This section contains documentation about how to use C# to develop your game in CRYENGINE, and provides more information about the templates and example projects. To jump right into it, the CRYENGINE Launcher provides several templates to get you familiar with C# in CRYENGINE, or to quickly get started with developing your CRYENGINE game in C#.

Getting Started

CRYENGINE supports two ways of working with C#, managed plugins and C# assets. Managed plugins are stand-alone plugins that can be imported into multiple projects. C# assets are C# files that are in the assets folder of your project, and are compiled automatically by the Sandbox. To start working with either the managed plugins or C# assets you will first have to create a new project.

The steps below assume that you already have a CRYENGINE account and have installed the most up to date version of the CRYENGINE Launcher and CRYENGINE.

1. Open the CRYENGINE Launcher from the icon on your desktop and login to your account.
2. Click the **+NEW** option in the **Projects** section (top left-hand side of the Launcher window).
3. Select the **C#** tab - this will reveal the templates that are available to you.
4. Select the template that you require.
5. Select a location where it will be located on your PC and choose a name for your project. Finally, click **Create Project.**

If you're using the engine-source from Github, you can create a new project by copying the template you want from **<root>/Code/GameTemplates/cs/** to the location where you want your project to be in your system. Once it is copied, open the **game.cryproject** file in Notepad and change the value of `"name"` to the name of your project. After changing the name, right-click the **game.cryproject** file and select **Switch Engine** and set it to the root-folder of your engine. If it asks you to register your engine, select **Yes**.
The generated C# solution for managed plugins and C# assets is a special CRYENGINE C# project, which can only be opened by editors with the CRYENGINE-extension. Opening the solution with an unsupported editor can result in unexpected behavior and corruption of data.

Managed Plugin

The solution for a managed plugin needs to be generated first before you can start programming.

1. To start programming, mouse over the relevant project name (left-hand side of the Launcher window) which will reveal a Settings icon  . Click the icon and then click on **Reveal in Explorer.**
2. This takes you to the folder where your project is located. Double click on the folder and this will expose other folders including the relevant template.
3. Right click the **Game.cryproject** file in this folder and then click **Generate solution.**
4. Open the **Code** folder and run the **Game.sln** file with a CRYENGINE compatible editor, such as **Visual Studio 2017**.

C# Assets

For the C# assets you don't need to generate and compile the solution yourself. Instead it's all done by the Sandbox whenever the files have changed. To get started you simply open the Sandbox and right-click in the Asset Browser. In the menu select **New...C# Script** which will create a new C# file in the Asset Browser. You can open the file by double-clicking it which will open a C# solution that contains the C# file.

By default the solution is opened with Visual Studio. If you want to open the solution with another IDE, or if you want to open the file directly instead of the solution, you can change this in the settings by going to **EditPreferences...GeneralFile**.

Developing in the CRYENGINE with C#

The C# project files have a special ProjectTypeGuid to identify them as CRYENGINE projects. This makes it possible for the CRYENGINE extension to attach the debugger to the CRYENGINE which is not possible by default. However, this also means that it's not possible to open the projects without the CRYENGINE extension.

Visual Studio 2017

To work with Visual Studio 2017 you first need to install the CRYENGINE extension for Visual Studio. This can be done by running the installer found at **<engine-root>/Tools/VisualStudioExtensions /CryEngine.Debugger.Mono.vsix**. After installing the extension you will be able to open the C# solution of the CRYENGINE.

Chapters:

Related Content:

- C# API Reference

With the extension installed, if you open a CRYENGINE solution you'll notice that the default **Start** button has changed to **Debug GameLauncher**. Pressing this button (or starting debugging with a hotkey) will now launch the GameLauncher and attach to it at the start. You can also press the dropdown-menu on the Start button to select other debug targets, for example the Sandbox or the dedicated server. These targets can also be changed from the Debug menu in Visual Studio. Switching the target will make either the **GameLauncher.exe**, **Sandbox.exe** or **Game_Server.exe** start, when debugging is started.

At the moment it's not yet possible to attach to an already running instance of the CRYENGINE.

Due to a bug in Visual Studio 2017 (version 15.7), the CRYENGINE extension will show an error due to missing files. To fix this:

1. First uninstall the previous extension in Visual Studio, **ToolsExtensions and Updates** and select **Uninstall** on the CRYENGINE Mono Debugger extension
2. Next install the fixed version of the extension by running the installer, **<engine-root>/Tools /VisualStudioExtensions/CryEngine.Debugger.Mono_15.7_FIX.vsix**

Note: The fixed version of the extension is included from CRYENGINE 5.5.0 Preview 4 onwards. Finally, the new extension will download the Xamarin workload which can take up several GB of disk-space.

Older Versions of Visual Studio

The C# project files have a special ProjectTypeGuid to identify them as CRYENGINE projects. This makes it possible for the CRYENGINE extension to attach the debugger to the CRYENGINE which is not possible by default.

You can however adjust the .csproj file of your project to make it possible for Visual Studio to open the solution.

First generate the solution as described in Getting Started. In your code folder, open the generated .csproj files with a text-editor like Notepad++ or Visual Studio Code.

Now search for the following line:

```
<ProjectTypeGuids>{E7562513-36BA-4D11-A927-975E5375ED10};{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}</ProjectTypeGuids>
```

Replace the line with:

```
<ProjectTypeGuids>{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}</ProjectTypeGuids>
```

Save the file. You can now open the solution file in Visual Studio.

Every time the solution is generated again by right clicking the .cryproject file and pressing **Generate Solution**, the ProjectTypeGuids will reset, so make sure to adjust the .csproj files again.

Xamarin Studio

Crytek no longer supports Xamarin Studio, we recommend the Visual Studio 2017 extension.