|  |
| --- |
| Helix Productions Magtheridon |
| MeGUI Expansion (MGE) |
| Software Design Document |
|  |
| **Andrew Johnson** |
| **5/10/2010** |

|  |
| --- |
| The significance of this document is to provide an overview of the design and functionality of the MeGUI Expansion software. |

# Revision History

5 May 2010: Initial Version

Table of Contents

[Revision History 2](#_Toc261336321)

[Preface 4](#_Toc261336322)

[1. Introduction 5](#_Toc261336323)

[1.1 Goals and Requirements 5](#_Toc261336324)

[1.2 Acronyms and Terms 5](#_Toc261336325)

[2. Design Overview 6](#_Toc261336326)

[3. Functional Overview 7](#_Toc261336327)

[4. Technical Design 8](#_Toc261336328)

[5. General Issues 9](#_Toc261336329)

[6. Glossary 10](#_Toc261336330)

# Preface

This document provides an overview of the design of MeGUI Expansion (MGE) software systems. It is a living document that is expected to evolve throughout the development process. During conceptual design, it provides a broad overview of the design with detail to be added during subsequent design phases. The focus during conceptual design is on describing enough of the design to allow an examination of the design’s suitability in meeting the system requirements.

Remember, this software is ultimately meant to be aimed at “ease-of-use,” not high level system’s management. The MeGUI platform is an open-source project that is subject to change and can be discontinued at any given time. Furthermore, this project is aimed at rapid development, while maintaining a moderate level of quality and scalability.

This software project will be the second implementation of the batch-encoding problem associated with MeGUI. This project is meant to fix the shortcomings that plagued the last product, which were the lack of encoding profiles and the generalization of AviSynth settings. Not only are we trying to develop a software system that can be used to batch-encode videos for the iPod and iPhone platform, we are also developing a system that can batch-encode any video for any platform or need.

When developing this project, if several avenues to the solution present themselves, pick the one that is the easiest, simplest, and most effective one to develop. Keep in mind, though, that you will need to be able to scale these solutions in the future and that there is no substitute for good programming practices.

# 1. Introduction

## 1.1 Goals and Requirements

The model presented in this document is intended to address the following goals and requirements:

* The ability to take an arbitrary amount of digital video and audio files and encode them using any “encoding profile” and any AviSynth script
* Ease of use; less than 4 steps between start and finish for the user
* Robust error handling. MGE should be able to handle all errors and relay them to the user if necessary
* Reduced external processing
* Modularity between the functions of MGE

## 1.2 Acronyms and Terms

* Encoding Profile – The XML based profiles loaded with MeGUI that have predefined X264 encoder settings
* MGE – MeGUI Expansion

# 2. Design Overview

## 2.1 System Overview

The MGE software system is a compact and environmentally dependent client-side system, which requires several preexisting pieces in order to function properly. MGE is broken into three parts: Model, Viewer, Controller.

The three parts are housed inside of a package called “Engine,” which separates it from the two other packages called “objects” and “exceptions.” This allows the Model module of MGE to stay uncluttered.

## 2.2 Model

The Model consists of all the business logic and algorithms of MGE. Any and all calculations should be in the Model. Furthermore, the Model package should have all the object types, with the exception of the specific Listener objects.

## 2.3 Viewer

The Viewer is the graphical user interface of MGE. There should be no other logic within the Viewer, and all Viewer classes should have a method that allows the Controller to attach all the Listeners by passing in a List of Listeners.

All Viewer classes extend the Viewer abstract class. The Viewer abstract class implements the Runnable Interface, and must

## 2.4 Controller

The Controller is the overall mediator between the Viewer and the Model. It handles all user-events, errors, launches, shutdowns, and progress information concerning MGE. All information and requests need to go through the Controller. The Main Driver class should be the one to launch the Main Controller class, which is a special case.

# 3. Functional Overview

# 4. Technical Design

# 5. General Issues

# 6. Glossary