INTERACTIVE GRAPHICS FINAL PROJECT

Andrea Popescu 1969614

1. Introduction

Several environments and libraries were used in the implementation of this project, namely:

- WebGL: a JavaScript API for rendering high-performance interactive 3D and 2D graphics within any compatible web browser without the use of plug-ins.
- Three.js: 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. The features of the library used are the scene, thanks to which the 3D models were inserted, perspective camera, lights as hemisphere, point and directional, cubes and planes to create the game world and the Gltf loader to import 3D models.
- Cannon.js: an open source JavaScript 3D physics engine. Used to insert physics into the game.
- Tween.js: a simple tweening library for use in JavaScript. Useful to create smooth animations.

All 3D models were downloaded from a site called Sketchfab, while other textures from planetpixelemporium and cleanpng. The textures applied on the player and on the enemies were done by my brother.

The project carried out we can classify it as an exploration game, where you will play the role of an astronaut who just got off his spacecraft on a new planet, inhabited by aliens.

The game has been tested on both google chrome and firefox, and on both browsers it works fine.

2. Environment

The game world is nothing more than a skybox, or a giant cube, where the sky, distant mountains and other unreachable objects are projected onto the faces of the cube (using a technique called cube mapping), thus creating the illusion of a distant three-dimensional environment. The environment is composed by a scene, which allow you to set up what and where is to be rendered by Three.js (this is where we can place objects, lights and cameras) and by a world, the physics world which we can create through Cannon.js.

The actual ground of the game is formed by a plane implemented with Cannon which has its surface at z=0 and everything below z=0 is assumed to be solid plane.

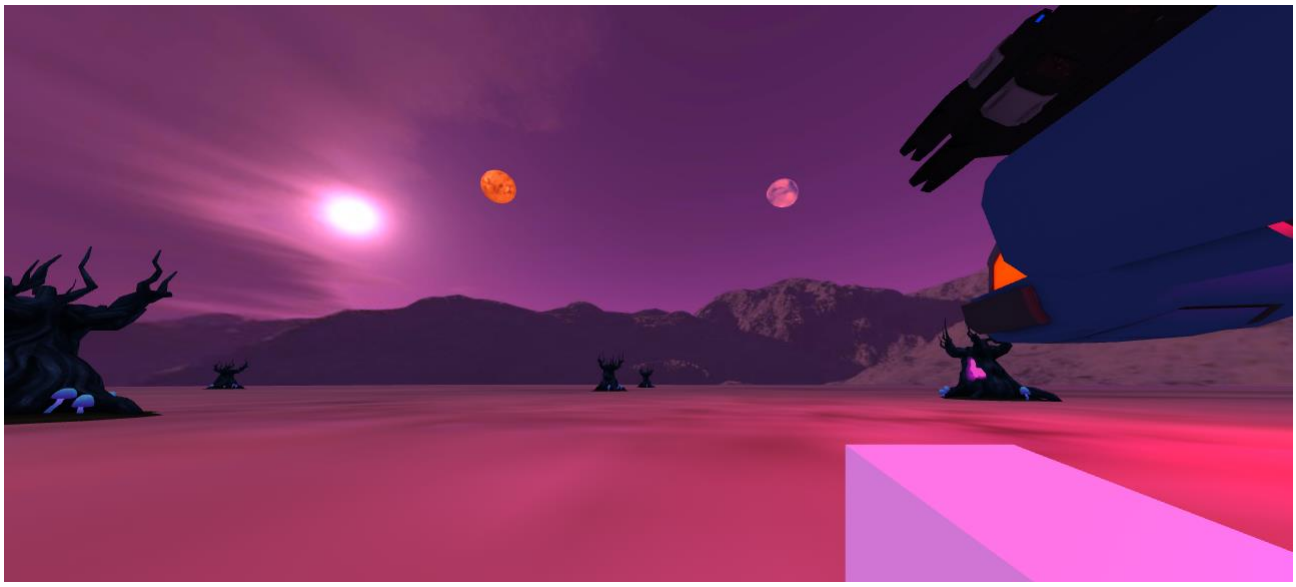


Figure 1: game world

In the background we can also see a giant black spaceship, which is nothing more than a 3D model imported through the gltfLoader() and also two planets rotating around its y-axis. A bump map was applied to them in addition to a normal texture.

As for the inanimate material objects that can be observed, we also have a blue spaceship, which by lore we can say is the one the player landed on the planet with. It is also a 3D model, but inside it there is a boxCollider (Cannon.box()), which makes the body not penetrable. It has a point light inside is mesh. There are also trees that by the very implementation of the spacecraft cannot be passed through.

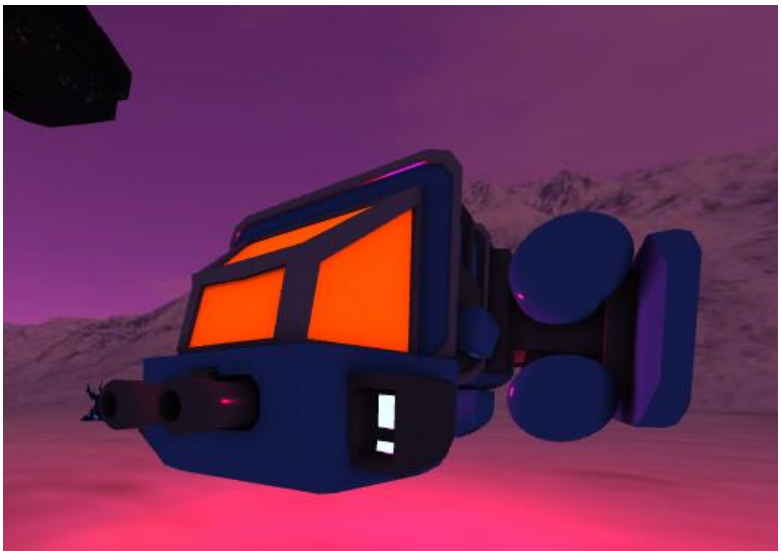


Figure 2: Blue Spaceship

All these models and their respective rigidBodies, once imported with the addModel method, are saved inside the list this.modelli and if they have a boxCollider in this.modelliCannon, which are then used in the update function. So, by scrolling through them we can then go and modify some properties of the individual models.

3. Hierarchical models

3.1 Player

The playable character is a hierarchical model, consisting of the head, body, leftLeg, rightLeg, leftArm, and rightArm nodes. They are all THREE.BoxGeometry(). As mentioned above, homemade textures have been applied. In particular, on one side of the head (the face) and on the main sides of the body, that is those that should simulate the belly and back. On the other parts instead has been applied only the white colour. All surfaces are MeshPongMaterial, so for calculating reflectance. All this is contained in the JavaScript file CharacterFactory.js, which takes care of "building" the spaceman.

Several animations such as classic walking, running and jumping were implemented. They can be activated by pressing the following keys: W A S D, tab and space. The direction in which the

movement takes place is given by the movement of the mouse, which also changes the position of the main camera. This part instead is entrusted to the CharacterController.js file.

3.2 Aliens

The hierarchical structure of these entities is the same as that of the astronaut, but several elements change: the size of the boxes that make up each part of their body, the colour applied, which in this case is green, and the texture (here we have one applied only to the face). All this implementation part is contained in the AlienFactory.js.

These will spawn in a random location within the game map and start heading towards a randomly assigned destination. If it is in the vicinity of a such location, it will be assigned a new one. The movement animation is the same as in the player.

An alien can interact with the player if the latter decides to get too close, in which case the alien will stop and start moving left and right while looking at us. If we try to move away, it will start to follow us. If we manage to escape it, it will return to its initial destination.

All this is handled in BasicAlienController.js.

Instead, the EntityManager.js file takes care of creating and initializing all the parameters of the entities in the game, be it the player or the aliens.

Figure: hierarchical models

4 Menu and Options

4.1 Menu

There are 3 different menus within the game:

- The Main Menu: is what you see if you enter the game's web page, where you can either start the game or change the settings, thus accessing the appropriate menu.
- Pause Menu: If during the game you decide to press esc you will pause the game and the menu will open, from which by pressing a button you will resume the game, or you can decide to return to the main menu.
- Option Menu: Here you can change the game options through sliders and buttons, save those changes, or return to the default ones. They will be analysed in more detail later.
  Cookies have also been managed: by changing one or more options and then saving those changes they will remain until the options are reset, even when accessing the site in different sessions or restarting the web page. Without cookies, even entering the game and then going back to the main menu the options would reset to "default".

4.2 Options

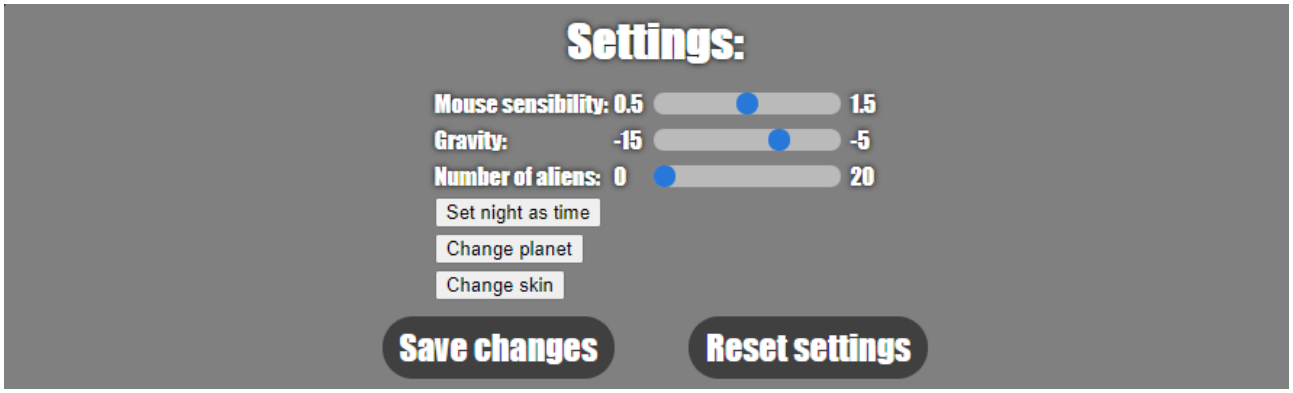
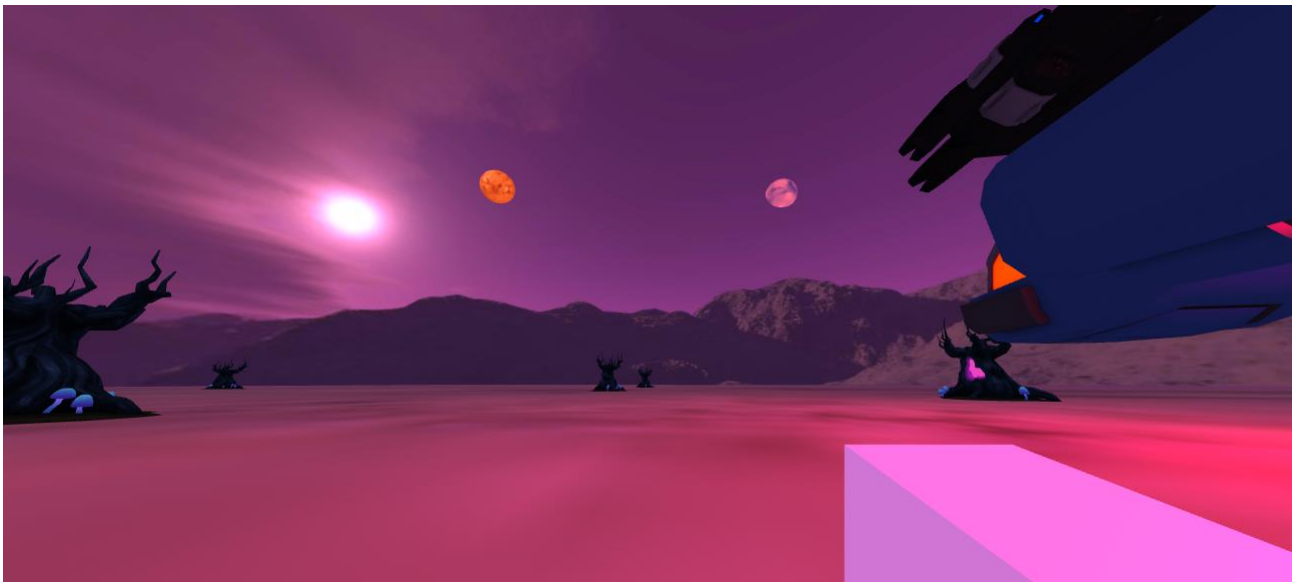


Figure: Option Menu

As you can see from the figure above, the options that can be changed are:

- Mouse sensibility: its default value is 1.

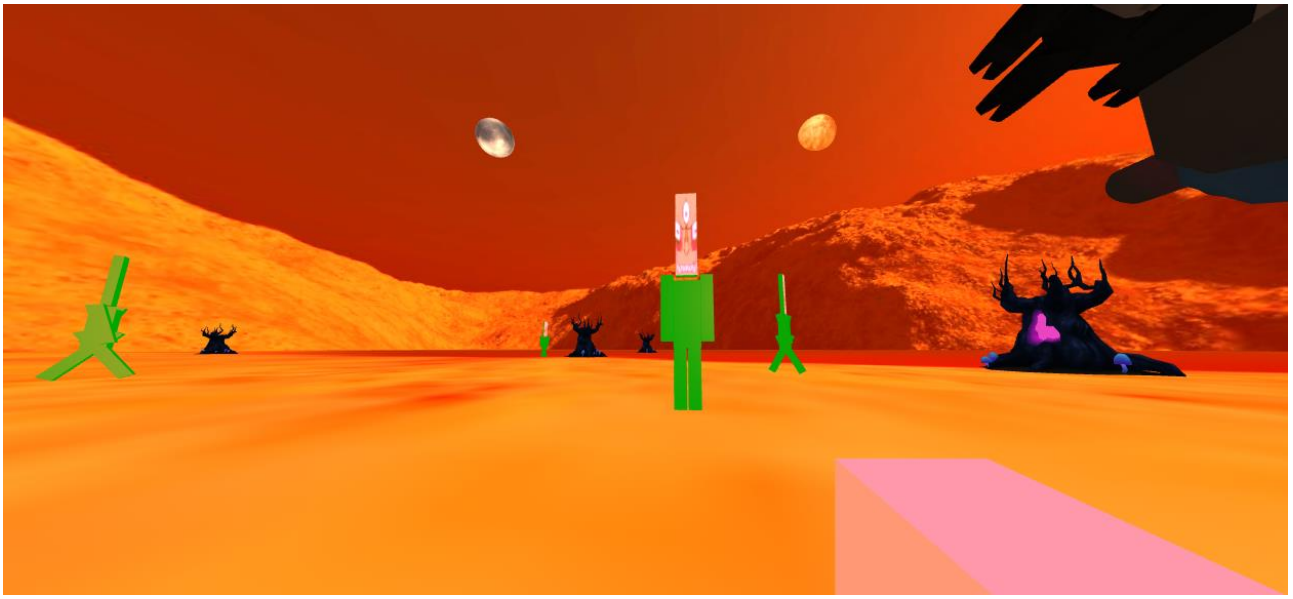
- Gravity: changing this parameter will modify the gravity in the game, which is a property of the world implemented with Cannon. The lower the value, the higher you will be able to get by jumping. Its default value is -8.
- Number of aliens: with it you'll be able to set the number of "enemies" within the game. As default number of aliens we have 10.
- Set night as time: Clicking this button will change the lights, so it will change from day to night, or vice versa if it has already been clicked. In particular the Hemisphere light will be removed or put back on. In the images below you can see the differences between day and night.



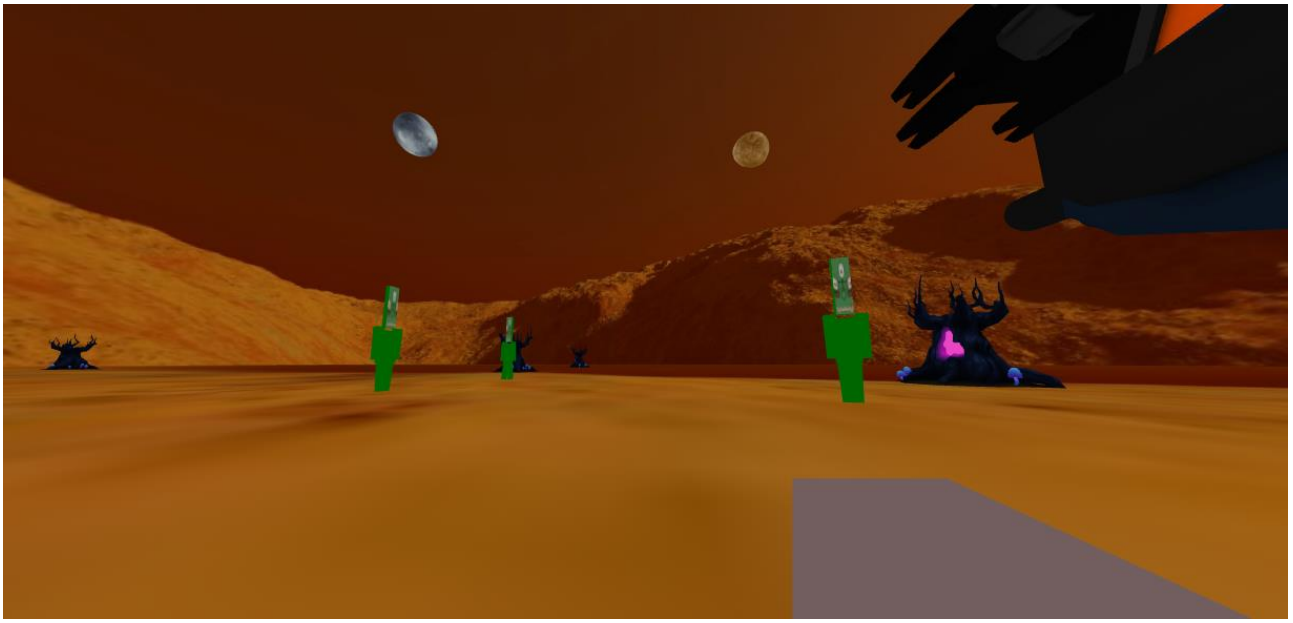
Planet with day as time

Planet with night as time

- Change planet: Clicking that button will change the textures of the skybox this time, thus changing the game world from this point of view. Being this a game of space exploration, with this option I wanted to give more variety. The textures of the two smaller planets that you see rotating in the background will also change. Also the main light, the Hemisphere one is different, as it is an orange light, while that of the first planet is purple. Below we can see the second planet by day and by night.

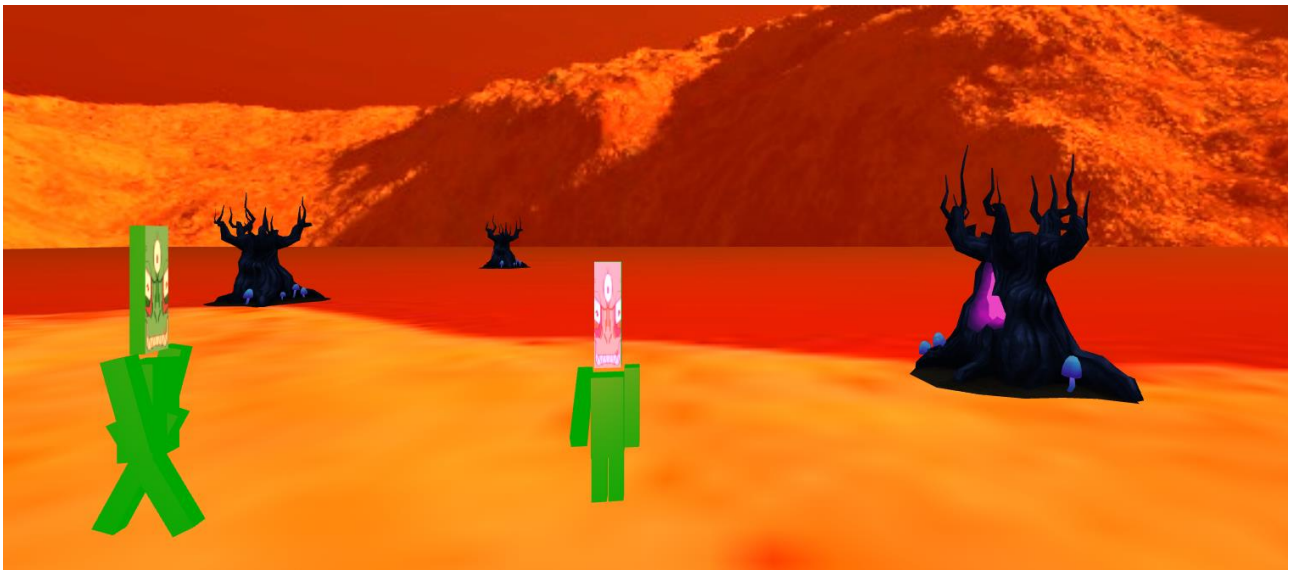


Second planet daytime.



Second planet at night

Change skin: with this option we can change the skin of the player, which will look like an alien.



The player is the one in the middle of the picture.

5 Camera

The game initially presents a first-person camera, but by pressing the V key, you can switch to the third one. They are both two perspective cameras, and their management is done in CharacterController.js. Pressing a third time that key switches to another camera, also of perspective type, which simulates the view from the blue spaceship.

Third-person camera