ГЛАВА 29

Виртуальное пространство VRML

Джим О'Доннелл и Марк Р. Браун

В этой главе...

VRML	**2**
Основы синтаксиса VRML	**3**
Простые объекты VRML	**6**
Цвет и текстура объектов VRML	**10**
Весь мир — театр...	**11**
Сотворение мира	**13**
Особенности VRML 2.0	**26**
Ресурсы	**27**

VRML

Язык моделирования виртуальной реальности (Virtual Reality Modeling Language, VRML) представляет собой язык, предназначенный для создания и использования трехмерного интерактивного моделирования.

World Wide Web базируется на использовании HTML, который, в свою очередь, является преемником SGML. Эти языки создавались как инструментарий двумерного форматирования текстов. В мае 1994 года Марк Пеш (Mark D. Pesce), Питер Кеннард (Peter Kennard) и Энтони Париси (Anthony S. Parisi) на первой международной конференции по Web представили статью под названием "Киберпространство" (Cyberspace). В ней излагалась точка зрения на внедрение третьего измерения в Web, что должно послужить совершенствованию ее организации, поскольку человек живет, работает и отдыхает в трехмерном мире. Они назвали свою разработку Virtual Reality Markup Language. Идея была благосклонно принята участниками конференции, и тут же начались работы по поиску формата, который мог бы послужить стандартом данных. Постепенно *M* в VRML изменилось с *Markup* на *Modeling* для отражения различия между текстовой природой Web и миром виртуальной реальности.

Статью "Cyberspace" вы можете найти по адресу:
`http://www.hyperreal.com/~mpesce/www.html`.

В качестве основы для стандарта VRML 1.0 был выбран Open Inventor фирмы Silicon Graphics, который представляет собой набор инструментария разработчика на объектно-ориентированном C++ и используется для быстрой разработки трехмерных графических сред. Open Inventor стал базой для множества стандартов, включая Keystone Interchange Format и метафайлы ANSI/ISO X3H3 3D.

По сравнению с предыдущей версией производительность второй версии VRML увеличена, а также введены дополнительные функции. Спецификация VRML преследует три основные цели:

- независимость от платформы;
- расширяемость;
- возможность работы по низкоскоростным линиям.

Объекты VRML

"Кирпичиками" пространства VRML являются объекты VRML. Язык содержит набор команд, называемых узлами (*nodes*), для построения множества простейших объектов (сфер, параллелепипедов, цилиндров), состоящих из набора вершин и граней.

На заметку

В трехмерном объекте грань (*face*) представляет собой плоскую поверхность, из которых и состоит поверхность объемного тела, а вершина (*vertices*) — точка пересечения нескольких граней. Так, куб полностью определяется восьмью вершинами, в каждой из которых пересекаются три грани.

VRML позволяет создавать из множества простейших объектов более сложные. Это — иерархический язык, в котором дочерние объекты присоединяются к родительским. Так, в модели человеческого тела кисть будет представлять дочерний объект, присоединенный к руке, которая в свою очередь присоединена к торсу. Дальше в этой главе мы поговорим о создании различных объектов VRML.

Пространство VRML

Семейство объектов VRML определяет пространство VRML. В Web можно найти множество примеров таких пространств, использующих парадигму трехмерности для самых различных целей. Для того чтобы определить размещение объектов и связи между ними, необходима возможность определения их относительных размеров и положения в пространстве с помощью координатной системы VRML. Кроме того, VRML позволяет определить источники света и точки наблюдения в пространстве.

Пространство VRML 1.0 было статичным. Движение сводилось к перемещению точек наблюдения, позволяя пользователю совершить путешествие по этому застывшему пространству. Версия 2.0 сделала объекты VRML динамичными.

Объекты VRML 2.0 могут перемещаться, кроме того, возможно использование трехмерного звука, то есть звука, зависящего от положения слушателя по отношению к источнику звука в пространстве VRML. Другое расширение возможностей VRML 2.0 — определение *поведения* объектов. Поведение — описание изменения характеристик объекта в зависимости от его положения относительно других объектов VRML, или от других характеристик пространства, например времени. Скажем, рыбки в аквариуме VRML могут "выпрыгнуть" из него, если вы сделаете его слишком маленьким.

Использование VRML

Как автор Web, вы должны сперва спросить себя, чего вы хотите добиться с помощью VRML. При этом следует помнить о следующем. Во-первых о том, что пространство VRML очень велико — в смысле количества информации, в особенности при использовании сложных объектов с достаточно кривыми поверхностями. Во-вторых, о том, что до сих пор скорость подключения 28,8 Кбит/с остается едва ли не самой распространенной скоростью подключения конечных пользователей, и время передачи информации о вашем пространстве будет слишком велико для большинства посетителей узла.

Более детальную информацию о VRML вы можете почерпнуть из книги издательства Que *Special Edition Using VRML*.

Одно из немногих применений VRML, не требующее загрузки огромных файлов — добавление специальных эффектов в страницы. Вы можете вставить в Web-страницу небольшую VRML-сцену, которая повысит зрелищность страницы, не сильно досаждая пользователям временем загрузки.

В этой главе мы уделим внимание изучению VRML именно в том объеме и с той направленностью, которая, с одной стороны, позволит вам создавать небольшие сцены и, с другой — даст необходимое количество базовых знаний для того, чтобы при желании легко разобраться в построении больших и сложных VRML-пространств.

Основы синтаксиса VRML

VRML-файлы представляют собой обычные текстовые файлы (хотя очень часто для уменьшения времени передачи они сжимаются с помощью программы `gzip` (GNU zip)), которые могут быть созданы при помощи любого текстового редактора. Для создания больших пространств VRML вы можете использовать специализированное программное обеспечение, но использование обычного редактора имеет свои преимущества: благодаря этому вы детально изучите VRML, что очень пригодится в дальнейшем при разработке больших проектов.

В листинге 29.1 вы можете увидеть простейший VRML-файл, отображающий красный конус на белом фоне (рис. 29.1).

Листинг 29.1. Отображение красного конуса на белом фоне

```
#VRML 2.0 utf8

WorldInfo {
    info "Special Edition, Using HTML, 4th Edition"
}

Background {
    skyColor 1 1 1
}

DEF RedCone Shape {
    appearance Appearance {
        material Material {
            diffuseColor 1 0 0 # красный цвет
        }
    }
    geometry Cone { }
}
```

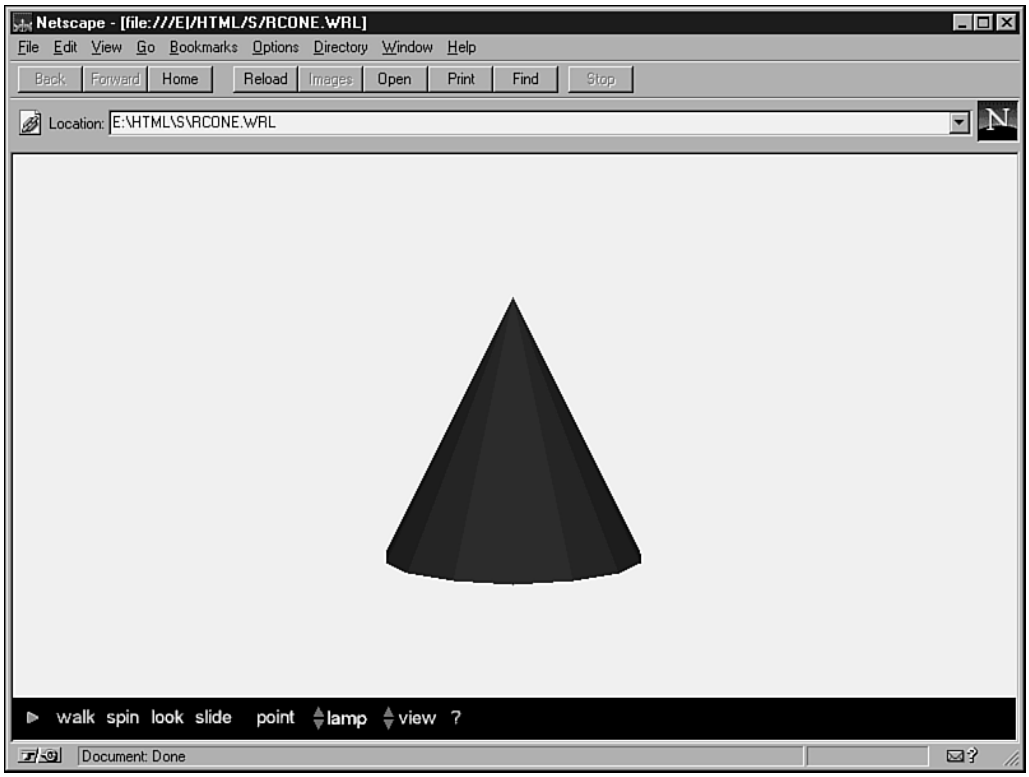

Рис. 29.1. Простой объект VRML можно определить буквально несколькими строками кода

В VRML в качестве символа комментария используется символ **#** — все, что находится в строке после этого символа, представляет собой комментарий. Первая строка файла также содержит комментарий, однако в отличие от прочих этот комментарий выполняет определенные функции и обязательно должен присутствовать в файле. Эта строка определяет, что файл содержит код VRML и указывает его версию. Так, строка

```
#VRML 1.0 ascii
```

указывает, что используется версия 1.0 VRML, и в файле применяется стандартный набор символов ASCII. Строка

```
#VRML 2.0 utf8
```

информирует, что в файле имеется код VRML 2.0, и используется международный набор символов, основанный на ASCII.

Не считая комментариев, содержимое файла может быть представлено как угодно — все символы пробелов, табуляции или перевода строки игнорируются, таким образом, код из листинга 29.1 может быть представлен следующим образом:

```
#VRML 2.0 utf8
WorldInfo {info "Special Edition, Using HTML, 4th Edition"}
Background {skyColor 1 1 1}
DEF RedCone Shape {appearance Appearance {material Material {
diffuseColor 1 0 0 # красный цвет
}} geometry Cone {}}
```

Как и в случае использования любого языка программирования, при написании кода стоит продуманно структурировать и хорошо комментировать текст — это может сэкономить немало времени впоследствии.

Узлы и поля

Если вы хорошо знакомы с C, C++ или Java, то, наверное, обратили внимание на фигурные скобки, определяющие блоки информации. Файлы VRML состоят из *узлов* (*nodes*), которые выглядят следующим образом: `nodeType { fields }`

`nodeType` представляет собой один из поддерживаемых VRML типов узлов. Полное описание поддерживаемых типов вы можете найти по адресу:

`http://www.sdsc.edu/vrml/`.

Конфигурирование с помощью полей

В фигурных скобках расположены *поля* (*fields*). Так, в приведенном ранее примере узел `WorldInfo` имеет поле `info`, а узел `Material` — поле `diffuseColor`. Каждое поле имеет имя и значение; например, для поля `diffuseColor` значение представляет собой набор трех чисел: `1 0 0`.

Значения полей могут быть числами, группами чисел, строками, изображениями, логическими значениями и прочим. Некоторые поля могут иметь несколько значений — в этом случае они разделены запятыми и окружены квадратными скобками. Например, вы можете определить три информационные строки в узле `WorldInfo`:

```
WorldInfo { info [ "First line", "Second line", "Third line" ] }
```

Именование узлов

Узлу может быть присвоено имя, по которому впоследствии можно ссылаться на этот узел. Это делается при помощи префикса `DEF`, как показано ниже (и было сделано в листинге 29.1):

```
DEF RedCone Shape { . . . }
```

Вы можете присвоить узлу любое имя.

Объекты, иерархии, преобразования

Пространство VRML можно представить как иерархию простых объектов VRML. В VRML узел `Transform` используется в качестве контейнера объектов. Не все узлы `Transform` содержат сведения о геометрии (вершины и грани объектов) — они используются для группирования объектов в более сложный объект. Именно таким образом в VRML и определяется иерархия объектов — узел может содержать дочернее поле, перечисляющие присоединенные к нему объекты.

Простые объекты VRML

Спецификация VRML 2.0 включает десять узлов, которые позволяют определять геометрические фигуры. Все объекты VRML 2.0 состоят из одной или нескольких простейших фигур из этого списка. Четыре из упомянутых десяти узлов служат для определения геометрических фигур, один — для определения текста, три — для создания объектов из точек, линий и многоугольников, два являются специализированными узлами для создания рельефа и выпуклых объектов.

Геометрические фигуры

В листинге 29.1 вы уже встречались с одной геометрической фигурой — конусом. VRML позволяет также определить сферу, параллелепипед и цилиндр. Синтаксис всех определений приведен ниже:

- Конус

```
Cone {
    bottomRadius        радиус
    height              высота
    side                TRUE или FALSE
    top                 TRUE или FALSE
}
```

- Параллелепипед

```
Box {
    size    ширина  высота  длина
}
```

- Цилиндр

```
Cylinder {
    radius        радиус
    height        высота
    sides         TRUE или FALSE
```

```
        top             TRUE или FALSE
        bottom          TRUE или FALSE
    }
```

- Сфера

```
Sphere {
    radius      радиус
}
```

Каждый из узлов имеет значения по умолчанию для всех своих полей; так, например, сферу вы можете определить как `Sphere {}`, получив при этом сферу единичного радиуса.

На заметку При создании сферы, конуса и цилиндра создается впечатление, что VRML позволяет определять кривые поверхности, однако на самом деле это не так: VRML создает эти объекты из множества вершин и небольших граней, что в конечном счете приводит к иллюзии гладкого объекта. Убедиться в этом можно, просмотрев объект в VRML-броузере.

Текст

Узел `Text` позволяет создать текстовый объект в VRML. Создаваемый текст — плоский объект, а потому с определенных точек (с торца) он невидим. Узел содержит четыре поля и имеет следующий синтаксис:

```
Text {
    string          ["строка(и)",...]
    fontStyle       определение шрифта, выравнивания, интервала
    maxExtent       максимальный размер в любом направлении
    legnth          длина текстовой строки в поле string
}
```

В листинге 29.2 приведен пример использования узла `Text`, результат вы можете увидеть на рис. 29.2.

Листинг 29.2. Двумерный текст VRML

```
#VRML V2.0 utf8

WorldInfo {
    info "Special Edition, Using HTML, 4th Edition"
}

Background {
    skyColor  1  1  1
}

Shape {
    appearance Appearance {
        material Material {
            diffuseColor  0  0  0
        }
    }
    geometry Text {
        string [ "This Is", "A Test", "Of Ascii Text" ]
        fontStyle FontStyle { spacing 2 justify "MIDDLE" }
    }
}
```

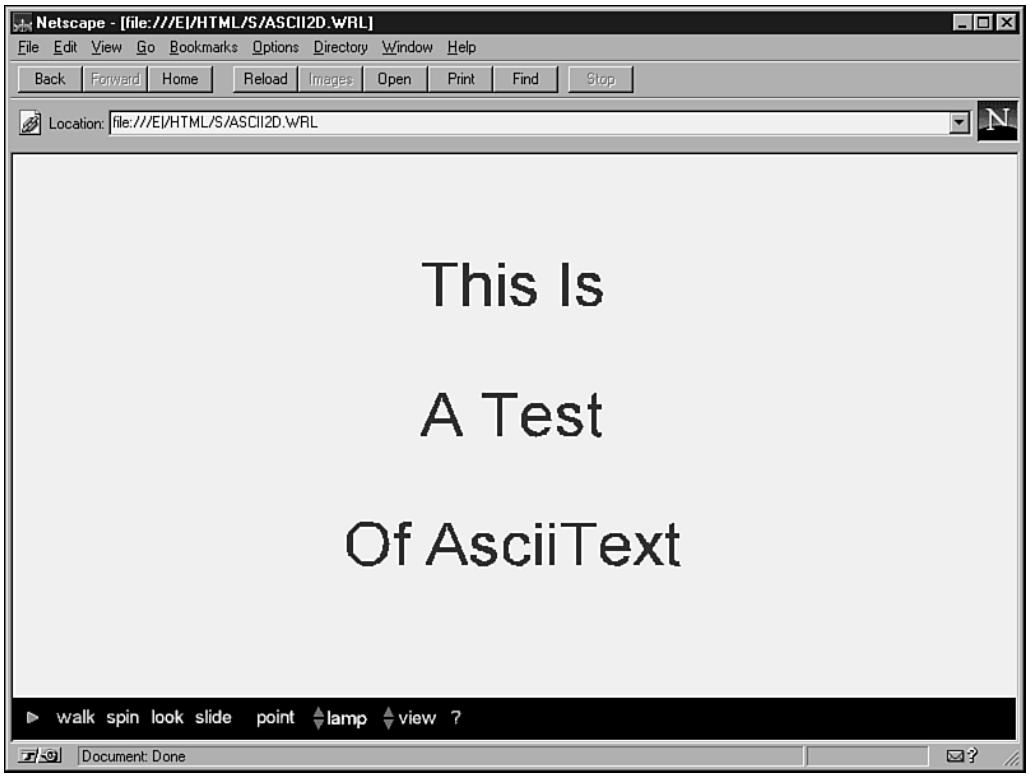

Рис. 29.2. Текст VRML двумерен; при повороте на 90 градусов он становится невидимым

Фигуры VRML

Обсуждавшиеся узлы используются в качестве отправной точки для изучения; однако в большинстве реальных VRML-файлов вы найдете множество других узлов — `IndexedFaceSet`. Такие узлы служат для создания объектов из наборов вершин и граней. В листинге 29.3 приведен пример создания пирамиды, которая показана на рис. 29.3.

Листинг 29.3. Построение фигур с помощью *IndexedFaceSet*

```
#VRML V2.0 utf8

WorldInfo {
    info "Special Edition, Using HTML, 4th Edition"
}

Background {
    skyColor  1  1  1
}

DEF Pyramid Shape {
    appearance Appearance {
        material Material {
            diffuseColor  0  0  1
        }
    }
```

```
    geometry IndexedFaceSet {
        coord Coordinate {
            point [ -1  0 -1,
                     1  0 -1,
                     1  0  1,
                    -1  0  1,
                     0  2  0 ]
        }
        coordIndex [ 0, 4, 1, -1,
                     1, 4, 2, -1,
                     2, 4, 3, -1,
                     3, 4, 0, -1,
                     0, 1, 2, 3, -1 ]
    }
}
```

Массив координат вершин определен с использованием узла `Coordinate` и его собственного поля `point`. Это поле получает множество значений, каждое из которых представляет триаду чисел (`X`, `Y` и `Z`) — координат вершины. Вершин может быть определено любое количество. Обратите внимание, что каждая вершина определяется только один раз, независимо от количества пересекающихся в ней граней.

Узел `IndexedFaceSet` содержит поле с именем `coordIndex`, которое содержит список граней, определяемых их вершинами. Так, набор чисел `0, 4, 1, -1` определяет грань между нулевой, четвертой и первой вершинами (нумерация вершин начинается с нуля). Значение `-1` указывает, что определение грани завершено и следующее значение — первая вершина следующей грани.

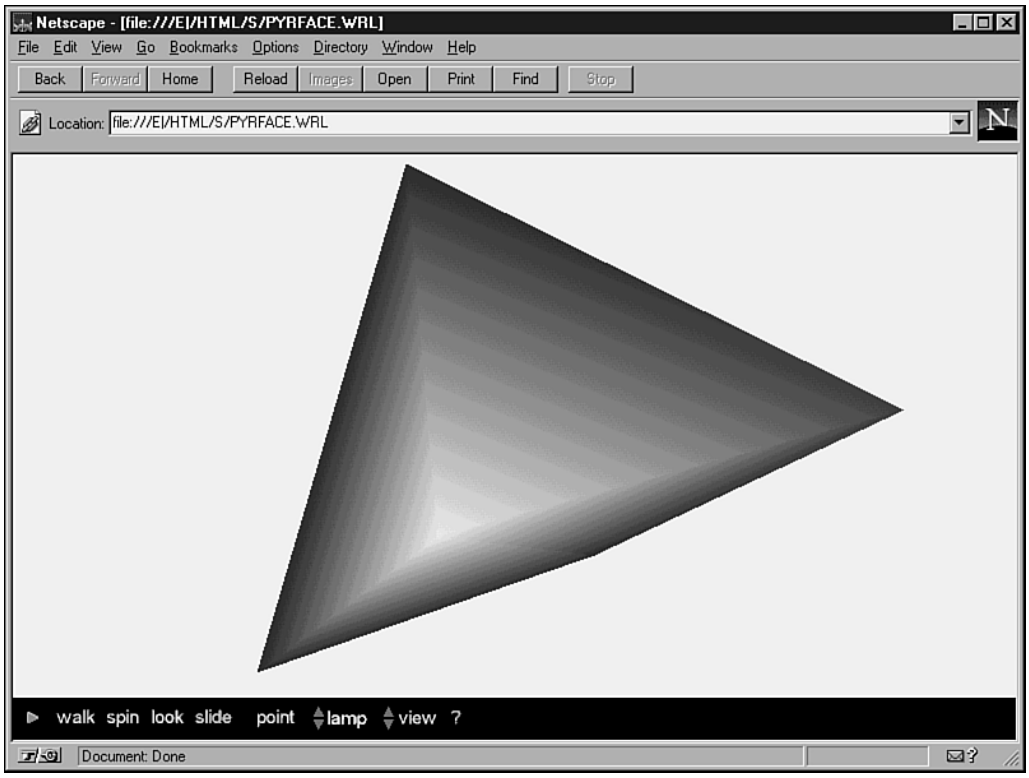

Рис. 29.3. Узел `IndexedFaceSet` используется для создания сложных геометрических фигур

На заметку

Хотя `IndexedFaceSet` — очень мощное средство, при его использовании необходимо соблюдать определенные правила.

Убедитесь, что все вершины одной грани лежат в одной плоскости, иначе вместо грани вы можете получить нечто странное (вспомните, впрочем, чему вас учили в школе — намекнем, в частности, что любые три точки, не лежащие на одной прямой, однозначно определяют плоскость... — *Прим. ред.*).

Избегайте пересечения плоскостей. Если какие-либо две грани объекта пересекаются, не имея общих вершин, — это может вызвать проблемы у броузера VRML.

И напоследок, учтите, что рендеринг большого количества объектов — весьма длительная процедура...

VRML предоставляет еще два узла, очень похожих на только что рассмотренный — `IndexedLineSet` и `IndexedPointSet`. Они используются так же, как и узел `IndexedFaceSet`, с тем отличием, что `IndexedLineSet` создает набор линий между вершинами, а `IndexedPointSet` — поле точек.

Цвет и текстура объектов VRML

VRML предлагает разные пути для изменения внешнего вида объектов. Один из них — узел `Material` — уже использовался в приведенных ранее примерах. Для определения цвета, текстуры и внешнего вида объектов в целом VRML предлагает узлы `Material` и `ImageTexture`.

Material

Наиболее общеупотребительное применение этого узла показано в листинге 29.1: для определения цвета объекта используется поле `diffuseColor`, хотя узел может использовать и другие поля для получения различных эффектов.

ImageTexture

Использование этого узла дает возможность получения большого разнообразия эффектов. В частности, этот узел позволяет отобразить на объект файл с изображением — пример такого отображения показан на рис. 29.4, а в листинге 29.4 приведен соответствующий код.

Листинг 29.4. Отображение изображения на объект VRML

```
#VRML V2.0 utf8

WorldInfo {
    info "Special Edition, Using HTML, 4th Edition"
}

Background {
    skyColor  1  1  1
}

DEF Pyramid Shape {
    appearance Appearance {
        material Material {
            diffuseColor  0  0  1
        }
```

```
        texture ImageTexture {
            url "bernie.gif"
        }
    }
    geometry Box {}
}
```

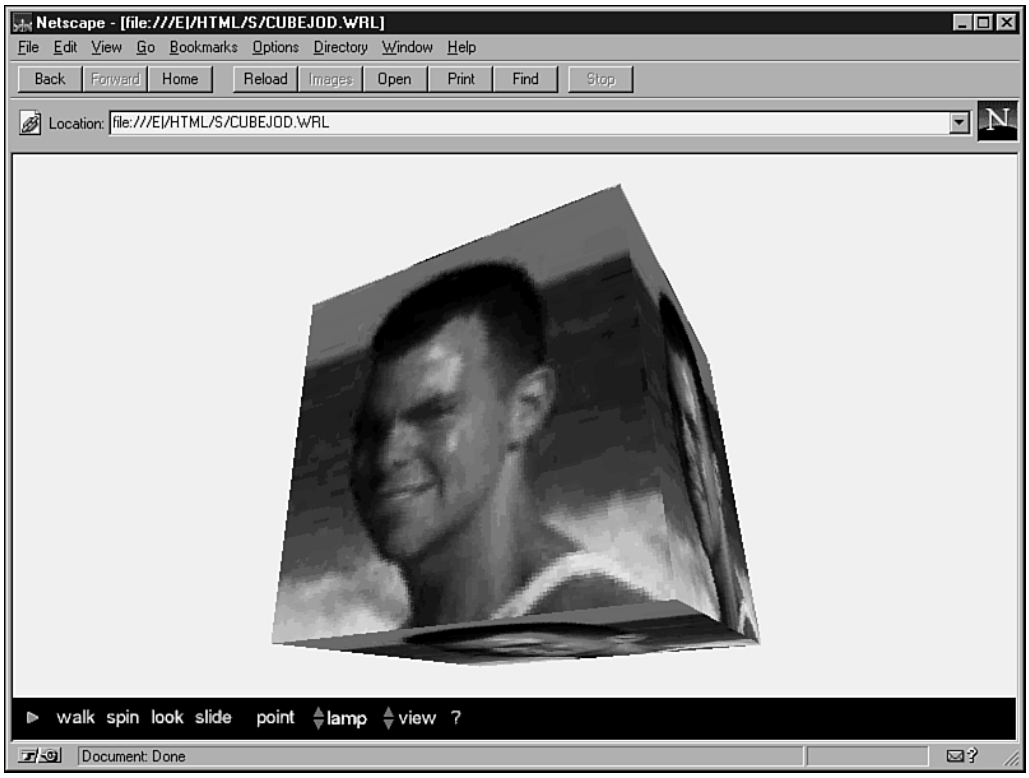

Рис. 29.4. Отображение изображения на поверхность объекта VRML

В приведенном примере грани объекта плоские, поэтому искажения изображения невелики. Однако в случае кривых поверхностей — например, сферы или конуса, с искажениями приходится считаться. Какой формат графического файла может быть использован в качестве текстуры — определяется возможностями VRML-броузера.

Весь мир — театр...

До сих пор мы говорили о пространстве VRML как о файлах, которые создаются отдельно от других приложений и данных и просматриваются в VRML-броузере — без какого-либо внедрения в Web-страницу.

Следует попутно заметить, что, поскольку не существует каких-либо ограничений на размеры и сложность VRML-пространства ни сверху, ни снизу, то обычно это достаточно большие пространства. Трехмерное VRML-пространство зачастую используется для того, чтобы “провести” пользователя сквозь такое пространство, продемонстрировав все, что может его заинтересовать.

Позвольте порекомендовать вам ознакомиться с примером удачного использования VRML на узле университета Эссекса, Великобритания (**`http://esewww.essex.ac.uk/campus-model.wrl`**), который представляет собой трехмерную модель студенческого городка университета. Посетитель может совершить путешествие по виртуальному городку, осматривая его с разных сторон, и даже воспользоваться ссылкой на страницу с информацией, имеющейся для каждого здания.

Подобные сцены VRML отлично уживаются на Web-страницах, придавая им особую красочность и эффектность. Используя небольшие специализированные сцены, вы можете добиться множества различных эффектов, особенно при помощи VRML 2.0 с его анимационными расширениями. Такие сцены в чем-то сходны с анимированным `GIF`, однако, как правило, соответствующий VRML-файл существенно меньше по размеру.

В начале было Слово

Если вы пришли к выводу о необходимости использования VRML — вам стоит задуматься не только о сугубо технических вопросах, но и о сопутствующих факторах. Важно решить, какие объекты будут присутствовать в сцене и как они будут взаимодействовать. Не менее важно принять решение относительно других вопросов, первоначально не представлявшихся важными. Насколько большое пространство вы намерены создать? Насколько оно должно быть детализировано? Какой сложности фигуры должны в нем присутствовать? Как они должны быть расположены, и как вы намерены создать их?

Размеры и детализация

Первое, о чем вам следует задуматься, — размеры. Размеры не в смысле объема вашего VRML-пространства, а размеры создаваемых файлов. Ведь может случиться, что только отдельные пользователи со сверхскоростными каналами связи смогут оценить вашу фантазию. Если ваше пространство настолько сложно, что рендеринг будет требовать, как минимум, двухпроцессорного Pentium с 256 Мбайт памяти — вряд ли его сможет увидеть много людей.

Старайтесь исходить из того, что основная масса посетителей вашего узла имеет скорость подключения 28,8 Кбит/с (или даже 14,4 Кбит/с) и компьютер класса 486-133 с 16 Мбайт оперативной памяти. Исходя из этого, вам придется выбирать размеры и степень детализации вашего VRML-пространства. Как и везде, вам придется искать компромисс. Вы можете поместить много объектов, но тогда придется пожертвовать их детализацией. При детальной прорисовке — объектов должно быть немного. И, как и в любом другом поиске компромисса, это, скорее, искусство, поэтому определить четкие правила поиска такого компромисса просто невозможно.

Прежде чем приступать к разработке, следует четко определить цель создания VRML-пространства, так как именно цель во многом предопределяет результат компромисса. Если ваша цель — продать нечто заглянувшему на вашу страницу, то наилучшее решение — пространство, состоящее только из одного продаваемого объекта, но зато во всех деталях. Если же вы намерены совершить экскурсию в Египет, то вряд ли стоит прорисовывать каждый блок пирамиды Тутанхамона. Когда покупатель прогуливается по вашему магазину, не стоит прорисовывать каждый предмет на прилавке, но вот если он заинтересовался чем-то — покажите это отдельно и во всех деталях.

Конструирование и макетирование

Итак, вы решили, насколько большое пространство вы хотите создать, — теперь надо подумать, как оно должно выглядеть. Ведь зачастую неважно, хорошо или плохо что-то работает — важно, как убеждает нас пример Microsoft, насколько хорошо это выглядит. А так как ваш мир, в отличие от нашего реального Мира отнюдь не единственный, то от его внешнего вида будет зависеть количество посетителей. Вам также надо продумать вопросы навигации по вашему пространству, прикинуть, как лучше подать его зрителю — издалека или с близкого расстояния.

И вновь — ответы на все эти вопросы в первую очередь зависят от ответа на основной вопрос — зачем? Какова цель создания вашего пространства? Если вы проводите зрителя по четко очерченной дороге — так сказать, показываете мир из окна экскурсионного автобуса — то вы можете смело забыть об обратной стороне вашего пространства. Если же вы пускаете зрителя в самостоятельную прогулку по своему пространству, вам следует принять меры, чтобы он не почувствовал особой разницы между турпоездкой и эмиграцией, то есть проработать все, что он может увидеть — а в последнем случае таких мест гораздо больше. Ведь если вы моделируете поездку на экскурсионном автобусе, например по Крещатику, нет смысла тщательно воссоздавать Прорезную, куда зритель при всем желании заглянуть не сможет...

Сотворение мира

Теперь мы пройдем этапы сотворения VRML-мира, в нашем случае — очень простого. Мы пройдем этапы макетирования, создания объектов, размещения их в нашем пространстве. Мы добавим немного реалистичности при помощи текстур и источников освещения и закончим связью нашего мира с большим миром Web.

И хотя создание такого пространства можно упростить использованием соответствующих программных средств, мы не будем искать легких путей и создадим его своими руками. Этот путь нелегок, но зато он даст вам бесценный опыт, который стоит трудов так же, как овчинка — выделки, игра — свеч, а Париж — мессы.

Когда вы приступите к сотворению своего пространства — посетите **http://www.yahoo.com/Computers_and_Internet/Internet/World_Wide_Web/Virtual_Reality_Modeling_Language_VRML_/Authoring/** и поинтересуйтесь инструментарием, который сможет облегчить вашу работу.

День первый: план

Первый шаг творения состоит в том, что вы должны представить, как будет выглядеть ваш мир. Самый главный инструмент на этом этапе — ручка, а также лист чертежной бумаги. Прикиньте вид сверху и вид сбоку вашего пространства, и попробуйте ответить на следующие вопросы.

- Какие простые объекты VRML вам понадобятся?
- Какие составные объекты необходимы, и можно ли (и как) создать их из простых объектов?
- Какое пространство требуется для вашего мира?
- Где должны располагаться объекты?
- Куда лучше всего поместить источники освещения и камеры?

Координатная система VRML

На этом этапе начинается привязка вашего пространства к координатной системе. Вам надо преобразовывать объекты в сухой язык координат — вот почему, кстати, использование при разработке чертежной бумаги и основных проекций — неплохая идея. Ввиду важности системы координат поговорим о ней немного подробнее.

В VRML используется общепринятая декартова система координат, которая уже несколько столетий служит для описания двух- и трехмерных объектов. По умолчанию считается, что ось `X` направлена слева направо, ось `Y` — снизу вверх, а ось `Z` — на зрителя (стандартная правосторонняя система координат — этот вопрос будет важен при рассмотрении вращений. Пока лишь заметим, что в этой системе координат положительное вращение вокруг оси определяется *правилом правой руки*). На рис. 29.5 показана используемая в VRML система координат.

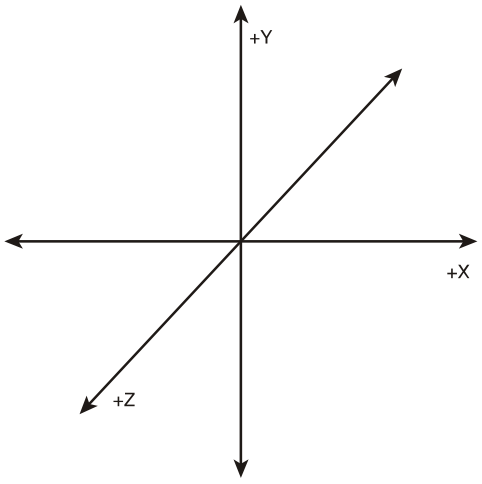

Рис. 29.5. В VRML используется правосторонняя декартова система координат

Векторы

Вершиной (*vertex*) называется точка, которая в декартовой системе координат описывается тремя числами — `x`, `y`, `z`. *Вектор* (*vector*) также представляется тройкой чисел. Отличие вершины и вектора в том, что если вершина представляет собой точку в пространстве, то вектор представляет направление, в котором движется точка от начала координат к вершине. (Позвольте порекомендовать читателю освежить свою память и вспомнить, по крайней мере, школьный курс математики. К сожалению, в последнее время волна некомпетентности захлестнула и программирование, и образ недоучки, считающего себя крутым хакером, становится скорее правилом, чем исключением. Тем не менее для сколь-нибудь серьезной работы необходимо твердое знание математики хотя бы в пределах школьной программы... — *Прим. ред.*).

Единицы VRML

Единицей измерения длины в VRML служит *метр*, а единицей измерения угловых величин — *радиан*.

На заметку Напомним, что 2π радиан равны 360°.

День второй: чертеж

Теперь пришло время поместить ваш мир (вернее, его изображение) на бумагу и определить основные координаты, размеры и расположение объектов в нем.

На рис. 29.6 и 29.7 показаны вид сверху и сбоку — по осям `Y` и `Z`. Естественно, что вы можете использовать и другие виды пространства — те, которые, по вашему мнению, помогут в его создании и описании.

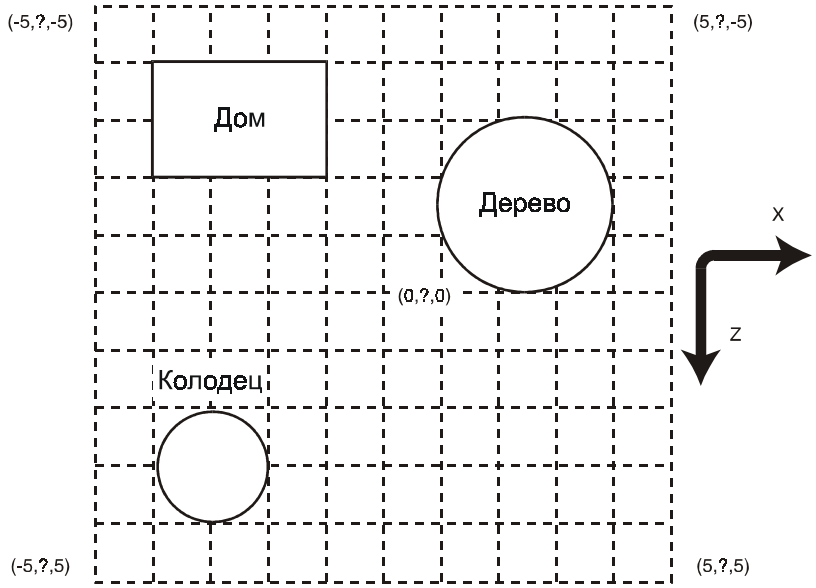

Рис. 29.6. Вид вашего VRML-пространства сверху

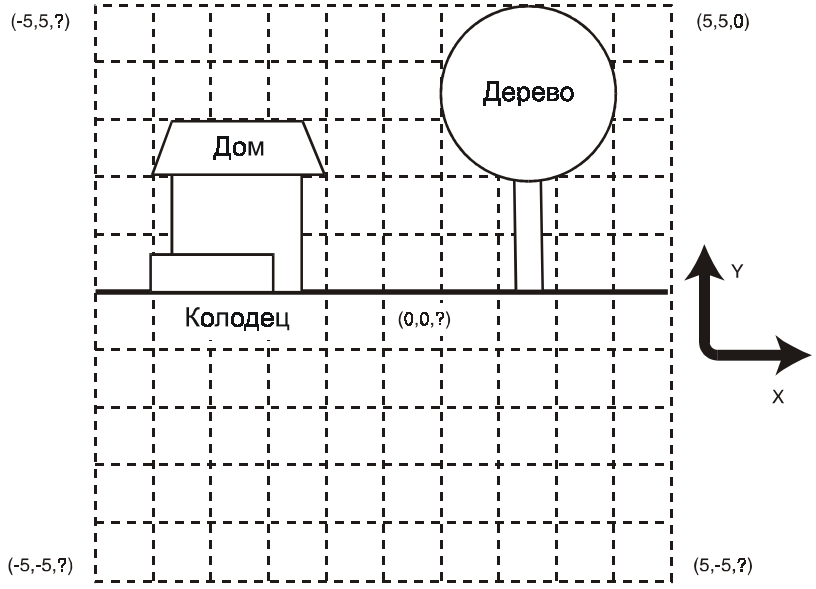

Рис. 29.7. Вид вашего VRML-пространства сбоку

Теперь, когда сделан набросок создаваемого пространства, вернемся немного назад и попробуем с помощью этого наброска ответить на следующие вопросы.

- **Какие простейшие VRML-объекты будут необходимы?**

  Нашему пространству понадобятся: плоскость — для земли, куб и индексированный объект — для дома, цилиндр и сфера — для дерева и один или несколько цилиндров — для колодца.

- **Какие составные объекты потребуются и как их создать из простейших?**

  В нашем случае такими объектами являются дом и его крыша, дерево и его ствол, и части колодца.

- **Каков объем создаваемого пространства?**

  Мы удовлетворимся десятиметровым кубом.

- **Где должны быть размещены объекты?**

  Координаты объектов показаны на набросках VRML-пространства.

День третий: перемещение объектов

Создание необходимых для нашего мира объектов не должно вызвать затруднений. Однако требуется возможность перемещения их в создаваемом пространстве. Без такой возможности все объекты будут представлять собой кучу малу в начале координат. Необходимое перемещение осуществляется с использованием узла `Transform` со следующим синтаксисом:

```
Transform {
    translation        x   y   z
    rotation          rx  ry  rz  ra
    scale              x   y   z
    scaleOrientation  rx  ry  rz  ra
    center             x   y   z
    children           [    ]
}
```

Узел `Transform` используется для перемещения, масштабирования и вращения всех объектов в его поле `children`. В листинге 29.5 представлено начало создания пространства — земля и дом после его перемещения в нужное место (рис. 29.8).

Листинг 29.5. Использование узла *Transform* для перемещения объектов

```
#VRML V2.0 utf8

WorldInfo {
    info "Special Edition, Using HTML, 4th Edition"
}

Background {
    skyColor  1  1  1
}
Shape {
    appearance Appearance {
        material Material {
            diffuseColor  0  0.75  0
        }
    }
    geometry IndexedFaceSet {
```

```
        coord Coordinate {
            point [  -5   0   5,
                      5   0   5,
                      5   0  -5,
                     -5   09  -5 ]
        }
        coordIndex [0, 1, 2, 3, -1]
    }
}

Transform {
    translation   -2.5   1   -3
    children [
        Shape {
            appearance Appearance {
                material Material {
                    diffuseColor  0  0  0
                }
            }
            geometry Box { size  2.3333  2  1.3333 }
        }
    ]
}
```

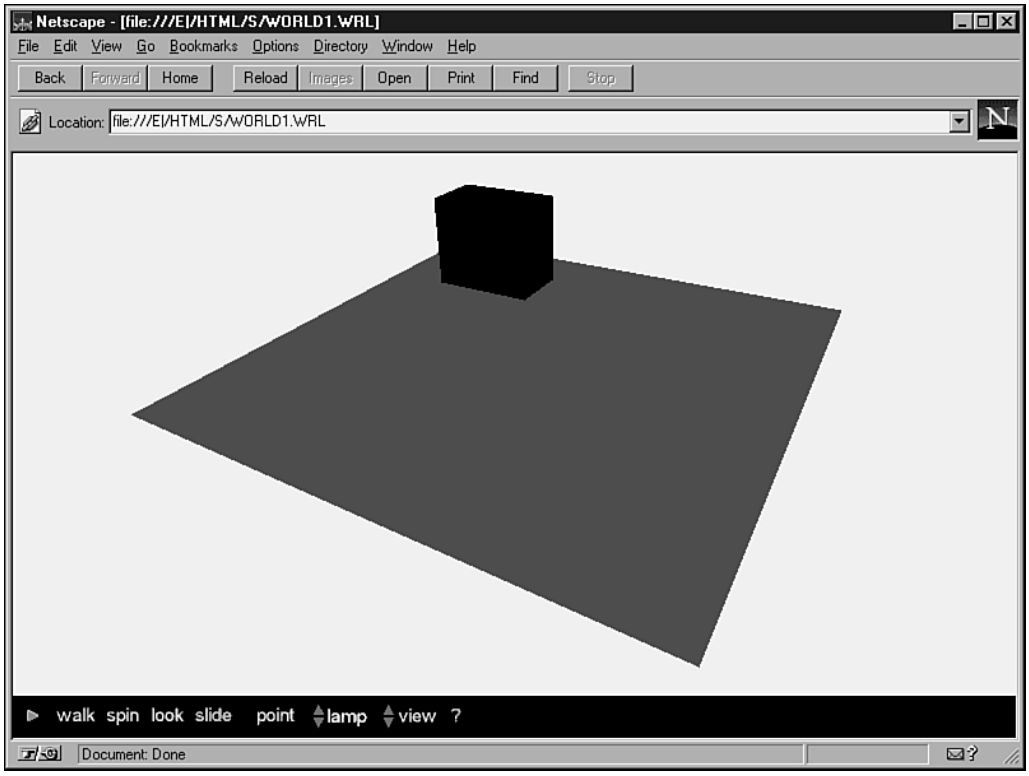

Рис. 29.8. Узел `Transform` позволяет перемещать, масштабировать и вращать как простые, так и составные объекты VRML

День четвертый: создание иерархии объектов

В более сложных пространствах — да и в нашем, конечно, — можно синхронно перемещать простейшие объекты, однако гораздо проще и надежнее сперва создать более сложные объекты из простейших и все операции производить уже с ними. Узел VRML `Transform` используется для создания составных объектов включением в него других узлов. Иерархия создается определением простейших объектов и их положений по отношению друг к другу в узле `Transform`. После этого со всей иерархией можно работать как с отдельным объектом.

В листинге 29.6 показаны очередные добавления в создаваемое VRML-пространство, заключающиеся в постройке крыши дома. Обратите внимание, что крыша создается при помощи `IndexedFaceSet` и позиционируется с использованием узла `Transform`. Затем составной объект, представляющий дом, перемещается в нужное положение (рис. 29.9).

Листинг 29.6. Создание составного объекта при помощи узла *Transform*

```
#VRML V2.0 utf8

WorldInfo {
    info "Special Edition, Using HTML, 4th Edition"
}

Background {
    skyColor  1  1  1
}

#
#  Земля
#

Shape {
    appearance Appearance {
        material Material {
            diffuseColor  0  0.75  0
        }
    }
    geometry IndexedFaceSet {
        coord Coordinate {
            point [  -5   0   5,
                      5   0   5,
                      5   0  -5,
                     -5   09  -5 ]
        }
        coordIndex [0, 1, 2, 3, -1]
    }
}

#
# Дом
#

Transform {
    translation   -2.5   1   -3
    children [
        Shape {
```

```
            appearance Appearance {
                material Material {
                    diffuseColor  0  0  0
                }
            }
            geometry Box { size  2.3333  2  1.3333 }
        }
        Transform {
            translation  0  1  0
            children [
                Shape {
                    appearance Appearance {
                        material Material {
                            diffuseColor  0  0  1
                        }
                    }
                    geometry IndexedFaceSet {
                        solid  FALSE
                        coord Coordinate {
                            point [
                                -1.5     0   1,
                                -1.1667  1  -0.6667,
                                 1.5     0  -1,
                                 1.1667  1  -0.6667,
                                 1.5     0   1,
                                 1.1667  1   0.6667,
                                -1.5     0   1,
                                -1.1667  1   0.6667
                            ]
                        }
                        coordIndex [
                            0,  2,  4,  6,  -1,
                            1,  3,  5,  7,  -1,
                            0,  2,  3,  1,  -1,
                            2,  4,  5,  3,  -1,
                            4,  6,  7,  5,  -1,
                            6,  0,  1,  7,  -1
                        ]
                    }
                }
            ]
        }
    ]
}
```

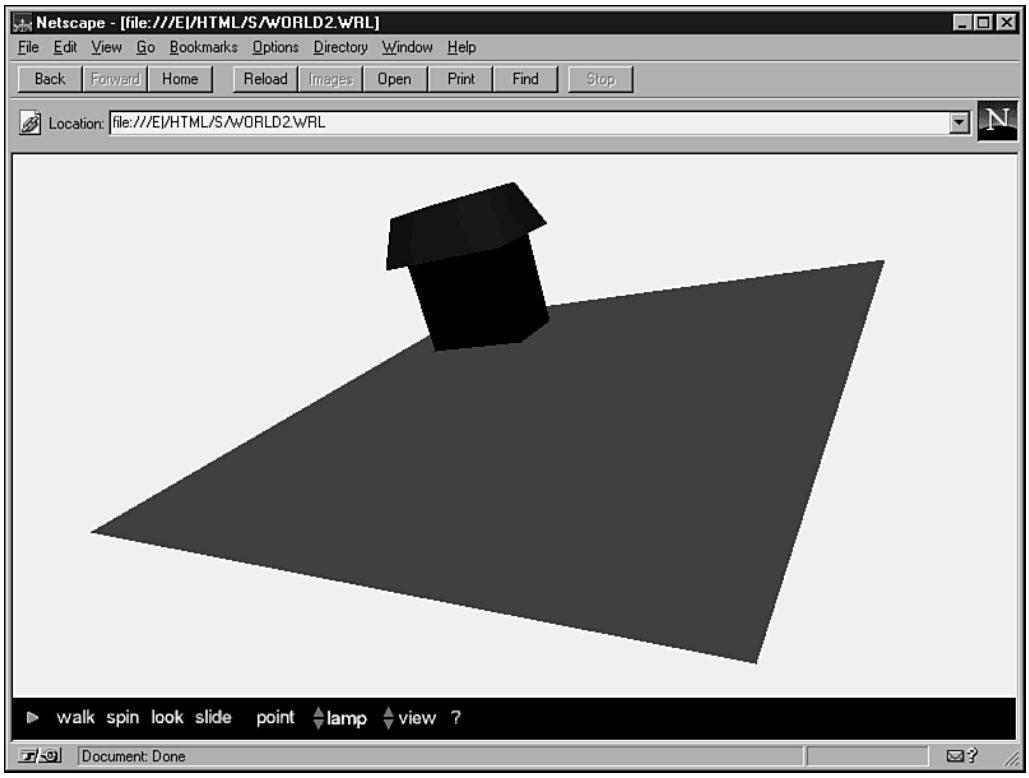

Рис. 29.9. Создание составного объекта упрощает создание и управление сложными сценами

Теперь, когда мы разобрались, как создавать составные объекты, добавим все недостающее в наше VRML-пространство — дерево, состоящее из цилиндра и сферы, и колодец, состоящий из двух цилиндров. В листинге 29.7 приведен окончательный текст, а на рис. 29.10 — то, что получилось.

Листинг 29.7. Законченный вариант VRML мира

```
#VRML V2.0 utf8

WorldInfo {
    info "Special Edition, Using HTML, 4th Edition"
}

Background {
    skyColor  1  1  1
}

#
#  Земля
#

Shape {
    appearance Appearance {
        material Material {
            diffuseColor  0  0.75  0
        }
```

```
    }
    geometry IndexedFaceSet {
        coord Coordinate {
            point [  -5   0   5,
                      5   0   5,
                      5   0  -5,
                     -5   09  -5 ]
        }
        coordIndex [0, 1, 2, 3, -1]
    }
}

#
# Дом
#

Transform {
    translation   -2.5   1   -3
    children [
        Shape {
            appearance Appearance {
                material Material {
                    diffuseColor  0  0  0
                }
            }
            geometry Box { size  2.3333  2  1.3333 }
        }
        Transform {
            translation  0  1  0
            children [
                Shape {
                    appearance Appearance {
                        material Material {
                            diffuseColor  0  0  1
                        }
                    }
                    geometry IndexedFaceSet {
                        solid  FALSE
                        coord Coordinate {
                            point [
                                -1.5     0   1,
                                -1.1667  1  -0.6667,
                                 1.5     0  -1,
                                 1.1667  1  -0.6667,
                                 1.5     0   1,
                                 1.1667  1   0.6667,
                                -1.5     0   1,
                                -1.1667  1   0.6667
                            ]
                        }
                        coordIndex [
                            0,  2,  4,  6,  -1,
                            1,  3,  5,  7,  -1,
                            0,  2,  3,  1,  -1,
                            2,  4,  5,  3,  -1,
                            4,  6,  7,  5,  -1,
                            6,  0,  1,  7,  -1
                        ]
                    }
```

```
                }
            ]
        }
    ]
}

#
# Дерево
#

Transform {
    translation  2.5  1.5  -1.5
    children [
        Shape {
            appearance Appearance {
                material Material {
                    diffuseColor  0.5  0.25  0
                }
            }
            geometry Cylinder {
                radius  0.1667
                height  3
            }
        }
        Transform {
            translation  0  1.5  0
            children [
                Shape {
                    appearance Appearance {
                        material Material {
                            diffuseColor  0  1  0
                        }
                    }
                    geometry Sphere {
                        radius  1.5
                    }
                }
            ]
        }
    ]
}

#
# Колодец
#

Transform {
    translation  -3  0.3333  3
    children [
        Shape {
            appearance Appearance {
                material Material {
                    diffuseColor  1  1  1
                }
            }
            geometry Cylinder {
                radius  1
                height  0.6667
            }
```

```
        }
        Shape {
            appearance Appearance {
                material Material {
                    diffuseColor  0  0  0
                }
            }
            geometry Cylinder {
                radius  0.8
                height  0.7
            }
        }
    ]
}
```

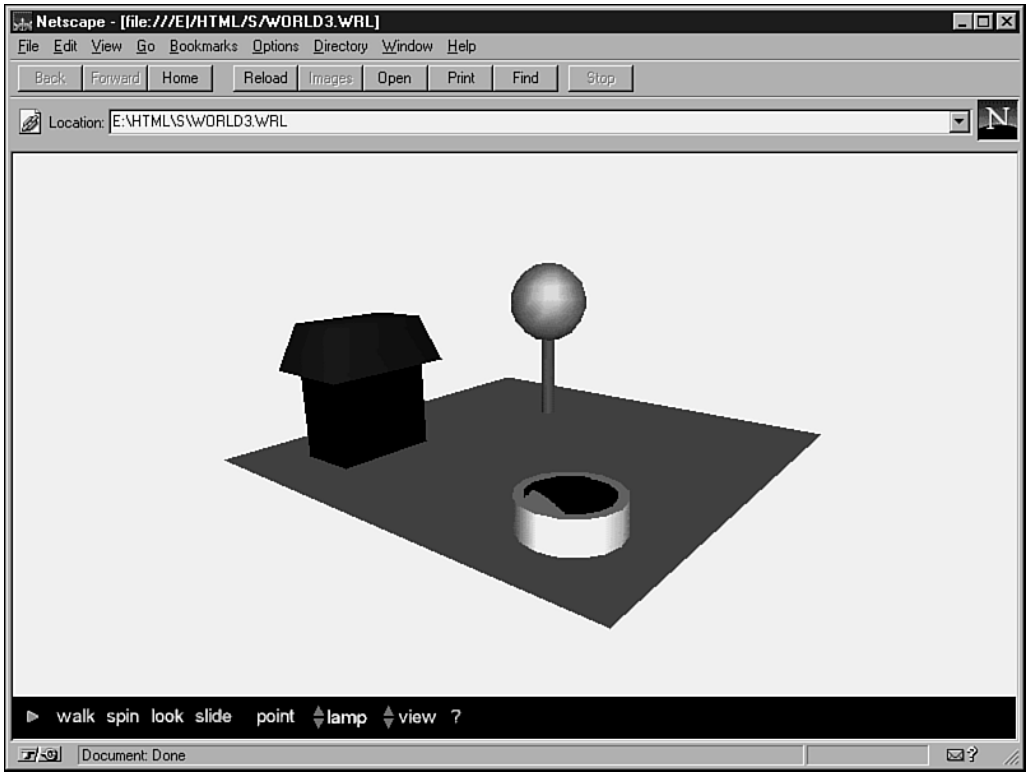

Рис. 29.10. Все объекты помещены в VRML-пространство

День пятый: придание реалистичности

Ранее мы упоминали об использовании узла `ImageTexture` для придания реалистичности объектам путем назначения текстур его поверхностям. По умолчанию VRML отображает изображение, определенное узлом, на каждую поверхность объекта. При отображении текстуры на поверхность для достижения большей реалистичности лучше использовать маленькие изображения, располагая их на поверхности в виде мозаики. Этого можно достичь при помощи узла `TextureTransform` с синтаксисом

```
TextureTransform {
    translation        x    y
    rotation           angle
    scale              x    y
    center             x    y
}
```

Эти поля позволяют перемещать, вращать, масштабировать и центровать изображения. Реалистичность мозаичных текстур достигается благодаря полю `scale`, которое определяет, сколько раз повторяется изображение. Обратите внимание на следующее: чем больше значение этого поля, тем меньше размер изображения и тем большее количество раз оно повторяется.

В листинге 29.8 приведен пример наложения текстуры на один из объектов создаваемого пространства — с результатом вы можете ознакомиться на рис. 29.11.

Листинг 29.8. Использование текстур

```
#
# Колодец
#

Transform {
    translation  -3  0.3333  3
    children [
        Shape {
            appearance Appearance {
                material Material {
                    diffuseColor  1  1  1
                }
                texture ImageTexture {
                    url "bernie.gif"
                }
                textureTransform TextureTransform {
                    scale  8  1
                }
            }
            geometry Cylinder {
                radius  1
                height  0.6667
            }
        }
        Shape {
            appearance Appearance {
                material Material {
                    diffuseColor  0  0  0
                }
            }
            geometry Cylinder {
                radius  0.8
                height  0.7
            }
        }
    ]
}
```

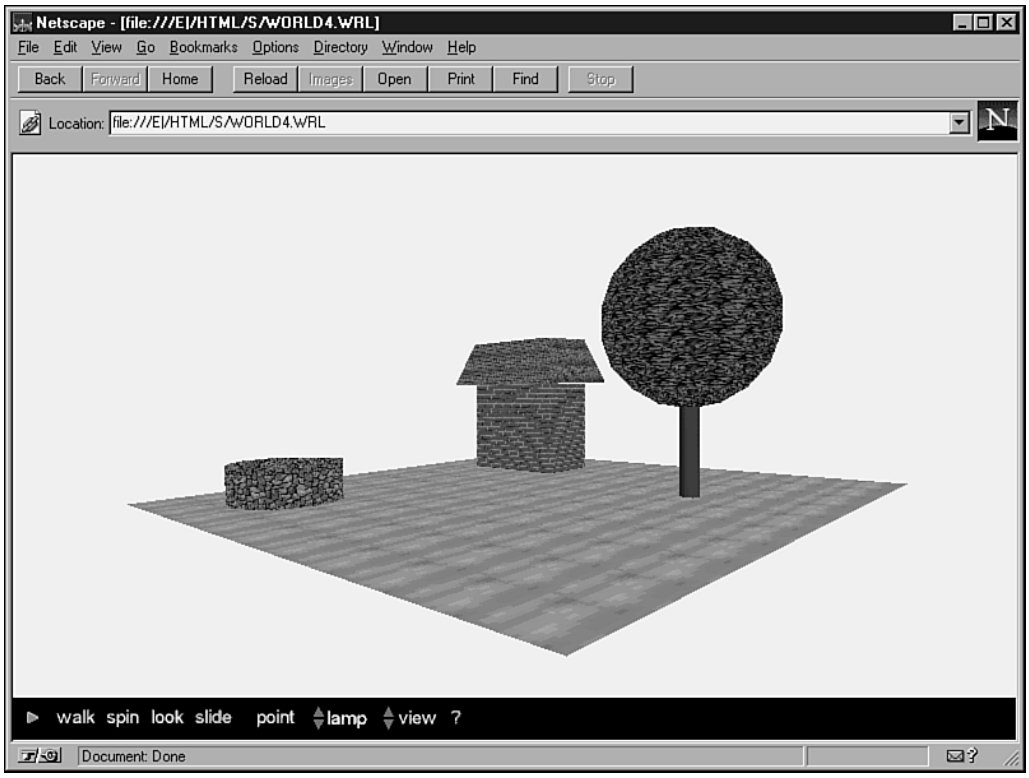

Рис. 29.11. Использование текстур придает большую реалистичность создаваемому пространству, однако при этом замедляется передача и рендеринг, поэтому используйте текстуры только при необходимости

День шестой: связь с Web

VRML, как и HTML, создавался для использования в Internet и Web. Основной элемент такого использования — гиперсвязи. Они, в частности, позволяют связать ваше пространство с другими VRML пространствами.

Inline

Один из вариантов связи VRML с Web — использование узла `Inline`, который позволяет включать в VRML-пространство объекты VRML из любого локального файла или любого VRML-пространства в Web. Синтаксис такого использования следующий:

```
WWWInline {
    url "путь к локальному файлу или URL"
}
```

В этом случае код, размещенный в указанном файле, обрабатывается так, как если бы он был размещен в соответствующем месте вашего файла.

Anchor

Гиперсвязи реализованы в VRML с использованием узлов `Anchor`. Поле `url` определяет гиперсвязь, а поле `description` — текстовое описание связи. Объекты, определенные в узле `Anchor`, являются объектами, с которыми связаны гиперсвязи. Как и

узел Transform, Anchor является контейнером; другие объекты VRML, содержащиеся в нем, становятся элементами привязки. Синтаксис узла следующий:

```
WWWAnchor {
    url             "url некоего VRML мира"
    description     "описание этого мира"
    children        другие VRML узлы
}
```

Особенности VRML 2.0

VRML 1.0 предназначался для создания, представления и просмотра трехмерных сцен и пространств. VRML 2.0, кроме всего этого, включает в себя и многое другое. Стандарт VRML 2.0 предоставляет инструментарий для создания трехмерных пространств, для которых предусмотрено движение и звуковые эффекты. Кроме того, он позволяет программировать поведение объектов таким образом, что они реагируют на ваше присутствие и действия в этом пространстве. Например, VRML-рыбка может быть запрограммирована так, что уплывает от вас, если вы приближаетесь к ней слишком близко.

Кроме того, VRML 2.0 создает основу для представления в Web трехмерного содержимого, которое может расти и развиваться. Компьютеры все время развиваются и становятся все более мощными и быстродействующими; Internet также не стоит на месте, а высокоскоростные соединения уже сейчас становятся в развитых странах, скорее, правилом, чем исключением, и стандарт VRML должен поспевать за этими изменениями. Основные отличия VRML 2.0 от VRML 1.0 описаны далее; если же приведенного в этой главе материала вам окажется недостаточно — обратитесь по адресу: http://vrml.sgi.com/moving-worlds/.

Придание реалистичности VRML-пространству

VRML 2.0 поддерживает несколько новых типов узлов и полей, увеличивающих реалистичность геометрических построений VRML. Вы можете создавать различный фон для земли и неба, с использованием цвета или изображений (например, облаков или гор), при этом вы можете использовать для повышения реалистичности такие эффекты, как дымка или туман. Для создания ландшафта используются специальные эффекты — в отличие от прежних плоских поверхностей. Добавлены не только визуальные эффекты — новый стандарт VRML обеспечивает трехмерный звук (зависящий от положения точки наблюдения) для придания еще большего реализма.

Взаимодействие

VRML 2.0 включает в себя новый класс узлов, называемых чувствительными узлами (sensor node), которые обеспечивают реакцию на различные воздействия, например на касание объекта или сближение с ним точки просмотра. Другие узлы позволяют отслеживать прошедшее время, третьи — взаимодействие объектов (например, не давая объекту пройти сквозь другой).

Анимация

VRML 2.0 имеет специализированные узлы, позволяющие создавать анимационный эффект для любого объекта вашего VRML-пространства. Такая анимация может быть запрограммирована на автоматическое воспроизведение под воздействием какого-либо фактора (например, приближения точки просмотра). Таким образом можно создавать объекты, изменяющие свой цвет или форму при движении, например Луну или Солнце. Точка просмотра может также быть анимирована, предоставляя посетителю экскурсионный маршрут.

Сценарии

Ключ ко множеству возможностей VRML 2.0, в частности к движению объектов, заключается в его поддержке сценариев. Сценарии дают возможность не только перемещать объекты, но и заставляют их реагировать на другие объекты. Сценарий представляет собой связующее звено между событием, генерируемым чувствительным узлом, и вызываемым им действием.

Прототипы

Еще одно расширение VRML 2.0 связано с прототипами, что позволяет создавать собственные узлы; при этом имеется возможность повторного использования созданного кода.

Дополнительную информацию, учебники и примеры использования VRML, а также множество другой информации вы можете найти на сервере по адресу: **`http://www.neca.com/~vmis/vrml.htm`**.

Ресурсы

Основным хранилищем информации о VRML следует считать, вероятно, узел VRML Repository (`http://www.sdsc.edu/vrml/`). Здесь вы найдете, если не всю информацию по VRML, то уж наверняка — ссылки на другие узлы, где имеется исчерпывающая информация по тому или иному вопросу, связанному с VRML.

Программное обеспечение

Широкий диапазон программного обеспечения для работы с VRML перечислен на указанном выше сервере. Основные категории программного обеспечения VRML:

- VRML броузеры и модули-приложения;
- VRML и другие трехмерные программы разработки;
- приложения VRML.

Библиотеки

Если вы намерены всерьез заняться VRML, вряд ли стоит начинать работу совсем с нуля, особенно если учесть, что в Web имеется множество мест, где можно найти готовые объекты VRML, которые вы сможете использовать в своих целях.

Наилучшее место для начала поисков — уже упоминавшийся ранее узел VRML Repository.

Еще одно место в Web, где можно найти множество объектов и миров VRML, — VRML Models по адресу: `http://www.ocnus.com/models/models.html`. Основное достоинство VRML Models — трехмерная галерея объектов VRML Mall, в которой вы можете просмотреть все интересующие вас объекты.

И, наконец, Mesh Mart, расположенный по адресу: `http://www.meshmart.org/`, который представляет собой набор исходных текстов трехмерных объектов. Хотя большинство этих объектов имеют не VRML-формат, их очень легко конвертировать в VRML. Здесь же, на сервере Mesh Mart, вы можете найти программу для такой конвертации (`wcvt2pov.exe`), работающую с файлами следующих форматов:

- AOFF (GEO);
- AutoCAD (DXF);
- 3D Studio (3DS);
- Neutral File Format (NFF);
- RAW (RAW);
- TPOLY (TPOLY);
- TrueType fonts (TTF);
- Wavefront (OBJ).

При этом программа может конвертировать данные и создавать файлы следующих форматов:

- AutoCAD (DXF);
- 3D Studio (3DS);
- Neutral File Format (NFF);
- OpenGL;
- POVRay V2.2 (POV, INC);
- PovSB (PSB);
- RAW (RAW);
- TPOLY (TPOLY);
- VRML 1.0 (WRL);
- Wavefront (OBJ).