BLACKJACK

The Blackjack game created is a simple implementation of the classic card game using Pygame. It allows the player to play against an AI dealer. The game includes basic functionalities such as drawing cards, hitting or passing, and determining winners.

### **Key Components**

1. **Card Class:**
   * Represents a playing card with attributes for its value, name, suit, and image.
   * **Constructor Parameters:**
     + V (Value): Numeric value of the card (e.g., 2, 3, 10, Ace).
     + N (Name): Human-readable name of the card (e.g., 1 for Ace, 11 for Jack).
     + S (Suit): Suit of the card (e.g., Spades, Hearts).
     + P (Picture): Image associated with the card.
2. **Deck Initialization:**
   * **initializeDeck() Function:** Creates a full deck of 52 cards by combining all possible values and suits.
   * Each card image is loaded and scaled to a standard size.
3. **Card Drawing:**
   * **get\_random\_card() Function:** Randomly selects a card from the deck and removes it from the deck.
   * **player\_draw\_cards() Function:** Adds a card to the player's hand and updates the card’s position on the screen.
   * **AI\_draw\_card() Function:** Adds a card to the AI's hand if its total value is less than 18. The AI also receives hidden cards (face-down).
4. **Card Value Calculation:**
   * **get\_card\_value() Function:** Calculates the total value of a hand, handling Aces specially (they can be worth 1 or 11).
5. **Display Functions:**
   * **display\_card() Function:** Updates the screen with the current state of the player's and AI's hands. The AI's cards may be face down if the reveal flag is not set.
   * **draw\_texts() Function:** Displays game information on the screen, such as hand values, winner messages, and instructions.
6. **Game Loop:**
   * **Main Loop:** Continuously processes user inputs, updates the game state, and redraws the screen.
   * **Event Handling:** Listens for key presses (e.g., [SPACE] to hit, [ENTER] to pass, [ESC] to restart, [TAB] to toggle spectator mode).
   * **Game Logic:** Manages game flow by determining whether the player or AI wins, and handles different game states like busting or tying.

### **Game Flow**

1. **Initialization:**
   * The game initializes Pygame and sets up the game window and assets (card images, fonts).
   * The deck is created and shuffled, and initial hands are dealt to the player and AI.
2. **Player Interaction:**
   * The player can draw cards (hit) or end their turn (pass) using the [SPACE] and [ENTER] keys.
   * The game checks the player's and AI's hands to determine the outcome after each action.
3. **AI Interaction:**
   * The AI draws cards based on its strategy (e.g., drawing until its hand value is at least 18).
   * The AI’s hidden cards are revealed when the game ends or when the spectator mode is enabled.
4. **Winning Conditions:**
   * The game checks for different outcomes: player or AI win, busts, or ties, and displays the result on the screen.
   * The game can be restarted with the [ESC] key.
5. **Spectator Mode:**
   * Spectator mode allows the player to view the AI’s hand value without affecting the game’s outcome. This is toggled with the [TAB] key.

### **Game Termination**

* The game continues until the player quits by closing the window or pressing [ESC] to restart.

Overall, this implementation covers the core aspects of Blackjack, including card handling, game state management, and user interface updates. You can enhance the game further by adding features such as betting, different AI strategies, or more detailed graphics and animations.