SAPIENZA
UNIVERSITÀ DI ROMA

Interactive Graphics

CAR TRAVEL

Professor:

Marco Schaerf

Students:

Roberto Adduci (ID: 1484232)
Luca Deodati (ID: 1488696)
Moreno Labbate (ID: 1532087)

Summary

Introduction

CAR TRAVEL is a WebGL infinite runner game in which the player drives a car (a Nissan GT-R Nismo) avoiding obstacles to obtain the highest possible score.
There aren't only negative obstacles, like rock and snowflakes, but also positive ones like hearts and coins. The hearts are useful to recover life points, coins are useful to unlock bonus like shield.
The rocks obviously take away life points if they are hit by the car, instead the snowflake doesn't allow the driver to change the position of the car by blocking its movements for 3 seconds.
The user can choose three different difficulties like easy, medium and hard and this influences the number of obstacles produced in the scene. Furthermore, the score obtained at the end of the game, i.e. when the player's life points end, will be multiplied by a coefficient that is higher the higher the difficulty chosen.

The Project

Structure

The project is structured as follows:

- Index.html file that represents the game menu in which the player can launch the game, can choose his username, can change the difficulty, can look of the record in that computer (via record.html file), can see a quick explanation of the emojis that are in the game and understand the controls of the games through another html file called controls.html.
- Record.html file that displays the different scores obtained by the player in a table sorted in a decrescent order.
- Control.html file that helps the user to understand the controls of the game.
- Game.html / Game.js that represent the real game.
- Related.css files that we develop to handle the style of html files.
- A folder, named "models", containing the JSON files for the geometries of the obstacles and the car.
- Various folder that contains images, textures, sounds, icons, fonts that contains elements useful for the project.

User Interaction

The user can interact with the game through a series of commands that are specified in the controls.html file, in fact there is a listener that catch the user interaction represented by the handleKeyDown(event) method.
It is possible change camera pressing the key C on the keyboard, pausing the game through the key P, and the player can handle the movement of the car on axe x with left and right arrow.
So, the hierarchical model is linked with the user especially in the above explained movements, but it has also independent animations that involve the wheels in their continuous rotation.
The user can add a child to the hierarchical model pressing the down arrow when he has enough money to unlock the shield, that obviously become a part of the model.
He can also decide to turn off the sound when he wants using the mouse on the related icon.

Code

Main functions

init() :

In this function we initialized the whole scene and we set the camera in the right position coherently with our idea of player's view in the context of the game. Here, we created the "sound object" that we use in the other functions, and the light. We put here the loading of the textures because they require more time than the other stuff.
We created different planes that represent the road and the guardrails.

$$\textbf{addX():}\quad \mathbf{X} \in \begin{cases} \textbf{Rock} \\ \textbf{Heart} \\ \textbf{Snow} \\ \textbf{Coin} \\ \textbf{Car} \end{cases}$$

We exploit these functions to load the model that represent our obstacles in the game as rock, heart, snowflake, coin and the main character of the game that is the car. We check the number of objects correctly loaded to understand when the render function can start without any problems. We also set the parameters to obtain a right view of the various objects.
In addCar() we also keep the references for the various part of the hierarchical object (car) that will be useful for the wheels movements and to change the color (after collision with a snowflake).

addObs() / chooseObs() / addObjToScene(obj)

The first function is in charge of the insertion of the obstacles in the scene. It consists of a "for" loop in which another function randomly chose the object to be loaded and then added to the scene through addObjToScene(obj). It puts the object in the road avoiding that object is in an already occupied position (working on axe z). The object array is sorted continuously to have in the last position the farthest object on axe z.
The above explained function inserts an object for each type and then, thanks to chooseObs(), fill the scene randomly choosing objects between coin, snowflake, heart and rock. The number of inserted objects depend on the difficulty chosen by the players. To avoid a trivial game, we augment the probability of inserting rock in the scene (probability > 50%) respect to the other obstacles.

animateObs()

The movements of the objects are linked with this function that moves the obstacles on axe z adding a factor that depends on the current game speed. When an object overcomes the camera, we are sure that it is the nearest to the player, so following our representation of the obstacles array, it is the one in the first position. For this reason, we remove the first entry of the array and we put it in the last position because it will be the farthest in the scene. We compare the obstacle position and the car position to understand when the collision happens working only on two axes (z and x).

handleCollisions(code)

This function triggers different behaviors depending on the objects that the car impacts thanks to a code that is defined at creation time of the object.

- If the car hits a rock the player loses life points (the progress bar changes aspect), a sad emoji appears on the screen and a sound is launched;
- If the car hits a heart the player recovers life points, a "fall in love" emoji is loaded, and another sound is launched;
- If the car hits a coin the player augments his portfolio, and this can be useful to unlock the shield;
- Finally, if the car hits a snowflake the car movements are blocked for 3 seconds, and the car changes color seeming frozen.

The maluses are avoided when the player uses the shield.

levelup()

This function is launched each ten seconds, in the render function, and increases the speed of the car by 10% compared to the current speed, and the obstacles speed by 20%. Then it makes visible an alert on the screen. This alert lasts 1 seconds.

gameOver()

This one is launched when the player loses all the life points and the game ends. It displays the game over alert and the score. The score depends on the difficulty chosen by the player at the start of the game. It allows to restart the game or to go back to main menu. Furthermore, it can understand if the result is a "normal" score or a new personal record working on "localStorage" variables.

handleKeyDown(keyEvent)

Here, we manage the different behaviors to be implemented because of the pressure of the keys by the player as follow:

- Left arrow: It moves the car in the left changing its position on axe x and move the wheels working on the rotation angle on axe y. After a very short interval it restores the initial position of the wheels thanks to an auxiliary function.
- Right arrow: It is the same of the left arrow, but with opposite values.
- Down arrow: It activates the shield for 5 seconds only if the player has the right amount of coins to unlock it adding another child in the hierarchical model of the car.
- P: It stops the render function (the sounds and the various timers) if it is not already pressed, otherwise it resumes the render function and the other stuffs previously stopped.
- C: It changes the position of the camera zooming on the car or it restores the initial view of the game.

Other functions

We use other functions to handle the images, the sounds and the various timers used in our game. These functions are sound(src), handleImg(img), removeImg(name, bool), handleSound(name, time), Timer(callback, delay), chronometer().

References

- https://threejs.org/docs/
- https://threejs.org/editor/
- https://clara.io/
- http://texturelib.com/
- https://www.bestcssbuttongenerator.com
- https://github.com/mrdoob/stats.js
- https://www.w3schools.com/html/html5_webstorage.asp