Project B: A Broken Rocket With Satellites

By: Dhruv Koul
NetID: dko269

Goals:

The project is a depiction of a broken down rocket that has its own satellites orbiting around it. The goal of the project was to shift into making 3D objects without using glVertex deprecated commands – instead we were to use vertex arrays in order to create these 3D objects, defining vertices, colors, etc. We were to get used to vertex arrays, camera projections, including the Perspective and Ortho views, as well as multiple viewports.

Improvements After Demo Day:

I edited my code to make the smoothly adjustable 3D view control happen and put the instructions to be able to maneuver through the viewport in the help menu.

I also added the bitmaps to appear on screen and one can move them using the RIGHT and LEFT arrow keys. They're the same images and same code used from Professor Tumblin's starter code because I was unable to get my own images on the screen, but there are 3 images and they toggle onto the screen by pressing 'B'. Everything is explained in the help menu.

User's Guide:

When one starts the program, they see 4 viewports, three of which are fixed (top left, bottom left and right), and the last one (top right) is one that isn't fixed, but can be rotated using the mouse to explore all angles. Everything is stationary except for 4 little "satellites" that are moving around the screen, around the "rocket".

The user is prompted to press the 'H' key in order to get the manual of instructions to appear on screen. They can interact with the top right viewport by clicking, holding and dragging around to see the view as they like. There is a grid in each of the viewports so the user can see how the objects have rotated.

One nice thing is that if the user maximizes the graphics screen or from a maximized size decides to bring it down to the normal size, the images don't distort in odd manners in response to the changing of the display window's size. They stay the same on the screen and don't get weirdly stretched out or squeezed in due to a change in graphics screen size.

The user can press the 'Z' key to change between the Perspective and the Ortho views while not changing the orientation of the current view that the camera is on. They can use the 'R' key to get the top right viewport back into the default view as occurs when the user first runs the program. They can also use the 'Left' and 'Right' arrow keys to have the other "satellites" orbit the stationary yet spinning "rocket".

Bitmaps toggle on and off the screen by pressing the 'B' button and moving them using the left and right arrow keys. They're the same images from Professor Tumblin's code and show up using the exact same starter code as Professor Tumblin provided. The user can press either 'Enter', 'Space bar' or 'Q' to quit the program.

Code Guide:

I used quite a bit of the starter code that Professor Tumblin provided in his starter code projects posted on Blackboard. The myDisplay function within the program is set up in the following manner: set up viewport, set up camera view, draw the scenery. I am doing all of the drawing in a function called drawScene, that itself contains different function calls to set up the scenery. I got help from classmates on how to create shapes using vertex arrays and ideas on how to switch between Perspective and Ortho views, and proceeded to make the scenery move and rotate just like was occurring in Project A. All of the sources used will be presented in a “Sources” section at the end of this report.

Per a friend's suggestion, I included classes in the .h file that are used to draw the Pyramid and Prism shapes; each class contains an array for vertices, indices and colors that are set and drawn within the .cpp file. I also changed the doCamChoice function to take a parameter, camChoice, which made it easier to switch between Perspective and Ortho views, as per suggestion of another classmate. I am also declaring all of the member functions of the different shape classes within the .cpp file, where all of the drawing is happening, and I learned how to draw using help from a website, which will be included in the “Sources” section at the end.

I am using PushMatrix and PopMatrix to save and return my matrix stacks as I am drawing objects, and am using similar methodology that I used in Project A to make shapes move and rotate, including using glRotated and glTranslated.

I got the bitmaps to appear on screen by using all of Professor Tumblin's starter code and images and got help on how to make them appear on screen using help from a friend. I commented out some code in the CByte.cpp file in order to help them appear on screen.

I apologize for the untidiness of the code again. I didn't realize this project would be as tough as it was and lost track of commenting as I went through. I have highlighted in the code where I have utilized classmates or websites help, so that should be documented. I am also including the sources in this report at the end.

Results:

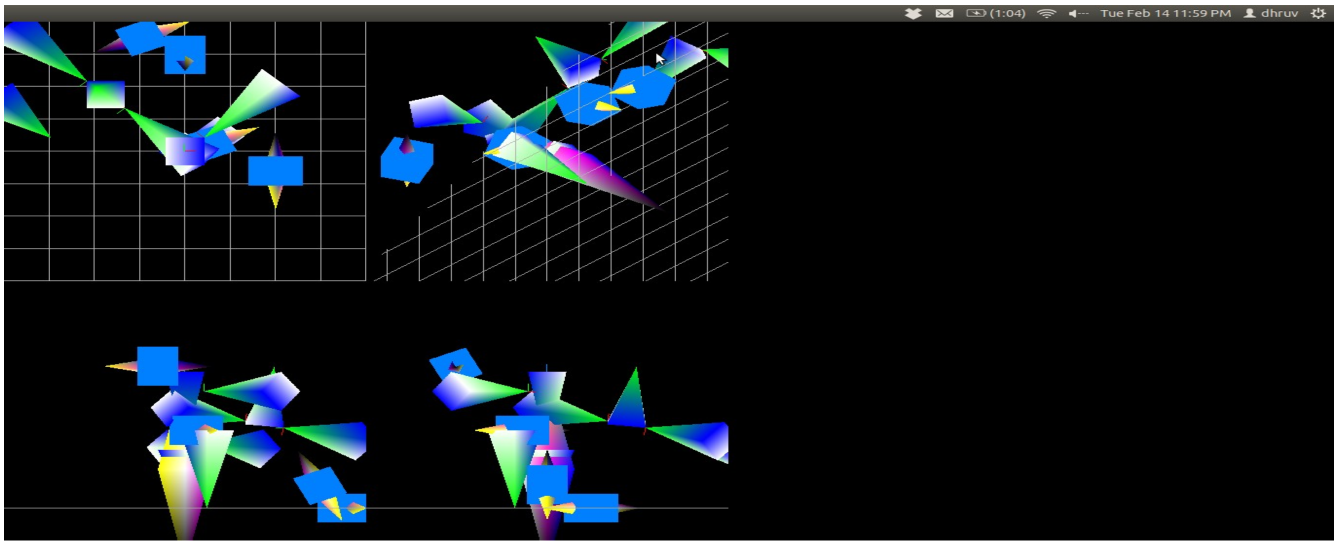

Figure 1 – Showing one perspective when it's first loaded

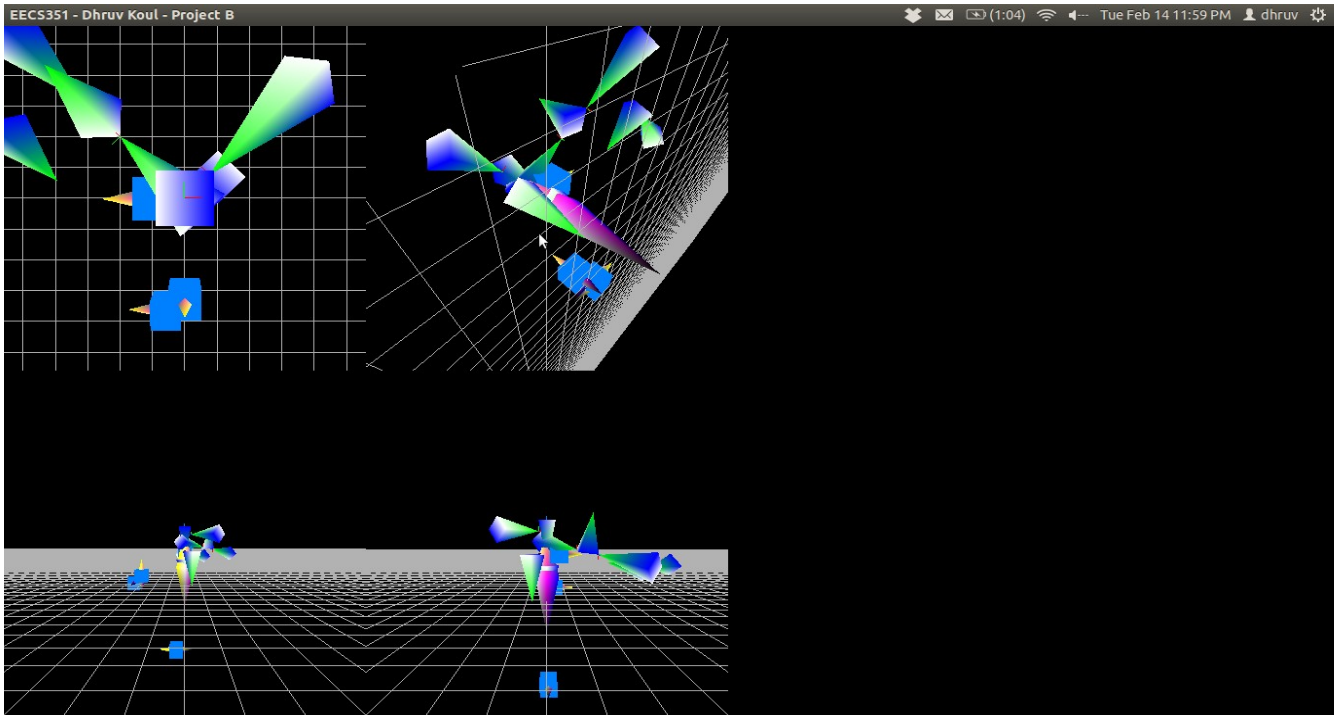

Figure 2 – Other perspective after pressing the “Z” key

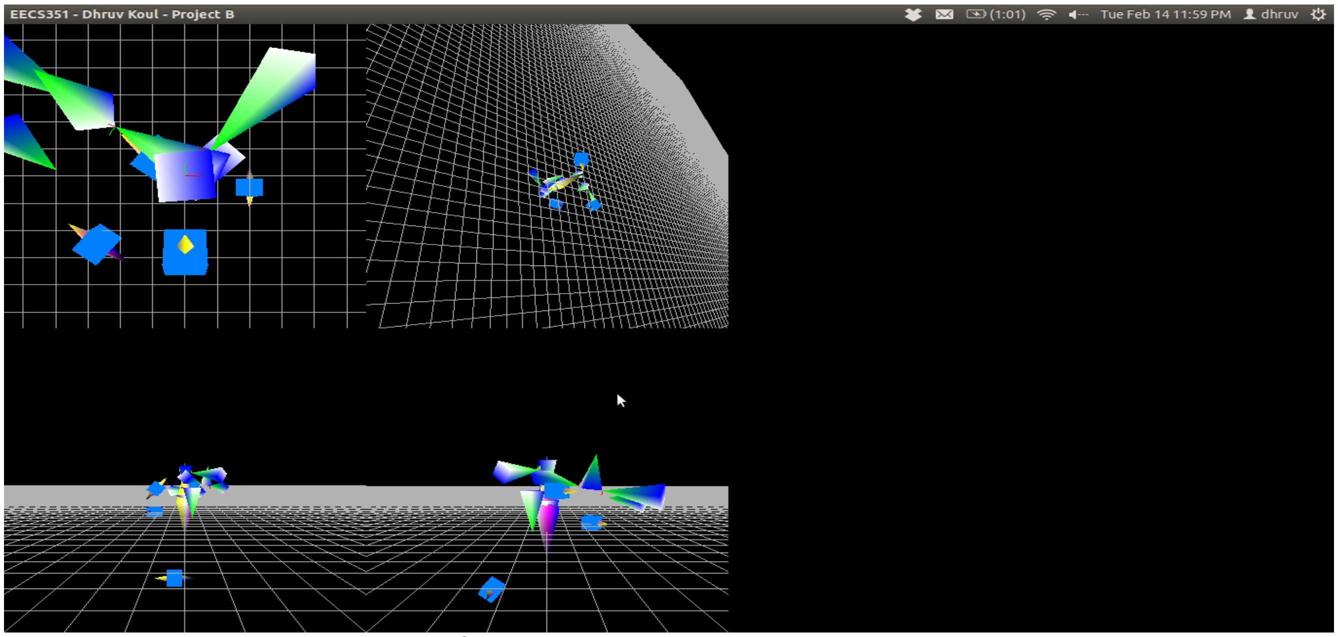

Figure 3 – Moving the 4th viewport camera to show smooth moving of the camera

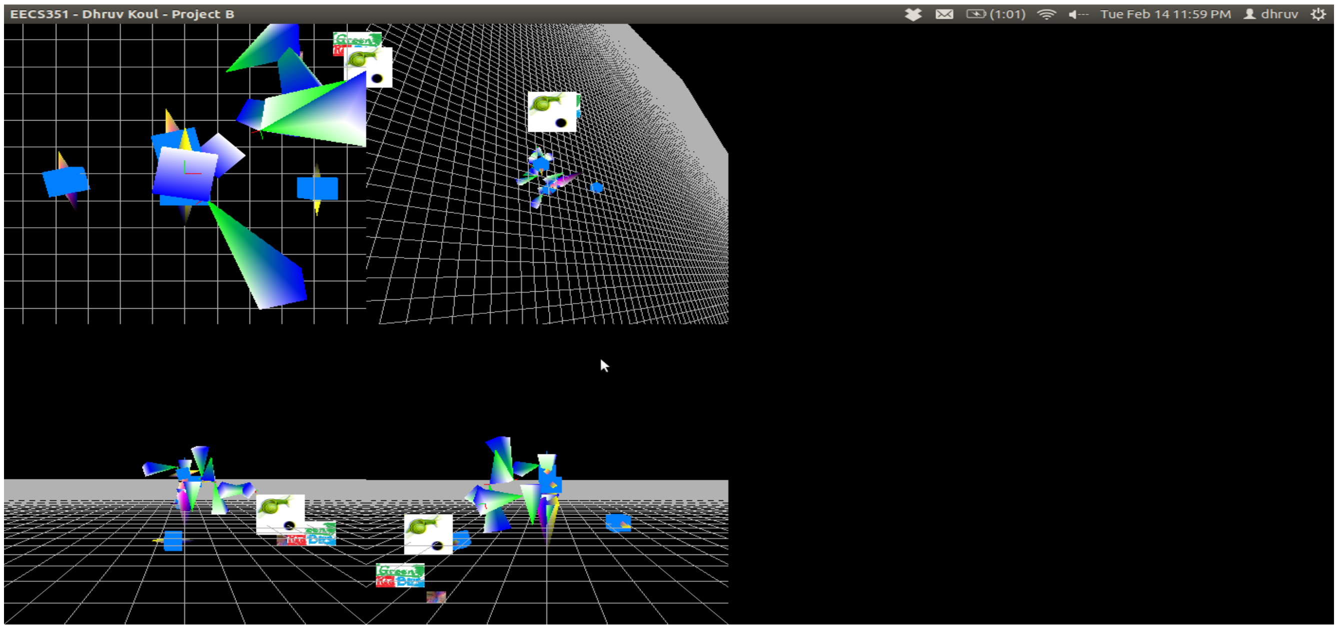

Figure 4 – Bitmaps appear on screen (can be adjusted using the left and right arrow keys)

Sources:

1. I adapted a lot of code from Professor Tumblin. I modified code and built off what had already been provided so it was helpful for me to learn how some of the methods worked. I used almost all of it to get the bitmaps to work and used the images as well because I couldn't get my own images to load. However, it works with Professor Tumblin's code. I got help from Khalid Aziz to structure the code in order to get it to work.

2. I got a lot of help on how to attack the vertex arrays from Khalid as well and he helped me understand how to build vertex arrays for shapes such as the prisms and cubes that were drawn on the screen. He was very helpful to me in understanding vertex arrays. He also helped me get the smooth camera maneuvering to work and how to switch from perspectives. The views I demoed on Demo Day were subpar.

3. I used this website: **http://www.songho.ca/opengl/gl_vertexarray.html** to help me draw the shapes once I had the vertex arrays formulated. I also got help from Khalid in drawing the shapes as well.