Alien Arena Developer Reference

Table of Contents


```
$Id: devref-en.asciidoc 2432 2011-04-21 06:18:57Z strat $
```

1. About this Document

Warning

This is an informal document and assumes general knowledge of programming and specific knowledge of Alien Arena development.

This document is intended to be informal reference for Alien Arena development. Its reason for being is it organize general information, notes, and details about development tools, build procedures, packaging, etc. In 2010, the Windows build was converted to use Microsoft Visual Studio 2010 and the Unix/Linux version was converted to use GNU Autotools. Also, some work was done on MinGW and Mac OS X/Darwin X11 builds. A big reason for creating this document was to organize the important details about all that in one place.

The source for this document uses AsciiDoc. Briefly, AsciiDoc is a frontend for a variety of XSL document programs and runs on Linux. Linux distributions have it in their package managers. Running it on Windows apparently requires using Cygwin, unfortunately. In some ways, it is "DocBook-for-Dummies" and is relatively easy to edit.

The home page for AsciiDoc is http://www.methods.co.nz/asciidoc/.

The SourceForge repository is http://sourceforge.net/projects/asciidoc/.

In the *alienarena/docs* directory these are the commands used to generate output.

text
: a2x --format=text --destination-dir=. docsrc/devref-en.asciidoc

html
: a2x --format=xhtml --destination-dir=. --stylesheet=*CSSPATH* docsrc/devref-en.asciidoc

pdf
: a2x --format=pdf --fop --destination-dir=. --stylesheet=*CSSPATH* docsrc/devref-en.asciidoc

Where *CSSPATH* is something like:

/usr/share/asciidoc/stylesheets/docbook-xsl.css

2. Subversion Repository

Alien Arena resources are stored in a Subversion (SVN) repository at http://icculus.org. To browse the repository, go to http://svn.icculus.org/alienarena/trunk. Use a Subversion client program for downloading. Read access to the repository is not restricted. Irritant administers commit access.

Caution

The SVN versions of the *.sln* and *.vcxproj* files are official. Do not commit these without clearance from Irritant. Doing so could cause confusion and consternation.

3. Visual C++ 2010 Build

Steps for building Release version with Visual C:

- Install TortoiseSVN.
- Checkout Alien Arena from SVN.
- Install and register Microsoft Visual C++ Express 2010.
- Install the OpenAL SDK.
- Install the libraries from the *.zip* files.
- Install the libcurl patch.
- Run *premake* to generate *config.h* for libode.
- Open the *.sln* file in Visual Studio.
- Select the Release build and run it.

- Copy *VS2010\bin\Release\crx.exe* to the top directory.

  Note

  For the libode *config.h* (and maybe some other files), an alternative is to get a copy from someone who has a working installation.

3.1. Windows Subversion Client

Install TortoiseSVN, the free Subversion client from Tigris. The website is http://tortoisesvn.net/

To download Alien Arena from the SVN Repository:

- In Windows Explorer, go to the *Computer # C: folder*.
- Right click in the window and select *SVN Checkout* in the context menu.
- For the *URL of repository*, enter: *svn://icculus.org/alienarena/trunk*.
- For the *Checkout directory*, enter: *C:\alienarena_w32*.
- Leave the *Checkout Depth* at *Fully Recursive*, *Omit Externals* unchecked, and *Revision* at *Head Revision*.
- Click *OK*, and let TortoiseSVN do its thing.

3.2. Microsoft Visual C++ 2010 Express

Microsoft Visual C++ 2010 Express is part of Microsoft Visual Studio 2010 and is provided free of charge. Registration is required. Download Visual C++ 2010 Express from the Microsoft website (http://www.microsoft.com/) and install it using the default setups.

Running programs compiled with VC++ 2010 requires the redistributable dll, *msvcr100.dll*. Some systems may not have it installed; a copy is provided in SVN.

3.3. Libraries

Extract the libraries in *lib_zipfiles* to *VS2010\libsrc*.

Patch for curl-7.20.1.

Unzip *libcurl_patch.zip* and replace *curl-7.20.1/lib/setup_once.h*. The original file, renamed *setup_once_unpatched.h*, is included in the *.zip* for reference.

Using premake for ODE config

A version of premake that supports VC++ 2010 is available at http://industriousone.com/premake.

(TODO: add instructions for running *premake4* with the proper options).

Some commentary on libraries

There is a rationale for keeping the libraries in the original *.zip* archives in SVN, rather than extracting them into SVN. For one thing, they are large and it would add a lot of clutter to the repository. Also, the build is setup to allow for multiple co-existing versions when libraries are updated. It would be good to be able to bring in a new version's *.zip* into SVN and get it working before making the switch in the official build.

All the libraries supply build procedures for a variety of environments. But, most did not have VC10 builds at the time they were setup. The solution was to create projects in Visual Studio for each of them. This has advantages, but is more complicated than just using a supplied build. It does take some work to get the build details correct and consistent.

3.4. OpenAL 1.1 SDK

Download the OpenAL SDK from OpenAL.org: http://www.openal.org/. (Redirects to the CreativeLabs site.) In the Download section, get the *OpenAL 1.1 Core SDK*. Install it in the default location; the build procedure is setup to look for it there.

Run the *oalinst.exe* program to setup the OpenAL driver, if you have not done that previously.

The *oalinst.exe* program installs whatever is required for OpenAL to use a particular systems sound hardware. It is included in the Alien Arena Windows distribution and is in SVN.

3.5. Directory Tree

Here are the directories and files related to the Windows build.

alienarena_w32\

alienarena_w32.sln
: Solution file. Mostly just a list of the Projects in VS2010\.

alienarena_w32.props
: Project property file. Generated and editted with the Visual Studio *Property Manager*. It is fairly easy, and sometimes more convenient, to manually edit it. Projects can inherit compiler options, etc. from here; which makes it much easier to make changes that affect all projects.

msvcr100.dll
: Redistributable for Visual C++ 2010.

oalinst.exe
: OpenAL installer.

vs2010\
: Build directory for Windows version of Alien Arena.

  Project Files
  : Project files. Generated and editted with Visual Studio. Somewhat easy to manually edit.

    - crx.vcxproj
    - game.vcxproj

- freetype.vcxproj
- libcurl.vcxproj
- libjpeg.vcxproj
- libode.vcxproj
- libogg.vcxproj
- libopcode.vcxproj
- libvorbis.vcxproj
- libvorbisfile.vcxproj
- zlib.vcxproj

bin\
: Target directory. The *crx.exe* executable is placed here. There are *RELEASE* and *DEBUG* subdirectories created by the build procedure.

lib\
: Target directory. The game and other libraries are placed here. There are *RELEASE* and *DEBUG* subdirectories created by the build procedure.

libsrc\
: Source directory for libraries.

include\
: Header files. The build procedure copies header files from the *libsrc* subdirectories to these directories for most of the libraries.

- config.h — *config.h* for *crx.exe* and *game.lib*.
- curl\
- jpeg\
- ode\
- ogg\
- vorbis\
- freetype-config\
  - ftconfig.h
  - ftheader.h
  - ftmodule.h
  - ftoption.h

- ftstdlib.h

The build procedure populates the *include* directories for the various libraries. Makes the build more complicated, but should make it easier to work with a new library version in parallel with a current one.

3.6. The config.h file

The *config.h* file for *crx.exe* and *game.lib* is an artifact from the Unix/Linux build. In Unix/Linux, *config.h* is generated by the *configure* procedure. For Windows, it is manually editted.

4. Unix/Linux Build

4.1. Introduction

Beginning in August of 2010 the Unix/Linux build for Alien Arena was converted from a Makefile to Autotools. There are two main goals for the implementation: One, to improve support for the various Linux distributions' package maintainence process. The other, to improve portability and ease of installation for users.

It is important to recognize that the goal of GNU Autotools is NOT to make the developer’s job easier. From a coder’s viewpoint, it is not, nor is it intended to be, the ideal software construction tool. There are alternatives, but for open source software, none have developed to the point of being generally adopted as standard.

4.2. Program Changes in 7.50

Previous versions had support for configurable *DATADIR* and *LIBDIR* installation paths. However, the implementation was not complete; due largely to the file system organization inherited from the Quake source. The installation procedure now uses configure arguments for determining installation paths.

Linux distributions install the game in the */usr* hierarchy in conformance with Unix file system standards. The */usr* hierarchy requires root privileges for writing, of course. While the game already had support for a user writeable directory, *.codered/*, in the users home directory, it did not implement a place to write bot information. Therefore, support for a *botinfo* directory in the user’s home directory was added.

To simplify the installation and to avoid the complications of using a shared library, the game module shared library, *game.so*, is now a static library. As a result, the *LIBDIR* installation variable is eliminated. Internally, the interface to the game module retains the dynamic link structure. Thus, it is still possible, in theory, to have a dynamic *game.so* that overrides the statically linked version. This is not tested nor supported in version 7.50.

Two new functions were added to the server/game interface for file system access. The location of game data files depends on how the game is installed. The code should not make assumptions about the base path since it might be affected by "prefix" and "datadir" configuration options. The game module will now determine all file paths using those functions (eliminating problems where the code assumed the legacy Quake style file system). The additional functions make it possible to place a *botinfo* directory, in user-writeable file space. The two new functions supply the full filesystem path when given a relative path. The functions are **FullPath()** and **FullWritePath()**. They are declared in *game/game.h* and *qcommon/qcommon.h*. They are defined in *qcommon/files.c*.

The name of the stand-alone dedicated server is changed from *crded* to *crx-ded*. The purpose is to make it simple to change the name of the executable files using the standard built-in configure option.

4.3. Subversion Repository

Linux Subversion Client

Install the free Apache (formerly Tigris) Subversion client using your package manager. Or download it from the website at http://subversion.apache.org/ .

To download Alien Arena from the SVN Repository:

- Create a *'PATH'* for installation somewhere in your home directory.
- Run: *svn checkout svn://icculus.org/alienarena/trunk PATH*.

Autotools Related Files

Warning

Certain files necessary for non-developer users to build the program are in the repository. These include the generated configure script and Autotools-supplied scripts in the *config/* and *m4/* subdirectory. Normally, these scripts are automatically added to the distribution package by Autotools. Having them in the repository can lead to version mismatch problems with a developer's Autotools version.

SOURCE FILES:

```
configure.ac ......... source file for autoconf
Makefile.am .......... top source file for automake
game_data.am ......... included in Makefile.am
source/Makefile.am ... subdirectory source file for automake
```

GENERATED FILES:

```
configure ............ the user's configure script
Makefile.in .......... template for the user's top directory Makefile
source/Makefile.in ... template for the user's subdirectory Makefile
aclocal.m4 ........... generated by aclocal
config/config.h.in ... template for the user's config.h
```

COPIED FILES:

```
m4/*.m4 .............. ax_* m4 macros, from autoconf-archive
INSTALL .............. GNU build/install reference
COPYING .............. GNU GPL
```

Comments on game_data.am

The file, *game_data.am* contains a **long** list of game resources. It is used to generate the distribution archive and for copying files to the *datadir* in a standard install. The downside is that it needs to be kept up-to-date with additions and deletions. The advantage is that the *install* program handles the copying of game resources. For *make uninstall* it handles deletion of what *make install* installed.

For development, when using the standard install, use *make install-exec* to limit copying to the executables.

4.4. Configure options

Run `./configure --help` to see all *configure* options. These are *configure* options added for Alien Arena.

--disable-client
: build dedicated server only, not the client (default:no)

--enable-alternate-install
: traditional single directory, in-place installation (default:no)

--enable-ansi-color
: ANSI terminal color (default: no) Added in 7.51.

--enable-debugging-symbols
: Compile Alien Arena with full debugging information (default:no)

--enable-full-warnings
: Display all compiler warnings (default:no)

--enable-irc-debugging
: Debug IRC messages (default: no)

--enable-buffer-debugging
: Debug buffers (default: no)

--enable-paranoid-io
: Activate paranoid, speed-sapping message checking (default: no)

--enable-assert-statements
: Enable debug assert() statements (default: no)

--disable-build-status
: hide the status message at the end of the configuration script (default:no)

--with(out)-xf86vm
: include XF86 VidMode support (default: check) Note: Needed to support full-screen mode. Note: If *with* is specified, then the library is required.

--with(out)-xf86dga
: include XF86 DGA support (default: check) Note: DGA appears to only used for mouse pointer input and does not appear to be required. It is disabled when Xxf86dga is not installed, *without_xf86dga* is specified, or when the cvar, *in_dgamouse*, is set to 0. Note: If *with* is specified, then the library is required.

4.5. GCC Compiler Options

Linux distro’s use a variety of different gcc compile options. These can give more error warnings; and in rare cases program errors.

Also for users who compile from source, there are CPU dependendent options that might give better performance. Might be good to document some of these in the README.

Wall , Wextra
: With -Wextra many warnings are produced, especially for unused parameters and signed/unsigned comparisons. With just -Wall, a few unused variable warnings are produced, which are easily fixed. Some bogus uninitialized warnings are produced; it is possible, but annoying, to eliminate these.

O2 , O3
: O3 produces different warning messages.

m32 , m64
: TBD.

march , mtune
: Set for specific architectures. Might give better performance than generic builds. For instance, `-march=core2` enables higher performance features.

mfpmath
: Testing shows that `-mfpmath=sse` gives better performance.

Wp
: TBD.

FORTIFY_SOURCE
: From fedora: `-Wp,D_FORTIFY_SOURCE=2` TBD.

fexceptions
: TBD.

fstack-protector
: TBD.

param
: From fedora: `--param=ssp-buffer-size=4`. TBD.

fasynchronous-unwind-tables
: From fedora. TBD.

4.6. Generating an Archive Package for Distribution

Verify Autotools Versions

Verify that your versions of *autoconf*, *automake*, and *autoconf-archive* are current.

The 7.50 release uses *autoconf 2.65*, *automake 1.11* and *autoconf-archive 2010.07.06*

Verify Alien Arena Version

Verify that the version in *configure.ac* is correct.

Export from Subversion Repository

Export from the SVN trunk to a directory for the distribution build.

Verify that the files in *config/* and *m4/* subdirectories are current. One way to do this is to rename the *config/* and *m4/* directories and then run:

```
aclocal --force --install -I m4
```

```
autoreconf --force --install
```

Compare the files in the new *config/* and *m4/* with the previous ones. If they match, proceed. If not, then the files in SVN need to be updated. The cleanest thing to do would be to update SVN and do a new export.

Generate the Distribution Archive

Run this command to build and verify the distribution package:

```
make distcheck
```

If this succeeds, the alienarena-7.50.tar.gz has been created.

Test the Distribution Package

In a test directory, extract the distribution archive. Run the configure, make, make install sequence. Use the --prefix configure option to install into a test directory.

To test the dedicated server only build, create a subdirectory and build with an alternative test directory. In the subdirectory, *configure* is invoked with *../configure*.

For completeness, the alternate install should also be tested.

Example

An example of distribution package generation:

- With a development directory in *$HOME/alienarena.*
- Using standard install into */usr/local*.

```
$ cd ~
$ mkdir aadist
$ cd $HOME/alienarena
$ svn export . $HOME/aadist/aaexport
$ cd $HOME/aadist/aaexport
$ ./configure
$ make distcheck
$ cd ..
$ mkdir aainstall
$ cd aainstall
$ cp ../aaexport/alienarena-7.50.tar.gz .
```

```
$ tar -xzf alienarena-7.50.tar.gz
$ cd alienarena-7.50
$ ./configure
$ make
$ sudo make install
$ cd ~
$ crx
```

BASH script for distribution tarball generation

This script uses several SVN *export* commands to retrieve the subset of files needed for the release tarball. It then invokes *make dist-check* to build the tarball. Run the script in a RELEASE directory.

The following commands generate the checksums and then run a *diff* to check for possibly missing files.

In the RELEASE directory:

```
$ mv alienarena/alienarena-<version>.tar.gz .
$ md5sum alienarena-<version>.tar.gz >md5
$ sha1sum alienarena-<version>.tar.gz >sha1
$ tar -xzf alienarena-<version>.tar.gz
$ diff -r -q alienarena alienarena-<version>
```

The Script:

```
#!/bin/bash
#
# Alien Arena release tarball generation
#
# --- top level ---
svn export --ignore-externals --non-recursive \
 svn://icculus.org/alienarena/trunk alienarena
###
# --- config/ ---
svn export --ignore-externals \
 svn://icculus.org/alienarena/trunk/config alienarena/config
###
# --- m4/ ---
svn export --ignore-externals \
 svn://icculus.org/alienarena/trunk/m4 alienarena/m4
###
# --- source/ ---
svn export --ignore-externals \
 svn://icculus.org/alienarena/trunk/source alienarena/source
###
# --- docs/ ---
svn export --ignore-externals \
 svn://icculus.org/alienarena/trunk/docs alienarena/docs
###
# --- Tools/ (fuse.tar.gz only) ---
svn export --ignore-externals --non-recursive \
 svn://icculus.org/alienarena/trunk/Tools alienarena/Tools
###
# --- Tools/LinuxScripts/ ---
svn export --ignore-externals \
 svn://icculus.org/alienarena/trunk/Tools/LinuxScripts alienarena/Tools/LinuxScripts
###
```

```
# --- arena/ ---
svn export --ignore-externals \
 svn://icculus.org/alienarena/trunk/arena alienarena/arena
 ###
# --- botinfo/ ---
svn export --ignore-externals \
 svn://icculus.org/alienarena/trunk/botinfo alienarena/botinfo
###
# --- data1/ ---
svn export --ignore-externals \
  svn://icculus.org/alienarena/trunk/data1 alienarena/data1
###
# --- tarball construction ---
cd alienarena
./configure --enable-maintainer-mode
 make distcheck
```

4.7. config.h

The config.h file is auto-generated for configurable builds (e.g Linux). It is manually edited for non-configurable builds (e.g. Windows). The *configure.ac* file does generate Windows related definitions in *config.h* so it can be used as a basis for the Windows *config.h*.

Rather than use symbols built into the compiler these symbols (and some others related to targets) are defined: *WIN32_VARIANT* and *UNIX_VARIANT*. This should make it easier to keep system dependent variations organized. It is a good idea, of course, to minimize system dependent conditional compilation in the common code.

4.8. Open Dynamics Engine (ODE) Integration

A static library, *libode.a*, is built and linked with the main program, *crx*. Only files that are used in the library build are included in the source tree in SVN and in the distribution. There is only one *config.h* and it includes what is needed for building *libode.a*.

4.9. Reading List for Autotools

- John Calcote. *Autotools: A Practitioner's Guild to GNU Autoconf, Automake, and LibTool.*
- Diego E. "Flameeyes" Petteno. *Autotools Mythbuster* http://www.flameeyes.eu/autotools-mythbuster/.
- David MacKenzie, Ben Elliston, Akim Demaille. *Autoconf: Creating Automatic Configuration Scripts* For version 2.65, 4 November 2009.
- David MacKenzie, Tom Tromey, Alexandre Duret-Lutz. *GNU Automake*. For version 1.11.1, 8 December 2009.
- Richard M. Stallman, Roland McGrath, Paul D. Smith. *GNU Make: A Program for Directing Recompilation*. GNU make Version 3.81, April 2006.
- Gordon Matzigkeit, Alexandre Oliva, Thomas Tanner, Gary V. Vaughan. *GNU Libtool*. For version 2.2.6, 1 August 2008.

- Richard Stallman, et al. *GNU Coding Standards*. last updated June 10, 2008.
- Edited by Rusty Russell, Daniel Quinlan, Christopher Yeoh. *Filesystem Hierarchy Standard* Filesystem Hierarchy Standard Group.
- Gary V. Vaughan, Ben Elliston, Tom Tromey and Ian Lance Taylor. *GNU AutoConf, AutoMake, and LibTool* "The Goat Book", Version 1.5, February 2006. http://sourceware.org/autobook.

5. MinGW Build

Experimentally, Alien Arena has been built using MinGW using the Autotools build. In the future, details and procedures may appear here.

6. Mac OS X / Darwin X11 Build

The Autotools build has experimental support for Mac OS X using the Darwin X11 environment. There are reports of some progress with this using MacPorts and Homebrew.

In February 2011, a native port to Mac OS X was developed. Details and test results to be reported as they become available.

7. Developer’s Not-Autotools Makefile

In the *docs/* directory, the file, *dev-makefile*, is a *Makefile* intended to support experimental builds. Its primary purpose is to allow custom builds with additional or different sources without having to hack the Autotools build. Documentation on how to customize it is included within the *dev-makefile* file.

It is simpler to make a full copy of the trunk sources, but that is not necessary. It takes some trickiness setting up *vpaths* and customizing *-I* options, but it is possible to do an alternate build with a small subset of sources in an alternate source subdirectory. Good for experimenting with optimizations.