CameraFollow - First-Person Camera Controller Documentation

Version: 1.0.0

Author: Sakanfar Productions

Contact: sakanfarproductions@gmail.com

1. Introduction

Welcome to the CameraFollow script! This Unity C# script provides a comprehensive and customizable first-person camera solution for easy integration into your projects. It features smooth mouse look, dynamic Field of View (FOV) adjustments for aiming and running, immersive camera shake effects, and more.

This script is designed to be attached to a Camera GameObject, typically a child of your main player GameObject. It works with a `PlayerController` script to access player state information for features like running FOV and landing shakes.

2. Key Features

- **Smooth Mouse Look:** Configurable sensitivity (X and Y), Y-axis inversion, and vertical angle clamping.
- **Camera Smoothing:** Optional `SmoothDamp` for mouse movements.
- **Dynamic FOV:**
    - Normal FOV.
    - Zoomed FOV for aiming (key press activated).
    - Increased FOV when the player is running.
- **Camera Shake System:**
    - Shake during walking and running.
    - More intense shake on landing.
    - Configurable intensity, frequency, and duration.
    - Shake reduction when aiming.
- **Cursor Management:** Automatic cursor locking and hiding on start, toggled with the Escape key.
- **Automatic Reference Assignment:** Attempts to find `PlayerBody` and `PlayerController` components if not manually assigned.
- **Inspector Customization:** Most parameters are editable in the Unity Inspector.

- **Public API:** Methods to control sensitivity, rotation limits, and other settings externally.

3. Setup Instructions

1. **Create a Player:**
    - Ensure a player GameObject exists in your scene (e.g., a Capsule).
    - Your player GameObject should have your main `PlayerController` script attached.
2. **Create a Camera:**
    - Create a new Camera in your scene (GameObject -> Camera).
    - **Parent the Camera** to your player GameObject and position it for a first-person view (e.g., eye level).
3. **Add the Script:**
    - Attach the `CameraFollow.cs` script to the **Camera GameObject**.
4. **Assign References (Important):**
    - In the Inspector for the Camera GameObject, locate the "Camera Follow (Script)" component.
    - ***Player Body:*** Drag your main player GameObject (camera's parent) to this slot. Used for horizontal rotation.
    - ***Player Camera:*** Should auto-assign if the script is on the Camera. If not, drag the Camera GameObject itself here.
    - ***Player Controller:*** Drag the GameObject with your `PlayerController` script (usually the main player) to this slot. Crucial for running FOV and state-based camera shake.
5. *Initial Setup (Example)*
   PlayerParent (with PlayerController.cs)
6. └── MainCamera (with CameraFollow.cs)
7. **Configure Settings:**
    - Adjust the parameters in the Inspector (Section 4) to achieve your desired feel and gameplay.

4. Inspector Parameters

The `CameraFollow` script offers the following adjustable parameters in the Unity Inspector:Header: Mouse Look Settings

- ***Mouse Sensitivity X*** (`float`, Default: `100f`): Horizontal camera rotation speed based on mouse movement (left/right).
- ***Mouse Sensitivity Y*** (`float`, Default: `100f`): Vertical camera rotation speed based on mouse movement (up/down).
- ***Invert Mouse Y*** (`bool`, Default: `false`): Inverts vertical mouse movement if checked (up looks down).

Header: Camera Constraints

- ***Min Vertical Angle*** (`float`, Default: `-90f`): Lowest downward vertical look angle (in degrees).
- ***Max Vertical Angle*** (`float`, Default: `90f`): Highest upward vertical look angle (in degrees).

Header: Smoothing

- ***Smooth Time*** (`float`, Default: `0.1f`): Approximate time to reach the target mouse rotation. Lower values are faster; higher are smoother (only if `Enable Smoothing` is true).
- ***Enable Smoothing*** (`bool`, Default: `true`): Enables smoothing of mouse input using `Vector2.SmoothDamp`.

Header: References

- ***Player Body*** (`Transform`, Default: `null`): **Crucial:** Assign the Transform of the main player object for horizontal mouse rotation (typically the root).
- ***Player Camera*** (`Camera`, Default: `null`): Assign the Camera component this script controls (usually auto-assigned if the script is on the same GameObject).
- ***Player Controller*** (`PlayerController`, Default: `null`): **Crucial:** Assign your custom `PlayerController` script instance for player state access (`IsRunning()`, `IsGrounded()`, `IsMoving()`). **Ensure your `PlayerController` script has these public methods.**

Header: Zoom/Aim Settings

- ***Aim Key*** (`KeyCode`, Default: `Mouse1` - Right Mouse Button): Key to trigger aiming/zooming.
- ***Normal FOV*** (`float`, Default: `60f`): The camera's default Field of View (initialized from the camera's FOV at `Start()`).
- ***Zoomed FOV*** (`float`, Default: `30f`): The camera's FOV when aiming.
- ***Running FOV*** (`float`, Default: `70f`): The camera's FOV when the player is running (via `PlayerController.IsRunning()`).
- ***Zoom Speed*** (`float`, Default: `5f`): Speed of FOV transitions.
- ***Aim Sensitivity Multiplier*** (`float`, Default: `0.5f`): Multiplier applied to mouse sensitivity when aiming (values < 1 reduce sensitivity). Also reduces camera shake intensity while aiming.

Header: Camera Shake Settings

- ***Enable Camera Shake*** (`bool`, Default: `true`): Enables all camera shake effects.
- ***Walk Shake Intensity*** (`float`, Default: `0.02f`): Camera shake intensity when walking.
- ***Run Shake Intensity*** (`float`, Default: `0.05f`): Camera shake intensity when running.
- ***Shake Frequency*** (`float`, Default: `10f`): Speed/frequency of the shake motion's sine wave.

- ***Landing Shake Intensity*** (`float`, Default: `0.15f`): Maximum camera shake intensity upon landing.
- ***Landing Shake Duration*** (`float`, Default: `0.3f`): Duration of the landing shake effect (in seconds).
- ***Shake Reduction*** (`float`, Default: `2f`): Speed at which the shake offset returns to zero when inactive (higher values are faster).

5. Public Methods

These methods can be called from other scripts to control the `CameraFollow` behavior at runtime.

- `void SetMouseSensitivity(float sensitivityX, float sensitivityY)`: Updates horizontal and vertical mouse sensitivity.
    - `sensitivityX`: New horizontal sensitivity.
    - `sensitivityY`: New vertical sensitivity.
- `void SetVerticalLimits(float minAngle, float maxAngle)`: Sets the minimum and maximum vertical look angles.
    - `minAngle`: New minimum vertical angle.
    - `maxAngle`: New maximum vertical angle.
- `void ResetRotation()`: Resets the camera's local rotation and the player body's rotation to their initial states. Useful for respawning or specific events.
- `void SetInvertMouse(bool invert)`: Enables or disables Y-axis mouse inversion.
    - `invert`: `true` to invert, `false` for normal.
- `void SetSmoothing(bool enable, float time = 0.1f)`: Enables/disables mouse input smoothing and optionally sets the smooth time.
    - `enable`: `true` to enable, `false` to disable.
    - `time` (optional): The new smooth time (defaults to `0.1f`).

6. Dependencies

- `PlayerController.cs` (User-defined): This script requires an external `PlayerController` script to determine player states. Your `PlayerController` **must** implement the following public methods:
    - `public bool IsGrounded()`: Returns `true` if the player is on the ground.
    - `public bool IsMoving()`: Returns `true` if the player is currently moving.
    - `public bool IsRunning()`: Returns `true` if the player is currently sprinting/running.
- If these methods are absent or named differently in your `PlayerController`, you'll need to modify the `CameraFollow` script or adapt your `PlayerController`.

7. Troubleshooting & Frequently Asked Questions

Q: The camera fails to rotate or exhibits incorrect rotational behavior.

A: Please verify that the "Player Body" reference has been accurately assigned within the Inspector to the principal player object. Ensure that the "Player Camera" is also properly assigned. Confirm that mouse sensitivity values are non-zero.

Q: The camera unexpectedly inverts vertically.

A: Adjust the "Minimum Vertical Angle" and "Maximum Vertical Angle" parameters to constrain the rotation range and prevent excessive movement. Default values of -90 and 90 degrees are generally recommended.

Q: The Running Field of View (FOV) effect or Landing Shake effect is not functioning as expected.

A: Confirm that the "Player Controller" reference is correctly assigned. Verify that your "PlayerController" script includes the requisite public methods ("IsRunning()", "IsGrounded()", "IsMoving()") and that they are implemented correctly. Check for any error messages in the console output.

Q: The cursor is not being locked or remains visible when it should not.

A: The script is designed to lock and hide the cursor during the "Start()" function. If another script is interfering with the cursor's state, issues may arise. The "Escape" key can be utilized to toggle cursor lock functionality.

Q: The camera shake effect is either excessively intense or imperceptible.

A: Modify the "Walk Shake Intensity", "Run Shake Intensity", "Landing Shake Intensity", and "Shake Frequency" parameters within the Inspector to achieve the desired effect.

Q: The error message "Player Body not found!" is displayed.

A: The script was unable to automatically locate the "Player Body" object. Manually assign the "Player Body" within the Inspector. For optimal auto-detection, it is recommended that the camera be a child object of the player body, unless manually assigned.

Q: An error related to "PlayerController" occurs (e.g., NullReferenceException).

A: Ensure that the "Player Controller" field in the Inspector is assigned to the GameObject that contains your "PlayerController" script. Furthermore, verify that your "PlayerController" script is not null and that it implements all necessary methods.8. Contact & Support

For any technical issues, inquiries, or feature requests, please contact. We highly appreciate

your support and trust that this asset will significantly enhance your project development.-----*Thank you for utilizing CameraFollow.*