Answers for Project 4 Written Questions

CS354, Fall 2003

December 10, 2003

Question 1. How will the running time of your ray tracer change for each of the following changes (will it increase or decrease or remain the same)?
   (a) the window size is increased
   (b) all the objects are increased in size
   (c) more objects are added
   Give a short explanation for each.

Answer 1. (a) If the window size is increased, there are more pixels to color in. For each new pixel, a new ray needs to be traced, and these new rays may also spawn additional shadow, reflection, and refractive rays. Since the number of rays increases, the running time will increase.

(b) If all objects are increased in size, then it is unclear what will happen. If scene objects are visible and do not take up the entire screen, then these objects are now more likely to overlap more screen pixels, thus increasing the number hits and the number of shadow rays that need to be traced. However, if the entire scene is covered with objects (no background visible), then there is not necessarily an increase in the number of shadow tests. Additionally, changing object size may have unpredictable effects on the number of reflection and refraction rays generated (imagine an opaque, non-reflective object covering the whole scene and reducing the count; or a transparent, reflective object covering the whole scene and increasing the count). In a lot of cases, however, the running time increases.

(c) If more objects are added, the running time usually increases. In the new scene, everytime a ray is intersected with all objects in the scene, there are more objects, so the intersection test takes longer. Additionally, there is a higher chance of generating more shadow, reflection, and refraction rays. Some of the pathological cases mentioned in the solution to Part (b) are still present, however. In a lot of cases, though, the running time increases.

Question 2. In this project, you implemented reflection, transparency, and shadows using ray tracing. Out of these effects, which ones are better suited for a ray tracer and which ones are better suited for OpenGL? Be sure to consider ease of implementation and running time for each effect.

Answer 2. All of these effects are better suited for a ray tracer in terms of ease of implementation. OpenGL does not have a clear model for global illumination (reflections and shadows) and the alpha-blending model for transparency does not contain the robustness of different indices of refraction. In a ray tracer, the reflection, refraction, and shadow rays (which produce these effects) make geometric sense and are supported by the basic raytracing infrastructure, such as routines that determine a first hit or trace a ray. Note, however, that with the proper hacks, approximations to all three effects can be implemented in OpenGL by using shadow maps and other tricks.

As for running time, there is a tradeoff between speed and accuracy on both platforms. OpenGL can approximate these effects with relatively higher error, although pipeline is faster than that of a ray tracer. A ray tracer, on the other hand, requires more time but provides a better approximation of these effects.
Question 3. Let a regular lattice for 3-D be defined as a set of identically-sized 3-D cubes that divide up a finite amount of 3-D space. The cubes in the lattice are nonoverlapping and the union of all cubes is the original space. If the entire scene is divided into a regular lattice and if every object is placed in the cube or cubes that it occupies, then how can you use the lattice to speed up your ray tracer?

Answer 3. The lattice reduces the time it takes to perform intersection tests. Under a simple ray tracer, when a ray needs to be tested for intersection, the ray is intersected with all objects in the scene.

By using the lattice, however, the ray is placed in a lattice cell and only needs to be intersected with all scene objects in the current cell. A ray begins in the lattice cell that contains its origin. Since we know all scene objects that overlap this cell, test only these objects for intersection, rather than all objects in the scene. If there are any hits, return the closest one.

If there are no hits, move the ray to the next cell, along its direction. By computing 6 plane intersection tests, it can be determined which of the 6 neighboring cells a ray enters next. Move the ray into the new cell and perform the intersection test for this cell, using only the objects in this new cell. The entire process is repeated from here on until the ray either hits on object or leaves the bounds of the entire scene.

The speedup of using the lattice comes from the fact that in many scenes, a lot of lattice cells are either empty, meaning the ray can move on immediately to the next cell, or contain only a few objects, meaning only a few intersection tests are needed. By creating the lattice, the objects in the scene are sorted according to their location in space, so that a ray is only intersected with objects that are near its path through the scene, rather than intersected with all objects in the scene.

The lattice, which is also referred to as a grid, is one of many acceleration structures used to speed up ray tracing.