CS380 Introduction to Computer Graphics
Homework Assignment #2

Do you want to build a Snowman?

20160042 Inyong Koo

In this document, I will explain how my implementation satisfies the given specifications. I will not copy the code itself, but rather explain in plain text. If you want to see how the actual implementation looks like, please refer to the code lines and comments of attached files. I'll note where to look for each task.

1 Background

On homework 1, I designed a snowing night scenery with the full moon and the stars. I wanted to reproduce the scenery, but in 3D. The full moon was simply implemented as a yellow sphere. Ground is no longer a rectangle, but A circle normal to y-axis with radius. And I randomly assigned position on my 'celestial hemisphere' that covers our ground. The moon is also on this celestial hemisphere, and they rotate around the z-axis with angular velocity of 1 degree per second.

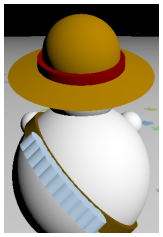
3D-rendered snow animation follows similar logistic of the homework 1 implementation - *smooth random translation, remains on ground after fallen* - but with increased degree of freedom. I restricted the snowing area around the snowman, since our flat-earth is quite broad (radius `100.0f`).

(Please refer to `Geometry.cpp` and `Snowflake.cpp`)

2 Snowman

2.1 Primitives

(a) Front



(b) Back

Figure 1: Snowman

Our next objective is to build a snowman using at least 3 different kinds of objects(primitives) other than cubes. The snowman I built is shown in Figure 1. I composed it with spheres, cones and cylinders.

Sphere I referred to the link on KLMS webpage [1] and implemented approach using icosahedron. I chose the approach because the article suggested that it shows minimum error. I manually wrote initial 12 vertices and 20 faces, and recursively proceeded subdivision (5 times) for each triangle to create a sphere mesh.

Cone For given input of height and base radius, I set position of apex, and 100 vertices along the base cicle. Then I created the cone mesh, but did not render the base surface, but rather render both inner and outer surface of the side. This is because I used cone to render the nose and the hat of the snowman, and I needed to show inner surface of the hat.

Cylinder For given input of height and base radius, I set 100 vertices each along the upper and lower circle. I rendered the tube and two circles.

(Please refer to `Geometry.cpp`)

2.2 Building the snowman

My snowman is composed by two spheres, `body` and `head`. `body` is the parent of `head`.
`body` has children - two `hand` spheres, and a `belt` cylinder. `belt` has children of a set of `load` cylinder.
`head` has children - two `eye` spheres, a `nose` cone, and a `hat` object's cone. The `hat` is composed of a cone, a cylinder, and a sphere; the cylinder and the sphere has the cone as its parent.

Scale, position, and orientation of each object are inidividually set with respect to its parent. I'm pretty sure that the codes in `main` function and the result will be sufficient explanation of my snowman design.

(Please refer to `main.cpp`)

3 Animation

I tried to implement smooth gaming control of the snowman.

You can use WASD or arrows keyboard to move around the snowman. Keyboard input does not make the snowman to move in uniform motion principle, but rather act as applying force towards the direction, so that the snowman will accelerate towards the direction. Snowman also turns its body towards the direction. Press left control to immideatly brake all horizontal movement. If the snowman tries reaches the boundary of the environment, it will respawn at the origin.

Pressing space bar will make the snowman to jump, and the snowman will fall smoothly in my custom gravitational field. And yes, double jump is implemented.

You can also change the viewpoint of the camera by placing your cursor to the left/right edge of the window. Keyboard input will follow your viewpoint, so you can always use W key to go forward towards your screen direction. Pressing F key will immediately align your viewpoint to the snowman's viewpoint.

Click the snowman's head to change color of its hat. There are 4 options.

If you click anywhere besides the snowman, you can throw a snowball towards the snowman's moving direction. Hold click to throw the snowball further. You have 12 loads, and you should reload the snowballs by pressing R key or clicking the snowman's body. The remaining loads are shown on the belt around the snowman's body.

Maybe Try to hit the moon with your snowball?!

Press H key to see the instructions at the program.

(Please refer to `animation.cpp`)

4 Creativity

- My 3D background is an authentic model of flat-earth celestial sphere model (maybe in the pole area) with constant animation. (Animating background was not mandatory.)
- All my creative implementations of Homework 1 were all implemented in 3D for this assignment too. For example, Each snowflake has an independent, smooth, and natural descending path due

to setting random variables on its horizontal motion. Fallen snowflakes will remain on the ground and make a flowerbed.

- I implemented basic first-person shooter (FPS) gaming physics and logistics, with acceleration and gravity, respawn and reloading system. Camera adjustment was also not mandatory, but was implemented to follow the snowman and change viewpoint with intuitive control.
- I implemented a small easter egg. If you hit the moon with a snowball, the moon will instantly change to red moon and the snow will disappear. You should respawn to undo the process.

References

[1] Four Ways to Create a Mesh for a Sphere,
`https://medium.com/game-dev-daily/four-ways-to-create-a-mesh-for-a-sphere-d7956b825db4`