CECS 528 2020 HW2 (3) Unity game project notes – 3 (Using Unity Animation system)

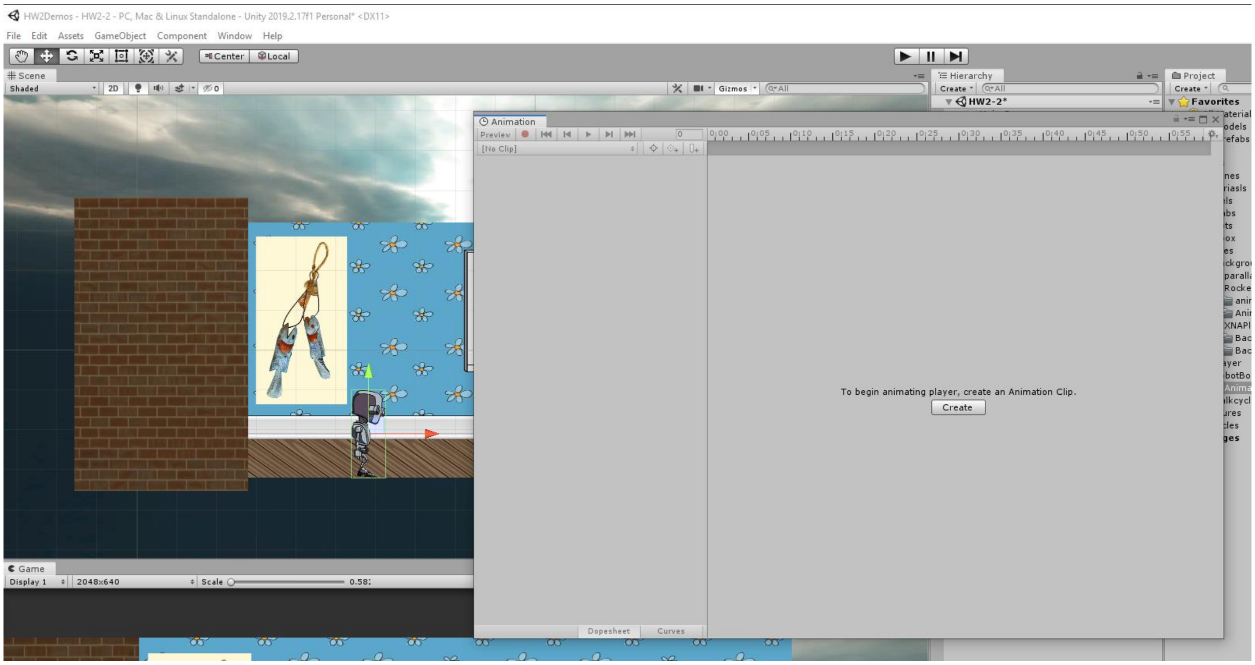
Reference: `4_Unity_2DSpriteAnimation.pptx`

3. Create animation clips from the `RobotBoy` sprites using the Animation view and create an Animator controller to play these animation clips using the Animator view

```
Assets/Sprites/RobotBoy/RobotBoy/RobotBoyCrouchSprite.png
                                 RobotBoyIdleSprite.png
                                 RobotBoyJumpSprite.png
                                 RobotBoyRunSprite.png
                                 RobotBoyWalkSprite.png
```

3.1 Import above .png files (Texture Type Sprite (2D and UI and Sprite Mode Multiple) and slice them into sprites.

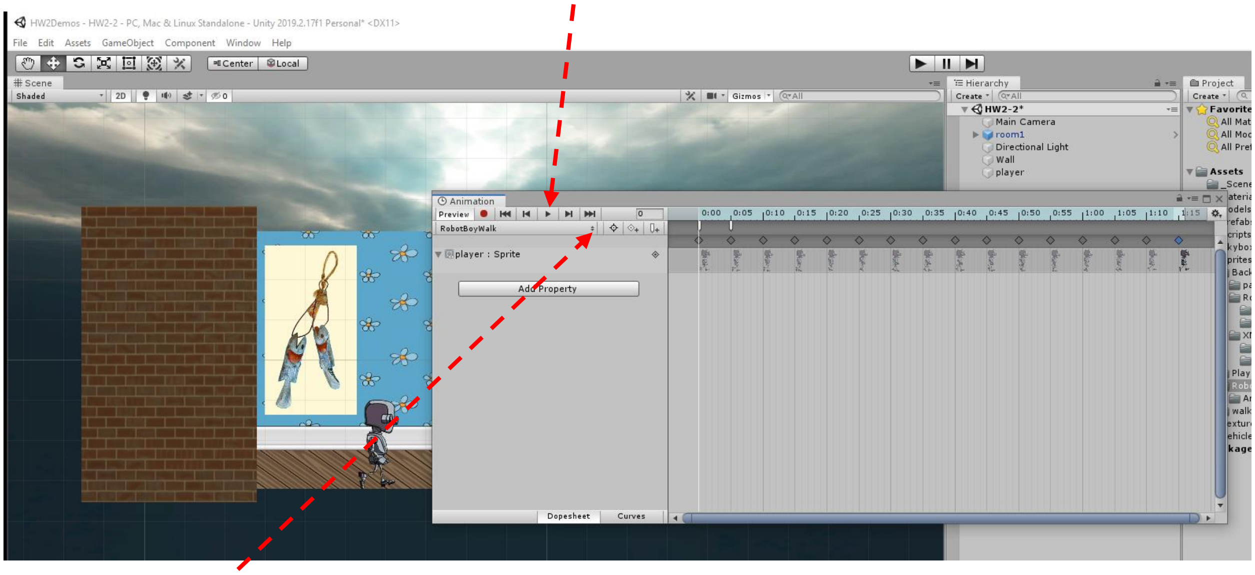
3.2 Select the `Player` (RobotBoy sprite) game object on the scene and open the Animation view (from the menu: Windows/Animation/Animation Ctrl+6):



To begin animating player, create an Animation Clip.

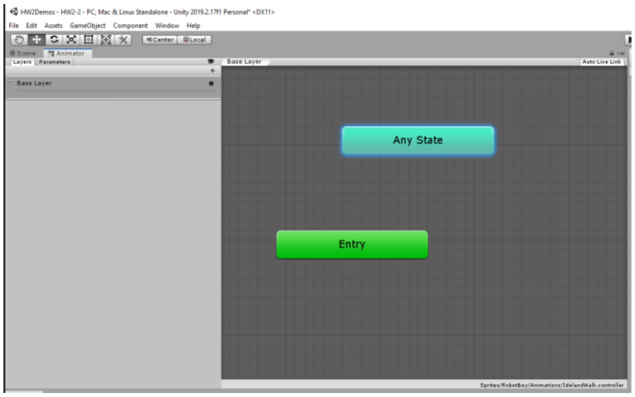
Click the Create button and save the animation clip as `RobotBoyWalk` in the Animations folder. Click `Add Property` button and click the add button (+) of Sprite Renderer → Sprite and add `RobotBoyWalkSprite`'s sprites (0-15) to the Animation view's Timelines.

Click the Play button to play the animation in the editor.

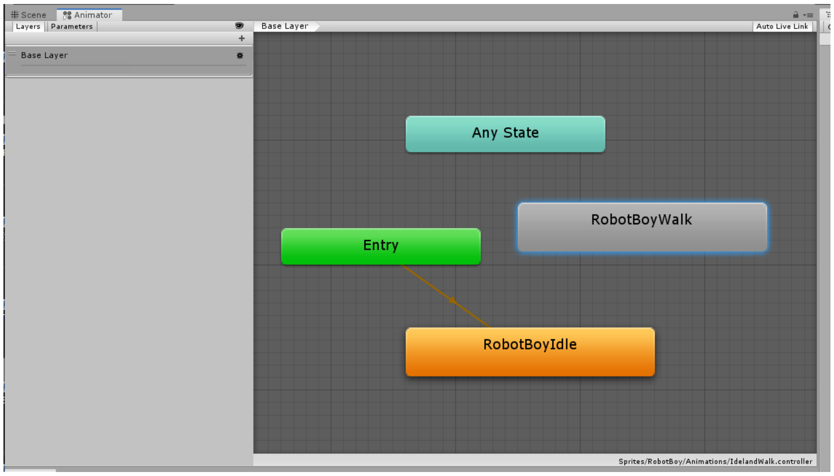


Click the dropdown arrow next to the animation name to add a new animation click and call it `RobotBoyIdle`. Repeat the same process to add `RobotBoyIdleSprite`'s sprites to create the desired animation clip. Similarly, create animation clips, `RobotBoyCrouch`, `RobotBoyJump` and `RobotBoyRun`. Next, we will create an Animator Controller to play these animation clips under scripting control.

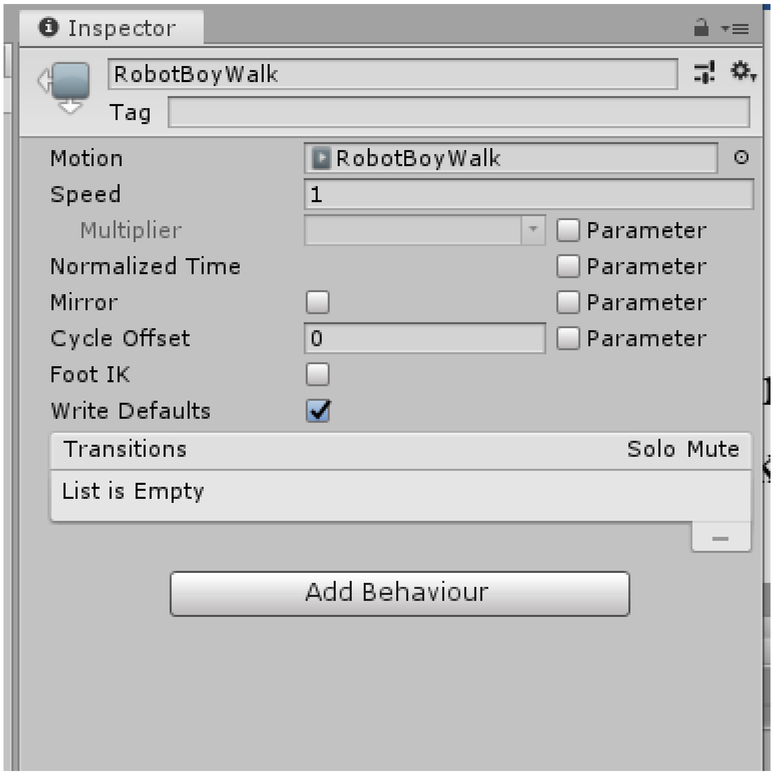
3.3 Create an Animator Controller in the Animation folder called `RobotBoy` and double click it to edit it in the Animator view. Note the Animator view can be opened by clicking the menu Windows→ Animation→Animator to edit the last open Animator Controller. The new Animator Controller looks like this:


HW2Demos - HW2-2 - PC, Mac & Linux Standalone - Unity 2019.2.17f1 Personal* <DX11>
File Edit Assets GameObject Component Window Help
Center
Local
Scene
Animator
Layers
Parameters
Base Layer
Base Layer
Auto Live Link
Any State
Entry
Sprites/RobotBoy/Animations/IdleandWalk.controller

Drag and drop the animation clips, `RobotBoyIdle` and `RobotBoyWalk` in the Animator view. The Animator view looks like this:



Selecting the `RobotBoyWalk` state, the Inspector will show the properties of the state:


Inspector

RobotBoyWalk

Tag

Motion: RobotBoyWalk

Speed: 1

Multiplier □ Parameter

Normalized Time □ Parameter

Mirror □ □ Parameter

Cycle Offset 0 □ Parameter

Foot IK □

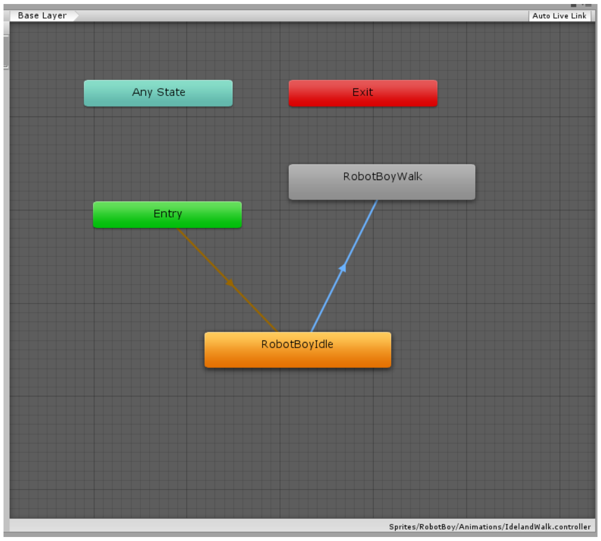
Write Defaults ☑

Transitions Solo Mute

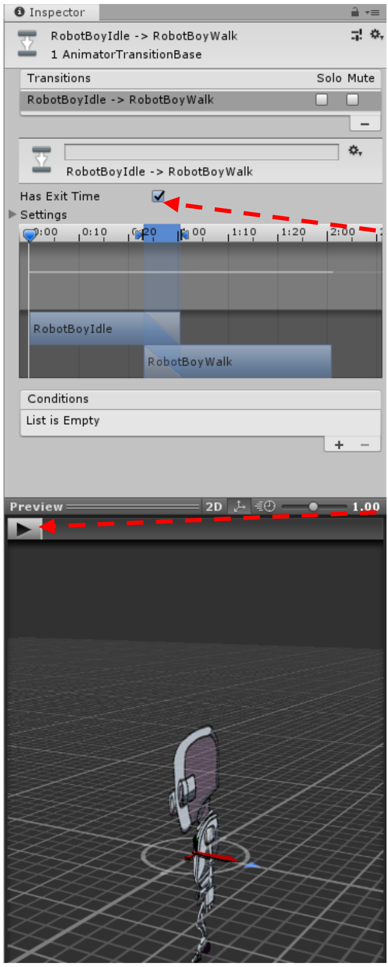
List is Empty

Add Behaviour

The Animator Controller is a finite-state machine, where the start state is the Entry (in green). Right click `RobotBoyIdle` to make a new transition and drag it to the end of the transition to `RobotBoyWalk` like this (shown in blue):



Selecting the new transition, the Inspector will show the properties of the transition likes this:


RobotBoyIdle -> RobotBoyWalk
1 AnimatorTransitionBase

Has Exit Time ☑

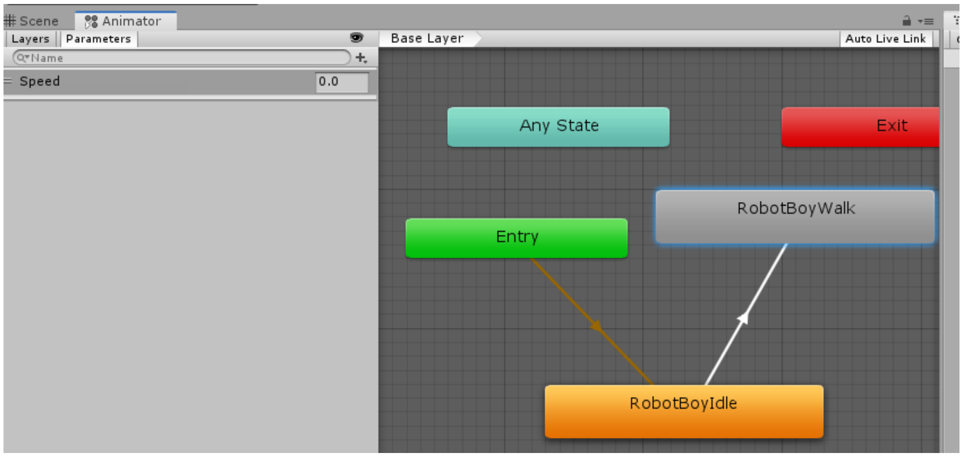
Conditions
List is Empty

Uncheck Has Exit Time to make no wait in transition

Click the Play button to preview the transition

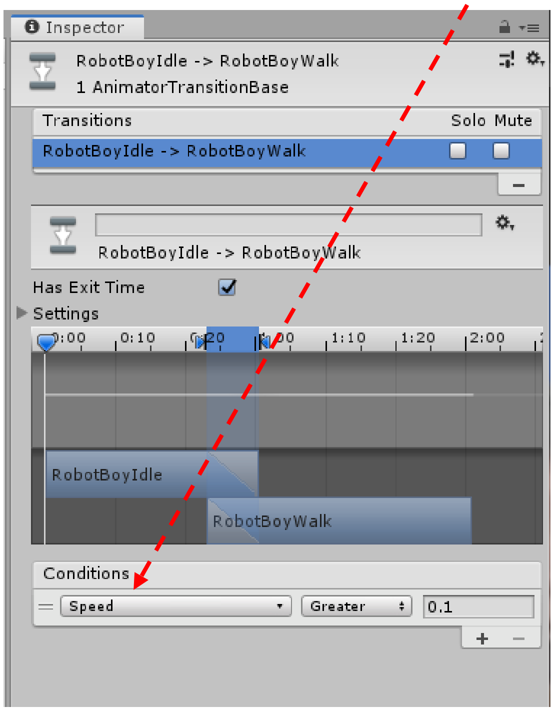
3.4 Create Parameters of the Animator Controller to set transition conditions.

In the Animator view, click the tab Parameters and click + to add a float parameter, `Speed`:



Speed 0.0

Use the Speed parameter to set the transition conditions of the transition, `RobotBoyIdle` → `RobotBoyWalk`, like this:



RobotBoyIdle -> RobotBoyWalk
1 AnimatorTransitionBase

RobotBoyIdle -> RobotBoyWalk □ □

RobotBoyIdle -> RobotBoyWalk

Has Exit Time ☑

Speed | Greater | 0.1

Similarly make a transition from `RobotBoyWalk` to `RobotBoyIdle` and use this condition, Speed less `0.1`. Follow the same process to add other animation clips (e.g. `RobotBoyRun`) and desired transitions to the animator controller.

3.5. Write a C# script to test the Animator controller,

Create a C# script called, `AnimatorTest.cs`, to test the `RobotBoy` animator controller as given below:

```
public class AnimatorTest : MonoBehaviour
{
    public float xSpeedInc = 0.01f;
    public float maxSpeed = 3.5f;

    float speed;
    Animator animator;
    // Start is called before the first frame update
    void Start()
    {
        speed = 0;
        animator = GetComponent<Animator>();
    }

    // Update is called once per frame
    void Update()
    {
    }
    void FixedUpdate()
    {
        // slow down
        if (Input.GetKey(KeyCode.A))
        {
            speed -= xSpeedInc;
            if (speed < 0) speed = 0;
        }
        // Speed up
        if (Input.GetKey(KeyCode.D))
        {
            speed += xSpeedInc;
            if (speed > maxSpeed) speed = maxSpeed;
        }

        transform.Translate(Time.deltaTime * speed, 0, 0);

        animator.SetFloat("Speed", speed);
    }

}
```

Add this script to the `Player` game object and play the game.