FINAL REPORT of the SEMESTER PROJECT in INFOGRAPHICS

Tugce Celik

My term project for computer graphics was to design a character and give it a texture. But I had problems to use some libraries on macOS, and I had to change the project because I couldn't find the appropriate texture, image, object.

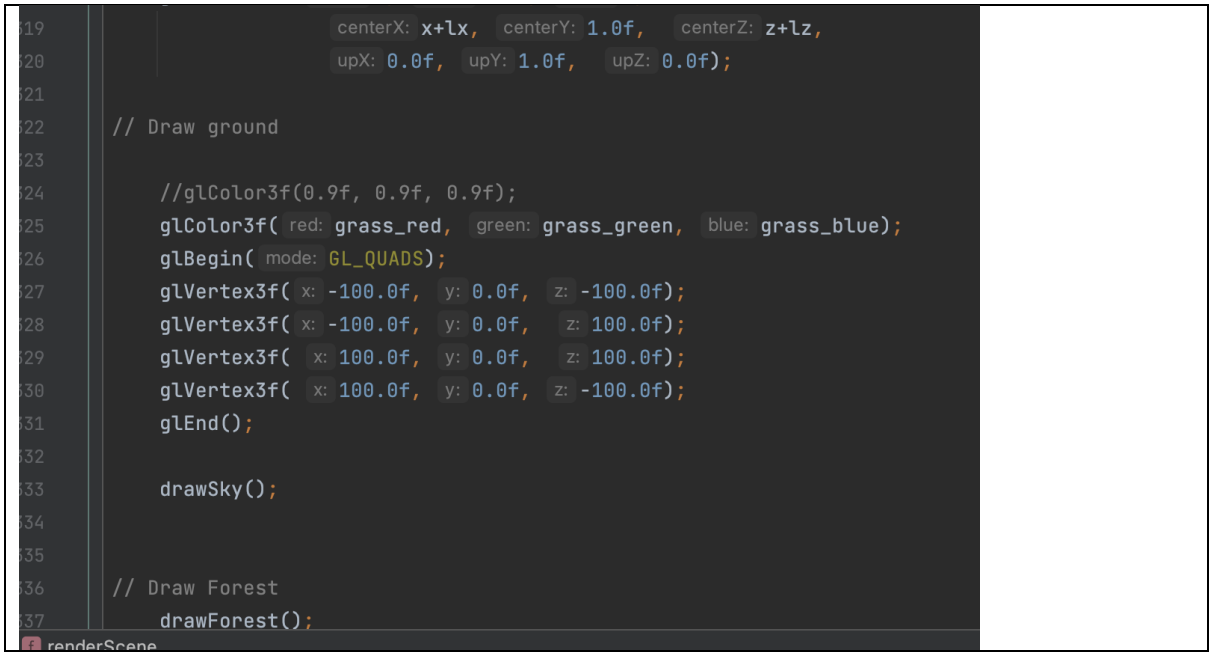
The project mentioned in this report is to create a simple forest using opengl and c++ and modify this forest according to the seasons.

The main feature I wanted to create in the project was walking through the forest created with the camera. I was able to do this, but I could have improved the viewing angle by adding camera movement in the y axis.

I followed these steps to complete this project:

1- Determine the limits and characteristics of the forest and the sky

The created forest was built on the XZ plane with the size of 200x200. Quads were pulled from the summit for this purpose.



```
                    centerX: x+lx,  centerY: 1.0f,   centerZ: z+lz,
                    upX: 0.0f,  upY: 1.0f,   upZ: 0.0f);

// Draw ground

    //glColor3f(0.9f, 0.9f, 0.9f);
    glColor3f( red: grass_red,  green: grass_green,  blue: grass_blue);
    glBegin( mode: GL_QUADS);
    glVertex3f( x: -100.0f,  y: 0.0f,  z: -100.0f);
    glVertex3f( x: -100.0f,  y: 0.0f,   z: 100.0f);
    glVertex3f(  x: 100.0f,  y: 0.0f,   z: 100.0f);
    glVertex3f(  x: 100.0f,  y: 0.0f,  z: -100.0f);
    glEnd();

    drawSky();


// Draw Forest
    drawForest();
renderScene
```

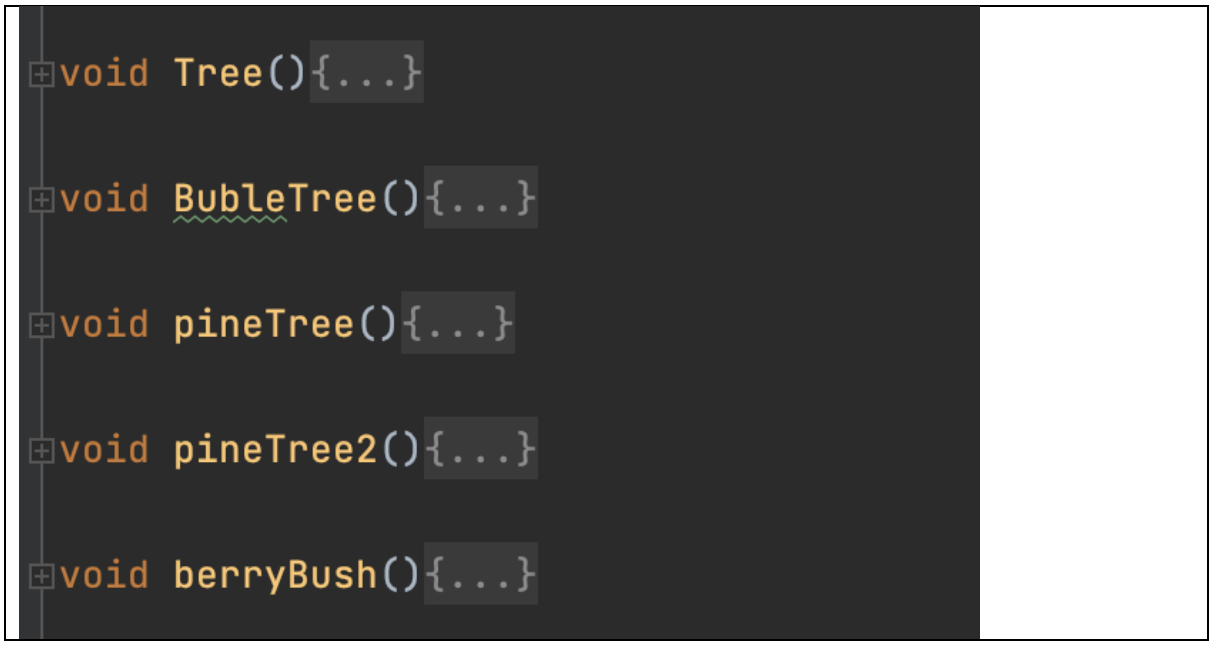
In order to adjust and change the color of the sky, the drawSky function was written and the glutSolidCube() function was used. The size of the cube was chosen at 90 because it is only at these dimensions that a more precise sky is obtained.

2- Create functions to draw forest elements

There are 5 types of trees in the forest I designed.

To draw these trees, the glutSolidCone and glutSolidSphere methods were used.

The main objective is to obtain a tree structure by adjusting the size and position of cones and spheres.



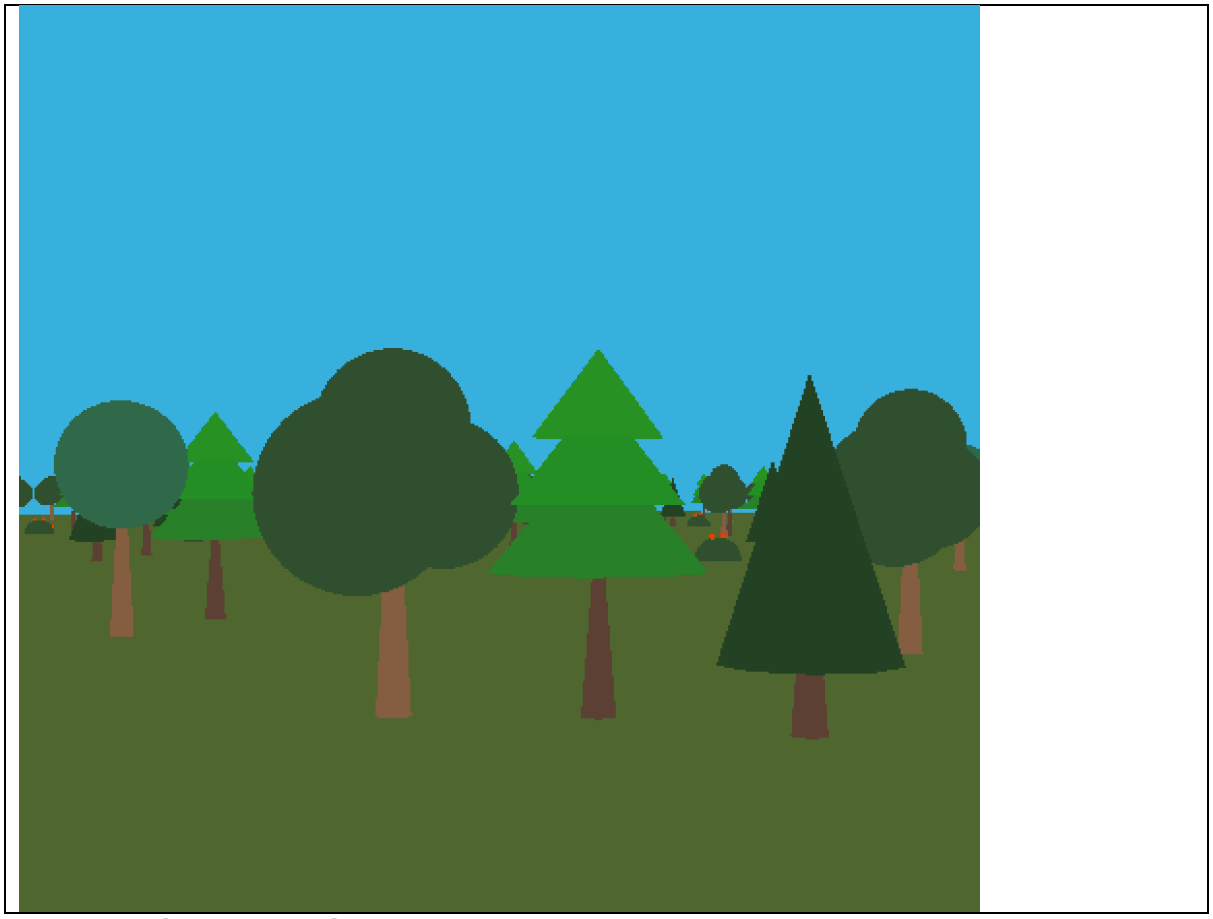
```
void Tree(){...}

void BubleTree(){...}

void pineTree(){...}

void pineTree2(){...}

void berryBush(){...}
```



You can see the tree types above.

The drawForet() function was created to place the trees in the field. With this function, the pen location is set using the nested for loop. This way the designs do not overlap.

To do this, glTranslatef(); is used.

3- Make color adjustments for seasonal transitions automatically

Special colors have been assigned to each season to ensure seasonal transitions.

The colors of the fall, summer and winter seasons have been set for the 'a', 's' and 'w' keys respectively.

glutKeyboardFunc(); We call the processNormalKeys() function we wrote there.

4- Define the camera to move in the defined area and control the angle and movements of the camera with the keyboard keys

We used the gluLookAt() function to set the camera. We have assigned the parameters of these function variables so that we can apply the position change commands we need to move around.

First, we used the arrow keys to move the camera.

While the left and right keys rotate the camera around the Y axis, i.e. in the XZ plane,

The up and down keys move the camera forward and backward in the current direction.

We needed the angle when rotating on the XZ axis. We calculated this as follows:

lx = sin(ang)

lz = -cos(ang)

Just as we want to convert polar coordinates to Euclidean coordinates. Since the initial value is -1, lz is negative. We would like to point out that when updating lx and lz, the camera does not move, the camera position remains the same, only the perspective changes.

And here are the new values of x and z to move the camera forward

x = x + lx * fraction

z = z + lz * fraction

A fraction is a possible application of speed. We know that (lx,lz) is a unit vector, so if the fraction is held constant, the velocity will also be held constant.

The other issue I ran into after doing this was that I didn't want my camera to move while the button was pressed. and when I stopped pressing the key, my camera should have stopped too. To fix this, we will disable callbacks when a key repeat occurs with glutIgnoreKeyRepeat.

When a key was pressed, a variable was set to a non-zero value.

When the key was released, the variable was reset to zero.

On the other hand, since there is no active callback between the pressed key and the released key, we need to check these variables in the render function and update the position and orientation of the camera accordingly. In the initialization section, we have two new variables: deltaAngle and deltaMove . These variables control the rotation and movement of the camera respectively. When it is non-zero, camera action occurs, when it is zero, the camera is stationary. These two variables take the initial value zero, which means that the camera is initially at rest. At the beginning of our code, we will add two variables to track the state of the key, one for the orientation, deltaAngle, and the other for the displacement position delta.

Through these processes, we solved our problem.

5- Assignment of the ESC key to close the program.

Closing the program easily is just as important as running it. I chose the ESC key to close the window, and I made the definitions in the processNormalKeys() function to process the command received from the keyboard.