Driver-Inter Ltd.

Практические задания по курсу компьютерной графики

Индивидуальные задания для группы 4057/2

Владимир Беляев, Наталья Смирнова
27.08.2009

Задание № 1. **Иерархическая анимация**

Иерархическая анимация с построением дерева узлов и расчетом (например, рекурсивным) матриц.

Внимание! В задании обязательно должна присутствовать структура, позволяющая создавать любые иерархические модели. Рекомендуется структура типа Node, состоящая из:

- матрицы
- функции-callback'а, вычисляющего эту матрицу в зависимости от времени
- указателя на геометрию

Обход иерархии производится сверху вниз. При этом функция callback вычисляет локальную матрицу. С помощью стека матриц получаем мировую матрицу, и с ней рендерим объект.

Варианты объектов с иерархической анимацией представлены ниже. Задания выдаются индивидуально. **Самостоятельный выбор задания запрещен!**

1.1 Птица

Состоит из туловища, крыльев (каждое крыло из 3х частей) и хвоста.

Птица машет крыльями и вращает хвостом. Она при этом летает по кругу над ландшафтом с деревьями из четвертого задания. Поворот хвоста осуществляется вокруг оси, параллельной продольной оси тела птицы.

Иерархия:

Туловище

- правое крыло внутренняя часть
  - правое крыло средняя часть
    - правое крыло внешняя часть
- левое крыло внутренняя часть
  - левое крыло средняя часть
    - левое крыло внешняя часть
- хвост

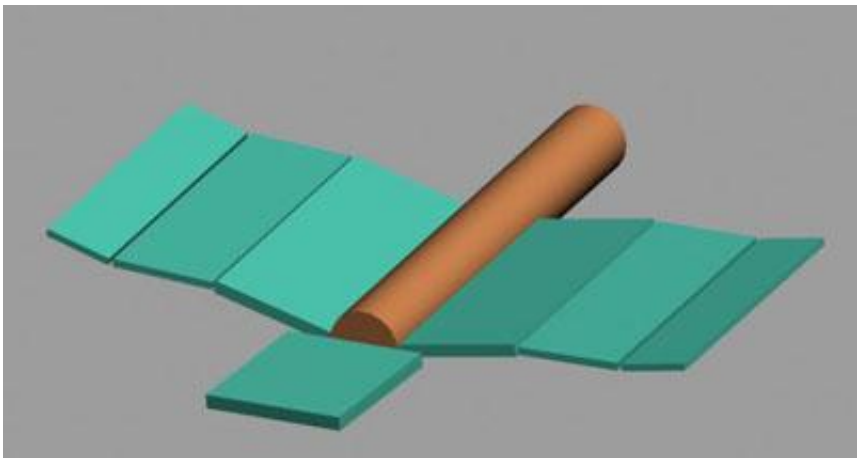

Рис. 1-1: Птица.

1.2 Рука робота

Основание поворачивается вокруг вертикальной оси. Сама рука изгибается. Пальцы хватают.

Иерархия:

Опора + Шар

- Сочленение 1 (цилиндр + шар)
  - Сочленение 2 (цилиндр + шар)
    - Левый палец базовая фаланга (цилиндр + шар)
      - Левый палец конечная фаланга (цилиндр)
    - Правый палец базовая фаланга (цилиндр + шар)
      - Правый палец конечная фаланга (цилиндр)

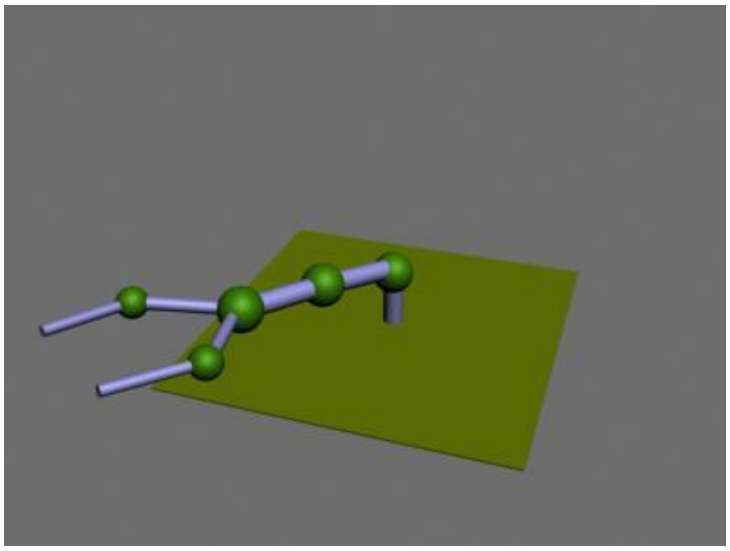

Рис. 1-2: Рука робота.

1.3 Маятник

Маятник состоит из цилиндров и сфер, совершает колебания (Рис. 1-3). В центре последней сферы находится точечный источник света, который совершает движения вместе

с этой сферой. Под маятником необходимо поместить еще какой-либо объект (например, плоскость), чтобы было видно движение этого источника света.

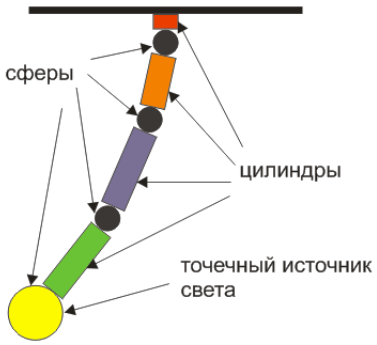

Рис. 1-3: Колеблющийся маятник

1.4 Распускающийся цветок

Цветок состоит из N лепестков, каждый лепесток из двух частей, каждая из которых, в свою очередь, из двух треугольников. Цветок раскрывается и закрывается. При этом каждый следующий лепесток отгибается (закрывается) с небольшим опозданием относительно предыдущего соседнего лепестка.

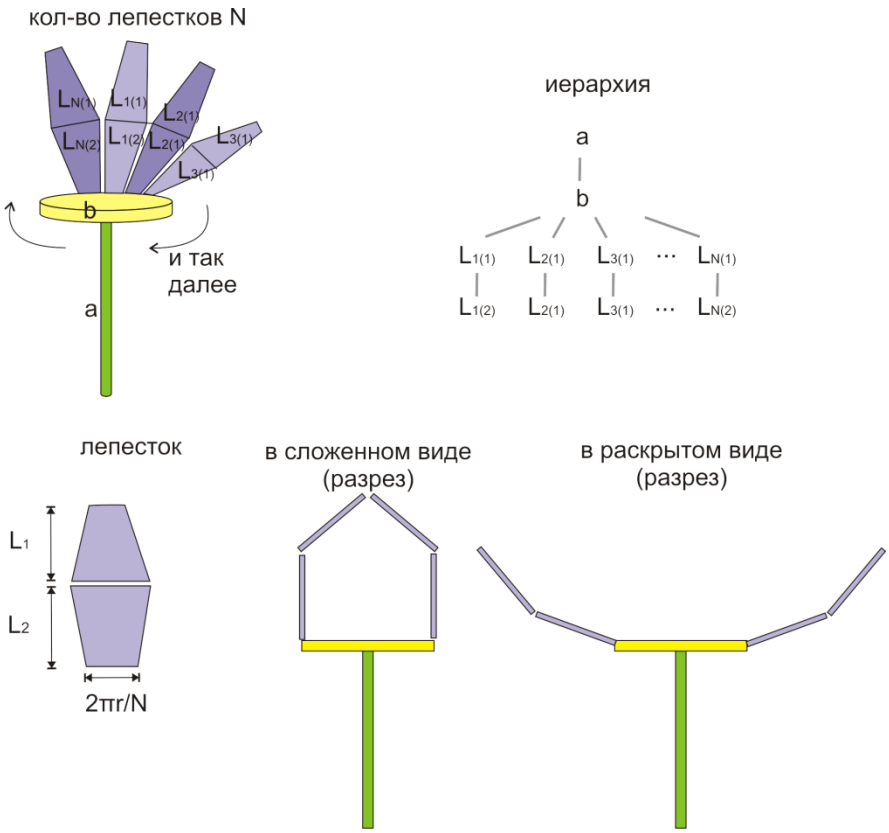

Рис. 1-4: Распускающийся цветок

1.5 Рыба

Тело рыбы состоит из нескольких (не меньше 5-ти) частей, которые представляют собой усеченные приплюснутые конусы. Также имеется хвост и два плавника, котрые

изображаются плоской геометрией (трапеция и 2 треугольника). Рыба делает «плавательные» движения телом (и хвостом), при это подгребая плавниками.

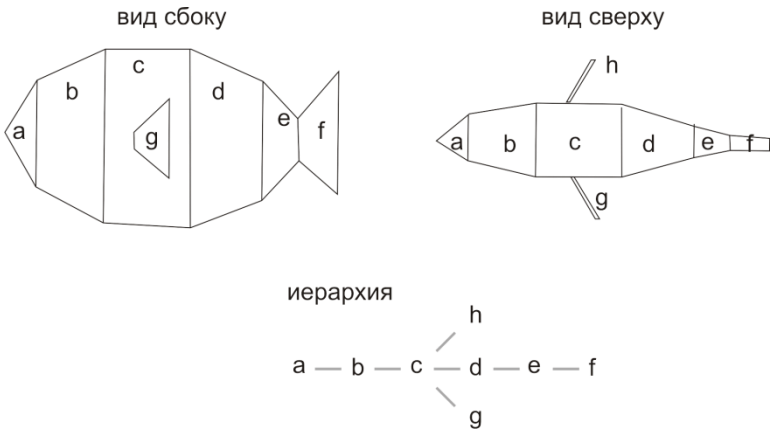

Рис. 1-5: Рыба

1.6 Музыка ветра

Весь объект состоит из простейших фигур: цилиндры, сферы. Веревочки это тоже цилиндры только тонкие. Количество висящих цилиндров можно увеличивать по желанию студента. Веревочки могут совершать небольшие вращения/качания относительно своей точки крепления. Сжатие/растяжение веревочек отсутствует. Каждая гирька ведет себя аналогично своей веревочке.

Вариант I

Простейший вариант объекта изображен на Рис. 1-6. Крестовина (b) состоит из двух цилиндров и должна вращаться.

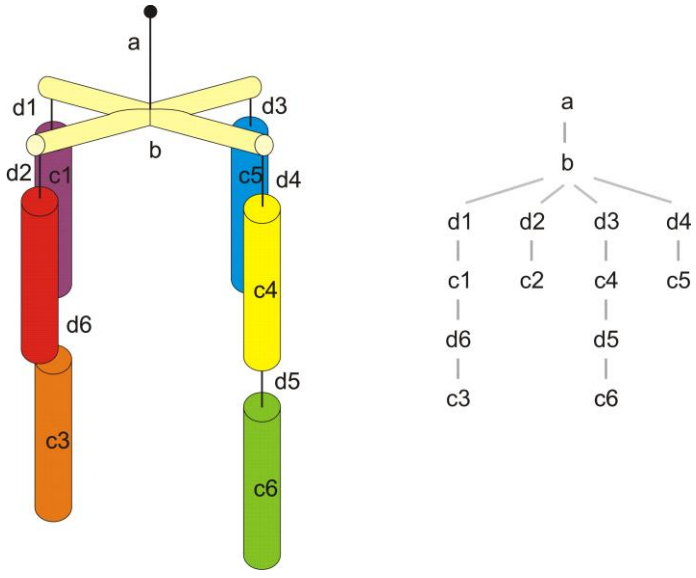

Рис. 1-6. Музыка ветра, вариант I (справа иерархия для данного объекта)

Вариант II

Простейший вариант объекта изображен на Рис. 1-7. Диск (b) и горизонтальная перекладина (c5) должны вращаться.

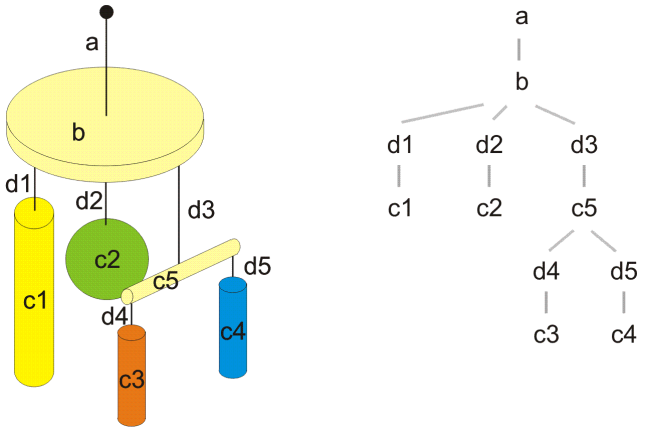

Рис. 1-7. Музыка ветра, вариант II (справа иерархия для данного объекта)

1.7 Космический корабль

Сцена: Космический корабль - коробка с турелями, которые вращаются по азимуту, на них коробки, вращающиеся по склонению, на них пушки, ходящие туда-сюда (см Рис. 1-8). При этом сам корабль летает в пространстве или вращается вокруг продольной оси. Или можно просто добавить управление кораблем по кнопкам.

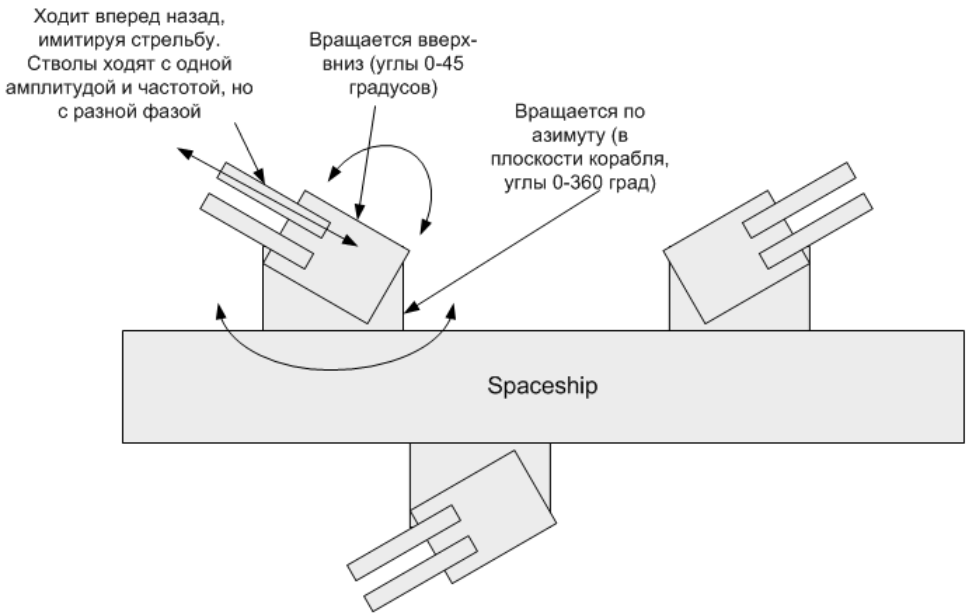

Рис. 1-8: Космический корабль

1.8 Тараканы

Таракан состоит из коробки-тела (голову уже открутили) и 6 суставчатых ног. Каждая нога состоит из двух цилиндров. Вместе ноги двигаются так, чтобы создать видимость бега по ровной поверхности (см Рис. 1-9).

Сцена: плоскость, по которой по разным траекториям бегает несколько тараканов. Можно не заморачиваться на то, что они проходят друг через друга.

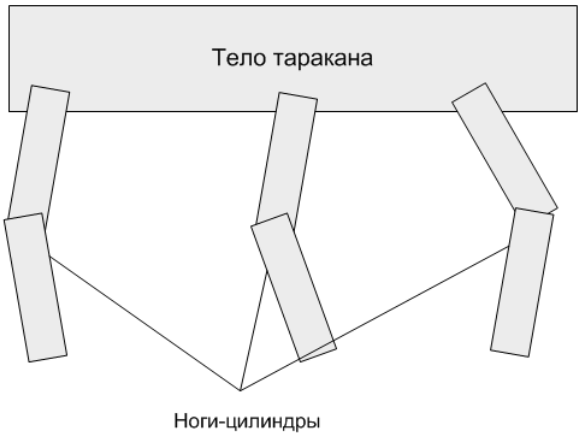

Рис. 1-9: Тараканище

Задание № 2. **Шейдеры**

Все задачи (если не сказано иного) должны быть написаны с использованием HLSL шейдеров. В каждой задаче рекомендуется использовать как вершинные, так и пиксельные (фрагментные) шейдеры, даже если последние окажутся тривиальными.

2.1 Анизотропное освещение (anisotropic shading)

Созданные руками объекты (тор, диск, шар) освещаются направленным источником света (объекты создаются руками, так как в их вершинах есть дополнительная информация – тангенциальные вектора). Альтернатива «ручному» созданию объектов: клонировать мешь, а затем дописать туда информацию о тангенциальных векторах. Можно вращаться относительно объектов, вращать сами объекты (см. первое задание). Объекты не имеют базовой текстуры.

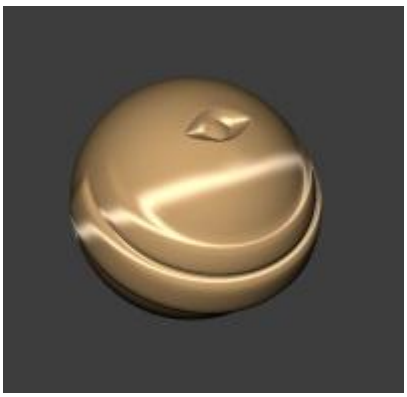

Рис. 2-1 Анизотропное освещение

Математика и прочая информация есть в презентации HWAnisotropicLight.ppt, которую можно найти на сайте.

2.2 Мультяшное освещение (cartoon shading)

Объект, прочитанный из x-mesh, освещается направленным источником света. Можно вращать объект, можно вращаться вокруг объекта.

Объект имеет базовую текстуру (желательно простенькую) или один цвет, взятый из материала (на выбор студента).

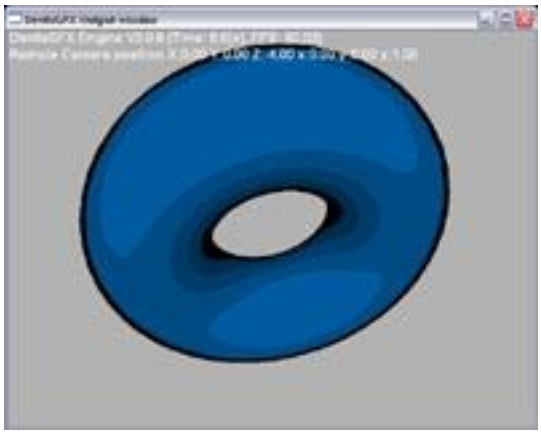

Рис. 2-2 Мультяшное освещение

Объект состоит из 2 частей:

- сам объект, с cartoon освещением: освещение рассчитывается в шейдере, а затем полученным значением [0,1] сэмплируют текстуру, которая задает градиент. Обычно используют градиент из 3 цветов.
- силуэт объекта (outline), который делается вторым рендером объекта с расширением оного по нормалям, при этом рендерятся только backfaces (culling CW) объекта, а цвет объекта делается однотонным

Порядок рендера: сначала силуэт, потом сам объект.

Должна быть возможность включать outline и cartoon освещение по отдельности.

Внимание! Расчет текстурных координат делается в пиксельном шейдере (это медленнее, зато краше).

2.3 Попиксельное освещение Фонга

Нужно закодировать освещение в духе DirectX (см. формулы в методичке), но только попиксельное. Т.е. в вершинном шейдере нужно просто передавать нормали в мировой системе координат в пиксельный шейдер, где и производить вычисления.

Параллельно делаем возможность порендерить все обычным, не пиксельным Фонгом. Все равно как, с использованием шейдеров или без (fixed pipeline). Должна быть возможность переключаться между этими двумя режимами освещения.

Сцена должна содержать 1 направленный и 1 точечный источник, которые должны включаться/выключаться повторным нажатием на кнопки “1” (направленный) и “2” (точечный). Объекты должны иметь базовую текстуру. В освещении должна присутствовать как diffuse, так и specular компонента. Объекты должны состоять из небольшого количества крупных полигонов. Должна быть возможность включить режим wireframe, чтобы посмотреть эти полигоны.

2.4 Отражения с использованием cube-map

Самостоятельно изучить, что такое cube map текстуры.

Реализовать следующую сцену: объект покрыт отражающей оболочкой. Если оболочка просвечивает, то видно базовую текстуру, если нет, то отражение (cube map текстуру). Коэффициент смешивания базовой текстуры с текстурой отражения определяется коэффициентом Френеля (вспомнить курс физики). Можно использовать аппроксимацию для коэффициента Френеля для воды:

$$F(x) = \frac{1}{(1+x)^8},$$

где $x = \cos i$, где i – угол между падающим лучом и нормалью.

Итого:

- вершинный шейдер – расчет вектора отражения, расчет коэффициента Френеля
- пиксельный шейдер – выборка из базовой текстуры, из cube map текстуры, и смешивание их (линейная интерполяция) через коэффициент Френеля

2.5 Detailed mapping

Подробности в статье, которая находится в архиве detail_map.zip на сайте (особенно интересно в конце статьи).

Вкратце идея: на объект кладется 2 текстуры: базовая (с оригинальным маппингом) и детальная (с маппингом умноженным на какое-то число 4 там, или 8). Эти текстуры блендятся друг с другом, причем коэффициент блендинга зависит от расстояния до вершины объекта. Чем ближе мы к объекту, тем более явно проявляется детальная текстура, забивая/заменяя расползающиеся пиксели основной текстуры.

Задача: реализовать это в вершинном/пиксельном шейдере для какого-нибудь объекта. Чтобы это все хорошо смотрелось разрешение базовой текстуры надо взять 128x128 или 256x256. Иначе текстура сама будет слишком детальной.

2.6 Попиксельное освещение Миннаерта

Попиксельное освещение Миннаерта + карта нормалей. Необходимо реализовать освещение объекта по модели Миннаерта, при этом при освещении следует использовать объект, на котором лежит карта нормалей. Освещение кодировать в пиксельном шейдере.

Модель брать отсюда http://www.realistic3d.com (можно даже исходники скачать) и из презентации gdc2002_textureuses.pdf, которая размещена на сайте. Рекомендую использовать эффекты и x-mesh. Если использовать модель dwarf.x из xdk, то там есть текстура с картой нормалей.

У примера должно быть 2 режима: обычный diffuse lighting и Minnaert.

2.7 Цепочка со скиннингом

Надеюсь, вам объясняли на лекциях, что такое скиннинг.

Итак, вам нужно сделать цепь звеньев-костей (5-ти будет достаточно) и натянуть на них кожу-цилиндр.

Реализовывать следует на шейдерах. Сам скиннинг: интерполяция матриц (или результатов умножения матриц на вектор) делается в вершинном шейдере. Пиксельный шейдер простой - просто вывод текстуры (освещение можно не делать).

В вершине следует хранить 2 весовых коэффициента и 2 индекса костей. Задавать их следует на создании вершинного буфера цилиндра (см. Рис. 2-3).

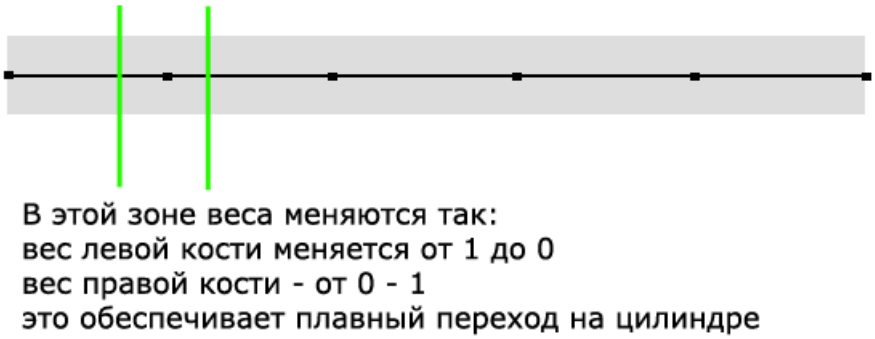

Рис. 2-3: Весовые коэффициенты для скиннинга

Вариант А: делаем змею

Вариант В: подвешиваем веревочку

2.8 Система частиц

Все частицы это один объект, который создается следующим образом:

1. вершинный буфер, в котором лежит набор квадов (4 вершины). Для всех 4х вершин скорость и время создания одинаковые. Смещения: (-1,-1), (+1,-1), (+1,+1), (-1,+1). Формат вершины:
   - смещение - float2. (offset)
   - скорость - float3. (speed)
   - время создания частицы - float. (time_created)
2. правильный индексный буфер
3. шейдер, в котором вычисляем позицию вершины, так чтобы это был спрайт, повернутый на наблюдателя. Для этого в шейдер передаем камерные вектора "вправо" (CamRight) и "вверх" (CamUp). Так же в шейдер передаем текущее время и вектор гравитации. Позицию частицы вычисляем так: $t = cur\_time - time\_created$, $pos = speed \cdot t - gravity \cdot t \cdot t \cdot 0.5f$. Из позиции частицы легко вычислить позицию вершины $vtx\_pos = pos + scale \cdot CamRight \cdot offset.x + scale \cdot CamUp \cdot offset.y$, где $scale$ тоже задается извне.
4. раз в сколько-то секунд вы добавляете в вершинный буфер еще один квад и прописываете время создания. Когда буфер заполнится переинициализируете первую частицу и так далее. Скорости задаете случайные.

5. На частицы кладете текстуру. Вычисляете текстурные координаты из offset. Блендинг делаете аддитивным. Текстура - световое пятно - тогда частицы будут казаться горящими точками.

Результат - фонтан светящихся частиц. Фон - плоскость с текстурой.

2.9 Формы

Вы должны сделать вершинный и индексный буфер, содержащий плотно тесселированную плоскость 100x100 где-то. Формат вершины float2. Больше не надо. Координаты в интервале [0,1]. В вершинном шейдере вы вычисляете некую функцию float3=f(float2 uv). Результат этой функции - координаты позиции в мировой системе координат. Далее, умножение на матрицу - как обычно. Таким образом получается, что вы можете рендерить параметрические поверхности.

Вы должны уметь рендерить различные поверхности (с переключением по кнопке).

$$x = \sin(2\pi \cdot u)\cos(\pi \cdot v - \pi \cdot 0.5)$$

Первая поверхность - сфера: $y = \cos(2\pi \cdot u)\cos(\pi \cdot v - \pi \cdot 0.5)$

$$z = \sin(\pi \cdot v - \pi \cdot 0.5)$$

Далее, очевидно, можно сделать и цилиндр. Варианты выдаваемы преподавателем:

- тор
- спираль
- http://en.wikipedia.org/wiki/M%C3%B6bius_Strip
- http://en.wikipedia.org/wiki/Klein_bottle (первая и вторая "immersion").

Цвет вершины, как и на картинках в Википедии - просто float3(u, v, 0) - или что-то в таком духе. При этом хочется видеть черный wireframe.

Режимы визуализации:

- Объект с цветом
- Объект с цветом + черный wireframe
- Белый wireframe

Есть возможность по кнопке включать, выключать culling. Есть возможность по кнопке включать, выключать z-buffer.

Есть возможность по кнопке включать, выключать блендинг для объекта (аддитивный: one * srccolor + one * dstcolor).

2.10 Area light

Нужно реализовать подобие area light, который действует также как и обычный точечный источник, но расстояние (и направление) считается не до точки, а до ближайшей точки некоторого объекта (допустим треугольника или четырехугольника). Код по расчету ближайшей точки на треугольнике (прямоугольнике) можно нарыть в интернете. Далее достаточно закодировать это в вершинном шейдере. Для того, чтобы показать, что все работает, освещаемая сцена должна содержать плоскость и несколько сфер. Сам источник (геометрию) тоже надо порендерить.