Health Management

Death and Destruction

CONTENTS

HEALTH AND DAMAGE

In games, we often have objects that have health and a way to damage them. This can be the player itself, enemies, or even objects in the world.

To do this we need a way to store the health of the object and a way to alter the health.

Then when the life is at 0, then do something to the object, like destroy it.

CREATING HEALTH

1. Create a script called Health

Attach this to any Pawn that you wish to have destroyed. Objects, NPCs, Players. Each should be able to use this

- In the script
  - In the Class block
  2. Create a public int called startingHealth
     a. Set to 100
  3. Create a private int called currentHealth

Current Health will be the actual health of the object. Starting health is the health we will allocate to it when we play a level.

4. Create a private bool called isDead

This will check to see if the object is dead and do conditions based off of that

- In the Start function
  5. Set currentHealth to equal startingHealth
  6. Set isDead to false

This will set and initialise our values.

Now we need a way to alter them and check to see if we are dead.

```
public class Health : MonoBehaviour {
    public int startingHealth = 100;
    private int currentHealth;
    private bool isDead;
```

```
void Start () {
    currentHealth = startingHealth;
    isDead = false;
}
```

OnDeath

Before we can call OnDeath when our object dies, we need the function.

OnDeath will manage what happens when the object Dies. To start with, we should get it so we set the bool to OnDeath to true, then we disable ourselves.

1. Create a void function called OnDeath
2. Set isDead to true
3. Call gameObject.SetActive
    a. Pass in false

So now when the Pawn dies, it will be hidden from the game world, but it will not be destroyed.

```
void OnDeath()
{
    isDead = true;
    gameObject.SetActive(false);
```

Additional Option

Some pawns you do not want to spawn back, just want to destroy when defeated.

- Set it up so there is the option to destroy an object if needed.

Taking Damage

The function we are going to create is a function that we want to call from other gameObjects, like our bullets.

This will be called with a damage amount that we will alter our health with. We then do a check to see if we are dead and then do something when we die.

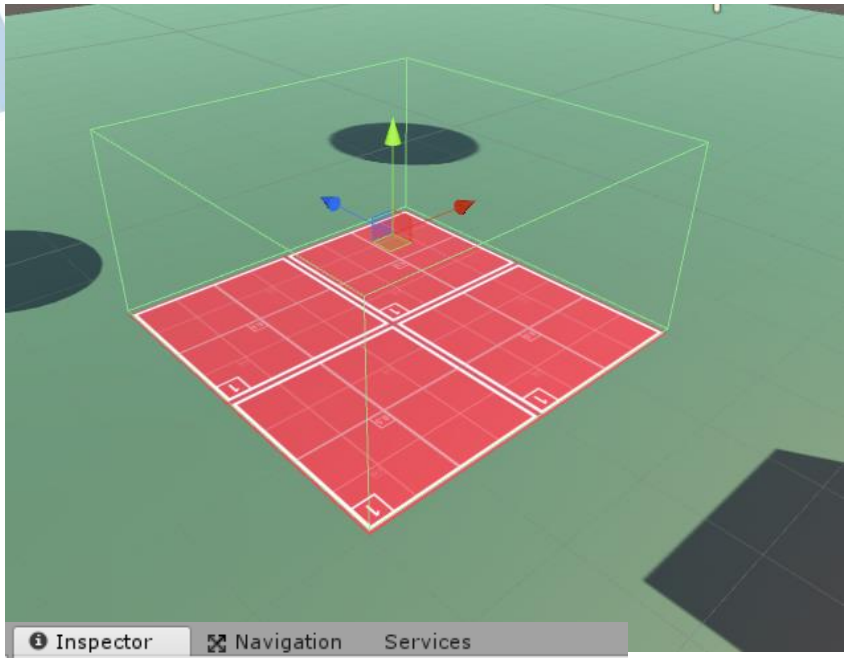
1. Create a public function called TakeDamage()
    a. Pass in a float called amount
- In the function
2. Have a check to see if we are already dead
    a. If we are, return out of the function
3. Set currentHealth to equal itself minus the damage amount
4. Do a check to see if currentHealth is less than or equal to 0, AND is not dead yet
    a. call the function OnDeath()

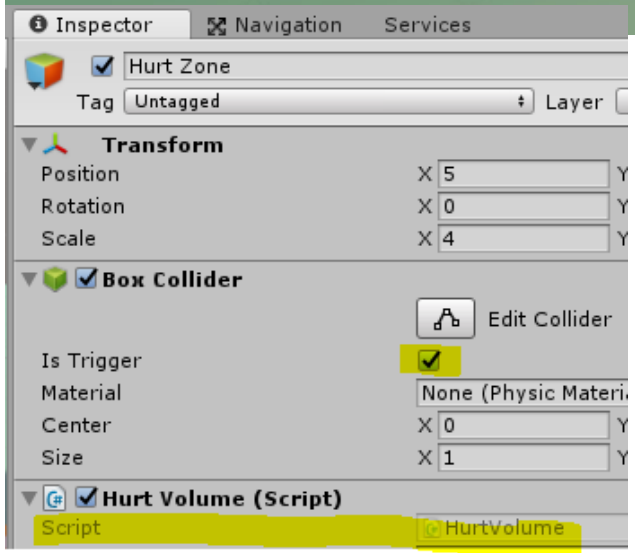
```
public void Damage(int amount)
{
    if (isDead)
        return;

    currentHealth -= amount;
    //Add hurt animation here
    if (currentHealth <=0)
    {
        OnDeath();
    }
}
```

TESTING THE DAMAGE (HURT ZONE)

For testing the damage, we can create a hurt volume. This is using ideas from the trigger volumes

1. Create an empty game object
    a. Call it HurtZone
    b. Create a TriggerVolume
    c. Set it to the size you would like
    d. Make it visible in the game that you will be hurt here
        i. Adding an art asset and/or
2. Create a script called HurtVolume
    a. Attach it to the HurtZone object





- In the script
  - In the class block
    3. Create a public integer called damage
       a. Set it to 10

This trigger will hurt someone, when they are inside the volume. For continuous damage, we will use OnTriggerStay

4. Create an OnTriggerStay function
   a. Pass in the Collider parameter called other

This will grab our health script component off the object we wish to hurt. However not everything that enters is damageable, so we want to check if it has health.

- In the function
  5. Create a Health variable called health
     a. Pass in others GetComponent for the Health class
  6. Do a check to see if the health class is not null
- If it is true
  7. Call Health's Damage function
     a. Pass in damage

This will damage the player. Every update. Killing the player very fast. We want to manage it so we don't get a bunch of damage done so fast.

```
public class HurtVolume : MonoBehaviour {
    public int damage = 10;
    0 references
    public void OnTriggerStay(Collider other)
    {
        Health health = other.GetComponent<Health>();
        if (health != null)
        {
            health.Damage(damage);
        }
    }

}
```

Temporary Invulnerability

Often in games, units gain a temporary invulnerability, this is so that if swarmed, they don't die instantaneously but still have a fighting chance.

We are going to set it up so that the health script deals with temporary invulnerability.

- In the Health Class
  - In the Class block
    1. Create a public bool called invulnerableTime
       a. Set it to 0.5f seconds
    2. Create a float called timer
       a. Set it to 0
    3. Create a bool called canHurt
       a. Set it to true

Next we create a function that counts down our involnurability

4. Create a void function called InvulnerableTimer

- In the function
  5. Check if canHurt is false
- If it is
  6. Get the timer to count up
  7. Check if timer is larger than invulnerableTime
- If it is
  8. Reset timer
  9. Set canHurt to true
  10. Call the function in the Update function

This should now make you temporarily invulnerable to damage

```
public class Health : MonoBehaviour {
    //Hit temp involnurability
    public float invulnerableTime = 0.5f;
    float timer = 0;
    bool canHurt = true;
```

```
void InvulnerableTimer()
{
    if (canHurt == false)
    {
        timer += Time.deltaTime;
        if (timer >= invulnerableTime)
        {
            timer = 0;
            canHurt = true;
        }
    }
}
```

```
void Update () {
    InvulnerableTimer();
}
```

AUTO KILL

Sometimes you want the pawn to automatically die when something happens.

If it leaves the world somehow

If it gets hit by a super powerful object

If we just want an object to go

We can create a simple function called AutoKill and set the Pawn to die automatically

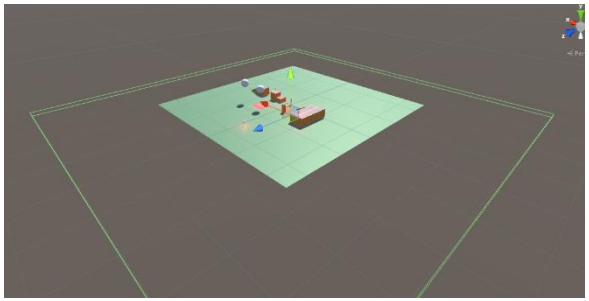
1. Create a new void function called AutoKill
2. Check if isDead is not false

- If it is

3. Call the OnDeath function

Using this idea, we can create a KillVolume that auto kills pawns that enter it. Similarly, to our Hurt volume.

I used it to create a large volume surrounding the world if the player falls off.



A KILL VOLUME AROUND THE ZONE IF SOMETHING FALLS OFF WITH HEALTH, IT WILL KILL IT

```
public void AutoKill()
{
    if(!isDead)
        OnDeath();
}
```

```
public class KillVolumeSpawnTest : MonoBehaviour {

    0 references
    public void OnTriggerEnter(Collider other)
    {

        Health health = other.GetComponent<Health>();
        if (health != null)
        {
            health.AutoKill();
        }
    }
```