О создании обощенной графической среды для разработки 3D-приложений

Ломакин Г.А.
Кафедра многопроцессорных систем и сетей
Факультет прикладной математики и информатики
Белорусский государственный университет
Минск, Беларусь
e-mail: spellbound.fpmi@gmail.com

***Аннотация* — рассматриваются подходы к созданию обобщенной графической среды, которая позволяет визуализировать трехмерные графические объекты и реализовывать приложения на их основе. Основная идея предлагаемого решения связана с созданием набора классов и утилит, которые реализуют более общие подходы к визуализации и скрывают от пользователя непосредственное использование математических моделей и графических алгоритмов.**

Ключевые слова: 3D-приложение, графическое ядро, фреймворк, контент, графический объект, класс, графический интерфейс пользователя

I. Введение

Большой 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, интерактивной экранизации.

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 предметной области. Более того, чрезмерная трудоемкость создания объектов реального мира приводит к необходимости разработки специальных моделей и новых методов экранизации для таких объектов.

Достаточно часто разработчики трехмерных изображений используют две основные библиотеки: 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-фреймворка [1], используя соответствующие классы. Однако XNA не позволяет полностью устранить проблемы, возникающие при разработке 3D-приложений, т.к. указанный фреймворк является .NET реализацией DirectX.

В связи с вышеизложенным, создание некоторой обобщенной графической среды (графического ядра) для визуализации различных сложных трехмерных объектов является актуальной задачей. Основная идея, положенная в основу предлагаемого решения для графической среды, 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, что, несомненно, достаточно актуально на современном этапе развития трехмерных приложений.

II. О структуре графического ядра

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

Отметим, что графические объекты представляются в качестве классов. Класс каждого графического объекта описывает его основные свойства, а также методы для отображения объекта в пространстве и обновления его состояния. Каждый класс связывается с определенным контентом, в нем же определяется, как данный контент будет использоваться.

XNA поддерживает программируемый графический конвейер, т.е. класс, реализующий графический объект, содержит файл эффекта, в котором на языке HLSL [2] 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 [3], parallax mapping, shadow mapping, shadow volumes, multisampling.

III. О построении обобщенной графической среды

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.

Отметим, что пространство имен разработанного 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.

Приложения, созданные с использованием CoreX, являются кроссплатформенными, что, несомненно, является большим достоинством предлагаемой обобщенной графической среды.

В настоящий момент предлагаемое графическое ядро поддерживается Windows 7 и Xbox 360, разрабатывается поддержка Windows Phone 7. Более того, так как CoreX является пространством имен в .Net, это позволяет использовать его классы также и в любом .Net-приложении.

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 т.п. По окончании конфигурации обобщенной системы, все данные передаются ядру, после чего запускается непосредственно визуализация.

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

IV. КРАТКИЕ ИТОГИ

Оптимальная и технологичная графика открывает новые возможности как для создания пользовательского интерфейса, так и для решения конкретных практических задач. Поэтому процесс визуализации трехмерных объектов должен быть максимально упрощен и предоставлять готовые паттерны-решения для разработчика. При разработке и создании конкретных 3D-приложений это позволит уделять достаточное внимание таким важным аспектам как дизайн, логика, взаимодействие объектов и т.д. В дальнейшем предполагается разработка графического ядра CoreX путем добавление нового контента, упрощения процесса взаимодействия с пользователем, расширения класса объектов. Несомненно, разработка обобщенной графической среды найдет широкое применение как в научных, так в сцено-игровых и промышленных визуализациях, что является актуальным при моделировании отдельных аспектов виртуального мира и явлений окружающей среды.

[1] Введение в XNA [Electronic resource] – Mode of access: http://www.intuit.ru/department/se/intxna/1/.

[2] Программирование шейдеров на HLSL. [Electronic resource] – Mode of access: http://www.gamedev.ru/code/articles/HLSL.

[3] Создание карт нормалей из фотографии [Electronic resource] – Mode of access: http://www.render.ru/books/show_book.php?book_id=765.