# Developing On Scaffold

Copyright (c) 2020. Ascendant Design and Training, LLC.  
Last updated by Daniel Patterson, Tuesday, October 20, 2020

Developers in the community are welcome to help out, test the runtime, or even use the beta versions for making your own educational projects (once we get to that point).

## Description

This project is developed in the .NET Framework, written in C#, and has a WinForms user interface. It currently runs on Windows and will be made cross-platform compatible after some of the first versions have been stabilized.

The current version is configured to work on Windows 64-bit desktop systems like the following.

* Windows 10.
* Windows 8 (64-bit).
* Windows 7 (64-bit).

## Distribution

The application currently contains the CefSharp HTML rendering subsystem for various variable display purposes. When distributing the application, that library is dependent upon the following[[1]](#footnote-1):

Minimum of .Net 4.5.2

Minimum of 'Visual C++ 2015 Redist' is installed (either 'x86' or 'x64' depending on your application). VC++ 2017/2019 are backwards compatible.

Please ensure your binaries directory contains these required dependencies:

* libcef.dll (CEF code)
* icudtl.dat (Unicode Support data)
* CefSharp.Core.dll, CefSharp.dll,
* CefSharp.BrowserSubprocess.exe, CefSharp.BrowserSubProcess.Core.dll
* These are required CefSharp binaries that are the common core logic binaries of CefSharp.
* One of the following UI presentation approaches:
  + CefSharp.WinForms.dll
  + CefSharp.Wpf.dll
  + CefSharp.OffScreen.dll
* Additional optional CEF files are described at: https://github.com/cefsharp/cef-binary/blob/master/README.txt#L82

NOTE: CefSharp does not currently support CEF sandboxing.

By default 'CEF' has it's own log file, 'Debug.log' which is located in your executing folder. e.g. 'bin'

More information can be found in the ReadMe.txt file at {GitHubProjects}/Scaffold/packages/CefSharp.WinForms.84.4.10

## Installation

To get a developer instance of Scaffold running, follow these steps.

* Verify that you have Microsoft Visual Studio 2019 or later installed on your PC.
* Open a command-line or terminal window.
* Issue the following commands on the command line.

<pre><code>git clone <https://github.com/ascendantdesign/Scaffold>  
cd Scaffold/Source/Scaffold</code></pre>

If Visual Studio 2019 or later is your default Visual Studio editor, you can type the following filename directly from the command-line to open the solution.

<pre><code>Scaffold.sln</code></pre>

Otherwise, you can open the project from Visual Studio 2019 or later from the folder {GitHubProjects}/Scaffold/Source/Scaffold/Scaffold.sln

Note that the redundancy in naming policy has to do with the fact that multiple projects are present in the Scaffold/Source folder, and Scaffold/Source/Scaffold contains the central solution of the project.

## Non-Linear Decision Tree Editor

The non-linear decision tree feature of Scaffold uses node network editing components to allow freeform expression of extremely complicated flows.

### Things You Can Do With Decision Tree Editor

A node editor for designing a flowing decision tree can be valuable in a number of cases.

* Designing branching scenario and non-linear course modules. This includes the following applications.
  + Adaptive course.
  + Changing stakes course.
  + Conversational session.
  + Escape-the-room game.
* Defining non-sequential pathways.

### Editor Make-Up

The editor uses only three types of nodes. Each of the node types is specialized in what can possibly happen at a certain step.

### Start

The start node has no inputs and allows for an output fan-out condition. Each output can only be sent to one destination.

Following are some of the characteristics of the start node.

|  |  |
| --- | --- |
| Name | Description |
| Inputs | 0 |
| Input Source |  |
| Question | Mandatory |
| Answers | Multiple |
| Output Style | Per answer |
| Output Targets | Singular |

### Fork

A fork node has a single input capable of receiving signals from multiple nodes. The fork has a fanned output where each output node is only capable of being directed to a single target, similarly to the start node.

Following are some of the identifiable features of the fork node.

|  |  |
| --- | --- |
| Name | Description |
| Inputs | 1 |
| Input Source | Multiple |
| Question | Optional |
| Answers | Multiple |
| Output Style | Per answer |
| Output Targets | Singular |

### Termination

The termination node has a single input capable of receiving signals but no output nodes.

Following are some of the notable traits of the termination node.

|  |  |
| --- | --- |
| Name | Description |
| Inputs | 1 |
| Input Source | Multiple |
| Question | None |
| Answers | Singular |
| Output Style | None |
| Output Targets |  |

## Windows and Views

In this discussion, a view control is normally similar to a canvas of potentially infinite size, where anything might be drawn or portrayed without hard boundaries. Conversely, a window control is typically an area that can be drawn upon but has a distinct limitation in position and size.

In many cases, view and window controls are used together. In those cases, the view sits behind the window sliding back and forth, while the window effectively crops the view into the form.

## Scrollbar Behavior

Scrollbars move the current view along the X and Y axis within the window. When the scrollbar is at the very top of its range, the top of the window is aligned with the top or left sides of the view. The opposite is also true. When the scrollbar is set to the bottom of its range, the bottom or right edges of the window are aligned with the bottom or right edges of the view, respectively.

1. From CefSharp NuGet Readme.txt at <https://github.com/cefsharp/CefSharp/blob/master/NuGet/Readme.txt> [↑](#footnote-ref-1)