Use case: Place a piece

Primary Actor: Player

Stakeholders and Interests:

Player: wants to place their piece on the game board, wants all placements of pieces to abide by the rules of Blokus.

Development team: wants pieces to appear on the game board exactly how they were intended to be placed, wants the system to correctly validate that the placement of a piece is legal and inform the player if their placement isn't valid, wants a piece to become unavailable to the player after it is place on the game board.

Preconditions:

Blokus game has been initiated by a player and set up by the system.

Postconditions:

The piece stays on the board for the remainder of the game and is removed from the pieces that are currently available to the player who made the placement of that piece.

Main Success Scenario:

1.) The system provides the user with the pieces that are available to be placed (i.e. those that haven't been placed yet).
2.) The player selects a piece and chooses whether to place it, rotate it or flip it. [Alt1: Player chooses to flip the piece] [Alt2: Player chooses to rotate the piece]
3.) The system checks to see if the placement is legal. [Alt3: Illegal piece placement]
4.) The player sees the piece placed on the game board in the position that they chose [Use case ends].

Alternative Flows:

Alt1: Player chooses to rotate the piece.

1. The piece is updated to appear in its rotated form based on whether clockwise or counterclockwise rotation was chosen and is then displayed to the player.
2. Flow resumes at Main Success Scenario step 2.

Alt2: Player chooses to flip the piece.

1. The piece is updated to appear in its flipped form based on which type of flip was chosen and is then displayed to the player.
2. Flow resumes at Main Success Scenario step 2.

Alt3: Illegal piece placement.

1. The system informs the player that their choice of placement is illegal and asks them to try again.
2. Flow resumes at Main Success Scenario step 2.

Exceptions:

If the player has no available pieces, then the system informs the player and then the use case ends.

Special Requirements:

N/A

Open Issues:

Should the system only show pieces that can currently fit on the board legally or just show all of the player's available pieces?

Use case: Save a game

Primary Actor: Player

Stakeholders and Interests:

Player: Wants the progress of their game to be saved so that they can resume it at another time.

Development team: Wants the system to save the state of the current game so that players can load and resume their game exactly how they left off.

Preconditions:

A game of Blokus has been set up (or loaded) by the system thus making the option to save available to the player.

Postconditions:

Progress of the current game up to when the option of saving was chosen is now made available for the player to load and resume playing at said point.

Main Success Scenario:

1. The player selects the option to save their game.
2. The system records progress of the game at this point (i.e. what pieces have been placed by each player, state of the game board, etc.) and informs the user that the game has been saved successfully.
3. The player can then continue to play or quit the game with their progress intact.

Alternative Flows:

N/A

Exceptions:

If the system is unable to save current progress, inform the player that their game couldn't be saved, and the use case ends.

Special Requirements:

Saving the game should appear to happen instantly for the player.

Open issues:

Should players be given an option of naming their saves?