2D Game Design in Unity

Lab 3 – Player creation, and basic movement

Prepared by Sarah Amer

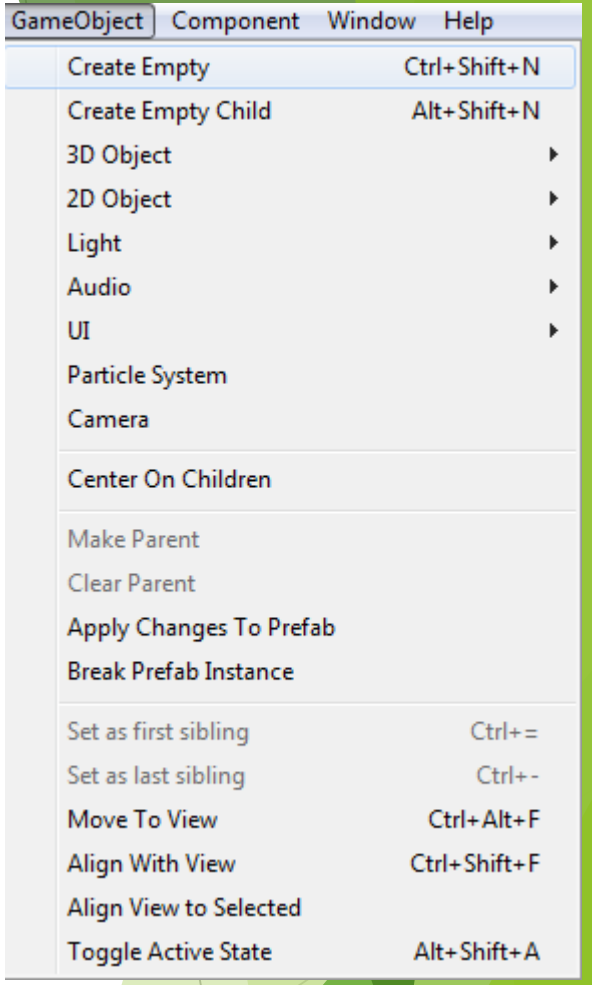
2D Games and Dimensions

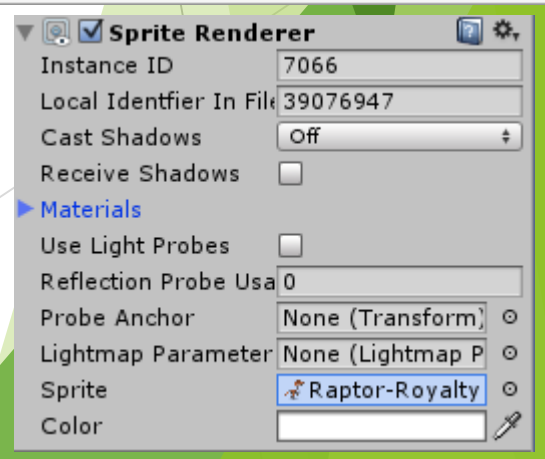
- In 2D games, anything can move in only 2 directions: along the x-axis, along the y-axis, or along both. Values can be positive or negative

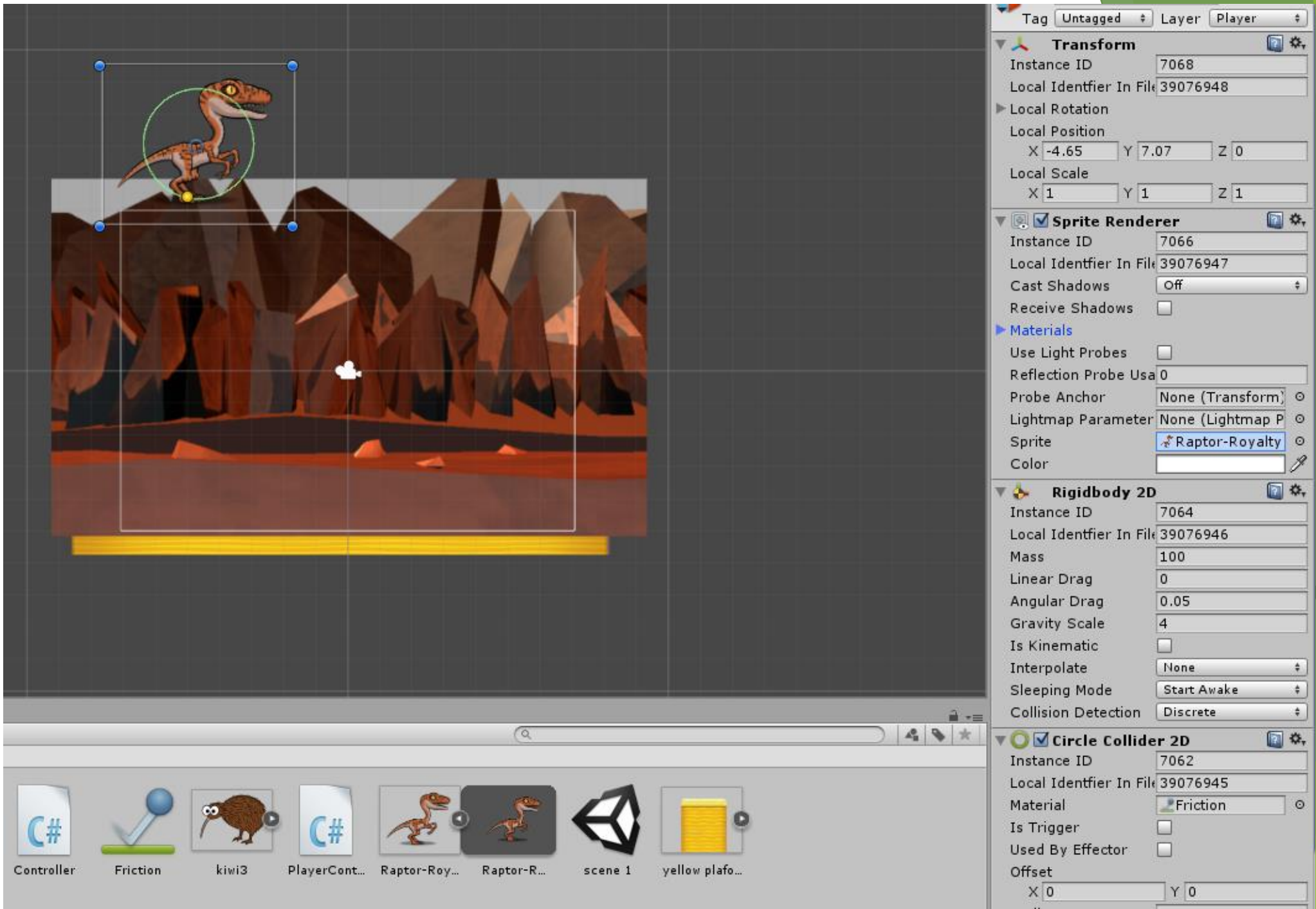
- When a character jumps, the value of y changes
- When a character walks/runs, the value of x changes
- When two or more buttons are pressed, the values of x and y change together

Exercise #9 - Create the Player GameObject

- **To create your player character:**
- Import a sprite of your character to the Project window in Unity (you don't need a complete sprite sheet this lab, as there will be no animation)
- Go to the taskbar above, and choose GameObject. Create an empty GameObject and name it Player GameObject in the Hierarchy window
  - A GameObject acts as a containers for Components, and these Components (e.g. Rigidbody, Light, Sprite, etc.) are what implement the real functionalities
- Select the GameObject, and from the Inspector window click Add Component>Rendering>Sprite Renderer
- Choose the sprite of your character and drag and drop it into the Sprite property in the Sprite Renderer component in the Inspector pane





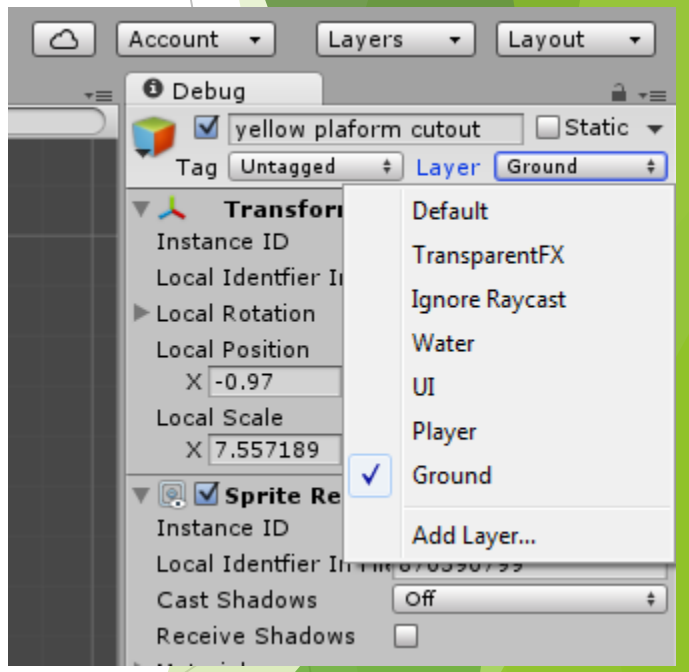

Tag Untagged Layer Player
Transform
Instance ID 7068
Local Identfier In File 39076948
Local Rotation
Local Position
X -4.65 Y 7.07 Z 0
Local Scale
X 1 Y 1 Z 1
Sprite Renderer
Instance ID 7066
Local Identfier In File 39076947
Cast Shadows Off
Receive Shadows
Materials
Use Light Probes
Reflection Probe Usa 0
Probe Anchor None (Transform)
Lightmap Parameter None (Lightmap P
Sprite Raptor-Royalty
Color
Rigidbody 2D
Instance ID 7064
Local Identfier In File 39076946
Mass 100
Linear Drag 0
Angular Drag 0.05
Gravity Scale 4
Is Kinematic
Interpolate None
Sleeping Mode Start Awake
Collision Detection Discrete
Circle Collider 2D
Instance ID 7062
Local Identfier In File 39076945
Material Friction
Is Trigger
Used By Effector
Offset
X 0 Y 0
Controller
Friction
kiwi3
PlayerCont...
Raptor-Roy...
Raptor-R...
scene 1
yellow plafo...

Exercise #9 - Create the Player (Cont.)

- Create a RigidBody2D component (see Lab 1)
  - Set the Interpolate property to Interpolate
    - (Interpolation can be useful to control jerky movement. With *interpolate* mode, motion is smoothed based on the object's positions in previous frames)
  - Freeze rotation around axis Z
- Create a Physics2D boxcollider (see Lab 1)
  - So that the player can interact with other objects, like the ground he stands on

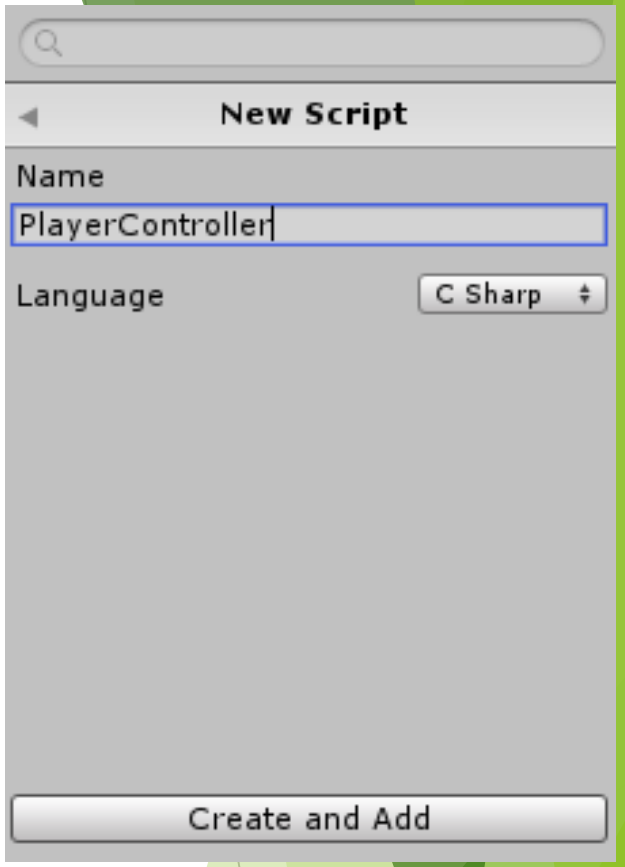
Exercise #9 - Create the Player (Cont.)

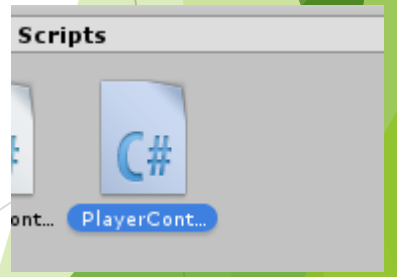
- Adjust the Layers
  - Create a layer for solid ground that you want the character to land on. Name it Ground
  - Create a layer for the character(s) that can interact with the ground. Name it Player
  - After you've created those layers, select the Ground layer for all grounds, and Player for your character

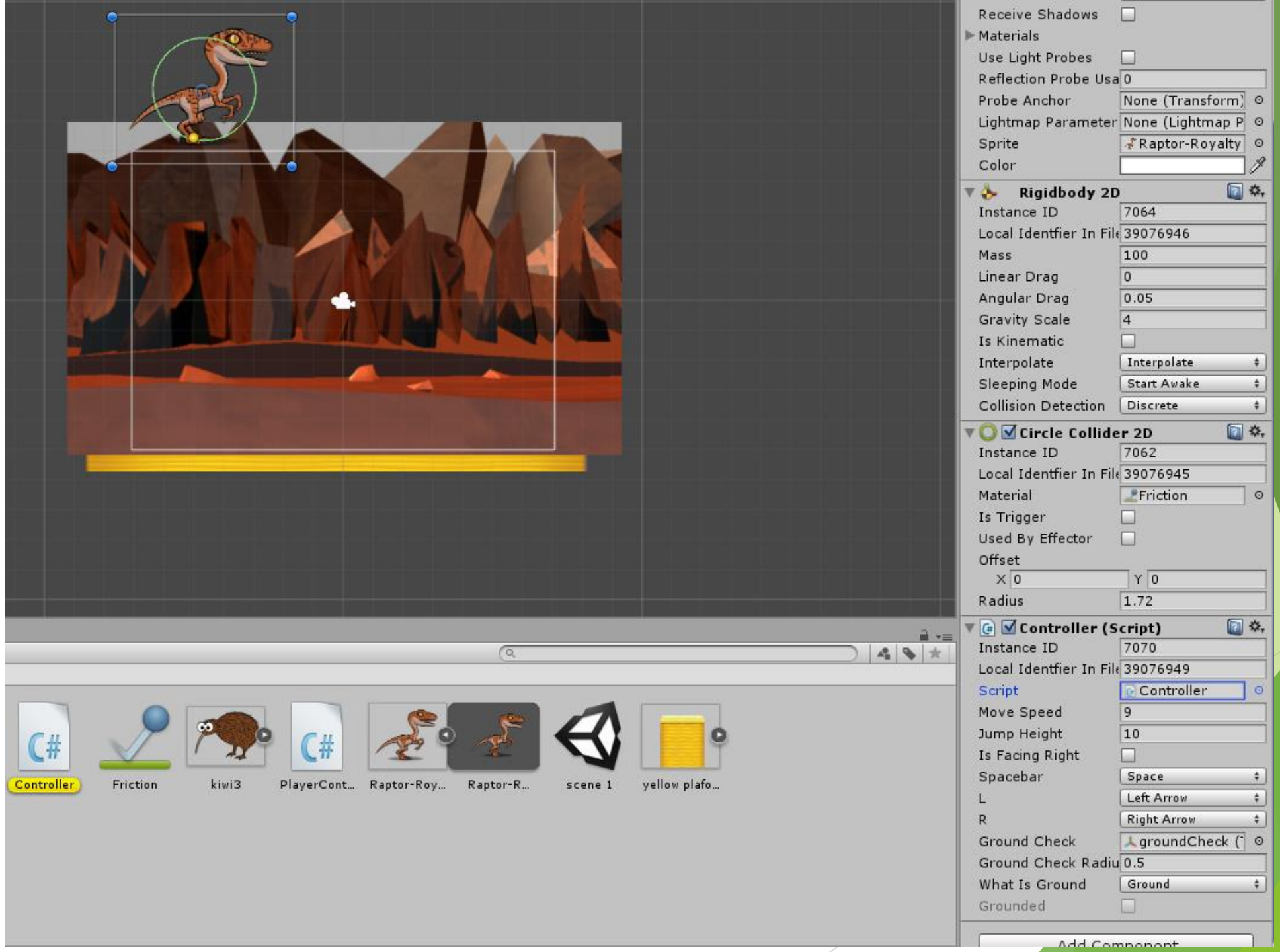


Exercise #10 - Adding a Script to Your Player

- To write code for the player, we must add a script component.
- Go to Add Component>New Script>C Sharp
- Name the script PlayerController then Create
- Drag the new script file to the Script property in Inspector window
- Double-click the script file in Project window
- Visual Studio will open automatically






Receive Shadows
Materials
Use Light Probes
Reflection Probe Usa 0
Probe Anchor None (Transform)
Lightmap Parameter None (Lightmap P
Sprite Raptor-Royalty
Color
Rigidbody 2D
Instance ID 7064
Local Identfier In File 39076946
Mass 100
Linear Drag 0
Angular Drag 0.05
Gravity Scale 4
Is Kinematic
Interpolate Interpolate
Sleeping Mode Start Awake
Collision Detection Discrete
Circle Collider 2D
Instance ID 7062
Local Identfier In File 39076945
Material Friction
Is Trigger
Used By Effector
Offset
X 0 Y 0
Radius 1.72
Controller (Script)
Instance ID 7070
Local Identfier In File 39076949
Script Controller
Move Speed 9
Jump Height 10
Is Facing Right
Spacebar Space
L Left Arrow
R Right Arrow
Ground Check groundCheck
Ground Check Radiu 0.5
What Is Ground Ground
Grounded
Controller
Friction
kiwi3
PlayerCont...
Raptor-Roy...
Raptor-R...
scene 1
yellow plafo...

Exercise #10 - Adding a Script to Your Player (Cont.)

- **In Visual Studio, inside the default MonoBehaviour class, we will need to declare a few variables:**

```
public float moveSpeed; //how fast the character moves
public float jumpHeight; //how high the character jumps

public KeyCode Spacebar; //Spacebar is the name we gave a keyboard button we chose to be the jump button. In this case,
//we chose the Space button, and called it Spacebar. To allocate the Space button to the name Spacebar, go to the Script
//component of your player character, and choose Space from the dropdown list
public KeyCode L;//L is the name we gave a keyboard button we chose to be the left movement button.
public KeyCode R;//R is the name we gave a keyboard button we chose to be the right movement button.
public Transform groundCheck; //an invisible point in space. We use it to see if the player is touching the ground
public float groundCheckRadius; //a value to determine how big the circle around the player's feet is, and therefore determine how
//close he is to the ground
public LayerMask whatIsGround; //this variable stores what is considered a ground to the character
private bool grounded; //check if the character is standing on solid ground;
```

Exercise #10 - Adding a Script to Your Player (Cont.)

- **In the Start() function, create 3 variables and assign the scale (size) of the character's sprite along the x,y and z axes:**

```
// Use this for initialization
void Start () {

    x = transform.localScale.x; //assign the size of the character sprite along x-axis to a variable x
    y = transform.localScale.y; //assign the size of the character sprite along y-axis to a variable y
    z = transform.localScale.z; //assign the size of the character sprite along z-axis to a variable z
}
```

Exercise #10 - Adding a Script to Your Player (Cont.)

- To make your character flip and face in the direction you want it to move, create a flip() function like this:

```
void flip()
{
    if (GetComponent<Rigidbody2D>().velocity.x > 0) //if movement is on positive side of x-axis and not facing to
        //the right
    {
        transform.localScale = new Vector3(x, y, z);//flip the character to the right
    }
    else if (GetComponent<Rigidbody2D>().velocity.x < 0) //if movement is on negative side of x-axis and facing to
        //the right
    {
        transform.localScale = new Vector3(-x, y, z);//flip the character to the left
    }

}
```

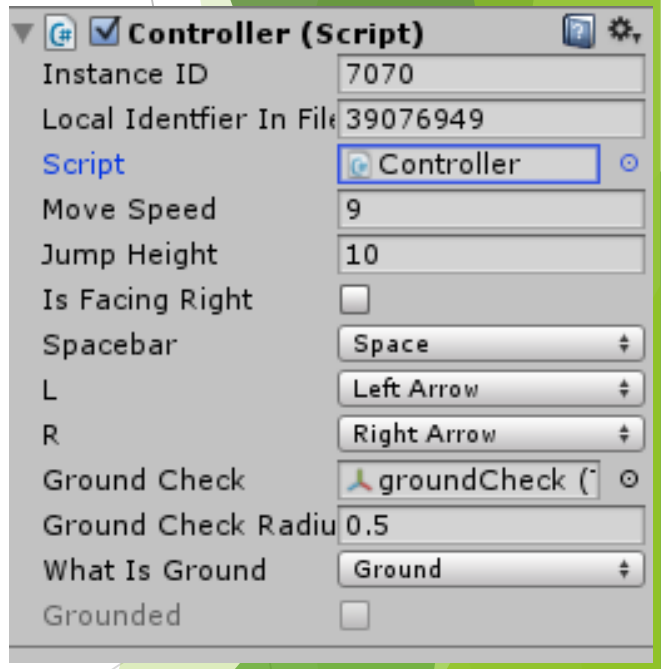
Exercise #10 - Adding a Script to Your Player (Cont.)

- The Update() function is called once every single frame in your game. This is where things like character movement and animation are added, because they are things that happen every frame the game is in play mode.

```
// Update is called once per frame
void Update () {

    if(Input.GetKeyDown(Spacebar) && grounded) //When user presses the space button ONCE
    {
        Jump(); //see function definition below
    }

    if (Input.GetKey(L)) //When user presses the left arrow button
    {
        GetComponent<Rigidbody2D>().velocity = new Vector2(-moveSpeed, GetComponent<Rigidbody2D>().velocity.y);
        //player character
        //moves horizontally to the left along the x-axis without disrupting jump
        flip();
    }

    if (Input.GetKey(R)) //When user presses the left arrow button
    {
        GetComponent<Rigidbody2D>().velocity = new Vector2(moveSpeed, GetComponent<Rigidbody2D>().velocity.y);
        //player character moves
        //horizontally to the right along the x-axis without disrupting jump
        flip();
    }
```

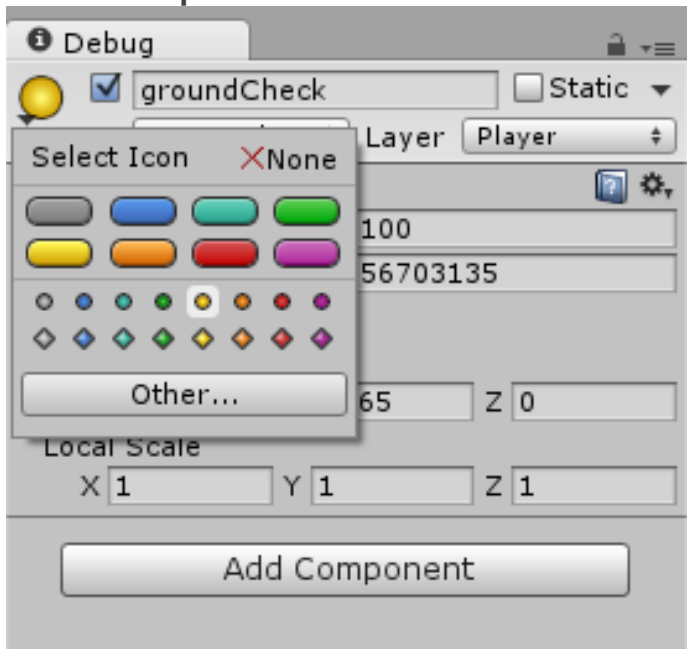
Exercise #10 - Adding a Script to Your Player (Cont.)

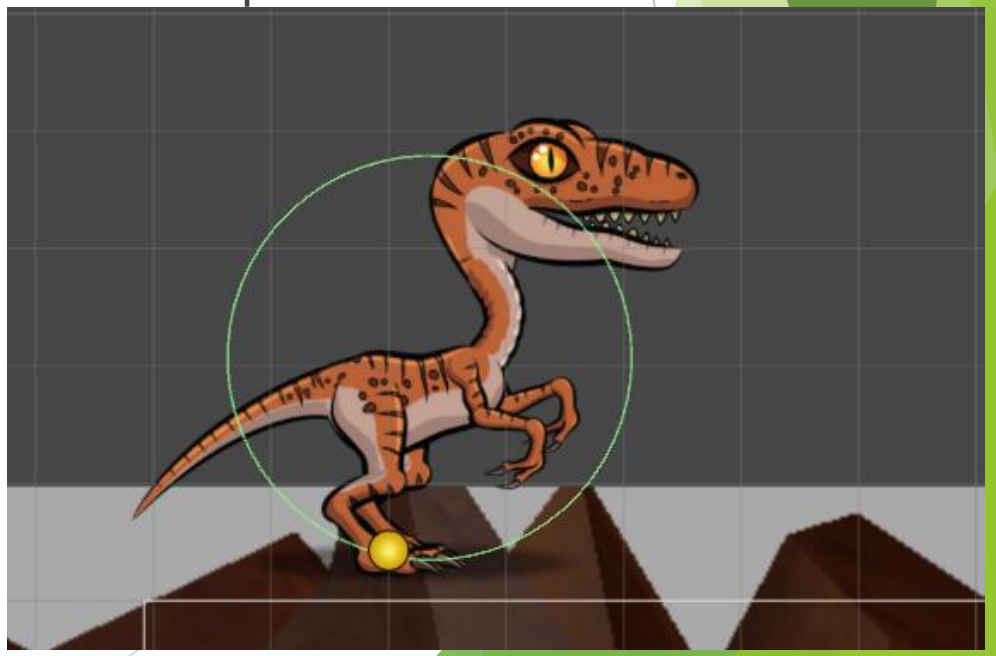
- Save your work and return to Unity. You'll find that the variables have been added as properties in the Script component in the Inspector window
- Give the variables values. You can use the screenshot as example
- For the What Is Ground property, select the Ground layer
- You'll notice a property called groundCheck. How do we create that?
  - groundCheck is very important – we need it to be able to tell the engine when the character is standing on ground, and when they are in the air



Exercise #10 - Adding a Script to Your Player (Cont.)

- Create an Empty GameObject and name it groundCheck. Make it a child of Player GameObject by dragging it and dropping it on Player GameObject in the Hierarchy window
- It is currently invisible on the Scene window. Give it a color by going to its Transform component in the Inspector window. Choose a circle and position it at the character's feet





Exercise #10 - Adding a Script to Your Player (Cont.)

- After you've created the groundCheck GameObject, drag it from the Hierarchy window onto the property textbox in the Inspector window

Exercise #10 - Adding a Script to Your Player (Cont.)

- Go back to the code. We'll begin with a simple Jump() function

```
void Jump()
{
    GetComponent<Rigidbody2D>().velocity = new Vector2(GetComponent<Rigidbody2D>().velocity.x, jumpHeight); //player character jumps
    //vertically along the y-axis without disrupting horizontal walk
}
```

- GetComponent<Rigidbody2D> only works if your character responds to physics, so you MUST attach a Rigidbody2D component to it
- This function makes your character jump by calculating its velocity using its walking speed (x-axis value) and the height to which it can jump (y-axis value)

Exercise #10 - Adding a Script to Your Player (Cont.)

- Create a FixedUpdate() function to calculate when the character is standing on ground

```
void FixedUpdate()
{
    grounded = Physics2D.OverlapCircle(groundCheck.position, groundCheckRadius, whatIsGround); //this statement calculates when
    //exactly the character is considered by Unity's engine to be standing on the ground.
}
```

- The groundCheck GameObject we have created is circular, and has a radius. This statement calculates if the circle overlaps with the ground's boxcollider. If yes, the character is standing on the ground. If not, they are in the air.

Exercise #10 - Adding a Script to Your Player (Cont.)

- Create an Update() function to control character's movement along x-axis

```
void Update () {

    if(Input.GetKeyDown(Spacebar) && grounded) //When user presses the space button ONCE
    {
        Jump(); //see function definition below
    }

    if (Input.GetKey(L)) //When user presses the left arrow button
    {
        GetComponent<Rigidbody2D>().velocity = new Vector2(-moveSpeed, GetComponent<Rigidbody2D>().velocity.y); //player character
        //moves horizontally to the left along the x-axis without disrupting jump
    }

    if (Input.GetKey(R)) //When user presses the left arrow button
    {
        GetComponent<Rigidbody2D>().velocity = new Vector2(moveSpeed, GetComponent<Rigidbody2D>().velocity.y); //player character moves
        //horizontally to the right along the x-axis without disrupting jump
    }
```

Exercise #10 - Adding a Script to Your Player (Cont.)

- How do we make the character flip horizontally when it changes direction?
- Inside the update function, after calculating the character's movement, we create an if-else statement to see whether the value of GetComponent<Rigidbody2D>().velocity.x is positive or negative
- If positive, we use the equation transform.localScale = new Vector(x, y, z) to make the character's image flip to the right of the x-axis without any changes to its shape or size
- If false, then transform.localScale = new Vector(-x, y, z) to make the image of the character flip to the left of the x-axis

Game of the Week: Yoshi's Island (for SNES)

Useful References:

- Johnson, M., Hasankolli, R., & Henley, J. A. (2014). *Learning 2D Game Development with Unity: A Hands-on Guide to Game Creation*. Pearson Education.
- **2D Character Controllers. URL retrieved from:** https://unity3d.com/learn/tutorials/modules/beginner/2d/2d-controllers?playlist=17093
- **How to make a 2D Platformer - Unity Tutorial by Brackeys. URL retrieved from:**https://www.youtube.com/playlist?list=PLPV2KyIb3jR42oVBU6K2DIL6Y22Ry9J1c
- **Unity 2D Platformer Tutorial - Part 1 - Moving and Jumping. URL retrieved from:** https://www.youtube.com/watch?v=86Bgt--Ww7w
- **The Difference Between Update() and FixedUpdate. URL retrieved from:** https://unity3d.com/learn/tutorials/topics/scripting/update-and-fixedupdate