Школа современного программирования

Авраамова Ольга Дмитриевна

СОЗДАНИЕ VRML-МИРОВ. ЧАСТЬ 1

1. ЧТО ТАКОЕ VRML НЕОБХОДИМЫЕ ИНСТРУМЕНТЫ

Язык моделирования виртуальной реальности был призван расширить плоский мир HTML-страниц путем создания трехмерного пространства в WWW. Однако эта технология с тем же успехом может быть использована в интранет-сетях и в локальных приложениях.

VRML (Virtual Reality Modeling Language) – это формат файлов для описания интерактивных трехмерных объектов и миров. VRML – весьма развитый декларативный язык, он способен представлять статические и динамические трехмерные объекты, обладающие гиперсвязями с другими средами, такими, как текст, звуки, видео и картинки. Очень широк спектр его возможных приложений – от простейшей иллюстрации геометрических понятий до разработки Doom-образных игр. Он может быть использован в самых разных областях – в инженерной и научной визуализации, мультимедиа-презентациях, развлекательных продуктах, при создании веб-страниц, но нас, конечно, в первую очередь, интересуют его возможности в образовании. Даже простейшие конструкции, состоящие из геометрических примитивов, могут сослужить хорошую службу в преподавании математики, физики, астрономии, химии.

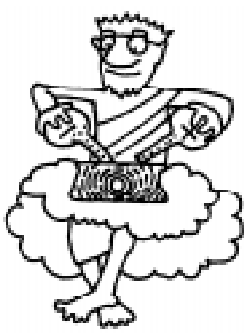
VRML-файлы могут содержать ссылки на файлы многих форматов. Так, например, JPEG, PNG, GIF и MPEG файлы могут быть использованы в качестве текстур объектов, звук может существовать в формате WAV или MIDI. В качестве встроенного языка сценариев используется ECMAScript – базовая стандартизированная версия языка JavaScript. Кроме того, узлы сценариев могут ссылаться на байт-код на языке Java.

Программы визуализации виртуальных миров встроены в основные браузеры (WorldView – в Internet Explorer (IE) и CosmoPlayer – в Netscape Communicator (NC)). Кроме того, свободно доступен VRML-браузер Cortona фирмы ParallelGraphics (http://www.paragraph.ru). Он может работать и с IE, и с NC. Из инструментальных средств разработки на VRML строго необходим лишь текстовый редактор. Да, файл VRML – это текстовый файл открытого формата, доступный для создания и редактирования простым текстовым редактором. Для второй версии языка VRML действует утвержденный международный стандарт ISO/IES 14772-1:1997, спецификация может быть свободно получена на сервере Web3D-консорциума (http://www.web3d.org). Таким образом, использование VRML может стать бесплатным и качественным способом визуализации пространственных объектов.

Суммируя сказанное, отметим, что для создания VRML-мира вам нужен текстовый редактор (как вспомогательное средство пригодятся лист бумаги и каран-

даш), а для просмотра результатов вашего труда – VRML-браузер. VRML-файл должен иметь расширение .wrl.

2. РОЖДЕНИЕ НОВОГО МИРА



Первое, что нужно знать о VRML – это то, что все файлы начинаются со строки заголовка. Вот она:

#VRML V2.0 utf8

Эта строка сообщает браузеру, что перед ним – файл VRML, а также, какая именно версия VRML используется. В нашем случае – это версия 2.0. Нужно отметить, что VRML различает строчные и заглавные буквы, поэтому строчка-заголовок должна выглядеть точно так, как приведено выше. Подстрока «utf8» говорит браузеру, что была использована кодировка utf8. Собственно, только utf8 и разрешена в VRML97, поэтому выбора у нас нет. Кодировка utf8 является прозрачной для ASCII символов с кодами от 0 до 127, поэтому если использовать только латинские буквы и стандартные символы, можно вообще ни о чем не беспокоиться и работать так, как вы работали бы с любым текстовым файлом.

Любая строка, кроме первой, начинающаяся с символа #, является комментарием и будет проигнорирована интерпретатором VRML.

А теперь перейдем к реальной структуре VRML-файла. VRML-мир описывается при помощи иерархического графа сцены. Основные строительные объекты графа называются узлами (nodes). Среди них – геометрические примитивы, узлы, определяющие внешний вид объектов, узлы звука и его свойств, несколько типов группирующих узлов. Узел записывается в файл при помощи своего названия и пары фигурных скобок. В фигурных скобках перечисляются поля.

В некоторых полях узла могут содержаться другие узлы, в тех узлах, в свою очередь, – еще узлы, и так далее. Такая структура графа сцены делает возможной сборку больших миров или сложных объектов из более простых частей.

Обычно название узлов начинается с большой буквы (Group, Transform, IndexedFaceSet), в то время как названия полей – с маленькой. Тем не менее, в названиях полей иногда встречаются заглавные буквы посередине (coordIndex).

Постойте, все это очень мило, но посмотреть-то пока и не на что. Давайте создадим простейший объект. Мы создадим объект «сфера» и постараемся понять структуру геометрического узла. Все геометрические объекты определяются в узле Shape (форма). Узел Shape содержит два поля, описывающие объект. Это поля appearance (внешность) и geometry (геометрия). Они соответственно задают внешний вид и форму объекта.

```
#VRML V2.0 utf8
Shape {
    appearance Appearance {
       material Material {
           diffuseColor 1 0 0
       }
    }
     geometry Sphere {
        radius 2
    }
}
```

Да, у нас действительно получилась сфера (рисунок 1). Если воспроизвести этот пример на экране, можно убедиться в том, что цвет сферы – красный. Расшифруем чуть подробнее, что означают записи в файле.

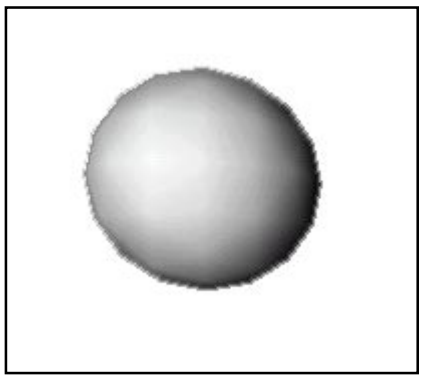


Рисунок 1

В поле appearance должен находиться узел Appearance. Внешний вид объекта определяется либо путем задания свойств материала (узел Material), либо заданием ссылки на файл с текстурой, то есть в узле Appearance могут быть поля material или texture. Они могут появляться и одновременно, но, как правило, свойства текстуры будут тогда доминировать. Мы выбрали Material и в поле diffuseColor указали красный цвет (1 0 0).

Цвета в VRML представляются в RGB-кодировке, как и в языке HTML. Каждый цвет можно представить как комбинацию красной (Red), зеленой (Green) и синей (Blue) компонент, причем все они принимают значения от 0 до 1. В этой кодировке ярко-синий получит значение 0 0 1, белый – 1 1 1, серый – 0.5 0.5 0.5, абсолютно черный – 0 0 0. Если не хочется особо экспериментировать, можно воспользоваться опцией выбора дополнительного цвета в программе Paint, но полученные значения для RGB-компонент разделить на 255. Если и этого делать не хочется, можно скачать с сайта нашей лаборатории (www.srcc.msu.su/vrml) программу перекодировки, включающую в себя стандартный для Windows цветовой диалог и три окошка для готовых значений красного, зеленого и синего.

В поле geometry мы использовали готовый примитив для сферы (узел Sphere), а в единственном поле этого узла – raduis – задали радиус 2. В поле geometry могут находиться также куб, цилиндр, конус или вообще произвольная поверхность, явно заданная своими гранями.

Кстати, узлы часто имеют массу полей, но указывать каждый раз значение каждого поля вовсе не обязательно. Если значение какого-либо поля узла не указано, в большинстве случаев браузер автоматически подставит весьма разумное значение по умолчанию.

Синтаксис, использующий узел Appearance в поле appearance, может показаться странным, но это только на первый взгляд. Когда мы займемся повторным использованием объектов, мы увидим, что такая структура позволяет корректно наследовать описания внешних свойств, что бывает полезным при наличии множества объектов с одинаковыми внешними параметрами.

3. СИСТЕМА КООРДИНАТ И ОСИ

Все расстояния в VRML измеряются в метрах. Если вы делаете абсолютно изолированный мир, то этот факт не имеет особого значения, но если вы захотите впоследствии связать свой мир с мирами других разработчиков или использовать для построения своего мира внешние файлы, то лучше придерживаться стандарта. С другой стороны, для астрономической модели в качестве единицы измерения может лучше подойти, скажем, световой год.

Система координат VRML-мира приведена на рисунке 2. Ось X горизонтальна, ось Y направлена вертикально вверх и ось Z направлена прямо на зрителя.

Вращения в VRML осуществляются по правилу правой руки (рисунок 3): если смотреть с положительного конца оси вращения, вращение в плоскости, перпендикулярной оси, осуществляется против часовой стрелки.

Нужно еще запомнить, что все углы измеряются в радианах. Поворот на прямой угол в положительном направлении будет иметь величину 1.57, в отрицательном – величину -1.57.

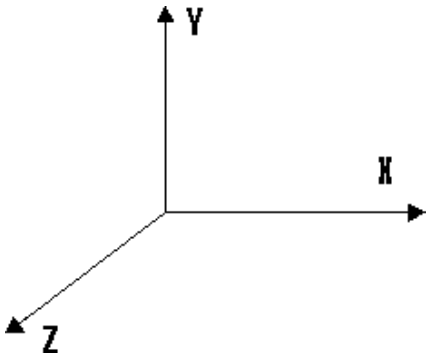


Рисунок 2

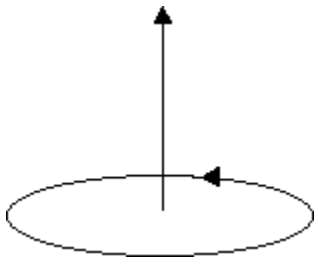


Рисунок 3

4. ПРЕОБРАЗОВАНИЯ ОБЪЕКТОВ

Вернемся к нашему примеру. Центр получившейся сферы расположен в начале координат. Согласитесь, что не очень интересно, если все объекты мира будут расположены в одном месте. Как изменить такую ситуацию? Для этого в VRML существует три типа преобразований (transforms). Это перемещения (translations), вращения (rotations) и масштабирование (scale). Узел Transform не обязан содержать все три поля сразу. Там может находиться, например, только вращение. Все преобразования, описываемые узлом Transform, относятся к объектам-потомкам этого узла, то есть к узлам, перечисленным в поле children.

```
Transform {
   translation 1 1 1
   rotation 0 1 0 0.78
   scale 2 1 2
   children [
      …
   ]
}
```

Перемещения и масштабирования очень схожи. Оба имеют три аргумента – значения *x*, *y* и *z*. Перемещение сдвигает центр объекта на *x* по оси X, на *y* по оси Y и на *z* по оси Z. Масштабирование же умножает размер объекта на эти значения в соответствующем направлении. Перемещение на 0 в каком-либо направлении не изменит положения объекта по этой оси. Предполагается, что все коэффициенты масштабирования неотрицательны.

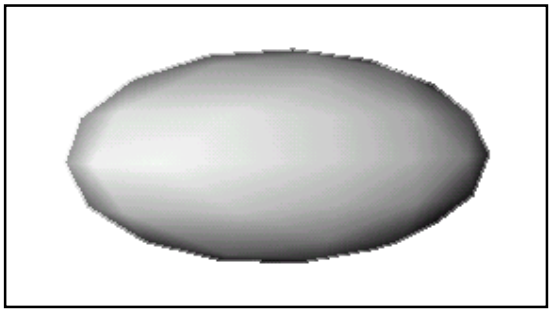


Рисунок 4

Таким образом, если мы захотим получить эллипсоид вращения, у которого большая полуось в два раза больше малой, то возьмем сферу и растянем ее в два раза по оси X (рисунок 4):

```
#VRML V2.0 utf8
Transform {
   scale 2 1 1
   children [
      Shape {
            appearance Appearance {
                material Material {
              }
            }
            geometry Sphere {
            }
      }
  ]
}
```

Очень важно помнить, что масштабирование происходит по отношению к исходной точке мира – точке O с координатами (0,0,0), – а не по отношению к центру объекта. Поэтому для того, чтобы масштабирование шло от центра объекта, сам этот центр должен оказаться в точке O. Заметьте также, что мы решили использовать заданные по умолчанию значения радиуса исходной сферы (будет использован радиус 1) и ее цвета (будет использован серый). Но то, что приведено выше, – действительно корректный VRML-файл.

Вращение слегка отличается от преобразований двух первых типов. Оно воспринимает четыре аргумента. Первые три из них – координаты оси вращения, а четвертый определяет угол поворота в радианах. Чтобы повернуть объект, например, на 45 градусов вокруг оси X, следует написать:

```
Transform {
   rotate 1 0 0 0.78
   children [
        …]
}
```

Длина направляющего вектора оси вращения особого значения не имеет, понятно, что она не должна быть нулевой. Но будет она равна 1 или 50, не важно.

5. ГЕОМЕТРИЧЕСКИЕ ПРИМИТИВЫ

В этом разделе мы рассмотрим простейшие геометрические узлы: Box (прямоугольный параллелепипед), Cylinder (цилиндр), Sphere (сфера) и Cone (конус).

Нужно обратить внимание на то, что эти узлы могут быть помещены только в поле geometry узла Shape, а не могут быть использованы в качестве самостоятельных корневых или дочерних узлов.

Форма, задаваемая узлом Box, ясна из названия. Это куб или прямоугольный параллелепипед, в общем, действительно, ящик. Узел Box со всеми параметрами по умолчанию – это куб со стороной два метра, с центром в начале координат и сторонами, параллельными координатным осям. Можно задать размеры и явно:

```
geometry Box {
   size 5.5 3.75 1.0
}
```

В поле size задаются размеры параллелепипида по осям X, Y и Z. Все эти числа должны быть положительны.

Узел Цилиндр чуть сложнее. Его размеры по умолчанию – от –1 до +1 по всем направлениям, то есть он имеет высоту 2 и радиус 1. Он точно так же отцентрирован вокруг начала координат. Его ось параллельна вертикальной оси координат. Для того чтобы задать иные измерения, нужно ввести желаемые значения в поля radius (радиус) и height (высота). Эти числа должны быть неотрицательны.

Следующий код

```
geometry Cylinder {
   radius 0.5
   height 10
}
```

произведет тонкий длинный цилиндр.

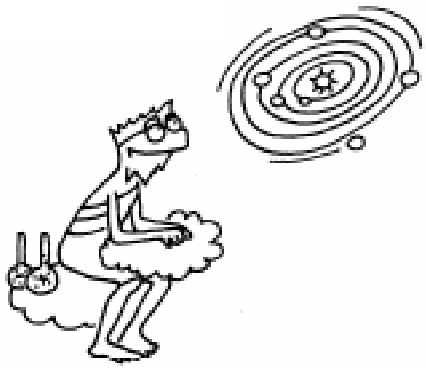
Ось цилиндра изначально ориентирована параллельно оси Y. Для того чтобы задать цилиндр произвольной ориентации, нужно соответствующий ему узел Shape сделать дочерним узлом узла Transform.

Конус сходен с цилиндром, только поле radius (радиус) заменено полем bottomRadius (радиус нижнего основания), что вполне естественно:

```
geometry Cone {
   bottomRadius 5
   height 10
}
```

ЗАКЛЮЧЕНИЕ

Настало время оглянуться на пройденный путь. Нужно осознать, что мы уже в состоянии создать как модель Солнечной системы (пока статическую и без кольца у Сатурна), так и модель молекулы воды. В будущем нас ждут гиперссылки, точки наблюдения, сложные поверхности и источники освещения, текстуры и тонкие свойства материалов, звуки и анимация. Оставайтесь с нами!



Авраамова Ольга Дмитриевна, зав. лабораторией информационных систем математических наук Научно-исследовательского вычислительного центра МГУ им. М.В.Ломоносова.

НАШИ АВТОРЫ