

KText Editor

Generated by Doxygen 1.9.6

1 Kamil Editor	1
1.1 Analysis	1
1.1.1 Background and Identifying the problem	1
1.1.2 End User needs	2
1.2 Structure	4
2 Namespace Index	5
2.1 Namespace List	5
3 Hierarchical Index	7
3.1 Class Hierarchy	7
4 Class Index	9
4.1 Class List	9
5 File Index	11
5.1 File List	11
6 Namespace Documentation	13
6.1 Command Namespace Reference	13
6.1.1 Detailed Description	13
6.2 KEYS Namespace Reference	13
6.2.1 Detailed Description	13
6.2.2 Enumeration Type Documentation	13
6.2.2.1 anonymous enum	13
7 Class Documentation	15
7.1 CmdBox Class Reference	15
7.1.1 Detailed Description	18
7.1.2 Member Function Documentation	19
7.1.2.1 TextBox() [1/2]	19
7.1.2.2 TextBox() [2/2]	19
7.2 Document Class Reference	19
7.2.1 Detailed Description	21
7.2.2 Constructor & Destructor Documentation	21
7.2.2.1 Document() [1/2]	21
7.2.2.2 Document() [2/2]	21
7.2.3 Member Function Documentation	22
7.2.3.1 createDir()	22
7.2.3.2 createFile()	22
7.2.3.3 docHasText()	22
7.2.3.4 getAbsPath()	22
7.2.3.5 getLineCount()	22
7.2.3.6 getRelPath()	23

7.2.3.7 hasChanged() [1/2]	23
7.2.3.8 hasChanged() [2/2]	23
7.2.3.9 init() [1/2]	23
7.2.3.10 init() [2/2]	24
7.2.3.11 readFile()	24
7.2.3.12 saveFile() [1/2]	24
7.2.3.13 saveFile() [2/2]	24
7.2.3.14 setBuffInfo()	25
7.2.4 Member Data Documentation	25
7.2.4.1 absPath	25
7.2.4.2 buffInfo	25
7.2.4.3 docChanged	25
7.2.4.4 relPath	25
7.3 Editor Class Reference	26
7.3.1 Detailed Description	27
7.3.2 Constructor & Destructor Documentation	27
7.3.2.1 Editor()	27
7.3.2.2 ~Editor()	27
7.3.3 Member Function Documentation	27
7.3.3.1 draw()	27
7.3.3.2 handleEvent()	28
7.3.3.3 makeLineNum()	29
7.3.4 Member Data Documentation	29
7.3.4.1 camera	29
7.3.4.2 cbox	29
7.3.4.3 doc	30
7.3.4.4 event	30
7.3.4.5 kb	30
7.3.4.6 lineBox	30
7.3.4.7 loadFromFile	30
7.3.4.8 textBox	30
7.3.4.9 window	30
7.4 EditorCam Class Reference	31
7.4.1 Constructor & Destructor Documentation	33
7.4.1.1 EditorCam()	33
7.4.2 Member Function Documentation	33
7.4.2.1 draw()	33
7.4.2.2 getBottomLimitPx()	33
7.4.2.3 getLineHeight()	34
7.4.2.4 getRightLimitPx()	34
7.4.2.5 rotateLeft()	34
7.4.2.6 rotateRight()	34

7.4.2.7 scrollDown()	34
7.4.2.8 scrollLeft()	35
7.4.2.9 scrollRight()	35
7.4.2.10 scrollTo()	35
7.4.2.11 scrollUp()	35
7.4.2.12 setCameraBounds()	35
7.4.2.13 zoomIn()	36
7.4.2.14 zoomOut()	36
7.4.3 Member Data Documentation	36
7.4.3.1 bottomLimitPx	36
7.4.3.2 camera	36
7.4.3.3 deltaRotation	36
7.4.3.4 deltaScroll	36
7.4.3.5 deltaZoomIn	36
7.4.3.6 deltaZoomOut	37
7.4.3.7 lineHeight	37
7.4.3.8 marginXOffset	37
7.4.3.9 rightLimitPx	37
7.4.3.10 window	37
7.5 Keyboard Class Reference	38
7.5.1 Detailed Description	39
7.5.2 Constructor & Destructor Documentation	39
7.5.2.1 Keyboard()	39
7.5.3 Member Function Documentation	40
7.5.3.1 backSpace()	40
7.5.3.2 getBounds()	40
7.5.3.3 getCmdTextEntered()	41
7.5.3.4 getLineNumber()	41
7.5.3.5 getTextEntered()	41
7.5.3.6 handleCmdKeyEvent()	42
7.5.3.7 handleKeyEvent()	43
7.5.3.8 handleMouseEvent()	43
7.5.3.9 isCmdTextEntered()	44
7.5.3.10 isKeyPressed()	44
7.5.3.11 isTextDeleted()	45
7.5.3.12 isTextEntered()	46
7.5.3.13 kbrCmd()	46
7.5.3.14 setCmdTextEntered()	46
7.5.3.15 setTextEntered()	46
7.5.4 Member Data Documentation	48
7.5.4.1 bounds	48
7.5.4.2 ctDeleted	48

7.5.4.3 ctEntered	48
7.5.4.4 tDeleted	48
7.5.4.5 tEntered	49
7.5.4.6 window	49
7.6 MyRect Class Reference	49
7.6.1 Detailed Description	52
7.6.2 Constructor & Destructor Documentation	52
7.6.2.1 MyRect() [1/2]	52
7.6.2.2 MyRect() [2/2]	52
7.6.3 Member Function Documentation	52
7.6.3.1 draw()	52
7.6.3.2 getPos()	53
7.6.3.3 getSize()	53
7.6.3.4 setFillColour()	53
7.6.3.5 setPosition()	54
7.6.3.6 setSize()	54
7.6.4 Member Data Documentation	54
7.6.4.1 fillColour	54
7.6.4.2 fRect	54
7.6.4.3 outlineColour	55
7.6.4.4 outlineThicknes	55
7.6.4.5 pos	55
7.6.4.6 size	55
7.7 Command::Stack< T > Class Template Reference	55
7.7.1 Constructor & Destructor Documentation	56
7.7.1.1 Stack()	57
7.7.2 Member Function Documentation	57
7.7.2.1 extend()	57
7.7.2.2 getMax()	57
7.7.2.3 pop()	57
7.7.2.4 printStack()	57
7.7.2.5 push() [1/2]	57
7.7.2.6 push() [2/2]	58
7.7.3 Member Data Documentation	58
7.7.3.1 max_size	58
7.7.3.2 SP	58
7.7.3.3 SP_pos	58
7.7.3.4 stack_array	58
7.8 TextBox Class Reference	59
7.8.1 Detailed Description	61
7.8.2 Constructor & Destructor Documentation	61
7.8.2.1 TextBox() [1/2]	62

7.8.2.2 <code>TextBox()</code> [2/2]	62
7.8.3 Member Function Documentation	63
7.8.3.1 <code>deleteChar()</code>	63
7.8.3.2 <code>draw()</code>	63
7.8.3.3 <code>enterPress()</code>	63
7.8.3.4 <code>getString()</code>	63
7.8.3.5 <code>getTextBox()</code>	64
7.8.3.6 <code>getTextColour()</code>	64
7.8.3.7 <code>getTextSize()</code>	64
7.8.3.8 <code>isMouseHover()</code>	64
7.8.3.9 <code>setFont()</code>	64
7.8.3.10 <code>setString()</code>	65
7.8.3.11 <code>setTextColour()</code>	65
7.8.3.12 <code>setTextSize()</code>	66
7.8.4 Member Data Documentation	66
7.8.4.1 <code>fcol</code>	66
7.8.4.2 <code>fname</code>	66
7.8.4.3 <code>font</code>	66
7.8.4.4 <code>fsize</code>	66
7.8.4.5 <code>mouseHover</code>	66
7.8.4.6 <code>tbox</code>	67
7.8.4.7 <code>window</code>	67
8 File Documentation	69
8.1 <code>include/Kamil/CmdBox.h</code> File Reference	69
8.2 <code>CmdBox.h</code>	70
8.3 <code>include/Kamil/Commands.h</code> File Reference	70
8.4 <code>Commands.h</code>	70
8.5 <code>include/Kamil/Document.h</code> File Reference	71
8.5.1 Detailed Description	72
8.6 <code>Document.h</code>	72
8.7 <code>include/Kamil/Editor.h</code> File Reference	73
8.7.1 Detailed Description	74
8.8 <code>Editor.h</code>	74
8.9 <code>include/Kamil/EditorCam.h</code> File Reference	75
8.10 <code>EditorCam.h</code>	76
8.11 <code>include/Kamil/Keyboard.h</code> File Reference	76
8.11.1 Detailed Description	77
8.12 <code>Keyboard.h</code>	78
8.13 <code>include/Kamil/MyRect.h</code> File Reference	79
8.13.1 Detailed Description	80
8.14 <code>MyRect.h</code>	80

8.15 include/Kamil/TextBox.h File Reference	80
8.16 TextBox.h	81
8.17 include/Kamil/Utils/Stack.h File Reference	82
8.18 Stack.h	83
8.19 README.md File Reference	84
8.20 src/Document.cpp File Reference	84
8.21 src/Editor.cpp File Reference	84
8.22 src/EditorCam.cpp File Reference	85
8.23 src/kamil.cpp File Reference	85
8.23.1 Function Documentation	86
8.23.1.1 main()	86
8.24 src/Keyboard.cpp File Reference	86
8.25 src/MyRect.cpp File Reference	87
8.26 src/TextBox.cpp File Reference	87
8.27 src/Utils/Stack.cpp File Reference	87
8.28 src/Utils/tet.cpp File Reference	88
8.28.1 Function Documentation	89
8.28.1.1 main()	89
Index	91

Chapter 1

Kamil Editor

A Text [Editor](#) for kamil

1.1 Analysis

1.1.1 Background and Identifying the problem

The Project I will be developing will be in answer to the challenge set out by the end user and friend of mine, Kamil. He challenged me to make a light weight editor that he can use in his day to day life and when doing python projects.

This leads me on to identifying the base problems at hand and summing them into 3 questions:

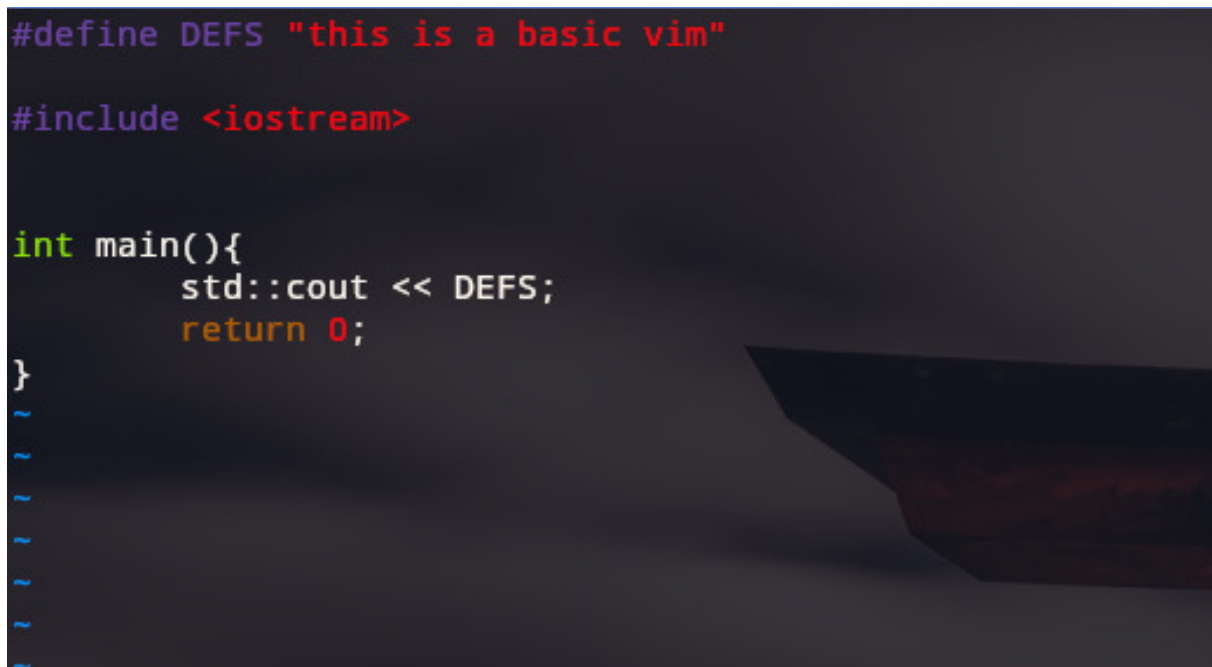
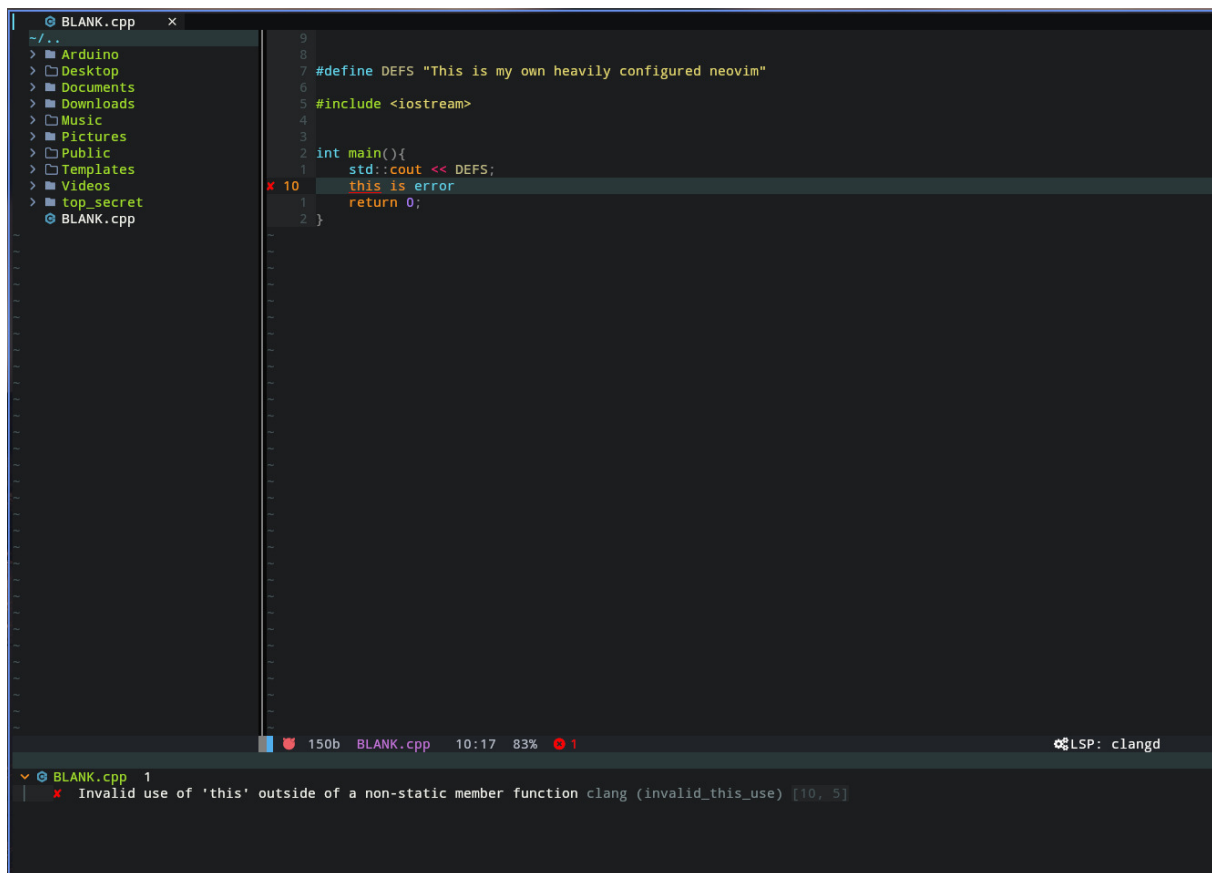
- 1) What is a text editor and how does it differ from an IDE?
- 2) How do I make a text editor or IDE for kamil
- 3) How do I make it efficient enough to meet his standards?

To kick things along I began to do research on Text editors and IDE's and found out that the difference between isn't limited to Operating System platforms or by how much better one is at a specific task but by the features each can do. Text Editors, as the name suggest are specifically desinged for manipulating any form of text that it can open. While an IDE (Integrated Development Environment) is specifically desinged for software development and comes with a multitude of features that engineers can make use of to streemline their workflow.

A table of pros and cons:

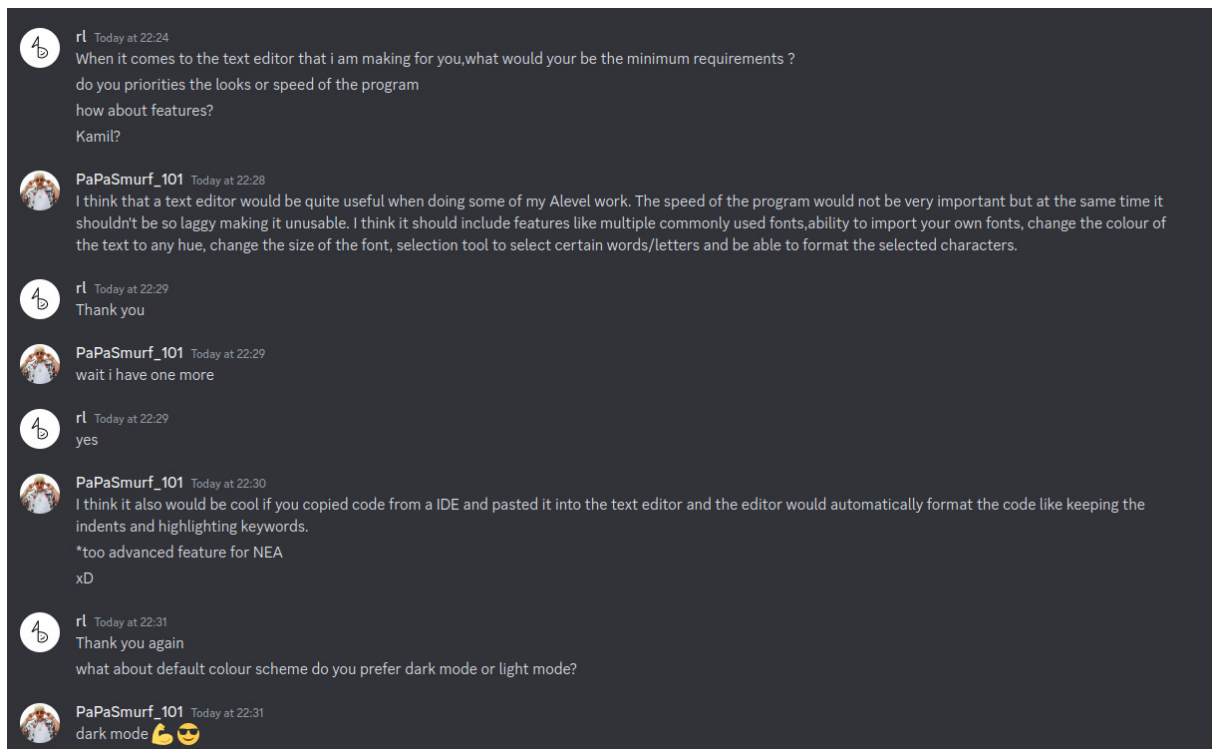
	Pros	cons
Text Editor	Light weight,	Limited in capability
	Fast,	
	Resource efficient,	
	Very Modular	
IDE	Has everything out	Slow
	the box,	Not very Resource efficient
	Modular	

Here are pictures of some text editors and IDE's:



1.1.2 End User needs

When talking to Kamil about his needs it was apparant that he wanted something modular in the sense that it comes with what he needs so its not a hassle to work with and it works with multiple different file types.



To conclude the User needs: Kamil would like:

- Not laggy
- multiple fonts
- import own fonts
- change colour of text
- change size of font
- select and format characters

Extra Features:

- Zoom in / Out
- Scroll up and down
- change background colour
- change text colour
- (potentially) load default colourScheme
- Handle commands such as cmd + s to save etc
- Use arrow keys and H,J,K,L to move through the text
- Use mouse position to place cursor in text
- select text using mouse
- Save files
- Load files
- create directory tree
- traverse directory
- handled in .txt format

1.2 Structure

Kamil.cpp - Main window (text window is used) -Window (has the text box window and the line number) -inputs (handles events like keyboard clicks and mouse events, state changes happen here) -textdoc (handles the file like saving it)

window

- Main Window [Editor](#)
- Where text is handled (same size as window)
- Rectangle on left for margin
- font
- line info
- drawn on main window Input
- mouse and keyboard window TextDocument

– Disliked buttons no button in mine – Dislike having to hunt for library to get stuff done – doesnt like writing code - no response – maybe, stong yes

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Command	A stack in the Command namespace	13
KEYS	An enum for Keyboard characters in hex form	13

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Document	19
sf::Drawable	
EditorCam	31
MyRect	49
TextBox	59
CmdBox	15
Editor	26
sf::FloatRect	
MyRect	49
Keyboard	38
Command::Stack< T >	55

Chapter 4

Class Index

4.1 Class List

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

CmdBox		
	Class to handle the command TextBox	15
Document		
	Document class	19
Editor		
	Class that handles and draws everything in the Editor	26
EditorCam		31
Keyboard		
	A class to handle Keyboard input	38
MyRect		
	Gives extra functionality to FloatRect	49
Command::Stack< T >		55
TextBox		
	A class that makes a Textbox in SFML	59

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

include/Kamil/CmdBox.h	69
include/Kamil/Commands.h	70
include/Kamil/Document.h	
Interface file for the Document class	71
include/Kamil/Editor.h	
Interface file for the Editor class	73
include/Kamil/EditorCam.h	75
include/Kamil/Keyboard.h	
Interface file for Keyboard.h	76
include/Kamil/MyRect.h	
Interface file for the MyRect class	79
include/Kamil/TextBox.h	80
include/Kamil/Utils/Stack.h	82
src/Document.cpp	84
src/Editor.cpp	84
src/EditorCam.cpp	85
src/kamil.cpp	85
src/Keyboard.cpp	86
src/MyRect.cpp	87
src/TextBox.cpp	87
src/Utils/Stack.cpp	87
src/Utils/tet.cpp	88

Chapter 6

Namespace Documentation

6.1 Command Namespace Reference

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

Classes

- class [Stack](#)

6.1.1 Detailed Description

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

6.2 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 }

6.2.1 Detailed Description

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

6.2.2 Enumeration Type Documentation

6.2.2.1 anonymous enum

anonymous enum

Enumerator

ESCAPE	
ENTER	
BS	
Shift_A	
CTRL	
DELETE	

Chapter 7

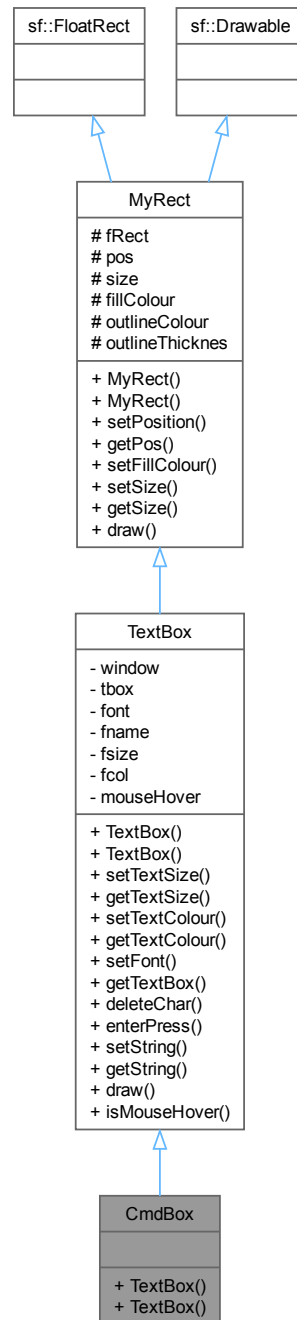
Class Documentation

7.1 CmdBox Class Reference

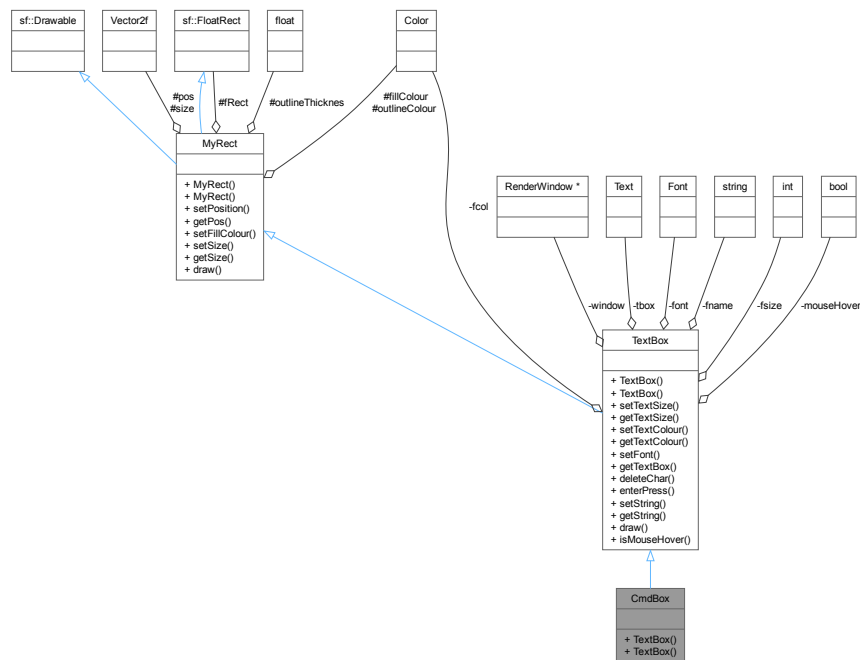
Class to handle the command [TextBox](#).

```
#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.
- [TextBox](#) ()
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](#).
- [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 that's overridden 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`
- `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

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`

7.1.1 Detailed Description

Class to handle the command `TextBox`.

7.1.2 Member Function Documentation

7.1.2.1 `TextBox()` [1/2]

```
TextBox::TextBox ( )
```

Using teh Parent class constructor.

7.1.2.2 `TextBox()` [2/2]

```
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:

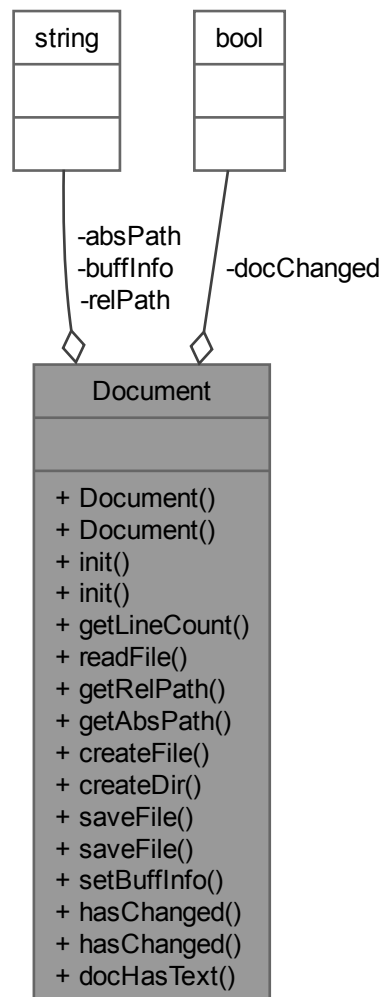
- [include/Kamil/CmdBox.h](#)

7.2 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 [init](#) (std::string inF)
- int [getLineCount](#) ()
- std::string [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 [saveFile](#) ()
- void [setBuffInfo](#) (std::string info)
- bool [hasChanged](#) ()
if the file has changed
- bool [hasChanged](#) (bool val)
- bool [docHasText](#) ()

Private Attributes

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

7.2.1 Detailed Description

[Document](#) class.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 Document() [1/2]

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

Constructor for [Document](#) class.

7.2.2.2 Document() [2/2]

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

Constructor for [Document](#) class.

Parameters

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

7.2.3 Member Function Documentation

7.2.3.1 createDir()

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

create a directory

7.2.3.2 createFile()

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

create the file

7.2.3.3 docHasText()

```
bool Document::docHasText ( )
```

7.2.3.4 getAbsPath()

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

get the relative path

7.2.3.5 getLineCount()

```
int Document::getLineCount ( )
```

7.2.3.6 getRelPath()

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

get the relative path

7.2.3.7 hasChanged() [1/2]

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

if the file has changed

Returns

bool - true if file has changed

Here is the caller graph for this function:



7.2.3.8 hasChanged() [2/2]

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

7.2.3.9 init() [1/2]

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

initialise the file TODO: overload this with file path

Here is the caller graph for this function:



7.2.3.10 init() [2/2]

```
void Document::init (
    std::string inF )
```

7.2.3.11 readFile()

```
std::string Document::readFile ( )
```

read the file

Returns

std::string to the buffer info

Here is the caller graph for this function:



7.2.3.12 saveFile() [1/2]

```
bool Document::saveFile ( )
```

7.2.3.13 saveFile() [2/2]

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

save to a file

Here is the caller graph for this function:



7.2.3.14 setBuffInfo()

```
void Document::setBuffInfo (
    std::string info )
```

Here is the caller graph for this function:



7.2.4 Member Data Documentation

7.2.4.1 absPath

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

absolute path

7.2.4.2 buffInfo

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

7.2.4.3 docChanged

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

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

7.2.4.4 relPath

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

relative path

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

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

7.3.1 Detailed Description

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

7.3.2 Constructor & Destructor Documentation

7.3.2.1 Editor()

```
Editor::Editor (
    sf::RenderWindow * window,
    sf::Event * event,
    Document * doc )
```

Constructor for [Editor](#).

Parameters

<i>window</i>	- pointer to main RenderWindow
<i>event</i>	- pointer to main event
<i>doc</i>	- pointer to document

7.3.2.2 ~Editor()

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

Destructor for [Editor](#) class.

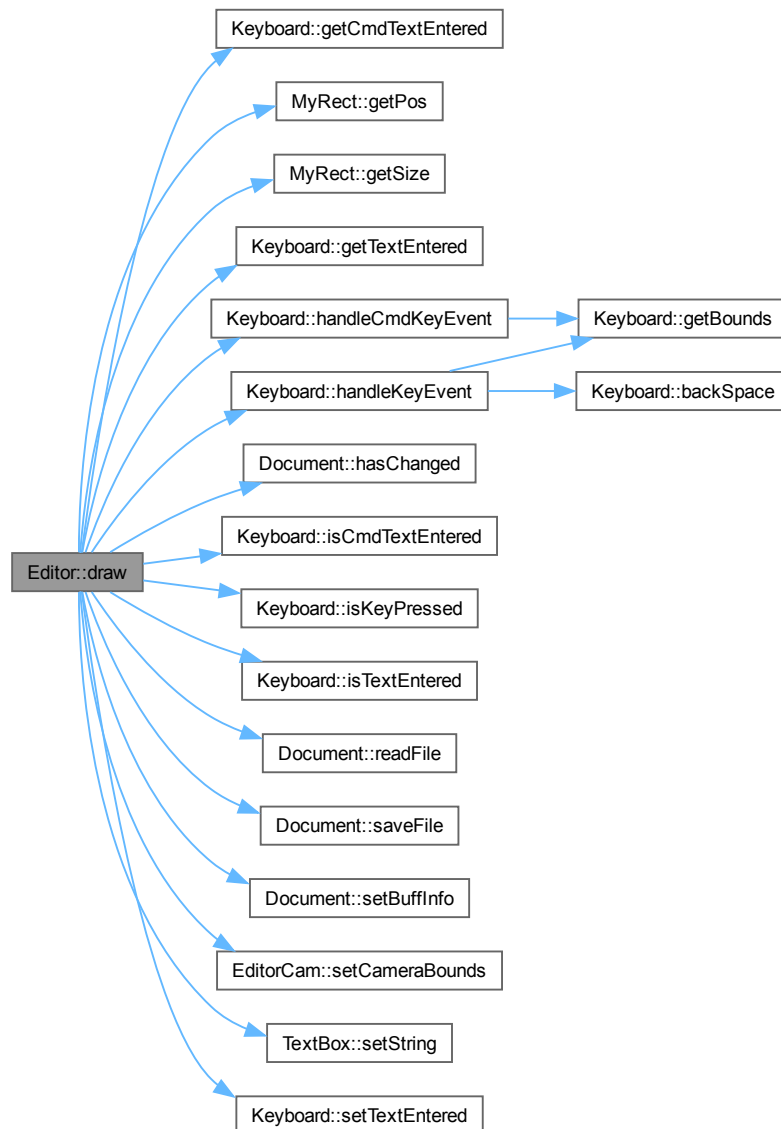
7.3.3 Member Function Documentation

7.3.3.1 draw()

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

function that draws everything to RenderWindow

SOON DEPRECATED Here is the call graph for this function:

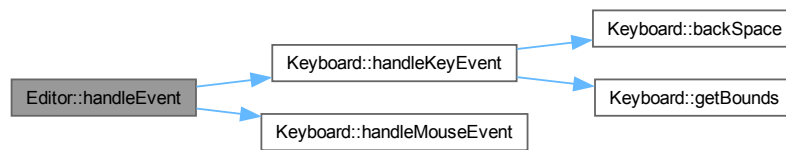


7.3.3.2 `handleEvent()`

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

handle the events for the [Editor](#)

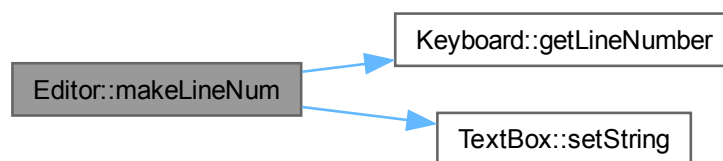
where all event handles are called when interacting with other classes e.g. `kb.handleEvent(); kb.handleMouseEvents();` Here is the call graph for this function:



7.3.3.3 makeLineNum()

```
void Editor::makeLineNum ( )
```

Here is the call graph for this function:



7.3.4 Member Data Documentation

7.3.4.1 camera

```
EditorCam Editor::camera [private]
```

7.3.4.2 cbox

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

reference to command box that we draw

7.3.4.3 doc

```
Document* Editor::doc [private]
```

pointer to the working document

7.3.4.4 event

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

reference to event

7.3.4.5 kb

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

handles keyboard events

7.3.4.6 lineBox

```
TextBox Editor::lineBox [private]
```

7.3.4.7 loadFromFile

```
bool Editor::loadFromFile [private]
```

7.3.4.8 textBox

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

reference to textbox that we draw

7.3.4.9 window

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

reference to RenderWindow

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

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

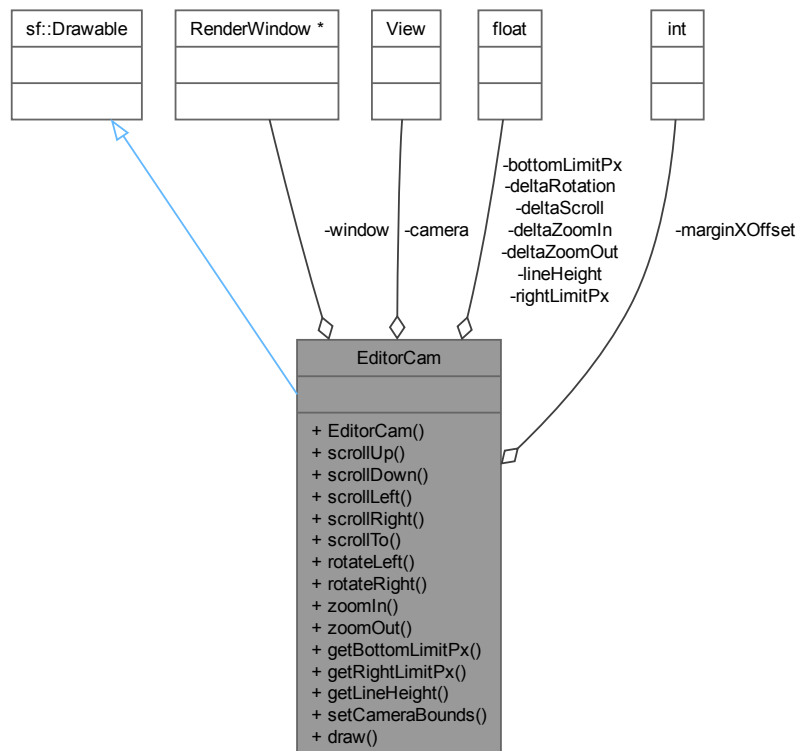
7.4 EditorCam Class Reference

```
#include <EditorCam.h>
```

Inheritance diagram for EditorCam:



Collaboration diagram for EditorCam:



Public Member Functions

- [EditorCam](#) (sf::RenderWindow *[window](#), float [deltaScroll](#), float [deltaRotation](#), float [deltaZoomIn](#), float [deltaZoomOut](#))
- void [scrollUp](#) ()
- void [scrollDown](#) ()
- void [scrollLeft](#) ()
- void [scrollRight](#) ()
- void [scrollTo](#) (float x, float y)
- void [rotateLeft](#) ()
- void [rotateRight](#) ()
- void [zoomIn](#) ()
- void [zoomOut](#) ()
- float [getBottomLimitPx](#) ()
- float [getRightLimitPx](#) ()
- int [getLineHeight](#) ()
- void [setCameraBounds](#) (int width, int height)
- void [draw](#) (sf::RenderTarget &target, sf::RenderStates states) const override

Private Attributes

- sf::RenderWindow * [window](#)
- sf::View [camera](#)

- float [deltaScroll](#)
- float [deltaRotation](#)
- float [deltaZoomIn](#)
- float [deltaZoomOut](#)
- float [rightLimitPx](#)
- float [bottomLimitPx](#)
- float [lineHeight](#)
- int [marginXOffset](#)

7.4.1 Constructor & Destructor Documentation

7.4.1.1 EditorCam()

```
EditorCam::EditorCam (
    sf::RenderWindow * window,
    float deltaScroll,
    float deltaRotation,
    float deltaZoomIn,
    float deltaZoomOut )
```

7.4.2 Member Function Documentation

7.4.2.1 draw()

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

7.4.2.2 getBottomLimitPx()

```
float EditorCam::getBottomLimitPx ( )
```

Here is the caller graph for this function:



7.4.2.3 getLineHeight()

```
int EditorCam::getLineHeight ( )
```

7.4.2.4 getRightLimitPx()

```
float EditorCam::getRightLimitPx ( )
```

Here is the caller graph for this function:



7.4.2.5 rotateLeft()

```
void EditorCam::rotateLeft ( )
```

7.4.2.6 rotateRight()

```
void EditorCam::rotateRight ( )
```

7.4.2.7 scrollDown()

```
void EditorCam::scrollDown ( )
```

Here is the call graph for this function:



7.4.2.8 scrollLeft()

```
void EditorCam::scrollLeft ( )
```

7.4.2.9 scrollRight()

```
void EditorCam::scrollRight ( )
```

Here is the call graph for this function:



7.4.2.10 scrollTo()

```
void EditorCam::scrollTo (
    float x,
    float y )
```

7.4.2.11 scrollUp()

```
void EditorCam::scrollUp ( )
```

7.4.2.12 setCameraBounds()

```
void EditorCam::setCameraBounds (
    int width,
    int height )
```

Here is the caller graph for this function:



7.4.2.13 zoomIn()

```
void EditorCam::zoomIn ( )
```

7.4.2.14 zoomOut()

```
void EditorCam::zoomOut ( )
```

7.4.3 Member Data Documentation

7.4.3.1 bottomLimitPx

```
float EditorCam::bottomLimitPx [private]
```

7.4.3.2 camera

```
sf::View EditorCam::camera [private]
```

handles camera manipulation

7.4.3.3 deltaRotation

```
float EditorCam::deltaRotation [private]
```

delta time for rotation

7.4.3.4 deltaScroll

```
float EditorCam::deltaScroll [private]
```

delta time for scrolling

7.4.3.5 deltaZoomIn

```
float EditorCam::deltaZoomIn [private]
```

7.4.3.6 deltaZoomOut

```
float EditorCam::deltaZoomOut [private]
```

7.4.3.7 lineHeight

```
float EditorCam::lineHeight [private]
```

7.4.3.8 marginXOffset

```
int EditorCam::marginXOffset [private]
```

7.4.3.9 rightLimitPx

```
float EditorCam::rightLimitPx [private]
```

delta time for zoomin/out

7.4.3.10 window

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

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

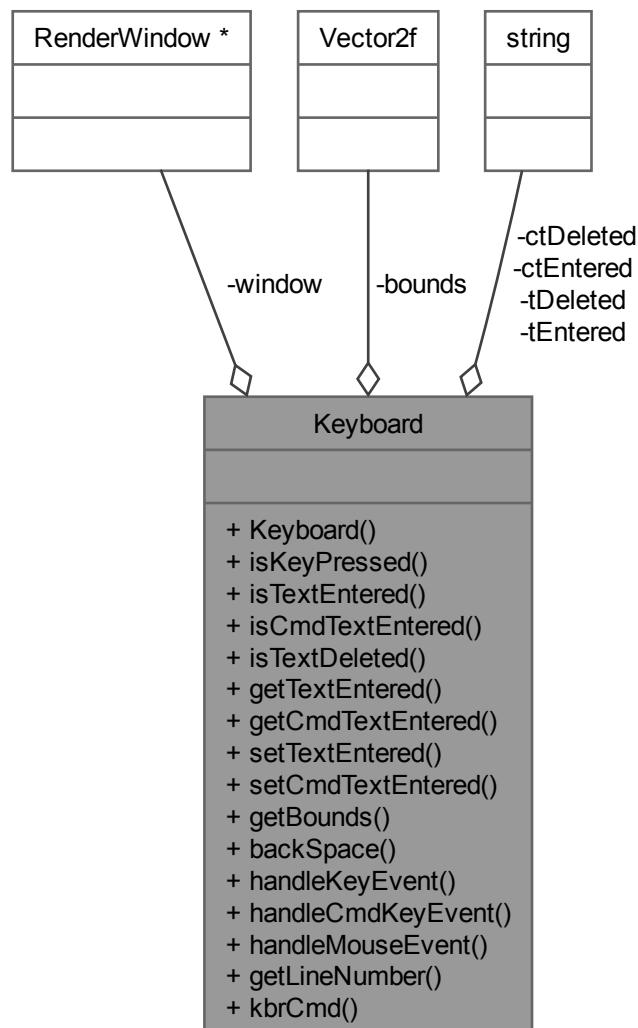
- include/Kamil/[EditorCam.h](#)
- src/[EditorCam.cpp](#)

7.5 Keyboard Class Reference

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

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

Collaboration diagram for Keyboard:



Public Member Functions

- [Keyboard](#) (sf::RenderWindow *win, [Document](#) *doc, 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*
• int `getLineNumber` ()
- template<typename T , size_t N, typename... Args>
void `kbrCmd` (Args... args)

Private Attributes

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

7.5.1 Detailed Description

A class to handle `Keyboard` input.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 Keyboard()

```
Keyboard::Keyboard (
    sf::RenderWindow * win,
    Document * doc,
    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

7.5.3 Member Function Documentation

7.5.3.1 backSpace()

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

when we backspace on teh text

Here is the caller graph for this function:



7.5.3.2 getBounds()

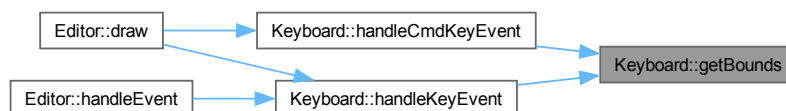
```
sf::Vector2f Keyboard::getBounds ( ) const
```

get the bounds of the area we are in

Returns

`sf::Vector2f` bounded area

Here is the caller graph for this function:



7.5.3.3 getCmdTextEntered()

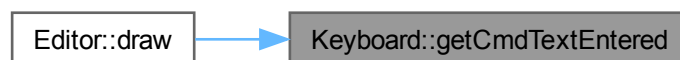
```
std::string Keyboard::getCmdTextEntered ( )
```

returns text entered

Returns

std::string text entered

Here is the caller graph for this function:



7.5.3.4 getLineNumber()

```
int Keyboard::getLineNumber ( )
```

Here is the caller graph for this function:



7.5.3.5 getTextEntered()

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

returns text entered

Returns

std::string text entered

Here is the caller graph for this function:

**7.5.3.6 handleCmdKeyEvent()**

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

handle keyboard events

Parameters

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

Here is the call graph for this function:



Here is the caller graph for this function:



7.5.3.7 handleKeyEvent()

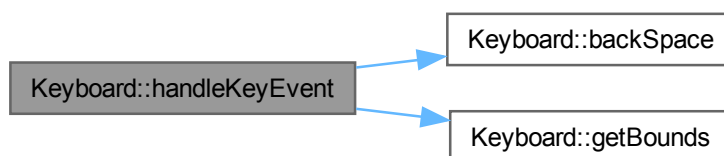
```
void Keyboard::handleKeyEvent (
    sf::Event & event )
```

handle keyboard events

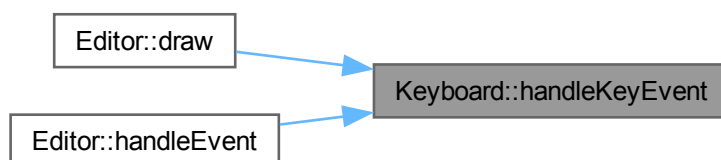
Parameters

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

Here is the call graph for this function:



Here is the caller graph for this function:



7.5.3.8 handleMouseEvent()

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

mouse keyboard events

Parameters

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

Here is the caller graph for this function:

**7.5.3.9 isCmdTextEntered()**

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

checks if text is entered to the command box

Returns

bool true if key is pressed false if not

Here is the caller graph for this function:

**7.5.3.10 isKeyPressed()**

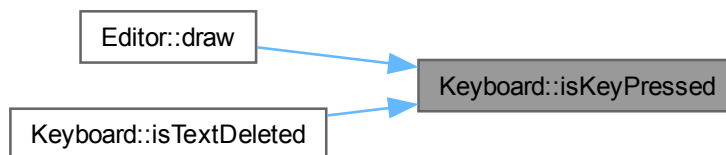
```
bool Keyboard::isKeyPressed (
    sf::Keyboard::Key key )
```

checks if a key is pressed

Returns

bool true if key is pressed false if not

Here is the caller graph for this function:

**7.5.3.11 isTextDeleted()**

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

check if text is being deleted

Returns

bool true if text is being deleted

Here is the call graph for this function:



7.5.3.12 isTextEntered()

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

checks if a text is entered

Returns

bool true if key is pressed false if not

Here is the caller graph for this function:



7.5.3.13 kbrCmd()

```
template<typename T , size_t N, typename... Args>
void Keyboard::kbrCmd (
    Args... args ) [inline]
```

7.5.3.14 setCmdTextEntered()

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

sets text

Parameters

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

7.5.3.15 setTextEntered()

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

sets text

Parameters

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

Here is the caller graph for this function:



7.5.4 Member Data Documentation

7.5.4.1 bounds

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

store the bounded area

7.5.4.2 ctDeleted

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

tmp for text deleted to cmd

7.5.4.3 ctEntered

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

tmp for text entered to cmd

7.5.4.4 tDeleted

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

the text deleted from main box

7.5.4.5 tEntered

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

the text entered to main box

7.5.4.6 window

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

refernce to window

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

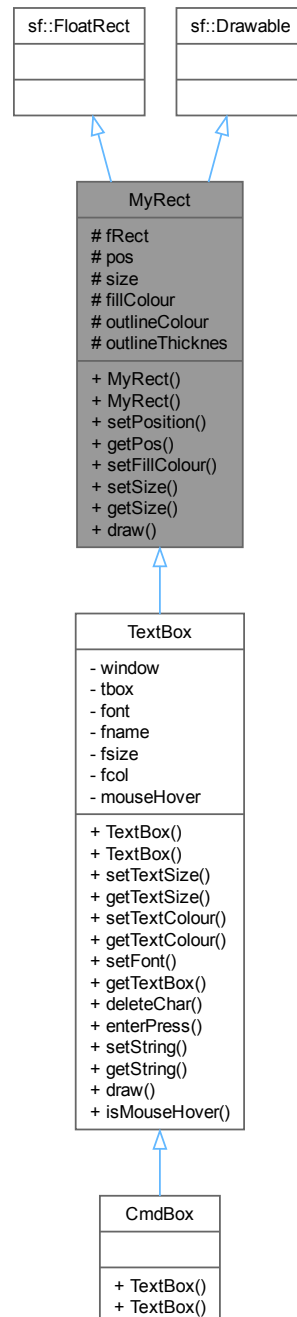
- include/Kamil/[Keyboard.h](#)
- src/[Keyboard.cpp](#)

7.6 MyRect Class Reference

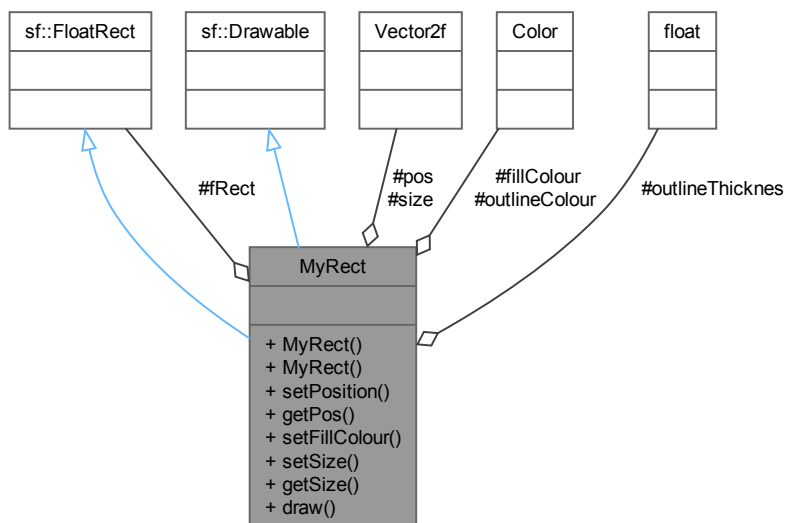
gives extra functionality to FloatRect

```
#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
- **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**

7.6.1 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)

7.6.2 Constructor & Destructor Documentation

7.6.2.1 MyRect() [1/2]

```
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

7.6.2.2 MyRect() [2/2]

```
MyRect::MyRect ( )
```

7.6.3 Member Function Documentation

7.6.3.1 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

7.6.3.2 getPos()

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

get the position of rect

Returns

sf::Vector2f pos

Here is the caller graph for this function:



7.6.3.3 getSize()

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

get the size of the rect

Returns

sf::Vector2f size

Here is the caller graph for this function:



7.6.3.4 setFillColour()

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

set the fill colour of the rect

Parameters

<i>sf::Color</i>	colour
------------------	--------

7.6.3.5 setPosition()

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

sets the position of rect

Parameters

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

7.6.3.6 setSize()

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

set the size of the rect

Parameters

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

7.6.4 Member Data Documentation**7.6.4.1 fillColour**

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

colour of rect

7.6.4.2 fRect

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

for collision checking

7.6.4.3 outlineColour

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

outline colour of rect

7.6.4.4 outlineThicknes

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

outline thickness of rect

7.6.4.5 pos

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

position of rect

7.6.4.6 size

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

size of rect

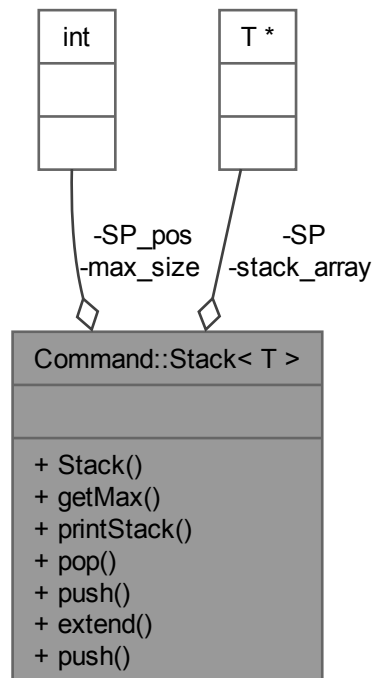
The documentation for this class was generated from the following files:

- include/Kamil/[MyRect.h](#)
- src/[MyRect.cpp](#)

7.7 Command::Stack< T > 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`)
- `int push` (`std::string` value)

Private Attributes

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

7.7.1 Constructor & Destructor Documentation

7.7.1.1 Stack()

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

7.7.2 Member Function Documentation

7.7.2.1 extend()

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

7.7.2.2 getMax()

```
template<typename T >
int Command::Stack< T >::getMax
```

7.7.2.3 pop()

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

7.7.2.4 printStack()

```
template<typename T >
void Command::Stack< T >::printStack
```

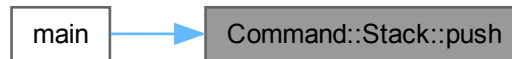
7.7.2.5 push() [1/2]

```
int Command::Stack< std::string >::push (
    std::string value )
```

7.7.2.6 push() [2/2]

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

Here is the caller graph for this function:



7.7.3 Member Data Documentation

7.7.3.1 max_size

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

7.7.3.2 SP

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

7.7.3.3 SP_pos

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

7.7.3.4 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:

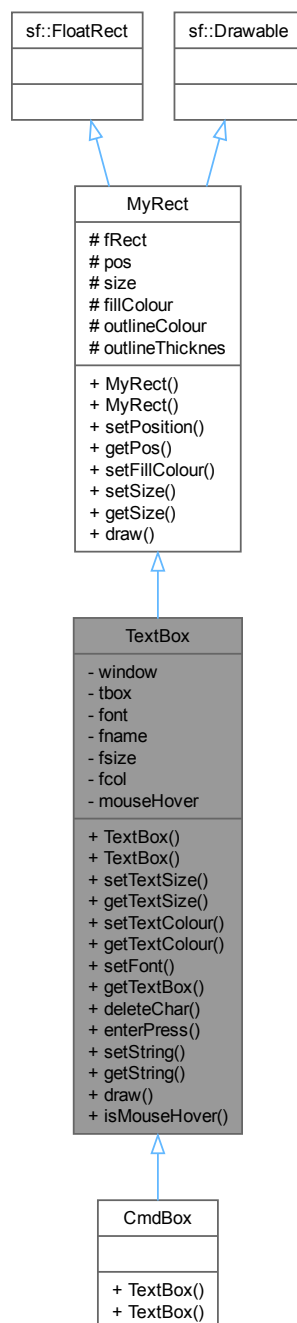
- include/Kamil/Commands.h
- include/Kamil/Utils/Stack.h
- src/Utils/Stack.cpp

7.8 TextBox Class Reference

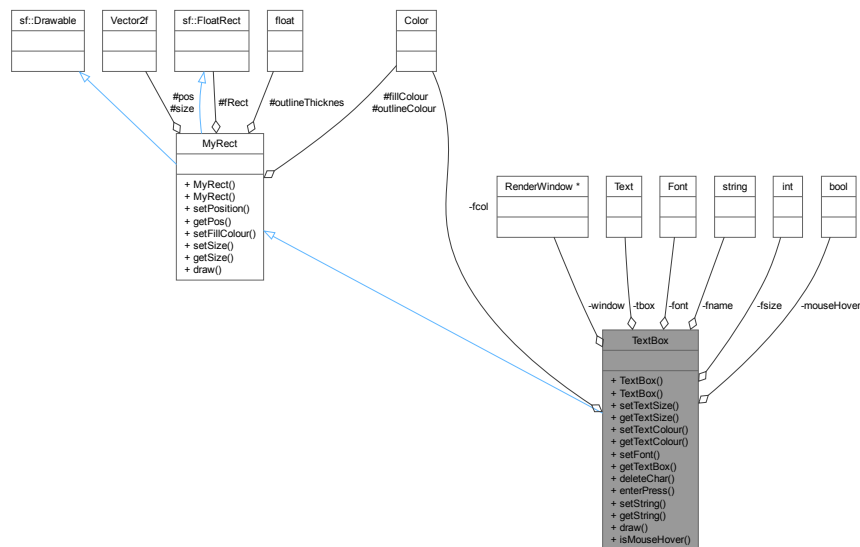
A class that makes a Textbox in SFML.

```
#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**.*
- **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](#)
- [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](#)

7.8.1 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

7.8.2 Constructor & Destructor Documentation

7.8.2.1 TextBox() [1/2]

```

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](#).

Constructor Implementation for [TextBox](#) class.

Parameters

<i>win</i>	- RenderWindow the TextBox is drawn onto
<i>pos</i>	- the initial position of the TextBox
<i>size</i>	- the initial size of the TextBox
<i>sfont</i>	- the initial font used by the TextBox
<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

Implementation of the [TextBox](#) class

Note

other structs or classes may be used here

Parameters

<i>win</i>	- RenderWindow the TextBox is drawn onto
<i>pos</i>	- the initial position of the TextBox
<i>size</i>	- the initial size of the TextBox
<i>sfont</i>	- the initial font used by the TextBox
<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

setting up the text and font

7.8.2.2 TextBox() [2/2]

```

TextBox::TextBox ( )

```

7.8.3 Member Function Documentation

7.8.3.1 deleteChar()

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

Delete last character entered.

7.8.3.2 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

7.8.3.3 enterPress()

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

Handles Enter key press.

7.8.3.4 getString()

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

returns the text in tbox

Returns

type std::string

7.8.3.5 `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

7.8.3.6 `getTextColour()`

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

Get the colour of the text.

Returns

sf::Colour textColour

7.8.3.7 `getTextSize()`

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

Get the size of the text.

Returns

an int of the text size

7.8.3.8 `isMouseHover()`

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

check if mouse is hovering over current textbox

Returns

bool - yes if hovering

7.8.3.9 `setFont()`

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

set what font you use

Parameters

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

7.8.3.10 setString()

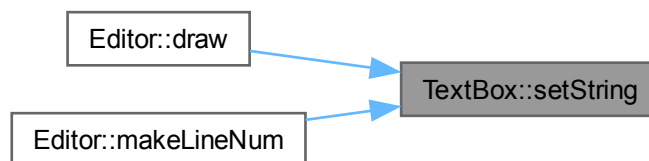
```
void TextBox::setString (
    std::string nstring )
```

Sets the string.

Parameters

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

Here is the caller graph for this function:

**7.8.3.11 setTextColour()**

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

Set the colour of the text.

Parameters

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

7.8.3.12 setTextSize()

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

Set the size of the text.

Parameters

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

7.8.4 Member Data Documentation

7.8.4.1 fcol

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

the font colour

7.8.4.2 fname

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

the name of the font used

7.8.4.3 font

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

the font that the [TextBox](#) uses

7.8.4.4 fsize

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

the font size

7.8.4.5 mouseHover

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

if the mouse is hovering over

7.8.4.6 tbox

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

the text that everything is written onto

7.8.4.7 window

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

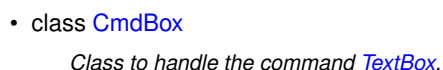
pointer to the main RenderWindow variable

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

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

File Documentation

```
#include "TextBox.h"
// Include dependency graph for CmdBox.h:
```



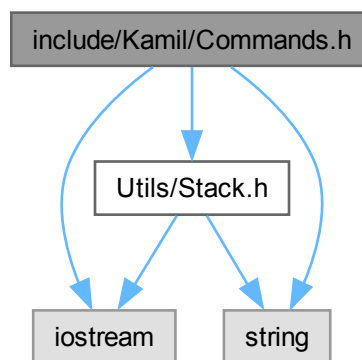
8.2 CmdBox.h

[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
```

8.3 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.

8.4 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

```

8.5 include/Kamil/Document.h File Reference

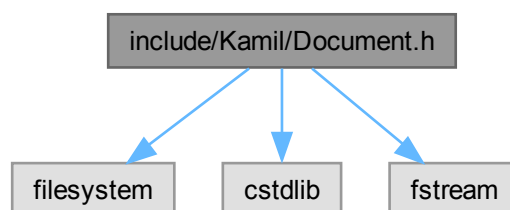
Interface file for the [Document](#) class.

```

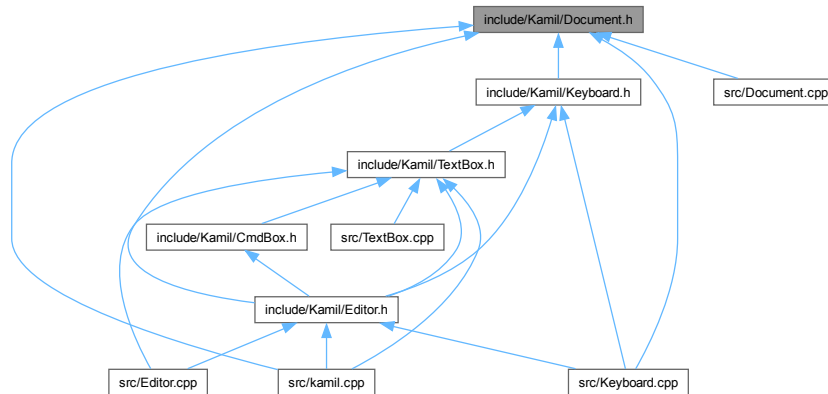
#include <filesystem>
#include <cstdlib>
#include <fstream>

```

Include dependency graph for Document.h:



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



Classes

- class [Document](#)
Document class.

8.5.1 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

8.6 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
00040     void init(std::string inF);
00041
00042
00043     int getLineCount();
00044
00049     std::string readFile();
00050
00054     std::string getRelPath();
00055
00059     std::string getAbsPath();
00060

```



```

00064     void createFile();
00065
00069     void createDir();
00070
00074     bool saveFile(const std::string& filename);
00075
00076     bool saveFile();
00077
00078
00079     void setBuffInfo(std::string info);
00080
00085     bool hasChanged();
00086
00087     bool hasChanged(bool val);
00088
00089     bool docHasText();
00090
00091     // void addTextToPos(std::string txt, int pos);
00092
00093 private:
00094     std::string relPath;
00095     std::string absPath;
00097     std::string buffInfo;
00099     bool docChanged;
00100 };
00101 #endif // KAMIL_DOCUMENT_H

```

8.7 include/Kamil/Editor.h File Reference

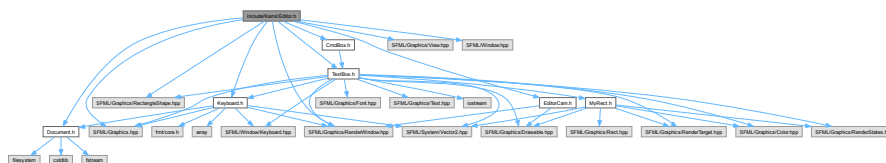
Interface file for the [Editor](#) class.

```

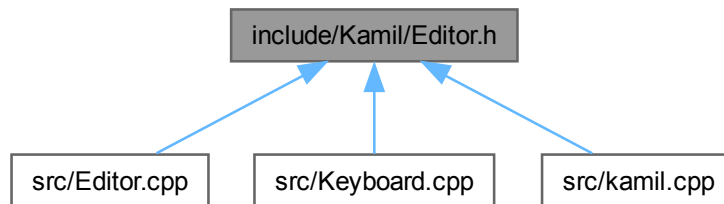
#include <SFML/Graphics.hpp>
#include <SFML/Graphics/RectangleShape.hpp>
#include <SFML/Graphics/RenderWindow.hpp>
#include <SFML/Graphics/View.hpp>
#include <SFML/Window.hpp>
#include "TextBox.h"
#include "Keyboard.h"
#include "CmdBox.h"
#include "Document.h"
#include "EditorCam.h"

```

Include dependency graph for Editor.h:



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



Classes

- class [Editor](#)

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

8.7.1 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

8.8 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/Graphics/View.hpp>
00020 #include <SFML/Window.hpp>
00021
00022 #include "TextBox.h"
00023 #include "Keyboard.h"
00024 #include "CmdBox.h"
00025 #include "Document.h"
00026 #include "EditorCam.h"
00027
00028
00032 class Editor{
00033     public:
00040         Editor(sf::RenderWindow* window, sf::Event* event, Document* doc);
00041
00045         ~Editor();
00046
00052         void draw();
00053
00054         void makeLineNum();
00055
00062         void handleEvent();
00063
00064     private:
00065         Document* doc;
00066         TextBox* textBox;
  
```

```

00067     CmdBox*  cbox;
00068     sf::RenderWindow* window;
00069     sf::Event* event;
00070     TextBox  lineBox;
00071     EditorCam camera;
00072     Keyboard kb;
00073     bool loadFromFile;
00074
00075 };
00076
00077 #endif // KAMIL_EDITOR_WINDOW_HPP

```

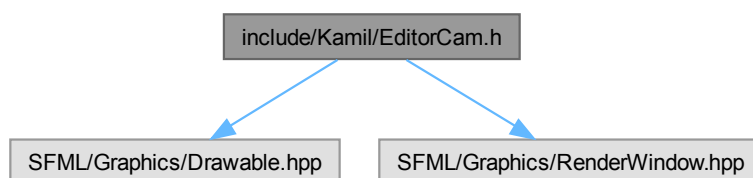
8.9 include/Kamil/EditorCam.h File Reference

```

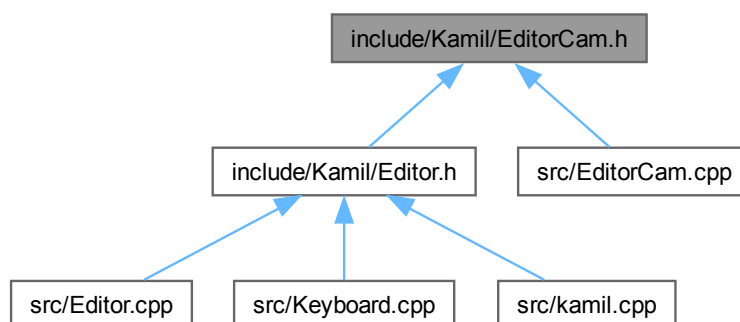
#include <SFML/Graphics/Drawable.hpp>
#include <SFML/Graphics/RenderWindow.hpp>

```

Include dependency graph for EditorCam.h:



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



Classes

- class [EditorCam](#)

8.10 EditorCam.h

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

```

00001 #ifndef KAMIL_EDITOR_CAM_H
00002 #define KAMIL_EDITOR_CAM_H
00003
00004
00005
00006 #include <SFML/Graphics/Drawable.hpp>
00007 #include <SFML/Graphics/RenderWindow.hpp>
00008
00009
00010 class EditorCam : public sf::Drawable
00011 {
00012 public:
00013     EditorCam(sf::RenderWindow* window, float deltaScroll, float deltaRotation, float deltaZoomIn,
00014             float deltaZoomOut);
00015     void scrollUp();
00016     void scrollDown();
00017     void scrollLeft();
00018     void scrollRight();
00019
00020     void scrollTo(float x, float y);
00021
00022     void rotateLeft();
00023     void rotateRight();
00024
00025     void zoomIn();
00026     void zoomOut();
00027
00028     float getBottomLimitPx();
00029     float getRightLimitPx();
00030     int getLineHeight();
00031
00032     void setCameraBounds(int width, int height);
00033
00034     void draw(sf::RenderTarget& target, sf::RenderStates states) const override;
00035 private:
00036     sf::RenderWindow* window;
00037     sf::View camera;
00038     float deltaScroll;
00039     float deltaRotation;
00040     float deltaZoomIn, deltaZoomOut;
00041     float rightLimitPx;
00042     float bottomLimitPx;
00043     float lineHeight;
00044     int marginXOffset;
00045 };
00046
00047 #endif // KAMIL_EDITOR_CAM_H

```

8.11 include/Kamil/Keyboard.h File Reference

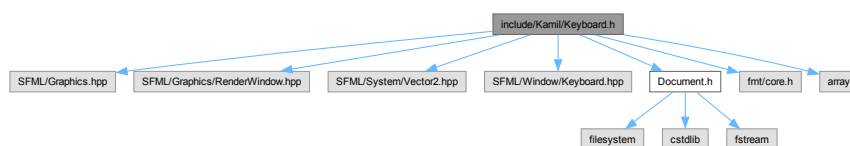
Interface file for [Keyboard.h](#).

```

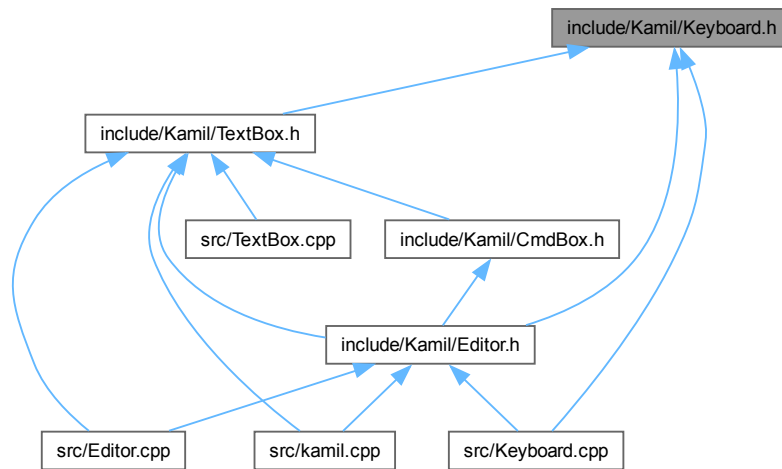
#include <SFML/Graphics.hpp>
#include <SFML/Graphics/RenderWindow.hpp>
#include <SFML/System/Vector2.hpp>
#include <SFML/Window/Keyboard.hpp>
#include "Document.h"
#include <fmt/core.h>
#include <array>

```

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 }

8.11.1 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

8.12 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
00020 #include "Document.h"
00021 #include <fmt/core.h>
00022
00023 #include <array>
00024
00028 namespace KEYS{
00029     enum {
00030         ESCAPE = 0x1B,
00031         ENTER = 0xD,
00032         BS = 0x8,
00033         Shift_A = 0x41,
00034         CTRL = 0x11,
00035         DELETE = 0x7f,
00036     };
00037 }
00038
00039
00040 #ifdef USE_KEYS
00041
00042 #define "LControl" sf::Keyboard::KEYS::LControl
00043
00044 #endif
00045
00046
00047
00051 class Keyboard{
00052     public:
00058         Keyboard(sf::RenderWindow* win, Document* doc, sf::Vector2f bounds);
00059
00060
00065         bool isKeyPressed(sf::Keyboard::Key);
00066
00071         bool isTextEntered();
00072
00077         bool isCmdTextEntered();
00078
00083         bool isTextDeleted();
00084
00089         std::string getTextEntered();
00090
00091
00096         std::string getCmdTextEntered();
00097
00098
00103         void setTextEntered(std::string);
00104
00105
00110         void setCmdTextEntered(std::string);
00111
00112
00117         sf::Vector2f getBounds() const;
00118
00122         void backSpace();
00123
00128         void handleKeyEvent(sf::Event& event);
00129
00130
00135         void handleCmdKeyEvent();
00136
00141         void handleMouseEvent(sf::Event& event); // not implemented yet
00142
00143
00144         int getLineNumber();
00145
00146         template<typename T, size_t N, typename... Args>
00147         void kbrCmd(Args... args){
00148             std::array<T,N> val{args...};
00149             for(const auto& element: val){
00150                 fmt::print("{} ", element);
00151             }
00152         }
00153
00154

```

```

00155 // get position in text
00156
00157     private:
00158         sf::RenderWindow* window;
00159         // Document* doc;
00160         sf::Vector2f bounds;
00161         std::string tEntered;
00162         std::string tDeleted;
00164         std::string ctEntered;
00165         std::string ctDeleted;
00166     };
00167 #endif // KAMIL_KEYBOARD_H

```

8.13 include/Kamil/MyRect.h File Reference

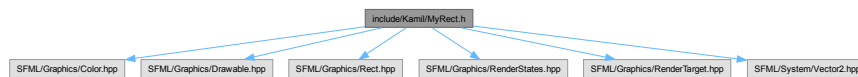
Interface file for the [MyRect](#) class.

```

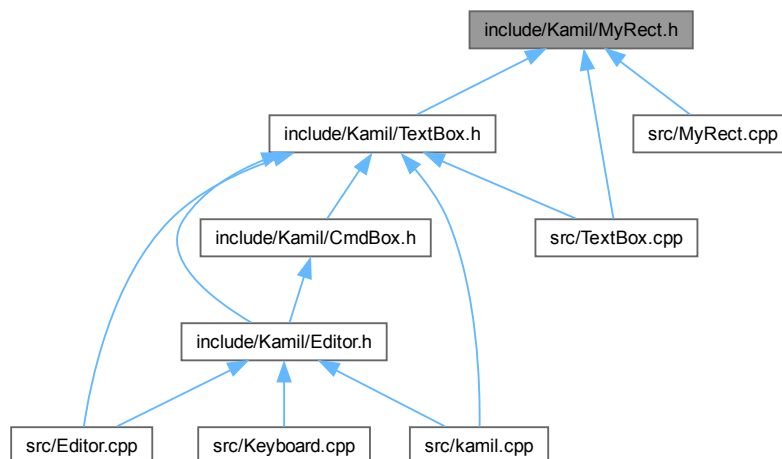
#include <SFML/Graphics/Color.hpp>
#include <SFML/Graphics/Drawable.hpp>
#include <SFML/Graphics/Rect.hpp>
#include <SFML/Graphics/RenderStates.hpp>
#include <SFML/Graphics/RenderTarget.hpp>
#include <SFML/System/Vector2.hpp>

```

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

8.13.1 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.

8.14 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         MyRect();
00053
00058         void setPosition(sf::Vector2f pos);
00059
00064         sf::Vector2f getPos() const;
00065
00070         void setFillColour(sf::Color colour);
00071
00076         void setSize(sf::Vector2f size);
00077
00082         sf::Vector2f getSize() const;
00083
00094         void draw(sf::RenderTarget& target, sf::RenderStates states) const override;
00095
00096     protected:
00097         sf::FloatRect fRect;
00098         sf::Vector2f pos;
00099         sf::Vector2f size;
00100         sf::Color fillColour;
00101         sf::Color outlineColour;
00102         float outlineThicknes;
00104 };
00105
00106 #endif // KAMIL_MYRECT_H
00107
```

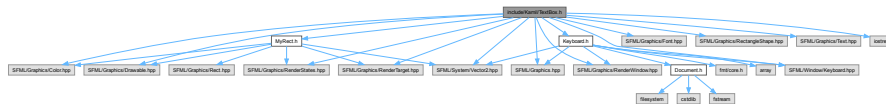
8.15 include/Kamil/TextBox.h File Reference

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

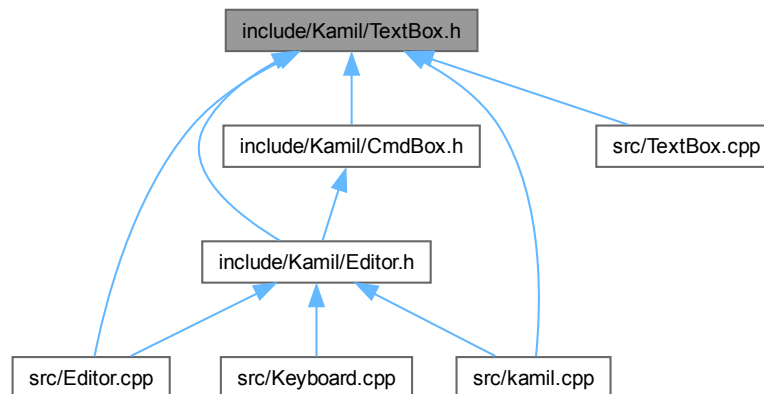


```
#include <SFML/Graphics/Font.hpp>
#include <SFML/Graphics/RectangleShape.hpp>
#include <SFML/Graphics/RenderStates.hpp>
#include <SFML/Graphics/RenderTarget.hpp>
#include <SFML/Graphics/RenderWindow.hpp>
#include <SFML/Graphics/Text.hpp>
#include <SFML/System/Vector2.hpp>
#include <SFML/Window/Keyboard.hpp>
#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.

8.16 TextBox.h

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

```
00001 #ifndef KAMIL_TEXTBOX_HPP
00002 #define KAMIL_TEXTBOX_HPP
00003
00004 #include <SFML/Graphics.hpp>
00005 #include <SFML/Graphics/Color.hpp>
00006 #include <SFML/Graphics/Drawable.hpp>
00007 #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
00042 class TextBox : public MyRect
00043 {
00044     public:
00045         TextBox(sf::RenderWindow* win, sf::Vector2f pos, sf::Vector2f size, std::string sfont, int
fsize, sf::Color fcol, sf::Color background, float thicc);
00046         TextBox();
00047
00048         void setTextSize(int size);
00049
00050         int getTextSize() const;
00051
00052         void setTextColour(sf::Color colour);
00053
00054         sf::Color getTextColour() const;
00055
00056         void setFont(sf::Font& font);
00057
00058         sf::Text getTextBox() const;
00059
00060         void deleteChar();
00061
00062         void enterPress();
00063
00064         void setString(std::string nstring);
00065
00066         std::string getString() const;
00067
00068         void draw(sf::RenderTarget& target, sf::RenderStates states) const override;
00069
00070         bool isMouseHover();
00071
00072     private:
00073         sf::RenderWindow* window;
00074         sf::Text tbox{};
00075         sf::Font font{};
00076         std::string fname{};
00077         int fsize{};
00078         sf::Color fcol{};
00079         bool mouseHover;
00080 };
00081
00082 #endif // KAMIL_TEXTBOX_HPP

```

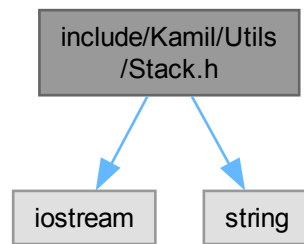
8.17 include/Kamil/Utils/Stack.h File Reference

```

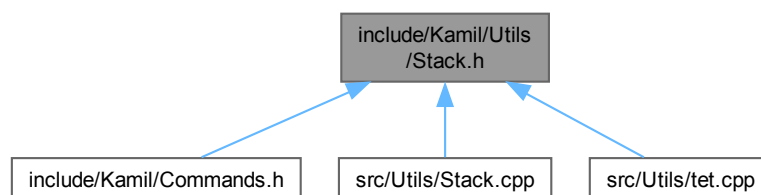
#include <iostream>
#include <string>

```

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.

8.18 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

```

8.19 README.md File Reference

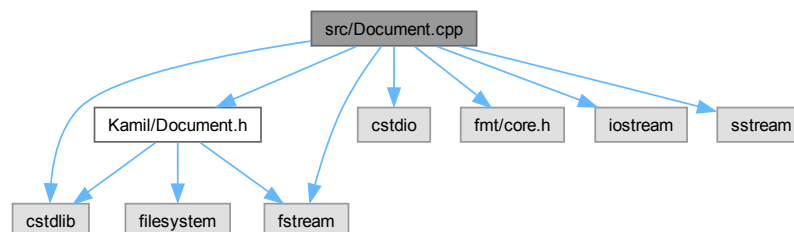
8.20 src/Document.cpp File Reference

```

#include <Kamil/Document.h>
#include <cstdio>
#include <cstdlib>
#include <fmt/core.h>
#include <fstream>
#include <iostream>
#include <sstream>

```

Include dependency graph for Document.cpp:



8.21 src/Editor.cpp File Reference

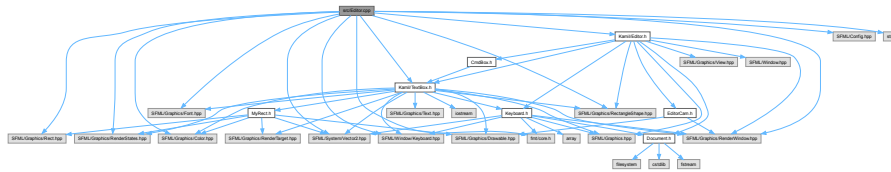
```

#include <Kamil/TextBox.h>
#include <Kamil/Editor.h>
#include <SFML/Config.hpp>
#include <SFML/Graphics/Color.hpp>
#include <SFML/Graphics/Font.hpp>
#include <SFML/Graphics/Rect.hpp>
#include <SFML/Graphics/RectangleShape.hpp>
#include <SFML/Graphics/RenderStates.hpp>
#include <SFML/Graphics/RenderWindow.hpp>
#include <SFML/System/Vector2.hpp>

```

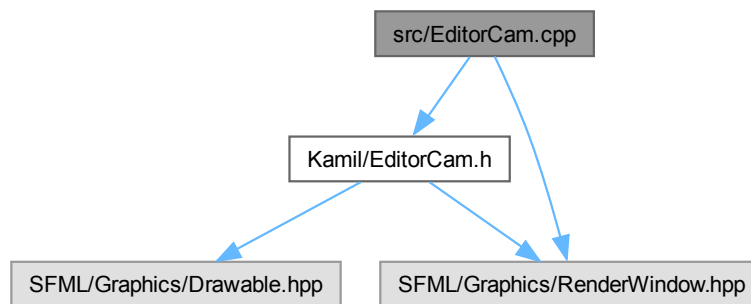
```
#include <SFML/Window/Keyboard.hpp>
#include <fmt/core.h>
#include <string>
```

Include dependency graph for Editor.cpp:



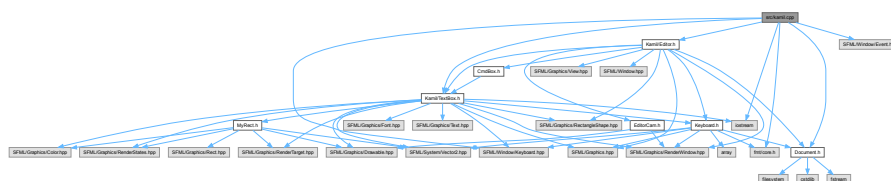
8.22 src/EditorCam.cpp File Reference

```
#include <Kamil/EditorCam.h>
#include <SFML/Graphics/RenderWindow.hpp>
Include dependency graph for EditorCam.cpp:
```



8.23 src/kamil.cpp File Reference

```
#include "Kamil/TextBox.h"
#include <SFML/Window/Event.hpp>
#include <SFML/Window/Keyboard.hpp>
#include <iostream>
#include <Kamil/Editor.h>
#include <fmt/core.h>
#include <Kamil/Document.h>
Include dependency graph for kamil.cpp:
```



Functions

- int `main` (int argc, char *argv[])

8.23.1 Function Documentation

8.23.1.1 `main()`

```
int main (
    int argc,
    char * argv[] )
```

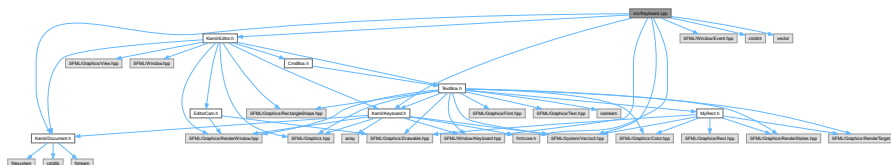
Here is the call graph for this function:



8.24 src/Keyboard.cpp File Reference

```
#include <Kamil/Document.h>
#include <Kamil/Keyboard.h>
#include <Kamil/Editor.h>
#include <SFML/System/Vector2.hpp>
#include <SFML/Window/Event.hpp>
#include <SFML/Window/Keyboard.hpp>
#include <cstdlib>
#include <fmt/core.h>
#include <vector>
```

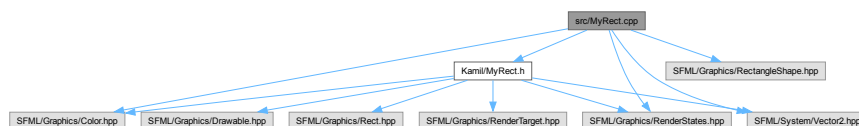
Include dependency graph for Keyboard.cpp:



8.25 src/MyRect.cpp File Reference

```
#include <Kamil/MyRect.h>
#include <SFML/Graphics/Color.hpp>
#include <SFML/Graphics/RectangleShape.hpp>
#include <SFML/Graphics/RenderStates.hpp>
#include <SFML/System/Vector2.hpp>
```

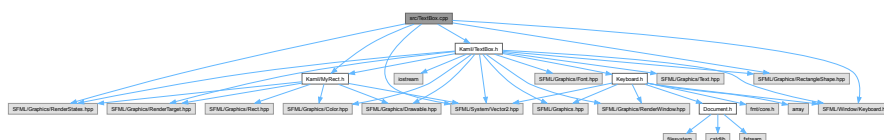
Include dependency graph for MyRect.cpp:



8.26 src/TextBox.cpp File Reference

```
#include <Kamil/MyRect.h>
#include <Kamil/TextBox.h>
#include <SFML/Graphics/RectangleShape.hpp>
#include <SFML/Graphics/RenderStates.hpp>
#include <SFML/System/Vector2.hpp>
#include <SFML/Window/Keyboard.hpp>
```

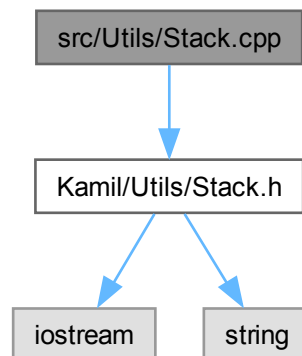
Include dependency graph for TextBox.cpp:



8.27 src/Utils/Stack.cpp File Reference

```
#include <Kamil/Utils/Stack.h>
```

Include dependency graph for Stack.cpp:



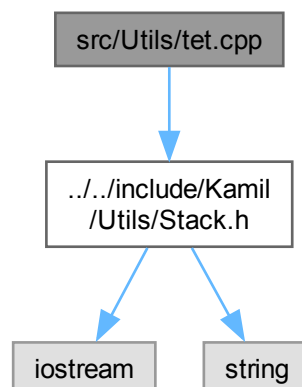
Namespaces

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

8.28 src/Utils/tet.cpp File Reference

```
#include "../..//include/Kamil/Utils/Stack.h"
```

Include dependency graph for tet.cpp:



Functions

- int `main` ()

8.28.1 Function Documentation

8.28.1.1 `main()`

```
int main ( )
```

Here is the call graph for this function:



Index

- ~Editor
 - Editor, [27](#)
- absPath
 - Document, [25](#)
- backSpace
 - Keyboard, [40](#)
- bottomLimitPx
 - EditorCam, [36](#)
- bounds
 - Keyboard, [48](#)
- BS
 - KEYS, [14](#)
- buffInfo
 - Document, [25](#)
- camera
 - Editor, [29](#)
 - EditorCam, [36](#)
- cbox
 - Editor, [29](#)
- CmdBox, [15](#)
 - TextBox, [19](#)
- Command, [13](#)
- Command::Stack< T >, [55](#)
 - extend, [57](#)
 - getMax, [57](#)
 - max_size, [58](#)
 - pop, [57](#)
 - printStack, [57](#)
 - push, [57](#)
 - SP, [58](#)
 - SP_pos, [58](#)
 - Stack, [56](#)
 - stack_array, [58](#)
- createDir
 - Document, [22](#)
- createFile
 - Document, [22](#)
- ctDeleted
 - Keyboard, [48](#)
- ctEntered
 - Keyboard, [48](#)
- CTRL
 - KEYS, [14](#)
- DELETE
 - KEYS, [14](#)
- deleteChar
 - TextBox, [63](#)
- deltaRotation
 - EditorCam, [36](#)
- deltaScroll
 - EditorCam, [36](#)
- deltaZoomIn
 - EditorCam, [36](#)
- deltaZoomOut
 - EditorCam, [36](#)
- doc
 - Editor, [29](#)
- docChanged
 - Document, [25](#)
- docHasText
 - Document, [22](#)
- Document, [19](#)
 - absPath, [25](#)
 - buffInfo, [25](#)
 - createDir, [22](#)
 - createFile, [22](#)
 - docChanged, [25](#)
 - docHasText, [22](#)
 - Document, [21](#)
 - getAbsPath, [22](#)
 - getLineCount, [22](#)
 - getRelPath, [22](#)
 - hasChanged, [23](#)
 - init, [23](#)
 - readFile, [24](#)
 - relPath, [25](#)
 - saveFile, [24](#)
 - setBuffInfo, [24](#)
- draw
 - Editor, [27](#)
 - EditorCam, [33](#)
 - MyRect, [52](#)
 - TextBox, [63](#)
- Editor, [26](#)
 - ~Editor, [27](#)
 - camera, [29](#)
 - cbox, [29](#)
 - doc, [29](#)
 - draw, [27](#)
 - Editor, [27](#)
 - event, [30](#)
 - handleEvent, [28](#)
 - kb, [30](#)
 - lineBox, [30](#)
 - loadFromFile, [30](#)

- makeLineNum, [29](#)
- textBox, [30](#)
- window, [30](#)
- EditorCam, [31](#)
 - bottomLimitPx, [36](#)
 - camera, [36](#)
 - deltaRotation, [36](#)
 - deltaScroll, [36](#)
 - deltaZoomIn, [36](#)
 - deltaZoomOut, [36](#)
 - draw, [33](#)
 - EditorCam, [33](#)
 - getBottomLimitPx, [33](#)
 - getLineHeight, [33](#)
 - getRightLimitPx, [34](#)
 - lineHeight, [37](#)
 - marginXOffset, [37](#)
 - rightLimitPx, [37](#)
 - rotateLeft, [34](#)
 - rotateRight, [34](#)
 - scrollDown, [34](#)
 - scrollLeft, [34](#)
 - scrollRight, [35](#)
 - scrollTo, [35](#)
 - scrollUp, [35](#)
 - setCameraBounds, [35](#)
 - window, [37](#)
 - zoomIn, [35](#)
 - zoomOut, [36](#)
- ENTER
 - KEYS, [14](#)
- enterPress
 - TextBox, [63](#)
- ESCAPE
 - KEYS, [14](#)
- event
 - Editor, [30](#)
- extend
 - Command::Stack< T >, [57](#)
- fcol
 - TextBox, [66](#)
- fillColour
 - MyRect, [54](#)
- fname
 - TextBox, [66](#)
- font
 - TextBox, [66](#)
- fRect
 - MyRect, [54](#)
- fsize
 - TextBox, [66](#)
- getAbsPath
 - Document, [22](#)
- getBottomLimitPx
 - EditorCam, [33](#)
- getBounds
 - Keyboard, [40](#)
- getCmdTextEntered
 - Keyboard, [40](#)
- getLineCount
 - Document, [22](#)
- getLineHeight
 - EditorCam, [33](#)
- getLineNumber
 - Keyboard, [41](#)
- getMax
 - Command::Stack< T >, [57](#)
- getPos
 - MyRect, [52](#)
- getRelPath
 - Document, [22](#)
- getRightLimitPx
 - EditorCam, [34](#)
- getSize
 - MyRect, [53](#)
- getString
 - TextBox, [63](#)
- getTextBox
 - TextBox, [63](#)
- getTextColour
 - TextBox, [64](#)
- getTextEntered
 - Keyboard, [41](#)
- getTextSize
 - TextBox, [64](#)
- handleCmdKeyEvent
 - Keyboard, [42](#)
- handleEvent
 - Editor, [28](#)
- handleKeyEvent
 - Keyboard, [42](#)
- handleMouseEvent
 - Keyboard, [43](#)
- hasChanged
 - Document, [23](#)
- include/Kamil/CmdBox.h, [69, 70](#)
- include/Kamil/Commands.h, [70](#)
- include/Kamil/Document.h, [71, 72](#)
- include/Kamil/Editor.h, [73, 74](#)
- include/Kamil/EditorCam.h, [75, 76](#)
- include/Kamil/Keyboard.h, [76, 78](#)
- include/Kamil/MyRect.h, [79, 80](#)
- include/Kamil/TextBox.h, [80, 81](#)
- include/Kamil/Utils/Stack.h, [82, 83](#)
- init
 - Document, [23](#)
- isCmdTextEntered
 - Keyboard, [44](#)
- isKeyPressed
 - Keyboard, [44](#)
- isMouseHover
 - TextBox, [64](#)
- isTextDeleted
 - Keyboard, [45](#)

- isTextEntered
 - Keyboard, [45](#)
- kamil.cpp
 - main, [86](#)
- kb
 - Editor, [30](#)
- kbrCmd
 - Keyboard, [46](#)
- Keyboard, [38](#)
 - backSpace, [40](#)
 - bounds, [48](#)
 - ctDeleted, [48](#)
 - ctEntered, [48](#)
 - getBounds, [40](#)
 - getCmdTextEntered, [40](#)
 - getLineNumber, [41](#)
 - getTextEntered, [41](#)
 - handleCmdKeyEvent, [42](#)
 - handleKeyEvent, [42](#)
 - handleMouseEvent, [43](#)
 - isCmdTextEntered, [44](#)
 - isKeyPressed, [44](#)
 - isTextDeleted, [45](#)
 - isTextEntered, [45](#)
 - kbrCmd, [46](#)
 - Keyboard, [39](#)
 - setCmdTextEntered, [46](#)
 - setTextEntered, [46](#)
 - tDeleted, [48](#)
 - tEntered, [48](#)
 - window, [49](#)
- KEYS, [13](#)
 - BS, [14](#)
 - CTRL, [14](#)
 - DELETE, [14](#)
 - ENTER, [14](#)
 - ESCAPE, [14](#)
 - Shift_A, [14](#)
- lineBox
 - Editor, [30](#)
- lineHeight
 - EditorCam, [37](#)
- loadFromFile
 - Editor, [30](#)
- main
 - kamil.cpp, [86](#)
 - tet.cpp, [89](#)
- makeLineNum
 - Editor, [29](#)
- marginXOffset
 - EditorCam, [37](#)
- max_size
 - Command::Stack< T >, [58](#)
- mouseHover
 - TextBox, [66](#)
- MyRect, [49](#)
 - draw, [52](#)
 - fillColour, [54](#)
 - fRect, [54](#)
 - getPos, [52](#)
 - getSize, [53](#)
 - MyRect, [52](#)
 - outlineColour, [54](#)
 - outlineThicknes, [55](#)
 - pos, [55](#)
 - setFillColour, [53](#)
 - setPosition, [54](#)
 - setSize, [54](#)
 - size, [55](#)
- outlineColour
 - MyRect, [54](#)
- outlineThicknes
 - MyRect, [55](#)
- pop
 - Command::Stack< T >, [57](#)
- pos
 - MyRect, [55](#)
- printStack
 - Command::Stack< T >, [57](#)
- push
 - Command::Stack< T >, [57](#)
- readFile
 - Document, [24](#)
- README.md, [84](#)
- relPath
 - Document, [25](#)
- rightLimitPx
 - EditorCam, [37](#)
- rotateLeft
 - EditorCam, [34](#)
- rotateRight
 - EditorCam, [34](#)
- saveFile
 - Document, [24](#)
- scrollDown
 - EditorCam, [34](#)
- scrollLeft
 - EditorCam, [34](#)
- scrollRight
 - EditorCam, [35](#)
- scrollTo
 - EditorCam, [35](#)
- scrollUp
 - EditorCam, [35](#)
- setBuffInfo
 - Document, [24](#)
- setCameraBounds
 - EditorCam, [35](#)
- setCmdTextEntered
 - Keyboard, [46](#)
- setFillColour

- MyRect, 53
- setFont
 - TextBox, 64
- setPosition
 - MyRect, 54
- setSize
 - MyRect, 54
- setString
 - TextBox, 65
- setTextColour
 - TextBox, 65
- setTextEntered
 - Keyboard, 46
- setTextSize
 - TextBox, 65
- Shift_A
 - KEYS, 14
- size
 - MyRect, 55
- SP
 - Command::Stack< T >, 58
- SP_pos
 - Command::Stack< T >, 58
- src/Document.cpp, 84
- src/Editor.cpp, 84
- src/EditorCam.cpp, 85
- src/kamil.cpp, 85
- src/Keyboard.cpp, 86
- src/MyRect.cpp, 87
- src/TextBox.cpp, 87
- src/Utils/Stack.cpp, 87
- src/Utils/tet.cpp, 88
- Stack
 - Command::Stack< T >, 56
- stack_array
 - Command::Stack< T >, 58
- tbox
 - TextBox, 66
- tDeleted
 - Keyboard, 48
- tEntered
 - Keyboard, 48
- tet.cpp
 - main, 89
- TextBox, 59
 - CmdBox, 19
 - deleteChar, 63
 - draw, 63
 - enterPress, 63
 - fcol, 66
 - fname, 66
 - font, 66
 - fsize, 66
 - getString, 63
 - getTextBox, 63
 - getTextColour, 64
 - getTextSize, 64
 - isMouseHover, 64
 - mouseHover, 66
 - setFont, 64
 - setString, 65
 - setTextColour, 65
 - setTextSize, 65
 - tbox, 66
 - TextBox, 61, 62
 - window, 67
- textBox
 - Editor, 30
- window
 - Editor, 30
 - EditorCam, 37
 - Keyboard, 49
 - TextBox, 67
- zoomIn
 - EditorCam, 35
- zoomOut
 - EditorCam, 36