Air Hockey Game Kit

by ETI

v 1.2

Change Log

Always backup you project before updating !

v 1.2 :

- Air Hockey Table low poly 3D model added
- **ColliderBounds** object mesh updated
- **Puck** Sphere collider replaced with capsule collider
- **PuckScript** : puck bound limits deactivated in **Update()** function
- **MalletPlayerScript**, **MalletCPUScript** : Mallets move limits slightly modified
- In demo mode (**menu** game State) puck is now replaced after scoring.
  (changes made in **MainScript** **UpdateScore()** function)
- Game preferences : difficulty, score to win, camera view, are now stored in Player Preferences
  (changes made in **MainScript** and **UIButtonScript**)

v 1.1 : Pause, camera selection, Level Up system example.

v 1.0 : first release

Requirements

This asset requires Unity 3D (Free or Pro) 3.5 or later version

Overview

General Description

'Air Hockey Game Kit' is a Unity complete project package aimed to give you hints on creation of your own physic based air hockey or similar game.

The project was designed to be cross-platform, and optimized to run smoothly on mobile devices. You'll find an example of how to handle 2 players game on touch screen devices.

Scripts are thoroughly commented and available in both UnityScript (.js) and C Sharp (.cs) languages.

Key Features

- Real world scale and physics, local space based code : this is meant to facilitate possible integration and placement into another project
- Well organized, fully commented scripts
- Tight to essential : Even though an object-oriented User Interface example is provided, you will not find unnecessary assets or code

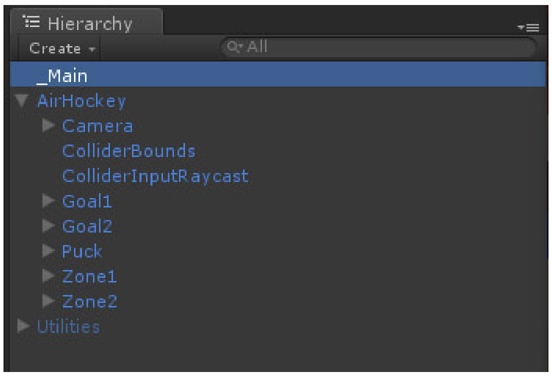
Use cases

- Help beginner developer to learn how to create his own game, or have a working template to build upon.
- Give intermediate developer another point of view on workflow, tips, and optimization
- Give a starting point to integrate a playing air hockey table into another project, for example a FPS or sandbox game. 'Air Hockey Game Kit' could be a handy way to do that (you should look at the main script for reference)

System Overview

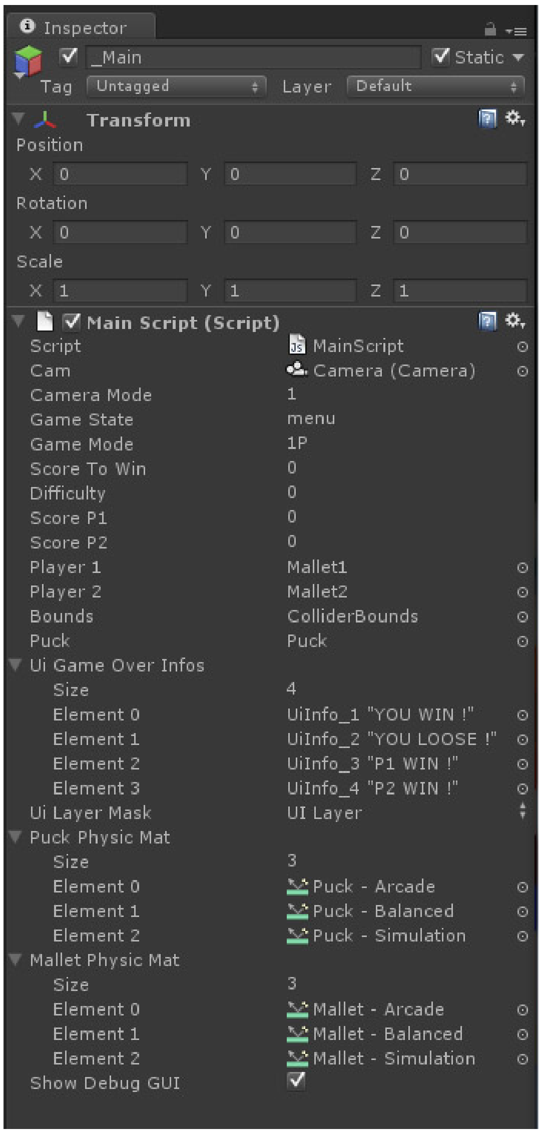
Main script

MainScript controls the main functions of the game. It receives and manages the different game and UI events. This script is attached to **_Main** gameObject.



```
_Main
▼ AirHockey
    ▶ Camera
      ColliderBounds
      ColliderInputRaycast
    ▶ Goal1
    ▶ Goal2
    ▶ Puck
    ▶ Zone1
    ▶ Zone2
▶ Utilities
```

This asset use a single scene for the various game states (menu, game sessions, gameover...), this avoids loading times and optimizes the project, especially if you target low-end mobile devices.



_Main ✓ Static
Tag Untagged Layer Default

Transform

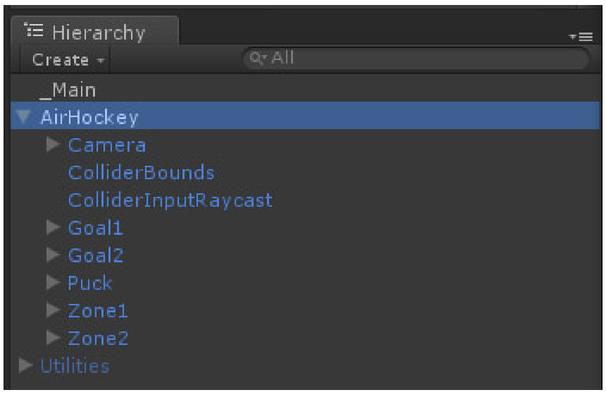
	X	Y	Z
Position	0	0	0
Rotation	0	0	0
Scale	1	1	1

Main Script (Script)

Script	MainScript
Cam	Camera (Camera)
Camera Mode	1
Game State	menu
Game Mode	1P
Score To Win	0
Difficulty	0
Score P1	0
Score P2	0
Player 1	Mallet1
Player 2	Mallet2
Bounds	ColliderBounds
Puck	Puck
Ui Game Over Infos	
Size	4
Element 0	UiInfo_1 "YOU WIN !"
Element 1	UiInfo_2 "YOU LOOSE !"
Element 2	UiInfo_3 "P1 WIN !"
Element 3	UiInfo_4 "P2 WIN !"
Ui Layer Mask	UI Layer
Puck Physic Mat	
Size	3
Element 0	Puck - Arcade
Element 1	Puck - Balanced
Element 2	Puck - Simulation
Mallet Physic Mat	
Size	3
Element 0	Mallet - Arcade
Element 1	Mallet - Balanced
Element 2	Mallet - Simulation
Show Debug GUI	✓

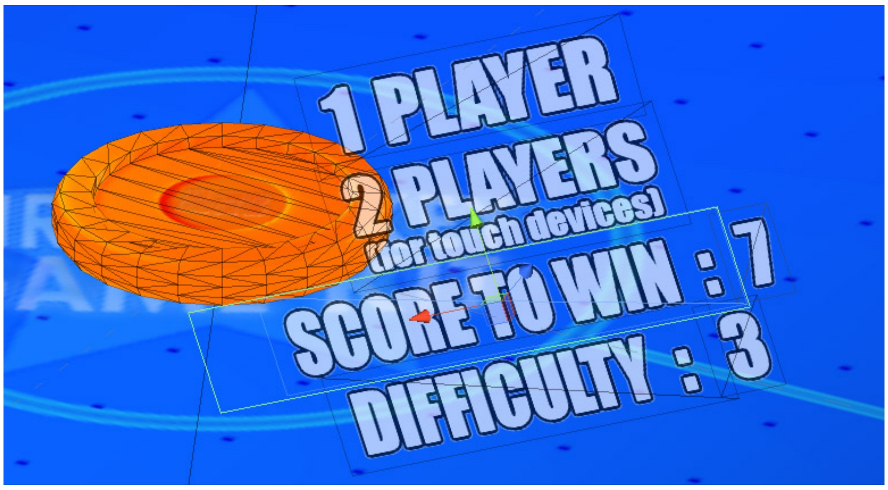
« AirHockey » container

AirHockey gameObject contains all the other assets of the project, including the camera. All chidrens' behaviors are constrained to **AirHockey** position and rotation.



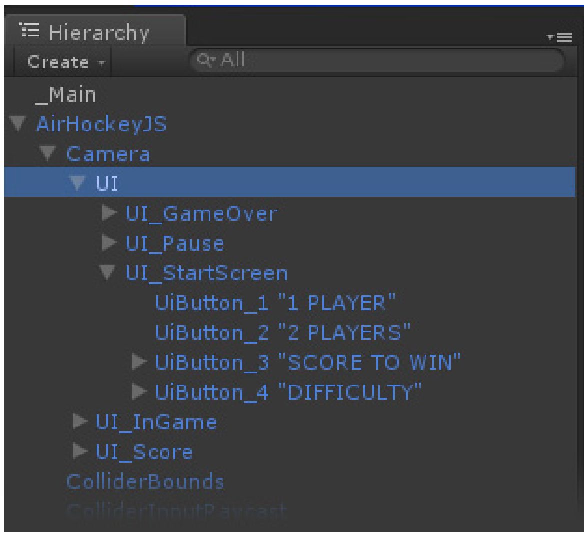
```
Hierarchy
Create ▾        All
_Main
▼ AirHockey
   ▶ Camera
     ColliderBounds
     ColliderInputRaycast
   ▶ Goal1
   ▶ Goal2
   ▶ Puck
   ▶ Zone1
   ▶ Zone2
▶ Utilities
```

Custom User Interface



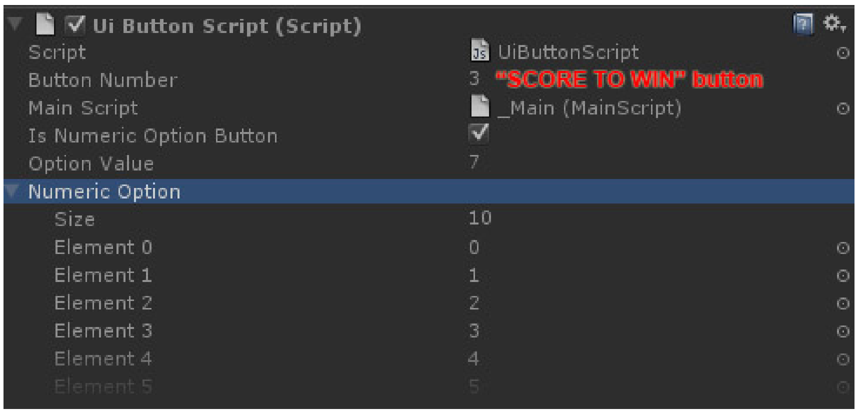
A simple custom UI system example is provided. This UI is based on gameObjects, instead of relying on native Unity “OnGUI” UI which is significantly slower.

All the UI elements are located inside the **Camera** gameObject.



```
Hierarchy
Create ▾        All
_Main
▼ AirHockeyJS
   ▼ Camera
      ▼ UI
         ▶ UI_GameOver
         ▶ UI_Pause
         ▼ UI_StartScreen
              UiButton_1 "1 PLAYER"
              UiButton_2 "2 PLAYERS"
            ▶ UiButton_3 "SCORE TO WIN"
            ▶ UiButton_4 "DIFFICULTY"
      ▶ UI_InGame
      ▶ UI_Score
     ColliderBounds
     ColliderInputRaycast
```

UI buttons are controlled with **UIButtonScript**, attached to them.
Each button has a unique ID number, and optional numeric extension.



Ui Button Script (Script)	
Script	UiButtonScript
Button Number	3 "SCORE TO WIN" button
Main Script	_Main (MainScript)
Is Numeric Option Button	✓
Option Value	7
Numeric Option	
Size	10
Element 0	0
Element 1	1
Element 2	2
Element 3	3
Element 4	4
Element 5	5

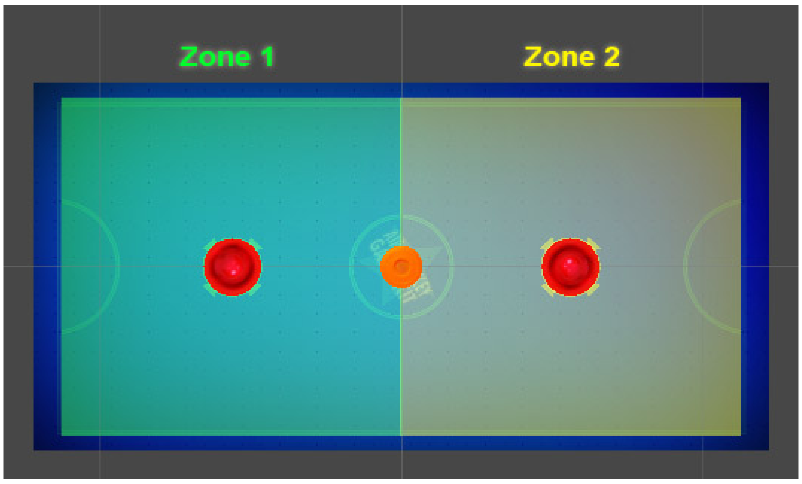
Parented to the camera is also attached **UI_Score** and his script **UIScoreScript** which controls the in-game score display.

Zones

Zones depicts the two sides of the playing surface.

Each zone contains several elements relative to it : **Mallet**, **MalletTarget**, and **PuckTarget**. These elements are described bellow.

Zone objects are displayed in **Debug Layer**.
Zones and their elements' display are differentiated by color.



Mallets

Each mallet contains two scripts to act as a CPU or an user.

The CPU (AI) control script is named **MalletCpuScript,** and the user control script **MalletPlayerScript**. These scripts use "target objects" described bellow.

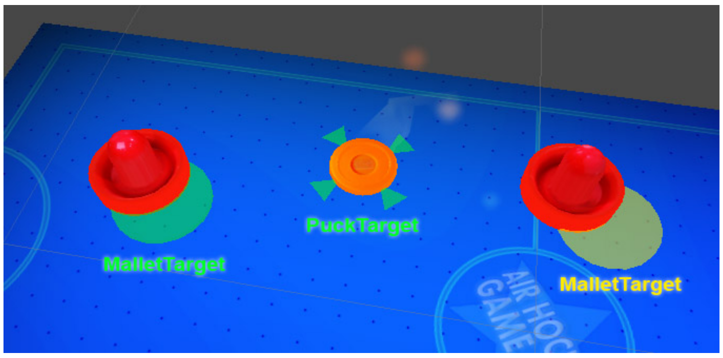
The CPU script uses **PuckTarget** to follow the puck, whereas the player script uses **MalletTarget** to compute mallet input position.

Target objects

In each **zone**, beside the **mallet** gameObject, you'll find two target dummy objects : **MalletTarget** and **PuckTarget**.

These objects are used to reference mallet and puck positions instead of directly using them. This is useful for debugging and tweaking.

Target objects are displayed in **Debug Layer**.

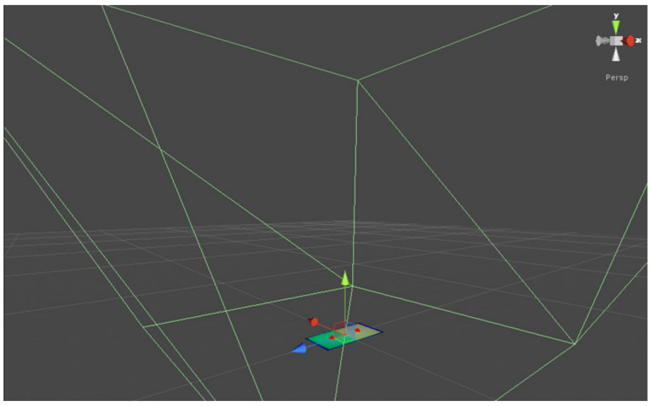


Bounds collider

The gameObject **ColliderBounds** contains the collider wich defines the playing surface's boundaries.

Input Raycast collider

The gameObject **ColliderInputRaycast** contains the collider which receives the mouse/touch inputs. It is assigned to the **InputRaycast Layer**.



Puck

Attached to the **Puck** gameObject, **PuckScript** controls the rigidbody's velocity limit and prevents the object to go out of the playing surface.

Goals

Each goal is controlled by is **GoalScript** and contains a **GoalAnimator** children object.

GoalAnimator object is a plane with a fading texture animation attached to it.

GoalScript simply triggers collision between puck and goal's box collider, then send the goal event to **MainScript**. It also plays **GoalAnimator** fading texture animation.

Additional notes

Included in the Main script, you'll find an example of gameplay presets, displayed by Unity native “OnGUI” UI. This is aimed to give an example on how to customize the gameplay.

Also, the “DIFFICULTY” option button show you a quick implementation of a game option, and actually just change the mallet cpu script's speed.

The assets are arranged in layers for convenience. All the debug assets are linked to **Debug Layer**. The UI assets are linked to **UI Layer**.

Zones debug elements, including raycasts are differentiated by color. **Zone1** elements are displayed in green, **Zone2** ones in yellow.

You should find all the remaining needed informations in the scripts, where nearly each line is commented.

Support

If you have any question you can send an e-mail at support@eti-software.com

Follow ETI

ETI on facebook : http://www.facebook.com/ETISoftware
ETI on Twitter : http://twitter.com/ETISoftware