Система материалов и эффектов для реалистичной визуализации трехмерных сцен

А.Н. Артиков, Т.Н. Артиков
Новосибирский Государственный Университет
Институт автоматики и электрометрии СО РАН, Новосибирск, Россия
{a.artikov, t.artikov}@gmail.com

Аннотация

В работе предлагается способ реализации материалов и эффектов для систем визуализации реального времени. Ключевыми особенностями данного решения являются использование автоматизированной генерации шейдеров для материалов и представление эффектов в виде графов фильтров, обрабатывающих изображение. Большая часть расчетов, связанных с визуализацией, выполняется на графическом ускорителе, что обеспечивает высокую производительность системы.

Ключевые слова: *реалистичная визуализация, подготовка шейдеров, обработка изображений на видеокарте.*

1. ВВЕДЕНИЕ

В современных системах, предназначенных для визуализации интерактивных трехмерных сцен, важным требованием является реалистичность визуализации. Эти системы применяются в виртуальных тренажерах, компьютерных играх, различных обучающих демонстрациях, приложениях научной визуализации. Для достижения реалистичности визуализация сцены осуществляется с учетом материалов объектов (визуальных свойств поверхности), а также применяются эффекты обработки изображений [1].

Видеокарты предоставляют широкие возможности для управления конвейером визуализации с помощью шейдеров [8]. Материал объекта задается функцией, выполняемой на графическом процессоре. В ней происходит текстурирование объекта, расчет освещения, микрорельефа, прозрачности и других свойств. Эффекты обработки изображений также представляют собой функции, вычисляемые на видеокарте, но, в отличие от материалов, эффекты можно комбинировать. Для этого выходы одних эффектов подаются на входы другим. Эффекты могут быть применены как к изображению, полученному при визуализации сцены, так и к входным изображениям (например, к изображениям, считанным из видео-файлов).

Существующие системы материалов разрабатываются для использования с конкретными типами сцен (ландшафты, интерьеры) и ограничены заранее заготовленными свойствами и эффектами. В отличие от них, разработанная авторами система материалов и эффектов лишена этих недостатков.

2. МАТЕРИАЛЫ

Материалом трехмерного объекта будем называть совокупность визуальных свойств его поверхности. К этим свойствам относятся цвет поверхности, рельефность, прозрачность, самосвечение, способность объекта отражать и преломлять свет.

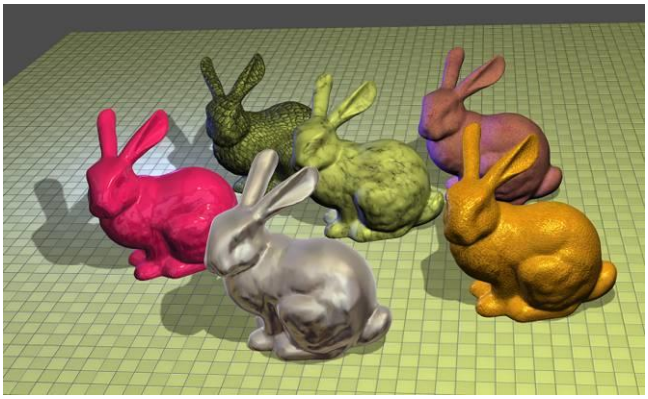

Рис 1: Один и тот же объект с различными материалами

В системах визуализации свойства материала задаются вещественными числами и текстурами. Во время визуализации видеокарта вычисляет изображение объекта с помощью некоторого шейдера на основе информации о геометрии объекта, его материала и свойств сцены (освещение, туман). В зависимости от того, какие свойства имеет материал и сцена, нужно использовать тот или иной шейдер. Количество различных свойств, поддерживаемых системой, велико (несколько десятков), соответственно количество возможных сочетаний свойств составляет несколько тысяч, т.е. написание всех необходимых шейдеров вручную не представляется возможным. Для решения этой задачи в системе реализована автоматизированная генерация кода шейдеров. Генерация кода осуществляется при помощи комбинации двух подходов: условная компиляция и высокоуровневые шейдеры.

Условная компиляция основана на использовании директив препроцессора для генерации кода шейдеров. Ниже приведен пример директивы препроцессора, в котором в зависимости от наличия у материала диффузной текстуры, осуществляется либо считывание из этой текстуры, либо берется цвет, заданный вектором вещественных чисел.

```
#ifdef  HAS_DIFFUSE_MAP
    diffuse = tex2D(diffuseMap, tc);
#else
    diffuse = diffuseColor;
#endif
```

Шейдер, содержащий директивы препроцессора, называется убершейдером [4]. Один и тот же убершейдер компилируется несколько раз с разными флагами компиляции, в результате чего получаются разные шейдеры. Помимо булевых флагов так же реализована поддержка целочисленных констант, задаваемых на этапе компиляции, что может быть использовано для задания количества источников света. За счет использования условной компиляции код шейдера

получается максимально производительным: в нем отсутствуют лишние вычисления, считывания из текстур и операции условного перехода, а границы циклов имеют константные значения, за счет чего циклы лучше оптимизируются компилятором. Однако чтобы сделать систему расширяемой, одного этого подхода недостаточно.

Необходима возможность добавлять в систему новые способы расчета характеристик поверхности объекта и новые модели освещения. Для этой цели предлагаются высокоуровневые шейдеры: шейдер поверхности и шейдер освещения. В шейдере поверхности вычисляются следующие характеристики поверхности: диффузный и зеркальный цвет, прозрачность, нормаль, самосвечение.

Пример шейдера поверхности, в котором цвет поверхности берется из текстуры:

```
SurfaceOutput surface(SurfaceInput in)
{
   SurfaceOutput out;
   out.diffuse = tex2D(diffuseMap, in.tc);
   out.normal = in.normal;
   out.specular = float3(1.0, 1.0, 1.0);
   out.emission = float3(0.0, 0.0, 0.0);
   return out;
}
```

Шейдер освещения описывает некоторую модель освещения, то есть то, как поверхность взаимодействует с источниками света. Этот шейдер принимает на вход нормаль поверхности, направление взгляда и на источник света, а возвращает силу диффузной и зеркальной составляющих освещения.

Пример шейдера освещения по Фонгу:

```
float sP : SPECULAR_POWER;
LightingOutput lighting(LightingInput in)
{
   LightingOutput out;
   float3 R = reflect(in.lDir, in.normal);
   out.diffK = max(0, dot(in.normal, -in.lDir));
   out.specK = pow(max(0, dot(R, -in.vDir)), sP);
   return out;
}
```

На основе высокоуровневых шейдеров генерируется код вершинного и пиксельного шейдеров по определенному шаблону, в котором реализована поддержка нескольких источников света, теней и тумана. В коде высокоуровневых шейдеров могут содержаться директивы препроцессора, т.е. получаемые из них шейдеры являются убершейдерами. Далее убершейдер подается в модуль условной компиляции. Этот модуль выставляет нужные флаги и производит шейдер, с помощью которого будет осуществляться визуализация объекта.

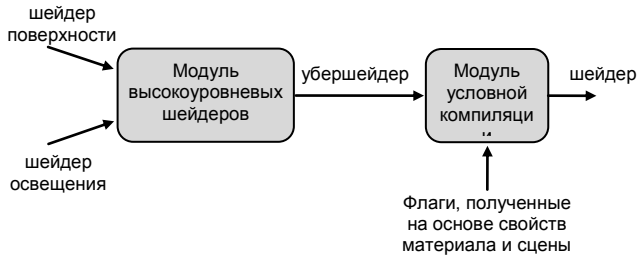

Рис 2: Генерация шейдеров

Для добавления новых возможностей пишутся только высокоуровневые шейдеры, весь остальной код шейдеров генерируется системой. Шейдеры поверхности могут комбинироваться с любым из шейдеров освещения, что дает широкие возможности для настройки внешнего вида объектов. Условная компиляция позволила повысить производительность визуализации в 1.5-2 раза по сравнению с использованием шейдеров, в которых вместо директив препроцессора применяются операции условного перехода. С помощью описанной системы был реализован стандартный материал, поддерживающий до 8 различных текстурных карт и использующий освещение по Фонгу. Так же реализованы трипланарное текстурирование [9] и модель освещения Кука-Торренса [6].

3. ЭФФЕКТЫ

Эффекты в системе – это преобразования, которые можно назначить на изображения. В простейшем случае эффект представляет собой фильтр – алгоритм, принимающий на вход несколько изображений и возвращающий одно в качестве результата. Для обеспечения высокой производительности алгоритмы фильтров реализуются для графических ускорителей с использованием пиксельных шейдеров [2].

Выходы одних фильтров можно подавать на входы другим, соединяя их в ориентированный граф, что позволяет реализовать более сложные эффекты из нескольких простых. Процесс обработки изображений графом фильтров заключается в последовательном применении фильтров в порядке, определяемом зависимостями по данным.

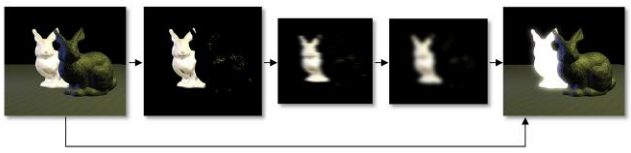

Рис 3: Граф фильтров для эффекта «сияние»

Эффекты, назначенные на окно вывода сцены, могут использовать не только изображение сцены, но и дополнительные изображения, в пикселях которых хранится некоторая информация о сцене: расстояние до камеры, нормаль к поверхности, позиция в пространстве. Эти изображения получаются путем повторной визуализации объектов сцены, при этом пиксельные шейдеры материалов заменяется специальным шейдером, который вместо цвета возвращает нужную информацию о сцене.

Кроме этого, реализована поддержка временной фильтрации изображений – фильтр может принимать на вход несколько последовательных кадров одного динамического изображения. Данная возможность необходима для эффектов обработки видео.

Эффекты в системе описываются на декларативном скриптовом языке. Формат описания позволяет задать фильтры, содержащиеся в эффекте, указать каждому фильтру пиксельный шейдер, соединить выходы одних фильтров с входами других, задать свойства выходных изображений фильтров – абсолютный или относительный размер, цветовое пространство, формат пикселей.

Приведем пример описания для эффекта «сияние»:

```
EFFECT.glow = {
    PASS = {
        brightPass = SHADER.brightPass {
            img = IN
        }
        blurX = SHADER.blurX {
            img = PASS.brightPass
            OUT.SCALE = (0.5, 0.5)
        }
        blurY = SHADER.blurY {
            img = PASS.blurX
            OUT.SCALE = (0.5, 0.5)
        }
        final = SHADER.add {
            img1 = IN
            img2 = PASS.blurY
        }
    }
    OUT = final
}
```

Система считывает описание эффекта и строит по нему граф фильтров. На основе топологии графа автоматически определяется порядок выполнения фильтров, а так же последовательность создания и удаления временных изображений, необходимых для хранения промежуточных результатов вычислений.

С помощью разработанной системы реализованы эффекты обработки видео: преобразование форматов, устранение чересстрочности (deinterlacing), выделение актера на монохромном фоне [3]; а так же эффекты постобработки изображений трехмерных сцен: сияние, имитация рисунка карандашом, глубина резкости [7], туман, глобальное освещение [10] и др.

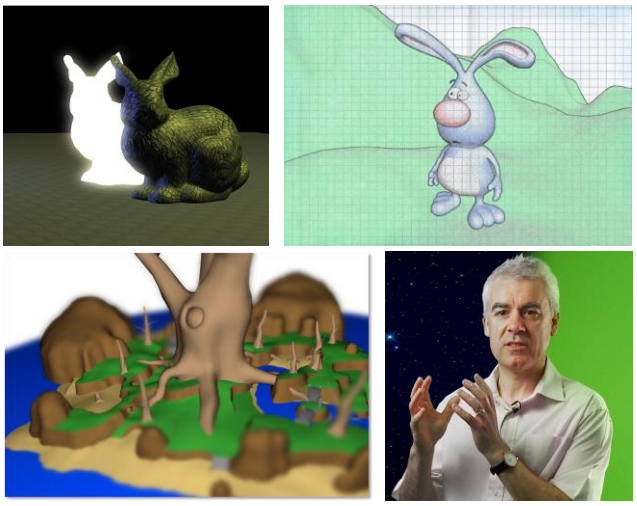

Рис 4: Примеры эффектов

4. ЗАКЛЮЧЕНИЕ

Авторами разработаны подходы для реалистичной визуализации трехмерных сцен и реализована расширяемая система материалов и эффектов, позволяющая достичь высокого качества визуализации. Система обладает высокой производительностью и может быть использована для широкого класса интерактивных графических приложений. С помощью системы разработан набор примеров материалов и эффектов, демонстрирующих её возможности.

5. ССЫЛКИ

[1] Городилов М. А., Коростелев Е. И. Программируемые материалы и спецэффекты для систем визуализации реального времени. // Труды конференции МНСК-2011. 2011.

[2] Дьячков В. Реализация процессора эффектов постобработки. // Статья на сайте www.uraldev.ru. 2008.

[3] Ковальков М.А. Разработка и реализация алгоритма рирпроекции на базе современного графического акселератора. // Труды конференции Графикон-2006. 2006.

[4] Сладков Д. Промышленные убершейдеры. // Труды конференции КРИ-2008. 2008.

[5] Akenine-Moller T. Real-time rendering. 2008.

[6] Cook R., Torrance K. A Reflectance Model for Computer Graphics. 1982.

[7] Hammon E. Practical Post-Process Depth of Field. // GPU Gems 3 Chapter 28. 2008.

[8] Kessenich J. The OpenGL® Shading Language. 2010.

[9] Nicholson K. GPU Based Algorithms for Terrain Texturing. 2008.

[10] Ritschel T. Approximating Dynamic Global Illumination in Image Space. 2009.

Об авторах

Артиков Артиков Неъматжанович – студент факультета информационных технологий Новосибирского Государственного Университета, инженер-программист лаборатории синтезирующих систем визуализации Института Автоматики и Электрометрии СО РАН.
e-mail: a.artikov@gmail.com

Артиков Тимур Неъматжанович – студент факультета информационных технологий Новосибирского Государственного Университета, инженер-программист лаборатории синтезирующих систем визуализации Института Автоматики и Электрометрии СО РАН.
e-mail: t.artikov@gmail.com