Design Document - Stock Market Game (1963) App

A Groen

May 2022
Version: 1.0

Introduction

0.1 Document Purpose

This document serves to lay out the design objectives and methodology towards the design of the stock market game (1963 edition - hereafter denoted as **board game**). This document will be used during development, and may drastically change over time as features are considered. Each chapter will cover one major aspect, and will contain an introduction explaining its purpose and intent.

0.2 New Version Changes

None at this time - V1

0.3 Programs and Packages in use

Section	Selected System	Reasoning
Language	Rust	Static Typing + Compiled. Wasm allows for packaging for online use if needed
Graphics	TBD.	
Audio	TBD	
Other	fastrand	Dice rolling system
	std	General Purpose

Part I

App Content

Chapter 1

Background

The core concept of this application comes from the 1963 board game "The Stock Market Game" published by Timesmart limited, and created by John S. Koster. In order to properly adapt and impove this system in the application, the core gameplay, strategy and any concerns with the owner of the origional game will be treated here to ensure full understanding.

1.1 Premise

The board game consists of a game played between 2 or more players based on reaching 100,000 dollars net worth by purchasing and selling stock as the price rises and falls. The origional game facilitated the stock price action by a proprietary slider placed in the center of the board, labeled "B" on refBoard; this would move up and down as the game progressed to mimic the stock market itself moving. Each space on the board would provide the opportunity to purchase a specific stock, facilitate a shareholders meeting or would have some special event related.

1.2 Gameplay

The game begins with each player selecting a career from one of the 4 options, labeled "A" on refBoard. This assigns the player to a set of dice numbers, and a payout amount if those numbers are rolled. Each player rolls the dice in turn, and pays out accordingly. Each player remains in this phase until they wish to depart and move to phase 2, or once they accumulate 1000 dollars.

Phase 2 consists of moving around the edge of the board, labeled "C" on refBoard purchasing and selling stock. On each players turn, they Sell any stock that they wish to, roll 2 dice, then move to the corresponding square. Once at the correct square the market moves in the corresponding direction, the player has the option to buy the listed stock and the turn moves to the next player.

If the player lands on a "Stockbrokers meeting" square (labeled "D" on refBoard) and own at least 1 share of the listed stock, they have the option to attend the Stockbrokers meeting. If they do not own the listed stock, they can buy a maximum of one share at the current market price then attend. Once inside the Stockbrokers meeting, stock splits allow for the player to increase their ownership proportionally to how many shares they already own. Only the player at the Stockbroker meeting can take advantage of these splits.

if an event or special card ever causes a player to go bankrupt (total worth less than 0), the player forfeits all assets and cash and moves back to phase 1. Players being forced to sell shares to cover a special card are forced to sell for the minimum listed price, not current market price.

The first player to 100,000 total worth as of the start of their turn is the winner. This includes cash and all stock valued at the current market price.

1.3 Strategy

Due to the random nature of the market price, and what stocks a player is able to purchase, the game has a high element of chance. While it has not been solved, or brute forced with algorithems due to the age of the board game, the prevaling strategy appears to aim to purchase large amounts of stock of the high value companies, and try to stay on one side of the stock ticker as much as possible. This means the player is not hedged if the ticker moves against their holdings, but also means they are not losing value with half their portfolio as the other half increases.

Although the highly random nature means skill has less impact on outcome relative to a skill based game like poker, there are enough decisions to keep players engaged and ensure a feeling of control.

1.4 Public perception

Currently, the game has largely fallen out of the public eye. New boards are not being produced, and the lack of a secondhand market means new potential players have no supply, and the game has entered the realm of collectors. This provides a market for a revitalized game, as is the intent with the app.

One consideration is the choice of jobs and companies used. The board game uses companies such as Alcoa, International Shoe Company, Maytag, J.I. Case. With modern audiences these companies will not have major, if any brand perception. A modern equivalent should therefor be used in order to exploit brand recognition - companies such as apple, wallmart, Nvidea etc. in the top of the S&P 500. This will be treated in refCompanies and Jobs

1.5 Patents

This game concept and system was first patented in the US in 1963 and last patented in 1990. As such, it is currently in the public domain, and no US patent concerns should arise.

Chapter 2

Overview of Application Content

When adapting this board game, the oppertunity arises to update and modernize the content. While the intent remains to be true to the origional source material as much as possible, modernization will be vital to ensure brand recognition, and certain event spaces can be updated to be more in line with modern events. The board look itself can also be updated to bring it in line with modern wall street.

2.1 Companies and Jobs selected

2.2 Board Squares in use and layout

Part II

Backend

Chapter 3

Core Functionality

This application will be built using the RUST programming language, and its associated packages. As packages are needed they will be introduced in this part, a full tabulated list of packages used can be seen in the introduction. The first step in development is the core concepts required to generate a functional prototype. In this application, these concepts consist of; dice rolling, player location tracking, stock market value tracking and buying/selling of stock.

3.1 Game Operation

Each turn will be computed by taking the next player in order, presenting the option to sell any stock, look at the board or roll. Upon rolling, the dice will roll, the player will be moved to the assigned square and the market will move as needed. Then the player is given the chance to buy as many shares as they wish/can of the given stock or end their turn. This aspect of the code prompts the player and handles the passing of info between the frameworks below.

3.2 Rolling Dice

At the core of most board games is the dice roll to generate a random number, and for this purpose the fastrand package in rust will be used. This will allow us to randomly select 2 numbers ranging from 1-6, providing the correct distrbution of rolls 2 real dice would provide. This specific random package was chosen due to its speed, and ease of implementation. By rolling 2 distinct "dice" the rendering of the roll can be done graphically, by drawing die faces on the screen to correspond to the 2 numbers rolled.

Within the codebase, this element will be responsable for generating the rolled number, then passing it to both the GUI system, and to the player character for the backend movement.

3.3 Board State

The total board state, namely a list of all squares and their locations relative to each other, as well as the current position of the market ticker, turn counter and potentially game clock.

The board squares will be listed in a fixed length array, containing tuples of the square number (numbered CCW from Bottom center start square) The stock associated with the square, the odd/even roll direction and any other considerations (shareholder meeting, start, etc.)

The Market ticker position will be tracked similarly, numbered with distance from top. This will not store the corresponding stock price values, just the core ticker location. if the ticker over or underflows, it will "bounce" rather than loop. that is if the ticker is in position 1 and moves up 3, it will end in position 2 ($1- > 0- > 1- > 2$).

The turn counter will simply count the number of full turns to pass, where a turn is 1 action from each player.

Game clock (potential) is similar, and simply tracks the total time of the game.

3.4 Player Location and Status

Each player will have its own instance of the player class, that will consist of the five elements that describe a player; Money, Stocks owned, Current square, profession, Game piece/Name.

The game piece or name of the player is used to track turn order, and pass the correct info along to the GUI for drawing piece changes. This element is selected at the start of the game, and is set at initialization of the struct.

The Money counter will simply be the value of the players current cash balance. It will not include the current market value of the stocks owned, only the immediately accessable cash. This value will be read to check if purchases can be completed, display to player and to factor in to the win calculation.

The stock owned tracker will measure the count of shares owned in each of the 8 companies. these values will be read to facilitate sales, display to player if asked, win checking and shareholders meetings. Included in this will be the functionality to find the current market value of all shares owned.

Current square will store the current location of the player on the board at any time. At the start of the turn it will be read to find any applicable rules (such as even-odd direction). Then after the roll the field is updated with the new location, then read to find the market move, stock available for purchase and any other factors.

Profession will store the players chosen profession, as well as what rolls pay their salary. Not included in this section is which phase the player is in, that logic will be handled by setting their current square to the work square. This field will be read for showing to the player, as well as checking relevent dice rolls against the salary paying ones.

3.5 Placing Orders and valuing

This component contains the stock price of each company (expressed as a fixed length array), and handles getting the price of an asset for the purposes of display, valuation, and transactions. This component also handles the grunt work of a transaction, simply outputting the updated money and stock values to the player component.

Part III

GUI and UX

Part IV

Release and beyond