**Testing**

Test Strategy

I will be using both black box testing and white box testing for my game. Black box testing will be

used to test the functionality of my application without looking at the structure of my code. White box testing will be used to not only test if each function gives the desired output, but also to test the efficiency of my code. I will also be using different data types to test it does not crash and the system handles everything correctly.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test #** | **Description** | **How to test/test data used** | **Expected Outcome** | **Actual Outcome** | **Times tamp** |
| 1 | Test to see if “Single-Player” button loads the single-player version of the game. Should also test to see if the button turns into a darker shade of its colour if it is hovered over. This shows mouse detection. | Hover over the button, then press the “Singleplayer” button. | The button turns into a darker shade of its colour then the Single Player game loads in and is displayed onto the screen. | Success: The button turned into a darker shade of its colour then the  Single Player game loads and is displayed onto the screen. | 0:02 |
| 2 | Test to see if “Local Multiplayer” button loads the co-op multiplayer version of the game. | Hover over the button, then press the “Local Multiplayer” button. | The Local Multiplayer game loads in and is displayed onto the screen. | Success: The  Local Multiplayer game loads and is displayed onto the screen. | 7:32 |
| 3 | Test to see if “Settings” button loads the settings part of the game | Hover over the button, then press the “Settings” button. | The Settings load in and are displayed onto the screen. | Success: The Settings load in and are displayed onto the screen. | 10:47 |
| 4 | Check if music is being played in the main menu | Being in the “Menu” game state and making sure that music is turned on in the settings (Music being on is the initial setting). | Music should be playing in the main menu. | Success: Music is playing in the main menu. | 0:01 |
| 5 | Test to see if pressing the Escape button exits out of the whole program or exits out the current gamemode the user(s) is playing. | Pressing the escape key will do this. | The program/gamemode terminates. | Success: The program/gamemode terminates. | 7:29 |
| 6 | Test to see if the “Quit” button exits out of the whole program. | Hover over the button, then press the “Quit” button. | The program/game terminates. | Success: The program/game terminates. | 15:00 |
| 7 | Test to see if the game resets after the user exits the game (single player or co-op multiplayer), and enters back into the game. | The user must exit their current game by either losing all three lives, pressing the escape key, or get to a score of 500 if they are playing co-op multiplayer. Then the user must re-enter and start another game. | The game will reset back to its starting initial state. | Success: The game reset back to its starting initial state. | 5:45 |
| 8 | Test to see if the “GAME OVER” and “\*player\* WINS”, is animated and appears when the game ends | End the game on either the gamemode, by either dying three times, or getting to a score of 500 if the second gamemode is being played. | The program should show an animated “GAME OVER” of gamemode one, or “\*player\* WINS” in gamemode two. | Success: The program shows an animated “GAME OVER” of gamemode one, or “\*player\* WINS” in gamemode two. | 5:39  10:40 |
|  | **SINGLE-PLAYER GAME** |  |  |  |  |
| 9 | Check if music is being played in single player. | Being in the “Single” game state and making sure that music is turned on in the settings (Music being on is the initial setting). | Music should be playing in the single player gamemode. This includes eating the dots, dying to an enemy, killing an enemy and the continuous Pacman music throughout the whole game. | Success: Music is played throughout the single player game mode in all aspects of the game. | 0:04 |
| 10 | Test if movement works using arrow keys works. This checks for player collision with the board. | Press UP, DOWN, LEFT and RIGHT arrow keys | Player is able to move if there is free space. (Not able to move through walls). | Success:  Players is able to move wherever there is free space. | 0:04 |
| 11 | Check if the player obtains dots (items that increase score) | Pacman is moved to a space where there is a yellow dot, using the arrow keys | Pacman collects a dot to increase score | Success: Pacman collects a dot which increases the players score. | 0:04 |
| 12 | Check if the score is displayed and gets updated on the left side of the screen | Move Pacman over to a yellow dot. | When Pacman passes over a dot the score should update on the left side of the screen. | Success: The score was displayed and updated on the left side of the screen. | 0:05 |
| 13 | Check if Inky (ghost/enemy) is taking the shortest path to get to Pacman, by using Dijkstra’s Algorithm. | Go to multiple specific parts of the Maze where there are multiple routes to get to that point. | The ghost Inky to move towards Pacman using the shortest path possible. | Success: The ghost Inky moves towards Pacman using the shortest path possible. | 0:30 |
| 14 | Check if Pinky (ghost/enemy) is able to locate Pacman using the Breadth-First Search Algorithm. | Constantly move Pacman and check if Pinky follows Pacman. | Pinky will move towards Pacman using the BFS Algorithm. | Success: Pinky moves towards Pacman using the BFS Algorithm | 0:50 |
| 15 | Check if Clyde (ghost/enemy) moves intersection to intersection and does not follow Pacman | Once Clyde spawns, move Pacman away from Clyde and check to see if he follows Pacman. In addition, check to see if he only changes direction at intersections. | Clyde to move intersection to intersection. Also does not search for Pacman. | Success: Clyde moves intersection to intersection. Also, he does not search for Pacman. | 1:20 |
| 16 | Check to see if Blinky (ghost/enemy) moves randomly and does not search for Pacman. | Once Blinky spawns, move Pacman away from Blinky to see if he follows Pacman. Random values will be used for Blinky to move randomly. | Blinky to move in random directions and to not search for Pacman. | Success: Blinky moved in random directions and did not search for Pacman. | 1:27 |
| 17 | Check if Inky, Pinky and Blinky use the line of sight algorithm to detect Pacman is in their sights. To also make sure that those three ghosts cannot see Pacman through a wall. | Move Pacman to the same x or y axis as the three ghosts individually to whom the line of sight algorithm applies. To check this works Pacman has to be on the same x or y axis as the ghost(s). Furthermore, to make sure the ghosts cannot see Pacman through a wall, the user should also go on the same x or y axis as the ghost but have a wall between them, which will test if Pacman can be seen through a wall. | Inky, Pinky and Blinky can see Pacman in their sight using the line of sight algorithm, provided Pacman is on the same x or y axis and there is no wall between them. | Success: Inky, Pinky and Blinky can see Pacman in their sight using the line of sight algorithm. Pacman can be seen when he is on the same x or y axis as those three ghosts as long as there is no wall. Therefore, this test is successful. | 1:50  2:00  4:10 |
| 18 | Test to see if Pacman collects the laser power up. | Move Pacman to the red dot on the maze. Then move Pacman away from that cell. | The dot disappears once Pacman moves over the cell where the red (laser) power up is. | Success: Pacman collects the laser power up after moving over where it spawned. | 2:05 |
| 19 | Test to see if the screen display updates to “laser” when Pacman collects the power up. | Move Pacman over a power up if Pacman is currently not holding one. | The text on the right side of the screen that displays “empty” will change to “laser”. | Success: The text changes to “laser” | 2:05 |
| 20 | Test to see if Pacman changes to red when the power up is activated, then back to yellow after the user’s activation of the power up finishes. | Move Pacman to the location of the power up to collect it. The user should activate it using the key ‘f’ or ‘F’. | Pacman should change to the colour red, then back to yellow. | Success: Pacman changes to the colour red, then back to yellow. | 2:08 |
| 21 | Test to see if the laser power up can be activated and if it lasts for a specific period of time (200 game loops) | Get Pacman to move towards a power up location then while carrying a power up (laser power up in this case), the user can activate it using the key ‘f’ or ‘F’. In addition, the user must wait some time to see if the power up eventually runs out. | The laser beam will appear on the maze after the user presses the key ‘f’ or ‘F’ and will run out after a specific period of time. | Success: The laser beam appears on the maze after the user presses the key ‘f’ or ‘F’ and it runs out after a specific period of time. | 2:37 |
| 22 | Test to see if the power up kills the ghost(s), and the ghost is sent back to the spawn area. | The user moves Pacman to be in the line of sight of a ghost which means the laser beam will collide with the ghost. Once the ghost dies, check to see if the ghost is now in the spawn area. | The laser beam should kill the Ghost and the ghost should be sent back to spawn. | Success: The laser beam killed the ghost and changed the ghost’s location to spawn. | 2:10 |
| 23 | Test to see if the laser beam does not go through walls. This checks for the collision detection. | The user constantly moves Pacman in all directions to check if the laser beam collides with the walls. | The laser beams starting position is Pacman and the end position is wherever the first wall is. | Success: The laser beams starting position is Pacman and the end position is where the beam collides with the first wall. | 2:13 |
| 24 | Test to see if the user can pick up another power up if Pacman is carrying the laser powerup | Once the user has picked up the laser power up and activated it, they should move Pacman to the “invisibility” power up to check if the laser power up gets removed/replaced. | The user should not be able to collect the “invisibility” power up as they are carrying the “laser” power up. | Success: The user is not able to collect the “invisibility” power up as they are carrying another power up. | 6:01 |
| 25 | Once the laser power up runs out, test to see if the invisibility power up is able to be collected. | Move Pacman towards the spawn location of the invisibility power up. Make sure Pacman is not currently holding a power up. | The invisibility power up should be removed from the board. | Success: The invisibility power up gets removed from the board when Pacman moves over it while not carrying a power up. | 3:05 |
| 26 | Check if the power up display on the right of the screen is updated from “empty” to “invisibility”. | Once Pacman has collected the power up, the user must check the right of their screen. | The power up displayed on the right side of the screen should display “invisibility”. | Success: The power up display says “invisibility” once the invisibility power up is collected. | 3:05 |
| 27 | Check if the invisibility power up works by activating it using the key ‘f’ or ‘F’. By activating the power up no enemies should be searching for Pacman for a specific period of time. | The user should press the key ‘f’ or ‘F’ once they have collected the power up and then move away from all the enemies to see if they follow Pacman. | Pacman should turn red for a specific period of time, which shows the power up has activated and is only available for a period of time. In addition, the ghosts should end up moving randomly and not be following Pacman. | Success: Pacman turns red for a period of time and all the ghosts end up moving randomly and do not search for Pacman, until the power up runs out. | 3:05 |
| 28 | Check the algorithm that allows for a dynamic adjacency matrix for the ghost Inky that uses the shortest-path algorithm, Dijkstra. The algorithm changes the adjacency matrix’s weights depending on the location of the other three ghosts. Furthermore, the matrix uses intersections as nodes and stores the vertices between them. To parse these vertices a complex method had to be used, which included figuring out how to implement XOR alongside other things. | Move Pacman to a part of the map where a ghost other than Inky is following you. Once this happens, make sure Inky is behind the first ghost. As this happens Inky will end up taking another route. | Inky will take another path to the ghost in front of him, to try trap Pacman and kill him. | Success: Inky uses another path than the ghost in front of him to try to trap Pacman rather than just following another ghost (which makes Inky useless). | 5:35 |
| 29 | Check that the player collision with the ghost’s work. | Pacman must move into the same space as an enemy and die, and therefore lose a life. | Pacman is expected to die, lose a life and then respawn and the starting location. | Success: Pacman dies when colliding with an enemy. He then loses a life and respawns in the starting location | 5:25 |
| 30 | Check the player lives indicator is updated on the screen when the player dies. | Let Pacman collide with the enemy, provided he is not immune, to lose a player life. | The integer displayed on the left side of the screen that states the players lives should decrease by one when the player dies. | Success: The players lives displayed decreases by one when the player (Pacman) dies. | 5:25 |
| 31 | Test to see if the player is immune to dying for a specific period of time. Check to also see if Pacman loses his power up when he dies. | Pacman is moved to a spot where he can collect a power up, then he should be holding this power up before he dies and should immediately try colliding into an enemy after dying. Then after a specific amount of time, the user should try dying again. | Pacman should be immune to the enemy the first time he collides with the enemy. The second time (when the specific immune time period is finished), Pacman should lose another life. Pacman should lose his power up (It should say “empty”). | Success: Pacman is immune the first time he collides with the enemy, and then the second time he dies. In addition, Pacman loses his power up. | 3:20  6:28 |
| 32 | Test to see if the cost function works. The cost function is used to speed up the game every 100 points, by using an inverse logarithmic function. | Get Pacman to keep collecting dots until he reaches a multiple of 100. To show the cost function obviously works, get the player to get a score of 200 rather than 100. This will increase the cost function twice, making it obvious in the game. | The ghosts should increase in speed as Pacman reaches a score which is a multiple of 100. | Success: The ghosts are visible much faster than when they started, after reaching a score of 100 and 200, etc... | 2:43 |
| 33 | Test to see if all the dots and power ups respawn once Pacman collects all items available on the board. | Pacman must collect all items available on the board/Maze (power ups and dots). | All dots and power ups should respawn on the board/Maze. | Success: All dots and power ups respawn on the board/Maze. | 5:10 |
| 34 | Test to see if the enemy spawn timer works every time the enemy dies from a laser beam. | Get Pacman to pick up the laser power up and kill a ghost. | Each ghost has a designated time in which it takes for them to spawn in, so depending on what ghost dies (in this case it is Inky) the spawn timer would be different. In this case Inky will spawn in the same amount of time as it took him to spawn in the first time (when the game started). | Partial Success: The ghosts spawn in using a spawn timer every time after they die which is what I wanted, however after the first time they spawn in (start of the game), their spawn timers become random, which is not what I intended. | 2:08 |
| 35 | Test to see if enemies move away from each other when they collide outside of spawn. (Enemy Detection Algorithm) | To test this enemy detection algorithm, enemies need to be within one vector distance away from each other. | The enemies move in opposite directions if they are within one vector distance away from each other. This is to prevent enemies overlapping. | Success: The ghosts/enemies do move in opposite directions increasing the distance between them if they are within one vector distance away from each other. | 3:38  7:10 |
| 36 | Test to see if pausing the game works. | The user must press the ‘p’ or ‘P’ key to pause the game. | The game loop should pause (the game becomes paused) and the text “Paused” should be displayed onto the centre of the screen. | Success: The game loop is paused (the game becomes paused) and the text “Paused” is displayed onto the centre of the screen. | 6:51 |
| 37 | Test to see if the game ends and returns to the “menu” state once the player loses all three lives. | This requires the player to die three times and lose all their lives. | The game should end and “GAME OVER” should appear on the screen. Then the program should return to the main menu. | Success: The game ends and “GAME OVER” appears on the screen. Then the program returns to the main menu. | 5:40 |
|  | **CO-OP (LOCAL)**  **MULTIPLAYER** |  |  |  |  |
| 38 | Test if the players spawn in the opposite corners of the board. | This can be tested as soon as the player is loaded in | Players should spawn on opposite sides of the board as either a red or blue colour. | Success: Players are spawned on opposite sides of the board as either a red or blue colour. | 7:32 |
| 39 | Check if music is being played in the co-op multiplayer game. | Being in the “Multi” game state and making sure that music is turned on in the settings (Music being on is the initial setting). | Music should be playing in the co-op local multiplayer game. | Success: Music is playing in the co-op local multiplayer game. | 7:32 |
| 40 | Test for Player Two’s movement and collision detection (Player One was already checked in Single Player, using the arrow keys). | Use the WASD keys (UP, LEFT, DOWN and RIGHT) to move Player Two. | Player two should only be moving in the free cells (available spaces), and the collision detection should stop player two from moving into a wall. | Success: Player two only moves into free cells, and not walls, due to collision detection. | 7:34 |
| 41 | Test to see if the players’ score is updated on the correct side of the screen | Both players move after one another to see if the score updates on the correct side of the screen. Left side for Player One and right side for Player Two. | When player one moves, the score on the left side of the screen should update (increase). When player two moves, the score on the right side of the screen should update (increase). | Success: When player one moves, the score on the left side of the screen updates (increases). When player two moves, the score on the right side of the screen updates (increases). | 7:34 |
| 42 | Check if the lava spawns and is able to kill a player if they step into it. | Move a player into lava to see if the player dies a respawns in a new random location that is not occupied by lava. | When the player collides with the lava, they should lose a life and respawn in a location which is not occupied by lava. | Success: When the player collides with the lava, they lose a life and respawn in a location which is not occupied by lava. | 7:41 |
| 43 | Check if the players’ lives are updated on the correct side of the screen. | Once the player moves into a lava tile, their lives will decrease by one. | When player one moves into a lava tile, the lives on the left side of the screen should update (decrease). When player two moves into a lava tile, the lives on the right side of the screen should update (decrease). | Success: When player one moves into a lava tile, the lives on the left side of the screen updates (decreases). When player two moves into a lava tile, the lives on the right side of the screen updates (decreases). | 7:41 |
| 44 | Once a player dies, I will test to see if the player is immune from a lava tile for a short period of time. | Move the player through lava tiles to test they won’t die. Then after a period of time test again and they should die. | The player should be immune from lava for a short period of time after they die. | Success: The player is immune from lava for a short period of time after they die. | 7:55 |
| 45 | Test to see if the lava continues to spawn in random locations which are not occupied by a player, until there are thirty lava tiles. | Do not move any of the players until there are thirty lava tiles. At this point no more lava tiles should spawn. | Thirty lava tiles should spawn in random locations that are not occupied by any players. | Success: Thirty lava tiles (and not a single one more) spawns in random locations that are not occupied by any players | 8:31 |
| 46 | Test to see if lava tiles turn into rock tiles in a random order. This should happen periodically until all lava tiles are turned into rock tiles. | Wait for a period of time to check if all the lava tiles turn to rock tiles, in random order. | All lava tiles should eventually turn into rock tiles. Each lava tile should turn into a rock tile in the same given amount of time. | Success: All lava tiles eventually turn into rock tiles. Each lava tile turns into a rock tile in the same given amount of time. | 8:34  9:05 |
| 47 | Test to see if the user can now walk over the rock tiles | Move either player over a rock tile (it is the grey colour tile). | The user should be able to move over the rock tile, without dying. | Success: The user is be able to move over the rock tile, without dying. | 9:45 |
| 48 | Test to see if the game ends and returns to the “menu” state once a player loses all three lives. | This requires a player to die three times and lose all their lives. | The game should end and “PLAYER 1 WINS”, or, “PLAYER 2 WINS” should appear on the screen. Then the program should return to the main menu. | Success: The game ends and “PLAYER 1 WINS”, or, “PLAYER 2 WINS” appears on the screen. Then the program returns to the main menu. | 10:41 |
| 49 | Test to see if the game ends and returns to the “menu” state once a player gets to a score of 500 | This requires a player to gain enough dots to get a score of 500. | The game should end and “PLAYER 1 WINS”, or, “PLAYER 2 WINS” should appear on the screen. Then the program should return to the main menu. | Success: The game ends and “PLAYER 1 WINS”, or, “PLAYER 2 WINS” appears on the screen. Then the program returns to the main menu. | 9:55 |
| 50 | Check to see if the player will lose 100 points off their score if they die with a score >= 100 | The player has to collide with a lava tile and die, with a score >= 100. | The player should get deducted 100 off their score. | Success: The player is deducted 100 off their score. | 10:22 |
| 51 | Check if the player loses all current points off their score if they die with a score <= 100 | The player has to collide with a lava tile and die, with a score <= 100. | The player should get deducted all their points off their score. | Success: The player is deducted all their points off their score. | 10:33 |
| 52 | Check to see all dots are removed underneath the lava/rock tiles. | The player can check this by moving underneath a lava tile while being immune. If no “Pacman eating music” takes place, there is no dot underneath. Otherwise the user can also test this by collecting all dots visible and see if the dots respawn across the maze/board. | The dots should be removed from underneath the lava/rock tiles. | Success: The dots are removed from underneath the lava/rock tiles. | 7:56 |
|  | **SETTINGS** |  |  |  |  |
| 53 | Test to see if Music is working while the game is in “Settings” state. | Enter the Settings part of the program. | Music should be playing. | Success: Music is playing. | 10:48 |
| 54 | Check if the “Help” button displays text onto the screen. | Hover the mouse over the “Help” button, then press it. | Text should be displayed, line by line, onto the screen. | Success: Text is displayed, line by line, onto the screen. | 10:52 |
| 55 | Check if the “Music” button can turn music on and off | Hover the mouse over the “Music” button, then press it. Then proceed to leave the Settings and go to every part of the game to check if Music has turned off. | Music will have been stopped throughout the game. | Success: Music was stopped throughout the game. | 10:58 |
| 56 | Check if the “Highscore” button works. | To test this, the player must beat their previous score by at least one, to check that the program read the highscore file and updated it to the new highscore. | The new highscore is displayed onto the screen. | Success: The new highscore gets displayed | 11:40  12:10 |
| 57 | Check if the “Difficulty” button works. It should display the difficulty chosen and also change the cost function in the single-player game. | To test this, change to “Medium” difficulty, then go to the single-player game mode and test to see if the ghost(s) are faster than before. After change to “Hard” difficulty and repeat. | The ghost(s) should be faster on “Medium” than easy, and they should be faster on “Hard” than “Medium”. | Success: The ghost(s) are faster on “Medium” than easy, and they are faster on “Hard” than “Medium”. | 12:12 till 14:53 |
| 58 | Check if the “Exit to Main Menu” button returns the user back to the menu game state. | To test this, the user must hover over the “Exit to Main Menu” button and press it. | The user should be returned to the main menu. | Success: The user is returned to the main menu. | 14:55 |

I used OBS studio to record my testing video

<https://youtu.be/dQR2URc0VJ4>