KText Editor

Generated by Doxygen 1.9.6

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.

The challenge started when he commented on my use of neovim and how it would be better if i used an actual IDE. I told him that ive used IDE's in the past and overall prefer the look and feel of a customised neovim. I then suggested him to learn vim himself and that he wouldnt regret it, but he declined. Kamil then told me that I should create something easier for him to use and that could potentially change his use of IDE's.

Upon being issues this challenge I had a few initial questions that I needed answering:

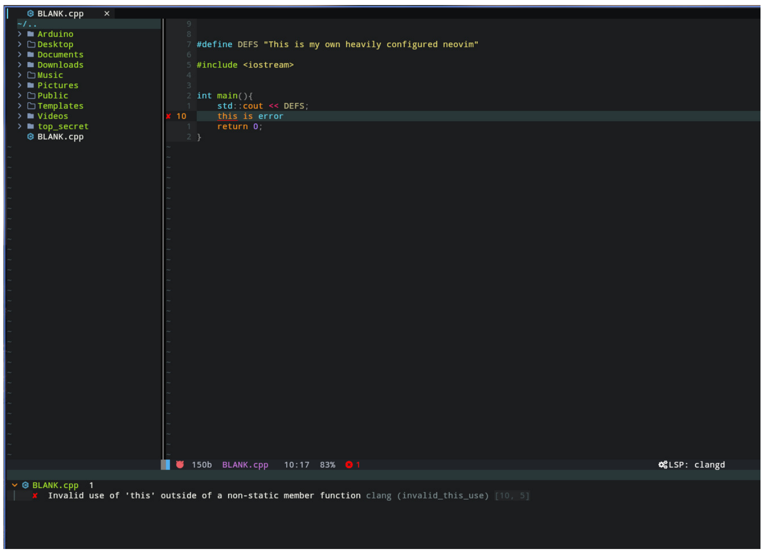
1) What is a text editor and how does it differ from an IDE? 2) How do I make a text editor for kamil 3) How do I make it efficient enough to meet his standards?

To kick things along I began to do research on Text editors and IDE's and found out that the difference between isnt 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 manipulting 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	Too many menus
		Limited in compatability

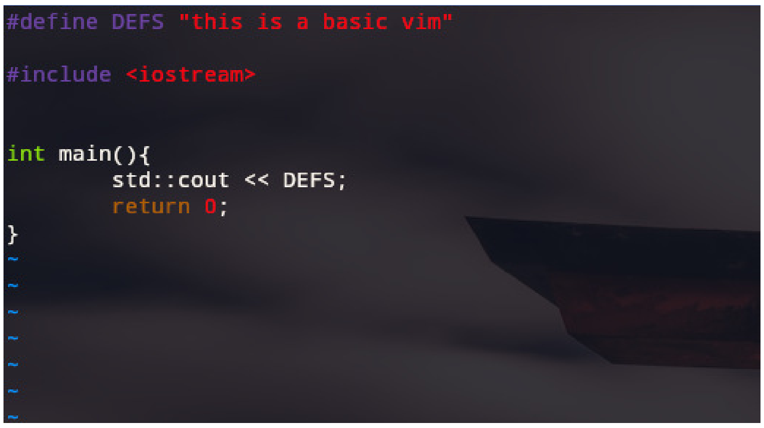
Here are pictures of some text editors and IDE's:



```
#define DEFS "This is my own heavily configured neovim"

#include <iostream>


int main(){
    std::cout << DEFS;
    this is error
    return 0;
}
```

```
150b  BLANK.cpp   10:17  83%  1                                   LSP: clangd
BLANK.cpp  1
  Invalid use of 'this' outside of a non-static member function clang (invalid_this_use) [10, 5]
```

Figure 1.1 My Neovim



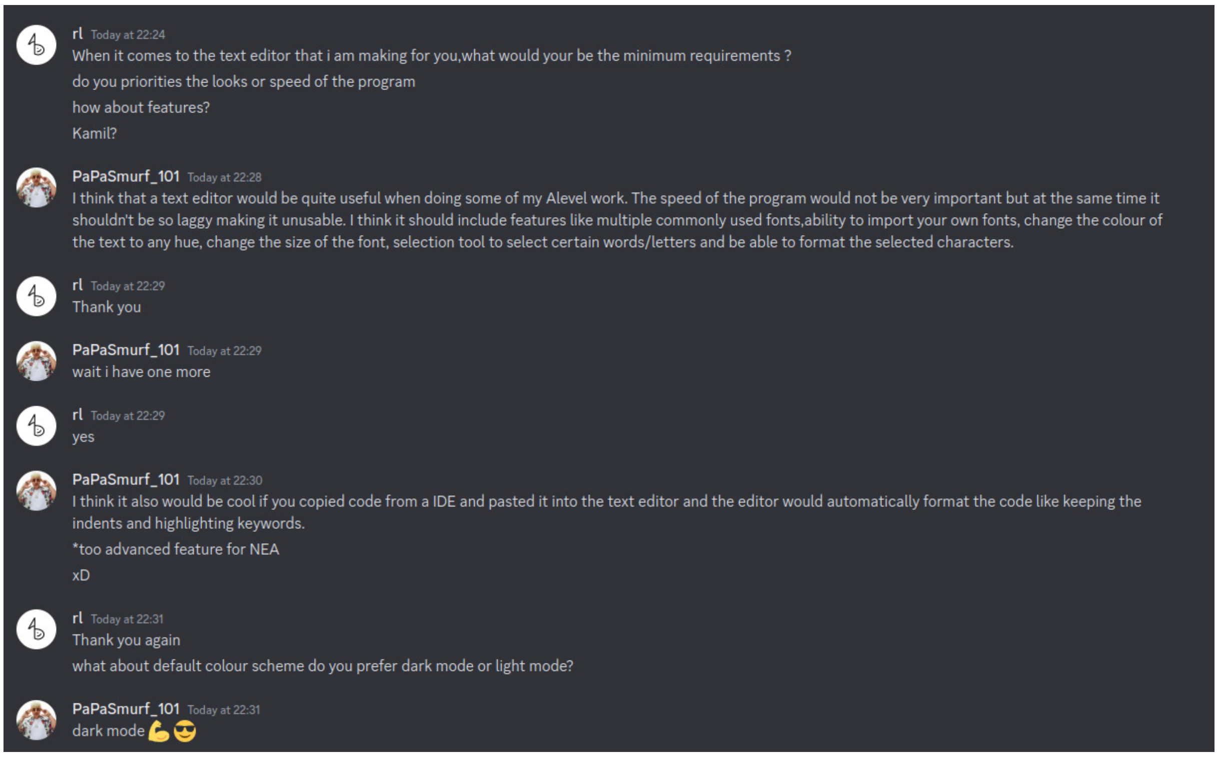
```
#define DEFS "this is a basic vim"

#include <iostream>


int main(){
        std::cout << DEFS;
        return 0;
}
```

Figure 1.2 My Vim

(annotate hte image)

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.



rl Today at 22:24
When it comes to the text editor that i am making for you,what would your be the minimum requirements ?
do you priorities the looks or speed of the program
how about features?
Kamil?

PaPaSmurf_101 Today at 22:28
I think that a text editor would be quite useful when doing some of my Alevel work. The speed of the program would not be very important but at the same time it shouldn't be so laggy making it unusable. I think it should include features like multiple commonly used fonts,ability to import your own fonts, change the colour of the text to any hue, change the size of the font, selection tool to select certain words/letters and be able to format the selected characters.

rl Today at 22:29
Thank you

PaPaSmurf_101 Today at 22:29
wait i have one more

rl Today at 22:29
yes

PaPaSmurf_101 Today at 22:30
I think it also would be cool if you copied code from a IDE and pasted it into the text editor and the editor would automatically format the code like keeping the indents and highlighting keywords.
*too advanced feature for NEA
xD

rl Today at 22:31
Thank you again
what about default colour scheme do you prefer dark mode or light mode?

PaPaSmurf_101 Today at 22:31
dark mode 💪😎

Since this is a project that could quickly grow in scale due to all the different parts of handling the editor, text and documents etc. I am willing to set a few minimum requirements my program can achieve to be usable to Kamil. The requirements are: the prgram can load and save files, change text and background colour, change font and font size.

To conclude, the objectives of the projet:

We need a program that is not laggy and has minimal delay between text being pressed and it being displayed on the screen. This can be measured by taking the time taken between a key press and the setString function by SFML.

We also need the ability to load and use multiple fonts and for it to load dynamically and save when we close the program. This is easier to check since all we need is to check the font save folder and make sure it is there when saved and loaded.

All the dynamic editor features like zooming in/out; changing the text colour and size moving around the text and text selection can be checked at runtime and can be benched mark to ensure it is still decently fast so it is not laggy and meets Kamil's preferences.

- Not laggy
- multple 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 arrorw 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

Minimum Requirements:

- Load/Save files
- Change txt and background colours
- Change font and font size

1.1.3 Limitations

The Limitations of my program are what give it a general architecure to work with. The Limits include: Time, Programming Language, formating standards, Operating System, Libraries

The project is due on 16th May 2023 leaving me only 1month and 2weeks to get everything together.

When it comes to the Programming Language I wrote my project in C++ (Cpp, Cxx, cc) with access to the C++17 language standard. I chose this language becauase I am most familiar with it and prefer it over python for larger projects like this. It is fast, efficient and allows the use of pointers for memory and data management. An example of this can be shown when passing Classes to other Classes via pointer.

The formatting standard im using is own defined by LLVM in a .clang-format file, it essentially dictates the formatting of files from how many spaces are used in a tab to length of lines and how many parameters apear on one line.

By having a seperate program keep track of all code formatting and making sure its all standardised it makes the code more modular and easy to work with since any new programmers will have an easier time understanding code if its all similar.

(include the clang-format file here)

An example being:
```
//without a formatting standard

int printAninttoOutput
```

```
    (int val) {return val;}


int setintTooutPut
(int val){
    reutrn val
}
// with a formatting standard

int Print_Int_To_Out(int val){
    return val;
}

int Set_Int_To_Out(int val){
    return val
}
```

From the examples shown above its clear that with the formatting the code is easier to read without any weird (but legal) C++ syntax, it also allows programmers to see a pattern and predict what the function they want to call is called without checking documentation.

The operating System is a default limiter and denotes how everything comes together. By default I use Linux. This has the benefit of having more support for C++ coding and development in general with the caveat of programs not being very portable to other devices like windows machines. This means that I will either need to cross-compile my program or convert Kamil, who is a windows user, over to Linux.

In addition to the operating system, Libraries, specifically graphical Libraries in conjuction with config files can decide wheather a program is cross-platform or not. Some libraries make use of Os specific functionality and function calls that arent available elsewhere.

The issue for me here is that I use Linux and Kamil uses Windows, so how do I get my program to him on windows? Well the answer is by choosing libraries that are cross-compatible and using configuration files.

For the Libraries ill be using SFML to handle the events and graphics and fmt for normal printing to standard out. Both are cross platform and are built using a cmake file.

The cmake file I use to compile and build my project is: (link to cmake file)

1.1.4 Design

Throught the creation of the project I utilised an iterative deseign procedure where I would develop a basic version of the code, test it then improve on it.This form of design procedure requires a very modular and heavily commented code base so we dont get lost when adding new features and testing and checking old ones.

(show pics of the program before and after for iterative)

My workflow is as:

- Identify feature I want to add
- Write out features it should be able to do
- Create the class in a seperate file around a template SFML project i.e. similar style to main project but not 1-1 copy
- Make sure the class follows DRY (Dont Repeat Yourself)
- Test the code against what-if cases
- Implement the code to the main project and check if it runs
- Test program
- Repeat

(example of written work for TextBox class)

Moreover, when designing the project I made use of OOP and Geneic programming using templates. Each section of my code is modular so that if someone where to take parts of it like the TextBox class, It would be similar in style to a normal SFML class with little to no difference.

(TextBox)

1.1.4.1 Design Choices

When developing the project I made a series of design choices that I thought would be best for the project.

In SFML when writing text to a screen it takes a, const sf::String& string, which devolves into std::string types and char[] arrays. Due to this and a need to be efficient I made the choice to manipulate all text input and output in a dynamic one dimensional character array (std::string), By doing this any changes that can be made I just need to loop through the string checking each character for what im looking for. They take up minimal space since it is stored as a single string which is by default 24 bytes.

This also has the added benefit of always knowing its length and size as well as being able to convert into other types when needed.

Furthermore, I also made some optimisation decisions like, passing classes used through by pointer and dynamically allocating them on the heap when created. I did this because when they are passed through by pointer the program is only accessing one instance of it and not copying the class, manipulating it and then passing the values back to it when its done like what happens by default when passing a class through parameters. This choice speeds up the program since it doesnt need to copy and directly access the class.

finish commenting the header file

include teh cmake file show python thing

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

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

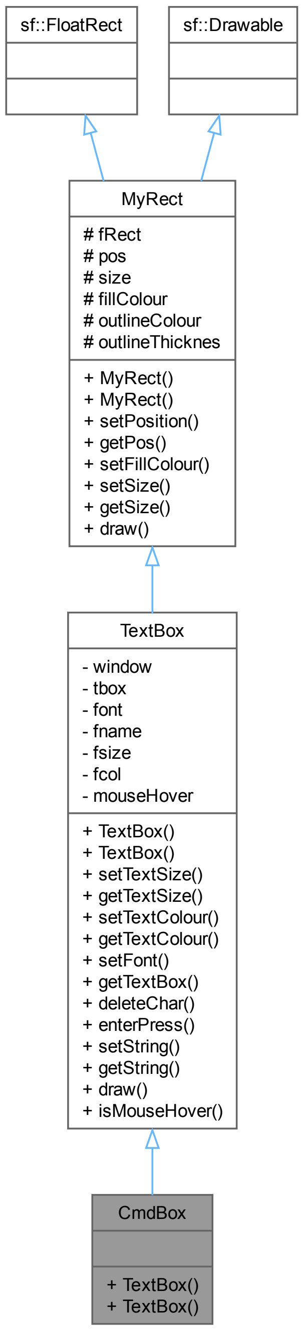
Class Documentation

7.1 CmdBox Class Reference

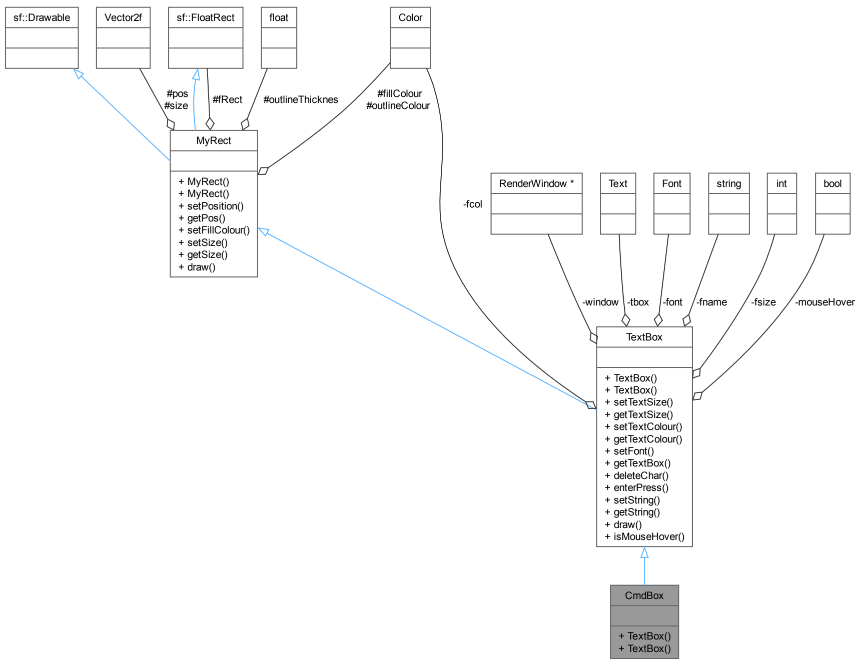
Class to handle the command TextBox.

```
#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 Text.

- 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

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:

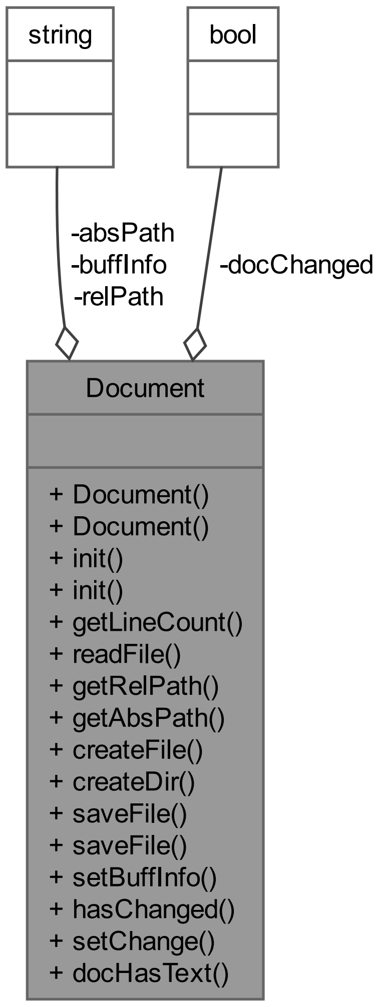
- include/Kamil/CmdBox.h

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
- void init (std::string inF)

    initialise the file
- 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 ()

  save to a file
- void setBuffInfo (std::string info)

  save file infor to buffer
- bool hasChanged ()

  if the file has changed
- void setChange ()

  set file has changed
- bool docHasText ()

  check if theres text in the file

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

fileP	- file path

7.2.3 Member Function Documentation

7.2.3.1 createDir()

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

create a directory

Parameters

void	

Returns

void

7.2.3.2 createFile()

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

create the file

Parameters

void	

Returns

void

7.2.3.3 docHasText()

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

check if theres text in the file

Parameters

void	

Returns

bool - true if contains text

7.2.3.4 getAbsPath()

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

get the relative path

Parameters

void	

Returns

string for absolute path

7.2.3.5 getLineCount()

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

@breif get the count of lines

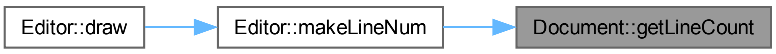
Parameters

void	

Returns

int - line count

Here is the caller graph for this function:



7.2.3.6 getRelPath()

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

get the relative path

Parameters

void	

Returns

string for relative path

7.2.3.7 hasChanged()

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

if the file has changed

Parameters

void	

Returns

bool - true if file has changed

7.2.3.8 init() `[1/2]`

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

initialise the file

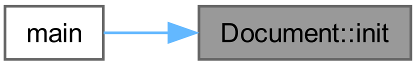
Parameters

void	

Returns

void

Here is the caller graph for this function:



7.2.3.9 init() [2/2]

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

initialise the file

Parameters

inF	- file location

Returns

void

7.2.3.10 readFile()

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

read the file

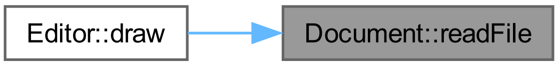
Parameters

void	

Returns

string containing the file info

Here is the caller graph for this function:



7.2.3.11 saveFile() [1/2]

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

save to a file

Parameters

void	

Returns

bool - true if saved

7.2.3.12 saveFile() [2/2]

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

save to a file

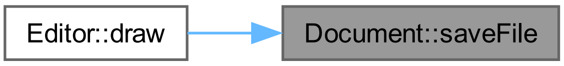
Parameters

string	- filename to save to

Returns

bool - true if saved

Here is the caller graph for this function:



7.2.3.13 setBuffInfo()

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

save file infor to buffer

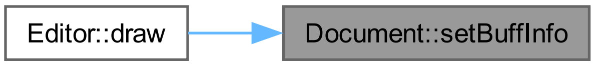
Parameters

string	buffer info

Returns

void

Here is the caller graph for this function:



7.2.3.14 setChange()

```
void Document::setChange ( )
```

set file has changed

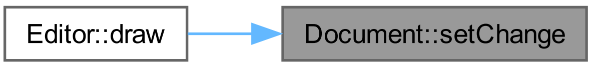
Parameters

void	

Returns

void

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:

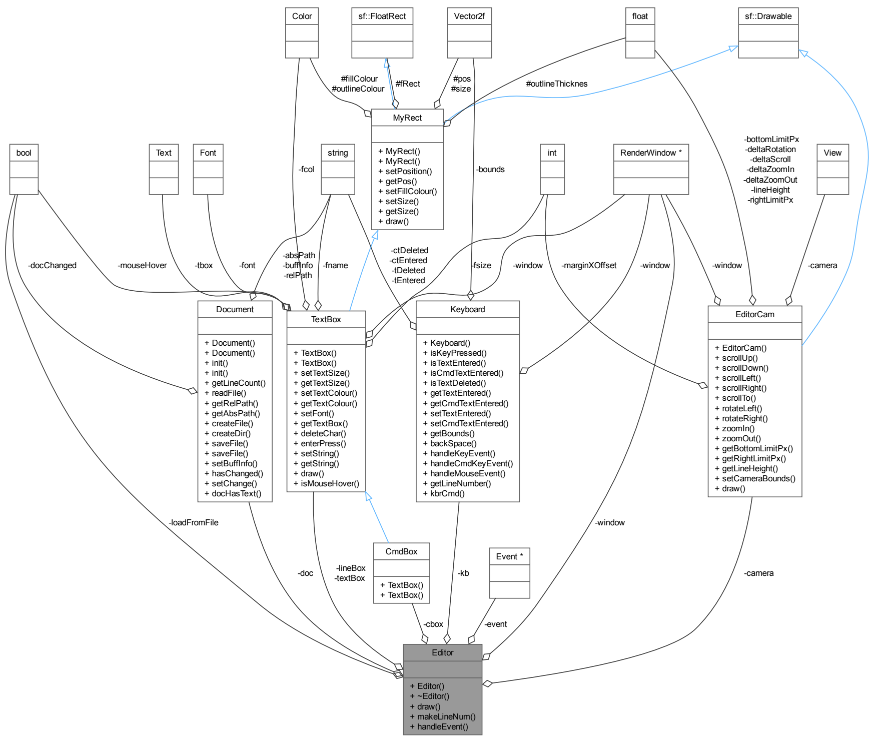
- include/Kamil/Document.h
- src/Document.cpp

7.3 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 ∗window, sf::Event ∗event, Document ∗doc)

  Constructor for Editor.
- ∼Editor ()

  Destructor for Editor class.
- void draw ()

  function that draws everything to RenderWindow
- void makeLineNum ()

  making the line numbers
- void handleEvent ()

  handle the events for the Editor

Private Attributes

- Document ∗ doc
- TextBox ∗ textBox
- CmdBox ∗ cbox
- sf::RenderWindow ∗ window
- sf::Event ∗ event
- TextBox lineBox
- EditorCam camera
- Keyboard kb
- bool loadFromFile

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

window	- pointer to main RenderWindow
event	- pointer to main event
doc	- pointer to document

7.3.2.2 ∼Editor()

```
Editor::∼Editor ( )
```

Destructor for Editor class.

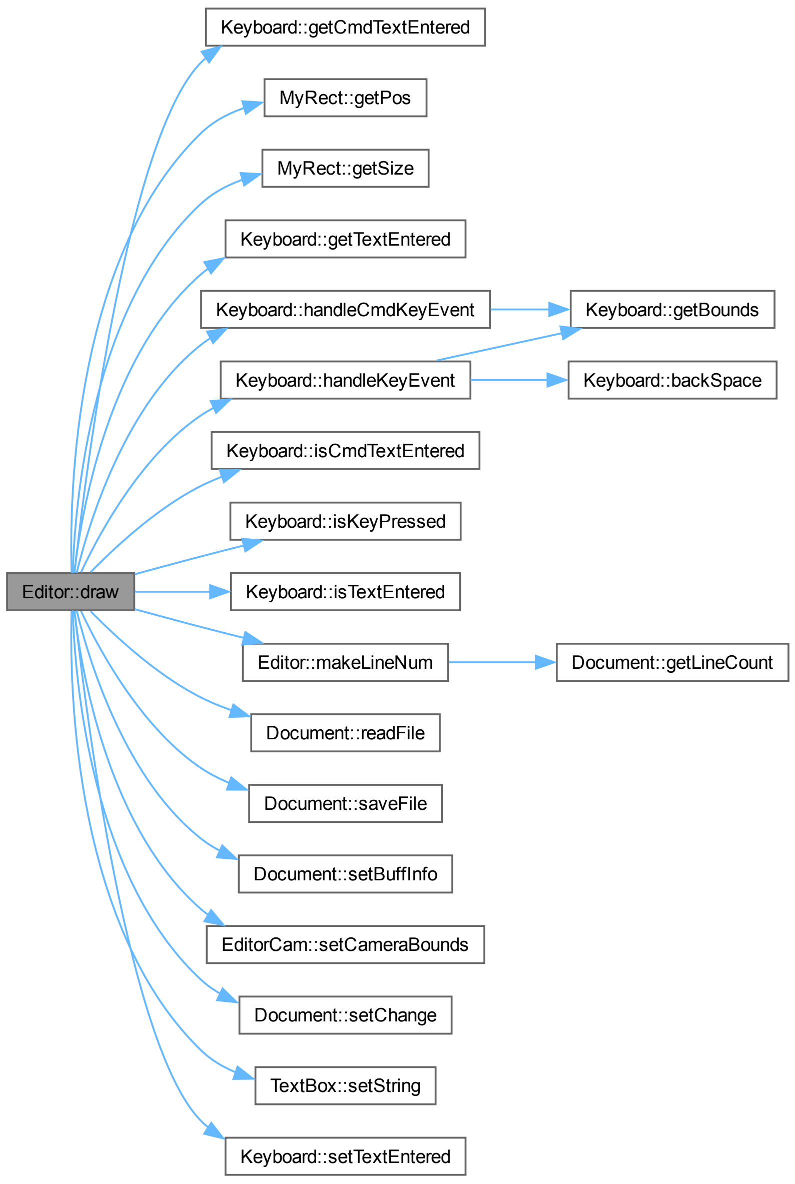
7.3.3 Member Function Documentation

7.3.3.1 draw()

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

function that draws everything to RenderWindow

SOON DEPRECATED Here is the call graph for this function:

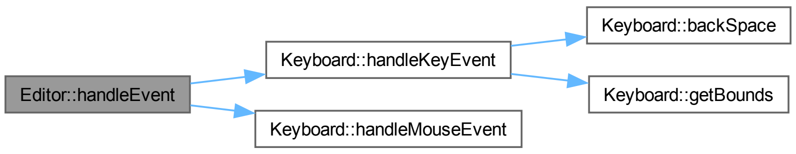


7.3.3.2 handleEvent()

```
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.handleMouse←
Events(); Here is the call graph for this function:



7.3.3.3 makeLineNum()

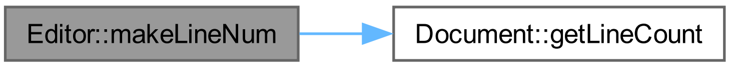
```
void Editor::makeLineNum ( )
```

making the line numbers

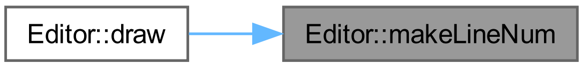
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



7.3.4 Member Data Documentation

7.3.4.1 camera

`EditorCam Editor::camera [private]`

for the camera

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]`

refernce to event

7.3.4.5 kb

`Keyboard Editor::kb [private]`

handles keyboard events

7.3.4.6 lineBox

`TextBox Editor::lineBox [private]`

for the line number

7.3.4.7 loadFromFile

`bool Editor::loadFromFile [private]`

check if we are loading from file

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]
```

refernce to RenderWindow

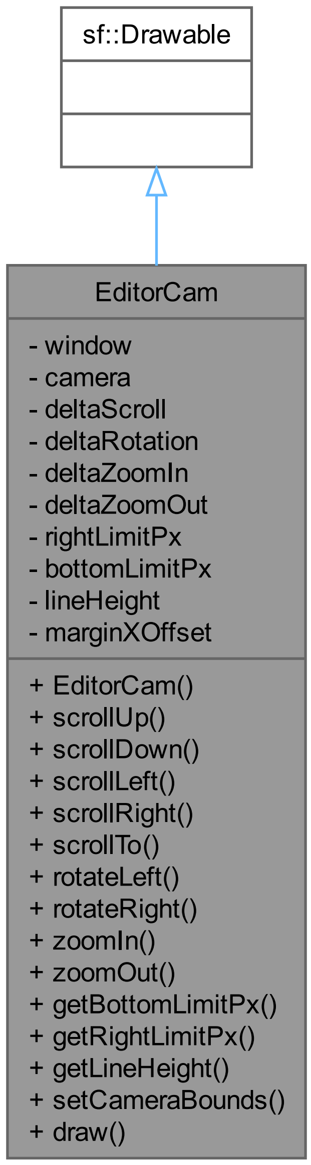
The documentation for this class was generated from the following files:

- include/Kamil/Editor.h
- src/Editor.cpp

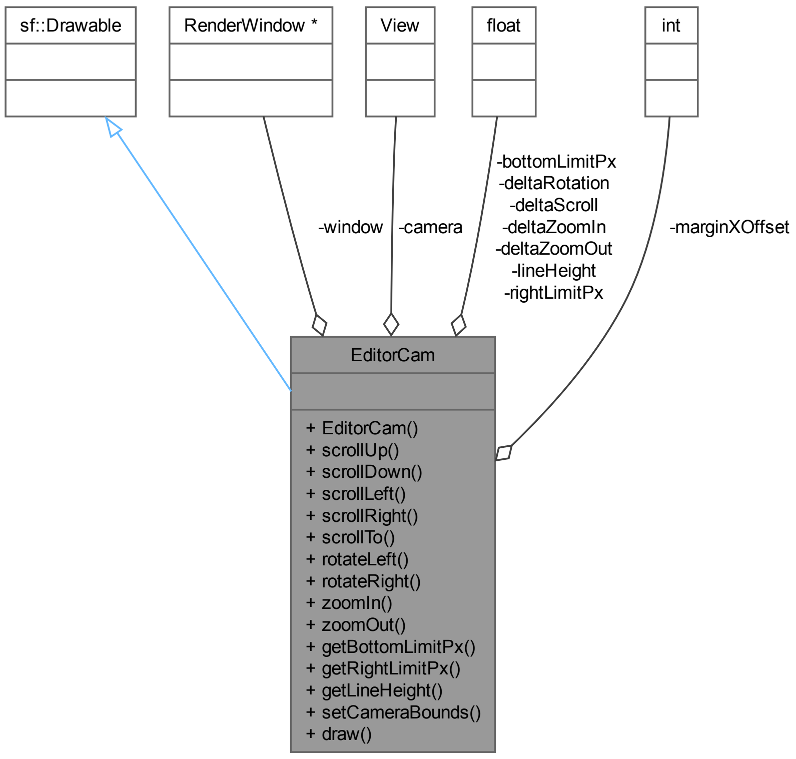
7.4 EditorCam Class Reference

```
#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)

  Constructor for EditorCam.
- 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)

  set camera bounds
- void draw (sf::RenderTarget &target, sf::RenderStates states) const override

  draw to window

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 )
```

Constructor for EditorCam.

Parameters

sf::RenderWindow∗	- pointer to main window
float	- scrolling delta value
float	- rotation delta value
float	- zoom in delta value
float	- zoom out delta value

7.4.2 Member Function Documentation

7.4.2.1 draw()

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

draw to window

Parameters

sf::RenderTarget&	- render target reference
sf::RenderStates	- rendr states

Returns

void

7.4.2.2 getBottomLimitPx()

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

@breif get bottom pixel limit

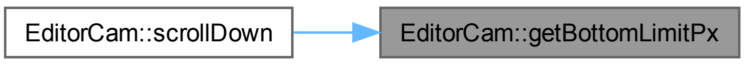
Parameters

void	

Returns

float - pixel limit

Here is the caller graph for this function:



7.4.2.3 getLineHeight()

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

@breif get line height

Parameters

void	

Returns

int - line height

7.4.2.4 getRightLimitPx()

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

@breif get right pixel limit

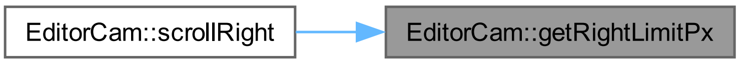
Parameters

void	

Returns

float - pixel limit

Here is the caller graph for this function:



7.4.2.5 rotateLeft()

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

@breif rotate camera left

Parameters

void	

Returns

void

7.4.2.6 rotateRight()

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

@breif rotate camera right

Parameters

void	

Returns

void

7.4.2.7 scrollDown()

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

@breif move camera down

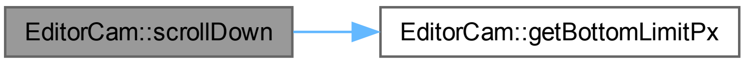
Parameters

void	

Returns

void

Here is the call graph for this function:



7.4.2.8 scrollLeft()

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

@breif move camera left

Parameters

void	

Returns

void

7.4.2.9 scrollRight()

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

@breif move camera right

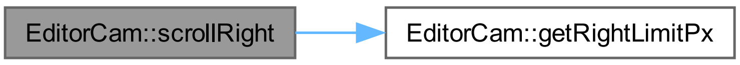
Parameters

void	

Returns

void

Here is the call graph for this function:



7.4.2.10 scrollTo()

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

@breif move camera to position

Parameters

float	- x value
float	- y value

Returns

void

7.4.2.11 scrollUp()

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

@breif move camera up

Parameters

void	

Returns

void

7.4.2.12 setCameraBounds()

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

set camera bounds

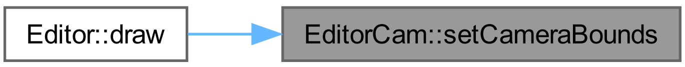
Parameters

int	- width
int	- height

Returns

void

Here is the caller graph for this function:



7.4.2.13 zoomIn()

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

@breif zoom camera in

Parameters

void	

Returns

void

7.4.2.14 zoomOut()

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

@breif zoom camera out

Parameters

void	

Returns

void

7.4.3 Member Data Documentation

7.4.3.1 bottomLimitPx

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

bottom pixel limit

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 tiem 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]
```

line height

7.4.3.8 marginXOffset

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

margin offset

7.4.3.9 rightLimitPx

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

delta time for zoomin/out right pixel limit

7.4.3.10 window

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

refernce to window

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

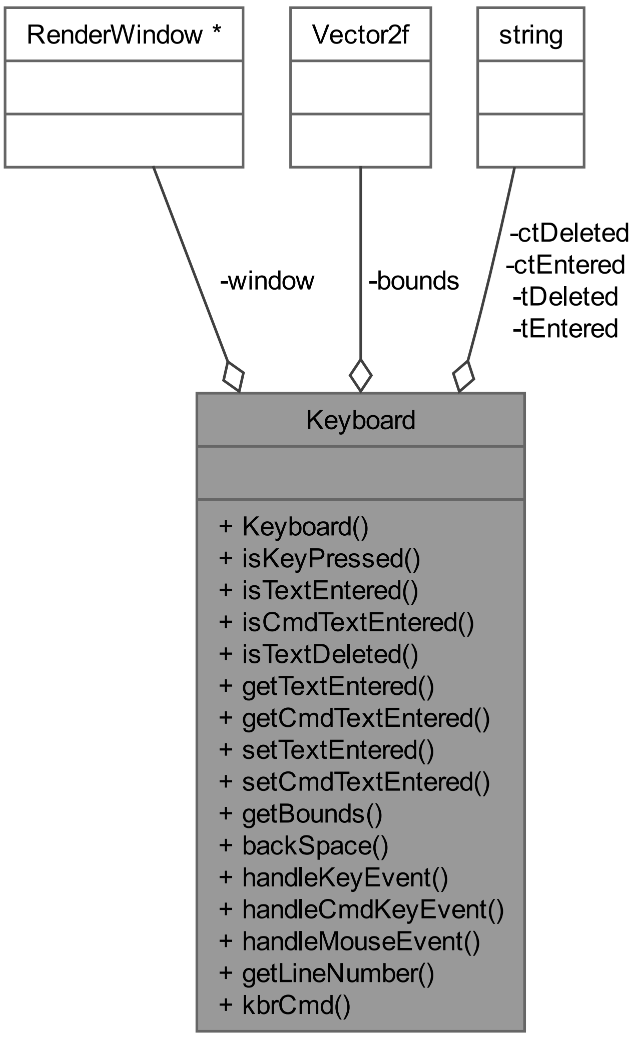
- include/Kamil/EditorCam.h
- src/EditorCam.cpp

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 ()

  get line number
- 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

win	- reference to main window
bounds	- 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

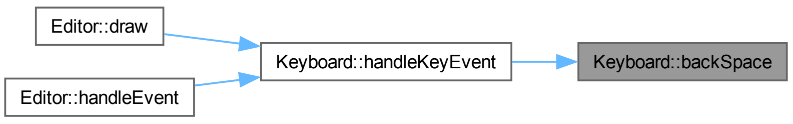
Parameters

void	

Returns

void

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

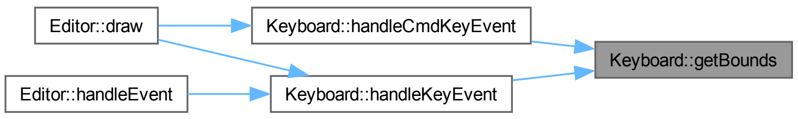
Parameters

void	

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

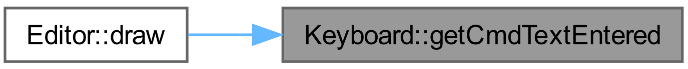
Parameters

void	

Returns

std::string text entered

Here is the caller graph for this function:



7.5.3.4 getLineNumber()

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

get line number

Parameters

void	

Returns

int - line number

7.5.3.5 getTextEntered()

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

returns text entered

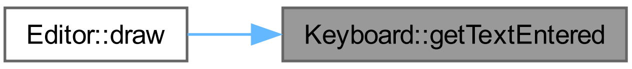
Parameters

void	

Returns

std::string text entered

Here is the caller graph for this function:



7.5.3.6 handleCmdKeyEvent()

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

handle keyboard events

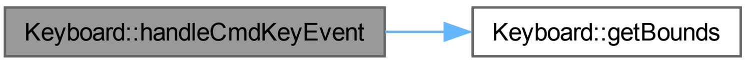
Parameters

event	- to get text entered from events

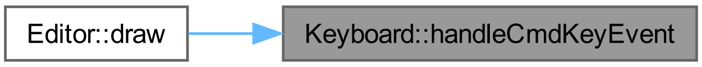
Returns

void

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

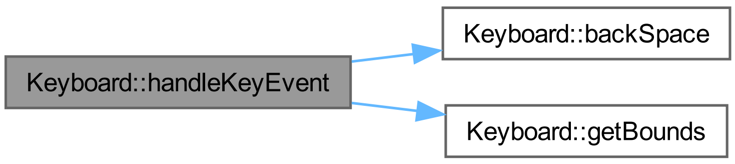
Parameters

event	- to get text entered from events

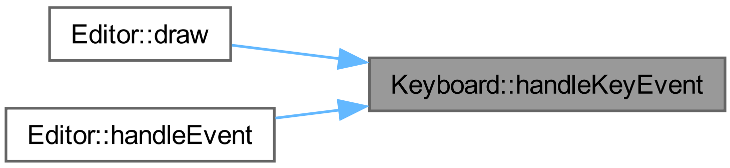
Returns

void

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

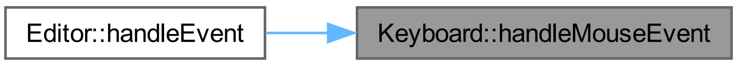
Parameters

event	- to get text entered from events

Returns

void

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

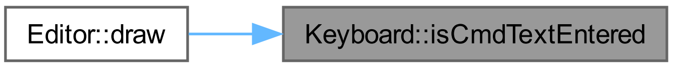
Parameters

void	

Returns

bool tru eif key is pressed false if not

Here is the caller graph for this function:



7.5.3.10 isKeyPressed()

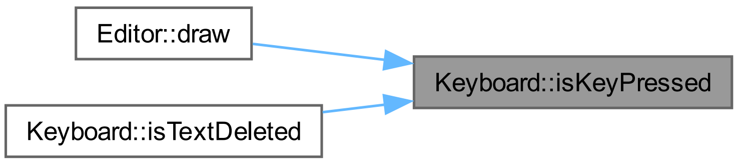
```
bool Keyboard::isKeyPressed (
            sf::Keyboard::Key key )
```

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

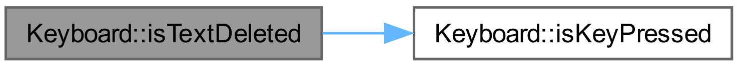
Parameters

void	

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

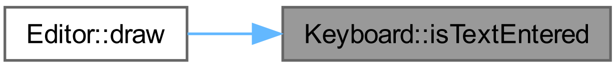
Parameters

void	

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

nstring	- new string

Returns

void

7.5.3.15 setTextEntered()

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

sets text

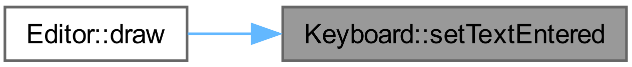
Parameters

nstring	- new string

Returns

void

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 enterd 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:

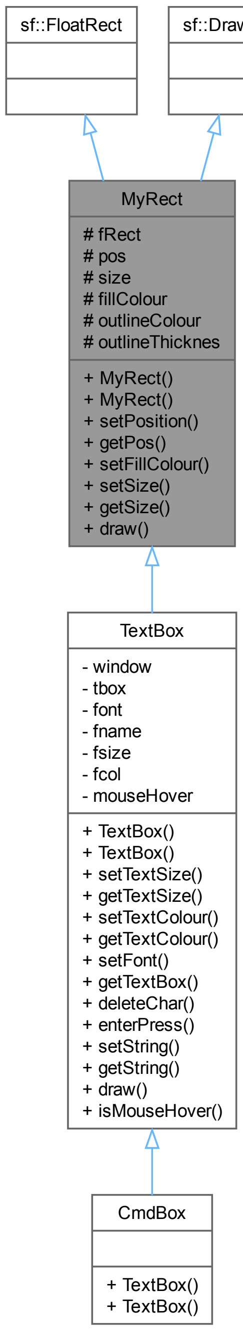
- include/Kamil/Keyboard.h
- src/Keyboard.cpp

7.6 MyRect Class Reference

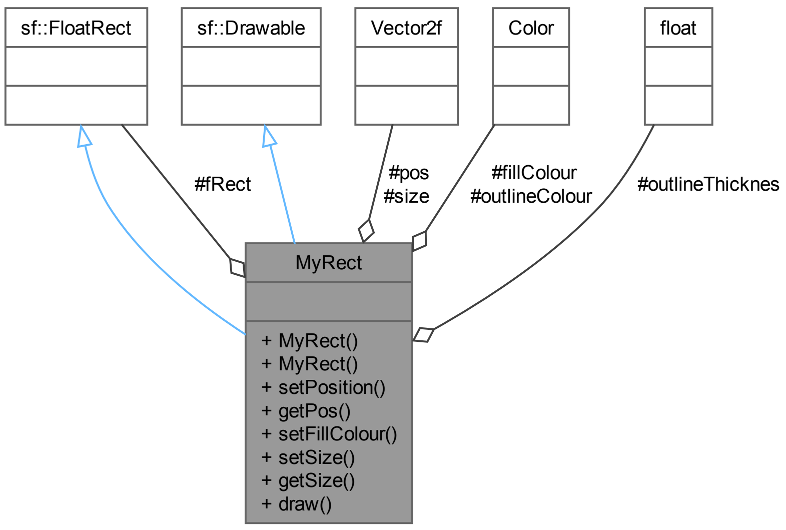
gives extra functionality to FloatRect

```
#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

  virutal 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 sytax 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

pos	- position of rect
size	- size of rect
fillColour	- fill colour of rect
outlineColour	- ouline colour of rect
outlineThicknes	- 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 My↩ Rect.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 by overriding a virtual method

7.6.3.2 getPos()

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

get the position of rect

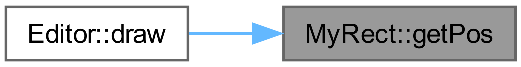
Parameters

void	

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

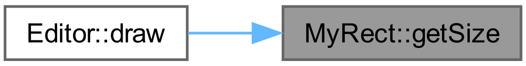
Parameters

void	

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

sf::Color	colour

Returns

void

7.6.3.5 setPosition()

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

sets the position of rect

Parameters

sf::Vector2f	pos

7.6.3.6 setSize()

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

set the size of the rect

Parameters

sf::Vector2f	size

Returns

void

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

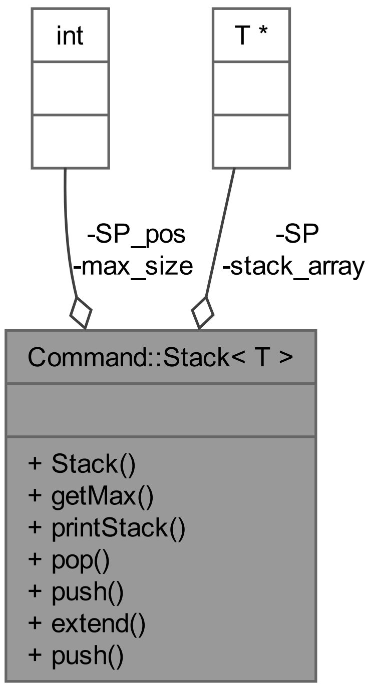
The documentation for this class was generated from the following files:

- 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
```

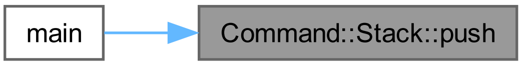
7.7.2.5 push() [1/2]

```
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:

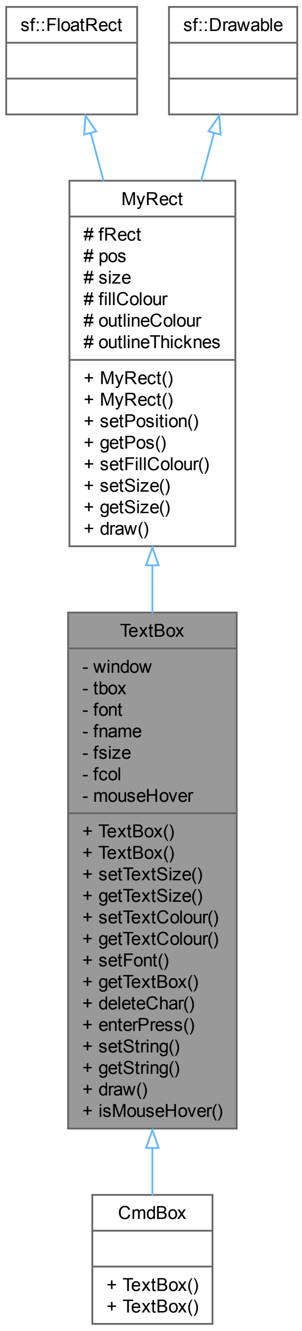
- include/Kamil/Commands.h
- include/Kamil/Utils/Stack.h
- src/Utils/Stack.cpp

7.8 TextBox Class Reference

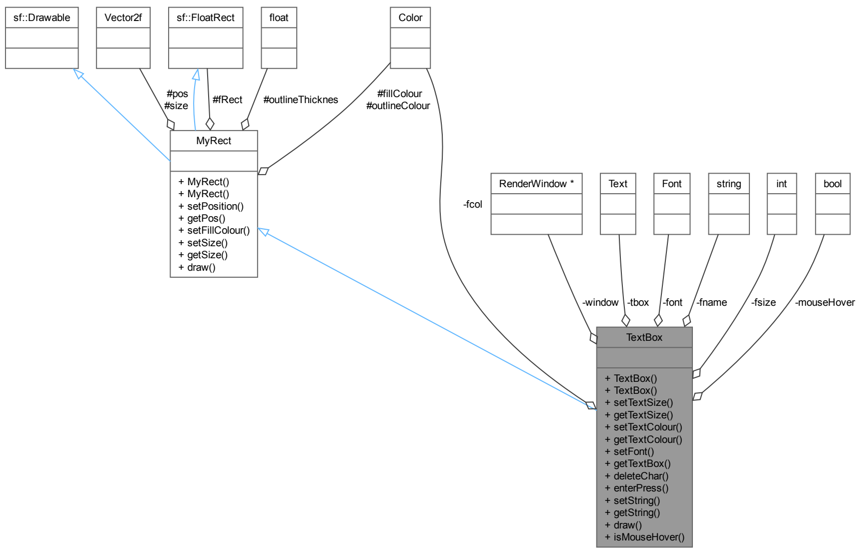
A class that makes a Textbox in SFML.

```
#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 Text.
- 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.

Constrcutor Implementation for TextBox class.

Parameters

win	- RenderWindow the TextBox is drawn onto
pos	- the initial position of the TextBox
size	- the initial size of the TextBox
sfont	- the initial font used by the TextBox
fsize	- the inital font size
fcol	- the initial font colour
background	- the initial background colour
thicc	- the padding for the RectangleShape

Implementation of the TextBox class

Note

other structs or classes may be used here

Parameters

win	- RenderWindow the TextBox is drawn onto
pos	- the initial position of the TextBox
size	- the initial size of the TextBox
sfont	- the initial font used by the TextBox
fsize	- the inital font size
fcol	- the initial font colour
background	- the initial background colour
thicc	- 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.

Parameters

void	

Returns

void

7.8.3.2 draw()

```
void TextBox::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)

Example of polymorphism

7.8.3.3 enterPress()

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

Handles Enter key press.

Parameters

void	

Returns

void

7.8.3.4 getString()

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

returns the text in tbox

Parameters

void	

Returns

type std::string

7.8.3.5 getTextBox()

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

Get both the Text.

Parameters

void	

Returns

type Boxv2 that contains textbox and cmdbox

7.8.3.6 getTextColour()

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

Get the colour of the text.

Parameters

void	

Returns

sf::Colour textColour

7.8.3.7 getTextSize()

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

Get the size of the text.

Parameters

void	

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

font	file dir of font

Returns

void

7.8.3.10 setString()

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

Sets the string.

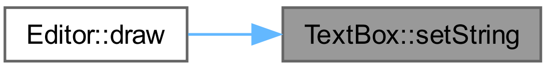
Parameters

nstring	- new string placed on tbox

Returns

void

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

fill	font colour

Returns

void

7.8.3.12 setTextSize()

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

Set the size of the text.

Parameters

size	text size

Returns

void

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:

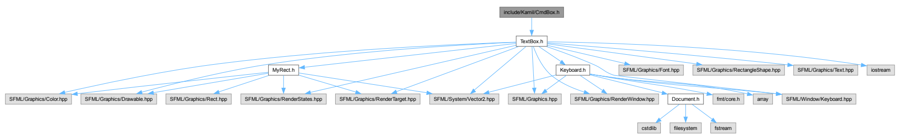
- include/Kamil/TextBox.h
- src/TextBox.cpp

Chapter 8

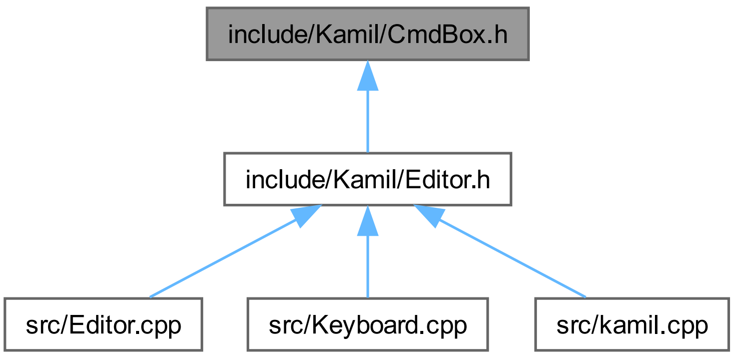
File Documentation

8.1 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

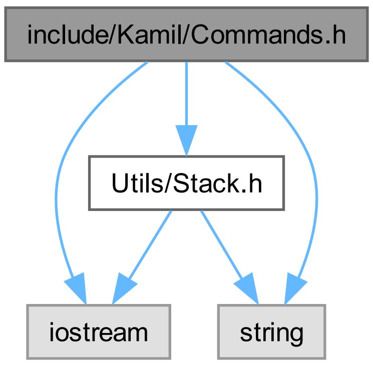
  Class to handle the command TextBox.

8.2 CmdBox.h

Go to the documentation of this file.
```
#ifndef KAMIL_CMDBOX_H
#define KAMIL_CMDBOX_H

#include "TextBox.h"

class CmdBox : public TextBox {
public:
  using TextBox::TextBox;
};
#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.
```
#ifndef KAMIL_COMMANDS_H
#define KAMIL_COMMANDS_H

#include <iostream>
#include <string>

#include "Utils/Stack.h"

```

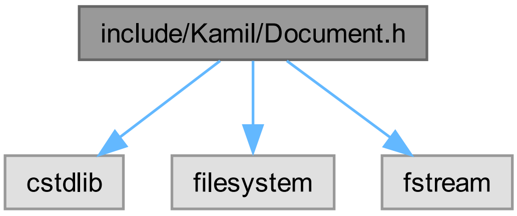
```
namespace Command {
// template <typename T>
// class Node{ // used for LinkedList
//     public:
//         Node();
//         Node(T);
//         T data;
//         Node* next;
// };

// template <typename T>
// class LinkedList{
//     public:
//         LinkedList();
//         void insertNode(int);
//         void printList();
//         void deleteNode(int);
//     private:
//         Node<T>* head;
// };
template <typename> class Stack;

//    class Undo{};

//    class Redo{};
} // namespace Command

#endif // KAMIL_COMMANDS_H
```

8.5 include/Kamil/Document.h File Reference

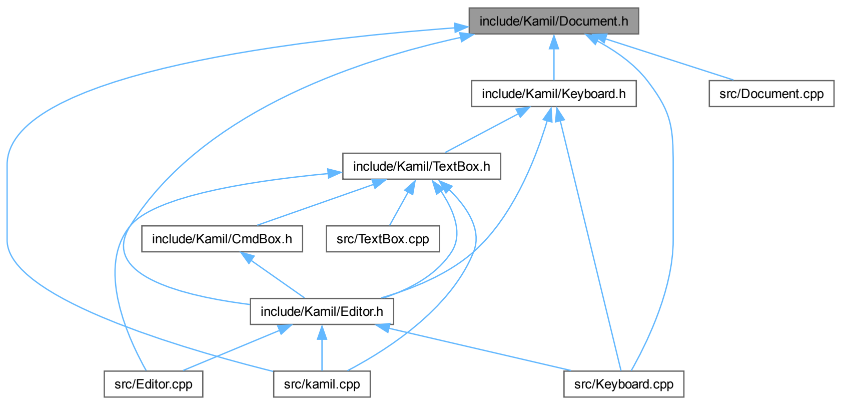
Interface file for the Document class.

```
#include <cstdlib>
#include <filesystem>
#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.

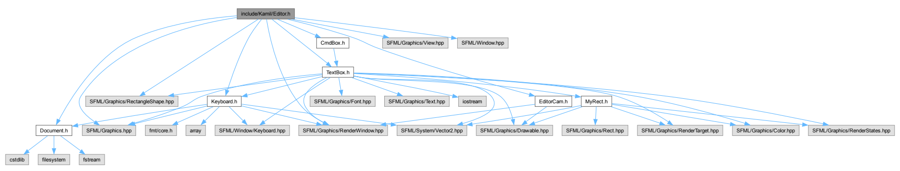
```
#ifndef KAMIL_DOCUMENT_H
#define KAMIL_DOCUMENT_H

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

class Document {
public:
  Document();

  Document(std::string fileP);

  void init();

  void init(std::string inF);

  int getLineCount();

  std::string readFile();

  std::string getRelPath();

  std::string getAbsPath();

  void createFile();

  void createDir();

  bool saveFile(const std::string &filename);

  bool saveFile();

  void setBuffInfo(std::string info);

  bool hasChanged();

  void setChange();

  bool docHasText();

  // void addTextToPos(std::string txt, int pos);

private:
  std::string relPath;
  std::string absPath;
  std::string buffInfo;
  bool docChanged;
};
#endif // KAMIL_DOCUMENT_H
```

8.7 include/Kamil/Editor.h File Reference

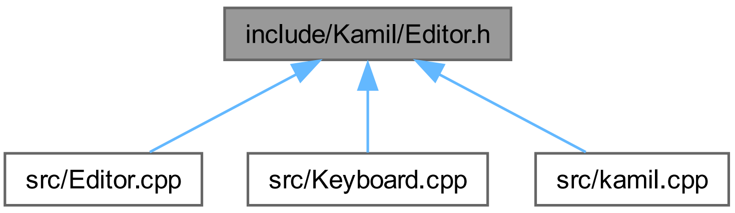
Interface file for the Editor class.

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

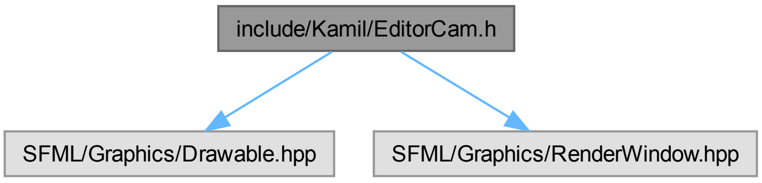
```
#ifndef KAMIL_EDITOR_WINDOW_HPP
#define KAMIL_EDITOR_WINDOW_HPP

#include <SFML/Graphics.hpp>
#include <SFML/Graphics/RectangleShape.hpp>
#include <SFML/Graphics/RenderWindow.hpp>
#include <SFML/Graphics/View.hpp>
#include <SFML/Window.hpp>

#include "CmdBox.h"
#include "Document.h"
#include "EditorCam.h"
#include "Keyboard.h"
#include "TextBox.h"

class Editor {
public:
  Editor(sf::RenderWindow *window, sf::Event *event, Document *doc);

  ~Editor();

  void draw();

  void makeLineNum();

  void handleEvent();

private:
  Document *doc;
  TextBox *textBox;
  CmdBox *cbox;
  sf::RenderWindow *window;
  sf::Event *event;
  TextBox lineBox;
  EditorCam camera;
  Keyboard kb;
  bool loadFromFile;
};

#endif // KAMIL_EDITOR_WINDOW_HPP
```

8.9 include/Kamil/EditorCam.h File Reference

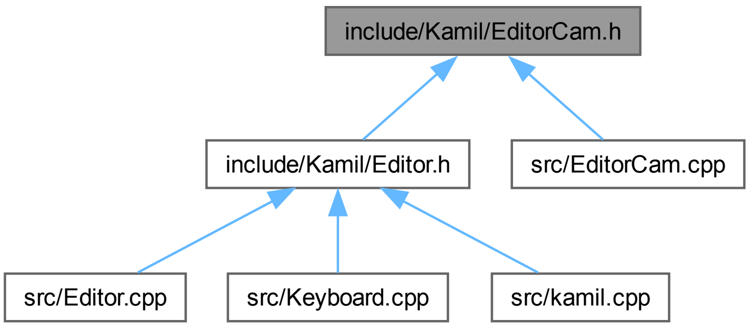
Implementation of EditorCam class.

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

Include dependency graph for EditorCam.h:


include/Kamil/EditorCam.h
SFML/Graphics/Drawable.hpp
SFML/Graphics/RenderWindow.hpp

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



Classes

- class EditorCam

8.9.1 Detailed Description

Implementation of EditorCam class.

Contains methods to manipulate the camera for the Editor

8.10 EditorCam.h

Go to the documentation of this file.

```
#ifndef KAMIL_EDITOR_CAM_H
#define KAMIL_EDITOR_CAM_H


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

class EditorCam : public sf::Drawable {
public:

  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:
  sf::RenderWindow *window;
  sf::View camera;
  float deltaScroll;
  float deltaRotation;
  float deltaZoomIn, deltaZoomOut;
  float rightLimitPx;
  float bottomLimitPx;
  float lineHeight;
  int marginXOffset;
};

#endif // KAMIL_EDITOR_CAM_H
```

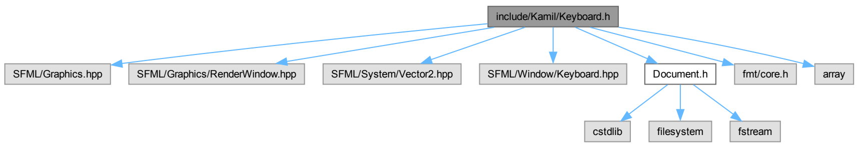
8.11 include/Kamil/Keyboard.h File Reference

Interface file for Keyboard.h.

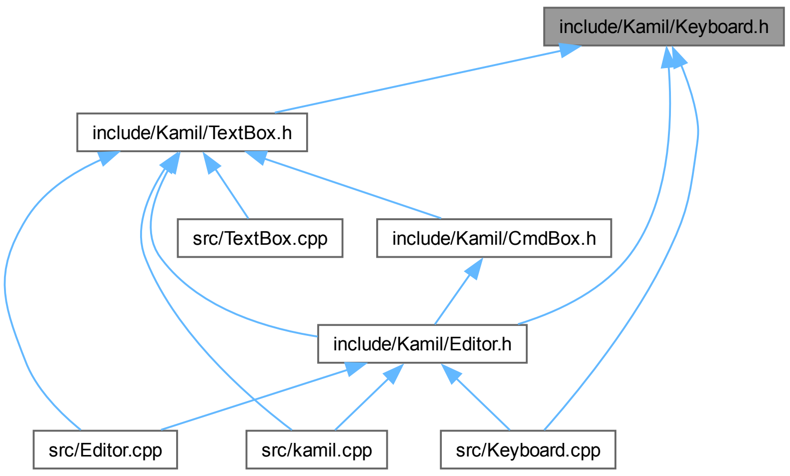
```
#include <SFML/Graphics.hpp>
#include <SFML/Graphics/RenderWindow.hpp>
#include <SFML/System/Vector2.hpp>
#include <SFML/Window/Keyboard.hpp>
#include "Document.h"
#include <fmt/core.h>
```

```
#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 manging input of keyboard data and their corresponding command

8.12 Keyboard.h

Go to the documentation of this file.

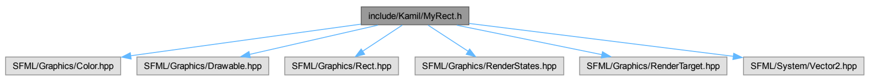
```
#ifndef KAMIL_KEYBOARD_H
#define KAMIL_KEYBOARD_H

#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>

namespace KEYS {
enum {
  ESCAPE = 0x1B,
  ENTER = 0xD,
  BS = 0x8,
  Shift_A = 0x41,
  CTRL = 0x11,
  DELETE = 0x7f,
};
}

#ifdef USE_KEYS

#define "LControl" sf::Keyboard::KEYS::LControl

#endif

class Keyboard {
public:
  Keyboard(sf::RenderWindow *win, Document *doc, sf::Vector2f bounds);

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

  bool isTextEntered();

  bool isCmdTextEntered();

  bool isTextDeleted();

  std::string getTextEntered();

  std::string getCmdTextEntered();

  void setTextEntered(std::string);

  void setCmdTextEntered(std::string);

  sf::Vector2f getBounds() const;

  void backSpace();

  void handleKeyEvent(sf::Event &event);

  void handleCmdKeyEvent();

  void handleMouseEvent(sf::Event &event); // not implemented yet

  int getLineNumber();

  template <typename T, size_t N, typename... Args> void kbrCmd(Args... args) {
    std::array<T, N> val{args...};
    for (const auto &element : val) {
      fmt::print("{}", element);
    }
  }

  // get position in text

private:
  sf::RenderWindow *window;
  // Document* doc;
  sf::Vector2f bounds;
  std::string tEntered;
  std::string tDeleted;
  std::string ctEntered;
  std::string ctDeleted;
};
#endif // KAMIL_KEYBOARD_H
```

8.13 include/Kamil/MyRect.h File Reference

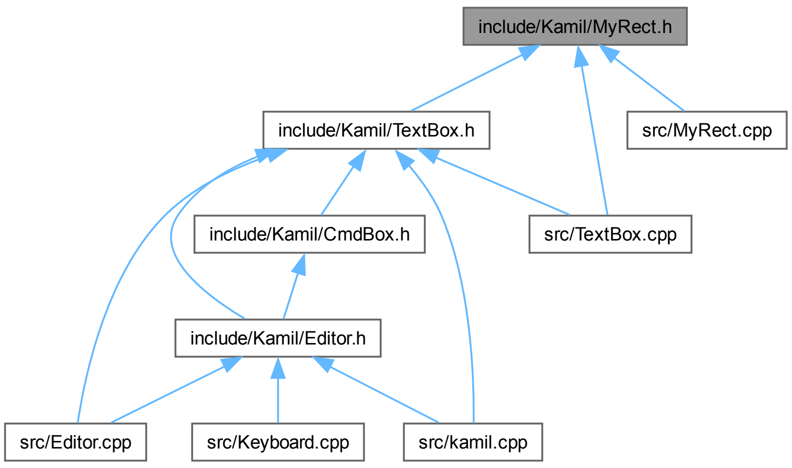
Interface file for the MyRect class.

```
#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.

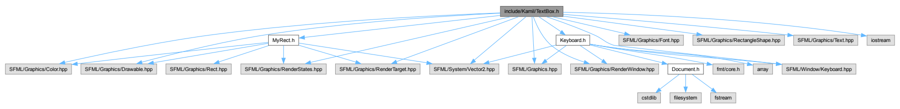
```
#ifndef KAMIL_MYRECT_H
#define KAMIL_MYRECT_H

#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>

class MyRect : public sf::FloatRect, public sf::Drawable {
public:
  MyRect(sf::Vector2f pos, sf::Vector2f size, sf::Color fillColour,
         sf::Color outlineColour, float outlineThicknes);
  MyRect();

  void setPosition(sf::Vector2f pos);

  sf::Vector2f getPos() const;

  void setFillColour(sf::Color colour);

  void setSize(sf::Vector2f size);

  sf::Vector2f getSize() const;

  void draw(sf::RenderTarget &target, sf::RenderStates states) const override;

protected:
  sf::FloatRect fRect;
  sf::Vector2f pos;
  sf::Vector2f size;
  sf::Color fillColour;
  sf::Color outlineColour;
  float outlineThicknes;
};

#endif // KAMIL_MYRECT_H
```

8.15 include/Kamil/TextBox.h File Reference

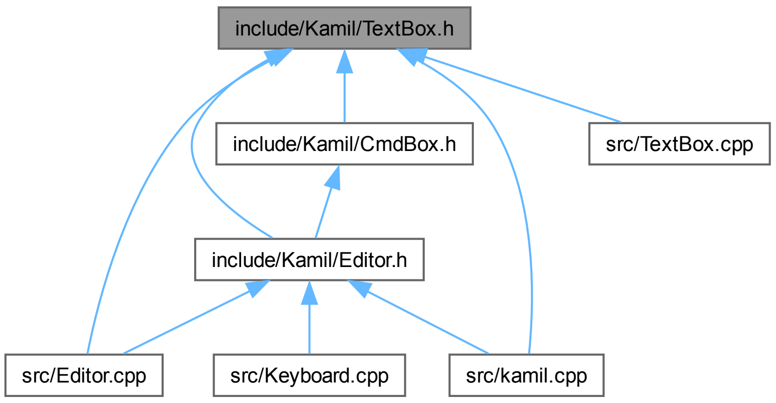
```
#include <SFML/Graphics.hpp>
#include <SFML/Graphics/Color.hpp>
#include <SFML/Graphics/Drawable.hpp>
#include <SFML/Graphics/Font.hpp>
#include <SFML/Graphics/RectangleShape.hpp>
#include <SFML/Graphics/RenderStates.hpp>
```

```
#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.

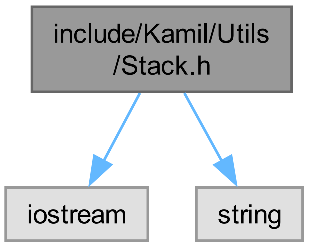
```
00001 #ifndef KAMIL_TEXTBOX_HPP
00002 #define KAMIL_TEXTBOX_HPP
00003
00012 #include <SFML/Graphics.hpp>
00013 #include <SFML/Graphics/Color.hpp>
00014 #include <SFML/Graphics/Drawable.hpp>
00015 #include <SFML/Graphics/Font.hpp>
00016 #include <SFML/Graphics/RectangleShape.hpp>
00017 #include <SFML/Graphics/RenderStates.hpp>
00018 #include <SFML/Graphics/RenderTarget.hpp>
00019 #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"

/*
 *
 * TODO:
 *       Make a RectangleShape that acts as the bounds of the TextBox
 *       then add limits to the textbox so it stays in the limits
 *
 *       Add the Keybord manager class here and use its methods
 *       to handle the key events
 */

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

  void setTextSize(int size);

  int getTextSize() const;

  void setTextColour(sf::Color colour);

  sf::Color getTextColour() const;

  void setFont(sf::Font &font);

  sf::Text getTextBox() const;

  void deleteChar();

  void enterPress();

  void setString(std::string nstring);

  std::string getString() const;

  void draw(sf::RenderTarget &target, sf::RenderStates states) const override;

  bool isMouseHover();

private:
  sf::RenderWindow *window;
  sf::Text tbox{};
  sf::Font font{};
  std::string fname{};
  int fsize{};
  sf::Color fcol{};
  bool mouseHover;
};
#endif // KAMIL_TEXTBOX_HPP
```

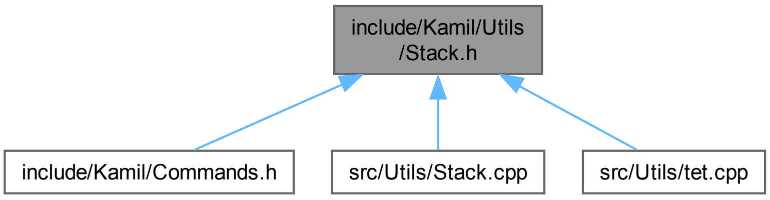
8.17 include/Kamil/Utils/Stack.h File Reference

```
#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.

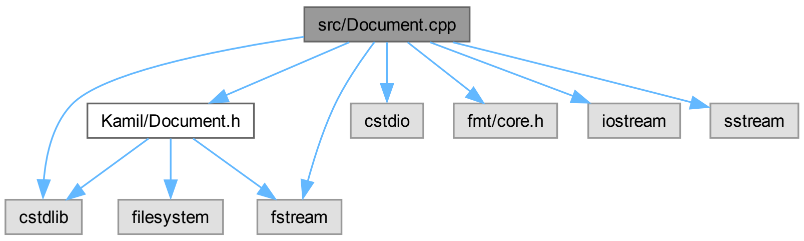
```
#ifndef KAMIL_STACK_H
#define KAMIL_STACK_H

#include <iostream>
#include <string>

namespace Command{

// Dynamic stack array
template<typename T>
class Stack{
public:
    Stack(int);
    int getMax()const;
    void printStack()const;
    int pop();
    int push(T);
    void extend(int);
private:
    int max_size{};
    T* stack_array{new T[max_size]};
    T* SP = &stack_array[max_size];
    int SP_pos = max_size;
};

#endif

} // 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:

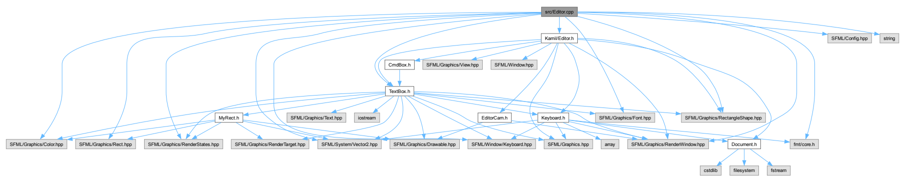


8.21 src/Editor.cpp File Reference

```
#include <Kamil/Editor.h>
#include <Kamil/TextBox.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>
```

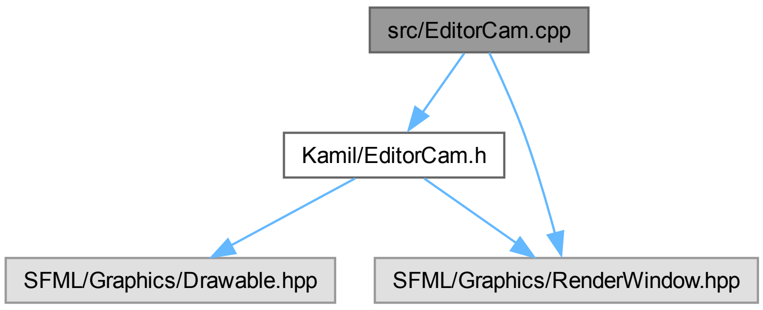
Include dependency graph for Editor.cpp:



8.22 src/EditorCam.cpp File Reference

```
#include <Kamil/EditorCam.h>
#include <SFML/Graphics/RenderWindow.hpp>
```

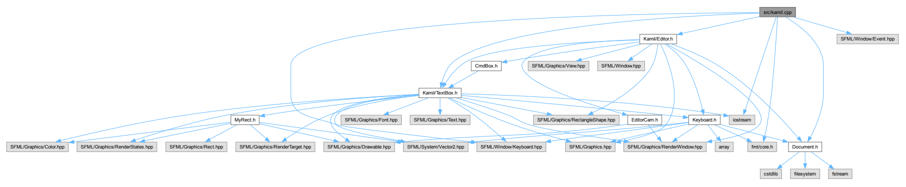
Include dependency graph for EditorCam.cpp:



8.23 src/kamil.cpp File Reference

```
#include "Kamil/TextBox.h"
#include <Kamil/Editor.h>
#include <SFML/Window/Event.hpp>
#include <SFML/Window/Keyboard.hpp>
#include <fmt/core.h>
#include <iostream>
#include <Kamil/Document.h>
```

Include dependency graph for kamil.cpp:



Functions

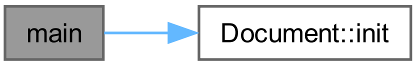
- int main (int argc, char ∗argv[ ])

8.23.1 Function Documentation

8.23.1.1 main()

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

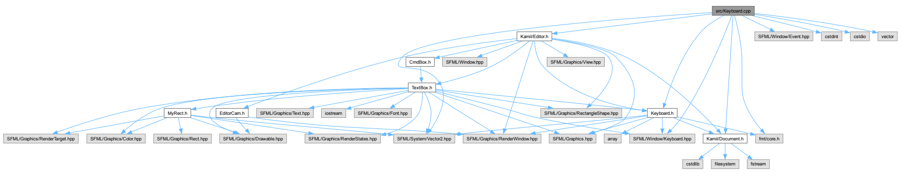
Here is the call graph for this function:



8.24 src/Keyboard.cpp File Reference

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

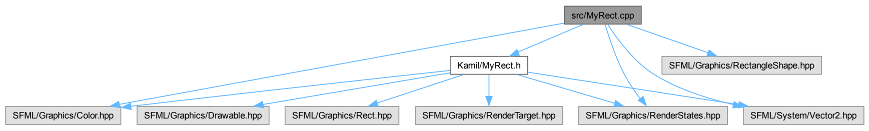
Include dependency graph for Keyboard.cpp:



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>
```

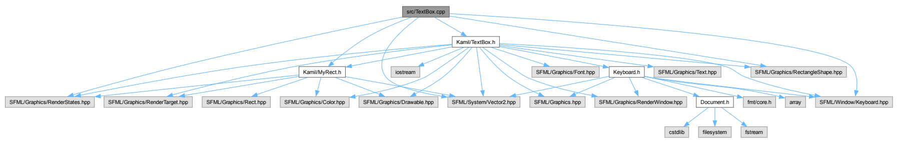
Include dependency graph for MyRect.cpp:


src/MyRect.cpp
Kamil/MyRect.h
SFML/Graphics/RectangleShape.hpp
SFML/Graphics/Color.hpp
SFML/Graphics/Drawable.hpp
SFML/Graphics/Rect.hpp
SFML/Graphics/RenderTarget.hpp
SFML/Graphics/RenderStates.hpp
SFML/System/Vector2.hpp

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>
```

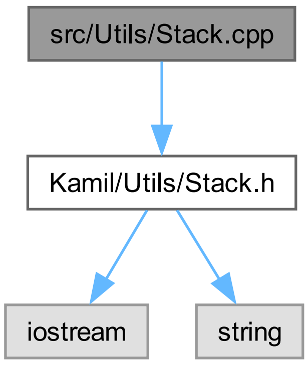
Include dependency graph for TextBox.cpp:



8.27 src/Utils/Stack.cpp File Reference

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

Include dependency graph for Stack.cpp:



Namespaces

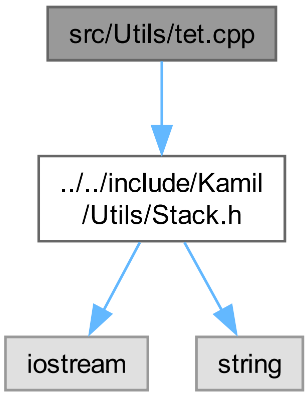
- namespace Command

  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

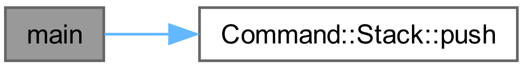
- int main ()

8.28.1 Function Documentation

8.28.1.1 main()

```
int main ( )
```

Here is the call graph for this function:



Index