



BASIC2AGKTier2

Introduction

Basic2AGKTier2 is an application that allow a developer to program in a basic language that is similar to AGK Tier1 and translate the basic code to a c++ that is compatible with Tier2 libraries. This basically gives you an easy to use language in the form a "BASIC" and the power of "C++". With that said, this is not a tier to tier converter. This will not take a tier 1 agc file and out of the box convert it to c++. Sorry. Even though the languages are similar there are some differences. With that said, it is very easy to port a tier 1 file over to basic2agktier2 with minimal tweaking of the code (mainly syntax). Most of the differences come with variables and arrays. Everything else is syntax; Ex. "if" statement require the statement "then" at the end of the if line for both single line if statements and multiple line l statements.

Basic2AGKTier2 does use third party software to accomplish its goal. These are used as backends much like gcc is used as a backend to many language compilers out there. The programs used are BC9Basic which does a big chunk of the work load. It translates the basic code to c++. The plus side is that I don't have reinvent the wheel because it does a pretty good job at what it needs to do. The down side is that it was made for Windows desktop and console development so like many automated code generators, it adds code to the c++ source code that is not needed and conflicts with AGK Tier 2. The other program is astyle which is just a c++ code beautifier and nothing really more than that.

Requirement

Basic2AGKTier2 is considered portable to a point. What I mean by that is, that there is no installer for the software. It is a zip file that gets extracted to anywhere and you can run it anywhere. It doesn't use special config folders or registry entries. It can be even run from a flash drive.

Basic2AGKTier2 does however require that it be used on a machine that can handle .NET 4.7 or higher. So older machines may not be able to run it. It must be run on a Windows machine as some of the backend tools are developed for Windows. Of Course, the generated c++ file can be transferred to any Tier 2 Compatible platform.

Basic2AGKTier2 does need to be extracted to a Writeable location. To make it as portable as possible it must be able to write a config file to its own folder. Also, if you use the Project folder it will need to be able to write to it.

AGK Tier 2 Library must be installed at a writable location. If you utilize Basic2AGKTier2's ability to create a Tier 2 project, the folder where tier 2 library is installed must be writable and not require admin right to do so (*not a good practice to give admin rights to a program like this*). Of course, Tier 2 Library isn't needed to be able to run Basic2AGKTier2 directly. You just won't be able to do anything with the generated c++ code without the library.

What's included in the package.

Basic2AGKTier2.exe – This is the main program. It is the GUI Front end a long with code that takes the generated c++ code and fixes it to work with AGK Tier2 App Template system. I chose to follow this route as it is easier to work with cross platform by using the templates included.

AlphaFS.dll – This is a .NET library that Basic2AGKTier2 uses for File System access. It has some commands that the .NET System.IO doesn't have that makes it easier for Basic2AGKTier2 to do its file access.

FastColoredTextBox.dll – This is a .NET Library that is usually made for creating source editors. As you will see Basic2AGKTier currently has no editor. Basic2AGKTier uses it internally as it has some powerful functions to make it easier to find and remove code from the generated c++ source code.

Newtonsoft.Json.dll – This is a .NET Library for use in reading and writing Json files. Basic2AGKTier2 project files and config file is a json file as I found it easier to work with than xml. My opinion.

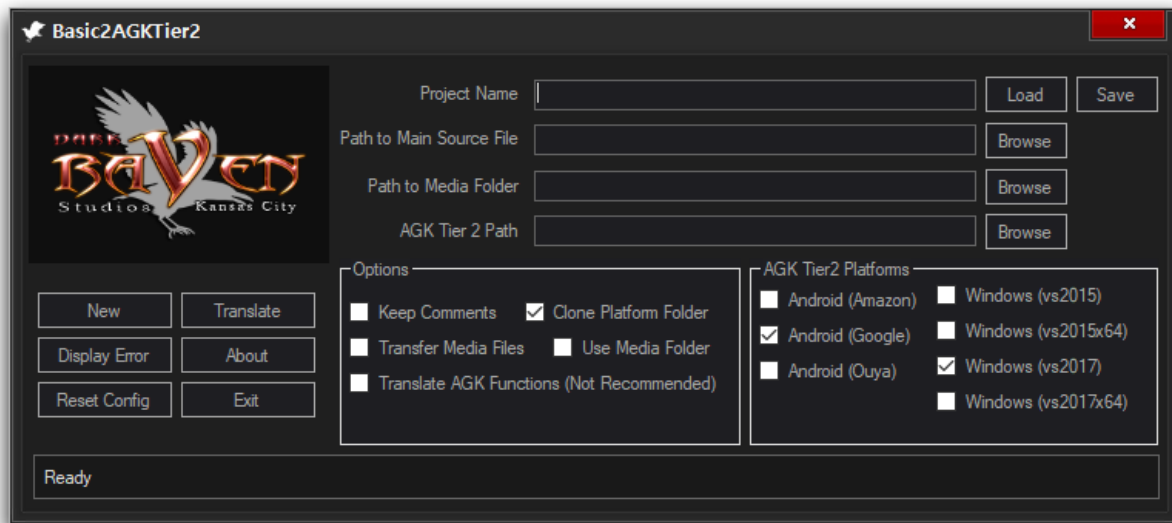
Astyle.exe, astyle.ini, visualstudio.ini – These all are part of the Astyle app. This is used in the backend for code formatting. These have to be present as some of my internal searching is based on spacing that Astyle creates when its run. If these aren't present an error may happen or the translation won't be successful.

Bc9.exe, bc9Help.chm – These two files are part of the BC9Basic package. Bc9.exe is used as the backend. It is a fork of the original Basic to c translator. It added c++ functions to the translator making it work best with AGK Tier 2. This is run first to generate large template.cpp (single file at the current time). But as said earlier that BC9 is really built to create Windows desktop applications and console applications. But I didn't really want to create a translator myself when I knew this existed. Bc9Help.chm is a help file that I've included as I didn't really include documentation on the specifics of the language, just how to use the gui.

AGKFunctions.fun – This file is a json file that contains a list of the AGK Functions. This is used for a part of the translation that will convert agk function that are presented in the basic of function calling: Ex. CreateSprite(1,1) to the C++ way of function calling: Ex. Agk::CreateSprite(1,1). *(See below for recommendations on using this).*

If you downloaded the supplemental zip file Projects.zip and extracted it to the Basic2AGKTier2 folder, then you should have a folder called Projects. These are samples of code ported to Basic2AGKTier2. These project are not mine nor do I take credit for any of it. They were taken from the forum and are used only a demos. These can give you an idea about Basic2AGKTier2 and how easy it is to port tier1 files over and also how powerful this language can really be. These have been tested on Window 10 desktop using Visual Studio and Android using Android Studio. I don't have any other platform to test the generated c++ code at the current time. In theory it should work by placing the generated "template.cpp" file in the app templates on you selected platform but I do not guarantee anything will work for every body.

Step by Step Guide



- 1) **Project Name:** This is a required field. This is only a name. If you wish to save your project settings for use later on then this is filename that is saved.
- 2) **Path to Main Source File:** This is a required field. This is the path to either a “.bas”. The recommended filename is “main.bas” though it can be anything thing. Just want to make sure it gives access to all included files. The translator will not check to see if this is a main file but this file is important. The translator will create a stub file which includes translator directives and also includes this file. If this isn’t the main file then there may be source files that would be included. The translator will not include all files from a project and will do nothing to the original basic files. This is also used for path purposes. If you use the project folder within the Basic2AGKTier2 application folder then you can use %PROJECT% macro instead of the base path to Basic2AGKTier2. This internal to the app and not an environment variable.
- 3) **Path to Media Folder:** This is an optional field. If you have media for your app and want it to be included in the Tier 2 project; Basic2AGKTier can move these files when it creates the projects. At the current time media assets to be copied must be in a single folder. The translator will copy everything in it except for tier1 byte code files. If you use the project folder within the Basic2AGKTier2 application folder then you can use %PROJECT% macro instead of the base path to Basic2AGKTier2. This internal to the app and not an environment variable.
- 4) **AGK Tier2 Path:** This is an optional field. This is the base path for the tier 2 library. This folder must contain the App folder. This folder must be writable to. This is path is not

required to run the translator. This path is required if you want the translator to create a tier project. Basic2AGK Tier code is based on the Appgamekit template idea as I've found it the easiest way to build projects for tier2. So for Basic2AGKTier2 to setup a Tier2 project it must have the path. Make sure this is not the App folder itself. It's the one that contains the App folder.

5) **Translator Options:**

- a. **Keep Comments:** The backend translator has the option to translate over basic comments to c++. If you check this option, then the translator will add the directive in the stub file that it generates. You can still use the directive anywhere in your code. All you have to is use "\$REMS". Read the BC9Basic Help file for more info on this directive. It is however not recommended to use this option. I've found a bug in the translation. The backend translator will move code around when generating the c++ code. If the comments are outside of functions or classes, they may not get moved with the code they are needing to be with and you may end up with comments sitting alone that make no sense. So, if you do use this option make sure comments always lie within functions or classes.
- b. **Clone Platform Folder:** If checked then Basic2AGKTier2 will copy over one of the Tier 2 app template folders and then insert the generated "template.cpp" file with in this project folder. This will only work if you had set the AGK Tier 2 Path setting up above. If this isn't checked then Basic2AGKTier2 will place the generated c++ file in the same folder as the main basic file.
- c. **Transfer Media Files:** If checked then Basic2AGKTier2 will copy over media files to the newly created Tier2 Project folder. This will only work if you have set the Media path, AGK Tier 2 Path, and Checked "Clone Platform Folder" option.
- d. **Use Media Folder:** If checked and Transfer Media Files is checked then Media files will be copied into a media folder in the Tier 2 Project instead of the default location. This makes it operate more like Tier 1. Source code will need to have the command set: *SetFolder("/media")* for it to work correctly. The translator will check to see if it can find the SetFolder function call and warn if it doesn't find it.
- e. **Translate AGK Functions:** This is probably a strange option as you would think that this should be automatic. Well it isn't because a problem came to light while translating. Since the fixer code doesn't really parse through the code; it doesn't know if it's in a block of code or not meaning if you have a class method that you named as the same name as an AGK function name then it will translate this to the C++ equivalent when it shouldn't. If you don't use classes or have user defined functions with same name as AGK's then this will work just fine. If you do you are recommended in not using this feature. This feature can also take a lot of time to run, depending on how much code you have. If used this basically will translate something like this: `CreateSprite(1,1)` to `agk::CreateSprite(1,1)`; If not used then you will need to make sure all AGK function calls are like `agk::CreateSprite(1,1)`;

- 6) **AGK Tier2 Platforms:** If you are familiar with how AGK Tier 2 platform templates look then this section will look familiar. This list the platforms available to Windows development. Because Basic2AGKTier2 will only work on Windows as a development system, the platforms available to the program are what you would find in the AGK Tier 2 package for Windows. You can still create a template.cpp for other platforms but you will be responsible in inserting the code file in those projects. I won't go into the specifics of this section as this in the AGK Documentation.
- 7) **Command Buttons (On the left side of the interface):**
- a. **New:** This button doesn't really create a new project but rather resets all the project settings.
 - b. **Display Error:** This button will pop up a message box that shows the most recent translator error. When source code is run through the backend translator and an error occurs, then the translator will create a template-stub.ERR file. This file contains the error message. The translator will stop when it hits an error. So the error needs to be fixed before it will go any further. Display error button will redisplay this message. If no error file is present it will display nothing.
 - c. **Reset Config:** This button will reset all settings that get saved in the config.json file on close. This will clear the Tier 2 Path and other settings that get set behind the scenes and resets them to the default settings.
 - d. **Exit:** This button will exit Basic2AGKTier2. This is the same as clicking the red X in the top right corner.
 - e. **About:** This button will display the about window. This contains licenses for the tools included along with their copyright info.
 - f. **Translate:** This button is the main button. It runs the translator. This will only be able to run if the required settings are set.

Conclusion

Thanks for using Basic2AGKTier2. This documentation is totally in depth in programming in this version of basic, so it is recommended to use the include BC9Basic help file; just ignore all the windows gui and console specific functions.

As you will see using Basic2AGKTier2 is so easy to port over your tier 1 code or just program somewhat like you would in tier1 to create a tier 2 app which gives you a lot more power than tier1.

It is recommended to take a look at the project samples as they show from basic porting to more advanced (classes) .