

Politechnika Śląska
Wydział Automatyki, Elektroniki i Informatyki

Programowanie Komputerów 2

Pasjans / Klondike

Autor	Karol Ziaja
Prowadzący	Dr inż. Wojciech Łabaj
Rok akademicki	2022/2023
Kierunek	Informatyka
Rodzaj studiów	SSI
Semestr	2
Termin laboratorium	Środa, parzysta, 13:30 – 15:00
Sekcja	62
Termin oddania sprawozdania	2023-09-01

1 Treść zadania

Napisać program w języku C++, który będzie wzorowany na klasycznej grze karcianej pasjans. W grze użytkownik może zalogować się na swoje konto, a następnie rozpocząć rozgrywkę, starając się uzyskać jak najwyższy wynik punktowy.

2 Analiza zadania

Zagadnienie przedstawia konstrukcję programu „Pasjans”, który został napisany w języku C++ z wykorzystaniem biblioteki graficznej SFML służącej do wyświetlania programu. Obsługuje ona akcelerację sprzętową grafiki 2D przy użyciu OpenGL.

2.1 Struktury danych

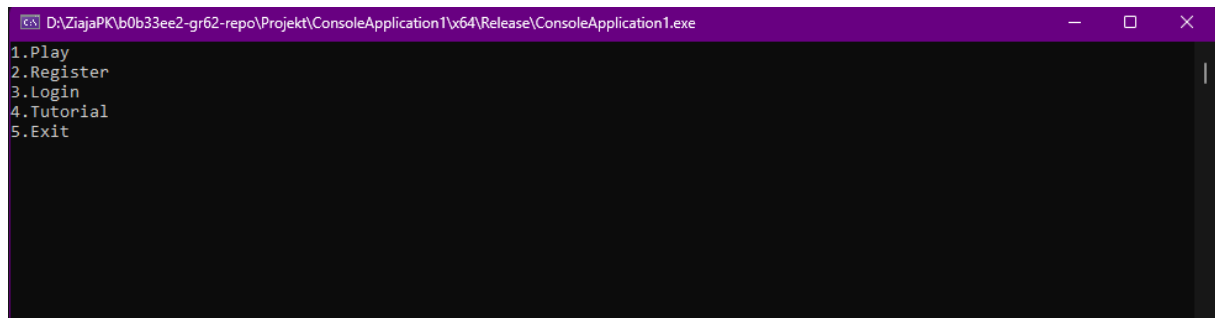
Program napisany jest obiektowo, a więc opiera się na klasach. Najważniejsze z nich to klasa „Stack” oraz klasy po niej dziedziczące i klasa „Card”. Każdy istniejący stos zawiera swój wektor kart, dzięki temu możliwa jest łatwa implementacja metod, które powodują oddziaływanie kart między sobą. Stosy i karty dziedziczą również po klasie „drawable” należącej do biblioteki SFML i umożliwiającej rysowanie obiektów. Z tych oraz wielu innych klas korzysta główna klasa „Klondike”, jest w niej zawarta główna pętla gry.

2.2 Algorytmy

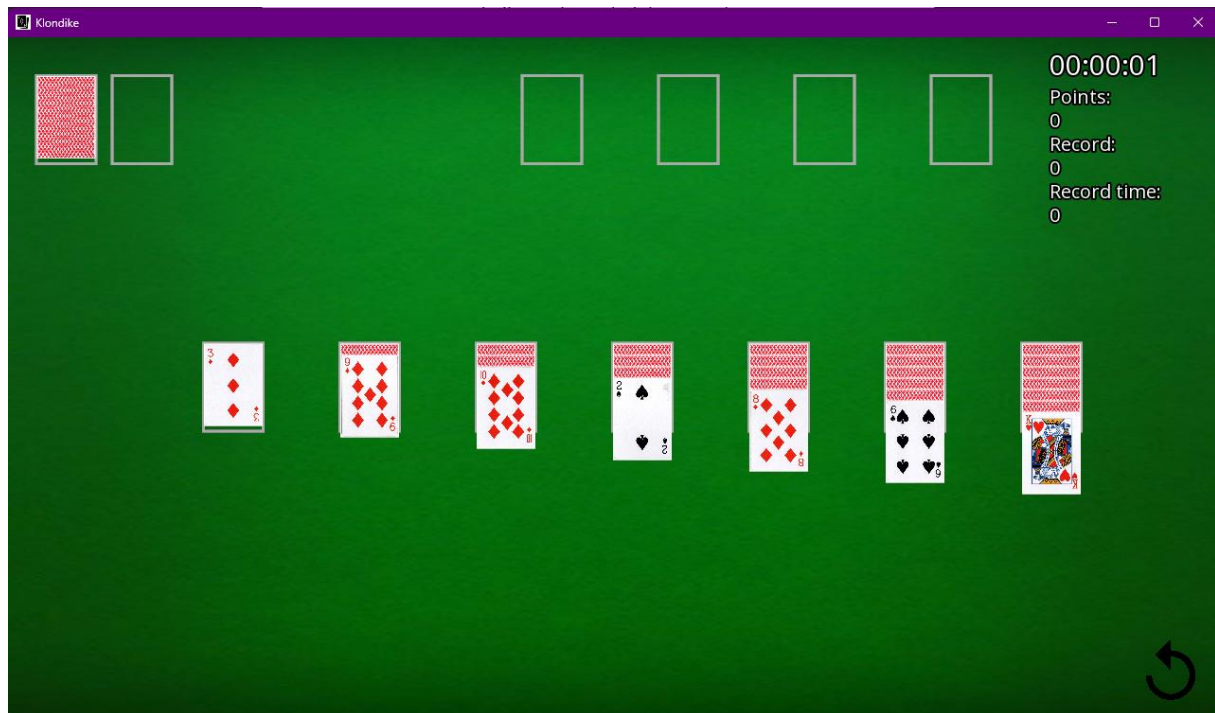
Większość algorytmów zawartych w programie znajduje się w statycznej metodzie Game klasy Klondike. To w niej zawiera się główna pętla gry. Algorytmy te definiują poprawne przenoszenie kart między każdym ze stosów zgodnie z zasadami gry Pasjans. Dodatkowo metody w klasie „Shuffle” odpowiadają za poprawne generowanie i tasowanie talii kart.

3 Specyfikacja zewnętrzna

Uruchomienie programu spowoduje pojawienie się menu w konsoli. Użytkownik ma w nim możliwość rejestracji, logowania, wyświetlenia tutorialu(instrukcji gry) oraz rozpoczęcia rozgrywki, a także wyjścia z programu.

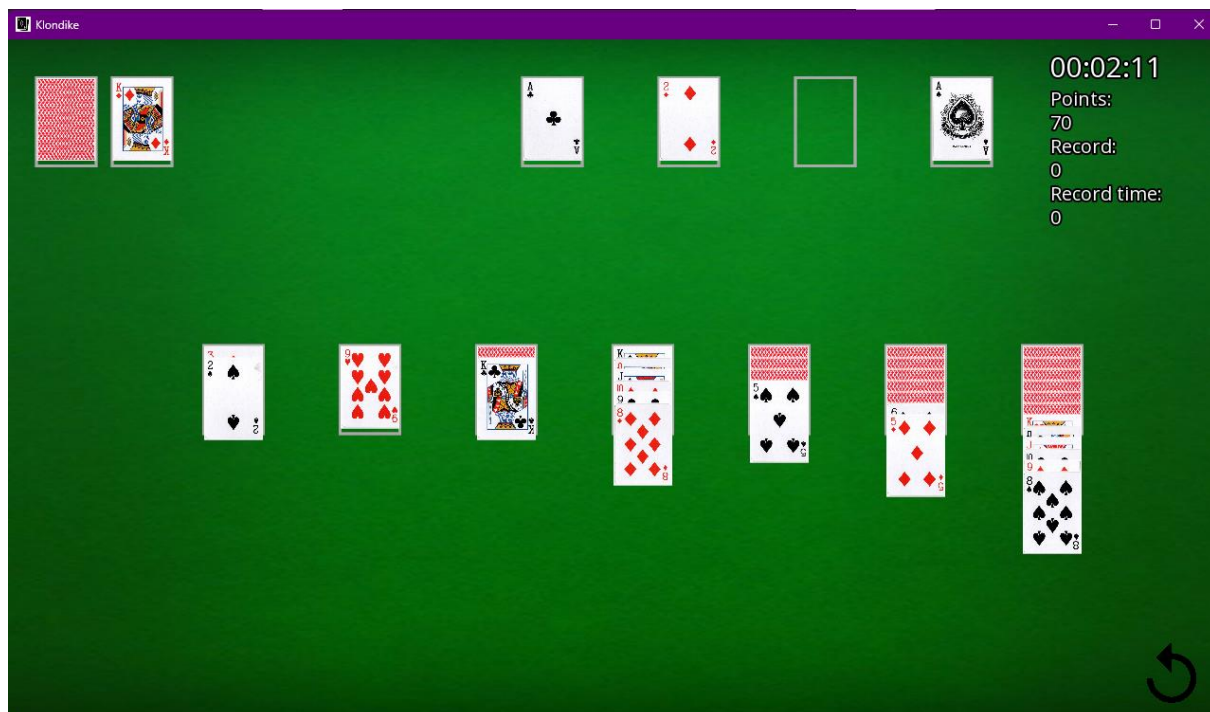


Rysunek 1: Menu programu.



Rysunek 2: Przykładowe początkowe pseudo-losowe ułożenie kart.

Wraz z rozpoczęciem rozgrywki zaczyna naliczać się czas. Oprócz bieżącego czasu wyświetlane są informacje takie jak: aktualna liczba zdobytych punktów, punktowy rekord użytkownika, oraz rekordowy czas ułożenia talii przez użytkownika. Chcąc chwycić kartę klikamy ją lewym przyciskiem myszy, oraz prawym by ją zwolnić (umieścić w innym miejscu). Niepoprawny ruch skutkuje wolnieniem karty bez zmiany jej położenia. Gracz ma też możliwość cofania ruchów za pomocą przycisku w dolnym prawym rogu ekranu. Więcej informacji o ruchach oraz punktacji zawartych jest w tutorialu uruchamianego z poziomu menu. Gra kończy się gdy gracz ułoży wszystkie karty na stosach zbierających. Po skończonej grze wyświetli się okienko z gratulacjami, którego zamknięcie spowoduje zakończenie gry.



Rysunek 3: Plansza w trakcie rozgrywki.

4 Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z obowiązującymi zasadami programowania obiektowego w języku C++. W programie rozdzielono interfejs (wygląd planszy i kart) od logiki aplikacji (ustawianie kart oraz ich ruchy).

4.1 Ogólna struktura programu

W funkcji głównej programu po wybraniu opcji Play, uaktywnia się nieskończona pętla, która będzie działać, dopóki każdy ze stosów zbierających nie będzie zawierał wektora z trzynastoma kartami. Metoda Game jest odpowiedzialna za odpowiednie uaktywnianie innych metod. Metoda setCards odpowiada za początkowe ustawienie kart na planszy. Reszta algorytmów znajduje się bezpośrednio w metodzie Game. Określają one przemieszczenia kart oraz obsługę przycisku cofającego ruchy. Oprócz tego metoda Game wywołuje metody rejestrujące i logujące użytkownika.

4.2 Szczegółowy opis typów i funkcji

Szczegółowy opis klas i metod zawarty jest w dokumentacji wygenerowanej przez doxygen.

5 Testowanie

Program został przetestowany dla różnych wygenerowanych talii kart. Zostało sprawdzona czy gra poprawnie przenosi karty między stosami niezależnie od kolejności ich generowania. Sprawdzono również system naliczania punktów podczas przenoszenia kart i analogicznie odejmowania ich podczas cofania ruchów. Po zakończonych testach stwierdzono, że program działa bezproblemowo i nie posiada błędów.

Program został sprawdzony pod kątem wycieków pamięci.

6 Wnioski

Aplikacja Pasjans jest programem złożonym wewnątrz, lecz czytelnym i zrozumiałym dla osób ze znajomością języka C++. Zarówno projekt jak i dokumentacja wygenerowana przez doxygen napisane są w języku angielskim, którego znajomość jest wymagana do prawidłowego zrozumienia struktury programu. Implementacja biblioteki graficznej SFML sprawiała małe trudności, ale ogółem rzecz biorąc jest łatwa i wygodna w obsłudze. Najwięcej problemów sprawiało prawidłowe zaimplementowanie ruchów kart zgodnie z logiką gry. Zdecydowanie zalecam wcześniejszą znajomość zarówno języka C++ jak i samej karcianki, zanim ktokolwiek powoła się na napisanie takiego projektu, niezwykle to pomaga. Sam projekt dostarczył mi wielu cennych lekcji oraz poszerzył moje umiejętności związane z programowaniem obiektowym w języku C++.

Literatura

<https://youtube.com/playlist?list=PLk6mhiZKpyW4KRTZc8sc0aYOLFmTSLA7r>

<https://www.sfml-dev.org/index.php>

Szczegółowy opis klas i funkcji - doxygen

Klondike

Generated by Doxygen 1.9.7

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 Card Class Reference	7
4.1.1 Constructor & Destructor Documentation	9
4.1.1.1 Card() [1/2]	9
4.1.1.2 Card() [2/2]	9
4.1.1.3 ~Card()	9
4.1.2 Member Function Documentation	9
4.1.2.1 draw()	9
4.1.2.2 getColor()	9
4.1.2.3 getHidden()	10
4.1.2.4 getId()	10
4.1.2.5 getPosition()	10
4.1.2.6 getShape()	10
4.1.2.7 getType()	10
4.1.2.8 getValue()	11
4.1.2.9 setHidden()	11
4.1.2.10 setOutline()	11
4.1.2.11 setPosition()	12
4.1.2.12 setTexture()	12
4.2 Heap Class Reference	12
4.2.1 Constructor & Destructor Documentation	15
4.2.1.1 Heap() [1/2]	15
4.2.1.2 Heap() [2/2]	15
4.2.1.3 ~Heap()	15
4.2.2 Member Function Documentation	16
4.2.2.1 draw()	16
4.2.2.2 getStackType()	16
4.3 Klondike Class Reference	16
4.3.1 Member Function Documentation	18
4.3.1.1 coords()	18
4.3.1.2 Game()	19
4.3.1.3 heaps()	20
4.3.1.4 Login()	20
4.3.1.5 Menu()	21

4.3.1.6 piles()	23
4.3.1.7 Register()	23
4.3.1.8 setCards()	24
4.3.1.9 setRecord()	25
4.3.1.10 stacks()	26
4.3.1.11 Tutorial()	27
4.3.2 Member Data Documentation	27
4.3.2.1 choice	27
4.3.2.2 login	28
4.3.2.3 password	28
4.3.2.4 player	28
4.3.2.5 record	28
4.3.2.6 screenSize	28
4.3.2.7 timeRecord	28
4.4 myEvent Class Reference	29
4.4.1 Constructor & Destructor Documentation	29
4.4.1.1 myEvent() [1/4]	29
4.4.1.2 myEvent() [2/4]	30
4.4.1.3 myEvent() [3/4]	30
4.4.1.4 myEvent() [4/4]	30
4.4.2 Member Function Documentation	31
4.4.2.1 getChangedTexture()	31
4.4.2.2 getDestinationCard()	31
4.4.2.3 getDestinationStackPtr()	31
4.4.2.4 getGrabbedStackPtr()	31
4.4.2.5 getMovedCardsAmount()	31
4.4.2.6 getType()	31
4.4.2.7 setChangedTexture()	31
4.5 Pile Class Reference	32
4.5.1 Constructor & Destructor Documentation	35
4.5.1.1 Pile() [1/2]	35
4.5.1.2 Pile() [2/2]	35
4.5.2 Member Function Documentation	36
4.5.2.1 draw()	36
4.5.2.2 getStackType()	36
4.5.2.3 getType()	36
4.5.2.4 setType()	36
4.6 Player Class Reference	36
4.6.1 Constructor & Destructor Documentation	37
4.6.1.1 Player()	37
4.6.2 Member Function Documentation	38
4.6.2.1 addPoints()	38

4.6.2.2 checkTimeRecord()	38
4.6.2.3 getCurrentRecord()	39
4.6.2.4 getLogin()	39
4.6.2.5 getPassword()	40
4.6.2.6 getPoints()	40
4.6.2.7 getTimeRecord()	40
4.6.2.8 operator+()	40
4.6.2.9 operator-()	41
4.6.2.10 setCurrentRecord()	41
4.6.2.11 setLogin()	41
4.6.2.12 setPassword()	41
4.6.2.13 setPoints()	42
4.6.2.14 setTimeRecord()	42
4.7 Shuffle Class Reference	42
4.7.1 Member Function Documentation	43
4.7.1.1 GenerateDeck()	43
4.7.1.2 ShuffleDeck()	43
4.8 Stack Class Reference	44
4.8.1 Constructor & Destructor Documentation	47
4.8.1.1 Stack() [1/2]	47
4.8.1.2 Stack() [2/2]	47
4.8.1.3 ~Stack()	47
4.8.2 Member Function Documentation	48
4.8.2.1 addCard()	48
4.8.2.2 draw()	48
4.8.2.3 getPos()	48
4.8.2.4 getShape()	49
4.8.2.5 getStackType()	49
4.8.2.6 isLastInStack()	49
4.8.2.7 removeCards()	50
4.8.2.8 removeFirstCard()	50
4.8.2.9 returnCardInStackIndex()	50
4.8.2.10 returnLastCard()	51
4.8.2.11 setPos()	51
4.8.3 Member Data Documentation	51
4.8.3.1 cards	51
4.8.3.2 height	51
4.8.3.3 id	52
4.8.3.4 pos_X	52
4.8.3.5 pos_Y	52
4.8.3.6 Shape	52
4.8.3.7 width	52

4.9 TextureManager Class Reference	52
4.9.1 Constructor & Destructor Documentation	53
4.9.1.1 ~TextureManager()	53
4.9.2 Member Function Documentation	53
4.9.2.1 getLength()	53
4.9.2.2 getTexture() [1/2]	53
4.9.2.3 getTexture() [2/2]	54
4.9.2.4 loadTexture()	54
5 File Documentation	55
5.1 Card.cpp File Reference	55
5.2 Card.h File Reference	55
5.2.1 Detailed Description	56
5.3 Card.h	57
5.4 ConsoleApplication1.cpp File Reference	58
5.4.1 Function Documentation	58
5.4.1.1 main()	58
5.5 Heap.cpp File Reference	58
5.6 Heap.h File Reference	59
5.6.1 Detailed Description	60
5.7 Heap.h	60
5.8 Klondike.cpp File Reference	61
5.9 Klondike.h File Reference	61
5.9.1 Detailed Description	62
5.10 Klondike.h	63
5.11 Lib.h File Reference	63
5.12 Lib.h	63
5.13 main.cpp File Reference	64
5.13.1 Function Documentation	64
5.13.1.1 main()	64
5.14 myEvent.cpp File Reference	65
5.15 myEvent.h File Reference	66
5.15.1 Detailed Description	67
5.16 myEvent.h	68
5.17 Pile.cpp File Reference	68
5.18 Pile.h File Reference	69
5.18.1 Detailed Description	70
5.19 Pile.h	70
5.20 Player.cpp File Reference	71
5.21 Player.h File Reference	71
5.21.1 Detailed Description	72
5.22 Player.h	72

5.23 Shuffle.cpp File Reference	73
5.24 Shuffle.h File Reference	73
5.24.1 Detailed Description	74
5.25 Shuffle.h	75
5.26 Stack.cpp File Reference	75
5.27 Stack.h File Reference	75
5.27.1 Detailed Description	76
5.28 Stack.h	77
5.29 TextureManager.cpp File Reference	78
5.30 TextureManager.h File Reference	78
5.30.1 Detailed Description	79
5.31 TextureManager.h	79
Index	81

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

sf::Drawable	
Card	7
Stack	44
Heap	12
Pile	32
Klondike	16
myEvent	29
Player	36
Shuffle	42
TextureManager	52

Chapter 2

Class Index

2.1 Class List

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

Card	7
Heap	12
Klondike	16
myEvent	29
Pile	32
Player	36
Shuffle	42
Stack	44
TextureManager	52

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Card.cpp	55
Card.h	
Card class	55
ConsoleApplication1.cpp	58
Heap.cpp	58
Heap.h	
Heap of cards in top left corner (class)	59
Klondike.cpp	61
Klondike.h	
Main Game Class	61
Lib.h	63
main.cpp	64
myEvent.cpp	65
myEvent.h	
Class made to enable undoing movements. Named myEvent because Event already exists as a part of SFML library	66
Pile.cpp	68
Pile.h	
Four (collecting) piles of cards at the top (class)	69
Player.cpp	71
Player.h	
Current player data class	71
Shuffle.cpp	73
Shuffle.h	
Class being used to generate and shuffle card deck	73
Stack.cpp	75
Stack.h	
Stack class, heap and pile classes inherit from this class	75
TextureManager.cpp	78
TextureManager.h	
Class that manages all loaded textures from files	78

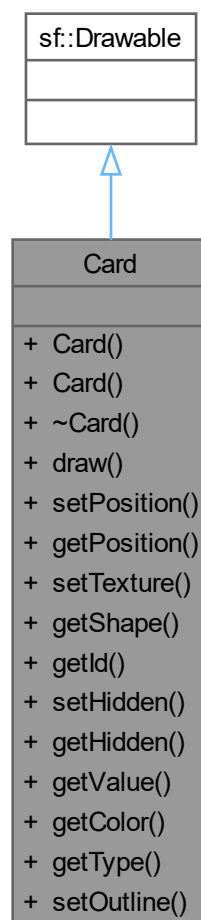
Chapter 4

Class Documentation

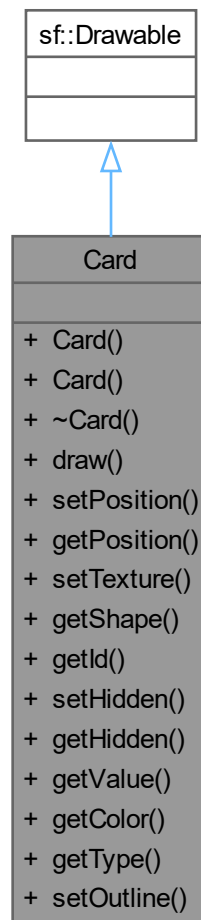
4.1 Card Class Reference

```
#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()

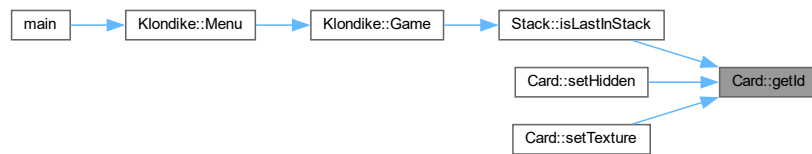
```
bool Card::getHidden ( )
```

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()

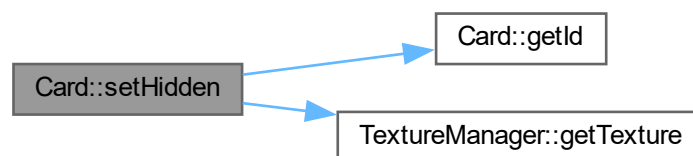
```
void Card::setHidden (
    const bool i )
```

Setting card visibility method.

Parameters

<i>i</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 )
```

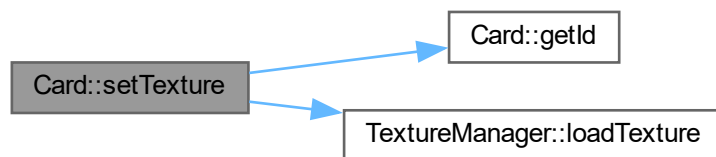
[Card](#) positioning method Here is the caller graph for this function:



4.1.2.12 setTexture()

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

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



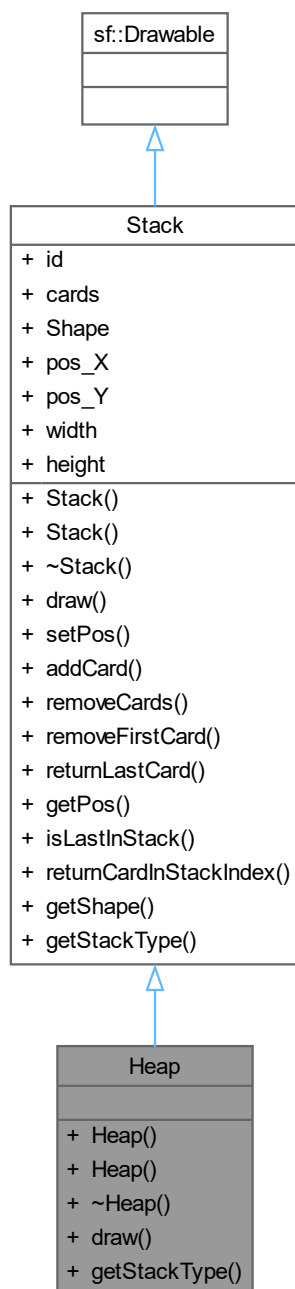
The documentation for this class was generated from the following files:

- [Card.h](#)
- [Card.cpp](#)

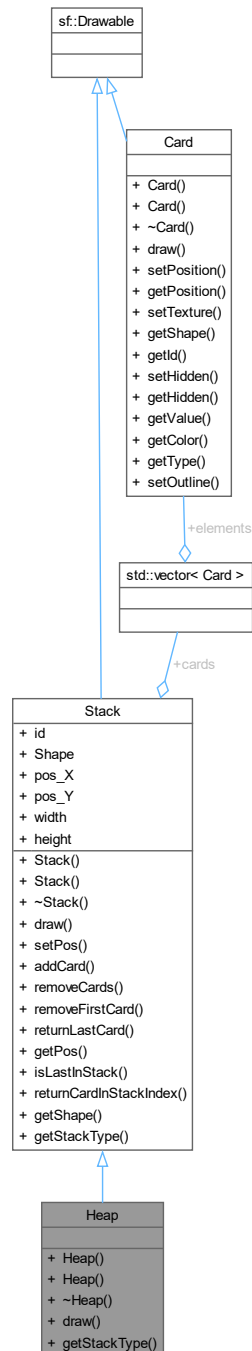
4.2 Heap Class Reference

```
#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

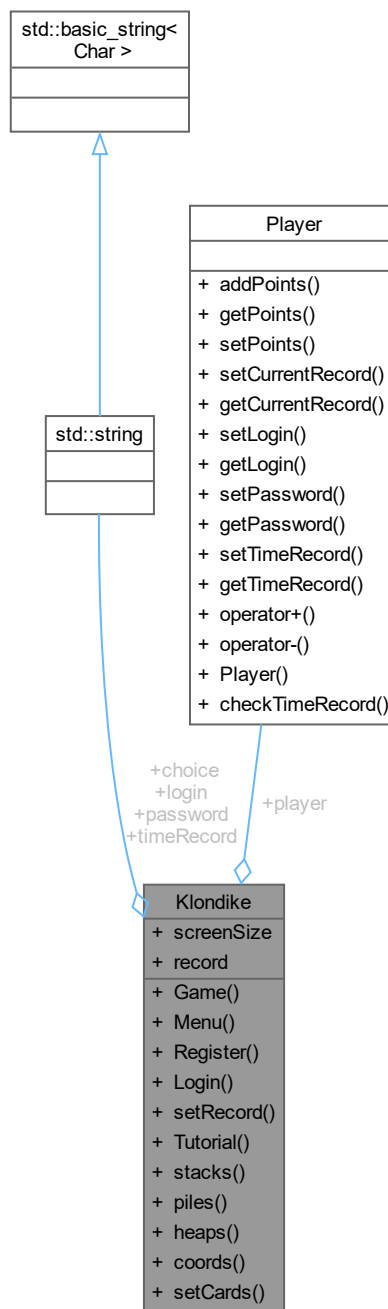
The documentation for this class was generated from the following files:

- [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

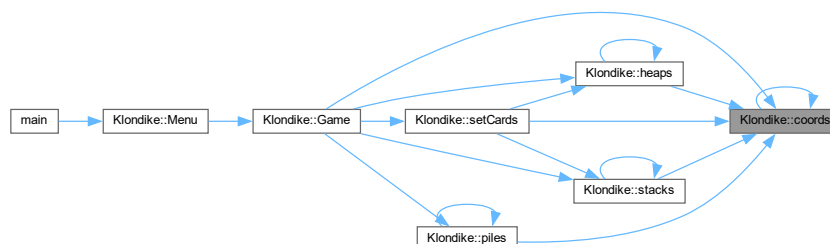
4.3.1.1 coords()

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

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



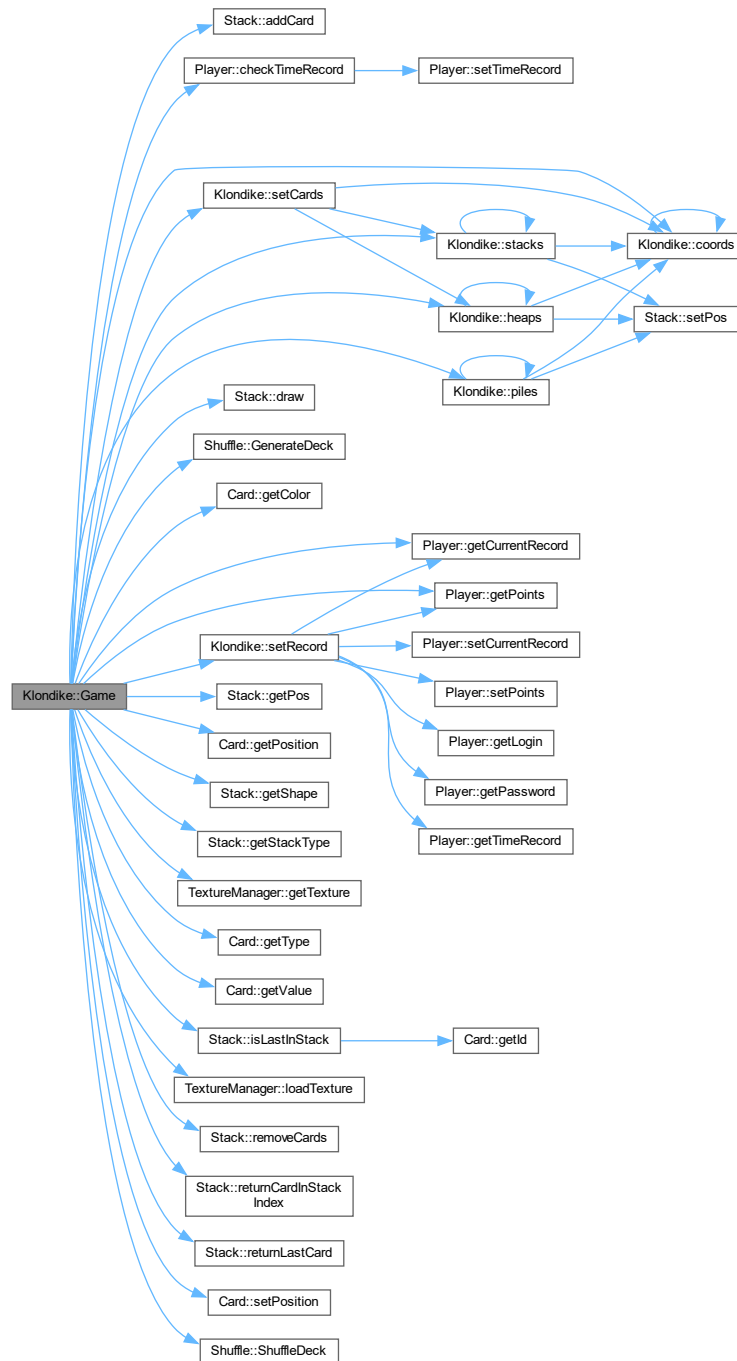
Here is the caller graph for this function:



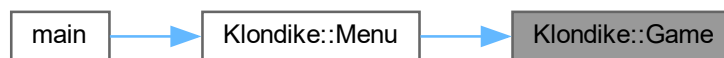
4.3.1.2 Game()

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

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



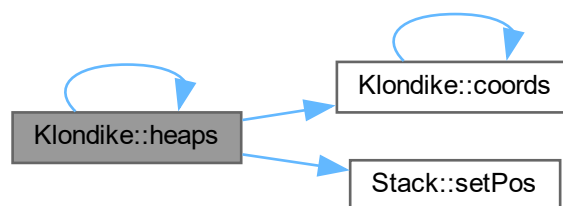
Here is the caller graph for this function:



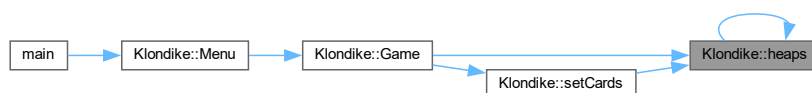
4.3.1.3 heaps()

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

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



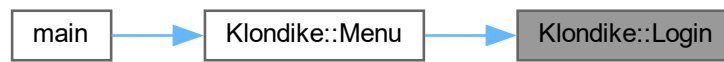
Here is the caller graph for this function:



4.3.1.4 Login()

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

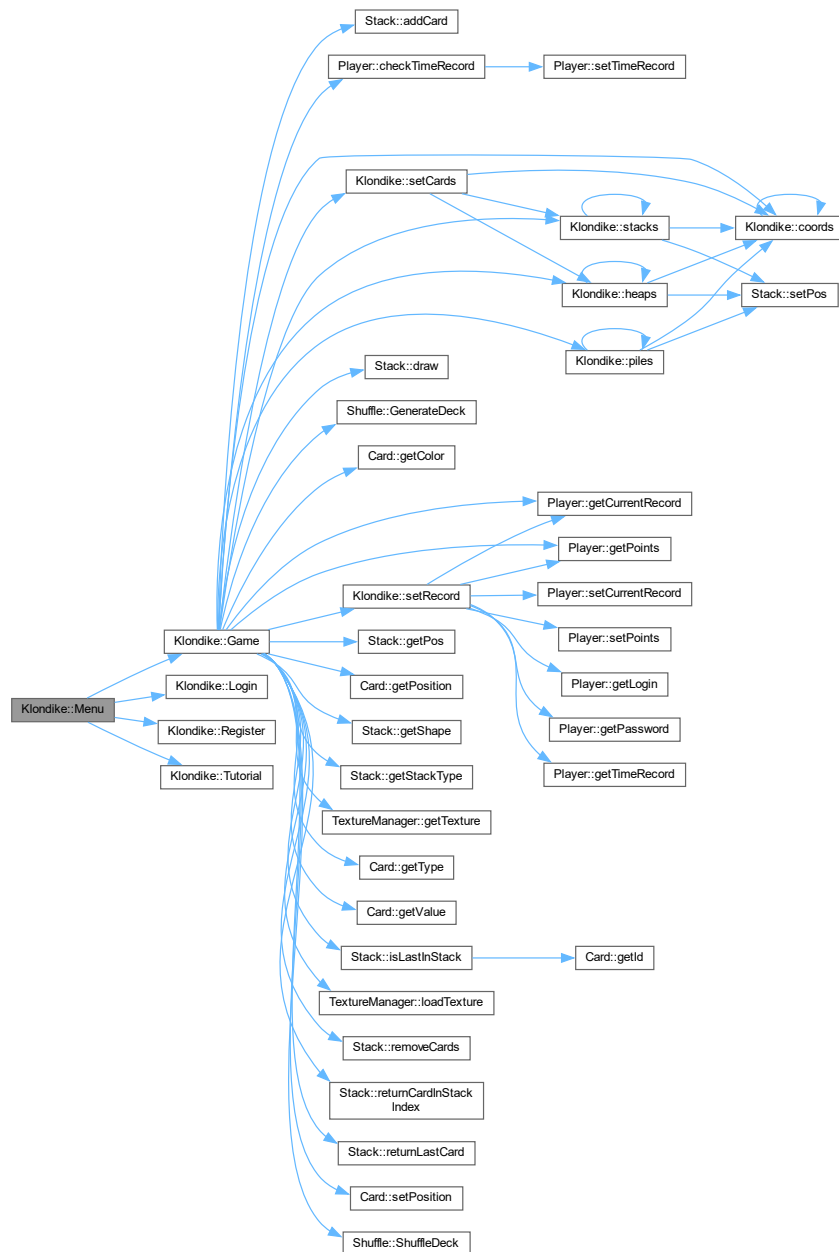
[Player](#) logging method Here is the caller graph for this function:



4.3.1.5 Menu()

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

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



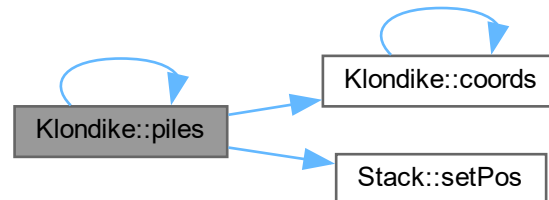
Here is the caller graph for this function:



4.3.1.6 piles()

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

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



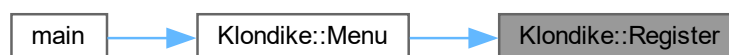
Here is the caller graph for this function:



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()

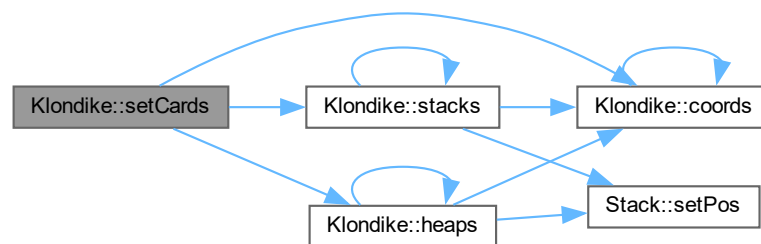
```
void Klondike::setCards (
    std::vector< Stack > & stacks,
    std::vector< Pile > & piles,
    std::vector< Heap > & heaps,
    std::vector< Card > & cards,
    std::vector< float > & coords ) [static]
```

Starting cards positioning method.

Parameters

<i>stacks</i>	Stacks vector
<i>piles</i>	Piles vector
<i>heaps</i>	Heaps vector
<i>cards</i>	Cards vector
<i>coords</i>	Coords vector

Here is the call graph for this function:



Here is the caller graph for this function:



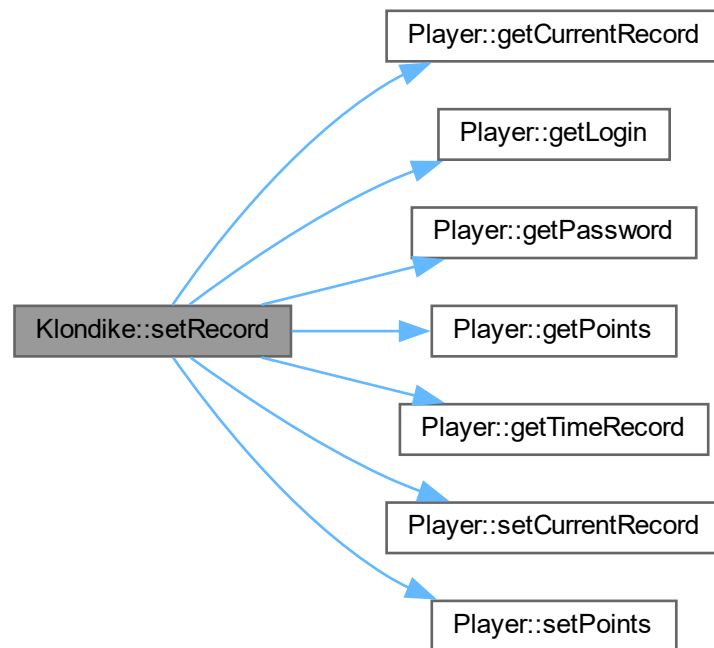
4.3.1.9 setRecord()

```

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

```


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



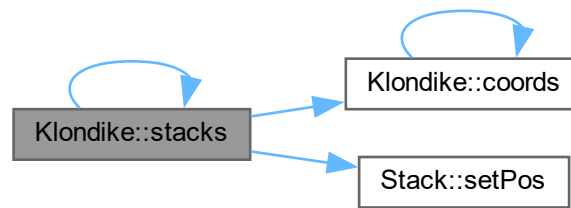
Here is the caller graph for this function:



4.3.1.10 stacks()

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

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



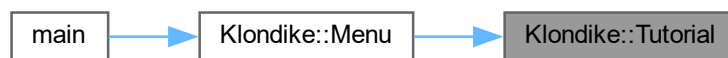
Here is the caller graph for this function:



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

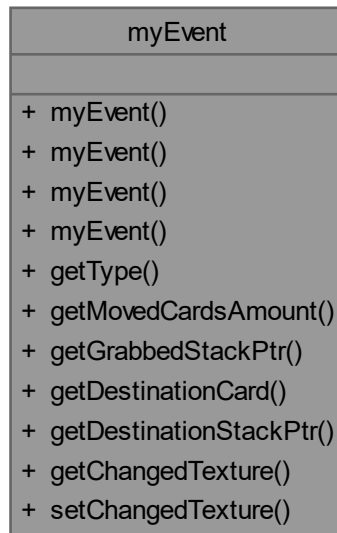
The documentation for this class was generated from the following files:

- [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

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

4.4.1.3 myEvent() [3/4]

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

Event constructor.

Parameters

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

4.4.1.4 myEvent() [4/4]

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

Event constructor.

Parameters

<i>type</i>	Amount of cards that are being moved from stack to stack
<i>movedCardsAmount</i>	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
---	-------------------------

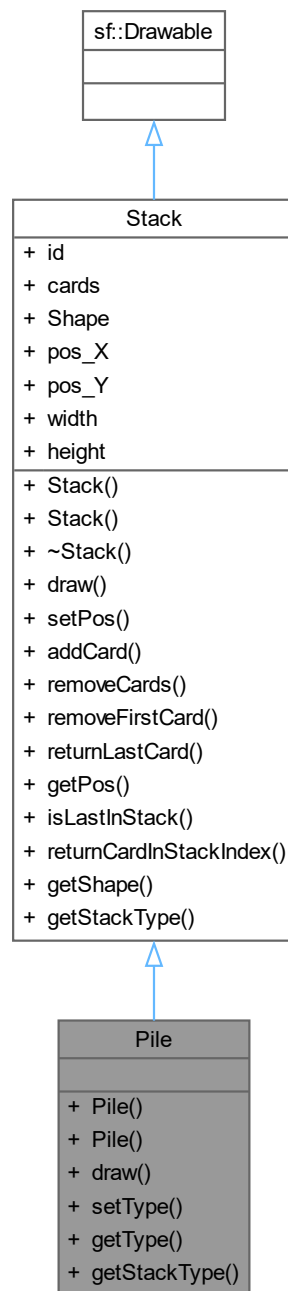
The documentation for this class was generated from the following files:

- [myEvent.h](#)
- [myEvent.cpp](#)

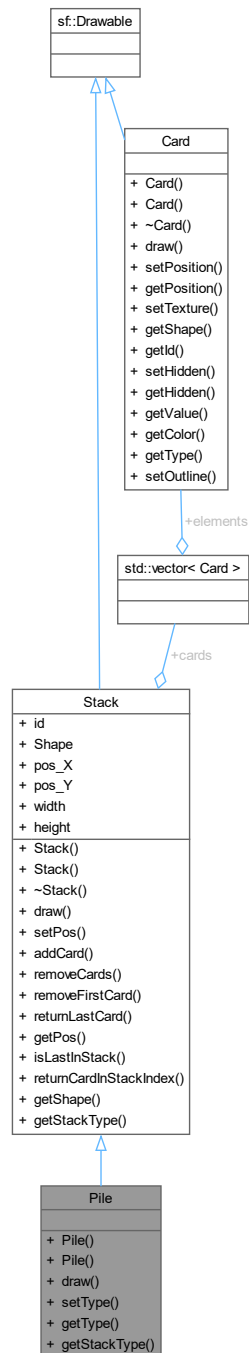
4.5 Pile Class Reference

```
#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

<i>RectangleX</i>	Pile position (x)
<i>RectangleY</i>	Pile position (y)
<i>stackType</i>	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
---	-------------------------

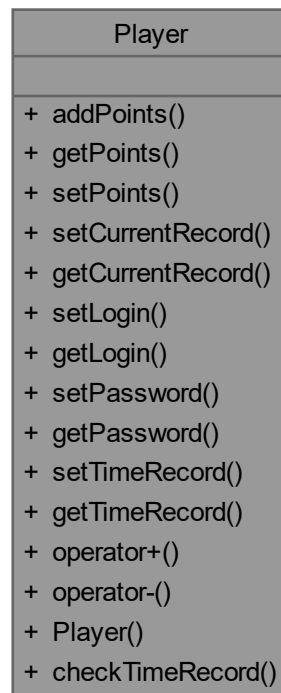
The documentation for this class was generated from the following files:

- [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

<i>login</i>	Login
<i>password</i>	Password
<i>record</i>	Record
<i>timeRecord</i>	Time record

4.6.2 Member Function Documentation

4.6.2.1 addPoints()

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

Adding points method.

Parameters

<i>points</i>	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

<i>time</i>	Previous saved time record text
<i>hours</i>	Current time hours
<i>minutes</i>	Current time minutes
<i>seconds</i>	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 ( )
```

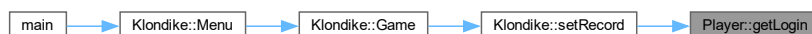
Getting current record value Here is the caller graph for this function:



4.6.2.4 `getLogin()`

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

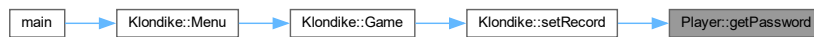
Getting login method Here is the caller graph for this function:



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

<i>points</i>	value
---------------	-------

4.6.2.9 operator-()

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

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

Parameters

<i>points</i>	value
---------------	-------

4.6.2.10 setCurrentRecord()

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

Setting current record method.

Parameters

<i>record</i>	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

<i>login</i>	Login
--------------	-------

4.6.2.12 setPassword()

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

Setting password method.

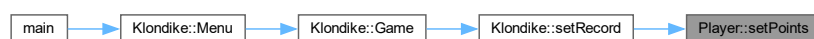
Parameters

<i>password</i>	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

<i>timeRecord</i>	Time record (xx::yy::zz)
-------------------	--------------------------

Here is the caller graph for this function:



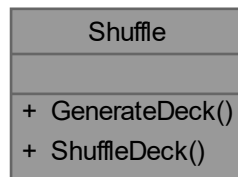
The documentation for this class was generated from the following files:

- [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

<i>Deck</i>	Deck we want to shuffle
-------------	-------------------------

Here is the caller graph for this function:



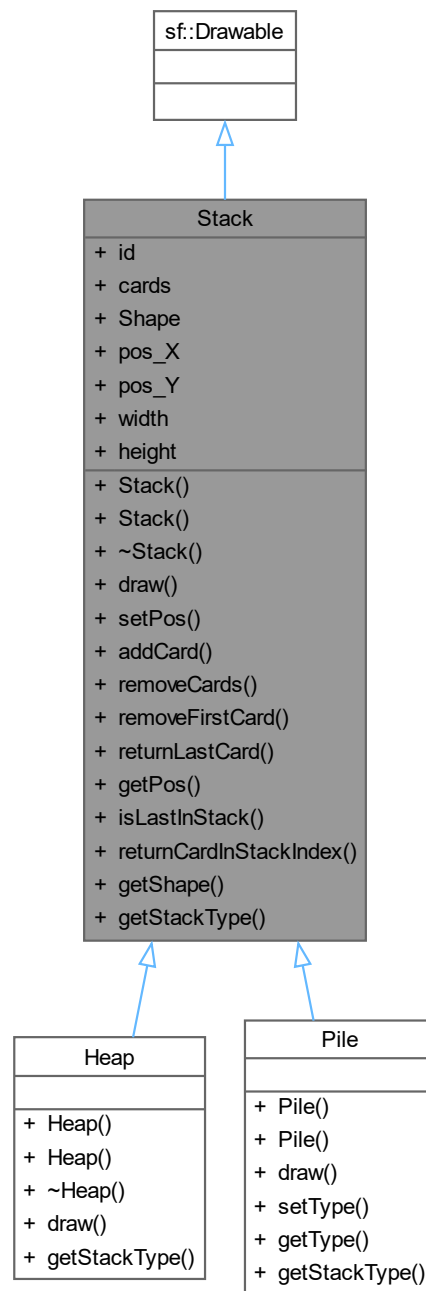
The documentation for this class was generated from the following files:

- [Shuffle.h](#)
- [Shuffle.cpp](#)

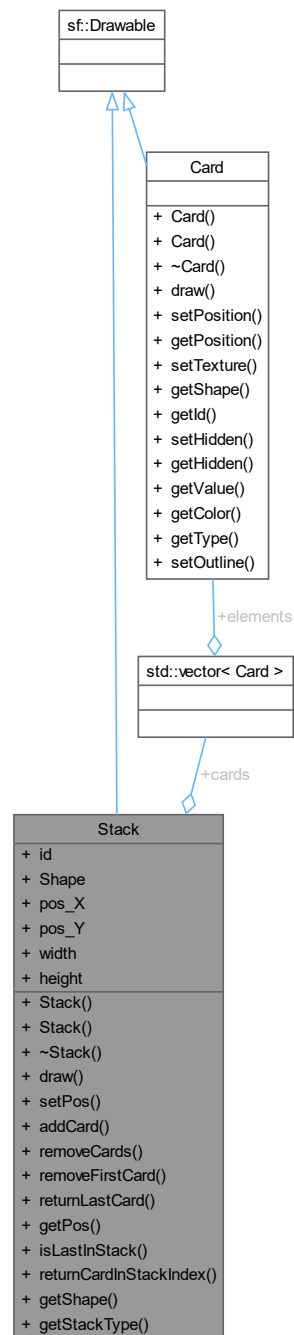
4.8 Stack Class Reference

```
#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

<i>number</i>	Stack ID
<i>RectangleX</i>	Stack position (x)
<i>RectangleY</i>	Stack position (y)
<i>stackType</i>	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

<i>card</i>	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

<i>card</i>	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

<i>number</i>	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 )
```

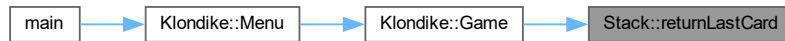
Method that returns index of last card in stack's vector Here is the caller graph for this function:



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()

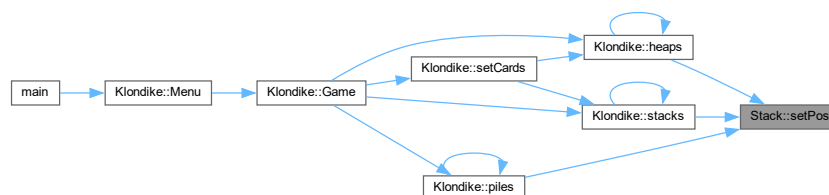
```
void Stack::setPos (
    float x,
    float y ) [virtual]
```

Virtual setting position method.

Parameters

<i>x</i>	X
<i>y</i>	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)

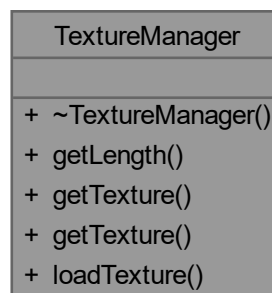
The documentation for this class was generated from the following files:

- [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]
```

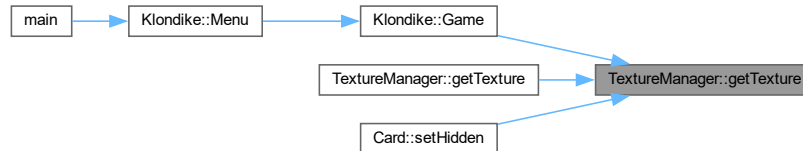
Get texture by index in map, or return null Here is the call graph for this function:



4.9.2.3 `getTexture()` [2/2]

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

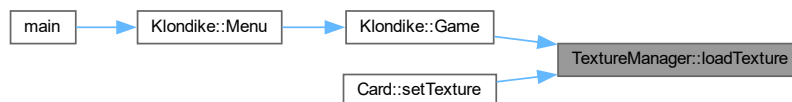
Get texture by name specified in loadTexture, or return null Here is the caller graph for this function:



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:

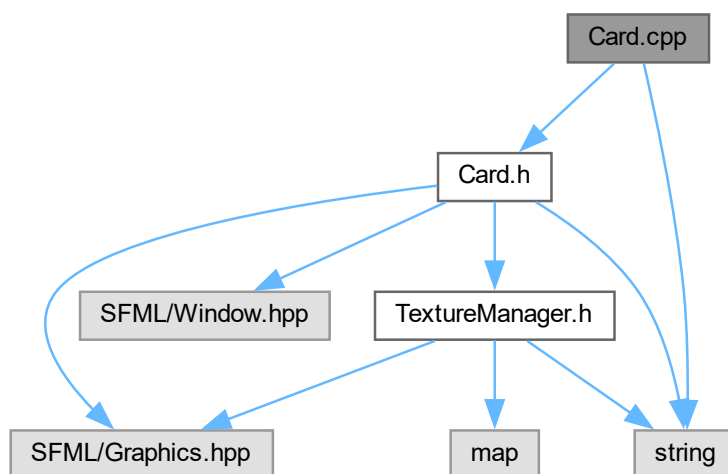
- [TextureManager.h](#)
- [TextureManager.cpp](#)

Chapter 5

File Documentation

5.1 Card.cpp File Reference

```
#include "Card.h"  
#include <string>  
Include dependency graph for Card.cpp:
```



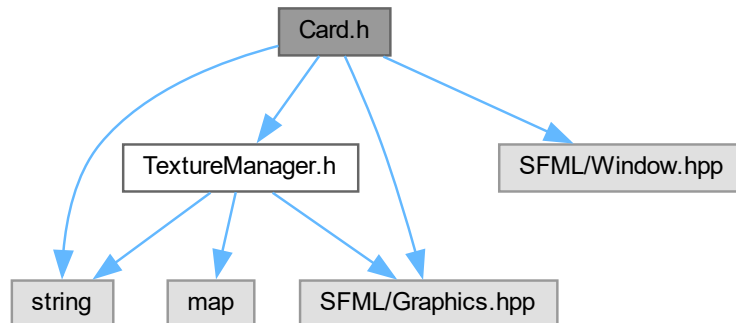
5.2 Card.h File Reference

[Card](#) class.

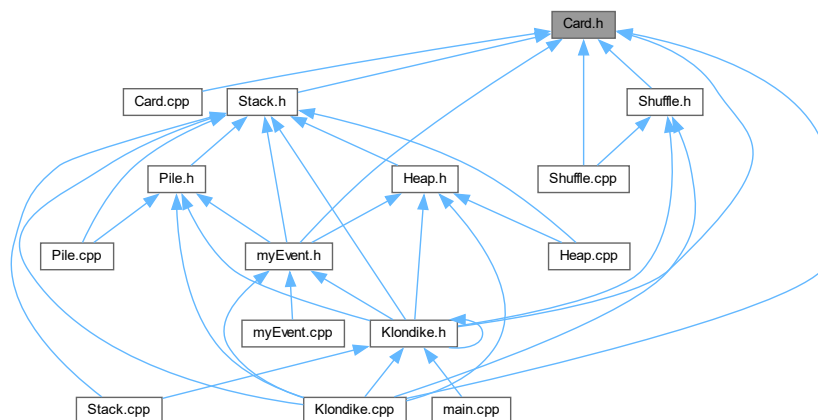
```
#include <string>  
#include <SFML/Graphics.hpp>  
#include <SFML/Window.hpp>
```

```
#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

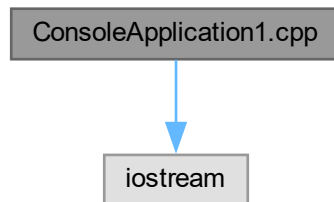
[Go to the documentation of this file.](#)

```
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()`

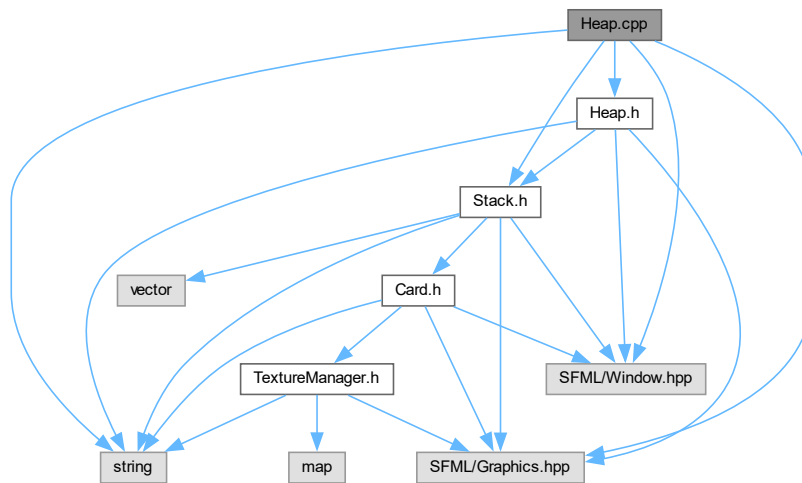
```
int main ( )
```

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)

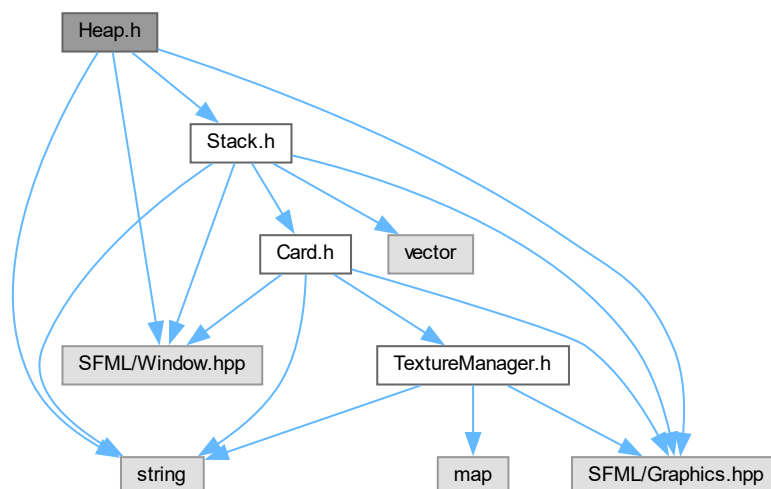
```
#include <string>
```

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

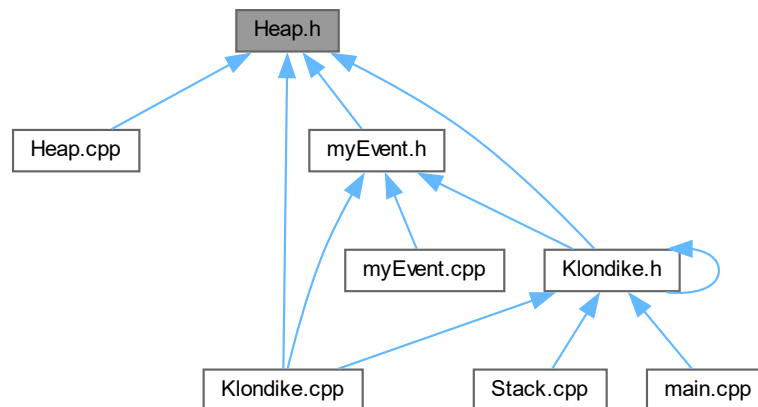
```
#include <SFML/Window.hpp>
```

```
#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.](#)

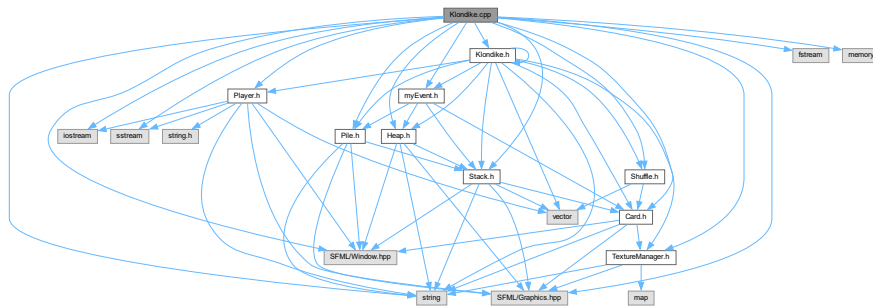
```

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:

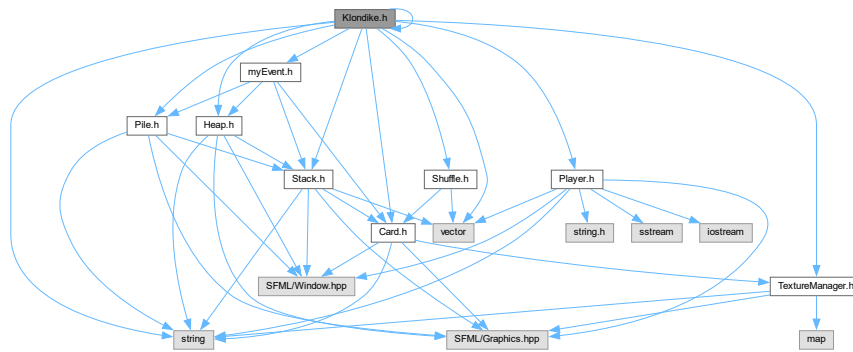


5.9 Klondike.h File Reference

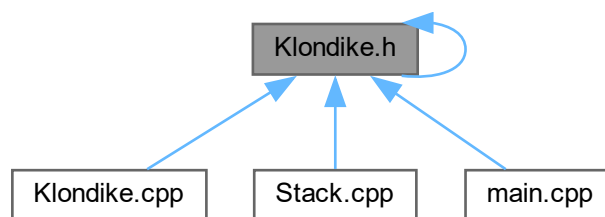
Main Game Class.

```
#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

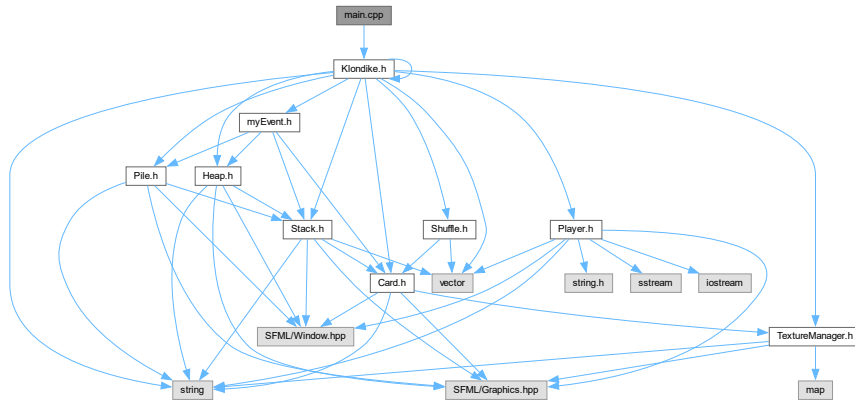
5.12 Lib.h

[Go to the documentation of this file.](#)

5.13 main.cpp File Reference

```
#include "Klondike.h"
```

Include dependency graph for main.cpp:



Functions

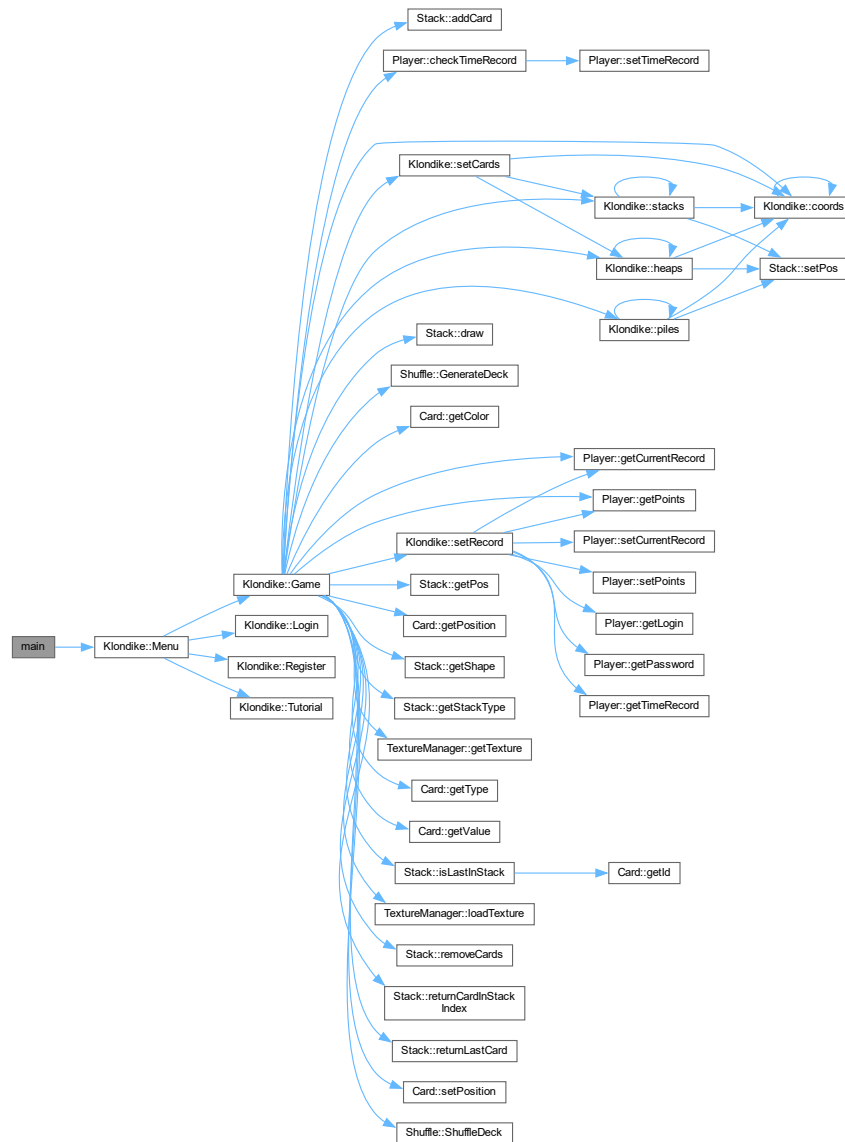
- int [main](#) ()

5.13.1 Function Documentation

5.13.1.1 main()

```
int main ( )
```

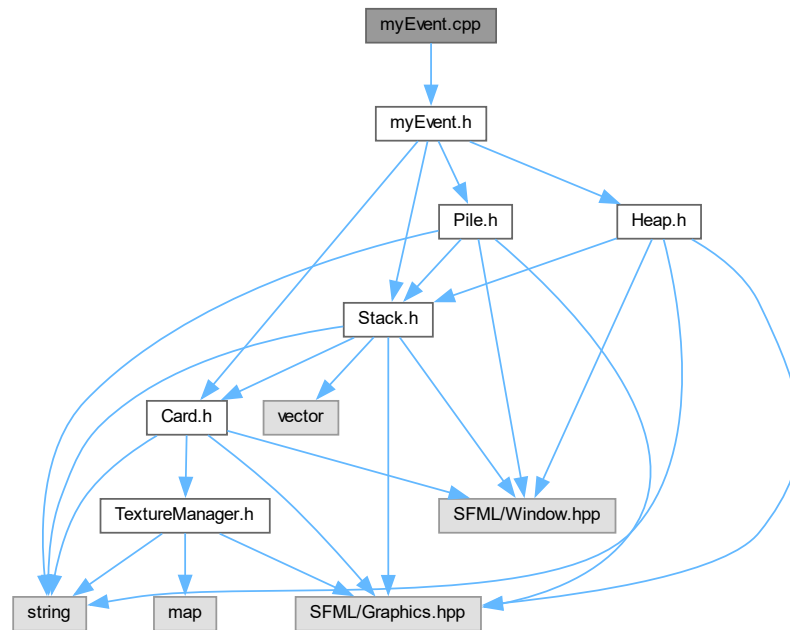
Here is the call graph for this function:



5.14 myEvent.cpp File Reference

```
#include "myEvent.h"
```


Include dependency graph for myEvent.cpp:

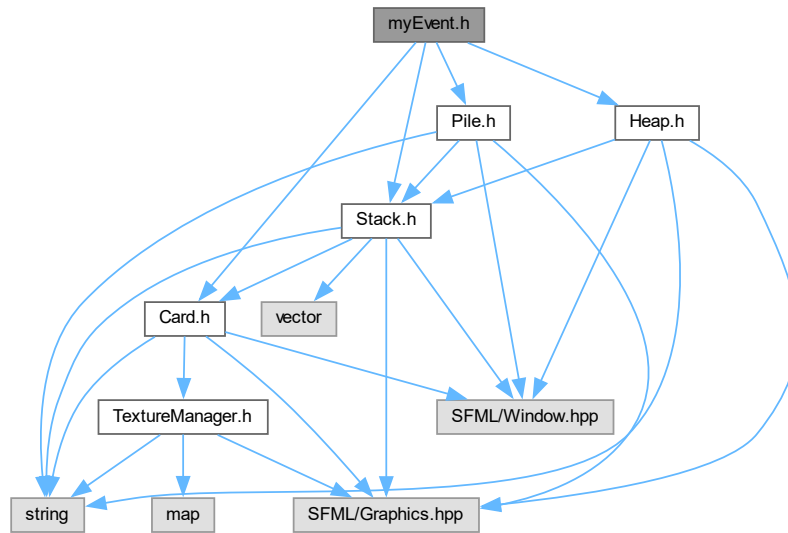


5.15 myEvent.h File Reference

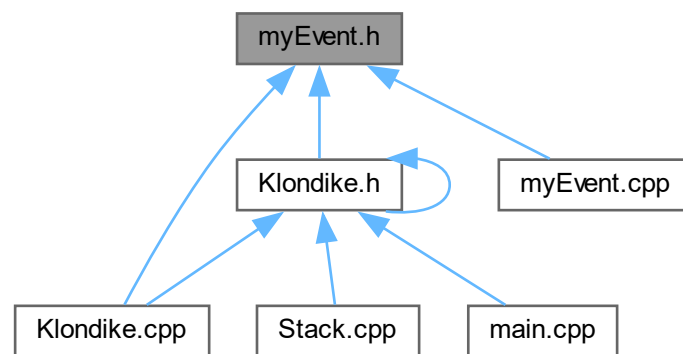
Class made to enable undoing movements. Named `myEvent` because Event already exists as a part of SFML library.

```
#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

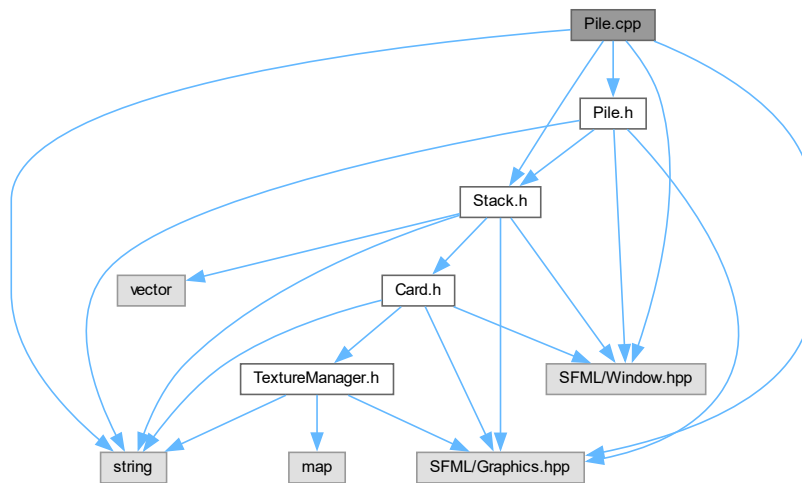
```

#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)

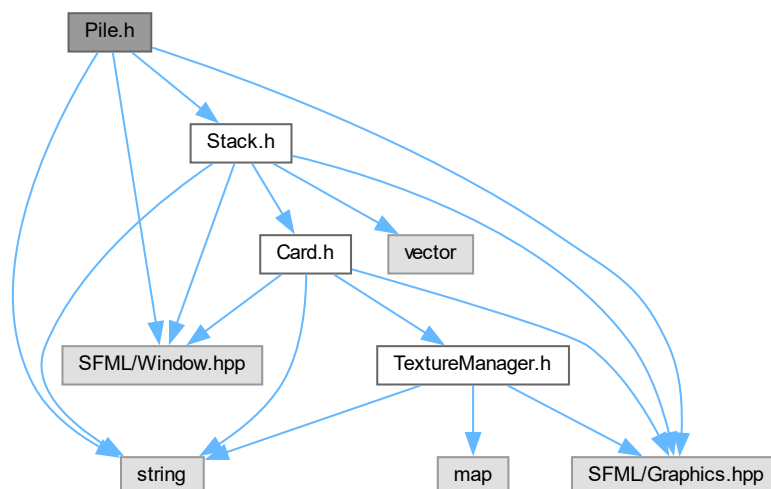
```
#include "Stack.h"
```

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

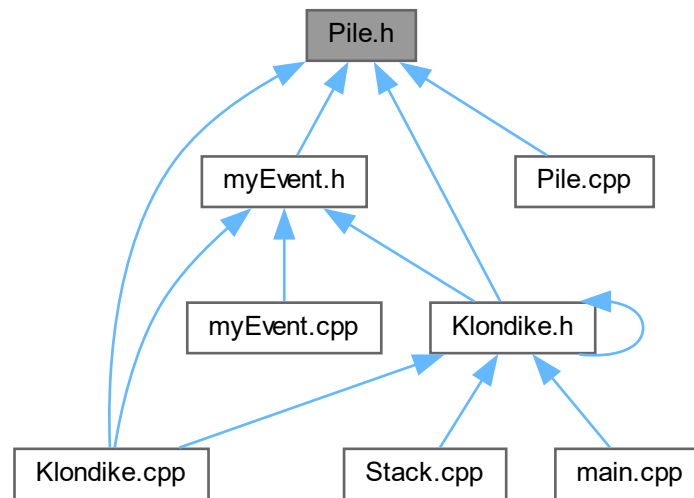
```
#include <SFML/Window.hpp>
```

```
#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     File();
00032     File(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 };

```

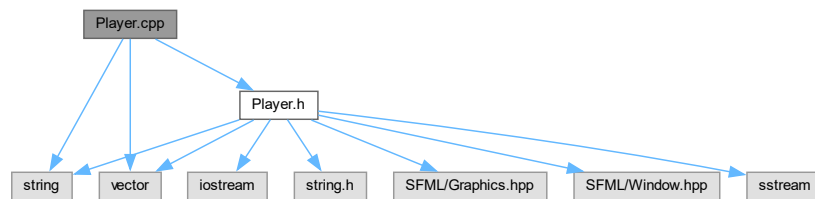
5.20 Player.cpp File Reference

```

#include "Player.h"
#include <string>
#include <vector>

```

Include dependency graph for Player.cpp:



5.21 Player.h File Reference

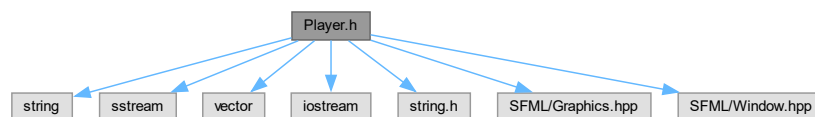
Current player data class.

```

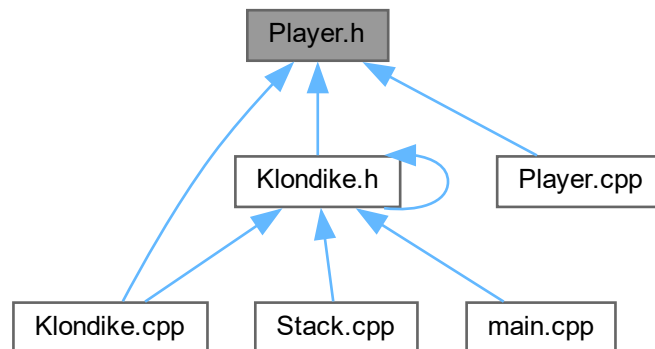
#include <string>
#include <sstream>
#include <vector>
#include <iostream>
#include <string.h>
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>

```

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 };

```

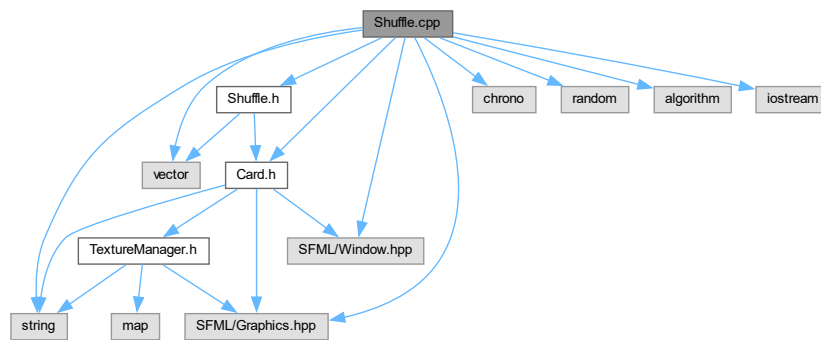
5.23 Shuffle.cpp File Reference

```

#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

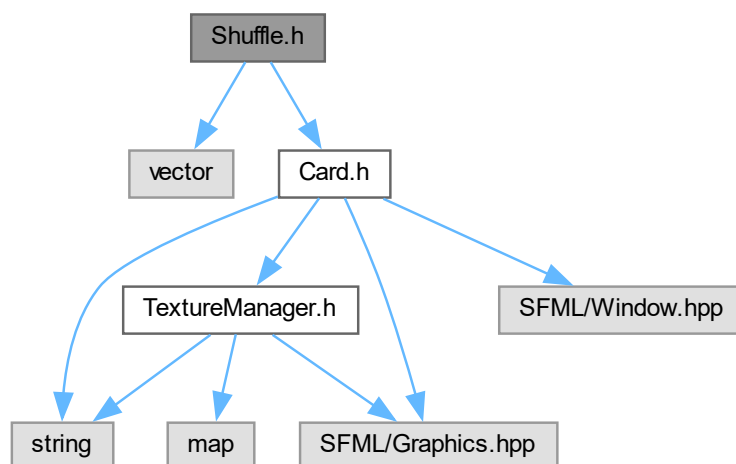
Class being used to generate and shuffle card deck.

```

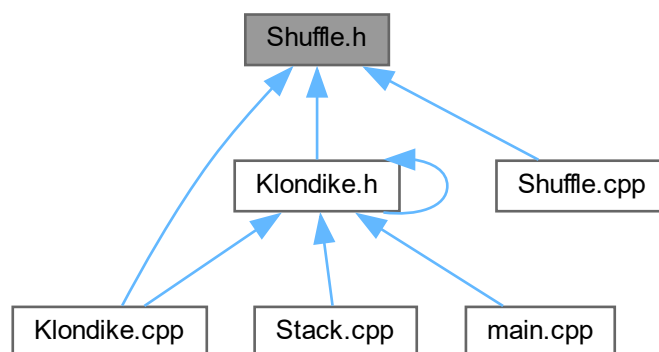
#include <vector>
#include "Card.h"

```


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  };

```

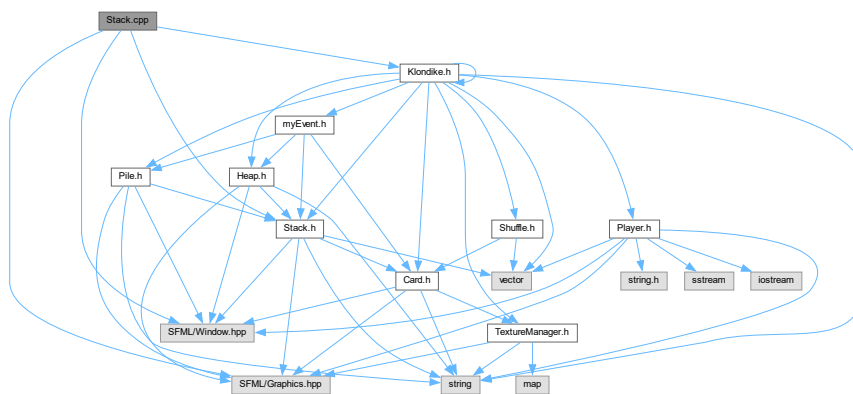
5.26 Stack.cpp File Reference

```

#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

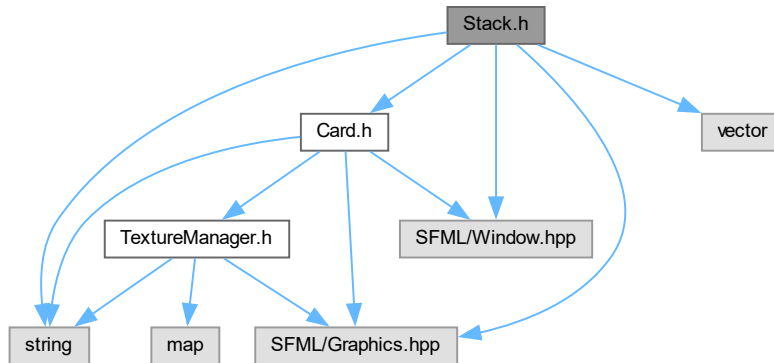
[Stack](#) class, heap and pile classes inherit from this class.

```

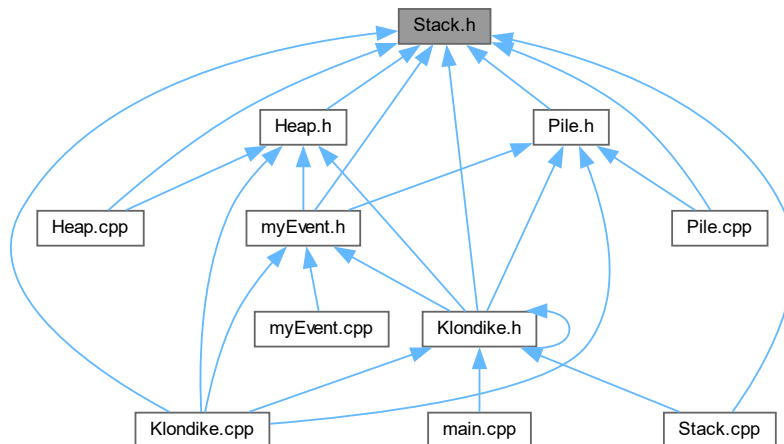
#include "Card.h"
#include <string>
#include <vector>

```

```
#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

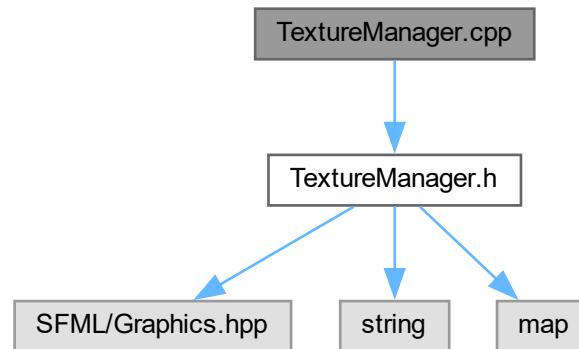
[Go to the documentation of this file.](#)

```
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

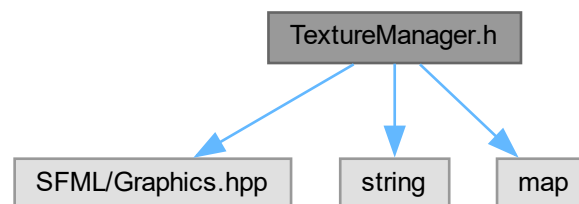
Class that manages all loaded textures from files.

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

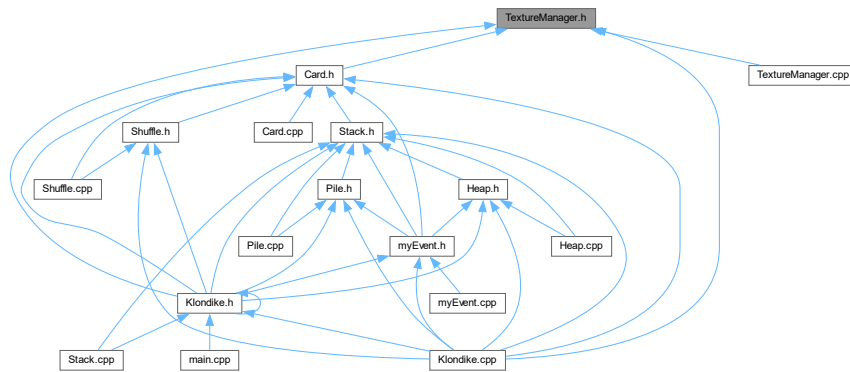
```
#include <string>
```

```
#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

- ~Card
 - Card, [9](#)
- ~Heap
 - Heap, [15](#)
- ~Stack
 - Stack, [47](#)
- ~TextureManager
 - TextureManager, [53](#)
- addCard
 - Stack, [48](#)
- addPoints
 - Player, [38](#)
- Card, [7](#)
 - ~Card, [9](#)
 - Card, [9](#)
 - draw, [9](#)
 - getColor, [9](#)
 - getHidden, [9](#)
 - getId, [10](#)
 - getPosition, [10](#)
 - getShape, [10](#)
 - getType, [10](#)
 - getValue, [10](#)
 - setHidden, [11](#)
 - setOutline, [11](#)
 - setPosition, [11](#)
 - setTexture, [12](#)
- Card.cpp, [55](#)
- Card.h, [55](#)
- cards
 - Stack, [51](#)
- checkTimeRecord
 - Player, [38](#)
- choice
 - Klondike, [27](#)
- ConsoleApplication1.cpp, [58](#)
 - main, [58](#)
- coords
 - Klondike, [18](#)
- draw
 - Card, [9](#)
 - Heap, [16](#)
 - Pile, [36](#)
 - Stack, [48](#)
- Game
 - Klondike, [18](#)
- GenerateDeck
 - Shuffle, [43](#)
- getChangedTexture
 - myEvent, [31](#)
- getColor
 - Card, [9](#)
- getCurrentRecord
 - Player, [39](#)
- getDestinationCard
 - myEvent, [31](#)
- getDestinationStackPtr
 - myEvent, [31](#)
- getGrabbedStackPtr
 - myEvent, [31](#)
- getHidden
 - Card, [9](#)
- getId
 - Card, [10](#)
- getLength
 - TextureManager, [53](#)
- getLogin
 - Player, [39](#)
- getMovedCardsAmount
 - myEvent, [31](#)
- getPassword
 - Player, [39](#)
- getPoints
 - Player, [40](#)
- getPos
 - Stack, [48](#)
- getPosition
 - Card, [10](#)
- getShape
 - Card, [10](#)
 - Stack, [48](#)
- getStackType
 - Heap, [16](#)
 - Pile, [36](#)
 - Stack, [49](#)
- getTexture
 - TextureManager, [53](#)
- getTimeRecord
 - Player, [40](#)
- getType
 - Card, [10](#)
 - myEvent, [31](#)
 - Pile, [36](#)
- getValue
 - Card, [10](#)

- Heap, 12
 - ~Heap, 15
 - draw, 16
 - getStackType, 16
 - Heap, 15
- Heap.cpp, 58
- Heap.h, 59
- heaps
 - Klondike, 20
- height
 - Stack, 51
- id
 - Stack, 51
- isLastInStack
 - Stack, 49
- Klondike, 16
 - choice, 27
 - coords, 18
 - Game, 18
 - heaps, 20
 - Login, 20
 - login, 27
 - Menu, 21
 - password, 28
 - piles, 22
 - player, 28
 - record, 28
 - Register, 23
 - screenSize, 28
 - setCards, 23
 - setRecord, 25
 - stacks, 26
 - timeRecord, 28
 - Tutorial, 27
- Klondike.cpp, 61
- Klondike.h, 61
- Lib.h, 63
- loadTexture
 - TextureManager, 54
- Login
 - Klondike, 20
- login
 - Klondike, 27
- main
 - ConsoleApplication1.cpp, 58
 - main.cpp, 64
- main.cpp, 64
 - main, 64
- Menu
 - Klondike, 21
- myEvent, 29
 - getChangedTexture, 31
 - getDestinationCard, 31
 - getDestinationStackPtr, 31
 - getGrabbedStackPtr, 31
 - getMovedCardsAmount, 31
 - getType, 31
 - myEvent, 29, 30
 - setChangedTexture, 31
- myEvent.cpp, 65
- myEvent.h, 66
- operator+
 - Player, 40
- operator-
 - Player, 41
- password
 - Klondike, 28
- Pile, 32
 - draw, 36
 - getStackType, 36
 - getType, 36
 - Pile, 35
 - setType, 36
- Pile.cpp, 68
- Pile.h, 69
- piles
 - Klondike, 22
- Player, 36
 - addPoints, 38
 - checkTimeRecord, 38
 - getCurrentRecord, 39
 - getLogin, 39
 - getPassword, 39
 - getPoints, 40
 - getTimeRecord, 40
 - operator+, 40
 - operator-, 41
 - Player, 37
 - setCurrentRecord, 41
 - setLogin, 41
 - setPassword, 41
 - setPoints, 42
 - setTimeRecord, 42
- player
 - Klondike, 28
- Player.cpp, 71
- Player.h, 71
- pos_X
 - Stack, 52
- pos_Y
 - Stack, 52
- record
 - Klondike, 28
- Register
 - Klondike, 23
- removeCards
 - Stack, 50
- removeFirstCard
 - Stack, 50
- returnCardInStackIndex
 - Stack, 50

- returnLastCard
 - Stack, [50](#)
- screenSize
 - Klondike, [28](#)
- setCards
 - Klondike, [23](#)
- setChangedTexture
 - myEvent, [31](#)
- setCurrentRecord
 - Player, [41](#)
- setHidden
 - Card, [11](#)
- setLogin
 - Player, [41](#)
- setOutline
 - Card, [11](#)
- setPassword
 - Player, [41](#)
- setPoints
 - Player, [42](#)
- setPos
 - Stack, [51](#)
- setPosition
 - Card, [11](#)
- setRecord
 - Klondike, [25](#)
- setTexture
 - Card, [12](#)
- setTimeRecord
 - Player, [42](#)
- setType
 - Pile, [36](#)
- Shape
 - Stack, [52](#)
- Shuffle, [42](#)
 - GenerateDeck, [43](#)
 - ShuffleDeck, [43](#)
- Shuffle.cpp, [73](#)
- Shuffle.h, [73](#)
- ShuffleDeck
 - Shuffle, [43](#)
- Stack, [44](#)
 - ~Stack, [47](#)
 - addCard, [48](#)
 - cards, [51](#)
 - draw, [48](#)
 - getPos, [48](#)
 - getShape, [48](#)
 - getStackType, [49](#)
 - height, [51](#)
 - id, [51](#)
 - isLastInStack, [49](#)
 - pos_X, [52](#)
 - pos_Y, [52](#)
 - removeCards, [50](#)
 - removeFirstCard, [50](#)
 - returnCardInStackIndex, [50](#)
 - returnLastCard, [50](#)
 - setPos, [51](#)
 - Shape, [52](#)
 - Stack, [47](#)
 - width, [52](#)
- Stack.cpp, [75](#)
- Stack.h, [75](#)
- stacks
 - Klondike, [26](#)
- TextureManager, [52](#)
 - ~TextureManager, [53](#)
 - getLength, [53](#)
 - getTexture, [53](#)
 - loadTexture, [54](#)
- TextureManager.cpp, [78](#)
- TextureManager.h, [78](#)
- timeRecord
 - Klondike, [28](#)
- Tutorial
 - Klondike, [27](#)
- width
 - Stack, [52](#)