Jasamine Jackson
Hanan Alnizami
June 10, 2008

Rendering a 3-Dimensional Cube Applet Using Light Weight Java Graphing Library (LWJGL) with Java Swing with NetBeans IDE 6.1

This project is a step towards the main project which is the Dance Tool application which is a learning tool that uses 3D dancing characters to teach computer science. The characters dance moves are determine by the code that the user enters into the program. The interface for the Dance Tool is similar to scratch. The Dancing Tool is still a work in progress. This 3D application was created in order to better understand the graphics and user interaction with the tool (as well as learn how to implement graphics based on user driven input in Java).

The Layout

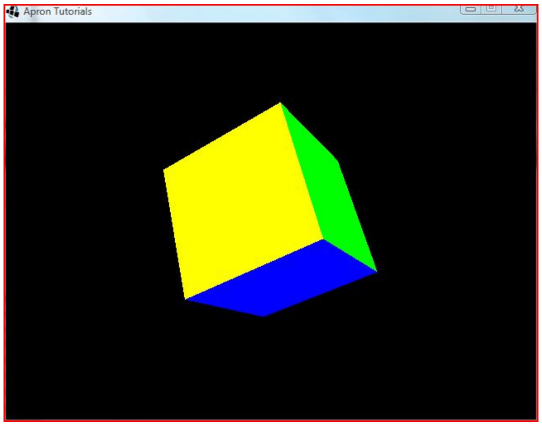
- The Design

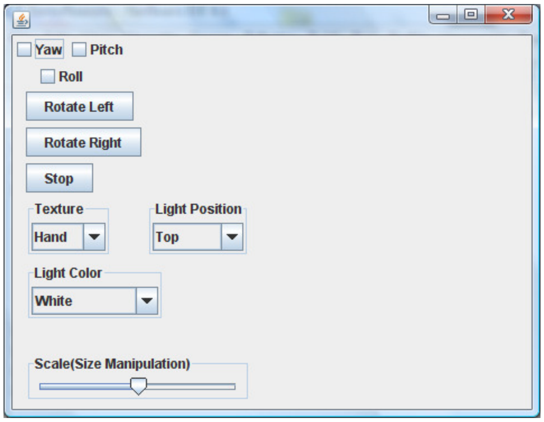
First step is to create a JApplet Swing GUI Form application. The swing controls used consist of three checkboxes, three buttons, three combo boxes and a slider bar. The checkboxes manipulate the yaw, pitch, and roll rotation of the cube; the buttons are used to control the direction of the rotation (left and right); the slider bar is used to manage the size of the cube; and the combo boxes that contain pre-defined drop down menus of lighting positions, lighting color and texture.

The next step is to create and set the canvas' bounds in which the cube is drawn. In the AppletCanvas file, vertices are drawn into the virtual space of the canvas then connected together to form the shape of cube. A camera is set in one position across from the cube to be able to view it at all times. A beam of light is also set to a default position and color and manipulated according to user's selection. A user can also change the texture of the cube by selecting the desired pattern from a drop down menu. The canvas application is grouped to the swing window where the cube is manipulated by the swing controls. Once a user activates an option in the panel, the cubeApplet file reads the functionality of that option then calls the necessary behavior from the AppletCanvas file and applies it to the cube.

- The Code

The applet includes the canvas and the GUI which provides user controls. (Figure 2 and Figure 3 respectively):





□ Yaw □ Pitch
□ Roll

Texture: Hand

Light Position: Top

Light Color: White

Scale(Size Manipulation)

Figure 2: Cube Canvas

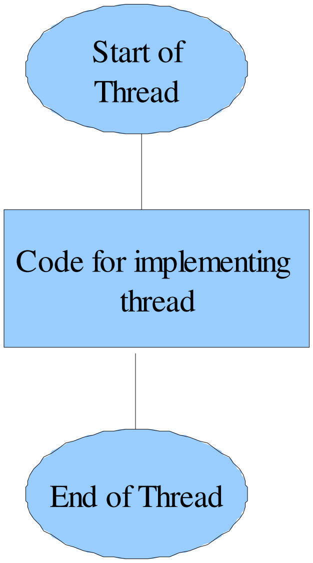
Figure 3: User Control GUI for the 3D cube

As mentioned earlier, two files were created to achieve the task of rendering the cube: AppletCanvas and CubeApplet. Below are diagrams that explain the procedure:

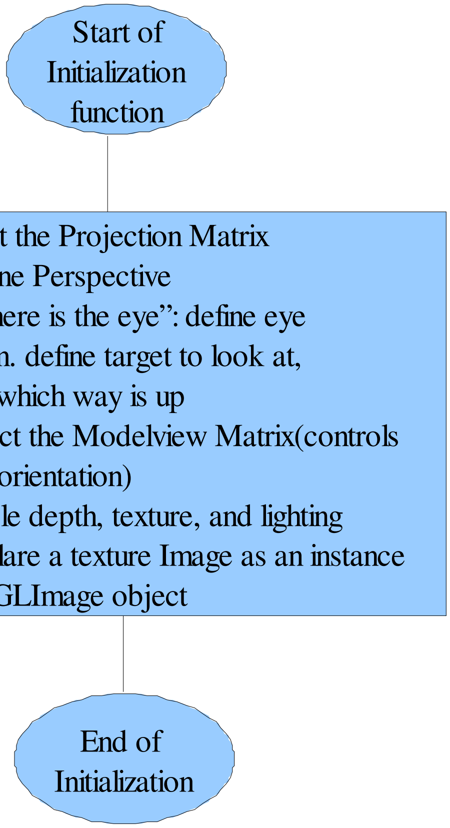
- *AppletCanvas.java Class File Flow Chart*

<u>Calling the thread function:</u>

This thread enables the canvas and the applet to function at the same time.



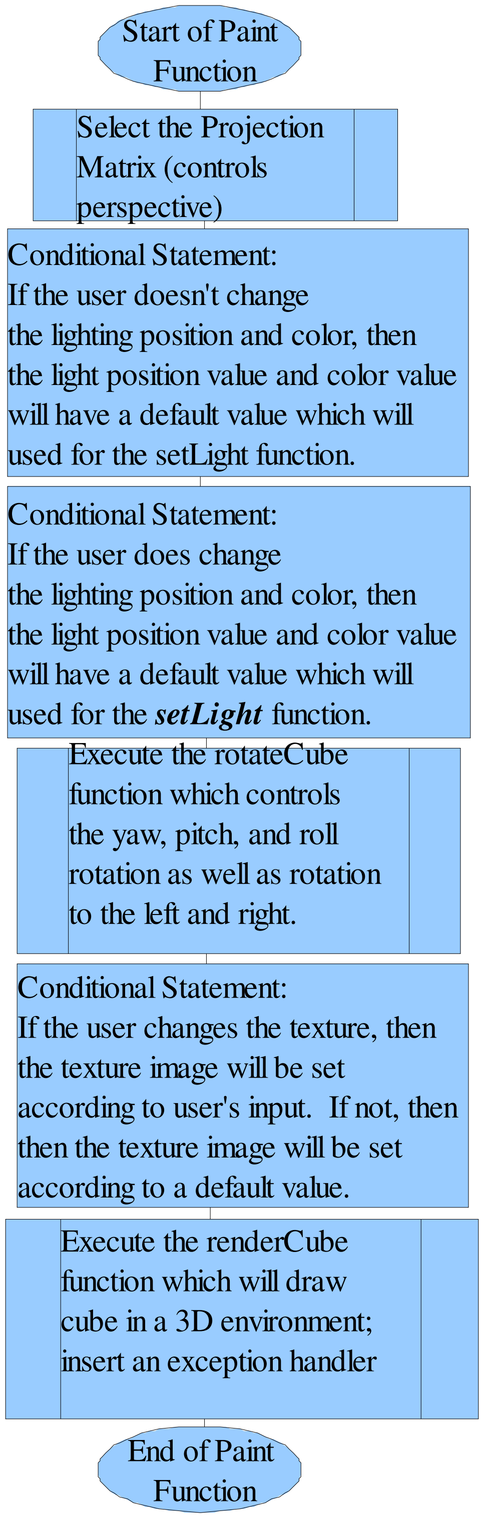
Initializing the Camera and Enabling Depth, Texture, and Light:



1.Reset the Projection Matrix
2. Define Perspective
3. “Where is the eye”: define eye position. define target to look at, define which way is up
4. Select the Modelview Matrix(controls model orientation)
5.Enable depth, texture, and lighting
6. Declare a texture Image as an instance of the GLImage object

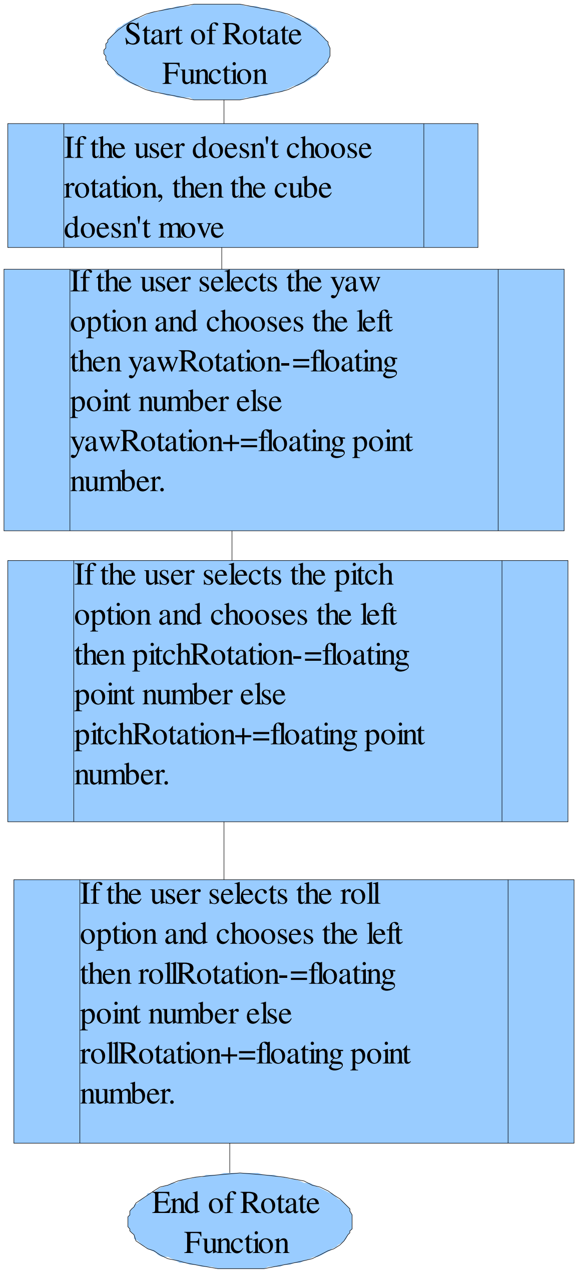
Painting the Canvas:

This function will draw the cube according the user's input. This function draws the rotating textured cube by drawing the cube frame by frame.


Start of Paint Function
Select the Projection Matrix (controls perspective)
Conditional Statement: If the user doesn't change the lighting position and color, then the light position value and color value will have a default value which will used for the setLight function.
Conditional Statement: If the user does change the lighting position and color, then the light position value and color value will have a default value which will used for the ***setLight*** function.
Execute the rotateCube function which controls the yaw, pitch, and roll rotation as well as rotation to the left and right.
Conditional Statement: If the user changes the texture, then the texture image will be set according to user's input. If not, then then the texture image will be set according to a default value.
Execute the renderCube function which will draw cube in a 3D environment; insert an exception handler
End of Paint Function

Rotate Cube Function:

This function controls the rotation of the cube.



Start of Rotate Function

If the user doesn't choose rotation, then the cube doesn't move

If the user selects the yaw option and chooses the left then yawRotation-=floating point number else yawRotation+=floating point number.

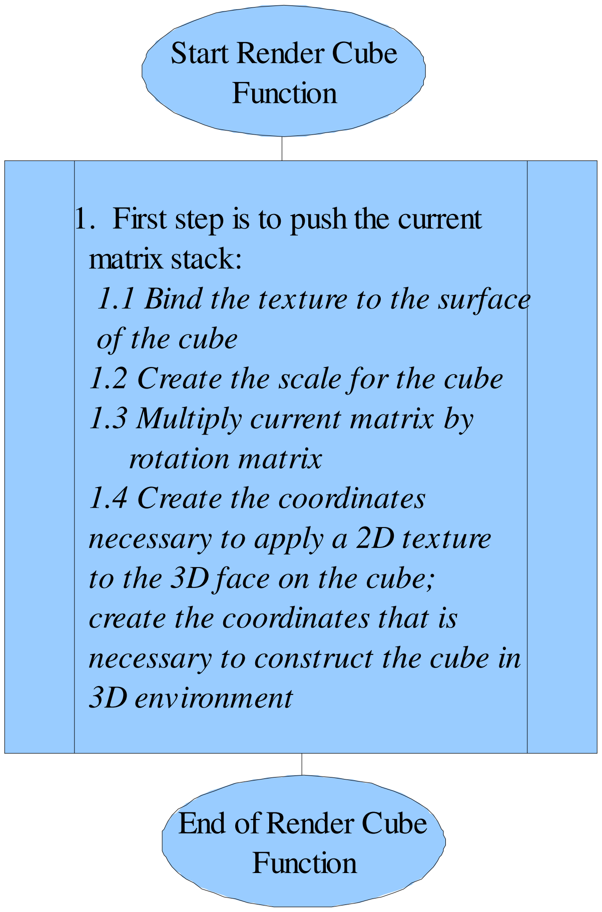
If the user selects the pitch option and chooses the left then pitchRotation-=floating point number else pitchRotation+=floating point number.

If the user selects the roll option and chooses the left then rollRotation-=floating point number else rollRotation+=floating point number.

End of Rotate Function

Render Cube Function:

This function creates the cube in a 3D environment.



1. First step is to push the current matrix stack:

1.1 Bind the texture to the surface of the cube

1.2 Create the scale for the cube

1.3 Multiply current matrix by rotation matrix

1.4 Create the coordinates necessary to apply a 2D texture to the 3D face on the cube; create the coordinates that is necessary to construct the cube in 3D environment

Texture Image

This step consists of two functions. One function loads the image from a file and the other stores pixels passed in a byte buffer in which at the end applies the images where they are being called in the program.

- *CubeApplet.java Class Flow Chart*

This file processes the user's input from the GUI. The swing components for the GUI consist of:

1. Three check boxes: yaw, pitch, roll
2. Three buttons: rotate left, rotate right, stop
3. slider bar: manipulates size
4. Three combo boxes: texture, lighting position, and lighting color

Check Box Control:

If the user chooses the yaw option, then yawRotation = true. However, the cube won't rotate unless the user selects rotate left or rotate right. The same logic goes for the pitch and roll options.

Button Control:

If the user chooses the rotate left option, then the yawRotation = true and the direction set to left. The same logic goes for the rotate right button.

Slider Bar:

If the user manipulates the slider bar, then the size manipulation=true and the scale is manipulated according to the value of the slider bar.

Combo Box Control:

For the texture combo box control, if the user doesn't choose an option from the texture box, then the default texture is used. Else if the user does pick texture, the index position is set to the image which is loaded from a file and the pixels are stored in a byte buffer.

For the light position combo box, if the user doesn't choose an option from the light position box, then the light position is set at a default position. If the user does change the position, then the light position is set according to the user's input. Logic goes for the light color.

Conclusion

As a form of preparing for the big picture, that is making the CSDT, the cube rendering project is one that introduces us to the idea of merging to palettes in one and manipulating one through the other. The project teaches a person how to create a canvas in space then paint in a 3Dimensional figure, in this case a cube, and setting up an eye of a camera and adjusting its position in a way to capture the image of that cube in the open space. The trick is to set bounds, locations, and characteristics of the canvas and its shapes and light beams then tell the eye to look towards that direction. This project will be followed with one that follows the same steps with some tweaks. In the upcoming project, the goal will be to expand our horizons and try to think of creating three panels on one platform in which each panel does a specific job, and at the end affects the whole picture. First, there will be a panel of pieces of different shapes that already have some coding built into them. The panel or little window under that will be an open space in which those little pieces of shapes can be dragged and dropped. The third panel will be the space in which those shapes will appear according to the shapes being dropped into the second panel. After this mission is accomplished, then we should be set to start the primary project which is the dance tool.

Acknowledgments

A word of thank you and appreciation goes to our mentor Dr. Tiffany Barnes and Evie Powell for their guidance through this process.

References:

The tutorial that was used to draw the cube:
http://www.morrowland.com/apron/tutorials/gl/gl_rotating_cube.php

The tutorial that was used to create the GUI:

http://java.sun.com/docs/books/tutorial/uiswing/learn/index.html