**4** РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

**4.1** Общие положения

Представляемый программный продукт имеет достаточно занчительный объем, поэтому целесообразно описывать только ключевые алгоритмы использованые при разработке.

В данном разделе пояснительной записке будут отражены алгоритмы, которые непосредственно формируют результирующее изображение. Эти алгоритмы реализованы в шейдерных программах написанных на языке программирования glsl (OpenGL shader language).

Рендеринг изображения в демонстрации работы программного продукта происходит в семь этапов. Каждый этап описывается одним или несколькими шагами (RenderStep). Из этих этапов четыре – предварительные, то есть производятся один раз перед началом основного цикла, остальные три этапа повторяются для каждого кадра. Перичислим этапы рендеринга:

* преобразование плоской текстуры окружения в кубическую текстуру;
* подготовка карты диффузного освещения на базе изображения окружения;
* подготовка нескольких карт для рендерига отражений от окружения;
* подготовка карты значений двулучевой функции отражательной способности;
* подготовка кубических карт теней для каждого источника освещения, использованного в сцене;
* рендеринг объектов сцены в hdr (high dynamic range – высокий динамический диапазон) текстуру;
* рендеринг изображения окружения в hdr текстуру.
* преобразование hdr текстуры в 24-битное цветное изображение и вывод его на экран.

**4.2** glsl

Для упрощения понимания фрагментов кода, которые будут представлены далее следует пояснить некоторые особенности языка glsl.

Язык glsl имеет синтаксис подобный языку С. Программа на языке glsl состоит из следующих основных частей:

* объявления входных данных;
* объявления выходных данных;
* функции main().

При необходимости программист может объявлять свои функции и константы. Синтаксих объявлений и принцип их обработки соответсвует языку С.

Отличительной особенностью являются глобальные константы с модификатором uniform.

uniform float roughness;

Значения этих констант могут быть изменены со стороны приложения, но не могут быть изменены в рамках шейдерной программы.

Перечислим ряд встроенных типов объявленных в glsl:

1. vec2, vec3, vec4. Вектор из 2, 3 или 4 значений с плавающей запятой. Отдельные значения могут адресоваться как элемент массива vec3[1] или с помощью именованных перечислений vec3.xyz, vec3.rgb, vec3.xxx.
2. mat{i x j}. Матрица размерностью i \* j, минимальная размерность 2\*2, максимальная - 4\*4, для квадратных матриц второе число опускается (mat4 вместо mat4x4). Важно отметить, что в glsl сначала идет индекс колонки, а потом индекс строки.
3. sampler\*. Переменные представляющие доступ к связанной текстуре в шейдерной программе, для каждого типа текстуры в OpenGL имеется соответствующий ей тип sampler\* (sampler2D, samplerCube, samplerCubeArray).
4. Базовые встроенные типы C++: булевые переменные, знаковые и беззнаковые целые числа, числа с плавающей запятой одинарной и двойной точности.

Входные данные шейдерной программы пмечаются модификатором in.

in vec2 uvCoords;

Выходные данные помечаются модификатором out.

out vec4 outColor;

Для каждого типа шейдеров в glsl имеется ряд встроенных входных и выходных переменных, эти переменные начинаются с gl\_.

Для векторных и матричных типов в glsl определены различные математические операция такие как: векторное умножение (cross()), скалярное умножение (dot()), матричное умножение, транспонирование, нахождение определителя и другие.