Interactive Graphics - Final project

Nicolas Benko 1756287

1 Libraries

I used **THREE.js** as 3D library and **TWEEN.js** to perform the animations.
I used **OBJLOADER** and **MTLLOADER** to load the 3D OBJ models and their materials, and **GLTFLOADER** to load GLTF models.
I used also **OrbitControls** to unlock the camera and let the user move it as he prefers.
The code contains some snippets from the threejs.org and from threejsfundamentals

2 Assets

I created some normal maps using Normal map online.
I used Magica voxel to create the head of my character.

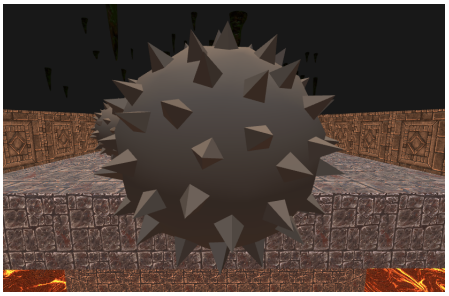
- Objects and gltf
  - Stalagmites GLTF files are taken from Sketchfab
  - Spike-ball Obj files are taken from Opengameart
  - Star OBJ files are taken from turbosquid
  - ninjaHead.obj, ninjaHead.mtl, ninjaHead.png are made with Magica voxel
- Textures
  - Textures of the bridge rest arm are take from awesomeTextures
  - Textures of the bridge come from artstation
  - Textures of the lava come from texturecan
- CSS
  - Css file of the buttons of pause and game over menu are taken from freefrontend

3 Gameplay description

The game take place over a bridge placed in a lake of lava, there is no way to survive!
But before the end you can show your abilities and collecting more star as you can.

The stars spawns over three lanes, but you have to pay attention to the spike-balls.
You start with three lifes.
If you get hit by a spike-ball you lose one life and get one step closer to your end, but there is no problem, you can collect an heart in order to restore life.
After get hit by a spike-ball you acquire a momentary invulnerability, during this time you are not able to collect hearts or stars, but neither spike-balls can hit you.

Collect stars, avoid spike-balls and try to get extra hearts.



As follows the list of commands:

- Press ESC to pause the game.
- Press 1,2 or 3 to change the camera position.
- Press 4 to lock/unlock the camera. Once unlocked u can control it by pressing the left click to and moving it to rotate and right click to move the position.
- Press SPACE to jump
- Press A or ← to perform a dash left.
- Press D or → to perform dash right.

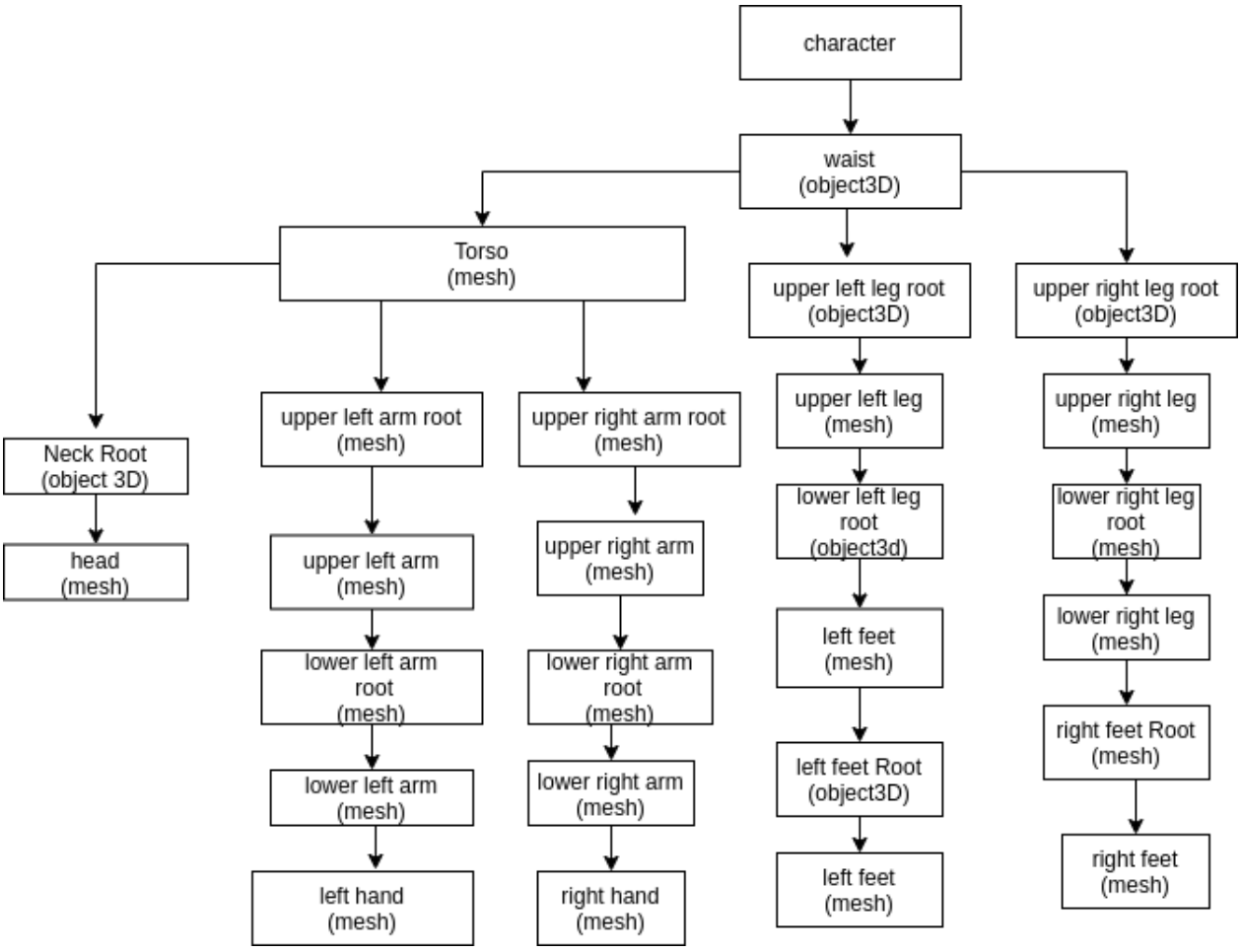
4 Technical description

4.1 Hierarchical models

There are two main hierarchical models one for the character and one for the bridge.

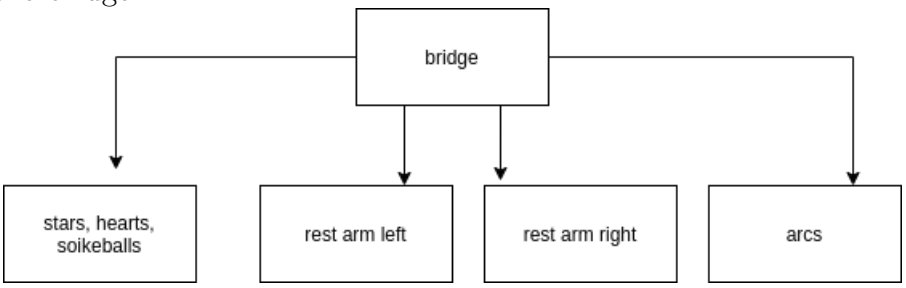
4.1.1 Character

I implemented the model of the character exploiting THREE.Mesh and THREE.Object3D. For the head i used an THREE.Object3D in which i inserted the OBJ file which i have build. For the hands,torso,upper/lower arms, upper/ lower legs, and feet i used THREE.Mesh. I also used THREE.Mesh for the roots in order to insert spherical geometries in them,so as to achieve a more realistic rotation of the limbs. In the root of the upper leg and the one of the feet i used an THREE.Object3D, since they have no geometries.



4.1.2 Bridge

As for the character i used THREE.Mesh and THREE.Object3D, and for the geometries BoxGeometry and ExtrudeGeometry.
The bridge is made by 4 parts, the main is a BoxGeometry inserted in a mesh. The arcs are child of the mesh of the bridge and they are made exploiting ExtrudeGeometry and in particular defining bezier curves. Also the arm rest are made using ExtrudeGeometry. They are made following the Three.js documentation. All the objects that spawn over the bridge are imported from OBJ files and inserted in a THREE.Object3D and are child of the mesh of the bridge.



4.1.3 OBJ and GLTF

Spike-balls, star and hearts OBJ have their own material files and are implemented as classes in the "loader.js" file. They are scaled and set in the correct place using the **Box3** class.
For spike-ball i decided to modify the color, reflectivity and specular properties of the material.
Stalagmites are GLTF files and i defined a class to load them.

4.2 Textures

4.2.1 Bridge

I tried in many ways to make the texture of the bridge move in order to have a sort of treadmill of stone, but unfortunately i could not find a reasonable solution. For the boxGeometry of the bridge i used as color map the "assets/bridge/stonetext.jpg" file and a normal map "assets/bridge/StoneNormalMap.png", every side of the box has its own texture parameters. I used the same files for the arcs.
For the rest arms i used also a color map "'assets/bridge/restArmStone.jpg"' and a normal map "assets/bridge/restArmStonNMap.jpg".

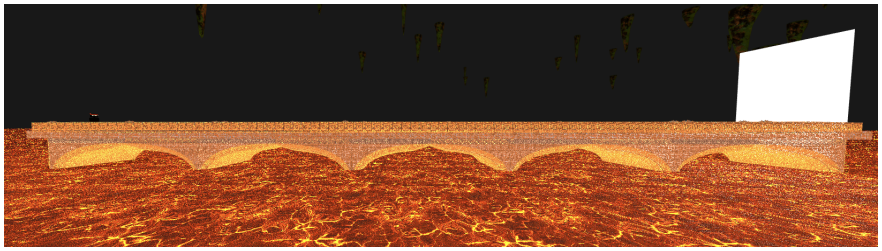


Figure 1: the bridge

4.2.2 Lava

I wanted to use an gltf model with different textures, but it was to heavy about 200MB and github doens't allow me to upload it. So for i decided to attach textures to a plane with. Here a list of the textures i inserted.

- color map "./assets/lava/color.jpg"
- ao map "./assets/lava/ao.jpg"
- normal map ./assets/lava/normal.jpg"
- roughness map "./assets/lava/roughness.jpg"
- emissive map "./assets/lava/emissive.jpg"
- bump map "./assets/lava/height.jpg"

4.2.3 Stalagmites

They have two types of textures a normal one "$./assets/stalagmites/textures/stalagmite_normal.png$" and a color one "$./assets/stalagmites/textures/stalagmite_baseColor.png$". I wanted to simulate the atmosphere of a cave.

4.3 Lights

I used five lights an **AmbientLight**, a **Directional light** that always follow the position of the character, in order to cast his shadow under his feets, an **HemisphereLight** to have the darkeness of the top and the light of the lava, two **RectLight** once to better simulate the light of the lava under the bridge the second one to simulate a light at the end of the bridge. In last one i used also RectAreaLightHelper to show physically the light .

4.4 Animations

All the animation are based on keyframes.

4.4.1 Jump

The **jump** is made by seven keyframes chained between them. The first one is the standard position followed by the jump initial position.

Figure 2: the first two keyframes

the third keyframe bring the character at (y 3.5) with a cubic out transition. the fourth keyframe start when the character star to fall, followed by the the tween that brings the character at the ground with a cubic out transition, and the last one make to assume to the character the standard position.

Figure 3: start falling frame and initial position when touch the ground

4.4.2 Running

The transition between run and jump is possible due to at the end of the jump the character return at the standard position, from where start the run animation. Also at the start of

the jump animation the character return to a standard position.

The **running** animation start from the standard position and is made by three tweens chained. The first has a linear transaction, the others two have a sinusoidal out. The last two tweens are chained between them to create an infinite effect run. To make it more realistic during the run animation i rotate also the upper arm roots.

Figure 4: three tweens of the run

I decided to insert also a torso rotation during the run but i had to separate that animation from run due to an incompatibility with the dashed animations. This animation is called trumbling.

4.4.3 Dashes

There are four dashes animations to distinguish the different movements of the lanes. All the dashes are composed of two keyframes. These keyframes are just rotation of the torso.

4.4.4 Fall animation

The **falling animation** is composed of four frames, the first one is the standard position of the character, the other two are rotation and movements of various parts of his body, and the last one is the gravity that bring the character one the lava ground. The gravity has a separated tween, the others three are chained. The last two are chained between them to create an infinite animation.

Figure 5: last two tween of the falling animation

4.4.5 Objects

I decided to not use tween js to make the object move over the bridge, due to some conflict with the implementation of the pause. The objects are moved exploiting a clock defined from threejs library and a speed. the multiplication of the speed and the time passed give as result the new position of the object over the bridge.

The spike-balls are continuously rotated using tween and making use of onUpdate method. This allow me to not define different keyframe to make the ball rotate, but simple update the rotation of the ball summing the rotation of tween.

4.5 Physics

I decided to not use a physic engine.
I implemented the collision system exploiting **Box3** class.
the box of every object is matched with he box of the character. Every object over the bridge has is own type. When a collision between character and an object happens, the code read the type of the object and behave in a different way.
This must be done for every lane. The fall of the character is implemented in the tween animations.
the movement of the object is implemented in world.js using the time elapsed multiplied for a speed.

5 Other notable stuff

5.0.1 Pause

To pause the game i exploited the **Clock** class of threejs, so i defined a clock in order to block the time count when ESC is pressed. The delta time of the clock is passed to the world in order to correctly move the objects over the bridge.
For the animation i use a method of the TWEEN.Tween class in order to get all the active

tweens and stop them till resume button is clicked.
So when ESC is pressed the clock and all the actives tweens are stopped, and overlay is made visible.
When resume is clicked the clock and also all the tweens that has been stopped starts again, and the overlay is set at invisible.

5.0.2 Blink effect

The blink is implemented making use of the emissive property of the materials.
I defined a recursive function that utilize setTimeout function of javascript in order to set the correct color for a time interval.

5.0.3 Loading screen

This is realized using THREE.LoadingManager class.
The loader is passed to all the objects and textures to be loaded . The loader is not passed to the objects over the bridge that spawns at runtime.
After all object are loaded requestAnimationFrame(render) function is called in order to start the game.