Use case: Playable character gameplay

Primary Actor: Player/User

Goal in context: Move character within the boundaries of the map and interact with the objects on it.

Preconditions: Playable character move functions have been implemented.

Trigger: W, A, S, D or arrow keys pressed.

Scenario:

1. There are no collisions with barriers.
2. There is a collision with an enemy entity.
3. There is a collision with a reward.
4. Valid movement key pressed.

Exceptions:

1. **Barrier**: The playable character cannot move in that direction and remains in place, 1 “tick” has passed.
2. **Moving Enemy**: The playable character loses a life/health decreases and score decreases.
3. **Punishment**: The game is over and the player loses.
4. **Invalid key**: The playable character does not move, 1 “tick” has passed.

Priority: Essential, must be implemented.

When available: On level load.

Frequency of use: Frequent, whenever W, A, S, D or arrow keys are pressed.

Channel to actor: Keyboard inputs

Open issues:

1. What to do when a movement key is held?
2. How will collisions between objects be detected and what will happen to these objects?

Use case: Menu buttons

Primary Actor: Player/User

Goal in context: Change to corresponding screen/change an option on clicking a button.

Preconditions: Screen and button classes have been implemented.

Trigger: User clicks a button.

Scenario:

1. Player clicks the “Start” button.
2. Player clicks the “Exit” button.
3. Player clicks the “Settings” button.
4. Player clicks the “Restart” button.
5. Player clicks the “Credit” button.
6. Player clicks the “Main menu” button.
7. Player clicks the “Continue” button.

Exceptions:

1. **Start**: Screen changes to Stage 1.
2. **Exit**: Game closes.
3. **Settings**: Settings screen is displayed.
4. **Restart**: Screen changes back to Stage 1.
5. **Credit**: Credits are displayed.
6. **Main menu**: Main menu is displayed.
7. **Continue**: Game resumes, exits out of pause menu.

Priority: Essential.

When available: On game launch.

Frequency of use: Whenever the player/user clicks a button

Channel to actor: Mouse

Open issues: How will mouse clicks be registered as inputs?

Use case: Exit stage cell

Primary Actor: Player

Goal in context: Exit/Progress to next stage.

Preconditions: Cell class and movement functions implemented.

Trigger: All regular rewards collected and the player moves to the exit cell.

Scenario:

1. Player is not currently on the last stage.
2. Player is currently on the last stage.

Exceptions:

1. **Final stage**: Game Over screen displays with player score and return to the Main menu and play again button.
2. **Not final stage**: Player progresses to next stage(map) with lives/health restored.

Priority: Essential.

When available: Once the player has collected all regular rewards.

Frequency of use: Once per stage.

Channel to actor: Screen.

Open issues:

1. How will we determine if all regular rewards have been collected?
2. How will we detect the collision between the player and the exit cell?

Use case: Collect reward (Bonus/Regular)

Primary Actor: Player

Goal in context: Increase score and/or restore health and unlock exit cell by collecting them all.

Preconditions: Player class, movement class and reward class implemented.

Trigger: Player walks over a cell which contains a reward.

Scenario:

1. Player walks over a cell containing a regular reward.
2. Player walks over a cell containing a bonus reward.

Exceptions:

1. Player moves to a cell containing a reward but is occupied by a moving enemy.
2. Moving enemy walks over a cell containing a reward.

Priority: Essential to progress through stages and scoring.

When available: On game start.

Frequency of use: When the player enters a cell containing a reward. Occurs a set amount of times per stage.

Channel to actor: Screen.

Open issues:

1. How will we detect collisions between player and rewards?
2. How will we detect collisions?

Use case: Player death

Primary Actor: Player

Goal in context: Player dies and results in a game over.

Preconditions: Player class, enemy class, and map must be implemented and instantiated.

Trigger: Enemy catches up with player, player collides with a moving enemy, player's score becomes negative.

Scenario:

1. Player moves to a cell containing a moving enemy.
2. Moving enemy catches up with the player.
3. Player score becomes negative(i.e. Health becomes 0).

Exceptions:

1. Player collects a reward.
2. Player moves to the exit cell.
3. Player collides with a punishment but still has a positive score.

Priority: Moderate priority, required to end the game not required to play the game.

When available: On game start.

Frequency of use: Once per game.

Channel to actor: Screen.

Open issues:

1. How will we detect when the player reaches a negative score?

Use case: Negative scoring(collision with punishments)

Primary Actor: Player

Goal in context: Player score decreases.

Preconditions: Player class, punishment class implemented and instantied.

Trigger: Player enters a cell containing a punishment.

Scenario:

1. Player has a score greater than zero and collides with a punishment.

Exceptions:

1. Colliding with a punishment that will cause the score to become negative will instead cause a game over.

Priority: Low priority, not essential to playing the game or ending it.

When available: On game start.

Frequency of use: Every time the player collides with a punishment.

Channel to actor: Screen.

Open issues:

1. How do we detect collisions between the player and punishments.