For this month's we will continue the game we started in the first issue. We will go through all of the new and edited scripts, explaining what it does and how. Experiment with these scripts and see if you can change them. You can see the game creation itself at: https://youtu.be/Eul8-e-SC6U

Before we start, I want to talk about the Order in layer numbers. Each game Object has its Order layer number. The ones with a higher number will be shown on top of those with a lower number. Make sure you use it otherwise you might not see game objects when they are in fact behind another.

Now let's start with the PipeMoving script.

As you did issue 1 you should understand parts of the script we create now. I will only explain the new or changed things.

First, we create a bool variable and call it movingRightLeft. A bool variable uses only true or false statements. We set it to true. Next is the float variable for the Speed this one should be familiar.

```
public bool movingRightLeft = true;
public float Speed;
```

In the Update function we use the bool variable to determine if the pipe should move to the right (is true) or to the left (false). When the distance in that direction passes the given border number (Coordinates can be taken from Inspector's view) it changes the bool variable into the opposite statement. Since it is in the update function this will keep happening, so you have a left right movement.

```
void Update()
{
if (movingRightLeft == true)
{
transform.Translate(Vector2.right * Speed * Time.deltaTime);
if(transform.position.x > 5.5f)
{
movingRightLeft = false;
}
}
if (movingRightLeft == false)
{
transform.Translate(Vector2.left * Speed * Time.deltaTime);
if (transform.position.x < -5.5f)
{
movingRightLeft = true;
}
}
}
}
```

What you learned so far:

- Bool variable
- True / false
- Vector2.right
- Vector2.left
- Order in layer number

The Enemymove script.

Place an enemy game object and give it a Box Collider 2D with is trigger selected. Give it also a name tag Enemy. Next you give it this script.

The familiar float variable for Speed.

```
public float Speed;
```

In the Update function we make sure that the game Object will move up until its position on the Y-axis is bigger than 5.5f (Coordinates can be taken from the inspector's view). When the position is over the maximum (Bigger then 5.5f) it will be removed.

```
void Update()
{
transform.Translate(Vector2.up*Speed*Time.deltaTime);
if(transform.position.y > 5.5f)
{
Destroy(this.gameObject);
}
}
}
```

Drag this game Object to the prefab folder so it becomes a prefab. Delete the one in the level (Scene). The enemymovement should be clear to you.

The MonsterSpawn script.

The goal is to make monsters rise up from the ground to the top so the player will need to avoid them. Create a new empty game object. Give it the name MonsterSpawner and use any icon you like for it. Give it this script.

First a game Object is defined Monster. This makes sure that you can drag the prefab monster into the inspector's view. Next there it is again the float variable but this time for a countdown timer that we set to start on 5.

```
public GameObject Monster;
public float CountDown = 5;
```

We create a new function that will create a monster using the prefab monster and place it at the X-axis on a random place.

Instantiate = create and then use the name of the prefab to place it at the position 5.8f on the x-axis but added with a random number between 2 and 10. The coordinates on the y-axis is 6f and on the Z stays 0. Although we create a 2D game it's built in a 3D environment. Click the 2D button in the scene window to see it if you don't believe me. Quaternion.identity means use the rotation of the game object as it is so no need of changing it. Coordinates are taken from the inspector's view as usual. See how many things repeat and you get used to it?

```
public void Spawn()
{
Instantiate(Monster,newVector3(5.8f+Random.Range(2,10),6f,0),Quaternion.identity);
}
```

In the update function we make the countDown variable go down 1 second at the time -=Time.deltaTime; When the countdown is smaller than or equal to 0 it will start the spawn function. The countdown is set back to 5 so the countdown can restart.

```
void Update()
{
CountDown -= Time.deltaTime;
if(CountDown <= 0)
{
Spawn();
CountDown = 5;
}
}
}
```

What you learned so far:

- Instantiate
- Quaternion.identity
- Random.Range
- Countdown variable
- If

Changing the MarioFly script.

Give the game object a Rigidbody2D and a box collider 2D. Give the game object the Name tag Player. Edit the existing script into this one.

We create a bool variable that will determine the direction the player is facing. (Remember the bool?) Next, we create a Vector3 and call it characterScale. A float variable characterScaleX is also created.

```
public bool FaceRight = true;
Vector3 characterScale;
float characterScaleX;
```

In the Start function we set that our float variable is equal to the game objects localScale. It uses our made Vector3. This will be used to flip the game object. There are more ways to do this but this way a child object will flip also and won't end up misplaced or on the wrong side of the game Object.

```
characterScale = transform.localScale;
characterScaleX = characterScale.x;
```

You know how collision works. Triggers work the same but have a different script. When a trigger is entered (The collision box that is selected with is trigger) it will check if it collided with a game object with the name tag Enemy. If so than it will set the score to 0 and reloads the scene. The reloaded part should be familiar to you as we handled it in the previous issue.

```
public void OnTriggerEnter2D(Collider2D other)
{
if(other.gameObject.tag == ("Enemy"))
{
Scores.scorePoints = 0;
SceneManager.LoadScene("SampleScene");
Destroy(this.gameObject);
}
}
```

Final is the Update function part. When the bool variable is set to true it will not flip the character. When it is set to false it will flip the character into the opposite direction. Using the key press with the left and right arrow makes the game object move left or right. The movement part you should understand. The bool Faceright is set by using this key.

```
if(FaceRight == true)
{
characterScale.x = characterScaleX;
}
if (FaceRight == false)
{
characterScale.x = -characterScaleX;
}
if (Input.GetKey(KeyCode.RightArrow))
```

```
{
FaceRight =true;
transform.Translate(Vector2.right * FlySpeed * Time.deltaTime);
}
if (Input.GetKey(KeyCode.LeftArrow))
{
FaceRight = false;
transform.Translate(Vector2.left * FlySpeed * Time.deltaTime);
}
```

By adding this line in the Update function, the localScale is used. Without this line the flipping of the game object won't work.

transform.localScale = characterScale;

What you learned so far:

- OnTriggerEnter2D
- Collider2D
- LocalScale

With these first 2 issues you have a lot of scripts and learned how to use it. You should be able to do some movement coding and create little games. In the future we will do more small projects so we can handle different kind of games. Have fun!

How To ?

Teleporters are not only used in star trek. In games those handy portals transport us anywhere in the digital gameworld. You can see the creation here:

https://youtu.be/wHP49W4RXtY

Let's have a look at the script which will have some new parts.

The player in this example has a very basic script. It moves left and right and that's all. No explenation of this script is needed as you should understand it by now.

Make sure your player has the name tag Player and a Box collider2D and Rigidbody2D are added to it.

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

public class Playermove : MonoBehaviour
{
public float RickSpeed;

void Update()
{
if(Input.GetKey(KeyCode.LeftArrow))
{
transform.Translate(Vector2.left * RickSpeed * Time.deltaTime);
}
if (Input.GetKey(KeyCode.RightArrow))
{
transform.Translate(Vector2.right * RickSpeed * Time.deltaTime);
}
}
}
```

Now let's go through the teleport script.

A int variable is created called code. This way we can easily tell each teleporter where it should move the player to. Number 1 should move to the other number 1 and so one.
A timer variable is created so we can disable the teleporter for as long as we want when we use it. Since I made round portals it would be nice to see them rotate, a float variable for the rotation speed is made so each portal can rotate on its own speed. I added a public soundfile for when the portal is used.

```
public int code;
float disableTimer = 0;
public float RotationSpeed;
public AudioClip Portalsound;
```

In the update function we make the rotation happen at the given speed. When the disable timer is bigger than 0 it will count down to 0.

```
void Update()
{
transform.Rotate(Vector3.back*RotationSpeed*Time.deltaTime);
if (disableTimer > 0)
disableTimer -= Time.deltaTime;
}
```

Make sure the portal has a BoxCollider2D with is trigger selected. On a trigger Exit the disableTimer will be set to 0.5f;

```
void OnTriggerExit2D(Collider2D coll)
{
disableTimer = 0.5f;
}
```

On a trigger stay it will check if the trigger is a game object with the tag Player. When the disable timer is smaller than zero it will check all objects with the Teleportpad script for a corrosponding number (Code) so it knows where to teleport the player to. When a matching teleport is found and the spacebar is pressed it will move the player (the coll collider) to that spot using the sound we defined earlier. I moved the position of the gameobject a little up so it looks like it pops out of the portal.The disableTimer is set to 0.5f.

```
void OnTriggerStay2D(Collider2D coll)
{
if (coll.gameObject.tag == ("Player") && disableTimer <= 0 )
{
foreach (Teleportpad tp in FindObjectsOfType<Teleportpad>())
{
if (tp.code == code && tp != this && Input.GetKey(KeyCode.Space))
{
AudioSource.PlayClipAtPoint(Portalsound, new Vector3(0, 0, -10));
tp.disableTimer = 0.5f;
Vector3 position = tp.gameObject.transform.position;
coll.gameObject.transform.position = position;
position.y += 1;
}
}
}
}
}
```

You can create as many portals as needed and give effects and sounds to it that fit your game. Enough material to practice with before becoming a master game maker.

All of the assets used in this issue can be downloaded here : https://rp-interactive.nl/magazine.html

All scripts used in issue 02 are here for you.

PipeMoving

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

public class PipeMoving : MonoBehaviour
{

//////////CoinDrop Game Script by René Pol RP-Interactive.nl© 08-04-2021/////////
public bool movingRightLeft = true;
public float Speed;

void Update()
{
if (movingRightLeft == true)
{
transform.Translate(Vector2.right * Speed * Time.deltaTime);
if(transform.position.x > 5.5f)
{
movingRightLeft = false;
}
}
if (movingRightLeft == false)
{
transform.Translate(Vector2.left * Speed * Time.deltaTime);
if (transform.position.x < -5.5f)
{
movingRightLeft = true;
}
}
}
}
```

EnemyMove

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

public class Enemymove : MonoBehaviour
{
//////////CoinDrop Game Script by René Pol RP-Interactive.nl© 08-04-2021/////////
public float Speed;
void Update()
{
transform.Translate(Vector2.up*Speed*Time.deltaTime);
if(transform.position.y > 5.5f)
{
Destroy(this.gameObject);
}
}
}
```

MonsterSpawn

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

public class MonsterSpawn : MonoBehaviour
{
//////////CoinDrop Game Script by René Pol RP-Interactive.nl© 08-04-2021/////////

public GameObject Monster;
public float CountDown = 5;

public void Spawn()
{
Instantiate(Monster,newVector3(5.8f+Random.Range(2,10),6f,0),Quaternion.identity);
}

void Update()
{
CountDown -= Time.deltaTime;
if(CountDown <= 0)
{
Spawn();
CountDown = 5;
}
}
}
```

MarioFly

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

//////////CoinDrop Game Script by René Pol RP-Interactive.nl© 08-04-2021/////////

public class MarioFly : MonoBehaviour
{
public float FlySpeed;
public float Coinnr = 5;
public GameObject Coins,Restart;
public Transform CoinDropper;
public bool FaceRight = true;
Vector3 characterScale;
float characterScaleX;
```

```
void Start()
{
characterScale = transform.localScale;
characterScaleX = characterScale.x;
Restart.SetActive(false);
}

public void OnTriggerEnter2D(Collider2D other)
{
if(other.gameObject.tag == ("Enemy"))
{
Scores.scorePoints = 0;
SceneManager.LoadScene("SampleScene");
Destroy(this.gameObject);
}
}

void Update()
{
if(FaceRight == true)
{
characterScale.x = characterScaleX;
}
if (FaceRight == false)
{
characterScale.x = -characterScaleX;
}
if (Input.GetKey(KeyCode.RightArrow))
{
FaceRight =true;
transform.Translate(Vector2.right * FlySpeed * Time.deltaTime);
}
if (Input.GetKey(KeyCode.LeftArrow))
{
FaceRight = false;
transform.Translate(Vector2.left * FlySpeed * Time.deltaTime);
}
if (Input.GetKeyDown(KeyCode.R))
{
Scores.scorePoints = 0;
SceneManager.LoadScene("SampleScene");
}
if(Coinnr == 0)
{
Restart.SetActive(true);
}
if (Input.GetKeyDown(KeyCode.Space) && Coinnr >0)
{
Instantiate(Coins, CoinDropper.position, Quaternion.identity);
Coinnr -= 1;
}
transform.localScale = characterScale;
}
}
```