Interactive Graphics Final Project
-
Report

Valeria Pirro
Mat. 1969509

Academic Year 2020/2021

1. Game Overview

The game implemented is a simple game where the user has the goal to catch as many Pikachus as possible, without running out of Pokeballs and while avoiding the “fake Pikachus”.
The collision with a Pokeball gives the user the possibility to catch another Pikachu, while a collision with a “fake Pikachu” corresponds to Game Over.
The user can collect Pokeballs without limits as long as the objects will spawn.

When the user opens the game’s webpage, there will be a “Welcome screen” where the rules and commands are explained:

The difference implemented between the three levels is an increasing velocity and spawn rate of the objects that will spawn from the horizon line. Easy corresponds to a the slowest velocity and spawn rate while Hard corresponds to the fastest ones.

The game is entirely controlled by keyboard commands, so when the user will reach the end of the game (after the spawn of 30 total objects) or the Game Over state (collision with a fake Pikachu or collision with a Pikachu with 0 Pokeballs), there will be also a key to restart the game.

2. Libraries

The libraries used are **ThreeJS** and **TweenJS**: the first one was used to implement and build the whole scene with hierarchical models while the second one has been used to implement the smooth movements of the objects moving towards the player during the game and the floating of the user's Pokeball.

3. Hierarchical Models

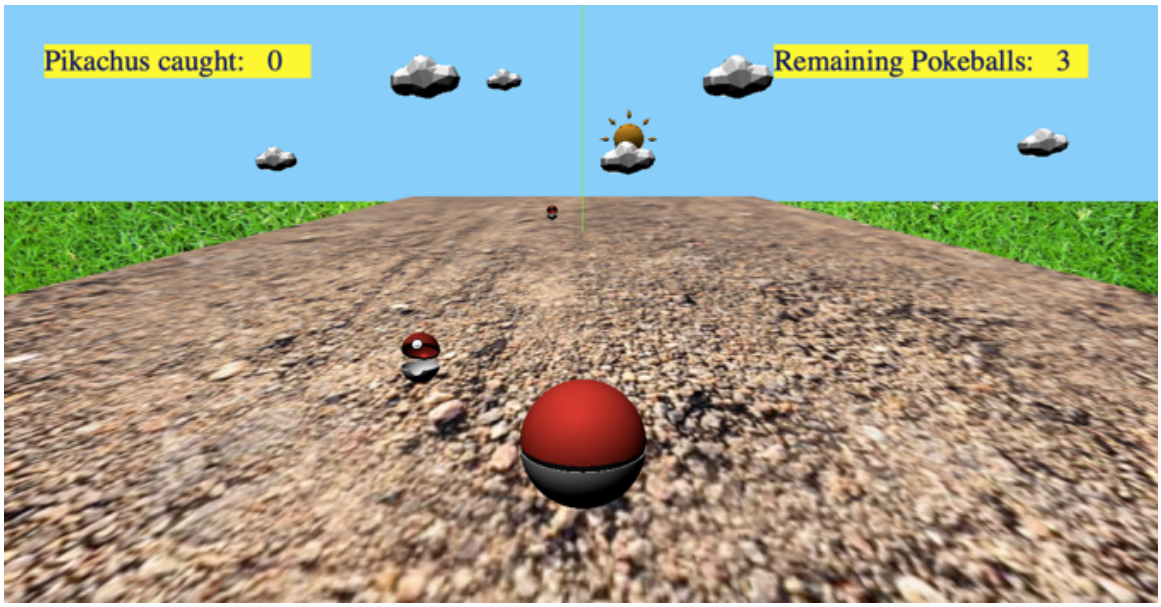
In this game, the whole scene is built as a hierarchical model, because thanks to this approach the management of the appearing and disappearing objects in the path is very easy and lightweight.

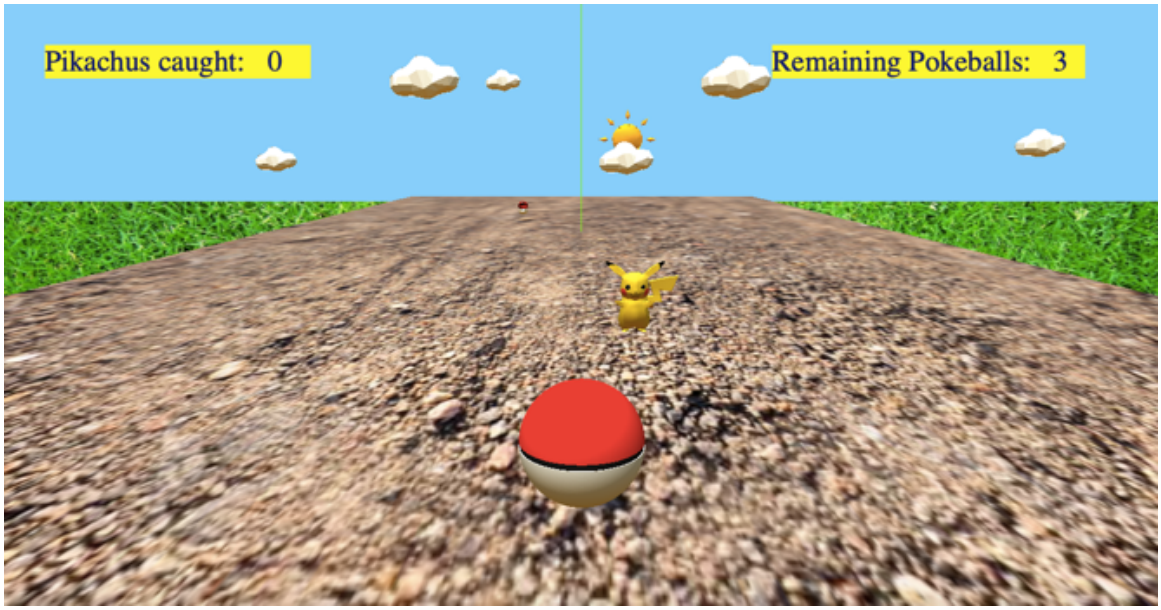
The simplicity of this approach comes from the fact that when an object spawns it is added to the scene, while when it reaches the User's Pokeball has to disappear so it is removed from the scene.

The lightweight aspect of this approach is that when the object is no longer relevant for the state in which the game is, if it is not removed it takes out a discrete amount of computational power which can be otherwise used in further states of the game.

4. Lights and textures

- Lights: the illumination chosen is a HemisphereLight aided with a DirectionalLight because the scene is modeled to give the impression of an outdoor ambientation, illuminated by the diffused sunlight.
  The other methods of illumination like for example only spotlight illumination or only directional light were not optimal for this case because they gave the scene an illumination indoor-like, while the effect to pursue was an outdoor-like and possibly cartoon-like illumination.
  Here is shown a comparison between only directional light (top) and hemisphere and directional light (bottom):





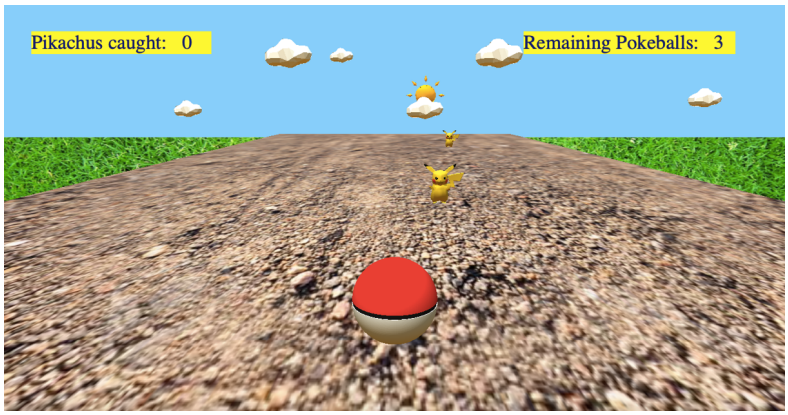
- Textures: two different kinds of texture have been used:
  - Sky: the sky is modeled by a tridimensional rectangle at which the color has been simply set to sky blue;
  - Ground: the ground in which the objects move towards the user has modeled by another tridimensional rectangle wrapped with the image of a pebbles trail;
  - Grass: this part is modeled adding the image of real grass as background of the scene, to fill all the side gaps left by the sky and trail in a natural way.

5. User Interaction

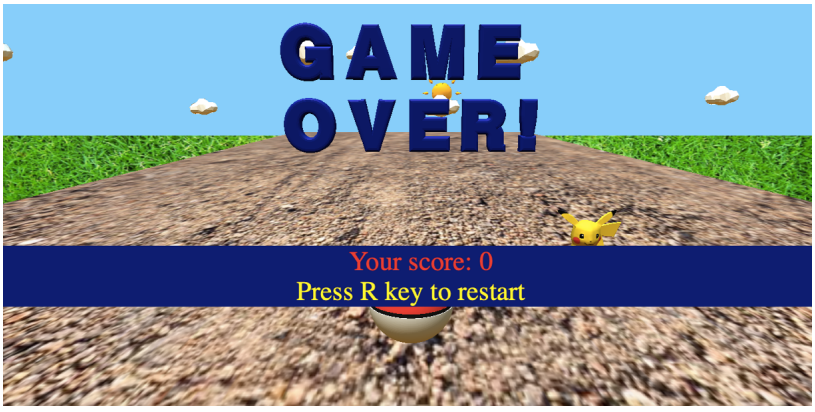
The game allows the player to start the game with a desired difficulty, restart the game and move the Pokeball around, all from the keyboard with dedicated keys.

When the user press the key corresponding to the desired level, the game will begin with an initial number of 3 Pokeballs, and will be able to increase the score by using them to catch Pikachus.

During the game the player is able to keep track of its progresses reading the labels placed in the top left and top right corners of the screen, where are shown the remaining Pokeballs and the number of the Pikachus caught (which will represent the final score when the game will end):



When the player loses, the page shows the Game Over alert along with the score and the key to press to restart the game:



6. Animations

The main animations implemented in the game are the smooth linear movements of the spawned objects from the horizon line towards the player.

As said, the objects are rendered far away in the horizon to give the impression that they appear when they enter the player's field of view. Then they are smoothly moved towards him with a basic Tween.

Another animation implemented is the little floating effect of the user commanded Pokeball: in this case the Tween exploits a quadratic easing, to give the impression of a movement which starts slightly fast and then slows down when the object reaches the final position. This results in a delicate bounce effect when the objects stands still in its place.

7. Logic of the Spawn

When the user presses the key of the desired level, the game starts and objects start to spawn in the scene after some seconds.
The spawn of those object has been implemented through a simple timed loop, internally split in three main relevant states of the game:

- Case of negative number of remaining Pokeballs or collision with a fake Pikachu: represents the loss of the user, this case triggers the functions that remove the objects from the scene and stops the loop, showing the user its final score and a text "Game Over" on the trail;

- Case of maximum object already spawned: in this case the user does not loose, instead the game is "won" since no Game Over situation occurred and the score is shown in the screen. The loop starts from this particular case triggering at every timed repetition the function that renders a random object between Pokeball, Pikachu or Fake Pikachu;

- Case of ongoing game: is the case where the user did not hit any losing circumstance and the objects are not yet all spawned.

8. Collisions

To detect collision has been exploited a check which is triggered when the Tween of the nearest object towards the user ends: if at that moment the user's Pokeball has a position which corresponds to the zone of the object, the corresponding action is performed (based on the type of the object).

Plus, in order to make the collision realistic, an interval of tolerance has been considered: if the object touches the user's Pokeball in a range starting from -4 to +4 around its center, the collision is considered valid. If this fix hadn't been implemented, the object and the user had to be exactly in the same place and even a displacement of 0.0001 would invalidate the collision, which is a very unnatural behavior.