Respawning

Rise from the digital grave

Contents

Re-spawning our player

In games, we want ways to re-spawn our player when they die.

This will use similar ideas to teleporting, with the added effect of re-enabling our player and letting them play the game again.

Setting up a Player Manager

When our player is “Dead” we often disable the game object to hide it from the game. This also disables all functionality of the object. This means we cannot put the respawn code inside the player script if it is on the player game object. Other things to think about

- What if we want to change characters, or even move to a vehicle?
- What if we need to control multiple units at once?
- When they player dies, how do we manager the scripts if they are disabled?

Because of this, we can set up a Player Manager for each player (have a number or ID if we have multiple players) that keeps track of that players constant information and functionality.

Respawn Manager

We can break down the Player manager into smaller scripts that have dedicated tasks. This will also allow us to use it on other types of game objects to respawn other items that have been destroyed.

Setting up Respawn Manager

Basic Spawning

1. Create a new script called RespawnManager

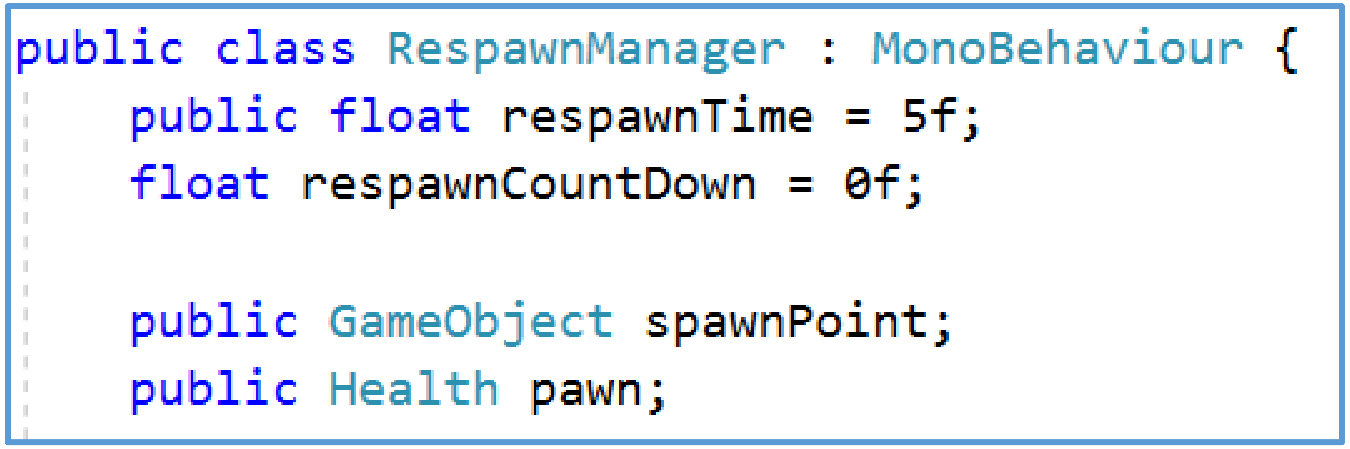
- In the class block

2. Create a public float called respawnTime
   a. Set it to how many seconds you want till respawn
3. Create a float called repsawnCountDown
   a. Set it to 0f
4. Create a public GameObject variable called spawnPoint

This will be the spawnPoint the respawn manager will send the Pawn to when the timer is done.

5. Create a public Health variable called pawn

This will keep track of the Pawn we want to communicate to



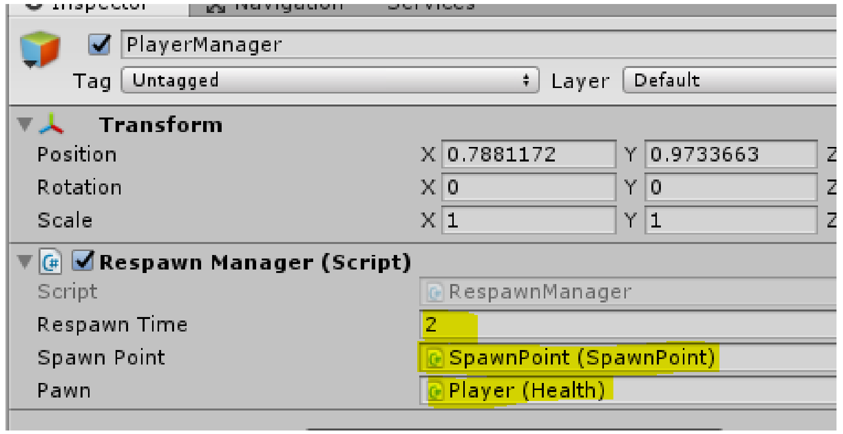
```
public class RespawnManager : MonoBehaviour {
    public float respawnTime = 5f;
    float respawnCountDown = 0f;

    public GameObject spawnPoint;
    public Health pawn;
```

Plugging in the information.

Now that we have the variables we will need. Let's plug in the information

- In the Inspector

6. Set respawn Time
7. Set spawn point
8. Set the pawn with the Player

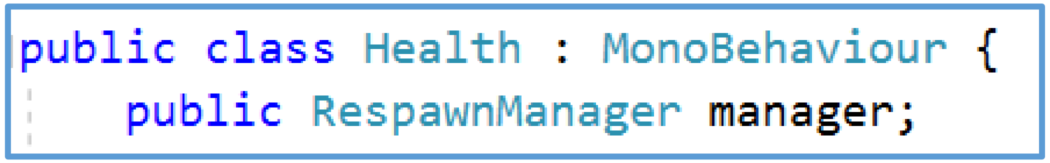


```
PlayerManager
Tag Untagged    Layer Default
Transform
Position   X 0.7881172   Y 0.9733663
Rotation   X 0           Y 0
Scale      X 1           Y 1
Respawn Manager (Script)
Script         RespawnManager
Respawn Time   2
Spawn Point    SpawnPoint (SpawnPoint)
Pawn           Player (Health)
```

Setting up communication between Pawn and Manager

We have the Health variable pawn, but that is a one-way communication. For now, we will send out a call to the pawn so it sets who its manager is.

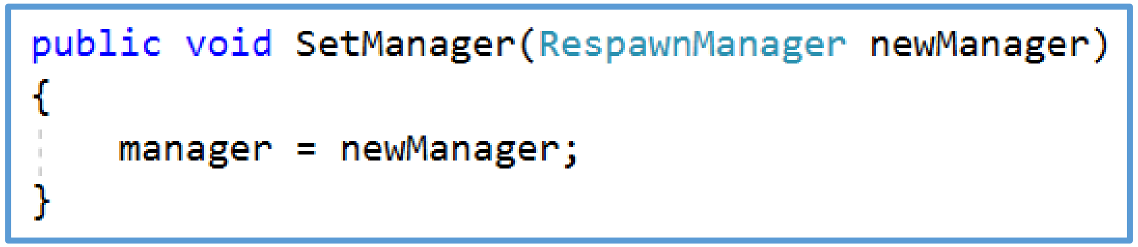
- In the Health script
  - In the Class block
    9. Create a RespawnManager variable called manager



```
public class Health : MonoBehaviour {
    public RespawnManager manager;
```

Now we need a way to set that variable

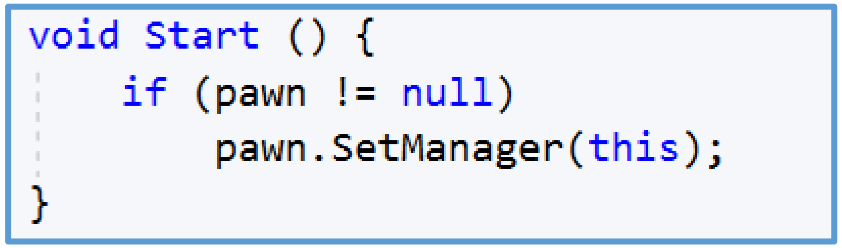
10. Create a public void function called Set Manager
    a. Pass in a Reference to the RespawnManager called newManager
- In the function
11. Set manager to equal newManager



```
public void SetManager(RespawnManager newManager)
{
    manager = newManager;
}
```

Now that this is sorted, we can call this function in the start function of the Manager

- In the Manager
  - In the Start function
12. Check if the pawn variable is not null
  - If it isn't
13. Call pawns SetManager function
    a. Pass in this – the RespawnManager script



```
void Start () {
    if (pawn != null)
        pawn.SetManager(this);
}
```

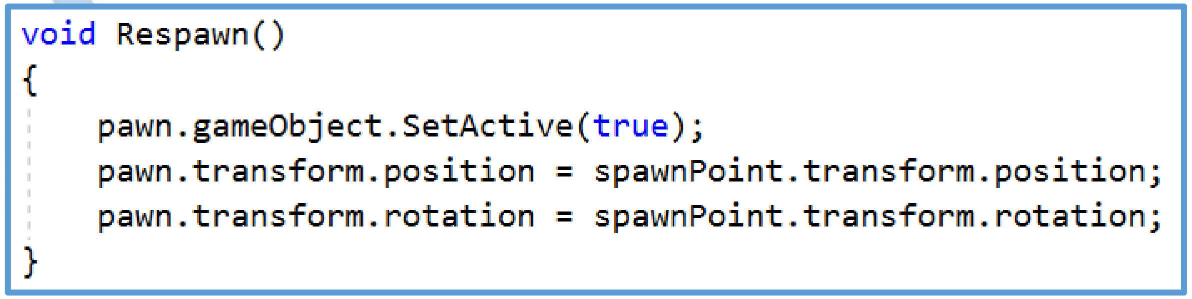
Respawning

With this version of respawning, we are not creating an asset, we are resetting it. So we need to activate it and position it

14. Create a function called Respawn

- In the function

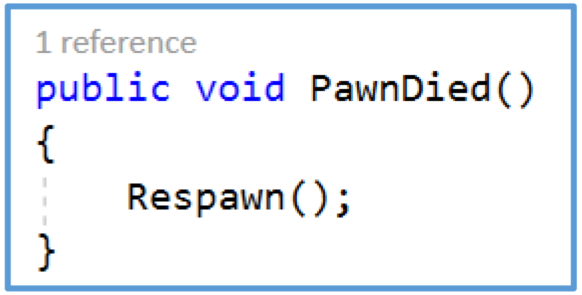
15. Set pawns gameObject to call SetActive to true
16. Set pawns position to spawnPoints position
17. Set pawns rotation to spawnpoints rotation



```
void Respawn()
{
    pawn.gameObject.SetActive(true);
    pawn.transform.position = spawnPoint.transform.position;
    pawn.transform.rotation = spawnPoint.transform.rotation;
}
```

Catching the Death

Our Health script automatically knows when it dies, but the manager does not yet. We need a way to tell our Respawn manager that we have died so we can respawn



```
1 reference
public void PawnDied()
{
    Respawn();
}
```

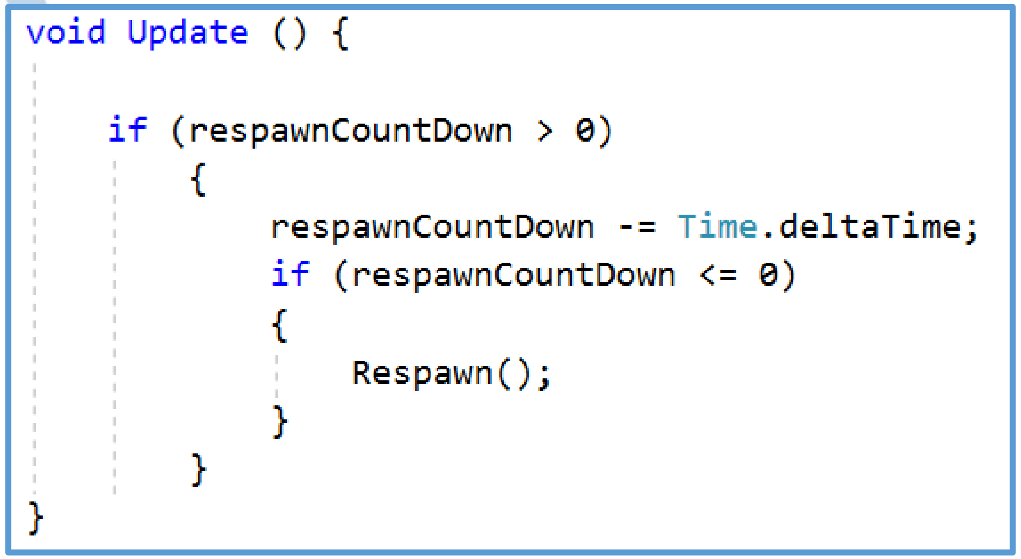
Respawning

Rise from the digital grave

Way 1 - Timed Respawn

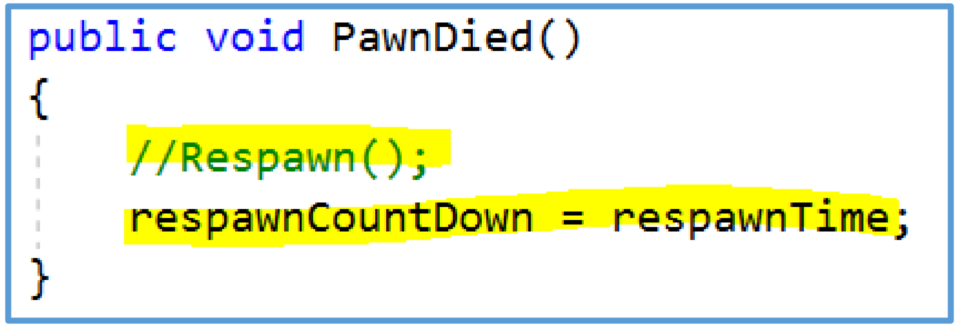
We can have a timer, that counts down when it is over 0. If it is less than 0 it ignores everything inside preventing constantly respawning.

When the timer counts down to 0 it will then call the Respawn function.



```
void Update () {

    if (respawnCountDown > 0)
        {
            respawnCountDown -= Time.deltaTime;
            if (respawnCountDown <= 0)
            {
                Respawn();
            }
        }
}
```

As we want to respawn in the timer, we down want to do it automatically. Instead we want to set the respawnCountDown to a value above 0.



```
public void PawnDied()
{
    //Respawn();
    respawnCountDown = respawnTime;
}
```

Way 2 - Button click Respawn

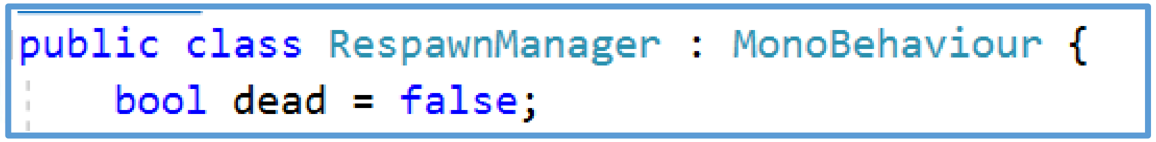
We can have us respawn only when a button is pressed.

We don't necessarily want to always respawn when we hit the button. So, we need a way to identify if the player is dead.

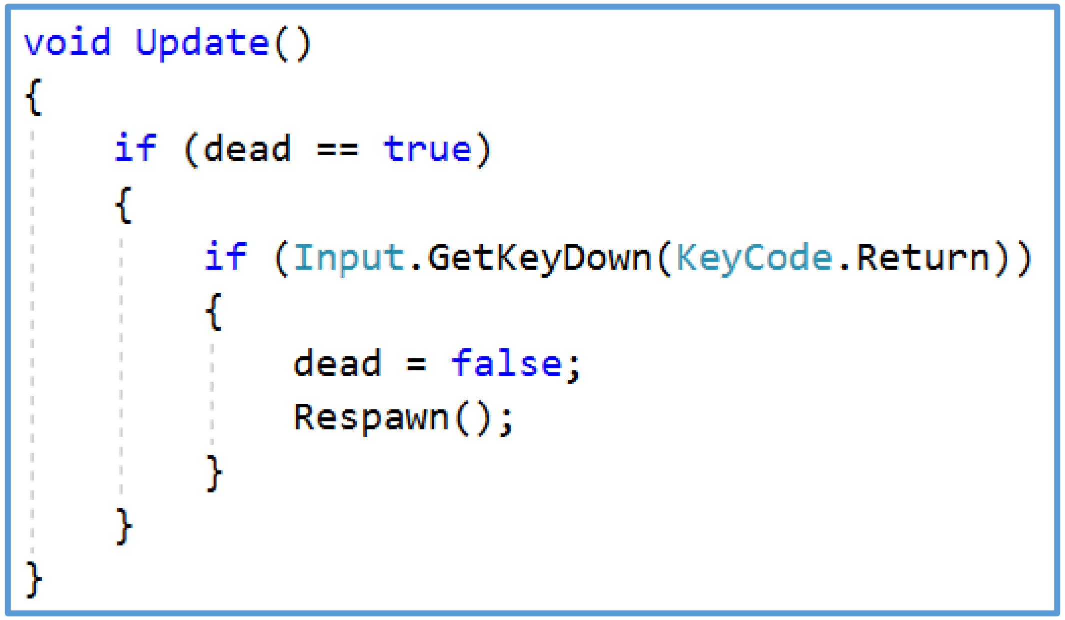
That is where the bool dead comes in. This will be set to true in the pawnDied function

In the update function we want it so that if dead is equal to true then we will listen for a button press.

If the button is pressed, then dead is equal false and then we respawn our character.



```
public class RespawnManager : MonoBehaviour {
    bool dead = false;
```



```
void Update()
{
    if (dead == true)
    {
        if (Input.GetKeyDown(KeyCode.Return))
        {
            dead = false;
            Respawn();
        }
    }
}
```

Respawning

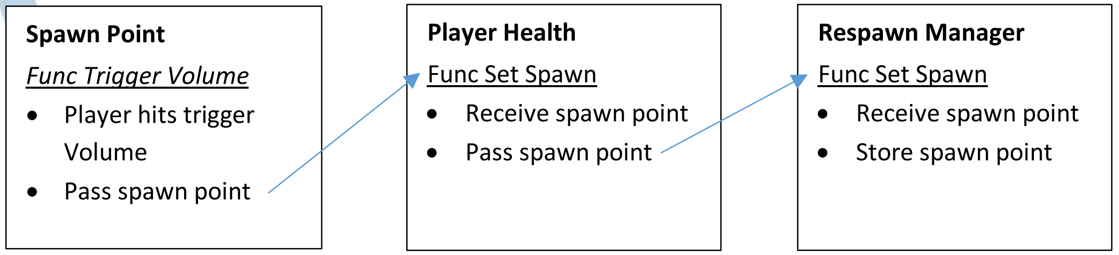
Rise from the digital grave

Respawn points

Now that we can respawn, it would be good to have the ability to switch where we spawn. Some games have checkpoints along the way.

When you hit them, they will be set as the players new spawn point.

However, because our spawn point is stored in the manager, but the player interacts with the respawn point, we need the player script to act as a middle point



Spawn Point

Func Trigger Volume

- Player hits trigger Volume
- Pass spawn point

Player Health

Func Set Spawn

- Receive spawn point
- Pass spawn point

Respawn Manager

Func Set Spawn

- Receive spawn point
- Store spawn point

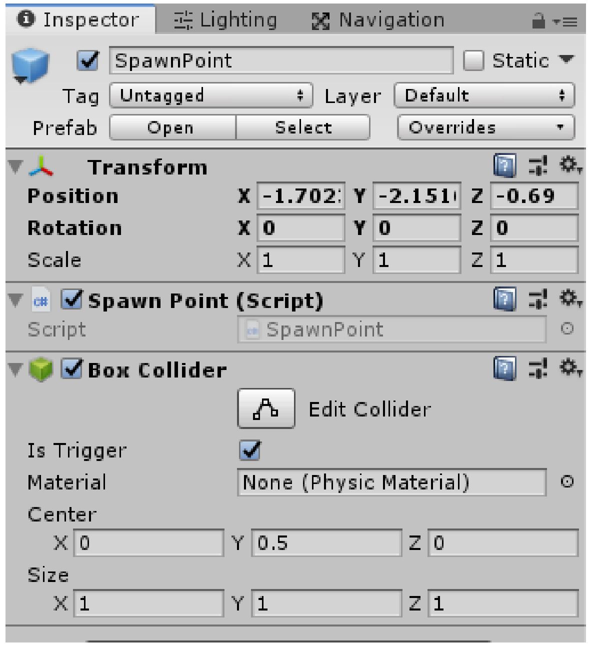
Adding a spawn point script

For our dynamic spawn points, we need a script and a trigger collider.

- On the Spawn point game object
  1. Create a new script called SpawnPoint
  2. Add a BoxCollider
     a. Set its size and position
     b. Set Is Trigger to true
  3. To make it visible
     a. Attach a light (we can adjust this in code)

You can also setup game objects to make it look like a spawn point for the game.

4. Add multiple points into the game



Inspector | Lighting | Navigation

SpawnPoint ☐ Static

Tag: Untagged | Layer: Default

Prefab: Open | Select | Overrides

Transform

Position	X	-1.702	Y	-2.151	Z	-0.69
Rotation	X	0	Y	0	Z	0
Scale	X	1	Y	1	Z	1

✓ Spawn Point (Script)

Script: SpawnPoint

✓ Box Collider

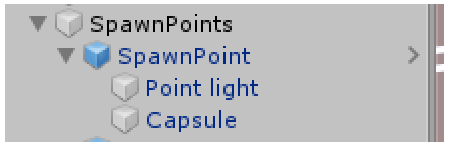
Edit Collider

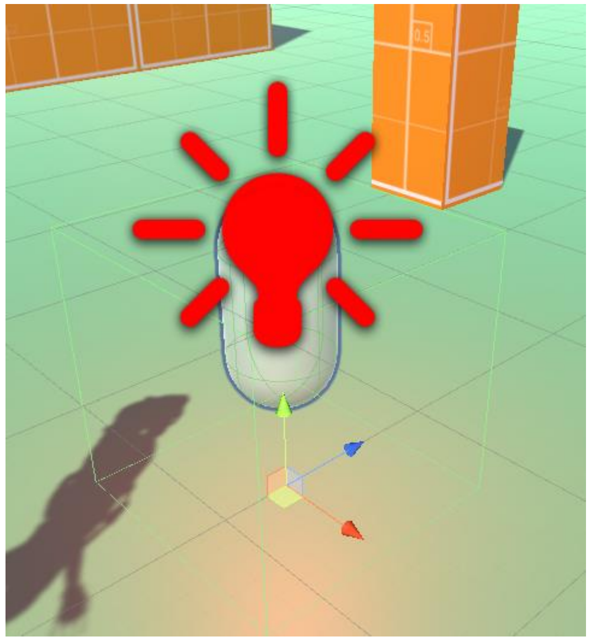
Is Trigger: ✓

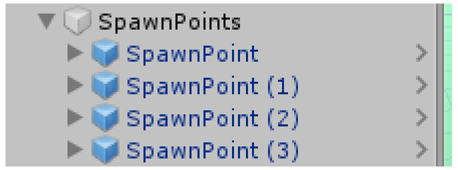
Material: None (Physic Material)

Center: X 0, Y 0.5, Z 0

Size: X 1, Y 1, Z 1





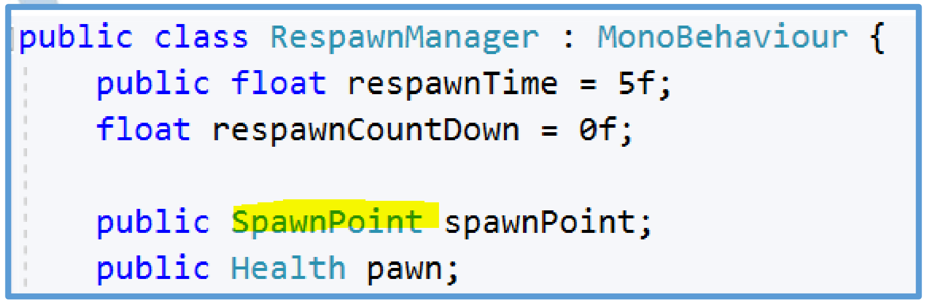


www.aie.edu.au

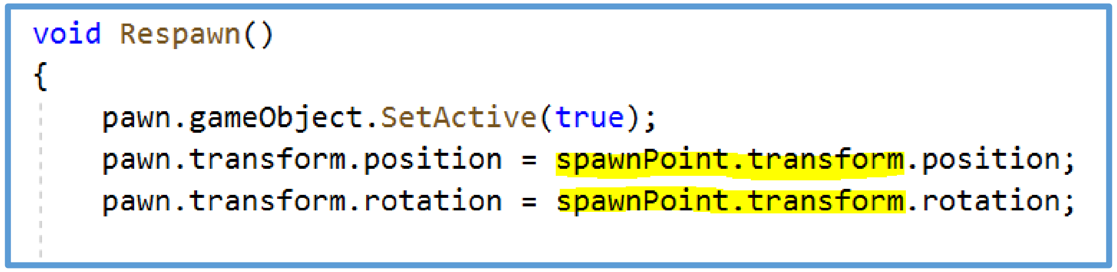
Fixing the Respawn Manager

Before we code our spawn point code. Let's fix it so that the Respawn Manager stores the Script instead of just the Transform

- In the RespawnManager
  5. Change spawnPoints variable type from transform to SpawnPoint
- In the respawn function
  6. Change spawnPoint to spawnPoint.transform



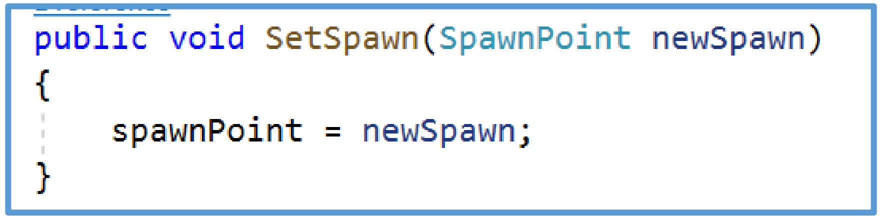
```
public class RespawnManager : MonoBehaviour {
    public float respawnTime = 5f;
    float respawnCountDown = 0f;

    public SpawnPoint spawnPoint;
    public Health pawn;
```



```
void Respawn()
{
    pawn.gameObject.SetActive(true);
    pawn.transform.position = spawnPoint.transform.position;
    pawn.transform.rotation = spawnPoint.transform.rotation;
```

While we are here, let's add a function that we can set the spawn point in code

7. Create a new public function called SetSpawn
   a. Set the parameter SpawnPoint called newSpawn
8. Set the spawnPoint variable to equal newSpawn

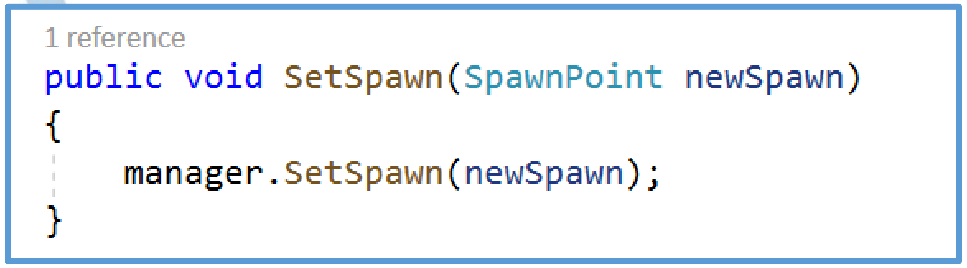


```
public void SetSpawn(SpawnPoint newSpawn)
{
    spawnPoint = newSpawn;
}
```

Next on our Players Health script, we want to add a function that sets the spawn. Which just passes on the new spawn point to the manager.

- In the Health script

9. Create a new public function called SetSpawn
    a. Set the parameter SpawnPoint called newSpawn
10. call managers SetSpawn function
    b. pass in newSpawn



```
1 reference
public void SetSpawn(SpawnPoint newSpawn)
{
    manager.SetSpawn(newSpawn);
}
```

INSIDE THE HEALTH SCRIPT

Lastly, we go the the origin point, the Spawn Point script to add our trigger check

- In the SpawnPoint script

11. Create a public void OnTriggerEnter function
    a. Pass in the parameter Collider called other
12. Check if other has the tag “Player”

- If true

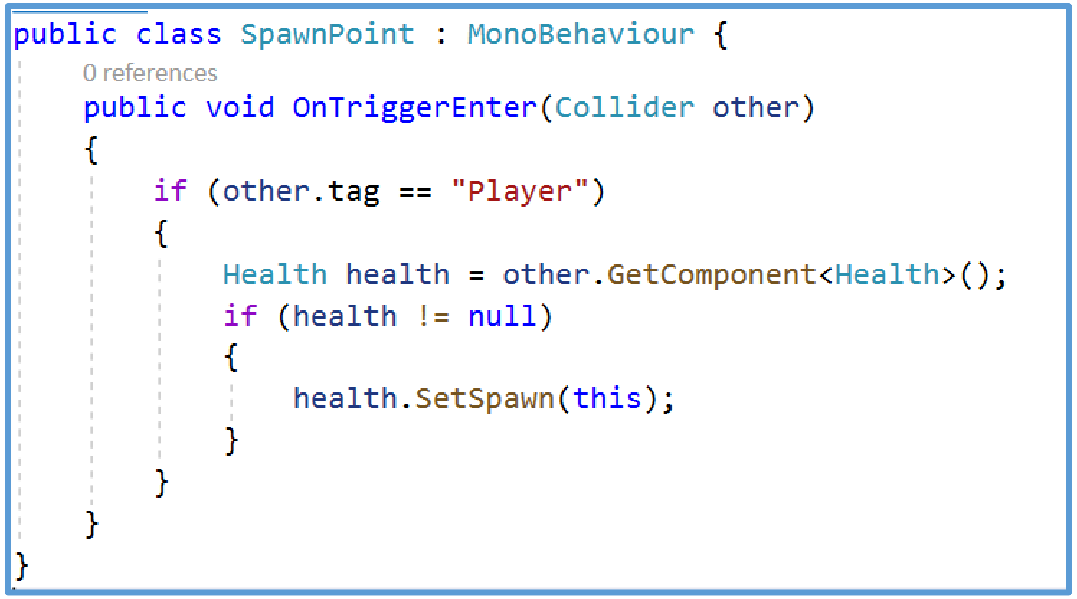
13. Grab the health script from the other object

- If it has the health script

14. Call health’s SetSpawn function.
    a. Pass in this script

This should now set and change the spawn point.

Display wise however nothing really changes.



```
public class SpawnPoint : MonoBehaviour {
    0 references
    public void OnTriggerEnter(Collider other)
    {
        if (other.tag == "Player")
        {
            Health health = other.GetComponent<Health>();
            if (health != null)
            {
                health.SetSpawn(this);
            }
        }
    }
}
```

INSIDE THE SPAWN POINT SCRIPT. WE DO A ON TRIGGER CHECK TO SEE IF THE PLAYER HAS WALKED THROUGH THE SPAWN POINT. IF TRUE. ACTIVATE IT.

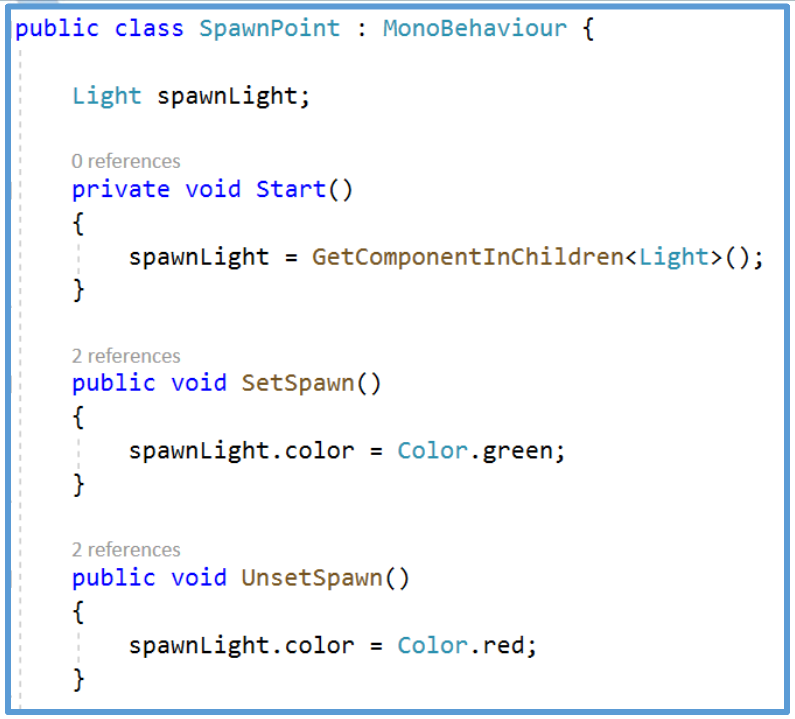
Respawning

Rise from the digital grave

CHANGING THE LIGHT

For the spawn point, I've set up a little set of code that changes the light when the spawn manager calls to it.

- It will set the light to red when the spawn manager changes to another spawn type.
- The game manager will also call SetSpawn to change the light to green. This is to make sure that the Spawn manager has set it as the new spawn point.



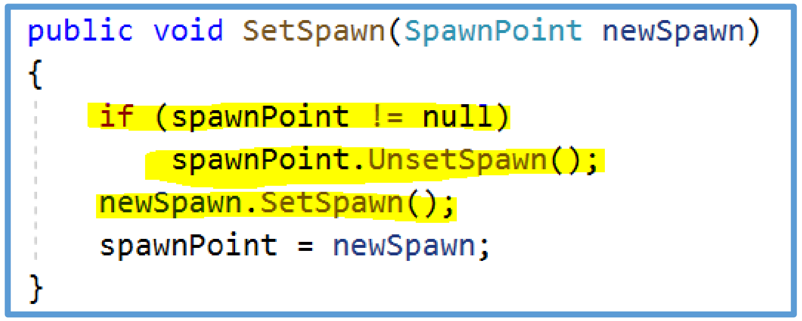
```
public class SpawnPoint : MonoBehaviour {

    Light spawnLight;

    0 references
    private void Start()
    {
        spawnLight = GetComponentInChildren<Light>();
    }

    2 references
    public void SetSpawn()
    {
        spawnLight.color = Color.green;
    }

    2 references
    public void UnsetSpawn()
    {
        spawnLight.color = Color.red;
    }
```

SPAWNPOINT CLASS

For our respawn manager class

- We check to see if we have a previous spawn point before we try to unset it
- We call newSpawns SetSpawn to set light to green

Now when we walk over a spawn point, it will change colour.



```
public void SetSpawn(SpawnPoint newSpawn)
{
    if (spawnPoint != null)
        spawnPoint.UnsetSpawn();
    newSpawn.SetSpawn();
    spawnPoint = newSpawn;
}
```

RESPAWN MANAGER CLASS