unity

Animations and States

🟩 🟩 ⬜ ⬜ ⬜

Animation

- Animation in a 2D Unity game can be done in one of two methods: 1)using a character **sprite sheet**, or using a 2)character **atlas**.
- You are free to choose either for your project, however the lab session will focus on animation using a character **sprite sheet**.
- For this method to be successful, you must first have your character drawn in all the positions and movements it needs.
- Second, save them all in the same image file in PNG format
- Third, import the sprite sheet into Unity, set the Sprite Mode to Multiple, and open the Sprite Editor

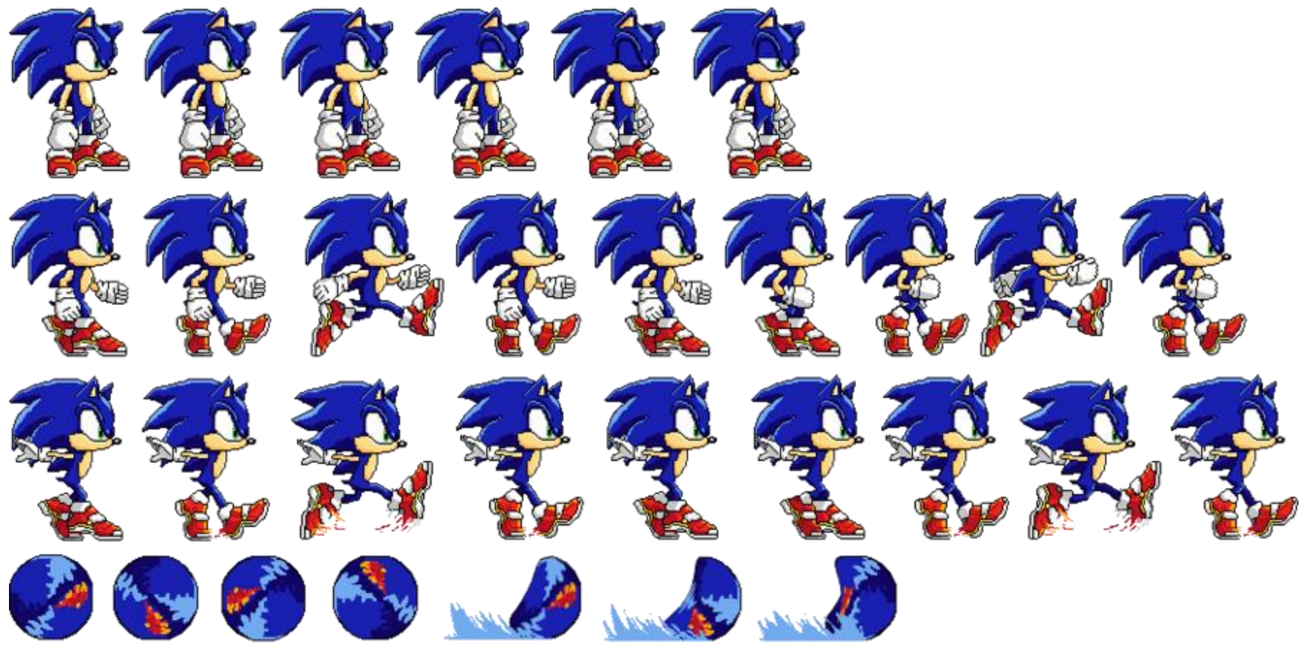
Animation (Cont.)

- In Unity, you can assign many animations to a GameObject (e.g. your character can walk, run, stand idle, and more)
- The Animation Component has several properties including:
  - **Animation:** The default animation to be played
  - **Animations:** An array of animations that can be assigned to the GameObject
  - **Play Automatically:** If checked, this will play the default animation automatically
  - **Animate Physics:** If checked, the animations will interact with physics

- We have to create the animation first BEFORE assigning it to the GameObject

To Animate the Player

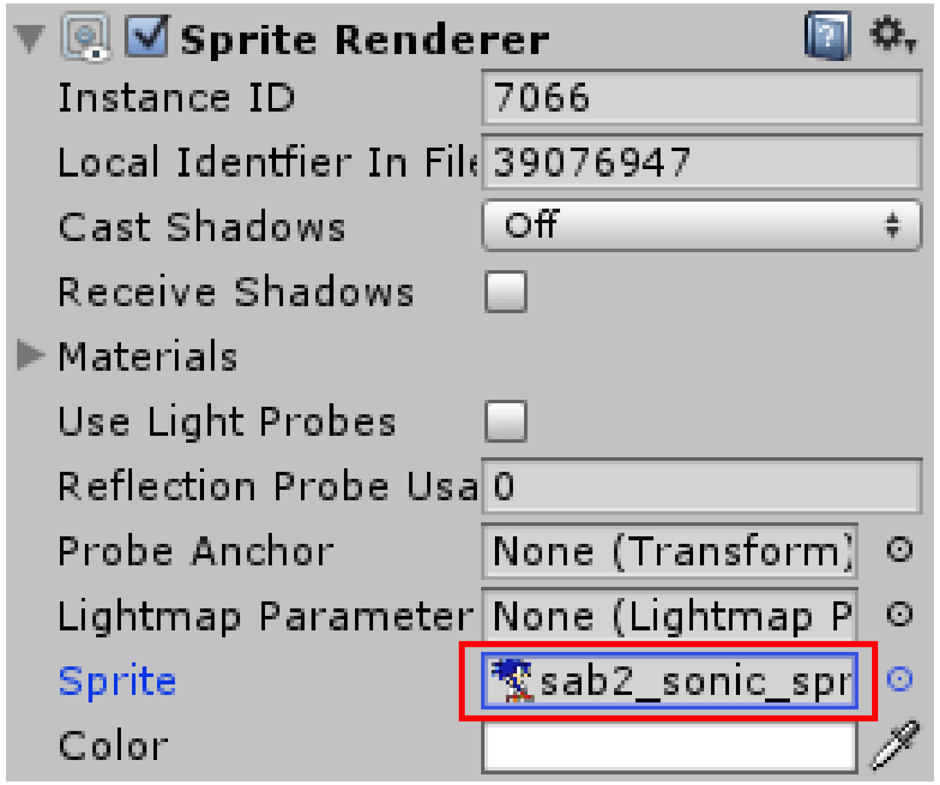
- Last week, our player character was a single sprite that can move along the x-axis, jump along the y-axis and flip in the correct direction it's supposed to move in. But it was a frozen image that did not look alive.
- This week, we will animate the character. Choose (or create) a sprite sheet containing all the character's poses and movements, such as the run cycle below:



Exercise #11 – Create Animations

- For best results, the drawings on your sprite sheet should all be the same size (same width x height)
- Slice your sprite sheet (see Lab 2)
- Go to Window in the Taskbar above, and open the Animation window. Dock it next to the Project window below by dragging and dropping its tab
- Choose the 1st sprite in your sequence, and replace the sprite you were using in last week's lab with the new one. Drag and drop the new sprite onto the Sprite property in the Sprite Renderer component in the Inspector window

Exercise #11 – Create Animations (Cont.)

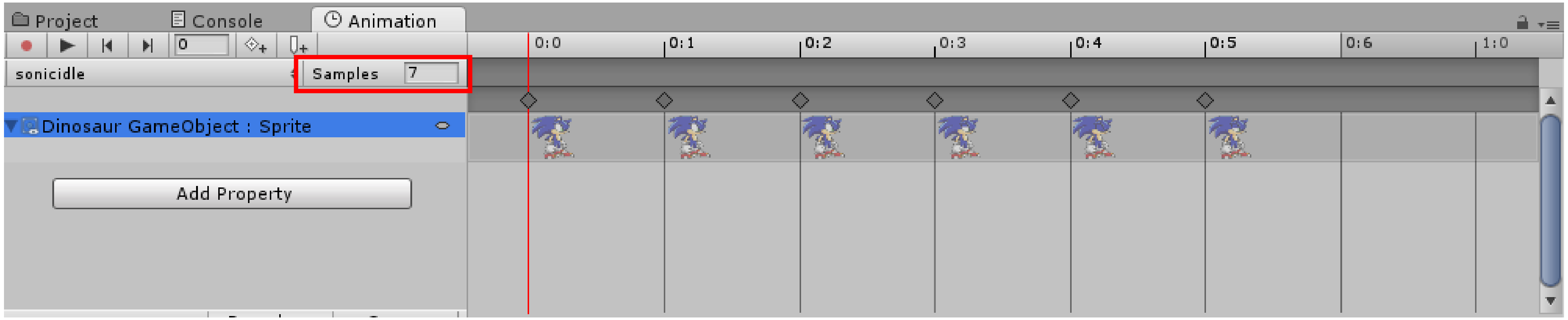


Sprite Renderer

Instance ID	7066
Local Identfier In Fil	39076947
Cast Shadows	Off
Receive Shadows	☐
Materials	
Use Light Probes	☐
Reflection Probe Usa	0
Probe Anchor	None (Transform)
Lightmap Parameter	None (Lightmap P
Sprite	sab2_sonic_spr
Color	

Exercise #11 – Create Animations (Cont.)

- Go to the Animation window, and click Create and name your animation
- You need to create an animation file for every type of movement, so for example if you have art for the character's idle stance, walk and run cycles, you're going to create 3 animations in total.
- To create the idle animation, drag and drop the idle sprites into the Animation window
- Change the value for 'Samples' to whatever value is suitable to avoid having the animation play too fast or too slow. To test your animation's fluidity, click the Play button above the Scene window



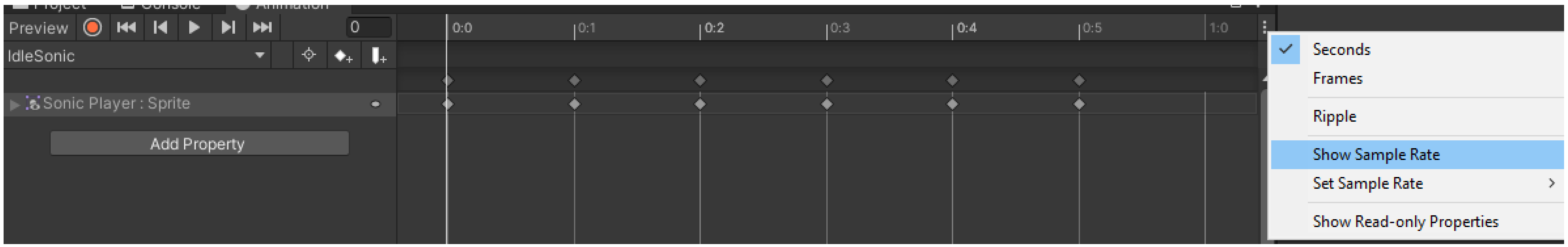
sonicidle

Samples 7

Dinosaur GameObject : Sprite

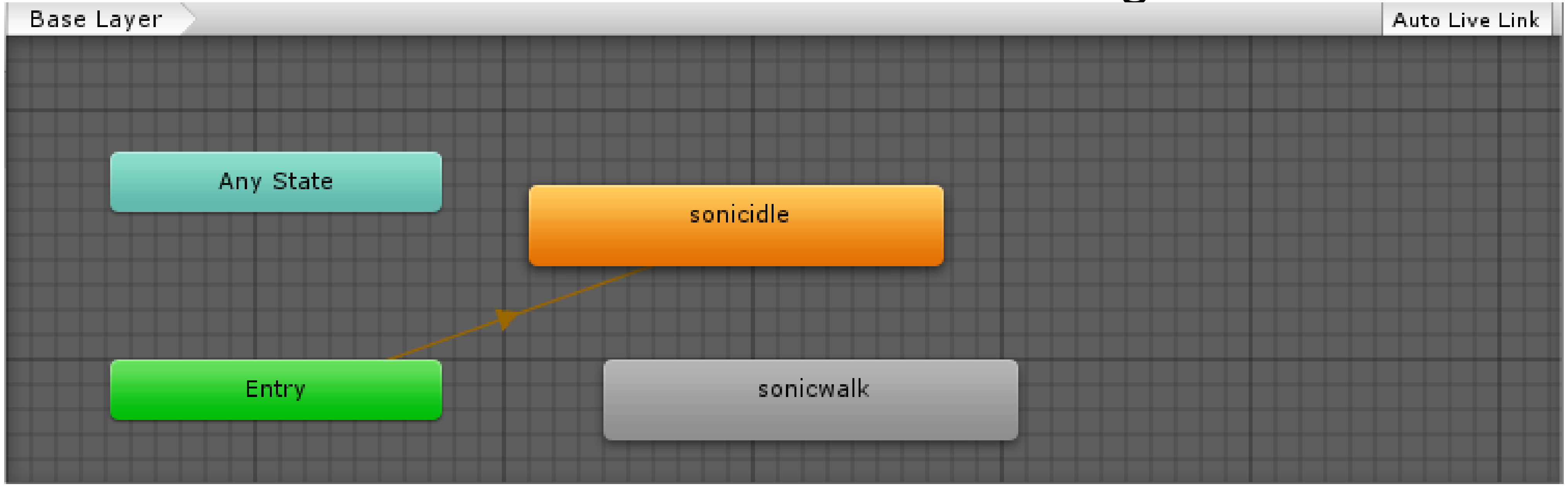
Exercise #11 – Create Animations (Cont.)

- Quick note:
- If you work with a version that doesn't automatically show the 'Samples' property, click on the 3 vertical dots on the right side of the Timeline, and enable 'Show Sample Rate'



Exercise #11 – Create Animations (Cont.)

- When you're satisfied with the animation, save it, then create the animations for walk and run cycles
    - To create a new animation clip, click the bar underneath the Record button
- Once you've finished all animations, go to the taskbar above and choose >Window>Animator. You'll find something like this:

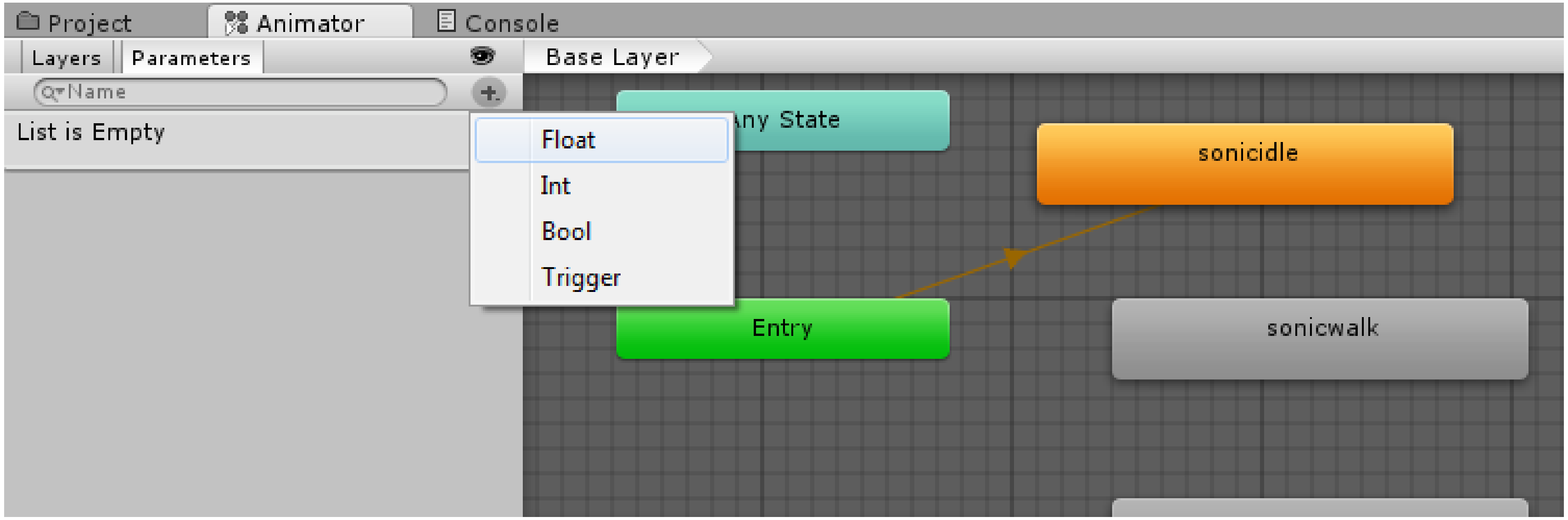


Exercise #11 – Create Animations (Cont.)

- Now you must begin linking your animations with your code
- Go to your player controller script
- Add an Animator type to your list of variables
  - **private Animator anim;** //instance of Animator object to control the character's animation in code
- Go into your Start() function and add
  - **anim = GetComponent<Animator>();**//The player now has an Animator component attached to it, and the animation will play accordingly when the character moves

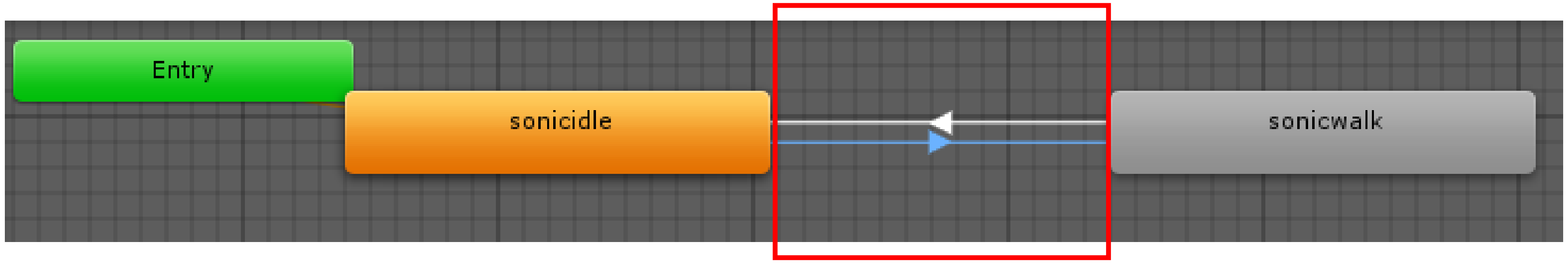
Exercise #12 – Set Idle and Walking States(Cont.)

- Now go back to the Animator tab in Unity
- You'll find a tab called Parameters on the left side. Click the + sign to create a new Float and name it Speed



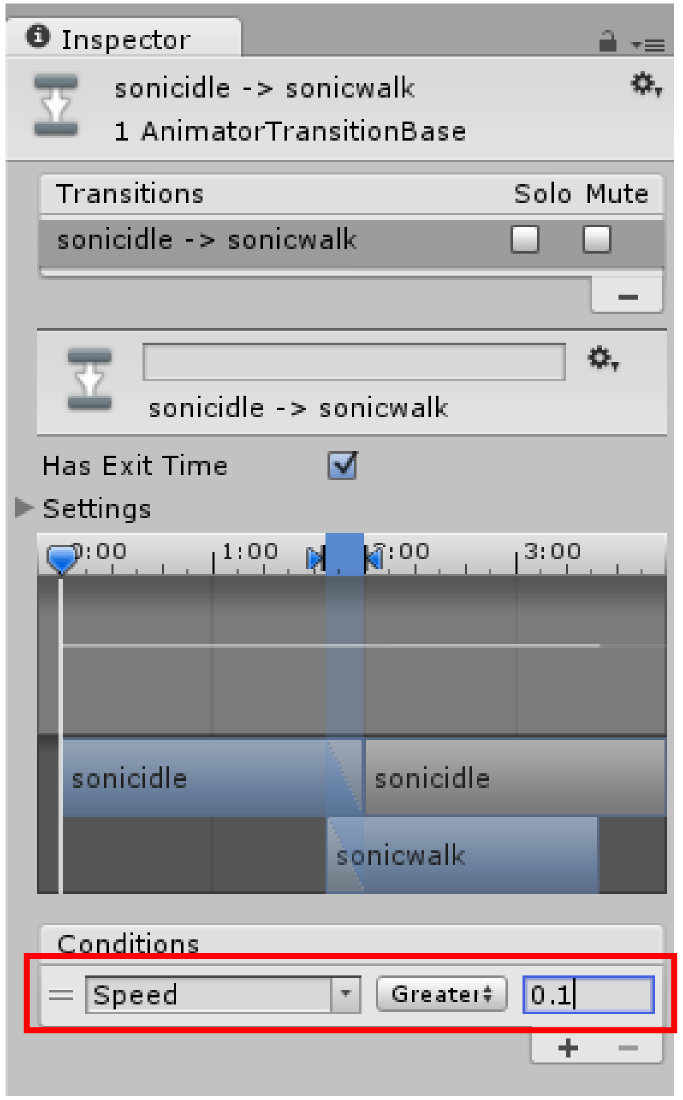
Exercise #12 – Set Idle and Walking States (Cont.)

- The orange state represents the first animation your character will play when the level begins. Normally, an idle animation is the default
- To allow transition between states (e.g. Idle to Walk), right click on the idle state and choose Make Transition. Drag the arrow to the walking state and do the same back to the idle state



Exercise #12 – Set Idle and Walking States (Cont.)

- Click the arrow from the idle to walking animation, and go to the Inspector view, and adjust the Condition property so that the Speed from idle to walking is Greater than 0.1.
- Do the same for the transition from walking to idle, but make it so that the Speed is Less than 0.1


Inspector

sonicidle -> sonicwalk

1 AnimatorTransitionBase

Transitions	Solo	Mute
sonicidle -> sonicwalk	☐	☐

sonicidle -> sonicwalk

Has Exit Time ✓

Settings

Conditions

Speed Greater 0.1

Exercise #12 - Set Idle and Walking States (Cont.)

- Go back to your code, and write a statement that connects the character’s velocity on the x-axis with the Speed condition we have added in the parameters of the Animator

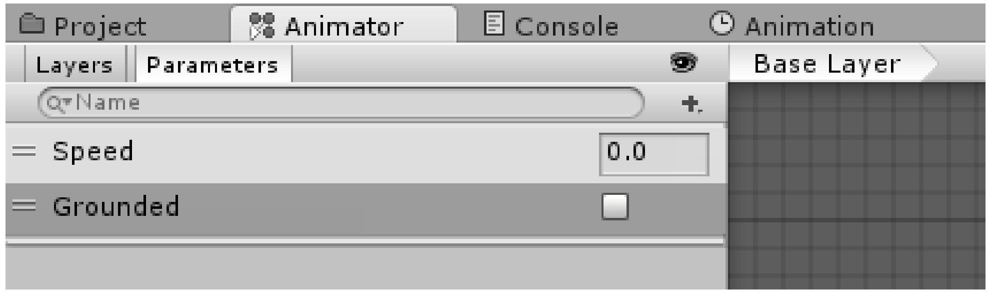
- Underneath your basic left-right movement code, add:
  - anim.SetFloat("Speed",Mathf.Abs(GetComponent<Rigidbody2D>().velocity.x));
  - this function takes the Animator’s parameter name(in this case Speed), and gives it the value of the character’s current velocity on the x-axis. However, the fact that x could be a negative value can cause problems, so Mathf.Abs turns any -ve number into positive without affecting the player character's direction

Exercise #13 – Set Jumping State

- Create a jump animation for your player character
- This animation should only play when your character is NOT on the ground
- Create a new Boolean parameter, and name it

Grounded. So far you have 2 parameters:

Speed and Grounded



Speed 0.0

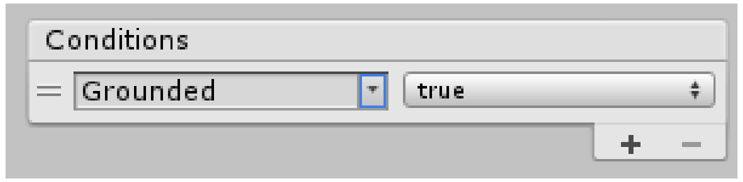
Grounded

- Go back to your script and after your jump statement, add:
  - anim.SetBool("Grounded", grounded);//Animator understands that its

  parameter Grounded and the boolean value grounded are related
- Make Transition arrows from Idle to Jump and back, and from Walk to Jump and back

Exercise #13 – Set Jumping State (Cont.)

- Add a Condition in the Inspector window where Grounded is false
- That means that whether the character is walking or standing idle, the jump will interrupt their animation
- Create another Transition arrow between Jump and Idle, because the character will need to stand for a moment before they begin running again. Set the condition of Grounded to true

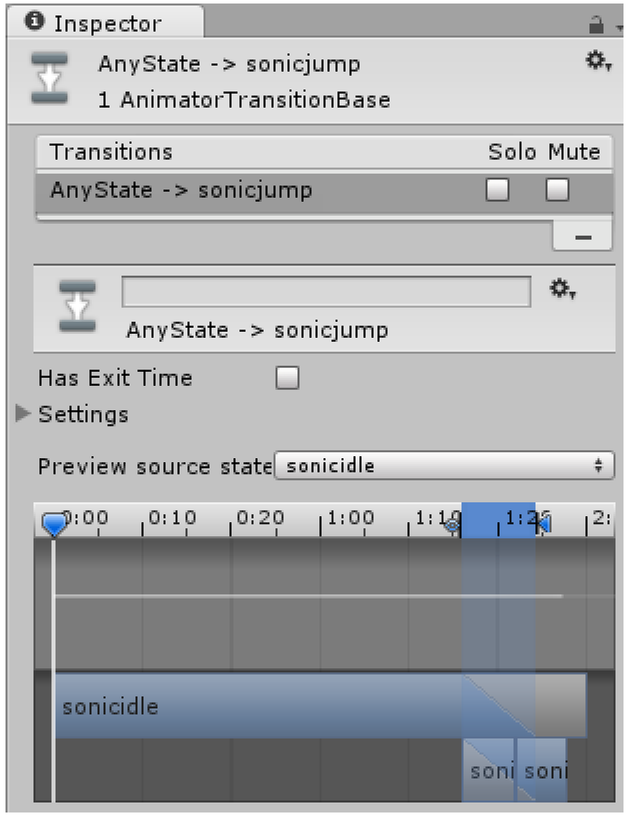


Grounded true

- Now your character should be able to transition from idle stance, to walk, to jump in any order

Exercise #13 – Set Jumping State (Cont.)

- Note:
  - Don’t forget to **uncheck** the ‘has exit time’ condition
  - Open the Settings menu, and uncheck Fixed Duration, and make Transition Duration be zero. This is to remove delays in transitions between one animation and the other
  - If you feel like your animations take too long to transition (for example, the character stays idle for one extra second before they begin walking), try experimenting with the toggles you see in the Inspector view when you click on any Transition arrow
  - Observe the screenshot. There are 2 toggles that you can move around to lengthen or shorten the time it takes to transition from one animation to the other


Inspector

AnyState -> sonicjump

1 AnimatorTransitionBase

Transitions Solo Mute

AnyState -> sonicjump

AnyState -> sonicjump

Has Exit Time

Settings

Preview source state sonicidle

Useful References

Sprite Sheet Animation tutorial from the official Unity3D website:

https://unity3d.com/learn/tutorials/modules/beginner/2d/2d-controllers?playlist=17093

Unity 2D Platformer Tutorial 6 - How To Add Jump And Falling Animations To 2D Character Controller. URL retrieved from:

https://www.youtube.com/watch?v=xHargJbONls&list=PLjAb99vXJuCRD04EUp8p2az1ILZbq_ZfY&index=8

The 12 Principles of Animation. URL retrieved from:
http://blog.digitaltutors.com/understanding-12-principles-animation/

Animating In Unity Using Character Atlas. URL Retrieved from:
https://www.youtube.com/watch?v=NjBcgLqNnwI

Create a Sprite Sheet for your Own 2D Game. URL retrieved from:
https://www.youtube.com/watch?v=cRE2G96591E

A system for planning and timing animation by Disney's Glen Keane. URL retrieved from:
http://www.animationmeat.com/pdf/featureanimation/Glen_Keane_Animation.pdf