Report: Interactive Graphics Final project

Fabrizio Casadei - Matricola: 1952529
Edoardo Papa - Matricola: 1962169

September 11, 2021

1 Game and dependencies

Our final project of the course "Interactive graphics" concerns the development of a browser game using a library called **Three.js** based on **WebGL**.
The concept under the game is pretty easy. The user controls a magic ball that moves by the imprinting of directional motion, to reach the goal which is represented by a teleport which leads to the next level and eventually the won of the game.

Figure 1: "The main title of the project"

Crazy ball lab is a kind of classical platform game, in which not only the correct management of the ball is important but also the ability of the user to control the camera in a proper manner. Hence, a key aspect of the game is the user interaction.

In the game has been used an open-source physics library called *PhysiJs* build on top of the *ammo.js.* This library allows the definition and realization of realistic animations for several available shapes like spheres and cubes. Integrating the *Three.js* with this library we have managed physics-based scenes introducing the concept of hitbox for models, gravity, angular velocity and so on.

Therefore, to sum up we have utilized these dependencies:

1. WebGL, a JavaScript API for rendering 2D and 3D graphics in the browser using the GPU.
2. Three.js, an open-source library based on WenGL which simplify the usage introducing a lot of classes and functions to handle: Camera, Lights, type of shaders (Phong, Lambert, etc...), geometry models, objects for complex models, and a lot more features.
3. PhysiJS, an high-level library that allows introducing physics in projects that use *Three.js.*

2 Environment and User interaction

This game uses very simple models, we have more focused our attention on how to realize a realistic ability-game using physics. Hence, all the models that we present are realized directly using three.js geometries. The creations of elements for the environment and their animations are managed in the file called "utils.js" which allows the creation of a large part of the scene with a simple call. Moreover, in this file, we have defined functions for the camera, the lights, and general information about the statistics of the game, and more.

2.1 Models

2.1.1 The ball

The model that would be controlled by the user is the ball and is a very simple spherical geometry with a high number of *subdivision* which reaches a good visual approximation. A color texture with squared pattern has been chosen, wrapped and repeated to reach the following result.

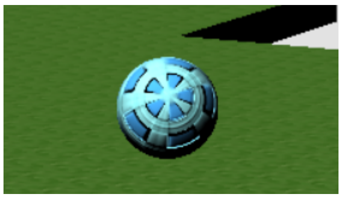


Figure 2: "The crazy ball controlled by the player."

Moreover, using online tools has been extracted and used the normal and specular map for the texture, in order to achieve better results with *Phong shading*.

2.1.2 Platforms

Platforms build up the environment of the game, designing paths to reach the end of each level. The main challenge of this game is to don't fall from these platforms with the ball, imprinting forces on the 4 main directions (according to the camera position). Each platform is made up of an atom element of the game, which are the boxes.
For the platforms eight distinct sets of textures have been used, to form platforms of grass, rock, etc. platforms have different morphologies and behaviours, for example, uphill/downhill platforms (stairs), animated platforms, falling platforms, etc.

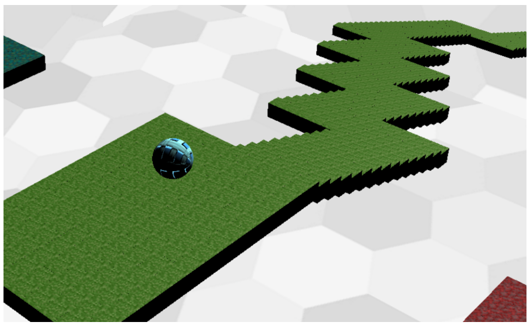


Figure 3: "An example of platforms utilized in the first level."

2.1.3 Cyber Guy

The Cyber Guy is a model that allows the user, making him fall, to gain extra lives to avoid the game over.
This is a hierarchical model that has been build from us, as the root we have the *hitbox* handled through physiJS, then follows the head, the body, left and right arm, left and right leg. Using this hierarchical structure is possible to move manually the model with animations and also interact physically using the ball model. Several color textures have been used for each part of the body, and normal map textures as well (always obtained through online tools).
The animations of the Cyber Guy are implemented using only JavaScript (like in the second homework) and are these:

1. Jump
2. Walk back and forth
3. Walk in circle

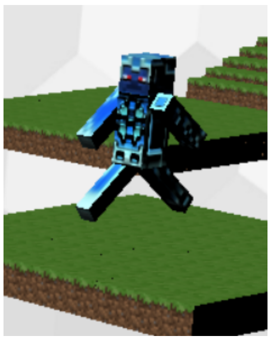


Figure 4: "The Cyber Guy."

2.1.4 Checkpoint button

The Checkpoint button is a very simple model created using a flat cylinder with 64 subdivisions. we have fixed a certain base color, specular and emissive to characterized the material using *Phong* shading. This button has a very simple animation which makes it change the color (texture, specular and emissive) when is hit by the ball. This highlight the fact of an active checkpoint. So the next time that you fail will restart from the button. Moreover, when a button is activated all the models that are no more useful to continue the level are eliminated, in order to have a lighter scene.

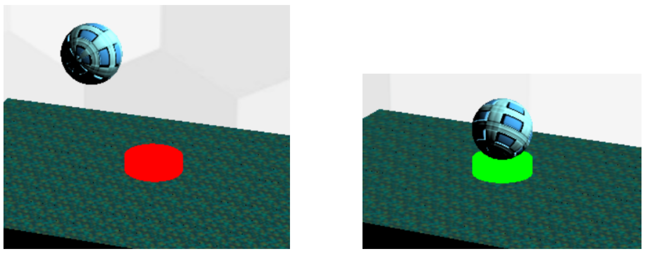


Figure 5: "The checkpoint button, before and after the activation."

2.1.5 Destructible Walls

In the game are present walls that are prepared to be hit. The walls are build of cubes that are seen as "dynamic" from the physiJS, and can have the same set of textures that are used for the platforms.

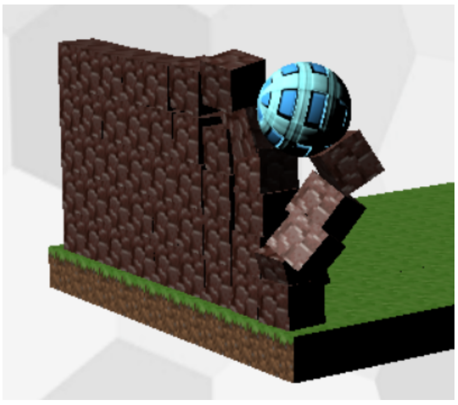


Figure 6: "The wall hit by the ball."

2.1.6 The teleport

The teleport is another simple model build using a cylindrical geometry and 32 as a number of subdivisions. For the teleport is used a pixelated style texture with an opacity of the 80%. The teleport is always animated to rotate on itself. To complete the level it's enough to enter in the teleport with the ball.

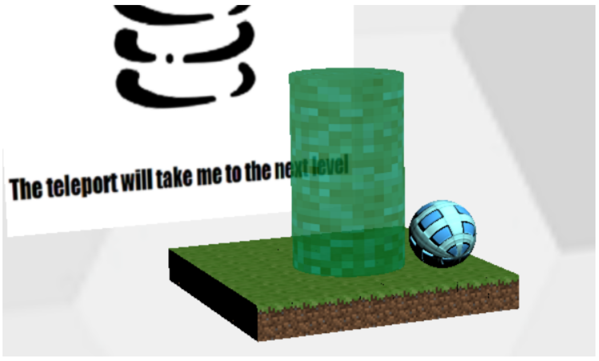


Figure 7: "The Teleport."

2.2 User interaction

Before entering in the game the user can select using buttons and sliders several aspects of the game. For example, it's possible to choose to start from a specific level or to handle the parameters of the *perspective camera*.
While in-game the player use the following keyboard commands:

- **W button**, move the ball forward.
- **S button**, move the ball backwards.
- **D button**, move the ball to right.
- **A button**, move the ball to left.
- **Space button**, jump with the ball.

All the motion is thanks to the imprinting of force in a certain direction, and all the directions are relative respect the position of the camera.
Given the fact that we are simulating real physics, for example, whether we want to change direction in the opposite way, we first have to contrast the inertia of the object, and only then our object will accelerate in the other direction (if we are still pressing buttons for the same direction).
This leads to a not easy system of motion for the ball, that the user is called to master.

The directional motion imprinted for each command is displayed on the screen using a green arrow on top of the sphere.

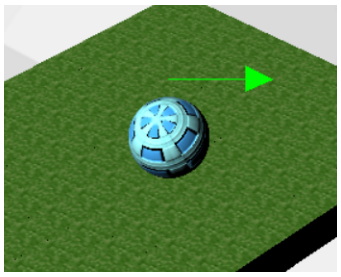


Figure 8: "The ball in motion."

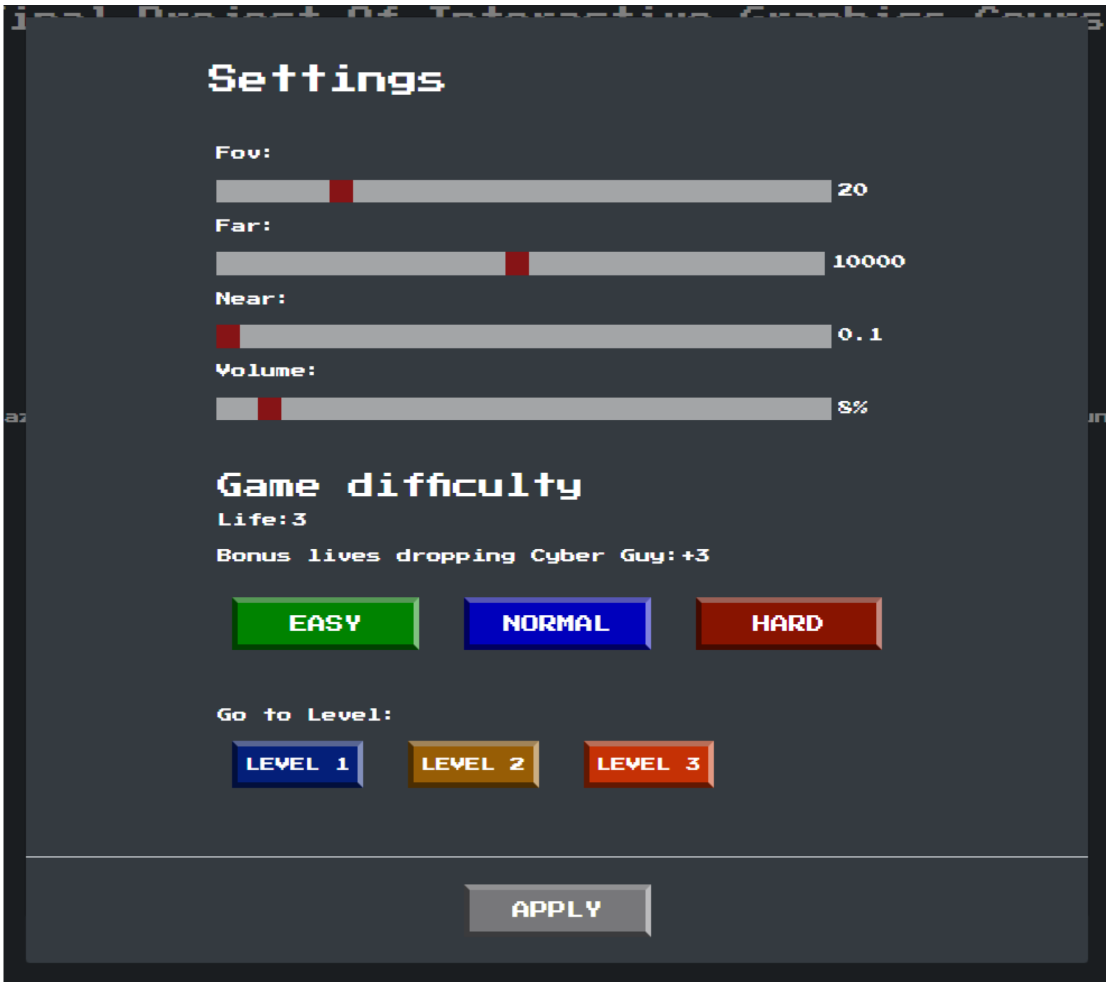
One another important command for the user is the re-positioning of the camera using the mouse. In specific pressing the left button and moving the mouse will cause the rotation of the camera around the ball, which is the anchor point.
This aspect is crucial to learn how to move the ball most accurately, ad avoid obstacles between the camera and the ball for a clear view.

3 Technical details

In this final section, we discuss several technical aspects, starting from the menu and what is possible to choose, then we talk about the camera, light system and audio of the game. To conclude we expose the results of testing the game on the main browsers.

3.1 The Menu

On the main page, the user can select to start the game, from the first level, or change the settings of the game.



Settings

Fov: 20

Far: 10000

Near: 0.1

Volume: 8%

Game difficulty

Life:3

Bonus lives dropping Cyber Guy:+3

Figure 9: "Settings for the game."

Here the player can change settings for the *perspective camera*: FoV, Far and Near and the volume of the audio. Moreover, can change the difficulty of the game. Below it's shown a table to expose the differences.

	Easy	***Normal***	***Hard***
Lives	3	2	1
Bonus Lives for Cyber Guy	3	2	1

After having changed some settings, the user has to press apply to confirm them. It's also possible to select the level from which we want to start directly here pressing one of the buttons which are present.

Two different pages to inform the player of the game over or the win are been developed. Furthermore, a foreground layer for the loading of the scene has been produced which disappears when the progress bar reaches the 100%.

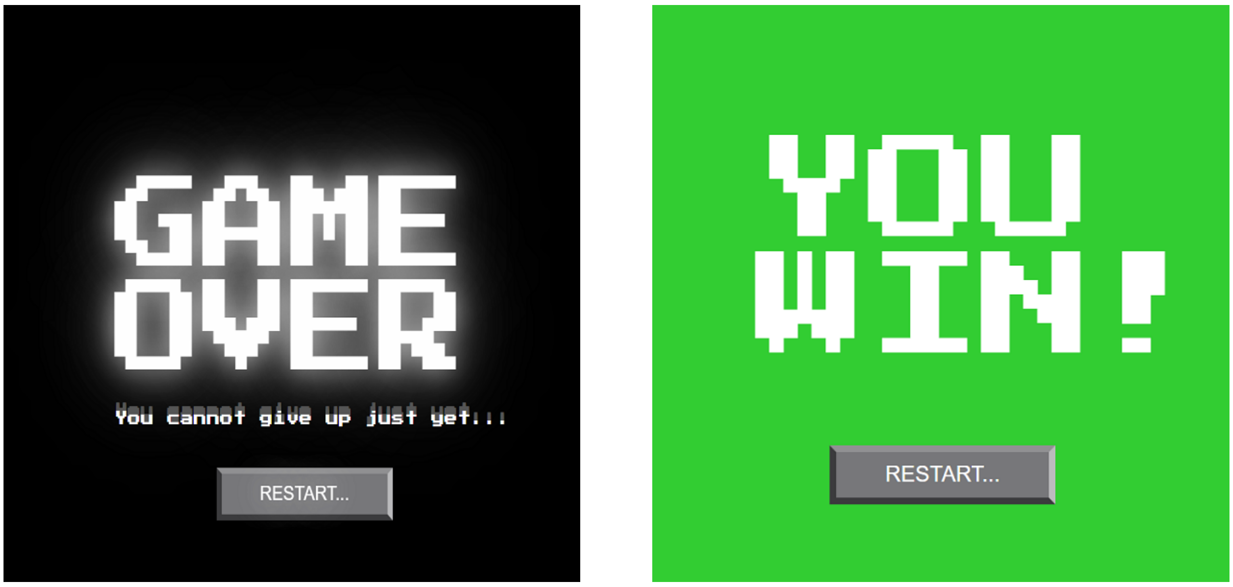


Figure 10: "Game over and won pages."

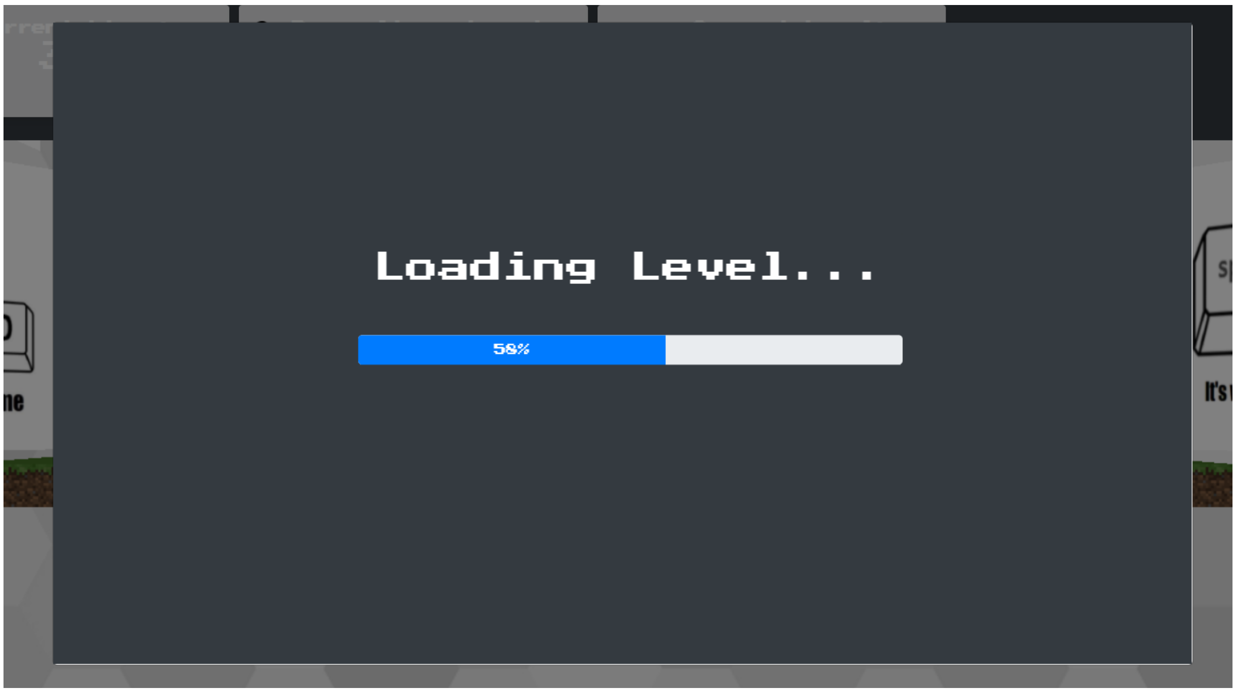


Figure 11: "Loading of the levels."

3.2 Camera and Light

In the game scene we have obviously inserted the camera which is perspective and is characterized from the attributes that we have exposed in the settings menu.
The camera aspect is modified considering the canvas width and height used, in order to better representing the scene. The camera follows the y and x position of the camera, while for the z a certain constant ,that characterized the distance, is summed to the z of the sphere. The camera position is controlled using *Orbit controls* a *Three.js* module.

Several light systems have been tried, but in the final version of the project, the directional light has been the choice. The direction of the light is described from this unit vector: **Vec3(0,1,1)**, while the color is neutral white.

3.3 Audio

The audio of the game is composed by 6 different tracks that are mainly background sounds. all these files are been download from several internet sites that allows to download free sound effects. Here we present the list for them.

- Background music for the first level.
- Background music for the second level.
- Background music for the third level.
- jump sound effect.
- Game over music.
- Game won music.

3.4 Testing on browsers

The project has been tested on the following browsers:

- Microsoft Edge.
- Google Chrome.
- Mozilla Firefox.

We have recorded better performance using Microsoft Edge, without any technical problems.
Chrome runs the software correctly but a bit slower with respect to Edge. While using Firefox the rendering is very slow and there are several problems with the commands and the position of some models.