Building a Chess Game with Amazon Q CLI: A Fun Journey with AI-Powered Coding

Introduction

As a participant in the AWS "Build Games with Amazon Q CLI" campaign, I set out to create a chess game using Python, Pygame, and the innovative Amazon Q CLI. This campaign, running from May 20 to June 20, 2025, encourages developers in the Asia Pacific, Japan, and Greater China regions to explore AI-assisted coding by building games and sharing their experiences. I'm thrilled to share how I built a fully functional chess game and how Amazon Q CLI transformed my development process. If you're curious about AI-powered coding or want to build your own game, read on! #AmazonQCLI

Getting Started: Setting Up the Environment

After signing up for an AWS Builder ID and joining the AWS Community Discord server, I installed Amazon Q CLI on my machine. Since I'm using Windows, I followed Ricardo Sueiras' guide to set it up via the Windows Subsystem for Linux (WSL), as Amazon Q CLI doesn't yet support native Windows. I also installed Pygame, a popular Python library for game development, using the command pip install pygame==2.5.2. With my environment ready, I was excited to dive into the coding phase with Amazon Q CLI as my AI pair programmer.

Building the Chess Game with Amazon Q CLI

My goal was to create a classic chess game with a graphical interface, complete with standard chess rules, piece movements, and a user-friendly board. I started a chat session with Amazon Q CLI in my terminal, typing:

q /chat "Create a Python chess game using Pygame with an 8x8 board, piece movement validation, and a graphical interface. Include valid move highlighting and turn-based gameplay."

Within seconds, Amazon Q CLI generated a complete code structure, including a ChessPiece class for handling piece logic and a ChessGame class for managing the game loop and UI. The generated code included:

- An 8x8 chessboard with alternating colors (sky blue and red for visual contrast).
- Piece movement rules for pawns, rooks, knights, bishops, queens, and kings.
- A graphical interface using Pygame to display the board and pieces.

- Move validation to ensure only legal moves are allowed.
- Highlighting for valid moves when a piece is selected.

The initial code was a great starting point, but I wanted to enhance it. I prompted Amazon Q CLI again:

q /chat "Add support for loading chess piece images in the chess game and provide a fallback to text representation if images are missing."

Amazon Q CLI updated the code to include image loading for chess pieces (e.g., white_pawn.png, black_king.png) in a pieces directory, with a fallback to text-based piece representation (e.g., ‘P’ for white pawn, ‘p’ for black pawn) if images weren’t available. This made the game more flexible and user-friendly.

I also encountered a bug where the valid move highlights weren’t updating correctly after a piece was moved. I asked Amazon Q CLI for help:

q /chat "Fix the bug where valid move highlights persist after a piece is moved in my chess game."

The tool suggested clearing the selected_piece variable after a valid move, which I implemented in the handle_click method. This iterative process of prompting, refining, and debugging with Amazon Q CLI saved me hours of manual troubleshooting.

Challenges and Solutions

One challenge was ensuring the game followed all chess rules, especially for complex moves like pawn captures and initial two-square pawn advances. The initial code from Amazon Q CLI handled basic pawn movements but missed some edge cases, like diagonal captures. I refined the prompt to include these rules, and Amazon Q CLI updated the ChessPiece.get_valid_moves method to account for pawn captures, making the game more accurate.

Another challenge was setting up the development environment on WSL, as I’m more familiar with native Windows development. Amazon Q CLI guided me through the installation process by providing clear commands and troubleshooting tips when I hit permission issues (e.g., using the --force option for non-root installation). This allowed me to focus on building the game rather than wrestling with setup.

The Final Game

The final chess game is a fully functional, turn-based chess application with the following features:

- **Graphical Board**: An 8x8 board with alternating sky blue and red squares for a vibrant look.
- **Piece Movement**: Accurate movement rules for all chess pieces, including special pawn moves (e.g., two-square advances and diagonal captures).

- **Valid Move Highlighting**: Green circles highlight valid moves when a piece is selected, making gameplay intuitive.
- **Turn-Based Play**: Alternates between white and black players, enforcing standard chess turn rules.
- **Image Support**: Loads custom piece images if available, with a text-based fallback for accessibility.

You can check out the complete code in my GitHub repository: [insert your GitHub repo link here]. The game is simple yet polished, and I'm proud of what I accomplished in just a few days with Amazon Q CLI's help.

Why Amazon Q CLI is a Game-Changer

Amazon Q CLI made this project a breeze by:

- **Rapid Prototyping**: Generating a working chess game from a single prompt in seconds.
- **Iterative Refinement**: Allowing me to tweak features like image support and bug fixes with natural language prompts.
- **Debugging Support**: Helping me identify and resolve issues like the move highlight bug without diving deep into the code.
- **Learning Opportunity**: Teaching me Pygame concepts and chess logic as I reviewed and customized the generated code.

As someone who loves coding but isn't a game development expert, Amazon Q CLI felt like having a senior developer by my side, guiding me through each step. It's an incredible tool for both beginners and experienced developers looking to prototype ideas quickly.