GameState.CS

Properties:

Serialized baseHandSize : *int*

Number of cards in hand each player starts with and number of cards in hand that each player refills up to.

Serialized initialAttacker : *int*

Number of the player to the right of the first player to attack in a game.

Private currentDefender : *int*

Number of the player that is defending this turn.

Private currentAttacker : *int*

Number of the player that is attacking this turn.

Private trumpSuit : *Suit*

Suit that is the trump suit this game.

Private players : *Player[]*

Array containing all the players in this game.

Private deck : *Deck*

Reference to the deck gameobject.

Private board : *Board*

Reference to the board gameobject.

Private defenseSuccessful : *bool*

Property to keep track whether defense was successful this turn.

Private humanPlayer : *Player*

Reference to which player is not controlled by an AI.

Private endGameHandler : *EndGameHandler*

Reference to the endGameHandler gameobject.

Lifecycle Methods:

Awake:

Initializes the players, deck, board, and endGameHandler properties.

Start:

Runs StartGame to set up initial game state.

Methods:

Private SetTrumpSuit

Parameters:

suit : *Suit*

The Suit that trumpSuit will be set to.

Return: None

Public GetTrumpSuit

Parameters: None

Return: *Suit*

Returns the value of the trumpSuit property.

Private StartGame

Parameters: None

Return: None

Sets up initial game state. Needs to determine trump suit, start the player turn rotation, and initialize player state.

Public TryToEndTurn

Parameters: None

Return: None

Starts Coroutine for ending the turn.

Private WaitForAIToFinishThinking

Parameters: None

Return: None

Coroutine to wait for all AI’s to finish thinking before ending turn.

Private CheckForAIDoneThinking

Parameters: None

Return: *bool*

Returns whether the AI is done thinking.

Private EndTurn

Parameters: None

Return: None

Ends the current turn. If game is not over, sets up game to start next turn. Otherwise starts the end of game process. Needs to initiate card movement based on board state, reset players to a neutral state, and rotate attacker and defender.

Private EndGame

Parameters: None

Return: None

Sends control to the endGameHandler object.

Private CheckIfPlayerWon

Parameters: None

Return: None

Returns whether the non-AI player has won the game.

Private CheckForGameEnd

Parameters: None

Return: *bool*

Returns whether the game has finished.

Game should if human player runs out of cards or if human player is last one with cards.

Private ResetPlayers

Parameters: None

Return: None

Resets players to a neutral state.

Private DealHandsUp

Parameters: None

Return: None

Causes each player to draw cards until they have at least as many cards in hand as the baseHandSize property. Players draw cards one player at a time.

Private CheckForDefenseSuccess

Parameters: None

Return: None

Checks the board state to see if the defending player successfully defended all the attacks.

Private NextPlayer

Parameters:

currentPlayer : *int*

The index of current player

Return: *int*

Returns the index of the next player in the rotation

Private GetNextAttacker

Parameters:

defenseSuccessful : *bool*

Whether or not the defending player successfully defended all the attacks

Return: *int*

Returns the index of the player that will be the attacker next turn.

If the defender was successful, the next player with cards remaining should be the next attacker, otherwise it should skip the defender.

Private GetNextDefender

Parameters: None

Return: *int*

Returns the index of the player who will defend next turn.

This function should skip players who have no cards remaining and any allies of the current attacker.

Public EndTurnChecker

Parameters: None

Return: *bool*

Returns whether all players have ended their turn

Public Static CheckCardDefense

Parameters:

cardInHand : *Card*

Card to be verified if can be played.

cardOnBoard : *Card*

Card on board to be verified against.

Return: *bool*

Returns whether cardInHand can defend against cardOnBoard.

Public GetDefendingPlayer

Parameters: None

Return: *Player*

Returns a reference to the defending player.

Scene Setting

This component assumes that there is a *Board* gameobject, a *Deck* gameobject, and at least 1 *Player* gameobject.