Introduction

Last week, we explored the topic of anti-aliasing by increasing the number of samples. However, you may have noticed that the ray tracer takes an unbelievably long time to run (the Cornell Box is only some 30 triangles). Part of the reason that was the case is the fact that to trace a ray into the scene, we did a ray-triangle intersection against every triangle in the scene. This is costly and unnecessary! In this homework, we will briefly explore the usage of uniform grids and bounding volume hierarchies to accelerate the ray tracer. This will allow us to shoot more rays into the scene and thus increase the number of samples we can take as well as perform refraction and reflection. As always, if you find a bug, please report it on Piazza.

Downloading the Code

To merge in the changes for assignment 7 run the following command in your ray tracer directory:

git pull origin master

Assignment

The only code that you have to modify to complete this homework is located within assignment7/Assignment7.cpp and all the tasks below should be run on the default scene (aside from the task where it explicitly tells you to run it on your own scene...). The code is setup so that you can specify two separate acceleration types: one
for the entire scene, one for each scene object and its mesh objects. To put it briefly, the scene is composed up of multiple scene objects. A scene object is composed of multiple mesh objects. A mesh object is composed of multiple primitives. As a result, the scene’s acceleration structure will partition the scene objects. The scene object’s acceleration structure will partition its mesh objects. The mesh object’s acceleration structure will partition the underlying primitives (e.g. triangles). For this homework, we will primarily tweak the scene’s acceleration structure while keeping the scene object and mesh object acceleration structures fixed.

Part 1

In Assignment7.cpp, find where it says

```cpp
#define ACCELERATION_TYPE 1
```

When this macro is set to 0, the scene will use naive acceleration (it will check every scene object for an intersection). When the macro is set to 1, the scene will use a bounding volume hierarchy. and when it is set to 2, the scene will use a uniform grid. Record the time it takes to run the scene using naive acceleration, a bounding volume hierarchy (BVH), a uniform grid. Note that this macro is only changing the SCENE’s acceleration structure. Each mesh is by default using a BVH, which is why the scene does not take forever to render even with naive acceleration.

Part 2

One of the properties of a uniform grid is its dimensions. Turn on the unifrom grid acceleration and change the suggested grid size. Find the line that says:

```cpp
accelerator->SetSuggestedGridSize(glm::ivec3(10, 10, 10));
```

This line sets the uniform grid dimensions to (10, 10, 10) which is fairly dense for this small and simple scene. Try changing the grid size to (5, 5, 5) and (3, 3, 3). Record the run-time of the ray tracer for all three options. Although you do not have to do it in the homework, you can also change the properties of the bounding volume hierarchy. Namely, you can change how many children each node has—you do this via the functions SetMaximumChildren and SetNodesOnLeaves.

Part 3

Render your own scene. Note that you will want to do put an acceleration structure on your scene as well as your individual scene objects. This piece of code creates and configures an acceleration structure for a scene object:

```cpp
cubeSceneObject->CreateAccelerationData(AccelerationTypes::BVH); cubeSceneObject->ConfigureAccelerationStructure([](AccelerationStructure* genericAccelerator) { 
    BVHAcceleration* accelerator = dynamic_cast<BVHAcceleration*>(genericAccelerator); 
    accelerator->SetMaximumChildren(2); 
    accelerator->SetNodesOnLeaves(2); 
});
cubeSceneObject->ConfigureChildMeshAccelerationStructure( [](AccelerationStructure* genericAccelerator) { 
    BVHAcceleration* accelerator = dynamic_cast<BVHAcceleration*>(genericAccelerator); 
    accelerator->SetMaximumChildren(2); 
    accelerator->SetNodesOnLeaves(2); 
});
```
This piece of code creates an acceleration structure for your scene:
newScene->GenerateAccelerationData(AccelerationTypes::BVH);

Grading

This assignment has the following requirements:

• Show a spreadsheet that contains the run-time from the tasks in parts 1 and 2 (be sure to clearly label what is what).
• Show the test scene rendered using one sample per pixel.
• Show your rendered scene.

according to the following rubric:

• 4 – Show everything.
• 2 – Show everything except your own rendered scene.
• 1 – Only a picture, no spreadsheet.
• 0 – Do not show up to the grading session. :(