Project A: A Wacky Satellite Station

Author: Dhruv Koul
NetID: dko269

Goals:

This project is a depiction of a very wacky satellite station. The goal of the project was to get introduced to OpenGL, GLUT and Graphics programming using C++ in Code::CodeBlocks, and to utilize some of the things we learned in class, such as using the matrix stack, rotations, translation, transformations and coloring as we got our feet wet in graphics programming.

User's Guide:

When one starts the program, they see the animations going on on screen. I have kept the rotating arm that was supplied by the starter code, and have included a couple of my own hinged objects, one that acts as the satellite base (the trapezoid, triangle and cone stack) and another that acts as moving UFOs across the screen (the cylinders). In addition, after a certain period of time the background color toggles between Cyan and Black, and depending on which color background shows up, it describes that in text in the upper right hand corner, either saying “Cyan Screen” or “Black Screen”.

One can press the 'H' key to get a menu of options appear on the screen. In order to pause the animation, they can press 'S'. Upon hitting 'S' again, the animation resumes from the point where it had been stopped. The user can also hit the 'C' key, which makes the animation “go crazy”. It reveals a trail of the movements of all of the components on the screen. The moving cylinders and satellite cone, as well as the moving lines across the screen reveal a trail of their movements. Upon hitting 'C', it clears the screen of the trails and continues the animation like normal.

For mouse clicks, clicking the mouse repeatedly shows a flashing, rotating cone in the middle that rotates a little bit with each click of the mouse. Upon dragging the mouse, the animation stops and text appears on the screen that asks the user to stop dragging the mouse, so that the animation can go back to running (probably to detect a meteor of some sort).

In order to exit the program, the user can press the 'space bar', or the 'Q' key to quit. Again, all of the instructions show up when the user presses the 'H' key for help.

Code Guide:

I have used the base that Professor Tumblin's starter code provides and modified off of that largely. Again, I kept the rotating arm and have added my own objects and animations to the program as well. I wrote a couple of my own functions, and found some code online to help with drawing some of the OpenGL primitives, including the trapezoid, triangle and circle. I also got some ideas from classmates on how to toggle the background color and the “descriptive” text that shows up in the background when the color changes, as well as to make the user instructions appear on screen. All of the sources used will be provided at the end of this report, in a separate “Sources” section.

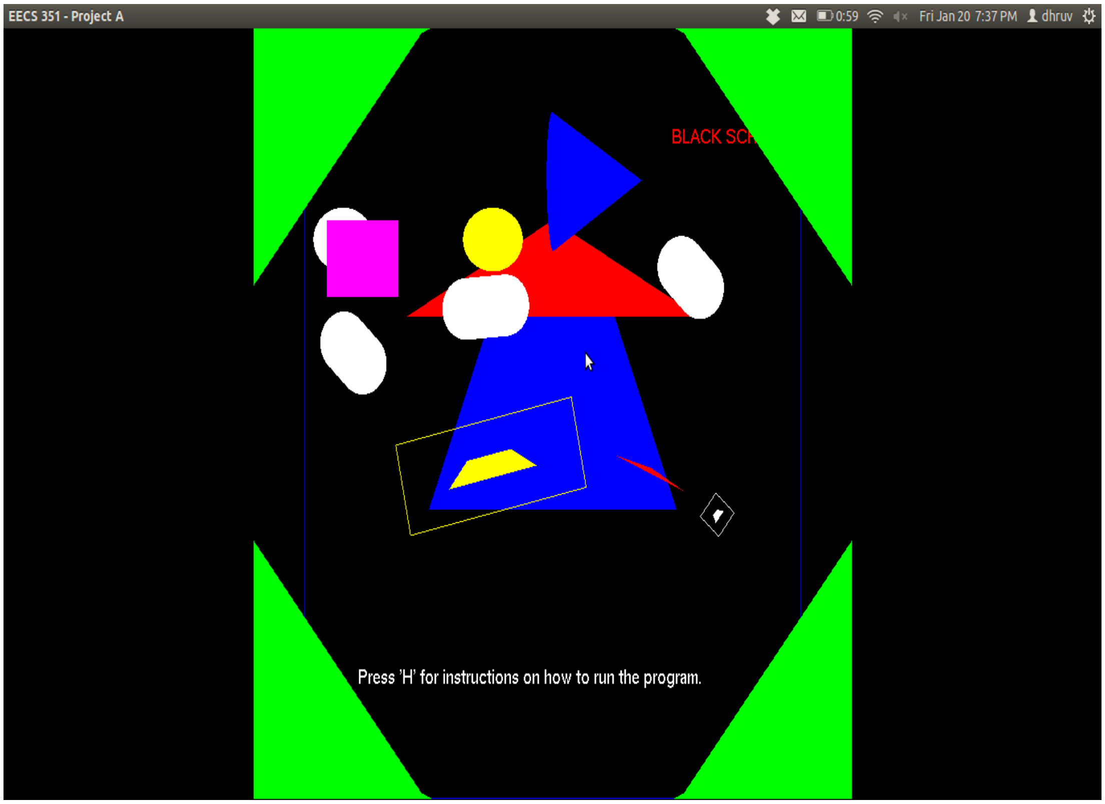
I included some global variables, including a toggle, helpButton as a integer, to display the user instructions when the user hits the 'H' key. I also introduced a count variable that helps me keep track of when to change the background color and subsequently display the descriptive text in the background saying what color background is being displayed.

I wrote some new functions that I added to the glutStart.h header file. I wrote functions to print the instructions when the user asks for them, and functions to draw the circle, triangle and trapezoids used. I also used some of the inbuilt functions, including ones to draw the cylinders, cones, cube and the dodecahedron that surrounds the background that is always changing.

The global variables that I have included for the toggling are accessed by the GLUT callback functions, and then changed to a new value. Every time an object needs to be drawn, the draw function checks those values to see what properties must be drawn to the graphics window. I change the background color every time the "count" variable I've included mod 100 is greater than 50 to a certain color, and then change it back when the mod result is no longer greater than 50.

I apologize that my actual code isn't commented too well, I normally do a good job of commenting my code, but this was my first project and I got overwhelmed by the intricacies of what I was trying to do and ignored commenting. In future projects, it will be a lot better so the user can read it better and understand what the code is doing by just reading the source files.

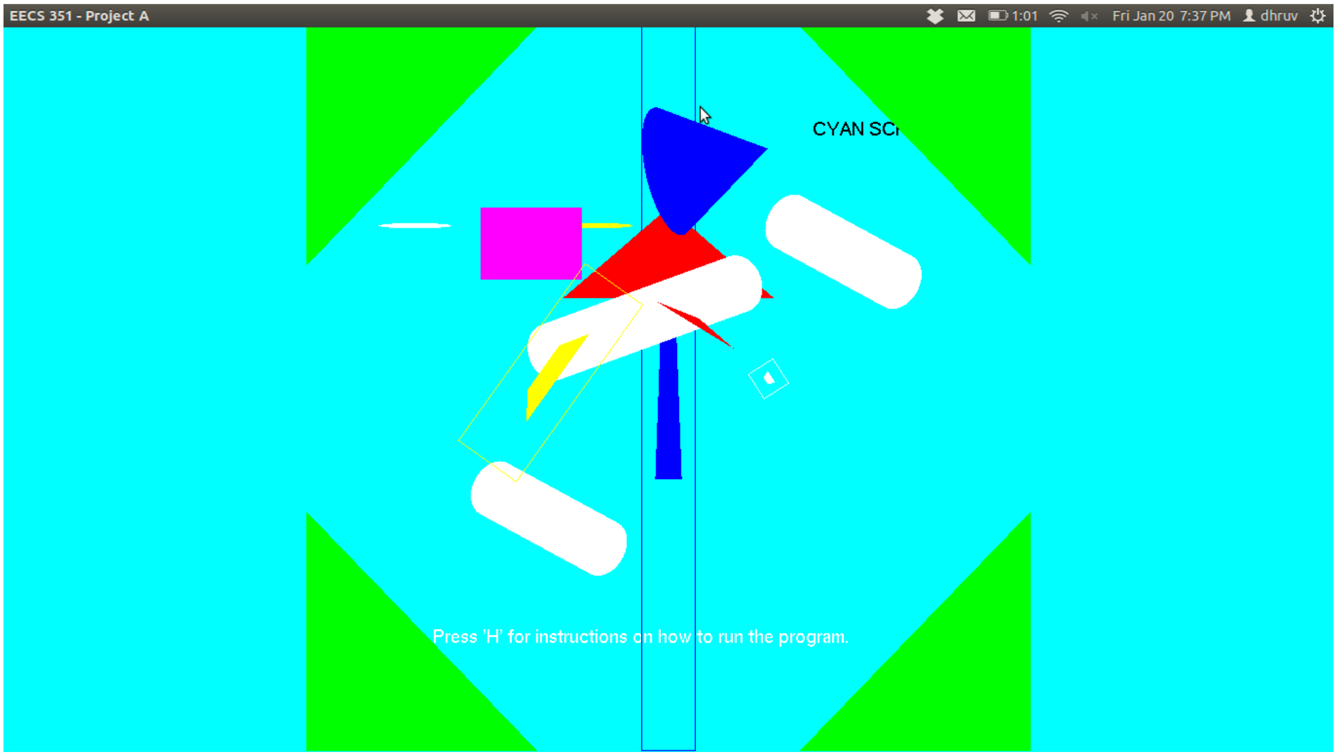
<u>Results:</u>



BLACK SC

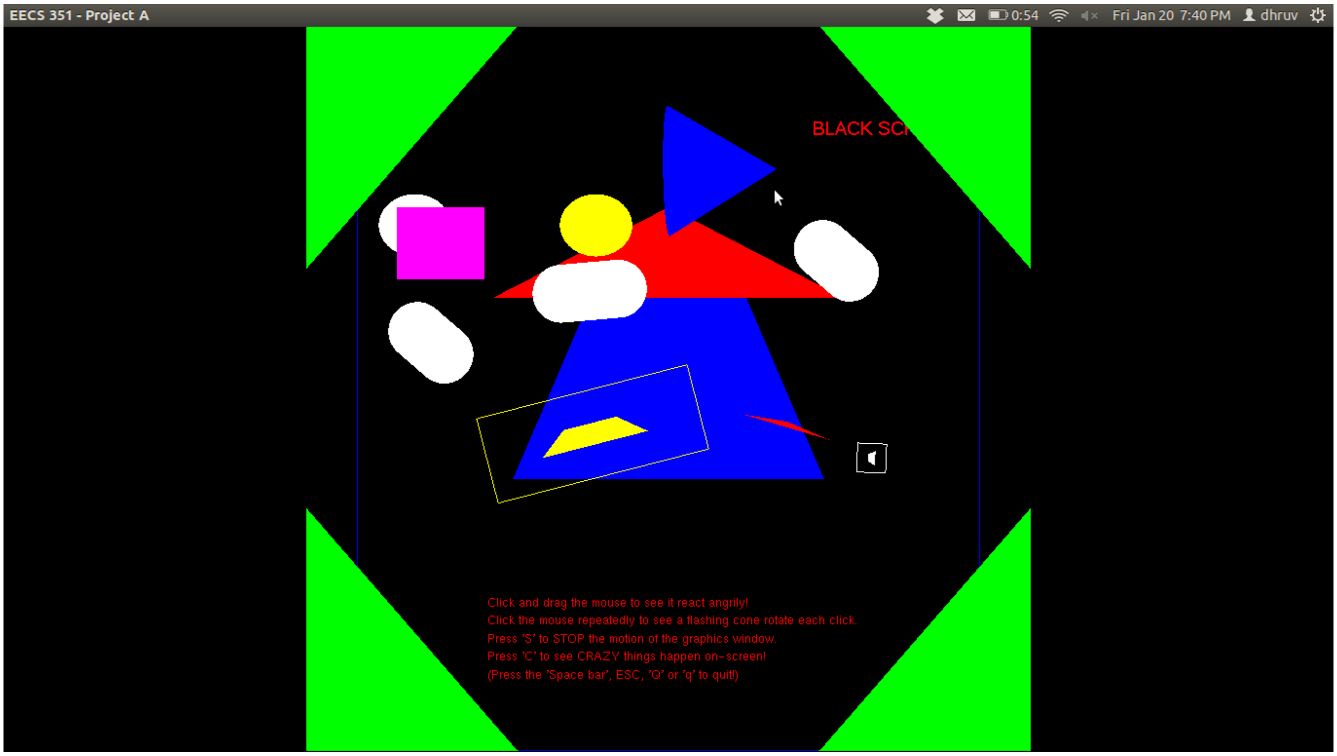
Press 'H' for instructions on how to run the program.

Figure 1 – This figure shows the black background as the animation is running and the user can see "Black Screen" in the upper right corner.



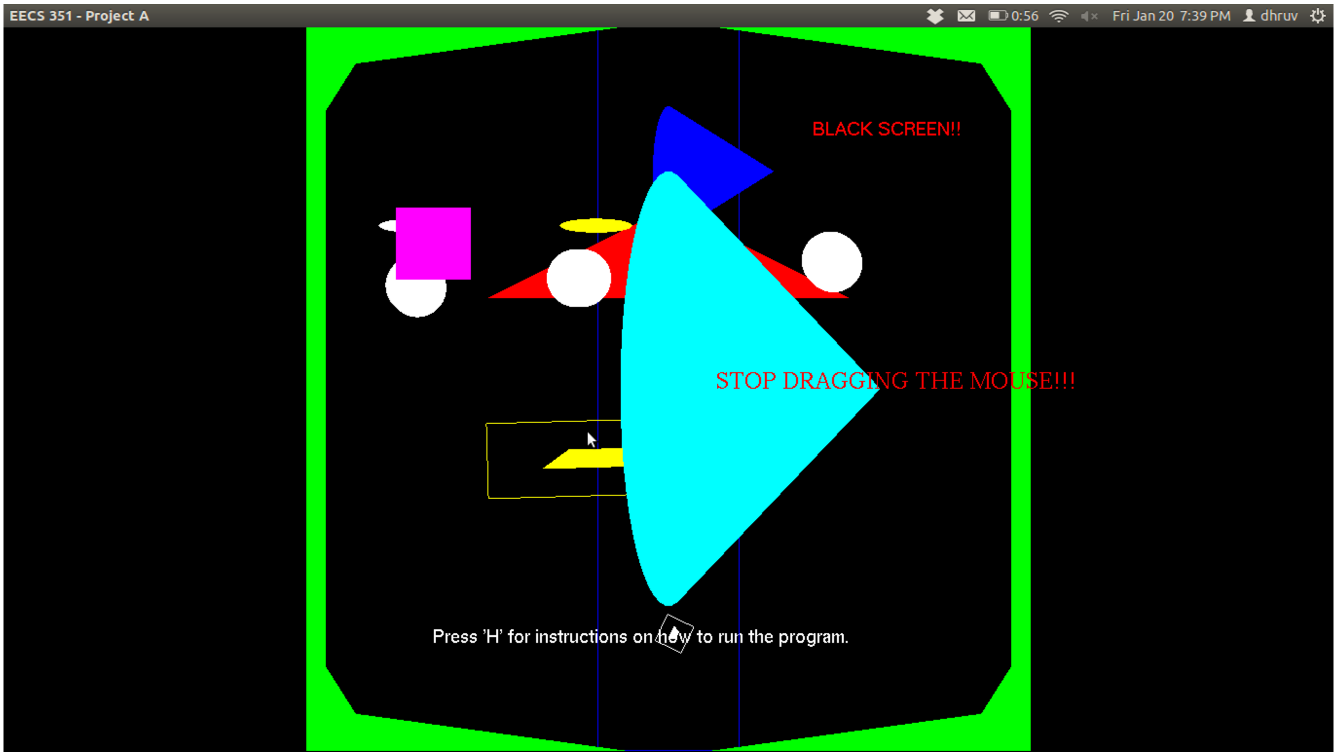
Press 'H' for instructions on how to run the program.

Figure 2 - This figure shows the cyan background as the animation is running and the user can see “Cyan Screen” in the upper right corner.



Click and drag the mouse to see it react angrily!
Click the mouse repeatedly to see a flashing cone rotate each click.
Press 'S' to STOP the motion of the graphics window.
Press 'C' to see CRAZY things happen on-screen!
(Press the 'Space bar', ESC, 'Q' or 'q' to quit!)

Figure 3 – The help menu is showing up on the bottom in red text, and the user can see what the animation does from this menu when they proceed with the instructions. This is accessed upon hitting the 'H' key when the animation is running.

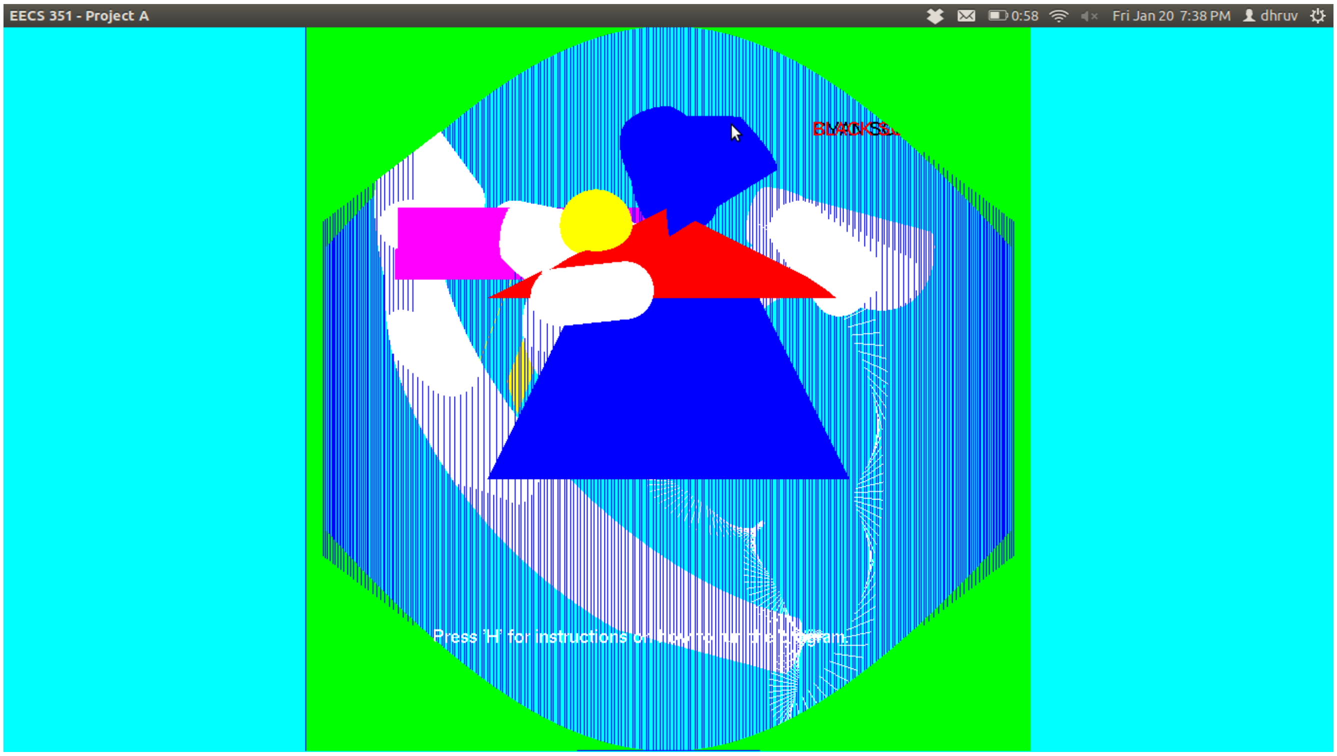


BLACK SCREEN!!

STOP DRAGGING THE MOUSE!!!

Press 'H' for instructions on how to run the program.

Figure 4 – This is when the mouse is being clicked and dragged. The cone in the middle is shown as a result of the mouse click, and the dragging makes the "STOP DRAGGING THE MOUSE!!!" on screen. The animation is paused while the mouse is being dragged.



Press 'H' for instructions on how to run the program.

Figure 5 – This is the trail of motion showing up on screen when the user hits the 'C' key. This is used for keyboard interaction, and the trail of motion disappears when the user presses the 'C' key again.

Sources:

1. I adapted from a lot of the starter source code provided by Professor Tumblin. I modified my code and built off of what was provided, so this was definitely a valuable source for me.

2. http://en.wikibooks.org/wiki/OpenGL_Programming/Basics/2DObjects – This URL provided me the code that I used for help to write the OpenGL shape primitives. I drew the triangle, circle and trapezoid using this code. My classmate, Yash Siddhartha, pointed this out for me.

3. I got suggestions on how to do the toggling for the help menu and the background color from another classmate, Khalid Aziz. I used his suggestion to write my own functions and code that was executed by the GLUT callbacks.