CSC 433 / 533: Spring 2017
Assignment 3

Assigned: Tuesday, March 21st
Due: 11:59pm Thursday, April 6th

For this assignment you extend your object viewer from assignment 2. The primary focus will be on fragment shaders, although there are some changes needed in your main program. You will develop the capability of applying two shaders to objects: (1) Blinn-Phong with diffuse texture maps and (2) TOON shader – a non-photorealistic rendering shader. Your program will be graded on the Mac workstations on the 9th floor in the Gould-Simpson building.

As before:

1. Provide a Makefile that builds the program. We will not grade programs that do not compile.
2. Your directory needs to contain all source code for both the application on the CPU and shader files for the GPU, and required data files.
3. Report any errors to stderr and provide a usage message when the program is invoked with no scene file name.
4. Include a readme.txt file and include any special instructions or assumptions.
5. Comply with the University of Arizona Code of Academic Integrity. Review the course policies at https://www2.cs.arizona.edu/classes/cs433/spring17/policies.html

Tasks:

1. **viewer**
   a. In your makefile create an executable named ***viewer***.
   b. Input to the ***viewer***:
      a. The program takes one parameter which is the scene file name, for example

         viewer texturedCube.txt

      b. The format for the scene file required some extensions and the new format is described later in this document.
   c. Read and display geometry described in the wavefront obj format. These files must be in ASCII format and the polygons must be triangles, just as with assignment 2.
   d. Support two shaders; more information is in the next section.
   e. The format for the scene file is changed to allow for two shaders, BLINN_PHONG and TOON.
   f. Keyboard commands are just as in assignment 2.
2. Shaders
   a. Blinn-Phong – add the capability of diffuse color texture mapping. Modify the Blinn-Phong shader from assignment 2.

   i. Two previously unused fields in the objInfo structure were intended to use for this capability: `char diffuseTex[128]` which is the name of the diffuse color texture map and `int diffuseTexMapID` which can be the texture unit identifier for this texture map.
   ii. The file name of the texture map will be loaded by the loadObjFile function, and this file name is specified in the associated material file.
   iii. OpenGL texture mapping for a specific image file is <u>**not**</u> set up in loadObjFile. You can set up texture mapping in either your main program or by modifying loadObjFile. I do this in my main program, but you could reasonably do this in the loadObjFile function, too. After an OpenGL texture map is established, you can store the texture unit ID in `diffuseTexMapID`.
   iv. Minor changes are needed in the vertex shader. You will need to receive and send down the pipeline the texture map indices. If texture map indices are specified in the OBJ file, you will get a data buffer of 2D texture map indices.
   v. In the fragment shader you will need to use the texture map indices (that have been bi-linearly interpolated by the rasterizer) and set a uniform variable to tell the fragement shader if this is a texture mapped surface. If it is a texture mapped surface, use a sampler to get the diffuse color, and use that color rather than the surface diffuse color specified.
   vi. Texture maps can have transparent values. If a color returned by the sampler has an alpha value of 0, use the GLSL statement `discard`; to leave the fragment shader without setting a color.

b. TOON shader (cartoon shader and also called cel shading) – a non-photorealistic renderering technique where the basic idea is to have large areas of constant color with obvious transitions. It is an approximation to the way an artist might shade an object using strokes of a brush.
   i. We are not including texture mapping in this model.
   ii. We are not including attenuation in this model.
   iii. In our approach, the lighting model has only ambient and diffuse components, so there is no specular component. The ambient and diffuse terms are the same as in the classic lighting model. Ka is a constant Kd is multiplied by dot(N,L), and both of those terms are multiplied by the light color.
   iv. However, in the fragment shader and we want to quantize the term dot(N,L) into 7 levels. For 4 levels the following table describes the levels.

Cosine of angle between L and N	Value used
Between 1 and 0.75	0.75
Less than 0.75 and greater than or equal to 0.5	0.5
Less than 0.5 and greater than or equal to 0.25	0.25
Less than 0.25 and greater than or equal to 0.	0.

For n levels, the highest break is (n-1)*(1/n), the next break is (n-2)*(1/n), and so on until when less than 1/n the value is 0.

        v. Use this lighting model on the first light described in the scene file, and ignore other lights.
        vi. Allow for both directional and local lights, however not spot lights.

1. **Hints**
    1. Set Boolean flag “doingTextureMaps” in loadObj.cpp to true for this assignment.
    2. Modified scene file format to support two shader choices.
    3. Add processing of diffuse color texture maps based on map_Kd field in OBJ material file.
        a. loadObjFile function reads map_Kd field in the material file and passing texture map file name back in objInfo structure.
        b. Read variety of image files using SOIL (Simple OpenGL Image Library)
        c. Need to flip image file. Depending on which function used in SOIL, flip either with a parameter in the call or with your own code.
        d. http://www.fileformat.info/format/material/
            1. Map-Ka (ambient) map-Kd (diffuse) map-Ks (specular) map-Ns (scalar) map-d (scalar) … bump (bump map) … disp (displacement map)
    4. Add texture mapped (map-Kd) surfaces in Blinn-Phong shader
    5. Use 8th edition of Red Book to read about textures, 9th edition uses some OpenGL 4.5 functions.

2. **Grading rubric – 100 points total**
    1. [50 points] Rendering objects with texture maps using BLINN_PHONG shader.
        a. [10 points] render one object with a single texture map
        b. [10 points] render one object with a single texture map that includes transparency. Some image files support transparency, such as PNG format, while others do not, such as JPEG. Files in PNG format can include a fourth channel, an alpha channel, ranging from 0 to 255, where 0 is transparent.
        c. [10 points] render one object with multiple texture maps
        d. [5 points] render multiple objects each with a single texture map
        e. [5 points] render multiple objects each with either a single texture map or multiple texture maps
        f. [10 points] Rendering objects without texture maps, e.g., ensure that capability still works.
    2. [20 points] Rendering objects with TOON shader.
        a. [10 points] render one object.
        b. [10 points] render multiple objects.
    3. [20 points] Rendering with both shader types in one scene
        a. [10 points] a scene with one object, without texture, with BLINN_PHONG shader and one object with TOON shader.
        b. [10 points] a scene with one object, with texture, with BLINN_PHONG shader and one object with TOON shader.
    4. [10 points] Create a custom scene with the following characteristics
        a. A minimum of 4 OBJ files. Each OBJ file must be 1) different (i.e., not all teapots), 2) must not be ones provided with these assignments, and 3) must be texture mapped.

    Test cases

    1. Can include up to 4 lights of any of the three types from assignment 2
    2. Texture maps may contain transparent sections

Scene File Format

(shader field modified for assignment 3)

An overview of the format is a collection of sections that are specified with keywords and values. The sections are 1) view, 2) lights, and 3) objects. Allow for a maximum of 4 light sources.

View section: keyword **view**

eye <vec3>
center <vec3>
viewup <vec3>

Lights section: keyword **light**

type [local|spot|directional]
ambient <vec3>
color <vec3>
position <vec3>
constAtt <float>
linearAtt <float>
quadAtt <float>
coneDirection <vec3>
spotCosCutoff <vec3>
spotExponent <vec3>

Objects section:

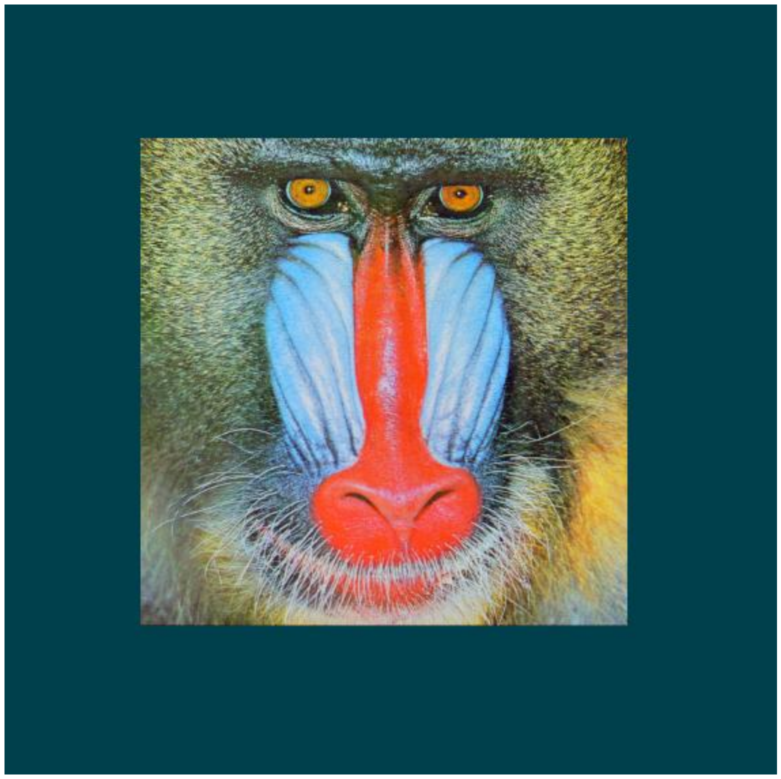
object <filename of Wavefront OBJ file>
shader [Blinn_Phong|TOON]
[
rx <angle (degrees)> **ry** <angle (degrees)> **rz** <angle (degrees)>
t <vec3>
s <vec3>
]

A light description can have a variable number of lines, depending on the type of light. Position is actually direction for a directional light source, and actual location for spot and local light sources.

Modeling transforms for objects are not required, in which case the modeling transform for the object will be the identity matrix. The transformations are to be applied in the order listed in the file. There is not limit of the number of canonical transforms that can be applied.

Use forward slashes for paths to OBJ file descriptions.

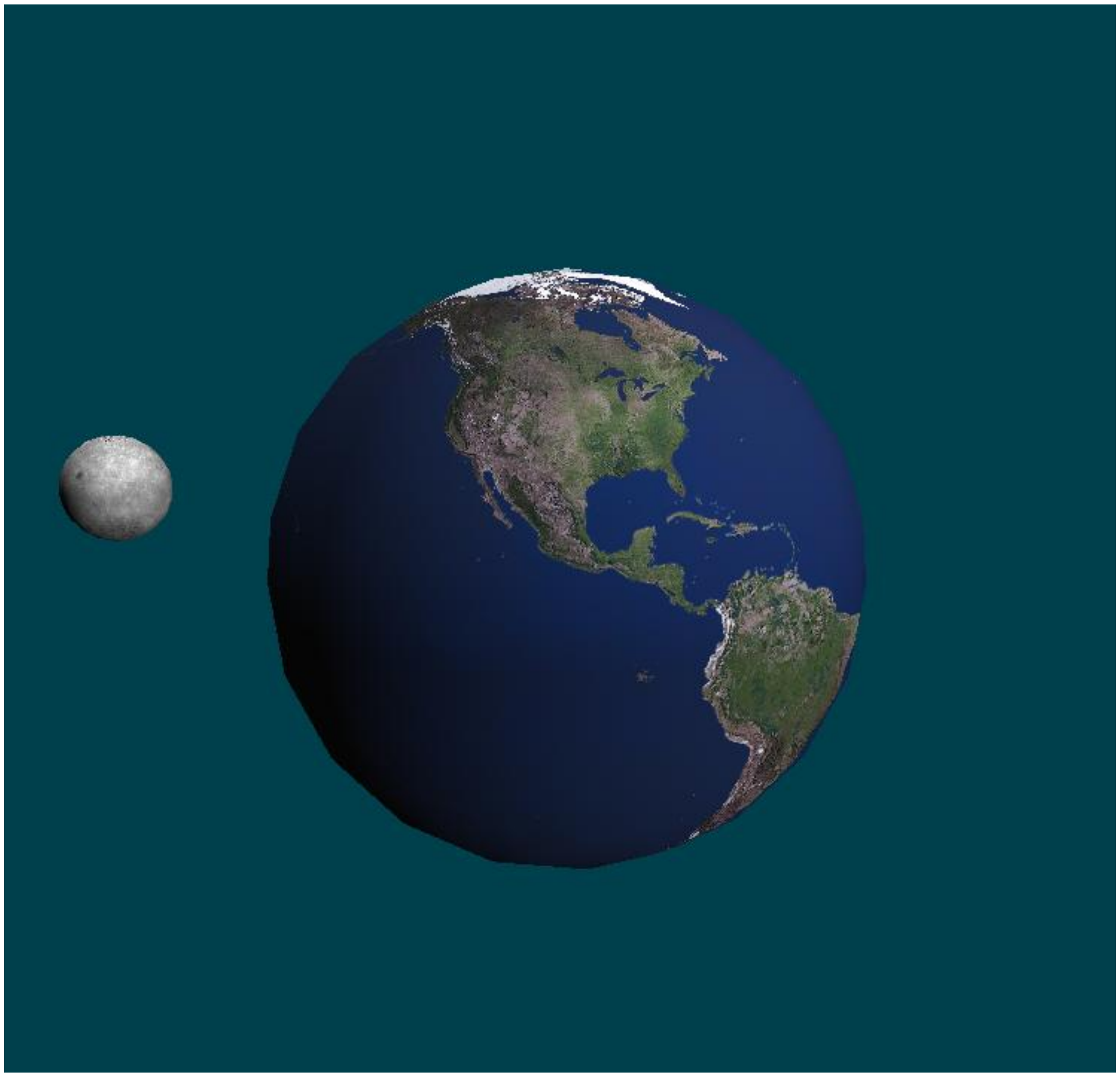
Sample output



```
view
eye .5 .5 2.
center .5 .5 0.
viewup 0. 1. 0.

light
type directional
ambient .1 .1 .1
color 1. 1. 1.
position 0. 0. 1.

object OBJfiles/square.obj
shader BLINN_PHONG
```



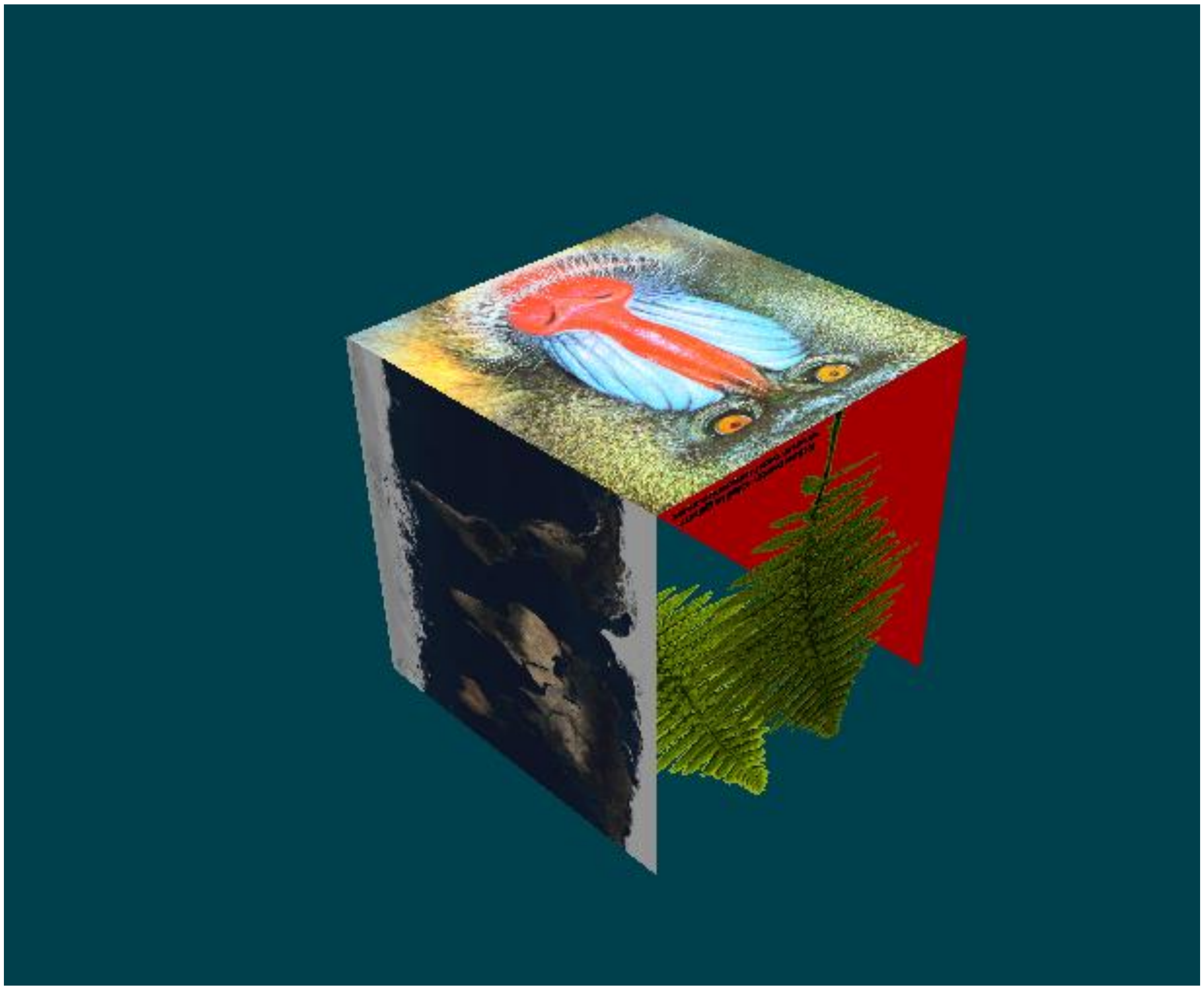
```
view
eye 5. 5. 2.  center 0. 0. 0.  viewup 0. 0. 1.

light
type directional
ambient .1 .1 .1  color 1. 1. 1.  position 0. 1. 0.

light
type directional
ambient .1 .1 .1  color 1. 1. 1.  position 0. 0. 1.


object OBJfiles/solarSystem/earth.obj
shader BLINN_PHONG
rx 90 rz 270
s 2. 2. 2.

object OBJfiles/solarSystem/moon.obj
shader BLINN_PHONG
rx 90
s .25 .25 .25
t 3. 0. 1.
```



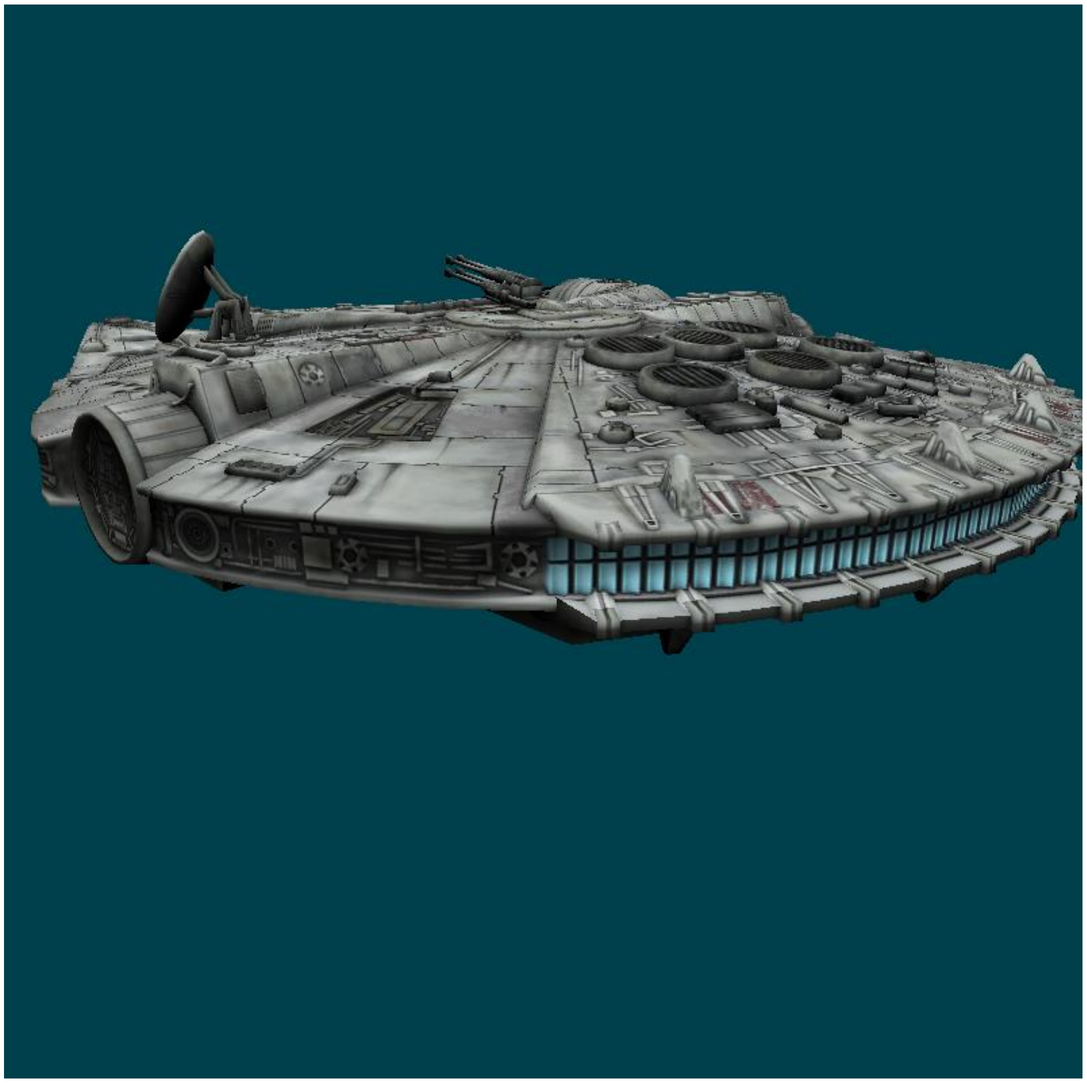
```
view
eye 2 2 2 center 0 0 0
viewup 0. 0. 1.

light
type directional ambient .2 .2 .2 color 1. 1. 1.
position 1 1 1

light
type directional ambient .2 .2 .2 color 1. 1. 1.
position -1 -1 1

light
type directional ambient .2 .2 .2 color 1. 1. 1.
position 0 0 -1


object OBJfiles/texturedCube/cube_with_texture.obj
```

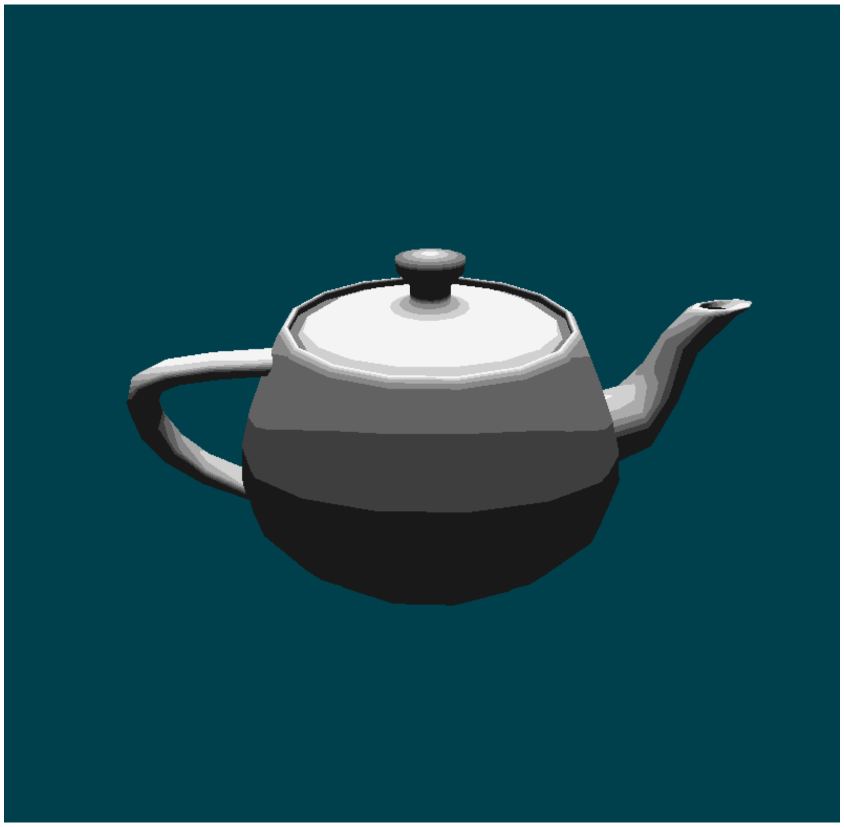


```
view
eye 400 400 200
center 0. 0. 0.
viewup 0. 0. 1.

light
type directional ambient .1 .1 .1
color 1. 1. 1.  position 0. 0. 1.

light
type directional ambient .1 .1 .1
color .5 .5 .5    position 4 4 3

object
OBJfiles/Millennium_Falcon/Millennium_Falcon.
obj
shader Blinn_Phong
rx 90.
```

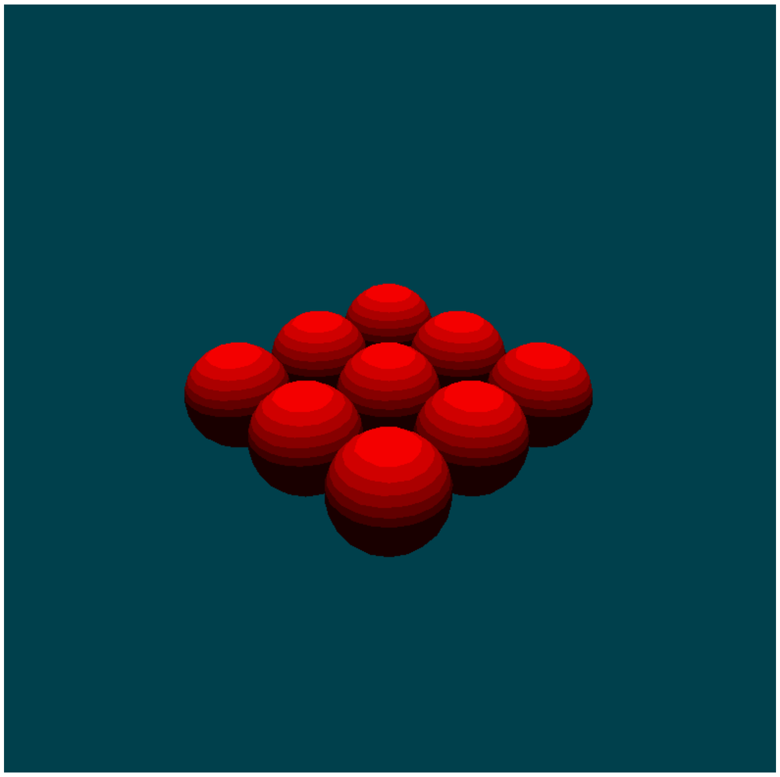


```
view
eye 150. 150. 75.
center 0. 0. 0.
viewup 0. 0. 1.

light
type directional ambient .1 .1 .1
color 1. 1. 1.  position 0. 0. 1.

light
type local
ambient .1 0 0 color 1. 0 0 position 25.
25. 40.
constAtt 0.01 linearAtt 0.01 quadAtt .002

object OBJfiles/teapot.obj
shader TOON
rx 90
```



```
view
eye 10. 10. 7. center 3. 3. 1.
viewup 0. 0. 1.

light
type directional ambient .1 .1 .1
color 1. 1. 1.  position 0. 0. 1.

object OBJfiles/sphere2.obj
shader TOON

object OBJfiles/sphere2.obj
shader TOON
t 2 0 0

object OBJfiles/sphere2.obj
shader TOON
t 4 0 0

object OBJfiles/sphere2.obj
shader TOON
t 0 2 0

object OBJfiles/sphere2.obj
shader TOON
t 2 2 0

object OBJfiles/sphere2.obj
shader TOON
t 4 2 0

object OBJfiles/sphere2.obj
shader TOON
t 0 4 0

object OBJfiles/sphere2.obj
shader TOON
t 2 4 0

object OBJfiles/sphere2.obj
shader TOON
t 4 4 0
```

Submission Instructions

Submit your files from the host **lectura.cs.arizona.edu** using the command
turnin cs433s17-assg3 [files]

Assignment Advice

1. Start early.
2. Do your own work.
3. Check piazza regularly.