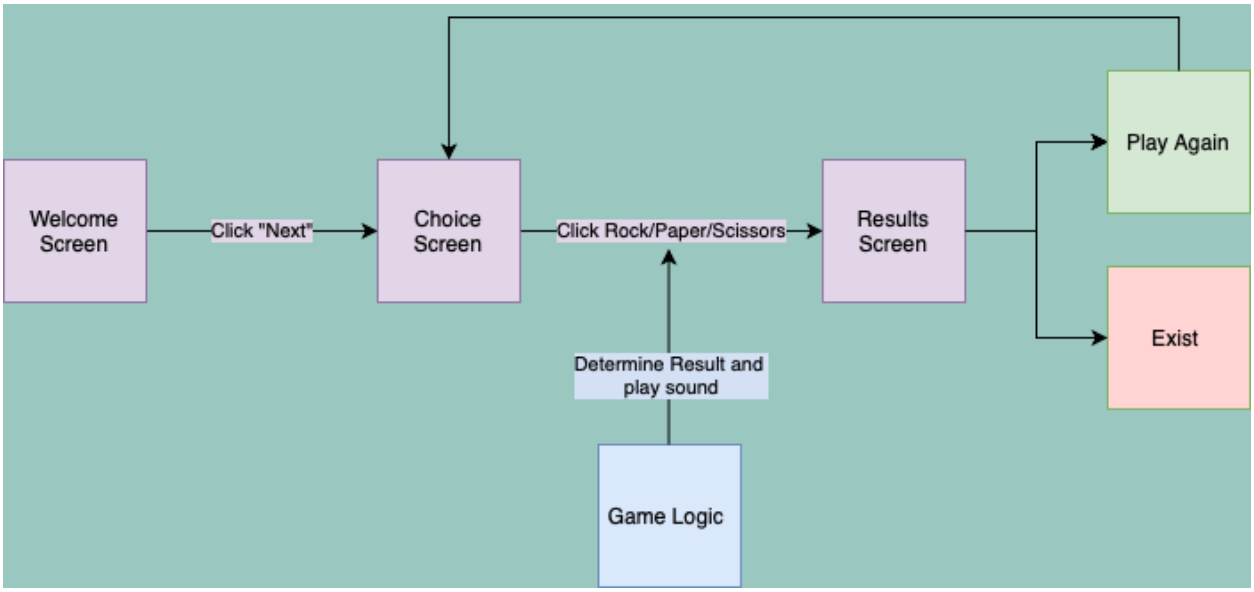
Rock, Paper, Scissors Game - Function Workflow



1. Main Program Flow

- `root.mainloop()`: Starts the Tkinter event loop, which runs the entire GUI application.

The program consists of multiple screens (welcome, choice, result), with functions that manage the transitions between them. Below is the breakdown of the key functions involved.

---

2. Welcome Screen

- `show_welcome_screen()`:
  - **Purpose**: Displays the initial welcome screen when the program starts.
  - **Components**:
    - `Label`: Shows the "Welcome to..." message.

        - `Button`: "Next" button leads the user to the choice screen.
    - **Called By**: The main script at the start.
    - **Next**: When the user clicks "Next", it calls `show_choice_screen()`.

---

3. Choice Screen

- `show_choice_screen()`:
    - **Purpose**: Allows the player to select between rock, paper, or scissors.
    - **Components**:
        - `Label`: Shows the game title and instructions ("Choose your weapon").
        - `Frame`: Contains three buttons for rock, paper, and scissors, each with animated GIFs.
        - **GIF Animation**: The `animate_gif()` function animates the GIFs for rock, paper, and scissors.
    - **GIF Loading**:
        - `load_and_resize_gif_frames()`: This function loads the GIFs and resizes them to 100×100 pixels.
    - **Called By**: `show_welcome_screen()` after clicking "Next".
    - **Next**: When the user clicks a button, the `play_game()` function is called.

---

4. Game Logic

- `play_game(user_choice)`:
    - **Purpose**: Implements the core game logic (deciding winner/loser/tie).
    - **Steps**:
        - The computer makes a random choice between rock, paper, or scissors.
        - Calls `determine_winner()` to decide the result (win, lose, or tie).
        - Plays the corresponding sound effect based on the result (win, lose, tie).

- **Next**: After determining the result, it calls `show_result_screen()` to display the outcome to the player.

---

5. Determine Winner

- `determine_winner(user, computer)`:
  - **Purpose**: Determines the winner of the game based on the player's and computer's choices.
  - **Returns**: A string that indicates the result of the game: `"win"`, `"lose"`, or `"tie"`.
  - **Called By**: `play_game()`.

---

6. Result Screen

- `show_result_screen(result, user_choice, computer_choice)`:
  - **Purpose**: Displays the outcome of the game (win, lose, or tie) and shows the user's choice and the computer's choice.
  - **Components**:
    - `Label`: Displays the result message ("You Win!", "You Lose!", or "It's a Tie!").
    - `Frame`: Contains the user's choice, a "VS" label, and the computer's choice, all with animated GIFs.
    - **GIF Animation**: Uses `animate_gif()` to animate both the user's and computer's choice GIFs.
  - **Next**:
    - "Play Again" button leads back to the choice screen by calling `show_choice_screen()`.
    - "Exit" button terminates the game with `root.quit()`.

---

7. GIF Animation & Resizing

- `load_and_resize_gif_frames(image_path, width, height)`:

- **Purpose**: Loads and resizes all frames of a GIF image to the specified width and height (100×100 in this case).
- **Returns**: A list of resized frames that can be animated.
- **Called By**:
  - `show_choice_screen()` for the player's choices (rock, paper, scissors).
  - `show_result_screen()` for both the player's and computer's selected GIFs.

- `animate_gif(label, gif_frames, delay)`:
  - **Purpose**: Handles the animation of a GIF by cycling through the frames in the `gif_frames` list.
  - **Parameters**:
    - `label`: The Tkinter label that will display the animated GIF.
    - `gif_frames`: The list of frames loaded and resized by `load_and_resize_gif_frames()`.
    - `delay`: The time (in milliseconds) between each frame, controlling the animation speed.
  - **Called By**:
    - `show_choice_screen()` to animate the buttons.
    - `show_result_screen()` to animate the user and computer's choices.

---

8. Sound Effects

- **pygame.mixer.Sound**:
  - The sounds for winning, losing, and tying are loaded using `pygame.mixer.Sound()` and played based on the game result in the `play_game()` function.
  - **Called By**: `play_game()` depending on the outcome of the game.

Summary of Workflow:

1. **Welcome Screen**: Starts with `show_welcome_screen()`, where the user clicks "Next" to start the game.
2. **Choice Screen**: The player selects their weapon on the `show_choice_screen()` using animated buttons (rock, paper, or scissors).
3. **Game Logic**: The player's choice and the computer's random choice are passed to `play_game()` which then calculates the result.
4. **Result Screen**: Displays the result with animated GIFs of the user's and computer's choices on `show_result_screen()`.
5. **Repeat or Exit**: The player can either choose to play again (`show_choice_screen()`) or exit the game.
