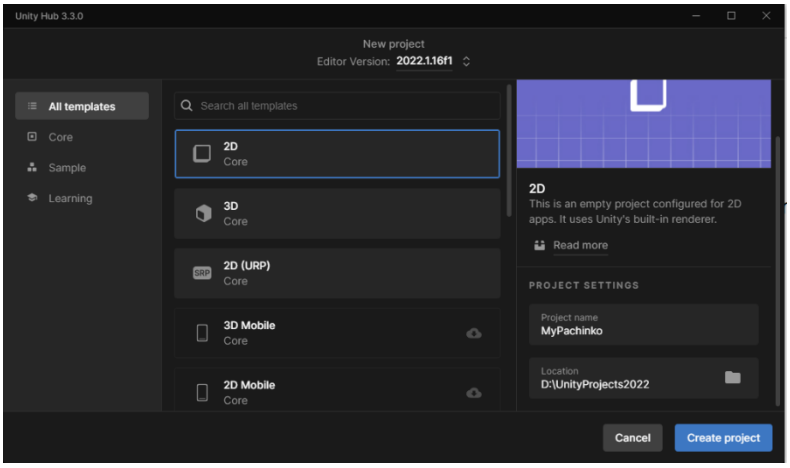
Current Trends in Gaming

Module 2

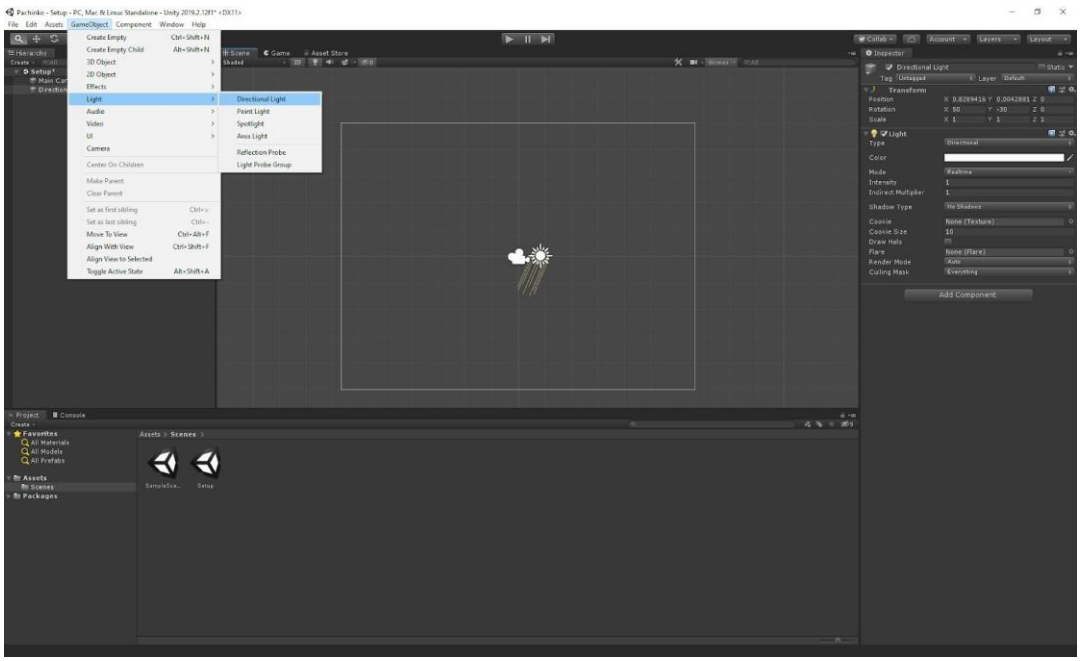
Make a clone: Pachinko

In this exercise you will implement a simple version of Pachinko using the Unity game engine. For reference, an executable of the completed version of this exercise will be uploaded to the Canvas.

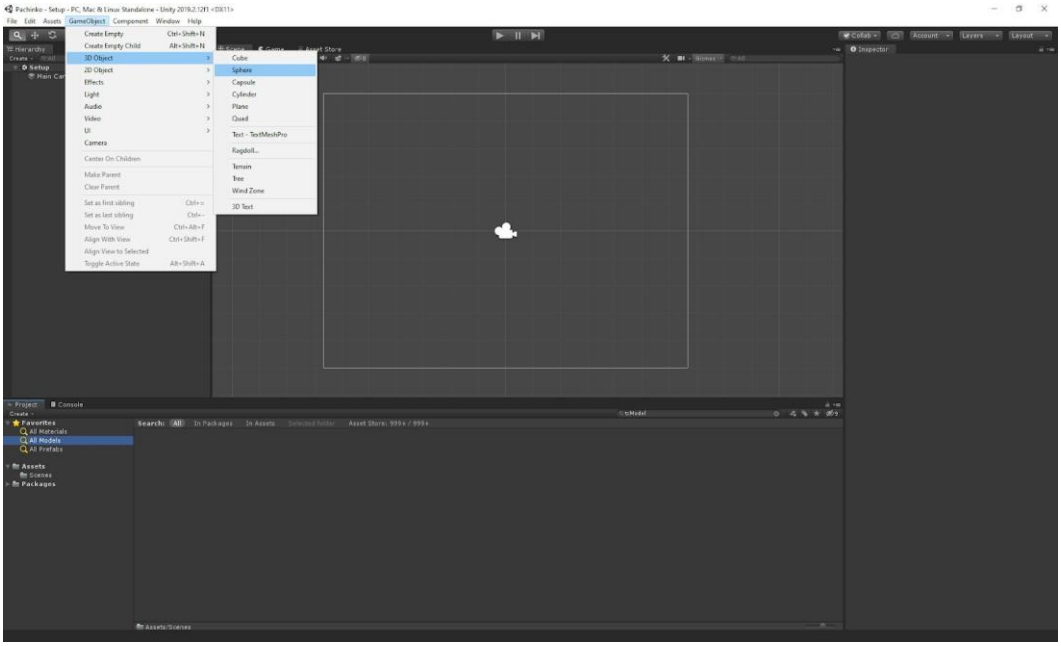
Task 1. Start a new project in Unity. Use the 2D template.



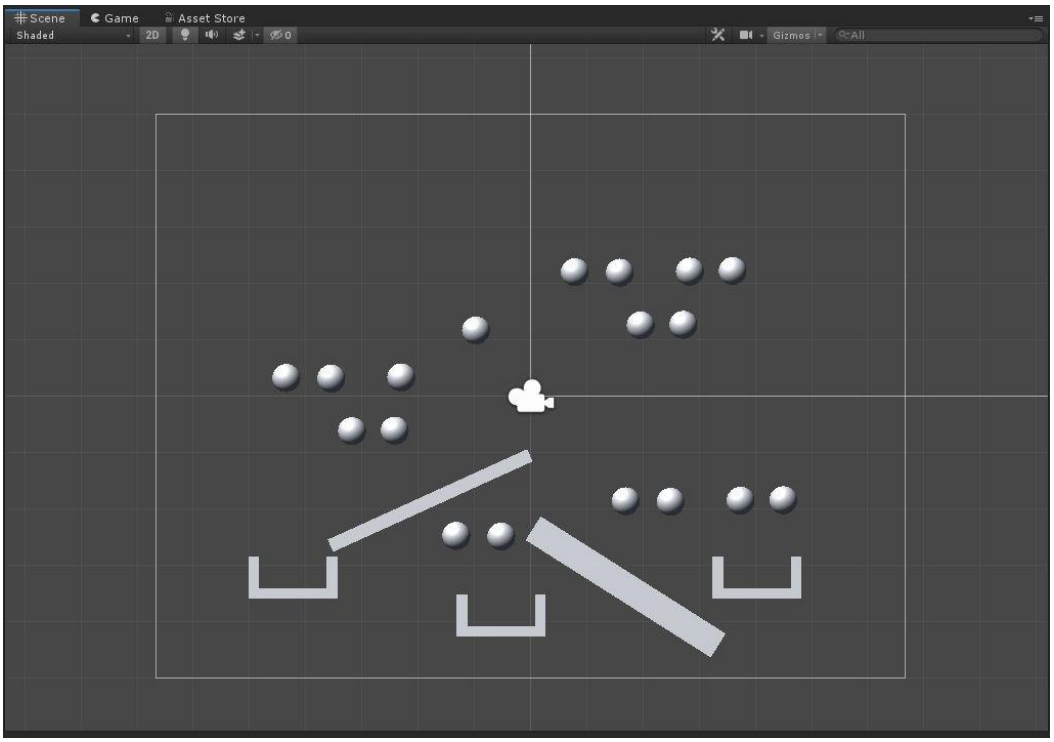
Task 2. Add a light source to the scene.



Task 3. Add new GameObjects to the scene view.



Using spheres and cubes, create a setup similar to what is shown in the image below (you are free to set it up however you want, as long as you include “baskets” and obstacles, this is just a suggestion):



Since we are in 2D, make sure the z component of every GameObject’s position is always 0, otherwise you will quickly run into issues with collision detection.

Task 3a. Create an Empty GameObject and name it Game, attach to it a new script named GameScript. Have this script generate 100 little balls somewhat uniformly distributed above the scene at the start of the game.

To make the balls have bounce when they collide, create a Physic Material, set its Bounciness to 1, and attach it to the ball's Sphere Collider.

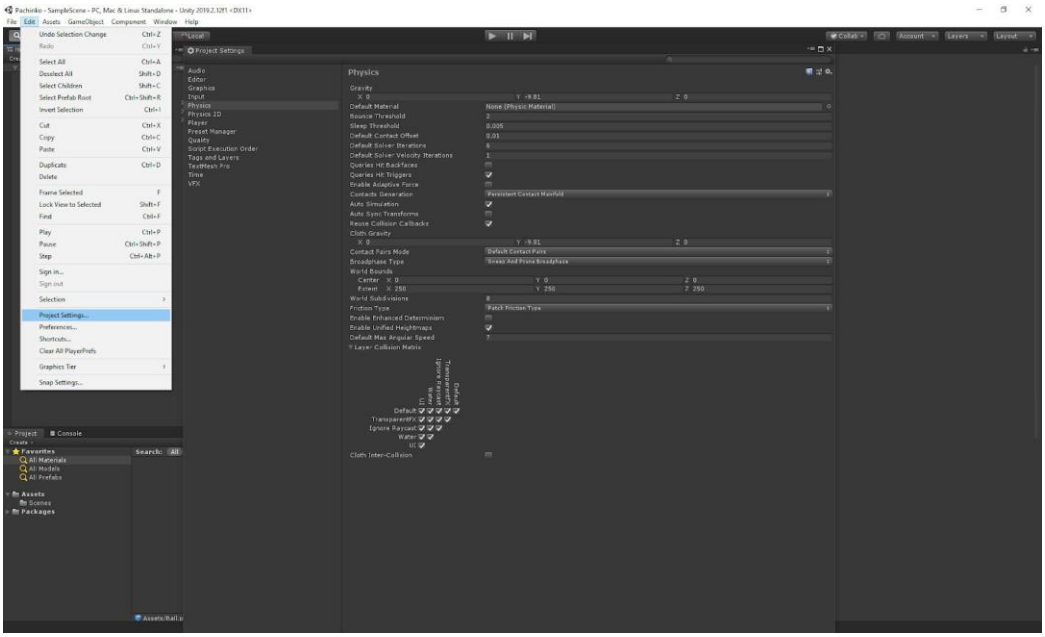
To create a "blueprint" for the ball that will be generated by the script, you can create a Prefab:

1. Create a Sphere GameObject with position (0,0,0).
2. Add to it a Rigidbody and the Physic Material you created.
3. Drag the Sphere GameObject from the Hierarchy window to the Assets in the Project window.

In the GameScript use the Instantiate(...) method to create the ball using the Prefab.

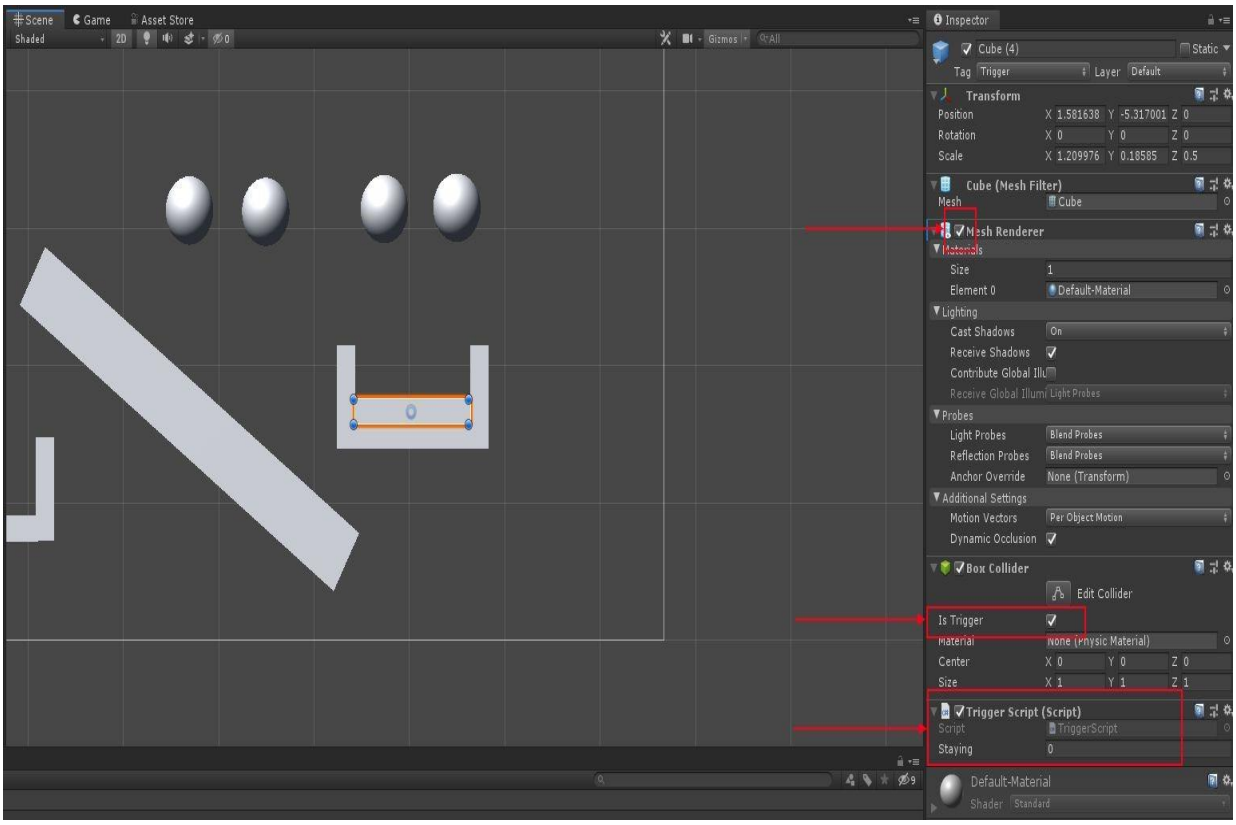
To reference the ball Prefab in the script you can define a public member variable for the ball in the GameScript class, then you will be able to drag the ball Prefab onto the script component in the Inspector window.

Task 3b. Experiment with the Default Solver Iterations in the Project Settings.



What happens when the number of iterations is low (e.g. 1) and what happens when the number of iterations is increased (to e.g. 12)?

Task 4a. Now we want to be able to detect when a ball falls into one of the “baskets”. Using Unity’s built-in collision detection system we can detect overlaps between a ball and an invisible cube object (pictured below) placed inside the basket.
If we set the Box Collider on the cube object to be “Is Trigger”, the ball will pass through the cube, but the cube will report when something overlaps with it.
To make the cube invisible simply turn off the check box for its Mesh Renderer (as highlighted in the image below).
Make sure the invisible cube fills the entire space inside the basket.

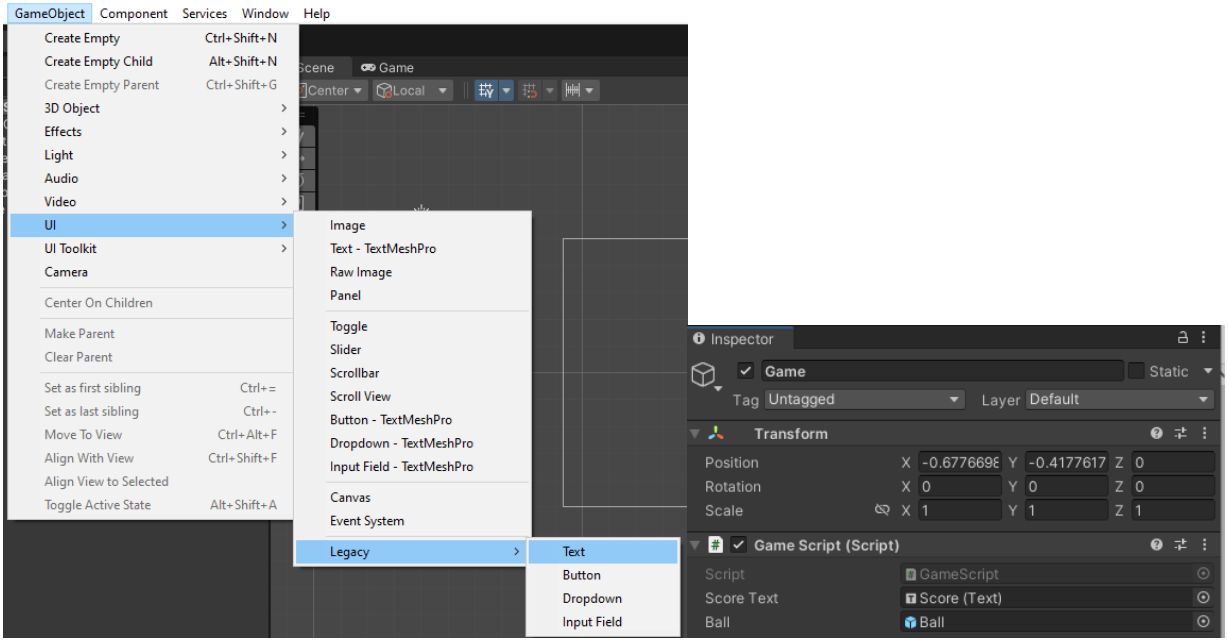


Task 4b. The overlap information can be accessed using OnTriggerEnter(Collider other) and OnTriggerExit(Collider other) in a script:

https://docs.unity3d.com/ScriptReference/Collider.OnTriggerEnter.html.

Create a new script named TriggerScript and attach it to the cube object. Use OnTriggerEnter(...) and OnTriggerExit(...) to keep track of whether anything is overlapping with this particular invisible cube object.

Task 5. Add a new Text object and attach it to GameScript (which is attached to the Game object) as a public variable in the script.



Have GameScript keep track of how many balls have been caught in baskets and display it in a Text object at the top of the screen in real-time.