CS606: Computer Graphics / Term 2 (2022-23) / Programming Assignment 3 & 4
International Institute of Information Technology Bangalore

Announcement Date: Apr 11, 2023 (Tuesday)
Submission Deadline: 11:59 pm IST, Apr 28, 2023 (Friday)

Summary:

- **(A3)** Lighting & texture mapping – 5% of final grade
- **(A4)** Animation – 15% of final grade

Learning Objectives:

- [A3] Implementing Gouraud and Phong shading models, and toggle between them.
- [A3] Setting up local illumination with multiple light sources using Blinn-Phong illumination models.
- [A3] Texture mapping experiments with checkerboard.
- [A4] Modeling a dynamic scene with a Scene Graph and relative transforms
- [A4] Procedural animation of a scene including collision detection
- [A4] Dynamic lights and camera

Deliverables:

- Separate submissions shall be done for A3 and A4.
- Each team member shall own up a distinct part of both A3 and A4. Both A3 and A4 are also conveniently split into 3 objectives each to allow each student to own an objective from both.
- Both A3 and A4 will have a joint demo and viva component with both instructors.
    - The demo will involve a code walk-through by individual contributors.
- A3 requires the submission of code and a folder of screenshots on LMS.
- A4 requires the submission of code, a folder of screenshots of a demo in video format on LMS. The video demo can focus on the animation generated.
- It is expected that for both A3 and A4, the team shall integrate the individual parts to submit joint work.
    - It is sufficient that the team representative makes a single submission on LMS for the entire team.
    - All team members may also submit individual contributions that are over and above the joint work as separate submissions through LMS, but this will be evaluated only after the concerned students mention the same during the demo with the instructors.
- The evaluation for A3 and A4 is based on the submissions on LMS

A3 - Lighting and Texture Mapping

The objective of this assignment is to set up lighting, implement local illumination shading models, and texture mapping using a checkerboard pattern. The implementation of A3 is to be considered as a sandbox/experimental testbed for that for A4. There are 3 tasks in this assignment.

Model: Take 9 different spheres of the same size and color but different optical properties in an arrangement as shown in Figure 1. These are a set of objects in a scene in A3. Repeat the same exercise with a cylinder, thus creating a second scene. Keyboard controls may be used to toggle between the scenes.

Task-1 (Shading Model):
Implement Gouraud and Phong shading using vertex and fragment shaders, respectively. Gouraud shading may be set as the default shader for all meshes. Keyboard control is to be used to toggle between Gouraud and Phong shading.

Task-2 (Local Illumination Model):
Implement Blinn-Phong (local) illumination model to perform lighting using 2 or more light sources. Here, use scenarios of (a) 1 light source (to reproduce the result in Figure 1), and (b) light sources. These light sources are stationary.

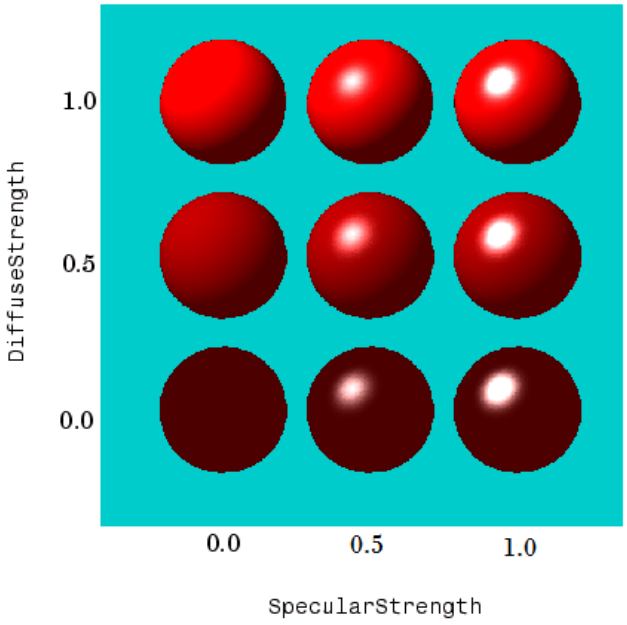
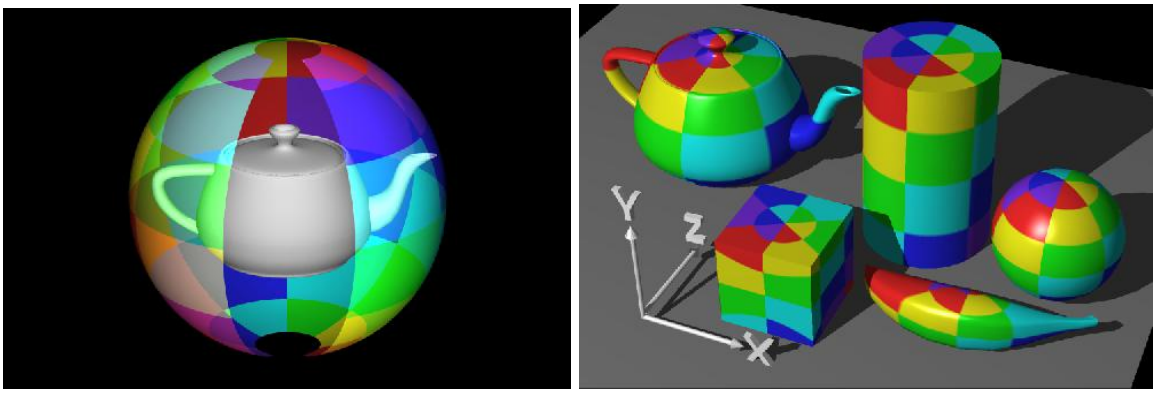


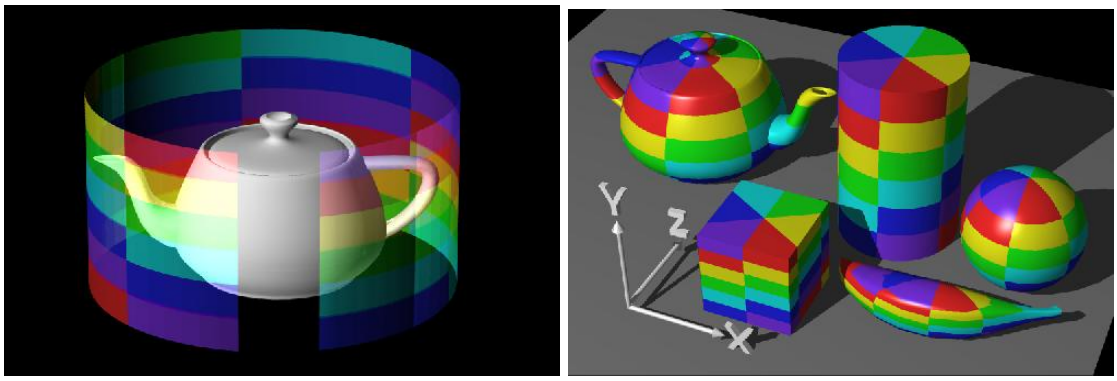
Figure 1: Results rendering using lighting with varying specular and diffuse reflectances.
Image courtesy: https://in.mathworks.com/help/matlab/creating_plots/reflectance-characteristics-of-graphics-objects.html

Task-3 (Texture Mapping):

Use the checkerboard texture[1] and perform both spherical and cylindrical texture maps for different objects, given in the "Models". The rendering of objects in "Models" should look as given in Figure 2.



Spherical Mapping



Cylindrical mapping

Figure 2: Texture mapping rendering using spherical and cylindrical mapping.

Image courtesy: Wolfe, R. (1997). Teaching Texture Mapping Visually. *ACM Computer Graphics*, *31*(4), 66-70.

Implementation notes:

1. The students are strongly encouraged to use the Javascript library Three.js in this assignment. This library provides high-level objects such as scene objects, mesh objects, lights, cameras, etc. The properties and others may be manipulated using high-level operations to achieve your objectives. This implementation is to be reused for A4.

[1] The texture can be generated as given in the function `makeCheckImage` in https://www.glprogramming.com/red/chapter09.html or can be downloaded as a high-resolution image. Programmatic generation of the image gives more control for getting high-resolution images without the quality drop owing to compression in different image formats, and also, opportunities to experiment with different image sizes for the texture.

A4 - Chain Reaction Animation

The objective of this assignment is to simulate a "chain reaction" machine or activity, where a simple movement of one object in the scene sets off a chain reaction that generates a sequence of movements of other objects to achieve a desired goal or effect. A simple example is Newton's Cradle". More general examples include the "Rube Goldberg Machine", "Domino Toppling", "Perchang Game", etc. These typically require the application of simple but precise physics principles to achieve their chaining of reactions.
For some fascinating (and very advanced) demonstrations, see Honda - The Cog, NVIDIA - Marbles at Night, Amazing Domino Chains, etc.

We will attempt something less ambitious and more achievable!

The animation to be designed comprises N reaction steps (N >= 3), where each step is one action that eventually triggers another action on a different object. Examples of steps are a ball rolling till it hits another ball (which then moves), an object dropping till it hits a lever (which then rotates), a set of balls falling into a cup suspended by a string (till the cup becomes heavy enough to break the string), a domino falling onto another domino (which then falls), etc.

Task 1: Model and Scene Graph
You can use any geometry, including simple spheres, cylinders, and cuboids, as the moving/rolling objects, and any combination of mechanisms/arrangements. Use textures and lighting/shading models from A3 to color the objects.

Use a scene graph to model the objects in the scene, and especially to manage dependencies between objects. At least one step in the animation should involve the motion of one object constrained by (relative to) the motion of a second object. And this should be modeled as a dependency in the scene graph.

Task 2: Animation including Collision Detection
Implement each animation step in 3 parts:

1. animation of one or more objects - rolling, falling, etc. The motion may be constrained by other objects (e.g. a ball rolling down a face of a box)
2. detection of collision with another object
3. reaction to the collision - on the first object and on the second, which then executes step 1 above

Task 3: Dynamic Lights & Camera

The scene is lit by a set of 3 lights:

- a point source at a fixed location and illuminating the entire scene
- a directional spotlight fixed at a certain height on one side of the scene and lighting the middle of the scene
- a moving spotlight that follows the object that is currently moving (the focus of the animation at that point). This spotlight is at a fixed location but points toward the moving object

Each of these lights can be toggled on or off. Use different colors for each of the lights, so that their effect is visible.

The scene can be viewed through one of 2 cameras:

1. A camera located above the scene, and can be rotated about the scene using a trackball mechanism
2. A camera that is fixed a certain small distance above one of the moving objects and follows the (translation) motion of the object. The camera can be rotated about a vertical axis through user commands.

Controls allow switching between the camera modes.

Implementation notes:

1. This assignment is structured to enable the maximum reuse of designs and code from A2 and A3. The students should use techniques such as rotations using quaternions and trackball, mechanisms for keyboard controls, shaders, etc.
2. Note that effects such as a ball rolling down an incline can be computed as the (constrained) translation of the ball down the incline, and a rotation at a rate proportional to the translation speed
3. For detecting collision between a moving object and a mechanism, you can build in simplifying assumptions about where and how to check for collisions, so long as the actual collision detection is done at run-time based on the position of the relevant objects.
