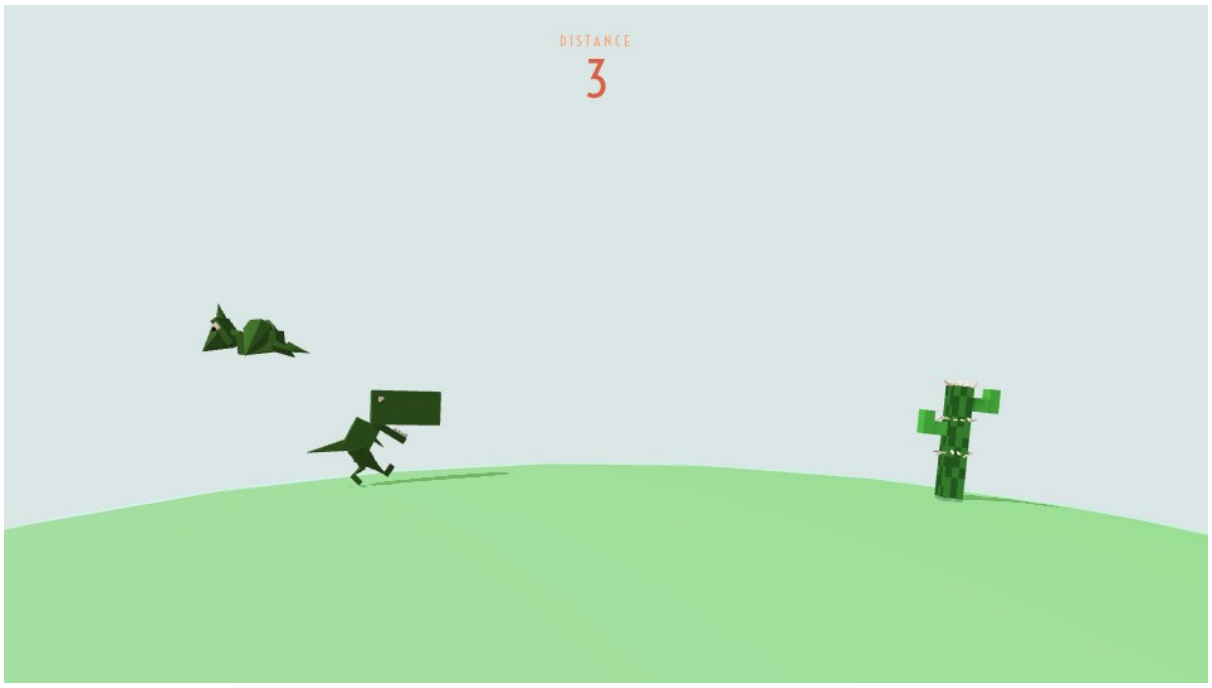
Interactive Graphics' project:

Dino Runner Game 3D



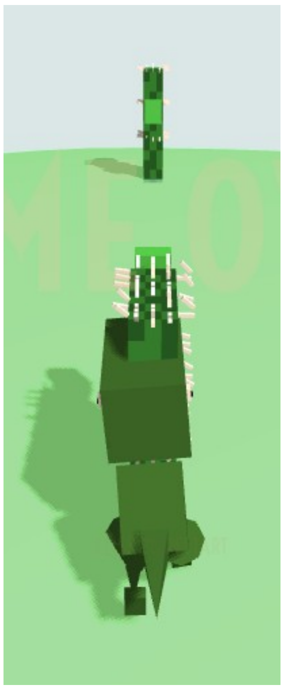
Raffaele Nuccio - 1534953
Erika Puglisi - 1601231

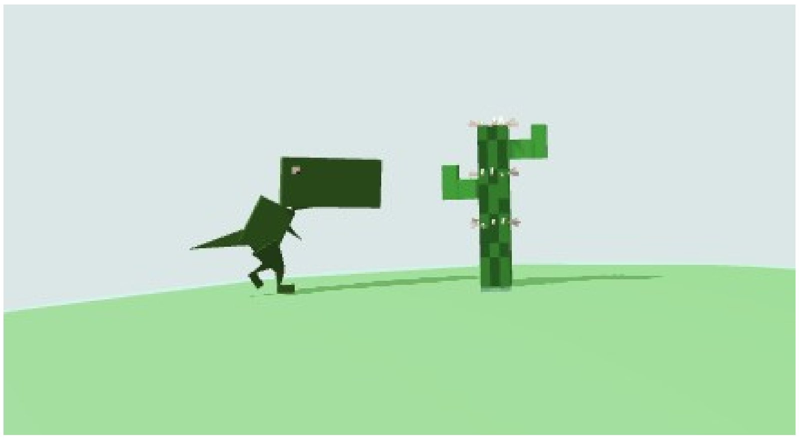
1. Introduction

The project’s purpose is to present a 3D version of the famous offline T-rex runner game created by Google. The game rules are not changed, the player has to manage a runner T-rex avoiding obstacles, like cactuses and pterodactyls, and trying to arrive as far as possible in the virtual world. To make the experience more pleasing different animations have been implemented, the running movement of the T-Rex as well as the flying movement of the pterodactyl have been created to be as natural as possible, but keeping a certain degree of imperfection for entertainment.

2. The world, the lights and the cameras

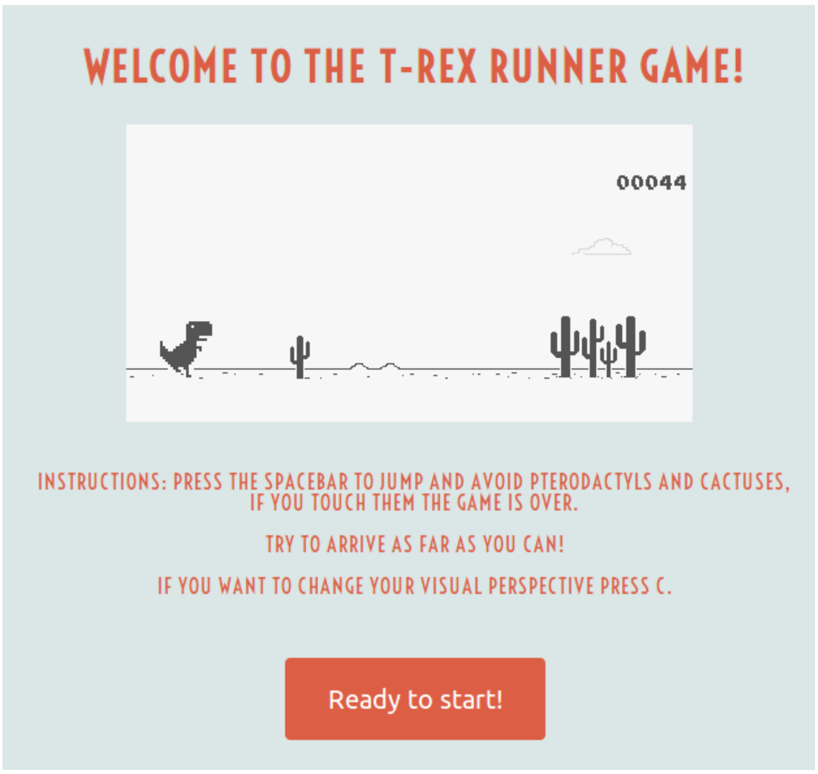
Following the idea of entertainment, the environment in which the T-Rex runs has been created as a sphere to better stage the idea of an infinite loop. The world is created starting by a main component (the floor) which includes a floorGrass sphere, which is the texture of the grass, and a floorShadow sphere, which contains the shadows of the objects. The main component and the floorShadow component rotate.
Two types of lights are in our world: an ambient global light and a directional light which creates the shadows of the objects (look at the createLights() function).
We also created two different cameras: the first one with the standard lateral view and the second one with a top centered point of view. The player can change camera when he wants pressing “c” on the keyboard, the switch between the cameras is managed in the render() function.
Also an onresize event was added in case the browser window has been resized.





3. The user interaction

We create an homepage for the game to explain the rules to the player, when he’s ready he can start the game pressing the button “Start”, which is controlled by the HTML DOM getElementById() method.
During the game, the player can change his point of view pressing ‘c’ (see the camera2 in the paragraph above) and can jump pressing the spacebar on the keyboard. If the game ends, he can restart the game simply clicking where he wants on the screen. The first two interactions are controlled by onkeyup events, while the game’s restart is controlled by an EventListener.
All these structures (labels, buttons, instructions, etc..) are implemented in the style.css file.

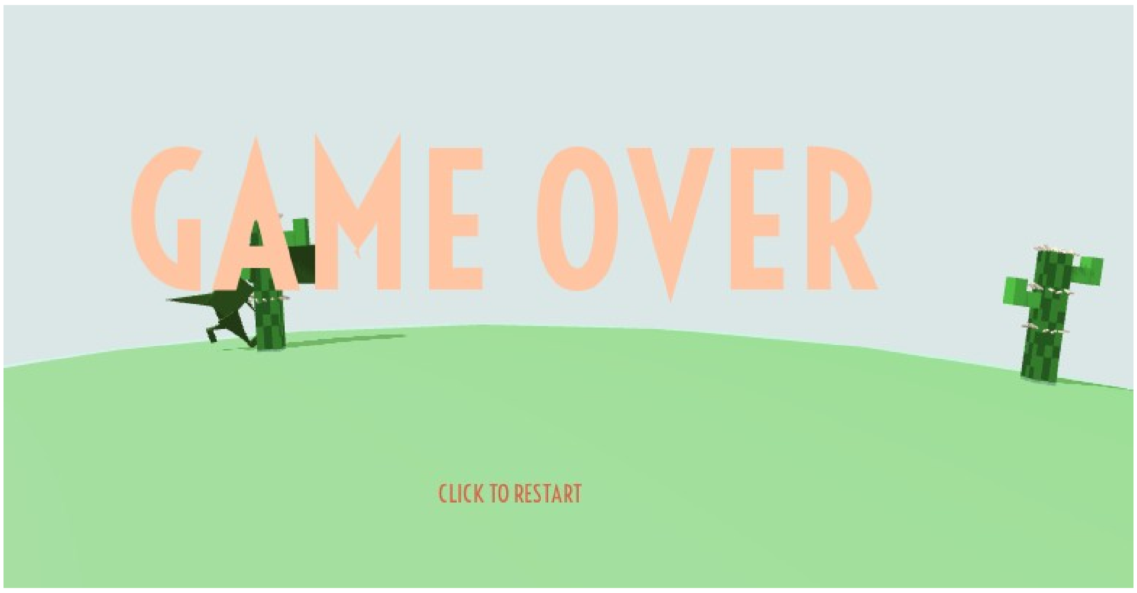


WELCOME TO THE T-REX RUNNER GAME!

INSTRUCTIONS: PRESS THE SPACEBAR TO JUMP AND AVOID PTERODACTYLS AND CACTUSES, IF YOU TOUCH THEM THE GAME IS OVER.

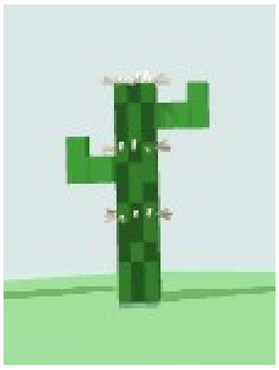
TRY TO ARRIVE AS FAR AS YOU CAN!

IF YOU WANT TO CHANGE YOUR VISUAL PERSPECTIVE PRESS C.



4. The hierarchical models

Every object inside the game is created as a hierarchical model, the dino and the pterodactyls are created starting from the torso, while the cactuses starting from the trunk.
In particular for the cactuses we create two different hierarchical models: a single cactus and a couple of cactuses, we took that decision because they take up a different amount of space inside the world and therefore represent a small difference of difficulty when jumped.
These two obstacles also have a checkerboard texture, with two shades of green, implemented in the main functions of the hierarchical models (CoupleCactus() and Cactus() functions).
The animations of the dino and the pterodactyls are created using the rotation function on the motion parts of the bodies, the velocity of the movements is controlled by the velocity of the whole game in relation to the level of complexity reached by the player.
During the jumps of the dino, his other movements are stopped.
The spawn of the obstacles is managed in that way: 30 random obstacles (decided by a random number, one for each type of object: one cactus, two cactuses and pterodactyl) are generated at the start of the game. The distance between the obstacles is composed by a fixed variable plus a random variable, anyway is calculated to leave enough space to make dino's jumps possible. When an obstacle reaches a certain position degree in the world's sphere, this object is substituted with another random obstacle (could be also the same type) and the random variable of the distance between obstacles changes (this part is in the updateObstaclesPosition() function).



5. The loop function and the update functions

The loop function is necessary to manage the movements of the hierarchical models, to update the position of the objects and the speed of the whole game. The level is updated after a certain time space sets by a TimeInterval (+1 level every 5000 milliseconds), the increase of the level involves an increase of the game's speed. When the game restarts, the level, the speed and the distance are re-initialized to the native values.
The updateFloorRotation() function updates the velocity of the world's sphere rotation in relation to the level reached by the player.
The updateDistance() function updates the distance reached by the dino using the speed and the seconds passed since the TimeInterval was set.
In the end, the checkCollision() function checks the dino position respect to the obstacles' positions, if a collision is detected, the game is over.

6. References

[1] Karim Maaloul WebSite:
https://codepen.io/Yakudoo/pen/YGxYZj

[2] WebSite of the course:
https://piazza.com/uniroma1.it/spring2018/1044398/resources

[3] Three.js documentation:
https://threejs.org/docs/

[4] Other documentation:
https://www.w3schools.com/