**Задача 1**

**Основные шаги графического пайплайна на примере рендеринга куба**

**Цель задания**

● **Познакомиться с основными шагами графического пайплайна** на примере графического пайплайна **OpenGL** с использованием библиотеки **Qt**; ● Ознакомиться со способами задания геометрии: освоить использование **вершинного буфера** (**VBO**) и **индексного буфера** (**IBO**), а также ознакомиться с базовыми графическими примитивами **TRIANGLE** и **TRIANGLE\_STRIP**; ● Научиться писать и компилировать простые **шейдерные программы**, с использованием **атрибутов** (**attributes**) и **юниформ** (**uniforms**);

● Получить базовое представление о **World-View-Projection** преобразованиях.

**Задача в общих словах**

На основе приложения по рендерингу треугольника нарисовать куб, центр масс которого расположен в начале системы координат *(0, 0, 0)*. Поддержать возможность смены цвета этого куба, а также его вращения его вокруг произвольного единичного вектора, выходящего из начала системы координат.

**Приблизительный алгоритм выполнения задания**

● Разобраться с демо приложением по рендерингу треугольника на **OpenGL 2.1**; ● Задать геометрию куба, используя один из графических примитивов **TRIANGLE** или **TRIANGLE\_STRIP**, правильно заполнить **вершинный** и **индексный** буферы (необходимо учесть используемый графический примитив); ● Задать необходимые **атрибуты** и **юниформы** (минимальный набор: координаты вершин, цвет и MVP матрица);

● Написать **вершинный** (**vertex shader**) и **фрагментный**(**пиксельный**) (**fragment shader**) шейдеры, использующие заданные атрибуты и юниформы; ● **Скомпилировать** и **слинковать** написанную шейдерную программу; ● Добавить возможность **динамически** (**runtime**) задать цвет куба, желательно через простой и понятный **UI** (можно воспользоваться **QColorDialog**); ● Добавить возможность динамически задать **ось вращения**, а затем поддержать и само вращение куба;

● Попробовать включить/выключить **отсечение** (**clipping**, **culling**), **тест глубины** и **трафарета** (**depth test**, **stencil**), **сглаживание множественной выборкой** (**multisampling**) и посмотреть на результаты.

**Дополнительно можно**

● **При наличии “современной” графической карты, поддерживающей OpenGL API > 2.1**. Попробовать запустить пример, используя более новую

версию API, необходимо будет переписать шейдерные программы, используя новый стандарт языка **GLSL**, соответствующий вашей версии графического API.

● Попробовать рисовать только **FRONT\_FACE** поверхности, наглядно убедится, как работает **отсечение** (**clipping** и **face culling**).

● Попробовать нарисовать “много” кубиков, затем “включать и выключать” отдельные шаги пайплайна и попробовать оценить, как каждый из них влияет на производительность.

**Полезные ссылки**

● https://www.qt.io - Qt Download page;

● https://doc.qt.io/qt-5/ - Qt 5.15 docs;

● https://www.opengl.org - OpenGL references;

● https://www.khronos.org/opengl/wiki/Getting\_Started - Khronos getting started with OpenGL guidelines;

● https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.3.30.pdf - GLSL 3.30 specification;

● https://www.khronos.org/opengl/wiki/Vertex\_Specification - VBO/IBO and vertex/index buffers, attributes specifications;

● https://www.khronos.org/opengl/wiki/GLSL\_Object - Program objects (shaders objects);

● https://www.khronos.org/opengl/wiki/Uniform\_(GLSL) - GLSL uniforms; ● http://www.opengl-tutorial.org - OpenGL tutorials.