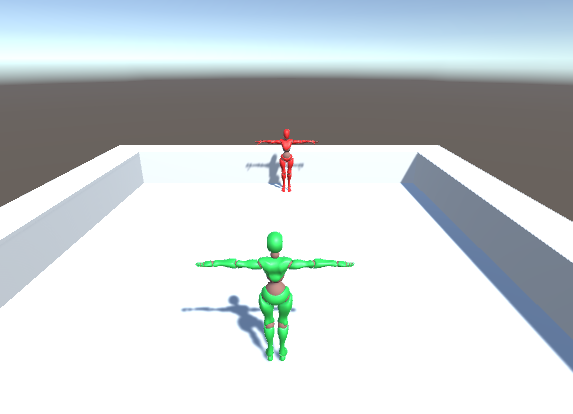
**Lab 5 : Setting Up Animations and Game Mechanics**

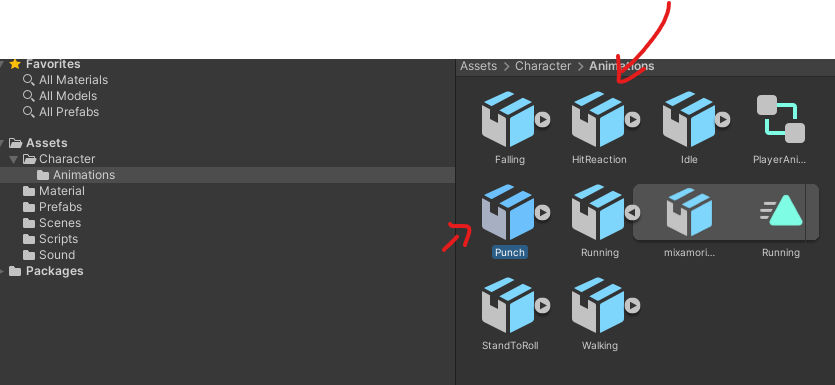
Follow these step-by-step instructions to complete the lab tasks.

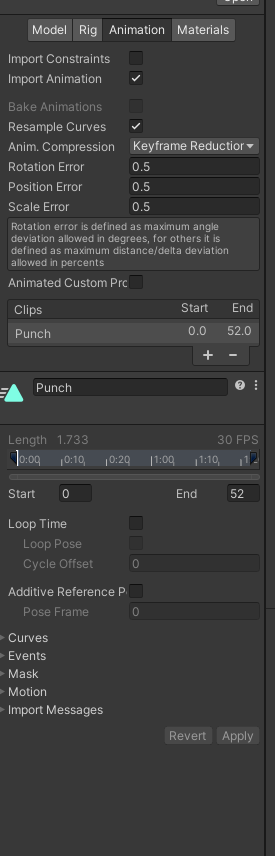
Open the git assignment :

<https://classroom.github.com/a/u1C4vjaH>



### **Setup the Punch and Hit Reaction Animation**

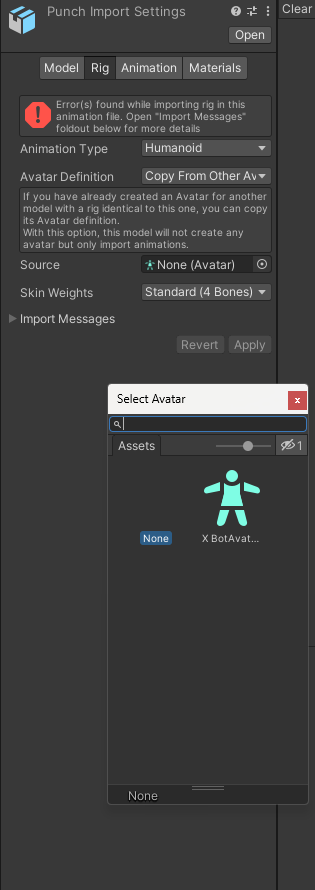
1. **Set Up Animations:**
   * Select each animation file.

In the **Inspector**, go to the **Rig** tab and set **Animation Type** to **Humanoid**.

Set **Avatar Definition** to **Copy From Other Avatar**.



Select the enemy avatar or create one if not existing. Click **Apply**.



Do the same thing for the HitReaction Animation

Bake the rotation into the pose

### **Select the Animation Clip**

* In the **Project** window, navigate to the folder where your character’s animations are stored. In this case it is under Character/Animations.
* Click on the **Punch** animation clip so its **Inspector** details appear.

### **Open the Rig Settings**

* In the **Inspector**, you’ll see the animation import settings (these are visible if the animation is part of an FBX model).
* Across the top, click on the **Animations** tab.

### **Locate the Root Transform Rotation Section**

* Scroll down until you find the section called **Root Transform Rotation**.
* You’ll see options like:  
  + **Based Upon (Original/Root Node/Body Orientation)**
  + **Offset**
  + **Bake Into Pose** (the checkbox we need).

### **Enable Bake Into Pose**

* Check the box **Bake Into Pose**.
* This tells Unity to remove any root rotation from the animation curves and instead apply it directly to the character’s pose.

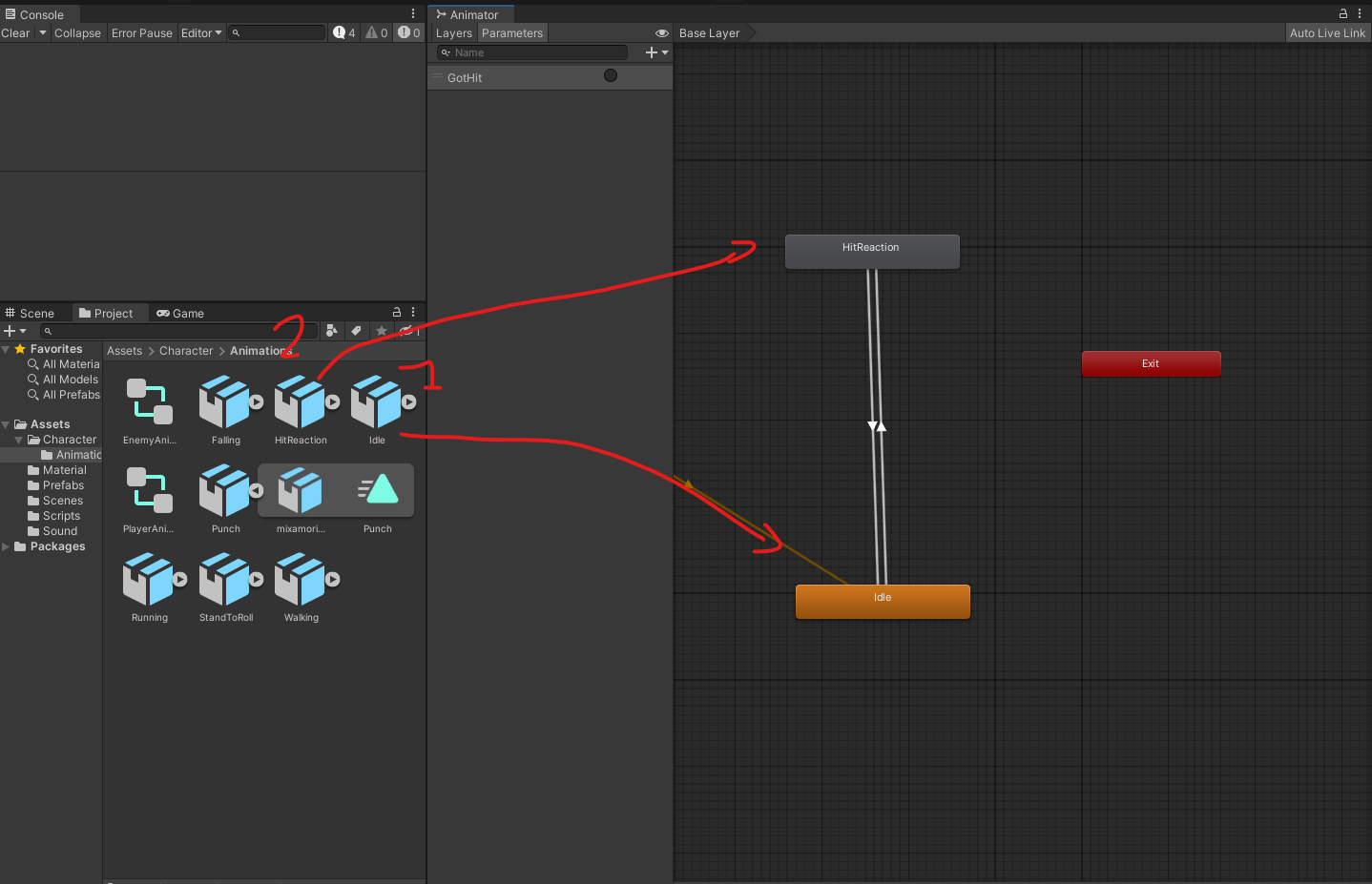
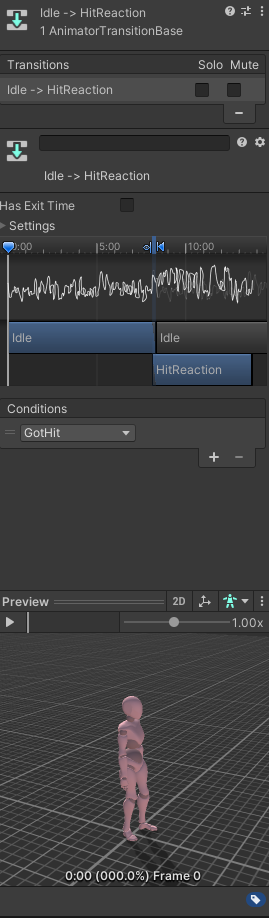
### **Apply Changes**

* Scroll back up and click **Apply** (bottom-right of the Import Settings Inspector).
* Unity will re-import the FBX/animation with the updated settings.

That will make Unity ignore the root rotation curve and apply the rotation directly to the character’s pose, preventing unwanted turning or spinning during the Punch animation.

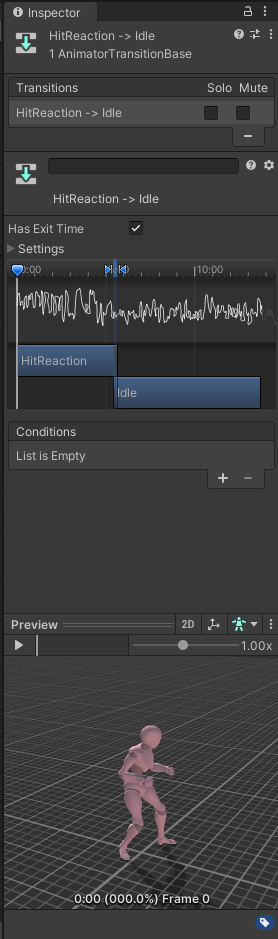


### **Create an Animator for the Enemy**

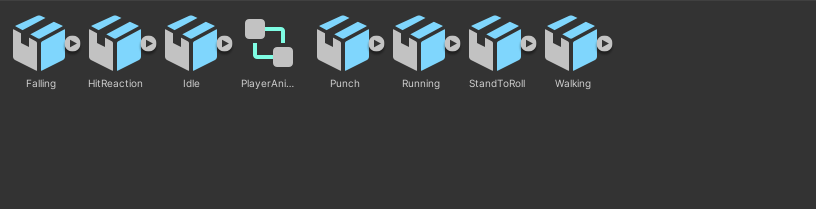
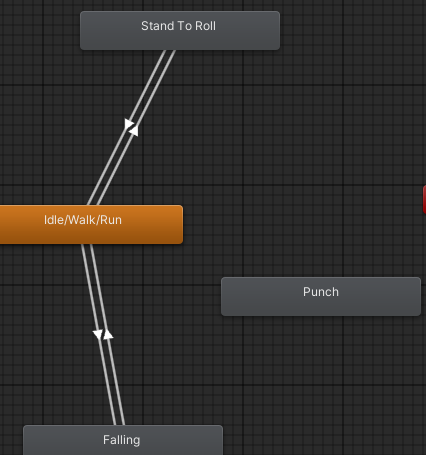
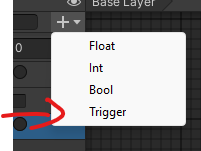
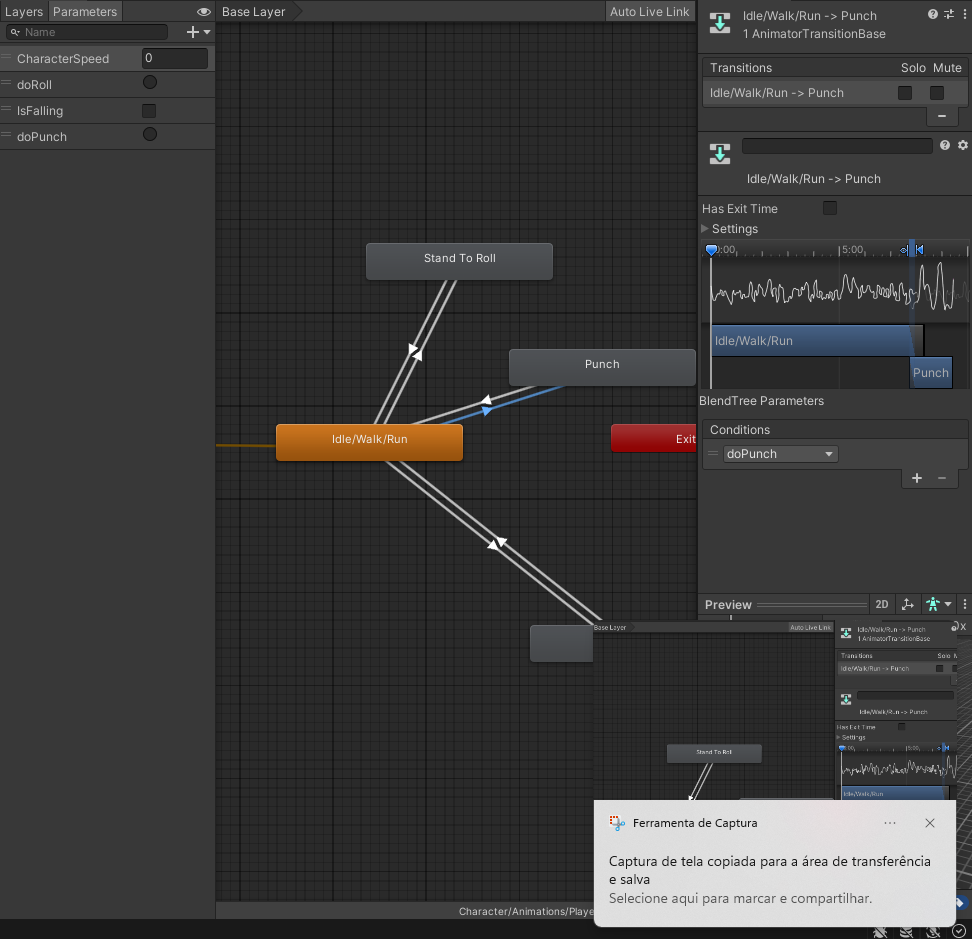
1. **Create Animator Controller:**
   * Right-click in the **Project** window and select **Create > Animator Controller**.
   * Name it EnemyAnimator.
2. **Assign Animator to Enemy:**
   * Select your enemy GameObject in the **Hierarchy**.
   * In the **Inspector**, click **Add Component** and choose **Animator**.
   * Drag and drop the EnemyAnimator onto the **Controller** field of the Animator component.
3. **Open Animator Window:**
   * Select EnemyAnimator.
   * Go to **Window > Animator** to open the Animator window.
4. **Add Idle Animation:**
   * Drag the idle animation clip from the **Project** window into the Animator.
   * Set it as the **Default State** by right-clicking and selecting **Set as Layer Default State**.
   * 
5. **Add Hit Reaction Animation:**
   * Drag the hit reaction animation clip into the Animator.
6. **Create Transition with Trigger:**
   * Select the transition from **Idle** to **Hit Reaction**.
   * In the **Inspector**, under **Conditions**, add a new **Trigger** parameter (e.g., GotHit).
   * Set the condition for the transition to the GotHit trigger.
   * 

**Setup a Transition Back to Idle:**

Create a transition from **Hit Reaction** back to **Idle**.Make sure the only condition is HasExitTime for the Transition back to Idle.



### **Modify the Player Animator to Add the Attack ( Punch ) Animations**

1. **Open Player Animator Controller:**
   * Select your player's Animator Controller (e.g., PlayerAnimator).
   * 
   * Open the Animator window.
2. **Add the Punch Animations:**
   * Drag the Punch animation clips into the Animator.
   * 
3. **Create Parameters:**
   * In the Animator window, click the **Parameters** tab.
   * 
   * Add a **Trigger** parameter for the attack (e.g., doPunch)
   * 
4. **Set Up Transitions:**
   * Create transitions from **Idle/walk/run** to the Punch animation.
   * Set the condition for each transition to its corresponding trigger.
   * 
5. **Transition Back to Idle/walk/run:**
   * Create transitions from each attack animation back to **Idle/walk/run** or **Movement**.
   * Make sure to enable HasExitTime for the transition back to idle.
6. Modify the PlayerAnimationManager to add your Trigger like so :

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PlayerAnimationManager : MonoBehaviour

{

private Animator animator;

private PlayerMovement movement;

private Rigidbody rb;

public void Start()

{

animator = GetComponent<Animator>();

movement = GetComponent<PlayerMovement>();

rb = GetComponent<Rigidbody>();

}

public void Update()

{

animator.SetFloat("CharacterSpeed", rb.velocity.magnitude);

animator.SetBool("IsGrounded",movement.isGrounded);

if (Input.GetButtonUp("Fire1"))

{

animator.SetTrigger("doRoll");

}

if (Input.GetButtonUp("Fire2"))

{

animator.SetTrigger("doPunch");

}

}

}

### **Add to the Colliders to the fists**

### **Access the Character's Bone Hierarchy**

1. **Expand the Character's Hierarchy:**
   * In the **Hierarchy** window, expand your character's GameObject by clicking the arrow next to its name.
   * Continue expanding the child objects until you reach the hand bones.
2. **Locate the Hand Bones:**
   * Typically, the path to the hand bones is:
     + **CharacterName**
       - **Hips**
         * **Spine**

**Chest**

**UpperChest** (if available)

**LeftShoulder** / **RightShoulder**

**LeftArm** / **RightArm**

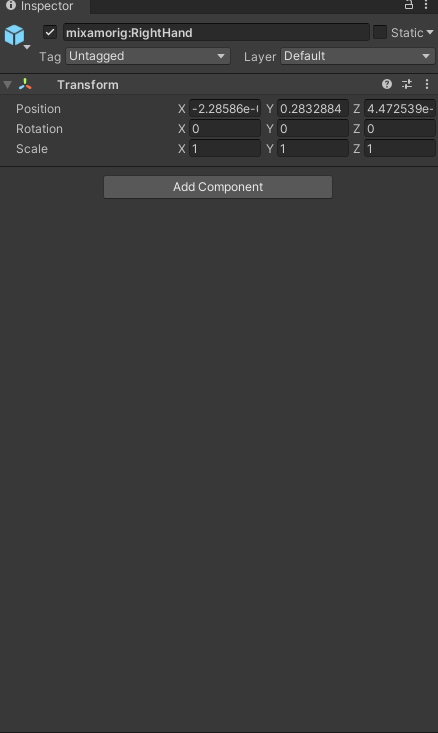
**Left ForeArm/RightForearm**

**LeftHand/RigthHand,**

### **Step 4: Add Colliders to the Hands**

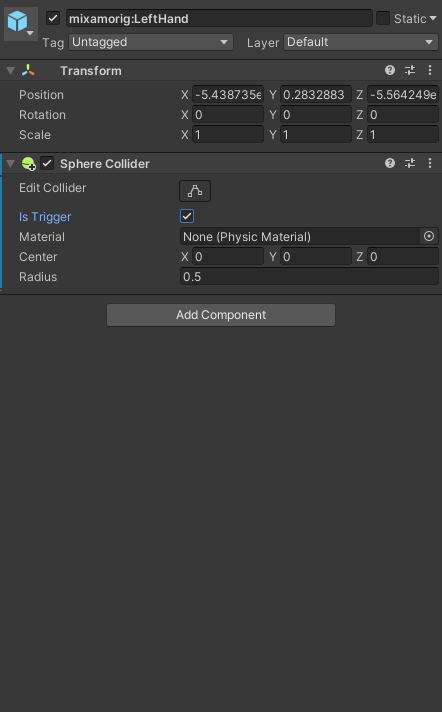
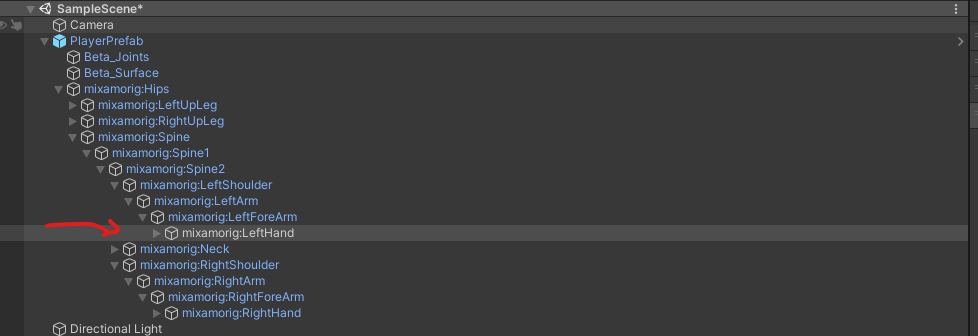
#### **For the Right Hand:**

1. **Select the RightHand Bone:**
   * **Click on the RightHand GameObject in the Hierarchy.**
   * **Note : If you want to add a weapon to a Character you would add it as a child of the RightHand object**
2. **Add a Collider Component:**
   * **In the Inspector, click on Add Component.**
   * **Type Capsule Collider in the search bar and select it.**
3. **Adjust the Collider:**
   * **Center: Adjust the Center values to align the collider with the hand mesh.**
   * **Radius and Height: Modify the Radius and Height to encompass the hand.**
   * **Direction: Set the Direction (X, Y, or Z) based on the orientation of the hand bone.**
4. **Make sure it is a Trigger**

****

#### **For the Left H**

1. **Select the LeftHand Bone:**
   * **Click on the LeftHand GameObject in the Hierarchy.**
2. **Add a Collider Component:**
   * **Repeat the same steps as for the right hand.**
3. **Adjust the Collider:**
   * **Configure the collider to fit the left hand, similar to how you did for the right hand.**



### **Implement the Hit Detection Logic**

Open the script and write the following code:

using UnityEngine;

public class HitDetector : MonoBehaviour

{

private void OnTriggerEnter(Collider other)

{

if (other.CompareTag("Enemy"))

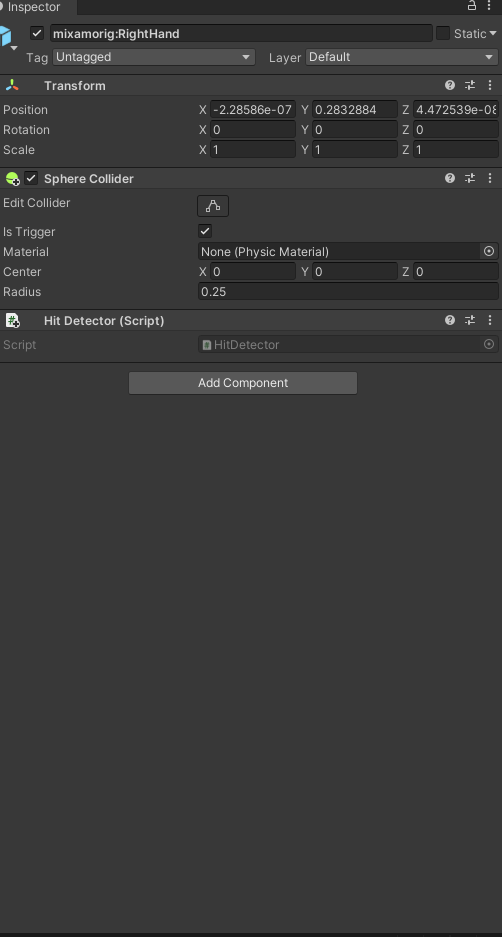
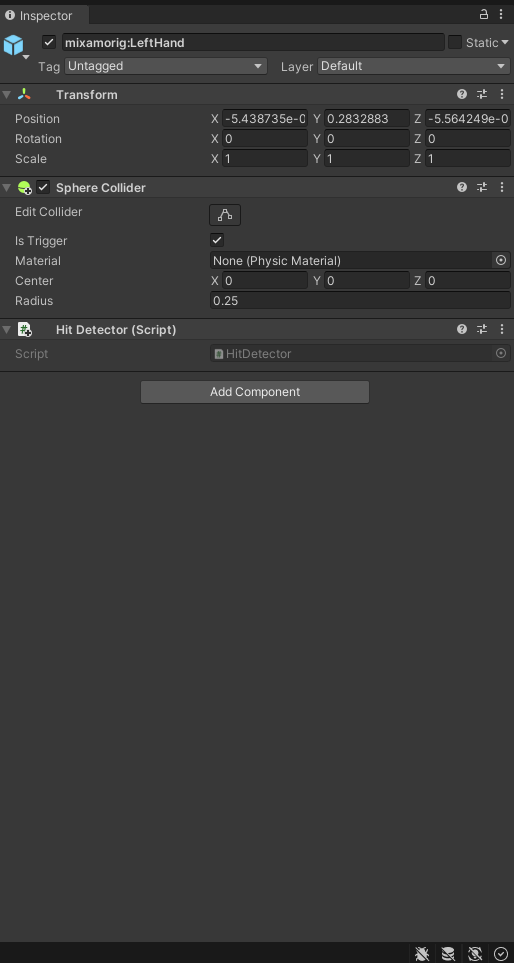
{

}

}

}

Add this script to both fists we will finish modifying it later.



Make sure to set the Collider as Trigger by enabling the IsTrigger Checkbox and also scale down the collider so that they are roughly the size of your player hands.

## **Synchronizing Hit Detection with Animations**

### **Disabling the Collider by Default**

Create a **PlayerHitboxManagerScript** script, modify the Start method to disable the colliders initially we will want to add this script on the Player :

public Collider[] attackColliders;

void Start()

{

foreach( Collider attackCollider in attackColliders)

{

attackCollider.enabled = false; // Disable collider at start

}

}

### **Creating Functions to Enable/Disable the Collider**

Add the following methods to the script:

public void EnableHitbox()

{

foreach( Collider attackCollider in attackColliders)

{

attackCollider.enabled = true;

}

}

public void DisableHitbox()

{

foreach( Collider attackCollider in attackColliders)

{

attackCollider.enabled = false;

}

}

The full script should look like this :

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PlayerHitBoxManager : MonoBehaviour

{

// Start is called before the first frame update

public Collider[] attackColliders;

void Start()

{

foreach( Collider attackCollider in attackColliders)

{

attackCollider.enabled = false; // Disable collider at start

}

}

public void EnableHitbox()

{

foreach( Collider attackCollider in attackColliders)

{

attackCollider.enabled = true;

}

}

public void DisableHitbox()

{

foreach( Collider attackCollider in attackColliders)

{

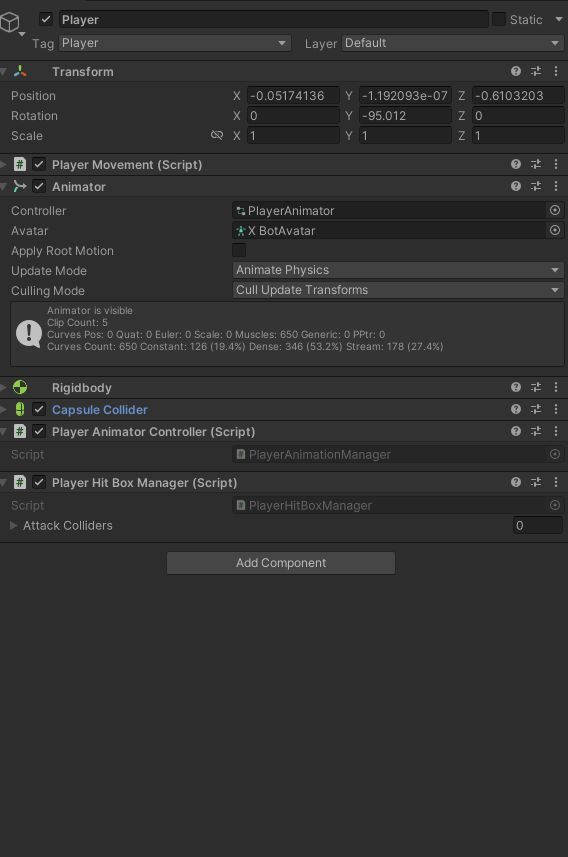
attackCollider.enabled = false;

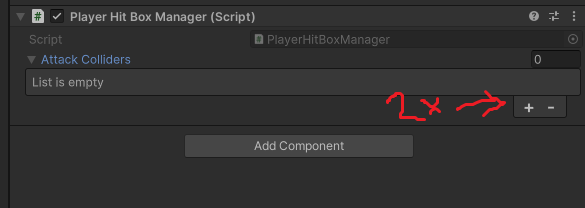
}

}

}

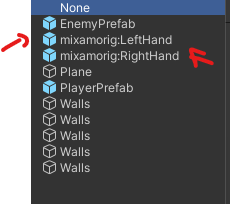
Add the script to the Player

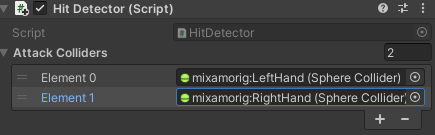




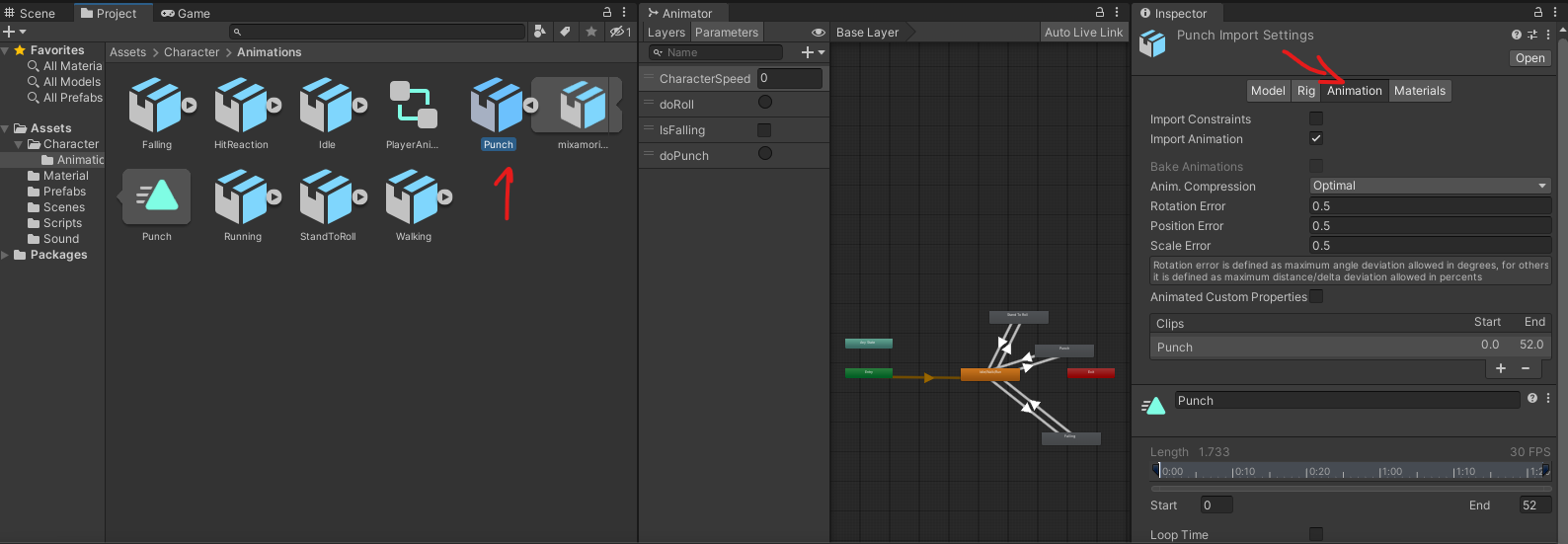
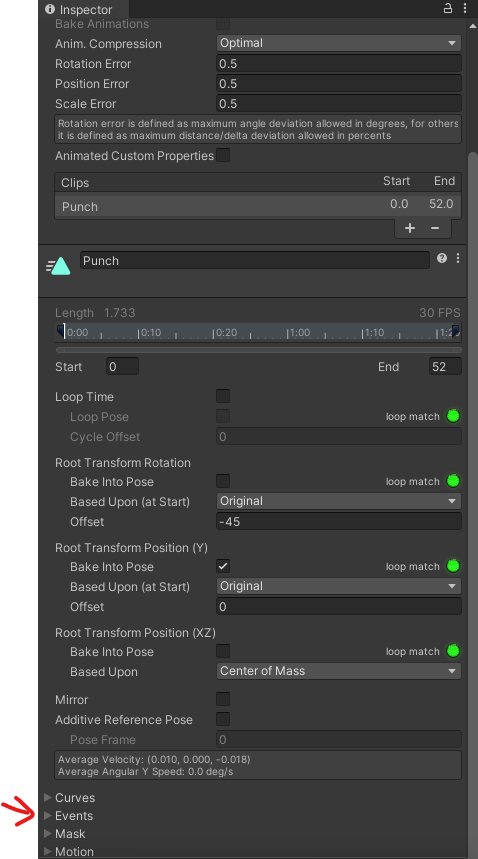
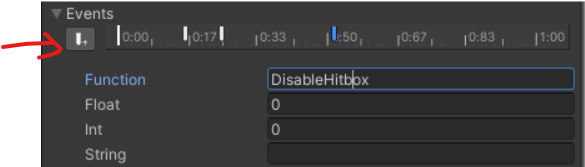
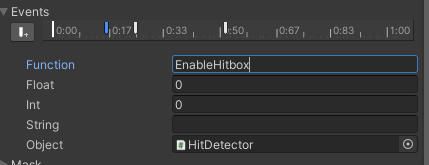
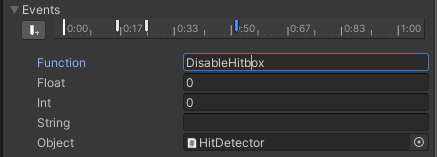
Click the button on the right

Select the two hand colliders you created





### **Adding Animation Events**

* Select the Punch animation in the project Hierachy
* In the animation Tab go down to the section called Events
* 
* 
* Click the **Add Event** button (looks like a white marker).
* 
* In the **Inspector**, set the first marker to around 0:17 and set the function to EnableHitbox.
* 
* Move to the frame where the attack should stop dealing damage around 0:50 Add another event and set the function to DisableHitbox.
* 

**Creating the Enemy Animation Controller**

1. **Create Hit Sound:**
   * Import the hit sound effect into the **Project** window.
2. **Add Audio Source to Enemy:**
   * Select the enemy GameObject.
   * Click **Add Component** and choose **Audio Source**.
   * Uncheck **Play On Awake**.
3. **Create Enemy Script:**
   * In the **Project** window, create a new C# script named EnemyController.
4. **Edit the Enemy Script:**

using UnityEngine;

public class EnemyController : MonoBehaviour

{

private Animator animator;

void Start()

{

animator = GetComponent<Animator>();

}

public void GotHit()

{

animator.SetTrigger("GotHit");

}

}

Modify the Hit detector logic to so that it communicates with the enemy controller

using UnityEngine;

public class HitDetector : MonoBehaviour

{

private void OnTriggerEnter(Collider other)

{

if (other.CompareTag("Enemy"))

{

EnemyController enemyController = other.GetComponent<EnemyController>();

if (enemyController != null)

{

enemyController.GotHit();

}

}

}

Then drag and drop your EnemyController Script on your Enemy Gameobject in your Scene graph.

Your Enemy should now react to getting hit by your punch animation.

What happens if you punch the enemy multiple times. Does he react every single time you hit him ?  
  
Initially, no, the enemy does not react every single time you hit him. This is because the Hit Reaction animation may still be playing when another punch lands, and Unity’s Animator by default won’t interrupt an animation that’s already playing.

Lets fix this

**Make the Enemy Hit Reaction Play Even If Already Playing**

1. **Open the Animator Window**:

○ Select your enemy character in the Hierarchy.

○ Open the Animator window by navigating to Window > Animation > Animator.

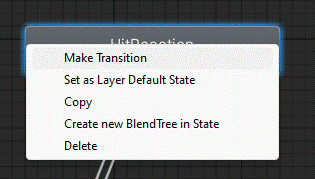
1. **Locate the Hit Reaction Animation**:

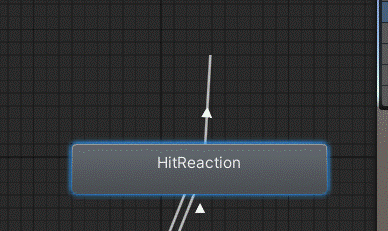
○ Find the "Hit Reaction" animation node in the Animator graph.

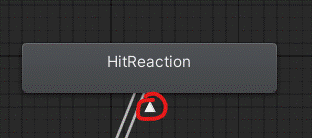
1. **Set Up the Transition**:

○ Right-click on the Hit Reaction animation node and select Make Transition.

○ **Drag the arrow back to the same** Hit Reaction animation node (creating a loop transition).

○ 

○ 



○ If it work it will have a little arrow.

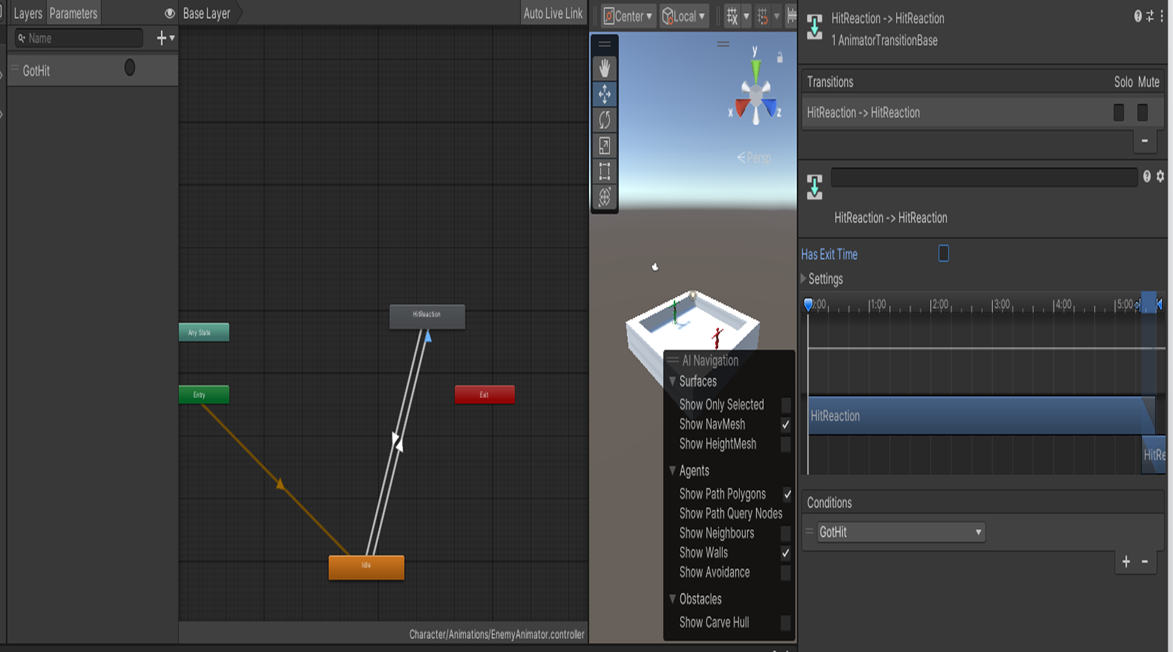
**Configure the Transition Conditions**:

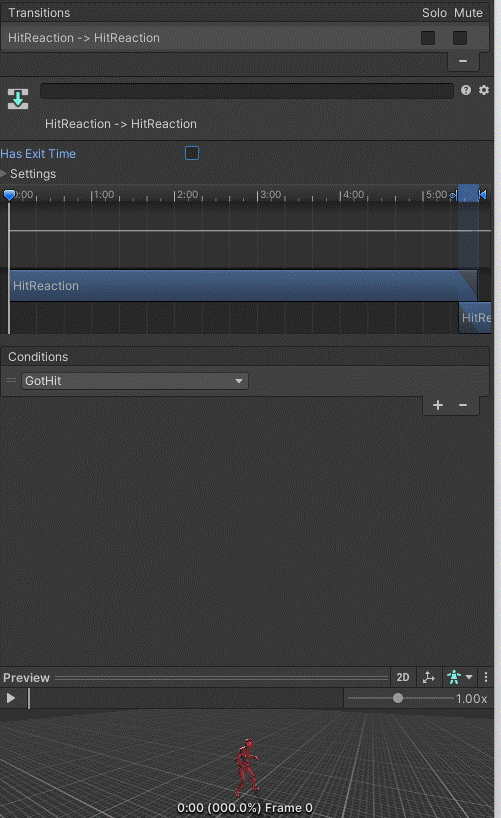
○ Click on the newly created transition arrow.

○ In the Inspector window, find the Conditions section.

○ Set the condition to GotHit

○ Disable Has Exit Time and set the Transition Duration to 0 to ensure the animation can interrupt itself instantly.





**Test the Transition**:

○ Play the game and ensure that hitting the enemy multiple times triggers the Hit Reaction each time, even if it was already playing.

## **Lets add some simple UI to our game.**

Now lets a create a Game Manager and make it track our score. Our score will increase every time I hit the Enemy.

On the game object GameManager add the following script :

using UnityEngine;

using UnityEngine.UI;

public class GameManager : MonoBehaviour

{

public static GameManager Instance { get; private set; }

public Text scoreText;

private int score = 0;

private void Awake()

{

// Enforce one instance

if (Instance != null && Instance != this)

{

Destroy(gameObject);

return;

}

Instance = this;

}

public void AddScore(int amount)

{

score += amount;

UpdateUI();

}

private void UpdateUI()

{

if (scoreText != null)

{

scoreText.text = $"Score: {score}";

}

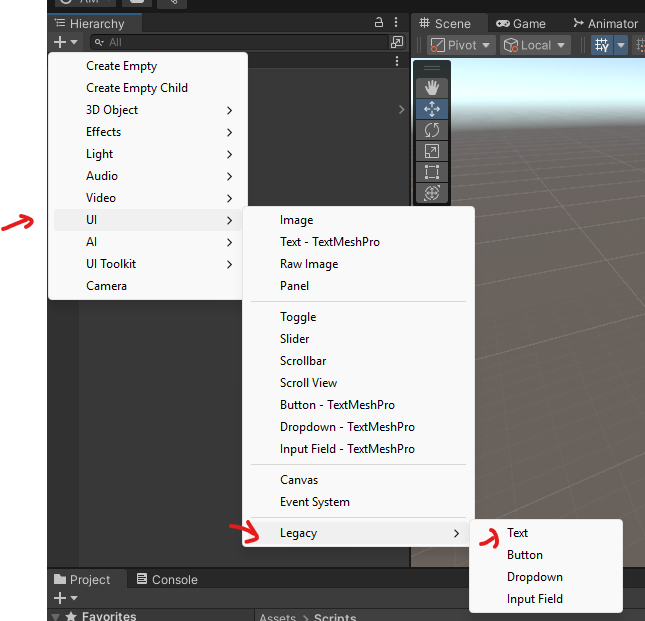
}

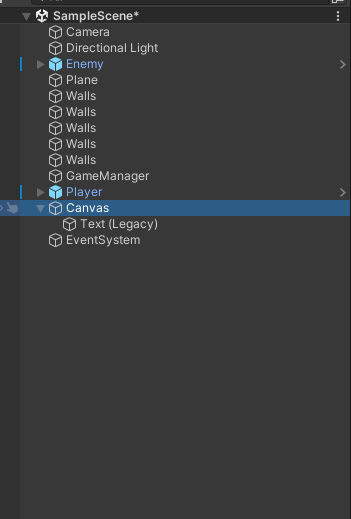
}



**This will hold the score and update the UI. It is a singleton so it’s easy to access.**

**Now lets create the UI element that we will use for the Score :**

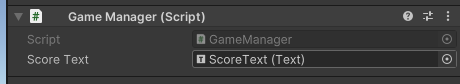
**  
You should see**

****

**Rename Text ( Legacy ) with ScoreText.**

****

**And drag the ScoreText game object in the GameManager script**

****

## **2) Enemy notifies the GameManager**

**Update your EnemyController so it calls the GameManager when you hit the enemy like so.**

**using UnityEngine;**

**public class EnemyController : MonoBehaviour**

**{**

**private Animator animator;**

**void Start()**

**{**

**animator = GetComponent<Animator>();**

**}**

**public void GotHit()**

**{**

**animator.SetTrigger("GotHit");**

**// Notify GameManager**

**if (GameManager.Instance != null)**

**{**

**GameManager.Instance.AddScore(1); // Add 1 point per hit**

**}**

**}**

**}**

**Your Game should now have a score that keeps getting up everytime you hit the Enemy and the Enemy Should react to your punches.**

Submit your project on github.