Leah Ferguson and Ashley Thomas
CS 230
Final Project
12/17/2013

Technical Report: Cows vs. Monsters

ADTs

- Graph: Our game board is shaped like a graph; cows and monsters reside at some of the vertices. Monsters can kill any cow that is adjacent to it (ie. they share an edge). The graph stores the location of each of the cows and what is adjacent to each cow. We picked a graph because it perfectly exemplifies what our game board should be and act like.
- LinkedList: This ADT will hold a list of top 10 scores. We will add them in such a way so that they remain in order. We picked a LinkedList because the top ten scores is a fairly small collection of data that doesn't need much manipulation.

Classes

- GameBoard
    - The GameBoard class contains a graph, which is implemented by AdjMatGraphPlus. Each vertex of this graph is of type cow, and an edge represents that two cows are next to each other. This is important because a monster can only kill cows that are adjacent to it, and only those cows adjacent to a killed cow will become scared. This class contains various methods that get cows with specific characteristics, such as those that are adjacent to other cows or the living monsters. It also contains a method to move a cow from its current position to an empty vertex, as well as a method to reset the board, which recreates the original graph.
- AdjMatGraphPlus
    - This implements the graph ADT which we will use in GameBoard. We have somewhat changed our version of AdjMatGraphPlus to solve some specific problems we are running into with our gameboard, such as constructing the cows from a provided .tgf file.
- Cow
    - Cows are the pieces of our game. Since cows are only at some of the vertices of the graph, our cow class will have a boolean instance variable that indicates whether it is visible or not. Cows can move to other "open" vertices of the GameBoard (the ones that are not visible). Cows have a name instance variable that gives the name of the cow (which is the initial location of the cow in the graph). Cows have another boolean instance variable "scared" that indicates if the cow is scared or not. The cow also has an instance variable indicating how many times it has been scared. Cows also have an

instance variable indicating whether they are dead or not. A dead cow cannot be moved. Finally, a cow has an instance variable indicating whether it is a cow or a monster. Monsters have special functionality, such as killing cows. Most of these variables will be accessed through getter methods, and the Cow class has some setter methods as well.

- Game
    - The game class contains most of the logic for our game. It contains an instance of GameBoard and an instance of ComputerPlayer. The game class will contain the methods that initiates gameplay and control the logic of the game. Game will also include a LinkedList static variable that keeps track of the high scores of the players.
- Player
    - A simple class that links a person's name with their score. We use this class in our Game class for the High Scores LinkedList.
- LinkedList
    - We use java's provided LinkedList class.
- ComputerPlayer
    - This class controls the actions of the computer. It contains an instance of the GameBoard class, and it will contain algorithms that determine the moves of the computer, such as where to move cows, which cow its monsters should kill, and how to guess where its opponent's monsters are.
- GamePanel
    - Creates the panels (all the interactive elements) and the listeners to perform actions in the game.
- GameGUI
    - GameGUI creates the frame and creates an instance of the game class. It has listeners that detect what mode the game is in and calls methods in Game appropriately.
- BackgroundPanel
    - Class that we used to have a background image in the GUI.

Actions

- GameBoard
    - Constructor: The GameBoard constructor uses the second constructor in the AdjMatGraphPlus class, which takes a .tgf file and creates a graph with vertices of type cow. The game board populates its vertices with the cows at the beginning of the game. Each vertex will contain a cow, but only a certain number of them will be visible.
    - cowsAdjacentTo (int index): Gets the cows adjacent to the given index, dead cows and invisible cows are not included.
    - getAllEmptySpaces(): Gets all the empty spaces, ie vertices containing an invisible cow.
    - getAllLivingCows(): Gets all the living cows, includes monsters but does not include

invisible cows.
    - getCow(int index): Gets a cow object of a given index (ie vertex).
    - getCowsAdjacentToMonsters(): Gets the cows adjacent to living monsters, does not include duplicates of any cows, any dead cows, any invisible cows, or any monsters.
    - getCowsRemainingI): Gets the number of living cows remaining, does not include monsters.
    - getIndex(Cow cow): Gets the index of a cow object in the graph (ie where the cow is in the graph).
    - getIndexWithName(String name): Gets the index of the cow given its name.
    - getLivingMonsters(): Gets the living monsters.
    - moveCow(int index1, int index2): Moves a cow from its current index to a given index.
    - randomlyMoveCow(int index): Randomly moves a cow, we only use this at the beginning of the first round, so we don't need to check for dead cows.
    - resetBoard(): Resets the gameBoard to be the original graph using the given tgf file.
    - setCow(int index, Cow cow): Sets the given index to be the cow object given.
    - toString(): Uses the toString in the AdjMatGraphPlus class to print the graph contained in the GameBoard.
- AdjMatGraphPlus
    - Constructor: The second constructor creates a graph from a .tgf, specifically creating each vertex as a cow. Each vertex will contain a cow, but only a certain number of them will be visible.
    - getVertex(int index): Gets the Cow contained at index in the array.
    - setVertex(int index, T vertex): Sets the object at position index in the array to be vertex, overriding what was previously there.
- BackgroundPanel
    - Constructor: Creates a new BackgroundPanel with an image as its background.
    - paintComponent(Graphics g): Draws the image.
- Player
    - Constructor: Creates a player object with the name and score specified by the user.
    - compareTo(Player player): Implements the Comparable interface; compares two players according to their scores.
    - getName(): Gets the name of the player.
    - getScore(): Gets the score of the player.
    - toString(): Returns a String containing the name and score of the player.
- Cow
    - Constructor: Creates a cow object, assigns whether it is visible or invisible. Initializes it to be alive, not scared, and a cow (rather than a monster).
    - becomeCow(): Makes the cow a cow, rather than a monster.
    - becomeMonster(): If the cow is visible makes it a monster, otherwise does nothing.

    - becomeNotScared(): Makes the cow not scared.
    - becomeScared(): Makes the cow scared and increments the timesScared variable by 1.
    - die(): Sets the dead instance variable to true.
    - getName(): Gets the cow's name.
    - getTimesScared(): Gets the number of times a cow has been scared.
    - isDead(): Checks if a cow is dead.
    - isMonster(): Checks if a cow is a monster.
    - isScared(): Checks if a cow is scared.
    - isVisible(): Checks if a cow is visible.
    - toString(): Returns the name of the cow.
- Game
    - Constructor: The constructor of game initializes an instance of GameBoard and an instance of ComputerPlayer. It creates a new Linked List of Linked Lists of Strings, to store the names of each cow and how many times they have been scared. It initializes a list of all the possible modes that the game can be in, which control the sequence of the game. It initializes variables to store the cow that was most recently killed and a LinkedList of integers storing the indices of the most recently scared cows. Finally it initializes a LinkedList of high scores.
    - computerChoosesMonsters(): Calls the ComputerPlayer method chooseMonsters to choose a monster, and changes the mode to computerKillsCow.
    - computerGuessesMonster(): Uses the computerPlayer method guessMonster to kill a cow that has been scared the most number of times; if the round is over, changes the mode to round1Over, and if the cow guessed was not a monster, changes the mode to userKillsCow; if JEF was correct or there are not more moves, he guesses again. Also updates the computer score.
    - computerKillsCow(): Calls the ComputerPlayer method killCow to kill a cow and updates the inforamtion about the scared cows; if the game is over, changes the mode to gameOver and updates the scores, and otherwise updates the mode to computerMovesScaredCows.
    - computerMovesCows(): JEF randomly moves cows at the beginning of the game, and this method also changes the mode to userChoosesMonsters.
    - computerMovesScaredCows(): Calls the ComputerPlayer method moveScaredCow to move a scared cow and updates this cow to no longer be scared. If all of the scared cows have been moved, change the mode to userGuessesMonster.
    - getCow(int index): Calls the GameBoard getCow method to return a cow object given an index representing its location.
    - getCowsAdjacentToMonsters(): Uses the gameBoard method to get all the cows adjacent to the monsters.
    - getCurrentScores(): Returns a String containing the current scores of JEF and the

player.

- getFinalScore(): Return playerScore - computerScore, which is the final score of the game.
- getGameBoard(): Returns the GameBoard object in Game.
- getHighScores(): Returns a String containing the list of the top 10 scores.
- getListOfScaredCows(): Returns the list keeping tack of how many times each cow has been scared.
- getLivingMonsters(): Calls the getLivingMonsters method in GameBoard to get a list of the indices of all living monsters.
- getMode(): Returns a STring indicating the current mode the game is in.
- isGameOver(): If there are no more living monsters or not more living cows, returns true.
- moveCow(int index1, int index2): Calls the GameBoard moveCow method to move a cow, given the index of the cow and the index of its desired location.
- newGame(): Resets the entire game to prepare for a new game.
- resetGame(): Resets the game (except for the mode and round and computerScore) to prepare for the second round.
- saveScores(String fileName): Saves the list of high scores to a file with the specified name.
- setName(String name): Sets the name of the player to be the given name.
- toString(): Calls the GameBoard toString method to print the gameBoard.
- updateSocres(): Updates the list of high scores and calls saveScores to save the updated list to the external txt file; only allows the players who have the top 10 scores to remain in this list.
- userChoosesMonster(int index): JEF randomly moves cows at the beginning of the game, does not change the mode because another method is provided to check whether the user is done choosing monsters.
- userDoneChoosingMonsters(): Changes the mode to userKillsCow.
- userDoneMovingCows(): CHanges the mode to computerChoosesMonsters.
- userGuessesMonster(int index): If the cow at the given index if allowed to be guessed (it is not dead or invisible) then it is killed. If the game is over, change the mode to gameOver and update the scores. If the cow was a monster or if JEF has no more moves the mode stays the same, but if the cow was not a monsters the mode changes to computerKillsCow.
- userKillsCow(int index): Kills a selected cow, if it is allowed to be killed, and makes the surrounding cows scared and updates the list of scared cows; if successful, changes the mode to userMovesScaredCows, and if the round is over, changes the mode to round1Over instead.
- userMovesCow(int index1, int index2): Calls the moveCow method to move a cow, but

does not change the mode because there is another method to do that.
    - userMoves ScaredCow(int index1, int index2): Given the index of the current cow and the index to move the cow to, moves a scared cow, and makes the cow no longer scared; when all of the scared cows have been moved, changes the mode to computerGuessesMonster.
    - updateListOfScaredCows(): Updates the list keeping track of how many times each cow has been scared.
- ComputerPlayer
    - Constructor: Creates a computer player associated with a certain instance of GameBoard.
    - chooseMonsters(): The computer determines the cows that are adjacent to the most number of cows and adjacent to the second to most number of cows and chooses three random cows from these two groups.
    - guessMonster(LinkedList<LinkedList<String>> numTimesScared): To guess a monster, the computer determines the group of cows that have been scared the most number of times and chooses a random cow from that group to guess.
    - killCow(): The computer determines the cows that are adjacent to most other cows and randomly kills one.
    - moveScaredCow(int index): The computer moves a cow to a random empty spot adjacent to the greatest possible number of other cows.
    - randomlyMoveCows(): At the beginning of the game, the computer moves 5-10 cows randomly.
    - getAllAdjacencies(LinkedList<Integer> aList): This helper method takes in a LinkedList<Integer> and finds the number of cows each is vertex is adjacent to.
    - removeEmptyLists(LinkedList<LinkedList<Integer>> aList): Helper method that removes any empty lists from the end of a LinkedList<LinkedList<Integer>>.
- GamePanel
    - makePanel methods: These methods make the necessary panels for the GUI.
    - Listeners: The listeners detect moves made by the user and perform the appropriate actions. We have two listeners: a ButtonListener and a MouseListener. The Button Listener listens to buttons that help move the game along to different modes and the MouseListener deals with the more interactive parts of the game (ie. moving cows, killing cows, etc…). We also use Timers in our GamePanel to deal with the ComputerPlayer moves.
    - updateGUIAppearanceJEFMonsters(): Changes the appearance of the GUI so that the user is able to see different icons. Does not show monsters unless they're dead.
    - updateGUIAppearanceUserMonsters(): Changes the appearance of the GUI in the first round so that the user is able to see different icons.
- GameGUI

- Main method to run the program.