Justin Hughes & Yacine Belaifa

COMP 3721

Dr. Laurie Ricker

Robot Turtles Milestone 5

December 12, 2020

**Use Case Specifications:**

**Use Case 1: Player Turns Left**

|  |  |
| --- | --- |
| Use Case Name: | Player turns left |
| Actor(s) | The current player, GameModel, GUI, Command Line View, GameController |
| Summary Description: | Some given player is told that it is their turn to play, and in doing so, they opt to turn left within the same square. |
| Priority: | Must Have |
| Status: | Medium level of details |
| Pre-Condition: | The player preceding the current player would have already completed their turn. |
| Post-Conditions: | * The Player would have their direction changed by 90 degrees counterclockwise * The player’s direction is updated in the following way, depending on their current direction: North -> West, West -> South, South -> East, East -> North * The View (both command line and GUI) is updated to reflect the current player’s change in direction. |

|  |  |
| --- | --- |
| Basic Path: | 1. The current player types in the command line that they would like to turn left. 2. The GameModel verifies that the current player has at least one leftTurn card in their deck. 3. The turn\_left() method of the MoveFunctions class takes care of orienting the position of the current player in the appropriate direction. 4. Information is returned about the current player’s Deck, and their new location. 5. The controller updates the View (both command line and GUI) with the status of each Player and Jewel. 6. The next player gets prepared to make their turn. |

|  |  |
| --- | --- |
| Alternative Paths: | 1a. There are no remaining left turn cards remaining in the player’s hand.   * In this case, the player must choose another card (or rule) to play. Use cases for these other methods follow.   3a. The player chooses to use the Bug to undo their turn   * In this case, the “left turn” card is re-added to the Player’s deck, the player is re-oriented in the direction he was in prior to the start of the turn, and the player makes another turn. |

|  |  |
| --- | --- |
| Business Rules: | B1. Number of left turn cards that each player has in their hand  B2. Left turn card must be played before the turtle’s direction is changed  B3: Each player is only allowed one move. |

|  |  |
| --- | --- |
| Non-Functional Requirements: | NF1. The time between when the previous player ends their turn and when the current player chooses to turn left. |

**Use Case 2: Player Turns Right**

|  |  |
| --- | --- |
| Use Case Name: | Player turns right |
| Actor(s): | The current Player, Controller, RobotTurtlesGUI and CommandLineView |
| Summary Description: | Some given player is told that it is their turn to play, and in doing so, they opt to turn their turtle right within the same square. |
| Priority: | Must Have |
| Status: | Medium level of details |
| Pre-Condition: | The player preceding the current player would have already completed their turn. |
| Post-Conditions: | * The Player would have their direction changed by 90 degrees clockwise * These are updated in the following way, depending on their current direction: North -> East, East -> South, South -> West, West -> North * The View (both command line and GUI) is updated to reflect the current player’s change in direction. |

|  |  |
| --- | --- |
| Basic Path: | 1. The current Player types in the command line that they would like to turn right. 2. The GameModel verifies that the current player has at least one rightTurn card in their deck. 3. The GameModel calls the turn\_right() method of the MoveFunctions class, which takes care of orienting the position of the current player in the appropriate direction. 4. Information is returned about the current player’s Deck, and their new location. 5. The controller updates the GUIView with the status of each Player and Jewel. 6. The next player gets prepared to make their turn. |

|  |  |
| --- | --- |
| Alternative Paths: | 1a. There are no remaining right turn cards remaining in the player’s hand.   * In this case, the player must choose another card (or rule) to play. Use cases for these other methods follow.   3a. The player chooses to use the Bug to undo their turn   * In this case, the “right turn” card is re-added to the Player’s deck, the Player is re-oriented in the direction he was in prior to the start of the turn, and the player makes another turn. |

|  |  |
| --- | --- |
| Business Rules: | B1. Number of right turn cards that each player has in their hand  B2. Right turn card must be played before the turtle’s direction is changed  B3: Each player is only allowed one move. |

|  |  |
| --- | --- |
| Non-Functional Requirements: | NF1. The time between when the previous player ends their turn and when the current player chooses to turn right. |

**Use Case 3: Player Goes Forward**

|  |  |
| --- | --- |
| Use Case Name: | Player Goes Forward |
| Actor(s): | The current player, Controller, GameModel, GUIView and CommandLineView |
| Summary Description: | Some given player is told that it is their turn to play, and in doing so, they opt to move their turtle forward one square (from the direction they are currently facing). |
| Priority: | Must Have |
| Status: | Medium level of details |
| Pre-Condition: | The player preceding the current player would have already completed their turn. |
| Post-Conditions: | * The Player is moved by one square (in the same direction) from the square they previously occupied. * The controller will call on the View to physically display the player 1 square away from their previous location. |

|  |  |
| --- | --- |
| Basic Path: | 1. The current Player types in the command line that they would like to move forward. 2. The GameModel verifies that the current player has at least one forward card in their deck. 3. The GameModel verifies that the player is not facing a wall. 4. The GameModel calls the forward() method of the MoveFunctions class, which takes care of advancing the Player . 5. Information is returned about the current player’s Deck, and their new location. 6. The GameController updates the GUIView with the status of each Player and Jewel. 7. The next player gets prepared to make their turn. |

|  |  |
| --- | --- |
| Alternative Paths: | 1a. There are no remaining move forward cards remaining in the player’s hand.   * In this path, the player must choose another card (or rule) to play.   3a. The player chooses to use “Bug” to undo their turn   * In this path, the player reclaims their move forward card, and the Player is moved to the space they were in prior to the start of the turn * Once this happens, the Player can take their turn.   3b. Prior to taking their turn, the player is facing either a corner or the wall.   * In this situation, the player should invoke their “Bug” card, and make a different move.   4a. As the player moves, they collect their appropriately – colored jewel.   * In this case, the game ends, and this player is declared the winner. |

|  |  |
| --- | --- |
| Business Rules: | B1. Number of forward cards that each player has in their hand.  B2. Forward card must be played before the player moves ahead.  B3: Each player is only allowed one move.  B4: In alternative path 4a, there must be at least 1 jewel on the board. |

|  |  |
| --- | --- |
| Non-Functional Requirements: | NF1. The time between when the previous player ends their turn and when the current player chooses to go forward. |

**Use Case 4: Bug Card**

|  |  |
| --- | --- |
| Use Case Name: | Bug Card |
| Actor(s): | The Current Player and Controller, RobotTurtlesGUI and CommandLineView |
| Summary Description: | The current player has just made a move, but has just changed their mind, and wants to reverse what they have done to restore their previous position. |
| Priority: | Must Have |
| Status: | Medium Level of Details |
| Pre-Condition: | * The player will have completed a move: either going forward, turning left, or turning right. * The player will be dissatisfied with the code they have written. This could be for various reasons, such as thinking of a better action they could accomplish with their turn or running into a wall with their existing code. |
| Post-Conditions: | * The player will have completely undone the turn they just played, and it will be their turn once again. Now, they can proceed with making a different move. |

|  |  |
| --- | --- |
| Basic Path: | 1. The current player chooses to either move forward, turn left, or turn right. The Controller calls the appropriate function from the Move class (use cases for these above), and the player’s position/direction is changed. 2. The current player provides input to the program to indicate that they would like to use the Bug to undo their move. 3. The Controller retrieves this user input from the user interface and retracts the move. 4. The player’s deck is changed: specifically, they will have one more card of the move they just undid then they did before. 5. Ultimately, the Player will have all cards back in their hand and they will occupy the position they occupied in the previous turn. 6. The player makes their new, official turn. |

|  |  |
| --- | --- |
| Alternative Paths: | 3a. The player is unable to complete their initial turn, since their sequence of code involves running into a corner   * In this path, the user will be notified that their move was invalid, and they will be prompted to enter “YES” to undo their move. * Since the error message about the move being invalid comes first, the Player’s deck will not be updated. |

|  |  |
| --- | --- |
| Business Rules: | B1. The player is only able to use the Bug call  B2. The Bug call is only able to undo 1 move. |

|  |  |
| --- | --- |
| Non-Functional Requirements: | NF1. Time for the user to decide on using the “Bug” call or not. |

**Use Case 5: Collect Jewel**

|  |  |
| --- | --- |
| Use Case Name: | Collect Jewel |
| Actor(s): | Current Player, Controller, RobotTurtlesGUI and CommandLineView |
| Summary Description: | The current player takes a turn so that it is possible for them to collect one of the jewels and finish their gameplay. |
| Priority: | Must Have |
| Status: | Medium Level of Details |
| Pre-Condition: | The player preceding the current player would have already completed their turn. |
| Post-Conditions: | * The current player will have finished collecting their jewel and be done their gameplay. * If all the other players have taken their jewels already, the game ends. * Otherwise, this player is removed from the listing of all players and gameplay continues as normal with the remaining players attempting to locate their jewels. |

|  |  |
| --- | --- |
| Basic Path: | 1. The current player prepares their turn in a way such that they will be able to capture the jewel. 2. The player and the Jewel appear on the same square. 3. The Player and the jewel both disappear from the screen. 4. The next person begins playing their turn. |

|  |  |
| --- | --- |
| Alternative Paths: | 2a. Two possible paths for collecting the jewel:   * If the current player is the last player to collect their jewel, then the game ends. * Otherwise, this player is removed from the listing of active players, and the remaining players continue taking turns until they collect their jewel. |

|  |  |
| --- | --- |
| Business Rules: | B1. There must be at least 1 player and 1 jewel remaining on the board.  B2. It does not matter what color jewel each player goes for.  B3. For this implementation, each player collects exactly 1 jewel. |

|  |  |
| --- | --- |
| Non-Functional Requirements: | NF1. The time between when the previous player ends their turn and when the current player begins the process of collecting their jewel. |

**Use Case 6: Win Game**

|  |  |
| --- | --- |
| Use Case Name: | Win Game |
| Actor(s): | Current Player, Controller,GameModel, RobotTurtlesGUI and CommandLineView |
| Summary Description: | The final player who has not yet collected their jewel will do so, and in turn, will win the game. |
| Priority: | Must Have |
| Status: | Medium Level of Details |
| Pre-Condition: | The last remaining player will collect their jewel. |
| Post-Conditions: | The game will be complete, with a message being sent to the user along the lines of “Congratulations! All players have found their jewels. Stay tuned for further updates to this game!” |

|  |  |
| --- | --- |
| Basic Path: | 1. The final player of the game moves one square forward in such a way that they will capture the jewel. 2. The jewel and the player are currently on the same square on the GUI. 3. The player and the jewel both disappear from the GUI. 4. The victory message is output to the program user. |

|  |  |
| --- | --- |
| Alternative Paths: | 1a. There are no remaining move forward cards remaining in the player’s hand.   * In this path, the player must choose another card (or rule) to play. They will not be able to win. |

|  |  |
| --- | --- |
| Business Rules: | B1. There must be exactly 1 player and 1 jewel remaining on the board.  B2. Prior to starting their turn, the current Player must be adjacent to and in the direction of the Jewel. |

|  |  |
| --- | --- |
| Non-Functional Requirements: | No specific non-functional requirements. |

**Use Case 7: Start Game**

|  |  |
| --- | --- |
| Use Case Name: | Start Game |
| Actor(s): | CommandLineView, Controller,GameModel,GUIView |
| Summary Description: | Once user decisions (the number of players, the color of each player, and the color of each jewel) are made, then gameplay begins. |
| Priority: | Must Have |
| Status: | Medium Level of Details |
| Pre-Condition: | Some of the basic settings for beginning the game will be put into play, such as deciding how many players there are, deciding which color each player is, and what color jewels are placed on the GUI. Users will be able to provide these options once the application launches, and prior to officially starting. |
| Post-Conditions: | The board will be displayed, showing the active locations of the players and the jewels. |

|  |  |
| --- | --- |
| Basic Path: | 1. The user is prompted to enter several settings in the Command Line View:  a) The number of players actively playing the game  b) The colors of the players used  c) The colors of the jewels used  2. The user makes their selections for these settings.  3. Gameplay begins, with the first player entered taking the first turn.  4. The GUI is updated with the current positions of the Players and jewels. |

|  |  |
| --- | --- |
| Alternative Paths: | 4a. The user does not actively start playing the game   * In this case, the screen waits idly for user input: until it is either closed, or the user selects to play. |

|  |  |
| --- | --- |
| Business Rules: | B1. There must be at least 1 player and 1 jewel selected.  B2. It does not matter what jewel each player goes for.  B3. There must be exactly as many jewels as there are players.  B4. No more than 4 players and 4 jewels may be selected. |

|  |  |
| --- | --- |
| Non-Functional Requirements: | NF1. The time between when the user selects their initial settings and when “Start Game” is selected.  NF2. The time between when “Start Game” is pressed and when the board is loaded. |

**Use Case 8: Check Validity of Move**

|  |  |
| --- | --- |
| Use Case Name: | Check Validity of Move |
| Actor(s): | Current Player, Controller, GameModel |
| Summary Description: | The player decides on a move that they would like to execute, and the Controller verifies that this move is possible. |
| Priority: | Must Have |
| Status: | Medium Level of Details |
| Pre-Condition: | The player preceding the current player would have already completed their turn. |
| Post-Conditions: | The program will determine whether the move is invalid or valid and call on the appropriate code. |

|  |  |
| --- | --- |
| Basic Path: | 1. The current player types in the Command Line the type of move they would like to perform. 2. The GameModel scans the surroundings of the player (both the surrounding positions and directions) to ensure that they can perform their desired move without running into any walls. 3. Based on whether the player collides with a wall or not, the GameModel determines if the move is valid or invalid. Use cases for these conditions follow. |

|  |  |
| --- | --- |
| Alternative Paths: | 3a. The move is deemed valid:   * In the event where the move is deemed valid, then a message is sent out (using the controller), informing the user of this and calling on the valid move use case.   3b. The move is deemed invalid:   * In the event where the move is deemed invalid, then the user is informed of this. The use case for an invalid move follows. |

|  |  |
| --- | --- |
| Business Rules: | B1. There must be at least 1 player and 1 jewel remaining on the board.  B2. The player must have a minimum of one move card in their hand (whether it is a Move Forward, Turn Left, or Turn Right). |

|  |  |
| --- | --- |
| Non-Functional Requirements: | NF1. The effectiveness of the code used to check if a player’s surrounding squares are occupied. |

**Use Case 9: Display GUI**

|  |  |
| --- | --- |
| Use Case Name: | Display GUI |
| Actor(s): | RobotTurtlesGUI, Controller |
| Summary Description: | After the game is initialized with the appropriate settings, the board is displayed for all players to see not only their positions, but the locations of the jewels and of the other players. |
| Priority: | Must Have |
| Status: | Medium Level of Details |
| Pre-Condition: | The start button was previously clicked to initialize the game. |
| Post-Conditions: | The first player prepares to take their turn. |

|  |  |
| --- | --- |
| Basic Path: | 1. The system retrieves all the settings regarding the number of players, number of jewels, and color choices that were specified by the user prior to starting the game. 2. The Controller creates an instance of the GUI, first setting up the grid, then drawing jewels and players that were specified. 3. The board is shown to the user of the program. |

|  |  |
| --- | --- |
| Alternative Paths: | There are no alternative paths the board can take. Ensuring that the board is working and rendering properly is also quite significant to the remaining behaviors of this game. |

|  |  |
| --- | --- |
| Business Rules: | B1. There must be at least 1 player and 1 jewel initialized on the board.  B2. The colors of the jewels and players do not have to match.  B3. The players and jewels have fixed starting positions as specified in the rules of Robot Turtles. |

|  |  |
| --- | --- |
| Non-Functional Requirements: | NF1. The time between when the user starts the game and when the first player makes their first move. |

**Use Case 10: Update GUI**

|  |  |
| --- | --- |
| Use Case Name: | Update GUI |
| Actor(s): | RobotTurtlesGUI, Controller |
| Summary Description: | After the player has just made a turn, the GUI is updated to show the player’s updated position and direction. |
| Priority: | Must Have |
| Status: | Medium Level of Details |
| Pre-Condition: | The start button was previously clicked to initialize the game. |
| Post-Conditions: | The first player prepares to take their turn. |

|  |  |
| --- | --- |
| Basic Path: | 1. The system retrieves information about the move the user would like to make. 2. The current GUI frame is hidden temporarily to update the GUI. 3. The revalidate() method is called to erase the current settings of the window 4. The repaint() method is called to paint the new image, with the updated player’s position and direction. 5. The frame is shown once again, with the updated changes. |

|  |  |
| --- | --- |
| Alternative Paths: | There are no alternative paths this can take. Ensuring that the board is rendering properly is quite significant to the remaining behaviors of this game. |

|  |  |
| --- | --- |
| Business Rules: | B1. There must be at least 1 player and 1 jewel initialized on the board.  B2. The colors of the jewels and players do not have to match.  B3. There must not be any 2 players or jewels occupying the same position on the board. |

|  |  |
| --- | --- |
| Non-Functional Requirements: | NF1. The time between when the user starts the game and when the updated board is shown. |

**Use Case 11: Make a Move**

|  |  |
| --- | --- |
| Use Case Name: | Make a Move |
| Actor(s): | Current Player |
| Summary Description: | This condition deals with a player being prompted to take their turn: and, more specifically, to make a move. |
| Priority: | Must Have |
| Status: | Medium Level of Details |
| Pre-Condition: | The previous player would have completed their turn. |
| Post-Conditions: | The move specified by the current player will be checked to see if it can be done. |

|  |  |
| --- | --- |
| Basic Path: | 1. The program waits for the previous player to complete their turn. 2. The current player receives a prompt asking for a move selection: either to turn left, turn right, or move forward. 3. The appropriate move method is called, and its validity is checked. |

|  |  |
| --- | --- |
| Alternative Paths: | 1a. The move being made is the first of the game   * In this case, there is no waiting for the previous player to complete their move: the current player simply starts the game. |

|  |  |
| --- | --- |
| Business Rules: | B1. For this method to be called, there must be at least 1 active player who has not yet found a jewel. |

|  |  |
| --- | --- |
| Non-Functional Requirements: | NF1. The time between when the player has specified their move of choice, and when the validity of the move is checked. |

**Use Case 12: Valid Move**

|  |  |
| --- | --- |
| Use Case Name: | Valid Move |
| Actor(s): | Current Player, Controller, GameModel |
| Summary Description: | This use case uses the fact that the desired move (whose status was called on by the Controller) is a valid move, and calls the function associated with this desired move. |
| Priority: | Must Have |
| Status: | Medium Level of Details |
| Pre-Condition: | The controller verified that the user’s desired move was a valid move. |
| Post-Conditions: | The user’s desired move will be performed. |

|  |  |
| --- | --- |
| Basic Path: | 1. The controller completes its verification that the desired move is valid. 2. The controller calls on MoveFunctions to use the appropriate function. |

|  |  |
| --- | --- |
| Alternative Paths: | Not applicable |

|  |  |
| --- | --- |
| Business Rules: | B1. There must be at least 1 player and 1 jewel remaining on the board.  B2. The current player must have at least 1 card describing the move they are choosing to make. |

|  |  |
| --- | --- |
| Non-Functional Requirements: | NF1. The time between when the validity of the player’s move is checked, and when the move is made. |

**Use Case 13: Invalid Move**

|  |  |
| --- | --- |
| Use Case Name: | Invalid Move |
| Actor(s): | Current Player, Controller, Check Validity |
| Summary Description: | This condition occurs if the user’s move is proven to be invalid and urges the user to use the Bug card to reset their move. |
| Priority: | Must Have |
| Status: | Medium Level of Details |
| Pre-Condition: | The controller verified that the user’s desired move was an invalid move. |
| Post-Conditions: | The Bug card is called and the user’s move is reset. |

|  |  |
| --- | --- |
| Basic Path: | 1. The controller completes its verification that the desired move is invalid. 2. The controller outputs a statement describing this and urges the current player to use their Bug card to undo their move. 3. The Player uses their Bug card, undoing the move and going about a new move. |

|  |  |
| --- | --- |
| Alternative Paths: | 3 a) The Player refuses to use the Bug card   * If they refuse to use the bug card, a firm error message will be output indicating that their move was invalid and needed to be undone. |

|  |  |
| --- | --- |
| Business Rules: | B1. There must be at least 1 player and 1 jewel remaining on the board. |

|  |  |
| --- | --- |
| Non-Functional Requirements: | NF1. The time between when the previous player ends their turn and when the current player attempts to make their move. |

**Use Case 14: No Forward Cards Left**

|  |  |
| --- | --- |
| Use Case Name: | No Forward Cards Remaining |
| Actor(s): | Current Player, Controller, Deck |
| Summary Description: | This condition occurs if the user has no more forward cards remaining. If they have any left or right turn cards remaining, then gameplay continues. Otherwise, it stops. |
| Priority: | Must Have |
| Status: | Medium Level of Details |
| Pre-Condition: | The player has used 18 different move forward cards. |
| Post-Conditions: | 4 possibilities:   * Player has no cards left at all, and they have no more turns. * Player can keep moving with only left turns. * Player can keep moving with only right turns. * Player can keep moving with left and right turns. |

|  |  |
| --- | --- |
| Basic Path: | 1. The player attempts to move forward. 2. The controller outputs a statement to the user that their Deck does not contain any more forward cards. 3. The remaining play varies depending on what the Player has left in their hand. |

|  |  |
| --- | --- |
| Alternative Paths: | 3 (a) Player has no cards left at all, and they have no more turns.   * Here, there is a message printed saying that the Player has run out of cards, and their turn is skipped over.   3 (b) Player can keep moving with only left turns.   * Here, there is a message printed saying that the Player has run out of forward cards, and they can redo their turn. * If they later try to print “right turn”, then they will also get an error message.   3(c) Player can keep moving with only right turns.   * Here, there is a message printed saying that the Player has run out of forward cards, and they can redo their turn. * If they later try to print “left turn”, then they will also get an error message.   3(d) Player can keep moving with left and right turns   * Here, there is a message printed saying that the Player has run out of forward cards, and they can redo their turn. * They can choose from doing ether type of turn. |

|  |  |
| --- | --- |
| Business Rules: | B1. The player must not have any move forward cards in their hand. |

|  |  |
| --- | --- |
| Non-Functional Requirements: | NF1. Not Applicable |

**Use Case 15: No Left Cards Remaining**

|  |  |
| --- | --- |
| Use Case Name: | No Left Cards Remaining |
| Actor(s): | Current Player, Controller, Deck |
| Summary Description: | This condition occurs if the user has no more cards remaining. If they have any left or right turn cards remaining, then gameplay continues. Otherwise, it stops. |
| Priority: | Must Have |
| Status: | Medium Level of Details |
| Pre-Condition: | The player has used 8 different turn left cards. |
| Post-Conditions: | 4 possibilities:   * Player has no cards left at all, and they have no more turns. * Player can keep moving with only right turns. * Player can keep moving with only move forward cards. * Player can keep moving with move forward cards and right turns. |

|  |  |
| --- | --- |
| Basic Path: | 1. The player attempts to turn left. 2. The controller outputs a statement to the user that their Deck does not contain any turn left cards. 3. The remaining play varies depending on what the Player has left in their hand. |

|  |  |
| --- | --- |
| Alternative Paths: | 3 (a) Player has no cards left at all, and they have no more turns.   * Here, there is a message printed saying that the Player has run out of cards, and their turn is skipped over.   3 (b) Player can keep moving with only right turns.   * Here, there is a message printed saying that the Player has run out of forward cards, and they can redo their turn. * If they later try to print “move forward”, then they will also get an error message.   3(c) Player can keep moving with only move forward.   * Here, there is a message printed saying that the Player has run out of forward cards, and they can redo their turn. * If they later try to print “right turn”, then they will also get an error message.   3(d) Player can keep moving with move forward and right turns   * Here, there is a message printed saying that the Player has run out of turn left and they can redo their turn. * They can choose either type of turn. |

|  |  |
| --- | --- |
| Business Rules: | B1. The player must not have any turn left cards in their hand. |

|  |  |
| --- | --- |
| Non-Functional Requirements: | NF1. Not Applicable |

**Use Case 16: No Right Turn cards remaining**

|  |  |
| --- | --- |
| Use Case Name: | No Right Turn Cards Remaining |
| Actor(s): | Current Player, Controller, Deck |
| Summary Description: | This condition occurs if the user has no more right turn cards remaining. If they have any left turn or move forward cards remaining, then gameplay continues. Otherwise, it stops. |
| Priority: | Must Have |
| Status: | Medium Level of Details |
| Pre-Condition: | The player has used 8 different turn right cards. |
| Post-Conditions: | 4 possibilities:   * Player has no cards left at all, and they have no more turns. * Player can keep moving with only left turns. * Player can keep moving with only move forward cards. * Player can keep moving with move forward cards and left turns. |

|  |  |
| --- | --- |
| Basic Path: | 1. The player attempts to turn right. 2. The controller outputs a statement to the user that their Deck does not contain any turn right cards. 3. The remaining play varies depending on what the Player has left in their hand. |

|  |  |
| --- | --- |
| Alternative Paths: | 3 (a) Player has no cards left at all, and they have no more turns.   * Here, there is a message printed saying that the Player has run out of cards, and their turn is skipped over.   3 (b) Player can keep moving with only left turns.   * Here, there is a message printed saying that the Player has run out of forward cards, and they can redo their turn. * If they later try to print “move forward”, then they will also get an error message.   3(c) Player can keep moving with only move forward.   * Here, there is a message printed saying that the Player has run out of forward cards, and they can redo their turn. * If they later try to print “left turn”, then they will also get an error message.   3(d) Player can keep moving with move forward and left turns   * Here, there is a message printed saying that the Player has run out of turn right cards and they can redo their turn. * They can choose either type of turn. |

|  |  |
| --- | --- |
| Business Rules: | B1. The current player must not have any turn right crds in their hand. |

|  |  |
| --- | --- |
| Non-Functional Requirements: | NF1. Not Applicable |

**Use Case 17: No Cards Remaining**

|  |  |
| --- | --- |
| Use Case Name: | No Cards Remaining |
| Actor(s): | Current Player, Controller, Deck |
| Summary Description: | This condition occurs if the player tries to make a move, but they have no cards left on their deck. |
| Priority: | Must Have |
| Status: | Medium Level of Details |
| Pre-Condition: | The player has used all their cards. |
| Post-Conditions: | The player does not have any more turns. |

|  |  |
| --- | --- |
| Basic Path: | 1. The Player does not have any cards remaining. |

|  |  |
| --- | --- |
| Alternative Paths: | Not Applicable |

|  |  |
| --- | --- |
| Business Rules: | B1. There must be at least 1 player and 1 jewel remaining on the board. |

|  |  |
| --- | --- |
| Non-Functional Requirements: | Not Applicable |

**Use Case 18 Function frog check**

|  |  |
| --- | --- |
| Use Case Name: | Function frog check |
| Actor(s): | Current Player, Controller, Deck |
| Summary Description: | This condition occurs if the player does not have a valid function (ie: function does not contain frog card ) |
| Priority: | Must Have |
| Status: | Medium Level of Details |
| Pre-Condition: | The player should have function frog in main program |
| Post-Conditions: | 2-possibilities:   * Player has no frog cards left at all. * Player has no frogs cards left but can player other moves |

|  |  |
| --- | --- |
| Basic Path: | 1. The player attempts to use frog card 2. The controller outputs a statement to the user that their Deck does not contain any frog cards 3. The remaining play varies depending on what the Player has left in their hand. |

|  |  |
| --- | --- |
| Alternative Paths: | Player can use other move cards |

|  |  |
| --- | --- |
| Business Rules: | B1. The player must have a frog card in order to play function frog |

|  |  |
| --- | --- |
| Non-Functional Requirements: | NF1. Not Applicable |

**Use Case 19 : not enough cards to play3**

|  |  |
| --- | --- |
| Use Case Name: | Not enough cards to play 3 |
| Actor(s): | Current Player, Controller, Deck |
| Summary Description: | This condition occurs if the user has less than 3 cards remaining |
| Priority: | Must Have |
| Status: | Medium Level of Details |
| Pre-Condition: | The player has used 18 different move forward cards. |
| Post-Conditions: | Player has to play with single cards |

|  |  |
| --- | --- |
| Basic Path: | 1. The player attempts to play 3 moves 2. The controller outputs a statement to the user that their Deck does not have enough cards 3. The remaining play varies depending on what the Player has left in their hand. |

|  |  |
| --- | --- |
| Alternative Paths: | Player can play a single move card |

|  |  |
| --- | --- |
| Business Rules: | B1. The player must not have exactly 3 cards in had to play3 |

|  |  |
| --- | --- |
| Non-Functional Requirements: | NF1. Not Applicable |