COLLISION COMMANDER

User Guide

Unreal Engine Collision Management Plugin

Plugin	Collision Commander v1.0
Engine	Unreal Engine 5.3 – 5.7
Platforms	Windows · macOS · Linux (Editor)
Publisher	Paracosm · paracosm.gg
Support	support@paracosm.gg

Table of Contents

SECTION 1

Introduction & Installation

Collision Commander is a single dockable editor panel for Unreal Engine that gives developers a complete, always-live view of how every collision preset in their project interacts with every other.

The core is a color-coded N×N matrix showing every preset pair resolved to BLOCK, OVERLAP, or IGNORE. From the same panel, developers can run focused actor comparisons, what-if experiments, and validation sweeps — all without leaving the plugin or manually cross-referencing Project Settings.

Installation

- Install from your Fab.com library via the Epic Games Launcher.
- In the editor: **Edit** → **Plugins** → search "Collision Commander" → Enable → Restart Editor.
- Unreal Engine requires at least one C++ module in a project before it can compile any plugin. C++ projects satisfy this automatically. Blueprint-only projects need a one-time setup — see below.

Blueprint-Only Projects: One-Time C++ Setup

Unreal Engine plugins are compiled code. Even if your game contains no C++, the engine still needs a C++ module to exist in your project before it can build any plugin. This is an engine requirement, not something specific to Collision Commander.

You do not need to write any C++ code. The setup takes about two minutes and you will never need to touch it again.

Step	Action
1	In the Unreal Editor, open **Tools** → **New C++ Class**.
2	Select **None** (the empty class option at the top of the list) and click Next.
3	Leave the class name as the default (e.g. MyObject) and click **Create Class**.
4	The editor will prompt you to compile. Click **Yes** and wait for the build to finish (typically under a minute).
5	After the editor restarts, enable Collision Commander via **Edit** → **Plugins** as normal.

> ■ **What this actually does**
>
> Creating an empty C++ class tells the engine to generate the project's C++ infrastructure (a Source/ folder, a .Build.cs file, and a module entry in the .uproject). Collision Commander then compiles into that infrastructure. The empty class itself does nothing — it just satisfies the engine's requirement. Your game logic remains entirely in Blueprint.

Opening the Panel

Go to **Window** → **Collision Commander** in the main menu bar, or click the Collision Commander button in the Level Editor toolbar. The panel is a dockable Nomad Tab — drag it to any position in your editor layout.

System Requirements

Requirement	Minimum	Recommended
Unreal Engine	5.3	5.7
Project Type	C++ or Blueprint-only	Any
OS	Windows 10 / macOS 12 / Ubuntu 20.04	Windows 11
Disk Space	~15 MB (binary)	~15 MB

SECTION 2

Understanding Collision in Unreal Engine

Before using the matrix, it helps to understand how Unreal resolves collision interactions. Collision Commander surfaces these rules visually — but understanding them helps you interpret the results correctly.

Query vs Physics Responses

Every collision preset stores two independent sets of channel responses:

- **Query** — governs line traces, shape sweeps, and overlap events. Does not affect physical simulation. Used for hit detection, trigger volumes, line-of-sight, and interaction checks.
- **Physics** — governs rigid-body simulation. Whether objects physically push against each other, land on surfaces, or pass through geometry.

A preset can Block a channel for Physics (objects collide) while Ignoring it for Query (traces pass through). This is intentional UE design. The Query/Physics toggle at the top right of most Collision Commander panels controls which set you are viewing.

The Pairwise Resolution Rule

UE determines the interaction between two presets using the minimum of both actors' responses:

Resolution Formula

```
result = min(A's response to B's ObjectType, B's response to A's ObjectType)
Where: Ignore (0) < Overlap (1) < Block (2)
```

The key consequence: if either side is Ignore, the result is always Ignore — regardless of what the other side specifies. A Block on one side is silently overridden when the other side is Ignore.

Collision Commander computes this rule twice for every preset pair — once for Query responses, once for Physics — and stores both results independently.

Overlap Events

A resolved result of OVERLAP is a necessary but not sufficient condition for a BeginOverlap event to fire at runtime. Both actors must additionally have **Generate Overlap Events** enabled on their collision component. The Actor Compare panel flags this condition with an inline warning.

Matrix Cell Colors

Color	Meaning
BLOCK (red)	Physical or query contact occurs between these two presets.
OVERLAP (gold)	Overlap event can fire (subject to Generate Overlap Events flags).
IGNORE (dark grey)	No interaction — neither physical nor query contact.

Color	Meaning
Diagonal (dark blue)	Same preset compared against itself.

PANEL 01

Collision Matrix

The Matrix panel is the primary view — an N×N color-coded grid showing how every preset in your project interacts with every other. It reads directly from UCollisionProfile and is always current.

Layout

- Rows = Actor A preset. Columns = Actor B preset.
- Cell color reflects the resolved interaction (Block, Overlap, Ignore, or same-preset diagonal).
- A thick visual separator divides UE's 14 built-in presets (top-left) from your custom presets (bottom-right).
- Column and row headers show the preset name. Hovering a truncated name shows the full name in a tooltip.

Toolbar Controls

Control	Description
Query / Physics toggle	Switches between Query response view (traces, overlaps) and Physics response view (simulation). Affects the entire panel — Matrix and hover tooltips both update.
Refresh button	Re-reads UCollisionProfile and rebuilds the matrix. Also triggers automatically when Project Settings change via the settings-changed delegate.

Hover Tooltip

Hovering any non-diagonal cell shows all three values:

Tooltip Format

```
PresetA → PresetB:
PresetB → PresetA:
Resolved:
```

Missing channel entries (channel defined in one preset but not the other) show as **?** in the tooltip and resolve to Ignore safely — no crash.

Clicking a Cell

Left-clicking any non-diagonal cell switches to the Actor Compare panel and pre-populates it: the row preset becomes the Primary, and the column preset becomes the first comparison entry. A hand cursor on non-diagonal cells provides visual affordance for this interaction.

Scrolling

Large preset lists create a matrix larger than the panel. Scroll vertically with the mouse wheel. Scroll horizontally by dragging the horizontal scrollbar. The header row and column remain fixed as you scroll.

■ **Tip**

On projects with many custom presets, use the built-in/custom separator as a visual anchor. Most collision bugs involve mismatches between a custom preset and a UE built-in — these cells are in the bottom-left and top-right quadrants of the matrix.

PANEL 02

Actor Compare

Actor Compare lets you pick a **primary** actor and compare it against a list of other actors all in one view. For each actor you add to the compare list, the panel evaluates every collision-capable component on that actor and shows how it resolves against the primary — so you can immediately see the full collision surface without jumping between presets.

A practical example: set your **Projectile** actor as the primary, then add a range of enemy and player actor types to the compare list. You'll see in one glance exactly what the projectile blocks, overlaps, or ignores across your entire cast of characters — and catch any mismatches before they become bugs in play.

Adding Actors to the Compare List

- **Click and drag** any actor from the level viewport or Content Browser directly into the compare list panel.
- **Click the import button** (the folder icon) to browse and select an actor asset from the Content Browser.
- Both methods support adding multiple actors — build up the comparison list as large as you need.
- Remove any entry using the × button on its row.
- Press Refresh to re-evaluate if Project Settings have changed.

Reading a Result Row

Each row in the comparison list shows:

Column	Description
Preset Name	The comparison preset being evaluated against the primary.
Result Badge	BLOCK (red), OVERLAP (gold), or IGNORE (grey) — the resolved interaction for the current Query/Physics mode.
A→B Response	The primary preset's raw response to the comparison preset's ObjectType channel.
B→A Response	The comparison preset's raw response to the primary preset's ObjectType channel.
■ Overlap Warning	Shown when the resolved result is OVERLAP. Reminds you that Generate Overlap Events must be enabled on both actors' relevant collision components.

The Overlap Event Warning in Detail

A resolved OVERLAP means the UE pairwise rule evaluates to Overlap — but the BeginOverlap event will only actually fire at runtime if both actors have **Generate Overlap Events** enabled on their relevant collision component.

The warning is informational, not an error. It appears whenever the resolved result is OVERLAP to remind you to verify this flag. It does not indicate a bug — it indicates a runtime condition to confirm.

■ Fastest Way to Start an Investigation

Click any cell in the Matrix panel. The two presets are automatically loaded into Actor Compare — the row preset as Primary, the column preset as the first compare entry. No manual setup required. From there, drag in more actors from the level to build out your comparison list instantly.

PANEL 03

Experiment Mode

Experiment Mode is a what-if sandbox. Temporarily override any channel response value and see the Collision Matrix update live to reflect how those changes would affect all interactions — without touching your actual Project Settings.

Active Experiment Indicator

When any override is active, a prominent banner in the Experiment panel reads **"Experiment Mode Active — matrix does not reflect real project state."** This visual indicator prevents confusion between experimental and real values. The Matrix panel also reflects the experiment state.

Setting Overrides

- Each channel has separate Query and Physics override dropdowns.
- Options for each: Ignore, Overlap, or Block.
- The Matrix updates immediately as you change a dropdown — no evaluate button.
- Cells that differ from the real project baseline are highlighted with a distinct border.

Committing an Experiment

When satisfied with an experiment, press **Commit to Project Settings**. This writes your overrides to the project's collision configuration — the same change you would make manually in **Project Settings** → **Collision**.

Commit requires an explicit button press. There is no auto-commit. Closing the panel without committing discards all overrides silently.

Resetting

Press **Reset** to clear all overrides instantly. The matrix returns to the real project baseline. This cannot be undone, but since the overrides were never committed, your project state is unchanged.

> ■ **Important**
>
> Committing an experiment modifies your project's collision configuration — the same data modified by Project Settings → Collision. Always run a refresh and review the matrix after committing to confirm the result is what you intended.

PANEL 04 Validation

The Validation panel runs ten built-in rules against your collision configuration and surfaces common misconfigurations before they become bugs. Results are grouped by severity and accessible from the main toolbar.

Validation runs automatically on every matrix refresh and after every experiment commit.

Severity Levels

Severity	Meaning
Error	Configuration that will likely cause bugs or is known to cause incorrect behavior. Fix before shipping.
Warning	Potential misconfiguration worth reviewing. May be intentional in some projects.
Info	Non-critical observation such as unused channels. Worth knowing, not urgent.

All 10 Validation Rules

Rule ID	Sev.	Trigger & Suggested Fix
CS_CHAN_BUDGET_WARN	Warn	14+ of 18 custom channel slots in use. Audit unused channels before adding new ones.
CS_CHAN_BUDGET_CRIT	Error	17+ custom channel slots used. Consolidate channels — you are near the 18-slot limit.
CS_CHAN_UNUSED	Info	Custom channel defined but no preset responds non-Ignore. Remove or mark as reserved.
CS_PRESET_DUPLICATE	Warn	Two or more presets with identical ObjectType and response arrays. Consolidate or document why both are needed.
CS_PRESET_ALL_IGNORE	Warn	A preset responds Ignore to every channel. Likely misconfiguration — confirm intent or use NoCollision instead.
CS_STATIC_IGNORED	Warn	Dynamic ObjectType preset ignores WorldStatic. Dynamic actors that ignore WorldStatic will fall through floors.
CS_OVERLAP_MISMATCH	Info	A wants to Block B, but B ignores A's channel (result: Ignore). Review intent — the Block is silently overridden.
CS_NAME_CONFLICT	Error	Custom preset name matches a UE built-in name exactly. Rename to avoid lookup ambiguity.
CS_OVERLAP_EVENT	Warn	Two presets resolve to Overlap but Generate Overlap Events may not be enabled on both actors' components.
CS_VISIBILITY_BLOCK	Info	A preset blocks the Visibility channel. Confirm intent — affects AI sight, shooting, and camera traces.

■ **Tip**

Individual rules can be toggled on or off in **Edit** → **Project Settings** → **Plugins** → **Collision Commander**. Useful if a rule produces false positives for your project's intentional configuration.

SECTION 7

Support & Contact

For help, bug reports, and feature requests, email **support@paracosm.gg**. Using the subject line format below helps us triage and respond faster.

Subject Line Format

Subject Prefix	Use When
Help	Something isn't working as expected or you need usage guidance.
Bug Report	A reproducible crash, incorrect behavior, or data error.
Feature Request	A capability you'd like to see added or changed.
License	Studio licensing or billing questions.

What to Include in a Bug Report

Including the following information helps us reproduce the issue quickly and reduces back-and-forth. Copy the fields below into your email body:

Field	What to Write
Unreal Engine version	e.g. 5.4.4, 5.7.0 — visible in the editor title bar or the About dialog.
Plugin version	e.g. 1.0.0 — visible in Edit → Plugins → Collision Commander.
Operating system	e.g. Windows 11, macOS 14.4, Ubuntu 22.04.
Project type	C++ project or Blueprint-only (with C++ stub).
Steps to reproduce	A numbered list of the exact steps that trigger the issue, starting from a fresh editor launch.
Expected behavior	What you expected to happen.
Actual behavior	What happened instead. Include any error messages or log output.
Collision setup	Relevant preset names, channel names, or a screenshot of the matrix if the issue relates to incorrect values.
Crash log (if applicable)	Output log text or a .dmp file from Saved/Crashes/. Paste the relevant section or attach the file.

■ **Tip**

The fastest way to capture your setup: in the Collision Commander panel, take a screenshot of the Matrix or Actor Compare view showing the unexpected result and attach it to the email. A picture of the wrong interaction is worth a thousand words.

Quick Troubleshooting

- **Unexpected IGNORE results:** Check the Query/Physics toggle. A channel that Blocks for Physics may Ignore for Query. Use the hover tooltip to see both contributing responses.
- **Overlap events not firing:** A resolved OVERLAP result is not enough. Both actors must have Generate Overlap Events enabled on their collision component. The Actor Compare panel flags this with an inline ■ warning.
- **Matrix looks different after restarting:** Project Settings may have been modified outside of Collision Commander. Press Refresh to resync.
- **Experiment Mode accidentally left active:** If the "Experiment Mode Active" banner is showing, press Reset to return to real project state.
- **Plugin won't compile on Blueprint-only project:** You need a minimal C++ stub module. See the Fab.com listing or email support for the stub setup guide.

Links

Plugin page and latest documentation:	paracosm.gg/collisioncommander
Fab.com listing:	fab.com (search "Collision Commander")
Support email:	support@paracosm.gg