Final project Interactive Graphics

Master's degree in Artificial Intelligence and Robotics

Course: Interactive Graphics
Professor: Marco Schaerf

Authors:
Alessandro Lambertini – 1938390
Denise Landini – 1938388

SAPIENZA
UNIVERSITÀ DI ROMA

La Sapienza University - Roma
Academic year: 2020-2021

SUMMARY

1. BASIC WebGL, Three.js, Cannon.js

In order to do the project, we have used:

1) WebGL;
2) Three.js;
3) Cannon.js.

by using html and by programming in JavaScript.

We have downloaded the 3D models from Sketchfab that allow us to directly upload our 3D models in the project.

WebGL is a JavaScript API library that is used in order to render 2D and 3D graphics in the browser.

Three.js is an open-source library that is based on WebGL. Three.js contains a lot of features such as:

- **Scene** that allows us to add/remove the 3D objects;
- **Cameras** (perspective and orthographic);
- **Lights** and we can add the shadows;
- **Materials:** Lambert, Phong, etc…;
- **Objects**: bones, meshes, etc…;
- **Geometry primitives**: cubes, planes, spheres, cylinders;
- **gltf loader**: in order to load our model and textures.

In particular, in our project we have built the environment by using cubes, cylinders and planes to which we have applied textures in order to make it more realistic.

Cannon.js is an open-source JavaScript 3D physics engine that supports different kind of shapes such as box, plane, cylinder.
We have used Cannon.js with Three.js in order to generate physics-based 3D scenes.
For example, with Cannon.js, Mario cannot cross the bricks, enemies, etc….

2. ENVIRONEMENT AND ENTITIES

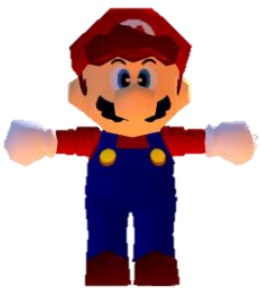
2.1 MAP

***Figure 1**: Map of the environment*

2.2 MODELS

All the 3D models we have used are imported from an open-source web sites that is called Sketchfab[1].
The models are loaded when the game starts and we use the function clone( ) in order to copy the models that are used several times in the environment. This approach improves the execution time because we do not load many times the same model but simply we clone it. So, we have not the slow down phenomenon due to the loading of the model.

2.2.1.0 Mario



It is a hierarchical model and we have done the animations by using basic WebGL. Mario is the character that is used by the player.

We have done 4 different animations:
1) idle;
2) walk;
3) run;
4) jump.

All the animations are implemented by using JavaScript.
In order to do the animations in a simpler way, we have created the *marioDict* that allow us to access to the different part of the body (bones).
This dictionary is useful because it reduces the computation time; in fact, each time that we have to access to a different bone in Mario's model, we have not to search this in the hierarchical structure, but we have the reference in the dictionary.

In general, for all enemies and objects that has a hierarchical structure we have created the dictionary for the same reason.

[1] https://www.sketchup.com/

2.2.2 ENEMIES

2.2.2.0 Bullet Bill

This model is not a hierarchical model, but it is simple.
The animation is done manually by using the basic JavaScript and it consists in changing the x position. This animation is always active during the game, independently of the position of Mario.

If Mario collides with Bullet bill, it loses one life.

2.2.2.1 Cube enemy

This model is not a hierarchical model. We have simply built it by using a cube and we have applied different texture for up and down face of the cube.
The animation is done by using JavaScript and it is active all the time. The animation consists only in changing the position of the y axis.

If Mario is crushed by cube enemy, it loses one life.

2.2.2.2 Goomba

This is a hierarchical model and we have done manually the animations of the foots and body by using basic JavaScript.

For Goomba, we have implemented 2 different animations:
1) Goomba is stationary, and he waits for Mario who approaches him to start to walk;
2) Goomba chases Mario.

The animations are implemented with JavaScript, and we have also implemented an AI algorithm in order to have a more realistic chase to the player.

If Mario collides with Goomba, it loses one life.
This is the only enemy that Mario can kill by jumping on his head. In this case, Mario gains 20 points on the score.

2.2.2.3 Whomp

It is the final boss that Mario has to avoid in order to collect the last star.
This model is a hierarchical model. The animations are done manually by using basic JavaScript and we have implemented 2 animations by using the same approach that we have used for Goomba:
1) Whomp is stationary, and he waits for Mario who approaches him to start to walk;
2) Whomp chases Mario.

If Mario collides with Whomp, it loses one life.

2.2.2.4 Piranha

It is a complex hierarchical model.
We have added this model only one time in the garden.
The animations are created by using basic JavaScript and we have 2 animations:
1) Piranha is stationary and he waits for Mario who approaches him to start to run;
2) Piranha cashes Mario by running and by moving the plant on the head.

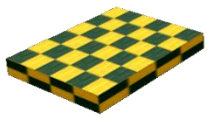
If Mario collides with Piranha, it loses one life.

2.2.3 OBSTACLES

2.2.3.0 Bridge

We have another obstacle that Mario has to pass in order to win the game which is the bridge that spins in a circle. For this bridge we have implemented a simple rotation around the y axis and this animation is active during all the game. In the game we have other 2 bridges, but they are static.

2.2.3.1 Transporter



Before to arrive to the towers, Mario has to jump on the transporter that allows him to arrive to the last layer of the game. This is a small platform that is animated by using JavaScript and consists of changing the y position.

2.2.4 COLLECTABLES

2.2.4.0 Coin

This model is cloned a lot of times because we have placed these coins several times in the environment (i.e., on the garden and on the bricks). In this case, the animation consists only in a rotation around the y axis.
Every time that Mario takes one coin, he gains 10 points in the score.

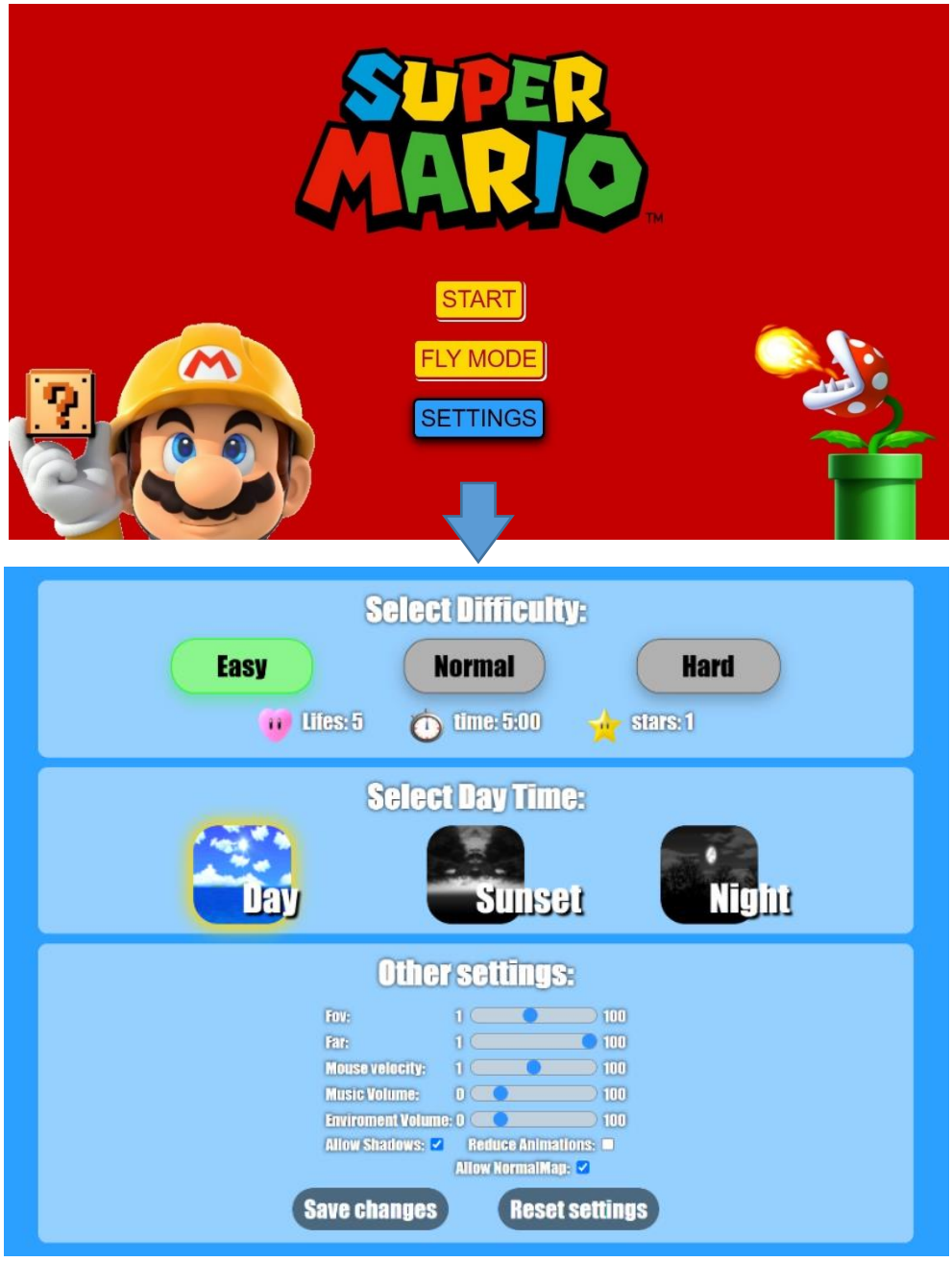
2.2.4.1 Star

This model is cloned three times because we have placed these stars at most three times in the environment according to the difficulty. In this case, the animation consists only in a rotation around the y axis.
Every time that Mario takes one star, he gains 100 points in the score.

2.2.4.2 Block

This model is cloned five times in the environment. Each block contains five coins.
Every time that Mario hits from the bottom the block, he gains 1 coin.

3. TECHINICAL ASPECTS



3.1 SETTINGS OF THE GAME

3.1.0 Difficulty settings

Some parameters change according to the difficulty that the player choose.

	Stars	*Times*	*Lifes*	*Enemies*
Easy	1	5.00 min	5	7
Normal	2	3.30 min	3	9
Difficult	3	1.30 min	1	11

***Table 1**: Difficulty setting*

3.1.1 Light settings

We have thought to allow the player to choose between three different parts of the day.
In each part of the day, we have a different sky box and different lights.

1) Day and Sunset: we have a directional light to simulate the sun and an ambient light.
   In the Day case we have a white light, in the Sunset case we have an orange light.
2) Night: we have an ambient light, and the yellow blocks emits light.

***Figure 2**: Day modality*

***Figure 3**: Sunset modality*

***Figure 4**: Night modality*

3.1.2 Other settings

The 3 most important settings are the 3 checkboxes that are added for improving the performance of the computer or in order to have a better graphic result.
In fact:

- allow shadows: add/remove the shadows in the environment;
- reduce animations: add/remove the animation from the enemies, Mario, coins and stars;
- allow NormalMap: add/remove normal map to the textures.

We use the cookies to save the setting of the user.

3.2 TEXTURES

When the game starts, we load all the textures individually. For the textures that require it, by using the textures that we have just loaded, we assign a texture to each face of the cube/cylinder in order to have a better effect.
The textures are taken from internet, and they are loaded, if necessary, together with the normal map.
The normal maps are generated with an online tool [2].

3.3 COMPATIBLE BROWSERS

We have tested the software on:

- Google Chrome;
- Microsoft Edge;
- Firefox.

We have seen that we have better performance with Microsoft Edge, on Google Chrome with some computer it can slow down and finally on Firefox we have seen that the software is slow but it is perfectly executable.

[2] https://cpetry.github.io/NormalMap-Online/

3.4 HANDLE OF ENEMIES AND ANIMATIONS

- In order to handle the enemy, we have created the EntityManager.js.
- For Mario and enemies animation, we have created a Finite State Machine (FSM) for each character because they have different animation as written before; so, we want to change between them when some conditions are verified.
  In conclusion, the FSM is used in order to handle the animations.
- In order to initialize all the parameters of the scene we have used static factories;
- Finally, we have created some controllers in order to move the character and the enemies commanded by the AI algorithm.

3.5 AUDIO

The audio tracks played during the game is download from internet and we have used those that are used in the original game.
In particular, we have used different sound (especially for interactions) when:

- We are in the main menu;
- We start the game; we have a continuous sound during all the game;
- We push the buttons;
- Mario loses one life;
- Mario takes a coin or a star;
- Game is over;
- Game is won.

We also have different screens if the player wins the game or loses it.

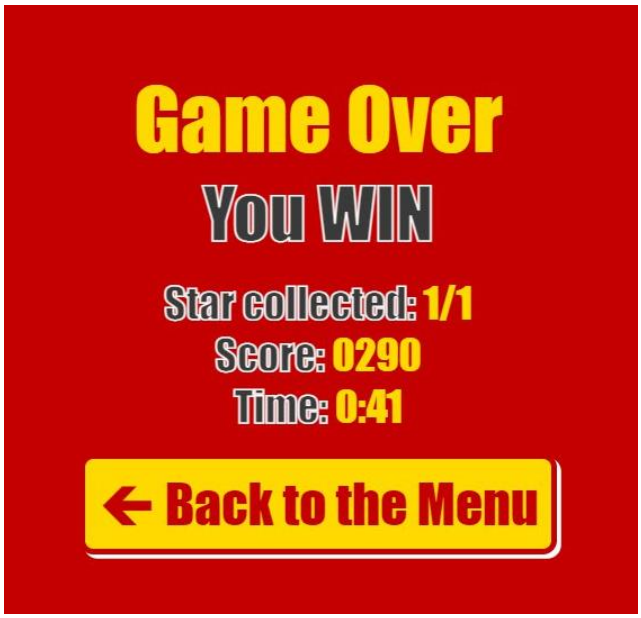


Figure 5: *Screen when player wins*

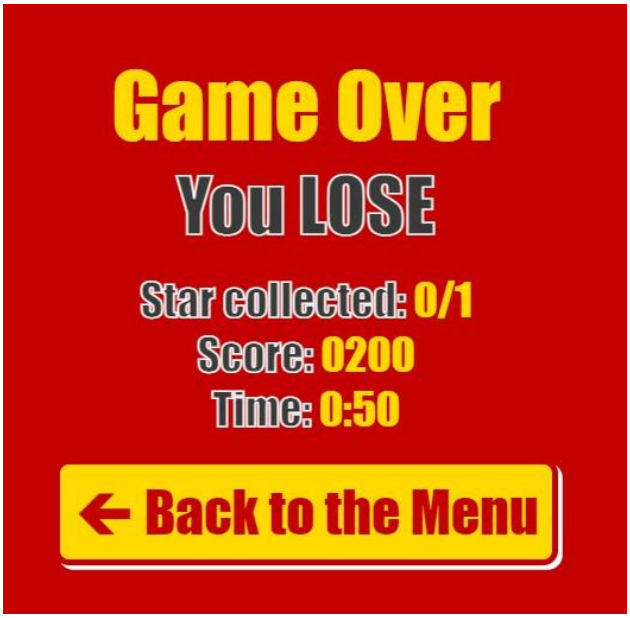


Figure 6*: Screen when player loses*

These images show the total count of star collected, the score gained and the total time of the game.

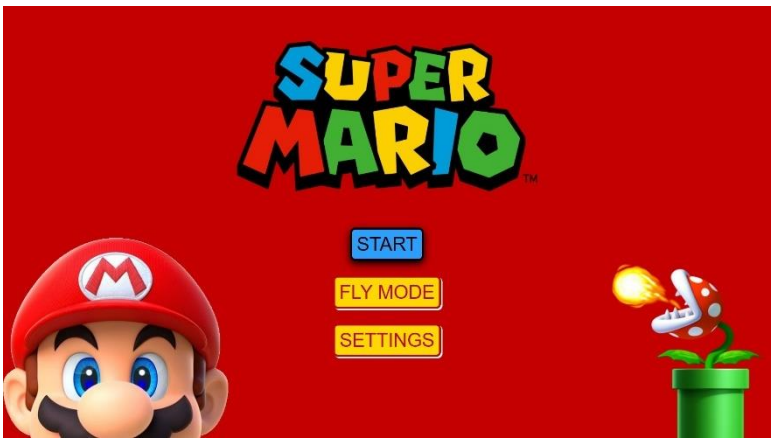
4. FLY MODALITY

We have also decided to implement a fly mode that is useful only for better visualize the environment. For this modality, we have implemented a different animation in JavaScript for Mario in which he flies. In this modality, Mario can pass through everything (bricks, enemies, etc…), but obviously he does not lose lifes.

COMMANDS:

- **W A S D**: directional movements;
- **SPACE**: go up;
- **SHIFT**: go down;
- **MOUSE**: move the camera;
- **ESC**: pause.

5. GAME MODALITY



GOAL: During the game, Mario has to collect the highest number of coins and stars in the shortest possible time.

COMMANDS:

- **W A S D**: directional movements;
- **SPACE**: jump;
- **SHIFT**: run;
- **MOUSE**: move the camera;
- **ESC**: pause.