

Developer  
/ Documentation  
/ Unreal Engine ▾  
/ Unreal Engine 5.4 Documentation  
/ Programming and Scripting  
/ Development Setup  
/ Live Coding

# Live Coding

Recompile and patch your game's binaries at runtime.



**Unreal Engine (UE)** supports **Live Coding** using an integration of [Live++](#). Live Coding is a system that can rebuild your application's C++ code and patch its binaries while the engine is running. This functionality is available during the following scenarios:

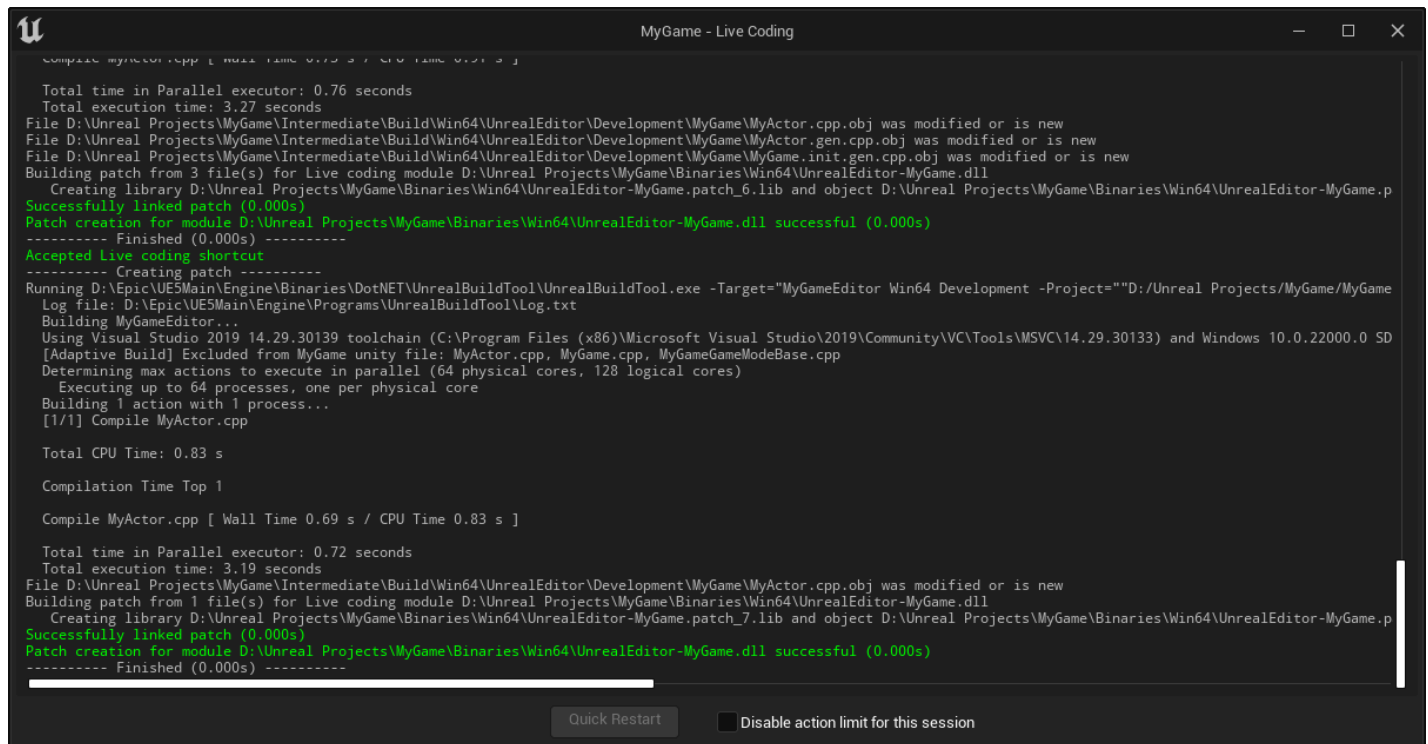
- Editing your application in Unreal Editor.
- Running your application with Play In Editor (PIE).
- Running a packaged Desktop build of your application attached to the editor for debugging.

This means that without interrupting playtesting sessions or work in the editor itself, you can make changes to C++ classes, compile, and immediately see those changes take effect. This provides significant benefits for iterative development when using C++ runtime logic, such as gameplay code or frontend user interactions. Although the **Hot Reload** system is still available as an alternative, Live Coding is significantly faster and more flexible.

## How to Use Live Coding

Live Coding is enabled by default for all new Unreal Engine installations. When you open your IDE the Live Coding Console will start automatically but remain hidden. If the console is

hidden, it will open when you initiate a Live Coding build.



```
u
MyGame - Live Coding

Compile MyActor.cpp [ Wall Time 0.72 s / CPU Time 0.83 s ]

Total time in Parallel executor: 0.76 seconds
Total execution time: 3.27 seconds
File D:\Unreal Projects\MyGame\Intermediate\Build\Win64\UnrealEditor\Development\MyGame\MyActor.cpp.obj was modified or is new
File D:\Unreal Projects\MyGame\Intermediate\Build\Win64\UnrealEditor\Development\MyGame\MyActor.gen.cpp.obj was modified or is new
File D:\Unreal Projects\MyGame\Intermediate\Build\Win64\UnrealEditor\Development\MyGame\MyGame.init.gen.cpp.obj was modified or is new
Building patch from 3 file(s) for Live coding module D:\Unreal Projects\MyGame\Binaries\Win64\UnrealEditor-MyGame.dll
Creating library D:\Unreal Projects\MyGame\Binaries\Win64\UnrealEditor-MyGame.patch_6.lib and object D:\Unreal Projects\MyGame\Binaries\Win64\UnrealEditor-MyGame.p
Successfully linked patch (0.000s)
Patch creation for module D:\Unreal Projects\MyGame\Binaries\Win64\UnrealEditor-MyGame.dll successful (0.000s)
----- Finished (0.000s) -----
Accepted Live coding shortcut
----- Creating patch -----
Running D:\Epic\UE5Main\Engine\Binaries\DotNET\UnrealBuildTool\UnrealBuildTool.exe -Target="MyGameEditor Win64 Development -Project=""D:\Unreal Projects\MyGame\MyGame
Log file: D:\Epic\UE5Main\Engine\Programs\UnrealBuildTool\Log.txt
Building MyGameEditor...
Using Visual Studio 2019 14.29.30139 toolchain (C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.29.30133) and Windows 10.0.22000.0 SD
[Adaptive Build] Excluded from MyGame unity file: MyActor.cpp, MyGame.cpp, MyGameGameModeBase.cpp
Determining max actions to execute in parallel (64 physical cores, 128 logical cores)
Executing up to 64 processes, one per physical core
Building 1 action with 1 process...
[1/1] Compile MyActor.cpp

Total CPU Time: 0.83 s

Compilation Time Top 1

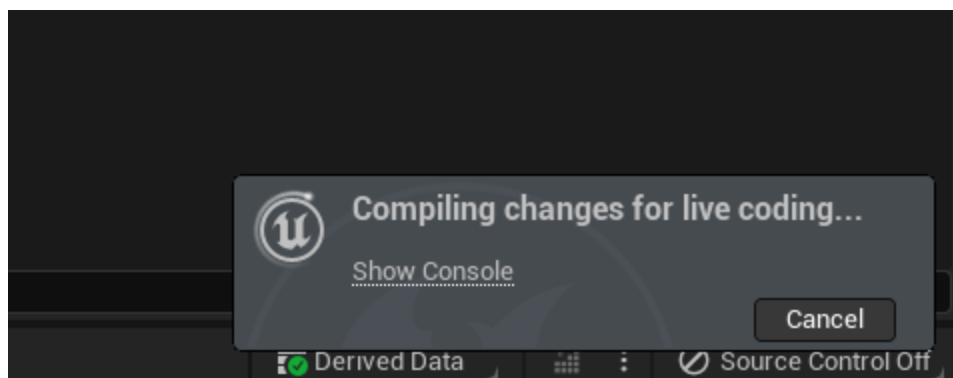
Compile MyActor.cpp [ Wall Time 0.69 s / CPU Time 0.83 s ]

Total time in Parallel executor: 0.72 seconds
Total execution time: 3.19 seconds
File D:\Unreal Projects\MyGame\Intermediate\Build\Win64\UnrealEditor\Development\MyGame\MyActor.cpp.obj was modified or is new
Building patch from 1 file(s) for Live coding module D:\Unreal Projects\MyGame\Binaries\Win64\UnrealEditor-MyGame.dll
Creating library D:\Unreal Projects\MyGame\Binaries\Win64\UnrealEditor-MyGame.patch_7.lib and object D:\Unreal Projects\MyGame\Binaries\Win64\UnrealEditor-MyGame.p
Successfully linked patch (0.000s)
Patch creation for module D:\Unreal Projects\MyGame\Binaries\Win64\UnrealEditor-MyGame.dll successful (0.000s)
----- Finished (0.000s) -----

Quick Restart Disable action limit for this session
```

The Live Coding console provides an output log for compilation status. This is separate from the standard **Output Log**, and only shows Live Coding build information.

To start a build, press **CTRL+ALT+F11** on your keyboard while using either your IDE or Unreal Engine. A notification will appear in the lower-right corner of the screen to show the status of your build. You can continue working in the editor or testing your project uninterrupted while the build runs.



If your build succeeds, you will immediately see changes according to your code.

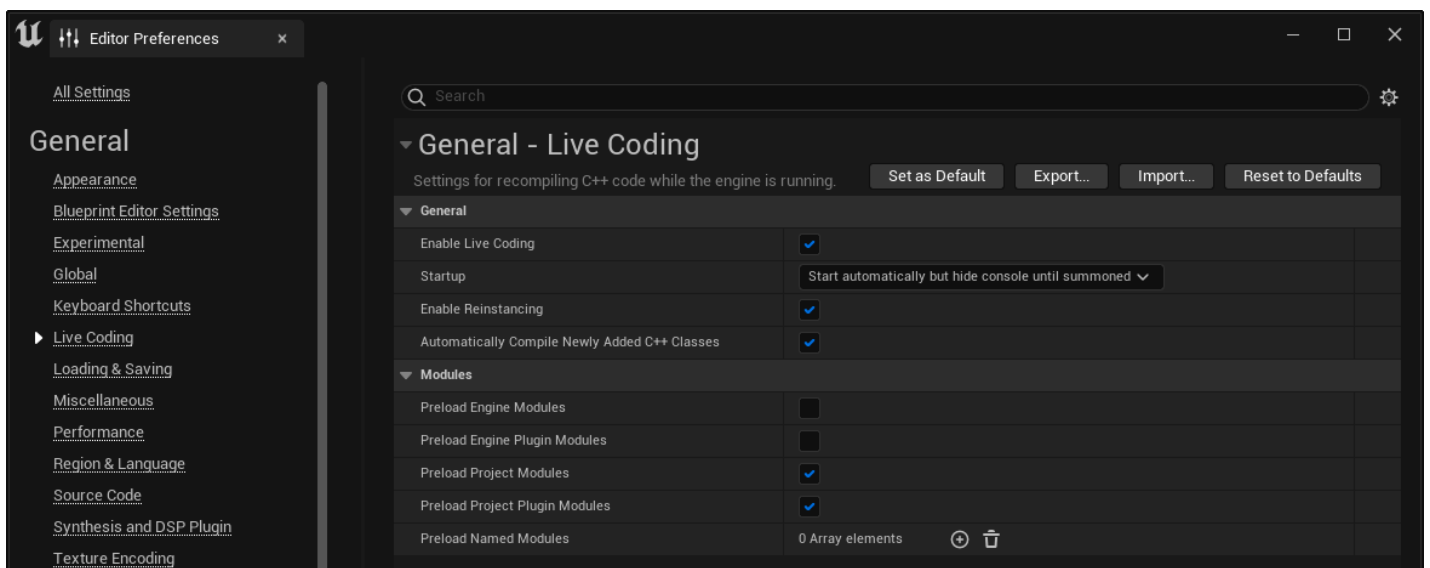
Live Coding is available when Unreal Editor is open, when using PIE, and when attached to a packaged build in your native desktop environment. It is not available when launching on consoles and mobile devices.

# Changing Default Values for Variables With Live Coding

When changing default values for variables, values set in the constructor implemented in the `.cpp` file will not update in existing instances of objects. However, if you change them in your `.h` file, you will see the change take place.

## Configuration

You can find the settings for Live Coding in **Editor Preferences > General > Live Coding**.



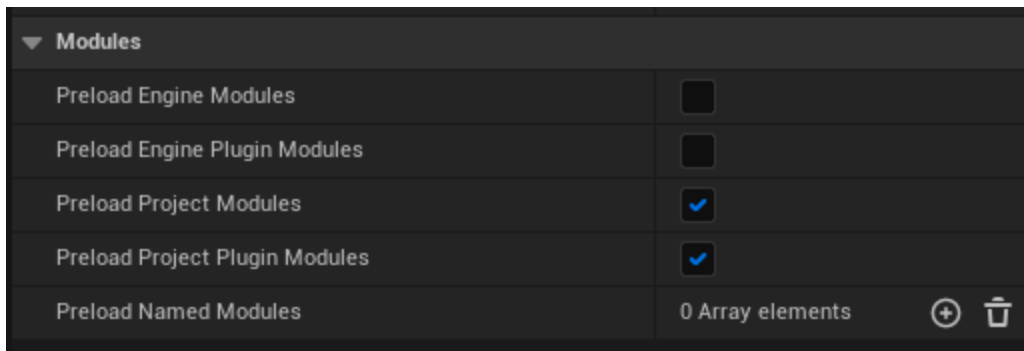
## General Live Coding Settings

General	
Enable Live Coding	<input checked="" type="checkbox"/>
Startup	Start automatically but hide console until summoned ▼
Enable Reinstancing	<input checked="" type="checkbox"/>
Automatically Compile Newly Added C++ Classes	<input checked="" type="checkbox"/>

The **General** settings control how Live Coding will behave in your development environment.

Setting	Description
Enable Live Coding	When enabled, Live Coding is Unreal Editor's compilation method. If disabled, Unreal Editor falls back to using Hot Reload instead.
Startup	Controls how Live Coding behaves when Unreal Editor starts, including whether the console is active or visible.
Enable Reinstancing	Controls whether or not Live Coding uses Object Reinstancing. When enabled, instances of objects are replaced so that large-scale changes in their code can take effect. Disabling this setting is not recommended, as these kinds of changes can be unstable without it. See Object Reinstancing With Live Coding for more details.
Automatically Compile Newly Added C++ Classes	When enabled, adding C++ classes will automatically prompt Live Coding to compile them and add them to your application.

## Modules



The **Modules** section designates which modules should be pre-loaded for Live Coding when you run the editor or your application. This makes Live Coding's performance when iterating on these modules faster, but the more modules that you set to pre-load, the longer it will take for the editor or your game to start.

Option	Description
Engine Modules	Modules that are part of Unreal Engine's source code.
Engine Plugin Modules	Modules associated with plugins in your Engine's install directory, under Engine/Plugins.
Project Modules	Modules that are part of the current project's source code. This includes your primary module as well as any additional modules you create in your Source directory.
Project Plugin Modules	Modules associated with plugins located in your project's directory, under (Project Name)/Plugins.



You should avoid preloading Engine modules unless you are actively iterating on the engine's source code.

In addition to these sets of modules, you can use the **Preload Named Modules** array to choose specific modules that you want to load. This is useful if you are working only in a few specific modules within a large project, or if you want to load specific engine modules without loading the entire engine.



For more information about using Modules, refer to the [Unreal Engine Modules guide](#).

## Object Reinstancing With Live Coding

**Object Reinstancing** is an additional system that replaces existing instances of objects throughout the application and the editor. This means those objects can immediately take on any large, structural changes from code compiled through Live Coding. Reinstanced objects will include anything that uses the UCLASS, UFUNCTION, USTRUCT, UENUM, and UDELEGATE macros, as well as objects that **Unreal Header Tool** generates using those macros, such as Blueprint nodes.

Object Reinstancing is enabled by default. If you disable Object Reinstancing, Live Coding can still easily handle small changes to code such as changes in variable values or minor changes to existing functions. However, large-scale changes to code, such as new functions, new sets of variables, or dramatic re-factors, will behave unpredictably when you attempt to compile them without Object Reinstancing. This will usually result in crashes.

However, when Object Reinstancing is enabled, you need to follow additional steps to clean up pointers. If your code maintains pointers to objects that can be reinstanced, you need to use `ReloadReinstancingCompleteDelegate` and `ReloadCompleteDelegate` to update these pointers or invalidate caches so they can be repopulated later.

Hot Reload is more likely to tolerate reinstanced objects being improperly dereferenced, but Live Coding will cause crashes when the editor is being shut down. This is due to the fact that there can only be one version of any given class's destructor, and lingering instances of objects that haven't been cleaned up will cause conflicts between the old and new versions of reinstanced objects' destructors.