Final Project Interactive Graphics - Air Hockey 3D

Game developed by Gian Marco Nobile 1836319 and Alfredo Lipari 2084896

Contents

Description of the game

Air Hockey 3D is a game based on one of the most popular arcade games. It’s very self explanatory and while staying to its roots it brings a light spinoff. The difference from the original is that you have to challenge a robot arm that varies in strength depending on the difficulty selected. Finally, our goal is to create an exciting air hockey arcade experience in our game. We hope you enjoy it and have fun!

Libraries

Air hockey 3D is mainly built on top of Three.js and Cannon.es

We used the javascript library Three.js to render the two scenes of the game: the main menu and the game itself (with two separate canvas).

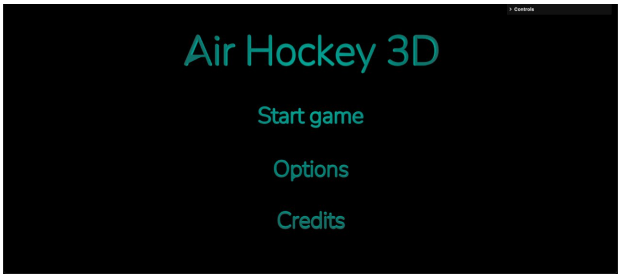

Photo 1 - Main menu

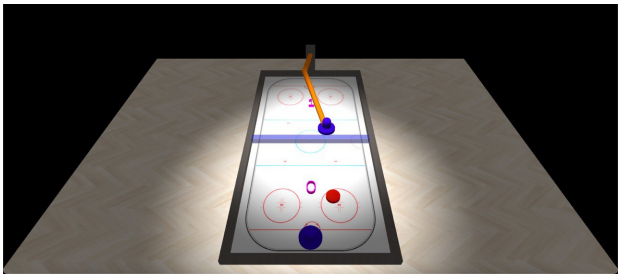

Photo 2 - Game

We separated the two scenes because the game one also employs Cannon.es, a javascript 3D physics engine. We were able to define rigid bodies (eg. wall of the arena, plain surface of the board ecc.), as well as the dynamic ones (eg. the puck and the two paddles), and thanks to Cannon, their movement can be mostly managed by the physics engine.

Additionally, we used the Three.js addon called CCDIKSolver to handle the inverse kinematics calculations necessary for the movement of the robot arm used in the game. This addon utilizes the CCD algorithm, which helps create a more realistic scene without sacrificing computation time.

Project Structure

Following a brief description of the files present in the project and how they contributes to the final result.

- Index.html ⇒ Starting point of the project, here are declared the two canvas with two div menuCanvas and gameCanvas**.** Also there is the option menu within the menu container built with the relatively new dialog tag. Finally, there are also defined the sounds and the font.
- Public ⇒ The folder that holds all the static resources like css, fonts, images, sounds as wells as the javascript files of the project

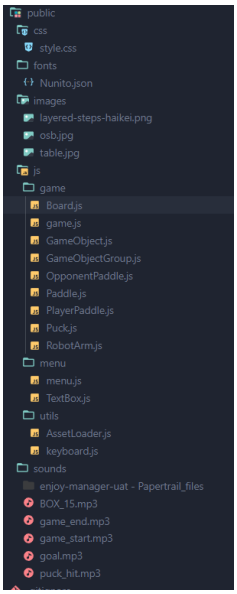

Photo 3 - public folder structure

Javascript files

The javascript files are arranged in three main folders:

- The menu folder contains the files used to render the menu canvas of the photo 1.
- The game folder contains the files used to render the game and contains the logic to play the actual game (Photo 2)
- The utils folder contains files shared by the other folders like the assetLoader and the keyboard input handler

Main Menu

The main menu is composed of four text geometries that composed the available actions and the title, to give the effect of a real 3D game.

Since the texts are not dom elements, to achieve the on click functionality, we used the ray caster to find the exact mesh intercepted by the vector casted by the camera

position passing through the mouse position. In the menu.js are located all the handler functions to listen to mouse events. The texts can be also moved in the 3D space by dragging them with the mouse. Also we decided to leave the gui in the main menu to quickly change the color and the bevel of the text, options that are not available in the game section.

Lastly, the credits and the settings texts uses the newly introduced dialog html tag that gives helpful functions to open and close the dialog and an attribute to check if it's open. Meanwhile the Start game action, on click, hides the main menu with the none css property and calls the setup function of the game.js file to initialize the game.

Game Models

The models in the game are custom made to allow easier integration with the Cannon engine and each of them has their own file containing the rendering definition as well as the game logic. We will provide now a brief description of the main ones:

Puck

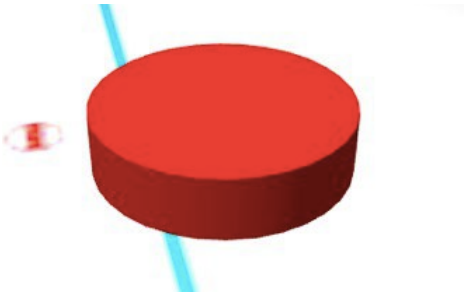

Figure 4 - The Red Puck

The puck is the only game object where the movement is totally managed by the physics engine, following collision with other game objects.

Even though the object mesh is modeled as a cylinder, the cannon body is that of a sphere to allow for faster collision detection. All the logic is located in the Puck.js file.

Paddles

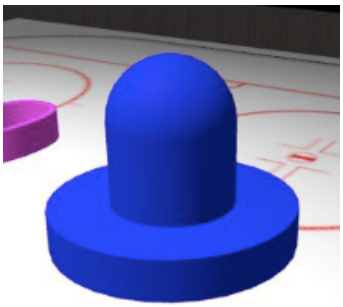

Figure 5 - The Blue Paddle

Both the player and the opponent paddles are based on the Paddle.js file which defines the Mesh and the body. The upper structure of the paddle doesn't ever come in contact with any other game objects. For this reason, only the lower cylinder of the paddle (the base) is also a physics body, the rest (the upper and the sphere) is purely aesthetic, eliminating physics calculations for bodies that will never come in contact with any other objects.

What differs between the player and the opponent paddles is how their movement is managed. The player paddle is controlled by using the WASD keys on the keyboard, while the opponent's movement is "reactionary" in the sense that it follows the target, which in our case is the Puck. The opponent can thus ne considered a really "simple" artificial opponent, which can still be difficult at higher speeds.

Robot Arm

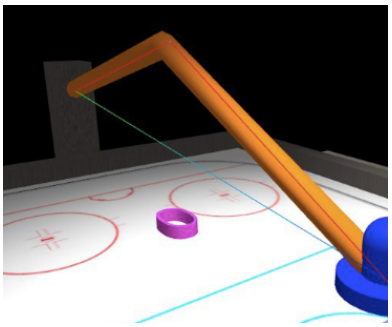

Figure 6 - The Robot Arm with skeleton debugger

The Robot Arm “controls” the paddle and is the main antagonist of the game. The Arm is composed of a hierarchical bone structure, which is then bound to a cylindrical mesh. The root of the arm is fixed while the endpoint, aka target, is the opponent paddle. Through the use of inverse kinematics, the skeleton of the arm is moved in such a way that the endpoint is always on target, creating the effect that the arm is moving the paddle, and not the other way around.

Board

The board.js file handles the main functionalities of the game logic, such as creating the game surfaces (eg. walls and plane), checks on the goal, updates the players score and checking whether the max score is reached to change the state in game over.

The logic to decide whether a goal is done by one of the two players is simple and is based on the position of the puck.

Finally, if either of the two player reaches the maxScore, the game stop and the end game dialog is shown.

Additional Features

Here there is a brief descriptions of other interesting features added to the game to improve the general experience

Sound effects

There are loaded in total 5 mp3 files, one of which is the background soundtrack taken by a known game “Dangaronpa”. The other fours are taken from youtube and trimmed to extract only the sound needed.

Settings

The settings dialog let’s the player change some options of the game, like the following:

- Difficulty: There are four levels of difficulty, each one sets the speed of the enemy player, so he will be slower/faster in following the puck. We think that impossible is a nice challenge to try. This cannot be change in game
- Points: This cannot be change in game

- Volume: Controls the volume sound of only the background music. This can be changed in game
- Start/Stop Music: Controls only the background music. This can be changed in game

Settings can also be opened in game by using the ‘Escape’ key. Opening the settings in game cause the game to pause and it will restart only by clicking the ‘Escape’ key again.

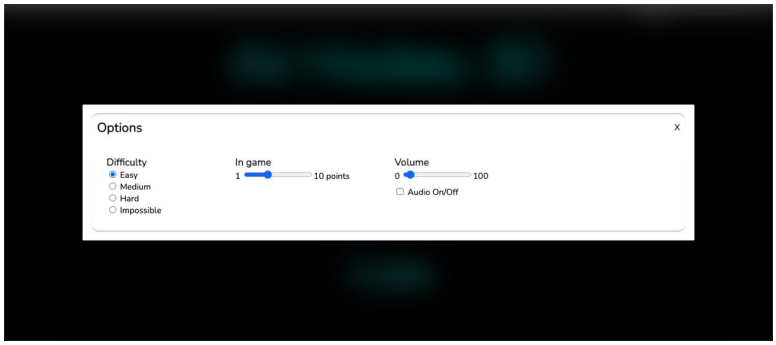

Figure 7 - Option Dialog

End game dialog

When the game finishes, it displays a text showing the outcome of the game. Then it lets the user decide to start a new game, to open the settings (to only controls the music) or to go back to the main menu.

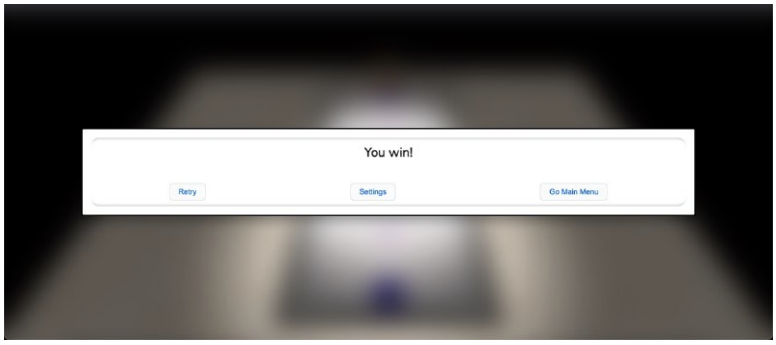

Figure 8 - End Game Dialog