1 Known issues

No: 1	**Short desc.:** disadvantages of using cxxtestgen as prebuild command
Description	Using cxxtestgen as prebuild commands was made the default behaviour for all environments except gmake and vs2010. Prebuildcommands will always run when the test-project rebuilds and will cause all the test-cpp-files to be recreated and rebuilt. The problem is that rebuild times for the test-project are increased. Currently it's only a few seconds, but it's bad anyway. For vs2010, only test_root.cpp will be recreated and rebuilt, so the loss of time is much smaller.
Workaround(s)	You either accept increased rebuild times or choose an environment which isn't affected or less affected (Makefiles or VS2010).
Solution(s)	Each environment must be customized to meet the requirements. I didn't do it for all environments to reduce the amount of work. On VS2010 I build the sourcefiles when the corresponding headerfiles change. Because test_root.cpp has no corresponding headerfile, I had to use prebuild commands instead. Any better ideas? The approach of the premake3 could be worth looking at, but I guess it won't bring a perfect solution either.
Status Premake3	The Premake3 scripts used a dedicated testgen project. I didn't yet look at this in all the details.

No: 2	**Short desc.:** Makefiles not exchangeable between OSX and Linux
Description	0ad needs the options “start-group” and “end-group” to compile successfully on Linux while they aren't supported an not needed on OSX. Premake had to be customized so that it includes these options on Linux but not on OSX. The problem is that this check happens during build-time of the makefiles and not during runtime. If you build a makefile on Linux and try to use it on OSX, it will fail.
Workaround(s)	When distributing a new customized version of premake4, you have to either create the makefile on OSX or remove the start-group and end-group flags manually (premake doesn't need them). Linux: LINKCMD = $(CC) -o $(TARGET) $(OBJECTS) $(LDFLAGS) $(RESOURCES) $(ARCH) -Xlinker --start-group $(LDDEPS) -Xlinker --end-group $(LIBS) OSX: LINKCMD = $(CC) -o $(TARGET) $(OBJECTS) $(LDFLAGS) $(RESOURCES) $(ARCH) $(LDDEPS) $(LIBS)

	For the makefiles of 0ad there's no workaround. You simply can't exchange them between OSX and Linux.
Solution(s)	Probably the makefile could detect the operating system and set the flags at runtime.
Status Premake3	Also affected.

No: 3	**Short desc.:** *.app on OSX
Description	OSX automatically creates a *.app "package" instead of a plain executable. Because some of the libraries are just referred to by their name they are expected to be in the same directory as the executable.
Workaround(s)	The easiest workaround is just copying the plain executable from the *.app package/directory to binaries/system.
Solution(s)	Solutions for the future could involve changing install names of libraries and copying libraries and contents inside the *.app package.
Status Premake3	Also affected.

No: 4	**Short desc.:** Linking fcollada with xcode
Description	Xcode creates a dylib for fcollada and the current code doesn't support loading dylibs.
Workaround(s)	Build foclllada using makefiles.
Solution(s)	I didn't find a way to let Xcode create *.so libraries instead of dylibs, but probably there is one. The other possible solution would be extending the current code to support loading dylibs.
Status Premake3	Not affected because Xcode was not supported.