Klondike

Generated by Doxygen 1.9.7

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

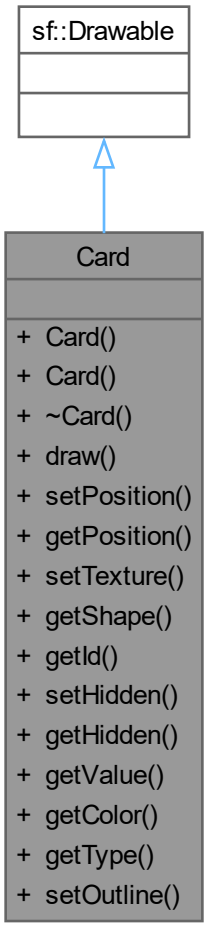
Chapter 4

Class Documentation

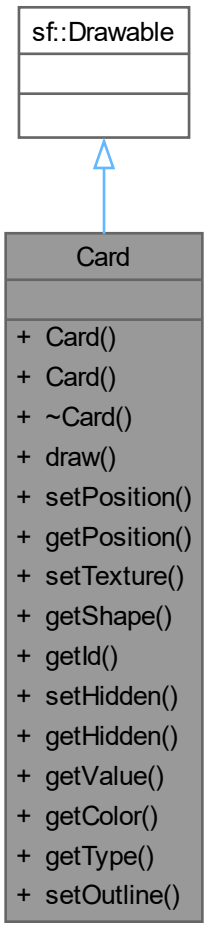
4.1 Card Class Reference

`#include <Card.h>`

Inheritance diagram for Card:



Collaboration diagram for Card:



Public Member Functions

- Card ()
- Card (int index, int value, std::string Color, std::string Type, std::string texturePath)
- ~Card ()=default
- void draw (RenderTarget &target, RenderStates state) const override
- void setPosition (float X, float Y)
- std::pair< float, float > getPosition ()
- void setTexture ()
- RectangleShape getShape ()
- int getId ()
- void setHidden (const bool i)
- bool getHidden ()
- int getValue ()
- std::string getColor ()
- std::string getType ()
- void setOutline ()

4.1.1 Constructor & Destructor Documentation

4.1.1.1 Card() [1/2]

```
Card::Card ( )
```

Card basic constructor

4.1.1.2 Card() [2/2]

```
Card::Card (
            int index,
            int value,
            std::string Color,
            std::string Type,
            std::string texturePath )
```

Card constructor

4.1.1.3 ~Card()

```
Card::~Card ( )  [default]
```

Card destructor

4.1.2 Member Function Documentation

4.1.2.1 draw()

```
void Card::draw (
            RenderTarget & target,
            RenderStates state ) const  [override]
```

Card draw (SFML) method

4.1.2.2 getColor()

```
std::string Card::getColor ( )
```

Getting card color method Here is the caller graph for this function:

4.1.2.3 getHidden()

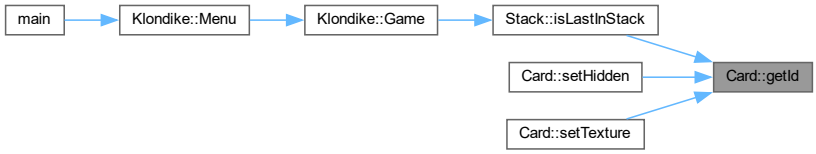
```
bool Card::getHidden ( )
```

Getting card visibility method

4.1.2.4 getId()

```
int Card::getId ( )
```

Getting card ID method Here is the caller graph for this function:



4.1.2.5 getPosition()

```
std::pair< float, float > Card::getPosition ( )
```

Getting card position method Here is the caller graph for this function:

4.1.2.6 getShape()

```
RectangleShape Card::getShape ( )
```

Getting card shape method (SFML feature)

4.1.2.7 getType()

```
std::string Card::getType ( )
```

Getting card type method Here is the caller graph for this function:

4.1.2.8 getValue()

```
int Card::getValue ( )
```

Getting card value method Here is the caller graph for this function:

4.1.2.9 setHidden()

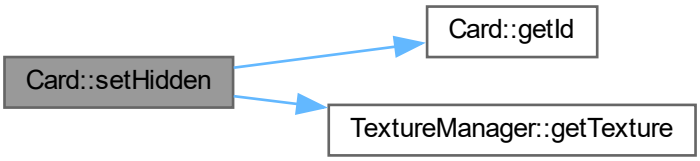
```
void Card::setHidden (
            const bool i )
```

Setting card visibility method.

Parameters

i	Hidden variable

Here is the call graph for this function:



4.1.2.10 setOutline()

```
void Card::setOutline ( )
```

Setting outline of card shape method

4.1.2.11 setPosition()

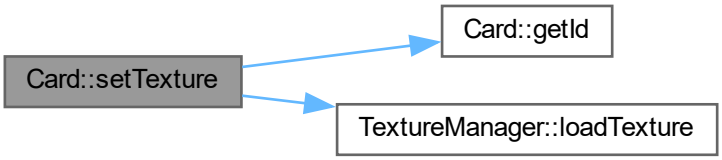
```
void Card::setPosition (
            float X,
            float Y )
```

Card possitioning method Here is the caller graph for this function:

4.1.2.12 setTexture()

```
void Card::setTexture ( )
```

Setting texture method Here is the call graph for this function:



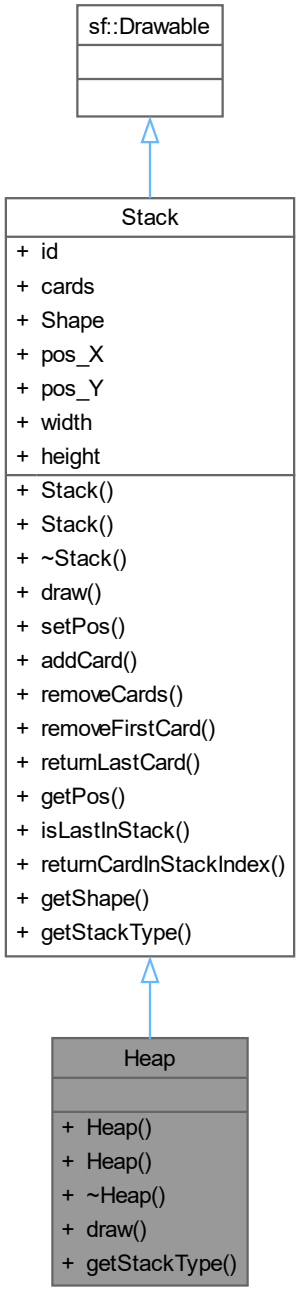
The documentation for this class was generated from the following files:

- Card.h
- Card.cpp

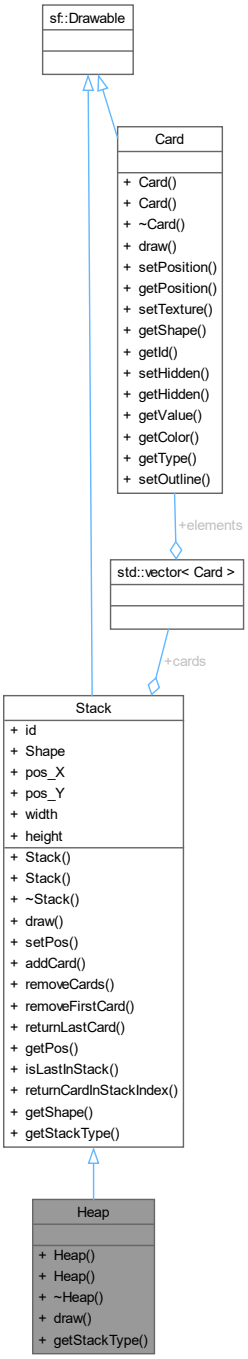
4.2 Heap Class Reference

```
#include <Heap.h>
```

Inheritance diagram for Heap:



Collaboration diagram for Heap:



Public Member Functions

- Heap ()
- Heap (float RectangleX, float RectangleY, std::string stackType)
- ~Heap ()=default
- void draw (RenderTarget &target, RenderStates state) const override
- std::string getStackType ()

Public Member Functions inherited from Stack

- Stack (const int number, float RectangleX, float RectangleY, std::string stackType)
- Stack ()
- ~Stack ()=default
- virtual void draw (RenderTarget &target, RenderStates state) const override
- virtual void setPos (float x, float y)
- virtual void addCard (Card card)
- virtual void removeCards (int number)
- virtual void removeFirstCard ()
- virtual Card returnLastCard ()
- std::pair< float, float > getPos ()
- bool isLastInStack (Card &card)
- int returnCardInStackIndex (int card)
- RectangleShape getShape ()
- std::string getStackType ()

Additional Inherited Members

Public Attributes inherited from Stack

- int id
- std::vector< Card > cards
- RectangleShape Shape
- float pos_X
- float pos_Y
- const float width = 60
- const float height = 90

4.2.1 Constructor & Destructor Documentation

4.2.1.1 Heap() [1/2]

```
Heap::Heap ( )
```

Heap basic constructor

4.2.1.2 Heap() [2/2]

```
Heap::Heap (
            float RectangleX,
            float RectangleY,
            std::string stackType )
```

Heap constructor

4.2.1.3 ~Heap()

```
Heap::~Heap ( )  [default]
```

Heap destructor

4.2.2 Member Function Documentation

4.2.2.1 draw()

```
void Heap::draw (
            RenderTarget & target,
            RenderStates state ) const  [override], [virtual]
```

Heap drawing method (SFML)

Reimplemented from Stack.

4.2.2.2 getStackType()

```
std::string Heap::getStackType ( )
```

Getting stack type method

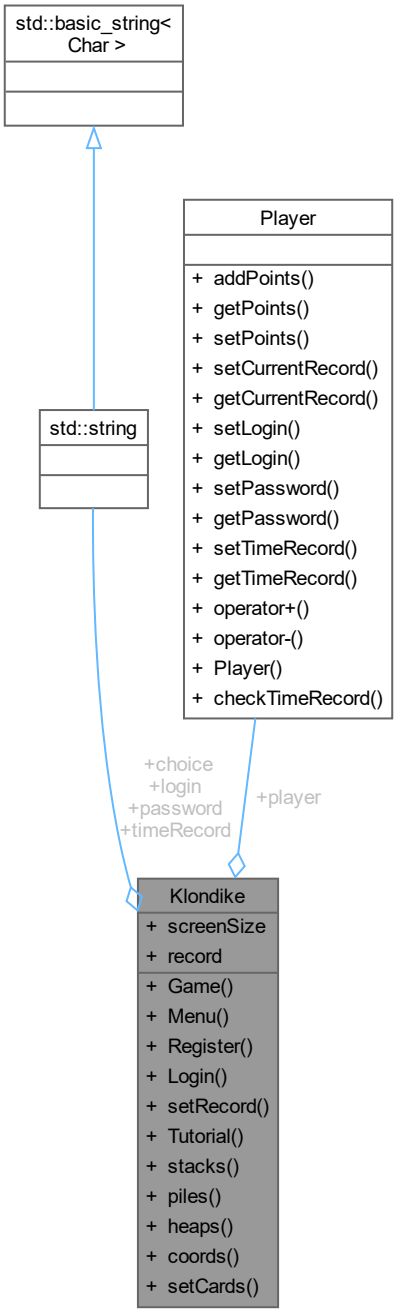
The documentation for this class was generated from the following files:

- Heap.h
- Heap.cpp

4.3 Klondike Class Reference

```
#include <Klondike.h>
```

Collaboration diagram for Klondike:



Static Public Member Functions

- static void Game ()
- static void Menu ()
- static void Register ()
- static bool Login (std::string &loginPlayer, std::string &passwordPlayer, int &record, std::string &timeRecord)
- static void setRecord (Player &player)

- static void Tutorial ()
- static std::vector< Stack > stacks ()
- static std::vector< Pile > piles ()
- static std::vector< Heap > heaps ()
- static std::vector< float > coords ()
- static void setCards (std::vector< Stack > &stacks, std::vector< Pile > &piles, std::vector< Heap > &heaps, std::vector< Card > &cards, std::vector< float > &coords)

Static Public Attributes

- static std::pair< float, float > screenSize
- static std::string choice
- static Player player
- static std::string login
- static std::string password
- static int record
- static std::string timeRecord

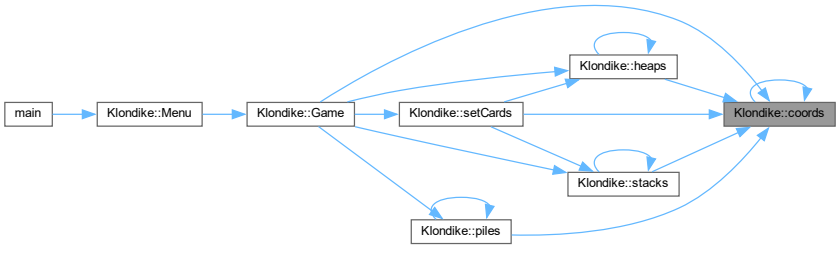
4.3.1 Member Function Documentation

4.3.1.1 coords()

```
std::vector< float > Klondike::coords ( )  [static]
```

Creating differents coords / values being used in program. They are dependent of the screen size. Here is the call graph for this function:

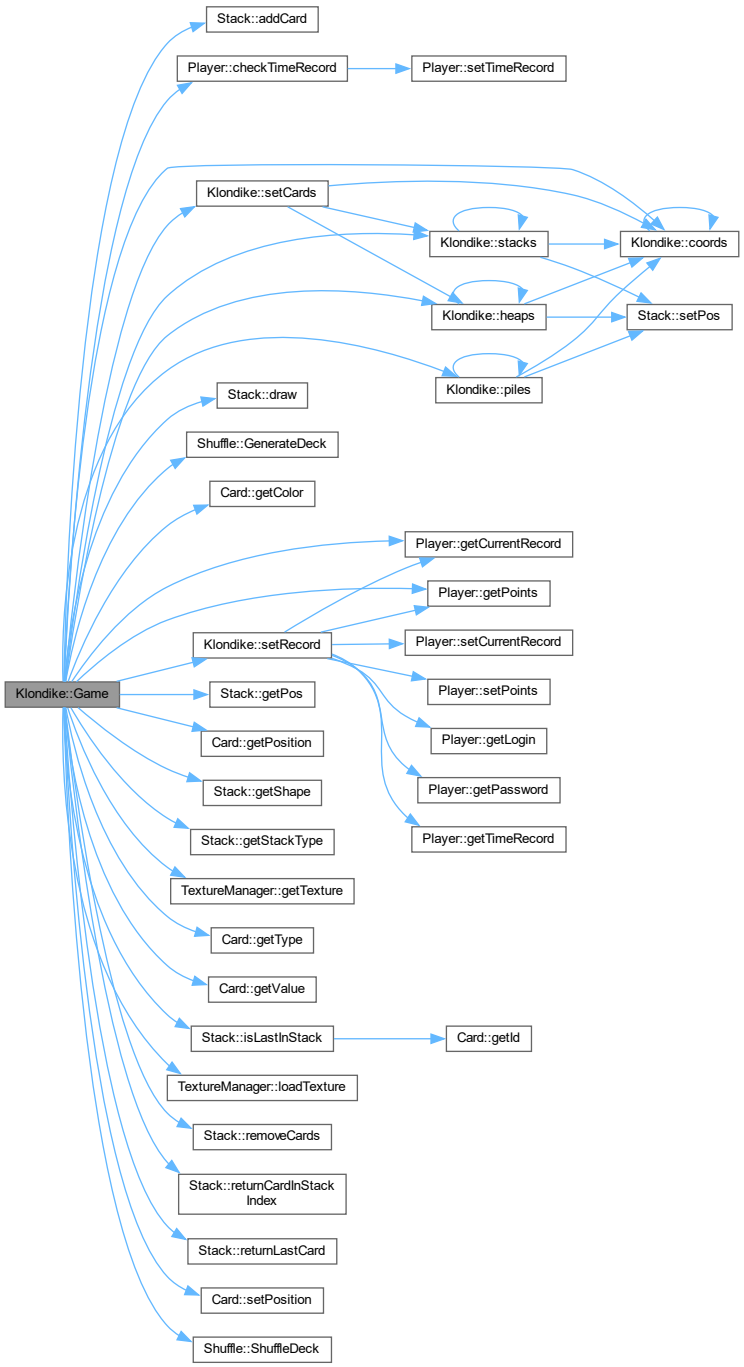
Here is the caller graph for this function:



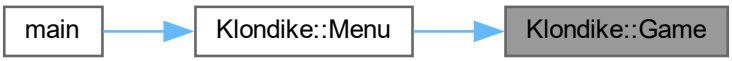
4.3.1.2 Game()

```
void Klondike::Game ( )  [static]
```

Whole Game Method Here is the call graph for this function:



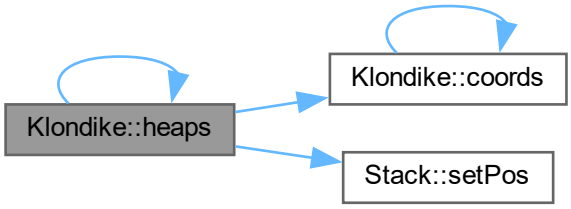
Here is the caller graph for this function:



4.3.1.3 heaps()

```
std::vector< Heap > Klondike::heaps ( )  [static]
```

Creating heaps method Here is the call graph for this function:

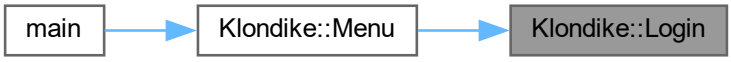


Here is the caller graph for this function:

4.3.1.4 Login()

```
bool Klondike::Login (
            std::string & loginPlayer,
            std::string & passwordPlayer,
            int & record,
            std::string & timeRecord )  [static]
```

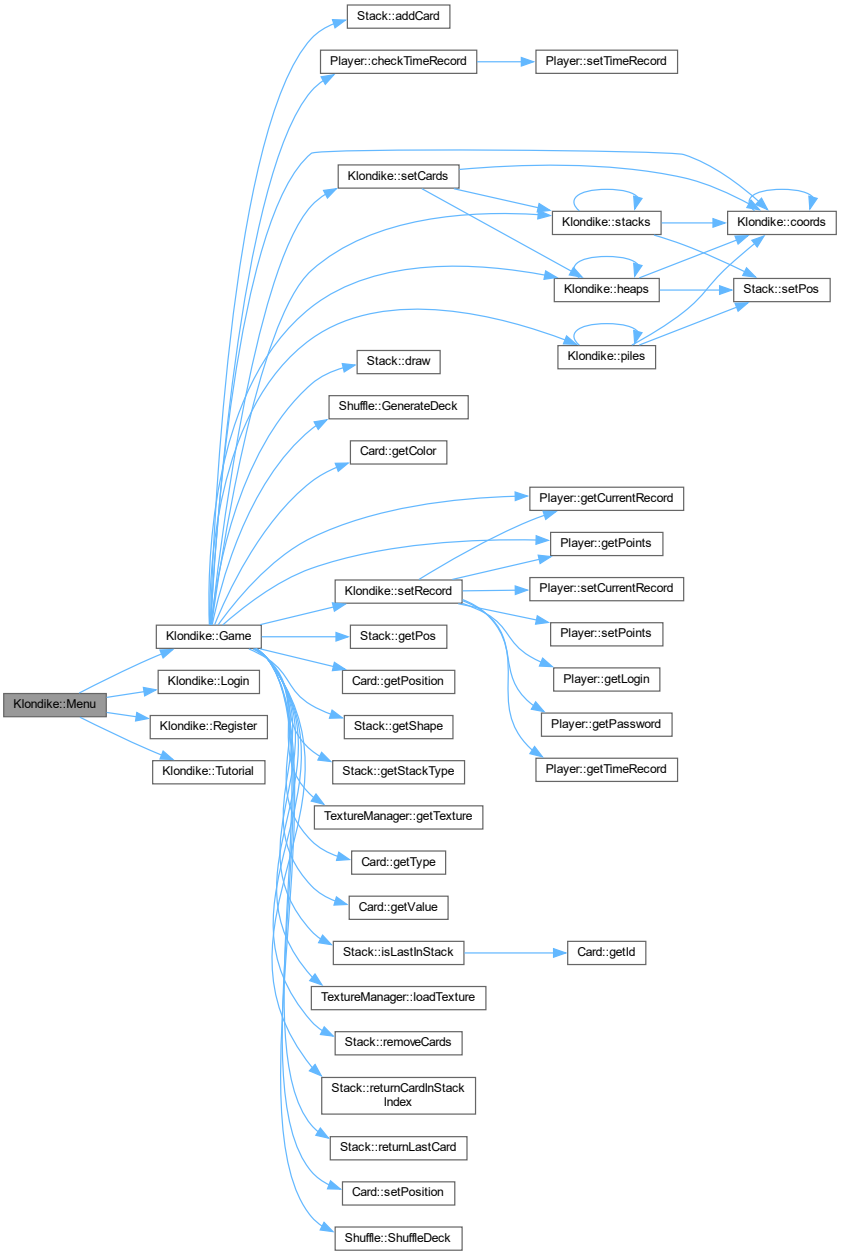
Player logging method Here is the caller graph for this function:



4.3.1.5 Menu()

```
void Klondike::Menu ( )  [static]
```

Displaying menu method Here is the call graph for this function:

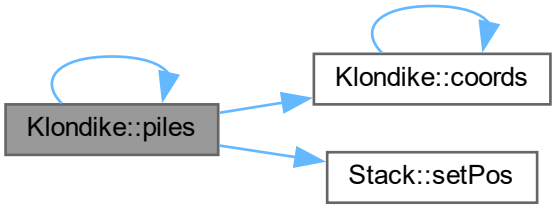


Here is the caller graph for this function:

4.3.1.6 piles()

```
std::vector< Pile > Klondike::piles ( )  [static]
```

Creating piles method Here is the call graph for this function:

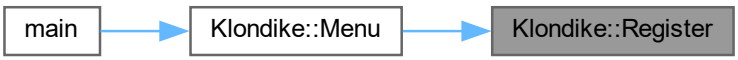


Here is the caller graph for this function:

4.3.1.7 Register()

```
void Klondike::Register ( )  [static]
```

Player registering method Here is the caller graph for this function:



4.3.1.8 setCards()

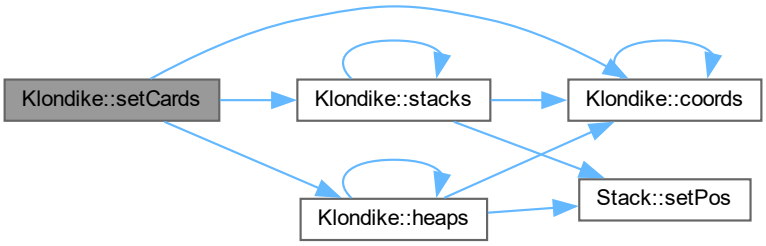
```
void Klondike::setCards (
            std::vector< Stack > & stacks,
            std::vector< Pile > & piles,
            std::vector< Heap > & heaps,
            std::vector< Card > & cards,
            std::vector< float > & coords )  [static]
```

Starting cards positioning method.

Parameters

stacks	Stacks vector
piles	Piles vector
heaps	Heaps vector
cards	Cards vector
coords	Coords vector

Here is the call graph for this function:

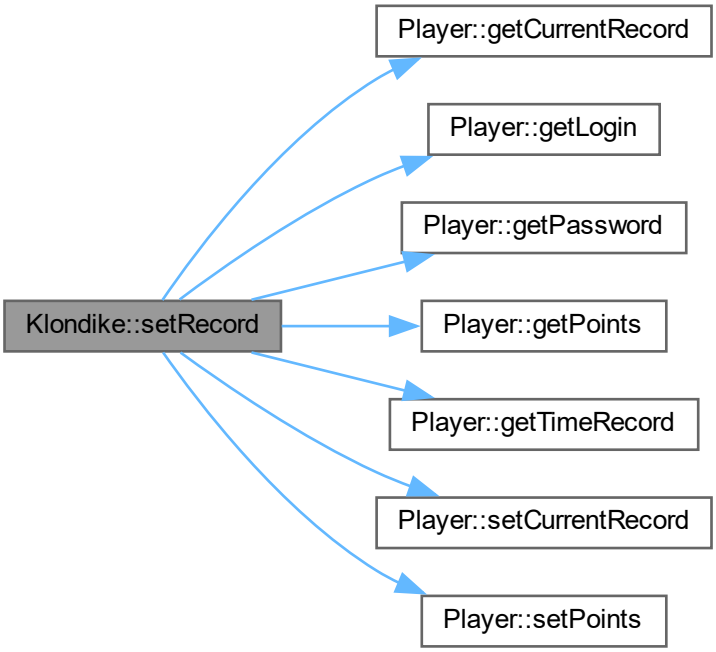


Here is the caller graph for this function:

4.3.1.9 setRecord()

```
void Klondike::setRecord (
            Player & player )  [static]
```

Setting new player's record method Here is the call graph for this function:

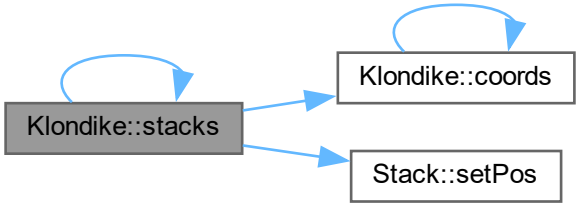


Here is the caller graph for this function:

4.3.1.10 stacks()

```
std::vector< Stack > Klondike::stacks ( )  [static]
```

Creating stacks method Here is the call graph for this function:



Here is the caller graph for this function:

4.3.1.11 Tutorial()

```
void Klondike::Tutorial ( )  [static]
```

Displaying tutorial method Here is the caller graph for this function:

4.3.2 Member Data Documentation

4.3.2.1 choice

```
std::string Klondike::choice  [static]
```

Menu choice variable

4.3.2.2 login

```
std::string Klondike::login  [static]
```

Player login

4.3.2.3 password

```
std::string Klondike::password  [static]
```

Player password

4.3.2.4 player

```
Player Klondike::player  [static]
```

Logged player object

4.3.2.5 record

```
int Klondike::record  [static]
```

Player saved record

4.3.2.6 screenSize

```
std::pair< float, float > Klondike::screenSize  [static]
```

Static variable - X and Y of screen

4.3.2.7 timeRecord

```
std::string Klondike::timeRecord  [static]
```

Record time of won game by the player

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

- Klondike.h
- Klondike.cpp

4.4 myEvent Class Reference

`#include <myEvent.h>`

Collaboration diagram for myEvent:

Public Member Functions

- myEvent ()
- myEvent (int type, int movedCardsAmount, Stack ∗grabbedStackPtr, Card &destinationCard, Stack ∗destinationStackPtr)
- myEvent (int type, int movedCardsAmount, Stack ∗grabbedStackPtr, Stack ∗destinationStackPtr)
- myEvent (int type, int movedCardsAmount)
- int getType ()
- int getMovedCardsAmount ()
- Stack ∗ getGrabbedStackPtr ()
- Card getDestinationCard ()
- Stack ∗ getDestinationStackPtr ()
- bool getChangedTexture ()
- void setChangedTexture (bool x)

4.4.1 Constructor & Destructor Documentation

4.4.1.1 myEvent() [1/4]

`myEvent::myEvent ( )`

Event basic constructor

4.4.1.2 myEvent() [2/4]

```
myEvent::myEvent (
            int type,
            int movedCardsAmount,
            Stack * grabbedStackPtr,
            Card & destinationCard,
            Stack * destinationStackPtr )
```

Event constructor.

Parameters

type	Amount of cards that are being moved from stack to stack
movedCardsAmount	Grabbed stack pointer (stack from which card/s were moved)
grabbedStackPtr	Grabbed stack pointer (stack from which card/s were moved)
destinationCard	Destination card place (by reference)
destinationStackPtr	Destination stack pointer

4.4.1.3 myEvent() [3/4]

```
myEvent::myEvent (
            int type,
            int movedCardsAmount,
            Stack * grabbedStackPtr,
            Stack * destinationStackPtr )
```

Event constructor.

Parameters

type	Amount of cards that are being moved from stack to stack
movedCardsAmount	Grabbed stack pointer (stack from which card/s were moved)
grabbedStackPtr	Grabbed stack pointer (stack from which card/s were moved)
destinationStackPtr	Destination stack pointer

4.4.1.4 myEvent() [4/4]

```
myEvent::myEvent (
            int type,
            int movedCardsAmount )
```

Event constructor.

Parameters

type	Amount of cards that are being moved from stack to stack
movedCardsAmount	Grabbed stack pointer (stack from which card/s were moved)

4.4.2 Member Function Documentation

4.4.2.1 getChangedTexture()

```
bool myEvent::getChangedTexture ( )
```

Getting changedTexture variable

4.4.2.2 getDestinationCard()

```
Card myEvent::getDestinationCard ( )
```

Getting destination card method

4.4.2.3 getDestinationStackPtr()

```
Stack * myEvent::getDestinationStackPtr ( )
```

Getting pointer to destination stack method

4.4.2.4 getGrabbedStackPtr()

```
Stack * myEvent::getGrabbedStackPtr ( )
```

Getting pointer to grabbed stack method

4.4.2.5 getMovedCardsAmount()

```
int myEvent::getMovedCardsAmount ( )
```

Getting amount of moved cards method

4.4.2.6 getType()

```
int myEvent::getType ( )
```

Getting event type method

4.4.2.7 setChangedTexture()

```
void myEvent::setChangedTexture (
            bool x )
```

Setting changedTexture variable.

Parameters

x	ChangedTexture variable

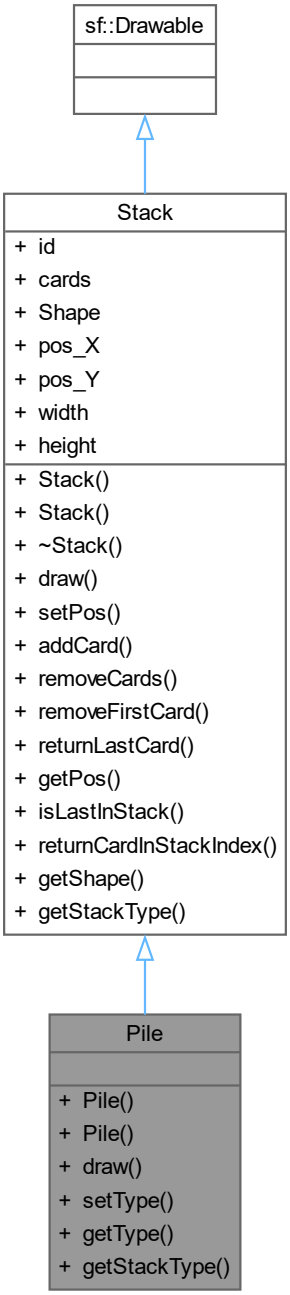
The documentation for this class was generated from the following files:

- myEvent.h
- myEvent.cpp

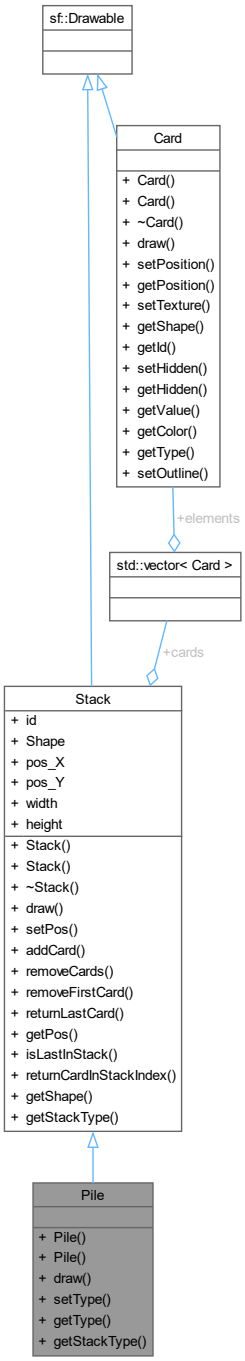
4.5 Pile Class Reference

`#include <Pile.h>`

Inheritance diagram for Pile:



Collaboration diagram for Pile:



Public Member Functions

- Pile ()
- Pile (float RectangleX, float RectangleY, std::string stackType)
- void draw (RenderTarget &target, RenderStates state) const override
- void setType (std::string x)
- std::string getType ()
- std::string getStackType ()

Public Member Functions inherited from Stack

- Stack (const int number, float RectangleX, float RectangleY, std::string stackType)
- Stack ()
- ~Stack ()=default
- virtual void draw (RenderTarget &target, RenderStates state) const override
- virtual void setPos (float x, float y)
- virtual void addCard (Card card)
- virtual void removeCards (int number)
- virtual void removeFirstCard ()
- virtual Card returnLastCard ()
- std::pair< float, float > getPos ()
- bool isLastInStack (Card &card)
- int returnCardInStackIndex (int card)
- RectangleShape getShape ()
- std::string getStackType ()

Additional Inherited Members

Public Attributes inherited from Stack

- int id
- std::vector< Card > cards
- RectangleShape Shape
- float pos_X
- float pos_Y
- const float width = 60
- const float height = 90

4.5.1 Constructor & Destructor Documentation

4.5.1.1 Pile() [1/2]

```
Pile::Pile ( )
```

Pile basic constructor

4.5.1.2 Pile() [2/2]

```
Pile::Pile (
            float RectangleX,
            float RectangleY,
            std::string stackType )
```

Pile construcor.

Parameters

RectangleX	Pile position (x)
RectangleY	Pile position (y)
stackType	Stack type

4.5.2 Member Function Documentation

4.5.2.1 draw()

```
void Pile::draw (
            RenderTarget & target,
            RenderStates state ) const  [override], [virtual]
```

Pile drawing method (SFML)

Reimplemented from Stack.

4.5.2.2 getStackType()

```
std::string Pile::getStackType ( )
```

Getting stack type method

4.5.2.3 getType()

```
std::string Pile::getType ( )
```

Getting pile type method

4.5.2.4 setType()

```
void Pile::setType (
            std::string x )
```

Setting pile type method.

Parameters

x	Type that we're setting

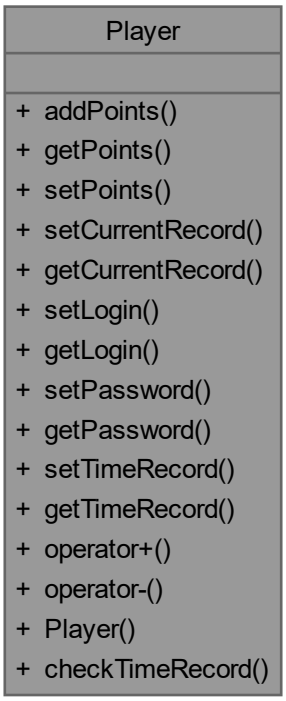
The documentation for this class was generated from the following files:

- Pile.h
- Pile.cpp

4.6 Player Class Reference

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

Collaboration diagram for Player:



Public Member Functions

- void addPoints (int points)
- int getPoints ()
- void setPoints (int points)
- void setCurrentRecord (int record)
- int getCurrentRecord ()
- void setLogin (std::string login)
- std::string getLogin ()
- void setPassword (std::string password)
- std::string getPassword ()
- void setTimeRecord (std::string timeRecord)
- std::string getTimeRecord ()
- Player operator+ (int points)
- Player operator- (int points)
- Player (std::string login, std::string password, int record, std::string timeRecord)
- void checkTimeRecord (Text &time, int hours, int minutes, int seconds)

4.6.1 Constructor & Destructor Documentation

4.6.1.1 Player()

```
Player::Player (
            std::string login,
```

```
            std::string password,
            int record,
            std::string timeRecord )
```

Player constructor.

Parameters

login	Login
password	Password
record	Record
timeRecord	Time record

4.6.2 Member Function Documentation

4.6.2.1 addPoints()

```
void Player::addPoints (
            int points )
```

Adding points method.

Parameters

points	Points value

4.6.2.2 checkTimeRecord()

```
void Player::checkTimeRecord (
            Text & time,
            int hours,
            int minutes,
            int seconds )
```

Method that checks if player set new time record.

Parameters

time	Previous saved time record text
hours	Current time hours
minutes	Current time minutes
seconds	Current time seconds

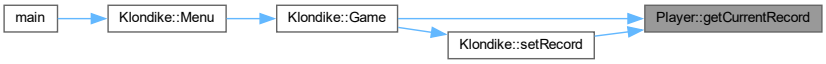
Here is the call graph for this function:

Here is the caller graph for this function:

4.6.2.3 getCurrentRecord()

```
int Player::getCurrentRecord ( )
```

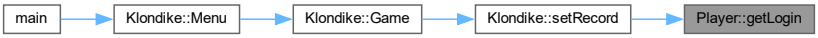
Getting current record value Here is the caller graph for this function:



4.6.2.4 getLogin()

```
std::string Player::getLogin ( )
```

Getting login method Here is the caller graph for this function:



4.6.2.5 getPassword()

```
std::string Player::getPassword ( )
```

Getting password method Here is the caller graph for this function:

4.6.2.6 getPoints()

```
int Player::getPoints ( )
```

Getting points method Here is the caller graph for this function:

4.6.2.7 getTimeRecord()

```
std::string Player::getTimeRecord ( )
```

Getting time record method Here is the caller graph for this function:

4.6.2.8 operator+()

```
Player Player::operator+ (
            int points )
```

Operator made in order to add player points in a different way.

Parameters

points	value

4.6.2.9 operator-()

```
Player Player::operator- (
            int points )
```

Operator made in order to subtract player points in a different way.

Parameters

points	value

4.6.2.10 setCurrentRecord()

```
void Player::setCurrentRecord (
            int record )
```

Setting current record method.

Parameters

record	Record (points) value

Here is the caller graph for this function:

4.6.2.11 setLogin()

```
void Player::setLogin (
            std::string login )
```

Setting login method.

Parameters

login	Login

4.6.2.12 setPassword()

```
void Player::setPassword (
            std::string password )
```

Setting password method.

Parameters

password	Password

4.6.2.13 setPoints()

```
void Player::setPoints (
            int points )
```

Setting a fixed value of points method Here is the caller graph for this function:

4.6.2.14 setTimeRecord()

```
void Player::setTimeRecord (
            std::string timeRecord )
```

Setting time record method.

Parameters

timeRecord	Time record (xx::yy::zz)

Here is the caller graph for this function:

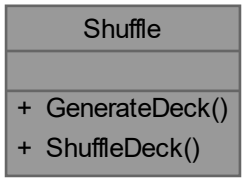
The documentation for this class was generated from the following files:

- Player.h
- Player.cpp

4.7 Shuffle Class Reference

```
#include <Shuffle.h>
```

Collaboration diagram for Shuffle:



Static Public Member Functions

- static std::vector< Card > GenerateDeck ()
- static std::vector< Card > ShuffleDeck (std::vector< Card > Deck)

4.7.1 Member Function Documentation

4.7.1.1 GenerateDeck()

```
std::vector< Card > Shuffle::GenerateDeck ( )  [static]
```

Generating deck method Here is the caller graph for this function:

4.7.1.2 ShuffleDeck()

```
std::vector< Card > Shuffle::ShuffleDeck (
            std::vector< Card > Deck )  [static]
```

Shuffling deck method.

Parameters

Deck	Deck we want to shuffle

Here is the caller graph for this function:

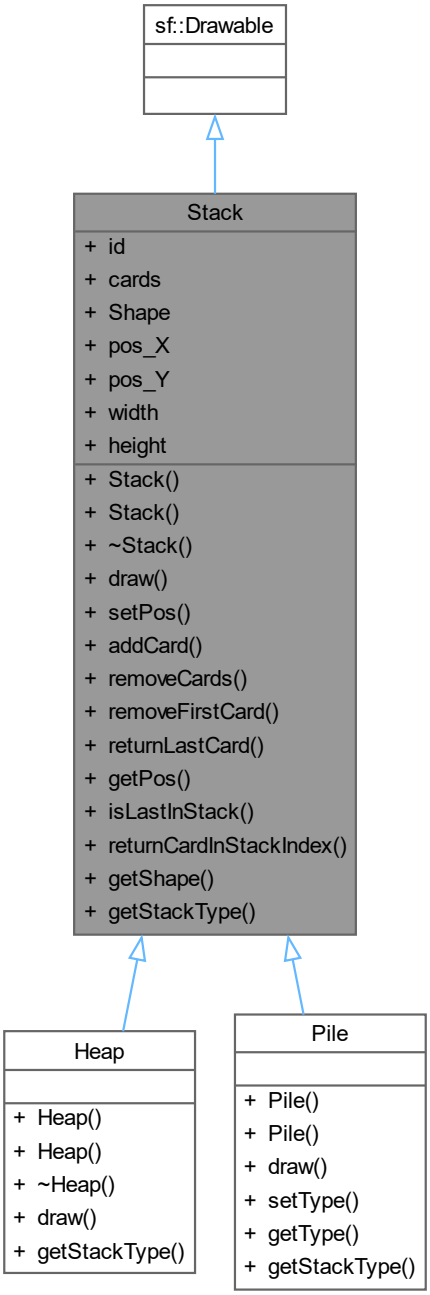
The documentation for this class was generated from the following files:

- Shuffle.h
- Shuffle.cpp

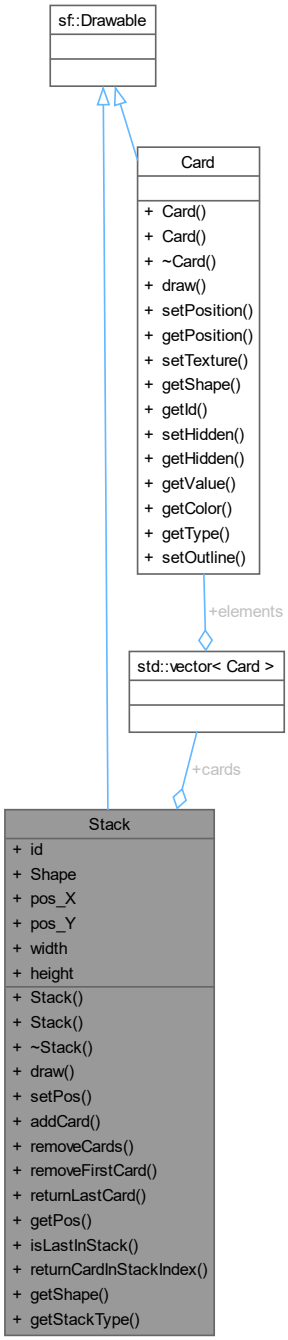
4.8 Stack Class Reference

`#include <Stack.h>`

Inheritance diagram for Stack:



Collaboration diagram for Stack:



Public Member Functions

- Stack (const int number, float RectangleX, float RectangleY, std::string stackType)
- Stack ()
- ~Stack ()=default
- virtual void draw (RenderTarget &target, RenderStates state) const override
- virtual void setPos (float x, float y)

- virtual void addCard (Card card)
- virtual void removeCards (int number)
- virtual void removeFirstCard ()
- virtual Card returnLastCard ()
- std::pair< float, float > getPos ()
- bool isLastInStack (Card &card)
- int returnCardInStackIndex (int card)
- RectangleShape getShape ()
- std::string getStackType ()

Public Attributes

- int id
- std::vector< Card > cards
- RectangleShape Shape
- float pos_X
- float pos_Y
- const float width = 60
- const float height = 90

4.8.1 Constructor & Destructor Documentation

4.8.1.1 Stack() [1/2]

```
Stack::Stack (
            const int number,
            float RectangleX,
            float RectangleY,
            std::string stackType )
```

Stack constructor.

Parameters

number	Stack ID
RectangleX	Stack position (x)
RectangleY	Stack position (y)
stackType	Stack type

4.8.1.2 Stack() [2/2]

```
Stack::Stack ( )
```

Stack basic constructor

4.8.1.3 ~Stack()

```
Stack::~Stack ( )  [default]
```

Stack destructor

4.8.2 Member Function Documentation

4.8.2.1 addCard()

```
void Stack::addCard (
            Card card )  [virtual]
```

Adding card to stack's vector method.

Parameters

card	Card

Here is the caller graph for this function:

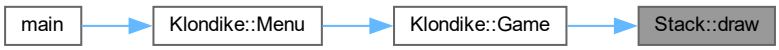
4.8.2.2 draw()

```
void Stack::draw (
            RenderTarget & target,
            RenderStates state ) const  [override], [virtual]
```

Virtual drawing method (SFML)

Reimplemented in Heap, and Pile.

Here is the caller graph for this function:



4.8.2.3 getPos()

```
std::pair< float, float > Stack::getPos ( )
```

Getting position of stack (x,y) Here is the caller graph for this function:

4.8.2.4 getShape()

```
RectangleShape Stack::getShape ( )
```

Getting stack's shape method (SFML) Here is the caller graph for this function:

4.8.2.5 getStackType()

```
std::string Stack::getStackType ( )
```

Getting stack type method Here is the caller graph for this function:

4.8.2.6 isLastInStack()

```
bool Stack::isLastInStack (
            Card & card )
```

Method that defines if card is last in stack's vector

Parameters

card	Card

Here is the call graph for this function:

Here is the caller graph for this function:

4.8.2.7 removeCards()

```
void Stack::removeCards (
            int number )  [virtual]
```

Removing x cards from the back of stack's vector method.

Parameters

number	Number of cards we want to remove

Here is the caller graph for this function:

4.8.2.8 removeFirstCard()

```
void Stack::removeFirstCard ( )  [virtual]
```

Removing first cards from vector method

4.8.2.9 returnCardInStackIndex()

```
int Stack::returnCardInStackIndex (
            int card )
```

Method that returnes index of last card in stack's vector Here is the caller graph for this function:

4.8.2.10 returnLastCard()

```
Card Stack::returnLastCard ( )  [virtual]
```

Method returning last card in stack's vector Here is the caller graph for this function:

4.8.2.11 setPos()

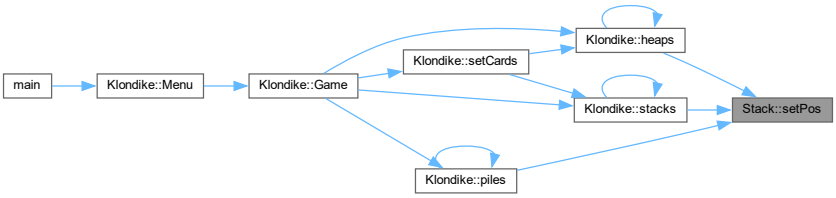
```
void Stack::setPos (
            float x,
            float y )  [virtual]
```

Virtual setting position method.

Parameters

x	X
y	Y

Here is the caller graph for this function:



4.8.3 Member Data Documentation

4.8.3.1 cards

```
std::vector<Card> Stack::cards
```

Vector of cards assigned to that stack

4.8.3.2 height

```
const float Stack::height = 90
```

Stack height (y)

4.8.3.3 id

```
int Stack::id
```

Stack ID

4.8.3.4 pos_X

```
float Stack::pos_X
```

Stack position (x)

4.8.3.5 pos_Y

```
float Stack::pos_Y
```

Stack position (y)

4.8.3.6 Shape

```
RectangleShape Stack::Shape
```

Shape of the stack (SFML)

4.8.3.7 width

```
const float Stack::width = 60
```

Stack width (x)

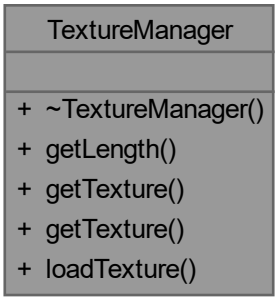
The documentation for this class was generated from the following files:

- Stack.h
- Stack.cpp

4.9 TextureManager Class Reference

```
#include <TextureManager.h>
```

Collaboration diagram for TextureManager:



Public Member Functions

- ~TextureManager ()

Static Public Member Functions

- static int getLength ()
- static sf::Texture * getTexture (string name)
- static sf::Texture * getTexture (int index)
- static sf::Texture * loadTexture (string name, string path)

4.9.1 Constructor & Destructor Documentation

4.9.1.1 ~TextureManager()

```
TextureManager::~TextureManager ( )
```

Destructor which deletes the textures previously loaded

4.9.2 Member Function Documentation

4.9.2.1 getLength()

```
int TextureManager::getLength ( )  [static]
```

4.9.2.2 getTexture() [1/2]

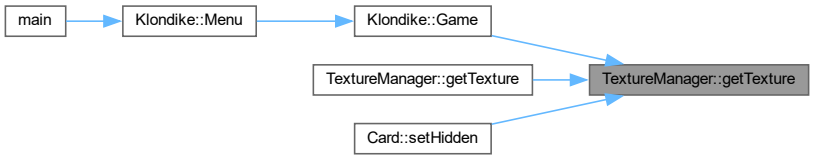
```
sf::Texture * TextureManager::getTexture (
            int index )  [static]
```

Get texutre by index in map, or return null Here is the call graph for this function:

4.9.2.3 getTexture() [2/2]

```
sf::Texture * TextureManager::getTexture (
            string name )  [static]
```

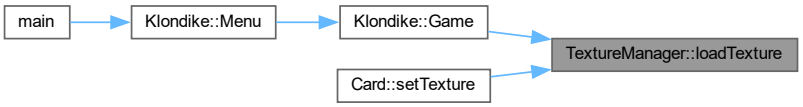
Get texutre by name specified in loadTexture, or return null Here is the caller graph for this function:



4.9.2.4 loadTexture()

```
sf::Texture * TextureManager::loadTexture (
            string name,
            string path )  [static]
```

Loads the texture and returns a pointer to it. If it is already loaded, this function just returns it. If it cannot find the file, returns NULL Here is the caller graph for this function:



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

- TextureManager.h
- TextureManager.cpp

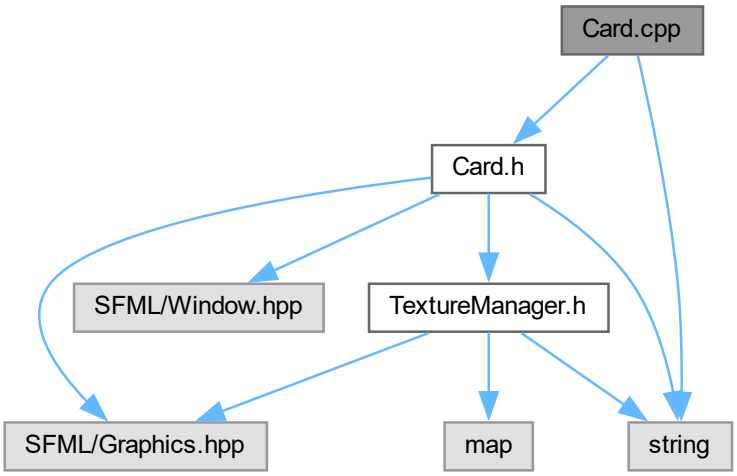
Chapter 5

File Documentation

5.1 Card.cpp File Reference

```
#include ¨Card.h¨
#include <string>
```

Include dependency graph for Card.cpp:



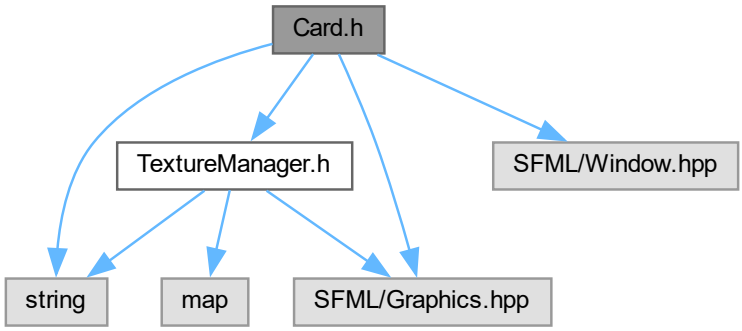
5.2 Card.h File Reference

Card class.

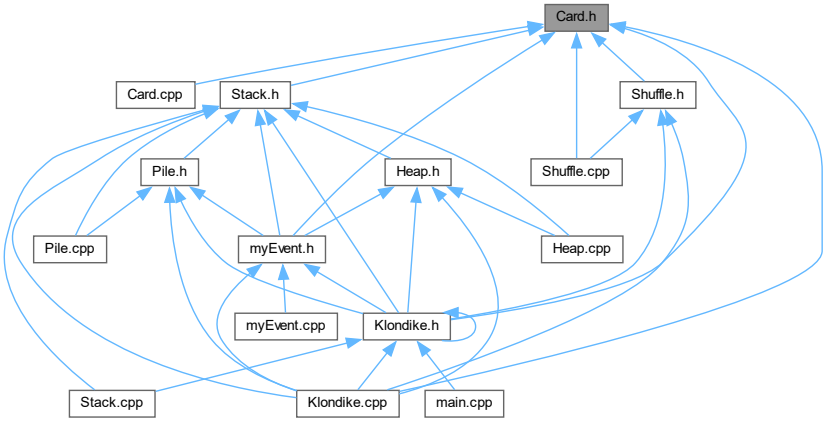
```
#include <string>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
```

```
#include ¨TextureManager.h¨
```

Include dependency graph for Card.h:



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



Classes

- class Card

5.2.1 Detailed Description

Card class.

Author

Karol Ziaja

Date

August 2023

5.3 Card.h

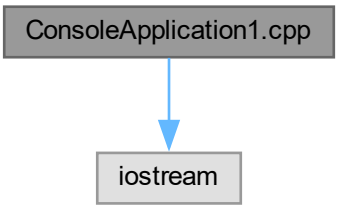
Go to the documentation of this file.

```
00001 /*****************************************************************/
00009 #pragma once
00010
00011 #include <string>
00012 #include <SFML/Graphics.hpp>
00013 #include <SFML/Window.hpp>
00014 #include "TextureManager.h"
00015
00016 using namespace sf;
00017
00018 class Card : public sf::Drawable {
00020     int index;
00022     int Value;
00024     std::string Color;
00026     std::string Type;
00028     std::string texturePath;
00030     RectangleShape cardShape;
00032     float width = 65;
00034     float height = 90;
00036     bool hidden;
00038     float currentPositionX;
00040     float currentPositionY;
00041 public:
00043     Card();
00045     Card(int index, int value, std::string Color, std::string Type, std::string texturePath);
00047     ~Card() = default;
00049     void draw(RenderTarget& target, RenderStates state) const override;
00051     void setPosition(float X, float Y);
00053     std::pair<float, float> getPosition();
00055     void setTexture();
00057     RectangleShape getShape();
00059     int getId();
00065     void setHidden(const bool i);
00067     bool getHidden();
00069     int getValue();
00071     std::string getColor();
00073     std::string getType();
00075     void setOutline();
00076 };
00077
00078 //1-A
00079 //2-2
00080 //3-3
00081 //4-4
00082 //5-5
00083 //6-6
00084 //7-7
00085 //8-8
00086 //9-9
00087 //10-10
00088 //11-J
00089 //12-Q
00090 //13-K
00091
```

5.4 ConsoleApplication1.cpp File Reference

```
#include <iostream>
```

Include dependency graph for ConsoleApplication1.cpp:



Functions

- int main ()

5.4.1 Function Documentation

5.4.1.1 main()

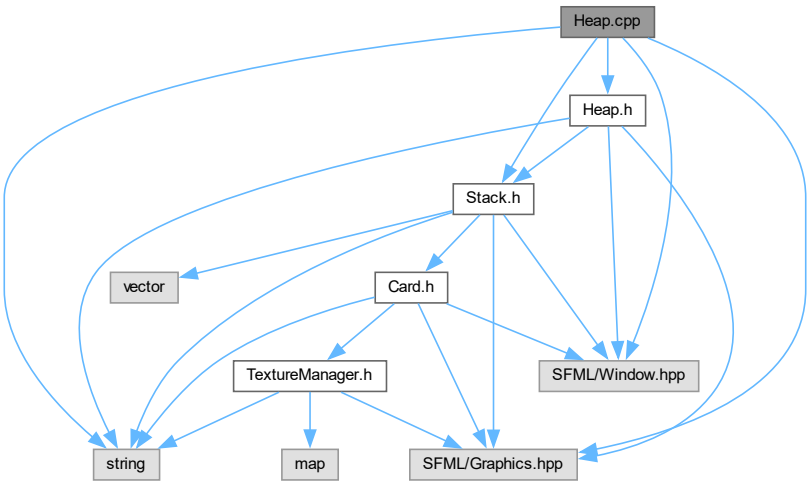
```
int main ( )
```

5.5 Heap.cpp File Reference

```
#include ¨Stack.h¨
#include ¨Heap.h¨
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
```

```
#include <string>
```

Include dependency graph for Heap.cpp:

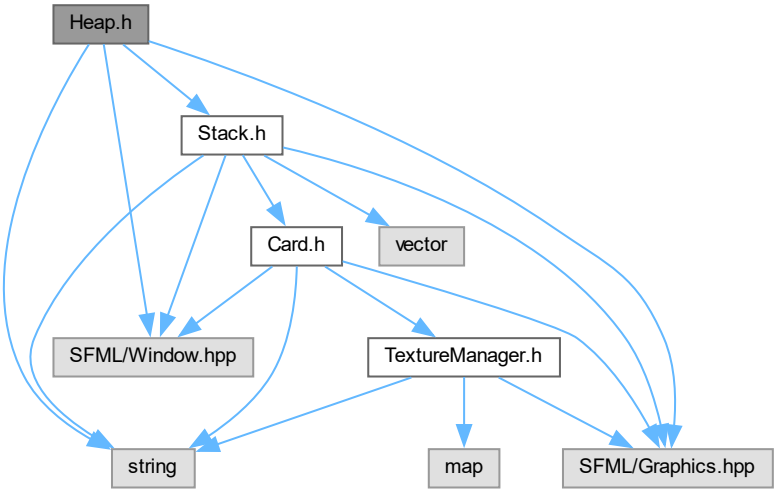


5.6 Heap.h File Reference

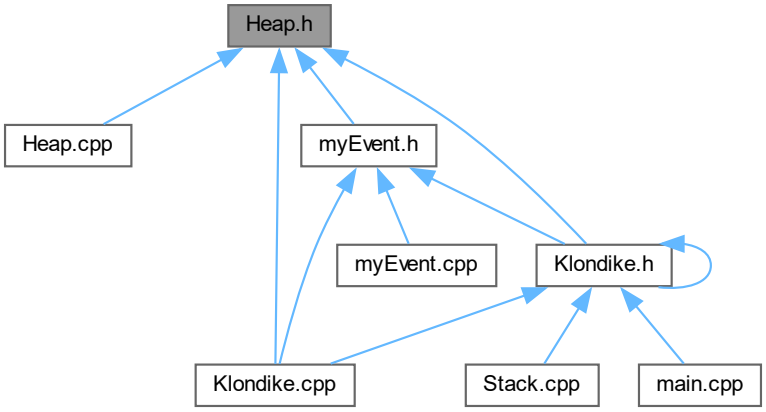
Heap of cards in top left corner (class)

```
#include <string>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include ¨Stack.h¨
```

Include dependency graph for Heap.h:



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



Classes

- class Heap

5.6.1 Detailed Description

Heap of cards in top left corner (class)

Author

Karol Ziaja

Date

August 2023

5.7 Heap.h

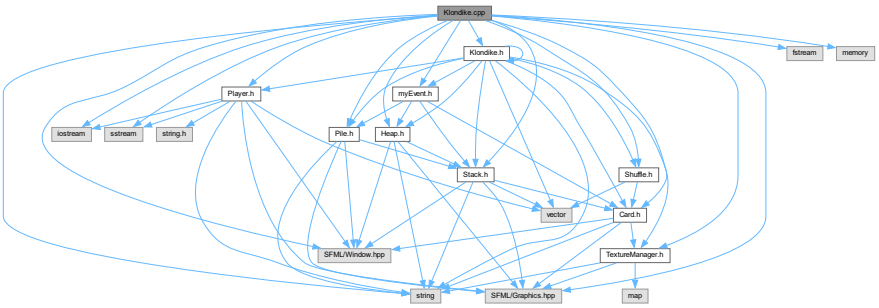
Go to the documentation of this file.

```
00001 /*****************************************************************/
00009 #pragma once
00010
00011 #include <string>
00012 #include <SFML/Graphics.hpp>
00013 #include <SFML/Window.hpp>
00014 #include "Stack.h"
00015 using namespace sf;
00016
00017 class Heap : public Stack {
00019     std::string stackType;
00020 public:
00022     Heap();
00024     Heap(float RectangleX, float RectangleY, std::string stackType);
00026     ~Heap() = default;
00028     void draw(RenderTarget& target, RenderStates state) const override;
00030     std::string getStackType();
00031 };
```

5.8 Klondike.cpp File Reference

```
#include <iostream>
#include <sstream>
#include <string>
#include <fstream>
#include "Klondike.h"
#include "Stack.h"
#include "Pile.h"
#include "Heap.h"
#include "Shuffle.h"
#include "myEvent.h"
#include "TextureManager.h"
#include "Card.h"
#include "Player.h"
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <memory>
```

Include dependency graph for Klondike.cpp:

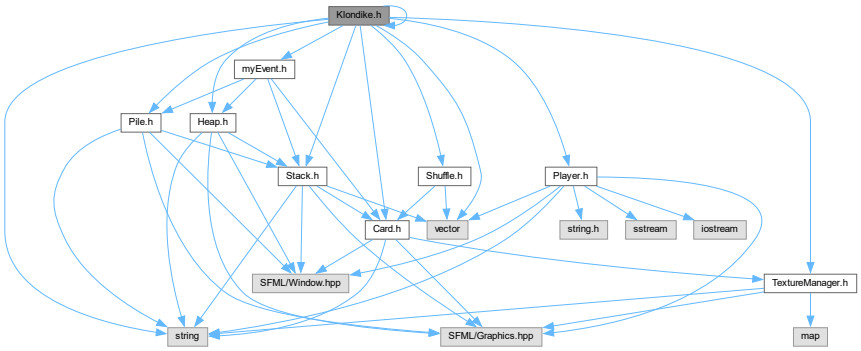


5.9 Klondike.h File Reference

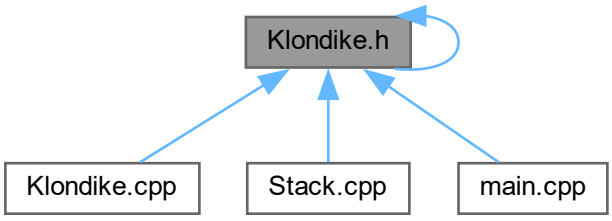
Main Game Class.

```
#include <vector>
#include <string>
#include "Klondike.h"
#include "Stack.h"
#include "Pile.h"
#include "Heap.h"
#include "Shuffle.h"
#include "myEvent.h"
#include "TextureManager.h"
#include "Card.h"
#include "Player.h"
```

Include dependency graph for Klondike.h:



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



Classes

- class Klondike

5.9.1 Detailed Description

Main Game Class.

Author

Karol Ziaja

Date

August 2023

5.10 Klondike.h

Go to the documentation of this file.

```
00001 /*****************************************************************/
00009 #pragma once
00010
00011 #include <vector>
00012 #include <string>
00013 #include "Klondike.h"
00014 #include "Stack.h"
00015 #include "Pile.h"
00016 #include "Heap.h"
00017 #include "Shuffle.h"
00018 #include "myEvent.h"
00019 #include "TextureManager.h"
00020 #include "Card.h"
00021 #include "Player.h"
00022
00023 class Klondike {
00024 public:
00026     static std::pair<float, float> screenSize;
00028     static std::string choice;
00030     static Player player;
00032     static std::string login;
00034     static std::string password;
00036     static int record;
00038     static std::string timeRecord;
00040     static void Game();
00042     static void Menu();
00044     static void Register();
00046     static bool Login(std::string& loginPlayer, std::string& passwordPlayer, int& record, std::string&
      timeRecord);
00048     static void setRecord(Player& player);
00050     static void Tutorial();
00052     static std::vector<Stack> stacks();
00054     static std::vector<Pile> piles();
00056     static std::vector<Heap> heaps();
00058     static std::vector<float> coords();
00068     static void setCards(std::vector<Stack> &stacks, std::vector<Pile> &piles, std::vector<Heap>
      &heaps, std::vector<Card> &cards, std::vector<float>& coords);
00069 };
```

5.11 Lib.h File Reference

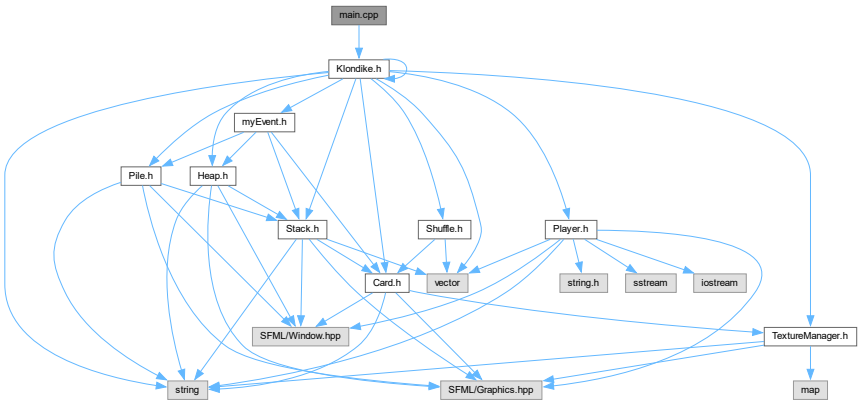
5.12 Lib.h

Go to the documentation of this file.

5.13 main.cpp File Reference

```
#include ¨Klondike.h¨
```

Include dependency graph for main.cpp:



Functions

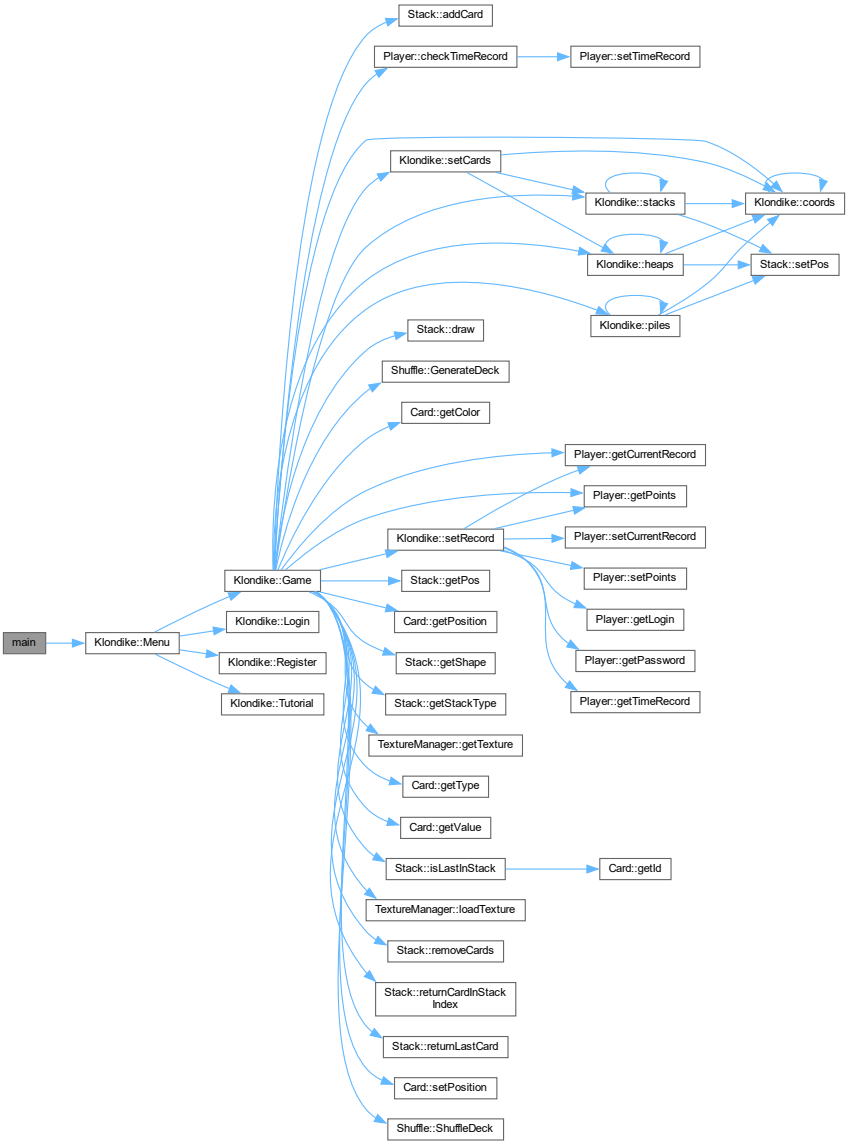
- int main ()

5.13.1 Function Documentation

5.13.1.1 main()

```
int main ( )
```

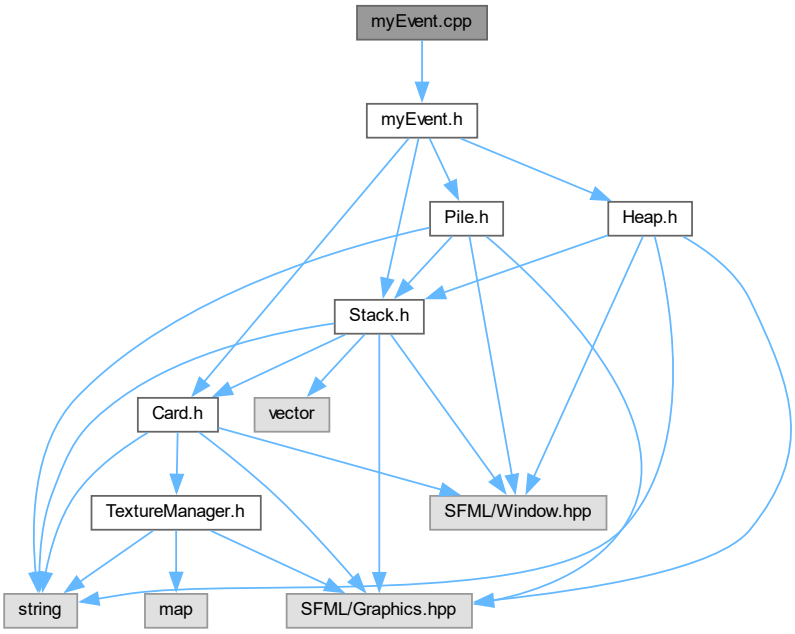
Here is the call graph for this function:



5.14 myEvent.cpp File Reference

```
#include ¨myEvent.h¨
```

Include dependency graph for myEvent.cpp:

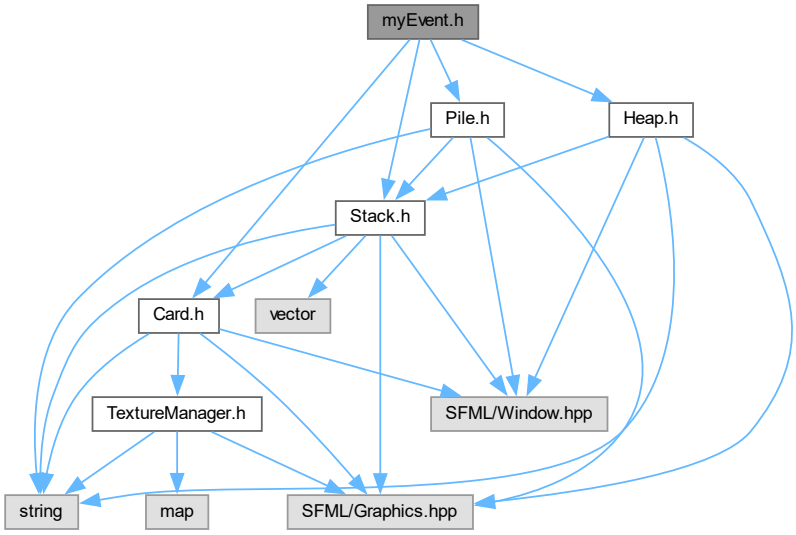


5.15 myEvent.h File Reference

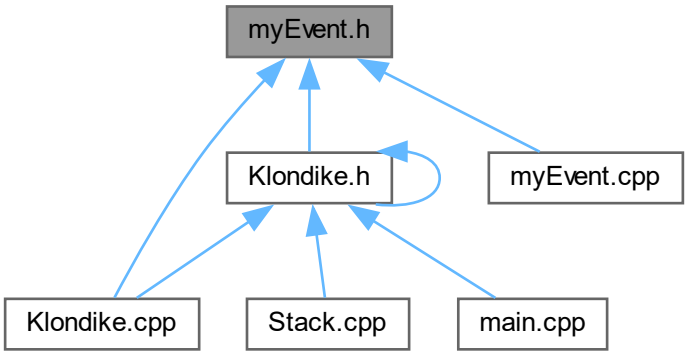
Class made to enable undoing movements. Named myEvent because Event already exists as a part of SFML library.

```
#include ¨Card.h¨
#include ¨Stack.h¨
#include ¨Pile.h¨
#include ¨Heap.h¨
```

Include dependency graph for myEvent.h:



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



Classes

- class myEvent

5.15.1 Detailed Description

Class made to enable undoing movements. Named myEvent because Event already exists as a part of SFML library.

Author

Karol Ziaja

Date

August 2023

5.16 myEvent.h

Go to the documentation of this file.

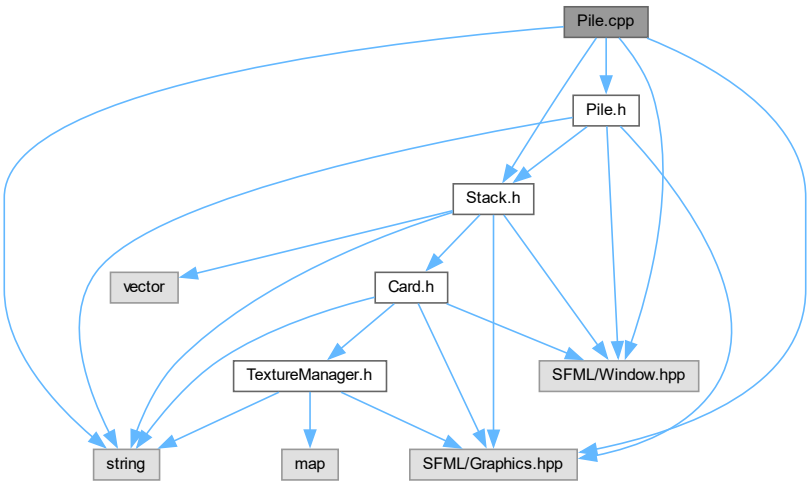
```
00001 /*************************************************************/
00009 #pragma once
00010
00011 #include "Card.h"
00012 #include "Stack.h"
00013 #include "Pile.h"
00014 #include "Heap.h"
00015
00016 class myEvent {
00018     int type;
00020     int movedCardsAmount;
00022     Stack* grabbedStackPtr;
00024     Card destinationCard;
00026     Stack* destinationStackPtr;
00028     bool changedTexture;
00029 public:
00031     myEvent();
00041     myEvent(int type, int movedCardsAmount, Stack* grabbedStackPtr, Card& destinationCard, Stack*
      destinationStackPtr);
00050     myEvent(int type, int movedCardsAmount, Stack* grabbedStackPtr, Stack* destinationStackPtr);
00057     myEvent(int type, int movedCardsAmount);
00059     int getType();
00061     int getMovedCardsAmount();
00063     Stack* getGrabbedStackPtr();
00065     Card getDestinationCard();
00067     Stack* getDestinationStackPtr();
00069     bool getChangedTexture();
00075     void setChangedTexture(bool x);
00076 };
00077
00078
00079 //Types
00080 //1-Card To Stack
00081 //2-Cards to Stack
00082 //3-King to Stack
00083 //4-King and more cards from Stack to Stack
00084 //5-Ace to Pile
00085 //6-Cards to Pile
00086 //7-empty heap1
00087 //8-card to heap2
```

5.17 Pile.cpp File Reference

```
#include ¨Pile.h¨
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include ¨Stack.h¨
```

```
#include <string>
```

Include dependency graph for Pile.cpp:

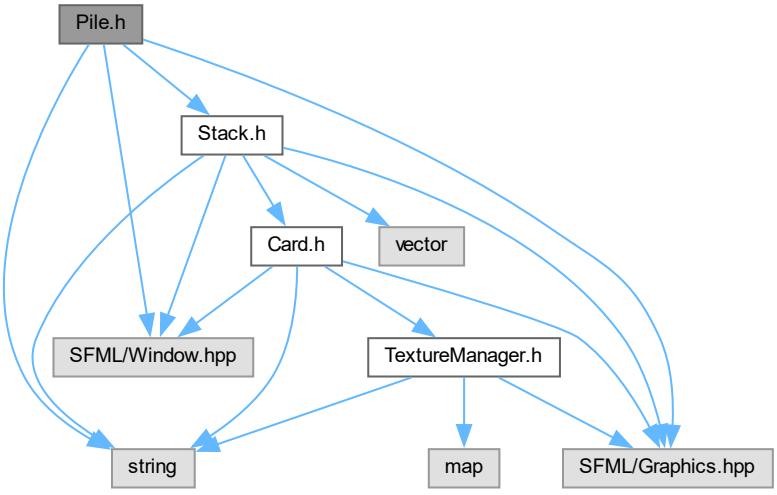


5.18 Pile.h File Reference

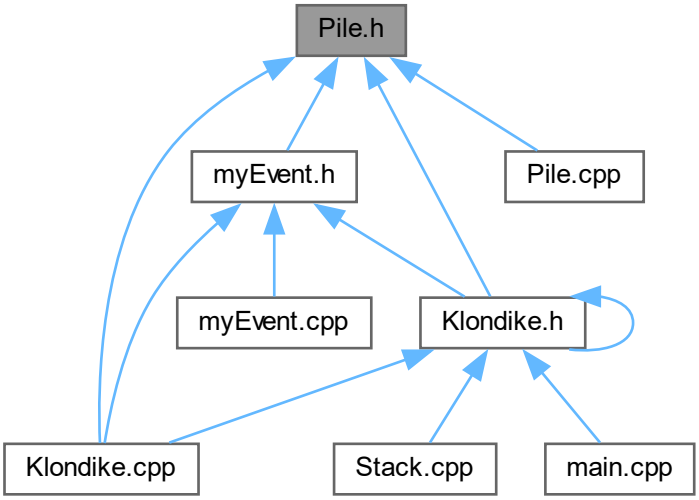
Four (collecting) piles of cards at the top (class)

```
#include ¨Stack.h¨
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <string>
```

Include dependency graph for Pile.h:



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



Classes

- class Pile

5.18.1 Detailed Description

Four (collecting) piles of cards at the top (class)

Author

Karol Ziaja

Date

August 2023

5.19 Pile.h

Go to the documentation of this file.

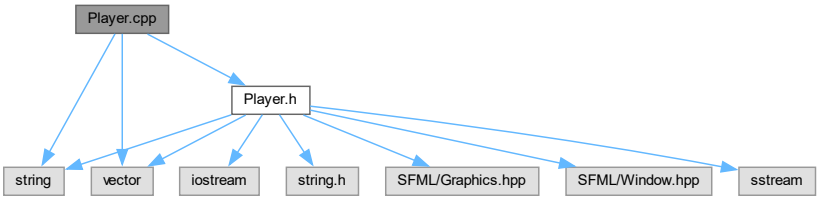
```
00001 /*************************************************************/
00009 #pragma once
00010
00011 #include "Stack.h"
00012 #include <SFML/Graphics.hpp>
00013 #include <SFML/Window.hpp>
00014 #include <string>
00015 using namespace sf;
00016
00017 class Pile : public Stack {
```

```
00019     std::string Type;
00021     std::string stackType;
00022 public:
00024     Pile();
00032     Pile(float RectangleX, float RectangleY, std::string stackType);
00034     void draw(RenderTarget& target, RenderStates state) const override;
00040     void setType(std::string x);
00042     std::string getType();
00044     std::string getStackType();
00045 };
```

5.20 Player.cpp File Reference

```
#include ¨Player.h¨
#include <string>
#include <vector>
```

Include dependency graph for Player.cpp:

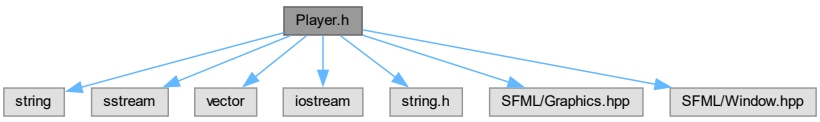


5.21 Player.h File Reference

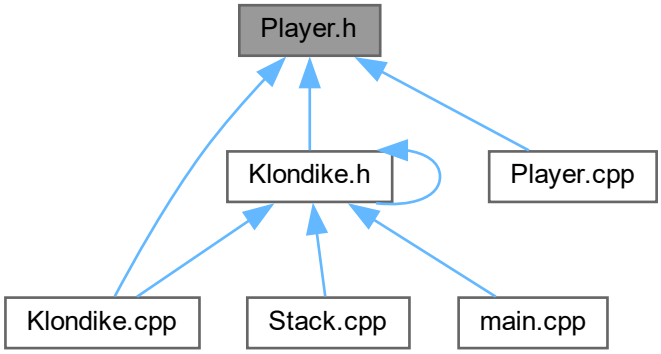
Current player data class.

```
#include <string>
#include <sstream>
#include <vector>
#include <iostream>
#include <string.h>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
```

Include dependency graph for Player.h:



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



Classes

- class Player

5.21.1 Detailed Description

Current player data class.

Author

Karol Ziaja

Date

August 2023

5.22 Player.h

Go to the documentation of this file.

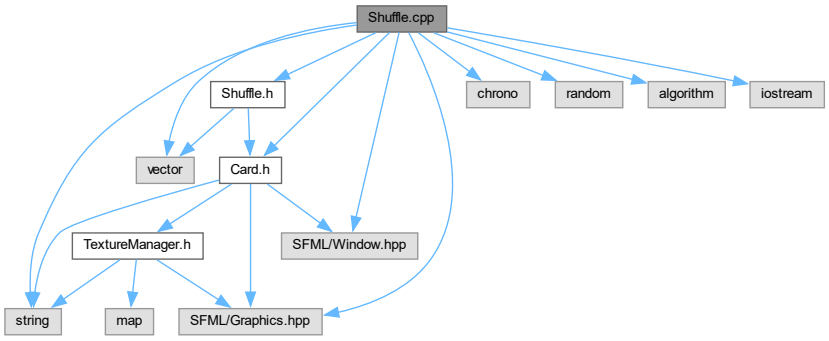
```
00001 /*************************************************************************/
00009 #pragma once
00010
00011 #include <string>
00012 #include <sstream>
00013 #include <vector>
00014 #include <iostream>
00015 #include <string.h>
00016 #include <SFML/Graphics.hpp>
00017 #include <SFML/Window.hpp>
00018 using namespace sf;
00019
00020 class Player {
00022     int points;
00024     int currentRecord;
00026     std::string Login;
00028     std::string Password;
```

```
00030     std::string timeRecord;
00031 public:
00037     void addPoints(int points);
00039     int getPoints();
00041     void setPoints(int points);
00047     void setCurrentRecord(int record);
00049     int getCurrentRecord();
00055     void setLogin(std::string login);
00057     std::string getLogin();
00063     void setPassword(std::string password);
00065     std::string getPassword();
00071     void setTimeRecord(std::string timeRecord);
00073     std::string getTimeRecord();
00079     Player operator+(int points);
00085     Player operator-(int points);
00094     Player(std::string login, std::string password, int record, std::string timeRecord);
00103     void checkTimeRecord(Text& time, int hours, int minutes, int seconds);
00104 };
```

5.23 Shuffle.cpp File Reference

```
#include ¨Shuffle.h¨
#include ¨Card.h¨
#include <chrono>
#include <random>
#include <algorithm>
#include <iostream>
#include <vector>
#include <string>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
```

Include dependency graph for Shuffle.cpp:

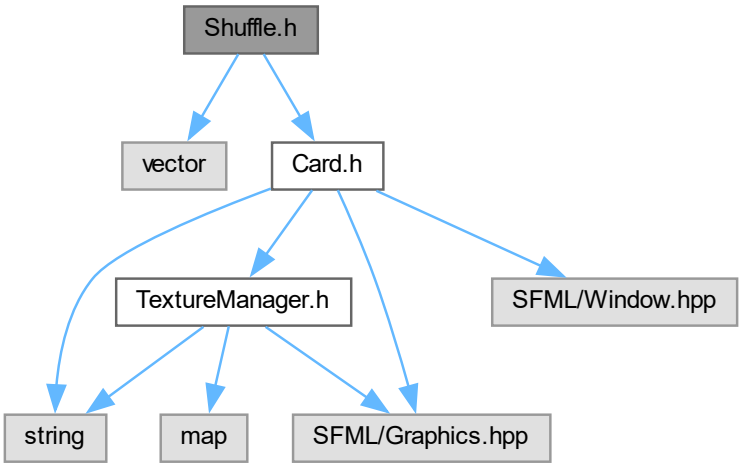


5.24 Shuffle.h File Reference

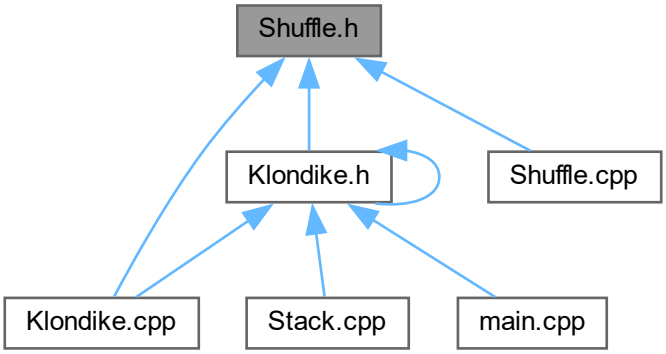
Class being used to generate and shuffle card deck.

```
#include <vector>
#include ¨Card.h¨
```

Include dependency graph for Shuffle.h:



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



Classes

- class Shuffle

5.24.1 Detailed Description

Class being used to generate and shuffle card deck.

Author

Karol Ziaja

Date

August 2023

5.25 Shuffle.h

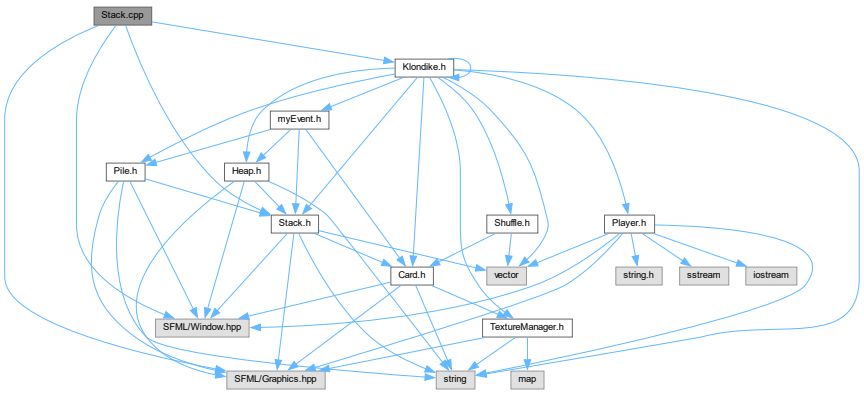
Go to the documentation of this file.

```
00001 /*****************************************************************/
00009 #pragma once
00010
00011 #include <vector>
00012 #include "Card.h"
00013
00014 class Shuffle {
00015 public:
00017     static std::vector<Card> GenerateDeck();
00023     static std::vector<Card> ShuffleDeck(std::vector<Card> Deck);
00024 };
```

5.26 Stack.cpp File Reference

```
#include ¨Stack.h¨
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include ¨Klondike.h¨
```

Include dependency graph for Stack.cpp:



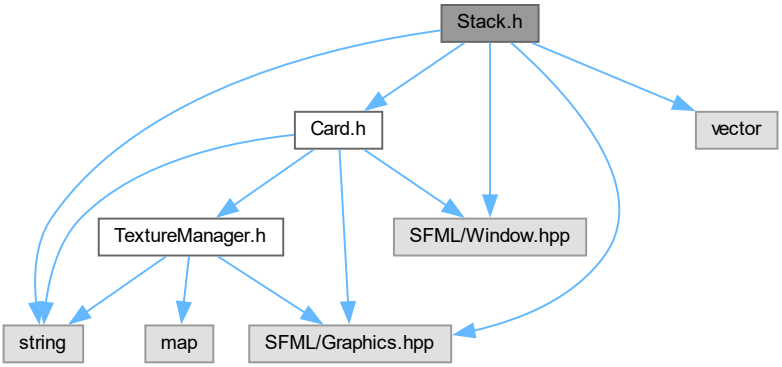
5.27 Stack.h File Reference

Stack class, heap and pile classes inherit from this class.

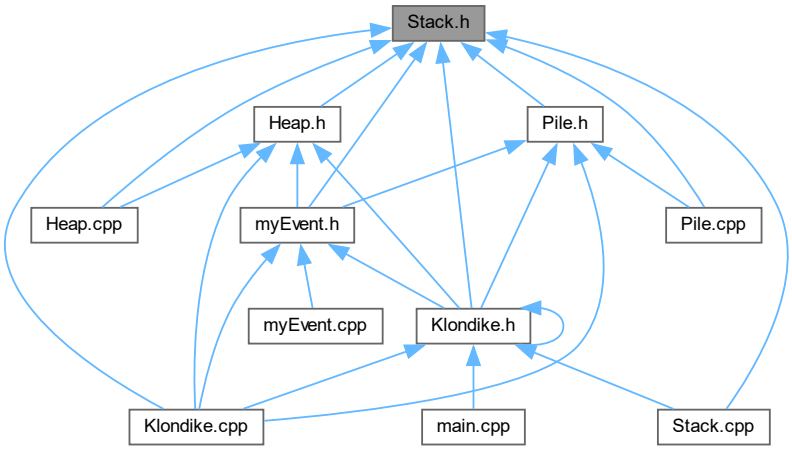
```
#include ¨Card.h¨
#include <string>
#include <vector>
```

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

Include dependency graph for Stack.h:



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



Classes

- class Stack

5.27.1 Detailed Description

Stack class, heap and pile classes inherit from this class.

Author

Karol Ziaja

Date

August 2023

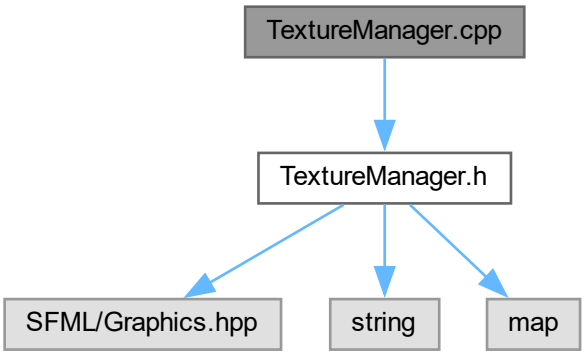
5.28 Stack.h

Go to the documentation of this file.

```
00001 /*****************************************************************/
00009 #pragma once
00010
00011 #include "Card.h"
00012 #include <string>
00013 #include <vector>
00014 #include <SFML/Graphics.hpp>
00015 #include <SFML/Window.hpp>
00016
00017 using namespace sf;
00018
00019 class Stack : public sf::Drawable{
00021     std::string stackType;
00022 public:
00024     int id;
00026     std::vector<Card> cards;
00028     RectangleShape Shape;
00030     float pos_X;
00032     float pos_Y;
00034     const float width = 60;
00036     const float height = 90;
00045     Stack(const int number, float RectangleX, float RectangleY, std::string stackType);
00047     Stack();
00049     ~Stack() = default;
00051     void virtual draw(RenderTarget& target, RenderStates state) const override;
00058     void virtual setPos(float x, float y);
00064     void virtual addCard(Card card);
00070     void virtual removeCards(int number);
00072     void virtual removeFirstCard();
00074     Card virtual returnLastCard();
00076     std::pair<float, float> getPos();
00082     bool isLastInStack(Card& card);
00084     int returnCardInStackIndex(int card);
00086     RectangleShape getShape();
00088     std::string getStackType();
00089 };
00090
00091
```

5.29 TextureManager.cpp File Reference

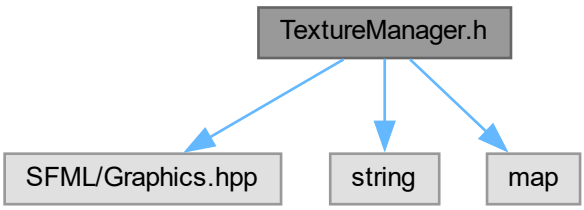
```
#include ¨TextureManager.h¨
```
Include dependency graph for TextureManager.cpp:



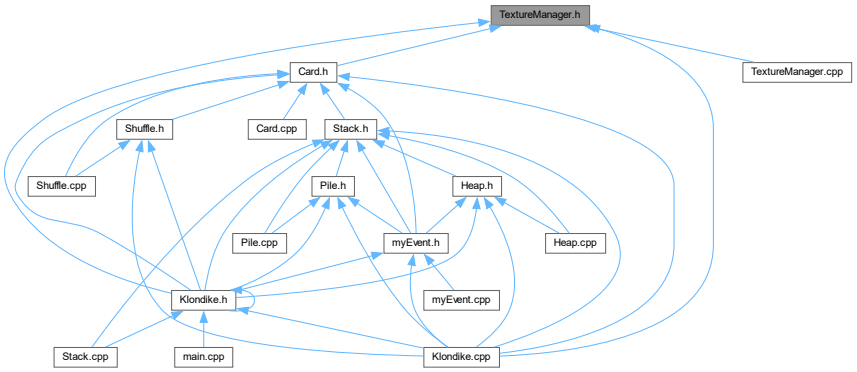
5.30 TextureManager.h File Reference

Class that manages all loaded textures from files.

```
#include <SFML/Graphics.hpp>
#include <string>
#include <map>
```
Include dependency graph for TextureManager.h:



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



Classes

- class TextureManager

5.30.1 Detailed Description

Class that manages all loaded textures from files.

Author

https://github.com/netpoetica

Date

August 2023

5.31 TextureManager.h

Go to the documentation of this file.

```
00001 /*************************************************************************/
00009 #ifndef TEXTUREMANAGER_H
00010 #define TEXTUREMANAGER_H
00011
00012 #include <SFML/Graphics.hpp>
00013 #include <string>
00014 #include <map>
00015
00016 using namespace std;
00017
00018 class TextureManager
00019 {
00021     static map<string, sf::Texture*> textures;
00022
00024     static std::vector<string> order;
00025
00027     TextureManager();
00028 public:
00030     ~TextureManager();
00031
00032     static int getLength();
00033
00035     static sf::Texture* getTexture(string name);
00036
00038     static sf::Texture* getTexture(int index);
00039
00041     static sf::Texture* loadTexture(string name, string path);
00042 };
00043
00044 #endif
```

Index