Criteria	Weight	A+ (90 - 100)	A (80 - 90)	B (65 - 79)	C (50 - 64)	D (0 - 49)
Programming Practices	10	As A grade, plus the following additions: > Function headers are consistent and present across all files and functions. > Classes and functions are appropriately used to create systems and demonstrate a higher understanding of modular code and proper C++ OOP concepts.	As B grade, plus the following additions: > No intermediate and/or unnecessary files have been included. > No warnings are generated during building that originates from student project files. > Very good use and consistency of naming conventions, code formatting, and accessors to increase readability across all files. > Function headers are consistent across most files and functions. > Classes and functions are appropriately used to create systems and demonstrate an understanding of modular code.	As C grade, plus the following additions: > Good use of naming conventions, code formatting, and accessors to increase readability. > Comments are used to clarify the purpose and use of data and functions demonstrating an understanding of the key areas of related code. > Some function headers are present. > Some additional classes and functions have been used where appropriate to refine the codebase.	> The source builds (debug and release) without errors as submitted. > Submission has the correct file/folder structure although some extra intermediate files may have been included. > Some use of naming conventions and code formatting. > Some commenting is present to clarify the purpose of key areas of code. > 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.	> Insufficient or no attempt at meaningful and consistent programming practices such as commenting, readability, naming conventions, code structure, etc. OR > As submitted, the release executable does not run or the source fails to build in either debug or release.
Shadows **(Scene 1)**	25	As A grade, plus the following additions: > A second directional light with a noticeably different direction is added to the scene. > All objects cast shadows from both light sources and the shadows are correctly displayed.	As B grade, plus the following additions: > A 3D terrain object is placed in the scene with the other objects casting shadows onto it. > The 3D terrain also casts shadows and is capable of self-shadowing. The height values of the terrain and light direction must show this.	As C grade, plus the following additions: > An additional 3D model is loaded and placed in the scene touching the flat plane (or terrain), also casting a shadow. > The movable object can also be moved up and down on the Y-axis. > The movable object can cast a shadow on the above 3D model. > A shadow bias and Percentage Closer Filter (PCF) are added to the shadows.	> As submitted, both the release executable and source (debug and release) build and run without errors or crashing. > The scene can be correctly loaded by pressing the standard keyboard '1' key. > At least a 3D object, a flat plane (or terrain), and a directional light source are rendered in a scene with all objects having Blinn-Phong lighting applied. > A shadow map for the directional light is correctly generated with a texture (depth information) using a framebuffer object. > A shadow from the 3D object is correctly cast onto the flat plane (or terrain) using the shadow map created from the directional light. > The original 3D object can be moved around on the X and Z planes, and its shadow will update accordingly.	> Insufficient or no attempt at creating shadows using a shadow map and framebuffer object within the programmable pipeline in OpenGL. OR > As submitted, the release executable does not run or the source fails to build in either debug or release.
Deferred Rendering **(Scene 2)**	25	As A grade, plus the following additions: > The light source objects have the correct depth in the scene. > The light source objects are semi-transparent and blended correctly with the screen-space quad.	As B grade, plus the following additions: > The point lights are represented as unlit and untextured (pure color only) 3D objects (light source objects) that match the color of the light showing the lights' positioning in the scene. > The light source objects are rendered separately using a forward rendering pass.	As C grade, plus the following additions: > At least 10 point lights (with attenuation) are placed in the scene, each with a different color. > At least 20 3D models and a flat plane (or terrain) are rendered in the scene with all objects having Blinn-Phong lighting applied. > The screen-space lighting calculations use all lights within the scene.	> As submitted, both the release executable and source (debug and release) build and run without errors or crashing. > The scene can be correctly loaded by pressing the standard keyboard '2' key'. > At least 5 3D models, a flat plane (or terrain), and a point light (with attenuation) are rendered in a scene with all objects having Blinn-Phong lighting applied. > A geometry pass renders all scene objects to a framebuffer with multiple attachments (position, normal, albedo, depth). > A screen-space lighting pass of the scene is rendered to a screen-space quad using the textures generated by the geometry pass.	> Insufficient or no attempt at creating a deferred rendering technique using the programmable pipeline in OpenGL. OR > As submitted, the release executable does not run or the source fails to build in either debug or release.
Compute Shader (GPU Particles) **(Scene 3)**	25	As A grade, plus the following additions: > Each firework is split into two effects: The firework launches upwards with a particle trail, and after a timer, it explodes at its current position. > Each initial firework/trail timer is set to a different time so the fireworks explode at different times.	As B grade, plus the following additions: > Blending is enabled and each particle individually fades based on its lifetime (1 - 0). > Each particle's initial direction is normalized so that the overall burst is a spherical shape.	As C grade, plus the following additions: > Pressing the 'F' key triggers multiple (at least 4) visible firework particle explosions in random origin positions, resetting after the full lifetime, and can be played again. > Each firework explosion is a different color.	> As submitted, both the release executable and source (debug and release) build and run without errors or crashing. > The scene can be correctly loaded by pressing the standard keyboard '3' key. > A firework particle system (burst of particles outwards in all directions) is created using the compute shader and associated buffers to control each particle's movement. > The particle system is either continuous, with each particle resetting at the end of its individual lifetime, or it plays out only once, with each particle not being rendered after the end of its lifetime, with the whole system able to be played again by pressing the 'F' key. >Each firework particle system has at least 100,000 particles overall. > The draw call of all particle systems uses GL_POINTS.	> Insufficient or no attempt at rendering a particle system using the Compute Shader and programmable pipeline in OpenGL. OR > As submitted, the release executable does not run or the source fails to build in either debug or release.
Tessellation Shaders + Level of Detail (LOD) **(Scene 4)**	15	As A grade, plus the following additions: > Tessellation is used to render a terrain in the scene with at least 32x32 sections. > The height of the tessellated coordinates is sampled from a height map texture in the TES. > An appropriate texture is applied to the terrain.	As B grade, plus the following additions: > A new mesh with 4 points creates two triangle patches to form a quad rendered separately in the scene. > The tessellated quad (made from 2 connected triangle patches) has LOD applied, with the tessellation level increasing as the camera gets closer and decreasing as the camera moves further away. > The camera has movement controls to move towards the tessellated quad and backward.	As C grade, plus the following additions: > Texture coordinate information is also calculated using barycentric coordinates for the triangle. > An appropriate texture is applied to the tessellated triangle.	> As submitted, both the release executable and source (debug and release) build and run without errors or crashing. > The scene can be correctly loaded by pressing the key '4'. > Wireframe mode can be toggled on/off with a key press. > Tessellation shaders (TCS and TES) create a triangle patch from a mesh of 3 points. Render the triangle patch to the scene. > The triangle patch is subdivided, with the outer levels, each having 5 sections and the inner levels having 7 sections. > Barycentric coordinates are used to calculate the tessellated coordinates of the triangle.	> Insufficient or no attempt at implementing tessellation shaders using barycentric coordinates to subdivide triangle patches within the programmable pipeline in OpenGL. OR > As submitted, the release executable does not run, or the source fails to build in either debug or release.