Tatedrez

1 - Code architecture

Code architecture follows a MVC pattern based on Entities. It's splitted in three components in order to keep isolated and centralized pure game logic (Core) from visual elements and player interactions (Client).

- Config (Model)
- Core (Controller)
- Client (View)

This code architecture enforces the single-responsibility and code reutilization concepts simplifying code scalability and maintenance. The Config component could be extracted to be stored in cloud databases or the Core component to be run as a remote server.

1.1 - Architecture components

→ Config ← Core ← Client →

1.1.1 - Config

Config defines the game data structure using Scriptable Object classes.

1.1.2 - Core

Core defines builders and entities. Builders use config data to populate entities, and those run the game core logic.

Builder	Entity	Properties	Functions	Events
Match Builder	Match	Players Current Player Local Player	Launch Request Movement	Start Switch Turn End
Player Builder	Player	Id Name Pieces	Open Turn	-
Board Builder	Board	Size	Get Location Data Try Locate	-
Piece Builder	Piece	Id Owner Id Location	Get Valid Moves	Locate

1.1.3 - Client

Client defines visual elements, manages player interactions and keeps updated using an observer pattern.

1.2 - Code structure

Context	Config	Core	Client
Game	Game Config	-	Game Manager
Match	Match Config	Match	Match Manager
Player	Player Config	Human Player (Player)	-
		AI Player (Player)	
Board	Board Config	Board	Board Behaviour
			Match Piece Selector
Piece	Piece Config	Knight (Strategy)	Piece Behaviour
		Rook (Strategy)	
		Bishop (Strategy)	
UI	-	-	UI Manager
			Main Menu Screen
			Match Screen
			Match Result Popup
Input	-	-	Input Manager
			Input Controls
			Match Input Handler

1.2.1 Plugins

- Input System
- TextMesh Pro
- DoTween

2 - Use cases

2.1 - Match Workflow

1. [Client] Main Menu Screen - Click Start Button
2. [Client] Match Manager - Request Start
3. [Core] Match Builder - Build
4. [Core] Match - Start
5. *Game Match Loop*
6. [Core] Match - End
7. [Client] Match Manager - Display Result
8. [Client] Match Result Popup - Confirm

2.2 - Game Match Loop

1. [Core] Player - Open Turn
2. *Player interactions / AI Player interactions*
3. [Core] Player - Close Turn
4. [Core] Match - Check Player Victory

2.3 - UI Navigation

→ Main Menu Screen → Match Screen → Match Result Popup →

3 - Scalability

- User Interface
  - UI Manager allows the implementation of new UI components such as Screens and Popups easily.
- Content
  - New pieces and testing players can be added easily
- Cloud database:
  - Game Data can be moved to cloud storages
- Multiplayer:
  - Architecture allows the implementation of an entity based network layer.

4 - Design suggestions

1. Knight at center position is provoking infinite turn loops in some cases. Alternatives:
    - Prohibit move Knight to center
    - End the game as ‘Draw’ after a number of looping turns

2. White player starts
    - Since it includes chess pieces, it would be coherent to start with whites.
3. Turn timeout