**Kandy Dungeon Framework**  


Marquis Smith

5/18/2014

Framework Changes



Document Changes



Table of Contents

1. Introduction

Introduction

The application domain of the framework that we will be creating will be a board game. The board will be composed of customizable tiles that will be navigated by the user’s game character. When certain tiles are landed on, random events, created by the designer, will occur prompting the user for action. The user’s character will have persistent attributes and possibly items that can be modified throughout the course of the game.

**Example Application:**

An example application using our framework would be a game similar to Candy Land, called Escape from Kandy Dungeon. Using the framework described above, there will be customizable tiles that the user traverses with his game character depending on a randomly generated “dice throw”. The tile that the user lands on will initiate a randomly chosen event, good or bad. These events may manipulate the character’s attributes, initiate combat that causes the user to advance or retreat back to their previous tile, or alter the normal rules of the game. If the character lands on a space already inhabited by another player’s character, those characters will have to battle for control over that tile. The ultimate goal is to get to the end of the board alive before the other players. Several enter; only one player may leave the Kandy Dungeon.

Framework

**Overview:**

* A carefully written introduction paragraph(s) summarizing the section's contents and previewing the subsections that follow
* A description of the functionality that the framework provides
* A list of user stories or use cases for using the framework
* Each user story should have a 2-3 sentence description
* Each use case should have a scenario describing the user interaction
* Use-case diagrams and use case scenarios are an effective way to describe the interaction

Use Cases or User Stories:

**Architectural Design:**

**Board:**

The Board-object serves as the foundation of the framework. It contains a collection of Tile-objects that are used to control the progress of the players and determine the actions that take place upon each passing of a player’s turn. It also is responsible for parsing for the “win-state” of the game, eventually determining when to terminate further progress and end the game.

**Tile:**

The Tile-object serves as the primary means of progressing the flow of the game. They contain an instance of a character object for means of determining where each player of the game is located on the board. The Tile-objects also serve as the means for polling the collection of Event-objects to determine what actions are taken at the end of each turn.

**Character:**

The Character-object is used to track the state of each individual player of the game. Each instance of the Character class maintains its personal specifications (health, name, position). The properties of each character are used to determine the outcome of each “Event” that is triggered during the course of the game. At all points during the course of the game each Character-object is stored in one of each of the Tile-objects of the game board. This is what determines the location of each character at any given time.

**Event:**

The Event-object is used to manipulate the game environment. At any given moment the game board may invoke an Event-object, causing it to execute all of its specific actions. This involves the manipulation of Character-objects, such as moving them or otherwise altering their properties to another state. These events serve as the primary way to progress the state of the game until it reaches the “win-state” for one character.

**Design Patterns:**

Sequence Diagram 

**Software Design:**

• A carefully written introduction paragraph(s) summarizing the section's contents and previewing the subsections that follow

• Present a high-level view showing the static structure of the framework and the interactions between components

• Text descriptions that refer to graphical representations such as class and sequence diagrams that are included as figures in this section

**File Structure:**

**Software Requirements:**

**Example Application:**

• A carefully written introduction paragraph(s) previewing and summarizing the section's contents and its intended audience

• An explanation of how the administrative and user functions work from a non-technical point of view. In other words, provide a user manual for your software.

**User Interaction:**

* A list of user stories or use cases for the example application
* Each user story should have a 2-3 sentence description
* Each use case should have a scenario describing the user interaction
* Use-case diagrams and use case scenarios are an effective way to describe the interaction

**Software Design:**

• A carefully written introduction paragraph(s) summarizing the section's contents and previewing the subsections that follow

• Present a high-level view showing the static structure of the example application that uses the framework and the interactions between components

• Text descriptions that refer to graphical representations such as class and sequence diagrams that is included as figures in this section

**Quality Assurance:**

This section of the iteration report describes how your team will ensure that you are developing a high-quality software. You will describe how your team plans to produce high-quality iteration reports, what the coding standards for the project are, how you will ensure a useable interface, how the team will manage changes to project, and how the team plans to test the software.

**Coding Standards:**

* A carefully written introduction paragraph(s) summarizing the section's contents and previewing the subsections that follow
* Describe the standards and procedures in place to ensure consistent, correct, and useful code comments and code documentation
* Code examples are a good way to illustrate your standards

**Testing Process:**

* A carefully written introduction paragraph(s) summarizing the section's contents and previewing the subsections that follow
* Describe the process your team will follow for testing your software
* Include details on your team's methods for each of the following: unit testing, integration testing, and system testing. Do not include unit test code.
* Present this material in project-specific terms. Avoid stating this material in generic software engineering terms

**Conclusion:**

A short conclusion that summarizes what the problem being solved is and how you will be solving it. It signals to the reader that the document is ending.

**Retrospective:**

1. A review of the team’s activities for the iteration

2. What the team did well during the iteration

3. What the team needs to improve in the next iteration

4. Specific and concrete plans to address the identified improvements

**Appendices:**

**Honor Code:**