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
1.1.3 Limitations	4
1.1.4 Design	5
1.1.4.1 Design Choices	6
2 Namespace Index	7
2.1 Namespace List	7
3 Hierarchical Index	9
3.1 Class Hierarchy	9
4 Class Index	11
4.1 Class List	11
5 File Index	13
5.1 File List	13
6 Namespace Documentation	15
6.1 Command Namespace Reference	15
6.1.1 Detailed Description	15
6.2 KEYS Namespace Reference	15
6.2.1 Detailed Description	15
6.2.2 Enumeration Type Documentation	15
6.2.2.1 anonymous enum	15
7 Class Documentation	17
7.1 CmdBox Class Reference	17
7.1.1 Detailed Description	20
7.1.2 Member Function Documentation	21
7.1.2.1 TextBox() [1/2]	21
7.1.2.2 TextBox() [2/2]	21
7.2 Document Class Reference	21
7.2.1 Detailed Description	23
7.2.2 Constructor & Destructor Documentation	23
7.2.2.1 Document() [1/2]	23
7.2.2.2 Document() [2/2]	23
7.2.3 Member Function Documentation	24
7.2.3.1 createDir()	24
7.2.3.2 createFile()	24
7.2.3.3 docHasText()	24
7.2.3.4 getAbsPath()	25

7.2.3.5 getLineCount()	25
7.2.3.6 getRelPath()	26
7.2.3.7 hasChanged()	26
7.2.3.8 init() [1/2]	26
7.2.3.9 init() [2/2]	27
7.2.3.10 readFile()	27
7.2.3.11 saveFile() [1/2]	28
7.2.3.12 saveFile() [2/2]	28
7.2.3.13 setBuffInfo()	29
7.2.3.14 setChange()	29
7.2.4 Member Data Documentation	30
7.2.4.1 absPath	30
7.2.4.2 buffInfo	30
7.2.4.3 docChanged	30
7.2.4.4 relPath	30
7.3 Editor Class Reference	31
7.3.1 Detailed Description	32
7.3.2 Constructor & Destructor Documentation	32
7.3.2.1 Editor()	32
7.3.2.2 ~ Editor()	32
7.3.3 Member Function Documentation	32
7.3.3.1 draw()	33
7.3.3.2 handleEvent()	34
7.3.3.3 makeLineNum()	34
7.3.4 Member Data Documentation	35
7.3.4.1 camera	35
7.3.4.2 cbox	35
7.3.4.3 doc	35
7.3.4.4 event	35
7.3.4.5 kb	35
7.3.4.6 lineBox	35
7.3.4.7 loadFromFile	35
7.3.4.8 textBox	36
7.3.4.9 window	36
7.4 EditorCam Class Reference	36
7.4.1 Constructor & Destructor Documentation	39
7.4.1.1 EditorCam()	39
7.4.2 Member Function Documentation	39
7.4.2.1 draw()	39
7.4.2.2 getBottomLimitPx()	40
7.4.2.3 getLineHeight()	40
7.4.2.4 getRightLimitPx()	41

7.4.2.5 rotateLeft()	. 41
7.4.2.6 rotateRight()	. 42
7.4.2.7 scrollDown()	. 42
7.4.2.8 scrollLeft()	. 42
7.4.2.9 scrollRight()	. 43
7.4.2.10 scrollTo()	. 43
7.4.2.11 scrollUp()	. 44
7.4.2.12 setCameraBounds()	. 44
7.4.2.13 zoomln()	. 45
7.4.2.14 zoomOut()	. 45
7.4.3 Member Data Documentation	. 45
7.4.3.1 bottomLimitPx	. 45
7.4.3.2 camera	. 45
7.4.3.3 deltaRotation	. 46
7.4.3.4 deltaScroll	. 46
7.4.3.5 deltaZoomln	. 46
7.4.3.6 deltaZoomOut	. 46
7.4.3.7 lineHeight	. 46
7.4.3.8 marginXOffset	. 46
7.4.3.9 rightLimitPx	. 46
7.4.3.10 window	. 47
7.5 Keyboard Class Reference	. 47
7.5.1 Detailed Description	. 48
7.5.2 Constructor & Destructor Documentation	. 49
7.5.2.1 Keyboard()	. 49
7.5.3 Member Function Documentation	. 49
7.5.3.1 backSpace()	. 49
7.5.3.2 getBounds()	. 50
7.5.3.3 getCmdTextEntered()	. 50
7.5.3.4 getLineNumber()	. 51
7.5.3.5 getTextEntered()	. 51
7.5.3.6 handleCmdKeyEvent()	. 51
7.5.3.7 handleKeyEvent()	. 52
7.5.3.8 handleMouseEvent()	. 53
7.5.3.9 isCmdTextEntered()	. 54
7.5.3.10 isKeyPressed()	. 54
7.5.3.11 isTextDeleted()	. 55
7.5.3.12 isTextEntered()	. 55
7.5.3.13 kbrCmd()	. 56
7.5.3.14 setCmdTextEntered()	. 56
7.5.3.15 setTextEntered()	. 57
7.5.4 Member Data Documentation	. 57

7.5.4.1 bounds	. 57
7.5.4.2 ctDeleted	. 57
7.5.4.3 ctEntered	. 57
7.5.4.4 tDeleted	. 58
7.5.4.5 tEntered	. 58
7.5.4.6 window	. 58
7.6 MyRect Class Reference	. 58
7.6.1 Detailed Description	. 61
7.6.2 Constructor & Destructor Documentation	. 61
7.6.2.1 MyRect() [1/2]	. 61
7.6.2.2 MyRect() [2/2]	. 61
7.6.3 Member Function Documentation	. 61
7.6.3.1 draw()	. 61
7.6.3.2 getPos()	. 62
7.6.3.3 getSize()	. 62
7.6.3.4 setFillColour()	. 63
7.6.3.5 setPosition()	. 63
7.6.3.6 setSize()	. 64
7.6.4 Member Data Documentation	. 64
7.6.4.1 fillColour	. 64
7.6.4.2 fRect	. 64
7.6.4.3 outlineColour	. 64
7.6.4.4 outlineThicknes	. 64
7.6.4.5 pos	. 64
7.6.4.6 size	. 65
7.7 Command::Stack< T > Class Template Reference	. 65
7.7.1 Constructor & Destructor Documentation	. 66
7.7.1.1 Stack()	. 66
7.7.2 Member Function Documentation	. 66
7.7.2.1 extend()	. 66
7.7.2.2 getMax()	. 66
7.7.2.3 pop()	. 66
7.7.2.4 printStack()	. 67
7.7.2.5 push() [1/2]	. 67
7.7.2.6 push() [2/2]	. 67
7.7.3 Member Data Documentation	. 67
7.7.3.1 max_size	. 67
7.7.3.2 SP	. 67
7.7.3.3 SP_pos	. 68
7.7.3.4 stack_array	. 68
7.8 TextBox Class Reference	. 68
7.8.1 Detailed Description	. 71

72
72
73
73
73
73
74
74
74
75
75
75
75
77
77
78
78
78
78
78
78
78
78
79
79
80
80
80
81
82
82
83
84
84
84
85
86
86
86

8.13 include/Kamil/MyRect.h File Reference	89
8.13.1 Detailed Description	90
8.14 MyRect.h	90
8.15 include/Kamil/TextBox.h File Reference	90
8.16 TextBox.h	91
8.17 include/Kamil/Utils/Stack.h File Reference	92
8.18 Stack.h	93
8.19 README.md File Reference	94
8.20 src/Document.cpp File Reference	94
8.21 src/Editor.cpp File Reference	94
8.22 src/EditorCam.cpp File Reference	95
8.23 src/kamil.cpp File Reference	95
8.23.1 Function Documentation	96
8.23.1.1 main()	96
8.24 src/Keyboard.cpp File Reference	96
8.25 src/MyRect.cpp File Reference	97
8.26 src/TextBox.cpp File Reference	97
8.27 src/Utils/Stack.cpp File Reference	97
8.28 src/Utils/tet.cpp File Reference	98
8.28.1 Function Documentation	99
8.28.1.1 main()	99
Index 1	101

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

Generated by Doxygen

2 Kamil Editor

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

Figure 1.1 My Neovim

```
#define DEFS "this is a basic vim"
#include <iostream>
int main(){
        std::cout << DEFS;
        return D;
}
~</pre>
```

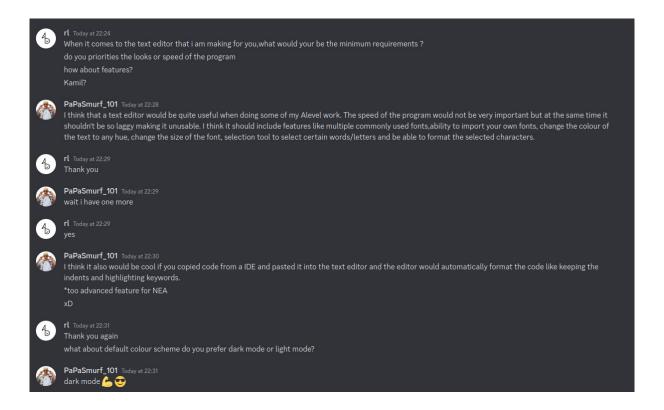
Figure 1.2 My Vim

(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.

1.1 Analysis 3



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:

4 Kamil Editor

- · 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 1 month and 2 weeks 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

1.1 Analysis 5

```
(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)

6 Kamil Editor

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

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Commar	nd	
	A stack in the Command namespace	15
KEYS		
	An enum for Keyboard characters in hex form	15

8 Namespace Index

Hierarchical Index

3.1 Class Hierarchy

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

ıment	2
rawable	
ditorCam	
MyRect	5
TextBox	6
CmdBox	1
r	3
oatRect	
lyRect	
oard	4
$mand::Stack < T > \dots \dots$	6

10 Hierarchical Index

Class Index

4.1 Class List

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

CmdBox		
	Class to handle the command TextBox	17
Docume	nt .	
	Document class	21
Editor		
	Class that handles and draws everything in the Editor	31
EditorCa	ım	36
Keyboard	d	
	A class to handle Keyboard input	47
MyRect		
	Gives extra functionality to FloatRect	58
Commar	nd::Stack < T >	65
TextBox		
	A class that makes a Textbox in SFML	68

12 Class Index

File Index

5.1 File List

Here is a list of all files with brief descriptions:

include/Kamil/CmdBox.h	'9
include/Kamil/Commands.h	30
include/Kamil/Document.h	
Interface file for the Document class	31
include/Kamil/Editor.h	
Interface file for the Editor class	33
include/Kamil/EditorCam.h	
Implementation of EditorCam class	34
include/Kamil/Keyboard.h	
Interface file for Keyboard.h	36
include/Kamil/MyRect.h	
Interface file for the MyRect class	39
include/Kamil/TextBox.h	90
include/Kamil/Utils/Stack.h	92
src/Document.cpp	94
src/Editor.cpp	94
src/EditorCam.cpp	95
src/kamil.cpp	95
src/Keyboard.cpp	96
src/MyRect.cpp	97
src/TextBox.cpp	97
src/Utils/Stack.cpp	97
src/Litils/tet.cnn	ลล

14 File Index

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	

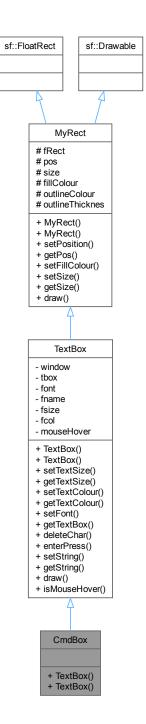
Class Documentation

7.1 CmdBox Class Reference

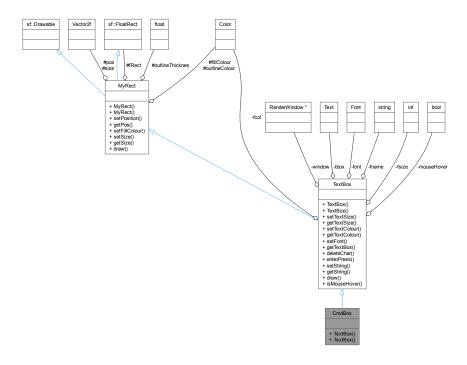
Class to handle the command TextBox.

#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]

Using teh Parent class constructor.

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

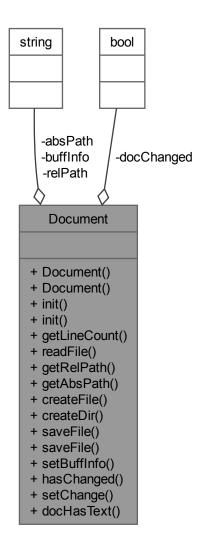
• include/Kamil/CmdBox.h

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

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:

Editor::makeLineNum

Document::getLineCount

Editor::draw

7.2.3.6 getRelPath() std::string Document::getRelPath () get the relative path Parameters void Returns

7.2.3.7 hasChanged()

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

string for relative path

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

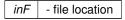
Here is the caller graph for this function:



7.2.3.9 init() [2/2]

initialise the file

Parameters



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]

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()

save file infor to buffer

Parameters



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

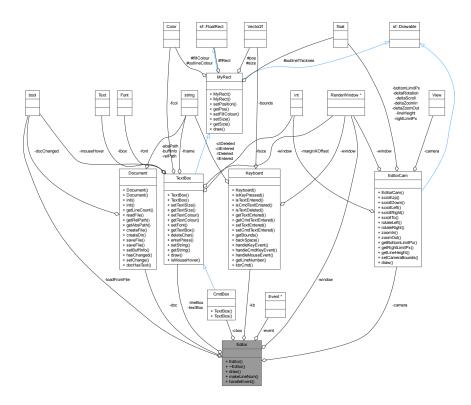
7.3 Editor Class Reference 31

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 \sim Editor()

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

Destructor for Editor class.

7.3.3 Member Function Documentation

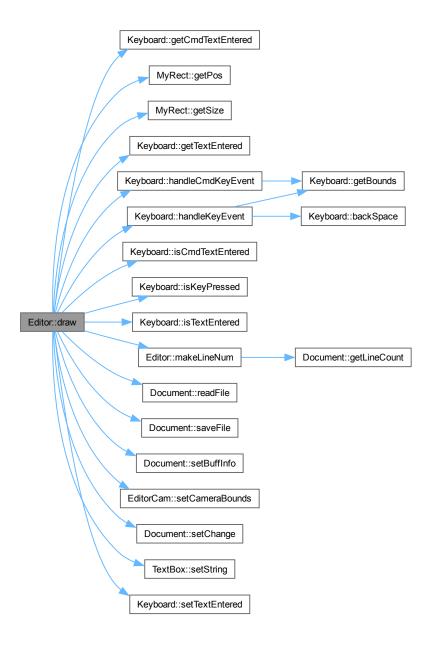
7.3 Editor Class Reference 33

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 Editor Class Reference 35

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

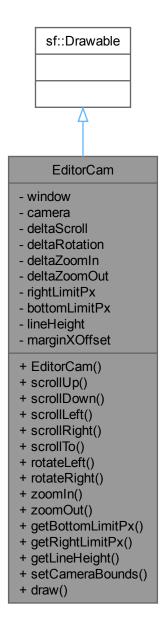
The documentation for this class was generated from the following files:

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

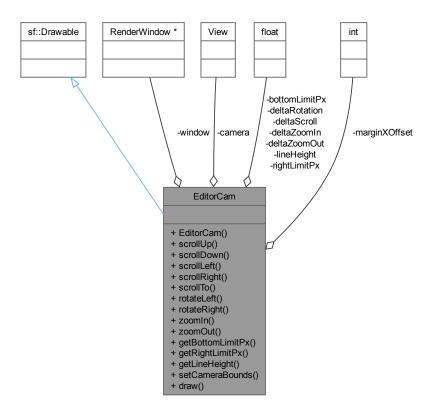
7.4 EditorCam Class Reference

#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()

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:

EditorCam::getBottomLimitPx

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:

EditorCam::getRightLimitPx

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

Returns

void

7.4.2.9 scrollRight()

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

@breif move camera right

Parameters



Returns

void

Here is the call graph for this function:

```
EditorCam::getRightLimitPx
```

7.4.2.10 scrollTo()

```
void EditorCam::scrollTo ( \label{eq:float x, float y, float y} 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()

set camera bounds

Parameters

	int	- width
	int	- height

Returns

void

Here is the caller graph for this function:



7.4.2.13 zoomln()

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:

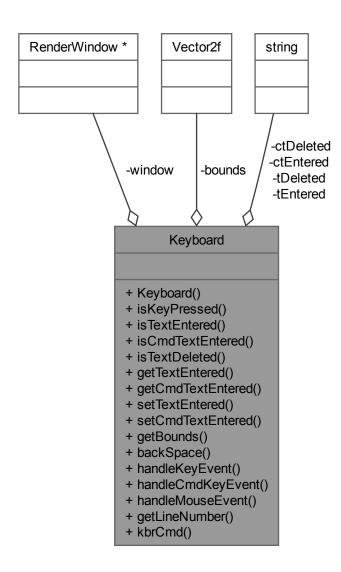
- include/Kamil/EditorCam.h
- src/EditorCam.cpp

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()

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



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

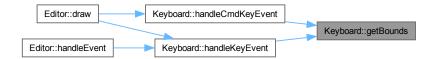
Parameters



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



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()

handle keyboard events

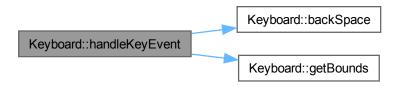
Parameters

event - to get text entered from events

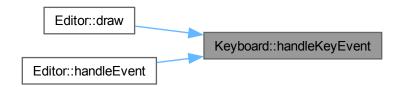
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



7.5.3.8 handleMouseEvent()

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:



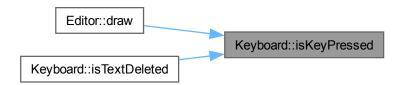
7.5.3.10 isKeyPressed()

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:

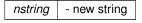


7.5.3.13 kbrCmd()

7.5.3.14 setCmdTextEntered()

sets text

Parameters



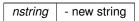
Returns

void

7.5.3.15 setTextEntered()

sets text

Parameters



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:

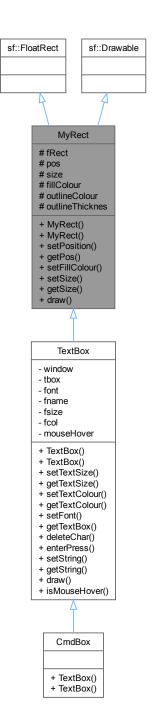
- include/Kamil/Keyboard.h
- src/Keyboard.cpp

7.6 MyRect Class Reference

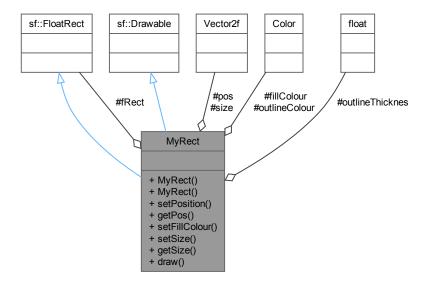
gives extra functionality to FloatRect

```
#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]

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()

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 $My \leftarrow Rect.draw(window)$ 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



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()

set the fill colour of the rect

Parameters

```
sf::Color colour
```

Returns

void

7.6.3.5 setPosition()

sets the position of rect

Parameters

sf::Vector2f pos

7.6.3.6 setSize()

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

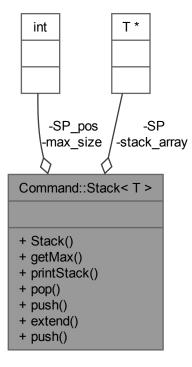
The documentation for this class was generated from the following files:

- 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)

66 Class Documentation

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()

7.7.2 Member Function Documentation

7.7.2.1 extend()

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]

7.7.2.6 push() [2/2]

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]
```

68 Class Documentation

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:

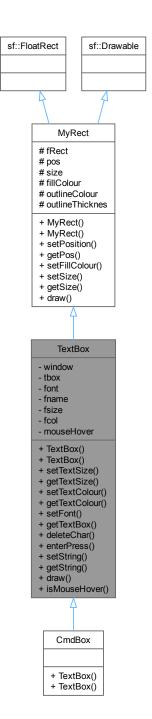
- include/Kamil/Commands.h
- include/Kamil/Utils/Stack.h
- src/Utils/Stack.cpp

7.8 TextBox Class Reference

A class that makes a Textbox in SFML.

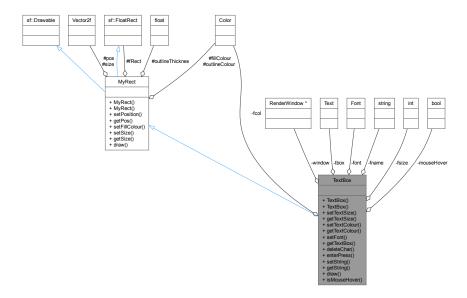
```
#include <TextBox.h>
```

Inheritance diagram for TextBox:



70 Class Documentation

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

72 Class Documentation

7.8.2.1 TextBox() [1/2]

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

void

Returns

74 Class Documentation

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()

set what font you use

Parameters

font file dir of font

Returns

void

7.8.3.10 setString()

76 Class Documentation

Sets the string.

Parameters



Returns

void

Here is the caller graph for this function:



7.8.3.11 setTextColour()

Set the colour of the text.

Parameters

fill font colour

Returns

void

7.8.3.12 setTextSize()

Set the size of the text.

Parameters

size text size

78 Class Documentation

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

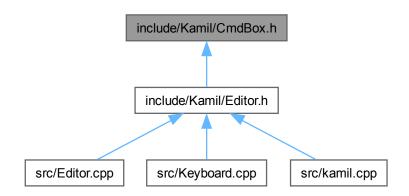
File Documentation

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.

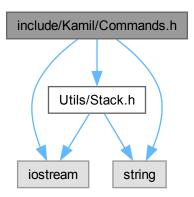
8.2 CmdBox.h

Go to the documentation of this file.

```
00001 #ifndef KAMIL_CMDBOX_H
00002 #define KAMIL_CMDBOX_H
00003
00014 #include "TextBox.h"
00015
00019 class CmdBox : public TextBox {
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
00006
00007 #include "Utils/Stack.h"
```

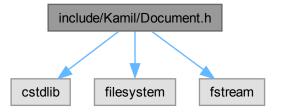
```
00009 namespace Command {
 00010 // template <typename T>
00011 // class Node{ // used for LinkedList
00012 // public:
00013 // Node();
00014 // Node(T);
                        Node();
Node(T);
T data;
Node* next;
 00015 //
 00016 //
 00017 // };
 00018
 00018
00019 // template <typename T>
00020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

8.5 include/Kamil/Document.h File Reference

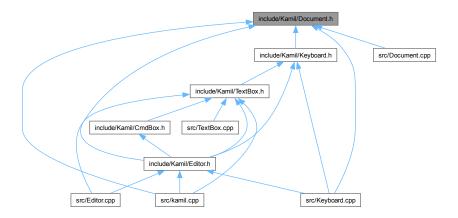
Interface file for the **Document** class.

```
#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.

```
00001 #ifndef KAMIL_DOCUMENT_H
00002 #define KAMIL_DOCUMENT_H
00003
00014 #include <cstdlib>
00015 #include <filesystem>
00016 #include <fstream>
00017
00021 class Document {
00022 public:
00026
         Document();
00027
00032
         Document(std::string fileP);
00033
00039
         void init();
00040
00046
         void init(std::string inF);
00047
00053
         int getLineCount();
00054
00060
         std::string readFile();
00061
00067
         std::string getRelPath();
00068
00074
         std::string getAbsPath();
00075
00081
         void createFile();
00082
```

```
00088
        void createDir();
00089
00095
        bool saveFile(const std::string &filename);
00096
00102
        bool saveFile();
00103
00109
        void setBuffInfo(std::string info);
00110
00116
        bool hasChanged();
00117
00123
        void setChange();
00124
00130
        bool docHasText();
00131
00132
        // void addTextToPos(std::string txt, int pos);
00133
00134 private:
00135
        std::string relPath;
00136
       std::string absPath;
00138 std::string buffInfo;
00140 bool docChanged;
00141 };
00142 #endif // KAMIL_DOCUMENT_H
```

8.7 include/Kamil/Editor.h File Reference

Interface file for the Editor class.

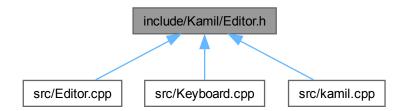
```
#include <SFML/Graphics.hpp>
#include <SFML/Graphics/RectangleShape.hpp>
#include <SFML/Graphics/RenderWindow.hpp>
#include <SFML/Graphics/View.hpp>
#include <SFML/Window.hpp>
#include "CmdBox.h"
#include "Document.h"
#include "EditorCam.h"
#include "TextBox.h"

#arkelded "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

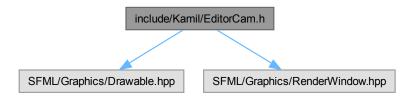
Go to the documentation of this file.

```
00001 #ifndef KAMIL_EDITOR_WINDOW_HPF
00002 #define KAMIL_EDITOR_WINDOW_HPP
00003
00014 #include <SFML/Graphics.hpp>
00015 #include <SFML/Graphics/RectangleShape.hpp>
00016 #include <SFML/Graphics/RenderWindow.hpp>
00017 #include <SFML/Graphics/View.hpp>
00018 #include <SFML/Window.hpp>
00019
00020 #include "CmdBox.h"
00021 #include "Document.h"
00022 #include "EditorCam.h"
00023 #include "Keyboard.h"
00024 #include "TextBox.h"
00025
00029 class Editor {
00030 public:
00037
        Editor(sf::RenderWindow *window, sf::Event *event, Document *doc);
00038
00042
        ~Editor();
00043
00049
        void draw();
00050
00055
        void makeLineNum();
00056
00063
        void handleEvent();
00064
00065 private:
00066 Document *doc;
00067
        TextBox *textBox;
        CmdBox *cbox;
00068
00069
        sf::RenderWindow *window;
00070
        sf::Event *event;
00071
         TextBox lineBox;
00072
        EditorCam camera;
Keyboard kb;
00073
00074
        bool loadFromFile;
00075 };
00077 #endif // KAMIL_EDITOR_WINDOW_HPP
```

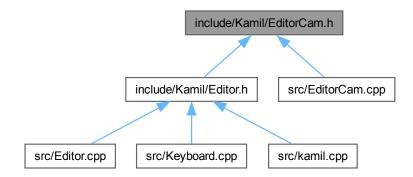
8.9 include/Kamil/EditorCam.h File Reference

Implementation of EditorCam class.

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



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. 00001 #ifndef KAMIL_EDITOR_CAM_H

```
00001 #ifndef KAMIL_EDITOR_CAM_H 00002 #define KAMIL_EDITOR_CAM_H
00003
00004
00015 #include <SFML/Graphics/Drawable.hpp>
00016 #include <SFML/Graphics/RenderWindow.hpp>
00017
00018 class EditorCam : public sf::Drawable {
00019 public:
00020
00029
       EditorCam(sf::RenderWindow *window, float deltaScroll, float deltaRotation,
00030
                  float deltaZoomIn, float deltaZoomOut);
00031
00037
       void scrollUp();
00038
00044
       void scrollDown();
00045
00051
       void scrollLeft();
00052
       void scrollRight();
00058
00059
00066
       void scrollTo(float x, float y);
00067
00073
       void rotateLeft();
00074
08000
       void rotateRight();
00081
00087
       void zoomIn();
00088
00094
       void zoomOut();
00095
00096
00102
       float getBottomLimitPx();
00103
00109
       float getRightLimitPx();
00110
00116
       int getLineHeight();
00117
       void setCameraBounds(int width, int height);
00124
00125
00132
       void draw(sf::RenderTarget &target, sf::RenderStates states) const override;
00133
00134 private:
       sf::RenderWindow *window;
00135
00136
       sf::View camera;
       float deltaScroll:
00137
00138
       float deltaRotation;
00139
       float deltaZoomIn, deltaZoomOut;
00140
       float rightLimitPx;
00141
       float bottomLimitPx;
00142
       float lineHeight;
00143
       int marginXOffset;
00144 };
00145
00146 #endif // KAMIL_EDITOR_CAM_H
```

8.11 include/Kamil/Keyboard.h File Reference

Interface file for Keyboard.h.

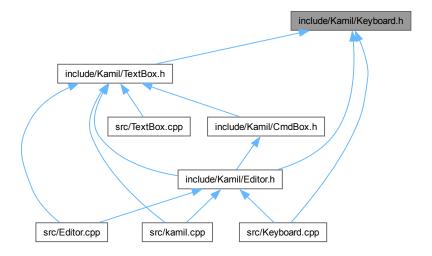
```
#include <SFML/Graphics.hpp>
#include <SFML/Graphics/RenderWindow.hpp>
#include <SFML/System/Vector2.hpp>
#include <SFML/Window/Keyboard.hpp>
#include "Document.h"
#include <fmt/core.h>
```

#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.

```
00001 #ifndef KAMIL_KEYBOARD_H
00002 #define KAMIL_KEYBOARD_H
00003
00014 #include <SFML/Graphics.hpp>
00015 #include <SFML/Graphics/RenderWindow.hpp>
00016 #include <SFML/System/Vector2.hpp>
00017 #include <SFML/Window/Keyboard.hpp>
00018
00019 #include "Document.h"
00020 #include <fmt/core.h>
00021
00022 #include <array>
00023
00027 namespace KEYS {
00028 enum {
        ESCAPE = 0x1B,
00029
00030
        ENTER = 0xD,
        BS = 0x8,
00031
00032
        Shift_A = 0x41,
00033
        CTRL = 0x11,
       DELETE = 0x7f,
00034
00035 };
00036 }
00037
00038 #ifdef USE_KEYS
00039
00040 #define "LControl" sf::Keyboard::KEYS::LControl
00041
00042 #endif
00043
00047 class Keyboard {
00048 public:
00054
        Keyboard(sf::RenderWindow *win, Document *doc, sf::Vector2f bounds);
00055
00060
        bool isKeyPressed(sf::Keyboard::Key);
00061
00067
        bool isTextEntered();
00068
00074
        bool isCmdTextEntered();
00075
00081
        bool isTextDeleted();
00082
00088
        std::string getTextEntered();
00089
00095
        std::string getCmdTextEntered();
00096
00102
        void setTextEntered(std::string);
00103
00109
        void setCmdTextEntered(std::string);
00110
00116
        sf::Vector2f getBounds() const;
00117
00123
        void backSpace();
00124
00130
        void handleKeyEvent(sf::Event &event);
00131
00137
        void handleCmdKeyEvent();
00138
00144
        void handleMouseEvent(sf::Event &event); // not implemented yet
00145
00151
        int getLineNumber();
00152
00153
        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);
00154
00155
00156
00157
00158
```

```
00159
00160  // get position in text
00161
00162 private:
00163  sf::RenderWindow *window;
00164  // Document* doc;
00165  sf::Vector2f bounds;
00166  std::string tEntered;
00167  std::string tDeleted;
00169  std::string ctEntered;
00170  std::string ctDeleted;
00171 };
00172 #endif // KAMIL_KEYBOARD_H
```

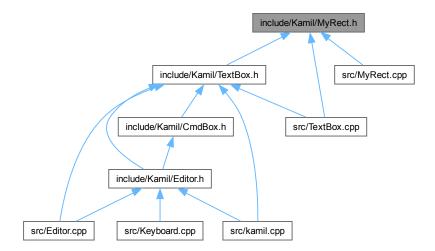
8.13 include/Kamil/MyRect.h File Reference

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_cobject.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
00027 #include <SFML/Graphics/Color.hpp>
00028 #include <SFML/Graphics/Drawable.hpp>
00029 #include <SFML/Graphics/Rect.hpp>
00030 #include <SFML/Graphics/RenderStates.hpp>
00031 #include <SFML/Graphics/RenderTarget.hpp>
00032 #include <SFML/System/Vector2.hpp>
00033
00041 class MyRect : public sf::FloatRect, public sf::Drawable {
00042 public:
00051
       MyRect(sf::Vector2f pos, sf::Vector2f size, sf::Color fillColour,
00052
               sf::Color outlineColour, float outlineThicknes);
00053
       MyRect();
00054
00059
        void setPosition(sf::Vector2f pos);
00060
00066
       sf::Vector2f getPos() const;
00067
00073
        void setFillColour(sf::Color colour);
00074
08000
        void setSize(sf::Vector2f size);
00081
00087
        sf::Vector2f getSize() const;
88000
00099
        void draw(sf::RenderTarget &target, sf::RenderStates states) const override;
00100
00101 protected:
00102
      sf::FloatRect fRect;
00103
        sf::Vector2f pos;
00104
        sf::Vector2f size;
sf::Color fillColour;
00105
00106
        sf::Color outlineColour;
00107
        float outlineThicknes;
00108 };
00109
00110 #endif // KAMIL_MYRECT_H
```

8.15 include/Kamil/TextBox.h File Reference

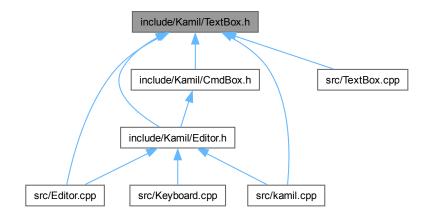
```
#include <SFML/Graphics.hpp>
#include <SFML/Graphics/Color.hpp>
#include <SFML/Graphics/Drawable.hpp>
#include <SFML/Graphics/Font.hpp>
#include <SFML/Graphics/RectangleShape.hpp>
#include <SFML/Graphics/RenderStates.hpp>
```

8.16 TextBox.h 91

```
#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:
```

SECONOLOGICA STATE SECONOLOGICA

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/RenderTarget.hpp>
```

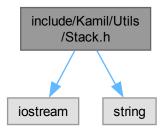
```
00020 #include <SFML/Graphics/Text.hpp>
00021 #include <SFML/System/Vector2.hpp>
00022 #include <SFML/Window/Keyboard.hpp>
00023 #include <iostream>
00024
00025 #include "Keyboard.h"
00026 #include "MyRect.h"
00027
00028 /*
00029 *
00030 * TODO:
00031 *
             Make a RectangleShape that acts as the bounds of the TextBox
00031 * 00033 *
              then add limits to the textbox so it stays in the limits
00034 *
              Add the Keybord manager class here and use its methods
00035 *
00036 */
             to handle the key events
00037
00044 class TextBox : public MyRect {
00045 public:
      00057
00058
00059
               float thicc);
00060
       TextBox();
00061
00067
       void setTextSize(int size);
00068
00074
       int getTextSize() const;
00075
00081
       void setTextColour(sf::Color colour);
00082
00088
       sf::Color getTextColour() const;
00089
00095
       void setFont(sf::Font &font);
00096
00102
       sf::Text getTextBox() const;
00103
00109
       void deleteChar();
00110
00116
       void enterPress();
00117
00123
       void setString(std::string nstring);
00124
00130
       std::string getString() const;
00131
00139
       void draw(sf::RenderTarget &target, sf::RenderStates states) const override;
00140
00145
       bool isMouseHover();
00146
00147 private:
       sf::RenderWindow *window;
00148
00149
       sf::Text tbox{};
00150
       sf::Font font{};
00151
       std::string fname{};
00152
       int fsize{};
00153
       sf::Color fcol();
       bool mouseHover;
00155 };
00156 #endif // KAMIL_TEXTBOX_HPP
```

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

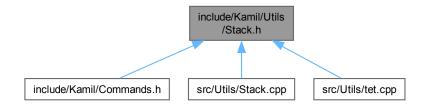
```
#include <iostream>
#include <string>
```

8.18 Stack.h 93

Include dependency graph for Stack.h:



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



Classes

class Command::Stack

Namespaces

• namespace Command A stack in the Command namespace.

Stack.h 8.18

```
Go to the documentation of this file. 00001 #ifndef KAMIL_STACK_H 00002 #define KAMIL_STACK_H
00004 #include <iostream>
00005 #include <string>
00006
00010 namespace Command{
00012 // Dynamic stack array
00013 template<typename T>
```

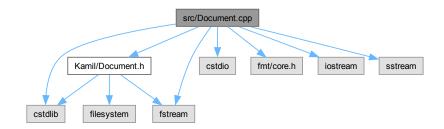
```
00014 class Stack{
00015 public:
00016
           Stack(int);
00017
           int getMax()const;
           void printStack()const;
00018
00019
         int pop();
int push(T);
00020
00021
           void extend(int);
00022 private:
00023
          int max_size{};
          T* stack_array{new T[max_size]};
T* SP = &stack_array[max_size];
00024
00025
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:

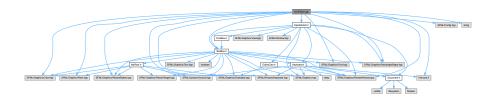


8.21 src/Editor.cpp File Reference

```
#include <Kamil/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>
```

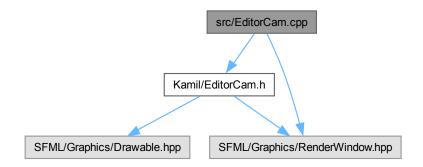
#include <SFML/Window/Keyboard.hpp>
#include <fmt/core.h>
#include <string>

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:
```



8.26 src/TextBox.cpp File Reference

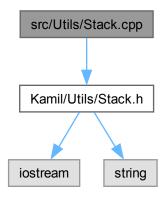
```
#include <Kamil/MyRect.h>
#include <Kamil/TextBox.h>
#include <SFML/Graphics/RectangleShape.hpp>
#include <SFML/Graphics/RenderStates.hpp>
#include <SFML/System/Vector2.hpp>
#include <SFML/Window/Keyboard.hpp>
Include dependency graph for TextBox.cpp:
```



8.27 src/Utils/Stack.cpp File Reference

#include <Kamil/Utils/Stack.h>

Include dependency graph for Stack.cpp:



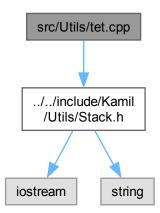
Namespaces

• 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

\sim Editor	TextBox, 73
Editor, 32	deltaRotation
	EditorCam, 45
absPath	deltaScroll
Document, 30	EditorCam, 46
	deltaZoomIn
backSpace	EditorCam, 46
Keyboard, 49	deltaZoomOut
bottomLimitPx	EditorCam, 46
EditorCam, 45	doc
bounds	Editor, 35
Keyboard, 57	docChanged
BS	Document, 30
KEYS, 16	docHasText
buffInfo	Document, 24
Document, 30	Document, 21
	absPath, 30
camera	buffInfo, 30
Editor, 35	createDir, 24
EditorCam, 45	createFile, 24
cbox	,
Editor, 35	docChanged, 30
CmdBox, 17	docHasText, 24
TextBox, 21	Document, 23
Command, 15	getAbsPath, 25
Command::Stack< T >, 65	getLineCount, 25
extend, 66	getRelPath, 26
getMax, 66	hasChanged, 26
max size, 67	init, 26, 27
pop, 66	readFile, 27
printStack, 66	relPath, 30
push, 67	saveFile, 28
SP, 67	setBuffInfo, 29
SP_pos, 67	setChange, 29
Stack, 66	draw
stack_array, 68	Editor, 32
createDir	EditorCam, 39
Document, 24	MyRect, 61
createFile	TextBox, 73
Document, 24	E 19 04
ctDeleted	Editor, 31
Keyboard, 57	\sim Editor, 32
ctEntered	camera, 35
Keyboard, 57	cbox, 35
CTRL	doc, 35
	draw, 32
KEYS, 16	Editor, 32
DELETE	event, 35
KEYS, 16	handleEvent, 33
deleteChar	kb, 35
dolotoonal	lineBox 35

102 INDEX

loadFromFile, 35	Keyboard, 49
makeLineNum, 34	getCmdTextEntered
textBox, 35	Keyboard, 50
window, 36	getLineCount
EditorCam, 36	Document, 25
bottomLimitPx, 45	getLineHeight
camera, 45	EditorCam, 40
deltaRotation, 45	getLineNumber
deltaScroll, 46	Keyboard, 51
deltaZoomln, 46	getMax
deltaZoomOut, 46	Command::Stack <t>, 66</t>
draw, 39	getPos
EditorCam, 39	MyRect, 61
getBottomLimitPx, 40	getRelPath
getLineHeight, 40	Document, 26
getRightLimitPx, 41	getRightLimitPx
lineHeight, 46	EditorCam, 41
marginXOffset, 46	getSize
rightLimitPx, 46	MyRect, 62
rotateLeft, 41	getString
rotateRight, 41	TextBox, 73
scrollDown, 42	getTextBox
scrollLeft, 42	TextBox, 74
scrollRight, 43	getTextColour
scrollTo, 43	TextBox, 74
scrollUp, 44	getTextEntered
setCameraBounds, 44	Keyboard, 51
window, 46	getTextSize
zoomln, 44	TextBox, 74
zoomOut, 45	handla CmdVavEvant
ENTER	handleCmdKeyEvent
KEYS, 16	Keyboard, 51 handleEvent
enterPress	
TextBox, 73	Editor, 33 handleKeyEvent
ESCAPE	Keyboard, 52
KEYS, 16	handleMouseEvent
event	Keyboard, 53
Editor, 35	hasChanged
extend	Document, 26
Command::Stack< T >, 66	Document, 20
fcol	include/Kamil/CmdBox.h, 79, 80
TextBox, 78	include/Kamil/Commands.h, 80
fillColour	include/Kamil/Document.h, 81, 82
MyRect, 64	include/Kamil/Editor.h, 83, 84
fname	include/Kamil/EditorCam.h, 84, 86
TextBox, 78	include/Kamil/Keyboard.h, 86, 88
font	include/Kamil/MyRect.h, 89, 90
TextBox, 78	include/Kamil/TextBox.h, 90, 91
fRect	include/Kamil/Utils/Stack.h, 92, 93
MyRect, 64	init
fsize	Document, 26, 27
TextBox, 78	isCmdTextEntered
	Keyboard, 54
getAbsPath	isKeyPressed
Document, 25	Keyboard, 54
getBottomLimitPx	isMouseHover
EditorCam, 40	TextBox, 75
getBounds	isTextDeleted

INDEX 103

Keyboard, 55	MyRect, 58
isTextEntered	draw, 61
Keyboard, 55	fillColour, 64
	fRect, 64
kamil.cpp	getPos, 61
main, 96	getSize, 62
kb	MyRect, 61
Editor, 35	outlineColour, 64
kbrCmd	outlineThicknes, 64
Keyboard, 56	pos, 64
Keyboard, 47	setFillColour, 63
backSpace, 49	setPosition, 63
bounds, 57	setSize, 63
ctDeleted, 57	size, 64
ctEntered, 57	
getBounds, 49	outlineColour
getCmdTextEntered, 50	MyRect, 64
getLineNumber, 51	outlineThicknes
getTextEntered, 51	MyRect, 64
handleCmdKeyEvent, 51	
handleKeyEvent, 52	pop
handleMouseEvent, 53	Command::Stack< T >, 66
isCmdTextEntered, 54	pos
isKeyPressed, 54	MyRect, 64
isTextDeleted, 55	printStack
isTextEntered, 55	Command::Stack< T >, 66
kbrCmd, 56	push
Keyboard, 49	Command::Stack< T >, 67
setCmdTextEntered, 56	
setTextEntered, 56	readFile
tDeleted, 57	Document, 27
tEntered, 58	README.md, 94
window, 58	relPath
KEYS, 15	Document, 30
BS, 16	rightLimitPx
CTRL, 16	EditorCam, 46
DELETE, 16	rotateLeft
ENTER, 16	EditorCam, 41
ESCAPE, 16	rotateRight
Shift A, 16	EditorCam, 41
Jilli_A, 10	
lineBox	saveFile
Editor, 35	Document, 28
lineHeight	scrollDown
EditorCam, 46	EditorCam, 42
loadFromFile	scrollLeft
Editor, 35	EditorCam, 42
	scrollRight
main	EditorCam, 43
kamil.cpp, 96	scrollTo
tet.cpp, 99	EditorCam, 43
makeLineNum	scrollUp
Editor, 34	EditorCam, 44
marginXOffset	setBuffInfo
EditorCam, 46	Document, 29
max size	setCameraBounds
Command::Stack< T >, 67	EditorCam, 44
mouseHover	setChange
TextBox, 78	Document, 29
TOMBON, TO	•

104 INDEX

setCmdTextEntered	getTextColour, 74
Keyboard, 56	getTextSize, 74
setFillColour	isMouseHover, 75
MyRect, 63	mouseHover, 78
setFont	setFont, 75
TextBox, 75	setString, 75
setPosition	setTextColour, 77
MyRect, 63	setTextSize, 77
setSize	tbox, 78
MyRect, 63	TextBox, 71, 72
setString	window, 78
TextBox, 75	textBox
setTextColour	Editor, 35
TextBox, 77	
setTextEntered	window
	Editor, 36
Keyboard, 56	
setTextSize	EditorCam, 46
TextBox, 77	Keyboard, 58
Shift_A	TextBox, 78
KEYS, 16	
size	zoomln
MyRect, 64	EditorCam, 44
SP	zoomOut
	EditorCam, 45
Command::Stack< T >, 67	
SP_pos	
Command::Stack< T >, 67	
src/Document.cpp, 94	
src/Editor.cpp, 94	
src/EditorCam.cpp, 95	
src/kamil.cpp, 95	
src/Keyboard.cpp, 96	
src/MyRect.cpp, 97	
•	
src/TextBox.cpp, 97	
src/Utils/Stack.cpp, 97	
src/Utils/tet.cpp, 98	
Stack	
Command::Stack< T >, 66	
stack_array	
Command::Stack< T >, 68	
tbox	
TextBox, 78	
tDeleted	
Keyboard, 57	
•	
tEntered	
Keyboard, 58	
tet.cpp	
main, 99	
TextBox, 68	
CmdBox, 21	
deleteChar, 73	
draw, 73	
enterPress, 73	
fcol, 78	
fname, 78	
font, 78	
fsize, 78	
getString, 73	
getTextBox, 74	