Project Interactive Graphics

Di Marco Gianfranco - ID 1962292

July 2021

1 Introduction

Cyber-Fishing is the work I made for this project: it consists in a fishing game in which a robot should catch all the fishes that are in the lake. The fishing can be as relaxing as stressful, it depends on the points of view.

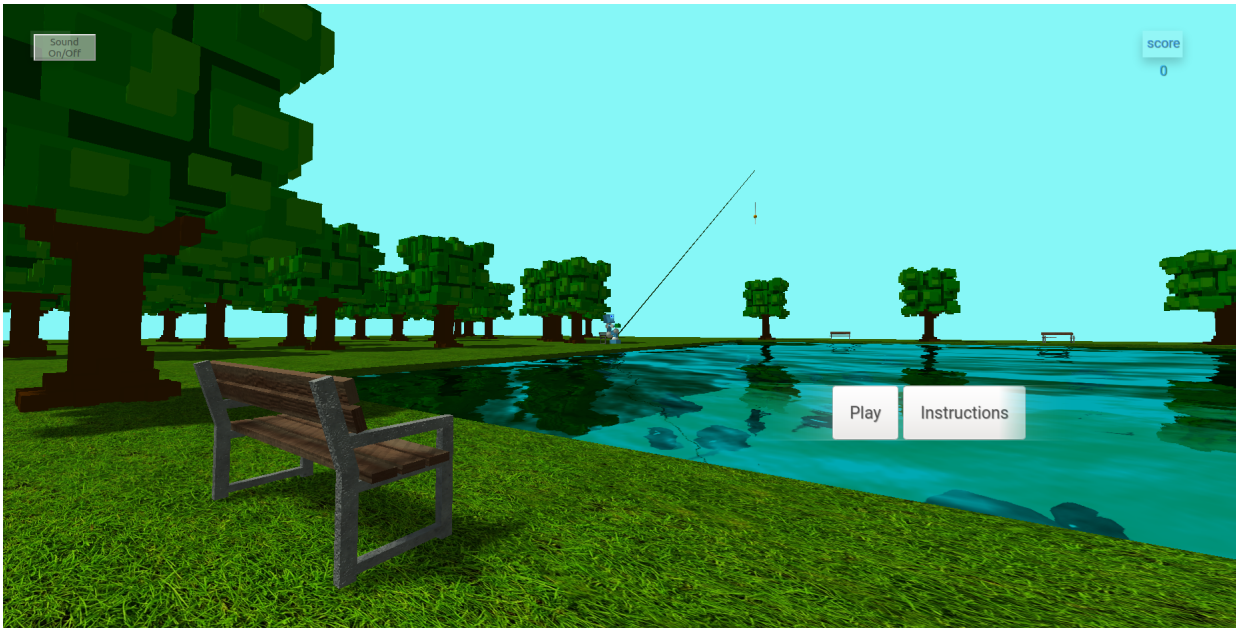

Figure 1: Start Menu

As we can see the design is very minimal: the start menu is simply a series of keys with a mild transparency and basic shapes. This choice was made principally for two reasons, first to remain on the same scene and enjoy the landscape or the lake by watching the animations or the objects, and second to maintain the simplicity of the user interaction in order to concentrate only on game and animations.

2 Technical Aspects

1. Environment. The environment of this project is based mainly on the **Three.js** library, which provide a very rich framework and a lot of handy function for a variety of projects and user requirements. It's not the best choice for a professional project with a big size and complexity, but it can be exploited almost everywhere.
Another important contribute given to this project is given by the **Tween.js** library, that is a simple interpolation library useful for basic animations like the ones in this project. All animation details are reported later.

2. Models. Regarding the models used in this work, most of them are taken from the website **CGTrader**: [1]. It provides a lot of stuffs for each type of project, with different file format and size. For this project I've used only **.obj** files, because they are very simple to manage and are quite good for simple scenes. The objects downloaded from this website and put into this project's scene are:

- Trees
- Benches
- Fishes
- Nature elements

Regarding the Nature elements, only the rocks were used in this project, but is very easy to add other stuffs with the smart programming interface provided in this project. One more detail to know is that the water of the lake is a mesh taken from the Three.js library, so it's not part of an external package.

3. Files organization. The files in the main directory are organized as following:

- *index.html* - main html page
- **css**

  style.css - stylesheet
- **objects**

  trees
  benches
  fishes
  nature
- **sounds**
- **textures**
- **js**

  definitions.js - contains the properties of all objects
  utilities.js - utility functions
  initialization.js - initialization and interaction functions
  draw.js - draw functions of all objects
  animations.js - animation functions
  game.js - explicates main gaming parts
  main.js - the rendering file
  lib
  * **three.js-master**
  * *tween.esm.js* - Tween.js library

[1] https://www.cgtrader.com/

4. Programming style One of the main strengths of this project is the programming style. It follows all the principles of modularity and low dependency in order to provide functions that are very comprehensible and readable, besides being efficient. Furthermore, here is very few repeated code and the only library which is a little bit more flexible ro these rules is *animations.js*, in order to have lower readability but higher efficiency and versatility in the animation process.
All the functions provided in this project can be exploited and extended to support bigger projects, without the need to rewrite entirely the function or add a lot of global variables.
One last note is that in the whole program there isn't the **var** statement. This is because it's becoming deprecate, and its substitute is the statement **let**, more flexible and lightweight.

3 Robot

The robot is the main character of this game. It's structure is not complex, but allows to make 3D movements to all interested directions for the purpose of this game. Note that **not all** the degrees of freedom of this model have been used in this project. This allows us to extend the robot and its behaviour for other works or refine the already existing one, by adding more functionalities.

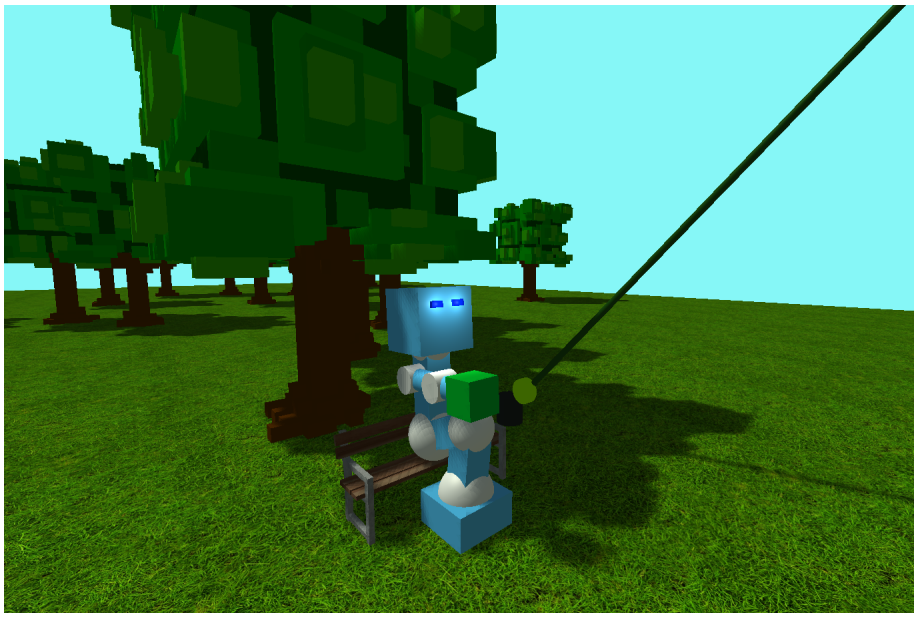

Figure 2: Start Menu

The robot is a **hierarchical model** composed of 14 parts, diveded into:

- **Joints**: ankle, knee, hip, neck, shoulder, elbow
- **Parts**: base, lower leg, higher leg, bust, head, lower arm, higher arm, actuator

Also two luminescent eyes have been added to the model. All these components are exploited to make the animation, except for the base that is fixed. The actuator is an equipable module, on which we mount our fishing pole; in future also other fishing tools can be applied here (for example, guns or fishing net).

4 Interactions

The implemented interactions in this project are very few, in fact as I stated before it is not the main point of this project. The first interaction that a user can make is to read the instructions, fundamental for the game. Then it is possible to turn on/off the sound, that is a nature background. Once clicked on the Play button, there is a camera animation and then the user is allowed to make three more interactions: the first is to move the robot (in order to adapt to the movements or explore the structure), the second is to equip the fishing pole to the robot and the third is to adjust the view. Once equipped, the fishing line won't be removable anymore.

Regarding the game itself, the concept is very simple: to catch as much fishes as possible. The controls are the following:

- **W** - move forward
- **S** - move backward
- **A** - move leftward
- **D** - move rightward
- **E** - equip the fishing pole
- **SpaceBar** - Try to catch a fish

In order to catch a fish, it's necessary that it stays near to the fish line. When it happens, we can press the SpaceBar key to catch it. If we fail (fish too far) the score is reduced by one point, otherwise it's increased by the same. The game terminates when:

- **(Victory)** - The lake is void
- **(Defeat)** - The score reaches -10 points

Another important interaction is represented by the **OrbitControls** class provided by the Three.js library, which allows the user to move the mouse and change the view during the game; this is fundamental to make good catches (a good technique is to catch the fish when the floating's shadow is in the fish's shadow).

5 Animations

Robot

The robot is animated by exploiting its hierarchical model. So, each joint rotation influences the movement of all other children, making the robot move in all the necessary directions. When the fishing pole is equipped, it's added to the hierarchical model, so each movement of the robot moves also the fishing pole. It's necessary in order to make this task, otherwise the animation would have been much more complicated. All the joints participate to the moving animation, except for the elbow, that is exploited for the catching animation. In fact, when we press SpaceBar, the elbow move up, like simulating the capture of a real fish.

Fishes

The fishes in the lake start all from a specific position, and then all the future movements are completely random (but constrained). The animation of a fish is composed of two parts: the main movement (fish moves toward), and the tail movement. This is necessary to represents more realistic fishes. Also the time exploited to complete the whole animation is random, so that a fish makes some burstes alternated with slow swims. One more thing to say is that for this part I used the Easing function **Quadratic** of Tween.js: this increments the realism of the fishes, that usually move faster at the beginning of their movement. It is possible, in the future, to exploit this animation to make more difficulty levels: in a simple case the fish will move random but slowly, and they can be captured

easily; in the medium case they will behave like in this project, and in the hard case will be very difficult to catch them because of the increased speed and the more precision required.

Camera

There is also a small animation of the camera at the start of the game; in fact, it's first positioned with the robot as target and a wide distance from it, but when the user click on the "Play" button, it reaches smoothly the fishing point. This is done each time the game starts, so also after the moment in which you lose or win the game.

Fishing Line

The last animation implemented is relative to the fishing line, and it's divided into two animations: the rotation (which will remain almost perpendicular to the water) in order to simulate the gravity, and the scaling, in order to maintain the floating on the surface of the water. Furthermore, the floating is unscaled, because it's a child of the main line and so will change its shape (undesirable behaviour). The terminal line (which is different from the main line (like in the real case) and the reel do not take part of the animation, but I added them for completeness.

6 Lights and Textures

Lights

It seems that the sun is the only light in this scene, but in reality there are four different types of light in this project: the sun itself is composed by two component, one directional and one hemisphere. The third light exploited is an ambient light, in order to make the scene more bright and colorful. The last is the light of the robot's eyes, which is dim like to simulate two led eyes.

Textures

The texture used in this project are various, and they do not follow the same scheme for all the objects. For example, the benches have a Base texture and a Normal Map texture, while the Trees have only a Base Color texture. Another important texture used is the one of the water, that make the lake very realistic and smooth. The more important texture, anyway, is the grass texture, implemented for the field. All the objects have their own texture (except for the robot and the fishing line, that are just colored meshes).