Maze Game Testing documentation

*Test plan - p1*

*Testing - p8*

*Test Plan*

Testing Methods

The test plan will have two phases. The first of these two phases will be component testing. Upon completion of each module/set of functions, the module/set of functions will be tested to meet all criteria. The second phase will be integration testing. This will be testing how the modules/functions run together and checking for any bugs once all modules/functions are integrated to create the full program.

Input validation will be tested alongside other tests throughout all stages of testing, using the same results table. Input validation includes testing that:

* Nothing happens when a user inputs a click outside of a Rect() in the main menu
* Nothing happens in the maze when a user inputs anything other than the ‘W’,’A’,’S’,’D’ and ‘Esc’ keys
* Nothing happens in the in-game menu unless a user inputs a click inside any of the button Rect()s associated

The results of this test plan for component testing will be recorded in the following table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function/class to be tested** | **Input** | **Expected Outcome (Deliverable being tested)** | **Actual outcome** | **Pass/Fail** | **Further Action** |
|  |  |  |  |  |  |

The results of this test plan for integration testing will be recorded in the following table:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Stage** | **Input** | **Expected Outcome** | **What is being tested** | **Actual outcome** | **Pass/Fail** | **Further Action** |
|  |  |  |  |  |  |  |

The results of this test plan for input validation testing will be recorded in the following table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Expected Outcome** | **Actual outcome** | **Pass/Fail** | **Further Action** |
|  |  |  |  |  |

Test Plan

*Main Menu module component test plan - to be completed upon the the main menu function being completed*

|  |  |  |
| --- | --- | --- |
| **Component to be tested** | **Input** | **Expected Outcome (Deliverable being tested)** |
| Start Button | Click within the rectangular button with ‘Start’ text in it. | When clicked:  Until Main Game function has been developed, the screen should clear and an error message should be displayed saying ‘ Main Game is under construction’ or similar. |
| Continue Button | Click within the rectangular button with ‘Continue’ text in it. | When clicked:  Until Main Game function and Load function had been fully developed, the screen should clear and an error message should be displayed saying ‘Main Game is under construction’ or similar |
| Quit Button | Click within the rectangular button with ‘Quit’ text in it. | When clicked:  Pygame and python should uninitialize, closing the game. |

*Main Menu Input Validation test plan - to be completed upon the main menu function being completed.*

|  |  |
| --- | --- |
| **Input** | **Expected Outcome** |
| Click within the rectangular button with ‘Start’ text in it. | (See component testing above) |
| Click within the rectangular button with ‘Continue’ text in it. | (See component testing above) |
| Click within the rectangular button with ‘Quit’ text in it. | (See component testing above) |
| Click anywhere outside of the rectangular buttons in three separate places | Nothing will happen, screen will remain the same. |
| Press down the ‘Esc’ button and two other keys. | Nothing will happen, screen will remain the same. |

*Maze component test plan - to be completed upon the maze class being completed*

|  |  |  |
| --- | --- | --- |
| **Component to be tested** | **Input** | **Expected Outcome (Deliverable being tested)** |
| Maze initialisation  (Unrendered maze) | Ask class to print unrendered maze upon initialisation. (InCode) | A 2D array of 1s and 0s being printed in the shell, representative of the maze to be drawn |
| Maze initialisation  (Rows and columns) | Ask class to print rows and columns of class upon initialisation (InCode) | Two Integers being printed in shell, representative of the rows and columns of the maze to be drawn |
| Maze Initialisation (Box width/height) | Ask class to print width and height of boxes of class upon initialisation (InCode) | Printing two integers to the shell, representative of the width and height of each box |
| Maze drawing | Call the draw maze function, to see if it draws maze correctly. (In draw maze function print an array of the boxes to shell, to see if boxes have been stored for collision) | Unrendered maze of the maze class should be drawn out onto the screen in blue boxes. Maze should fit the screen fully and it should be an accurate representation of the unrendered maze. In the shell a list of Rect Objects representative of wall placements should be printed. |

Input validation testing for this component is not required as the user does not directly interact with the maze class

*After component testing of maze class, remove all in code testing aspects/print statements from the code.*

*Player Movement/Spawn component test plan - to be completed upon the player class being completed*

|  |  |  |
| --- | --- | --- |
| **Component to be tested** | **Input** | **Expected Outcome (Deliverable being tested)** |
| Spawn | Create player instance and use spawn class function in code. | Red block (Player) should appear at x=0 y=0 |
| Movement (right) | Once player has spawned, press right arrow on keyboard | Red block should move one space to the right. |
| Movement (Up) | Once player has spawned, press Up arrow on keyboard | Red block should move one space to the up. |
| Movement (left) | Once player has spawned, press left arrow on keyboard | Red block should move one space to the left |
| Movement (Down) | Once player has spawned, press down arrow on keyboard | Red block should move one space down. |

*Collision detection component testing/Input validation for walls - to be completed upon both the player and maze components of the game being completed*

|  |  |  |
| --- | --- | --- |
| **Component to be tested** | **Input** | **Expected Outcome (Deliverable being tested)** |
| Collision detection (‘Wall’ blocks) | Attempt to move the red block (player block) into a blue block using the left, right, up or down movement keys. (See movement testing) | The red block should not move. |
| Collision detection (‘Winning’ the maze) | Disable collision detection for walls temporarily and move block to the edge of the screen. (see movement testing and remember to turn collision detection back on afterwards) | Screen should clear and white text saying “You win!” should appear. |

*Further player Input validation test planning - To be completed upon the player class being completed*

|  |  |
| --- | --- |
| **Input** | **Expected Outcome** |
| Press Left Arrow Key once player has spawned | (See component testing above) |
| Press Right Arrow Key once player has spawned | (See component testing above) |
| Press Up Arrow Key once player has spawned | (See component testing above) |
| Press Down Arrow key once player has spawned | (See component testing above) |
| Press any three buttons on the keyboard other than Left Arrow, Right Arrow, Down Arrow, Up Arrow | Nothing should happen. Screen should remain the same. |
| Click any three places on the screen. | Nothing should happen. Screen should remain the same. |

*In-Game Menu component test planning - To be completed upon the in game menu class, player class and maze class has been completed*

|  |  |  |
| --- | --- | --- |
| **Component to be tested** | **Input** | **Expected Outcome (Deliverable being tested)** |
| Open menu | Press ‘Esc’ on keyboard whilst maze is open | A menu should open over the maze which has two buttons: ‘SAVE’ and ‘QUIT’ |
| Close Menu | Press ‘Esc’ on keyboard after the open menu component has been activated | The menu should close and the maze should be revealed again. Buttons that were previously clickable should no longer be |
| Save Button | Click within the save button | Nothing should appear to happen on the screen but a file called ‘sav.txt’ should appear in the folder with the program. Nothing should be in this until the save/load features have been implemented |
| Quit Button | Click within quit button | Until all components have been integrated, button should clear screen to black. |

*In-Game menu Input validation test plan - to be completed upon maze class, player class and ingame menu class being completed*

|  |  |
| --- | --- |
| **Input** | **Expected Outcome** |
| Click within the borders of the menu when its open, but not within any of the buttons. | Nothing should happen, the screen should remain the same. |
| Press Esc when menu is not open | (See component testing) |
| Press Esc when menu is open | (See component testing) |
| Click within borders of save button | (See component testing) |
| Click within borders of quit button | (See component testing) |
| Press Up arrow, Down Arrow, Left Arrow, Right Arrow on keyboard. | Nothing should happen, player should not move, screen should remain the same. |
| Press anything on the keyboard that isn’t the above keys. | Nothing should happen, screen should remain the same. |

*Save/Load feature component testing - To be completed upon in-game menu, main menu and save/load feature being completed*

|  |  |  |
| --- | --- | --- |
| **Component to be tested** | **Input** | **Expected Outcome (Deliverable being tested)** |
| Save feature | Press ‘save’ on in-game menu | Until integration is carried out, this should create a sav.txt file, which has PLACEHOLDER written in it. |
| Load feature | Press ‘load’ on main menu | Until integration is completed, this should print the contents of the sav.txt file to shell. |

No input validation required, as user does not directly interact with save/load feature components.

*Integration Testing - Should only be completed once program modules, functions and classes have been fully integrated*

This integration testing will be completed on both a Windows 7 OS and Windows 10 OS, to ensure all requirements specifications have been met.

Integration testing will be carried out in 3 Stages, based on the part of the game being tested. The following will be tested:

* Starting the game
* Saving/Loading the game
* Winning the game

For this software to be successful it must be able to:

* Start the program
* Render a maze
* Have a working sprite which can move left,right, up and down
* Be able to save the maze ID and player position
* Be able to load a maze ID and player position
* Have a main menu with Start, Continue and Quit.

The below test plan should ensure all of this is true.

|  |  |  |  |
| --- | --- | --- | --- |
| **Stage** | **Input** | **What is being tested** | **Expected Outcome** |
| Starting (1) | Start program | Whether program can start on its own | There should be a menu on screen that has the title ‘Maze’, and has three buttons; Start, Continue and Quit. |
| Starting (2) | Press Start button on screen | Whether the main menu can lead onto the main game (Integration between two modules) | A maze loads and a player (Red Box) is spawned in the first space where there is no blue box. |
| Starting (3) | Press Right Arrow Key | Whether player can move right | Red box should move one space to the right. |
| Starting (4) | Press Left Arrow Key | Whether player can move left | Red box should move one space to the left. |
| Starting (5) | Press Down arrow key | Whether player can move down | Red box should move one space downwards. |
| Starting (6) | Press Up arrow key | Whether player can move up | Red box should move one space upwards. |
| Starting (7) | Attempt to move the red box into a block where a blue box is using the up, right, left or down keys. | Whether collision detection works | Red box should remain in the same place. |
| Save/Load (1) | Press ‘Esc’ | Whether in-game menu can be opened | Menu should open in the centre of the screen with two buttons saying ‘SAVE’ or ‘QUIT’ |
| Save/Load (2) | Press ‘Esc’ | Whether in-game menu can be closed | Menu should disappear revealing the maze and player (if player is behind in-game menu) |
| Save/Load (3) | Move player to a place that is not the spawn block. Press ‘Esc’ and click the ‘Save’ button on in-game menu. | Whether data can be saved. | Upon save, nothing should appear to happen but a ‘sav.txt’ file should be created in the same folder in which the program is. This file should contain the ‘maze ID’, player x coordinate and player y coordinate. |
| Save/Load (4) | Press ‘quit’ button on in-game menu. | Whether pressing the quit button takes you back to the main menu (Integration between main game and main menu) | The screen should clear, and the main menu should appear again and work. |
| Save/Load (5) | Press the ‘continue’ button | Whether pressing the continue button loads your save game correctly  (Integration between main menu, main game and load game) | The screen should clear and the maze that the user was in previously should generate and player appear in previous player position. |
| Winning (1) | Lead the player block to the maze’s exit | Whether the player can win | The player block should leave the screen and the maze should clear to a black screen and white text display saying ‘You win!’, wait a few seconds, clear the screen and head back to the main menu |
| Winning (2) | Press the quit button | Whether the game can be quit. | The game should uninitialize and close. |

Furthermore, this game will be given to end-users alongside some documentation with controls. The users will be asked about:

* Appropriateness of interface - Do the buttons aid your use of the program? Is there anything that could be done better in the UI?
* Customisation - What customisable options do you want in the future?
* Accessibility - Are there any added features you feel could be added for those who are impaired?
* Consistency - Would you say the interface is consistent?
* Control - Did you feel as though you did not have the ability to go back on your decisions?
* Helpfulness - Where do you feel there could be more help?

Keep in mind that this game will be considered in the very first stages of beta testing when it reaches end-user testing, as there is not much time to complete this game.

Only three people will be given the game to be tested to save time. This project will most likely have further work done on it past the date which the SQA picks the project up.

*Main menu component testing iteration 1*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function/class to be tested** | **Input** | **Expected Outcome (Deliverable being tested)** | **Actual outcome** | **Pass/Fail** | **Further Action** |
| Start Button | Click within the rectangular button with ‘Start’ text in it. | When clicked:  Until Main Game function has been developed, the screen should clear and an error message should be displayed saying ‘Main Game is under construction’ or similar. | Screen clears and error message displays saying ‘Main Game is under construction’ | PASS | N.A |
| Continue Button | Click within the rectangular button with ‘Continue’ text in it. | When clicked:  Until Main Game function and Load function had been fully developed, the screen should clear and an error message should be displayed saying ‘Main Game is under construction’ or similar | Pygame and python uninitialized, closing the game. | **FAIL** | Input validation error, Quit Button Rect() is larger than the visual part of the button. Change Rect() size of quit button |
| Quit Button | Click within the rectangular button with ‘Quit’ text in it. | When clicked:  Pygame and python should uninitialize, closing the game. | Pygame and python uninitialized, closing the game. | PASS | N.A |

*Main menu component testing iteration 2*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function/class to be tested** | **Input** | **Expected Outcome (Deliverable being tested)** | **Actual outcome** | **Pass/Fail** | **Further Action** |
| Start Button | Click within the rectangular button with ‘Start’ text in it. | When clicked:  Until Main Game function has been developed, the screen should clear and an error message should be displayed saying ‘Main Game is under construction’ or similar. | Screen clears and error message displays saying ‘Main Game is under construction’ | PASS | N.A |
| Continue Button | Click within the rectangular button with ‘Continue’ text in it. | When clicked:  Until Main Game function and Load function had been fully developed, the screen should clear and an error message should be displayed saying ‘Main Game is under construction’ or similar | Pygame and python uninitialized, closing the game. | PASS | N.A |
| Quit Button | Click within the rectangular button with ‘Quit’ text in it. | When clicked:  Pygame and python should uninitialize, closing the game. | Pygame and python uninitialized, closing the game. | PASS | N.A |

*Main Menu Input validation testing iteration 1*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Expected Outcome** | **Actual outcome** | **Pass/Fail** | **Further Action** |
| Click within the rectangular button with ‘Start’ text in it. | (See component testing above) | (See component testing above) | PASS | N.A |
| Click within the rectangular button with ‘Continue’ text in it. | (See component testing above) | (See component testing above) | PASS | N.A |
| Click within the rectangular button with ‘Quit’ text in it. | (See component testing above) | (See component testing above) | PASS | N.A |
| Click anywhere outside of the rectangular buttons in three separate places | Nothing will happen, screen will remain the same. | Nothing happened, screen remained the same. | PASS | N.A |
| Press down the ‘Esc’ button and two other keys. | Nothing will happen, screen will remain the same. | Nothing happened, screen remained the same. | PASS | N.A |

*Maze class component testing iteration 1*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function/class to be tested** | **Input** | **Expected Outcome (Deliverable being tested)** | **Actual outcome** | **Pass/Fail** | **Further Action** |
| Maze initialisation  (Unrendered maze) | Ask class to print unrendered maze upon initialisation. (InCode) | A 2D array of 1s and 0s being printed in the shell, representative of the maze to be drawn | 2D array of 1s and 0s was printed to the shell | PASS | N.A |
| Maze initialisation  (Rows and columns) | Ask class to print rows and columns of class upon initialisation (InCode) | Two Integers being printed in shell, representative of the rows and columns of the maze to be drawn | Two integers printed to shell | PASS | N.A |
| Maze Initialisation (Box width/height) | Ask class to print width and height of boxes of class upon initialisation (InCode) | Printing two integers to the shell, representative of the width and height of each box | Two integers printed to shell | PASS | N.A |
| Maze drawing | Call the draw maze function, to see if it draws maze correctly. (In draw maze function print an array of the boxes to shell, to see if boxes have been stored for collision) | Unrendered maze of the maze class should be drawn out onto the screen in blue boxes. Maze should fit the screen fully and it should be an accurate representation of the unrendered maze. In the shell a list of Rect Objects representative of wall placements should be printed. | Blue boxes were drawn but the last row of the maze was cut off at the bottom of the display. List of Rect objects was printed in shell. | **FAIL** | Incorrect number of rows input into the maze class, change rows to fit unrendered 2D array ‘width’ and ‘height’ |

*Maze class component testing iteration 2*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function/class to be tested** | **Input** | **Expected Outcome (Deliverable being tested)** | **Actual outcome** | **Pass/Fail** | **Further Action** |
| Maze initialisation  (Unrendered maze) | Ask class to print unrendered maze upon initialisation. (InCode) | A 2D array of 1s and 0s being printed in the shell, representative of the maze to be drawn | 2D array of 1s and 0s was printed to the shell | PASS | N.A |
| Maze initialisation  (Rows and columns) | Ask class to print rows and columns of class upon initialisation (InCode) | Two Integers being printed in shell, representative of the rows and columns of the maze to be drawn | Two integers printed to shell | PASS | N.A |
| Maze Initialisation (Box width/height) | Ask class to print width and height of boxes of class upon initialisation (InCode) | Printing two integers to the shell, representative of the width and height of each box | Two integers printed to shell | PASS | N.A |
| Maze drawing | Call the draw maze function, to see if it draws maze correctly. (In draw maze function print an array of the boxes to shell, to see if boxes have been stored for collision) | Unrendered maze of the maze class should be drawn out onto the screen in blue boxes. Maze should fit the screen fully and it should be an accurate representation of the unrendered maze. In the shell a list of Rect Objects representative of wall placements should be printed. | Blue boxes were drawn and a correct rendition of the unrendered 2D array was created. List of Rect objects was printed in shell. | PASS | N.A |

*Player component testing iteration 1*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function/class to be tested** | **Input** | **Expected Outcome (Deliverable being tested)** | **Actual outcome** | **Pass/Fail** | **Further Action** |
| Spawn | Create player instance and use spawn class function in code. | Red block (Player) should appear at x=0 y=0 | Red block appeared at x=0 y=0 | PASS | N.A |
| Movement (right) | Once player has spawned, press right arrow on keyboard | Red block should move one space to the right. | Recursion error: Max recursion depth reached | **FAIL** | Condition for recursion ending (if self.x == expected) is never met. This is due to movement speed of player not being a factor of the width of each block. Change (if self.x == expected: STOP recursion) to (if self.x > expected: STOP recursion) |
| Movement (Up) | Once player has spawned, press Up arrow on keyboard | Red block should move one space to the up. | Recursion error: Max recursion depth reached | **FAIL** | Problem is same as above. Change (if self.y == expected:  STOP recursion) to (if self.y < expected:  STOP recursion) |
| Movement (left) | Once player has spawned, press left arrow on keyboard | Red block should move one space to the left | Recursion error: Max recursion depth reached | **FAIL** | Problem is same as above. Change (if self.x == expected: STOP recursion) to (if self.x < expected:  STOP recursion) |
| Movement (Down) | Once player has spawned, press down arrow on keyboard | Red block should move one space down. | Recursion error: Max recursion depth reached | **FAIL** | Problem is same as above. Change (if self.y == expected:  STOP recursion) to (if self.y > expected:  STOP recursion) |

*Player component testing iteration 2*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function/class to be tested** | **Input** | **Expected Outcome (Deliverable being tested)** | **Actual outcome** | **Pass/Fail** | **Further Action** |
| Spawn | Create player instance and use spawn class function in code. | Red block (Player) should appear at x=0 y=0 | Red block appeared at x=0 y=0 | PASS | N.A |
| Movement (right) | Once player has spawned, press right arrow on keyboard | Red block should move one space to the right. | Red block does appear one space to the right but you cannot see the block moving. | **FAIL** | Program does not think to let users see how the red block moves from A to B, as I have not told it to slow down the process slightly. Use pygame’s pygame.time.clok and pygame.time.Clock.tick to give the red block a framerate at which to move. |
| Movement (Up) | Once player has spawned, press Up arrow on keyboard | Red block should move one space to the up. | Red block does appear one space upwards but you cannot see the block moving. | **FAIL** | ^ |
| Movement (left) | Once player has spawned, press left arrow on keyboard | Red block should move one space to the left | Red block does appear one space to the left but you cannot see the block moving. | **FAIL** | ^ |
| Movement (Down) | Once player has spawned, press down arrow on keyboard | Red block should move one space down. | Red block does appear one space downwards but you cannot see the block moving. | **FAIL** | ^ |

*Player component testing iteration 3*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Function/class to be tested** | **Input** | **Expected Outcome (Deliverable being tested)** | **Actual outcome** | **Pass/Fail** | **Further Action** |
| Spawn | Create player instance and use spawn class function in code. | Red block (Player) should appear at x=0 y=0 | Red block appeared at x=0 y=0 | PASS | N.A |
| Movement (right) | Once player has spawned, press right arrow on keyboard | Red block should move one space to the right. | Red block moves one space to the right | PASS | N.A |
| Movement (Up) | Once player has spawned, press Up arrow on keyboard | Red block should move one space to the up. | Red block moves one space upwards | PASS | N.A |
| Movement (left) | Once player has spawned, press left arrow on keyboard | Red block should move one space to the left | Red block moves one space to the left | PASS | N.A |
| Movement (Down) | Once player has spawned, press down arrow on keyboard | Red block should move one space down. | Red block moves one space downwards. | PASS | N.A |

*Collision detection component testing iteration 1*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Component to be tested** | **Input** | **Expected Outcome (Deliverable being tested)** | **Actual outcome** | **Pass/Fail** | **Further Action** |
| Collision detection (‘Wall’ blocks) | Attempt to move the red block (player block) into a blue block using the left, right, up or down movement keys. (See movement testing) | The red block should not move. | Red block does not move | PASS | N.A |
| Collision detection (‘Winning’ the maze) | Disable collision detection for walls temporarily and move block to the edge of the screen. (see movement testing and remember to turn collision detection back on afterwards) | Screen should clear and white text saying “You win!” should appear. | Screen clears, displaying white text saying “You win!” | PASS | N.A |

*Further player input validation testing iteration 1*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Expected Outcome** | **Actual outcome** | **Pass/Fail** | **Further Action** |
| Press Left Arrow Key once player has spawned | (See component testing above) | (See component testing above) | PASS | N.A |
| Press Right Arrow Key once player has spawned | (See component testing above) | (See component testing above) | PASS | N.A |
| Press Up Arrow Key once player has spawned | (See component testing above) | (See component testing above) | PASS | N.A |
| Press Down Arrow key once player has spawned | (See component testing above) | (See component testing above) | PASS | N.A |
| Press any three buttons on the keyboard other than Left Arrow, Right Arrow, Down Arrow, Up Arrow | Nothing should happen. Screen should remain the same. | Nothing happens. Chosen keys are ‘H’,’A’, and ‘1’. | PASS | N.A |
| Click any three places on the screen. | Nothing should happen. Screen should remain the same. | Nothing happens. | PASS | N.A |

*In Game menu component testing iteration 1*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Component to be tested** | **Input** | **Expected Outcome (Deliverable being tested)** | **Actual outcome** | **Pass/Fail** | **Further Action** |
| Open menu | Press ‘Esc’ on keyboard whilst an instance of ingame menu exists, whilst menu is not ‘open’ | A menu should open over the maze which has two buttons: ‘SAVE’ and ‘QUIT’ | A menu opens over the maze which has two options ‘Save’ and ‘Quit’ | PASS | N.A |
| Close Menu | Press ‘Esc’ on keyboard whilst an instance of ingame menu exists, whilst menu is ‘open’ | The menu should close and the maze should be revealed again. | The menu semi-closes. The menu border remains. | **FAIL** | As menu is effectively painted over with a black background, then the maze painted over that, only the filler was painted over but the border remains. Fill border with black and paint over maze/player. |
| Save Button | Click within the save button | Nothing should appear to happen on the screen but a file called ‘sav.txt’ should appear in the folder with the program. Nothing should be in this until the save/load features have been fully implemented | A sav.txt file appears in the folder with the program. | PASS | N.A |
| Quit Button | Click within quit button | Until all components have been integrated, button should clear screen to black. | Screen clears to black | PASS | N.A |

*In Game menu component testing iteration 2*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Component to be tested** | **Input** | **Expected Outcome (Deliverable being tested)** | **Actual outcome** | **Pass/Fail** | **Further Action** |
| Open menu | Press ‘Esc’ on keyboard whilst an instance of ingame menu exists, whilst menu is not ‘open’ | A menu should open over the maze which has two buttons: ‘SAVE’ and ‘QUIT’ | A menu opens over the maze which has two options ‘Save’ and ‘Quit’ | PASS | N.A |
| Close Menu | Press ‘Esc’ on keyboard whilst an instance of ingame menu exists, whilst menu is ‘open’ | The menu should close and the maze should be revealed again. | The menu semi-closes. The menu border remains. | PASS | N.A |
| Save Button | Click within the save button | Nothing should appear to happen on the screen but a file called ‘sav.txt’ should appear in the folder with the program. Nothing should be in this until the save/load features have been fully implemented | A sav.txt file appears in the folder with the program. | PASS | N.A |
| Quit Button | Click within quit button | Until all components have been integrated, button should clear screen to black. | Screen clears to black | PASS | N.A |

*In Game menu input validation testing iteration 1*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Expected Outcome** | **Actual outcome** | **Pass/Fail** | **Further Action** |
| Click within the borders of the menu when its open, but not within any of the buttons. | Nothing should happen, the screen should remain the same. | Nothing happened | PASS | N.A |
| Press Esc when menu is not open | (See component testing) | (See component testing) | PASS | N.A |
| Press Esc when menu is open | (See component testing) | (See component testing) | PASS | N.A |
| Click within borders of save button | (See component testing) | (See component testing) | PASS | N.A |
| Click within borders of quit button | (See component testing) | (See component testing) | PASS | N.A |
| Press Up arrow, Down Arrow, Left Arrow, Right Arrow on keyboard. | Nothing should happen, player should not move, screen should remain the same. | Nothing happened | PASS | N.A |
| Press anything on the keyboard that isn’t the above keys. | Nothing should happen, screen should remain the same. | Nothing happened | PASS | N.A |
| Whilst menu is not open, try clicking where the quit and save buttons were previously | Nothing should happen, screen should remain the same. | Nothing happened | PASS | N.A |

*Save/Load component testing iteration 1*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Component to be tested** | **Input** | **Expected Outcome (Deliverable being tested)** | **Actual outcome** | **Pass/Fail** | **Further Action** |
| Save feature | Call save function | Until integration is carried out, this should create a sav.txt file, which has PLACEHOLDER written in it. | Sav.txt file is created which has the word PLACEHOLDER written in it | PASS | N.A |
| Load feature | Call load function | Until integration is completed, this should print the contents of the sav.txt file to shell. | PLACEHOLDER is printed in shell | PASS | N.A |

***Integration testing - carried out on Windows 10 OS***

*Integration testing iteration 1*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Stage** | **Input** | **Expected Outcome** | **What is being tested** | **Actual outcome** | **Pass/Fail** | **Further Action** |
| Starting (1) | Start program | Whether program can start on its own | There should be a menu on screen that has the title ‘Maze’, and has three buttons; Start, Continue and Quit. | There is the described menu on the screen | PASS | N.A |
| Starting (2) | Press continue button on screen | Whether the error message displays properly when there is no save file | An error message should cover the continue button saying ‘No Save File Available’ | An error message appears covering the continue button saying ‘No Save File Available’ | PASS | N.A |
| Starting (3) | Press Start button on screen | Whether the main menu can lead onto the main game (Integration between two modules) | A maze loads and a player (Red Box) is spawned in the first space where there is no blue box. | There is a maze with a red box in the first space where there is no box | PASS | N.A |
| Starting (4)  (Input validation) | Press where Start button was previously on main menu. | Whether previous menu functions still take place. | Nothing should happen. Screen should remain the same | Screen seems to clear and maze reappears | **FAIL** | Button function is still there even though visuals of button is not. Break loop that is in main menu function when start or load function is called. |
| Starting (5) (Input validation) | Press where continue button | Whether previous menu functions still take place. | Nothing should happen. Screen should remain the same | The error message that occurs when there is no save file covers the area where there should be a continue button | **FAIL** | ^ |
| Starting (6) (Input validation) | Press where quit button was previously on main menu. | Whether previous menu functions still take place. | Nothing should happen. Screen should remain the same | Game uninitialized and quits | **FAIL** | ^ |

***Integration testing had to begin after last error had been fixed as program had stopped.***

*Integration testing iteration 2*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Stage** | **Input** | **Expected Outcome** | **What is being tested** | **Actual outcome** | **Pass/Fail** | **Further Action** |
| Starting (1) | Start program | Whether program can start on its own | There should be a menu on screen that has the title ‘Maze’, and has three buttons; Start, Continue and Quit. | There is the described menu on the screen | PASS | N.A |
| Starting (2) | Press continue button on screen | Whether the error message displays properly when there is no save file | An error message should cover the continue button saying ‘No Save File Available’ | An error message appears covering the continue button saying ‘No Save File Available’ | PASS | N.A |
| Starting (3) | Press Start button on screen | Whether the main menu can lead onto the main game (Integration between two modules) | A maze loads and a player (Red Box) is spawned in the first space where there is no blue box. | There is a maze with a red box in the first space where there is no box | PASS | N.A |
| Starting (4)  (Input validation) | Press where Start button was previously on main menu. | Whether previous menu functions still take place. | Nothing should happen. Screen should remain the same | Nothing happens | PASS | N.A |
| Starting (5) (Input validation) | Press where continue button | Whether previous menu functions still take place. | Nothing should happen. Screen should remain the same | Nothing happens | PASS | N.A |
| Starting (6) (Input validation) | Press where quit button was previously on main menu. | Whether previous menu functions still take place. | Nothing should happen. Screen should remain the same | Nothing happens | PASS | N.A |
| Starting (7) | Press Left Arrow Key | Whether player can move left | Red box should move one space to the left. | Upon further inspection, the red box does not move exactly one space to the left and instead moves just over one space to the left. This causes a lack of alignment. | **FAIL** | The ((if self.x < expected:  STOP recursion)  ) statement(s) are not good enough. The horizontal speed of the player must be a factor of the width of each box and the (self.x <expected:) part of the statement must be changed to self.x==expected) |
| Starting (8) | Press Down arrow key | Whether player can move down | Red box should move one space downwards. | Upon further inspection, the red box does not move exactly one space downwards and instead moves just over one space downwards. This causes a lack of alignment. | **FAIL** | Same problem as above. Change (if self.y > expected) to (if self==expected). Ensure the vertical speed of the player is a factor of 720/(number of rows) |
| Starting (9) | Press Up arrow key | Whether player can move up | Red box should move one space upwards. | Upon further inspection, the red box does not move exactly one space upwards and instead moves just over one space upwards. This causes a lack of alignment. | **FAIL** | Same problem as above. Change (if self.y < expected) to (if self==expected). Ensure the vertical speed of the player is a factor of 720/(number of rows) |
| Starting (10) | Press Right arrow key | Whether player can move right | Red box should move one space to the right | Upon further inspection, the red box does not move exactly one space to the right and instead moves just over one space to the right. This causes a lack of alignment. | **FAIL** | Same problem as above. Change (if self.x > expected) to (if self==expected). Ensure the vertical speed of the player is a factor of 720/(number of columns) |
| Starting (11) | Attempt to move the red box into a block where a blue box is using the up, right, left or down keys. | Whether collision detection works | Red box should remain in the same place. | N.A due to the previous few parts of integration test failing. | < | < |
| Save/Load (1) | Move player to centre of the screen. | Whether in-game menu can be opened | Menu should open in the centre of the screen with two buttons saying ‘SAVE’ or ‘QUIT’, covering the player | Menu opens and covers the player/maze, showing two buttons ‘SAVE’ and ‘QUIT’ | PASS | N.A |
| Save/Load (2) | Press ‘Esc’ | Whether in-game menu can be closed | Menu should disappear revealing the maze and player | Menu disappears revealing maze and player | PASS | N.A |
| Save/Load (3) | Move player to a place that is not the spawn block. Press ‘Esc’ and click the ‘Save’ button on in-game menu. | Whether data can be saved. | Upon save, nothing should appear to happen but a ‘sav.txt’ file should be created in the same folder in which the program is. This file should contain the ‘maze ID’, player x coordinate and player y coordinate. | Sav.txt file is created with the three integers representing maze ID, player x coordinate, and player y coordinate on individual lines. | PASS | N.A |
| Save/Load (4) | Press ‘quit’ button on in-game menu. | Whether pressing the quit button takes you back to the main menu (Integration between main game and main menu) | The screen should clear, and the main menu should appear again and work. | Main menu reappears, start button works, therefore it can be assumed that other buttons work. (Returned to main menu via. The same function used before) | PASS | N.A |
| Save/Load (5) | Press the ‘continue’ button | Whether pressing the continue button loads your save game correctly  (Integration between main menu, main game and load game) | The screen should clear and the maze that the user was in previously should generate and player appear in previous player position. | Black screen appears, not loading the previously saved player/maze data. | **FAIL** | When the data is copied into the array the invisible formatting from the file is copied as well. I.e ‘\n’ is still in the arrays, meaning the data garnered is unexpected input which is badly handled. (Write about this in evaluation/documentation). Format array to get rid of ‘\n’ in each element of data. |

**Integration testing failed and could not continue past the 5th part of Save/Load phase. See iteration 3 for error fixes.**

*Integration testing iteration 3*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Stage** | **Input** | **Expected Outcome** | **What is being tested** | **Actual outcome** | **Pass/Fail** | **Further Action** |
| Starting (1) | Start program | Whether program can start on its own | There should be a menu on screen that has the title ‘Maze’, and has three buttons; Start, Continue and Quit. | There is the described menu on the screen | PASS | N.A |
| Starting (2) | Press continue button on screen | Whether the error message displays properly when there is no save file | An error message should cover the continue button saying ‘No Save File Available’ | An error message appears covering the continue button saying ‘No Save File Available’ | PASS | N.A |
| Starting (3) | Press Start button on screen | Whether the main menu can lead onto the main game (Integration between two modules) | A maze loads and a player (Red Box) is spawned in the first space where there is no blue box. | There is a maze with a red box in the first space where there is no box | PASS | N.A |
| Starting (4)  (Input validation) | Press where Start button was previously on main menu. | Whether previous menu functions still take place. | Nothing should happen. Screen should remain the same | Nothing happens | PASS | N.A |
| Starting (5) (Input validation) | Press where continue button | Whether previous menu functions still take place. | Nothing should happen. Screen should remain the same | Nothing happens | PASS | N.A |
| Starting (6) (Input validation) | Press where quit button was previously on main menu. | Whether previous menu functions still take place. | Nothing should happen. Screen should remain the same | Nothing happens | PASS | N.A |
| Starting (7) | Press Left Arrow Key | Whether player can move left | Red box should move one space to the left. | Red box moves one space to the left. | PASS | N.A |
| Starting (8) | Press Down arrow key | Whether player can move down | Red box should move one space downwards. | Red box moves one space downwards. | PASS | N.A |
| Starting (9) | Press Up arrow key | Whether player can move up | Red box should move one space upwards. | Red box moves one space upwards. | PASS | N.A |
| Starting (10) | Press Right arrow key | Whether player can move right | Red box should move one space to the right | Red box moves one space to the right. | PASS | N.A |
| Starting (11) | Attempt to move the red box into a block where a blue box is using the up, right, left or down keys. | Whether collision detection works | Red box should remain in the same place. | N.A due to the previous few parts of integration test failing. | < | < |
| Save/Load (1) | Move player to centre of the screen. | Whether in-game menu can be opened | Menu should open in the centre of the screen with two buttons saying ‘SAVE’ or ‘QUIT’, covering the player | Menu opens and covers the player/maze, showing two buttons ‘SAVE’ and ‘QUIT’ | PASS | N.A |
| Save/Load (2) | Press ‘Esc’ | Whether in-game menu can be closed | Menu should disappear revealing the maze and player | Menu disappears revealing maze and player | PASS | N.A |
| Save/Load (3) | Move player to a place that is not the spawn block. Press ‘Esc’ and click the ‘Save’ button on in-game menu. | Whether data can be saved. | Upon save, nothing should appear to happen but a ‘sav.txt’ file should be created in the same folder in which the program is. This file should contain the ‘maze ID’, player x coordinate and player y coordinate. | Sav.txt file is created with the three integers representing maze ID, player x coordinate, and player y coordinate on individual lines. | PASS | N.A |
| Save/Load (4) | Press ‘quit’ button on in-game menu. | Whether pressing the quit button takes you back to the main menu (Integration between main game and main menu) | The screen should clear, and the main menu should appear again and work. | Main menu reappears, start button works, therefore it can be assumed that other buttons work. (Returned to main menu via. The same function used before) | PASS | N.A |
| Save/Load (5) | Press the ‘continue’ button | Whether pressing the continue button loads your save game correctly  (Integration between main menu, main game and load game) | The screen should clear and the maze that the user was in previously should generate and player appear in previous player position. | Previously seen maze and player position loads onto screen. | PASS | N.A |
| Winning (1) | Lead the player block to the maze’s exit | Whether the player can win | The player block should leave the screen and the maze should clear to a black screen and white text display saying ‘You win!’, wait a few seconds, clear the screen and head back to the main menu | Player block leaves screen and the mentioned ‘You win!’ message appears on a black background, clearing back to main menu after a few seconds | PASS | N.A |
| Winning (2) | Press the quit button | Whether the game can be quit. | The game should uninitialize and close. | Game uninitialized and closes. | PASS | N.A |

***Integration testing - carried out on Windows 7 OS, after previous integration design, implementation and testing has been carried out.***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Stage** | **Input** | **Expected Outcome** | **What is being tested** | **Actual outcome** | **Pass/Fail** | **Further Action** |
| Starting (1) | Start program | Whether program can start on its own | There should be a menu on screen that has the title ‘Maze’, and has three buttons; Start, Continue and Quit. | There is the described menu on the screen | PASS | N.A |
| Starting (2) | Press continue button on screen | Whether the error message displays properly when there is no save file | An error message should cover the continue button saying ‘No Save File Available’ | An error message appears covering the continue button saying ‘No Save File Available’ | PASS | N.A |
| Starting (3) | Press Start button on screen | Whether the main menu can lead onto the main game (Integration between two modules) | A maze loads and a player (Red Box) is spawned in the first space where there is no blue box. | There is a maze with a red box in the first space where there is no box | PASS | N.A |
| Starting (4)  (Input validation) | Press where Start button was previously on main menu. | Whether previous menu functions still take place. | Nothing should happen. Screen should remain the same | Nothing happens | PASS | N.A |
| Starting (5) (Input validation) | Press where continue button | Whether previous menu functions still take place. | Nothing should happen. Screen should remain the same | Nothing happens | PASS | N.A |
| Starting (6) (Input validation) | Press where quit button was previously on main menu. | Whether previous menu functions still take place. | Nothing should happen. Screen should remain the same | Nothing happens | PASS | N.A |
| Starting (7) | Press Left Arrow Key | Whether player can move left | Red box should move one space to the left. | Red box moves one space to the left. | PASS | N.A |
| Starting (8) | Press Down arrow key | Whether player can move down | Red box should move one space downwards. | Red box moves one space downwards. | PASS | N.A |
| Starting (9) | Press Up arrow key | Whether player can move up | Red box should move one space upwards. | Red box moves one space upwards. | PASS | N.A |
| Starting (10) | Press Right arrow key | Whether player can move right | Red box should move one space to the right | Red box moves one space to the right. | PASS | N.A |
| Starting (11) | Attempt to move the red box into a block where a blue box is using the up, right, left or down keys. | Whether collision detection works | Red box should remain in the same place. | N.A due to the previous few parts of integration test failing. | < | < |
| Save/Load (1) | Move player to centre of the screen. | Whether in-game menu can be opened | Menu should open in the centre of the screen with two buttons saying ‘SAVE’ or ‘QUIT’, covering the player | Menu opens and covers the player/maze, showing two buttons ‘SAVE’ and ‘QUIT’ | PASS | N.A |
| Save/Load (2) | Press ‘Esc’ | Whether in-game menu can be closed | Menu should disappear revealing the maze and player | Menu disappears revealing maze and player | PASS | N.A |
| Save/Load (3) | Move player to a place that is not the spawn block. Press ‘Esc’ and click the ‘Save’ button on in-game menu. | Whether data can be saved. | Upon save, nothing should appear to happen but a ‘sav.txt’ file should be created in the same folder in which the program is. This file should contain the ‘maze ID’, player x coordinate and player y coordinate. | Sav.txt file is created with the three integers representing maze ID, player x coordinate, and player y coordinate on individual lines. | PASS | N.A |
| Save/Load (4) | Press ‘quit’ button on in-game menu. | Whether pressing the quit button takes you back to the main menu (Integration between main game and main menu) | The screen should clear, and the main menu should appear again and work. | Main menu reappears, start button works, therefore it can be assumed that other buttons work. (Returned to main menu via. The same function used before) | PASS | N.A |
| Save/Load (5) | Press the ‘continue’ button | Whether pressing the continue button loads your save game correctly  (Integration between main menu, main game and load game) | The screen should clear and the maze that the user was in previously should generate and player appear in previous player position. | Previously seen maze and player position loads onto screen. | PASS | N.A |
| Winning (1) | Lead the player block to the maze’s exit | Whether the player can win | The player block should leave the screen and the maze should clear to a black screen and white text display saying ‘You win!’, wait a few seconds, clear the screen and head back to the main menu | Player block leaves screen and the mentioned ‘You win!’ message appears on a black background, clearing back to main menu after a few seconds | PASS | N.A |
| Winning (2) | Press the quit button | Whether the game can be quit. | The game should uninitialize and close. | Game uninitialized and closes. | PASS | N.A |

*End User Feedback - Completed after all functional requirements/SQA requirements have been tested and is more evaluative than anything. It will most likely be taken into account upon further development.*

*User 1*

Q1:Do the buttons aid your use of the program? How so?

*A1: “The buttons are fairly helpful in showing where each button takes you, due to the labelling of each button. It would be helpful if there were more options for the game, though.”*

Q2:Do you have any suggested improvements for the UI?

*A2: “I’d like it if the ‘no save available’ error message could be closed. I accidentally opened it, but it was mildly irritating that I couldn’t close it again. Also, if the cursor changed when hovering over buttons, it would be better. That was a little confusing.”*

Q3:Would there be any customisations you would want to be implemented in the future? (e.g. changing screen size, audio volume etc)

*A3: “Yeah, changing screen size would be nice. My laptop is a bit too small for that resolution.”*

Q4:Are there any features you feel could be added for those that are impaired?

*A4: “I’m not sure.”*

Q5:Would you say the interface is consistent?

*A5: “Yeah. I like the colour scheme.”*

Q6:Do you feel as though you had the ability to go back on your decisions throughout the game?

*A6: “Like I said earlier, I’d like it if the ‘no save file available’ error message could be closed. Other than that, I can go back on my decisions throughout.”*

Q7:Where do you think the game could have helped you more?

*A7: “There wasn’t a tutorial for controls. I didn’t like having to go into txt documents for the game.”*

Q8: In your opinion, was this game intuitive? (i.e, easy to understand and use without much help)

*A8: “It was pretty easy to understand on the most part, but like I said a tutorial would be good.”*  
Q9: Is there any further developments you would like to see?

*A9: “More levels which increase in difficulty. Maybe a ‘randomized’ mode, where you complete a series of randomized mazes.”*

*User 2*

Q1:Do the buttons aid your use of the program? How so?

*A1: “The buttons placement and labelling is really helpful”*

Q2:Do you have any suggested improvements for the UI?

*A2: “It would be nice if there was a background image for the menu.”*

Q3:Would there be any customisations you would want to be implemented in the future? (e.g. changing screen size, audio volume etc)

*A3: “Have you tried implementing a full screen feature?”*

Q4:Are there any features you feel could be added for those that are impaired?

*A4: “I don’t know.”*

Q5:Would you say the interface is consistent?

*A5: “Yes.”*

Q6:Do you feel as though you had the ability to go back on your decisions throughout the game?

*A6: “Yeah.”*

Q7:Where do you think the game could have helped you more?

*A7:* *“I think everything was fairly laid out for me.”*

Q8: In your opinion, was this game intuitive? (i.e, easy to understand and use without much help)

*A8: “Yeah, it was easy to navigate.”*

Q9: Is there any further developments you would like to see?

*A9: “More levels with greater difficulty. Maybe better graphics.”*

*User 3*

Q1:Do the buttons aid your use of the program? How so?

A1: “The buttons’ placement looks nice and the colouring helps me see the buttons better.”

Q2:Do you have any suggested improvements for the UI?

*A2: “Although this isn’t really user interface, I’d like for there to be a soundtrack in the game.”*

Q3:Would there be any customisations you would want to be implemented in the future? (e.g. changing screen size, audio volume etc)

*A3: “If I could change the controls. I’m more comfortable with WASD than arrow keys.”*

Q4:Are there any features you feel could be added for those that are impaired?

*A4: “Maybe some higher contrast on the in-game menu.”*

Q5:Would you say the interface is consistent?

*A5: “The colour of the player block is a little jarring but yeah, I like it.”*

Q6:Do you feel as though you had the ability to go back on your decisions throughout the game?

A6: *“Yeah.”*

Q7:Where do you think the game could have helped you more?

*A7: “Even though the controls were fairly obvious, it’s bad form to not include a tutorial.”*

Q8: In your opinion, was this game intuitive? (i.e, easy to understand and use without much help)

*A8: “It was very intuitive and not too hard to understand. Other than the tutorial, I would make very little change to that aspect.”*

Q9: Is there any further developments you would like to see?

*A9: “A soundtrack and higher difficulty would be good.”*