Proposal for Allegro Design Scheme

Caleb “Carrus85” Deveraux

Revision 1

I. Current Design System

The current design scheme for the allegro programming library divides the community into 3 basic groups; those who want more in the distribution, those who think things should be removed from the distribution, and those who like it the way it is. The following is a proposal to attempt to appease all of these wants and needs, in addition to defining some of the ideas that should be behind various segments of allegro development.

Currently, the following "modules" exist within the standard, 4.1.8 WIP distribution of allegro.

1. Graphics Subsystem
   1. Initialization Routines
   2. BITMAP structure
      1. Blending functions
      2. Blitting functions
   3. Hardware Acceleration routines (basic 2d)
   4. Basic bitmap file parser routines
2. Drawing Primitives
3. 3d routines (software)
4. Keyboard Subsystem
   1. Localization handling
5. Joystick Subsystem
   1. Calibration Routines
6. Mouse Routines
7. Sound Routines
   1. MIDI Playback routines
   2. Audio Input routines
   3. WAV input/output routines
   4. Basic Audio Parser Routines
8. Config Routines
9. System Identification Routines
10. Datafile Routines
    1. Packfile output functions
11. Timer Routines
12. GUI Routines
13. Miscellaneous other routines.

II. Proposed System of Design

Now, the system of design that I propose would not only provide for those persons who want more out of the library, it also would provide for those who want to remove certain functions from the library for whatever reason. So, here goes the description.

The current library would be divided into two categories: Core routines, and Accessory (Level 1) Routines.

Core Routines (Level 0)

These are all of the hardware interfaces and necessary internal allegro functions. Anything that falls under this category must follow all of the following guidelines:

1) Must be directly referencing either a readily available operating system library or the hardware of a system.
2) Must not depend on another library that could not be packaged together with allegro freely.
3) Must not conflict with the allegro license.
4) Must be written in C or C with Assembly Optimizations.
5) Cannot reference any "higher level" routines.
6) This entire layer can either be statically linked, or dynamically linked into an executable.

Level 1 Routines

These are all of your "accessory" routines. Examples of these in the currently library would be the GUI subsystem, the 3d routines, and the fixed math routines. Must meet following criteria:

1) Must use only interfaces defined by the Core Routines Application Layer (aka. Can't do any dirty hacking by directly manipulating hardware at this level). This is kinda stretchable (you can directly access the memory behind a BITMAP or a WAV file, but just don't mess with the hardware without going through the proper channels).
2) Must be license compatible with allegro.
3) Must be able to be packed with an allegro distribution.
4) Can be written in either C or C++. (Wrapper routines for the core library would go at this application level). C functions must provide compatibility defines for C++ compilers and users.
5) Cannot refrence any "higher level" routines.
6) Routines must support "static-linking" capability. (Routine sets can be independently linked to an executable so you can use Level 1 functions without including the

entire level).

These two levels consist of what I call the "standard allegro distribution." All Layers depend on Layer 0, and higher level layers may depend on other lower-level layers.

On to the next level:

Level 2 Routines (The full distribution)

These would be all of the helper routines you could ever want that do not conflict with the base library license. These would include PNG loaders, OGG loaders, OpenGL wrappers, network abstraction layers, etc.

Level 2 Routines must follow these guidelines:

1) May rely on external libraries, to a certain extent. Libraries must be A) easy to install on all officially supported platforms, B) readily available and C) Have a compatible license that allows for library integration into allegro.
2) Must be compilable into allegro.
3) Must be statically linkable (in addition to dynamically linkable)
4) Can be written in either C or C++.

Level 3 Routines – The Third Party

These are all of your other, third party routines. These include routines that rely on other large libraries which the allegro developers have deemed "unfit" for integration into the core. Since these are independently maintained external to allegro, these have no restrictions.

III. Possible Distributions

Now, using these schemes, we can generate three distribution models, each with their own unique strengths.

Here they are:

allegro-core:
    The Barebones allegro distribution
    Library: liballegcore
    DLL: alcore4x.dll
allegro-standard:
    Standard allegro distribution
    Library: liballeg
    DLL: alleg4x.dll
allegro-full:
    Full-blown kitchen sink distro
    Library: liballegfull
    DLL: alfull4x.dll

Each successive distribution contains all of the previous distributions, and will not run unless all of the previous distros have been installed (downloading a vanilla full distribution without allegro-core and allegro-standard installed will result in major compile errors).