Assignment 4

Sai Sri Harsha Vallabhuni(IMT2016073)

May 2019

1 Problem Statement

Outline:

- Phase 1: Design a scene with multiple models and with multiple light sources. Use texture maps to render the model objects, using different schemes for generating texture parameters/coords.
- Phase 2: Define a scene graph to manage these models. Animate the objects on the lines described below, using interactive controls to vary the animation. Manage lights and camera settings to create the effects described below.

Details of Phase 1:
Create a scene comprising of a floor/ground and at least 4 objects. The floor/ground is a plane. Each of the objects is a triangulated surface model, read in as a ply file or equivalent, such as those at `https://people.sc.fsu.edu/~jburkardt/data/ply/ply.html`. References to ply files below are to models available at this link.

1. The objects should be of different geometrical shapes, with at least one each of the following:
   (a) An object with large flat areas such as a cube (cube.ply) or a table or an icosahedron (icosahedron.ply)
   (b) An object with cylinder-like symmetry and shape, such as canstick.ply, ketchup.ply etc

   (c) An object which is approximately spherical: sphere.ply, apple.ply, skull.ply, etc

   (d) An object with arbitrary geometry such as streetlamp.ply, head1.ply, shark.ply, galleon.ply etc

2. Identify at least 4 texture images, covering the following kinds of patterns:

   (a) Wood, fur, bricks or similar pattern that will be replicated over a surface

   (b) A well-defined geometric pattern such as a checkerboard

   (c) A photograph with a person's face clearly visible (a selfie would work!)

   (d) Map or image of the surface of the earth, mars or other planet, with moderate level of detail

3. Arrange the model objects on the floor/ ground, sufficiently far apart.

4. Set up 4 point sources of light, positioned at different locations, and each shining directly on one of the objects. Lights can be turned on/off individually. Use keys/mouse-clicks (or GUI controls) to turn lights on/off. For example, hitting numbers 1-4 will toggle the state (on or off) of that light. Note that each light points to one of the model objects.

5. Each of the objects should be render-able with each of the texture images and with each of the mapping schemes listed below. Use the following schemes to generate texture coords for vertices:

   - A cylindrical map
   - A spherical map
   - An arbitrary pre-image mapping (such as that computed by OpenGL as a default mapping)

6. Use keys/mouse-clicks or GUI controls to change the textures and texture coordinates mappings. For instance, hitting the "t" key could cycle through the set of textures and apply a different one each time. Hitting the "m" key changes the mapping used (to generate texture coords).

All objects (apart from the floor) are rendered with the same texture image and mapping at each step. (Other interaction mechanisms, such as GUI controls, are welcome).

Details of Phase 2:
Add animation to the scene and introduce controls as described below.

1. Design and implement a scene graph and add the models to this, The structure of the scene graph should help the implementation of program features described below. Rendering of the scene for each frame should be implemented by rendering this scene graph. (You might find it useful to set up the scene graph as part of Phase 1, to reduce subsequent effort)

2. Add animation to the objects in the scene as follows:Assume the 4 objects are named A, B, C, D. The objects move as follows:

    - Object A is fixed with respect to the floor
    - Object B moves in a circular (or similar path) around object A, and wobbles left and right as it traverses this path
    - Object C starts some distance away from B, and tries to move towards B, constantly adjusting its direction depending on the current position of B. The speed of C is roughly half that of B
    - Object D sits on top of C, and jumps up and down while staying on top of C.

3. The four lights used in Phase 1 now track their target objects, as they move around.

4. The speed of the objects can be increased or decreased uniformly, while maintaining the relative ratios of their speeds. For example, hitting the up and down arrows could speed up or slow down all the objects by some factor.

5. The textures of the objects can be modified as in Phase 1

6. Clicking on an object (picking it) causes it to start spinning about a vertical axis through its “center”, while it continues its movement as above. Picking it again stops the spinning.

2 Approach

2.1 Phase 1

Selected Models:

1. Aeroplane(arbitrary geometry)
2. Apple(spherical)
3. Cube(large flat area)
4. Canstick(cylinder-like symmetry)

Texture Images:

1. Wall
2. Checkerboard
3. Leonardo DiCaprio face image
4. Worls map

Generating Texture buffers:

- Created texture buffers with the above mentioned images. Library stb_image is used to read the images.
- Each model has four texture buffers. Which texture to apply can be controlled. By selecting the model and clicking 'T' key changes texture to all the images.

Generating Texture coordinates:
Sphere parametric equation:

$$p(\theta, \varphi) = (rsin\theta cos\varphi, rsin\theta sin\varphi, rcos\theta)$$

where
r = radius,
θ = angle from z-axis($0 \leq \theta \leq \pi$),
φ = angle from x-axis($0 \leq \varphi \leq 2\pi$)

Spherical Texture coordinates (s, t)

$$s = \varphi/2\pi = cos^{-1}(z/r)\ /\pi$$
$$t = \theta/\pi = cos^{-1}(x/rsin(t\pi))\ /2\pi$$

Cylinder side surface equation:

$$p(\theta, y) = (rsin\theta, rcos\theta)$$

Cylindrical Texture coordinates (s, t)

$$s = \theta/2\pi = tan^{-1}(x/z)\ /2\pi$$
$$t = (y + h/2)/h$$

So s varies from 0 to 1 as π changes from 0 to 2π, t varies from 0 to 1 as y changes from $-h/2$ to $h/2$.
Planar mapping coordinates (s, t)

$$s = x$$
$$t = y$$

Each models have above mentioned three mapping stored in general buffers. Which mapping to apply can be controlled. By selecting the model and clicking 'M' key changes mapping to all the images.

Lights:
Turn on/off was implemented by using setmethod. When light is turned off, the color of the light is set to black(0, 0, 0). When light is turned on, the color of the light is set to white(1, 1, 1). Selecting model and clicking 'L' turns on and off the light of each model.

2.2 Phase 2

1. Added Scenegraph class which takes all the models stored vector structure. The tree in scenegraph stores index of models as key values of each nodes.

2. The levelOrderTraversal traverses through the whole tree and call animation function according to the level of the node.

3. For level 0, no animation is done. The model is kept stationary.

4. For level 1, the model circles around level 0 model. As it moves around, it wobbles left and right using simple harmonic motion equation.

5. For level 2, the model takes (x, z) values of parent model and computes next step on the line joining current model (x, z) and parent model (x, z).

6. For level 3, the model jumps up and down on the level 2 model. The positions are computed using simple harmonic equation again.

7. The speed with which each object moves can be controlled using key 'S' and 'D'. 'S' speeds up the models and 'D' slows down the models.

8. While traversing the tree, each model angle is updated if model is selected and rotate is on. Rotation can be switched on and off using 'R' key.

3 References

`http://cse.csusb.edu/tongyu/courses/cs520/notes/texture.php`
`https://learnopengl.com/Getting-started/Camera`
`https://learnopengl.com/Lighting/Basic-Lighting`
`https://www.geeksforgeeks.org/generic-tree-level-order-traversal/`
`https://math.stackexchange.com/questions/175896/finding-a-point-along-a-line-a-ce`