Main BFS Algorithm

```
FUNCTION find_shortest_path(maze, start, end):
    INITIALIZE queue WITH start
    INITIALIZE visited SET WITH start
    INITIALIZE parent MAP WITH start -> NULL

    WHILE queue IS NOT EMPTY:
        current = DEQUEUE from queue

        IF current == end:
            RETURN reconstruct_path(parent, end)

        FOR EACH direction IN [up, down, left, right]:
            neighbor = current + direction

            IF neighbor IS valid position AND
               neighbor IS NOT wall AND
               neighbor NOT IN visited:

                ENQUEUE neighbor TO queue
                ADD neighbor TO visited
                SET parent[neighbor] = current

    RETURN NO_PATH_FOUND

FUNCTION reconstruct_path(parent, end):
    INITIALIZE path AS empty list
    current = end

    WHILE current IS NOT NULL:
        ADD current TO path
        current = parent[current]

    RETURN REVERSED path
```

Game Implementation Pseudocode

```
CLASS BFSMazeExplorer:
  METHOD __init__():
    maze = CREATE_15x15_MAZE()
    start_position = (1, 1)
    end_position = (13, 13)
    current_position = start_position
    bfs_queue = EMPTY_DEQUE
    bfs_visited = EMPTY_SET
    bfs_parent = EMPTY_DICTIONARY

  METHOD start_bfs():
    CLEAR bfs_queue, bfs_visited, bfs_parent
    ENQUEUE start_position TO bfs_queue
    ADD start_position TO bfs_visited
    SET bfs_parent[start_position] = NULL
    DRAW_MAZE()

  METHOD bfs_step():
    IF bfs_queue IS EMPTY:
      RETURN

    current = DEQUEUE from bfs_queue

    IF current == end_position:
      highlight_shortest_path(current)
      RETURN

    FOR EACH direction IN [(-1,0), (1,0), (0,-1), (0,1)]:
      new_row = current.row + direction.row
      new_col = current.col + direction.col
      new_pos = (new_row, new_col)

      IF new_pos IS valid AND maze[new_row][new_col] IS NOT WALL AND new_pos NOT IN
bfs_visited:
        ENQUEUE new_pos TO bfs_queue
        ADD new_pos TO bfs_visited
        SET bfs_parent[new_pos] = current

    DRAW_MAZE()

  METHOD highlight_shortest_path(end):
    path = EMPTY_LIST
```

```
current = end

WHILE current IS NOT NULL:
    ADD current TO path
    current = bfs_parent[current]

shortest_path = REVERSED path
DISPLAY shortest_path ON MAZE
```