

Chess Engine & AI

A fully-featured chess engine, game, and AI opponents.

V 1.3.5

- Incomplete documentation parts will be improved over time.
- Get the most up to date documentation by [clicking here](#).
- If you have any questions or need assistance email support at intuitivegamingsolutions@gmail.com.

Table of Contents

1. [Table Of Contents](#)
2. [API References](#)
 - Useful documentation for coders.
 - 2.a. [API Reference: ChessEngine](#) ([Google Drive](#))
 - 2.b. [API Reference: ChessEngine.AI](#) ([Google Drive](#))
3. [Getting Started](#)
 - Simply import the package and you are ready to go!
 - 3.a. [Importing the Asset](#)
 - 3.b. [Included Demos](#)
4. [The 'Undo & Redo' System](#)
 - 4.a. [The ChessUndoManager Component](#)
 - 4.b. [Using Editor Events To Drive UI](#)
5. [Unity Components](#)
 - 5.a. [The VisualChessPiece Component](#)
 - A component that handles the visual representation of the underlying chess engine [ChessPiece](#) object.
 - 5.b. [The VisualChessTable Component](#)
 - A component that handles the visual representation of the underlying chess engine [ChessTable](#) object.
 - 5.c. [The VisualChessTableTile Component](#)
 - A component that handles the visual representation of the underlying chess engine [ChessTableTile](#) object.
 - 5.d. [The RotateChessPieceByColor Component](#)
 - Allows chess pieces to be rotated when created based on their color.
 - 5.e. [The ChessGameManager Component](#)
 - The base component for all chess game managers. Manages 'local play' only.
 - 7.e.i. [The ChessAIGameManager Component](#)
 - Manages chess games that involve AI opponents.
 - 7.e.ii. [The NetworkChessGameManager Component](#)
 - Manages network (multiplayer) chess matches.
 - 5.f. [The Clicker Component](#)
 - Manages cursor and touch based 'click' inputs.
 - 5.g. [The Clickable Component](#)
 - Allows a 'Clicker' to fire some events after 'clicking' on the Clickable.
6. [AI](#)
 - 6.a. [Included AI Opponents](#)

- 6.a.i. [Doofus AI Opponent](#)
 - An aggressive and simple chess AI that does not think about the consequences of its moves.
- 6.a.ii. [Guy AI Opponent](#)
 - An advanced chess AI that utilizes all provided think time, think depth, and threads to score moves to make intelligent plays. Heavily prioritizes checkmate and will prioritize position over taking pieces.
- 6.a.iii. [Riddle AI Opponent](#)
 - An advanced chess AI that utilizes all provided think time, think depth, and threads to score moves to make intelligent plays. Balanced approach between chasing checkmates and capturing pieces.
- 6.b. [Configuring AI Difficulty](#)
- 6.c. [Useful AI Settings](#)
- 6.d. [Making Custom AI Opponents](#)
- 7. [UI Overview](#)
 - 7.a. [Turn Indicator UI](#)
 - 7.b. [Game Over UI](#)
 - 7.c. [Player Indicator UI](#)
 - 7.d. [Undo & Redo Button UI](#)
- 8. [Extensions](#)
 - 8.a. [Extension: Main Menu System](#)
 - Adds a main menu to the provided chess game.
 - 8.b. [Extension: Multiplayer Netcode Demo](#)
 - Adds multiplayer support to the chess game.
- 9. [FAQ](#)

API References

2.a. API Reference: ChessEngine

- The scripting reference for the Chess Engine
- [Google Drive: ChessEngine API Reference](#)

2.b. API Reference: ChessEngine.AI

- The scripting reference for the Chess Engine AI.
- [Google Drive: ChessEngine.AI API Reference](#)

Getting Started

3.a. Importing the Asset

There are 2 ways to import the 'Chess Engine & AI: Doofus' package.

- a. (Recommended) Using the Unity Editor 'Package Manager'.
 - i. Open the Windows→Package Manager using the Unity editor toolbar.
 - ii. In the upper-left corner of the Package Manager window select 'Packages: My Assets'.
 - iii. Search for 'Chess Engine & AI: Doofus' in the list or use the search bar in the window.
 - iv. Select the asset in the package manager, select 'Download'.
 - v. After the package has finished downloading click 'Import' to import it into the project.
- b. Importing ChessEngineAndAI.unitypackage
 - i. Using the Unity Editor's toolbar select Assets→Import Package
 - ii. In the file explorer that opens navigate to ChessEngineAndAI.unitypackage
 - iii. Double click the package and import it.

3.b. Included Demos

- **ChessDemo_DoofusAI** - a demo showing the chess AI in action.
 - Play against Doofus or pit him against himself or other AIs head-to-head.
- **ChessDemo_FENStateLoad** - a demo showing the loading of FEN string states.
 - Load any FEN string and start the game from that state.
- **ChessDemo_LocalPlay** - single device local play.
 - Play against yourself or your friends locally on 1 device.
- **ChessDemo_Multiplayer** - connect 2 devices to play head-to-head over the internet.
 - Play online over the internet between 2 application instances using Unity's Netcode. *(Make sure to import Game/Extension_Netcode_Multiplayer to import the multiplayer demo into your project.)*
 - Requires you to have Unity's 'Netcode for GameObjects' package installed.
[\(Instructions\)](#)

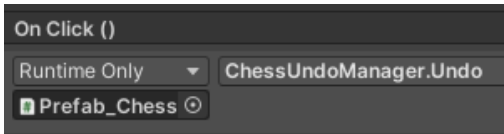
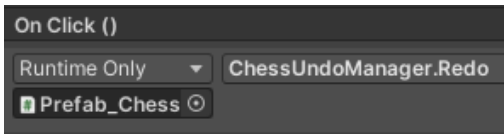
The 'Undo & Redo' System

4.a. The ChessUndoManager Component

- The ChessUndoManager component is intended to be attached to the same GameObject as a ChessGameManager component (*or derived type*) that then allows you to easily undo and redo chess moves in a game.
- The ChessUndoManager component provides the following public C# methods for managing history:
 - void ChessUndoManager.Undo()
 - Undoes the previous move (if there is one).
 - Note that if a move is made after an undo that does not match the top-most move in the 'redo' stack then the 'redo' stack is automatically cleared.
 - void ChessUndoManager.Redo()
 - Redoes the previously undone move (if there is one).

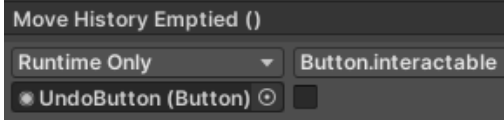
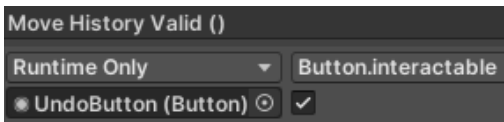
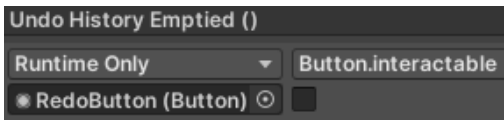
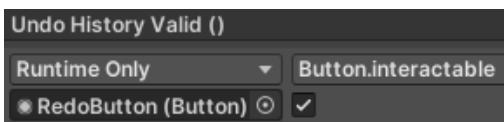
4.b. Using Editor Events To Drive UI

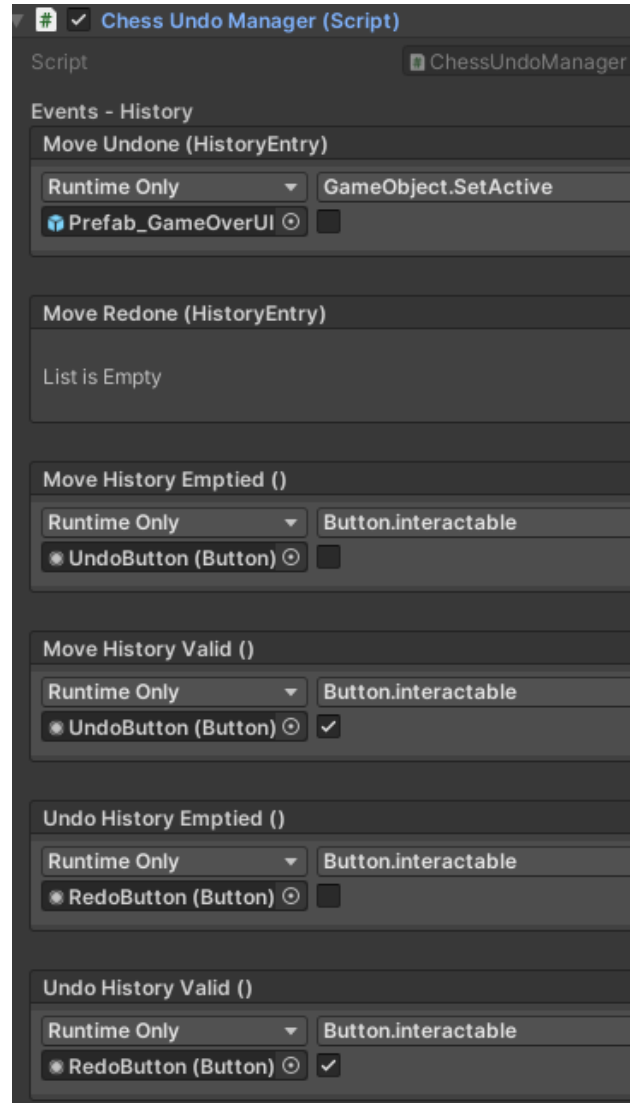
- One of the most useful things about Unity editor events in general is that they are a convenient and simple way to drive UI. In the demo scenes, and in examples below, you will see how these events can be used to easily drive UI elements in your game.
- You will in general need to use Unity editor events to perform the following 2 functions to integrate undo & redo functionality in your game:
 1. As shown in the screenshots below you will likely want to use two events to invoke the actual **Undo()** and **Redo()** methods from the **ChessUndoManager** components. In the demo the '**UndoButton**' and '**RedoButton**' Buttons 'OnClick ()' editor events are used.

Event	Action
(UndoButton) Button.OnClick ()	
(RedoButton) Button.OnClick ()	

2. As shown in the screenshot (described by the table) below you will likely want the 'undo' button to be non-interactable while there is no valid move history so you will want to configure the following events:

Event	Action
Move Undone (HistoryEntry) <i>This event is invoked when a move is undone.</i>	 <p><i>This will make it so the game over UI is no longer visible after 'undoing' a game ending move.</i></p>
Move Redone (HistoryEntry) <i>This event is invoked when an undone move is redone.</i>	 <p><i>The demo does not make use of this event.</i></p>

<p>Move History Emptied ()</p> <p><i>This event is invoked when the 'move history' stack becomes empty.</i></p>	 <p><i>This will make it so the undo button is not interactable when the 'move history' stack is empty.</i></p>
<p>Move History Valid ()</p> <p><i>This event is invoked when the 'move history' stack becomes non-empty after being empty.</i></p>	 <p><i>This will make it so the undo button is interactable when the 'move history' stack becomes non-empty after being empty.</i></p>
<p>Undo History Emptied ()</p> <p><i>This event is invoked when the 'undo history' stack becomes empty.</i></p>	 <p><i>This will make it so the redo button UI is no longer interactable after the 'undo history' stack becomes empty.</i></p>
<p>Undo History Valid()</p> <p><i>This event is invoked when the 'undo history' stack becomes non-empty after being empty.</i></p>	 <p><i>This will make it so the redo button UI is interactable after the 'undo history' stack becomes non-empty after being empty.</i></p>



A screenshot showing the recommended event setup in the 'Inspector' pane of a ChessUndoManager component.

Unity Components

5.a. The VisualChessPiece Component

- A component that handles the visual representation of the underlying chess engine [ChessPiece](#) object.

5.b. The VisualChessTable Component

- A component that handles the visual representation of the underlying chess engine [ChessTable](#) object.

5.c. The VisualChessTableTile Component

- A component that handles the visual representation of the underlying chess engine `ChessTableTile` object.

5.d. The RotateChessPieceByColor Component

- A component that allows a number of degrees of rotation to be applied to a [VisualChessPiece](#) (on the same [GameObject](#)) around some local space axis when it is first spawned based on the team it is on.

5.e. The ChessGameManager Component

- The base component for all chess game managers.
- Manages 'local play' only.
- May be inherited from to create 'derived' chess game managers.

5.e.i. The ChessAIGameManager Component

- Manages chess matches that involve AI opponent(s).

5.e.ii. The NetworkChesGameManager Component

- Manages networked (multiplayer) chess matches.

5.f. The Clicker Component

- Manages cursor and touch-based 'click' inputs.
- Fires Unity events when a Clicker 'clicks' on a Clickable.
- Used to manage user inputs.

5.g. The Clickable Component

- Allows a [Clicker](#) to fire some events after 'clicking' on the [Clickable](#).
- Used to interpret user inputs to chess game moves.

AI

6.a. Included AI Opponents

6.a.i. Doofus AI Opponent

- An aggressive and simple chess AI that does not think about the consequences of its moves.
- [Documentation](#) ([Google Drive](#))
- [API Reference: ChessEngine.AI.Guy](#) ([Google Drive](#))

6.a.ii. Guy AI Opponent

- An advanced chess AI that utilizes all provided think time, think depth, and all threads to quickly score moves and make intelligent plays.
- Guy has scalable difficulty. Altering its 'max think time', 'max think depth', or maximum threads (assuming hardware allows) will change the difficulty and behavior of the AI opponent. See the '[Configuring AI Difficulty](#)' section of the documentation for more information.
- Guy will prefer to chase a potential checkmate as opposed to taking an opponent's piece.
- [Documentation](#) ([Google Drive](#))
- [API Reference: ChessEngine.AI.Guy](#) ([Google Drive](#))

6.a.iii. Riddle AI Opponent

- An advanced chess AI that utilizes all provided think time, think depth, and all threads to quickly score moves and make intelligent plays.
- Riddle has scalable difficulty. Altering its 'max think time', 'max think depth', or maximum threads (assuming hardware allows) will change the difficulty and behavior of the AI opponent. See the '[Configuring AI Difficulty](#)' section of the documentation for more information.
- Riddle will prefer to take valuable pieces as opposed to chasing a potential checkmate.
- [Documentation](#) ([Google Drive](#))
- [API Reference: ChessEngine.AI.Riddle](#) ([Google Drive](#))

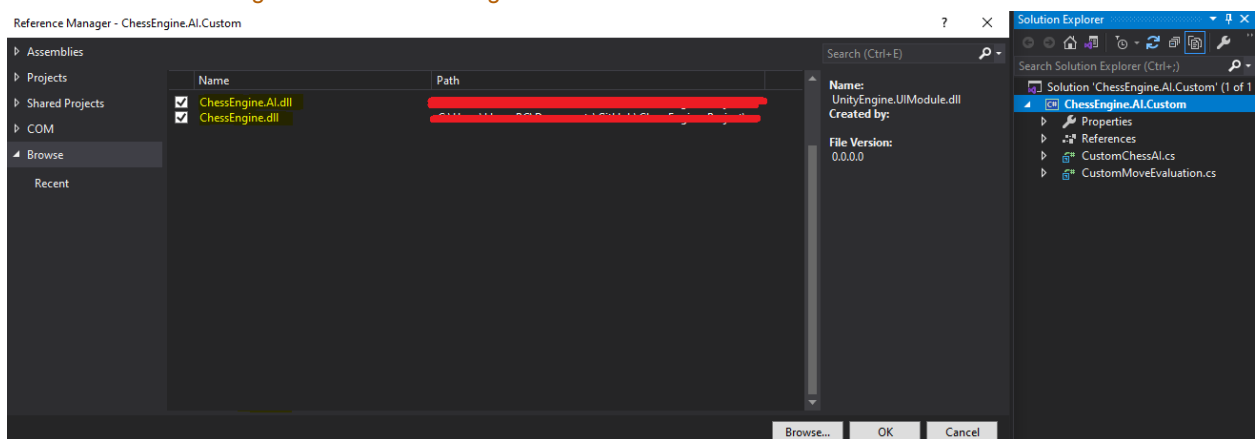
6.b. Configuring AI Difficulty

- *Improved documentation coming soon.*
- AI difficulty can be adjusted by modifying the following settings:
 - Max Depth: the maximum number of moves ahead the AI may consider.
 - Max Time: the number of seconds the AI may 'think' for.
 - *Max Threads: the number of threads (at most) the AI may 'think' with.*
 - *Max threads only exists in some AI opponents.*
- When using the included [ChessGameManager](#) (or derived) components you can adjust 'Max Depth' and 'Max Time' in the components 'Inspector' pane in the Unity Editor.

6.c. Useful AI Settings

6.d. Making Custom AI Opponents

- To make a custom AI opponent easily you should have Visual Studio 2019 or newer installed, otherwise you may have to make your .NET Library project in your respective IDE yourself.
- 1. In order to make a custom AI opponent you will first want to download the '[Custom AI Toolkit](https://drive.google.com/file/d/1ft06Oa344Rd8NkdS9DrqihkDYDHYciOW/view?usp=sharing)' from here: <https://drive.google.com/file/d/1ft06Oa344Rd8NkdS9DrqihkDYDHYciOW/view?usp=sharing>
 - The **password to unzip** the AI toolkit can be found in the '[ChessEngineAndAI/Documentation/Password_For_CustomAIToolkit.txt](#)' file.
- 2. After downloading the '[Custom AI Toolkit](#)' you may want to rename the AI. (**Optional**)
 - a. First, open all files and replace all occurrences (case-sensitive) of 'Custom' with 'MyAIName' (whatever AI name you want) and replace all occurrences of 'custom' with 'myAIName' (again, whatever lower-case AI name you want).
 - b. Next, rename any file or folder with '.Custom' to '.MyAIName' (again, whatever AI name you want).
- 3. Open the Visual Studio project (or your custom project if not using visual studio) and add project library references to '[ChessEngine.dll](#)' and '[ChessEngine.AI.dll](#)'.



4. Build your managed dll for your new AI, this will be named (unless you changed the project) something like '[ChessEngine.AI.Custom.dll](#)'. Place this anywhere in your project's '[Assets](#)' folder.

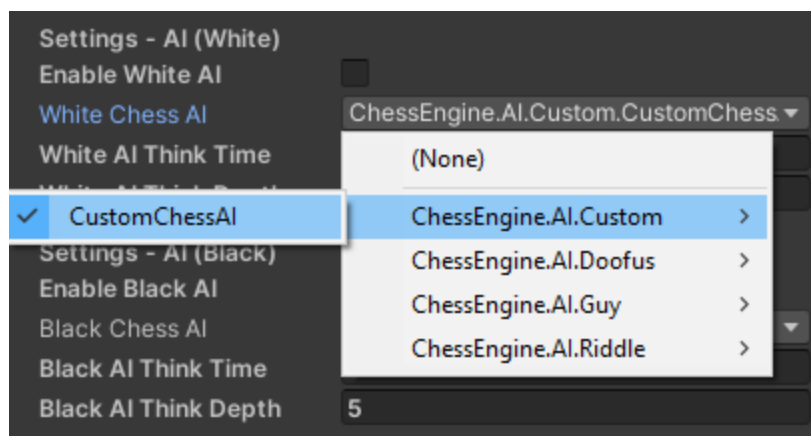
- For consistency you may want to review how existing AI opponents are organized into the '[ChessEngineAndAI/Modules/](#)' folder and copy this structure.

	ChessEngine.AI.Custom.dll	9/12/2023 11:00 AM	Application exten...	13 KB
	ChessEngine.AI.Custom.pdb	9/12/2023 11:00 AM	Program Debug D...	12 KB
	ChessEngine.AI.Custom.xml	9/12/2023 11:00 AM	XML File	15 KB

- In the above screenshot:

- [ChessEngine.AI.Custom.dll](#) is your new managed AI dll. (**Required**)
- [ChessEngine.AI.Custom.pdb](#) is your debugging file for your new AI. (**Optional**)
- [ChessEngine.AI.Custom.xml](#) contains intellisense comments for your new AI. (**Optional**)

5. Open the '[FOR_UNITY_PROJECT](#)' folder and copy the relevant script(s) into your Unity project. The directory you put these in does not matter except for that editor scripts need to be nested in an 'Editor' folder as Unity requires.
6. If you have successfully added your new AI opponent you will now see it appear in the dropdown for AI selection in the Inspector pane for the [ChessAIGameManager](#) component.

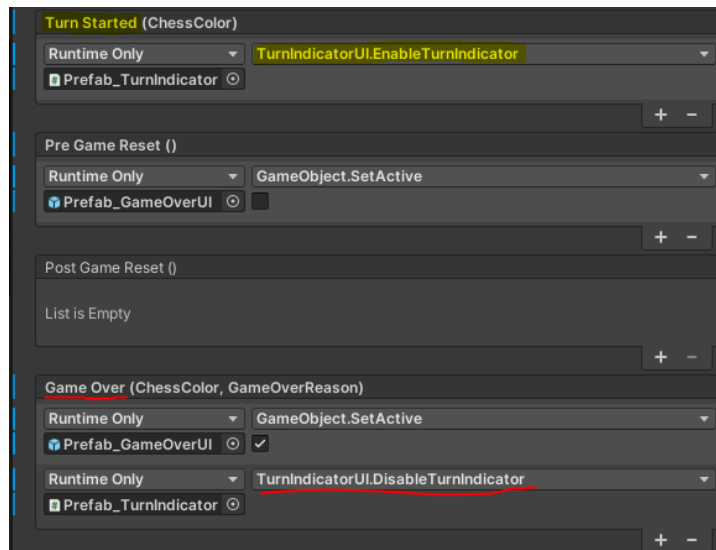


A screenshot showing the Inspector pane for the ChessAIGameManager component showing the newly added AI opponent.

UI Overview

7.a. Turn Indicator UI

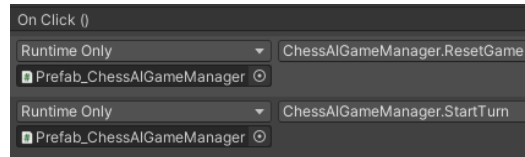
- The turn indicator UI displays whose turn it is whenever a turn starts.
- You will find the [GameObject](#) that controls the UI for turn indications in any demo scene under '[Canvas/Prefab_TurnIndicator](#)' in the scene 'Hierarchy'. On this [GameObject](#) you will see the script that controls the turn indicator UI, '[TurnIndicatorUI.cs](#)', which contains the following settings:
 - **Turn Text** - A reference to a UI [Text](#) object.
 - If you are using TextMesh pro you can modify the '[TurnIndicatorUI.cs](#)' script to use the [TMP_Text](#) object instead of the legacy [Text](#) object.
 - **Turn Indicator Object** - A reference to the [GameObject](#) in the scene that is the actual turn indicator UI object. This object is enabled/disabled based on when the turn indicator should be active.
 - **Turn Indicator Timeout** - The number of seconds the turn indicator remains visible for.
- The 'Turn Indicator UI' is enabled through editor-events configured in the [ChessGameManager](#) (or similar like [ChessAIGameManager](#)) component as shown in the screenshot below:



- The event highlighted in yellow is where the turn indicator UI is activated when a turn is started.
- The event underlined in red is used to instantly deactivate the turn indicator UI when the game ends.
- The turn indicator UI will deactivate itself after 'turn indicator timeout' seconds.

7.b. Game Over UI

- The game over UI displays when the game is over and the reason for it ending.
- You will find the [GameObject](#) that controls the UI for the game over notice in any demo scene under '[Canvas/Prefab_GameOverUI](#)'.
- The 'Game Over UI' includes a 'Restart' game button that restarts the game using the 'OnClick()' event from a standard Unity UI button as shown in the screenshot below:



7.c. Player Indicator UI

- The player indicator UI simply displays whether a human or AI is controlling a team.

7.d. Undo & Redo Button UI

- The undo & redo button UI elements pass commands to the underlying chess engine to undo changes in the game state or redo undone changes.

Extensions

8.a. Extension: Main Menu System

- An extension that adds a fully-functional main menu (without multiplayer since that is its own optional package) to the provided chess game.
- Import the '[Extension_MainMainSystem.unitypackage](#)' package to use this extension.
- Check out the [documentation for the 'Main Menu System' extension](#).
- It is recommended to put the scene '[MenuScene_Main](#)' as the first scene in your game's build order so it is loaded first.

8.b. Extension: Multiplayer Netcode Demo

- An extension that adds multiplayer support to the chess game using Unity's Netcode.
- To import Netcode multiplayer support simply import the extension package `'Extension_Netcode_Multiplayer.unitypackage'` into your project.

FAQ

(Frequently Asked Questions)

Q: Does this asset work for mobile games like on Android and iOS?

A: Yes! The chess engine and AI both work on any Unity supported platform including Android and iOS.

Q: Why are the 'PreChessPieceMoved' and 'ChessPieceMoved' events not invoked on a Rook when involved in a 'castle' move?

A: This is because for the purposes of these events (and most chess rules) castling is a king's move and therefore only the king is considered to have performed a 'move'. You may use the 'Castled' events found in the [Rook](#), [King](#), [ChessTable](#), and [Instance](#) classes in the Chess Engine to detect when a rook is involved in castling.

Q: Why should I generally use serialization over FEN strings for saving/loading game states?

A: FEN strings are great for saving and loading game states but come with many drawbacks that serialization does not, here are some examples:

- When loading the game state with FEN strings many assumptions and inferences to load the proper game state.
- FEN strings are missing some key game information such as non-standard chess game information, move count of each piece, and more.
- Relatively poor performance due to computation tasks in resolving whether or not pieces like pawns for example have moved, which rook is the 'king side' rook, and many more.
- Poor modding/custom gamemode support with FEN strings.

Q: I am unable or unreliably able to select chess pieces after adding custom models and/or colliders.

A: The [TouchClicker.cs](#) system is actually looking for you to click on the chess table tile and not the piece itself. It is likely that in this case your chess piece's collider is blocking tile selection. In this case it is recommended that you (in your prefabs) set your chess pieces to be in their own 'Layer', then set this layer to be ignored in the 'Ignore Pointer Raycast Layers' setting for the [TouchClicker](#) component.

