Hybrid Player Controller

Manual

Version: 1.0

Unity Version: Unity 5 & 6

Last Updated: 8/5/2025

Table of Contents

1. Requirements & Setup

Requirements

1. Unity 5 or 6
2. Unity's New Input System
3. TextMeshPro (for demo scenes)

Installation

1. Import the package into your project via the Unity Package Manager or Asset Store.
2. Import the included **TagManager.asset** to ensure all required layers are created.
3. Confirm project layers match those expected by the controller (see Layer & Tag Setup).
4. Open one of the demo levels in **HybridPlayerController/Extras/DemoScenes** and press Play to verify functionality. URP and Built-in render pipeline versions are included.

Layer & Tag Setup

LAYERS:

Ensure the following layers exist and are assigned on the **PlayerController**, and platforming prefabs:

- **World** - On all world objects that the player will come into contact with (walls, floors, ceilings, slopes, etc.) except for platforming elements (see **HybridPlayerController/Prefabs/Platforming**).
- **Platforming** - On platforming element prefabs in **HybridPlayerController/Prefabs/Platforming**.
- **Player** - On the **Player** prefab in **HybridPlayerController/Prefabs**.

TAGS:

Ensure the following tags exist and are assigned on the **PlayerController**, and platforming prefabs:

- "**Player**" - On **Player** prefab in **HybridPlayerController/Prefabs.**
- "**SwingBar**" - On the **SwingBar** prefab in **HybridPlayerController/Platforming**.
- "**GrappleSurface**" - On any surface that is intended to be aimed at, and grappled. And the **GrappleSurface** prefab in **HybridPlayerController/Prefabs/Platforming**.

IMPORTANT:

On the **Player** prefab's inspector, you must assign the **PlatformingLayer**, **WorldLayer**, and **PlayerLayer** fields.

2. Quick Start

Opening a Demo Scene

Navigate to **Assets/HybridPlayerController/Extras/DemoScenes** and open a scene. URP and Built-in render pipeline versions are included.

Playing the Demo

Press Play in the Unity Editor and use the default input mappings to test traversal abilities.

Integrating into Your Project

1. Drag the Player prefab into your scene (**Assets/HybridPlayerController/Prefabs**).
2. Adjust starting position and camera offsets as desired.
3. Review scripts on the prefab for tunable parameters.
4. Test and iterate.

3. Key Features

1. First- and third-person camera modes with configurable positions, field of view, and sensitivity.
2. Modular state machine architecture with numerous built-in movement states.
3. Traversal abilities: ledge climbing, wall running, vaulting, swing bars, grappling, diving, and more.
4. Robust ground, slope, and wall detection for realistic movement.
5. Ability locking system via **UnlockStateTrigge**r.
6. Seamless interaction with moving platforms.
7. Fully configured with the new Input System.
8. Checkpoint handling and respawn logic included.

4. Player States

State	Description
Idle	Player stands still.
Walk	Standard walking movement.
Sprint	Faster ground movement.
Jump	Standard jump with double jumps if desired.
Rising	Vertical movement after jump.
Falling	Downward movement with coyote time.
Dive	Quick forward dive motion.
Crouch	Lower stance for small spaces.
CrouchWalk	Movement while crouched.
Slide	Ground slide initiated from sprint.
Slip	Automatic slide down steep slopes.
Ledge	Grab, hang, and move along ledges.
Vault	Vault low walls.
VaultJump	Optional jump off a vault.
SwingBar	Swing from horizontal bars.
WallRun	Run along walls from sprint.
Grapple	Swing via a grappling hook.

5. Input Reference

Keyboard & Mouse

1. Move: WASD / Arrow Keys
2. Look: Mouse Movement
3. Jump: Space
4. Sprint: Left Shift
5. Crouch/Dive: Left Ctrl
6. Fire/Interact: Left Mouse Button
7. Pause: Tab

Controller

1. Move: Left Stick
2. Look: Right Stick
3. Jump: South Button
4. Sprint: Left Stick Press
5. Crouch/Dive: West Button
6. Fire/Interact: Right Trigger
7. Pause: Start

6. Component Overview

1. **PlayerController.cs** – Core movement logic and state transitions.
2. **PlayerUtils.c**s – Parameters for debug drawing and default locked/unlocked player abilities.
3. **CamUtils.cs** – Camera positioning, roll, and FOV adjustments.
4. **WallChecker.cs / GroundChecker.cs** – Environment detection for the PlayerController and individual states.
5. **MovingPlatform.cs** – Handles player sticking to moving objects.
6. **UnlockStateTrigger.cs** – Enables abilities during gameplay.
7. **HybridPlayerControls.inputactions** – Input System action map.
8. Refer to the source code comments for detailed explanations.

7. Extending the Controller

Adding New States/Custom Abilities

1. Create a new state script by pasting the template from **StateTemplate.txt** found in **Assets/HybridPlayerController/Scripts/PlayerController/States**.
2. Name the file and class as desired, and replace the PlayerPrefsKey string with the name of your state.
3. Override EnterState, UpdateState, FixedUpdateState, and ExitState as needed.
4. New states are automatically registered in the PlayerController.
5. To switch to your new state from another script, get a reference to the **PlayerController** script ("player" if in a state script) and call **player.TransitionToState<State>()**.

Locking/Unlocking States

Set the default lock state of a state in the **PlayerUtils** inspector on the **PlayerController** object.

Custom Animations

The default states are already hooked up to the player's animators and are triggered with code within the states, not through the animator.

Add new animations or modify existing ones in the animator attached to either **ThirdPersonBodyVisual** or **FirstPersonArms** on the **PlayerController** prefab.

8. FAQ & Troubleshooting

Q: The controller throws errors after import or collisions behave oddly.

A: Verify that all required layers are present and correctly assigned on the **Player** prefab's inspector. See section one "Requirements & Setup" and "Layer Setup" above. You should have been prompted to import the **TagManager.asset** from the package.

9. Support & Contact

For support, feature requests, or bug reports, please contact:

Email: stearnsgamedev@gmail.com

Thank you for using Hybrid Player Controller!