6. Performance Measurement and Optimization:

Metric	Ai_easy	Ai_medium	Ai_hard	Two player
Initialization Timing(sec)	2.6 $*10^{-4}$	3.7$*10^{-4}$	2.7$*10^{-4}$	2.5$*10^{-4}$
Player Move Timing average(sec)	1.1$*10^{-3}$	7.79	0.51	3.4$*10^{-3}$
AI Move Timing average(sec)	1.1$*10^{-4}$	3.7$*10^{-2}$	5.9$*10^{-3}$	---------
Game Status Check Timing average(nsec)	2900	1725	1884	3200
Game Reset Timing average(sec)	3.2$*10^{-5}$	4.9$*10^{-6}$	1.7 $*10^{-5}$	2.9$*10^{-5}$
Game History Saving Timing(sec)	1.1$*10^{-3}$	9.7$*10^{-4}$	9.7$*10^{-4}$	1.1$*10^{-3}$
Finding Best Move Timing average	900 nsec	3.1$*10^{-2}$sec	5.7$*10^{-3}$sec	--------

Optimization:

- Use Arrays Instead of Vectors: using arrays can be faster than vectors due to lower overhead.
- Minimize Redundant Operations: Optimize Check Winner and isBoardFull Functions O(N) as use one for loop.
- Optimizing AI Move Calculation: The AI uses the minimax algorithm with alpha-beta pruning to determine the best move.
- Code Refactoring: Separate concerns such as UI updates, game logic, and AI decision-making into different functions or classes.
- Optimizing Memory Usage:377.5MB
- CPU Utilization: 0.1%