### Revision History

Version 1.0 Friday April 10th 2015

Introduction

Glossary

Application Overview

Application

Structure

Requirements Specifications

Version 1.1 Wednesday April 22nd 2015

Foreword

Table of Contents

Revision

History

Version 1.2 Friday April 24th 2015

Coding Style

Guide Update

TOC

Version 2.0 Sunday August 30th 2015

Complete document revision

Version 2.1 Saturday October 3rd 2015

General maintainance

Updated Glossary

### Table of Contents

Revision History 1

Table of Contents 2

Introduction 3

Glossary 3

Application Overview 4

Application Structure 5

Additional Dependencies 7

### Introduction

Our aim is to develop an interactive learning tool that visually demonstrates the behaviors and uses of key algorithms and data structures, herein referred to as Professor Alberton’s Algorithmic Adventures. This document will inform the developers and stakeholders about the applications goals, design and core requirements.

### Glossary

GUI (Graphical User Interface) –Application output as represented for the user.

MVC (Model View Controller) – An architectural pattern designed to separate the GUI from the data.

State Driven Design – The data the application software has access to at any given time is represented by a program state.

SFML – The Simple Fast Multimedia Library is a high level API for C++, which includes an OpenGL wrapper and provides a robust interface for graphics and audio programming.

Iterative Design – A methodology based on cyclic prototyping, testing and refining.

Notes on terminology - This document contains references to several commonly known algorithms and data structures and it is expected the reader has a reasonable understanding of what is meant by terms such as binary tree, heap, stack, Conway’s Game of Life and so on.

### Application Overview

Professor Alberton’s Algorithmic Adventures is an interactive demonstration and learning tool, targeted at secondary and tertiary students. Its primary objective is to demonstrate the functionality of key algorithms and data structures in a way that is entertaining and engaging.

Professor Alberton’s Algorithmic Adventures will have a menu driven interface via which the user will be able to visit a particular realm. A subsequent themed interface will lead them to a selection of several relevant algorithms or data structures, which can then be explored.

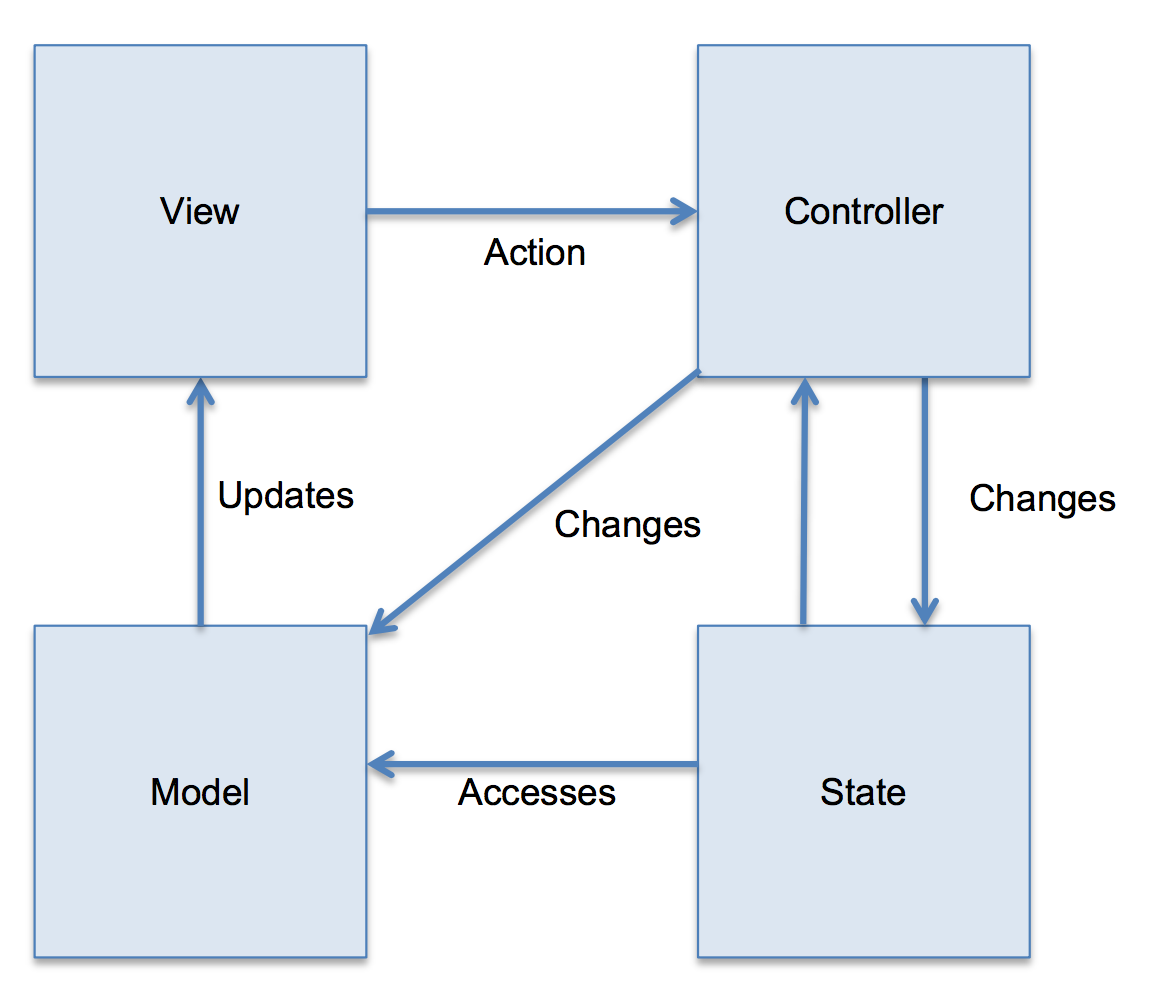
Each algorithm or data structure will have a background or history, followed by a visual demonstration with accompanying pseudo code, and finally a related game or interactive demonstration.

Games and interaction will make up the core focus of the application and will be aimed at developing some fundamentals skills required to fully understand the practical use of algorithms and data structures.

### Application Structure

The internal structure of the application will combine elements of MVC and state driven design. This will allow for prototyping design methodology, due in part to the separation of interface and data that MVC informs, and inherently modular nature of state driven architecture.

* The GUI will listen for actions, which are subsequently passed to and handled by the controller.
* The controller then changes the program state (for example from displaying menu to demonstrating Quick Sort), which then has exclusive access to the relevant data, which in turn updates the view and displays the output/listens for the next action.
* Actions don’t always lead to a change in program state. Inserting data into a binary tree for example, will update the model directly, rather than altering the program state.



## State:

#### Enum ProgramState { Unitialized,

#### ShowingSplash,

#### Paused,

#### ShowingMenu,

#### Running,

#### Exiting };

## View:

#### static sf::RenderWindow MainWindow;

## Controller (changing state):

#### if (currentEvent.type == sf::Event::Closed){

#### ProgramState = MainLoop::Exiting;

#### }

## Controller (changing model):

#### if (currentEvent.type == sf::Event::KeyPressed){

#### if(currentEvent.key.control){

#### std::cout << “The control key was pressed”;

#### }

#### }

## Model:

#### MainWindow.clear(sf::Color::Black);

### Additional Dependencies

Though the application will be stand-alone and programmed in C++/SFML/OpenGL, it will have dependencies within its own structure.

Professor Alberton will explain the history and context with the help of an animated chalk-board. Such draw-able objects will be shared by the different realms.

The same can be said for the animations of Professor Alberton himself, as the base animations will be the same across the application.

Such dependencies not only reduce the amount of duplicated code, but increase the modularity of the project, allowing changes to be made to shared assets quickly and without breaking existing code.