**Anyone Can Make Games**

From 0 to Flappy Bird in 2 hours

# **Core Concepts from the Lesson**

## Unity IDE - Integrated Development Environment

* Project: Everything in your project. Basically the **Library**
* Hierarchy: Everything that exists in your current **Scene**
* Inspector: Everything to know and edit about the currently selected **Game Object**
* Scene: Your current scene. Imagine this as a level!
* Game: What your player will actually see!

## Colliders and Physics

* Collisions need a Collider2D on every collidable object
* At least one object must have a RigidBody2D, which automatically enables physics on that object

## Component Based Programming: Jumper

* Unity has a unique structure called Component Based Programming
* Every script does one distinct thing, like jumping
* This allows you to slap a jumper on anything that needs to know how to jump

## Object Oriented Programming: Pipe

* An alternate way to program is Object Oriented Programming
* This is where there is one script for the whole “Object” like the pipe
* This method is less reusable but easier to understand!
* You can always mix the two methods

## Game GUI

* All UI in Unity exists inside a Canvas, so make that first
* Experiment with what you can build, start with Text!

# **What’s Next? Challenges!**

## Easy: Block the top and bottom of the screen

* If we bounce too high, we shoot off the top of the screen.
* If we fall too low, we fall past the bottom of the screen and don’t die!
* How can you fix this?
* Hint: It has to do with collision and placing some new sprites!

## Medium: Play Some Sounds!

* We have a music element that plays a song whenever we play the game, but sound effects are great too!
* Find some free sound effects and download them. We have a jump sound in the project folder already. But if you want to make your own, an old (but good!) tool for making your own 8-bits sounds is called [JSFXR](https://sfxr.me/)
* Once you have the sounds in your game, put an AudioSource on any object you want to make sounds. Then attach the new sound in the AudioClip box!
* Then use the code below to make them play that sound, on a script that is attached to the same object, and at a time that makes sense! For instance, this could go in your InputController in the Jump method!

GetComponent<AudioSource>().Play();

## Hard: Reset the score and respawn the player on death!

* When the player touches a pipe, they disappear, and the score keeps counting!
* Try to implement some code that makes the player respawn, the score reset, and the pipes move backward when the player dies!
* For the player, you’ll have to assign a vector value to **transform.position**. You can assign a new vector to the transform’s position with **transform.position = new Vector3(0, 0, 0);**
* You’ll also need a reference to the score manager to reset the score!
* How can you move the pipes backwards?

## Do Some Game Jams!

The absolute best way to learn how to make games is…. to make games! Try some game jams at websites like Ludum Dare or itch.io/jams to test your abilities. Or just build some on your own time! Feel free to reach out for pointers, help, or just for a chat at **team@indieGameAcademy.com**