Problem Statement - The Dragons of Kir

created by

Eric Guilford

Ian Hallam

Devon Timaeus

Contents

Executive Summary 3

High Level Problem Summary 3

Elevator Statement 3

Primary Success Criteria 3

Scope 3

Within Project Scope: 3

Outside Project Scope: 3

Detailed Problem Statement 4

Function 4

Form 4

Availability 4

Usability 4

Safety 4

Maintainability 4

Economy 5

Business Context 5

Customer Organization Constraints 5

Development Organization Constraints 5

Key Risks and Uncertainty 5

Timeline 6

Time 7

Historical Context 7

Current Context 7

Future Context 7

Key Stakeholders 7

UI Mockups and Turn Structure 8

UML Diagram 10

Code Coverage 11

Executive Summary

The purpose of this document is to address the issues that the project will be solving. This document is the first approach and interpretation of the problem. Following documents will further describe the problem and its solutions. This document will also discuss the depths of the problem as well as the different facets that will be considered during development.

High Level Problem Summary

Elevator Statement

We are creating an implementation of the board game "The Dragons of Kir" in the C# programming language. The primary reason for this is to allow for players to enjoy the game without having to set up the board, or worry about having lost pieces. Additionally, it will allow for people to throw dragons at each other, which is an excellent pastime.

Primary Success Criteria

The primary goal of this project is to allow for two players to play a complete game of "The Dragons of Kir" on a basic GUI that will display all of the needed information, and work at a reasonably pace. The success of the project is dependent on the completion of this goal by the end of the Spring 2012-13 quarter.

Scope

To further define the problem, an assessment of which elements of a payroll system will fall into the scope of the project follows:

Within Project Scope:
Board Setup
Player Turns
Player placing pieces (taking turns)
Dragon Movement
Win/Loss Checking

Outside Project Scope:
Extra Rules
"Mulligans" (Turn Redos)
Web-Based gameplay

Detailed Problem Statement

Function

1. Ability to set up a game from the start
2. Facilitate alternating player turns, and the actions therein
    a. Allow players to view board
    b. Allow players to place pieces
    c. Allow players to end their turn
    d. Allow Players to view their hand
3. Dragons move according to the rules
    a. Dragons wrap around the board
    b. Dragons interact with tiles appropriately
4. Wins and Losses are correctly identified
5. Allow for network-play (Secondary goal)

Form

Availability

- Distributed as a C# executable
- Allow for there to be local "accounts" for network and multi play

Usability

- Fast response times
- Clear board readability
- Easy to use/play
    - Clean, well-made design
    - Useful help text, and error messages

Safety

- Only game data is sent across network
- Users cannot break their system

Maintainability

- Software must be self-sufficient and stable

Economy

The expected value of this project is nothing more than entertainment. The project is meant to be game for people to play, and has no other contributions.

Business Context

The board game is not meant to directly interact with any sort of group, the game is simply meant to be played, and would only offer a form of entertainment.

Customer Organization Constraints

Anyone playing the game would need to have a computer, other than that, there are no constraints that would directly affect what the user would be allowed to do outside of the game.

Development Organization Constraints

The developers will be the members listed in the project team. At this time there are no plans on expanding the team nor future development after initial development is complete. All of the members are located very close to one another, this will help facilitate communication between the team and agile development of the system. The organizational structure is very flat, there is not a particular lead, each member is just responsible for their own work. Finally, the solution does not have a set budget, there are no plans to make any purchases for the project, however if the need arises there are channels which the team can use to acquire what is needed.

Key Risks and Uncertainty

The largest risk is the short span of time that the development team has to produce a working system. This is coupled with the fact that the team is limited in the amount of time they are able to spend on the project due to other activities that the members are also pursuing. Another risk is the unfamiliarity of the language that the team will be using.

Timeline

Week 3	• Board system functions. • Tiles can placed • Dragons exist • War tents exist • Turns exist • Tiles are drawn by players
Week 4	• War Tents get placed on the board • Dragons move forwards • Placing tiles refined
Week 5	• Effects are placed when tiles are placed • Starting implementation of effects for each different tile type • Starting implementation of Tile Effect resolution
Week 6	• All of Impact tiles have effects finished • Continue implementation of Tile Effect resolution • Impact Tiles effect Dragons • Players can win • Start on implementing Proximity Tiles
Week 7	• Proximity tile Effects finished • Proximity and Impact Tiles effect Dragons • Loop handling started • Effects are resolved properly
Week 8	• Dragon Tiles work properly • Effects for all tiles resolved properly • Loop handling refined and finished • Dragons movement and destroying of tiles finished
Week 9	• Playtesting and bugfixing • Polish • Possibly implement other “new” tiles
Week 10	• Prep for release

Time

Historical Context

The game is relatively new, and as such, the only historical context is that made by previous board game implementations on the computer.

Current Context

Currently, the board game is being sold at certain locations, but relatively few people have played it.

Future Context

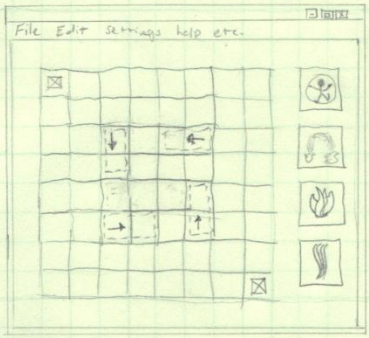
The game may take off in popularity, and this distribution of the game may reach a high demand. Alternatively, the availability of people to try the game may lead to people wanting the play the game more.

Key Stakeholders

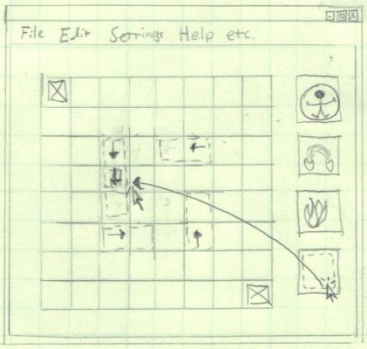
Name	Role
Michael "Buffalo" Hewner	Project Advisor
Eric Guilford	Project Team
Devon Timaeus	Project Team
Ian Hallam	Project Team
...	End Users

UI Mockups and Turn Structure

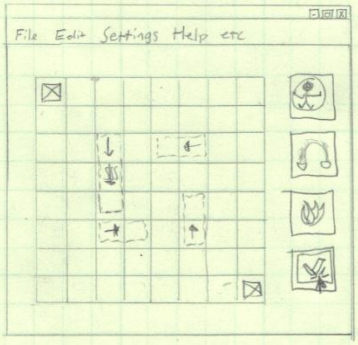
Csse 376-02 | Dragons of Kir Project | Eric Guilford cm2221 2013-03-17 | Pg 1/2



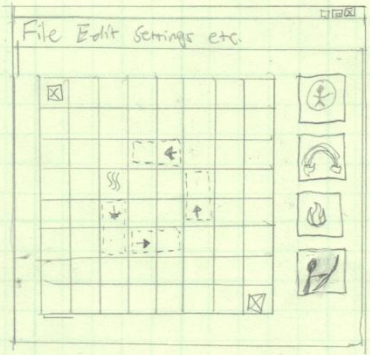
initial game state. Tiles that the user can place are shown on right game board is shown on left the path the dragon will take next turn is highlighted (indicated by dotted lines)



player drags tile to playing area and places it on the board dragons' path is updated to reflect new tiles.

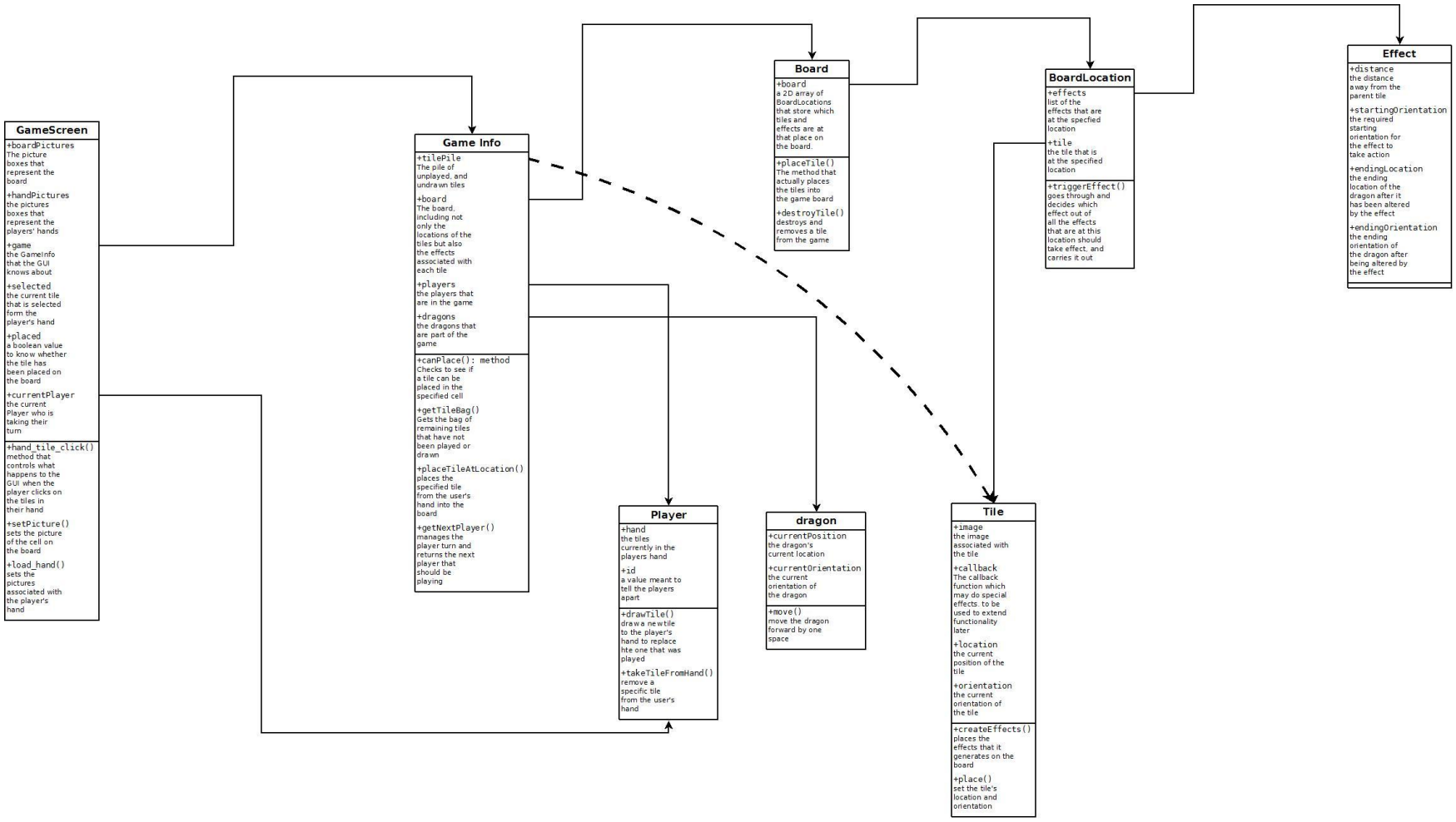


empty spot changes to a checkbox clicking it allows the player to confirm his tile placement and allow the game to advance.



dragons move to their updated positions, and a new rule is given to the user.

UML Diagram


GameScreen
+boardPictures
The picture boxes that represent the board
+handPictures
the pictures boxes that represent the players' hands
+game
the GameInfo that the GUI knows about
+selected
the current tile that is selected form the player's hand
+placed
a boolean value to know whether the tile has been placed on the board
+currentPlayer
the current Player who is taking their turn
+hand_tile_click()
method that controls what happens to the GUI when the player clicks on the tiles in their hand
+setPicture()
sets the picture of the cell on the board
+load_hand()
sets the pictures associated with the player's hand
Game Info
+tilePile
The pile of unplayed, and undrawn tiles
+board
The board, including not only the locations of the tiles but also the effects associated with each tile
+players
the players that are in the game
+dragons
the dragons that are part of the game
+canPlace(): method
Checks to see if a tile can be placed in the specified cell
+getTileBag()
Gets the bag of remaining tiles that have not been played or drawn
+placeTileAtLocation()
places the specified tile from the user's hand into the board
+getNextPlayer()
manages the player turn and returns the next player that should be playing
Board
+board
a 2D array of BoardLocations that store which tiles and effects are at that place on the board.
+placeTile()
The method that actually places the tiles into the game board
+destroyTile()
destroys and removes a tile from the game
BoardLocation
+effects
list of the effects that are at the specfied location
+tile
the tile that is at the specified location
+triggerEffect()
goes through and decides which effect out of all the effects that are at this location should take effect, and carries it out
Effect
+distance
the distance away from the parent tile
+startingOrientation
the required starting orientation for the effect to take action
+endingLocation
the ending location of the dragon after it has been altered by the effect
+endingOrientation
the ending orientation of the dragon after being altered by the effect
Player
+hand
the tiles currently in the players hand
+id
a value meant to tell the players apart
+drawTile()
drawa newtile to the player's hand to replace hte one that was played
+takeTileFromHand()
remove a specific tile from the user's hand
dragon
+currentPosition
the dragon's current location
+currentOrientation
the current orientation of the dragon
+move()
move the dragon forward by one space
Tile
+image
the image associated with the tile
+callback
The callback function which may do special effects. to be used to extend functionality later
+location
the current position of the tile
+orientation
the current orientation of the tile
+createEffects()
places the effects that it generates on the board
+place()
set the tile's location and orientation

Code Coverage

For this project, we will use the dotCover coverage tool to record all of our testing coverage, and for general testing of our code. Below is a snapshot of our most recent coverage report.