CREATE GAMES !

Welcome to issue 7.

This month we will create a simple game called Whack A Mole. It's a small project that holds a lot of Different scripts. How I created all can be seen here :

https://youtu.be/DUvoqsISK0A

Ready let's inspect those scripts. I will explain the parts in short as you getting to understand scripting better and better like I do. All projects I made, are done with scripts I could take from my online C# library. First we start with 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 OwnCursor : MonoBehaviour
{
/// Whack A Mole 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;
}
}
```

CREATE GAMES !

The holes are sprites place with sprites of blocks that hold the same color as the background. Their order in layer number will be higher then the mole so it looks like the mole sprite is coming from behind it (Out of the holes in the ground). The magic of game creation happening. The mole is a Sprite with a box 2d Collider and this Script.

Mole Script.

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

public class Mole : MonoBehaviour
{

/// Whack A Mole René Pol RP-Interactive.nl ©-2021-///
```

2 bool variables are created one to go up and the other for going down. A float variable is made for the moving speed of the mole and is set to 4. We create a Vector3 (Coordinates in the game) and call this Vector3 StartP. Finally I defined a sound file to use for when the mole is hit. Those variables are always needed in game creation.

```
public bool GoUp;
private float Speed = 4;
public Vector3 StartP;
public bool GoDown;
public AudioClip Auch;
```

When we click on the mole object (Box 2d Collider) it will add 1 point to the score and plays the audio file we defined once in a spot (Coordinates) near the main camera. The it removes itself.

```
void OnMouseDown()
{
Scoring.Score += 1;
AudioSource.PlayClipAtPoint(Auch, new Vector3(0, 0, -10));
Destroy(this.gameObject);
}
```

On start we set our godown variable to false and the goup variable to true. We give the Vector3 we created (StartP) the exact coordinates of where the object is at start. We do this by setting it to its transformposition.position. So now the StartP will always remember our start position.

```
void Start()
{
GoDown = false;
GoUp = true;
StartP = transform.position;
}
```

A Coroutine (IEnumerator) is created and called reset. When called it will change the variable bool settings after 1 second.

```
IEnumerator Reset()
{
yield return new WaitForSeconds(1f);
GoUp = false;
GoDown = true;
}
```

When goup is false and the position on th y axis is bigger than the start position on the y.axis and godown is set to true, The mole will move straight down until it's position is smaller than the Start position. There it will remove itself.

```
void Update()
{
if (GoUp == false && transform.position.y > StartP.y && GoDown == true)
{
transform.Translate(Vector2.down * Speed * Time.deltaTime);
if(transform.position.y < StartP.y)
{
Destroy(this.gameObject);
}
}
```

When goup is true and the position on the y axis is smaller than the start position + 1.5 on the y.axis and godown is set to false, The mole will move straight up until it's position is bigger than the Start position +1.5. there it will call the Coroutine we made (Reset). Once this Coroutine is executed the mole will go down as we have seen in the previous part of the script.

```
if (GoUp == true && transform.position.y < StartP.y + 1.5f && GoDown == false)
{
transform.Translate(Vector2.up * Speed * Time.deltaTime);
if (transform.position.y > StartP.y + 1.5f)
{
StartCoroutine(Reset());
}
}
}
}
```

For scoring how many moles are hit we create a UI element and call it Scoring. We give it this script and make sure that the line using UnityEngine.UI is included in this script. Without this line the script wont work as it could not find the UI elements.

Scoring script.

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

public class Scoring : MonoBehaviour
{
/// Whack A Mole René Pol RP-Interactive.nl ©-2021-///
```

A static float is created for score. Static makes it accessible from other scripts to change it when needed. Than a Text element is set called scoreT.

```
public static float Score;
public Text scoreT;
```

The scoreT text element is getting it's UI element. (GetComponent)

```
void Start()
{
scoreT = GetComponent<Text>();
}
```

This scoreT text element should show our score float as a string and make it visible always in 2 numbers (00).

```
void Update()
{
scoreT.text = Score.ToString("00");
}
}
```

You can drag the Scoring UI element into the Inspectors view to the scoreT box. Remember the line : Scoring.Score += 1; from the mole script ? it adds 1 to the score variable located a static float in the Scoring script.

Create an empty game Object and give it an icon. Call it GameManager and give it this script :

<u>GameManager script.</u>

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

public class GameManeger : MonoBehaviour
{

/// Whack A Mole René Pol RP-Interactive.nl ©-2021-///
```

Besides the variables created I also will use 2 gameobjects that are in the level called Nice and startbutton.

```
public int MoleNumber;
public float count = 2f;
public GameObject Moles;
public float Roundtime = 30;
public GameObject Nice,startbutton;
public bool runtimers = false;
```

the first function is a Spawn(set) mole function. First it will change the mole number by setting it random to a number from 1 to 6 (Random.Range starts with first number and ends before last given number). When the number is 1 a mole is created at given coordinates and it will execute its script. (Popping up). All 6 positions work this way.

```
public void SetMole()
{
MoleNumber = Random.Range(1, 7);
if (MoleNumber == 1)
{
Instantiate(Moles, new Vector3(-4, -4.44f, 0), Quaternion.identity);

}
if (MoleNumber == 2)
{
Instantiate(Moles, new Vector3(0, -4.44f, 0), Quaternion.identity);

}
if (MoleNumber == 3)
{
Instantiate(Moles, new Vector3(4.2f, -4.44f, 0), Quaternion.identity);

}
if (MoleNumber == 4)
{
Instantiate(Moles, new Vector3(4.27f, -1.1f, 0), Quaternion.identity);

}
if (MoleNumber == 5)
{
Instantiate(Moles, new Vector3(0f, -1.1f, 0), Quaternion.identity);
```

```
}
if (MoleNumber == 6)
{
Instantiate(Moles, new Vector3(-3.8f, -1.1f, 0), Quaternion.identity);

}
}
```

At start we make the nice game object invisible by using SetActive and setting it to false;

```
void Start()
{
Nice.SetActive(false);
}
```

When a game is started the startbutton and nice gameobjects are made invisible. Score is set to 0 and the roundtime to 30. Popup count is set to 2 and all timers are activated.

```
public void StartGame()
{
startbutton.SetActive(false);
Nice.SetActive(false);
Scoring.Score = 0;
Roundtime = 30;
count = 2f;
runtimers = true;
}
```

When the runtimers bool is set to true it will countdown the Roundtime to 0. When the roundtime is smaller or equal to 0 it will make the nice and startbutton objects visible again by setting SetActive to true. Round time is set to 0 and count is reset back to 2. On game restart all wil work again after the resets done in the startgame function.

```
void Update()
{
if (runtimers == true)
{
Roundtime -= Time.deltaTime;
if (Roundtime <= 0)
{
Nice.SetActive(true);
startbutton.SetActive(true);
Roundtime = 0;
count = 2f;
}
```

When the roundtime is bigger than 0 the count will also count down to 0. When this count is smaller than or equal to 0 it will activate te setmole function (Spawns a mole) and then resets itself to 2. So every 2 seconds a mole is spawned as long as the Roundtime is running.

```
if (Roundtime > 0)
{
count -= Time.deltaTime;
}
if (count <= 0)
{
{
SetMole();
count = 2f;
}
}
}
}
}
```

The start button is a sprite with a box collider 2D. When clicked on it it starts the game from the gamemanager script. Give this startbutton this script.

Start button script.

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

public class Startgame : MonoBehaviour
{

/// Whack A Mole René Pol RP-Interactive.nl ©-2021-///
```

A private link to the script GameManeger is made and called GM.

```
private GameManeger GM;
```

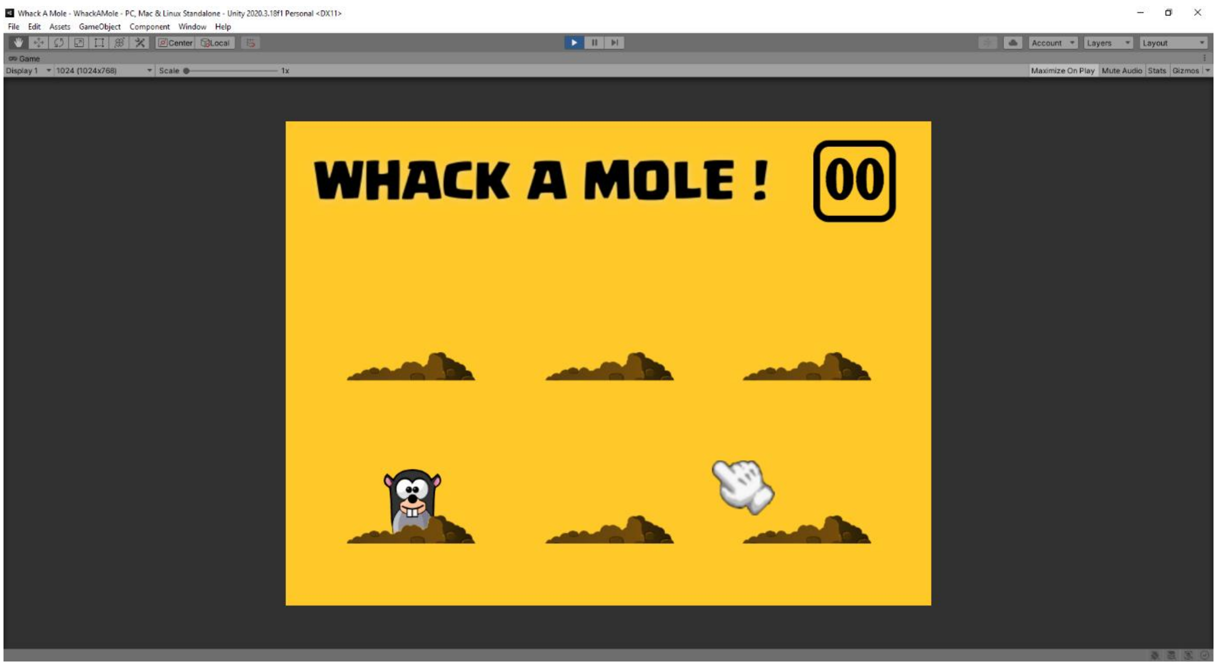
In start the script will look for GM which should be the GameManeger script

```
void Start()
{
GM = FindObjectOfType<GameManeger>();
}
```

When clicked on it executes the StartGame function located in GM(gamemanager script)

```
void OnMouseDown()
{
GM.StartGame();
}
}
```

For the game music just drop the soundfile into the scene. Make sure the loop box is selected so it will keep repeating after it played the file. Select the Play on Awake box to so it will start to play as soon as the game starts.
Whack some moles ! Use your own ideas and imagination to make it your own. Change themes or sprites, add new music and sound effects. Perhaps use cool particle effects. The basic is here for you to use.


Whack A Mole - WhackAMole - PC, Mac & Linux Standalone - Unity 2020.3.18f1 Personal <DX11>
File Edit Assets GameObject Component Window Help
Center Local
Account Layers Layout
Game
Display 1 1024 (1024x768) Scale 1x
Maximize On Play Mute Audio Stats Gizmos
WHACK A MOLE !
00

How To ?

Games that play on a grid. Is there an easy way to create a grid so you can have a gameboard setup fast. In this How to we will create a grid with 2 simple scripts. From here you can expand with your own imagination.

Tile Script.

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

public class Tile : MonoBehaviour
{
/// GridMaker tiles René Pol©19-09-2021-///
```

We create some color variables and define a game object used as highlight. Also we use a Spriterenderer to determine if a sprite is visible or not. This 2D sprite is a Square white sprite and so is it’s child.

```
[SerializeField] private Color _baseColor, _offsetColor;
[SerializeField] private SpriteRenderer _renderer;
[SerializeField] private GameObject _highlight;
```

A bool variable is used to determine if the isOffset is true or false;

```
public void Init(bool isOffset)
{
_renderer.color = isOffset ? _offsetColor : _baseColor;
}
```

When the mouse cursor moves over the tile it will show the highlight sprite on mouse exit it will hide the highlight sprite again.

```
void OnMouseEnter()
{
_highlight.SetActive(true);
}

void OnMouseExit()
{
_highlight.SetActive(false);
}
}
```

You can drag the sprite renderer and highlight sprites to the inspector’s view.

Place an empty 2D GameObject and call it GridCreator give it an icon if you want. Give it this script:

<u>Grid creator script.</u>

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

public class Gridcreator : MonoBehaviour
{

/// GridMaker tiles René Pol©19-09-2021-///
```

Int variables are used to set width and height of grid squares, A tile prefab to use is defined. A transform is defined so the grid will be always setup in the center of that transform (The camera).

```
[SerializeField] private int _width, _height;
[SerializeField] private Tile _tilePrefab;
[SerializeField] private Transform _cam;
private Dictionary<Vector2, Tile> _tiles;
```

In the start function the grid is immediately created. Tis function you can call from any place in the script.

```
void Start()
{
GenerateGrid();
}
```

The generate grid builds the grid using the Vector2 and Tile for the size that is given in the inspector's view. It will lay down the tile prefab and every even/uneven tile gets to use its own given color. The transform (Camera) given in the inspector's view will make sure the full grid is visible and in the center.

```
void GenerateGrid()
{
_tiles = new Dictionary<Vector2, Tile>();
for (int x = 0; x < _width; x++)
{
for (int y = 0; y < _height; y++)
{
var spawnedTile = Instantiate(_tilePrefab, new Vector3(x, y),
Quaternion.identity);
spawnedTile.name = $"Tile {x} {y}";

var isOffset = (x % 2 == 0 && y % 2 != 0) || (x % 2 != 0 && y % 2 == 0);
spawnedTile.Init(isOffset);
_tiles[new Vector2(x, y)] = spawnedTile;
}
}
_cam.transform.position = new Vector3((float)_width / 2 - 0.5f, (float)_height / 2
- 0.5f, -10);
}
public Tile GetTileAtPosition(Vector2 pos)
{
if (_tiles.TryGetValue(pos, out var tile)) return tile;
return null;
}
}
```

You can see how it all works here: https://youtu.be/S-5dwVC7J6g

All scripts used in issue 07 are here for you.

Own Cursor Script

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

public class OwnCursor : MonoBehaviour
{

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

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

Mole Script

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

public class Mole : MonoBehaviour
{

public bool GoUp;
private float Speed = 4;
public Vector3 StartP;
public bool GoDown = false;
public AudioClip Auch;

void OnMouseDown()
{
Scoring.Score += 1;
AudioSource.PlayClipAtPoint(Auch, new Vector3(0, 0, -10));
Destroy(this.gameObject);
}
```

```
void Start()
{
GoDown = false;
GoUp = true;
StartP = transform.position;
}

IEnumerator Reset()
{
yield return new WaitForSeconds(1f);
GoUp = false;
GoDown = true;
}

void Update()
{
if (GoUp == false && transform.position.y > StartP.y && GoDown == true)
{
transform.Translate(Vector2.down * Speed * Time.deltaTime);
if(transform.position.y < StartP.y)
{
Destroy(this.gameObject);
}
}
if (GoUp == true && transform.position.y < StartP.y + 1.5f && GoDown == false)
{
transform.Translate(Vector2.up * Speed * Time.deltaTime);
if (transform.position.y > StartP.y + 1.5f)
{
StartCoroutine(Reset());
}
}
}
}
```

Scoring Script

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

public class Scoring : MonoBehaviour
{
public static float Score;
public Text scoreT;
void Start()
{
scoreT = GetComponent<Text>();
}

void Update()
{
scoreT.text = Score.ToString("00");
}
}
```

GameManager Script

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


public class GameManeger : MonoBehaviour
{

public int MoleNumber;
public float count = 2f;
public GameObject Moles;
public float Roundtime = 30;
public GameObject Nice,startbutton;
public bool runtimers = false;

public void SetMole()
{
MoleNumber = Random.Range(1, 7);
if (MoleNumber == 1)
{
Instantiate(Moles, new Vector3(-4.45f, -4.30f, 0), Quaternion.identity);
}
if (MoleNumber == 2)
{
Instantiate(Moles, new Vector3(-0.15f, -4.30f, 0), Quaternion.identity);
}
if (MoleNumber == 3)
{
Instantiate(Moles, new Vector3(4.52f, -4.30f, 0), Quaternion.identity);
}
if (MoleNumber == 4)
{
Instantiate(Moles, new Vector3(4.52f, -0.61f, 0), Quaternion.identity);
}
if (MoleNumber == 5)
{
Instantiate(Moles, new Vector3(-0.15f, -0.61f, 0), Quaternion.identity);
}
if (MoleNumber == 6)
{
Instantiate(Moles, new Vector3(-4.45f, -0.61f, 0), Quaternion.identity);
}
}


void Start()
{
Nice.SetActive(false);
}
```

```
public void StartGame()
{
startbutton.SetActive(false);
Nice.SetActive(false);
Scoring.Score = 0;
Roundtime = 30;
count = 2f;
runtimers = true;
}

void Update()
{
if (runtimers == true)
{
Roundtime -= Time.deltaTime;
if (Roundtime <= 0)
{
Nice.SetActive(true);
startbutton.SetActive(true);
Roundtime = 0;
count = 2f;
}
if (Roundtime > 0)
{
count -= Time.deltaTime;
}
if (count <= 0)
{
{
SetMole();
count = 2f;
}
}
}
}
}
```

Start Button Script

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

public class Startgame : MonoBehaviour
{
private GameManeger GM;

void Start()
{
GM = FindObjectOfType<GameManeger>();
}
void OnMouseDown()
{
GM.StartGame();
}
}
```

Tile Script

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

public class Tile : MonoBehaviour
{
/// GridMaker tiles René Pol©19-09-2021-///

[SerializeField] private Color _baseColor, _offsetColor;
[SerializeField] private SpriteRenderer _renderer;
[SerializeField] private GameObject _highlight;

public void Init(bool isOffset)
{
_renderer.color = isOffset ? _offsetColor : _baseColor;
}

void OnMouseEnter()
{
_highlight.SetActive(true);
}

void OnMouseExit()
{
_highlight.SetActive(false);
}
}
```

GridCreator script

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

public class Gridcreator : MonoBehaviour
{

/// GridMaker tiles René Pol©19-09-2021-///

[SerializeField] private int _width, _height;
[SerializeField] private Tile _tilePrefab;
[SerializeField] private Transform _cam;
private Dictionary<Vector2, Tile> _tiles;

void Start()
{
GenerateGrid();
}
```

```
void GenerateGrid()
{
_tiles = new Dictionary<Vector2, Tile>();
for (int x = 0; x < _width; x++)
{
for (int y = 0; y < _height; y++)
{
var spawnedTile = Instantiate(_tilePrefab, new Vector3(x, y),
Quaternion.identity);
spawnedTile.name = $"Tile {x} {y}";
var isOffset = (x % 2 == 0 && y % 2 != 0) || (x % 2 != 0 && y % 2 == 0);
spawnedTile.Init(isOffset);
_tiles[new Vector2(x, y)] = spawnedTile;
}
}
_cam.transform.position = new Vector3((float)_width / 2 - 0.5f, (float)_height / 2
- 0.5f, -10);
}

public Tile GetTileAtPosition(Vector2 pos)
{
if (_tiles.TryGetValue(pos, out var tile)) return tile;
return null;
}
}
```