Interactive Graphics Final Project

Weihao Wang, 1988339

Contents

1 Introduction

Cube Man Adventure is a html interactive game developed based on WebGL technology. User interacts with the game through the mouse and keyboard and completes goal. The goal of the game is that player operates cube man to reach the exit in a fastest time. Moreover, the obstacles and bonus props set up in the game make it more entertaining.

2 Environment and Libraries

Developer uses VSCode as the development environment, meanwhile JavaScript and HTML as the main languages, and references Three.js as main library, and Tween.js Physijs library etc. for better effects.

2.1 WebGL

WebGL (Web Graphics Library) is a JavaScript API for rendering high-performance interactive 3D and 2D graphics. WebGL does so by introducing an API that closely conforms to OpenGL ES 2.0 and this conformance makes it possible for the API to take advantage of hardware graphics acceleration provided by the user's device.

2.2 Three.js

Three.js is a cross-browser JavaScript library and application programming interface (API) used to create and display animated 3D computer graphics in a web browser using WebGL.

2.3 Other Libraries

Tween.js: Tween manages the shift from one property value to another property value for an object. Tween - Abbreviation for "In-between". In animation, tweens are the frames that depict a character's or objects' motion between key frames.

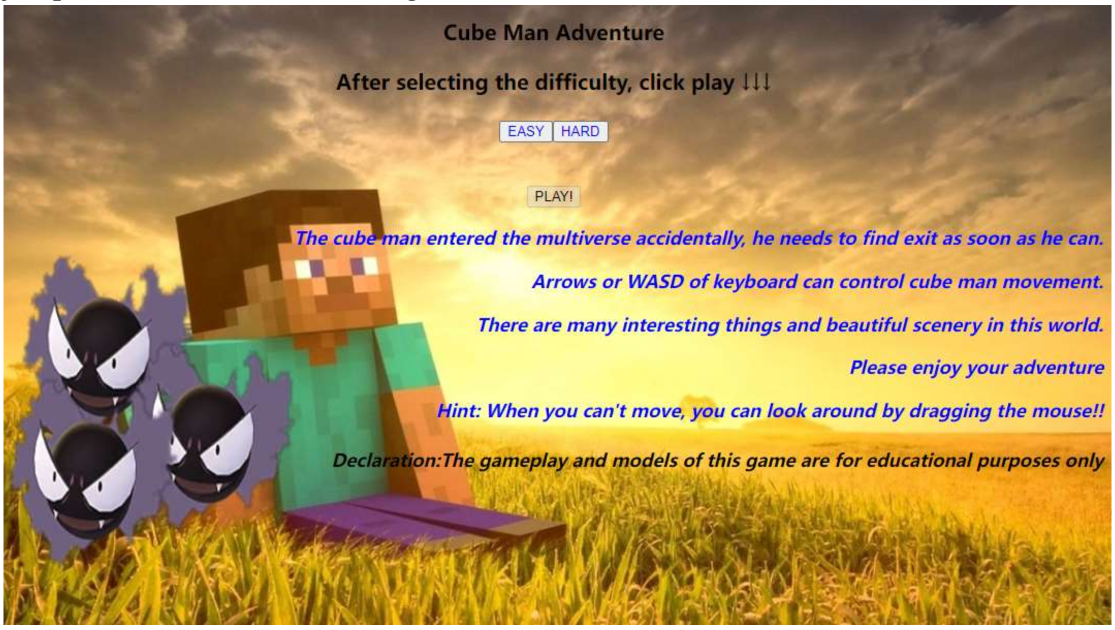
Physijs.js: Physijs.js brings a very easy to use interface to the three.js framework. One of the reasons three.js is so popular is because it is so incredibly easy for graphics newbies to get into 3D programming. Physijs.js takes that philosophy to heart and makes physics simulations just as easy to run.

GLTFLoader.js: A loader for glTF 2.0 resources. glTF (GL Transmission Format) is an open format specification for efficient delivery and loading of 3D content.

OrbitControls.js: Orbit controls allow the camera to orbit around a target.

3 Interactions

Through a button on index.html, players can choose easy or hard mode to play the game by ***easy()*** and ***hard()*** functions. At the end of the game interface (**finsh.html**, **gameover.html**, **gameover2.html**), the player can return to the index.html through the "play again" button. When the player clicks the play button, the ***paly()*** function jumps to main.html to start the game.


Cube Man Adventure

After selecting the difficulty, click play ↓↓↓

The cube man entered the multiverse accidentally, he needs to find exit as soon as he can.

Arrows or WASD of keyboard can control cube man movement.

There are many interesting things and beautiful scenery in this world.

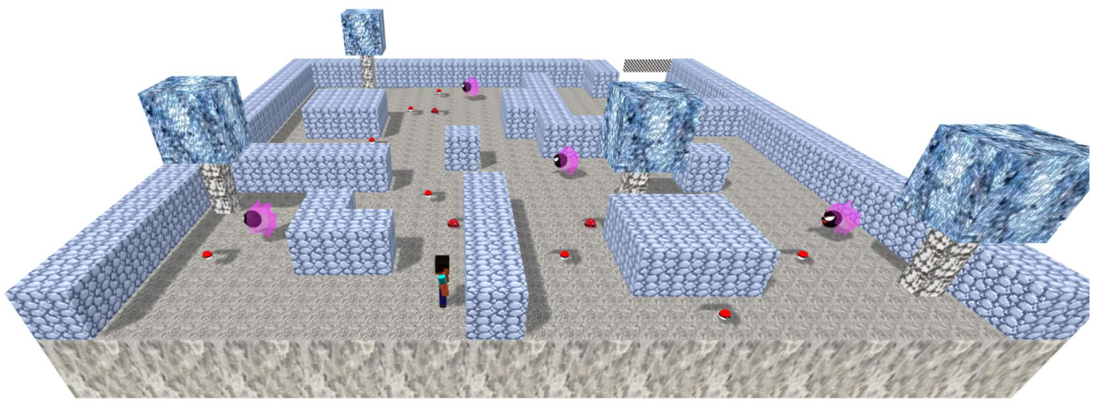
Please enjoy your adventure

Hint: When you can't move, you can look around by dragging the mouse!!

Declaration:The gameplay and models of this game are for educational purposes only

4 Scene

The game shows a world composed of land, walls, blocks, trees. Among them are ghosts and balls, mushrooms. Walls protect players from falling from land, trees and blocks add to the fun of the game.



4.1 Screens

This game contains a total of 5 pages: **index.html**, **main.html**, **finish.html**, **gameover.html**, **gameover2.html**.

index.html: Let the player choose the difficulty and then jump to main.html.

main.html: Display the motion of the model, feedback user input via the following function.

```
function loadingFunction() {
                document.getElementById("game").style.display = "initial";
          }
```

finish.html: Calculate the player's game time, provide a button to return to the main page. Computation time is calculated by the following function:

```
fine = new Date().getTime();
duration = (fine - initial)/1000;
```

Other pages: If cube man encounters ghosts or mushrooms, determine ***death()*** function and execute subsequent functions, jump to **gameover.html**.

4.2 Camera and lights

The camera is initialized on the player's eye position, and the ***lookAt()*** function is used to move the camera position with the cube man's movement, ensuring that the player will not lose the cube man during the game.

```
function animate() {
     delta = clock.getDelta();
```

```
    TWEEN.update();
    scene.simulate();
    if(orbitControls) {
        controls.target.set(steve_box.position.x, steve_box.position.y,
steve_box.position.z);
        controls.update();
    } else{
        camera.position.set(steve_box.position.x, 11, steve_box.position.z+15);
        camera.lookAt(steve_box.position.x, steve_box.position.y,
steve_box.position.z);

    }
    renderer.render(scene, camera);
    requestAnimationFrame(animate);
}
```

When the player reaches the exit or comes into contact with ghosts or mushrooms, the player can freely observe the surroundings by dragging the perspective with the mouse for a limited time. The library **OrbitControls.js** is used here to accomplish this effect.
The sky background of the game uses pure white to create a winter atmosphere, but also to make the game experience smoother.
Two different lights are used, ambient and directional, the light position is placed on the left side of the canvas, and all models in the game have their own shadows.

4.3 Textures

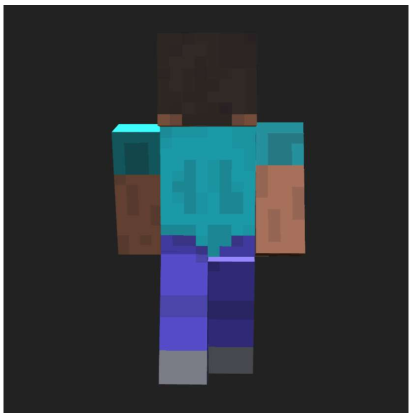
The texture pictures of the land, walls, blocks, leaves, and tree trunks are all from the Internet, and the leaves used bump texture to increase the authenticity. The advantage of using map textures is that I can achieve more realistic results in a short time. If the number of teams is increasing, I will consider using code to generate textures

5 Models

To load 3D models I used **GLTFLoader.js** in the initialization, gltf models are all downloaded from the Internet.

```
function load_characters() {

    load_steve();
    load_ghost();
    load_ball();
    load_mushroom();

}
```

5.1 Cube man



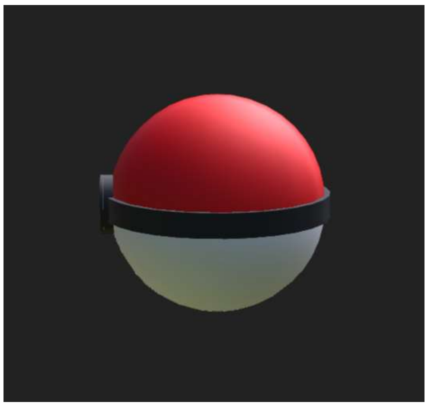
Cube man is a hierarchical model, and its components are found to be used in animation, this model is the character that the player can control Its components are as follows:

```
const steve_parts = {

    Root: "_12",
    Head: "head_10",
    Torso: "body_5",
    UpperArmR: "arm1_9",
    UpperArmL: "arm0_7",
    UpperLegR: "leg1_3",
    UpperLegL: "leg0_1",


};
```

5.2 Ghost

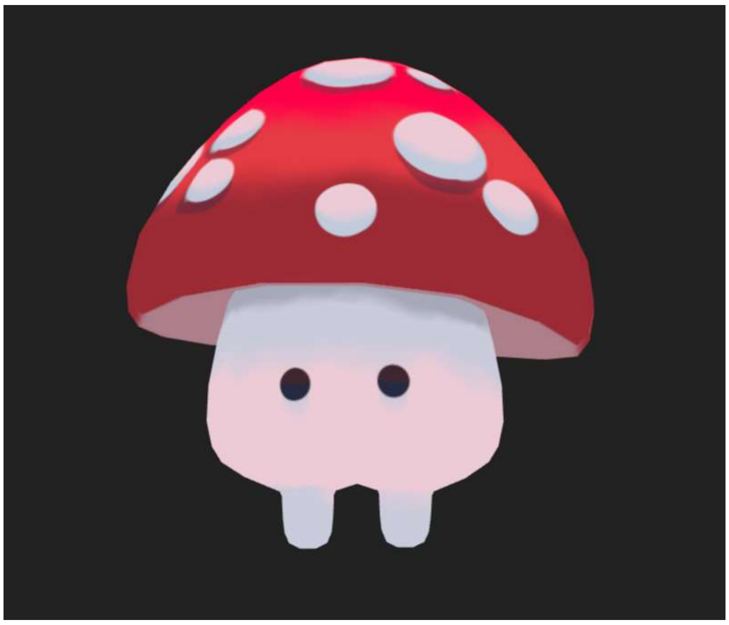
Ghosts are not hierarchical models, their generation is replicated through arrays, and there will be different numbers of ghosts corresponding to different difficulties.

5.3 Balls



Ghosts are not hierarchical models, their generation is replicated through arrays.

5.4 Mushroom


Mushrooms are not hierarchical models, their generation is replicated through arrays, and there will be different numbers of mushrooms corresponding to different difficulties.

6 Animations

To generate smooth animations the **tween.js** is used in this project. The most important feature of this library is the interpolation function: tween.js will perform the interpolation between values in a linear manner by default, so the change will be directly proportional to the elapsed time

6.1 Cube man animation

Run

The tweens library makes the action smooth by interpolation, so in order to realize the cube man running animation, I designed two animation chains, and completed a smooth action by setting two target positions. In more detail, I set two target positions(front, back), by using tweens **easing()** function to implement an action, and then use **yoyo()** and **repeat()** functions to repeat the action when running input is detected.

```
tween_run_middle = new TWEEN.Tween(start)
                        .to(middle, time)
                        .easing(TWEEN.Easing.Linear.None)
                        .onUpdate(function () {

                            if(!collided || collision_direction != direction ){
                                if(direction == 'right' || direction == 'left'){
```

```
                camera.position.x += step;
                camera.updateProjectionMatrix();
                steve_box.position.x += step;
                collided = false;
            }else if (direction == 'up' || direction == 'down'){
                camera.position.z += step;
                camera.updateProjectionMatrix();
                steve_box.position.z += step;
                collided = false;
            }
        }
```

Rotate
This animation just rotates the cube man smoothly when the movement direction changes.
Perform a check to determine the correct rotation angle. The implemented algorithm uses various auxiliary variables to rotate in the correct direction.
This feature also restores the rotation of the collision box of cube man in the case of rotational instability due to collisions. Change the rotation of the model by calculating the transformation of the front and rear angles, which can correspond to the direction command input by the player.

Wave
When the player reaches the exit point, the ***wave()*** function will be triggered. Like above animations logic, use **tween.js** to complete the smooth animation after setting two target angles, and then cycle the animation through the ***yoyo()*** and ***repeat()*** functions until screens jumps to the next one page.

Death
When the player meet with mushrooms or ghosts, the ***death()*** function will be triggered, similar to the ***wave()*** function,it changes the movement angle of each part of the hierarchical model to simulate the animation of death.

6.2 ghost animation

Clockwise movement by continuously changing the forward angle, also use the animation chain to complete the angle switch.

6.3 Balls animation

Changing the up and down position to simulate bouncing animation. Similarly, using ***yoyo()*** and ***repeat()*** to infinitely loop this animation.

7 Physics and controls

7.1 Collisions detected

For the collisions the **Physijs** library has been used, each model has a built-in Physijs BoxMesh, and the collision detection is actually to detect the contact of the Physijs BoxMesh. After a collision, there will be actual effect feedback to simulate physical effects.

```
steve_box.addEventListener( 'collision', function( other_object, relative_velocity,
relative_rotation, contact_normal )
```

I placed Physijs BoxMesh on cube man, walls, blocks, land, ghosts, balls, mushrooms. In addition to preventing the cube man from moving into the wall, collision detection can also trigger a series of events to complete the logic design of the game.

7.2 inputs

Through binding of keyboard input and animation, the movement of the cube man can be controlled by the keyboard. Therefore, user can interact with the game using both the directional arrows and the WASD keys.

8 Notes

1. This game is tested with Google Chrome (105.0.5195.127) and Microsoft Edge (105.0.1343.42), the effect of other browsers is unknown.
2. There are still many problems in the game, the cube man's model often wears the mold that makes contact with ghosts and other props uncomfortable, probably related to the location of the BoxMesh.
3. If the cube man hits the edge of the wall at a fast speed, his z-axis may be tilted. The reason is the physical effect provided by the Physijs library which I think change some specific parameters can slow down this effect.

9 References

Three.js: https://github.com/mrdoob/three.js/
Tween.js: https://github.com/tweenjs/tween.js/

Physi.js: https://chandlerprall.github.io/Physijs/
GLTFLoader.js: https://github.com/johh/three-gltf-loader
Cube man: https://sketchfab.com/3d-models/steve-27fab6181ffa4fae9c69eb097630eaf3
Ghost: https://sketchfab.com/3d-models/gastly-e3edf38780594b2d9a1675b9f20b0da8
Balls: https://sketchfab.com/3d-models/pokemon-poke-ball-9f63803d93714eadaa869c23cee0581d
Mushroom: https://sketchfab.com/3d-models/mushroom-friend-de9cc47fae19421a84db8192dc3cd92e