Interactive Graphics Project - Report

Argenziano Francesco 1957976 Bartoli Ermanno 1794103

Academic Year 2020-2021

Contents

1 Game Description

We've chosen as final project to implement a simple game. The player will command a car that will race in a desertic scenario, having as a goal to go as far as he can while succesfully avoiding the other vehicles. When the starting page is loaded, the player can choose to consult the guide and/or the difficulty, and then start the game.
As soon as the player press Enter, the car will be sent on the road and the game will begin. On the top right corner of the screen it's shown the current score of the player, that increases as the player survives. On the top left of the screen there are the current lives of the player: everytime the car hits another vehicle, it loses a life and everytime the car picks up a gas tank, it will gain one life back, with the cap at 3.
If the player loses all his lives, the game ends and the "Game Over" end screen will appear, reporting the score obtained during the run. By clicking again Enter, the game will restart and the player can play again and try to improve his record.

2 Framework

The two main libraries that we have used in order to develop this project are **ThreeJS** and **TweenJS**:

- **ThreeJS** is a high-level JavaScript library that allows to create in an easy way 3D content on webpages. Despite ThreeJS uses ultimately the WebGL renderer to render things on screen, we've chosen to use this library because it allows to handle lights, scenes, textures, 3D models and so on in a much simpler way than native WebGL, thanks to its several classes and function.

- **TweenJS** is a simple JavaScript library used for tweening and animating. With this library we can change the properties of the vehicle in a smooth way (with proper interpolation functions) and so it will results at the end in a smooth animation.

Although collisions are a big part of this game, we've chosen to not use a physics engine, but we exploited the RayCaster class of ThreeJS in a smart

way in order to detect them. Further details in the "Technical Aspects" section.

3 Hierarchical Models

In our game, every vehicle is a hierarchical model, and so even our car. By choosing the cars as hierarchical models, we have more freedom every components, in particular their wheel, because a racing car must give the impression to run at high speed, and with a one-block model this was impossibile because we couldn't control the wheels. For the cacti, we had no particular preference since they are present to give a deeper sense of immersion in the desertic scenario, and so we've just chosen a model that doesn't make the scene too much heavier. Same as for the fuel tank.

4 Lights and Textures

Regarding the light, we've chosen to implement only one directional light in order to simulate the fact that we are racing under the sky and the only light source is the sun that's far way from the scene.
For the textures instead, different kinds are used as requested: we have a simple image texture for the road, bump textures for the sand outside the road, and each object of the scene has a unique combination of differente textures that was designed by their author.

5 User Interaction

Being the project a game, user interactions are a big aspect of it.
As soon as the initial page loads, the user can do 3 different things:

- Change the **difficulty** of the game and choosing among 3 different options;
- Display the commands available while playing;
- Start the game.

When the game starts, the player can move the car accordingly to what the menu has said and he can, at his will, either pause the game or mute the music.

Lastly, when the game ends, he can either press enter to try again or reload the page and try with a different difficulty.

6 Animations

The animation of the vehicles exploits their structure being hierarchical. In fact, as we anticipated, we've chosen them such that we can enable the wheel rotations in order to give them the appearance of being racing on the road. Also, we animated the car turning left and right when it changes lane, turning back to an idle position when no turning key is pressed.
Moreover, to visually suggest a player that the fuel tank is an item to be collected, we gave to it an animation that resembles a '90 videogame's item.

7 Technical Aspects

Crucial points of this game are the collisions, both the collisions with another car and the collisions with the items. As we mentioned before, we didn't use a physics engine to detect collision, but we managed everything within ThreeJS. To do so, we exploited the *THREE.Raycaster()* class.
Each collidable object of the scene (so vehicles, our car and fuel tanks, but not the cacti since the car is handled to not guide outside the road) has an hitbox, that's just a AABB that contains the object and it's not visible. What our program does is to create a raycaster and then it casts several rays in several directions and checks for a possibile intersection. Then, if this intersection has a positive length (meaning it has encountered the hitbox of an object somewhere along the given direction) and if the length of this intersection is less than the length of the direction vector (not normalized), then a collision between hitbox occurred, and so it returns the collision (which is then handled accordingly to what kind of object we've met).
Another important aspect regards the spawning of the vehicles. Different difficulties influence the spawn rate of both the items and the cars: as the difficulty increases, the probability of a vehicle increases and the probability of a fuel tank decreases, viceversa as the difficulty lowers. Since the vehicles spawn randomly, one critical aspect was to avoid situations in which all the lane are occupied by a row of car and so the player was forced to take a hit.

We managed this situation in our code, and so there's should always at least a way that allows the player to avoid hits.
Lastly, we added a lot of sound effects to enhance the player experience, and once again ThreeJS allowed us to do it in a very simple way.

8 Credits

All the models and the textures are available on the following links:
Cars:

- 1964 Ferrari 250 gto Model : `https://sketchfab.com/3d-models/1964-ferrari-250-gto-849c88c65911496d92363dc2980f6f4e`
- Fiat 500 Model: `https://sketchfab.com/search?q=fiat+500&sort_by=-relevance&type=models`
- Truck: `https://sketchfab.com/3d-models/gmc-sierra-work-truck-1fae2b50fbe14d2c98296a2560a38399`
- Police car: `https://sketchfab.com/3d-models/police-car-9166b13b6ae341f4bfc093edb71d74f4`
- Smart: `https://sketchfab.com/3d-models/vehicle-smart-e3c882bacc604e2abe8784035af3fc75`

Environment:

- Cactus: `https://sketchfab.com/3d-models/cactus-low-poly-8027a1cceedb4d8189592f316b0c4704`
- Road: `https://sketchfab.com/3d-models/simple-road-texture1-41b5e535a9f7494a88f0e48819e3b744`
- Fuel Tank: `https://sketchfab.com/3d-models/fuel-tank-free-8117be222b2c4e3dad3da973cb750cf5`
- Sand: `https://everytexture.com/everytexture-com-stock-nature-sand-00029/`