Team: Adam Holt, Joseph Rener, Brent Pivnik, Connor Guerrieri

Title: Escape!

Project Summary: Escape is a puzzle game based on the Android operating system, utilizing a touch interface in order to move a character on the screen. The user's goal is to navigate the character through a maze filled with enemies in order to reach the next level.

Project Requirements:
Business Requirements: N/A

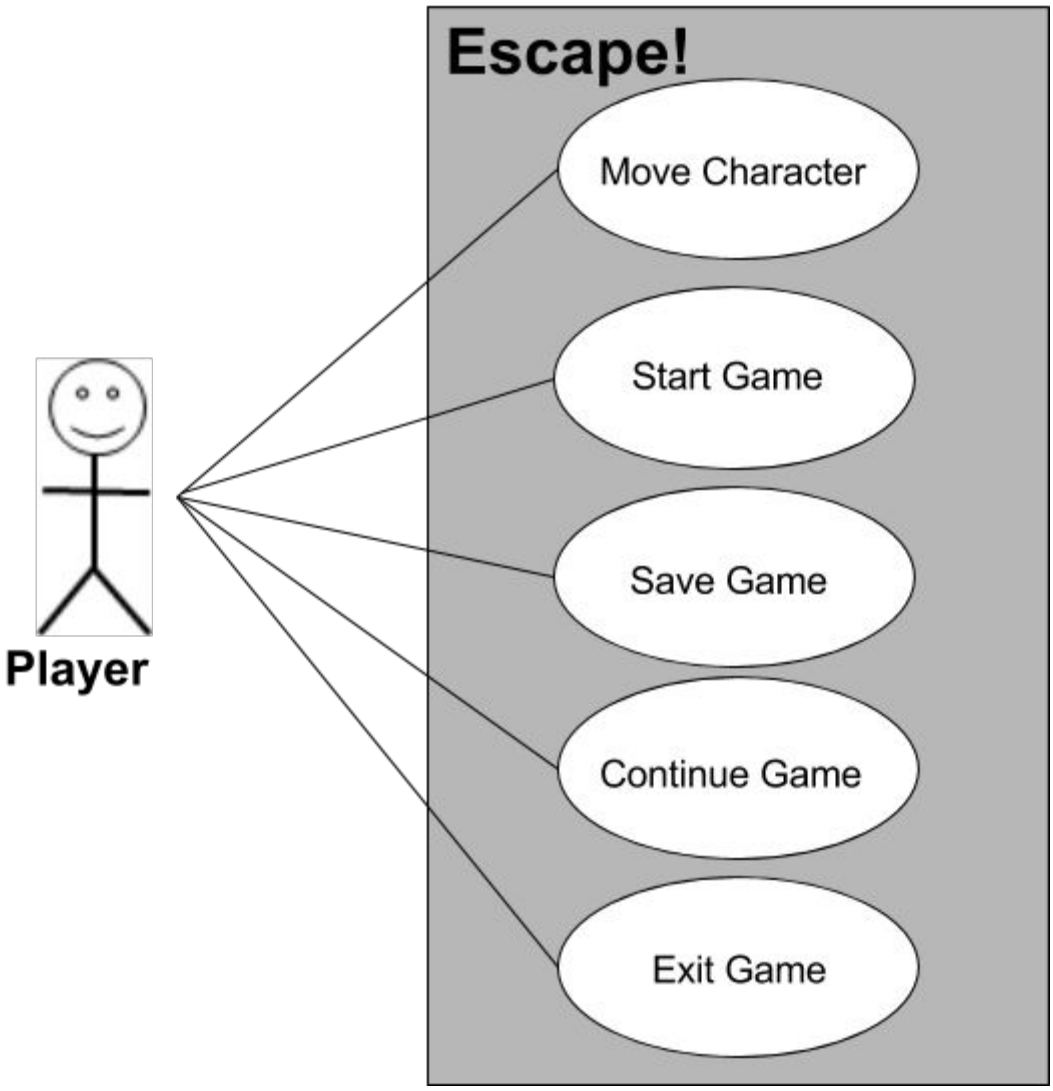
User Requirements			
ID	**Requirement**	**Topic Area**	**Priority**
UR-01	As a user I want to be able to press the on-screen controls to make the character move in the cardinal directions.	User Experience	High
UR-02	As a user I would like the option to start a new game when opening the app instead of resuming from my previous save	User Experience	Medium
UR-03	As a user I want to be able to end the current game and start a new game at any point.	User experience	Low
UR-04	As a user I want the option to save my current game at any point	Data storage	Low

Functional Requirements			
ID	**Requirement**	**Topic Area**	**Priority**
FR-01	The app must be able to save the game state to be opened later	Data storage	Medium
FR-02	When the player returns to the game, they should be given the option to resume from their last position or begin a new game.	User experience	Medium
FR-03	Enemies will follow a random pattern of movement, triggered by character movement	Artificial Intelligence	Medium
FR-04	When the user selects to start a new game, the app should prompt the user to confirm that they want to overwrite their previous game	User experience	Medium
FR-05	There should be a “continue” button on the opening screen that allows the user to resume their previous game, but is greyed-out if there is no saved data	User experience	

Non-Functional Requirements			
ID	**Requirement**	**Topic Area**	**Priority**
NFR-01	Character must have predictable movement based on predefined rules	User experience	High
NFR-02	The game should prevent the player's device from sleeping while the game is being played	User experience	High
NFR-03	No noticeable lag between user input and character movement	User experience	Medium
NFR-04	The app should save the most recent state of the game board and replace the saved data	Data storage	Medium
NFR-05	The app should automatically save the current game at thirty second intervals in such a way that does not affect the game play	Data Storage	Low
NFR-06	Character/Enemy motion should be smooth and animated	Animation	High

Users and Tasks:

- **User:**
  - Player
- **Use Cases:**
  - Move character
  - Start Game
  - Save game
  - Continue game
  - Exit game
- **Use Case Diagram:**



- **Use Case Documents:**

Use Case ID:	**UC-01**
Use Case Name:	Move Character
Description:	The player can move the character using an onscreen touch directional pad.

Actors:	Player	
Pre-conditions:	Character is in cover and wants to move to another location	
Post-conditions:	Character has stopped at the next cover object	
Frequency of Use:	Whenever the player needs to relocate their character	
Flow of Events:	**Actor Action**	**System Response**
	Touch the onscreen directional pad	The character moves in the chosen direction. This will trigger enemy movement.
Variations:	Depending on the direction chosen, the player model moves in that direction	

Notes and Issues:	Player can only move character in cardinal directions
Developer Notes:	Initial release will use a control pad - long term goal is to use swipe gestures

Use Case ID:	**UC-02**
Use Case Name:	Start Game
Description:	The player can start a new game instead of loading a saved game

Actors:	Player	
Pre-conditions:	Player is at the opening or loss screen	
Post-conditions:	A new game is begun	
Frequency of Use:	Whenever the player wishes to start a new game	
Flow of Events:	**Actor Action**	**System Response**
	Player clicks the start or restart game button	New game is loaded
Variations:	A new game can be started from the opening screen or after dying	

Notes and Issues:	This action can be performed from multiple screens within the game
Developer Notes:	When a new game is started, saved data should be purged

Use Case ID:	**UC-03**
Use Case Name:	Save Game
Description:	The player can save the game at any point during game-play.

Actors:	Player	
Pre-conditions:	Player is in an active game	
Post-conditions:	The game state is successfully stored in internal memory	
Frequency of Use:	Whenever the player wishes to leave the game and return at a later time; likely once per session	
Flow of Events:	**Actor Action**	**System Response**
	Player clicks an on screen save button to save the game	State of current game is saved
Variations:	The game can be saved at any level at any point.	

Notes and Issues:	Information is saved in internal storage of the device, and can only be accessed by the game (private)
Developer Notes:	Saving the game should overwrite any existing saved data

Use Case ID:	**UC-04**
Use Case Name:	Continue Game
Description:	The player can choose to continue a pre-existing game.

Actors:	Player	
Pre-conditions:	Game has been launched and a game state has been previously saved	
Post-conditions:	The saved game is reinitialized	
Frequency of Use:	Whenever the app is reopened and the player chooses to load a saved game as opposed to start a new one	
Flow of Events:	**Actor Action**	**System Response**
	Player opens the app	Display initial game screen
	Player selects “Continue Game”	System restores state of saved game
Variations:	N/A	

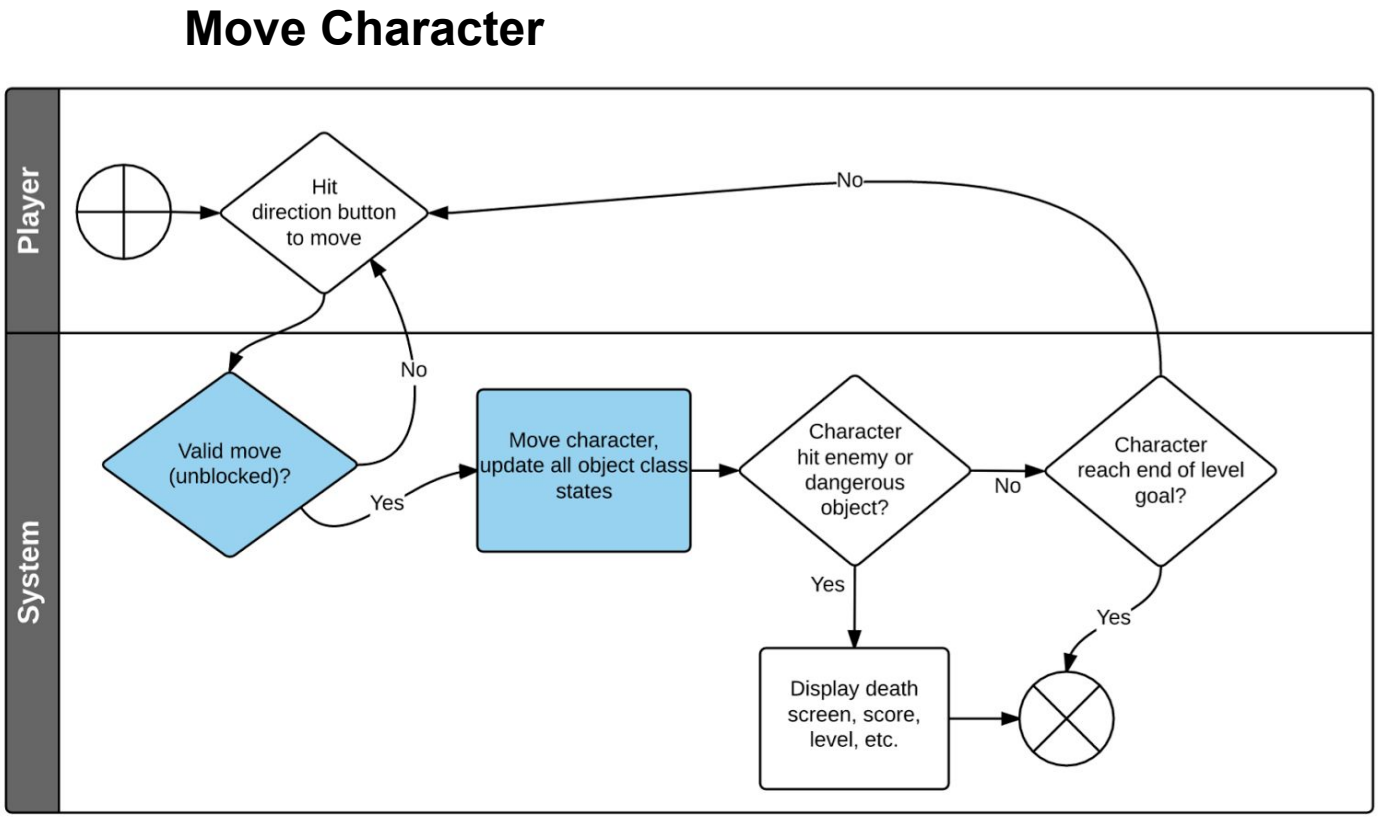
Notes and Issues:	If there is no saved game state, the option to continue is not clickable
Developer Notes:	

Use Case ID:	**UC-05**
Use Case Name:	Exit Game
Description:	The game can be minimized and the current game state is temporarily saved

Actors:	Player	
Pre-conditions:	The game is open	
Post-conditions:	The game is not open	
Frequency of Use:	As often as the game is opened	
Flow of Events:	**Actor Action**	**System Response**
	Hit the home button or the back button on the device	Close the app
	Move on with his/her life	Be sad
Variations:	May be accompanied with option to save progress	

Notes and Issues:	Game state is saved temporarily, but not in external storage. This information will go away when the app is killed (which is a device property, not an app one)
Developer Notes:	

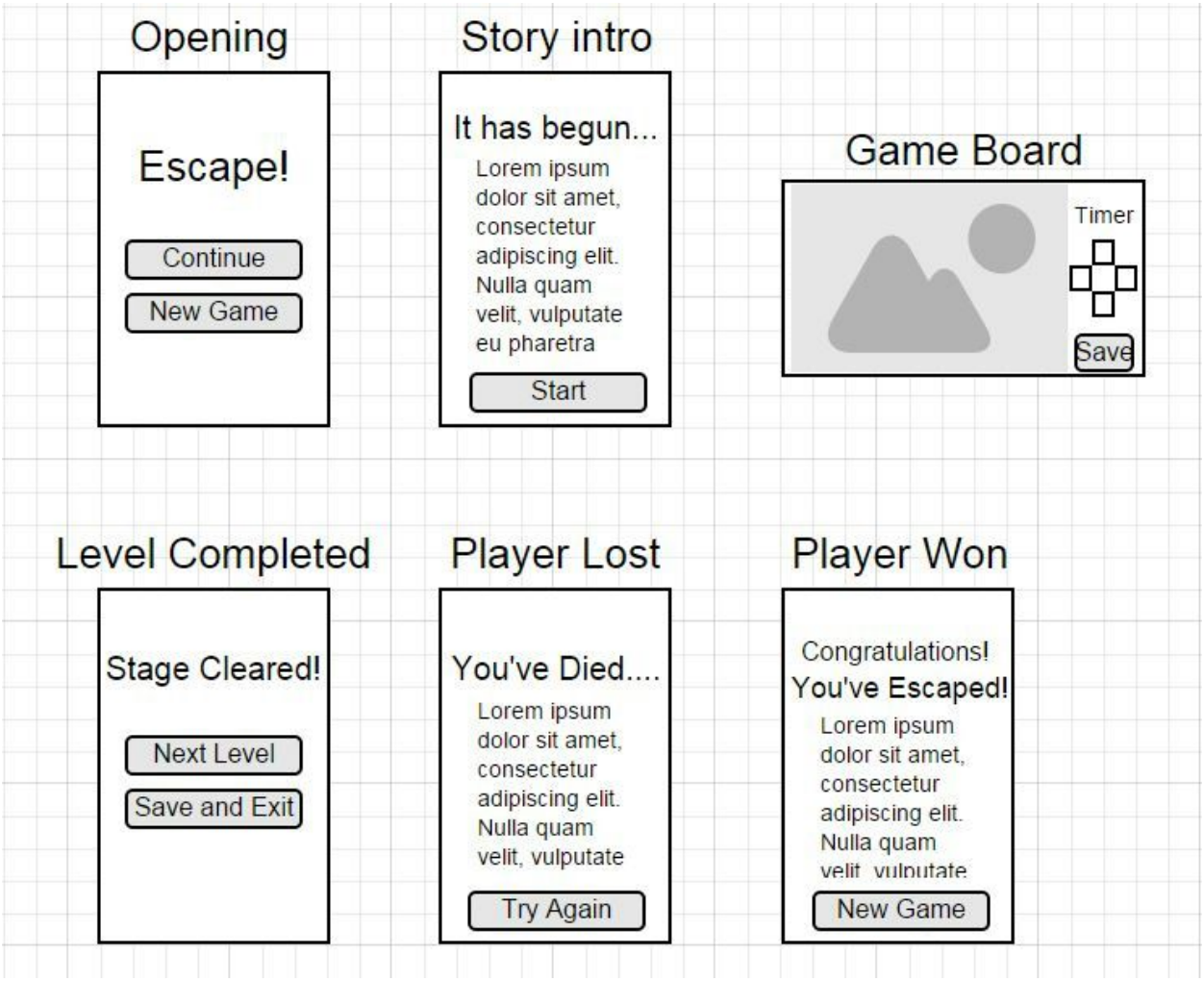
Activity Diagram:



Data Storage:

As an android app, the game state will automatically be stored in memory when the app is exited. However, the user will be able to hard save the game state so that even when the app is killed, when reopened, the user can retrieve his data. This information will be saved into internal storage (a native android operation for saving information in device memory.)

UI Mockups:

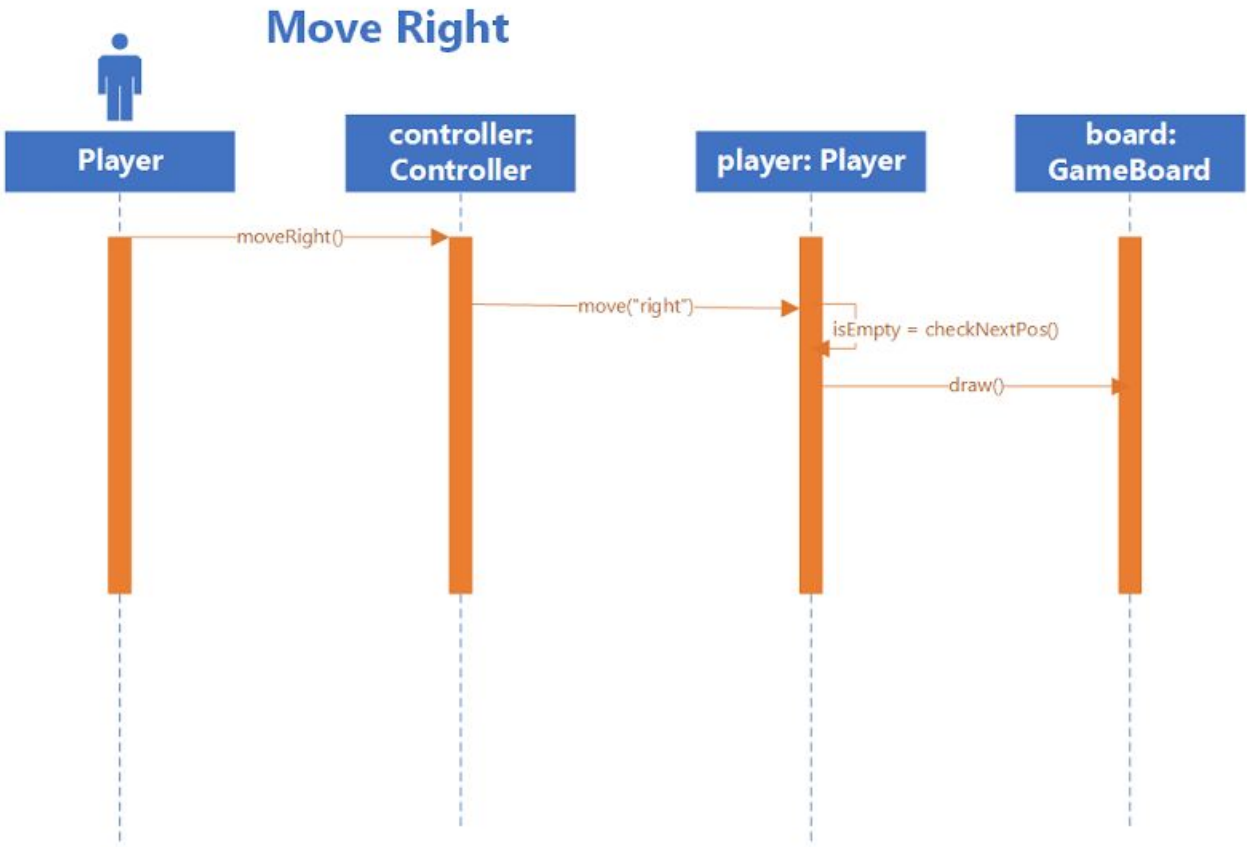


- Navigation:
  - Opening
    - Displayed whenever game is launched
    - Continue button leads to saved Game Board. This button is grayed out if none exists
    - New Game Button leads to Story intro screen
  - Story Intro
    - Start button leads to Game Board
  - Game Board
    - Can lead to Level Completed, Player Lost, or Player Won screens

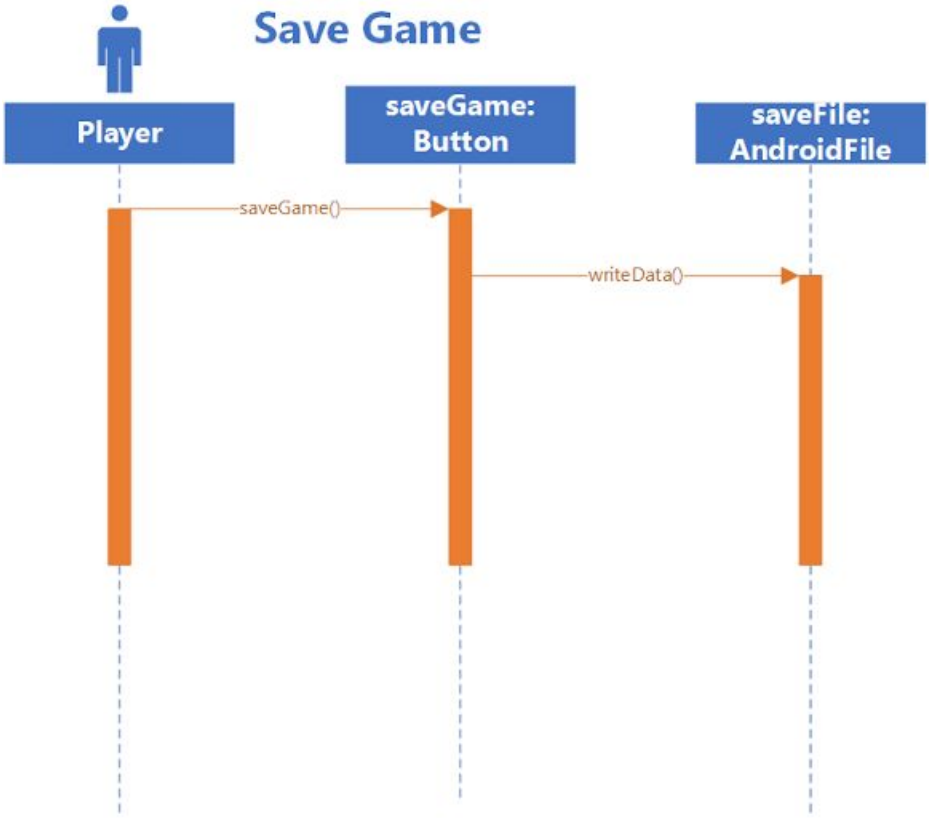
- Level Completed
  - Triggered on level complete
  - Next Level returns to Game Board screen (with new baord)
- Player Lost
  - Try Again button opens Story Intro screen
- Player Won
  - New Game button opens Story Intro screen

User Interaction:

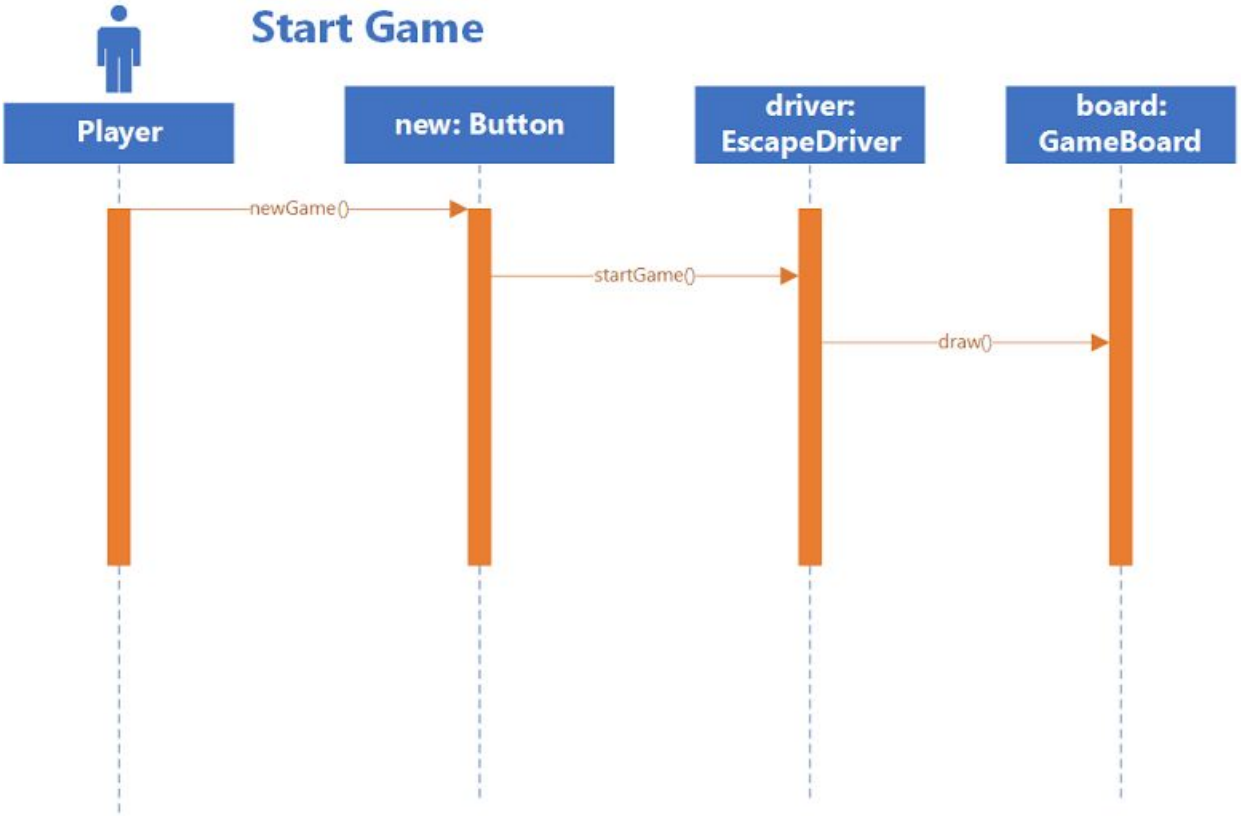
- The user can move the character in a direction
- The user can save the game state
- The user can continue a saved game
- The user can start a new game
- The user can exit the game



Save Game



Start Game



Class Diagram: