# TD 2 - Experiment Prototype

**Vincent Casamayou**

---

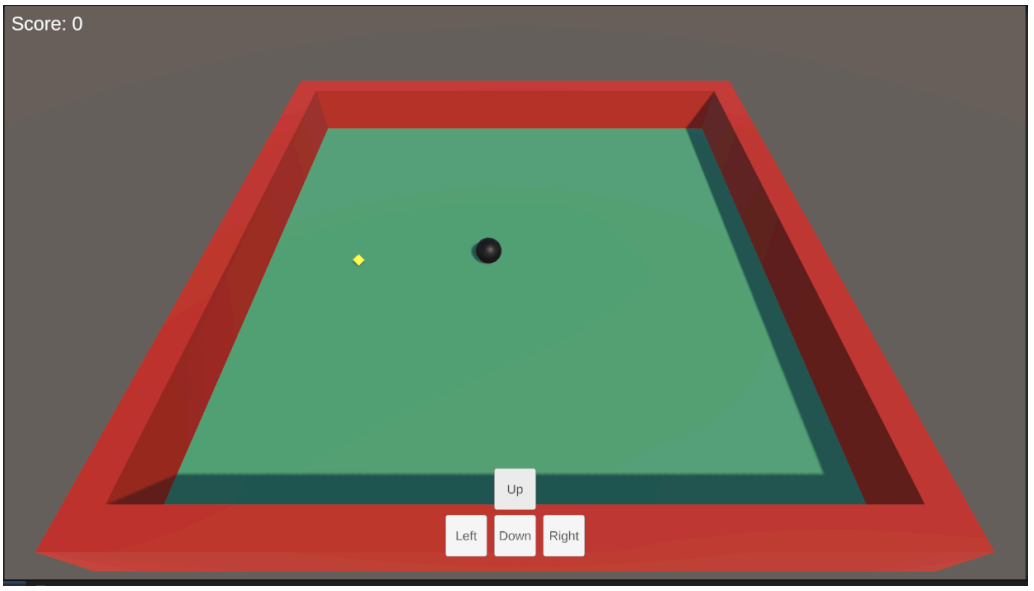
In this TD, you will use the project that you did on TD 1 (Roll-A-Ball). You will transform it into a prototype to perform an experiment on HCI (Human-Computer-Interaction)

The goal of the experiment is to gather 12 objects on the map, by moving the ball.
We want to evaluate two types of interactions to move the ball:

- Classic movement with the keyboard
- Movement by clicking on a UI with the mouse

To achieve this, we will divide the implementation in multiples steps:

1. Implementation of the second interaction
2. Progress of the experiment
3. Retrieve of metrics.




# Part 1 - Interact with the Interface

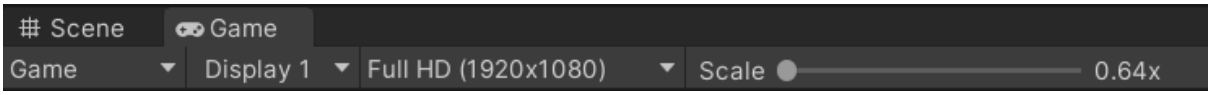
---

For this part, we will implement the alternative controls of the ball.
Instead of using directional arrows on the keyboard, the user will have to use 4 buttons in the interface, to apply forces to the ball.

### Step 1 - Creating the interface

Inside the `Canvas` who handled the score previously, create 4 `Buttons`. As for the Text, you can use Buttons from menu UI/Legacy.
Place them where you want on the scene, but make sure they are easy to use.

To make sure they are correctly displayed, open the **Game** panel.




With this control bar on top (by default in **Free Aspect**), you can adjust the resolution in which you want to launch your game. Choose a resolution (1920x1080 is pretty standard) and adjust your interface accordingly.

## Step 2 - Implementing movement

In the script that manages the movement of the ball, you simply need to add two methods that respectively manage the horizontal movement and the depth movement (on the z axis) of the ball. These methods will take a float as input, which we can adjust directly in the Button's Inspector.

You can implement these methods in exactly the same way as for the previous movement, by applying a force to the RigidBody on the ball.

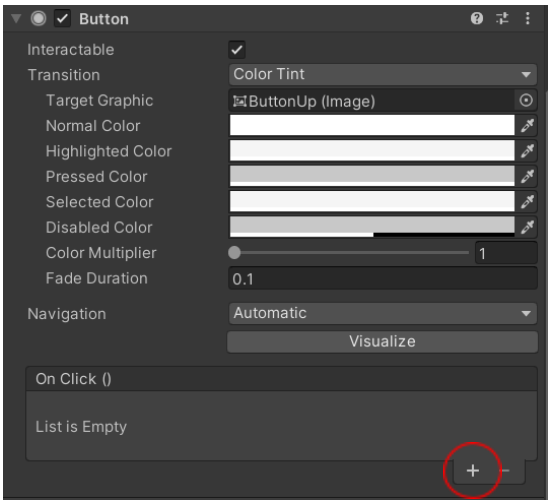
Don't forget to add a condition to **activate/deactivate the classic control mode** and to **add an interface to indicate to the participant which control mode is currently being used** (via text or other means)/.

## Step 3 - Linking methods and buttons

To link the methods you've created to the click of a button, you don't need to create a script.
Using the Inspector of the GameObject containing the Button, you can click on the '+' in the 'OnClick' window to add an action to be performed when the button is clicked.
You'll need to:

- Specify the GameObject (by drag-and-drop) which contains the Component with the method to be performed.
- Specify the method to be performed.

You can also define an input for this method (if it has one).



# Part 2 - Progress of the Experiment

For our study, participants will have to **move the ball to grab each objective.**
However, if we only want to evaluate the method of controlling the ball, and not the strategy used to recover the collectibles, we will have to **control the study's progression.**

Instead of making all the cubes appear at the start of the experiment, it would be interesting t**o make the first one appear, then make the appearance of the next one conditional on the previous one being collected.**
In this way, we could generate a sequence of appearances that would be the **same for each participant**.

Once again, this part can be broken down into several steps:

1. Initialisation of sequence variables
2. Implementation of the sequence
3. Add a tutorial phase.

You are going to transform the `GameManager` script, which was used to manage the score and the victory condition for Roll-A-Ball, so that it manages the course of the experiment.

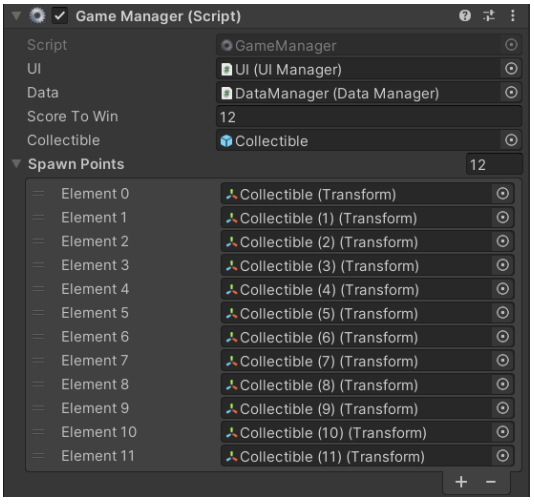
## Etape 1 - Initialisation of sequence variables

You will need the following variables in this script:

- Reference to the `Prefab` of the collectible.
- A list of the 12 positions at which cubes should appear.

For the first, `Prefab` can simply be defined as a `GameObject`, and be added via the Inspector (click and drag).

For the second one, there are several ways of making this list. You could simply retrieve the coordinates of each position and input them into the list.
Another methods would consist to create empty `GameObjects` and to place them at the desired positions. An empty `GameObject` isn't really empty at all. Each `GameObject` has a `Transform` which contains information such as its position, rotation and size. (Doc)
By creating a list of `Transforms`, then referencing each `Transform` in the empty `GameObjects` you've created, you can control the position of each point of appearance simply by moving them.



### Etape 2 - Implementation of the sequence

The first method to implement,  will be to **make a collectible appear.**
This method will use the method ***Instantiate*** of  `MonoBehaviour` **(Doc).** All you need to do is tell it which `GameObject` to instantiate (in this case, the `Prefab` of the collectible). You can also define its position in the ***Instantiate***, or on the next line.
For its position, you need to **find a way to instantiate the collectibles in sequence**, and therefore to go through the list of positions defined previously.

This method will then have to be called up **at the start of your study** (to make the first cube appear) and each time the participant picks up a cube until the last one.

### Step 3 - Tutorial phase

In a study, it is important to not make the participant start directly with the study. In order to not to bias the results with how well the participant grasps the controls.

Therefore, you need to implement a similar phase in your study. To do this, you can repeat the previous steps, but instead of having the cubes collected in the 12 positions. Set them up in the 4 corners of the arena.

It is also **at the end of this phase that a control mode will be chosen automatically by the programme.**

It's important not to make the transition from this tutorial phase to the study automatically, but rather when the participant is ready.
**At the end of the getting started sequence, add a button to launch the study sequence.**

## Step 3 - Retrieve metrics

---

We want to know **how long it takes to retrieve each cube** to determine which method is the fastest.

We're going to need a new script to manage data acquisition.

Create a DataManager script, and place it on an empty object in your `Scene`. Don't place all your 'Manager' scripts on a single empty object, for reasons of organization and good practice.

This script will contain an empty **float** list which we will use to store the values for our study.

Next, you need to create a method that will **calculate the time taken to collect each collectible** and then add it to the list. The idea behind this method is to take care to calculate the time, either **since the start of the study** (to get the cumulative total), or **since the last collection**. Using `Time.time` should help you (Doc).

You can then call this function when a collectible is picked up.

# Bonus - Your own study

---

Now that you've produced the prototype for this study, it's your turn to propose a study!

There are many other interactions you can study in this context and many other types of metrics you can extract from this type of experimentation.

In this bonus part of the TD:

- Search and implement a new type of interaction.
- Search and implement another measure to be extracted.