CSD2151/CSD2150/CS250 Introduction to Real-Time Rendering

Assignment 5.1 (Environment Mapping VPL)

Topics and references

- Multi-pass rendering
- Skybox
- Cube mapping
- Gamma correction
- Environment mapping

Task

In this first part of submission for assignment 5 you have to implement a few functions that are used later without any change in shaders for the environment mapping. All functions are specified in this document below and are the subject of discussion in the second class this week.

To start, download and unzip all provided files into a new folder. Inside the folder you find source files with data structures and unit tests, including:

- Makefile to build the program by using GCC (g++17) compiler with the command line interface. Successful build with GCC on your computer is required in order to submit your code without compilation, execution, and evaluation problems in the VLP.
- Solution files (.sln) to open the project, build, and test the program in Visual Studio IDE. It is optional. Use it to develop and debug your code in a comfortable environment.

Important notes:

- Your implementation must pass all given tests in order to get the full mark for this assignment.
- You have to submit two required files and nothing else. It means, that any changes in the other files can cause your submission failed in the Moodle.
- Every edited by you source and header file must begin with a file-level documentation block. Read more below.
- The compiler, tools and library for this assignment are exactly the same as for the first VPL-based assignment. All information regarding the installation you can find there.

Specifications

1. Function **vec4 checkerboardTexture(vec2 uv, float size)**

This function implements a procedural texture mapping to be used to apply a checkerboard-like texture to a surface. The first square, started from the corner (0, 0), must be colored black.

Parameters:

- uv - normalized texture coordinates (0 <= uv.x < 1, and 0 <= uv.y < 1),

- size - defines the number of black and white squares along the texture's width or height.

Return:

- color either black or white, where black is vec4(0.0f, 0.0f, 0.0f, 1.0f) and white is vec4(1.0f, 1.0f, 1.0f, 1.0f).

2. Function **vec2 vec2uv(vec3 v)**

You have to implement a function that can solve a simplified version of the cube mapping problem, when only back faces of the skybox (which is a cube) are mapped, the other two (top and bottom) are not used. Each internal face of the cube has the same texture coordinates from (0, 0) to (1, 1) at the bottom-left and top-right corners of the face.

Given:

- a camera located at the center of a reference frame,
- normalized view vector (v) of the camera,
- skybox - an axis aligned cube with extreme corners (-1,-1, -1) and (1, 1, 1).

Return:

- The function must calculate and return the texture coordinate where the given view ray (defined by vector v) intersects the side of the skybox.
- If the view ray intersects the top or bottom sides of the cube, then the function must return vec2(0.0f, 0.0f).

Solution for this problem involves a simple math for 4 different orientations of the vector and is subject to discuss in the class.

Compiling, executing, and testing

- Run **make** to bring program executable main.out up to date.
- Run **make test** to test the last knows successfully compiled executable.
- Run **make leak** to test your implementation for memory leakage. Make sure that the output does not show any memory related issue.

File-level documentation

Every **edited by student** source file *must* begin with a *file-level* documentation block. This documentation serves the purpose of providing a reader the purpose of this source file at some later point of time. It has simple format. This module will not use any documentation generator like Doxygen.

```
/!*****************************************************************************
\file unittests_functions.cpp
\author Vadim Surov (vsurov\@digipen.edu)
\co-author YOUR NAME (DIGIPEN EMAIL)
\par Course: CSD2151/CSD2150/CS250
\par Assignment: 5.1 (Environment Mapping)
\date 02/05/2022 (MM/DD/YYYY)
\brief This file has definitions of functions for unit tests.
```

```
This code is intended to be completed and submitted by a student,
so any changes in this file will be used in the evaluation on the VPL server.
You should not change functions' name and parameter types in provided code.
*****************************************************************************/
```

Submission and automatic evaluation

Your submission will receive:

- **F** grade if your submitted code doesn't compile with the full suite of g++ options.
- **F** grade if your submitted code doesn't link to create an executable.
- **A+** grade if output of function matches correct output of auto grader.
- **Not A+** grade if your implementation's output doesn't match correct output of the grader. The auto grader will provide a proportional grade based on how many incorrect results were generated by your submission.
- **Not A+** grade if your submission files do not have or have incorrect information in the file-level documentation block. Each missing or incomplete or irrelevant block will result in a deduction of a letter grade.

Honor code: By submitting your solutions and/or source code for this assignment, you acknowledge that you will not give or take answers to solutions and/or source code from your peers or from online sources; in addition, you also acknowledge that you will take an active part in ensuring that others will uphold the spirit and letter of this honor code.