
SUPER
MARIO
TM

What are the characteristics of Super Mario Bros?

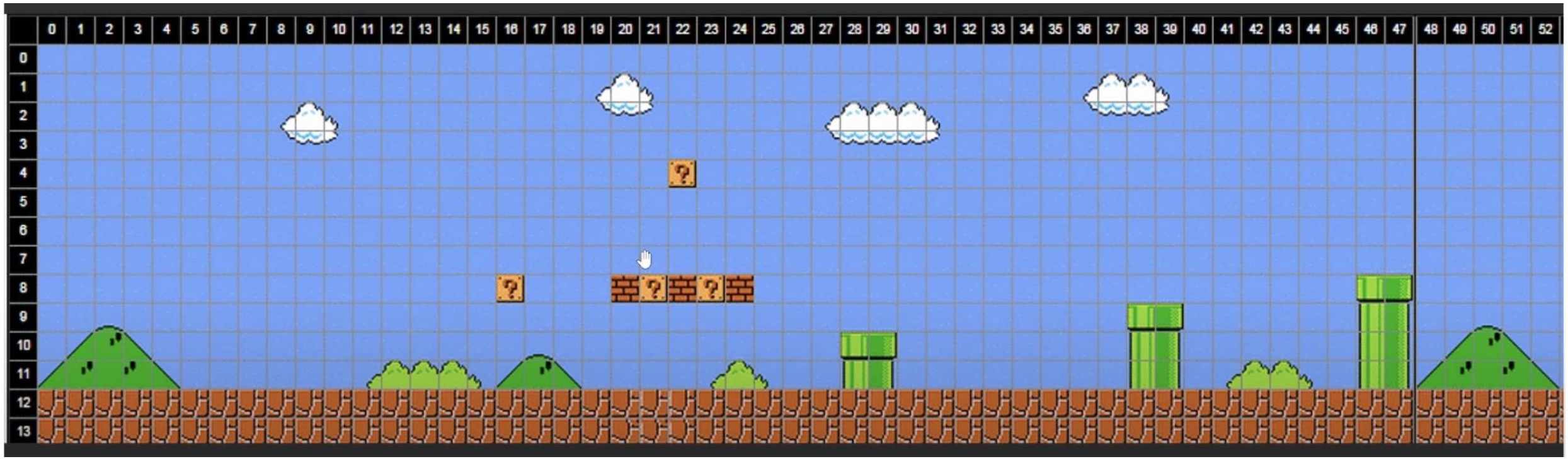
Key characteristics (requirements)

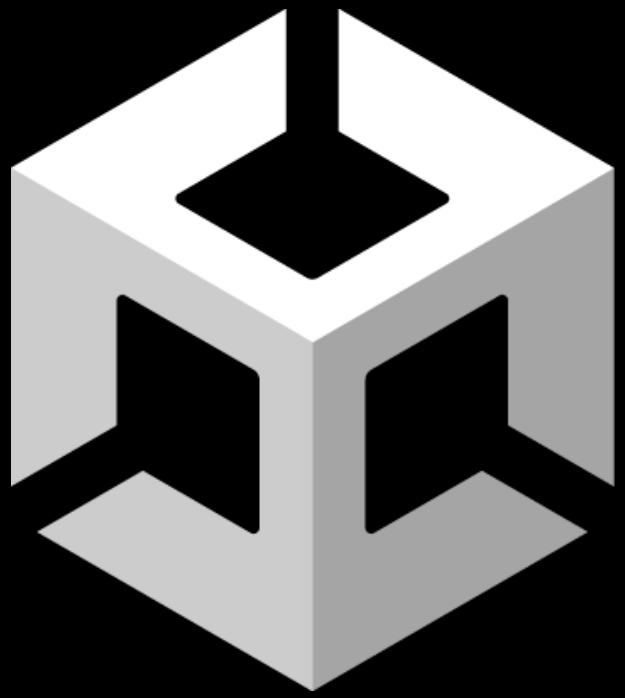
- Side scroller: Mario/Camera moves right
- Gravity effect of jump
- Collects coins and optional powerups
- Enemies (Goombas) try to take lives.
- Time limit in which Mario must reach the end of the level

Steps / Requirements

1. Level Design in Unity
2. Mario Movement (left + right)
3. Object detection / collision
4. Jumping over blocks / gravity
5. Enemies (Goombas) alternate directions
6. Jumping on Goombas – further collisions
7. Decoration – animation – walking
8. Camera to track Mario (once tile map created)

Unity Level tilemap



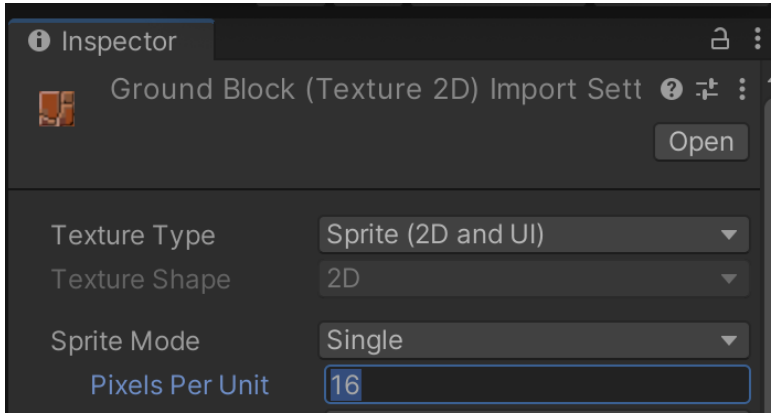


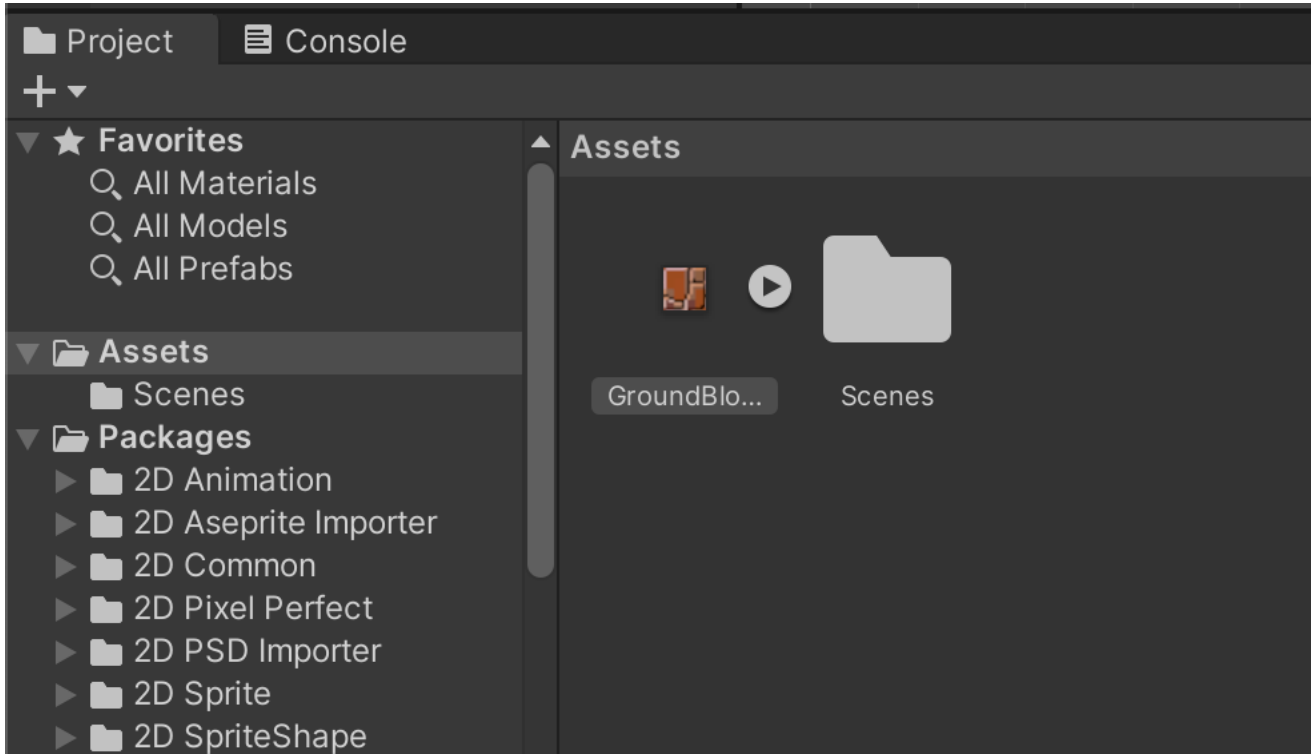
Unity®

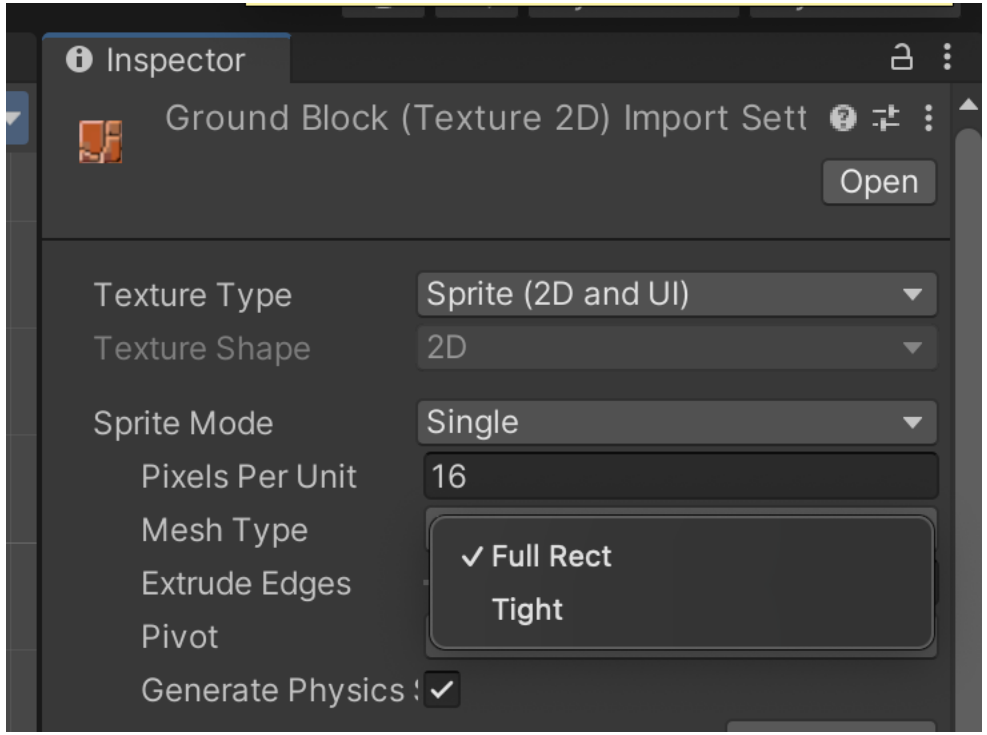
Steps / Requirements

1. **Level Design in Unity**
2. Mario Movement (left + right)
3. Object detection / collision
4. Jumping over blocks / gravity
5. Enemies (Goombas) alternate directions
6. Jumping on Goombas – further collisions
7. Decoration – animation – walking
8. Camera to track Mario (once tile map created)

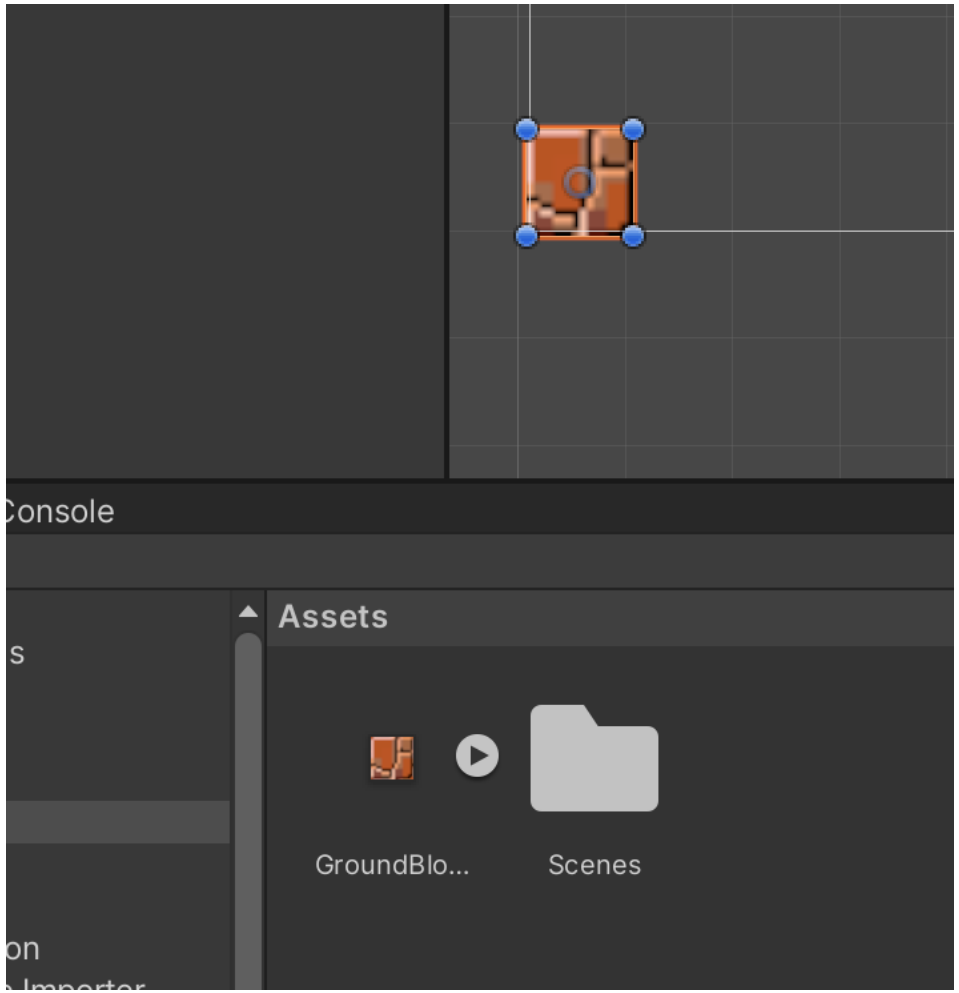
Level Design - load in Ground Block Pixels Per Unit = 16

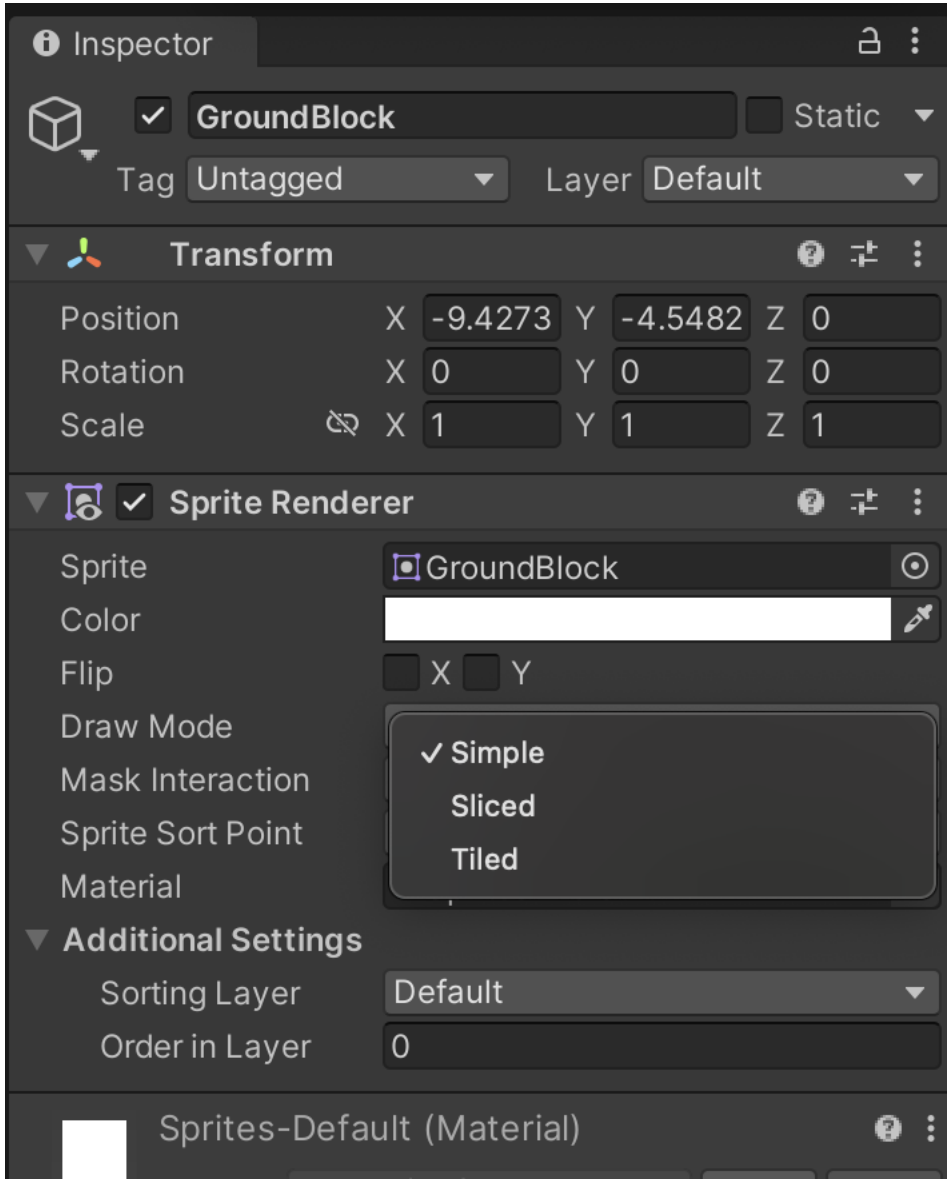




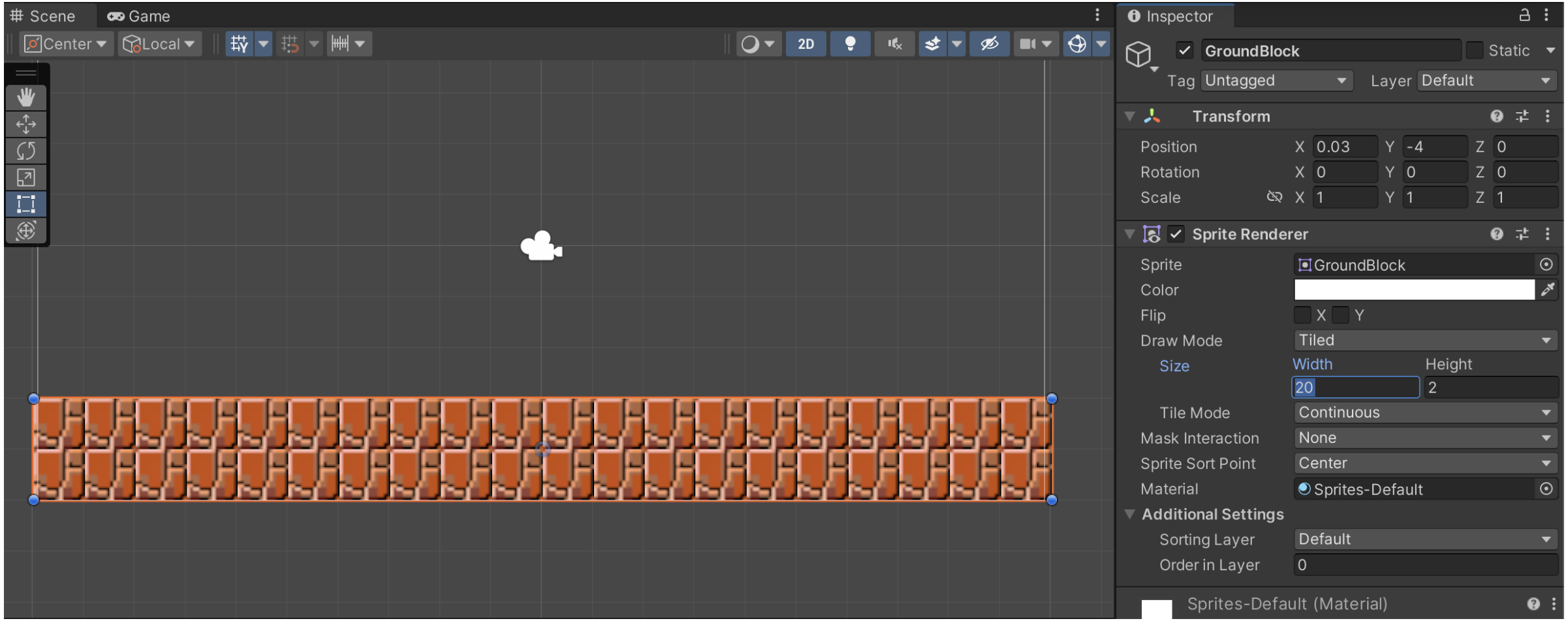


Select ‘Tiled’

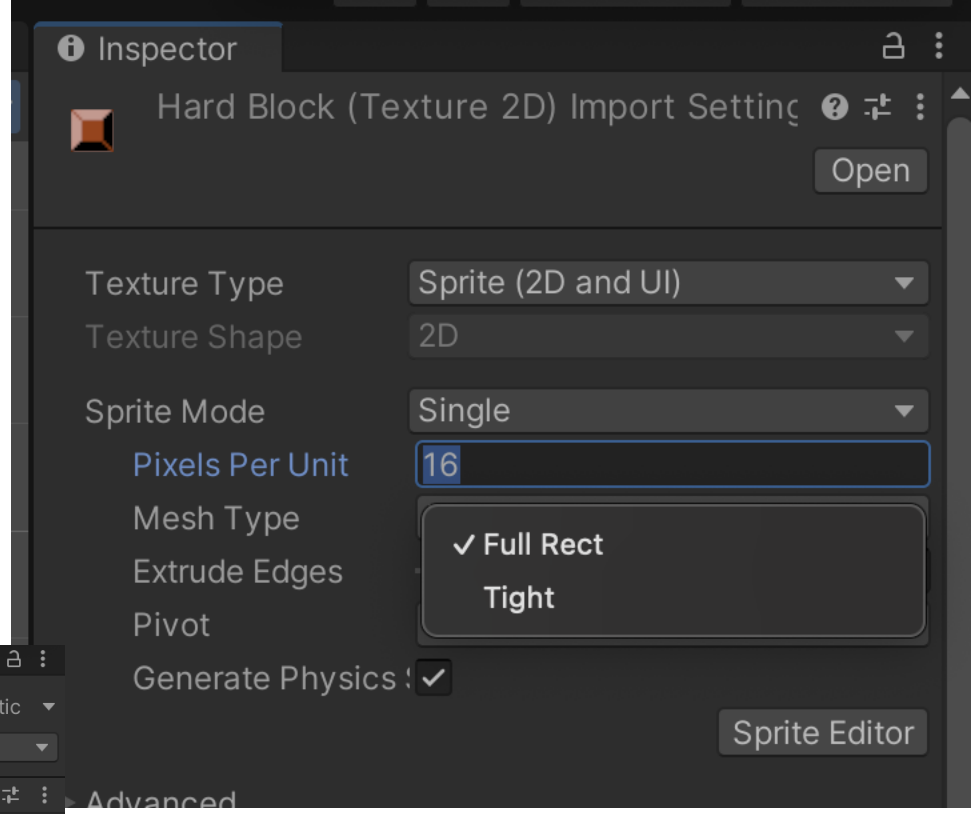


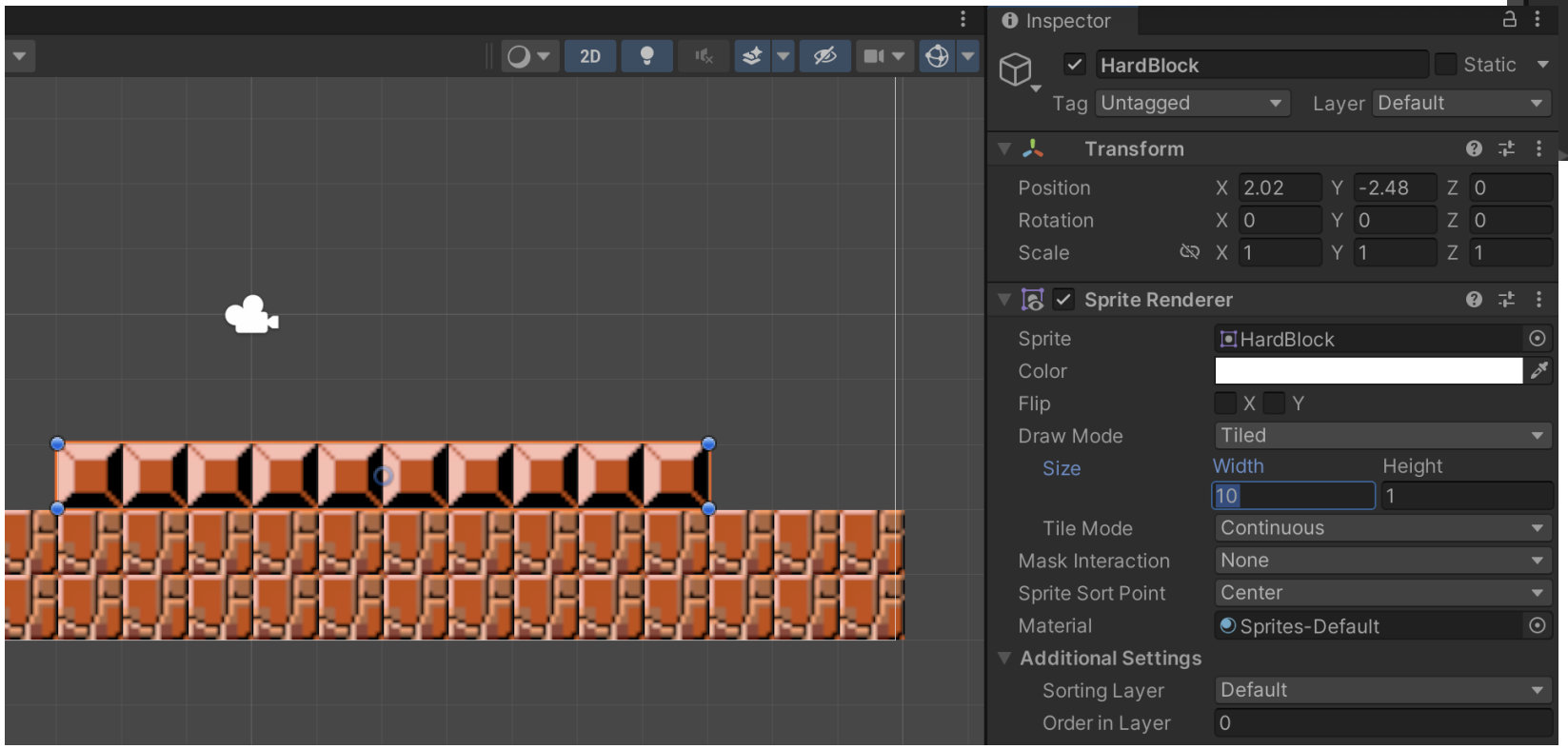


Tiled Draw Mode – 20 tiles wide x 2 high

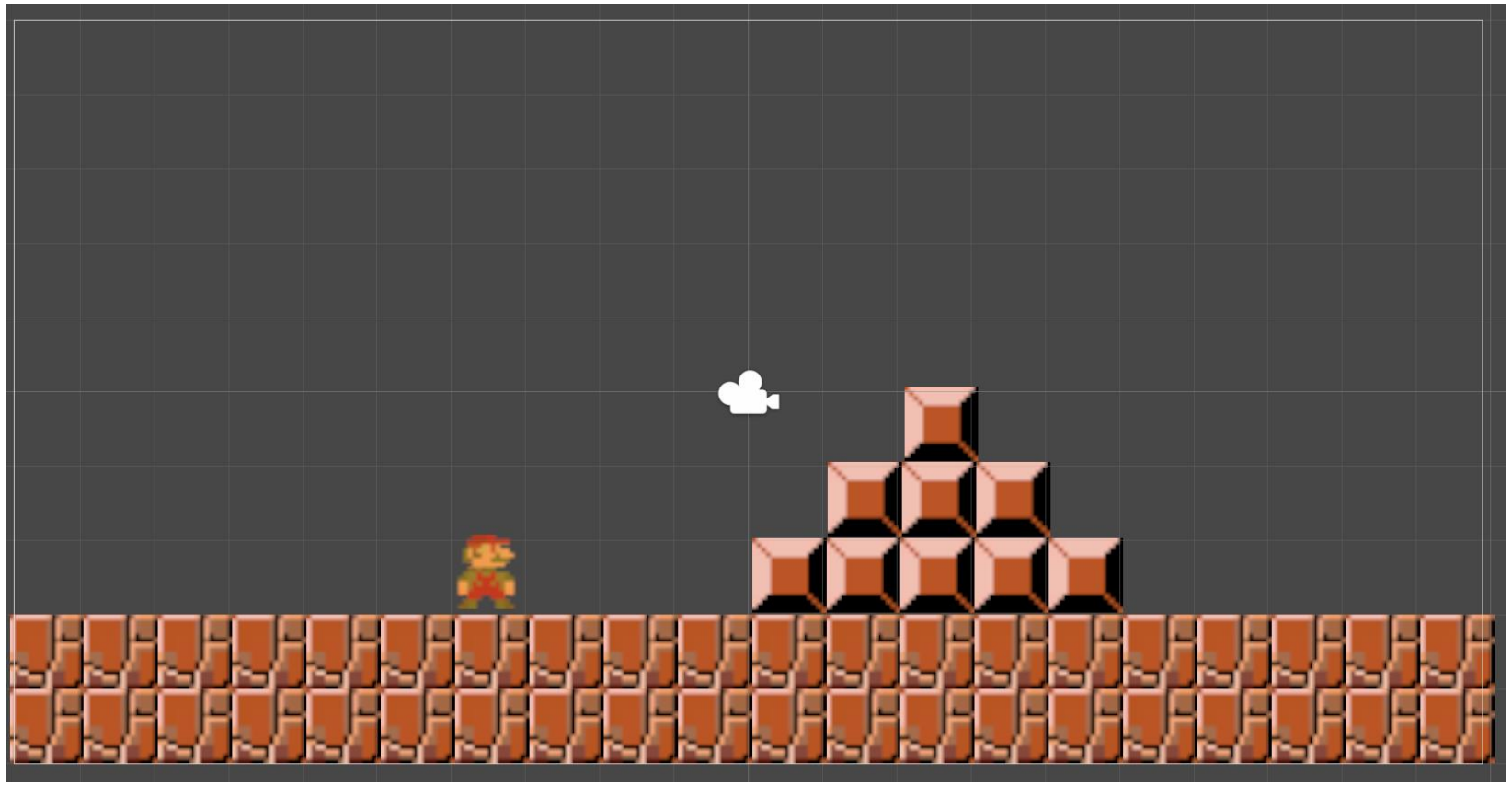


In Unity – Select Tiled





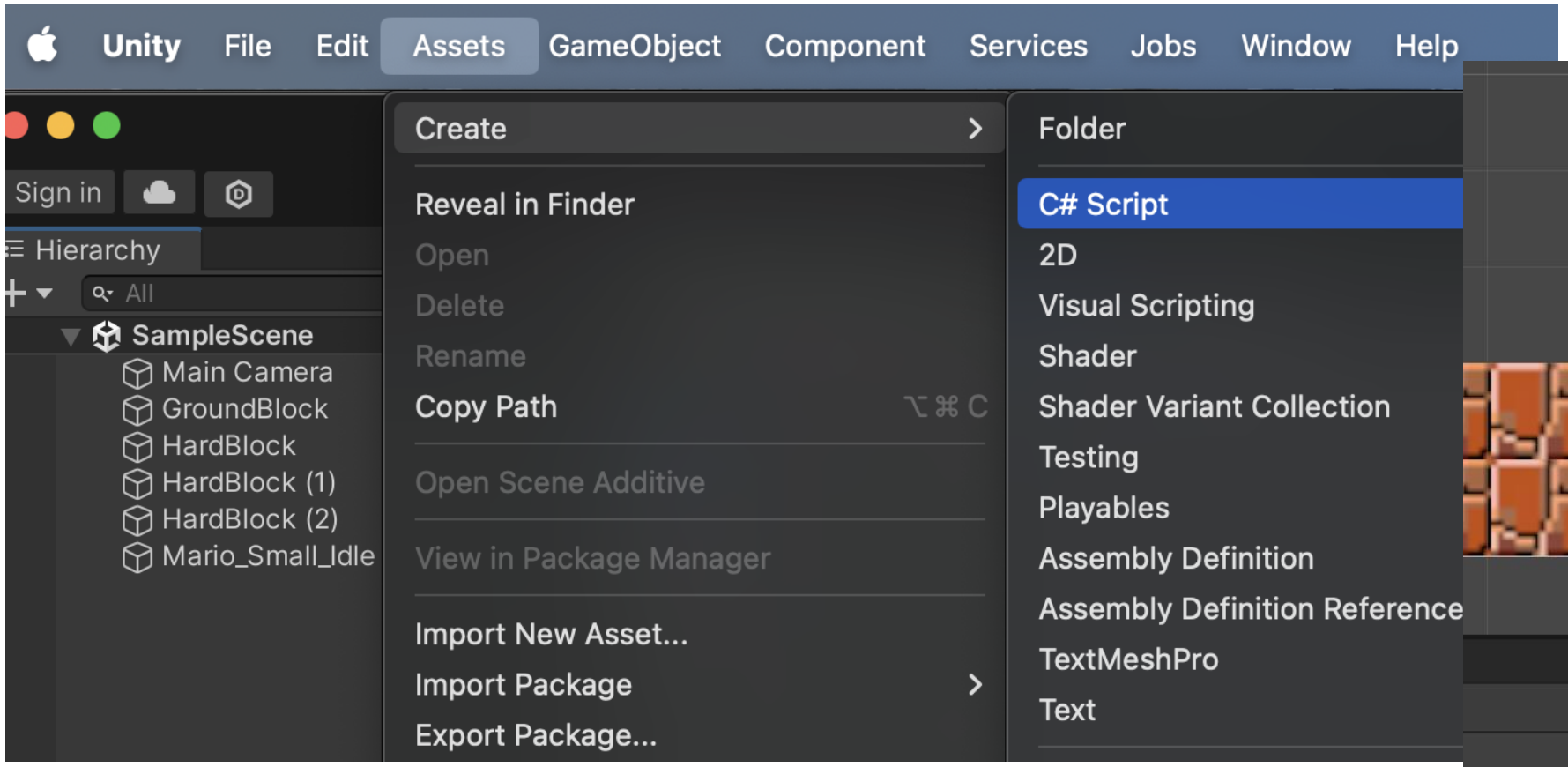
Simple Level Design

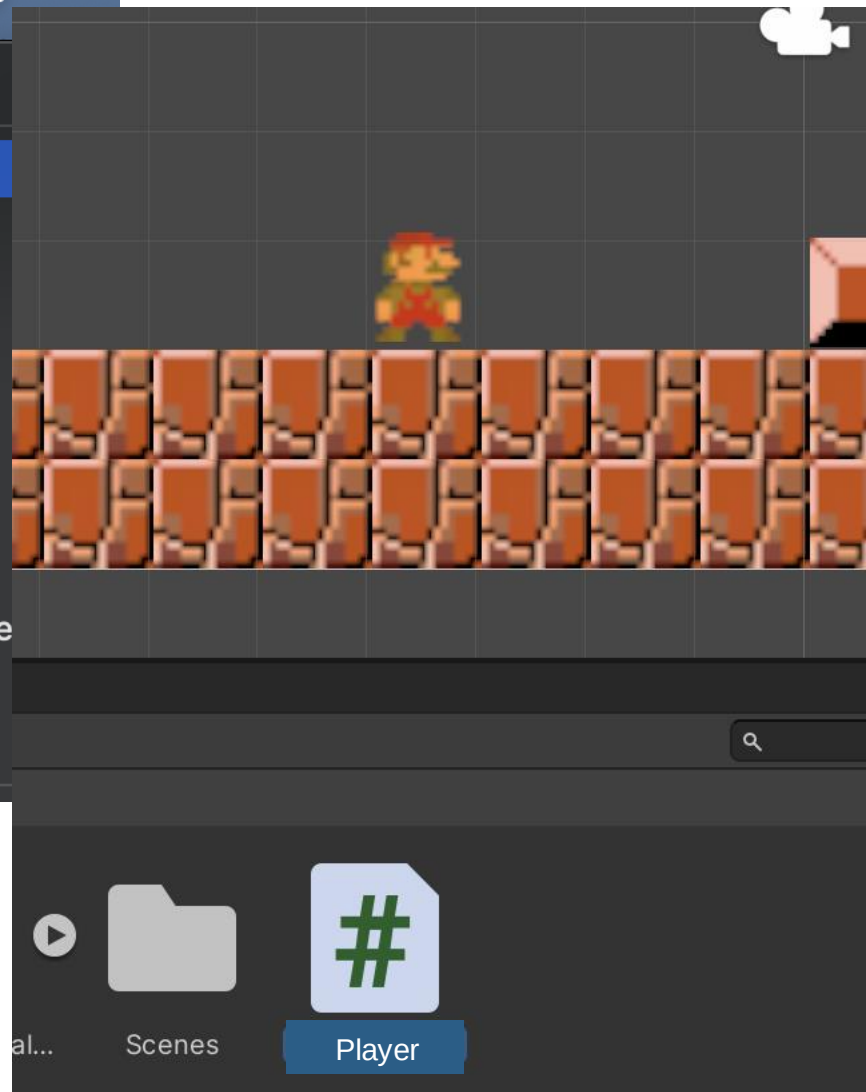


Steps / Requirements

1. Level Design in Unity ✅
2. **Mario Movement (left + right)**
3. Object detection / collision
4. Jumping over blocks / gravity
5. Enemies (Goombas) alternate directions
6. Jumping on Goombas – further collisions
7. Decoration – animation – walking
8. Camera to track Mario (once tile map created)

2. Mario Movement



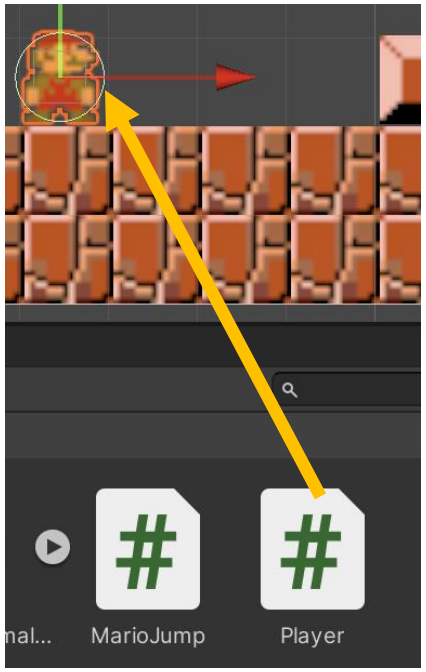


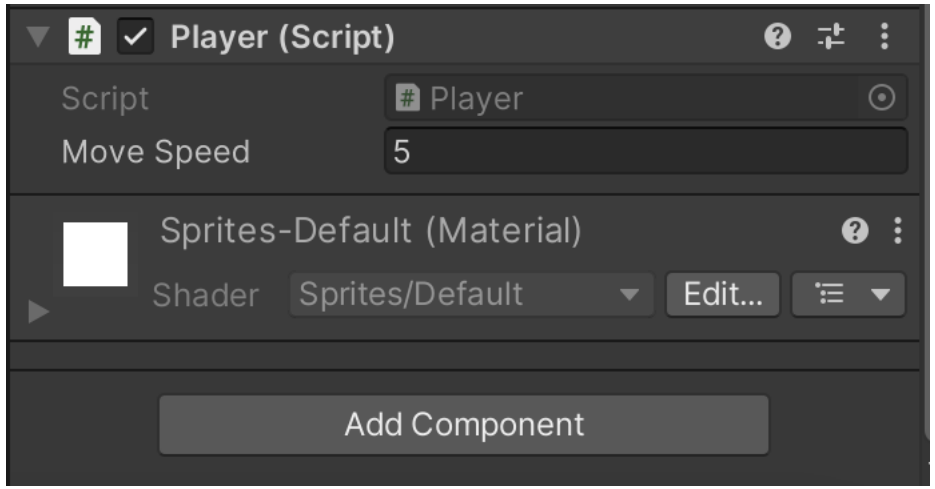
Player.cs

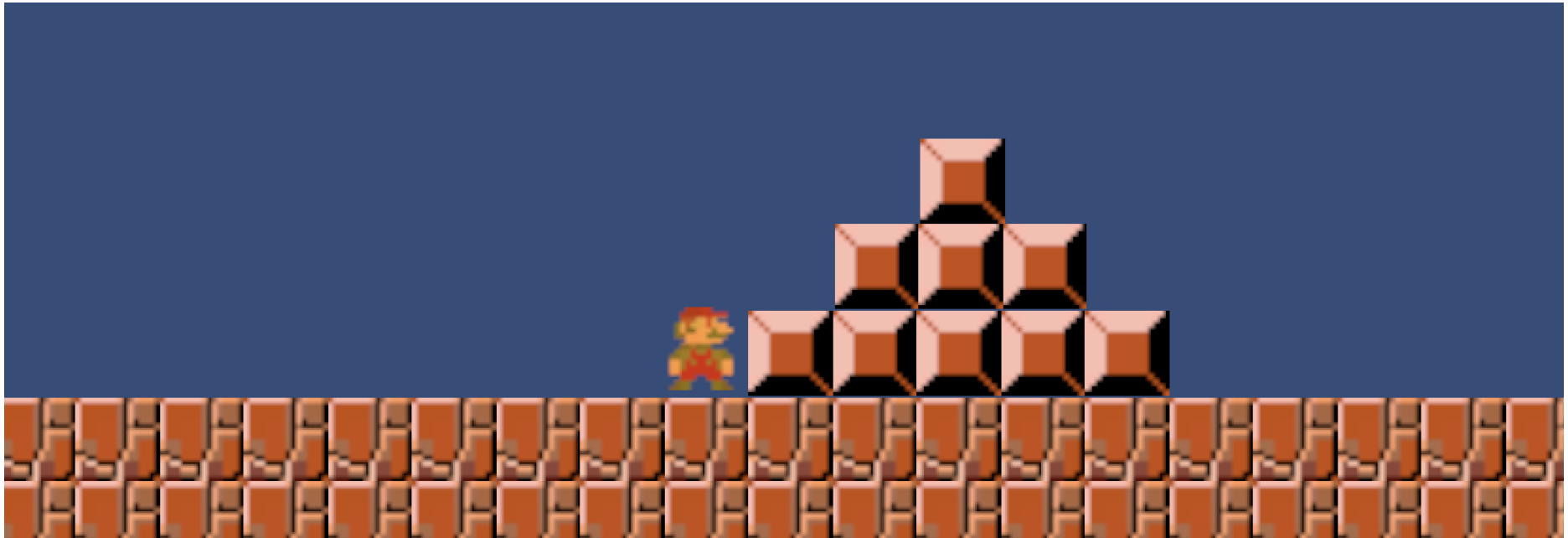
Users > nick > Documents > test_p_1 > Assets > Player.cs

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

public class Player : MonoBehaviour
{
    public float moveSpeed = 5f;

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        // Get horizontal input (-1 for left, 1 for right)
        float horizontalInput = Input.GetAxis("Horizontal");

        // Calculate movement vector
        Vector3 movement = new Vector3(horizontalInput, 0f, 0f) * moveSpeed * Time.deltaTime;

        // Move the sprite
        transform.position += movement;
    }
}

```

2. Mario Movement



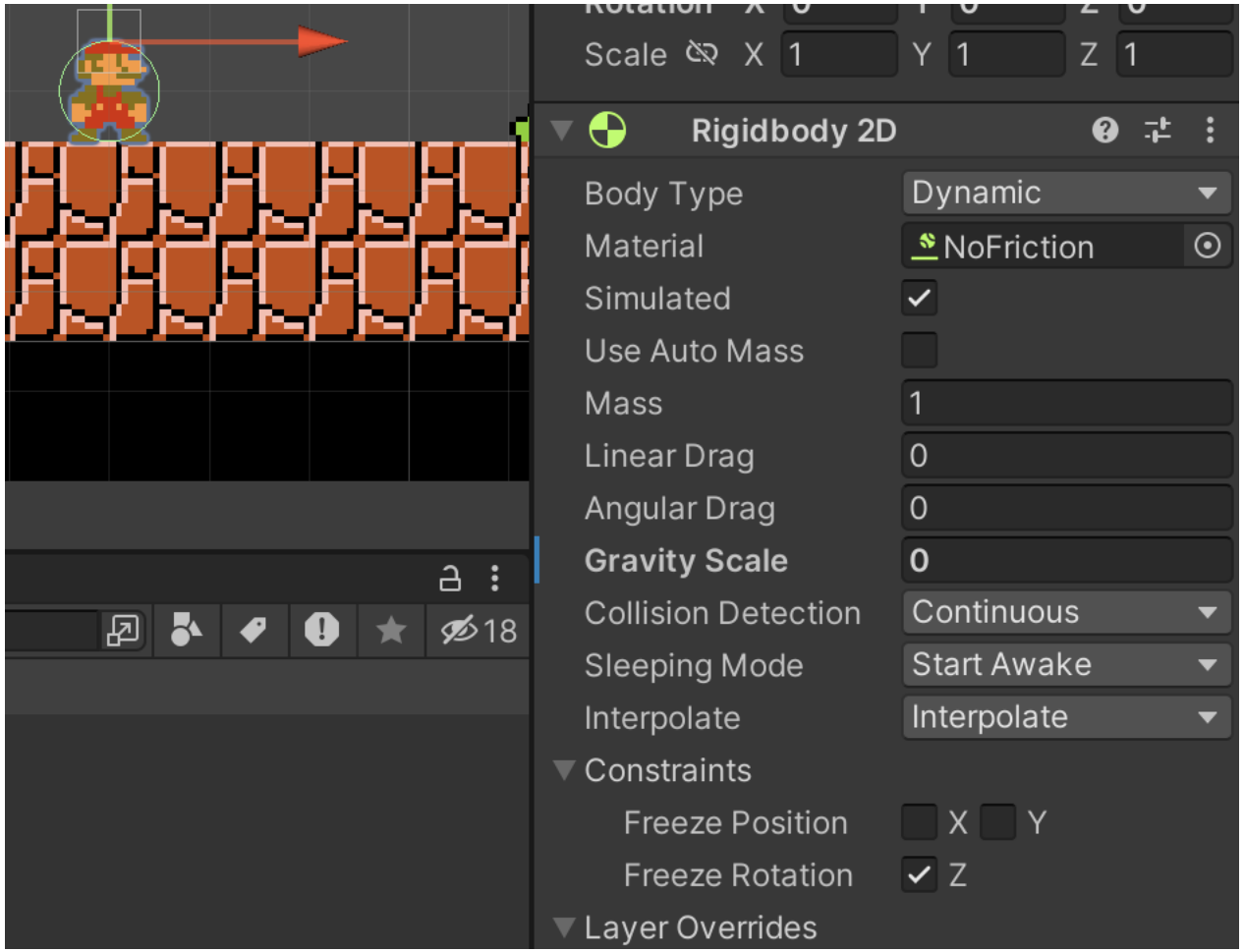


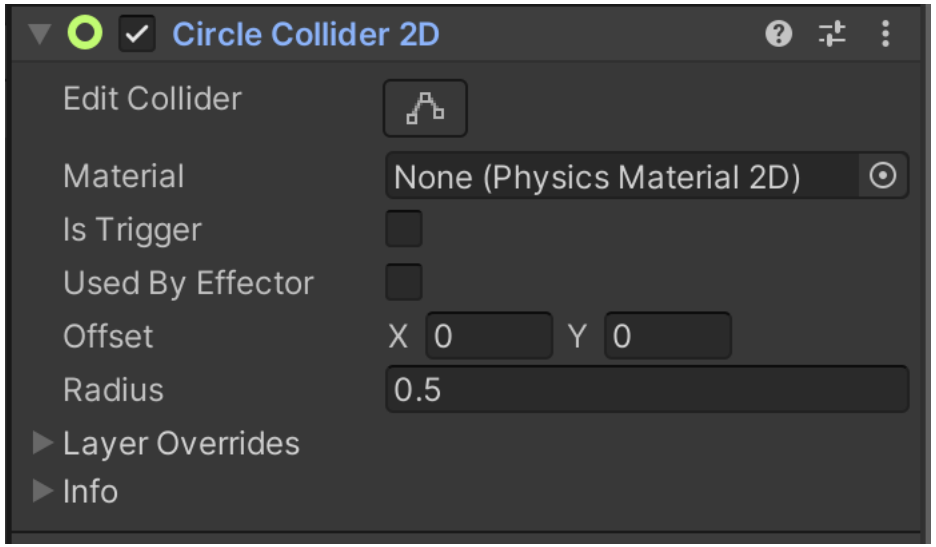


Steps / Requirements

1. Level Design in Unity ✅
2. Mario Movement (left + right) ✅
3. **Object detection / collision**
4. Jumping over blocks / gravity
5. Enemies (Goombas) alternate directions
6. Jumping on Goombas – further collisions
7. Decoration – animation – walking
8. Camera to track Mario (once tile map created)

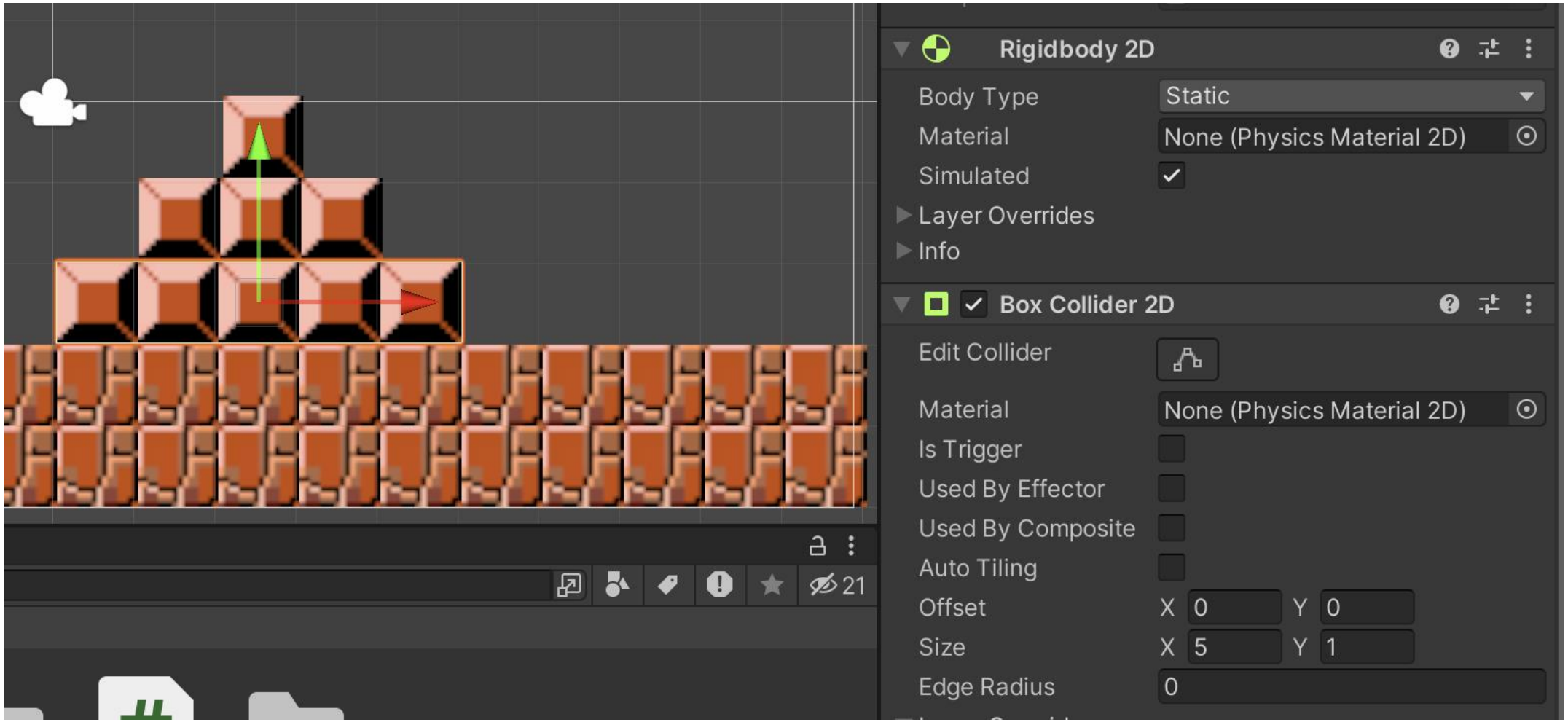
3. Object Detection – Colliders for Mario





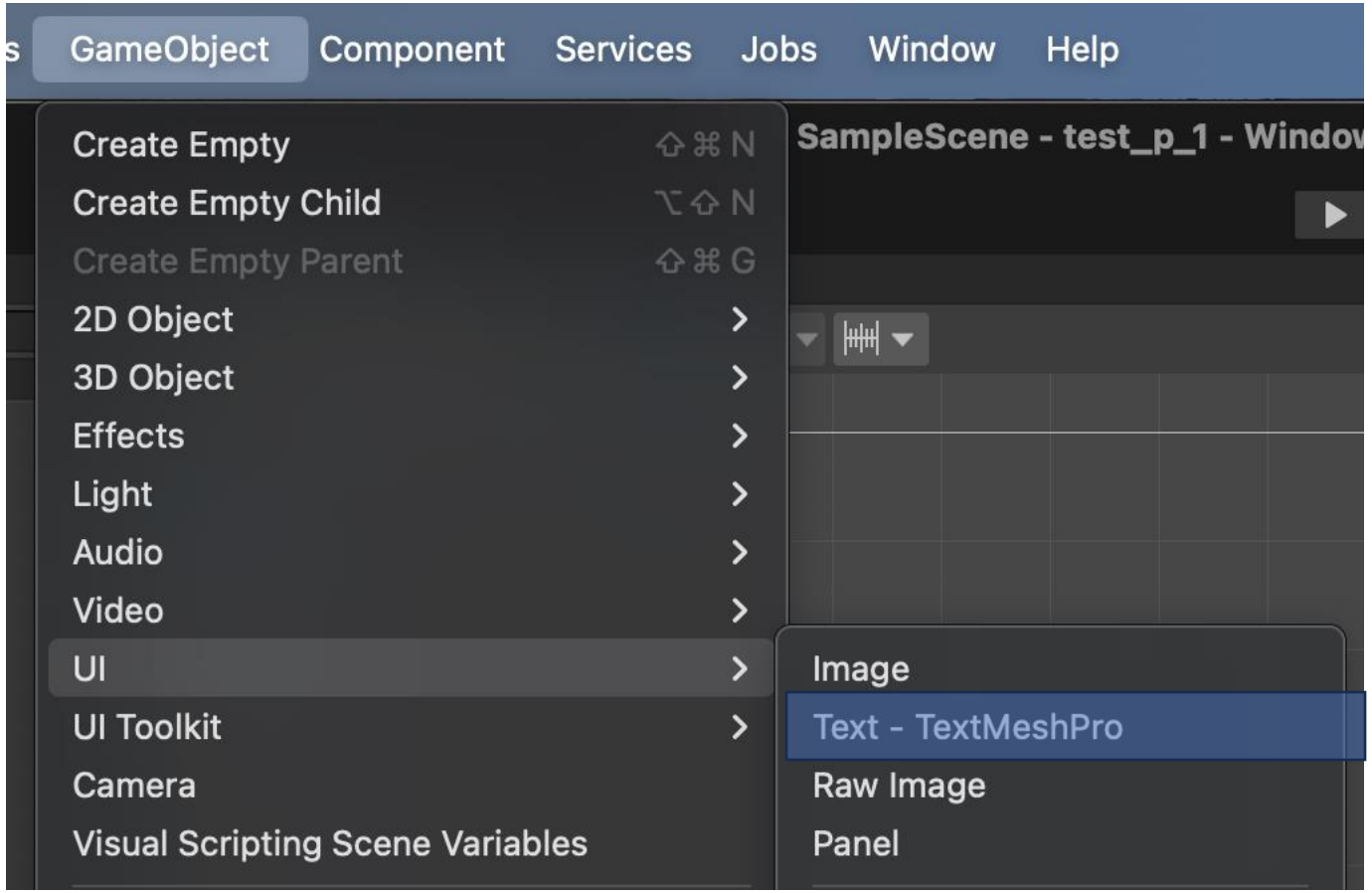
Click on Mario – add new Component -> Physics -> Rigidbody 2D
Click on Mario – add new Component -> Physics -> Circle Collider 2D

3. Object Detection – Colliders for Block



Click on Block – add new Component -> Physics -> Rigidbody 2D
Click on Block – add new Component -> Physics -> Box Collider 2D

3. Collision text – for visual cues



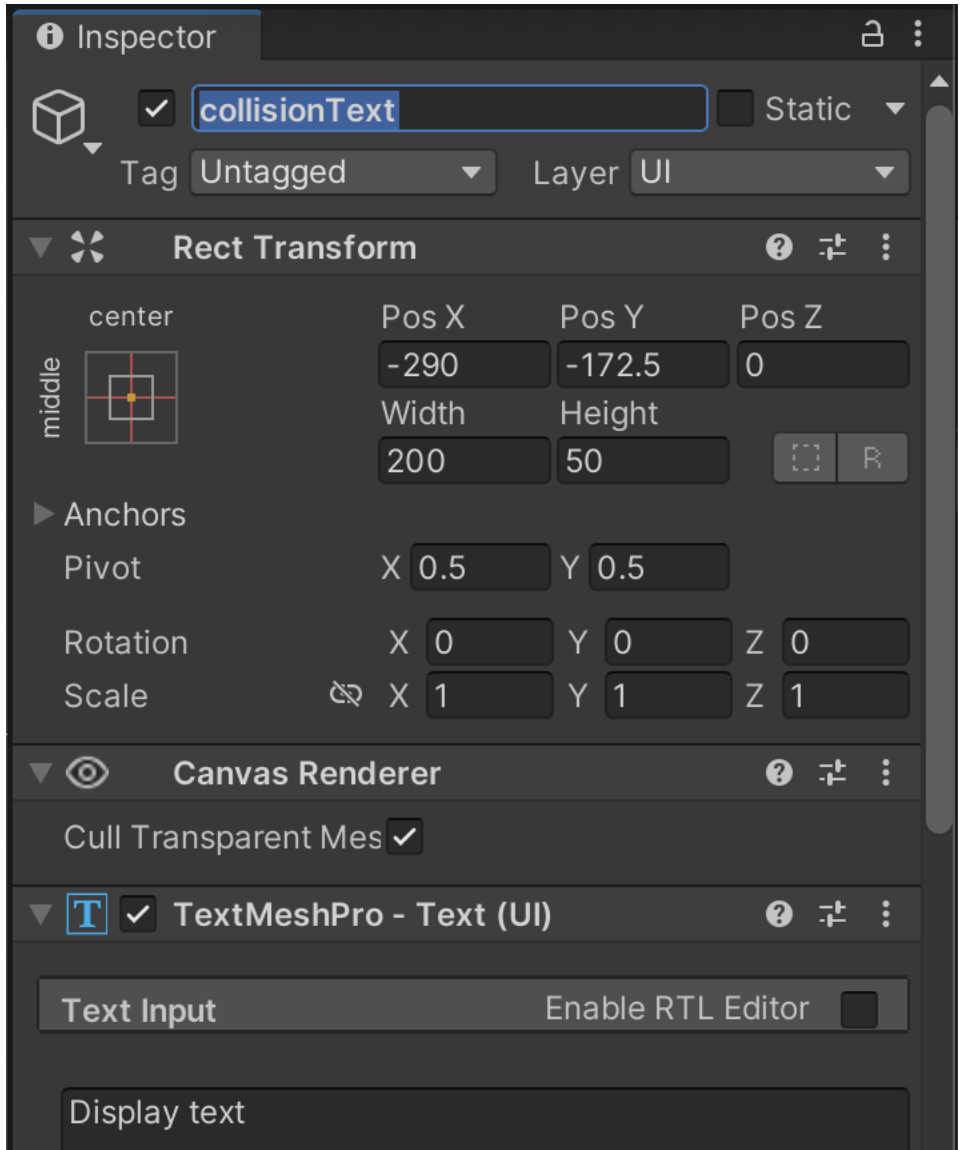
GameObject -> UI -> Text - TextMeshPro

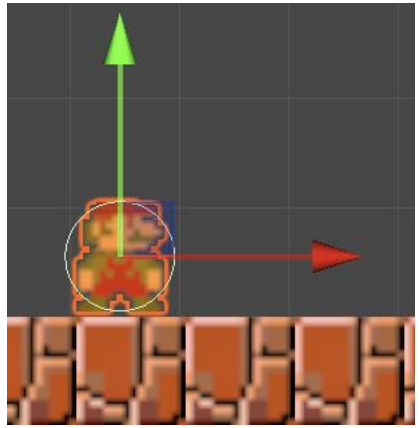
3. Collision code in our C# Script - for visual cues

```
using UnityEngine;

using TMPro; // For TextMeshPro

public class Player : MonoBehaviour
{
    public float moveSpeed = 5f;

    public TextMeshProUGUI collisionText;
```

```
void OnCollisionEnter2D(Collision2D collision)
{
    collisionText.text = "Collided with: " + collision.gameObject.name;
    // Print a message to the Console when a 2D collision occurs
    Debug.Log("Collided with: " + collision.gameObject.name);
}
```

3. Update parameters in the Inspector





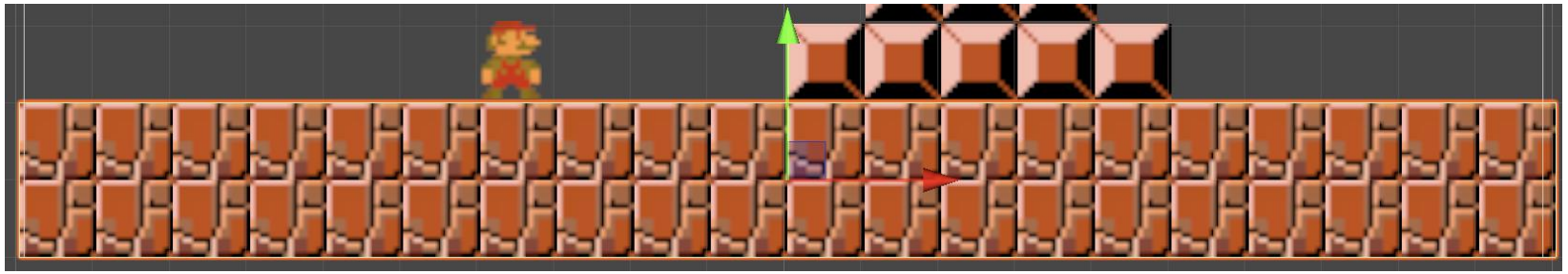
3. Change the TextMesh.text upon collision!

Steps / Requirements

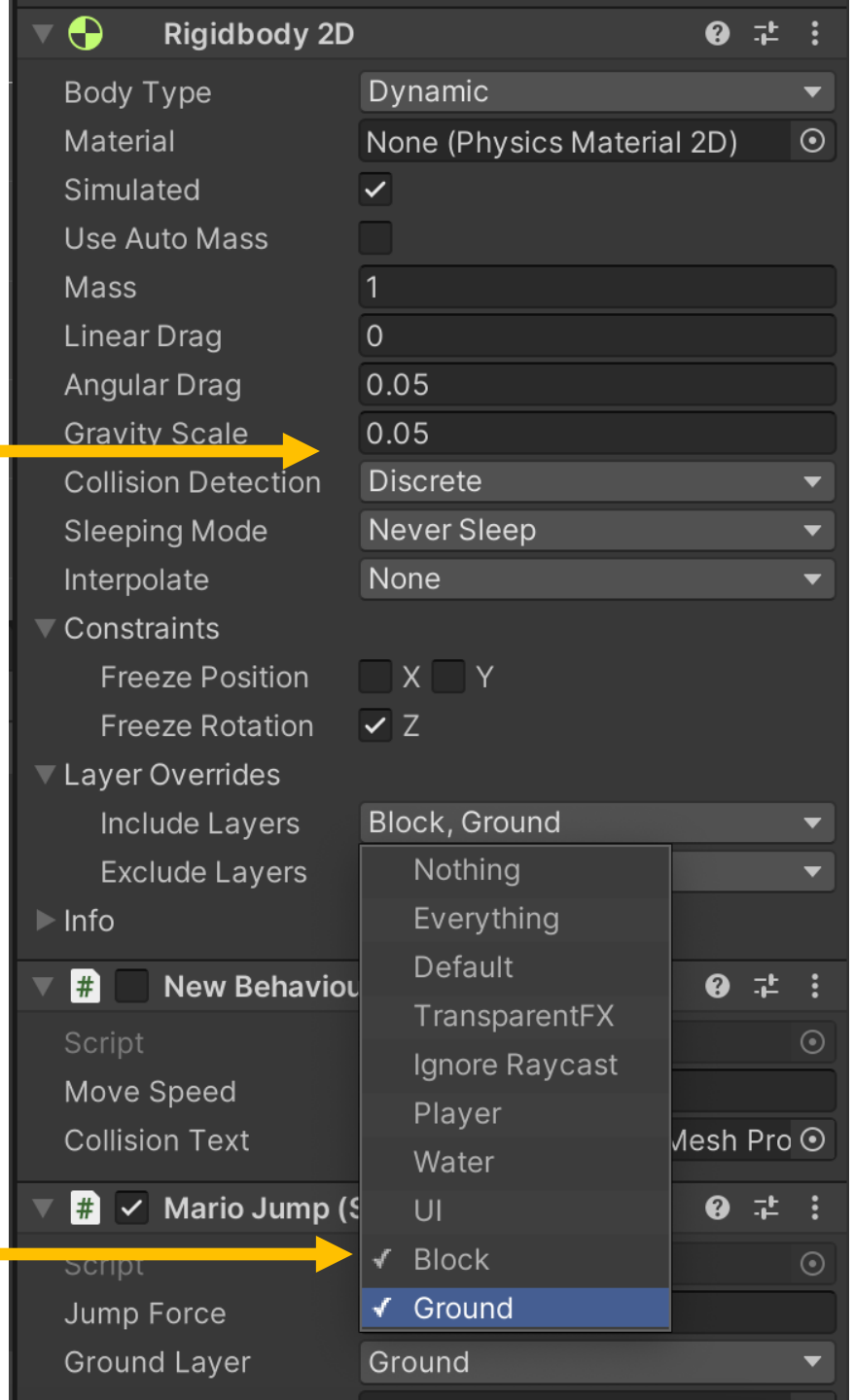
1. Level Design in Unity ✅
2. Mario Movement (left + right) ✅
3. Object detection / collision ✅
4. **Jumping over blocks / gravity**
5. Enemies (Goombas) alternate directions
6. Jumping on Goombas – further collisions
7. Decoration – animation – walking
8. Camera to track Mario (once tile map created)

4. Gravity / Jump – Mario

Start by setting Mario's RigidBody 2D Gravity to a low value > 0 but < 1 so you can check whether he falls through the floor or not…

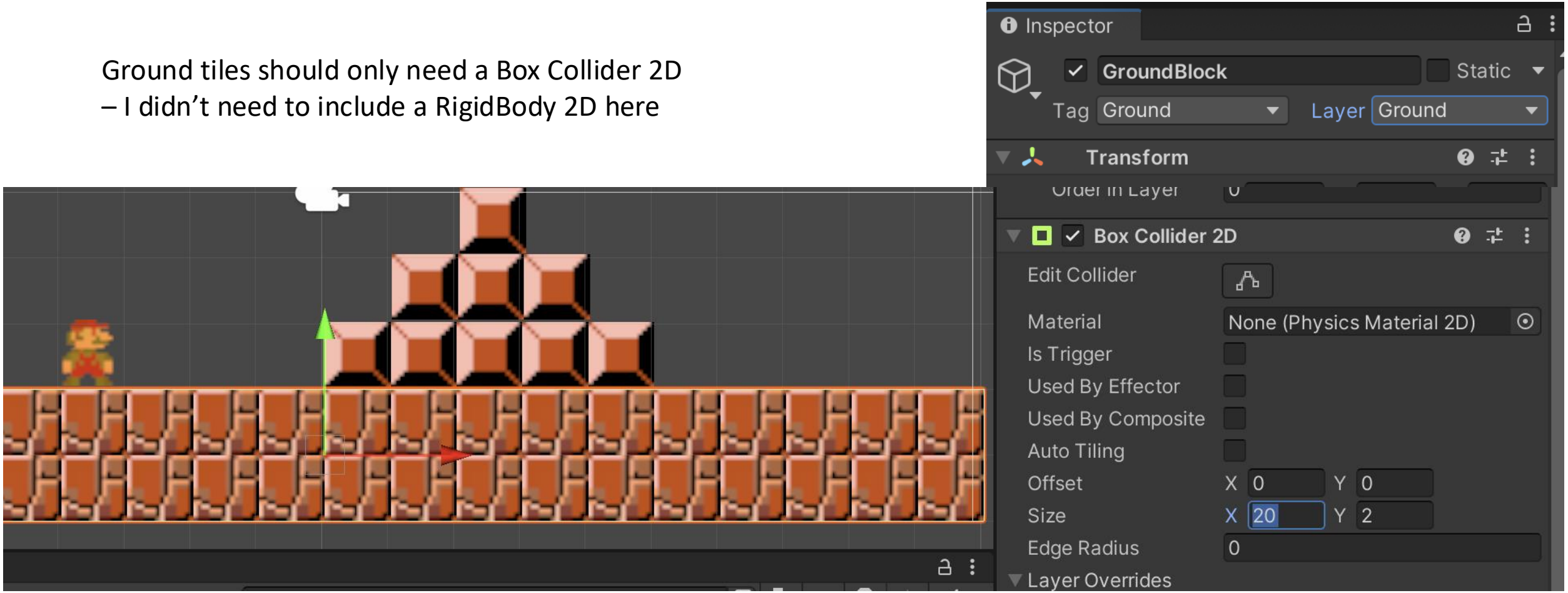


Also make sure to 'INCLUDE' Ground tiles/layers you have



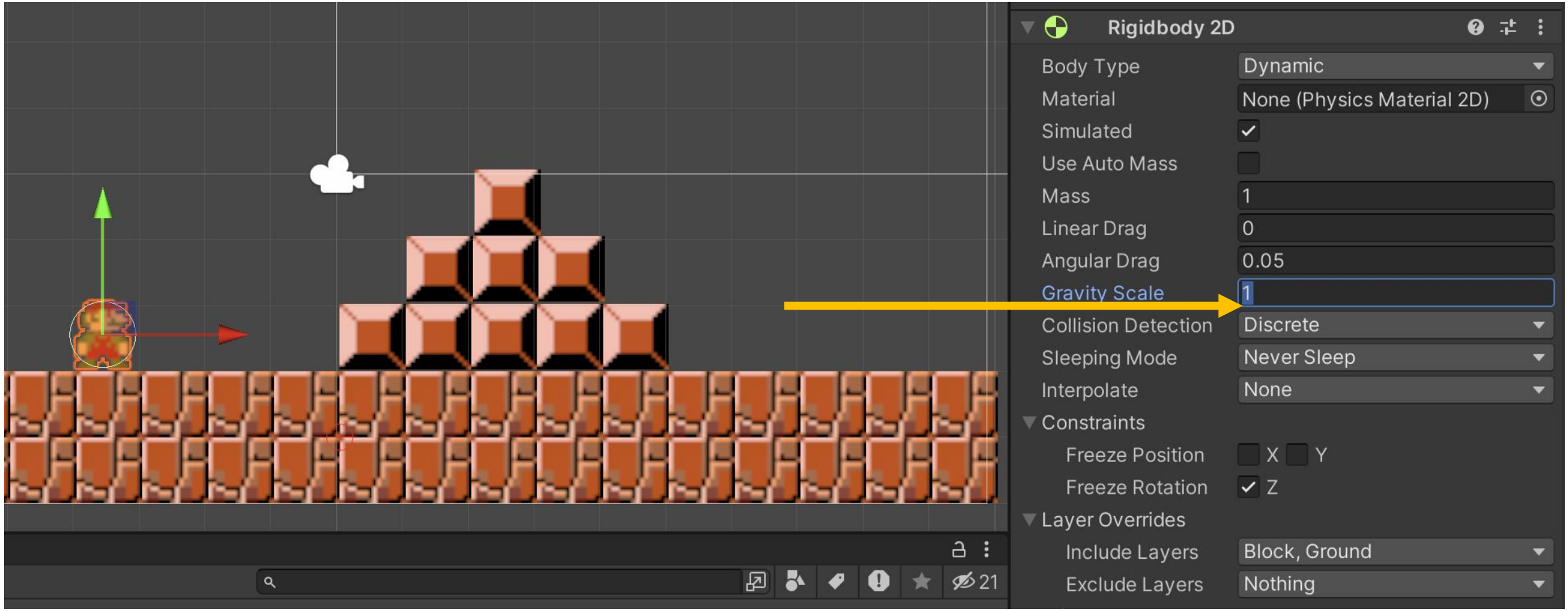
4. Gravity / Jump – Check Ground tiles

Ground tiles should only need a Box Collider 2D
– I didn't need to include a RigidBody 2D here



4. Gravity / Jump – Mario

If Mario doesn't fall through the floor – then increase the Gravity Scale to 1 (or higher)



4

Player.cs | MarioJump.cs

Users > nick > Documents > test_p_1 > Assets > MarioJump.cs

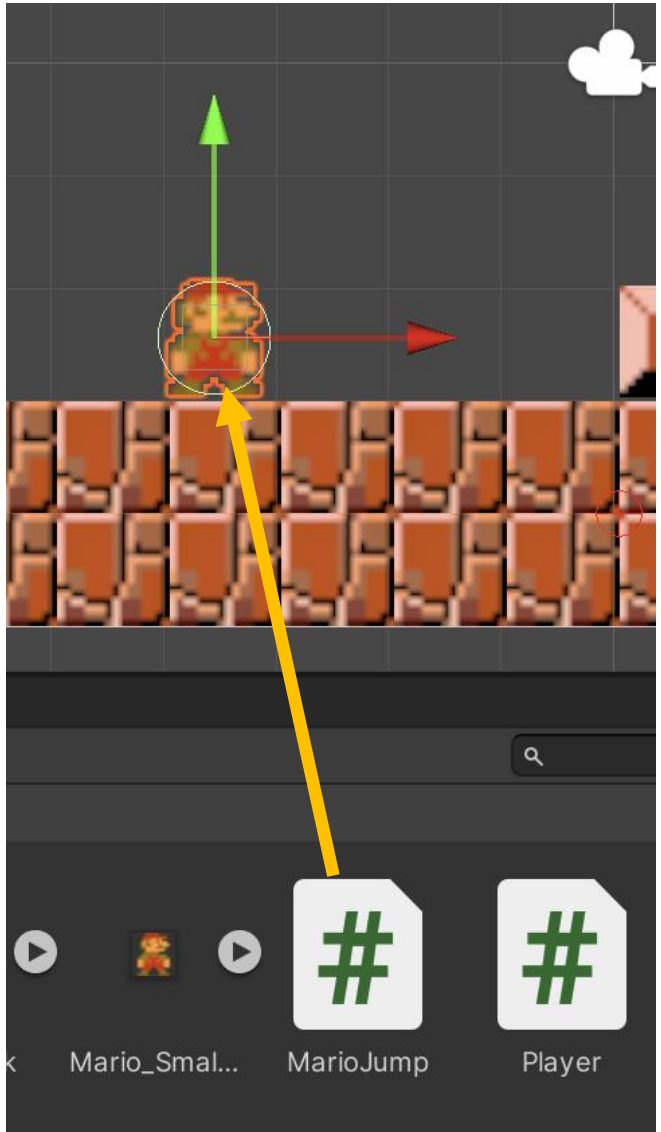
```
using UnityEngine;
using TMPro; // For TextMeshPro

public class MarioJump : MonoBehaviour
{
    private Rigidbody2D rb;
    public float jumpForce = 10f; // Force of the jump
    public LayerMask groundLayer; // Layer representing ground
    public Transform groundCheck; // Empty GameObject to check if on the ground
    public float groundCheckRadius = 0.2f; // Radius of the ground check circle

    private bool isGrounded;

    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
    }

    void Update()
    {
        // Check if Mario is on the ground
        isGrounded = Physics2D.OverlapCircle(groundCheck.position, groundCheckRadius, groundLayer);

        // Jump logic
        if (Input.GetButtonDown("Jump") && isGrounded)
        {
            Jump();
        }
    }
```

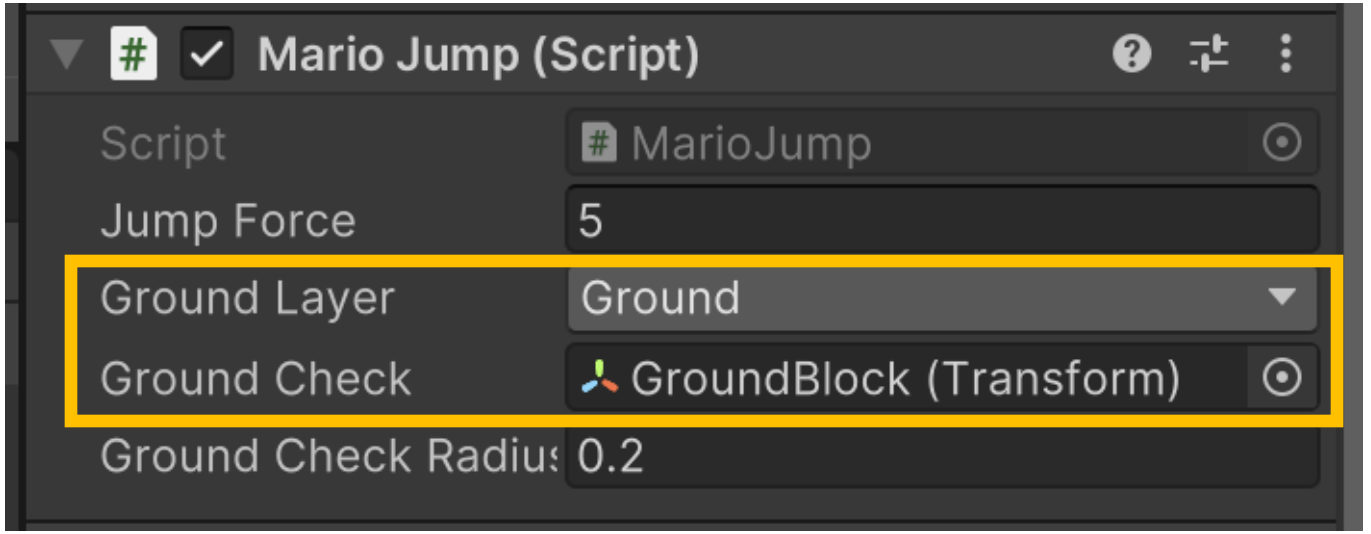
Player.cs | MarioJump.cs

Users > nick > Documents > test_p_1 > Assets > MarioJump.cs

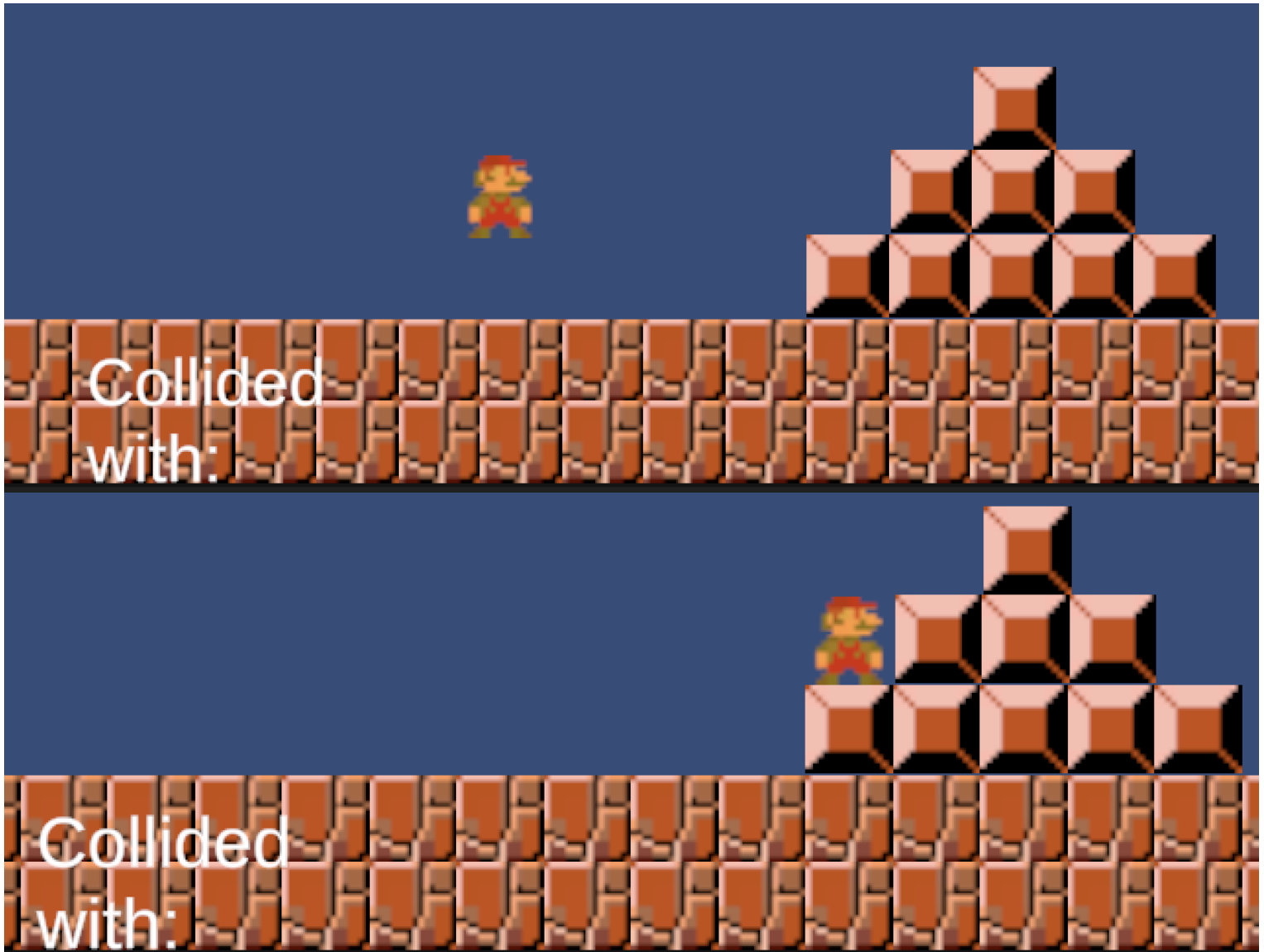
```
5   {
18
19      void Update()
20      {
21          // Check if Mario is on the ground
22          isGrounded = Physics2D.OverlapCircle(groundCheck.position, groundCheckRadius, groundLayer);
23
24          // Jump logic
25          if (Input.GetButtonDown("Jump") && isGrounded)
26          {
27              Jump();
28          }
29      }
30
31      void Jump()
32      {
33          rb.velocity = new Vector2(rb.velocity.x, jumpForce); // Apply upward velocity
34      }
35
36      private void OnDrawGizmosSelected()
37      {
38          // Draw ground check radius for debugging
39          if (groundCheck != null)
40          {
41              Gizmos.color = Color.red;
42              Gizmos.DrawWireSphere(groundCheck.position, groundCheckRadius);
43          }
44      }
45  }
```

4. Gravity / Jump – Mario





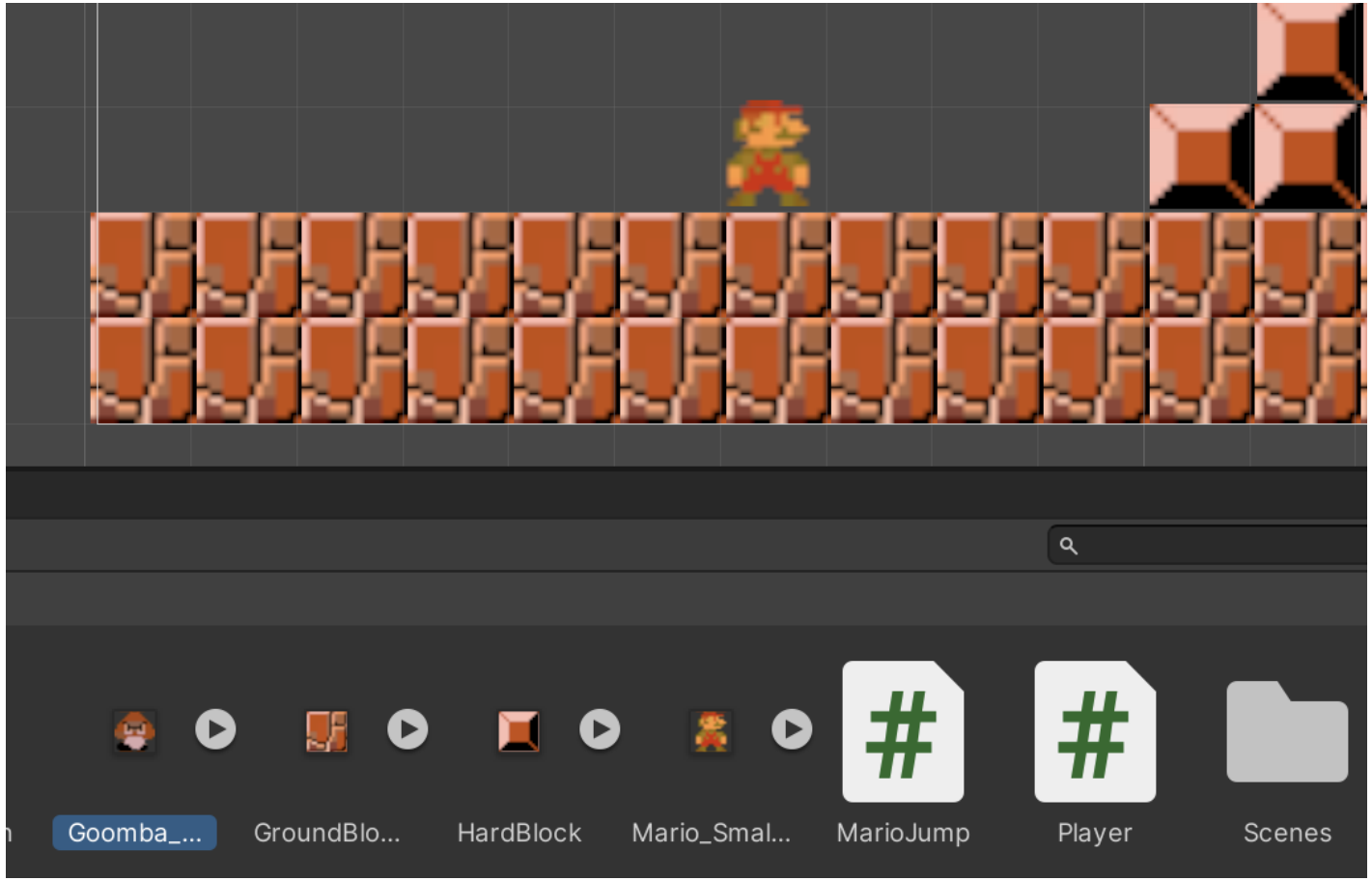
4. Gravity / Jump – Mario

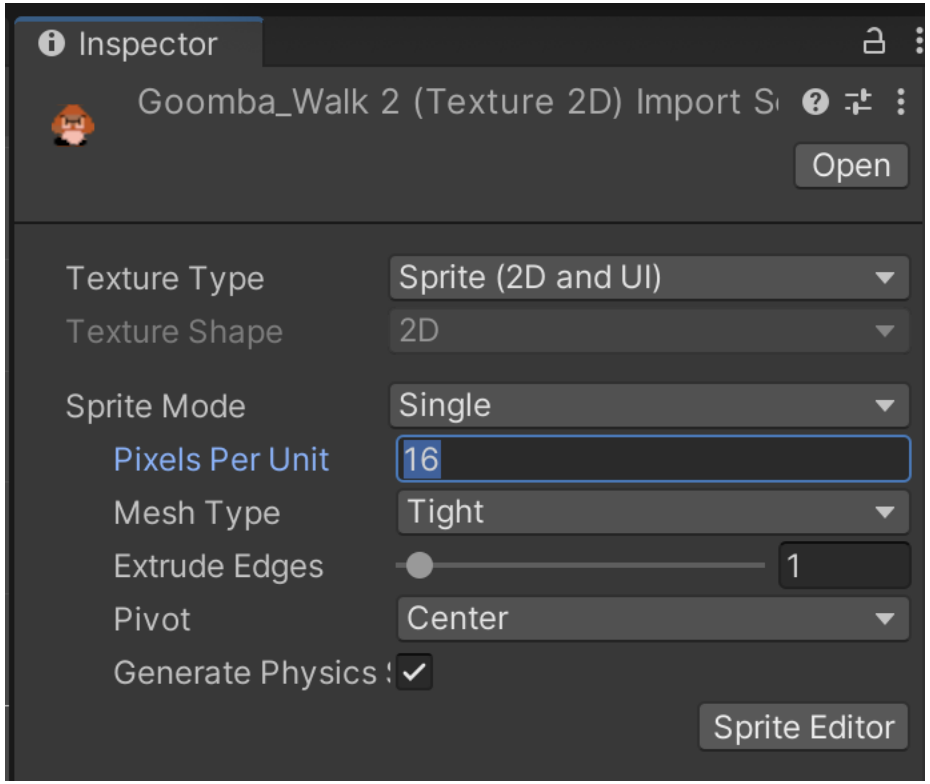


Steps / Requirements

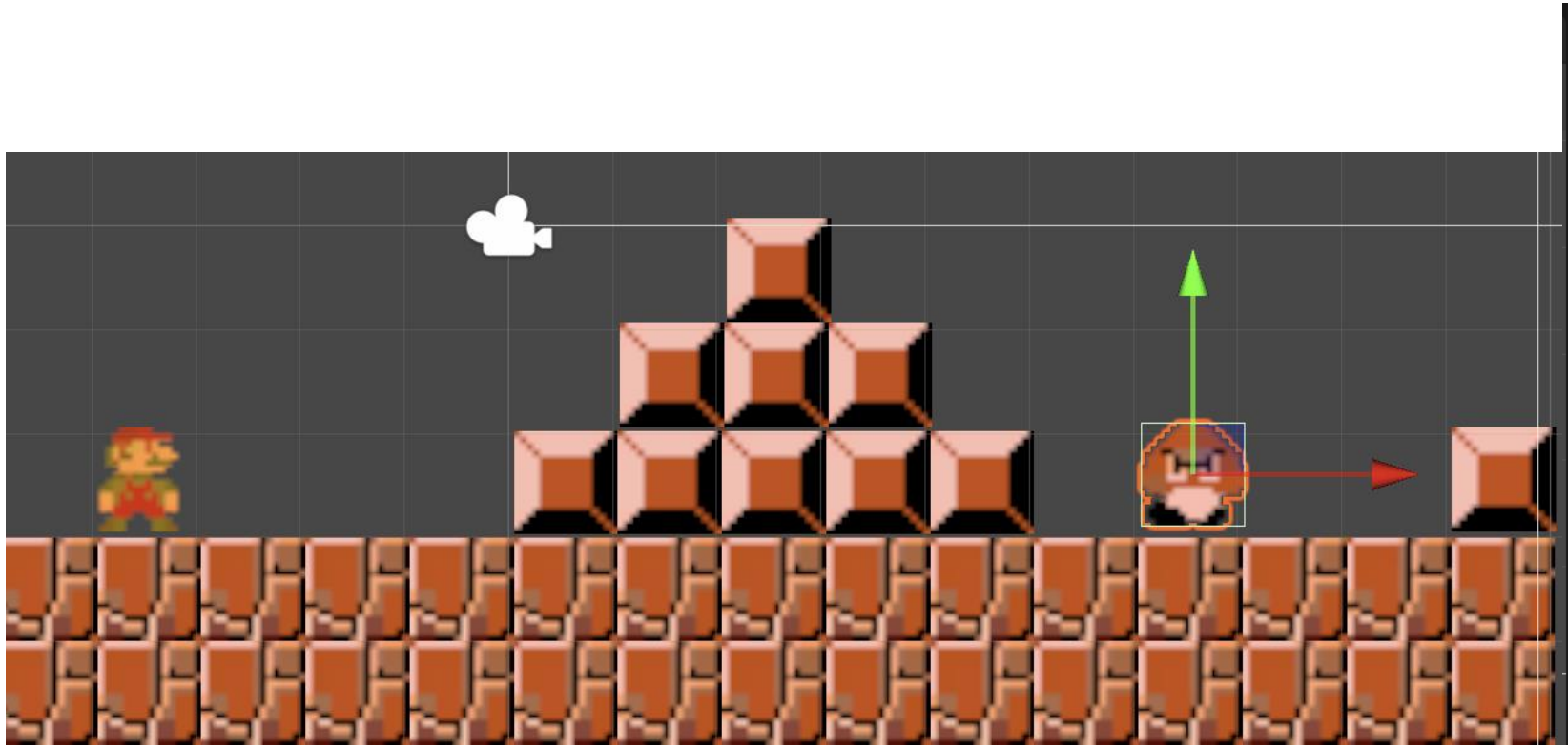
1. Level Design in Unity ✅
2. Mario Movement (left + right) ✅
3. Object detection / collision ✅
4. Jumping over blocks / gravity ✅
5. **Enemies (Goombas) alternate directions**
6. Jumping on Goombas – further collisions
7. Decoration – animation – walking
8. Camera to track Mario (once tile map created)

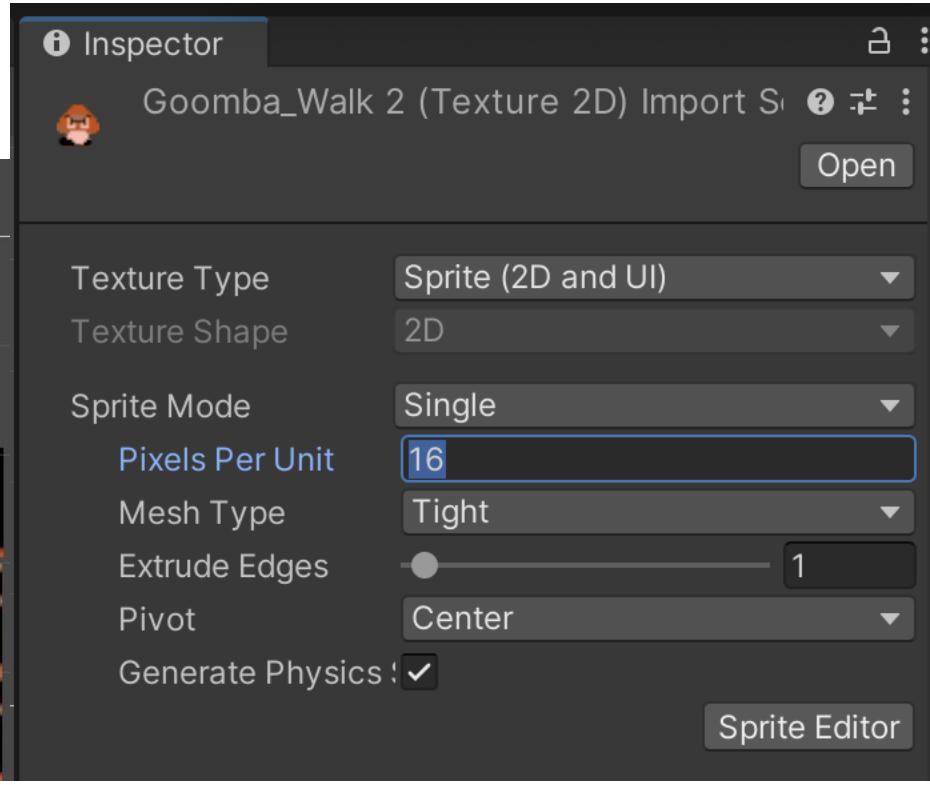
5. Goombas – set pixel per unit for asset



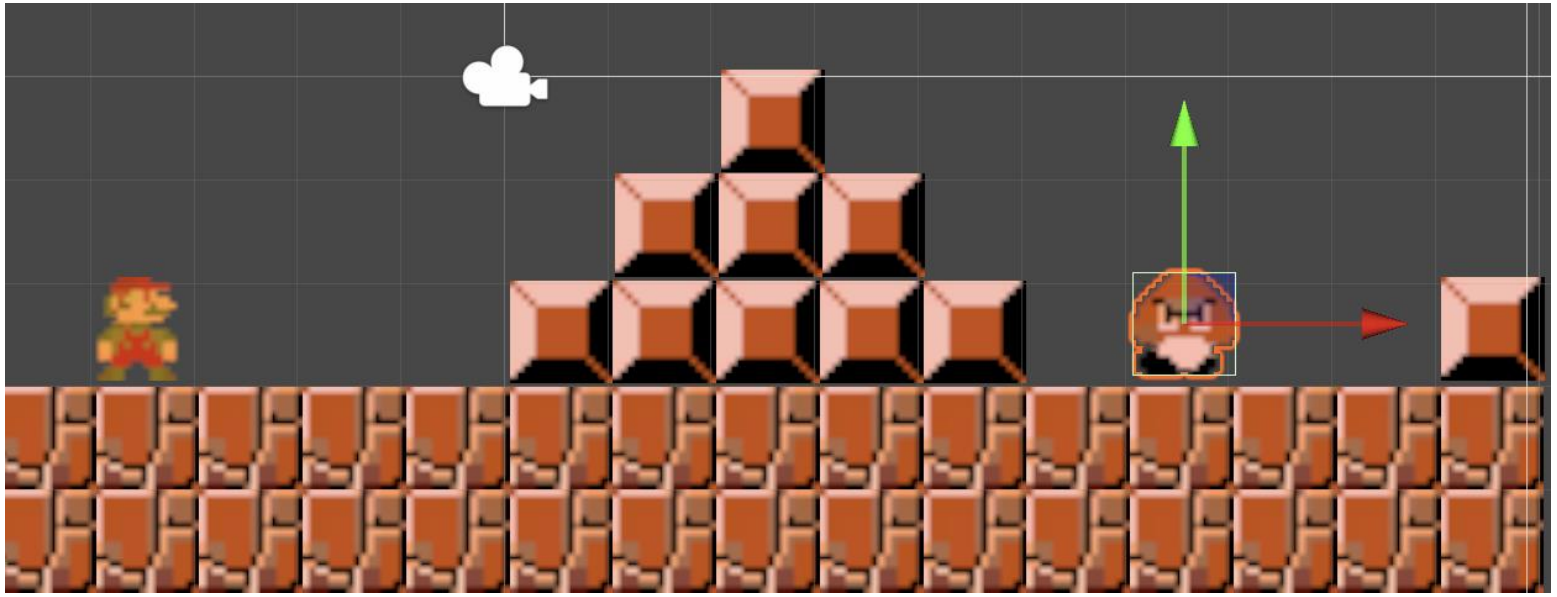


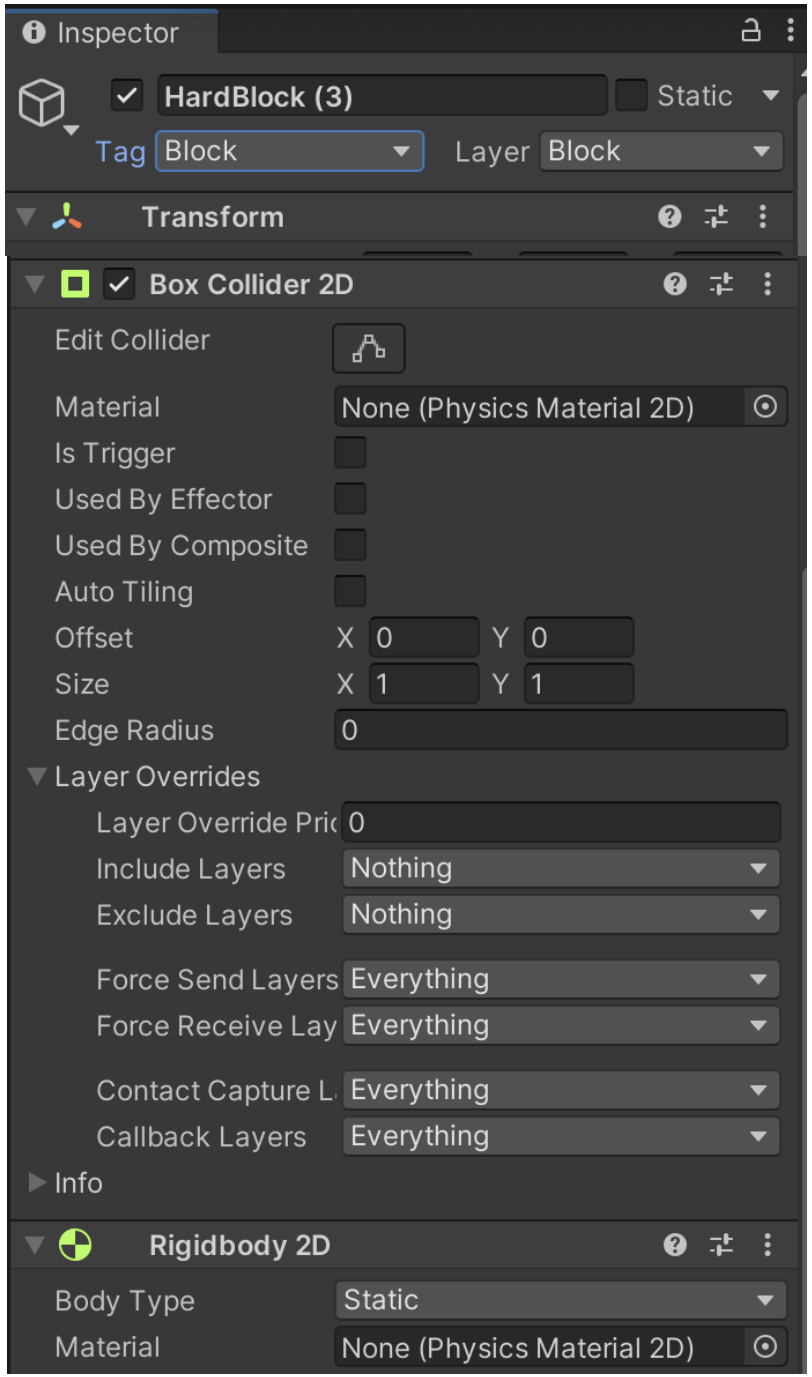
5. Goombas + Block



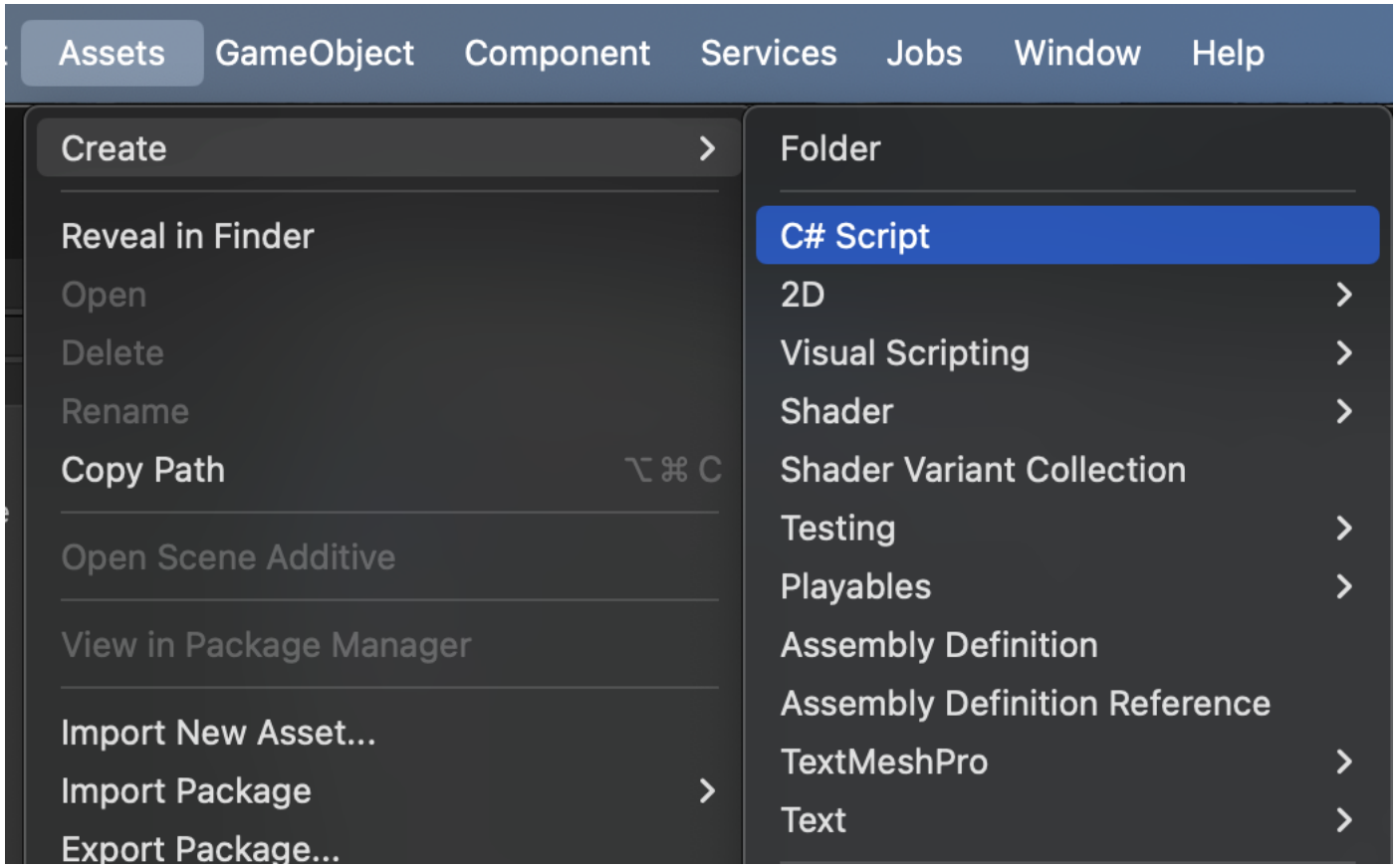


5. Goombas – add colliders





5. Goombas – add C# Script



Player.cs | MarioJump.cs | Goomba.cs

Users > nick > Documents > test_p_1 > Assets > Goomba.cs

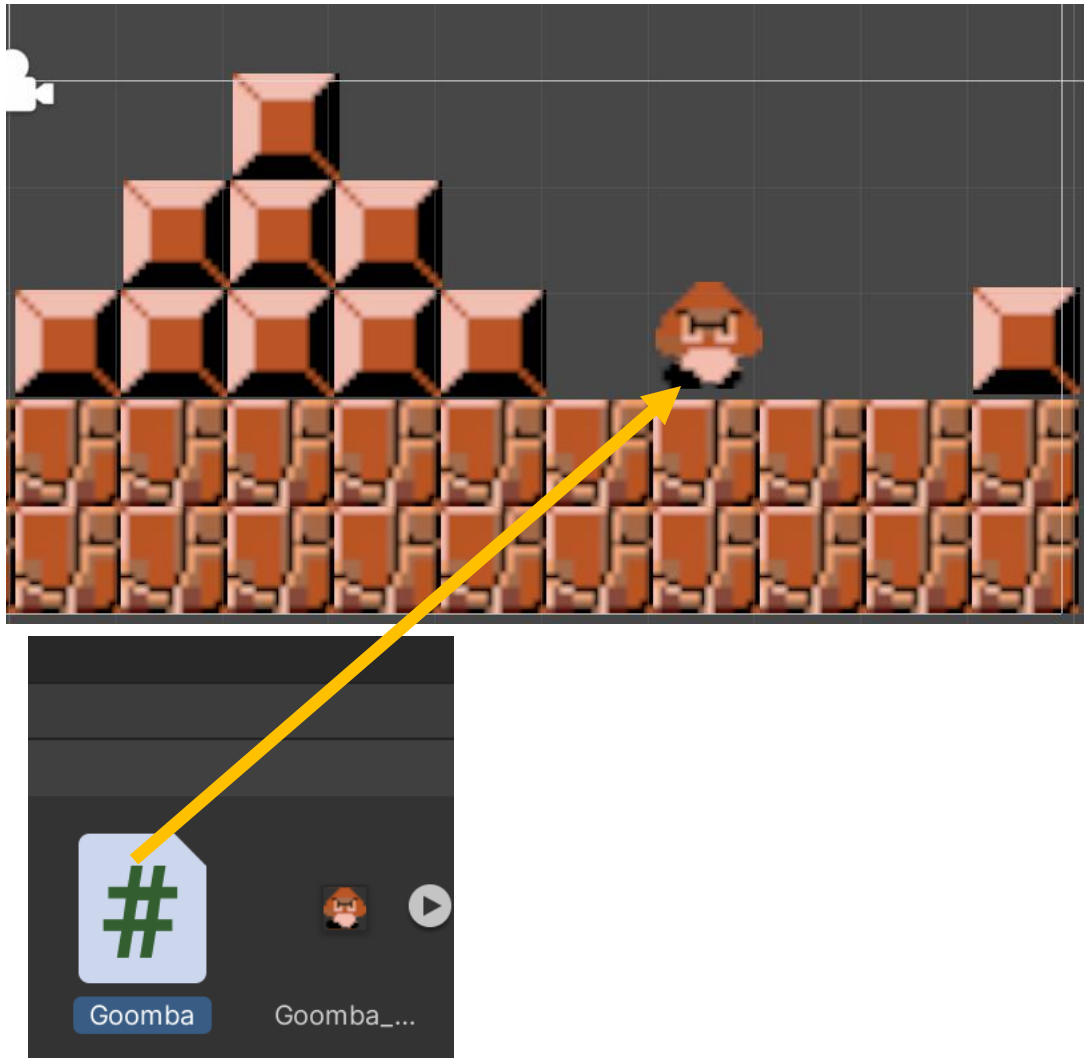
```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class Goomba : MonoBehaviour
6   {
7       public float moveSpeed = 2f; // Movement speed
8       private Rigidbody2D rb;
9       private Vector2 movementDirection = Vector2.left; // Start by moving left
10
11      void Start()
12      {
13          rb = GetComponent<Rigidbody2D>();
14      }
15
16      void FixedUpdate()
17      {
18          // Move the Goomba in the current direction
19          rb.velocity = new Vector2(movementDirection.x * moveSpeed, rb.velocity.y);
20      }
21
```

Player.cs | MarioJump.cs | Goomba.cs

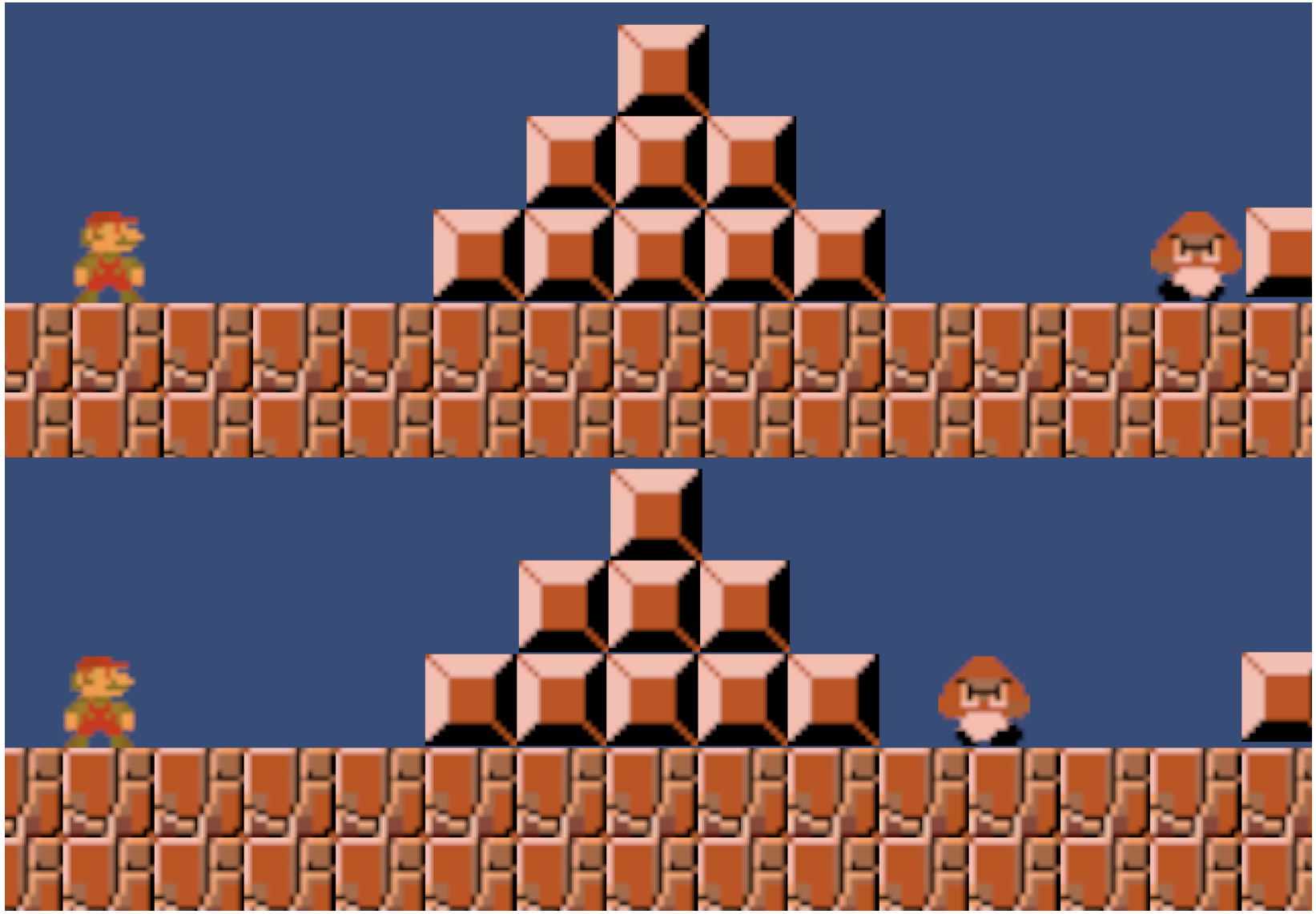
Users > nick > Documents > test_p_1 > Assets > Goomba.cs

```
6   {
21
22      void OnCollisionEnter2D(Collision2D collision)
23      {
24          // Check for collision with "Block" or walls
25          if (collision.gameObject.CompareTag("Block") || collision.gameObject.CompareTag("Wall"))
26          {
27              // Reverse the movement direction
28              movementDirection = -movementDirection;
29
30              // Optionally, flip the sprite for correct direction
31              FlipSprite();
32          }
33      }
34
35      void FlipSprite()
36      {
37          Vector3 scale = transform.localScale;
38          scale.x *= -1; // Flip horizontally
39          transform.localScale = scale;
40      }
41  }
42
```

5. Goombas – add script



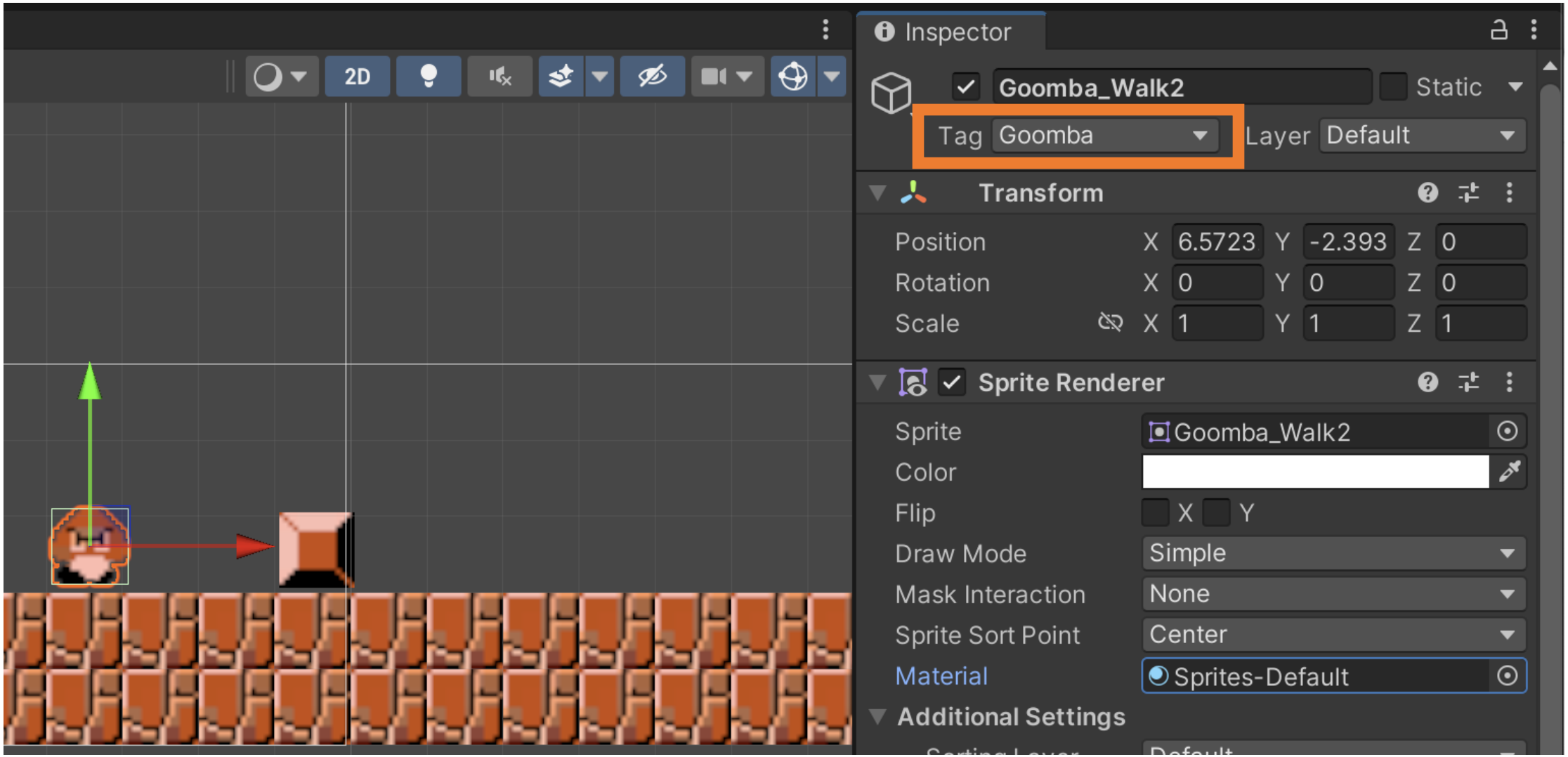
5. Goombas – alternate direction



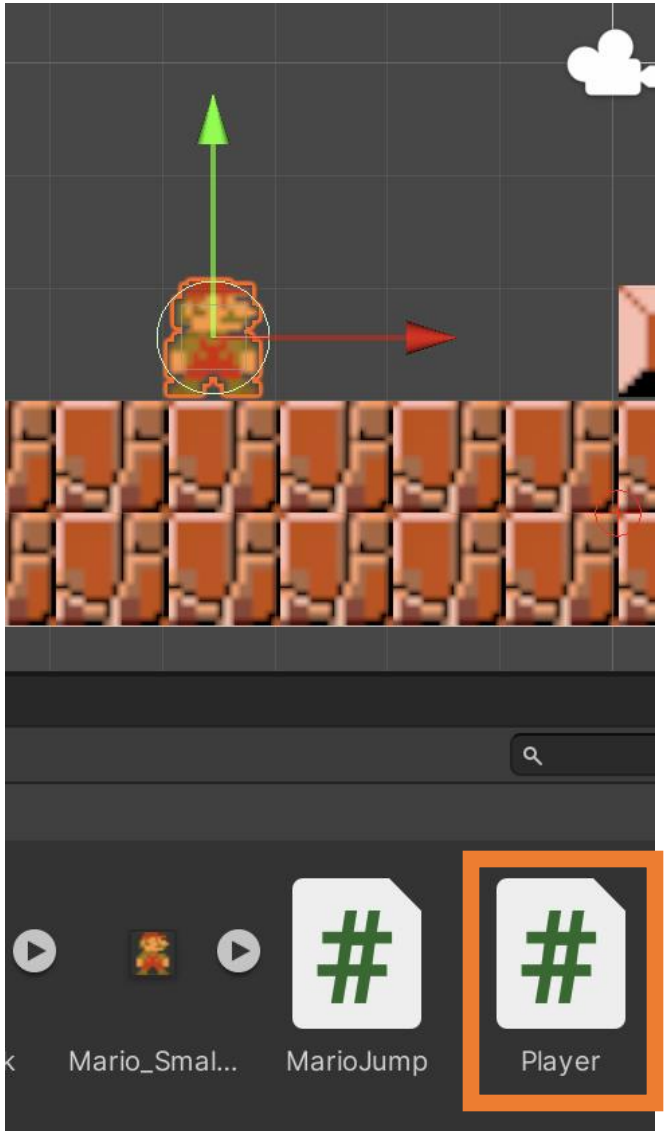
Steps / Requirements

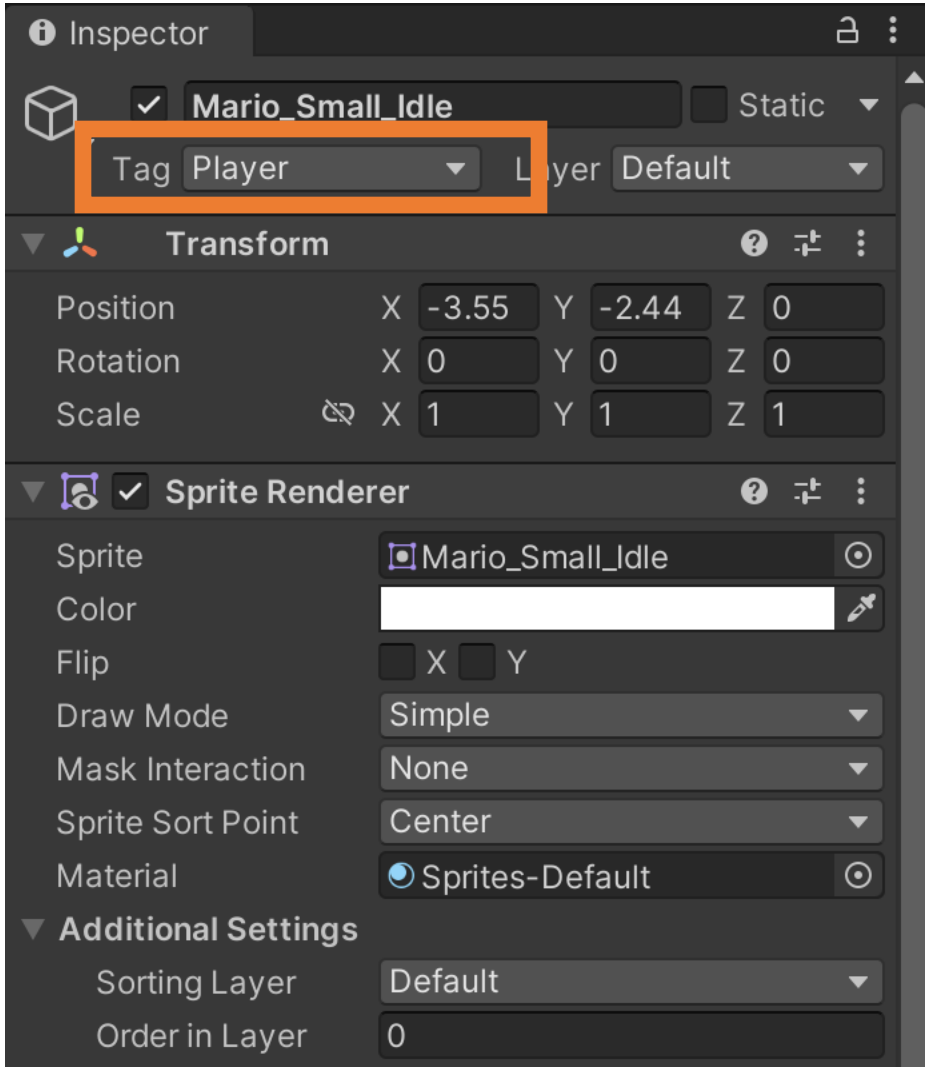
1. Level Design in Unity ✅
2. Mario Movement (left + right) ✅
3. Object detection / collision ✅
4. Jumping over blocks / gravity ✅
5. Enemies (Goombas) alternate directions ✅
6. **Jumping on Goombas – further collisions**
7. Decoration – animation – walking
8. Camera to track Mario (once tile map created)

6. Jumping on Goombas – set tags



6. Jumping on Goombas – modify the collision



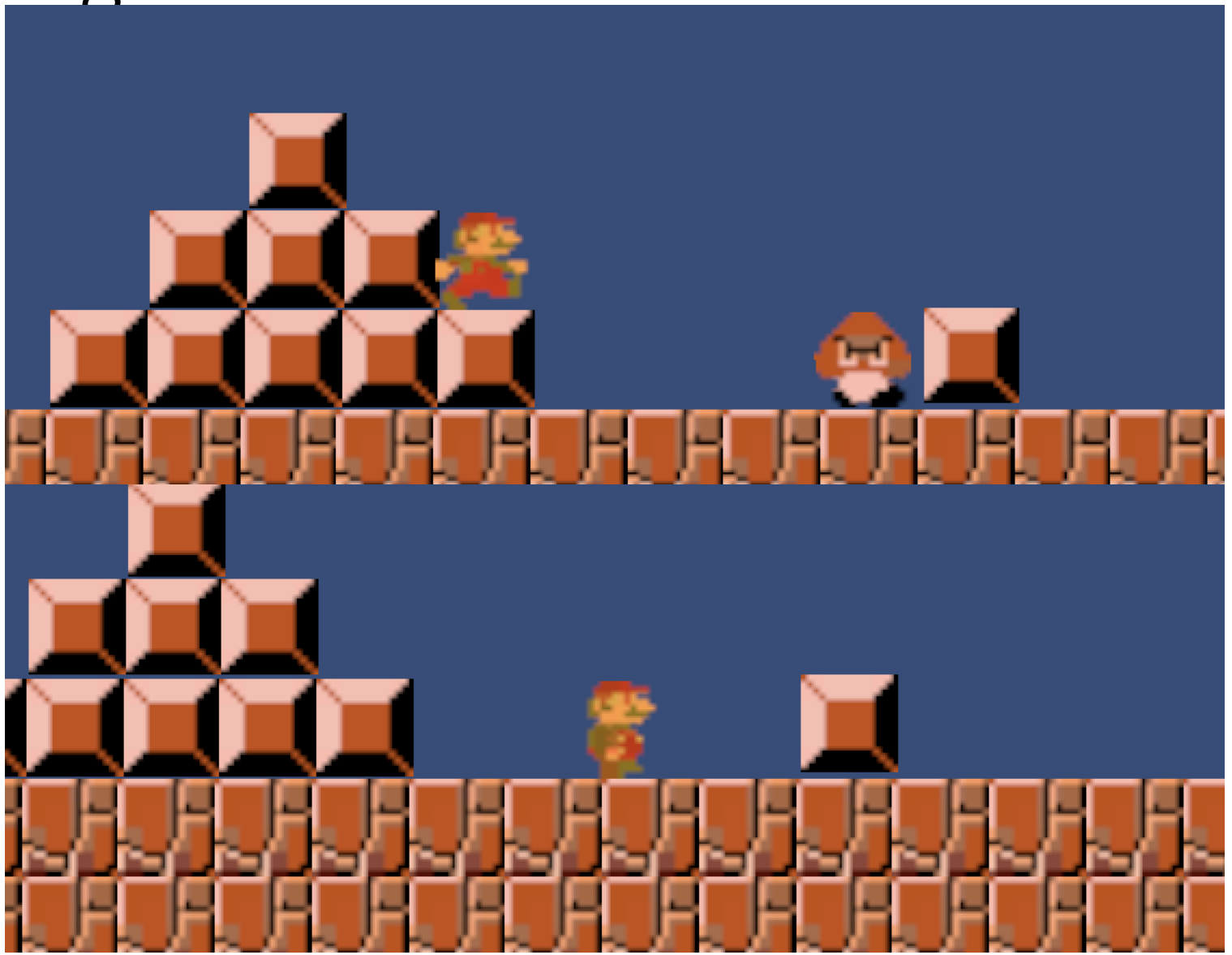


Player.cs

Users > nick > Documents > test_p_1 > Assets > Player.cs > ...

```
7   public class Player : MonoBehaviour
        0 references
43      void OnCollisionEnter2D(Collision2D collision)
44      {
45          // Check if Mario collides with a Goomba
46          if (collision.gameObject.CompareTag("Goomba"))
47          {
48              // Determine if Mario is above the Goomba (stomp)
49              float marioHeight = transform.position.y;
50              float goombaHeight = collision.transform.position.y;
51
52              if (marioHeight > goombaHeight + 0.5f) // Adjust the offset as needed
53              {
54                  Debug.Log("Mario stomps the Goomba!");
55                  Destroy(collision.gameObject); // Remove the Goomba
56              }
57              else
58              {
59                  Debug.Log("Mario is hit by the Goomba!");
60                  // Trigger Mario's damage logic, e.g., lose a life
61                  HandleMarioDamage();
62              }
63          }
64      }
```

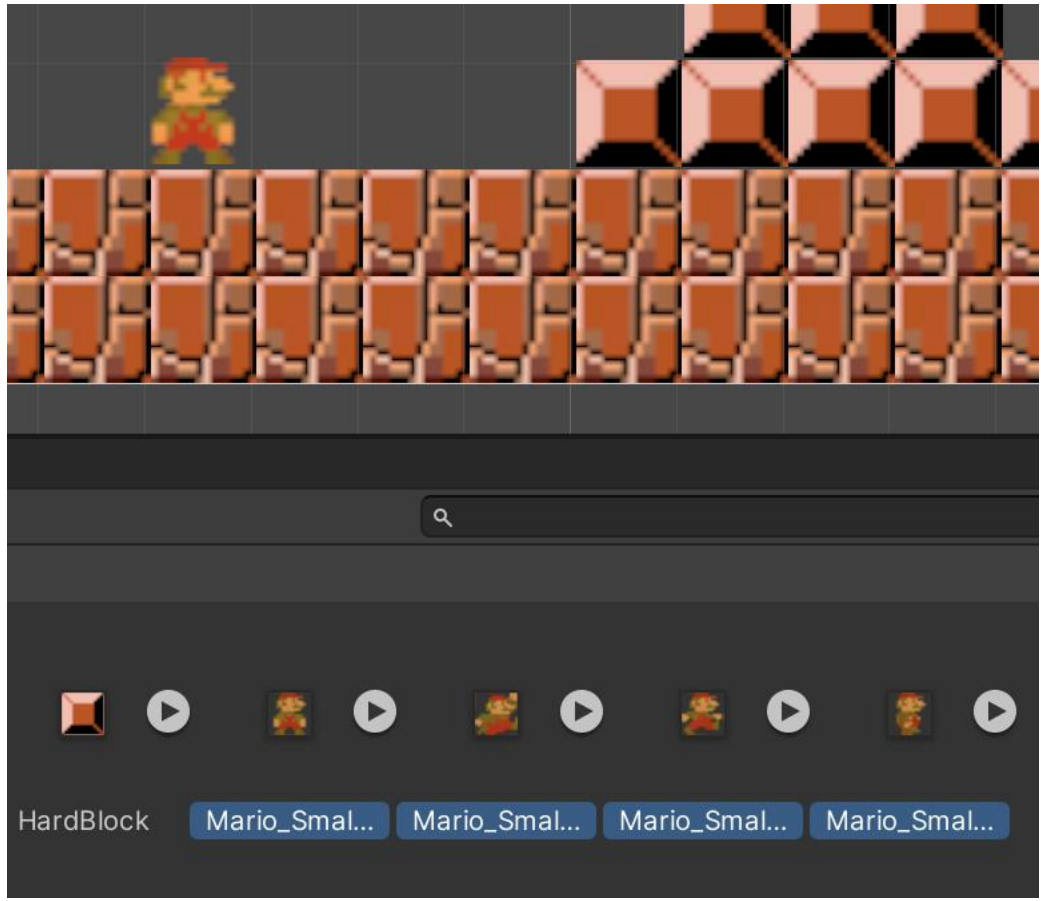
6. Jumping on Goombas

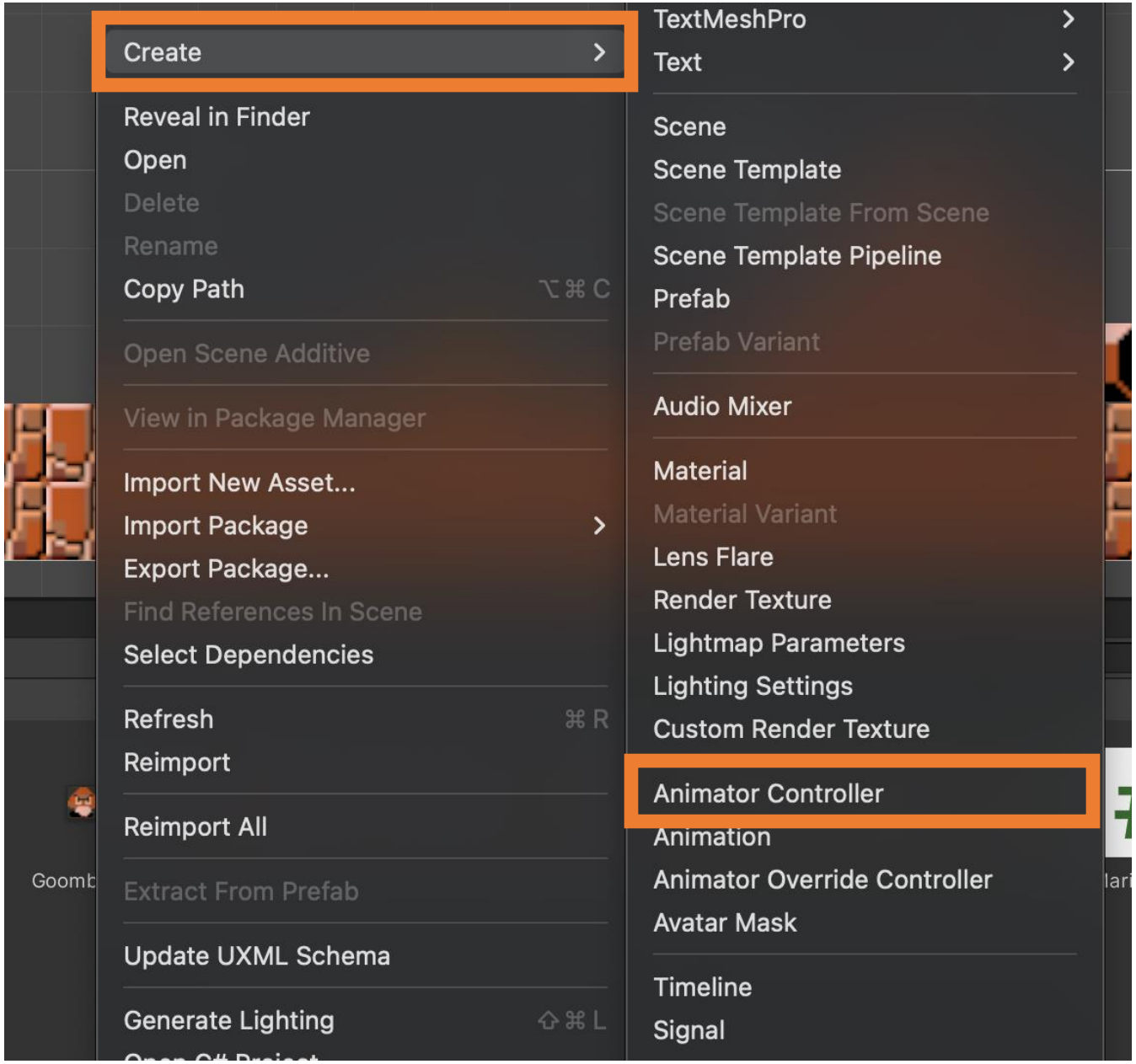


Steps / Requirements

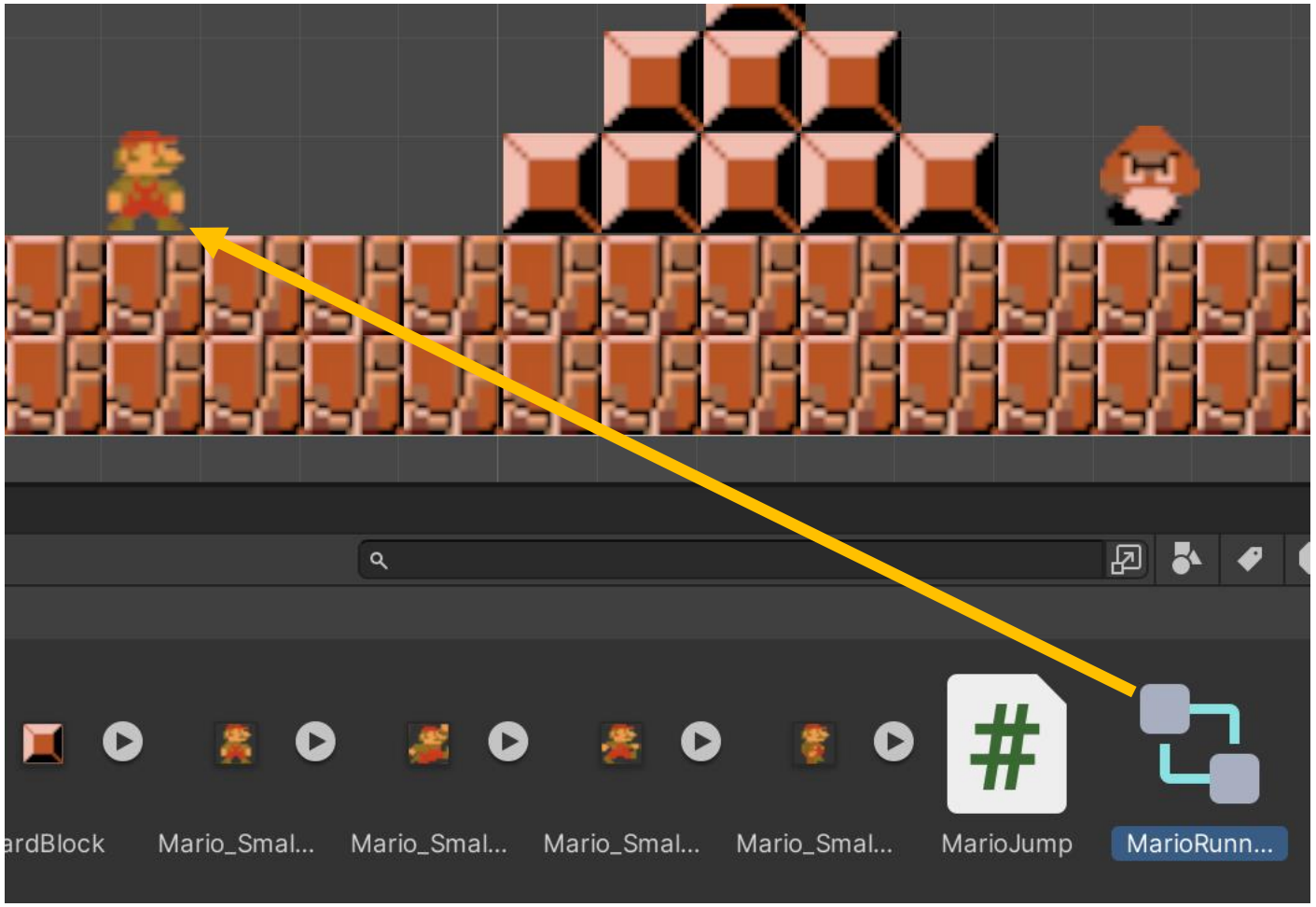
1. Level Design in Unity ✅
2. Mario Movement (left + right) ✅
3. Object detection / collision ✅
4. Jumping over blocks / gravity ✅
5. Enemies (Goombas) alternate directions ✅
6. Jumping on Goombas – further collisions ✅
7. **Decoration – animation – walking**
8. Camera to track Mario (once tile map created)

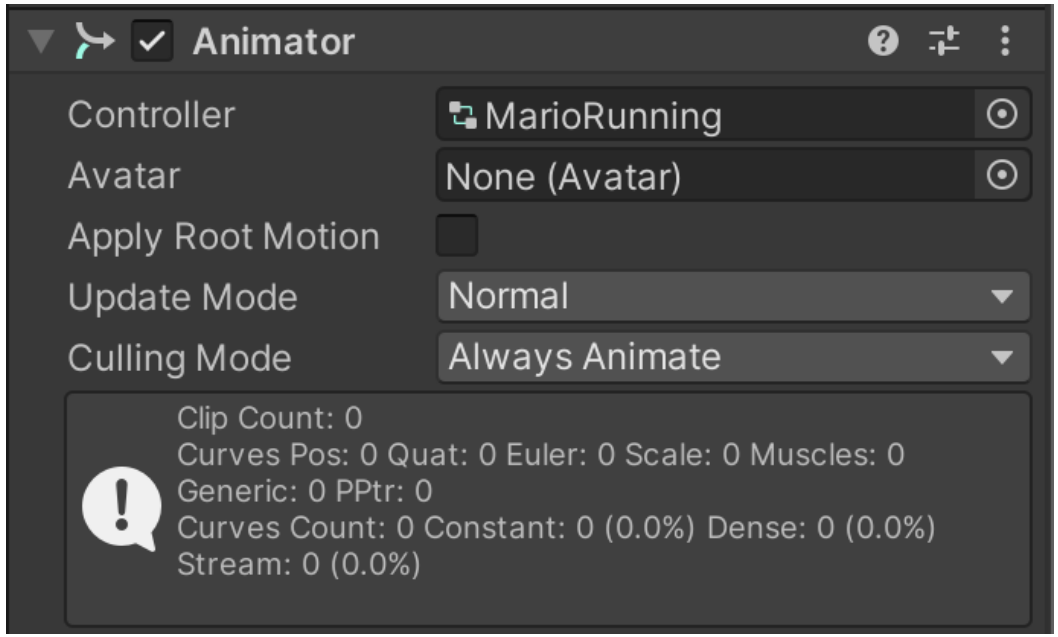
7. Animation

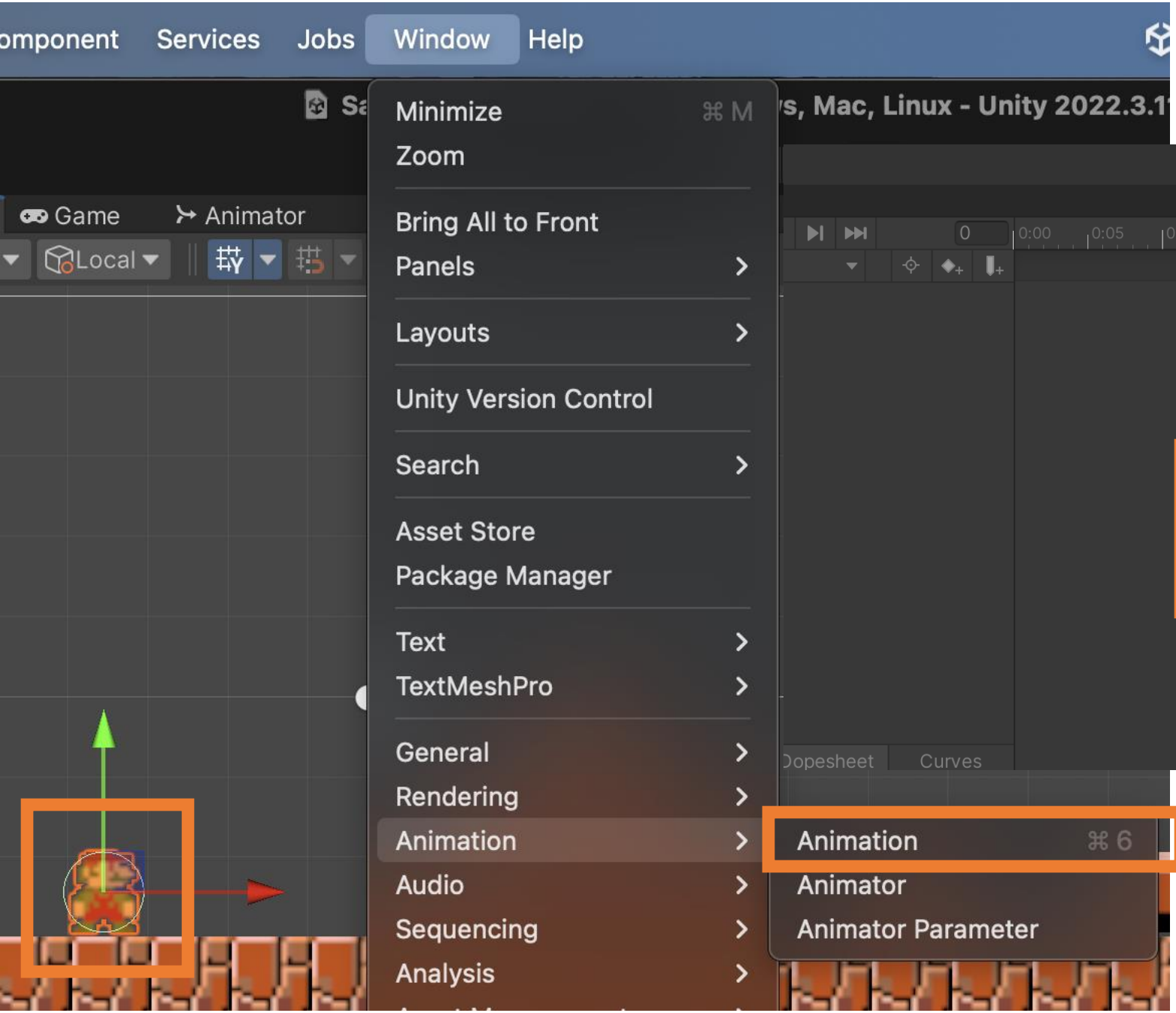

HardBlock
Mario_Smal...
Mario_Smal...
Mario_Smal...
Mario_Smal...


Create
Reveal in Finder
Open
Delete
Rename
Copy Path ⌥⌘C
Open Scene Additive
View in Package Manager
Import New Asset...
Import Package
Export Package...
Find References In Scene
Select Dependencies
Refresh ⌘R
Reimport
Reimport All
Extract From Prefab
Update UXML Schema
Generate Lighting ⇧⌘L
TextMeshPro
Text
Scene
Scene Template
Scene Template From Scene
Scene Template Pipeline
Prefab
Prefab Variant
Audio Mixer
Material
Material Variant
Lens Flare
Render Texture
Lightmap Parameters
Lighting Settings
Custom Render Texture
Animator Controller
Animation
Animator Override Controller
Avatar Mask
Timeline
Signal

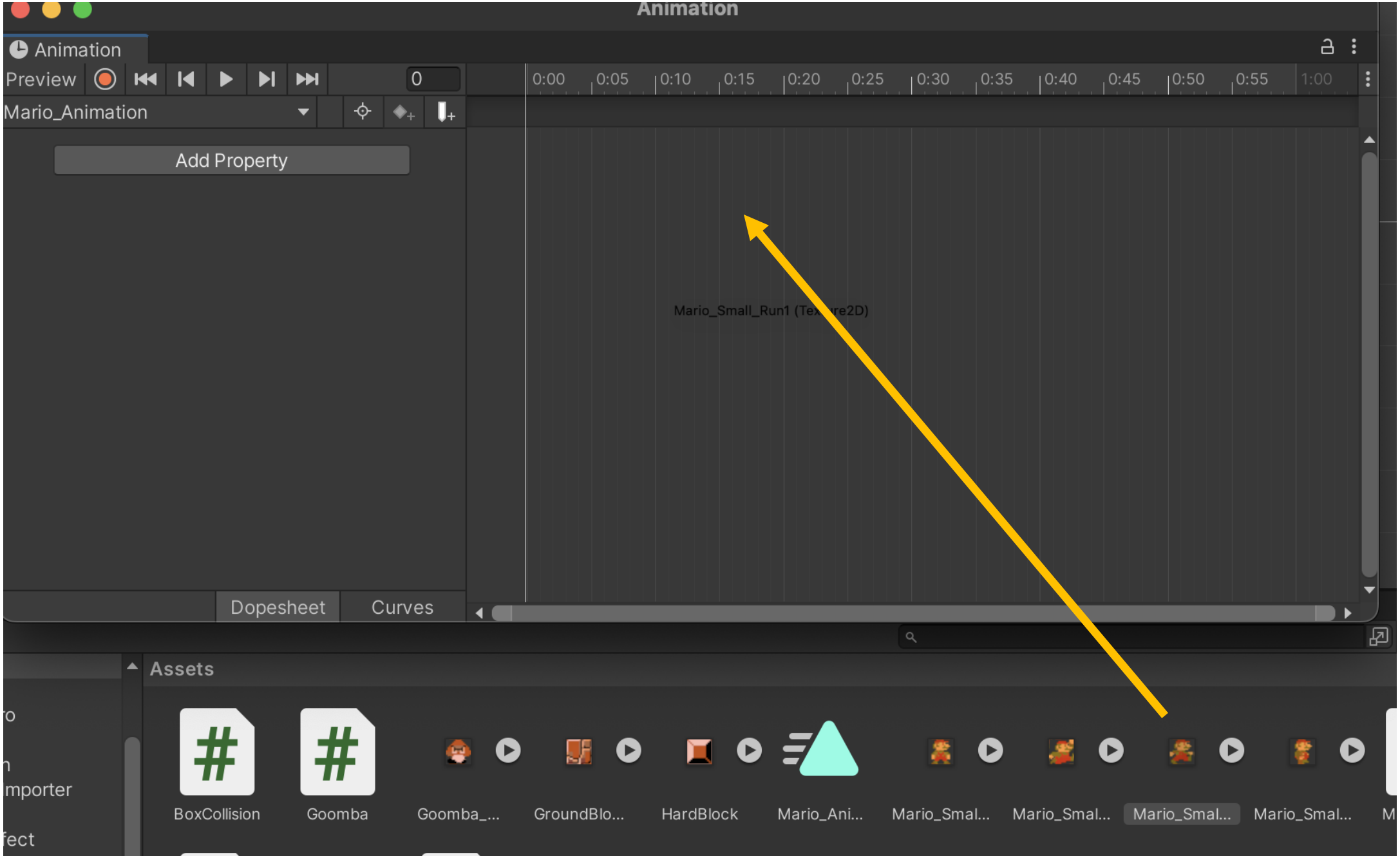
7. Animation


ardBlock
Mario_Smal...
Mario_Smal...
Mario_Smal...
Mario_Smal...
MarioJump
MarioRunn...

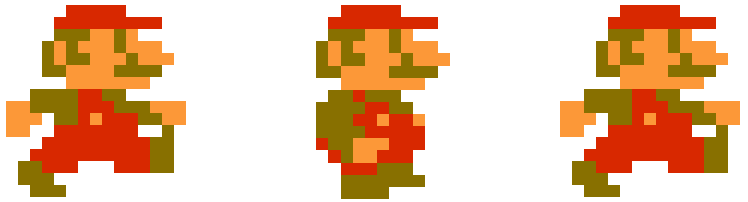

Animator
Controller
MarioRunning
Avatar
None (Avatar)
Apply Root Motion
Update Mode
Normal
Culling Mode
Always Animate
Clip Count: 0
Curves Pos: 0 Quat: 0 Euler: 0 Scale: 0 Muscles: 0
Generic: 0 PPtr: 0
Curves Count: 0 Constant: 0 (0.0%) Dense: 0 (0.0%)
Stream: 0 (0.0%)

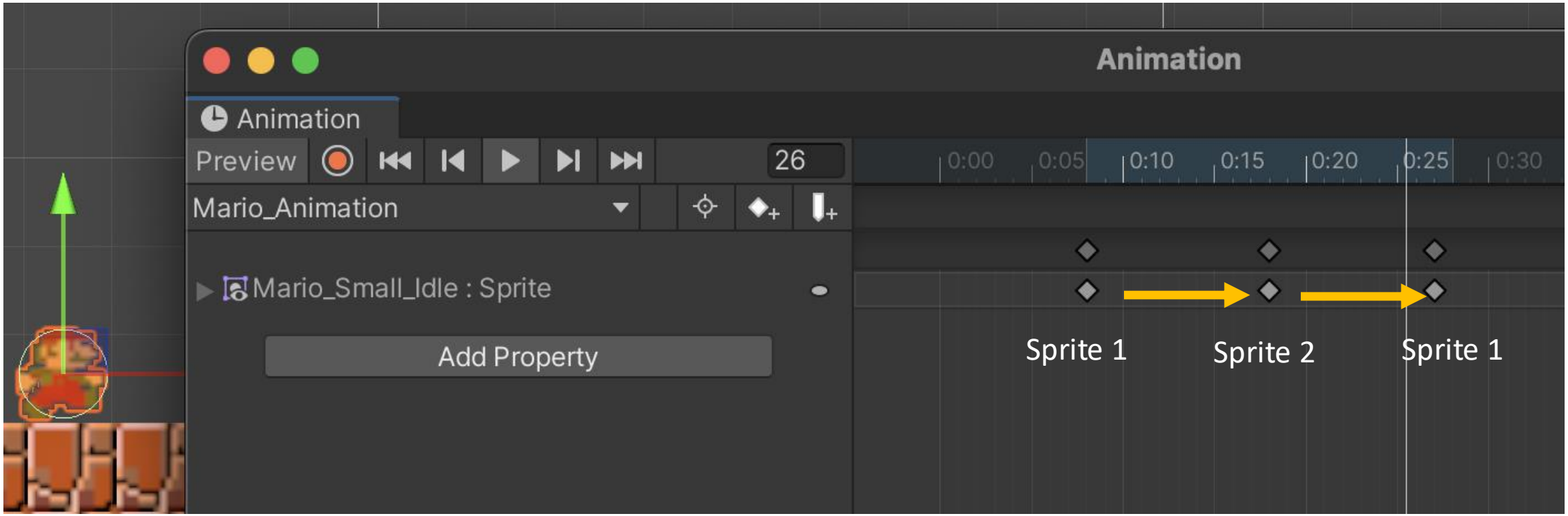


Make sure you have Mario selected as you open the ‘Animation’ tab


Animation
Animation
Preview
0
Mario_Animation
Add Property
0:00
0:05
0:10
0:15
0:20
0:25
0:30
0:35
0:40
0:45
0:50
0:55
1:00
Mario_Small_Run1 (Texture2D)
Dopesheet
Curves
Assets
BoxCollision
Goomba
Goomba_...
GroundBlo...
HardBlock
Mario_Ani...
Mario_Smal...
Mario_Smal...
Mario_Smal...
Mario_Smal...

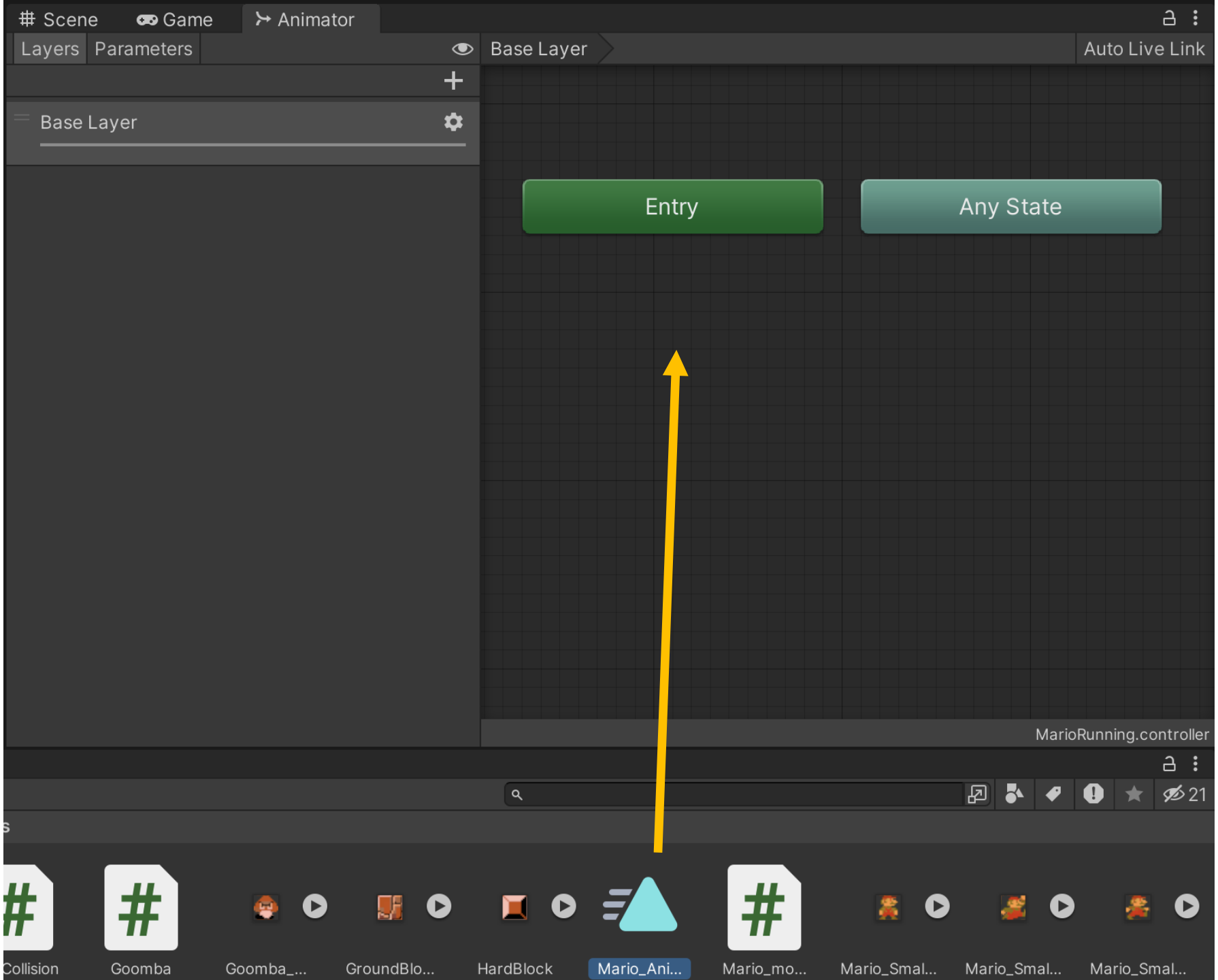
7. Animation

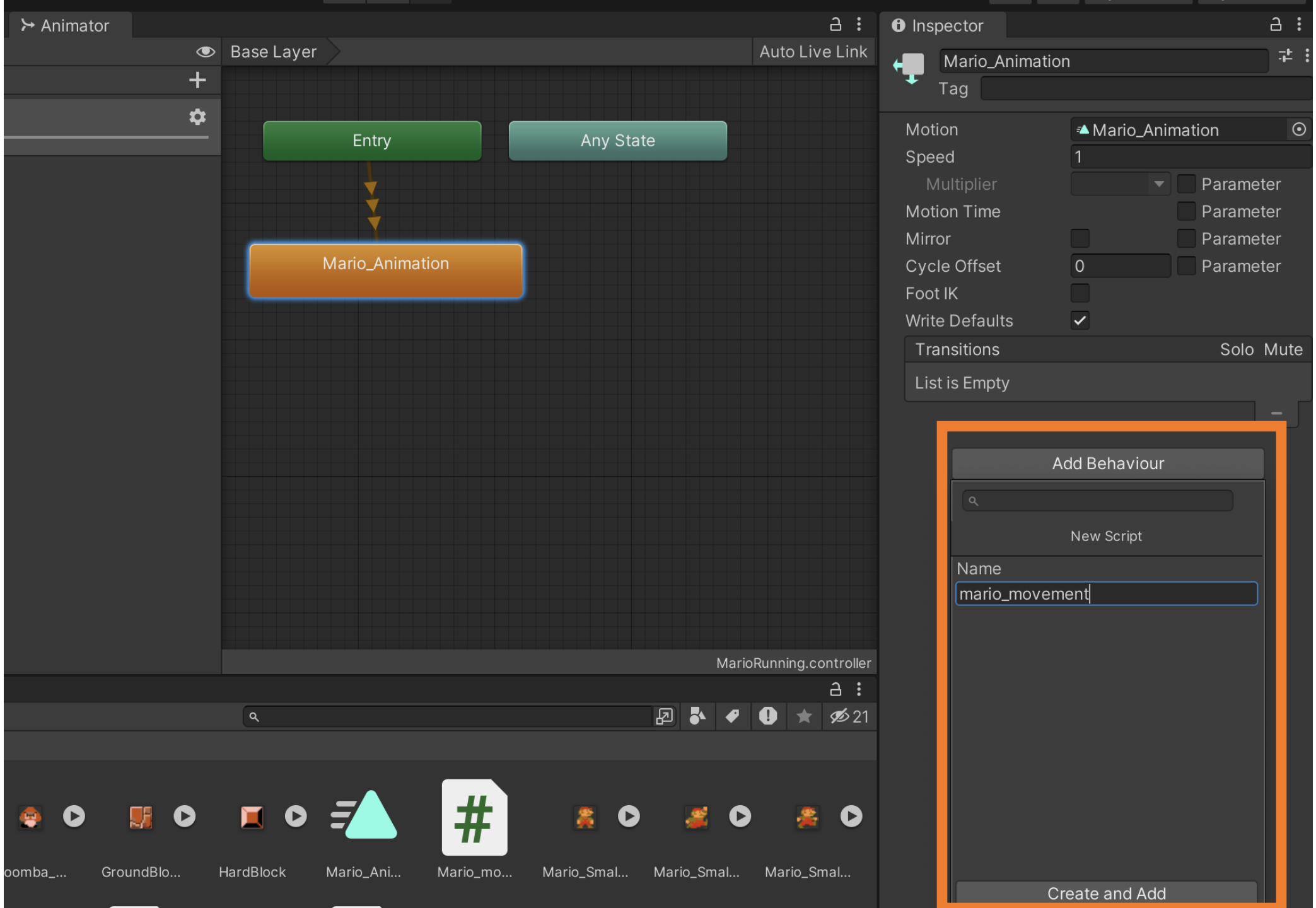


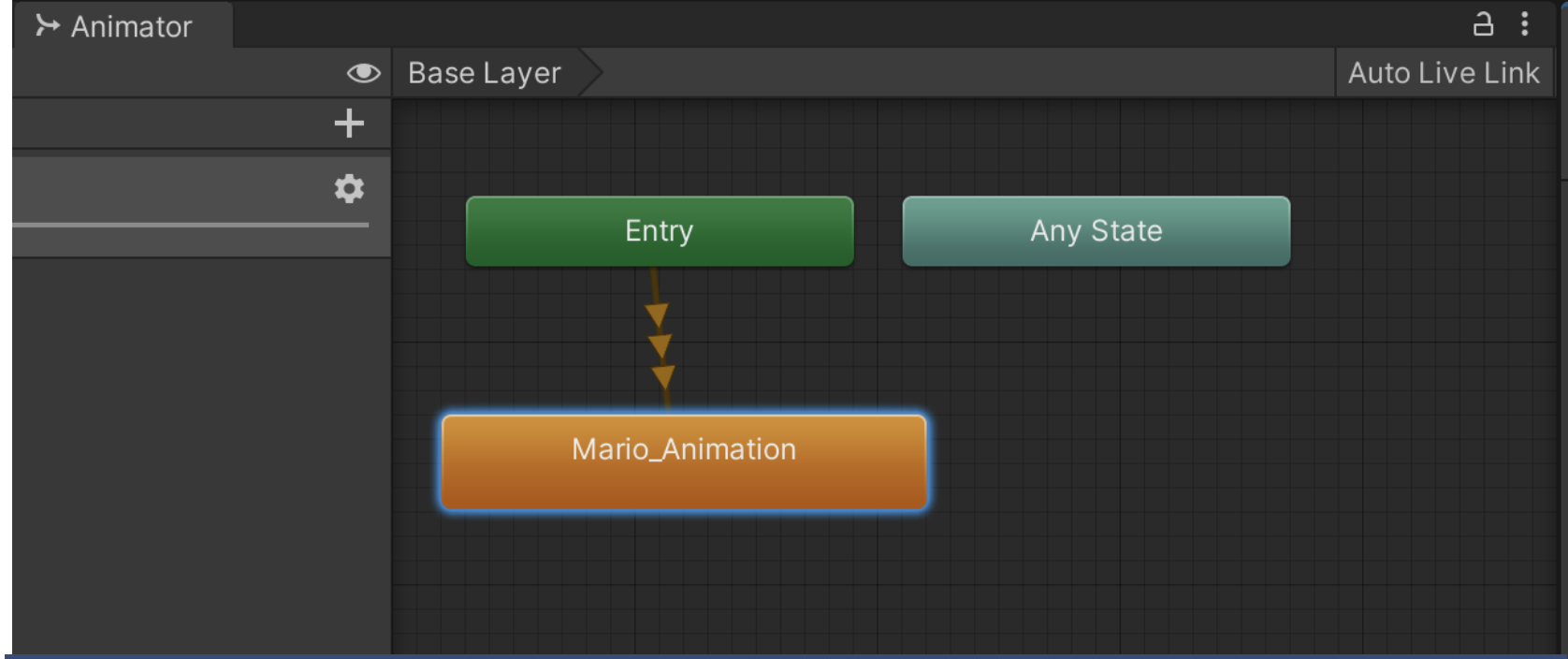


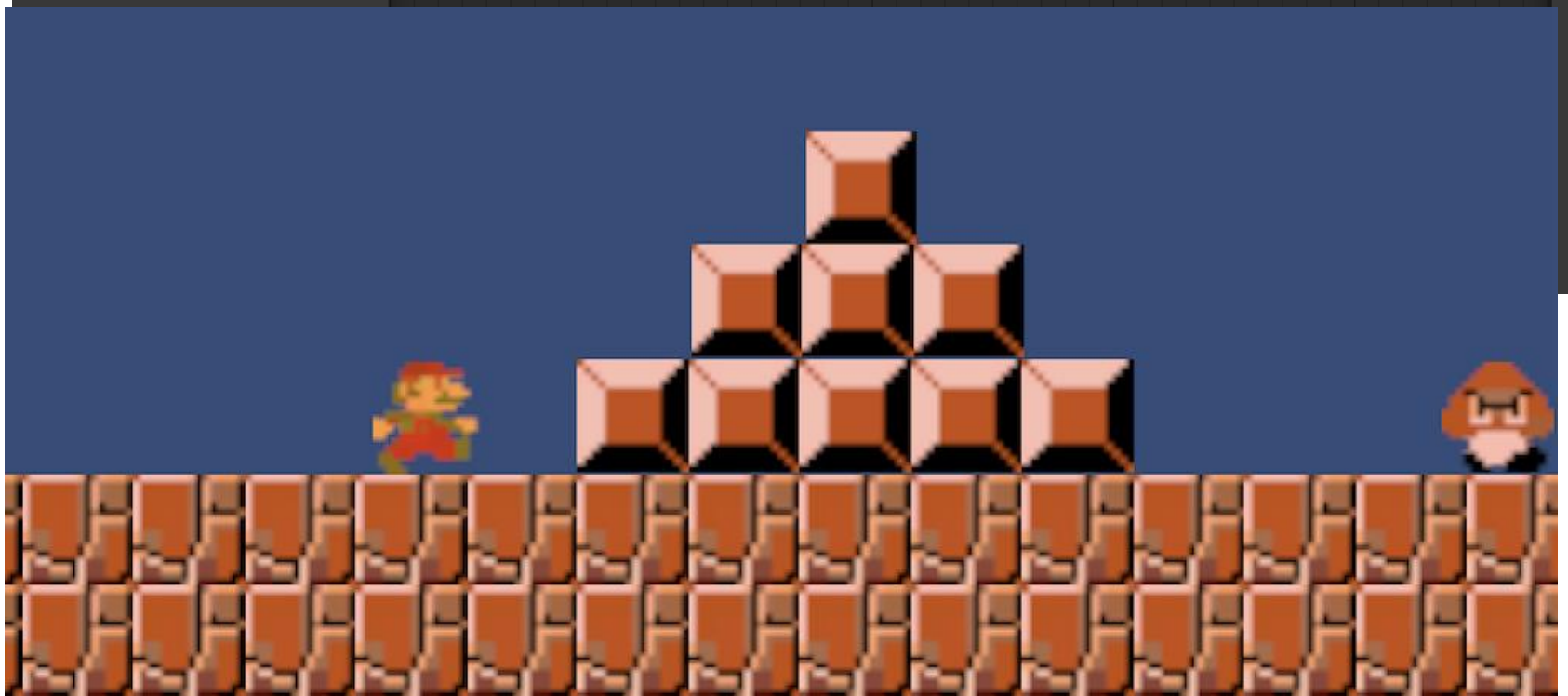
I found I needed a ‘copy’ of the first sprite at the end for a ‘smoother’ animation
This is so Sprite 1 and Sprite 2 had roughly the same time in the loop.

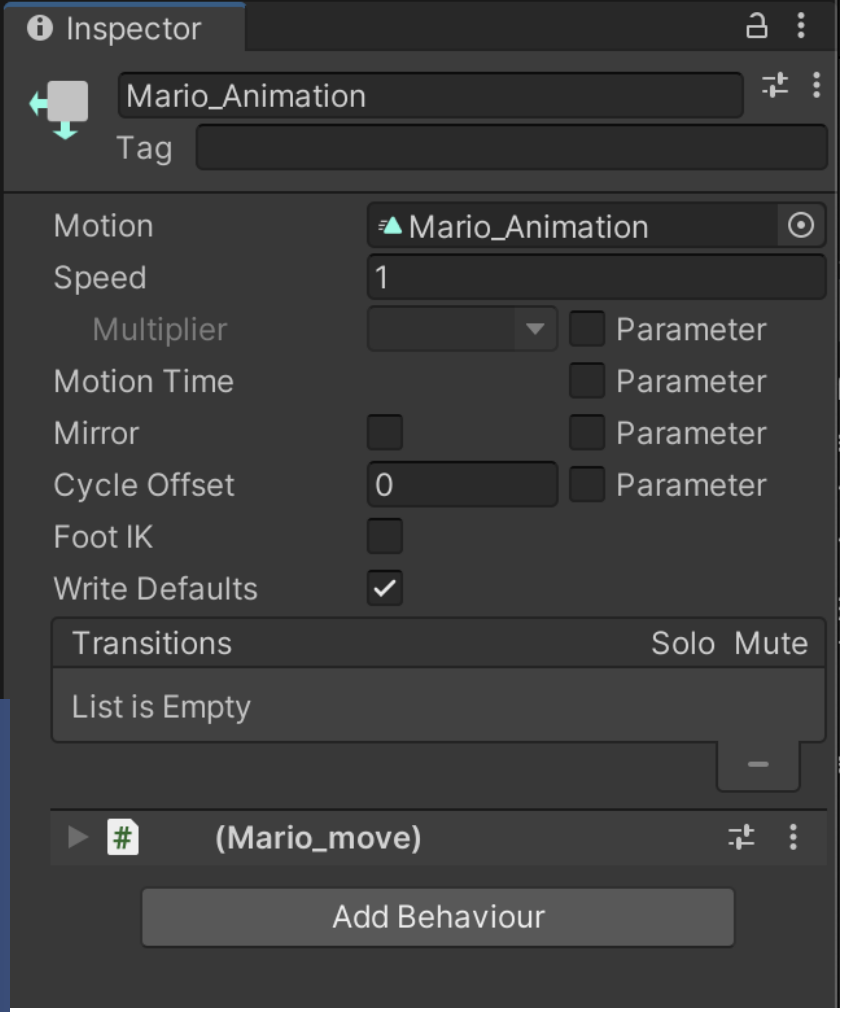
7.


Scene
Game
Animator
Layers
Parameters
Base Layer
Auto Live Link
Base Layer
Entry
Any State
MarioRunning.controller
21
Collision
Goomba
Goomba_...
GroundBlo...
HardBlock
Mario_Ani...
Mario_mo...
Mario_Smal...
Mario_Smal...
Mario_Smal...


Animator
Base Layer
Auto Live Link
Entry
Any State
Mario_Animation
MarioRunning.controller
Inspector
Mario_Animation
Tag
Motion
Mario_Animation
Speed
1
Multiplier
Parameter
Motion Time
Parameter
Mirror
Parameter
Cycle Offset
0
Parameter
Foot IK
Write Defaults
Transitions
Solo
Mute
List is Empty
Add Behaviour
New Script
Name
mario_movement
Create and Add
21
oomba_...
GroundBlo...
HardBlock
Mario_Ani...
Mario_mo...
Mario_Smal...
Mario_Smal...
Mario_Smal...


Animator
Base Layer
Auto Live Link
Entry
Any State
Mario_Animation

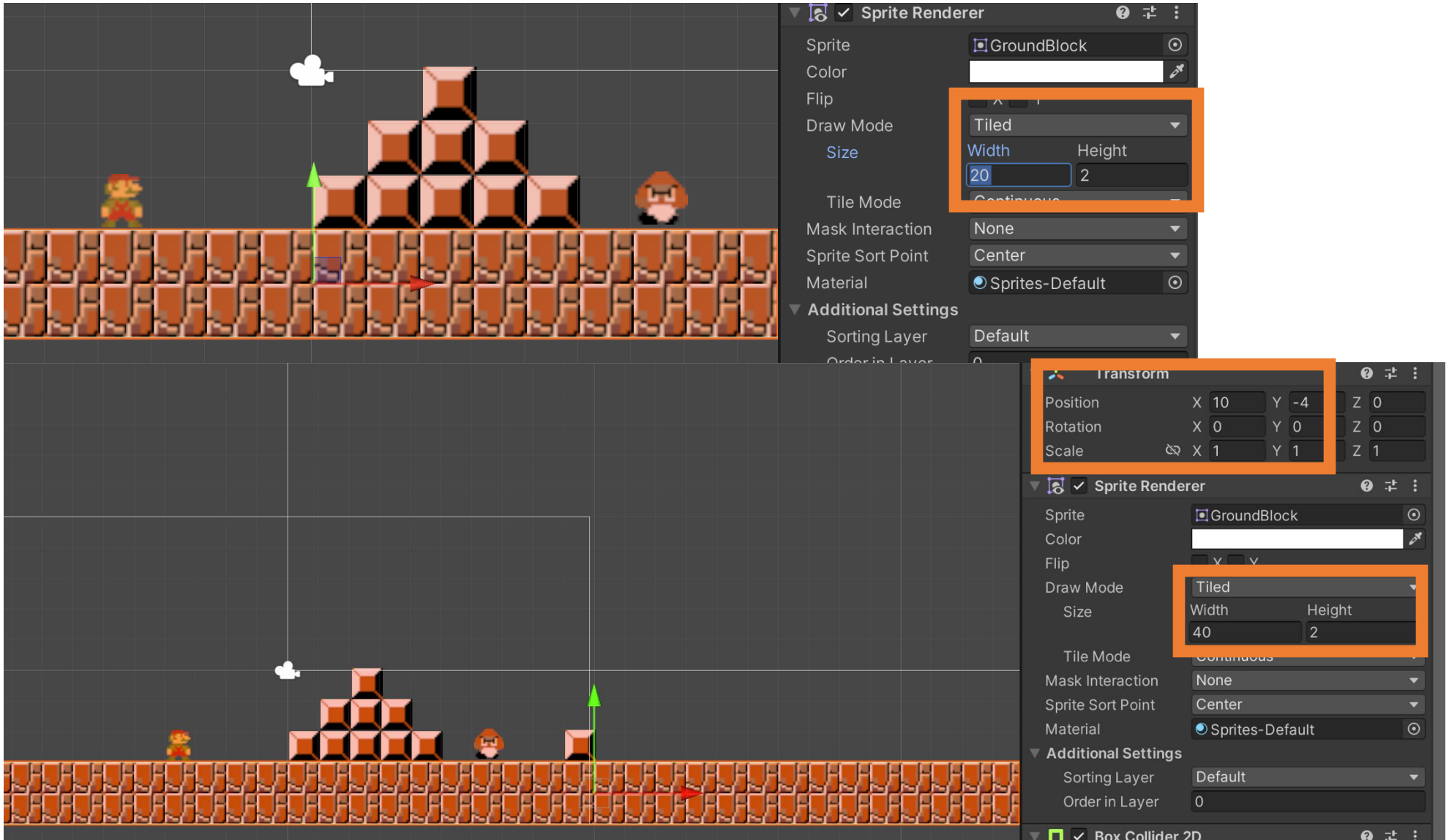



Inspector
Mario_Animation
Tag
Motion
Mario_Animation
Speed
1
Multiplier
Parameter
Motion Time
Parameter
Mirror
Parameter
Cycle Offset
0
Parameter
Foot IK
Write Defaults
Transitions
Solo
Mute
List is Empty
(Mario_move)
Add Behaviour

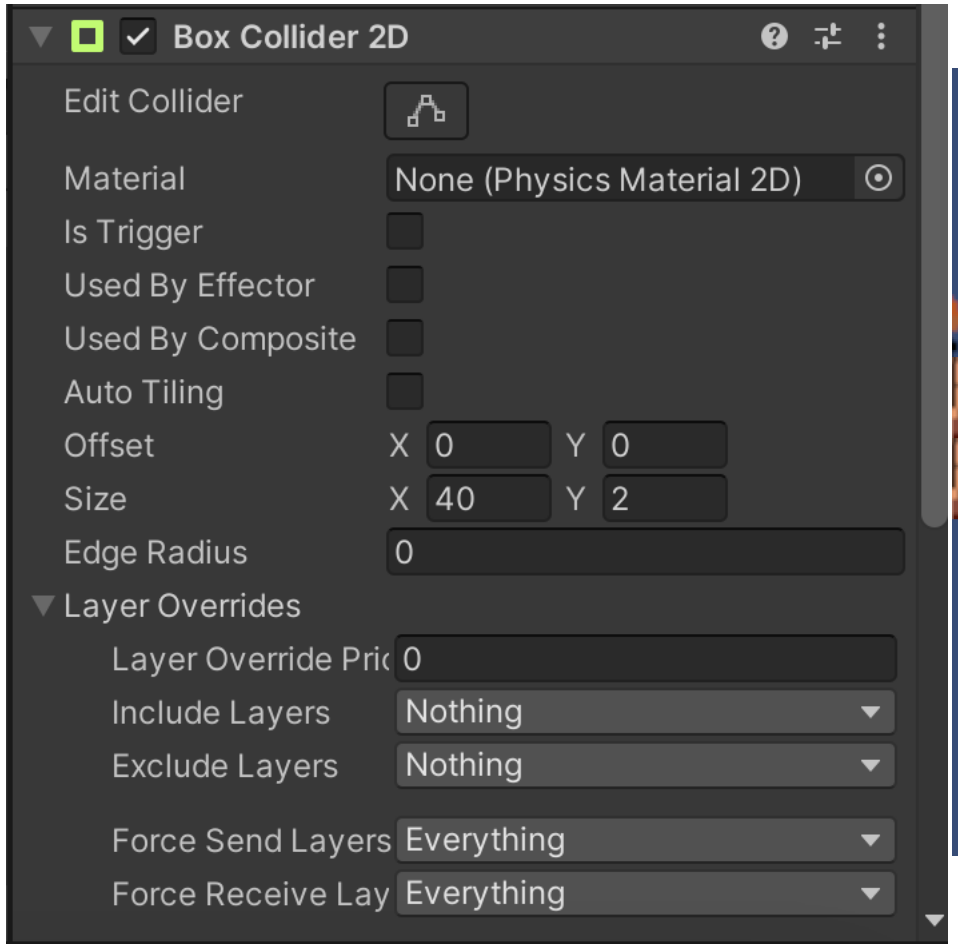
Steps / Requirements

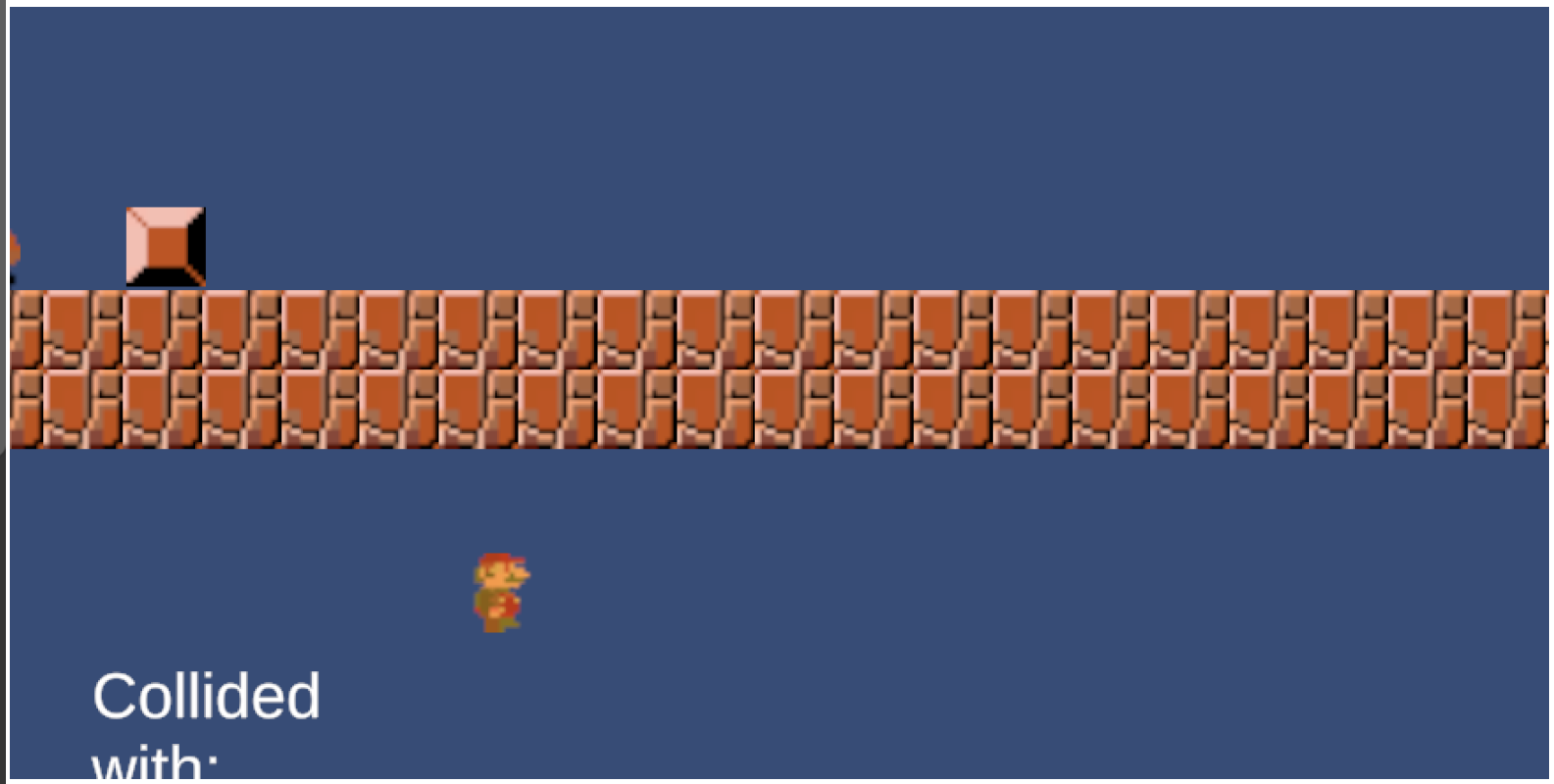
1. Level Design in Unity ✅
2. Mario Movement (left + right) ✅
3. Object detection / collision ✅
4. Jumping over blocks / gravity ✅
5. Enemies (Goombas) alternate directions ✅
6. Jumping on Goombas – further collisions ✅
7. Decoration – animation – walking ✅
8. **Camera to track Mario (once tile map created)**

8. Camera – first extend the floor

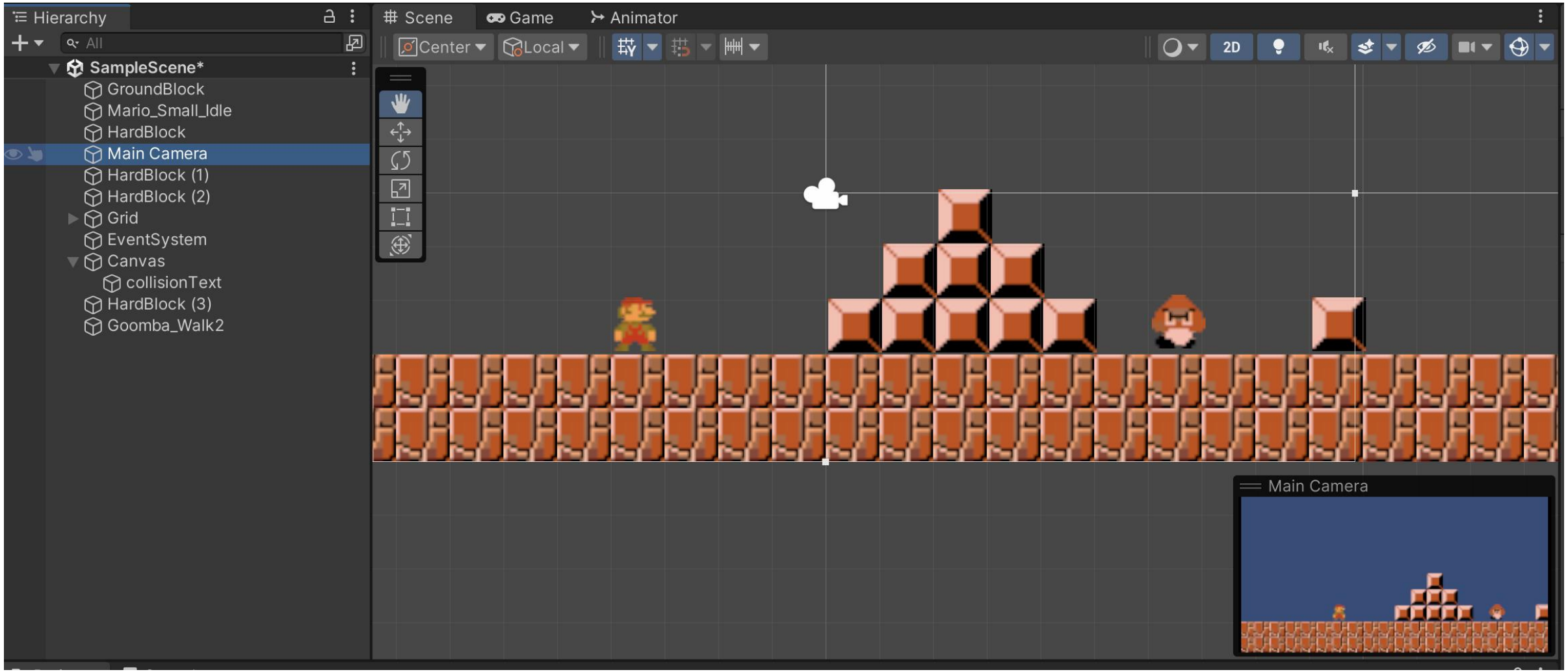


8. Camera – update your box collider!

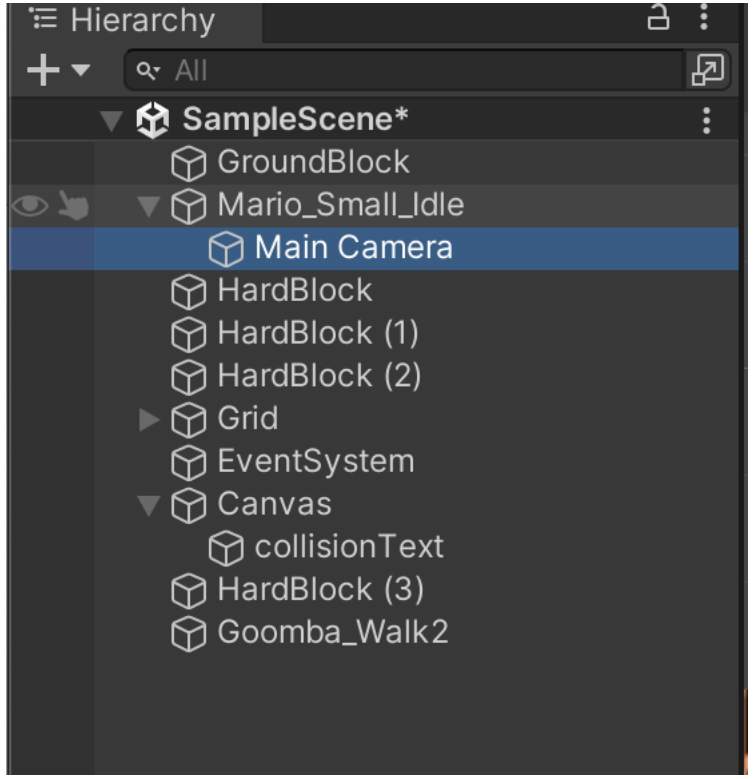


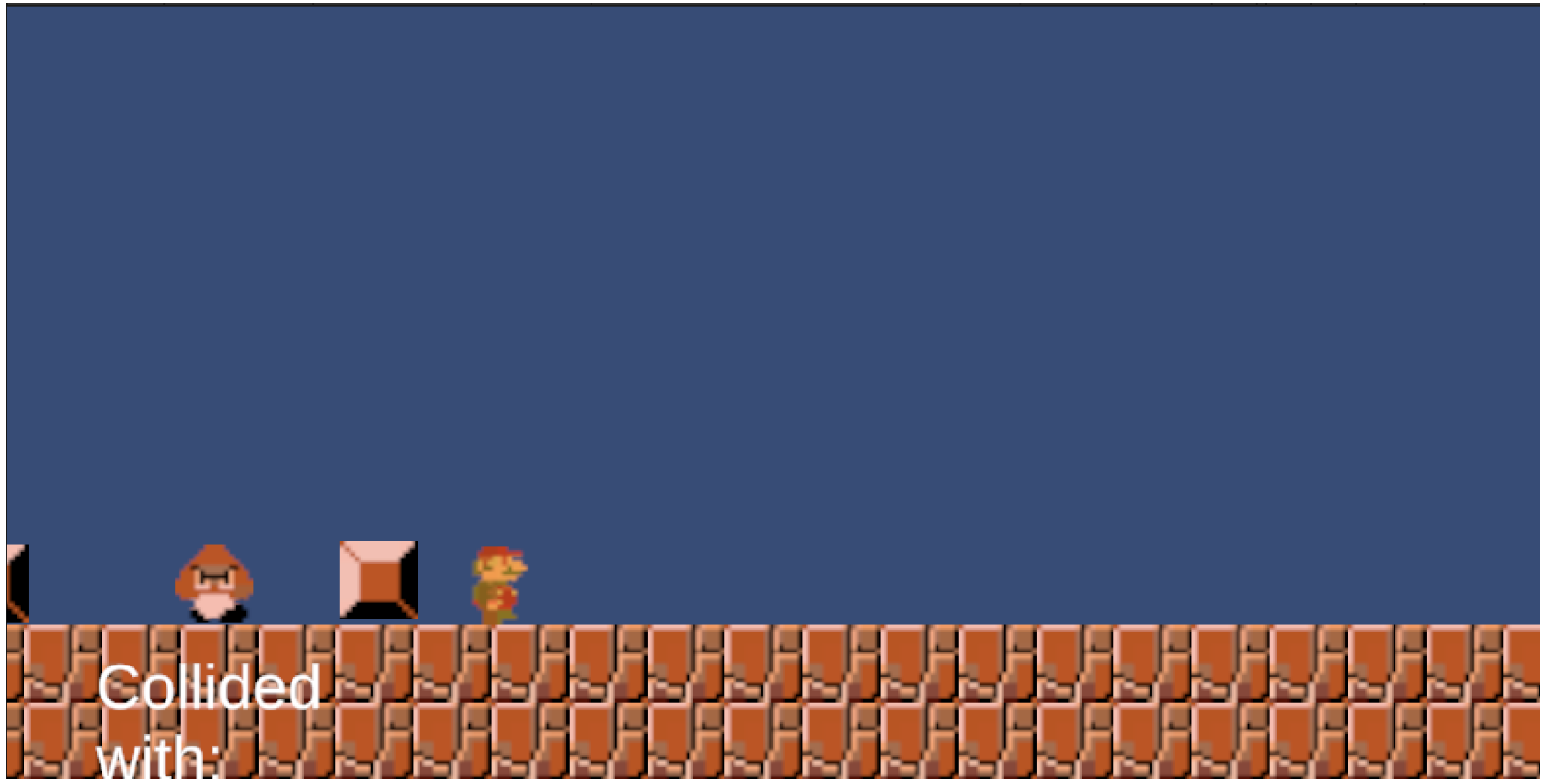


8. Camera – locate the ‘Main Camera’ object



8. Camera – Drag the Main Camera onto Mario





Then the camera should then move with Mario as he moves right

Steps / Requirements

1. Level Design in Unity ✅
2. Mario Movement (left + right) ✅
3. Object detection / collision ✅
4. Jumping over blocks / gravity ✅
5. Enemies (Goombas) alternate directions ✅
6. Jumping on Goombas – further collisions ✅
7. Decoration – animation – walking ✅
8. Camera to track Mario (once tile map created) ✅