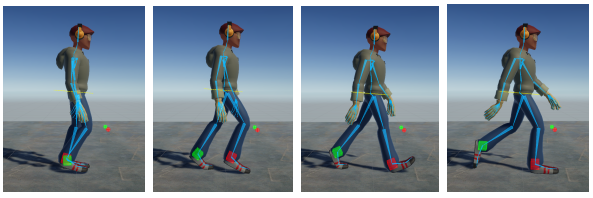
Character assignment 8 - Animating a jump

UW CSE490 summer 2023
Instructor: Dave Hunt

Introduction

In this assignment we will use the control rig and 3rd person character controller we made in the previous assignments. We will animate a jump and then add it to the 3rd person character controller, which we will update to work with rigid body physics. Finally, we will record a video of the character walking and jumping.

The jump animation provides a great opportunity to practice the fundamentals of animation: **anticipation**, **follow-through**, **squash & stretch** and **overlapping action**. The lecture will elaborate more on these concepts and the differences between linear (passive) and interactive animation media. While you are working on these poses, take time to experiment with how to get the best results with your jump by putting these animation fundamentals into action.



Start from the AnimationWalkScene with your character's control rig prefab and character controller that was built in assignment 7

Save it as a new Unity scene named *AnimationJumpScene.unity*

Part 1, Blocking in the main jump poses

It is helpful to think of a jump in these main poses. We will set keys on all rig controls for these main poses spaced 10 frames apart to start with and afterward we will adjust the timing. Please make note that we will be animating the **jump in-place** so it will not translate forward in the clip because we will apply forward velocity in the character controller instead. Similar to the walk it is common to animate a jump with forward motion however, here we will avoid complications by keeping it animated in-place for this assignment.

It really helps to try acting out the poses and motion yourself so you can feel how the weight shifts when your body transitions between poses. If possible use a mirror so you can observe how your body shifts weight. Especially watch how the hips are positioned relative to the feet and how this changes over time. It also helps to record video and sketch thumbnail drawings of the poses so you can make them read well on your model and rig. (These planning steps are recommended, but they are optional and not required for turning in on this assignment.)

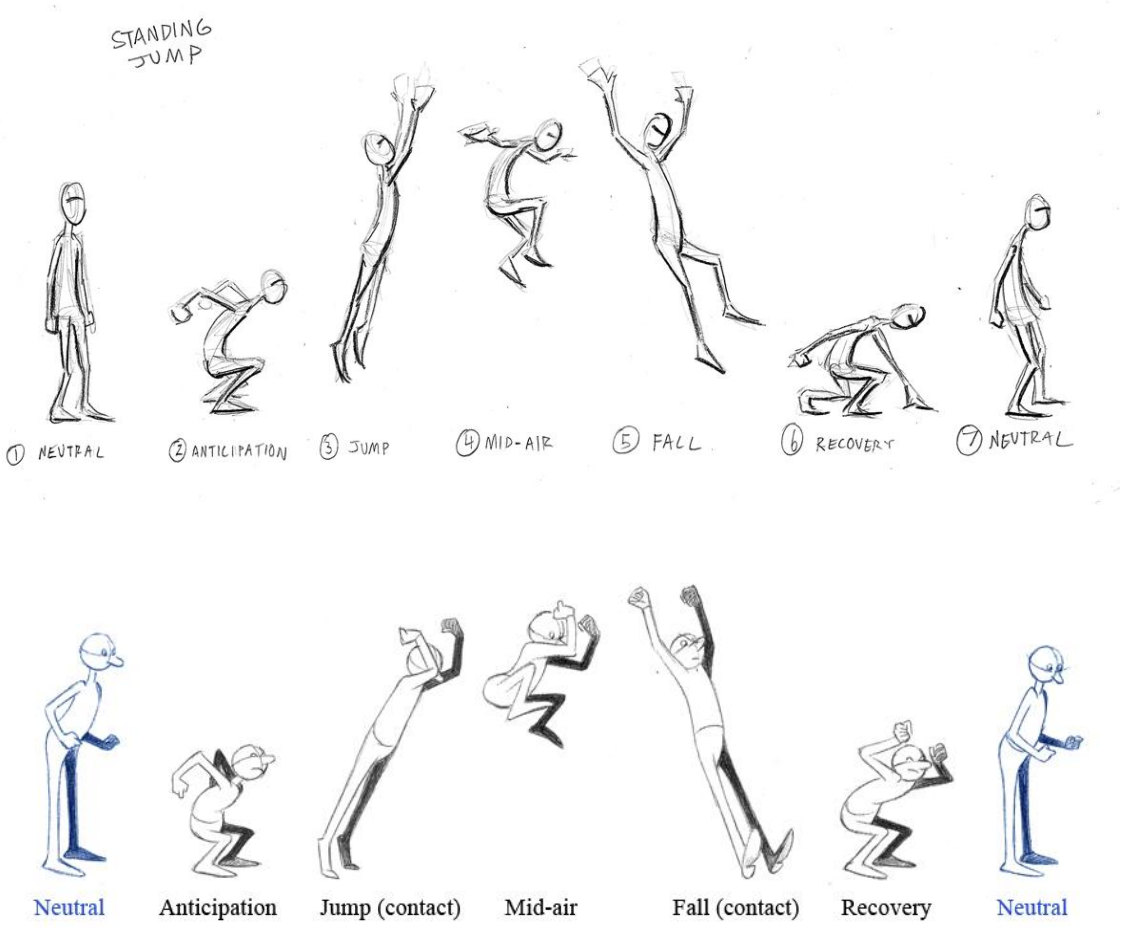
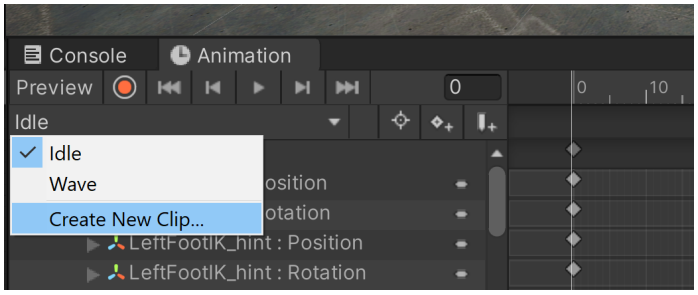


Image source: Animators Survival Kit, Richard Williams

1.1 Create a new animation clip

Start from the *AnimationWalkScene* from the last assignment and save a new copy of the scene called *AnimationJumpScene*.

1. Select your character prefab in the scene Hierarchy
2. In the **Animation Window** click on the clip menu and Create New Clip...

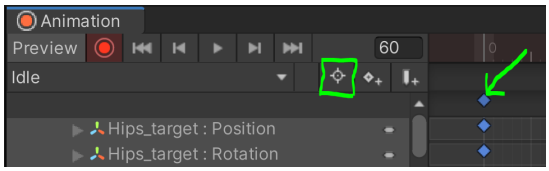


3. Name it *Jump* and save the clip in your character's animations folder `Assets/Assignments/Characters/Sly/Animations`
4. Lock the Animation Window (lock button is in the top-right corner)

1.2 Set the Neutral pose

We will start and end with the character in a neutral pose. This can be the same as the idle pose we defined earlier so the easiest way to get started is by copying the first frame of the idle animation into the new jump clip.

1. Load the *Idle* clip in Animation Window
2. Go to the **Dopesheet** tab and turn off Filter by Selection to display all keys



3. Select all keys and press ctrl-c to copy
4. Load the *Jump* clip in Animation Window
5. Press Record in Animation Window (red circle button)
6. On frame 1 press ctrl-v to paste the keys for the idle pose
7. Important! Add keys to the prefab root position and rotation
    a. Select the top level GameObject of the character prefab
    b. Right-click position and rotation and Add Key

1.3 Set the Anticipation pose

The anticipation pose is very *squashed* as it builds up energy for the jump. It visually telegraphs to the viewer what action is coming next. The line of action is compressed like a C-curve.

1. Move forward in time to frame 10
2. With nothing selected press the "**k**" hotkey in Animation Window to key all properties
3. Set the Anticipation pose
    a. The hips are crouched down low with the arms recoiled back to build up energy for the jump
    b. Spine is curved forward to compress more energy
    c. The head is looking upward in the direction where the character will move next

1.4 Set the Jump pose

The jump pose is very *stretched*. The line of action is relatively straight and angled forward along the *motion arc* or trajectory of the jump. It should represent the straightest pose the body will get to during the upward motion of the jump.

1. Move forward in time to frame 20
2. With nothing selected press the "**k**" hotkey in Animation Window to key all properties
3. Set the Jump pose
    a. The body is straightened upward as it releases energy in the action of jumping
    b. Arms are raised up high
    c. The toes should be straightened to point downward with the very tip of one toe barely contacting the ground
    d. It is ok to raise one of the knees upward depending on the style of your character's jump

1.5 Set the Mid-air pose

The mid-air pose is somewhat *squashed*. It should feel light and weightless as the character experiences a brief moment of hanging in the air as the upward motion gradually turns to falling.

1. Move forward in time to frame 30
2. With nothing selected press the "**k**" hotkey in Animation Window to key all properties
3. Move the whole character up by adding a position key to the top level GameObject of the prefab. This will make the collision capsule move up. Make sure it also has a key at position 0 in the previous pose.
4. Set the Mid-air pose
    a. The body is more relaxed with limbs tucked closer to the core
    b. The elbows and knees are more bent
    c. The head is looking forward to where the character will land

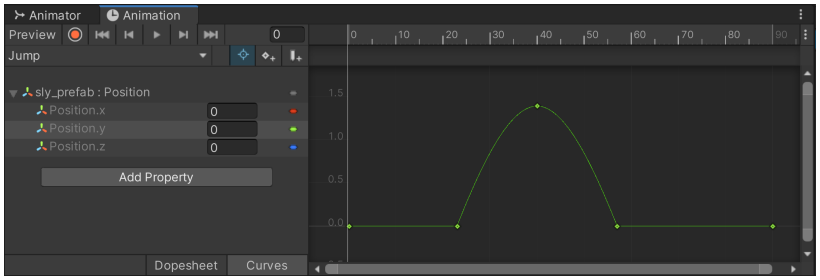
1.6 Set the Fall pose

The fall pose is somewhat *stretched*. The line of action is again straightened out with the opposite angle of the jump pose so it can mostly align with the motion arc of the jump. This pose is the most straightened out the character will get during the fall.

1. Move forward in time to frame 40
2. With nothing selected press the "**k**" hotkey in Animation Window to key all properties
3. Add a key to the prefab root at Position Y = 0
   ***See the image below for reference
4.
5. Set the Fall pose
   a. The body straightened out more by the pull of gravity
   b. The arms can lag behind and stay up longer than the body
   c. One of the feet can be slightly touching the ground (also compare this to the other reference drawings above)

***It is important for the vertical motion of the jump to be done with a single Position Y curve on the prefab root so that we can make it work with player control in part 3 of the assignment.

- You can modify the tangents to look sharp like this by going to the take-off and landing keys and right-click Break Tangents.This makes it possible for the in-tangent and out-tangent to be adjusted independently.



1.7 Set the Recovery pose

The recovery pose is very *squashed* while the body compresses when it hits the ground. The line of action is curved forward and the knees are very bent as the body catches its own weight.

1. Move forward in time to frame 50
2. With nothing selected press the "**k**" hotkey in Animation Window to key all properties
3. Set the Recovery pose
   a. The hips are crouched down low with the knees bent
   b. The follow-through the momentum of the downward motion

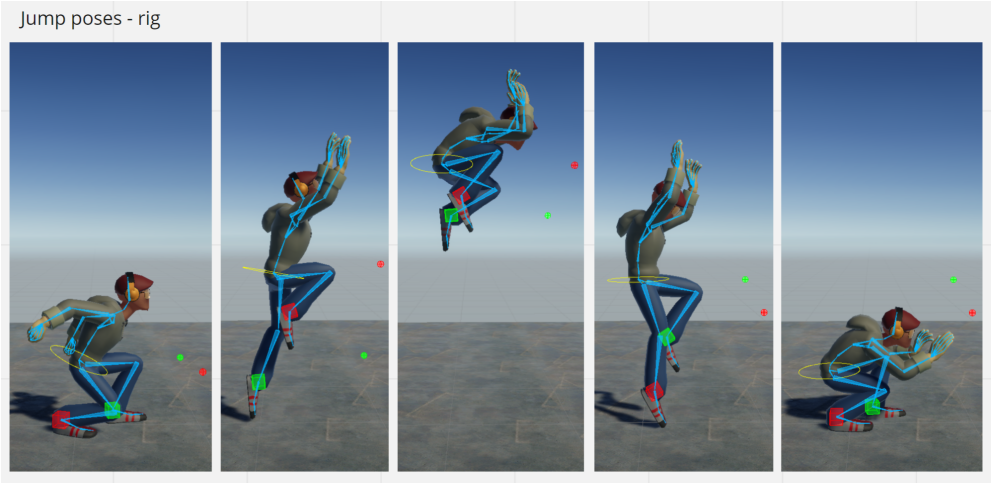
1.8 Set the Neutral pose

This is the same as the first neutral pose.

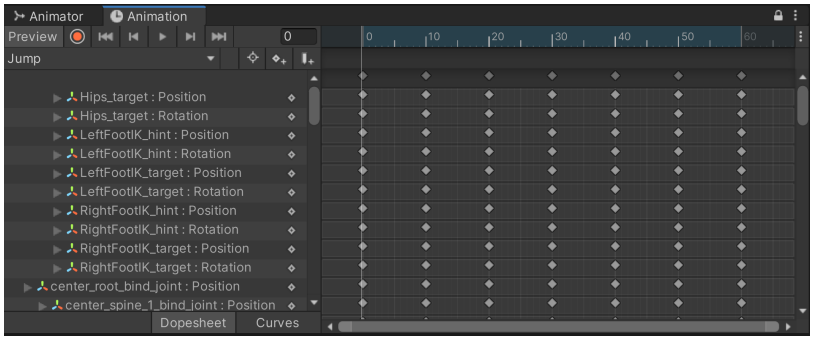
1. Copy and paste the keys on the first frame to frame 60
    a. In the Animation Window go to the **Dopesheet** tab and turn off Filter by Selection to show all keyframes
    b. Click the top key on frame one to select all keys
    c. Press ctrl-c to copy
    d. Move forward in time to frame 60
    e. Press ctrl-v to paste

End of part 1.

Take screenshots of your jump poses and save them to Miro to turn in with your assignment. Show one set of jump screenshots with the rig controls visible and a second with the rig hidden. (disable BoneRenderer and ControlRig to hide the controls) Also take a screenshot of the Animation Window Dopesheet with keys showing on all controls.



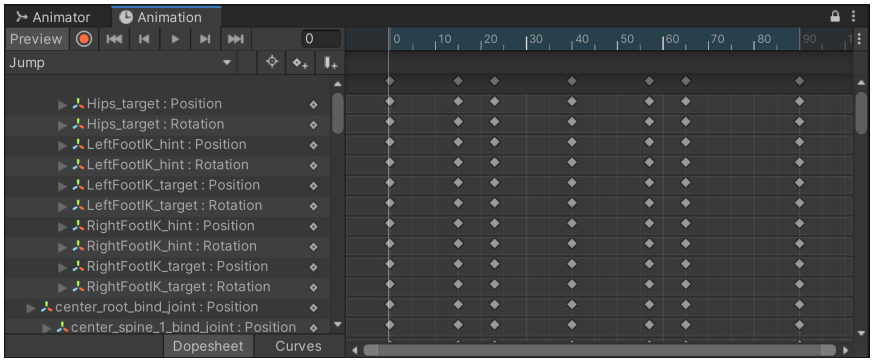
Dopesheet - jump keyframes



Part 2, Refining the timing and interpolation

2.1 Adjust the timing of the jump poses

1. Play the animation in Animation Window preview mode and watch the timing
    a. Where is it popping and going too fast, between which poses?
    b. Where is it looking floaty and going too slow?
    Note: try not to worry about how it is interpolating between poses yet. Animators often evaluate the timing with the main keyframes set to **stepped tangents** with no interpolation so that it's easier to get a sense of the timing.
2. Adjust timing by moving complete rows of keys Dopesheet forward and back in time. For now keep keys aligned for the main poses, all keys at the same frame.



Save a screenshot of the Dopesheet with timing for the main jump keyframes and add it to Miro to turn in with your assignment.

2.2 Add in-betweens

There will need to be more keyframes added in between the main jump poses to help convey a sense of weight. Since these go between the extremes they are called **in-betweens**. For these poses we want to try to create full body poses so that individual parts of the body don't move in isolation.

1. Add an in-between to exaggerate the **follow-through**
    a. Take a look at how the last three keyframes interpolate by scrubbing the time slider. As the fall hits the recovery extreme pose and comes back up into the neutral pose it doesn't feel like there is very much weight at all. It needs more time in a crouch as the rest of the body settles and comes to a rest after the fall.
    b. Add an in-between key after the recovery pose to catch the body's weight
        i. Move the arms, hips and head to react to the settling of the body's weight
        ii. Be sure to key all rig controls with the "**k**" hotkey in Animation Window
2. Add an in-between to exaggerate the **anticipation**
    a. Look closely how the first three keyframes are interpolating, scrub the time slider and watch it play back in Animation Window preview mode. As the body drops

down into the anticipation pose it feels somewhat weak because of how the hips, chest and arms all start and stop moving together at the same time. There is not enough buildup of energy in anticipation of the big action that comes next.

b. Add an in-between key after the neutral pose and before the anticipation extreme pose. Try to make it look like your character is building up as much energy as possible.
    i. The hips should drop down first
    ii. Have the arms rotate down a bit later than the hips
    iii. You can rotate the shoulders, elbows and hands all at different rates
    Note: this is an example of overlapping action

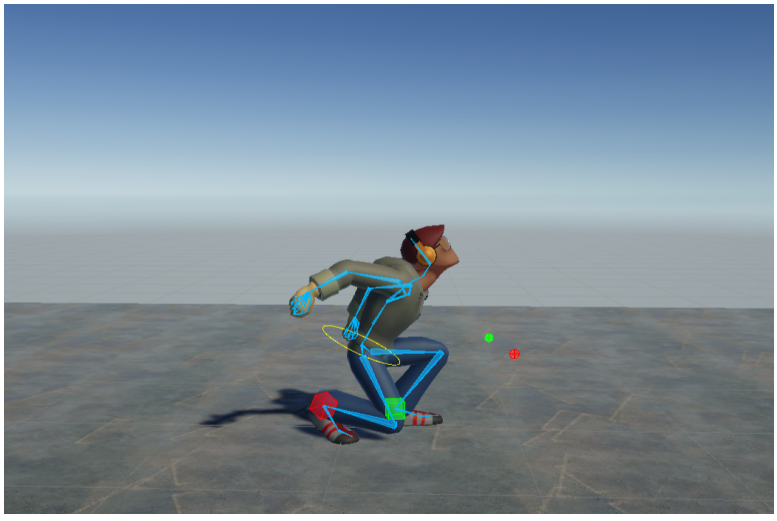
2.3 Splining

Splining is the final stage of polishing the animation where the animator works with individual tangents to smooth out the interpolations between keys.

1. Scrub the time slider and watch it play back in Animation Window
    a. Look for any times when the motion of a particular body part is stuttering or looking floaty
2. Smooth out the key tangents
    a. Go to the Curves tab of Animation Window and adjust individual tangents
    b. Add additional keys to individual channels where needed
3. Work with the hips vertical movement during the mid-air part of the jump. Give it a realistic sense of weight by increasing the time at the top of the arc.
4. Make sure there is no foot sliding or feet going through the ground

End of part 2.

Capture a video or animated gif of your jump and add it to Miro to turn in with your assignment.



Part 3, Adding jump to the character controller

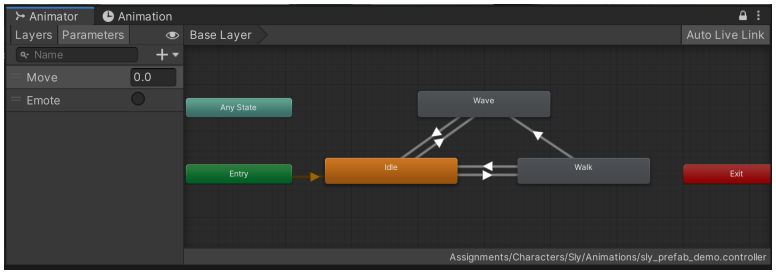
Next we will hook up the Jump animation clip in the Animator Controller state machine and add new player input controls. To do this we will first make copies of the jump animation with modifications so it will look correct and feel responsive to the player.

3.1 Make copies of the jump animation clip

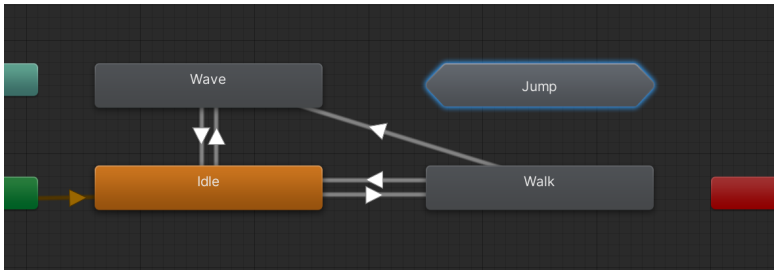
1. In Project view go to your character's Animations folder
2. Select Jump and duplicate it 4 times with ctrl-d
3. Name the new clips:
    a. *JumpFall*, *JumpMidair*, *JumpRecovery* and *JumpUp*

3.2 Create a new sub-state machine for the jump

1. Open the Animator window from Windows → Animation → Animator
2. Select your character prefab in the scene Hierarchy and the Animator window should display the state machine that was built in the previous assignment like this:



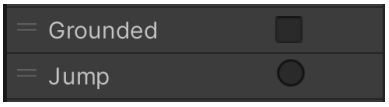
3. Move the Walk state aside to the left and create a new sub-state machine for Jump
    a. Right-click the background and Create Sub-State Machine, name it *Jump*



4. Double-click the Jump sub-state machine to open it
5. Drag and drop the four new jump clips into the window and arrange them like this

Note: the orange arrow represents the default entry transition. In this situation we won't be using that so it doesn't matter if yours looks different than this.

6. Add new **parameters** for the jump. These will be used to pass input and state data from our ThirdPersonRBController script to the **Animator Controller** state machine.



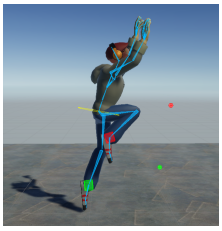
Add new **Parameters**, click the "**+**" button

1. *Grounded*, type = Boolean
2. *Jump*, type = Trigger

3.3 Modify the copies of the jump animation clip

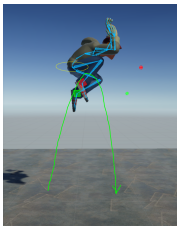
We need to break up the jump into segments so they can be used independently by the state machine. In each clip we will delete unused keys and move the rest to the beginning of the clip. We will also delete the vertical motion on the prefab root Position Y because the player will be instead moved vertically by applying physical impulses to the RigidBody.

JumpUp	A single keyframe of the most extreme jump pose



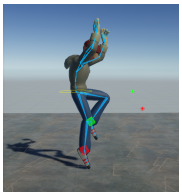
1. Open the *JumpUp* clip in Animation Window
2. In the **Dopesheet** tab turn off Filter by Selection to show all keys
3. Go to frame 1 and remove position properties on the prefab root
    a. Right-click <character>_prefab position, remove properties
4. Delete all keys except for the single row of main Jump keys
5. Move the row of keys to frame 1

JumpMidair	The segment of the jump while the character is in the air



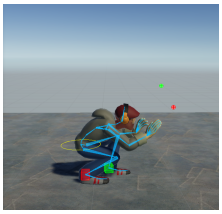
1. Open the *JumpMidair* clip in Animation Window
2. Go to frame 1 and remove position properties on the prefab root
    a. Right-click <character>_prefab position, remove properties
3. Delete all keys except for the section in the middle where the character is in mid air
4. Move the segment of keys to start at frame 1

JumpFall	A single keyframe of the most extreme fall pose



1. Open the *JumpFall* clip in Animation Window
2. Go to frame 1 and remove position properties on the prefab root
    a. Right-click <character>_prefab position, remove properties
3. Delete all keys except for the single row of main Jump keys
4. Move the row of keys to frame 1

JumpRecovery	The segment where the character recovers back to idle pose

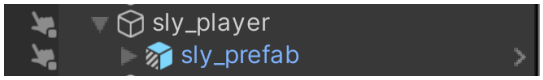


1. Open the *JumpRecovery* clip in Animation Window
2. Go to frame 1 and remove position properties on the prefab root
    a. Right-click <character>_prefab position, remove properties
3. Delete all keys except for the section at the end where the character lands on the ground and recovers back to idle pose
4. Move the segment of keys to start at frame 1

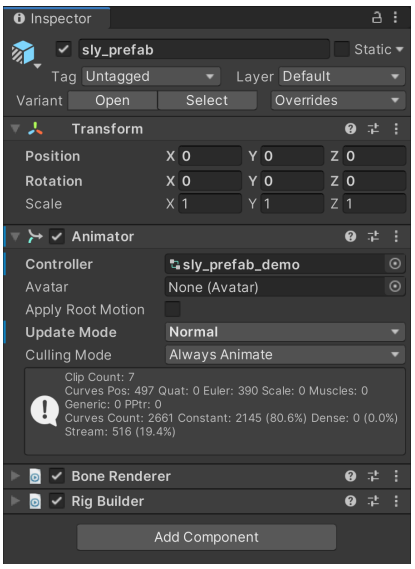
3.4 Set up the third person rigid body controller

This assignment introduces another way of controlling character movement in Unity involving Rigid Body physics simulation. To build this we will modify the Animator Controller state machine from the previous assignment. Here we will move the capsule through the world by applying physical impulses based on player input in the controller script. This will require changes to how our character prefab is set up.

1. In the scene Hierarchy create a new GameObject for the rigid body controller
    a. GameObject → Create Empty
    b. Name it <*your_character_name*>*_player*
    c. Set position to [0,0,0]
    d. Set the **Layer** to Ignore Raycast
2. Parent your character prefab to it
    a. In the scene Hierarchy drag and drop your character prefab below it

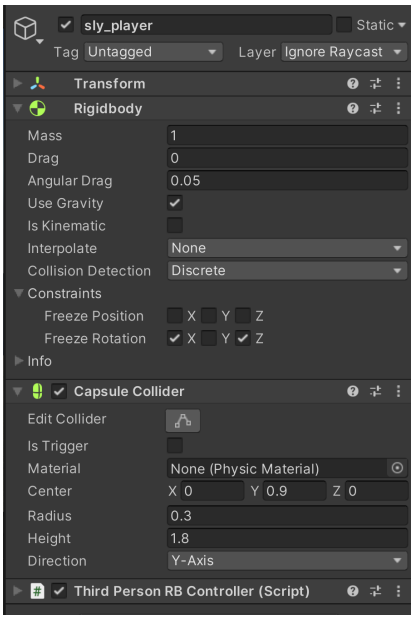


3. Modify the components on your character prefab



1. Copy the **Capsule Collider** component
    a. Right-click Copy Component
2. Remove the Capsule Capsule component
    a. Right-click Remove Component
3. Remove the **RigidBody** component
    a. Right-click to Remove Component
4. Remove the old **ThirdPersonPlayerController**
    a. Right-click to Remove Component
5. Animator component changes
    a. Uncheck Apply Root Motion
    b. Set Update Mode to Normal
6. Check to make sure your prefab root object has the same components and settings as shown in this screenshot
7. Apply all overrides to the prefab
    a. In the Overrides menu click Apply Overrides

4. Set up the new components on the player root game object



1. Select the player root game object
2. Add a RigidBody component
    a. Component → Physics → RigidBody
    b. Freeze Rotation on X and Z as shown
3. Paste the Capsule Collider
    a. Right-click in Inspector on one of the title-bars of another component and Paste Component as New

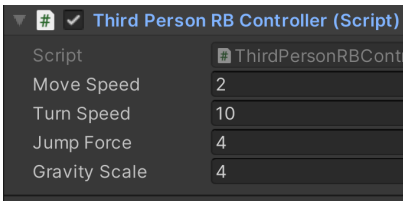
    (Or you can add a new Capsule Collider and fit it to your character model)
4. Create a new C# script for the rigid body controller
    a. Click the Add Component button
    b. Scroll to the bottom of the list and click New Script
    c. Name it *ThirdPersonRBController*
    d. Save it to your character's Scripts folder

5. Double-click *ThirdPersonRBController* in Project view to open it in Visual Studio
    a. Copy and paste the following code into the script and save it with ctrl-s

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

public class ThirdPersonRBController : MonoBehaviour
{
    public float moveSpeed = 3f;
    public float turnSpeed = 10.0f;
    public float jumpForce = 4f;
```

```
    public float gravityScale = 4f;
    private Rigidbody rb;
    private Animator animator;
    private Collider collider;

    void Start()
    {
        rb = gameObject.GetComponent<Rigidbody>();
        animator = gameObject.GetComponentInChildren<Animator>();
        collider = gameObject.GetComponent<Collider>();
    }

    void LateUpdate()
    {
        // player input
        Vector2 moveInput = new Vector2(Input.GetAxis("Horizontal"),
Input.GetAxis("Vertical"));
        if (Input.GetKeyDown(KeyCode.Alpha1)) { animator.SetTrigger("Emote"); }

        // check grounded
        bool grounded = Physics.CheckCapsule(collider.bounds.center, new
Vector3(collider.bounds.center.x, collider.bounds.min.y - 0.1f, collider.bounds.center.z),
0.02f);
        animator.SetBool("Grounded", grounded);

        // forward movement
        animator.SetFloat("Move", moveInput.y);
        rb.MovePosition(transform.position + transform.forward * moveInput.y * moveSpeed *
Time.deltaTime);

        // turning
        Quaternion deltaRotation = Quaternion.Euler(new Vector3(0, moveInput.x * turnSpeed
*30f, 0) * Time.deltaTime);
        rb.MoveRotation(rb.rotation * deltaRotation);

        // jumping
        if (Input.GetButtonDown("Jump") && grounded)
        {
            animator.SetTrigger("Jump");
            rb.AddForce(new Vector3(0,1,0) * jumpForce * 100f);
        }
        rb.AddForce(new Vector3(0, -1, 0) * gravityScale);
    }
}
```

6. [optional] Set up a **Cinemachine Follow Camera** - instructions in assignment 7
7. Tune the values of the ThirdPersonRBController to match your character's walk

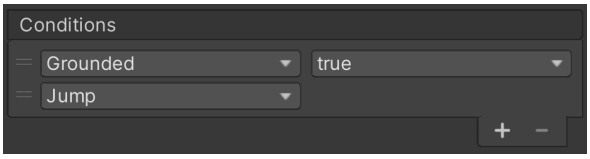


1. Press Play in the Unity editor and walk the character around using the arrow keys or WASD
2. Modify **Move Speed** and **Turn Speed** to look good for your character's walk cycle
3. Right-click the component and Copy Component
4. Exit Play mode, right-click Paste Component Values

3.5 Set up Transitions for Idle to Jump

1. Add a **Transition** from Idle to JumpUp

a. In the Base Layer right click the *Idle* state and Make Transition
b. Drag the arrow onto the *Jump* sub-state machine, **States** → **JumpUp**
c. Select the Transition arrow and set up the following **Conditions**



d. Apply these settings to the Transition
    i. Has Exit Time = false
    ii. Transition Duration = 0.15

2. Add a **Transition** from JumpUp to JumpMidair
    a. Double-click the Jump sub-state machine to show its contents
    b. Right-click the *JumpUp* state and Make Transition
    c. Drag the arrow onto *JumpMidair*
    d. Select the Transition arrow and apply the following settings
        i. Has Exit Time = true
        ii. Transition Duration = 0.2
3. Add a **Transition** from JumpMidair to JumpFall
    a. Right-click the *JumpMidair* state and Make Transition
    b. Drag the arrow onto *JumpFall*
    c. Select the Transition arrow and apply the following settings
        i. Has Exit Time = true
        ii. Exit Time = 0
        iii. Fixed Duration = false
        iv. Transition Duration = 0
4. Add a **Transition** from JumpFall to JumpRecovery
    a. Right-click the *JumpFall* state and Make Transition
    b. Drag the arrow onto *JumpRecovery*
    c. Select the Transition arrow and set up the following condition
        i. Grounded = true
    d. Apply the following settings
        i. Has Exit time = false
        ii. Transition Duration = 0
5. Add a **Transition** from JumpMidair to JumpRecovery
    a. Right-click the JumpMidair state and Make Transition
    b. Drag the arrow onto JumpRecovery
    c. Select the Transition arrow and apply the following settings
        i. Has Exit Time = false
        ii. Fixed Duration = false
        iii. Transition Duration = 0
6. Add a **Transition** from JumpRecovery to Idle
    a. Right-click the *JumpRecovery* state and Make Transition
    b. Drag the arrow onto the orange **(Up) Base Layer** node, **States** → **Idle**

c. Select the Transition arrow and apply the following settings
    i. Has Exit Time = true
    ii. Transition Duration = 0.2

7. Tune the jump values in the ThirdPersonRBController to look good for your character
    a. Press Play in the Unity editor and use the **spacebar** to jump
    b. Select the player root game object in the scene Hierarchy and find the ThirdPersonRBController component in the Inspector
    c. Tune the **Jump Force** and **Gravity Scale** values to look good for your jump
    d. Right-click the ThirdPersonRBController component and Copy Component
    e. Exit Play mode and right-click Paste Component Values

Discussion: How does Anticipation apply to interactive characters?

Notice that we did not use the anticipation part of the jump animation in the player character's state machine. And to be honest, it doesn't look as good without it so why not add it? (This is a subjective choice based on the desired look and feel of the player controller and you totally could add it as a new clip with transitions if you want to!) But just for a minute let's consider how the animation principle of Anticipation works…

In animation the **Anticipation** is there to visually telegraph what is about to happen next to the viewer. When the anticipation does its job well the following action can be really fast and snappy and therefore it feels very impactful to the viewer. This works great for fast actions like a punch or a kick or a jump.

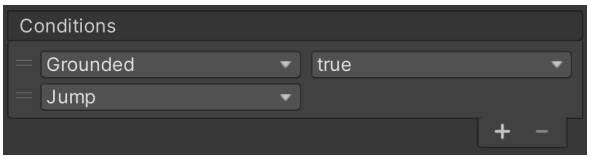
However, when a player is in control of the animated character there is more going on than just watching. The player has already thought that they want the character to jump and pressed the jump button - they are expecting it to happen already so the anticipation is not needed in the same way. When there is a lengthy delay between the button press and the desired action it can feel frustrating for the player and this actually works *against* the intended psychological result of anticipation. Therefore we need to consider the important concept of **Responsiveness** in interactive animation.

Some interactive characters are famous for how great it "feels" to jump, for example, *Mario* by Nintendo. There is an artform to building interactive characters with just the right feeling of control and physicality that is the perfect balance of looking good and feeling responsive. Other playable characters sacrifice some responsiveness to get a more realistic visual quality with longer transitions and anticipation, for example the player characters in *Assassin's Creed*. These character controllers are very advanced and produce a realistic result by sampling from thousands of individual clips that make up all of the required transitions. This of course comes with trade-offs to responsiveness. There are no right or wrong answers here and is ultimately based on the desired art style and personal preference.

3.6 Set up Transitions for Walk to Jump

There are some important differences for transitioning to and from jump while walking. Most importantly, we don't want the player to stop moving forward to play the recovery follow-through clip because that would affect the feeling of responsiveness. The visual trade-off here is pretty noticeable, so in a more advanced player controller we would want to add a special recovery animation for walking. However, in the interest of keeping it simple enough to actually finish on time we will just skip the jump recovery if the player is trying to move.

1. Add a **Transition** from Walk to JumpUp
    a. In the Base Layer right click the *Walk* state and Make Transition
    b. Drag the arrow onto the *Jump* sub-state machine, **States → JumpUp**
    c. Select the Transition arrow and set up the following **Conditions**



    d. Apply these settings to the Transition
        i. Has Exit Time = false
        ii. Transition Duration = 0.15
2. Add a new **Transition** from JumpRecovery to Walk
    a. Right-click the JumpRecovery state and Make Transition
    b. Drag the arrow onto the orange **(Up) Base Layer** node, **States → Walk**
    c. Select the Transition arrow and apply the following settings
        i. Has Exit Time = false
        ii. Fixed Duration = false
        iii. Transition Duration = 0.15
    d. Add a condition: **Move** is **Greater** than **0.1**
3. Test the walk/jump transition in Play mode

3.7 Add a Transition for Walk to Fall

Our third person player controller is almost finished. But what happens if the player walks off a ledge? We need to set up another transition for this.

1. Add a Transition from Walk to JumpFall
    a. Right-Click the *Walk* state and Make Transition
    b. Drag the arrow onto the *Jump* sub-state machine, **States → JumpFall**
    c. Select the Transition arrow and apply the following settings
        i. Has Exit Time = false
        ii. Fixed Duration = true
        iii. Transition Duration = 0.2
    d. Add a condition: **Grounded** = **false**
2. Test the walk/fall transition in Play mode

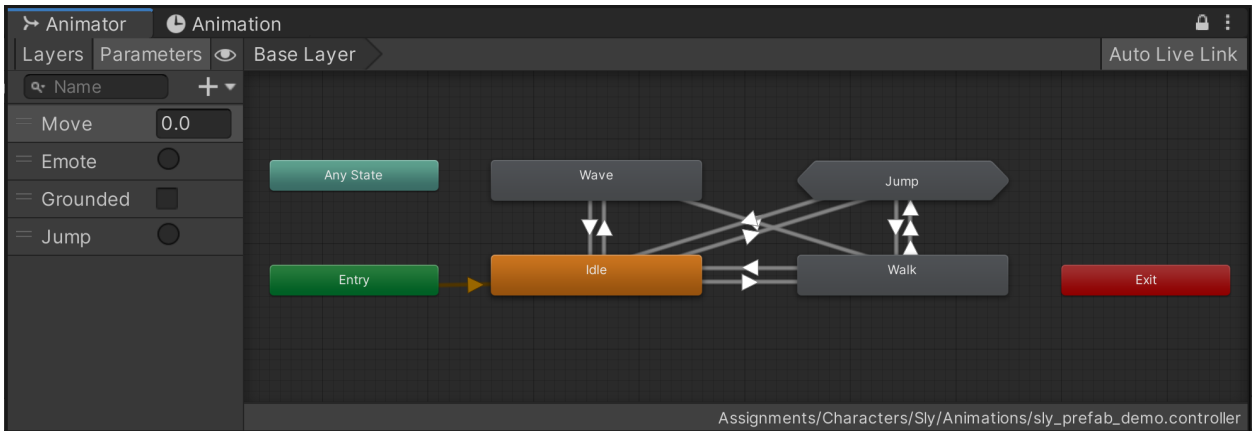
3.8 Make a new prefab variant of your player character

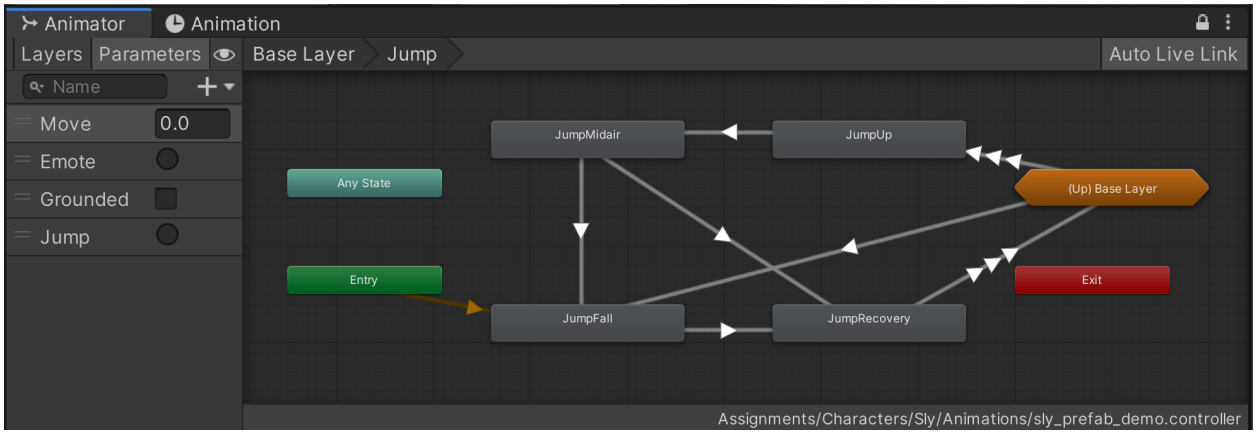
Now that the player controller setup is finished we need to save it as a prefab variant so we can easily add it to other scenes.

1. Drag and drop your player game object into the character Prefabs folder in Project view
    a. Assets/Characters/<YourCharacterName>/Prefabs

End of part 3.

Save screenshots of your finished Animator Controller state machine to Miro to turn in with your assignment. Include screenshots of both the Base Layer and the Jump sub-state machine.





Part 4, Recording a video of walking and jumping

Let's put everything together and capture a short video of the character freeling walking around and jumping in the scene.

1. Save a new copy of the AnimationJumpScene and delete the existing character
2. Drag and drop your player character prefab into the scene
3. Optionally add 3D objects such as cubes of various scale to jump on
4. Add a Cinemachine Follow Camera
    a. Cinemachine → Create Follow Camera
    b. Assign your character to the **Follow** and **Look At** fields
    c. Tune the settings of the Follow Camera to get a good view of the scenery
5. Walk around the scene and try jumping
    a. Check to make sure you can jump high enough for your character's design
    b. Try falling off ledges and make sure the fall and fall recovery transitions play
    c. Fine tune the settings and save the overrides to the prefab

End of part 4.

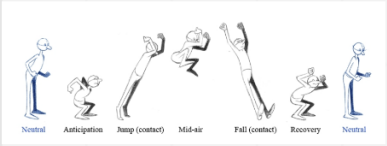
Capture a gif or video of your character walking and jumping around in the focal point of your world. Add it to Miro to turn in with your assignment.

Grading rubric

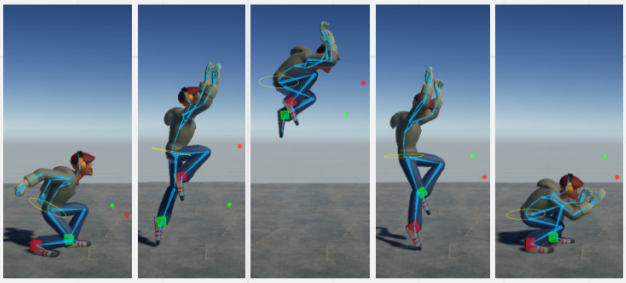
Criteria	**Achievement levels**			
	level 1	level 2	level3	level4
Jump poses format: images [Miro]	20 points: (0 incorrect) Screenshots show all 5 main jump poses both with and without rig controls	18 points: (1-2 incorrect) Screenshots show all 5 main jump poses both with and without rig controls	16 points: (3+ incorrect) Screenshots show all 5 main jump poses both with and without rig controls	0 points: files were missing
Dopesheet main jump poses format: images [Miro]	20 points: (0 incorrect) Two Dopesheet screenshots of complete rows of keys. One with keys every 10 frames and one with timing	18 points: (1 - 2 incorrect) Two Dopesheet screenshots of complete rows of keys. One with keys every 10 frames and one with timing	16 points: (3+ incorrect) Two Dopesheet screenshots of complete rows of keys. One with keys every 10 frames and one with timing	0 points: files were missing
Jump animation format: gif/video [Miro]	20 points: (0 incorrect) Jump animation has all 5 main poses plus idle at start and end. Timing feels believable. Upward trajectory is smooth	18 points: (1-2 incorrect) Jump animation has all 5 main poses plus idle at start and end. Timing feels believable. Upward trajectory is smooth	16 points: (3+ incorrect) Jump animation has all 5 main poses plus idle at start and end. Timing feels believable. Upward trajectory is smooth	0 points: files were missing
Player controller state machine format: images [Miro]	20 points: (0 incorrect) Two screenshots of the Animator window with Base Layer and Jump sub-state machine.	18 points: (1-2 incorrect) Two screenshots of the Animator window with Base Layer and Jump sub-state machine.	16 points: (3+ incorrect) Two screenshots of the Animator window with Base Layer and Jump sub-state machine.	0 points: files were missing
Jumping in the world format: gif/video [Miro]	20 points: (0 incorrect) Screen capture of player walking and jumping in a scene	18 points: (1-2 incorrect) Screen capture of player walking and jumping in a scene	16 points: (3+ incorrect) Screen capture of player walking and jumping in a scene	0 points: files were missing

Character assignment 8 - jump

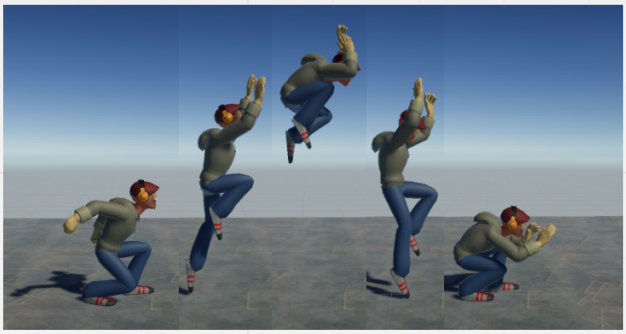
Jump reference



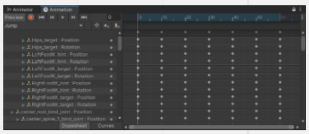
Jump poses - rig



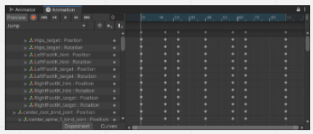
Jump poses



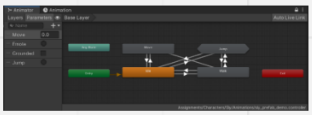
Dopesheet - jump keyframes



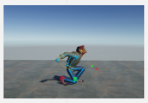
Dopesheet - jump timing



Third person rigid body player controller



Jump sub-state machine



Jump animation

Jumping around the world