COMP3811 Coursework2
Theme: Merry Christmas
Bingle Wang sc20b2w, Ruoheng Dong sc20rd

Techniques

1. Commonly geometry construction (Plane/Grid, Cube, Sphere, Cylinder).
   By creating a custom class GeometryGenerator, it provides the ability to create basic geometry and calculate information about the vertex coordinates, normals, and texture coordinates of each geometry by using the corresponding mathematical algorithms.

2. Texture Mapping.
   Texture mapping techniques are fully used for objects in the scene, including snowmen, Christmas trees, gift boxes, snowy ground, houses, sky boxes and snowflakes.

3. Skybox.
   The sky box is composed of a cube, and the six sides of the cube are each pasted with an overall seamless background mapping to form a panoramic effect. One of the implementation details is to cancel the panning part of the sky box view matrix so that the sky box will move with the camera, creating a feeling of being far away from the sky.

4. Shadow Mapping.
   Shadow mapping is a common technique used in modern graphics/games, and this program further implements a soft shadow effect with PCF based on the implementation of basic shadow mapping.
   The idea behind Shadow Mapping is very simple. We render from the perspective of the location of the light, what we can see will be lit, and what is invisible must be in the shadows. Suppose there is a floor with a large box between the light source and it. As the light source at the direction of the light, you can see this box, but cannot see part of the floor, this part should be in the shadows.
   A value in the depth buffer is a depth value between 0 and 1 corresponding to a slice element in the camera view. What if we render the scene from the perspective view of the light source and store the result of the depth value in the texture? In this way, we can sample the nearest depth value seen by the perspective view of the light source. Eventually, the depth value will show the first slice seen from under the perspective view of the light source.

Details of application in OpenGL

The following steps are roughly divided into OpenGL.
1. A texture and frame buffer object (FBO) frame buffer is created to hold the depth value information. The depth value information does not need to be rendered in color, so there is glDrawBuffer(GL_NONE); glReadBuffer(GL_NONE);

2. Calculate the illumination space matrix. Here because it is parallel projection, you need to set the parallel projection matrix first, the light direction should be from the light source to the center, so light space transform (light space matrix) = ortho * lookat(-light direction,center,up), which is the projection matrix multiplied by the light direction.

3. In the vertex shader, the coordinates in the model space are transformed into the light space: Light Space Position = Light space transform * Model * Position. Note that here the position should be transformed to the homogeneous coordinate system. (Tip: the w value of the vector into the homogeneous coordinate system is 0 and the point is 1, so the position is vec4(position, 1)).

4. In the fragment shader, you should first find the depth value of each point. This is also very simple, the light space coordinates XYZ in the vertex shader divided by w can be found, and transformed to the [-1,1] interval to get the coordinates of the light space. At this point, the z value is the current depth value, and the XY of this coordinate is brought into the depth texture to index the r value as the nearest depth value. If the current depth value is greater than the nearest depth value, then the pixel is in the shadow.

Blinn-Phong Lighting

The so-called Lambert diffuse reflection model is based on the floodlight model with the addition of the diffuse reflection term. Diffuse reflection is the reflection of light in all directions from the point of incidence after being incident at a certain angle, and the intensity of the light reflected in each different direction is equal.

Only when the incident light is perpendicular to the plane can it receive all the light energy in its entirety, and the more inclined the angle of incidence the greater the loss of energy, specifically, we should multiply the light intensity by a $\cos\theta = l \cdot n$, where l is the direction of the incident light, and n is the direction normal to the plane.

In addition to the distance from the light source to the reflection point mentioned in diffuse reflection, it should be noted that the observation direction is important in specular reflection, specifically, the reflected light can be seen only when the observation direction is concentrated very close around the reflection direction. Therefore, in specular reflection will be considered R and v of the angle α is given by the following equation.

$$L_s = k_s \left(I/r^2\right) \max(0, \cos\alpha)^p$$

Where k_s is the specular reflection coefficient, 'I' is the incident light intensity, and 'r' is the distance from the light source to the point of incidence, note that here in the max to eliminate the light greater than 90 °, we also multiplied by an index p, add the reason for the direct, because the further away from the reflected light should not see the reflected light, the need for an index p to accelerate the attenuation

Particles

Particles class implements simple particle objects, including position, rotation, scaling, affected by gravity, survival period, survival time and other common properties, the program uses particle effects to achieve snowflakes and smoke from chimneys.

ImGui

ImGUI is integrated into the program to adjust various rendering parameters and switch on/off effects at runtime.

1. Diffuse(Ground)

The rough surface of the snow mostly shows the diffuse reflection effect.

2. Specular(Snowman)

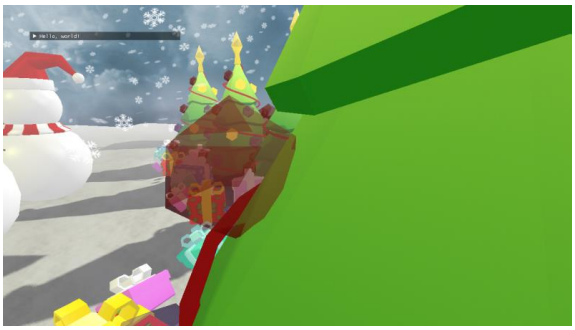
The snowman uses a higher level of specularly reflected light to express texture.

3. Emission(Red Light)

The decorative lights on the Christmas tree are self-luminous when they are on

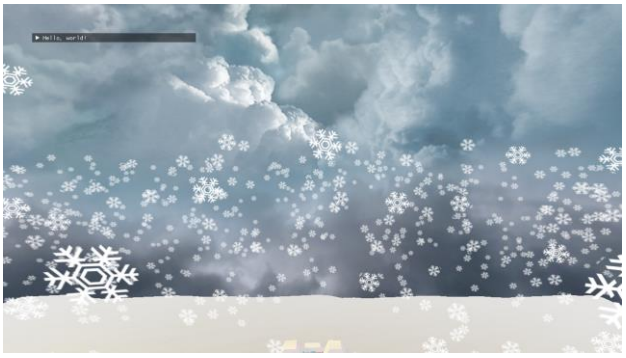
4. Transparent

The decorative lights on the Christmas tree are translucent when they are off.

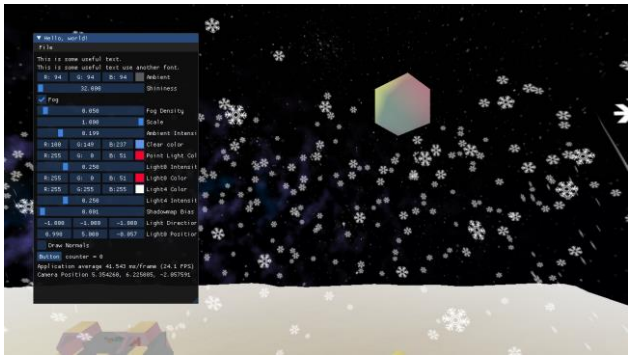


5. Particles

Particle class was created to implement simple particle effects such as snowflakes, and smoke from chimneys.

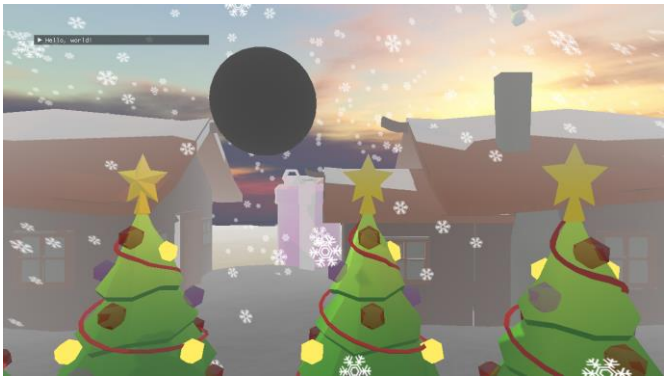


ImGui provides adjustment of various rendering parameters, including ambient light, diffuse light, specular reflection, fog effect, point light color, point light position, directional light direction, and directional light color.



6. Movement-based on bezier cubic curve

The program implements a moving path based on three (four control points) Bezier curves.



Interaction

W/S/A/D - To move the camera for roaming.
Q/E - Lifting camera.
Drag when RMB is down - Rotating camera view.
Draw when MMB is down - Pan camera view.
Num Pad 1/2/3/4/5 - Turn on and off the light source.
H - Turn on and off the texture mapping of the house windows (on/off lights).
Num 1/2/3 - Switch to snowman mode.
Up/Down(Snowman mode) - Move the snowman around.
Left/Right(Snowman mode) - Rotating snowman(and view).
Space(Snowman mode) - Jump.
Num 4/5/6/7/8/9 - Toggle Christmas tree decoration lights.
I - Toggle ImGUI display.
+/- -Increase / Decrease skybox rotating rate.
P - Write screenshot to screenshot.png file.

References

(1) For writing screenshots to png files. Referenced in the writeToPNG function in main.cpp. lodepng https://github.com/lvandeve/lodepng
(2) ShadowMappingalgorithm:https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping

Task Attribution

Bingle Wang Attribution: 50%	Snowman modeling/texturing(Blender),lighting model, GeometryGenerator, bezier cubic curve, Skybox, Particle, ImGui.
Ruoheng Dong Attribution:50%	Skybox, Particle, shadow mapping, interaction, ImGui, screenshot saving function., scene design, texturing.