ПРОГРАММНАЯ СИСТЕМА ВИЗУАЛИЗАЦИИ ВИРТУАЛЬНЫХ ПРОСТРАНСТВ

В. Е. Талызин

Санкт-Петербургский информатики и автоматизации РАН
199178, Санкт-Петербург, 14-я линия В.О., д.39
tvemail@euro.ru

УДК 519.8.658.012.011.56(07)
В. Е. Талызин. **Программная система визуализации виртуальных пространств** // Труды СПИИРАН. Вып. 1, т. 3. — СПб: СПИИРАН, 2003.
Аннотация. *Предложена система, облегчающая разработку и управление трехмерными виртуальными пространствами. Рассматриваются новые методы, позволяющие ускорить вывод трехмерной графики на экран, а также способы сжатия данных системы, что является необходимым при пересылке данных через internet*. — Библ. 5 назв.

UDC 519.8.658.012.011.56(07)
V. E. Talyzin. **Program system for visualization of virtual spaces** // SPIIRAS Proceedings. Issue 1, v. 3. — SPb: SPIIRAS, 2003.
Abstract. *The system facilitating development and management by 3D virtual spaces is offered. The new methods allowing accelerating a conclusion the 3D graphics on the screen are considered and also ways of compression of the given system, that is necessary at transfer of the data through an internet.* — Bibl. 5 items.

В связи с возрастающей потребностью современного информационного общества в представлении трехмерных моделей на экранах мониторов, возрастает и потребность в системах, которые будут осуществлять эту визуализацию, упрощая работу программистов и дизайнеров. При использовании таких систем нужно спрограммировать или изобразить сами трехмерные модели и описать те процессы, которые должны происходить в виртуальном пространстве. Все остальное должна сделать система визуализации.

Автором была разработана такая система — Virtual Worlds Engine предоставляющая все необходимые для визуализации возможности. Особое внимание при ее разработке было уделено повышению скорости работы, минимизации размеров данных (для использования системы в internet) а также простоте управления системой.

Самой главной особенностью этой системы является именно простота управления.

Весь процесс управления системой можно поделить на четыре этапа (рис. 1). Каждый из этих этапов может быть пропущен или заменен другим. Это вызвано тем, что система предлагает для использования специальный **скриптовый язык**, который предназначен для упрощения управления возможностями системы, но его использование является необязательным.

В том случае, если процесс создания виртуальной сцены идет при помощи скрипта, то тогда первым этапом будет, естественно, написание программы на скриптовом языке. В результате компиляции и будет получен конечный файл виртуальной сцены, понятный системе.

В процессе компиляции производится анализ того, какие файлы должны быть еще откомпилированы и включены в сцену. Это могут быть файлы, содержащие образы виртуального пространства, созданные либо при помощи конвертера, позволяющего создавать простейшие примитивы и поверхности вращения, либо описанные в одном из популярных форматов виртуальных ми-

Труды СПИИРАН. 2003. Вып. 1. Т. 3. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2003. Issue 1. V. 3. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

ров – **VRML 2.0** и **3DS**, файлы, написанные в одном из этих форматов, умеет создавать практически любой трехмерный графический редактор.

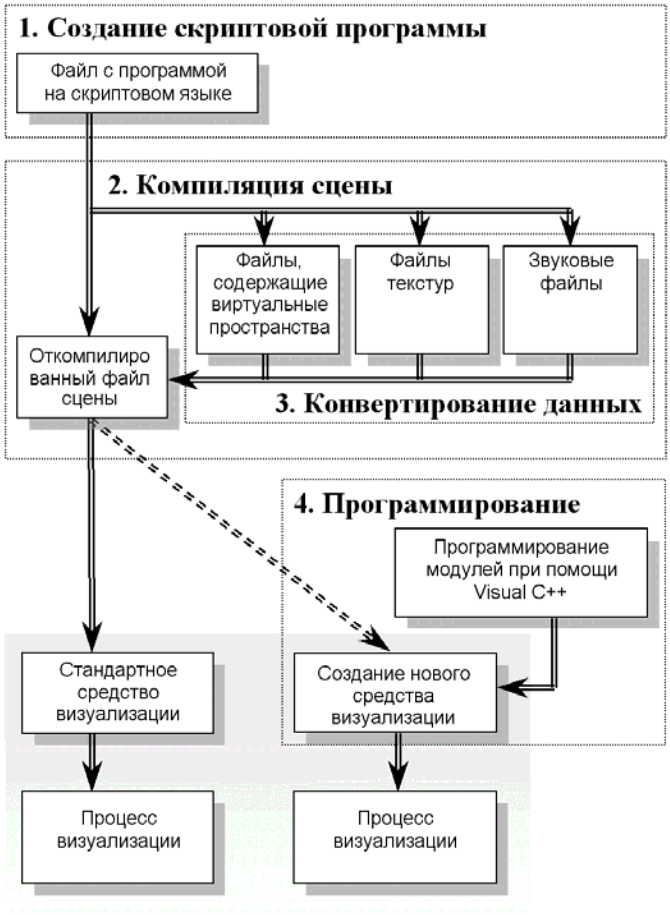

Рис. 1. Преобразование мировых координат объекта в видовые

Также можно подключать текстуры, хранящиеся в одном из двух форматов — BMP или JPEG. Текстуры, описанные в VRML файлах, подключаются автоматически.

Возможно также подключение звуковых файлов, записанных в формате WAV, которые затем можно использовать либо в качестве звукового фона или «привязать» к какому-либо объекту, создав эффект трехмерного звучания. Все это компилируется в специальный VWE-файл, который может быть прочитан специальной программой визуализации, которая и является конченой программой для пользователя. Так представляется стандартный процесс создания сцены, однако он может быть изменен при использовании SDK системы и программировании при помощи Visual C++. Такой способ является гораздо сложнее для разработки, но он позволяет создать нестандартные средства, как компиляции данных, так и средства визуализации.

В состав системы входит специальный компилятор, позволяющий писать программы, описывающие процессы, которые должны происходить в виртуальном пространстве при помощи специального скриптового языка. Этот язык является алгоритмическим. По стилю он напоминает такой язык программирования, как C++. Т.е. он позволяет писать программы в таком виде, в каком хочется пользователю, при соблюдении некоторых синтаксических правил языка, использовать стандартные для всех алгоритмических языков конструкции, такие

Труды СПИИРАН. 2003. Вып. 1. Т. 3. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2003. Issue 1. V. 3. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

как **if** и **for**, создание функций пользователя, математические выражения, макросы и т.п. Но, за счет направленности на конкретную задачу (программирование трехмерных пространств) он значительно проще для понимания и использования в этой области. В него включены специальные директивы, позволяющие загрузить в сцену файлы, описывающие внешний вид виртуальных пространств, файлы текстур, звуковые файлы (рис. 1) . Все остальное не требует подключения дополнительных файлов, и может быть описано прямо при помощи команд скриптового языка.

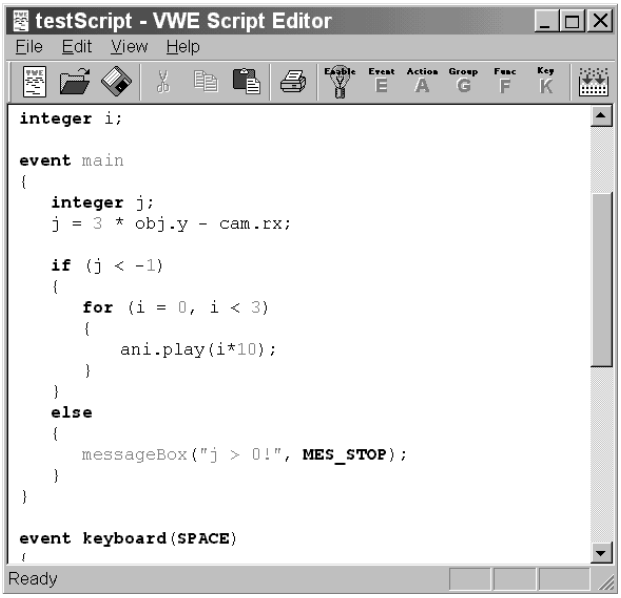

Рис. 2. Внешний вид программы-компилятора

К таким вещам относится создание анимации в сцене, описание реакций на действия пользователя, создание камер, примитивов, поверхностей вращения, а также и всех тех объектов, которые берутся из файлов, описывающих виртуальные пространства – объектов, материалов, источников освещения и т.п. Можно также создавать группы каких-либо объектов (причем не только виртуальных объектов, но и, например, материалов, камер и т.п.), и работать с ними точно также, как и с обычными объектами.

В целом, скрипт позволяет очень быстро и просто задействовать все возможности системы.

Для создания скриптов в систему входит специальная программа, позволяющая упростить вставку в скрипт специальных директив и при помощи цветового кодирования сделать скриптовую программу более читаемой (рис. 2). При использовании автоматического создания директив, в текст программы вставляются шаблоны их написания, и остается только заполнить необходимые области соответствующими значениями.

Достаточно просто можно описать анимацию, которая является, наверное, одной из самых главных директив для подобной системы.

Например, для создания анимации используется директивы **enableAnimation** и **enableAnimationLoop**. Если, например, нужно передвинуть объект с именем «*box1*» по координате *x* с 5 до 10 «виртуальных метров» за 2 секунды, то это можно сделать, написав следующий фрагмент на скриптовом языке:

Труды СПИИРАН. 2003. Вып. 1. Т. 3. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2003. Issue 1. V. 3. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

```
enableAnimation
{
  0: box1.x = 5;
  2: box1.x = 10;
}
```

Слева описывается время, к которому указанная переменная (или свойство объекта) должны принять указанное справа значение. Если же подобное присвоение встречается в директиве только один раз, или указан просто вызов какой-либо функции, то это будет означать, что данное событие должно произойти в указанное время.

К одному значению времени можно привязать сразу несколько событий, при помощи операторных скобок, например:

```
enableAnimation
{
 0: {
     box1.x = 5;
     plane->material.blend = 1;
    }
 2: {
     box1.x = 10;
     plane->material.blend = 0;
     playWave("Dissolve.wav", TRUE);
    }
}
```

В данном примере опять-таки анимируется позиция x объекта *box1*, также, как и свойство «*blend*» назначенного объекту *plane* материала. Анимация этих значений будет плавно протекать на протяжении 2^x секунд, а вызов функции *playWave* произойдет мгновенно и всего один раз, по истечении $2^{ой}$ секунды.

Для описания реакций на действия пользователя, в скрипте существуют директивы *event Keyboard* и *action*. Первая из них позволяет описать реакцию на нажатие указанной в директиве клавиши. Например, чтобы при нажатии на клавишу *Enter* происходил запуск файла демонстрации «Demo.cam», надо написать, следующий фрагмент:

```
keyboard(ENTER)
{
  playDemo("Demo.cam", TRUE);
}
```

Эту директиву можно назначить к абсолютно любой клавише, либо указав одну из преопределенных констант, либо символ, соответствующий этой клавише в кавычках, либо код этой клавиши.

Для обработки более сложных событий существует директива *action*, которая позволяет связать выполнение фрагмента программы с одним из преопределенных событий.

Например, если необходимо переместить объект *door* при входе пользователя в область *doorArea*, можно написать следующий фрагмент программы:

```
action enterArea
{
```


Труды СПИИРАН. 2003. Вып. 1. Т. 3. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2003. Issue 1. V. 3. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

```
    causing DoorArea:
    {
        door.x = 5;
    }
}
```

Эта конструкция позволяет также назначить реакцию на щелчок мышью по объекту, завершению анимации и т.п. Причем можно назначить обработчик для неопределенного действия. Например, если нужно написать скрипт, в котором при щелчке по любому объекту сцены, на экран выводилось бы его название, а объектов в сцене было бы достаточно много, то написать действие на щелчок по каждому из них, было бы достаточно утомительно. А, используя указатель на объект, вызвавший действие в директиве anything, для решения этой задачи надо будет написать всего лишь пару строк программы.

Также, скрипт включает и множество других возможностей, направленных на облегчение создания программы управления виртуальным миром.

Поскольку одним из вариантов использования данной системы является ее использование в сети internet, то при ее создании большое внимание было уделено минимизации размеров, как программ, так и данных. Например, размер стандартной программы визуализации составляет всего лишь около 170 Кб. А после сжатия ZIP-упаковщиком, ее размер составляет около 75 Кб, что сильно упрощает пересылку этой программы по сети internet. При этом, он не требует никаких дополнительных библиотек, кроме стандартных библиотеки OpenGL – opengl32.dll и glu32.dll, которые входят в стандартную поставку Windows.

Рис. 3. Внешний вид сцены в демо-версии системы

Размер данных тоже минимизируется. Так, например, для сцены, поставляемой в демо-версии системы (рис. 3), размер файла данных составляет около 250 Кб, что также достаточно приемлемо для пересылки по internet. При этом в нем содержится сцена, состоящая из 85 трехмерных объектов (2000 треугольников и четырехугольников и свыше 5000 вершин), 35 текстур, фонового звука (плеск поды) и достаточно большая программа, написанная на скриптовом языке.

Достигнуть такой минимизации размера программы визуализации удалось за счет использования только стандартных средств при ее разработке, а также то, что она полностью написана как Win32 программа (без использования дополнительных библиотек и подпрограмм).

Труды СПИИРАН. 2003. Вып. 1. Т. 3. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2003. Issue 1. V. 3. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Рис. 4. Модель объекта в конверторе системы

Минимизация данных осуществляется за счет компрессии. Текстуры сжимаются по алгоритму JPEG-сжатия, а данные сжимаются по LZW алгоритму.

Например, сцена, содержащая в себе одну модель ростральной колонны (рис. 4) занимает всего 45 Кб, в то время, как такой же объект, записанный в VRML формате занимает около 500 Кб. (Хотя конечно, такое сравнение не совсем уместно, т.к. система Virtual Worlds Engine включает в себя гораздо большее количество функций).

Таким образом, выгода от использования данной системы для пересылки данных по сети internet очевидна.

Также, сам формат файла с откомпилированной сценой также обладает некоторыми особенностями. Он рассчитан на то, что возможна такая ситуация, при которой версии программы-компилятора и программы визуализации не совпадают. При этом, например, программа-компилятор, с более новой версией включило в файл сцены какие-либо новые возможности, которые были еще неизвестны программе визуализации с более ранней версией. При этом не возникнет никакой конфликтной ситуации, т.к. в файл сцены автоматически добавляется кодирующая информация, позволяющая определить версию записанных в сцену данных и обеспечить максимально-возможную совместимость. Т.е. в результате системой гарантируется чтение любого откомпилированного файла сцены, независимо от его версии. Просто восприниматься и отображаться будет только та информация, работу с которой поддерживает данная версия программы визуализации.

Также система обеспечивает высокую скорость работы. Высокая скорость выполнения скриптовых программ обеспечивается за счет компиляции текста программы в специальный формат, предназначенный для последующего быстрого чтения. В таком виде он и хранится в откомпилированном файле сцены.

Для повышения скорости отображения, помимо некоторых оптимизаций вывода с использованием библиотеки OpenGL, было применено также отсечение невидимых в данный момент на экране объектов.

Основная идея этого отсечения состоит в преобразовании мировых (трехмерных) координат в видовые (двумерные) координаты экрана. Затем из полученных восьми точек следует построить выпуклый шестиугольник (на рис. 5 это точки 1–6), отбросив из рассмотрения две лишние точки (7 и 8), которые находятся «внутри» шестиугольника. Затем следует проверить, существует ли какая-либо область пересечения шестиугольника и четырехугольника, который

Труды СПИИРАН. 2003. Вып. 1. Т. 3. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2003. Issue 1. V. 3. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

получается из видовых координат окна визуализации. Для этого следует осуществить три проверки:

- Находится ли какая-либо точка шестиугольника внутри четырехугольника
- Находится ли какая-либо точка четырехугольника внутри шестиугольника
- Существует ли пересечение какой-либо стороны шестиугольника со стороной четырехугольника

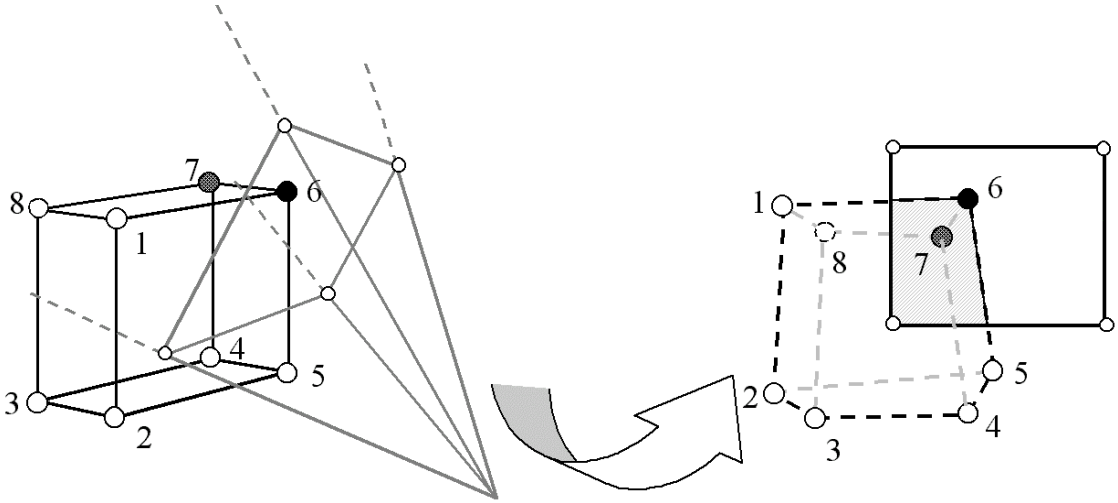

Рис. 5. Преобразование мировых координат объекта в видовые

Если выполняется хотя бы одно из этих условий, то это значит, что область пересечения существует и объект виден на экране.

Система включает в себя множество других возможностей, и спецэффектов.

Вот краткое описание некоторых из них:

1. Возможность создания гравитации в сцене и проверка столкновений. Используемый в этой программе математический алгоритм позволяет делать это с достаточно большой скоростью, к тому же, при проверке столкновений производится оптимизация их поиска за счет предварительной сортировки геометрических образов объектов и проверки столкновений с bounding box объектов.

2. Использование аппаратов частиц. Система позволяет создать с их помощью такие эффекты, как дым, брызги вулкана, брызги гейзера, бесформенные облака, эффекты дождя и снега. При этом можно задать любые другие алгоритмы движения частиц, при помощи SDK системы. Также можно определить, что именно будет частицами. Именно это и определит внешний вид спецэффекта. На данный момент возможно использование в качестве частиц 2D и 3D примитивов (для создания объемных спецэффектов), а также спрайтов. Хотя созданные при помощи спрайтов эффекты не будут являться трехмерными, для пользователя они, скорее всего будут казаться наиболее красивыми.

3. Использование спрайтов. Эта возможность может быть полезна, например, для создания в сцене деревьев, которые, будут отнимать минимум ресурсов компьютера, к тому же они будут гораздо меньше занимать места при сохранении. И при этом они выглядят достаточно реалистично. Также, спрайтами можно создавать такие эффекты, как огонь, взрывы и т.п., являющиеся спрайтами с анимационной текстурой. В моей системе анимационную текстуру можно

Труды СПИИРАН. 2003. Вып. 1. Т. 3. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2003. Issue 1. V. 3. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

создать как набор нескольких обычных текстур. Требуется только указать частоту смены кадров.

4. Прозрачность, смешение цветов и текстур, создание хромовых текстур и их смешение, а также другие эффекты для задания особых свойств объектов в сцене. В сцене демо-версии программы в частности, были применены многие из этих эффектов — при помощи смешения цветов был создан эффект переливающегося реалистичного неба, из хромовых текстур была «сделана» антенна, кажущаяся зеркальной, также, при помощи хромовых текстур было сделано переливающееся оттенками синего море. Эффект прозрачности позволил создать модель высоковольтного столба, и деревьев. Также интересен эффект смешения обычной и хромовой текстуры. Он не был использован в сцене демо-версии, но его использование вполне возможно. В результате получаются полузеркальные объекты (в качестве примера можно привести лакированный платяной шкаф — в нем можно увидеть отражение, но и он сам имеет определенный вид и цвет). Этот спецэффект позволяет создавать наиболее реалистичные объекты, правда их использование немного снижает скорость отображения сцены.

5. Математическое деформирование объектов. При помощи этой возможности, можно создать, например, бурлящее море, развивающийся на ветру флаг, или например создать эффект расходящихся на воде кругов. Такое деформирование осуществляется по определенным математическим законам, и также как и в случае с частицами, при помощи SDK системы можно создать любые другие математические эффекты, просто указав закон их деформации во времени.

6. Разнообразные возможности по навигации в сцене. По сцене можно перемещаться как во всех привычных для VRML браузеров режимах (WALK, SLIDE, EXAMINE, ROTATE) но и в режиме, характерном для современных компьютерных игр, когда при перемещении одновременно используются и клавиатура и мышь. Клавиатура определяет положение в пространстве, а мышь – направление взгляда пользователя.

Данный проект изначально не был направлен на решение какой-либо узкоспециализированной задачи, В принципе, он может быть использован для решения абсолютно любой задачи визуализации. Все зависит лишь от исходных данных. Например, можно создать красивейшую сцену, пестрящую красками и запускать ее в качестве демонстрации внешнего вида какого-либо виртуального мира. Можно наоборот, слегка пренебречь внешним видом, и провести простую оптимизацию сцены – уменьшить размеры текстур, уменьшить количество геометрических образов и т.п. в этом случае получится большой выигрыш в размерах файлов, что может быть очень существенно для работы в сети internet.

Литература

[1] Guide to OpenGL® on Windows®. Silicon Graphics, 1997.

[2] OpenGL® For Windows® From Silicon Graphics® Frequently Asked Question. Silicon Graphics, 1997.

[3] Vu M., Neyder D., Davis T. OpenGL Programmer Guide, Second Edition. Silicon Graphics, 1996.

[4] Valnum K., 3D Graphics Programming with OpenGL, QUE, 1997.

[5] www.gdmag.com. Game developer, 1998.