Documentation for

Push and Pull State

for MalberS Animal Controller.

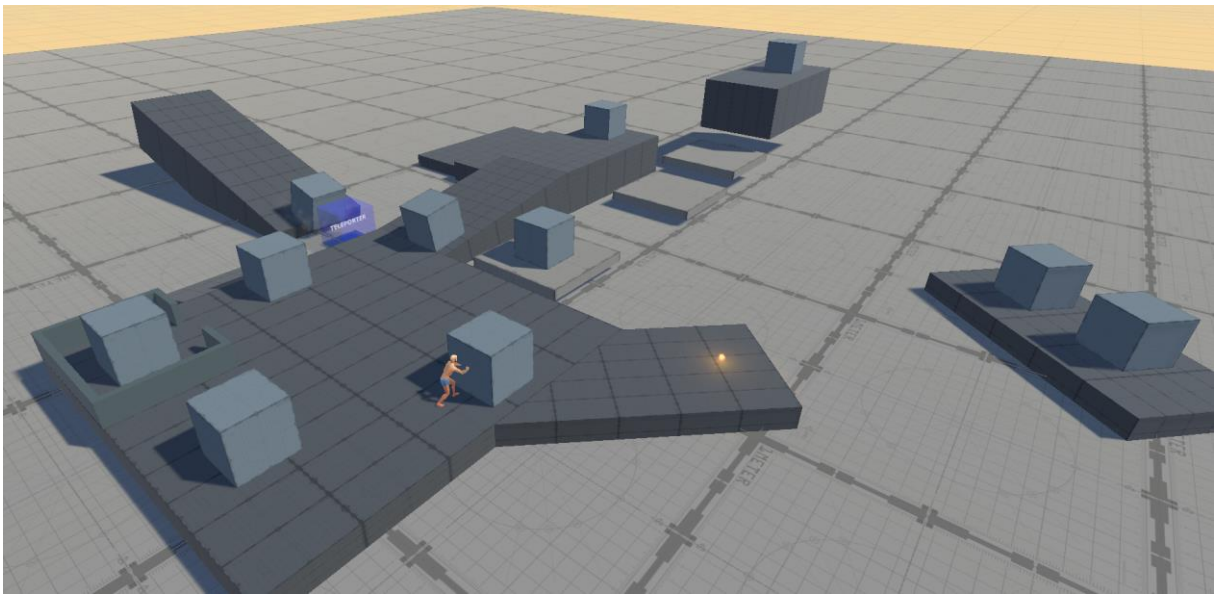


Table of contents

1. Disclaimer:

This push and pull state for Malbers Animal Controller is provided for your use and modification according to your specific project needs. However, please be aware of the following:

1.1 Dependency on Malbers Animations Assets:

This push and pull state is designed to work with Malbers Animations' assets, specifically the Animal Controller or Horse Animset Pro. It relies on functionalities provided by these assets and may not function correctly without them.

1.2 Assets Not Included:

This asset itself does not include any animations, textures, or meshes. It is primarily scripts that enhances the interaction between the player character and objects within the game world.

1.3 Usage and Modification:

You are free to use and modify this asset according to your project's requirements. Feel free to adapt the code, tweak parameters, or extend its functionality to better suit your needs.

1.4 No Guarantees:

While effort has been made to ensure the functionality and reliability of this asset, there are no guarantees of its performance in all situations or environments. It is recommended to thoroughly test the asset in your specific project context.

1.5 Support and Updates:

Support and updates for this asset may be limited. If you encounter issues or have questions about its usage, you may need to contact me or adapt the asset yourself.

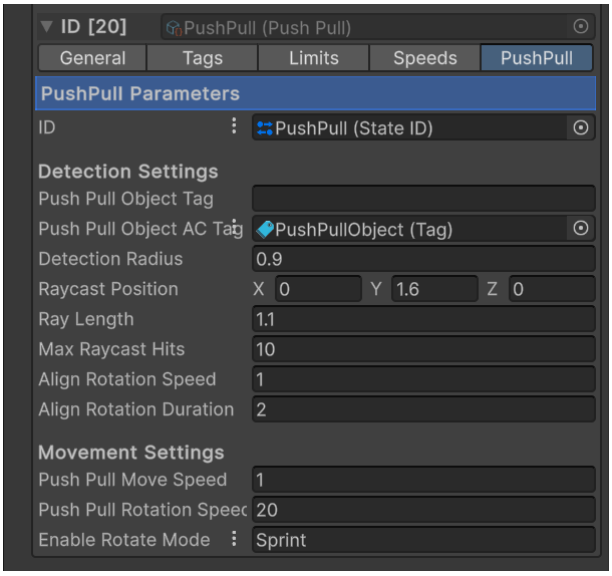
By using this push and pull state, you acknowledge and accept the above disclaimers and agree to use the asset at your own risk. If you have any concerns or questions about this asset, please feel free to reach out to me.

2. PushPull Script

2.1 Overview

The PushPull script represents a state that allows a character to push and pull objects within the game environment. It provides functionality for detecting push/pull objects, handling movement, rotation, and collision checking.

2.2 Detection Settings



detectionTag: Unity tag used to identify push/pull objects.

pushPullObjectTag: Malbers tag used to identify push/pull objects.

detectionRadius: Radius for detecting push/pull objects.

raycastPosition: Position offset for the raycast used for detection.

rayLength: Length of the raycast used for detection.

alignRotationSpeed: Speed of rotation when aligning to push/pull objects.

alignRotationDuration: Duration for character align rotation.

2.3 Movement Settings

pushPullMoveSpeed: Movement speed of the character and the push/pull object.

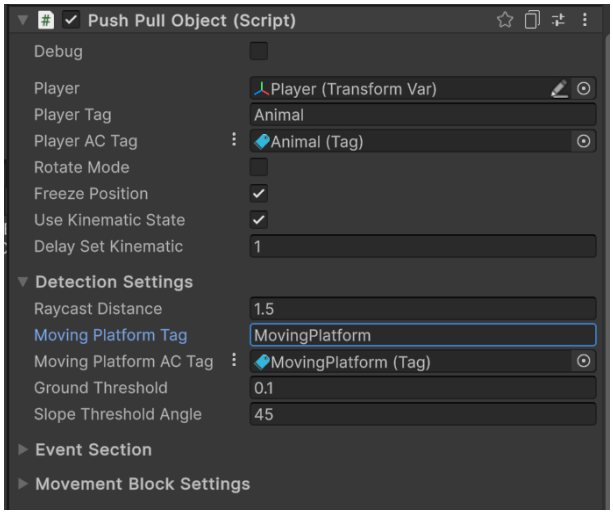
pushPullRotationSpeed: Rotation speed of the character and the push/pull object.

enableRotateMode: Set a string value from an input that you would like to use to enable the rotation mode on the push-pull object.

3. PushPullObject Script

3.1 Overview

The PushPullObject script provides functionality for objects that can be pushed and pulled by the player within the game environment. It handles ground detection, collision detection, and event triggering based on movement thresholds.



debug: Enables debug mode to receive console reports.

player: Reference to the player's transform variable.

playerTag: Tag used to identify the player (Unity Tag-System).

playerACTag: Tag used to identify the player (Malbers Tag-System).

rotateMode: Enable rotation mode for the object

useKinematicState: Toggle to make the object unaffected by external forces like explosions.

freezePosition: Prevent the push-pull object from sliding down slopes while in push-pull state.

delaySetKinematic: Delay in seconds before the collider stops registering as grounded and applies kinematic settings if grounded.

3.2 Detection Settings

raycastDistance: Distance of the raycast used to detect ground and moving platforms.

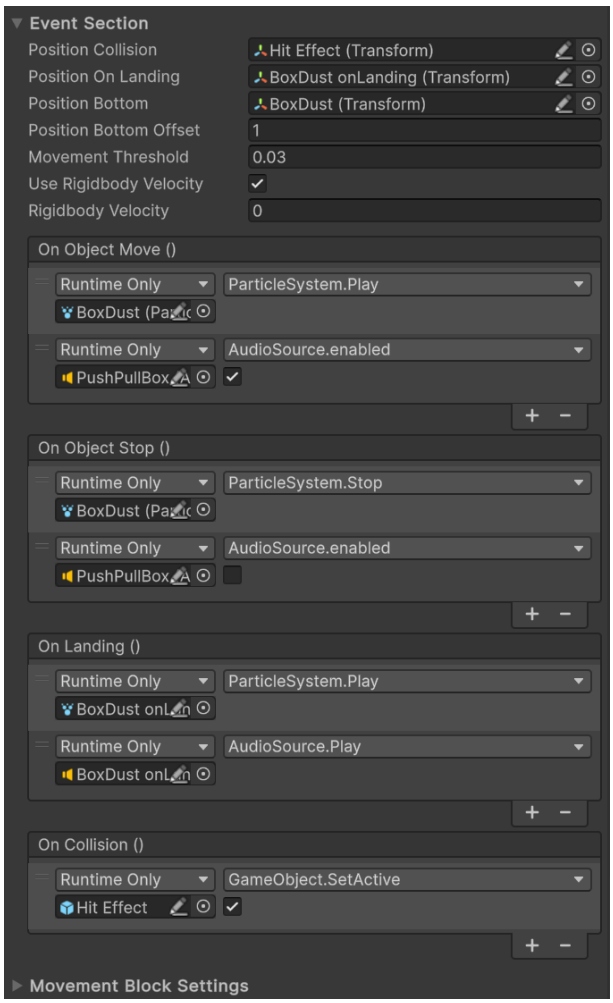
movingPlatformTag: Tag used to identify moving platforms (Unity Tag-System).

movingPlatformACTag: Tag used to identify moving platforms (Malbers Tag-System).

groundThreshold: Vertical distance threshold to determine when the object is considered grounded.

slopeThresholdAngle: Angle limit to distinguish flat ground from sloped surfaces.

3.3 Event Section:



positionCollision: Represents the initial collision position when the object collides.

positionOnLanding: Represents the initial collision position when the object lands.

positionBottom: Denotes a position located underneath the object, suitable for particle effects beneath the object.

positionBottomOffset: Allows adjustment of the vertical offset for the bottom position as needed.

movementThreshold: Movement threshold value. Events trigger based on this threshold.

useRigidbodyVelocity: When enabled, triggers the 'OnMoving' event if the object's velocity exceeds the specified threshold.

rigidbodyVelocity: Specifies the velocity threshold for triggering the 'OnMoving' event.

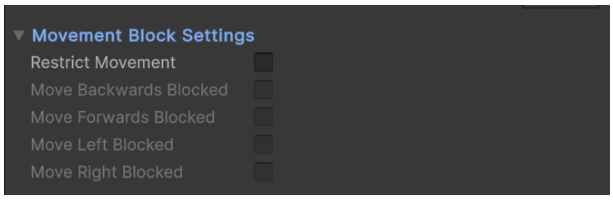
OnObjectMove: Event triggered when the object's movement input is above the movement threshold.

OnObjectStop: Event triggered when the object's movement input is below the movement threshold.

OnLanding: Event triggered when the object becomes grounded.

OnCollision: Event triggered when the object collides.

3.4 Movement Block Settings:



restrictMovement: Can be used to restrict movement. For example, in a 2D scenario, you can block the unused movements.

3.5 Public Methods:

SetRotateMode (bool ShouldRotate): Set the Rotation mode of the PushPull Object.

SetBlockMovement (bool blockBackwards, bool blockForwards, bool blockLeft, bool blockRight): Block movement in specified directions.

GetBlockMovement(): Returns the value of the MovementBlocks.

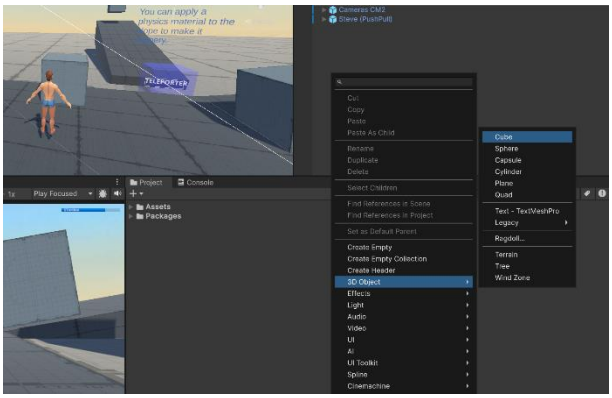
Teleport(Transform targetPosition): Teleport the object to a specified position.

4. Step-by-Step guide

4.1 Setting Up the Object:

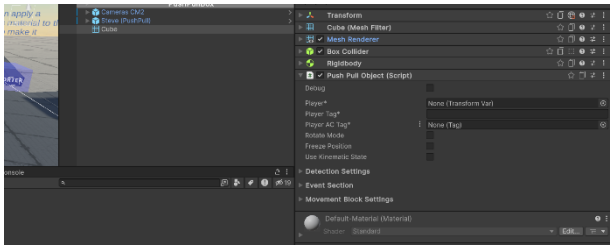
4.1.1 Create the Object:

Create the object you want to push and pull in your Unity scene. Make sure it has a collider attached to it.



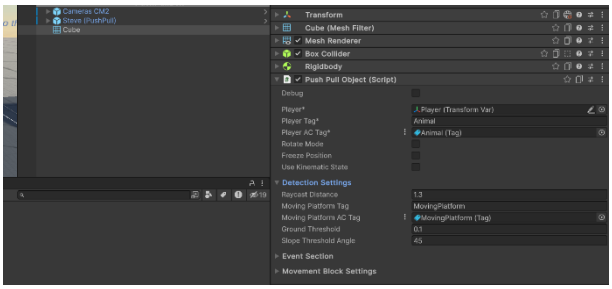
4.1.2 Attach Script:

Attach the PushPullObject script onto the object. This script will handle the push and pull collision and detection logic. A Rigidbody component will be automatically attached as well.



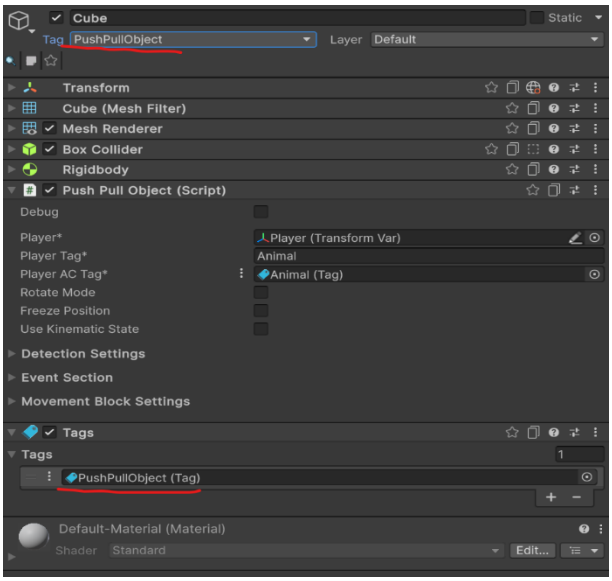
4.1.3 Set Script Variables:

In the Inspector window, set up the variables of the PushPullObject script according to your requirements. These variables include tags for identifying states, movement thresholds, and other parameters.



4.1.4 Tag Configuration:

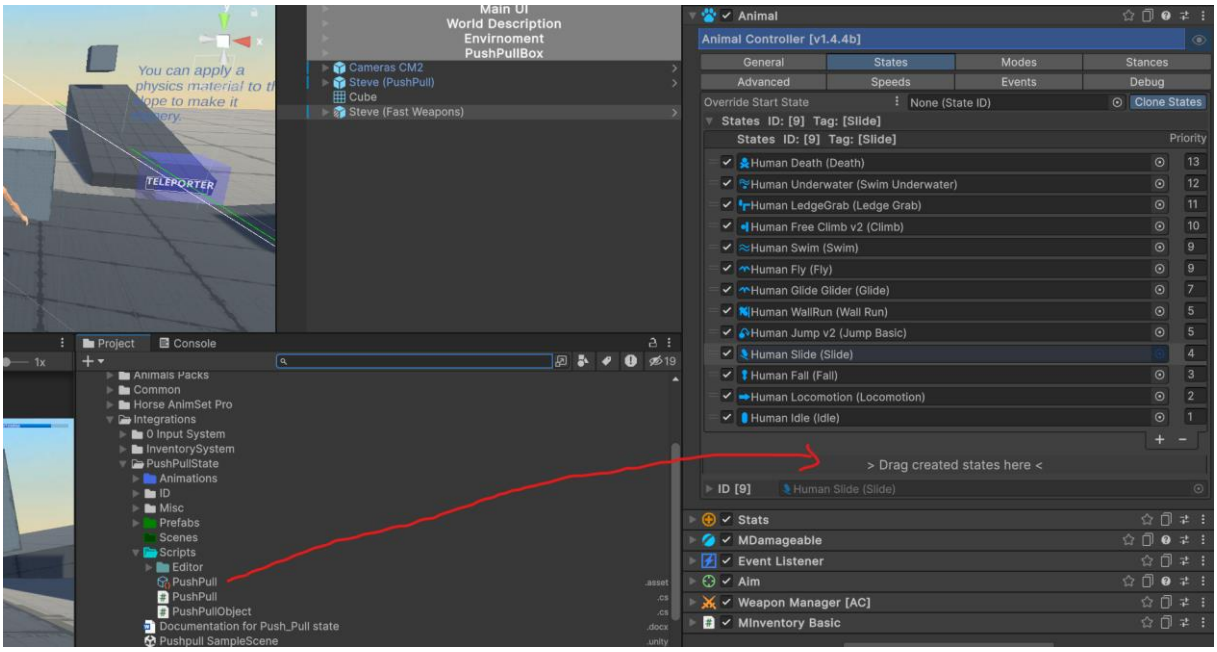
Ensure that the object has tags used to identify it. You can use either Unity's tag system or Malbers Tag-System for this purpose.



4.2 Setting Up the Character:

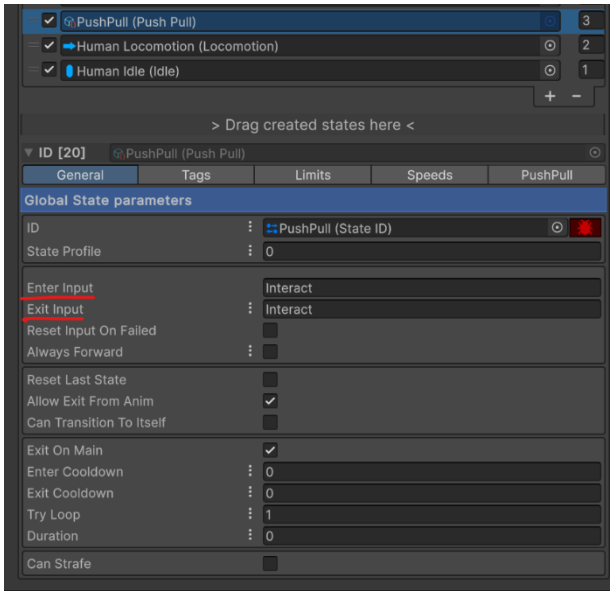
4.2.1 Configure Character State:

Head to your character's setup and drag and drop the scriptable object PushPull into the animal states list. Arrange it in the order you need within the list.



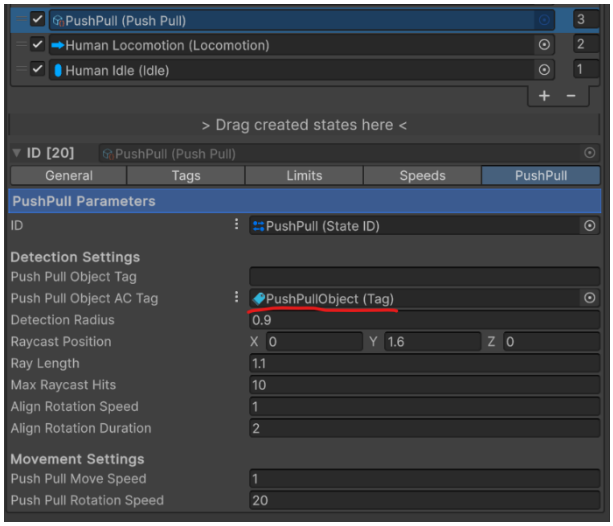
4.2.2 Modify State:

Modify the state settings as per your requirements. Set the input that will activate the state for push and pull actions.



4.2.3 Ensure Tag Matching:

Double-check that the tags used to identify the object's state in the character's scriptable object match the ones set up in the PushPullObject script.



3.2.4 Adjust Ray Height:

Ensure that the height of the ray used for detecting the object can reach the object. Adjust the raycast distance if necessary.

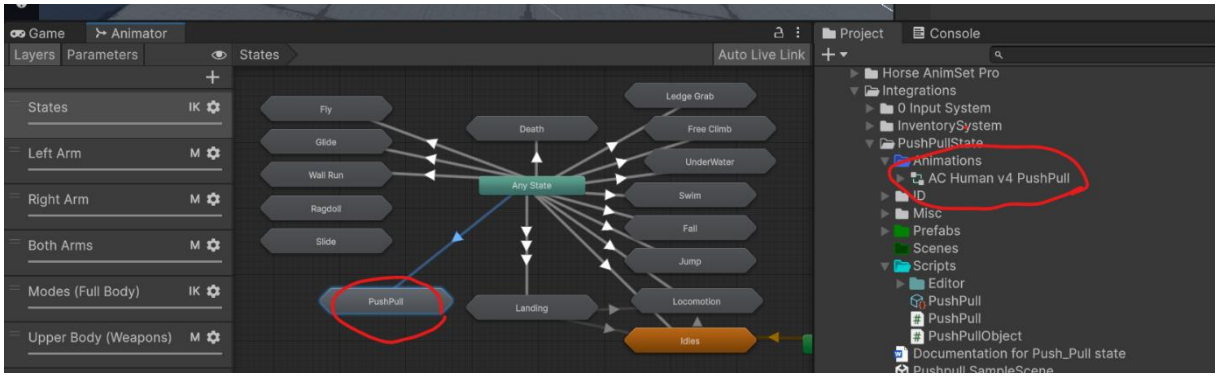
4.2.5 Animator Controller Configuration:

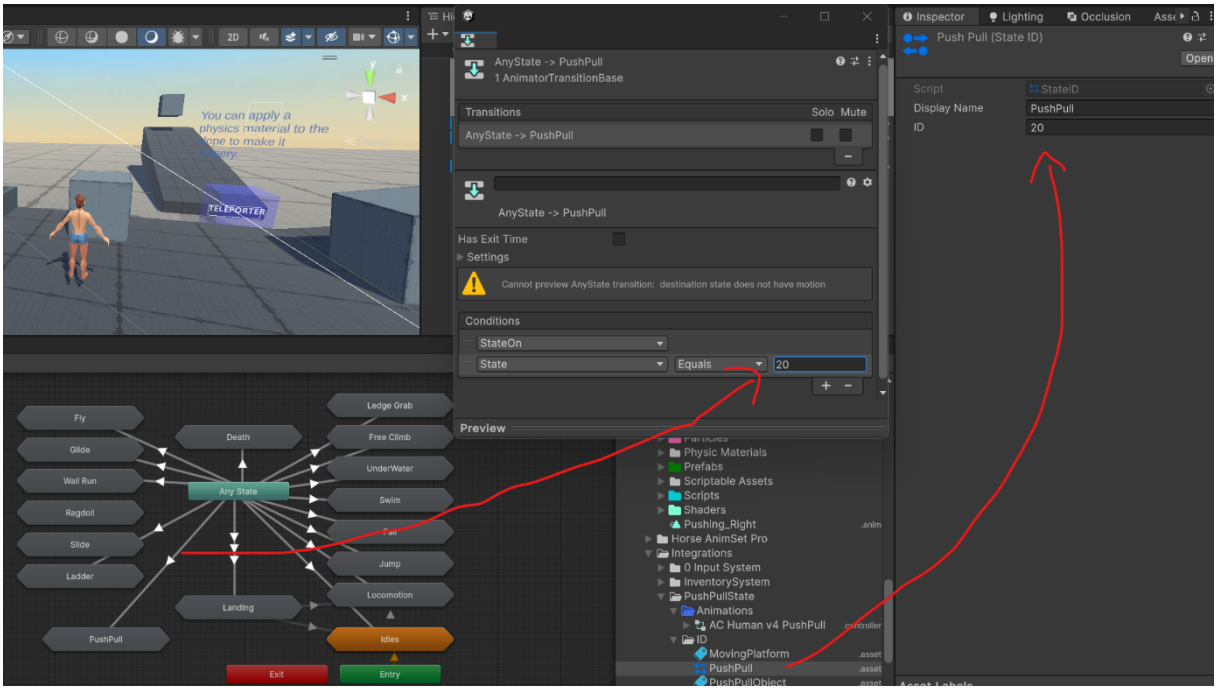
Copy Sub-State with Transition:

Navigate to the animator controller in the project folder of your character. Copy the sub-state containing the transition related to push and pull actions.

Paste into Character's Animator Controller:

Paste the copied sub-state into the animator controller of your character. Ensure that the State ID in the transition matches and is unique within the controller.





4.2.6 Finalization:

Customize Animations:

Now that the setup is complete, you can customize the animations for push and pull actions according to your preferences.

By following these steps, you'll be able to set up an object for push and pull functionality and integrate it seamlessly with your character's interactions.

If you have any ideas for additions or suggestions to enhance the functionality of the asset, please feel free to reach out to me. Your feedback is valuable, and I'm open to incorporating new features or improvements based on user input.