**Software Design & Development HSC: Major Programming Project**

ProblemDefinition

The problem is to develop a command line software solution which emulates the game known as Mastermind. The needs of the client include the defining and understanding and planning and designing stages of this software solution to be completed by the 8th of March 2016.

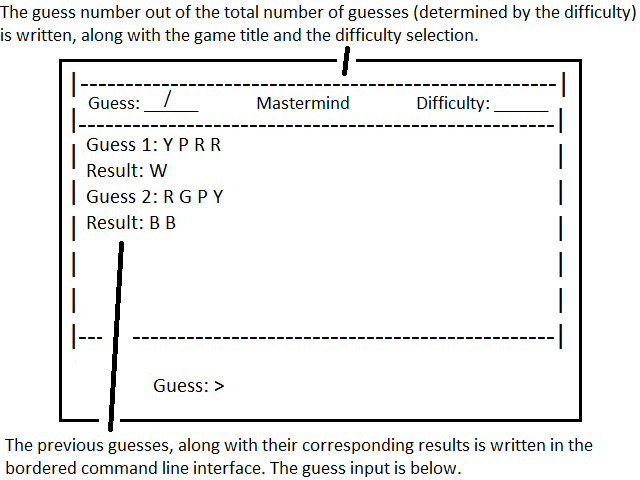
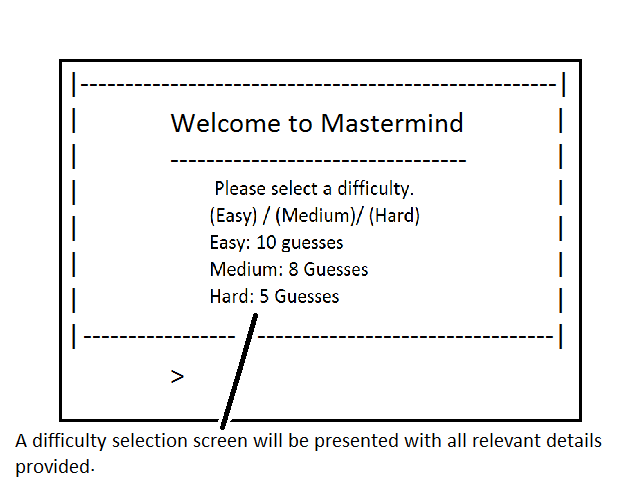
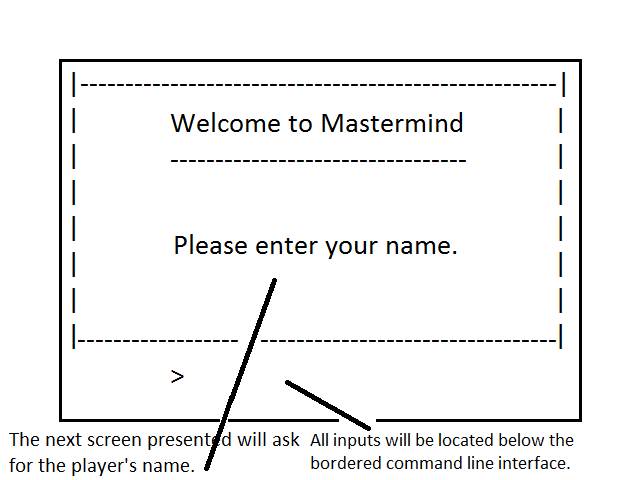
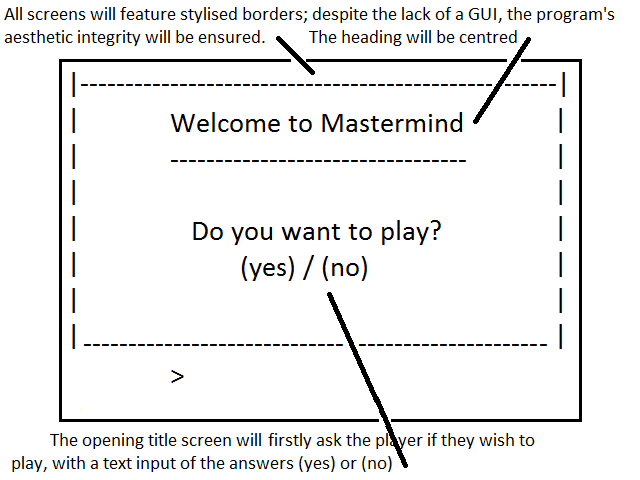
The functionality requirements consist of the successful emulation of the game, with the rules as follows: a random combination of 4 colours from red, green, blue, yellow and purple are selected as the ‘code to be broken’. The player has a certain number of chances to select the correct colours in the correct order, depending on the selected difficulty (easy: 10 turns, medium: 8 turns, hard: 5 turns). However, if the player guesses a right colour in the wrong location, a W will be displayed. If the player guesses a right colour in the right location, a B will be displayed. Multiple W’s and B’s are permitted. W stands for a white peg and B stands for a black peg. The player will input their guesses via text input and the command-line interface will record the user’s inputs and output the corresponding results of white or black pegs. If the player does not correctly guess all 4 pegs’ correct colours and correct locations in the allotted number of guesses, the computer is deemed the winner.

It must be developed in the Python programming language and utilise classes and subprograms (i.e. an object oriented approach) and must also access an external text file for the data used in the program. The software solution must be compatible with the client’s version of python. It must also have quick performance, with minimal delay between guess inputs and resulting outputs. This will be achieved by careful consideration of algorithm design and minimal use of unnecessary loops and global variables. The boundaries of the software solution include limiting input from the user to solely keyboard-based input for the guesses, due to the solution’s command line nature. The output only consists of text-based output to the command-line interface.

Gantt Chart

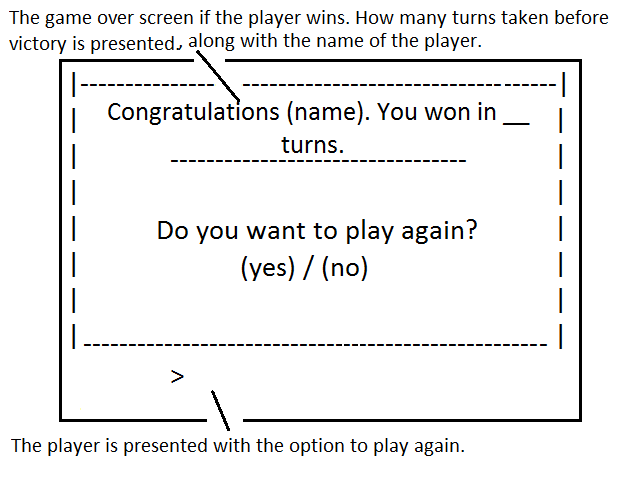
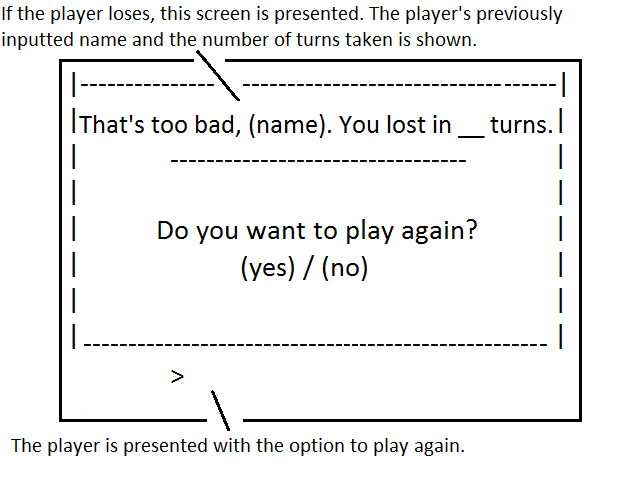


Storyboard



(If player wins)

(If player loses)



Context Diagram



Structure Chart



System Flowchart



IPO Chart

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| Yes/No | Start program |  |
| Player name | Store name in memory |  |
| Difficulty | Store difficulty in memory |  |
| Available colours | Generate combination and store in memory |  |
|  | Initialise interface |  |
| Guess | Check if guess is correct |  |
|  | Generate guess result |  |
|  | Check if guess number limit has been met |  |
|  | Update interface | Guess result |
|  | Calculate number of turns taken for victory/loss | Victory/loss game over screen |

Data Dictionary

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | **Data Type** | **Scope** | **Description** |
| WantsToPlay | Boolean | Local | Holding the initial input checking if the player wishes to play |
| PlayerName | String | Local | The player’s inputted name |
| PlayerDifficulty | Integer | Local | The player’s selected difficulty: easy (0), medium (1), hard (2) |
| GuessesAvailable | Integer | Local | The number of guesses available to the user: easy (10), medium (8), hard (5) |
| ColoursAvailable | Array | Local | An array holding the colours available for the combination, and their respective identification letters (e.g. b = blue) |
| CorrectCombination | Array | Local | An array holding the correct combination to be guessed by the user |
| Guess | String | Local | The user’s inputted guess |
| GuessCorrect | Boolean | Local | The result after checking if the user has guessed the correct combination |
| GuessNum | Integer | Local | The number of guesses the user has made |
| GuessExpired | Boolean | Local | The result after checking if the user has made their allotted number of guesses |
| GuessResult | String | Local | The resulting output for the user’s guess (i.e. presence of black or white pegs) |

Algorithm

BEGIN MAIN

Initialise GAME object: //the application is created as an object, so local variables of the application can be accessed across functions, without utilising global variables

BEGIN PlayMastermind:

Initialise GAME.difficulties with: //as a dictionary

Easy corresponding to 10

Medium corresponding to 8

Hard corresponding to 5

Call InputUserInfo

Call GenerateCombination

END PlayMastermind

BEGIN InputUserInfo:

Print Initial Interface

Input WantsToPlay

IF WantsToPlay = True:

Input PlayerName to GAME.PlayerName //GAME.x refers to variables attributed to the GAME object

Input PlayerDifficulty to GAME.PlayerDifficulty

IF WantsToPlay = False:

Goto END

ENDIF

END InputUserInfo

BEGIN GenerateCombination:

Access colours.txt

Input colours.txt to ColoursAvailable //created as an array Randomly select 4 elements from ColoursAvailable //repetitions accepted

Set GAME.CorrectCombination to 4 elements

END GenerateCombination

BEGIN PlayGame:

Print Initial Interface

Set GuessNum to 0

Set GuessCorrect to 0

Set GuessExpired to 0

WHILE GuessCorrect ≠ 1 and GuessExpired ≠ 1: //while game has not ended

Input guess to Guess

Add 1 to GuessNum

IF Guess = GAME.CorrectCombination:

Set GuessCorrect to 1

ELSE:

Set RightColourRightPlace to 0

Set RightColourWrongPlace to 0

FOR i = 0 to length of CorrectCombination:

FOR j = 0 to length of Guess:

IF Guess[j] = CorrectCombination[i] and i = j:

Add 1 to RightColourRightPlace

ELSEIF Guess[j] = CorrectCombination[i]:

Add 1 to RightColourWrongPlace

ENDIF

ENDFOR

ENDFOR

IF RightColourRightPlace > 0: //if any pegs are in the right colour right place

Print B, (RightColourRightPlace) times //symbolising the number of black pegs the player ‘receives’

ENDIF

IF RightColourWrongPlace > 0:

Print W, (RightColourWrongPlace) times //symbolising the number of white pegs the player ‘receives’

ENDIF

IF GuessNum = GAME.Difficulties[GAME.PlayerDifficulty]:

//If the current guess number is equal to the maximum guess (determined by the difficulty)

Set GuessExpired to 1

ENDIF

ENDIF

END WHILE

IF GuessExpired = 1:

Print Game Over Screen

ENDIF

IF GuessCorrect:

Print Game Victory Screen

ENDIF

END PlayGame

END MAIN

Note: This program has no need for class-based functionality, and its inclusion would only complicate the algorithm, thereby increasing computation time.

Log Book

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Date** | **Task that was attempted** | **How the task was achieved** | **The end result of the task** | **Follow up required** | **Signature and dated by teacher** |
| 10/2/16 | Writing the problem definition | Utilising Microsoft Word, I wrote the problem definition; considering all aspects of the solution. | The problem definition has been fully written. | Possibly |  |
| 10/2/16 | Constructing the Gantt chart | Utilising Microsoft Excel, I created the Gantt chart; factoring in time constraints of each deadline. | The Gantt chart has been completed. | No |  |
| 13/2/16 | Drawing up storyboard | I initially considered what type of interface I would seek; also utilising my prior knowledge of the Python programming language in order to develop a storyboard of the interface achievable in Python with my skillset. | The storyboard has been completed. | No |  |
| 16/2/16 | Rewrote problem definition | After consulting with my peers, I realized I did not include all aspects of the problem definition, consisting of: the needs of the client, functionality requirements, compatibility issues and performance issues. I believe I have finished the problem definition now. | The problem definition has been rewritten. | No |  |
| 17/2/16 | Created context diagram | Utilising the free to use website [www.draw.io](http://www.draw.io), I created the context diagram of the software solution. | The context diagram has been completed. | Possibly |  |
| 17/2/16 | Created structure chart | Utilising the same website, [www.draw.io](http://www.draw.io), I created the structure chart. I had to brush up on my knowledge of the syntactical arrangement of structure charts prior to its construction. | The structure chart has been completed. | Possibly |  |
| 17/2/16 | Added on to context diagram | I realised I left out several aspects of data that flowed between the user and the system in my context diagram, so I added them. | The context diagram has been finished. | No |  |
| 17/2/16 | Started IPO chart | I have created the IPO chart, utilising my previously completed structure chart to outline inputs, processes and outputs. | The IPO chart is finished. | No |  |
| 19/2/16 | Streamline structure chart’s mainline | After presenting the structure chart, I have been made aware that several subroutines are unnecessary and may clutter the mainline. As such, I have consolidated several subroutines to improve performance. | The structure chart has been reconfigured. | Possibly |  |
| 21/2/16 | First draft of the system flowchart | I have created the first draft of the system flowchart; however I have questions about the diagram’s syntax. I will most likely need to update the diagram later. | The first draft of the system flowchart has been completed | No |  |
| 23/2/16 | Fixed error on structure chart | I realised there was a parameter being passed from a subroutine that would not actually exist in the subroutine, so I fixed the error by removing the parameter on the structure chart. | The structure chart has been updated | No |  |
| 23/2/16 | Wrote the data dictionary | Utilising the structure chart, I created the data dictionary of the data that will be used. | The data dictionary has been created. | No |  |
| 29/2/16 | First draft of algorithm | Utilising Microsoft Word, I created the solution’s algorithm in pseudocode. | The first draft has been completed. | No |  |
| 4/3/16 | Finalised defining and understanding and planning and designing documentation | I went through the marking scheme and ensured all documentation would be deemed acceptable. Some corrections were made to various aspects. | The documentation has been finalised. | No |  |
| 4/3/16 | Printed out documentation | The documentation has been printed out, ready for submission. | The documentation has been printed out. | No |  |