**Sinister Transistor**

**High Level Design**

**COP 4331, Spring 2016**

Team Name: The Mega Bytes

Team Members:

* Greg Kelso
* Mark Boutwell
* Joel Gardyasz

Modification history:

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Who | Comment |
| v0.0 | 02/18/16 | Greg | Template |
| v1.0 | 03/3/16 | Greg/Mark | Completed High Level Architecture diagram |
| v2.0 | 03/4/16 | Greg | Completed High Level Architecture table and Design Issues |

**Contents of this Document**

High-Level Architecture

Design Issues

**High-level Architecture**



**Interface Descriptions**

|  |  |
| --- | --- |
| **Name** | **Description** |
| User Input | This will store and send all input from the user (either mouse or keyboard input). |
| Character Controller | This will take input from the User Input interface, store and calculate data relevant to the character, and send data to the Map Model to update the character’s settings in the game. |
| NPC Controller | The NPC Controller will store and calculate data for all NPCs within the game, and send data to the Map Model to update the NPCs’ settings within the game. |
| System Controller | The system controller will handle overall settings such as the main menu, game settings, and player inventory. It will take input from the User Input interface and send data to the Display interface. |
| Map Model | The Map Model will take data from the Character Controller and NPC Controller interfaces, make necessary adjustments to the map and game world, and send those adjustments to the Display interface. |
| Display | This takes input from the System Controller and Map Model interfaces, and uses that data to update the game’s display that the user is able to view. This interface completes the Model-View-Controller architecture. |

**Design Issues**

**Reusability**

By combining the Model-View-Controller architecture with strong object-oriented principles, we will allow many aspects of this game to be reusable. By having an NPC Controller we can use the same classes and methods to create as many NPC as we would like. In addition to this, while the game is intended to be single-player, the Character Controller class would make it extremely simple to add another player to the game.

**Maintainability**

We have been maintaining the project on a private Github repository. All changes are logged and reverting the project back to any version is extremely easy to do. The strong object-oriented principles will also make the code readable and easy to find bugs.

**Testability**

Because each component is encapsulated, and our requirements are clearly defined, testing individual parts will be very easy. Unit testing will be done throughout the project and several integration tests will be completed near the end of the project.

**Performance**

As long as the user’s computer meets the Computer Specifications outlined in Section 42 of the Software Requirements Specification document, performance should not be an issue. The video game will feature simple graphics and will not be scaled to any level larger than a standard Unity video game.

**Portability**

Unity is able to port to multiple platforms, and we will be focusing on OS X, Windows, and Linux executables. However it is possible to export the game to mobile platforms as well.

**Safety**

The video game will not store any personal data from the user, and will be hosted from a secure site so that the user does not have to worry about damage to their Personal Identifiable Information (PII) or computer security.