MAT 594G / CS 280: Advanced Computer Graphics	Oct. 11, 2012
Fall Quarter, 2012	Assignment 1

Due by **midnight, the evening of October 23, 2012** via GauchoSpace.

1 RenderMan Shading

In this assignment, you will implement some RenderMan shaders and become familiar with the RenderMan Shading Language (RSL). There is no C programming for this assignment, as I have provided you with the RenderMan Interface Bytestream (RIB) files that would be output by a C program. Pixar's official documentation for RenderMan is available from multiple sources online such as here and here. Further tutorials are also available in RenderMan University, and the RenderMan Repository. And finally, when in doubt, consult the RenderMan Interface Specification.

2 Procedural Texturing

Most of the enclosed RIB files contain a sphere with default `(u,v)` coordinates assigned to it. The `u` coordinate runs longitudinally (Figure 1(b)), where black is `u = 0` and white is `u = 1`. The `v` coordinate runs latitudinally (Figure 1(c)), where black is `v = 0` and white is `v = 1`.

- Implement a checkerboard shader and use it on the sphere. With `frequency = 10`, you should get something that looks like Figure 1(a).
- Instead of using the checkerboard texture to change the diffuse color, use it to modulate the shininess of the specular component (Figure 1(f)).
- Instead of producing a checkerboard by superposing two square waves, make a "polka-dot" texture by modifying your checkerboard shader (Figure 1(d)).
- The jaggies are pretty bad on the polka-dot shader, so smooth it out according to the distance to the center of each polka-dot using the `smoothstep` function (Figure 1(e) – red is optional). Now the sphere looks like it has measles.
- Deal with the parametric distortion problems from the `(u,v)` mapping by instead creating a *solid texture* that depends on the `(x,y,z)` components of the spatial position `P`. Make a solid texture that consists of *packed spheres*. (Figure 1(g), Again, colors are optional.)

  Imagine all of space is a solid block of material whose insides consist of spheres packed together in a regular, polka-dotted grid. You carve away at this block until just a bunny is left. The color of each surface point should be determined by the color of the material at that point inside the original solid block.

- Convert your measles shader into a displacement shader to obtain a spiky sphere, as shown in Figure 2(a). For large displacements, you may see artifacts appear, such as the missing chunks in Figure 2(a). Read up on displacement shaders and figure out how to modify your RIB file (not your RSL file!) to eliminate this artifact.

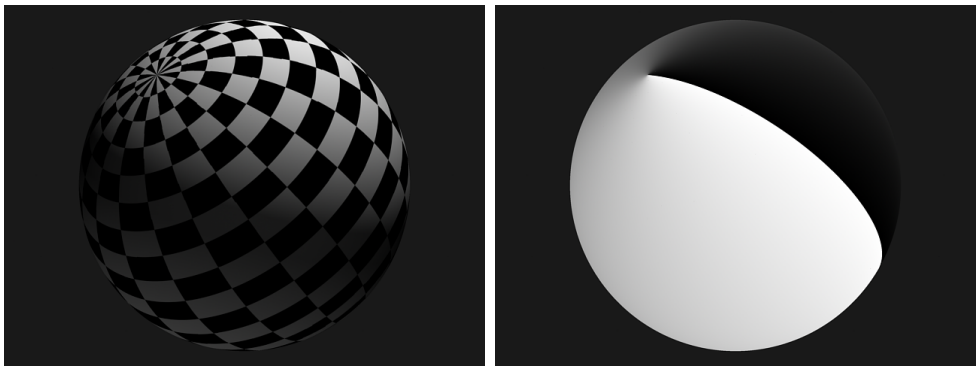
3 Points Breakdown:

For each item, there is a corresponding RIB and SL file. When run, it outputs a TIFF file of the same name. Modify each SL file appropriately, and turn in the RIB file, SL file, and the output image you get when you run it to GauchoSpace. **In your final SL files, set `frequency` to something greater than 10.**

1. Implement the checkerboard pattern. **(checker.sl, checker.rib) (5 points)**
2. Implement the shiny checkerboard pattern. **(shiny.sl, shiny.rib) (5 points)**
3. Implement the polka-dotted sphere. **(polka.sl, polka.rib) (10 points)**
4. Implement the measles sphere. **(measles.sl, measles.rib) (10 points)**
5. Implement the solid textured bunny. **(bunny.sl, bunny.rib) (10 points)**
6. Implement the spiked ball. **(spike.sl, spike.rib) (10 points)**

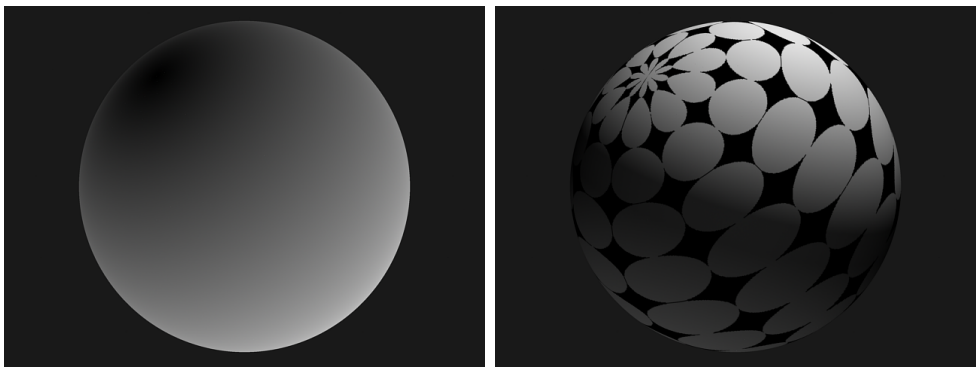
4 Debugging Tips

You may find it tricky at first to work with the sphere parameterization, so I have also provided a debugging scene containing a square with the simplest possible surface parameterization **(debug.rib, debug.sl)**:
Copy and paste whatever shader you are working on into **debug.sl**, compile it, render **debug.rib**, and take a look at **debug.tif** to figure out what's going wrong.



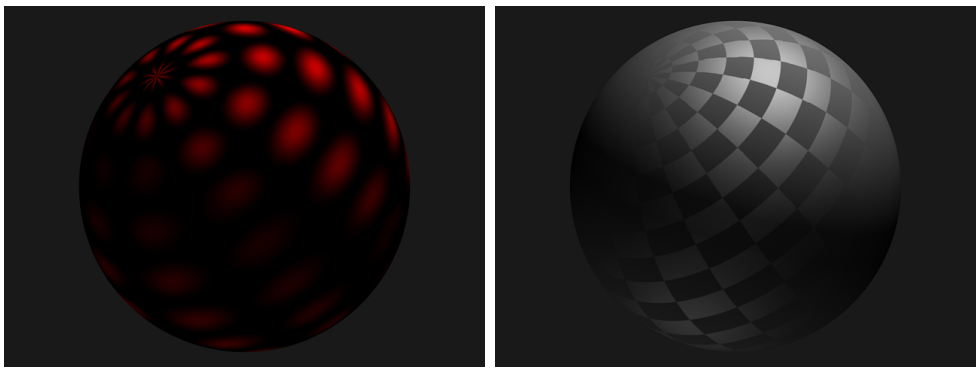
(a) Checkerboard sphere

(b) `u` texture coordinate



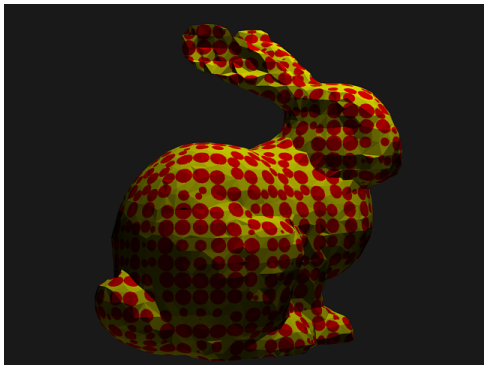
(c) `v` texture coordinate

(d) Polka-dotted sphere



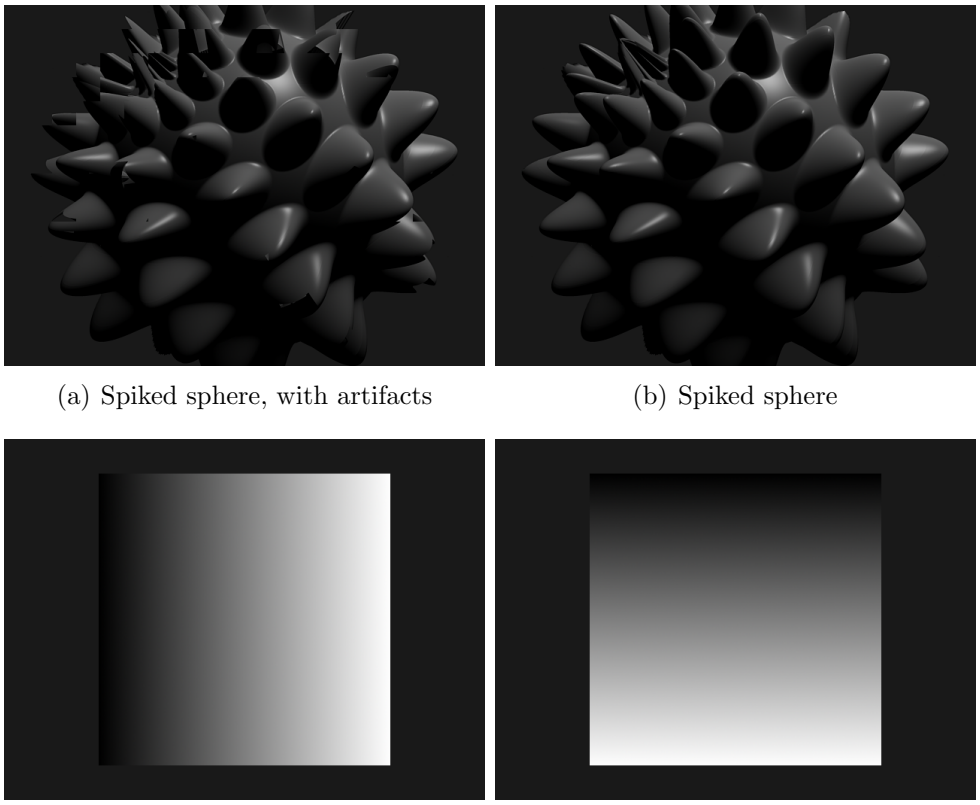
(e) Measles sphere

(f) Shiny checkerboard sphere



(g) Solid textured bunny

Figure 1: Example outputs



(a) Spiked sphere, with artifacts

(b) Spiked sphere

(c) Square `u` texture coordinate

(d) Square `v` texture coordinate

Figure 2: Top: More example outputs. Bottom: Debugging outputs.