Interactive Graphics

Final Project

Matteo Marinacci 1754769

Academic year 2021/2022

Contents

1 Used Technologies

The most widely used technology in the project is JavaScript and its libraries, the other technologies used were of support for the development of the project.

1.1 HTML

HTML has been used for minor aspects of the project, in particular has been used to handle a counter for the kills achieved by an user, and to display hint, to pick a weapon, when the user is near it.

1.2 CSS

CSS has been used to handle the style of HTML tags.

1.3 JavaScript

JavaScript has been used to perform all actions and interactions from the user with the project.

1.3.1 Threejs

Threejs is the main JavaScript library used, it manages **cameras**, **scenas**, **renderer** and **objects**. In the project both **Perspective** and **Orthographic** cameras have been used, they were handled by two different scenes. The reason why two cameras were used is to display the main scene with the perspective camera and the UID with the orthographic camera.

1.3.2 Tween

Tween is a JavaScript library used to realize all animations of the objects. Tween has been used to animate the movement and the attack of the character, the movement of the enemy, the weapons and to produce some movement of the camera.

1.3.3 Ammo

Ammo is a direct port of the Bullet physics engine to JavaScript, it manages all the physics of the project. Ammo has been used to avoid collision with walls and to insert gravity in the project.

1.3.4 Loaders

Loaders are used to load into the project GLTF models, fonts and textures.

- **GLTFLoader** has been used to load all gltf files downloaded from **Sketchfab** (*grim_reaper*, *tutorial_room*, *eye*, *scythe*, *sword*, *broken_sword*, *granit_rock_wall*, *portal*, *rock*).
- **FontLoader** has been used to load the font file in JSON format (*gypsy-curse.json*).
- **TextureLoader** has been used to load textures files (*forest*).

1.4 Blender

Blender is a modelling software that has been used to build the first room of the project.

2 Character

The principle class of the project is the **Character** class. This class is used to handle the *grim_reaper* model (**IMPORTANT NOTICE**, the model contains a predefined animation that it is not used inside the project), at the beginning it is possible to find all used variables and the **constructor** of the class in which there are the definitions of all components of the model (more than 100 components) and their relative positions and rotations. After that it is possible to find the **update** function in which there is the initialization of the model as a physical object in order to make gravity work, to allow the movement of the model according to a given velocity, to prevent the model from crossing walls and floor, and to give to the model the ability to climb steps of the stairs when it collides with them. After that, always inside the update function, it is possible to find all operations needed to produce the animations:

- **tweenStill:** it is used to produce the fluctuation of the character after 4 seconds without any interaction (movement or action), the animations involves the movement of the skirt, of the belts, of the wings and of the model along the y axis. At the **stop** of this animation it is invoked the *tweenBackGround* animation.

- **tweenBackGround:** it is used to reset all values changed by the *tweenStill* animation.

- **tweenFront/tweenBack/tweenLeft/tweenRight:** are used to produce the animations of the movement according to the selected direction, they involve the movement of the skirt, of the belts and of the wings. At the **stop** of any of these animations it is invoked the *tweenEndMov* animation.

- **tweenEndMov:** it is used to reset all values changed by any of *tweenFront/tweenBack/tweenLeft/tweenRight* animations.

- **tweenActionRight:** it is used to produce the animation of the right arm when attacking or picking the weapon it involves the movement of the arm and the sleeve and the movement of the hand and the fingers if the weapon it is not picked otherwise involves also the movement of

the weapon. The velocity of such animation is determined by the chosen weapon. This animation will reset the initial values itself thanks to the using of **repeat(1)** and **yoyo(true)** which together states that after one animation forward and backward the animation it is completed.

- **tweenLeftHand:** it is used to permanently animate the left arm and involves the movement of the arm, of the sleeve and of the fingers.
- **tweenExitMenu:** it is used to exit from menu when an user want to start the game and such animation involves the movement of the camera.
- **tweenTraversePortal:** it is used to make the model traverse the portal such animation will involve the movement of the camera and of the model. At the **completion** of this animation it is invoked the *tweenZoomIn* animation.
- **tweenZoomIn:** this animation it is used to zoom on the portal with the camera. At the **completion** of this animation it is invoked the *tweenZoomOut* animation.
- **tweenZoomOut:** this animation it is used to zoom out of the portal with the camera. At the **completion** of this animation it is invoked the *tweenZoomOut* animation.
- **tweenExitPortal:** it is used to move the character outside the portal. At the **completion** of this animation it is invoked the *tweenCamBackAndPortalDisappear* animation.
- **tweenCamBackAndPortalDisappear:** it is used to make the portal disappear and to reset the position of the camera to be behind the character.
- **tweenDeath:** it is used to make the character disappear when its health points run out.

To make the game in third person the camera it has been added to the model in order to move the character and the camera together.
Traversing the whole model it has been made possible to the character to **cast and receive shadows** from other models.

3 Tutorial Room

The second class of the project is the **TutorialRoom** class. This class is used to handle the *tutorial_room* model which was downloaded from *Sketchfab* and edited with *Blender* by me in order to obtain the desired room. At the beginning it is possible to find all used variables and the **constructor** of the class in which there are the definitions of all necessary components of the model and their relative dimensions that will be used by the physics engine to avoid collisions. After that it is possible to find the **update** function in which there is the initialization of the components of the model as physical objects in order to allow the *character* to interact with them without letting him go through them. After that it is possible to find the definitions of ten **PointLight** each with a different position, color, intensity, distance in which the light keep its effect and decay and a **DirectionalLight**, covering the whole room, used to **cast shadows**.
All components of the model are able to **receive the shadows** but only the altar, a candle and the portcullis (exit gate) are able to **cast shadows**.

4 Weapons

The third class of the project is the **Weapon** class. This class is used to handle the *scythe* and *sword* models. They do not have any physics engine and for this reason the character can pass through them, the main reason of this choice is because I would let the user the feeling that the character could be hit only by monster and not by weapons otherwise it would be easy to put physics also on them. At the beginning it is possible to find all used variables and the **constructore** in which there are the definitions of the weapons and their relative parameters (rotations and positions). After that we can find the **update** function which contains methods to **cast** and **receive shadows** and the tween that allows the weapons to fluctuate changing their position along y axis. When a player choose a weapon the following actions will be executed:

- Both weapons will go back to the original position with **tweenBackOriginalPos**
- Both weapons will disappear and the gate of the room will opens with **tweenFinalAnimation**
- Chosen weapon will be anchored to the character using an **Object3D** as pivot, and it start a small movement thanks to **tweenMoveWeapon** to give the feeling of a floating magical weapon

5 Outside World

The fourth class of the project is the **Outside** class. This class is used to handle *rock*, *granit_rock_wall* and *broken_sword* models, a **BoxBufferGeometry()** to which is associated a texture with all necessary types of **map** to make it more realistic and two lights, one which is an **AmbientLight** and one which is a **DirectionalLight** targeting the character to produce its shadow (this light is inside the Character class).
The only model to which is associated the physics engine is the *granit_rock_wall* to make the border of the world and to the *BoxBufferGeometry* to prevent the character from falling.
The *broken_sword* and *rock* models are randomly positioned around the world to avoid an empty world and to make it more realistic.
Another important component of the whole project is the **Fog** (is inside the Character class) that contributes to give the project a good setting.

6 Enemies

The fifth class of the project is the **Enemy** class. This class is used to handle the **eye** model. The aim of this class is to handle 15 enemies by spawning them using **tweenSpawnEyeX** for each eye of each enemy, when an enemy fully appears it is oriented to the position of the character and will follow him.
When an enemy is hit by or hit the player will disappear using **tweenRemoveEyeX** for each eye, after that another enemy will re-spawn in a different position to substitute it. If the enemy is hit by the player will increase the count of kills, the hit by the player is determined by the intersection between the box of the enemy and the box of the weapon, if the enemy hit the player will decrease the life bar of the character. Moreover each enemy is provided with an animation that rotate each eye in a different direction with random values in order to differentiate a little bit each enemy, this animation is handled by **tweenAnimate**.

7 UID

The sixth class of the project is the **Uid** class. This class is used to handle:

- the initial menu that will show commands of the game as a tutorial (camera, action/attack, movement) and the **Start Game** text which is clickable thanks the using of the **Raycaster** that assist *raycasting* that is used for mouse picking

- the life bars (actual life in green and max/lost life in red) and the bar containing the number of enemies defeated, these three bars will appear when the character traverse the portal to give the player the feeling that the battle is about to begin. The life bars will decrease or increase according to how much enemy the player will defeat

- the game over menu that will display the text *Game Over* and what is the score of the player, at this point I had to make a choice, inserting a *button* to restart the game or not, at the end I chosen to not put it in order to let the user manually refresh the page to give to him the feeling that what was done should be repeated again and again

8 Controls

Controls of the game are really simple and intuitive:

- the user is free to move the character in the world with the keyboard arrow keys
- with the left click of the mouse it is possible to rotate the view and the character to move it in different directions
- with the right click of the mouse the player can pick weapons or make an attack