GD2P04 – Advanced Graphics for Games
Render Passes and Terrain Generation (30%)

Component code and name	GD2P04
Assignment name	Multiple Render Passes and Terrain Generation
Weighting	30%
Submission deadline	Week 8
Week issued	Week 4

Brief

Instructions/Requirements

Create a technical demo with multiple scenes (input to switch) to showcase the following:

- **Scene 1: Stencil Test (5%)**
  - Scene is loaded by pressing the key '1'.
  - Create a solid outline effect around a 3D asymmetrical model using a stencil test.
    - Apply the outline effect to multiple objects using the same stencil buffer to that shine through non-stenciled objects and have only one merged outline when the objects overlap.
    - Rotate at least one stenciled object around its local center with the outline correctly updating each frame.

- **Scene 2: Terrain Rendering (30%)**
  - Scene is loaded by pressing the key '2'.
  - Load heightmap data from a file then generate and render a 3D terrain object.
  - Smoothen the heightmap data before generating the 3D terrain object.
  - Generate a normal for each point on the terrain using the slope and surrounding points.
  - Apply directional lighting and appropriate texture to the 3D terrain object.
  - Based on height and/or normals load and apply at least 4 terrain textures (e.g. grass, dirt, stone, snow) to the 3D terrain object and smoothly blend together where they meet.

- **Scene 3: Perlin Noise Generation (30%)**
  - Scene is loaded by pressing the key '3'.
  - Create an algorithm for Perlin noise generation that generates unique noise data on each run. Seed to time.
  - Save the generated noise data as a JPG image and render in the scene on a large quad.
  - Apply gradient coloring with at least 4 colors (e.g., Fire effect: White, Yellow, Red, Black) to the quad using the noise data.
  - Save the generated noise data as a RAW file, load the data as a heightmap and apply to a 3D terrain object.
  - Create a second quad to display an animation of real time transitions using Perlin noise.

- **Scene 4: Framebuffers + Post Processing (25%)**
  - Scene is loaded by pressing the key '4'.
  - Render an entire scene (at least 3 objects) to a Framebuffer Object (FBO).
  - Render a full window quad using the texture from the FBO.
  - Apply the following post-processing effects (one at a time) to the full window quad:
    - Color inversion.
    - Greyscale using the luminosity method.
    - ‘Raining on your screen’ – ShaderToy.
    - Any additional post-processing effect from ShaderToy.
  - Cycle through the post-processing effects using the ‘Tab’ key, showing one effect at a time (including no effect).

- **Programming Practices (10%)**

- A ReadMe.txt file is included stating the functionality, controls, and any additional triggers or needed information for the project to showcase all included features.
- Function headers are consistent and present across all files and functions.
- Comments are used to clarify the purpose and use of data and functions demonstrating an understanding of the key areas of related code.
- Classes and functions are appropriately used to create systems and demonstrate a higher understanding of modular code and proper C++ OOP concepts.
- consistency of naming conventions, code formatting, and accessors to increase readability across all files.
- No warnings are generated during the build that originate from student project files.
- No Intermediate files are included.

Guidelines

Follow the guidelines, standards, and specifications regarding the tasks outlined in the instructions section.

The source code is required to display the following features:

- Compiling code:
  - Code must build as submitted in both debug and release.
  - No errors or warnings (originating from student files) at warning level 3 for all build targets.

Submission Guidelines

Place the work in a .zip file and submit it to Blackboard by the time and date specified (see Blackboard).

Naming conventions

The file structure and file names of the submission must follow the file hierarchy listed below.

🗍 YYYY-MM-DD – GD2P04 – Assignment1– *Student Name*.zip
- 🗁 Source – *Student Name*
  - 🗋 Assignment 1.sln
  - … Project and source code, etc.
- 🗍 Release Build – *Student Name*.zip.
  - … Include additional resources and DLLs

Submission structure

Source code folder (Source runs in Visual Studio 2019/2022):

- Solution file (.sln).
- Project file (.vcproj).
- Source files (.cpp, .h).

Release build zip:

- Standalone executable (.exe).
- Any additional files required to run the executable.
- Readme file (.txt).

Intermediate and repository files are removed to reduce file size.

Submission Policy

See the component overview for details.