**Use Case #1:** Initiate the game and traverse the maze.

**Primary actor:** Player

**Goal in context:** To view the maze after opening the application and traverse it using the arrow keys on the player's keyboard.

**Preconditions:** The game application must be successfully downloaded and installed.

**Trigger:** The player decides to play the Cheese Run desktop game.

**Scenario:**

1. The player double clicks on the icon for the Cheese Run desktop game to launch it.

2. The system displays the homescreen and the major function button labelled "Play."

3. The player clicks the "Play" button

4. The system displays the game screen where the maze layout is displayed along with all of its components at their starting positions:
   mouse(at the entrance),
   cats(at random locations on the maze),
   mousetraps(at random locations on the maze),
   cheese(at random locations on the maze), and
   organic cheese(at random locations on the maze).
   The game screen also displays the player's current score, time elapsed, and the number of cheese pellets eaten; all of which are set to 0 at the beginning of each game.

5. The player presses one of the arrow keys on their keyboard to move the mouse character in the direction corresponding with the arrows' direction (e.g: up, down, left, right) one cell at a time.

6. The system performs a check to see whether the given input for each movement is valid. For example, if the down arrow key is pressed and the cell directly below the mouse is not obstructed by a barrier object, the system moves the mouse to that new cell (See use case #2, #3, and #4 for other types of valid movements). If the cell is obstructed by a maze barrier, the mouse is not moved; however, the system moves all cats to an adjacent unobstructed cell that brings them closer to the mouse.

**Exceptions:**

1) Due to slow response by the system when rendering the maze, the player clicks the “Play” button multiple times which results in these requests queuing inappropriately and generating an error condition.

2) The player exits the application by clicking the back button at the bottom of the window or the “X” at the top left of the window: use case #5.

3) Player move to a cell which is not obstructed by a barrier but has a reward, or enemy on it: see use cases #2, #3,and #4.

**Priority:** High Priority: movement encompasses the basic functionality of the game.

**When Available:** First increment.

**Frequency of use:** High: character movement is the essence of the game

**Channel to actor:** Desktop with Java RunTime Environment.

**Open Issues:**

1) Will the system be able to render the maze and make updates in a responsive manner?
2) Will there be multiple levels with different maze layouts available for the user to play?

**Use Case #2:** Eating a Cheese pellet.

**Primary actor:** Player

**Goal in context:** To move to a cell on the maze which has a cheese pellet or an organic cheese pellet, collect it and update the maze and the score accordingly.

**Preconditions:** The maze should be successfully rendered on the game screen and the mouse should be adjacent to a cell containing a cheese pellet.

**Trigger:** The player wants to collect the cheese pellet adjacent to the mouse while in the game.

**Scenario:**

1. The player presses an arrow key to move to an adjacent cell which contains a cheese pellet.
2. The system removes the cheese pellet from the maze and adds the reward value associated with that cheese pellet to the player's score. Also, the system updates the number of cheese pellets collected which is displayed above the maze.

**Exceptions:**

1) If there is a cat in a cell adjacent to the cheese pellet, moving to that cell will cause the player to lose the game because that movement will result in a collision. See use case #4.

**Priority:** High Priority: collection of cheese is required for the successful completion of a given round of the game.

**When Available:** First increment.

**Frequency of use:** Frequent, the player has to collect the cheese in order to win the game.

**Channel to actor:** Desktop with Java RunTime Environment.

**Open Issues:** None

**Use Case #3:** Stepping on mousetrap.

**Primary actor:** Player

**Goal in context:** The player moves the mouse to a cell on the maze occupied by a mousetrap.

**Preconditions:** The mouse must be on a cell which is adjacent to a cell containing a mousetrap.

**Trigger:** The player gets cornered by a cat and has to move to a cell with a mouse trap to avoid colliding with the cat.

**Scenarios:**

1. The player presses an arrow key which moves the mouse to a cell containing a mousetrap.
2. The system removes the mouse trap from the maze and deducts the punishment amount from the player's score.

**Exceptions:**

1) If the mousetrap is adjacent to a cat, moving to that cell will cause the player to lose the game because that movement will result in a collision. See use case #4.
2) If the player's total score is less than the mousetrap's punishment value, see Use Case #7.

**Priority:** Medium Priority: The mousetraps can be added after the maze structure and the mouse's movement have been implemented.

**When Available:** First increment.

**Frequency of use:** Frequent: every map will contain many mouse traps.

**Channel to actor:** Desktop with Java RunTime Environment.

**Open Issues:** None

**Use Case #4:** Losing a game due to collision with a cat.

**Primary actor:** Player

**Goal in context:** To have the mouse collide with one of the cats and cause the game to be over.

**Preconditions:** The mouse must be at most one cell away from a cat with no barriers in between them.

**Trigger:** Mouse gets trapped in a corner of the maze by a cat.

**Scenario:**

1. The player presses an arrow key which either moves the mouse to a cell that a cat occupies, or moves the mouse to a cell to which a cat was already adjacent to.

2. The system detects the collision, removes the mouse from the map, places the cat on the mouse's final square if it wasn't already there, and displays the "Game Over" screen with the "Back to Homepage" button.

3. The player clicks the "Back to Homepage button" and returns to homepage.

**Exceptions:**

1) Instead of pressing the "Back to Homepage button" the player can exit the application using the "X" on the top left of the window if they don't wish to play again.

**Priority:** High Priority: The cats are the primary mechanism for losing the game.

**When Available:** First increment.

**Frequency of use:**Frequent : Every maze will contain many cats.

**Channel to actor:** Desktop with Java RunTime Environment.

**Open Issues:**

1. Does a player have multiple lives?
2. Should we have a limited number of cats in different levels of games?

**Use Case #5:** Exit Mid-Game

**Primary actor:** Player

**Goal in context:** To exit the game while it is still in progress.

**Preconditions:** The application must be running and the user should be on the game page.

**Trigger:** The player gets bored or needs to do something else so he or she wants to exit the application quickly.

**Scenario:**

1. The player clicks the “X” symbol on the top left of the window.
2. The Application terminates and the window is closed.

**Exceptions:**

1) The player can also first click the” Back to Homepage” button located at the top of the game page first to exit the current game and then click the “X” symbol at the top left of the Homepage.

**Priority:** High Priority: the user needs to be able to have full control in terms of when they want the application to run and when they don’t because the application uses their computer’s resources.

**When Available:** First increment.

**Frequency of use:** Frequent: the “X” button will be available on all pages to allow users to exit the application easily.

**Channel to actor:** Desktop with Java RunTime Environment.

**Open Issues:**

1. If the application crashes, should we start from the beginning?

**Use Case #6:** Winning the game

**Primary actor:** Player

**Goal in context:** To collect all regular cheese pellets on the maze and reach the exit cell while maintaining a positive score.

**Preconditions:** The player must be on the game page.

**Trigger:** Player wants to successfully complete the maze and win with the highest score possible.

**Scenario:**

1. The player initiates the game by moving to a valid cell adjacent to the entrance cell.
2. The system starts time and moves the cats.
3. The player moves the mouse to a cell containing a cheese pellet.
4. System adds the reward amount to the player's score, removes the cheese from the map and moves the cats. **Go back to step 3 and repeat until all cheese is collected
5. The player uses the arrow key to navigate to the exit.
6. The system checks to see whether the player has collected all of the cheese and displays the win screen. The win screen displays the score that the player attained during that play through and how long that session lasted. The win screen also has a "Back to Homepage" button.
7. The player clicks the "Back to Homepage" button.
8. The system displays the homepage.

**Exceptions:**

1) When on the win screen, the player can simply exit using the "X" button on the top left of the screen rather than navigating to the homepage first.
2) The player can also exit mid game if he or she doesn't wish to complete the session. See use case #5
3) In addition to the regular cheese, the player can also try to collect the special reward called organic cheese. The organic cheese appears in the maze randomly for a few

"Ticks" of the game and then disappears. Collecting this cheese is not required to win the game but serves to increase the players overall score.

**Priority:** High Priority: Reaching the exit after collecting all the cheese is the only way to win the game.

**When Available:** First increment.

**Frequency of use:** Frequent: everytime a player plays the game, it is assumed they will try to win the game by collecting all the cheese and reaching exit.

**Channel to actor:** Desktop with Java RunTime Environment.

**Open Issues:** None

**Use Case #7:** Losing the game due to negative score

**Primary actor:** Player

**Goal in context:** To have the player's total score become negative

**Preconditions:** The player must be on the game page; The player's total score is less than mousetraps' punishment value.

**Trigger:** Player moves the mouse character to many mousetraps and causes the score to be negative; Player does not earn enough score prior to stepping on the mousetrap.

**Scenario:**

1. The player moves the mouse character onto a mousetrap.
2. The system deducts the mousetrap's punishment value from the player's total score.
3. The system checks the total score and sees that it is negative.
4. The system displays the "Game Over" screen with the "Back to Homepage" button.
5. The player clicks the "Back to Homepage button".
6. The system displays the Homepage screen.

**Exceptions:**

1) When on the losing screen, the player can simply exit using the "X" button on the top left of the screen rather than navigating to the homepage first.

**Priority:** High Priority: In addition to colliding with moving enemies, having a negative score is another way a player can lose the game.

**When Available:** First increment.

**Frequency of use:** Semi-frequent (Varied based on player's experience with the game.)

**Channel to actor:** Desktop with Java RunTime Environment.

**Open Issues:** None