MASTER OF COMPUTER SCIENCE IN ENGINEERING IN COMPUTER SCIENCE

INTERACTIVE GRAPHICS

A.Y. 2020/2021

POLLUTION BATTLE

Author:
Davidde Francesca 1792368

SAPIENZA
UNIVERSITÀ DI ROMA

Contents

1 Introduction

1.1 Basic Idea

Pollution Battle is a game whose purpose is to destroy a certain number of spheres depending on the difficulty level and the level of play (there are two levels). The spheres represent the enemies and so the pollution factors for the environment. The aim is to complete the two levels.

1.2 Rules

Cleaner is the main character of the game and he is a robot.

Figure 1 Cleaner

The robot *Cleaner*, which uses a hoverboard scooter to move, is intended to destroy a certain number of spheres, depending on the level and degree of difficulty selected in the main menu, to be able to either win or move to the next level. As we can see in the figure below, the difficulty level can be chosen through the radiobutton in the menu page:

Figure 2 Difficulty selection

The spheres are positioned on the pitch and, when the game starts, they follow *Cleaner* to reach and defeat it. If one of the spheres touches him, then he's lost. The difficulty level selected in the main menu has an impact on:

- The speed of the spheres;
- The range of the bullet fired by *Cleaner*;
- The number of spheres to shoot.

We have three difficulty levels:

- Easy;
- Medium;
- Hard.

Now let's see the differences between these three difficulty levels:

- Easy: *Cleaner* must shoot 2 enemies out of 6 to pass the first level (the score is shown on *Cleaner*'s back, so that the player can always figure out the current status). Then, if he reaches the second level, he must shoot 4 enemies to win the game. Moreover, the second level presents other differences compared to the first level:
  - The range covered by the bullet is shorter than the first level;
  - The enemies approach the robot at a faster speed than the first level;
  - The number of spheres to defeat;
  - The emissive light is lower (we will see it in detail in the light chapter).

```
if(difficulty == 0){
    winCountLVL1 = 2;
    winCountLVL2 = 4;
    bulletsRange = 4500;
    bulletsRate = 400;
    sphspeed = 0.3;
```

Figure 3 Easy level

- Medium: *Cleaner* must shoot 3 enemies out of 8 to pass the first level (the score is shown on *Cleaner*'s back, so that the player can always figure out the current status). Then, if he reaches the second level, he must shoot 5 enemies to win the game. Moreover, the second level presents other differences compared to the first level:
  - The range covered by the bullet is shorter than the first level;
  - The enemies approach the robot at a faster speed than the first level;
  - The number of spheres to defeat;
  - The emissive light is lower (we will see it in detail in the light chapter).

```
else if(difficulty == 1){
  winCountLVL1 = 3;
  winCountLVL2 = 5;
  bulletsRange = 4000;
  bulletsRate = 400;
  sphspeed = 0.4;
```

Figure 4 Medium level

- Hard: *Cleaner* must shoot 4 enemies out of 10 to pass the first level (the score is shown on *Cleaner*'s back, so that the player can always figure out the current status). Then, if he reaches the second level, he must shoot 6 enemies to win the game. Moreover, the second level presents other differences compared to the first level:
  - The range covered by the bullet is shorter than the first level;
  - The enemies approach the robot at a faster speed than the first level;

- The number of spheres to defeat;
- The emissive light is lower (we will see it in detail in the light chapter).

```
else if(difficulty == 2){
  winCountLVL1 = 4;
  winCountLVL2 = 6;
  bulletsRange = 3500;
  bulletsRate = 400;
  sphspeed = 0.5;
```

Figure 5 Hard level

1.3 Commands

This is a computer game. Indeed, the commands to use are keyboard keys:

- W: the character moves forward;
- S: the character moves backward;
- A: the character moves left;
- D: the character moves right;
- Space bar: the character shoots bullet.

Since WASD are close together, the player can move and shoot with only one hand. So, with the other hand, he can she can use the mouse to change the camera's view.

As we can see in the figure below, all these commands are widely described in the Commad page of the game:

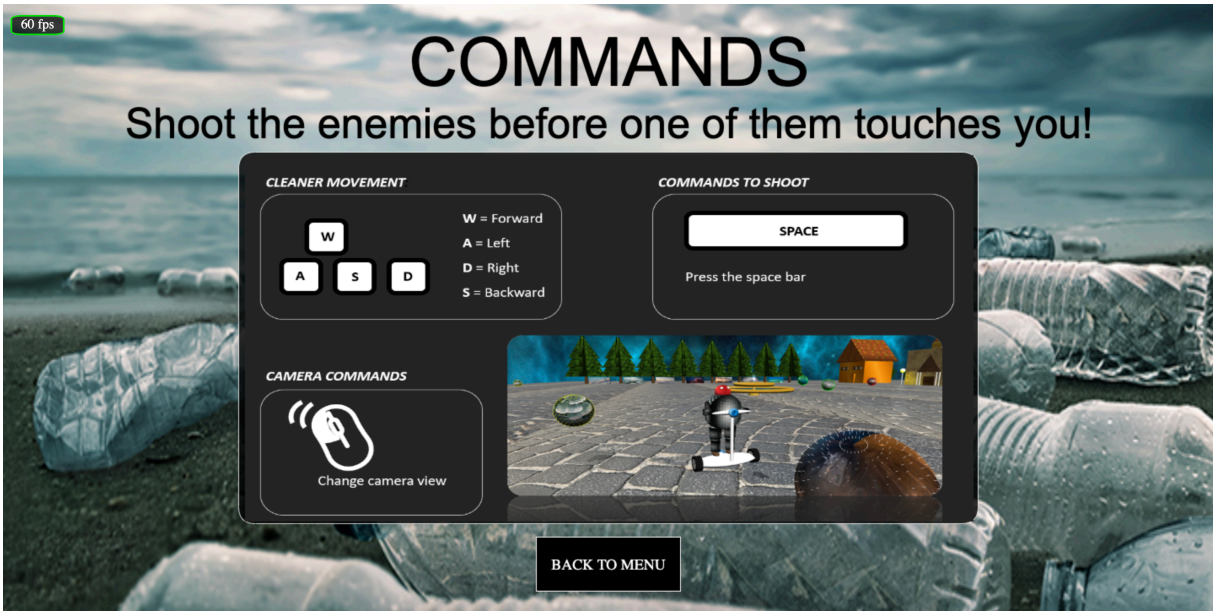

Figure 6 Commands page

2 GUI

Babylon.GUI has been used to create an interactive user interface for the scenes.

2.1 Menu GUI

When we launch the game, the main Menu is presented as shown:

Figure 7 Menu page

In this page the player can choose the difficulty level (the default value is Medium). Moreover, he can click on the "Commands" button to read the rules and command of the game. Then, once the user is ready to play the game, he has to click on the button whose label is "Start battle". In addition, he can also click on the "About the author" button to read infos about the author.

2.2 Commands GUI

When the user clicks on the "Commands" button, he is directed to the Command scene. Here he will find the rules and instructions needed to play the game. Let's see the Commands GUI:

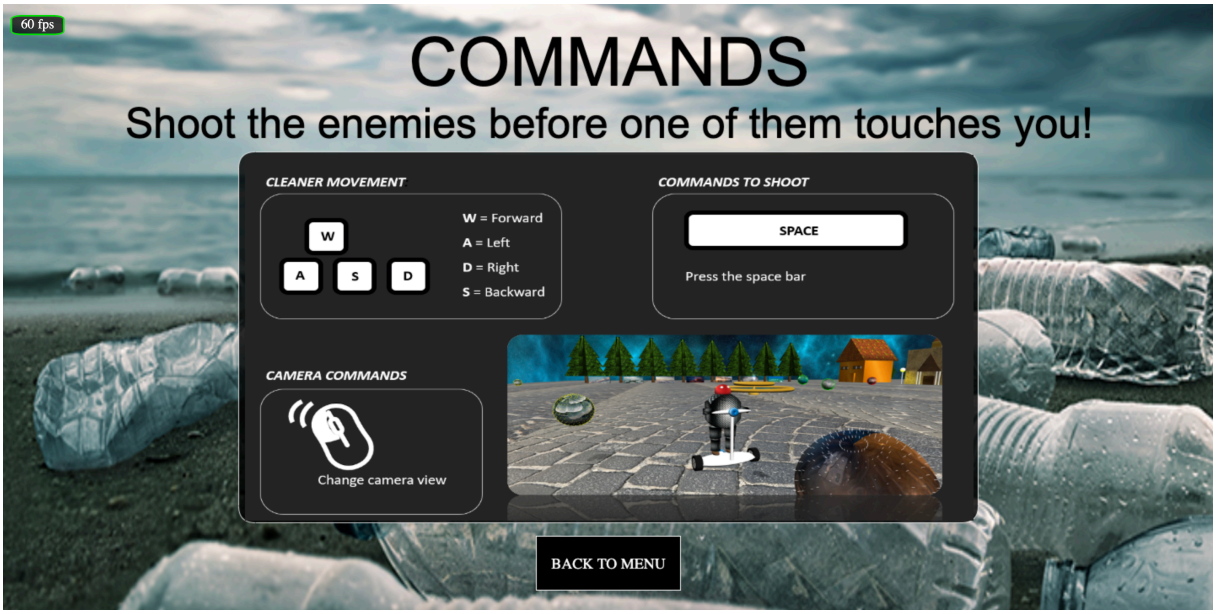

Figure 8 Commands page

2.3 About The Author GUI

When the user clicks on the "About the author" button, he is directed to the About scene. Here he will find more information about the author. Let's see the GUI:

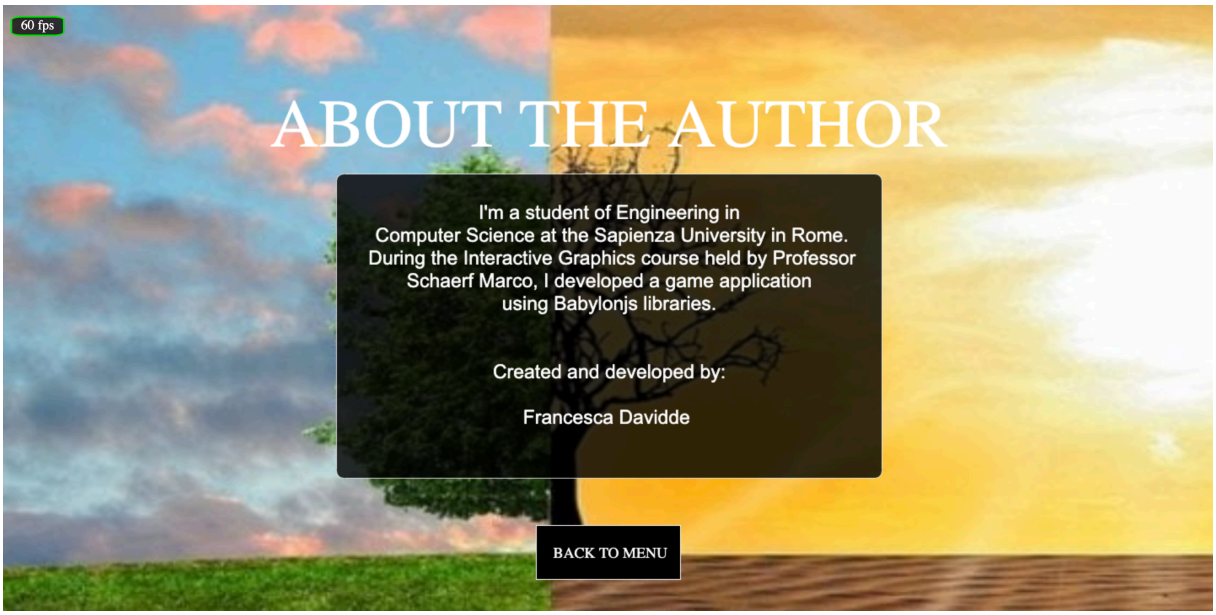

Figure 9 About page

2.4 Game GUI

When the user clicks on “Start battle”, the game begins. The player has a certain number of spheres to knock down depending on the difficulty level. This number, along with the number of those already destroyed, is shown for each level on the body of *Cleaner*.

Figure 10 Game GUI

When the first level is completed, there is a “Level up” image with the victory sound.

Figure 11 Level up

Now the second level can start and it is presented with the two lamps on, the windows of the houses lit, and with a lower emissive light until it becomes dark (we

will see this feature in detail later). The hoverboard wheels are yellow and the handlebar will light up to illuminate the scene before getting to a complete dark:

Figure 12 Second level game

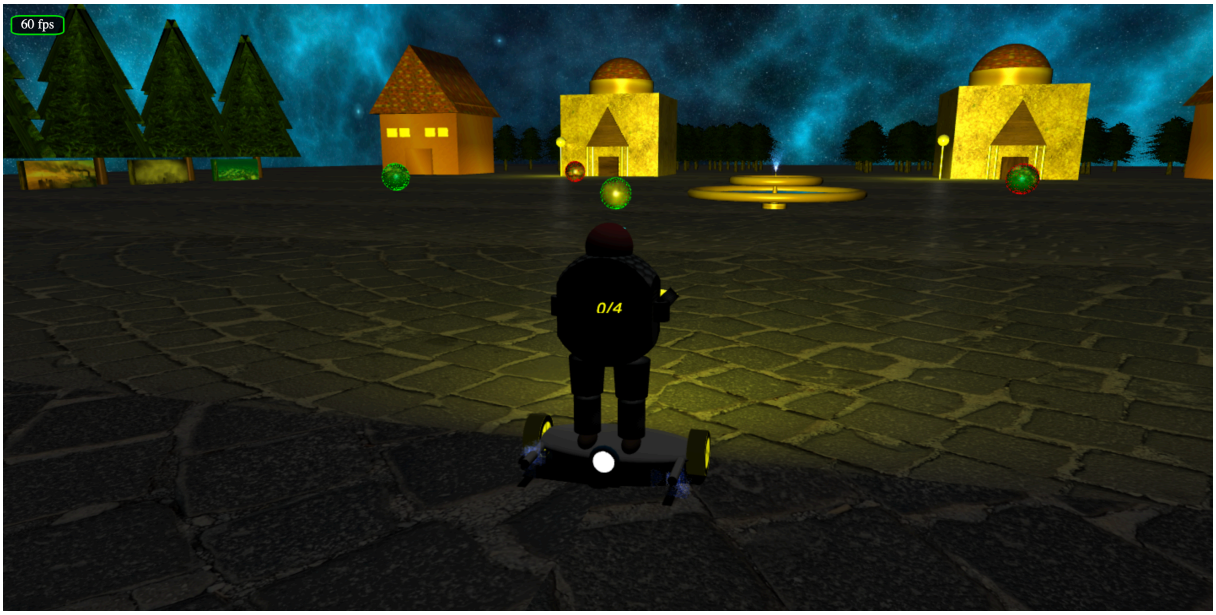

Figure 13 Second level game 2

2.5 Win GUI

When the player completes the two levels, he wins the game and so the following page is shown (accompanied with the victory sound):

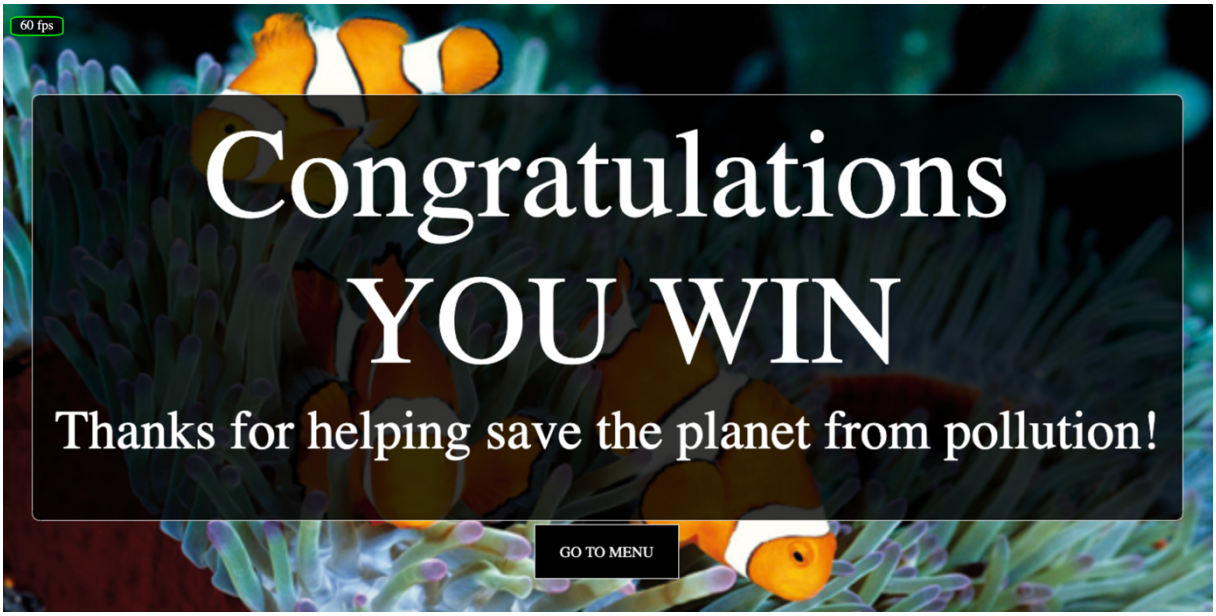

Figure 14 Win GUI

2.6 Game Over GUI

When the player loses the game (so he has been touched by a sphere), the following page is shown (accompanied with the game over sound):

Figure 15 Game Over GUI

3 Environments

All the static scenes (main Menu, Commands, About the author, Win, Game Over) have the UniversalCamera as a camera (on which the GUI is dislayed).

3.1 Game Lights

In the first level I have used a DirectionalLight and an HemisphericLight as shown:

```
const light = new BABYLON.HemisphericLight("light", new BABYLON.Vector3(0, 100, 0),scene);
light.position = new BABYLON.Vector3(-3800, 100, -200);
light.diffuse = new BABYLON.Color3(1, 1, 1);
light.specular = new BABYLON.Color3(0, 0, 0);
light.intensity = 0.8;
light.setEnabled(false);



const light2 = new BABYLON.DirectionalLight("light2", new BABYLON.Vector3(0, -1, 1),scene);
light2.position = new BABYLON.Vector3(0, 100, -200);
light2.diffuse = new BABYLON.Color3(1, 1, 1);
light2.specular = new BABYLON.Color3(0, 0, 0);
light2.intensity = 0.85;
light2.setEnabled(false);
```

Figure 16 Light first level

In the second level I have used a DirectionalLight with an intensity of 0.5 slowly trending to 0 (complete dark):

```
const light1 = new BABYLON.DirectionalLight("light1", new BABYLON.Vector3(0, -1, 0.2),scene);
light1.position = new BABYLON.Vector3(-3800, 100, -200);
light1.diffuse = new BABYLON.Color3(1, 1, 1);
light1.specular = new BABYLON.Color3(0, 0, 0);
light1.intensity = 0.5;
light1.setEnabled(false);
```

Figure 17 Light level 2

In the second level, the wheels are yellow, the lights of the windows of the houses are on, the handlebar of the hoverboard lights up when the intensity of the external light is less than 0.30.

I used three spotlights:

- Two for the streets lamps;
- One for the handlebar of the overboard.

Figure 18 Spotlight Example

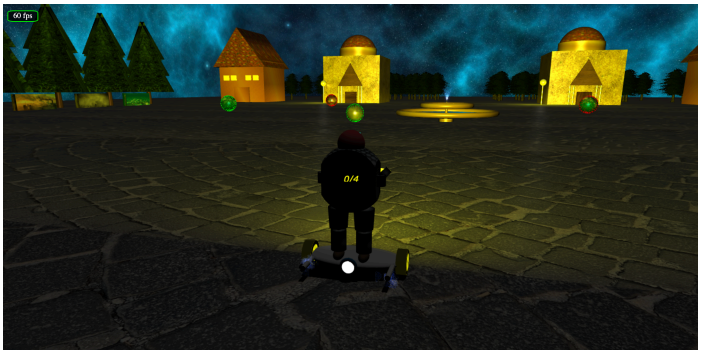

Figure 19 Level two dark

Figure 20 Level two dark 2

3.2 Game Camera

I used a FollowCamera. So I gave a start position, a target to follow and a goal position from which to view the target.

```
var camera = new BABYLON.FollowCamera("FollowCam", new BABYLON.Vector3(-40, 10, 0), scene);
 camera.radius = 150;
 camera.heightOffset = 50;
 camera.rotationOffset = 270;
 camera.cameraAcceleration = 0.1;
 camera.maxCameraSpeed = 100;
 camera.lowerRadiusLimit = 50;
 camera.upperRadiusLimit = 190;
 camera.lowerHeightOffsetLimit = 0;
 camera.upperHeightOffsetLimit = 180;
 camera.lowerRotationOffsetLimit = 10;
 camera.upperRotationOffsetLimit = 320;
 camera.minZ = 0;
 camera.attachControl(canvas, true);
 camera.checkCollisions = true;
```

Figure 21 Follow camera

```
camera.lockedTarget = cleaner;
```

3.3 Skybox and ground

I applied a skybox to simulate the sky, so I used six suitable images (px, nx, py, ny, pz, nz). Then, I built a ground simulating the cobblestones. The playing field is reminiscent of Piazza del Popolo in Rome.

3.4 Game Textures

I used many textures to make the game more realistic. Let's see how the textures has been used:

- Ground: cobblestones texture;
- Spheres: since they represent the enemies, I used images of pollution for them;
- Paintings: images of pollution;
- Trees: leaves effect;
- Houses: I used different textures to give the idea of brick house effect (for example I used different textures for the roof and the door);
- Churches: I used different textures to give the idea of marble effect (for example I used different textures for the dome and the columns);
- Wheels: I created a cylinder with a wheel texture;
- Fountain: it has a texture, but the interesting thinks is that it reflect the sky, it simulates the movement of water, etc.. (we will see it later);
- Robot: I used many textures to give the idea of steel effect.

4 Imported Meshes

4.1 Tree (behind houses and churches)

In the game scene I used two types of trees. One of these two models, located behind the houses and churches, has been imported from Babylon.js. Then, I created instances to increase the number of trees.

Figure 22 Imported mesh

5 Hierarchical Models

The whole scenography has been created by constructing hierarchical structures.

5.1 Cleaner robot

In order to build the *Cleaner* robot, I built different meshes connected through the parent attribute. I used the CreateSphere command for:

- Head (whose parent is the body called Cleaner);
- Eyes (whose parent is Head);
- Knees (whose parent is Thigh);
- Feet (whose parent is Leg).

I used the CreateCylinder command for:

- Thighs (whose parent is the body called Cleaner);
- Legs (whose parent is Knee);
- Arms (whose parent is the body called Cleaner).

I used the CreateCapsule command for:

- Hand (whose parent is Arm).

I used the CreateBox command for:

- Cleaner (the body).

Figure 23 Cleaner and Hoverboard

5.2 Hoverboard

I used the CreateSphere command for:

- Base of the hoverboard (whose parent is Cleaner);
- Handlebar (whose parent is the vertical pale): it includes also spotlight for the hoverboard's night light;
- Lamp in the back (whose parent is the hoverboard lamp support in the back).

Figure 24 Cleaner and Hoverboard

I used the CreateCylinder command for:

- Vertical pale (whose parent is the base of the hoverboard);
- Wheels (whose parent is the base of the hoverboard);
- Exhaust pipes (whose parent is the base of the hoverboard): they create the smoking effect thanks to the Particle System set up.

I used the CreateTorus command for:

- Hoverboard lamp support in the back (whose parent is the base of the hoverboard);
- Bullet exit hole (whose parent is the handlebar).

5.3 Fountain

I used the CreateCylinder command for:

- Vertical structure.

I used the CreateTorus command for:

- Circolar tubes.

I used the CreateSphere command for:

- Base of the circolar tubes.

Then, I created a Particle System to simulate the splashing water. Moreover, to make it more realistic, I used a texture for the water where I provided the wind force, the wind direction, the bump height, wave length and the wave height. All these characteristics has been included into an addToRenderList adding the SkyBox to the reflection and refraction.

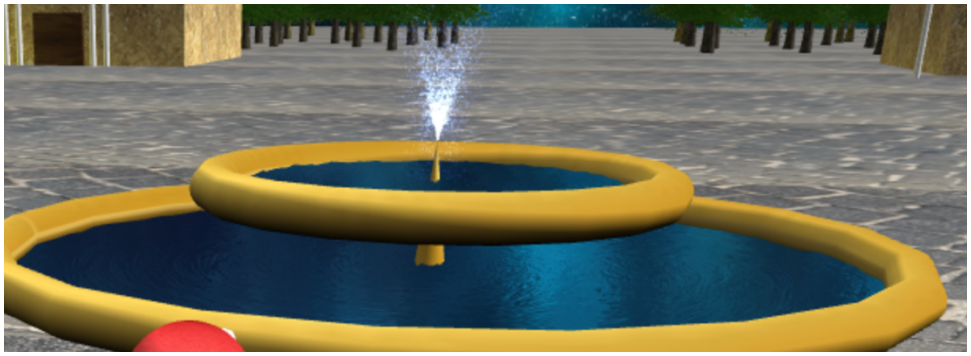

Figure 25 Fountain

5.4 Houses

I realized two houses. I created a function requiring as input the house position. From there I got the whole house. In order to build the house I used the CreateBox commads for:

- Windows;
- Door;
- House structure.

I used CreateCylinder for:

- Roof (using tessellation = 3).

Figure 26 House

5.5 Churches

I realized two churches.

I used CreateBox for:

- Church structure;
- Door.

I used CreateCylinder for:

- Cylinder on top of the roof;
- 8 columns;
- Triangle on top of the columns.

I used CreateSphere for:

- Dome.

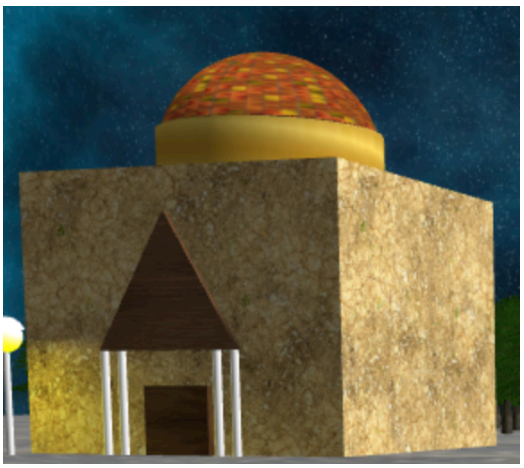

Figure 27 Church

5.6 Trees

I used CreateCylinder for:

- Bush (with tessellation = 3);
- Trunk.

The tree is structured in three levels and in order to make it more realistic, I had to intersect two triangles for each level of the tree. The resulting effect is the following:

Figure 28 Tree

5.7 Paintings

In order to make the game scene more realistic, I added some paintings representing the pollution. Moreover, the same images used as textures have been added also to the spheres enemies.
I created the paintings using the CreateBox command.

Figure 29 Trees and paintings

5.8 Spheres enemies

I created ten spheres representing pollutants and I added normalized facets of different colors. The spheres have been created using the CreateSphere command and then I also used the clone command to multiplicate the spheres created.

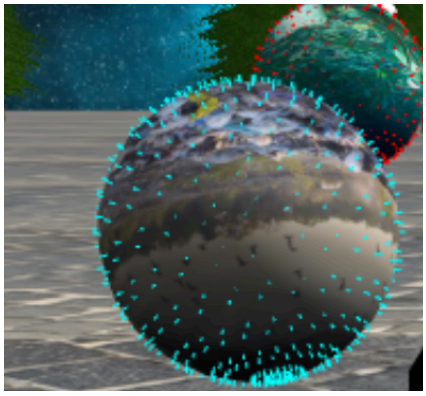

Figure 30 Sphere enemies

5.9 Lamps

I used the CreateSphere for:

- Bulb of the lamp;
- Bulb anchor (specifying slice = 0.4).

I used the CreateCylinder for the vertical pale.

Figure 31 Lamp

6 Animations

I have used movement and rotations within the render loop cycle in addition to using the Action Manager provided by BabylonJS.

6.1 Cleaner and hoverboard animation

Cleaner's animation takes place through the hoverboard that rotates the wheels through Y axis positively/negatively, depending on the direction. If the direction is forward than the rotation is positive, while if the direction is backward than the rotation is negative. The hoverboard can also turns left and right.
When the movement is forward, the light on the back of the hoverboard becomes blue. It turns red when the hoverboard goes backward. If the hoverboard is stopped, then the light is white.

6.2 Sphere enemies animation

The spheres rotate by a speed set based on the difficulty level and current level. They move toward Cleaner position (since they want to defeat him), computed in a render loop. They start moving from their own initial position, then once the game is started, they move toward Cleaner's position.

6.3 Bullets animation

The bullets move from the handlebar of the hoverboard toward a direction which is equal to Cleaner direction. When the bullet reaches the maximum range, then it disappears. Moreover, we have different ranges according to the difficulty level and current level. The harder the game, the more the range decreases.

7 Interactions

The interactions managed in the game are related to:

- Player-Enemies;
- Bullets-Enemies;
- Player-Ground;
- Player-Environment;

.

.

The interactions happen in two ways:

- Plugins for Physics engine;
- Meshes collision.

7.1 Physics interactions

I have chosen Cannon.js and PhysicImpostor provided by BabylonJS to enable the physics on an object. The Physics engine uses an Impostor, so it is a simpler representation of complex object. Cannon has many impostor shapes and I used box and spheres. The Physics has been applied for example to all the spheres enemies.

7.2 Mesh collision interactions

I used the mesh collision to make interaction between entities using the intersectMesh(). It takes in input two parameters:

- Mesh to be checked;
- Precision of intersection (false or true).

Since the check is done only once, I included it into a render loop. I used the mesh collision to detect when bullets intersect spheres and when spheres intersect Cleaner. Moreover, I applied mesh collision when Cleaner intersects:

- Fountain;
- Trees;
- Paintings;
- Lamps;
- Churches;
- Houses.

The check is performed in each direction in which Cleaner moves (so whether the character moves forward, or whether it moves backward or to the left or to the right).

8 Music and sounds

To make the video game more realistic, I added sounds and music. I added music in the main menu, which stops when the player leaves it. Then, I also added the sounds of the water, since I have a fountain in the center of the game scene. Moreover, other sounds are:

- Bullet shot;
- Winning;
- Defeat;
- Level up.

9 Libraries

To perform many functions, I have imported the following libraries of BabylonJS (which provides beautiful, powerful and simple web rendering engines):

- babylon.js
- babylon.objFileLoader.js
- babylonjs.loaders.js
- babylonjs.proceduralTextures.js
- babylon.viewer.js
- cannon.js
- babylon.grassProceduralTexture.js
- babylon.gui.js
- babylon.glTF1FileLoader.js
- babylon.glTFFileLoader.js
- babylon.objSerializer.js
- babylonjs.serializers.js
- babylonjs.materials.min.js
- babylon.inspector.bundle.js