Chess Game

OOP Python Application

A two-player CLI & GUI chess game

Developer: Eyal Grinberg

Motivation & Problem Statement

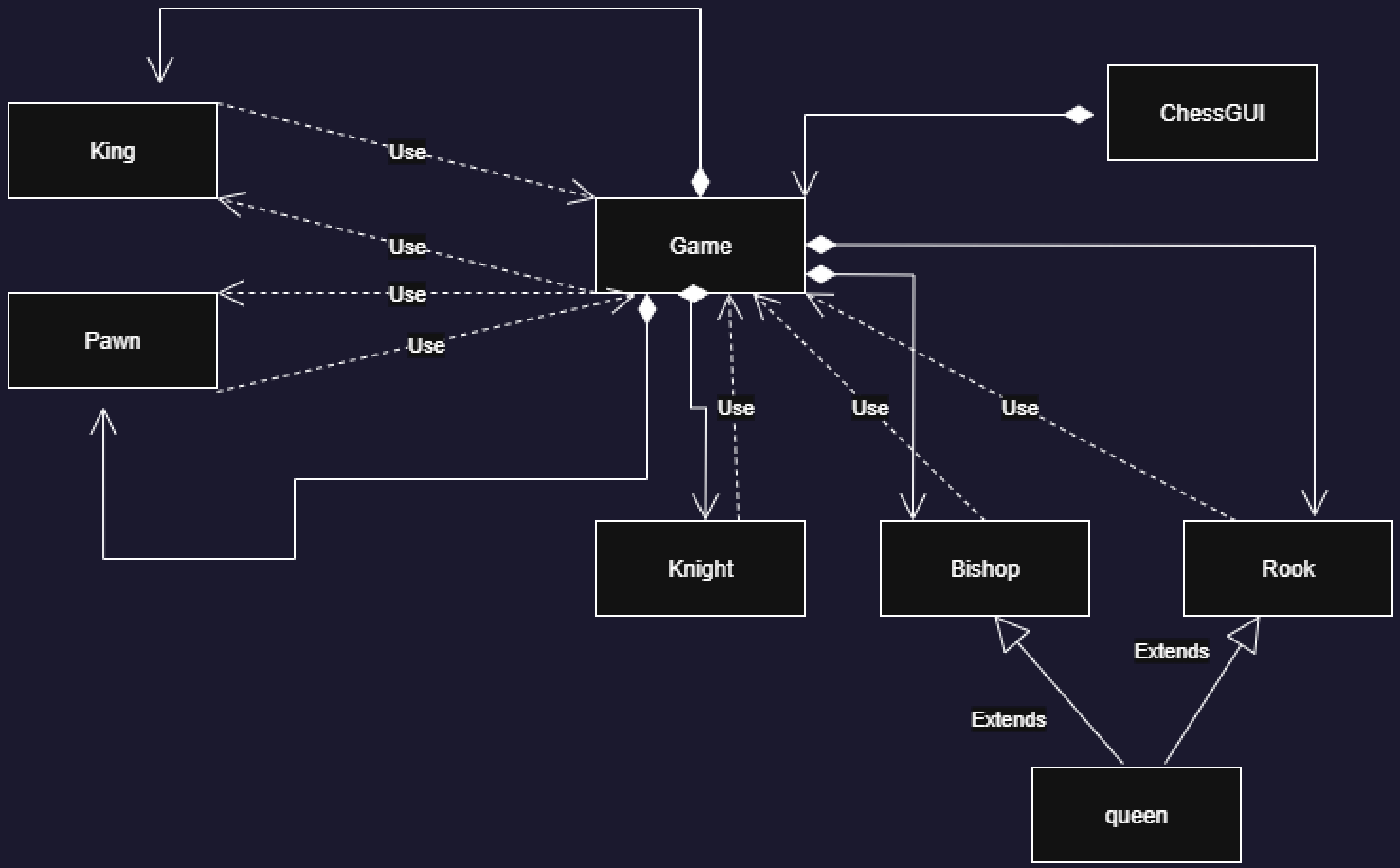
Goal - Build a complete chess game application with:

- Accurate move validation and rule enforcement
- Interactive GUI and testable CLI
- Have fun

Motivation:

- Practice end-to-end software engineering from backend to UI
- Apply OOP and system design principles in a real-world game

Architecture



Features

CLI VERSION FOR LOGIC TESTING AND DEBUGGING – INCLUDING A TESTING FUNCTION

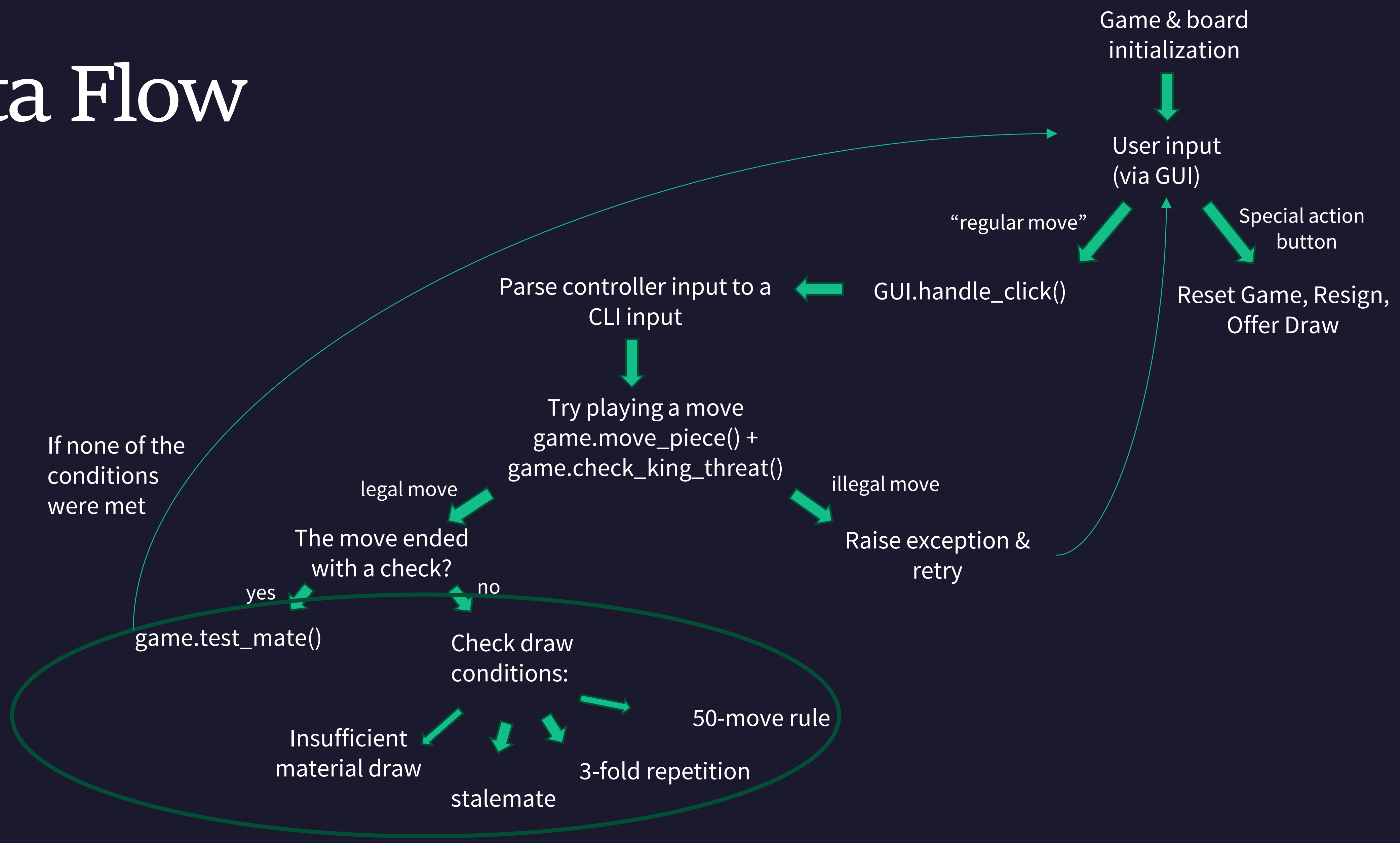
GUI VERSION USING TKINTER: INTERACTIVE BOARD, STATUS UPDATES, ADDITIONAL ACTION BUTTONS

MOVE VALIDATION BASED ON SIMULATIONS (CHECK, MATE, STALEMATE DETECTION, CASTLING LOGIC, PINS, ETC.)

ERROR HANDLING AND INFORMATIVE EXCEPTIONS USAGE

DATA FLOW SEPARATES UI FROM LOGIC FOR MODULARITY

Data Flow



Game & board initialization

User input (via GUI)

“regular move”

Special action button

GUI.handle_click()

Reset Game, Resign, Offer Draw

Parse controller input to a CLI input

Try playing a move game.move_piece() + game.check_king_threat()

legal move

illegal move

Raise exception & retry

The move ended with a check?

yes

no

game.test_mate()

Check draw conditions:

Insufficient material draw

stalemate

3-fold repetition

50-move rule

If none of the conditions were met

Challenges

DESIGNING TESTABLE COMPONENTS WHILE BUILDING INTERACTIVELY

COMPLEX GAME LOGIC ALGORITHMS WITHOUT RELYING ON EXTERNAL LIBRARIES OR TOOLS

BUILDING WITHOUT CODE ASSISTANTS OR INTERNET ACCESS AT TIMES

Results & Future work

✓ Outcomes:

Fully working chess game: all rules supported

CLI and GUI modes for flexibility

Clear architecture and testability

Future Features

- Evaluation bar
- AI opponent (RL)

Demo