Use Cases

Main Program

Use Case: *OpenProgram*

Primary actor: Player

Goal in context: To run the game's Java program and reach the main menu

Precondition: Java code is compiled and executable file is ready

Trigger: The player decides to play the game

Scenario:

1. Player: turns on computer
2. Player: navigates to the game file
3. Player: double clicks the executable

Exceptions: n/a

Priority: Essential, game must be able to be run

When available: All times

Frequency of use: Any time the player wishes to play the game

Channel to actor: Via mouse

Secondary actors: n/a

Channels to secondary actors: n/a

Main Menu

Use Case: *StartGame*

Primary actor: Player

Goal in context: Transition from main menu to the game screen

Preconditions: O*penProgram* use case

Trigger: The player decides to begin a round

Scenario:

1. Player: clicks start game
2. MenuLogic.startGame() method is called to switch the game menu
3. Screen.GameScreen() method is called to display the cave maze on the screen

Exceptions: n/a

Priority: Essential, round must be able to start for the player to be able to play the game

When available: On main menu

Frequency of use: Any time the player wishes to start a round

Channel to actor: Via mouse

Secondary actors: n/a

Channels to support secondary actors: n/a

Open issues: n/a

Use Case: *ExitGame*

Primary actor: Player

Goal in context: Terminate the program

Preconditions: *OpenProgram* use case

Trigger: The player wishes to exit the game

Scenario:

1. Player: clicks the exit button from the main/game over menu OR the x-button in the top right of the screen
2. GameRunner.gameEnd() is called to terminate the game loop
3. MenuLogic.exitProgram() is called to terminate the program

Exceptions:

1. The game crashes for whatever reason: the executable stops running
2. The user externally terminates the Java file: the executable stops running

Priority: Essential, player must be able to stop playing

When available: Exit button only on main/game over menu, x-button from anywhere

Frequency of use: Any time the player wishes to stop playing

Channel to actor: Via mouse

Secondary actors: n/a

Channels to support secondary actors: n/a

Open issues: n/a

Gameplay

Use Case: *RestartAfterGameOver*

Primary actor: Player

Goal in context: The game restarts with all entities reset

Preconditions: At least one round played, ended via game mechanics

Trigger: The player either touches an enemy or reaches negative score, then chooses to restart

Scenario:

1. Player: achieves game over
2. Screen.GameOverScreen() is called to show player options
3. Player: selects restart
4. game loop restarts

Exceptions: n/a

Priority: Essential, restart must work in case player wants to try again

When available: On game over menu

Frequency of use: When the player wishes to try again

Channel to actor: Via mouse

Secondary actors: n/a

Channels to support secondary actors: n/a

Open issues:

1. How to restart game loop and make sure all game mechanics are how they were at the start?

Use Case: *MainCharacterMovement*

Primary actor: Player

Goal in context: Allow player to control the main character's movement

Preconditions: Game loop running

Trigger: Player clicks one of: WASD

Scenario:

1. Player: inputs key (W, A, S, or D)
2. GameLogic.mainCharacterMovement() is called
3. MainCharacter.takeUserInput(int keyPressed) is called to process what the user pressed on
4. MainCharacter.updatePosition() is called, moving the main character onto a new tile depending on which key was pressed
5. Player: moves onto tile if the tile is empty
6. Player: collides with another object if tile is not empty (see corresponding use cases)

Exceptions:

1. Not on game menu: MainCharacter cannot take user input

Priority: Essential, character must be able to move

When available: On game menu

Frequency of use: Frequent, player is usually in movement

Channel to actor: Via WASD keys

Secondary actors: n/a

Channels to support secondary actors: n/a

Open issues:

1. Making sure collision detection works as intended
2. Should user input be handled in GameLogic instead of MainCharacter?

Use Case: *CollectReward*

Primary actor: Player

Goal in context: Perform corresponding reward effect when reward collected

Preconditions: Game loop is running, main character is adjacent to reward

Trigger: Collision detected between main character and reward

Scenario:

1. Player: moves main character onto tile with a reward
2. CollisionDetecter.detectCollision() is called between the two objects
3. GameLogic.adjustScore() is called to increase MainCharacter.points by correct amount depending on whether reward is regular or bonus
4. Reward: disappears from screen
5. If bonus reward, MainCharacter.setInvincibility() is called

Exceptions: n/a

Priority: Essential, rewards are necessary for scoring

When available: On tiles with rewards

Frequency of use: Number of rewards on map

Channel to actor: Via keyboard

Secondary actors: n/a

Channels to support secondary actors: n/a

Open issues:

1. How to make the reward disappear once collected?

Use Case: *TouchPunishment*

Primary actor: Player

Goal in context: Take damage when punishment is touched

Preconditions: Game loop is running, main character is adjacent to punishment

Trigger: Collision detected between main character and punishment

Scenario:

1. Player: moves main character onto tile with a punishment
2. CollisionDetecter.detectCollision() is called between the two objects
3. GameLogic.adjustScore() is called to decrease MainCharacter.points by some amount
4. If MainCharacter.points is negative, see *GameOver* use case

Exceptions: n/a

Priority: Essential, punishments part of requirements

When available: On tiles with punishments

Frequency of use: Depends on how often the player accidentally touches punishments

Channel to actor: Via keyboard

Secondary actors: n/a

Channels to support secondary actors: n/a

Open issues: n/a

1. **Use Case:** *TouchEnemy*

Primary actor: Player

Goal in context: Perform correct effect when player touches an enemy

Preconditions: Game loop is running, enemy is adjacent to main character

Trigger: Collision detected between main character and enemy

Scenario:

1. Player: moves main character onto the tile with an enemy OR Enemy: moves onto the player's tile
2. If MainCharacter is invincible, Enemy: disappears from screen
3. If MainCharacter is not invincible, see G*ameOver* use case

Exceptions: n/a

Priority: Essential, enemies are important to the stakes of the game

When available: On tiles with enemies

Frequency of use: Depends on how often player accidentally touches enemies

Channel to actor: Via keyboard

Secondary actors: n/a

Channels to support secondary actors: n/a

Open issues:

1. How to make enemy disappear if player invincible?
2. CollisionDetecter.detectCollision() is called between the two objects
3. GameLogic.adjustScore() is called to increase MainCharacter.points by correct amount depending on whether reward is regular or bonus
4. Reward: disappears from screen
5. If bonus reward, MainCharacter.setInvisible() is called

Preconditions: Game loop is running, main character is adjacent to punishment

Trigger: Collision detected between main character and punishment

Use Case: *GameOver*

Primary actor: Player

Goal in context: End the game via game mechanics

Preconditions: Game is running

Trigger: *TouchPunishment* OR *TouchEnemy* with qualifying conditions

Scenario:

1. Screen.GameOverScreen() is called which displays options to player: restart or exit (as well as their score)
2. If Player: presses restart, see *RestartAfterGameOver* use case
3. If Player: presses exit, see *ExitGame* use case

Exceptions: n/a

Priority: Essential, game over is a requirement

When available: On tiles with enemies/punishments

Frequency of use: Once per round

Channel to actor: Via mouse and keyboard

Secondary actors: n/a

Channels to support secondary actors: n/a

Open issues: n/a

Use Case: *WinGame*

Primary actor: Player

Goal in context: Show winning logic

Preconditions: Player achieves a high enough score to reveal cave exit

Trigger: Player reaches exit

Scenario:

1. Player: moves MainCharacter onto exit tile
2. MenuLogic.WinningScreen() is called

Exceptions: n/a

Priority: Essential, player needs a way to win

When available: On exit tile

Frequency of use: Once per game

Channel to actor: Keyboard

Secondary actors: n/a

Channels to support secondary actors: n/a

Open issues:

1. What should the player be able to do from the winning screen?
    a. Restart game? Go back to main menu? Exit game?