Tic-Tac-Toe AI Battle

An advanced Tic-Tac-Toe game featuring sophisticated AI algorithms, real-time performance analysis, and a beautiful modern interface. Built with Angular 19 and Node.js, this project demonstrates the power of game theory algorithms including Minimax and Alpha-Beta pruning.

Features

Game Modes

- **Human vs Human**: Classic two-player mode
- **Human vs AI**: Challenge sophisticated AI opponents
- **AI vs AI**: Watch two AIs battle with different algorithms

AI Algorithms

- **Minimax Algorithm**: Classic game tree search algorithm
- **Alpha-Beta Pruning**: Optimized minimax with branch elimination
- **Real-time Performance Comparison**: See the efficiency gains of pruning
- **Algorithm Switching**: Choose different algorithms for each AI player

Advanced Analytics

- **Move Timers**: Track thinking time for each player
- **Performance Metrics**: Nodes explored, execution time, and efficiency ratings
- **Algorithm Comparison Tool**: Side-by-side performance analysis
- **Game Statistics**: Win/loss records and move history

Modern Interface

- **Responsive Design**: Works on desktop, tablet, and mobile
- **Beautiful Animations**: Smooth transitions and visual feedback
- **Real-time Updates**: Live performance metrics and game state
- **Accessibility**: Keyboard navigation and screen reader support

Quick Start

System Setup

Automated Installation (Recommended)

We provide automated installation scripts to ensure you have the correct Node.js version:

For Linux/WSL:

```
chmod +x install.sh
./install.sh
```

For Windows:

```
# Run as Administrator
install.bat
```

These scripts will automatically detect your system and install Node.js v20+ and npm.

Install pnpm (Recommended Package Manager):

```
npm install -g pnpm
```

Why pnpm? pnpm is faster, more disk-efficient, and creates a stricter, more reliable dependency resolution than npm. It creates hard links to shared dependencies, saving disk space and improving installation speed.

Prerequisites

- **Node.js** (v18 or higher) - *Use our install scripts above if needed*
- **pnpm** (recommended) or **npm** (v8 or higher) - *pnpm is faster and more efficient*

Installation

1. **Install dependencies**

   ```
   # With pnpm (recommended)
   pnpm install

   # Or with npm
   npm install
   ```

2. **Start the development server**

   ```
   # With pnpm (recommended)
   pnpm start

   # Or with npm
   npm start
   ```

3. **Open your browser** Navigate to `http://localhost:4200`

The application will automatically reload when you make changes to the source files.

Development Commands

Command	pnpm (Recommended)	npm Alternative	Description
Start dev server	`pnpm start`	`npm start`	Start development server on port 4200
Build production	`pnpm run build`	`npm run build`	Build the project for production
Run tests	`pnpm test`	`npm test`	Run unit tests
Watch tests	`pnpm run test:watch`	`npm run test:watch`	Run tests in watch mode
Run linter	`pnpm run lint`	`npm run lint`	Run linting checks
Serve SSR	`pnpm run serve:ssr`	`npm run serve:ssr`	Serve server-side rendered version

Project Structure

```
src/
|-- app/
|   |-- game.service.ts              # Core game logic and API service
|   |-- timer.service.ts             # Player timing service
|   |-- tic-tac-toe/
|   |   |-- tic-tac-toe.component.*  # Main game component
|   |   |-- components/              # Modular sub-components
|   |       |-- game-board/          # 3x3 game grid
|   |       |-- game-mode-selector/  # Game mode selection
```

```
|   |       |-- player-controls/     # AI/Human controls
|   |       |-- move-timer/          # Timing display
|   |       |-- game-stats/          # Statistics panel
|   |       |-- algorithm-comparison/# Performance comparison
|   |       |-- ai-performance/      # AI metrics display
|   |       |-- move-history/        # Game move tracking
|   |       |-- help-modal/          # Tutorial and help
|-- server.ts                        # Express.js backend server
|-- styles.scss                     # Global styles
```

How to Play

Basic Gameplay

1. **Choose Game Mode**: Select Human vs Human, Human vs AI, or AI vs AI
2. **Configure Players**: Set each player as Human or AI
3. **Select AI Algorithm**: Choose Minimax or Alpha-Beta for AI players
4. **Make Moves**: Click on empty squares to place your mark
5. **Win Condition**: Get three in a row (horizontal, vertical, or diagonal)

AI Features

- **Manual AI Moves**: Force an AI to move using the "Make AI Move" button
- **Algorithm Comparison**: Use the comparison tool to analyze performance
- **Performance Metrics**: View detailed AI decision-making statistics
- **Real-time Analysis**: See nodes explored and execution time for each move

API Endpoints

The backend provides RESTful endpoints for game management:

Endpoint	Method	Description
`/api/game/state`	GET	Get current game state
`/api/game/move`	POST	Make a human move
`/api/game/ai-move`	POST	Request AI move
`/api/game/reset`	POST	Reset the game
`/api/game/mode`	POST	Change game mode
`/api/game/player-control`	POST	Switch player control
`/api/game/player-algorithm`	POST	Change AI algorithm
`/api/game/compare-algorithms`	POST	Compare algorithm performance
`/api/game/stats`	GET	Get game statistics

Algorithm Details

Minimax Algorithm

- **Purpose**: Finds the optimal move by exploring all possible game states
- **Complexity**: O(b^d) where b is branching factor and d is depth
- **Characteristics**: Guarantees optimal play but can be slow

Alpha-Beta Pruning

- **Purpose**: Optimized minimax that eliminates unnecessary branches
- **Optimization**: Typically explores 40-60% fewer nodes
- **Efficiency**: Same optimal results with significantly better performance

Performance Metrics

- **Nodes Explored**: Number of game states evaluated
- **Execution Time**: Time taken to find the best move
- **Pruning Efficiency**: Percentage of nodes eliminated by alpha-beta
- **Decision Quality**: Comparison of move selection between algorithms

Component Architecture

The application follows a modular architecture with well-defined responsibilities:

Core Components

- **GameBoardComponent**: Manages the 3x3 grid and move interactions
- **PlayerControlsComponent**: Handles AI/human switching and algorithm selection
- **MoveTimerComponent**: Displays timing information and AI performance metrics
- **GameStatsComponent**: Shows win/loss statistics and game history

Analysis Components

- **AlgorithmComparisonComponent**: Provides side-by-side performance analysis
- **AIPerformanceComponent**: Displays detailed AI decision metrics
- **MoveHistoryComponent**: Tracks and displays game move sequence

UI Components

- **GameModeSelectorComponent**: Game mode selection interface
- **HelpModalComponent**: Tutorial and feature documentation

Styling and Design

- **TailwindCSS**: Utility-first CSS framework for rapid styling
- **Responsive Grid**: CSS Grid and Flexbox for responsive layouts
- **Dark Theme**: Modern dark interface with glassmorphism effects
- **Animations**: Smooth transitions and loading states
- **Visual Feedback**: Hover states, win highlighting, and move indicators

Testing

The project includes comprehensive unit tests for all components:

```
# Run all tests (pnpm recommended)
pnpm test
# npm test

# Run tests in watch mode (pnpm recommended)
pnpm run test:watch
# npm run test:watch

# Run tests with coverage (pnpm recommended)
pnpm run test:coverage
# npm run test:coverage
```

Performance Optimization

Frontend Optimizations

- **Lazy Loading**: Components loaded on demand
- **OnPush Change Detection**: Optimized Angular change detection
- **Service Worker**: Caching for offline functionality
- **Bundle Optimization**: Tree-shaking and code splitting

Backend Optimizations

- **Algorithm Efficiency**: Alpha-beta pruning for faster AI decisions
- **Memory Management**: Efficient game state handling
- **Caching**: Static asset caching and compression
- **Error Handling**: Graceful degradation and recovery

Requirements

System Requirements

- **Node.js**: v18.0.0 or higher
- **pnpm**: latest version (recommended) or **npm**: v8.0.0 or higher
- **Modern Browser**: Chrome, Firefox, Safari, or Edge (last 2 versions)

Browser Compatibility

- Chrome 90+
- Firefox 88+
- Safari 14+
- Edge 90+
- Mobile browsers (iOS Safari, Chrome Mobile)

Troubleshooting

Common Issues

Port 4200 already in use

```
# Kill existing process
lsof -ti:4200 | xargs kill -9
# Or use different port
ng serve --port 4201
```

Build errors after updates

```
# Clear node modules and reinstall
rm -rf node_modules package-lock.json pnpm-lock.yaml
# With pnpm (recommended)
pnpm install
# Or with npm
npm install
```

Tests failing

```
# Clear Angular cache
ng cache clean
# With pnpm (recommended)
pnpm test
```

```
# Or with npm
npm test
```

License

This project is licensed under the MIT License - see the LICENSE file for details.