Use Cases for 2D Arcade Style Game

Use Case 1: Collecting Coins

Actors:

Players

Goals:

The player successfully collects all coins on the map to unlock the door and progress to the next level.

Trigger:

The player starts a level and begins moving toward objectives.

Basic Flow:

1. The player moves towards the coin on the map.
2. The player steps on the objective's tile.
3. The game increments the coin count and removes the coin from the map.
4. If all coins are collected, the door tile opens for level progression.
5. The player moves to the door to finish the level.

Postconditions:

Player progresses to the next level or every time when the user enters the game.

Exceptions:

1. Do Not Allow Invalid Actions: If the player steps on an invalid tile (e.g., a wall or obstacle), they do not collect the coin. The game should not allow the player to step on the invalid tile.
2. Collision with Enemy: If the player encounters a moving or stationary enemy while attempting to collect a coin, they lose a life, and the level soft-resets. The collected objective remains available until it is reached again.
3. Field of View Blocked: The player may be unable to see all objectives due to limited field of view (in harder difficulty settings), potentially delaying objective collection. The game should still allow movement and coin collection if the player navigates correctly.

Use Case 2: Avoiding Enemies

Actors:

Players

Goals:

The player successfully avoids moving enemies by navigating the map strategically, maintaining their lives and continuing progress without being caught. This will be the main challenge to play the game while requiring the users to take the coin.

Trigger:

A new level begins, or the player makes a move while enemies are active.

Basic Flow:

1. The player moves by pressing a directional key.
2. The enemies move one cell closer to the player at each tick.
3. The player avoids being caught by enemies by moving strategically.
4. If caught, the player loses all the given lives (given 3 lives), and the level soft-resets.
5. If it is caught after arriving at the door to the end, the player does not lose the life.

Postconditions:

The player loses a life and must restart the level if there is no life left. If the player avoids all enemies, they progress in the game.

Exceptions:

1. No Available Moves: If the player is surrounded by enemies or obstacles and cannot move, the player will inevitably lose a life. The game should trigger a life deduction, soft-reset the level, and notify the player.
2. Enemy Pathing Error: If an enemy cannot correctly calculate a path toward the player (e.g., if blocked by walls), the enemy should remain in its current position until a path becomes available.
3. Invalid Enemy Movement: In case an enemy “glitches” and moves into an inaccessible tile (e.g., through a barrier), the game should force a reset of the enemy’s position to a valid tile or prevent movement entirely.

Use Case 3: Scoring and Life Mechanics

Actors:

Players

Goal:

The player completes a sequence of levels and challenges themselves leading to motivation of playing the game.

Trigger:

The player starts the game and begins moving, which activates the scoring system.

Basic Flow:

1. The player starts the game with 3 lives and a score of 0.
2. Objectives collected and lives remaining are tracked and added to the score.
3. The score updates based on the number of ticks (number of movements) taken and remaining lives in order to make the game challenging.
4. At the end of the game (either winning or losing), the final score is displayed.

Postconditions:

The player's score is displayed and logged after each level and at the end of the game.

Exceptions:

1. Negative Score Issue: If penalties or lost lives drive the player's score below zero, the game should trigger a game-over sequence, and not allow further progression with a negative score.

Use Case 4: Campaign Mode

Actors:

Players

Goal:

There are sequence of levels from beginner to hard mode challenging the players to feel the achievement of progressing through the game.

Trigger:

The player selects "Campaign" mode from the main menu.

Basic Flow:

1. The player chooses the campaign option.
2. They select a difficulty level (easy, normal, or hard).
3. The first level loads, and the player progresses through levels sequentially.

Postconditions:

The player completes levels in sequence and progresses through the campaign.

Exceptions:

1. Game Over in Campaign: If the player loses all lives (given 3 lives), the game should reset to level 1 rather than crash or stop progressing.
2. Difficulty Change: If the player tries to change difficulty mid-campaign, the system should block this action or allow only for restarts.

Use Case 5: Practice Mode

Actors:

Players

Goal:

The player can practice specific levels without the risk of losing lives which leads them to improve their skills.

Trigger:

The player selects “Practice” mode from the main menu.

Basic Flow:

1. The player chooses “Practice” mode.
2. They select a difficulty level.
3. They choose any previously completed level to practice.
4. The player repeats the level without losing lives. There are no lives in the game. So whenever the player dies, he/she can repeat the game.

Postconditions:

The player practices levels without penalty.

Exceptions:

1. No Levels Unlocked: If the player attempts to enter practice mode before completing any levels, the system should notify them that no levels are available to practice.
2. Failed to Load Practice Level: If a practice level does not load properly, the system should return the player to the menu with a clear error message.

3. Infinite Loop in Practice: If the player repeatedly fails in practice mode (e.g., repeatedly gets caught by enemies), the game should allow the player to restart or exit without locking them in a failure loop.

Use Case 6: Bonus Point System

Actors:

Player

Goal:

Collect bonus objectives to increase their final score giving more reward with challenges

Preconditions:

Bonus objectives appear after a certain time or randomly in the level. The player's initial field of view is limited. However, allow the player to increase their field of view as they collect bonus points, making it easier to navigate and avoid enemies.

Basic Flow:

1. As the player progresses through a level, a bonus objective appears in an area off the main path.
2. The player chooses to deviate from the main objectives to collect the bonus points. Gaining increased field of view.
3. The player can use the expanded view to spot enemies and objectives from farther away, reducing the chance of losing lives.
4. The player navigates toward the bonus points, avoiding enemies.
5. If the player collects the bonus points, they earn additional points.
6. Bonus points must be collected before completing all main objectives, or they disappear.

Postconditions:

The player earns bonus points that contribute to the final score.

Exceptions:

1. Bonus Objective Disappears: If the player takes too long to reach the bonus objective, it should disappear after a set number of ticks (number of movements).
2. Bonus Objective Conflict: If the player completes the last main objective before collecting the bonus, the bonus disappears. The system should ensure the bonus is no longer collectable after that point.
3. Bonus Objective After Death: If the player loses all lives before collecting sufficient points, the FOV does not expand.