Авраамова Ольга Дмитриевна

СОЗДАНИЕ VRML-МИРОВ. ЧАСТЬ 2

1. ПОВТОРНОЕ ИСПОЛЬЗОВАНИЕ ОБЪЕКТОВ

Если у вас есть много похожих объектов, повторение одних и тех же описаний может стать раздражающим. Лучше их один раз определить, а затем использовать повторно. Для этого нужно дать объекту имя, скажем, MySphere. Это делается при помощи ключевого слова DEF. Затем, когда сфера потребуется вновь, мы сможем просто набрать **USE MySphere** вместо того, чтобы повторять все описание.

Наш код будет выглядеть так:

```
#VRML V2.0 utf8
WorldInfo {
 title «Example 1»
 info [«VRML - часть 2, пример 1»
 ]
}
DEF MySphere Shape {
 appearance Appearance {
    material Material {
    }
 }
 geometry Sphere {
 }
}
Transform{
   translation 2.5 0.5 0.25
   children [USE MySphere]
}
```

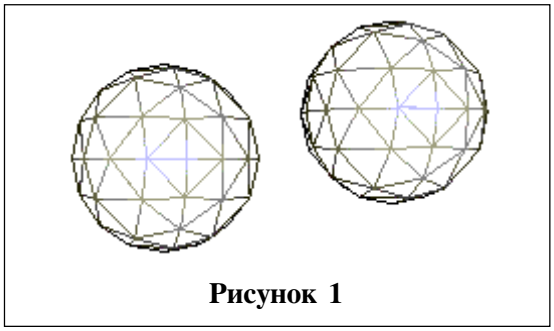
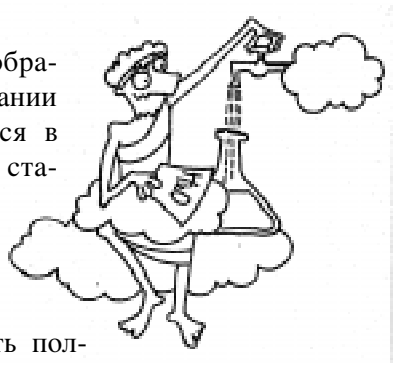


Рисунок 1

Заметим, что при воспроизведении сцены была использована опция Wire Frame (каркасный рендеринг) в окне параметров браузера.



Таким образом, при создании упоминавшейся в первой части статьи молекулы воды вовсе не обязательно два раза давать полное описание каждого атома водорода:

```
#VRML V2.0 utf8
WorldInfo {
 title «Example 2»
 info [«VRML - часть 2, пример 2»,
 «молекула воды»
 ]
}
DEF O Shape {
 appearance Appearance {
   material Material {
    diffuseColor 1 0.2 0.2 #red
    shininess 0.9
   }
 }
 geometry Sphere { radius 0.66
 }
}
DEF H Transform{
  translation 1.4 0.0 0.0
  children [
     Shape {
        appearance Appearance {
         material Material {
            diffuseColor 0.6 0.6
                    1.0 #light blue
            shininess 0.9
            specularColor 1 1 1
         }
        }
       geometry Sphere { radius 0.32
       }
      }
   ]
}
Transform{
  rotation 0 0 1 1.8316 # 105 degrees
  children [USE H]
}
```

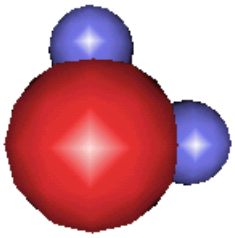
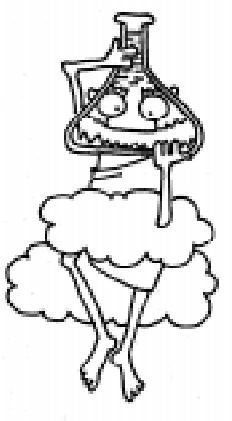


Рисунок 2

Здесь мы создали первый атом водорода на некотором расстоянии от центра атома кислорода, назвав его H, а затем при помощи ключевого слова USE добавили второй, полученный из первого поворотом на 105 градусов вокруг центра во фронтальной плоскости. Заметим, что в узле Material кроме знакомого нам основного цвета – diffuseColor - были использованы поля shininess – степень блеска и specularColor – цвет бликов. Цвет бликов мы сделали ярко-белым (1 1 1), а степень блеска установили близкой к максимуму (0.9). Чуть позже, в примере 4, нам встретится еще одно свойство материала – прозрачность (transparency). Прозрачность, как и степень блеска, задается числом от 0 до 1.



Другим способом повторного использования кода могут служить узлы Inline. Они берут данные из внешнего файла и вставляют в ваш мир. Если у вас есть модель, скажем, цветка, расположенная в файле flower.wrl, то можно вставить ее в сцену следующим образом:

```
Inline {
    url "flower.wrl"
}
```

Включаемый таким образом в поле url файл непременно должен быть синтаксически правильным VRML-файлом, то есть должен иметь заголовок и все прочие атрибуты. Если файл не грузится в браузер самостоятельно, то использовать его в качестве inline-файла тоже нельзя.

Возвращаясь к приведенному выше примеру, попробуем вставить готовую модель молекулы в другой файл и затем ее размножить.

```
#VRML V2.0 utf8
WorldInfo {
 title «Example 3»
 info [«VRML - часть 2, пример 3»,
 «много молекул воды»
 ]
}
DEF H2O Inline {
    url «example2.wrl»
}
Transform{
   translation 2.0 0.0 0.0
   rotation 1 0 0 1.5
   children [USE H2O]
}
Transform{
   translation -2.0 0.0 0.0
   rotation 0 1 0 -1.5
   children [USE H2O]
}
```

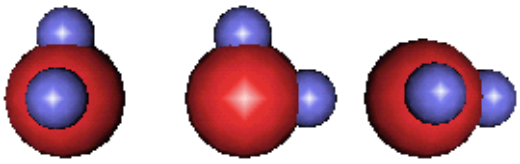


Рисунок 3

Не правда ли, код получился достаточно компактным и вполне читабельным. А если бы это была не вода, а какой-нибудь из высших углеводородов, то польза от повторного использования объектов была бы еще более очевидна.

Эти черты могут очень пригодиться при коллективной разработке миров.

Скажем, при создании раствора один ученик делает модель молекулы воды, другой – поваренной соли, при построении модели класса один ученик делает окно, а другой – дверь. Но, впрочем, до построения помещений нам еще нужно изучить, как же создаются в VRML объекты произвольной формы.

2. СЛОЖНЫЕ ГЕОМЕТРИЧЕСКИЕ ОБЪЕКТЫ

Пока мы умели создавать только простейшие геометрические формы – кубы, конусы, цилиндры и сферы. Это красиво, но не охватывает всех наших потребностей. Иногда бывают нужны лист Мебиуса или гиперболический параболоид.

Для их построения потребуются другие геометрические узлы, к рассмотрению которых мы и переходим. Наиболее важным из всех является узел IndexedFaceSet.

Узел IndexedFaceSet – индексированное множество граней – представляет собой набор вручную задаваемых граней. С его помощью можно определить фигуры абсолютно произвольной формы. Основная идея чрезвычайно проста. Предположим, мы хотим создать четырехугольную пирамиду (рисунок 4).

Перенумеруем вершины, начиная с нуля. (Первая вершина нашего списка получит номер 0, вторая – 1 и т.д.). Дальше нужно определить грани. Чтобы задать грань, заштрихованную на рисунке серым цветом, следует перечислить вершины, ее определяющие: 2, 3 и 4. Вот и все. Процесс следует повторить для всех граней. Небольшое предупреждение. Если вершины, определяющие грань, окажутся некомпланарны (не будут лежать в одной плоскости), результат рендеринга такой грани будет непредсказуем. Надежнее всего использовать триангуляцию (три точки всегда лежат в одной плоскости). При использовании большего количества определяющих грань точек ответственность за компланарность используемых вершин лежит на авторе создаваемого мира.

Узел IndexedFaceSet содержит следующие поля: coord, coordIndex и solid.

Поле coord содержит узел Coordinate (координаты). Он представляет собой просто список координат точек пространства, как показано в приведенном ниже примере. Это те точки, из которых будут строиться собственно грани.

Следующее поле – coordIndex. Это – список граней. Для того чтобы определить грань, следует последовательно ввести номера определяющих ее вершин. Завершить перечисление следует символом –1, чтобы обозначить переход к описанию следующей грани. Номер вершины, напомним, берется из списка в поле coord.

Если вы смотрите на грань с лицевой стороны, направление обхода грани должно быть против часовой стрелки. Это важно, поскольку обратная сторона грани отрисовывается отнюдь не всегда. Впрочем, поле solid (в данном случае – «замкнутый») позволяет откорректировать эту ситуацию. Если объект не является полностью замкнутым, нужно дать указание браузеру делать рендеринг и внутренних поверхностей граней. Это можно сделать,

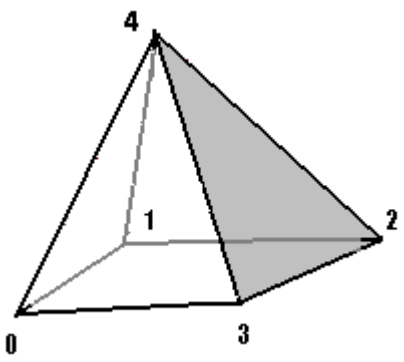


Рисунок 4

задав значение FALSE в поле solid. Иначе из соображений увеличения скорости такая отрисовка производиться не будет.

```
#VRML V2.0 utf8
WorldInfo {
 title «Example 4»
}
Shape{
geometry
   IndexedFaceSet {
      coord Coordinate {
           point [  -1 0 1, 1 0 1,
           1 0 -1, -1 0 -1, 0 2 0]
           }
         coordIndex [0 1 4  -1
                     1 4 2  -1
                     4 2 3  -1
                     0 4 3  -1
                   0 3 2 1  -1 ]
                solid TRUE
   }
   appearance  Appearance{
           material Material {
                diffuseColor 1 0 0
                 transparency 0.5
           }
   }
}
```

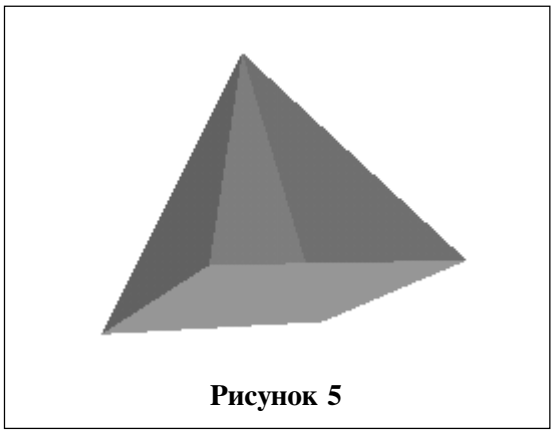


Рисунок 5

Грани должны пересекаться только по установленным ребрам.

Нужно подчеркнуть, что узел IndexedFaceSet дает нам практически неограниченную власть над формой предметов. В частности, он может быть использован для точного построения сложных функционально заданных математических объектов. Любой VRML-браузер обязан поддерживать поверхности, состоящие как минимум из пяти тысяч граней.

При планировании сложного объекта не нужно забывать о таких полезных инструментах, как бумага и карандаш. Даже когда нам нужно использовать всего пять-восемь точек, лучше набросать чертеж на листке бумаги и пометить на нем номера точек и их координаты. Эта простая вещь сбережет в процессе работы очень много времени и сил.

Близким родственником узла IndexedFaceSet является узел IndexedLineSet (индексированное множество линий). Этот узел определяет список линий, которыми нужно соединить точки с заданными координатами. Он полезен тогда, когда нужно провести тонкую линию или создать «проволочную» поверхность. Его структура задается так же, как и у IndexedFaceSet, но отсутствует поле solid, а в поле coordIndex указываются последовательные индексы точек, определяющих линию. Данные для разных серий отрезков отделяются друг от друга минус единицей. Вот фрагмент кода, описывающего линии координат мира, приведенного на рисунке 6:

```
Shape {
    geometry IndexedLineSet {
      coord Coordinate{
         point [-3.5 0 0, 3.5 0 0,
                    0 -1 0, 0 1 0]
      }
      colorPerVertex FALSE
      color Color { color [1 1 1,
                           1 1 1]}
      coordIndex [ 0 1 -1
                   2 3 -1
      ]
    }
}
```

Полностью код этого примера, как и всех остальных, можно найти по адресу www.srcc.msu.su/vrml/part2.

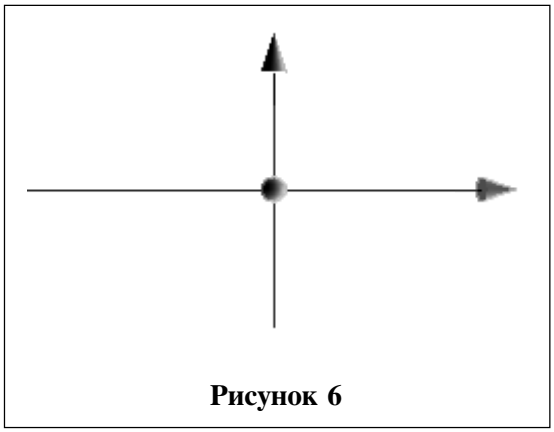
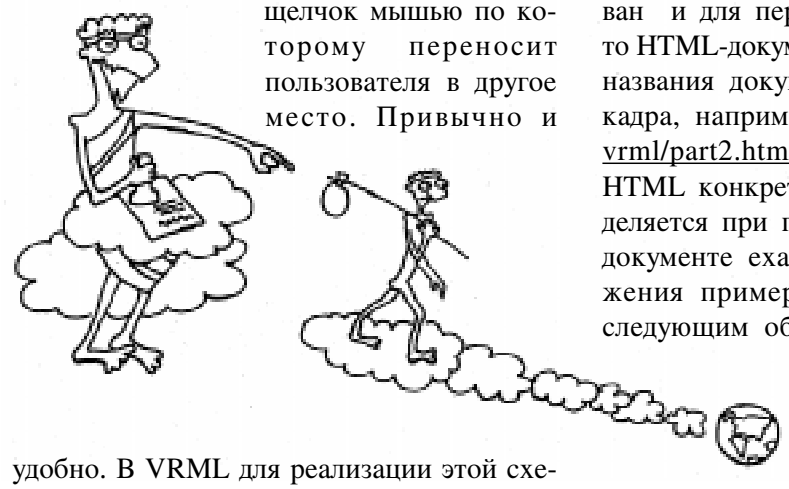


Рисунок 6

Линии считаются бесконечно тонкими. Они могут иметь цвет, но к ним неприменимы текстуры, так как их не на что, собственно говоря, наклеивать. В поле colorPerVertex мы можем поставить FALSE, тогда цвета из списка color применяются к соответствующим линиям последовательно. Если же в этом поле будет стоять TRUE, то цвета будут прикладываться к вершинам, а на линиях, их соединяющих, будут линейно интерполироваться. Так линия, соединяющая точки красного и белого цвета, будет переливаться всеми оттенками розового.

3. ГИПЕРССЫЛКИ. УЗЕЛ ANCHOR

В языке VRML перемещение между страницами осуществляется по тому же принципу, что и в HTML. Имеется объект, щелчок мышью по которому переносит пользователя в другое место. Привычно и удобно. В VRML для реализации этой схемы и существует узел Anchor (якорь).



Узел Anchor – это группирующий узел. В его поле children может находиться целый список, скажем, геометрических узлов, заключенный в квадратные скобки. Щелчок мышью по изображению любого из узлов-потомков перенесет нас на страничку, адрес которой указан в поле url. Попадая в область ссылки, указатель мыши меняет свою форму.

В поле descriprtion (описание) мы поместим словесное пояснение. Оно будет появляться где-нибудь на экране всякий раз, как только указатель «мыши» попадет в область гиперссылки, как в HTML. Конкретное место воспроизведения текста зависит от браузера.

```
#VRML V2.0 utf8
WorldInfo {
 title «Example 6»
}
Anchor{
 children[
   Inline {url «example2.wrl»}
 ]
 url[«example3.wrl»
 ]
}
```

В приведенном примере на экране будет видна все та же молекула воды. При приближении к ней указателя мыши форма его меняется, что указывает на то, что перед нами – активный объект. И действительно, щелчок мышью перенесет нас в другой мир, содержащий уже три молекулы.

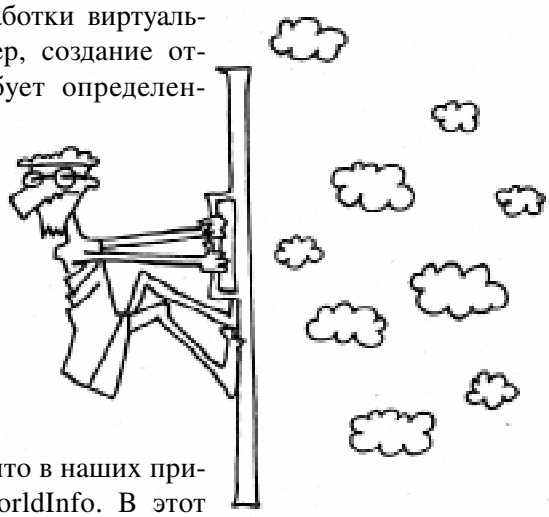
Узел Anchor может быть использован и для перехода в определенное место HTML-документа. Для этого в URL после названия документа нужно добавить имя кадра, например, http://www.srcc.msu.su/vrml/part2.htm#anchor. Напомним, что в HTML конкретная точка документа определяется при помощи атрибута NAME. В документе examples2.htm место расположения примера должно быть помечено следующим образом:

```
<A NAME=»anchor»></A>
```

Гиперссылки могут быть полезны при создании учебных материалов, когда щелчок мышью на определенном объекте виртуального мира перенесет в HTML-документ, содержащий разъяснения и дополнительную информа-

цию. Например, в виртуальном музее живописи каждая картина может служить «входной точкой» в документ, посвященный создавшему ее художнику или, шире, художественному направлению, к которому он принадлежал.

Вторая польза от этой конструкции - просто облегчение разработки виртуальных миров. Так, например, создание открывающейся двери требует определенного времени и терпения. Можно поступить проще – снабдить закрытую дверь гиперссылкой на файл, содержащий модель расположенной за этой дверью комнаты.



4. УЗЕЛ WORLDINFO

Вы могли заметить, что в наших примерах появились узлы WorldInfo. В этот узел помещается название мира (поле title), и там может также содержаться поле info, куда вы можете поместить дополнительную информацию: сведения об авторских правах, инструкции по использованию мира или ключевые слова для поисковых машин. В поле info может находиться произвольное число строк. Их нужно заключить в квадратные скобки.

Нужно заметить, что очень часто перед пересылкой мира с сервера строки комментариев удаляются оптимизатором из VRML-файла. Поэтому та информация, которую вы считаете необходимой для пользователя, должна помещаться в узел WorldInfo.

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