Rush Hour

Brian Herman & Spencer Cleveland

Software Design Document

Table of Contents

Purpose
Scope
System Overview
Architectural Design Diagrams
Decomposition Description
Data Dictionary
Overview of User Interface
Requirements Matrix
Data File Format

Purpose

This software design document describes the architecture and system design of a Rush Hour Game with a hint function.

Scope

The program is to provide a graphic user interface to play Rush Hour. It is also to include a hint button, a solve button and a reset button which will allow the user to see the solution (or get a hint at the solution) and to try the puzzle again from the start. As the player moves the pieces around on the game, your program is to keep track of the "move count" or how many times a piece has been moved. Your program is to keep track of the minimum number of moves needed to solve a puzzle (i.e. fewest moves so far to solve the puzzle). Menu items for Exit, About and Help must exist.

System Overview

Our program consists of eight classes, a thread for the solver, a board class, a piece class, the Rushhour window class, a board class for the BFS, a piece class for the BFS, and a First Move class that initially makes the moves for the BFS.

Architectural Design Diagrams

Canvas inherits from JPanel (Allows us to paint on the screen)
RushHour inherits from JFrame (Creates a new window)
MainWindowListener implements ActionListener (Handles actions)
BackgroundSolver implements Runnable (Allows for the creation of a thread)
PanelMouseAdapter extends MouseAdapter (Allows for dragging of pieces)

Super Class JFrame
Class RushHour
Methods: Main - Starts up the RushHour frame Constructor **-** Sets up the RushHour frame **SubClasses** MainWindowListener Handles all MenuBar Events

Super Class JPanel
Class Canvas
Methods: paintComponent - Draws the Pieces onto the canvas Constructor **-** Sets up the Canvas Reset - resets the board load - loads a new board getHint - Returns the hint. **SubClasses** PanelMouseAdapter Handles all MouseClicks

Interface Runnable
Class BackgroundSolver2
Methods: Constructor **-** Sets up the Background Solver Run - runs the thread MakeFirstMoves - Makes the inital moves for the solver.

Decomposition Description

RushHour class

Method	**Description**
main	Starts the application and creates a new instance of the RushHour class.
Constructor	Sets up the window adds all menu items and adds an instance of the Canvas to the JFrame.
SubClass	
MainWindowListener	Handles all menu item events.

Canvas class

Method	**Description**
Constructor	Sets up the canvas.
paintComponent	paints the cars onto the JPanel
reset	resets the board
load(int i)	loads a built in level based on the index provided
load(File f)	loads a file based on the file provided
SubClass	**PanelMouseAdapter**
mousePressed	Captures the initial click
mouseReleased	Checks if the board is a winning board, displays a message if it is a winning one. Calculates the distance between each mouse click and change the board accordingly.

BackgroundSolver2 class

Method	Description
run	Starts the thread, performs the BFS algorithm.
Constructor	Sets up the window adds all menu items and adds an instance of the Canvas to the JFrame.
MakeFirstMoves	Recursively performs the BFS algorithm.
getHint	Returns the hint.

Piece class

Method	Description
Constructor	Sets up the piece
getX	returns the X value
getY	return the Y value
setX	Sets the X value and updates the rectangle
setY	Sets the Y value and updates the rectangle
direction	returns the orientation of the piece either VERTICAL or HORIZONTAL
length	returns the length of the piece
setBounds	sets the bounding rectangle
bounds	gets the bounding rectangle
isGoal	returns true if the piece is the goal piece.
whichSideClicked	determines which side was clicked either UP, DOWN, LEFT or RIGHT.

SearchBoard class

Method	Description
Constructor	Sets up the Board based on a piece list
getPiece	gets a piece at the provided index
MovesToWin	returns the number of moves to a winning state
getPositions	returns the number of positions there are available
setPositions	creates the number of positions that are available
setBoardString	creates a unique board string based on the positions of the pieces.
setBoardSize	Sets the size of the board.
isLegalBoard	Returns true if the board is a vaild board.
isWinningBoard	Returns true board is a winning board.
getBoard	Returns an array of all the search pieces
getMoveNumber	Returns the number of moves preformed.
moveOne	Moves a piece left or up
moveTwo	Moves a piece down or right.
generateBoards	Preforms the recursive calls on the board
isLegalMoveOne	Determines if a move is legal
isLegalMoveTwo	Determines if a move is legal

SearchPiece class

Method	**Description**
Constructor	Sets up the piece
getX	returns the X value
getY	return the Y value
setX	Sets the X value and updates the rectangle
setY	Sets the Y value and updates the rectangle
getPositions	gets the Number of Positions
setBoardSize	sets the board size
getBoardSize	returns the Board Size
getLength	returns the length of the piece
bounds	gets the bounding rectangle
getPieceType	returns the type of the piece 1=2horizontal 2=3horizontal 3=2vertical 4=3vertical
printPiece	prints a piece representation onto the screen.

FirstMove class

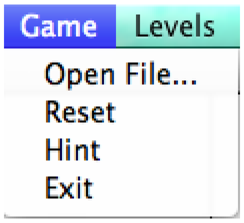
Method	**Description**
Constructor	Sets up the first move class
getBoardSize	returns the Board Size
setBoardSzie	sets the Board Size
getMoves	Return the number of preformed moves
getPiece	Gets the index of the piece being moved.
getDirection	returns the direction of the piece being moved
getSearchBoard	Returns the search board of the pieces being moved
getHistory	Returns the history of the moves preformed
isLegalMoveOne	Determines if that move is legal
isLegalMoveTwo	Determines if that move is legal
setBoardString	Generates the board String
generateBoards	Generates all the boards by preforming the BFS algorithm.

Data Dictionary see javadoc.

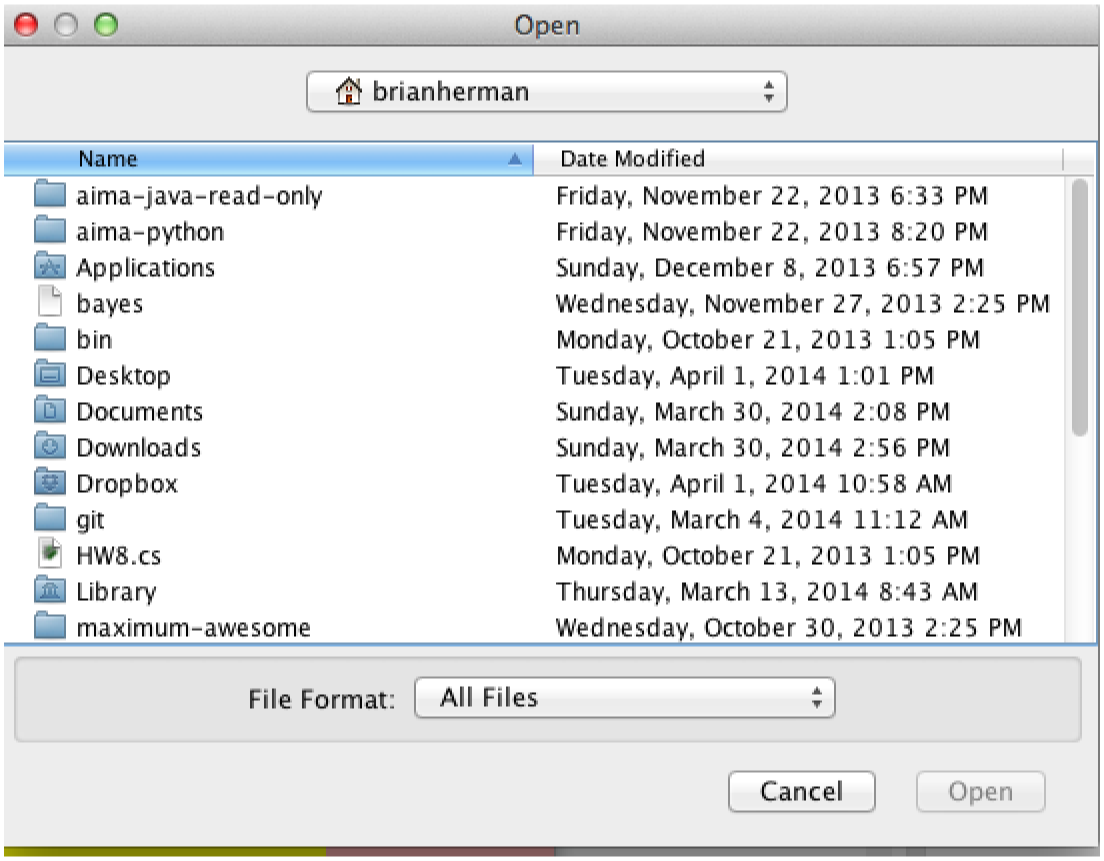
Overview of the GUI

The menubar has the following menu options

Game

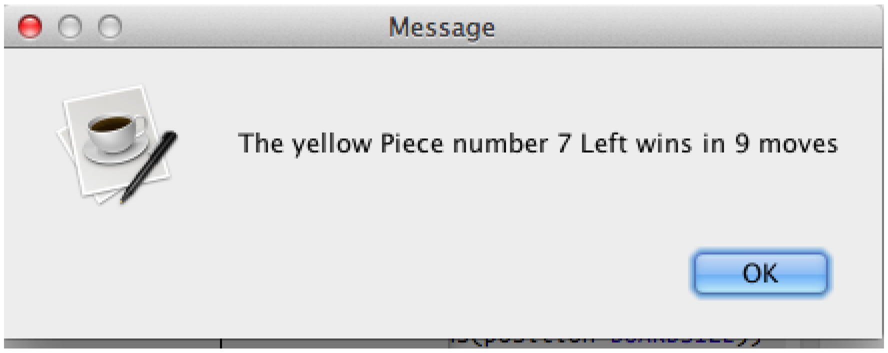


Open File opens the file chooser window.



Name	Date Modified
aima-java-read-only	Friday, November 22, 2013 6:33 PM
aima-python	Friday, November 22, 2013 8:20 PM
Applications	Sunday, December 8, 2013 6:57 PM
bayes	Wednesday, November 27, 2013 2:25 PM
bin	Monday, October 21, 2013 1:05 PM
Desktop	Tuesday, April 1, 2014 1:01 PM
Documents	Sunday, March 30, 2014 2:08 PM
Downloads	Sunday, March 30, 2014 2:56 PM
Dropbox	Tuesday, April 1, 2014 10:58 AM
git	Tuesday, March 4, 2014 11:12 AM
HW8.cs	Monday, October 21, 2013 1:05 PM
Library	Thursday, March 13, 2014 8:43 AM
maximum-awesome	Wednesday, October 30, 2013 2:25 PM

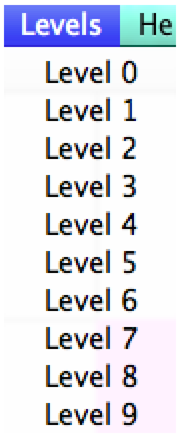
Reset resets the board to the inital state.



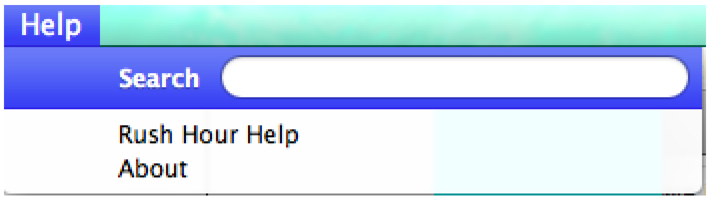
The yellow Piece number 7 Left wins in 9 moves

Example of the hint it gives you when you click the hint menu.

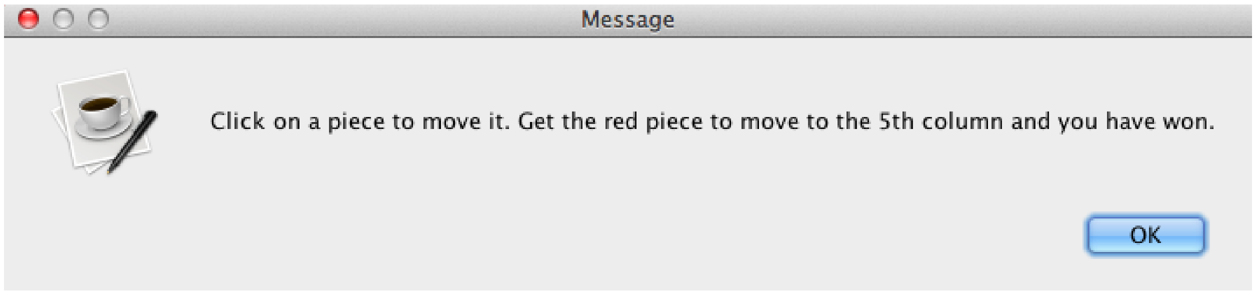
Levels menu, this allows you to choose a level.



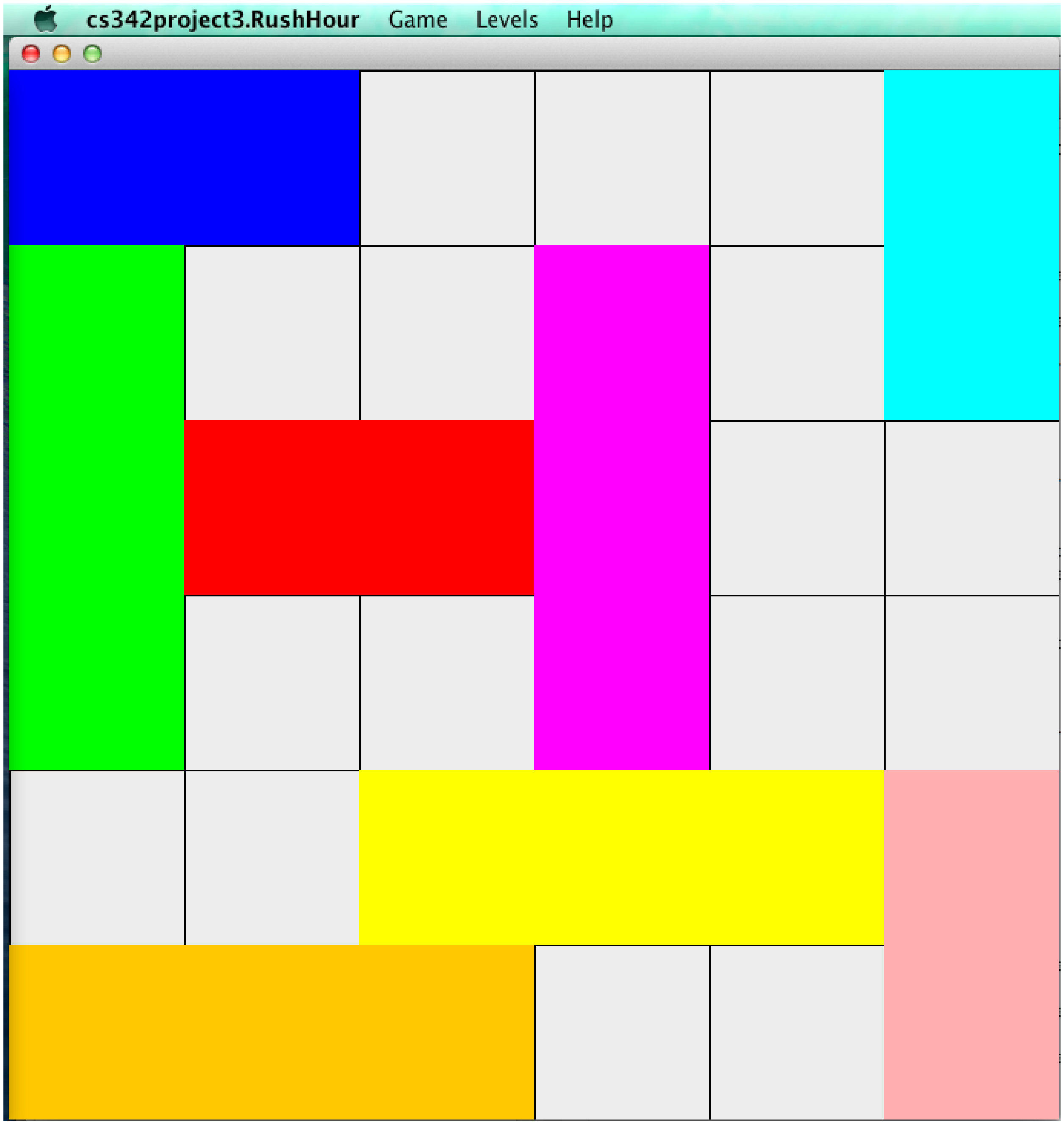
Help



Rush Hour Help Displays the following window.



Click on a piece to move it. Get the red piece to move to the 5th column and you have won.



To move a piece on the board click a piece and move it in the opposite direction you want to move it.

Requirements Matrix

Requirement	**Fulfilled**
Moving of Pieces, Drawing of Pieces onto the screen	True
Breadth First Search Algorithm	True
Running of BFS Search algorithm in the background	True
Reset Menu	True
Hint Menu	True
10 Different Levels One level is invalid this shows the levels are validated	True
One Level Is Unsolvable	True
Solve Menu	False
Allow loading of Custom Levels	True

Data File Format

6 6 this is the number of rows and the columns, each line after this is a piece
2 3 2 1 h 2 is the x value 3 is the y value 2 is the height and 1 is the width H=Horizontal
1 1 2 1 h
1 2 1 3 v V is for vertical
6 1 1 2 v
4 2 1 3 v
1 6 3 1 h
6 5 1 2 v
3 5 3 1 h

The first piece in the data file is the goal piece.