Triggers & Loops

Make a Infinite Runner

Objective

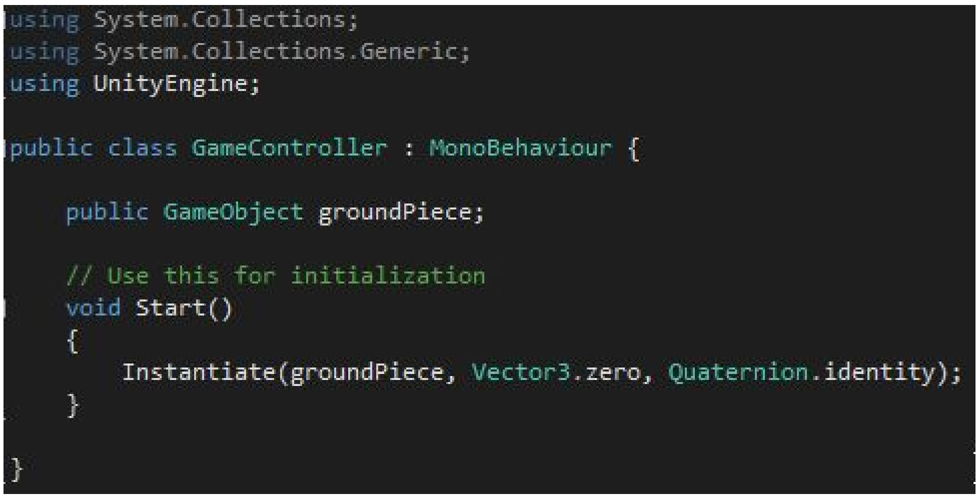
Make a runner that goes on forever and have the player move out of the way on oncoming objects

Setup

- Create a new project
- Import the provided runner package
- Make an empty game object called **GameController**
- Make script called **GameController**, attach it to this game object

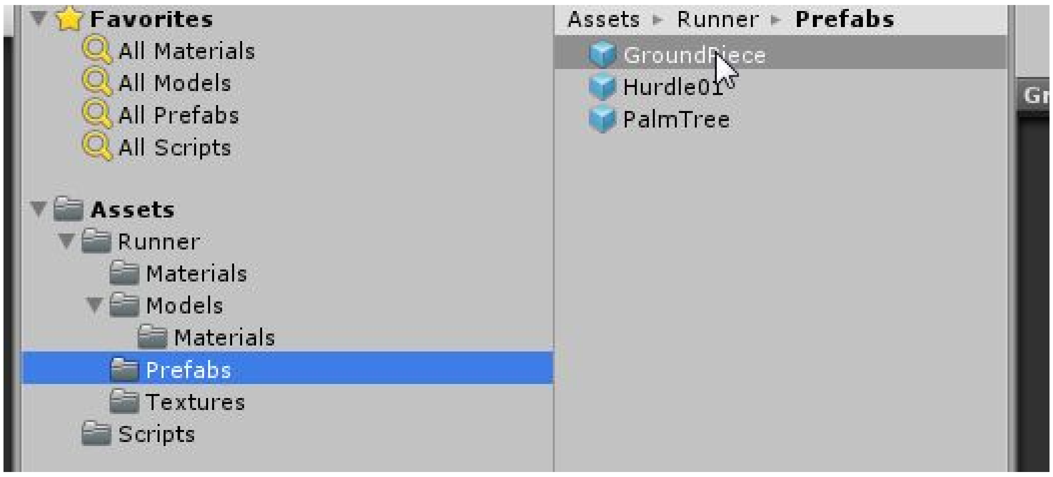
Lesson

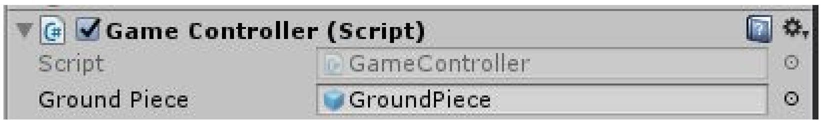
Edit the **GameController** script



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

public class GameController : MonoBehaviour {

    public GameObject groundPiece;

    // Use this for initialization
    void Start()
    {
        Instantiate(groundPiece, Vector3.zero, Quaternion.identity);
    }

}
```

Reference the **GroundPiece** prefab from the provided package and add the inspector

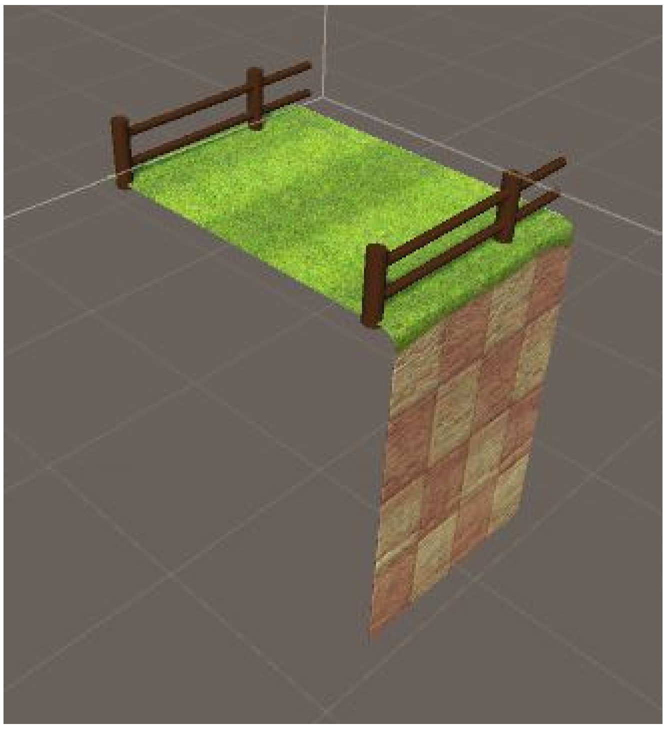




Game Controller (Script)

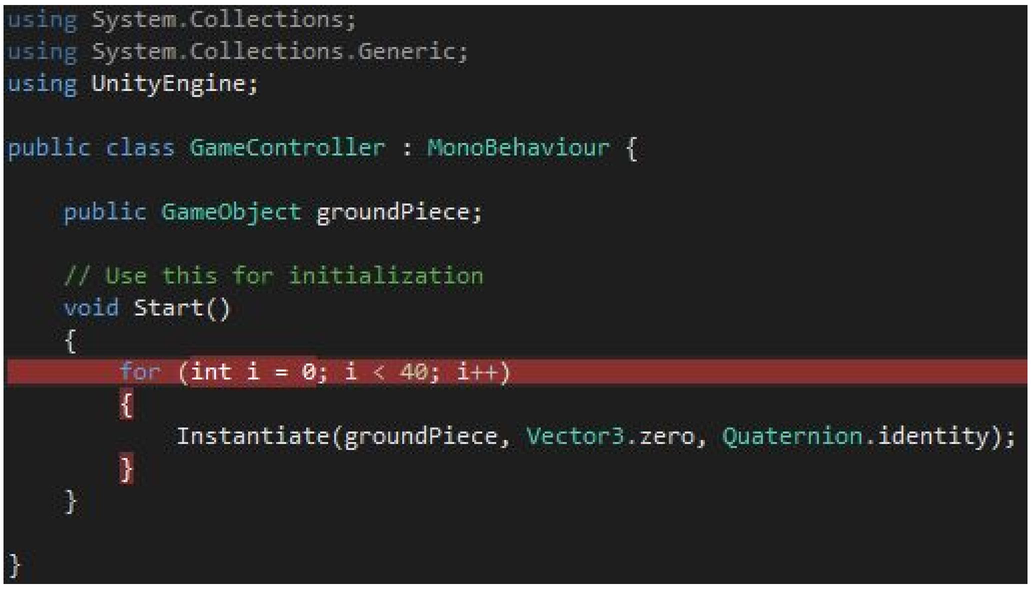
Script	GameController
Ground Piece	GroundPiece

Run the game and you should see that it just spawns or referenced prefab.



__Note:__ The Vector3.zero.This is the same as writing 'new Vector3(0,0,0)

Let's spawn more

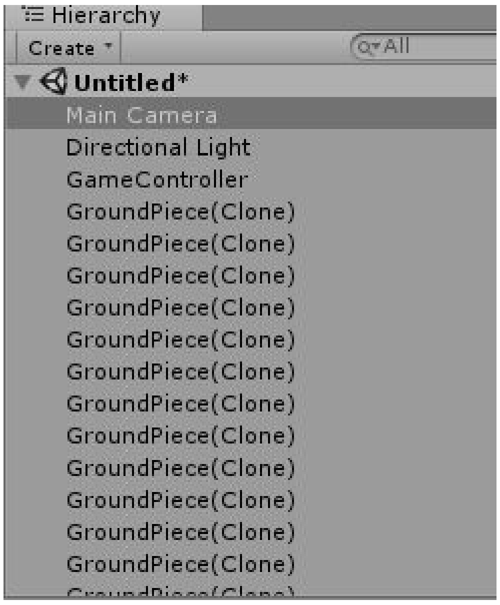


```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GameController : MonoBehaviour {

    public GameObject groundPiece;

    // Use this for initialization
    void Start()
    {
        for (int i = 0; i < 40; i++)
        {
            Instantiate(groundPiece, Vector3.zero, Quaternion.identity);
        }
    }

}
```

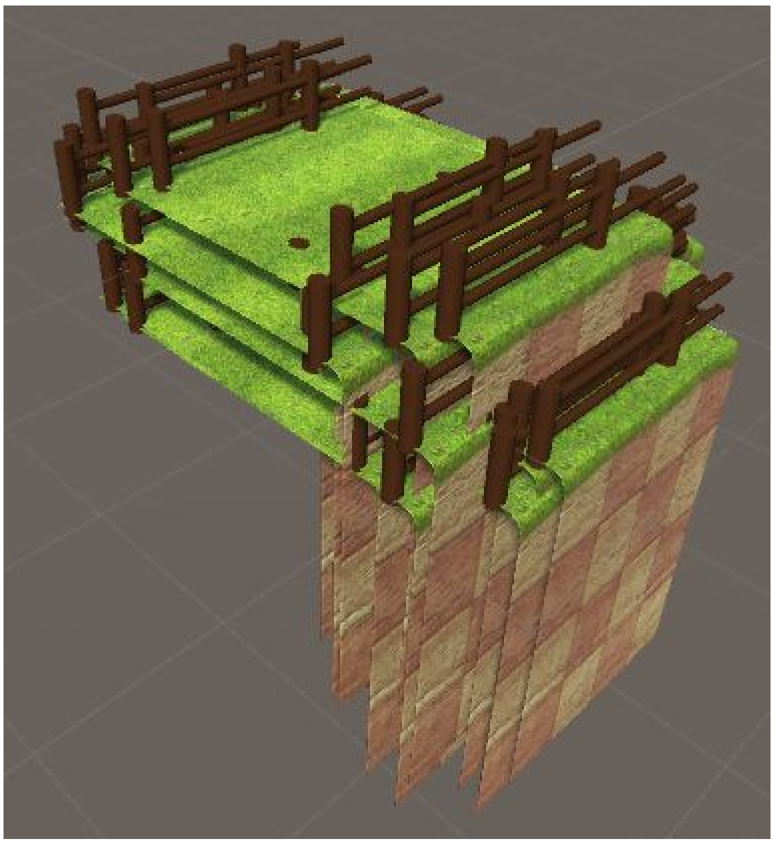
This is a for loop, It will loop 40 times around the instantiate code to spawn 40 ground pieces all on top of each other

***Note:** A shortcut for a for loop is to type “**for**” and then press tab twice to auto write the command.*

Run game again and it will look the same except…


Hierarchy
Create
All
Untitled*
Main Camera
Directional Light
GameController
GroundPiece(Clone)
GroundPiece(Clone)
GroundPiece(Clone)
GroundPiece(Clone)
GroundPiece(Clone)
GroundPiece(Clone)
GroundPiece(Clone)
GroundPiece(Clone)
GroundPiece(Clone)
GroundPiece(Clone)
GroundPiece(Clone)
GroundPiece(Clone)

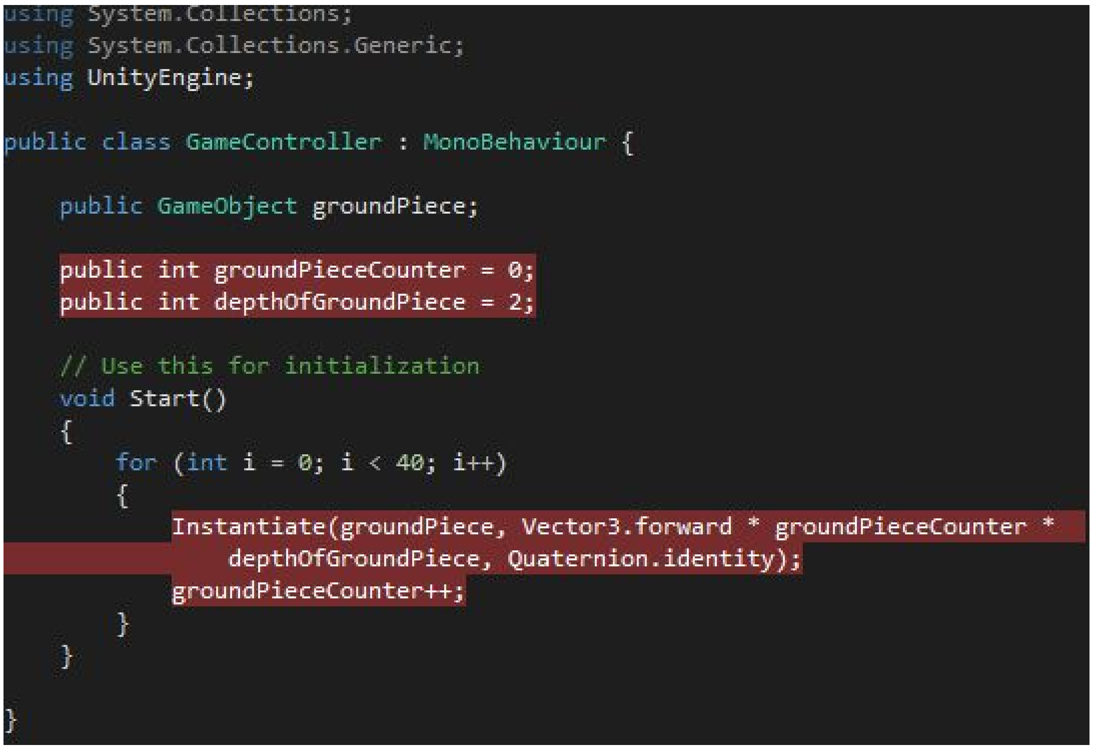
All the pieces are spawning on top of each other, If you drag them around you can see that.



Our aim is to have them placed one after each other in a straight line, not on top of each other.

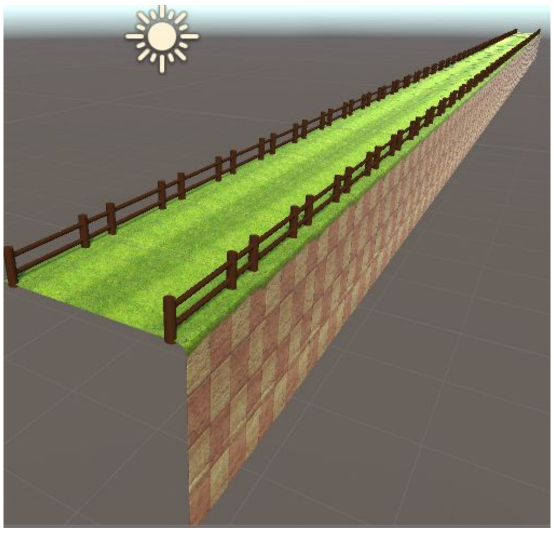
So let’s add a few thing to allow this to happen

Update our script.



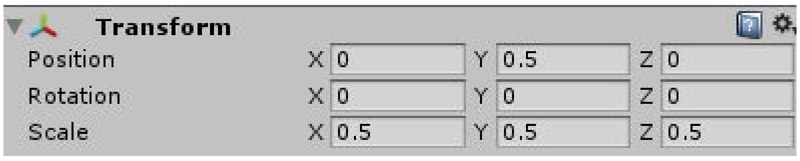
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GameController : MonoBehaviour {

    public GameObject groundPiece;

    public int groundPieceCounter = 0;
    public int depthOfGroundPiece = 2;

    // Use this for initialization
    void Start()
    {
        for (int i = 0; i < 40; i++)
        {
            Instantiate(groundPiece, Vector3.forward * groundPieceCounter *
                depthOfGroundPiece, Quaternion.identity);
            groundPieceCounter++;
        }
    }

}
```

Run this and you should see all our pieces are placed one after the other.



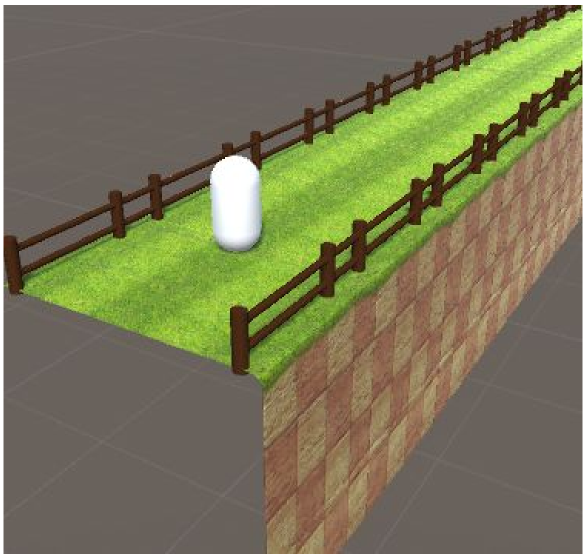
Let's make someone run along this road.

Make a capsule and set the transform to these values



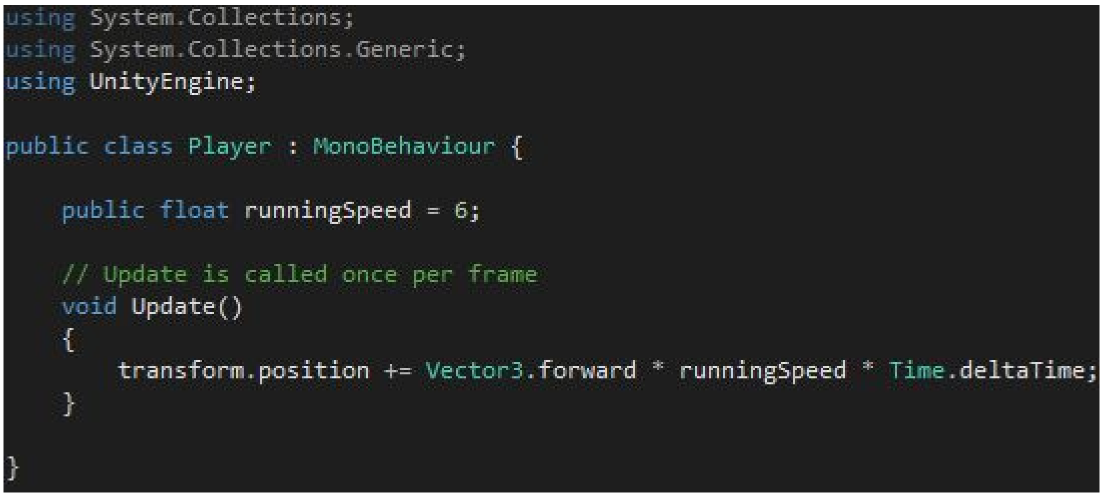
Transform			
Position	X 0	Y 0.5	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 0.5	Y 0.5	Z 0.5

This should start them at the start of the road in the center.



Rename the capsule to ‘**Player**’

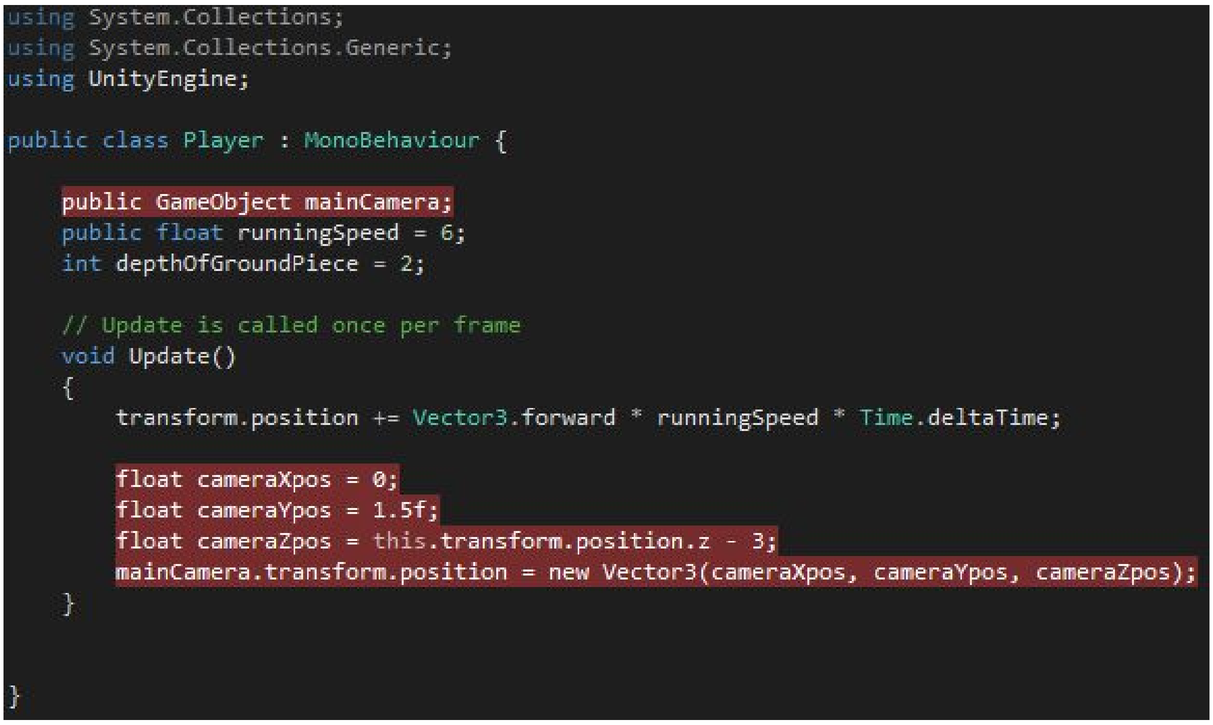
Make a script called **Player**
And place it on the capsule



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

public class Player : MonoBehaviour {

    public float runningSpeed = 6;

    // Update is called once per frame
    void Update()
    {
        transform.position += Vector3.forward * runningSpeed * Time.deltaTime;
    }

}
```

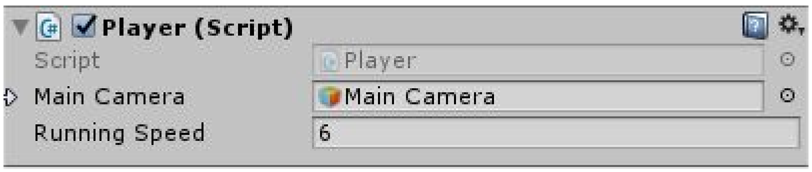
***Note:** that we are using Time.delta time to ensure this runs the same speed on any platform.*

Let's make the camera follow the player



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

public class Player : MonoBehaviour {

    public GameObject mainCamera;
    public float runningSpeed = 6;
    int depthOfGroundPiece = 2;

    // Update is called once per frame
    void Update()
    {
        transform.position += Vector3.forward * runningSpeed * Time.deltaTime;

        float cameraXpos = 0;
        float cameraYpos = 1.5f;
        float cameraZpos = this.transform.position.z - 3;
        mainCamera.transform.position = new Vector3(cameraXpos, cameraYpos, cameraZpos);
    }


}
```

Make sure you reference the camera in the inspector

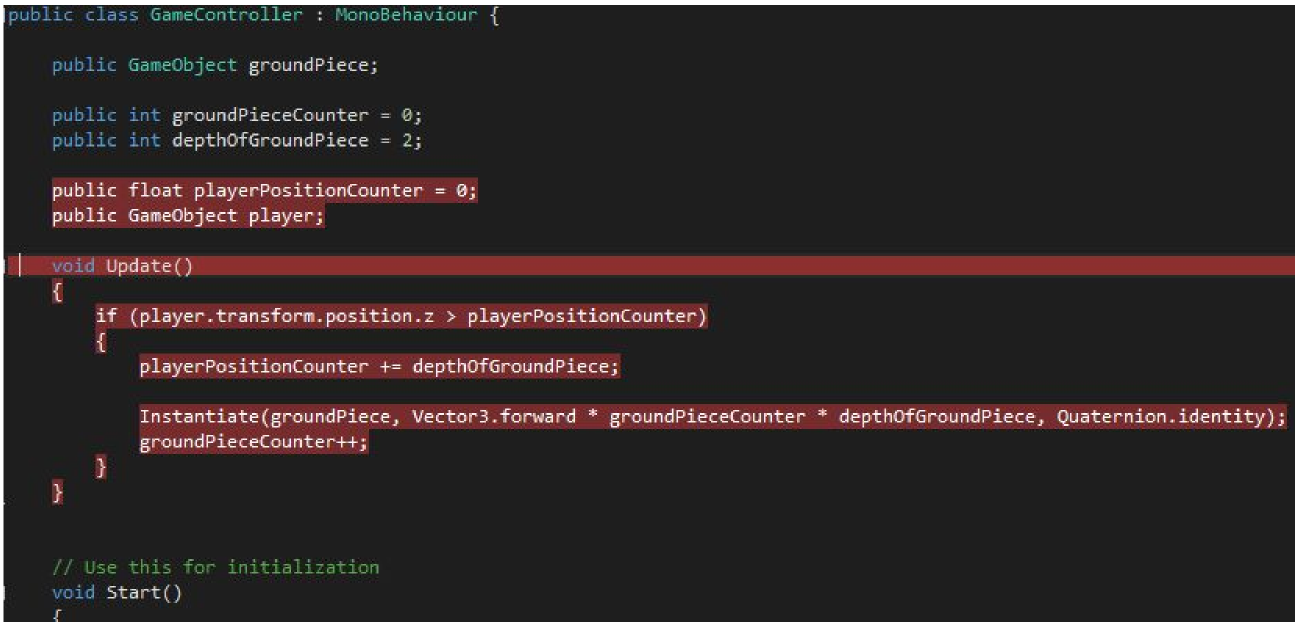


Player (Script)

Script	Player
Main Camera	Main Camera
Running Speed	6

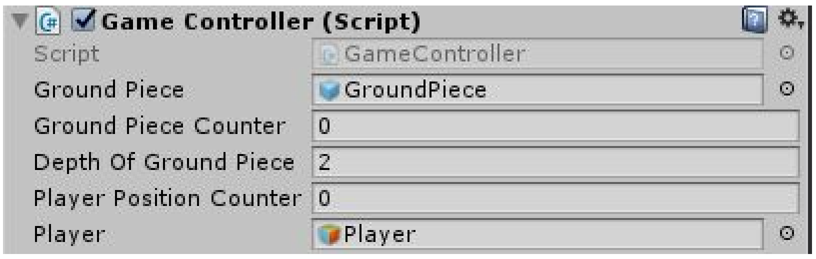
Now if you run the game the camera will follow our player.
But we run out of road pretty quickly.

Let’s keep coding.
In the **GameController** script



```
public class GameController : MonoBehaviour {

    public GameObject groundPiece;

    public int groundPieceCounter = 0;
    public int depthOfGroundPiece = 2;

    public float playerPositionCounter = 0;
    public GameObject player;

    void Update()
    {
        if (player.transform.position.z > playerPositionCounter)
        {
            playerPositionCounter += depthOfGroundPiece;

            Instantiate(groundPiece, Vector3.forward * groundPieceCounter * depthOfGroundPiece, Quaternion.identity);
            groundPieceCounter++;
        }
    }


    // Use this for initialization
    void Start()
    {
```

Make sure you reference the player in the inspector



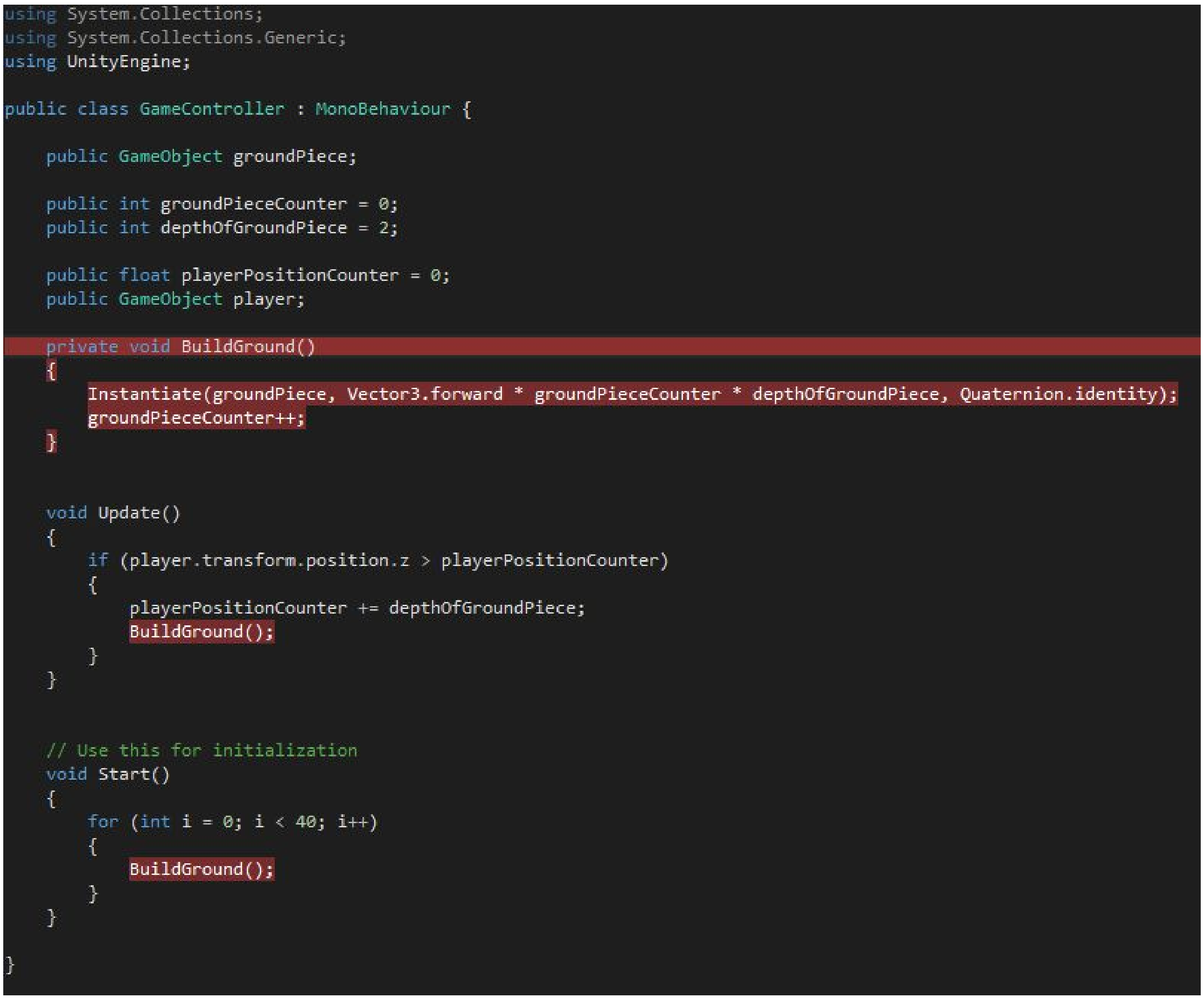
Game Controller (Script)

Script	GameController
Ground Piece	GroundPiece
Ground Piece Counter	0
Depth Of Ground Piece	2
Player Position Counter	0
Player	Player

What is happening here is that every frame the script will check where the player is and if they are close the end it will spawn in another ground piece.

Run the game and see for yourself.

But notice we are repeating code, we are instantiating code in the Start AND the Update functions.
And we aim to never repeat code remember?
So let’s move a few things around to make it easier to maintain.

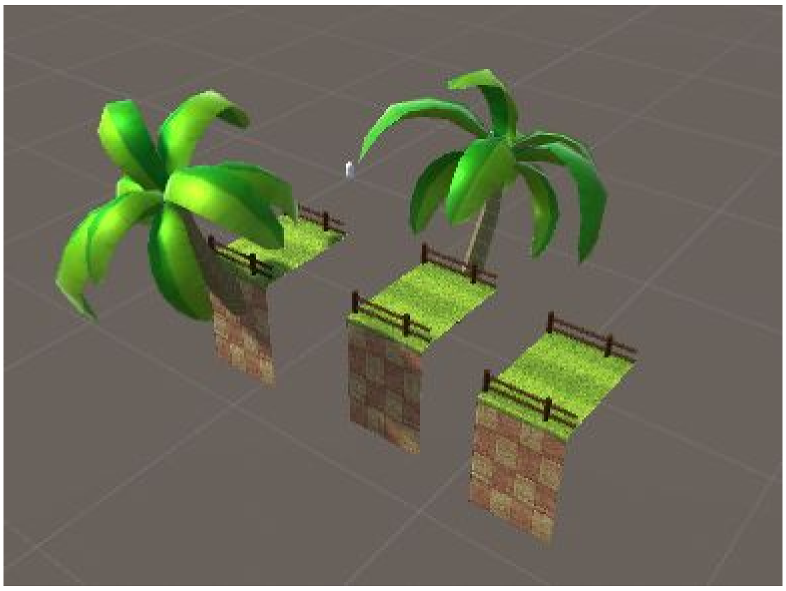


```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GameController : MonoBehaviour {

    public GameObject groundPiece;

    public int groundPieceCounter = 0;
    public int depthOfGroundPiece = 2;

    public float playerPositionCounter = 0;
    public GameObject player;

    private void BuildGround()
    {
        Instantiate(groundPiece, Vector3.forward * groundPieceCounter * depthOfGroundPiece, Quaternion.identity);
        groundPieceCounter++;
    }


    void Update()
    {
        if (player.transform.position.z > playerPositionCounter)
        {
            playerPositionCounter += depthOfGroundPiece;
            BuildGround();
        }
    }


    // Use this for initialization
    void Start()
    {
        for (int i = 0; i < 40; i++)
        {
            BuildGround();
        }
    }

}
```

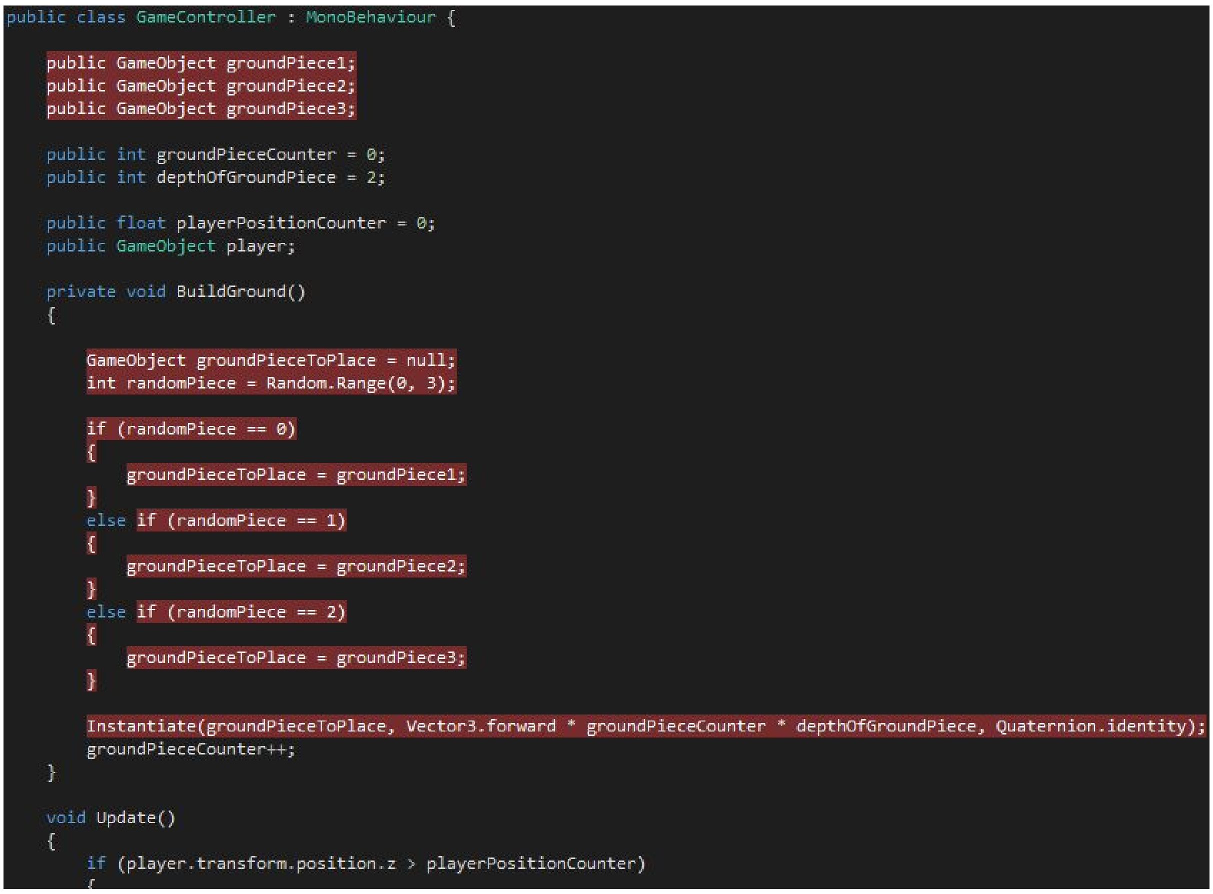
We have added a new function called **BuildGround** that has all the instantiation code, and both **Update** and **Start** call this function when they need it.

We need to make our track a little more interesting, so let’s mix up the ground pieces by adding a few more

Copy the existing ground prefab and make each slightly different (Add the palm tree provided. One on the left, One on the right) and save them as new prefabs

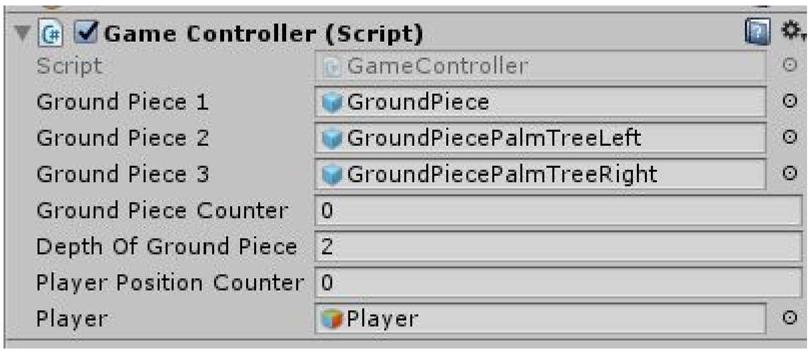


Update the **Gamecontroller** script and update the **BuildGround** function to allow it to pick randomly from the three prefabs.



```
public class GameController : MonoBehaviour {

    public GameObject groundPiece1;
    public GameObject groundPiece2;
    public GameObject groundPiece3;

    public int groundPieceCounter = 0;
    public int depthOfGroundPiece = 2;

    public float playerPositionCounter = 0;
    public GameObject player;

    private void BuildGround()
    {

        GameObject groundPieceToPlace = null;
        int randomPiece = Random.Range(0, 3);

        if (randomPiece == 0)
        {
            groundPieceToPlace = groundPiece1;
        }
        else if (randomPiece == 1)
        {
            groundPieceToPlace = groundPiece2;
        }
        else if (randomPiece == 2)
        {
            groundPieceToPlace = groundPiece3;
        }

        Instantiate(groundPieceToPlace, Vector3.forward * groundPieceCounter * depthOfGroundPiece, Quaternion.identity);
        groundPieceCounter++;
    }

    void Update()
    {
        if (player.transform.position.z > playerPositionCounter)
```

Make sure you update the Inspector



Game Controller (Script)

Script	GameController
Ground Piece 1	GroundPiece
Ground Piece 2	GroundPiecePalmTreeLeft
Ground Piece 3	GroundPiecePalmTreeRight
Ground Piece Counter	0
Depth Of Ground Piece	2
Player Position Counter	0
Player	Player

Results

Now you have a runner that can run FOREVER… well kinda.

Add Player Movement

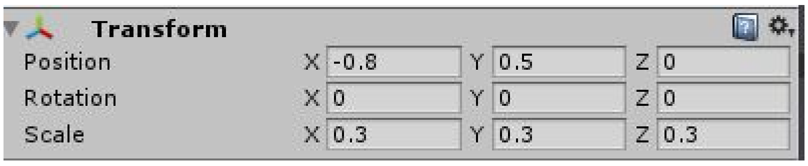
Objective

Add player input to move the player

Setup

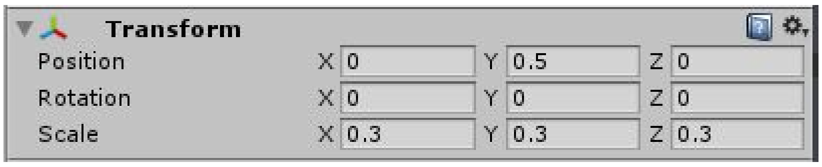
Continued from previous lesson

- Make and empty GameObject called **'Player'** at 0 - 0 - 0
- Move the player capsule to be a child of this object.
- Rename the old Player object (the capsule) to **PlayerMesh**
- Move the player script from the **PlayerMesh** to the **Player** empty game object
- Make 3 spheres
    - Remove their colliders
    - Change their scale to 0.3,0.3,0.3
    - Put that at positions:
        - -0.8f - 0.5f - 0
        - 0 - 0.5f - 0
        - 0.8f - 0.5f - 0
    - Make them children of the player empty game object.
    - Name them **PlayerPostion1** , **PlayerPostion2** , and **PlayerPostion3**



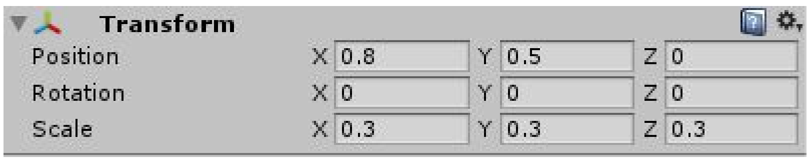
Transform

Position	X	-0.8	Y	0.5	Z	0
Rotation	X	0	Y	0	Z	0
Scale	X	0.3	Y	0.3	Z	0.3



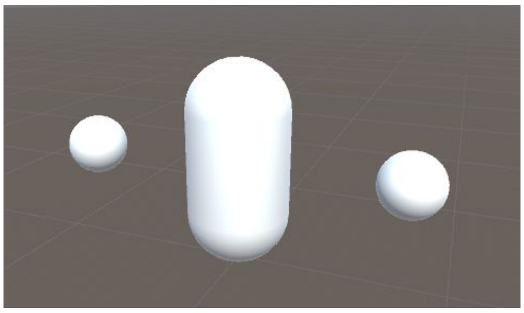
Transform

Position	X	0	Y	0.5	Z	0
Rotation	X	0	Y	0	Z	0
Scale	X	0.3	Y	0.3	Z	0.3



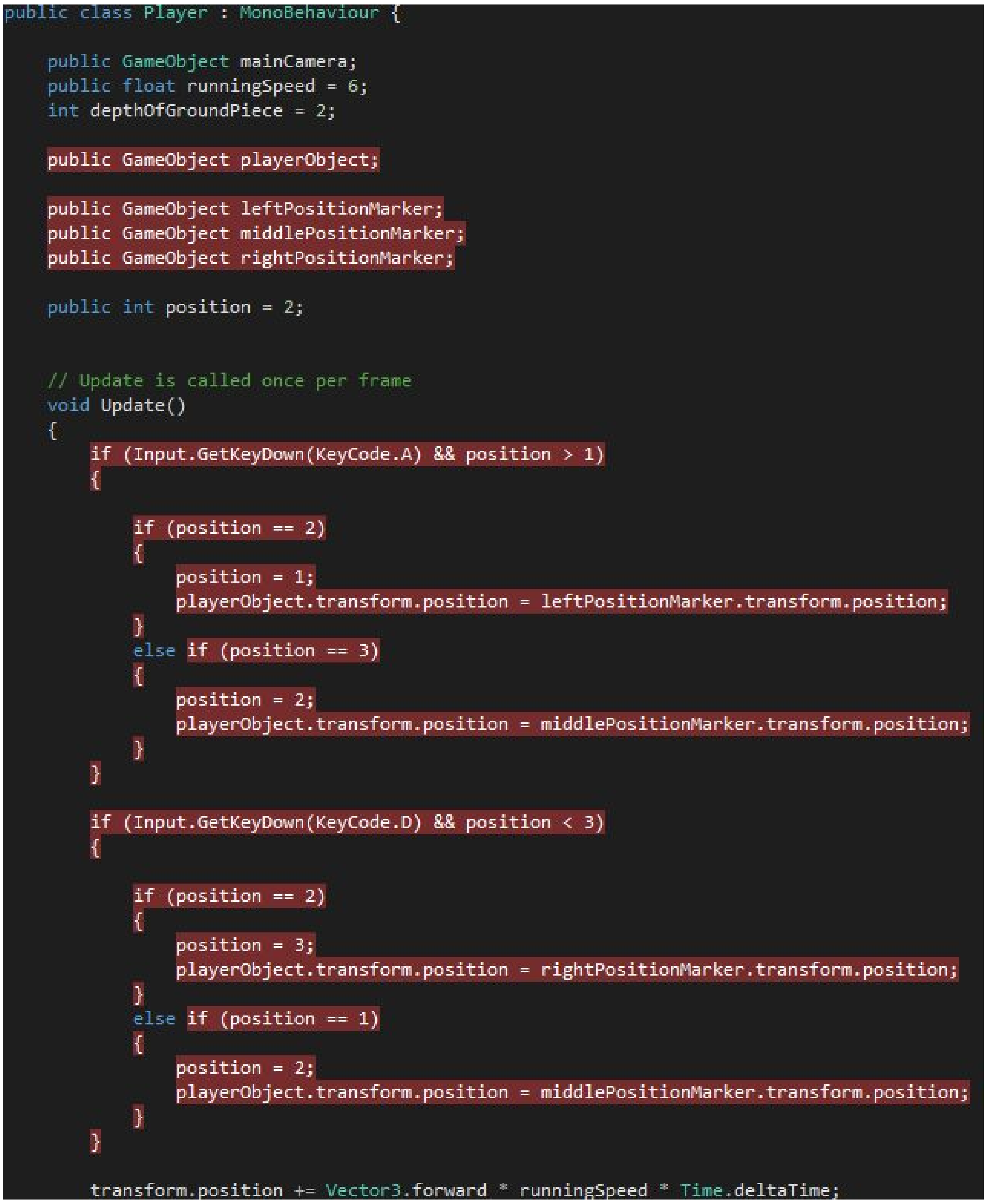
Transform

Position	X	0.8	Y	0.5	Z	0
Rotation	X	0	Y	0	Z	0
Scale	X	0.3	Y	0.3	Z	0.3



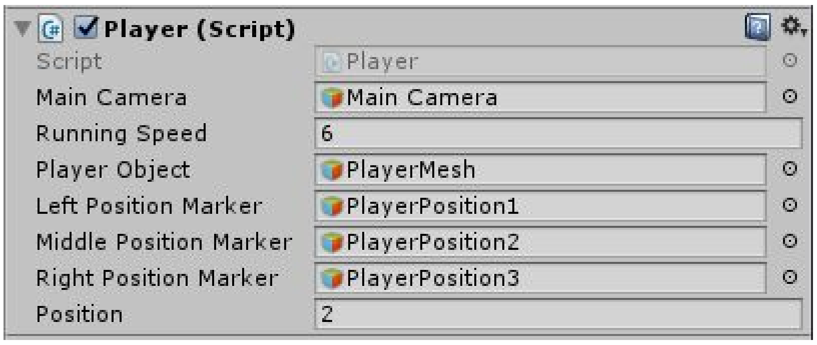
Lesson

Let’s work on the Player script to make them move



```
public class Player : MonoBehaviour {

    public GameObject mainCamera;
    public float runningSpeed = 6;
    int depthOfGroundPiece = 2;

    public GameObject playerObject;

    public GameObject leftPositionMarker;
    public GameObject middlePositionMarker;
    public GameObject rightPositionMarker;

    public int position = 2;


    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.A) && position > 1)
        {

            if (position == 2)
            {
                position = 1;
                playerObject.transform.position = leftPositionMarker.transform.position;
            }
            else if (position == 3)
            {
                position = 2;
                playerObject.transform.position = middlePositionMarker.transform.position;
            }
        }

        if (Input.GetKeyDown(KeyCode.D) && position < 3)
        {

            if (position == 2)
            {
                position = 3;
                playerObject.transform.position = rightPositionMarker.transform.position;
            }
            else if (position == 1)
            {
                position = 2;
                playerObject.transform.position = middlePositionMarker.transform.position;
            }
        }

        transform.position += Vector3.forward * runningSpeed * Time.deltaTime;
```

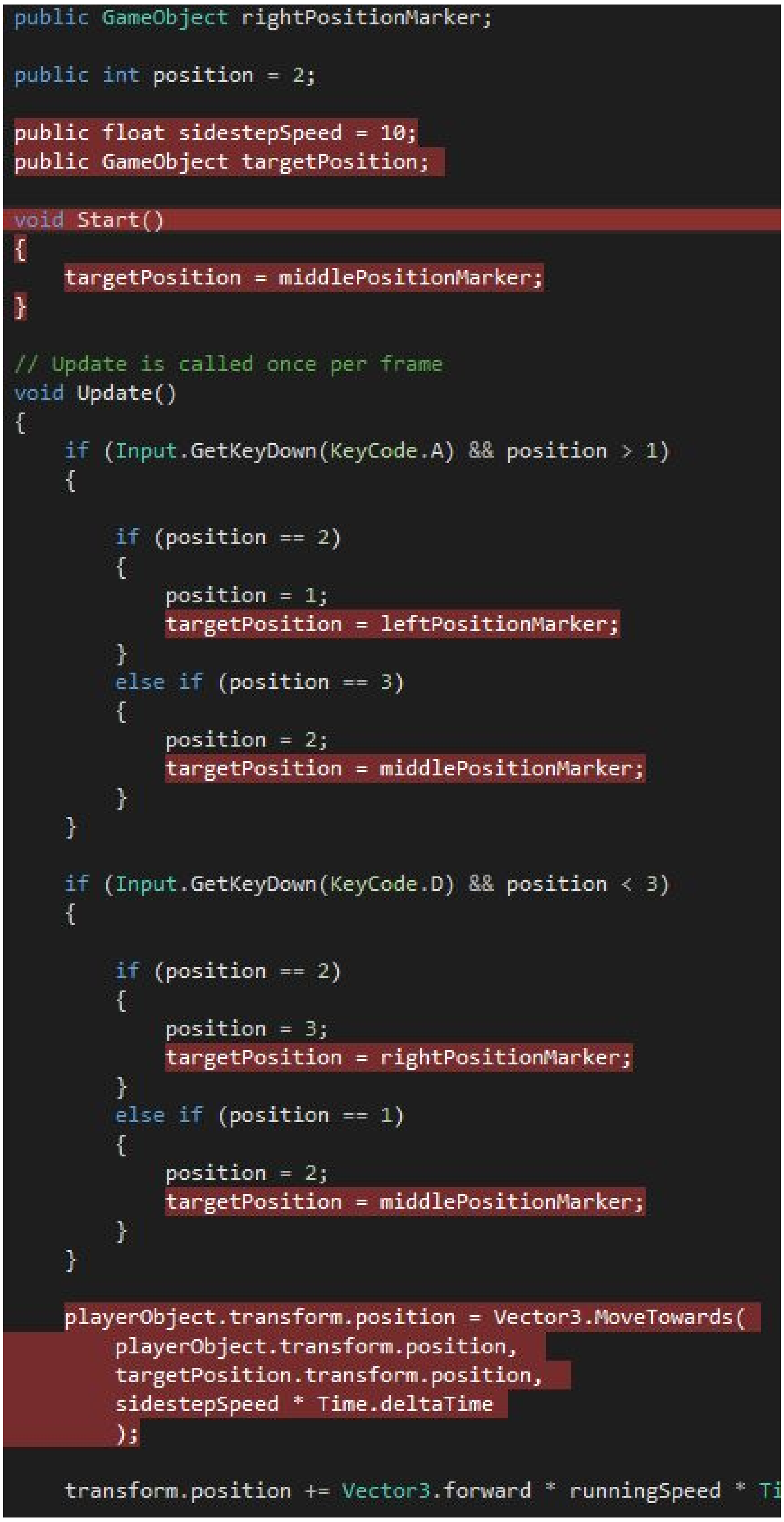
Make sure you update the inspector


Player (Script)

Script	Player
Main Camera	Main Camera
Running Speed	6
Player Object	PlayerMesh
Left Position Marker	PlayerPosition1
Middle Position Marker	PlayerPosition2
Right Position Marker	PlayerPosition3
Position	2

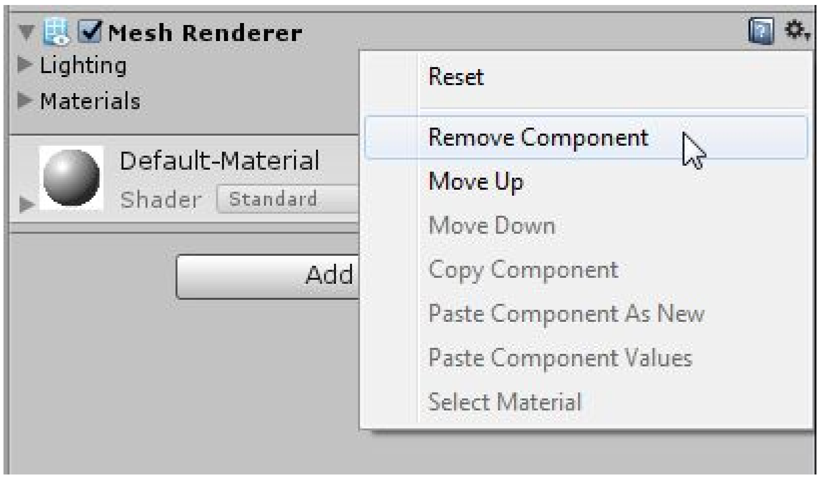
Now when you play the player should jump between the three points.

Lets add a few more lines to make them move smoothly.



```
public GameObject rightPositionMarker;

public int position = 2;

public float sidestepSpeed = 10;
public GameObject targetPosition;

void Start()
{
    targetPosition = middlePositionMarker;
}

// Update is called once per frame
void Update()
{
    if (Input.GetKeyDown(KeyCode.A) && position > 1)
    {

        if (position == 2)
        {
            position = 1;
            targetPosition = leftPositionMarker;
        }
        else if (position == 3)
        {
            position = 2;
            targetPosition = middlePositionMarker;
        }
    }

    if (Input.GetKeyDown(KeyCode.D) && position < 3)
    {

        if (position == 2)
        {
            position = 3;
            targetPosition = rightPositionMarker;
        }
        else if (position == 1)
        {
            position = 2;
            targetPosition = middlePositionMarker;
        }
    }

    playerObject.transform.position = Vector3.MoveTowards(
        playerObject.transform.position,
        targetPosition.transform.position,
        sidestepSpeed * Time.deltaTime
        );

    transform.position += Vector3.forward * runningSpeed * Ti
```

Now we need to hide the three spheres. Select them and and remove the mesh render



Results

The player can make the player move smoothly between the tree points while running down the track

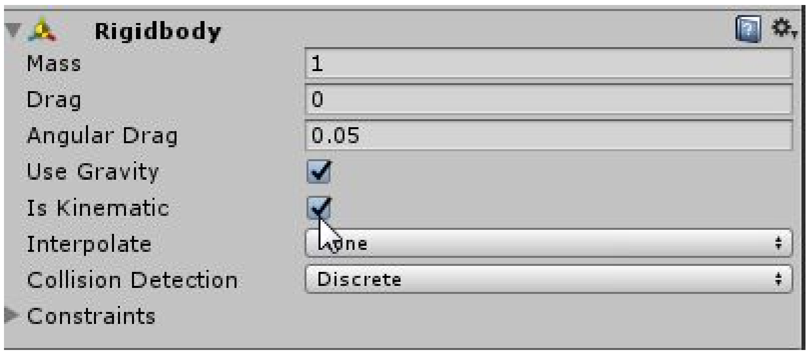
Add Obstacles

Objective

Make things for the player to avoid

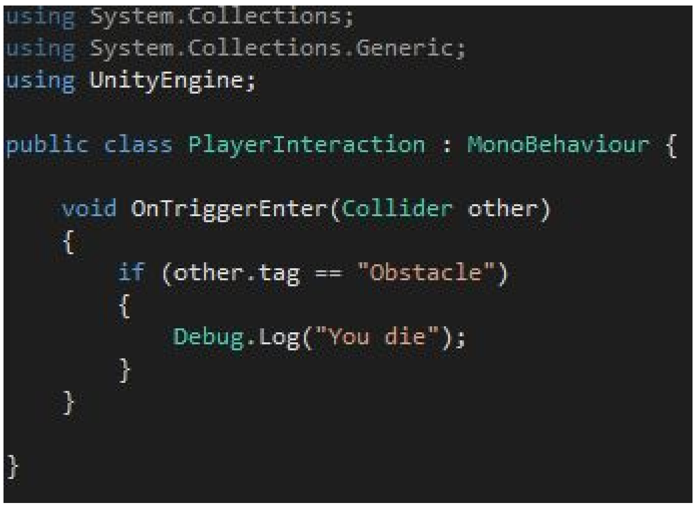
Setup

Continued from previous lesson

- Make new prefabs with a ground pieces with a Hurdles in each of the players lanes that are provided in the package.
    - Place a box collider on the hurdle
    - Place a RigidBody on the hurdle
    - Mark the RigidBody as **Is Kinematic**
    - 
      Rigidbody
      Mass 1
      Drag 0
      Angular Drag 0.05
      Use Gravity ✓
      Is Kinematic ✓
      Interpolate None
      Collision Detection Discrete
      Constraints
    - Tag the hurdle with a new tag called ‘Obstacle’
    - Save the new prefabs
- On the **PlayerMesh** Capsule, change the collider into a trigger
- Create a new script called **PlayerInteraction** and place it on the player capsule

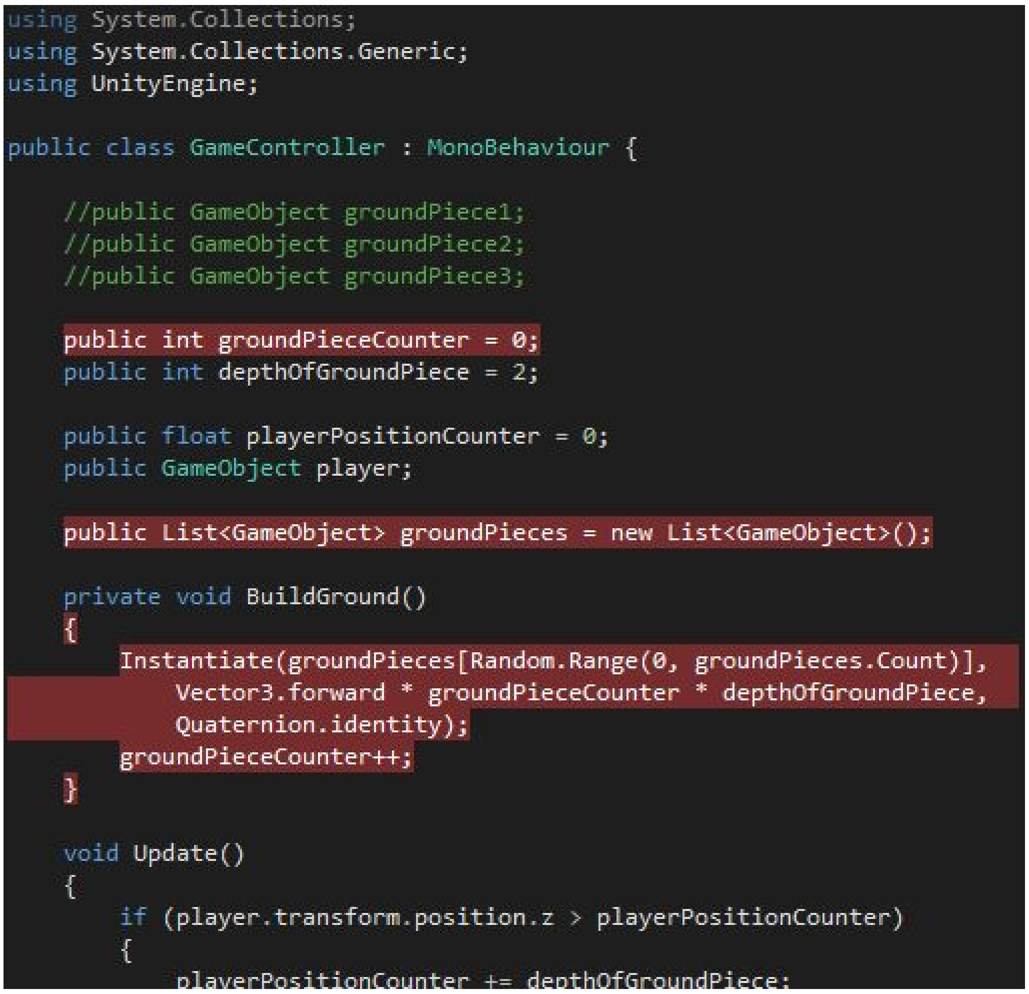
Lesson

Open up the **PlayerInteraction** script and add this function



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

public class PlayerInteraction : MonoBehaviour {

    void OnTriggerEnter(Collider other)
    {
        if (other.tag == "Obstacle")
        {
            Debug.Log("You die");
        }
    }

}
```

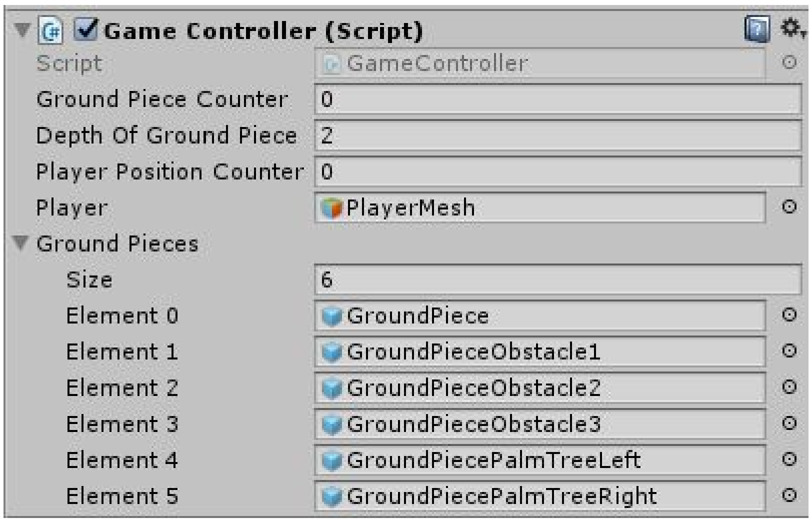
Open up the **GameController** and replace the BuildGround function with this



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

public class GameController : MonoBehaviour {

    //public GameObject groundPiece1;
    //public GameObject groundPiece2;
    //public GameObject groundPiece3;

    public int groundPieceCounter = 0;
    public int depthOfGroundPiece = 2;

    public float playerPositionCounter = 0;
    public GameObject player;

    public List<GameObject> groundPieces = new List<GameObject>();

    private void BuildGround()
    {
        Instantiate(groundPieces[Random.Range(0, groundPieces.Count)],
            Vector3.forward * groundPieceCounter * depthOfGroundPiece,
            Quaternion.identity);
        groundPieceCounter++;
    }

    void Update()
    {
        if (player.transform.position.z > playerPositionCounter)
        {
            playerPositionCounter += depthOfGroundPiece;
```

*Note: We are rewriting the **BuildGround** function and Commenting out the three **groundPiece** GameObjects*

Now you will have public list to place all your ground pieces, you can make as many as you like and store them in the list.



Game Controller (Script)

Script	GameController
Ground Piece Counter	0
Depth Of Ground Piece	2
Player Position Counter	0
Player	PlayerMesh
Ground Pieces	
Size	6
Element 0	GroundPiece
Element 1	GroundPieceObstacle1
Element 2	GroundPieceObstacle2
Element 3	GroundPieceObstacle3
Element 4	GroundPiecePalmTreeLeft
Element 5	GroundPiecePalmTreeRight

Results

If you run the game you should have it pull from your random list, and get a message everytime you hit a barrier.

Pickups

Objective

Make something for the player to pick up.

Setup

Continued from previous lesson

Lesson

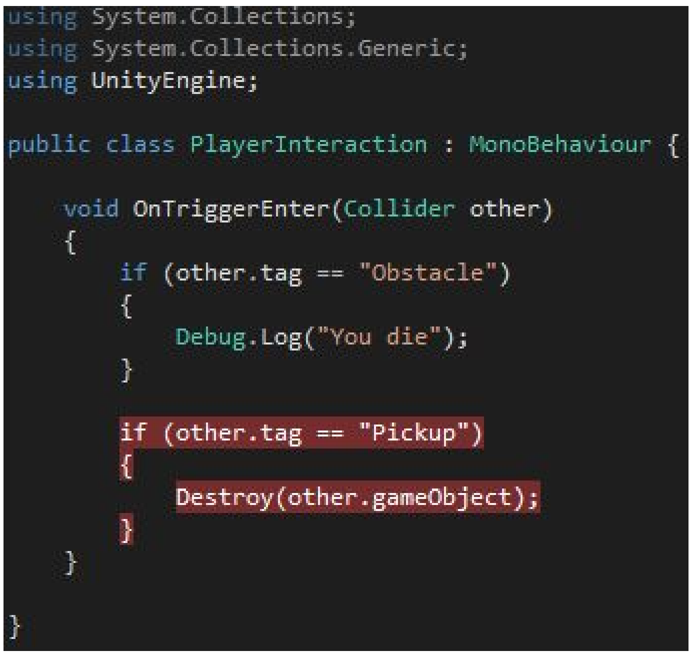
Add in the **Pickup** prefab (in your prefabs folder) and add them to some of your ground pieces like like you did with the hurdles

Make sure you

- Add a box collider
- Add a Rigidbody (kinimatic)
- Tag the pickup with a **Pickup** tag
- Update the prefab

Open some of your ground pieces and add the new pickups just like you added the hurdles

Update the **Playerinteraction** script to allow you to pick up the Pickups



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

public class PlayerInteraction : MonoBehaviour {

    void OnTriggerEnter(Collider other)
    {
        if (other.tag == "Obstacle")
        {
            Debug.Log("You die");
        }

        if (other.tag == "Pickup")
        {
            Destroy(other.gameObject);
        }
    }

}
```

Results

Now your game should run with you being able to pick up the Pickups and avoid the hurdles

Chunking

Objective

Use of chunked random generation

Setup
Continued from previous lesson

Lesson
You might have noticed that while the game is random, it's not fun.
One lesson you will learn is that making a level with pure randomness is really quite boring.

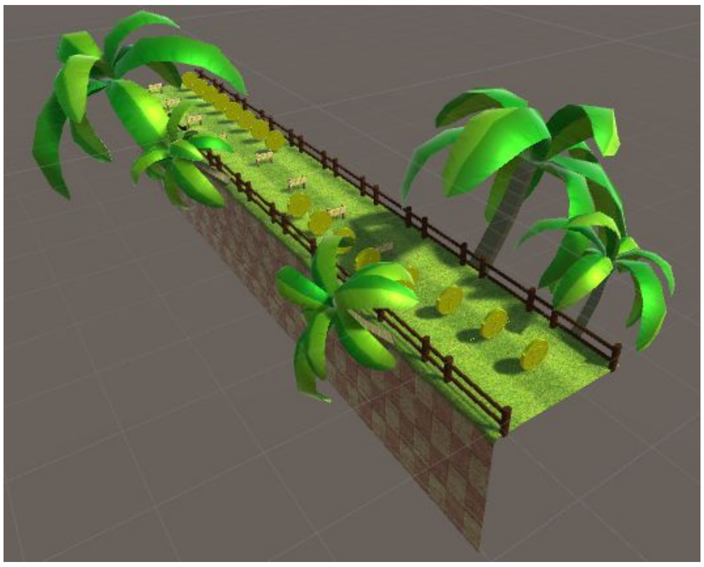
So how do other games make their endless runners more interesting?

They use a level design tool called chunking.

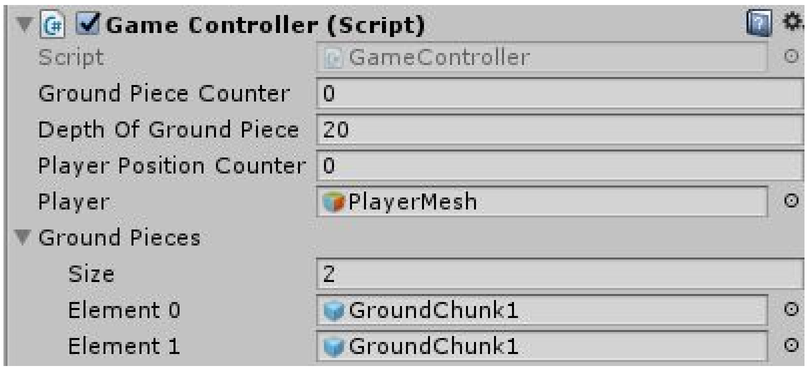
Basically instead of randomly generating every few steps, they design little fun areas called chunks and group them together.
This way the player is always in a fun tested area.

So let's update our game to be made with chunks

Make an empty game object and place 10 ground pieces in a line, then place the coins and hurdles on it in an way that looks good and is fun



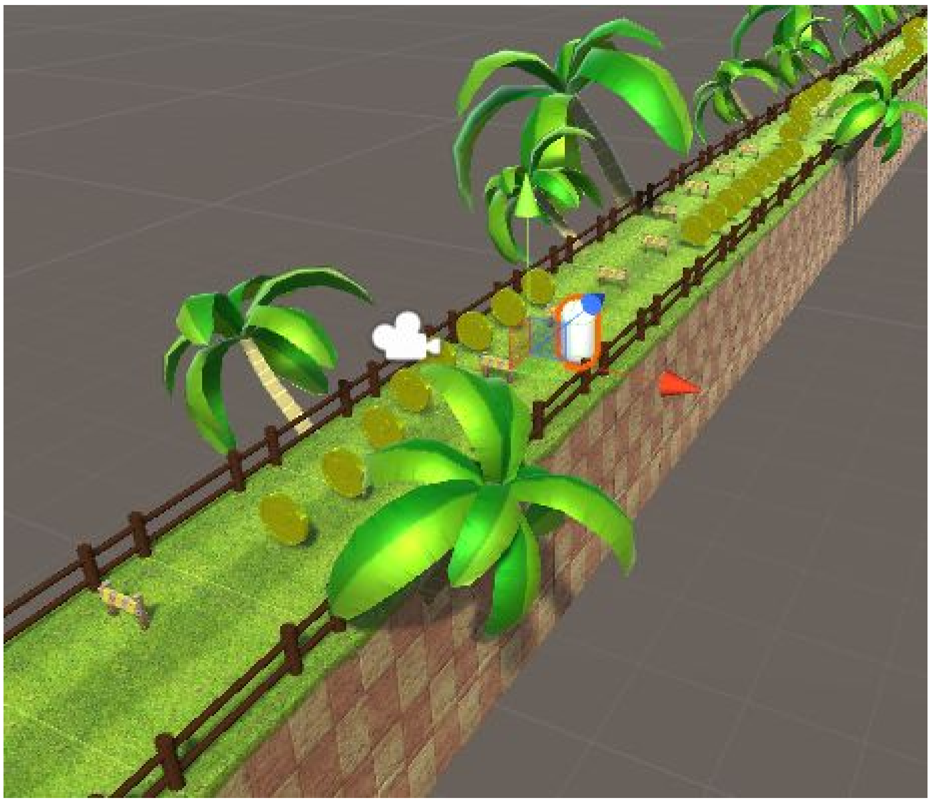
Save this as a ground chunk, Make a couple more and place them in the **GameController** script



Game Controller (Script)

Script	GameController
Ground Piece Counter	0
Depth Of Ground Piece	20
Player Position Counter	0
Player	PlayerMesh
Ground Pieces	
Size	2
Element 0	GroundChunk1
Element 1	GroundChunk1

And when you run the game you should get a much better experience



Results
Now the game should have a more interesting flow and look much better.

Can you think of any other types of games that do this?