Learning to Program with Haiku

Lesson 23

Written by DarkWyrm

Writing a finished program is more than just writing the code that makes it run. Our fortune program still needs an icon and some other polishing.

Tweaking the Resources

One of the tasks remaining to further polish our program is to take care of the application's resources. They are kept in a separate file and are bundled into our program after it is built. Let's start by adding a resource file to our project.

1. In Paladin, choose Add New File from the Project menu as if you were going to add a new text file.
2. Enter Resources.rsrc for the new file's name and press Enter.

A shiny new resource file has been added to your project. Double-clicking on it will open it in the default resource file editor.

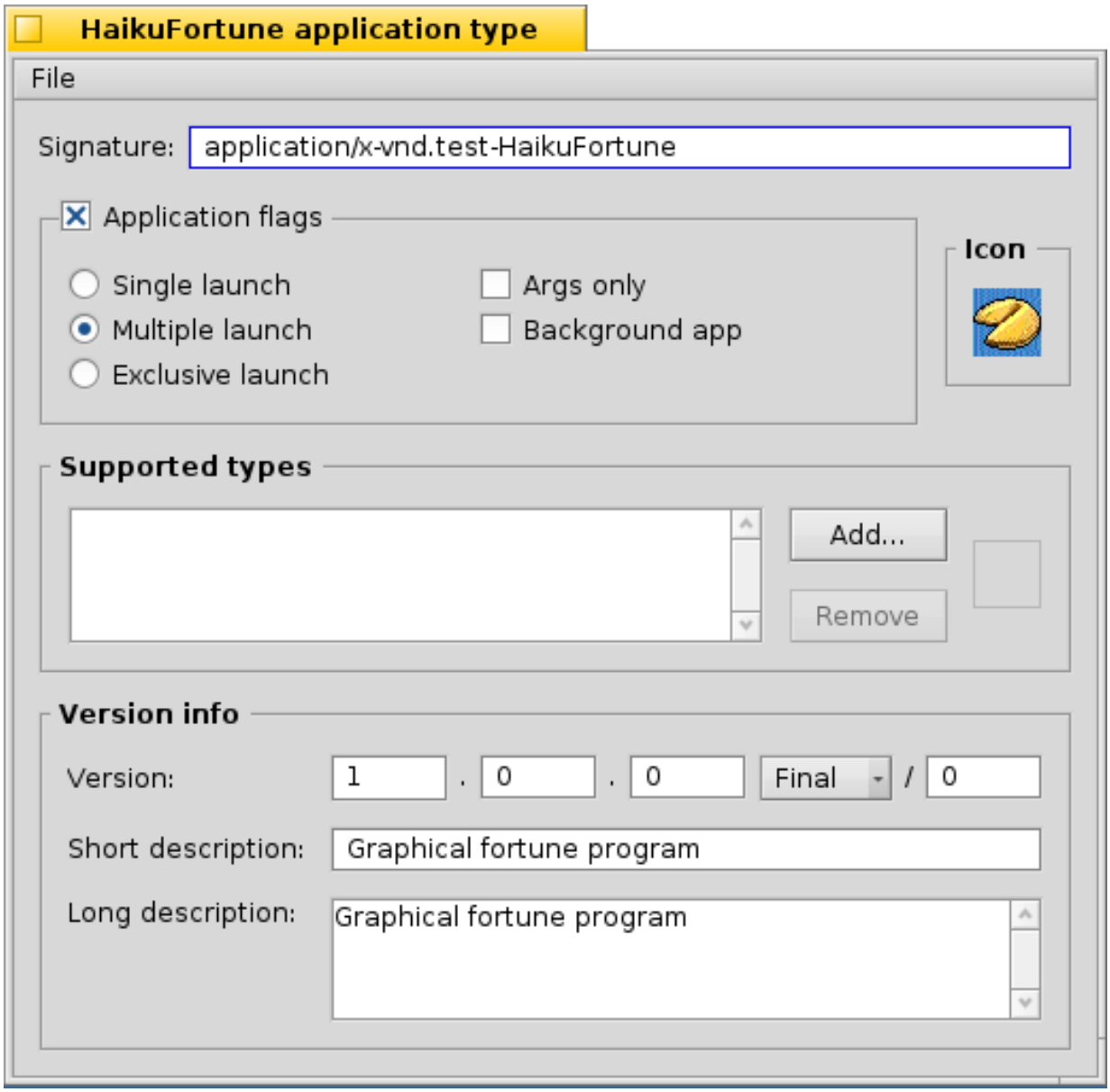
The first thing that we should do is set the program's icon, but let's address which icon we'll be setting. Haiku has three different icons that it can use: a vector icon, a large icon, and a mini icon. The last two are standard across all BeOS operating systems and are 256 color bitmaps 32 and 16 pixels square, respectively. The vector icon is a specialty format (**H**aiku **V**ector **I**con **F**ormat) developed specifically for Haiku which looks great while taking up very little storage space. Creating HVIF icons is beyond the scope of these lessons, so we'll focus on the bitmap formats here.

1. Open Resources.rsrc in QuickRes.
2. Drag HaikuFortune16.png and HaikuFortune32.png to the QuickRes window to add them to the resource file.
3. Click on the New dropdown menu in the top left corner of the QuickRes window and choose Icon. This will create two new entries: one of ICON type and one of MICN type. These correspond to the large and mini icons.
4. Click on the entry ICON type entry once to highlight it. Wait a little bit – long enough that QuickRes won't think you're double-clicking – and click on the entry's name, which should be "New Icon". If you've done it right, a text box will appear and you can edit the name.
5. Change the ICON entry's name to `BEOS:L:STD_ICON` and press Enter to finish editing the entry's name. Capitalization matters here, by the way.
6. Do the same for the MICN entry, but name it `BEOS:M:STD_ICON`.
7. Double-click on the entry for HaikuFortune32.png to open it into an editing window.
8. Select the picture and copy it to the clipboard using the Select All and Copy commands in the Edit menu.
9. Close the window for HaikuFortune32.png and double click on the ICON entry that you just created.
10. Click on the rectangular selection tool icon and click inside the bigger grid. This will make a blue box appear around the border of the bigger grid if there isn't one already.
11. Click on Paste in the Edit menu. The picture data from HaikuFortune32.png will appear in the bigger grid.
12. Do these same steps to copy the data from HaikuFortune16.png to the smaller grid in the window.

Once you've followed these steps, save the changes to your resource file, close QuickRes, and rebuild your project. If you've done everything correctly, your program's icon will change to a nice little

fortune cookie on a blue background. If not, go back and double-check your work to find what you've missed and make the necessary changes.

We'll also need to set the version, signature, and launch flags for our application. For this, we will use the FileType Tracker add-on. Open your project's folder, select the resource file, right-click on it and choose FileType from the Add-ons submenu.



Some of the parts of this window are self-explanatory, but it's best we go over all of them anyway, starting with the application signature at the top. The signature should be set to the same string as the one passed to the BApplication constructor in our code minus the quotes. It's not an earth-shattering mistake to leave this out of your program's resources, but it should match the signature sent to BApplication if it's there.

The Application Flags group sets some options for how your program can be started. Single launch mode, which is the normal – but not default – behavior, loads one copy of our program into memory

and runs it. Attempting to run a second copy doesn't do anything except switch to the program if it hasn't been minimized. However, if the user renames the executable and tries to run it, it will run that second copy, as well. Exclusive launch prevents this and ensures that only one copy of your program will be running at one time. Multiple launch allows for multiple instances of your program to be running at the same time. If you leave out the application flags in your program's resources, this is the default behavior. Checking the Args only checkbox means that the only information given to your program will be via the command line arguments it is given. No BeOS-style messages will be sent to it. This flag is most commonly used in command line applications. The Background app checkbox allows your program to be launched in the background without showing an entry in the Deskbar. This flag is typically used for server applications like the Media Server and the Mail Daemon.

The Supported Types box is related to any file types that your program explicitly supports. For example, if you were writing a spreadsheet application, you could put in various types of spreadsheets which it was able to manipulate. Our fortune program doesn't do anything that would need this information, so we'll skip this section.

The Version Info box contains controls for the version number of our application. The dropdown box is for the quality of the release, such as alpha, beta, golden master (also known as release candidate), and final. The short description should contain a short phrase describing what your program does. The long description field should contain more information – a sentence or three – but shouldn't contain a small novel.

The icon box is for more than just looks. Right-clicking on an existing icon will allow you to remove it or edit it. An icon can be dropped onto the box to add one. By double-clicking on the icon you can edit an existing icon or add a new one, but it is edited in the Icon-O-Matic vector icon editor and not QuickRes. As a result, if you edit the icon from here, you can edit only the vector icon for your application. Once you have everything set the way you like, save your work and close the window.

Source Code Licensing

Once the resources have been set, we should give some thought to choosing a license for our source code. All source code is licensed. Which license it falls under depends on the wishes of its author. Before distributing your program, you need to consider the license under which it will be published. Doing so will determine the course and lifespan of your program. In the United States, failure to choose a license causes a restrictive set of copyright clauses to be applied to your code. If you live outside the U.S., check with the appropriate authorities in your area.

First of all, consider whether or not your source code will be available to others – closed or open source. Haiku is founded upon open source software and the community around it is largely the same way. Unless you plan on eventually charging for your program, open source is a better option. If, at some point in the future, you decide that you don't want to maintain your project any more, it will be possible for someone else to continue your efforts. Many programs on the BeBits software website are now abandoned and development cannot continue for many of these great programs because the source code was not open and the developer just disappeared.

If you do choose to make the source code for your program available, there are a plethora of open source licenses from which to choose. Here is a list of the most popular open source licenses and a quick summary of what they say:

- MIT – The code can be used pretty much however a person wants so long as the included copyright information is left intact. This is the license under which Haiku is released. It is popular with other BeOS / Haiku developers for their own projects, as well.
- GNU Public License (GPL) – Any public binary distribution of GPL software must also make the sources available. Any code which uses GPL code must also be released under the GPL. This is the most popular license for software for the Linux operating system.
- Lesser GNU Public License (LGPL) – Any public binary distribution must also make the sources available. Any code which directly uses LGPL code must also be released under the LGPL, but if a program merely links against LGPL code, this is not required. In other words, linking against an LGPL library does not require the program that links against it to have the same license.
- Mozilla Public License (MPL) – As per the GPL, but any changes to MPL-licensed code must also be submitted back to the original project.
- Public Domain – The author relinquishes all rights to the code. Literally anything can be done with the code, including removing all copyright information and anything else you could think of. Also, once code is released into the public domain, it cannot be taken back. However, nothing would stop you from using it, improving it, and re-licensing your code.

Packaging Programs for Distribution

Publishing our program is another topic that should be addressed before concluding our work. There are two methods for distribution: package files and zip archives. Both have notable benefits and drawbacks.

Packages are the more powerful format of the two. They are created using the BeOS development tool PackageBuilder. Because PackageBuilder is part of the BeOS R5 Development tools package, they are free to download, but they can't be redistributed as part of another distribution. PackageBuilder is no longer being developed and will not run in a non-hybrid GCC4 Haiku environment. The packages that it generates can be installed in such a situation, however. It can install files in multiple locations on the computer. While it does not have facilities to create links in the Deskbar, symlinks can be created and installed just like any other file in the package. As a tradeoff for more flexibility, creating a package file takes more time.

Zip archives are the simpler and faster of the two formats. Set everything up the way you want it in the application folder, run the Zip-O-Matic Tracker addon, and that's it. This, unfortunately, places more burden on the user to install the application folder and create a symlink in the Deskbar. Zip archives are recommended for simple applications which do not need files in more than one location on the hard drive.

Where to Go from Here

We have covered an enormous amount of information since Lesson 1. Amazingly, there is still a lot of ground to go if you would like to go beyond being a beginner. The best way to improve is to write code and keep learning as you go. What you know now will give you the ability to write a many different kinds of programs for Haiku and will also give you the tools to learn the rest of the Haiku API and other aspects of programming in C++. Here are some topics that you are encouraged to explore:

C++

- Exceptions
- Templates and the Standard Template Library
- Multiple Inheritance

Usability

- *The Design of Everyday Things*, Donald Norman
- *The Humane Interface*, Jef Raskin

Good Programming

- *Design Patterns: Elements of Reusable Object-Oriented Software*, Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides

Programming for BeOS / Haiku

- *Programming the Be Operating System*, Dan Parks Sydow. This is out-of-print but is available from O'Reilly's website as a free PDF.
- The BeOS sample code projects and accompanying articles
- *The Be Book*. This is the authoritative manual on the BeOS / Haiku API.