

AutoFocus LITE

DEPTH OF FIELD SYSTEM

Official Documentation

Developed by Abdelbassit Bouazzaoui

Tutorial on **Youtube**

AutoFocus LITE

Automatic Depth of Field System for Unreal Engine

Version: 1.0
Engine: Unreal Engine 5.7+
Developed with: Unreal Engine 5.8
Tested Platform: Windows
System Type: Blueprint
Distribution: Fab Marketplace

1. Introduction

1.1 What is AutoFocus LITE?

AutoFocus LITE is a lightweight, Blueprint-based automatic Depth of Field system for Unreal Engine.

The system automatically determines what the player's view is targeting and adjusts the scene's focal distance accordingly.

AutoFocus LITE is designed to operate independently from the project's existing camera architecture. Instead of requiring you to rebuild or replace your camera system, AutoFocus applies Depth of Field through a **Post Process Volume**.

This makes it suitable for a wide variety of camera configurations, including:

- Third-person games
- First-person games
- Top-down games
- Side-scrollers
- Exploration games
- Adventure games
- Custom camera systems
- Camera Animation Blueprint setups
- Hybrid camera systems

The basic workflow is:

Add AutoFocus → Assign the required references → Configure focus behavior → Play.

2. Core Features

AutoFocus LITE 1.0 includes:

Automatic Focus Detection

The system traces into the scene from the player's current view and calculates an appropriate focal distance from the detected target.

Smooth Focus Transitions

Focus distance can transition gradually rather than snapping instantly between objects.

Infinity Focus

When no valid object is detected, AutoFocus can transition toward a configurable Infinity Focus Distance instead of leaving the focus at an arbitrary previous distance.

Auto and Manual Focus Modes

The system supports automatic focusing as well as manually controlled focus behavior.

Configurable Focus Range

Minimum and maximum focus distances can be defined to prevent unwanted extreme focus values.

Configurable Trace Radius

The detection area can be adjusted according to the needs of the game.

Configurable Trace Channel

Projects can determine which collision channel AutoFocus uses rather than being locked to a specific project collision configuration.

Ignore Actor Support

Specific actors can be excluded from autofocus detection through an editable Actors to Ignore list.

Timer-Based Updates

Runtime autofocus updates are timer-driven instead of requiring the entire system to execute every frame.

This provides direct control over the balance between responsiveness and update frequency.

Editor Preview

Depth of Field behavior can be previewed while working in the editor.

Debug HUD

An optional runtime debugging interface exposes useful autofocus information while testing.

Preset System

Reusable AutoFocus configurations can be stored using AutoFocus Preset Data Assets.

Post Process Based

AutoFocus controls Depth of Field through a Post Process Volume and therefore does not require replacing the project's camera implementation.

3. Compatibility

Unreal Engine

AutoFocus LITE is intended for:

Unreal Engine 5.7 and later

The system was developed and tested primarily with **Unreal Engine 5.8**.

Future Unreal Engine versions may require additional validation.

Platform Compatibility

Tested

- Windows

Not officially validated in AutoFocus LITE 1.0

- Android
- iOS
- macOS
- Linux
- Consoles
- VR/XR

The Blueprint implementation may function on additional platforms, but platforms that have not been validated should not be considered officially supported.

4. Package Overview

The exact package organization may vary slightly between releases, but AutoFocus LITE revolves around the following components.

BP_AutoFocusDOF

The main AutoFocus actor.

This actor contains the runtime focus detection, focus calculation, smoothing, Post Process interaction, timer management, Editor Preview functionality, and related system logic.

This is the primary actor that users place inside their level.

PDA_AutoFocusPreset

The AutoFocus Preset Data Asset type.

Presets allow frequently used AutoFocus configurations to be saved and reapplied without manually recreating settings.

Debug HUD

Optional debugging UI used to inspect AutoFocus behavior during development.

Depending on the current version, debug information can include values such as:

- Detected actor
- Trace/focus distance
- Current focal distance
- Target focal distance
- Aperture
- Focal length
- Focus state

The Debug HUD is intended as a development tool and can be disabled for normal gameplay.

5. Quick Start

This section covers the minimum setup required to get AutoFocus running.

Step 1 — Add AutoFocus to the Level

Locate:

BP_AutoFocusDOF

in the Content Browser.

Drag the Blueprint into your level.

Only one primary AutoFocus actor is normally required for a single active player view.

Step 2 — Set Up the Post Process Volume

AutoFocus applies Depth of Field through a Post Process Volume.

Your level therefore requires a suitable Post Process Volume.

For most gameplay levels, enable:

Infinite Extent (Unbound)

if you want the Post Process Volume to affect the entire level.

Assign the appropriate Post Process reference to the AutoFocus actor if required by the supplied version.

Step 3 — Configure Collision Detection

AutoFocus needs to detect objects that should influence focus.

Select an appropriate **Trace Channel** in the AutoFocus settings.

You can either:

- Use a suitable existing project trace channel, or
- Create a dedicated AutoFocus trace channel.

A dedicated channel provides greater control but is not mandatory if your project's existing collision setup already provides the desired behavior.

Step 4 — Configure Focus Settings

Start with the supplied defaults or preset.

The most important settings are:

- Focus Mode
- Minimum Focus Distance
- Maximum Focus Distance
- Infinity Focus Distance
- Focus Smoothing
- Focus Interpolation Speed
- Trace Radius
- Trace Channel
- Timer Interval

Step 5 — Play

Enter Play In Editor.

Move the camera between near and distant objects.

The Depth of Field focal plane should transition according to the object detected by AutoFocus.

Your basic AutoFocus setup is now operational.

6. How AutoFocus Works

The runtime system can be understood as the following pipeline:

Player View → Trace → Hit Detection → Distance Calculation → Validation → Smoothing → Post Process DOF

At regular intervals, AutoFocus determines the current player view.

A trace is projected into the world.

If the trace detects a valid object, the system determines the distance between the relevant view/focus positions and the detected target.

That value becomes the target focus distance.

The target is constrained according to the configured focus range.

The focus value can then be interpolated toward the target according to the smoothing configuration.

Finally, the resulting focal distance is applied to the Depth of Field settings controlled through the Post Process Volume.

7. Focus Modes

Auto Focus

In Auto mode, the system automatically determines the focal distance from the player's current view.

This is the primary operating mode for normal gameplay.

Typical applications include:

- Third-person exploration
- First-person exploration
- Adventure games
- Interactive environments
- Side-scrollers
- Top-down games

Manual Focus

Manual mode allows focus behavior to be controlled without automatic target detection.

This can be useful when a specific focal distance is required or when another gameplay system needs temporary control over focus.

Typical applications include:

- Scripted sequences
- Dialogue scenes
- Inspection systems
- Cutscenes
- Special gameplay states
- Controlled focus transitions

8. Focus Distance

Minimum Focus Distance

Defines the closest usable focus distance.

This prevents AutoFocus from attempting to focus excessively close to the camera.

Increase this value if extremely close geometry causes undesirable Depth of Field behavior.

Maximum Focus Distance

Defines the maximum normal focus distance.

Objects beyond this range can be prevented from generating unnecessarily large focus values.

Infinity Focus Distance

Defines the distance used when AutoFocus does not detect a valid focus target.

Example:

The player looks toward:

- Open sky
- A distant horizon
- Empty space

The trace returns no valid target.

Instead of keeping focus locked to the last nearby object, AutoFocus transitions toward the configured **Infinity Focus Distance**.

This creates more natural behavior when moving from foreground objects toward distant scenery.

9. Trace System

The trace system determines what AutoFocus considers a valid focus target.

Trace Radius

Controls the radius of the focus detection trace.

A smaller radius provides more precise targeting.

A larger radius makes detection more forgiving.

Smaller Radius

Useful for:

- First-person games
- Precise targeting
- Narrow interaction zones

Larger Radius

Useful for:

- Third-person games
- Controller input
- Moving cameras
- Shoulder cameras
- More forgiving cinematic focus

Avoid unnecessarily large values because they can cause nearby objects to influence focus even when the player is not visually targeting them.

10. Trace Channel

AutoFocus LITE allows the trace channel to be configured according to the project.

This is important because Unreal Engine projects frequently use different collision configurations.

A project may already have channels for:

- Visibility
- Camera
- Interaction
- Weapons
- AI
- Traversal
- Custom gameplay systems

AutoFocus should not force those systems to change.

Using an Existing Channel

If an existing trace channel already produces the desired results, select that channel in AutoFocus.

This is usually the fastest setup method.

Creating a Dedicated AutoFocus Channel

For maximum control, create a custom trace channel.

Navigate to:

Project Settings → Engine → Collision

Create a new Trace Channel.

Suggested name:

AutoFocus

Then configure which object types should respond to it.

Important for Large Existing Projects

Projects containing hundreds or thousands of assets can make collision-channel migration time-consuming.

For an existing production project, do **not** automatically assume every mesh needs to be manually modified.

First determine whether an existing trace channel already provides acceptable focus detection.

A dedicated AutoFocus channel is useful when you need fine control, but it is not worth restructuring an established project's entire collision architecture unless necessary.

11. Actors to Ignore

The **Actors to Ignore** array allows specific actors to be excluded from AutoFocus detection.

Use this for actors that visually appear in the trace path but should not determine focus.

Examples can include:

- Invisible gameplay actors
- Trigger actors
- Helper geometry
- Camera-related actors
- Large background objects
- Special FX actors
- Non-focusable gameplay objects

Add the appropriate actor reference to:

Actors to Ignore

AutoFocus will exclude those actors from its normal focus detection.

12. Trace Offsets

Some games do not use a simple camera positioned directly behind the player.

Examples include:

- Over-the-shoulder cameras
- Shoulder switching
- Aim offsets
- First/third-person transitions
- Camera Animation Blueprints

- Custom camera managers
- Cinematic gameplay cameras

In these cases, the logical focus trace may need to be offset from its default position or direction.

Where available in the supplied LITE build, use the exposed trace offset controls to align autofocus detection with the actual gameplay composition.

The objective is simple:

The AutoFocus trace should represent where the player is visually looking, not merely where the pawn itself is facing.

When tuning offsets, enable AutoFocus debugging so you can verify the actual trace behavior before adjusting Depth of Field values.

13. Focus Smoothing

Real cameras generally do not change focus instantaneously.

AutoFocus can interpolate between the current focus distance and the newly detected target distance.

This produces a more natural focus pull.

Focus Smoothing

Controls whether/how smoothing behavior is applied according to the supplied configuration.

Focus Interpolation Speed

Controls how quickly the current focal distance approaches the target focal distance.

Lower values

Produce:

- Slower transitions
- More cinematic focus movement
- Softer focus changes

Higher values

Produce:

- Faster reactions
- More responsive gameplay
- Quicker transitions between targets

Extremely high values can make autofocus appear abrupt.

Extremely low values can make the system feel unresponsive.

The appropriate value depends heavily on camera speed and game type.

14. Timer-Based Updating

AutoFocus LITE uses a timer-driven update system for runtime autofocus calculations.

Rather than assuming focus logic must execute every frame, the system performs autofocus updates at a configurable interval.

This provides a useful performance/responsiveness tradeoff.

Timer Interval

Controls how frequently AutoFocus recalculates its target.

Shorter Interval

Advantages:

- Faster target detection
- More responsive autofocus

Tradeoff:

- More frequent calculations

Longer Interval

Advantages:

- Fewer autofocus calculations
- Suitable for slower gameplay

Tradeoff:

- Focus may react more slowly to rapid camera movement

Recommended Approach

Use the longest interval that still feels visually responsive for your game.

Fast first-person cameras generally require more responsive updates than slow top-down or side-scrolling cameras.

Do not reduce the interval simply because a smaller number appears more precise.

Tune it according to actual gameplay.

15. AutoFocus Presets

AutoFocus LITE includes a Data Asset based preset system.

Preset type:

PDA_AutoFocusPreset

Presets allow different AutoFocus configurations to be stored and reused.

Creating a Preset

Create an AutoFocus Preset Data Asset from the appropriate Data Asset menu.

Configure the desired settings.

Examples of useful project presets include:

- Third Person
- First Person
- Top Down
- Side Scroller
- Exploration
- Interior
- Exterior

These are examples of project-specific naming strategies rather than mandatory built-in presets.

16. Preset Functions

Apply Preset

Applies the selected AutoFocus preset to the system.

Use this when switching the actor to a predefined configuration.

Reload Current Preset

Reloads the currently assigned preset.

This is useful after editing preset values and wanting the AutoFocus actor to reflect the latest configuration.

Reset to Default Preset

Returns the system to its configured default preset.

Is Preset Active

Checks whether a specified preset is currently active.

This is useful for Blueprint logic that needs to determine the current AutoFocus configuration.

17. Editor Preview

AutoFocus LITE includes an Editor Preview workflow that allows Depth of Field behavior to be inspected during level development.

Enable:

Editor Preview

when you want to preview AutoFocus-related Depth of Field without relying exclusively on Play In Editor.

Editor Preview and Level Design

Strong Depth of Field can make level design difficult because geometry outside the focal plane may appear blurred while editing.

If this interferes with your workflow, disable Depth of Field in the editor viewport.

Use the viewport's Post Process visualization controls and disable:

Depth of Field

while performing normal level design.

Re-enable it when you want to evaluate AutoFocus.

This allows Editor Preview to remain configured without forcing you to design the level through a heavily blurred viewport.

18. Debug HUD

AutoFocus LITE includes an optional debugging interface.

Enable:

Use Debug HUD

when diagnosing or tuning AutoFocus.

Depending on the supplied build, the HUD can expose information such as:

- Current detected actor
- Hit distance
- Target focus distance
- Current focal distance
- Focal length
- Aperture
- Focus state

The Debug HUD is particularly useful when diagnosing unexpected focus behavior.

Recommended Debugging Workflow

If focus appears incorrect:

1. Enable the Debug HUD.
2. Observe which actor is being detected.
3. Verify the Trace Channel.
4. Verify collision responses.
5. Check Actors to Ignore.
6. Check minimum/maximum focus distances.
7. Check Infinity Focus Distance.
8. Verify the Post Process Volume.
9. Only then adjust smoothing.

This avoids trying to solve a collision problem by changing visual focus settings.

19. Post Process Volume Setup

AutoFocus LITE does not require replacing the player's camera system.

Depth of Field is applied through a Post Process Volume.

This separation is one of the core architectural goals of AutoFocus.

Recommended Setup

For global gameplay autofocus:

1. Add a Post Process Volume.
2. Enable **Infinite Extent (Unbound)**.
3. Configure the required Depth of Field parameters.
4. Assign/reference the volume through AutoFocus as required.
5. Allow AutoFocus to control the focal-distance component of the effect.

Existing Post Process Volumes

AutoFocus can be integrated into a project that already uses post processing.

However, be aware of overlapping volumes and competing Depth of Field settings.

If focus behaves differently in specific areas, check:

- Post Process Volume priority
- Blend weight
- Blend radius
- Infinite Extent
- Camera Post Process settings
- Other Blueprints modifying Depth of Field

Multiple systems attempting to control the same Depth of Field properties can produce inconsistent results.

20. Third-Person Games

AutoFocus is well suited to third-person cameras.

Recommended tuning areas include:

- Trace Radius
- Trace offsets
- Focus Interpolation Speed
- Minimum Focus Distance

Over-the-shoulder cameras require particular attention because the camera may not be aligned with the character's center.

Always tune autofocus based on the **actual player view**, not the character's forward vector alone.

21. Shoulder Switching

Games that allow switching between left and right shoulders can change the relationship between the camera and the character.

If the trace no longer corresponds to the visual center after switching shoulders, adjust or drive the available trace offset configuration accordingly.

Use debugging during both shoulder states.

Verify:

Left Shoulder → Center Target → Right Shoulder

The focus system should continue detecting the visually intended target.

22. First-Person Games

First-person games generally benefit from:

- Smaller trace radius
- Faster focus interpolation
- Responsive timer intervals
- Careful minimum focus distance

Very close objects such as:

- Weapons
- Hands
- First-person body meshes

may need special handling depending on the game's visual design.

If these objects should not influence world autofocus, configure collision or ignore behavior appropriately.

23. Top-Down Games

Top-down cameras typically operate farther from the environment.

Consider:

- Increasing the useful focus range
- Adjusting Infinity Focus Distance
- Using a suitable trace radius
- Using slower focus transitions where appropriate
- Using a slightly longer timer interval if gameplay allows it

AutoFocus LITE has been tested with Unreal Engine's standard Top-Down template as part of its development validation.

24. Side-Scroller Games

Side-scrolling games usually have more predictable camera movement than free-camera third-person games.

This can make autofocus easier to tune.

Consider:

- Limiting the focus range
- Using moderate smoothing
- Ignoring foreground decorative geometry where necessary
- Adjusting collision responses for background elements

AutoFocus LITE has been tested in a 3D side-scrolling game environment during development.

25. Custom Camera Systems

AutoFocus is specifically designed to avoid requiring users to replace an existing camera architecture.

This is useful for projects using systems such as:

- Camera Managers
- Camera Animation Blueprints
- Camera rigs
- First/third-person switching
- Custom camera interpolation
- Gameplay-specific camera controllers

The important requirement is that AutoFocus receives a meaningful player view from which focus detection can be performed.

If your custom camera significantly changes the visual center, configure the available trace offsets accordingly.

26. Performance Guidelines

AutoFocus LITE is intended to remain lightweight.

However, every runtime system has a cost.

The primary variables affecting AutoFocus workload are:

- Update frequency
- Trace complexity
- Collision configuration
- Number of active AutoFocus systems
- Debug functionality

Recommended Practices

Use sensible timer intervals

Do not update autofocus more frequently than the game visually requires.

Disable debugging for normal gameplay

Debug visualization and HUD functionality should primarily be used during development.

Keep collision configuration intentional

Do not make every object respond to AutoFocus unless necessary.

Ignore irrelevant actors

Exclude actors that should never become focus targets.

Avoid unnecessary duplicate systems

A single-player game with one active view normally requires one active AutoFocus controller.

27. Troubleshooting

AutoFocus Does Nothing

Possible causes

- Post Process Volume is missing.
- Post Process reference is invalid.
- Depth of Field is disabled.
- AutoFocus is not running.
- Required settings are not enabled.

Check

Verify the Post Process Volume first, then confirm that AutoFocus is updating.

Enable the Debug HUD if necessary.

Focus Never Detects Objects

Possible causes

- Incorrect Trace Channel.
- Target objects ignore the selected channel.
- Collision is disabled.
- Trace configuration is incorrect.

Fix

Enable debugging and verify that the trace reaches the intended object.

Then inspect that object's collision response.

Focus Detects the Wrong Object

Possible causes

- Trace Radius is too large.
- Foreground geometry is blocking the trace.
- An unwanted actor responds to the Trace Channel.
- Trace offset does not match the camera composition.

Fix

Reduce Trace Radius, adjust collision responses, add unwanted actors to Actors to Ignore, or correct the trace offset.

Focus Snaps Too Quickly

Cause

Focus interpolation is too fast.

Fix

Reduce Focus Interpolation Speed or increase smoothing according to the desired visual style.

Focus Feels Too Slow

Possible causes

- Focus Interpolation Speed is too low.
- Timer Interval is too long.

Fix

Increase interpolation speed and/or reduce the Timer Interval.

Looking at the Sky Keeps the Previous Object in Focus

Check

Verify the Infinity Focus configuration.

When no valid target is detected, the system should transition toward the configured Infinity Focus Distance.

Everything is Blurry

Possible causes

- Incorrect focus distance.
- Extreme aperture/Depth of Field configuration.
- Post Process conflict.
- Focus distance outside the expected gameplay range.

Fix

Use the Debug HUD to inspect the calculated focus distance before changing visual settings.

Focus Changes When Entering Another Area

Possible cause

Another Post Process Volume is overriding or blending with the AutoFocus Post Process settings.

Fix

Inspect volume priority, blend weight, blend radius, and Infinite Extent settings.

Editor Preview Makes Level Design Difficult

Disable Depth of Field in the editor viewport while designing the environment.

Re-enable it when testing AutoFocus.

Custom Trace Channel Requires Editing Too Many Assets

Do not automatically modify every asset.

First evaluate whether:

- An existing channel can be used.
- Default collision behavior can be configured appropriately.
- Only important focusable assets require changes.

Large production projects should integrate AutoFocus into their existing collision architecture rather than unnecessarily restructuring thousands of assets.

28. Known Limitations — LITE 1.0

AutoFocus LITE is intentionally focused on the core automatic Depth of Field workflow.

The following should not be assumed to be part of LITE 1.0 unless explicitly stated in the current product listing:

- Advanced cinematic rack-focus sequencing
- Focus regions/volumes
- Full multi-camera management
- Multiplayer-specific focus management
- Virtual Camera integration
- C++ plugin implementation
- Mobile-specific performance profiles
- Per-level automatic configuration management
- Advanced runtime preset orchestration
- Cinematic focus timelines

These may belong to future updates or other editions of the AutoFocus system.

29. Best Practices

For the most predictable results:

1. Configure AutoFocus only after your basic camera behavior is working.
2. Verify trace behavior before tuning Depth of Field.
3. Use collision filtering intentionally.
4. Keep the Trace Radius as small as gameplay allows.
5. Use Infinity Focus for open environments.
6. Tune focus interpolation while actually moving the camera.
7. Test both near-to-far and far-to-near transitions.
8. Test rapidly moving across multiple objects.

9. Test empty space and sky.
10. Test the system with the final gameplay camera.
11. Disable development debugging before shipping.
12. Validate performance on the actual target hardware.

30. Recommended Testing Sequence

Before considering your integration complete, test the following:

Near Object → Medium Object → Far Object → Sky/Empty Space

Then test:

Far Object → Near Object

Then:

Rapid Camera Pan Across Multiple Objects

Then:

Slow Camera Movement

Finally test any game-specific camera states:

- Aiming
- Shoulder switching
- Sprint camera
- Crouch camera
- First-person mode
- Third-person mode
- Dialogue camera
- Interior camera
- Exterior camera

This provides a much better representation of real gameplay than testing AutoFocus against a single static object.

31. FAQ

Does AutoFocus LITE replace my camera system?

No.

AutoFocus LITE is designed to work alongside the existing camera architecture and applies Depth of Field through a Post Process Volume.

Do I need to modify my Character Blueprint?

For the standard setup, AutoFocus is designed to minimize modifications to existing gameplay systems.

Projects using highly customized camera behavior may require configuration to ensure the focus trace corresponds to the actual player view.

Does it work with third-person games?

Yes.

Third-person gameplay was part of development testing.

Does it work with first-person cameras?

The system architecture supports first-person view-based autofocus. First-person projects may require additional tuning for close weapon/body geometry.

Does it work with top-down games?

Yes.

AutoFocus LITE has been tested with Unreal Engine's Top-Down template.

Does it work with side-scrollers?

Yes.

The system has been tested in a 3D side-scrolling game environment.

Can I use my own Trace Channel?

Yes.

The Trace Channel is configurable.

Do I have to create an AutoFocus Trace Channel?

No.

A dedicated channel provides additional control, but you can use an appropriate existing channel if it works with your project's collision architecture.

Can I ignore specific actors?

Yes.

Use the Actors to Ignore configuration.

What happens when the player looks at the sky?

When no valid focus target is detected, AutoFocus can use the configured Infinity Focus Distance.

Can I change how quickly focus transitions?

Yes.

Use the focus smoothing/interpolation settings.

Does AutoFocus run every frame?

Runtime focus updates use a configurable timer-based approach.

This allows you to balance responsiveness against update frequency.

Can I preview AutoFocus in the editor?

Yes.

AutoFocus LITE includes Editor Preview functionality.

Can I temporarily disable Depth of Field while designing my level?

Yes.

Disable Depth of Field through the editor viewport's Post Process visualization controls and re-enable it when testing the system.

Does AutoFocus include presets?

Yes.

AutoFocus LITE includes a Data Asset based preset workflow.

Is AutoFocus LITE multiplayer ready?

LITE 1.0 should not be assumed to provide dedicated multiplayer-specific management.

If using the system in a multiplayer project, validate ownership and local-view behavior according to your project's architecture.

Does AutoFocus support mobile?

Mobile platforms are not officially validated for AutoFocus LITE 1.0.

Do not assume official Android or iOS support unless the current product listing explicitly states otherwise.

Which platforms have been tested?

The current validated platform is:

Windows

Which Unreal Engine versions are supported?

AutoFocus LITE targets **Unreal Engine 5.7+** and was developed primarily with **Unreal Engine 5.8**.

Always check the current Fab listing for the latest supported-engine information.

32. Before Shipping Your Game

Disable development-only functionality that you do not want exposed to players.

Recommended final check:

- AutoFocus enabled
- Correct Focus Mode selected
- Correct Trace Channel selected
- Collision responses validated
- Actors to Ignore configured
- Minimum Focus Distance validated
- Maximum Focus Distance validated
- Infinity Focus Distance tested
- Smoothing tested
- Timer Interval optimized
- Post Process Volume assigned
- Post Process conflicts checked
- Debug HUD disabled
- Editor-only preview functionality not relied upon at runtime
- Final gameplay camera tested
- Packaged build tested
- Target hardware tested

Do not rely exclusively on Play In Editor testing.

Always test AutoFocus in a packaged build before shipping your project.

33. Support

Before requesting support, please prepare the following information:

- Unreal Engine version
- AutoFocus LITE version
- Camera type
- Focus Mode
- Trace Channel
- Relevant collision settings
- Whether Editor Preview works
- Whether runtime AutoFocus works
- Screenshot or video of the issue
- Debug HUD information if applicable
- Steps required to reproduce the problem

Providing this information significantly reduces troubleshooting time.

Support Links

Documentation:
Documentation

Community / Discord:
Discord

Support Email:
aabdelbassit@outlook.com

Fab Product Page:
[Fab Product URL]

---

34. Bug Reports

When reporting a bug, include:

Engine Version:
Example: Unreal Engine 5.8

AutoFocus Version:
Example: AutoFocus LITE 1.0

Platform:
Example: Windows

Project Type:
Example: Third Person

Camera System:
Default / Custom / Camera Animation Blueprint / Other

Description:
Clearly describe the problem.

Expected Result:
Describe what you expected AutoFocus to do.

Actual Result:
Describe what happened instead.

Reproduction Steps:
Provide the shortest sequence that consistently reproduces the issue.

Screenshots and short videos are strongly recommended for visual focus problems.

35. Updating AutoFocus LITE

Before updating AutoFocus in an existing production project:

1. Back up or commit your project.
2. Record any custom AutoFocus settings.
3. Review the release notes.
4. Update the asset.
5. Verify presets.
6. Verify Post Process references.
7. Verify collision configuration.
8. Test Editor Preview.
9. Test runtime autofocus.
10. Test a packaged build.

Never perform major asset updates on the only copy of a production project.

Use source control whenever possible.

36. Version Information

AutoFocus LITE 1.0

Initial public LITE release.

Core functionality includes:

- Automatic focus detection
- Post Process based Depth of Field
- Auto/Manual focus workflow
- Configurable focus range
- Infinity Focus
- Focus smoothing
- Configurable Trace Radius
- Configurable Trace Channel
- Actor ignore support
- Timer-based runtime updates
- Editor Preview
- Debug HUD
- Data Asset presets
- Preset management functions
- Support for varied gameplay camera architectures

37. AutoFocus LITE Philosophy

AutoFocus LITE is designed around three principles:

Non-Invasive

The system should work alongside an existing camera architecture rather than forcing developers to rebuild it.

Configurable

Different games require different focus behavior, collision setups, camera positions, and response speeds.

AutoFocus exposes the important controls instead of assuming a single configuration.

Lightweight

Automatic Depth of Field should improve presentation without becoming an unnecessarily large gameplay framework.

The LITE edition therefore focuses on the core functionality required to achieve reliable automatic Depth of Field while remaining straightforward to integrate and tune.

AutoFocus LITE

Automatic focus. Existing camera. Minimal integration.

Developed for Unreal Engine.

For current compatibility, updates, tutorials, and support, refer to the official AutoFocus LITE product page and documentation.

Fab: AutoFocus LITE
Tutorial: **Youtube**
Documentation: Full Documentation
Discord: Discord