Documentation proof of concept- Nathan & Saud

Pick up script and count constraints

Pick ups and Counting script

For this 2D game we have some pick up objects used to create friendly units.

The first concerns with scripting this is being able to have objects in our game that the player can pick up: in other words count when collided with and then make the object disappear.

Lets take a look at our script... in C#

```
public class PlayerController : MonoBehaviour {
    public Text countText;
    public Text winText;
    private Rigidbody rb;
    private int count;
    void Start() {
        rb = GetComponent<Rigidbody>();
        count = 0;
        SetCountText();
        winText.text = ""; }
    void OnTriggerEnter(Collider other) {
      if (other.gameObject.CompareTag("Pick Up")) {
          other.gameObject.SetActive(false);
          count = count + 1;
          SetCountText(); }
    }
```

```
void SetCountText ()
  {
      countText.text = "Count: " + count.ToString();
      if (count >=14)
      {
          winText.text = "You Win!";
      }
  }
```

…….. You get all that?

no?

Well, let’s break it down.

First We need the script what information we need to get started

So we make a few of our own variables, name them, and also grab a rigid body:

```
public class PlayerController : MonoBehaviour {
    public Text countText;
    private Rigidbody rb;
    private int count;
```

Now let's tell it what to do with that information

```
void Start() {
    rb = GetComponent<Rigidbody>();
    count = 0;
    SetCountText();}
```

So let me translate:

When the scene/level starts Grab the rigidbody componet of this object.

Set the count to zero and fire the custom function count text.

Time to Play: Check if we are hitting something

```
void OnTriggerEnter(Collider other) {
    if (other.gameObject.CompareTag("Pick Up")) {
        other.gameObject.SetActive(false);
        count = count + 1;
        SetCountText(); }
 }
```

English: When this object collides with another object fire this function. Check the other objects Tag and if that Tag is pick up. Turn it off (hide it).
After that Add 1 to the count and fire our custom function.

Custom Functions

so this entire time we have been telling the computer to fire a custom function. Basically a function that we are about to make.

So let's tell it what to do like a good little doggy.

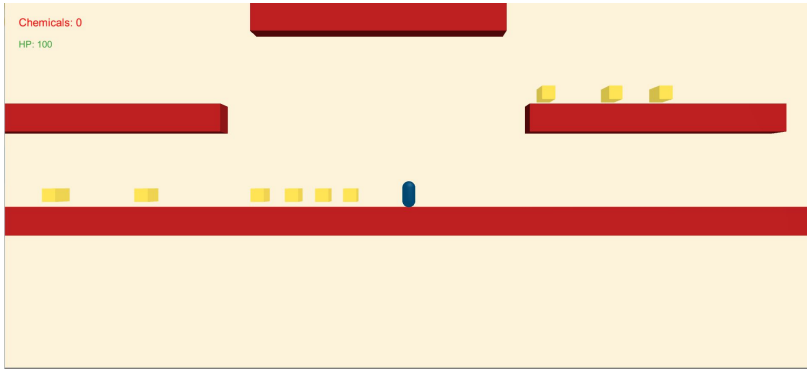
```
void SetCountText ()
  {
    countText.text = "Count: " + count.ToString();}
```

So this is what we want it to do.

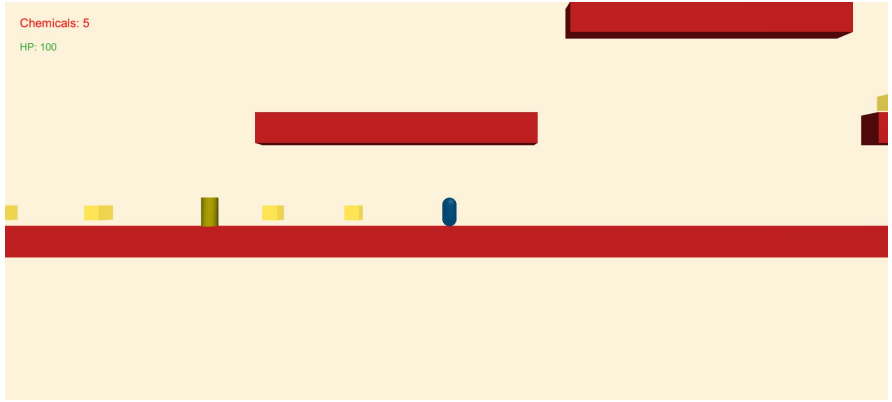
If the Function called: SetCountText is called fire this function: Take the UI Text called Count and write this "Count :" then and the count string that you have been keeping to the end of this.

You Following me?
Good, Now lets see it.

Before picking up object



After



So now we need to move on to the count constraint

We are now trying to tell the game to only fire the when they have ammo. Or for this game spawn units when they have chemicals.

So we went in to our spawn code and added a few things. That is written in Javascript.

But if you remember our count script was in C#. No, worries we can still talk between them.

Count check and disable

Only pay attention to the Red.

```
var count : UI.Text;
public var curCount : int;
private var player : GameObject;
public var CapsulePrefab : Rigidbody;
public var Spawnpoint : Transform;
function Awake () {
       player = GameObject.FindGameObjectWithTag ("Player");
       var cSharpScript = player.GetComponent("ChemicalCount");
    curCount = cSharpScript.count;
    SetCountText(); }
function Update () {
             var cSharpScript = player.GetComponent("ChemicalCount");
    curCount = cSharpScript.count;
   if(curCount <= 0) {
   return; }
```

```
if(Input.GetButtonDown("Fire1")) {
    var CapsuleInstance : Rigidbody;
      CapsuleInstance = Instantiate(CapsulePrefab, Spawnpoint.position,
Spawnpoint.rotation);
      curCount = curCount - 1;
      cSharpScript.count = curCount;
      SetCountText(); }
  }
    function SetCountText () {
                    count.text = "Chemicals: " + curCount.ToString();
                    }
```

Okay, Breakdown time

```
var count : UI.Text;
public var curCount : int;
private var player : GameObject;

function Awake () {
      player = GameObject.FindGameObjectWithTag ("Player");
      var cSharpScript = player.GetComponent("ChemicalCount");
   curCount = cSharpScript.count;
   SetCountText(); }
```

Create these public variables and private variable : our information grab.

When the Game starts:

- use the gameobject tagged player for the private variable named player.
- Grab the script/component from the player variable named “Chemical Count”. and just so you know computer it is a C# script.
- Set curCount equal to the C# scripts variable named count.
- Now fire our custom script.

Part two: what we want to check every frame

```
function Update () {
            var cSharpScript= player.GetComponent("ChemicalCount");
    curCount = cSharpScript.count;
   if(curCount <= 0) {
   return; }

if(Input.GetButtonDown("Fire1")) {
     var CapsuleInstance : Rigidbody;
       CapsuleInstance = Instantiate(CapsulePrefab, Spawnpoint.position,
Spawnpoint.rotation);
      curCount = curCount - 1;
      cSharpScript.count = curCount;
      SetCountText(); }
   }
```

This is our Custom function:

```
function SetCountText () {
                    count.text = "Chemicals: " + curCount.ToString();
                    }
```

English: Every frame check this.

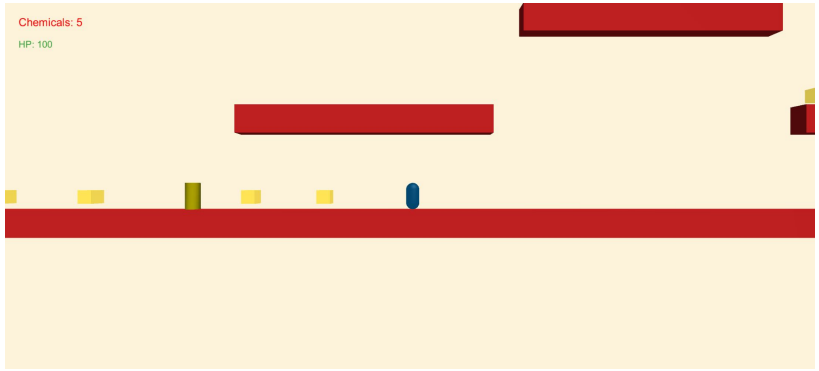
- Check the C# Script named Chemical count from The Game object named Player and see if anything has changed.
- Set curCount to the same as the C# Variable Count.
- Now if the curCount is less than or equal to 0. Don't go any further in this function.
- If the Player input button Fire 1 is pressed. Fire. we're not going into detail for this one.
- Now if they fire subtract 1 from the curCount. Then set the C# Variable Count to the same as curCount.
- Lastly fire our custom function.

Cool, Cool, Cool!

- So, we have now told the computer to check if we have ammo and if we don’t do not let us fire.
- But, if we do have ammo and we fire subtract one from our clip/inventory and display the current amount to us.

Enough talk it's picture time.

Before Firing



After