My Project

Generated by Doxygen 1.8.11

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

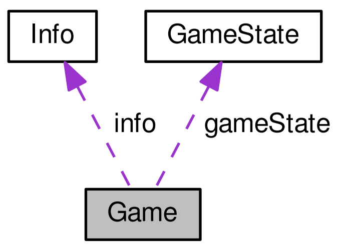
Class Documentation

3.1 Game Class Reference

The main Game class. You should make a derived class from it.

`#include <Game.h>`

Collaboration diagram for Game:



Public Member Functions

- sf::Vector2f getMousePosition ()

  Returns mouse cursor position after applying camera transform.
- void moveCamera (float cx, float cy, float zoom=1.f)

  Moves camera so that its centered at (cx, cy) and zoomed with specified factor.
- void run ()

  This function simply runs the game until the main window is closed or an exception fired.
- template<class T >
  void addDrawable (const T &x)

  This function simply calls gameState.addDrawable(x)
- void showOutputWindow ()

  Shows window with content of mout and merr.
- int getTurnsLeft ()

  Returns the number of turns left.

Protected Member Functions

- virtual void draw ()

  You can override this function for custom real-time drawing.
- virtual void update ()

  You can override this function for custom real-time updates.
- virtual void sync ()

  This function should contain all synchronisation with server.
- virtual void firstSync ()

  You can override this function if you want the first sync to behave differently. Otherwise normal sync() will be called.
- virtual void sendCommands ()

  You can write sending commands code in this function.
- virtual void myProcessEvent (const sf::Event &event)

  You can override this function for event handling. However, it doesn't handle mouse events.
- virtual void leftClick (sf::Vector2f position)

  This function is called whenever the user clicks left mouse button.
- virtual void rightClick (sf::Vector2f position)

  This function is called whenever the user clicks right mouse button.
- virtual void selectedRect (sf::FloatRect rect)

  This function is called after selecting a rectangle area with mouse.

Protected Attributes

- sf::RenderWindow window

  The window. You can use it to call window.draw(something)
- int turnsLeft = 1e9

  The turns left counter. It updates every turn.
- GameState gameState

  The game state. It contains all drawables and logs. Can be used to call gameState.addDrawable(something)

3.1.1 Detailed Description

The main Game class. You should make a derived class from it.

3.1.2 Member Function Documentation

3.1.2.1 virtual void Game::leftClick (sf::Vector2f *position*) `[inline], [protected], [virtual]`

This function is called whenever the user clicks left mouse button.

Parameters

position	the coordinates of mouse pointer after applying camera transform

3.1.2.2 void Game::moveCamera (float *cx*, float *cy*, float *zoom* = `1.f`)

Moves camera so that its centered at (cx, cy) and zoomed with specified factor.

Parameters

cx	the x-coordinate of center
cy	the y-coordinate of center
zoom	the zoom factor

3.1.2.3 virtual void Game::myProcessEvent (const sf::Event & *event*) `[inline]`, `[protected]`, `[virtual]`

You can override this function for event handling. However, it doesn't handle mouse events.

Parameters

event	the sfml event

See also

leftClick, rightClick, selectedRect

3.1.2.4 virtual void Game::rightClick (sf::Vector2f *position*) `[inline]`, `[protected]`, `[virtual]`

This function is called whenever the user clicks right mouse button.

Parameters

position	the coordinates of mouse pointer after applying camera transform

3.1.2.5 virtual void Game::selectedRect (sf::FloatRect *rect*) `[inline]`, `[protected]`, `[virtual]`

This function is called after selecting a rectangle area with mouse.

Parameters

rect	the selected area

3.1.2.6 virtual void Game::sendCommands ( ) `[inline]`, `[protected]`, `[virtual]`

You can write sending commands code in this function.

When SEND_COMMANDS_LATE if off it is run right after sync(). Otherwise it is run after turnDuration which is either specified or measured automatically (not recommended)

CAUTION: this function is not run after firstSync() (you can do it manually)

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

- Game.h
- Game.cpp

3.2 GameState Class Reference

This class contains the state of the game. It draws it, shows the output window and saves game for further viewing.

```
#include <GameState.h>
```

Public Member Functions

- GameState (string title="Output")

  GameState.
- void showWindow ()

  Shows the window with the content of mout and merr.
- template<class T >
  void addDrawable (const T &x)

  Adds x to the drawables list.

3.2.1 Detailed Description

This class contains the state of the game. It draws it, shows the output window and saves game for further viewing.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 GameState::GameState (string *title* = "Output")

GameState.

Parameters

title	The output window's title

3.2.3 Member Function Documentation

3.2.3.1 template<class T > void GameState::addDrawable (const T & *x*) [inline]

Adds x to the drawables list.

The drawable can be one of the following types: sf::CircleShape, sf::RectangleShape, sf::ConvexShape, sf::Text. It is shown until the next turn and saved for later replays

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

- GameState.h
- GameState.cpp

3.3 HexMap Class Reference

Useful class for using hex maps.

`#include <HexMap.h>`

Public Member Functions

- HexMap ()=default
  The default constructor.
- HexMap (int rows, int columns, float size=30.f)
  Simply calls init(rows, colums, size)
- void init (int rows, int columns, float size=30.f)
  Initializes the map. Initially, all hexes are white.
- int rowCount () const
  Returns the number of rows.
- int colCount () const
  Returns the number of columns.
- void setColor (int row, int column, sf::Color color)
  Sets the color of a selected hex.
- sf::Color getColor (int row, int column) const
  Gets the color of a selected hex.
- sf::Vector2f getPosition (int row, int column) const
  Returns position of the center of a selected hex.
- pair< int, int > getHex (sf::Vector2f position) const
  Returns pair of (row, column) coords of the hex which contains the position.
- void draw (GameState &gameState) const
  adds all drawables to the gameState
- void draw (sf::RenderWindow &window) const
  draws the map directly to the window
- vector< pair< int, int > > getNeighbours (int row, int column) const
  Returns position of each neighbour of the hex on (row, column)

3.3.1 Detailed Description

Useful class for using hex maps.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 HexMap::HexMap (int *rows,* int *columns,* float *size* = `30.f`) `[inline]`

Simply calls init(rows, colums, size)

Parameters

rows	the number of rows
columns	the number of columns
size	the size of one hex (2 $*$ radius)

3.3.3 Member Function Documentation

3.3.3.1 void HexMap::draw (GameState & *gameState*) const

adds all drawables to the gameState

Parameters

gameState	the game state

3.3.3.2 void HexMap::draw (sf::RenderWindow & *window*) const

draws the map directly to the window

Parameters

window	the render window

3.3.3.3 sf::Color HexMap::getColor (int *row,* int *column*) const

Gets the color of a selected hex.

Parameters

row	the row of a hex
column	the column of a hex

Returns

The color of the hex

3.3.3.4 pair$<$ int, int $>$ HexMap::getHex (sf::Vector2f *position*) const

Returns pair of (row, column) coords of the hex which contains the position.

Parameters

position	the position where hex is looked for

Returns

The row and column of the hex or (-1, -1) if position is outside the map

3.3.3.5 vector< pair< int, int > > HexMap::getNeighbours (int *row*, int *column*) const

Returns position of each neighbour of the hex on (row, column)

Parameters

row	the row of the hex
column	the column of the hex

Returns

A vector of hex' neighbours' positions

3.3.3.6 sf::Vector2f HexMap::getPosition (int *row*, int *column*) const

Returns position of the center of a selected hex.

Parameters

row	the row of a hex
column	the column of a hex

Returns

The position of the center of the specified hex

3.3.3.7 void HexMap::init (int *rows*, int *columns*, float *size* = `30.f`)

Initializes the map. Initially, all hexes are white.

Parameters

rows	the number of rows
columns	the number of columns
size	the size of one hex (2 $*$ radius)

3.3.3.8 void HexMap::setColor (int *row*, int *column*, sf::Color *color*)

Sets the color of a selected hex.

Parameters

row	the row of a hex
column	the column of a hex
color	the new color

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

- HexMap.h
- HexMap.cpp

3.4 Info Class Reference

Usable class for printing real-time info.

`#include <Info.h>`

Public Member Functions

- void addFunction (string label, function< string()> function)

  Adds function returning string to the info.
- template<class T >
  void addItem (string label, T &variable)

  Adds a reference to the variable.
- void removeItem (string label)

  Removes all infos with specified label.
- bool hasItem (string label)

  Checks if there exists any info with specified label.
- void clear ()

  Removes all items.
- void draw (sf::RenderWindow &window)

  Draws info to the window.
- void log (ostream &o)

  Saves info for later replays.

3.4.1 Detailed Description

Usable class for printing real-time info.

3.4.2 Member Function Documentation

3.4.2.1 void Info::addFunction (string *label,* function< string()> *function*) [inline]

Adds function returning string to the info.

Parameters

label	the label
function	the function that should take 0 arguments and return a string

3.4.2.2 template<class T > void Info::addItem (string *label,* T & *variable*) [inline]

Adds a reference to the variable.

Parameters

label	the label
variable	a reference to the variable. It must be possible to call to_string(variable). The variable should exist until the end of the game.

3.4.2.3 void Info::draw (sf::RenderWindow & *window*)

Draws info to the window.

Parameters

window	the render window

3.4.2.4 bool Info::hasItem (string *label*)

Checks if there exists any info with specified label.

Parameters

label	the label

Returns

3.4.2.5 void Info::log (ostream & *o*)

Saves info for later replays.

Parameters

o	the stream to save to

3.4.2.6 void Info::removeItem (string *label*)

Removes all infos with specified label.

Parameters

label	the label

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

- Info.h
- Info.cpp

3.5 MultiStream Class Reference

Class used to combining multiple streams into one.

```
#include <Log.h>
```

Public Member Functions

- MultiStream ()=default

  Default constructor.
- MultiStream (initializer_list< ostream ∗ > list)

  Initialization of multistream with a list of streams.
- void addStream (ostream &str)

  Adds a stream.
- void clear ()

  Removes all streams.
- template<class T >
  MultiStream & operator<< (const T &obj)

  writes obj into all streams (unfortunately, manipulators like endl don't work) the object to write
- void flush ()

  Flushes all streams.

3.5.1 Detailed Description

Class used to combining multiple streams into one.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 MultiStream::MultiStream (initializer_list< ostream ∗ > *list*) [inline]

Initialization of multistream with a list of streams.

Parameters

a	list of ofstream pointers

3.5.3 Member Function Documentation

3.5.3.1 void MultiStream::addStream (ostream & *str*) `[inline]`

Adds a stream.

Parameters

the	stream to add

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

- Log.h

Chapter 4

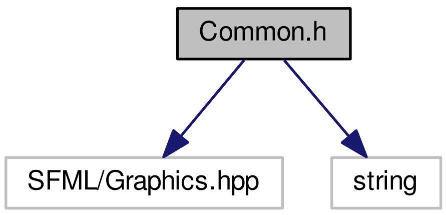
File Documentation

4.1 Common.h File Reference

Contains some useful functions and commonFont.

```
#include <SFML/Graphics.hpp>
#include <string>
```

Include dependency graph for Common.h:



Functions

- double length (const double &x, const double &y)

  Returns length of a vector (x, y)
- double lengthSquare (const double &x, const double &y)

  Returns length^2 of a vector (x, y)
- double dist (const double &x1, const double &y1, const double &x2, const double &y2)

  Returns euclidean distance between vectors (x1, y1) and (x2, y2)
- double distSquare (const double &x1, const double &y1, const double &x2, const double &y2)

  Returns square of euclidean distance between vectors (x1, y1) and (x2, y2)
- sf::CircleShape makeCircle (float radius, float x, float y, sf::Color fillColor)

  A factory function for creating circles.
- sf::RectangleShape makeRectangle (float width, float height, float x, float y, sf::Color fillColor)

  A factory function for creating rectangles.
- sf::RectangleShape makeLine (float x1, float y1, float x2, float y2, float thickness, sf::Color fillColor)

  A factory function for creating lines.
- sf::Text makeText (string caption, int fontSize, float x, float y, sf::Color color)

  A factory function for creating texts.

Variables

- sf::Font commonFont

  You should use only this font in your program. It loads font from FONT_PATH defined in Config.h.

4.1.1 Detailed Description

Contains some useful functions and commonFont.

4.1.2 Function Documentation

4.1.2.1 sf::CircleShape makeCircle (float *radius,* float *x,* float *y,* sf::Color *fillColor*)

A factory function for creating circles.

Parameters

radius	the radius of the circle
x	the x-coordinate of the circle's center
y	the y-coordinate of the circle's center
fillColor	the color of the circle

Returns

an sf::CircleShape with center at (x, y)

4.1.2.2 sf::RectangleShape makeLine (float *x1,* float *y1,* float *x2,* float *y2,* float *thickness,* sf::Color *fillColor*)

A factory function for creating lines.

Parameters

x1	the x-coordinate of first end of line
y1	the y-coordinate of first end of line
x2	the x-coordinate of second end of line
y2	the y-coordinate of second end of line
thickness	the thickness of the line
fillColor	the color of the line

Returns

an sf::RectangleShape which is a line segment with specified thickness between (x1, y1) and (x2, y2)

4.1.2.3 sf::RectangleShape makeRectangle (float *width,* float *height,* float *x,* float *y,* sf::Color *fillColor*)

A factory function for creating rectangles.

Parameters

width	the width of the rect
height	the height of the rect
x	the x-coordinate of the rect's center
y	the y-coordinate of the rect's center
fillColor	the color of the rect

Returns

an sf::RectangleShape with center at (x, y)

4.1.2.4 sf::Text makeText (string *caption,* int *fontSize,* float *x,* float *y,* sf::Color *color*)

A factory function for creating texts.

Parameters

caption	the text to be displayed
fontSize	the size of characters
x	the x-coordinate of the center of text
y	the y-coordinate of the center of text
color	the color of text

Returns

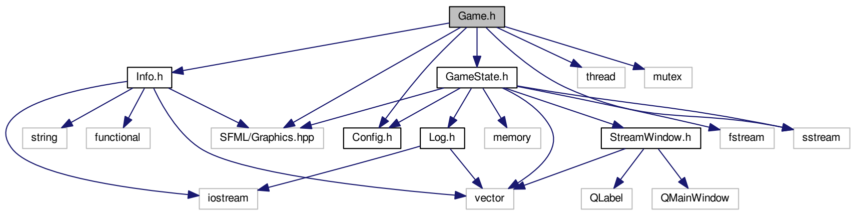
an sf::Text with center at (x, y)

4.2 Game.h File Reference

Contains the main Game class.

```
#include <SFML/Graphics.hpp>
#include "Info.h"
#include "Config.h"
#include "GameState.h"
#include <thread>
#include <mutex>
#include <sstream>
```

Include dependency graph for Game.h:



Classes

- class Game

  The main Game class. You should make a derived class from it.

4.2.1 Detailed Description

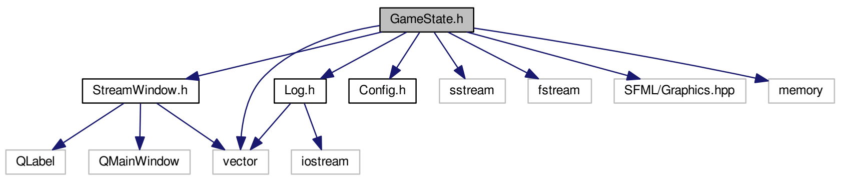
Contains the main Game class.

4.3 GameState.h File Reference

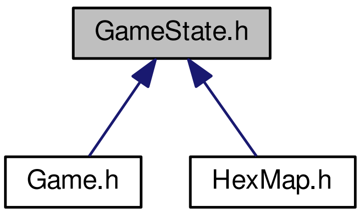
Contains the GameState class.

```
#include "StreamWindow.h"
#include "Log.h"
#include "Config.h"
#include <sstream>
#include <fstream>
#include <vector>
#include <SFML/Graphics.hpp>
#include <memory>
```

Include dependency graph for GameState.h:



This graph shows which files directly or indirectly include this file:



Classes

- class GameState

  This class contains the state of the game. It draws it, shows the output window and saves game for further viewing.

4.3.1 Detailed Description

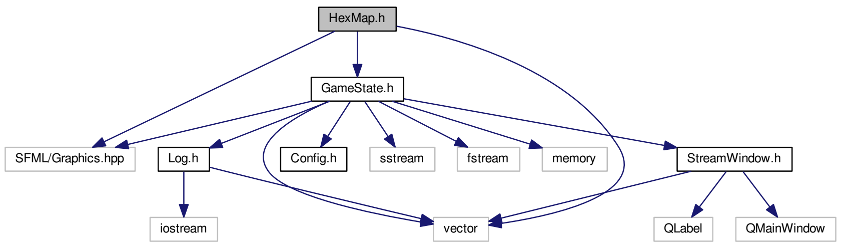
Contains the GameState class.

4.4 HexMap.h File Reference

Contains the HexMap class.

```
#include <vector>
#include <SFML/Graphics.hpp>
#include "GameState.h"
```

Include dependency graph for HexMap.h:



Classes

- class HexMap

  Useful class for using hex maps.

4.4.1 Detailed Description

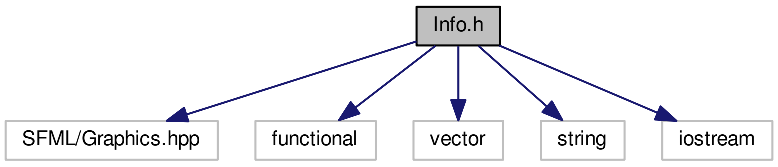
Contains the HexMap class.

4.5 Info.h File Reference

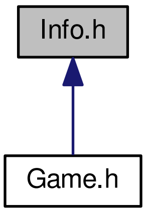
Contains the Info class.

```
#include <SFML/Graphics.hpp>
#include <functional>
#include <vector>
#include <string>
#include <iostream>
```

Include dependency graph for Info.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Info

  Usable class for printing real-time info.

4.5.1 Detailed Description

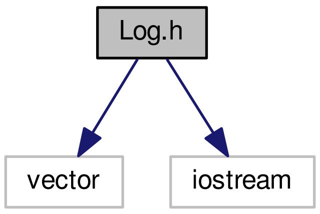
Contains the Info class.

4.6 Log.h File Reference

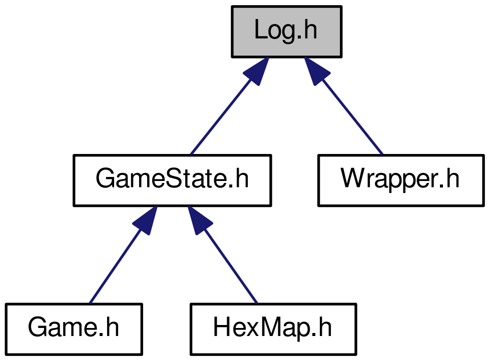
Contains the MultiStream class.

```
#include <vector>
#include <iostream>
```

Include dependency graph for Log.h:



This graph shows which files directly or indirectly include this file:



Classes

- class MultiStream

  Class used to combining multiple streams into one.

Functions

- void open_log (ofstream &of, string dir)

  Reopens the stream into a new file in a specified directory. If the directory doesn't exist it is created. The name of the file is HH_MM_SS.log where HH, MM, SS is current hour, minute and second.

Variables

- MultiStream mout

    MultiStream which initially contains only cout.
- MultiStream merr

    MultiStream which initially contains only cerr.

4.6.1 Detailed Description

Contains the MultiStream class.

4.6.2 Function Documentation

4.6.2.1 void open_log (ofstream & *of,* string *dir*)

Reopens the stream into a new file in a specified directory. If the directory doesn't exist it is created. The name of the file is HH_MM_SS.log where HH, MM, SS is current hour, minute and second.

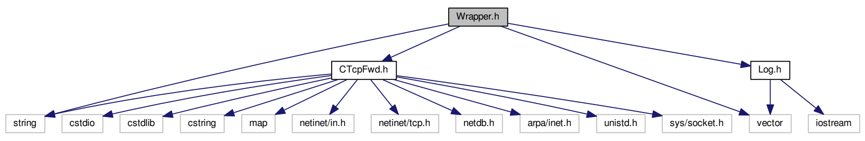
Parameters

of	the stream to reopen
dir	the name of the directory

4.7 Wrapper.h File Reference

Contains a few useful functions for communication with server.

```
#include "CTcpFwd.h"
#include "Log.h"
#include <string>
#include <vector>
```
Include dependency graph for Wrapper.h:



Functions

- void connect (string host, int port, string login, string password)

    Initializes TCP connection. After calling this function, stdout sends output to the server and stdin reads input from it.
- bool sendMessage (string message=string())

Writes message followed by endline to the stdout, and then checks if the response is OK or handles errors.

- void wait ()

  Sends WAIT message and reads 2 OKs.
- int turnsLeft ()

  Calls TURNS_LEFT_COMMAND (defined in Config.h) and reads the response.

4.7.1 Detailed Description

Contains a few useful functions for communication with server.

4.7.2 Function Documentation

4.7.2.1 void connect (string *host,* int *port,* string *login,* string *password*)

Initializes TCP connection. After calling this function, stdout sends output to the server and stdin reads input from it.

Parameters

host	the name of the host
port	the port number (integer)
login	team login
password	team password

4.7.2.2 bool sendMessage (string *message* = `string()`)

Writes message followed by endline to the stdout, and then checks if the response is OK or handles errors.

Parameters

message	the message to write

Returns

true if server returned OK, false otherwise

4.7.2.3 int turnsLeft ( )

Calls TURNS_LEFT_COMMAND (defined in Config.h) and reads the response.

Returns

the number of turns left

Index