Immersive Media Programming – Lab 2: Unity scripting 1

In this lab, we practice basic Unity scripting with focus on:

- Instantiating and destroying GameObjects
- User input
- Moving a Rigidbody
- Detecting collisions
- Using triggers

Submission instructions:

- For each task, record a short video that demonstrates all the required features of the task. Name the video according to the task, like Task1.mp4.
    - Place all videos in a separate folder OUTSIDE the project folder.
- Close Unity.
- Make a backup copy of your project folder.
- Remove unnecessary folders from your project folder: Build, Library, Logs, obj.
- Zip the project folder AND the video folder into a single ZIP file.
- Submit the ZIP file via AjouBB.

Deadline: March 24 at 23:59

Maximum points: 17p

NOTE: there are bonus tasks that give you bonus points that can be used to recover points that you lose in other tasks.

Task 1: Symbol instantiation (4p)

Demo: Lab2_Task1.mp4

a) Scene setup (1p)

1. Create new project with name “Lab2-[your ID]” (e.g. Lab2-201912345).
- Then create a new scene “Task1”.
- Find a nice skybox asset and apply to the scene (see example in the lecture video).
- Create a game object using one of the basic models (e.g. sphere or cube)
    - Create a new material and apply to it
    - Save it as a prefab named “Dot” and remove from the scene.
- Create an empty GameObject called “Symbol”.

b) Instantiate a symbol (2p)

- Add a script to Symbol:
    - Instantiate the Dot prefab multiple times to draw a two-dimensional symbol. You can decide the symbol yourself, but try to keep it fairly simple (but it should be more than just a line).
        - Try to “draw” the symbol using one or more loops instead of a lot of single statements (see p.25 in lecture notes for a circle generator example)
    - All Dots should be children of the Symbol GameObject.
        - Assuming that an instantiated dot is assigned to a variable “dot”, you can set the dot’s parent by:

```
// assign Symbol’s parent to be the parent of the dot
dot.transform.parent = transform;
```

    - Rotate the Symbol GameObject around the y-axis over time.

- Adjust the camera so that it shows Symbol with all the dots.

c) Add user controls to Symbol (you may add new variables as needed) (1p)

- Increase rotating speed (angle) when the user presses 'd'.
- Decrease rotating speed (angle) when the user presses 'a'.
- Destroy Symbol when the user presses right mouse button.

Hint: to change the scale, use the localScale property of the Symbol's transform:
https://docs.unity3d.com/ScriptReference/Transform-localScale.html

[BONUS] d) Complete the following features to earn bonus (1p)

- Increase Symbol's scale when the user presses 'w'.
- Decrease Symbol's scale when the user presses 's'.
- Use multiple materials to draw the symbol.
  - Method 1: create multiple dot prefabs with different materials.
  - Method 2: define multiple materials for Dot and choose the material at runtime.

Task 2: CatchMe (4p)

Demo: Lab2_Tasks2-5.mp4

a) Create new scene "CatchMe". (1p)
- Add a ground plane, scale it up.
- Add walls (cubes) around the ground.
- Add a sphere ("Player") with Rigidbody above the ground.
- Create and apply materials to the gameobjects.
- Move and rotate the main camera to show the ground from a top-down view (the camera shows only a part of the game area!)

Figure 1: Game view (including Tasks 3-4)

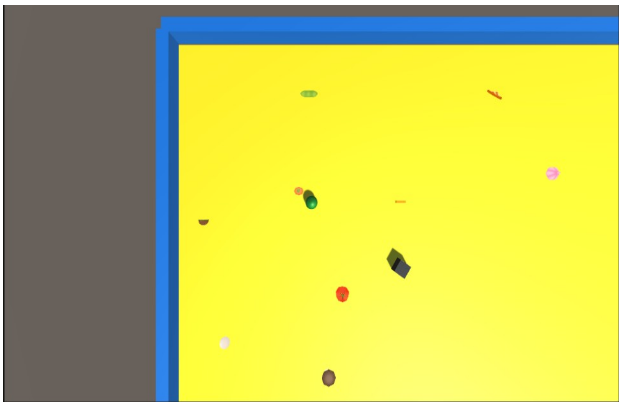
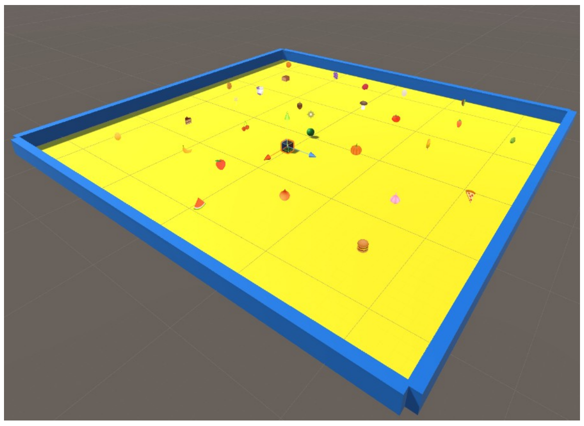


Figure 2: Scene view (including Tasks 3-4)



b) Add new script PlayerController to your Player game object. (2p)

- Add force to Player's Rigidbody when WASD keys are pressed. For example, if A is pressed, the Player sphere rolls to the left.
  - Declare a variable to store the applied force.
  - Read input in Update(), and apply force in FixedUpdate()!

- When "p" is pressed, pause the gameplay. When "p" is pressed again, resume the gameplay.
  - https://gamedevbeginner.com/the-right-way-to-pause-the-game-in-unity/

- Add a variable to store the player's health value (for later use).

c) Make Main Camera follow the player. (1p)

- Here is how to: **https://www.maxester.com/blog/2020/02/24/how-do-you-make-the-camera-follow-the-player-in-unity-3d/**

Task 3: Edible items (5p)

Continue the CatchMe scene.

a) Implement edible items (2p)

- Find at least 5 different prefabs/models (e.g. from Asset Store) that represent edible items. Add them to your scene.
  - Make sure that the edible items have a collider with “Is Trigger” enabled.
- Assign a tag like “Food” to each edible item (used when the player eats them).

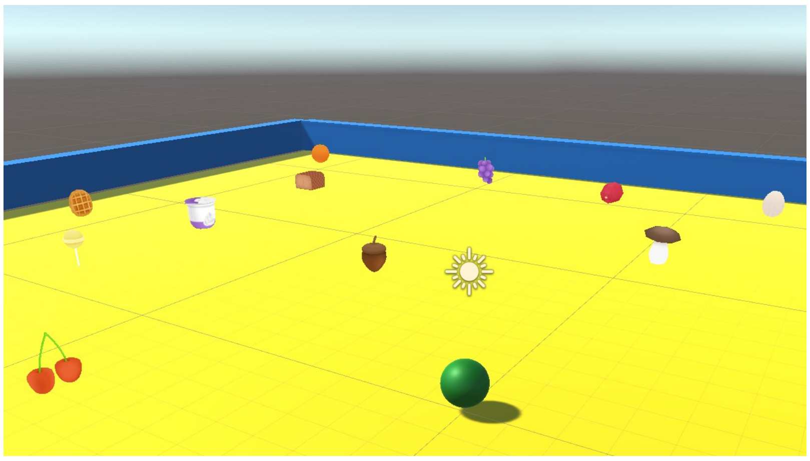
- Make each edible item rotate around its X and Z axes.
  - Method 1 (a bit inconvenient): add a rotation script to all edible items
  - Method 2: add just one script that finds all edible items on the scene and rotates them in a loop.

- Distribute at least 20 edible GameObjects around your scene
  - Hint: if you need to copy the items, select one from the Hierarchy view and ctrl+d to duplicate.

b) Implement non-edible items (1p)

- Find also some models/prefabs for non-edible items. Repeat the same steps as in (a) except:
  - choose a different tag name.
  - The number of non-edible items can be lower.

Figure 3: Player (green sphere) and some edible items



c) Implement trigger collision detection (eating) (2p)

- Override OnTriggerEnter() in PlayerController (see the trigger example in lecture notes).
  - When an edible item is touched.
    - Play a sound of success.
    - The edible item is destroyed.
    - In Console, print how many edible items have been collected so far.
  - When a non-edible item is touched,
    - Play a sound of failure.
    - Player loses some health.
    - In Console, print a message to express:
      - The unpleasant feeling of trying to eat a non-edible item
      - How much health is left.

Task 4: Enemy (2p)

Continue the CatchMe scene.
a) Implement Enemy (3p)

- Add a cube (or another model) and assign “Enemy” tag to it.
- Make Enemy follow the player (see example in lecture notes).
- Add collision detection to Player
    - If Player collides with Enemy, give Player some damage and print a message in Console about the remaining health.
    - If Player’s health reaches 0, stop the play mode (i.e. game over)
        - `UnityEditor.EditorApplication.isPlaying = false;`

BONUS Task 5: Additional CatchMe features (max 2p)

a) Implement power-ups (2p)
- When space is pressed, enable a speed booster for a short time (e.g. 1 second). The speed booster increases the force that is applied to the player.
    - An example of implementing a power-up timer using coroutine: https://forum.unity.com/threads/powerup-for-a-certain-amount-of-time.546758/
- When “h” is pressed, make the Player invisible for a short time (e.g. 3 seconds).
    - The enemy stops following the player during invisibility.
    - Hint 1: Renderer component has the “enabled” property that you can modify.
    - Hint 2: Use the same coroutine technique than with the speed booster above.

b) Jumping (1p)
- When Fire1 is pressed, make the player jump (upward impulse). This way the player could jump over the enemy.
- Only allow jumping when the player is on the ground (detect collision with the ground plane).
- Make the walls high enough to prevent the player from jumping outside the game area.

Task 6: Reflection (2p)

Write a short reflection text based on your lab experience (e.g. a text file or a Word file) using the questions below. The positivity or negativity of your answers does not affect scoring.

Please answer to these questions (at least):

- What did you learn? What did you know already?
- What was difficult in the assignment? How did you overcome it?
- What was good about the assignment?
- Do you have any suggestions to improve the assignment? Please tell us!

Save your reflections as Reflections.txt/docx to the root of your project.