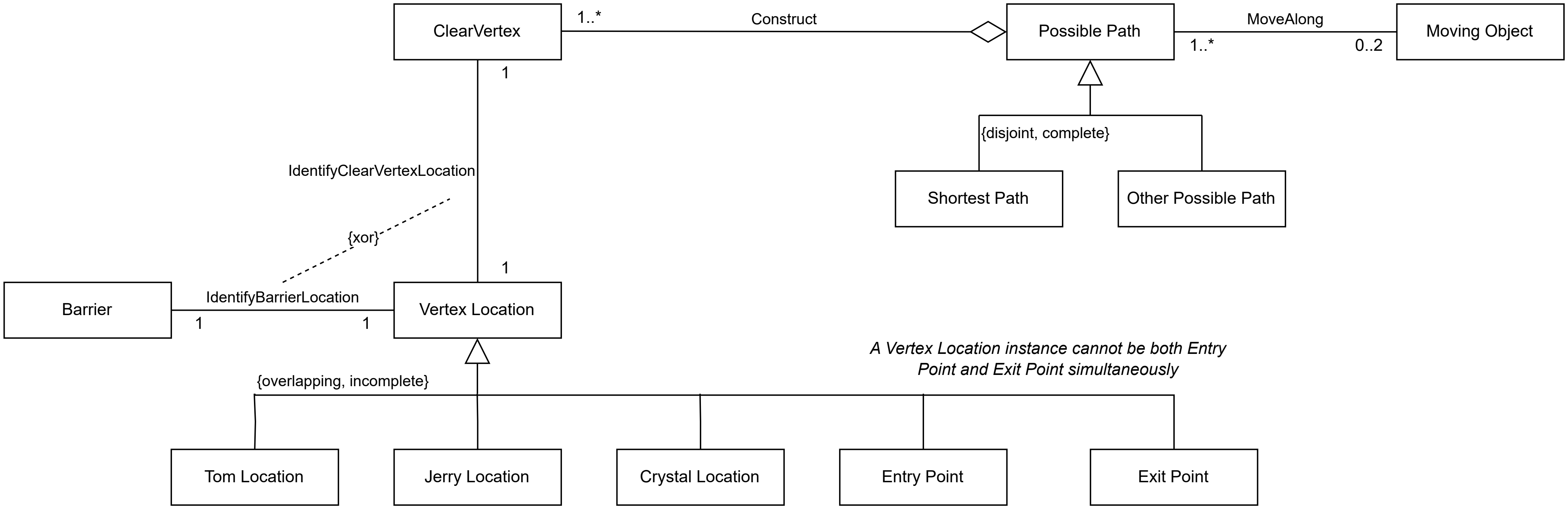
Group 34

Class Diagram



Use Case: Creates Maze Map

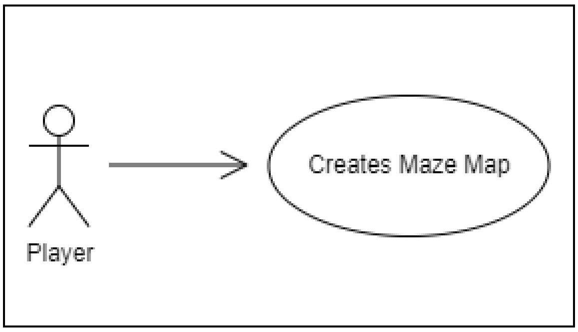
Brief Description

This use case describes the creation and display of a maze map.

Actor

Player

Use-case Diagram



Assumptions

A. The maze map always has a fixed size on the user interface.
   a. A Pixel Square ("PX-Square") is always composed of 10*10 pixels.
   b. The maze map is always displayed in a square-shaped grid with 30*30 PX-Squares.

B. A PX-Square is either a barrier or a clear vertex.
   a. A barrier is always filled by dark grey color on the user interface.
   b. A clear vertex is always filled by white color on the user interface.
   c. A Path is a series of connected clear vertices.

C. Both characters in the game (Tom and Jerry) can only travel along Paths.

D. There is always only one Entry-Point located at the left-side border of the maze.

E. There is always only one Exit-Point located at the right-side border of the maze.

F. The Entry-Point and Exit-Point have different locations.

G. There are always at least 2 possible Paths from Entry-Point to Exit-Point.

Basic Flow

1. The use case begins when the player actor launches the system.
2. The system automatically generates and stores the maze map.
3. The system displays the maze map on the user interface.
4. The use case ends.

Detail Flow

1. The use case begins when the player actor launches the system.

2. The system automatically generates and stores the maze map.
   2.1 The system designs a new maze map subject to constraints set in *Assumptions*.
   2.2 The system exports the maze map data into a .csv file.
      2.2.1 The system creates a 30 * 30 empty matrix.
      2.2.2 For each matrix entry with coordinate $[i, j]$
         2.2.2.1 If the i^{th} row and j^{th} column of the designed map is a barrier
            2.2.2.1.1 Fill 1 into the matrix entry.
         2.2.2.2 If the i^{th} row and j^{th} column of the designed map is a clear vertex
            2.2.2.2.1 Fill 0 into the matrix entry.
      2.2.3 The system creates an empty .csv file in the player's current directory to store the matrix.
3. The system displays the maze map on the user interface.
   3.1 The system displays a square-shaped grid with 30 * 30 PX-Squares.
   3.2 For each PX-Square at the i^{th} row and j^{th} column of the grid
      3.2.1 If the i^{th} row and j^{th} column of the designed map is a barrier
         3.2.1.1 The system changes the color of the PX-Square to dark grey.
      3.2.2 If the i^{th} row and j^{th} column of the designed map is a clear vertex
         3.2.2.1 The system changes the color of the PX-Square to white.
4. The use case ends

Postconditions

A. The system displays a maze map of 30*30 PX-Squares on the user-interface subject to constraints in *Assumptions*.

B. A .csv file that stores the matrix representing the maze map is generated.

Use Case: Generates Shortest Path

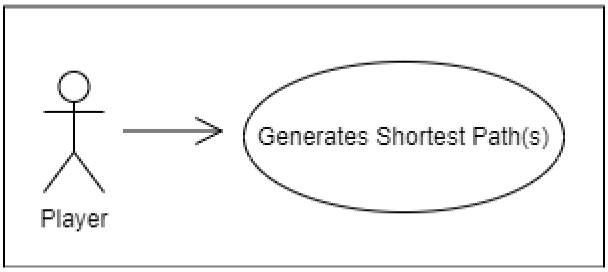
Brief Description

This use case describes the display of one of the shortest Paths from the Entry-Point to the Exit-Point.

Actor

Player

Use-case Diagram



Assumptions

A. The length of a Path is defined as the number of clear vertices that the Path is composed of.

Preconditions

A. The system displays a maze map of 30*30 PX-Squares on the user-interface.
B. The maze map has one Entry-Point located on the left-side border.
C. The maze map has one Exit-Point located on the right-side border.
D. There are at least 2 Paths connecting the Entry-Point and the Exit-Point.

Basic Flow

1. The use case begins after the player actor creates the maze map.
2. The system highlights one of the shortest Paths from the start position to the exit position on the user interface.
3. The system stores the vertices passed along the displayed Path in a .csv file.
4. The use case ends.

Detail Flow

1. The use case begins after the creation and display of a maze map.
2. The system highlights one of the shortest Paths from the Entry-Point to the Exit-Point on the user interface.
    - 2.1. The system retrieves the Entry-Point and Exit-Point.
    - 2.2. The system finds at least 2 shortest Paths from the Entry-Point to the Exit-Point.

    2.3. The system randomly selects one of the shortest Paths found and records the vertices along it.
    2.4. The system changes the color of the PX-Squares underlying the clear vertices along the recorded shortest Path to green on the user interface.
3. The system stores the vertices passed along the displayed Path in a .csv file.
    3.1. Line 1 contains the text “s1”.
    3.2. Each line after line 1 contains one vertex location, which row and column are separated by a comma (,), e.g. “15,9”
        3.2.1. Line 2 represents the starting position, line 3 represents the next vertex along the shortest Path after the start position, etc., until reaching the last line, representing the exit position.
    3.3. The system exports the file
4. The use case ends.

Postconditions

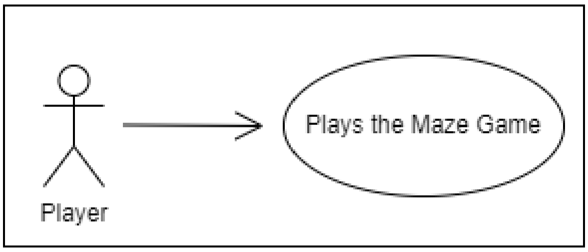
A. One of the shortest Paths between the Entry-Point and Exit-Point is highlighted on the user interface by having the compositing PX-Squares turned to green.

B. A .csv file that includes the coordinates of the compositing clear vertices of the shortest Path highlighted is stored in the user’s directory.

Use Case: Plays the Maze Game

Brief Description
This use case describes the starting conditions, gameplay and ending conditions of the game.

Actor
Player

Use-case Diagram


Preconditions
A. The system displays a maze map of 30*30 PX-Squares on the user-interface.
B. The maze map has one Entry-Point located on the left-side border.
C. The maze map has one Exit-Point located on the right-side border.
D. There are at least 2 Paths connecting the Entry-Point and the Exit-Point.
E. One of the shortest Paths between the Entry-Point and Exit-Point is highlighted on the user interface by having the compositing PX-Squares turned to green.

Assumptions
A. The roles controlled by the player and system are fixed.
   a. The system only controls Tom.
   b. The player only controls Jerry.
   c. There are no other movable objects in the game.
B. The game is composed of multiple rounds.
   a. In each round, only one of Tom or Jerry is given the chance to move.
   b. Tom and Jerry take turns to have their rounds to move
   c. The first round always belongs to Jerry, i.e. the player makes the first move.
   d. Both Tom and Jerry must move to a different clear vertex in their respective rounds.
   e. When it is Tom's round, Tom always approaches Jerry along the shortest Path from its current position to Jerry's.
C. The starting positions of Tom and Jerry are fixed.
   a. Tom always starts at the Exit-Point.
   b. Jerry always starts at the Entry-Point.
D. The game ends if and only either the player or the system wins.

    a. The player wins if Jerry reaches the Exit-Point.
    b. The system wins if Tom catches Jerry, defined by either of the following scenarios:
        i. Tom moves across or to the location of Jerry in Tom's round
        ii. Jerry moves across or to the location of Tom in Jerry's round
        iii. Tom and Jerry's location overlaps at the end of either's round
E. The speed is measured in (number of vertices allowed to travel/round.)
    a. The speed of Jerry is pre-determined and fixed throughout the game.
    b. The speed of Tom is always faster than that of Jerry.

Basic Flow

1. The use case begins when the player actor enters the first round.
2. The system initializes the game.
    2.1. The system removes the highlighted colors of the PX-Squares compositing the shortest Path shown on the user interface.
    2.2. The system places the figures of Tom and Jerry at the Exit-Point and Entry-Point respectively.
3. While Not (player wins OR system wins)
    3.1. If this round is Jerry's turn to move
        3.1.1. Player controls Jerry to move to a different vertex along a Path
    3.2. If this round is Tom's turn to move
        3.2.1. The system controls Tom to move to a different vertex along a Path.
    3.3. The system determines whether player wins or system wins
4. The system notifies the result of the game and exits
5. The use case ends.

Detail Flow

1. The use case begins when the player actor enters the first round.
2. The system initializes the game.
    2.1. The system removes the highlighted colors PX-Squares compositing the shortest Path shown on the user interface.
    2.2. The system places the figures of Tom and Jerry at the Exit-Point and Entry-Point respectively.
3. While Not (player wins OR system wins)
    3.1. If this round is Jerry's turn to move
        3.1.1. While no. of blocks traveled by player in this round <= speed of Jerry
            3.1.1.1. Player presses one of "W", "A", "S", "D" on keyboard to control Jerry to move to either the upper, left, down, and right neighbor vertices of the current vertex respectively
            3.1.1.2. If the target neighbor vertex indicated by player is a clear vertex
                3.1.1.2.1. The system shows Jerry at the target neighbor vertex on the user interface
                3.1.1.2.2. The system increments no. of blocks traveled in this round by player by 1
            3.1.1.3. If the target neighbor vertex indicated by player is a barrier

3.1.1.3.1. The system shows Jerry remaining at current position on the user interface

3.1.1.3.2. The system does not record change in no. of blocks traveled in this round by player

3.2. If this round is Tom's turn to move

3.2.1. The system finds the shortest Path between Tom's current location and Jerry's current location

3.2.2. The system calculates the new position of Tom after moving Tom by speed of Tom vertices along the Path found in 2.2.1

3.2.3. The system shows Tom at the new position on the user interface

3.3. The system determines whether player wins or system wins

4. The system notifies the result of the game and exits

4.1. If player wins

4.1.1. The system outputs "Congratulations! You won" on user interface

4.2. If system wins

4.2.1. The system outputs "What a pity! Tom caught you" on user interface

4.3. The system exits.

5. The use case ends.