Tracy L Clark
Senior Project Proposal
Computer Science
Fall 2016

Project Overview

The goal of my senior project is to create a web based version of the game Catan. I plan to use JavaScript, HTML5, and CSS. The technologies I plan to use include MongoDB and Node.js with libraries such as Socket.io, Express, and BodyParser. The main objective is to create a fluid experience for the users that mimics Catan gameplay.

The game Catan is a resource management/strategy board game for by 3-4 players. It is played on a map made up of hexagonal tiles. This game is turn based. Each turn resources are generated according to a single dice roll, if the number on a tile matches the roll, all players with a token on a vertex of that tile gain that resource. Resources may be traded between players or with the Bank (at a 4:1 ratio) or Harbors at different ratios depending on Harbor type. On a players turn, more tokens can be purchased and deployed based on the rules for that token type (road, settlement, and city). Roads may be played on edges of tiles and may only be placed adjacent to tokens owned by that player. Settlements may only be place on vertexes adjacent to a road owned by the player and not within one edge of an existing city or settlement. Cities are placed by replacing an already existing player settlement.

Resources may also purchase development cards. Development cards come from a deck and come in five varieties: Knights, Victory Points, Monopoly, Road Building, and Year of Plenty, all of which have different impacts on the game play. Victory points can be obtained in a variety of ways other than cards. Settlements and cities are worth victory points, as are special cards for the largest army and longest road. The game ends when a player has 10 victory points on their turn.

This game is incredibly complicated due to the number of different changes that can take place in a single turn, making it an intriguing challenge to attempt. Determining the longest road alone requires implementing a longest path algorithm, as that problem is known to be NP-Hard. Implementing the complete game on its own under my grading system below is worth a C.

Feature Point Estimation

Core Game	**Points**
Create a default game map for beginner players	3
Create a randomized game map for advanced players	3
Implement a tiled map grid	3
Implement a hexagonal tiles for the map grid	8
Implement tile types	1
Implement resource number tokens	2

Implement resource generation through dice rolls	5
Track player resources	2
Implement development cards	5
Implement road building	2
Implement settlement building	3
Implement city upgrades	1
Implement robber	3
Implement victory point system (includes checking for a win)	2
Implement victory points for longest road: Longest Path Problem, NP-Hard	13
Implement victory points for largest army	1
Implement Harbors	3
Implement trading with other players	5
Implement 4:1 resource trade in	2
Implement turn system	2
Implement desert tile	2
Implement Set-up Phase	5
Implement roll-off for starting player	1
Implement starting resources	2
Implement even number token distribution	3
Implement random number token distribution	2
Implement game options for players to choose style of game play	3
Total Available for Core Game	87

Graphical Interface	**Points**
Draw game map with HTML Canvas	8
Use image sprites instead of just drawing tiles	2
Implement map camera (ability to pan and zoom on game map)	3
Show graphical dice	1
Animate dice	5
Viewable current player scores	2
Login Screen	1
Rules Overlay	1
Game play message displays	1
Use Socket.IO to keep game synchronized between players	3
Make a mobile friendly UI design	13
Total Available for Graphical Interface	40

User Features	**Points**
Implement spectators	3
Implement a “local hotseat” multiplayer option	5
Implement “local hotseat” with online play (so that a single PC can be used for	8

multiple players in in a single location playing against people in other locations)	
Saveable game state	8
Persistent game state	3
If a player leaves the ability for a spectator to take their place	1
Create game lobby	5
Create multiple play rooms for simultaneous games	8
Users have the ability to “lock” a game and put a password on it	2
Implement being able to choose to spectate or play the game	1
Implement a chat box	2
Implement voice chat	8
Implement game login (with MongoDB validation)	2
Implement user account creation (with MongoDB validation)	2
Maintain server-centric security policy with clients handling no game logic	3
Total Available for User Features	61

Grading Scale

Grading Scale	**Grade**
95+	A
90-94	B
85-89	C
80-84	D
<80	E

I used the Agile Fibonacci sequence for assigning point values. I assigned points based on the perceived difficulty of each task. For the grading scale I chose a total based on implementing every feature of the game plus very bare-bones UI as a baseline for an A. Many of the features on this list are niceties for gameplay and not difficult or necessarily challenging, so I made the A baseline as not implementing these optional features.