Interactive Graphics

Chichi Francesco, Jary Pomponi

June 28, 2017

1 Introduction

The project developed from us is a local multi player game (1-4 players), inspired to the film *Tron*.
The goal of each player is to eliminate all the other players, and to do so, he have to traps them in some walls. When a player bump a wall or another player, he dies. At this point, the animation of the death start, and when it's ended, the ship and the relative walls disappear from the map.

2 List of all the libraries, tools and models used in the project but not developed by the team

- **Three.js:** Three.js is a cross-browser JavaScript library/API used to create and display animated 3D computer graphics in a web browser exploiting the power of WebGL in an high level mode.
- **OrbitControls.js:** We used OrbitControls to allow the user to move the camera in the space.
- **Detector.js:** Detector is necessary to verify if the user's browser support *WebGL*.
- **stats.min.js:** stats.min is used to show the FPS, frame per second, or the MS, the millisecond used for each render.

3 Description of all the technical aspects of the project

3.1 Ship

Each player has a ship of the choosen colour, which one is a three.js Group, composed by a **cabin** (fig. 2), a clock wise rotating **ring** (fig. 3) and two **motors** (fig. 4).

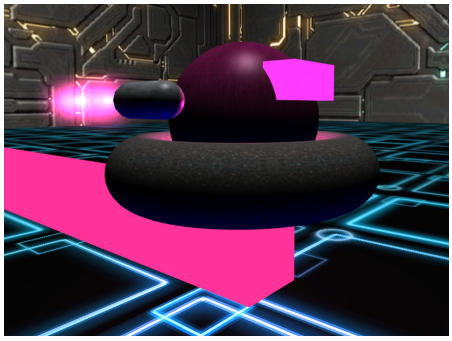

Figure 1: This is the character of a player that choosen the pink colour.

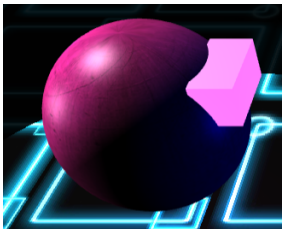

Figure 2: The cabin of the ship.

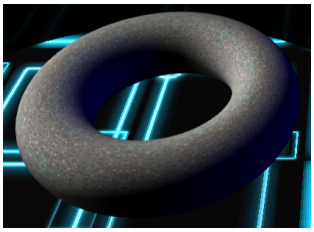

Figure 3: The rotating torus.

Figure 4: A motor with the particles.

The cabin is also a Group of element coloured with the player's colour, composed by a glass and a cockpit with a pointlight inside. The cockpit is a sphere covered with a metallic texture, the glass is a parallelepiped made by a *MeshToonMaterial*, that is illuminated by the cockpit's light.

Each motor is another independent Group, composed by a cylinder, an hemisphere on the top and another hemisphere with an hole, that represent the exhaust pipe, each one is covered with the metallic texture of the cabin. Furthermore, at the end of the exhaust pipe, there are 50 particles of the colour of the player. The wake's length where this particles are distributed is choosen stochastically: with a probability of 80%, the wake's length is equal to 40, with 10% is 0 or 10.

Finally, there is a rotating ring, that is a torus coloured with the same texture of the torus which rotates around the map.

3.2 Halo

The halo object is used to simulate the day-night cycle and it is implemented in the Halo.js file.
We have implemented the halo object with a group, which is composed by a torus and two spotlight, one for the sun and the other for the moon. Those spotlight are positioned in opposite part of the torus and both points to the centre of the game plane. The difference between the lights is the colour of the light. In order to simulate the cycle we

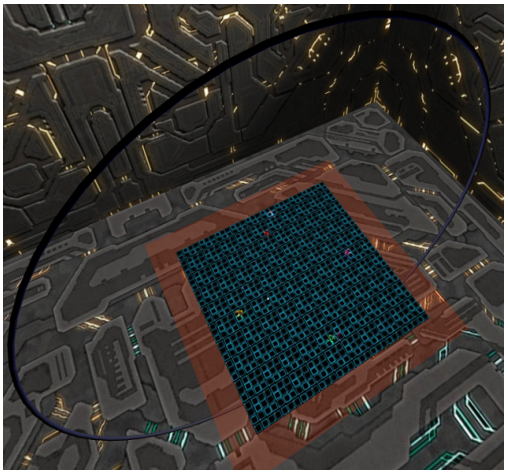

Figure 5: Day modality

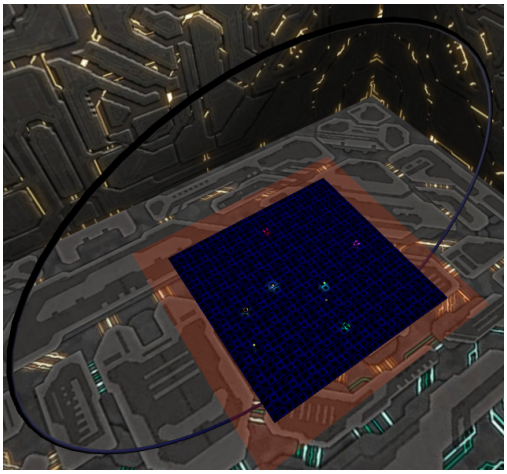

Figure 6: Night modality

used a rander function, that will rotate the torus by 0.5 degrees each frame.
The modality of the Halo can be choose in the main menu; we have:

- day: the torus starts in the day configuration and won't rotate.
- night: the torus starts in the day configuration and won't rotate.
- cycle: day: the torus starts in the day configuration and will rotate.

The differences between day and night can be seen in the figure above.

3.3 Animated Light

An animated Light is a sphere that emits light and move in the space. This is implemented in the animatedLight.js file.
Each light is a point light, and is constructed by generating a set of random points in the space, delimited by the plane space, and a random colour. Then those points are interpolated using a closed loop CatmullRomCurve3, from THREE. Since the position of the light in the curve is defined in the interval from 0 and 1, we increment the position, at each frame, with a small delta, in order to simulate the movement . This is done is the render function.

3.4 Plane and skybox

The plane is a transparent square, in which the ships can move; the material is a Phong type. Below the plane we have a red transparent flat square, that indicates the zone in which the ships cannot go. The material of the red square is a basic mash material.

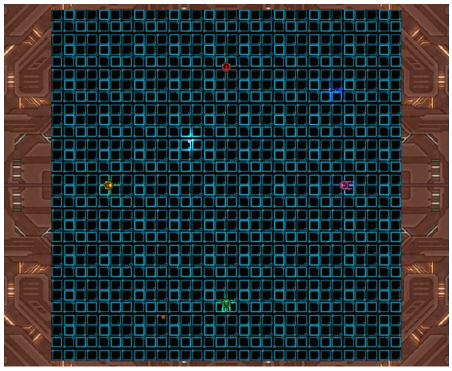

Figure 8: the plane and the red square

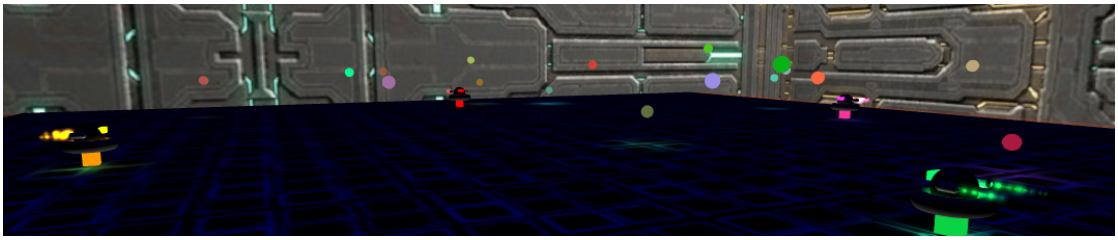

Figure 7: some animated lights

The skybox is simply the composition of two texture, one repeated two time, for the bottom and the top, and another repeated and rotated four time. Those texture composed form the skybox cube.

4 Game Logic and rendering

In this section we will analyse the game logic, the render part of the game and the animations of the ship.

4.1 Main loop and render

The main loop is the *animate* function.
If the game is running we check for collisions and see if the game ending conditions are met. Respectively using the *collision* function and the *checkEnd*. The collision logic will be explained later. In the *checkEnd* we also move the players; the movement of the player will be explained in *animations and player movement*.
If the game is on pause we check for every button is pressed, and based on the state of the game we display some menu; the menu section will cover this part. Then, depending on the active scene, the game or the menu, the function will call a different render function. In the render of the menu we rotate the ships, update theirs particle motors and modify the colour of each ship, depending on the chosen one, while in the render of the game we move the ships, the animated lights and rotate the halo, if the cycle is selected as modality of the light. Moreover we set the camera to follow the player if the single player mode is selected. In the render of the game we also animate the ships; the animation *animations and player movement*.

4.1.1 Collision

The collision are implemented using the Box3 class from Three.js. For each ship we compute te bounding box of the cabin, and for each wall we compute the bounding box of the entire wall segment. Each player control contains the box of the cabin and an array with the box of the players walls. The collision check consist of some basic controls inside some loops.
If the player cross the plane boundary then will die. Elsewhere, for each player, we iterate over all the walls and over all others player's ship box. If the collision test with walls will result positive then the player will die, but if the player collides with another player both will die. If a player is dead there is no reason to check collision.

4.2 animations and player movement

Each ship is animated in the render function, calling the render function in the player.js file. The animation consists in moving the ship on y axis following a sin function and rotating the torus by certain amount of degrees. Moreover all the particles of the motors will be updated.
In this file we check if the player is dead or not. If it is dead then the explosion animation will start. The explosion animation consist on a certain amount of particles (sprites) and an animation of the ship.Each particle will move from the explosion point to a random

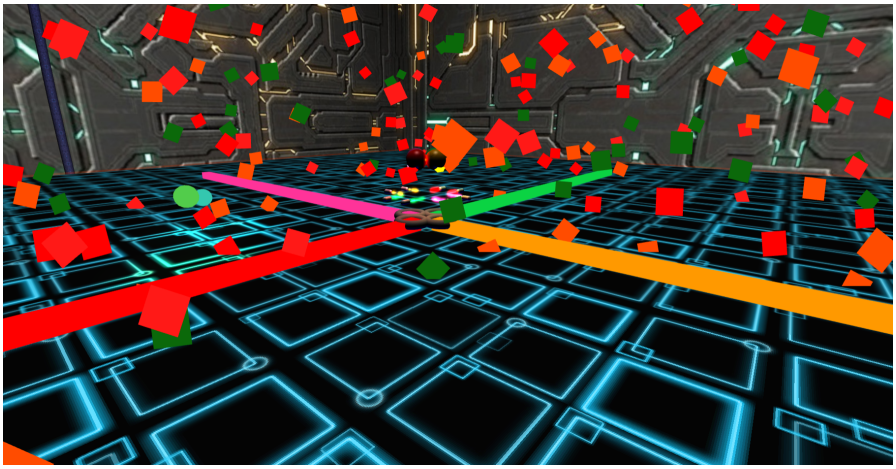

Figure 9: Collision between 4 ships

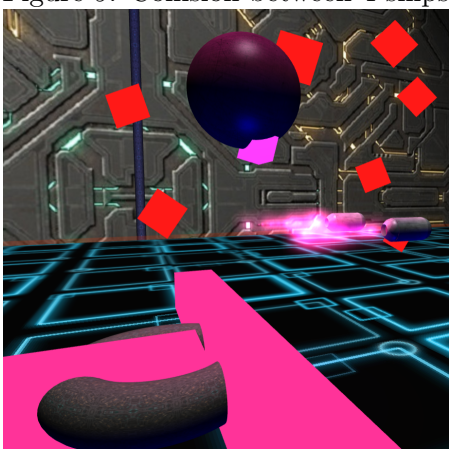

Figure 10: Animation of the explosion

point in the plane, with a parabolic movement, using CatmullRomCurve3 from *Three.js*. Like in the case of the animated lights, the curve position is defined from 0 to 1. So each frame start at 0 position, and when reach the position 1 will be deleted. In the animation of the ship the motors will continues to go straight ahead, the torus will fall on the ground and the cabin with the glass will rotate and going up. Meanwhile the ship and the will shrink.
When the animation is over the ship and the walls will disappear from the scene.
Since the ships can move only on two coordinates, and rotate only by 90 degrees, we implemented the movement with a unitary vector of dimension 3, in which only one value is set to 1 or -1, and the other are set to 0. This value indicates the direction of the ship respect to the centre of the plane.
This vector is used also to generate the walls, from the last point in which the ship

turned to the current point.

4.3 Menu

The game contains several menu views: The **main menu** (fig. 15), **key settings** (fig. 14), **colour settings** (fig. 16), **end game** (fig. 11,12,13), **pause** (fig. 17).

The main menu allow the users to start the game, to go in the key settings menu or to the colour settings menu. Furthermore, it allow to change the principal settings of the game: choose the number of playing players, decide if allow the main soundtrack and the in-game sound effect, only the last one or to play without any kind of sounds, and finally to choose between the three light modalities, **day**, **night** or **cycle**.
Next to the main menu, there are a view of all the 4 ships, below there is a pillar representing the ranking position of the relative player (fig. 18). Furthermore, the first ranked player is lit up by a spotlight.

The key settings menu is used in order to modify the default value of the command for each player. In the menu is shown, fore each player, the relative number of the player with the choosen colour and two button, usable to change the *turn-left* and *turn-right* commands values.

In the colour setting menu we can find three slider fore each player, relative to the RGB colour, and a bar coloured with the result of the three component.
On the left of this menu are showed the four ships with their choosen relative colors.
When only one player still alive, if the initial number of player was at least 2, the end game menu is showed, in which we find a table showing the ranking of the players in game. In this table there are the numbers relative of each player, if the colour is grey, it means that the relative player is not playing, otherwise are shown the colours relative to the player. Furthermore, in the same menu are shown two button, one to come back to main menu, other to play again the game.

Finally there is a pause menu, where we can find three buttons: the same two button described in the end game menu and another one to show the key settings menu.

Figure 11: with 2 player

Figure 12: with 3 player

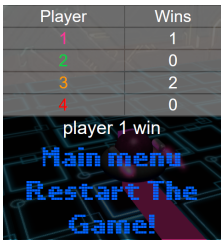

Figure 13: with 4 player

Figure 14: This is the menu used to change the default controls value.

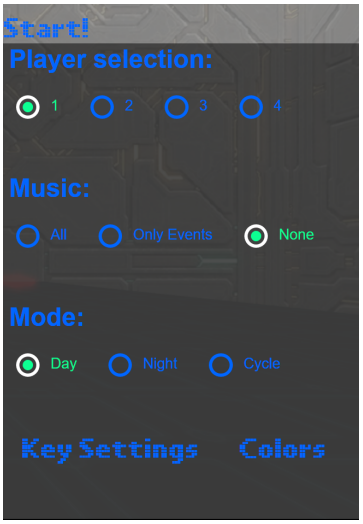

Figure 15: This is the main menu, used to start the game or to change some of the principal settings.

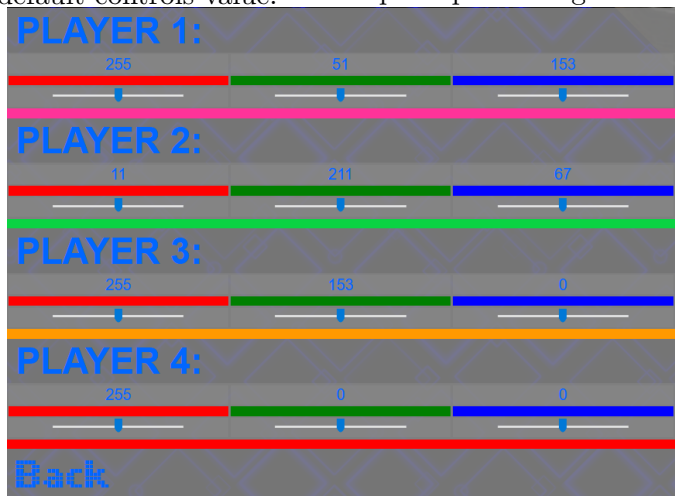

Figure 16: This is the menu used to change the default colours value.

Figure 17: This is the pause menu

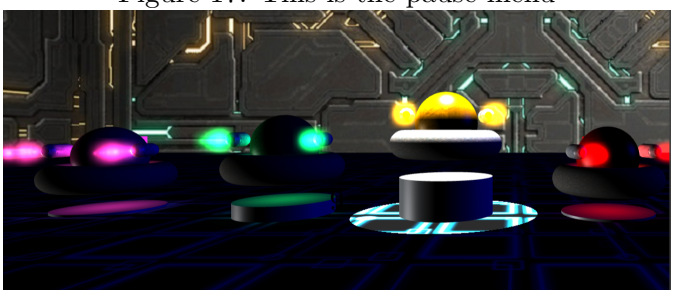

Figure 18: This is the ranking shown in the main menu.