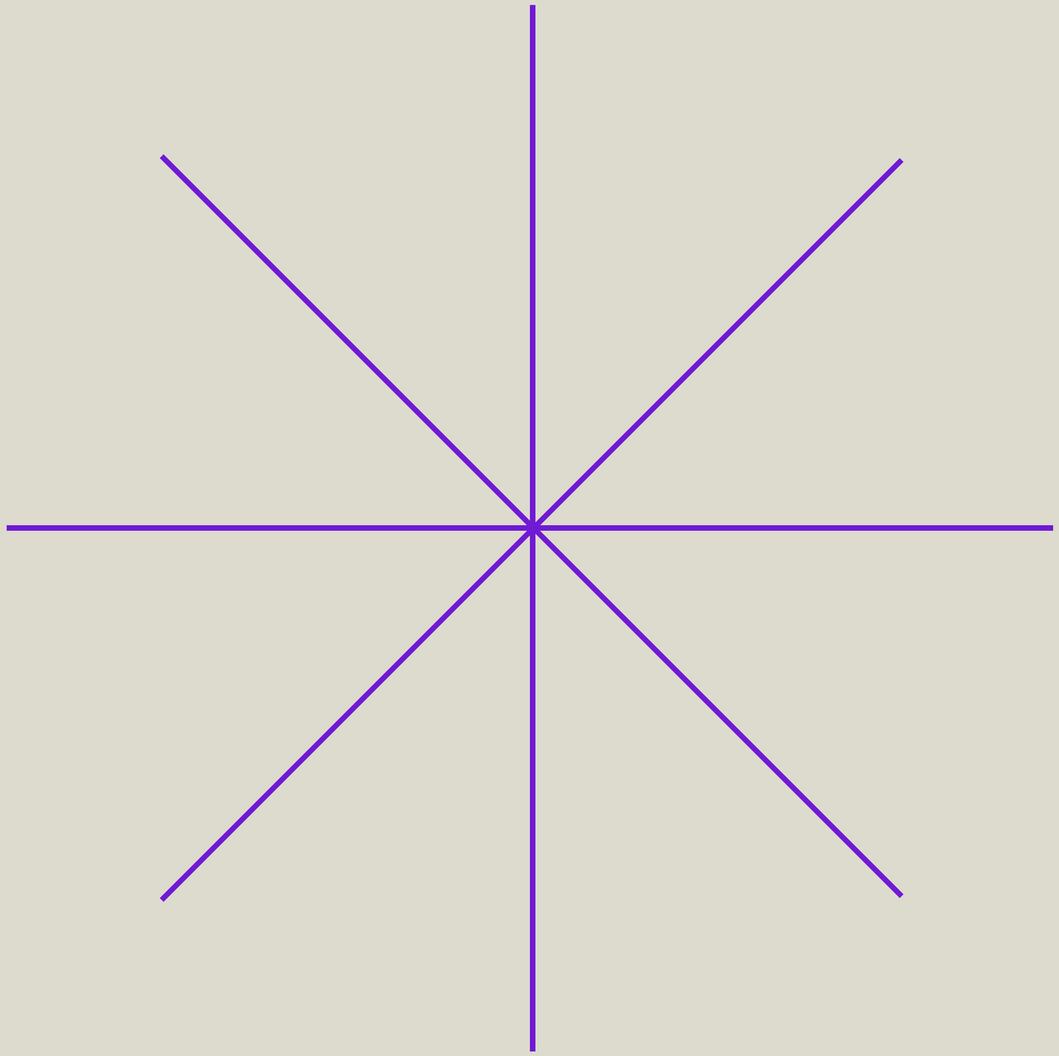
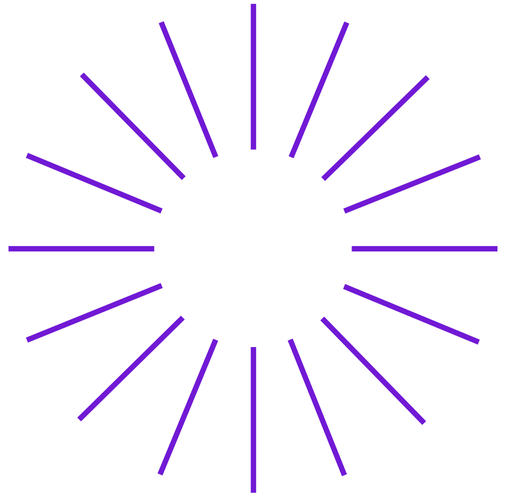
DESENVOLVIMENTO PARA ÂMBIENTES MÓVEIS

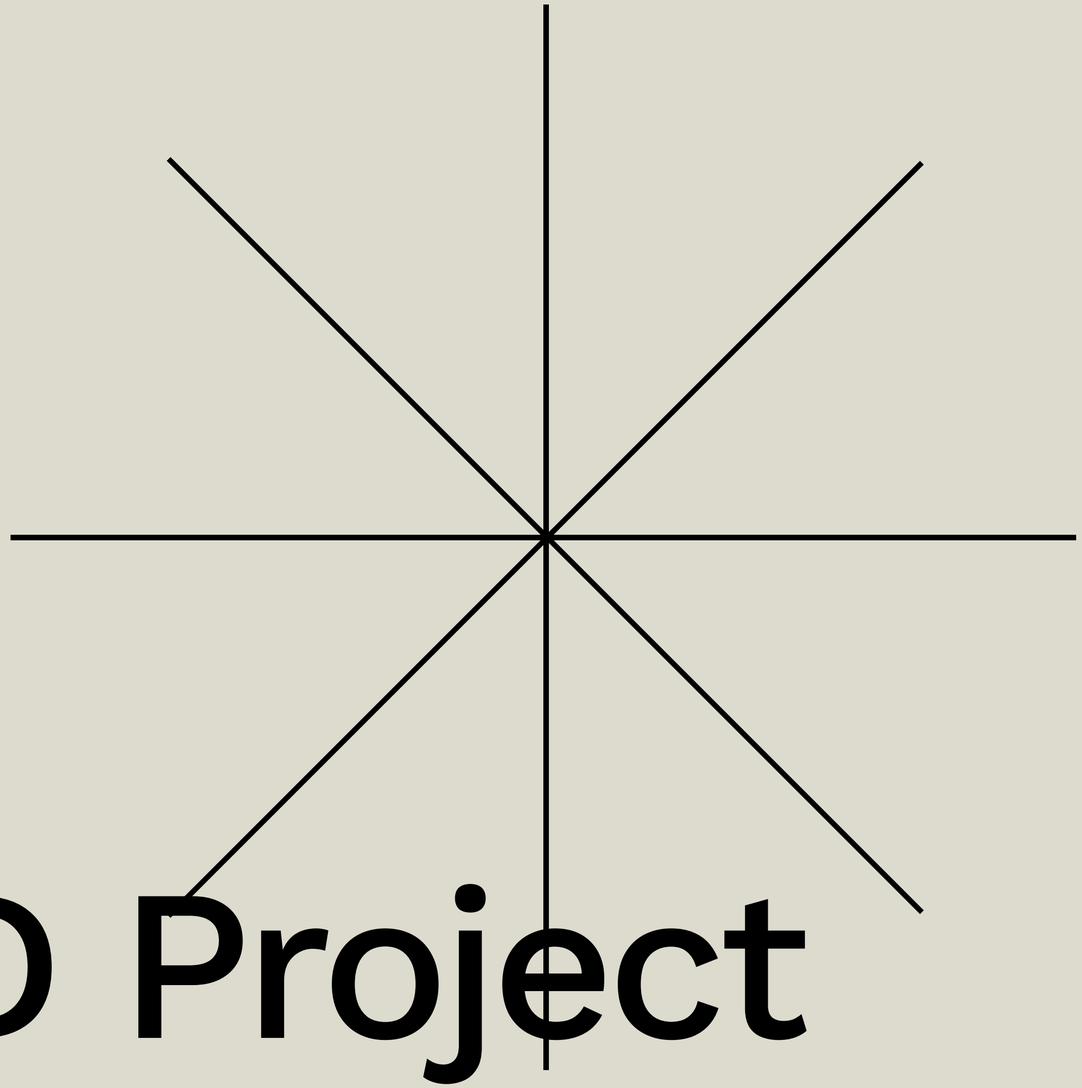


# 2D Exercise

DANIELA MONTEIRO

# INDEX

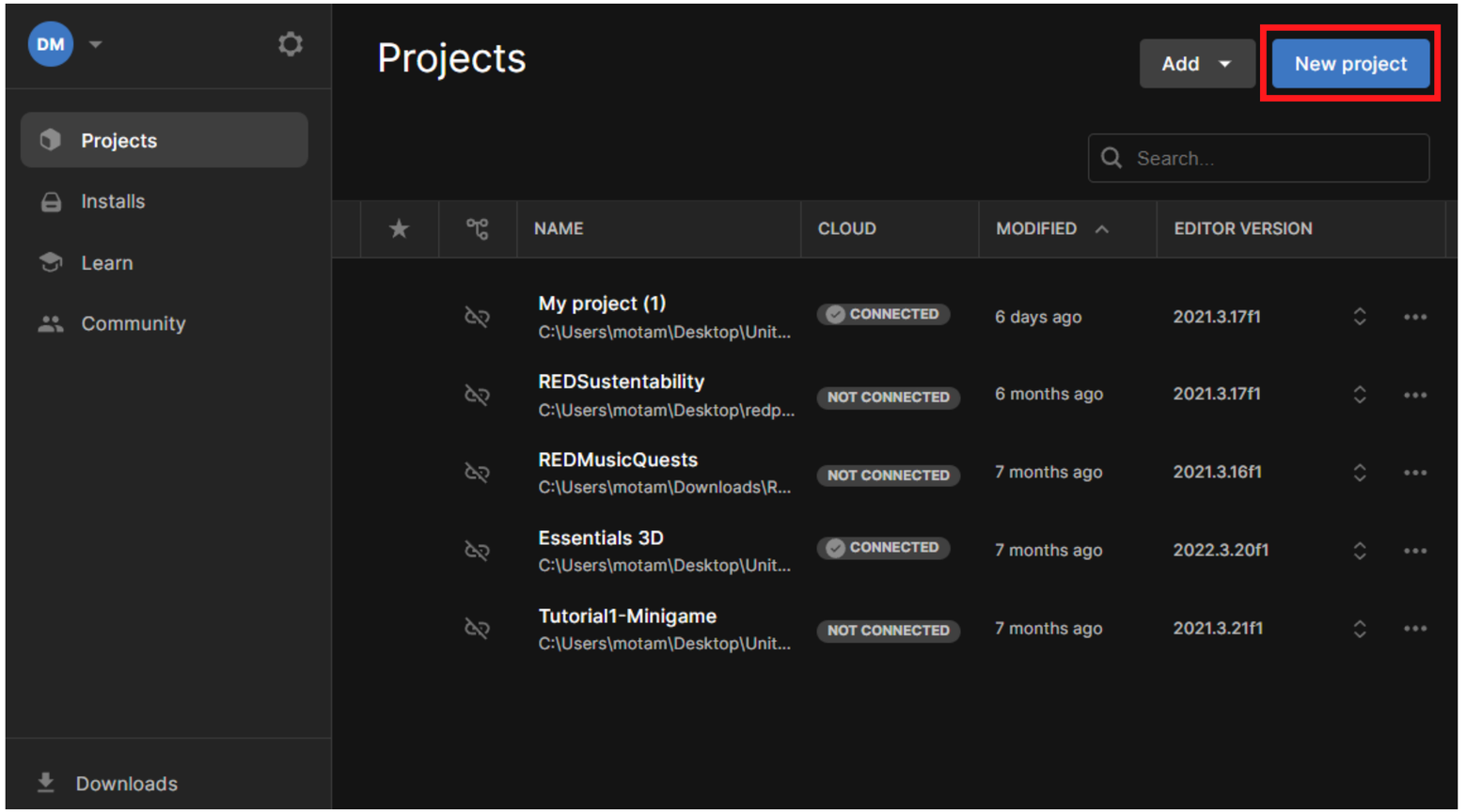




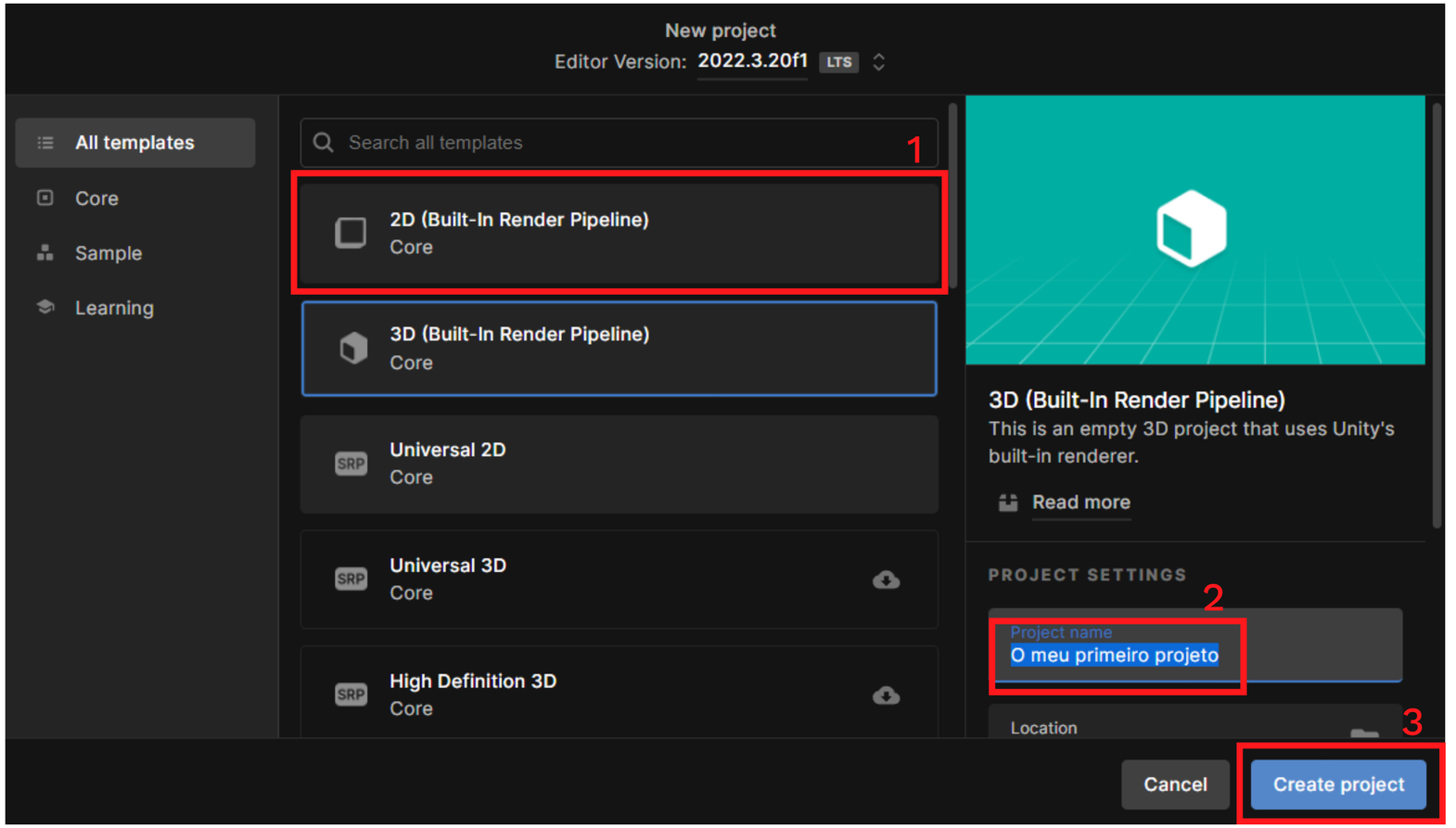
# 01
# Start a new 2D Project for Mobile

DANIELA MONTEIRO

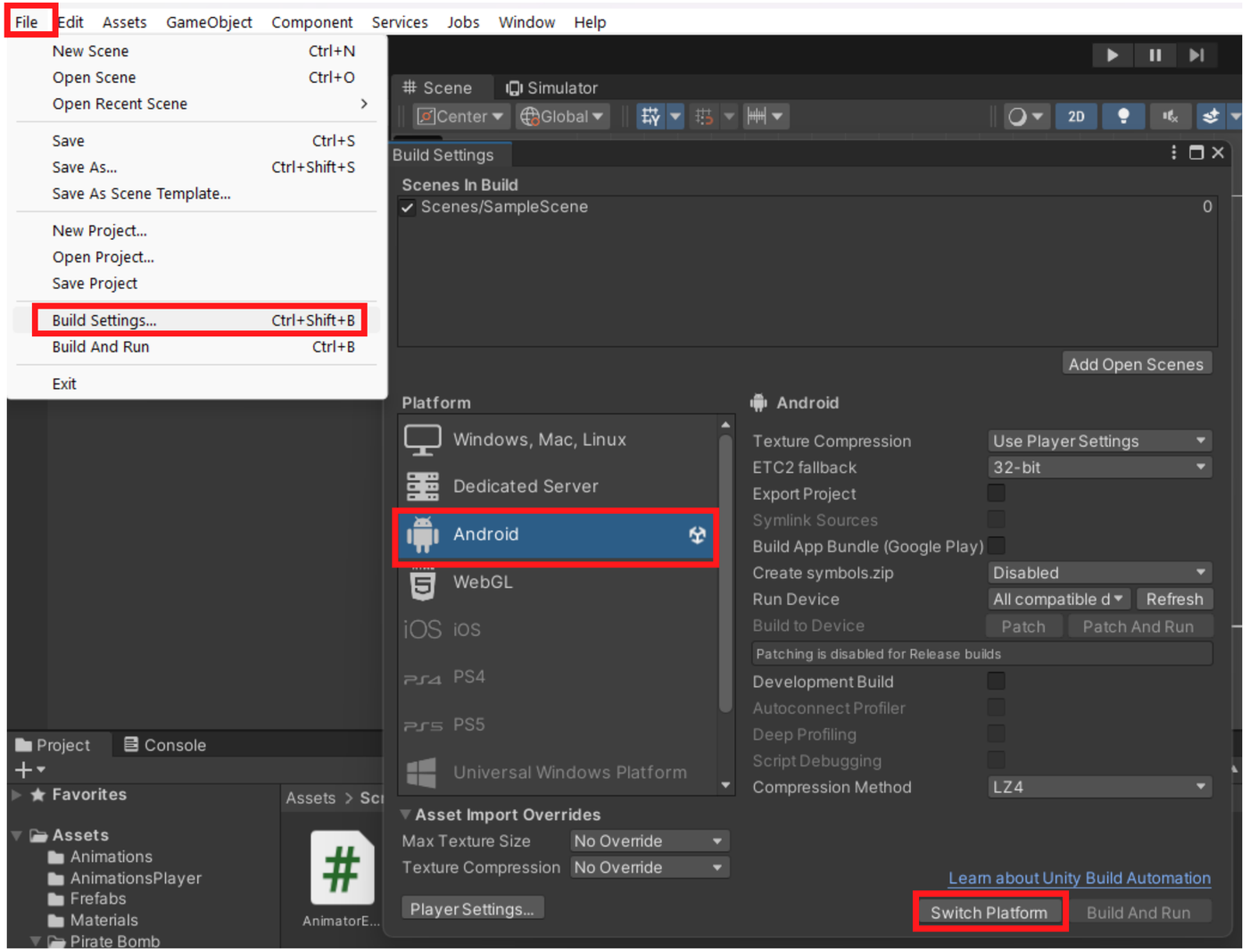
# CREATE A NEW PROJECT



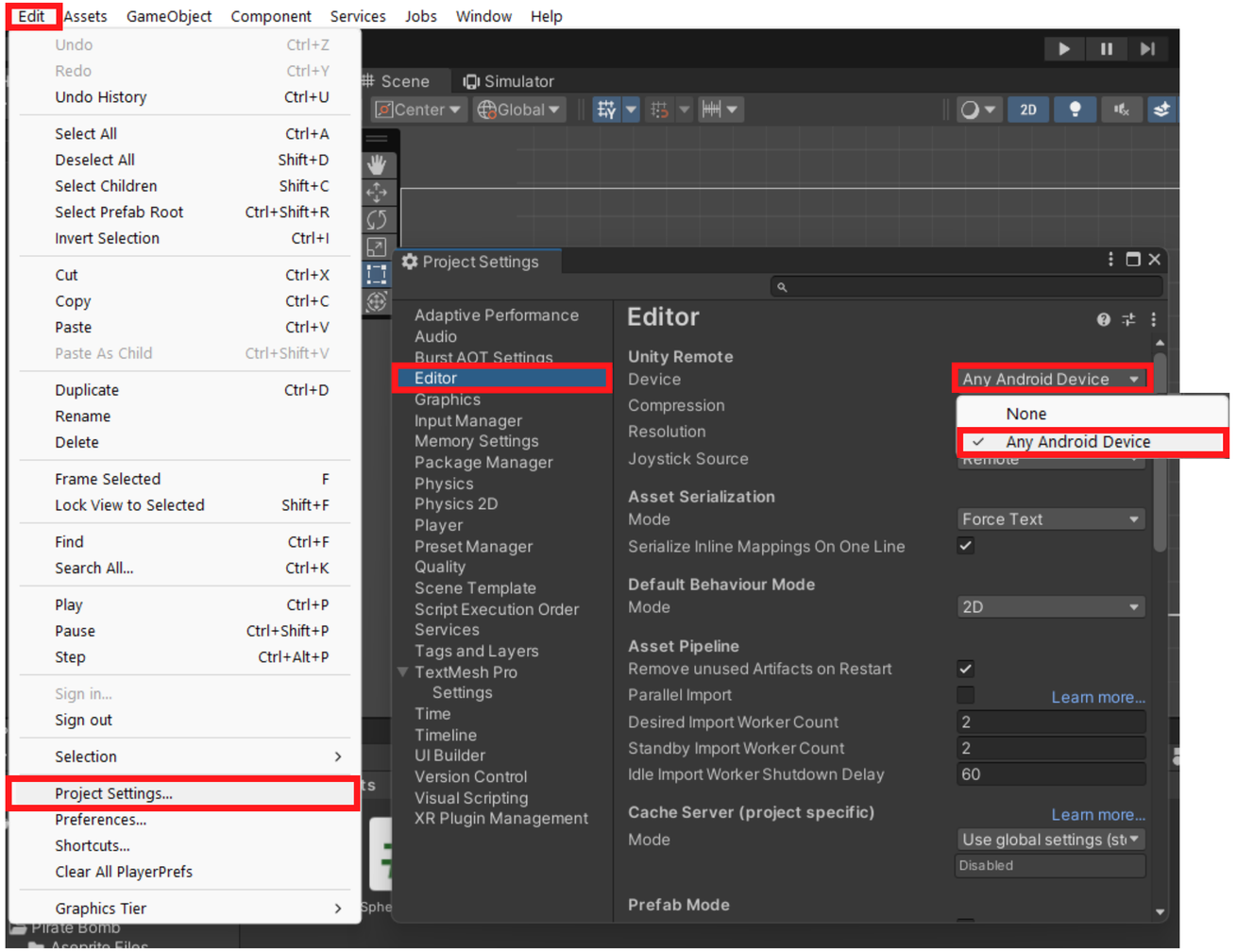
# CREATE A NEW 2D PROJECT

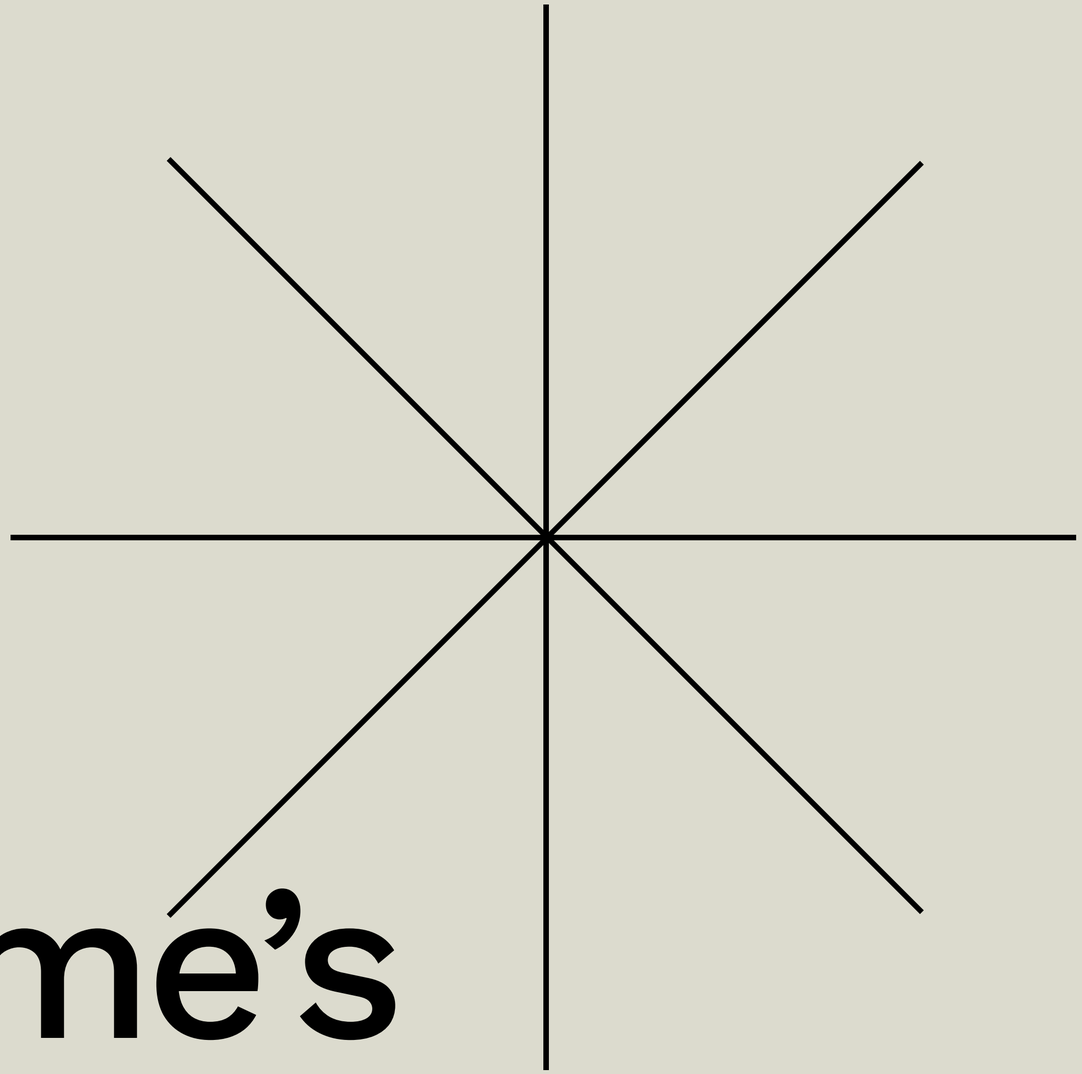


# SWITCH PLATFORMS



# CHANGE YOUR PROJECTS PREFERENCES

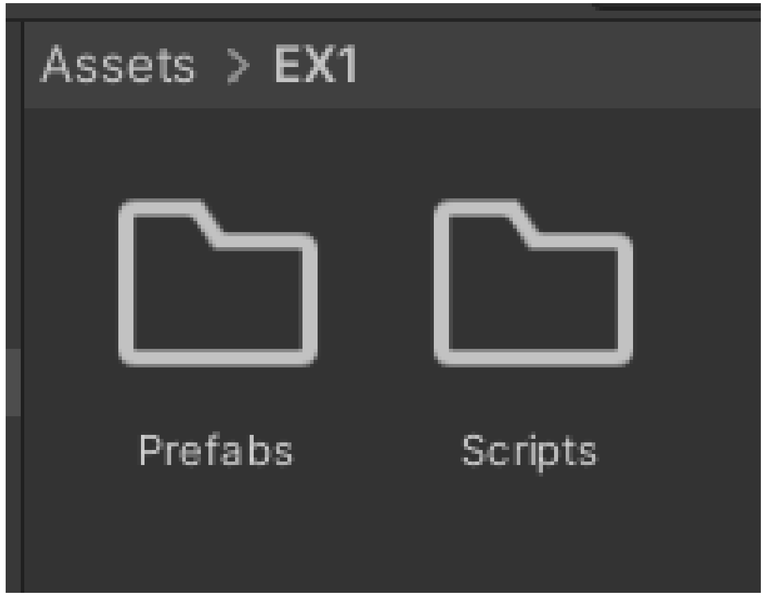




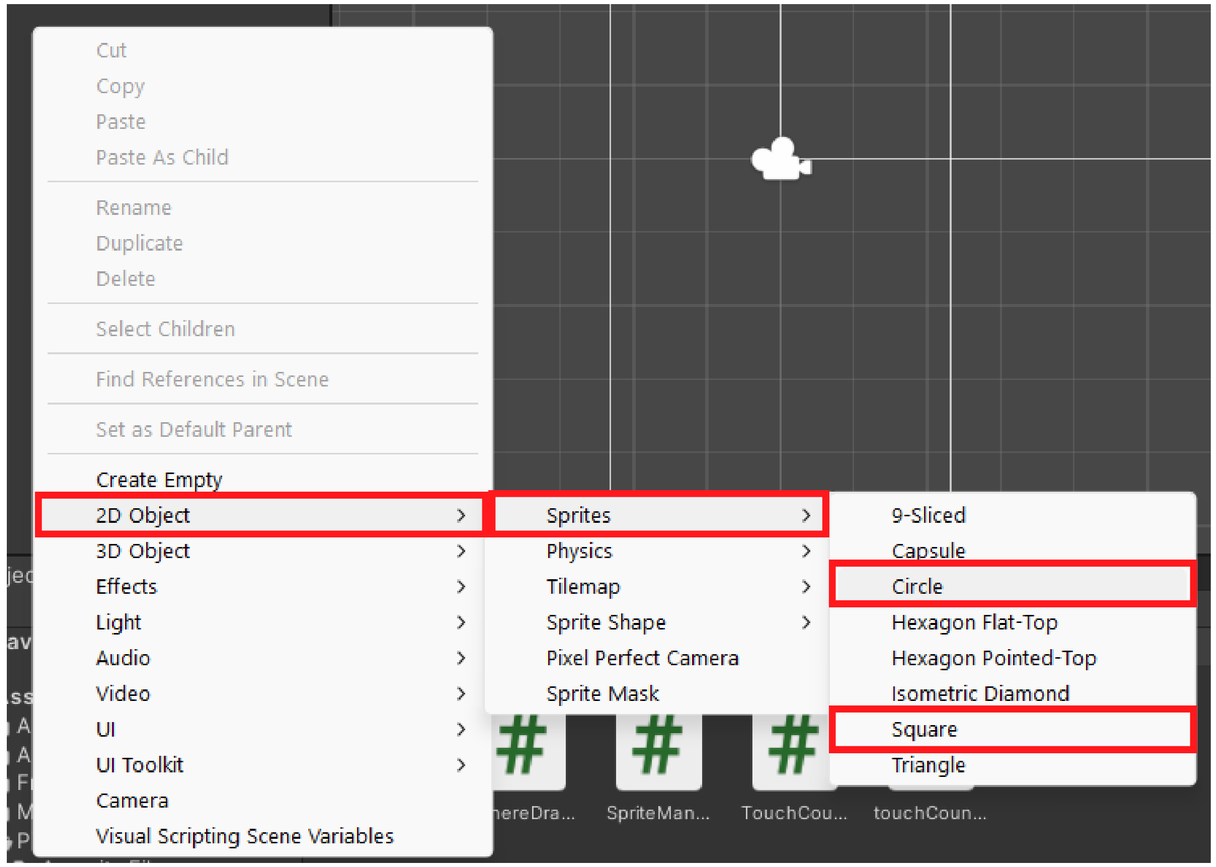
# 02
# Create the Game's Environment

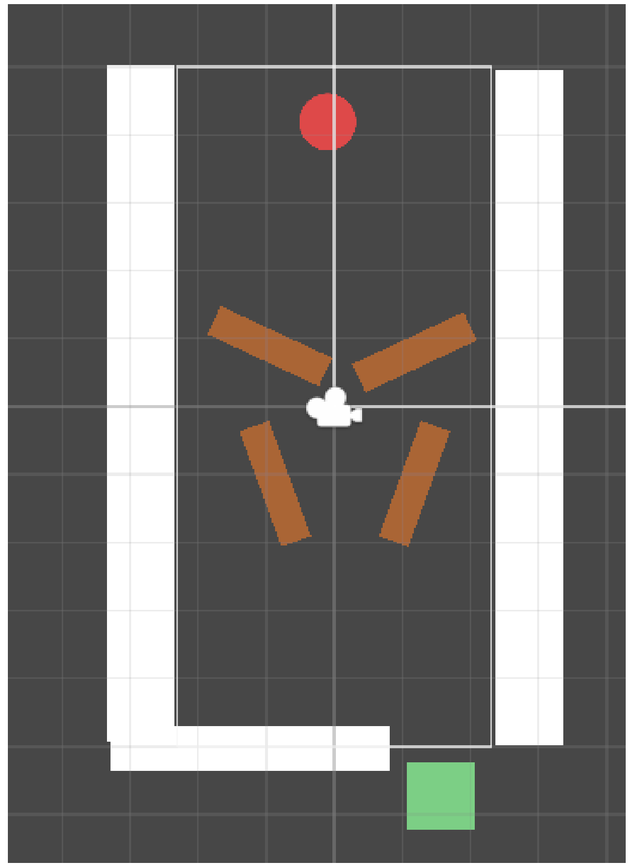
DANIELA MONTEIRO

# ADD 2 MORE FOLDERS TO YOUR PROJECT

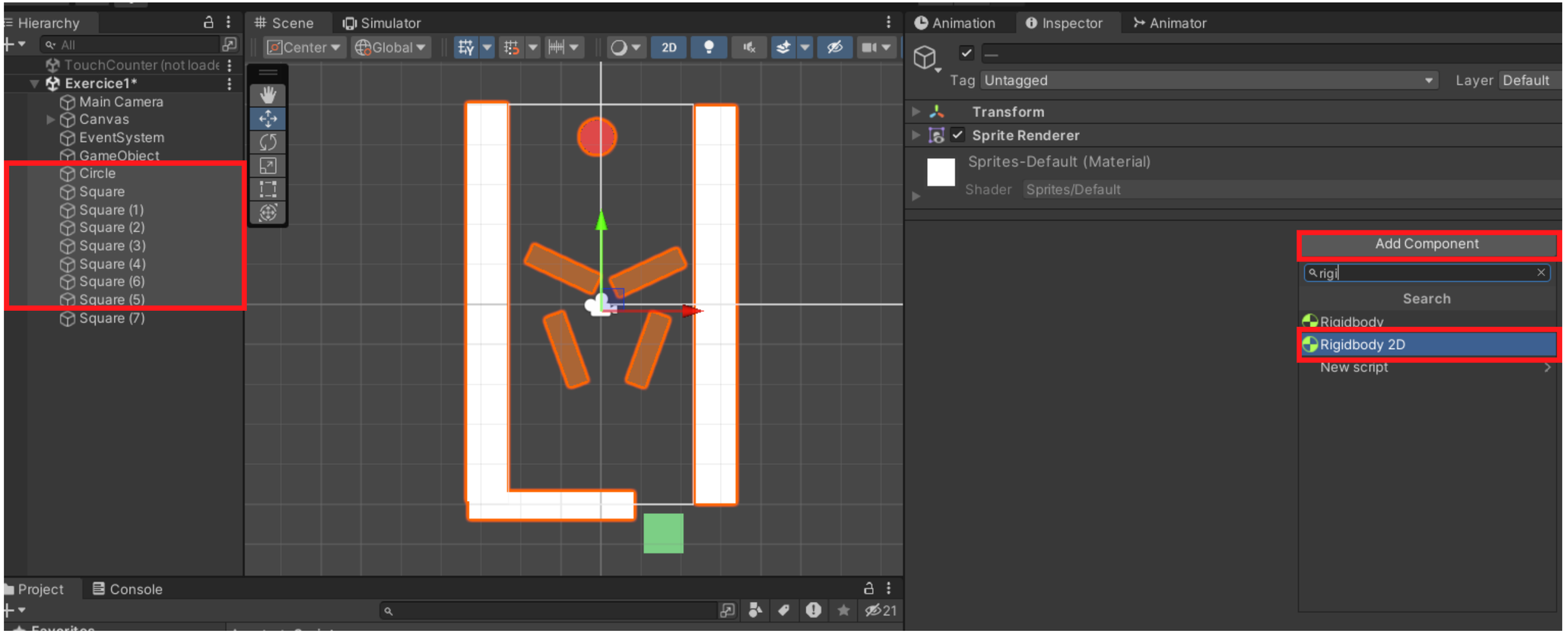


# ADD A CIRCLE, FOUR RECTANGLES AND THE GAME'S WALLS

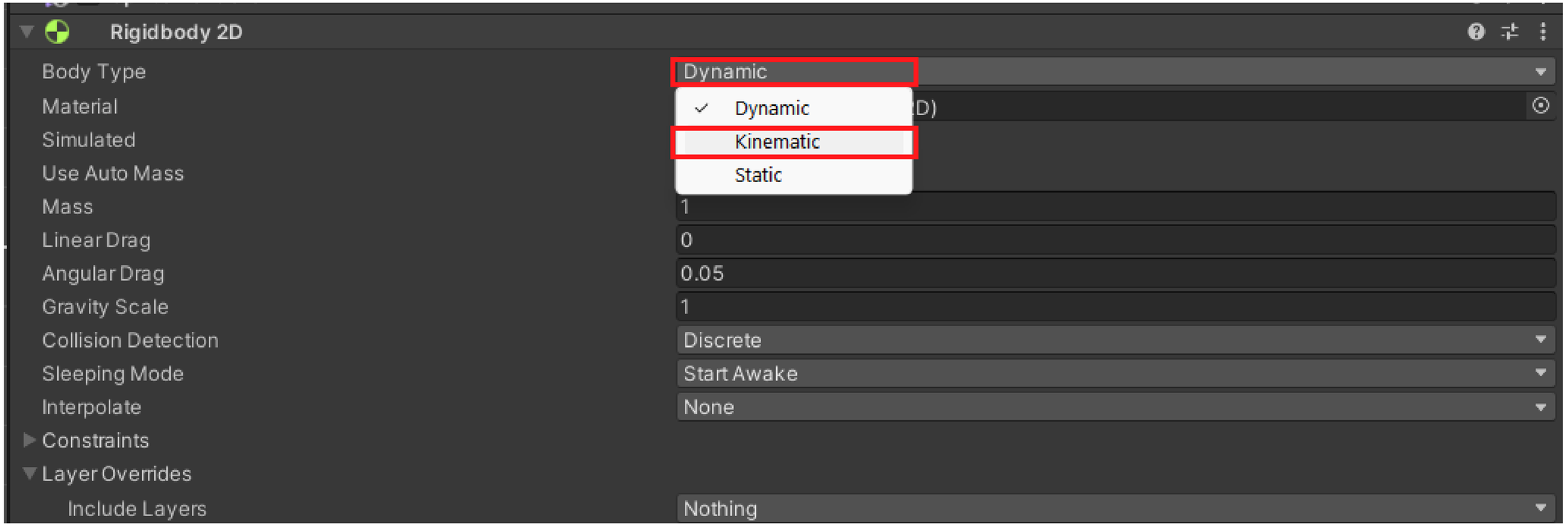


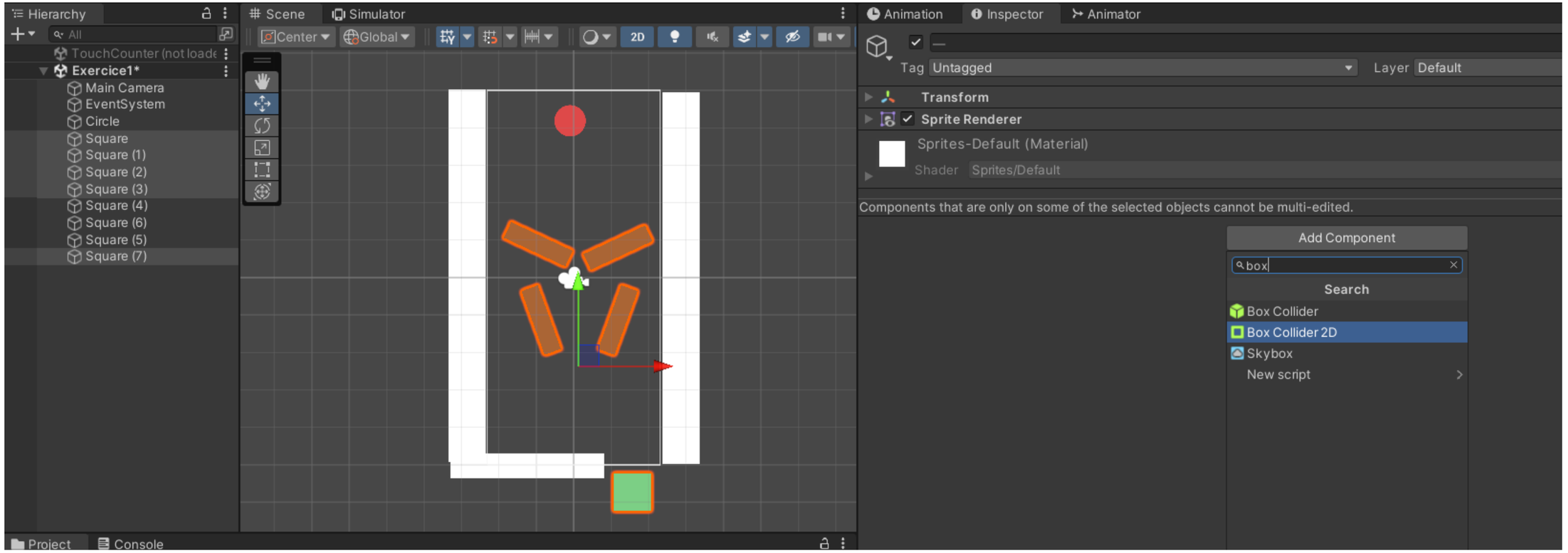
# ADD THE COMPONENT ‘RIGID BODY 2D’ TO ALL ELEMENTS, EXCEPT THE GREEN SQUARE



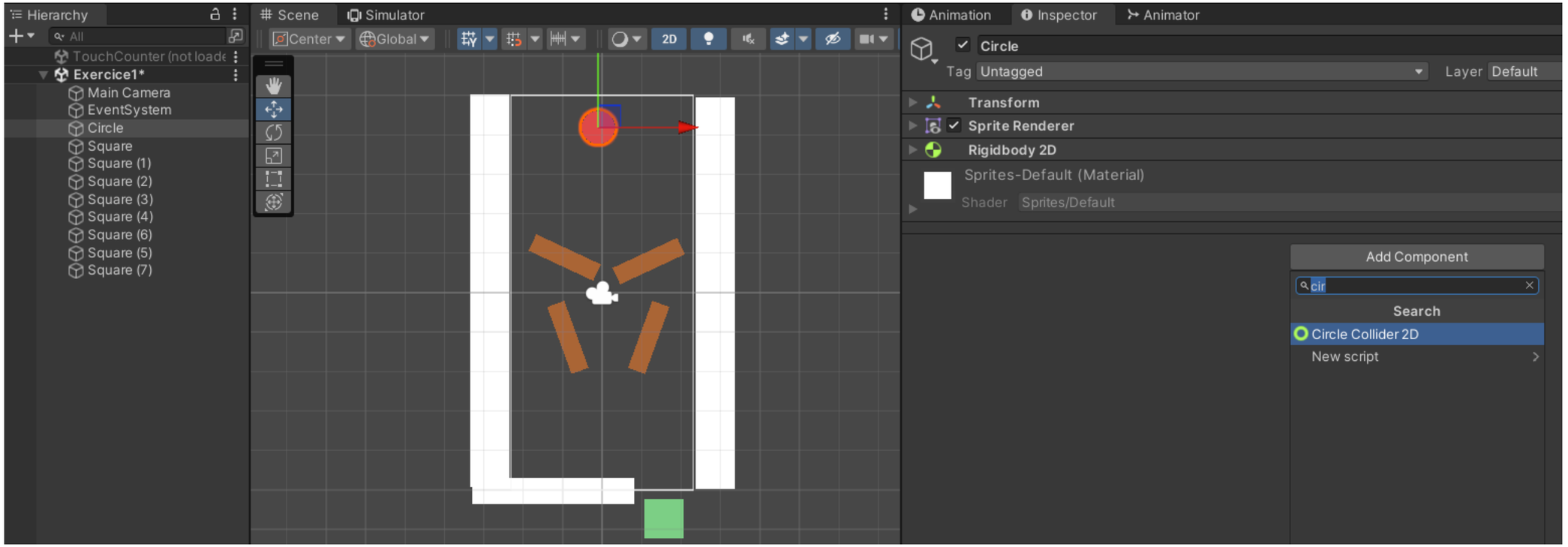
# CHANGE THE BODYTYPE TO ‘KINEMATIC’



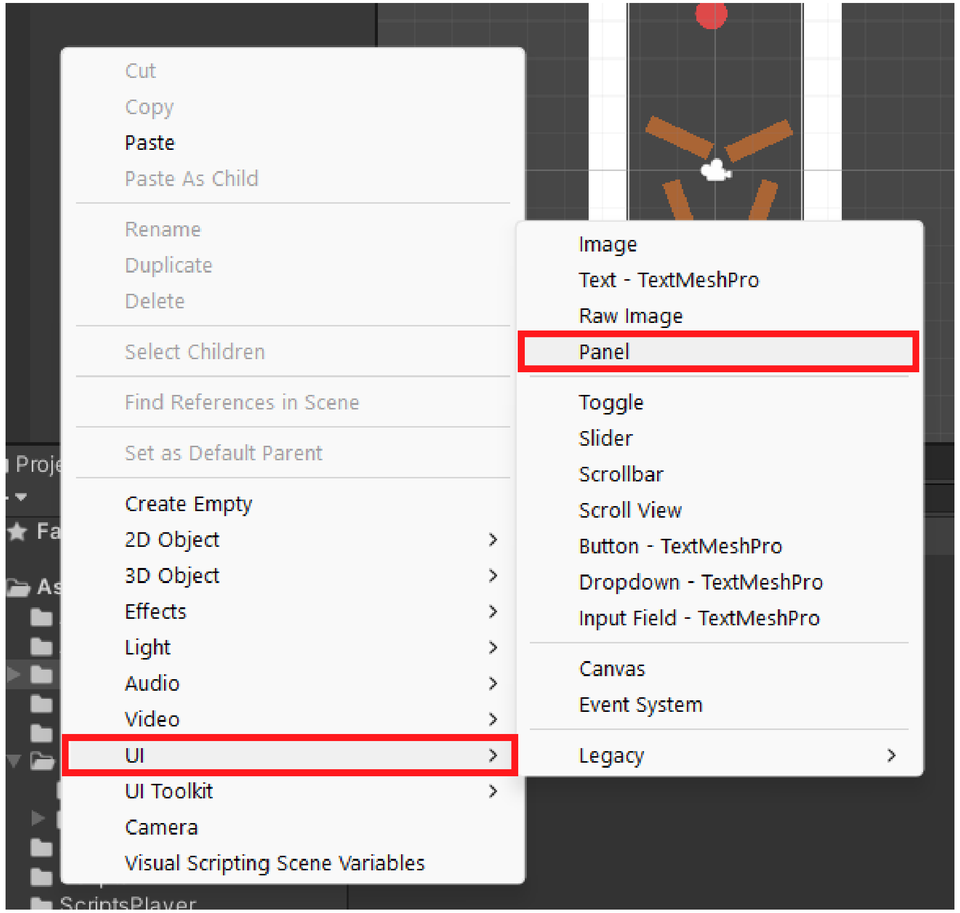
# ADD THE COMPONENT ‘BOX COLLIDER 2D’ TO ALL ELEMENTS, EXCEPT THE CIRCLE

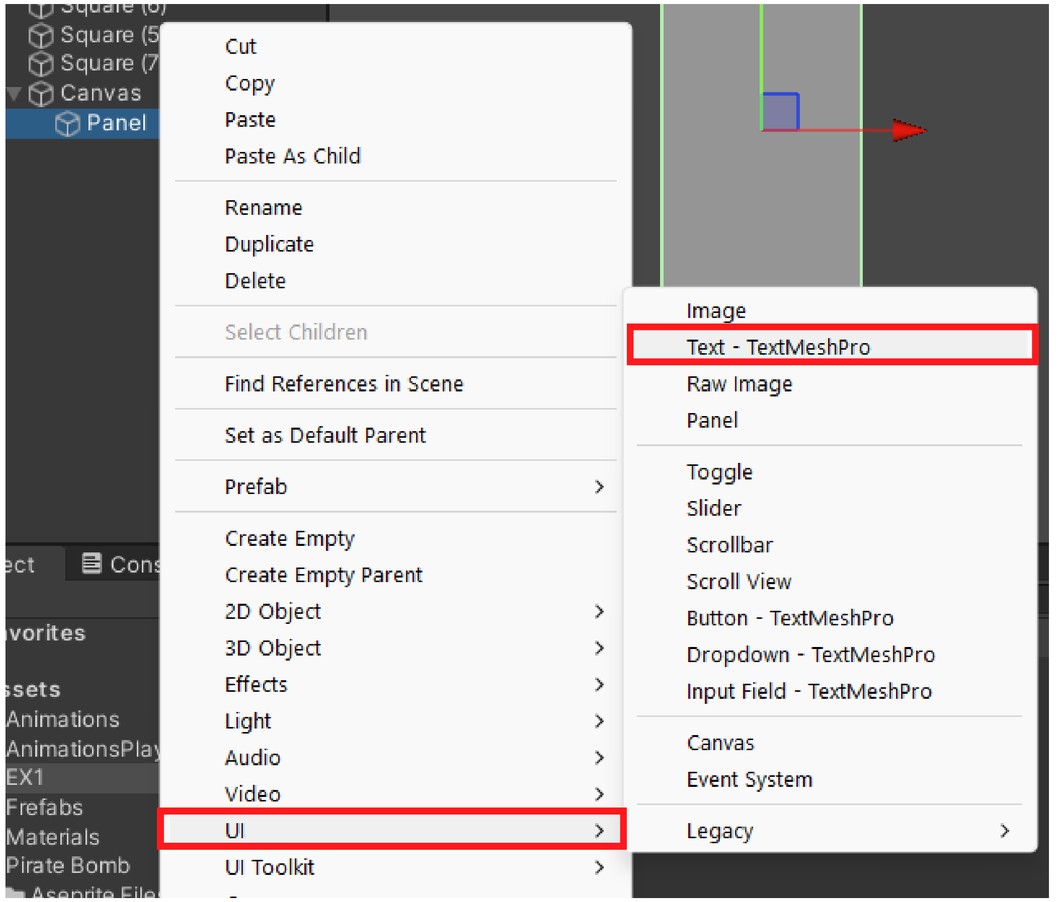


# ADD THE COMPONENT ‘CIRCLE COLLIDER 2D’ TO THE CIRCLE

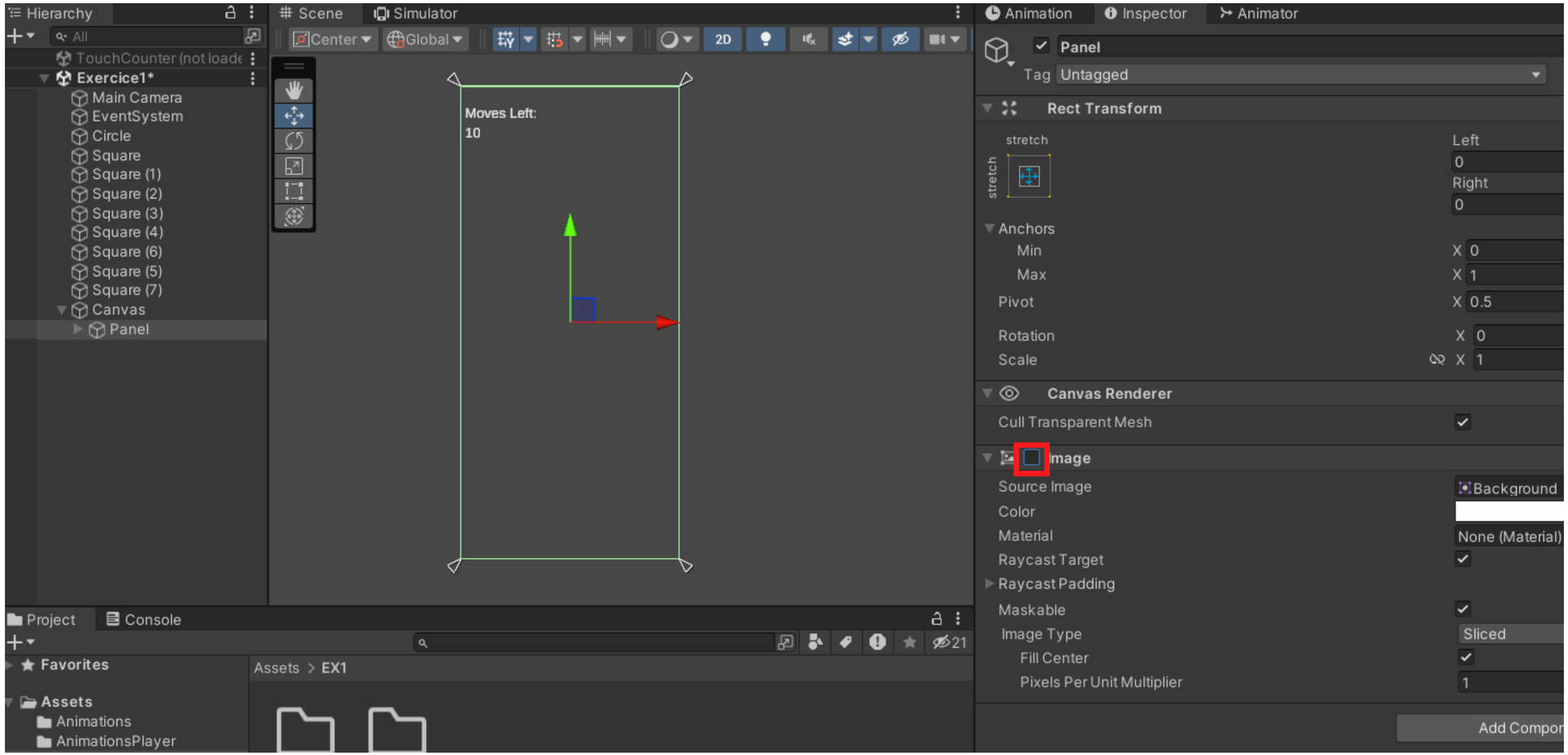


# ADD A PANEL FOR THE UI, WITH TWO TEXT BOXES

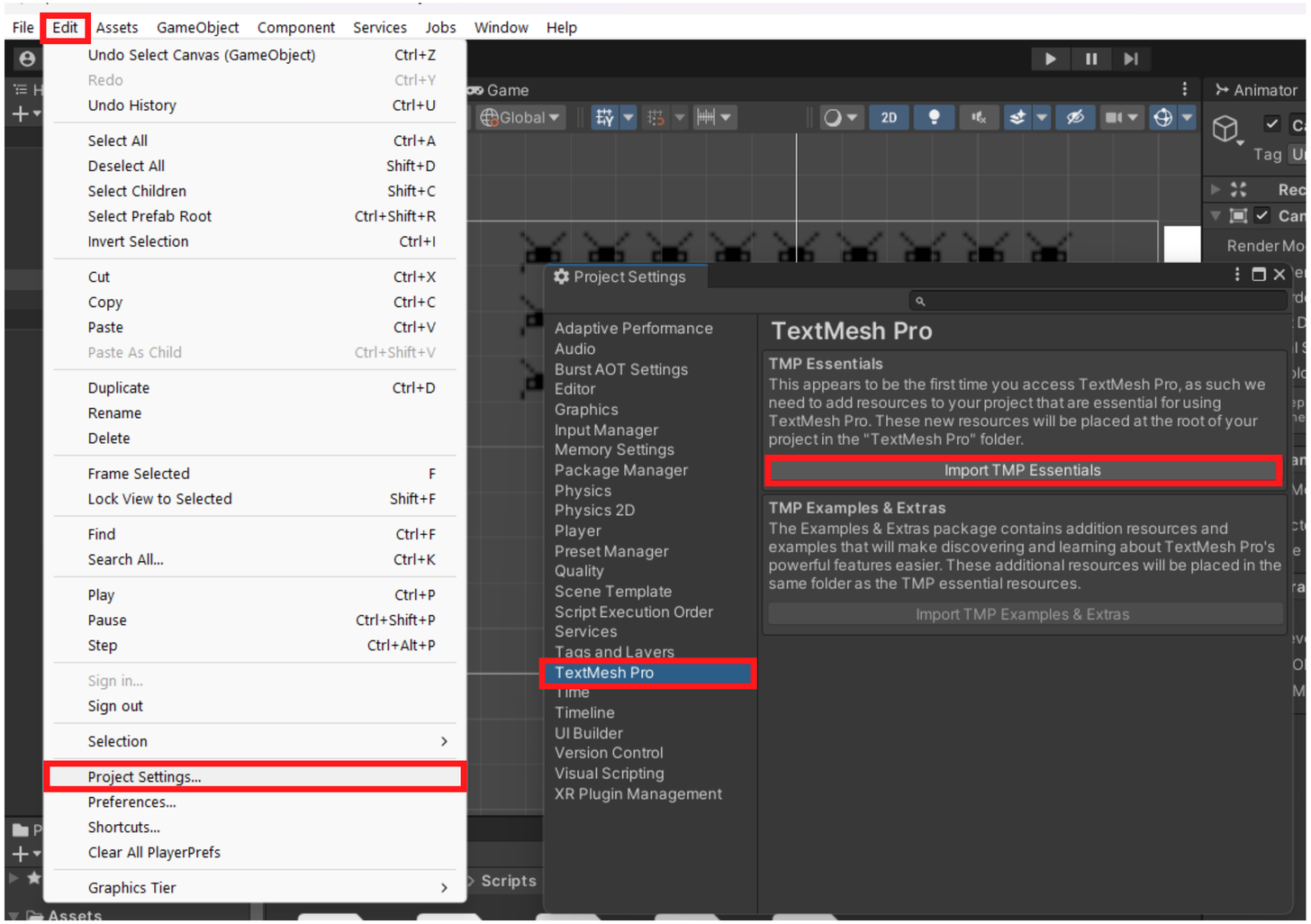


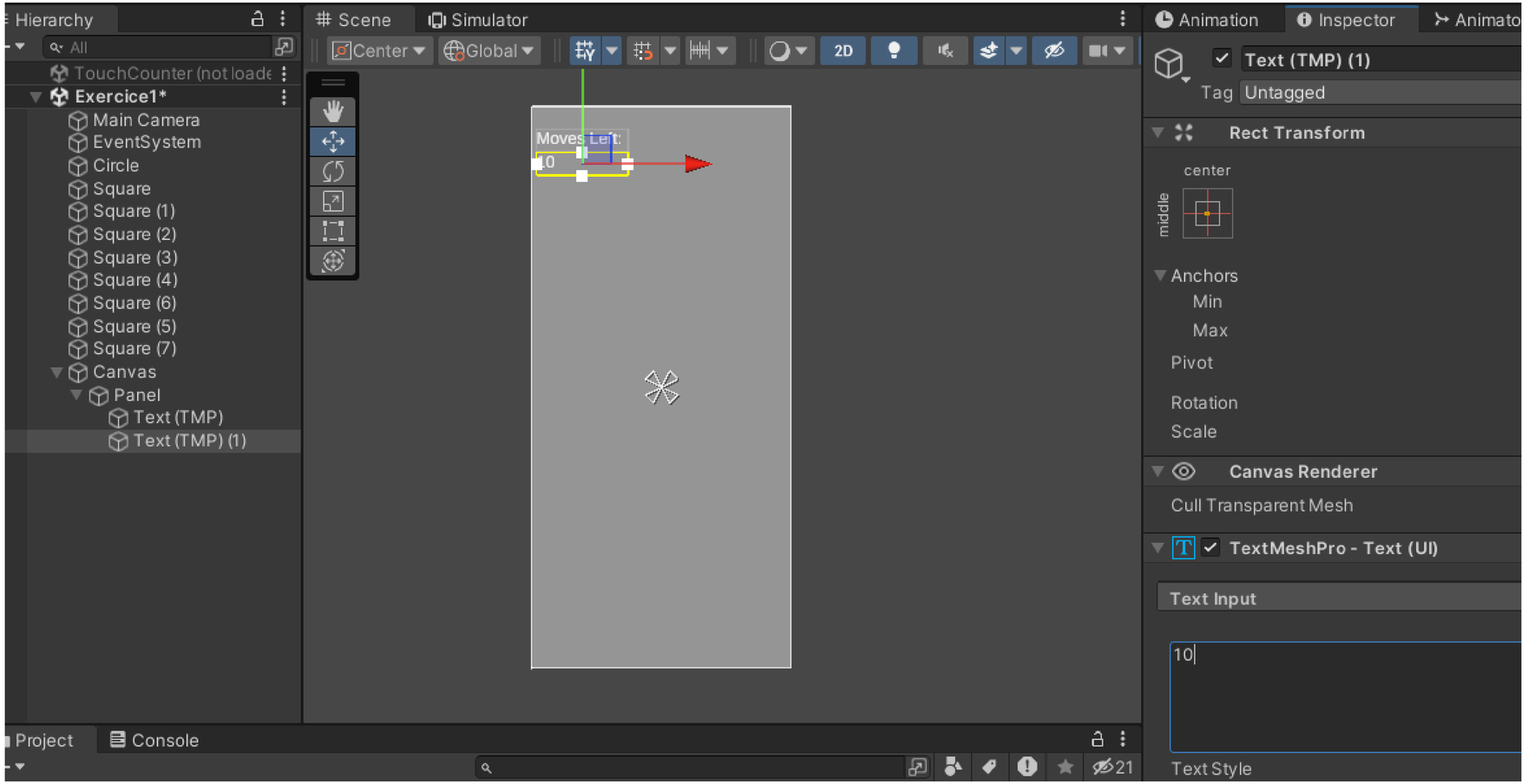
# DEACTIVATE THE IMAGE COMPONENT OF THE PANEL (OR REMOVE IT)



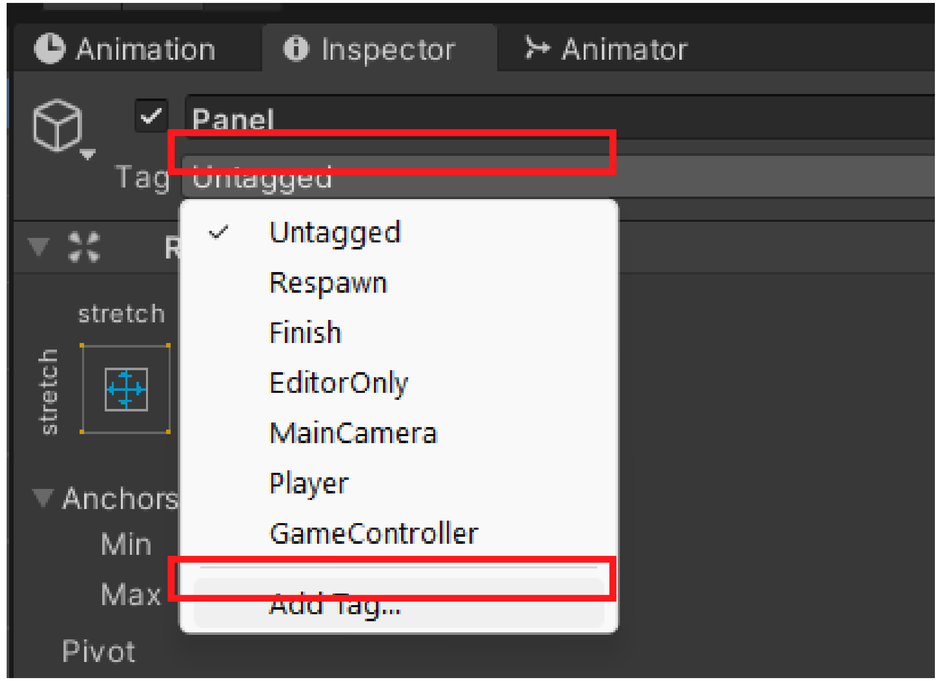
# IMPORT THE TEXT MESH PRO RESOURCES

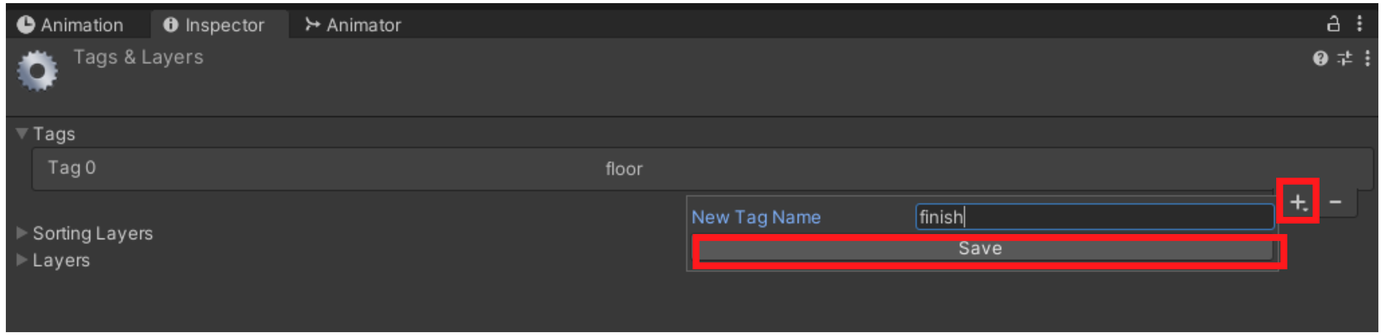


# CHANGE THE TEXT OF THE TEXT BOXES TO ‘MOVES LEFT:’ AND ‘10’

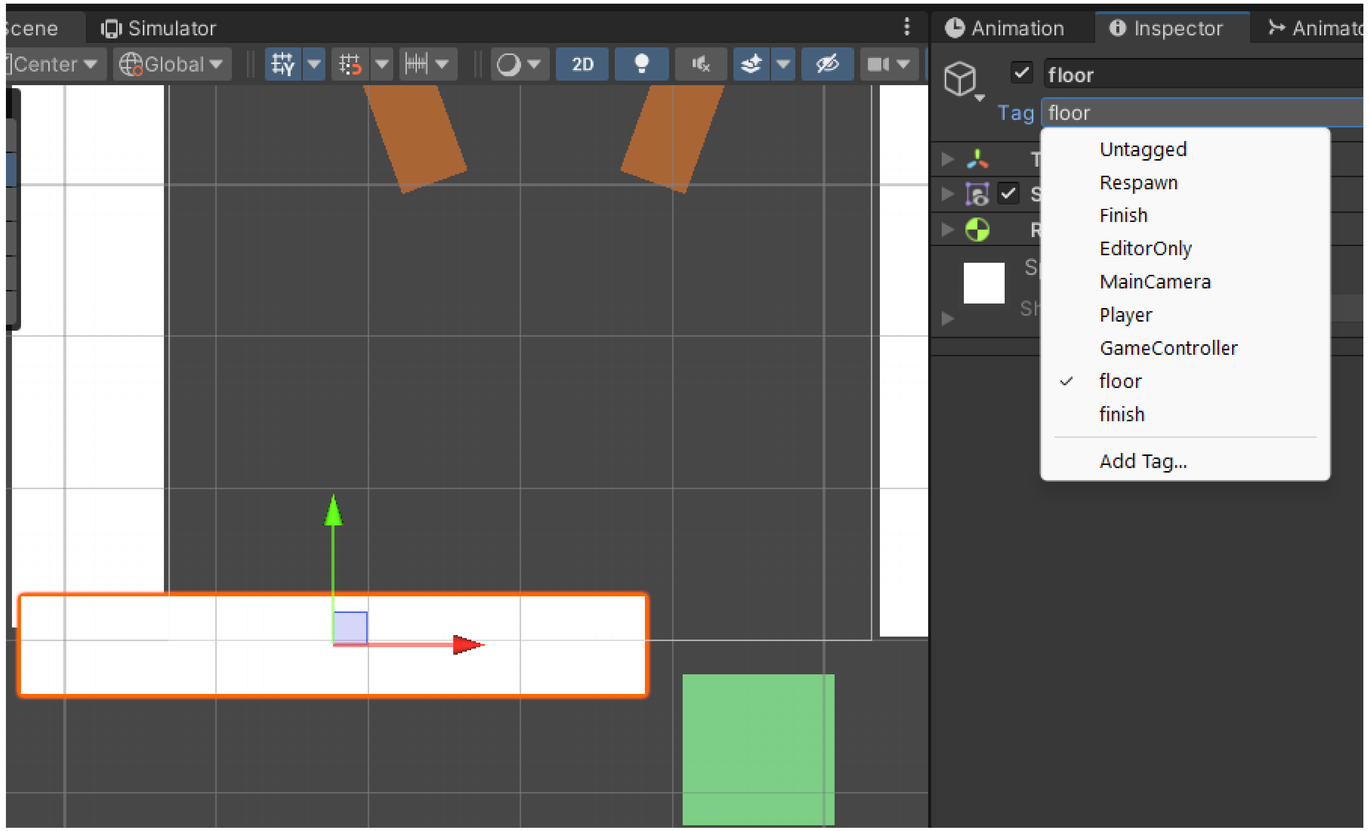


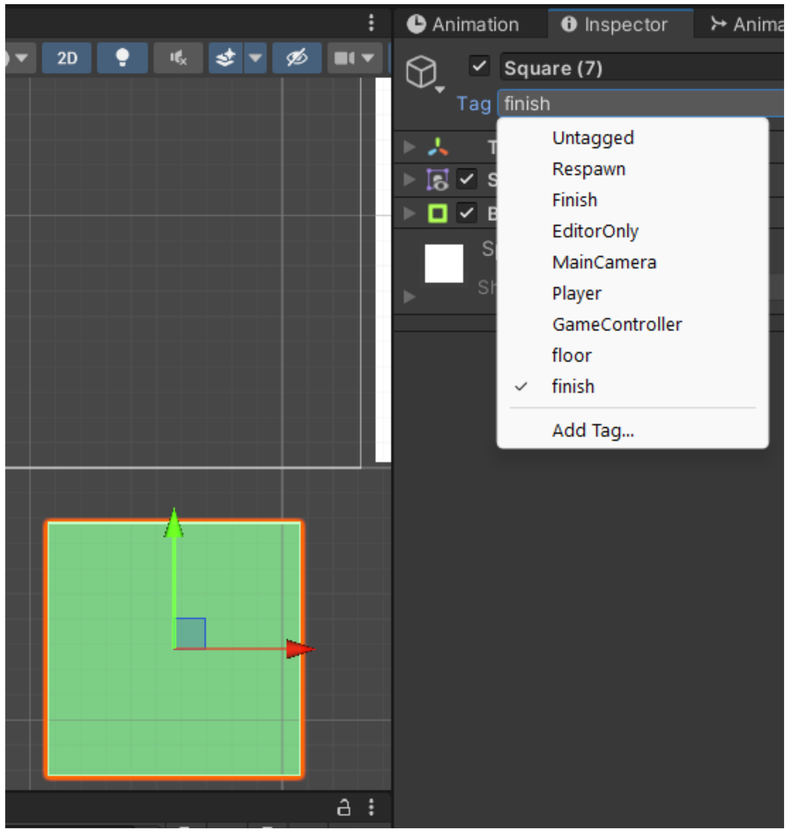
# ADD THE TAGS: ‘FLOOR’ AND ‘FINISH’

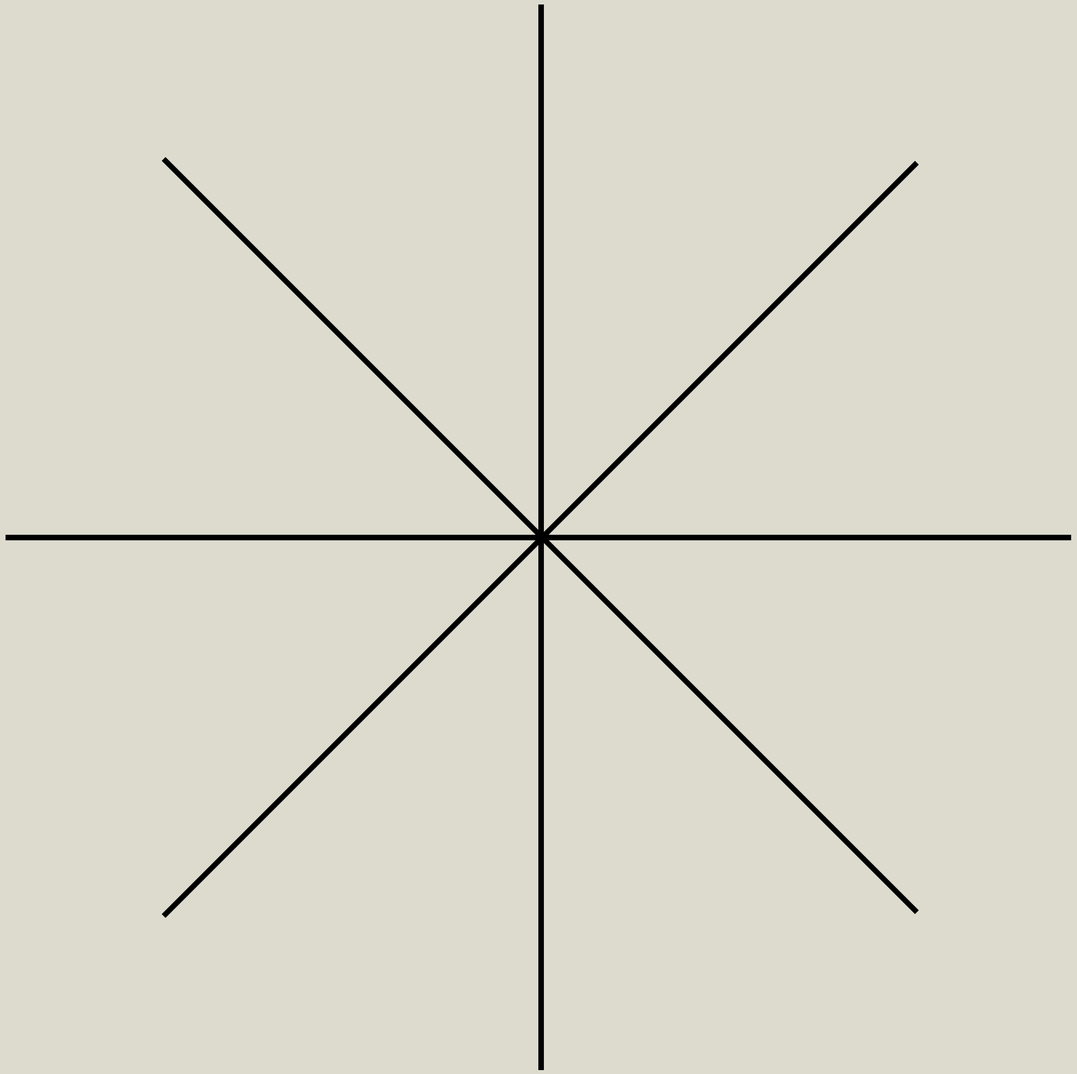


# APLY EACH TAG TO IT'S CORRESPONDING ELEMENT, THE GREEN SQUARE AND THE FLOOR RESPECTIVELY



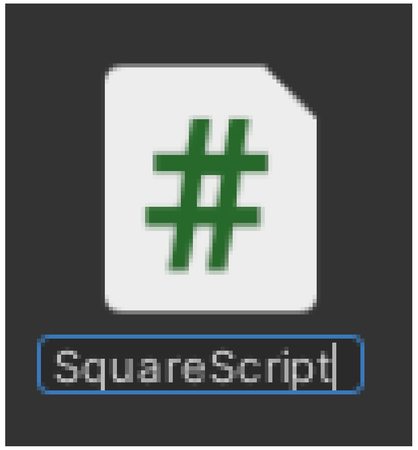



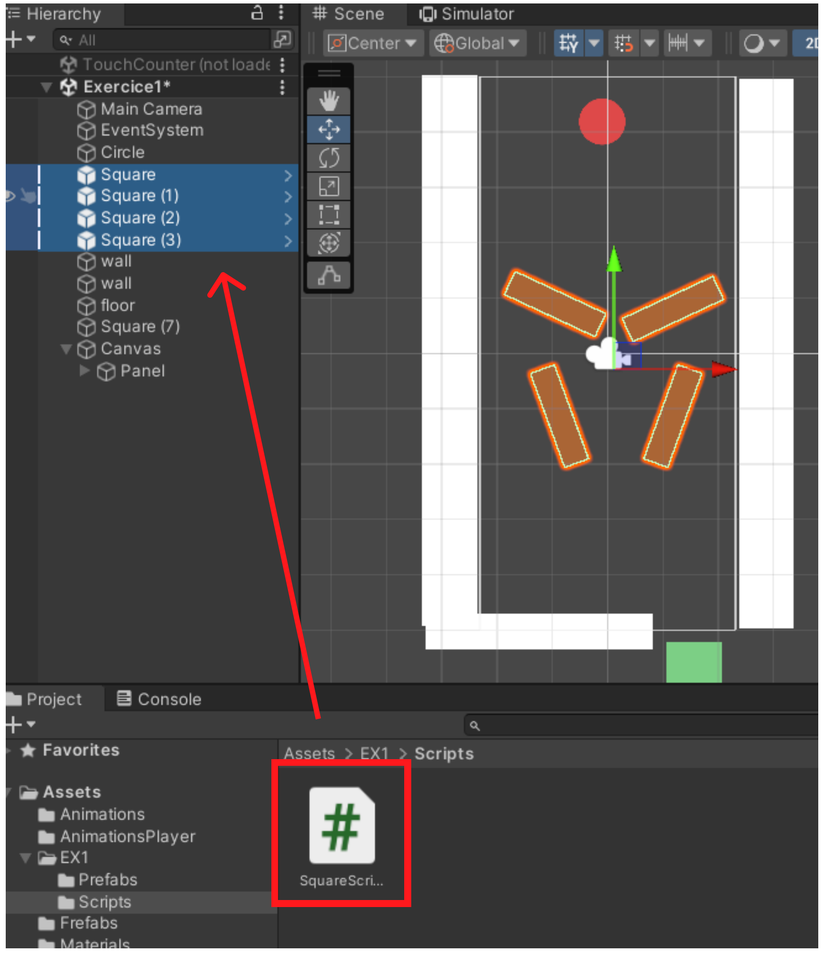


# 03
# List of Prefabs

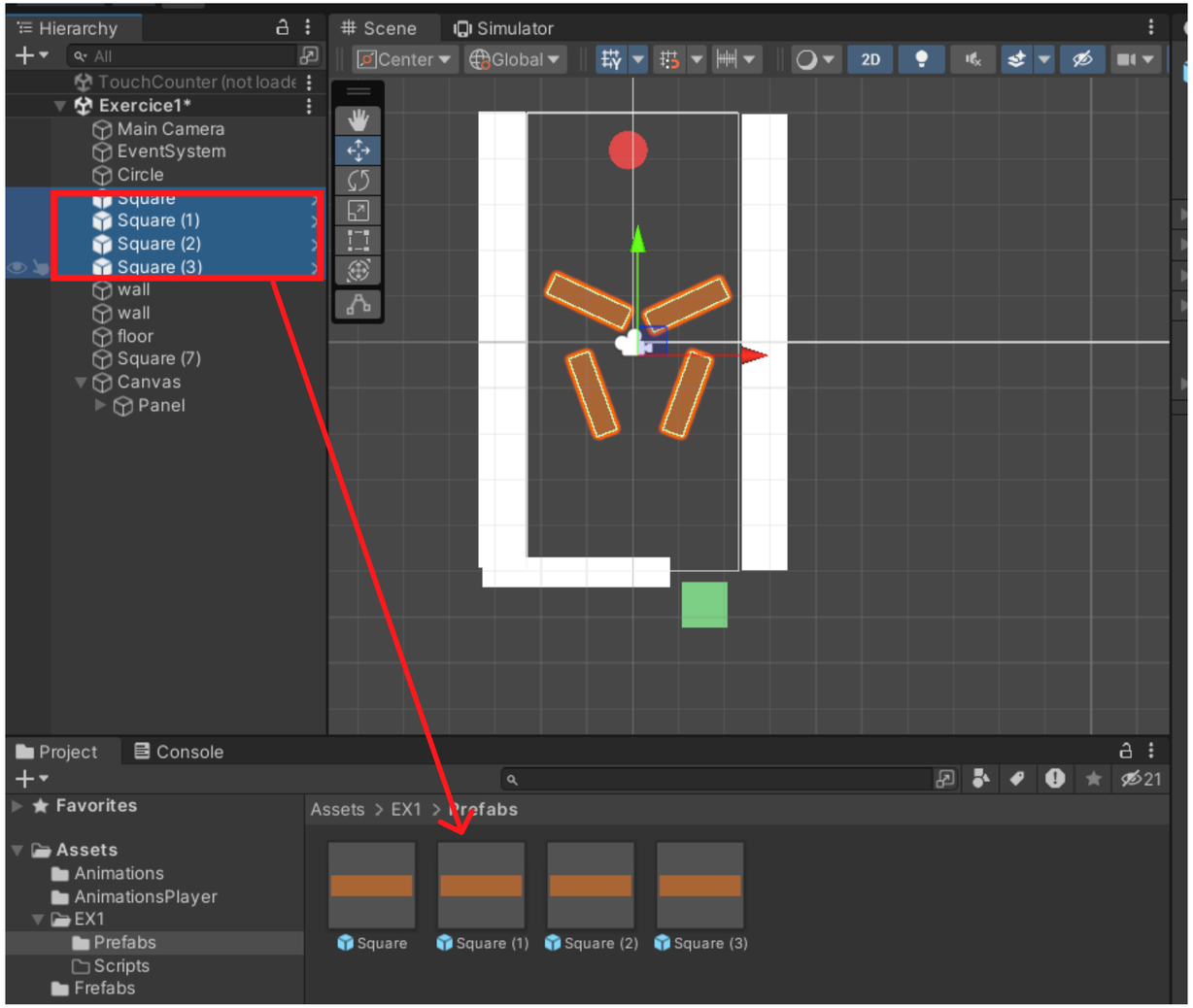
DANIELA MONTEIRO

# CREATE A SCRIPT CALLED ‘SQUARESCRIPT’ AND APPLY IT TO EACH OF THE FOUR RECTANGLES

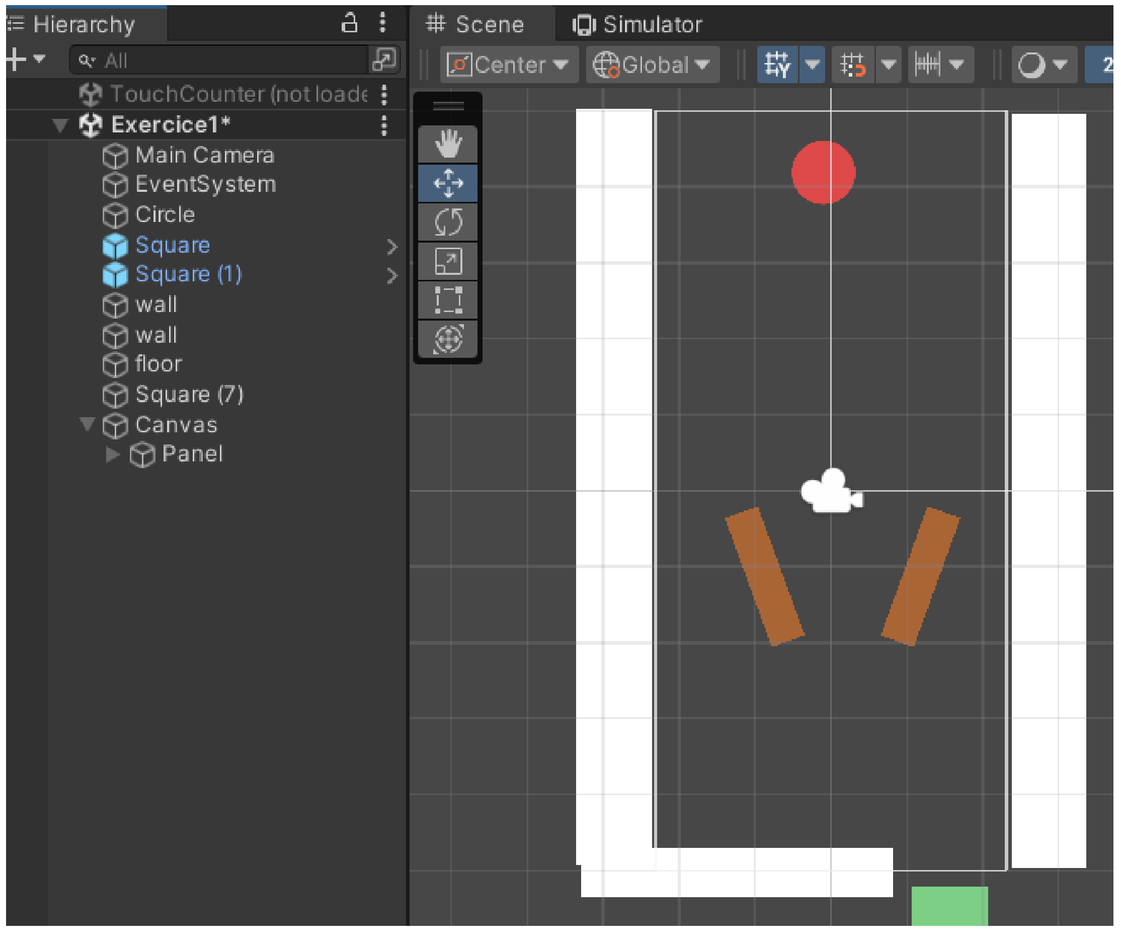


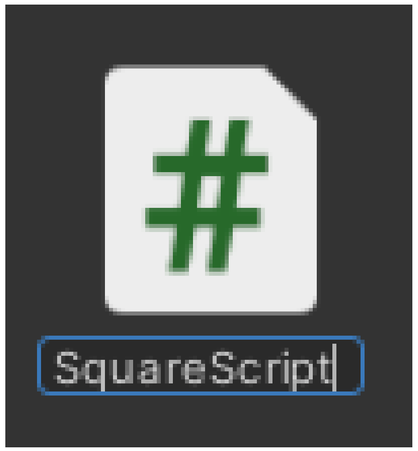
# DRAG THE RECTANGLES TO YOUR ‘PREFABS’ FOLDER



# DELETE TWO OF THE RECTANGLES ON SCENE

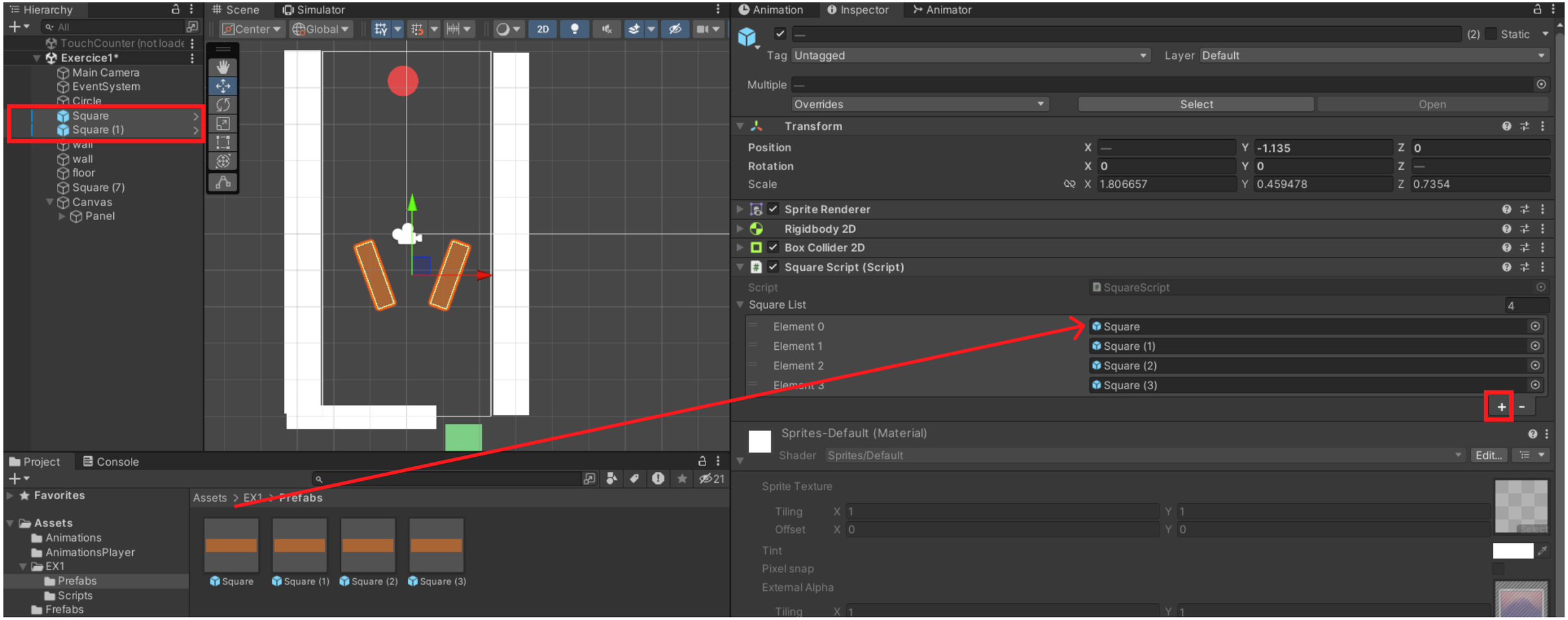


# OPEN THE SCRIPT ‘SQUARESCRIPT’ AND CREATE A LIST OF OBJECTS




```
public class SquareScript : MonoBehaviour
{
    //list of objects
    [SerializeField] public List<GameObject> SquareList = new List<GameObject>();
```

# ADD EACH  OF THE PREFABS TO THE LIST



# DETECT A TOUCH IN EACH RECTANGLE

## !!! IMPORTANT NOTE: ALWAYS START A DETECTION OF A TOUCH WITH THIS CONDITION

```
void Update()
{
    if (Input.touchCount > 0)
    {

    }
```

# DETECT A TOUCH IN EACH RECTANGLE: VARIABLES FOR THE TOUCH AND IT'S POSITION

```
if (Input.touchCount > 0)
{
    //saves the current touch
    Touch touch = Input.GetTouch(0);
    //saves the position of the touch in relation to the world
    Vector3 touchPos = Camera.main.ScreenToWorldPoint(touch.position);
```

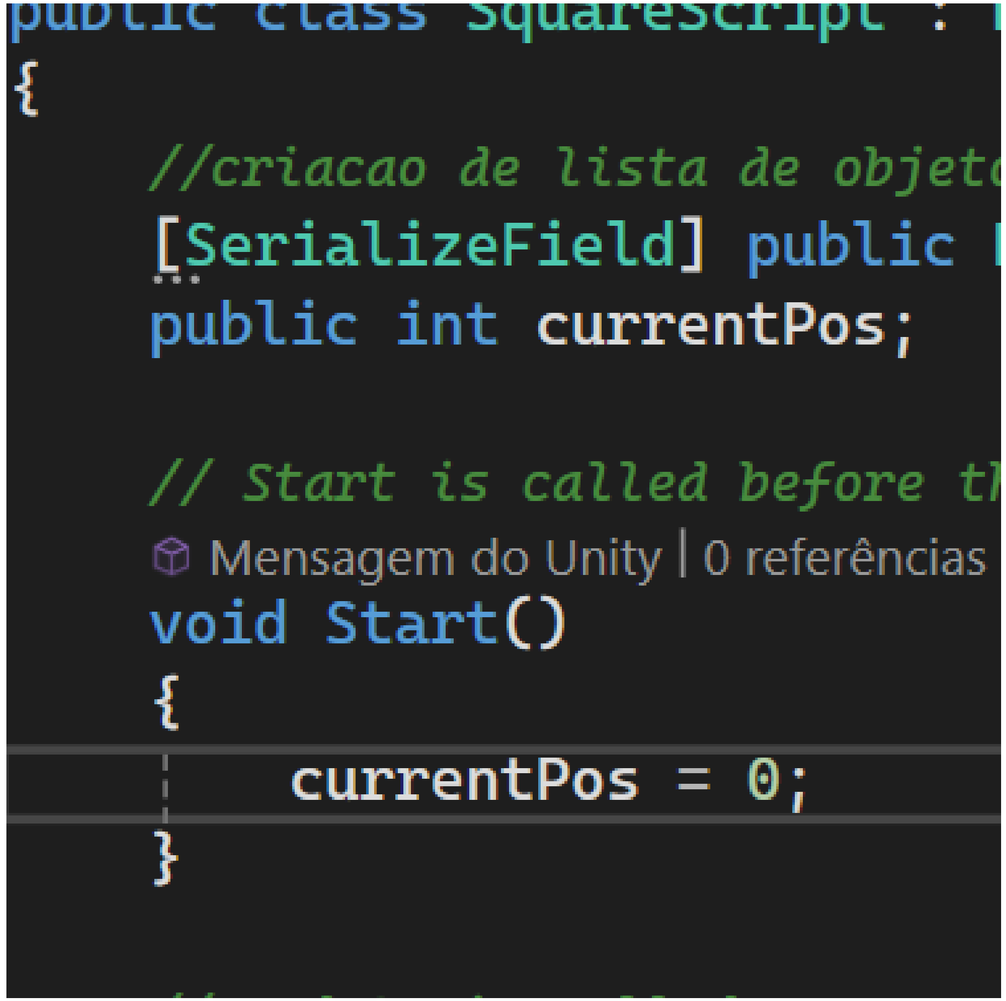
# DETECT A TOUCH IN EACH RECTANGLE: VERIFY IF THE POSITION OF THE TOUCH IN RELATION TO THE RECTANGLE

```
if (Input.touchCount > 0)
{
    //saves the current touch
    Touch touch = Input.GetTouch(0);
    //saves the position of the touch in relation to the world
    Vector3 touchPos = Camera.main.ScreenToWorldPoint(touch.position);

    //a simple touch (as soon as it beggins)
    if (touch.phase == TouchPhase.Began)
    {
        if (GetComponent<Collider2D>() == Physics2D.OverlapPoint(touchPos))
        {
            print("you are touching the square!");
        }
    }
}
```

# CHANGE THE RECTANGLE'S ROTATION TO THE SAME AS THE FIRST ELEMENT OF THE LIST

```
if (GetComponent<Collider2D>() == Physics2D.OverlapPoint(touchPos))
{
    print("you are touching the square!");
    this.gameObject.transform.rotation = SquareList[0].transform.rotation;
}
```

# STORE THE POSITION ON THE LIST (START WITH 0)

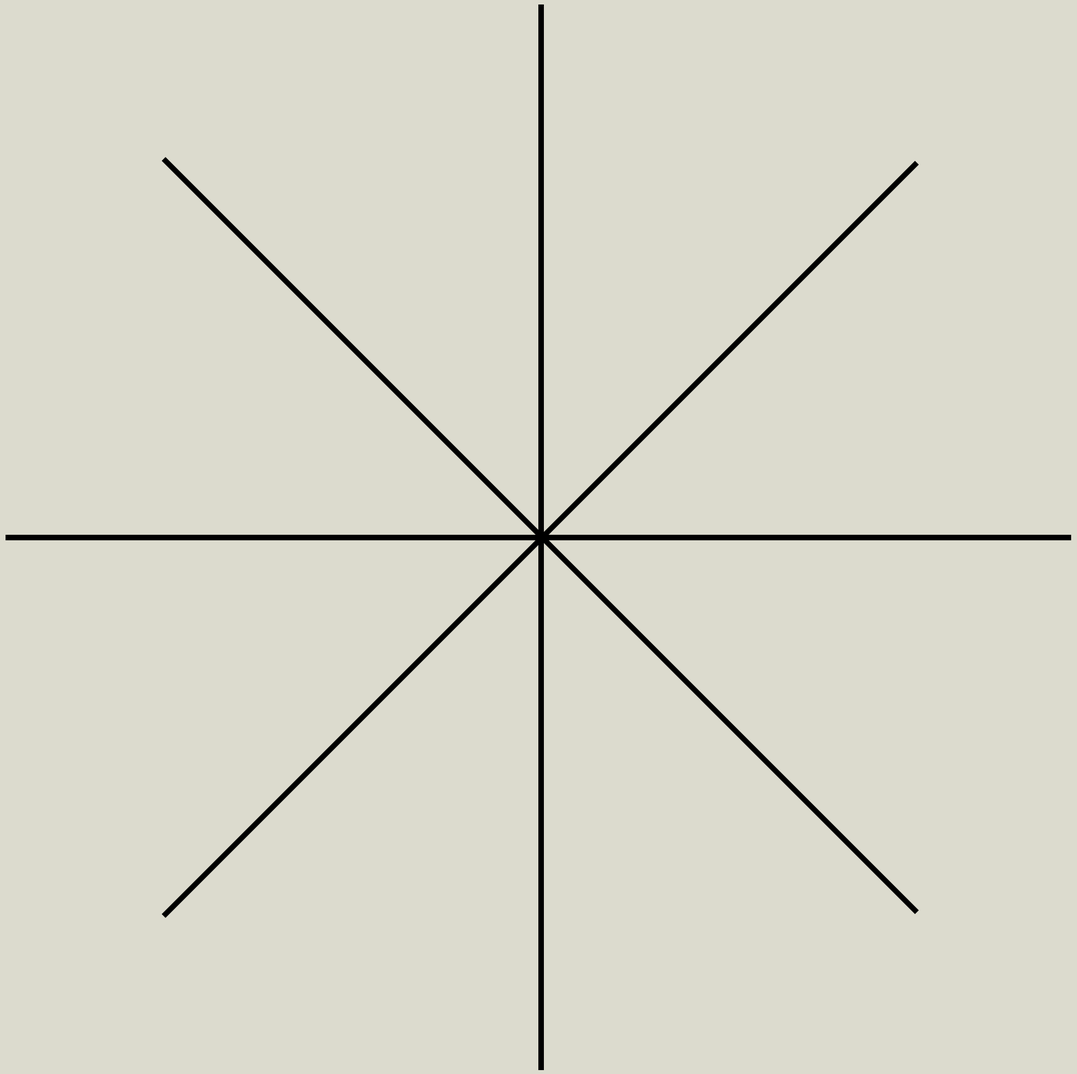


# CHANGE THE ROTATION OF THE RECTANGLE TO GO THROUGH THE ENTIRE LIST

```
if (GetComponent<Collider2D>() == Physics2D.OverlapPoint(touchPos))
{
    print("you are touching the square!");
    this.gameObject.transform.rotation = SquareList[currentPos].transform.rotation;
    currentPos++;
}
```

# WHEN THE POSITION REACHES THE FOURTH ELEMENT, IT SHOULD GO BACK TO THE FIRST

```
if (GetComponent<Collider2D>() == Physics2D.OverlapPoint(touchPos))
{
    print("you are touching the square!");
    if (currentPos > 3)
    {
        currentPos = 0;
    }
    this.gameObject.transform.rotation = SquareList[currentPos].transform.rotation;
    currentPos++;
}
```



# 04
# Drag and Drop

DANIELA MONTEIRO

# CREATE A VARIABLE THAT STORES IF THE TOUCH WAS ON THE RECTANGLE OR NOT

```
public class SquareScript : MonoBehaviour
{
    //list of objects
    [SerializeField] public List<GameObject> SquareL
    public int currentPos;

    //saves whether or not the touch is on the ractangle
    public bool moveAllowed = false;
```

# ACTIVATE THE VARIABLE WITH THE TOUCH ON THE RECTANGLE

```
//a simple touch (as soon as it beggins)
if (touch.phase == TouchPhase.Began)
{
    if (GetComponent<Collider2D>() == Physics2D.
    {
        print("you are touching the square!");
        moveAllowed = true;
        if (currentPos > 3)
        {
```

# DETECT THE MOVEMENT

```
//a simple touch (as soon as it beggins)
if (touch.phase == TouchPhase.Began)
{
    if (GetComponent<Collider2D>() == Ph
}
//for drag
if (touch.phase == TouchPhase.Moved)
{

}
```

# ADD A CONDITION WITH THE VARIABLE

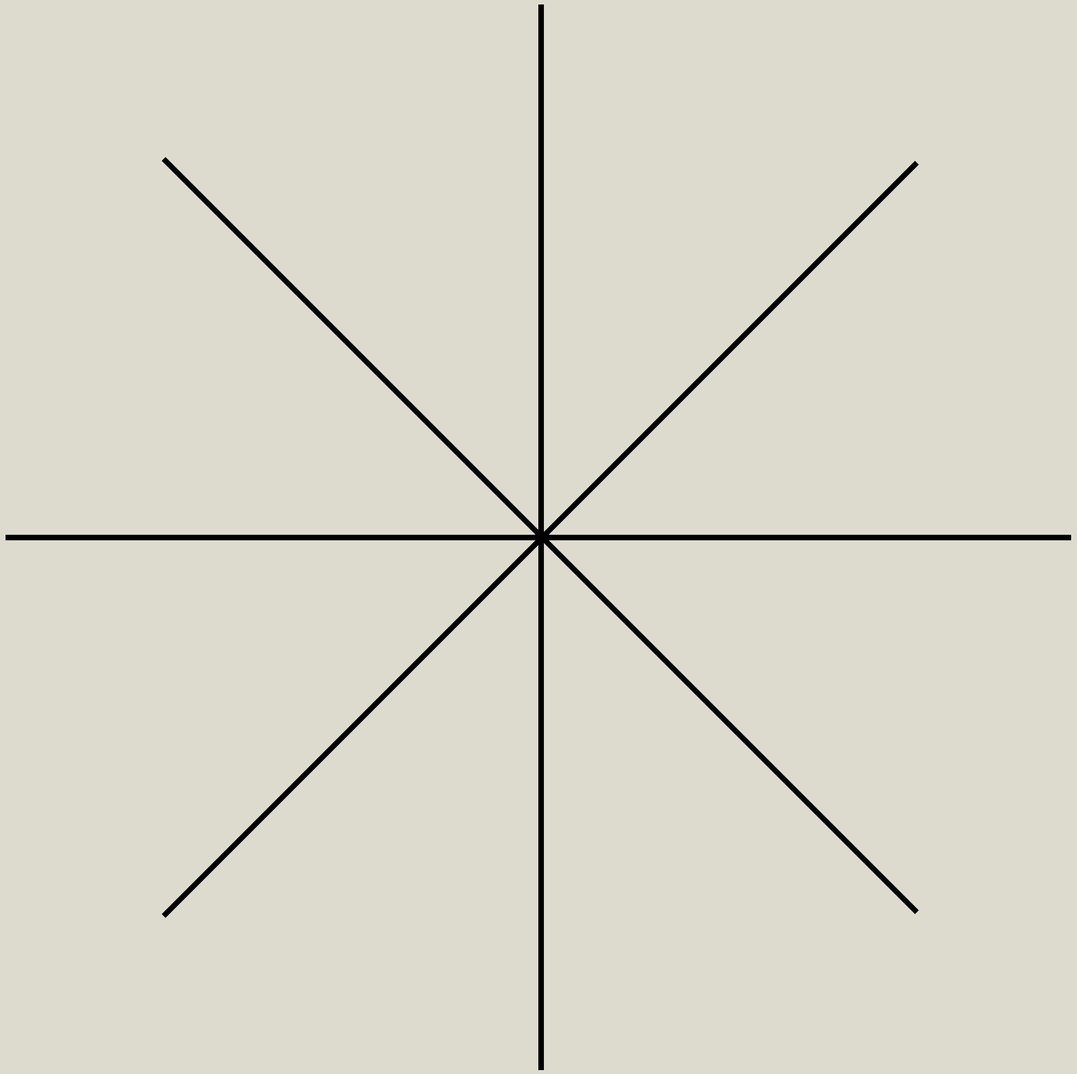
```
//for drag
if (touch.phase == TouchPhase.Moved && moveAllowed)
{

}
```

# CHANGE THE POSITION TO FOLLOW THE TOUCH

```
//for drag
if (touch.phase == TouchPhase.Moved && moveAllowed)
{
    this.transform.position =
        Camera.main.ScreenToWorldPoint(new Vector3(touch.position.x, touch.position.y, 0));
}
```

# DEACTIVATE THE VARIABLE AT THE END OF THE TOUCH

```
//for drag
if (touch.phase == TouchPhase.Moved && moveAllowed)
{
    this.transform.position =
        Camera.main.ScreenToWorldPoint(new Vector3(tou
}
//for the drop
if (touch.phase == TouchPhase.Ended && moveAllowed)
{
    moveAllowed = false;
}
```



# 05
# Moves Counter

DANIELA MONTEIRO

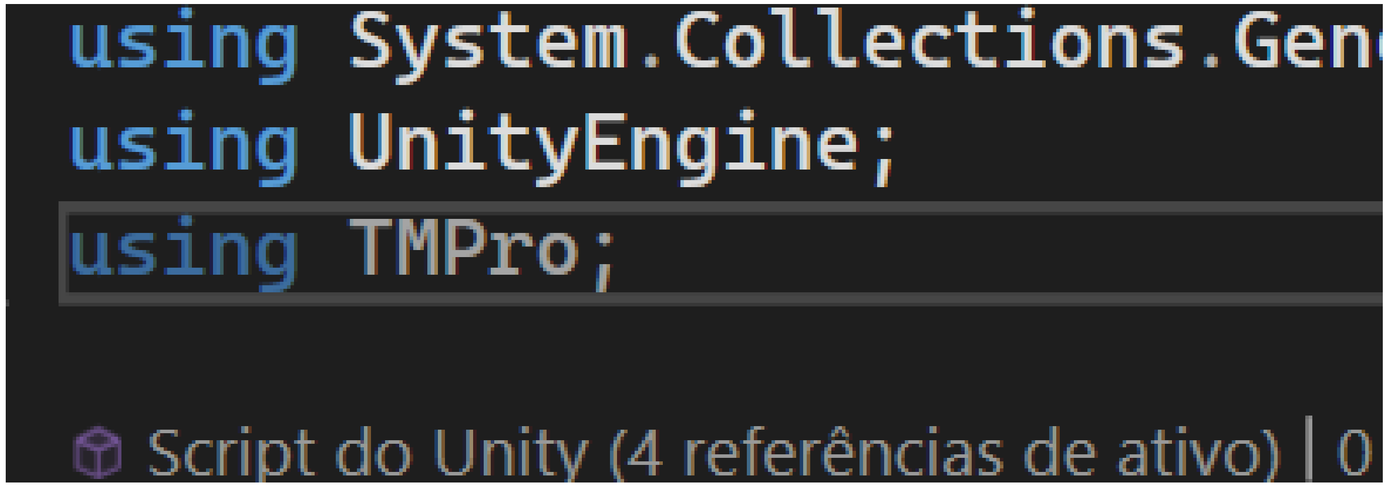
# ADD A VARIABLE THAT STORES THE AMMOUNT OF MOVES LEFT

```
public class SquareScript : MonoBehaviour
{
    //list of objects
    [SerializeField] public List<GameObject
    public int currentPos;

    public int movesLeft = 10;

    // Start is called before the first frame up
```

# DECREASE THE VARIABLE AT THE END OF A TOUCH

```
//for the drop
if (touch.phase == TouchPhase.Ended && moveAllowed)
{
    movesLeft--;
    moveAllowed = false;
}
```

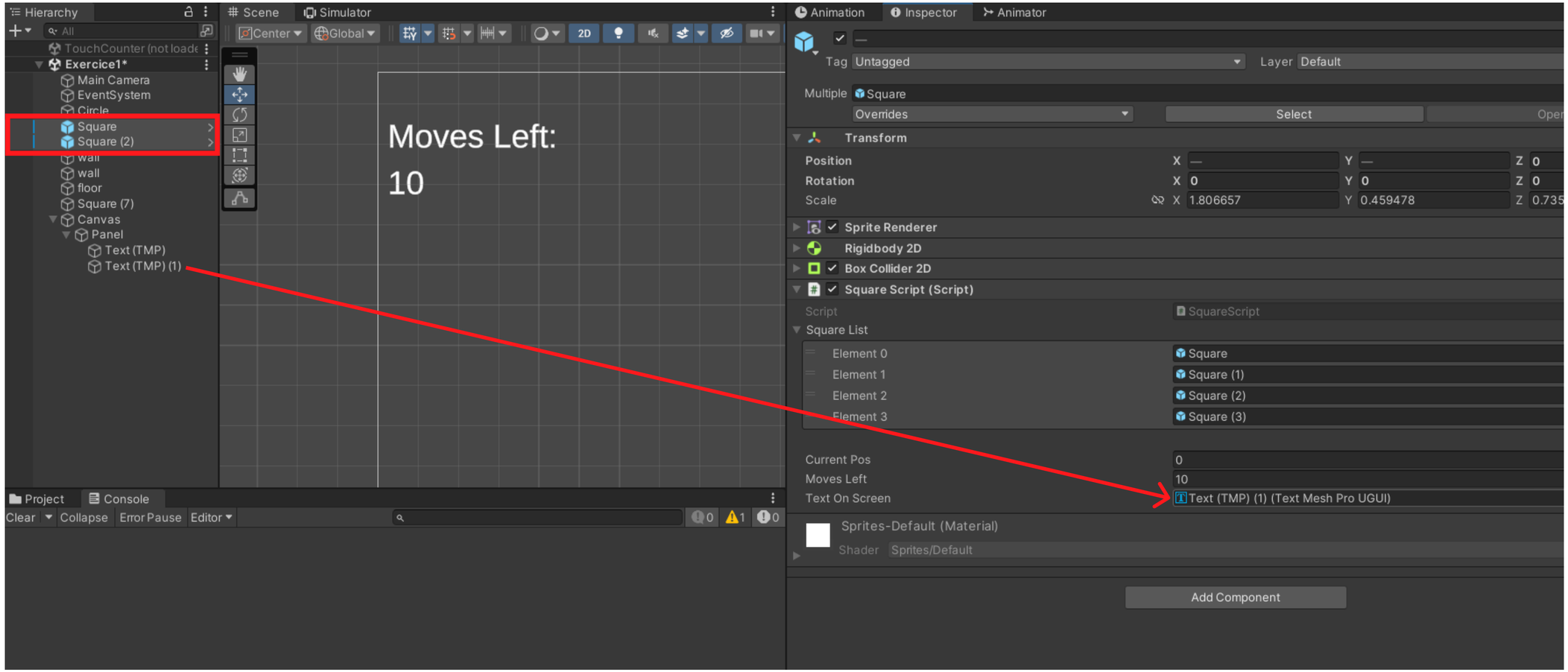
# IMPORT THE LIBRARY TMPRO



# CREATE A VARIABLE FOR THE TEXT ON THE PANEL

```
//saves the ammount of moves left
public int movesLeft = 10;
//for the text on screen to change
[SerializeField] public TextMeshProUGUI textOnScreen;

// Start is called before the first frame update
```

# ATRIBUTE THE TEXT OBJECT TO THE VARIBLE FROM THE EDITOR

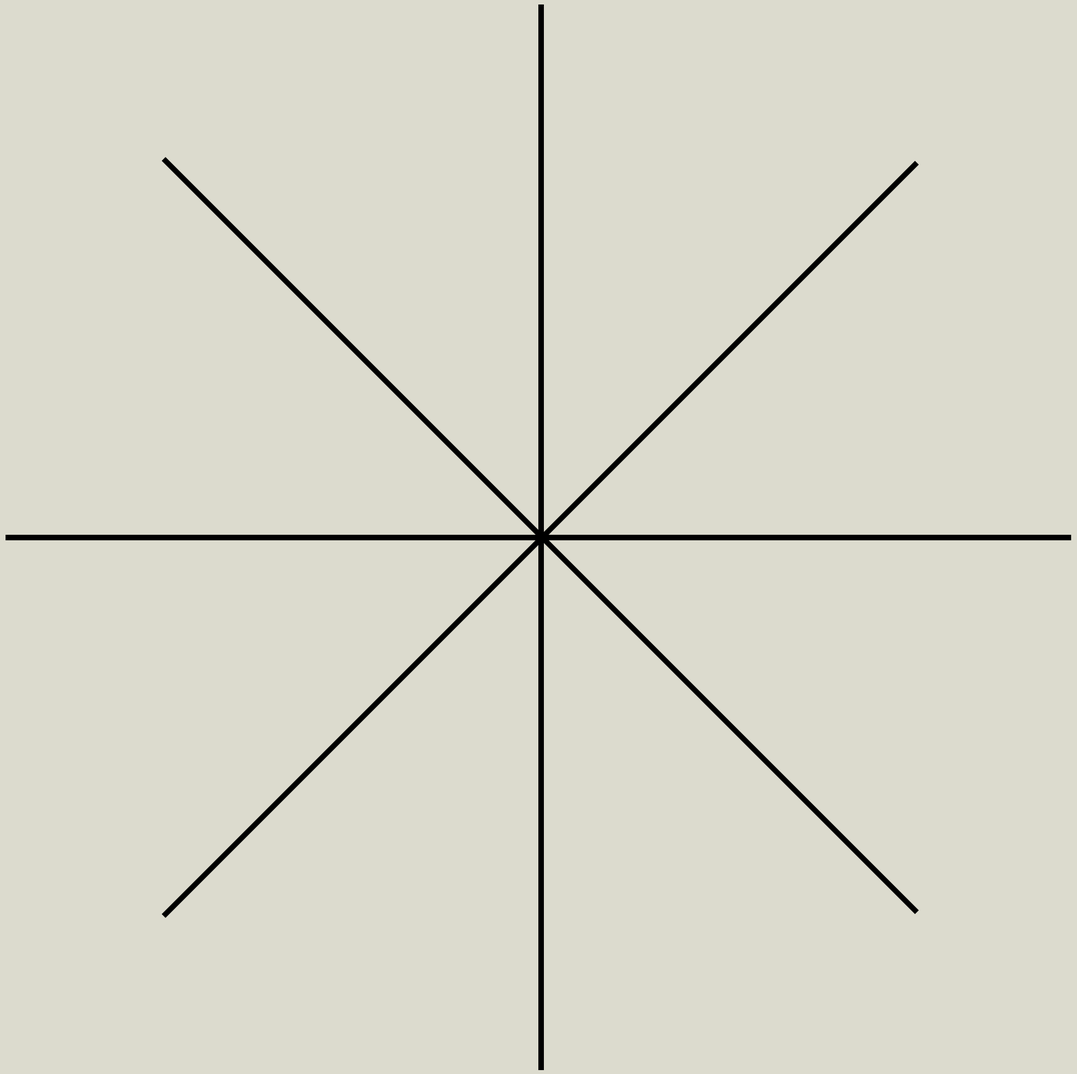


# CHANGE THE TEXT AT THE END OF EACH TOUCH

```
//for the drop
if (touch.phase == TouchPhase.Ended && moveAllowed)
{
    movesLeft--;
    moveAllowed = false;
    textOnScreen.text = movesLeft.ToString();
}
```

# ADD A CONDITION TO THE PHASE ‘MOVED’ SO THE MOVES DON’T SURPASS 10

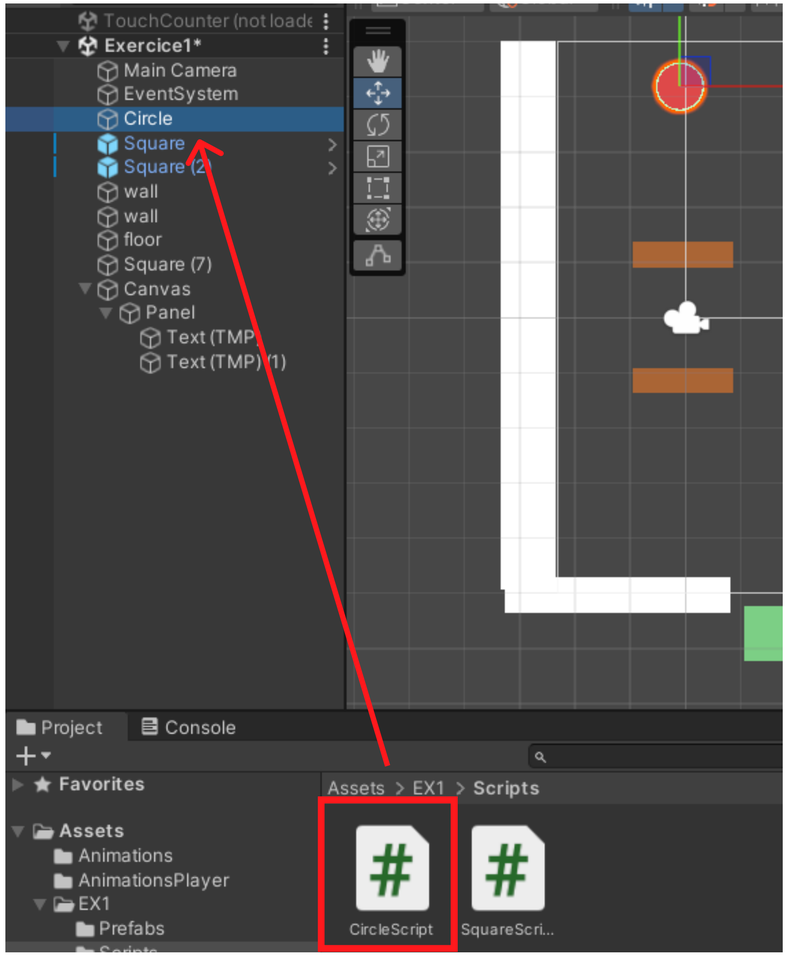
```
//for drag
if (touch.phase == TouchPhase.Moved && moveAllowed && movesLeft > 0)
{
    this.transform.position =
        Camera.main.ScreenToWorldPoint(new Vector3(touch.position.x, touch.position.y, 0));
}
//for the drop
```



# 06
# End of Game

DANIELA MONTEIRO

# CREATE A SCRIPT CALLED ‘CIRCLESCRIPT’ AND APPLY IT TO THE CIRCLE

# DETECT THE TOUCH ON THE CIRCLE, THE SAME WAY AS FOR THE RECTANGLES

```
Mensagem do Unity | 0 referências
void Update()
{
    if (Input.touchCount > 0)
    {
        //saves the current touch
        Touch touch = Input.GetTouch(0);
        //saves the position of the touch in relation to the world
        Vector3 touchPos = Camera.main.ScreenToWorldPoint(touch.position);

        //a simple touch (as soon as it beggins)
        if (touch.phase == TouchPhase.Began)
        {
            if (GetComponent<Collider2D>() == Physics2D.OverlapPoint(touchPos))
            {
                print("you are touching the circle!");
            }
        }
    }
}
```

# MAKE THE CIRCLE FALL BY CHANGING THE BODY TYPE OF THE RIGID BODY'S COMPONENT TO DYNAMIC

```
if (GetComponent<Collider2D>() == Physics2D.OverlapPoint(touchPos))
{
    print("you are touching the circle!");
    GetComponent<Rigidbody2D>().bodyType = RigidbodyType2D.Dynamic;
}
```

# CREATE A METHOD FOR THE END OF THE GAME

```
void Update()
{
    if (Input.touchCount > 0)...
}

0 referências
public void EndOfGame(string state)
{
    if (state == "win")
    {
        print("You Won!");
    }
    else
    {
        print("You Lost!");
    }
}
```

# DETECT THE COLISION BETWEEN THE CIRCLE AND THE GREEN SQUARE (FINISH LINE)

```
public void EndOfGame(string state)
{
    if (state == "win")...
    else...
    panelUI.SetActive(true);
    Time.timeScale = 0;
}


Mensagem do Unity | 0 referências
private void OnCollisionEnter2D(Collision2D collision)
{
    if (collision.gameObject.tag == "finish")
    {
        EndOfGame("win");
    }

}
```

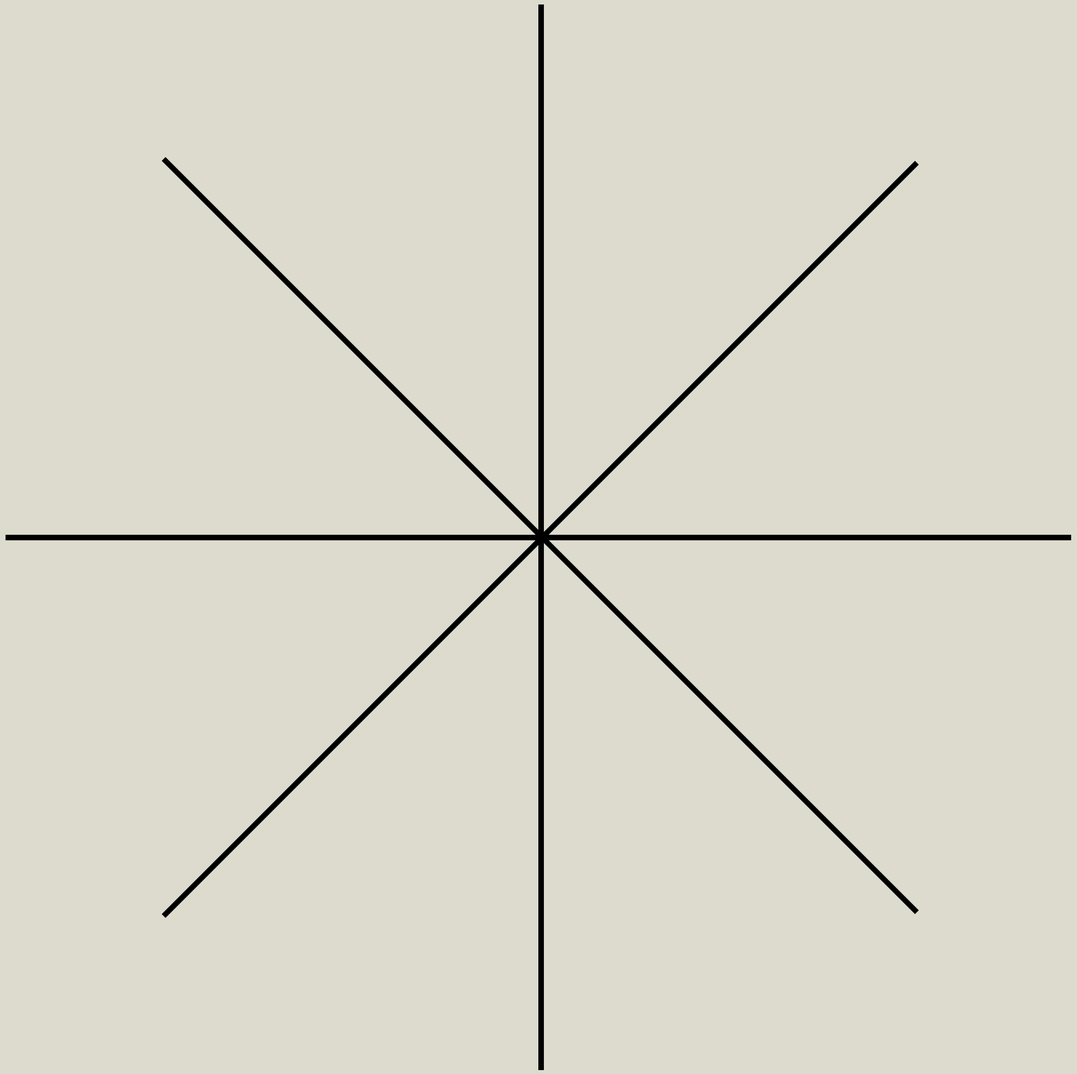
# DETECT THE COLLISION BETWEEN THE CIRCLE AND THE FLOOR

```
private void OnCollisionEnter2D(Collision2D collision)
{
    if (collision.gameObject.tag == "finish")
    {
        EndOfGame("win");
    }
    else if (collision.gameObject.tag == "floor")
    {
        EndOfGame("loose");
    }
}
```

# PAUSE THE GAME

```
2 referências
public void EndOfGame(string state)
{
    if (state == "win")
    {
        print("You Won!");
    }
    else
    {
        print("You Lost!");
    }
    Time.timeScale = 0;
}
```

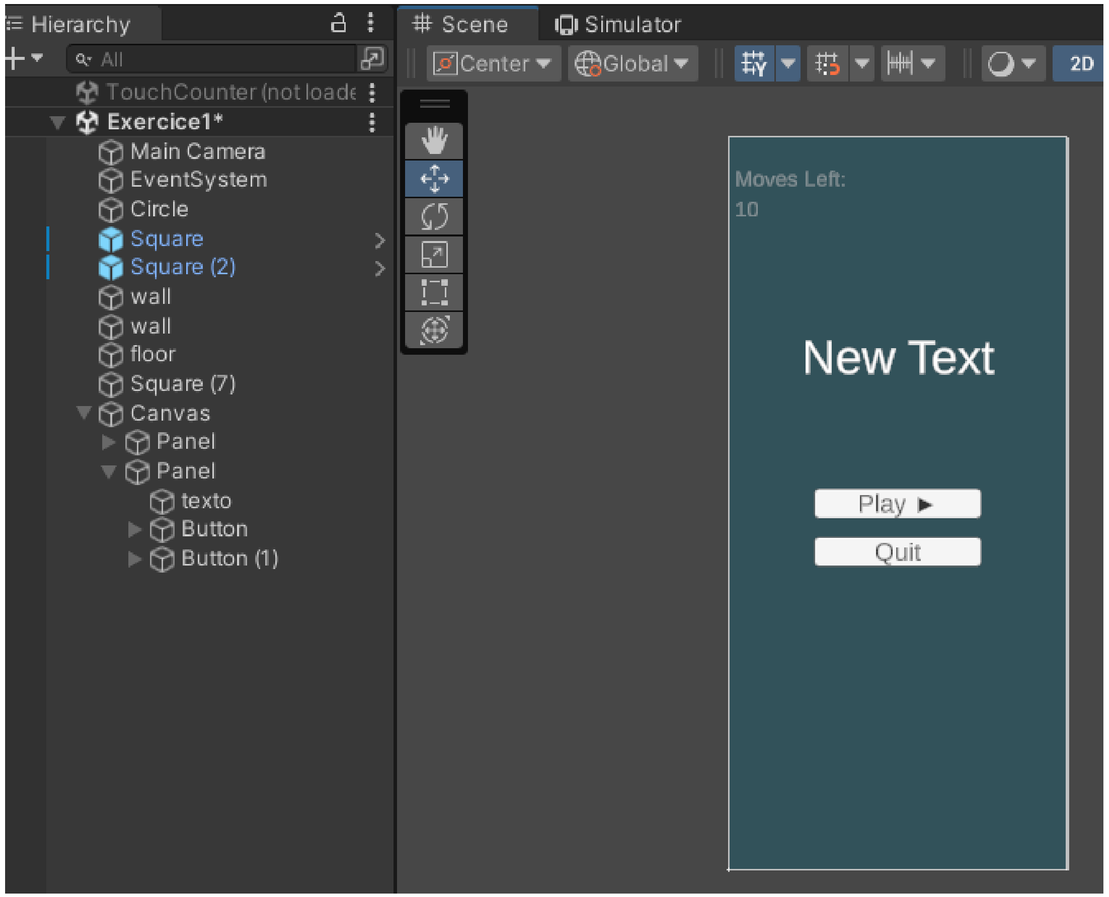
DESENVOLVIMENTO PARA ÂMBIENTES MÓVEIS



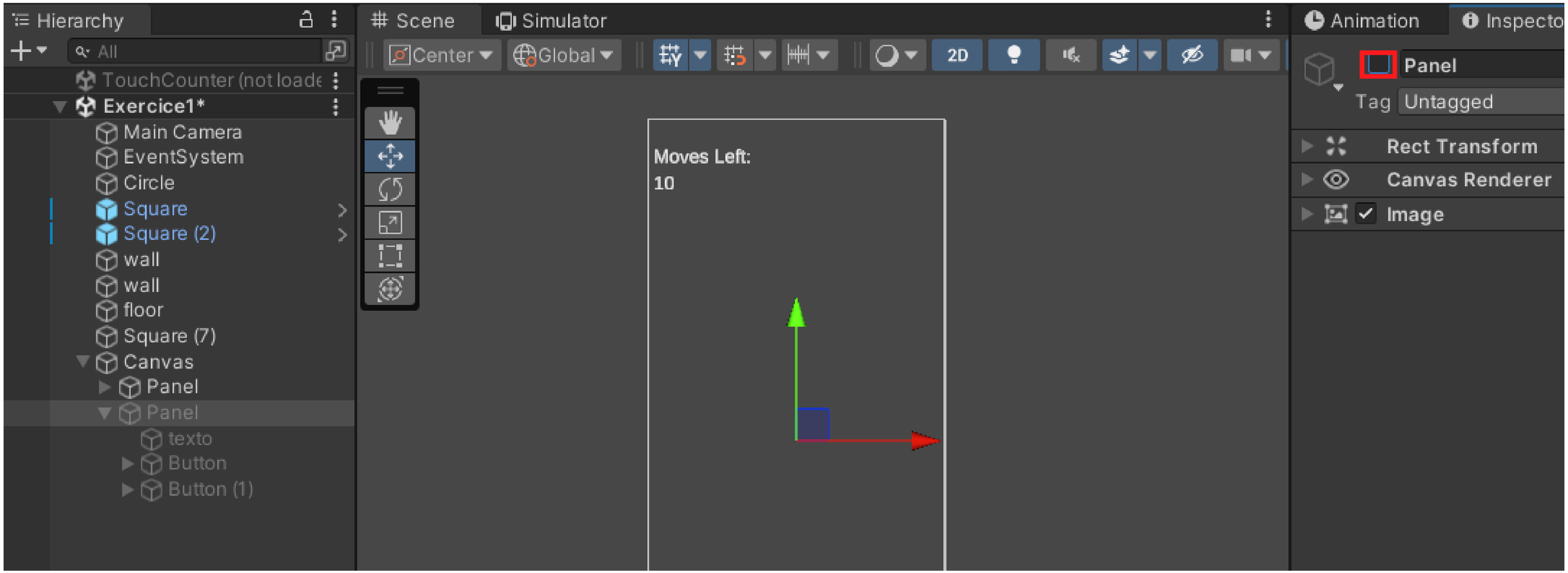
# 08
# Game Menu

DANIELA MONTEIRO

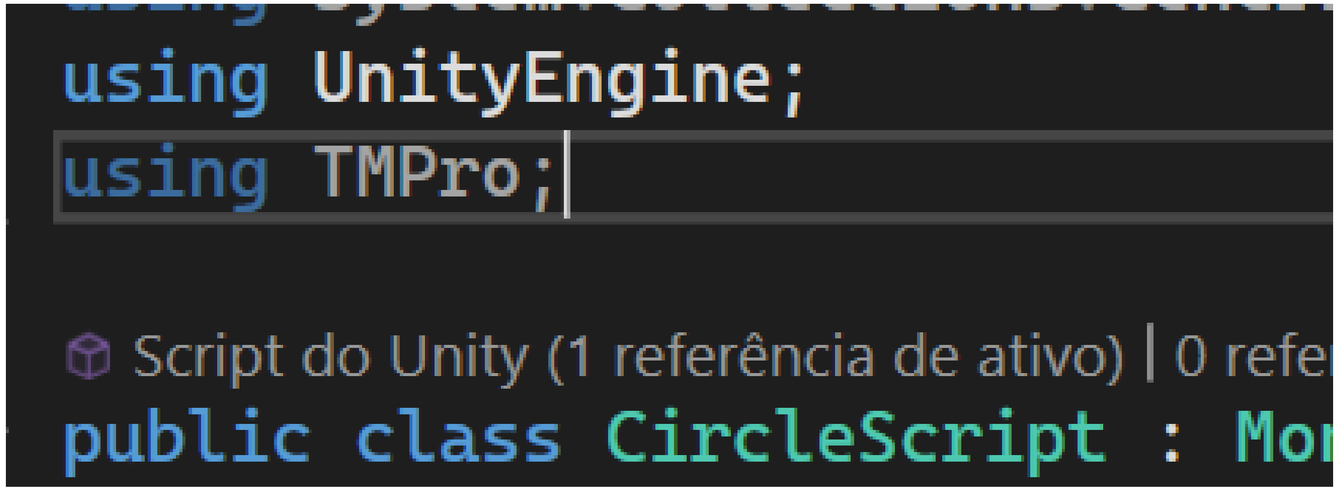
# ADD A PANEL WITH A TEXT BOX AND TWO BUTTONS (PLAY AND QUIT)



# DEACTIVATE THE NEW PANEL



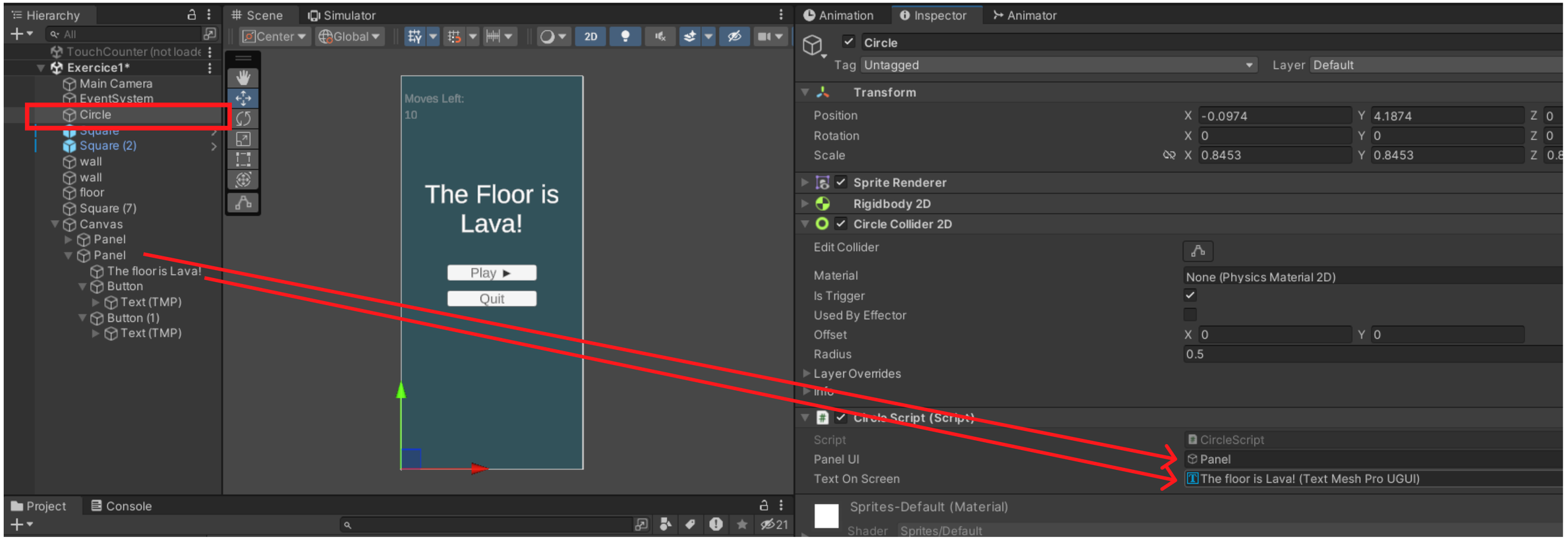
# IMPORT THE LIBRARY ‘TMPRO’ TO THE ‘CIRCLESCRIPT’ SCRIPT



# CREATE TWO VARIBLES FOR THE PANEL AND THE TEXT

```
public class CircleScript : MonoBehaviour
{
    [SerializeField] public GameObject panelUI;
    [SerializeField] public TextMeshProUGUI textOnScreen;

    // Start is called before the first frame update
    Mensagem do Unity | – referências
```

# ASSOCIATE THE VARIBLES TO THEIR ELEMENTS



# ACTIVATE THE PANEL AT THE END OF THE GAME

```
2 referências
public void EndOfGame(string state)
{
    if (state == "win")
    {
        print("You Won!");
    }
    else
    {
        print("You Lost!");
    }
    panelUI.SetActive(true);
    Time.timeScale = 0;
}
```

# CHANGE THE TEXT ON THE PANEL TO MATCH THE STATE OF THE GAME (WON OR LOST)

```
2 referências
public void EndOfGame(string state)
{
    if (state == "win")
    {
        print("You Won!");
        textOnScreen.text = "You Won!";
    }
    else
    {
        print("You Lost!");
        textOnScreen.text = "You Lost!";
    }
    panelUI.SetActive(true);
    Time.timeScale = 0;
}
```

# CREATE A METHOD TO START OR RESTART THE GAME

```
Mensagem do Unity | 0 referências
void Start()
{

}

0 referências
public void RestartGame()
{

}

// Update is called once per frame
Mensagem do Unity | 0 referências
```

# UNPAUSE THE GAME

```
0 referências
public void RestartGame()
{
    Time.timeScale = 1;
}
```

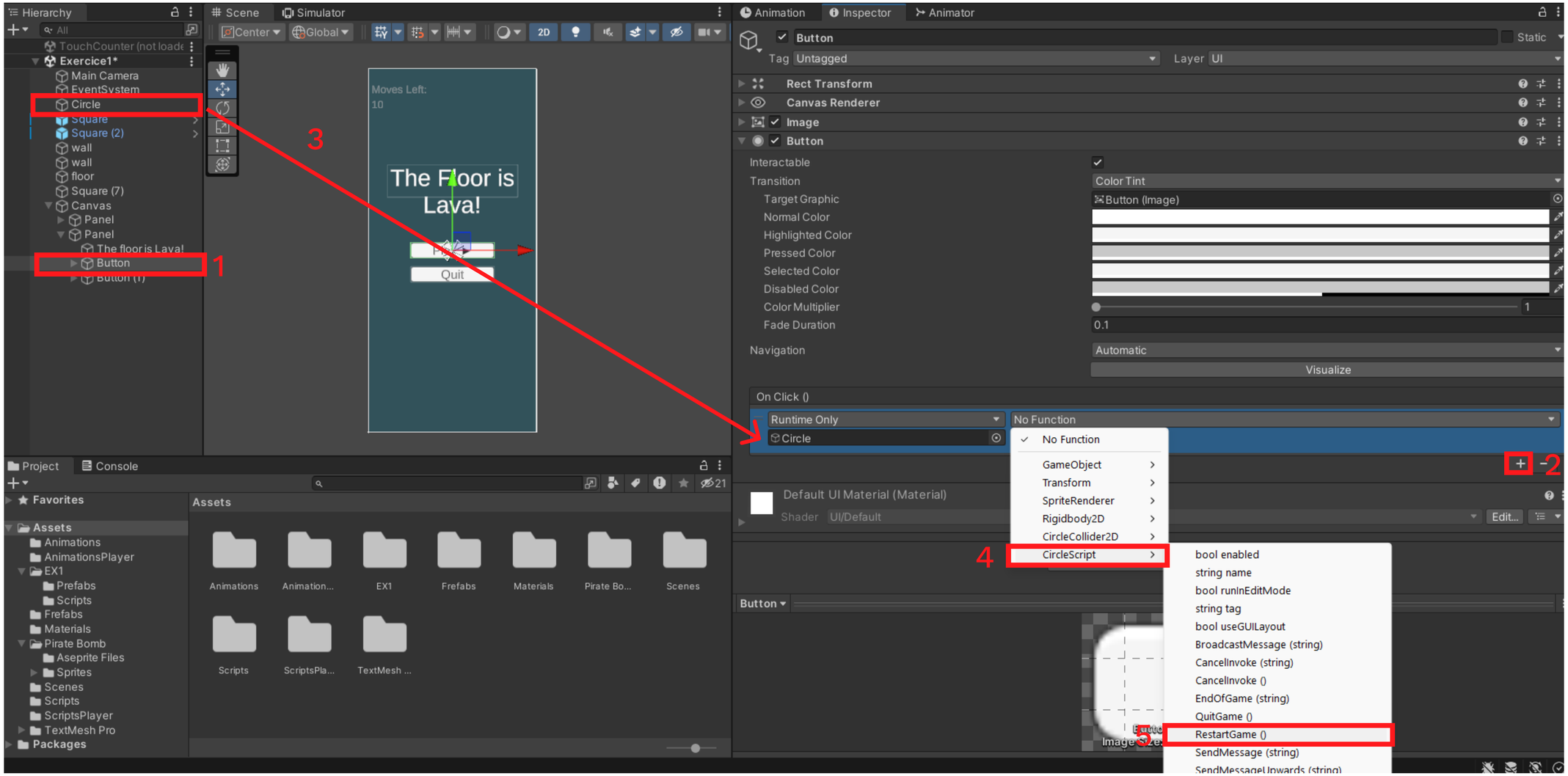
# IMPORT THE LIBRARY FOR SCENE MANAGEMENT

```
using TMPro;
using UnityEngine.SceneManagement;

– referências
public class CircleScript : MonoBehaviour
```

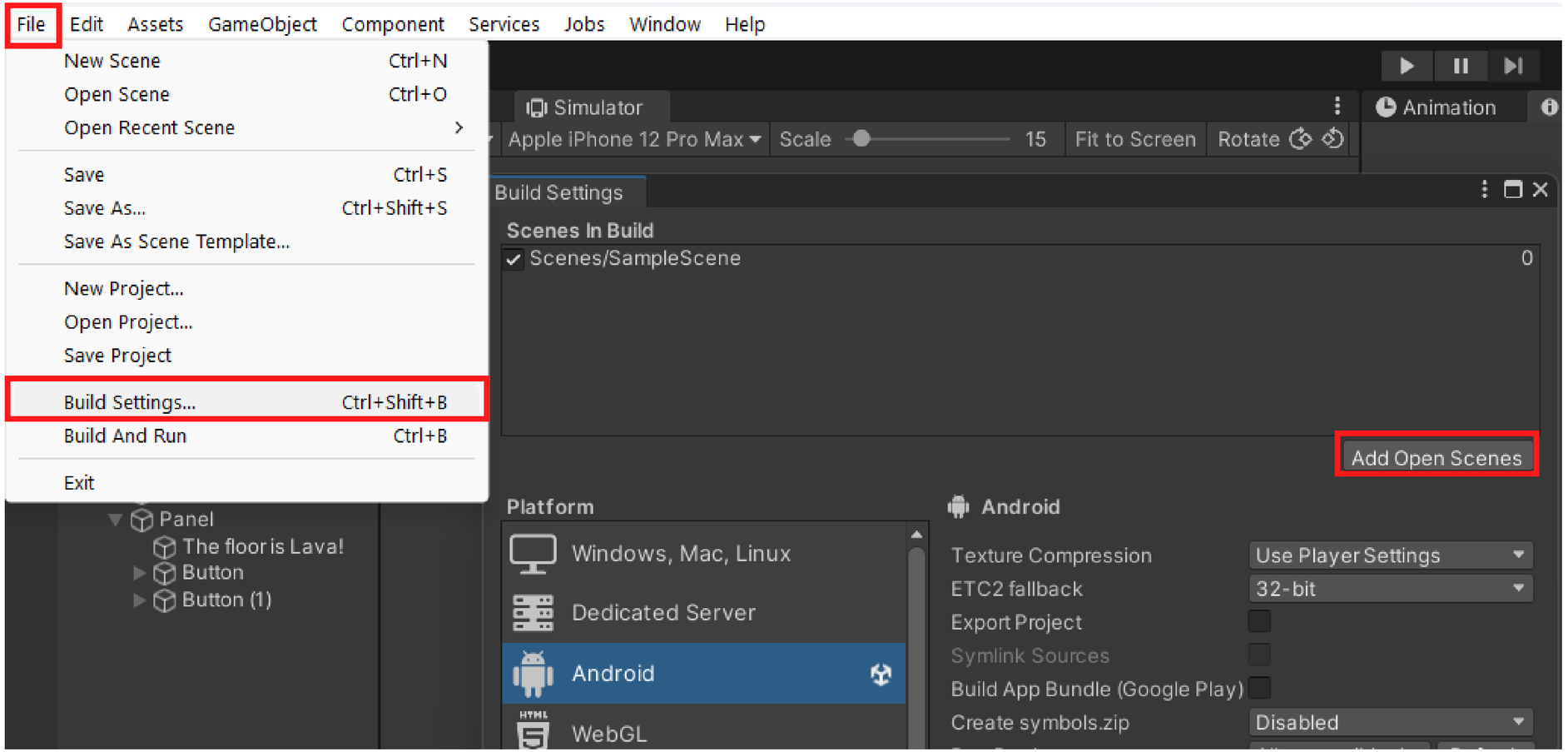
# RESTART THE SCENE

```
0 referências
public void RestartGame()
{
    SceneManager.LoadScene(0);
    //alternative: name of the scene:
    SceneManager.LoadScene("SampleScene");
    Time.timeScale = 1;

}
```

# CONNECT THE FIRST BUTTON TO THE METHOD



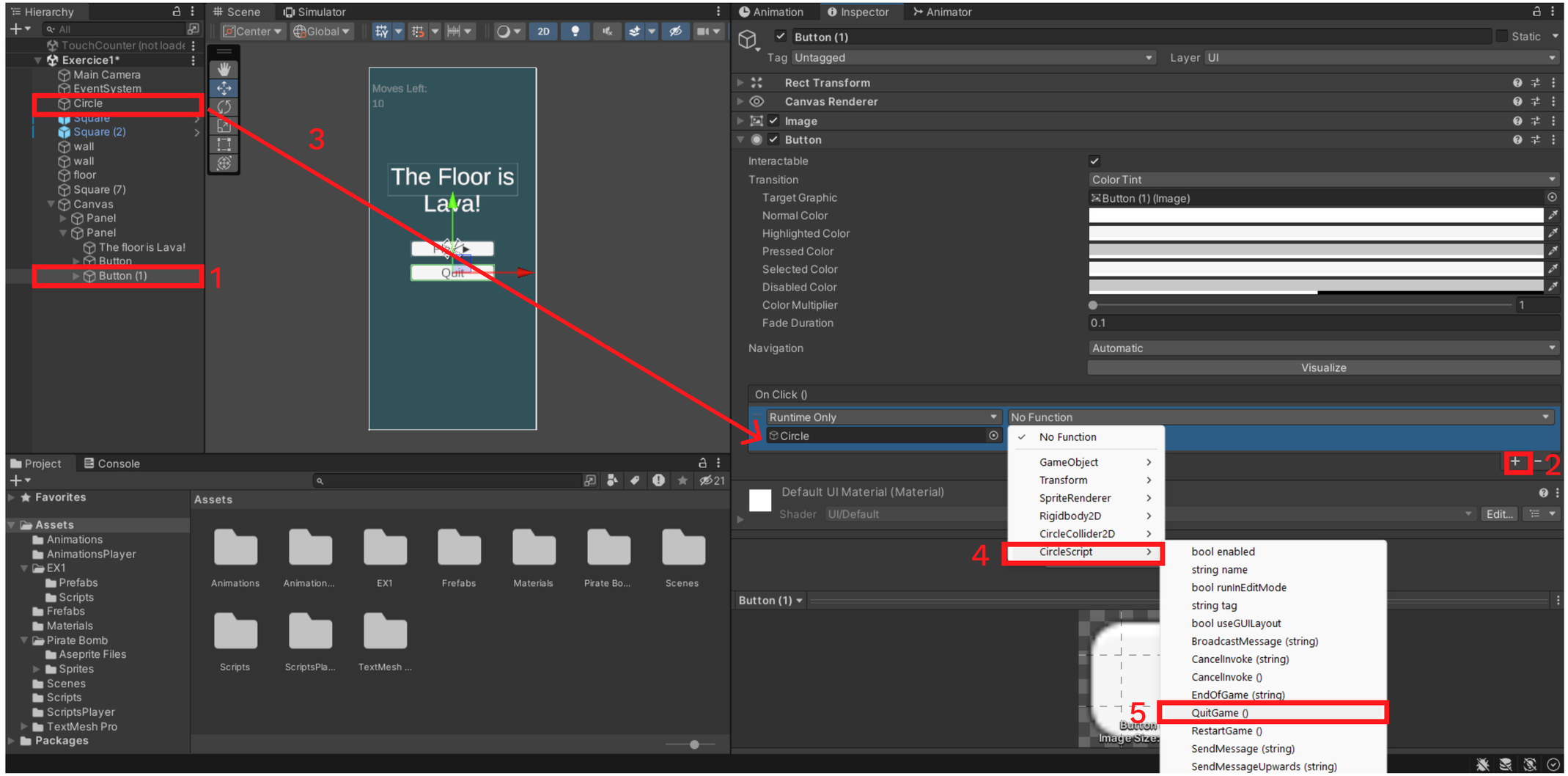
# ADD YOUR SCENE TO YOUR BUILD DEFINITIONS



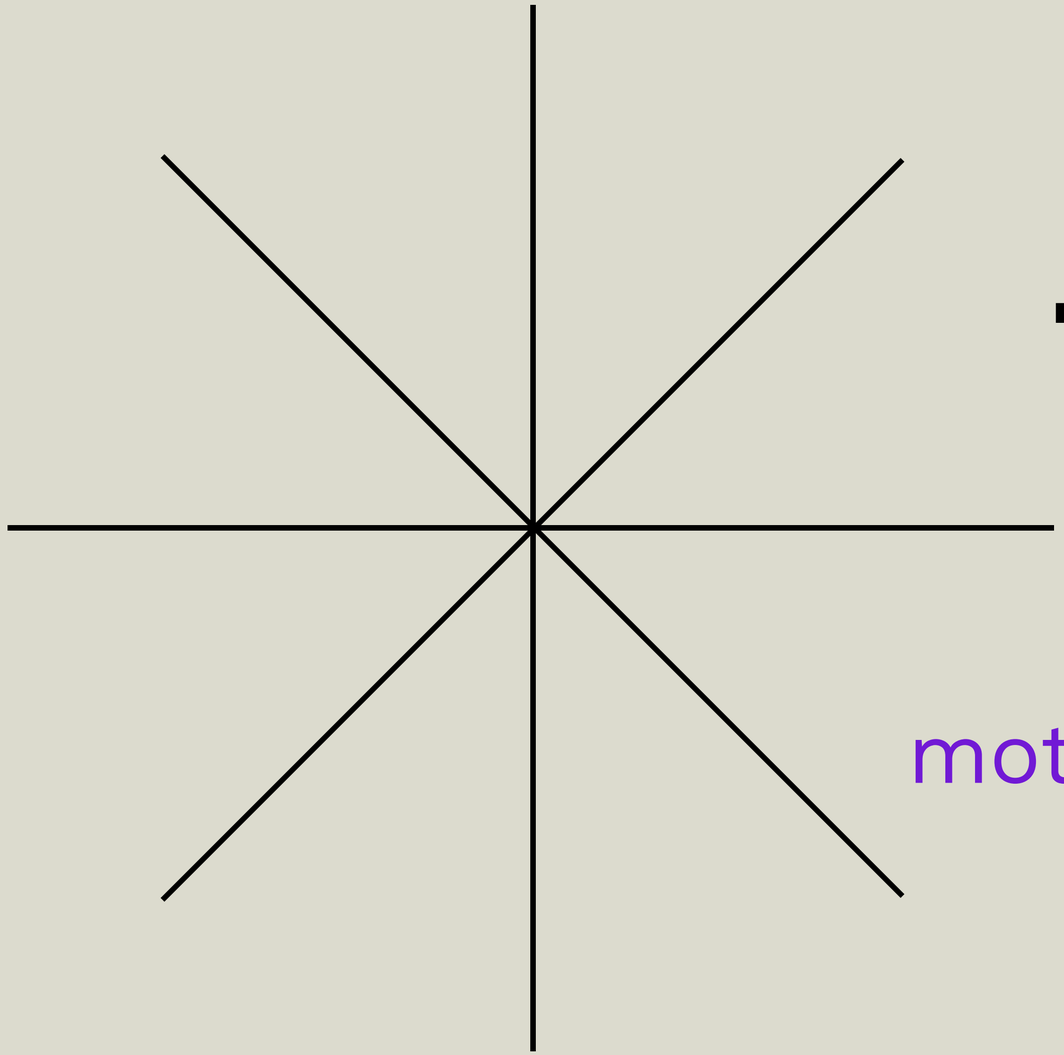
# CREATE A METHOD TO QUIT THE GAME

```
0 referências
public void QuitGame()
{
    Application.Quit();
}
```

# CONNECT THE SECOND BUTTON TO THE METHOD



# CHALLANGE:
# MAKE THE CIRCLE FALL AUTOMATICALLY WHEN THERE ARE NO MOVES LEFT!



# Thank you!

Don’t forget where to find this powerpoint:

motamdaniela.github.io/dam