Interactive Graphics
Final Course Project

Authors:
Sveva Pepe - 1743997
Simone Tedeschi - 1762897
Claudia Medaglia - matricola
Christian Marinoni - matricola

Professor: Marco Schaerf

June 2020

Contents

1 Introduction

The goal of the project is to implement an interactive application that make use of basic WebGL or advanced libraries, as in our case ThreeJS. The project cover the main aspects treated during the course, such as lights, textures, hierarchical models and animations.

2 Game

We decided to develop a 3D version of "Duck Hunt", a famous game from the 80s, in which the objective is to hit as many ducks as possible. We designed the scene as a first-person game, where the player is located in a tall grass field and controls a rifle. Thanks to the help of his dog, that frighten the ducks hidden in the tall grass, the hunt can start. The game ends when the player miss five ducks. In addition, our game provides other functionalities like enable and disable sounds, pause the game or restart it.

The above described scene is depicted in the following figure:

Figure 1: Starting scene of the game.

3 Scene

The scene shown in Figure 1, which is a ThreeJS object, has been obtained adding textures, lights, and different 3D models. The dog and the ducks are hand-made models while the other are taken from SketchFab. We used perspective camera to make the scene as realistic as possible to let the center of projection coincide with the user's eyes. The prospective uses as parameters: *fovy, aspect, near and far*. *Fovy*, which stands for "field of view y-axis", identifies how wide the eyes open along the y direction. *Aspect* represent the ratio between the width and height of the canvas. *Near* and *far* are any positive numbers representing the minimum and maximum distances of the object, with the restriction that near is always less than far. For the camera is defined also the *lookAt(x, y, z)* method, where the *x, y* and *z* are the coordinates of the scene. The texts on the bottom right corner instead, have beed modeled using TTFLoader of ThreeJS, where through a Mesh the relative colors have been applied. Moreover, in order to make our application responsive we add a dedicated Listener to adapt the window size based on the device resolution. Finally, to improve the user experience antialiasing has been used.

3.1 Lights and Textures

The ground texture was created by repeatedly applying a texture on a plane, using a TextureLoader. Then, the texture is added to the scene through the use of meshes that map texture coordinates into world coordinates. Three directional lights have also been added to the scene. Two of them were placed on the left upper and bottom right corner respectively of the scene to reproduce a sunny day, otherwise using only one of them we obtain either dark clouds or dark objects. The third one has been introduced to illuminate texts because they are ahead of other elements and so the previous lights were not able to light up also them.

4 3D Models

In this section we explain the models we have included in our project. They are splitted into the following two categories:

- linear models: objects that are treated as an atomic entity;
- hierarchical models: objects composed by various sub-objects.

4.1 Linear Models

The linear models are models that are treated as singles entities. In our game are present the following four groups of linear models: Rifle, Trees, Clouds and Bushes. They contain 1, 4, 5 and 11 instances respectively that can be observed in Figure 1. We did this categorization to handle different kinds of objects in different ways, because objects belonging to different groups are indipendent to each other. Trees/bushes positions and orientations have been preset and remain static along the entire gameplay. Clouds positions instead, vary over time and rifle orientation can be controlled by the user. These last two aspects will be further explained in Sections 5 and 6 where we will provide technical details.

4.2 Hierarchical Models

Hierarchical models are models composed by different sub-objects that allow to represent relationships between such objects. The major benefit of hierarchical structures is the possibility to handle animations in a simple and efficient way, because, for instance, if we want to apply a rotation to the whole object we need only to perform it to the root of the object itself, instead of applying n rotations to each individual component.

4.2.1 3D Duck Model

The first hierarchical model that we introduced in our project is an hand-made 3D model of a duck, created with Blender. Its structure is divided into four components: left and right wings, torso (including also the head) and legs. Such structure allowed us to reproduce the desired behavior, which consists in a simultaneous diagonal translation and a synchronous upward rotation of the wings. The above described hiearchical structure is shown in the following figure:

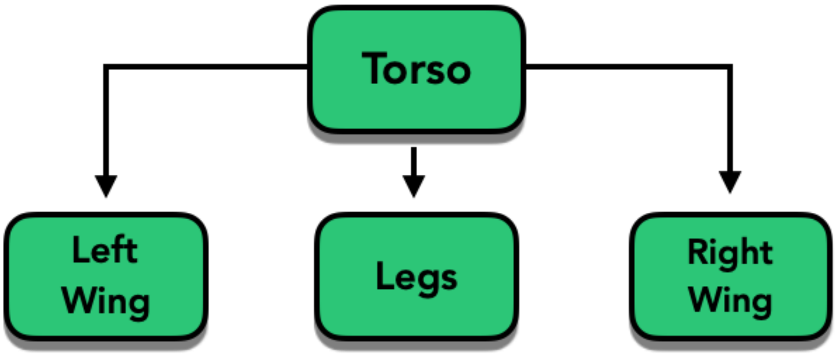

Figure 2: Hierarchical model of the duck.

4.2.2 3D Dog Model

The second hierarchical model that we used in our project is again an hand-made 3D model created with Blender, representing a dog. Its structure is divided into ten components: left and right upper front legs, left and right upper back legs, left and right lower front legs, left and right lower back legs, torso (including also the head) and tail.

The hierarchical structure of the dog is depicted below:

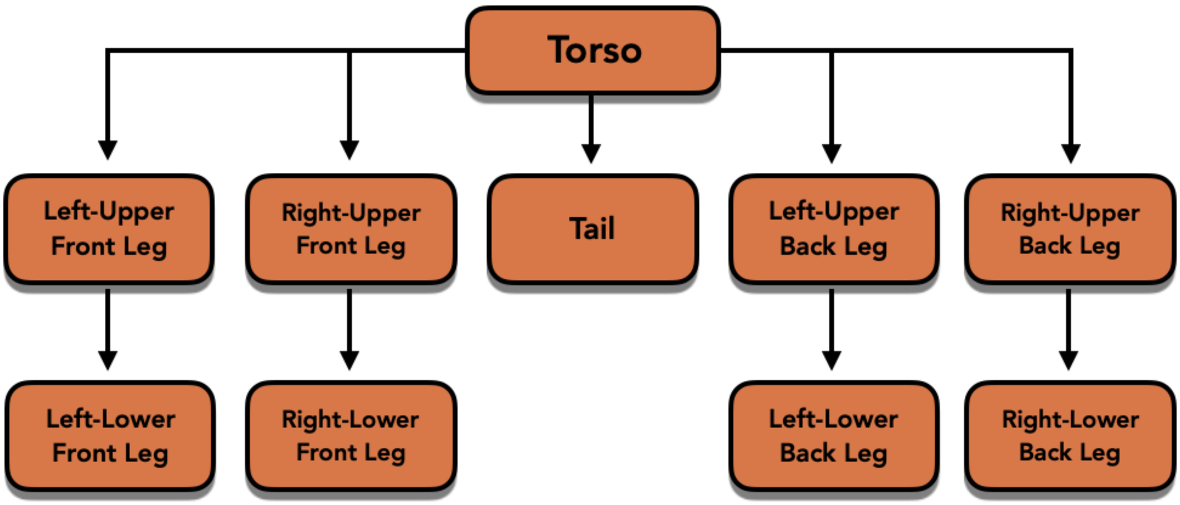

Figure 3: Hierarchical model of the dog.

We exploit such structure to let the dog move towards the bushes with few basic movements. To achieve this, we simply translate the "Torso" node, which is the root of the hierarchy tree, and automatically all other components, which are attached to it, will follow its movement. In the meanwhile, we alternate the legs movements back and forth to reproduce a walk. Additional details will be provided in the Animations section (5).

5 Animations

6 User Interaction

6.1 Sounds

7 Conclusion