Domination

1.0.0

Generated by Doxygen 1.8.13

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

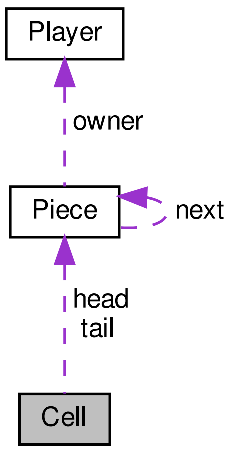
Class Documentation

2.1 Cell Struct Reference

Represents a cell on the 8x8 game board.

`#include <components.h>`

Collaboration diagram for Cell:



Public Attributes

- Piece ∗ head

    Pointer to the top-most piece in that cell.
- Piece ∗ tail

    Pointer to the bottom most piece in that cell.
- uint8_t length

    The number of Pieces on the cell.
- uint8_t rowIndex

    The row index of the cell.
- uint8_t columnIndex

    The column index of the cell.

Related Functions

(Note that these are not member functions.)

- void movePieces (Cell ∗source, Cell ∗destination, unsigned int count)
- static void shortenCell (Cell ∗cell)
- unsigned getDistance (Cell ∗cell1, Cell ∗cell2)
- Cell ∗ askUserForCell (Game ∗game, Cell ∗sourceCell, bool ∗placeReservedPiece, unsigned int maxDist)

2.1.1 Detailed Description

Represents a cell on the 8x8 game board.

2.1.2 Friends And Related Function Documentation

2.1.2.1 askUserForCell()

```
Cell * askUserForCell (
            Game * game,
            Cell * sourceCell,
            bool * placeReservedPiece,
            unsigned int maxDist )  [related]
```

Allows players to select a cell on the game board.
If source is NULL. Player will be asked whether to move the stack or place a piece
In that case, placeReservedPiece will be set to true is player wants to place a piece. Must not be NULL in that case maxDist only used if sourceCell is not NULL. maxDist specifies the maximum (taxicab) distance sourceCell can be from selected cell

2.1.2.2 getDistance()

```
unsigned getDistance (
            Cell * cell1,
            Cell * cell2 )  [related]
```

Gets the taxicab distance between two cells

Parameters

cell1	Cell1
cell2	Cell2

Returns

The taxicab distance between the two cells

2.1.2.3 movePieces()

```
void movePieces (
            Cell * source,
            Cell * destination,
            unsigned int count )  [related]
```

Moves **count** number of pieces from **source** to **destination**

Parameters

destination	Where to move the pieces
source	Where to move the pieces from
count	How many pieces to move

2.1.2.4 shortenCell()

```
static void shortenCell (
            Cell * cell )  [related]
```

Performs extra logic when a stack is greater than 5 pieces.
Cell must be $>$ 5 when function is called
All extra pieces are free'd
If the removed pieces are the player's, the player's reservedCounter is increased appropriately

Parameters

cell	The cell which to shorten

Attention

an assertion is made that **cell->length** $>$ **5**

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

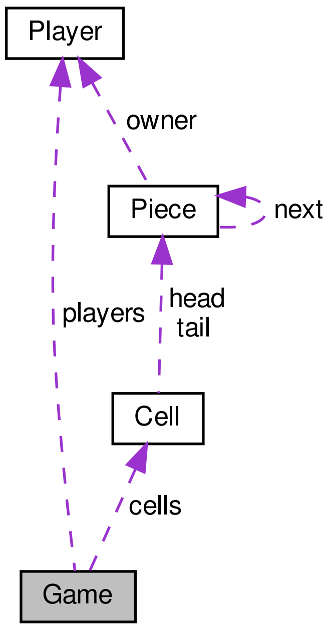
- /home/nskobelevs/Projects/Domination/src/components.h
- /home/nskobelevs/Projects/Domination/src/gameLogic.c
- /home/nskobelevs/Projects/Domination/src/gui.c

2.2 Game Struct Reference

Represents an instance of the game.

```
#include <components.h>
```

Collaboration diagram for Game:



Public Attributes

- Player ∗ players [2]

  Pointers to player1 and player2.
- Cell ∗ cells [8][8]

  an 8x8 2D array of cells. If NULL, cell is not a valid position.
- unsigned short moveIndex

  The current move index.

Related Functions

(Note that these are not member functions.)

- static void pushPiece (Cell ∗cell, Player ∗player)
- void runGame (Game ∗game)
- static bool playerCanMakeMove (Game ∗game, Player ∗player)
- Game ∗ initialiseGame (void)
- void freeBoard (Game ∗game)

2.2.1 Detailed Description

Represents an instance of the game.

2.2.2 Friends And Related Function Documentation

2.2.2.1 freeBoard()

```
void freeBoard (
             Game * game )  [related]
```

Free's all allocated memory

Parameters

game	Game instance

2.2.2.2 initialiseGame()

```
Game * initialiseGame (
             void  )  [related]
```

Initialised the game and it' players

Returns

The game variables

2.2.2.3 playerCanMakeMove()

```
static bool playerCanMakeMove (
             Game * game,
             Player * player )  [related]
```

Returns true/false whether a player can move **any** piece on the board

Parameters

game	Game instance
player	The player being checked

Returns

bool signifying whether player can move

2.2.2.4 pushPiece()

```
static void pushPiece (
            Cell * cell,
            Player * player )  [related]
```

Creates a new piece owned by player and places it on top of cell

Parameters

cell	The cell where to place the new cell
player	The player which will own the cell

Note

Assume's player's reservedCounter $>$ 0

2.2.2.5 runGame()

```
void runGame (
            Game * game )  [related]
```

Runs the main game loop

Parameters

game	A game instance

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

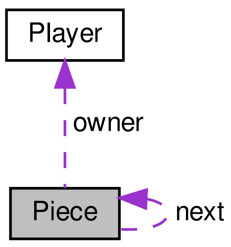
- /home/nskobelevs/Projects/Domination/src/components.h
- /home/nskobelevs/Projects/Domination/src/gameLogic.c
- /home/nskobelevs/Projects/Domination/src/init.c

2.3 Piece Struct Reference

Represents a single game piece.

```
#include <components.h>
```

Collaboration diagram for Piece:



Public Attributes

- Player ∗ owner

  A pointer to the player that owns the piece.
- struct Piece ∗ next

  A pointer to the piece below it. NULL if this is the bottom-most piece.

2.3.1 Detailed Description

Represents a single game piece.

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

- /home/nskobelevs/Projects/Domination/src/components.h

2.4 Player Struct Reference

Represents a player in a game.

```
#include <components.h>
```

Public Attributes

- char name [24]

  The player name.
- Colour colour

  The player's chosen colour representation.
- unsigned int reservedCounter

  The number of pieces a player has reserved.
- unsigned int capturedCounter

  The number of opponent's pieces a player has captured.

Related Functions

(Note that these are not member functions.)

- void askPlayerForName (Player ∗player, Player ∗otherPlayer)
- void askPlayerForColour (Player ∗player, Player ∗otherPlayer)
- static Player ∗ initialisePlayer (Player ∗otherPlayer)

2.4.1 Detailed Description

Represents a player in a game.

2.4.2 Friends And Related Function Documentation

2.4.2.1 askPlayerForColour()

```
void askPlayerForColour (
            Player ∗ player,
            Player ∗ otherPlayer )  [related]
```

Asks a player what colour they want.
If otherPlayer is not NULL, player won't be allowed to choose the same colour

Parameters

player	The player being asked for colour
otherPlayer	Other player. Both players can't have the same colour

2.4.2.2 askPlayerForName()

```
void askPlayerForName (
            Player ∗ player,
            Player ∗ otherPlayer )  [related]
```

Parameters

player	The player which is being asked for their name
otherPlayer	If not NULL, will stop player from having same name as otherPlayer

2.4.2.3 initialisePlayer()

```
static Player * initialisePlayer (
            Player * otherPlayer )  [related]
```

Initialises a player

Parameters

otherPlayer	The player1Colour to avoid both players having same colour

Returns

A pointer to the initialised player

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

- /home/nskobelevs/Projects/Domination/src/components.h
- /home/nskobelevs/Projects/Domination/src/gui.c
- /home/nskobelevs/Projects/Domination/src/init.c

Index