COMP361D1
Use Cases

Alek Bedard, Nuri Amiraslan, Francis Piché
Teymur Azim-zada, Seng Chiat Haw, Abhijay Gupta

October 2018

Play Flash Point

Use Case: Play Flash Point
Scope: Flash Point
Level: User Goal
Intention in context: The intention of *Player* is to play a game of Flash Point with other Players
Multiplicity: Multiple *Players* can play together at the same time. There must be at least 3 *Players* with the maximum being 6.
Primary actor: *Player*
Secondary actor(s): Other Players
Main Success Scenario:

1. Player Log In to the *System.*
2. *System* prompts the *Player* to choose between Host Game or Join Game.
3. *Player* informs the *System* of their choice.
4. *System* prompts the *player* to select a Character.
5. *Player* informs the *System* of their choice.
6. *Host* informs the *System* to start the game.
7. *System* prompts the *Host* to choose a preset or generate a random *Game Board.*
8. *Player* informs *system* of his starting position

   System waits until enough Players are connected to continue to next step
9. *Players* agree over Chat who goes first.
10. *Player* votes to inform system who should play first.

    Step 11 is executed until the game is over, or until Players quit.
11. *Players* take turns one after another.
12. *System* informs the *Players* whether they won the game.

Extensions:

- 1a. *Player* was not able to log in with the right credentials. Use case continues at step 1.
- 3a. The player chose Host Game.
  - 3a.1. Host Game
- 3b. The Player chose Join Game.
  - 3b.1a. Join Game
- 3c. *Player* was not able to create or join a game. Use Case continues at step 2.
- 6a. *System* detects that there are not enough *Players* connected. Use Case continues at step 6.
- 8a. *Player's* Starting position was not legal. Use Case continues at step 8.
- 10a. *Player* decides to quit game before game is over.
  - 10a.1. *System* Prompts *Player* whether they want to save game
  - 10a.2. *System* informs *Player* of the IP address stored for this game.
  - 10a.3. Game is saved in *Database.* End of Use Case.

Player Login

Use Case: Player Login
Scope: Flash Point
Level: Subfunction
Intention in Context The intention of *Player* is to log in to the main *system* server.
Multiplicity: Many *Players* can log in to the game server. Each *Player* can only log in once per session.
Primary actor: *Player*
Secondary actors: *Database*
Main Success Scenario

1. *Player* provides credentials to the *System*
2. *System* searches *database* to find *Player's* credentials
3. *System* informs *Player* the login was successful

Extensions:

2a. *System* fails to find *Player's* credentials within *database*. Use case continues at step 1.

Join Game

Use Case: Join Game
Scope: Flash Point
Level: Subfunction
Intention in Context The intention of *Player* is to join a game that has already been created by someone else.
Multiplicity: Many *Players* can join an existing game.
Primary actor: Player.
Main Success Scenario

1.*System* displays a list of existing games that *Player* could potentially join
2.*Player* chooses the available game of his choice.

Extensions:

2a. *System* fails to connect *Player* to the game board. Player is redirected to step 1

Host Game

Use Case: Host Game
Scope: Flash Point
Level: Subfunction
Intention in Context: The intention of *Player* is to Host a game.
Multiplicity: Many *Players* can host a game
Primary actor: *Player*
Main Success Scenario:

1. *System* prompts *Player* whether he/she wants to Host Existing Game or Create Game

Create Game

Use Case: Create Game
Scope: Flash Point
Level: Subfunction
Intention in Context The intention of *Player* is to create a new game.
Multiplicity: Multiple instances of the *Game* can be created within the *network*.

Primary actor: *Player*
Secondary actor(s): *System*
Main Success Scenario

1. *System* prompts *Player* if Host Existing Game instead
2. *System* displays an input menu to *Player* for game settings
3. *Player* informs the *System* whether the game should be private or public
4. *Player* informs the *System* which Game Mode they would like to play.
   Game mode can either be Family, Beginner. Veteran, Heroic
5. *Player* specifies number of *Players*
6. *Player* specifies to *System* to play on a preset Classic Board or Generate Random GameBoard
7. *System* informs the *Player* the creation of the game was successful

Extensions:

1a. *Player* selected Host Existing Game. End of Use Case.
7a. *System* was not able to create the desired game. Use Case continues to step 1.

Host Existing Game

Use Case: Host Existing Game
Scope: Flash Point
Level: Subfunction
Intention in Context: The intention of *Player* is to host a pre-existing game within the *database.*
Multiplicity: Many *Players* can host pre-existing games simultaneously.
Primary actor: *Player*
Secondary Actor: *Database*
Main Success Scenario:

1. The *Player* informs the *System* which game they would like to join.
2. *System* requests the *Database* for this game
3. *System* displays previously created game board to *Player*
4. *Player* confirms the game board to *System.*
   System waits until enough Players have joined the game.

Extensions:

2a. *Database* informs the *System* that the requested game was not found. Use case continues to step 1.
4a. *Player* declines the loaded game board. Use case continues at step 1.

Generate Random GameBoard

Use Case: Generate Random GameBoard
Scope: Flash Point
Level: Subfunction
Intention in Context: *Player* requested the *System* to generate a random board setup.
Multiplicity: Many Players may Generate Random GameBoard, several times.
Primary actor: *Player*
Main Success Scenario:

1. *System* generates a random game board.
2. *Player* notifies *system* he accepts the generated game board.

Extensions:

2a. *Player* does not accept the generated game board. Use Case continues to step 1.

CharacterSelectionMenu

Use Case: CharacterSelectionMenu
Scope: FlashPoint

Level: Subfunction
Intention in Context: The intent of *Player* is to select a character in order to join the Game Board
Multiplicity: Many *Players* may choose a character simultaneously.
Primary actor: *Player*
Main Success Scenario:

1. *System* diplays to *Player* a menu of available characters
   Available characters are: Imaging Technician, Driver/Operator, Rescue Specialist Paramedic, CAFS, HazMat tech, Generalist and Fire Captain
2. *Player* informs *System* of the character he selected.

2a. Selected Character was already taken by another *Player*. Use case continues to step 1.
System has to update the list of available characters.

Executing Turn

Use Case: Execute Turn
Scope: FlashPoint
Level: User Goal
Intention in Context: The intention of the *Player* is to perform their turn within a game of FlashPoint.
Multiplicity: One Player may execute their turn at one time. They will execute a turn multiple times in *Game*.
Primary Actor: *Player*
Secondary Actors: *Player (other than the Current Player).*
Main Success Scenario:

1.) The *System* informs the *Player* that it is their turn.
2.) The *System* informs the *Player* of their available *Action Points*
Step 3 may be repeated until the Player runs out of Action Points or informs the System of their desire to end their Turn.
3.) The *Player* spends their available *Action Points*
4.) The *System* informs the *Player* that their *Turn* has ended.
5.) The *System* processes the *Game State* and displays the result to all *Players*
6.) Advance Fire
7.) Replenish POI

Extensions:

3a) The *Player* performs a Move
3b) The *Player* performs a Chop
3c) The *Player* performs a Extinguish
3d) The *Player* performs a Open/Close Door
3e) The *Player* performs a Drive
3f) The *Player* performs a Rescusitate
3g) The *Player* performs a Identify
3h) The *Player* performs a Dispose HazMat
3i) The *Player* performs a Command Player
3j) The *Player* performs a Swap Role

Move

Use Case: Move
Scope: FlashPoint
Level: Subfunction
Intention in Context: The intention of the *Player* is to *Move* to a specified *Tile.*

Multiplicity: One Player may *Move* at one time. A *Player* may *Move* more than once per Turn.
Primary Actor: *Player*
Secondary Actors:
Main Success Scenario:

1.) The *Player* informs the *System* of their desire to *Move* to a specified location within the Game Grid.
2.) The *System* validates that the *Player* has enough *Action Points* to *Move* to the specified location.
3.) The *System* informs all *Players* of the result of the *Move*
4.) The case returns to <u>Execute Turn</u>, Step 3.

Extensions:
2a) The requested tile contains Fire and the *Player* does not have at least 2 *Action Points* or 2 *Movement Action Points* if the *Player* is currently in the *Rescue Specialist* role.
2a.1) The *System* signals to the *Player* that the *Move* was invalid.
2a.2) The case returns to <u>Executing Turn</u>, Step 3.
2b) The requested tile contains Smoke and the *Player* does not have at least 1 *Action Point* or 1 *Movement Action Points* if the *Player* is currently in the *Rescue Specialist* role.
2b.1) The *System* signals to the *Player* that the *Move* was invalid.
2b.2) The case returns to <u>Executing Turn</u>, Step 3.
2c) The *Player* is holding a Victim, and the specified location contains Fire.
2c.1) The *System* signals to the *Player* that the *Move* was invalid.
2c.2) The case returns to <u>Executing Turn</u>, Step 3.
2d) The specified location contains Fire, and *Player* does not have enough *Action Points* after the requested *Move* to <u>Extinguish Fire</u> or <u>Move</u> again.
2d.1) The *System* signals to the *Player* that the *Move* was invalid.
2d.2) The case returns to <u>Executing Turn</u>, Step 3.
2e) The specified location is on the other side of a *Wall*, and the *Wall* is not *Destroyed*.
2e.1) The *System* signals to the *Player* that the *Move* was invalid.
2e.2) The case returns to <u>Executing Turn</u>, Step 3.
2f) The specified location is on the other side of a *Door*, and the *Door* is *Closed*.
2f.1) The *System* signals to the *Player* that the *Move* was invalid.
2f.2) The case returns to <u>Executing Turn</u>, Step 3.
2f)The *Player* requested a *Move* to an adjacent space with *Fire*.
3a) The *Player* moved to a Tile containing a POI.
3a.1)The *System* displays the result of the POI to all *Players*.

Chop

Use Case: Chop
Scope: FlashPoint
Level: Subfunction
Intention in Context: The intention of the *Player* is to *Chop* a specified *Wall*.
Multiplicity: One Player may *Chop* at one time. A *Player* may *Chop* more than once per Turn.
Primary Actor: *Player*
Secondary Actors:
Main Success Scenario:
1.) The *Player* informs the *System* that they wish to *Chop* a *Wall*.
2.) The *System* validates that the request can be performed.
3.) The *System* informs all *Players* of the result of the *Chop*
4.) The case returns to <u>Execute Turn</u>, Step 3.

Extensions

2a) The *Player* is not standing next to a *Wall*.
  2a.1) The *System* signals to the *Player* that the *Chop* was invalid.
  2a.2) The case returns to Executing Turn, Step 3.
2b) The *Player* does not have at least 2 *Action Points*
  or 1 if the *Player* is currently in the *Rescue Specialist* role.
  2b.1) The *System* signals to the *Player* that the *Chop* was invalid.
  2b.2) The case returns to Executing Turn, Step 3.
2c) The *Wall* is already *Destroyed*.
  2c.1) The *System* signals to the *Player* that the *Chop* was invalid.
  2c.2) The case returns to Executing Turn, Step 3.

Extinguish

Use Case: Extinguish
Scope: FlashPoint
Level: Subfunction
Intention in Context: The intention of the *Player* is to *Extinguish* a *Fire* or *Smoke*.
Multiplicity: One Player may *Extinguish* at one time. A *Player* may *Extinguish* more than once per Turn.
Primary Actor: *Player*
Secondary Actors:
Main Success Scenario:

1.) The *Player* informs the *System* of their desire to *Extinguish* a specified *Tile*.
2.) The *System* validates the request.
3.) The *System* informs all *Players* of the result of the *Extinguish*
4.) The case returns to Execute Turn, Step 3.

Extensions:

1a) The *Tile* is a *Smoke Tile*.
  1a.1) Extinguish Smoke.
1b) The *Tile* is a *Fire Tile*.
  1b.1) Extinguish Fire.
2a) The *Player Card* of the *Player* is not adjacent or directly on the specified *Tile*
  2a.1) The *System* signals to the *Player* that the *Extinguish* was invalid.
  2a.2) The case returns to Executing Turn, Step 3.
2b) The specified *Tile* is not a *Smoke Tile* or *Fire Tile*
  2b.1) The *System* signals to the *Player* that the *Extinguish* was invalid.
  2b.2) The case returns to Executing Turn, Step 3.

Extinguish Smoke

Use Case: Extinguish Smoke
Scope: FlashPoint
Level: Subfunction
Intention in Context: The intention of the *Player* is to *Extinguish Smoke*
Multiplicity: One Player may *Extinguish Smoke* at one time. A *Player* may *Extinguish Smoke* more than once per Turn.
Primary Actor: *Player*
Secondary Actors:
Main Success Scenario:

1.) The *Player* specifies a *Smoke Tile* to *Extinguish Smoke*
2.) The *System* validates the request.
3.) The case returns to Extinguish, Step 3.

Extensions:
2a) The *Player Card* of the *Player* is not adjacent or directly on the specified *Smoke Tile*
 2a.1) The *System* signals to the *Player* that the *Extinguish Smoke* was invalid.
 2a.2) The case returns to Executing Turn, Step 3.
2b) The *Player* does not have at least 1 *Ability Point*, or *Extinguish Ability Point* if playing the *CAFSFirefighter* role. The *Player* must have 2 *Ability Points* if they are currently playing the *Paramedic* role.
 2b.1) The *System* signals to the *Player* that the *Extinguish* was invalid.
 2b.2) The case returns to Executing Turn, Step 3.

Extinguish Fire

Use Case: Extinguish Fire
Scope: FlashPoint
Level: Subfunction
Intention in Context: The intention of the *Player* is to *Extinguish Fire*
Multiplicity: One Player may *Extinguish Fire* at one time. A *Player* may *Extinguish Fire* more than once per Turn.
Primary Actor: *Player*
Secondary Actors:
Main Success Scenario:
1.) The *Player* specifies a *Fire Tile* to *Extinguish Fire*, and whether they would like to turn the Fire to Smoke, or to a Regular Tile.
2.) The *System* validates the request.
3.) The case returns to *Extinguish*, Step 3.
Extensions:
2a) The *Player Card* of the *Player* is not adjacent or directly on the specified *Fire Tile*
 2a.1) The *System* signals to the *Player* that the *Extinguish Fire* was invalid.
 2a.2) The case returns to Executing Turn, Step 3.
2b) The *Player* does not have at least 1 *Ability Point* or *Extinguish Ability Points* if playing the *CAFSFirefighter* role, and the request was to turn the Fire to Smoke. The *Player* must have 2 *Ability Points* if they are currently playing the *Paramedic* role.
 2b.1) The *System* signals to the *Player* that the *Extinguish* was invalid.
 2b.2) The case returns to Executing Turn, Step 3.
2c) The *Player* does not have at least 2 *Ability Point* or *Extinguish Ability Points* if playing the *CAFSFirefighter* role, and the request was to turn the Fire to Regular Tile. The *Player* must have 4 *Ability Points* if they are currently playing the *Paramedic* role.
 2c.1) The *System* signals to the *Player* that the *Extinguish* was invalid.
 2c.2) The case returns to Executing Turn, Step 3.

Open/Close Door

Use Case: Open/Close Door
Scope: FlashPoint
Level: Subfunction
Intention in Context: The intention of the *Player* is to *Open* or *Close* a *Door*
Multiplicity: One Player may *Open/Close Door* at one time. A *Player* may *Open/Close Door* more than once per Turn.
Primary Actor: *Player*
Secondary Actors:

Main Success Scenario:
1.) The *Player* indicates to the *System* that they wish to interact with a *Door*
2.) The *System* validates the request.
3.) The *System* informs all *Players* of the result of *Open/Close Door*
4.) The case returns to Execute Turn, Step 3.

Extensions:
2a) The *Player* does not have enough *Ability Points* to *Open/Close Door*
  2a.1) The *System* signals to the *Player* that the *Open/Close Door* was invalid.
  2a.2) The case returns to Executing Turn, Step 3.
2b) The *Player Card* is not directly adjacent to the *Door*.
  2b.1) The *System* signals to the *Player* that the *Open/Close Door* was invalid.
  2b.2) The case returns to Executing Turn, Step 3.
3a) If the *Door* was Closed.
  3a.1) All *Players* are informed that the *Door* is now Open.
  3a.2) The case returns to Executing Turn, Step 3.
3b) If the *Door* was Open.
  3b.1) All *Players* are informed that the *Door* is now Closed.
  3b.2) The case returns to Executing Turn, Step 3.

Drive

Use Case: Drive
Scope: FlashPoint
Level: Subfunction
Intention in Context: The intention of the *Player* is to *Drive* a Vehicle.
Multiplicity: One Player may *Drive* at one time. A *Player* may *Drive* more than once per Turn.
Primary Actor: *Player*
Secondary Actors:
Main Success Scenario:
1.) The *Player* informs the System they wish to *Drive* a Vehicle.

Extensions:
1a) The *Player* informed the *System* that they wish to *Drive Ambulance*.
  1a.1) Call Ambulance.
1b) The *Player* informed the *System* that they wish to *Drive Engine*.
  1b.1) Drive Engine

Call Ambulance

Use Case: Call Ambulance
Scope: FlashPoint
Level: Subfunction
Intention in Context: The intention of the *Player* is to *Call Ambulance*
Multiplicity: One Player may *Call Ambulance* at one time. A *Player* may *Call Ambulance* more than once per Turn.
Primary Actor: *Player*
Secondary Actors:
Main Success Scenario:
1.) The *Player* informs the System they wish to *Call Ambulance*.
2.) The *System* validates that the *Player* has enough *Action Points* to *Call Ambulance*.
3.) The *System* displays the result of the *Call Ambulance* to all *Players*

Extensions:

2a) The *Player* does not have at least 2 *Action Points*
  2a.1) The *System* informs the Player that the request was invalid.
  2a.2) The case returns to <u>Executing Turn</u>, Step 3.

Drive Engine

Use Case: Drive Engine
Scope: FlashPoint
Level: Subfunction
Intention in Context: The intention of the *Player* is to *Drive Engine.*
Multiplicity: One Player may *Drive Engine* at one time. A *Player* may *Drive Engine* more than once per Turn.
Primary Actor: *Player*
Secondary Actors:
Main Success Scenario:
1.) The *Player* informs the System they wish to *Drive Engine.*
2.) The *System* validates the request.
3.) The *System* displays the result of the *Drive Engine* to all *Players*

Extensions:
2a) The *Player* does not have at least 2 *Action Points.*
  2a.1) The *System* informs the Player that the request was invalid.
  2a.2) The case returns to <u>Executing Turn</u>, Step 3.
2b) The *Player Character* is currently not on the same Tile as the Engine.
  2b.1) The *System* informs the Player that the request was invalid.
  2b.2) The case returns to <u>Executing Turn</u>, Step 3.

Resuscitate

Use Case: Resuscitate Victim
Scope: Flash Point
Level: User Goal
Intention in Context: The intention of *Player* is to resuscitate a victim.
Multiplicity: Many *Players* can resuscitate victims.
Primary actor: *Player*
Main Success Scenario:
1. *Player* informs *System* he wants to resuscitate a victim.
2. *System* checks if the *Player* is currently playing the Paramedic Role and that the *Player* has enough *Action Poi*
3. *System* deducts 1 action point from *Player*
4. *System* informs all *Players* a victim has been resuscitated

Extensions:
2a.*Player* did not have enough action points, or is not playing the Paramedic Role.
  2a.1)The *System* informs the *Player* that the request was invalid.
  2a.2) The use case returns to <u>Turn</u> step 3.

Dispose

Use Case: Disposal of Hazardous Materials
Scope: Flash Point
Level: User Goal
Intention in Context: The intention of *Player* is to dispose of hazardous materials.
Multiplicity: Many *Players* can dispose of hazardous materials.
Primary actor: *Player*
Main Success Scenario:

1. *Player* informs *System* he wants to Dispose Hazardous Material.
2. *System* checks if *Players* has the Hazmat Technician Role and has enough Action Points
3. *System* deducts 2 action points from *Player*
4. *System* informs all *Players* a hazardous material has been disposed

Extensions:
2a.*Player* did not have enough action points, or is not playing the HazMat Technician Role.
2a.1)The *System* informs the *Player* that the request was invalid.
2a.2) The use case returns to <u>Turn</u> step 3.

Identify

Use Case: Identify Victim
Scope: Flash Point
Level: User Goal
Intention in Context: The intention of *Player* is to identify a victim.
Multiplicity: Many *Players* can identify victims.
Primary actor: *Player*
Main Success Scenario:
1. *Player* informs *System* identify a victim.
2. *System* checks if *Players* has the Imaging Technician Role and has enough Action Points
3. *System* deducts 1 action points from *Player*
4. *System* informs all *Players* of the revealed POI

Extensions:
2a.*Player* did not have enough action points, or is not playing the Imaging Technician Role.
2a.1)The *System* informs the *Player* that the request was invalid.
2a.2) The use case returns to <u>Turn</u> step 3.

Chat

Use Case: Chat
Scope: Flash Point
Level: User Goal
Intention in Context: The intention of the *Player* is to chat with one another.
Multiplicity: Every *Player* can chat with one another.
Primary Actor: *Player*
Secondary Actors: *Player (other than the Current Player)*
Main Success Scenario:
1. *Player* informs the *System* that they want to chat using either <u>text message</u> or <u>audio message</u>.
2. The *System* displays the message to all *Players*

Send text message

Use Case: Send text message
Scope: Flash Point
Level: Subfunction
Intention in Context: The intention of the *Player* is to send a message to other *Players.*
Multiplicity: Several *Players* can send messages simultaneously.
Primary Actor: *Player*
Secondary Actors: *Player (other than the Current Player)*
Facilitator Actors: Peripheral Devices
Main Success Scenario:

1. The *Player* informs the *System* that they want to Chat using text messages.
2. The *Player* enters a text message and send it to the *System*.
3. The *System* informs all *Players* of the text message.

Extensions:

2a The *Player* cancels the action. Nothing is sent.

Send audio message

Use Case: Send audio message
Scope: Flash Point
Level: Subfunction
Intention in Context: The intention of the *Player* is to send an audio message to other *Players*.
Multiplicity: Several *Players* can send messages simultaneously.
Primary Actor: *Player*
Secondary Actors: *Player (other than the Current Player)*
Facilitator Actors: Peripheral Devices
Main Success Scenario:

1. The *Player* informs the *System* that they want to Chat using audio messages.
2. The *Player* records an audio message and send it to the *System*.
3. The *System* informs all *Players* of the text message.

Extensions

2a. The *Player* cancels the action. Nothing is sent.
3b. The *System* detects that one *Player* has muted the audio messages. Message not played.

Mute audio messages

Use Case: Mute audio messages
Scope: Flash Point
Level: Subfunction
Intention in Context: The intention of the *Player* is to mute incoming audio messages.
Multiplicity: Several *Players* can choose to mute the audio messages at the same time.
Primary Actor: *Player*
Secondary Actors:
Main Success Scenario:

1.) The *Player* informs the *System* that they want to mute all incoming audio messages.
2.) The *System* will stop playing all incoming audio messages.

Extensions:

1a) The *Player* informs the *System* that they want to unmute all incoming audio messages.
2a) The *System* will start playing all incoming audio messages.

Call for Vote

Use Case: Call for Vote
Scope: Flash Point
Level: User Goal
Intention in Context: The intention of the *Player* is to call for vote to decide on something.
Multiplicity: Every Player may Call for Vote, but only one vote can exist at a time.
Primary Actor: *Player*
Secondary Actors: *Player (other than the Current Player)*
Main Success Scenario:

1.) The *Player* informs the *System* that they want to <u>Call for Vote</u>.
2.) The *Player* informs the *System* of their choice to <u>Call for timeout</u>, or <u>Vote to kick Player</u>.
3.) The *System* validates the request.
4.) Other *Players* cast their votes to the *System*
5.) The *System* collects the *Players'* choices and calculates the result.
6.) The *System* informs the *Players* of the result.

Extensions:
2a) The *Player* cancels the action.
2a.1) End of use case.
4b) Not enough *Players* vote, end of use case.

Call for Timeout

Use Case: Call for timeout
Scope: Flash Point
Level: Subfunction
Intention in Context: The intention of the *Player* is to pause the game state.
Multiplicity: Every Player may <u>Call for Vote</u>, but only one vote can exist at a time.
Primary Actor: *Player*
Secondary Actors: *Player (other than the Current Player)*
Main Success Scenario:
1.) The *Player* initiates the vote.
2.) The other *Players* <u>cast their vote</u>.
3.) Most *Players* agree to pause the game, the game state is paused for a predetermined lenght of time.
4.) The system shows all *Players* the time remaining for timeout
5.) Repeat (4) until timer reaches 0.
6.) Resume game state.

Extensions:
3a) Most Players disagree to pause the game, the vote is canceled and the game state is not paused.
3b) Majority of the Players do not cast a vote, voting is canceled and the game state is not paused.

Vote to kick Player

Use Case: Vote to kick Player
Scope: Flash Point
Level: Subfunction
Intention in Context: The intention of the *Player* is to kick another *Player* from the game.
Multiplicity: One *Player* may <u>vote to kick</u> another *Player* at a time.
Primary Actor: *Player*
Secondary Actors: *Player (other than the Current Player)*
Main Success Scenario:
1. The *Player* informs the *System* that they would like to initiate the vote.
2. The other *Players* <u>cast their votes</u>, except the *Player* who is being voted to kick out.
   the *System* then removes the targeted *Player* from the game.

Extensions:
2a Most *Players* disagree to kick the targeted *Player*,
   the vote is canceled and the targeted *Player* remains in game.
2b Majority of the *Players* do not cast a vote,
   voting is canceled and the *Player* remains in game.

Cast vote

Use Case: Cast vote
Scope: Flash Point
Level: User Goal
Intention in Context: The intention of the *Player* is to cast a vote.
Multiplicity: Each *Player*, except the one who initiated the vote and the one being targeted for kick out, can cast a vote.
Primary Actor: *Player*
Main Success Scenario:

1. The *System* asks the *Player* to cast a vote.
2. All *Players* inform the *System* of their choice.
3. The *System* informs all *Players* of the result.

Place a Marker

Use Case: Place a Marker
Scope: Flash Point
Level: User Goal
Intention in Context: The intention of the *Player* is to place a marker on the game board.
Multiplicity: Each *Player* can place can place a marker for each type
Primary actor: *Player*
Main Success Scenario:

1. *Player* informs *system that they want to place a marker*
2. *Player* chooses the type of marker to be placed.
3. *Player* informs *system* of the location on the board where the marker is to be placed.
4. *System* displays to all *Players* the marker on the board.

4a. *Player* informs *system* he wishes to to cancel. Marker was not place and End of Use Case.

View Player's Information

Use Case: View Player's Information
Scope: Flash Point
Level: User Goal
Intention in Context: The intention of the *Player* is to view other *Player's* information.
Multiplicity: Each *Player* can view other *Players'* information.
Primary Actor: *Player (who requested the information)*
Secondary Actors: *Player (whose information is being requested)*
Main Success Scenario:

1. *Player* informs the *System* that they want to view a *Player's* information.
2. The *System* retrieves the requested *Player's* information.
3. The *System* returns the requested information to the *Player*.

Command a Player

Use Case: Command a Player
Scope: Flash Point
Level: Subfunction
Intention in Context: The intention of the *Player* who is playing as the *Captain* role is to command other *Player* to the specified location.
Multiplicity: A *Captain Player* may command one *Player* at a time, and can command two times for free, or more as long as the Action Point (AP) is enough.

Primary Actor: *Captain Player*
Secondary Actors: *Player*
Main Success Scenario:

1. *Captain Player* informs the *System* that they wish to command a *Player*.
2. The *Captain Player* informs the *System* of which *Player* that they wish to command.
3. The *Captain Player* informs the *System* of their command for the *Player*, such as Move to a location, Open/Close Doors or Pick Up/Down Victim/Hazmat.
4. The *System* deducts the AP from the *Captain Player*.
5. The *System* executes the command on the *Player*.
6. The *System* informs all *Player* of the result.
7. The case returns to Execute Turn, Step 3.

Extensions:

2a) The *System* detects that the *Captain Player* does not have enough AP.
System signals the *Captain Player*.
2a)1. The case returns to Execute Turn, Step 3.

2-3b) *Captain Player* cancels the action.
2a-3b)1. The case returns to Execute Turn, Step 3.

4c) *Captain Player* provided an invalid command to the *Player*.
The *System* informs the *Captain Player* and prompts the action again (Repeat (3)).

Fire Deck Gun

Use Case: Fire Deck Gun
Scope: Flash Point
Level: Subfunction
Intention in Context: The intention of the *Player* is to fire a deck gun at a specified Quadrant.
Multiplicity: Only one *Player* can use the deck gun at a time. A *Player* can fire the deck gun multiple times, as long as they have enough Action Points (AP).
Primary Actor: *Players*
Secondary Actors:
Main Success Scenario:

1. The *Player* informs the *System* that they wish to fire a deck gun.
2. The *System* deducts the AP from the *Player*, depending on their role.
(2AP for Driver/Operator and 4AP for others)
3. The *System* calculates the deck gun's target space. Then, the *System* extinguishes all smoke and fire in the target space, and splashes over into each adjacent space, completely extinguishing any fire or smoke.
4. The *System* informs all *Player* of the result.
5. The case returns to Execute Turn, Step 3.

Extensions:

2a) The *System* detects that the *Player* does not have enough AP.
System signals the *Player*.
2a)1. The case returns to Execute Turn, Step 3.

Swap Role

Use Case: Swap Role
Scope: Flash Point
Level: Subfunction
Intention in Context: The intention of the *Player* is to change into other role.
Multiplicity: Every *Player* can swap role.

Primary Actor: *Players*
Secondary Actors:
Main Success Scenario:

1. The *Player* informs the *System* that they wish to swap role.
2. The *System* shows the Specialists available that are not currently in play and prompts the *Player* for their choice.
3. The *Player* chooses a Specialist role.
4. The *System* deducts 2 AP from the *Player* and assigns new role to the *Player*. of the newly selected Specialist for the entire turn.
6. The *System* informs all *Player* of the result.
7. The case returns to Execute Turn, Step 3.

Extensions:

1a) The *System* detects that the *Player* does not have enough AP.
System signals the *Player*.
2a)1. The case returns to Execute Turn, Step 3.
1b) The *System* detects that the *Player* is not in the same space as the Engine.
System signals the *Player*.
’ 2b)1. The case returns to Execute Turn, Step 3.
2-3c) *Player* cancels the action.
2-3c)1. The case returns to Execute Turn, Step 3.

ReplenishPointOfInterest

Use Case: ReplenishPointOfInterest
Scope: Flash Point
Level: Subfunction
Intention in Context: The intent of Controller is to replenish the game board with according points of interest.
Multiplicity: The Controller *Replenish Point Of Interest* once per Turn.
Primary actor: *Controller*
Main Success Scenario:
Steps 1 through 3 can be repeated until there are 3 POI's present

1. *Controller* goes through game board squares to check if 3 POIs are present.
2. *Controller* randomly chooses a tile to place missing POI.
3. *System* displays to all *Players* a new POI a the randomly chosen tile.

Extensions:

2a. *Controller* counted 3 POIs. End of Use Case.
3a. *Controller* placed new POI on fire or smoke token.
3a.1 *System* replaces token by POI. Use Case continues at step 3.
3b. *Controller* placed new POI on a character position.
3b.1 *System* reveals POI to all *Players*. Use Case continues at step 1.

Advance Fire

Use Case: Advance Fire
Scope: Flash Point
Level: Subfunction
Intention in Context: A Turn has ended and the *Controller* needs to Advance Fire.
Multiplicity: This occurs once per turn by only the *Controller*.
Primary actor: *Controller*
Secondary actor: *Player*
Main Success Scenario:

1.) The *Controller* randomly chooses a Tile on which to place a Smoke.
Step 2 can be repeated until there are no secondary effects remaining
2.) The *Controller* checks all secondary effects.
3.) The *System* displays the result to all *Players*.

Extensions:
1a) If the Smoke is placed on an existing Smoke Tile, the Smoke Tile becomes a Fire Tile.
1b) If the Smoke is placed adjacent to a Fire Tile, a Fire Tile is placed instead of Smoke.
1c) If the Smoke is placed on an existing Fire Tile, Explosion
1d) If the Smoke is placed on a HotSpot, Advance Fire.
2a) Any Smoke Tile adjacent to a Fire Tile becomes a Fire Tile.
2b) In Family Mode, any *Player Character* in a space with a Fire are knocked down.
The *Player Character* is placed on the closest Ambulance Parking Spot.
If the *Player Character* was carrying a Victim, that victim is Lost. In Experienced Mode, the *Player Character* is placed on wherever the Ambulance is.
2c) Any Victim or POI on a Fire Tile are Lost.
2d) If a HazMat is in a space with fire, Explosion. The HazMat is replaced with a HotSpot.

Explosion

Use Case: Explosion
Scope: Flash Point
Level: Subfunction
Intention in Context: During Advance Fire, the *Controller* must spread Fire in all 4 directions from the Target Tile.
Multiplicity: This can occur more than once per Turn by only the *Controller*.
Primary actor: *Controller*
Secondary actor: *Player*
Main Success Scenario:
1.) The *Controller* resolves the Explosion

Extensions:
1.a) If the Tile adjacent to the Target Space is Open, the Open Tile becomes a Fire Tile.
1.b) If the Tile adjacent to the Target Space is Smoke Tile, then the Smoke Tile becomes a Fire Tile.
1.c) If the Tile adjacent to the Target is a Wall, the Wall is damaged by 1.
1.d) If the Tile adjacent to the Target is a closed door, the Door is removed from the game.

Pick up Victim/Hazmat

Use Case: Pick up Victim/Hazmat
Scope: FlashPoint
Level: Subfunction
Intention in Context: The *Player* intends to pick up the victim or hazmat.
Multiplicity: Only the *Player* whose turn it is currently can do pick up action.
Primary Actor: *Player*
Main Success Scenario:
1. *System* validates if *Player Character* is adjacent to an object that can be picked up (i.e. Victim or HazMat), then unlocks the pick up action for the *Player*.
2. *Player* informs the *System* that they wish to pick up the specified object.
3. *System* deducts 2 AP from the *Player*. *Player* picks up the object.
4. *System* informs all *Players* the result of this action.
5. Use Case returns to Execute Turn, Step 3.

Put down victim/Hazmat

Use Case: Put down Victim/Hazmat
Scope: FlashPoint
Level: Subfunction
Intention in Context: The Player intends to put down the victim or hazmat.
Multiplicity: Only the Player whose turn it is currently and is carrying an object can do put down action.
Primary Actor: Player
Main Success Scenario:

1. *Player* informs the *System* they wish to put down the object they are carrying.
2. *System* checks if the location is valid for the *Player*.
3. *Player* puts down the object.
4. *System* informs all *Players* the result of this action.
5. Use Case returns to Execute Turn, Step 3.

Extensions:

2a) *System* determines that the *Player's* location is invalid to put down the object.
It signals the Player with the error. System informs *Player* of an error.
2a)-1. Use Case returns to Execute Turn, Step 3.