

CS 148 Final Project Report

A Christmas Snow Globe

By Boyu Liu, Rui Yan, Tianci WANG

1. Problem Statement

Our target is to construct, in real time, a Christmas Snow Globe, mainly consisting of a glass ball, a system of drifting snowflakes, a base propping up the glass globe, a desk plain supporting the base and a sky background. Below is a prototype of our project.



To be more specific, our background is expected to display 3D environment characteristics, allowing users to navigate within. Our glass ball is expected to possess optical properties of glass, reflecting and refracting lights. The system of snowflakes will simulate drifting, melting and regenerating and spraying. And above features will be implemented in real time, providing controls of snowflake mode and choice of perspectives.

2. Overview of Steps

- a. Setup skybox
- b. Use small triangles to build a huge sphere
- c. Add refraction and reflection to the sphere
- d. Draw snowflake and simulation different mode of movement
- e. Set base

3. Detailed Progression

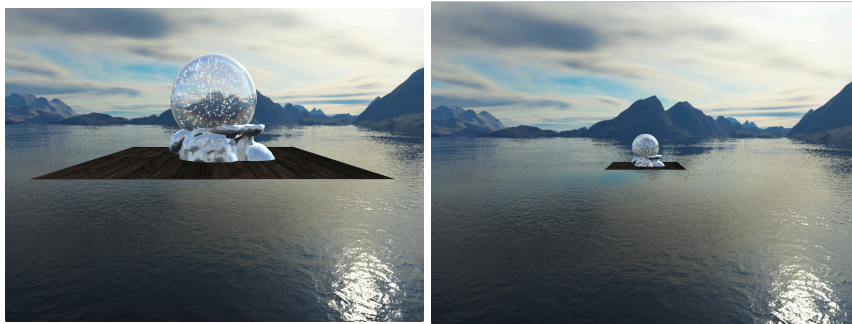
- a. **Setup skybox**

We use GL_TEXTURE_CUBE_MAP to store the textures of six sides, then we can access the texture using “samplerCube” in the shader by bind GL_TEXTURE_CUBE_MAP before draw the elements.

And to make the skybox looks far away(infinity), we manually eliminate the shift transform in view matrix, such that the relative position of skybox and camera in NDC remains unchanged.

```
glm::mat4(glm::mat3(camera.GetViewMatrix()))
```

So the effect after:



b. Use small triangles to build a huge sphere

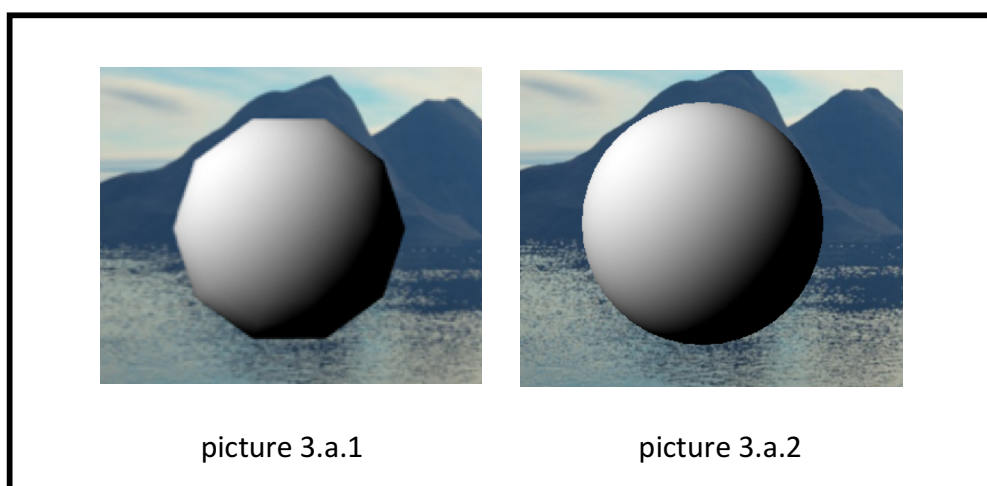
Since we stuck to not using GLUT library, the sphere has to be made up by triangles.

By enumerating θ and ϕ in spherical coordinates in certain number of longitude *longs* and latitude *lats*, we have

$$x = \cos\theta\cos\phi, y = \sin\theta\cos\phi, z = \sin\phi$$

and the sphere is divided into *longs* * *lats* number of quadrilateral, or 2 * *longs* * *lats* number of triangles. If *longs* and *lats* are large enough, the polyhedron looks like a sphere.

Here is what does the sphere look like when *longs* = *lats* = 10 (picture 3.a.1) and when *longs* = *lats* = 40 (picture 3.a.2)

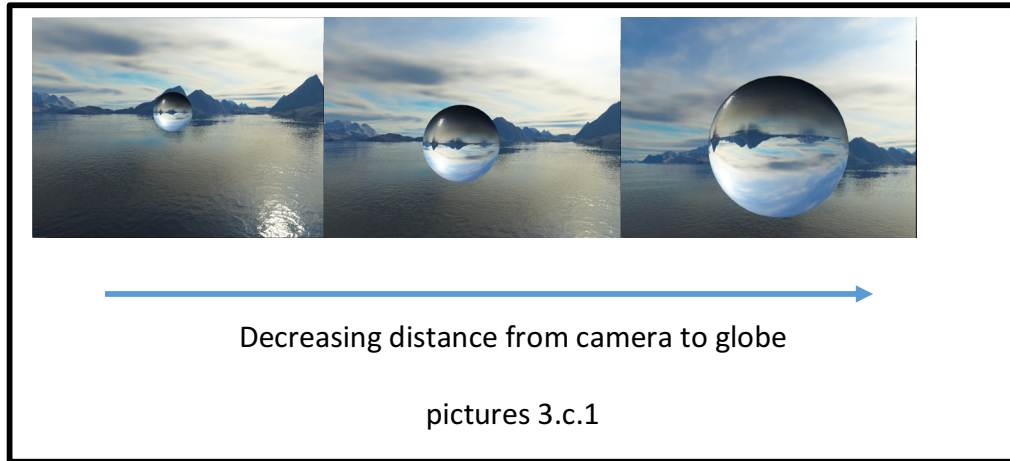


picture 3.a.1

picture 3.a.2

c. Add refraction and reflection to the sphere

For milestone, we simply implement static environment mapping and map the skybox to the snow globe. In this manner the distance between the viewing position and the sight incident position doesn't change the reflection and refraction we get looking at the ball (see pictures 3.c.1). The visual effect is similar to looking at a painted ball with the image of sky on it.



To obtain glass optical characteristics, we've done some improvements.

Firstly, we take the relative position between the camera and the incident point into account when performing shading. This approach delivers better user experience when one explores the globe from different aspects.

Secondly, we simulate two physics phenomenon, namely Fresnel Reflection and Chromatic Aberration. Fresnel Reflection is the reflection of light predicted by Fresnel equations, which describe the behaviors of light when moving between media of different refractive indices. In actual implementation, we adopt Schlick's approximation, which gives us an approximate value of the specular reflection coefficient R (see pictures 3.c.2). Chromatic aberration is a dispersion effect, where lights of different colors correspond to various refractive indices for the lenses (,in this case, the globe). To implement this, we simply assign different refractive indices for Red, Green and Blue, the calculate the refractive color separately for these three. The resulting refraction is the combination of the three parts. (see pictures 3.c.3)

$$R(\theta) = R_0 + (1 - R_0)(1 - \cos \theta)^5$$

$$R_0 = \left(\frac{n_1 - n_2}{n_1 + n_2} \right)^2$$

Schlick's Approximation
pictures 3.c.2



Decreasing distance from camera to globe
(Fresnel reflection, no chromatic aberration)



Decreasing distance from camera to globe
(both Fresnel reflection and chromatic aberration)

pictures 3.c.3



picture 3.d.1

d. Draw snowflake and simulation different mode of movement

Firstly, I used a cube to represent a piece of snowflake. This is how it looks like.
(500 snowflakes, the cube length of which is $\frac{1}{30}$ of the radius of the sphere)

Obviously, this did not look like snowflakes at all. When we were considering loading a snowflake object or even drawing by ourselves with a lot of triangles, we came up with a tricky solution: Add an alpha value for each face of the cube; the closer to the center of the face, the larger

alpha value it be. Let d be the distance of the fragment with the center of the cube, l be the length of the cube side. After a few trials, I chose exponential function and here's where I finally settled.

$$Alpha = e^{-(\frac{d}{l*2-1})^{*10}} * 0.7$$

The result of 800 snowflakes, the cube length of which are 1/10 of the radius of the sphere (3.d.2). By zooming in, we admit that the snowflake was more like an ice cube or snowflakes with spark (3.d.3), but looking from a distance, it works pretty well.



picture 3.d.2



picture 3.d.3

e. Set base & floor

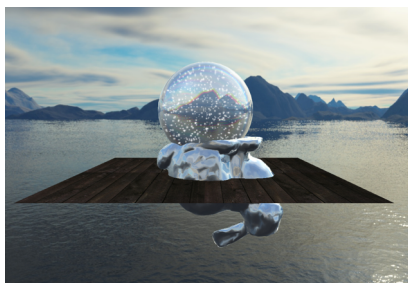
We used the bunny legs as the base, which looks perfect and like a miracle!

To be more detailed, we loaded the bunny into the program, and set the reflection shader, then we wanted to put it inside the snowball.

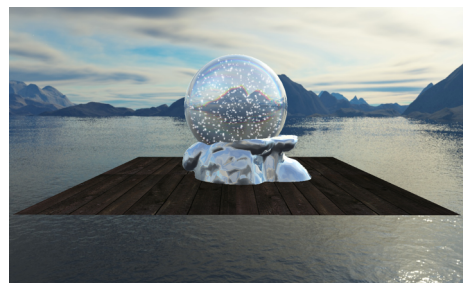
And as we adjusted, we see the following effect:

This seems artistic if we don't see the head of the rabbit. (picture 3.e.1)

So we make that part transparent, by enable blending and set the alpha to be 0 on certain condition: (picture 3.e.2)



picture 3.e.1



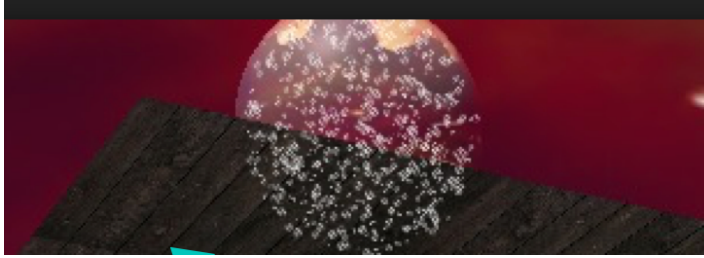
picture 3.e.2

4. Depth problem:

To let the snowflake show in the glass ball, we tried several methods.

(0) This is a problem because by default, the depth test will treat the inside object as they are covered by the glass ball, which prevent us from seeing the beautiful flakes.

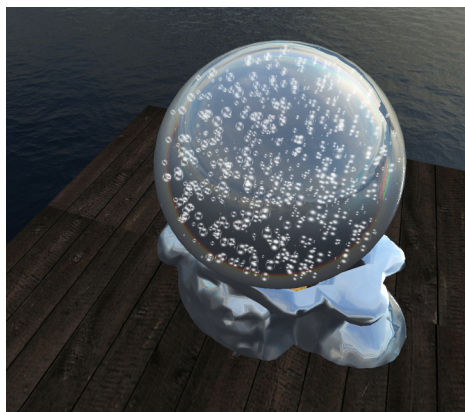
(1) Clear Depth Buffer: we first try to clear depth buffer to let the depth test forgot the glass ball, but then there is a dilemma: floor need to precede the snowflake from bottom, snowflake need to precede the glass ball, the glass ball need to precede floor from up. So without consistent depth test, it is possible to solve this dilemma.



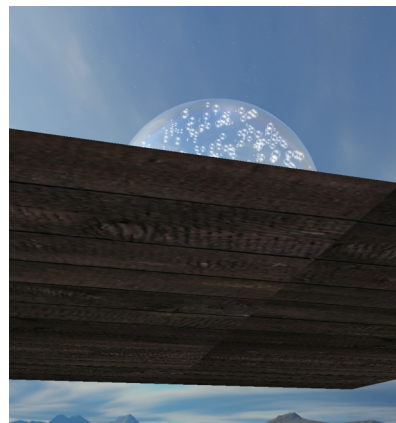
(2) Blending: If we blend the snowflake with the ball, it will deprecate the reflection-refraction effect on the glass ball.

(3) Depth mask: this make the glass not write but only read the depth buffer, so that the snowflake will not be covered by the ball. This does not work because the back of the ball will also show in the front, destroying the view of the ball.

(4) Disable depth test when drawing snow. This works with cautious order of drawing! We draw the ball first, then the snowflakes (disable depth test). Then we draw floor, And because snowflakes are in the ball, so the whether the snow is covered by floor is similar to whether the ball is covered by floor, so it works!



picture 4.1



picture 4.2

5. Division of Labor

- (1) Boyu LIU: skybox setup, various modes of snowflakes movement, adjustment of depths and relationships between the objects.
- (2) Rui YAN: Build the large sphere, draw the snowflakes, set a dull base (not used in the end), and falling movement of the snowflakes.
- (3) Tianci WANG: Shading glass globe and bunny base.

6. Source

Physics from:

https://en.wikipedia.org/wiki/Fresnel_equations

https://en.wikipedia.org/wiki/Schlick%27s_approximation

https://en.wikipedia.org/wiki/Chromatic_aberration

OpenGL Tutorial:

<http://learnopengl.com/>

Libraries:

glew, glfw, glm, No glut!

OpenGL 3.3 core profile

Shader.h, Camera.h from Homework 3

Bitmap.h from:

https://course.cs.ust.hk/comp5421/Password_Only/projects/impressionist/index.html

...

7. Demo link

<https://youtu.be/q9jQPXC0De4>