

Intro to C#

# C++ Framework Generator

---

Sebastian Toy

8th September, 2017



# Table of Contents

[Table of Contents](#)

[Purpose](#)

[Target Audience](#)

[Requirements](#)

[Features](#)

[UI Mockups](#)

[Save File Format](#)

## Purpose

The C++ Framework Generator is a C# Winform tool that allows the user to generate an XML database of C++ classes, members and parameters. It will serve as a way for programmers familiar with the language to visually plan out their projects before-hand, as well as provide files in a format standard enough to be easily externally interpreted and converted into actual C++ .h and .cpp files. The Visual Studio IDE contains a similar tool called the Class Wizard, however it contains issues that the C++ Framework Generator solves such as a poor user interface, inability to save and load files, and cumbersome functionality.

## Target Audience

The C++ Framework Generator is designed specifically for any programmer familiar with the C++ programming language and provides customization options that facilitate both beginners and advanced users of C++. The tool provides an avenue for these programmers to extensively plan out the framework of any object-oriented project by breaking it down into classes, members and parameters, and provides extensive customization for each item. Programmers will be able to use this application independent of any IDE or program, as well as be able to save their progress, allowing for portable project planning that can be easily shared between programmers working on the same project.

## Requirements

The C++ Framework Generator will run on any device that can run Windows 7 or higher, with the only external dependency being the compiled-help manual format which is already intrinsic to the Windows operating system.

## Features

The C++ Framework Generator (CFG) supports much of the same functionality as the Class Wizard (CW) in Visual Studio, however it also expands and refines on it.

- Add and modify multiple classes, members and parameters in a way that isn't overwhelming  
**Description:** The CFG will be able to display multiple items at a time via a list with text representation of the customization options in a manner similar to the C++ syntax. Single-clicking on the class will reveal its associated member list, and when adding a member there will be the option to make it a function which will reveal the parameter list. Double-clicking on an item will allow its options to be modified in case the user changes their mind. Finally, using shift-click and ctrl-click will allow for multiple items to be removed at once.

---

**Class Wizard Comparison:** While the CW allows for multiple classes, each class is hidden in a drop-down menu which makes it difficult to have a broader understanding of the current project. Classes and members also can't be modified after being added, nor can multiple items be selected to be deleted at once. Finally, the addition of members is unnecessarily segmented into three separate tabs whereas the CFG will be streamlined to allow the member to become a virtual function, variable or method depending on the options selected in the member pop-up form.

- Extensive customization options for classes, members and parameters

**Description:** Adding an item will produce a pop-up with the relevant customization options. For classes this will include the name, option of virtual destructor, and additional option of inheritance with separate access level and base class name fields. Members will have a variety of options including access level, optional modifier keyword, type, name and memory type, as well as the addition of specifying if it is a function, and if so whether it is virtual or not. Additionally, if the member is a function there will be the opportunity to add parameters which in turn will have all plausible customization including type, name, const status and memory type.

**Class Wizard Comparison:** The CFG matches the customization options of the CW as well as expanding on them, such as the ability to give members modifier keywords.

- Represent C++ object information in a detailed yet understandable way

**Description:** Each C++ item will be represented in a list, with each entry serving as a text representation of the options that have been selected for that item in a manner similar to the syntax of actual C++ code. This will allow for programmers to tell the attributes of each class, member and parameter at only a glance.

**Class Wizard Comparison:** The CW abstracts information about members and parameters so that the programmer has to waste time double-clicking on each item, and instead of displaying the currently chosen options it jumps to the relevant point in the actual C++ code itself. Class information is also incomplete, such as the access level of its inheritance, and it is difficult to discern whether certain information is relevant to the classes, members or parameters.

- Provide a minimalistic, aesthetically pleasing and user-friendly interface

**Description:** The CFG will be designed with a complimenting color palette of orange, black and gray to ensure that text is clear and easy to read, and that there is a consistency of design across all forms. Functionality will be streamlined so that the programmer only has to consider the relevant design elements for what task they currently want to do, examples include hiding the members list until a class is selected, and hiding the parameter list when a member isn't a function. Finally, each form will be intuitive to use with appropriate labelling and universal representation of items such as the add and remove buttons.

**Class Wizard Comparison:** The CW has a standard form color scheme that is very bright and straining on the eyes. The interface has been divided into separate tabs with no attempt to streamline functionality, meaning that the user is constantly bombarded with information not relevant to their current task. Form elements are kept very close together making it hard for the user to visually separate functionality in their mind. Finally, by presenting the user with so many options at once it interrupts workflow and their focus on the current task.

- Ability to save and load data-bases containing information about classes, members and parameters in the XML format

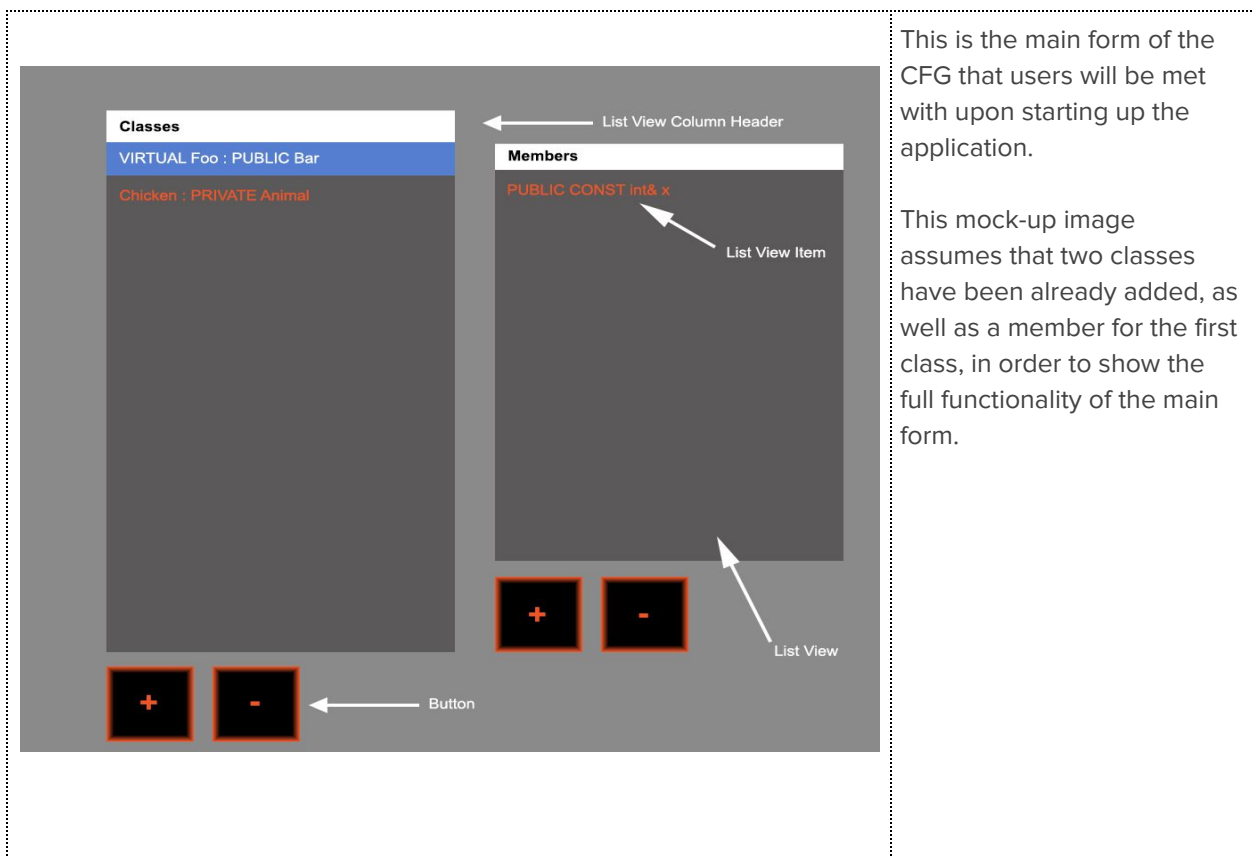
**Description:** The CFG will use its own custom extension '.fwg' in order to read and write files in a specific way as per the specifications of the XML format. This will ensure that progress can be easily saved and files can be easily shared between programmers working on the same project.


**Description:** The CW is unable to independently save or load files as it directly produces code into .h and .cpp files within the Visual Studio IDE.

## UI Mockups

In order to develop a consistent theme of a minimalist and ergonomic interface, two mock-up images have been designed as a projection of what the application will look like, and how the user will interact with it. The first image is of the main form, and the second screen is the class pop-up form. It is important to note that these are not the only screens featured in the C++ Framework Generator, but they are representative of the goals for the complete user interface.

Screen 1:





#### Scene 1: Events:

- Classes List-view Item single click:
  - Selected class is visually highlighted in the Classes list-view.
  - Members list-view and buttons are made visible.
  - Members list-view is populated by whatever member data is stored within the selected class.
- List-view Item double click:
  - Check if relevant pop-up form is already active, and if not load it in editing mode (item data is already filled out based off the previous options)
- + Button clicked:
  - Check if relevant pop-up form is already active, and if not load and display it to the user.
- - Button clicked:
  - Remove relevant list-view items based on selection:
    - Nothing selected - Remove item at bottom of the list
    - Item(s) selected - Remove selected items

## Screen 2:

The diagram shows a dark gray rectangular form with the following elements and labels:

- Class Name**: A label with an arrow pointing to a gray text box.
- Virtual Destructor**: A label with an arrow pointing to a white checkbox.
- Inherit From Class**: A label with an arrow pointing to a white checkbox.
- Base-Class-Options**: A label with an arrow pointing to a white-bordered box containing:
  - PUBLIC**: A label with an arrow pointing to a dropdown menu (Combo-box).
  - Group-box**: A label with an arrow pointing to a gray text box.
- Generate Class**: A label with an arrow pointing to a red-bordered button.

This is the class pop-up form that is displayed when the user clicks the '+' button underneath the Classes list-view in the main form.

It is representative of the style of the member and parameter pop-up forms, and features all customisable options for adding a new class.

Each pop-up form will have two modes, generate and edit. Generate will be the default mode and will add the item to the list, whereas edit mode will modify the item selected in the list, as well as change the 'Generate Class' text to 'Confirm Changes'.

## Scene 2: Events:

- Inherit from class check-box checked:
  - Flip activation and visibility of inheritance options group-box.
- ENTER key is pressed or Generate Class button is clicked:
  - Check if all text fields are filled out, if not then class is not generated and instead a message box informing the user of this error is shown.
  - On successful class generation all the fields for customization will be represented via a string in the Classes list-view of the main-form, and finally the class pop-up form will close.
- ESC key is pressed
  - Class pop-up form will close.

## Save File Format

The C++ Framework Generator will save files with a unique '.fwg' extension in the XML format. This unique extension will ensure that only files produced by the C++ Framework Generator can be loaded due to the specific nature of the in-built XML interpreter. Files will be written and read in the following custom format:

```
<CLASSES>
  <Class name="".../>
    <MEMBERS>
      <Member name="".../>
        <PARAMETERS>
          <Parameter name=""/>
        </Parameter>
      </PARAMETERS>
    </Member>
  </MEMBERS>
</Class>
</CLASSES>
```

It is important to note that the CFG will also support Winform dragging and dropping functionality by allowing a single .fwg file to be loaded via dragging it into the main form of the application.