Use Cases

Use Case - tick forward

Use case: update the game with each tick
Primary actor: Player
Goal in context: update the game, letting the player and the enemies move and update the locations.
Preconditions: the program runs and the game starts.
Trigger: as the game starts.
Scenario:

1. the game starts
2. ticks are updated every so often
3. events happen according to the tick.

Exceptions:

➔ Every tic, the program will check for any updates.
➔ All moving characters should be able to move and update the display on map.

Priority: Essential
When available: Initial implementation
Frequency of use: Frequent

Use Case - Controls

Use case: control the main character's movement
Primary actor: Player
Goal in context: Manage the character's action by controlling using the keyboard.
Preconditions:

- "The Twins" was launched successfully.
- Program loaded and started running properly.
- Player has clicked a key for which direction he or she wants the character to go towards

Trigger: When the player presses either one of direction key on their keyboard.
Scenario:

1. The player starts the game – see use-case: "Game start."
2. The player presses a direction's key that he or she wants the character to walk towards.
3. For every tick, the character moves 1 cell towards the direction that the player pressed.
4. The player let go of that key.
5. The character stops moving.

Exceptions:

➔ Keys other than up, down, left, right will not consider a control of movement to the character, and so the character will not move.
➔ When hitting the barriers or walls, the character should not be able to pass through.
➔ If the character meets an enemy or rewards, the scoreboard will calculate the points correctly.

Priority: Essential.
When available: Initial implementation
Frequency of use: Frequent, on each tick.

Use Case - Scoreboard

Use case: Keep track of the player's score
Primary actor: Player
Goal in context: Calculate the score whenever the player triggers any enemies or rewards.
Preconditions:
Trigger:

- After a game starts, whenever the player collides with reward or punishment

Scenario:

1. The player starts a new game – see use-case: "Game start."
2. The program displays initial scores in the corner of the board.
3. The player walks into any type of reward, the score should increase.
4. The player walks into the trap, the score should decrease.
5. The player walks into the moving enemy, the score should become 0 and end the game

Exceptions:

➔ Once the game started, all score shown in scoreboard will be initialized properly
➔ When the player hits any enemy, the score shown in the scoreboard should decrease by the damage that was caused by the enemy.
➔ When the player collects any reward, the score shown in the scoreboard should increase by the amount of that reward's cost.
➔ If the player did not collect rewards or run into any enemy, the score should not change.

Priority: Moderate priority
When available: Second implementation
Frequency of use: Frequent.

Use Case - Game start

Use case: Let player start the game
Primary actor: Player
Goal in context: To call and start a new game of "The Twins" whenever the player clicks on the start game button.
Preconditions:

- "The Twins" was launched successfully.
- Program loaded and started running properly.
- The player clicks on the start button to start a new game.

Trigger: After the game starts running, the player clicks the start button in the main menu.
Scenario:

1. The player launches "The Twins" and starts the program.
2. The program start running and shows starting scene and main menu
3. The player clicks on the start game button.
4. The program starts a new game and creates a new maze-like map.
5. The map got all barriers, walls, enemies, and rewards set properly.
6. The board got a scoreboard showing in the corner. – see use-case: "Scoreboard."

Exceptions:

➔ If the player did not click on the start button, the program will not start a new game
➔ Once the player clicks on the start button, the program will have all game components ready on the board.

Priority: Moderate priority
When available: Second implementation
Frequency of use: Infrequent.

Use Case - Pause game

Use case: Allowed player to pause the game, and resume whenever he or she wants.
Primary actor: Player
Goal in context: Stop the current game and wait for the player's next action.
Preconditions:

- “The Twins” was launched successfully.
- Program loaded and started running properly.

Trigger: After a game starts, the player clicks on the pause game button during the game.
Scenario:

1. The player starts a new game – see use-case: “Game start.”
2. The player clicks on the pause button during the game.
3. The program displays a pause menu and waits for the player to proceed to the next step.

Exceptions:

- ➔ If the player did not click on the pause button, the game would not pause.
- ➔ Once the player clicks on the pause button, the program will display a pause scene and all moving characters stop moving.
- ➔ If the player clicks on the start a new game button, the program should end the current game and start a new map.
- ➔ If the player clicks on the resume button, the player should be able to control the character, and all moving enemies should resume moving.
- ➔ If the player clicks on the back to home page button, the program should end the current game and display the home page.

Priority: Moderate priority
When available: Third implementation
Frequency of use: Infrequent

Use Case - End/Quit Game

Use case: End the current game.
Primary actor: Player
Goal in context: End the current game and main menu page.
Preconditions:

- “The Twins” was launched successfully.
- Program loaded and started running properly.
- Pause button was clicked.

Trigger: The player clicks the exit button from the pause menu.
Scenario:

1. The player starts a new game – see use-case: “Game start.”
2. The player clicks on the pause button during the game. – see use-case: “Pause game.”
3. The player clicks on the exit button.
4. The program ends the current game and displays the main menu.

Exceptions:

- ➔ When the player clicked the pause button but not the exit button, the program should stay at the pause scene.
- ➔ If hte player clicked exit button in the pause menu, the program should stop the current gama and display home page.

Priority: Moderate priority
When available: Third implementation
Frequency of use: Infrequent

Use Case - Winning

Use case: The player win the game
Primary actor: Player
Goal in context: Win the game and display the winning scene with the score that player got.
Preconditions: The game has started and reached the winning condition.
Trigger: The player must have enough coins to be able to collect the key/keys, once the key/keys are collected the player must reach the door to trigger the win.
Scenario:

1. The player starts the game – see use-case: “Game start.”
2. The player control the character to collect rewards while avoiding the enemies. – see use-case: “Controls.”
3. The program display that the score of the character increases every time as the player collects a reward
4. The player collects enough coins to be able to pick up key/keys needed to open the door.
5. The exit door opens
6. The player walks exits the map through the door
7. The program stops the game and displays the winning scene which shows the score that player got for this game.

Exceptions:

- ➔ if not all keys are collected, the door remains closed and will act as a barrier/wall.
- ➔ If the player hasn't collected enough coins, the player will not be able to collect the keys.

Priority: Moderate priority
When available: Second implementation
Frequency of use: Infrequent

Use Case - Losing

Use case: the player loses the game
Primary actor: Player
Goal in context: end the game and display a losing screen.
Preconditions: the game has started
Trigger: the player has encountered a moving enemy OR score dropped to 0 due to colliding with traps
Scenario:

1. the game is running
2. The score drops to 0, or the player collides with a moving enemy.
3. The game ends and the losing screen is played

Exceptions:

- ➔ The player collides with a moving enemy, the game stops and displays the losing screen.
- ➔ The player didn’t collide with a moving enemy, the game should keep going.
- ➔ The losing screen should display a

Priority: Moderate priority
When available: Third implementation
Frequency of use: at most once per game.