User Guide

InGame Code Editor

Fully customizable code editor for Unity

Trivial Interactive

Version 1.0.x

InGame Code Editor is an advanced input field for Unity that makes use of TextMesh Pro to display syntax highlighted code. Much of the inspiration for the syntax highlighting is taken from notepad++ and as a result it is extremely customizable allowing you to add syntax highlighting support for new languages.

Requirements

- Requires TextMesh Pro (Installable via package manager).

Features

- Very easy to use. Just drop the prefab into your scene and customize
- Full text editing features that you would expect
- Includes optional line numbers column
- Fully customizable editor themes
- Includes 3 pre-set themes: Light, Dark and Terminal
- Fully customizable syntax highlighting
- Highly optimized lexer for quick syntax highlighting of large text
- Supports syntax highlighting of keywords, symbols, numbers, comments and quote strings
- Keyword groups mean that you can highlight keywords in different colors
- Built in syntax highlighting support for C# (multiple), Lua, MiniScript and JSON
- You can easily add support for additional syntax highlighted languages
- Supports basic auto indentation using opening and closing tags
- Comprehensive .chm documentation of the API for quick and easy reference
- Fully commented C# source code included

Quick Start

This section will cover the steps required to get up and running as quickly as possible. More detailed information is provided later in the document.

1. **Install package**

   First you will need to install text mesh pro if it is not already installed. Open the package manager window by going to 'Window -> Package Manager' and switch to the 'All' tab to view a list of available packages. Find the package named 'TextMesh Pro' and click the install button.

   You can now import the InGame Code Editor package from the asset store as you would normally.

2. **Create code editor**

   Find the main UI prefab at 'Assets/InGameCodeEditor/InGameCodeEditor.prefab' and drag it into the scene making sure that it is a child object of a Canvas. Resize the code editor as required so that it will be usable in game. Note that this prefab is setup for C# syntax highlighting by default but you can customize this as covered later in this document.

3. **Test Code editor**

   Hit the play button to enter play mode and begin writing or pasting C# source code. The code should now be fully highlighted as you type.

Useful Tips

Line highlighter is displayed incorrectly.

If you are using Unity 2018 or newer you will see an additional compatibility option on the 'Code Editor' component called 'Apply Line Offset Fix'. This is added for compatibility in newer Unity versions which make use of the TMP package distributed with the Unity install. In some versions you may need to enable this option to correct the line highlighter positioning as it may be displayed on the line above.

Adjusting the tab character size

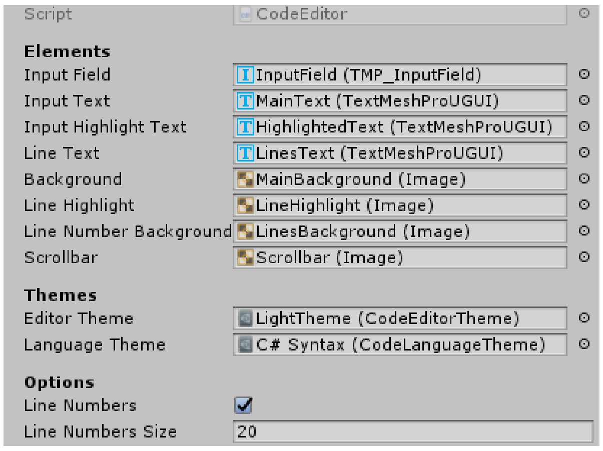
If you are using the later TMP version which is automatically included in Unity projects then you may see abnormally large tab spacing when running the code editor. This can easily be fixed by selecting the following font asset in the project window: 'Assets/InGameCodeEditor/Fonts/RobotoMono-Regular SDF.asset'. In the inspector window you should see the following options:

- **Tab Width** – Adjust the size of the tab spacing.
- **Tab Multiple** – Adjust the number of tab spacings that make up a tab character.

You can adjust these values to reduce the tab character to a normal size spacing or to any other spacing that you may require.

Code Editor Inspector

The ingame code editor has a number of inspector values that can change the appearance or behaviour of the code editor. There are also a number of references that must be assigned in order for the code editor to work correctly:



Script	CodeEditor
Elements	
Input Field	InputField (TMP_InputField)
Input Text	MainText (TextMeshProUGUI)
Input Highlight Text	HighlightedText (TextMeshProUGUI)
Line Text	LinesText (TextMeshProUGUI)
Background	MainBackground (Image)
Line Highlight	LineHighlight (Image)
Line Number Background	LinesBackground (Image)
Scrollbar	Scrollbar (Image)
Themes	
Editor Theme	LightTheme (CodeEditorTheme)
Language Theme	C# Syntax (CodeLanguageTheme)
Options	
Line Numbers	✓
Line Numbers Size	20

Figure 1

Elements

The elements section contains all required component references that must be assigned in order for the code editor to function correctly. It is recommended that you use the included prefabs which have all of these references already setup.

- **Input Field:** A reference to a TMP input field that is used for typing code.
- **Input Text:** A reference to a TMP text component that is used to display the text in the input field.
- **Input Highlight Text:** A reference to a TMP text component that will receive highlighted text. This text component must have the same configuration as 'Input Text' and should be directly overlaid.
- **Line Text:** A reference to a TMP text component that is used to display the line numbers.
- **Background:** A reference to a Unity UI image representing the main background of the code editor.
- **Line Highlight:** A reference to a Unity UI image that is overlaid on the current line.
- **Line Number Background:** A reference to a Unity UI image representing the background area of the line numbers area.
- **Scrollbar:** A reference to a Unity UI image representing the scrollbar slider.

Themes

The themes section allows you to assign a number of theme assets which can be used to skin the editor and change its appearance.

- **Editor Theme:** A theme asset that contains color information for every aspect of the code editor. See the 'Custom Editor Theme' section for more information.

- **Language Theme:** A theme asset that contains syntax highlighting colours and keywords which the code editor will use to syntax highlight any entered text. See the ‘Custom Language Theme’ section for more information.

Options

- **Line Numbers:** Should the code editor display line numbers to the left of the input field.
- **Line Numbers Size:** The width of the line numbers column when line numbers are enabled.

Custom Editor Theme

You are able to create a number of themes that can be applied to a code editor either at runtime or edit time. There are also a number of included preset themes that you can choose from.

Themes are stored as assets and to create a new theme you will need to navigate to the project window. Select the folder where you would like to create the theme and then right click and select ‘Create -> InGame Code Editor -> Code Editor Theme’ to create a new asset. You will then be prompted to enter a name for the asset.

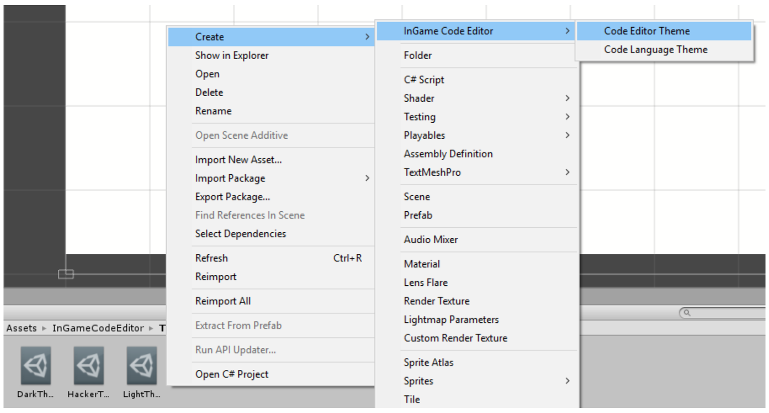
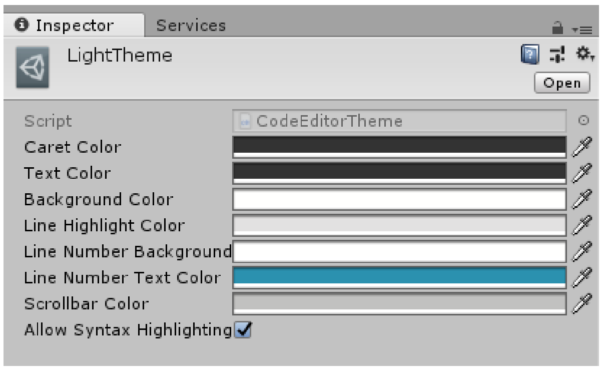


Figure 2

Once created you can select the asset to customize it in the inspector window:



LightTheme

Script CodeEditorTheme
Caret Color
Text Color
Background Color
Line Highlight Color
Line Number Background
Line Number Text Color
Scrollbar Color
Allow Syntax Highlighting ✓

Figure 3

- **Caret Color:** The color of the input field caret. For best results this should contrast the background color.
- **Text Color:** The color of the input field text. For best results this should contrast the background color.
- **Background Color:** The color of the input field background area.
- **Line Highlight Color:** The color of the current line highlight bar. Note that this color will overlay the text so generally colors closer to the background color will be preferred.
- **Line Number Background Color:** The color of the line numbers column background.
- **Line Number Text Color:** The color of the line numbers text. For best results this should contrast the line number background color.
- **Scrollbar Color:** The color of the scrollbar handle.

- **Allow Syntax Highlighting:** Should the theme allow the input field text to be syntax highlighted. When false syntax highlighting will be disabled on the code editor and all input text will be given the main text color.

Once you are happy with a theme you can apply it to a code editor script either at runtime or edit time.

1. **Assign at runtime:**

   To assign a theme at runtime you will need a reference to a 'CodeEditor' script and a 'CodeEditorTheme' scriptable object. This example will use public fields for the references which can be assigned via the inspector window. You can then assign the theme as shown below:

   C# Code

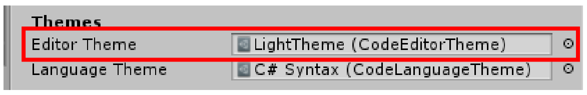
```csharp
using UnityEngine;
using InGameCodeEditor;

class Example : MonoBehaviour
{
    // We assume that these values are assigned in the inspector
    public CodeEditor editor;
    public CodeEditorTheme theme;

    void SetTheme()
    {
        // Assign the property and the UI colors will be updated
        editor.EditorTheme = theme;
    }
}
```

2. **Assign at edit time:**

   To assign a theme at edit time you will need to select the code editor script in the hierarchy. The code editor script is always on the root object of the InGameCodeEditor prefab so you can just select the root object. You can then drag the desired theme asset to the 'Editor Theme' slot of the code editor which can be found under the 'Themes' heading.



Themes

Editor Theme	LightTheme (CodeEditorTheme)
Language Theme	C# Syntax (CodeLanguageTheme)

Figure 4

Custom Language Theme

InGame Code Editor uses language themes to make the syntax highlighting system fully customizable as well as making it very easy to switch between different languages. Much of the inspiration for the syntax highlighting was taken from notepad++ and as a result it is a very flexible and customizable way of highlighting not only keywords but symbols, numbers and more. There are a couple of pre-set themes included mainly for the C# language but you can easily add your own language.

As with editor themes, language themes are stored as assets and can be created from the project window. Simply right click on the folder where you would like the theme asset to be created and select 'Create -> InGame Code Editor -> Code Language Theme'.

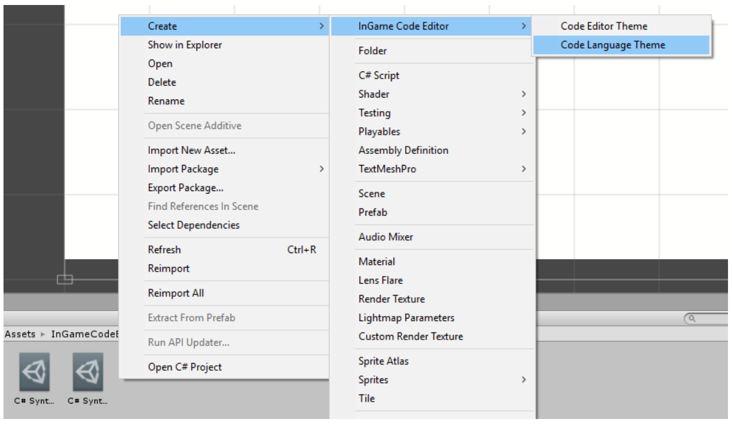
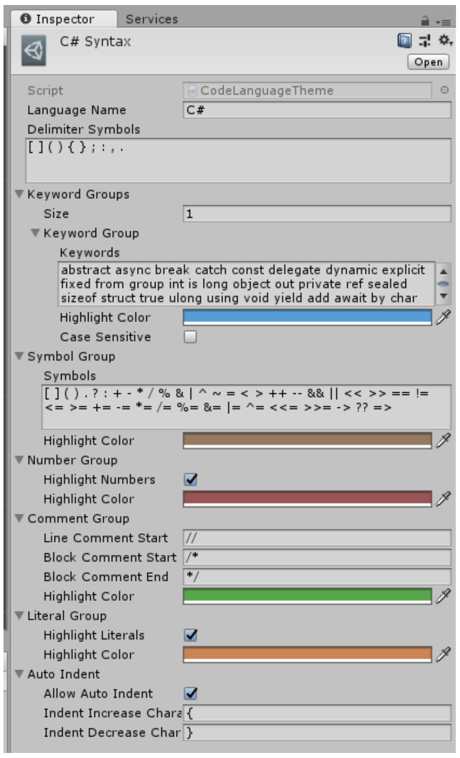


Figure 5

Once created you can select the asset to customize it in the inspector window:



```
Inspector    Services
C# Syntax                                    Open

Script                  CodeLanguageTheme
Language Name           C#
Delimiter Symbols
[ ] ( ) { } ; : , .
▼ Keyword Groups
    Size                1
  ▼ Keyword Group
      Keywords
      abstract async break catch const delegate dynamic explicit
      fixed from group int is long object out private ref sealed
      sizeof struct true ulong using void yield add await by char
      Highlight Color
      Case Sensitive    ☐
▼ Symbol Group
    Symbols
    [ ] ( ) . ? : + - * / % & | ^ ~ = < > ++ -- && || << >> == !=
    <= >= += -= *= /= %= &= |= ^= <<= >>= -> ?? =>
    Highlight Color
▼ Number Group
    Highlight Numbers   ✓
    Highlight Color
▼ Comment Group
    Line Comment Start  //
    Block Comment Start /*
    Block Comment End   */
    Highlight Color
▼ Literal Group
    Highlight Literals  ✓
    Highlight Color
▼ Auto Indent
    Allow Auto Indent   ✓
    Indent Increase Chara {
    Indent Decrease Char  }
```

Figure 6

- **Language Name:** The name of the language that the theme provides syntax highlighting for. For example: C#
- **Delimiter Symbols:** Delimiter symbols are special characters that may appear immediately before or after a keyword and will allow the keyword to be highlighted correctly as if it was surrounded by whitespace. Note that only single characters may appear in this text area and a space can be used to separate multiple symbols.

- **Keyword Groups:** Keyword groups is an array of keyword group where any number of groups can be specified. A keyword group defines a number of words and a color which will be used to highlight these words. By specifying multiple keyword groups you can have batches of words highlighted in different colors.
- **Keyword Group - Keywords:** Keywords can be any word ideally containing letters only and will be highlighted with the specified color when matched in the code editor input field. In order for a keyword to be highlighted all characters of the word must be matched based on the specified case sensitivity. A keyword must also be immediately proceeded or terminated by either whitespace / newline, a symbol specified in 'Delimiter Symbols', a symbol specified in 'Symbol Group', a symbol specified in 'Comment Group' or a quote string opening / closing. Multiple keywords can be specified by separating them with a space.
- **Keyword Group – Highlight Color:** The color that all keywords in the group will be highlighted with.
- **Keyword Group – Case Sensitive:** Should the keyword match be case sensitive.

- **Symbol Group:** A symbol group defines a number of characters of short words containing only symbols that can be highlighted and also act as a delimiter for keywords. If you want a symbol to act as a delimiter but not be highlighted then you should use the 'Delimiter Symbols' field instead.
- **Symbol Group – Symbols:** Symbols can be one or more characters long and should ideally only contain language symbols such as brackets, colons operators and more. Symbols are treated specially by the parser and act as delimiters as they can appear immediately before or after a keyword and will still cause the word to be highlighted correctly. Multiple symbols can be specified by separating them with a space.
- **Symbol Group – Highlight Color:** The color that all symbols in the group will be highlighted with.

- **Number Group:** A number group is used to highlight numeric characters including decimal points. Numbers of any length and containing any number of decimal points will be highlighted.
- **Number Group – Highlight Numbers:** Should number highlighting be enabled. When false all numbers will not be highlighted and will be displayed in the active text color of the assigned editor theme.
- **Number Group – Highlight Color:** The color that all numbers in the group will be highlighted with.

- **Comment Group:** A comment group is used to specify how comments will be highlighted for the language. Note that atleast one of the 'Start' fields must be filled out in order for comment highlighting to be enabled.

- **Comment Group – Line Comment Start:** The character or word that is used to denote the start of a line comment. Note that the comment start characters will be highlighted as well as the following text until a new line or the end of the text is found.
- **Comment Group – Block Comment Start:** The character or word that is used to denote the start of a block comment. Note that the comment start characters will be highlighted as well as the following text until the 'Block Comment End' string is matched or the end of the text is found.
- **Comment Group – Block Comment End:** The character or word that is used to denote the end of a block comment. Note that the comment end characters will also be highlighted.
- **Comment Group – Highlight Color:** The color that all comments in the group will be highlighted with.
- **K**
- **Literal Group:** A literal group is used to highlight text enclosed in double quotes. Note that only double quotes are supported and single quotes will not be highlighted.
- **Literal Group – Highlight Literals:** Should literal highlighting be enabled. When false all literals will not be highlighted and will be displayed in the active text color of the assigned editor theme. When true all text enclosed in double quotes will be highlighted with the specified color.
- **Literal Group – Highlight Color:** The color that all literals in the group will be highlighted with.

- **Auto Indent:** InGame Code Editor supports basic auto indentation but you must enable it in the language theme. This is because the auto indentation relies on special opening and closing indent characters which may be specific to the language you are highlighting. When enabled, hitting the return key will cause the correct amount of tab characters to be inserted based upon the caret location in the text and the current indent level.
- **Auto Indent – Allow Auto Indent:** When enabled auto indent will automatically insert tab characters before the caret when you move to a new line to bring the caret inline with the indent level.
- **Auto Indent – Indent Increase Character:** A special character used to denote the increase in indent level. Note that only languages that support and opening and closing structure such as bracing can auto indent enabled.
- **Auto Indent – Indent Decrease Character:** A special character used to denote the decrease in indent level.

Once you are happy with a language theme you can apply it to a code editor script either at runtime or edit time.

1. **Assign at runtime:**

   To assign a theme at runtime you will need a reference to a 'CodeEditor' script and a 'CodeLanguageTheme' scriptable object. This example will use public fields for the references which can be assigned via the inspector window. You can then assign the theme as shown below:

C# Code

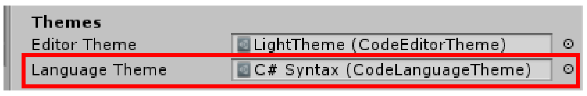
```csharp
using UnityEngine;
using InGameCodeEditor;

class Example : MonoBehaviour
{
    // We assume that these values are assigned in the inspector
    public CodeEditor editor;
    public CodeLanguageTheme theme;

    void SetTheme()
    {
        // Assign the property and the keywords will be highlighted
        editor.LanguageTheme = theme;
    }
}
```

2. **Assign at edit time:**

   To assign a theme at edit time you will need to select the code editor script in the hierarchy. The code editor script is always on the root object of the InGameCodeEditor prefab so you can just select the root object. You can then drag the desired language theme asset to the 'Language Theme' slot of the code editor which can be found under the 'Themes' heading.



Themes

Editor Theme	LightTheme (CodeEditorTheme)
Language Theme	C# Syntax (CodeLanguageTheme)

Figure 7

Report a Bug

At Trivial Interactive we test our assets thoroughly to ensure that they are fit for purpose and ready for use in games but it is often inevitable that a bug may sneak into a release version and only expose its self under a strict set of conditions.

If you feel you have exposed a bug within the asset and want to get it fixed then please let us know and we will do our best to resolve it. We would ask that you describe the scenario in which the bug occurs along with instructions on how to reproduce the bug so that we have the best possible chance of resolving the issue and releasing a patch update.

http://trivialinteractive.co.uk/bug-report/

Request a feature

InGame Code Editor was designed as a complete runtime code editor input field, however if you feel that it should contain a feature that is not currently incorporated then you can request to have it added into the next release. If there is enough demand for a specific feature then we will do our best to add it into a future version.

http://trivialinteractive.co.uk/feature-request/

Contact Us

Feel free to contact us if you are having trouble with the asset and need assistance. Contact can either be made by the contact options on the asset store or buy the link below.

Please attempt to describe the problem as best you can so we can fully understand the issue you are facing and help you come to a resolution. Help us to help you :-)

http://trivialinteractive.co.uk/contact-us/