Snowman 3D Animation

20150860 김태윤

1. Background

The background is made up of three large parts: the sky, ground and the mountains. The sky contains the moon, stars and the clouds. The moon and the stars are 2D objects, so they will not be discussed here. The clouds, however, are 3D objects made up of 10 spheres. They are placed in the sky in random positions. Unlike the stars that are placed in the same z-coordinates, I gave some randomness to the z position of the clouds so that they do not overlap. The ground is a flat surface that is rotated by the x-axis to lay flat. A gradient color is given to show the color of a snowy land. On the ground, there is a mountain range that spans along the x-axis. They are a collection of cones which have random base radius and height. They are positioned behind the snowman so that they appear as a scene in the background.

1. Snowman
   1. Three different kinds of objects other than cubes

The snowman contains three kinds of 3D primitives: sphere, cone and cylinder. The head, body and the eyes are spheres, nose is a cone, and the mouth is a cylinder. The sphere was created by dividing the sphere into longitudes and latitudes and using the spherical coordinate system. Each layer of vertices were connected to fill the shell of the sphere. It was more intuitive to connect vertices one-to-one, so I defined a function that adds four vertices that form a quadrangle in counter-clockwise order. Making a cone and a cylinder was relatively easy after implementing a sphere. The base circle was created first and each edge was connected to the apex to fill the side faces of a cone. For a cylinder, the two circles are drawn first then the quadrangles on the sides are filled. Other than these primary 3D objects, I created custom objects for the arms, the hat and the cane. The vertex data are added to their meshes and positioned on the snowman accordingly.

* 1. Building a snowman

The snowman has two main parts: head and the body. The body is the parent of the head. Other parts are attached to either head or the body. Parts such as the face and the hat are attached to the head because they have to rotate with the head if needed, and similarly the arms and the bow tie are added to the body. The meshes for the additions to the snowman are created in the main function and passed into the snowman object. The snowman object has functions such as *AddFace* and *AddArms* to attach respective items to the snowman. Those functions also position the objects with respect to their parents. They scale, and specify the parent of the objects as well.

1. Animation
   1. Animation of the snowman

The animations of the snowman include waving of the left arm and swinging of the cane. These animations take the elapsed time into account for constant animation in different machines, just like assignment 1. The swinging and waving animations has similar algorithms, however, previous animations I had implemented were all constant motions. In other words, I moved an object in one direction or rotated around an axis. However, the waving animation included changing of the direction. The direction of the rotation has to change when it reaches either ends. To implement this animation, I tried getting the current angle of the object from the orientation matrix but failed. Instead, I created a global variable that stores the current angles of the arm and the cane, and the rotating directions. This way, I could check the angle value and determine if I should update the value or change the rotation direction.

* 1. Keyboard and mouse interaction

There are several interaction features in this project. For the mouse, the snowman rotates following the mouse cursor. Pressing Q on the keyboard activates and deactivates mouse cursor interaction. The implementation of the rotation includes calculating the distance of the cursor from the center and the angle it makes with the x and y axes. The distance was used to calculate how much of an angle to rotate and the angle of the cursor vector was used to determine the axis of rotation. The axis of rotation is perpendicular to the angle vector, which is the vector from the center to the cursor. The angle of rotation was limited to 100 degrees so that it does not rotate too much. Right clicking on the mouse switches the part of the snowman that rotates between the head and the body. Pressing Z fixes the rotation axis as the z-axis. Pressing it again deactivates the z-axis rotation. Interaction for pressing A, W, X, and C was explained in assignment 1.

1. Creativity

Making objects that are mathematically defined, such as spheres and cones, are relatively easy. However, I wanted to add other objects that are not mathematically defined, such as a tree branch, a hat and a cane etc. To create a mesh that formed these kinds of shapes, I needed the vertices of triangular faces in the counter-clockwise order. I modeled the 3D objects in a software called Blender and exported the models as obj files. The obj file contained the coordinates of the vertices and indices of vertices that formed faces, but the vertices appeared only once in the list and were not in the counter-clockwise order. Instead, they were just a list of vertices. Therefore, I created a parser class called *ObjImporter* that reads the list of vertices and the indices of vertices in each face. Then it made a new list that has vertices of triangular faces in order. I could simply add these vertices to the mesh and form a face for every three vertices in a class called *BlenderObject*. The downside of this method is that the rendering is slow due to large number of faces. *So rendering will take a few minutes until the console output displays “vertices done” and “faces done” for all objects..*

One difficulty in implementing these custom objects is that the origin point of each object is ambiguous. For example, the origin point of the cane object is on the stem. Thus when I wanted to swing the cane that hangs on the arm, the rotation was not natural because it rotate across the stem. It should have been at the center of the curved handle part. To fix this, I added an anchor at the rotation center of the cane and set the parent of the cane as the anchor. This way, if I rotate the anchor, the cane rotates with it, turning around the correct axis.