WELCOME TO
THE WONDERFUL WORLD OF EAMON CS

[Note: this document is a work in progress]

Now that work on the Eamon CS game engine has been completed, it's time to turn to the documentation set once again. This document is called WorkingDraft, but in fact it will be a giant brain-dump of information – everything I can think of will go in here, complete with screen shots, diagrams, anecdotes, samples, etc.

Please be aware that it will be stream-of-consciousness with little or no attention payed to grammatical correctness, sentence/paragraph structure, or clever segues between topics. It will also be lacking a formality found in most polished documentation sets. You may find that topical material has been duplicated or sprawled across different sections. The scope being discussed may switch abruptly, from wide-angle flyover of the terrain down into the dirt at the source code level, then back again.

The hope is that you will get the information conveyed and the fit and finish can be left for later when all the factual bits are written down. To allow you to keep up with the progress of the document, which may grow quite large, I will try to avoid making changes within it to already-written text and instead append all new changes to the end with appropriate datestamps.

I'll start off by adding the text of the now-superseded README.htm document:

This minimal document is intended to describe the various facets of Eamon CS. The real documentation will be a formal set: full source code commenting, a Player's Manual, a Dungeon Designer's Manual and a Class Reference. It will take quite a while to put all that together, so in the mean time, this document will have to suffice.

Eamon CS (ECS) is a port of the Eamon roleplaying game to the C# programming language. It is the production version of a prior system called Eamon AC (EAC), an ANSI C prototype intended to extract Eamon out of BASIC. If you have EAC on your system you should uninstall it before using ECS, as it is now obsolete.

This system is a hybrid of Donald Brown's Classic Eamon and the most modern BASIC Eamon available, Frank Black Production's Eamon Deluxe. It is also directly based on the EAC prototype. Eamon CS borrows liberally from and bears a strong resemblance to all these sources in various areas.

The game has evolved rapidly over its short lifespan, first making the leap to a plugin-based architecture unique to this branch of Eamon, then into the various flavors of Unix, and most recently into the Android mobile device space. Along the way, a small but growing collection of custom-built adventures has been created, and the game engine has improved with each one produced. The different facets of Eamon CS share a common code base, but where necessary it will be distinguished as Eamon CS Desktop for traditional workstations and Eamon CS Mobile for mobile devices.

PROGRAMS, LIBRARIES AND PLUGINS

The plugin-based architecture used by Eamon CS extends to the Main Hall, the Dungeon Designer and all adventures. They are managed by a Plugin Manager program specific to either Desktop or Mobile environments:

System\Bin\EamonPM.WindowsUnix.dll	Eamon CS Windows/Unix Plugin Manager
System\Bin\EamonPM.Android-Signed.apk	Eamon CS Android Plugin Manager
System\Bin\EamonDD.dll	Eamon CS Dungeon Designer Plugin
System\Bin\EamonMH.dll	Eamon CS Main Hall Plugin
System\Bin\EamonRT.dll	Eamon CS Adventure Plugin
System\Bin\Eamon.dll	Eamon CS Library
System\Bin\Polenter.SharpSerializer.dll	SharpSerializer.Library

Additionally you have some adventures:

System\Bin\TheBeginnersCave.dll	Eamon CS Adventure Plugin
System\Bin\BeginnersForest.dll	Eamon CS Adventure Plugin
System\Bin\TheTrainingGround.dll	Eamon CS Adventure Plugin
System\Bin\TheSubAquanLaboratory.dll	Eamon CS Adventure Plugin
System\Bin\ARuncibleCargo.dll	Eamon CS Adventure Plugin
System\Bin\StrongholdOfKahrDur.dll	Eamon CS Adventure Plugin
System\Bin\TheTempleOfNgurct.dll	Eamon CS Adventure Plugin
System\Bin\WrenholdsSecretVigil.dll	Eamon CS Adventure Plugin

Conceptually, each plugin can be thought of as a discrete program; it exposes a Program class with a Main method that takes a collection of arguments, much like any C-based program. The difference is that the Plugin Manager is what "executes" the plugin, not the calling C# environment. At the bottom of the software stack, you have Polenter.SharpSerializer.dll which handles loading and saving of the game's textfiles and Eamon.dll which holds code common to all ECS plugins. The EamonRT.dll contains the vanilla game engine used by all non-customized adventures. For customized adventures, the game plugin (eg, TheBeginnersCave.dll) contains custom code specific to that game, built by leveraging EamonRT.dll. It is a "modded" version of the game engine.

This implementation allows the logic for any plugin to be shared with any other plugin. For example, an interesting idea would be to create a "campaign library" that contains common code for multiple derived adventures.

The source code for Plugin Managers and system plugins resides in the appropriate directory under System; for adventures, it resides in the adventure-specific directory under Adventures.

The Main Hall textfiles reside in System\Bin, making them easily accessible to all plugins, while game-specific textfiles (both original and save game) reside under Adventures in their respective game directories.

A final note on the architectural differences between ECS Desktop and Mobile. For Desktop, the plugins reside in the System\Bin directory and are loaded by EamonPM.WindowsUnix.dll only when needed (and reused once loaded). In contrast, when EamonPM.Android-Signed.apk is built, all plugins are statically linked in and all textfiles are embedded, producing a monolithic application. When the .apk is delivered onto the mobile device, only the textfiles are replicated (when appropriate) to the device's file system - the plugins remain part of the application. In spite of this, the plugin managers are very similar internally.

BATCH FILES

The plugins take a variety of command line parameters (which will be described below). However, to get you up and running quickly, there is a QuickLaunch folder (since there is no formal ECS installer, you may want to manually create a shortcut to it on your desktop). Inside this folder is a set of batch files that can be run directly. The batch files are organized into sub-folders based on the underlying plugin they invoke:

EamonDD\EditAdventures.bat	Edit the flat Adventures database
EamonDD\EditCatalog.bat	Edit the adventure category Catalog
EamonDD\EditCharacters.bat	Edit the Characters file
EamonDD\EditContemporary.bat	Edit the Contemporary adventures category
EamonDD\EditFantasy.bat	Edit the Fantasy adventures category
EamonDD\EditSciFi.bat	Edit the Sci-Fi adventures category
EamonDD\EditTest.bat	Edit the Test adventures category
EamonDD\EditWorkbench.bat	Edit the Developer's Workbench
EamonDD\EditWorkInProgress.bat	Edit the Work-In-Progress adventures category
EamonDD\EditARuncibleCargo.bat	Edit A Runcible Cargo
EamonDD\EditBeginnersForest.bat	Edit Beginner's Forest
EamonDD\Edit[AdventureName].bat	Edit [AdventureName]
EamonMH\EnterMainHallUsingAdventures.bat	Enter the Main Hall using a flat adventure database[1]
EamonMH\EnterMainHallUsingCatalog.bat	Enter the Main Hall using a hierarchical adventure database[1]
EamonRT\ResumeARuncibleCargo.bat	Resume A Runcible Cargo[2]
EamonRT\ResumeBeginnersForest.bat	Resume Beginner's Forest[2]
EamonRT\Resume[AdventureName].bat	Resume [AdventureName][2]

[1]Run these batch files to create a new character or send an existing one into the Main Hall. The only difference between the two batch files is the nature of the adventure database loaded (the same characters will be available regardless).

[2]Run these batch files to return to an in-progress adventure.

You can study the batch files to see how various programs are launched; you can also create your own batch files using these as templates if you decide to try your hand at adventure writing, or if you want to run the system in a non-default manner.

Eamon CS Mobile mirrors this QuickLaunch hierarchy using a series of ListViews to provide a similar experience.

QUICK ARCHITECTURAL OVERVIEW

The entire Eamon CS system was written from scratch and aggressively exploits the C# language. This is no port, more like an expansive toolkit used to build games based on the Eamon ruleset. It has more in common with traditional Interactive Fiction systems like TADS or Inform. Eamon CS games are built by subclassing existing classes (in any library/plugin, but especially Eamon.dll or EamonRT.dll) and overriding their behavior, calling back into base classes where the default behavior is needed. Every class in the system can be subclassed, and generic improvements made to individual adventures can be pushed back into the base framework for use by all games. As time goes on, the system will grow in flexibility and power.

The textfile format used by Eamon CS deserves an explanation. The game produces ASCII XML files that represent C# object graphs. This is radically different than any other Eamon, and plays a critical role in the development of customized adventures. Take a look in the MONSTERS.XML file in Test Adventure (a non-customized adventure) and you will see the serialized class is Eamon.Game.Monster (from Eamon.dll). But in The Beginner's Cave, you will see it's TheBeginnersCave.Game.Monster (from TheBeginnersCave.dll). These are two different classes, the latter is a subclass of the former, and implements custom code that is used specifically by The Beginner's Cave. Now compare the batch file EditTestAdventure.bat, which contains the command line argument "-pfn EamonRT.dll" with EditTheBeginnersCave.bat, which contains "-pfn TheBeginnersCave.dll". In the former the flag instructs EamonDD to load the base engine, while in the latter it loads TheBeginnersCave.dll and uses any customized classes it finds (like the Monster class) when creating new objects. This whole topic is fairly complex and begs to be part of a formal document set, but at least this gives you an idea of how it works.

With BASIC Eamons that use files containing just data, you can bypass the Dungeon Designer program completely and build these files by hand. You should avoid this practice in Eamon CS due to the specialized textfile format. The EamonDD program is instrumental in producing textfiles of the correct format and you should rely on it to do so. However, you can still manually tweak the textfiles with your favorite editor. Just be very careful to ensure the file format is not violated, as the Polenter.SharpSerializer.dll library can be picky.

The EamonRT.dll base adventure runtime takes the form of a Finite State Machine (FSM). For those unfamiliar, this is a technique used to model complex program behavior. The main game loop is 15 lines long; all complex behavior has been delegated to subclasses of the State class. Player commands are derived from the Command class, which itself is derived from State. Customized adventures will almost always subclass various States or Commands to provide special behavior. It is also very easy to create new States or Commands and link them

into the FSM if needed. Finally, if you look through the code you will see that special care has been taken to avoid the use of GOTO except in specific circumstances (goto Cleanup). This gives the game engine a clean, deterministic quality.