Interactive Graphics-Final Project

Authors:
Peng Kai - 1947951
Tian Jiexin - 1961801
Sheng Chengcheng - 1933736

August 2021

SAPIENZA
UNIVERSITÀ DI ROMA

Professor: Marco Schaerf

Contents

1 Game User Guide

1.1 Game Introduction

The evil pizza king summons evil dragons to destroy the world. So our hero embarks on a journey!
The hero originally owned 100 grid bloods.When the hero approaches the monster, the monster will automatically attack the hero and each attack will reduce the blood volume of the hero by the ten percents. The hero can use two attack skills to kill monsters. Monsters killed will immediately rebirth in the summoning area of evil pizza king.
If the HP of the hero is reduced to 0, the hero will die, the game will end and the final score of the game will be displayed.

1.2 User Interaction

1.2.1 Game Interaction

The lowercase [w, a, s, d] control the hero to move forward, left, right, and backward respectively.
The lowercase [j] and [k] are the two attack skills of the hero, press them to destroy the monster.

1.2.2 Camera Interaction

$[\uparrow, \downarrow, \leftarrow, \rightarrow]$ can adjust the position of the camera.The numbers [4, 5, 6, 8] can adjust the perspective of the camera, but we do not recommend you to use them.

2 Basic Libraries

The libraries we used for this project:

- WebGL
- Three.js
- TWEEN.js

2.1 WebGL

WebGL is used to create website pages with complex 3D structures and can be used to design 3D web games.

2.2 Three.js

Three.js is a WebGL third-party library written in JavaScript. It provides a lot of 3D display functions. There are lots of features we used in this project:

- **GLTF Loader** : It is used for efficient delivery and loading of 3D content.
- **Scene**: Scenes are used to set up what and where is to be rendered by three.js. This is where we can place objects, lights and cameras.
- **PerspectiveCameras**: This projection mode is designed to mimic the way the human eye sees. It is the most common projection mode used for rendering a 3D scene.
- **WebGLRenderer**: WebGL renderer displays the beautifully crafted scenes using WebGL.
- **AmbientLight**: This light globally illuminates all objects in the scene equally.
- **DirectionalLight**: A light that gets emitted in a specific direction.

2.3 TWEEN.js

A tween is a concept that allows to change the values of the properties of an object in a smooth way. We can use it to smoothly move the components of the model to realize action animation.

There are some functions we used to realize animations:

- **TWEEN.Group()** : It allows users to manage different animations separately.
- **TWEEN.Tween()** : Create a tween for attributes of an object.
- **TWEEN.Tween().to()** : Tell the tween we want to animate the properties over some time interval.
- **TWEEN.Tween().repeat()** : Make a tween animation to repeat.
- **TWEEN.Tween().yoyo()** : The behaviour of the tween will be like a yoyo, i.e. it will bounce to and from the start and end values.
- **TWEEN.Tween().onRepeat()** : Executed a function whenever a tween has just finished one repetition and will begin another.
- **TWEEN.Tween().start()** : Start a tween animation.
- **TWEEN.update()** : Run animations as smoothly as possible.

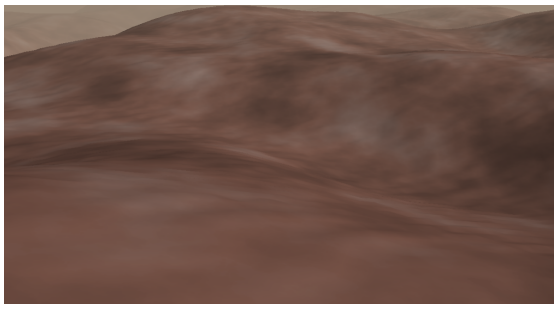

Figure 3.1: Partial map of terrain map

3 Environment

3.1 Terrain Generation

- **Step 1: Setup three.js**: In this step we are setting up a basic three.js application and main loop. The end result is a blank screen.
- **Step 2: Create terrain**: This step is all about creating a terrain mesh, giving it a material, and adding a light.
- **Step 3: Load heightmap**: This step is all about loading a heightmap image and using the pixel values to set the height of the terrain.
- **Step 4: Texture terrain**: This step adds a simple texture to the terrain.

3.2 Scene

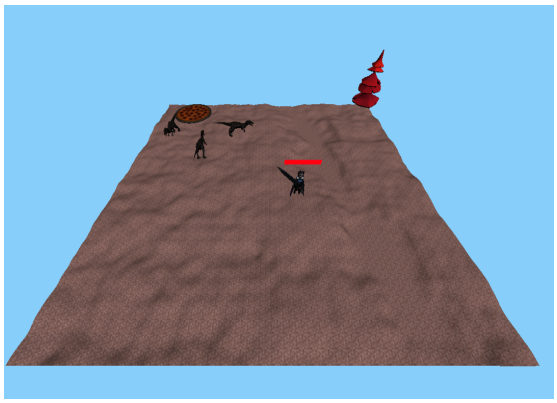

Figure 3.2: Complete scene of the game

The scene shown in Figure 3.2 is a Three.js object which is used to build our game scene by adding terrain, lights and 3D models.

The different 3D models, Hero and Monster, are from SketchFab and the animation is implemented manually through TWEEN.js.
We initialized a suitable fixed perspective camera view to make the scene better viewed, while they can be changed manually.

3.3 Lights

We create a direct light source with the THREE.js package. Then we set the position of the light source, the shadow, and the camera of the shadow, and finally add the light to the scene, which can be used to illuminate the whole scene.

4 3D Models

All the 3D models we used were imported from Sketchfab. TWEEN.js is then used to control the parts in the models to achieve the desired animation.

4.1 Hero

This is a GLTF format 3d model, which is the role of the player control.
We have 4 different animations:

- Move;
- Run;
- Skill1;
- Skill2.

All the animations are implemented by using TWEEN.js. This 3D model is a hierarchical model, we implement those animations by controlling different parts of this model.

4.2 Monster

This is also a GLTF format hierarchical 3d model, which is the opponent the player will face.
We have 3 different animations:

- Move;
- Walk;
- Attack;

The move animation controls monster to move with random direction and random distance, it is always activated when game start, walk animation will be activated during move animation. When the monster is close enough to the hero, it has 5% probability to decide whether active attack animation.

4.3 Tree

The 3D tree model. It's only a scene decoration. And it is displayed in the upper right corner of the map.

4.4 Pizza

The 3D pizza model. It's only a scene decoration at the location of the monsters rebirth.

5 Game Logic

5.1 Controls

The game logic is composed mainly by animation start control and hero blood loss and monster death control. We control it mainly by the change of flag's value.

- **Animations Controls**: There are mainly animation start control, hero movement direction control. We divide the animation into different groups, and then put the Tween.update() into the loop() function of three.js to control flags by different keys of keyboard to control the update of the animation or not.
- **Other Controls**: Set the harm and death flags to determine if the hero is under attack and perform the appropriate blood reduction or game over

action.
The control of the attack determination is written in the keyboard press listener event. The different attack determination functions are called by different keys's press on the keyboard.

5.2 Attacks

(a) Hero skill-1 attack range (b) Hero skill-2 attack range

Figure 5.1: Attack range of hero's skills

- **Hero Skill-1 Attack**: The skill-1 is the hero's common attack skill, the hero simply waved her sword by hands. Considering that this skill can only cause damage to monsters within the attack range in front of the hero, so we put the center of the attack range at a distance of 20 units from the hero's front, The attack radius is 200 units, if hero launch this skill and the distance of the attack range center and monster less than attack radius, monster will be killed.
- **Hero Skill-2 Attack**: This skill is a range damage skill, the hero spins with her sword. The attack range center is the position of the hero, attack radius is 200 units. All monsters in this attack range will be killed.
- **Monster Attack**: When the distance of the hero and monster less than 200 units, it has a 5% random probability to launch monster attack.

5.3 Hero's Death

The Hero has 100 HP(Hit Point), it is displayed by a simple red 3D cube model on hero's head. When the hero be attacked by monsters, hero's HP will be reduced by 10% at each attack. And the cube width will be reduced by 10% accordingly. When HP is 0, hero will die and will be removed from scene, then web-page will show "Game Over!" and your grade, which is the number of monsters the hero killed.

5.4 Monster's Death and Rebirth

When monsters are killed by hero's skill, monsters' position will be refreshed at location (300, 300, 0) and grade will add one.

6 Animations

6.1 Monster Animation

The Monster have three different animations which are Move, Walk and Attack.

- **Move**: We use tween.js to animate the monster movement and set the direction and distance of the monster movement randomly in real time.
    1) **Set movement speed**;
    2) **Implement the monster pan animation**: The pan destination is the monster's self position.
    3) **Set random function**: Randomly generate the direction and distance of monster movement.
    4) **Update monster position**: Update the monster position according to the newly generated direction and distance.
- **Walk**: The dragon walking animation is implemented by function LongWalk(). In order to realize the animation of the dragon walk, we need to control the four components of the dragon model through the code, which are the upper right leg "Bone040_Armature", the lower right leg "Bone044_Armature", the upper left leg "Bone039_Armature" and the lower left leg "Bone043_Armature". Then use the .rotation(), .to() and .delay() functions in TWEEN.js. Finally, we use function repeat(Infinity).yoyo() to repeat the component rotation infinitely to realize the dragon walking animation.
- **Attack**: Monster Attack animation implementation. The animation process and principle of LongAttack are the same, except that the components controlling the model are different from the dragon walking. Dragon attack is to control the upper jaw "Bone006_Armature" and lower jaw "Bone010_Armature" of the model component to realize the dragon bite action to achieve the dragon attack animation effect.

6.2 Hero Animation

The Hero have four different animations which are Move, Walk, Skill-1 and Skill-2.

- **Move**: The hero's movement animation is similar to the monster's movement animation, but the direction of movement is controlled by the keyboard. There is no need to generate random directions and distances.

1) **Set movement speed**;
2) **Implement the hero pan animation**: The pan destination is the monster's self position.
3) **Determine the direction of the hero move**
4) **Update hero position**: Update the monster position according to the newly generated direction and distance.

- **Walk**: The animation of the hero walking is implemented similar to the monster. Only the components controlled in the animation are different.
- **Attack**:
  1) **Skill 1**: Hero Skill 1 attack animation is implemented by function HeroSkill1(). It needs to control the left hand "Hand_L_FIN" of the hero model component.
  2) **Skill 2**: Hero skill 2 attack animation is implemented by function HeroAttack(). It needs to control the left hand "Hand_L_FIN", left upper arm "UpperArm_L_FIN", left lower arm "LowerArm_L_FIN", right upper arm "UpperArm_R_FIN" and right lower arm "LowerArm_R_FIN" of the hero model component.

7 Conclusion

In this project, in order to run more complex models, we chose to use three.js to achieve this goal, and we use Tween.js to achieve better animations. But this project has many shortcomings, we still need to optimize the program, and the hardware requirement is not friendly. And the UI design of the whole game and the game architecture still need to be improved.

8 References

[1]Javascript: `https://www.javascript.com`
[2]HTML: `https://www.w3.org/html`
[3]CSS: `https://www.w3.org/Style/CSS/Overview.en.html`
[4]CSS: `https://sketchfab.com/feed`
[5]ThreeJS: `https://threejs.org`
[6]TweenJS: `https://www.createjs.com/tweenjs`