.4.7.	Контент, созданный игроком.....	54
1.4.8.	Виртуальная, дополненная и смешанная реальность .....	56
1.4.9.	Другие жанры .....	60
1.5.	Обзор игровых движков .....	60
1.5.1.	Семейство движков Quake .....	60
1.5.2.	Движок Unreal .....	61
1.5.3.	Source — движок Half-Life .....	62
1.5.4.	DICE Frostbite .....	62
1.5.5.	RAGE.....	62
1.5.6.	CRYENGINE.....	63
1.5.7.	Sony PhyreEngine.....	63
1.5.8.	Microsoft XNA Game Studio.....	63
1.5.9.	Unity.....	64
1.5.10.	Другие коммерческие игровые движки .....	64
1.5.11.	Внутренние движки, находящиеся в частной собственности.....	65
1.5.12.	Движки с открытым исходным кодом .....	65
1.5.13.	Игровые движки 2D для непрограммистов .....	66
1.6.	Архитектура среды выполнения движка .....	67
1.6.1.	Целевое аппаратное обеспечение.....	67
1.6.2.	Драйверы устройств .....	67
1.6.3.	Операционная система .....	70
1.6.4.	Сторонние SDK и промежуточное ПО.....	70
1.6.5.	Уровень независимости от платформы.....	73
1.6.6.	Основные системы .....	74
1.6.7.	Управление ресурсами.....	75
1.6.8.	Движок рендеринга.....	75
1.6.9.	Инструменты профилирования и отладки.....	79
1.6.10.	Столкновения и физика .....	81
1.6.11.	Анимация.....	82
1.6.12.	Устройства ввода.....	83
1.6.13.	Аудио.....	83
1.6.14.	Многопользовательская онлайн-игра (игра по сети).....	84

1.6.15.	Система основного геймплея .....	85
1.6.16.	Специфические для игры подсистемы.....	88
1.7.	Инструменты и конвейер ресурсов.....	88
1.7.1.	Инструмент создания контента.....	92
1.7.2.	Конвейер подготовки ресурсов.....	92
1.7.3.	Редактор мира.....	95
1.7.4.	База данных ресурсов.....	96
1.7.5.	Некоторые подходы к архитектуре инструментов .....	97
<b>Глава 2.</b>	<b>Полезные инструменты .....</b>	<b>100</b>
2.1.	Контроль версий.....	100
2.1.1.	Зачем использовать контроль версий .....	101
2.1.2.	Распространенные системы контроля версий.....	101
2.1.3.	Обзор Subversion и TortoiseSVN .....	102
2.1.4.	Настройка репозитория с кодом .....	103
2.1.5.	Установка TortoiseSVN .....	103
2.1.6.	Версии файлов, обновление и коммиты.....	105
2.1.7.	Множественная синхронизация, ветвление и слияние.....	107
2.1.8.	Удаление файлов.....	109
2.2.	Компиляторы, компоновщики и IDE .....	109
2.2.1.	Исходные файлы, заголовки и единицы компиляции .....	110
2.2.2.	Библиотеки, исполняемые файлы и динамически компонуемые библиотеки.....	110
2.2.3.	Проекты и решения.....	111
2.2.4.	Конфигурации сборки .....	112
2.2.5.	Отладка кода .....	122
2.3.	Инструменты профилирования .....	130
2.3.1.	Список профилировщиков .....	132
2.4.	Утечка и нарушение целостности памяти .....	132
2.5.	Другие инструменты .....	133
<b>Глава 3.</b>	<b>Основы разработки игрового ПО.....</b>	<b>135</b>
3.1.	Обзор C++ и лучшие практики .....	135
3.1.1.	Краткий обзор ООП .....	136
3.1.2.	Стандартизация языка C++ .....	142
3.1.3.	Стандарты программирования: для чего и сколько .....	147

3.2.	Поиск и обработка ошибок .....	148
3.2.1.	Типы ошибок.....	148
3.2.2.	Обработка ошибок .....	149
3.2.3.	Реализация обнаружения ошибок и их обработки.....	151
3.3.	Данные, код и схема памяти .....	159
3.3.1.	Числовые представления.....	159
3.3.2.	Примитивные типы данных.....	165
3.3.3.	Килобайт или киббайт .....	170
3.3.4.	Объявления, определения и компоновка.....	171
3.3.5.	Структура памяти программы C/C++ .....	177
3.3.6.	Переменные-члены .....	182
3.3.7.	Расположение объектов в памяти.....	184
3.4.	Основы аппаратного обеспечения компьютеров .....	191
3.4.1.	Обучение на более простых компьютерах прошлых лет .....	191
3.4.2.	Архитектура компьютера.....	192
3.4.3.	Центральный процессор.....	193
3.4.4.	Частота .....	197
3.4.5.	Память .....	198
3.4.6.	Шины.....	199
3.4.7.	Машинный и ассемблерный языки.....	201
3.5.	Архитектура памяти.....	207
3.5.1.	Сопоставление памяти.....	207
3.5.2.	Виртуальная память .....	209
3.5.3.	Архитектуры памяти для уменьшения задержки .....	213
3.5.4.	Иерархии кэш-памяти .....	215
3.5.5.	Неоднородный доступ к памяти.....	222
<b>Глава 4.</b>	<b>Параллелизм и конкурентное программирование .....</b>	<b>226</b>
4.1.	Определение конкурентности и параллелизма .....	227
4.1.1.	Конкурентное выполнение.....	227
4.1.2.	Параллелизм .....	228
4.1.3.	Параллелизм данных и задач .....	229
4.1.4.	Таксономия Флинна.....	230
4.1.5.	Ортогональность конкурентных вычислений и параллелизма .....	233
4.1.6.	Вопросы, рассматриваемые в главе .....	233

4.2.	Неявный параллелизм.....	233
4.2.1.	Конвейерная обработка.....	234
4.2.2.	Задержка и пропускная способность.....	236
4.2.3.	Глубина конвейера .....	236
4.2.4.	Потеря скорости конвейера .....	237
4.2.5.	Зависимости данных .....	237
4.2.6.	Зависимости ветвления.....	239
4.2.7.	Суперскалярные процессоры .....	243
4.2.8.	Очень длинные командные слова .....	245
4.3.	Явный параллелизм.....	247
4.3.1.	Гиперпоточность .....	247
4.3.2.	Многоядерные процессоры.....	248
4.3.3.	Симметричная и асимметричная многопроцессорная обработка.....	250
4.3.4.	Распределенные вычисления .....	251
4.4.	Основы операционной системы.....	252
4.4.1.	Ядро .....	252
4.4.2.	Прерывания .....	254
4.4.3.	Вызовы ядра .....	255
4.4.4.	Вытесняющая многозадачность .....	256
4.4.5.	Процессы .....	257
4.4.6.	Потоки .....	262
4.4.7.	Фиберы.....	271
4.4.8.	Потоки пользовательского уровня и корутины.....	273
4.4.9.	Что еще почитать о процессах и потоках .....	276
4.5.	Введение в параллельное программирование .....	277
4.5.1.	Зачем писать параллельное программное обеспечение .....	277
4.5.2.	Модели параллельного программирования.....	278
4.5.3.	Состояние гонки .....	279
4.5.4.	Критические операции и атомарность.....	282
4.6.	Примитивы синхронизации потоков .....	287
4.6.1.	Мьютексы.....	288
4.6.2.	Критически важные секции .....	290
4.6.3.	Переменные условия.....	291
4.6.4.	Семафоры.....	295
4.6.5.	События Windows .....	299

4.7.	Проблемы с параллелизмом на основе блокировки .....	300
4.7.1.	Взаимная блокировка .....	300
4.7.2.	Динамическая взаимная блокировка .....	302
4.7.3.	Ресурсное голодание .....	303
4.7.4.	Инверсия приоритетов .....	303
4.7.5.	Обедающие философы .....	304
4.8.	Несколько лучших практик параллелизма .....	305
4.8.1.	Правила глобального порядка .....	305
4.8.2.	Алгоритмы на основе транзакций.....	306
4.8.3.	Минимизация раздоров.....	307
4.8.4.	Безопасность потоков .....	307
4.9.	Параллелизм без блокировок.....	308
4.9.1.	Причины ошибок в гонке данных.....	310
4.9.2.	Реализация атомарности.....	311
4.9.3.	Барьеры .....	319
4.9.4.	Семантика упорядочения памяти.....	322
4.9.5.	Атомарные переменные.....	332
4.9.6.	Параллельность в интерпретируемых языках программирования.....	335
4.9.7.	Спин-блокировки .....	336
4.9.8.	Транзакции без блокировок .....	344
4.9.9.	Связанный список без блокировки .....	345
4.9.10.	Дополнительная литература по программированию без блокировки .....	346
4.10.	SIMD/векторная обработка .....	347
4.10.1.	Набор инструкций SSE и его регистры .....	348
4.10.2.	Использование SSE для векторизации цикла.....	352
4.10.3.	Векторизованное скалярное произведение .....	353
4.10.4.	Векторно-матричное умножение с помощью SSE .....	358
4.10.5.	Матрица-матричное умножение с SSE .....	358
4.10.6.	Обобщенная векторизация .....	359
4.10.7.	Предикация векторов.....	360
4.11.	Введение в программирование GPGPU.....	363
4.11.1.	Параллельные вычисления данных .....	364
4.11.2.	Вычислительные ядра.....	365

4.11.3.	Выполнение ядра.....	367
4.11.4.	Потоки GPU и группы потоков.....	368
4.11.5.	Дополнительная литература .....	372
<b>Глава 5.</b>	<b>3D-математика для игр .....</b>	<b>373</b>
5.1.	Решение 3D-задач в 2D.....	373
5.2.	Точки и векторы.....	374
5.2.1.	Точки и декартовы координаты.....	374
5.2.2.	Левосторонние и правосторонние системы координат.....	375
5.2.3.	Векторы.....	376
5.2.4.	Векторные операции .....	377
5.2.5.	Линейная интерполяция точек и векторов .....	388
5.3.	Матрицы.....	388
5.3.1.	Умножение матриц.....	389
5.3.2.	Представление точек и векторов в виде матриц.....	390
5.3.3.	Единичная матрица .....	391
5.3.4.	Инвертирование матриц .....	391
5.3.5.	Транспонирование.....	391
5.3.6.	Однородная система координат .....	392
5.3.7.	Матрицы базовых преобразований .....	394
5.3.8.	Матрицы $4 \times 3$ .....	397
5.3.9.	Координатное пространство .....	397
5.3.10.	Переход к новому базису .....	401
5.3.11.	Преобразование векторов нормали.....	404
5.3.12.	Хранение матриц в памяти.....	404
5.4.	Кватернионы.....	406
5.4.1.	Единичные кватернионы как трехмерные вращения .....	407
5.4.2.	Операции с кватернионами .....	408
5.4.3.	Вращение векторов через кватернионы.....	409
5.4.4.	Кватернионно-матричная эквивалентность .....	410
5.4.5.	Линейная интерполяция вращения .....	412
5.5.	Сравнение представлений вращения .....	414
5.5.1.	Углы Эйлера .....	415
5.5.2.	Матрицы $3 \times 3$ .....	415
5.5.3.	Ось + угол.....	416

5.5.4.	Кватернионы .....	416
5.5.5.	Преобразования SRT.....	416
5.5.6.	Двойные кватернионы .....	417
5.5.7.	Вращения и степени свободы.....	418
5.6.	Другие полезные математические объекты .....	419
5.6.1.	Прямые, лучи и отрезки .....	419
5.6.2.	Сферы .....	420
5.6.3.	Плоскости.....	420
5.6.4.	Параллельные осям ограничивающие параллелепипеды .....	422
5.6.5.	Ориентированные ограничивающие параллелепипеды .....	422
5.6.6.	Усеченная пирамида .....	423
5.6.7.	Выпуклые многогранные области .....	424
5.7.	Генерация случайных чисел.....	424
5.7.1.	Линейные конгруэнтные генераторы .....	424
5.7.2.	Вихрь Мерсенна .....	425
5.7.3.	Мать всего, Xorshift и KISS99 .....	425
5.7.4.	PCG .....	426

## **Часть II. Низкоуровневые системы движка**

<b>Глава 6.</b>	<b>Системы поддержки движка.....</b>	<b>428</b>
6.1.	Подсистема запуска и остановки.....	428
6.1.1.	Порядок статической инициализации C++ (или его отсутствие) .....	428
6.1.2.	Простой работающий подход .....	431
6.1.3.	Некоторые примеры реальных движков .....	433
6.2.	Управление памятью .....	436
6.2.1.	Оптимизация динамического распределения памяти.....	436
6.2.2.	Фрагментация памяти .....	446
6.3.	Контейнеры .....	450
6.3.1.	Операции с контейнерами.....	452
6.3.2.	Итераторы .....	452
6.3.3.	Алгоритмическая сложность.....	454
6.3.4.	Создание пользовательских контейнерных классов.....	455
6.3.5.	Динамические массивы и выделение памяти фрагментами .....	459
6.3.6.	Словари и хеш-таблицы .....	460



6.4.	Строки.....	464
6.4.1.	Проблема со строками .....	464
6.4.2.	Классы строк .....	465
6.4.3.	Уникальные идентификаторы.....	466
6.4.4.	Локализация.....	469
6.5.	Конфигурация движка .....	479
6.5.1.	Параметры загрузки и сохранения.....	479
6.5.2.	Параметры для каждого пользователя.....	480
6.5.3.	Управление конфигурацией в некоторых реальных движках.....	481
<b>Глава 7.</b>	<b>Ресурсы и файловая система .....</b>	<b>486</b>
7.1.	Файловая система.....	487
7.1.1.	Имена файлов и пути к ним.....	487
7.1.2.	Базовый файловый ввод/вывод.....	491
7.1.3.	Асинхронный файловый ввод/вывод .....	493
7.2.	Менеджер ресурсов .....	497
7.2.1.	Автономное управление ресурсами и цепочка инструментов.....	498
7.2.2.	Управление ресурсами среды выполнения.....	507
<b>Глава 8.</b>	<b>Игровой цикл и симуляция в реальном времени .....</b>	<b>526</b>
8.1.	Цикл рендеринга .....	526
8.2.	Цикл игры.....	527
8.2.1.	Простой пример: пинг-понг .....	527
8.3.	Архитектурные стили цикла игры .....	529
8.3.1.	Конвейер сообщений Windows.....	529
8.3.2.	Фреймворки на основе обратных вызовов.....	530
8.3.3.	Обновление на основе событий.....	531
8.4.	Абстрактные временные шкалы.....	532
8.4.1.	Реальное время.....	532
8.4.2.	Игровое время.....	532
8.4.3.	Локальное и глобальное время.....	533
8.5.	Измерение времени и работа с ним .....	534
8.5.1.	Частота смены кадров и время.....	534
8.5.2.	От частоты кадров к скорости.....	535
8.5.3.	Измерение реального времени с помощью таймера высокого разрешения .....	539

8.5.4.	Единицы измерения времени и переменные часов .....	540
8.5.5.	Работа с точками останова .....	543
8.6.	Многопроцессорные игровые циклы .....	544
8.6.1.	Разложение задания .....	544
8.6.2.	Один поток на подсистему .....	545
8.6.3.	Разбиение/сборка .....	546
8.6.4.	Система заданий .....	549
<b>Глава 9.</b>	<b>Устройства HID .....</b>	<b>559</b>
9.1.	Виды HID-устройств .....	559
9.2.	Взаимодействие с HID-устройствами .....	561
9.2.1.	Опрашивание .....	561
9.2.2.	Прерывания .....	562
9.2.3.	Беспроводные устройства .....	562
9.3.	Виды элементов управления .....	562
9.3.1.	Цифровые кнопки .....	562
9.3.2.	Аналоговые оси и кнопки .....	564
9.3.3.	Относительные оси .....	565
9.3.4.	Акселерометр .....	565
9.3.5.	Положение в трехмерном пространстве при использовании Wiimote или DualShock .....	566
9.3.6.	Камеры .....	566
9.4.	Виды вывода .....	569
9.4.1.	Вибрация .....	569
9.4.2.	Силовая обратная связь .....	569
9.4.3.	Звук .....	569
9.4.4.	Другие разновидности ввода и вывода .....	569
9.5.	Системы игрового движка для работы с HID-устройствами .....	570
9.5.1.	Типичные требования .....	570
9.5.2.	Мертвые зоны .....	571
9.5.3.	Фильтрация аналоговых сигналов .....	571
9.5.4.	Обнаружение событий ввода .....	573
9.5.5.	Управление несколькими HID-устройствами для нескольких игроков .....	580
9.5.6.	Межплатформенные HID-системы .....	580
9.5.7.	Переназначение элементов управления .....	582

9.5.8.	Контекстный ввод .....	583
9.5.9.	Отключение элементов управления .....	584
9.6.	НID-устройства на практике .....	585
<b>Глава 10.</b>	<b>Инструменты для отладки и разработки.....</b>	<b>586</b>
10.1.	Журналирование и трассировка.....	586
10.1.1.	Форматированный вывод с помощью функции OutputDebugString().....	587
10.1.2.	Уровень детализации.....	588
10.1.3.	Каналы.....	589
10.1.4.	Копирование вывода в файл.....	590
10.1.5.	Отчеты о сбоях .....	590
10.2.	Средства отладочной отрисовки.....	591
10.2.1.	API для отладочной отрисовки .....	593
10.3.	Внутриигровые меню.....	597
10.4.	Внутриигровая консоль .....	598
10.5.	Отладочные камеры и остановка игры.....	599
10.6.	Читы.....	600
10.7.	Снимки экрана и запись видео .....	601
10.8.	Внутриигровое профилирование .....	602
10.8.1.	Иерархическое профилирование.....	603
10.8.2.	Экспорт в Excel.....	608
10.9.	Внутриигровые показатели использования памяти и обнаружение утечек.....	609

### **Часть III. Графика, движение и звук**

<b>Глава 11.</b>	<b>Движок рендеринга.....</b>	<b>614</b>
11.1.	Основы растеризации треугольников с буферизацией глубины .....	614
11.1.1.	Описание сцены .....	616
11.1.2.	Описание визуальных свойств поверхности .....	626
11.1.3.	Основы освещения.....	640
11.1.4.	Виртуальная камера .....	649
11.2.	Конвейер рендеринга .....	660
11.2.1.	Общая схема конвейера рендеринга.....	661
11.2.2.	Инструментальный этап .....	663
11.2.3.	Этап подготовки ресурсов .....	665

11.2.4.	Конвейер графического процессора .....	666
11.2.5.	Программируемые шейдеры.....	671
11.2.6.	Сглаживание .....	677
11.2.7.	Этап приложения .....	680
11.3.	Продвинутые методы освещения и глобальное освещение .....	691
11.3.1.	Освещение на основе изображения.....	691
11.3.2.	Освещение с расширенным динамическим диапазоном .....	695
11.3.3.	Глобальное освещение .....	696
11.3.4.	Отложенный рендеринг .....	703
11.3.5.	Физически корректное затенение .....	704
11.4.	Визуальные эффекты и наложения .....	705
11.4.1.	Эффекты частиц .....	705
11.4.2.	Декали.....	706
11.4.3.	Эффекты окружения.....	707
11.4.4.	Наложения .....	710
11.4.5.	Гамма-коррекция .....	712
11.4.6.	Полноэкранные постэффекты .....	714
11.5.	Дополнительная литература .....	714
<b>Глава 12.</b>	<b>Системы анимации .....</b>	<b>716</b>
12.1.	Разновидности анимации персонажей .....	716
12.1.1.	Келевая анимация .....	717
12.1.2.	Жесткая иерархическая анимация.....	717
12.1.3.	Поверхинная анимация и морфинг-мишени.....	718
12.1.4.	Скиновая анимация .....	720
12.1.5.	Методы анимации как методы сжатия данных.....	721
12.2.	Скелеты.....	722
12.2.1.	Скелетная иерархия.....	723
12.2.2.	Представление скелета в памяти .....	723
12.3.	Позы и положения .....	724
12.3.1.	Поза привязки.....	725
12.3.2.	Локальные позы .....	725
12.3.3.	Глобальные позы.....	728
12.4.	Клипы .....	729
12.4.1.	Локальная временная шкала .....	730
12.4.2.	Глобальная временная шкала .....	734

12.4.3.	Сравнение глобального и локального таймеров.....	736
12.4.4.	Простой формат анимационных данных .....	739
12.4.5.	Непрерывные каналные функции .....	740
12.4.6.	Метаканалы .....	741
12.4.7.	Отношения между мешами, скелетами и клипами .....	742
12.5.	Скининг и генерация палитры матриц.....	744
12.5.1.	Сведения о скининге для отдельной вершины .....	744
12.5.2.	Математический аспект скининга.....	745
12.6.	Слияние анимации .....	749
12.6.1.	Линейная интерполяция.....	749
12.6.2.	Способы применения слияния методом LERP.....	751
12.6.3.	Сложные слияния методом LERP.....	756
12.6.4.	Частичное скелетное слияние.....	760
12.6.5.	Аддитивное слияние.....	761
12.6.6.	Способы применения аддитивного слияния .....	764
12.7.	Постобработка.....	766
12.7.1.	Процедурная анимация .....	766
12.7.2.	Инверсная кинематика.....	767
12.7.3.	Тряпичные куклы .....	769
12.8.	Методы сжатия .....	769
12.8.1.	Пропуск каналов.....	769
12.8.2.	Квантование .....	770
12.8.3.	Частота дискретизации и пропуск семплов .....	774
12.8.4.	Сжатие на основе кривых .....	774
12.8.5.	Сжатие с использованием вейвлетов .....	775
12.8.6.	Выборочные загрузка и потоковая передача .....	775
12.9.	Конвейер анимации.....	775
12.10.	Конечные автоматы действий.....	778
12.10.1.	Подход с плоским средним взвешенным .....	779
12.10.2.	Деревья слияния .....	783
12.10.3.	Параметры состояния и дерева слияния.....	787
12.10.4.	Переходы .....	792
12.10.5.	Управляющие параметры .....	795
12.11.	Ограничения .....	797
12.11.1.	Крепления .....	797
12.11.2.	Выравнивание объектов.....	798

12.11.3. Инверсная кинематика движений захвата рукой .....	802
12.11.4. Извлечение движений и инверсная кинематика ног .....	803
12.11.5. Другие виды ограничений .....	805
<b>Глава 13. Столкновения и динамика твердого тела.....</b>	<b>807</b>
13.1. Нужна ли в вашей игре физика? .....	808
13.1.1. Что можно делать с системой симуляции физики .....	808
13.1.2. Делает ли физика игру интересной?.....	809
13.1.3. Влияние физики на игру.....	810
13.2. Промежуточный слой для столкновений/физики.....	813
13.2.1. ODE.....	813
13.2.2. Bullet .....	813
13.2.3. TrueAxis.....	814
13.2.4. PhysX .....	814
13.2.5. Havok .....	814
13.2.6. Physics Abstraction Layer (PAL) .....	814
13.2.7. Digital Molecular Matter .....	815
13.3. Система обнаружения столкновений.....	815
13.3.1. Элементы, которые могут сталкиваться между собой .....	816
13.3.2. Мир столкновений/физики.....	817
13.3.3. Концепции геометрических форм .....	819
13.3.4. Примитивы столкновения.....	820
13.3.5. Проверки на столкновение и аналитическая геометрия .....	825
13.3.6. Оптимизация производительности.....	834
13.3.7. Запросы о столкновениях.....	836
13.3.8. Фильтрация столкновений .....	840
13.4. Динамика твердого тела.....	842
13.4.1. Некоторые основы.....	843
13.4.2. Линейная динамика.....	845
13.4.3. Решение уравнений движения.....	847
13.4.4. Численное интегрирование.....	849
13.4.5. Угловая динамика в двухмерном пространстве.....	853
13.4.6. Угловая динамика в трехмерном пространстве .....	857
13.4.7. Реакция на столкновение.....	862
13.4.8. Ограничения .....	869
13.4.9. Управление движениями твердых тел .....	873
13.4.10. Отдельный шаг симуляции столкновений/физики.....	875

13.5. Интеграция физического движка в игру .....	877
13.5.1. Связывание игровых объектов и твердых тел.....	877
13.5.2. Обновление симуляции .....	882
13.5.3. Пример использования систем столкновений и физики в игре .....	885
13.6. Расширенные физические возможности .....	894
<b>Глава 14. Звук .....</b>	<b>896</b>
14.1. Физика звука .....	897
14.1.1. Свойства звуковых волн .....	897
14.1.2. Субъективная громкость звука и децибел.....	899
14.1.3. Распространение звуковой волны.....	901
14.1.4. Восприятие положения в пространстве .....	907
14.2. Математика звука.....	908
14.2.1. Сигналы .....	908
14.2.2. Преобразование сигналов.....	909
14.2.3. Линейные стационарные системы.....	910
14.2.4. Импульсная характеристика систем ЛСС.....	911
14.2.5. Частотная область и преобразование Фурье .....	917
14.3. Аудиотехнологии.....	925
14.3.1. Аналоговые аудиотехнологии .....	925
14.3.2. Цифровые аудиотехнологии.....	930
14.4. Рендеринг звука в 3D.....	937
14.4.1. Краткий обзор процесса рендеринга 3D-звука .....	938
14.4.2. Моделирование мира звука .....	938
14.4.3. Затухание в зависимости от расстояния .....	939
14.4.4. Панорамирование.....	941
14.4.5. Распространение, реверберация и акустика .....	947
14.4.6. Эффект Доплера .....	954
14.5. Архитектура звукового движка.....	955
14.5.1. Конвейер обработки звуков .....	956
14.5.2. Концепции и терминология.....	959
14.5.3. Голосовая шина.....	960
14.5.4. Главный микшер .....	962
14.5.5. Главная шина вывода .....	964
14.5.6. Реализация шины .....	965
14.5.7. Управление ресурсами.....	967

14.5.8.	Микширование в игре.....	969
14.5.9.	Обзор звуковых движков.....	973
14.6.	Звуковые возможности, характерные для разных видов игр.....	976
14.6.1.	Поддержка разделенного экрана.....	977
14.6.2.	Система диалогов.....	977
14.6.3.	Музыка.....	989

## **Часть IV. Игровой процесс**

<b>Глава 15.</b>	Введение в системы игрового процесса.....	992
15.1.	Структура игрового мира.....	993
15.1.1.	Элементы мира.....	993
15.1.2.	Области игрового мира.....	995
15.1.3.	Высокоуровневый игровой поток.....	997
15.2.	Реализация динамических элементов: игровые объекты.....	997
15.2.1.	Объектные модели игры.....	999
15.2.2.	Игровые объекты на этапах проектирования и выполнения.....	1000
15.3.	Игровые движки на основе данных.....	1001
15.4.	Редактор игрового мира.....	1002
15.4.1.	Типичные возможности редактора игрового мира.....	1004
15.4.2.	Интегрированные средства управления ресурсами.....	1011
<b>Глава 16.</b>	Системы организации игрового процесса на этапе выполнения.....	1014
16.1.	Компоненты системы организации игрового процесса.....	1014
16.2.	Архитектуры объектной модели времени выполнения.....	1017
16.2.1.	Архитектуры объектного стиля.....	1018
16.2.2.	Архитектуры на основе свойств.....	1031
16.3.	Форматы областей игрового мира.....	1035
16.3.1.	Двоичные образы объектов.....	1036
16.3.2.	Описание сериализованных игровых объектов.....	1037
16.3.3.	Спаунеры и схемы типов.....	1038
16.4.	Загрузка и потоковая передача игровых миров.....	1042
16.4.1.	Простая загрузка уровней.....	1043
16.4.2.	Шаг в направлении бесшовной загрузки: шлюзы.....	1044
16.4.3.	Потоковая загрузка игрового мира.....	1045
16.4.4.	Управление памятью для создания объектов.....	1047
16.4.5.	Сохраненные игры.....	1049



16.5.	Ссылки на объекты и запросы к игровому миру.....	1051
16.5.1.	Указатели.....	1051
16.5.2.	Умные указатели.....	1052
16.5.3.	Дескрипторы .....	1054
16.5.4.	Запросы игровых объектов .....	1056
16.6.	Обновление игровых объектов в реальном времени .....	1058
16.6.1.	Простой подход (который не работает).....	1059
16.6.2.	Влияние пакетных обновлений на производительность.....	1061
16.6.3.	Взаимные зависимости объектов и подсистем.....	1064
16.7.	Применение конкурентности к обновлению игровых объектов .....	1071
16.7.1.	Конкурентные подсистемы движка .....	1072
16.7.2.	Асинхронный подход к проектированию.....	1073
16.7.3.	Зависимости заданий и степень параллелизма.....	1075
16.7.4.	Распараллеливание самой объектной модели игры.....	1078
16.8.	События и передача сообщений.....	1083
16.8.1.	Проблема со статически типизированным связыванием функций .....	1083
16.8.2.	Инкапсуляция события в виде объекта .....	1085
16.8.3.	Типы событий .....	1086
16.8.4.	Аргументы событий.....	1087
16.8.5.	Обработчики событий.....	1089
16.8.6.	Распаковка аргументов событий.....	1090
16.8.7.	Цепочки обязанностей.....	1090
16.8.8.	Подписка на события .....	1092
16.8.9.	Использовать ли очередь .....	1093
16.8.10.	Некоторые проблемы с незамедлительной отправкой событий.....	1098
16.8.11.	Системы передачи событий/сообщений на основе данных .....	1099
16.9.	Скрипты.....	1102
16.9.1.	Для выполнения кода и описания данных .....	1103
16.9.2.	Характеристики языков программирования .....	1103
16.9.3.	Некоторые распространенные игровые скриптовые языки .....	1105
16.9.4.	Архитектуры для скриптования.....	1109
16.9.5.	Возможности игровых скриптовых языков на этапе выполнения .....	1111
16.10.	Высокоуровневый игровой поток .....	1123

## Часть V. Подведение итогов

<b>Глава 17.</b> Хотите сказать, что это еще не все? .....	1126
17.1. Некоторые подсистемы движка, которые мы не рассмотрели .....	1126
17.1.1. Видеоплеер .....	1126
17.1.2. Сетевые возможности для многопользовательских игр .....	1127
17.2. Системы игрового процесса .....	1127
17.2.1. Игровая механика .....	1127
17.2.2. Камеры .....	1127
17.2.3. Искусственный интеллект .....	1128
17.2.4. Другие системы игрового процесса .....	1129
Список литературы .....	1130

# 12 Системы анимации

В большинстве современных 3D-игр центральную роль играют *персонажи* — часто это люди или человекоподобные существа, иногда животные или монстры. Отличительной чертой персонажей является то, что они должны двигаться плавно и органично. Это порождает множество дополнительных технических проблем, помимо тех, которые требуется решить в случае моделирования и анимации таких твердых объектов, как транспортные средства, снаряды, футбольные мячи и фигуры тетриса. Задачу наделения персонажей естественно выглядящими движениями решает компонент движка, называемый *системой анимации персонажей*.

Как мы увидим далее, система анимации предоставляет разработчикам игр мощный набор инструментов, которые можно применять не только к персонажам, но и к другим объектам. Возможности системы анимации могут с успехом задействоваться в любом мало-мальски подвижном объекте игры. Поэтому, когда вы видите в игре транспортное средство с подвижными частями, шарнирно сочлененный механизм, слегка покачивающиеся на ветру деревья или даже взрывающееся здание, велика вероятность того, что в этом объекте хотя бы отчасти используется анимационная система игрового движка.

## 12.1. Разновидности анимации персонажей

Технологии анимации персонажей прошли долгий путь развития со времен серии игр *Donkey Kong*. В первых играх использовались простейшие методы имитации движений живых существ. По мере роста возможностей игрового аппаратного обеспечения стало возможным применение более сложных методов в режиме реального времени.

Сегодня в распоряжении разработчиков игр имеется множество мощных методов анимации. В этом разделе мы кратко коснемся эволюции анимации персонажей и в общих чертах рассмотрим три метода, которые получили наибольшее распространение в современных игровых движках.

### 12.1.1. Келевая анимация

Предшественником всех методов игровой анимации является *традиционная рисованная анимация*. Речь идет о методе, который использовался уже в первых мультипликационных фильмах. Иллюзия движения создается отображением последовательности быстро сменяющихся неподвижных изображений — *кадров*. 3D-рендеринг в реальном времени можно рассматривать как электронную разновидность традиционной анимации, в которой перед зрителем проходит последовательность неподвижных полноэкранных изображений, создавая иллюзию движения.

Особой разновидностью традиционной анимации является *келевая анимация*. *Кель* — это прозрачный лист целлулоида (cel — от слова celluloid), на котором можно рисовать изображения. Размещая анимированную последовательность келей поверх фиксированного фонового рисунка, можно создавать иллюзию движения, не перерисовывая раз за разом статический фон.

Аналогом келевой анимации в компьютерной графике является так называемая *спрайтовая анимация*. Спрайт — это небольшое растровое изображение, которое можно накладывать поверх полноэкранного фонового изображения, не искажая его. Обычно оно отрисовывается с помощью специализированного графического аппаратного обеспечения. То есть в двухмерной игровой анимации спрайт играет ту же роль, которую играл кель в традиционной анимации. Этот метод анимации был основным в эпоху двухмерных игр. На рис. 12.1 показана хорошо известная последовательность спрайтов, применявшаяся для создания иллюзии бегущего гуманоидного персонажа практически во всех играх для приставок Mattel Intellivision. Эта последовательность кадров организована таким образом, чтобы она плавно проигрывалась и при бесконечном повторении, — такой подход называется *циклической анимацией*. Пользуясь современной терминологией, данную анимацию можно назвать *циклом бега*, поскольку она создает иллюзию того, что персонаж бежит. Обычно персонаж имеет несколько циклов анимации: различные циклы бездействия, цикл ходьбы и цикл бега.



Рис. 12.1. Последовательность спрайтов, применявшаяся в большинстве игр для приставок Intellivision

### 12.1.2. Жесткая иерархическая анимация

Системы анимации первых 3D-игр, таких как *Doom*, незначительно отличались от спрайтовой анимации: монстры в этих играх представляли собой не что иное, как обращенные к камере квадранты, создающие иллюзию движения за счет отображения на них последовательности текстурных растровых изображений, называемых

*анимированными текстурами.* Эта техника используется и сегодня для объектов с низким разрешением и/или удаленных, таких как толпа на стадионе или множество солдат, сражающихся на заднем плане. Однако развитие трехмерной графики потребовало применения для высококачественных персонажей переднего плана более совершенных методов анимации персонажей.

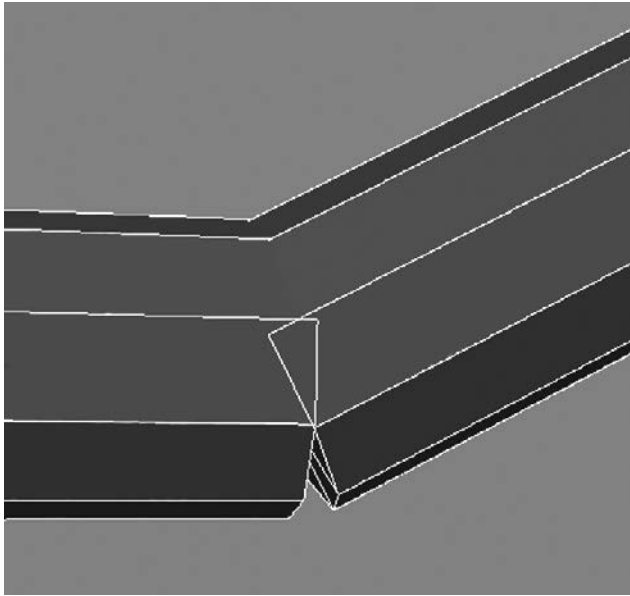
Изначально для трехмерной анимации персонажей использовали метод *жесткой иерархической анимации*. При этом персонаж моделируется как набор жестких фигур. Так, гуманоидный персонаж обычно разбивается на следующие составные части: таз, торс, верхние и нижние части рук, верхние и нижние части ног, кисти рук, ступни ног, голова. Эти жесткие части связываются друг с другом в иерархическом порядке, подобно тому как соединены в суставах кости млекопитающих животных. Это заставляет персонаж двигаться естественным для него образом. Так, например, когда начинает двигаться верхняя часть руки, за ней автоматически следуют нижняя часть руки и кисть. Обычно в качестве корневого узла иерархии выступает таз, его непосредственными дочерними узлами являются торс и верхние части ног и т. д.:

```
Pelvis
  Torso
    UpperRightArm
      LowerRightArm
        RightHand
    UpperLeftArm
      UpperLeftArm
        LeftHand
  Head
  UpperRightLeg
    LowerRightLeg
      RightFoot
  UpperLeftLeg
    UpperLeftLeg
      LeftFoot
```

Большой проблемой жесткой иерархической анимации является то, что тело персонажа часто начинает выглядеть не очень привлекательно из-за «растрескивания» в суставах (рис. 12.2). Этот метод хорошо подходит для роботов и машин, которые действительно состоят из жестких компонентов, но не выдерживает никакой критики в случае его применения к персонажам, которые хоть в какой-то мере обладают плотью.

### 12.1.3. Поверхинная анимация и морфинг-мишени

Жесткая иерархическая анимация, часто выглядит неестественно именно в силу своей жесткости. Поэтому нужно перемещать отдельные вершины таким образом, чтобы треугольники могли растягиваться, обеспечивая более естественный вид движений.



**Рис. 12.2.** «Растрескивание» в суставах — большая проблема жесткой иерархической анимации

Помимо других способов, для этого можно использовать такой прямолинейный метод, как *поверхинная анимация*. При этом вершины сетки анимируются художником, после чего данные о движении экспортируются и используются в качестве указания игровому движку о том, как следует перемещать каждую из них в динамическом режиме. Данный метод позволяет деформировать сетку практически любым воображимым способом (ограничением является лишь степень тесселяции поверхности). Однако это информационно емкий метод, поскольку для каждой вершины сетки требуется сохранять изменяющиеся во времени данные о движении. В силу этого он практически не применяется в играх в режиме реального времени.

В некоторых играх в режиме реального времени используется разновидность этой техники, известная как *анимация по морфинг-мишеням*. При этом аниматор перемещает вершины сетки, задействуя сравнительно небольшой набор фиксированных крайних поз. Анимация выполняется *смешиванием* двух или большего количества фиксированных поз в динамическом режиме. Положение каждой вершины рассчитывается посредством простой линейной интерполяции (linear interpolation, LERP) между положениями вершины в каждой из крайних поз.

Метод морфинг-мишеней часто используется для лицевой анимации, потому что у человеческого лица чрезвычайно сложная анатомия и оно приводится в движение примерно 50 мышцами. Это дает аниматору полный контроль над каждой



**Рис. 12.3.** Набор лицевых морфинг-мишеней для персонажа Элли из игры *The Last of Us: Remastered* (снимок экрана с сайта <https://www.gameenginebook.com>)

вершиной лицевой сетки, позволяя реализовывать как едва уловимые, так и очень большие движения, хорошо имитирующие мускулатуру лица. На рис. 12.3 показано, как может выглядеть набор лицевых морфинг-мишеней.

По мере роста вычислительных мощностей некоторые студии вместо морфинг-мишеней начали использовать суставные лицевые риги, включающие в себя сотни суставов. Другие студии сочетают оба метода, создавая базовое выражение лица с помощью суставных ригов и внося небольшие изменения с помощью морфинг-мишеней.

#### 12.1.4. Скинковая анимация

По мере дальнейшего роста возможностей игрового аппаратного обеспечения был разработан метод *скинковой анимации*. Он обладает почти всеми плюсами поверхностной анимации и анимации по морфинг-мишеням, позволяя деформировать треугольники анимируемой сетки. Все это сочетается с намного более высокой эффективностью в плане производительности и потребления памяти, характерной для жесткой иерархической анимации. С помощью этого метода можно довольно реалистично имитировать движение кожи и одежды.

Скинковая анимация была впервые применена в таких играх, как *Super Mario 64*, и по сей день остается наиболее распространенным методом анимации как в индустрии компьютерных игр, так и в кино. Персонажи многих известных современных игр и кинофильмов как минимум частично, а то и полностью анимировались с помощью этого метода. Это, в частности, динозавры из фильма «Парк юрского периода», Солид Снейк из серии игр *Metal Gear*, Голлум из фильма «Властелин колец», Натан Дрейк из игры *Uncharted*, Базз Лайтер из мультфильма «История игрушек», Маркус Феникс из серии игр *Gears of War* и Джоэл из серии игр *The Last of Us*. Далее в этой главе мы сосредоточимся главным образом на изучении скинковой/скелетной анимации.

Как и в жесткой иерархической анимации, в скинковой анимации составляется *скелет* из жестких костей. Но вместо того, чтобы отрисовывать эти жесткие части на экране, их оставляют скрытыми. К суставам скелета привязывается так называемый *скин* — плавная непрерывная триангулярная сетка, вершины которой отслеживают движения суставов. Для каждой вершины можно определить весовые коэффициенты влияния нескольких суставов, что обеспечивает реалистичное растяжение скина при их движении.

На рис. 12.4 представлен игровой персонаж Горноста́й Крэ́нк (*Crank the Weasel*), созданный Эриком Браунингом в 2001 году для компании Midway Home Entertainment. Внешняя оболочка, или *скин*, Крэнка состоит из сетки треуголь-

ников, как и любая другая 3D-модель. А внутри него мы видим жесткие кости и суставы, которые приводят скин в движение.

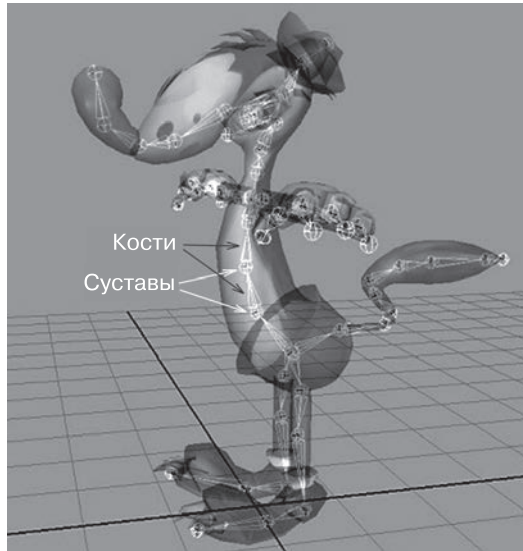


Рис. 12.4. Горностай Крэнк с внутренней скелетной структурой

### 12.1.5. Методы анимации как методы сжатия данных

В идеализированно гибкой системе анимации у аниматора будет контроль буквально над каждым бесконечно малым участком поверхности объекта. Конечно, такой подход приведет к созданию анимации, потенциально содержащей бесконечно большой объем данных. Анимация вершин триангулярной сетки является, по сути, упрощением этого идеального случая — мы *сжимаем* информацию, необходимую для описания анимации, ограничиваясь только возможностью перемещать вершины. (Анимация набора контрольных точек представляет собой аналог вершинной анимации для моделей, состоящих из патчей более высокого порядка.) Морфинг-мишени можно рассматривать как дополнительный уровень сжатия, обеспечиваемый за счет наложения на систему дополнительных ограничений, допускающих перемещение вершин только по линейным траекториям между фиксированным количеством заранее заданных положений. Скелетная анимация, по сути, представляет собой еще один способ сжатия данных вершинной анимации путем наложения ограничений. В этом случае ограничение состоит в том, что сравнительно большое количество вершин должно отслеживать движения сравнительно небольшого количества суставов скелета.

Решая, какой из методов анимации лучше выбрать в том или ином случае, часто полезно думать о них как о методах сжатия, во многом аналогичных методам

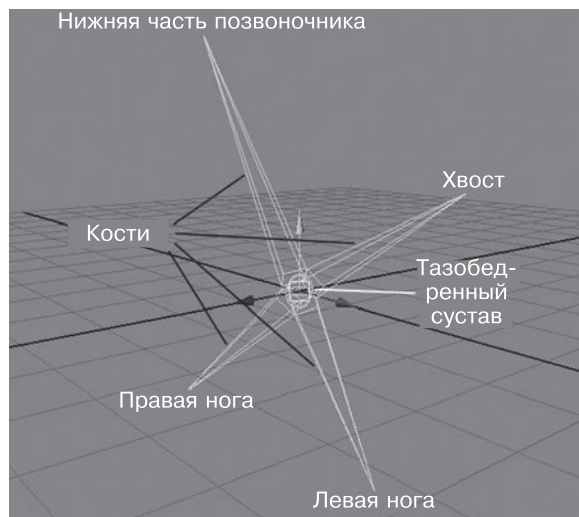


сжатия видео. В общем случае нужно выбирать метод анимации, который обеспечивает наилучшее сжатие, не создавая неприемлемых визуальных артефактов. Скелетная анимация дает наилучшее сжатие, когда движение одного сустава транслируется в движение множества вершин. Так, скелет позволяет очень эффективно перемещать конечности персонажа, поскольку они ведут себя почти как твердые тела.

Однако движения лица обычно намного сложнее и требуют более независимого перемещения отдельных вершин. Для того чтобы скелетный метод обеспечивал достаточно реалистичную анимацию лица, количество используемых суставов должно быть ненамного меньше количества вершин в сетке, что снижает его эффективность в качестве метода сжатия. Это один из главных доводов в пользу того, чтобы использовать для анимации лица метод морфинг-мишеней, а не скелетный подход. (Другой веский довод таков: применение морфинг-мишеней обычно более привычный для аниматоров способ работы.)

## 12.2. Скелеты

Скелет состоит из *иерархии* жестких элементов, называемых *суставами*. Хотя в игровой индустрии под суставами и костями часто понимают одно и то же, такое употребление слова «кости» не совсем корректно. Если судить с технической точки зрения, то суставы — это объекты, которыми напрямую манипулирует аниматор, а кости — пустое пространство между ними. В качестве примера рассмотрим тазобедренный сустав модели Горностая Крэнка. Он соединен с четырьмя другими суставами (в хвосте, позвоночнике и двух ногах) и в силу этого образует четыре выходящие из него кости (рис. 12.5). Игровые движки не обращают никакого



**Рис. 12.5.** Тазобедренный сустав персонажа соединен с четырьмя другими суставами (в хвосте, позвоночнике и двух ногах) и, соответственно, образует четыре кости

внимания на кости — для них важны только суставы. Поэтому, когда вы слышите слово «кость» в контексте игровой индустрии, знайте, что при этом почти всегда имеются в виду суставы.

### 12.2.1. Скелетная иерархия

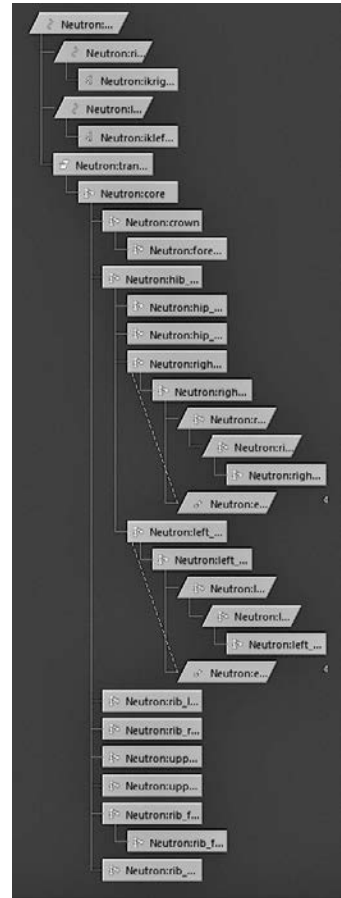
Как мы уже упоминали, суставы в скелете образуют иерархию или древовидную структуру. Один сустав выбирается корневым, а все остальные становятся по отношению к нему дочерними, внучатыми и т. д. Типичная иерархия суставов для скиновой анимации выглядит практически так же, как типичная иерархия для жесткой иерархической анимации. Так, иерархия суставов гуманоидного персонажа может выглядеть примерно так, как показано на рис. 12.6.

Любому суставу присваивается индекс в диапазоне от 0 до  $N - 1$ . Поскольку каждый сустав имеет один и только один родительский сустав, иерархическая структура скелета может быть полностью определена путем сохранения в каждом суставе индекса его родительского сустава. Так как у корневого сустава нет родительского сустава, его индексу родительского сустава обычно присваивается недопустимое значение, например  $-1$ .

### 12.2.2. Представление скелета в памяти

Для представления скелета обычно используется небольшая структура данных верхнего уровня, содержащая массив структур данных отдельных суставов. Чаще всего выбирается такой порядок следования суставов в массиве, при котором дочерний сустав располагается после родительского. Это означает, что корнем скелета всегда является нулевой сустав.

Для обращения к отдельным суставам в пределах структуры данных обычно используются *индексы суставов*. Дочерний сустав ссылается на родительский, указывая его индекс. Аналогичным образом в заскиненной триангулярной сетке вершина ссылается по индексу на суставы, к которым привязана. Это гораздо эффективнее, чем ссылаться на суставы по имени, с точки зрения как потребления памяти (если принять максимальное число суставов в скелете равным 256, индекс сустава может занимать всего 8 бит), так и затрат времени на поиск нужного сустава (мы можем мгновенно перейти к нужному суставу в массиве, используя его индекс).



**Рис. 12.6.** Пример скелетной иерархии, представленной в виде гиперграфа в приложении Maya

Структуры данных отдельных суставов обычно содержат следующую информацию:

- имя сустава в виде строки либо ее хэшированного 32-битного идентификатора;
- индекс родительского сустава в скелете;
- *обратную матрицу преобразования позы привязки* для данного сустава. Поза привязки сустава представляет собой данные о том, какими были положение, ориентация и масштаб этого сустава в момент его связывания с вершинами сетки кожи. По причинам, которые будут подробно рассмотрены в последующих разделах, обычно сохраняется *обратная матрица* этого преобразования.

Типичная структура данных скелета может выглядеть примерно так:

```
struct Joint
{
    Matrix4x3  m_invBindPose; // обратная матрица преобразования позы привязки
    const char* m_name;      // удобное для чтения человеком имя сустава
    U8        m_iParent;    // индекс родительского сустава или 0xFF
                                // в случае корневого сустава
};

struct Skeleton
{
    U32      m_jointCount; // количество суставов
    Joint*   m_aJoint;    // массив суставов
};
```

## 12.3. Позы и положения

Какой бы подход ни использовался для создания анимации (жесткий иерархический или скиновый/скелетный), она всегда воспроизводится постепенно. Чтобы создать иллюзию движения, тело персонажа последовательно принимает статические *позы*, которые быстро чередуются на экране обычно с частотой 30 или 60 *поз в секунду* (на самом деле, как вы увидите в подразделе 12.4.1, вместо строгого отображения каждой позы часто используют *интерполяцию* между соседними позами). В скелетной анимации поза скелета напрямую управляет вершинами меша, а позиционирование является основным инструментом аниматора, с помощью которого тот вдыхает жизнь в своих персонажей. Поэтому очевидно, что прежде, чем мы сможем анимировать скелет, нужно понять, как придать ему *позу*.

Скелет позиционируется за счет произвольного вращения, перемещения и иногда масштабирования его суставов. *Положение* сустава определяется его положением, направлением и масштабом относительно какой-то системы отсчета. Положение сустава обычно представлено в виде матрицы размерностью  $4 \times 4$  или  $4 \times 3$  или же структуры данных SRT (scale, quaternion rotation, vector translation — масштаб, кватернионное вращение, векторное перемещение). Поза скелета — это просто набор положений всех его суставов, который обычно имеет вид массива матриц или структур SRT.

### 12.3.1. Поза привязки

На рис. 12.7 показаны две позы одного и того же скелета. Слева мы видим специальную *позу привязки*, которую иногда называют *позой отсчета* или *позой покоя*. Это состояние трехмерного меша, в котором он находится перед привязкой к скелету (отсюда и название). Иными словами, это поза, которую принял бы обычный, нескиновый треугольный меш без какого-либо скелета. Эту позу также называют *T-образной*, поскольку персонаж напоминает по форме букву «Т» — стоит со слегка расставленными ногами и вытянутыми руками. Эта стойка была выбрана потому, что в ней конечности отдалены от туловища и друг друга, что облегчает процесс привязки вершин к суставам.

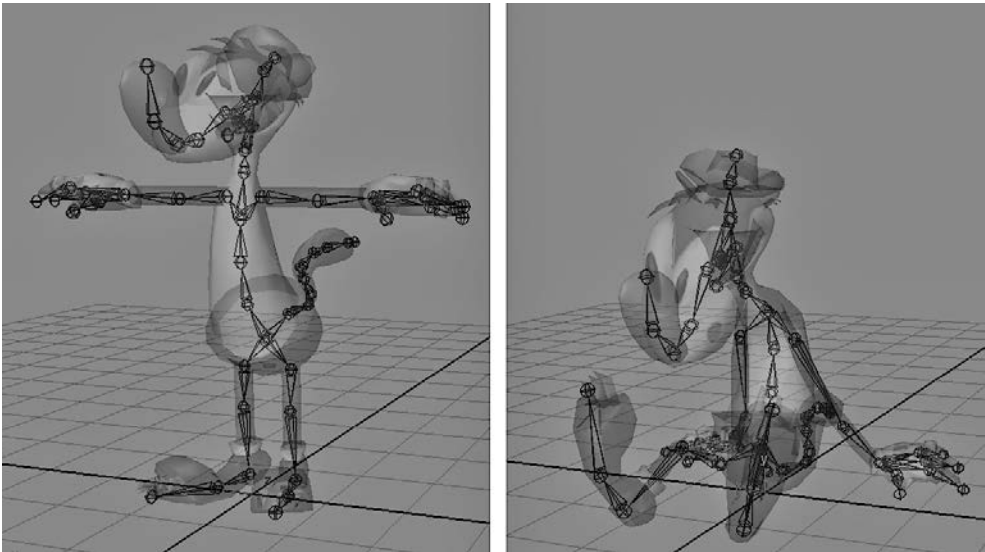


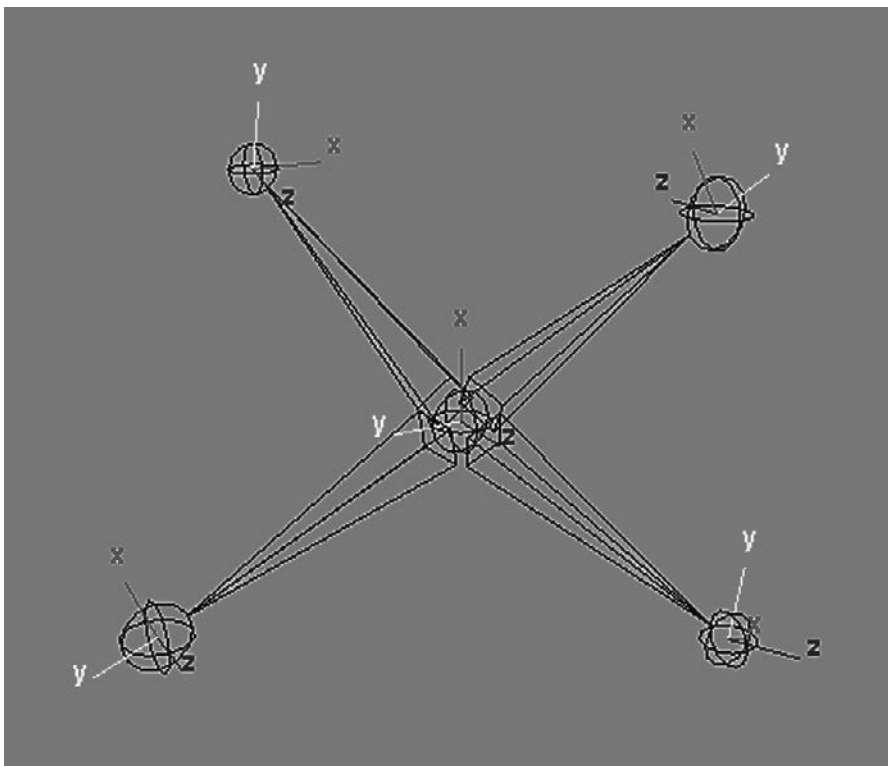
Рис. 12.7. Две позы одного и того же скелета. Слева показана поза привязки

### 12.3.2. Локальные позы

Положения дочерних суставов чаще всего указываются относительно *родительских*. Это обеспечивает естественное движение суставов. Например, если мы повернем плечевой сустав, а положения локтя, запястья и пальцев оставим без изменений, вокруг плеча жестко повернется вся рука, как и ожидалось. Позу или положение, заданные относительно родительского сустава, иногда называют *локальными*. Локальные позы почти всегда хранятся в формате SRT. Причины этого мы исследуем при обсуждении слияния анимации.

Многие пакеты 3D-моделирования, такие как Maya, отображают суставы в виде небольших сфер. Однако, помимо положения, у сустава также есть поворот и масштаб, поэтому его визуализация может быть немного обманчивой. На самом деле

сустав определяет координатное пространство, которое принципиально ничем не отличается от других пространств, рассмотренных ранее, таких как пространство модели, пространство игрового мира или пространство просмотра. Поэтому его лучше представить себе в виде набора осей в прямоугольной системе координат. Мауа дает пользователю возможность отображать локальные оси координат сустава (рис. 12.8).



**Рис. 12.8.** Каждый сустав в скелетной иерархии определяет оси координат в локальном пространстве, известном как пространство сустава

В математическом смысле положение сустава — не что иное, как *аффинное преобразование*. Положение сустава  $j$  можно записать в виде матрицы преобразования  $\mathbf{P}_j$  размером  $4 \times 4$ , которая состоит из вектора преобразования  $\mathbf{T}_j$ , диагональной матрицы масштабирования  $\mathbf{S}_j$  размером  $3 \times 3$  и матрицы вращения  $\mathbf{R}_j$  того же размера. Позу всего скелета  $\mathbf{P}_{\text{skel}}$  можно записать в виде набора всех поз  $\mathbf{P}_j$ , где  $j$  находится в диапазоне от 0 до  $N-1$ :

$$\mathbf{P}_j = \begin{bmatrix} \mathbf{S}_j \mathbf{R}_j & 0 \\ \mathbf{T}_j & 0 \end{bmatrix}; \mathbf{P}_{\text{skel}} = \{\mathbf{P}_j\}_{j=0}^{N-1}.$$