Brief Step-by-Step Description of Hangman Pygame Notebook

1. **Imports & Initialization**
- Imports pygame and random.
- Calls pygame.init() and creates the main window with width and height, storing the display surface.

2. **Constants & UI Setup**
- Defines color constants (e.g., BLACK, WHITE, GREEN, etc.).
- Creates fonts for drawing text.
- Sets up variables and images (hangmanPics).

3. **Helper Data Structures**
- `buttons`: Stores the 26 letter buttons as [color, x, y, radius, visible, ascii_code].
- `hangmanPics`: Holds sequence of hangman images.
- `limbs` and `guessed` track incorrect guesses and chosen letters.

4. **Function: redraw_game_window()**
- Clears the screen.
- Draws visible buttons as circles with letters.
- Displays the spaced word using guessed letters.
- Shows hangman image based on wrong guesses.
- Updates the display.

5. **Function: randomWord()**
- Reads words.txt, selects and returns a random word.

6. **Function: hang(guess)**
- Returns True if guess is incorrect, otherwise False.

7. **Function: spacedOut(word, guessed=[])**
- Builds spaced string of word with guessed letters revealed and underscores for unguessed ones.

8. **Function: buttonHit(x, y)**
- Returns ASCII code of letter button clicked, or None if no button hit.

9. **Function: end(winner=False)**
- Displays win/lose message and full word.
- Resets game state (buttons visible, limbs = 0, clears guessed, new word).

10. **Buttons Creation**
- Creates 26 letter buttons positioned in two rows.

11. **Main Game Loop**
- Sets initial word and enters loop.
- Each frame calls redraw_game_window().
- Handles events:
- Quit or ESC exits game.
- Mouse click checks button hit.
- Updates guessed letters and hides button.
- Calls hang() to update limbs if wrong.
- Ends game when player wins or loses.

12. **Shutdown**
- Calls pygame.quit() after loop ends.