Авраамова Ольга Дмитриевна

МУЛЬТИМЕДИА-КОМПОНЕНТЫ ВИРТУАЛЬНЫХ МИРОВ

Во вводном цикле статей по программированию на языке VRML («Создание VRML-миров», Компьютерные инструменты в образовании, №№ 5, 6, 2000 г.) мы рассмотрели геометрические примитивы, создание тел и линий произвольной формы, гиперссылки и повторное использование объектов. Теперь мне хотелось бы остановиться на тех вещах, которые помогают оживить виртуальный мир, – на звуках, текстурах, видеофрагментах, а также привести пример интерактивного взаимодействия с пользователем.

1. ЗВУКИ

Для того чтобы мир стал более занимательным, нужно добавить в него звук. Это может быть тихая фоновая музыка, могут быть звонки и сирены, может быть журчание ручья и пение птиц – что больше подходит по сюжету. И все это создается при помощи всего двух узлов – узлов Sound и AudioClip. Узел Sound (звук) отвечает за расположение источника и область слышимости звука, в то время как сам звук задается в узле AudioClip. Итак, звук...

```
Sound {
   minBack 5
   minFront 5
   maxFront 50
   minFront 50
   location      0 0 0
   direction     0 0 1
   intensity     1
   source AudioClip {
      loop TRUE
      url "music.mid"
   }
}
```

Звук распространяется по эллипсоиду, как показано на рисунке 1. Внутри маленького эллипса интенсивность звука максимальна. Внутри большого звук затухает с расстоянием, а вне его уже ничего не слышно.

В поле direction (направление) находится вектор, указывающий ориентацию источника звука. Intensity – это громкость звука. Значение этого поля, равное единице, соответствует громкости звука, заданной в звуковом файле, а ноль – это полная тишина. Location – это три числа, определяющие координаты источника звука. Четыре max/min Front/Back параметра отвечают за форму нашего эллипсоида. Для того чтобы получить звук, одинаково распространяющийся по всем направлениям, нужно положить maxFront=maxBack и minFront=minBack. Если вам не хочется, чтобы звук затухал на расстоянии, положите все эти четыре значения равными и достаточно большими, чтобы покрыть весь

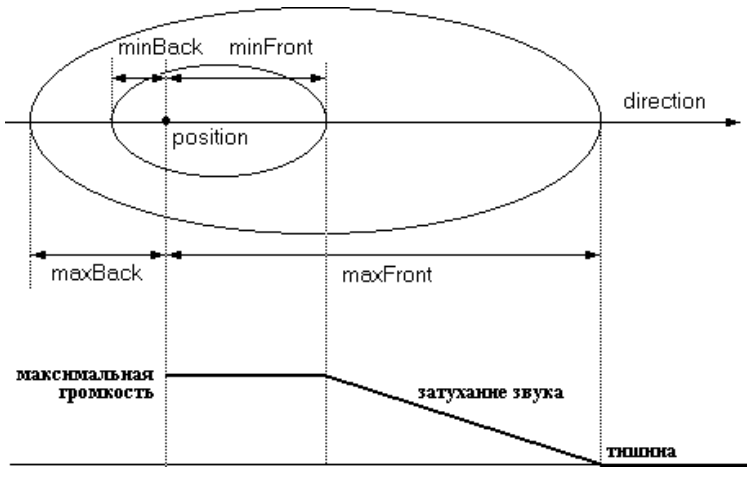

Рисунок 1

мир. Оставив максимальные значения в десять раз больше минимальных, мы получим реалистичное уменьшение звука на расстоянии.

Нерассмотренным осталось только поле source. На самом деле это очень важное поле. Там находится узел AudioClip, в котором и определяется, что и когда будет играть источник звука. В поле url размещен список файлов в формате несжатого wav или General MIDI. Браузер воспроизведет первый найденный файл из этого списка. В wav-файлах могут находиться оцифрованные звуки, а MIDI хорош для компактного представления музыки. Напомним, что wav-файлы можно легко создавать при помощи входящей в комплект Windows стандартной утилиты «Звукозапись». Поле loop (цикл) определяет, будет ли звук воспроизводиться повторно. Циклическое воспроизведение полезно при создании фоновых звуков.

В полях startTime и stopTime определяется, когда начнется воспроизведение звука и когда оно закончится. Время в VRML измеряется в секундах, прошедших с полуночи 1 января 1970 года. Впрочем, абсолютное значение времени особого значения не имеет, существенную роль играет только связывание точек начала и конца с сенсорами (рисунок 2). Сенсор (в нашем примере – датчик касания), будучи активированным, породит исходящее событие touchTime с параметром, равным времени активации. Если его теперь маршрутизировать на поле startTime узла AudioClip, то тут же начнется воспроизведение звука (рисунок 2).

Для того чтобы осуществить эту маршрутизацию, мы используем следующую строчку кода в конце файла:

```
ROUTE MySensor.touchTime TO MySound.startTime.
```

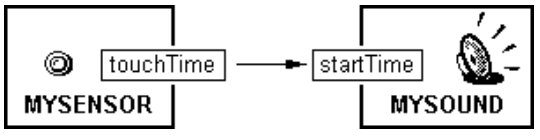

Рисунок 2

По щелчку на сфере заиграет музыка. Вот она, интерактивность!

```
#VRML V2.0 utf8
WorldInfo {
   title "Example 1"
}
Group{
    children[
       Shape {
           appearance Appearance {
               material Material {
                       diffuseColor
0.5 0.5 0.5
                   }
               }
               geometry Sphere {
               }
       }
   DEF MySensor TouchSensor {
   }
   ]
}
Sound {
   minFront 50
   minBack 50
   maxFront 50
   maxBack 50
   source DEF MySound AudioClip {
      url "music.mid"
    }
}
ROUTE MySensor.touchTime TO
MySound.startTime
```

Нужно еще заметить, что наш датчик был сконструирован из узла TouchSensor, не имеющего визуального представления в мире, и сферы, осуществляющей такое представление. Они объединены одним группирующим узлом Group. Вообще же, сфера действия того или иного сенсора распространяется на все узлы-потомки его родительского узла.

2. ТЕКСТУРЫ

Поле Texture (текстура) узла Appearance может содержать один из нескольких типов текстурных узлов. Мы начнем с узла ImageTexture (текстура-рисунок). Она отображает неподвижную картинку на поверхность объекта. Стандарт строго требует поддержки браузерами форматов JPEG и PNG, но поддержка

иных форматов не возбраняется. В частности, прямо рекомендована поддержка формата GIF (включая прозрачность).

Узел ImageTexture содержит поле url, в которое и помещается ссылка на файл с картинкой текстуры:

```
ImageTexture {
      url     ["granpa.jpg"]
}
```

Квадратные скобки возникли из-за того, что таких URL может быть несколько. При работе в сети браузер переберет ссылки одну за другой, пока какая-нибудь из них не окажется доступной.

Используемые изображения могут обладать прозрачностью, и в этом случае их свойства прозрачности будут обладать приоритетом перед начальной прозрачностью объекта. Если вы используете текстуру из оттенков серого, diffuseColor (нормальный цвет) объекта будет умножаться на интенсивность оттенка текстуры, и полученная таким образом текстура будет наложена на объект.

На самом деле, путем комбинирования свойств материала (узел Material) и ImageTexture можно создать множество занятных эффектов. Как правило, они ведут себя предсказуемым образом, а эксперименты доставят вам массу удовольствия.

Итак, чтобы повесить картинку на стенку комнаты, нужно сделать следующее:

```
#VRML V2.0 utf8
WorldInfo {
   title "Example 2"
}
Shape {#wall
      appearance Appearance {
      material Material {
      diffuseColor 1 1 1
      }
   }
   geometry Box {
         size 4 4 0.4
   }
}
Shape {#picture
   appearance Appearance {
          texture ImageTexture{
              url ["monaliza.png"]
          }
   }
   geometry IndexedFaceSet {
       coord Coordinate{
point[-1.11 1.05 0.3, 1.11 1.05 0.3,
      1.11 -1.05 0.3,-1.11 -1.05 0.3]
       }
       coordIndex [3 2 1 0 -1]
   }
}
```

Рисунок 3

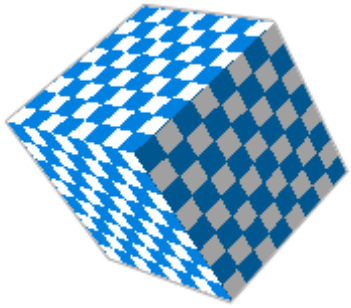

Рисунок 4

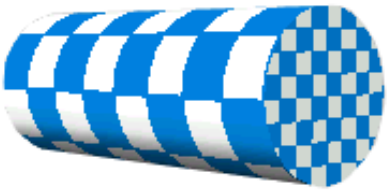

Рисунок 5

Мы создали белый параллелепипед, изображающий стенку, и при помощи узла IndexedFaceSet расположили вблизи стенки картину (рисунок 3). Само изображение на картине создается при помощи наложения текстуры.

Как же взаимодействуют текстуры со знакомыми нам из первой части статьи объектами? На куб текстуры накладываются грань за гранью, на каждую грань индивидуально (рисунок 4). Вокруг цилиндров текстуры обертываются таким образом, чтобы края текстуры сходились сзади. Затем из текстуры вырезаются кружочки и накладываются на верхнее и нижнее основание (рисунок 5). Текстуры обертываются вокруг сферы, а затем скалываются у полюсов. Таким образом, если текстура несет на себе прямоугольную сетку, то ближе к полюсам вертикальные линии будут сходиться, в то время как горизонтальные сохранят дистанцию (рисунок 6).

Рассмотрим еще одну весьма распространенную задачу – нанести на достаточно большую поверхность несколько экземпляров повторяющейся текстуры.

Для этого нам нужно поставить узел TextureTransform в поле textureTransform узла Appearance. В узле TextureTransform, в частности, находится поле scale, отвечающее за «размножение» текстуры по горизонтали и по вертикали.

```
TextureTransform {
   scale       1 1     # (-inf, inf)
}
```

Для того чтобы расположить на поверхности три экземпляра текстуры в горизонтальном направлении и два – по вертикали, нужно указать scale 3 2.

Рисунок 6

3. АНИМИРОВАННЫЕ ТЕКСТУРЫ

При помощи узла MovieTexture MPEG-фильм отображается на поверхности объекта. Этот узел имеет те же три поля, что и ImageTexture, а также ряд других полей. Вот они:

speed – скорость прокрутки. Значение поля speed, равное единице, означает естественную скорость фильма, двойка – удвоенную частоту кадров, а ноль заморозит на поверхности первый кадр фильма. Отрицательные значения скорости заставят фильм крутиться в обратном направлении.

loop – логическое значение TRUE или FALSE, определяющее, будет ли фильм воспроизводиться повторно.

Поле startTime (время начала) определяет, когда фильм должен стартовать. Поле stopTime определяет время окончания фильма.

Синтаксис этого узла приведен ниже:

```
MovieTexture {
    loop                FALSE
    speed               1
    startTime           0
    stopTime            0
    url                 ["film.mpg"]
}
```

Узел MovieTexture может быть также использован как источник звука для узла Sound. Представляется разумным использовать эти узлы одновременно, чтобы смотреть фильм и слушать его звуковую дорожку. Это легко осуществить при помощи ключевых слов DEF/USE:

```
Shape {
    appearance Appearance {
      texture DEF MOVIE MovieTexture {
                 url "http://..."
        }
        geometry Box {
        }
    }
}
Sound {
     source USE MOVIE
}
```

В этом случае аудио и видео будут автоматически синхронизированы, поскольку узел MovieTexture только один, и, соответственно, обладает только одним набором из времени начала, цикличности воспроизведения и времени окончания.

4. НЕКОТОРЫЕ СПЕЦИАЛЬНЫЕ ЭФФЕКТЫ

Обладающие прозрачностью текстуры могут выручить в самых неожиданных ситуациях.

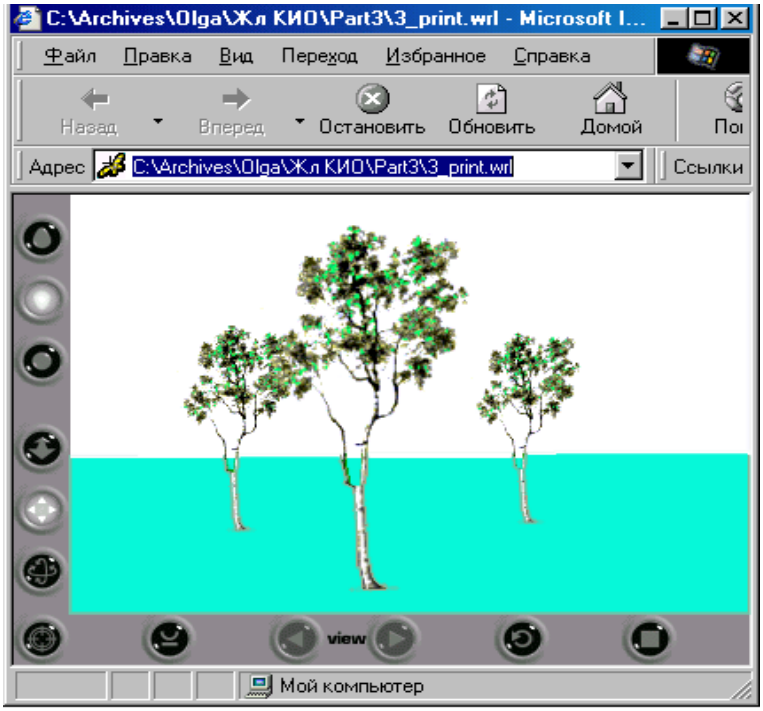

Рисунок 7

Так, например, изображение деревьев на прозрачном фоне позволит создать идиллический пейзаж, не заставляя в то же время автора мира скрупулезно программировать изображение каждого листочка. Единственная сложность – сделать так, чтобы гуляющий среди деревьев аватар не обнаружил, что они картонные. Но для этого в VRML существует специальное приспособление – узел BillBoard (доска для объявлений). Все находящиеся в этом узле геометрические объекты при передвижении путешественника поворачиваются так, чтобы быть обращенными к путешественнику все время одной стороной (рисунок 7).

```
#VRML V2.0 utf8
WorldInfo {
   title "Example 3"
}
Background {
    groundColor [0 0.6 0] #green
    skyColor  [0.7 0.7 1] #light blue
}
DEF Tree Billboard{
       axisOfRotation 0 1 0
       children[
           Shape{
             geometry Box{
               size 1.2 2.4 0.01
              }
             appearance Appearance{
              texture ImageTexture{
                 url [ "tree.gif"]
               }
              }
           }
        ]
}
Transform{
        translation 2  0  -3
        children USE Tree
}
Transform{
        translation -1  0  -3
        children USE Tree
}
```

Мы видим, что в узле BillBoard есть поле children, предназначенное для хранения геометрических объектов, составляющих доску объявлений. В поле axisOfRotation находится та ось, вокруг которой будет вращаться доска объявлений. По умолчанию ось вращения имеет координаты 0 1 0, то есть совпадает с вертикальной осью координат. Обычно это и бывает нужно.

Кроме того, доски для объявлений очень полезны для изготовления текстовых меток. Было бы удобно, чтобы надписи на осях координат были хорошо различимы независимо от того положения, из которого разглядывают график. Тогда дочерним узлом узла BillBoard нужно сделать геометрический узел, содержащий текст:

```
#VRML V2.0 utf8
WorldInfo {
   title "Example 4"
}
Billboard{
       axisOfRotation 0 1 0
       children[
          Shape{
            geometry Text {
               string  ["Text"]
             }
          }
        ]
}
```

По умолчанию текст располагается во фронтальной плоскости мира – плоскости Z=0. Ему, как и всякому геометрическому объекту, можно придавать цвета, и на него можно накладывать текстуры. Однако геометрическим текстом злоупотреблять не стоит. Помимо проблем с кириллицей, большой текст, порождающий огромное количество полигонов, может стать ловушкой для производительности мира. Если нужно передать пользователю большое количество текстовой информации, нанесите текст на рисунок-текстуру (ImageTexture) или же вложите свой мир в HTML-страничку при помощи тега **<EMBED SRC="vrml.wrl">**.

Напомню, что полная спецификация языка VRML находится на сайте www.web3D.org, а полные тексты всех примеров можно найти по адресу www.srcc.msu.su/vrml/.

НАШИ АВТОРЫ

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