Welcome to issue 6.

Would it be cool if you are able to create a full working memory game ? With this months issue you will !. I made a Marvel superhero themed game but you can use any theme you like. With these 3 scripts you can click and create a whole new game. You can see all of this created here:

https://www.youtube.com/watch?v=U238GbmTJxo&t=217s

Ready let's inspect those scripts. I will explain the parts you are new to as for familiar scripts you should know them if you read the previous issues. We will use our own designed cursor by adding a sprite as a game object and give it this script:

Own Cursor Script.

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

public class Cursormo : MonoBehaviour
{

/// Marvel Memory game René Pol RP-Interactive.nl ©-2021-///
```

At start we set the main cursor to invisible so it is hidden.

```
void Start()
{
Cursor.visible = false;
}
```

We create a point where the cursor is as a Vector2 we use the mouse position in the main camera where we set it screen to world. Our position is always set to this cursorPos position. So the game object is now moved with the mouse.

```
void Update()
{
Vector2 cursorPos = Camera.main.ScreenToWorldPoint(Input.mousePosition);
transform.position = cursorPos;
}
}
```

The memory card is a Sprite game object that holds 2 sprite. One for the back and One for the front of the card. Each new made card will be stored as a prefab. The cards have a box collider 2D. The cards hold al this script:

Memory Card Script.

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

public class MemoryCard : MonoBehaviour
{
/// Marvel Memory game René Pol RP-Interactive.nl ©-2021-///
```

A int variable for the cardValue is created and a bool variable for selecting the card. 2 Sprites are defined to use to show the back and the front and an audio file is defined to use when a card flips. Another bool variable is created to see if a card can be flipped or not.

```
public int cardValue;
public bool IsSelected;
public Sprite Back, Front;
public AudioClip Flip;
public bool abletoflip = false;
```

A reference is made to another script called CardChecker (Next script explained)

```
private CardChecker CC;
```

In the start function the soundfile plays one time at a position near the camera. It makes sure to find the Cardchecker script and set the bool variable IsSelected to false. The total of cards in the Cardchecker script is set to 1 (if 0 it will show found all pairs message).

```
void Start()
{
AudioSource.PlayClipAtPoint(Flip, new Vector3(0, 0, -5));
CC = FindObjectOfType<CardChecker>();
IsSelected = false;
CC.totalCards += 1;
}
```

When the gameobject Is clicked on and IsSelected is equal to false en the cardvalues in the Cardchecker script are equal to 0 and able to flip is true it will play the flip sound and sets IsSelected to be true.

```
void OnMouseDown()
{
if (IsSelected == false && CC.cardvalue02 ==0 | CC.cardvalue01 ==0 && abletoflip
== true)
{
IsSelected = true;
AudioSource.PlayClipAtPoint(Flip, new Vector3(0, 0, -5));
```

If the cardvalue 01 in the CardChecker script is equal to 0 it will set this cardvalue with its own Value. Return returns the script so it won't go through the rest of the script. If the cardvalue 01 in the cardcecker script is not equal to 0 it will set cardvalue 02 in the carcheckers.

```
if (CC.cardvalue01 == 0)
{
CC.cardvalue01 = cardValue;
return;
}
if (CC.cardvalue01 != 0)
{
CC.cardvalue02 = cardValue;
return;
}
}
}
```

I made a removeCard function that only when IsSelected is true it will reset to false and subtract 1 of the total cards in the card checkerscript. It calls the ResetCards function in that same script and than the gameobject will be removed.

```
public void RemoveCard()
{
if (IsSelected == true)
{
IsSelected = false;
CC.totalCards -= 1;
}
CC.ResetCards();
Destroy(this.gameObject);
}
```

I made a FlipCard function that only when IsSelected is true it will reset to false. It calls the ResetCards function in the cardchecker script and plays the flip sound.

```
public void FlipCard()
{
if (IsSelected == true)
{
CC.ResetCards();
AudioSource.PlayClipAtPoint(Flip, new Vector3(0, 0, -5));
IsSelected = false;
}
}
```

In the update we make sure that when all 20 cards a in the game abletoflip is set to true so we can now flip 2 cards.

```
void Update()
{
if (CC.totalCards == 20)
{
abletoflip = true;
}
```

When the cardvalue 02 in the cardchecker script is bigger than 0 and and cardvalue 02 is different than cardvalue 1 and when Isselected is true only then the flipcard function is called after 0.6f seconds.

```
if (CC.cardvalue02 > 0 && CC.cardvalue02 != CC.cardvalue01 && IsSelected == true)
{
Invoke("FlipCard", 0.6f);
}
```

When the cardvalue 01 in the cardchecker script is bigger than 0 and and cardvalue 01 is equal than cardvalue 02 and when Isselected is true only then the RemoveCard function is called after 1f seconds.

```
if (CC.cardvalue01 > 0 && CC.cardvalue01 == CC.cardvalue02 && IsSelected == true )
{
Invoke("RemoveCard", 1f);
}
```

When Isselecte is true it will show and use the front sprite. When it is false it uses and shows the back sprite.

```
if (IsSelected == true)
{
this.GetComponent<SpriteRenderer>().sprite = Front;
}
if (IsSelected == false)
{
this.GetComponent<SpriteRenderer>().sprite = Back;
}
}
}
```

Create an empty game object and all it cardchecker. Give it this script :

CardChecker script.

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


public class CardChecker : MonoBehaviour
{

/// Marvel Memory game René Pol RP-Interactive.nl ©-2021-///
```

We use an array that will hold all our cards. The number of cards is 20. Now in the inspector's view we can drag the prefab cards to that place. So Mcards hold 20 cards.

```
public GameObject [] MCards = new GameObject[20];
```

two int variables are created cardvalue01 and 02 aswell as an int variable called totalCards. Remember in the memory card script a reference to this script and it's variables was made and is used during the game.

```
public int cardvalue01;
public int cardvalue02;
public int totalCards;
```

A game object is defined (The foundpairs message) aswell as an audio file and a bool variable to determine if a sound can play or not.

```
public GameObject Foundpairs;
public AudioClip Magic;
public bool soundplay;
```

A shuffle function is created using the gameObjects in the array Mcards. It uses the length of this array and randomises all gameobjects in it on their positions in the array.

```
void reshuffle(GameObject[] MCards)
{
for (int t = 0; t < MCards.Length; t++)
{
GameObject tmp = MCards[t];
int r = Random.Range(t, MCards.Length);
MCards[t] = MCards[r];
MCards[r] = tmp;
}
}
```

A setup cards function is created as IE numerator. This is done so we can use waitForSeconds between instructions. First the cards get shuffled than the totalCards is set to 1. Every 0.5f seconds a card from the array is picked and placed at it's coordinates given. The Element number is used. When all cards are placed the totalCards subtracts 1 so the number is now even to the cards in game.

```
IEnumerator SetupCards()
{
reshuffle(MCards);
totalCards = 1;
yield return new WaitForSeconds(2f);
Instantiate(MCards[0], new Vector3(-5f, 3, 0), Quaternion.identity);
yield return new WaitForSeconds(0.5f);
Instantiate(MCards[1], new Vector3(-2.5f, 3, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[2], new Vector3(0f, 3, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[3], new Vector3(2.5f, 3, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[4], new Vector3(5f, 3, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[5], new Vector3(-5f, 0, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[6], new Vector3(-2.5f, 0, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[7], new Vector3(0f, 0, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[8], new Vector3(2.5f, 0, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[9], new Vector3(5f, 0, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[10], new Vector3(-5f, -3, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[11], new Vector3(-2.5f, -3, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[12], new Vector3(0f, -3, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[13], new Vector3(2.5f, -3, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[14], new Vector3(5f, -3, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[15], new Vector3(-5f, -6, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[16], new Vector3(-2.5f, -6, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[17], new Vector3(0f, -6, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[18], new Vector3(2.5f, -6, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[19], new Vector3(5f, -6, 0), Quaternion.identity);
totalCards -= 1;
}
```

At start the cards are shuffled and the Coroutine SetupCards is activated. Soundplay Is set to false and both cardvalues are set to 0. The foundpairs message is hidden.

```
void Start()
{
reshuffle(MCards);
StartCoroutine(SetupCards());
soundplay = false;
cardvalue01 = 0;
cardvalue02 = 0;
Foundpairs.SetActive(false);
}
```

The Resetcards function set boths cardvalues back to 0. Remember this function is called from the memory card script.

```
public void ResetCards()
{
cardvalue01 = 0;
cardvalue02 = 0;
}
```

The sound will only play if soundplay is set to true after 0.5f seconds it is reset back to false.

```
IEnumerator ResetIt()
{
yield return new WaitForSeconds(0.5f);
soundplay = false;
}
```

The relayCards function is made to restart a game when all pairs have been found. It hides the foundpairs message and restarts the SetupCards coroutine.

```
IEnumerator RelayCards()
{
yield return new WaitForSeconds(2f);
Foundpairs.SetActive(false);
StartCoroutine(SetupCards());
}
```

When cardvalue01 is different than 0 and cardvalue02 different than 0 but they are equal to each other and soundplay is set to false. Only then the sound will play once and the soundplay bool variable will be reset. (`IEnumerator ResetIt`)

```
void Update()
{
if (cardvalue01 != 0 && cardvalue02 != 0)
{
if (cardvalue01 == cardvalue02 && soundplay == false)
{
AudioSource.PlayClipAtPoint(Magic, new Vector3(0, 0, -10));
soundplay = true;
StartCoroutine(ResetIt());
}
}
```

When totalCards is equal to 0 it will show the foundpair message and sets totalCards to 1. It will start the relaycards coroutine and so the cards will be shuffled and placed all over again.

```
if (totalCards == 0)
{
Foundpairs.SetActive(true);
totalCards = 1;
StartCoroutine(RelayCards());
}
}
}
```

For the game music just create an empty game object and name it game Music. Add a audio source to it and dragg the audio file to use to it. Make sure the loop box is selected so it will keep repeating after it played the file. Select the Ply on Awake box to so it will start to play as soon as the game starts.

And there you have it a full working memory game. Use your own ideas and imagination to make it your own. Change themes or the way cards disappear when found, add new music and sound effects. Perhaps use cool particle effects. The basic is here for you to use.

How To ?

Sometimes the player character leaves somekind of ghosting trail of itself when moving. This 2D effect is easy in use once you know and understand how to do it.

You can see how it all works here: https://www.youtube.com/watch?v=KDBxGtApncI

First I use the sprite(gameobject) of the character. I place it and than make an animation of it by using the animation window and make the alpha color go to zero. Once the animation is done I give it this script :

Ghosting Script.

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

public class Ghosting : MonoBehaviour
{

void Start()
{
Destroy(this.gameObject,0.5f);
}
}
```

All it does is make the ghosting gameobject disappear at the given time. You can change it to your likings so it will be removed as soon as the animation is finished.

Then I made the character game object, placed it and gave it this script :

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

public class Runner : MonoBehaviour
{
```

3 float variables are made for speed,delay trail en the seconds of delay trail. A gameobject to use is defined.

```
public float Speed;
public float GtrailDelay;
public float GtrailDelaySeconds;
public GameObject ghost;
```

In the start function we make sure the Gtrai;delaySeconds is equal to the GtrailDelay variable.

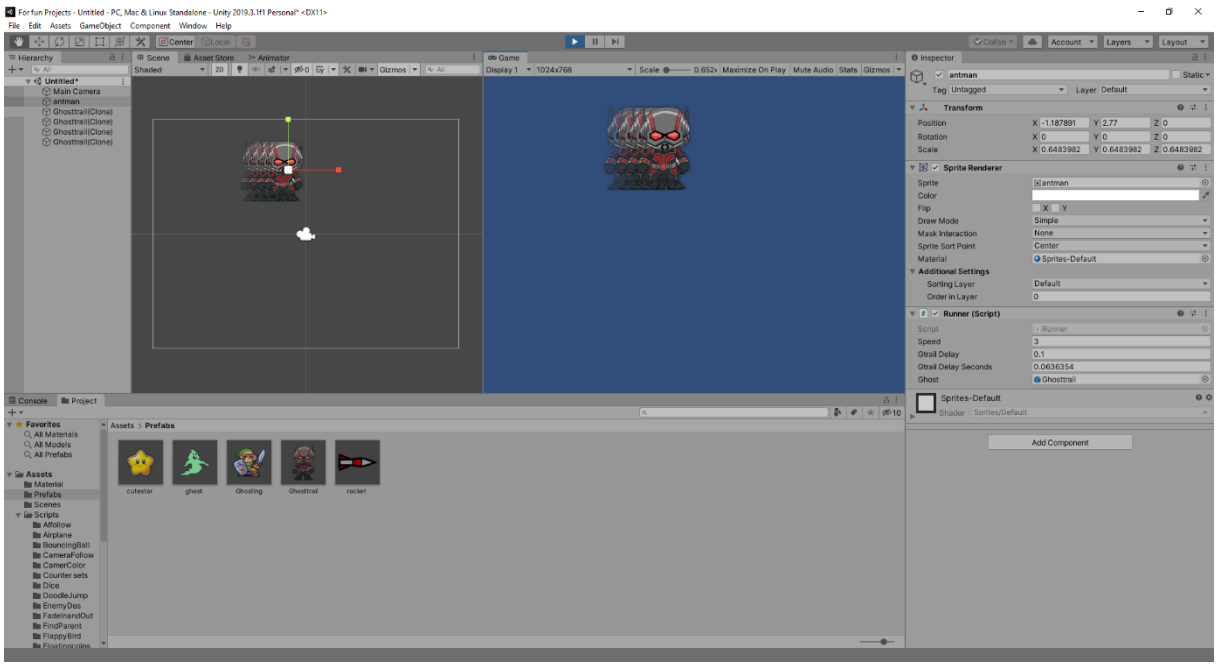
```
void Start()
{
GtrailDelaySeconds = GtrailDelay;
}
```

In the update function the gameobject moves to the right at the given speed. As long as the GtrailDelaySecons is bigger than 0 it counts down to 0. When zero is hit it instantiates our defined gameobject at the gameobjects position and rotation. The GtrailDelaySeconds is set to equal GtrailDelay again.

```
void Update()
{
transform.Translate(Vector2.right * Speed * Time.deltaTime);
if(GtrailDelaySeconds >0)
{
GtrailDelaySeconds -= Time.deltaTime;
}
else
{
Instantiate(ghost, transform.position, transform.rotation);
GtrailDelaySeconds = GtrailDelay;
}
}
}
```

Every time the ghost object is created, it will fade out and makes it all look like a longer trail. This way you can make movement trails, sword trails, engine jet trails and so on. You can set the delay and speed time in the inspector's view.

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



All scripts used in issue 06 are here for you.

Own Cursor Script.

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

public class Cursormo : MonoBehaviour
{
/// Marvel Memory game René Pol RP-Interactive.nl ©-2021-///
void Start()
{
Cursor.visible = false;
}

void Update()
{
Vector2 cursorPos = Camera.main.ScreenToWorldPoint(Input.mousePosition);
transform.position = cursorPos;
}
}
```

Memory Card Script.

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

public class MemoryCard : MonoBehaviour
{
/// Marvel Memory game René Pol RP-Interactive.nl ©-2021-///

public int cardValue;
public bool IsSelected;
public Sprite Back, Front;
public AudioClip Flip;
private CardChecker CC;
public bool abletoflip = false;

void Start()
{
AudioSource.PlayClipAtPoint(Flip, new Vector3(0, 0, -5));
CC = FindObjectOfType<CardChecker>();
IsSelected = false;
CC.totalCards += 1;
}
```

```
void OnMouseDown()
{
if (IsSelected == false && CC.cardvalue02 ==0 | CC.cardvalue01 ==0 && abletoflip
== true)
{
IsSelected = true;
AudioSource.PlayClipAtPoint(Flip, new Vector3(0, 0, -5));
if (CC.cardvalue01 == 0)
{
CC.cardvalue01 = cardValue;
return;
}
if (CC.cardvalue01 != 0)
{
CC.cardvalue02 = cardValue;
return;
}
}
}

public void RemoveCard()
{
if (IsSelected == true)
{
IsSelected = false;
CC.totalCards -= 1;
}
CC.ResetCards();
Destroy(this.gameObject);
}

public void FlipCard()
{
if (IsSelected == true)
{
CC.ResetCards();
AudioSource.PlayClipAtPoint(Flip, new Vector3(0, 0, -5));
IsSelected = false;
}
}
```

```
void Update()
{
if (CC.totalCards == 20)
{
abletoflip = true;
}
if (CC.cardvalue02 > 0 && CC.cardvalue02 != CC.cardvalue01 && IsSelected == true)
{
Invoke("FlipCard", 0.6f);
}
if (CC.cardvalue01 > 0 && CC.cardvalue01 == CC.cardvalue02 && IsSelected == true )
{
Invoke("RemoveCard", 1f);
}
if (IsSelected == true)
{
this.GetComponent<SpriteRenderer>().sprite = Front;
}
if (IsSelected == false)
{
this.GetComponent<SpriteRenderer>().sprite = Back;
}
}
}
```

CardChecker script.

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

public class CardChecker : MonoBehaviour
{
/// Marvel Memory game René Pol RP-Interactive.nl ©-2021-///

public GameObject [] MCards = new GameObject[20];
public int cardvalue01;
public int cardvalue02;
public int totalCards;
public GameObject Foundpairs;
public AudioClip Magic;
public bool soundplay;

void reshuffle(GameObject[] MCards)
{
for (int t = 0; t < MCards.Length; t++)
{
GameObject tmp = MCards[t];
int r = Random.Range(t, MCards.Length);
MCards[t] = MCards[r];
MCards[r] = tmp;
}
}
```

```
IEnumerator SetupCards()
{
reshuffle(MCards);
totalCards = 1;
yield return new WaitForSeconds(2f);
Instantiate(MCards[0], new Vector3(-5f, 3, 0), Quaternion.identity);
yield return new WaitForSeconds(0.5f);
Instantiate(MCards[1], new Vector3(-2.5f, 3, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[2], new Vector3(0f, 3, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[3], new Vector3(2.5f, 3, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[4], new Vector3(5f, 3, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[5], new Vector3(-5f, 0, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[6], new Vector3(-2.5f, 0, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[7], new Vector3(0f, 0, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[8], new Vector3(2.5f, 0, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[9], new Vector3(5f, 0, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[10], new Vector3(-5f, -3, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[11], new Vector3(-2.5f, -3, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[12], new Vector3(0f, -3, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[13], new Vector3(2.5f, -3, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[14], new Vector3(5f, -3, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[15], new Vector3(-5f, -6, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[16], new Vector3(-2.5f, -6, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[17], new Vector3(0f, -6, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[18], new Vector3(2.5f, -6, 0), Quaternion.identity);
yield return new WaitForSeconds(0.4f);
Instantiate(MCards[19], new Vector3(5f, -6, 0), Quaternion.identity);
totalCards -= 1;
}

void Start()
{
reshuffle(MCards);
StartCoroutine(SetupCards());
soundplay = false;
cardvalue01 = 0;
cardvalue02 = 0;
Foundpairs.SetActive(false);
}
```

```
public void ResetCards()
{
cardvalue01 = 0;
cardvalue02 = 0;
}

IEnumerator ResetIt()
{
yield return new WaitForSeconds(0.5f);
soundplay = false;
}

IEnumerator RelayCards()
{
yield return new WaitForSeconds(2f);
Foundpairs.SetActive(false);
StartCoroutine(SetupCards());
}

void Update()
{
if (cardvalue01 != 0 && cardvalue02 != 0)
{
if (cardvalue01 == cardvalue02 && soundplay == false)
{
AudioSource.PlayClipAtPoint(Magic, new Vector3(0, 0, -10));
soundplay = true;
StartCoroutine(ResetIt());
}
}
if (totalCards == 0)
{
Foundpairs.SetActive(true);
totalCards = 1;
StartCoroutine(RelayCards());
}
}
}
```

Ghosting Script.

```
///ghost trail René Pol ©-2021-///

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

public class Ghosting : MonoBehaviour
{

void Start()
{
Destroy(this.gameObject,0.5f);
}
}
```

Runner Script.

```
///ghost trail René Pol ©-2021-///


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

public class Runner : MonoBehaviour
{
public float Speed;
public float GtrailDelay;
public float GtrailDelaySeconds;
public GameObject ghost;


void Start()
{
GtrailDelaySeconds = GtrailDelay;
}

void Update()
{
transform.Translate(Vector2.right * Speed * Time.deltaTime);
if(GtrailDelaySeconds >0)
{
GtrailDelaySeconds -= Time.deltaTime;
}
else
{
Instantiate(ghost, transform.position, transform.rotation);
GtrailDelaySeconds = GtrailDelay;
}
}
}
```