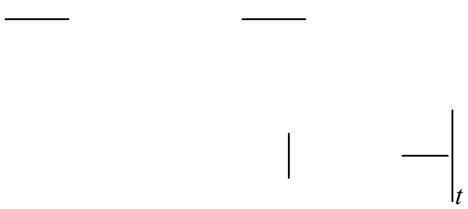

ОБ ОДНОМ ПОДХОДЕ К ПОСТРОЕНИЮ ГРАФИЧЕСКИХ ПРИЛОЖЕНИЙ

Г. А. Ломакин

ВВЕДЕНИЕ

В настоящее время широкое распространение получили синтезированные изображения различных объектов. Так, достаточно хорошо известны примеры научной визуализации, моделирования в промышленности (например, создание прототипов автомобилей), спецэффекты для кинематографа, трехмерные игры и т.д.

Значительный научный и практический интерес представляют также и задачи создания виртуальных реальностей, которые отождествляют реальные прототипы, например, визуализация различных памятников архитектуры и археологии. Как правило, подход к решению такого рода задач требует, в основном, интерактивного компьютерного дизайна.

Экранизация реальных моделей является достаточно сложной задачей. Для экранной визуализации синтезированной 3D-модели используются различные алгоритмы, выбор, адаптация и модификация которых зависит от рассматриваемой задачи и предметной области. Более того, чрезмерная трудоемкость создания объектов реального мира приводит к необходимости разработки специальных моделей и новых методов экранизации.

Отметим также, что при современном развитии графических систем и технологий актуальным является подход, связанный с переводом визуализации 3D-объектов на новый уровень. Разработка 3D-приложений характеризуется определенным набором ограничений, что, естественно, сказывается на результатах визуализации.

Наиболее часто разработчики 3D-приложений обращаются к двум основным библиотекам 3D-графики: DirectX и OpenGL. Однако работа с ними вызывает трудности, связанные, прежде всего, с написанием многих строк кода, использованием API и указателей. Часть проблем по реализации 3D-объектов можно решить с использованием XNA framework [1], применяя соответствующие классы. Однако XNA не позволяет полностью устранить проблемы, возникающие при разработке 3D-приложений, т.к. указанная среда является .NET реализацией DirectX.

Таким образом, создание некоторой обобщенной среды (графического ядра) для визуализации различных 3D-объектов является актуальной задачей. Базовая концепция, лежащая в основе предлагаемого решения, связана с созданием набора классов и утилит, которые реализуют более общие подходы к визуализации графических объектов и скрывают от пользователя непосредственное использование тех или иных математических моделей и графических алгоритмов. Все это позволит более широкому кругу заинтересованных лиц создавать требуемые 3D-объекты и сцены, затрачивая минимальное время на разработку необходимого решения.

СТРУКТУРНАЯ ЧАСТЬ ГРАФИЧЕСКОГО ЯДРА COREX

В качестве оптимального решения для визуализации сложных трехмерных объектов реального и виртуального мира, предлагается графическое ядро CoreX, базирующееся на использовании XNA framework. Отметим, что XNA включает Content Pipeline. Это упрощает работу с контентом и при необходимости позволяет расширить список поддерживаемых типов.

XNA поддерживает программируемый графический контейнер, т.е. класс, реализующий объект. Контейнер содержит файл эффекта, в котором на языке HLSL [1] написан пиксельный и вершинный шейдер, осуществляющий обработку 3D-объекта, преобразование геометрии, размещение в пространстве, текстурирование, расчет освещенности объекта в зависимости от положения источника света и многое другое.

На базе ядра CoreX реализован также и графический интерфейс пользователя (GUI), позволяющий производить интеграцию различных элементов управления в сцене и манипулировать происходящим на ней.

Кроме графической составляющей в CoreX присутствует также и звук.

Графическое ядро CoreX поддерживает DirectX 10, а также шейдерную модель версии 4.0. Реализованы такие эффекты, как: per-Pixel lighting, bump mapping, parallax mapping, shadow mapping, shadow volumes, multisampling [1].

ПОДХОДЫ, ИСПОЛЬЗУЕМЫЕ ПРИ ПОСТРОЕНИИ СИСТЕМЫ ВИЗУАЛИЗАЦИИ ОБЪЕКТОВ

Приложение, построенное на базе графического ядра, представляет собой универсальную утилиту для создания сцены, придания различным объектам требуемых физических свойств, а также задания необходимых взаимодействий объектов. Все это осуществляется благодаря работе с графическими объектами и контентом посредством графического интерфейса и изменения свойств объектов. После построения сцены осуществляется переход в режим наблюдения, в котором можно наблюдать все заданные эффекты, движения, столкновения, освещения и т.д., а также самому взаимодействовать с созданной средой.

Кроме того, пространство имен разработанного ядра CoreX можно применять для создания собственных приложений в качестве графической компоненты, например, при создании компьютерных игр. В данном случае пользователю требуется лишь определиться с контентом и задать игровую логику.

Отметим, что приложения, созданные с использованием CoreX, являются кроссплатформенными. В данный момент поддерживается Windows 7 и Xbox 360. В дальнейшем появится поддержка Windows Phone 7. Более того, так как CoreX является пространством имен в .Net, это позволяет использовать его классы в любом .Net-приложении.

ПРОЦЕСС ВЗАИМОДЕЙСТВИЯ ПОЛЬЗОВАТЕЛЯ С COREX

Как было упомянуто выше, пользователь взаимодействует с ядром при помощи универсальной утилиты. Утилита состоит из двух основных

частей. Первая часть отвечает за контент и объекты, использующие этот контент, а также за свойства и различные характеристики этих объектов. Вторая часть отвечает за настройку ядра CoreX: детализацию, качество текстур, используемые технологии и др.

После конфигурации все данные передаются ядру, после чего запускается визуализация.

Следует отметить, что при задании параметров объектов имеется также возможность предварительного просмотра и взаимодействия с объектами, их свойствами посредством работы в окне предварительного просмотра. Пользователь всегда видит, где он размещает объект, как он расположен относительно других объектов и др.

ОСНОВНЫЕ ВЫВОДЫ

Оптимальная и высокотехнологичная графика открывает новые возможности как для создания пользовательского интерфейса, так и для решения конкретных практических задач. В силу этого, сам процесс визуализации 3D-объектов должен быть максимально упрощен и предоставлять готовые паттерны-решения для разработчика. Это позволит при создании конкретных 3D-приложений акцентироваться на таких важных моментах как дизайн, логика, взаимодействие объектов и т.д. Дальнейшая разработка графического ядра CoreX предполагает добавление нового контента, упрощение процесса взаимодействия с пользователем, расширение класса объектов. Разработка такого рода может найти широкое применение как в научных, так и в игровых и промышленных визуализациях сложных сцен. Такая функциональность актуальна при моделировании отдельных аспектов виртуального мира и явлений окружающей среды.

Литература

1. *Wolfgang E.* Shader X5: Advanced Rendering Techniques / Charles River Media, 2006.