Project Interactive Graphics

Paolo Caruso, Cristian Fioravanti

September 2023

1 Introduction

Our project is aimed at the creation of a game named: "Tank Wars".
It's an arcade game in which The player takes control of a tank within a rectangular arena inhabited by enemy tanks. The objective of the game is to defeat all enemy tanks which spawn in waves without taking too much damage.

1.1 Libraries and Engine used

In order to implement the project, we use:

- BabylonJS: an opensource library built on top of WebGL.
- Havok: a physic engine that give the support for physics simulation.

This configuration permits us to:

- Built easily the structure of the tank in a hierarchical way.
- Have an easy way to apply: texture, light and shadow.
- Have great performance without any type of slowdown.
- Have the possibility to handle easily the animation and the collision of the bullets.

2 User manual

In this section, we provide all the necessities needed to discover and enjoy the game.

2.1 Main page

The main page is composed of the presence of the main title and the button to start playing. The player has just to click on the button "Start" when he is ready to play. After that, the loading page is presented.

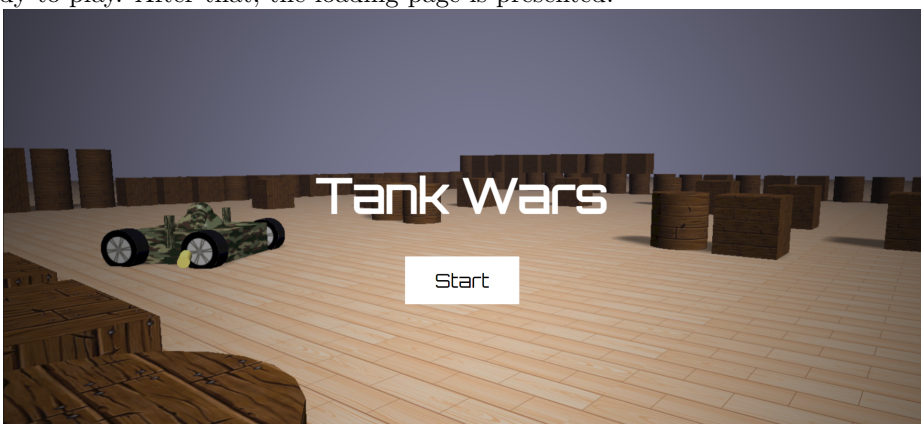

Figure 1: Main page

2.2 Loading page

The loading page, as the name suggests, starts the loading of the game and the environment and the song in background starts to play.

Figure 2: Loading page

2.3 GameOver

The gameover scene appears when the player tank has no more life.
On this page is visibile the final score of the player and the button "Restart" which permits them to start again the game.

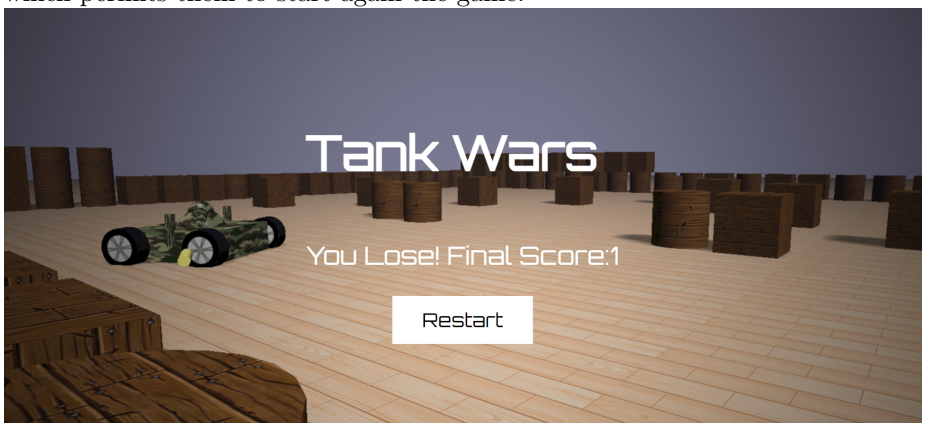

Figure 3: Gameover

3 The game

3.1 Controls

In order to move our tank the commands are:

- W: Allows to move the tank forward.
- A: Allows to move the tank left.
- S: Allows to move the tank backward.
- D: Allows to move the tank right.
- F / Mouse click: Allows to shoot with the tank.
- Space: Allows to brake the tank.
- Mouse position: Allows you to direct the tank's gun toward that position.

3.2 Gameplay

The arena plays in two dimensions within a three-dimensional landscape, consisting mainly of rectangular blocks. The gameplay involves maneuvering around the arena and attacking enemy tanks while avoiding incoming fire. Since the enemy tanks move around, combat between tanks often requires predictive aiming and angular judgment to strike moving targets. Although the playing field is two-dimensional, the vertical stacking of blocks can at times obscure visibility. Both the player and enemy tanks have a number of life points, a single hit point reduces that. If a tank takes enough damage that reduces to 0 the life points, it will be destroyed.
If the player destroys an enemy tank he will earn a point. If the player destroys all the enemy tanks of the wave, another wave of enemy tanks will spawn increasing the number of enemy tanks with respect to the previous wave up to a maximum of 4 enemy tanks when the score of 6 points is reached.
If the player's tank gets destroyed the game ends and the players can restart the game.

3.3 Game Arena

Tank Wars has a single map composed of ground which has a texture applied on it and cylindrical or cubic obstacles arranged along a rectangular perimeter that make up the boundaries of the map, after which there are other obstacles in the center that form a triangle and serve the player as areas where it is possible to take shelter from bullets.

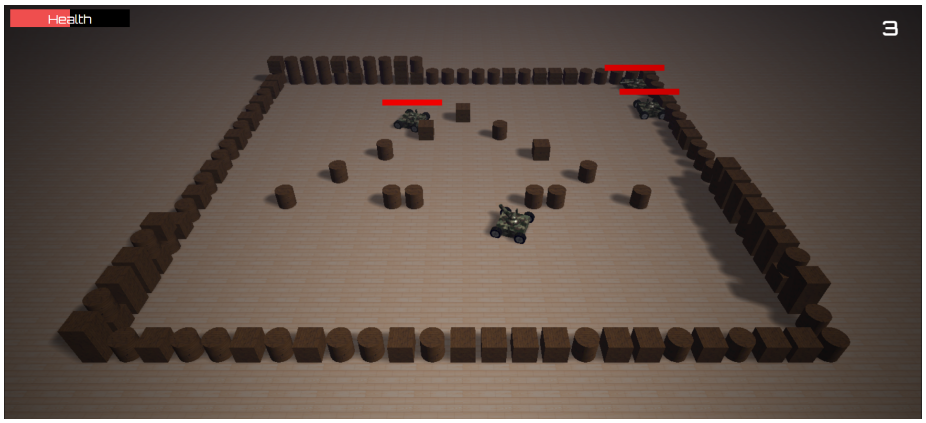

Figure 4: Game Arena

4 Models

4.1 Tank - Hierarchical model

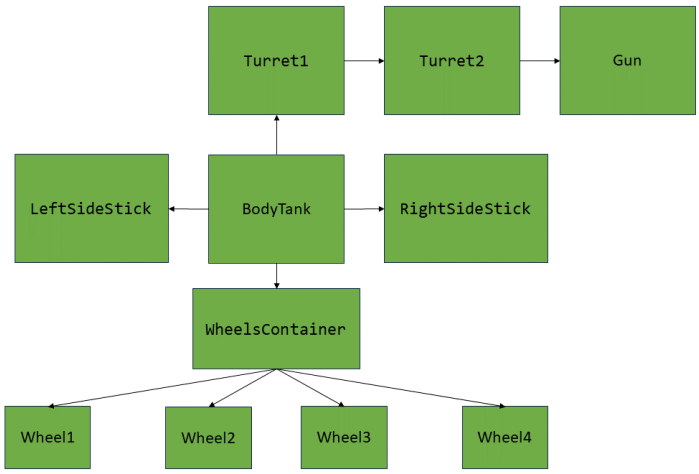

Figure 5: Tank Hierarchical Model

We define the structure of the tanks as a hierarchical model composed of different parts.
As we can see in the **Figure 3** the BodyTank is the root of the model, its children are the leftSideStick, RightSideStick, FirstCupola, and the wheelsContainer. The FirstCupola has its own child which is the SecondCupola part (that is a cylinder with a bigger diameter), it also has a child that is the Gun. The wheelsContainer contains 4 children that will represent the wheels of the tank, and they are: wheel1,wheel2,wheel3,wheel4.
We have also applied to the different parts of the vehicle some textures.

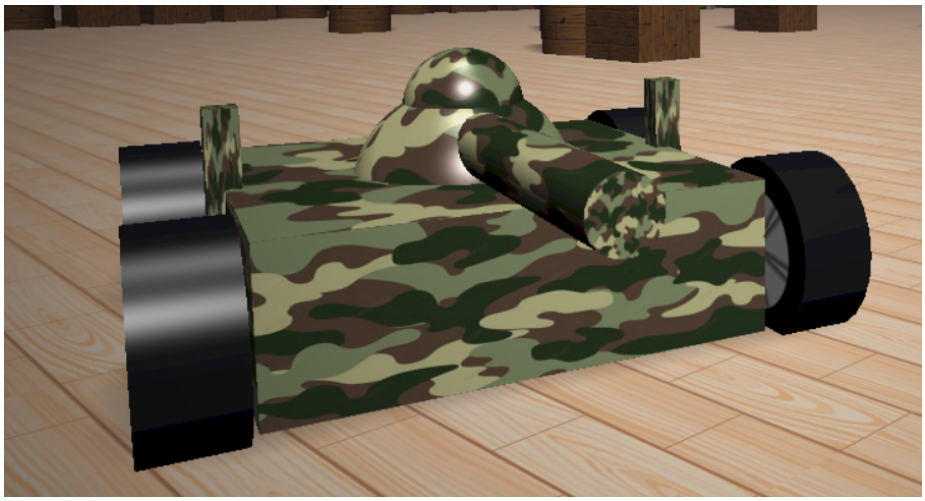

Figure 6: Tank

4.2 Obstacles

The obstacles in the map represent the border of the map and the wall in which the tank can hide itself in order to be protected against enemy bullets. To create this obstacle we used the functions of BabylonJS: CreateCylinder and CreateBox. Through the createMultipleObstacles function: a group of obstacles are created in series along the same direction, while thanks to the createObstacle function the obstacles are actually created and positioned. All the obstacles have a texture applied on them.

(a) Cube (b) Cylinder (c) Texture Used

Figure 7: Texture Images

5 Animations

For the animations we implemented our function in order to move the tank wherever possible and reproduce the shots from the tanks in particular the main used functions are: getAbsolutePosition(), setAngularVelocity(), getAngularVelocityToRef(), getLinearVelocityToRef(), setLinearVelocity(), applyImpulse(), getDirection(), setLinearVelocity(), getCollisionObservable(), toRotationMatrix().
With those functions we create our custom functions that are:

- createBullet(...): It permits the creation of the bullet in its correct position and to shoot it. After that it waits as soon as the projectile collides, when it happens the bullet is stopped and if the hit target is a tank 10 is taken from his total life. The function also applies, during the shoot, a recoil force to the tank to further stimulate the fired shot.
- rotateGun(...): It is responsible for calculating the rotation of a tank's turret so that it points towards a certain point. This function is also used to start a loop that will interpolate the rotation over time of the tank's gun creating a smooth animation of that movement.
- rotateTank(...): This function is used in order to apply to enemy tanks a specific rotation, this rotation permits to bring the body and the cannon of enemy tanks in the direction of player tank. Thanks to that, and the function rotateGun(...), when the enemy tanks are near the player they can start moving towards him in order to shoot.
- rotate(...): It permits the rotataion of the tank knowing the direction that has and its velocity.
- translate(...): Like the function described above knowing the direction that the tank has and its velocity permits to move it in the specified direction.
- applyBreak(...): The Function here apply a brake to the tank that slows down its progress.
- update(...): Based on the commands that the user executes and which are forced by the code itself, this function allows the rotation, translation, braking, creation and launch of the bullet to take place. It also takes care that the movement of the wheels complies with the movement of the tank and that the life of the tank is updated correctly.

6 Shadow

The BabyloJS "ShadowGenerator" class is used in the game to control how shadows are implemented. It is connected to a directional light that is present in the game space. Subsequently, the objects that will cast shadows on the other elements of the scene are defined, acting as "shadow casters". The other

elements in the scene will be covered in shadows from these objects. In the Tank class, for instance, elements like the tank's turret, wheels, and body are configured as shadow casters, allowing them to add to the overall lighting and shading of the scene.to make these shadows be projected the "receiveShadows" property has been enabled for the *tiledGround* mesh.