Report Project IG

Adrian Minut - Andrea Celestini

September 2022

Contents

1 Introduction

1.1 Game description

The project consists in creating a game with 3D engines based on WebGL, which we chose to be Three.js. Our game is called Virtual Tennis and it consists of a representation of the game of tennis, in particular it is based on isometric tennis.
The game simulates a tennis match between two players, but one is controlled by the user with keys (WASD and SPACEBAR), the other not playing, we simulated only one side of the game.
The goal of the game is to win two sets: every player has to win 6 games to win a set and has to score more than 40 points to win a game, the same as real tennis.

1.2 Libraries

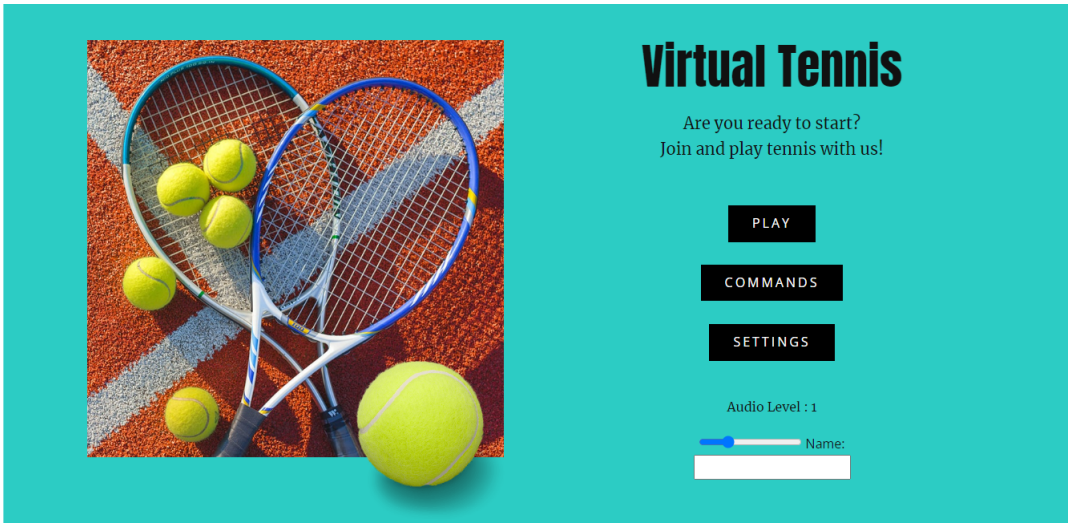
We used different libraries for different purposes:

- Three.js: main library in our project, we used this one to setup the camera, to render the scene, to create the audio listener and attach the sounds to the camera, to load the 3D models and to control them, in particular we used some modules of Three.js:
  - OrbitControl.js: a library to manage and control the camera (trackball, zoom..)
  - GLTFLoader.js and ColladaLoader.js to load models and the rig
  - postprocessing to manage postprocessing effects
- ammo.js: to setup physics
- Tween.js for animations
- jquery to interact with DOM and to manipulate HTML elements in the GUI

2 Game Setup

2.1 Scene

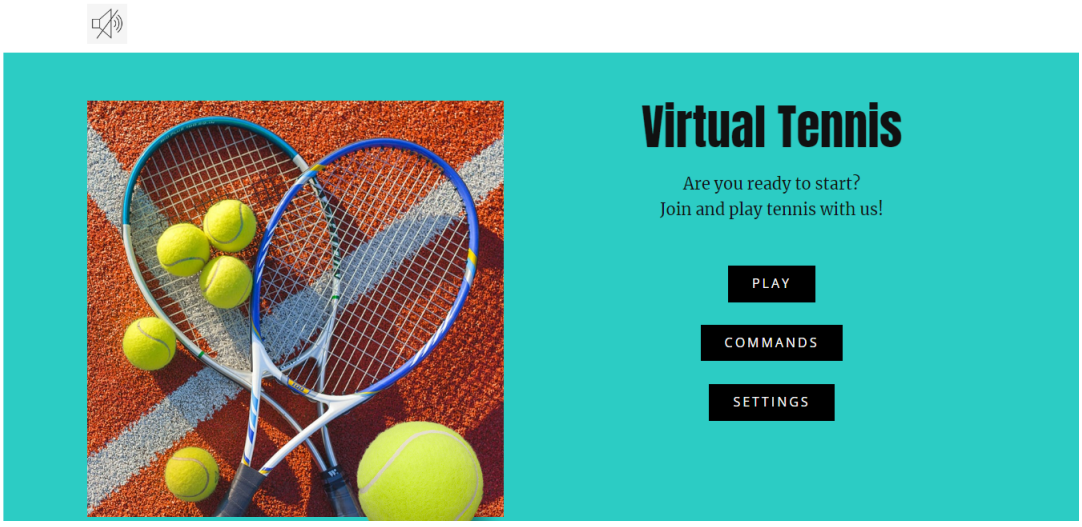
The structure is composed of the main menu and the game scene.
The menu presents a user interface to set the volume of sounds and to choose a player name.
Then from the main menu you can access the Game scene, which contains many different elements, such as the environment, the players, the court, the ball and the score. The switch of the two scenes is triggered by the "Play" button.
You can go back to the menu using the "Back" button.



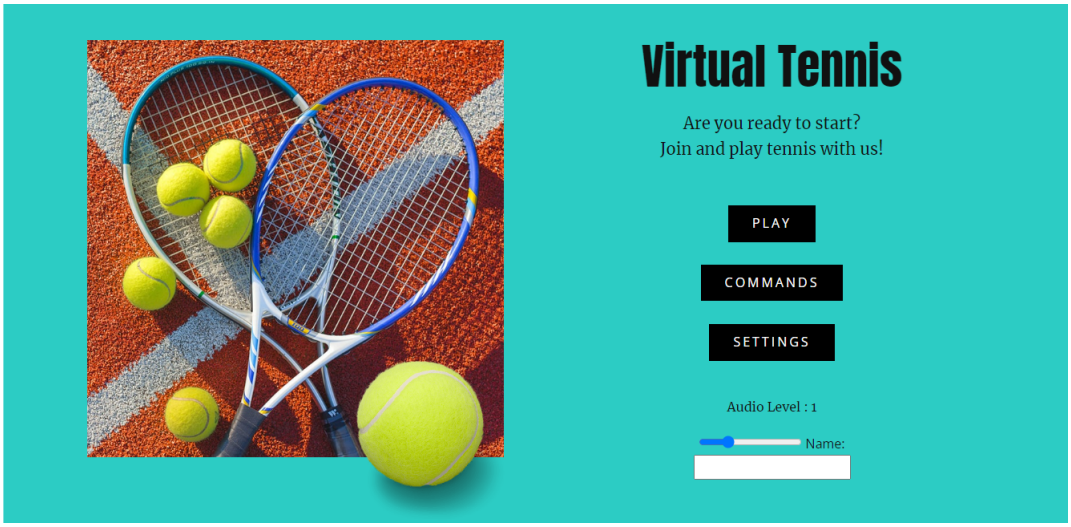
In the SceneManager module we loaded all the models, the lights and we trigger the physics setup, calling another specific module Physics once the models are loaded.

2.2 Game user interface

The application starts by opening a page with a menu, the icon of the volume and some pictures related to tennis.



Clicking on "Commands" button the user can read instructions to play the game, while clicking on "Settings" button the user can visualize and setting the level of the volume and the name of the player.



All of these settings are stored in the local storage of the browser and loaded by the JavaScript modules.
By clicking on the audio volume icon the user can turn on/off the intro song in the menu (the volume of the game sounds is set differently, using the audio slider).

2.3 Camera

Three.js provides many options for the camera, we chose the OrthographicCamera to have an isometric look for our game.
We set the viewport characteristics and we implemented in the Render module the camera panning, rotation and zoom to allow the user to control the camera using the mouse during the game.

2.4 Sounds

We used Three.js to implement sounds: we have created an audio listener and an audio loader, then we have created two different sound, one for the ball and one for to simulate a crowd stadium effect.
We attach these sounds to the camera and then we set the volume using the information stored in the local storage of the browser.
We call the play() function of the sound into GameManager module: the ball sound is called when the player hits the ball, while the other sound is called when a player scores a point.

3 Players

3.1 3D Models

We downloaded and imported our 3D models: the court, the ball and the player. The player model was rigged by us so that we could use the skeleton for animations. We have scaled, moved and rotated all the models, and also adding them textures and materials, applying opacity and transparency.

3.2 Controls

For the first player we have added controls: using the keys W (to move forward), A (to move left), S (to move backward), D (to move right) and the SPACEBAR (to load the power of the shoot and shoot releasing the key) the user can control the chosen player.
We have set the controls in the PlayerController module using as switch case structure and the input events.

3.3 Animations

For animations we used Tween.js library. We have implemented different animations for the players.
First we divided the model into different parts using a hierarchical structure with 12 components.
For each animation we set the coordinates for rotation and position for each component and then we performed animations using the interpolation of the values.
We have 6 keyframes and three animations: idle, walk (divided into walk1, walkLeft, walkRight), shoot plus the "default" keyframe.

4 Physics

4.1 Ball Collisions

We used ammo.js library to configure all the physics elements in our environment in particular using vectors and quaternions.
We implemented a JavaScript function to collisions between ball and pitch, ball and players through a coordinate system to detect if the ball is out of the pitch or into the pitch, using this knowledge for the score update.

4.2 Ball Shoot

We implemented the shoot modulating the force: it can be controlled by the user keeping pressed the SPACEBAR to load the power of the shoot that is to say increasing the module of the force.
We set also the velocity and direction and we realized a graphic of the shoot: a red circle around the player with a ray that increases when the module of the force increases.

5 Post-processing

We included some post processing to have a softer and brighter look, we implemented it with an EffectComposer chain in Three.js. The chain includes a RenderPass to render the main scene followed by a saturation effect and a brightness contrast effect, finally an EffectPass.

6 Gameplay

6.1 Score

The score is managed in the GameManager module using a class called GameState. We used a dictionary in which we stored: the sets won by each player, the current won games and the current points done by each player, the result of the previous sets and the final winner.
The score is updated using the following rules:

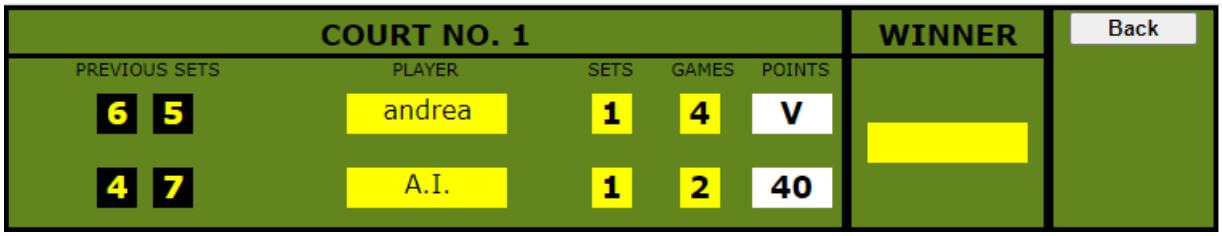
1. If a player hits the ball and it goes into the pitch of the other player, the first one scores a point, while if it goes out of the pitch the point is scored by the second player.
   If a player hit the ball and it goes into his pitch, then the other player scores the point.

2. When a player scores more than 40 points (the sequence is 15,30,40 plus the advantage) he wins a game.

3. When a player wins 6 games (or 7 if the other player won 5 games) he wins a set

4. When a player wins 2 sets he is the winner of the match

The score graphics is updated every time a player scores a point changing the DOM of the HTML file.



At the end the winner appears in the screen.