Omni Grid

Generated by Doxygen 1.9.8

Chapter 1

Todo List

Struct fmt::formatter< OGRID::PlayerNameAndPtr >
: Try to move this to ogrid_fmt.h at some point.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 OGRID Namespace Reference

Classes

- class BlackPiece
- class BlackPieceCheckers
- struct Button

  Button.
- struct Cell

  The Cell struct represents a single cell in the grid.
- class Checkers
- class CheckersStateCheck
- struct ConfigurationBuilder

  The ConfigurationBuilder interface.
- class ConnectFour
- class ConnectFourStateCheck
- struct GameConfiguration

  The GameConfiguration class. Used to represent a game configuration.
- class GameConfigurationBuilder

  The GameConfigurationBuilder class. Used to build a GameConfiguration object.
- class GameStateChecker

  The GameStateChecker class. Used to check the state of the game.
- class GameStateExtensions

  The GameStateExtensions class. Used to extend the GameStateChecker class.
- class Grid

  The Grid class represents a 2D grid of Cells.
- class IAttackRule

  The IGameState class. Used to check the state of the game.
- class IGame

  The IGame class. Used to represent a game.
- class IGameState

  The IGameState class. Used to check the state of the game.
- class IMoveRule

  The IMoveRule class. Used to check if the move is valid.
- class JumpNormalCheckersAttackRule

The PieceRules class. Used to represent the rules of a piece.
- class JumpSuperCheckersAttackRule

  The PieceRules class. Used to represent the rules of a piece.
- class NormalCheckersMoveRule

  The PieceRules class. Used to represent the rules of a piece.
- class OPiece

  TicTacToe O piece.
- class Piece

  The Piece class. Used to represent a piece.
- class Player

  The Player class. Used to represent a player.
- struct PlayerNameAndPtr

  Pair of player name and pointer.
- class RedPiece
- class SimplePlaceMoveRule

  The PieceRules class. Used to represent the rules of a piece.
- class SuperCheckersMoveRule

  The PieceRules class. Used to represent the rules of a piece.
- struct Text

  Text.
- class TicTacToe

  TicTacToe game logic.
- class TicTacToeStateCheck

  TicTacToe state check.
- class WhitePieceCheckers
- class XPiece

  TicTacToe X piece.

Enumerations

- enum GameState { NotStarted = 0 , InProgress = 1 , Paused = 2 , GameOver = 3 }

  The IGame class. Used to represent a game.
- enum GameOverType { None = 0 , Win = 1 , Draw = 2 }

  The GameOverType enum. Used to represent the type of game over.
- enum class Justify { NONE , CENTER_X , CENTER_Y , CENTER_BOTH }

  Justify the text.
- enum PlayerType { Human = 0 , AI = 1 }

  The type of the player.

Functions

- std::string PlayerNameAndPtrVecToString (const std::vector< PlayerNameAndPtr > &players)
- PlayerType PlayerTypeStringToEnum (const std::string &s)

  Converts a string to a PlayerType.
- std::string PlayerTypeEnumToString (PlayerType playerType)

  Converts a PlayerType to a string.
- std::string PlayerVecToString (const std::vector< OGRID::Player * > &players)
- std::string PlayerVecToString (const std::vector< Player * > &players)

  Converts a Vector of Players to a string.

6.1.1 Enumeration Type Documentation

6.1.1.1 GameOverType

```
enum OGRID::GameOverType
```

The GameOverType enum. Used to represent the type of game over.

It contains the type of game over: None, Win or Draw.

Date

2023-12-06

Enumerator

None	
Win	
Draw	

6.1.1.2 GameState

```
enum OGRID::GameState
```

The IGame class. Used to represent a game.

It contains the name of the game, the description of the game, the grid of the game, the maximum number of players and the players of the game.

Date

2023-12-06

Enumerator

NotStarted	
InProgress	
Paused	
GameOver	

6.1.1.3 Justify

```
enum class OGRID::Justify [strong]
```

Justify the text.

Date

2023-12-06

Note

This solution is not working as intended.

Enumerator

NONE	
CENTER_X	
CENTER_Y	
CENTER_BOTH	

6.1.1.4 PlayerType

```
enum OGRID::PlayerType
```

The type of the player.

The type of the player, either Human or AI. At the moment, the AI is not implemented.

Date

2023-12-06

Enumerator

Human	
AI	

6.1.2 Function Documentation

6.1.2.1 PlayerNameAndPtrVecToString()

```
std::string OGRID::PlayerNameAndPtrVecToString (
            const std::vector< PlayerNameAndPtr > & players )
```

6.1.2.2 PlayerTypeEnumToString()

```
std::string OGRID::PlayerTypeEnumToString (
            PlayerType playerType )
```

Converts a PlayerType to a string.

Converts a PlayerType to a string. If the PlayerType is not valid, it returns "Human".

Date

2023-12-06

Parameters

playerType	The PlayerType to convert.

Returns

The string corresponding to the PlayerType.

6.1.2.3 PlayerTypeStringToEnum()

```
PlayerType OGRID::PlayerTypeStringToEnum (
             const std::string & s )
```

Converts a string to a PlayerType.

Converts a string to a PlayerType. If the string is not a valid PlayerType, it returns PlayerType::Human.

Date

2023-12-06

Parameters

s	The string to convert.

Returns

The PlayerType corresponding to the string.

6.1.2.4 PlayerVecToString() [1/2]

```
std::string OGRID::PlayerVecToString (
             const std::vector< OGRID::Player * > & players )
```

6.1.2.5 PlayerVecToString() [2/2]

```
std::string OGRID::PlayerVecToString (
             const std::vector< Player * > & players )
```

Converts a Vector of Players to a string.

Date

2023-12-06

Parameters

player	The Vector of Player to convert.

Returns

The string corresponding to the Vector of Players.

6.2 Sandbox Namespace Reference

Classes

- class GameInitializer

  Game initializer.
- class GameWindow

  Game window.

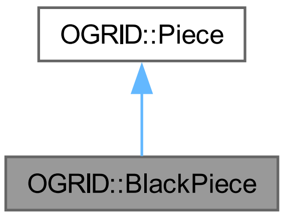
Chapter 7

Class Documentation

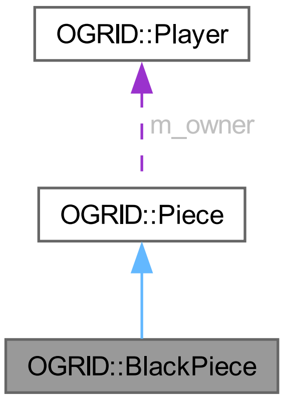
7.1 OGRID::BlackPiece Class Reference

```
#include <ConnectFourPieces.h>
```

Inheritance diagram for OGRID::BlackPiece:



Collaboration diagram for OGRID::BlackPiece:



Public Member Functions

- BlackPiece (Player ∗player)

Public Member Functions inherited from OGRID::Piece

- Piece (std::string rep, Player ∗player)

  Construct a new Piece object.
- ∼Piece ()

  Destroy the Piece object.
- void AddMoveRule (IMoveRule ∗rule)

  Add a move rule to the piece.
- void AddAttackRule (IAttackRule ∗rule)

  Add an attack rule to the piece.
- const std::string & GetRepresentation () const

  Get the representation of the piece.
- const Player ∗ GetOwner () const

  Get the owner of the piece.
- void SetOwner (Player ∗player)

  Set the owner of the piece.
- bool isValidMove (Grid ∗grid, int fromX, int fromY, int toX, int toY) const

  Check if the move is valid.
- bool isValidAttack (Grid ∗grid, int fromX, int fromY, int toX, int toY, bool &canContinue) const

  Check if the attack is valid.

Additional Inherited Members

Protected Attributes inherited from OGRID::Piece

- std::string m_representation

  The representation of the piece.
- std::vector< IMoveRule ∗ > m_moveRules

  The move rules of the piece.
- std::vector< IAttackRule ∗ > m_attackRules

  The attack rules of the piece.
- Player ∗ m_owner

  The owner of the piece.

7.1.1 Constructor & Destructor Documentation

7.1.1.1 BlackPiece()

```
OGRID::BlackPiece::BlackPiece (
            Player ∗ player )
```

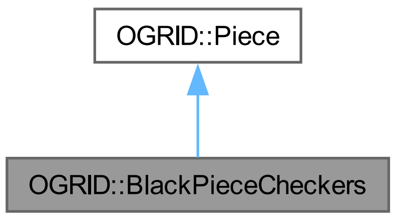
The documentation for this class was generated from the following files:

- Source/ogrid/Games/ConnectFour/ConnectFourPieces.h
- Source/ogrid/Games/ConnectFour/ConnectFourPieces.cpp

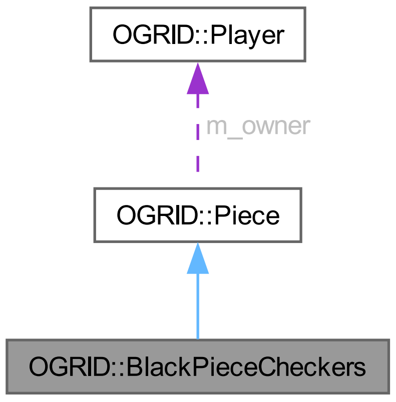
7.2 OGRID::BlackPieceCheckers Class Reference

```
#include <CheckersPieces.h>
```

Inheritance diagram for OGRID::BlackPieceCheckers:



Collaboration diagram for OGRID::BlackPieceCheckers:



Public Member Functions

- BlackPieceCheckers (Player ∗player)

Public Member Functions inherited from OGRID::Piece

- Piece (std::string rep, Player ∗player)

  Construct a new Piece object.
- ∼Piece ()

  Destroy the Piece object.

- void AddMoveRule (IMoveRule ∗rule)

  Add a move rule to the piece.
- void AddAttackRule (IAttackRule ∗rule)

  Add an attack rule to the piece.
- const std::string & GetRepresentation () const

  Get the representation of the piece.
- const Player ∗ GetOwner () const

  Get the owner of the piece.
- void SetOwner (Player ∗player)

  Set the owner of the piece.
- bool isValidMove (Grid ∗grid, int fromX, int fromY, int toX, int toY) const

  Check if the move is valid.
- bool isValidAttack (Grid ∗grid, int fromX, int fromY, int toX, int toY, bool &canContinue) const

  Check if the attack is valid.

Additional Inherited Members

Protected Attributes inherited from OGRID::Piece

- std::string m_representation

  The representation of the piece.
- std::vector< IMoveRule ∗ > m_moveRules

  The move rules of the piece.
- std::vector< IAttackRule ∗ > m_attackRules

  The attack rules of the piece.
- Player ∗ m_owner

  The owner of the piece.

7.2.1 Constructor & Destructor Documentation

7.2.1.1 BlackPieceCheckers()

```
OGRID::BlackPieceCheckers::BlackPieceCheckers (
            Player * player )
```

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

- Source/ogrid/Games/Checkers/CheckersPieces.h
- Source/ogrid/Games/Checkers/CheckersPieces.cpp

7.3 OGRID::Button Struct Reference

Button.

```
#include <Button.h>
```

Public Member Functions

- Button (Rectangle bounds, Color normal, Color hover, Color pressed, std::function< void()> clickCallback, std::string text="Button", bool isEnabled=true)

    Construct a new Button object.
- void Update ()

    Update the button.
- void Draw () const

    Draw the button.
- void SetEnabled (bool enabled)

    Set the enabled state of the button.

Public Attributes

- Rectangle bounds

    Bounds of the button.
- Color normalColor

    Colors of the button.
- Color hoverColor

    Colors of the button on mouse hover.
- Color pressedColor

    Colors of the button on mouse click.
- std::function< void()> onClick

    Delegate function for button click event.
- std::string text

    Text to be displayed on the button.
- bool isEnabled

    Is the button enabled.

7.3.1 Detailed Description

Button.

This is a warpper around raylib's Rectangle and Color.

Date

2023-12-06

See also

`https://www.raylib.com/`

7.3.2 Constructor & Destructor Documentation

7.3.2.1 Button()

```
OGRID::Button::Button (
            Rectangle bounds,
            Color normal,
            Color hover,
            Color pressed,
            std::function< void()> clickCallback,
            std::string text = "Button",
            bool isEnabled = true )  [inline]
```

Construct a new Button object.

Date

2023-12-06

Parameters

bounds	Bounds of the button.
normal	Color of the button when not interacted with.
hover	Color of the button on mouse hover.
pressed	Color of the button when pressed.
clickCallback	Delegate function for button click event.
text	Text to be displayed on the button.
isEnabled	Is the button enabled.

7.3.3 Member Function Documentation

7.3.3.1 Draw()

```
void OGRID::Button::Draw ( ) const  [inline]
```

Draw the button.

Date

2023-12-06

7.3.3.2 SetEnabled()

```
void OGRID::Button::SetEnabled (
            bool enabled )  [inline]
```

Set the enabled state of the button.

Date

2023-12-06

Parameters

enabled	Is the button enabled.

7.3.3.3 Update()

```
void OGRID::Button::Update ( )  [inline]
```

Update the button.

Date

2023-12-06

7.3.4 Member Data Documentation

7.3.4.1 bounds

```
Rectangle OGRID::Button::bounds
```

Bounds of the button.

Date

2023-12-06

7.3.4.2 hoverColor

```
Color OGRID::Button::hoverColor
```

Colors of the button on mouse hover.

Date

2023-12-06

7.3.4.3 isEnabled

```
bool OGRID::Button::isEnabled
```

Is the button enabled.

Date

2023-12-06

7.3.4.4 normalColor

```
Color OGRID::Button::normalColor
```

Colors of the button.

Date

2023-12-06

7.3.4.5 onClick

```
std::function<void()> OGRID::Button::onClick
```

Delegate function for button click event.

Date

2023-12-06

7.3.4.6 pressedColor

```
Color OGRID::Button::pressedColor
```

Colors of the button on mouse click.

Date

2023-12-06

7.3.4.7 text

```
std::string OGRID::Button::text
```

Text to be displayed on the button.

Date

2023-12-06

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

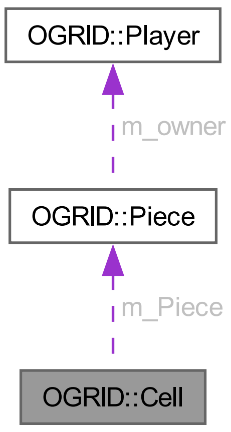
- Source/ogrid/GUI/Button.h

7.4 OGRID::Cell Struct Reference

The Cell struct represents a single cell in the grid.

```
#include <Grid.h>
```

Collaboration diagram for OGRID::Cell:



Public Attributes

- Piece $*$ m_Piece
- unsigned char m_Row
- unsigned char m_Col

7.4.1 Detailed Description

The Cell struct represents a single cell in the grid.

It contains a pointer to a Piece and the row and column of the cell. The Piece pointer can be nullptr if the cell is empty. The row and column are unsigned chars, which means the grid can be at most 255x255. This is more than enough for our purposes.

Date

2023-12-06

See also

Piece

Grid

7.4.2 Member Data Documentation

7.4.2.1 m_Col

```
unsigned char OGRID::Cell::m_Col
```

7.4.2.2 m_Piece

```
Piece* OGRID::Cell::m_Piece
```

7.4.2.3 m_Row

```
unsigned char OGRID::Cell::m_Row
```

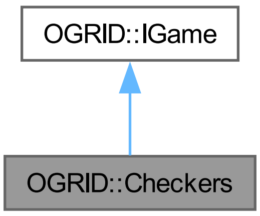
The documentation for this struct was generated from the following file:

- Source/ogrid/Grid/Grid.h

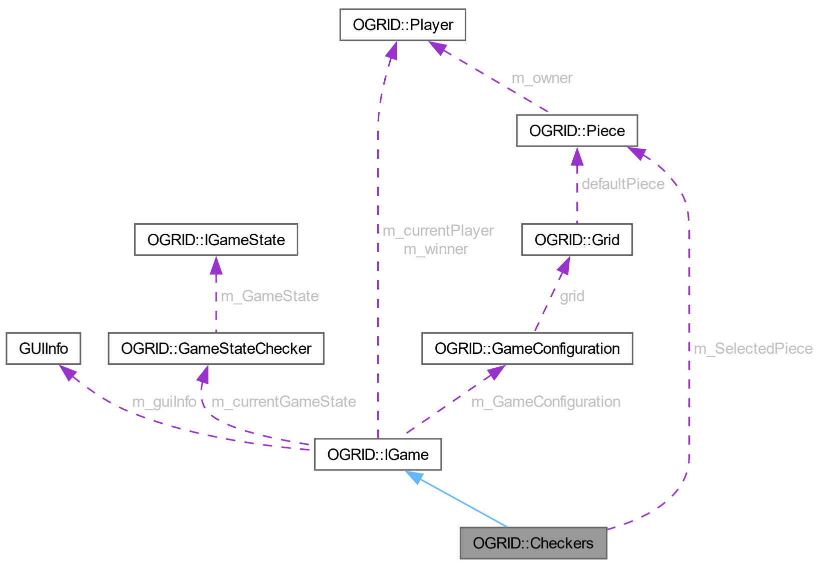
7.5 OGRID::Checkers Class Reference

```
#include <Checkers.h>
```

Inheritance diagram for OGRID::Checkers:



Collaboration diagram for OGRID::Checkers:



Public Member Functions

- Checkers ()=default
- ∼Checkers ()=default
- bool TryMakeMove (unsigned char &row, unsigned char &col) override

  Try making a move with the current player.
- bool IsWinningCondition () override

  Check if the winning condition is met.
- bool IsDrawCondition () override

  Check if the draw condition is met.
- void SetupPlayers () override

  Setup the players of the game.
- void Initialize () override

  Setup the core of the game.
- void OnGUIUpdateGrid () override

  Update the game's GUI.
- void OnGUIUpdateGridHover (Vector2 cell) override

  Update the game's GUI when hovering over a specific Cell.

Public Member Functions inherited from OGRID::IGame

- void SwapPlayerPositions ()

  Switches the current player to the next player.
- void ResetGrid ()

  Call the Grid object's ResetGrid() function.
- void ResetPlayers ()

  Reset the players of the game.

- void PrintPlayersTurnOrder () const

  Print the players of the game.
- void SetupGame ()

  Sets up the game.
- void ResetGame ()

  Resets the game.
- void StartGame ()

  Starts the game.
- void PrintPlayerMoves () const

  Prints the turn order.
- void MakeMove (unsigned char row, unsigned char col)

  Attempts to make a move.
- void Reset ()

  Resets the game.
- void SwitchPlayer ()

  Sets the current player to the next player.
- OGRID::GameOverType CheckGameOverState (OGRID::Grid ∗grid, unsigned char row, unsigned char col)

  Checks if the game is over.
- GameState GetGameState () const

  Get the state of the game.
- void SetGameState (GameState gameState)

  Set the state of the game.
- GameOverType GetGameOverType () const

  Get the game loop state of the game.
- Player ∗ GetWinner () const

  Get the winner of the game.
- GameConfiguration ∗ GetGameConfiguration () const

  Get the GameConfiguration object.
- void SetGameConfiguration (GameConfiguration ∗gameConfiguration)

  Set the GameConfiguration object.
- std::string GetGameName () const

  Get the name of the game.
- Grid ∗ GetGrid () const

  Get the Grid object of the game.
- std::vector< Player ∗ > GetPlayers () const

  Get the a Vector of the players of the game.
- void SetRandomizeTurnOrder (bool randomize)

  Toggle the randomization of the turn order.
- OGRID::PlayerNameAndPtr GetCurrentPlayer () const

  Get the current player of the game.
- void SetCurrentPlayer (OGRID::PlayerNameAndPtr player)

  Set the current player of the game.
- size_t GetCurrentTurn () const

  Get the current turn of the game.
- GameStateChecker ∗ GetGameStateChecker () const

  Get the current state of the game.
- void SetGameStateChecker (GameStateChecker ∗gameStateChecker)

  Set the current state of the game.
- std::vector< std::string > GetPlayerNames () const

  Get a Vector of the names of the players from GameConfiguration.
- std::vector< OGRID::Player ∗ > GetPlayerPtrs () const

Get a Vector of the pointers of the players from GameConfiguration.
- OGRID::PlayerNameAndPtr GetPlayerPair (size_t at) const

  Get the name and pointer of the player from GameConfiguration.
- std::vector< OGRID::PlayerNameAndPtr > GetPlayerPairs () const

  Get a Vector of the names and pointers of the players from GameConfiguration.
- void SetPlayerPairs (const std::vector< OGRID::PlayerNameAndPtr > &players)

  Set the names and pointers of the players from GameConfiguration.
- GUIInfo GetGUIInfo () const

  Get the GUIInfo object.
- void SetGUIInfo (const GUIInfo &guiInfo)

  Set the GUIInfo object.

Private Member Functions

- void SetupBoard ()
- void AddAsSuperPiece (Piece ∗piece)
- void RemoveSuperPiece (Piece ∗piece)
- bool IsSuperPiece (Piece ∗piece)
- void AddPieceToPieceManager (Piece ∗piece, std::pair< int, int > position)
- void RemovePieceFromPieceManager (Piece ∗piece)
- void RemovePieceFromPieceManager (std::pair< int, int > position)
- std::pair< int, int > GetPiecePosition (Piece ∗piece)
- void SetPiecePosition (Piece ∗piece, std::pair< int, int > position)
- void DrawPiece (int row, int col, Color color, bool blinking, bool super)
- void DrawCell (int row, int col)

Private Attributes

- std::vector< Piece ∗ > m_Supers
- Piece ∗ m_SelectedPiece = nullptr
- std::map< std::pair< int, int >, Piece ∗ > m_Pieces
- float alpha = 1.0f
- float alphaSpeed = 0.025f

Additional Inherited Members

Public Attributes inherited from OGRID::IGame

- GUIInfo m_guiInfo
- bool m_randomizeTurnOrder = true

  To randomize the turn order of the players.

Protected Member Functions inherited from OGRID::IGame

- IGame ()=default

  The constructor of the IGame class.
- IGame (IGameState ∗gameStateStrategy, const std::vector< OGRID::PlayerNameAndPtr > &players)

  The constructor of the IGame class.
- ∼IGame ()

  The destructor of the IGame class.

Protected Attributes inherited from OGRID::IGame

- GameStateChecker ∗ m_currentGameState

  Holds the logic to check the state of the specific game.
- GameState m_gameState = GameState::NotStarted

  The state of the game.
- GameOverType m_gameOverType = GameOverType::None

  The game loop state of the game.
- Player ∗ m_winner = nullptr

  The winner of the game.
- Player ∗ m_currentPlayer = nullptr

  The current player of the game.
- size_t m_currentTurn = 0

  The current turn of the game.
- unsigned int m_totalTurns = 0

  Keeps the total number of turns.
- GameConfiguration ∗ m_GameConfiguration = nullptr

  The GameConfiguration object.

7.5.1 Constructor & Destructor Documentation

7.5.1.1 Checkers()

```
OGRID::Checkers::Checkers ( )  [default]
```

7.5.1.2 ∼Checkers()

```
OGRID::Checkers::∼Checkers ( )  [default]
```

7.5.2 Member Function Documentation

7.5.2.1 AddAsSuperPiece()

```
void OGRID::Checkers::AddAsSuperPiece (
            Piece ∗ piece )  [private]
```

7.5.2.2 AddPieceToPieceManager()

```
void OGRID::Checkers::AddPieceToPieceManager (
            Piece ∗ piece,
            std::pair< int, int > position )  [private]
```

7.5.2.3 DrawCell()

```
void OGRID::Checkers::DrawCell (
            int row,
            int col )  [private]
```

7.5.2.4 DrawPiece()

```
void OGRID::Checkers::DrawPiece (
             int row,
             int col,
             Color color,
             bool blinking = false,
             bool super = false )  [private]
```

7.5.2.5 GetPiecePosition()

```
std::pair< int, int > OGRID::Checkers::GetPiecePosition (
             Piece * piece )  [private]
```

7.5.2.6 Initialize()

```
void OGRID::Checkers::Initialize ( )  [override], [virtual]
```

Setup the core of the game.

Date

2023-12-06

Implements OGRID::IGame.

7.5.2.7 IsDrawCondition()

```
bool OGRID::Checkers::IsDrawCondition ( )  [override], [virtual]
```

Check if the draw condition is met.

Date

2023-12-06

Returns

True if the draw condition is met, false otherwise.

Implements OGRID::IGame.

7.5.2.8 IsSuperPiece()

```
bool OGRID::Checkers::IsSuperPiece (
             Piece * piece )  [private]
```

7.5.2.9 IsWinningCondition()

```
bool OGRID::Checkers::IsWinningCondition ( )  [override], [virtual]
```

Check if the winning condition is met.

Date

2023-12-06

Returns

True if the winning condition is met, false otherwise.

Implements OGRID::IGame.

7.5.2.10 OnGUIUpdateGrid()

```
void OGRID::Checkers::OnGUIUpdateGrid ( )  [override], [virtual]
```

Update the game's GUI.

Date

2023-12-06

Implements OGRID::IGame.

7.5.2.11 OnGUIUpdateGridHover()

```
void OGRID::Checkers::OnGUIUpdateGridHover (
             Vector2 cell )  [override], [virtual]
```

Update the game's GUI when hovering over a specific Cell.

Date

2023-12-06

Parameters

cell	The cell of the grid.

See also

Cell

Implements OGRID::IGame.

7.5.2.12 RemovePieceFromPieceManager() [1/2]

```
void OGRID::Checkers::RemovePieceFromPieceManager (
            Piece * piece )  [private]
```

7.5.2.13 RemovePieceFromPieceManager() [2/2]

```
void OGRID::Checkers::RemovePieceFromPieceManager (
            std::pair< int, int > position )  [private]
```

7.5.2.14 RemoveSuperPiece()

```
void OGRID::Checkers::RemoveSuperPiece (
            Piece * piece )  [private]
```

7.5.2.15 SetPiecePosition()

```
void OGRID::Checkers::SetPiecePosition (
            Piece * piece,
            std::pair< int, int > position )  [private]
```

7.5.2.16 SetupBoard()

```
void OGRID::Checkers::SetupBoard ( )  [private]
```

7.5.2.17 SetupPlayers()

```
void OGRID::Checkers::SetupPlayers ( )  [override], [virtual]
```

Setup the players of the game.

Date

2023-12-06

Implements OGRID::IGame.

7.5.2.18 TryMakeMove()

```
bool OGRID::Checkers::TryMakeMove (
            unsigned char & row,
            unsigned char & col )  [override], [virtual]
```

Try making a move with the current player.

Date

2023-12-06

Parameters

row	The row of the grid.
col	The column of the grid.

Implements OGRID::IGame.

7.5.3 Member Data Documentation

7.5.3.1 alpha

```
float OGRID::Checkers::alpha = 1.0f  [private]
```

7.5.3.2 alphaSpeed

```
float OGRID::Checkers::alphaSpeed = 0.025f  [private]
```

7.5.3.3 m_Pieces

```
std::map<std::pair<int, int>, Piece *> OGRID::Checkers::m_Pieces  [private]
```

7.5.3.4 m_SelectedPiece

```
Piece* OGRID::Checkers::m_SelectedPiece = nullptr  [private]
```

7.5.3.5 m_Supers

```
std::vector<Piece *> OGRID::Checkers::m_Supers  [private]
```

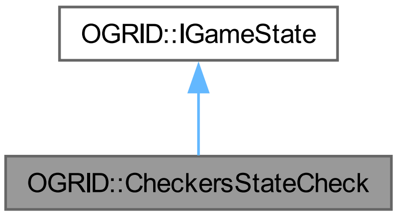
The documentation for this class was generated from the following files:

- Source/ogrid/Games/Checkers/Checkers.h
- Source/ogrid/Games/Checkers/Checkers.cpp

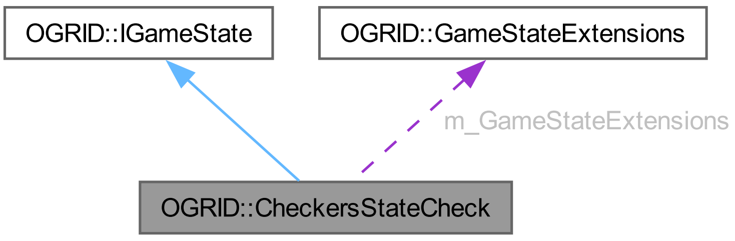
7.6 OGRID::CheckersStateCheck Class Reference

```
#include <CheckersStateCheck.h>
```

Inheritance diagram for OGRID::CheckersStateCheck:



Collaboration diagram for OGRID::CheckersStateCheck:



Public Member Functions

- int CheckWin (Grid ∗grid) const override

  Check if the game is over.
- bool IsDraw (Grid ∗grid) const override

  Check if the game is a draw.

Public Member Functions inherited from OGRID::IGameState

- virtual ∼IGameState ()

  Destroy the IGameState object.

Private Attributes

- GameStateExtensions m_GameStateExtensions = GameStateExtensions()

7.6.1 Member Function Documentation

7.6.1.1 CheckWin()

```
int OGRID::CheckersStateCheck::CheckWin (
            Grid * grid ) const  [override], [virtual]
```

Check if the game is over.

Check if the game is over using the strategy.

Date

2023-12-06

Parameters

grid	The grid of the game.

Returns

True if the game is over, false otherwise.

Implements OGRID::IGameState.

7.6.1.2 IsDraw()

```
bool OGRID::CheckersStateCheck::IsDraw (
            Grid * grid ) const  [override], [virtual]
```

Check if the game is a draw.

Check if the game is a draw using the strategy.

Date

2023-12-06

Parameters

grid	The grid of the game.

Returns

True if the game is a draw, false otherwise.

Implements OGRID::IGameState.

7.6.2 Member Data Documentation

7.6.2.1 m_GameStateExtensions

```
GameStateExtensions OGRID::CheckersStateCheck::m_GameStateExtensions = GameStateExtensions()
[private]
```

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

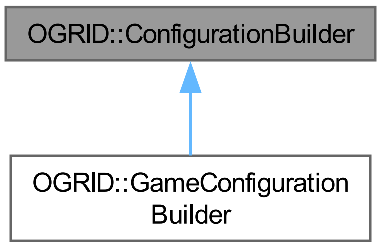
- Source/ogrid/Games/Checkers/CheckersStateCheck.h
- Source/ogrid/Games/Checkers/CheckersStateCheck.cpp

7.7 OGRID::ConfigurationBuilder Struct Reference

The ConfigurationBuilder interface.

```
#include <GameConfiguration.h>
```

Inheritance diagram for OGRID::ConfigurationBuilder:



Public Member Functions

- virtual ∼ConfigurationBuilder ()=default

  Destroy the ConfigurationBuilder object.
- virtual ConfigurationBuilder & setGameName (const std::string &gameName)=0

  Set the name of the game.
- virtual ConfigurationBuilder & setGameDescription (const std::string &gameDescription)=0

  Set the description of the game.
- virtual ConfigurationBuilder & setGrid (unsigned char rows, unsigned char cols, Piece ∗default←
Piece=nullptr)=0

  Set the grid of the game.
- virtual ConfigurationBuilder & setMaxPlayers (size_t maxPlayers)=0

  Set the maximum number of players.
- virtual ConfigurationBuilder & addPlayer (Player ∗player)=0

  Add a player to the game.
- virtual GameConfiguration ∗ build ()=0

  Build the GameConfiguration object.

7.7.1 Detailed Description

The ConfigurationBuilder interface.

It is used to build a GameConfiguration object.

Date

2023-12-06

See also

GameConfiguration

7.7.2 Constructor & Destructor Documentation

7.7.2.1 ∼ConfigurationBuilder()

```
virtual OGRID::ConfigurationBuilder::∼ConfigurationBuilder ( )  [virtual], [default]
```

Destroy the ConfigurationBuilder object.

Date

2023-12-06

See also

GameConfiguration

GameConfigurationBuilder

7.7.3 Member Function Documentation

7.7.3.1 addPlayer()

```
virtual ConfigurationBuilder & OGRID::ConfigurationBuilder::addPlayer (
            Player * player )  [pure virtual]
```

Add a player to the game.

Date

2023-12-06

Parameters

player	The player to be added.

Returns

The ConfigurationBuilder object.

See also

GameConfiguration

Implemented in OGRID::GameConfigurationBuilder.

7.7.3.2 build()

```
virtual GameConfiguration * OGRID::ConfigurationBuilder::build ( )  [pure virtual]
```

Build the GameConfiguration object.

Date

2023-12-06

Returns

The GameConfiguration object.

See also

GameConfiguration

Implemented in OGRID::GameConfigurationBuilder.

7.7.3.3 setGameDescription()

```
virtual ConfigurationBuilder & OGRID::ConfigurationBuilder::setGameDescription (
              const std::string & gameDescription )  [pure virtual]
```

Set the description of the game.

Date

2023-12-06

Parameters

gameDescription	The description of the game.

Returns

The ConfigurationBuilder object.

See also

GameConfiguration

Implemented in OGRID::GameConfigurationBuilder.

7.7.3.4 setGameName()

```
virtual ConfigurationBuilder & OGRID::ConfigurationBuilder::setGameName (
             const std::string & gameName )  [pure virtual]
```

Set the name of the game.

Date

2023-12-06

Parameters

gameName	The name of the game.

Returns

The ConfigurationBuilder object.

See also

GameConfiguration

Implemented in OGRID::GameConfigurationBuilder.

7.7.3.5 setGrid()

```
virtual ConfigurationBuilder & OGRID::ConfigurationBuilder::setGrid (
             unsigned char rows,
             unsigned char cols,
             Piece * defaultPiece = nullptr )  [pure virtual]
```

Set the grid of the game.

Date

2023-12-06

Parameters

rows	The number of rows of the grid.
cols	The number of columns of the grid.
defaultPiece	The default piece of the grid.

Returns

The ConfigurationBuilder object.

See also

GameConfiguration

Implemented in OGRID::GameConfigurationBuilder.

7.7.3.6 setMaxPlayers()

```
virtual ConfigurationBuilder & OGRID::ConfigurationBuilder::setMaxPlayers (
             size_t maxPlayers )  [pure virtual]
```

Set the maximum number of players.

Date

2023-12-06

Parameters

maxPlayers	The maximum number of players.

Returns

The ConfigurationBuilder object.

See also

GameConfiguration

Implemented in OGRID::GameConfigurationBuilder.

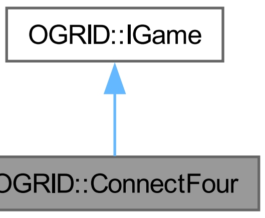
The documentation for this struct was generated from the following file:

- Source/ogrid/GameLogicImplementation/GameConfiguration.h

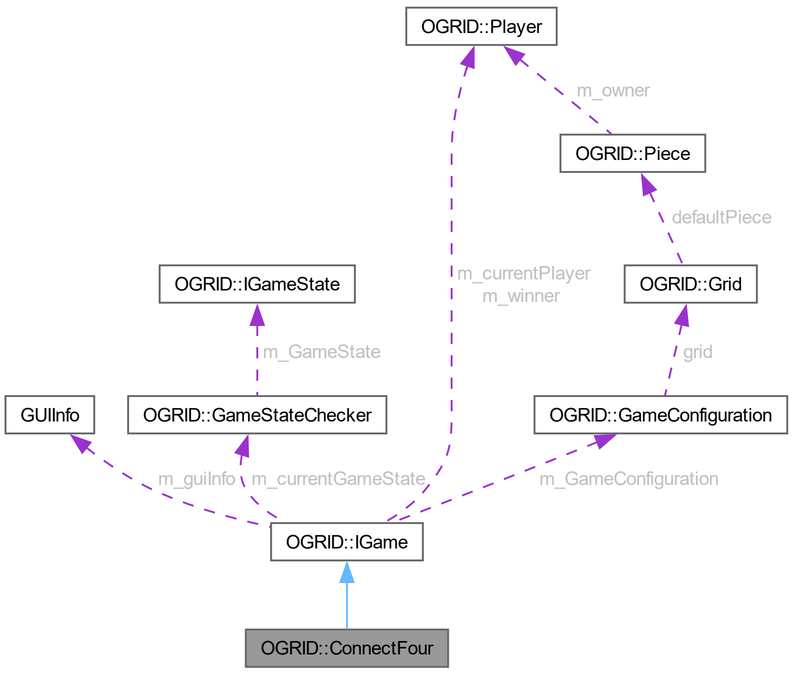
7.8 OGRID::ConnectFour Class Reference

```
#include <ConnectFour.h>
```

Inheritance diagram for OGRID::ConnectFour:



Collaboration diagram for OGRID::ConnectFour:



Public Member Functions

- ConnectFour ()=default
- ∼ConnectFour ()=default
- bool TryMakeMove (unsigned char &row, unsigned char &col) override

  Try making a move with the current player.
- bool IsWinningCondition () override

  Check if the winning condition is met.
- bool IsDrawCondition () override

Check if the draw condition is met.
- void SetupPlayers () override

  Setup the players of the game.
- void Initialize () override

  Setup the core of the game.
- void OnGUIUpdateGrid () override

  Update the game's GUI.
- void OnGUIUpdateGridHover (Vector2 cell) override

  Update the game's GUI when hovering over a specific Cell.

Public Member Functions inherited from OGRID::IGame

- void SwapPlayerPositions ()

  Switches the current player to the next player.
- void ResetGrid ()

  Call the Grid object's ResetGrid() function.
- void ResetPlayers ()

  Reset the players of the game.
- void PrintPlayersTurnOrder () const

  Print the players of the game.
- void SetupGame ()

  Sets up the game.
- void ResetGame ()

  Resets the game.
- void StartGame ()

  Starts the game.
- void PrintPlayerMoves () const

  Prints the turn order.
- void MakeMove (unsigned char row, unsigned char col)

  Attempts to make a move.
- void Reset ()

  Resets the game.
- void SwitchPlayer ()

  Sets the current player to the next player.
- OGRID::GameOverType CheckGameOverState (OGRID::Grid ∗grid, unsigned char row, unsigned char col)

  Checks if the game is over.
- GameState GetGameState () const

  Get the state of the game.
- void SetGameState (GameState gameState)

  Set the state of the game.
- GameOverType GetGameOverType () const

  Get the game loop state of the game.
- Player ∗ GetWinner () const

  Get the winner of the game.
- GameConfiguration ∗ GetGameConfiguration () const

  Get the GameConfiguration object.
- void SetGameConfiguration (GameConfiguration ∗gameConfiguration)

  Set the GameConfiguration object.
- std::string GetGameName () const

  Get the name of the game.

- Grid ∗ GetGrid () const

    Get the Grid object of the game.
- std::vector< Player ∗ > GetPlayers () const

    Get the a Vector of the players of the game.
- void SetRandomizeTurnOrder (bool randomize)

    Toggle the randomization of the turn order.
- OGRID::PlayerNameAndPtr GetCurrentPlayer () const

    Get the current player of the game.
- void SetCurrentPlayer (OGRID::PlayerNameAndPtr player)

    Set the current player of the game.
- size_t GetCurrentTurn () const

    Get the current turn of the game.
- GameStateChecker ∗ GetGameStateChecker () const

    Get the current state of the game.
- void SetGameStateChecker (GameStateChecker ∗gameStateChecker)

    Set the current state of the game.
- std::vector< std::string > GetPlayerNames () const

    Get a Vector of the names of the players from GameConfiguration.
- std::vector< OGRID::Player ∗ > GetPlayerPtrs () const

    Get a Vector of the pointers of the players from GameConfiguration.
- OGRID::PlayerNameAndPtr GetPlayerPair (size_t at) const

    Get the name and pointer of the player from GameConfiguration.
- std::vector< OGRID::PlayerNameAndPtr > GetPlayerPairs () const

    Get a Vector of the names and pointers of the players from GameConfiguration.
- void SetPlayerPairs (const std::vector< OGRID::PlayerNameAndPtr > &players)

    Set the names and pointers of the players from GameConfiguration.
- GUIInfo GetGUIInfo () const

    Get the GUIInfo object.
- void SetGUIInfo (const GUIInfo &guiInfo)

    Set the GUIInfo object.

Private Member Functions

- void DrawCircle (int row, int col, Color color, bool blinking=false)

Private Attributes

- float alpha = 1.0f
- float alphaSpeed = 0.025f

Additional Inherited Members

Public Attributes inherited from OGRID::IGame

- GUIInfo m_guiInfo
- bool m_randomizeTurnOrder = true

    To randomize the turn order of the players.

Protected Member Functions inherited from OGRID::IGame

- IGame ()=default

  The constructor of the IGame class.
- IGame (IGameState ∗gameStateStrategy, const std::vector< OGRID::PlayerNameAndPtr > &players)

  The constructor of the IGame class.
- ∼IGame ()

  The destructor of the IGame class.

Protected Attributes inherited from OGRID::IGame

- GameStateChecker ∗ m_currentGameState

  Holds the logic to check the state of the specific game.
- GameState m_gameState = GameState::NotStarted

  The state of the game.
- GameOverType m_gameOverType = GameOverType::None

  The game loop state of the game.
- Player ∗ m_winner = nullptr

  The winner of the game.
- Player ∗ m_currentPlayer = nullptr

  The current player of the game.
- size_t m_currentTurn = 0

  The current turn of the game.
- unsigned int m_totalTurns = 0

  Keeps the total number of turns.
- GameConfiguration ∗ m_GameConfiguration = nullptr

  The GameConfiguration object.

7.8.1 Constructor & Destructor Documentation

7.8.1.1 ConnectFour()

```
OGRID::ConnectFour::ConnectFour ( )  [default]
```

7.8.1.2 ∼ConnectFour()

```
OGRID::ConnectFour::∼ConnectFour ( )  [default]
```

7.8.2 Member Function Documentation

7.8.2.1 DrawCircle()

```
void OGRID::ConnectFour::DrawCircle (
             int row,
             int col,
             Color color,
             bool blinking = false )  [private]
```

7.8.2.2 Initialize()

```
void OGRID::ConnectFour::Initialize ( )  [override], [virtual]
```

Setup the core of the game.

Date

2023-12-06

Implements OGRID::IGame.

7.8.2.3 IsDrawCondition()

```
bool OGRID::ConnectFour::IsDrawCondition ( )  [override], [virtual]
```

Check if the draw condition is met.

Date

2023-12-06

Returns

True if the draw condition is met, false otherwise.

Implements OGRID::IGame.

7.8.2.4 IsWinningCondition()

```
bool OGRID::ConnectFour::IsWinningCondition ( )  [override], [virtual]
```

Check if the winning condition is met.

Date

2023-12-06

Returns

True if the winning condition is met, false otherwise.

Implements OGRID::IGame.

7.8.2.5 OnGUIUpdateGrid()

```
void OGRID::ConnectFour::OnGUIUpdateGrid ( )  [override], [virtual]
```

Update the game's GUI.

Date

2023-12-06

Implements OGRID::IGame.

7.8.2.6 OnGUIUpdateGridHover()

```
void OGRID::ConnectFour::OnGUIUpdateGridHover (
             Vector2 cell )  [override], [virtual]
```

Update the game's GUI when hovering over a specific Cell.

Date

2023-12-06

Parameters

cell	The cell of the grid.

See also

Cell

Implements OGRID::IGame.

7.8.2.7 SetupPlayers()

```
void OGRID::ConnectFour::SetupPlayers ( )  [override], [virtual]
```

Setup the players of the game.

Date

2023-12-06

Implements OGRID::IGame.

7.8.2.8 TryMakeMove()

```
bool OGRID::ConnectFour::TryMakeMove (
             unsigned char & row,
             unsigned char & col )  [override], [virtual]
```

Try making a move with the current player.

Date

2023-12-06

Parameters

row	The row of the grid.
col	The column of the grid.

Implements OGRID::IGame.

7.8.3 Member Data Documentation

7.8.3.1 alpha

```
float OGRID::ConnectFour::alpha = 1.0f  [private]
```

7.8.3.2 alphaSpeed

```
float OGRID::ConnectFour::alphaSpeed = 0.025f  [private]
```

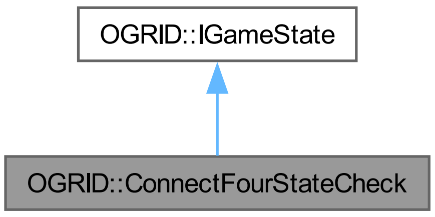
The documentation for this class was generated from the following files:

- Source/ogrid/Games/ConnectFour/ConnectFour.h
- Source/ogrid/Games/ConnectFour/ConnectFour.cpp

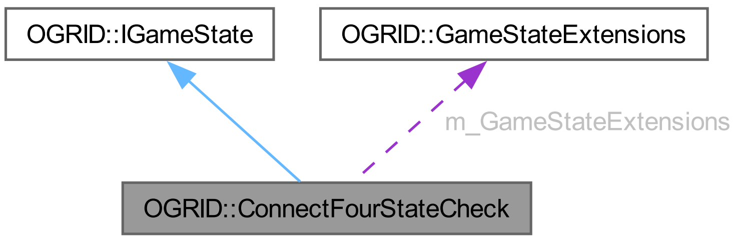
7.9 OGRID::ConnectFourStateCheck Class Reference

```
#include <ConnectFourStateCheck.h>
```

Inheritance diagram for OGRID::ConnectFourStateCheck:



Collaboration diagram for OGRID::ConnectFourStateCheck:



Public Member Functions

- int CheckWin (Grid ∗grid) const override

    Check if the game is over.
- bool IsDraw (Grid ∗grid) const override

    Check if the game is a draw.

Public Member Functions inherited from OGRID::IGameState

- virtual ∼IGameState ()

    Destroy the IGameState object.

Private Attributes

- GameStateExtensions m_GameStateExtensions = GameStateExtensions()

7.9.1 Member Function Documentation

7.9.1.1 CheckWin()

```
int OGRID::ConnectFourStateCheck::CheckWin (
            Grid * grid ) const  [override], [virtual]
```

Check if the game is over.

Check if the game is over using the strategy.

Date

2023-12-06

Parameters

grid	The grid of the game.

Returns

True if the game is over, false otherwise.

Implements OGRID::IGameState.

7.9.1.2 IsDraw()

```
bool OGRID::ConnectFourStateCheck::IsDraw (
            Grid * grid ) const  [override], [virtual]
```

Check if the game is a draw.

Check if the game is a draw using the strategy.

Date

2023-12-06

Parameters

grid	The grid of the game.

Returns

True if the game is a draw, false otherwise.

Implements OGRID::IGameState.

7.9.2 Member Data Documentation

7.9.2.1 m_GameStateExtensions

```
GameStateExtensions OGRID::ConnectFourStateCheck::m_GameStateExtensions = GameStateExtensions()
[private]
```

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

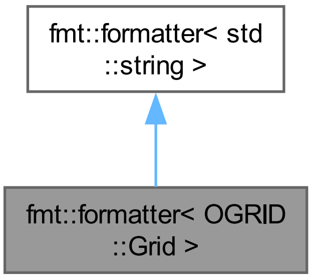
- Source/ogrid/Games/ConnectFour/ConnectFourStateCheck.h
- Source/ogrid/Games/ConnectFour/ConnectFourStateCheck.cpp

7.10 fmt::formatter< OGRID::Grid > Struct Reference

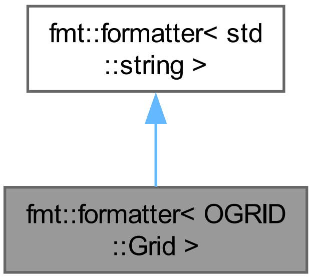
This is used to format a Grid object into a string using fmt.

`#include <ogrid_fmt.h>`

Inheritance diagram for fmt::formatter< OGRID::Grid >:


fmt::formatter< std ::string >
fmt::formatter< OGRID ::Grid >

Collaboration diagram for fmt::formatter< OGRID::Grid >:


fmt::formatter< std ::string >
fmt::formatter< OGRID ::Grid >

Public Member Functions

- constexpr auto parse (format_parse_context &ctx)
- template<typename FormatContext >
  auto format (const OGRID::Grid &grid, FormatContext &ctx)

7.10.1 Detailed Description

This is used to format a Grid object into a string using fmt.

This should be used like this: fmt::format("{}", grid);

Date

2023-12-06

7.10.2 Member Function Documentation

7.10.2.1 format()

```
template<typename FormatContext >
auto fmt::formatter< OGRID::Grid >::format (
            const OGRID::Grid & grid,
            FormatContext & ctx ) [inline]
```

7.10.2.2 parse()

```
constexpr auto fmt::formatter< OGRID::Grid >::parse (
            format_parse_context & ctx ) [inline], [constexpr]
```

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

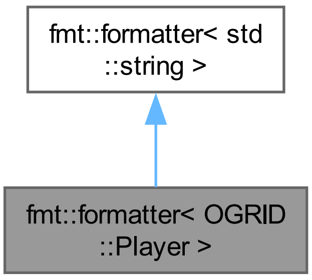
- Source/ogrid/ogrid_fmt.h

7.11 fmt::formatter< OGRID::Player > Struct Reference

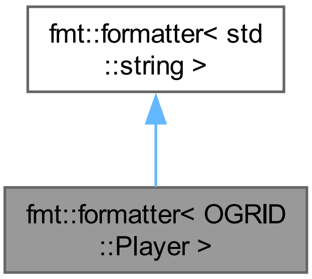
This is used to format a Player object into a string using fmt.

```
#include <ogrid_fmt.h>
```

Inheritance diagram for fmt::formatter< OGRID::Player >:


fmt::formatter< std ::string >
fmt::formatter< OGRID ::Player >

Collaboration diagram for fmt::formatter< OGRID::Player >:


fmt::formatter< std ::string >
fmt::formatter< OGRID ::Player >

Public Member Functions

- constexpr auto parse (format_parse_context &ctx)
- template<typename FormatContext >
  auto format (const OGRID::Player &player, FormatContext &ctx)

7.11.1 Detailed Description

This is used to format a Player object into a string using fmt.

This should be used like this: fmt::format("{}", player);

Date

2023-12-06

7.11.2 Member Function Documentation

7.11.2.1 format()

```
template<typename FormatContext >
auto fmt::formatter< OGRID::Player >::format (
            const OGRID::Player & player,
            FormatContext & ctx )  [inline]
```

7.11.2.2 parse()

```
constexpr auto fmt::formatter< OGRID::Player >::parse (
            format_parse_context & ctx )  [inline], [constexpr]
```

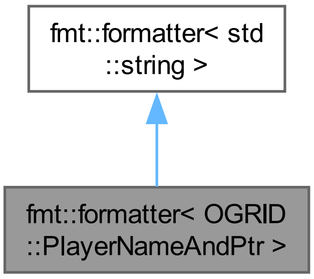
The documentation for this struct was generated from the following file:

- Source/ogrid/ogrid_fmt.h

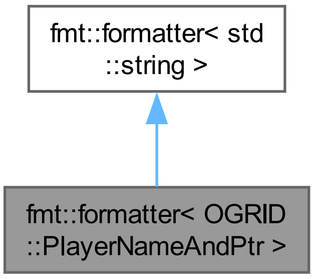
7.12 fmt::formatter< OGRID::PlayerNameAndPtr > Struct Reference

This is used to format a PlayerType enum into a string using fmt.

Inheritance diagram for fmt::formatter< OGRID::PlayerNameAndPtr >:


fmt::formatter< std ::string >
fmt::formatter< OGRID ::PlayerNameAndPtr >

Collaboration diagram for fmt::formatter< OGRID::PlayerNameAndPtr >:


fmt::formatter< std ::string >
fmt::formatter< OGRID ::PlayerNameAndPtr >

Public Member Functions

- constexpr auto parse (format_parse_context &ctx)
- template<typename FormatContext >
  auto format (const OGRID::PlayerNameAndPtr &player, FormatContext &ctx)

7.12.1 Detailed Description

This is used to format a PlayerType enum into a string using fmt.

This should be used like this: fmt::format("{}", playerType); The reason this is here and not in ogrid_fmt.h is because it needs to be declared before the Player class.

Date

2023-12-06

Todo Try to move this to ogrid_fmt.h at some point.

7.12.2 Member Function Documentation

7.12.2.1 format()

```
template<typename FormatContext >
auto fmt::formatter< OGRID::PlayerNameAndPtr >::format (
             const OGRID::PlayerNameAndPtr & player,
             FormatContext & ctx )  [inline]
```

7.12.2.2 parse()

```
constexpr auto fmt::formatter< OGRID::PlayerNameAndPtr >::parse (
             format_parse_context & ctx )  [inline], [constexpr]
```

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

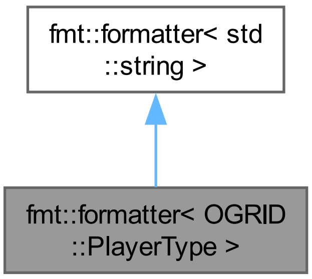
- Source/ogrid/GameLogicImplementation/GameConfiguration.cpp

7.13 fmt::formatter< OGRID::PlayerType > Struct Reference

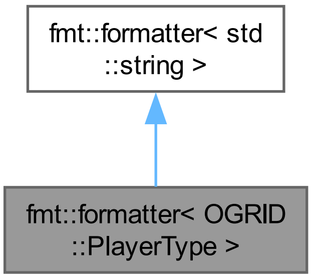
This is used to format a PlayerType enum into a string using fmt.

```
#include <ogrid_fmt.h>
```

Inheritance diagram for fmt::formatter< OGRID::PlayerType >:


fmt::formatter< std ::string >
fmt::formatter< OGRID ::PlayerType >

Collaboration diagram for fmt::formatter< OGRID::PlayerType >:


fmt::formatter< std ::string >
fmt::formatter< OGRID ::PlayerType >

Public Member Functions

- template<typename FormatContext >
  auto format (OGRID::PlayerType p, FormatContext &ctx)

7.13.1 Detailed Description

This is used to format a PlayerType enum into a string using fmt.

This should be used like this: fmt::format("{}", playerType);

Date

2023-12-06

7.13.2 Member Function Documentation

7.13.2.1 format()

```
template<typename FormatContext >
auto fmt::formatter< OGRID::PlayerType >::format (
            OGRID::PlayerType p,
            FormatContext & ctx )  [inline]
```

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

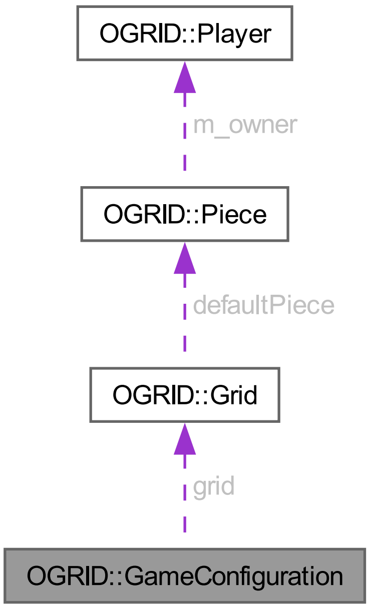
- Source/ogrid/ogrid_fmt.h

7.14 OGRID::GameConfiguration Struct Reference

The GameConfiguration class. Used to represent a game configuration.

```
#include <GameConfiguration.h>
```

Collaboration diagram for OGRID::GameConfiguration:



Public Attributes

- std::string gameName

  The name of the game.
- std::string gameDescription

  The description of the game.
- Grid $*$ grid = nullptr

  The grid of the game.
- size_t maxPlayers = 0

  The maximum number of players.
- std::vector$<$ Player $*$ $>$ players

  The players of the game.
- std::vector$<$ PlayerNameAndPtr $>$ playerPairs

  The player pairs of the game.

7.14.1 Detailed Description

The GameConfiguration class. Used to represent a game configuration.

The GameConfiguration class. It contains the name of the game, the description of the game, the grid of the game, the maximum number of players and the players of the game.

Date

2023-12-06

7.14.2 Member Data Documentation

7.14.2.1 gameDescription

```
std::string OGRID::GameConfiguration::gameDescription
```

The description of the game.

The description of the game. It is used to describe the game.

Date

2023-12-06

7.14.2.2 gameName

```
std::string OGRID::GameConfiguration::gameName
```

The name of the game.

The name of the game. It is used to identify the game.

Date

2023-12-06

7.14.2.3 grid

```
Grid* OGRID::GameConfiguration::grid = nullptr
```

The grid of the game.

The grid of the game. It is used to represent the game board.

Date

2023-12-06

7.14.2.4 maxPlayers

```
size_t OGRID::GameConfiguration::maxPlayers = 0
```

The maximum number of players.

The maximum number of players. It is used to limit the number of players.

Date

2023-12-06

7.14.2.5 playerPairs

```
std::vector<PlayerNameAndPtr> OGRID::GameConfiguration::playerPairs
```

The player pairs of the game.

The player pairs of the game. It is used to represent the player pairs.

Date

2023-12-06

7.14.2.6 players

```
std::vector<Player *> OGRID::GameConfiguration::players
```

The players of the game.

The players of the game. It is used to represent the players.

Date

2023-12-06

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

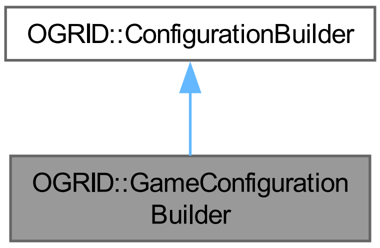
- Source/ogrid/GameLogicImplementation/GameConfiguration.h

7.15 OGRID::GameConfigurationBuilder Class Reference

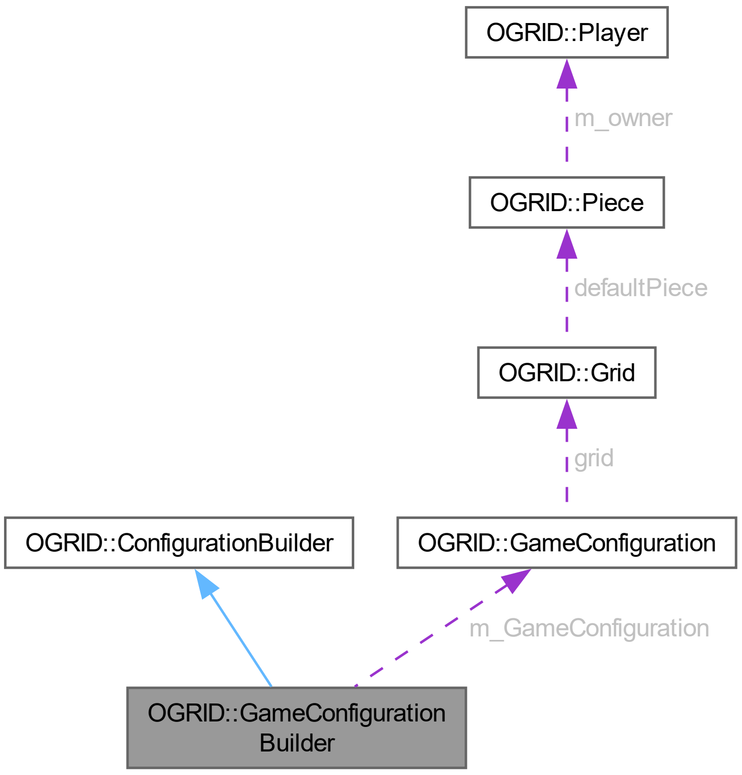
The GameConfigurationBuilder class. Used to build a GameConfiguration object.

```
#include <GameConfiguration.h>
```

Inheritance diagram for OGRID::GameConfigurationBuilder:



Collaboration diagram for OGRID::GameConfigurationBuilder:



Public Member Functions

- GameConfigurationBuilder ()=default

  Construct a new GameConfigurationBuilder object.
- ∼GameConfigurationBuilder () override=default

  Destroy the GameConfigurationBuilder object.
- ConfigurationBuilder & setGameName (const std::string &gameName) override

  Set the name of the game.
- ConfigurationBuilder & setGameDescription (const std::string &gameDescription) override

  Set the description of the game.
- ConfigurationBuilder & setGrid (unsigned char rows, unsigned char cols, Piece ∗defaultPiece=nullptr) override

  Set the grid of the game.
- ConfigurationBuilder & setMaxPlayers (size_t maxPlayers) override

  Set the maximum number of players.
- ConfigurationBuilder & addPlayer (Player ∗player) override

  Add a player to the game.
- GameConfiguration ∗ build () override

  Build the GameConfiguration object.

Public Member Functions inherited from OGRID::ConfigurationBuilder

- virtual ∼ConfigurationBuilder ()=default

  Destroy the ConfigurationBuilder object.

Private Attributes

- GameConfiguration m_GameConfiguration

  The GameConfiguration object.

7.15.1 Detailed Description

The GameConfigurationBuilder class. Used to build a GameConfiguration object.

It is used to build a GameConfiguration object.

Date

2023-12-06

See also

GameConfiguration

ConfigurationBuilder

7.15.2 Constructor & Destructor Documentation

7.15.2.1 GameConfigurationBuilder()

```
OGRID::GameConfigurationBuilder::GameConfigurationBuilder ( )  [default]
```

Construct a new GameConfigurationBuilder object.

Date

2023-12-06

7.15.2.2 ∼GameConfigurationBuilder()

```
OGRID::GameConfigurationBuilder::~GameConfigurationBuilder ( )  [override], [default]
```

Destroy the GameConfigurationBuilder object.

Date

2023-12-06

7.15.3 Member Function Documentation

7.15.3.1 addPlayer()

```
ConfigurationBuilder & OGRID::GameConfigurationBuilder::addPlayer (
            Player * player )  [override], [virtual]
```

Add a player to the game.

Date

2023-12-06

Parameters

player	The player to be added.

Returns

The GameConfigurationBuilder object.

Implements OGRID::ConfigurationBuilder.

7.15.3.2 build()

```
GameConfiguration * OGRID::GameConfigurationBuilder::build ( )  [override], [virtual]
```

Build the GameConfiguration object.

Date

2023-12-06

Returns

The GameConfiguration object.

Implements OGRID::ConfigurationBuilder.

7.15.3.3 setGameDescription()

```
ConfigurationBuilder & OGRID::GameConfigurationBuilder::setGameDescription (
             const std::string & gameDescription )  [override], [virtual]
```

Set the description of the game.

Date

2023-12-06

Parameters

gameDescription	The description of the game.

Returns

The GameConfigurationBuilder object.

Implements OGRID::ConfigurationBuilder.

7.15.3.4 setGameName()

```
ConfigurationBuilder & OGRID::GameConfigurationBuilder::setGameName (
             const std::string & gameName )  [override], [virtual]
```

Set the name of the game.

Date

2023-12-06

Parameters

gameName	The name of the game.

Returns

The GameConfigurationBuilder object.

Implements OGRID::ConfigurationBuilder.

7.15.3.5 setGrid()

```
ConfigurationBuilder & OGRID::GameConfigurationBuilder::setGrid (
            unsigned char rows,
            unsigned char cols,
            Piece * defaultPiece = nullptr )  [override], [virtual]
```

Set the grid of the game.

Date

2023-12-06

Parameters

rows	The number of rows of the grid.
cols	The number of columns of the grid.
defaultPiece	The default piece of the grid.

Returns

The GameConfigurationBuilder object.

Implements OGRID::ConfigurationBuilder.

7.15.3.6 setMaxPlayers()

```
ConfigurationBuilder & OGRID::GameConfigurationBuilder::setMaxPlayers (
            size_t maxPlayers )  [override], [virtual]
```

Set the maximum number of players.

Date

2023-12-06

Parameters

maxPlayers	The maximum number of players.

Returns

The GameConfigurationBuilder object.

Implements OGRID::ConfigurationBuilder.

7.15.4 Member Data Documentation

7.15.4.1 m_GameConfiguration

```
GameConfiguration OGRID::GameConfigurationBuilder::m_GameConfiguration  [private]
```

The GameConfiguration object.

The GameConfiguration object. It is used to store the GameConfiguration object.

Date

2023-12-06

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

- Source/ogrid/GameLogicImplementation/GameConfiguration.h
- Source/ogrid/GameLogicImplementation/GameConfiguration.cpp

7.16 Sandbox::GameInitializer Class Reference

Game initializer.

```
#include <GameInitializer.h>
```

Static Public Member Functions

- static void Start ()
  Starts the app and gives choice of games.

7.16.1 Detailed Description

Game initializer.

Date

2023-12-06

7.16.2 Member Function Documentation

7.16.2.1 Start()

```
void Sandbox::GameInitializer::Start ( )  [static]
```

Starts the app and gives choice of games.

Date

2023-12-06

See also

GameWindow

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

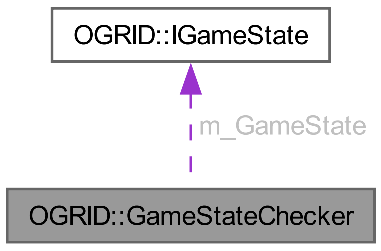
- Source/Sandbox/Core/GameInitializer.h
- Source/Sandbox/Core/GameInitializer.cpp

7.17 OGRID::GameStateChecker Class Reference

The GameStateChecker class. Used to check the state of the game.

```
#include <GameStateChecker.h>
```

Collaboration diagram for OGRID::GameStateChecker:



Public Member Functions

- GameStateChecker (IGameState ∗strategy)

  Construct a new GameStateChecker object.
- ∼GameStateChecker ()

  Destroy the GameStateChecker object.
- int CheckWin (Grid ∗grid) const

  Check the state of the game.
- bool IsDraw (Grid ∗grid) const

  Check if the game is a draw.
- bool IsColumnOccupied (Grid ∗grid, unsigned char colToCheck, unsigned char &rowToFill)

  Check if the game is over.
- unsigned char GetTopMostPiecePositionInColumn (Grid ∗grid, int col)

  Get the top most piece position in a column.

Private Attributes

- IGameState ∗ m_GameState

  The strategy to check the state of the game.

7.17.1 Detailed Description

The GameStateChecker class. Used to check the state of the game.

It contains the strategy to check the state of the game.

Date

2023-12-06

7.17.2 Constructor & Destructor Documentation

7.17.2.1 GameStateChecker()

```
OGRID::GameStateChecker::GameStateChecker (
            IGameState ∗ strategy )
```

Construct a new GameStateChecker object.

Construct a new GameStateChecker object using the strategy to check the state of the game.

Date

2023-12-06

7.17.2.2 ∼GameStateChecker()

```
OGRID::GameStateChecker::∼GameStateChecker ( )
```

Destroy the GameStateChecker object.

Date

2023-12-06

7.17.3 Member Function Documentation

7.17.3.1 CheckWin()

```
int OGRID::GameStateChecker::CheckWin (
            Grid * grid ) const
```

Check the state of the game.

Check the state of the game using the strategy.

Date

2023-12-06

7.17.3.2 GetTopMostPiecePositionInColumn()

```
unsigned char OGRID::GameStateChecker::GetTopMostPiecePositionInColumn (
            Grid * grid,
            int col )
```

Get the top most piece position in a column.

Get the top most piece position in a column using the strategy.

Date

2023-12-06

7.17.3.3 IsColumnOccupied()

```
bool OGRID::GameStateChecker::IsColumnOccupied (
            Grid * grid,
            unsigned char colToCheck,
            unsigned char & rowToFill )
```

Check if the game is over.

Check if the game is over using the strategy.

Date

2023-12-06

7.17.3.4 IsDraw()

```
bool OGRID::GameStateChecker::IsDraw (
            Grid * grid ) const
```

Check if the game is a draw.

Check if the game is a draw using the strategy.

Date

2023-12-06

7.17.4 Member Data Documentation

7.17.4.1 m_GameState

```
IGameState* OGRID::GameStateChecker::m_GameState  [private]
```

The strategy to check the state of the game.

It is used to check the state of the game.

Date

2023-12-06

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

- Source/ogrid/GameLogicImplementation/GameStateChecker.h
- Source/ogrid/GameLogicImplementation/GameStateChecker.cpp

7.18 OGRID::GameStateExtensions Class Reference

The GameStateExtensions class. Used to extend the GameStateChecker class.

```
#include <GameStateExtensions.h>
```

Public Member Functions

- bool CheckForRecurringStringInRow (Grid ∗grid, const std::string &pieceRepresentation, unsigned char dupeCount) const

  Check the exact amount of duplicate pieces in a row.
- bool CheckForRecurringPieceInRow (Grid ∗grid, const std::type_info &pieceType, unsigned char dupeCount) const

  Check the exact amount of duplicate pieces in a row.
- bool CheckForRecurringStringInCol (Grid ∗grid, const std::string &pieceRepresentation, unsigned char dupeCount) const

  Check the exact amount of duplicate pieces in a column.
- bool CheckForRecurringPieceInCol (Grid ∗grid, const std::type_info &pieceType, unsigned char dupeCount) const

  Check the exact amount of duplicate pieces in a column.
- bool CheckForRecurringStringInDiagonal (Grid ∗grid, const std::string &pieceRepresentation, unsigned char dupeCount) const

  Check the exact amount of duplicate pieces in a diagonal.
- bool CheckForRecurringPieceInDiagonal (Grid ∗grid, const std::type_info &pieceType, unsigned char dupe↩ Count) const

  Check the exact amount of duplicate pieces in a diagonal.
- bool CheckForRecurringStringInAntiDiagonal (Grid ∗grid, const std::string &pieceRepresentation, unsigned char dupeCount) const

  Check the exact amount of duplicate pieces in an anti-diagonal.
- bool CheckForRecurringPieceInAntiDiagonal (Grid ∗grid, const std::type_info &pieceType, unsigned char dupeCount) const

  Check the exact amount of duplicate pieces in an anti-diagonal.
- bool CheckIfAllSpotsFilled (Grid ∗grid) const

  Check if all the spots in the grid are filled.

7.18.1 Detailed Description

The GameStateExtensions class. Used to extend the GameStateChecker class.

It contains the extension methods for the GameStateChecker class.

Date

2023-12-06

7.18.2 Member Function Documentation

7.18.2.1 CheckForRecurringPieceInAntiDiagonal()

```
bool OGRID::GameStateExtensions::CheckForRecurringPieceInAntiDiagonal (
            Grid * grid,
            const std::type_info & pieceType,
            unsigned char dupeCount ) const
```

Check the exact amount of duplicate pieces in an anti-diagonal.

This should be used like this: CheckForRecurringPieceInAntiDiagonal(typeid(XPiece), 3))

Date

2023-12-06

Parameters

grid	The grid of the game.
pieceType	The type of the piece to be counted.
dupeCount	The exact number of duplicate pieces to be counted.

Returns

True if the exact amount of duplicate pieces are found, false otherwise.

7.18.2.2 CheckForRecurringPieceInCol()

```
bool OGRID::GameStateExtensions::CheckForRecurringPieceInCol (
            Grid * grid,
            const std::type_info & pieceType,
            unsigned char dupeCount ) const
```

Check the exact amount of duplicate pieces in a column.

This should be used like this: CheckForRecurringPieceInCol(typeid(XPiece), 3))

Date

2023-12-06

Parameters

grid	The grid of the game.
pieceType	The type of the piece to be counted.
dupeCount	The exact number of duplicate pieces to be counted.

Returns

True if the exact amount of duplicate pieces are found, false otherwise.

7.18.2.3 CheckForRecurringPieceInDiagonal()

```
bool OGRID::GameStateExtensions::CheckForRecurringPieceInDiagonal (
            Grid * grid,
            const std::type_info & pieceType,
            unsigned char dupeCount ) const
```

Check the exact amount of duplicate pieces in a diagonal.

This should be used like this: CheckForRecurringPieceInDiagonal(typeid(XPiece), 3))

Date

2023-12-06

Parameters

grid	The grid of the game.
pieceType	The type of the piece to be counted.
dupeCount	The exact number of duplicate pieces to be counted.

Returns

True if the exact amount of duplicate pieces are found, false otherwise.

7.18.2.4 CheckForRecurringPieceInRow()

```
bool OGRID::GameStateExtensions::CheckForRecurringPieceInRow (
             Grid * grid,
             const std::type_info & pieceType,
             unsigned char dupeCount ) const
```

Check the exact amount of duplicate pieces in a row.

This should be used like this: CheckForRecurringPieceInRow(typeid(XPiece), 3))

Date

2023-12-06

Parameters

grid	The grid of the game.
pieceType	The type of the piece to be counted.
dupeCount	The exact number of duplicate pieces to be counted.

Returns

True if the exact amount of duplicate pieces are found, false otherwise.

7.18.2.5 CheckForRecurringStringInAntiDiagonal()

```
bool OGRID::GameStateExtensions::CheckForRecurringStringInAntiDiagonal (
             Grid * grid,
             const std::string & pieceRepresentation,
             unsigned char dupeCount ) const
```

Check the exact amount of duplicate pieces in an anti-diagonal.

Date

2023-12-06

Parameters

grid	The grid of the game.
pieceRepresentation	The representation of the piece to be counted.
dupeCount	The exact number of duplicate pieces to be counted.

Returns

True if the exact amount of duplicate pieces are found, false otherwise.

7.18.2.6 CheckForRecurringStringInCol()

```
bool OGRID::GameStateExtensions::CheckForRecurringStringInCol (
            Grid * grid,
            const std::string & pieceRepresentation,
            unsigned char dupeCount ) const
```

Check the exact amount of duplicate pieces in a column.

Date

2023-12-06

Parameters

grid	The grid of the game.
pieceRepresentation	The representation of the piece to be counted.
dupeCount	The exact number of duplicate pieces to be counted.

Returns

True if the exact amount of duplicate pieces are found, false otherwise.

7.18.2.7 CheckForRecurringStringInDiagonal()

```
bool OGRID::GameStateExtensions::CheckForRecurringStringInDiagonal (
            Grid * grid,
            const std::string & pieceRepresentation,
            unsigned char dupeCount ) const
```

Check the exact amount of duplicate pieces in a diagonal.

Date

2023-12-06

Parameters

grid	The grid of the game.
pieceRepresentation	The representation of the piece to be counted.
dupeCount	The exact number of duplicate pieces to be counted.

Returns

True if the exact amount of duplicate pieces are found, false otherwise.

7.18.2.8 CheckForRecurringStringInRow()

```
bool OGRID::GameStateExtensions::CheckForRecurringStringInRow (
            Grid * grid,
            const std::string & pieceRepresentation,
            unsigned char dupeCount ) const
```

Check the exact amount of duplicate pieces in a row.

Date

2023-12-06

Parameters

grid	The grid of the game.
pieceRepresentation	The representation of the piece to be counted.
dupeCount	The exact number of duplicate pieces to be counted.

Returns

True if the exact amount of duplicate pieces are found, false otherwise.

7.18.2.9 CheckIfAllSpotsFilled()

```
bool OGRID::GameStateExtensions::CheckIfAllSpotsFilled (
            Grid * grid ) const
```

Check if all the spots in the grid are filled.

Date

2023-12-06

Parameters

grid	The grid of the game.

Returns

True if all the spots in the grid are filled, false otherwise.

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

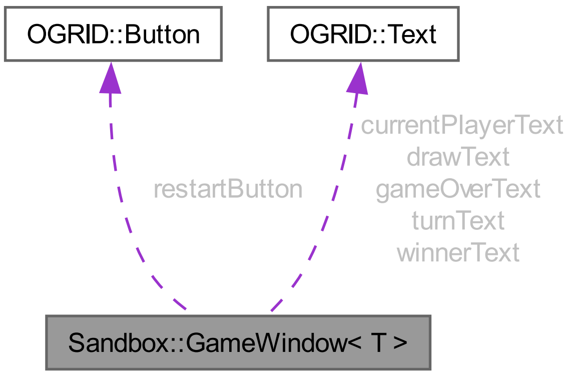
- Source/ogrid/GameLogicImplementation/GameStateExtensions.h
- Source/ogrid/GameLogicImplementation/GameStateExtensions.cpp

7.19 Sandbox::GameWindow< T > Class Template Reference

Game window.

`#include <GameWindow.h>`

Collaboration diagram for Sandbox::GameWindow< T >:



Public Member Functions

- GameWindow ()=default

  Construct a new GameWindow object.
- ∼GameWindow ()

  Destroy the GameWindow object.
- void Start ()

  Start the game.

Private Member Functions

- void Run ()

  Run the game.
- void PreRun ()

  Run checks before the game starts.
- void OnUpdate ()

  Update the game.
- void DrawGrid ()

  Draw the game.
- Vector2 GetCellFromMouse (Vector2 mousePosition)

  Get the cell from mouse position.
- void UpdateWindowDimensions ()

  Update the window dimensions.
- void MouseButtonPress ()

  Called when the mouse button is pressed.
- void InProgress ()

  Turns off logic while the game is in progress.
- void GameOver ()

  Turns on logic while the game is not in progress.

Private Attributes

- T ∗ m_Game

  Game logic.
- bool m_Running = false

  Is the game running.
- OGRID::Button ∗ restartButton

  Restart button.
- OGRID::Text ∗ gameOverText

  Game over text.
- OGRID::Text ∗ winnerText

  Winner text.
- OGRID::Text ∗ currentPlayerText

  Current player text.
- OGRID::Text ∗ turnText

  Turn text.
- OGRID::Text ∗ drawText

  Draw text.

7.19.1 Detailed Description

template<class T>
class Sandbox::GameWindow< T >

Game window.

Servers as a wrapper around raylib.

Date

2023-12-06

Template Parameters

T	Game logic, must inherit from IGame.

See also

https://www.raylib.com/
IGame

7.19.2 Constructor & Destructor Documentation

7.19.2.1 GameWindow()

```
template<class T >
Sandbox::GameWindow< T >::GameWindow ( )  [default]
```

Construct a new GameWindow object.

Date

2023-12-06

7.19.2.2 ∼GameWindow()

```
template<class T >
Sandbox::GameWindow< T >::∼GameWindow ( )
```

Destroy the GameWindow object.

Date

2023-12-06

7.19.3 Member Function Documentation

7.19.3.1 DrawGrid()

```
template<class T >
void Sandbox::GameWindow< T >::DrawGrid ( )  [private]
```

Draw the game.

Date

2023-12-06

7.19.3.2 GameOver()

```
template<class T >
void Sandbox::GameWindow< T >::GameOver ( )  [private]
```

Turns on logic while the game is not in progress.

Example: Restart button is enabled while the game is not in progress.

Date

2023-12-06

7.19.3.3 GetCellFromMouse()

```
template<class T >
Vector2 Sandbox::GameWindow< T >::GetCellFromMouse (
             Vector2 mousePosition )  [private]
```

Get the cell from mouse position.

Date

2023-12-06

Parameters

mousePosition	Mouse position.

Returns

Vector2 Cell position

7.19.3.4 InProgress()

```
template<class T >
void Sandbox::GameWindow< T >::InProgress ( )  [private]
```

Turns off logic while the game is in progress.

Example: Restart button is disabled while the game is in progress.

Date

2023-12-06

7.19.3.5 MouseButtonPress()

```
template<class T >
void Sandbox::GameWindow< T >::MouseButtonPress ( )  [private]
```

Called when the mouse button is pressed.

Date

2023-12-06

7.19.3.6 OnUpdate()

```
template<class T >
void Sandbox::GameWindow< T >::OnUpdate ( )  [private]
```

Update the game.

Date

2023-12-06

7.19.3.7 PreRun()

```
template<class T >
void Sandbox::GameWindow< T >::PreRun ( )  [private]
```

Run checks before the game starts.

Date

2023-12-06

7.19.3.8 Run()

```
template<class T >
void Sandbox::GameWindow< T >::Run ( )  [private]
```

Run the game.

Date

2023-12-06

7.19.3.9 Start()

```
template<class T >
void Sandbox::GameWindow< T >::Start ( )
```

Start the game.

Serves as an entry point for the game.

Date

2023-12-06

7.19.3.10 UpdateWindowDimensions()

```
template<class T >
void Sandbox::GameWindow< T >::UpdateWindowDimensions ( )  [private]
```

Update the window dimensions.

Date

2023-12-06

7.19.4 Member Data Documentation

7.19.4.1 currentPlayerText

```
template<class T >
OGRID::Text* Sandbox::GameWindow< T >::currentPlayerText  [private]
```

Current player text.

Date

2023-12-06

See also

Text

7.19.4.2 drawText

```
template<class T >
OGRID::Text* Sandbox::GameWindow< T >::drawText  [private]
```

Draw text.

Date

2023-12-06

See also

Text

7.19.4.3 gameOverText

```
template<class T >
OGRID::Text* Sandbox::GameWindow< T >::gameOverText  [private]
```

Game over text.

Date

2023-12-06

See also

Text

7.19.4.4 m_Game

```
template<class T >
T* Sandbox::GameWindow< T >::m_Game  [private]
```

Game logic.

Date

2023-12-06

See also

IGame

7.19.4.5 m_Running

```
template<class T >
bool Sandbox::GameWindow< T >::m_Running = false  [private]
```

Is the game running.

Date

2023-12-06

7.19.4.6 restartButton

```
template<class T >
OGRID::Button* Sandbox::GameWindow< T >::restartButton  [private]
```

Restart button.

Date

2023-12-06

See also

Button

7.19.4.7 turnText

```
template<class T >
OGRID::Text* Sandbox::GameWindow< T >::turnText [private]
```

Turn text.

Date

2023-12-06

See also

Text

7.19.4.8 winnerText

```
template<class T >
OGRID::Text* Sandbox::GameWindow< T >::winnerText [private]
```

Winner text.

Date

2023-12-06

See also

Text

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

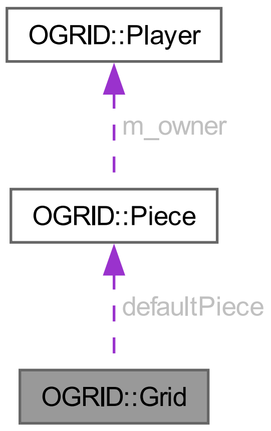
- Source/Sandbox/GUI/GameWindow.h
- Source/Sandbox/GUI/GameWindow.cpp

7.20 OGRID::Grid Class Reference

The Grid class represents a 2D grid of Cells.

```
#include <Grid.h>
```

Collaboration diagram for OGRID::Grid:



Public Member Functions

- Grid (unsigned char rows, unsigned char cols, Piece ∗defaultPiece=nullptr)

    Construct a new Grid object.
- ∼Grid ()

    Destroy the Grid object.
- unsigned char GetRows () const

    Get the number of rows in the grid.
- void SetRows (unsigned char rows)

    Set the number of rows in the grid.
- unsigned char GetCols () const

    Get the number of columns in the grid.
- void SetCols (unsigned char cols)

    Set the number of columns in the grid.
- const std::vector< std::vector< Cell ∗ > > & GetGrid () const

    Get the grid itself.
- void SetGrid (const std::vector< std::vector< Cell ∗ > > &newGrid)

    Set the grid itself.
- Piece ∗ GetDefaultPiece () const

    Get the default Piece for the grid.
- void SetDefaultPiece (Piece ∗defaultPiece)

    Set the default Piece for the grid.
- Piece ∗ GetPieceAt (unsigned char row, unsigned char col) const

    Access the Cell at the specified row and column within the grid and return a pointer to the Piece.
- void SetPieceAt (unsigned char row, unsigned char col, Piece ∗piece, bool force_null=false)

    Replace the specified Piece within the grid with the provided Piece.
- Cell ∗ GetCellAt (unsigned char row, unsigned char col) const

    Get a pointer to the specified Cell within the grid.
- void SetCellAt (unsigned char row, unsigned char col, Cell ∗cell, bool force_null=false)

Replace the specified Cell within the grid with the provided Cell.
- void SetCellAt (unsigned char row, unsigned char col, Piece ∗piece, bool force_null=false)

  Accesses the specified Cell within the grid and returns changes the Piece within the Cell with the provided PIece.
- std::pair< unsigned char, unsigned char > GetLastChangedChar () const

  Get the last changed element.
- std::vector< Cell ∗ > & operator[ ] (size_t index)

  Overload the [] operator to access the grid.
- const std::vector< Cell ∗ > & operator[ ] (size_t index) const

  Overload the [] operator to access the grid.
- const std::string GetGridSize () const

  Get the grid as a string.
- void ResetGrid ()

  Reset the grid.
- void ResetGridWithNewSize (unsigned char newRows, unsigned char newCols, Piece ∗defaultPiece=nullptr)

  Reset the grid with a new size.
- void ResetGridWithNewDefaultPiece (Piece ∗defaultPiece=nullptr)

  Reset the grid with a new default Piece.
- std::string GetGridAsString ()

  Get the grid as a string.

Private Attributes

- unsigned char rows

  The number of rows in the grid.
- unsigned char cols

  The number of columns in the grid.
- std::vector< std::vector< Cell ∗ > > grid

  The grid itself.
- Piece ∗ defaultPiece

  The default Piece for the grid.
- unsigned char lastChangedChar [2] = {0, 0}

  The last changed element.

7.20.1 Detailed Description

The Grid class represents a 2D grid of Cells.

It contains the number of rows and columns in the grid, as well as the grid itself. The Grid is a 2D array of Cell pointers. The Grid can be at most 255x255, which is more than enough for our purposes. The Grid class also contains a default Piece, which is used to reset the grid.

Date

2023-12-06

See also

Piece
Cell
Grid

7.20.2 Constructor & Destructor Documentation

7.20.2.1 Grid()

```
OGRID::Grid::Grid (
            unsigned char rows,
            unsigned char cols,
            Piece * defaultPiece = nullptr )
```

Construct a new Grid object.

This constructor will create a grid with the specified number of rows and columns. It will also set the default Piece to the specified Piece.

Date

2023-12-06

Parameters

rows	The number of rows in the grid.
cols	The number of columns in the grid.
defaultPiece	The default Piece for the grid. If this is nullptr.

See also

Piece

7.20.2.2 ∼Grid()

```
OGRID::Grid::∼Grid ( )
```

Destroy the Grid object.

This destructor will delete all the Cells in the grid.

Date

2023-12-06

7.20.3 Member Function Documentation

7.20.3.1 GetCellAt()

```
Cell * OGRID::Grid::GetCellAt (
            unsigned char row,
            unsigned char col ) const
```

Get a pointer to the specified Cell within the grid.

Date

2023-12-06

Parameters

row	The row of the cell.
col	The column of the cell.

Returns

The specified Cell within the grid.

7.20.3.2 GetCols()

```
unsigned char OGRID::Grid::GetCols ( ) const
```

Get the number of columns in the grid.

This is an unsigned char, which means the grid can be at most 255x255. This is more than enough for our purposes.

Date

2023-12-06

Returns

The number of columns in the grid.

7.20.3.3 GetDefaultPiece()

```
Piece * OGRID::Grid::GetDefaultPiece ( ) const
```

Get the default Piece for the grid.

This is the Piece that will be used to reset the grid.

Date

2023-12-06

Returns

The default Piece for the grid.

See also

Piece

7.20.3.4 GetGrid()

```
const std::vector< std::vector< Cell * > > & OGRID::Grid::GetGrid ( ) const
```

Get the grid itself.

This is a 2D array of Cell pointers.

Date

2023-12-06

Returns

The grid itself.

See also

Cell

7.20.3.5 GetGridAsString()

```
std::string OGRID::Grid::GetGridAsString ( )
```

Get the grid as a string.

This will return a string representation of the grid. If the Cell is a nullptr then "NULL" will be printed instead.

Date

2023-12-06

Returns

The grid as a string.

7.20.3.6 GetGridSize()

```
const std::string OGRID::Grid::GetGridSize ( ) const
```

Get the grid as a string.

This will return the size of the grid as a string. For example, a 3x3 grid will return "3x3".

Date

2023-12-06

Returns

The grid as a string.

7.20.3.7 GetLastChangedChar()

```
std::pair< unsigned char, unsigned char > OGRID::Grid::GetLastChangedChar ( ) const
```

Get the last changed element.

This is a pair of unsigned chars that stores the row and column of the last changed element.

Date

2023-12-06

Returns

The last changed element.

7.20.3.8 GetPieceAt()

```
Piece * OGRID::Grid::GetPieceAt (
             unsigned char row,
             unsigned char col ) const
```

Access the Cell at the specified row and column within the grid and return a pointer to the Piece.

Date

2023-12-06

Parameters

row	The row of the cell.
col	The column of the cell.

Returns

The specified Piece within the grid.

See also

Piece

7.20.3.9 GetRows()

```
unsigned char OGRID::Grid::GetRows ( ) const
```

Get the number of rows in the grid.

This is an unsigned char, which means the grid can be at most 255x255. This is more than enough for our purposes.

Date

2023-12-06

Returns

The number of rows in the grid.

7.20.3.10 operator[]() [1/2]

```
std::vector< Cell * > & OGRID::Grid::operator[] (
            size_t index )
```

Overload the [] operator to access the grid.

This is a 2D array of Cell pointers.

Date

2023-12-06

Returns

The grid itself.

See also

Cell

7.20.3.11 operator[]() [2/2]

```
const std::vector< Cell * > & OGRID::Grid::operator[] (
            size_t index ) const
```

Overload the [] operator to access the grid.

This is a 2D array of Cell pointers.

Date

2023-12-06

Returns

The grid itself.

See also

Cell

7.20.3.12 ResetGrid()

```
void OGRID::Grid::ResetGrid ( )
```

Reset the grid.

This will reset the grid to the default Piece. Deleting all the Cells (and Pieces if there are any) in the process.

Date

2023-12-06

See also

Cell

Piece

7.20.3.13 ResetGridWithNewDefaultPiece()

```
void OGRID::Grid::ResetGridWithNewDefaultPiece (
            Piece * defaultPiece = nullptr )
```

Reset the grid with a new default Piece.

This will reset the grid to the default Piece. Deleting all the Cells (and Pieces if there are any) in the process.

Date

2023-12-06

Parameters

defaultPiece	The new default Piece for the grid. Can be nullptr.

See also

Cell

Piece

7.20.3.14 ResetGridWithNewSize()

```
void OGRID::Grid::ResetGridWithNewSize (
            unsigned char newRows,
            unsigned char newCols,
            Piece * defaultPiece = nullptr )
```

Reset the grid with a new size.

This will reset the grid to the default Piece. Deleting all the Cells (and Pieces if there are any) in the process.

Date

2023-12-06

Parameters

newRows	The new number of rows in the grid.
newCols	The new number of columns in the grid.
defaultPiece	The new default Piece for the grid. Can be nullptr.

See also

Cell

Piece

7.20.3.15 SetCellAt() [1/2]

```
void OGRID::Grid::SetCellAt (
             unsigned char row,
             unsigned char col,
             Cell * cell,
             bool force_null = false )
```

Replace the specified Cell within the grid with the provided Cell.

Date

2023-12-06

Parameters

cell	The new Cell.
row	The row of the cell.
col	The column of the cell.
force_null	Allows to pass a nullptr.

See also

Piece

Cell

Grid

7.20.3.16 SetCellAt() [2/2]

```
void OGRID::Grid::SetCellAt (
             unsigned char row,
             unsigned char col,
             Piece * piece,
             bool force_null = false )
```

Accesses the specified Cell within the grid and returns changes the Piece within the Cell with the provided PIece.

Date

2023-12-06

Parameters

piece	The new Piece.
row	The row of the cell.
col	The column of the cell.
force_null	Allows to pass a nullptr.

See also

Piece

Cell

7.20.3.17 SetCols()

```
void OGRID::Grid::SetCols (
            unsigned char cols )
```

Set the number of columns in the grid.

This is an unsigned char, which means the grid can be at most 255x255. This is more than enough for our purposes.

Date

2023-12-06

Parameters

cols	The number of columns in the grid.

7.20.3.18 SetDefaultPiece()

```
void OGRID::Grid::SetDefaultPiece (
            Piece * defaultPiece )
```

Set the default Piece for the grid.

This is the Piece that will be used to reset the grid.

Date

2023-12-06

Parameters

defaultPiece	The new default Piece for the grid.

See also

Piece

ResetGrid()

7.20.3.19 SetGrid()

```
void OGRID::Grid::SetGrid (
             const std::vector< std::vector< Cell * > > & newGrid )
```

Set the grid itself.

This is a 2D array of Cell pointers.

Date

2023-12-06

Parameters

newGrid	The new grid.

See also

Cell

7.20.3.20 SetPieceAt()

```
void OGRID::Grid::SetPieceAt (
             unsigned char row,
             unsigned char col,
             Piece * piece,
             bool force_null = false )
```

Replace the specified Piece within the grid with the provided Piece.

Date

2023-12-06

Parameters

piece	The new Piece.
row	The row of the cell.
col	The column of the cell.
force_null	Allows to pass a nullptr.

See also

Piece
Cell
Grid

7.20.3.21 SetRows()

```
void OGRID::Grid::SetRows (
             unsigned char rows )
```

Set the number of rows in the grid.

This is an unsigned char, which means the grid can be at most 255x255. This is more than enough for our purposes.

Date

2023-12-06

Parameters

rows	The number of rows in the grid.

7.20.4 Member Data Documentation

7.20.4.1 cols

```
unsigned char OGRID::Grid::cols  [private]
```

The number of columns in the grid.

This is an unsigned char, which means the grid can be at most 255x255. This is more than enough for our purposes.

Date

2023-12-06

7.20.4.2 defaultPiece

```
Piece* OGRID::Grid::defaultPiece  [private]
```

The default Piece for the grid.

This is the Piece that will be used to reset the grid.

Date

2023-12-06

See also

Piece

7.20.4.3 grid

```
std::vector<std::vector<Cell *> > OGRID::Grid::grid  [private]
```

The grid itself.

This is a 2D array of Cell pointers.

Date

2023-12-06

See also

Cell

7.20.4.4 lastChangedChar

```
unsigned char OGRID::Grid::lastChangedChar[2] = {0, 0}  [private]
```

The last changed element.

This is a pair of unsigned chars that stores the row and column of the last changed element.

Date

2023-12-06

7.20.4.5 rows

```
unsigned char OGRID::Grid::rows  [private]
```

The number of rows in the grid.

This is an unsigned char, which means the grid can be at most 255x255. This is more than enough for our purposes.

Date

2023-12-06

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

- Source/ogrid/Grid/Grid.h
- Source/ogrid/Grid/Grid.cpp

7.21 GUIInfo Struct Reference

GUI info.

```
#include <GUIInfo.h>
```

Public Attributes

- int width

  Width of the window.
- int height

  Height of the window.
- std::string windowName

  Title of the window.
- int targetFPS

  Target FPS.
- float cellSize

  Cell size.
- float lineThickness

  Line thickness.
- float margin

  Margin.

7.21.1 Detailed Description

GUI info.

Contains information about the GUI. This is mostly used for the raylib window but it has core information for drawing buttons and texts.

Date

2023-12-06

See also

`https://www.raylib.com/`

Button

Text

Note

This provides the core information for the raylib window.

7.21.2 Member Data Documentation

7.21.2.1 cellSize

```
float GUIInfo::cellSize
```

Cell size.

Date

2023-12-06

7.21.2.2 height

```
int GUIInfo::height
```

Height of the window.

Date

2023-12-06

7.21.2.3 lineThickness

```
float GUIInfo::lineThickness
```

Line thickness.

Date

2023-12-06

7.21.2.4 margin

```
float GUIInfo::margin
```

Margin.

Date

2023-12-06

7.21.2.5 targetFPS

```
int GUIInfo::targetFPS
```

Target FPS.

Date

2023-12-06

7.21.2.6 width

```
int GUIInfo::width
```

Width of the window.

Date

2023-12-06

7.21.2.7 windowName

```
std::string GUIInfo::windowName
```

Title of the window.

Date

2023-12-06

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

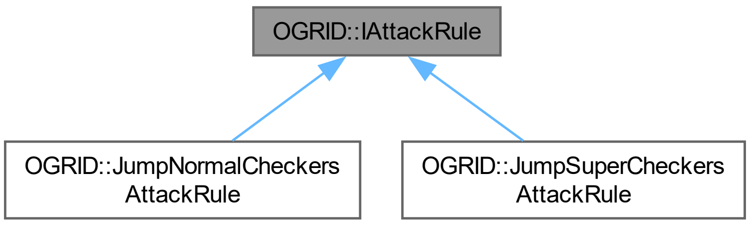
- Source/ogrid/GUI/GUIInfo.h

7.22 OGRID::IAttackRule Class Reference

The IGameState class. Used to check the state of the game.

```
#include <IAttackRule.h>
```

Inheritance diagram for OGRID::IAttackRule:


OGRID::IAttackRule
OGRID::JumpNormalCheckers AttackRule
OGRID::JumpSuperCheckers AttackRule

Public Member Functions

- virtual ∼IAttackRule ()
  Destroy the IGameState object.
- virtual bool IsValidAttack (Grid ∗grid, int x, int y, int x2, int y2, bool &canContinue) const =0
  Check if the attack is valid.

7.22.1 Detailed Description

The IGameState class. Used to check the state of the game.

It contains the strategy to check the state of the game.

Date

2023-12-06

7.22.2 Constructor & Destructor Documentation

7.22.2.1 ∼IAttackRule()

```
virtual OGRID::IAttackRule::∼IAttackRule ( )  [inline], [virtual]
```

Destroy the IGameState object.

Date

2023-12-06

7.22.3 Member Function Documentation

7.22.3.1 IsValidAttack()

```
virtual bool OGRID::IAttackRule::IsValidAttack (
              Grid * grid,
              int x,
              int y,
              int x2,
              int y2,
              bool & canContinue ) const  [pure virtual]
```

Check if the attack is valid.

Check if the attack is valid using the strategy.

Date

2023-12-06

Parameters

grid	The grid of the game.
x	The x coordinate of the piece.
y	The y coordinate of the piece.
x2	The x coordinate of the piece to be attacked.
y2	The y coordinate of the piece to be attacked.
canContinue	shows if there's another attack available after this one.

Returns

True if the attack is valid, false otherwise.

Implemented in OGRID::JumpNormalCheckersAttackRule, and OGRID::JumpSuperCheckersAttackRule.

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

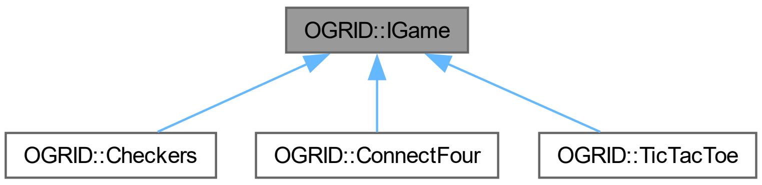
- Source/ogrid/GameLogicInterface/IAttackRule.h

7.23 OGRID::IGame Class Reference

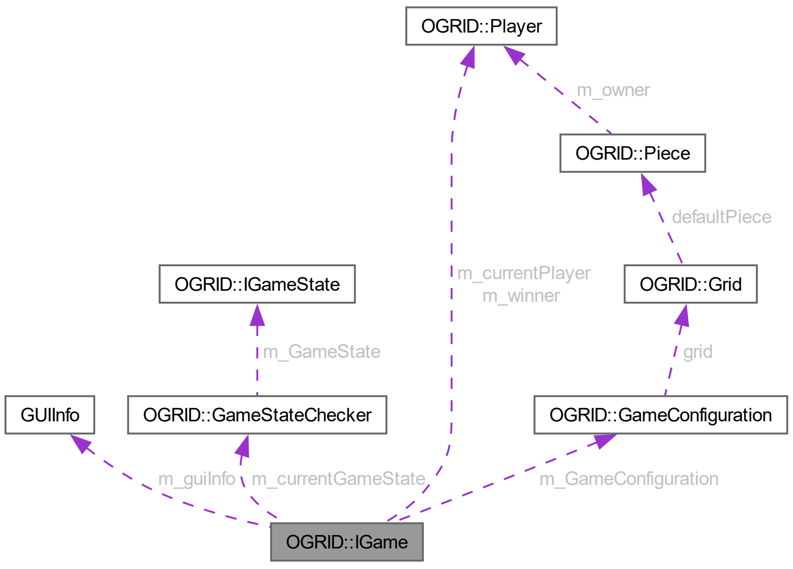
The IGame class. Used to represent a game.

```
#include <IGame.h>
```

Inheritance diagram for OGRID::IGame:



Collaboration diagram for OGRID::IGame:



Public Member Functions

- virtual bool TryMakeMove (unsigned char &row, unsigned char &col)=0

  Try making a move with the current player.
- virtual bool IsWinningCondition ()=0

  Check if the winning condition is met.
- virtual bool IsDrawCondition ()=0

Check if the draw condition is met.
- virtual void SetupPlayers ()=0

  Setup the players of the game.
- virtual void Initialize ()=0

  Setup the core of the game.
- virtual void OnGUIUpdateGrid ()=0

  Update the game's GUI.
- virtual void OnGUIUpdateGridHover (Vector2 cell)=0

  Update the game's GUI when hovering over a specific Cell.
- void SwapPlayerPositions ()

  Switches the current player to the next player.
- void ResetGrid ()

  Call the Grid object's ResetGrid() function.
- void ResetPlayers ()

  Reset the players of the game.
- void PrintPlayersTurnOrder () const

  Print the players of the game.
- void SetupGame ()

  Sets up the game.
- void ResetGame ()

  Resets the game.
- void StartGame ()

  Starts the game.
- void PrintPlayerMoves () const

  Prints the turn order.
- void MakeMove (unsigned char row, unsigned char col)

  Attempts to make a move.
- void Reset ()

  Resets the game.
- void SwitchPlayer ()

  Sets the current player to the next player.
- OGRID::GameOverType CheckGameOverState (OGRID::Grid ∗grid, unsigned char row, unsigned char col)

  Checks if the game is over.
- GameState GetGameState () const

  Get the state of the game.
- void SetGameState (GameState gameState)

  Set the state of the game.
- GameOverType GetGameOverType () const

  Get the game loop state of the game.
- Player ∗ GetWinner () const

  Get the winner of the game.
- GameConfiguration ∗ GetGameConfiguration () const

  Get the GameConfiguration object.
- void SetGameConfiguration (GameConfiguration ∗gameConfiguration)

  Set the GameConfiguration object.
- std::string GetGameName () const

  Get the name of the game.
- Grid ∗ GetGrid () const

  Get the Grid object of the game.
- std::vector< Player ∗ > GetPlayers () const

  Get the a Vector of the players of the game.

- void SetRandomizeTurnOrder (bool randomize)

    Toggle the randomization of the turn order.
- OGRID::PlayerNameAndPtr GetCurrentPlayer () const

    Get the current player of the game.
- void SetCurrentPlayer (OGRID::PlayerNameAndPtr player)

    Set the current player of the game.
- size_t GetCurrentTurn () const

    Get the current turn of the game.
- GameStateChecker ∗ GetGameStateChecker () const

    Get the current state of the game.
- void SetGameStateChecker (GameStateChecker ∗gameStateChecker)

    Set the current state of the game.
- std::vector< std::string > GetPlayerNames () const

    Get a Vector of the names of the players from GameConfiguration.
- std::vector< OGRID::Player ∗ > GetPlayerPtrs () const

    Get a Vector of the pointers of the players from GameConfiguration.
- OGRID::PlayerNameAndPtr GetPlayerPair (size_t at) const

    Get the name and pointer of the player from GameConfiguration.
- std::vector< OGRID::PlayerNameAndPtr > GetPlayerPairs () const

    Get a Vector of the names and pointers of the players from GameConfiguration.
- void SetPlayerPairs (const std::vector< OGRID::PlayerNameAndPtr > &players)

    Set the names and pointers of the players from GameConfiguration.
- GUIInfo GetGUIInfo () const

    Get the GUIInfo object.
- void SetGUIInfo (const GUIInfo &guiInfo)

    Set the GUIInfo object.

Public Attributes

- GUIInfo m_guiInfo
- bool m_randomizeTurnOrder = true

    To randomize the turn order of the players.

Protected Member Functions

- IGame ()=default

    The constructor of the IGame class.
- IGame (IGameState ∗gameStateStrategy, const std::vector< OGRID::PlayerNameAndPtr > &players)

    The constructor of the IGame class.
- ∼IGame ()

    The destructor of the IGame class.

Protected Attributes

- GameStateChecker ∗ m_currentGameState

  Holds the logic to check the state of the specific game.
- GameState m_gameState = GameState::NotStarted

  The state of the game.
- GameOverType m_gameOverType = GameOverType::None

  The game loop state of the game.
- Player ∗ m_winner = nullptr

  The winner of the game.
- Player ∗ m_currentPlayer = nullptr

  The current player of the game.
- size_t m_currentTurn = 0

  The current turn of the game.
- unsigned int m_totalTurns = 0

  Keeps the total number of turns.
- GameConfiguration ∗ m_GameConfiguration = nullptr

  The GameConfiguration object.

7.23.1 Detailed Description

The IGame class. Used to represent a game.

It contains the name of the game, the description of the game, the grid of the game, the maximum number of players and the players of the game.

Date

2023-12-06

7.23.2 Constructor & Destructor Documentation

7.23.2.1 IGame() [1/2]

```
OGRID::IGame::IGame ( )  [protected], [default]
```

The constructor of the IGame class.

Date

2023-12-06

7.23.2.2 IGame() [2/2]

```
OGRID::IGame::IGame (
            IGameState ∗ gameStateStrategy,
            const std::vector< OGRID::PlayerNameAndPtr > & players )  [protected]
```

The constructor of the IGame class.

Date

2023-12-06

Parameters

gameStateStrategy	The GameStateChecker object. This should be specific to each game type.
players	The players of the game.

7.23.2.3 ∼IGame()

```
OGRID::IGame::∼IGame ( )  [protected]
```

The destructor of the IGame class.

Date

2023-12-06

7.23.3 Member Function Documentation

7.23.3.1 CheckGameOverState()

```
OGRID::GameOverType OGRID::IGame::CheckGameOverState (
            OGRID::Grid * grid,
            unsigned char row,
            unsigned char col )
```

Checks if the game is over.

Date

2023-12-06

Returns

True if the game is over, false otherwise.

7.23.3.2 GetCurrentPlayer()

```
OGRID::PlayerNameAndPtr OGRID::IGame::GetCurrentPlayer ( ) const
```

Get the current player of the game.

Date

2023-12-06

Returns

The current player of the game.

7.23.3.3 GetCurrentTurn()

```
size_t OGRID::IGame::GetCurrentTurn ( ) const
```

Get the current turn of the game.

Date

2023-12-06

Returns

The current turn of the game.

7.23.3.4 GetGameConfiguration()

```
GameConfiguration * OGRID::IGame::GetGameConfiguration ( ) const
```

Get the GameConfiguration object.

Date

2023-12-06

Returns

The GameConfiguration object.

7.23.3.5 GetGameName()

```
std::string OGRID::IGame::GetGameName ( ) const
```

Get the name of the game.

Date

2023-12-06

Returns

The name of the game.

7.23.3.6 GetGameOverType()

```
GameOverType OGRID::IGame::GetGameOverType ( ) const
```

Get the game loop state of the game.

Date

2023-12-06

Returns

The game loop state of the game.

7.23.3.7 GetGameState()

```
GameState OGRID::IGame::GetGameState ( ) const
```

Get the state of the game.

Date

2023-12-06

Returns

The state of the game.

7.23.3.8 GetGameStateChecker()

```
GameStateChecker * OGRID::IGame::GetGameStateChecker ( ) const
```

Get the current state of the game.

Date

2023-12-06

Returns

The current state of the game.

7.23.3.9 GetGrid()

```
Grid * OGRID::IGame::GetGrid ( ) const
```

Get the Grid object of the game.

Date

2023-12-06

Returns

The Grid object of the game.

7.23.3.10 GetGUIInfo()

```
GUIInfo OGRID::IGame::GetGUIInfo ( ) const
```

Get the GUIInfo object.

Date

2023-12-06

Returns

The GUIInfo object.

7.23.3.11 GetPlayerNames()

```
std::vector< std::string > OGRID::IGame::GetPlayerNames ( ) const
```

Get a Vector of the names of the players from GameConfiguration.

Date

2023-12-06

Returns

A Vector of the names of the players from GameConfiguration.

7.23.3.12 GetPlayerPair()

```
OGRID::PlayerNameAndPtr OGRID::IGame::GetPlayerPair (
              size_t at ) const
```

Get the name and pointer of the player from GameConfiguration.

Date

2023-12-06

Parameters

at	The index of the player.

Returns

The name and pointer of the player from GameConfiguration.

7.23.3.13 GetPlayerPairs()

```
std::vector< OGRID::PlayerNameAndPtr > OGRID::IGame::GetPlayerPairs ( ) const
```

Get a Vector of the names and pointers of the players from GameConfiguration.

Date

2023-12-06

Returns

A Vector of the names and pointers of the players from GameConfiguration.

7.23.3.14 GetPlayerPtrs()

```
std::vector< OGRID::Player * > OGRID::IGame::GetPlayerPtrs ( ) const
```

Get a Vector of the pointers of the players from GameConfiguration.

Date

2023-12-06

Returns

A Vector of the pointers of the players from GameConfiguration.

7.23.3.15 GetPlayers()

```
std::vector< Player * > OGRID::IGame::GetPlayers ( ) const
```

Get the a Vector of the players of the game.

Date

2023-12-06

Returns

Get the a Vector of the players of the game.

7.23.3.16 GetWinner()

```
Player * OGRID::IGame::GetWinner ( ) const
```

Get the winner of the game.

Date

2023-12-06

7.23.3.17 Initialize()

```
virtual void OGRID::IGame::Initialize ( )  [pure virtual]
```

Setup the core of the game.

Date

2023-12-06

Implemented in OGRID::Checkers, OGRID::ConnectFour, and OGRID::TicTacToe.

7.23.3.18 IsDrawCondition()

```
virtual bool OGRID::IGame::IsDrawCondition ( )  [pure virtual]
```

Check if the draw condition is met.

Date

2023-12-06

Returns

True if the draw condition is met, false otherwise.

Implemented in OGRID::Checkers, OGRID::ConnectFour, and OGRID::TicTacToe.

7.23.3.19 IsWinningCondition()

```
virtual bool OGRID::IGame::IsWinningCondition ( )  [pure virtual]
```

Check if the winning condition is met.

Date

2023-12-06

Returns

True if the winning condition is met, false otherwise.

Implemented in OGRID::Checkers, OGRID::ConnectFour, and OGRID::TicTacToe.

7.23.3.20 MakeMove()

```
void OGRID::IGame::MakeMove (
             unsigned char row,
             unsigned char col )
```

Attempts to make a move.

Date

2023-12-06

7.23.3.21 OnGUIUpdateGrid()

```
virtual void OGRID::IGame::OnGUIUpdateGrid ( )  [pure virtual]
```

Update the game's GUI.

Date

2023-12-06

Implemented in OGRID::Checkers, OGRID::ConnectFour, and OGRID::TicTacToe.

7.23.3.22 OnGUIUpdateGridHover()

```
virtual void OGRID::IGame::OnGUIUpdateGridHover (
             Vector2 cell )  [pure virtual]
```

Update the game's GUI when hovering over a specific Cell.

Date

2023-12-06

Parameters

cell	The cell of the grid.

See also

Cell

Implemented in OGRID::Checkers, OGRID::ConnectFour, and OGRID::TicTacToe.

7.23.3.23 PrintPlayerMoves()

```
void OGRID::IGame::PrintPlayerMoves ( ) const
```

Prints the turn order.

Date

2023-12-06

7.23.3.24 PrintPlayersTurnOrder()

```
void OGRID::IGame::PrintPlayersTurnOrder ( ) const
```

Print the players of the game.

Date

2023-12-06

7.23.3.25 Reset()

```
void OGRID::IGame::Reset ( )
```

Resets the game.

7.23.3.26 ResetGame()

```
void OGRID::IGame::ResetGame ( )
```

Resets the game.

Date

2023-12-06

7.23.3.27 ResetGrid()

```
void OGRID::IGame::ResetGrid ( )
```

Call the Grid object's ResetGrid() function.

GameConfiguration must be set before calling this function.

Date

2023-12-06

7.23.3.28 ResetPlayers()

```
void OGRID::IGame::ResetPlayers ( )
```

Reset the players of the game.

Date

2023-12-06

7.23.3.29 SetCurrentPlayer()

```
void OGRID::IGame::SetCurrentPlayer (
             OGRID::PlayerNameAndPtr player )
```

Set the current player of the game.

Date

2023-12-06

Parameters

player	The current player of the game.

Note

This is solely for testing purposes.

7.23.3.30 SetGameConfiguration()

```
void OGRID::IGame::SetGameConfiguration (
            GameConfiguration * gameConfiguration )
```

Set the GameConfiguration object.

Date

2023-12-06

Parameters

gameConfiguration	The GameConfiguration object.

7.23.3.31 SetGameState()

```
void OGRID::IGame::SetGameState (
            GameState gameState )
```

Set the state of the game.

Date

2023-12-06

Parameters

gameState	The state of the game.

7.23.3.32 SetGameStateChecker()

```
void OGRID::IGame::SetGameStateChecker (
            GameStateChecker * gameStateChecker )
```

Set the current state of the game.

Date

2023-12-06

Parameters

gameStateChecker	The current state of the game.

7.23.3.33 SetGUIInfo()

```
void OGRID::IGame::SetGUIInfo (
             const GUIInfo & guiInfo )
```

Set the GUIInfo object.

Date

2023-12-06

Parameters

guiInfo	The GUIInfo object.

7.23.3.34 SetPlayerPairs()

```
void OGRID::IGame::SetPlayerPairs (
             const std::vector< OGRID::PlayerNameAndPtr > & players )
```

Set the names and pointers of the players from GameConfiguration.

Date

2023-12-06

Parameters

players	A Vector of the names and pointers of the players from GameConfiguration.

7.23.3.35 SetRandomizeTurnOrder()

```
void OGRID::IGame::SetRandomizeTurnOrder (
             bool randomize )
```

Toggle the randomization of the turn order.

Date

2023-12-06

Parameters

randomize	True to randomize the turn order, false otherwise.

7.23.3.36 SetupGame()

```
void OGRID::IGame::SetupGame ( )
```

Sets up the game.

Date

2023-12-06

7.23.3.37 SetupPlayers()

```
virtual void OGRID::IGame::SetupPlayers ( ) [pure virtual]
```

Setup the players of the game.

Date

2023-12-06

Implemented in OGRID::Checkers, OGRID::ConnectFour, and OGRID::TicTacToe.

7.23.3.38 StartGame()

```
void OGRID::IGame::StartGame ( )
```

Starts the game.

Date

2023-12-06

7.23.3.39 SwapPlayerPositions()

```
void OGRID::IGame::SwapPlayerPositions ( )
```

Switches the current player to the next player.

Date

2023-12-06

7.23.3.40 SwitchPlayer()

```
void OGRID::IGame::SwitchPlayer ( )
```

Sets the current player to the next player.

Date

2023-12-06

Note

This is solely for testing purposes.

7.23.3.41 TryMakeMove()

```
virtual bool OGRID::IGame::TryMakeMove (
            unsigned char & row,
            unsigned char & col )  [pure virtual]
```

Try making a move with the current player.

Date

2023-12-06

Parameters

row	The row of the grid.
col	The column of the grid.

Implemented in OGRID::Checkers, OGRID::ConnectFour, and OGRID::TicTacToe.

7.23.4 Member Data Documentation

7.23.4.1 m_currentGameState

```
GameStateChecker* OGRID::IGame::m_currentGameState  [protected]
```

Holds the logic to check the state of the specific game.

Example: Tic Tac Toe has different rules than Checkers.

Date

2023-12-06

7.23.4.2 m_currentPlayer

```
Player* OGRID::IGame::m_currentPlayer = nullptr  [protected]
```

The current player of the game.

This is the current player that is making a move, i.e. the current turn.

Date

2023-12-06

7.23.4.3 m_currentTurn

```
size_t OGRID::IGame::m_currentTurn = 0  [protected]
```

The current turn of the game.

This is the current turn of the game. It is used to keep track of the current turn, i.e. each Player has a team. Each team has a turn identifier. Example: Tic Tac Toe has 2 players. Player 1 is X and Player 2 is O. Player 1 has a turn identifier of 0 and Player 2 has a turn identifier of 1.

Date

2023-12-06

7.23.4.4 m_GameConfiguration

```
GameConfiguration* OGRID::IGame::m_GameConfiguration = nullptr  [protected]
```

The GameConfiguration object.

This stores the most basic information of the game, like the grid, the players, the max players, the name of the game etc.

Date

2023-12-06

7.23.4.5 m_gameOverType

```
GameOverType OGRID::IGame::m_gameOverType = GameOverType::None  [protected]
```

The game loop state of the game.

Date

2023-12-06

7.23.4.6 m_gameState

```
GameState OGRID::IGame::m_gameState = GameState::NotStarted  [protected]
```

The state of the game.

Date

2023-12-06

7.23.4.7 m_guiInfo

```
GUIInfo OGRID::IGame::m_guiInfo
```

brief Holds the information of the GUI.

This is specifically used for raylib.

Date

2023-12-06

7.23.4.8 m_randomizeTurnOrder

```
bool OGRID::IGame::m_randomizeTurnOrder = true
```

To randomize the turn order of the players.

Date

2023-12-06

7.23.4.9 m_totalTurns

```
unsigned int OGRID::IGame::m_totalTurns = 0  [protected]
```

Keeps the total number of turns.

Date

2023-12-06

Note

This is not used for anything yet.

7.23.4.10 m_winner

```
Player* OGRID::IGame::m_winner = nullptr  [protected]
```

The winner of the game.

Date

2023-12-06

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

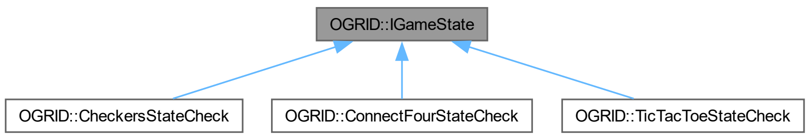
- Source/ogrid/GameLogicInterface/IGame.h
- Source/ogrid/GameLogicInterface/IGame.cpp

7.24 OGRID::IGameState Class Reference

The IGameState class. Used to check the state of the game.

```
#include <IGameState.h>
```

Inheritance diagram for OGRID::IGameState:


OGRID::IGameState
OGRID::CheckersStateCheck
OGRID::ConnectFourStateCheck
OGRID::TicTacToeStateCheck

Public Member Functions

- virtual ∼IGameState ()

    Destroy the IGameState object.
- virtual int CheckWin (Grid ∗grid) const =0

    Check if the game is over.
- virtual bool IsDraw (Grid ∗grid) const =0

    Check if the game is a draw.

7.24.1 Detailed Description

The IGameState class. Used to check the state of the game.

It contains the strategy to check the state of the game.

Date

2023-12-06

7.24.2 Constructor & Destructor Documentation

7.24.2.1 ∼IGameState()

```
virtual OGRID::IGameState::∼IGameState ( )  [inline], [virtual]
```

Destroy the IGameState object.

Date

2023-12-06

7.24.3 Member Function Documentation

7.24.3.1 CheckWin()

```
virtual int OGRID::IGameState::CheckWin (
             Grid * grid ) const  [pure virtual]
```

Check if the game is over.

Check if the game is over using the strategy.

Date

2023-12-06

Parameters

grid	The grid of the game.

Returns

True if the game is over, false otherwise.

Implemented in OGRID::CheckersStateCheck, OGRID::ConnectFourStateCheck, and OGRID::TicTacToeStateCheck.

7.24.3.2 IsDraw()

```
virtual bool OGRID::IGameState::IsDraw (
             Grid * grid ) const  [pure virtual]
```

Check if the game is a draw.

Check if the game is a draw using the strategy.

Date

2023-12-06

Parameters

grid	The grid of the game.

Returns

True if the game is a draw, false otherwise.

Implemented in OGRID::CheckersStateCheck, OGRID::ConnectFourStateCheck, and OGRID::TicTacToeStateCheck.

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

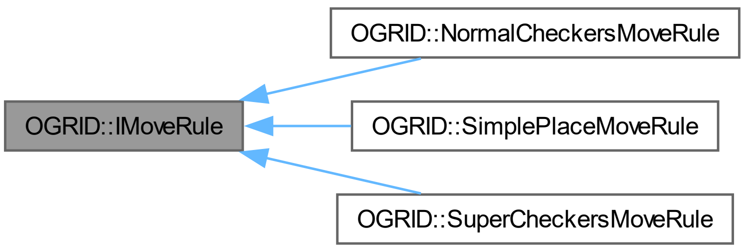
- Source/ogrid/GameLogicInterface/IGameState.h

7.25 OGRID::IMoveRule Class Reference

The IMoveRule class. Used to check if the move is valid.

```
#include <IMoveRule.h>
```

Inheritance diagram for OGRID::IMoveRule:


OGRID::NormalCheckersMoveRule
OGRID::IMoveRule
OGRID::SimplePlaceMoveRule
OGRID::SuperCheckersMoveRule

Public Member Functions

- virtual ∼IMoveRule ()

    Destroy the IMoveRule object.
- virtual bool IsValidMove (Grid ∗grid, int fromX, int fromY, int toX, int toY) const =0

    Check if the move is valid.

7.25.1 Detailed Description

The IMoveRule class. Used to check if the move is valid.

It contains the strategy to check if the move is valid.

Date

2023-12-06

7.25.2 Constructor & Destructor Documentation

7.25.2.1 ∼IMoveRule()

```
virtual OGRID::IMoveRule::∼IMoveRule ( )  [inline], [virtual]
```

Destroy the IMoveRule object.

Date

2023-12-06

7.25.3 Member Function Documentation

7.25.3.1 IsValidMove()

```
virtual bool OGRID::IMoveRule::IsValidMove (
             Grid * grid,
             int fromX,
             int fromY,
             int toX,
             int toY ) const  [pure virtual]
```

Check if the move is valid.

Check if the move is valid using the strategy.

Date

2023-12-06

Parameters

grid	The grid of the game.
fromX	The x coordinate of the piece.
fromY	The y coordinate of the piece.
toX	The x coordinate of the piece to be attacked.
toY	The y coordinate of the piece to be attacked.

Returns

True if the move is valid, false otherwise.

Implemented in OGRID::SimplePlaceMoveRule, OGRID::NormalCheckersMoveRule, and OGRID::SuperCheckersMoveRule.

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

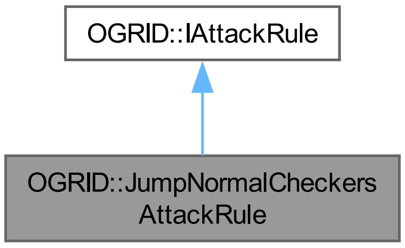
- Source/ogrid/GameLogicInterface/IMoveRule.h

7.26 OGRID::JumpNormalCheckersAttackRule Class Reference

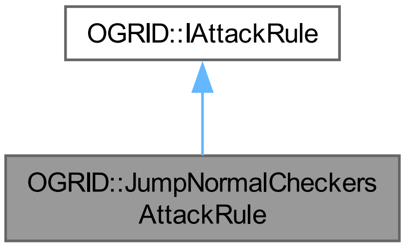
The PieceRules class. Used to represent the rules of a piece.

```
#include <PieceRules.h>
```

Inheritance diagram for OGRID::JumpNormalCheckersAttackRule:



Collaboration diagram for OGRID::JumpNormalCheckersAttackRule:



Public Member Functions

- bool IsValidAttack (Grid ∗grid, int fromX, int fromY, int toX, int toY, bool &canContinue) const override

  Check if the attack is valid.

Public Member Functions inherited from OGRID::IAttackRule

- virtual ∼IAttackRule ()

  Destroy the IGameState object.

7.26.1 Detailed Description

The PieceRules class. Used to represent the rules of a piece.

It contains the move rules of the piece and the attack rules of the piece.

Date

2023-12-06

7.26.2 Member Function Documentation

7.26.2.1 IsValidAttack()

```
bool OGRID::JumpNormalCheckersAttackRule::IsValidAttack (
            Grid * grid,
            int fromX,
            int fromY,
            int toX,
            int toY,
            bool & canContinue ) const  [override], [virtual]
```

Check if the attack is valid.

Check if the attack is valid. It is valid if the end cell is unoccupied.

Date

2023-12-06

Parameters

grid	The grid of the game.
fromX	The x coordinate of the start cell.
fromY	The y coordinate of the start cell.
toX	The x coordinate of the end cell.
toY	The y coordinate of the end cell.
canContinue	A boolean value that indicates if the attack can continue.

Returns

True if the attack is valid, false otherwise.

Note

This specifically checks the starting cell and the ending cell.

This is specifically used for Checkers.

Implements OGRID::IAttackRule.

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

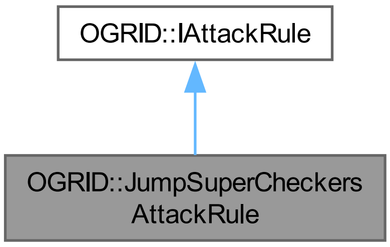
- Source/ogrid/GameLogicImplementation/PieceRules.h
- Source/ogrid/GameLogicImplementation/PieceRules.cpp

7.27 OGRID::JumpSuperCheckersAttackRule Class Reference

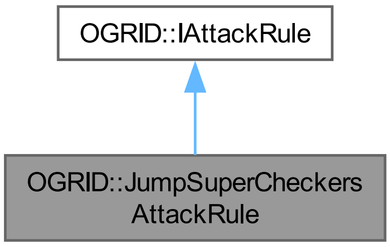
The PieceRules class. Used to represent the rules of a piece.

```
#include <PieceRules.h>
```

Inheritance diagram for OGRID::JumpSuperCheckersAttackRule:



Collaboration diagram for OGRID::JumpSuperCheckersAttackRule:



Public Member Functions

- bool IsValidAttack (Grid ∗grid, int fromX, int fromY, int toX, int toY, bool &canContinue) const override

  Check if the attack is valid.

Public Member Functions inherited from OGRID::IAttackRule

- virtual ∼IAttackRule ()

  Destroy the IGameState object.

7.27.1 Detailed Description

The PieceRules class. Used to represent the rules of a piece.

It contains the move rules of the piece and the attack rules of the piece.

Date

2023-12-06

7.27.2 Member Function Documentation

7.27.2.1 IsValidAttack()

```
bool OGRID::JumpSuperCheckersAttackRule::IsValidAttack (
            Grid * grid,
            int fromX,
            int fromY,
            int toX,
            int toY,
            bool & canContinue ) const  [override], [virtual]
```

Check if the attack is valid.

Check if the attack is valid. It is valid if the end cell is unoccupied.

Date

2023-12-06

Parameters

grid	The grid of the game.
fromX	The x coordinate of the start cell.
fromY	The y coordinate of the start cell.
toX	The x coordinate of the end cell.
toY	The y coordinate of the end cell.
canContinue	A boolean value that indicates if the attack can continue.

Returns

True if the attack is valid, false otherwise.

Note

This specifically checks the starting cell and the ending cell.

This is specifically used for Super Checkers.

This is used for the Super Checkers piece.

Implements OGRID::IAttackRule.

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

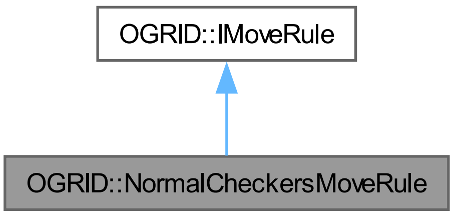
- Source/ogrid/GameLogicImplementation/PieceRules.h
- Source/ogrid/GameLogicImplementation/PieceRules.cpp

7.28 OGRID::NormalCheckersMoveRule Class Reference

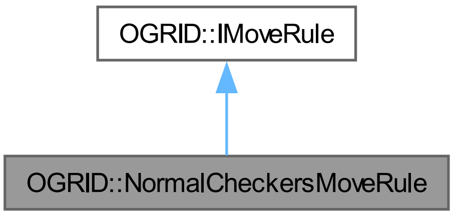
The PieceRules class. Used to represent the rules of a piece.

```
#include <PieceRules.h>
```

Inheritance diagram for OGRID::NormalCheckersMoveRule:



Collaboration diagram for OGRID::NormalCheckersMoveRule:



Public Member Functions

- bool IsValidMove (Grid ∗grid, int fromX, int fromY, int toX, int toY) const override

  Check if the move is valid.

Public Member Functions inherited from OGRID::IMoveRule

- virtual ∼IMoveRule ()

  Destroy the IMoveRule object.

7.28.1 Detailed Description

The PieceRules class. Used to represent the rules of a piece.

It contains the move rules of the piece and the attack rules of the piece.

Date

2023-12-06

7.28.2 Member Function Documentation

7.28.2.1 IsValidMove()

```
bool OGRID::NormalCheckersMoveRule::IsValidMove (
            Grid * grid,
            int fromX,
            int fromY,
            int toX,
            int toY ) const  [override], [virtual]
```

Check if the move is valid.

Check if the move is valid. It is valid if the end cell is unoccupied.

Date

2023-12-06

Parameters

grid	The grid of the game.
fromX	The x coordinate of the start cell.
fromY	The y coordinate of the start cell.
toX	The x coordinate of the end cell.
toY	The y coordinate of the end cell.

Returns

True if the move is valid, false otherwise.

Note

This specifically checks the starting cell and the ending cell.

This is specifically used for Checkers.

Implements OGRID::IMoveRule.

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

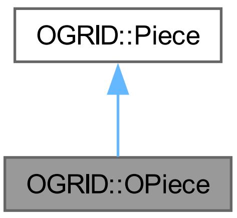
- Source/ogrid/GameLogicImplementation/PieceRules.h
- Source/ogrid/GameLogicImplementation/PieceRules.cpp

7.29 OGRID::OPiece Class Reference

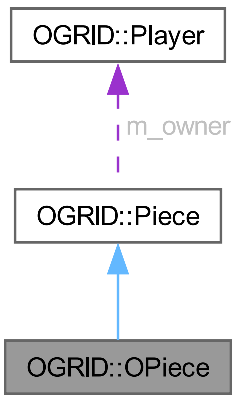
TicTacToe O piece.

```
#include <TicTacToePieces.h>
```

Inheritance diagram for OGRID::OPiece:



Collaboration diagram for OGRID::OPiece:



Public Member Functions

- OPiece (Player ∗player)

Public Member Functions inherited from OGRID::Piece

- Piece (std::string rep, Player ∗player)

  Construct a new Piece object.
- ∼Piece ()

  Destroy the Piece object.
- void AddMoveRule (IMoveRule ∗rule)

  Add a move rule to the piece.
- void AddAttackRule (IAttackRule ∗rule)

  Add an attack rule to the piece.
- const std::string & GetRepresentation () const

  Get the representation of the piece.
- const Player ∗ GetOwner () const

  Get the owner of the piece.
- void SetOwner (Player ∗player)

  Set the owner of the piece.
- bool isValidMove (Grid ∗grid, int fromX, int fromY, int toX, int toY) const

  Check if the move is valid.
- bool isValidAttack (Grid ∗grid, int fromX, int fromY, int toX, int toY, bool &canContinue) const

  Check if the attack is valid.

Additional Inherited Members

Protected Attributes inherited from OGRID::Piece

- std::string m_representation

  The representation of the piece.
- std::vector< IMoveRule ∗ > m_moveRules

  The move rules of the piece.
- std::vector< IAttackRule ∗ > m_attackRules

  The attack rules of the piece.
- Player ∗ m_owner

  The owner of the piece.

7.29.1 Detailed Description

TicTacToe O piece.

Date

2023-12-06

7.29.2 Constructor & Destructor Documentation

7.29.2.1 OPiece()

```
OGRID::OPiece::OPiece (
            Player * player )
```

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

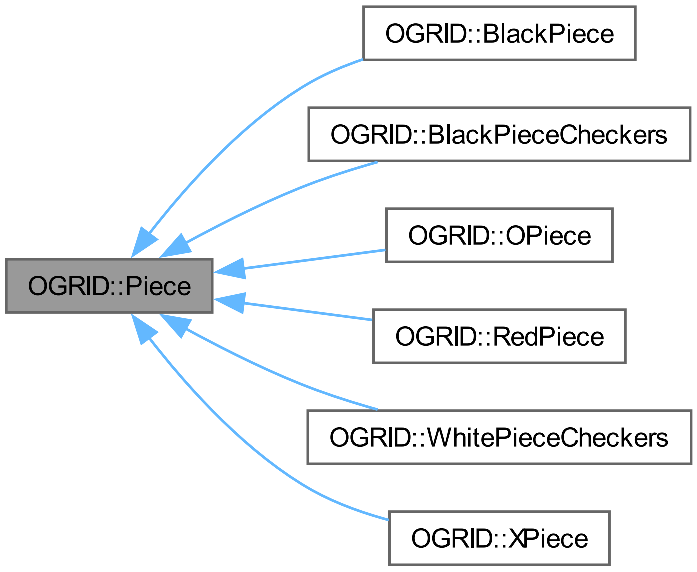
- Source/ogrid/Games/TicTacToe/TicTacToePieces.h
- Source/ogrid/Games/TicTacToe/TicTacToePieces.cpp

7.30 OGRID::Piece Class Reference

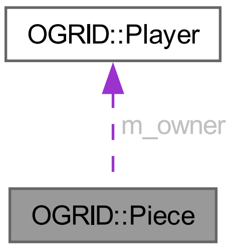
The Piece class. Used to represent a piece.

```
#include <Piece.h>
```

Inheritance diagram for OGRID::Piece:



Collaboration diagram for OGRID::Piece:



Public Member Functions

- Piece (std::string rep, Player ∗player)

Construct a new Piece object.
- ∼Piece ()

  Destroy the Piece object.
- void AddMoveRule (IMoveRule ∗rule)

  Add a move rule to the piece.
- void AddAttackRule (IAttackRule ∗rule)

  Add an attack rule to the piece.
- const std::string & GetRepresentation () const

  Get the representation of the piece.
- const Player ∗ GetOwner () const

  Get the owner of the piece.
- void SetOwner (Player ∗player)

  Set the owner of the piece.
- bool isValidMove (Grid ∗grid, int fromX, int fromY, int toX, int toY) const

  Check if the move is valid.
- bool isValidAttack (Grid ∗grid, int fromX, int fromY, int toX, int toY, bool &canContinue) const

  Check if the attack is valid.

Protected Attributes

- std::string m_representation

  The representation of the piece.
- std::vector< IMoveRule ∗ > m_moveRules

  The move rules of the piece.
- std::vector< IAttackRule ∗ > m_attackRules

  The attack rules of the piece.
- Player ∗ m_owner

  The owner of the piece.

7.30.1 Detailed Description

The Piece class. Used to represent a piece.

It contains the representation of the piece, the move rules of the piece, the attack rules of the piece and the owner of the piece.

Date

2023-12-06

7.30.2 Constructor & Destructor Documentation

7.30.2.1 Piece()

```
OGRID::Piece::Piece (
            std::string rep,
            Player * player )
```

Construct a new Piece object.

Construct a new Piece object using the representation of the piece and the owner of the piece.

Date

2023-12-06

Parameters

rep	The representation of the piece.
player	The owner of the piece.

7.30.2.2 ∼Piece()

```
OGRID::Piece::∼Piece ( )
```

Destroy the Piece object.

Date

2023-12-06

7.30.3 Member Function Documentation

7.30.3.1 AddAttackRule()

```
void OGRID::Piece::AddAttackRule (
             IAttackRule * rule )
```

Add an attack rule to the piece.

Add an attack rule to the piece.

Date

2023-12-06

Parameters

rule	The attack rule to add.

7.30.3.2 AddMoveRule()

```
void OGRID::Piece::AddMoveRule (
             IMoveRule * rule )
```

Add a move rule to the piece.

Add a move rule to the piece.

Date

2023-12-06

Parameters

rule	The move rule to add.

7.30.3.3 GetOwner()

```
const Player * OGRID::Piece::GetOwner ( ) const
```

Get the owner of the piece.

Date

2023-12-06

Returns

The owner of the piece.

7.30.3.4 GetRepresentation()

```
const std::string & OGRID::Piece::GetRepresentation ( ) const
```

Get the representation of the piece.

Date

2023-12-06

Returns

The representation of the piece.

7.30.3.5 isValidAttack()

```
bool OGRID::Piece::isValidAttack (
            Grid * grid,
            int fromX,
            int fromY,
            int toX,
            int toY,
            bool & canContinue ) const
```

Check if the attack is valid.

Check if the attack is valid using the attack rules.

Date

2023-12-06

Parameters

grid	The grid of the game.
fromX	The x coordinate of the start cell.
fromY	The y coordinate of the start cell.
toX	The x coordinate of the end cell.
toY	The y coordinate of the end cell.
canContinue	True if the attack can continue, false otherwise.

Returns

True if the attack is valid, false otherwise.

7.30.3.6 isValidMove()

```
bool OGRID::Piece::isValidMove (
            Grid * grid,
            int fromX,
            int fromY,
            int toX,
            int toY ) const
```

Check if the move is valid.

Check if the move is valid using the move rules.

Date

2023-12-06

Parameters

grid	The grid of the game.
fromX	The x coordinate of the start cell.
fromY	The y coordinate of the start cell.
toX	The x coordinate of the end cell.
toY	The y coordinate of the end cell.

Returns

True if the move is valid, false otherwise.

7.30.3.7 SetOwner()

```
void OGRID::Piece::SetOwner (
            Player * player )
```

Set the owner of the piece.

Date

2023-12-06

Parameters

player	The owner of the piece.

7.30.4 Member Data Documentation

7.30.4.1 m_attackRules

```
std::vector<IAttackRule *> OGRID::Piece::m_attackRules  [protected]
```

The attack rules of the piece.

It is used to check if the attack is valid.

7.30.4.2 m_moveRules

```
std::vector<IMoveRule *> OGRID::Piece::m_moveRules  [protected]
```

The move rules of the piece.

It is used to check if the move is valid.

7.30.4.3 m_owner

```
Player* OGRID::Piece::m_owner  [protected]
```

The owner of the piece.

It is used to identify the owner of the piece.

Date

2023-12-06

7.30.4.4 m_representation

```
std::string OGRID::Piece::m_representation  [protected]
```

The representation of the piece.

The representation of the piece. It is used to identify the piece.

Date

2023-12-06

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

- Source/ogrid/Player/Piece.h
- Source/ogrid/Player/Piece.cpp

7.31 OGRID::Player Class Reference

The Player class. Used to represent a player.

```
#include <Player.h>
```

Public Member Functions

- Player (std::string playerName="GenericName", PlayerType playerType=PlayerType::Human, int side=-1)

  Construct a new Player object.
- ∼Player ()

  Destroy the Player object.
- std::string GetPlayerName () const

  Get the name of the player.
- void SetPlayerName (std::string playerName)

  Set the name of the player.
- PlayerType GetPlayerType () const

  Get the type of the player.
- void SetPlayerType (PlayerType playerType)

  Set the type of the player.
- int GetSide () const

  Get the side to which the player belongs to.
- void SetSide (int side)

  Set the side to which the player belongs to.

Private Attributes

- std::string m_PlayerName

  The name of the player.
- PlayerType m_PlayerType

  The type of the player.
- int m_Side = -1

  The side to which the player belongs to.

7.31.1 Detailed Description

The Player class. Used to represent a player.

The Player class. It contains the name of the player, the type of the player and the side to which the player belongs to.

Date

2023-12-06

7.31.2 Constructor & Destructor Documentation

7.31.2.1 Player()

```
OGRID::Player::Player (
            std::string playerName = "GenericName",
            PlayerType playerType = PlayerType::Human,
            int side = -1 )
```

Construct a new Player object.

Construct a new Player object with the given name, type and side.

Date

2023-12-06

Parameters

playerName	The name of the player.
playerType	The type of the player.
side	The side to which the player belongs to. -1 is no side.

7.31.2.2 ∼Player()

```
OGRID::Player::∼Player ( )
```

Destroy the Player object.

Destroy the Player object.

Date

2023-12-06

7.31.3 Member Function Documentation

7.31.3.1 GetPlayerName()

```
std::string OGRID::Player::GetPlayerName ( ) const
```

Get the name of the player.

Get the name of the player.

Date

2023-12-06

Returns

The name of the player.

7.31.3.2 GetPlayerType()

```
PlayerType OGRID::Player::GetPlayerType ( ) const
```

Get the type of the player.

Get the type of the player.

Date

2023-12-06

Returns

The type of the player.

7.31.3.3 GetSide()

```
int OGRID::Player::GetSide ( ) const
```

Get the side to which the player belongs to.

Get the side to which the player belongs to.

Date

2023-12-06

Returns

The side to which the player belongs to.

7.31.3.4 SetPlayerName()

```
void OGRID::Player::SetPlayerName (
             std::string playerName )
```

Set the name of the player.

Set the name of the player.

Date

2023-12-06

Parameters

playerName	The name of the player.

7.31.3.5 SetPlayerType()

```
void OGRID::Player::SetPlayerType (
             PlayerType playerType )
```

Set the type of the player.

Set the type of the player.

Date

2023-12-06

Parameters

playerType	The type of the player.

7.31.3.6 SetSide()

```
void OGRID::Player::SetSide (
             int side )
```

Set the side to which the player belongs to.

Set the side to which the player belongs to.

Date

2023-12-06

Parameters

side	The side to which the player belongs to.

7.31.4 Member Data Documentation

7.31.4.1 m_PlayerName

```
std::string OGRID::Player::m_PlayerName  [private]
```

The name of the player.

The name of the player. It is used to identify the player.

Date

2023-12-06

7.31.4.2 m_PlayerType

```
PlayerType OGRID::Player::m_PlayerType  [private]
```

The type of the player.

The type of the player, either Human or AI. At the moment, the AI is not implemented.

Date

2023-12-06

7.31.4.3 m_Side

```
int OGRID::Player::m_Side = -1  [private]
```

The side to which the player belongs to.

The side to which the player belongs to. -1 is no side.

Date

2023-12-06

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

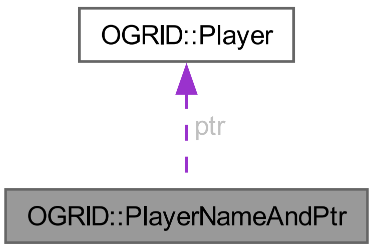
- Source/ogrid/Player/Player.h
- Source/ogrid/Player/Player.cpp

7.32 OGRID::PlayerNameAndPtr Struct Reference

Pair of player name and pointer.

```
#include <GameConfiguration.h>
```

Collaboration diagram for OGRID::PlayerNameAndPtr:



Public Attributes

- std::string name

  The name of the player.
- Player ∗ ptr

  The pointer to the player.

7.32.1 Detailed Description

Pair of player name and pointer.

Used to store the player name and pointer in the GameConfiguration class.

Date

2023-12-06

7.32.2 Member Data Documentation

7.32.2.1 name

```
std::string OGRID::PlayerNameAndPtr::name
```

The name of the player.

The name of the player. It is used to identify the player.

Date

2023-12-06

7.32.2.2 ptr

```
Player* OGRID::PlayerNameAndPtr::ptr
```

The pointer to the player.

The pointer to the player. It is used to access the player.

Date

2023-12-06

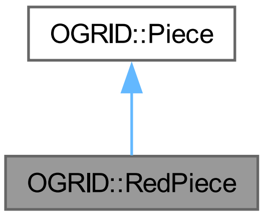
The documentation for this struct was generated from the following file:

- Source/ogrid/GameLogicImplementation/GameConfiguration.h

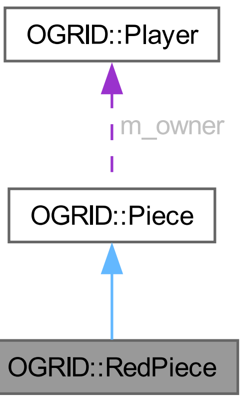
7.33 OGRID::RedPiece Class Reference

```
#include <ConnectFourPieces.h>
```

Inheritance diagram for OGRID::RedPiece:



Collaboration diagram for OGRID::RedPiece:



Public Member Functions

- RedPiece (Player ∗player)

Public Member Functions inherited from OGRID::Piece

- Piece (std::string rep, Player ∗player)

    Construct a new Piece object.
- ∼Piece ()

    Destroy the Piece object.

- void AddMoveRule (IMoveRule ∗rule)

  Add a move rule to the piece.
- void AddAttackRule (IAttackRule ∗rule)

  Add an attack rule to the piece.
- const std::string & GetRepresentation () const

  Get the representation of the piece.
- const Player ∗ GetOwner () const

  Get the owner of the piece.
- void SetOwner (Player ∗player)

  Set the owner of the piece.
- bool isValidMove (Grid ∗grid, int fromX, int fromY, int toX, int toY) const

  Check if the move is valid.
- bool isValidAttack (Grid ∗grid, int fromX, int fromY, int toX, int toY, bool &canContinue) const

  Check if the attack is valid.

Additional Inherited Members

Protected Attributes inherited from OGRID::Piece

- std::string m_representation

  The representation of the piece.
- std::vector< IMoveRule ∗ > m_moveRules

  The move rules of the piece.
- std::vector< IAttackRule ∗ > m_attackRules

  The attack rules of the piece.
- Player ∗ m_owner

  The owner of the piece.

7.33.1 Constructor & Destructor Documentation

7.33.1.1 RedPiece()

```
OGRID::RedPiece::RedPiece (
            Player * player )
```

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

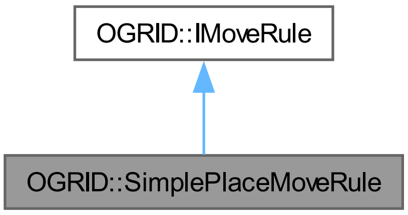
- Source/ogrid/Games/ConnectFour/ConnectFourPieces.h
- Source/ogrid/Games/ConnectFour/ConnectFourPieces.cpp

7.34 OGRID::SimplePlaceMoveRule Class Reference

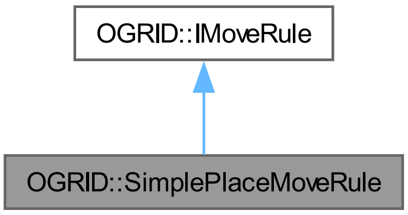
The PieceRules class. Used to represent the rules of a piece.

```
#include <PieceRules.h>
```

Inheritance diagram for OGRID::SimplePlaceMoveRule:



Collaboration diagram for OGRID::SimplePlaceMoveRule:



Public Member Functions

- bool IsValidMove (Grid ∗grid, int fromX, int fromY, int toX, int toY) const override

    Check if the move is valid.

Public Member Functions inherited from OGRID::IMoveRule

- virtual ∼IMoveRule ()

    Destroy the IMoveRule object.

7.34.1 Detailed Description

The PieceRules class. Used to represent the rules of a piece.

It contains the move rules of the piece and the attack rules of the piece.

Date

2023-12-06

7.34.2 Member Function Documentation

7.34.2.1 IsValidMove()

```
bool OGRID::SimplePlaceMoveRule::IsValidMove (
            Grid * grid,
            int fromX,
            int fromY,
            int toX,
            int toY ) const  [override], [virtual]
```

Check if the move is valid.

Check if the move is valid. It is valid if the end cell is unoccupied.

Date

2023-12-06

Parameters

grid	The grid of the game.
fromX	The x coordinate of the start cell.
fromY	The y coordinate of the start cell.
toX	The x coordinate of the end cell.
toY	The y coordinate of the end cell.

Returns

True if the move is valid, false otherwise.

Note

This is a simple move rule. It is used for games like tic tac toe.

Implements OGRID::IMoveRule.

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

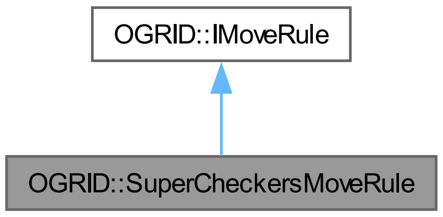
- Source/ogrid/GameLogicImplementation/PieceRules.h
- Source/ogrid/GameLogicImplementation/PieceRules.cpp

7.35 OGRID::SuperCheckersMoveRule Class Reference

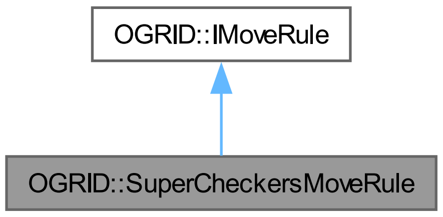
The PieceRules class. Used to represent the rules of a piece.

```
#include <PieceRules.h>
```

Inheritance diagram for OGRID::SuperCheckersMoveRule:



Collaboration diagram for OGRID::SuperCheckersMoveRule:



Public Member Functions

- bool IsValidMove (Grid ∗grid, int fromX, int fromY, int toX, int toY) const override

    Check if the move is valid.

Public Member Functions inherited from OGRID::IMoveRule

- virtual ∼IMoveRule ()

    Destroy the IMoveRule object.

7.35.1 Detailed Description

The PieceRules class. Used to represent the rules of a piece.

It contains the move rules of the piece and the attack rules of the piece.

Date

2023-12-06

7.35.2 Member Function Documentation

7.35.2.1 IsValidMove()

```
bool OGRID::SuperCheckersMoveRule::IsValidMove (
            Grid * grid,
            int fromX,
            int fromY,
            int toX,
            int toY ) const  [override], [virtual]
```

Check if the move is valid.

Check if the move is valid. It is valid if the end cell is unoccupied.

Date

2023-12-06

Parameters

grid	The grid of the game.
fromX	The x coordinate of the start cell.
fromY	The y coordinate of the start cell.
toX	The x coordinate of the end cell.
toY	The y coordinate of the end cell.

Returns

True if the move is valid, false otherwise.

Note

This specifically checks the starting cell and the ending cell.

This is specifically used for Super Checkers.

This is used for the Super Checkers piece.

Implements OGRID::IMoveRule.

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

- Source/ogrid/GameLogicImplementation/PieceRules.h
- Source/ogrid/GameLogicImplementation/PieceRules.cpp

7.36 OGRID::Text Struct Reference

Text.

```
#include <Text.h>
```

Public Member Functions

- Text (std::string text, int fontSize, int x, int y, Color color, Justify justify=Justify::NONE, int screenWidth=0, int screenHeight=0)

  Construct a new Text object.
- void Draw () const

  Draw the text.
- void SetText (std::string text)

  Set the text.
- void SetScreenSize (int width, int height)

  Set the screen size.
- void SetJustification (Justify newJustify)

  Set the justification.

Public Attributes

- std::string text

  Text to draw.
- int fontSize

  Font size.
- int x

  X position.
- int y

  Y position.
- int screenWidth

  Screen width.
- int screenHeight

  Screen height.
- Color color

  Color of the text.
- Justify justify

  Justification of the text.

7.36.1 Detailed Description

Text.

This is a warpper around raylib's DrawTextEx.

Date

2023-12-06

See also

https://www.raylib.com/

7.36.2 Constructor & Destructor Documentation

7.36.2.1 Text()

```
OGRID::Text::Text (
             std::string text,
             int fontSize,
             int x,
             int y,
             Color color,
             Justify justify = Justify::NONE,
             int screenWidth = 0,
             int screenHeight = 0 )  [inline]
```

Construct a new Text object.

Date

2023-12-06

Parameters

text	Text to draw.
fontSize	Font size.
x	X position.
y	Y position.
color	Color of the text.
justify	Justification of the text.
screenWidth	Screen width.
screenHeight	Screen height.

7.36.3 Member Function Documentation

7.36.3.1 Draw()

```
void OGRID::Text::Draw ( ) const  [inline]
```

Draw the text.

Date

2023-12-06

7.36.3.2 SetJustification()

```
void OGRID::Text::SetJustification (
             Justify newJustify )  [inline]
```

Set the justification.

Date

2023-12-06

Parameters

newJustify	Justification of the text.

7.36.3.3 SetScreenSize()

```
void OGRID::Text::SetScreenSize (
             int width,
             int height )  [inline]
```

Set the screen size.

Date

2023-12-06

Parameters

width	Screen width.
height	Screen height.

7.36.3.4 SetText()

```
void OGRID::Text::SetText (
             std::string text )  [inline]
```

Set the text.

Date

2023-12-06

Parameters

text	Text to draw.

7.36.4 Member Data Documentation

7.36.4.1 color

```
Color OGRID::Text::color
```

Color of the text.

Date

2023-12-06

7.36.4.2 fontSize

`int OGRID::Text::fontSize`

Font size.

Date

2023-12-06

7.36.4.3 justify

`Justify OGRID::Text::justify`

Justification of the text.

Date

2023-12-06

7.36.4.4 screenHeight

`int OGRID::Text::screenHeight`

Screen height.

Date

2023-12-06

7.36.4.5 screenWidth

`int OGRID::Text::screenWidth`

Screen width.

Date

2023-12-06

7.36.4.6 text

`std::string OGRID::Text::text`

Text to draw.

Date

2023-12-06

7.36.4.7 x

```
int OGRID::Text::x
```

X position.

Date

2023-12-06

7.36.4.8 y

```
int OGRID::Text::y
```

Y position.

Date

2023-12-06

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

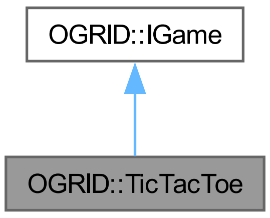
- Source/ogrid/GUI/Text.h

7.37 OGRID::TicTacToe Class Reference

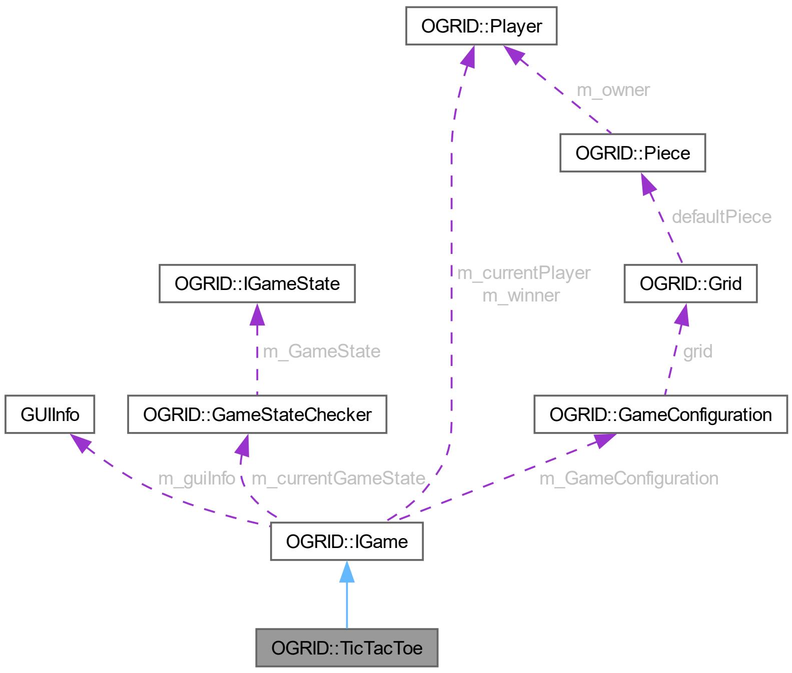
TicTacToe game logic.

```
#include <TicTacToe.h>
```

Inheritance diagram for OGRID::TicTacToe:



Collaboration diagram for OGRID::TicTacToe:



Public Member Functions

- TicTacToe ()=default

  Construct a new TicTacToe object.
- ∼TicTacToe ()=default

  Destroy the TicTacToe object.
- bool TryMakeMove (unsigned char &row, unsigned char &col) override

  Try to make a move.
- bool IsWinningCondition () override

  Check if the game has a winning condition.
- bool IsDrawCondition () override

  Check if the game has a draw condition.
- void SetupPlayers () override

  Setup the players.
- void Initialize () override

  Initialize the game.
- void OnGUIUpdateGrid () override

  Update the grid on the GUI.
- void OnGUIUpdateGridHover (Vector2 cell) override

  Update the grid on the GUI when hovering.

Public Member Functions inherited from OGRID::IGame

- void SwapPlayerPositions ()

  Switches the current player to the next player.
- void ResetGrid ()

  Call the Grid object's ResetGrid() function.
- void ResetPlayers ()

  Reset the players of the game.
- void PrintPlayersTurnOrder () const

  Print the players of the game.
- void SetupGame ()

  Sets up the game.
- void ResetGame ()

  Resets the game.
- void StartGame ()

  Starts the game.
- void PrintPlayerMoves () const

  Prints the turn order.
- void MakeMove (unsigned char row, unsigned char col)

  Attempts to make a move.
- void Reset ()

  Resets the game.
- void SwitchPlayer ()

  Sets the current player to the next player.
- OGRID::GameOverType CheckGameOverState (OGRID::Grid ∗grid, unsigned char row, unsigned char col)

  Checks if the game is over.
- GameState GetGameState () const

  Get the state of the game.
- void SetGameState (GameState gameState)

  Set the state of the game.
- GameOverType GetGameOverType () const

  Get the game loop state of the game.
- Player ∗ GetWinner () const

  Get the winner of the game.
- GameConfiguration ∗ GetGameConfiguration () const

  Get the GameConfiguration object.
- void SetGameConfiguration (GameConfiguration ∗gameConfiguration)

  Set the GameConfiguration object.
- std::string GetGameName () const

  Get the name of the game.
- Grid ∗ GetGrid () const

  Get the Grid object of the game.
- std::vector< Player ∗ > GetPlayers () const

  Get the a Vector of the players of the game.
- void SetRandomizeTurnOrder (bool randomize)

  Toggle the randomization of the turn order.
- OGRID::PlayerNameAndPtr GetCurrentPlayer () const

  Get the current player of the game.
- void SetCurrentPlayer (OGRID::PlayerNameAndPtr player)

  Set the current player of the game.
- size_t GetCurrentTurn () const

Get the current turn of the game.
- GameStateChecker ∗ GetGameStateChecker () const

  Get the current state of the game.
- void SetGameStateChecker (GameStateChecker ∗gameStateChecker)

  Set the current state of the game.
- std::vector< std::string > GetPlayerNames () const

  Get a Vector of the names of the players from GameConfiguration.
- std::vector< OGRID::Player ∗ > GetPlayerPtrs () const

  Get a Vector of the pointers of the players from GameConfiguration.
- OGRID::PlayerNameAndPtr GetPlayerPair (size_t at) const

  Get the name and pointer of the player from GameConfiguration.
- std::vector< OGRID::PlayerNameAndPtr > GetPlayerPairs () const

  Get a Vector of the names and pointers of the players from GameConfiguration.
- void SetPlayerPairs (const std::vector< OGRID::PlayerNameAndPtr > &players)

  Set the names and pointers of the players from GameConfiguration.
- GUIInfo GetGUIInfo () const

  Get the GUIInfo object.
- void SetGUIInfo (const GUIInfo &guiInfo)

  Set the GUIInfo object.

Private Member Functions

- void DrawX (int row, int col)

  Draw an X on the grid.
- void DrawO (int row, int col)

  Draw an O on the grid.

Additional Inherited Members

Public Attributes inherited from OGRID::IGame

- GUIInfo m_guiInfo
- bool m_randomizeTurnOrder = true

  To randomize the turn order of the players.

Protected Member Functions inherited from OGRID::IGame

- IGame ()=default

  The constructor of the IGame class.
- IGame (IGameState ∗gameStateStrategy, const std::vector< OGRID::PlayerNameAndPtr > &players)

  The constructor of the IGame class.
- ∼IGame ()

  The destructor of the IGame class.

Protected Attributes inherited from OGRID::IGame

- GameStateChecker ∗ m_currentGameState

  Holds the logic to check the state of the specific game.
- GameState m_gameState = GameState::NotStarted

  The state of the game.
- GameOverType m_gameOverType = GameOverType::None

  The game loop state of the game.
- Player ∗ m_winner = nullptr

  The winner of the game.
- Player ∗ m_currentPlayer = nullptr

  The current player of the game.
- size_t m_currentTurn = 0

  The current turn of the game.
- unsigned int m_totalTurns = 0

  Keeps the total number of turns.
- GameConfiguration ∗ m_GameConfiguration = nullptr

  The GameConfiguration object.

7.37.1 Detailed Description

TicTacToe game logic.

Date

2023-12-06

7.37.2 Constructor & Destructor Documentation

7.37.2.1 TicTacToe()

```
OGRID::TicTacToe::TicTacToe ( )  [default]
```

Construct a new TicTacToe object.

Date

2023-12-06

7.37.2.2 ∼TicTacToe()

```
OGRID::TicTacToe::∼TicTacToe ( )  [default]
```

Destroy the TicTacToe object.

Date

2023-12-06

7.37.3 Member Function Documentation

7.37.3.1 DrawO()

```
void OGRID::TicTacToe::DrawO (
             int row,
             int col )  [private]
```

Draw an O on the grid.

Date

2023-12-06

Parameters

row	Row of the cell
col	Column of the cell

7.37.3.2 DrawX()

```
void OGRID::TicTacToe::DrawX (
             int row,
             int col )  [private]
```

Draw an X on the grid.

Date

2023-12-06

Parameters

row	Row of the cell
col	Column of the cell

7.37.3.3 Initialize()

```
void OGRID::TicTacToe::Initialize ( )  [override], [virtual]
```

Initialize the game.

Date

2023-12-06

Implements OGRID::IGame.

7.37.3.4 IsDrawCondition()

```
bool OGRID::TicTacToe::IsDrawCondition ( )  [override], [virtual]
```

Check if the game has a draw condition.

Date

2023-12-06

Returns

true If the game has a draw condition, false otherwise

Implements OGRID::IGame.

7.37.3.5 IsWinningCondition()

```
bool OGRID::TicTacToe::IsWinningCondition ( )  [override], [virtual]
```

Check if the game has a winning condition.

Date

2023-12-06

Returns

true If the game has a winning condition, false otherwise

Implements OGRID::IGame.

7.37.3.6 OnGUIUpdateGrid()

```
void OGRID::TicTacToe::OnGUIUpdateGrid ( )  [override], [virtual]
```

Update the grid on the GUI.

Date

2023-12-06

Implements OGRID::IGame.

7.37.3.7 OnGUIUpdateGridHover()

```
void OGRID::TicTacToe::OnGUIUpdateGridHover (
             Vector2 cell )  [override], [virtual]
```

Update the grid on the GUI when hovering.

Date

2023-12-06

Parameters

cell	The cell that is being hovered

Implements OGRID::IGame.

7.37.3.8 SetupPlayers()

```
void OGRID::TicTacToe::SetupPlayers ( )  [override], [virtual]
```

Setup the players.

Date

2023-12-06

Implements OGRID::IGame.

7.37.3.9 TryMakeMove()

```
bool OGRID::TicTacToe::TryMakeMove (
             unsigned char & row,
             unsigned char & col )  [override], [virtual]
```

Try to make a move.

Date

2023-12-06

Parameters

row	Row of the cell
col	Column of the cell

Returns

true If the move was successful, false otherwise

Implements OGRID::IGame.

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

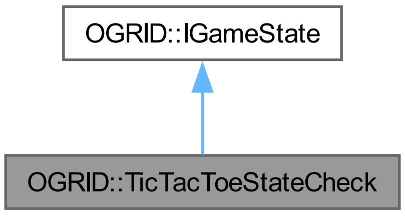
- Source/ogrid/Games/TicTacToe/TicTacToe.h
- Source/ogrid/Games/TicTacToe/TicTacToe.cpp

7.38 OGRID::TicTacToeStateCheck Class Reference

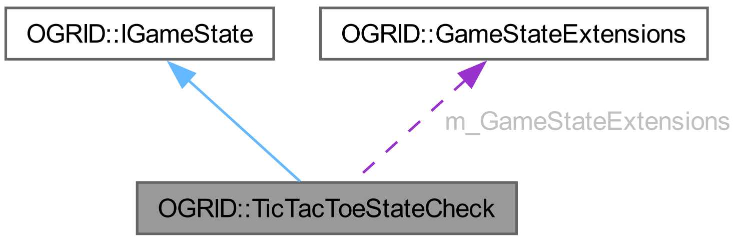
TicTacToe state check.

```
#include <TicTacToeStateCheck.h>
```

Inheritance diagram for OGRID::TicTacToeStateCheck:



Collaboration diagram for OGRID::TicTacToeStateCheck:



Public Member Functions

- int CheckWin (Grid ∗grid) const override
  Check if the game has a winning condition.
- bool IsDraw (Grid ∗grid) const override
  Check if the game has a draw condition.

Public Member Functions inherited from OGRID::IGameState

- virtual ∼IGameState ()
  Destroy the IGameState object.

Private Attributes

- GameStateExtensions m_GameStateExtensions = GameStateExtensions()
  TicTacToe state check.

7.38.1 Detailed Description

TicTacToe state check.

Check if the game has a winning condition or a draw condition.

Date

2023-12-06

See also

IGameState
GameStateExtensions

7.38.2 Member Function Documentation

7.38.2.1 CheckWin()

```
int OGRID::TicTacToeStateCheck::CheckWin (
             Grid * grid ) const  [override], [virtual]
```

Check if the game has a winning condition.

Date

2023-12-06

Parameters

grid	Grid to check

Returns

the side that won

See also

Player

Implements OGRID::IGameState.

7.38.2.2 IsDraw()

```
bool OGRID::TicTacToeStateCheck::IsDraw (
            Grid * grid ) const  [override], [virtual]
```

Check if the game has a draw condition.

Date

2023-12-06

Parameters

grid	Grid to check

Returns

true If the game has a draw condition, false otherwise

Implements OGRID::IGameState.

7.38.3 Member Data Documentation

7.38.3.1 m_GameStateExtensions

```
GameStateExtensions OGRID::TicTacToeStateCheck::m_GameStateExtensions = GameStateExtensions()
[private]
```

TicTacToe state check.

Date

2023-12-06

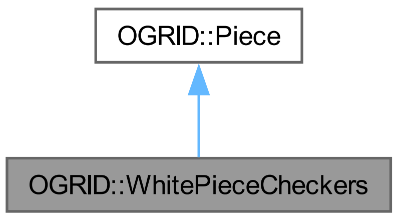
The documentation for this class was generated from the following files:

- Source/ogrid/Games/TicTacToe/TicTacToeStateCheck.h
- Source/ogrid/Games/TicTacToe/TicTacToeStateCheck.cpp

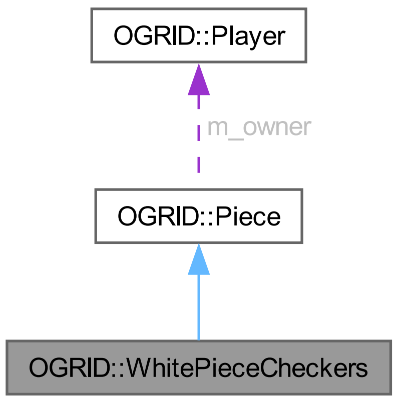
7.39 OGRID::WhitePieceCheckers Class Reference

```
#include <CheckersPieces.h>
```

Inheritance diagram for OGRID::WhitePieceCheckers:



Collaboration diagram for OGRID::WhitePieceCheckers:



Public Member Functions

- WhitePieceCheckers (Player ∗player)

Public Member Functions inherited from OGRID::Piece

- Piece (std::string rep, Player ∗player)

    Construct a new Piece object.
- ∼Piece ()

    Destroy the Piece object.

- void AddMoveRule (IMoveRule ∗rule)

  Add a move rule to the piece.
- void AddAttackRule (IAttackRule ∗rule)

  Add an attack rule to the piece.
- const std::string & GetRepresentation () const

  Get the representation of the piece.
- const Player ∗ GetOwner () const

  Get the owner of the piece.
- void SetOwner (Player ∗player)

  Set the owner of the piece.
- bool isValidMove (Grid ∗grid, int fromX, int fromY, int toX, int toY) const

  Check if the move is valid.
- bool isValidAttack (Grid ∗grid, int fromX, int fromY, int toX, int toY, bool &canContinue) const

  Check if the attack is valid.

Additional Inherited Members

Protected Attributes inherited from OGRID::Piece

- std::string m_representation

  The representation of the piece.
- std::vector< IMoveRule ∗ > m_moveRules

  The move rules of the piece.
- std::vector< IAttackRule ∗ > m_attackRules

  The attack rules of the piece.
- Player ∗ m_owner

  The owner of the piece.

7.39.1 Constructor & Destructor Documentation

7.39.1.1 WhitePieceCheckers()

```
OGRID::WhitePieceCheckers::WhitePieceCheckers (
            Player * player )
```

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

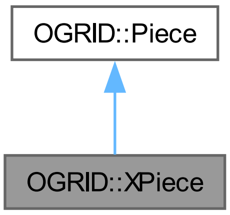
- Source/ogrid/Games/Checkers/CheckersPieces.h
- Source/ogrid/Games/Checkers/CheckersPieces.cpp

7.40 OGRID::XPiece Class Reference

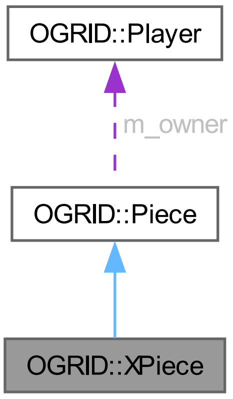
TicTacToe X piece.

```
#include <TicTacToePieces.h>
```

Inheritance diagram for OGRID::XPiece:



Collaboration diagram for OGRID::XPiece:



Public Member Functions

- XPiece (Player ∗player)

Public Member Functions inherited from OGRID::Piece

- Piece (std::string rep, Player ∗player)

  Construct a new Piece object.
- ∼Piece ()

  Destroy the Piece object.
- void AddMoveRule (IMoveRule ∗rule)

  Add a move rule to the piece.
- void AddAttackRule (IAttackRule ∗rule)

  Add an attack rule to the piece.
- const std::string & GetRepresentation () const

  Get the representation of the piece.
- const Player ∗ GetOwner () const

  Get the owner of the piece.
- void SetOwner (Player ∗player)

  Set the owner of the piece.
- bool isValidMove (Grid ∗grid, int fromX, int fromY, int toX, int toY) const

  Check if the move is valid.
- bool isValidAttack (Grid ∗grid, int fromX, int fromY, int toX, int toY, bool &canContinue) const

  Check if the attack is valid.

Additional Inherited Members

Protected Attributes inherited from OGRID::Piece

- std::string m_representation

  The representation of the piece.
- std::vector< IMoveRule ∗ > m_moveRules

  The move rules of the piece.
- std::vector< IAttackRule ∗ > m_attackRules

  The attack rules of the piece.
- Player ∗ m_owner

  The owner of the piece.

7.40.1 Detailed Description

TicTacToe X piece.

Date

2023-12-06

7.40.2 Constructor & Destructor Documentation

7.40.2.1 XPiece()

```
OGRID::XPiece::XPiece (
            Player ∗ player )
```

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

- Source/ogrid/Games/TicTacToe/TicTacToePieces.h
- Source/ogrid/Games/TicTacToe/TicTacToePieces.cpp

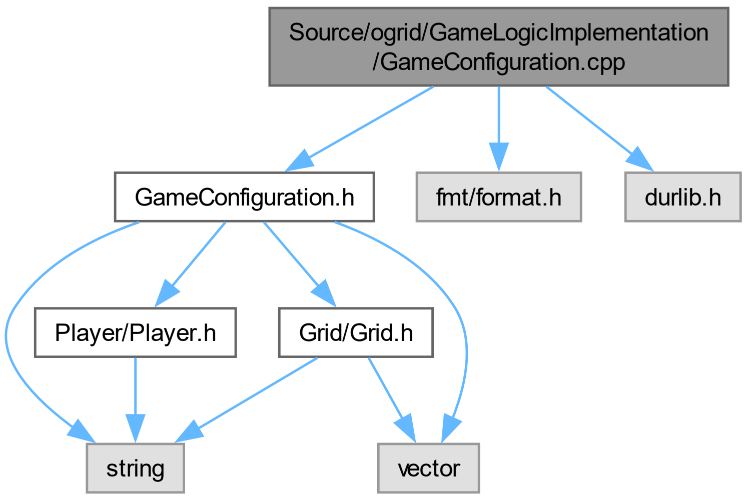
Chapter 8

File Documentation

8.1 Source/ogrid/GameLogicImplementation/GameConfiguration.cpp File Reference

```
#include "GameConfiguration.h"
#include "fmt/format.h"
#include <durlib.h>
```

Include dependency graph for GameConfiguration.cpp:



Classes

- struct fmt::formatter< OGRID::PlayerNameAndPtr >

    This is used to format a PlayerType enum into a string using fmt.

Namespaces

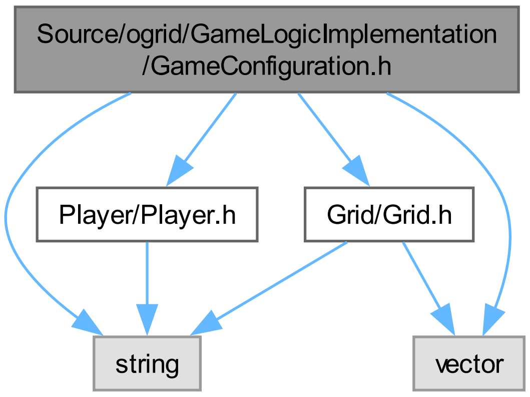
- namespace OGRID

8.2 Source/ogrid/GameLogicImplementation/GameConfiguration.h File Reference

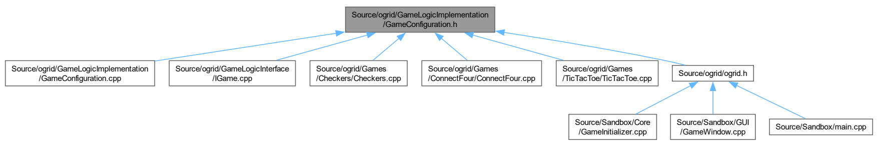
Contains the GameConfiguration class.

```
#include <string>
#include <vector>
#include "Grid/Grid.h"
#include "Player/Player.h"
```

Include dependency graph for GameConfiguration.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct OGRID::PlayerNameAndPtr

  Pair of player name and pointer.
- struct OGRID::GameConfiguration

  The GameConfiguration class. Used to represent a game configuration.
- struct OGRID::ConfigurationBuilder

  The ConfigurationBuilder interface.
- class OGRID::GameConfigurationBuilder

  The GameConfigurationBuilder class. Used to build a GameConfiguration object.

Namespaces

- namespace OGRID

Functions

- std::string OGRID::PlayerNameAndPtrVecToString (const std::vector< PlayerNameAndPtr > &players)

8.2.1 Detailed Description

Contains the GameConfiguration class.

Date

2023-12-06

8.3 GameConfiguration.h

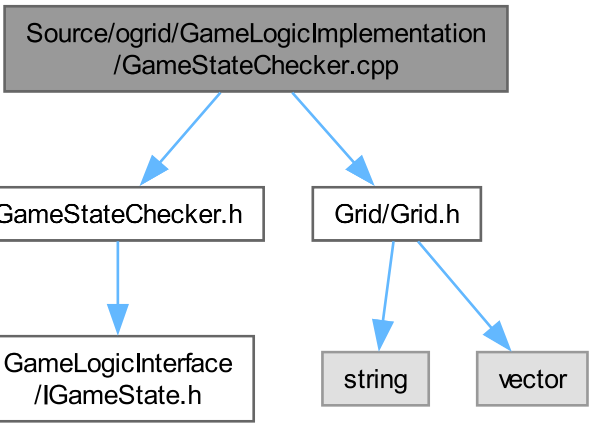
Go to the documentation of this file.

```cpp
#pragma once

#include <string>
#include <vector>

#include "Grid/Grid.h"
#include "Player/Player.h"

namespace OGRID
{
    // Forward declarations
    // class ConfigurationBuilder;
    // class ITurnManager;
    // class Player;

    struct PlayerNameAndPtr
    {
        std::string name;

        Player *ptr;
    };

    struct GameConfiguration
    {
        std::string gameName;

        std::string gameDescription;

        Grid *grid = nullptr;

        size_t maxPlayers = 0;

        std::vector<Player *> players;

        std::vector<PlayerNameAndPtr> playerPairs;
    };

    // Builder Interface
    struct ConfigurationBuilder
    {
        virtual ~ConfigurationBuilder() = default;

        virtual ConfigurationBuilder &setGameName(const std::string &gameName) = 0;

        virtual ConfigurationBuilder &setGameDescription(const std::string &gameDescription) = 0;

        virtual ConfigurationBuilder &setGrid(unsigned char rows, unsigned char cols, Piece *defaultPiece = nullptr) = 0;

        virtual ConfigurationBuilder &setMaxPlayers(size_t maxPlayers) = 0;

        virtual ConfigurationBuilder &addPlayer(Player *player) = 0;

        virtual GameConfiguration *build() = 0;
    };

    // Concrete Builder
    class GameConfigurationBuilder : public ConfigurationBuilder
    {
    private:
        GameConfiguration m_GameConfiguration;

    public:
        GameConfigurationBuilder() = default;

        ~GameConfigurationBuilder() override = default;

        ConfigurationBuilder &setGameName(const std::string &gameName) override;

        ConfigurationBuilder &setGameDescription(const std::string &gameDescription) override;

        ConfigurationBuilder &setGrid(unsigned char rows, unsigned char cols, Piece *defaultPiece = nullptr) override;

        ConfigurationBuilder &setMaxPlayers(size_t maxPlayers) override;

        ConfigurationBuilder &addPlayer(Player *player) override;

        GameConfiguration *build() override;
    };

    std::string PlayerNameAndPtrVecToString(const std::vector<PlayerNameAndPtr> &players);
}

// namespace OGRID
// {
//     static std::string PlayerNameAndPtrVecToString(const std::vector<PlayerNameAndPtr> &players)
//     {
//         std::ostringstream ss;
//         for (size_t i = 0; i < players.size(); ++i)
//         {
//             if (i > 0)
//                 ss « "\n";
//             ss « fmt::format("{}", players[i]);
//         }
//         return ss.str();
//     }
// }
```

8.4 Source/ogrid/GameLogicImplementation/GameStateChecker.cpp File Reference

```
#include "GameStateChecker.h"
#include "Grid/Grid.h"
```

Include dependency graph for GameStateChecker.cpp:



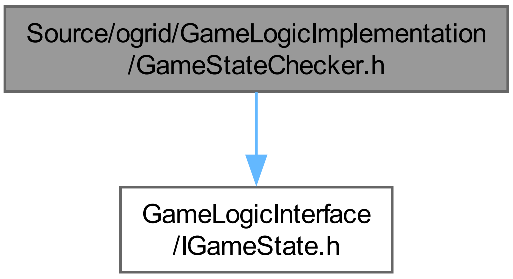
Namespaces

- namespace OGRID

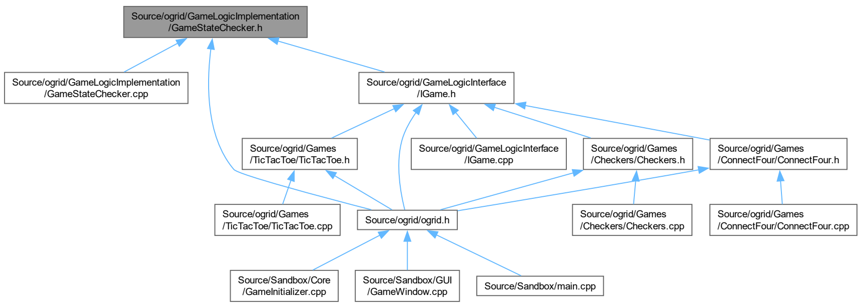
8.5 Source/ogrid/GameLogicImplementation/GameStateChecker.h File Reference

Contains the GameStateChecker class.

```
#include "GameLogicInterface/IGameState.h"
```
Include dependency graph for GameStateChecker.h:



This graph shows which files directly or indirectly include this file:



Classes

- class OGRID::GameStateChecker

  The GameStateChecker class. Used to check the state of the game.

Namespaces

- namespace OGRID

8.5.1 Detailed Description

Contains the GameStateChecker class.

Date

2023-12-06

8.6 GameStateChecker.h

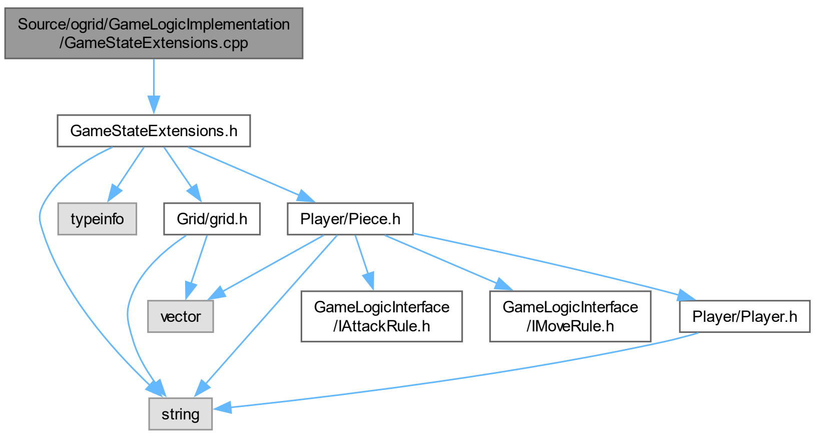
Go to the documentation of this file.

```
#pragma once

#include "GameLogicInterface/IGameState.h"

/*
How to use:
GameStateChecker gameStateCheckerTicTacToe(new TicTacToeStateCheck());
*/

namespace OGRID
{
    class GameStateChecker
    {
    private:
        IGameState *m_GameState;

    public:
        GameStateChecker(IGameState *strategy);

        ~GameStateChecker();

        int CheckWin(Grid *grid) const;

        bool IsDraw(Grid *grid) const;

        bool IsColumnOccupied(Grid *grid, unsigned char colToCheck, unsigned char &rowToFill);

        unsigned char GetTopMostPiecePositionInColumn(Grid *grid, int col);
    };
}
```

8.7 Source/ogrid/GameLogicImplementation/GameStateExtensions.cpp File Reference

```
#include "GameStateExtensions.h"
```

Include dependency graph for GameStateExtensions.cpp:



Namespaces

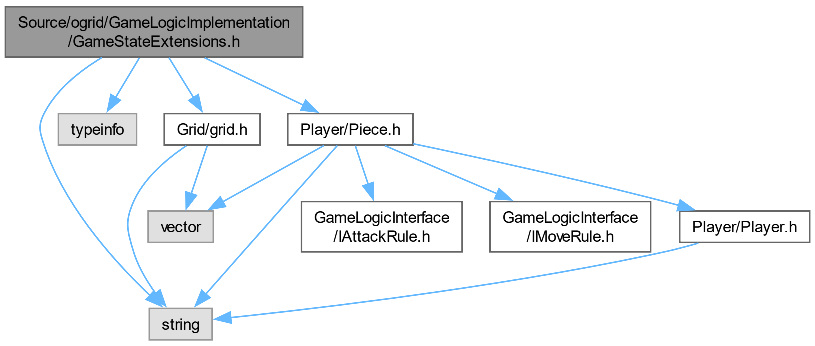
- namespace OGRID

8.8 Source/ogrid/GameLogicImplementation/GameStateExtensions.h File Reference

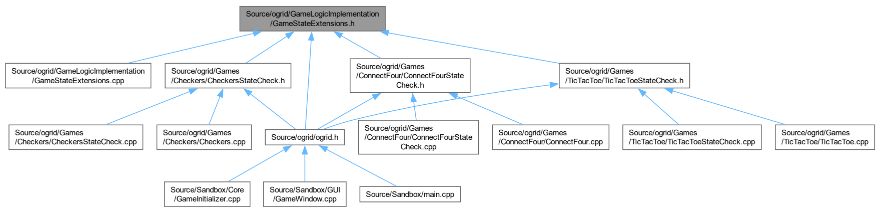
Contains the GameStateExtensions class.

```
#include <string>
#include <typeinfo>
#include "Grid/grid.h"
#include "Player/Piece.h"
```

Include dependency graph for GameStateExtensions.h:



This graph shows which files directly or indirectly include this file:


Source/ogrid/GameLogicImplementation /GameStateExtensions.h
Source/ogrid/GameLogicImplementation /GameStateExtensions.cpp
Source/ogrid/Games /Checkers/CheckersStateCheck.h
Source/ogrid/Games /ConnectFour/ConnectFourState Check.h
Source/ogrid/Games /TicTacToe/TicTacToeStateCheck.h
Source/ogrid/Games /Checkers/CheckersStateCheck.cpp
Source/ogrid/Games /Checkers/Checkers.cpp
Source/ogrid/ogrid.h
Source/ogrid/Games /ConnectFour/ConnectFourState Check.cpp
Source/ogrid/Games /ConnectFour/ConnectFour.cpp
Source/ogrid/Games /TicTacToe/TicTacToeStateCheck.cpp
Source/ogrid/Games /TicTacToe/TicTacToe.cpp
Source/Sandbox/Core /GameInitializer.cpp
Source/Sandbox/GUI /GameWindow.cpp
Source/Sandbox/main.cpp

Classes

- class OGRID::GameStateExtensions

  The GameStateExtensions class. Used to extend the GameStateChecker class.

Namespaces

- namespace OGRID

8.8.1 Detailed Description

Contains the GameStateExtensions class.

Date

2023-12-06

8.9 GameStateExtensions.h

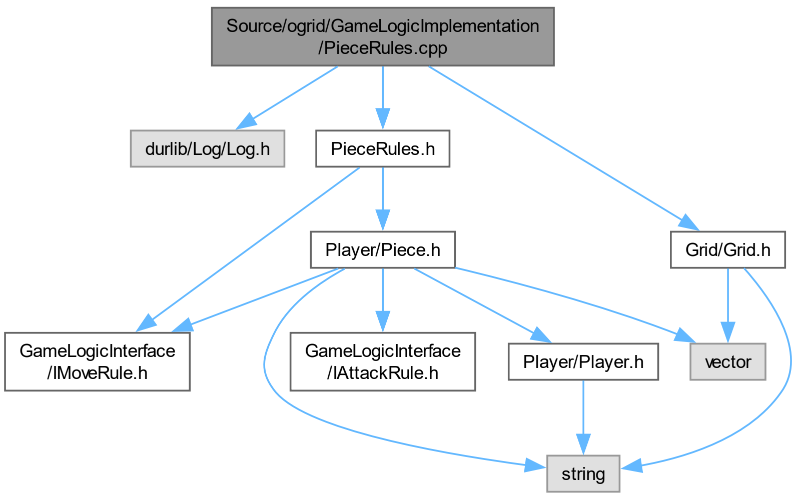
Go to the documentation of this file.

```
#pragma once

#include <string>
#include <typeinfo>

#include "Grid/grid.h"
#include "Player/Piece.h"

/*
Checking for class type of pieces would look like this:
CheckForRecurringPieceInRow(typeid(XPiece), 3))
*/
namespace OGRID
{
    // Forward declarations
    // class Grid;
    // class Piece;

    class GameStateExtensions
    {
    public:
        bool CheckForRecurringStringInRow(Grid *grid, const std::string &pieceRepresentation, unsigned char dupeCount) const;
        bool CheckForRecurringPieceInRow(Grid *grid, const std::type_info &pieceType, unsigned char dupeCount) const;

        bool CheckForRecurringStringInCol(Grid *grid, const std::string &pieceRepresentation, unsigned char dupeCount) const;

        bool CheckForRecurringPieceInCol(Grid *grid, const std::type_info &pieceType, unsigned char dupeCount) const;

        bool CheckForRecurringStringInDiagonal(Grid *grid, const std::string &pieceRepresentation, unsigned char dupeCount) const;

        bool CheckForRecurringPieceInDiagonal(Grid *grid, const std::type_info &pieceType, unsigned char dupeCount) const;

        bool CheckForRecurringStringInAntiDiagonal(Grid *grid, const std::string &pieceRepresentation, unsigned char dupeCount) const;

        bool CheckForRecurringPieceInAntiDiagonal(Grid *grid, const std::type_info &pieceType, unsigned char dupeCount) const;

        bool CheckIfAllSpotsFilled(Grid *grid) const;
    };
}
```

8.10 Source/ogrid/GameLogicImplementation/PieceRules.cpp File Reference

```
#include <durlib/Log/Log.h>
#include "PieceRules.h"
#include "Grid/Grid.h"
```

Include dependency graph for PieceRules.cpp:


Source/ogrid/GameLogicImplementation/PieceRules.cpp
durlib/Log/Log.h
PieceRules.h
Player/Piece.h
Grid/Grid.h
GameLogicInterface/IMoveRule.h
GameLogicInterface/IAttackRule.h
Player/Player.h
vector
string

Namespaces

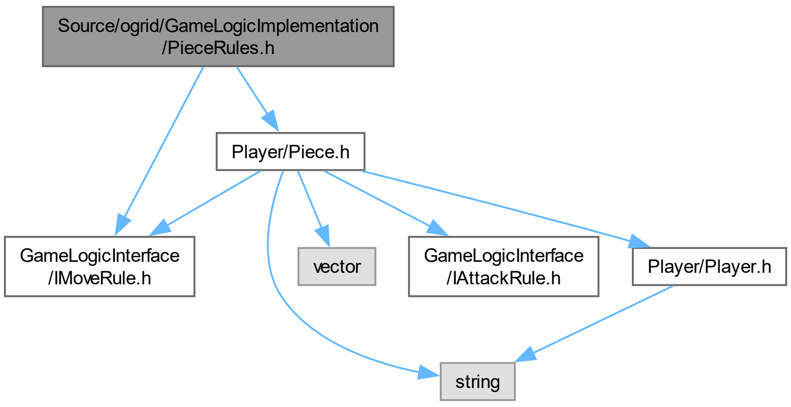
- namespace OGRID

8.11 Source/ogrid/GameLogicImplementation/PieceRules.h File Reference

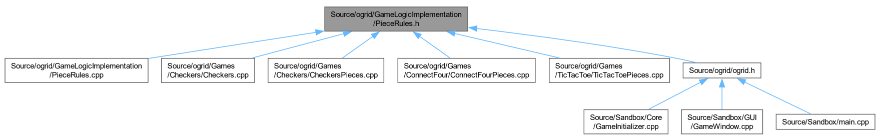
Contains the PieceRules class.

```
#include "GameLogicInterface/IMoveRule.h"
#include "Player/Piece.h"
```

Include dependency graph for PieceRules.h:



This graph shows which files directly or indirectly include this file:



Classes

- class OGRID::SimplePlaceMoveRule

  The PieceRules class. Used to represent the rules of a piece.
- class OGRID::NormalCheckersMoveRule

  The PieceRules class. Used to represent the rules of a piece.
- class OGRID::JumpNormalCheckersAttackRule

  The PieceRules class. Used to represent the rules of a piece.
- class OGRID::SuperCheckersMoveRule

  The PieceRules class. Used to represent the rules of a piece.
- class OGRID::JumpSuperCheckersAttackRule

  The PieceRules class. Used to represent the rules of a piece.

Namespaces

- namespace OGRID

8.11.1 Detailed Description

Contains the PieceRules class.

Date

2023-12-06

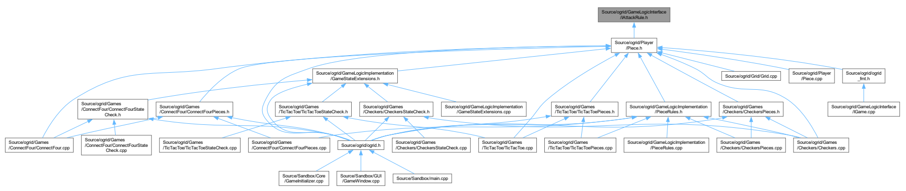
8.12 PieceRules.h

Go to the documentation of this file.

```
#pragma once

#include "GameLogicInterface/IMoveRule.h"
#include "Player/Piece.h"

namespace OGRID
{

    class SimplePlaceMoveRule : public IMoveRule
    {
    public:
        // We only need to check if the end cell is unoccupied, as there is no concept of a "start" cell in tic tac toe.
        bool IsValidMove(Grid *grid, int fromX, int fromY, int toX, int toY) const override;
    };

    class NormalCheckersMoveRule : public IMoveRule
    {
    public:
        bool IsValidMove(Grid *grid, int fromX, int fromY, int toX, int toY) const override;
    };

    class JumpNormalCheckersAttackRule : public IAttackRule
    {
    public:
        bool IsValidAttack(Grid *grid, int fromX, int fromY, int toX, int toY, bool &canContinue) const override;
    };

    class SuperCheckersMoveRule : public IMoveRule
    {
    public:
        bool IsValidMove(Grid *grid, int fromX, int fromY, int toX, int toY) const override;
    };

    class JumpSuperCheckersAttackRule : public IAttackRule
    {
    public:
        bool IsValidAttack(Grid *grid, int fromX, int fromY, int toX, int toY, bool &canContinue) const override;
    };
}
```

8.13 Source/ogrid/GameLogicInterface/IAttackRule.h File Reference

This graph shows which files directly or indirectly include this file:


Source/ogrid/GameLogicInterface/IAttackRule.h
Source/ogrid/Player/Piece.h
Source/ogrid/GameLogicImplementation/GameStateExtensions.h
Source/ogrid/Grid/Grid.cpp
Source/ogrid/Player/Piece.cpp
Source/ogrid/ogrid_fmt.h
Source/ogrid/Games/ConnectFour/ConnectFourStateCheck.h
Source/ogrid/Games/ConnectFour/ConnectFourPieces.h
Source/ogrid/Games/TicTacToe/TicTacToeStateCheck.h
Source/ogrid/Games/Checkers/CheckersStateCheck.h
Source/ogrid/GameLogicImplementation/GameStateExtensions.cpp
Source/ogrid/Games/TicTacToe/TicTacToePieces.h
Source/ogrid/GameLogicImplementation/PieceRules.h
Source/ogrid/Games/Checkers/CheckersPieces.h
Source/ogrid/GameLogicInterface/IGame.cpp
Source/ogrid/Games/ConnectFour/ConnectFour.cpp
Source/ogrid/Games/ConnectFour/ConnectFourStateCheck.cpp
Source/ogrid/Games/TicTacToe/TicTacToeStateCheck.cpp
Source/ogrid/Games/ConnectFour/ConnectFourPieces.cpp
Source/ogrid/ogrid.h
Source/ogrid/Games/Checkers/CheckersStateCheck.cpp
Source/ogrid/Games/TicTacToe/TicTacToe.cpp
Source/ogrid/Games/TicTacToe/TicTacToePieces.cpp
Source/ogrid/GameLogicImplementation/PieceRules.cpp
Source/ogrid/Games/Checkers/CheckersPieces.cpp
Source/ogrid/Games/Checkers/Checkers.cpp
Source/Sandbox/Core/GameInitializer.cpp
Source/Sandbox/GUI/GameWindow.cpp
Source/Sandbox/main.cpp

Classes

- class OGRID::IAttackRule

  The IGameState class. Used to check the state of the game.

Namespaces

- namespace OGRID

8.14 IAttackRule.h

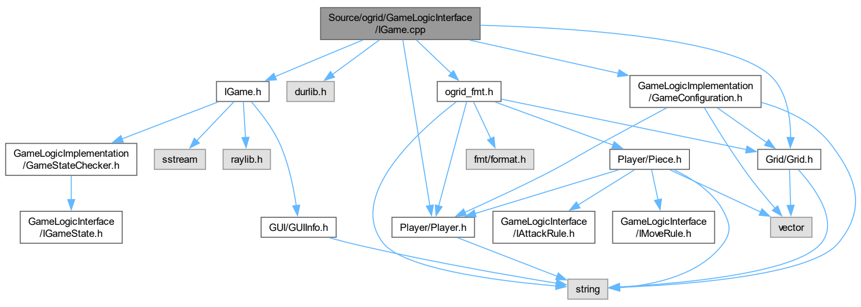
Go to the documentation of this file.

```
#pragma once

namespace OGRID
{
    class Grid;

    class IAttackRule
    {
    public:
        virtual ~IAttackRule() {}

        // canContinue shows if there’s another attack available after this one.
        virtual bool IsValidAttack(Grid *grid, int x, int y, int x2, int y2, bool &canContinue) const
    = 0;
    };
}
```

8.15 Source/ogrid/GameLogicInterface/IGame.cpp File Reference

```
#include "IGame.h"
#include <durlib.h>
#include "ogrid_fmt.h"
#include "Grid/Grid.h"
#include "Player/Player.h"
#include "GameLogicImplementation/GameConfiguration.h"
```

Include dependency graph for IGame.cpp:



Namespaces

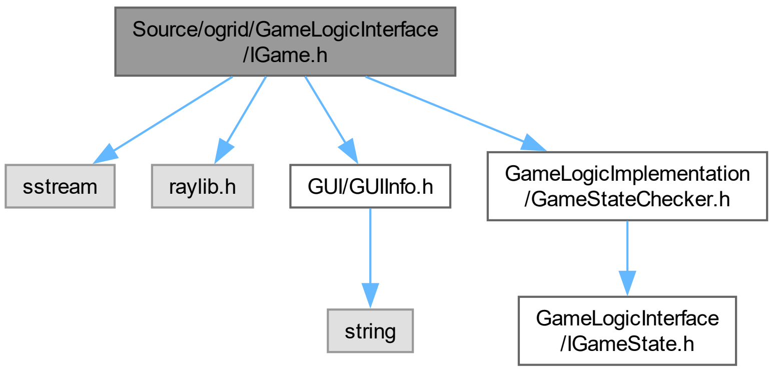
- namespace OGRID

8.16 Source/ogrid/GameLogicInterface/IGame.h File Reference

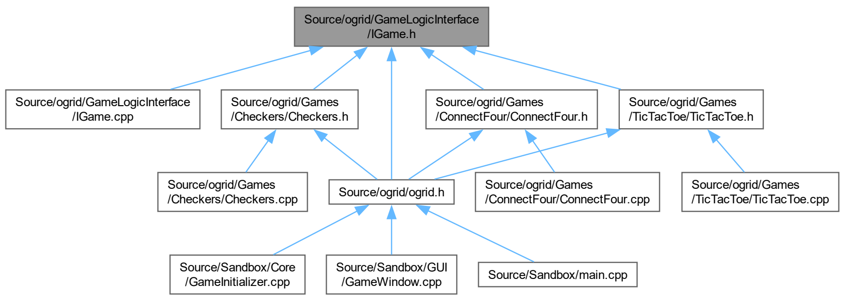
Contains the IGame class.

```
#include <sstream>
#include <raylib.h>
#include "GUI/GUIInfo.h"
#include "GameLogicImplementation/GameStateChecker.h"
```

Include dependency graph for IGame.h:



This graph shows which files directly or indirectly include this file:



Classes

- class OGRID::IGame

  The IGame class. Used to represent a game.

Namespaces

- namespace OGRID

Enumerations

- enum OGRID::GameState { OGRID::NotStarted = 0 , OGRID::InProgress = 1 , OGRID::Paused = 2 , OGRID::GameOver = 3 }

  The IGame class. Used to represent a game.
- enum OGRID::GameOverType { OGRID::None = 0 , OGRID::Win = 1 , OGRID::Draw = 2 }

  The GameOverType enum. Used to represent the type of game over.

8.16.1 Detailed Description

Contains the IGame class.

Date

2023-12-06

8.17 IGame.h

Go to the documentation of this file.

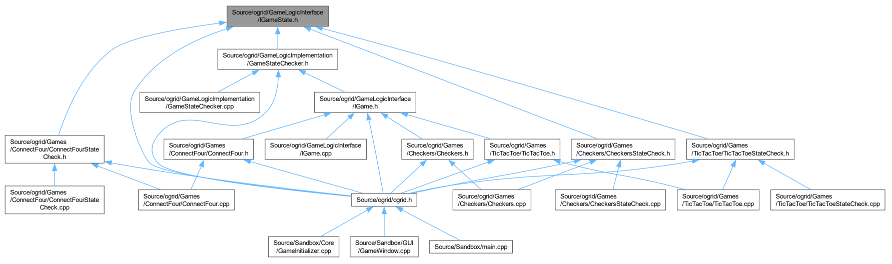
```
#pragma once

#include <sstream>

#include <raylib.h>

#include "GUI/GUIInfo.h"
#include "GameLogicImplementation/GameStateChecker.h"

// TODO: Keep track of the current player.

namespace OGRID
{
    // Forward declaration
    class GameConfiguration;
    class Player;
    class Grid;
    enum MoveType;
    struct PlayerNameAndPtr;

    enum GameState
    {
        NotStarted = 0,
        InProgress = 1,
        Paused = 2,
        GameOver = 3
    };

    enum GameOverType
    {
        None = 0,
        Win = 1,
        Draw = 2
    };

    class IGame
    {
    public:
        GUIInfo m_guiInfo;

        bool m_randomizeTurnOrder = true;

    protected:
        GameStateChecker *m_currentGameState;

        GameState m_gameState = GameState::NotStarted;

        GameOverType m_gameOverType = GameOverType::None;

        Player *m_winner = nullptr;

        Player *m_currentPlayer = nullptr;
```

```cpp

        size_t m_currentTurn = 0;

        unsigned int m_totalTurns = 0;

        GameConfiguration *m_GameConfiguration = nullptr;

        IGame() = default;

        IGame(IGameState *gameStateStrategy, const std::vector<OGRID::PlayerNameAndPtr> &players);

        ~IGame();

    public:
        virtual bool TryMakeMove(unsigned char &row, unsigned char &col) = 0;
        // virtual bool IsWinningCondition(unsigned char row, unsigned char col) = 0;
        // virtual bool IsWinningCondition(char playerChar) = 0;
        // virtual bool IsDrawCondition(unsigned char row, unsigned char col) = 0;

        virtual bool IsWinningCondition() = 0;

        virtual bool IsDrawCondition() = 0;
        // virtual void SetupPlayers(const std::vector<int> &totalValidSides) = 0;

        virtual void SetupPlayers() = 0;

        virtual void Initialize() = 0;

        // Game specific GUI Grid stuff drawing (X and O for Tic Tac Toe for example).
        virtual void OnGUIUpdateGrid() = 0;

        // On hovering over a grid spot.
        virtual void OnGUIUpdateGridHover(Vector2 cell) = 0;

        void SwapPlayerPositions();

        void ResetGrid();

        void ResetPlayers();

        void PrintPlayersTurnOrder() const;

        void SetupGame();

        void ResetGame();

        void StartGame();

        void PrintPlayerMoves() const;

        void MakeMove(unsigned char row, unsigned char col);

        void Reset();

        // Switch player turns forcefully after a move is made. This is added purely for testing purposes.
        void SwitchPlayer();

        OGRID::GameOverType CheckGameOverState(OGRID::Grid *grid, unsigned char row, unsigned char col);

        // void ChangeGridSize();

        // Getters and Setters
        GameState GetGameState() const;

        void SetGameState(GameState gameState);

        GameOverType GetGameOverType() const;

        Player *GetWinner() const;

        GameConfiguration *GetGameConfiguration() const;

        void SetGameConfiguration(GameConfiguration *gameConfiguration);

        std::string GetGameName() const;

        Grid *GetGrid() const;

        std::vector<Player *> GetPlayers() const;

        void SetRandomizeTurnOrder(bool randomize);

        OGRID::PlayerNameAndPtr GetCurrentPlayer() const;

        // This is solely for testing purposes.
        void SetCurrentPlayer(OGRID::PlayerNameAndPtr player);

        size_t GetCurrentTurn() const;

        GameStateChecker *GetGameStateChecker() const;

        void SetGameStateChecker(GameStateChecker *gameStateChecker);

        std::vector<std::string> GetPlayerNames() const;

        std::vector<OGRID::Player *> GetPlayerPtrs() const;

        OGRID::PlayerNameAndPtr GetPlayerPair(size_t at) const;

        std::vector<OGRID::PlayerNameAndPtr> GetPlayerPairs() const;

        void SetPlayerPairs(const std::vector<OGRID::PlayerNameAndPtr> &players);

        GUIInfo GetGUIInfo() const;

        void SetGUIInfo(const GUIInfo &guiInfo);
    };
}
```

8.18 Source/ogrid/GameLogicInterface/IGameState.h File Reference

Contains the IGameState class.

This graph shows which files directly or indirectly include this file:



Source/ogrid/GameLogicInterface/IGameState.h
Source/ogrid/GameLogicImplementation/GameStateChecker.h
Source/ogrid/GameLogicImplementation/GameStateChecker.cpp
Source/ogrid/GameLogicInterface/IGame.h
Source/ogrid/Games/ConnectFour/ConnectFourStateCheck.h
Source/ogrid/Games/ConnectFour/ConnectFour.h
Source/ogrid/GameLogicInterface/IGame.cpp
Source/ogrid/Games/Checkers/Checkers.h
Source/ogrid/Games/TicTacToe/TicTacToe.h
Source/ogrid/Games/Checkers/CheckersStateCheck.h
Source/ogrid/Games/TicTacToe/TicTacToeStateCheck.h
Source/ogrid/Games/ConnectFour/ConnectFourStateCheck.cpp
Source/ogrid/Games/ConnectFour/ConnectFour.cpp
Source/ogrid/ogrid.h
Source/ogrid/Games/Checkers/Checkers.cpp
Source/ogrid/Games/Checkers/CheckersStateCheck.cpp
Source/ogrid/Games/TicTacToe/TicTacToe.cpp
Source/ogrid/Games/TicTacToe/TicTacToeStateCheck.cpp
Source/Sandbox/Core/GameInitializer.cpp
Source/Sandbox/GUI/GameWindow.cpp
Source/Sandbox/main.cpp

Classes

- class OGRID::IGameState

  The IGameState class. Used to check the state of the game.

Namespaces

- namespace OGRID

8.18.1 Detailed Description

Contains the IGameState class.

Date

2023-12-06

8.19 IGameState.h

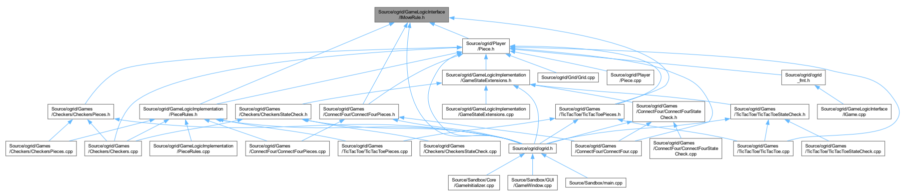
Go to the documentation of this file.

```
#pragma once

namespace OGRID
{
    // Forward declarations
    class Grid;

    class IGameState
    {
    public:
        virtual ~IGameState() {}

        // Returns side number of the winner. If less than 0, then there is no winner. We have a specific method for that checking draw.
        virtual int CheckWin(Grid *grid) const = 0;

        virtual bool IsDraw(Grid *grid) const = 0;
    };
}
```

8.20 Source/ogrid/GameLogicInterface/IMoveRule.h File Reference

Contains the IMoveRule class.

This graph shows which files directly or indirectly include this file:



Classes

- class OGRID::IMoveRule

  The IMoveRule class. Used to check if the move is valid.

Namespaces

- namespace OGRID

8.20.1 Detailed Description

Contains the IMoveRule class.

Date

2023-12-06

8.21 IMoveRule.h

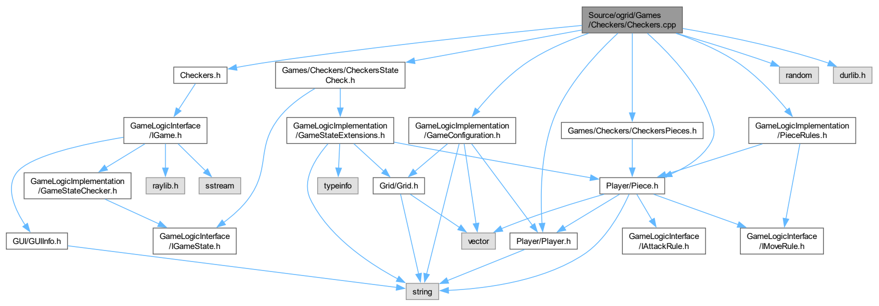
Go to the documentation of this file.

```
#pragma once

namespace OGRID
{
    class Grid;

    class IMoveRule
    {
    public:
        virtual ~IMoveRule() {}

        // ALL MOVES REQUIRE START <from> AND END <to> COORDINATES. BUT MAKE SURE THAT THE COORDINATE
YOU WANT TO PLACE YOUR PIECE AT IS THE END <to> COORDINATES.
        virtual bool IsValidMove(Grid *grid, int fromX, int fromY, int toX, int toY) const = 0;
    };
}
```

8.22 Source/ogrid/Games/Checkers/Checkers.cpp File Reference

```
#include "Checkers.h"
#include <random>
#include <durlib.h>
#include "GameLogicImplementation/GameConfiguration.h"
#include "Games/Checkers/CheckersPieces.h"
#include "Games/Checkers/CheckersStateCheck.h"
#include "Player/Piece.h"
#include "Player/Player.h"
#include "GameLogicImplementation/PieceRules.h"
```

Include dependency graph for Checkers.cpp:


Source/ogrid/Games/Checkers/Checkers.cpp
Checkers.h
Games/Checkers/CheckersStateCheck.h
random
durlib.h
GameLogicInterface/IGame.h
GameLogicImplementation/GameStateExtensions.h
GameLogicImplementation/GameConfiguration.h
Games/Checkers/CheckersPieces.h
GameLogicImplementation/PieceRules.h
GameLogicImplementation/GameStateChecker.h
raylib.h
sstream
typeinfo
Grid/Grid.h
Player/Piece.h
GUI/GUIInfo.h
GameLogicInterface/IGameState.h
vector
Player/Player.h
GameLogicInterface/IAttackRule.h
GameLogicInterface/IMoveRule.h
string

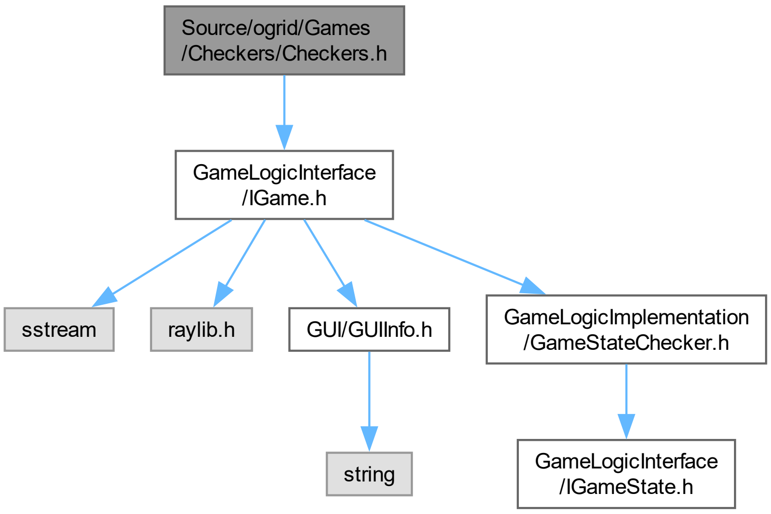
Namespaces

- namespace OGRID

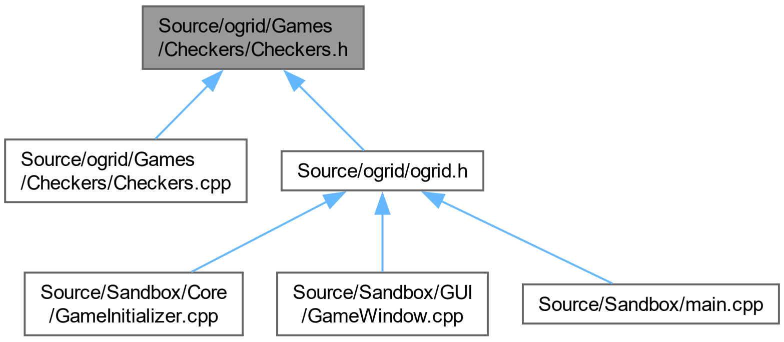
8.23 Source/ogrid/Games/Checkers/Checkers.h File Reference

```
#include "GameLogicInterface/IGame.h"
```

Include dependency graph for Checkers.h:


Source/ogrid/Games/Checkers/Checkers.h
GameLogicInterface/IGame.h
sstream
raylib.h
GUI/GUIInfo.h
GameLogicImplementation/GameStateChecker.h
string
GameLogicInterface/IGameState.h

This graph shows which files directly or indirectly include this file:


Source/ogrid/Games/Checkers/Checkers.h
Source/ogrid/Games/Checkers/Checkers.cpp
Source/ogrid/ogrid.h
Source/Sandbox/Core/GameInitializer.cpp
Source/Sandbox/GUI/GameWindow.cpp
Source/Sandbox/main.cpp

Classes

- class OGRID::Checkers

Namespaces

- namespace OGRID

8.24 Checkers.h

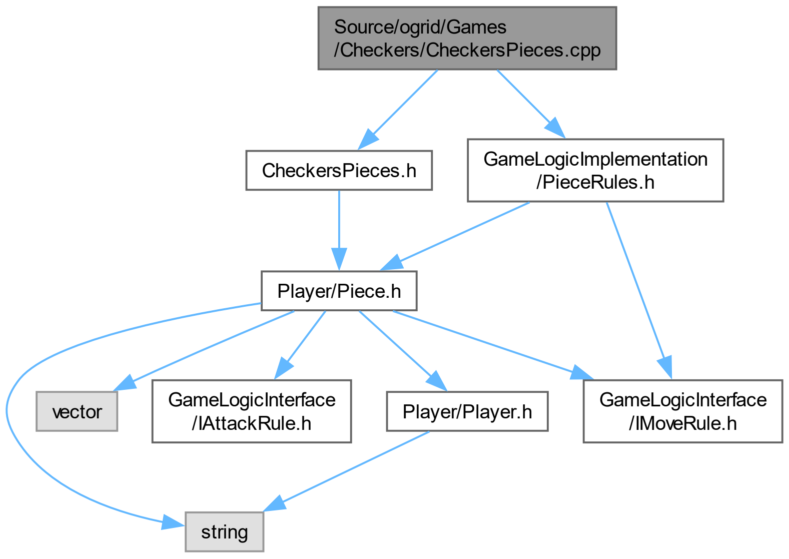
Go to the documentation of this file.

```
#pragma once

#include "GameLogicInterface/IGame.h"

namespace OGRID
{
    class Piece;

    class Checkers : public IGame
    {
        // Contains references to super pieces
        std::vector<Piece *> m_Supers;
        // Contains reference to the selected piece
        Piece *m_SelectedPiece = nullptr;
        // Just to keep track of positions
        std::map<std::pair<int, int>, Piece *> m_Pieces;

        // Variables for alpha of circles
        float alpha = 1.0f;
        // Speed of the transition
        float alphaSpeed = 0.025f;

    public:
        Checkers() = default;
        ~Checkers() = default;

        bool TryMakeMove(unsigned char &row, unsigned char &col) override;
        bool IsWinningCondition() override;
        bool IsDrawCondition() override;
        void SetupPlayers() override;

        void Initialize() override;
        void OnGUIUpdateGrid() override;
        void OnGUIUpdateGridHover(Vector2 cell) override;

    private:
        void SetupBoard();

        void AddAsSuperPiece(Piece *piece);
        void RemoveSuperPiece(Piece *piece);
        bool IsSuperPiece(Piece *piece);

        void AddPieceToPieceManager(Piece *piece, std::pair<int, int> position);
        void RemovePieceFromPieceManager(Piece *piece);
        void RemovePieceFromPieceManager(std::pair<int, int> position);
        std::pair<int, int> GetPiecePosition(Piece *piece);
        void SetPiecePosition(Piece *piece, std::pair<int, int> position);

        void DrawPiece(int row, int col, Color color, bool blinking, bool super);
        void DrawCell(int row, int col);
    };
}
```

8.25 Source/ogrid/Games/Checkers/CheckersPieces.cpp File Reference

```
#include "CheckersPieces.h"
#include "GameLogicImplementation/PieceRules.h"
```

Include dependency graph for CheckersPieces.cpp:



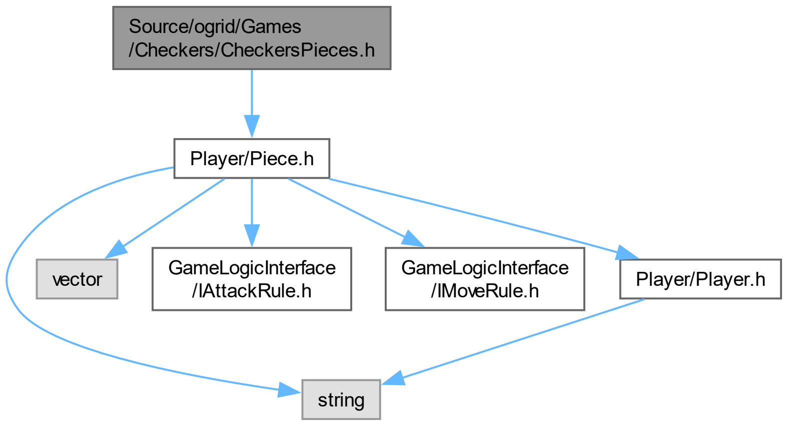
Namespaces

- namespace OGRID

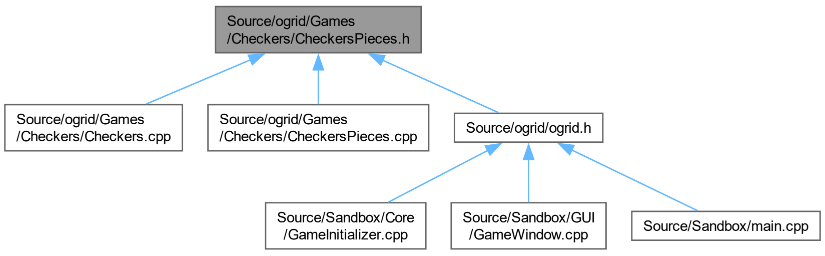
8.26 Source/ogrid/Games/Checkers/CheckersPieces.h File Reference

```
#include <Player/Piece.h>
```

Include dependency graph for CheckersPieces.h:



This graph shows which files directly or indirectly include this file:


Source/ogrid/Games/Checkers/CheckersPieces.h
Source/ogrid/Games/Checkers/Checkers.cpp
Source/ogrid/Games/Checkers/CheckersPieces.cpp
Source/ogrid/ogrid.h
Source/Sandbox/Core/GameInitializer.cpp
Source/Sandbox/GUI/GameWindow.cpp
Source/Sandbox/main.cpp

Classes

- class OGRID::WhitePieceCheckers
- class OGRID::BlackPieceCheckers

Namespaces

- namespace OGRID

8.27 CheckersPieces.h

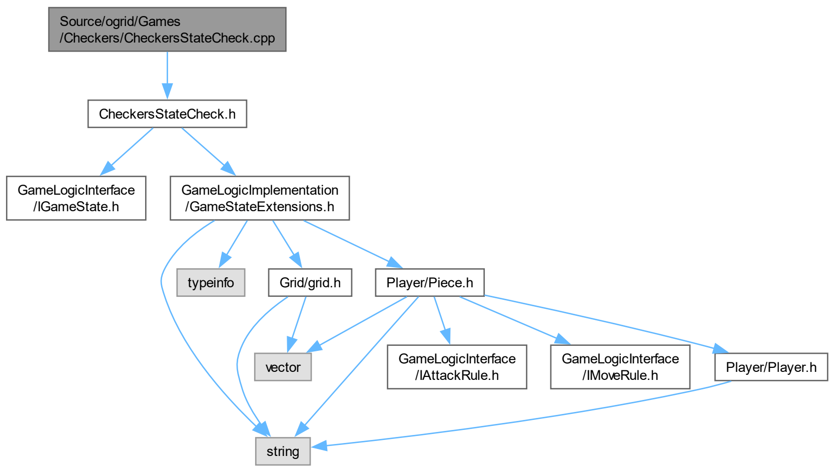
Go to the documentation of this file.

```
#pragma once

#include <Player/Piece.h>

namespace OGRID
{
    class Player;
    class Grid;

    class WhitePieceCheckers : public Piece
    {
    public:
        WhitePieceCheckers(Player *player);
    };

    class BlackPieceCheckers : public Piece
    {
    public:
        BlackPieceCheckers(Player *player);
    };
}
```

8.28 Source/ogrid/Games/Checkers/CheckersStateCheck.cpp File Reference

```
#include "CheckersStateCheck.h"
```

Include dependency graph for CheckersStateCheck.cpp:



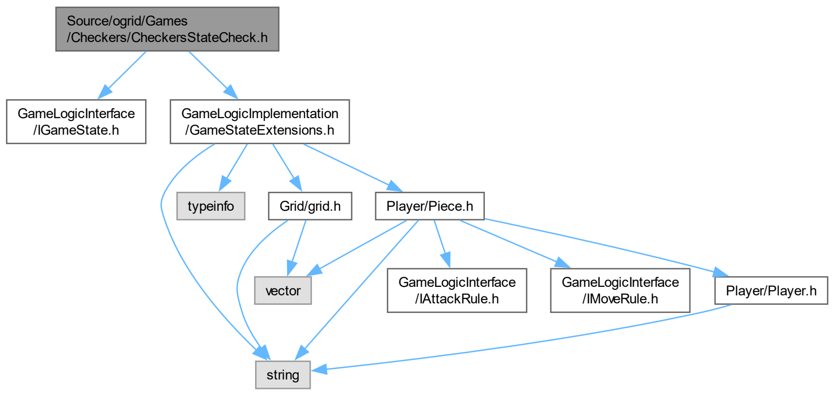
Namespaces

- namespace OGRID

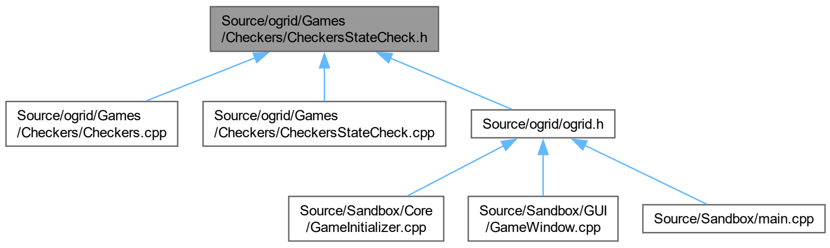
8.29 Source/ogrid/Games/Checkers/CheckersStateCheck.h File Reference

```
#include "GameLogicInterface/IGameState.h"
#include "GameLogicImplementation/GameStateExtensions.h"
```

Include dependency graph for CheckersStateCheck.h:



This graph shows which files directly or indirectly include this file:


Source/ogrid/Games /Checkers/CheckersStateCheck.h
Source/ogrid/Games /Checkers/Checkers.cpp
Source/ogrid/Games /Checkers/CheckersStateCheck.cpp
Source/ogrid/ogrid.h
Source/Sandbox/Core /GameInitializer.cpp
Source/Sandbox/GUI /GameWindow.cpp
Source/Sandbox/main.cpp

Classes

- class OGRID::CheckersStateCheck

Namespaces

- namespace OGRID

8.30 CheckersStateCheck.h

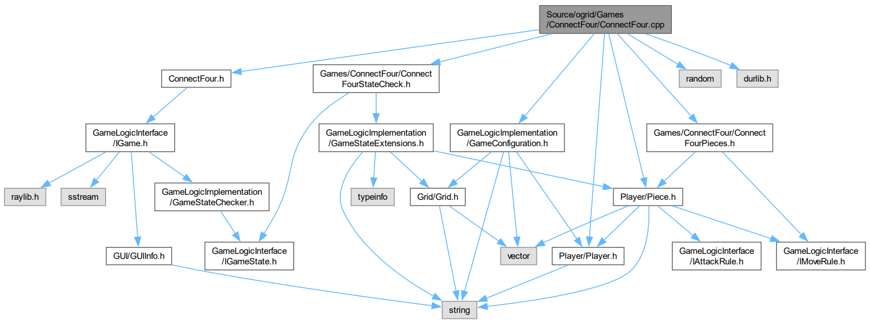
Go to the documentation of this file.

```
#pragma once

#include "GameLogicInterface/IGameState.h"
#include "GameLogicImplementation/GameStateExtensions.h"

namespace OGRID
{
    class CheckersStateCheck : public IGameState
    {
    private:
        GameStateExtensions m_GameStateExtensions = GameStateExtensions();

    public:
        int CheckWin(Grid *grid) const override;

        bool IsDraw(Grid *grid) const override;
    };
}
```

8.31 Source/ogrid/Games/ConnectFour/ConnectFour.cpp File Reference

```
#include "ConnectFour.h"
#include <random>
#include <durlib.h>
#include "GameLogicImplementation/GameConfiguration.h"
#include "Games/ConnectFour/ConnectFourPieces.h"
#include "Games/ConnectFour/ConnectFourStateCheck.h"
#include "Player/Piece.h"
```

`#include "Player/Player.h"`
Include dependency graph for ConnectFour.cpp:

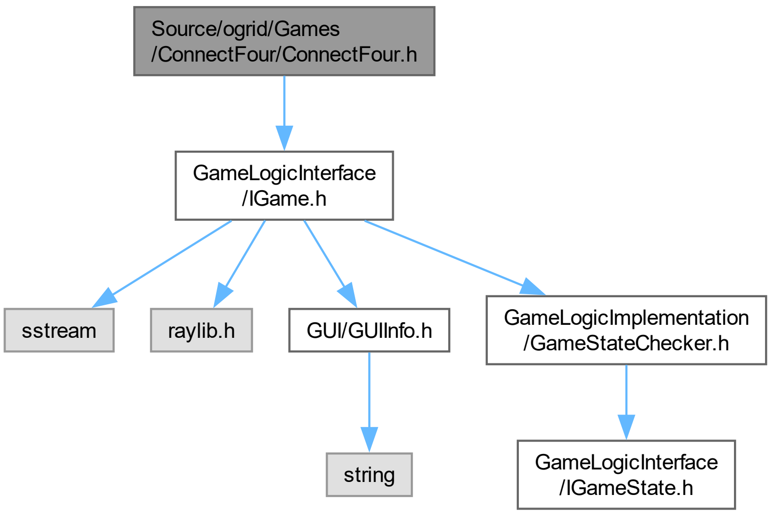

Source/ogrid/Games/ConnectFour/ConnectFour.cpp
ConnectFour.h
Games/ConnectFour/ConnectFourStateCheck.h
random
durlib.h
GameLogicInterface/IGame.h
GameLogicImplementation/GameStateExtensions.h
GameLogicImplementation/GameConfiguration.h
Games/ConnectFour/ConnectFourPieces.h
raylib.h
sstream
GameLogicImplementation/GameStateChecker.h
typeinfo
Grid/Grid.h
Player/Piece.h
GUI/GUIInfo.h
GameLogicInterface/IGameState.h
vector
Player/Player.h
GameLogicInterface/IAttackRule.h
GameLogicInterface/IMoveRule.h
string

Namespaces

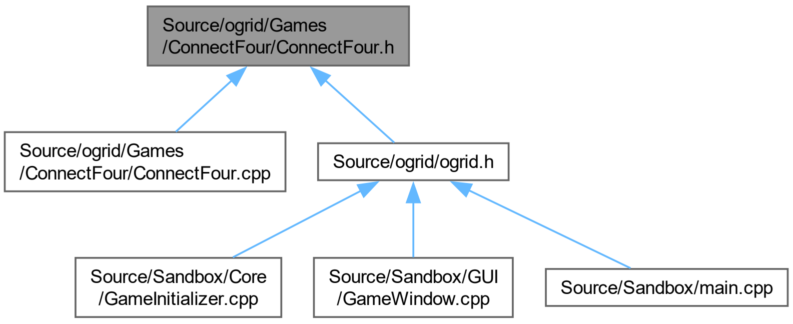
- namespace OGRID

8.32 Source/ogrid/Games/ConnectFour/ConnectFour.h File Reference

`#include "GameLogicInterface/IGame.h"`
Include dependency graph for ConnectFour.h:


Source/ogrid/Games/ConnectFour/ConnectFour.h
GameLogicInterface/IGame.h
sstream
raylib.h
GUI/GUIInfo.h
GameLogicImplementation/GameStateChecker.h
string
GameLogicInterface/IGameState.h

This graph shows which files directly or indirectly include this file:


Source/ogrid/Games /ConnectFour/ConnectFour.h
Source/ogrid/Games /ConnectFour/ConnectFour.cpp
Source/ogrid/ogrid.h
Source/Sandbox/Core /GameInitializer.cpp
Source/Sandbox/GUI /GameWindow.cpp
Source/Sandbox/main.cpp

Classes

- class OGRID::ConnectFour

Namespaces

- namespace OGRID

8.33 ConnectFour.h

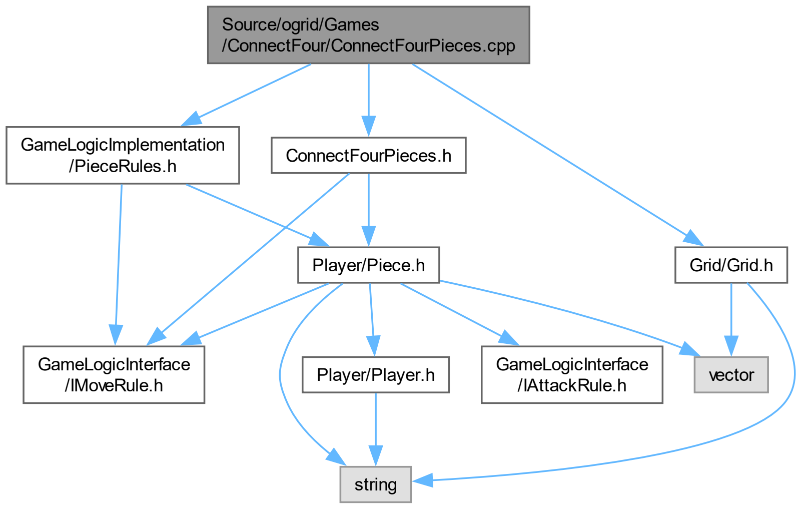
Go to the documentation of this file.

```
#pragma once

#include "GameLogicInterface/IGame.h"

namespace OGRID
{
    class ConnectFour : public IGame
    {
        // Variables for alpha of circles
        float alpha = 1.0f;
        // Speed of the transition
        float alphaSpeed = 0.025f;

    public:
        ConnectFour() = default;
        ~ConnectFour() = default;

        bool TryMakeMove(unsigned char &row, unsigned char &col) override;
        bool IsWinningCondition() override;
        bool IsDrawCondition() override;
        void SetupPlayers() override;

        void Initialize() override;
        void OnGUIUpdateGrid() override;
        void OnGUIUpdateGridHover(Vector2 cell) override;

    private:
        void DrawCircle(int row, int col, Color color, bool blinking = false);
    };
}
```

8.34 Source/ogrid/Games/ConnectFour/ConnectFourPieces.cpp File Reference

```
#include "ConnectFourPieces.h"
#include "Grid/Grid.h"
#include "GameLogicImplementation/PieceRules.h"
```

Include dependency graph for ConnectFourPieces.cpp:



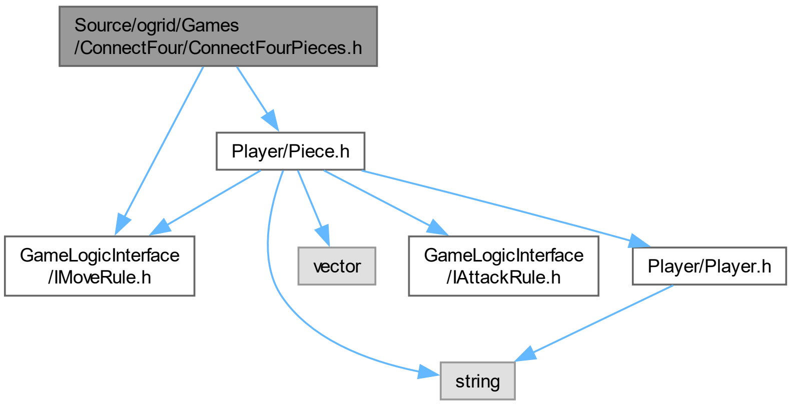
Namespaces

- namespace OGRID

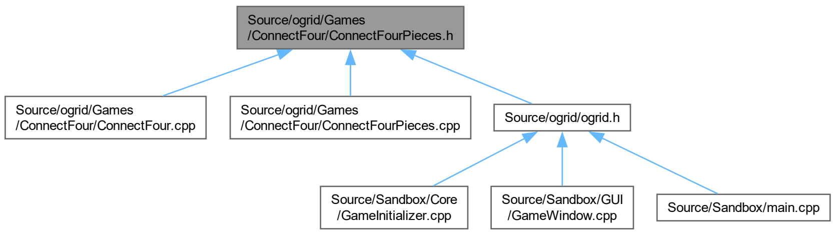
8.35 Source/ogrid/Games/ConnectFour/ConnectFourPieces.h File Reference

```
#include "GameLogicInterface/IMoveRule.h"
#include "Player/Piece.h"
```

Include dependency graph for ConnectFourPieces.h:



This graph shows which files directly or indirectly include this file:



Classes

- class OGRID::RedPiece
- class OGRID::BlackPiece

Namespaces

- namespace OGRID

8.36 ConnectFourPieces.h

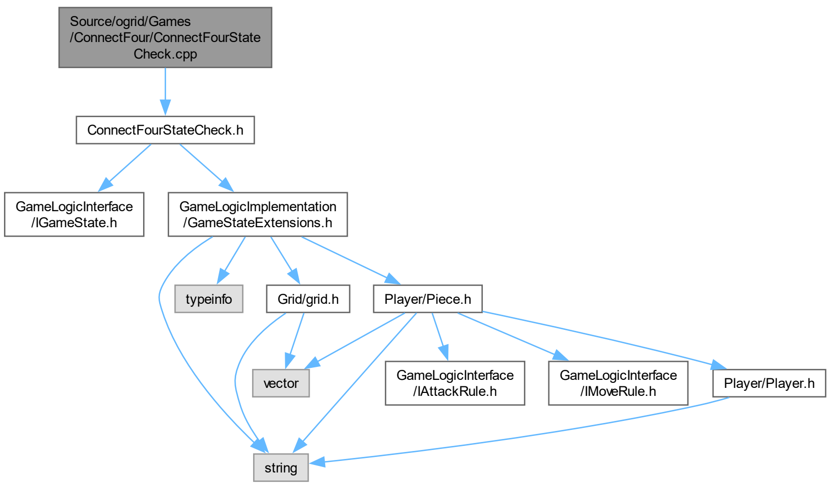
Go to the documentation of this file.

```
#pragma once

#include "GameLogicInterface/IMoveRule.h"
#include "Player/Piece.h"

namespace OGRID
{
    class Grid;

    class RedPiece : public Piece
    {
    public:
        RedPiece(Player *player);
    };

    class BlackPiece : public Piece
    {
    public:
        BlackPiece(Player *player);
    };
}
```

8.37 Source/ogrid/Games/ConnectFour/ConnectFourStateCheck.cpp File Reference

```
#include "ConnectFourStateCheck.h"
```

Include dependency graph for ConnectFourStateCheck.cpp:


Source/ogrid/Games/ConnectFour/ConnectFourStateCheck.cpp
ConnectFourStateCheck.h
GameLogicInterface/IGameState.h
GameLogicImplementation/GameStateExtensions.h
typeinfo
Grid/grid.h
Player/Piece.h
vector
GameLogicInterface/IAttackRule.h
GameLogicInterface/IMoveRule.h
Player/Player.h
string

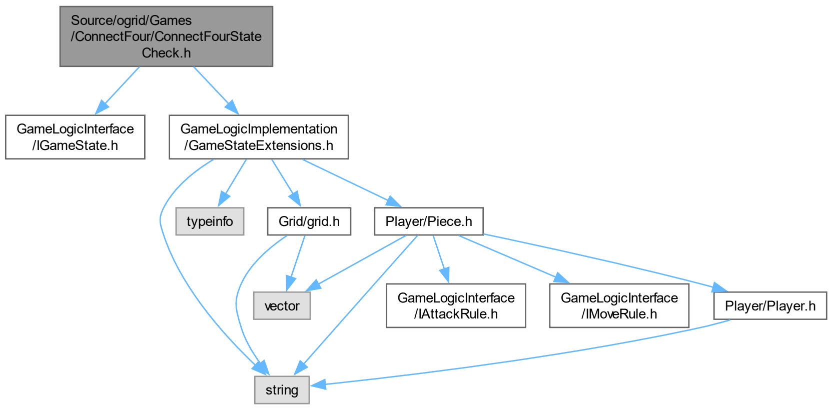
Namespaces

- namespace OGRID

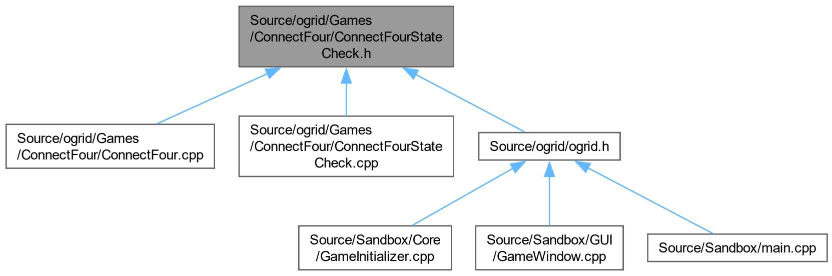
8.38 Source/ogrid/Games/ConnectFour/ConnectFourStateCheck.h File Reference

```
#include "GameLogicInterface/IGameState.h"
#include "GameLogicImplementation/GameStateExtensions.h"
```

Include dependency graph for ConnectFourStateCheck.h:


Source/ogrid/Games /ConnectFour/ConnectFourState Check.h
GameLogicInterface /IGameState.h
GameLogicImplementation /GameStateExtensions.h
typeinfo
Grid/grid.h
Player/Piece.h
vector
GameLogicInterface /IAttackRule.h
GameLogicInterface /IMoveRule.h
Player/Player.h
string

This graph shows which files directly or indirectly include this file:


Source/ogrid/Games /ConnectFour/ConnectFourState Check.h
Source/ogrid/Games /ConnectFour/ConnectFour.cpp
Source/ogrid/Games /ConnectFour/ConnectFourState Check.cpp
Source/ogrid/ogrid.h
Source/Sandbox/Core /GameInitializer.cpp
Source/Sandbox/GUI /GameWindow.cpp
Source/Sandbox/main.cpp

Classes

- class OGRID::ConnectFourStateCheck

Namespaces

- namespace OGRID

8.39 ConnectFourStateCheck.h

Go to the documentation of this file.

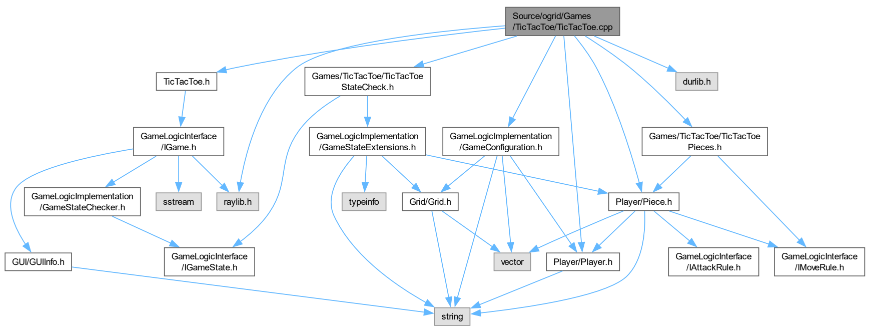
```
#pragma once

#include "GameLogicInterface/IGameState.h"
#include "GameLogicImplementation/GameStateExtensions.h"

namespace OGRID
{
    class ConnectFourStateCheck : public IGameState
    {
```

```
    private:
        GameStateExtensions m_GameStateExtensions = GameStateExtensions();

    public:
        int CheckWin(Grid *grid) const override;

        bool IsDraw(Grid *grid) const override;
    };
}
```

8.40 Source/ogrid/Games/TicTacToe/TicTacToe.cpp File Reference

```
#include "TicTacToe.h"
#include <durlib.h>
#include <raylib.h>
#include "GameLogicImplementation/GameConfiguration.h"
#include "Games/TicTacToe/TicTacToePieces.h"
#include "Games/TicTacToe/TicTacToeStateCheck.h"
#include "Player/Piece.h"
#include "Player/Player.h"
```

Include dependency graph for TicTacToe.cpp:


Source/ogrid/Games /TicTacToe/TicTacToe.cpp
TicTacToe.h
Games/TicTacToe/TicTacToe StateCheck.h
durlib.h
GameLogicInterface /IGame.h
GameLogicImplementation /GameStateExtensions.h
GameLogicImplementation /GameConfiguration.h
Games/TicTacToe/TicTacToe Pieces.h
GameLogicImplementation /GameStateChecker.h
sstream
raylib.h
typeinfo
Grid/Grid.h
Player/Piece.h
GUI/GUIInfo.h
GameLogicInterface /IGameState.h
vector
Player/Player.h
GameLogicInterface /IAttackRule.h
GameLogicInterface /IMoveRule.h
string

Namespaces

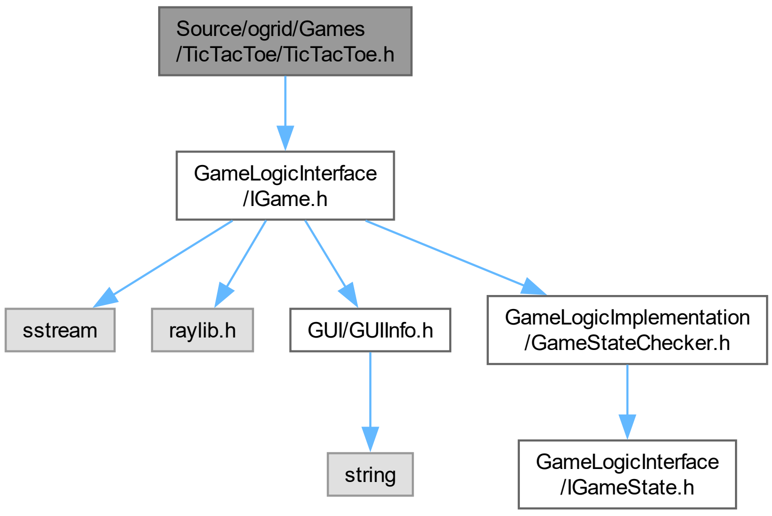
- namespace OGRID

8.41 Source/ogrid/Games/TicTacToe/TicTacToe.h File Reference

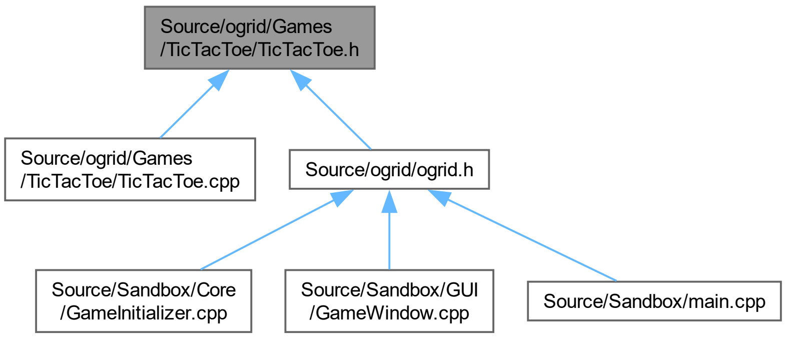
TicTacToe game logic.

```
#include "GameLogicInterface/IGame.h"
```

Include dependency graph for TicTacToe.h:



This graph shows which files directly or indirectly include this file:



Classes

- class OGRID::TicTacToe

  TicTacToe game logic.

Namespaces

- namespace OGRID

8.41.1 Detailed Description

TicTacToe game logic.

Date

2023-12-06

See also

https://en.wikipedia.org/wiki/Tic-tac-toe

8.42 TicTacToe.h

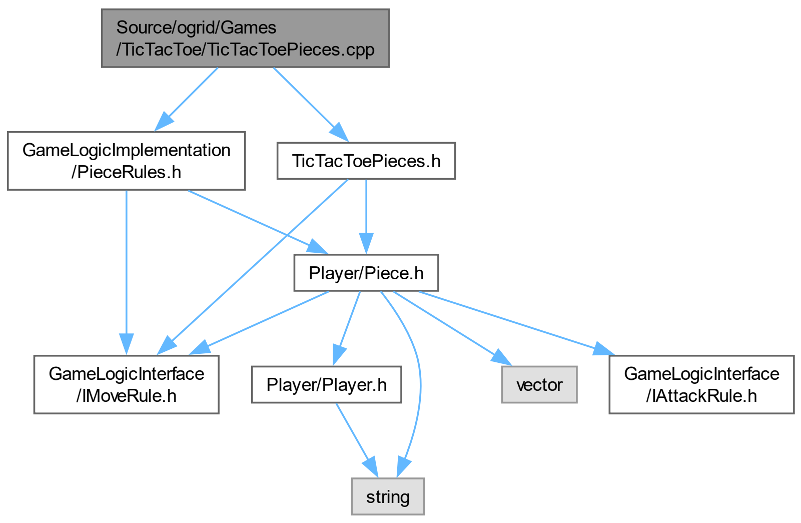
Go to the documentation of this file.

```
#pragma once

#include "GameLogicInterface/IGame.h"

namespace OGRID
{
    class TicTacToe : public IGame
    {
    public:
        TicTacToe() = default;

        ~TicTacToe() = default;

        bool TryMakeMove(unsigned char &row, unsigned char &col) override;

        bool IsWinningCondition() override;

        bool IsDrawCondition() override;
        // bool IsWinningCondition(unsigned char row, unsigned char col) override;
        // bool IsWinningCondition(char playerChar) override;
        // bool IsDrawCondition(unsigned char row, unsigned char col) override;
        // void SetupPlayers(const std::vector<int> &totalValidSides) override;

        void SetupPlayers() override;

        void Initialize() override;

        void OnGUIUpdateGrid() override;

        void OnGUIUpdateGridHover(Vector2 cell) override;

    private:
        void DrawX(int row, int col);

        void DrawO(int row, int col);
    };
}
```

8.43 Source/ogrid/Games/TicTacToe/TicTacToePieces.cpp File Reference

```
#include "TicTacToePieces.h"
#include "GameLogicImplementation/PieceRules.h"
```

Include dependency graph for TicTacToePieces.cpp:



Namespaces

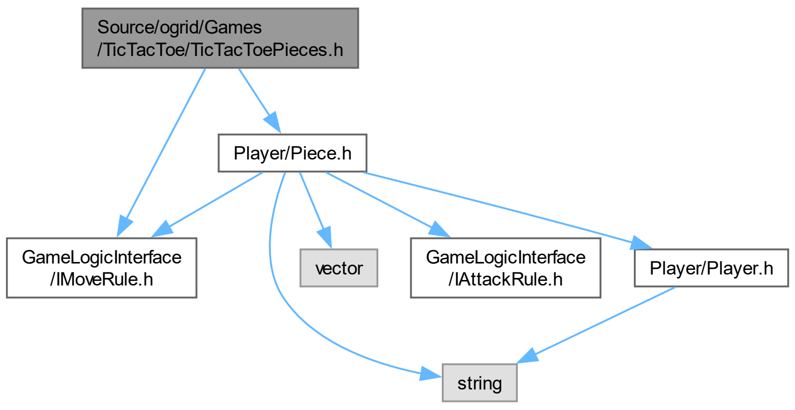
- namespace OGRID

8.44 Source/ogrid/Games/TicTacToe/TicTacToePieces.h File Reference

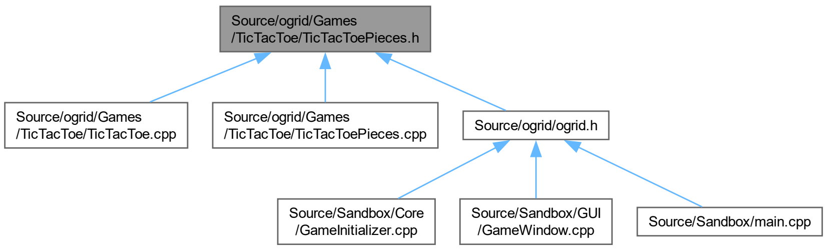
TicTacToe pieces.

```
#include "GameLogicInterface/IMoveRule.h"
#include "Player/Piece.h"
```

Include dependency graph for TicTacToePieces.h:



This graph shows which files directly or indirectly include this file:


Source/ogrid/Games /TicTacToe/TicTacToePieces.h
Source/ogrid/Games /TicTacToe/TicTacToe.cpp
Source/ogrid/Games /TicTacToe/TicTacToePieces.cpp
Source/ogrid/ogrid.h
Source/Sandbox/Core /GameInitializer.cpp
Source/Sandbox/GUI /GameWindow.cpp
Source/Sandbox/main.cpp

Classes

- class OGRID::XPiece

  TicTacToe X piece.
- class OGRID::OPiece

  TicTacToe O piece.

Namespaces

- namespace OGRID

8.44.1 Detailed Description

TicTacToe pieces.

Date

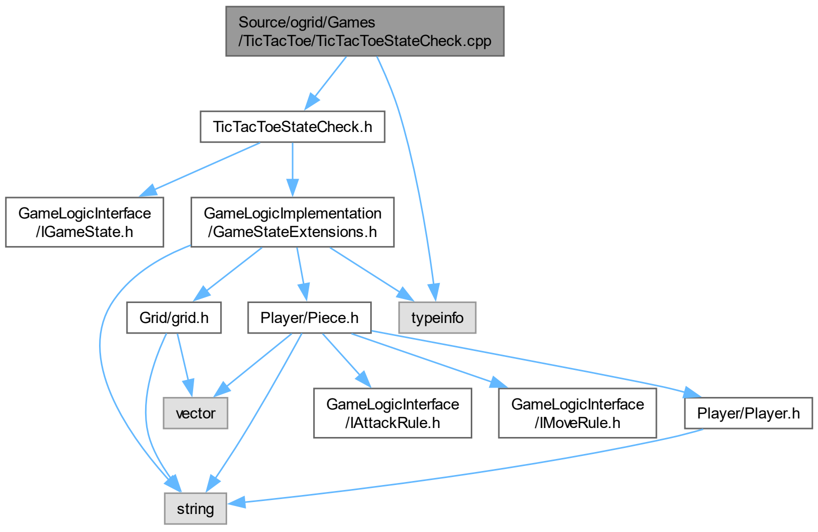
2023-12-06

8.45 TicTacToePieces.h

Go to the documentation of this file.
```
#pragma once

#include "GameLogicInterface/IMoveRule.h"
#include "Player/Piece.h"

// TODO: It might be better to define specific exception instead of returning booleans, except for the
case of invalid moves.

namespace OGRID
{
    class Grid;

    class XPiece : public Piece
    {
    public:
        XPiece(Player *player);
    };

    class OPiece : public Piece
    {
    public:
        OPiece(Player *player);
    };
}
```

8.46 Source/ogrid/Games/TicTacToe/TicTacToeStateCheck.cpp File Reference

```
#include "TicTacToeStateCheck.h"
#include <typeinfo>
```
Include dependency graph for TicTacToeStateCheck.cpp:



Namespaces

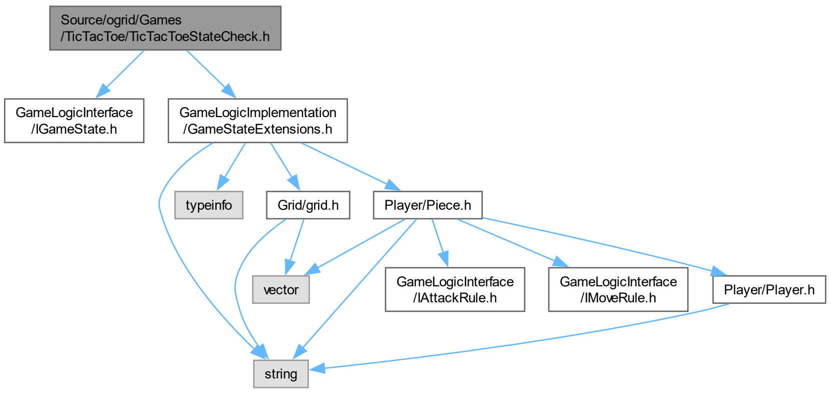
- namespace OGRID

8.47 Source/ogrid/Games/TicTacToe/TicTacToeStateCheck.h File Reference

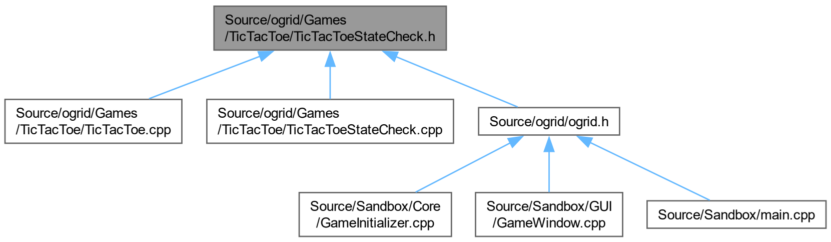
TicTacToe state check.

```
#include "GameLogicInterface/IGameState.h"
#include "GameLogicImplementation/GameStateExtensions.h"
```

Include dependency graph for TicTacToeStateCheck.h:



This graph shows which files directly or indirectly include this file:



Classes

- class OGRID::TicTacToeStateCheck

    TicTacToe state check.

Namespaces

- namespace OGRID

8.47.1 Detailed Description

TicTacToe state check.

Date

2023-12-06

8.48 TicTacToeStateCheck.h

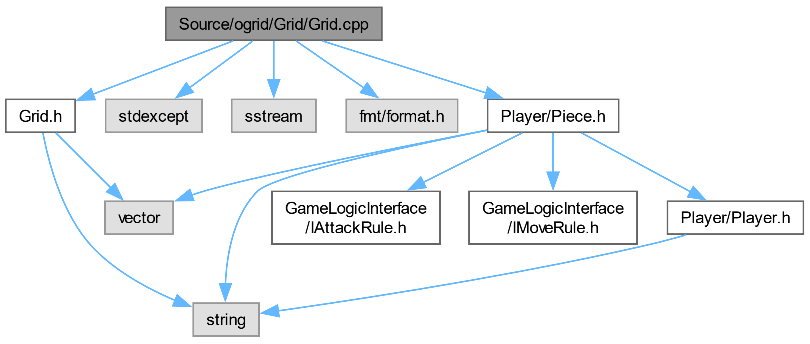
Go to the documentation of this file.

```
#pragma once

#include "GameLogicInterface/IGameState.h"
#include "GameLogicImplementation/GameStateExtensions.h"

namespace OGRID
{
    class TicTacToeStateCheck : public IGameState
    {
    private:
        GameStateExtensions m_GameStateExtensions = GameStateExtensions();

    public:
        int CheckWin(Grid *grid) const override;

        bool IsDraw(Grid *grid) const override;
    };
}
```

8.49 Source/ogrid/Grid/Grid.cpp File Reference

```
#include "Grid.h"
#include <stdexcept>
#include <sstream>
#include "fmt/format.h"
#include "Player/Piece.h"
```

Include dependency graph for Grid.cpp:


Source/ogrid/Grid/Grid.cpp
Grid.h
stdexcept
sstream
fmt/format.h
Player/Piece.h
vector
GameLogicInterface/IAttackRule.h
GameLogicInterface/IMoveRule.h
Player/Player.h
string

Namespaces

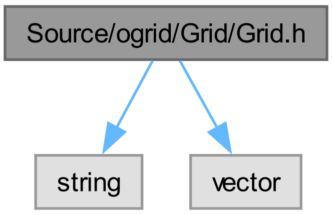
- namespace OGRID

8.50 Source/ogrid/Grid/Grid.h File Reference

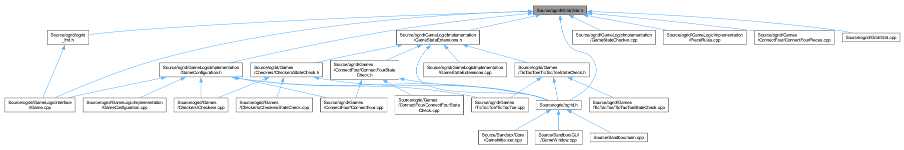
Contains the Grid class.

```
#include <string>
#include <vector>
```

Include dependency graph for Grid.h:



This graph shows which files directly or indirectly include this file:


Source/ogrid/Grid/Grid.h
Source/ogrid/ogrid _fmt.h
Source/ogrid/GameLogicImplementation /GameStateExtensions.h
Source/ogrid/GameLogicImplementation /GameStateChecker.cpp
Source/ogrid/GameLogicImplementation /PieceRules.cpp
Source/ogrid/Games /ConnectFour/ConnectFourPieces.cpp
Source/ogrid/Grid/Grid.cpp
Source/ogrid/GameLogicImplementation /GameConfiguration.h
Source/ogrid/Games /Checkers/CheckersStateCheck.h
Source/ogrid/Games /ConnectFour/ConnectFourState Check.h
Source/ogrid/GameLogicImplementation /GameStateExtensions.cpp
Source/ogrid/Games /TicTacToe/TicTacToeStateCheck.h
Source/ogrid/GameLogicInterface /IGame.cpp
Source/ogrid/GameLogicImplementation /GameConfiguration.cpp
Source/ogrid/Games /Checkers/Checkers.cpp
Source/ogrid/Games /Checkers/CheckersStateCheck.cpp
Source/ogrid/Games /ConnectFour/ConnectFour.cpp
Source/ogrid/Games /ConnectFour/ConnectFourState Check.cpp
Source/ogrid/Games /TicTacToe/TicTacToe.cpp
Source/ogrid/ogrid.h
Source/ogrid/Games /TicTacToe/TicTacToeStateCheck.cpp
Source/Sandbox/Core /GameInitializer.cpp
Source/Sandbox/GUI /GameWindow.cpp
Source/Sandbox/main.cpp

Classes

- struct OGRID::Cell

    The Cell struct represents a single cell in the grid.
- class OGRID::Grid

    The Grid class represents a 2D grid of Cells.

Namespaces

- namespace OGRID

8.50.1 Detailed Description

Contains the Grid class.

Date

2023-12-06

8.51 Grid.h

Go to the documentation of this file.

```
#pragma once

#include <string>
#include <vector>

namespace OGRID
{
    class Piece;

    struct Cell
    {
        Piece *m_Piece;
        unsigned char m_Row;
        unsigned char m_Col;
    };

    class Grid
    {
    private:
        unsigned char rows;

        unsigned char cols;

        std::vector<std::vector<Cell *» grid;

        // Store default Piece for resetting the grid.
        // Important! This will set the whole grid to the same Piece.
        // Careful when setting this to nullptr.
        Piece *defaultPiece;

        // Store which element was last changed.
        unsigned char lastChangedChar[2] = {0, 0};

        // Constructors & Destructors
    public:
        Grid(unsigned char rows, unsigned char cols, Piece *defaultPiece = nullptr);

        ~Grid();

        // Getters & Setters
    public:
        unsigned char GetRows() const;

        void SetRows(unsigned char rows);

        unsigned char GetCols() const;

        void SetCols(unsigned char cols);

        const std::vector<std::vector<Cell *» &GetGrid() const;

        void SetGrid(const std::vector<std::vector<Cell *» &newGrid);

        Piece *GetDefaultPiece() const;

        void SetDefaultPiece(Piece *defaultPiece);

        Piece *GetPieceAt(unsigned char row, unsigned char col) const;

        void SetPieceAt(unsigned char row, unsigned char col, Piece *piece, bool force_null = false);

        Cell *GetCellAt(unsigned char row, unsigned char col) const;

        void SetCellAt(unsigned char row, unsigned char col, Cell *cell, bool force_null = false);

        void SetCellAt(unsigned char row, unsigned char col, Piece *piece, bool force_null = false);

        std::pair<unsigned char, unsigned char> GetLastChangedChar() const;

    public:
        // Overload the [] operator to access the grid.
        std::vector<Cell *> &operator[](size_t index);

        const std::vector<Cell *> &operator[](size_t index) const;

        // Public methods
    public:
        const std::string GetGridSize() const;

        void ResetGrid();

        void ResetGridWithNewSize(unsigned char newRows, unsigned char newCols, Piece *defaultPiece =
nullptr);
```

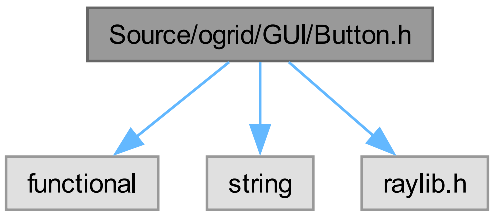
```
        void ResetGridWithNewDefaultPiece(Piece *defaultPiece = nullptr);

        std::string GetGridAsString();

        // bool CheckForRecurringStringInRow(const std::string &playerString, unsigned int dupCount);
        // bool CheckForRecurringStringInCol(const std::string &playerString, unsigned int dupCount);
        // bool CheckForRecurringStringInDiagonal(const std::string &playerString, unsigned int
dupCount);
        // bool CheckForRecurringStringInAntiDiagonal(const std::string &playerString, unsigned int
dupCount);

        // // This one is broken and should probably be removed...
        // std::string GetCharCenterMostElement() const;
        // std::pair<unsigned char, unsigned char> GetCenterMostCoords() const;
    };
}
```

8.52 Source/ogrid/GUI/Button.h File Reference

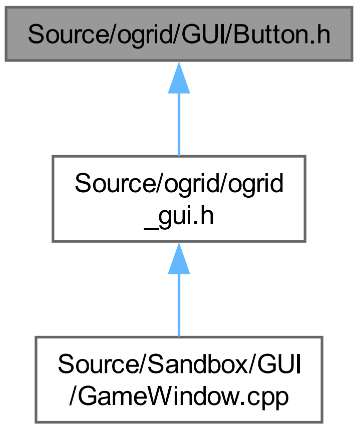
Button.

```
#include <functional>
#include <string>
#include <raylib.h>
```

Include dependency graph for Button.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct OGRID::Button

    Button.

Namespaces

- namespace OGRID

8.52.1 Detailed Description

Button.

Date

2023-12-06

See also

`https://www.raylib.com/`

8.53 Button.h

Go to the documentation of this file.

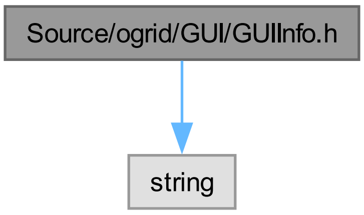
```cpp
#pragma once

#include <functional>
#include <string>

#include <raylib.h>

namespace OGRID
{
    struct Button
    {
        // Position and dimensions of the button
        Rectangle bounds;

        // Color of the button when not interacted with
        Color normalColor;

        // Color of the button on mouse hover
        Color hoverColor;

        // Color of the button when pressed
        Color pressedColor;

        // Delegate function for button click event
        std::function<void()> onClick;

        // Text to be displayed on the button
        std::string text;

        // Flag to enable or disable the button’s click functionality
        bool isEnabled;

        // Modify constructor to initialize isEnabled
        Button(Rectangle bounds, Color normal, Color hover, Color pressed, std::function<void()> clickCallback, std::string text = "Button", bool isEnabled = true)
            : bounds(bounds), normalColor(normal), hoverColor(hover), pressedColor(pressed), onClick(clickCallback), text(text), isEnabled(isEnabled) {}

        // Check if the button is hovered or clicked
        void Update()
        {
            if (isEnabled)
            {
                Vector2 mousePoint = GetMousePosition();
                if (CheckCollisionPointRec(mousePoint, bounds))
                {
                    if (IsMouseButtonReleased(MOUSE_LEFT_BUTTON))
                    {
                        // Call the delegate function if the button is enabled
                        onClick();
                    }
                }
            }
        }

        void Draw() const
        {
            Color currentColor = isEnabled ? normalColor : GRAY;

            if (CheckCollisionPointRec(GetMousePosition(), bounds))
            {
                currentColor = IsMouseButtonDown(MOUSE_LEFT_BUTTON) ? pressedColor : hoverColor;
            }
            DrawRectangleRec(bounds, currentColor);

            // Measure the text width and height
            int fontSize = 20;
            Vector2 textSize = MeasureTextEx(GetFontDefault(), text.c_str(), fontSize, 1);

            // Calculate text position to center it on the button
            float textX = bounds.x + (bounds.width - textSize.x) / 2;
            float textY = bounds.y + (bounds.height - textSize.y) / 2 - 10;

            // Adjust Y position to align text vertically in the middle
            textY += (textSize.y / 2);

            // Draw the text centered
            DrawText(text.c_str(), static_cast<int>(textX), static_cast<int>(textY), fontSize, WHITE);
        }

        void SetEnabled(bool enabled)
        {
            isEnabled = enabled;
        }
    };
}
```

8.54 Source/ogrid/GUI/GUIInfo.h File Reference

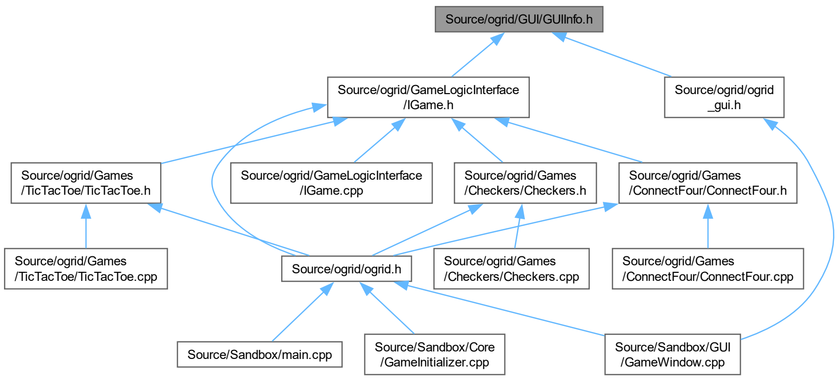
GUI info.

`#include <string>`

Include dependency graph for GUIInfo.h:



This graph shows which files directly or indirectly include this file:


Source/ogrid/GUI/GUIInfo.h
Source/ogrid/GameLogicInterface/IGame.h
Source/ogrid/ogrid_gui.h
Source/ogrid/Games/TicTacToe/TicTacToe.h
Source/ogrid/GameLogicInterface/IGame.cpp
Source/ogrid/Games/Checkers/Checkers.h
Source/ogrid/Games/ConnectFour/ConnectFour.h
Source/ogrid/Games/TicTacToe/TicTacToe.cpp
Source/ogrid/ogrid.h
Source/ogrid/Games/Checkers/Checkers.cpp
Source/ogrid/Games/ConnectFour/ConnectFour.cpp
Source/Sandbox/main.cpp
Source/Sandbox/Core/GameInitializer.cpp
Source/Sandbox/GUI/GameWindow.cpp

Classes

- struct GUIInfo

  GUI info.

8.54.1 Detailed Description

GUI info.

Date

2023-12-06

8.55 GUIInfo.h

Go to the documentation of this file.

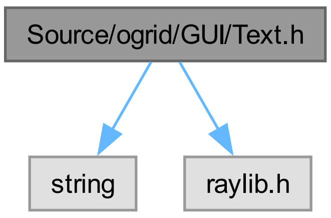
```
#pragma once

#include <string>

struct GUIInfo
{
    int width;

    int height;

    std::string windowName;

    int targetFPS;

    float cellSize;

    float lineThickness;

    float margin;
};
```

8.56 Source/ogrid/GUI/Text.h File Reference

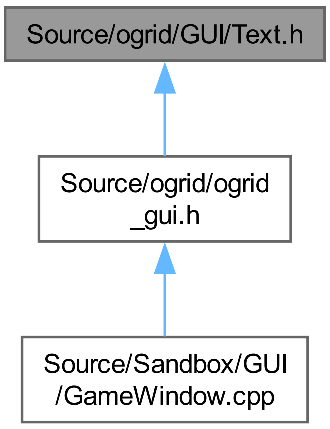
Text.

```
#include <string>
#include <raylib.h>
```

Include dependency graph for Text.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct OGRID::Text

  Text.

Namespaces

- namespace OGRID

Enumerations

- enum class OGRID::Justify { OGRID::NONE , OGRID::CENTER_X , OGRID::CENTER_Y , OGRID::CENTER_BOTH }

  Justify the text.

8.56.1 Detailed Description

Text.

Date

2023-12-06

See also

`https://www.raylib.com/`

8.57 Text.h

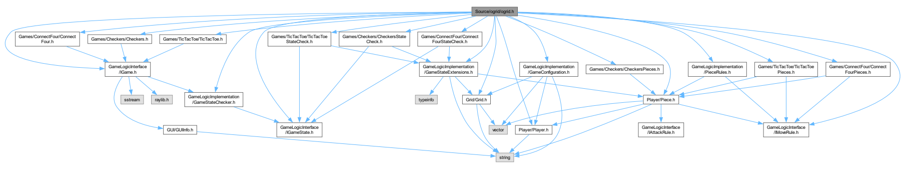
Go to the documentation of this file.

```
#pragma once

#include <string>

#include <raylib.h>

namespace OGRID
{
    // Enumeration to specify text justification options
    enum class Justify
    {
        // No alignment, use x and y as is
        NONE,
        // Center the text horizontally on the screen
        CENTER_X,
        // Center the text vertically on the screen
        CENTER_Y,
        // Center the text both horizontally and vertically
        CENTER_BOTH
    };

    struct Text
    {
        std::string text;

        int fontSize;

        int x;

        int y;

        int screenWidth;

        int screenHeight;

        Color color;

        Justify justify;

        Text(std::string text, int fontSize, int x, int y, Color color, Justify justify = Justify::NONE, int screenWidth = 0, int screenHeight = 0)
            : text(text), fontSize(fontSize), x(x), y(y), color(color), justify(justify), screenWidth(screenWidth), screenHeight(screenHeight) {}

        void Draw() const
        {
            int textX = x;
            int textY = y;

            // Only calculate text size if we need to justify it
            if (justify != Justify::NONE)
            {
                Vector2 textSize = MeasureTextEx(GetFontDefault(), text.c_str(), fontSize, 1);

                if (justify == Justify::CENTER_X || justify == Justify::CENTER_BOTH)
                {
                    textX = screenWidth / 2 - (textSize.x / 2);
                }

                if (justify == Justify::CENTER_Y || justify == Justify::CENTER_BOTH)
                {
                    textY = screenHeight / 2 - (textSize.y / 2);
                }
            }

            DrawText(text.c_str(), textX, textY, fontSize, color);
        }

        void SetText(std::string text)
        {
            this->text = text;
        }

        void SetScreenSize(int width, int height)
        {
            screenWidth = width;
            screenHeight = height;
        }

        void SetJustification(Justify newJustify)
        {
            justify = newJustify;
00164         }
00165     };
00166 }
```

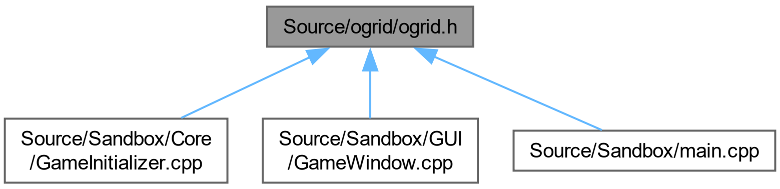
8.58 Source/ogrid/ogrid.h File Reference

```
#include "Player/Player.h"
#include "Player/Piece.h"
#include "Grid/Grid.h"
#include "GameLogicInterface/IGameState.h"
#include "GameLogicInterface/IMoveRule.h"
#include "GameLogicInterface/IGame.h"
#include "GameLogicImplementation/GameStateExtensions.h"
#include "GameLogicImplementation/GameStateChecker.h"
#include "GameLogicImplementation/GameConfiguration.h"
#include "GameLogicImplementation/PieceRules.h"
#include "Games/TicTacToe/TicTacToeStateCheck.h"
#include "Games/TicTacToe/TicTacToePieces.h"
#include "Games/TicTacToe/TicTacToe.h"
#include "Games/ConnectFour/ConnectFourStateCheck.h"
#include "Games/ConnectFour/ConnectFourPieces.h"
#include "Games/ConnectFour/ConnectFour.h"
#include "Games/Checkers/CheckersStateCheck.h"
#include "Games/Checkers/CheckersPieces.h"
#include "Games/Checkers/Checkers.h"
```

Include dependency graph for ogrid.h:


Source/ogrid/ogrid.h
Games/ConnectFour/Connect Four.h
Games/Checkers/Checkers.h
Games/TicTacToe/TicTacToe.h
Games/TicTacToe/TicTacToe StateCheck.h
Games/Checkers/CheckersState Check.h
Games/ConnectFour/Connect FourStateCheck.h
GameLogicInterface /IGame.h
GameLogicImplementation /GameStateExtensions.h
GameLogicImplementation /GameConfiguration.h
Games/Checkers/CheckersPieces.h
GameLogicImplementation /PieceRules.h
Games/TicTacToe/TicTacToe Pieces.h
Games/ConnectFour/Connect FourPieces.h
sstream
raylib.h
GameLogicImplementation /GameStateChecker.h
typeinfo
Grid/Grid.h
Player/Piece.h
GUI/GUIInfo.h
GameLogicInterface /IGameState.h
vector
Player/Player.h
GameLogicInterface /IAttackRule.h
GameLogicInterface /IMoveRule.h
string

This graph shows which files directly or indirectly include this file:


Source/ogrid/ogrid.h
Source/Sandbox/Core /GameInitializer.cpp
Source/Sandbox/GUI /GameWindow.cpp
Source/Sandbox/main.cpp

8.59 ogrid.h

Go to the documentation of this file.

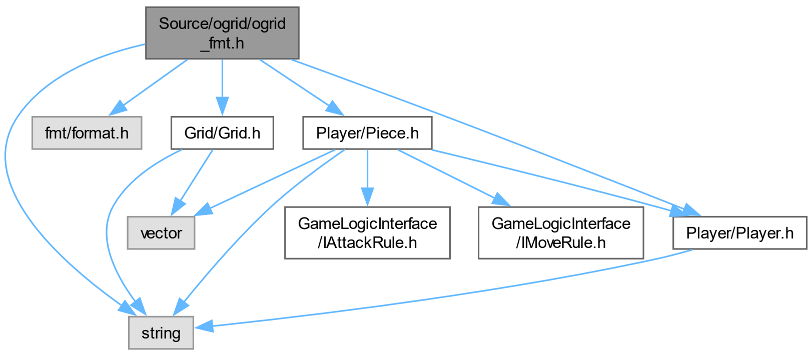
```
#pragma once

// Core
#include "Player/Player.h"
#include "Player/Piece.h"
#include "Grid/Grid.h"

// Interface
#include "GameLogicInterface/IGameState.h"
#include "GameLogicInterface/IMoveRule.h"
#include "GameLogicInterface/IGame.h"

// Implementation
#include "GameLogicImplementation/GameStateExtensions.h"
#include "GameLogicImplementation/GameStateChecker.h"
#include "GameLogicImplementation/GameConfiguration.h"
#include "GameLogicImplementation/PieceRules.h"

// Games
// Tic Tac Toe
#include "Games/TicTacToe/TicTacToeStateCheck.h"
#include "Games/TicTacToe/TicTacToePieces.h"
#include "Games/TicTacToe/TicTacToe.h"
// Connect Four
#include "Games/ConnectFour/ConnectFourStateCheck.h"
#include "Games/ConnectFour/ConnectFourPieces.h"
#include "Games/ConnectFour/ConnectFour.h"
// Checkers
#include "Games/Checkers/CheckersStateCheck.h"
#include "Games/Checkers/CheckersPieces.h"
#include "Games/Checkers/Checkers.h"
```

8.60 Source/ogrid/ogrid_fmt.h File Reference

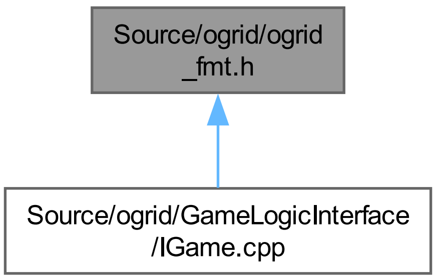
This is used for fmt formatting from the OGRID namespace.

```
#include <string>
#include "fmt/format.h"
#include "Grid/Grid.h"
#include "Player/Player.h"
#include "Player/Piece.h"
```

Include dependency graph for ogrid_fmt.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct fmt::formatter< OGRID::Grid >

  This is used to format a Grid object into a string using fmt.
- struct fmt::formatter< OGRID::PlayerType >

  This is used to format a PlayerType enum into a string using fmt.
- struct fmt::formatter< OGRID::Player >

  This is used to format a Player object into a string using fmt.

8.60.1 Detailed Description

This is used for fmt formatting from the OGRID namespace.

This is put in one file to avoid circular dependencies.

Date

2023-12-06

8.61 ogrid_fmt.h

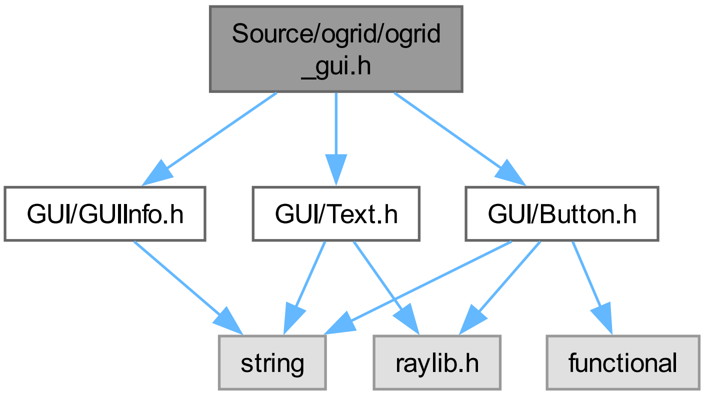
Go to the documentation of this file.

```
#pragma once

#include <string>

#include "fmt/format.h"

#include "Grid/Grid.h"
#include "Player/Player.h"
#include "Player/Piece.h"
// Formatting for fmt library.

// Grid formatting
template <>
struct fmt::formatter<OGRID::Grid> : fmt::formatter<std::string>
{
    // Parses format specifications of the form '[:...]' which you can ignore.
    constexpr auto parse(format_parse_context &ctx) { return ctx.begin(); }

    // Formats the Grid using provided format specifiers.
    template <typename FormatContext>
    auto format(const OGRID::Grid &grid, FormatContext &ctx)
    {
        // Use a memory buffer to store the temporary output.
        fmt::memory_buffer buf;

        fmt::format_to(std::back_inserter(buf), "\n");

        for (int i = 0; i < grid.GetRows(); i++)
        {
            for (int j = 0; j < grid.GetCols(); j++)
            {
                OGRID::Cell *cell = grid.GetGrid()[i][j];

                // Assuming you want a space between characters in a row.
                if (j > 0)
                    fmt::format_to(std::back_inserter(buf), " ");
                // Check if the cell pointer is not null and then access m_Piece
                if (cell != nullptr && cell->m_Piece != nullptr)
                    fmt::format_to(std::back_inserter(buf), "{}", cell->m_Piece->GetRepresentation());
                else
                    fmt::format_to(std::back_inserter(buf), " ");
                // fmt::format_to(std::back_inserter(buf), "{}", grid.GetGrid()[i][j]);
            }
            // Add a newline after each row, except the last one.
            if (i < grid.GetRows() - 1)
                fmt::format_to(std::back_inserter(buf), "\n");
        }

        // Output the buffer to the formatting context and return the iterator.
        return fmt::format_to(ctx.out(), "{}", to_string(buf));
    }
};

// Player formatting
template <>
struct fmt::formatter<OGRID::PlayerType> : formatter<std::string>
{
    template <typename FormatContext>
    auto format(OGRID::PlayerType p, FormatContext &ctx)
    {
        std::string name = p == OGRID::PlayerType::Human ? "Human" : "AI";
        return formatter<std::string>::format(name, ctx);
    }
};

template <>
struct fmt::formatter<OGRID::Player> : fmt::formatter<std::string>
{
    // Parses format specifications of the form '[:...]' which you can ignore.
    constexpr auto parse(format_parse_context &ctx) { return ctx.begin(); }

    // Formats the Player using provided format specifiers.
    template <typename FormatContext>
    auto format(const OGRID::Player &player, FormatContext &ctx)
    {
        // Use a memory buffer to store the temporary output.
        fmt::memory_buffer buf;

        fmt::format_to(std::back_inserter(buf), "{} | {}", player.GetPlayerName(),
player.GetPlayerType());

        // Output the buffer to the formatting context and return the iterator.
        return fmt::format_to(ctx.out(), "{}", to_string(buf));
    }
};
```

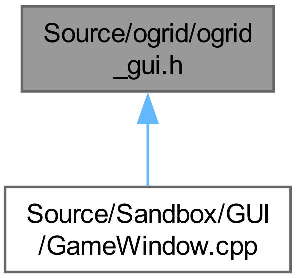
8.62 Source/ogrid/ogrid_gui.h File Reference

```
#include "GUI/GUIInfo.h"
#include "GUI/Button.h"
#include "GUI/Text.h"
```

Include dependency graph for ogrid_gui.h:



This graph shows which files directly or indirectly include this file:



8.63 ogrid_gui.h

Go to the documentation of this file.

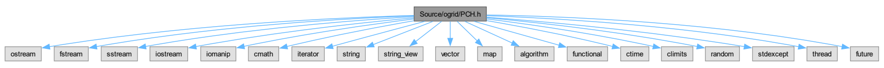
```
#pragma once

#include "GUI/GUIInfo.h"
#include "GUI/Button.h"
#include "GUI/Text.h"
```

8.64 Source/ogrid/PCH.h File Reference

```
#include <ostream>
#include <fstream>
```

```
#include <sstream>
#include <iostream>
#include <iomanip>
#include <cmath>
#include <iterator>
#include <string>
#include <string_view>
#include <vector>
#include <map>
#include <algorithm>
#include <functional>
#include <ctime>
#include <climits>
#include <random>
#include <stdexcept>
#include <thread>
#include <future>
```

Include dependency graph for PCH.h:



8.65 PCH.h

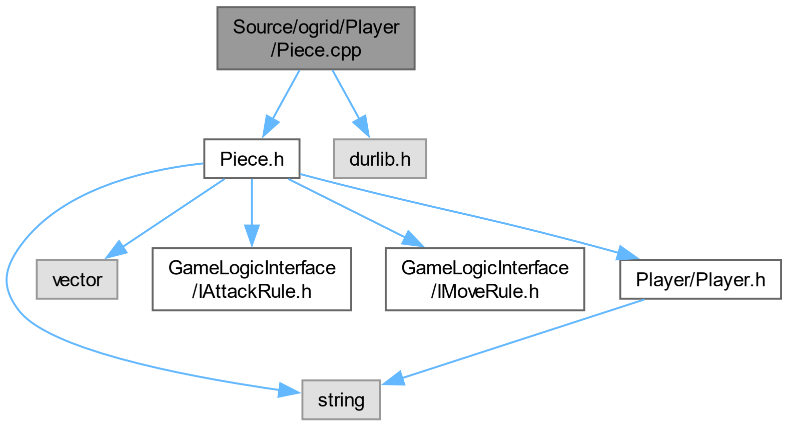
Go to the documentation of this file.

```
#pragma once

#include <ostream>
#include <fstream>
#include <sstream>
#include <iostream>
#include <iomanip>
#include <cmath>
#include <iterator>
#include <string>
#include <string_view>
#include <vector>
#include <map>
#include <algorithm>
#include <functional>
#include <ctime>
#include <climits>
#include <random>
#include <stdexcept>
#include <thread>
#include <future>
```

8.66 Source/ogrid/Player/Piece.cpp File Reference

```
#include "Piece.h"
#include "durlib.h"
```

Include dependency graph for Piece.cpp:



Namespaces

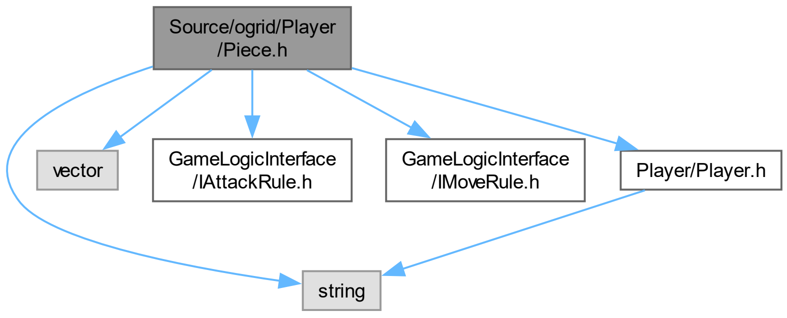
- namespace OGRID

8.67 Source/ogrid/Player/Piece.h File Reference

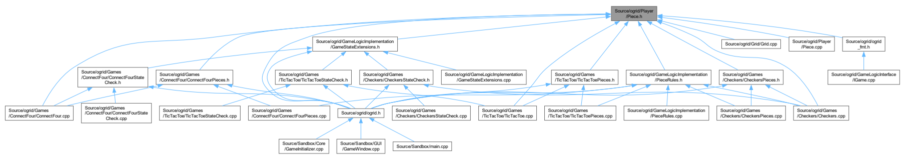
Contains the Piece class.

```
#include <string>
#include <vector>
#include <GameLogicInterface/IAttackRule.h>
#include "GameLogicInterface/IMoveRule.h"
#include "Player/Player.h"
```

Include dependency graph for Piece.h:



This graph shows which files directly or indirectly include this file:



Classes

- class OGRID::Piece

  The Piece class. Used to represent a piece.

Namespaces

- namespace OGRID

8.67.1 Detailed Description

Contains the Piece class.

Date

2023-12-06

8.68 Piece.h

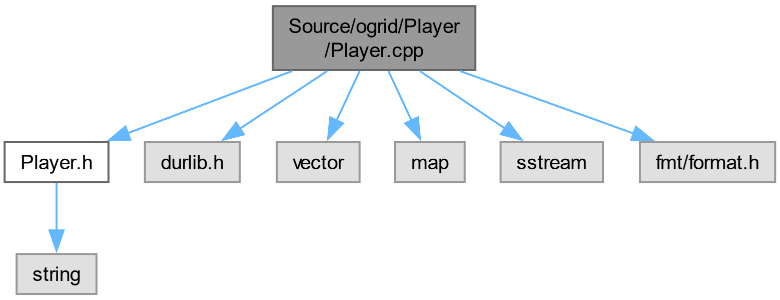
Go to the documentation of this file.

```cpp
#pragma once

#include <string>
#include <vector>

#include <GameLogicInterface/IAttackRule.h>
#include "GameLogicInterface/IMoveRule.h"
#include "Player/Player.h"

namespace OGRID
{
    class Piece
    {
    protected:
        // String representation of the move (char), like (chess) "K" for king, "Q" for queen, etc.
        std::string m_representation;

        // Rules for this move type
        std::vector<IMoveRule *> m_moveRules;

        // Rules for attacking
        std::vector<IAttackRule *> m_attackRules;

        // Owner of this piece
        Player *m_owner;

    public:
        Piece(std::string rep, Player *player);

        ~Piece();

        void AddMoveRule(IMoveRule *rule);

        void AddAttackRule(IAttackRule *rule);

        const std::string &GetRepresentation() const;

        const Player *GetOwner() const;

        void SetOwner(Player *player);

        bool isValidMove(Grid *grid, int fromX, int fromY, int toX, int toY) const;

        bool isValidAttack(Grid *grid, int fromX, int fromY, int toX, int toY, bool &canContinue)
const;
    };
}
```

8.69 Source/ogrid/Player/Player.cpp File Reference

```
#include "Player.h"
#include "durlib.h"
#include <vector>
#include <map>
#include <sstream>
#include "fmt/format.h"
```

Include dependency graph for Player.cpp:



Namespaces

- namespace OGRID

Functions

- PlayerType OGRID::PlayerTypeStringToEnum (const std::string &s)

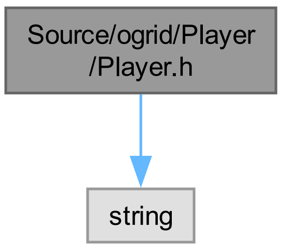
  Converts a string to a PlayerType.
- std::string OGRID::PlayerTypeEnumToString (PlayerType playerType)

  Converts a PlayerType to a string.
- std::string OGRID::PlayerVecToString (const std::vector< OGRID::Player * > &players)

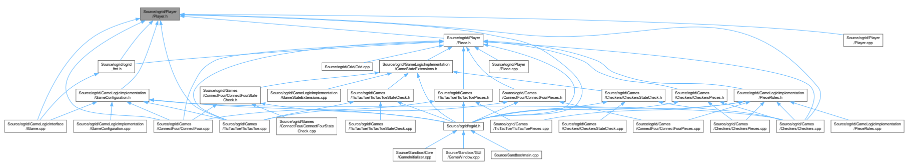
8.70 Source/ogrid/Player/Player.h File Reference

Contains the Player class.

`#include <string>`
Include dependency graph for Player.h:



This graph shows which files directly or indirectly include this file:



Classes

- class OGRID::Player

  The Player class. Used to represent a player.

Namespaces

- namespace OGRID

Enumerations

- enum OGRID::PlayerType { OGRID::Human = 0 , OGRID::AI = 1 }

  The type of the player.

Functions

- PlayerType OGRID::PlayerTypeStringToEnum (const std::string &s)

  Converts a string to a PlayerType.
- std::string OGRID::PlayerTypeEnumToString (PlayerType playerType)

  Converts a PlayerType to a string.
- std::string OGRID::PlayerVecToString (const std::vector< Player $*$ > &players)

  Converts a Vector of Players to a string.

8.70.1 Detailed Description

Contains the Player class.

Date

2023-12-06

8.71 Player.h

Go to the documentation of this file.

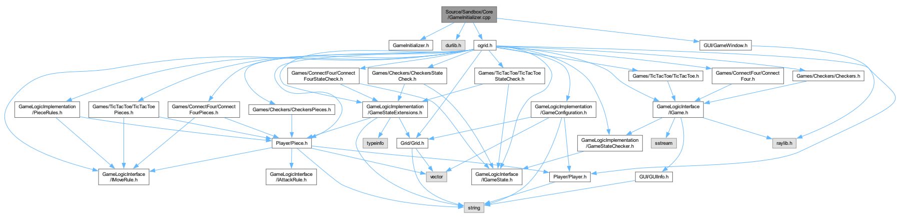
```
#pragma once

#include <string>

namespace OGRID
{
    // Forward declarations
    enum MoveType;

    enum PlayerType
    {
        Human = 0,
        AI = 1
    };

    PlayerType PlayerTypeStringToEnum(const std::string &s);

    std::string PlayerTypeEnumToString(PlayerType playerType);

    class Player
    {
    private:
        std::string m_PlayerName;

        PlayerType m_PlayerType;
        // The side to which the player belongs to -> -1 is no side

        int m_Side = -1;

        // Constructors & Destructors
    public:
        Player(std::string playerName = "GenericName", PlayerType playerType = PlayerType::Human, int
side = -1);

        ~Player();

        // Getters & Setters
    public:
        std::string GetPlayerName() const;

        void SetPlayerName(std::string playerName);

        PlayerType GetPlayerType() const;

        void SetPlayerType(PlayerType playerType);

        int GetSide() const;

        void SetSide(int side);
    };

    std::string PlayerVecToString(const std::vector<Player *> &players);
}
```

8.72 Source/Sandbox/Core/GameInitializer.cpp File Reference

```
#include "GameInitializer.h"
#include <durlib.h>
#include <ogrid.h>
```

```
#include "GUI/GameWindow.h"
```

Include dependency graph for GameInitializer.cpp:



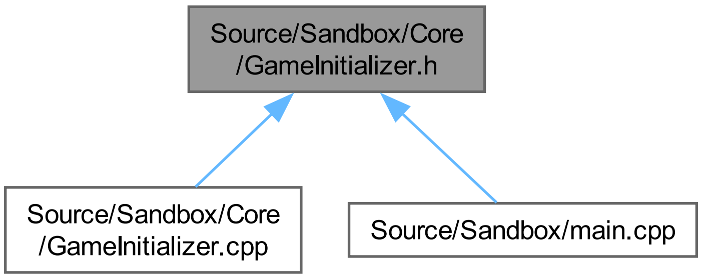
Namespaces

- namespace Sandbox

8.73 Source/Sandbox/Core/GameInitializer.h File Reference

Game initializer.

This graph shows which files directly or indirectly include this file:



Classes

- class Sandbox::GameInitializer

    Game initializer.

Namespaces

- namespace Sandbox

8.73.1 Detailed Description

Game initializer.

Date

2023-12-06

8.74 GameInitializer.h

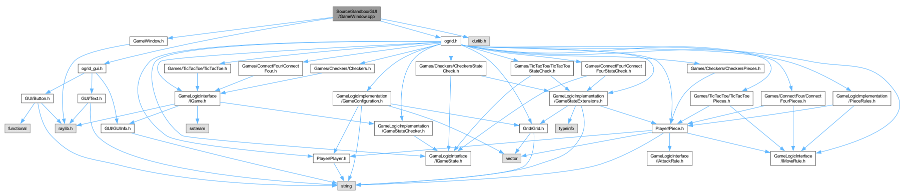
Go to the documentation of this file.

```
#pragma once

namespace Sandbox
{
    class GameInitializer
    {
    public:
        // Starts the app and gives choice of games
        static void Start();
    };
}
```

8.75 Source/Sandbox/GUI/GameWindow.cpp File Reference

```
#include "GameWindow.h"
#include <durlib.h>
#include <ogrid.h>
#include <ogrid_gui.h>
```

Include dependency graph for GameWindow.cpp:



Source/Sandbox/GUI /GameWindow.cpp
GameWindow.h
ogrid.h
durlib.h
ogrid_gui.h
Games/TicTacToe/TicTacToe.h
Games/ConnectFour/Connect Four.h
Games/Checkers/Checkers.h
Games/Checkers/CheckersState Check.h
Games/TicTacToe/TicTacToe StateCheck.h
Games/ConnectFour/Connect FourStateCheck.h
Games/Checkers/CheckersPieces.h
GUI/Button.h
GUI/Text.h
GameLogicInterface /IGame.h
GameLogicImplementation /GameConfiguration.h
GameLogicImplementation /GameStateExtensions.h
Games/TicTacToe/TicTacToe Pieces.h
Games/ConnectFour/Connect FourPieces.h
GameLogicImplementation /PieceRules.h
functional
raylib.h
GUI/GUIInfo.h
sstream
GameLogicImplementation /GameStateChecker.h
Grid/Grid.h
typeinfo
Player/Piece.h
Player/Player.h
GameLogicInterface /IGameState.h
vector
GameLogicInterface /IAttackRule.h
GameLogicInterface /IMoveRule.h
string

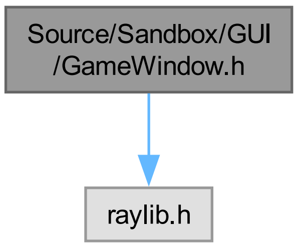
Namespaces

- namespace Sandbox

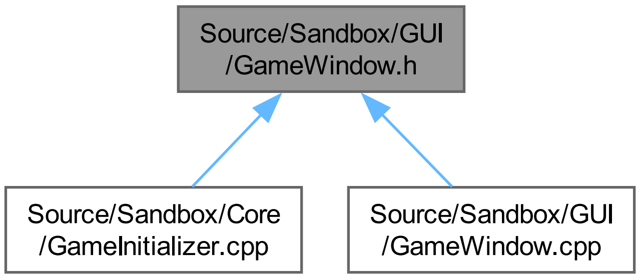
8.76 Source/Sandbox/GUI/GameWindow.h File Reference

Game window.

```
#include <raylib.h>
```
Include dependency graph for GameWindow.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Sandbox::GameWindow< T >

  Game window.

Namespaces

- namespace OGRID
- namespace Sandbox

8.76.1 Detailed Description

Game window.

Date

2023-12-06

See also

https://www.raylib.com/

8.77 GameWindow.h

Go to the documentation of this file.

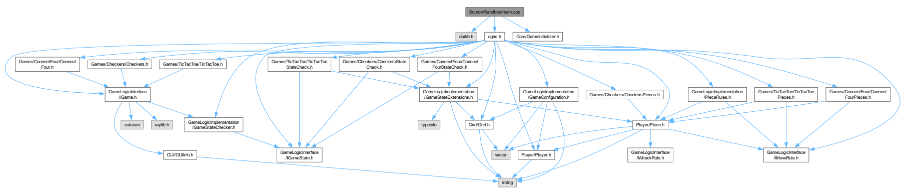
```
#pragma once

#include <raylib.h>

namespace OGRID
{
    class Button;
    class Text;
}

namespace Sandbox
{
    template <class T>
    class GameWindow
    {
    private:
        T *m_Game;

        bool m_Running = false;

        OGRID::Button *restartButton;

        OGRID::Text *gameOverText;

        OGRID::Text *winnerText;

        OGRID::Text *currentPlayerText;

        OGRID::Text *turnText;

        OGRID::Text *drawText;

    public:
        GameWindow() = default;

        ~GameWindow();

        void Start();

    private:
        void Run();

        void PreRun();

        void OnUpdate();

        void DrawGrid();

        Vector2 GetCellFromMouse(Vector2 mousePosition);

        void UpdateWindowDimensions();

        void MouseButtonPress();

        void InProgress();

        void GameOver();
    };
}
```

8.78 Source/Sandbox/main.cpp File Reference

Main entry point.

```
#include <durlib.h>
#include <ogrid.h>
#include "Core/GameInitializer.h"
```

Include dependency graph for main.cpp:


Source/Sandbox/main.cpp
durlib.h
ogrid.h
Core/GameInitializer.h
Games/ConnectFour/Connect Four.h
Games/Checkers/Checkers.h
Games/TicTacToe/TicTacToe.h
Games/TicTacToe/TicTacToe StateCheck.h
Games/Checkers/CheckersState Check.h
Games/ConnectFour/Connect FourStateCheck.h
GameLogicInterface /IGame.h
GameLogicImplementation /GameStateExtensions.h
GameLogicImplementation /GameConfiguration.h
Games/Checkers/CheckersPieces.h
GameLogicImplementation /PieceRules.h
Games/TicTacToe/TicTacToe Pieces.h
Games/ConnectFour/Connect FourPieces.h
sstream
raylib.h
GameLogicImplementation /GameStateChecker.h
typeinfo
Grid/Grid.h
Player/Piece.h
GUI/GUIInfo.h
GameLogicInterface /IGameState.h
vector
Player/Player.h
GameLogicInterface /IAttackRule.h
GameLogicInterface /IMoveRule.h
string

Functions

- int main ()

8.78.1 Detailed Description

Main entry point.

Date

2023-12-06

8.78.2 Function Documentation

8.78.2.1 main()

```
int main ( )
```