Лекция №9

Пример использования шейдеров
Алгоритм трассировки луча
Тени
Отражения
Преломления

Рябинин Константин Валентинович

e-mail: icosaeder@ya.ru
jabber: icosaeder@jabber.ru

Пермь, 2011

Встроенные переменные в GLSL

В GLSL ряд переменных передаются в шейдер неявно (автоматически). Такие переменные называются встроенными

Основные из них:

- **Входные**
  - **vec4 gl_Vertex – координаты вершины**
  - **vec3 gl_Normal – координаты нормали**
  - **vec4 gl_Color – цвет вершины**
  - **vec4 gl_MultiTexCoordX – координаты мультитекстура для юнита №X**
  - **mat4 gl_ModelViewMatrix – матрица MODELVIEW**
  - **mat4 gl_ModelViewProjectionMatrix – результат умножения PROJECTION x MODELVIEW**
  - **mat3 gl_NormalMatrix – матрица преобразования нормалей (транспонированный обратный первый минор 3x3 матрицы MODELVIEW)**
  - **struct gl_FrontMaterial – структура материала передней стороны полигона**
  - **struct gl_LightSource[X] – структура источника света №X**

Встроенные переменные в GLSL

В GLSL ряд переменных передаются в шейдер неявно (автоматически). Такие переменные называются встроенными

Основные из них:

- **Выходные**
  - **vec4 gl_Position – координаты вершины**
  - **vec4 gl_FragColor – цвет фрагмента (только для фрагментного шейдера)**
  - **float gl_FragDepth – значение глубины, которое должно быть записано в z-буфер (только для фрагментного шейдера)**
  - **vec4 gl_TexCoord[X] – текстурные координаты для мультитекстурного юнита №X**

Реализация освещения на GLSL

- **Освещение, реализованное через шейдеры, может быть повершинным (per-vertex lighting) и пофрагментным (per-fragment lighting)**
- **Более качественный результат достигается пофрагментным освещением, так как выше дискретизация расчётов (по аналогии с закраской Фонга)**

Пример:

- **Математическая модель:**

$$I = A_{obj} + \cos(\widehat{\bar{n}\ \bar{l}}) \cdot D_{obj} \cdot D_{light} + \cos^{\frac{1}{S}}(\widehat{\bar{r}\ \bar{e}}) \cdot S_{obj} \cdot S_{light}$$

I – итоговый цвет

A_{obj} – цвет окружающей подсветки объекта

D_{obj} – цвет диффузного отражения объекта

D_{light} – цвет диффузного освещения источника света

S – величина зеркального блика

S_{obj} – цвет зеркального блика объекта

S_{light} – цвет зеркального освещения источника света

$\bar{n}$ – нормаль к поверхности объекта

$\bar{l}$ – вектор падения света

$\bar{r}$ – вектор отражения света

$\bar{e}$ – вектор взгляда

Реализация освещения на GLSL

- **Вершинный шейдер:**

```
varying vec3 v_normal;
varying vec3 v_eye;
varying vec3 v_ray;

void main()
{
    vec3 vertex = vec3(gl_ModelViewMatrix * gl_Vertex);
    vec3 normal = vec3(gl_NormalMatrix * gl_Normal);
    vec3 light = vec3(gl_LightSource[0].position);

    v_normal = normalize(vertex - normal);
    v_eye = normalize(-vertex);
    v_ray = normalize(vertex - light);

    gl_Position = ftransform();
}
```

Реализация освещения на GLSL

- **Фрагментный шейдер:**

```
varying vec3 v_normal;
varying vec3 v_eye;
varying vec3 v_ray;

#define SHININESS 5.0

void main()
{
    vec3 reflection = reflect(v_ray, v_normal);
    float diffuse = max(dot(v_normal, v_ray), 0.0);
    float specular = pow(max(dot(reflection, v_eye), 0.0), SHININESS);

    vec4 color = gl_FrontMaterial.ambient +
                 diffuse * gl_FrontMaterial.diffuse *
                 gl_LightSource[0].diffuse +
                 specular * gl_FrontMaterial.specular *
                 gl_LightSource[0].specular;

    gl_FragColor = vec4(color.rgb, 1.0);
}
```

Текстурирование на GLSL

- **Вершинный шейдер:**

```
void main()
{
    gl_TexCoord[0] = gl_MultiTexCoord0;
    gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}
```

- **Фрагментный шейдер:**

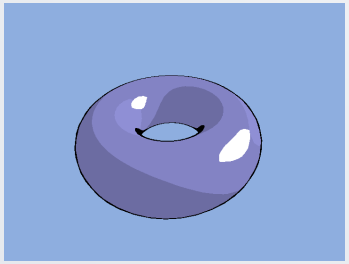
```
uniform sampler2D u_map;

void main()
{
    gl_FragColor = texture2D(u_map, vec2(gl_TexCoord[0]));
}
```

→ **Действия по загрузке и связыванию текстуры производятся обычным образом; дискретизатор u_map устанавливается в программе при помощи вызова glUniform1i(uMap, texUnit),**
где uMap – локация, полученная вызовом glGetUniformLocation(shaderProgram, "u_map"),
а texUnit – номер используемого текстурного юнита

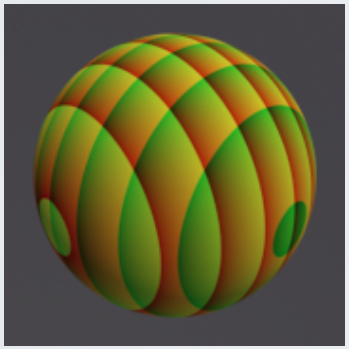
Другие эффекты на GLSL

Для самостоятельного изучения рекомендованы эффекты:

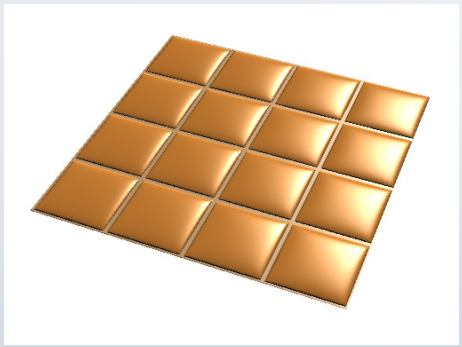
- Мультипликационное окрашивание (toon shading)



- Процедурное текстурирование (текстура генерируется на шейдере)



- Бамп-мэппинг (искривление поля нормалей на основе данных из текстуры)



Фотонная карта

Фотонная карта (***photon map***) **– это множество траекторий фотонов, излучённых источниками света на сцене**

- **Используется для построения фотореалистичных изображений**
- **Итоговое изображение формируется фотонами, попавшими на проекционную плоскость (например, на сетчатку глаза наблюдателя)**
- **Изначальный цвет фотона задаётся источником света, а затем корректируется через столкновения с объектами**
- **Построение фотонной карты требует колоссальных затрат по времени и памяти, причём порождает множество «холостых» фотонов, не попавших на проекционную плоскость**

Трассировка луча

Алгоритм трассировки луча (*ray tracing*) **– это универсальный алгоритм для получения фотореалистичного изображения**

Особенности:

- **Высокое качество и фотореализм результата**
- **Универсальность алгоритма**
- **Единообразное моделирование всех оптических свойств объектов и оптических эффектов**
- **Прозрачность идеи и лёгкость реализации**
- **Высокая вычислительная сложность; непригодность для использования в системах реального времени**
- **Хорошая распараллеливаемость**

Трассировка луча

- В реальном мире на сетчатку глаза наблюдателя попадают лучи света, которые были порождены некоторым источником и прошли некоторый путь, отражаясь и преломляясь
- Задача алгоритма – определить путь, проделанный каждым лучом света
- Идея алгоритма – проследить движение света в обратном направлении (то есть от наблюдателя к сцене, а не наоборот)
- Основные входные данные для алгоритма:
  - Разрешение изображения, которое должно получиться в итоге
  - Объекты трёхмерной сцены:
    - Положение
    - Геометрия
    - Цвет объекта в каждой точке
    - Коэффициент отражения в каждой точке
    - Коэффициент прозрачности в каждой точке
    - Коэффициент преломления в каждой точке
  - Источники света сцены:
    - Положение
    - Геометрия
    - Цвет
    - Интенсивность

Трассировка луча

- Реализация:
  - Через каждую точку изображения от наблюдателя к сцене запускается луч
  - Цвет точки определяется цветом луча
  - Цвет луча определяется на основании оптических свойств объектов, с которыми этот луч столкнулся
  - В результате столкновения в заданной точке возможны ситуации:
    - Объект поглощает луч
    - Объект отражает луч
    - Объект преломляет луч
    - Объект расщепляет луч на отражённый и преломлённый
  - В каждом случае цвет луча корректируется в соответствии с цветом поверхности и принятыми правилами корректировки

Моделирование теней

Затенение (*umbra*) – неосвещённый участок поверхности объекта, освещению которого препятствует его ориентация (нормаль к поверхности образует с вектором падения света тупой угол)

Тень (*shadow*) – неосвещённый участок поверхности объекта, освещению которого препятствует другой объект, находящийся на пути падения света

- В реальном мире природа этих явлений одинакова
- Однако при моделировании этих эффектов используются совершенно различные механизмы, причём механизмы моделирования тени значительно сложнее и в алгоритмическом, и в вычислительном плане

Моделирование теней

Жёсткая тень (***hard shadow***) **– это тень, имеющая резкие границы**

Мягкая тень (***soft shadow***) **– это тень, имеющая размытые границы**

- **В реальном мире все тени являются мягкими**
- **Однако моделирование мягких теней зачастую более сложная задача, как правило, сводимая к получению жёсткой тени и применению к ней фильтра размытия**
- **Ранее все моделируемые в реальном времени тени были исключительно жёсткими, так как оборудование просто не позволяло получать мягкие тени с необходимой скоростью**
- **На современном оборудовании мягкие тени в реальном времени не представляют особой проблемы и активно используются**

Моделирование теней

Наиболее простой способ моделирования теней – использование заранее подготовленной текстуры, которая накладывается на объекты и уже содержит в себе «тёмные пятна» теней

- **Нет никаких затрат на вычисление теней в реальном времени**
- **Допустимо использовать только в случае статических объектов и статических источников света**
- **Проблема состоит в подготовке соответствующих текстур – как правило для этого используются программы трёхмерного моделирования**

Моделирование теней

В ряде случаев тень от объекта может быть представлена спрайтом

- Спрайт масштабируется и позиционируется в зависимости от взаимного расположения источника света и объекта
- На спрайт накладывается заранее подготовленная текстура, приближённо совпадающая с контуром объекта, который получается путём проектирования объекта на плоскость из точки источника света
- Подход приемлем только в том случае, если тень падает на ровную поверхность, так как спрайт всегда остаётся плоским (чаще всего используется для имитации тени от персонажа, перемещающегося по некоторому ровному субстрату)
- Для упрощения текстурой может быть ортографический контур объекта
- О физической точности и высоком качестве результата моделирования не может быть и речи

Моделирование теней

Значительно более хороших результатов можно добиться, используя карту теней (*shadow mapping*)

- Выполняется рендеринг сцены из источника света
- Данные из z-буфера сохраняются в виде текстуры
- При рендеринге сцены из камеры осуществляется проектирование подготовленной текстуры из позиции источника света на сцену. Таким образом, для каждой точки сцены можно узнать соответствующее значение z-буфера из первого рендеринга (оно будет равно значению в точке на проекции текстуры)
- Для каждой точки сцены, видимой из камеры, определяется расстояние до источника света. Если это расстояние больше соответствующего данной точке значения z-буфера первого рендеринга, значит тока находится в тени (на пути света находится какой-то объект, данные о глубине которого и были сохранены в z-буфере). Иначе тока освещена
- Подход порождает жёсткие тени
- Подход требует двух проходов рендеринга
- Ввиду ограниченной точности z-буфера высока вероятность возникновения артефактов изображения
- Для уменьшения артефактов, в первом проходе объекты рисуются с отсечением передних граней. Это приводит к тому, что в z-буфер записываются только глубины тех граней объекта, которые находятся дальше от источника света, а они в тени по определению, так что для них результат сравнения расстояния не играет роли

Моделирование теней

Одним из наиболее популярных методов моделирования теней является метод
теневых объемов / трафаретных теней
(*shadow volumes / stencil shadows*)

- Для каждого объекта конструируется его теневой объём, то есть область пространства, которая затенена данным объектом
- При рендеринге определяется, попадает ли данная точка в какой-либо теневой объём, и в зависимости от этого определяется её освещённость
- Использовать чисто геометрические расчёты попадания в теневой объём крайне неэффективно, поэтому на практике для отделения освещённых точек от затенённых используют буфер трафарета
- Буфер трафарета (*stencil buffer*) – это матрица, размерность которой совпадает с размерностью буфера кадра, а элементами являются числа, фактическое назначение которых может быть определено программистом
- Теневые объёмы конструируются как полигональные модели «бесконечной» длины (длины, заведомо большей, чем протяжённость сцены), которые затем визуализируются на ряду с объектами сцены
- Во время этого «предварительного» рендеринга определяется, какие точки сцены попали внутрь теневого объёма, и для них устанавливается определённое значение в буфере трафарета (для такого определения существует целый ряд методик)
- Затем сцена визуализируется без теневых объёмов, а освещённость точке определяется на основе соответствующих им значений в буфере трафарета

Моделирование теней

Одним из наиболее популярных методов моделирования теней является метод
теневых объемов / трафаретных теней
(*shadow volumes / stencil shadows*)

- Подход (без каких-либо модификаций) порождает жёсткие тени
- Подход требует минимум двух проходов рендеринга
- Нетривиальной задачей является конструирование самих теневых объёмов

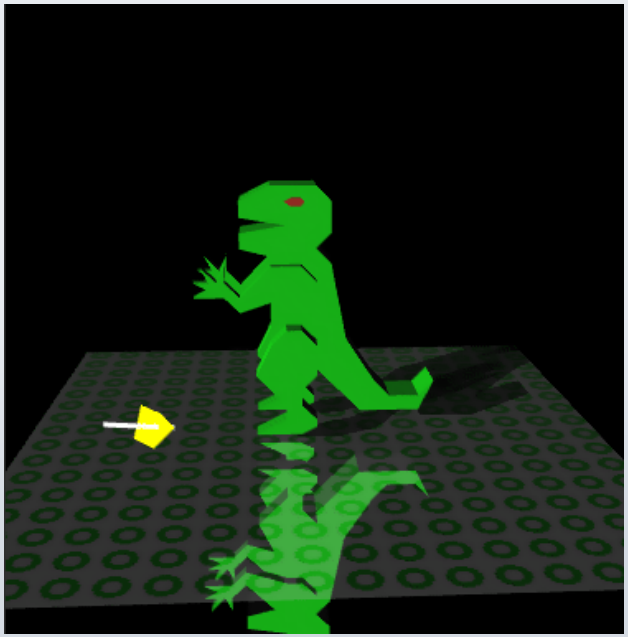
→ Более подробно о методах создания теней можно узнать по ссылкам

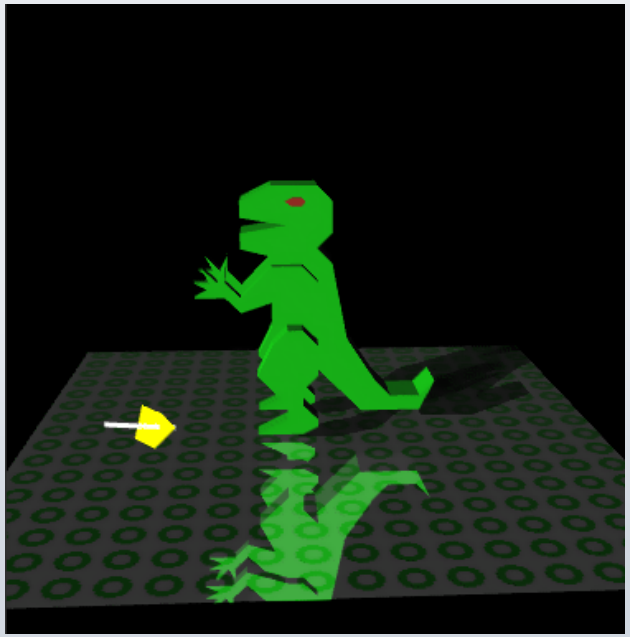
- Карты теней:
  - http://www.paulsprojects.net/tutorials/smt/smt.html
  - http://www.gamedev.ru/code/articles/ShadowMapGLSL
  - http://www.gamedev.ru/code/articles/?id=4177
  - http://code.google.com/p/gl33lessons/wiki/Lesson06
- Теневые объёмы:
  - http://joshbeam.com/articles/stenciled_shadow_volumes_in_opengl/
  - http://nehe.gamedev.net/tutorial/shadows/16010/

Моделирование отражений

Отражение – появление на поверхности объекта изображения окружающей сцены

- В ряде случаев для моделирования отражения достаточно:
  - Выполнить один рендеринг сцены из камеры, устанавливая для отражающего объекта значения в буфере трафарета
  - Преобразовать матрицу проекции соответствующим образом
  - Выполнить рендеринг окружения (без отражающего объекта), записывая (с учётом альфа-смешивания) в буфер цвета значения только для точек, имеющих ненулевые значения в буфере трафарета





Моделирование отражений

Отражение – появление на поверхности объекта изображения окружающей сцены

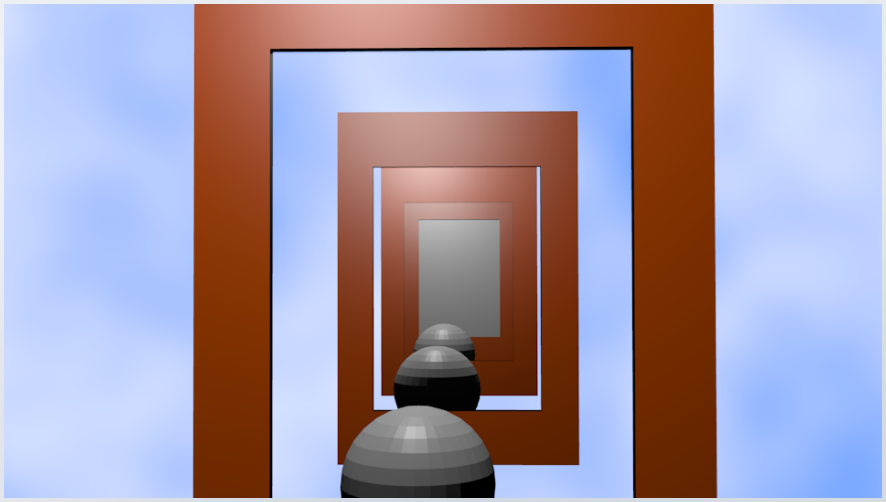
- В ряде случаев для моделирования отражения достаточно:
  - Выполнить один рендеринг сцены из камеры, устанавливая для отражающего объекта значения в буфере трафарета
  - Преобразовать матрицу проекции соответствующим образом
  - Выполнить рендеринг окружения (без отражающего объекта), записывая (с учётом альфа-смешивания) в буфер цвета значения только для точек, имеющих ненулевые значения в буфере трафарета
- Такой метод, однако, пригоден только для плоских отражающих поверхностей, причём, при условии отсутствия множественных отражений

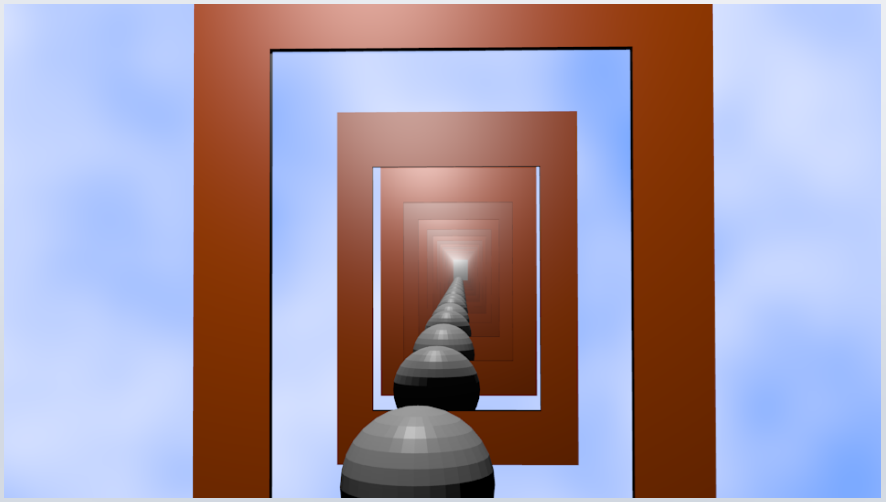
Моделирование отражений

Множественное отражение – это ситуация, когда в отражающем объекте отражается другой отражающий объект

- Возникает «проблема двух зеркал» (ситуация, когда два зеркала стоят напротив друг друга, а наблюдатель стоит между ними)

- Адекватно проблема множественных отражений решается только использованием трассировки луча (или схожих по принципу действия алгоритмов)
- Даже при использовании трассировки луча, отражения не могут вычисляться бесконечно: в алгоритме присутствует параметр максимальной глубины самоотражений





Моделирование отражений

Для моделирования отражения на произвольной поверхности, используется карта среды

Карта среды (*environment map*) – это текстура, содержащая в себе изображение окружающей объект сцены

- **Карты среды бывают сферические (одно иображение) или основанные на гранях (тетраэдрическая, кубическая, октаэдрическая, ...). Наиболее популярна кубическая карта**
- **Камера помещается в цента отражающего объекта и выполняется максимум шесть рендерингов в (во всех ортогональных направлениях)**
- **Из полученных шести (или менее) изображений формируется текстура, которая и накладывается на отражающий объект**
- **Как правило, используется меньше шести изображений: не имеет смысла создавать текстуру отражения для частей объекта, которые не видны из точки положения наблюдателя**
- **Проблему двух зеркал данный подход не решает, так как даёт возможность создать только отражение единичной глубины**
- **При наложении карты среды на объект используются данные о кривизне поверхности, на основе которых можно сгенерировать текстурные координаты, плавно искажающие текстуру и имитирующие реалистичные оптические эффекты кривого зеркала**
- **Для работы с картами среды (в частности, с кубической картой) в библиотеках визуализации есть соответствующие средства**

Моделирование преломлений

Прозрачность – это ситуация, когда сквозь объект частично видна закрываемая им часть сцены

- В самом простом случае прозрачность моделируется при помощи альфа-смешивания и требует лишь сортировки объектов по удалённости

Преломление – это искажение прозрачным объектом видимой сквозь него части сцены, основанное на кривизне его поверхности и на свойствах его материала

- Для моделирования преломлений без трассировки луча также используется карта среды
- Проблема «двух зеркал» видоизменяется, но не теряет актуальности (два отражающе-преломляющих объекта в ряд)
- Основная идея при наложении карты среды для моделирования преломления – корректный учёт оптических коэффициентов при генерации текстурных координат
- Библиотеки визуализации предоставляют генераторы текстурынх координат, принимающие на входе кубическую карту и направление выборки из неё в заданной точке
- Направление выборки считается через коэффициенты преломления
- Работа с картами среды эффективно выполняется на шейдерах

Моделирование каустики

Каустика – это сконцентрированные на небольшой площади лучи света, за счёт которых в пределах этой площади повышается яркость объектов

- Каустика – один из наиболее сложных для моделирования оптических эффектов, но вместе с тем этот эффект резко повышает реалистичность изображения
- В ряде случаев каустика может быть смоделирована спрайтом (по аналогии с тенью)
- Чаще всего для моделирования каустики используются фотонные карты
- Для моделирования каустики в реальном времени может быть использована фотонная карта в пространстве изображения (*Image Space Photon Mapping*) – без построения лучей света на сцене и поиска геометрических пересечений луча с трёхмерными объектами
- Однако даже с использованием такого подхода для получения качественного результата необходимо произвести несколько промежуточных визуализаций
- Бонусом при использовании фотонной карты является одновременное получение и тени от объекта

Моделирование каустики

- Более подробная информация о моделировании таких оптических свойств, как преломление и каустика, может быть найдена в статье: *Моделирование стеклянных поверхностей* – http://www.uraldev.ru/articles/id/39/page/1