Техническое задание
на разработку программного обеспечения

1. Название ПО: vtxgen

2. Описание ПО.

ПО является одним из компонентов более крупной программы, служащей для управления голографической камерой. В программе присутствует модуль трехмерной визуализации голограммы и алгоритмически найденных дефектов в ней, использующий OpenGL для визуализации. Визуализация строится на основе массива бинарных изображений. В OpenGL построение трехмерных моделей осуществляется из треугольных полигонов, имеющих по три вершины, расположенных в трех измерениях пространства. Для отрисовки каждого треугольного полигона используется информация о расположении каждой из его вершин в трех измерениях относительно центра оси. Задача ПО – расчитать координаты каждой вершины таким образом, чтобы можно было построить трехмерную модель, используя минимальное возможное количество полигонов. Также, ПО должно принимать три коэффицента масштабирования (по трем осям) модели, передаваемые через API.

Исполнителю также предоставляется исходный код программы visualise.exe в приложении к ТЗ. Эта программа является простым визуализатором результата работы vtxgen. К исходному коду также прилагается Readme с руководством пользователя и несколько папок с массивами изображений.

3. Задачи, выполняемые ПО.

3.1. API ПО должно иметь одну точку входа — функцию `CalculateGLVtx()`.

Прототип функции `CalculateGLVtx()`:

```
// Calculates vertex data for OpenGL rendering from the vector
// of 1-bit 2D images.
//
// Note that std::vector<bool> is a special standard container where
// each bool uses only one bit of space rather than one byte as
// a normal bool would. Since you can't take the address of a bit
// within a byte, things such as operator[] can't return a bool&
// but instead return a proxy object that allows to manipulate
// the particular bit in question. Since this proxy object is not
// a bool&, you can't assign its address to a bool* like you could
// with the result of such an operator call on a "normal" container.
//
// @param images Vector of 1-bit 2D images with [z][x][y] orientation.
// @param x_scale Scale factor in x axis.
// @param y_scale Scale factor in y axis.
// @param z_scale Scale factor in z axis.
// @return Vertex data for OpenGL. The first vector holds the vertex data
// for the "inside" objects found by the NN, the seconds vector holds the
// vertex data for the "outside" space in between those objects.
VTXGEN_API std::pair<std::vector<GLfloat>, std::vector<GLfloat>> CalculateGLVtx(
              const std::vector<std::vector<std::vector<bool>>>& images,
              float x_scale,
              float y_scale,
              float z_scale);
```

Тип GLfloat содержится в библиотеке glfw3.

3.2. ПО должно рассчитывать массив вершин треугольных полигонов, необходимых для построения 3D-модели, используя следующий алгоритм:

3.2.1. Белые пиксели входных изображений считаются вертикальными гранями внутренних объектов.
3.2.2. Черные пиксели входных изображений считаются вертикальными гранями внешних объектов.

3.2.3. Горизонтальные грани должны быть рассчитаны программой.

3.2.4. Все грани должны располагаться под прямым углом, наклон не допускается.

3.2.5. Если массив изображений содержит только одно изображение — передняя сторона 3D-модели должна быть равна задней.

3.2.6. В местах пересечения граней внешних и внутренних объектов должны присутствовать обе грани.

3.2.7. Внутри внешних и внутренних объектов граней быть не должно.

3.2.8. Все объекты должны быть «закрыты» со всех сторон и полые внутри, включая объекты расположенные с краю модели.

3.2.9. Модель должна быть «отзеркалена». То есть, первые изображения из массива должны учитываться как грани с большим значением по оси Z, чем последние.

3.2.10. Каждая грань должна состоять ровно из двух треугольников.

3.3. Возврат функции `CalculateGLVtx()` должен содержать пару векторов значений GLfloat для внутренних и внешних объектов. Они должны выглядеть следующим образов:
{
1 вершина 1 треугольника X, 1 вершина 1 треугольника Y, 1 вершина 1 треугольника Z,
2 вершина 1 треугольника X, 3 вершина 1 треугольника Y, 3 вершина 1 треугольника Z,
3 вершина 1 треугольника X, 2 вершина 1 треугольника Y, 3 вершина 1 треугольника Z,
1 вершина 2 треугольника X, 1 вершина 2 треугольника Y, 1 вершина 2 треугольника Z,
2 вершина 2 треугольника X, 3 вершина 2 треугольника Y, 3 вершина 2 треугольника Z,
3 вершина 2 треугольника X, 2 вершина 2 треугольника Y, 3 вершина 2 треугольника Z,
и.т.д}

Порядок треугольников не имеет значения, однако порядок вершин имеет. Треугольники не обязательно должны иметь общие грани или быть расположены рядом.

4. Компоненты ПО.

4.1. DLL vtxgen.

5. Требования к написанию и оформлению программного кода.

5.1. Выбор языка программирования — Visual C++ 17 стандарта ISO C++ 17.

5.2. Стиль написания и форматирования кода должен стремиться к стандарту Google (https://google.github.io/styleguide/cppguide.html).

5.3. Все функции, переменные и другие элементы кода должны иметь английские названия, использование транслитерации с русского языка не допускается.

5.4. Код должен быть снабжён понятными комментариями на английском языке.

5.5 Функции должны быть снабжены комментариями со следующим оформлением:

5.5.1. В файле заголовка:

```
// Краткое описание задачи, выполняемой функцией.
//
// Пояснение, если функция содержит неочевидное поведение, которое требуется принимать во
// внимание.
//
// @param аргумент_1 Описание аргумента.
// @param аргумент_2 Описание аргумента.
// @param аргумент_3 Описание аргумента.
// @return Описание возврата функции.
```

5.5.2. В файле исходного кода:

```
// Подробное описание
// работы функции.
//
// Пояснение, если функция содержит неочевидное поведение, которое требуется принимать во
// внимание.
```

5.6. Максимальная длина одной строки кода — 80 знаков.

5.7. Прототипы функций должны быть вынесены в отдельные файлы заголовка.

5.8. Платформа — Windows 10 64 бит.

6. Дополнительные пожелания заказчика.

6.1. ПО должно использовать параллельные вычисления там, где это возможно.

6.2. По возможности, минимальное использование сторонних библиотек.

7. Требования к приемке-сдаче проекта.

7.1. Комплект сдачи проекта.

7.1.1 Исходный код программы.

7.1.2. Инструкции по сбору и компиляции проекта с использованием Microsoft Visual Studio 2017.

7.1.3. Документация к API.

7.2. Работа всех компонентов ПО должна быть протестирована исполнителем на системах, удовлетворяющих требованиям Windows 10 64 бит.

7.3. Заказчику предоставляется 4 календарные недели на тестирование программы. В случае

некорректного функционирования – программа должна быть доработана заказчиком. После доработок заказчику предоставляется дополнительные две недели на тестирование.