

# Chess Engine

1.3.6

Generated by Doxygen 1.9.5



<b>1 Namespace Index</b>	<b>1</b>
1.1 Package List	1
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 Namespace Documentation</b>	<b>7</b>
4.1 ChessEngine Namespace Reference	7
4.1.1 Enumeration Type Documentation	8
4.1.1.1 ChessColor	8
4.2 ChessEngine.Delegates Namespace Reference	8
4.2.1 Function Documentation	8
4.2.1.1 ActionRef< T >()	8
4.2.1.2 ValueActionRef< VAL_T, REF_T >()	9
4.2.1.3 ValueActionRef< VAL_T, VAL_T2, REF_T >()	9
4.2.1.4 ValueActionRef< VAL_T, VAL_T2, VAL_T3, REF_T >()	10
4.2.1.5 ValueActionRef< VAL_T, VAL_T2, VAL_T3, VAL_T4, REF_T >()	10
4.3 ChessEngine.Serialization Namespace Reference	11
4.4 ChessEngine.TimeSystem Namespace Reference	11
4.5 ChessEngine.Undo Namespace Reference	12
4.6 ChessEngine.Utility Namespace Reference	12
<b>5 Class Documentation</b>	<b>13</b>
5.1 ChessEngine.AttackInfo Class Reference	13
5.1.1 Detailed Description	13
5.2 ChessEngine.Bishop Class Reference	13
5.2.1 Detailed Description	14
5.2.2 Member Function Documentation	14
5.2.2.1 GenerateAttacksList()	14
5.2.2.2 GenerateMovesList()	14
5.2.2.3 GetChessPieceType()	15
5.2.2.4 GetFENIdentifier()	15
5.3 ChessEngine.ChessPiece Class Reference	15
5.3.1 Detailed Description	17
5.3.2 Constructor & Destructor Documentation	17
5.3.2.1 ChessPiece() [1/2]	18
5.3.2.2 ChessPiece() [2/2]	19
5.3.3 Member Function Documentation	19
5.3.3.1 Capture()	19
5.3.3.2 GenerateAttacksList()	19
5.3.3.3 GenerateMovesList()	20

5.3.3.4 GetAttacks()	20
5.3.3.5 GetChessPieceType()	21
5.3.3.6 GetFENIdentifier()	21
5.3.3.7 GetMoves()	21
5.3.3.8 GetValidAttacks()	22
5.3.3.9 GetValidMoves()	22
5.3.3.10 Move()	22
5.3.3.11 OnPostMoved()	22
5.3.3.12 OnPreMoved()	23
5.3.3.13 OnTileIndexChanged()	23
5.4 ChessEngine.ChessTable Class Reference	23
5.4.1 Detailed Description	26
5.4.2 Constructor & Destructor Documentation	26
5.4.2.1 ChessTable() [1/3]	26
5.4.2.2 ChessTable() [2/3]	26
5.4.2.3 ChessTable() [3/3]	26
5.4.3 Member Function Documentation	27
5.4.3.1 CreateBishop()	27
5.4.3.2 CreateKing()	27
5.4.3.3 CreateKnight()	28
5.4.3.4 CreatePawn()	28
5.4.3.5 CreatePieceByType()	28
5.4.3.6 CreateQueen()	29
5.4.3.7 CreateRook()	29
5.4.3.8 CreateSerializedPiece()	30
5.4.3.9 DestroyPiece()	30
5.4.3.10 DestroyPieceByIndex()	30
5.4.3.11 DetermineRookSides()	30
5.4.3.12 GenerateEPDString()	31
5.4.3.13 GetKing()	31
5.4.3.14 GetPieceByIndex()	32
5.4.3.15 GetRookReferences()	32
5.4.3.16 GetTile() [1/2]	32
5.4.3.17 GetTile() [2/2]	33
5.4.3.18 GetTileByID()	33
5.4.3.19 IsInCheck()	33
5.4.3.20 SetState()	34
5.4.3.21 SetStateToEPD()	34
5.4.4 Event Documentation	35
5.4.4.1 Castled	35
5.4.4.2 ChessPieceMoved	35
5.4.4.3 CreatedSerializedChessPiece	35

5.4.4.4 LoadedSerializedTable	35
5.4.4.5 PreChessPieceMoved	36
5.5 ChessEngine.ChessTableTile Class Reference	36
5.5.1 Detailed Description	37
5.5.2 Constructor & Destructor Documentation	37
5.5.2.1 ChessTableTile() [1/2]	37
5.5.2.2 ChessTableTile() [2/2]	37
5.5.3 Member Function Documentation	37
5.5.3.1 GetPiece()	38
5.5.3.2 IsTileAttackable()	38
5.5.3.3 IsTileThreatened()	38
5.5.3.4 MovePieceToTile()	39
5.6 ChessEngine.Undo.HistoryEntry Class Reference	39
5.6.1 Detailed Description	40
5.7 ChessEngine.Instance Class Reference	40
5.7.1 Detailed Description	42
5.7.2 Constructor & Destructor Documentation	42
5.7.2.1 Instance() [1/3]	42
5.7.2.2 Instance() [2/3]	43
5.7.2.3 Instance() [3/3]	43
5.7.3 Member Function Documentation	43
5.7.3.1 EndTurn()	43
5.7.3.2 GenerateFENCastleString()	44
5.7.3.3 GenerateFENCastleStringForRook()	44
5.7.3.4 GenerateFENEnPassantString()	44
5.7.3.5 GenerateFENString()	45
5.7.3.6 IsGameOver()	45
5.7.3.7 OnGameOver()	45
5.7.3.8 OnTurnEnded()	45
5.7.3.9 OnTurnStarted()	46
5.7.3.10 QueenPawn()	46
5.7.3.11 SetState()	46
5.7.3.12 SetStateToFEN()	47
5.7.4 Event Documentation	47
5.7.4.1 Castled	47
5.7.4.2 ChessPieceMoved	47
5.7.4.3 IsGameOverCallback	48
5.7.4.4 LoadedSerializedInstance	48
5.7.4.5 PreChessPieceMoved	48
5.7.4.6 QueeningPawnCallback	48
5.8 ChessEngine.Undo.InstanceHistory Class Reference	49
5.8.1 Detailed Description	50

5.8.2 Constructor & Destructor Documentation	51
5.8.2.1 InstanceHistory()	51
5.8.3 Member Function Documentation	51
5.8.3.1 OnGameOver()	51
5.8.3.2 OnInstanceSet()	51
5.8.3.3 OnInstanceUnset()	52
5.8.3.4 OnPreChessPieceMoved()	52
5.8.3.5 OnTurnEnded()	52
5.8.3.6 PeekMove()	52
5.8.3.7 PeekUndoneMove()	53
5.8.3.8 PushMoveHistory()	53
5.8.3.9 RedoMove()	53
5.8.3.10 UndoMove()	53
5.9 ChessEngine.King Class Reference	54
5.9.1 Detailed Description	54
5.9.2 Member Function Documentation	55
5.9.2.1 GenerateAttacksList()	55
5.9.2.2 GenerateMovesList()	55
5.9.2.3 GetChessPieceType()	55
5.9.2.4 GetFENIdentifier()	56
5.9.2.5 OnCastled()	56
5.9.2.6 OnTileIndexChanged()	56
5.9.3 Event Documentation	57
5.9.3.1 Castled	57
5.10 ChessEngine.Knight Class Reference	57
5.10.1 Detailed Description	57
5.10.2 Member Function Documentation	58
5.10.2.1 GenerateAttacksList()	58
5.10.2.2 GenerateMovesList()	58
5.10.2.3 GetChessPieceType()	58
5.10.2.4 GetFENIdentifier()	59
5.11 ChessEngine.MoveData Class Reference	59
5.11.1 Detailed Description	59
5.11.2 Constructor & Destructor Documentation	59
5.11.2.1 MoveData()	59
5.12 ChessEngine.MoveInfo Class Reference	60
5.12.1 Detailed Description	60
5.13 ChessEngine.Pawn Class Reference	60
5.13.1 Detailed Description	61
5.13.2 Member Function Documentation	61
5.13.2.1 GenerateAttacksList()	62
5.13.2.2 GenerateMovesList()	62

5.13.2.3 GetChessPieceType()	62
5.13.2.4 GetFENIdentifier()	63
5.13.2.5 IsOnStartingRow()	63
5.13.2.6 OnPostMoved()	63
5.13.2.7 OnPreMoved()	63
5.13.2.8 OnTileIndexChanged()	64
5.14 ChessEngine.Queen Class Reference	64
5.14.1 Detailed Description	65
5.14.2 Member Function Documentation	65
5.14.2.1 GenerateAttacksList()	65
5.14.2.2 GenerateMovesList()	65
5.14.2.3 GetChessPieceType()	65
5.14.2.4 GetFENIdentifier()	66
5.15 ChessEngine.Rook Class Reference	66
5.15.1 Detailed Description	67
5.15.2 Member Function Documentation	67
5.15.2.1 GenerateAttacksList()	67
5.15.2.2 GenerateMovesList()	68
5.15.2.3 GetChessPieceType()	68
5.15.2.4 GetFENIdentifier()	68
5.15.3 Event Documentation	68
5.15.3.1 Castled	69
5.16 ChessEngine.RookReferences Class Reference	69
5.16.1 Detailed Description	69
5.17 ChessEngine.Serialization.SerializedChessInstance Class Reference	69
5.17.1 Detailed Description	70
5.17.2 Constructor & Destructor Documentation	70
5.17.2.1 SerializedChessInstance()	70
5.18 ChessEngine.Serialization.SerializedChessPiece Class Reference	71
5.18.1 Constructor & Destructor Documentation	71
5.18.1.1 SerializedChessPiece()	71
5.19 ChessEngine.Serialization.SerializedChessTable Class Reference	72
5.19.1 Detailed Description	72
5.19.2 Constructor & Destructor Documentation	72
5.19.2.1 SerializedChessTable()	72
5.20 ChessEngine.Serialization.SerializedInstanceHistory Class Reference	73
5.20.1 Detailed Description	73
5.20.2 Constructor & Destructor Documentation	74
5.20.2.1 SerializedInstanceHistory()	74
5.21 ChessEngine.Serialization.SerializedMoveInfo Class Reference	74
5.21.1 Detailed Description	75
5.21.2 Constructor & Destructor Documentation	75

5.21.2.1 SerializedMoveInfo() . . . . .	75
5.21.3 Member Function Documentation . . . . .	75
5.21.3.1 ToMoveInfo() . . . . .	75
5.22 ChessEngine.Serialization.SerializedPawn Class Reference . . . . .	76
5.22.1 Detailed Description . . . . .	76
5.22.2 Constructor & Destructor Documentation . . . . .	76
5.22.2.1 SerializedPawn() . . . . .	76
5.23 ChessEngine.Serialization.SerializedRook Class Reference . . . . .	77
5.23.1 Detailed Description . . . . .	77
5.23.2 Constructor & Destructor Documentation . . . . .	77
5.23.2.1 SerializedRook() . . . . .	77
5.24 ChessEngine.TileIndex Struct Reference . . . . .	78
5.24.1 Detailed Description . . . . .	79
5.24.2 Member Function Documentation . . . . .	79
5.24.2.1 Equals() . . . . .	79
5.24.2.2 GetTileID() . . . . .	79
5.25 ChessEngine.TimeSystem.TimeManager Class Reference . . . . .	79
5.25.1 Detailed Description . . . . .	80
5.25.2 Member Function Documentation . . . . .	80
5.25.2.1 SetElapsedTime() . . . . .	80
<b>Index</b>	<b>81</b>



# Chapter 1

## Namespace Index

### 1.1 Package List

Here are the packages with brief descriptions (if available):

<a href="#">ChessEngine</a>	7
<a href="#">ChessEngine.Delegates</a>	8
<a href="#">ChessEngine.Serialization</a>	11
<a href="#">ChessEngine.TimeSystem</a>	11
<a href="#">ChessEngine.Undo</a>	12
<a href="#">ChessEngine.Utility</a>	12



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ChessEngine.AttackInfo . . . . .	13
ChessEngine.ChessPiece . . . . .	15
ChessEngine.Bishop . . . . .	13
ChessEngine.King . . . . .	54
ChessEngine.Knight . . . . .	57
ChessEngine.Pawn . . . . .	60
ChessEngine.Queen . . . . .	64
ChessEngine.Rook . . . . .	66
ChessEngine.ChessTable . . . . .	23
ChessEngine.ChessTableTile . . . . .	36
IEquatable	
ChessEngine.TileIndex . . . . .	78
ChessEngine.Instance . . . . .	40
ChessEngine.Undo.InstanceHistory . . . . .	49
ISerializable	
ChessEngine.Serialization.SerializedChessInstance . . . . .	69
ChessEngine.Serialization.SerializedChessPiece . . . . .	71
ChessEngine.Serialization.SerializedPawn . . . . .	76
ChessEngine.Serialization.SerializedRook . . . . .	77
ChessEngine.Serialization.SerializedChessTable . . . . .	72
ChessEngine.Serialization.SerializedInstanceHistory . . . . .	73
ChessEngine.Serialization.SerializedMoveInfo . . . . .	74
ChessEngine.Serialization.SerializedPawn . . . . .	76
ChessEngine.Serialization.SerializedRook . . . . .	77
ChessEngine.TileIndex . . . . .	78
ChessEngine.Undo.HistoryEntry . . . . .	39
ChessEngine.MoveData . . . . .	59
ChessEngine.MoveInfo . . . . .	60
ChessEngine.RookReferences . . . . .	69
ChessEngine.TimeSystem.TimeManager . . . . .	79



## Chapter 3

# Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">ChessEngine.AttackInfo</a>	
Holds information about an attack. . . . .	13
<a href="#">ChessEngine.Bishop</a>	
Implementation of a Chess <a href="#">Bishop</a> . . . . .	13
<a href="#">ChessEngine.ChessPiece</a>	
An abstract component intended to be inherited from to define the behaviours of individual chess pieces. . . . .	15
<a href="#">ChessEngine.ChessTable</a>	
The <a href="#">ChessTable</a> component is to be attached to a gameObject on which 'ChessTableTiles' will be created . . . . .	23
<a href="#">ChessEngine.ChessTableTile</a>	
A <a href="#">ChessTableTile</a> component is to be attached to each individual tile that makes up a chess table, each tile holds information about itself like offset. . . . .	36
<a href="#">ChessEngine.Undo.HistoryEntry</a>	
A class that contains data about a move so it can be undone or redone. . . . .	39
<a href="#">ChessEngine.Instance</a>	
An instance of a chess engine. The core class that operates the chess engine. . . . .	40
<a href="#">ChessEngine.Undo.InstanceHistory</a>	
A class that tracks moves in a chess engine <a href="#">Instance</a> and allows for undoing and redoing of moves. . . . .	49
<a href="#">ChessEngine.King</a>	
Implementation of a Chess <a href="#">King</a> . . . . .	54
<a href="#">ChessEngine.Knight</a>	
Implementation of a Chess <a href="#">Knight</a> . . . . .	57
<a href="#">ChessEngine.MoveData</a>	
Move data contains simply the 'from' tile index, the 'to' tile index, and a readonly boolean 'isAttack' that tracks whether the move is an attack or just a move. . . . .	59
<a href="#">ChessEngine.MoveInfo</a>	
A class containing information about a move. . . . .	60
<a href="#">ChessEngine.Pawn</a>	
Implementation of a Chess <a href="#">Pawn</a> . . . . .	60
<a href="#">ChessEngine.Queen</a>	
Implementation of a Chess <a href="#">Queen</a> . . . . .	64
<a href="#">ChessEngine.Rook</a>	
Implementation of a Chess <a href="#">Rook</a> . . . . .	66

<a href="#">ChessEngine.RookReferences</a>	
A class containing references to <a href="#">Rook</a> pieces based on what color and side they are on. . . . .	69
<a href="#">ChessEngine.Serialization.SerializedChessInstance</a>	
A serializable class that provides a complete representation of a chess <a href="#">Instance</a> . . . . .	69
<a href="#">ChessEngine.Serialization.SerializedChessPiece</a> . . . . .	71
<a href="#">ChessEngine.Serialization.SerializedChessTable</a>	
A serializable class that provides a complete representation of a Chess board. . . . .	72
<a href="#">ChessEngine.Serialization.SerializedInstanceHistory</a>	
A serializable class that provides a complete representation of the moves and undone moves stacks for an InstanceHistory object. . . . .	73
<a href="#">ChessEngine.Serialization.SerializedMoveInfo</a>	
A serializable class that provides a complete representation of a <a href="#">MoveInfo</a> instance. . . . .	74
<a href="#">ChessEngine.Serialization.SerializedPawn</a>	
Derived from <a href="#">SerializedPawn</a> this class represents a serialized <a href="#">Pawn</a> piece. . . . .	76
<a href="#">ChessEngine.Serialization.SerializedRook</a>	
Derived from <a href="#">SerializedRook</a> this class represents a serialized <a href="#">Rook</a> piece. . . . .	77
<a href="#">ChessEngine.TileIndex</a>	
A <a href="#">TileIndex</a> represents the index of a <a href="#">ChessTableTile</a> along an 8x8 grid. Note that x == 0, y == 0, represents the lower-left origin of the chess table. . . . .	78
<a href="#">ChessEngine.TimeSystem.TimeManager</a>	
A class that provides easy-to-use time related methods. . . . .	79

## Chapter 4

# Namespace Documentation

### 4.1 ChessEngine Namespace Reference

#### Classes

- class [AttackInfo](#)  
*Holds information about an attack.*
- class [Bishop](#)  
*Implementation of a Chess [Bishop](#).*
- class [ChessPiece](#)  
*An abstract component intended to be inherited from to define the behaviours of individual chess pieces.*
- class [ChessTable](#)  
*The [ChessTable](#) component is to be attached to a gameObject on which 'ChessTableTiles' will be created.*
- class [ChessTableTile](#)  
*A [ChessTableTile](#) component is to be attached to each individual tile that makes up a chess table, each tile holds information about itself like offset.*
- class [Instance](#)  
*An instance of a chess engine. The core class that operates the chess engine.*
- class [King](#)  
*Implementation of a Chess [King](#).*
- class [Knight](#)  
*Implementation of a Chess [Knight](#).*
- class [MoveData](#)  
*Move data contains simply the 'from' tile index, the 'to' tile index, and a readonly boolean 'isAttack' that tracks whether the move is an attack or just a move.*
- class [MoveInfo](#)  
*A class containing information about a move.*
- class [Pawn](#)  
*Implementation of a Chess [Pawn](#).*
- class [Queen](#)  
*Implementation of a Chess [Queen](#).*
- class [Rook](#)  
*Implementation of a Chess [Rook](#).*
- class [RookReferences](#)  
*A class containing references to [Rook](#) pieces based on what color and side they are on.*
- struct [TileIndex](#)  
*A [TileIndex](#) represents the index of a [ChessTableTile](#) along an 8x8 grid. Note that x == 0, y == 0, represents the lower-left origin of the chess table.*

## Enumerations

- enum [ChessColor](#)  
*Represents the colors of the tiles and pieces on the chess table.*
- enum [ChessPieceType](#)  
*An enumerate that contains all valid chess piece types.*
- enum [GameOverReason](#)  
*The reason the game ended.*

### 4.1.1 Enumeration Type Documentation

#### 4.1.1.1 ChessColor

enum [ChessEngine.ChessColor](#)

Represents the colors of the tiles and pieces on the chess table.

Author: Mathew Aloisio

## 4.2 ChessEngine.Delegates Namespace Reference

### Functions

- delegate void [ActionRef< T >](#) (ref T pItem)  
*A simple delegate for events where an argument is passed by reference.*
- delegate void [ValueActionRef< VAL\\_T, REF\\_T >](#) (VAL\_T pValue, ref REF\_T pItem)  
*A simple delegate for events where the first argument is a reference to some type and the second argument is passed by reference.*
- delegate void [ValueActionRef< VAL\\_T, VAL\\_T2, REF\\_T >](#) (VAL\_T pValue, VAL\_T2 pValue2, ref REF\_T pItem)  
*A simple delegate for events where the first 2 arguments are a reference to some type and the last argument is passed by reference.*
- delegate void [ValueActionRef< VAL\\_T, VAL\\_T2, VAL\\_T3, REF\\_T >](#) (VAL\_T pValue, VAL\_T2 pValue2, VAL\_T3 pValue3, ref REF\_T pItem)  
*A simple delegate for events where the first 3 arguments are a reference to some type and the last argument is passed by reference.*
- delegate void [ValueActionRef< VAL\\_T, VAL\\_T2, VAL\\_T3, VAL\\_T4, REF\\_T >](#) (VAL\_T pValue, VAL\_T2 pValue2, VAL\_T3 pValue3, VAL\_T4 pValue4, ref REF\_T pItem)  
*A simple delegate for events where the first 4 arguments are a reference to some type and the last argument is passed by reference.*

### 4.2.1 Function Documentation

#### 4.2.1.1 ActionRef< T >()

```
delegate void ChessEngine.Delegates.ActionRef< T > (
    ref T pItem )
```

A simple delegate for events where an argument is passed by reference.



## Template Parameters

<i>T</i>	
----------	--

## Parameters

<i>pItem</i>	
--------------	--

## 4.2.1.2 ValueActionRef&lt; VAL\_T, REF\_T &gt;()

```
delegate void ChessEngine.Delegates.ValueActionRef< VAL_T, REF_T > (
    VAL_T pValue,
    ref REF_T pItem )
```

A simple delegate for events where the first argument is a reference to some type and the second argument is passed by reference.

## Template Parameters

<i>VAL</i> <sub>↔</sub> <i>_T</i>	The type of the non-reference parameter.
<i>REF</i> <sub>↔</sub> <i>_T</i>	The type of the reference parameter.

## Parameters

<i>pValue</i>	The non-reference value argument.
<i>pItem</i>	The reference.

## 4.2.1.3 ValueActionRef&lt; VAL\_T, VAL\_T2, REF\_T &gt;()

```
delegate void ChessEngine.Delegates.ValueActionRef< VAL_T, VAL_T2, REF_T > (
    VAL_T pValue,
    VAL_T2 pValue2,
    ref REF_T pItem )
```

A simple delegate for events where the first 2 arguments are a reference to some type and the last argument is passed by reference.

## Template Parameters

<i>VAL_T</i>	The type of the first non-reference parameter.
<i>VAL_T2</i>	The type of the second non-reference parameter.
<i>REF</i> <sub>↔</sub> <i>_T</i>	The type of the reference parameter.

## Parameters

<i>pValue</i>	The first non-reference value argument.
<i>pValue2</i>	The second non-reference value argument.
<i>pItem</i>	The reference.

**4.2.1.4 ValueActionRef< VAL\_T, VAL\_T2, VAL\_T3, REF\_T >()**

```
delegate void ChessEngine.Delegates.ValueActionRef< VAL_T, VAL_T2, VAL_T3, REF_T > (
    VAL_T pValue,
    VAL_T2 pValue2,
    VAL_T3 pValue3,
    ref REF_T pItem )
```

A simple delegate for events where the first 3 arguments are a reference to some type and the last argument is passed by reference.

## Template Parameters

<i>VAL_T</i>	The type of the first non-reference parameter.
<i>VAL_T2</i>	The type of the second non-reference parameter.
<i>VAL_T3</i>	The type of the third non-reference parameter.
<i>REF_T</i>	The type of the reference parameter.

## Parameters

<i>pValue</i>	The first non-reference value argument.
<i>pValue2</i>	The second non-reference value argument.
<i>pValue3</i>	The third non-reference value argument.
<i>pItem</i>	The reference.

**4.2.1.5 ValueActionRef< VAL\_T, VAL\_T2, VAL\_T3, VAL\_T4, REF\_T >()**

```
delegate void ChessEngine.Delegates.ValueActionRef< VAL_T, VAL_T2, VAL_T3, VAL_T4, REF_T > (
    VAL_T pValue,
    VAL_T2 pValue2,
    VAL_T3 pValue3,
    VAL_T4 pValue4,
    ref REF_T pItem )
```

A simple delegate for events where the first 4 arguments are a reference to some type and the last argument is passed by reference.

## Template Parameters

<i>VAL_T</i>	The type of the first non-reference parameter.
<i>VAL_T2</i>	The type of the second non-reference parameter.
<i>VAL_T3</i>	The type of the third non-reference parameter.
<i>VAL_T4</i>	The type of the fourth non-reference parameter.
<i>REF<sub>↔</sub>_T</i>	The type of the reference parameter.

## Parameters

<i>pValue</i>	The first non-reference value argument.
<i>pValue2</i>	The second non-reference value argument.
<i>pValue3</i>	The third non-reference value argument.
<i>pValue4</i>	The fourth non-reference value argument.
<i>pltem</i>	The reference.

## 4.3 ChessEngine.Serialization Namespace Reference

### Classes

- class [SerializedChessInstance](#)  
A serializable class that provides a complete representation of a chess [Instance](#).
- class [SerializedChessPiece](#)
- class [SerializedChessTable](#)  
A serializable class that provides a complete representation of a Chess board.
- class [SerializedInstanceHistory](#)  
A serializable class that provides a complete representation of the moves and undone moves stacks for an [Instance<sub>↔</sub>](#) History object.
- class [SerializedMoveInfo](#)  
A serializable class that provides a complete representation of a [MoveInfo](#) instance.
- class [SerializedPawn](#)  
Derived from [SerializedPawn](#) this class represents a serialized [Pawn](#) piece.
- class [SerializedRook](#)  
Derived from [SerializedRook](#) this class represents a serialized [Rook](#) piece.

## 4.4 ChessEngine.TimeSystem Namespace Reference

### Classes

- class [TimeManager](#)  
A class that provides easy-to-use time related methods.

## 4.5 ChessEngine.Undo Namespace Reference

### Classes

- class [HistoryEntry](#)  
*A class that contains data about a move so it can be undone or redone.*
- class [InstanceHistory](#)  
*A class that tracks moves in a chess engine [Instance](#) and allows for undoing and redoing of moves.*

## 4.6 ChessEngine.Utility Namespace Reference

### Classes

- class **FENUtility**  
*A public static class with utility methods that help parse FEN strings.*
- class **InputValidation**  
*A public static class that validates Chess notation related inputs.*

## Chapter 5

# Class Documentation

### 5.1 ChessEngine.AttackInfo Class Reference

Holds information about an attack.

#### Public Attributes

- [ChessTableTile](#) **moveToTile**  
*The [ChessTableTile](#) the attacker is moving to.*
- [ChessTableTile](#) **attackTile**  
*The [ChessTableTile](#) being attacked.*

#### 5.1.1 Detailed Description

Holds information about an attack.

Author: Mathew Aloisio

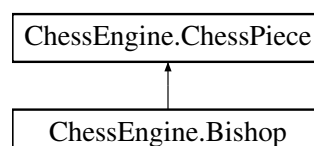
The documentation for this class was generated from the following file:

- AttackInfo.cs

### 5.2 ChessEngine.Bishop Class Reference

Implementation of a Chess [Bishop](#).

Inheritance diagram for ChessEngine.Bishop:



## Public Member Functions

- **Bishop** ([ChessTable](#) pTable, [ChessColor](#) pColor, [TileIndex](#) pTileIndex)
- **Bishop** ([ChessTable](#) pTable, [ChessPiece](#) pOther)
- override List< [ChessTableTile](#) > [GenerateMovesList](#) ()  
*Returns a list with the ChessTableTiles of all possible moves for this piece.*
- override List< [AttackInfo](#) > [GenerateAttacksList](#) ([ChessTableTile](#) pSpoofOccupiedTile, [ChessTableTile](#) p←  
 SpoofUnoccupiedTile)  
*Returns a list with the AttackInfos of all possible attacks for this piece.*
- override string [GetFENIdentifier](#) ()  
*Returns the FEN identifier for the chess piece.*
- override [ChessPieceType](#) [GetChessPieceType](#) ()  
*Returns the ChessPieceType for this [ChessPiece](#).*

## Additional Inherited Members

### 5.2.1 Detailed Description

Implementation of a Chess [Bishop](#).

Author: Mathew Aloisio

### 5.2.2 Member Function Documentation

#### 5.2.2.1 GenerateAttacksList()

```
override List< AttackInfo > ChessEngine.Bishop.GenerateAttacksList (
    ChessTableTile pSpoofOccupiedTile,
    ChessTableTile pSpoofUnoccupiedTile ) [virtual]
```

Returns a list with the AttackInfos of all possible attacks for this piece.

#### Returns

A list of AttackInfos for all possible attacks of this piece.

Implements [ChessEngine.ChessPiece](#).

#### 5.2.2.2 GenerateMovesList()

```
override List< ChessTableTile > ChessEngine.Bishop.GenerateMovesList ( ) [virtual]
```

Returns a list with the ChessTableTiles of all possible moves for this piece.

#### Returns

A list of ChessTableTiles for all possible moves of this piece.

Implements [ChessEngine.ChessPiece](#).

### 5.2.2.3 GetChessPieceType()

```
override ChessPieceType ChessEngine.Bishop.GetChessPieceType ( ) [virtual]
```

Returns the ChessPieceType for this [ChessPiece](#).

#### Returns

the ChessPieceType for this [ChessPiece](#).

Implements [ChessEngine.ChessPiece](#).

### 5.2.2.4 GetFENIdentifier()

```
override string ChessEngine.Bishop.GetFENIdentifier ( ) [virtual]
```

Returns the FEN identifier for the chess piece.

White Piece | Black Piece | Chess Piece P | p | [Pawn](#) N | n | [Knight](#) B | b | [Bishop](#) R | r | [Rook](#) Q | q | [Queen](#) K | k | [King](#)

#### Returns

a string representing the FEN identifier for the chess piece.

Implements [ChessEngine.ChessPiece](#).

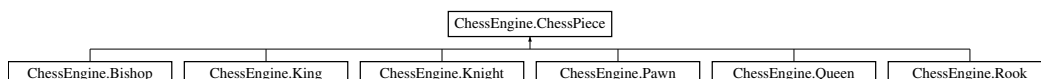
The documentation for this class was generated from the following file:

- Bishop.cs

## 5.3 ChessEngine.ChessPiece Class Reference

An abstract component intended to be inherited from to define the behaviours of individual chess pieces.

Inheritance diagram for ChessEngine.ChessPiece:



## Public Member Functions

- [ChessPiece](#) ([ChessTable](#) pTable, [ChessColor](#) pColor, [TileIndex](#) pTileIndex)  
*Invoked whenever a [ChessPiece](#) instance is constructed.*
- [ChessPiece](#) ([ChessTable](#) pTable, [ChessPiece](#) pOther)  
*Constructs a [ChessPiece](#) instance that is a copy of pOther except belonging to the pTable chess table.*
- [MoveInfo](#) Move ([TileIndex](#) pToTileIndex, [ChessPiece](#) pAttackingPiece)  
*Moves the chess piece to the given [TileIndex](#), invokes the 'Moved' event.*
- void [Capture](#) ([MoveInfo](#) pMoveInfo)  
*Marks the [ChessPiece](#) as captured.*
- List< [ChessTableTile](#) > [GetMoves](#) ()  
*Returns a list with the [ChessTableTile](#)s of all possible moves for this piece.*
- List< [AttackInfo](#) > [GetAttacks](#) ([ChessTableTile](#) pSpoofofOccupiedTile=null, [ChessTableTile](#) pSpoofofUnoccupiedTile=null)  
*Returns a list with the [AttackInfo](#)s of all possible attacks for this piece.*
- List< [ChessTableTile](#) > [GetValidMoves](#) ()  
*Uses overrideable [GetMove\(\)](#) to get a list of moves and then filters them based on current check status, and whether or not the move will put you in check.*
- List< [AttackInfo](#) > [GetValidAttacks](#) ()  
*Uses overrideable [GetAttacks\(\)](#) to get a list of attacks and then filters them based on current check status, and whether or not the attack will put you in check.*
- abstract List< [ChessTableTile](#) > [GenerateMovesList](#) ()  
*Returns a list with the [ChessTableTile](#)s of all possible moves for this piece. (These moves are not necessarily valid and may be prevented by check, or other scenarios.)*
- abstract List< [AttackInfo](#) > [GenerateAttacksList](#) ([ChessTableTile](#) pSpoofofOccupiedTile=null, [ChessTableTile](#) pSpoofofUnoccupiedTile=null)  
*Returns a list with the [AttackInfo](#)s of all possible attacks for this piece. (These attacks are not necessarily valid and may be prevented by check, or other scenarios.)*
- abstract string [GetFENIdentifier](#) ()  
*Returns the FEN identifier for the chess piece.*
- abstract [ChessPieceType](#) [GetChessPieceType](#) ()  
*Returns the [ChessPieceType](#) for the [ChessPiece](#).*

## Static Public Attributes

- const int **RightX** = 1  
*Returns the 'right' x direction for this chess piece, i.e. 1 on both teams.*

## Protected Member Functions

- virtual void [OnPreMoved](#) (ref [MoveInfo](#) pMoveInfo)  
*Invoked just before the piece moves.*
- virtual void [OnPostMoved](#) ([MoveInfo](#) pMoveInfo)  
*Invoked just after the piece moves.*
- virtual void [OnTileIndexChanged](#) (bool pWasEmpty, [TileIndex](#) pOldIndex)  
*Executed automatically when the [TileIndex](#) of the [ChessPiece](#) is changed.*



## Properties

- **int ForwardY** [get]  
Returns the 'forward' y direction for this chess piece, i.e. 1 on white team -1 on black.
- **ChessTable Table** [get, set]  
Returns the [ChessTable](#) that this [ChessPiece](#) belongs to.
- **ChessTableTile Tile** [get]  
Returns the Tile this [ChessPiece](#) lays on based on it's tile index.
- **ChessColor Color** [get, set]  
Returns the [ChessColor](#) team/color that this [ChessPiece](#) belongs to.
- **bool IsCaptured** [get, set]  
Returns true if this piece has been captured, otherwise false.
- **int MoveCount** [get, set]  
The number of moves this chess piece has made.
- **TileIndex TileIndex** [get, set]  
Represents the chess table tile this chess piece is on.

## Events

- **Action< [MoveInfo](#) > Moved**  
An event that is invoked when this chess piece is moved. Arg0: [MoveInfo](#) - The [MoveInfo](#) about the move.
- **Action< List< [ChessTableTile](#) > > MovesOverrideCallback**  
A simple callback that is invoked after potential moves for a chess piece are calculated, before the moves are validated. Arg): List of [ChessTableTiles](#) - the list of potential (not necessarily valid) moves for the chess piece.
- **Action< List< [AttackInfo](#) > > AttacksOverrideCallback**  
A simple callback that is invoked after potential attacks for a chess piece are calculated, before the attacks are validated. Arg): List of [AttackInfos](#) - the list of potential (not necessarily valid) attacks for the chess piece.
- **Action< [MoveInfo](#) > Captured**  
An event that is invoked when this chess piece is captured. Arg0: [MoveInfo](#) - The [MoveInfo](#) about the move the piece was captured on.
- **ValueActionRef< [ChessTableTile](#), bool > IsMoveValidCallback**  
Invoked during move validity checking to allow external programs to subscribe and override move validity for this chess piece. Arg0: [ChessTableTile](#) - The tile being moved to. Arg1: ref bool - A reference to a boolean value that controls whether or not a move is valid.
- **ValueActionRef< [AttackInfo](#), bool > IsAttackValidCallback**  
Invoked during attack validity checking to allow external programs to subscribe and override attack validity for this chess piece. Arg0: [AttackInfo](#) - The information about the attack. Arg1: ref bool - A reference to a boolean value that controls whether or not an attack is valid.

### 5.3.1 Detailed Description

An abstract component intended to be inherited from to define the behaviours of individual chess pieces.

Author: Mathew Aloisio

### 5.3.2 Constructor & Destructor Documentation

### 5.3.2.1 ChessPiece() [1/2]

```
ChessEngine.ChessPiece.ChessPiece (
    ChessTable pTable,
    ChessColor pColor,
    TileIndex pTileIndex )
```

Invoked whenever a [ChessPiece](#) instance is constructed.

## Parameters

<i>pTable</i>	The <a href="#">ChessTable</a> the piece belongs to.
<i>pColor</i>	The ChessColor of the team the piece belongs to.
<i>pTileIndex</i>	The <a href="#">TileIndex</a> the chess piece starts on.

**5.3.2.2 ChessPiece() [2/2]**

```
ChessEngine.ChessPiece.ChessPiece (
    ChessTable pTable,
    ChessPiece pOther )
```

Constructs a [ChessPiece](#) instance that is a copy of pOther except belonging to the pTable chess table.

## Parameters

<i>pTable</i>	
<i>pOther</i>	

**5.3.3 Member Function Documentation****5.3.3.1 Capture()**

```
void ChessEngine.ChessPiece.Capture (
    MoveInfo pMoveInfo )
```

Marks the [ChessPiece](#) as captured.

## Parameters

<i>pMoveInfo</i>	The <a href="#">MoveInfo</a> for the move that was piece was captured on.
------------------	---

**5.3.3.2 GenerateAttacksList()**

```
abstract List< AttackInfo > ChessEngine.ChessPiece.GenerateAttacksList (
    ChessTableTile pSpoofOccupiedTile = null,
    ChessTableTile pSpoofUnoccupiedTile = null ) [pure virtual]
```

Returns a list with the AttackInfos of all possible attacks for this piece. (These attacks are not necessarily valid and may be prevented by check, or other scenarios.)

**Parameters**

<i>pSpoofoffOccupiedTile</i>	The tile to test as 'occupied' or null.
<i>pSpoofoffUnoccupiedTile</i>	The tile to test as 'unoccupied' or null.

**Returns**

A list of AttackInfos for all possible attacks of this piece.

Implemented in [ChessEngine.Bishop](#), [ChessEngine.King](#), [ChessEngine.Knight](#), [ChessEngine.Pawn](#), [ChessEngine.Queen](#), and [ChessEngine.Rook](#).

**5.3.3.3 GenerateMovesList()**

```
abstract List< ChessTableTile > ChessEngine.ChessPiece.GenerateMovesList ( ) [pure virtual]
```

Returns a list with the ChessTabletiles of all possible moves for this piece. (These moves are not necessarily valid and may be prevented by check, or other scenarios.)

**Returns**

A list of ChessTableTiles for all possible moves of this piece.

Implemented in [ChessEngine.Bishop](#), [ChessEngine.King](#), [ChessEngine.Knight](#), [ChessEngine.Pawn](#), [ChessEngine.Queen](#), and [ChessEngine.Rook](#).

**5.3.3.4 GetAttacks()**

```
List< AttackInfo > ChessEngine.ChessPiece.GetAttacks (
    ChessTableTile pSpoofoffOccupiedTile = null,
    ChessTableTile pSpoofoffUnoccupiedTile = null )
```

Returns a list with the AttackInfos of all possible attacks for this piece.

**Parameters**

<i>pSpoofoffOccupiedTile</i>	The tile to test as 'occupied' or null.
<i>pSpoofoffUnoccupiedTile</i>	The tile to test as 'unoccupied' or null.

**Returns**

A list of ChessTableTiles for all possible attacks of this piece.

### 5.3.3.5 GetChessPieceType()

```
abstract ChessPieceType ChessEngine.ChessPiece.GetChessPieceType ( ) [pure virtual]
```

Returns the ChessPieceType for the [ChessPiece](#).

#### Returns

the ChessPieceType for the [ChessPiece](#).

Implemented in [ChessEngine.Bishop](#), [ChessEngine.King](#), [ChessEngine.Knight](#), [ChessEngine.Pawn](#), [ChessEngine.Queen](#), and [ChessEngine.Rook](#).

### 5.3.3.6 GetFENIdentifier()

```
abstract string ChessEngine.ChessPiece.GetFENIdentifier ( ) [pure virtual]
```

Returns the FEN identifier for the chess piece.

White Piece | Black Piece | Chess Piece P | p | [Pawn](#) N | n | [Knight](#) B | b | [Bishop](#) R | r | [Rook](#) Q | q | [Queen](#) K | k | [King](#)

#### Returns

a string representing the FEN identifier for the chess piece.

Implemented in [ChessEngine.Bishop](#), [ChessEngine.King](#), [ChessEngine.Knight](#), [ChessEngine.Pawn](#), [ChessEngine.Queen](#), and [ChessEngine.Rook](#).

### 5.3.3.7 GetMoves()

```
List< ChessTableTile > ChessEngine.ChessPiece.GetMoves ( )
```

Returns a list with the ChessTableTiles of all possible moves for this piece.

#### Returns

A list of ChessTableTiles for all possible moves of this piece.

### 5.3.3.8 GetValidAttacks()

```
List< AttackInfo > ChessEngine.ChessPiece.GetValidAttacks ( )
```

Uses overrideable [GetAttacks\(\)](#) to get a list of attacks and then filters them based on current check status, and whether or not the attack will put you in check.

#### Returns

A list of valid AttackInfos for attacks.

### 5.3.3.9 GetValidMoves()

```
List< ChessTableTile > ChessEngine.ChessPiece.GetValidMoves ( )
```

Uses overrideable [GetMove\(\)](#) to get a list of moves and then filters them based on current check status, and whether or not the move will put you in check.

#### Returns

A list of valid tiles to move to.

### 5.3.3.10 Move()

```
MoveInfo ChessEngine.ChessPiece.Move (
    TileIndex pToTileIndex,
    ChessPiece pAttackingPiece )
```

Moves the chess piece to the given [TileIndex](#), invokes the 'Moved' event.

#### Parameters

<i>pToTileIndex</i>	
<i>pAttackingPiece</i>	The <a href="#">ChessPiece</a> being attacked, or null if no attack (en passant excluded as it is handled specially by the pawns 'OnMoved' override).

#### Returns

[MoveInfo](#) containing information about the move.

### 5.3.3.11 OnPostMoved()

```
virtual void ChessEngine.ChessPiece.OnPostMoved (
    MoveInfo pMoveInfo ) [protected], [virtual]
```

Invoked just after the piece moves.

#### Parameters

<i>pMoveInfo</i>	The info for the move that was made.
------------------	--------------------------------------

Reimplemented in [ChessEngine.Pawn](#).

#### 5.3.3.12 OnPreMoved()

```
virtual void ChessEngine.ChessPiece.OnPreMoved (
    ref MoveInfo pMoveInfo ) [protected], [virtual]
```

Invoked just before the piece moves.

#### Parameters

<i>pMoveInfo</i>	A reference to the <a href="#">MoveInfo</a> . Can be changed before move is made.
------------------	---

Reimplemented in [ChessEngine.Pawn](#).

#### 5.3.3.13 OnTileIndexChanged()

```
virtual void ChessEngine.ChessPiece.OnTileIndexChanged (
    bool pWasEmpty,
    TileIndex pOldIndex ) [protected], [virtual]
```

Executed automatically when the [TileIndex](#) of the [ChessPiece](#) is changed.

#### Parameters

<i>pWasEmpty</i>	Was the space empty prior to the move?
<i>pOldIndex</i>	The previous index of the chess piece.

Reimplemented in [ChessEngine.King](#), and [ChessEngine.Pawn](#).

The documentation for this class was generated from the following file:

- ChessPiece.cs

## 5.4 ChessEngine.ChessTable Class Reference

The [ChessTable](#) component is to be attached to a gameObject on which 'ChessTableTiles' will be created.

## Public Member Functions

- [ChessTable](#) ([Instance](#) pChessInstance)  
*Invoked whenever a [ChessTable](#) instance is constructed.*
- [ChessTable](#) ([Instance](#) pChessInstance, [ChessTable](#) pOther)  
*Creates a copy of the chess table pOther that belongs to the [Instance](#) pChessInstance.*
- [ChessTable](#) ([Instance](#) pChessInstance, [SerializedChessTable](#) pSerializedTable)  
*Constructs a new chess table instance belonging to the pChessInstance [Instance](#). Copies the state of the provided pSerializedTable.*
- void [SetState](#) ([SerializedChessTable](#) pSerializedTable)  
*Overwrites the state of the Table with the information from pSerializedTable without needing to construct a new [ChessTable](#).*
- [ChessPiece](#) [CreateSerializedPiece](#) ([SerializedChessPiece](#) pSerializedPiece)  
*Creates a [ChessPiece](#) from a SerializedChessPiece and returns it.*
- void [ResetChessPieces](#) ()  
*When called removes all existing chess pieces and instantiates new ones at their proper initial spawn locations.*
- void [DestroyChessPieces](#) ()  
*Destroys all existing chess pieces.*
- bool [IsInCheck](#) ([ChessColor](#) pTeamColor)  
*Returns true if the team with the specified color is in check, otherwise false.*
- [ChessTableTile](#) [GetTile](#) ([TileIndex](#) pTileIndex)  
*Returns Tiles[pTileIndex.x][pTileIndex.y].*
- [ChessTableTile](#) [GetTile](#) (int pX, int pY)  
*Returns Tiles[pX][pY].*
- [ChessTableTile](#) [GetTileByID](#) (string pID)  
*Returns the [ChessTableTile](#) where ChessTableTile.TileIndex.GetTileID() == pID, otherwise null.*
- [ChessPiece](#) [CreateKing](#) ([TileIndex](#) pTileIndex, [ChessColor](#) pColor)  
*Instantiates a king and returns the [ChessPiece](#), or null.*
- [ChessPiece](#) [CreateQueen](#) ([TileIndex](#) pTileIndex, [ChessColor](#) pColor)  
*Instantiates a queen and returns the [ChessPiece](#), or null. This method is public because the [Pawn](#) needs to be able to replace itself with a queen.*
- [ChessPiece](#) [CreateBishop](#) ([TileIndex](#) pTileIndex, [ChessColor](#) pColor)  
*Instantiates a bishop and returns the [ChessPiece](#), or null.*
- [ChessPiece](#) [CreateKnight](#) ([TileIndex](#) pTileIndex, [ChessColor](#) pColor)  
*Instantiates a knight and returns the [ChessPiece](#), or null.*
- [ChessPiece](#) [CreateRook](#) ([TileIndex](#) pTileIndex, [ChessColor](#) pColor, bool plsKingSide)  
*Instantiates a rook and returns the [ChessPiece](#), or null.*
- [ChessPiece](#) [CreatePawn](#) ([TileIndex](#) pTileIndex, [ChessColor](#) pColor)  
*Instantiates a pawn and returns the [ChessPiece](#), or null.*
- [ChessPiece](#) [CreatePieceByType](#) ([ChessPieceType](#) pType, [TileIndex](#) pTileIndex, [ChessColor](#) pColor)  
*Instantiates a chess piece of the given type and returns it, or null.*
- void [DestroyPiece](#) ([ChessPiece](#) pPiece)  
*Removes the specified [ChessPiece](#) from the table if it is a part of it.*
- void [DestroyPieceByIndex](#) (int pPieceIndex)  
*Removes the [ChessPiece](#) in the specifeid index from the table.*
- [ChessPiece](#) [GetPieceByIndex](#) (int pIndex)  
*Returns the [ChessPiece](#) in the given index of the 'chess pieces' array. Note that 'PieceCount' can be used to get the total # of pieces on the table.*
- [King](#) [GetKing](#) ([ChessColor](#) pColor)  
*Returns a reference to the [King](#) piece of the team of the specified color.*
- [RookReferences](#) [GetRookReferences](#) ()



Returns a [RookReferences](#) object that contains references to all [Rook](#) pieces (captured or not) and organizes them based on their team color and 'king' or 'queen' side.

- void [DetermineRookSides](#) (bool pCanWhiteCastleKingside, bool pCanWhiteCastleQueenside, bool pCanBlackCastleKingside, bool pCanBlackCastleQueenside)

Generally used after loading an EPD state, this function will automatically attempt to determine the king and queen side Rooks or pick them at random. NOTE: Do not invoke this unless you know exactly why you want to invoke this. SIDE EFFECT: This will adjust the Rooks move counts based on the castle states.

- string [GenerateEPDString](#) ()

Generates an EPD string based on the current table layout and returns it.

- bool [SetStateToEPD](#) (string pEPD, bool pCanWhiteCastleKingside, bool pCanWhiteCastleQueenside, bool pCanBlackCastleKingside, bool pCanBlackCastleQueenside)

Sets the state of the chess table using the 'EPD' string given by pEPD.

## Protected Member Functions

- void [Protected\\_InitializeChessTable](#) ()

## Properties

- [ChessTableTile](#) [][] **Tiles** [get]  
A 2D array of ChessTableTiles.
- [Instance](#) **ChessInstance** [get]  
A reference to the [Instance](#) that this [ChessTable](#) belongs to.
- int **PieceCount** [get]  
Returns the # of chess pieces on the chess table.

## Events

- Action **ChessPiecesReset**  
An event that is invoked whenever the chess pieces on this table are reset.
- Action< [ChessPiece](#), [TileIndex](#) > **ChessPieceCreated**  
An event that is invoked whenever a chess piece is created. Arg0: [ChessPiece](#) - the [ChessPiece](#) that was being created. Arg1: [TileIndex](#) - the [TileIndex](#) the chess piece was created on.
- Action< [ChessPiece](#) > **ChessPieceDestroyed**  
An event that is invoked whenever a chess piece is removed from the table. Arg0: [ChessPiece](#) - the [ChessPiece](#) that is being destroyed.
- Action< [MoveInfo](#) > **ChessPieceMoved**  
An event that is invoked whenever a [ChessPiece](#) that belongs to this table moves for any reason. (Moved by 'Instance', moved by through ChessPiece.Move(...) directly, etc...) Unlike [Instance.ChessPieceMoved](#) this event is invoked regardless of what causes the chess piece to move except for the exception noted below.
- Action< [MoveInfo](#) > **PreChessPieceMoved**  
An event that is invoked just before a [ChessPiece](#) that belongs to this table moves for any reason. (Moved by 'Instance', moved by through ChessPiece.Move(...) directly, etc...) Unlike [Instance.PreChessPieceMoved](#) this event is invoked regardless of what causes the chess piece to move except for the exception noted below.
- Action< [ChessPiece](#), [ChessPiece](#), [TileIndex](#), [TileIndex](#) > **Castled**  
Invoked whenever chess pieces perform a castle move on this table.
- Action< [ChessPiece](#), [SerializedChessPiece](#) > **CreatedSerializedChessPiece**  
An event that is invoked whenever a chess piece is created via a serialized chess piece. Useful when extending serialization behavior outside of the library.
- Action< [ChessTable](#), [SerializedChessTable](#) > **LoadedSerializedTable**  
An event that is invoked whenever a chess table is constructed using a SerializedChessTable or has it's state set using one. Useful when extending serialization behavior outside of the library.

### 5.4.1 Detailed Description

The [ChessTable](#) component is to be attached to a gameObject on which 'ChessTableTiles' will be created.

NOTE: Chess tables are implemented with a bottom left origin meaning Tiles[0][0] is the bottom-left most corner when the white team is at the 'bottom' of the board. Author: Mathew Aloisio

### 5.4.2 Constructor & Destructor Documentation

#### 5.4.2.1 ChessTable() [1/3]

```
ChessEngine.ChessTable.ChessTable (
    Instance pChessInstance )
```

Invoked whenever a [ChessTable](#) instance is constructed.

##### Parameters

<i>pChessInstance</i>	The chess engine <a href="#">Instance</a> that this table belongs to.
-----------------------	---

#### 5.4.2.2 ChessTable() [2/3]

```
ChessEngine.ChessTable.ChessTable (
    Instance pChessInstance,
    ChessTable pOther )
```

Creates a copy of the chess table pOther that belongs to the [Instance](#) pChessInstance.

##### Parameters

<i>pChessInstance</i>	
<i>pOther</i>	

#### 5.4.2.3 ChessTable() [3/3]

```
ChessEngine.ChessTable.ChessTable (
    Instance pChessInstance,
    SerializedChessTable pSerializedTable )
```

Constructs a new chess table instance belonging to the pChessInstance [Instance](#). Copies the state of the provided pSerializedTable.

## Parameters

<i>pChessInstance</i>	The chess engine <a href="#">Instance</a> that this table belongs to.
<i>pSerializedTable</i>	The SerializedChessTable to copy the state of.

### 5.4.3 Member Function Documentation

#### 5.4.3.1 CreateBishop()

```
ChessPiece ChessEngine.ChessTable.CreateBishop (
    TileIndex pTileIndex,
    ChessColor pColor )
```

Instantiates a bishop and returns the [ChessPiece](#), or null.

## Parameters

<i>pTileIndex</i>	
<i>pColor</i>	

## Returns

[ChessPiece](#) component of the instantiated bishop or null.

#### 5.4.3.2 CreateKing()

```
ChessPiece ChessEngine.ChessTable.CreateKing (
    TileIndex pTileIndex,
    ChessColor pColor )
```

Instantiates a king and returns the [ChessPiece](#), or null.

## Parameters

<i>pTileIndex</i>	
<i>pColor</i>	

## Returns

[ChessPiece](#) component of the instantiated king or null.

#### 5.4.3.3 CreateKnight()

```
ChessPiece ChessEngine.ChessTable.CreateKnight (
    TileIndex pTileIndex,
    ChessColor pColor )
```

Instantiates a knight and returns the [ChessPiece](#), or null.

##### Parameters

<i>pTileIndex</i>	
<i>pColor</i>	

##### Returns

[ChessPiece](#) component of the instantiated knight or null.

#### 5.4.3.4 CreatePawn()

```
ChessPiece ChessEngine.ChessTable.CreatePawn (
    TileIndex pTileIndex,
    ChessColor pColor )
```

Instantiates a pawn and returns the [ChessPiece](#), or null.

##### Parameters

<i>pTileIndex</i>	
<i>pColor</i>	

##### Returns

[ChessPiece](#) component of the instantiated pawn or null.

#### 5.4.3.5 CreatePieceByType()

```
ChessPiece ChessEngine.ChessTable.CreatePieceByType (
    ChessPieceType pType,
    TileIndex pTileIndex,
    ChessColor pColor )
```

Instantiates a chess piece of the given type and returns it, or null.

## Parameters

<i>pType</i>	
<i>pTileIndex</i>	
<i>pColor</i>	

### 5.4.3.6 CreateQueen()

```
ChessPiece ChessEngine.ChessTable.CreateQueen (
    TileIndex pTileIndex,
    ChessColor pColor )
```

Instantiates a queen and returns the [ChessPiece](#), or null. This method is public because the [Pawn](#) needs to be able to replace itself with a queen.

## Parameters

<i>pTileIndex</i>	
<i>pColor</i>	

## Returns

[ChessPiece](#) component of the instantiated queen or null.

### 5.4.3.7 CreateRook()

```
ChessPiece ChessEngine.ChessTable.CreateRook (
    TileIndex pTileIndex,
    ChessColor pColor,
    bool pIsKingSide )
```

Instantiates a rook and returns the [ChessPiece](#), or null.

## Parameters

<i>pTileIndex</i>	
<i>pColor</i>	
<i>pIsKingSide</i>	true if the <a href="#">Rook</a> is the king side <a href="#">Rook</a> , otherwise false if it is the queen side <a href="#">Rook</a> .

## Returns

[ChessPiece](#) component of the instantiated rook or null.

#### 5.4.3.8 CreateSerializedPiece()

```
ChessPiece ChessEngine.ChessTable.CreateSerializedPiece (
    SerializedChessPiece pSerializedPiece )
```

Creates a [ChessPiece](#) from a SerializedChessPiece and returns it.

##### Parameters

<i>pSerializedPiece</i>	
-------------------------	--

##### Returns

the [ChessPiece](#) that was constructed from the SerializedChessPiece, otherwise null if failed.

#### 5.4.3.9 DestroyPiece()

```
void ChessEngine.ChessTable.DestroyPiece (
    ChessPiece pPiece )
```

Removes the specified [ChessPiece](#) from the table if it is a part of it.

##### Parameters

<i>pPiece</i>	
---------------	--

#### 5.4.3.10 DestroyPieceByIndex()

```
void ChessEngine.ChessTable.DestroyPieceByIndex (
    int pPieceIndex )
```

Removes the [ChessPiece](#) in the specifeid index from the table.

##### Parameters

<i>pPieceIndex</i>	
--------------------	--

#### 5.4.3.11 DetermineRookSides()

```
void ChessEngine.ChessTable.DetermineRookSides (
    bool pCanWhiteCastleKingside,
```

```

bool pCanWhiteCastleQueenside,
bool pCanBlackCastleKingside,
bool pCanBlackCastleQueenside )

```

Generally used after loading an EPD state, this function will automatically attempt to determine the king and queen side Rooks or pick them at random. NOTE: Do not invoke this unless you know exactly why you want to invoke this. SIDE EFFECT: This will adjust the Rooks move counts based on the castle states.

How [Rook](#) sides are determined:

- If a castle is possible we can guarantee the relevant [Rook](#) is in the initial position and can accurately set it to king or queenside.
- When a castle is not possible on both sides:
  1. The initial spawn locations will be checked for Rooks... if found they will be set to their relevant appropriate side.
  2. If no [Rook](#) found in both spawn positions the sides of the Rooks will be chosen at random.

#### Parameters

<i>pCanWhiteCastleKingside</i>	
<i>pCanWhiteCastleQueenside</i>	
<i>pCanBlackCastleKingside</i>	
<i>pCanBlackCastleQueenside</i>	

#### 5.4.3.12 GenerateEPDString()

```
string ChessEngine.ChessTable.GenerateEPDString ( )
```

Generates an EPD string based on the current table layout and returns it.

#### Returns

An EPD string based on the current table layout.

#### 5.4.3.13 GetKing()

```
King ChessEngine.ChessTable.GetKing (
    ChessColor pColor )
```

Returns a reference to the [King](#) piece of the team of the specified color.

#### Parameters

<i>pColor</i>	
---------------	--

**Returns**

a reference to the [King](#) piece of the team of the specified color.

**5.4.3.14 GetPieceByIndex()**

```
ChessPiece ChessEngine.ChessTable.GetPieceByIndex (
    int pIndex )
```

Returns the [ChessPiece](#) in the given index of the 'chess pieces' array. Note that 'PieceCount' can be used to get the total # of pieces on the table.

**Parameters**

<i>pIndex</i>	
---------------	--

**Returns**

the [ChessPiece](#) in the given index of the 'chess pieces' array.

**5.4.3.15 GetRookReferences()**

```
RookReferences ChessEngine.ChessTable.GetRookReferences ( )
```

Returns a [RookReferences](#) object that contains references to all [Rook](#) pieces (captured or not) and organizes them based on their team color and 'king' or 'queen' side.

**Returns**

a RookReferences object that contains references to all [Rook](#) pices (captured or not).

**5.4.3.16 GetTile() [1/2]**

```
ChessTableTile ChessEngine.ChessTable.GetTile (
    int pX,
    int pY )
```

Returns Tiles[pX][pY].

**Parameters**

<i>pX</i>	
<i>pY</i>	



**Returns**

A reference to the [ChessTableTile](#) in Tiles[pX][pY].

**5.4.3.17 GetTile()** [2/2]

```
ChessTableTile ChessEngine.ChessTable.GetTile (
    TileIndex pTileIndex )
```

Returns Tiles[pTileIndex.x][pTileIndex.y].

**Parameters**

<i>pTileIndex</i>	
-------------------	--

**Returns**

A reference to the [ChessTableTile](#) in Tiles[pTileIndex.x][pTileIndex.y].

**5.4.3.18 GetTileByID()**

```
ChessTableTile ChessEngine.ChessTable.GetTileByID (
    string pID )
```

Returns the [ChessTableTile](#) where ChessTableTile.TileIndex.GetTileID() == pID, otherwise null.

**Parameters**

<i>pID</i>	
------------	--

**Returns**

the [ChessTableTile](#) where ChessTableTile.TileIndex.GetTileID() == pID, otherwise null

**5.4.3.19 IsInCheck()**

```
bool ChessEngine.ChessTable.IsInCheck (
    ChessColor pTeamColor )
```

Returns true if the team with the specified color is in check, otherwise false.

## Parameters

<i>pTeamColor</i>	
-------------------	--

## Returns

true if the team *pTeamColor* is in check, otherwise false.

**5.4.3.20 SetState()**

```
void ChessEngine.ChessTable.SetState (
    SerializedChessTable pSerializedTable )
```

Overwrites the state of the Table with the information from *pSerializedTable* without needing to construct a new [ChessTable](#).

## Parameters

<i>pSerializedTable</i>	
-------------------------	--

**5.4.3.21 SetStateToEPD()**

```
bool ChessEngine.ChessTable.SetStateToEPD (
    string pEPD,
    bool pCanWhiteCastleKingside,
    bool pCanWhiteCastleQueenside,
    bool pCanBlackCastleKingside,
    bool pCanBlackCastleQueenside )
```

Sets the state of the chess table using the 'EPD' string given by *pEPD*.

## Parameters

<i>pEPD</i>	The EPD string that contains the chess table state.
<i>pCanWhiteCastleKingside</i>	
<i>pCanWhiteCastleQueenside</i>	
<i>pCanBlackCastleKingside</i>	
<i>pCanBlackCastleQueenside</i>	

**Returns**

true if the EPD string was successfully parsed, otherwise false.

**5.4.4 Event Documentation****5.4.4.1 Castled**

```
Action<ChessPiece, ChessPiece, TileIndex, TileIndex> ChessEngine.ChessTable.Castled
```

Invoked whenever chess pieces perform a castle move on this table.

NOTE: You can use the last invocation of the 'ChessPieceMoved' event to get the exact move details that led to the castle.

Arg0: [ChessPiece](#) - The king chess piece involved in the castle. Arg1: [ChessPiece](#) - The rook chess piece involved in the castle. Arg2: [TileIndex](#) - The [TileIndex](#) of the rook before castling. Arg3: [TileIndex](#) - The [TileIndex](#) of the rook after castling.

**5.4.4.2 ChessPieceMoved**

```
Action<MoveInfo> ChessEngine.ChessTable.ChessPieceMoved
```

An event that is invoked whenever a [ChessPiece](#) that belongs to this table moves for any reason. (Moved by 'Instance', moved by through ChessPiece.Move(...) directly, etc...) Unlike [Instance.ChessPieceMoved](#) this event is invoked regardless of what causes the chess piece to move except for the exception noted below.

EXCEPTION: This is not invoked when a rook is moved due to castling since it is not a 'move' for the rook, it is the kings move.

Arg0: [MoveInfo](#) - information about the move.

**5.4.4.3 CreatedSerializedChessPiece**

```
Action<ChessPiece, SerializedChessPiece> ChessEngine.ChessTable.CreatedSerializedChessPiece
```

An event that is invoked whenever a chess piece is created via a serialized chess piece. Useful when extending serialization behavior outside of the library.

Arg0: [ChessPiece](#) - The [ChessPiece](#) that was created. Arg1: [SerializedChessPiece](#) - The [SerializedChessPiece](#) the piece was created from.

**5.4.4.4 LoadedSerializedTable**

```
Action<ChessTable, SerializedChessTable> ChessEngine.ChessTable.LoadedSerializedTable
```

An event that is invoked whenever a chess table is constructed using a [SerializedChessTable](#) or has it's state set using one. Useful when extending serialization behavior outside of the library.

Arg0: [ChessTable](#) - The [ChessTable](#) the serialized table was loaded to. Arg1: [SerializedChessTable](#) - The [SerializedChessTable](#) that was 'loaded'.

#### 5.4.4.5 PreChessPieceMoved

Action<MoveInfo> ChessEngine.ChessTable.PreChessPieceMoved

An event that is invoked just before a [ChessPiece](#) that belongs to this table moves for any reason. (Moved by 'Instance', moved by through ChessPiece.Move(...) directly, etc...) Unlike [Instance.PreChessPieceMoved](#) this event is invoked regardless of what causes the chess piece to move except for the exception noted below.

EXCEPTION: This is not invoked when a rook is moved due to castling since it is not a 'move' for the rook, it is the kings move.

Arg0: [MoveInfo](#) - information about the upcoming move.

The documentation for this class was generated from the following file:

- ChessTable.cs

## 5.5 ChessEngine.ChessTableTile Class Reference

A [ChessTableTile](#) component is to be attached to each individual tile that makes up a chess table, each tile holds information about itself like offset.

### Public Member Functions

- [ChessTableTile](#) ([ChessTable](#) pTable, [TileIndex](#) pTileIndex, [ChessColor](#) pTileColor)  
*Constructs a [ChessTableTile](#) instance.*
- [ChessTableTile](#) ([ChessTable](#) pTable, [ChessTableTile](#) pOther)  
*Creates a copy of pOther except on the 'Table' pTable.*
- void [MovePieceToTile](#) ([ChessPiece](#) pPiece)  
*Moves the specified chess piece to this tile.*
- [ChessPiece](#) [GetPiece](#) ()  
*Returns the [ChessPiece](#) on this tile, otherwise null. Only valid when called after this tile has been placed on the board using 'PlaceAtIndex'.*

### Static Public Member Functions

- static bool [IsTileThreatened](#) (List< [AttackInfo](#) > pAttacks, [ChessTableTile](#) pTile)  
*Returns true if any attack in the pAttacks list is threatening the tile pTile (if a piece may be captured on said tile), otherwise false. This compares 'pTile' to [AttackInfo.attackTile](#).*
- static bool [IsTileAttackable](#) (List< [AttackInfo](#) > pAttacks, [ChessTableTile](#) pTile)  
*Returns true if any attack in the pAttacks list is able to move to the tile pTile, otherwise false. This compares 'pTile' to [AttackInfo.moveToTile](#).*

### Properties

- [TileIndex](#) [TileIndex](#) [get]  
*Returns the [TileIndex](#) this tile is located at.*
- [ChessTable](#) [Table](#) [get]  
*Returns the [ChessTable](#) this tile belongs to.*
- [ChessColor](#) [Color](#) [get]  
*Returns the [ChessColor](#) of this tile.*

### 5.5.1 Detailed Description

A [ChessTableTile](#) component is to be attached to each individual tile that makes up a chess table, each tile holds information about itself like offset.

Author: Mathew Aloisio

### 5.5.2 Constructor & Destructor Documentation

#### 5.5.2.1 ChessTableTile() [1/2]

```
ChessEngine.ChessTableTile.ChessTableTile (
    ChessTable pTable,
    TileIndex pTileIndex,
    ChessColor pTileColor )
```

Constructs a [ChessTableTile](#) instance.

##### Parameters

<i>pTable</i>	The <a href="#">ChessTable</a> the tile belongs to.
<i>pTileIndex</i>	The <a href="#">TileIndex</a> for the tile.
<i>pTileColor</i>	The ChessColor for the tile.

#### 5.5.2.2 ChessTableTile() [2/2]

```
ChessEngine.ChessTableTile.ChessTableTile (
    ChessTable pTable,
    ChessTableTile pOther )
```

Creates a copy of pOther except on the 'Table' pTable.

##### Parameters

<i>pTable</i>	
<i>pOther</i>	

### 5.5.3 Member Function Documentation

### 5.5.3.1 GetPiece()

```
ChessPiece ChessEngine.ChessTableTile.GetPiece ( )
```

Returns the [ChessPiece](#) on this tile, otherwise null. Only valid when called after this tile has been placed on the board using 'PlaceAtIndex'.

#### Returns

[ChessPiece](#) on this tile, otherwise null.

### 5.5.3.2 IsTileAttackable()

```
static bool ChessEngine.ChessTableTile.IsTileAttackable (
    List< AttackInfo > pAttacks,
    ChessTableTile pTile ) [static]
```

Returns true if any attack in the pAttacks list is able to move to the tile pTile, otherwise false. This compares 'pTile' to [AttackInfo.moveToTile](#).

#### Parameters

<i>pAttacks</i>	The valid 'enemy' attacks.
<i>pTile</i>	The tile being moved to.

#### Returns

true if any attack in the pAttacks list is able to move to the tile pTile, otherwise false.

### 5.5.3.3 IsTileThreatened()

```
static bool ChessEngine.ChessTableTile.IsTileThreatened (
    List< AttackInfo > pAttacks,
    ChessTableTile pTile ) [static]
```

Returns true if any attack in the pAttacks list is threatening the tile pTile (if a piece may be captured on said tile), otherwise false. This compares 'pTile' to [AttackInfo.attackTile](#).

#### Parameters

<i>pAttacks</i>	The valid 'enemy' attacks.
<i>pTile</i>	The tile being moved to.

**Returns**

true if any attack in the pAttacks list is threatening the tile pTile, otherwise false.

**5.5.3.4 MovePieceToTile()**

```
void ChessEngine.ChessTableTile.MovePieceToTile (
    ChessPiece pPiece )
```

Moves the specified chess piece to this tile.

**Parameters**

<i>pPiece</i>	The <a href="#">ChessPiece</a> to move.
---------------	---

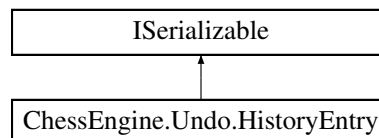
The documentation for this class was generated from the following file:

- ChessTableTile.cs

**5.6 ChessEngine.Undo.HistoryEntry Class Reference**

A class that contains data about a move so it can be undone or redone.

Inheritance diagram for ChessEngine.Undo.HistoryEntry:

**Public Member Functions**

- **HistoryEntry ()**  
*Default constructor.*

**Public Attributes**

- [SerializedChessInstance](#) **preMoveState**  
*The state for the game instance before the move.*
- [SerializedChessInstance](#) **postMoveState**  
*The state for the game instance after the move.*
- [TileIndex](#) **fromTileIndex**  
*The [TileIndex](#) the chess piece moved from.*
- [TileIndex](#) **toTileIndex**  
*The [TileIndex](#) the chess piece moved to.*

### 5.6.1 Detailed Description

A class that contains data about a move so it can be undone or redone.

Author: Mathew Aloisio

The documentation for this class was generated from the following file:

- HistoryEntry.cs

## 5.7 ChessEngine.Instance Class Reference

An instance of a chess engine. The core class that operates the chess engine.

### Public Member Functions

- **Instance** ()  
*Invoked when the chess engine [Instance](#) is constructed.*
- **Instance** (string pFEN)  
*Instantiates an [Instance](#) of the chess engine using the FEN string as the initial state.*
- **Instance** ([Instance](#) pOther)  
*Creates a unique copy of a chess instance.*
- **Instance** ([SerializedChessInstance](#) pSerializedInstance)  
*Constructs a new chess instance cloning the state of the provided pSerializedInstance. NOTE: The 'LastMove' property will not be loaded as it is not serialized due to difficulties in tracking 'capturedPiece' and 'piece' references since they may already be off the board or moved.*
- void **SetState** ([SerializedChessInstance](#) pSerializedInstance)  
*Sets the state of the chess instance using the information from pSerializedInstance without having to construct a new checkers [Instance](#).*
- void **ResetGame** ()  
*Resets the chess game starting a new game. Note that this method does not start the game, use [StartTurn\(\)](#) to start the first turn.*
- void **StartTurn** ()  
*Invokes the 'OnTurnStarted' callback. Useful to start an already set game without resetting it.*
- void **EndTurn** ([MoveInfo](#) pMove)  
*Ends a turn and runs win detection.*
- void **QueenPawn** ([Pawn](#) pPiece, [TileIndex](#) pOldTileIndex, bool pWasTileEmpty)  
*Queens a pawn while invoking relevant event(s) to provide an opportunity to override the behaviour.*
- virtual **GameOverReason IsGameOver** ([MoveInfo](#) pMove)  
*IsGameOver may be called to checks for game over events, returns GameOverReason on game over otherwise GameOverReason.NotOver if the game is not over.*
- string **GenerateFENEnPassantString** ()  
*Generates the en passant portion of a FEN string based on the current table layout and returns it.*
- string **GenerateFENCastleStringForRook** ([Rook](#) pRook)  
*Generates the FEN castle string character for a specific [Rook](#). If pRook is null an empty string will be returned.*
- string **GenerateFENCastleString** ()  
*Generates the castle portion of a FEN string based on the current table layout and returns it. Tracks whether the relevant rooks and kings have moved, not whether they can actually castle.*
- string **GenerateFENString** ()  
*Generates a FEN string based on the current table layout and returns it.*
- void **SetStateToFEN** (string pFEN)  
*Set the state of the chess instance using a FEN string.*



## Public Attributes

- [ChessColor](#) **turn**  
*The current turn.*

## Static Public Attributes

- static Action< [Instance](#) > **Initialized**  
*An event that is invoked whenever a new [Instance](#) is constructed. This is even invoked on instances constructed from serialized instances.*
- static Action< [Instance](#) > **Deinitialized**  
*An even that is invoked just before an [Instance](#) is destructed.*

## Protected Member Functions

- virtual void **OnGameOver** ([ChessColor](#) pTeam, [GameOverReason](#) pReason)  
*Called when the game ends, the winner is whoever's turn it currently is.*
- virtual void **OnTurnStarted** ([ChessColor](#) pTurn)  
*Invoked when a turn is started.*
- virtual void **OnTurnEnded** ([ChessColor](#) pLastTurn, [MoveInfo](#) pMove)  
*Invoked when a turn is ended.*
- void **Protected\_Initialize** ()  
*A private method that should be called at the start of all [Instance](#) constructors.*

## Properties

- [ChessTable](#) **Table** [get]  
*The [ChessTable](#) for this instance.*
- int **FullMoveCounter** [get]  
*The number of full moves this game. Incremented after black's move.*
- int **HalfMovesClock** [get]  
*The number of half moves since the last capture or pawn advance.*
- [MoveInfo](#) **LastMove** [get]  
*The [MoveInfo](#) from the last move or null if no moves have been made yet.*
- [Pawn](#) **EnPassantEligible** [get, set]  
*A reference to a [Pawn](#) eligible for en passant, or null.*

## Events

- Action **PreDestructed**  
*Invoked before the instance is destructed.*
- Action< [ChessColor](#), [MoveInfo](#) > **TurnEnded**  
*An event that is invoked when a turn is ended. Arg0: ChessColor - The color whose turn was ended. Arg1: [MoveInfo](#) - The [MoveInfo](#) from the turn that was ended.*
- Action< [ChessColor](#) > **TurnStarted**  
*An event that is invoked when a turn is started. Arg0: ChessColor - The team whose turn was started.*
- Action< [ChessColor](#), [GameOverReason](#) > **GameOver**  
*An event that is invoked when the game is finished. Arg0: ChessColor - The color whose turn it was when the game ended. Arg1: GameOverReason - The reason the game ended.*

- Action **PreGameReset**  
*An event that is invoked before the ChessGameManager is reset.*
- Action **PostGameReset**  
*An event that is invoked after the ChessGameManager is reset.*
- Action< [MoveInfo](#) > [ChessPieceMoved](#)  
*An event that is invoked when a chess piece that belongs to this [Instance](#) is moved using ChessPiece.Move(...)*
- Action< [MoveInfo](#) > [PreChessPieceMoved](#)  
*An event that is invoked just before a chess piece that belongs to this [Instance](#) is moved using ChessPiece.Move(...)*
- Action< [ChessPiece](#), [ChessPiece](#), [TileIndex](#), [TileIndex](#) > [Castled](#)  
*Invoked whenever chess pieces perform a castle move in this [Instance](#).*
- Action< [SerializedChessInstance](#) > [LoadedSerializedInstance](#)  
*An event that is invoked whenever a chess instance is loaded using a SerializedChessInstance. Useful for extending serialization behavior.*
- ActionRef< bool > [IsGameOverCallback](#)  
*Invoked during an 'IsGameOver' check to allow the game over state to be overridden using the event callback. Used to add additional 'Game Over' conditions. NOTE: To completely replace 'Game Over' conditions override the '[ChessEngine.Instance](#)' class and override the 'IsGameOver' virtual method.*
- ValueActionRef< [Pawn](#), [TileIndex](#), bool, bool > [QueeningPawnCallback](#)  
*Invoked just before a pawn is queened. Used to add custom 'queening' behaviour or to remove it completely. NOTE: To completely replace 'queening' behaviour set the reference 'bool' to true signalling the behaviour has been overridden – alternatively leaving the bool false (or not setting it) will allow you to expand on existing behaviour.*

### 5.7.1 Detailed Description

An instance of a chess engine. The core class that operates the chess engine.

Author: Mathew Aloisio

### 5.7.2 Constructor & Destructor Documentation

#### 5.7.2.1 Instance() [1/3]

```
ChessEngine.Instance.Instance (
    string pFEN )
```

Instantiates an [Instance](#) of the chess engine using the FEN string as the initial state.

**WARNING:** When creating an instance from a FEN string many things may not perfectly match the previous game as some information is not included in the FEN string. This includes:

1. The 'MoveCount' of pieces is not accurate.
2. Pawns that are not on their starting column will have their move count automatically incremented by 1.
3. Kings and Rooks may have their move count incremented to create conditions that match the 'castle string' from the FEN string.
4. If a castle is possible we can guarantee the relevant [Rook](#) is in the initial position and can accurately set it to king or queenside.
5. When a castle is not possible on both sides:
  - The initial spawn locations will be checked for Rooks... if found they will be set to their relevant appropriate side.
  - If no [Rook](#) found in both spawn positions the sides of the Rooks will be chosen at random.

## Parameters

<i>pFEN</i>	
-------------	--

**5.7.2.2 Instance()** [2/3]

```
ChessEngine.Instance.Instance (
    Instance pOther )
```

Creates a unique copy of a chess instance.

## Parameters

<i>pOther</i>	
---------------	--

**5.7.2.3 Instance()** [3/3]

```
ChessEngine.Instance.Instance (
    SerializedChessInstance pSerializedInstance )
```

Constructs a new chess instance cloning the state of the provided pSerializedInstance. NOTE: The 'LastMove' property will not be loaded as it is not serialized due to difficulties in tracking 'capturedPiece' and 'piece' references since they may already be off the board or moved.

## Parameters

<i>pSerializedInstance</i>	
----------------------------	--

**5.7.3 Member Function Documentation****5.7.3.1 EndTurn()**

```
void ChessEngine.Instance.EndTurn (
    MoveInfo pMove )
```

Ends a turn and runs win detection.

## Parameters

<i>pMove</i>	The move from the ending turn.
--------------	--------------------------------

### 5.7.3.2 GenerateFENCastleString()

```
string ChessEngine.Instance.GenerateFENCastleString ( )
```

Generates the castle portion of a FEN string based on the current table layout and returns it. Tracks whether the relevant rooks and kings have moved, not whether they can actually castle.

FORMAT: KQkq 1234

1. Can white [King](#) side [Rook](#) castle? K if yes, otherwise -
2. Can white [Queen](#) side [Rook](#) castle? Q if yes, otherwise -
3. Can black [King](#) side [Rook](#) castle? k if yes, otherwise -
4. Can black [Queen](#) side [Rook](#) castle? q if yes, otherwise -

#### Returns

The castle portion of a FEN string based on the current table layout.

### 5.7.3.3 GenerateFENCastleStringForRook()

```
string ChessEngine.Instance.GenerateFENCastleStringForRook (
    Rook pRook )
```

Generates the FEN castle string character for a specific [Rook](#). If pRook is null an empty string will be returned.

NOTE: This does not take into consideration whether or not the [King](#) has moved.

#### Parameters

<i>pRook</i>	
--------------	--

#### Returns

The FEN castle character for the [Rook](#).

### 5.7.3.4 GenerateFENEnPassantString()

```
string ChessEngine.Instance.GenerateFENEnPassantString ( )
```

Generates the en passant portion of a FEN string based on the current table layout and returns it.

#### Returns

The en passant portion of a FEN string based on the current table layout.

### 5.7.3.5 GenerateFENString()

```
string ChessEngine.Instance.GenerateFENString ( )
```

Generates a FEN string based on the current table layout and returns it.

#### Returns

A FEN string based on the current table layout.

### 5.7.3.6 IsGameOver()

```
virtual GameOverReason ChessEngine.Instance.IsGameOver (
    MoveInfo pMove ) [virtual]
```

IsGameOver may be called to checks for game over events, returns GameOverReason on game over otherwise GameOverReason.NotOver if the game is not over.

#### Parameters

<i>pMove</i>	
--------------	--

#### Returns

GameOverReason if the game is over, otherwise GameOverReason.NotOver if game is not over.

### 5.7.3.7 OnGameOver()

```
virtual void ChessEngine.Instance.OnGameOver (
    ChessColor pTeam,
    GameOverReason pReason ) [protected], [virtual]
```

Called when the game ends, the winner is whoever's turn it currently is.

#### Parameters

<i>pTeam</i>	The ChessColor of the team whose turn it was when the game ended.
<i>pReason</i>	The GameOverReason for the game ending.

### 5.7.3.8 OnTurnEnded()

```
virtual void ChessEngine.Instance.OnTurnEnded (
```

```

    ChessColor pLastTurn,
    MoveInfo pMove ) [protected], [virtual]

```

Invoked when a turn is ended.

#### Parameters

<i>pLastTurn</i>	The ChessColor whose turn ended.
<i>pMove</i>	The move the turn ended on.

### 5.7.3.9 OnTurnStarted()

```

virtual void ChessEngine.Instance.OnTurnStarted (
    ChessColor pTurn ) [protected], [virtual]

```

Invoked when a turn is started.

#### Parameters

<i>pTurn</i>	The color whose turn was started.
--------------	-----------------------------------

### 5.7.3.10 QueenPawn()

```

void ChessEngine.Instance.QueenPawn (
    Pawn pPiece,
    TileIndex pOldTileIndex,
    bool pWasTileEmpty )

```

Queens a pawn while invoking relevant event(s) to provide an opportunity to override the behaviour.

#### Parameters

<i>pPiece</i>	The <a href="#">Pawn</a> being queened.
<i>pOldTileIndex</i>	The tile index of the pawn before moving to the queening tile.
<i>pWasTileEmpty</i>	True if the tile the pawn moved to was empty, otherwise false.

### 5.7.3.11 SetState()

```

void ChessEngine.Instance.SetState (
    SerializedChessInstance pSerializedInstance )

```

Sets the state of the chess instance using the information from pSerializedInstance without having to construct a new checkers [Instance](#).

## Parameters

<i>pSerializedInstance</i>	
----------------------------	--

### 5.7.3.12 SetStateToFEN()

```
void ChessEngine.Instance.SetStateToFEN (
    string pFEN )
```

Set the state of the chess instance using a FEN string.

## Parameters

<i>pFEN</i>	
-------------	--

## 5.7.4 Event Documentation

### 5.7.4.1 Castled

```
Action<ChessPiece, ChessPiece, TileIndex, TileIndex> ChessEngine.Instance.Castled
```

Invoked whenever chess pieces perform a castle move in this [Instance](#).

NOTE: You can use the last invocation of the 'ChessPieceMoved' event to get the exact move details that led to the castle.

Arg0: [ChessPiece](#) - The king chess piece involved in the castle. Arg1: [ChessPiece](#) - The rook chess piece involved in the castle. Arg2: [TileIndex](#) - The [TileIndex](#) of the rook before castling. Arg3: [TileIndex](#) - The [TileIndex](#) of the rook after castling.

### 5.7.4.2 ChessPieceMoved

```
Action<MoveInfo> ChessEngine.Instance.ChessPieceMoved
```

An event that is invoked when a chess piece that belongs to this [Instance](#) is moved using ChessPiece.Move(...)

EXCEPTION: This is not invoked when a rook is moved due to castling since it is not a 'move' for the rook, it is the kings move.

Arg0: [MoveInfo](#) - Information about the move.

### 5.7.4.3 IsGameOverCallback

```
ActionRef<bool> ChessEngine.Instance.IsGameOverCallback
```

Invoked during an 'IsGameOver' check to allow the game over state to be overridden using the event callback. Used to add additional 'Game Over' conditions. NOTE: To completely replace 'Game Over' conditions override the '[ChessEngine.Instance](#)' class and override the 'IsGameOver' virtual method.

Arg0: ref bool - A reference to a boolean value that will flip the game to a 'Unknown' game over reason state if it becomes true.

### 5.7.4.4 LoadedSerializedInstance

```
Action<SerializedChessInstance> ChessEngine.Instance.LoadedSerializedInstance
```

An event that is invoked whenever a chess instance is loaded using a SerializedChessInstance. Useful for extending serialization behavior.

Arg0: SerializedChessInstance - The SerializedChessInstance the [Instance](#) was loaded from.

### 5.7.4.5 PreChessPieceMoved

```
Action<MoveInfo> ChessEngine.Instance.PreChessPieceMoved
```

An event that is invoked just before a chess piece that belongs to this [Instance](#) is moved using ChessPiece.Move(...)

EXCEPTION: This is not invoked when a rook is moved due to castling since it is not a 'move' for the rook, it is the kings move.

Arg0: [MoveInfo](#) - Information about the upcoming move.

### 5.7.4.6 QueeningPawnCallback

```
ValueActionRef<Pawn, TileIndex, bool, bool> ChessEngine.Instance.QueeningPawnCallback
```

Invoked just before a pawn is queened. Used to add custom 'queening' behaviour or to remove it completely. NOTE: To completely replace 'queening' behaviour set the reference 'bool' to true signalling the behaviour has been overridden – alternatively leaving the bool false (or not setting it) will allow you to expand on existing behaviour.

Arg0: [Pawn](#) - The pawn that is being queened. Arg1: [TileIndex](#) - The old tile index of the pawn before moving to the queening tile. Arg2: bool - A boolean value that represents whether or not the tile that was moved to was empty or not. If true the tile was empty, if false it was not. Arg3: ref bool - A reference to a boolean value that when true signals the queening behaviour has been overridden, otherwise when false (default) the default behaviour still executes.

The documentation for this class was generated from the following file:

- Instance.cs



## 5.8 ChessEngine.Undo.InstanceHistory Class Reference

A class that tracks moves in a chess engine [Instance](#) and allows for undoing and redoing of moves.

### Public Member Functions

- **InstanceHistory** ()  
*Constructs an [InstanceHistory](#) instance that is uninitialized (no '[Instance](#)' reference set).*
- **InstanceHistory** ([Instance](#) pInstance)  
*Constructs an [InstanceHistory](#) instance that tracks history for the specified chess engine [Instance](#), pInstance.*
- **InstanceHistory** ([Instance](#) pInstance, [SerializedInstanceHistory](#) pSerialized)
- void **ClearHistory** ()  
*Clears all tracked instance history.*
- void **ClearUndoneMoves** ()  
*Clears all the tracked instance undone move history.*
- [HistoryEntry](#) **PeekMove** ()  
*Returns the [HistoryEntry](#) at the top of the move history stack.*
- [HistoryEntry](#) **UndoMove** ()  
*Pops the top-most [MoveInfo](#) from the move history stack and returns it.*
- [HistoryEntry](#) **PeekUndoneMove** ()  
*Returns a reference to the [HistoryEntry](#) at the top of the undone moves stack.*
- [HistoryEntry](#) **RedoMove** ()  
*Redoes the last undid move.*

### Protected Member Functions

- void **PushMoveHistory** ([MoveInfo](#) pMove)  
*Pushes a [MoveInfo](#), pMove, to the top of the move history stack.*
- void **OnInstanceSet** ([Instance](#) pInstance)  
*Invoked whenever the '[Instance](#)' property of this [InstanceHistory](#) instance is set to any value (including a null one).*
- void **OnInstanceUnset** ([Instance](#) pInstance)  
*Invoked whenever the '[Instance](#)' property of this [InstanceHistory](#) object is changed from some non-null value to anything else.*
- void **OnPreChessPieceMoved** ([MoveInfo](#) pMoveInfo)  
*Invoked just before a chess piece for the relevant chess engine [Instance](#) is moved.*
- void **OnGameOver** ([ChessColor](#) pColor, [GameOverReason](#) pReason)  
*Invoked when the chess game is over. (Use '[LastMove](#)') to get the game ending move.*
- void **OnTurnEnded** ([ChessColor](#) pLastTurn, [MoveInfo](#) pMoveInfo)  
*Invoked after a turn is ended.*

### Protected Attributes

- bool **m\_SkipNextMove**  
*A boolean that tracks whether or not the next move should be skipped and not added to history.*
- [SerializedChessInstance](#) **m\_PreMoveState** = null  
*The game state before the last move.*

## Properties

- **Instance Instance** [get, set]  
A reference to the 'Instance' of the chess engine this component is tracking history for, otherwise null.
- **int MoveHistoryCount** [get]  
Returns the number of HistoryEntrys there are in the 'move history' array.
- **bool TrackHistory = true** [get, set]  
Should this class instance track history? If true history is tracked, otherwise not tracked.
- **int UndoneMovesCount** [get]  
Returns the number of HistoryEntrys there are in the 'undone moves' array.
- **bool TrackUndoneMoves = true** [get, set]  
Should this class instance track undone moves for 'redo' functionality? If true undone moves are tracked, otherwise not tracked.

## Events

- **Action< Instance > InstanceSet**  
Invoked whenever the 'Instance' property of this component is set to any value (including a null value). Arg0: Instance - the chess engine Instance that is now being referenced by the instance history tracker or null.
- **Action< Instance > InstanceUnset**  
Invoked whenever the 'Instance' property of this component is unset from some non-null value to anything. Arg0: Instance - the chess engine Instance that was being referenced by the instance history tracker.
- **Action< Instance, HistoryEntry > PreMoveUndone**  
Invoked just before a move is undone by this InstanceHistory instance just after it is 'popped' from the moves stack. Arg0: Instance - The InstanceHistory object that is about to the move. Arg1: HistoryEntry - The HistoryEntry containing information about the undone move.
- **Action< Instance, HistoryEntry > MoveUndone**  
Invoked after a move is undone by this InstanceHistory instance. Arg0: Instance - The InstanceHistory object that undid the move. Arg1: HistoryEntry - The HistoryEntry containing information about the undone move.
- **Action< Instance, HistoryEntry > PreMoveRedone**  
Invoked just before a move is redone by this InstanceHistory instance just after it is 'popped' from the undone moves stack. Arg0: Instance - The InstanceHistory object that is about to redo the move. Arg1: HistoryEntry - The HistoryEntry containing information about the redone move.
- **Action< Instance, HistoryEntry > MoveRedone**  
Invoked after a move is redone by this InstanceHistory instance. Arg0: Instance - The InstanceHistory object that redid the move. Arg1: HistoryEntry - The HistoryEntry containing information about the redone move.
- **Action< Instance > MoveHistoryEmptied**  
Invoked whenever the move history stack becomes empty. Arg0: Instance - the chess engine Instance that move history is being tracked for.
- **Action< Instance > MoveHistoryValid**  
Invoked whenever the move history stack goes from empty to non-empty. Arg0: Instance - the chess engine Instance that move history is being tracked for.
- **Action< Instance > UndoHistoryEmptied**  
Invoked whenever the undo history stack becomes empty. Arg0: Instance - the chess engine Instance that undo history is being tracked for.
- **Action< Instance > UndoHistoryValid**  
Invoked whenever the undo history stack goes from empty to non-empty. Arg0: Instance - the chess engine Instance that undo history is being tracked for.

### 5.8.1 Detailed Description

A class that tracks moves in a chess engine Instance and allows for undoing and redoing of moves.

Author: Mathew Aloisio

## 5.8.2 Constructor & Destructor Documentation

### 5.8.2.1 InstanceHistory()

```
ChessEngine.Undo.InstanceHistory.InstanceHistory (
    Instance pInstance )
```

Constructs an [InstanceHistory](#) instance that tracks history for the specified chess engine [Instance](#), pInstance.

#### Parameters

<i>pInstance</i>	
------------------	--

## 5.8.3 Member Function Documentation

### 5.8.3.1 OnGameOver()

```
void ChessEngine.Undo.InstanceHistory.OnGameOver (
    ChessColor pColor,
    GameOverReason pReason ) [protected]
```

Invoked when the chess game is over. (Use 'LastMove') to get the game ending move.

#### Parameters

<i>pColor</i>	The color whose turn it was when the game ended.
<i>pReason</i>	The reason the game is over.

### 5.8.3.2 OnInstanceSet()

```
void ChessEngine.Undo.InstanceHistory.OnInstanceSet (
    Instance pInstance ) [protected]
```

Invoked whenever the '[Instance](#)' property of this [InstanceHistory](#) instance is set to any value (including a null one).

#### Parameters

<i>pInstance</i>	
------------------	--

### 5.8.3.3 OnInstanceUnset()

```
void ChessEngine.Undo.InstanceHistory.OnInstanceUnset (
    Instance pInstance ) [protected]
```

Invoked whenever the 'Instance' property of this [InstanceHistory](#) object is changed from some non-null value to anything else.

#### Parameters

<i>pInstance</i>	
------------------	--

### 5.8.3.4 OnPreChessPieceMoved()

```
void ChessEngine.Undo.InstanceHistory.OnPreChessPieceMoved (
    MoveInfo pMoveInfo ) [protected]
```

Invoked just before a chess piece for the relevant chess engine [Instance](#) is moved.

#### Parameters

<i>pMoveInfo</i>	
------------------	--

### 5.8.3.5 OnTurnEnded()

```
void ChessEngine.Undo.InstanceHistory.OnTurnEnded (
    ChessColor pLastTurn,
    MoveInfo pMoveInfo ) [protected]
```

Invoked after a turn is ended.

#### Parameters

<i>pLastTurn</i>	
<i>pMoveInfo</i>	

### 5.8.3.6 PeekMove()

```
HistoryEntry ChessEngine.Undo.InstanceHistory.PeekMove ( )
```

Returns the [HistoryEntry](#) at the top of the move history stack.

**Returns**

the [HistoryEntry](#) at the top of the move history s tack.

**5.8.3.7 PeekUndoneMove()**

```
HistoryEntry ChessEngine.Undo.InstanceHistory.PeekUndoneMove ( )
```

Returns a reference to the [HistoryEntry](#) at the top of the undone moves stack.

**Returns**

A reference to the [HistoryEntry](#) a the top of the undone moves stack.

**5.8.3.8 PushMoveHistory()**

```
void ChessEngine.Undo.InstanceHistory.PushMoveHistory (
    MoveInfo pMove ) [protected]
```

Pushes a [MoveInfo](#), pMove, to the top of the move history stack.

**Parameters**

<i>pMove</i>	
--------------	--

**5.8.3.9 RedoMove()**

```
HistoryEntry ChessEngine.Undo.InstanceHistory.RedoMove ( )
```

Redoes the last undid move.

**Returns**

[HistoryEntry](#) for the move that was redone.

**5.8.3.10 UndoMove()**

```
HistoryEntry ChessEngine.Undo.InstanceHistory.UndoMove ( )
```

Pops the top-most [MoveInfo](#) from the move history stack and returns it.

**Returns**

the [MoveInfo](#) at the start of the move history stack.

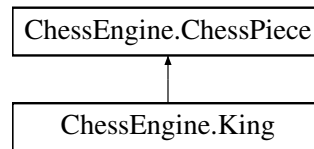
The documentation for this class was generated from the following file:

- InstanceHistory.cs

## 5.9 ChessEngine.King Class Reference

Implementation of a Chess [King](#).

Inheritance diagram for ChessEngine.King:



### Public Member Functions

- **King** ([ChessTable](#) pTable, [ChessColor](#) pColor, [TileIndex](#) pTileIndex)
- **King** ([ChessTable](#) pTable, [ChessPiece](#) pOther)
- override List< [ChessTableTile](#) > [GenerateMovesList](#) ()  
*Returns a list with the ChessTableTiles of all possible moves for this piece.*
- override List< [AttackInfo](#) > [GenerateAttacksList](#) ([ChessTableTile](#) pSpoofoccupiedTile, [ChessTableTile](#) pSpoofoUnoccupiedTile)  
*Returns a list with the AttackInfos of all possible attacks for this piece.*
- override string [GetFENIdentifier](#) ()  
*Returns the FEN identifier for the chess piece.*
- override [ChessPieceType](#) [GetChessPieceType](#) ()  
*Returns the ChessPieceType for this ChessPiece.*

### Protected Member Functions

- override void [OnTileIndexChanged](#) (bool pWasEmpty, [TileIndex](#) pOldIndex)  
*Executed automatically when the TileIndex of the ChessPiece is changed.*
- virtual void [OnCastled](#) ([ChessPiece](#) pRook, [TileIndex](#) pPreCastleTile, [TileIndex](#) pPostCastleTile)  
*Invoked after this King castles with a Rook. Note that the kings move information can be obtained from the last invocation of the 'ChessPieceMoved' event.*

### Events

- Action< [ChessPiece](#), [TileIndex](#), [TileIndex](#) > [Castled](#)  
*An event that is invoked whenever the King castles with a rook.*

### Additional Inherited Members

#### 5.9.1 Detailed Description

Implementation of a Chess [King](#).

Author: Mathew Aloisio

## 5.9.2 Member Function Documentation

### 5.9.2.1 GenerateAttacksList()

```
override List< AttackInfo > ChessEngine.King.GenerateAttacksList (
    ChessTableTile pSpoofoccupiedTile,
    ChessTableTile pSpoofunoccupiedTile ) [virtual]
```

Returns a list with the AttackInfos of all possible attacks for this piece.

#### Returns

A list of AttackInfos for all possible attacks of this piece.

Implements [ChessEngine.ChessPiece](#).

### 5.9.2.2 GenerateMovesList()

```
override List< ChessTableTile > ChessEngine.King.GenerateMovesList ( ) [virtual]
```

Returns a list with the ChessTableTiles of all possible moves for this piece.

#### Returns

A list of ChessTableTiles for all possible moves of this piece.

Implements [ChessEngine.ChessPiece](#).

### 5.9.2.3 GetChessPieceType()

```
override ChessPieceType ChessEngine.King.GetChessPieceType ( ) [virtual]
```

Returns the ChessPieceType for this [ChessPiece](#).

#### Returns

the ChessPieceType for this [ChessPiece](#).

Implements [ChessEngine.ChessPiece](#).

#### 5.9.2.4 GetFENIdentifier()

```
override string ChessEngine.King.GetFENIdentifier ( ) [virtual]
```

Returns the FEN identifier for the chess piece.

White Piece | Black Piece | Chess Piece P | p | Pawn N | n | Knight B | b | Bishop R | r | Rook Q | q | Queen K | k | King

##### Returns

a string representing the FEN identifier for the chess piece.

Implements [ChessEngine.ChessPiece](#).

#### 5.9.2.5 OnCastled()

```
virtual void ChessEngine.King.OnCastled (
    ChessPiece pRook,
    TileIndex pPreCastleTile,
    TileIndex pPostCastleTile ) [protected], [virtual]
```

Invoked after this [King](#) castles with a [Rook](#). Note that the kings move information can be obtained from the last invocation of the 'ChessPieceMoved' event.

##### Parameters

<i>pRook</i>	A reference to the rook <a href="#">ChessPiece</a> that the <a href="#">King</a> castled with.
<i>pPreCastleTile</i>	The <a href="#">TileIndex</a> the rook was on before castling.
<i>pPostCastleTile</i>	The <a href="#">TileIndex</a> the rook is on after castling.

#### 5.9.2.6 OnTileIndexChanged()

```
override void ChessEngine.King.OnTileIndexChanged (
    bool pWasEmpty,
    TileIndex pOldIndex ) [protected], [virtual]
```

Executed automatically when the [TileIndex](#) of the [ChessPiece](#) is changed.

##### Parameters

<i>pWasEmpty</i>	Was the space empty prior to the move?
<i>pOldIndex</i>	The previous index of the chess piece.

Reimplemented from [ChessEngine.ChessPiece](#).



### 5.9.3 Event Documentation

#### 5.9.3.1 Castled

Action<ChessPiece, TileIndex, TileIndex> ChessEngine.King.Castled

An event that is invoked whenever the [King](#) castles with a rook.

Arg0: [ChessPiece](#) - A reference to the [Rook](#) that castled with this [King](#). Arg1: [TileIndex](#) - The original tile index for the [Rook](#) before castling. Arg2: [TileIndex](#) - The new tile index for the [Rook](#) after castling.

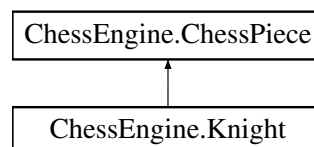
The documentation for this class was generated from the following file:

- King.cs

## 5.10 ChessEngine.Knight Class Reference

Implementation of a Chess [Knight](#).

Inheritance diagram for ChessEngine.Knight:



### Public Member Functions

- **Knight** ([ChessTable](#) pTable, [ChessColor](#) pColor, [TileIndex](#) pTileIndex)
- **Knight** ([ChessTable](#) pTable, [ChessPiece](#) pOther)
- override List< [ChessTableTile](#) > [GenerateMovesList](#) ()  
*Returns a list with the ChessTableTiles of all possible moves for this piece.*
- override List< [AttackInfo](#) > [GenerateAttacksList](#) ([ChessTableTile](#) pSpoofOccupiedTile, [ChessTableTile](#) p← SpoofUnoccupiedTile)  
*Returns a list with the AttackInfos of all possible attacks for this piece.*
- override string [GetFENIdentifier](#) ()  
*Returns the FEN identifier for the chess piece.*
- override [ChessPieceType](#) [GetChessPieceType](#) ()  
*Returns the ChessPieceType for this ChessPiece.*

### Additional Inherited Members

#### 5.10.1 Detailed Description

Implementation of a Chess [Knight](#).

Author: Mathew Aloisio

## 5.10.2 Member Function Documentation

### 5.10.2.1 GenerateAttacksList()

```
override List< AttackInfo > ChessEngine.Knight.GenerateAttacksList (
    ChessTableTile pSpoofOccupiedTile,
    ChessTableTile pSpoofUnoccupiedTile ) [virtual]
```

Returns a list with the AttackInfos of all possible attacks for this piece.

#### Returns

A list of AttackInfos for all possible attacks of this piece.

Implements [ChessEngine.ChessPiece](#).

### 5.10.2.2 GenerateMovesList()

```
override List< ChessTableTile > ChessEngine.Knight.GenerateMovesList ( ) [virtual]
```

Returns a list with the ChessTableTiles of all possible moves for this piece.

#### Returns

A list of ChessTableTiles for all possible moves of this piece.

Implements [ChessEngine.ChessPiece](#).

### 5.10.2.3 GetChessPieceType()

```
override ChessPieceType ChessEngine.Knight.GetChessPieceType ( ) [virtual]
```

Returns the ChessPieceType for this [ChessPiece](#).

#### Returns

the ChessPieceType for this [ChessPiece](#).

Implements [ChessEngine.ChessPiece](#).

#### 5.10.2.4 GetFENIdentifier()

```
override string ChessEngine.Knight.GetFENIdentifier ( ) [virtual]
```

Returns the FEN identifier for the chess piece.

White Piece | Black Piece | Chess Piece P | p | [Pawn](#) N | n | [Knight](#) B | b | [Bishop](#) R | r | [Rook](#) Q | q | [Queen](#) K | k | [King](#)

Returns

a string representing the FEN identifier for the chess piece.

Implements [ChessEngine.ChessPiece](#).

The documentation for this class was generated from the following file:

- Knight.cs

## 5.11 ChessEngine.MoveData Class Reference

Move data contains simply the 'from' tile index, the 'to' tile index, and a readonly boolean 'isAttack' that tracks whether the move is an attack or just a move.

### Public Member Functions

- **MoveData ()**  
*Constructs a [MoveData](#) instance and sets the readonly 'isCapture' field to false.*
- **MoveData (TileIndex pCaptureTileIndex)**  
*Constructs a [MoveData](#) instance and sets the readonly 'isCapture' field to true and stores the 'capture tile index'.*

### Public Attributes

- readonly bool **isCapture**  
*Returns true if the move is a capture, otherwise false.*
- readonly [TileIndex](#) **captureTileIndex**  
*Only relevant if 'isCapture' is true. Returns the tile index of the piece being captured.*
- [TileIndex](#) **fromTileIndex**  
*The tile index the move is from.*
- [TileIndex](#) **toTileIndex**  
*The tile index the move is to.*

#### 5.11.1 Detailed Description

Move data contains simply the 'from' tile index, the 'to' tile index, and a readonly boolean 'isAttack' that tracks whether the move is an attack or just a move.

#### 5.11.2 Constructor & Destructor Documentation

##### 5.11.2.1 MoveData()

```
ChessEngine.MoveData.MoveData (
    TileIndex pCaptureTileIndex )
```

Constructs a [MoveData](#) instance and sets the readonly 'isCapture' field to true and stores the 'capture tile index'.

## Parameters

<i>pCaptureTileIndex</i>	
--------------------------	--

The documentation for this class was generated from the following file:

- MoveData.cs

## 5.12 ChessEngine.MoveInfo Class Reference

A class containing information about a move.

### Public Attributes

- [ChessPiece](#) **piece**  
*The chess piece moved in the move.*
- [TileIndex](#) **fromTileIndex**  
*The [TileIndex](#) the chess piece moved from.*
- [TileIndex](#) **toTileIndex**  
*The [TileIndex](#) the chess piece moved to.*
- [ChessPiece](#) **capturedPiece**  
*The [ChessPiece](#) that was captured in the move, otherwise null if none was captured*
- [TileIndex](#) **capturedTileIndex**  
*Only valid when capturedPiece is non-null, the tile index the captured piece was taken from. (Exists due to en-passant allowing take without ending up on attacked tile.)*

### 5.12.1 Detailed Description

A class containing information about a move.

Author: Mathew Aloisio

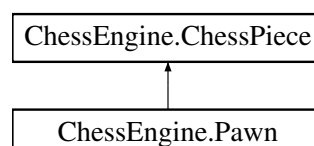
The documentation for this class was generated from the following file:

- MoveInfo.cs

## 5.13 ChessEngine.Pawn Class Reference

Implementation of a Chess [Pawn](#).

Inheritance diagram for ChessEngine.Pawn:



## Public Member Functions

- **Pawn** ([ChessTable](#) pTable, [ChessColor](#) pColor, [TileIndex](#) pTileIndex)
- **Pawn** ([ChessTable](#) pTable, [ChessPiece](#) pOther)
- bool **IsOnStartingRow** ()  
*Returns true if the [Pawn](#) is on its starting row, otherwise false.*
- override List< [ChessTableTile](#) > **GenerateMovesList** ()  
*Returns a list with the [ChessTableTiles](#) of all possible moves for this piece.*
- override List< [AttackInfo](#) > **GenerateAttacksList** ([ChessTableTile](#) pSpoofofOccupiedTile, [ChessTableTile](#) pSpoofofUnoccupiedTile)  
*Returns a list with the [AttackInfos](#) of all possible attacks for this piece.*
- override string **GetFENIdentifier** ()  
*Returns the FEN identifier for the chess piece.*
- override [ChessPieceType](#) **GetChessPieceType** ()  
*Returns the [ChessPieceType](#) for this [ChessPiece](#).*

## Public Attributes

- [Pawn](#) **enPassanting**  
*Reference to a pawn that is currently being taken en passant by this pawn or null.*

## Static Public Attributes

- const int **SPAWN\_ROW\_INDEX\_WHITE** = 1  
*The index of the row white pawns spawn in.*
- const int **SPAWN\_ROW\_INDEX\_BLACK** = 6  
*The index of the row black pawns spawn in.*

## Protected Member Functions

- override void **OnPreMoved** (ref [MoveInfo](#) pMoveInfo)  
*Invoked just before the [Pawn](#) is moved using the 'Move(...)' method.*
- override void **OnPostMoved** ([MoveInfo](#) pMoveInfo)  
*Invoked just after the pawn is moved using the Move(...) method.*
- override void **OnTileIndexChanged** (bool pWasEmpty, [TileIndex](#) pOldIndex)  
*Executed automatically when the [TileIndex](#) of the [ChessPiece](#) is changed.*

## Additional Inherited Members

### 5.13.1 Detailed Description

Implementation of a Chess [Pawn](#).

Author: Mathew Aloisio

### 5.13.2 Member Function Documentation

### 5.13.2.1 GenerateAttacksList()

```
override List< AttackInfo > ChessEngine.Pawn.GenerateAttacksList (
    ChessTableTile pSpoofOccupiedTile,
    ChessTableTile pSpoofUnoccupiedTile ) [virtual]
```

Returns a list with the AttackInfos of all possible attacks for this piece.

#### Returns

A list of AttackInfos for all possible attacks of this piece.

Implements [ChessEngine.ChessPiece](#).

### 5.13.2.2 GenerateMovesList()

```
override List< ChessTableTile > ChessEngine.Pawn.GenerateMovesList ( ) [virtual]
```

Returns a list with the ChessTableTiles of all possible moves for this piece.

#### Returns

A list of ChessTableTiles for all possible moves of this piece.

Implements [ChessEngine.ChessPiece](#).

### 5.13.2.3 GetChessPieceType()

```
override ChessPieceType ChessEngine.Pawn.GetChessPieceType ( ) [virtual]
```

Returns the ChessPieceType for this [ChessPiece](#).

#### Returns

the ChessPieceType for this [ChessPiece](#).

Implements [ChessEngine.ChessPiece](#).

#### 5.13.2.4 GetFENIdentifier()

```
override string ChessEngine.Pawn.GetFENIdentifier ( ) [virtual]
```

Returns the FEN identifier for the chess piece.

White Piece | Black Piece | Chess Piece P | p | [Pawn](#) N | n | [Knight](#) B | b | [Bishop](#) R | r | [Rook](#) Q | q | [Queen](#) K | k | [King](#)

##### Returns

a string representing the FEN identifier for the chess piece.

Implements [ChessEngine.ChessPiece](#).

#### 5.13.2.5 IsOnStartingRow()

```
bool ChessEngine.Pawn.IsOnStartingRow ( )
```

Returns true if the [Pawn](#) is on its starting row, otherwise false.

##### Returns

true if the [Pawn](#) is on its starting row, otherwise false.

#### 5.13.2.6 OnPostMoved()

```
override void ChessEngine.Pawn.OnPostMoved (
    MoveInfo pMoveInfo ) [protected], [virtual]
```

Invoked just after the pawn is moved using the Move(...) method.

##### Parameters

<a href="#">pMoveInfo</a>	
---------------------------	--

Reimplemented from [ChessEngine.ChessPiece](#).

#### 5.13.2.7 OnPreMoved()

```
override void ChessEngine.Pawn.OnPreMoved (
    ref MoveInfo pMoveInfo ) [protected], [virtual]
```

Invoked just before the [Pawn](#) is moved using the 'Move(...)' method.

## Parameters

<code>pMoveInfo</code>	
------------------------	--

Reimplemented from [ChessEngine.ChessPiece](#).

### 5.13.2.8 OnTileIndexChanged()

```
override void ChessEngine.Pawn.OnTileIndexChanged (
    bool pWasEmpty,
    TileIndex pOldIndex ) [protected], [virtual]
```

Executed automatically when the [TileIndex](#) of the [ChessPiece](#) is changed.

## Parameters

<code>pWasEmpty</code>	Was the space empty prior to the move?
<code>pOldIndex</code>	The previous index of the chess piece.

Reimplemented from [ChessEngine.ChessPiece](#).

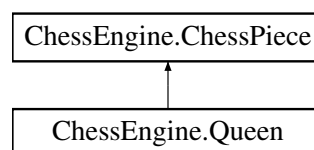
The documentation for this class was generated from the following file:

- Pawn.cs

## 5.14 ChessEngine.Queen Class Reference

Implementation of a Chess [Queen](#).

Inheritance diagram for ChessEngine.Queen:



### Public Member Functions

- **Queen** ([ChessTable](#) pTable, [ChessColor](#) pColor, [TileIndex](#) pTileIndex)
- **Queen** ([ChessTable](#) pTable, [ChessPiece](#) pOther)
- override List< [ChessTableTile](#) > [GenerateMovesList](#) ()  
*Returns a list with the ChessTableTiles of all possible moves for this piece.*
- override List< [AttackInfo](#) > [GenerateAttacksList](#) ([ChessTableTile](#) pSpoofofOccupiedTile, [ChessTableTile](#) pSpoofofUnoccupiedTile)  
*Returns a list with the AttackInfos of all possible attacks for this piece.*
- override string [GetFENIdentifier](#) ()  
*Returns the FEN identifier for the chess piece.*
- override [ChessPieceType](#) [GetChessPieceType](#) ()  
*Returns the ChessPieceType for this ChessPiece.*



## Additional Inherited Members

### 5.14.1 Detailed Description

Implementation of a Chess [Queen](#).

Author: Mathew Aloisio

### 5.14.2 Member Function Documentation

#### 5.14.2.1 GenerateAttacksList()

```
override List< AttackInfo > ChessEngine.Queen.GenerateAttacksList (
    ChessTableTile pSpoofofOccupiedTile,
    ChessTableTile pSpoofofUnoccupiedTile ) [virtual]
```

Returns a list with the AttackInfos of all possible attacks for this piece.

##### Returns

A list of AttackInfos for all possible attacks of this piece.

Implements [ChessEngine.ChessPiece](#).

#### 5.14.2.2 GenerateMovesList()

```
override List< ChessTableTile > ChessEngine.Queen.GenerateMovesList ( ) [virtual]
```

Returns a list with the ChessTableTiles of all possible moves for this piece.

##### Returns

A list of ChessTableTiles for all possible moves of this piece.

Implements [ChessEngine.ChessPiece](#).

#### 5.14.2.3 GetChessPieceType()

```
override ChessPieceType ChessEngine.Queen.GetChessPieceType ( ) [virtual]
```

Returns the ChessPieceType for this [ChessPiece](#).

##### Returns

the ChessPieceType for this [ChessPiece](#).

Implements [ChessEngine.ChessPiece](#).

#### 5.14.2.4 GetFENIdentifier()

```
override string ChessEngine.Queen.GetFENIdentifier ( ) [virtual]
```

Returns the FEN identifier for the chess piece.

White Piece | Black Piece | Chess Piece P | p | [Pawn](#) N | n | [Knight](#) B | b | [Bishop](#) R | r | [Rook](#) Q | q | [Queen](#) K | k | [King](#)

##### Returns

a string representing the FEN identifier for the chess piece.

Implements [ChessEngine.ChessPiece](#).

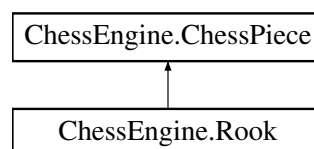
The documentation for this class was generated from the following file:

- [Queen.cs](#)

## 5.15 ChessEngine.Rook Class Reference

Implementation of a Chess [Rook](#).

Inheritance diagram for ChessEngine.Rook:



### Public Member Functions

- **Rook** ([ChessTable](#) pTable, [ChessColor](#) pColor, [TileIndex](#) pTileIndex)
- **Rook** ([ChessTable](#) pTable, [ChessPiece](#) pOther)
- override List< [ChessTableTile](#) > [GenerateMovesList](#) ()  
*Returns a list with the ChessTableTiles of all possible moves for this piece.*
- override List< [AttackInfo](#) > [GenerateAttacksList](#) ([ChessTableTile](#) pSpoofoccupiedTile, [ChessTableTile](#) pSpoofoUnoccupiedTile)  
*Returns a list with the AttackInfos of all possible attacks for this piece.*
- override string [GetFENIdentifier](#) ()  
*Returns the FEN identifier for the chess piece.*
- override [ChessPieceType](#) [GetChessPieceType](#) ()  
*Returns the ChessPieceType for this ChessPiece.*

## Static Public Attributes

- static readonly [TileIndex](#) **ROOK\_SPAWN\_KINGSIDE\_TILEINDEX\_WHITE** = new [TileIndex](#)(7, 0)  
The [TileIndex](#) the white kingside [Rook](#) spawns in when a new game is started.
- static readonly [TileIndex](#) **ROOK\_SPAWN\_QUEENSIDE\_TILEINDEX\_WHITE** = new [TileIndex](#)(0, 0)  
The [TileIndex](#) the white queenside [Rook](#) spawns in when a new game is started.
- static readonly [TileIndex](#) **ROOK\_SPAWN\_KINGSIDE\_TILEINDEX\_BLACK** = new [TileIndex](#)(7, 7)  
The [TileIndex](#) the black kingside [Rook](#) spawns in when a new game is started.
- static readonly [TileIndex](#) **ROOK\_SPAWN\_QUEENSIDE\_TILEINDEX\_BLACK** = new [TileIndex](#)(0, 7)  
The [TileIndex](#) the black queenside [Rook](#) spawns in when a new game is started.

## Properties

- bool **IsKingSide** [get, set]  
Returns true if this is the king side [Rook](#), otherwise false if it is the [Queen](#) side [Rook](#).

## Events

- Action< [ChessPiece](#), [TileIndex](#), [TileIndex](#) > **Castled**  
An event that is invoked whenever the rook castles with a king.

## Additional Inherited Members

### 5.15.1 Detailed Description

Implementation of a Chess [Rook](#).

Author: Mathew Aloisio

### 5.15.2 Member Function Documentation

#### 5.15.2.1 GenerateAttacksList()

```
override List< AttackInfo > ChessEngine.Rook.GenerateAttacksList (
    ChessTableTile pSpoofOccupiedTile,
    ChessTableTile pSpoofUnoccupiedTile ) [virtual]
```

Returns a list with the AttackInfos of all possible attacks for this piece.

#### Returns

A list of AttackInfos for all possible attacks of this piece.

Implements [ChessEngine.ChessPiece](#).

### 5.15.2.2 GenerateMovesList()

```
override List< ChessTableTile > ChessEngine.Rook.GenerateMovesList ( ) [virtual]
```

Returns a list with the ChessTableTiles of all possible moves for this piece.

#### Returns

A list of ChessTableTiles for all possible moves of this piece.

Implements [ChessEngine.ChessPiece](#).

### 5.15.2.3 GetChessPieceType()

```
override ChessPieceType ChessEngine.Rook.GetChessPieceType ( ) [virtual]
```

Returns the ChessPieceType for this [ChessPiece](#).

#### Returns

the ChessPieceType for this [ChessPiece](#).

Implements [ChessEngine.ChessPiece](#).

### 5.15.2.4 GetFENIdentifier()

```
override string ChessEngine.Rook.GetFENIdentifier ( ) [virtual]
```

Returns the FEN identifier for the chess piece.

White Piece | Black Piece | Chess Piece P | p | [Pawn](#) N | n | [Knight](#) B | b | [Bishop](#) R | r | [Rook](#) Q | q | [Queen](#) K | k | [King](#)

#### Returns

a string representing the FEN identifier for the chess piece.

Implements [ChessEngine.ChessPiece](#).

## 5.15.3 Event Documentation

### 5.15.3.1 Castled

Action<ChessPiece, TileIndex, TileIndex> ChessEngine.Rook.Castled

An event that is invoked whenever the rook castles with a king.

Arg0: ChessPiece - A reference to the King that castled with this Rook. Arg1: TileIndex - The original tile index for the Rook before castling. Arg2: TileIndex - The new tile index for the Rook after castling.

The documentation for this class was generated from the following file:

- Rook.cs

## 5.16 ChessEngine.RookReferences Class Reference

A class containing references to Rook pieces based on what color and side they are on.

### Public Attributes

- **Rook whiteKingSideRook**  
*The Rook on the King side of the white team.*
- **Rook whiteQueenSideRook**  
*The Rook on the Queen side of the white team.*
- **Rook blackKingSideRook**  
*The Rook on the King side of the black team.*
- **Rook blackQueenSideRook**  
*The Rook on the Queen side of the black team.*

### 5.16.1 Detailed Description

A class containing references to Rook pieces based on what color and side they are on.

Author: Mathew Aloisio

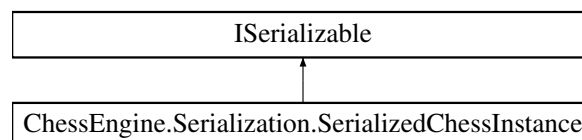
The documentation for this class was generated from the following file:

- RookReferences.cs

## 5.17 ChessEngine.Serialization.SerializedChessInstance Class Reference

A serializable class that provides a complete representation of a chess Instance.

Inheritance diagram for ChessEngine.Serialization.SerializedChessInstance:



## Public Member Functions

- **SerializedChessInstance** ()  
*Mandatory argument-less constructor.*
- **SerializedChessInstance** ([Instance](#) pInstance)  
*Constructs a serializable representation of the chess instance, pInstance.*

## Properties

- **ChessColor Turn** [get, set]  
*The turn the serialized chess instance is on.*
- **SerializedChessTable SerializedTable** [get, set]  
*The serialized chess table representation for this serialized chess instance.*
- **int FullMoveCounter** [get, set]  
*The number of full moves this game.*
- **int HalfMovesClock** [get, set]  
*The number of half moves this game.*
- **bool IsEnPassantEligible** [get, set]  
*Returns true if there is a valid en-passant eligible [Pawn](#) in the instance, otherwise false.*
- **TileIndex EnPassantEligibleTile** [get, set]  
*Only valid when 'IsEnPassantEligible' is true. The [TileIndex](#) of the pawn that is eligible to be taken en-passant.*
- **bool IsEnPassantSet** [get, set]  
*Tracks whether or not 'en passant' eligibility was set in a given turn for the chess instance. (NOTE: This is the serialized version of a private boolean inside of [Instance](#).)*

### 5.17.1 Detailed Description

A serializable class that provides a complete representation of a chess [Instance](#).

Author: Mathew Aloisio

### 5.17.2 Constructor & Destructor Documentation

#### 5.17.2.1 SerializedChessInstance()

```
ChessEngine.Serialization.SerializedChessInstance.SerializedChessInstance (
    Instance pInstance )
```

Constructs a serializable representation of the chess instance, pInstance.

#### Parameters

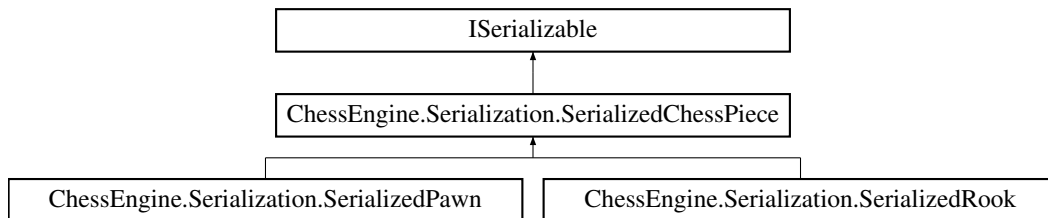
<i>pInstance</i>	
------------------	--

The documentation for this class was generated from the following file:

- SerializedChessInstance.cs

## 5.18 ChessEngine.Serialization.SerializedChessPiece Class Reference

Inheritance diagram for ChessEngine.Serialization.SerializedChessPiece:



### Public Member Functions

- **SerializedChessPiece** ()  
*Instantiates a blank SerializableChessPiece.*
- **SerializedChessPiece** ([ChessPiece](#) pPiece)  
*Instantiates a SerializableChessPiece from a [ChessPiece](#).*

### Properties

- [ChessColor](#) **Color** [get, set]  
*Returns the ChessColor team/color that this SerializableChessPiece belongs to.*
- bool **IsCaptured** [get, set]  
*Returns true if this serializable piece has been captured, otherwise false.*
- int **MoveCount** [get, set]  
*The number of moves this serializable Chess piece has made.*
- [ChessPieceType](#) **PieceType** [get, set]  
*The ChessPieceType of the serialized piece.*
- [TileIndex](#) **TileIndex** [get, set]  
*Represents the table tile index this serializable chess piece is on.*

### 5.18.1 Constructor & Destructor Documentation

#### 5.18.1.1 SerializedChessPiece()

```
ChessEngine.Serialization.SerializedChessPiece.SerializedChessPiece (
    ChessPiece pPiece )
```

Instantiates a SerializableChessPiece from a [ChessPiece](#).

## Parameters

<i>pPiece</i>	
---------------	--

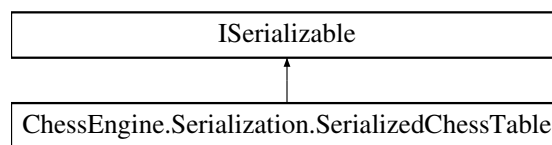
The documentation for this class was generated from the following file:

- SerializedChessPiece.cs

## 5.19 ChessEngine.Serialization.SerializedChessTable Class Reference

A serializable class that provides a complete representation of a Chess board.

Inheritance diagram for ChessEngine.Serialization.SerializedChessTable:



### Public Member Functions

- **SerializedChessTable** ()  
*Mandatory argument-less constructor.*
- **SerializedChessTable** ([ChessTable](#) pTable)  
*Constructs a serialized representation of the [ChessTable](#), pTable.*

### Properties

- int **SerializedPieceCount** [get, set]  
*The size of the 'SerializedPieces' list.*
- List< [SerializedChessPiece](#) > **SerializedPieces** [get, set]  
*An array of all serializable Chess piece on the serializable Chess table.*

#### 5.19.1 Detailed Description

A serializable class that provides a complete representation of a Chess board.

Author: Mathew Aloisio

#### 5.19.2 Constructor & Destructor Documentation

##### 5.19.2.1 SerializedChessTable()

```
ChessEngine.Serialization.SerializedChessTable.SerializedChessTable (
    ChessTable pTable )
```

Constructs a serialized representation of the [ChessTable](#), pTable.



## Parameters

<i>pTable</i>	
---------------	--

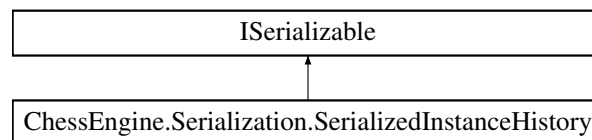
The documentation for this class was generated from the following file:

- SerializedChessTable.cs

## 5.20 ChessEngine.Serialization.SerializedInstanceHistory Class Reference

A serializable class that provides a complete representation of the moves and undone moves stacks for an InstanceHistory object.

Inheritance diagram for ChessEngine.Serialization.SerializedInstanceHistory:



### Public Member Functions

- **SerializedInstanceHistory** ()  
*Mandatory argument-less constructor.*
- **SerializedInstanceHistory** ([InstanceHistory](#) pHistory)  
*Constructs a serializable representation of the instance history object, pHistory.*

### Properties

- bool **TrackHistory** [get, set]  
*Should this class instance track history? If true history is tracked, otherwise not tracked.*
- int **MoveHistoryCount** [get, set]  
*The number of entries in the move history stack.*
- [HistoryEntry](#)[] **MoveHistory** [get, set]  
*A array of all HistoryEntrys from the 'moves' stack.*
- int **UndoneMovesCount** [get, set]  
*The number of entries in the undone move history stack.*
- [HistoryEntry](#)[] **UndoneMoves** [get, set]  
*A array of all HistoryEntrys from the 'undone moves' stack.*

#### 5.20.1 Detailed Description

A serializable class that provides a complete representation of the moves and undone moves stacks for an InstanceHistory object.

Author: Mathew Aloisio

## 5.20.2 Constructor & Destructor Documentation

### 5.20.2.1 SerializedInstanceHistory()

```
ChessEngine.Serialization.SerializedInstanceHistory.SerializedInstanceHistory (
    InstanceHistory pHistory )
```

Constructs a serializable representation of the instance history object, pHistory.

#### Parameters

<i>pHistory</i>	
-----------------	--

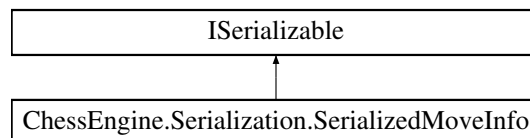
The documentation for this class was generated from the following file:

- SerializedInstanceHistory.cs

## 5.21 ChessEngine.Serialization.SerializedMoveInfo Class Reference

A serializable class that provides a complete representation of a [MoveInfo](#) instance.

Inheritance diagram for ChessEngine.Serialization.SerializedMoveInfo:



### Public Member Functions

- **SerializedMoveInfo ()**  
*Mandatory argument-less constructor.*
- **SerializedMoveInfo (MoveInfo pMoveInfo)**  
*Constructs a serialized representation of the [MoveInfo](#), pMoveInfo.*
- **MoveInfo ToMoveInfo (ChessTable pTable)**  
*Converts the [SerializedMoveInfo](#) into [MoveInfo](#) for the chess table, pTable and returns it. NOTE: This is only able to generate moves before they are performed, otherwise the 'piece' and 'capturedPiece' references may not be possible to find.*

### Properties

- **bool IsCapture** [get, set]  
*A boolean that tracks whether or not the move info represents an attack. True if attack, otherwise false.*
- **TileIndex CapturedTileIndex** [get, set]  
*The [TileIndex](#) of the tile being captured. This is only valid when 'IsCapture' is true.*
- **TileIndex ToTileIndex** [get, set]  
*The [TileIndex](#) being moved to.*
- **TileIndex FromTileIndex** [get, set]  
*The [TileIndex](#) being moved from.*

### 5.21.1 Detailed Description

A serializable class that provides a complete representation of a [MoveInfo](#) instance.

Author: Mathew Aloisio

### 5.21.2 Constructor & Destructor Documentation

#### 5.21.2.1 SerializedMoveInfo()

```
ChessEngine.Serialization.SerializedMoveInfo.SerializedMoveInfo (
    MoveInfo pMoveInfo )
```

Constructs a serialized representation of the [MoveInfo](#), pMoveInfo.

Parameters

<i>pMoveInfo</i>	
------------------	--

### 5.21.3 Member Function Documentation

#### 5.21.3.1 ToMoveInfo()

```
MoveInfo ChessEngine.Serialization.SerializedMoveInfo.ToMoveInfo (
    ChessTable pTable )
```

Converts the [SerializedMoveInfo](#) into [MoveInfo](#) for the chess table, pTable and returns it. NOTE: This is only able to generate moves before they are performed, otherwise the 'piece' and 'capturedPiece' references may not be possible to find.

Parameters

<i>pTable</i>	
---------------	--

Returns

a [MoveInfo](#) object that describes the serialized move on the chess table, pTable.

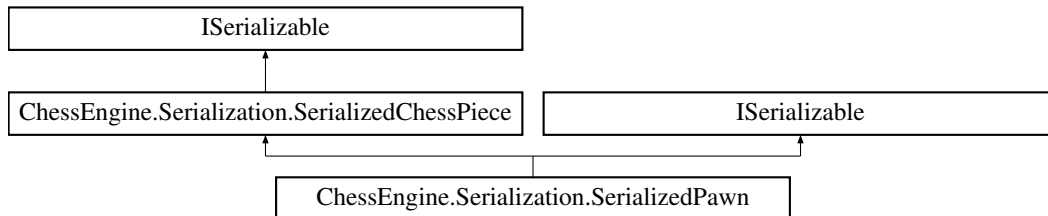
The documentation for this class was generated from the following file:

- SerializedMoveInfo.cs

## 5.22 ChessEngine.Serialization.SerializedPawn Class Reference

Derived from [SerializedPawn](#) this class represents a serialized [Pawn](#) piece.

Inheritance diagram for ChessEngine.Serialization.SerializedPawn:



### Public Member Functions

- **SerializedPawn ()**  
*Instantaites a blank SerializablePawn.*
- **SerializedPawn (Pawn pPawn)**  
*Instantiates a SerializablePawn from a [Pawn](#).*

### Properties

- **bool IsEnPassanting** [get, set]  
*Tracks whether or not the serialized pawn's 'enPassant' field is valid or null. (true of valid, otherwise null.)*
- **TileIndex EnPassantingTile** [get, set]  
*Tracks the tile of the 'enPassanting' pawn reference for a serialized pawn.*

#### 5.22.1 Detailed Description

Derived from [SerializedPawn](#) this class represents a serialized [Pawn](#) piece.

Author: Mathew Aloisio

#### 5.22.2 Constructor & Destructor Documentation

##### 5.22.2.1 SerializedPawn()

```
ChessEngine.Serialization.SerializedPawn.SerializedPawn (
    Pawn pPawn )
```

Instantiates a SerializablePawn from a [Pawn](#).

## Parameters

<code>pPawn</code>	
--------------------	--

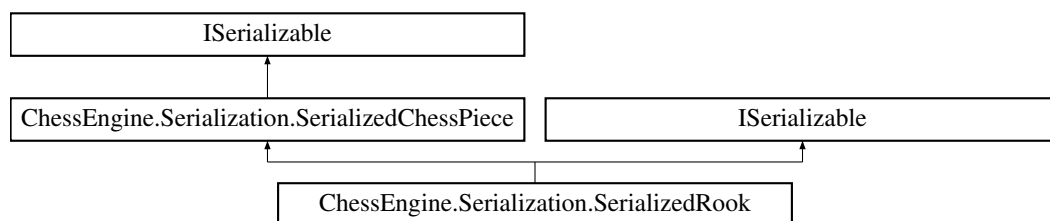
The documentation for this class was generated from the following file:

- SerializedPawn.cs

## 5.23 ChessEngine.Serialization.SerializedRook Class Reference

Derived from [SerializedRook](#) this class represents a serialized [Rook](#) piece.

Inheritance diagram for ChessEngine.Serialization.SerializedRook:



### Public Member Functions

- **SerializedRook ()**  
*Instantiates a blank SerializableRook.*
- **SerializedRook (Rook pRook)**  
*Instantiates a SerializableRook from a [Rook](#).*

### Properties

- **bool IsKingSide** [get, set]  
*Tracks whether or not the serialized rook is the kingside rook or not. (true for kingside, otherwise false.)*

#### 5.23.1 Detailed Description

Derived from [SerializedRook](#) this class represents a serialized [Rook](#) piece.

Author: Mathew Aloisio

#### 5.23.2 Constructor & Destructor Documentation

##### 5.23.2.1 SerializedRook()

```
ChessEngine.Serialization.SerializedRook.SerializedRook (
    Rook pRook )
```

Instantiates a SerializableRook from a [Rook](#).

## Parameters

<i>pRook</i>	
--------------	--

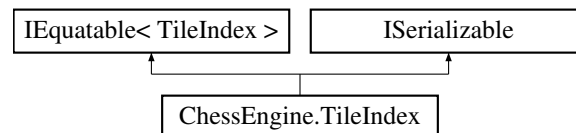
The documentation for this class was generated from the following file:

- SerializedRook.cs

## 5.24 ChessEngine.TileIndex Struct Reference

A [TileIndex](#) represents the index of a [ChessTableTile](#) along an 8x8 grid. Note that  $x == 0, y == 0$ , represents the lower-left origin of the chess table.

Inheritance diagram for ChessEngine.TileIndex:



### Public Member Functions

- string [GetTileID](#) ()  
*Gets the 'column' 'rank' identifier of the tile. (e.g: a1, e4, etc)*
- TileIndex** (int pX, int pY)
- TileIndex** (SerializationInfo pInfo, StreamingContext pContext)
- void ISerializable. **GetObjectData** (SerializationInfo pInfo, StreamingContext pContext)
- bool [Equals](#) ([TileIndex](#) pTileIndex)  
*Returns true if the TileIndexes match in both the x and y field, otherwise false.*
- override bool **Equals** (object pObject)
- override int **GetHashCode** ()
- override string **ToString** ()

### Static Public Member Functions

- static bool **operator==** ([TileIndex](#) pLHS, [TileIndex](#) pRHS)
- static bool **operator!=** ([TileIndex](#) pLHS, [TileIndex](#) pRHS)

### Public Attributes

- int **x**
- int **y**

### 5.24.1 Detailed Description

A [TileIndex](#) represents the index of a [ChessTableTile](#) along an 8x8 grid. Note that x == 0, y == 0, represents the lower-left origin of the chess table.

Author: Intuitive Gaming SOLutions

### 5.24.2 Member Function Documentation

#### 5.24.2.1 Equals()

```
bool ChessEngine.TileIndex.Equals (
    TileIndex pTileIndex )
```

Returns true if the TileIndexes match in both the x and y field, otherwise false.

##### Parameters

<i>pTileIndex</i>	The tile index to compare against.
-------------------	------------------------------------

##### Returns

true if the TileIndexes match in both the x and y field, otherwise false.

#### 5.24.2.2 GetTileID()

```
string ChessEngine.TileIndex.GetTileID ( )
```

Gets the 'column' 'rank' identifier of the tile. (e.g: a1, e4, etc)

##### Returns

a string representing the tile in classic 'column rank' idenfication.

The documentation for this struct was generated from the following file:

- TileIndex.cs

## 5.25 ChessEngine.TimeSystem.TimeManager Class Reference

A class that provides easy-to-use time related methods.

## Public Member Functions

- void **Pause** ()  
*Pauses time.*
- void **Unpause** ()  
*Unpauses time.*
- void **SetElapsedTime** (float pElapsedTime)  
*A public method that allows the TimeManagers elapsed time to be directly overridden.*

## Properties

- bool **IsPaused** [get, set]  
*Controls whether or not the [TimeManager](#) is paused.*
- float **ElapsedTime** [get]  
*Returns the number of seconds that have elapsed since the [TimeManager](#) was constructed or unpaused. (Will return the same value at any time while paused.)*

### 5.25.1 Detailed Description

A class that provides easy-to-use time related methods.

Author: Mathew Aloisio

### 5.25.2 Member Function Documentation

#### 5.25.2.1 SetElapsedTime()

```
void ChessEngine.TimeSystem.TimeManager.SetElapsedTime (  
    float pElapsedTime )
```

A public method that allows the TimeManagers elapsed time to be directly overridden.

#### Parameters

<i>pElapsedTime</i>	
---------------------	--

The documentation for this class was generated from the following file:

- TimeManager.cs



# Index

- ActionRef< T >
  - ChessEngine.Delegates, 8
- Capture
  - ChessEngine.ChessPiece, 19
- Castled
  - ChessEngine.ChessTable, 35
  - ChessEngine.Instance, 47
  - ChessEngine.King, 57
  - ChessEngine.Rook, 68
- ChessColor
  - ChessEngine, 8
- ChessEngine, 7
  - ChessColor, 8
- ChessEngine.AttackInfo, 13
- ChessEngine.Bishop, 13
  - GenerateAttacksList, 14
  - GenerateMovesList, 14
  - GetChessPieceType, 14
  - GetFENIdentifier, 15
- ChessEngine.ChessPiece, 15
  - Capture, 19
  - ChessPiece, 17, 19
  - GenerateAttacksList, 19
  - GenerateMovesList, 20
  - GetAttacks, 20
  - GetChessPieceType, 20
  - GetFENIdentifier, 21
  - GetMoves, 21
  - GetValidAttacks, 21
  - GetValidMoves, 22
  - Move, 22
  - OnPostMoved, 22
  - OnPreMoved, 23
  - OnTileIndexChanged, 23
- ChessEngine.ChessTable, 23
  - Castled, 35
  - ChessPieceMoved, 35
  - ChessTable, 26
  - CreateBishop, 27
  - CreatedSerializedChessPiece, 35
  - CreateKing, 27
  - CreateKnight, 27
  - CreatePawn, 28
  - CreatePieceByType, 28
  - CreateQueen, 29
  - CreateRook, 29
  - CreateSerializedPiece, 29
  - DestroyPiece, 30
  - DestroyPieceByIndex, 30
  - DetermineRookSides, 30
  - GenerateEPDString, 31
  - GetKing, 31
  - GetPieceByIndex, 32
  - GetRookReferences, 32
  - GetTile, 32, 33
  - GetTileByID, 33
  - IsInCheck, 33
  - LoadedSerializedTable, 35
  - PreChessPieceMoved, 35
  - SetState, 34
  - SetStateToEPD, 34
- ChessEngine.ChessTableTile, 36
  - ChessTableTile, 37
  - GetPiece, 37
  - IsTileAttackable, 38
  - IsTileThreatened, 38
  - MovePieceToTile, 39
- ChessEngine.Delegates, 8
  - ActionRef< T >, 8
  - ValueActionRef< VAL\_T, REF\_T >, 9
  - ValueActionRef< VAL\_T, VAL\_T2, REF\_T >, 9
  - ValueActionRef< VAL\_T, VAL\_T2, VAL\_T3, REF\_T >, 10
  - ValueActionRef< VAL\_T, VAL\_T2, VAL\_T3, VAL\_T4, REF\_T >, 10
- ChessEngine.Instance, 40
  - Castled, 47
  - ChessPieceMoved, 47
  - EndTurn, 43
  - GenerateFENCastleString, 44
  - GenerateFENCastleStringForRook, 44
  - GenerateFENEnPassantString, 44
  - GenerateFENString, 44
  - Instance, 42, 43
  - IsGameOver, 45
  - IsGameOverCallback, 47
  - LoadedSerializedInstance, 48
  - OnGameOver, 45
  - OnTurnEnded, 45
  - OnTurnStarted, 46
  - PreChessPieceMoved, 48
  - QueeningPawnCallback, 48
  - QueenPawn, 46
  - SetState, 46
  - SetStateToFEN, 47
- ChessEngine.King, 54
  - Castled, 57
  - GenerateAttacksList, 55

- GenerateMovesList, [55](#)
- GetChessPieceType, [55](#)
- GetFENIdentifier, [55](#)
- OnCastled, [56](#)
- OnTileIndexChanged, [56](#)
- ChessEngine.Knight, [57](#)
  - GenerateAttacksList, [58](#)
  - GenerateMovesList, [58](#)
  - GetChessPieceType, [58](#)
  - GetFENIdentifier, [58](#)
- ChessEngine.MoveData, [59](#)
  - MoveData, [59](#)
- ChessEngine.MoveInfo, [60](#)
- ChessEngine.Pawn, [60](#)
  - GenerateAttacksList, [61](#)
  - GenerateMovesList, [62](#)
  - GetChessPieceType, [62](#)
  - GetFENIdentifier, [62](#)
  - IsOnStartingRow, [63](#)
  - OnPostMoved, [63](#)
  - OnPreMoved, [63](#)
  - OnTileIndexChanged, [64](#)
- ChessEngine.Queen, [64](#)
  - GenerateAttacksList, [65](#)
  - GenerateMovesList, [65](#)
  - GetChessPieceType, [65](#)
  - GetFENIdentifier, [65](#)
- ChessEngine.Rook, [66](#)
  - Castled, [68](#)
  - GenerateAttacksList, [67](#)
  - GenerateMovesList, [67](#)
  - GetChessPieceType, [68](#)
  - GetFENIdentifier, [68](#)
- ChessEngine.RookReferences, [69](#)
- ChessEngine.Serialization, [11](#)
- ChessEngine.Serialization.SerializedChessInstance, [69](#)
  - SerializedChessInstance, [70](#)
- ChessEngine.Serialization.SerializedChessPiece, [71](#)
  - SerializedChessPiece, [71](#)
- ChessEngine.Serialization.SerializedChessTable, [72](#)
  - SerializedChessTable, [72](#)
- ChessEngine.Serialization.SerializedInstanceHistory, [73](#)
  - SerializedInstanceHistory, [74](#)
- ChessEngine.Serialization.SerializedMoveInfo, [74](#)
  - SerializedMoveInfo, [75](#)
  - ToMoveInfo, [75](#)
- ChessEngine.Serialization.SerializedPawn, [76](#)
  - SerializedPawn, [76](#)
- ChessEngine.Serialization.SerializedRook, [77](#)
  - SerializedRook, [77](#)
- ChessEngine.TileIndex, [78](#)
  - Equals, [79](#)
  - GetTileID, [79](#)
- ChessEngine.TimeSystem, [11](#)
- ChessEngine.TimeSystem.TimeManager, [79](#)
  - SetElapsedTime, [80](#)
- ChessEngine.Undo, [12](#)
- ChessEngine.Undo.HistoryEntry, [39](#)
- ChessEngine.Undo.InstanceHistory, [49](#)
  - InstanceHistory, [51](#)
  - OnGameOver, [51](#)
  - OnInstanceSet, [51](#)
  - OnInstanceUnset, [51](#)
  - OnPreChessPieceMoved, [52](#)
  - OnTurnEnded, [52](#)
  - PeekMove, [52](#)
  - PeekUndoneMove, [53](#)
  - PushMoveHistory, [53](#)
  - RedoMove, [53](#)
  - UndoMove, [53](#)
- ChessEngine.Utility, [12](#)
- ChessPiece
  - ChessEngine.ChessPiece, [17](#), [19](#)
- ChessPieceMoved
  - ChessEngine.ChessTable, [35](#)
  - ChessEngine.Instance, [47](#)
- ChessTable
  - ChessEngine.ChessTable, [26](#)
- ChessTableTile
  - ChessEngine.ChessTableTile, [37](#)
- CreateBishop
  - ChessEngine.ChessTable, [27](#)
- CreatedSerializedChessPiece
  - ChessEngine.ChessTable, [35](#)
- CreateKing
  - ChessEngine.ChessTable, [27](#)
- CreateKnight
  - ChessEngine.ChessTable, [27](#)
- CreatePawn
  - ChessEngine.ChessTable, [28](#)
- CreatePieceByType
  - ChessEngine.ChessTable, [28](#)
- CreateQueen
  - ChessEngine.ChessTable, [29](#)
- CreateRook
  - ChessEngine.ChessTable, [29](#)
- CreateSerializedPiece
  - ChessEngine.ChessTable, [29](#)
- DestroyPiece
  - ChessEngine.ChessTable, [30](#)
- DestroyPieceByIndex
  - ChessEngine.ChessTable, [30](#)
- DetermineRookSides
  - ChessEngine.ChessTable, [30](#)
- EndTurn
  - ChessEngine.Instance, [43](#)
- Equals
  - ChessEngine.TileIndex, [79](#)
- GenerateAttacksList
  - ChessEngine.Bishop, [14](#)
  - ChessEngine.ChessPiece, [19](#)
  - ChessEngine.King, [55](#)
  - ChessEngine.Knight, [58](#)

- ChessEngine.Pawn, [61](#)
- ChessEngine.Queen, [65](#)
- ChessEngine.Rook, [67](#)
- GenerateEPDString
  - ChessEngine.ChessTable, [31](#)
- GenerateFENCastleString
  - ChessEngine.Instance, [44](#)
- GenerateFENCastleStringForRook
  - ChessEngine.Instance, [44](#)
- GenerateFENEnPassantString
  - ChessEngine.Instance, [44](#)
- GenerateFENString
  - ChessEngine.Instance, [44](#)
- GenerateMovesList
  - ChessEngine.Bishop, [14](#)
  - ChessEngine.ChessPiece, [20](#)
  - ChessEngine.King, [55](#)
  - ChessEngine.Knight, [58](#)
  - ChessEngine.Pawn, [62](#)
  - ChessEngine.Queen, [65](#)
  - ChessEngine.Rook, [67](#)
- GetAttacks
  - ChessEngine.ChessPiece, [20](#)
- GetChessPieceType
  - ChessEngine.Bishop, [14](#)
  - ChessEngine.ChessPiece, [20](#)
  - ChessEngine.King, [55](#)
  - ChessEngine.Knight, [58](#)
  - ChessEngine.Pawn, [62](#)
  - ChessEngine.Queen, [65](#)
  - ChessEngine.Rook, [68](#)
- GetFENIdentifier
  - ChessEngine.Bishop, [15](#)
  - ChessEngine.ChessPiece, [21](#)
  - ChessEngine.King, [55](#)
  - ChessEngine.Knight, [58](#)
  - ChessEngine.Pawn, [62](#)
  - ChessEngine.Queen, [65](#)
  - ChessEngine.Rook, [68](#)
- GetKing
  - ChessEngine.ChessTable, [31](#)
- GetMoves
  - ChessEngine.ChessPiece, [21](#)
- GetPiece
  - ChessEngine.ChessTableTile, [37](#)
- GetPieceByIndex
  - ChessEngine.ChessTable, [32](#)
- GetRookReferences
  - ChessEngine.ChessTable, [32](#)
- GetTile
  - ChessEngine.ChessTable, [32](#), [33](#)
- GetTileById
  - ChessEngine.ChessTable, [33](#)
- GetTileId
  - ChessEngine.TileIndex, [79](#)
- GetValidAttacks
  - ChessEngine.ChessPiece, [21](#)
- GetValidMoves
  - ChessEngine.ChessPiece, [22](#)
- Instance
  - ChessEngine.Instance, [42](#), [43](#)
- InstanceHistory
  - ChessEngine.Undo.InstanceHistory, [51](#)
- IsGameOver
  - ChessEngine.Instance, [45](#)
- IsGameOverCallback
  - ChessEngine.Instance, [47](#)
- IsInCheck
  - ChessEngine.ChessTable, [33](#)
- IsOnStartingRow
  - ChessEngine.Pawn, [63](#)
- IsTileAttackable
  - ChessEngine.ChessTableTile, [38](#)
- IsTileThreatened
  - ChessEngine.ChessTableTile, [38](#)
- LoadedSerializedInstance
  - ChessEngine.Instance, [48](#)
- LoadedSerializedTable
  - ChessEngine.ChessTable, [35](#)
- Move
  - ChessEngine.ChessPiece, [22](#)
- MoveData
  - ChessEngine.MoveData, [59](#)
- MovePieceToTile
  - ChessEngine.ChessTableTile, [39](#)
- OnCastled
  - ChessEngine.King, [56](#)
- OnGameOver
  - ChessEngine.Instance, [45](#)
  - ChessEngine.Undo.InstanceHistory, [51](#)
- OnInstanceSet
  - ChessEngine.Undo.InstanceHistory, [51](#)
- OnInstanceUnset
  - ChessEngine.Undo.InstanceHistory, [51](#)
- OnPostMoved
  - ChessEngine.ChessPiece, [22](#)
  - ChessEngine.Pawn, [63](#)
- OnPreChessPieceMoved
  - ChessEngine.Undo.InstanceHistory, [52](#)
- OnPreMoved
  - ChessEngine.ChessPiece, [23](#)
  - ChessEngine.Pawn, [63](#)
- OnTileIndexChanged
  - ChessEngine.ChessPiece, [23](#)
  - ChessEngine.King, [56](#)
  - ChessEngine.Pawn, [64](#)
- OnTurnEnded
  - ChessEngine.Instance, [45](#)
  - ChessEngine.Undo.InstanceHistory, [52](#)
- OnTurnStarted
  - ChessEngine.Instance, [46](#)
- PeekMove

- ChessEngine.Undo.InstanceHistory, [52](#)
- PeekUndoneMove
  - ChessEngine.Undo.InstanceHistory, [53](#)
- PreChessPieceMoved
  - ChessEngine.ChessTable, [35](#)
  - ChessEngine.Instance, [48](#)
- PushMoveHistory
  - ChessEngine.Undo.InstanceHistory, [53](#)
- QueeningPawnCallback
  - ChessEngine.Instance, [48](#)
- QueenPawn
  - ChessEngine.Instance, [46](#)
- RedoMove
  - ChessEngine.Undo.InstanceHistory, [53](#)
- SerializedChessInstance
  - ChessEngine.Serialization.SerializedChessInstance, [70](#)
- SerializedChessPiece
  - ChessEngine.Serialization.SerializedChessPiece, [71](#)
- SerializedChessTable
  - ChessEngine.Serialization.SerializedChessTable, [72](#)
- SerializedInstanceHistory
  - ChessEngine.Serialization.SerializedInstanceHistory, [74](#)
- SerializedMoveInfo
  - ChessEngine.Serialization.SerializedMoveInfo, [75](#)
- SerializedPawn
  - ChessEngine.Serialization.SerializedPawn, [76](#)
- SerializedRook
  - ChessEngine.Serialization.SerializedRook, [77](#)
- SetElapsedTime
  - ChessEngine.TimeSystem.TimeManager, [80](#)
- SetState
  - ChessEngine.ChessTable, [34](#)
  - ChessEngine.Instance, [46](#)
- SetStateToEPD
  - ChessEngine.ChessTable, [34](#)
- SetStateToFEN
  - ChessEngine.Instance, [47](#)
- ToMoveInfo
  - ChessEngine.Serialization.SerializedMoveInfo, [75](#)
- UndoMove
  - ChessEngine.Undo.InstanceHistory, [53](#)
- ValueActionRef< VAL\_T, REF\_T >
  - ChessEngine.Delegates, [9](#)
- ValueActionRef< VAL\_T, VAL\_T2, REF\_T >
  - ChessEngine.Delegates, [9](#)
- ValueActionRef< VAL\_T, VAL\_T2, VAL\_T3, REF\_T >
  - ChessEngine.Delegates, [10](#)
- ValueActionRef< VAL\_T, VAL\_T2, VAL\_T3, VAL\_T4, REF\_T >
  - ChessEngine.Delegates, [10](#)