The Random Color Puzzle

CSC17C Project 1

Colleen Carleton

Write up

Introduction

The Random Color Puzzle is a game that creates a puzzle that the user must guide the character (denoted by 'c') through until they reach the end. Each step they take will place them on a different colored space, each of which has a different function. Some are impassible, and others can make colors impassible; this means that each step should be strategic to ensure the character does not get trapped in the puzzle. What color appears in each space is chosen randomly, and will differ each time the game is played. There are three modes; Easy, Medium, and Hard, and these modes determine the size of the puzzle that the user must navigate. Easy mode will create a puzzle of dimensions 3 rows by 5 columns; Medium mode will consist of a puzzle with dimensions of 5 rows by 5 columns, and Hard mode will create a puzzle of dimensions 7 rows by 10 columns.

The colors' functions are as follows:

Blue ('b') - can be crossed, but not if the color you were standing on previously was Orange ('o').

Orange ('o') - can be crossed, but has the effect that you cannot step on a blue space directly after stepping on an orange one.

Yellow ('y') - cannot be crossed, and will prevent you from stepping on a blue space that is directly in front of it.

Purple ('p') - can be crossed, and has no restrictions.

Green ('g') - can be crossed, and has no restrictions.

Red ('r') - cannot be crossed.

Controls

Movement through the puzzle is simple; pressing 'w' will cause the character to move up one position in the puzzle, pressing 'a' will cause the character to move backwards one position, pressing 's' will move the character down one position, and pressing 'd' will move the character forward one position.

Version History

(in order from earliest to latest)

Version 1.0

This version introduced the main components of the game, allowing the user to view the game information and play a basic version of the game, where none of the colors' effects were in place. This version introduced the modes, allowing the user to select the size of the puzzle, and creating the 2D array to store the puzzle accordingly. The mode was created using a structure, where the row and column sizes were the structure members.

Version 1.1

This version added in the functions of the various colors in the puzzle. An issue was noticed where the finish space could potentially be blocked completely by impassible tiles, which was to be fixed in the next version.

Version 1.2

This version prevented the finish and starting tiles from being blocked by impassible tiles. However, due to the random nature of the puzzle, it is still possible for there to be no possible way to reach the finish space, as impassible tiles can end up blocking all potential routes. This remains an issue with the program, though it seems to be relatively rare.

Version 1.3

This version added in scores; each score was calculated by the number of moves made less than a certain amount that the user completed the puzzle in, then multiplying the result by a number. For Easy mode, the score was multiplied by 5; for Medium, 10; and for Hard, 20. This version also included dynamic memory allocation, reading and writing to a binary file, and passing structures and files between functions. An issue arose with reading information back in from the binary file, where it seemingly read the entire file in for the first person’s information; following that, it output the same series of numbers as was output for the score of the first player, and ignored the name completely. This continues to be an issue when playing more than one time in a single run of the program.

Version 1.4

This version added the ability to save a puzzle and resume it later, using a class, SaveLoad, to save all relevant information to a file, which could later be read in to re-create the puzzle exactly as it had been. The Exceptions class was also added, so that when a game was saved/quit, the program would stop running that round of the game and return to main. This version included overloaded operators and overloaded functions in a class. Some issues arose, such as that the number of games would still increment for quit games, even though there was no information to output at the end. Furthermore, the score was not saving correctly to the file, an issue fixed in the following version.

Version 1.5

This version fixed the previously mentioned issue of the incorrect score being saved to the file when saving a game. It also ensured the character representing the user’s position in the puzzle appeared in the correct place in the puzzle when resuming a game, having before returned the character to the beginning of the puzzle.

Version 1.6

This version altered the way the number of games was incremented so the correct number of total games played was used to output the names and scores of players. This involved inclusion of inheritance of a class which stored the number of quit games.

Version 1.7

This version made some final adjustments and additions to the program, though there are still a few issues that remain, including that the scores of subsequent players are replaced by the score of the first player, and the names of subsequent players often end up partially cut off or do not appear at all in the final output.

Version 1.8

This version added a Linked List to store the player’s score.

Pseudocode

Main

//System Libraries

//Input/output

//File input/output

//For random number seed

//Time

//Case conversions

//String library

//Output manipulation

//User Libraries

//Structures

//Save/Load class

//Exceptions class

//Class to store number of games

//Class to store number of quit games

//Global Constants

//Such as PI, Vc, -> Math/Science values

//as well as conversions from system of units to

//another

//Function Prototypes

//Displays game information, controls, etc

//Plays game

//Plays saved game

//Fills array with random characters

//Executable code begins here!!!

//Declare variables

//Player's selection from the menu

//Does player want to continue?

//Binary file to store names and scores

// Structure returned from function

//Stores number of games played

//Holds quit games

//Stores a saved game

//Used to check number of games played

//Stores number of quit games

//Temp storage for quit games

//Temp variable

//Exceptions

//Open binfile for output

//Open save/load file for input. If user wants to save a game, it will be closed and reopened for output.

//Set random number seed

//Get player's choice

//Start game or display info, depending on choice

//If game was quit, increment quit games

//Do they want to play again/continue?

//Close binary file

//Open binary file for input

//Dynamically allocate memory for array

//Until end of file, read in and output information

//Output information

//Close binary file

//Delete dynamic memory

//Exit

//Function 1 – Display game info

//Declare Variable

//Stores player's choice

//Display info

//Display character/color information

//Display controls

//Display mode info

//Function 2 – Play game

//Declare variables

//Stores mode

//Opens puzzle-storing file

//Stores the characters as they are input from the file

//Stores character's movements

//Holds character's temporary location (rows)

//Holds character's temporary location (columns)

//Stores size of rows

//Stores size of columns

//Stores number of points

//Stores overall score

//Stores size of array to hold name/length of name

//Stores user's name

//Used to save game

//Used to throw exceptions

//Stores position of c in puzzle (rows)

//stores position of c in puzzle (columns)

//Structure to store user's name and score

// Get player's mode

//Set up array size based on mode

//Open file for input

//Create array to store the colors

//Fill array

//Clear eof flag, return to beginning

//Create second array to store character's position in array

//Output the array, but with c in the character's position

//Clear eof flag, return to beginning

//Start game

//Get user's first movement

//If quitting game

//Close and Open file for output

//Get location of 'c'

//Call savegame

//End current game

//Otherwise

//Move character according to user's input

//Store position in rows

//Store position in columns

//Input validation

//Movement = w

//Place 'c' in new location

//Clear old location of 'c'

//Movement = a

//Place 'c' in new location

//Clear old location of 'c'

//Movement = s

//Place 'c' in new location

//Clear old location of 'c'

//Movement = d

//Place 'c' in new location

//Clear old location of 'c'

//Ouput new position in puzzle

//Clear eof, return to beginning

//Once they reach the end, congratulate them

//Calculate score

//Display score

//Put score in structure

//Get user's name

//Input to file

//Close file

//Catch exception if game was saved/quit

return player;

Function 3 – Play saved game

//Declare variables

//Stores mode

//Stores the characters as they are input from the file

//Stores character's movements

//Holds character's temporary location (rows)

//Holds character's temporary location (columns)

//Stores size of rows

//Stores size of columns

//Stores number of points

//Stores overall score

//Stores size of array to hold name/length of name

//Stores user's name

//Used to save game

//Used to throw exceptions

//Stores position of c in puzzle (rows)

//stores position of c in puzzle (columns)

//Opens puzzle-storing file

//Stores info from loaded game

//Get saved game info

//Set rowsize and colsize

//Create array to store the colors

//Fill array

//Get character from file

//Places character in array

//Places character in file

//Clear eof flag, return to beginning

//Clear eof flag, return to beginning

//Create second array to store character's position in array

//Output the array, but with c in the character's position

//Clear eof flag, return to beginning

//Start game

//Get user's first movement

//If quitting game

//Close and Open file for output

//Get location of 'c'

//Call savegame

//End current game

//Otherwise

//Move character according to user's input

//Store position in rows

//Store position in columns

//Input validation

//Movement = w

//Place 'c' in new location

//Clear old location of 'c'

//Movement = a

//Place 'c' in new location

//Clear old location of 'c'

//Movement = s

//Place 'c' in new location

//Clear old location of 'c'

//Movement = d

//Place 'c' in new location

//Clear old location of 'c'

//Ouput new position in puzzle

//Clear eof, return to beginning

//Once they reach the end, congratulate them

//Calculate score

//Display score

//Put score in structure

//Get user's name

//Input to file

//Close file

//Catch exception if game was saved/quit

return player;

Function 4 – Fill array with random characters

//declare variables

//Stores random number

//Opens file for output

//Create Array

//Fill array with random colors

//Get random number

//Random number from 1-8

//Get character

//If 1

//If 2

//If 3

//If 4

//If 5

//If 6

//If 7

//If 8

//Preventing finish being blocked off

//Preventing start from being blocked off

//Change last element in array to f

//Write array to file

//Close file for output

Classes/Structures

Mode.h

//Struct Mode

//Size of rows

//Size of columns

//Struct UsrInfo

//Array to store name

//Stores total score

//Struct Scores

//Thousands

//Hundreds

//Tens

//Ones

//Struct Load

//mode

//row position

//col position

//Score (thousands)

//Score (hundreds)

//Score (tens)

//Score (ones)

SaveLoad.h

//Functions

//Save game

//Load game

//intochr (returns char)

//intochr (overloaded, returns Scores)

//Overloaded = operator

SaveLoad.cpp

//Functions

//intochr (char)

//Coverts an int to a char

//Overloaded intochr, for the score

//Separate score

//return scrs

//Save Game

//Variables

//Stores a character from the file

//row position

//col position

//stores Scores

//Store Mode

//Store puzzle

//Set position of 'c'

//Store score

//Load game

//Variables

//stores game info

//mode

//inputs a character value

//string for input

//Get mode

//skip to after first & symbol

//Get row position of c

//Skip next &

//Get score

//return info

Exceptions.h

//Class excepts

//Class to throw if game is quit

//Function to throw class

Exceptions.cpp

//Function to throw class

//Throw class

GmeInfo.h

//Class GmeInfo

//Private

//variable totgms

//Public

//Constructor

//Member functions

//Get

//Set

//virtual ++ operator

GmeInfo.cpp

//Get function

//return totgms

//Set function

//set totgms

//++ operator

//increment totgms;

QuitGIn.h

//Class QuitGIn

//private

//variable quitGms

//public

//Constructor

//Member functions

//Get

//Set

//Overloaded ++ operator

QuitGIn.cpp

//Get function

//return quitGms

//Set function

//set quitGms

//++ operator

//increment quitGms