A1—Depth Image Approximation of Geometry and Applications

CS 43400, Spring 2013

Due on Monday February 4 at 7am

Summary: implement a program that computes the intersection between a ray and a depth image on the GPU and apply it to the rendering of reflections and geometric surface detail.

1. **Intersection between a ray and a depth image**
    a. Implement a GPU function that takes as input a depth image and a ray, and returns the color of the first intersection between the ray and the depth image, if any.

2. **Application to rendering specular reflections**
    a. Create a scene that consists of a black and white checkered quad, of two side by side bunnies, and of an environment map (see Figure 1).
        i. One bunny is specular (silver or gold), and one bunny is diffuse.
        ii. The bunnies touch the checkered quad and they intersect each other as shown.
        iii. Use the bunny model provided (bunny.tris).
            1. It is a text file; you can look at it with a text editor, e.g. WordPad.
            2. The first line gives a description of the per vertex data, in this case coordinates “xyz” and texture coordinates “tcs”; thus, in this case, there are 5 floats per vertex: x, y, z, s, and t, where x, y, and z are the 3-D coordinates of the vertex and s and t are texture coordinates.
            3. The second line gives the number of vertices, i.e. here 35,947.
            4. The vertex data follows, one line per vertex.
            5. The number of triangles is next, here 69,451.

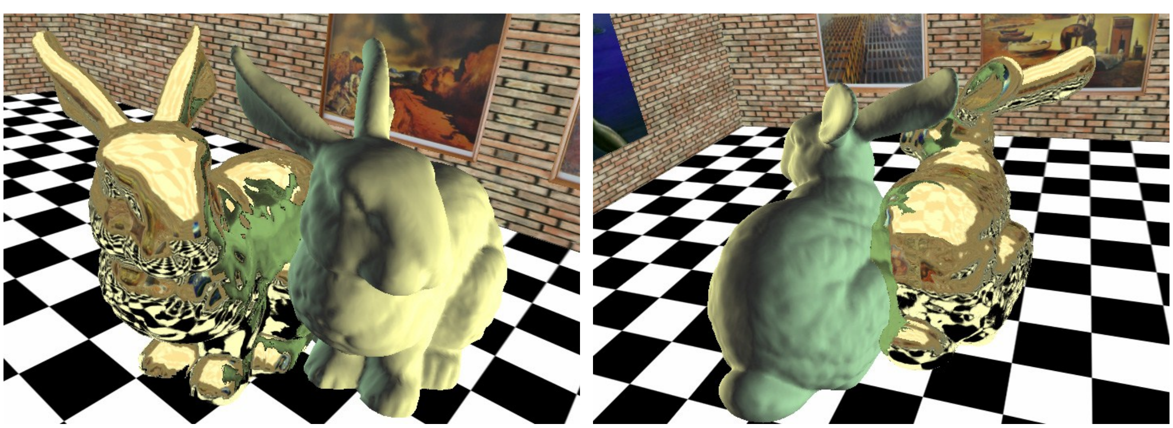

Figure 1. Illustration of scene to be used for the reflection rendering application.

6. The connectivity data follows, one line per triangle; a triangle is defined by a triple of indices in the vertex array.
7. The file ends with the string “eof”, which can be used to check whether the file was read correctly.

iv. Use the environment map provided (uffizi_cross.tiff).

b. Reflections on the specular bunny should be computed by intersecting the reflected ray of a pixel with the depth image approximating the diffuse bunny, with a billboard modeling the floor quad, and with the environment map.
   i. The bunny depth image does not change and thus it can be precomputed.
   ii. The bunny depth image should have a resolution of 640x480.
c. Camera navigation
   i. The user is able to navigate freely in the scene: 6 degrees of freedom, 3 rotations (pan, tilt, roll) and 3 translations (up-down, left-right, forward-backward)
   ii. The user is also able to revolve the camera around the two bunnies.
      1. The camera eye moves on a circle parallel to and above the floor quad.
      2. The look at point is fixed to the center of mass of the two bunnies.
      3. The trajectory radius and the field of view of the camera are chosen such that the two bunnies are seen completely at all time as the camera revolves.
d. Make a 30s 30Hz 720p movie during which the camera makes a complete revolution.

3. Application to rendering geometric surface detail

a. Create a scene that consists of a coarsely tessellated cylinder. Each rectangular facet of the cylinder is texture mapped with geometric detail using a depth image. See Figure 2.
   i. Cylinder parameters that can be set through the user interface: radius, height, number of sectors, and number of slices.
   ii. Depth image of Happy Buddha statue (happy2.bin).

Figure 2. Surface geometric detail mapping on a coarsely tessellated cylinder.

b. The geometric detail is rendered using eye ray / depth image intersection.
c. Interaction
    i. View is fixed, it shows a bit of the top base of the cylinder (see Figure 2).
    ii. The user can spin the cylinder about its vertical axis.
    iii. The user can also change the height of the surface geometric detail.
d. Make a 30s 30Hz 720p video during which the cylinder spins 360^0.

4. Extra credit

a. Include text file explaining extra credit attempted.
b. Include videos illustrating each extra credit feature attempted.
c. Morphing of reflecting bunny into a sphere (1%).
d. Morphing of reflected bunny into a sphere, which requires updating the depth image on the fly, for each frame (2%).
e. Acceleration of ray / depth image intersection (3%).
f. Color modulation of surface geometric detail as shown in Figure 2, left (1%).
g. Support for non-height field surface geometric detail, as shown in Figure 2, right (3%).
h. Shadow mapping of surface detail, with moving light, see Figure 2 (3%).

5. Turn in

a. Videos
b. Source files
c. Executables
d. Description of user interface
e. Extra credit as described above
f. Turn in a single zipped archive with all your files via Blackboard

---