CS562 Assignment 4: Horizon-Based Ambient Occlusion

OBJECTIVE

In this assignment, you are going to implement Image-Space Horizon-Based Ambient Occlusion (Bavoil, Sainz, & Dimitrov, 2008) that tries to approximate the effect of indirect lighting on shading an object.

1. Scene setup

Load the provided scene file that will import *sponza* model and *suzanne*.

2. Algorithm Overview

Implement the technique explained in the paper Image-Space Horizon-Based Ambient Occlusion.

1. **Application:** The HBAO technique is a screen space technique, which means that the amount of C++ code needed is very small. This code will add an extra couple of screen space passes between the geometry and lighting pass, so that the generated texture (ambient occlusion data) can be passed to the lighting stage and properly used. Note that if you keep the decal pass from assignment 3 ambient occlusion pass should happen after this one.

   Once the ambient occlusion texture is generated, the result will be noisy due to the sampling process. In order to solve that problem the ambient occlusion texture will be blurred before using it for the lighting pass. A regular two pass Gaussian blur might usually do the job, but it may produce haloing artifacts because edges information is not taken into account. For that reason a Bilateral Filter (Paris, 2007) is a better solution to solve the noise problem. The latter will be implemented for this assignment.

2. **Vertex Shader:** Usual matrix multiplication to set `gl_Position` in clip space.

3. **Fragment Shader:** This is the part where most of the work needs to be done. The goal of the algorithm is to compute the visibility factor of the pixel through a sampling process. That sampling happens starting from the pixel position in view space and marching along multiple directions (a user defined number of steps). This marching along directions will result in horizon values that will be averaged in the occlusion factor.

   For the sampling pattern not to produce aliasing artifacts in the final image some randomization needs to be added. Each pixels sampling directions should vary so that the pattern is not predictable.

   Follow the algorithm explained in class (the original paper and presentation are also provided) to implement the technique.

INPUT

With the goal of showcasing your work the best possible way the following specifications need to be meet:

- A first-person camera, controllable by the user via the keyboard and/or mouse.
  - WS: move the camera forward/backward along its forward vector.
  - AD: move the camera left/right along its side vector.
  - QE: move the camera up/down along the **global** up vector (0,1,0).
  - Click + Drag: the mouse should tilt the camera around its yaw and pitch.
  - Up/Down: rotate the camera around the pitch.
  - Left/right: rotate the camera around the yaw.
- Pressing the **Control + R** key should reload the scene from the file.
- Pressing **F5** should reload all the shaders.
- Allow the enabling/disabling of ambient occlusion rendering
- Two render modes: Render ambient occlusion texture/Scene with ambient occlusion applied
- Adjustable parameters for HBAO and blur:
  - Radius scale. This is a scale factor for the radius R. The radius is the distance outside which occluders are ignored.
  - Angle bias. For low-tessellated geometry, occlusion variations tend to appear at creases and ridges, which betray the underlying tessellation. To remove these artifacts, we use an angle bias parameter which restricts the hemisphere.
  - Number of directions. This is the number of randomly-rotated 2D directions in image space distributed around the current pixel. The higher this parameter, the lower is the noise in the ambient occlusion.
  - Number of steps. This is the maximum number samples per direction. The actual number depends on the projected size of the sphere of radius R around the current surface point.
  - Attenuation. This scale factor W0 is applied to the per-sample attenuation function. The occlusion contribution of a given sample is attenuated by $W0 * W(r/R)$ where $W(x) = 1 - x2$.
  - Scale. This value allows to scale up the ambient occlusion values.
  - Amount of blur passes: This is the amount of blur passes applied to the ambient occlusion.

COMMAND LINE ARGUMENTS

The program should allow for the following arguments passed to the main function through the command line arguments.

- `argv[1]`: JSON scene name to load
  - Can be omitted, a default scene (the one provided) should be loaded in that case
- `argv[2]`: Filename of the screenshot to save at the end of the first frame in PNG format
  - Can be omitted and no screenshot is saved in that case
  - Can only be used if the scene is also specified

Example: `cs562_jon.sanchez.exe scene.json screenshot.png`

SUBMISSION

The project will be submitted as a Visual Studio 2019 that will compile and execute without any adjustment. The solution will **only** have the **64-bit platform**, so the 32-bit configuration should be removed. The framework will be created from scratch and will have the following folder hierarchy:

- data -> Folder with all the resources
  - gltf -> Meshes and textures to load **(NOT included on submission)**
  - scenes -> Scene json files **(NOT included on submission)**
  - shaders -> Implementation of the shaders **(INCLUDED in submission)**
- src -> Folder with your code
  - Any header only library to compile (ImGui, stb…)
- cs562_<login>.vcxproj
- cs562_<login>.vcxproj.filters
- cs562_<login>.vcxproj.user
- Required DLLs (only x64 versions)
- include (all include folders for the libraries used)
- lib -> All library files for the libraries used)
- obj -> Folder where the compilation intermediate files when solution is compiled **(NOT included on submission)**
  - Debug
  - Release
- bin -> Folder where the executable will be generated when solution is compiled **(NOT included on submission)**
  - Debug
  - Release
- Solution file (cs562_<login>.sln)
- README.txt -> Text file containing header (see below) as well as controls for the application.

README FILE

Along with your Visual Studio 2019 solution you will attach a README.txt file that will contain the following contents:

1. **How to use your program:** Every control available when executing the program and hints on how to get use it.
2. **Important parts of the code:** Specify source code files and approximate function or lines that were most relevant for the assignment.
3. **Known issues and problems:** Any not completed part of the assignment or known bugs that can be encountered when running the program.

GRADING SCHEME

Section	Item	Points
HBAO	Ambient Occlusion	30
	HBAO Final Image	10
	Randomization	10
Blur	Bilateral Blur	15
Controls	Radius Scale	5
	Angle Bias	5
	Number of Directions	5
	Number of Steps	5
	Attenuation Scale Factor	5
	Contrast/Scale	5
	Blur Passes	5

REFERENCES

Bavoil, L., Sainz, M., & Dimitrov, R. (2008). *Image-Space Horizon-Based Ambient Occlusion.* NVIDIA Corporation.

Paris, S. (2007). *A Gentle Introduction to Bilateral Filtering and its Applications.* SIGGRAPH.