Homework 1

By Aashish Waikar(20196055)

In this assignment I have implemented Koch snowflake mesh generator, snowflake and animation class for generating smooth animations. I have used triangle primitives to depict the background as hilly. And I have implemented colour gradient in the rectangle mesh(background) to give aesthetic appearance to the background.

1)Koch snowflake:-

I have implemented the Koch snowflake in Fractal.cpp. The generateSnowflake() method generates a fractal mesh centred at the origin of the coordinate system. I have implemented a helper function : side() which takes the concerned mesh and two vertex coordinates(a,b) as input. In side() function I first calculate the 3 coordinates of the first protruding triangle(d,e,f) on the side (a,b) while generating fractal. First, I add d,e,f to the mesh and then recursively call side() on (a,d) , (d,e) , (e,f) , (f,b). And this function being called on the original 3 sides of the triangle(in generateSnowflake()) generates the whole fractal.

2)Animation and Snowflake class:-

The Snowflake class inherits from renderObject class. I have additionally added member variables to this class which control

a) The average x coordinate of the snowflake
b) Change in the xcoordinate(delta(x)) of the snowflake
c) Rotation speed of the snowflake

The constructor of this class takes a material and mesh instance as input and the Render function(inherited from renderObject class) is called on the snowflake instance(in animation class) to render animation.

In Animation class, I have added additional data members previous_t and current_t. These are needed to keep track of the changes in time happening each time animate() function is called in the while loop in main.cpp.

In animate() method of Animation class, I first use glfwGetTime() to calculate time gap. Then I iterate over the list of all snowflakes and find the previous position and orientation using getPosition() and getOrientation() methods. After calculating the respective changes in position and rotation according to time gap, I update them by the methods setPosition() and setOrientation() on the snowflake instance.

The translation motion of the snowflakes is both vertically as well as horizontally. In the horizontal direction they oscillate b/w 2 endpoints. In the vertical direction I have set the lowermost point. On reaching this point, the snowflake is made to remain that way for 2-3 seconds to give the perception of snowflake settling on the ground. Then I set its position back to the uppermost point and then this whole process repeats for continuous snowing appearance. For the rotation, I had to compute the rotation matrix according to change in orientation.

3)Background:-
For the background

a) I have implemented colour gradient in the rectangle mesh to give perception of the sky.(using perVertexColorMaterial class)
b)Used miniature triangle primitives for stars in the sky.
c)Used triangle primitive to make hills in the background. I have used colour gradient in these primitives to give depth perception.

s

4)Creativity:-
I have made a scenic background of hills and stars(used colour gradient for hills at different depths). Also the snowflakes on reaching the bottommost point are moved to the topmost point after waiting for 2-3 seconds to give the appearance of snow getting settled on the ground.