# Specialism production diary
**Connor Howard**

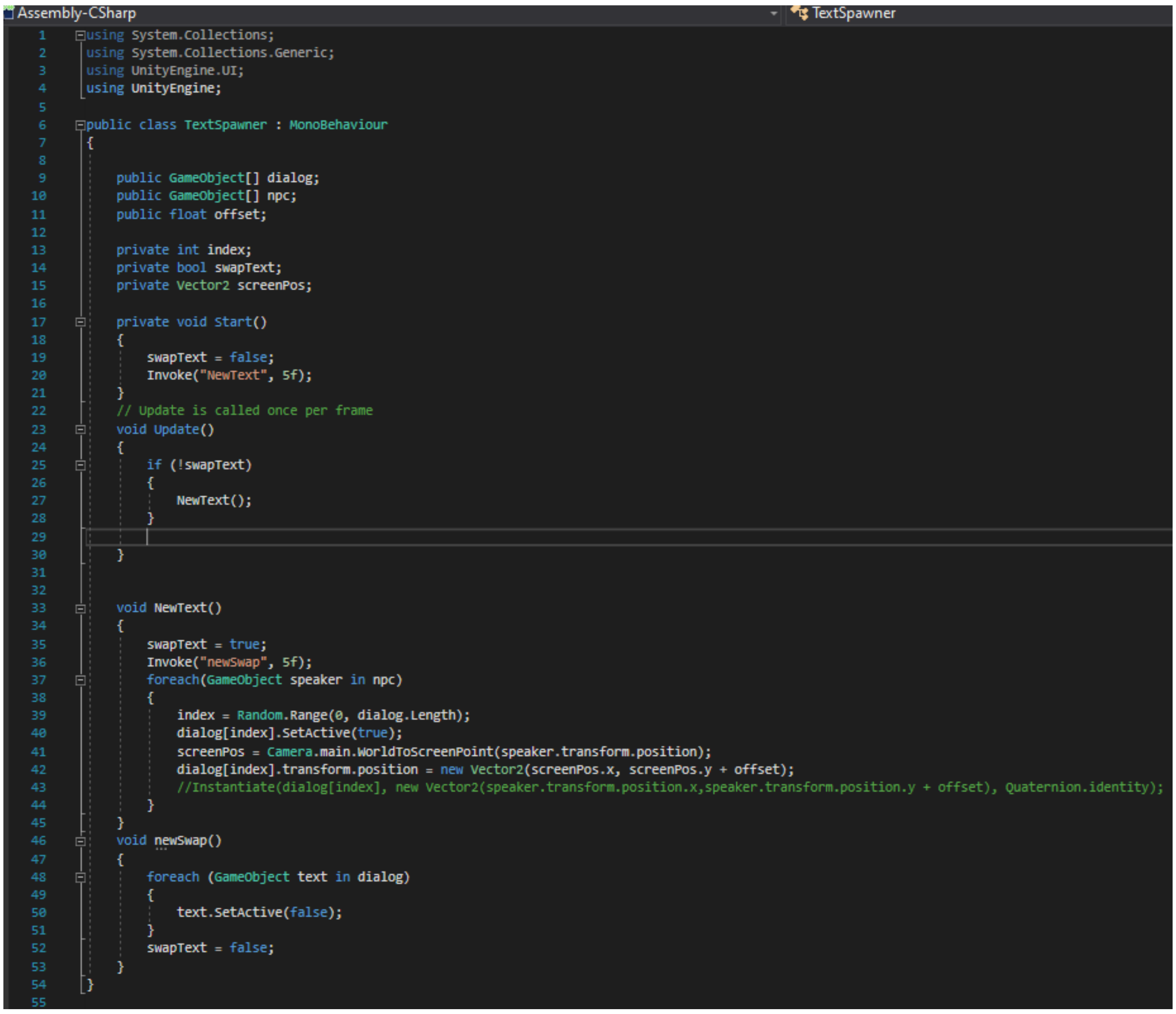
For this semester's specialism module I chose to go again with programming. I chose this because it is the field I am most interested in pursuing and will help me greatly when it comes to completing my BSc.

## Auto Scaling Text Box

The briefings themselves were the same as the previous year and I decided to first try out the auto scaling brief, in this brief I would have to create an auto scaling UI image that would adjust its size to fit text that appears on screen.

**16/02/21** -

I started off by placing 5 boxes around a blank scene with the idea of having a randomly chosen selection text strings appear above them and they would alternate every 5 seconds, I would then plan on working to scale the UI split image with each string. I created a script that would store the boxes as an array as well as an array of 10 text objects. I started by having a boolean named “swapText” that would start false. In update if this boolean is set to false it would run the “NewText” method, this method would run a foreach loop to randomly select a text object from the “dialog[]” array per box and set it to active above that box. It would then set the boolean to true so this method is not continuously run. Finally I would Invoke the “newSwap” method after 5 seconds which would use another foreach loop to go through each box and disable the text object above it ready for a new one, then the boolean would be set back to false and the whole cycle repeats.



```csharp
using System.Collections;
using System.Collections.Generic;
using UnityEngine.UI;
using UnityEngine;

public class TextSpawner : MonoBehaviour
{

    public GameObject[] dialog;
    public GameObject[] npc;
    public float offset;

    private int index;
    private bool swapText;
    private Vector2 screenPos;

    private void Start()
    {
        swapText = false;
        Invoke("NewText", 5f);
    }
    // Update is called once per frame
    void Update()
    {
        if (!swapText)
        {
            NewText();
        }

    }


    void NewText()
    {
        swapText = true;
        Invoke("newSwap", 5f);
        foreach(GameObject speaker in npc)
        {
            index = Random.Range(0, dialog.Length);
            dialog[index].SetActive(true);
            screenPos = Camera.main.WorldToScreenPoint(speaker.transform.position);
            dialog[index].transform.position = new Vector2(screenPos.x, screenPos.y + offset);
            //Instantiate(dialog[index], new Vector2(speaker.transform.position.x,speaker.transform.position.y + offset), Quaternion.identity);
        }
    }
    void newSwap()
    {
        foreach (GameObject text in dialog)
        {
            text.SetActive(false);
        }
        swapText = false;
    }
}
```

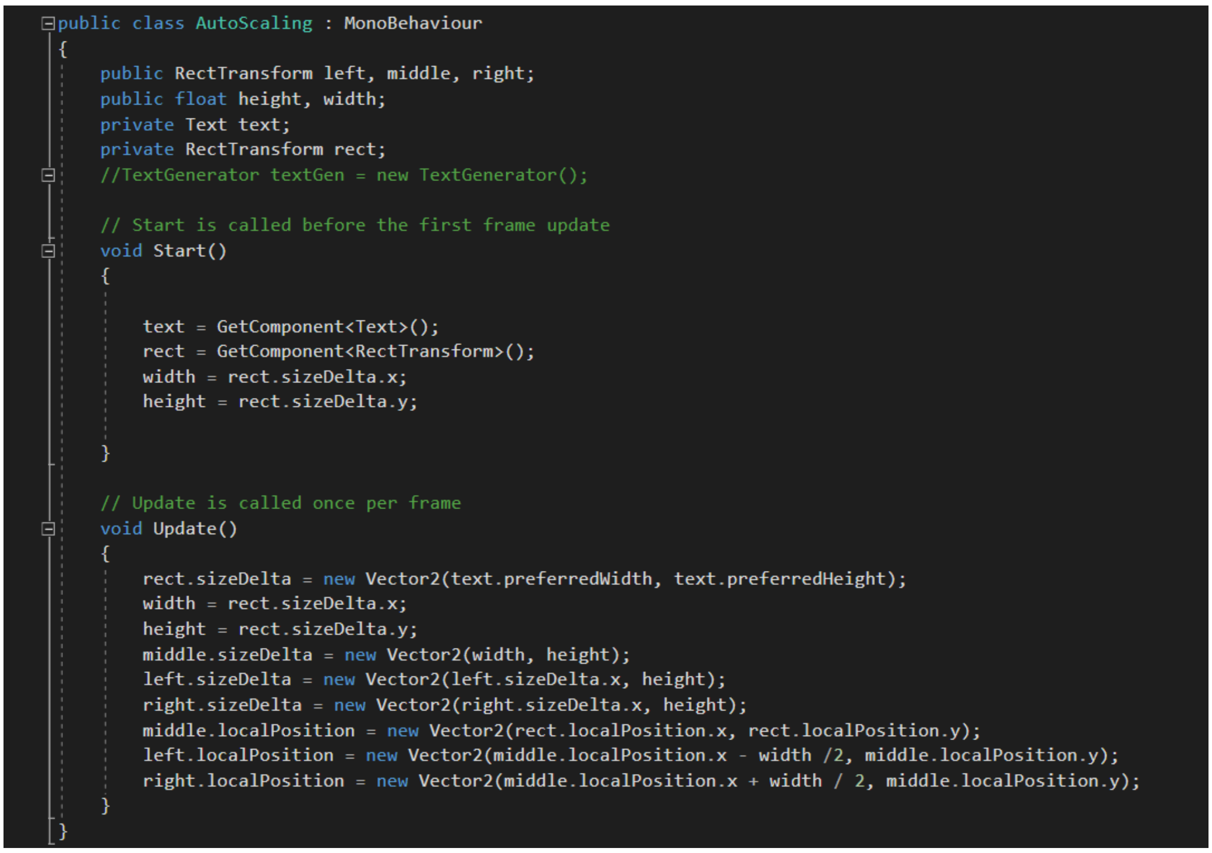
As shown in the image of my script above I had originally intended to instantiate the text objects above the box but I decided to change this as given there was only 10 text objects it made more sense in terms of performance to have them already loaded into the scene and simply disable and enable them, continuously instantiating new text objects would not be the efficient way to go about this.

**17/02/21** -

After doing all of the above yesterday I realised that my whole method for setting this briefing up may have to change, while it does work having boxes randomly show selected text objects above them then work on having the UI image scaled to that text, I realised that while I know I could program the UI image to scale it may look like I simply pre-placed and pre-scaled the image if im just showing and hiding text objects that do not dynamically change during play. I scrapped the idea and instead started anew with the simple idea of one text UI in the centre of the screen with a UI image that would scale dynamically as the textbox changed during play.

I cleared my scene of the boxes and set up a simple text UI in the centre of the screen, I then created the UI image using photopea. Once that was done I noticed in the briefing it said “sliced background image” so I took the UI image I made and used the sprite editor to slice it into 3 sections, a middle section that will be the piece that gets scaled along the x-axis and a left and right piece that will fit on either ends but not be scaled in the x-axis so the image as a whole does not look stretched.

I then created a script that would handle the auto scaling of the UI image, I made references to the images rect transform components which would be how I would scale them. I manipulated the rect transform of all gameobject using the sizeDelta parameter. Inside update I resized the rect transform of the text object to match the text’s preferred width and height, I then stored these values within two public floats and used them as the variables to adjust the sizeDelta of the UI images rect transforms. Once the size was adjusted to fit the text object I then adjusted the local position of the rect transform with the two ends to be adjusted to the middle UI images position - or + half of the width value to sit on either end.



```
public class AutoScaling : MonoBehaviour
{
    public RectTransform left, middle, right;
    public float height, width;
    private Text text;
    private RectTransform rect;
    //TextGenerator textGen = new TextGenerator();

    // Start is called before the first frame update
    void Start()
    {

        text = GetComponent<Text>();
        rect = GetComponent<RectTransform>();
        width = rect.sizeDelta.x;
        height = rect.sizeDelta.y;

    }

    // Update is called once per frame
    void Update()
    {
        rect.sizeDelta = new Vector2(text.preferredWidth, text.preferredHeight);
        width = rect.sizeDelta.x;
        height = rect.sizeDelta.y;
        middle.sizeDelta = new Vector2(width, height);
        left.sizeDelta = new Vector2(left.sizeDelta.x, height);
        right.sizeDelta = new Vector2(right.sizeDelta.x, height);
        middle.localPosition = new Vector2(rect.localPosition.x, rect.localPosition.y);
        left.localPosition = new Vector2(middle.localPosition.x - width /2, middle.localPosition.y);
        right.localPosition = new Vector2(middle.localPosition.x + width / 2, middle.localPosition.y);
    }
}
```

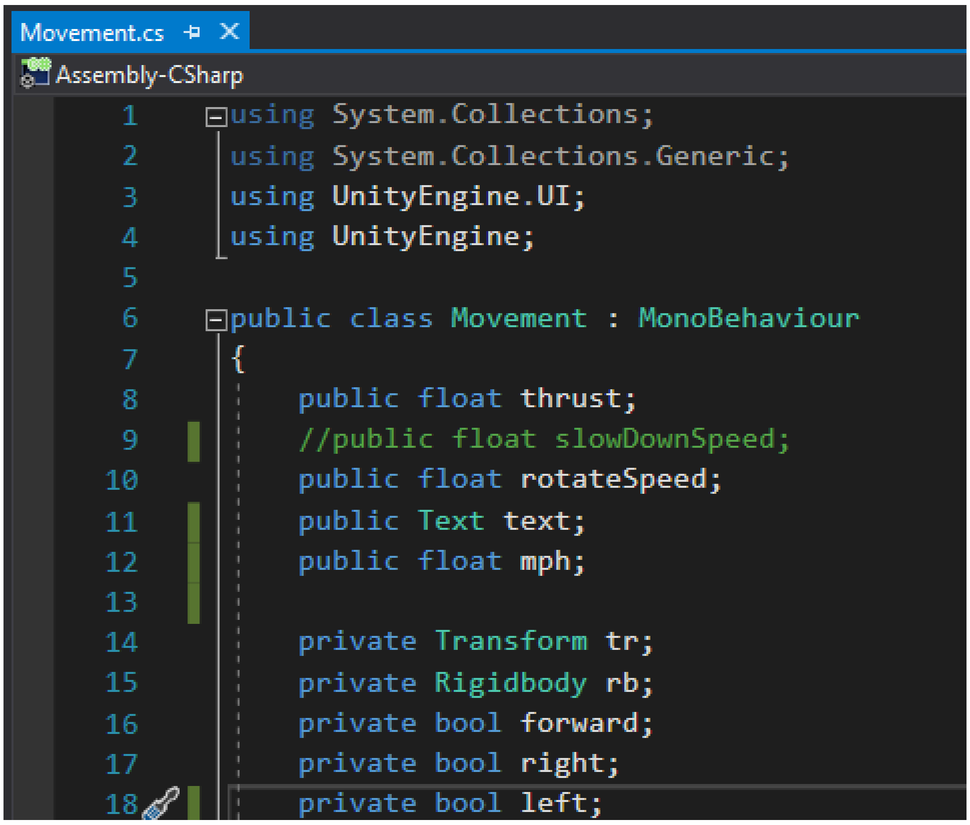
## Speedometer

In this brief I was tasked with measuring the current speed of an object through unity world space. It would have to take the speed from the rigidbody component and convert it to display miles per hour.

**02/03/21** -

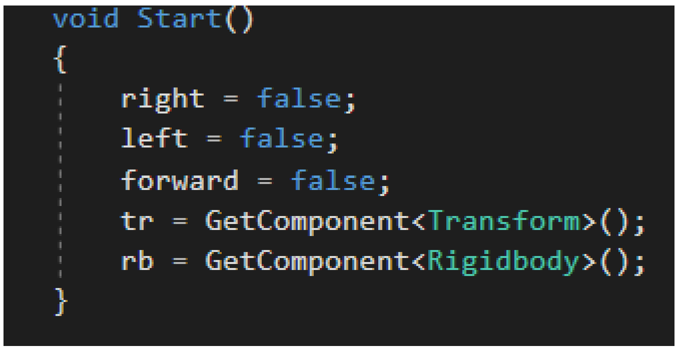
I started off by setting up the unity scene. I created a 3D cube and made sure it had a box collider and rigidbody component attached. I applied constraints to the x and z rotations so that when moving it would not flip or roll over. I then made a stretched plane for the object to have a surface to move about on. Lastly I needed something to display the speed on screen so created a canvas and a UI text object to do so.

Once done I made a movement script and opened it within microsoft visual studios to begin programming. First I made some public and private variables to represent the object's speed when moving forward, a rotation speed for rotating it left and right and references to the components mentioned earlier in the unity scene. I also needed a reference to the text object so after adding in “using UnityEngine.UI;” I then made the reference to the ui text. I needed to use a blank float variable which would act as the outcome value for the conversion to miles per hour to be displayed. I used booleans to detect whether the object was moving forward or not as well as being rotated left or right. I knew that I would be moving the object by applying a force to the rigidbody and decided to create a variable to represent a slow down speed when the force was no longer applied but decided to scrap this for an alternative method of slowing down later in the script.



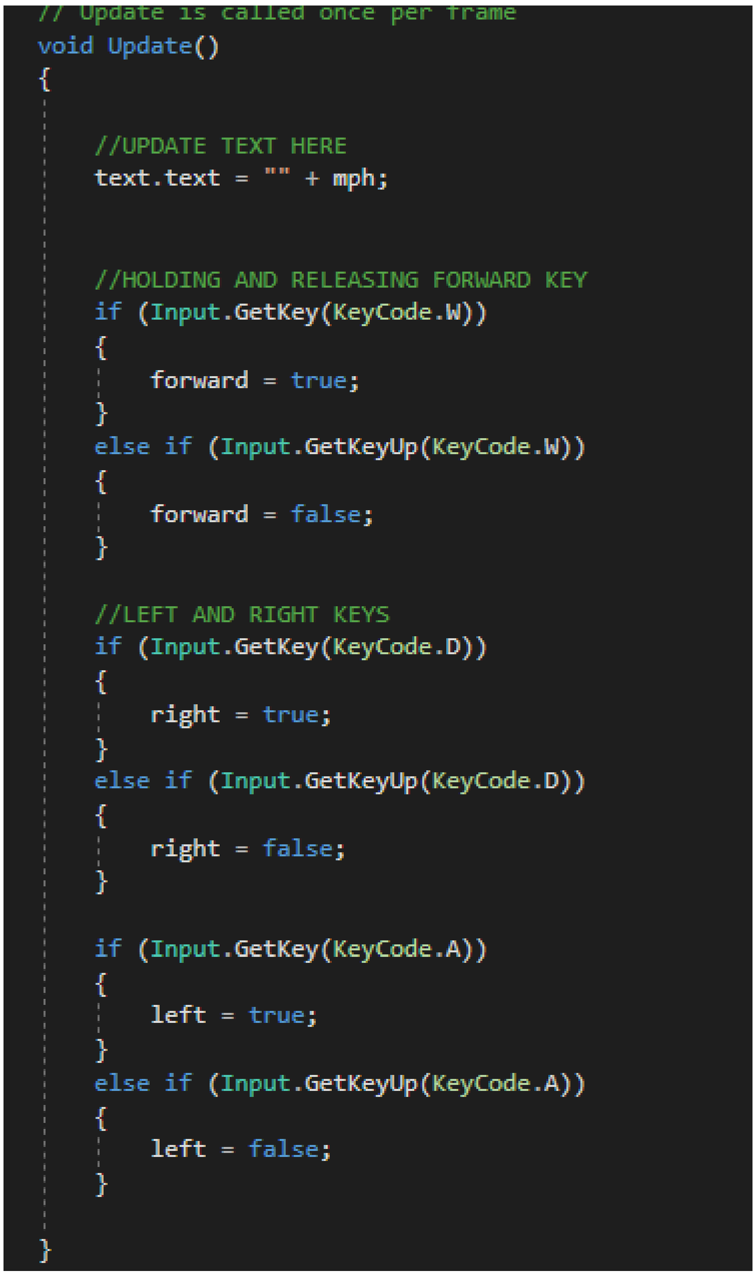
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine.UI;
using UnityEngine;

public class Movement : MonoBehaviour
{
    public float thrust;
    //public float slowDownSpeed;
    public float rotateSpeed;
    public Text text;
    public float mph;

    private Transform tr;
    private Rigidbody rb;
    private bool forward;
    private bool right;
    private bool left;
```

After making all my declarations I then made sure to assign some values to those needed within the start method.



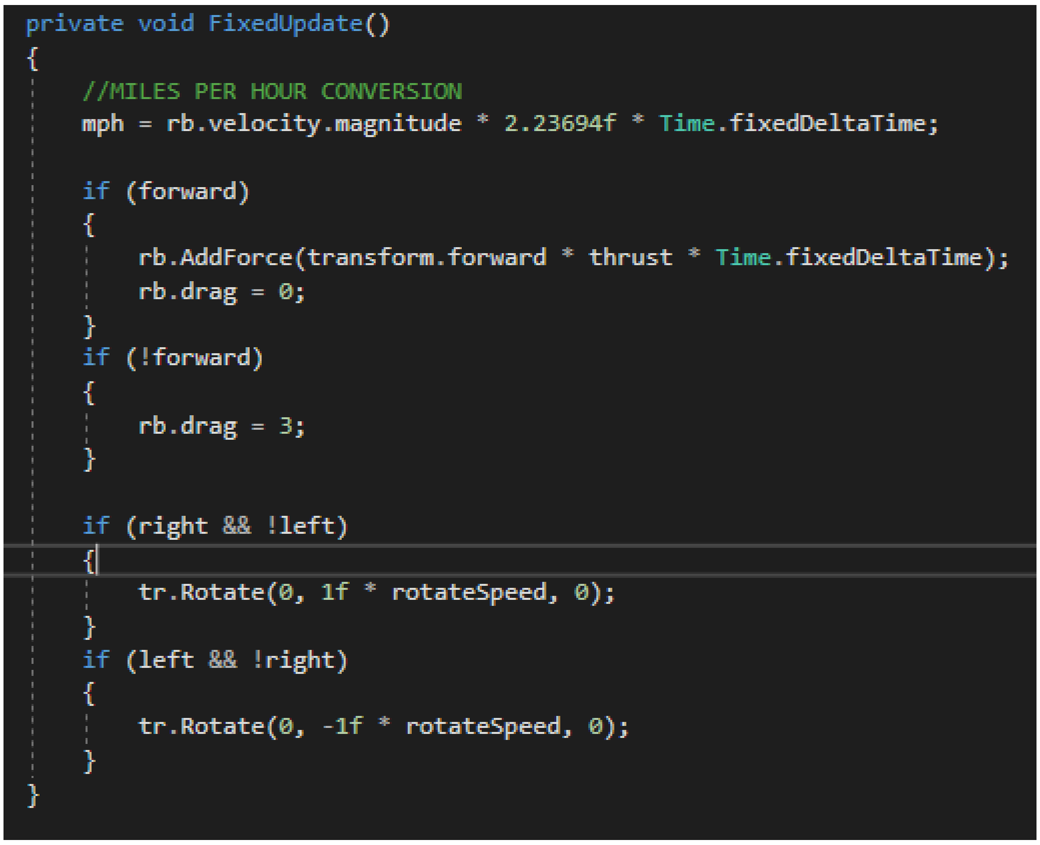
```
void Start()
{
    right = false;
    left = false;
    forward = false;
    tr = GetComponent<Transform>();
    rb = GetComponent<Rigidbody>();
}
```

Inside the update method I used input detection functions to check which key was pressed and released to turn booleans associated with that directional key to true and false. I made sure to continuously display the updated mph variable within the UI text object.



```
// Update is called once per frame
void Update()
{

    //UPDATE TEXT HERE
    text.text = "" + mph;


    //HOLDING AND RELEASING FORWARD KEY
    if (Input.GetKey(KeyCode.W))
    {
        forward = true;
    }
    else if (Input.GetKeyUp(KeyCode.W))
    {
        forward = false;
    }

    //LEFT AND RIGHT KEYS
    if (Input.GetKey(KeyCode.D))
    {
        right = true;
    }
    else if (Input.GetKeyUp(KeyCode.D))
    {
        right = false;
    }

    if (Input.GetKey(KeyCode.A))
    {
        left = true;
    }
    else if (Input.GetKeyUp(KeyCode.A))
    {
        left = false;
    }

}
```

I knew that I would want to do the physics calculations inside the fixed update method as it handles it better than the update method, so inside this method after the booleans are set within the update method I then start using physics to apply force to the rigidbody of the object and also perform rotations. I then set the value of the mph variable to be equal to the speed of the object in meters per second which is “rigidbody.velociity.magnitude” and multiplied it by 2.23694f which is the conversion rate from meters per second to miles per hour, I then multiplied this by “Time.FixedDeltaTime”. Lastly I revised how I wanted the object to slow down to a stop when the force was no longer applied so I decided to affect the rigidbody’s drag component directly to do so.



```
private void FixedUpdate()
{
    //MILES PER HOUR CONVERSION
    mph = rb.velocity.magnitude * 2.23694f * Time.fixedDeltaTime;

    if (forward)
    {
        rb.AddForce(transform.forward * thrust * Time.fixedDeltaTime);
        rb.drag = 0;
    }
    if (!forward)
    {
        rb.drag = 3;
    }

    if (right && !left)
    {
        tr.Rotate(0, 1f * rotateSpeed, 0);
    }
    if (left && !right)
    {
        tr.Rotate(0, -1f * rotateSpeed, 0);
    }
}
```

Once this was all done I saved the script and linked my public references within the inspector and pressed play, the object now displays it’s current speed in miles per hour.

## Shuffle button

For this brief I had to create a shuffle button that would randomly mix a list of tracks each time the function is called, I used strings to represent the song tracks and the amount of tracks must be customisable within the inspector to have larger or smaller lists of tracks.

**16/03/21** -

As always the first thing I did was set up the scene before moving on to the programming of the script. I created a UI button on a canvas and gave it the text display of “Shuffle”, I resized it to fit nicely on the screen and that’s all the setting up I needed to do so moved on to making the script which I called “Shuffle”.

The idea I used was to have two public lists, one containing all the available tracks and the other being empty. I planned on calling the shuffle function which would randomly shuffle through the available tracks list and then place them in the used tracks list, I then displayed them within the console window. So to do this I made two public lists, a boolean to activate the shuffle function, a private string to represent the individual chosen track in the available list to be moved over and a private integer to represent the amount of available tracks to use within a loop.

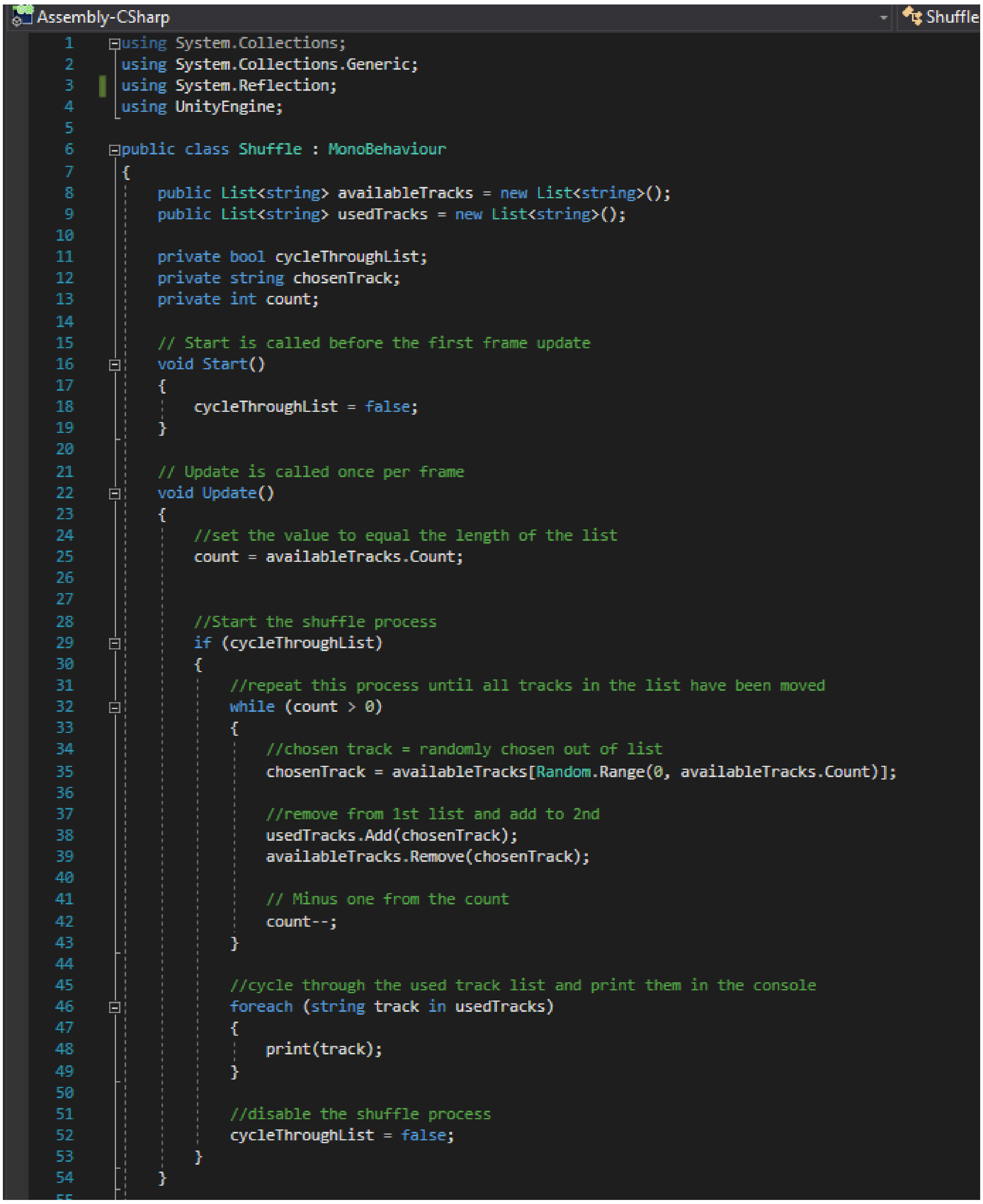
I made sure to start the boolean on false within the start method then moved into the update method. I set the value of the count integer to be equal to “.Count” value of the available track list then created an if statement checking if my boolean was true that would use a while loop to cycle through the list and assign the private string variable to be equal to a “[Random.Range(0, availableTracks.Count)];”. I then removed the variable attached to the private string from the first list and placed it within the second, lastly I decreased the integer value by one to represent a track has been taken from the first list so has less amounts to cycle through within the loop.

I still needed to display the tracks in the console once they have been moved over so still inside the if statement I used a ”foreach” loop to cycle through all the tracks within the second list and print them. I then simply set the boolean value that cycles through the list to false.

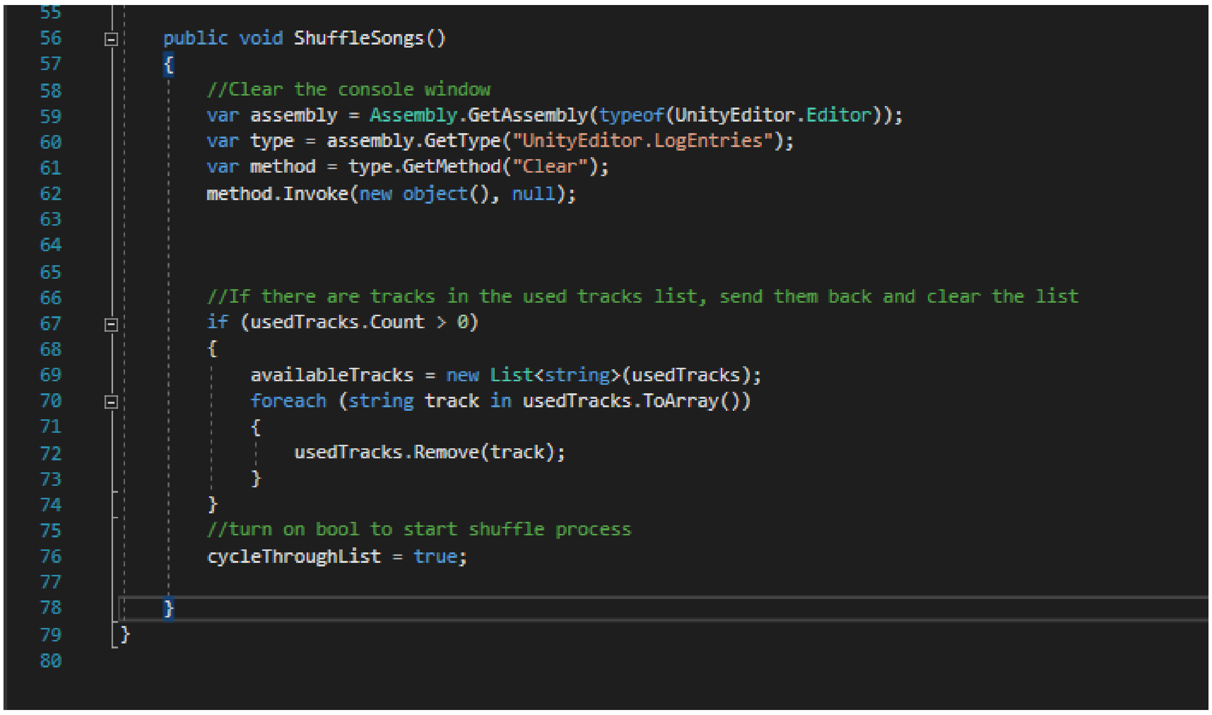
There was still no way to test this yet as the boolean does not get set to true and so I created a function called “ShuffleSongs()”, this would be the function that would be called whenever the UI button was pressed and the first thing I wanted to happen was the console window to be cleared as to not confuse with a display of previous calls to this function being displayed in the console window.

I did some research and found that by adding “using System.Reflection” at the top I could then create a variable that had access to the Assembly giving me further access to the editor log entries, I could then call a function that clears the window.

After this was done the function would then check using an if statement if the second list had any tracks in it and if so would then cycle through that list removing any tracks inside. Lastly the function after running all the necessary checks would then turn the boolean to true allowing the entire function within update to run.



```csharp
using System.Collections;
using System.Collections.Generic;
using System.Reflection;
using UnityEngine;

public class Shuffle : MonoBehaviour
{
    public List<string> availableTracks = new List<string>();
    public List<string> usedTracks = new List<string>();

    private bool cycleThroughList;
    private string chosenTrack;
    private int count;

    // Start is called before the first frame update
    void Start()
    {
        cycleThroughList = false;
    }

    // Update is called once per frame
    void Update()
    {
        //set the value to equal the length of the list
        count = availableTracks.Count;


        //Start the shuffle process
        if (cycleThroughList)
        {
            //repeat this process until all tracks in the list have been moved
            while (count > 0)
            {
                //chosen track = randomly chosen out of list
                chosenTrack = availableTracks[Random.Range(0, availableTracks.Count)];

                //remove from 1st list and add to 2nd
                usedTracks.Add(chosenTrack);
                availableTracks.Remove(chosenTrack);

                // Minus one from the count
                count--;
            }

            //cycle through the used track list and print them in the console
            foreach (string track in usedTracks)
            {
                print(track);
            }

            //disable the shuffle process
            cycleThroughList = false;
        }
    }
```



```
        public void ShuffleSongs()
        {
            //Clear the console window
            var assembly = Assembly.GetAssembly(typeof(UnityEditor.Editor));
            var type = assembly.GetType("UnityEditor.LogEntries");
            var method = type.GetMethod("Clear");
            method.Invoke(new object(), null);



            //If there are tracks in the used tracks list, send them back and clear the list
            if (usedTracks.Count > 0)
            {
                availableTracks = new List<string>(usedTracks);
                foreach (string track in usedTracks.ToArray())
                {
                    usedTracks.Remove(track);
                }
            }
            //turn on bool to start shuffle process
            cycleThroughList = true;

        }
    }

```

## Old Money System

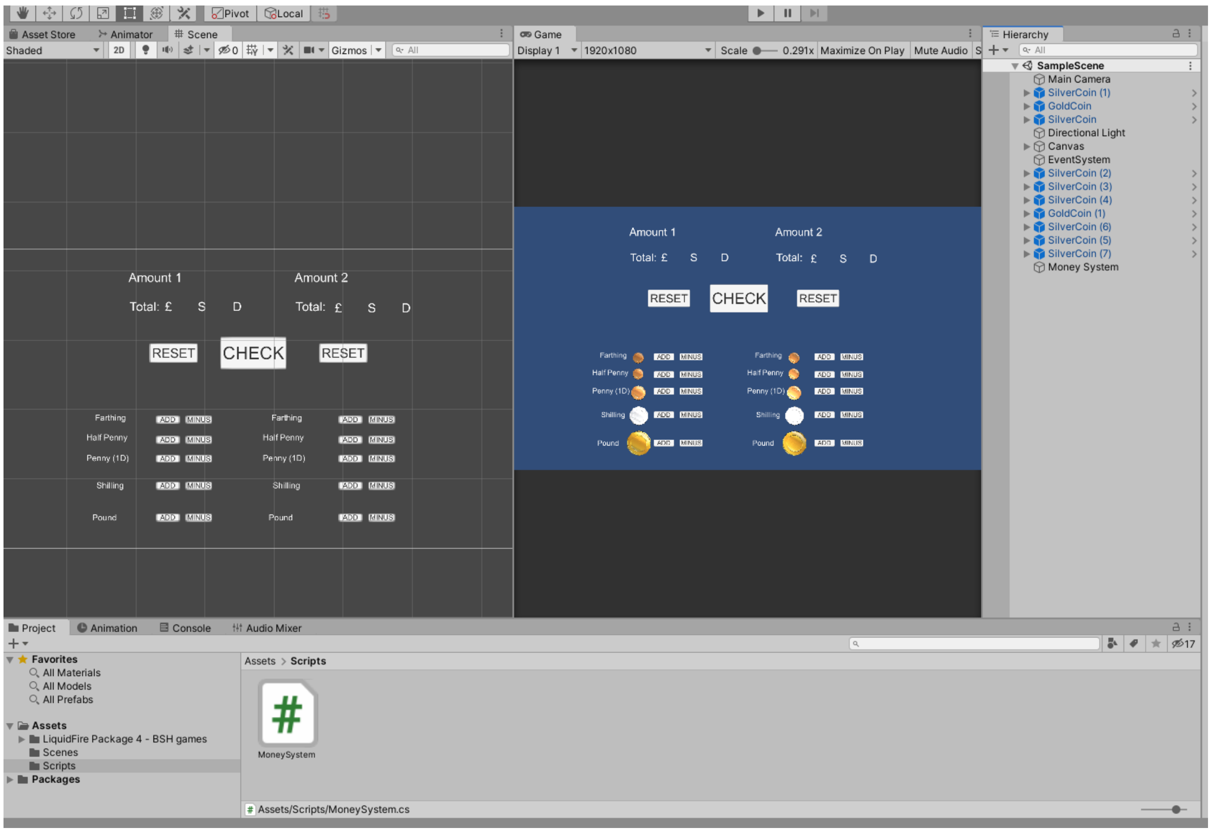
**06/04/21** -

For this briefing I had to create an old money system that was able to represent monetary values in terms of pound sterling (£), Shillings (S) and Pennies (D). This system would have to be able represent two amounts with functions to be able to add and subtract values. Lastly it should feature a function that compares the two amounts to each other and is able to detect which amount is equal, greater or less than the other.

First things first I have to set up the unity scene, I created multiple sets of UI text objects to represent the amounts, total and display the name of each monetary value that will be used within this project. The values I had chosen to use were:

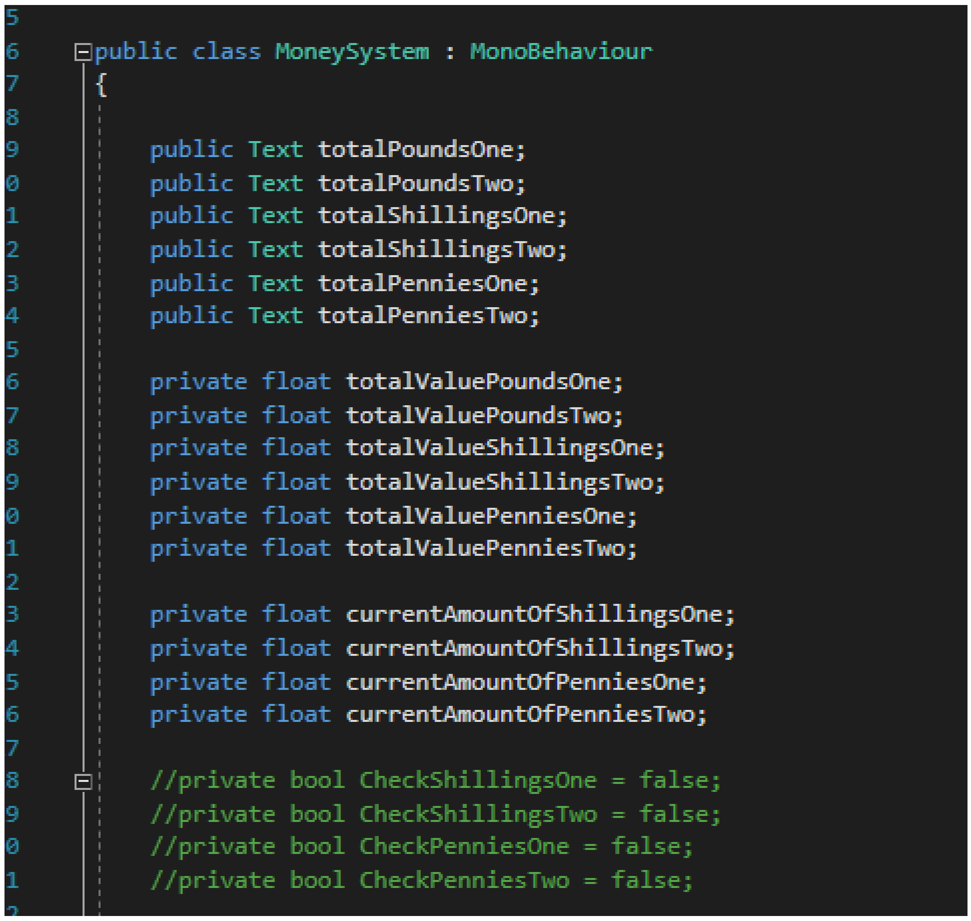
- Pounds (£)
- Shilling (S)
- Penny (1D)
- Half Penny (½ D)
- Farthing (¼ D)

I then downloaded some free coin assets from the asset store and sized/placed them within the scene to give a physical representation of the coin value next to it. Lastly I created multiple buttons to represent adding and subtracting the money values, a reset button for each amount and a large check button in the centre to perform the comparison checks for both amounts.



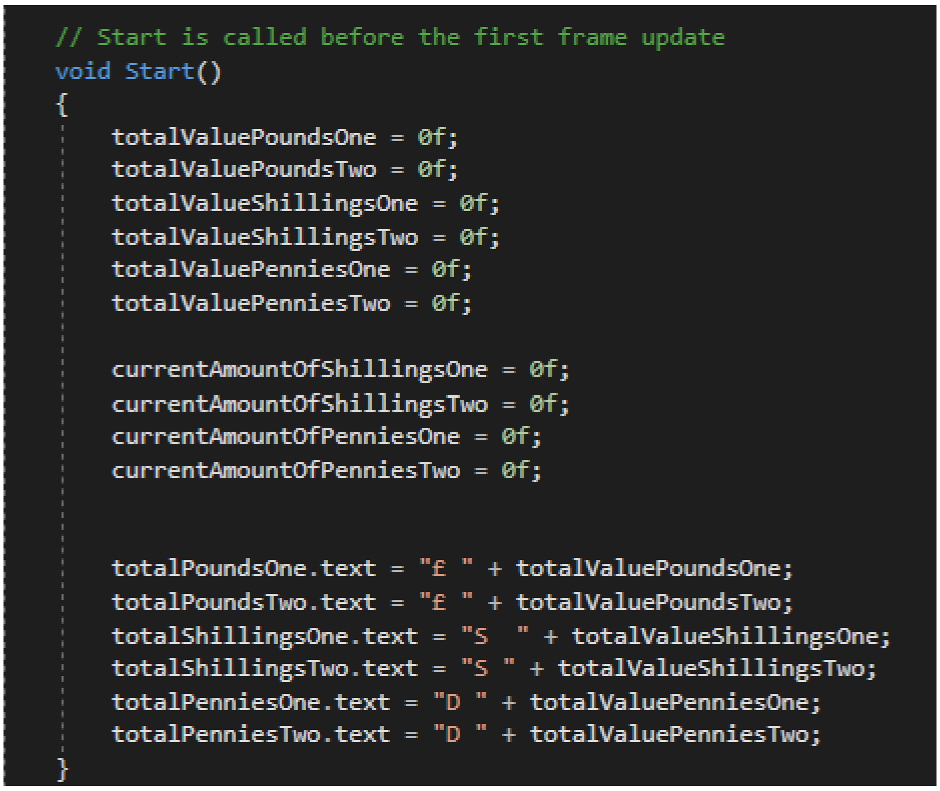
Once the scene was set up I spent the next hour researching the old fashioned currency online to gain a better understanding of the individual values of each and how they convert into one another. I then created a script named “MoneySystem” and opened it up within Microsoft Visual Studios.

My plan was to create multiple public methods to add and subtract from each list, inside these functions I would have to run checks to see if certain currency values reached a set amount to be able to be converted into the next currency up. I added in “using UnityEngine.Ui;” at the top to have access to UI elements and then created public references to all my text objects, I created multiple private floats to represent the final value of each currency twice for each amount, another set of floats were created to represent the current amount of each value which would be used to see if it had hit a set value to be able to be converted up (This would later not be used after I realised it would not work nor be efficient to do it this way). Lastly I created a set of booleans that would be set to true when a function was called to add or subtract values (also not used in the end).



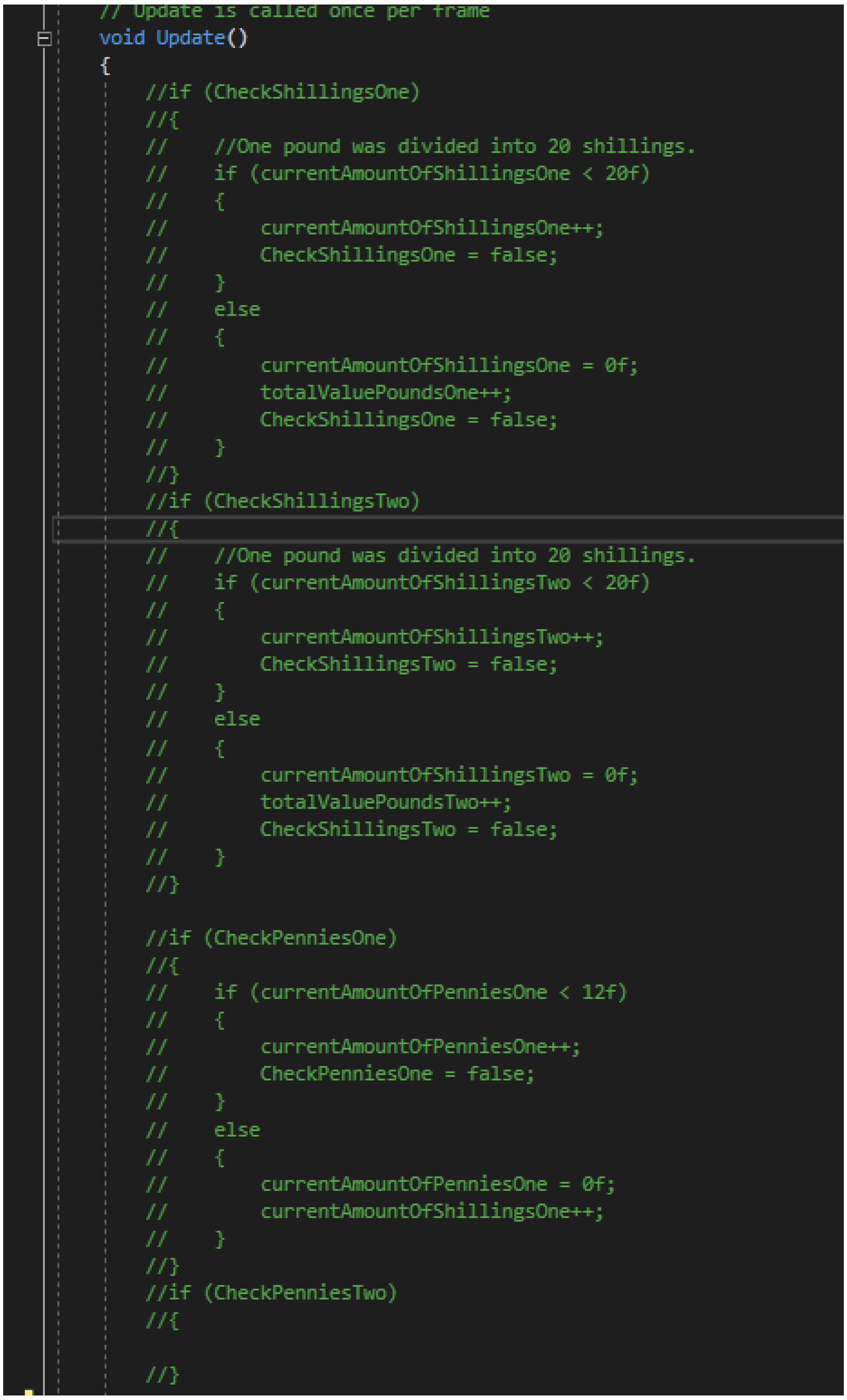
```
public class MoneySystem : MonoBehaviour
{

    public Text totalPoundsOne;
    public Text totalPoundsTwo;
    public Text totalShillingsOne;
    public Text totalShillingsTwo;
    public Text totalPenniesOne;
    public Text totalPenniesTwo;

    private float totalValuePoundsOne;
    private float totalValuePoundsTwo;
    private float totalValueShillingsOne;
    private float totalValueShillingsTwo;
    private float totalValuePenniesOne;
    private float totalValuePenniesTwo;

    private float currentAmountOfShillingsOne;
    private float currentAmountOfShillingsTwo;
    private float currentAmountOfPenniesOne;
    private float currentAmountOfPenniesTwo;

    //private bool CheckShillingsOne = false;
    //private bool CheckShillingsTwo = false;
    //private bool CheckPenniesOne = false;
    //private bool CheckPenniesTwo = false;
```

Inside the start method I made sure to set all final values to 0, current values to 0 and to display 0 at the start on the UI text objects.



```
// Start is called before the first frame update
void Start()
{
    totalValuePoundsOne = 0f;
    totalValuePoundsTwo = 0f;
    totalValueShillingsOne = 0f;
    totalValueShillingsTwo = 0f;
    totalValuePenniesOne = 0f;
    totalValuePenniesTwo = 0f;

    currentAmountOfShillingsOne = 0f;
    currentAmountOfShillingsTwo = 0f;
    currentAmountOfPenniesOne = 0f;
    currentAmountOfPenniesTwo = 0f;


    totalPoundsOne.text = "£ " + totalValuePoundsOne;
    totalPoundsTwo.text = "£ " + totalValuePoundsTwo;
    totalShillingsOne.text = "S  " + totalValueShillingsOne;
    totalShillingsTwo.text = "S " + totalValueShillingsTwo;
    totalPenniesOne.text = "D " + totalValuePenniesOne;
    totalPenniesTwo.text = "D " + totalValuePenniesTwo;
}
```

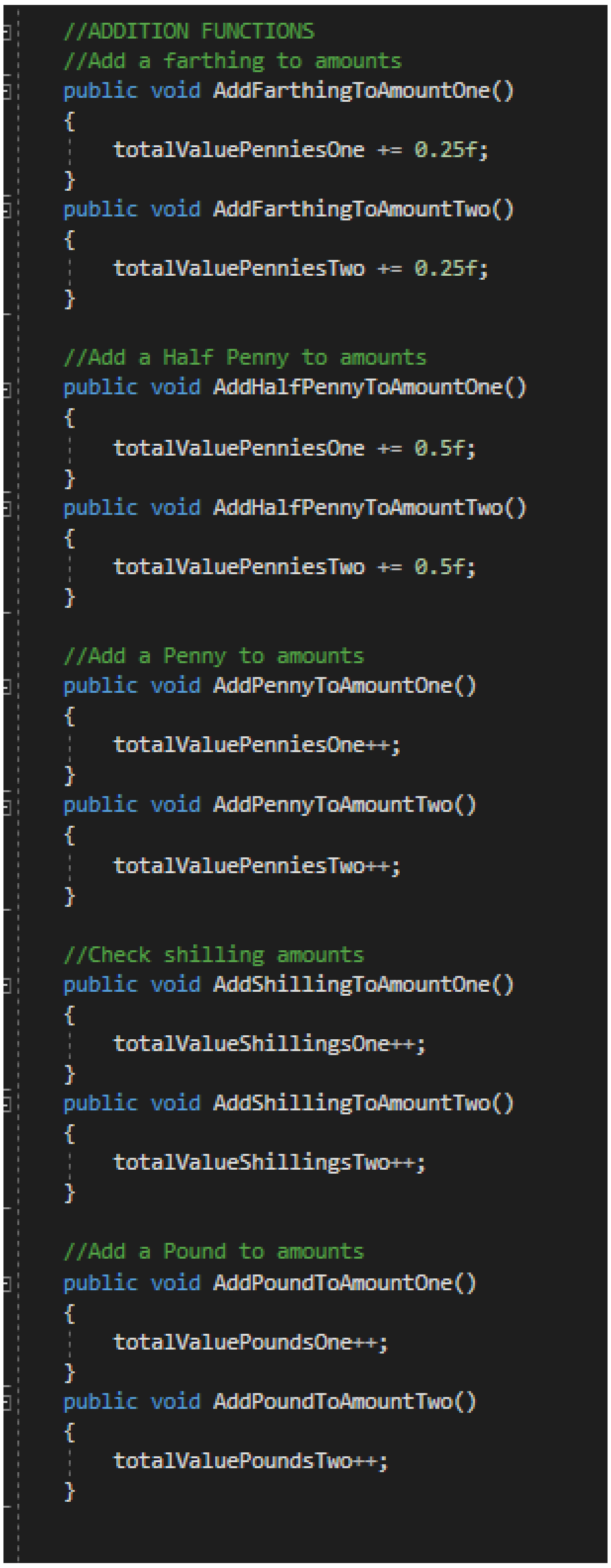
I used the same code to display the final values of each currency in the start within the update method to continuously display them during run time. I then went ahead and created public methods for adding and subtracting each value twice for each amount. Inside each function I would set the appropriate boolean to true, back in update I then ran an if statement for the booleans that would check if the current amount value of that coin had reached a set point to be converted to the coin above and if not increased its value by one, If so then reset its value to 0 and increase the value of the coin up by one. This is where I realised the error of my method to approaching this brief, I had made it more complex than it simply needed to be using two versions of the same currency, the total and the current also there was really no need to use booleans so I scraped them and changed my method to doing this with simply directly affecting the total values inside the button functions and have simple checks inside update to then convert them if needed. I took a break at this point to move on to other work and come back to this.



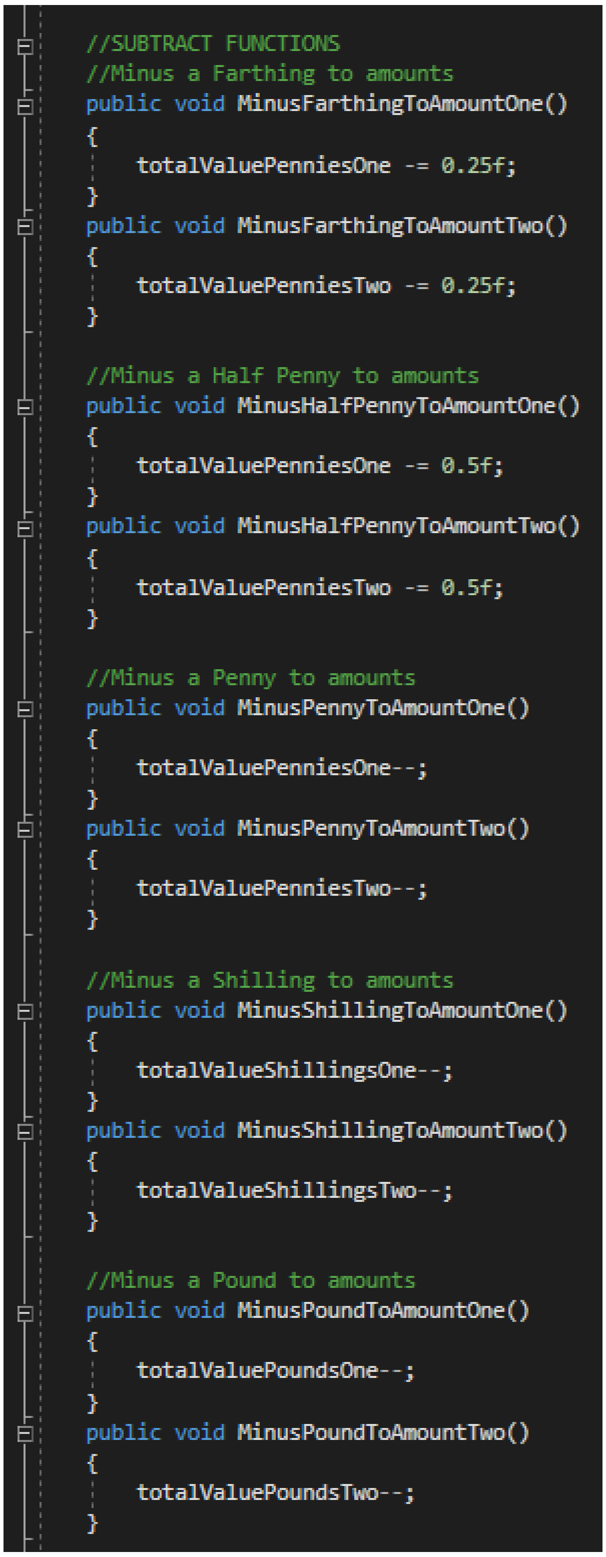
```
    // Update is called once per frame
    void Update()
    {
        //if (CheckShillingsOne)
        //{
        //    //One pound was divided into 20 shillings.
        //    if (currentAmountOfShillingsOne < 20f)
        //    {
        //        currentAmountOfShillingsOne++;
        //        CheckShillingsOne = false;
        //    }
        //    else
        //    {
        //        currentAmountOfShillingsOne = 0f;
        //        totalValuePoundsOne++;
        //        CheckShillingsOne = false;
        //    }
        //}
        //if (CheckShillingsTwo)
        //{
        //    //One pound was divided into 20 shillings.
        //    if (currentAmountOfShillingsTwo < 20f)
        //    {
        //        currentAmountOfShillingsTwo++;
        //        CheckShillingsTwo = false;
        //    }
        //    else
        //    {
        //        currentAmountOfShillingsTwo = 0f;
        //        totalValuePoundsTwo++;
        //        CheckShillingsTwo = false;
        //    }
        //}

        //if (CheckPenniesOne)
        //{
        //    if (currentAmountOfPenniesOne < 12f)
        //    {
        //        currentAmountOfPenniesOne++;
        //        CheckPenniesOne = false;
        //    }
        //    else
        //    {
        //        currentAmountOfPenniesOne = 0f;
        //        currentAmountOfShillingsOne++;
        //    }
        //}
        //if (CheckPenniesTwo)
        //{

        //}
```

**13/04/21 -**

After coming back to this I had written myself some notes to remind myself of the new method I wanted to use to do this brief and set about re writing the button functions, I simply either increased or decreased the total values of each monetary value by the respective amounts for that function, a pound would increase by 1 whereas a half penny would increase by 0.5f and farthings by 0.25f.

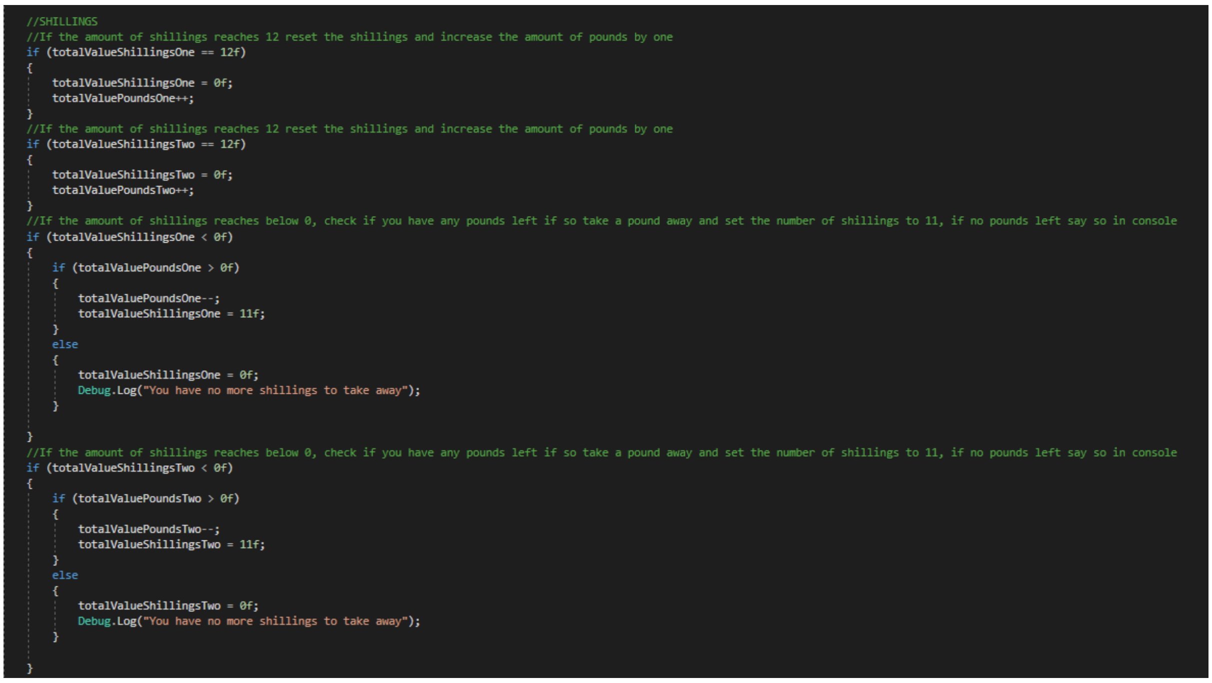


```
//ADDITION FUNCTIONS
//Add a farthing to amounts
public void AddFarthingToAmountOne()
{
    totalValuePenniesOne += 0.25f;
}
public void AddFarthingToAmountTwo()
{
    totalValuePenniesTwo += 0.25f;
}

//Add a Half Penny to amounts
public void AddHalfPennyToAmountOne()
{
    totalValuePenniesOne += 0.5f;
}
public void AddHalfPennyToAmountTwo()
{
    totalValuePenniesTwo += 0.5f;
}

//Add a Penny to amounts
public void AddPennyToAmountOne()
{
    totalValuePenniesOne++;
}
public void AddPennyToAmountTwo()
{
    totalValuePenniesTwo++;
}

//Check shilling amounts
public void AddShillingToAmountOne()
{
    totalValueShillingsOne++;
}
public void AddShillingToAmountTwo()
{
    totalValueShillingsTwo++;
}

//Add a Pound to amounts
public void AddPoundToAmountOne()
{
    totalValuePoundsOne++;
}
public void AddPoundToAmountTwo()
{
    totalValuePoundsTwo++;
}
```



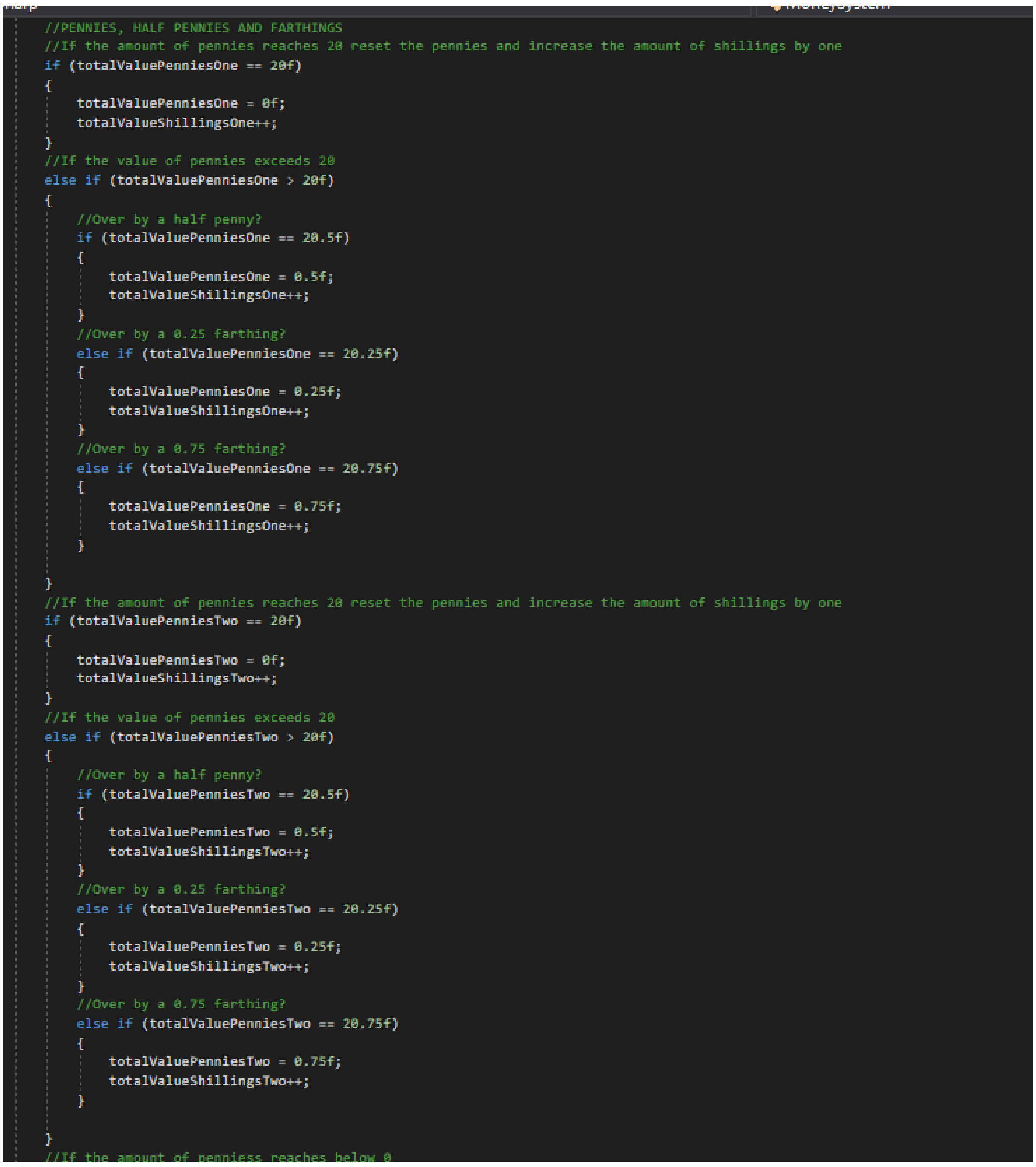
```
//SUBTRACT FUNCTIONS
//Minus a Farthing to amounts
public void MinusFarthingToAmountOne()
{
    totalValuePenniesOne -= 0.25f;
}
public void MinusFarthingToAmountTwo()
{
    totalValuePenniesTwo -= 0.25f;
}

//Minus a Half Penny to amounts
public void MinusHalfPennyToAmountOne()
{
    totalValuePenniesOne -= 0.5f;
}
public void MinusHalfPennyToAmountTwo()
{
    totalValuePenniesTwo -= 0.5f;
}

//Minus a Penny to amounts
public void MinusPennyToAmountOne()
{
    totalValuePenniesOne--;
}
public void MinusPennyToAmountTwo()
{
    totalValuePenniesTwo--;
}

//Minus a Shilling to amounts
public void MinusShillingToAmountOne()
{
    totalValueShillingsOne--;
}
public void MinusShillingToAmountTwo()
{
    totalValueShillingsTwo--;
}

//Minus a Pound to amounts
public void MinusPoundToAmountOne()
{
    totalValuePoundsOne--;
}
public void MinusPoundToAmountTwo()
{
    totalValuePoundsTwo--;
}
```

I then went back to the update method to run if statements that would check if the amount of that currency had reached its respective conversion point. For example, there are 12 shillings in a pound and 20 pennies in a shilling. I would have to run all if statements twice once for each amount. I also ran statements that would check if that currency value had gone below 0 if the user was to subtract from it and if so it would then check if the currency above was greater than 0, if so it would reduce that currency by 1 and set the value of the current value to be 1 before it’s conversion point. Finally if there was no amount above that currency to expend once it goes below 0 I would run a “Debug.Log” within the console telling the user they have no more of that currency left to decrease. I repeated this process for all currency values.

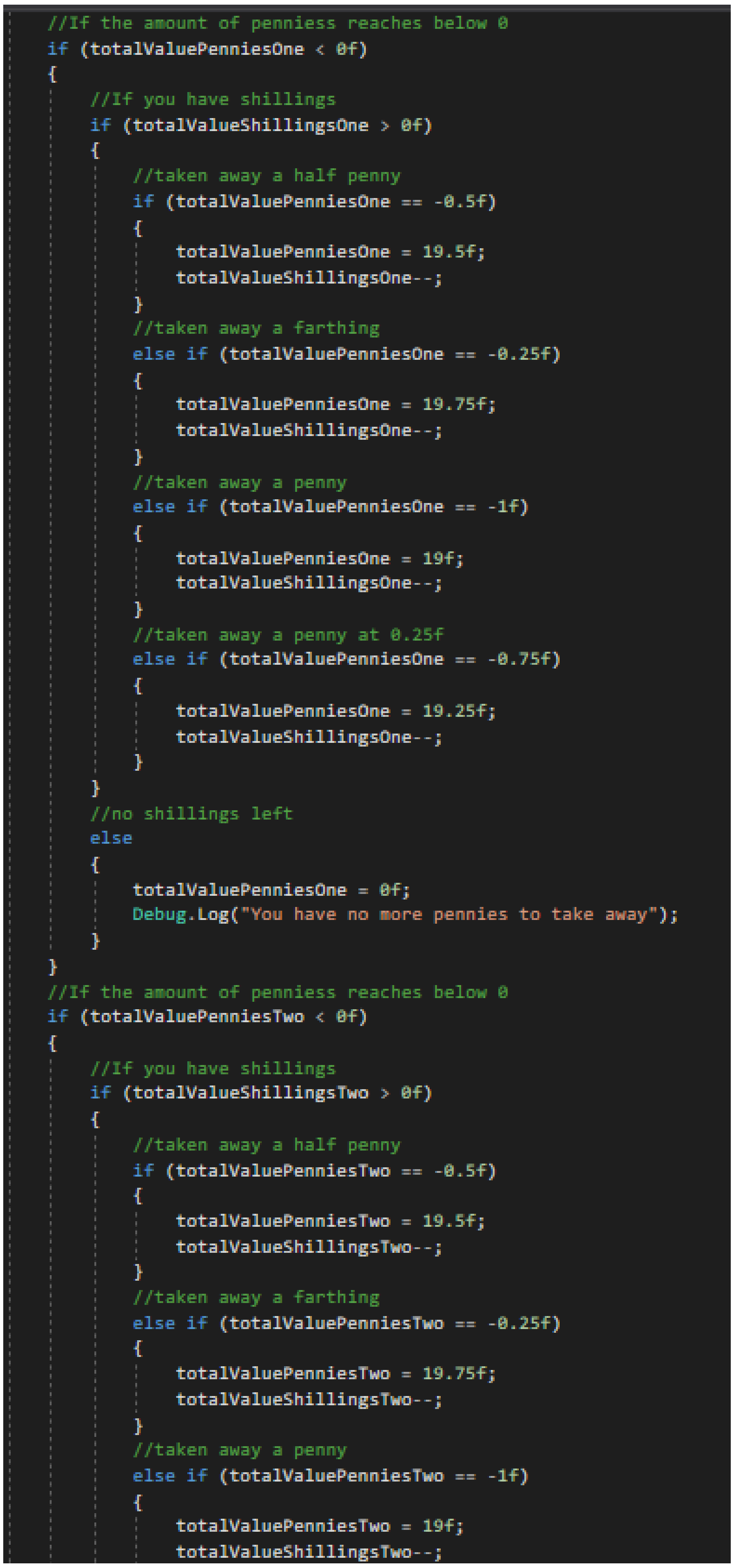


```
//SHILLINGS
//If the amount of shillings reaches 12 reset the shillings and increase the amount of pounds by one
if (totalValueShillingsOne == 12f)
{
    totalValueShillingsOne = 0f;
    totalValuePoundsOne++;
}
//If the amount of shillings reaches 12 reset the shillings and increase the amount of pounds by one
if (totalValueShillingsTwo == 12f)
{
    totalValueShillingsTwo = 0f;
    totalValuePoundsTwo++;
}
//If the amount of shillings reaches below 0, check if you have any pounds left if so take a pound away and set the number of shillings to 11, if no pounds left say so in console
if (totalValueShillingsOne < 0f)
{
    if (totalValuePoundsOne > 0f)
    {
        totalValuePoundsOne--;
        totalValueShillingsOne = 11f;
    }
    else
    {
        totalValueShillingsOne = 0f;
        Debug.Log("You have no more shillings to take away");
    }

}
//If the amount of shillings reaches below 0, check if you have any pounds left if so take a pound away and set the number of shillings to 11, if no pounds left say so in console
if (totalValueShillingsTwo < 0f)
{
    if (totalValuePoundsTwo > 0f)
    {
        totalValuePoundsTwo--;
        totalValueShillingsTwo = 11f;
    }
    else
    {
        totalValueShillingsTwo = 0f;
        Debug.Log("You have no more shillings to take away");
    }

}
```

The pennies would take more programming to sort through as there was also half pennies and farthings to take into consideration, especially when it came to running checks if they go over the set conversion rate to be changed into Shillings, if the current amount of pennies was 19 pennies and 1 half penny equaling 19.5f and the user added a penny on top it would not convert being the result would not equal 20f but rather 20.5f. I created further if statements to check for all possible over and under values and then converted them accordingly.

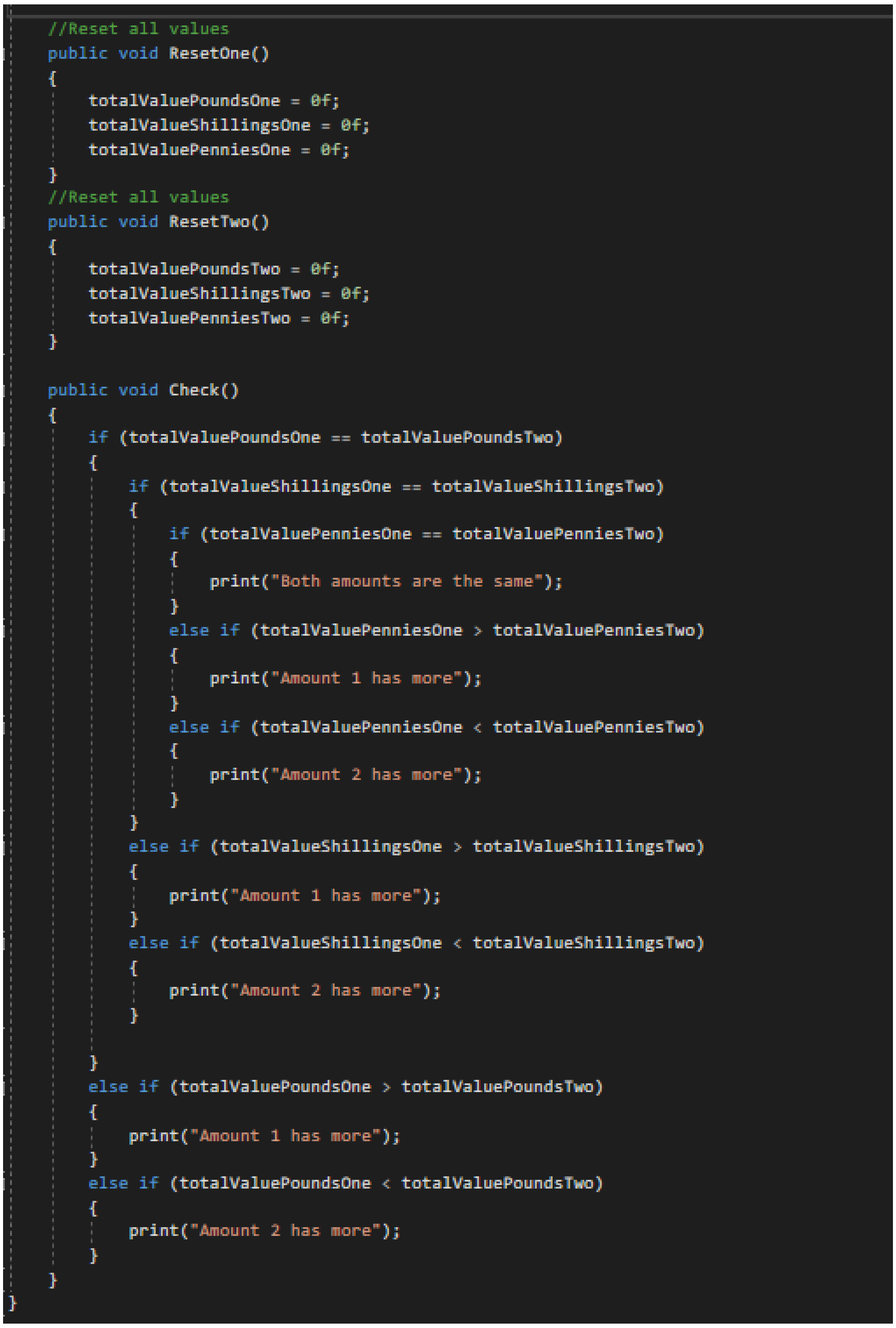


```
//PENNIES, HALF PENNIES AND FARTHINGS
//If the amount of pennies reaches 20 reset the pennies and increase the amount of shillings by one
if (totalValuePenniesOne == 20f)
{
    totalValuePenniesOne = 0f;
    totalValueShillingsOne++;
}
//If the value of pennies exceeds 20
else if (totalValuePenniesOne > 20f)
{
    //Over by a half penny?
    if (totalValuePenniesOne == 20.5f)
    {
        totalValuePenniesOne = 0.5f;
        totalValueShillingsOne++;
    }
    //Over by a 0.25 farthing?
    else if (totalValuePenniesOne == 20.25f)
    {
        totalValuePenniesOne = 0.25f;
        totalValueShillingsOne++;
    }
    //Over by a 0.75 farthing?
    else if (totalValuePenniesOne == 20.75f)
    {
        totalValuePenniesOne = 0.75f;
        totalValueShillingsOne++;
    }

}
//If the amount of pennies reaches 20 reset the pennies and increase the amount of shillings by one
if (totalValuePenniesTwo == 20f)
{
    totalValuePenniesTwo = 0f;
    totalValueShillingsTwo++;
}
//If the value of pennies exceeds 20
else if (totalValuePenniesTwo > 20f)
{
    //Over by a half penny?
    if (totalValuePenniesTwo == 20.5f)
    {
        totalValuePenniesTwo = 0.5f;
        totalValueShillingsTwo++;
    }
    //Over by a 0.25 farthing?
    else if (totalValuePenniesTwo == 20.25f)
    {
        totalValuePenniesTwo = 0.25f;
        totalValueShillingsTwo++;
    }
    //Over by a 0.75 farthing?
    else if (totalValuePenniesTwo == 20.75f)
    {
        totalValuePenniesTwo = 0.75f;
        totalValueShillingsTwo++;
    }

}
//If the amount of penniess reaches below 0
```



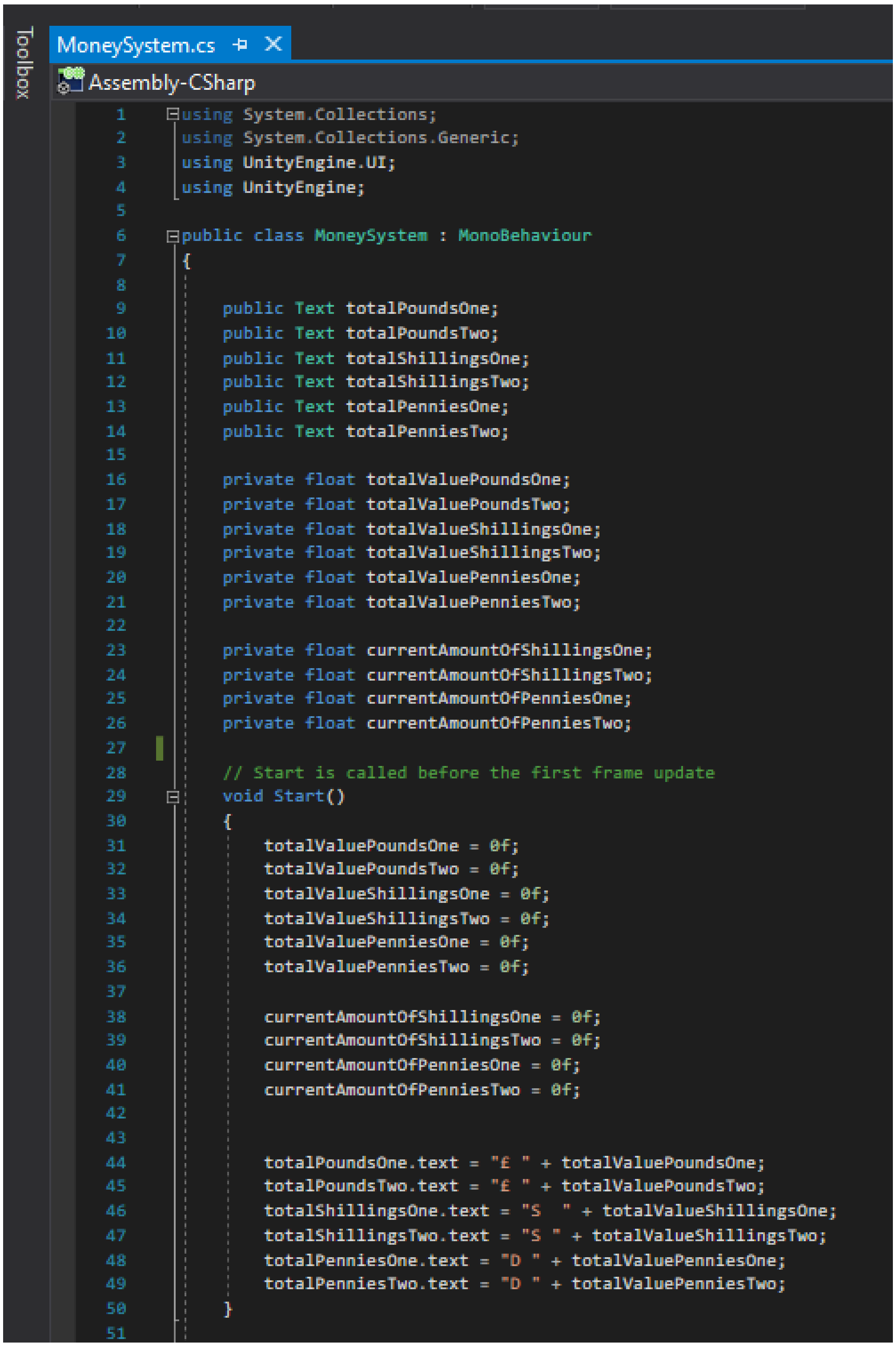
```
//If the amount of penniess reaches below 0
if (totalValuePenniesOne < 0f)
{
    //If you have shillings
    if (totalValueShillingsOne > 0f)
    {
        //taken away a half penny
        if (totalValuePenniesOne == -0.5f)
        {
            totalValuePenniesOne = 19.5f;
            totalValueShillingsOne--;
        }
        //taken away a farthing
        else if (totalValuePenniesOne == -0.25f)
        {
            totalValuePenniesOne = 19.75f;
            totalValueShillingsOne--;
        }
        //taken away a penny
        else if (totalValuePenniesOne == -1f)
        {
            totalValuePenniesOne = 19f;
            totalValueShillingsOne--;
        }
        //taken away a penny at 0.25f
        else if (totalValuePenniesOne == -0.75f)
        {
            totalValuePenniesOne = 19.25f;
            totalValueShillingsOne--;
        }
    }
    //no shillings left
    else
    {
        totalValuePenniesOne = 0f;
        Debug.Log("You have no more pennies to take away");
    }
}
//If the amount of penniess reaches below 0
if (totalValuePenniesTwo < 0f)
{
    //If you have shillings
    if (totalValueShillingsTwo > 0f)
    {
        //taken away a half penny
        if (totalValuePenniesTwo == -0.5f)
        {
            totalValuePenniesTwo = 19.5f;
            totalValueShillingsTwo--;
        }
        //taken away a farthing
        else if (totalValuePenniesTwo == -0.25f)
        {
            totalValuePenniesTwo = 19.75f;
            totalValueShillingsTwo--;
        }
        //taken away a penny
        else if (totalValuePenniesTwo == -1f)
        {
            totalValuePenniesTwo = 19f;
            totalValueShillingsTwo--;
```

Now approaching the end of my script I made sure to reset all total values of each currency for both amounts within the rest button functions and then it was time to programme the check button function, using this function would run a series of if statements to check the values of the pound first for both amounts, if they were equal it would then check shillings, if they were also equal it would then check pennies. For each value that was either an equal, greater than or less than result it would run a print inside the console window to display which amount had more or to display if they were both the exact same amount.

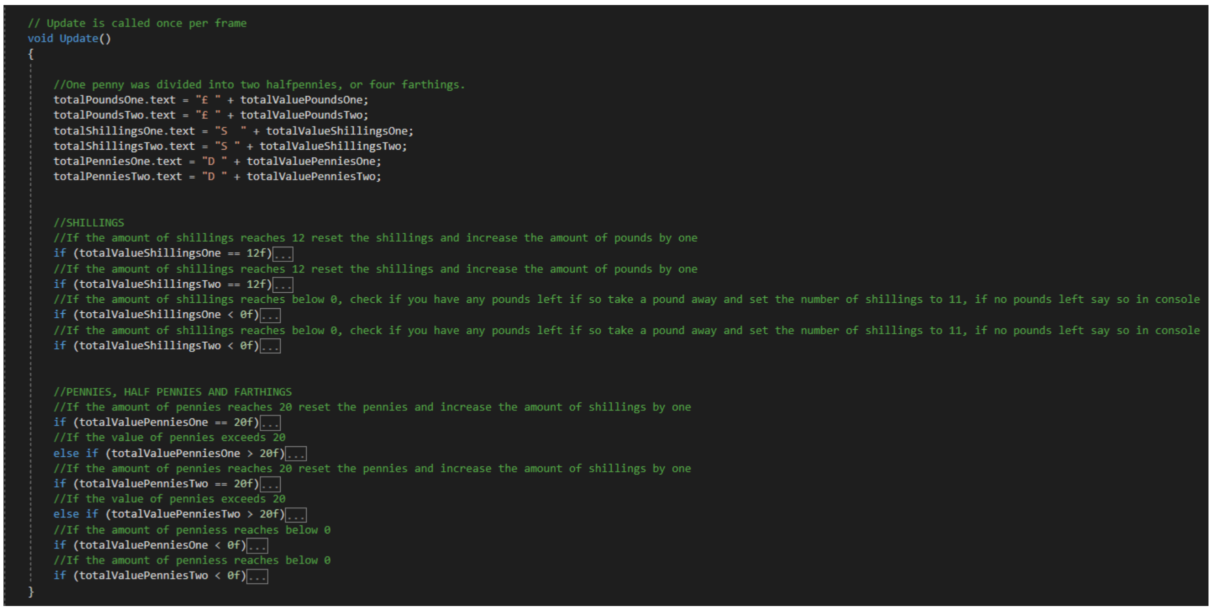


```
    //Reset all values
    public void ResetOne()
    {
        totalValuePoundsOne = 0f;
        totalValueShillingsOne = 0f;
        totalValuePenniesOne = 0f;
    }
    //Reset all values
    public void ResetTwo()
    {
        totalValuePoundsTwo = 0f;
        totalValueShillingsTwo = 0f;
        totalValuePenniesTwo = 0f;
    }

    public void Check()
    {
        if (totalValuePoundsOne == totalValuePoundsTwo)
        {
            if (totalValueShillingsOne == totalValueShillingsTwo)
            {
                if (totalValuePenniesOne == totalValuePenniesTwo)
                {
                    print("Both amounts are the same");
                }
                else if (totalValuePenniesOne > totalValuePenniesTwo)
                {
                    print("Amount 1 has more");
                }
                else if (totalValuePenniesOne < totalValuePenniesTwo)
                {
                    print("Amount 2 has more");
                }
            }
            else if (totalValueShillingsOne > totalValueShillingsTwo)
            {
                print("Amount 1 has more");
            }
            else if (totalValueShillingsOne < totalValueShillingsTwo)
            {
                print("Amount 2 has more");
            }

        }
        else if (totalValuePoundsOne > totalValuePoundsTwo)
        {
            print("Amount 1 has more");
        }
        else if (totalValuePoundsOne < totalValuePoundsTwo)
        {
            print("Amount 2 has more");
        }
    }
}
```

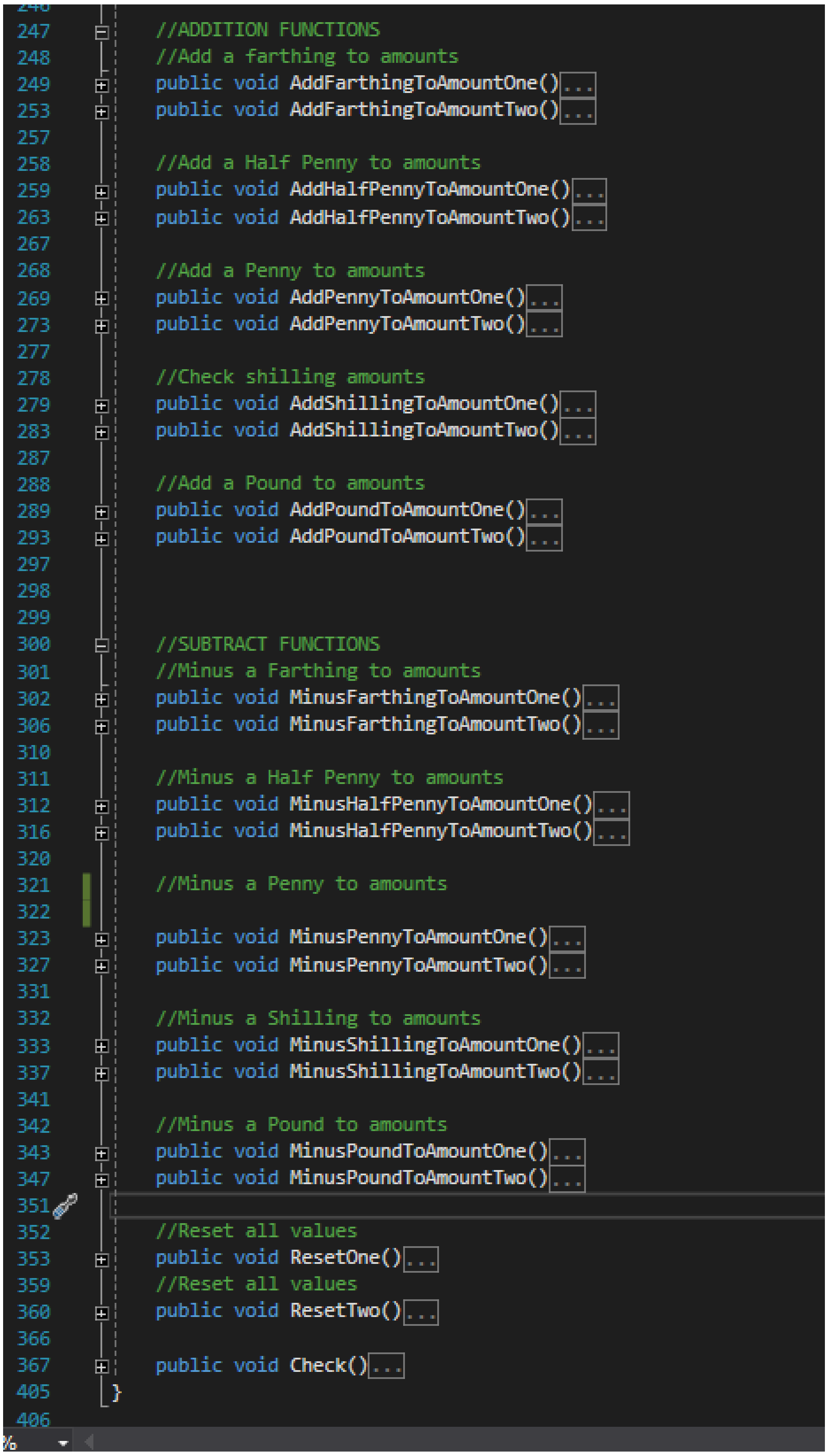
Now all of the programming was done the last thing I had to do was attach this script to an empty game object and then apply my public references within the inspector and create the link to each buttons function on their inspector window. Once done everything was checked and in working order. Here is the overview of the whole script.



```csharp
using System.Collections;
using System.Collections.Generic;
using UnityEngine.UI;
using UnityEngine;

public class MoneySystem : MonoBehaviour
{

    public Text totalPoundsOne;
    public Text totalPoundsTwo;
    public Text totalShillingsOne;
    public Text totalShillingsTwo;
    public Text totalPenniesOne;
    public Text totalPenniesTwo;

    private float totalValuePoundsOne;
    private float totalValuePoundsTwo;
    private float totalValueShillingsOne;
    private float totalValueShillingsTwo;
    private float totalValuePenniesOne;
    private float totalValuePenniesTwo;

    private float currentAmountOfShillingsOne;
    private float currentAmountOfShillingsTwo;
    private float currentAmountOfPenniesOne;
    private float currentAmountOfPenniesTwo;

    // Start is called before the first frame update
    void Start()
    {
        totalValuePoundsOne = 0f;
        totalValuePoundsTwo = 0f;
        totalValueShillingsOne = 0f;
        totalValueShillingsTwo = 0f;
        totalValuePenniesOne = 0f;
        totalValuePenniesTwo = 0f;

        currentAmountOfShillingsOne = 0f;
        currentAmountOfShillingsTwo = 0f;
        currentAmountOfPenniesOne = 0f;
        currentAmountOfPenniesTwo = 0f;


        totalPoundsOne.text = "£ " + totalValuePoundsOne;
        totalPoundsTwo.text = "£ " + totalValuePoundsTwo;
        totalShillingsOne.text = "S  " + totalValueShillingsOne;
        totalShillingsTwo.text = "S " + totalValueShillingsTwo;
        totalPenniesOne.text = "D " + totalValuePenniesOne;
        totalPenniesTwo.text = "D " + totalValuePenniesTwo;
    }

```



```
// Update is called once per frame
void Update()
{

    //One penny was divided into two halfpennies, or four farthings.
    totalPoundsOne.text = "£ " + totalValuePoundsOne;
    totalPoundsTwo.text = "£ " + totalValuePoundsTwo;
    totalShillingsOne.text = "S  " + totalValueShillingsOne;
    totalShillingsTwo.text = "S " + totalValueShillingsTwo;
    totalPenniesOne.text = "D " + totalValuePenniesOne;
    totalPenniesTwo.text = "D " + totalValuePenniesTwo;


    //SHILLINGS
    //If the amount of shillings reaches 12 reset the shillings and increase the amount of pounds by one
    if (totalValueShillingsOne == 12f)...
    //If the amount of shillings reaches 12 reset the shillings and increase the amount of pounds by one
    if (totalValueShillingsTwo == 12f)...
    //If the amount of shillings reaches below 0, check if you have any pounds left if so take a pound away and set the number of shillings to 11, if no pounds left say so in console
    if (totalValueShillingsOne < 0f)...
    //If the amount of shillings reaches below 0, check if you have any pounds left if so take a pound away and set the number of shillings to 11, if no pounds left say so in console
    if (totalValueShillingsTwo < 0f)...


    //PENNIES, HALF PENNIES AND FARTHINGS
    //If the amount of pennies reaches 20 reset the pennies and increase the amount of shillings by one
    if (totalValuePenniesOne == 20f)...
    //If the value of pennies exceeds 20
    else if (totalValuePenniesOne > 20f)...
    //If the amount of pennies reaches 20 reset the pennies and increase the amount of shillings by one
    if (totalValuePenniesTwo == 20f)...
    //If the value of pennies exceeds 20
    else if (totalValuePenniesTwo > 20f)...
    //If the amount of penniess reaches below 0
    if (totalValuePenniesOne < 0f)...
    //If the amount of penniess reaches below 0
    if (totalValuePenniesTwo < 0f)...
}
```


```
    //ADDITION FUNCTIONS
    //Add a farthing to amounts
    public void AddFarthingToAmountOne()...
    public void AddFarthingToAmountTwo()...

    //Add a Half Penny to amounts
    public void AddHalfPennyToAmountOne()...
    public void AddHalfPennyToAmountTwo()...

    //Add a Penny to amounts
    public void AddPennyToAmountOne()...
    public void AddPennyToAmountTwo()...

    //Check shilling amounts
    public void AddShillingToAmountOne()...
    public void AddShillingToAmountTwo()...

    //Add a Pound to amounts
    public void AddPoundToAmountOne()...
    public void AddPoundToAmountTwo()...



    //SUBTRACT FUNCTIONS
    //Minus a Farthing to amounts
    public void MinusFarthingToAmountOne()...
    public void MinusFarthingToAmountTwo()...

    //Minus a Half Penny to amounts
    public void MinusHalfPennyToAmountOne()...
    public void MinusHalfPennyToAmountTwo()...

    //Minus a Penny to amounts

    public void MinusPennyToAmountOne()...
    public void MinusPennyToAmountTwo()...

    //Minus a Shilling to amounts
    public void MinusShillingToAmountOne()...
    public void MinusShillingToAmountTwo()...

    //Minus a Pound to amounts
    public void MinusPoundToAmountOne()...
    public void MinusPoundToAmountTwo()...

    //Reset all values
    public void ResetOne()...
    //Reset all values
    public void ResetTwo()...

    public void Check()...
}
```