TIC TAC TOE mini-game by Oluwatola Joshua

I chose tic tac toe because it brought back memories of games I used to play with friends when I was young. Wanted to give it a try.

Feature added: Instead of just making the AI smarter with increased difficulty, I modified the gameplay to increase the grid size according to difficulty selected (easy – 3x3, medium – 4x4, and hard – 5x5) – while still making the bot smarter as the difficulty is increased.

Technical approach and architecture decisions: I utilized a manager-based architecture with *BoardManager* and *TurnManager* as singletons to handle the core game state, keeping core logic and UI separate.

- Implemented dynamic grid generation: Instead of creating separate grid objects for each difficulty, I implemented a system where the *BoardManager* generates Cell data objects at runtime. The UI (*UIBehavior*) listens for this and dynamically instantiates the cell button prefab into a *GridLayoutGroup*, setting the column constraint to match the grid size.
- Used Event-Driven Communication: I used C# Events so script components don't need direct references to each other.
- Scalable Algorithm: The win-checking logic is generic, rather than using hardcoded arrays, enabling the same logic to handle any board size.

Bonus feature implementation explanation: Went with option A (UI Recreation & Polish). Recreated the core UI screens and flow. Added animation feedback for turns, wins and losses. Added slider handle breathing effect, rotating stars (on win panel), etc. Design should respond well with different resolutions that have approximately 9:16 aspect ratio (at the moment). All UI elements in the project are fully functional.

Generative AI Tool Usage: Used Gemini (nano-banana) to recreate the UI elements, Gemini 3 Pro and Claude Sonnet to refine difficulty implementation (extend grids), the simple bot AI, and some minor help here and there. AI really sped things up and made it possible to make a quick prototype with fair UI without purchasing asset packs or working with a designer.

Challenges faced and solutions implemented:

- Scaling to larger grids: The original tic tac toe tutorial relied on static assets for cells, which wouldn't work for 4x4 or 5x5 grids without manually

creating dozens of cell data objects (scriptable objects) and manually managing grids. Solution: Code was refactored to use *ScriptableObject.CreateInstance* to generate cells data objects in memory, making the system scalable.

- AI Balancing: The AI was unbeatable in medium and hard mode (4x4 & 5x5) making it frustrating to play against. Solution: added an RNG system so AI sometimes doesn't do the logical thing.

With more time, I would research how screen transition animations are done and implement it to improve game feel, replicate the tutorial screen, and try to adopt a standard algorithm for tic tac toe bot playing (like minimax).