Project CD: Treasure in Aladdin's Cave of Wonders

By: Dhruv Koul
NetID: dko269

Goals:

The goal of the project was to implement programmable shaders in GLSL that work in an OpenGL depiction of an animated and interactive scene in 3D. My project is an add-on to Professor Tumblin's materials and lighting starter code (with the teapots) with my own 3D objects that look like some treasure that could be found in the Cave of Wonders that exists in Disney's *Aladdin* (the lamp, jewels and nuggets, etc). The program should be able to load, compile, link and use our own Vertex Shader (.vs file) and our own Fragment Shader (.fs file) to distort the geometry of each shape and render Phong shading for Phong materials and lighting.

Improvements After Demo Day:

1. I have added a time-varying fragment shader, it flashes (noise) a distortion after some time on a timely basis, and returns to the original non-noise shading afterwards. This was implemented for a portion of the extra credit.
2. You can turn the second light on at the start of the program using the 'K' key, and then afterwards you can switch the lights on/off using the 'L' key. One must press 'L' and press 'L' again and rotate the view to the other side of the object (move 180 degrees) to see the on/off effect having taken place. It sort of works.
3. You can right-click and move the green teapot light and one can see the lights follow it as it moves across the graphics display. The original working code with the fixed-pipeline for the movable light was overwritten by my shader and thus removed the proper functionality of providing specular highlights on other objects, but one can see the highlights follow the lamp as it moves across the screen. It again sort of works.

User's Guide:

When one starts the program, they see 4 different “lamps” (Aladdin themed) that are surrounding two vertex-array prisms that are built within another to give a different effect. On the left of the left-most lamp and on the right of the right-most lamp are two more of these vertex-array prisms (let's call them “jewels”). These are distorting back and forth on their own without any user interaction needed and are lit by the two lamps that have been placed at different coordinates away.

The user is prompted to press the 'H' key in order to get the manual of instructions to appear on the screen. They can interact with the entire screen by clicking and dragging that shows the smooth interaction with the screen, and they can use the arrow keys to maneuver to the left and right and the +/- keys in order to zoom in and out of the picture as necessary.

One can see that as they rotate the picture back and forth and left to right, it is constantly illuminated by the lamps. The user can then press the 'L' key to turn off one of the lamps as they please. The user can also see that the the objects are not fragmented and bumpy/rough as was shown in the starter code, and the surfaces are now appearing smooth and normal. Also, the beam of light highlighting the object shows as a smooth circle, not as a fragmented hexagon.

Lastly, the user can press the 'R' key in order to revert back to the original camera position no matter how they have interacted with the display window, they will be put back to the original camera location. The user can press the 'M' key in order to change the Phong material for one of the lamps (the right-most lamp) to see different materials and how they appear. In order to quit the program, the user can either press ENTER, SPACEBAR or 'Q'.

Code Guide:

I used a lot of the starter code that Professor Tumblin provided, starting originally with the starter code that was for materials and lighting (with the three original lamps and his mini-pyramid showing up on screen but in bumpy surface format). Professor Tumblin's instruction sheet said that the starter code took care of the 4 separate Phong lit materials, OpenGL lights turning on and off, a movable 3D light and the 3D view control. I added on a little bit in order to get the lights actually turning on and off because I couldn't figure out how to do that using the starter code, and I calculated the ambient, diffuse and specular characteristics of the Phong model in order to get the smooth lighting/shading to work. I utilized the help of a website (listed in the sources below), the Yungmann starter code on Blackboard and I worked with a couple of classmates (referenced in the Sources section) on how to get the lighting/shading and distortions to work.

In order to load the shaders in the first place, I named two files, “PassThroughFragmentShader.fs” and “PassThroughVertexShader.vs” with the appropriate shading code, and then loaded them into the program in the my_glutSetup() function. I also utilized some of the functions that were found in the Yungmann starter code posted to Blackboard in order to actually load, compile and link the shader files to the program. I am also passing a time variable and two position variables to the shader in order to help with the automatic geometric distortions, using help of the Yungmann starter code that was posted in order to show geometric distortion of the robotic arm that was one of the starter codes for previous projects. I use the glUniform1f function to pass the timer variable and tracking of the program run-time to help with the distortions as well. The Phong material is initialized in my_glutSetup() as are the initial lamp values setup.

In the Fragment shader file, the ambient, specular and diffuse values are calculated and combined to get the color for the shading on the objects on screen in order to get smooth surfaces. In the Vertex shader file, that is where the calculations are done in order to distort the vertices of the object(s) using the time variable that is passed.

The code is commented where I am doing adding my code to the original starter code and I have also identified where I have utilized help in order to achieve what I want to achieve, also noting where the help came from. I have also listed all of the sources used and named the people I worked with in order to achieve the project results.

(RESULTS/PICTURES ON NEXT PAGE, REFERENCES/SOURCES FOLLOW THAT)

Results:

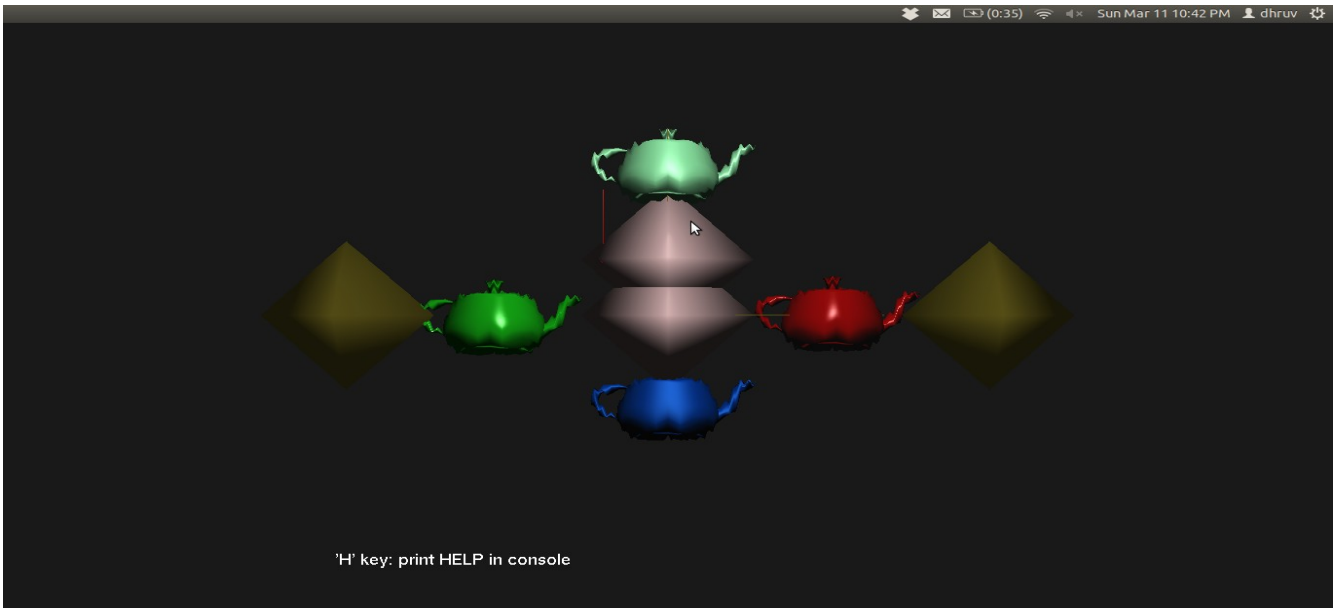

Figure 1 – The program first starts, one can see one light shining on the object and a vertex distortion. The distortion moves fast depending on how long the program has been running.

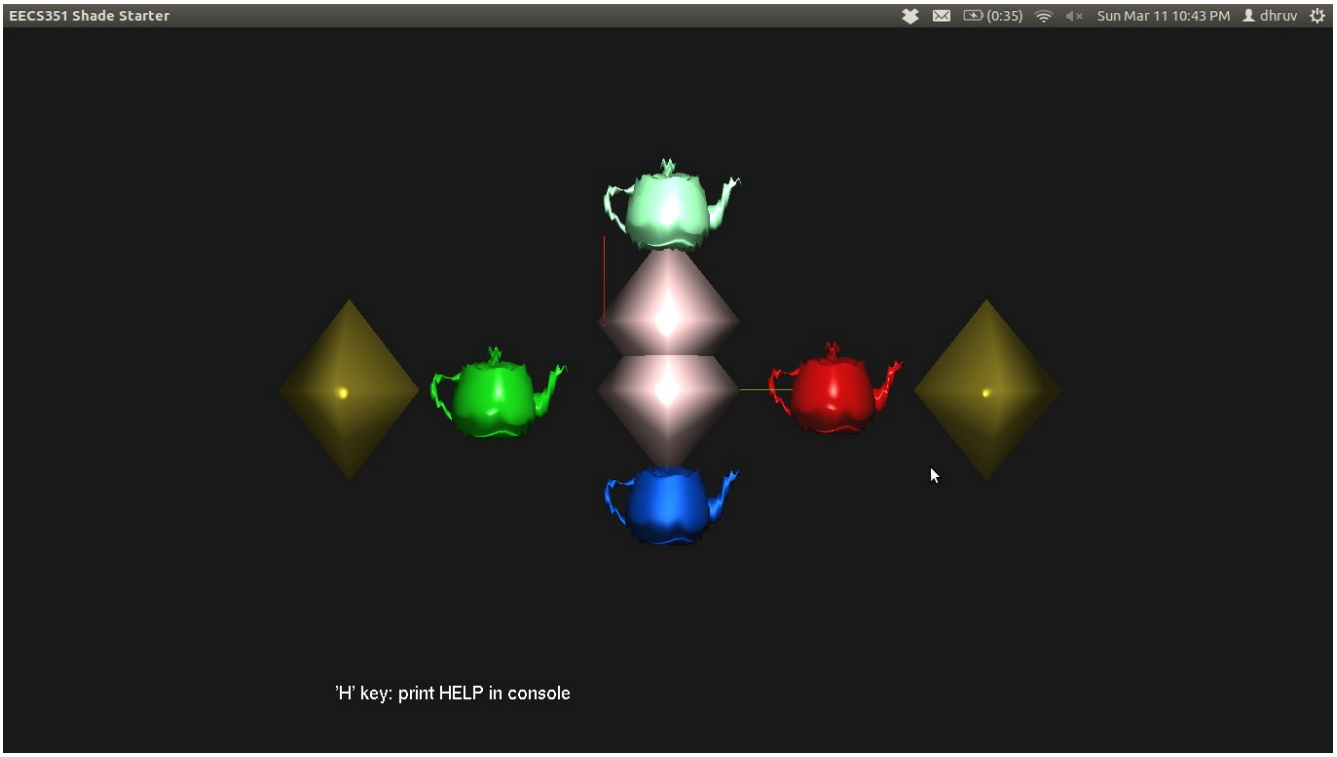

Figure 2 – The second light has been turned on using the 'K' key, and one can see the middle prisms that are surrounded by the lamps as having a brighter light shone upon it.

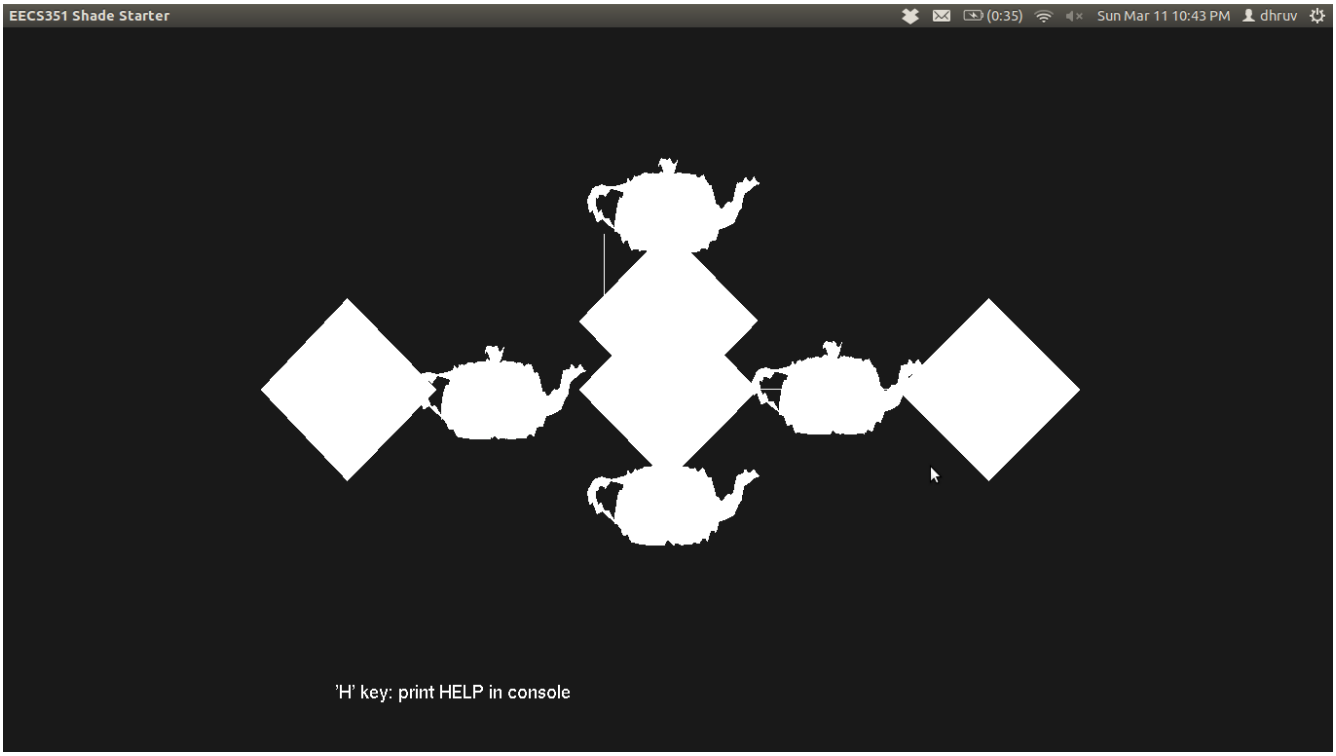

Figure 3 – There is a flash distortion of the fragment-shader. This is also time dependent and flashes after a certain amount of time.

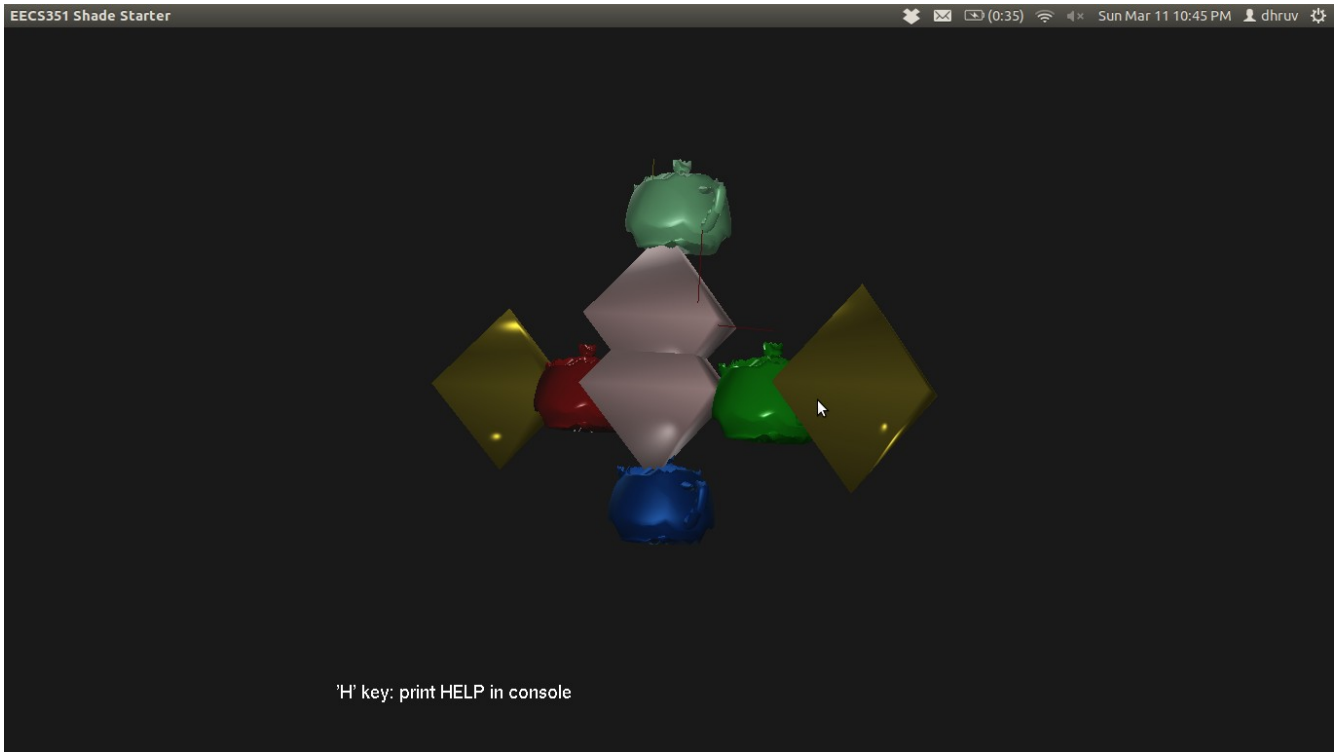

Figure 4 – 3D view control is working as can be seen by the change in view, and the light has been turned off, as can be seen by the dimness of the object.

References/Sources:

1. Help with Phong shading and lighting (link shared by Khalid Aziz): http://www.opengl.org/sdk/docs/tutorials/ClockworkCoders/lighting.php

2. GLSL Tutorial from Lighthouse3D (helped with writing the Phong shaders for smooth surface and lighting): http://www.lighthouse3d.com/tutorials/glsl-tutorial

3. Yungmann starter code on Blackboard for distortions (got help in passing variables to the shader and even to load, compile and link the shader files to the program)

4. Got help in understanding how to do shading, lighting and get distortions on screen online from this link: http://zach.in.tu-clausthal.de/teaching/cg_literatur/glsl_tutorial/

5. Khalid Aziz helped me to understand how to get the lamps to turn on and off (though it's not entirely working for me in my own shader)

6. Worked with Cassie Rommel and Khalid Aziz for the automatic geometric distortions (we shared ideas and worked together to get the original normal shading/distortions to work)