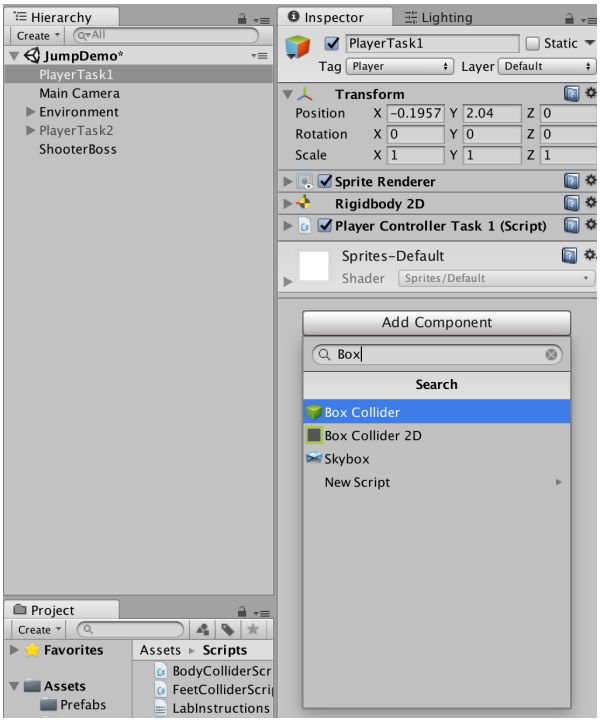
COLLIDER LAB :

Preface: Colliders are one of the basic building blocks of the game. Two possible uses are as physical boundaries and as event triggers, as you'll learn in this lab.

To add a collider to a gameobject, select the game object in the hierarchy, then in the inspector, click the ADD COMPONENT button. In the search bar, type collider, then select the Collider that you want to use.

For this lab, you will be using Box Collider 2D and Circle Collider 2D.

GameObject -> Add Component -> collider (search bar) -> Box Collider 2D

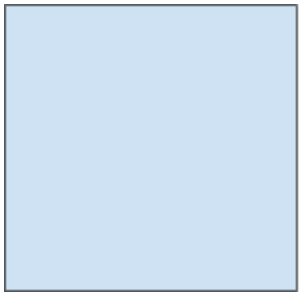


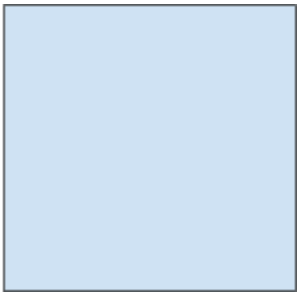
Picture 1 : Adding a Box Collider 2D Component

In this lab, you will be implementing parts of a naive jump function and an enemy turret that shoots projectiles at the player.

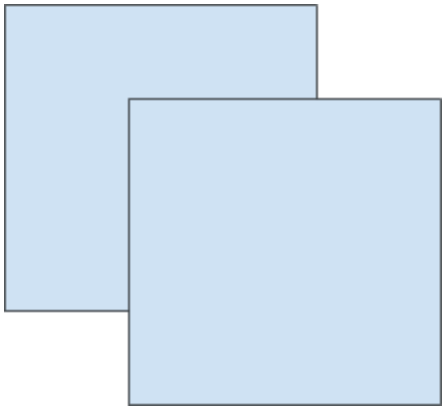
INTRO

If you already understand how colliders work, the OnCollisionEnter/Exit/Stay functions, and OnTriggerEnter/Exit/Stay functions, feel free to skip this section.

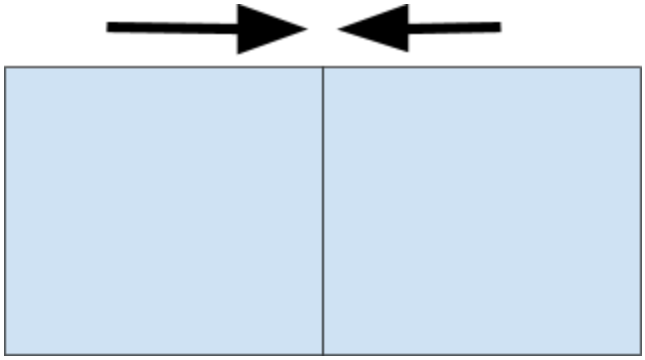




Imagine these are two gameobjects, without colliders. When they run into each other, since there are no colliders, there are no physical collisions, which causes some unexpected behavior.

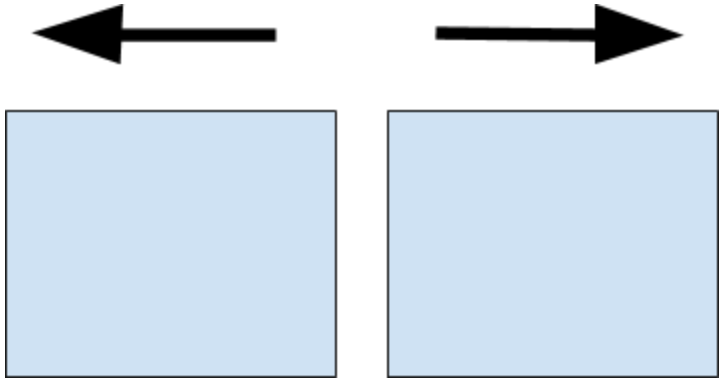


With Colliders (Physical, NOT trigger) these two squares will stop when they come into contact. The frame when they come into contact, OnCollisionEnter() is called.



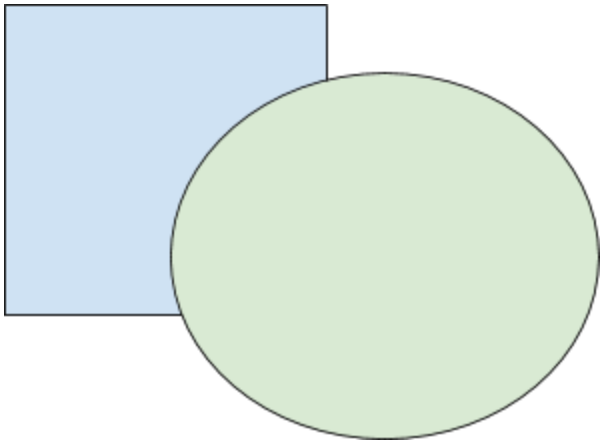
For every frame where they are stuck together, OnCollisionStay() is called.

On the frame where they leave contact, OnCollisionExit() is called.

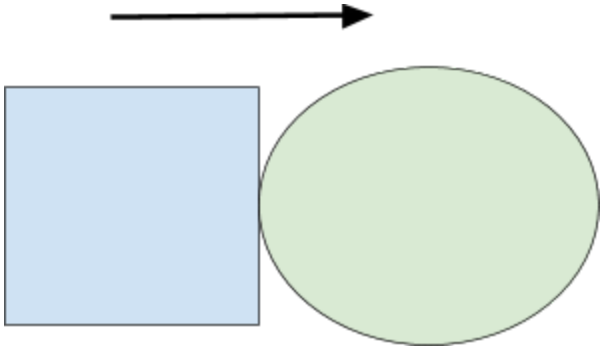


That’s all for physical colliders

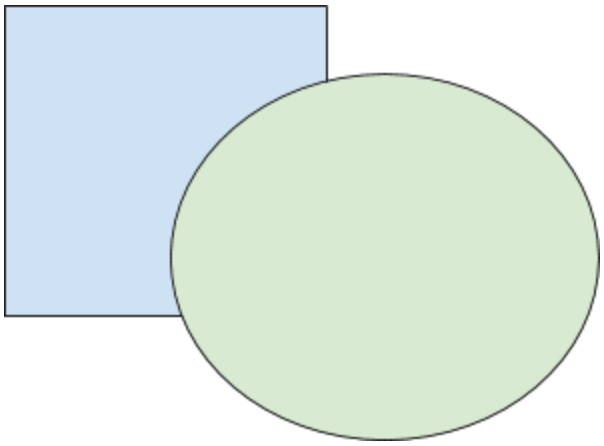
Triggers: If you have a trigger collider (Green), it will not physically collide with any other colliders (blue). This allows for cases like this.



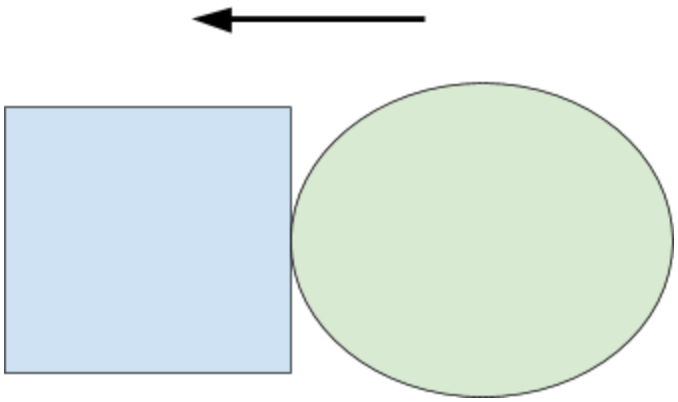
Similar to OnCollisionEnter(), OnTriggerEnter() is called on the frame where another collider comes into contact, or enters the trigger



While the collider remains in the trigger, OnTriggerStay() is called:



Finally, the frame when the collider leaves the trigger, OnTriggerExit() is called



It's important to note that these collider functions will only be called if at least one of the gameobjects involved in the collision has a Rigidbody component attached. No Rigidbody = No collision!

That should be everything you need to know about colliders! Good Luck!

Task 1:

For this task, you will need to enable the following: PlayerTask1, MainCamera, Environment. Disable all the others by clicking on their game objects in the hierarchy and unchecking the box to the right of the colored cube in the inspector.

a) Hit the play button. The player should fall right through the floor (this would be problematic in most games, so we should fix this). Add a Box Collider 2D to the PlayerTask1 game object and size it appropriately by modifying the variables in the inspector OR clicking Edit Collider and adjusting the size of the collider in the scene view.

b) Hit the play button again. The player should now be able to collide physically with all the platforms and walls in the scene. Try moving around and jumping. **Are there any times when you can jump when you shouldn't be able to?** Be able to answer this as part of the checkoff.

c) In the scripts folder, open PlayerControllerTask1.cs. This is the script that controls the player. Modify the sections labeled 'Task 1' to allow for the following:

1. You should only be able to move your player if you are touching the platform, wall, or floor.
2. You should only be able to jump if you are touching a platform, wall, or floor.

Some games allow for wall jumping. In later tasks we will fix this. **For now, be able to answer why our player can jump while touching these locations.**

Task 2:

For this task you will need to enable the following: PlayerTask2, MainCamera, Environment. Disable all other game objects.

a) For this task you will be implementing a smarter check for jumping. You should notice that the PlayerTask2 Game Object itself does not have a box collider. You can accomplish this task without a collider on the gameobject. After completing this task, your character should be able to jump ONLY if its feet are touching the ground. There are different ways to accomplish this. If you need hints, look in the *scripts* folder and the children of 'PlayerTask2'.
NOTE: if you accomplish this task the way hinted at, there will still be a couple cases that will cause weird behavior. Can you name one?

Task 3:

For this task you will need to enable the following: PlayerTask2, ShooterBoss, MainCamera, Environment. Disable all other game objects.

a) For this task you will be implementing a shooterboss. The ShooterBoss keeps a list of all targets within a radius (of your choosing). Upon entering the radius, the player should be added to the list. Upon leaving the radius, the player should be removed from the list. The shooting function is already completed, you just have to worry about adding and removing targets from the list.

HINT: this task will require the use of trigger collider. To use a trigger collider, click the "is trigger" box on the 2D collider tab in the inspector. This means that the collider will not cause collisions, but still call the OnTrigger functions.

Checkoff:

1. Show functionality of the task1 player, and that it can jump when touching any physical barrier
    a. Explain why this is the case.
2. Show functionality of the task2 player and that it can jump only when its feet are in contact with the ground
    a. Explain why this is the case.
3. Show that the shooterboss only fires projectiles while you are within its radius.

Task 4 (Challenge, but highly recommended as you might want to make a game that involves shooting people):

a) When you complete task 3, you will notice that the projectiles have no interactions with any other gameobjects. This task is open-ended: give the projectile a collider (trigger or not is up to you) to give it some functionality and interactions.
One idea would be to have it physically push the player if it hits them. Another would be to cause an explosion on impact with anything. Be Creative.

b) Give our player a method of fighting back. This might involve creating health scripts for the boss and/or our player. You can use the ShooterBoss Script as reference for shooting objects if you need.