unity

3.3 Don’t Just Stand There

Steps:

Step 1: Explore the player’s animations

Step 2: Make the player start off at a run

Step 3: Set up a jump animation

Step 4: Adjust the jump animation

Step 5: Set up a falling animation

Step 6: Keep player from unconscious jumping

Example of project by end of lesson

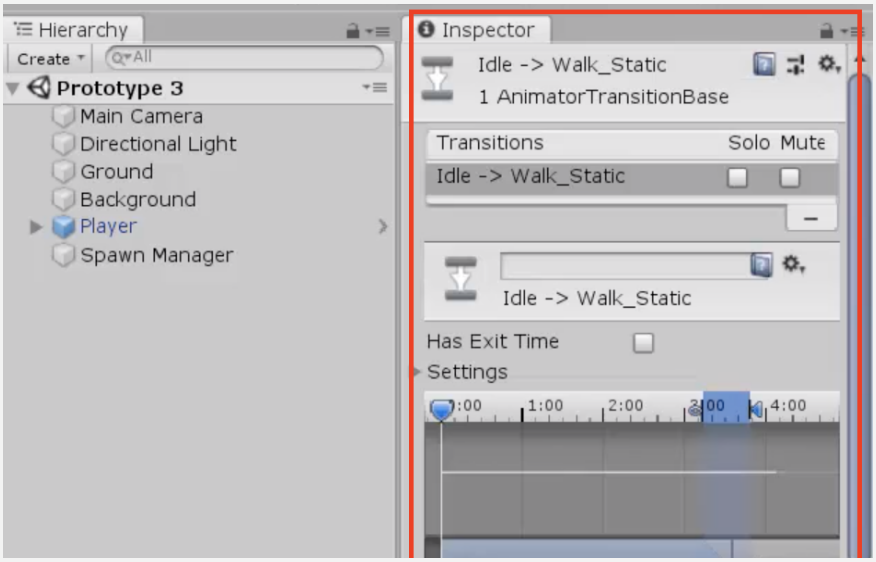
Length:	60 minutes
Overview:	The game is looking great so far, but the player character is a bit… lifeless. Instead of the character simply sliding across the ground, we’re going to give it animations for running, jumping, and even death! We will also tweak the speed of these animations, timing them so they look perfect in the game environment.
Project Outcome:	With the animations from the animator controller, the character will have 3 new animations that occur in 3 different game states. These states include running, jumping, and death, all of which transition smoothly and are timed to suit the game.
Learning Objectives:	By the end of this lesson, you will be able to: - Manage basic animation states in the Animator Controller - Adjust the speed of animations to suit the character or the game - Set a default animation and trigger others with anim.SetTrigger - Set a permanent state for “Game Over” with anim.SetBool

Step 1: Explore the player's animations

In order to get this character moving their arms and legs, we need to explore the Animation Controller.

1. Double-click on the Player's **Animation Controller**, then explore the different **Layers**, double-clicking on **States** to see their animations and **Transitions** to see their conditions

- **New Concept:** Animator Controller
- **New Concept:** States and Conditions

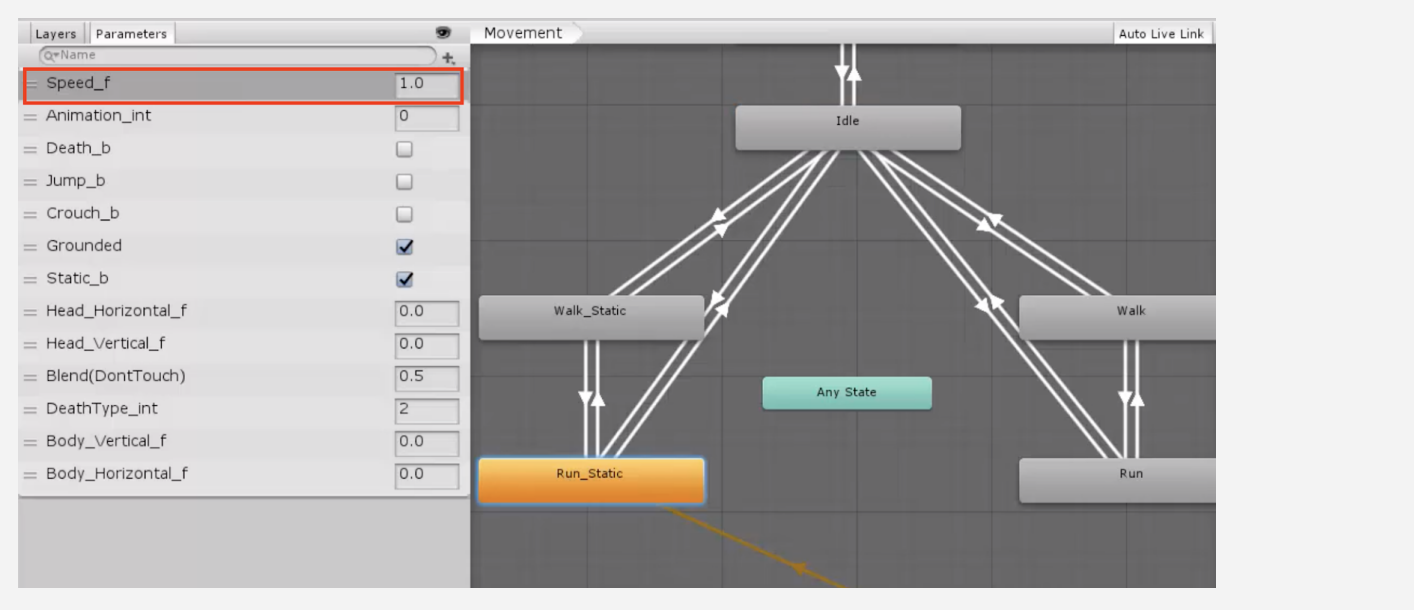


Step 2: Make the player start off at a run

Now that we're more comfortable with the animation controller, we can tweak some variables and settings to make the player look like they're really running.

1. In the **Parameters tab**, change the **Speed_f** variable to 1.0
2. **Right-click** on *Run_Static > Set as Layer Default State*
3. **Single-click** the the Run_Static state and adjust the **Speed** value in the inspector to match the speed of the **background**

- **Tip:** Notice how it transitions from idle to walk to Run - looks awkward - that's why need to make run default



Step 3: Set up a jump animation

The running animation looks good, but very odd when the player leaps over obstacles. Next up, we need to add a jumping animation that puts a real spring in their step.

1. In **PlayerController.cs**, declare a new ***private Animator playerAnim;***
2. In ***Start()***, set ***playerAnim = GetComponent<Animator>();***
3. In the **if-statement** for when the player jumps, trigger the jump: ***animator.SetTrigger("Jump_trig");***

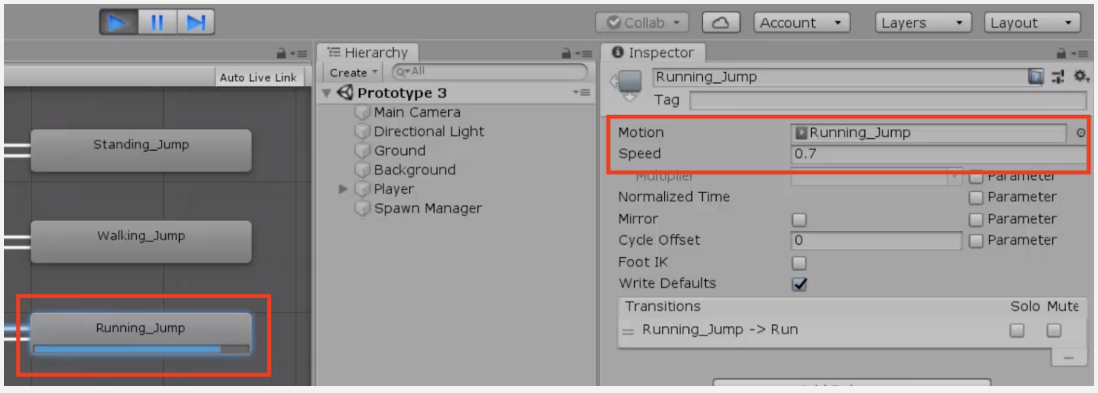
- **New Function:** anim.SetTrigger
- **Tip:** SetTrigger is helpful when you just want something to happen once then return to previous state (like a jump animation)

```
private Animator playerAnim;

void Start() {
  playerRb = GetComponent<Rigidbody>();
  playerAnim = GetComponent<Animator>();
  Physics.gravity *= gravityModifier; }

void Update() {
  if (Input.GetKeyDown(KeyCode.Space) && isOnGround) {
    playerRb.AddForce(Vector3.up * 10 jumpForce, ForceMode.Impulse);
    isOnGround = false;
    playerAnim.SetTrigger("Jump_trig"); } }
```

Step 4: Adjust the jump animation

The running animation plays, but it's not perfect yet, we should tweak some of our character's physics-related variables to get this looking just right.

1. In the Animator window, click on the **Running_Jump** state, then in the inspector and **reduce its Speed** value to slow down the animation
2. Adjust the player's **mass**, jump **force**, and **gravity** modifier to get your jump just right



Step 5: Set up a falling animation

The running and jumping animations look great, but there's one more state that the character should have an animation for. Instead of continuing to sprint when it collides with an object, the character should fall over as if it has been knocked out.

1. In the **condition** that player collides with Obstacle, set the **Death bool** to **true**
2. In the same **if-statement**, set the **DeathType** integer to 1

- **New Function:** anim.SetBool
- **New Function:** anim.SetInt

```
public bool gameOver = false;

private void OnCollisionEnter(Collision collision) {
  if (collision.gameObject.CompareTag("Ground")) {
       isOnGround = true;
  } else if (collision.gameObject.CompareTag("Obstacle")) {
       Debug.Log("Game Over")
       gameOver = true;
       playerAnim.SetBool("Death_b", true);
       playerAnim.SetInteger("DeathType_int", 1);
  }
}
```

Step 6: Keep player from unconscious jumping

Everything is working perfectly, but there's one small disturbing bug to fix: the player can jump from an unconscious position, making it look like the character is being defibrillated.

1. To prevent the player from jumping while unconscious, add ***&& !gameOver*** to the **jump condition**

- **New Concept:** ! "Does not" and != "Does not equal" operators
- **Tip:** gameOver != true is the same as gameOver == false

```
void Update() {
  if (Input.GetKeyDown(KeyCode.Space) && isOnGround && !gameOver) {
    playerRb.AddForce(Vector3.up * jumpForce, ForceMode.Impulse);
    isOnGround = false;
    animator.SetTrigger("Jump_trig");
  }
}
```

Lesson Recap

New Functionality	• The player starts the scene with a fast-paced running animation • When the player jumps, there is a jumping animation • When the player crashes, the player falls over
New Concepts and Skills	• Animation Controllers • Animation States, Layers, and Transitions • Animation parameters • Animation programming • SetTrigger(), SetBool(), SetInt() • Not (!) operator
Next Lesson	• We'll really polish this game up to make it look nice using particles and sound effects!