Pandemic Game Design Document

Maksym Turkot
05/26/2021

Introduction

The goal of the project is to implement an automatic Pandemic board game in Java using graphs, and to evaluate the performance of various settings of player agents.

This document outlines the final design of major components involved into the development of the game. It describes basic game rules to implemented, algorithms for map creation and pawn movements, descriptions of classes comprising the program, statistics to be generated by the program, and design of logging data.

Rules Implemented

Setup:
- At the start of the game, infection counter is set to 0, infection rate is respectively set to 2.
- Three random cities are infected with three cubes, three with two, and three with one cube.
- Four pawns are placed at random cities.
- Each pawn is given 2 city cards.
- There is a deck of player cards, and a deck of infection cards.

Playing:
- Each pawn searches the map to determine where to go, according to its algorithm.
- Each pawn will make 4 moves, either moving to a different city, fighting the disease in the city, or curing the disease. Once all actions are done, pawn will draw 2 cards.
- Pawns move to neighboring cities.
- Pawn can remove one disease block per action, or all in that city if that disease is marked as cured.
- If a pawn has five city cards of the same color, it can discover a cure.
- After each pawn performs four actions and draws cards, a number of city cards equal to the infection rate are drawn and infected with one cube each, only if disease of that color has not been eradicated.

Drawing Cards:
- If a pawn gets more than seven cards, cards are discarded until pawn again has seven cards.

Endgame:
- If pawns manage to cure all four diseases, pawns win.
- If no more cubes of certain color are left, pawns lose.
- If no more city cards are left, pawns lose.

Map Creation

Maps are generated manually and read by the program to construct a TreeMap.

To create a space graph, the default map is taken, and one edge is removed from each vertex, while also removing the same edge from the adjacent vertex, unless this creates an isolated vertex. If there are connected components, new edges are created until components have been merged into one connected graph.

To create a dense graph, new edges connecting random vertices are added, unless this creates multiple edges.

Pawn Action and Movement Algorithm

Whet it's given pawn's turn to play, it checks if it has five cards in its hand of the same color which allows it to develop a cure, and does so if possible. If not, or once cure was developed, pawn checks if there are infections in the city it is currently in. If so, if fights the disease. If not, it moves to a random neighboring city.

Class Descriptions

Class Controller

This class loads the maps and runs each configuration.

Fields:

- None

Methods:

- static void runGames() - runs each configuration with 5 random seeds.

Class Game

This class runs the game itself, managing underlying tasks such as card shuffling and distribution, city infection, and keeping track of all counters and objects.

Fields:

- TreeMap<String, City> cities - tree map of cities.
- LinkedList<String> baseDeck - base deck of cards.
- LinkedList<String> shuffledCityDeck - shuffled deck of city cards.
- LinkedList<String> shuffledInfectionDeck - shuffled deck of infection cards.
- LinkedList<Pawn> pawns - list of pawns in the game.
- Rand rand - randomizer.
- String mapFile - name of file containing adjacency list of this map.
- int cureCnt - how many diseases were cured.

- int seed - randomizer seed.
- int turnCnt - counts turns in the game.
- int infectCnt - counter of infection mark.
- int infectRate - rate of infection based on infectCnt.
- int yel - yellow infection cubes.
- int red - red infection cubes.
- int blu - blue infection cubes.
- int grn - green infection cubes.
- boolean isYCured - true if infection Y is cured.
- boolean isRCured - true if infection R is cured.
- boolean isBCured - true if infection B is cured.
- boolean isGCured - true if infection G is cured.
- boolean isYEradicated - true if infection Y is eradicated.
- boolean isREradicated - true if infection R is eradicated.
- boolean isBEradicated - true if infection B is eradicated.
- boolean isGEradicated - true if infection G is eradicated.

Methods:

- void incrCureCnt() - increments cure counter when cure is discovered.
- LinkedList<String> getBaseDeck() - returns base deck of this game.
- LinkedList<String> getShuffledCityDeck() - returns shuffled city card deck.
- LinkedList<String> getShuffledInfectionDeck() - returns shuffled infection card deck.
- Rand getRand() - returns randomizer for this game.
- TreeMap<String, City> getCities() - returns TreeMap of cities.
- int getYel() - returns number of yellow cubes.
- int getRed() - returns number of red cubes.
- int getBlu() - returns number of blue cubes.
- int getGrn() - returns number of green cubes.
- boolean getIsYCured() - returns true if disease Y is cured.
- boolean getIsRCured() - returns true if disease R is cured.
- boolean getIsBCured() - returns true if disease B is cured.
- boolean getIsGCured() - returns true if disease G is cured.

- void cureY() - marks disease Y as cured.
- void cureR() - marks disease R as cured.
- void cureB() - marks disease B as cured.
- void cureY() - marks disease Y as cured.
- void eradicateY() - marks disease Y as eradicated.
- void eradicateR() - marks disease R as eradicated.
- void eradicateB() - marks disease B as eradicated.
- void eradicateG() - marks disease G as eradicated.
- void setInfectionRate() - sets infection rate based on infectionCnt.
- LinkedList<Pawn> getPawns() - returns list of pawns.
- void runGame() - runs the game. In order makes each pawn do actions, distributes cards and infects cities.
- void endGame(String msg, String logString) - manages the end of the game by printing log file and outcome file based on the outcome of the game.
- LinkedList<String> toLinkedList(Set keySet) - converts a set to a linked list.
- LinkedList<String> shuffleDeck(LinkedList<String> base, boolean infectionDeck) - shuffles a base deck of cards to produce either a shuffled player or infection deck. *TESTED by checking if a shuffled deck is produced.*
- LinkedList<Pawn> createPawns() - creates pawns placing them in random cities.
- void dealCards(Pawn pawn) - deals two cards to pawns. If more than 7 cards are in pawn's hand, calls pawn to discard cards. *TESTED by checking if pawns receive city cards.*
- void infectCities(int cnt, int lvl) - infects a number of random cities with a passed level of infection.
- void infect(City city, char infection, int lvl) - infects a city with a given infection with a given infection level. *TESTED by checking if cities are being infected.*
- String logString() - compiles a log string to be printed in log file.
- String outcomeString() - compiles an outcome string to be printed in the outcome file.

Class City

This class constructs a city (vertex) of a map, keeps track of all relevant information.

Fields:

- static int cityCnt - counter of cities constructed.
- TreeMap<String, City> neighbors - TreeMap of neighbors of this city.
- String name - name of this city.
- char color - color of this city.

- int id - ID of this city.
- int infectionY - level of infection Y.
- int infectionR - level of infection R.
- int infectionB - level of infection B.
- int infectionG - level of infection G.

Methods:

- int getId() - returns ID of this city.
- TreeMap<String, City> getNeighbors() - returns TreeMap of neighbors of this city.
- void setNeighbors(TreeMap<String, City> neighbors) - sets neighbors of this city.
- String getName() - returns name of this city.
- char getColor() - returns color of this city.
- void setColor(char color) - sets color of this city.
- int getInfectionY() - returns level of infection Y.
- void increaseInfectionY(int lvl) - increases level of infection with Y by 1.
- void decrementInfectionY() - reduces level of infection with Y by 1.
- void eliminateInfectionY() - reduces level of infection with disease Y to 0.
- int getInfectionR() - returns level of infection R.
- void increaseInfectionR(int lvl) - increases level of infection with R by 1.
- void decrementInfectionR() - reduces level of infection with R by 1.
- void eliminateInfectionR() - reduces level of infection with disease R to 0.
- int getInfectionB() - returns level of infection B.
- void increaseInfectionB(int lvl) - increases level of infection with B by 1.
- void decrementInfectionB() - reduces level of infection with B by 1.
- void eliminateInfectionB() - reduces level of infection with disease B to 0.
- int getInfectionG() - returns level of infection G.
- void increaseInfectionG(int lvl) - increases level of infection with G by 1.
- void decrementInfectionG() - reduces level of infection with G by 1.
- void eliminateInfectionG() - reduces level of infection with disease G to 0.
- String toString() - compiles string with information about this city.

Class Pawn

This class constructs an individual pawn with its treating, moving, and action taking logic.

Fields:

- static int pawnCnt - counter of pawns created.
- Game game - game with which this pawn is associated.
- LinkedList<String> cards - list of cards in pawn's hand.
- LinkedList<String> cardColors - list of colors of cards in pawn's hand.
- LinkedList<String> cityTrace - trace of cities this path has visited during the game.
- City currentCity - city in which pawn is currently located.
- int id - ID of this pawn.

Methods:

- int getId() - returns ID of this pawn.
- LinkedList<String> getCards() - returns cards of this pawn.
- LinkedList<String> getCityTrace() - returns city trace of this pawn.
- void putCityTrace(City city) - adds a city to the city trace.
- void addCard(String card) - adds a card to the path's hand.
- City getCurrentCity() - returns current city of this pawn.
- void setCurrentCity(City currentCity) - sets a current city of this pawn.
- void doAction(TreeMap<String, City> cities) - configures what action pawn will take under given state of the game.
- LinkedList<String> toLinkedList(Set keySet) - converts set to a linked list.
- void cardsToColors() - creates a list of card colors.
- void treat() - treats infection in current city. *TESTED by checking if disease is removed from the city.*
- void discardCards(TreeMap<String, City> cities, boolean isYcured, boolean isRcured, boolean isBcured, boolean isGcured) - determines which cards should be discarded from pawns hand. *TESTED by checking if cards are correctly discarded.*
- void discardCard(TreeMap<String, City> cities, char infection) - discards a card from pawn's hand.
- void searchMinColorCnt(TreeMap<String, City> cities, int yel, int red, int blu, int grn) - determines - discards a card of the smallest color count.
- String toString() - returns a string with information about this pawn.

Class MapFileReader

This class manages data retrieval from map files.

Fields:

- Scanner fileScanner, lineScanner - scanners.

Methods:

- static TreeMap<String, City> readMap(String filename) - reads information from adjacency list in the map file. *TESTED by checking if information is correctly retrieved from the map file.*

Class OutputManager

This class manages output file creation.

Fields:

- static PrintWriter fileWriter - writer.
- static File outputFile - output file.

Methods:

- static void writeLog(String name, String data) - writes log data to a log file.
- static void writeStatistics(String name, String data) - writes outcome data to an outcome file.

Class Random

This class generates random numbers to be used throughout the program.

Fields:

- Random rand - random object.

Methods:

- int getRand(int upperbound) - gets random integer up to a passed bound.

Outcome Data Generation

Program determines if the game was won or lost and what was the reason for a loss. It also tells how many diseases were cured and what were levels of infection when the game ended.

Logging File Design

****

Map: map1-example | Seed: 1 | Turn: 1 | InfectRate: 2

==========

CityCards: 32 | InfectionCards: 31

==========

Diseases:

	Active	isCured	isEradicated
Yellow:	5	false	false
Red:	0	false	false
Blue:	4	false	false
Green:	8	false	false

==========

Pawns:

ID: 123

CityCards: [Chicago, Miami, Beijing, Madrid]

CardColors: [Y, Y, Y, G]

CurrentCity: Tokyo

CityTrace: [Seoul, Tokyo]

…

==========

Cities:

Algiers

Color: Y

Neighbors: [Cairo, Istanbul, Madrid, Paris]

Infections: 1Y,0R,0B,0G

…