Kandy Dungeon Framework

Author: Wesley Kosierowski & the Kandymen

Revision History

Version number	Description of Change	Date	Editor
0	Proposal of Framework	4/25/2014	The KandyMen
0.1	Application Framework Design	5/4/2014	Wesley Kosierowski
0.2	Added Introduction	5/15/14	Wesley Kosierowski
0.3	Updated Use Cases	5/16/14	Wesley Kosierowski
0.4	Added Revisions Section	5/16/14	Wesley Kosierowski
0.5	Added Table of Contents	5/16/14	Wesley Kosierowski
0.6	Added intro for framework & Example Application	5/16/14	Wesley Kosierowski
0.7	Added Content to most sections	5/16/14	Wesley Kosierowski
0.9	Added new diagrams	5/18/14	Wesley Kosierowski
1.0	Updated Content of Framework	5/18/14	Wesley Kosierowski

Table of Contents

Introduction & Project Overview

In the situation of the Kandymen, we needed to create a framework for a certain type, or genre, of computer program. Our group decided to create a framework for a video game as we thought it would be fun, interesting, and educational. It was not important as to what kind of game our framework designed for, all that mattered was that a framework for some type of game was created. So, the Kandymen decided on creating a framework for a board-game style of video game, comparable to the children's game of Chutes and Ladders or other like games. The entire point was to create the basic design and functionality for the board-game, while leaving the more specific details up to the game developers.

For the Kandy Dungeon Framework, of the many important features the most essential one would be the board itself. Similar in concept to a checker board, the game-board is made up of many individual tiles, the number of which can be customized at the game developer's whim. These tiles, another important part, contain the location of everything that happens during the game, from the player, to any events that occur. The Player and the Event are also important. The player is represented as it moves around the board by dice roll, and at any given point, the Player could land on a tile and trigger an event. These events are the largest driving force during the game and dictate the biggest changes and challenges to the character. These events are to be left open for the game development team to use, as they can be customized depending on the style of game they are looking for.

For example, the Kandymen created a demonstration application to show some of the features the Kandy Dungeon Framework is capable of. This application features multiple characters that that take turns rolling a die to make their way through our board. Whichever player reaches the end of the board first wins.

Further on in this document the example application will be discussed in greater detail along with its class diagrams, as well as architectural design for the framework, class and sequence diagrams for the framework. The document will be capped off with a look back on what our team did well and what we could have improved on.

Framework

In the Kandy Dungeon Framework there are several important key sections, the first of these is the board. The board size can be set by height and width, and many of the essential parts of the framework are contained within the board as well. The methods that simulate a dice roll as well as the methods that control player turns and movement around the board are contained within. The Event class is also a very important part of the framework. In its pure form, there is not much information to the event class, but it is the driving force behind player interaction inside of the game. In addition to the Event, the Player is necessary in the framework as well. It contains information such as the name, as well as the player's actual location on the board. With methods setting and getting location, it's easy for the game developers to find the Player and use him on the board. Lastly, the Tile is an essential part of the framework as well. The tiles are the building blocks on which everything is built on, so it has information regarding direction of travel, player occupation, and if certain moves are allowed or not.

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.

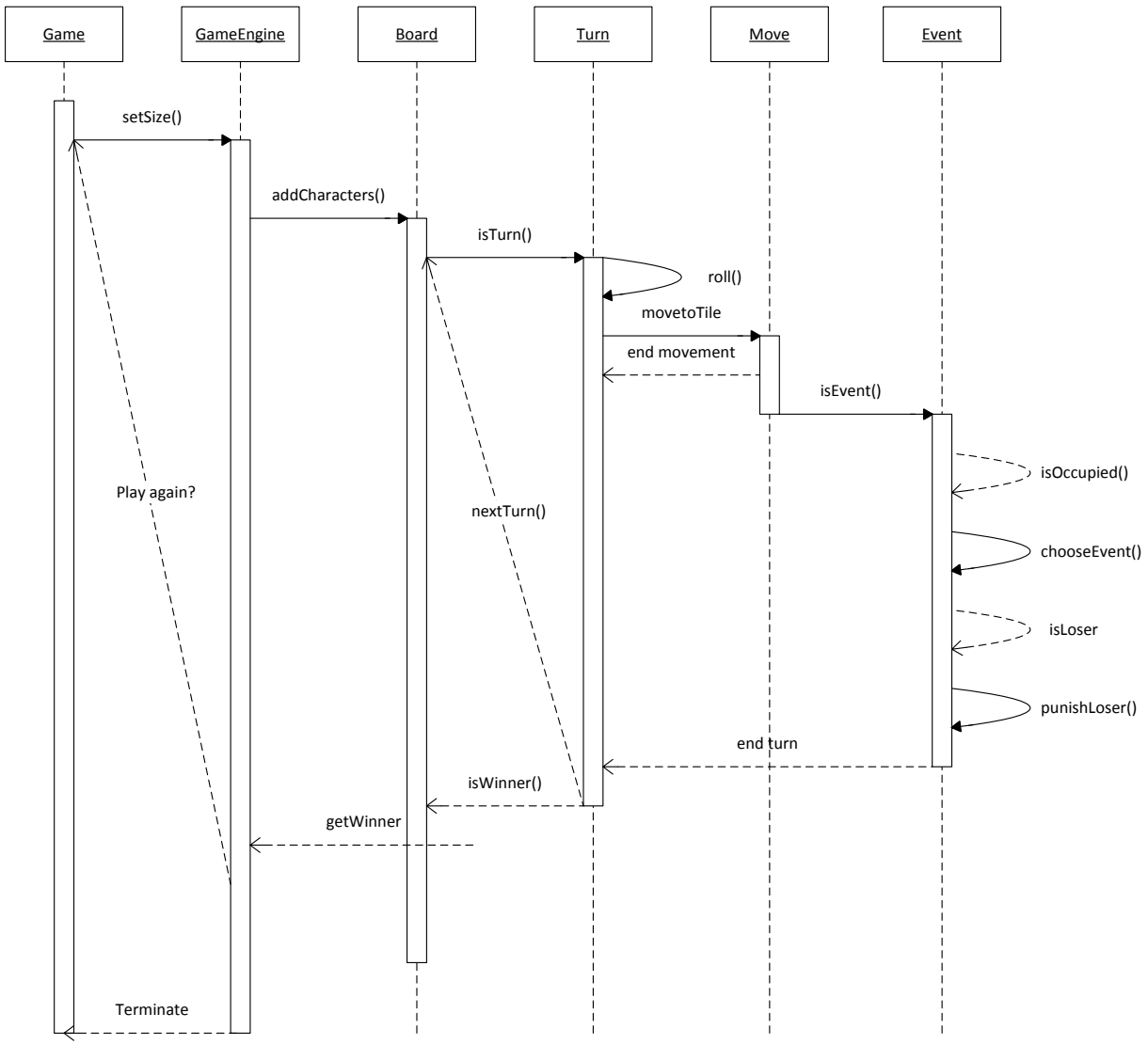
Player

The Player-object is used to track the state of each individual player of the game. Each instance of the Player 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 Player-object is stored in one of each of the Tile-objects of the gameboard. 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 Gameboard 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 event serve as the primary way to progress the state of the game until it reaches the “win-state” for one character.

Sequence Diagram



Framework Overview

This section will cover the functionality that the framework provides as well as use cases and a use case diagram.

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.

Player

The Player-object is used to track the state of each individual player of the game. Each instance of the Player 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 Player-object is stored in one of each of the Tile-objects of the gameboard. 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 Gameboard 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 event serve as the primary way to progress the state of the game until it reaches the "win-state" for one character

Framework Use Cases:

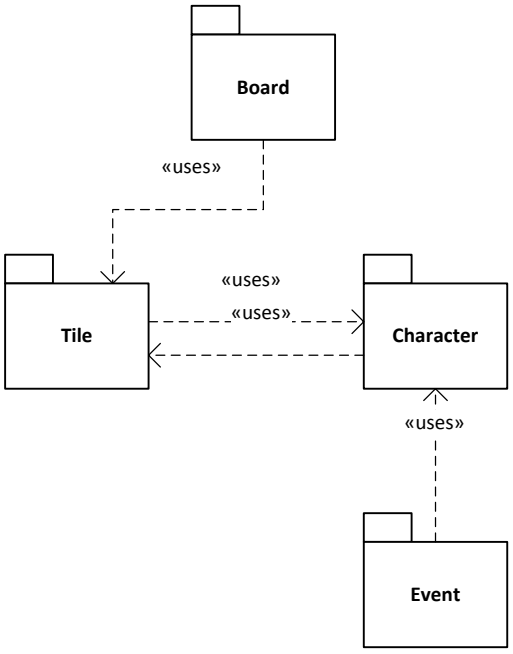
1. Start Application
    a. Set number of players
    b. Create Character
    c. Create Boarder
2. Start Game
    a. Determine who goes first
    b. Roll Dice
    c. Move Character to tile
        i. Tile is occupied
            1. Initiate combat
            2. Determine winner
            3. Penalize loser
        ii. Tile is vacant
            1. Randomly choose event
            2. Determine outcome
    d. Proceed through board
    e. First player to reach end wins
    f. Display game results
    g. Prompt user to play again
        i. Restart game
        ii. Close Application

.

Architectural Design:

Within this section, a high level view showing the major components of the framework will be demonstrated.

High Level View:

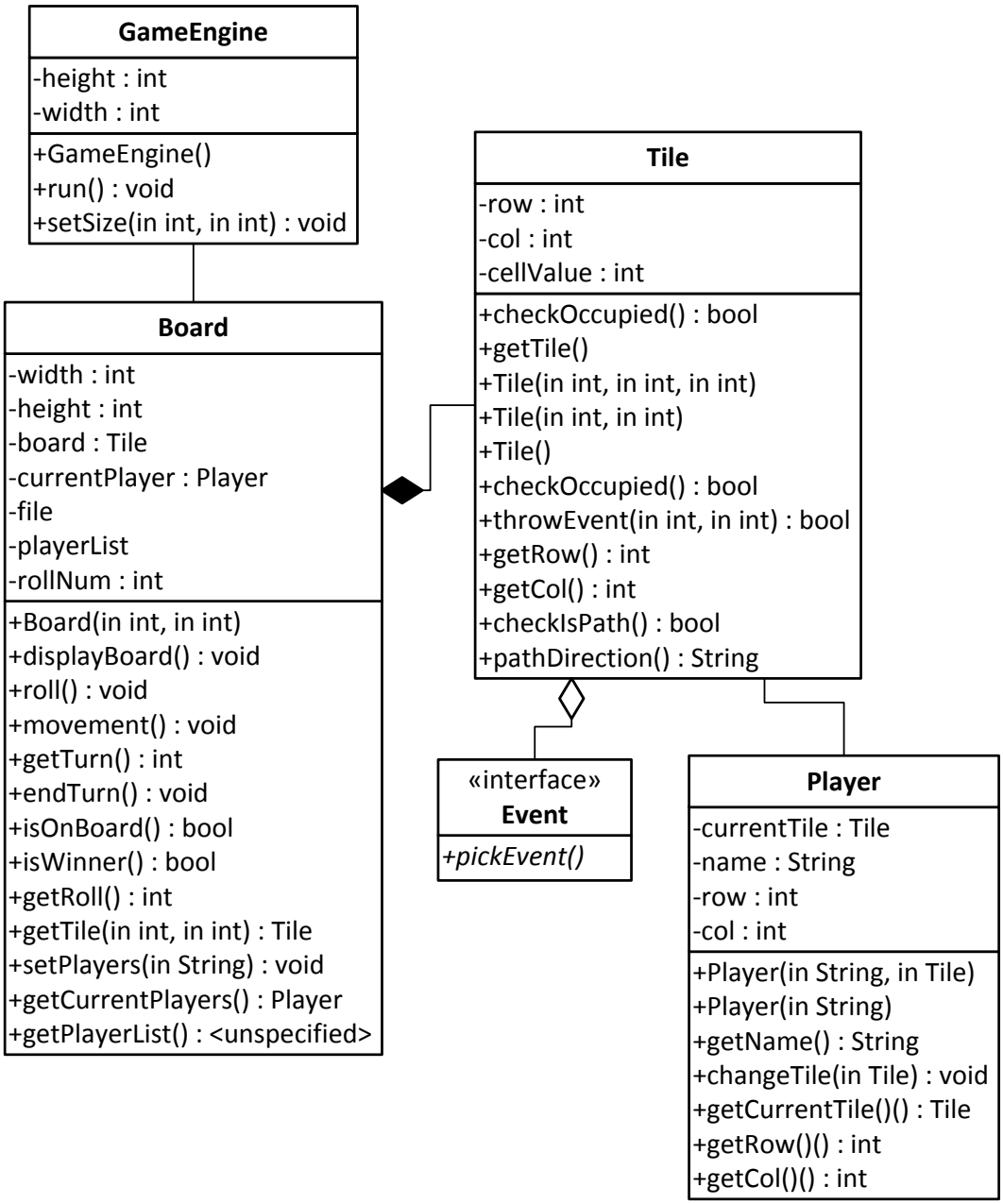


Design Patterns:

The use of design patters is often used as a general reusable solution to a frequently occurring problem. Design patterns are very useful, but in our situation, the use of design patterns was not necessary and was not used by our team. The solution we arrived at for our problem was unique and we did not use any formula or commonly used solution that we knew of to solve it.

Software Design:

Within the software design section a high level diagram will be shown of the static structure of the framework and the interaction between the framework's components.



Example Application:

To use the example application demonstrating some of the features of our framework is really quite easy:

1.After running the application, click the “Add Players” button located in the bottom right corner of the screen.
2. A second window will appear prompting to select number of players and enter the name of a player.
3. Enter number of players and respective player names.
4. Once this action is complete, proceed to the original screen to begin the game.
5. Click the “Roll” button to have your character roll a die to move forward on the board.
6. Character progress will be displayed in the top portion of the screen.
7. Textual messages and updates will be displayed in the lower center of the screen.
8. The game will end when the first player reaches the end of the board.
9. Have fun!

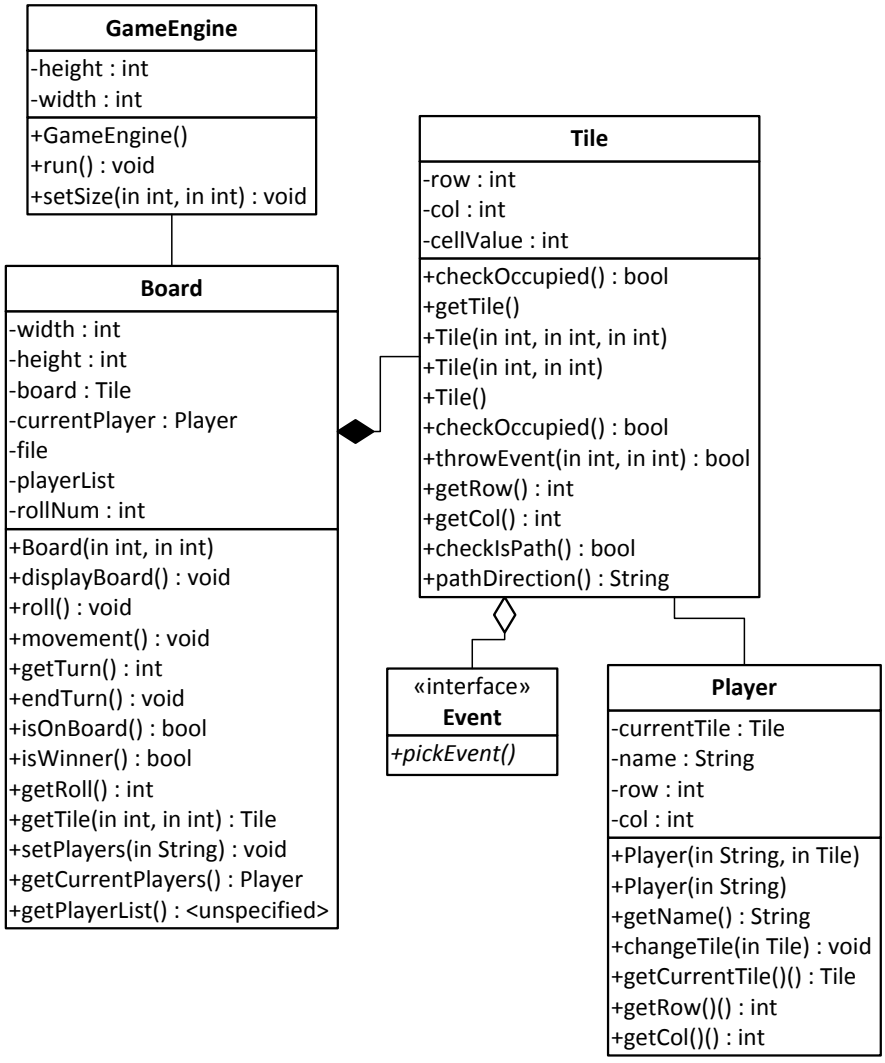
User Interaction:

Use Cases:

1. Start Application
    a. Set number of players (User clicks add player)
    b. Create Character (User chooses name)
    c. Create Board (User chooses size of board)
2. Start Game
    a. Determine who goes first (Highest roll wins)
    b. Roll Dice (Click the “Roll” button)
    c. Move Character to tile (Players are moved according to their roll)
        i. Tile is occupied (Two players occupy the same tile)
            1. Initiate combat (Players roll against each other)
            2. Determine winner (Highest roll wins)
            3. Penalize loser (Loser is randomly assigned an event)
        ii. Tile is vacant (Player lands on Tile)
            1. Randomly choose event (User is subjected to event)
            2. Determine outcome (User accepts event and outcome)
    d. Proceed through board (Players roll to move through the board)
    e. First player to reach end wins (The user wins)
    f. Display game results (results are displayed i.e. placing)
    g. Prompt user to play again (Box prompting to play again?)
        i. Restart game (Click “Play Again”)
        ii. Close Application (Close Application)

Software Design:

Included below is the static structure of the example application.



Quality Assurance:

We, as members of programming group known as the Kandymen, are committed to the development of high quality software. We will not sit idly by and let the days slide past, content on procrastinating to the last minute, just to push out a shoddy product with sloppy code. No. We will constantly be working to ensure well rounded and well thought out programs while using excellent coding practices. The Kandymen are dedicated to the practice of producing high-quality iteration reports, as these practices make life easier on everyone. We often have our Kandymen meetings where we discuss the upholding of our stratospheric coding standards as well as brainstorm on ever-evolving user interfaces. These are just some of the ways the manly members of the Kandymen are committed to producing a quality product every time.

Coding Standards:

In this section, the coding standards and procedures of our team are gone over in detail. The code comments and code documentation for our team is very simple, straight forward and consistent. Our team uses standard Javadoc comments and documentation and to ensure a uniform appearance and consistency, we had one of our members do all of the comments and documentation. This way everything was guaranteed to be the same.

Testing Process:

The Testing Process section will cover the process our team used for testing our software including details on the team's methods. Our team did not use the method of testing where the programmer writes the entire program and then tests it to see if it works. We decided it work be more efficient and more effective to test as we went. We were able to test many methods and classes individually such as our movement class and solved many problems before they became even more serious such as the way we had our player navigate his way through the board.. Occasionally these tests would lead us to discover problems we didn't know we had. Many times we had to go back over code we thought was complete to modify it further. In addition to this, we created and included unit tests for every single method in our program. Every method had at least one unit test, and some had two or more tests.

Retrospective:

We ended up splitting the project up among the 4 of us. Brandon did the user interface, Michael took charge on the framework design and example application. Marquis worked exclusively the implementation of the Spring framework, and Wesley did the documentation. The team successfully made time for meetings on several different occasions, but didn't do a good job of communicating outside of those meetings except for occasional contact over Canvas. Things to improve on would be to work more steadily on the project. We also did a poor job of dividing up the work, and we could have met more often assist other team members on their sections as we discovered fresh sets of eyes were very helpful.

Conclusion:

To conclude this documentation, our programming team, the Kandymen, needed to create a framework for a certain type, or genre, of computer program. Our group decided to create a framework for a video game as we thought it would be fun, interesting, and educational. It was not important as to what kind of game our framework designed for, all that mattered was that a framework for some type of game was created. So, the Kandymen decided on creating a framework for a board-game style of video game, comparable to the children's game of Chutes and Ladders or other like games. The entire point was to create the basic design and functionality for the board-game, while leaving the more specific details up to the game developers. We did so successfully.