

KText Editor

Generated by Doxygen 1.9.6



<b>1 Namespace Index</b>	<b>1</b>
1.1 Namespace List	1
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 File Index</b>	<b>7</b>
4.1 File List	7
<b>5 Namespace Documentation</b>	<b>9</b>
5.1 Command Namespace Reference	9
5.1.1 Detailed Description	9
5.2 KEYS Namespace Reference	9
5.2.1 Detailed Description	9
5.2.2 Enumeration Type Documentation	9
5.2.2.1 anonymous enum	9
<b>6 Class Documentation</b>	<b>11</b>
6.1 CmdBox Class Reference	11
6.1.1 Detailed Description	14
6.1.2 Member Function Documentation	14
6.1.2.1 TextBox()	15
6.2 Document Class Reference	15
6.2.1 Detailed Description	16
6.2.2 Constructor & Destructor Documentation	16
6.2.2.1 Document() [1/2]	16
6.2.2.2 Document() [2/2]	16
6.2.3 Member Function Documentation	17
6.2.3.1 createDir()	17
6.2.3.2 createFile()	17
6.2.3.3 getAbsPath()	17
6.2.3.4 getRelPath()	17
6.2.3.5 hasChanged()	17
6.2.3.6 init()	18
6.2.3.7 readFile()	18
6.2.3.8 saveFile()	18
6.2.4 Member Data Documentation	18
6.2.4.1 absPath	18
6.2.4.2 buffInfo	18
6.2.4.3 docChanged	18
6.2.4.4 relPath	19

6.3 Editor Class Reference	19
6.3.1 Detailed Description	20
6.3.2 Constructor & Destructor Documentation	20
6.3.2.1 Editor()	20
6.3.2.2 ~Editor()	20
6.3.3 Member Function Documentation	21
6.3.3.1 draw()	21
6.3.3.2 handleEvent()	21
6.3.4 Member Data Documentation	21
6.3.4.1 cbox	21
6.3.4.2 event	21
6.3.4.3 kb	21
6.3.4.4 textBox	22
6.3.4.5 window	22
6.4 Keyboard Class Reference	22
6.4.1 Detailed Description	24
6.4.2 Constructor & Destructor Documentation	24
6.4.2.1 Keyboard()	24
6.4.3 Member Function Documentation	25
6.4.3.1 backSpace()	25
6.4.3.2 getBounds()	25
6.4.3.3 getCmdTextEntered()	25
6.4.3.4 getTextEntered()	25
6.4.3.5 handleCmdKeyEvent()	25
6.4.3.6 handleKeyEvent()	26
6.4.3.7 handleMouseEvent()	26
6.4.3.8 isCmdTextEntered()	26
6.4.3.9 isKeyPressed()	27
6.4.3.10 isTextDeleted()	27
6.4.3.11 isTextEntered()	27
6.4.3.12 setCmdTextEntered()	27
6.4.3.13 setTextEntered()	28
6.4.4 Member Data Documentation	28
6.4.4.1 bounds	28
6.4.4.2 ctDeleted	28
6.4.4.3 ctEntered	28
6.4.4.4 tDeleted	28
6.4.4.5 tEntered	28
6.4.4.6 window	29
6.5 MyRect Class Reference	29
6.5.1 Detailed Description	32
6.5.2 Constructor & Destructor Documentation	32

6.5.2.1 MyRect()	32
6.5.3 Member Function Documentation	32
6.5.3.1 draw()	32
6.5.3.2 getPos()	33
6.5.3.3 getSize()	33
6.5.3.4 setFillColour()	33
6.5.3.5 setPosition()	33
6.5.3.6 setSize()	34
6.5.4 Member Data Documentation	34
6.5.4.1 fillColour	34
6.5.4.2 fRect	34
6.5.4.3 outlineColour	34
6.5.4.4 outlineThicknes	34
6.5.4.5 pos	34
6.5.4.6 size	35
6.6 Command::Stack< T > Class Template Reference	35
6.6.1 Constructor & Destructor Documentation	36
6.6.1.1 Stack()	36
6.6.2 Member Function Documentation	36
6.6.2.1 extend()	36
6.6.2.2 getMax()	36
6.6.2.3 pop()	36
6.6.2.4 printStack()	37
6.6.2.5 push()	37
6.6.3 Member Data Documentation	37
6.6.3.1 max_size	37
6.6.3.2 SP	37
6.6.3.3 SP_pos	37
6.6.3.4 stack_array	37
6.7 TextBox Class Reference	38
6.7.1 Detailed Description	40
6.7.2 Constructor & Destructor Documentation	40
6.7.2.1 TextBox()	41
6.7.3 Member Function Documentation	41
6.7.3.1 deleteChar()	41
6.7.3.2 draw()	41
6.7.3.3 enterPress()	42
6.7.3.4 getString()	42
6.7.3.5 getTextBox()	42
6.7.3.6 getTextColour()	42
6.7.3.7 getTextSize()	43
6.7.3.8 isMouseHover()	43

6.7.3.9 setFont()	43
6.7.3.10 setString()	43
6.7.3.11 setTextColour()	44
6.7.3.12 setTextSize()	44
6.7.4 Member Data Documentation	44
6.7.4.1 fcol	44
6.7.4.2 fname	44
6.7.4.3 font	44
6.7.4.4 fsize	45
6.7.4.5 mouseHover	45
6.7.4.6 tbox	45
6.7.4.7 window	45
<b>7 File Documentation</b>	<b>47</b>
7.1 include/Kamil/CmdBox.h File Reference	47
7.2 CmdBox.h	48
7.3 include/Kamil/Commands.h File Reference	48
7.4 Commands.h	48
7.5 include/Kamil/Document.h File Reference	49
7.5.1 Detailed Description	49
7.6 Document.h	50
7.7 include/Kamil/Editor.h File Reference	50
7.7.1 Detailed Description	51
7.8 Editor.h	51
7.9 include/Kamil/Keyboard.h File Reference	51
7.9.1 Detailed Description	52
7.10 Keyboard.h	53
7.11 include/Kamil/MyRect.h File Reference	53
7.11.1 Detailed Description	54
7.12 MyRect.h	55
7.13 include/Kamil/TextBox.h File Reference	55
7.14 TextBox.h	56
7.15 include/Kamil/Utils/Stack.h File Reference	57
7.16 Stack.h	58
<b>Index</b>	<b>59</b>

# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">Command</a>	A stack in the <a href="#">Command</a> namespace . . . . .	9
<a href="#">KEYS</a>	An enum for <a href="#">Keyboard</a> characters in hex form . . . . .	9





## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

Document . . . . .	15
sf::Drawable	
MyRect . . . . .	29
TextBox . . . . .	38
CmdBox . . . . .	11
Editor . . . . .	19
sf::FloatRect	
MyRect . . . . .	29
Keyboard . . . . .	22
Command::Stack< T > . . . . .	35



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">CmdBox</a>	Class to handle the command <a href="#">TextBox</a> . . . . .	11
<a href="#">Document</a>	<a href="#">Document</a> class . . . . .	15
<a href="#">Editor</a>	Class that handles and draws everything in the <a href="#">Editor</a> . . . . .	19
<a href="#">Keyboard</a>	A class to handle <a href="#">Keyboard</a> input . . . . .	22
<a href="#">MyRect</a>	Gives extra functionality to FloatRect . . . . .	29
<a href="#">Command::Stack&lt; T &gt;</a>	. . . . .	35
<a href="#">TextBox</a>	A class that makes a Textbox in SFML . . . . .	38



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

include/Kamil/ <a href="#">CmdBox.h</a> . . . . .	47
include/Kamil/ <a href="#">Commands.h</a> . . . . .	48
include/Kamil/ <a href="#">Document.h</a>	
Interface file for the <a href="#">Document</a> class . . . . .	49
include/Kamil/ <a href="#">Editor.h</a>	
Interface file for the <a href="#">Editor</a> class . . . . .	50
include/Kamil/ <a href="#">Keyboard.h</a>	
Interface file for <a href="#">Keyboard.h</a> . . . . .	51
include/Kamil/ <a href="#">MyRect.h</a>	
Interface file for the <a href="#">MyRect</a> class . . . . .	53
include/Kamil/ <a href="#">TextBox.h</a> . . . . .	55
include/Kamil/Utils/ <a href="#">Stack.h</a> . . . . .	57



## Chapter 5

# Namespace Documentation

### 5.1 Command Namespace Reference

A stack in the [Command](#) namespace.

#### Classes

- class [Stack](#)

#### 5.1.1 Detailed Description

A stack in the [Command](#) namespace.

### 5.2 KEYS Namespace Reference

An enum for [Keyboard](#) characters in hex form.

#### Enumerations

- enum {  
    [ESCAPE](#) = 0x1B , [ENTER](#) = 0xD , [BS](#) = 0x8 , [Shift\\_A](#) = 0x41 ,  
    [CTRL](#) = 0x11 , [DELETE](#) = 0x7f }

#### 5.2.1 Detailed Description

An enum for [Keyboard](#) characters in hex form.

#### 5.2.2 Enumeration Type Documentation

##### 5.2.2.1 anonymous enum

anonymous enum

## Enumerator

ESCAPE	
ENTER	
BS	
Shift_A	
CTRL	
DELETE	



## Chapter 6

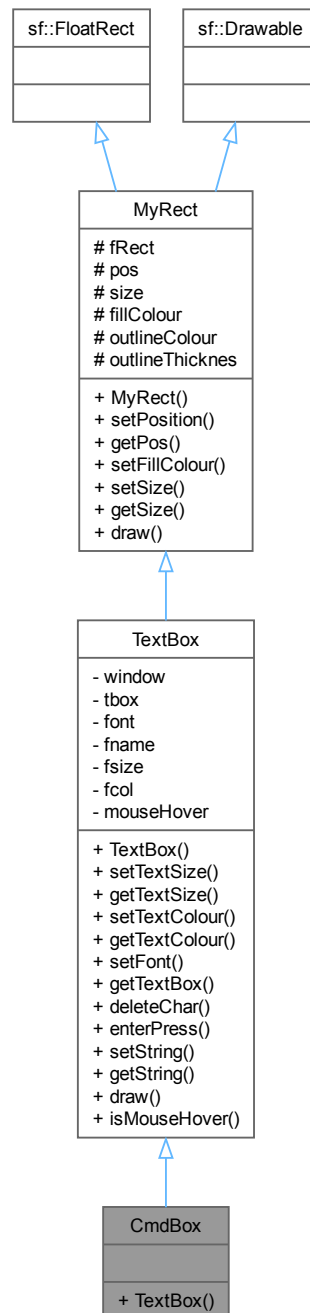
# Class Documentation

### 6.1 CmdBox Class Reference

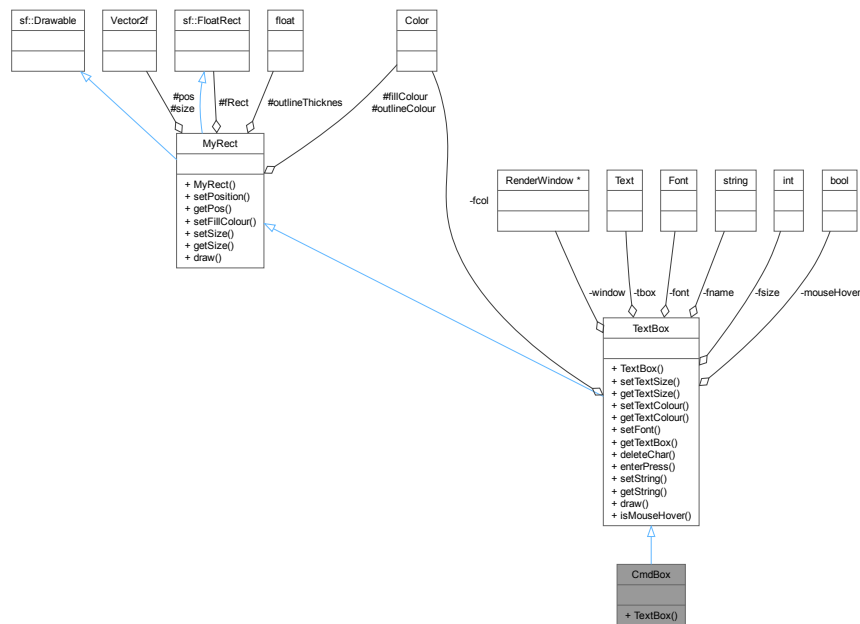
Class to handle the command [TextBox](#).

```
#include <CmdBox.h>
```

Inheritance diagram for CmdBox:



Collaboration diagram for CmdBox:



## Public Member Functions

- [TextBox](#) (sf::RenderWindow \*win, sf::Vector2f pos, sf::Vector2f size, std::string sfont, int fsize, sf::Color fcol, sf::Color background, float thicc)

Using teh Parent class constructor.

## Public Member Functions inherited from [TextBox](#)

- [TextBox](#) (sf::RenderWindow \*win, sf::Vector2f pos, sf::Vector2f size, std::string sfont, int fsize, sf::Color fcol, sf::Color background, float thicc)  
Constructor for [TextBox](#).
- void [setTextSize](#) (int size)  
Set the size of the text.
- int [getTextSize](#) () const  
Get the size of the text.
- void [setTextColour](#) (sf::Color colour)  
Set the colour of the text.
- sf::Color [getTextColour](#) () const  
Get the colour of the text.
- void [setFont](#) (sf::Font &font)  
set what font you use
- sf::Text [getTextBox](#) () const  
Get both the main textbox and the cmd textbox in a vector.
- void [deleteChar](#) ()  
Delete last character entered.
- void [enterPress](#) ()  
Handles Enter key press.

- void [setString](#) (std::string nstring)  
*Sets the string.*
- std::string [getString](#) () const  
*returns the text in tbox*
- void [draw](#) (sf::RenderTarget &target, sf::RenderStates states) const override  
*used to draw to the screen virutal method inherited from [MyRect](#) -> sf::Drawable thats overrided here is what allows us to draw to window using window.draw(TextBox)*
- bool [isMouseHover](#) ()  
*check if mouse is hovering over current textbox*

### Public Member Functions inherited from [MyRect](#)

- [MyRect](#) (sf::Vector2f [pos](#), sf::Vector2f [size](#), sf::Color [fillColour](#), sf::Color [outlineColour](#), float [outlineThicknes](#))  
*constructor for [MyRect](#)*
- void [setPosition](#) (sf::Vector2f [pos](#))  
*sets the position of rect*
- sf::Vector2f [getPos](#) () const  
*get the position of rect*
- void [setFillColour](#) (sf::Color colour)  
*set the fill colour of the rect*
- void [setSize](#) (sf::Vector2f [size](#))  
*set the size of the rect*
- sf::Vector2f [getSize](#) () const  
*get the size of the rect*
- void [draw](#) (sf::RenderTarget &target, sf::RenderStates states) const override  
*virutal method to draw to window*

### Additional Inherited Members

#### Protected Attributes inherited from [MyRect](#)

- sf::FloatRect [fRect](#)
- sf::Vector2f [pos](#)
- sf::Vector2f [size](#)
- sf::Color [fillColour](#)
- sf::Color [outlineColour](#)
- float [outlineThicknes](#)

#### 6.1.1 Detailed Description

Class to handle the command [TextBox](#).

#### 6.1.2 Member Function Documentation

### 6.1.2.1 TextBox()

```

TextBox::TextBox (
    sf::RenderWindow * win,
    sf::Vector2f pos,
    sf::Vector2f size,
    std::string sfont,
    int fsize,
    sf::Color fcol,
    sf::Color background,
    float thicc )

```

Using teh Parent class constructor.

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

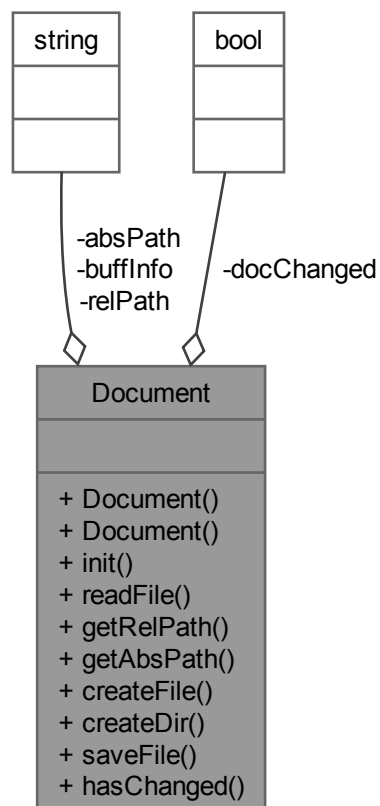
- [include/Kamil/CmdBox.h](#)

## 6.2 Document Class Reference

[Document](#) class.

```
#include <Document.h>
```

Collaboration diagram for Document:



## Public Member Functions

- [Document](#) ()  
*Constructor for [Document](#) class.*
- [Document](#) (std::string fileP)  
*Constructor for [Document](#) class.*
- void [init](#) ()  
*initialise the file TODO: overload this with file path*
- void [readFile](#) ()  
*read the file*
- std::string [getRelPath](#) ()  
*get the relative path*
- std::string [getAbsPath](#) ()  
*get the relative path*
- void [createFile](#) ()  
*create the file*
- void [createDir](#) ()  
*create a directory*
- bool [saveFile](#) (const std::string &filename)  
*save to a file*
- bool [hasChanged](#) ()  
*if the file has changed*

## Private Attributes

- std::string [relPath](#)
- std::string [absPath](#)
- std::string [buffInfo](#)
- bool [docChanged](#)

## 6.2.1 Detailed Description

[Document](#) class.

## 6.2.2 Constructor & Destructor Documentation

### 6.2.2.1 [Document](#)() [1/2]

```
Document::Document ( )
```

Constructor for [Document](#) class.

### 6.2.2.2 [Document](#)() [2/2]

```
Document::Document (
    std::string fileP )
```

Constructor for [Document](#) class.

## Parameters

<i>fileP</i>	- file path
--------------	-------------

## 6.2.3 Member Function Documentation

### 6.2.3.1 createDir()

```
void Document::createDir ( )
```

create a directory

### 6.2.3.2 createFile()

```
void Document::createFile ( )
```

create the file

### 6.2.3.3 getAbsPath()

```
std::string Document::getAbsPath ( )
```

get the relative path

### 6.2.3.4 getRelPath()

```
std::string Document::getRelPath ( )
```

get the relative path

### 6.2.3.5 hasChanged()

```
bool Document::hasChanged ( )
```

if the file has changed

## Returns

bool - true if file has changed

#### 6.2.3.6 init()

```
void Document::init ( )
```

initialise the file TODO: overload this with file path

#### 6.2.3.7 readFile()

```
void Document::readFile ( )
```

read the file

#### 6.2.3.8 saveFile()

```
bool Document::saveFile (
    const std::string & filename )
```

save to a file

### 6.2.4 Member Data Documentation

#### 6.2.4.1 absPath

```
std::string Document::absPath [private]
```

absolute path

#### 6.2.4.2 buffInfo

```
std::string Document::buffInfo [private]
```

#### 6.2.4.3 docChanged

```
bool Document::docChanged [private]
```

buffer information (the file text) if the file has changed



## 6.2.4.4 relPath

```
std::string Document::relPath [private]
```

relative path

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

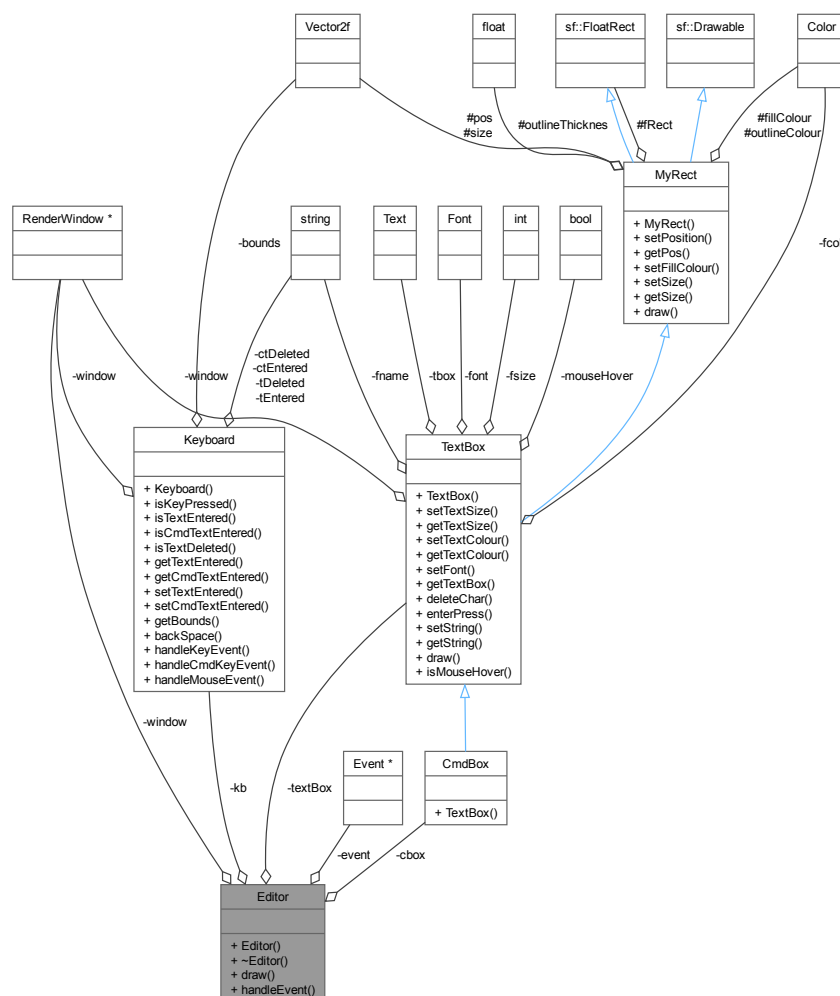
- include/Kamil/[Document.h](#)

## 6.3 Editor Class Reference

Class that handles and draws everything in the [Editor](#).

```
#include <Editor.h>
```

Collaboration diagram for Editor:



## Public Member Functions

- [Editor](#) (sf::RenderWindow \*, sf::Event \*)  
*Constructor for [Editor](#).*
- [~Editor](#) ()  
*Destructor for [Editor](#) class.*
- void [draw](#) ()  
*function that draws everything to RenderWindow*
- void [handleEvent](#) ()  
*handle the events for the [Editor](#)*

## Private Attributes

- [TextBox](#) \* [textBox](#)
- [CmdBox](#) \* [cbox](#)
- sf::RenderWindow \* [window](#)
- sf::Event \* [event](#)
- [Keyboard](#) [kb](#)

### 6.3.1 Detailed Description

Class that handles and draws everything in the [Editor](#).

### 6.3.2 Constructor & Destructor Documentation

#### 6.3.2.1 Editor()

```
Editor::Editor (
    sf::RenderWindow * ,
    sf::Event * )
```

Constructor for [Editor](#).

##### Parameters

<i>window</i>	- reference to main RenderWindow
<i>event</i>	- reference to main event

#### 6.3.2.2 ~Editor()

```
Editor::~~Editor ( )
```

Destructor for [Editor](#) class.

### 6.3.3 Member Function Documentation

#### 6.3.3.1 draw()

```
void Editor::draw ( )
```

function that draws everything to RenderWindow

DEPRECATED

#### 6.3.3.2 handleEvent()

```
void Editor::handleEvent ( )
```

handle the events for the [Editor](#)

where all event handles are called when interacting with other classes e.g. `kb.handleEvent(); kb.handleMouse↵Events();`

### 6.3.4 Member Data Documentation

#### 6.3.4.1 cbox

```
CmdBox* Editor::cbox [private]
```

reference to command box that we draw

#### 6.3.4.2 event

```
sf::Event* Editor::event [private]
```

refernce to event

#### 6.3.4.3 kb

```
Keyboard Editor::kb [private]
```

handles keyboard events

#### 6.3.4.4 textBox

```
TextBox* Editor::textBox [private]
```

reference to textbox that we draw

#### 6.3.4.5 window

```
sf::RenderWindow* Editor::window [private]
```

refernce to RenderWindow

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

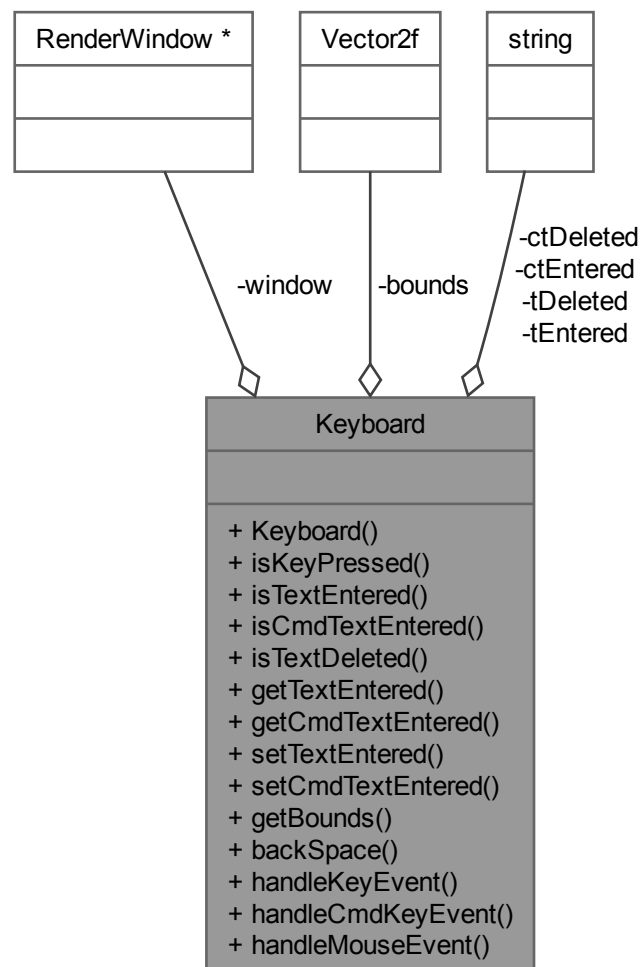
- include/Kamil/[Editor.h](#)

## 6.4 Keyboard Class Reference

A class to handle [Keyboard](#) input.

```
#include <Keyboard.h>
```

Collaboration diagram for Keyboard:



## Public Member Functions

- [Keyboard](#) (sf::RenderWindow \*win, sf::Vector2f [bounds](#))  
*Constructor for [Keyboard](#) class.*
- bool [isKeyPressed](#) (sf::Keyboard::Key)  
*checks if a key is pressed*
- bool [isTextEntered](#) ()  
*checks if a text is entered*
- bool [isCmdTextEntered](#) ()  
*checks if text is entered to the command box*
- bool [isTextDeleted](#) ()  
*check if text is being deleted*
- std::string [getTextEntered](#) ()  
*returns text entered*

- `std::string getCmdTextEntered ()`  
*returns text entered*
- `void setTextEntered (std::string)`  
*sets text*
- `void setCmdTextEntered (std::string)`  
*sets text*
- `sf::Vector2f getBounds () const`  
*get the bounds of the area we are in*
- `void backSpace ()`  
*when we backspace on teh text*
- `void handleKeyEvent (sf::Event &event)`  
*handle keyboard events*
- `void handleCmdKeyEvent ()`  
*handle keyboard events*
- `void handleMouseEvent (sf::Event &event)`  
*mouse keyboard events*

## Private Attributes

- `sf::RenderWindow * window`
- `sf::Vector2f bounds`
- `std::string tEntered`
- `std::string tDeleted`
- `std::string ctEntered`
- `std::string ctDeleted`

## 6.4.1 Detailed Description

A class to handle [Keyboard](#) input.

## 6.4.2 Constructor & Destructor Documentation

### 6.4.2.1 Keyboard()

```
Keyboard::Keyboard (
    sf::RenderWindow * win,
    sf::Vector2f bounds )
```

Constructor for [Keyboard](#) class.

#### Parameters

<i>win</i>	- reference to main window
<i>bounds</i>	- bounds of the window we are working in

## 6.4.3 Member Function Documentation

### 6.4.3.1 backSpace()

```
void Keyboard::backSpace ( )
```

when we backspace on teh text

### 6.4.3.2 getBounds()

```
sf::Vector2f Keyboard::getBounds ( ) const
```

get the bounds of the area we are in

#### Returns

sf::Vector2f bounded area

### 6.4.3.3 getCmdTextEntered()

```
std::string Keyboard::getCmdTextEntered ( )
```

returns text entered

#### Returns

std::string text entered

### 6.4.3.4 getTextEntered()

```
std::string Keyboard::getTextEntered ( )
```

returns text entered

#### Returns

std::string text entered

### 6.4.3.5 handleCmdKeyEvent()

```
void Keyboard::handleCmdKeyEvent ( )
```

handle keyboard events

**Parameters**

<i>event</i>	- to get text entered from events
--------------	-----------------------------------

**6.4.3.6 handleKeyEvent()**

```
void Keyboard::handleKeyEvent (
    sf::Event & event )
```

handle keyboard events

**Parameters**

<i>event</i>	- to get text entered from events
--------------	-----------------------------------

**6.4.3.7 handleMouseEvent()**

```
void Keyboard::handleMouseEvent (
    sf::Event & event )
```

mouse keyboard events

**Parameters**

<i>event</i>	- to get text entered from events
--------------	-----------------------------------

**6.4.3.8 isCmdTextEntered()**

```
bool Keyboard::isCmdTextEntered ( )
```

checks if text is entered to the command box

**Returns**

bool true if key is pressed false if not



#### 6.4.3.9 isKeyPressed()

```
bool Keyboard::isKeyPressed (
    sf::Keyboard::Key )
```

checks if a key is pressed

##### Returns

bool true if key is pressed false if not

#### 6.4.3.10 isTextDeleted()

```
bool Keyboard::isTextDeleted ( )
```

check if text is being deleted

##### Returns

bool true if text is being deleted

#### 6.4.3.11 isTextEntered()

```
bool Keyboard::isTextEntered ( )
```

checks if a text is entered

##### Returns

bool true if key is pressed false if not

#### 6.4.3.12 setCmdTextEntered()

```
void Keyboard::setCmdTextEntered (
    std::string )
```

sets text

##### Parameters

<i>nstring</i>	- new string
----------------	--------------

#### 6.4.3.13 setTextEntered()

```
void Keyboard::setTextEntered (
    std::string )
```

sets text

##### Parameters

<i>nstring</i>	- new string
----------------	--------------

### 6.4.4 Member Data Documentation

#### 6.4.4.1 bounds

```
sf::Vector2f Keyboard::bounds [private]
```

store the bounded area

#### 6.4.4.2 ctDeleted

```
std::string Keyboard::ctDeleted [private]
```

tmp for text deleted to cmd

#### 6.4.4.3 ctEntered

```
std::string Keyboard::ctEntered [private]
```

tmp for text entered to cmd

#### 6.4.4.4 tDeleted

```
std::string Keyboard::tDeleted [private]
```

the text deleted from main box

#### 6.4.4.5 tEntered

```
std::string Keyboard::tEntered [private]
```

the text entered to main box

#### 6.4.4.6 window

```
sf::RenderWindow* Keyboard::window [private]
```

reference to window

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

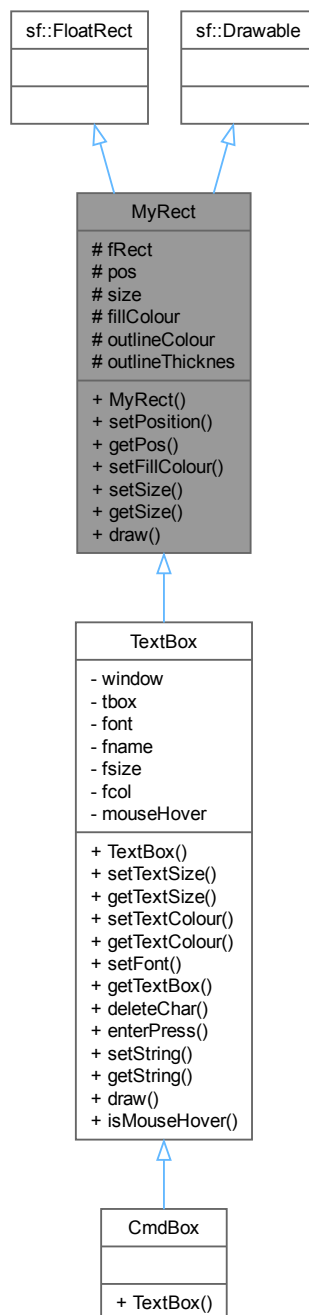
- include/Kamil/[Keyboard.h](#)

## 6.5 MyRect Class Reference

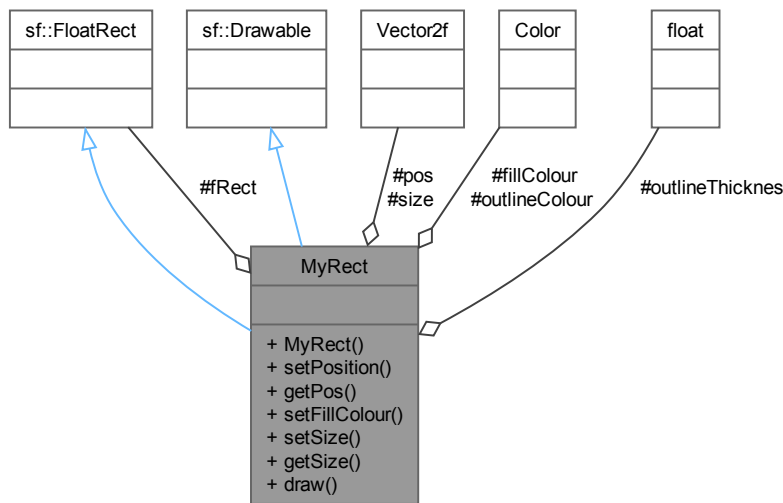
gives extra functionality to FloatRect

```
#include <MyRect.h>
```

Inheritance diagram for MyRect:



Collaboration diagram for MyRect:



## Public Member Functions

- **MyRect** (`sf::Vector2f pos`, `sf::Vector2f size`, `sf::Color fillColour`, `sf::Color outlineColour`, `float outlineThicknes`)  
*constructor for MyRect*
- void **setPosition** (`sf::Vector2f pos`)  
*sets the position of rect*
- `sf::Vector2f` **getPos** () const  
*get the position of rect*
- void **setFillColour** (`sf::Color colour`)  
*set the fill colour of the rect*
- void **setSize** (`sf::Vector2f size`)  
*set the size of the rect*
- `sf::Vector2f` **getSize** () const  
*get the size of the rect*
- void **draw** (`sf::RenderTarget &target`, `sf::RenderStates states`) const override  
*virtual method to draw to window*

## Protected Attributes

- `sf::FloatRect` **fRect**
- `sf::Vector2f` **pos**
- `sf::Vector2f` **size**
- `sf::Color` **fillColour**
- `sf::Color` **outlineColour**
- `float` **outlineThicknes**

### 6.5.1 Detailed Description

gives extra functionality to FloatRect

Uses FloatRect for the ability to collision detect better than RectangleShape and inherits from Drawable so we are able to keep uniform syntax of window.draw(Drawable object)

### 6.5.2 Constructor & Destructor Documentation

#### 6.5.2.1 MyRect()

```
MyRect::MyRect (
    sf::Vector2f pos,
    sf::Vector2f size,
    sf::Color fillColour,
    sf::Color outlineColour,
    float outlineThicknes )
```

constructor for [MyRect](#)

##### Parameters

<i>pos</i>	- position of rect
<i>size</i>	- size of rect
<i>fillColour</i>	- fill colour of rect
<i>outlineColour</i>	- outline colour of rect
<i>outlineThicknes</i>	- outline thickness of rect

### 6.5.3 Member Function Documentation

#### 6.5.3.1 draw()

```
void MyRect::draw (
    sf::RenderTarget & target,
    sf::RenderStates states ) const [override]
```

virutal method to draw to window

Inherited from sf::Drawable it is what allows us to draw to the screen using window.draw(MyRect); instead of MyRect.draw(window) keeping similar drawing standard to base SFML code making our class more modular and familiar to those who use SFML

Example of polymorphism

### 6.5.3.2 getPos()

```
sf::Vector2f MyRect::getPos ( ) const
```

get the position of rect

#### Returns

sf::Vector2f pos

### 6.5.3.3 getSize()

```
sf::Vector2f MyRect::getSize ( ) const
```

get the size of the rect

#### Returns

sf::Vector2f size

### 6.5.3.4 setFillColour()

```
void MyRect::setFillColour (
    sf::Color colour )
```

set the fill colour of the rect

#### Parameters

sf::Color	colour
-----------	--------

### 6.5.3.5 setPosition()

```
void MyRect::setPosition (
    sf::Vector2f pos )
```

sets the position of rect

#### Parameters

sf::Vector2f	pos
--------------	-----

#### 6.5.3.6 setSize()

```
void MyRect::setSize (
    sf::Vector2f size )
```

set the size of the rect

##### Parameters

<i>sf::Vector2f</i>	size
---------------------	------

### 6.5.4 Member Data Documentation

#### 6.5.4.1 fillColour

```
sf::Color MyRect::fillColour [protected]
```

colour of rect

#### 6.5.4.2 fRect

```
sf::FloatRect MyRect::fRect [protected]
```

for collision checking

#### 6.5.4.3 outlineColour

```
sf::Color MyRect::outlineColour [protected]
```

outline colour of rect

#### 6.5.4.4 outlineThicknes

```
float MyRect::outlineThicknes [protected]
```

outline thickness of rect

#### 6.5.4.5 pos

```
sf::Vector2f MyRect::pos [protected]
```

position of rect



## 6.5.4.6 size

```
sf::Vector2f MyRect::size [protected]
```

size of rect

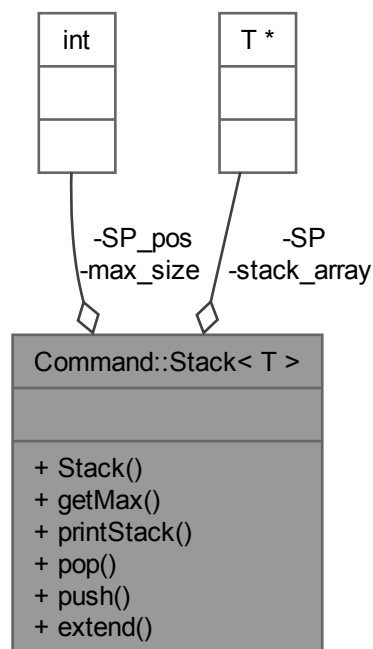
The documentation for this class was generated from the following file:

- include/Kamil/MyRect.h

## 6.6 Command::Stack&lt; T &gt; Class Template Reference

```
#include <Stack.h>
```

Collaboration diagram for Command::Stack< T >:



## Public Member Functions

- `Stack` (int)
- int `getMax` () const
- void `printStack` () const
- int `pop` ()
- int `push` (T)
- void `extend` (int)

## Private Attributes

- int `max_size` {}
- T \* `stack_array` {new T[`max_size`]}
- T \* `SP` = &`stack_array`[`max_size`]
- int `SP_pos` = `max_size`

## 6.6.1 Constructor & Destructor Documentation

### 6.6.1.1 Stack()

```
template<typename T >  
Command::Stack< T >::Stack (  
    int )
```

## 6.6.2 Member Function Documentation

### 6.6.2.1 extend()

```
template<typename T >  
void Command::Stack< T >::extend (  
    int )
```

### 6.6.2.2 getMax()

```
template<typename T >  
int Command::Stack< T >::getMax ( ) const
```

### 6.6.2.3 pop()

```
template<typename T >  
int Command::Stack< T >::pop ( )
```

#### 6.6.2.4 printStack()

```
template<typename T >
void Command::Stack< T >::printStack ( ) const
```

#### 6.6.2.5 push()

```
template<typename T >
int Command::Stack< T >::push (
    T )
```

### 6.6.3 Member Data Documentation

#### 6.6.3.1 max\_size

```
template<typename T >
int Command::Stack< T >::max_size {} [private]
```

#### 6.6.3.2 SP

```
template<typename T >
T* Command::Stack< T >::SP = &stack_array[max_size] [private]
```

#### 6.6.3.3 SP\_pos

```
template<typename T >
int Command::Stack< T >::SP_pos = max_size [private]
```

#### 6.6.3.4 stack\_array

```
template<typename T >
T* Command::Stack< T >::stack_array {new T[max_size]} [private]
```

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

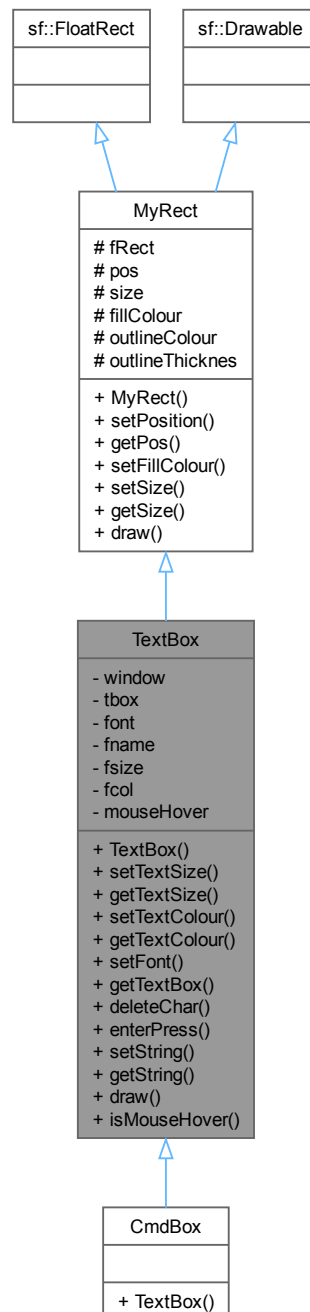
- include/Kamil/Commands.h
- include/Kamil/Utils/Stack.h

## 6.7 TextBox Class Reference

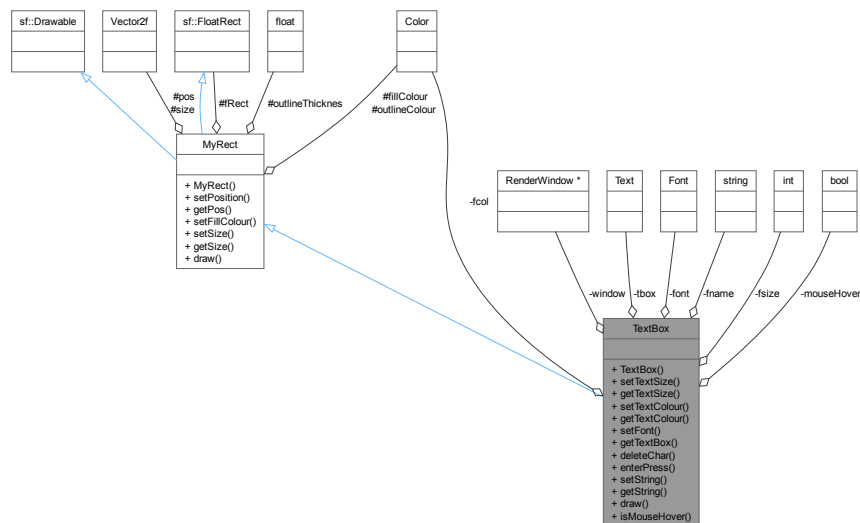
A class that makes a Textbox in SFML.

```
#include <TextBox.h>
```

Inheritance diagram for TextBox:



Collaboration diagram for TextBox:



## Public Member Functions

- **TextBox** (sf::RenderWindow \*win, sf::Vector2f pos, sf::Vector2f size, std::string sfont, int fsize, sf::Color fcol, sf::Color background, float thicc)  
*Constructor for TextBox.*
- void **setTextSize** (int size)  
*Set the size of the text.*
- int **getTextSize** () const  
*Get the size of the text.*
- void **setTextColour** (sf::Color colour)  
*Set the colour of the text.*
- sf::Color **getTextColour** () const  
*Get the colour of the text.*
- void **setFont** (sf::Font &font)  
*set what font you use*
- sf::Text **getTextBox** () const  
*Get both the main textbox and the cmd textbox in a vector.*
- void **deleteChar** ()  
*Delete last character entered.*
- void **enterPress** ()  
*Handles Enter key press.*
- void **setString** (std::string nstring)  
*Sets the string.*
- std::string **getString** () const  
*returns the text in tbox*
- void **draw** (sf::RenderTarget &target, sf::RenderStates states) const override  
*used to draw to the screen virtual method inherited from MyRect -> sf::Drawable thats overrided here is what allows us to draw to window using window.draw(TextBox)*
- bool **isMouseHover** ()  
*check if mouse is hovering over current textbox*

### Public Member Functions inherited from [MyRect](#)

- [MyRect](#) (sf::Vector2f [pos](#), sf::Vector2f [size](#), sf::Color [fillColour](#), sf::Color [outlineColour](#), float [outlineThicknes](#))  
*constructor for [MyRect](#)*
- void [setPosition](#) (sf::Vector2f [pos](#))  
*sets the position of rect*
- sf::Vector2f [getPos](#) () const  
*get the position of rect*
- void [setFillColour](#) (sf::Color colour)  
*set the fill colour of the rect*
- void [setSize](#) (sf::Vector2f [size](#))  
*set the size of the rect*
- sf::Vector2f [getSize](#) () const  
*get the size of the rect*
- void [draw](#) (sf::RenderTarget &target, sf::RenderStates states) const override  
*virutal method to draw to window*

### Private Attributes

- sf::RenderWindow \* [window](#)
- sf::Text [tbox](#) {}
- sf::Font [font](#) {}
- std::string [fname](#) {}
- int [fsize](#) {}
- sf::Color [fcol](#) {}
- bool [mouseHover](#)

### Additional Inherited Members

#### Protected Attributes inherited from [MyRect](#)

- sf::FloatRect [fRect](#)
- sf::Vector2f [pos](#)
- sf::Vector2f [size](#)
- sf::Color [fillColour](#)
- sf::Color [outlineColour](#)
- float [outlineThicknes](#)

#### 6.7.1 Detailed Description

A class that makes a Textbox in SFML.

The class creates a textbox for inputting and handling text and [Keyboard](#) commands and allows the use of commands in the secondary textbox cmdbox

#### 6.7.2 Constructor & Destructor Documentation

### 6.7.2.1 TextBox()

```

TextBox::TextBox (
    sf::RenderWindow * win,
    sf::Vector2f pos,
    sf::Vector2f size,
    std::string sfont,
    int fsize,
    sf::Color fcol,
    sf::Color background,
    float thicc )

```

Constructor for [TextBox](#).

#### Parameters

<i>win</i>	- RenderWindow the <a href="#">TextBox</a> is drawn onto
<i>pos</i>	- the initial position of the <a href="#">TextBox</a>
<i>size</i>	- the initial size of the <a href="#">TextBox</a>
<i>sfont</i>	- the initial font used by the <a href="#">TextBox</a>
<i>fsize</i>	- the initial font size
<i>fcol</i>	- the initial font colour
<i>background</i>	- the initial background colour
<i>thicc</i>	- the padding for the RectangleShape

## 6.7.3 Member Function Documentation

### 6.7.3.1 deleteChar()

```
void TextBox::deleteChar ( )
```

Delete last character entered.

### 6.7.3.2 draw()

```

void TextBox::draw (
    sf::RenderTarget & target,
    sf::RenderStates states ) const [override]

```

used to draw to the screen virtual method inherited from [MyRect](#) -> sf::Drawable that's overridden here is what allows us to draw to window using window.draw(TextBox)

Example of polymorphism

#### 6.7.3.3 enterPress()

```
void TextBox::enterPress ( )
```

Handles Enter key press.

#### 6.7.3.4 getString()

```
std::string TextBox::getString ( ) const
```

returns the text in tbox

##### Returns

type std::string

#### 6.7.3.5 getTextBox()

```
sf::Text TextBox::getTextBox ( ) const
```

Get both the main textbox and the cmd textbox in a vector.

##### Returns

type Boxv2 that contains textbox and cmdbox

#### 6.7.3.6 getTextColour()

```
sf::Color TextBox::getTextColour ( ) const
```

Get the colour of the text.

##### Returns

sf::Colour textColour



#### 6.7.3.7 getTextSize()

```
int TextBox::getTextSize ( ) const
```

Get the size of the text.

##### Returns

an int of the text size

#### 6.7.3.8 isMouseHover()

```
bool TextBox::isMouseHover ( )
```

check if mouse is hovering over current textbox

##### Returns

bool - yes if hovering

#### 6.7.3.9 setFont()

```
void TextBox::setFont (
    sf::Font & font )
```

set what font you use

##### Parameters

<i>font</i>	file dir of font
-------------	------------------

#### 6.7.3.10 setString()

```
void TextBox::setString (
    std::string nstring )
```

Sets the string.

##### Parameters

<i>nstring</i>	- new string placed on tbox
----------------	-----------------------------

#### 6.7.3.11 setTextColour()

```
void TextBox::setTextColour (
    sf::Color colour )
```

Set the colour of the text.

##### Parameters

<i>fill</i>	font colour
-------------	-------------

#### 6.7.3.12 setTextSize()

```
void TextBox::setTextSize (
    int size )
```

Set the size of the text.

##### Parameters

<i>size</i>	text size
-------------	-----------

### 6.7.4 Member Data Documentation

#### 6.7.4.1 fcol

```
sf::Color TextBox::fcol {} [private]
```

the font colour

#### 6.7.4.2 fname

```
std::string TextBox::fname {} [private]
```

the name of the font used

#### 6.7.4.3 font

```
sf::Font TextBox::font {} [private]
```

the font that the [TextBox](#) uses

#### 6.7.4.4 fsize

```
int TextBox::fsize {} [private]
```

the font size

#### 6.7.4.5 mouseHover

```
bool TextBox::mouseHover [private]
```

if the mouse is hovering over

#### 6.7.4.6 tbox

```
sf::Text TextBox::tbox {} [private]
```

the text that everything is written onto

#### 6.7.4.7 window

```
sf::RenderWindow* TextBox::window [private]
```

pointer to the main RenderWindow variable

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

- include/Kamil/[TextBox.h](#)



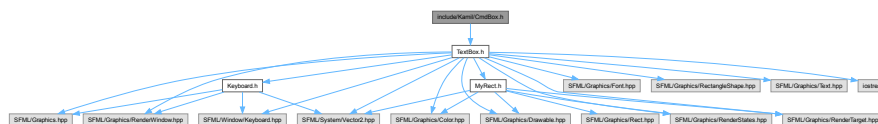
## Chapter 7

# File Documentation

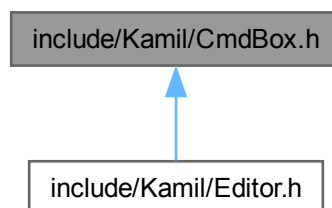
### 7.1 include/Kamil/CmdBox.h File Reference

```
#include "TextBox.h"
```

Include dependency graph for CmdBox.h:



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



### Classes

- class [CmdBox](#)  
*Class to handle the command [TextBox](#).*

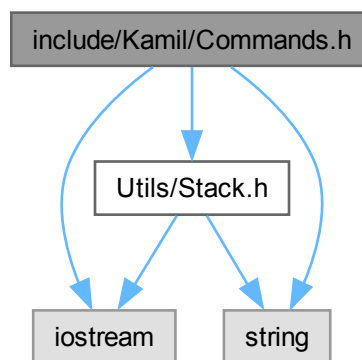
## 7.2 CmdBox.h

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

```
00001 #ifndef KAMIL_CMDBOX_H
00002 #define KAMIL_CMDBOX_H
00003
00013 #include "TextBox.h"
00014
00018 class CmdBox : public TextBox
00019 {
00020 public:
00024     using TextBox::TextBox;
00025 };
00026 #endif // KAMIL_CMDBOX_H
```

## 7.3 include/Kamil/Commands.h File Reference

```
#include <iostream>
#include <string>
#include "Utils/Stack.h"
Include dependency graph for Commands.h:
```



## Namespaces

- namespace `Command`  
A stack in the `Command` namespace.

## 7.4 Commands.h

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

```
00001 #ifndef KAMIL_COMMANDS_H
00002 #define KAMIL_COMMANDS_H
00003
00004 #include <iostream>
00005 #include <string>
00006
00007 #include "Utils/Stack.h"
```

```

00008
00009 namespace Command{
00010     // template <typename T>
00011     // class Node{ // used for LinkedList
00012     //     public:
00013     //         Node();
00014     //         Node(T);
00015     //         T data;
00016     //         Node* next;
00017     // };
00018
00019     // template <typename T>
00020     // class LinkedList{
00021     //     public:
00022     //         LinkedList();
00023     //         void insertNode(int);
00024     //         void printList();
00025     //         void deleteNode(int);
00026     //     private:
00027     //         Node<T>* head;
00028     // };
00029     template <typename>
00030     class Stack;
00031
00032 //     class Undo{};
00033
00034 //     class Redo{};
00035 }
00036
00037 #endif // KAMIL_COMMANDS_H

```

## 7.5 include/Kamil/Document.h File Reference

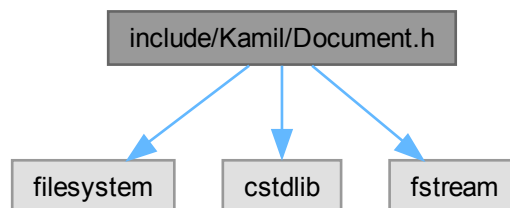
Interface file for the [Document](#) class.

```

#include <filesystem>
#include <cstdlib>
#include <fstream>

```

Include dependency graph for Document.h:



### Classes

- class [Document](#)  
*Document* class.

### 7.5.1 Detailed Description

Interface file for the [Document](#) class.

The [Document.h](#) file is responsible for all File I/O between the system and the program it can read and write files and will also push some work off to python scripts to handle config files

## 7.6 Document.h

Go to the documentation of this file.

```

00001 #ifndef KAMIL_DOCUMENT_H
00002 #define KAMIL_DOCUMENT_H
00003
00004
00014 #include <filesystem>
00015 #include <cstdlib>
00016 #include <fstream>
00017
00021 class Document{
00022 public:
00026     Document();
00027
00032     Document(std::string fileP);
00033
00038     void init();
00039
00043     void readFile();
00044
00048     std::string getRelPath();
00049
00053     std::string getAbsPath();
00054
00058     void createFile();
00059
00063     void createDir();
00064
00068     bool saveFile(const std::string& filename);
00069
00074     bool hasChanged();
00075
00076     // void addTextToPos(std::string txt, int pos);
00077 private:
00079     std::string relPath;
00080     std::string absPath;
00082     std::string buffInfo;
00084     bool docChanged;
00085 };
00086 #endif // KAMIL_DOCUMENT_H

```

## 7.7 include/Kamil/Editor.h File Reference

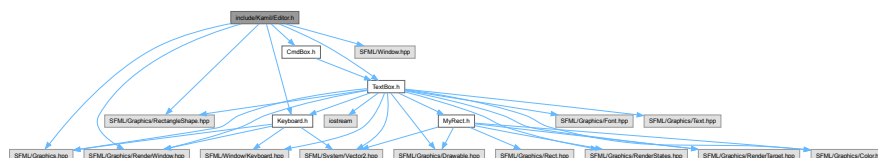
Interface file for the [Editor](#) class.

```

#include <SFML/Graphics.hpp>
#include <SFML/Graphics/RectangleShape.hpp>
#include <SFML/Graphics/RenderWindow.hpp>
#include <SFML/Window.hpp>
#include "TextBox.h"
#include "Keyboard.h"
#include "CmdBox.h"

```

Include dependency graph for Editor.h:



## Classes

- class [Editor](#)

*Class that handles and draws everything in the [Editor](#).*



### 7.7.1 Detailed Description

Interface file for the [Editor](#) class.

The [Editor](#) class is responsible for the interaction between the different classes. All things outside the main while loop will be checked or initialise. Anything to do with the [Editor](#) Window will happen here

## 7.8 Editor.h

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

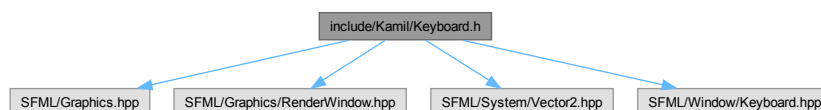
```
00001 #ifndef KAMIL_EDITOR_WINDOW_HPP
00002 #define KAMIL_EDITOR_WINDOW_HPP
00003
00016 #include <SFML/Graphics.hpp>
00017 #include <SFML/Graphics/RectangleShape.hpp>
00018 #include <SFML/Graphics/RenderWindow.hpp>
00019 #include <SFML/Window.hpp>
00020
00021 #include "TextBox.h"
00022 #include "Keyboard.h"
00023 #include "CmdBox.h"
00024
00025
00029 class Editor{
00030     public:
00036         Editor(sf::RenderWindow*, sf::Event*);
00037
00041         ~Editor();
00042
00048         void draw();
00049
00056         void handleEvent();
00057     private:
00058         TextBox* textBox;
00059         CmdBox* cbox;
00060         sf::RenderWindow* window;
00061         sf::Event* event;
00062         Keyboard kb;
00063 };
00064
00065 #endif // KAMIL_EDITOR_WINDOW_HPP
```

## 7.9 include/Kamil/Keyboard.h File Reference

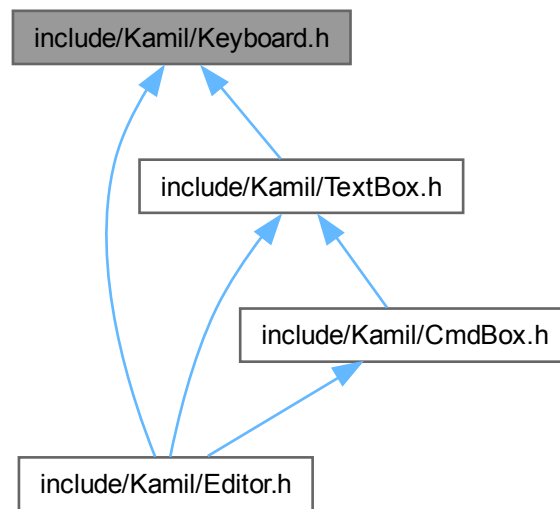
Interface file for [Keyboard.h](#).

```
#include <SFML/Graphics.hpp>
#include <SFML/Graphics/RenderWindow.hpp>
#include <SFML/System/Vector2.hpp>
#include <SFML/Window/Keyboard.hpp>
```

Include dependency graph for Keyboard.h:



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



## Classes

- class [Keyboard](#)  
*A class to handle [Keyboard](#) input.*

## Namespaces

- namespace [KEYS](#)  
*An enum for [Keyboard](#) characters in hex form.*

## Enumerations

- enum {  
[KEYS::ESCAPE](#) = 0x1B , [KEYS::ENTER](#) = 0xD , [KEYS::BS](#) = 0x8 , [KEYS::Shift\\_A](#) = 0x41 ,  
[KEYS::CTRL](#) = 0x11 , [KEYS::DELETE](#) = 0x7f }

### 7.9.1 Detailed Description

Interface file for [Keyboard.h](#).

A class that handles all keyboard and mouse events for the editor is responsible for mangning input of keyboard data and their corresponding command

## 7.10 Keyboard.h

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

```

00001 #ifndef KAMIL_KEYBOARD_H
00002 #define KAMIL_KEYBOARD_H
00003
00004
00015 #include <SFML/Graphics.hpp>
00016 #include <SFML/Graphics/RenderWindow.hpp>
00017 #include <SFML/System/Vector2.hpp>
00018 #include <SFML/Window/Keyboard.hpp>
00019
00023 namespace KEYS{
00024     enum {
00025         ESCAPE = 0x1B,
00026         ENTER = 0xD,
00027         BS = 0x8,
00028         Shift_A = 0x41,
00029         CTRL = 0x11,
00030         DELETE = 0x7f,
00031     };
00032 }
00033
00034
00038 class Keyboard{
00039     public:
00045         Keyboard(sf::RenderWindow* win, sf::Vector2f bounds);
00046
00047
00052         bool isKeyPressed(sf::Keyboard::Key);
00053
00058         bool isTextEntered();
00059
00064         bool isCmdTextEntered();
00065
00070         bool isTextDeleted();
00071
00076         std::string getTextEntered();
00077
00078
00083         std::string getCmdTextEntered();
00084
00085
00090         void setTextEntered(std::string);
00091
00092
00097         void setCmdTextEntered(std::string);
00098
00099
00104         sf::Vector2f getBounds() const;
00105
00109         void backSpace();
00110
00115         void handleKeyEvent(sf::Event& event);
00116
00117
00122         void handleCmdKeyEvent();
00123
00128         void handleMouseEvent(sf::Event& event); // not implemented yet
00129
00130     private:
00131         sf::RenderWindow* window;
00132         sf::Vector2f bounds;
00133         std::string tEntered;
00134         std::string tDeleted;
00136         std::string ctEntered;
00137         std::string ctDeleted;
00138 };
00139 #endif // KAMIL_KEYBOARD_H

```

## 7.11 include/Kamil/MyRect.h File Reference

Interface file for the [MyRect](#) class.

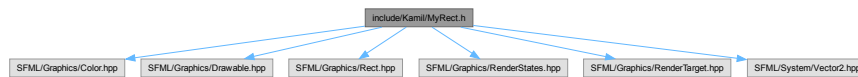
```

#include <SFML/Graphics/Color.hpp>
#include <SFML/Graphics/Drawable.hpp>
#include <SFML/Graphics/Rect.hpp>

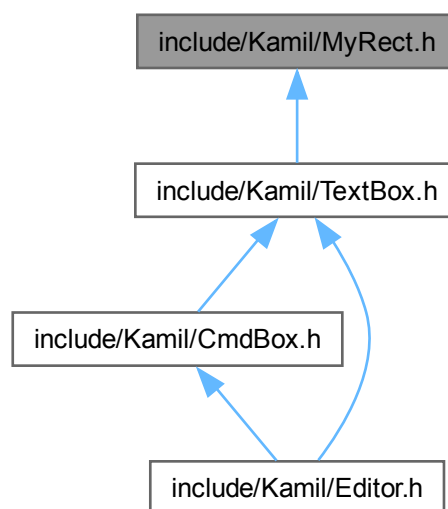
```

```
#include <SFML/Graphics/RenderStates.hpp>
#include <SFML/Graphics/RenderTarget.hpp>
#include <SFML/System/Vector2.hpp>
```

Include dependency graph for MyRect.h:



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



## Classes

- class [MyRect](#)  
*gives extra functionality to FloatRect*

### 7.11.1 Detailed Description

Interface file for the [MyRect](#) class.

Inherits from `sf::FloatRect` and `sf::Drawable`. `sf::FloatRect` is a templated class of `sf::Rect<float>` and its primary use is for defining the border and creating a hollow rectangle, as such it only has methods for collision detection and intersections. The normal `RectangleShape` class creates a basic rectangle without the collision and intersections checking so we inherit this functionality from `FloatRect` and in effect add it to the instantiated `RectangleShape` in the [MyRect](#) class.

The `sf::Drawable` is only here to add a draw property to our class so when we draw to the `RenderTarget`, in this case `RenderWindow`, we can use the same code of `window.draw(our_own_object)` instead of the general `our_own_object.draw(window)`. This is done so when others use this code it makes it easier for them to follow a standard way of drawing to the `RenderTarget` and not having to worry about passing parameters into the objects.

## 7.12 MyRect.h

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

```

00001 #ifndef KAMIL_MYRECT_H
00002 #define KAMIL_MYRECT_H
00003
00004
00025 #include <SFML/Graphics/Color.hpp>
00026 #include <SFML/Graphics/Drawable.hpp>
00027 #include <SFML/Graphics/Rect.hpp>
00028 #include <SFML/Graphics/RenderStates.hpp>
00029 #include <SFML/Graphics/RenderTarget.hpp>
00030 #include <SFML/System/Vector2.hpp>
00031
00032
00039 class MyRect : public sf::FloatRect
00040               , public sf::Drawable
00041 {
00042     public:
00051     MyRect(sf::Vector2f pos, sf::Vector2f size, sf::Color fillColour, sf::Color outlineColour,
float outlineThicknes);
00052
00057     void setPosition(sf::Vector2f pos);
00058
00063     sf::Vector2f getPos() const;
00064
00069     void setFillColour(sf::Color colour);
00070
00075     void setSize(sf::Vector2f size);
00076
00081     sf::Vector2f getSize() const;
00082
00093     void draw(sf::RenderTarget& target, sf::RenderStates states) const override;
00094
00095     protected:
00096         sf::FloatRect fRect;
00097         sf::Vector2f pos;
00098         sf::Vector2f size;
00099         sf::Color fillColour;
00100         sf::Color outlineColour;
00101         float outlineThicknes;
00103 };
00104
00105 #endif // KAMIL_MYRECT_H
00106

```

## 7.13 include/Kamil/TextBox.h File Reference

```

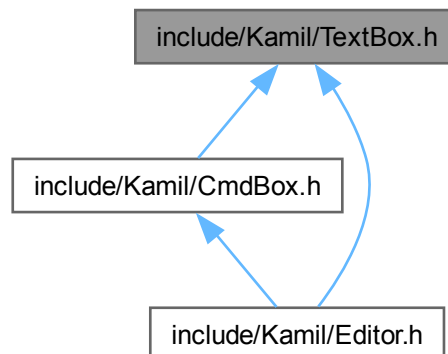
#include <SFML/Graphics.hpp>
#include <SFML/Graphics/Color.hpp>
#include <SFML/Graphics/Drawable.hpp>
#include <SFML/Graphics/Font.hpp>
#include <SFML/Graphics/RectangleShape.hpp>
#include <SFML/Graphics/RenderStates.hpp>
#include <SFML/Graphics/RenderTarget.hpp>
#include <SFML/Graphics/RenderWindow.hpp>
#include <SFML/Graphics/Text.hpp>
#include <SFML/System/Vector2.hpp>
#include <SFML/Window/Keyboard.hpp>
#include <iostream>
#include "Keyboard.h"
#include "MyRect.h"

```

Include dependency graph for TextBox.h:



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



## Classes

- class [TextBox](#)

*A class that makes a Textbox in SFML.*

## 7.14 TextBox.h

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

```

00001 #ifndef KAMIL_TEXTBOX_HPP
00002 #define KAMIL_TEXTBOX_HPP
00003
00014 #include <SFML/Graphics.hpp>
00015 #include <SFML/Graphics/Color.hpp>
00016 #include <SFML/Graphics/Drawable.hpp>
00017 #include <SFML/Graphics/Font.hpp>
00018 #include <SFML/Graphics/RectangleShape.hpp>
00019 #include <SFML/Graphics/RenderStates.hpp>
00020 #include <SFML/Graphics/RenderTarget.hpp>
00021 #include <SFML/Graphics/RenderWindow.hpp>
00022 #include <SFML/Graphics/Text.hpp>
00023 #include <SFML/System/Vector2.hpp>
00024 #include <SFML/Window/Keyboard.hpp>
00025 #include <iostream>
00026
00027 #include "Keyboard.h"
00028 #include "MyRect.h"
00029
00030
00031 /*
00032  *
00033  * TODO:
00034  *     Make a RectangleShape that acts as the bounds of the TextBox
00035  *     then add limits to the textbox so it stays in the limits
00036  *
00037  *     Add the Keyboard manager class here and use its methods
00038  *     to handle the key events
00039  */
00040
00041
00048 class TextBox : public MyRect
00049 {
00050     public:
00062     TextBox(sf::RenderWindow* win, sf::Vector2f pos, sf::Vector2f size, std::string sfont, int
fsize, sf::Color fcol, sf::Color background, float thicc);
00063
  
```

```

00068         void setTextSize(int size);
00069
00074         int getTextSize() const;
00075
00080         void setTextColor(sf::Color colour);
00081
00086         sf::Color getTextColor() const;
00087
00092         void setFont(sf::Font& font);
00093
00099         sf::Text getTextBox() const;
00100
00104         void deleteChar();
00105
00109         void enterPress();
00110
00115         void setString(std::string nstring);
00116
00121         std::string getString() const;
00122
00130         void draw(sf::RenderTarget& target, sf::RenderStates states) const override;
00131
00136         bool isMouseHover();
00137
00138     private:
00139         sf::RenderWindow* window;
00140         sf::Text tbox{};
00141         sf::Font font{};
00142         std::string fname{};
00143         int fsize{};
00144         sf::Color fcol{};
00145         bool mouseHover;
00146 };
00147 #endif // KAMIL_TEXTBOX_HPP

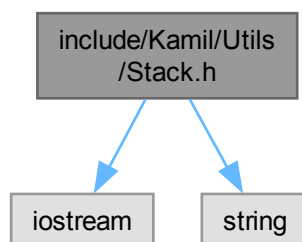
```

## 7.15 include/Kamil/Utils/Stack.h File Reference

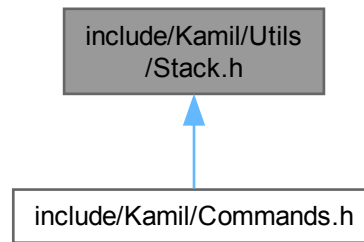
```
#include <iostream>
```

```
#include <string>
```

Include dependency graph for Stack.h:



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



## Classes

- class [Command::Stack< T >](#)

## Namespaces

- namespace [Command](#)  
*A stack in the [Command](#) namespace.*

## 7.16 Stack.h

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

```

00001 #ifndef KAMIL_STACK_H
00002 #define KAMIL_STACK_H
00003
00004 #include <iostream>
00005 #include <string>
00006
00010 namespace Command{
00011
00012 // Dynamic stack array
00013 template<typename T>
00014 class Stack{
00015 public:
00016     Stack(int);
00017     int getMax()const;
00018     void printStack()const;
00019     int pop();
00020     int push(T);
00021     void extend(int);
00022 private:
00023     int max_size{};
00024     T* stack_array{new T[max_size]};
00025     T* SP = &stack_array[max_size];
00026     int SP_pos = max_size;
00027 };
00028
00029 #endif
00030
00031 } // Command
  
```



# Index

- ~Editor
  - Editor, [20](#)
- absPath
  - Document, [18](#)
- backSpace
  - Keyboard, [25](#)
- bounds
  - Keyboard, [28](#)
- BS
  - KEYS, [10](#)
- buffInfo
  - Document, [18](#)
- cbox
  - Editor, [21](#)
- CmdBox, [11](#)
  - TextBox, [14](#)
- Command, [9](#)
- Command::Stack< T >, [35](#)
  - extend, [36](#)
  - getMax, [36](#)
  - max\_size, [37](#)
  - pop, [36](#)
  - printStack, [36](#)
  - push, [37](#)
  - SP, [37](#)
  - SP\_pos, [37](#)
  - Stack, [36](#)
  - stack\_array, [37](#)
- createDir
  - Document, [17](#)
- createFile
  - Document, [17](#)
- ctDeleted
  - Keyboard, [28](#)
- ctEntered
  - Keyboard, [28](#)
- CTRL
  - KEYS, [10](#)
- DELETE
  - KEYS, [10](#)
- deleteChar
  - TextBox, [41](#)
- docChanged
  - Document, [18](#)
- Document, [15](#)
  - absPath, [18](#)
  - buffInfo, [18](#)
  - createDir, [17](#)
  - createFile, [17](#)
  - docChanged, [18](#)
  - Document, [16](#)
  - getAbsPath, [17](#)
  - getRelPath, [17](#)
  - hasChanged, [17](#)
  - init, [17](#)
  - readFile, [18](#)
  - relPath, [18](#)
  - saveFile, [18](#)
- draw
  - Editor, [21](#)
  - MyRect, [32](#)
  - TextBox, [41](#)
- Editor, [19](#)
  - ~Editor, [20](#)
  - cbox, [21](#)
  - draw, [21](#)
  - Editor, [20](#)
  - event, [21](#)
  - handleEvent, [21](#)
  - kb, [21](#)
  - textBox, [21](#)
  - window, [22](#)
- ENTER
  - KEYS, [10](#)
- enterPress
  - TextBox, [41](#)
- ESCAPE
  - KEYS, [10](#)
- event
  - Editor, [21](#)
- extend
  - Command::Stack< T >, [36](#)
- fcoll
  - TextBox, [44](#)
- fillColour
  - MyRect, [34](#)
- fname
  - TextBox, [44](#)
- font
  - TextBox, [44](#)
- fRect
  - MyRect, [34](#)
- fsize
  - TextBox, [44](#)

- getAbsPath
  - Document, 17
- getBounds
  - Keyboard, 25
- getCmdTextEntered
  - Keyboard, 25
- getMax
  - Command::Stack< T >, 36
- getPos
  - MyRect, 32
- getRelPath
  - Document, 17
- getSize
  - MyRect, 33
- getString
  - TextBox, 42
- getTextBox
  - TextBox, 42
- getTextColour
  - TextBox, 42
- getTextEntered
  - Keyboard, 25
- getTextSize
  - TextBox, 42
- handleCmdKeyEvent
  - Keyboard, 25
- handleEvent
  - Editor, 21
- handleKeyEvent
  - Keyboard, 26
- handleMouseEvent
  - Keyboard, 26
- hasChanged
  - Document, 17
- include/Kamil/CmdBox.h, 47, 48
- include/Kamil/Commands.h, 48
- include/Kamil/Document.h, 49, 50
- include/Kamil/Editor.h, 50, 51
- include/Kamil/Keyboard.h, 51, 53
- include/Kamil/MyRect.h, 53, 55
- include/Kamil/TextBox.h, 55, 56
- include/Kamil/Utils/Stack.h, 57, 58
- init
  - Document, 17
- isCmdTextEntered
  - Keyboard, 26
- isKeyPressed
  - Keyboard, 26
- isMouseHover
  - TextBox, 43
- isTextDeleted
  - Keyboard, 27
- isTextEntered
  - Keyboard, 27
- kb
  - Editor, 21
- Keyboard, 22
  - backSpace, 25
  - bounds, 28
  - ctDeleted, 28
  - ctEntered, 28
  - getBounds, 25
  - getCmdTextEntered, 25
  - getTextEntered, 25
  - handleCmdKeyEvent, 25
  - handleKeyEvent, 26
  - handleMouseEvent, 26
  - isCmdTextEntered, 26
  - isKeyPressed, 26
  - isTextDeleted, 27
  - isTextEntered, 27
  - Keyboard, 24
  - setCmdTextEntered, 27
  - setTextEntered, 28
  - tDeleted, 28
  - tEntered, 28
  - window, 28
- KEYS, 9
  - BS, 10
  - CTRL, 10
  - DELETE, 10
  - ENTER, 10
  - ESCAPE, 10
  - Shift\_A, 10
- max\_size
  - Command::Stack< T >, 37
- mouseHover
  - TextBox, 45
- MyRect, 29
  - draw, 32
  - fillColour, 34
  - fRect, 34
  - getPos, 32
  - getSize, 33
  - MyRect, 32
  - outlineColour, 34
  - outlineThicknes, 34
  - pos, 34
  - setFillColour, 33
  - setPosition, 33
  - setSize, 34
  - size, 34
- outlineColour
  - MyRect, 34
- outlineThicknes
  - MyRect, 34
- pop
  - Command::Stack< T >, 36
- pos
  - MyRect, 34
- printStack
  - Command::Stack< T >, 36

- push
  - Command::Stack< T >, 37
- readFile
  - Document, 18
- relPath
  - Document, 18
- saveFile
  - Document, 18
- setCmdTextEntered
  - Keyboard, 27
- setFillColour
  - MyRect, 33
- setFont
  - TextBox, 43
- setPosition
  - MyRect, 33
- setSize
  - MyRect, 34
- setString
  - TextBox, 43
- setTextColour
  - TextBox, 44
- setTextEntered
  - Keyboard, 28
- setTextSize
  - TextBox, 44
- Shift\_A
  - KEYS, 10
- size
  - MyRect, 34
- SP
  - Command::Stack< T >, 37
- SP\_pos
  - Command::Stack< T >, 37
- Stack
  - Command::Stack< T >, 36
- stack\_array
  - Command::Stack< T >, 37
- tbox
  - TextBox, 45
- tDeleted
  - Keyboard, 28
- tEntered
  - Keyboard, 28
- TextBox, 38
  - CmdBox, 14
  - deleteChar, 41
  - draw, 41
  - enterPress, 41
  - fcol, 44
  - fname, 44
  - font, 44
  - fsize, 44
  - getString, 42
  - getTextBox, 42
  - getTextColour, 42
  - getTextSize, 42
  - isMouseHover, 43
  - mouseHover, 45
  - setFont, 43
  - setString, 43
  - setTextColour, 44
  - setTextSize, 44
  - tbox, 45
  - TextBox, 40
  - window, 45
- textBox
  - Editor, 21
- window
  - Editor, 22
  - Keyboard, 28
  - TextBox, 45