**APPENDIX A: SPRINT CYCLE 3 - Player Movement**

|  |  |
| --- | --- |
| 1. **Summary data** | |
| Team number |  |
| Sprint technical lead(s) | Ethan, Sarah |
| Sprint start date | 7/4/21 |
| Sprint end date | 14/4/21 |

|  |  |
| --- | --- |
| 1. **Individual key contributions** | |
| **Team member** | **Key contribution(s)** |
| Ethan | Programmer - game logic and mechanics |
| Sarah | Programmer - GUI |
| Umar | Sprint documentation |
| Will | Javadoc testing |
| Adam | Javadoc testing |
| Sam | Javadoc testing |

|  |
| --- |
| 1. **User stories / task cards** |
| * The game is played by moving around a playing board that represents the ground floor of Archers Avenue * The playing pieces are placed on the starting squares marked for them on the board * Players may move their pieces anywhere on the board on the squares according to the number thrown on the dice. Players must move in straight lines only i.e. forwards and crosswise, but never diagonally. * Players may enter rooms by their doors only, but cannot enter and leave a room in the same move. Entering a room ends the move count. If a player throws a 6 and only needs 4 to enter a room, they may enter the room and ignore the 2 moves left over. * No two pieces may occupy the same square. A room however may be occupied by any number of pieces and weapons. A player may choose to remain stationary during any number of turns in play. |

|  |
| --- |
| 1. **Requirements analysis** |
| F3 - Each player will control one of the 6 playing pieces which shall be able to walk across the board in increments according to the number received from the same player's dice roll. no diagonal movements allowed.  F6 - All players shall start the game located on one of the assigned starting locations on the board  F11 - The player shall be able to roll dice on their turn in order to determine the distance their character can travel on the board. There shall be two 6 sided dice for the player to ‘roll’ together  F15 - The player shall be able to move his/her character across the board via the keyboard  F16 - There could be a pair of dice with an animation when rolled, this would happen when the dice roll is triggered.  F22 - Players shall only be allowed to move in straight lines only i.e. forwards and crosswise, but never diagonally.  F23 - Players may enter rooms by their doors only, but cannot enter and leave a room in the same move. Entering a room ends the move count.  F24 - No two pieces may occupy the same square. A room however may be occupied by any number of pieces and weapons.  F25 - A player may choose to remain stationary during any number of turns in play.  NF1 - The software shall be written in Python to ensure maximum portability across platforms  NF4 - The software design shall accommodate future updates or maintenance through well designed code  D3 - The characters must be aesthetically pleasing and not be obviously offending to cultural group |

|  |
| --- |
| 1. **Design** |
| The player shall be able to move around the board in increments according to the players dice roll. The player should start on the prescribed starting tiles and no 2 players may occupy the same space.  The player shall consist of a token that physically moves around the board when instructed. The following image shows a UML diagram of the player and the player movement. The movement will work through the player token. The player token class inherits from the Token class which controls all objects on the board game.    The following image shows the use case diagram that includes the procedure for player movement. The player is required to roll a dice before they can move across the map. There is a separate process of events when a player enters a room as this causes the movement count to end. There is a loop between dice roll and movement as any player must roll before moving.  The GUI for the player movement will be fairly simple, there won't be any animation when the player is moving so the design shall be simple. The next image shows the sequence diagram from moving the player token.    To implement the player movement, we decided to use a single tile movement system controlled by the arrow keys. This involves more user input and we decided that it mirrors the physical board game better than the user being able to select any tile within range using the mouse. We also decided that this implementation would mitigate any bugs or errors in the player movement. The use of the arrow keys eliminates the issue of the player moving diagonally, which is not allowed. Therefore this simple design function makes the program more efficient as it requires less validation code to be applied. |

|  |
| --- |
| 1. **Test plan and evidence of testing** |
| from src.dice import Dice  import unittest  class MyTestCase(unittest.TestCase):  def test\_roll(self):  dice = Dice()  for i in range(100000):  roll1, roll2 = dice.roll()  roll\_total = roll1 + roll2  self.assertEqual(roll\_total <=12 and roll\_total >= 2, True)  if \_\_name\_\_ == '\_\_main\_\_':  unittest.main()  *import unittest*  *from src.playertoken import PlayerToken*  *from src.card import Card*  *from src.board import Board*  *from src.player import Player*  *class MyTestCase(unittest.TestCase):*  *# Unfinished test*  *def test\_move\_by\_direction(self):*  *board = Board()*  *card = Card('', 0, '')*  *player = Player('', '', '')*  *playerToken = PlayerToken(10, 10, card, board, player)*  *playerToken.move\_by\_direction(1, 1)*  *if \_\_name\_\_ == '\_\_main\_\_':*  *unittest.main()* |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **System Testing** | | | | | | | |
| **ID** | **Req** | **Description** | **Inputs** | **Expected** | **Actual** | **Pass/Fail** | **Action** |
| 1 | **F3** | Test that the player model moves left 1 tile when the a key is pressed | ‘a’ key | The player model moves left one tile | The player model moves left one tile | Pass | n/a |
| 2 | **F3** | Test that the player model moves up 1 tile when the w key is pressed | ‘w’ key | The player model moves up one tile | The player model moves up one tile | Pass | n/a |
| 3 | **F3** | Test that the player model moves right 1 tile when the d key is pressed | ‘d’ key | The player model moves right one tile | The player model moves right one tile | Pass | n/a |
| 4 | **F3** | Test that the player model moves left 1 tile when the a key is pressed | ‘s’ key | The player model moves down one tile | The player model moves down one tile | Pass | n/a |
| 5 | **F3** | Test that the player model or token does not move without input | n/a | The player model and token remains stationary unless moved | The player model and token remains stationary unless moved | Pass | n/a |
| 6 | **F6** | Test that the player(s) start on the assigned start squares | visual inspection | The player(s) start on the assigned start squares | The player(s) start on the assigned start squares | Pass | n/a |
| 7 | **F11** | The player can roll the dice and receive a random number between 2 and 12 | Click the dice button | The player receives a random number between 1 and 12 | The player receives a random number between 1 and 12 | Pass | n/a |
| 8 | **F16** | Test that there is a visual animation so signify the dice roll | Click the dice button | There is an dice animation when clicked | There is no animation when the dice button is clicked | Fail | Add an animation triggered on the button click |
| 9 | **F15** | Test that everytime the player moves by a tile, the move count is decremented | Move using keys and watch movement counter | The player cannot move further when the move count is 0 | The player cannot move further when the move count is 0 | Pass | n/a |
| 10 | **F23** | Test that players cannot travel through the walls | Use arrow keys to attempt to walk through any wall | The player will be stopped at the wall and the movement count will not be decremented | The player will be stopped at the wall and the movement count will not be decremented | Pass | n/a |
| 11 | **D3** | Test that the player model is appropriate and does not obviously offend any cultural groups | Visual inspection | The play model is appropriate and does not obviously offend any cultural groups | The play model is appropriate and does not obviously offend any cultural groups | Pass | n/a |
| 12 | **F24** | Test that no two players can occupy the same tile space | Use arrow keys to attempt to position player onto another | That the player is stopped a tile before the other player and the movement count is not decremented | That the player is stopped a tile before the other player and the movement count is not decremented | Pass | n/a |
| 13 | **F3** | Test that players can only move between tiles and cannot land between tiles | Use arrow keys to move | Players only travel on tiles, not between | Players only travel on tiles, not between | Pass | n/a |
| 14 | **F3** | Test that players cannot travel out of the game board | Click on tile outside game borders | Players do not move outside the board, they stop at the edge | Players do not move outside the board, they stop at the edge | Pass | n/a |
| 15 | **F25** | Test that the player can choose to remain stationary during their turn | Click on the ! key to remain stationary | When the ! key is pressed, the player remains stationary and the next players’ turn occurs | When the ! key is pressed, the player remains stationary and the next players’ turn occurs | Pass | n/a |
| 16 | **F3** | Test that the player cannot move if the movement count is 0 | Use arrow keys to move once movement count is 0 | Players cannot move if their move count is 0 | Players cannot move if their move count is 0 | Pass | n/a |
| 17 | **F23** | Test that players can only enter rooms through the doors | Use arrow keys to move player into room through door | Players can only enter rooms through doors | Players can only enter rooms through doors | Pass | n/a |
| 18 | **F23** | Test that upon entering a room, the player movement count ends | Use arrow keys to enter room and observe movement count | Upon entering a room, the player movement count ends | Upon entering a room, the player movement count ends | Pass | n/a |
| 19 | **NF3** | Test that the code for the player is programmed in python | Visual inspection | The code is written in python and only python | The code is written in python and only python | Pass | n/a |
| 20 | **NF4** | Test that the code for the player movement is programmed cleanly and allows for easy maintenance and updates | Visual inspection | The code is written well and allows for easy updates and changes if called | The code is written well and allows for easy updates and changes if called | Pass | n/a |

|  |
| --- |
| 1. **Summary of sprint** |
| *You should consider and discuss:*   * *Did you achieve your objectives for this sprint?*   + Yes, all the mandatory requirements for this sprint were met. * *Is there a working prototype?*   + Yes. The following image is the text output for the movement. * *What went well, and what did not go well? If things did not go well, what have you learned and what will you do differently for the next sprint?*   + Initially we did not keep the split between the UI and the backend, once programmed the cli and hardcoding the movement within the cli, we had to move it into the player token. Once we did this, everything could be simplified in the cli, and the split was created.   + Once the split was made, everything became simple to implement. The player movement works correctly, allowing the use to move around the entire map according to the dice roll. The implementation of the arrow keys made this simpler to finish.   + Most of not all the system tests were successful which ensure that we can focus on the next sprint completely without having to worry about fixing any issues in this sprint |