Final Project – Interactive Graphics 2022/2023

Andrea Gervasio – 1883259

1. Libraries used

I used **Three.js** as 3D library and **Tween.js** to implement the animations in the scene.
I used **OBJLoader** and **MTLLoader** to load the 3D object models and their materials. I also used **GLTFLoader** to load GLTF models.
To unlock the camera and let the user control it I used **OrbitControl**.

2. Assets

I created some assets from scratch using MagicaVoxel and I imported some object models from the internet.

2.1 Objects

- The stalagmites GLTF files are taken from Sketchfab
- The star OBJ files are taken from turbosquid
- The chainsaw model and the head of the player are made from scratch using MagicaVoxel

2.2 Textures

- The texture of the bridge rest arm are created using the stone brick texture from Minecraft
- The textures of the bridge are taken from artstation
- The textures of the lava are taken from texturecan

2.3 CSS

- Css file of the buttons in the main menu, pause menu and game over screen are taken from freefrontend

3. Gameplay

The player is spawned on a bridge above a lake of lava. They need to dodge the chainsaws which come their way to avoid falling in the pool of lava. While doing this the player can collect stars to increment their score and hearts to restore lives.

The game takes place on a bridge with three lanes, and the user can use the “a” and “d” buttons on the keyboard or the right and left arrows to switch lane. The player can also jump using the spacebar of the keyboard to avoid the chainsaws coming their way.
When the player is hit by a chainsaw, the life counter is decreased by one, and the player has a period of invincibility. This is communicated to the user by making the character's model turn invisible and then blink when the invincibility period is about to end. In the code this is obtained by changing the *visible* and *opacity* attributes of the model using a timeout function.

The user can also pause the game by pressing the ESC button on the keyboard. This is obtained by using the Clock class of Three.js and a method of the TWEEN.Tween class in order to get all the active tweens.
When the ESC button is pressed, the clock and all the active tweens are stopped until the game is resumed.

The user can also move the camera during the game. There are three predefined configurations, accessible by pressing 1, 2 and 3 on the keyboard, but the user can also move the camera freely by pressing 4, which unlocks the camera, and then dragging the mouse while keeping the left button pressed to rotate it and the right button to translate it.

4. Hierarchical models

There are two hierarchical models used in the code: one for the character and one for the bridge on which the game takes place.

4.1 Character's model

The model of the character is built using THREE.Mesh and THREE.Object3D. I used the Object3D for the parts of the model which don't have geometries and are not used during the animations, while I used Mesh for the rest.
The Objects are the waist, the neck, the roots of the legs and the feet, while all the rest is built using Meshes, as shown in Figure 1.

4.2 Bridge's model

The bridge is also made of Objects and Meshes, with the help of BoxGeometry and ExtrudeGeometry.
The bridge is composed of four parts:

- The main is a BoxGeometry inserted in a mesh
- The arcs are children of the main mesh, and they are made exploiting ExtrudeGeometry by defining Bezier curves
- The arms are also made using ExtrudeGeometry
- The objects that spawn during the game (chainsaws, hearts and stars) are imported from OBJ files and inserted into THREE.Object3D

Chainsaws, hearts and stars OBJ files have their own materials, and they are used in the *loader.js* file, each with its respective class. They are scaled and positioned in the scene using the **Box3** class.
The stalagmites in the cave are defined in a GLTF file and there is a specific class in the code to load them.

5. Textures

For the BoxGeometry of the bridge and the arms I used as color maps the .jpgs file in the *assets/bridge* subdirectory. The normals of these images were obtained using this normal maps maker online.

The lava pool is a plane to which different textures are applied. In particular, they are all stored in the *assets/lava* subdirectory. There are a color map, a normal map, an ambient occlusion map, an emissive map, a bump map and a roughness map.

The stalagmites have a color map and a normal map, both stored in the *assets/stalagmites/textures* subdirectory.

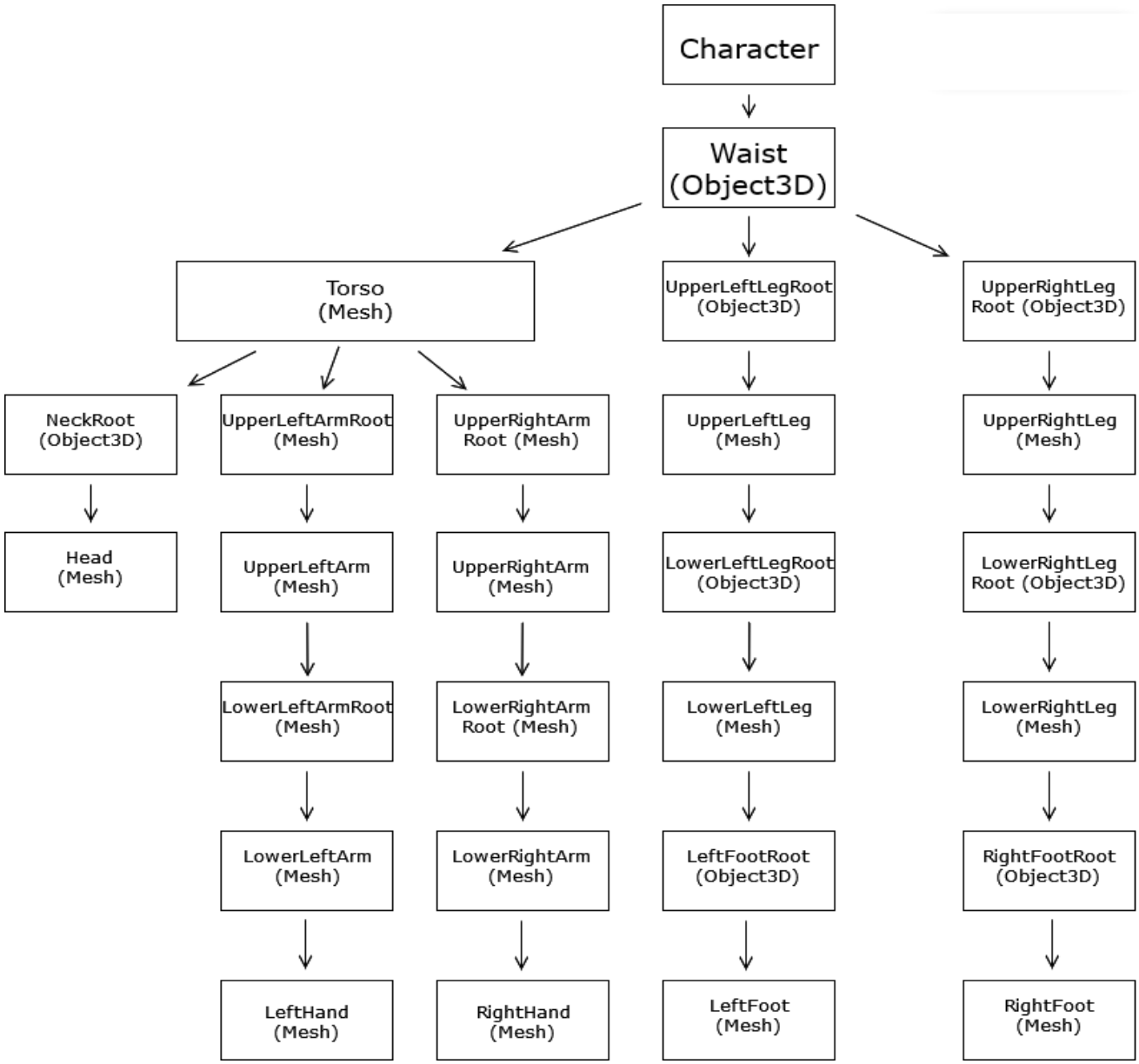

Figure 1: The character's hierarchical model

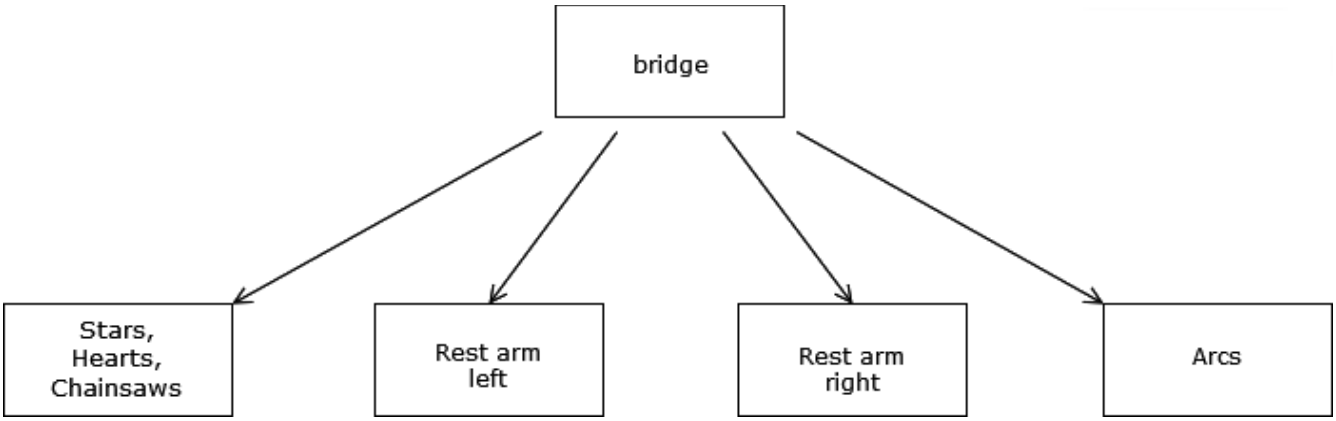

Figure 2: The bridge's hierarchical model

6. Lights

Five lights are used in the code:

- An **ambient** light and a **directional** light that always follow the player
- An **hemisphere** light to have darkness on top and to simulate the light of the lava
- Two **rectangular** light, one is used to better illuminate the lava under the bridge, the other is used at the end of the bridge to simulate an exit (which the player can never reach, though)

7. Animations

All the animations are based on keyframes.

7.1 Jump

The jump animation uses 7 keyframes. The first one is the standard position, followed by the charge keyframe, in which the character has the legs bent and the arms in front of them.
The third keyframe is the intermediate value of the jump, while the character is still going upwards.
The fourth is the peak of the curve of the jump, which follows a cubic polynomial curve.
The fifth keyframe is the one corresponding to the third, but in the falling part of the jump.

The sixth keyframe is the landing keyframe, similar to the second, while the seventh is the standard position again, from which the character can start moving again.

7.2 Run

The running animation is composed of three different tweens chained together. The first one moves the torso of the character forward and the arms backwards in a linear way. The second and the third are used to move the legs in a sinusoidal way to simulate the movements of a person running. These last two tweens are chained to create a loop so that the character keeps running smoothly until the player presses a key to move.
There is also a rotation of the torso during the run (which is called *tumbling* in the code) but it is separate from the three tweens mentioned above due to some wrong results during the dash animation.

7.3 Dash

The dash animation is performed when the character changes lane. They are very simple tweens composed of only two keyframes, which rotate the character's torso.

7.4 Fall

When the player loses all their hearts, the character will fall in the lake of lava.
The falling animation is composed of four keyframes: the first one is the standard position, the second and third are used to move various parts of the body, and the fourth changes the height of the model to simulate gravity.
In the second and third tweens the character brings their arms up and waves them, while also moving their legs.

7.5 Objects

The objects on the bridge (chainsaws, hearts and stars) are not moved using Tween.js, due to some problems arisen during the implementation of the pause function, but they are instead moved with a clock defined from Three.js and a

constant speed. The multiplication of the time between the previous update and the current one and the speed will result in the new position for the object.

The chainsaws use a simple tween to rotate on their y axis. This was done by exploiting the fact that the model is made from scratch, and it is in fact symmetric about all axes, so there is no need to define different keyframes and move the object between them.

8. Collisions

The collisions are implemented using the **Box3** class.
Every object on the bridge has its own type. When a collision between the player and an object happens, the code checks the type of the object and behaves differently.
The downside of this approach is that this must be done for every lane of the bridge, so there are some parts of the code which are basically the same, but they need to be repeated in order for them to work properly.