Treasure Hunt - Group 17

Members

111550012	鄭睿宏
111550093	朱驛庭
111550138	林辰翰

Introduction

After Mojang updated Minecraft to version 1.30, the developers announced that the theme of this version is treasure hunting. Having not played Minecraft for a long time, you immediately launched the game. Upon entering, you discovered a treasure chest not only appearing right in front of you but also moving toward you. Despite feeling a bit suspicious, you decided to open the chest. However, after opening it, you found that your Steve character was stuck. Suddenly, a Creeper appeared behind you, and you were blown up.

Story Outline: A Thrilling Accident

Act 1: The Temptation of Treasure

- In the glow of the setting sun, Steve crouches in a forest clearing, intently examining a mysterious treasure chest before him. The wooden exterior is weathered with age, and the lock gleams faintly with a metallic chill, as if mocking his efforts.
- “What could be inside?” His heart races with excitement and tension as he carefully fiddles with the lock.

Act 2: An Unexpected Crisis

- The camera shifts to Steve’s face, clearly focused, with a slight furrow in his brow. But behind him, a shadow of green silently emerges.
- It’s a Creeper. The atmosphere instantly freezes. The creature creeps closer, its body swelling unnaturally and emitting a blinding glow. A sense of danger fills the air.

Act 3: The Explosion’s End

- “Hiss—” The Creeper lets out a low sound. In the next moment, the screen is engulfed in a blinding white light. The explosion erupts with a deafening roar, obliterating everything in an instant— the treasure chest, Steve, and all objects vanish as if they never existed. All that remains is a desolate clearing.

Features

1. **Scene Setup**
    - **Environment Creation:**
        - **Create a scene using OpenGL with a cubemap shader. The bottom picture will serve as the ground where the events occur.**
    - **Object Models:**
        - A chest model for the treasure.
        - A Steve character model to represent Steve.
        - A Creeper model to appear and interact with the scene.

2. **Geometry Shader Utilization**
    - **Creeper Growth and Shimmering Effect:**
        - Use a geometry shader to dynamically adjust the size of the Creeper, making it grow larger as part of the buildup to the explosion.
        - Create a shimmering effect by modifying the Creeper's vertices or adding noise-based distortions to its surface.
3. **Animation and Interaction**
    - **Steve's Interaction with the Chest:**
        - Animate Steve attempting to open the chest (e.g., moving hands or changing pose).
        - The head is able to move to look around.
        - Steve is able to move the hands and legs while moving forward.
    - **Creeper Movement:**
        - Creeper is able to move the heads and legs while moving forward, and while moving the body will bounce.
    - **Camera Movement:**
        - Implement smooth transitions for the camera to shift its focus from the chest to facing Steve and revealing the Creeper behind him.
4. **Explosion Effect**
    - **Geometry Shader Explosion:**
        - Use the geometry shader to create an outward explosion effect by scattering fragments or particles representing debris.
        - Add fading effects to these particles for realism.

Implementation Details

Steve Implementation

> Source File (Steve.cpp), Header File (Steve.h)

- **Skeletal Animation System**

  The class is designed to manage different body parts independently, allowing for more precise control over animations. Each body part (e.g., head, body, hands, legs) is represented by a ModelPart structure, which includes position, scale, rotation, and a pointer to an Object.
- **Public Member Functions**
    - `setup()`: Initializes the model parts by loading their respective 3D models and textures. This function sets up the initial transformation matrices for each part.
    - `render()`: Uses a shader program to render each body part. It sets the necessary uniform values for the shader and calls the render function of each Object.
    - `update()`: Updates the animation state of the character. It handles walking animations, returning limbs to their original positions, and death animations.
    - Movement Functions:
        - `walk():` Toggles the walking state.
        - `moveForward()`, `moveBackward()`, `moveLeft()`, `moveRight()`: Move the character in the specified direction and update the position of all body parts.
        - `stopMoving()`: Stops the walking animation, let hands and legs return to their original position naturally.
        - `rotateHead()`: Adjusts the head's rotation based on input cursor offsets, with constraints to limit the rotation angles.
        - `die()`, `revive()`: Manage the character's death state, triggering a death animation and resetting the state upon revival.

- **First-Person Camera Position**

  To add a first-person camera view, we introduce a new function `getCameraPosition()`. This function calculates the camera's position based on the character's eye position and view direction, providing a realistic first-person perspective. And we can control the camera by our cursor movement as in a real Minecraft game.

Chest Implementation Techniques

> Source File (Chest.cpp), Header File (Chest.h)

- **Model Setup**
  - `setup()`: Initializes the chest's container and lid by loading models and textures, setting their initial positions, scales, and rotations.
- **Rendering**
  - `render()`: Uses a shader program to render the chest's container and lid, setting uniform values for view, projection, and model transformations.
- **Animation Control**
  - `update()`: Manages the opening and closing animations of the chest lid using a smooth step function for easing effects. The lid's rotation is updated based on its current state (open or close).
- **Public Member Functions**
  - `open()`: Sets the chest to open state.
  - `close()`: Sets the chest to close state.

Creeper Implementation Techniques

> Source File (creeper.cpp), Header File (Creeper.cpp)

- **Setup Method**
  - Initializes the creeper's position and sets up the body, head, and legs by loading their respective models and textures.
  - Toggle Scale and Shimmer:
    - Toggles the scaling and shimmering effect. If the creeper has already exploded, it resets the explosion and scale.
- **Update Method**
  - Handles jumping physics by applying gravity and updating the vertical position.
  - Updates the walking direction and position if the creeper is walking.
  - Updates the positions of all model parts based on the current position and body height, to avoid collision.
  - Manages the scaling and shimmering effect, transitioning to an explosion if the maximum scale is reached.
  - Updates the leg animation and body bounce during walking.
  - Gradually increases the explosion factor if the creeper is exploding.
- **Render Method**
  - Uses the shader program to set uniform values for view, projection, and material properties.
  - Renders each part of the creeper (body, head, legs) by setting the model matrix and calling the render method of the Object class.

Vertex Shader (creeper.vert)

- Transforms vertex positions and normals.
- Passes texture coordinates, explosion factor, and ash noise to the fragment shader.

Fragment Shader (creeper.frag)

- Shimmering Effect
    - If isShimmering is true, a sinusoidal oscillation based on shimmerTime is computed to simulate a shimmering effect. This oscillation brightens the fragment by adding a uniform intensity to the RGB channels.
- White Flash Effect
    - If whiteFlash is true and the explosion is active (gExplodeFactor > 0), the entire fragment is set to a bright white color.
- Explosion Effect
    - **Explosion Colors**
        - Create Explosion Colors with fireCore is Bright yellow, fireOuter is Orange, smoke is Gray to simulate fire and smoke.
    - **Noise Generation**
        - Creates a pseudo-random noise pattern using the texture coordinates, which is later used to introduce pixelation and randomness in the explosion effect.
    - **Color Mixing**
        - The explosion's color is a blend of fireCore and fireOuter, modulated by the noise. As the explosion progresses (gExplodeFactor > 0.7), the smoke color becomes dominant.
    - **Blocky Flame Effect**
        - Introduces a blocky, flame-like pattern by applying a stepped sine function along the texture's vertical axis.
    - **Opacity and Highlights**
        - **Reduces the fragment's opacity as the explosion progresses.**
        - **Adds pixelated highlights when the noise is high (noise > 0.8).**
- Discrete Alpha Transitions
    - Makes alpha transitions more discrete by quantizing the alpha value into four levels.

Geometry Shader (creeper.geom)

- **Quantization for Blocky Effects**
    - Creates discrete movement directions to give the explosion a "blocky" and Minecraft-like appearance.
    - The direction vector dir is quantized into a finite number of possible directions (steps = 8.0).
    - The result favors axis-aligned or major-axis movement to mimic blocky fragments.
- **Emitting a Single Triangle**
    - Compute Explosion Offset by the explodeFactor, giving a smoother outward movement as the explosion progresses.
    - Scale each triangle fragment (scale = 0.8 + randSeed * 0.3) and slightly rotated for a dynamic effect.
- Subdividing a Triangle into smaller fragments for a more detailed explosion. Intermediate points along the triangle edges are computed using linear interpolation (mix). Smaller triangles are generated by pairing edge points, the triangle center, and texture coordinates.

Environment Setup

We utilize the cubemap function and shader in HW3 and replace the image to create a Minecraft-style environment with image directly from Minecraft game by take screenshot inside the game.

Discussion

- Proposer (A) initially suggested a Minecraft-themed idea, focusing on Steve battling the Ender Dragon.
- The Critic (B), however, argued that making the Ender Dragon flap its wings naturally would be quite challenging and expressed uncertainty about where a geometry shader could be applied.

- The Negotiator (C) thought the Minecraft theme was a good choice but suggested replacing the Ender Dragon with a Creeper, as both its movement and explosion effects would be easier to implement.

Work assignment

- Proposer: 鄭睿宏
- Critics: 林辰翰
- Negotiator: 朱驛庭

References

- **Pananora Scene**

  Screenshot from Minecraft
- **Steve Model**

  Blender3D The Perfect Steve Rigged - Download Free 3D model by Blender…
- **Chest Model:**

  Blender3D Minecraft Chest - Download Free 3D model by Blender3D
- **Creeper Model:**

  Vader minecraft Creeper - Download Free 3D model by Vader (@mojtav…

Results

- **YouTube video link**

  Ray Cheng Treasure Hunt
- **Github Link**

  Treasure-Hunt