Assignment 3+4:

Vishal Gattani IMT2015508

10-03-2019

Problem Statement

- 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/coordiantes.
- Phase 2: Define a scene graph to manage these models. Animate the objects on the lines described in the next section below. Manage lights and camera settings to create the different effects.

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.
- 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.
- Each of the objects should be renderable 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:
  1. A cylindrical map
  2. A spherical map
  3. A planar map
  4. An arbitrary pre-image mapping (such as that computed by OpenGL as a default mapping)

Phase 2

Add animation to the scene and introduce controls as described below.

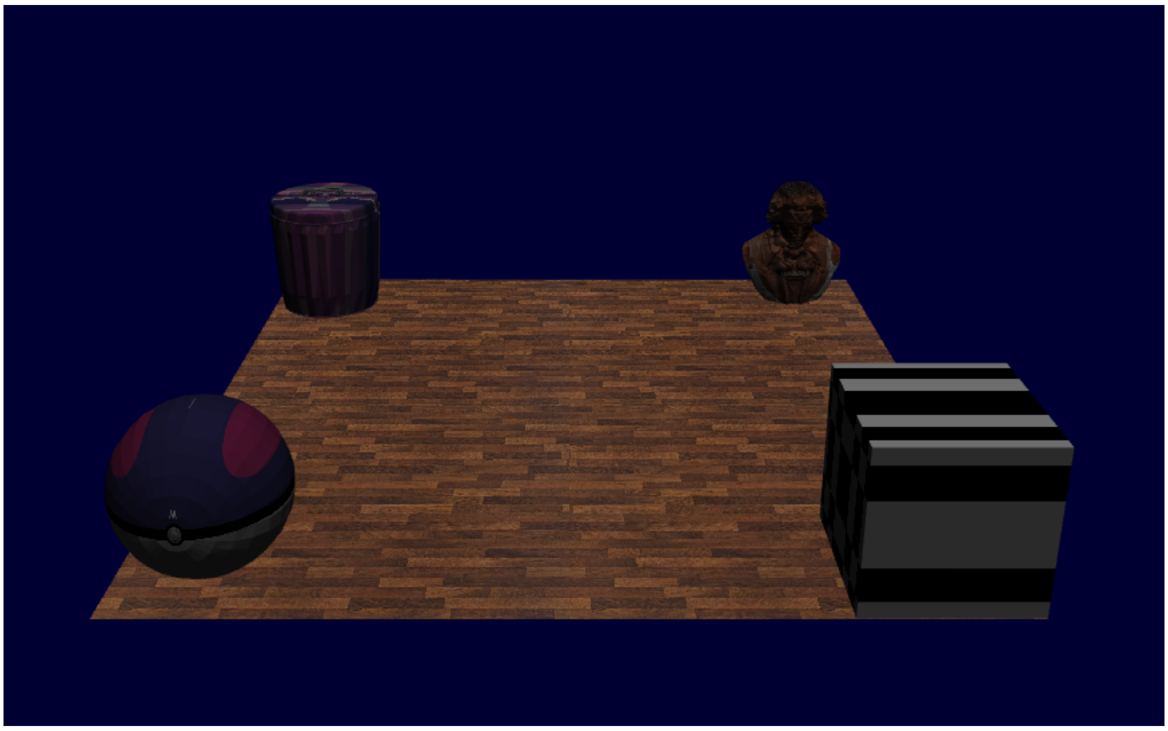
- Design and implement a scene graph and add the models to this, The structure of the scene graph should help the implementation of program features described below. Rendering of the scene for each frame should be implemented by rendering this scene graph.
- Add animation to the objects in the scene as follows:Assume the 4 objects are named A, B, C and D. The objects move as follows:
  1. Object A is fixed with respect to the floor
  2. Object B moves in a circular (or similar path) around object A, and wobbles left and right as it traverses this path
  3. 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.
  4. Object D sits on top ot C, and jumps up and down while staying on top of C.
- The four lights used in Phase 1 now track their target objects, as they move around.

For all the above, an appropriate scene graph is set up and manage all animations and rendering through traversal of the scene graph.

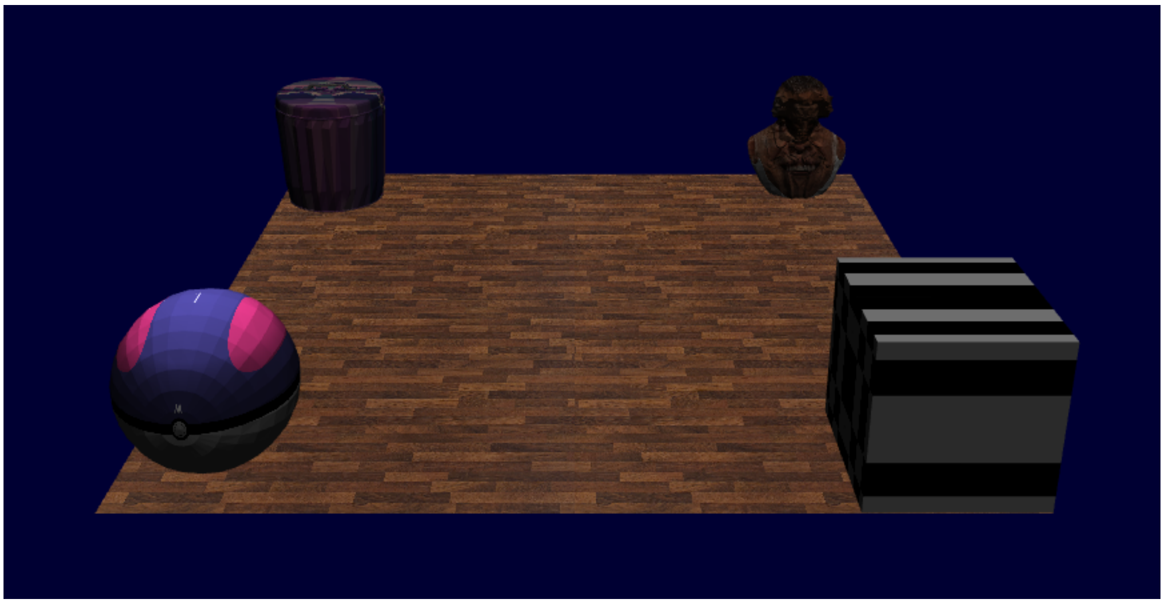
Phase 1 Implementation

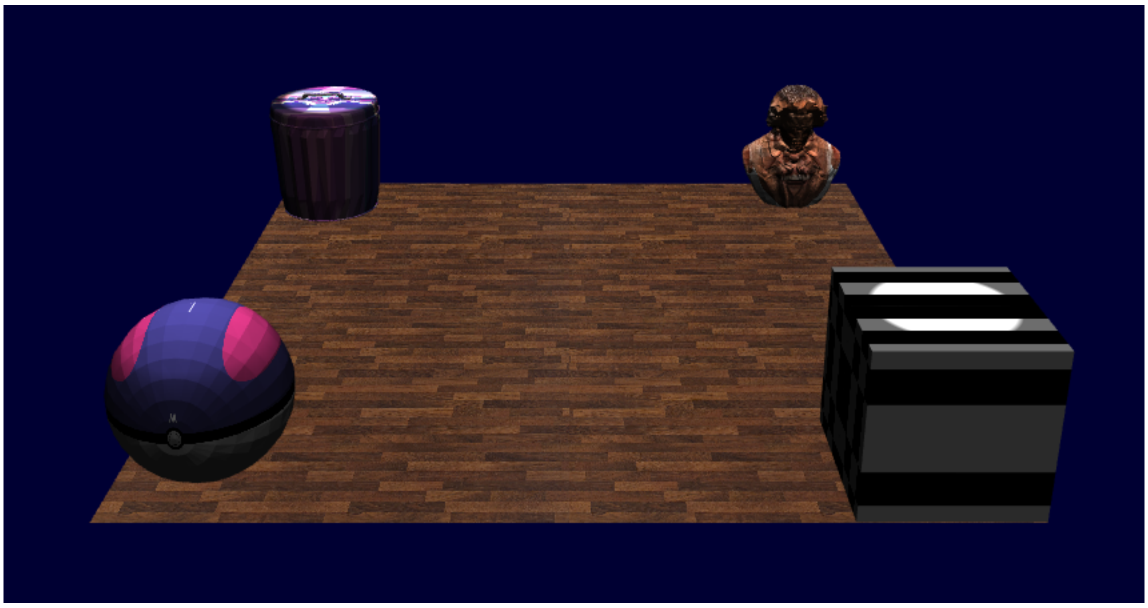
No scene graph is used to implement the scene. The controls are as follows in the implementation of Phase-1.

- The mouse acts like the camera. Moving the mouse moves the view.



- The lights on each object can be controlled by the keyboard buttons from **0** to **5** where:
  - 0: Switch off all lights
  - 1: Switch on light on 1^{st} object
  - 2: Switch on light on 2^{nd} object
  - 3: Switch on light on 3^{rd} object
  - 4: Switch on light on 4^{th} object
  - 5: Switch on all lights





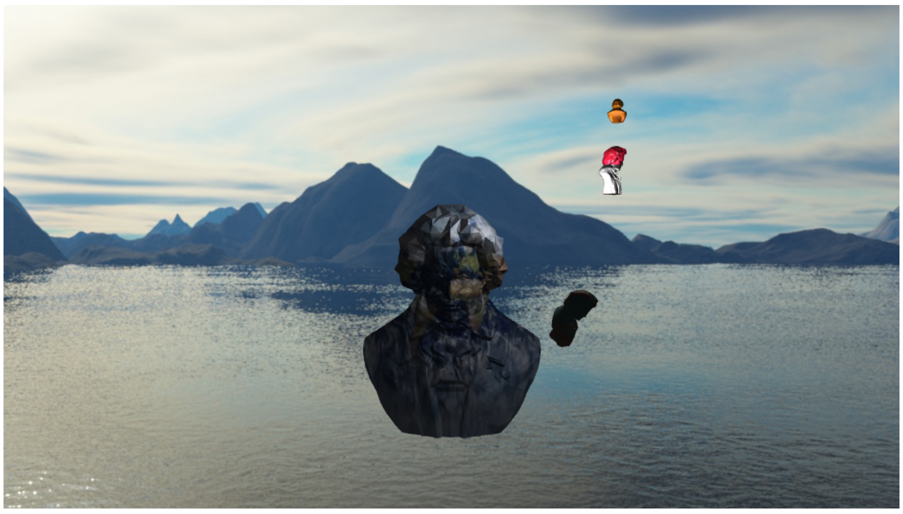
- Once this light is turned off only the ambient color of the object is rendered. Otherwise the diffuse and specular color of the object is rendered.
- Rotation of the scene can be done using the following keys which rotate in either of the *x,y,z* axes either in *clockwise* or *anticlockwise* direction.
  - I,J,O,K,P,L are the keys which help in rotation of the scene anticlockwise and clockwise in x,y,z axes respectively.

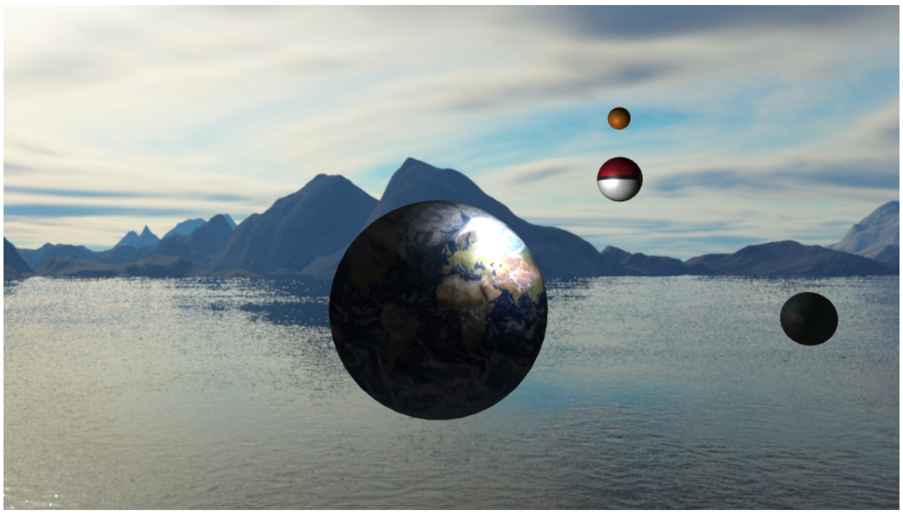
Phase 2 Implementation

1. Scene Graph
   - A model graph node is created such that the position of the current model node would depend on the position of the parent model node.
   - Creating a dependency for each model with its parent through its local matrix and world matrix for each node.
   - The draw function of the root of the scene graph results in recursive function calls to the draw function of its children sending its current position as the parameters.
2. Animation
   - Object ID is added to keep track of its motion in the scene or to procedurally animate it having less control over the object.
   - The draw function of a model uses **glfwGetTime()** as the variable to update the time elapsed of the scene graph node and update its position of itself according to the values.

In the above screenshot, all the lights are near the face of the model and they follow the models motion throughout the scene's display.

In below two screenshots, a single light source is initialized in the world space which effects all the objects near to it. In this case the lights don't follow the objects.





References

1. https://learnopengl.com
2. https://webglfundamentals.org/webgl/lessons/webgl-scene-graph.html
3. http://ogldev.atspace.co.uk/
4. http://www.opengl-tutorial.org/
5. https://github.com/wthibault/ssg