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.

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

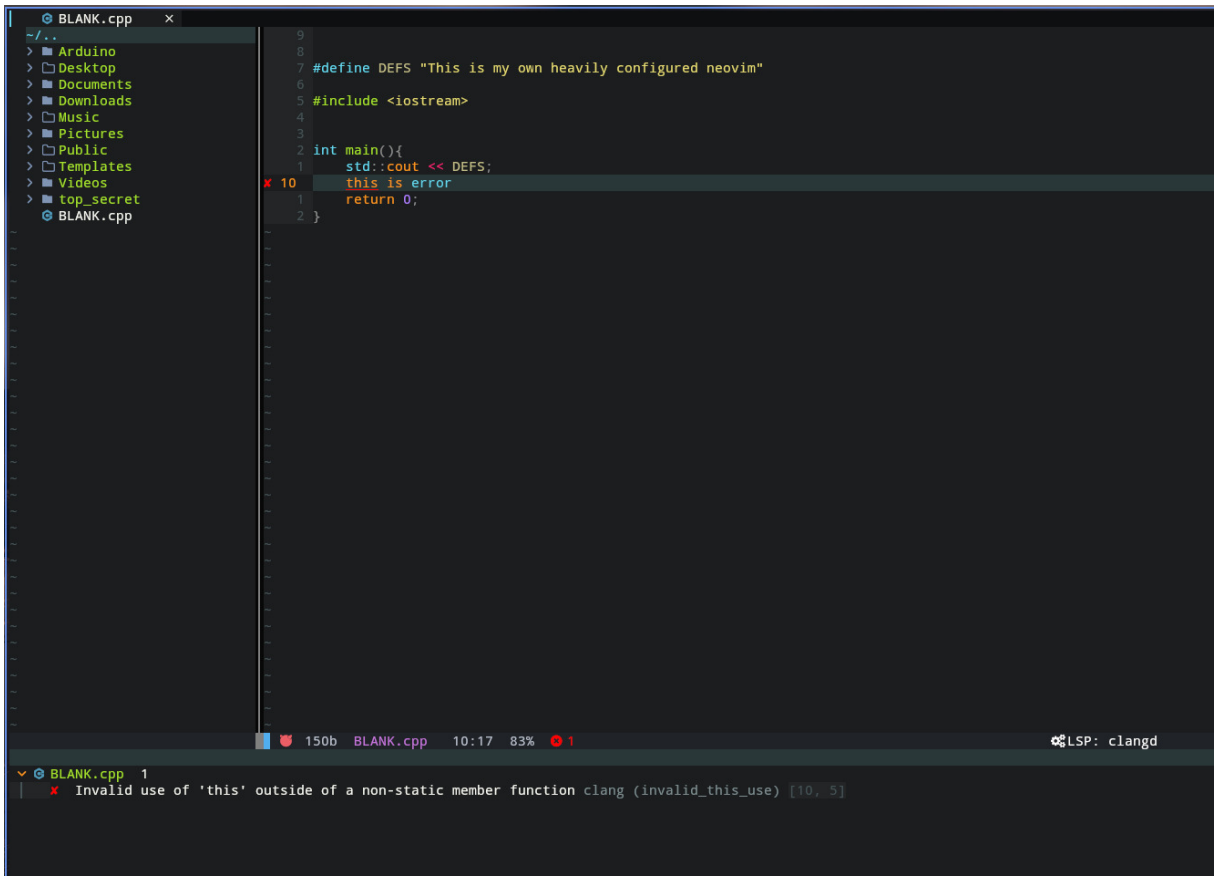
```
1) What is a text editor and how does it differ from an IDE?
2) How do I make a text editor or IDE for kamil
3) How do I make it efficient enough to meet his standards?
```

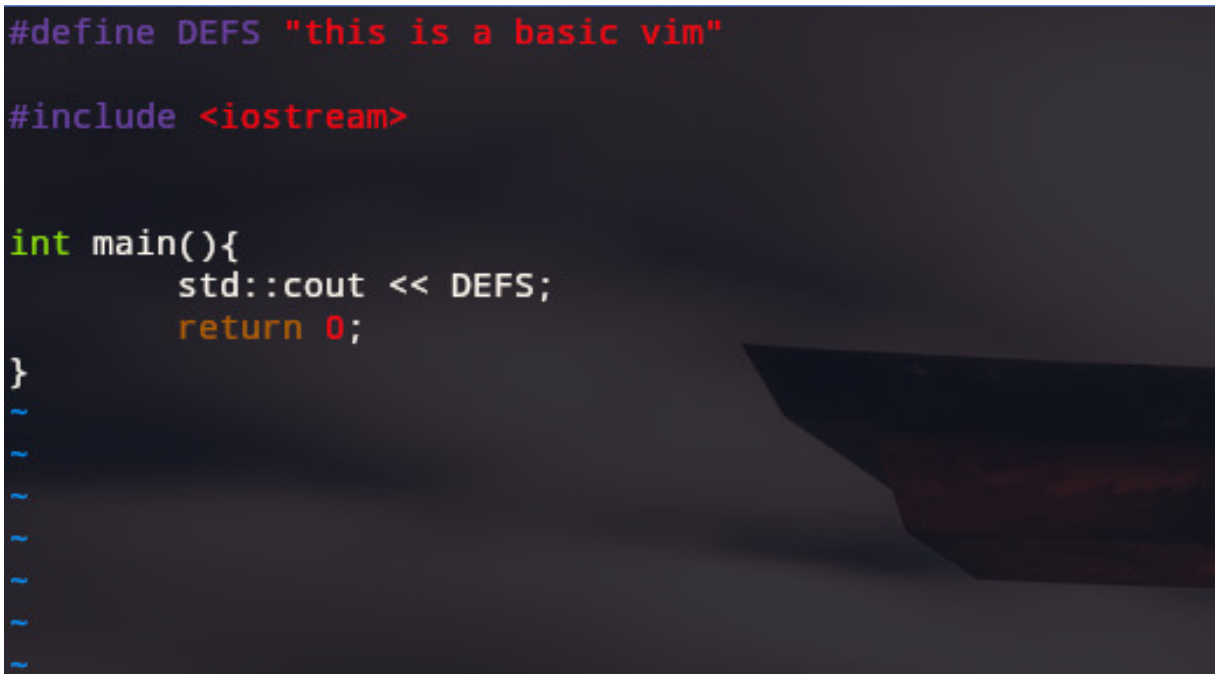
To kick things along I began to do research on Text editors and IDE's and found out that the difference between isnt limited to Operating System platforms or by how much better one is at a specific task but by the features each can do. Text Editors, as the name suggest are specifically desinged for manipulting 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	

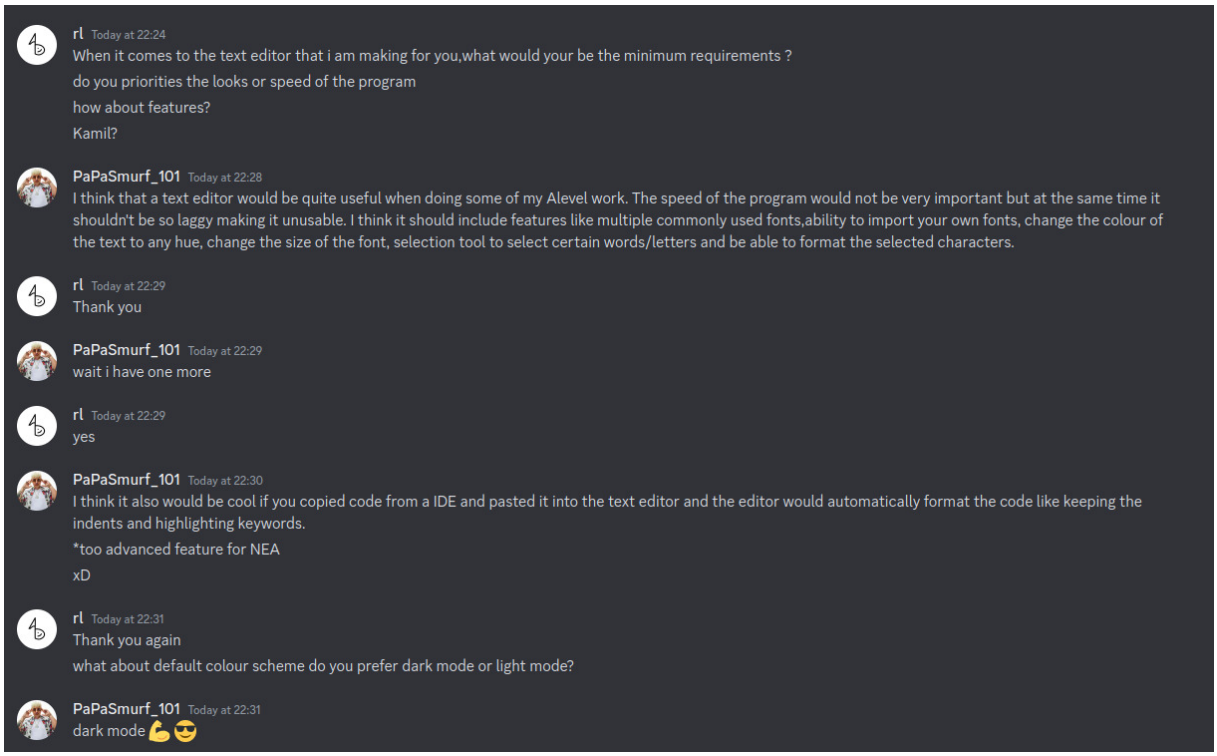
Here are pictures of some text editors and IDE's:





1.1.2 End User needs

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



To conclude the User needs: Kamil would like:

- Not laggy
- 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

1.2 Structure

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

window

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

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

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

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

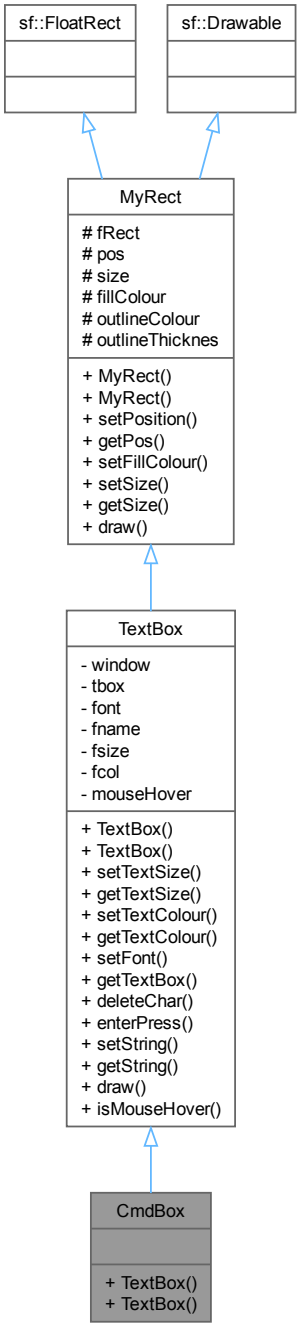
Class Documentation

7.1 CmdBox Class Reference

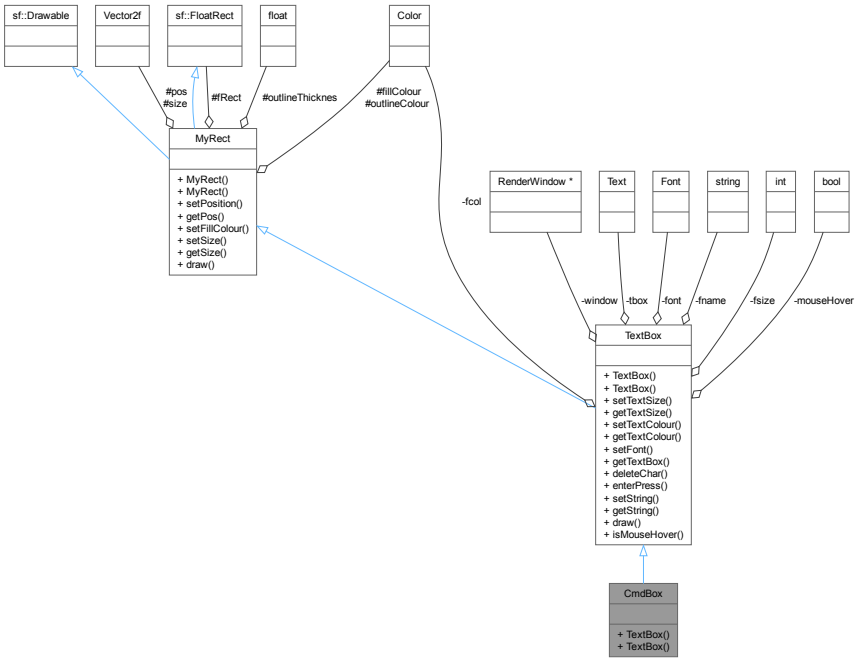
Class to handle the command TextBox.

`#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 main textbox and the cmd textbox in a vector.

- void deleteChar ()

  Delete last character entered.
- void enterPress ()

  Handles Enter key press.
- void setString (std::string nstring)

  Sets the string.
- std::string getString () const

  returns the text in tbox
- void draw (sf::RenderTarget &target, sf::RenderStates states) const override

  used to draw to the screen virutal method inherited from MyRect -> sf::Drawable thats overrided here is what allows us to draw to window using window.draw(TextBox)
- bool isMouseHover ()

  check if mouse is hovering over current textbox

Public Member Functions inherited from MyRect

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

  constructor for MyRect
- MyRect ()
- void setPosition (sf::Vector2f pos)

  sets the position of rect
- sf::Vector2f getPos () const

  get the position of rect
- void setFillColour (sf::Color colour)

  set the fill colour of the rect
- void setSize (sf::Vector2f size)

  set the size of the rect
- sf::Vector2f getSize () const

  get the size of the rect
- void draw (sf::RenderTarget &target, sf::RenderStates states) const override

  virutal method to draw to window

Additional Inherited Members

Protected Attributes inherited from MyRect

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

7.1.1 Detailed Description

Class to handle the command TextBox.

7.1.2 Member Function Documentation

7.1.2.1 TextBox() [1/2]

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

Using teh Parent class constructor.

7.1.2.2 TextBox() [2/2]

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

Using teh Parent class constructor.

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

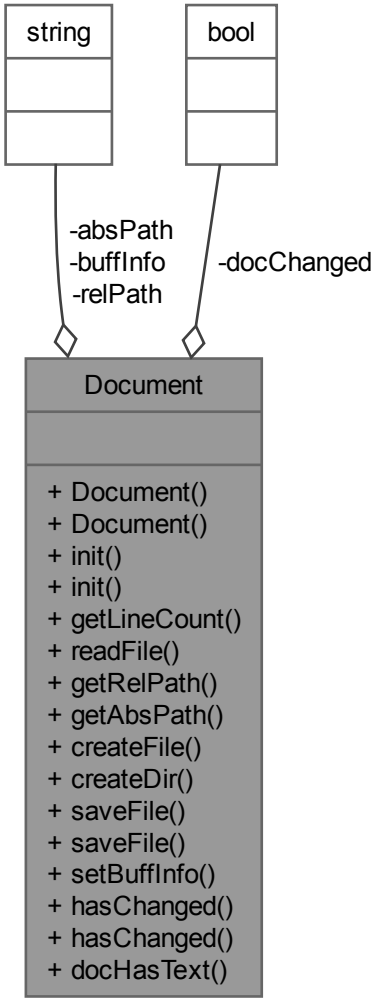
- include/Kamil/CmdBox.h

7.2 Document Class Reference

Document class.

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

Collaboration diagram for Document:



Public Member Functions

- Document ()

  Constructor for Document class.
- Document (std::string fileP)

  Constructor for Document class.
- void init ()

  initialise the file TODO: overload this with file path
- void init (std::string inF)
- int getLineCount ()
- std::string readFile ()

  read the file
- std::string getRelPath ()

  get the relative path

- std::string getAbsPath ()
  get the relative path
- void createFile ()
  create the file
- void createDir ()
  create a directory
- bool saveFile (const std::string &filename)
  save to a file
- bool saveFile ()
- void setBuffInfo (std::string info)
- bool hasChanged ()
  if the file has changed
- bool hasChanged (bool val)
- bool docHasText ()

Private Attributes

- std::string relPath
- std::string absPath
- std::string buffInfo
- bool docChanged

7.2.1 Detailed Description

Document class.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 Document() [1/2]

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

Constructor for Document class.

7.2.2.2 Document() [2/2]

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

Constructor for Document class.

Parameters

fileP	- file path

7.2.3 Member Function Documentation

7.2.3.1 createDir()

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

create a directory

7.2.3.2 createFile()

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

create the file

7.2.3.3 docHasText()

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

7.2.3.4 getAbsPath()

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

get the relative path

7.2.3.5 getLineCount()

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

7.2.3.6 getRelPath()

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

get the relative path

7.2.3.7 hasChanged() [1/2]

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

if the file has changed

Returns

bool - true if file has changed

Here is the caller graph for this function:

7.2.3.8 hasChanged() [2/2]

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

7.2.3.9 init() [1/2]

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

initialise the file TODO: overload this with file path

Here is the caller graph for this function:

7.2.3.10 init() [2/2]

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

7.2.3.11 readFile()

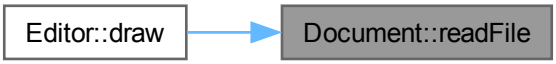
```
std::string Document::readFile ( )
```

read the file

Returns

std::string to the buffer info

Here is the caller graph for this function:



7.2.3.12 saveFile() [1/2]

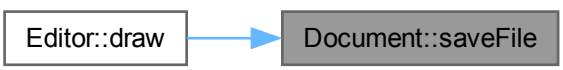
```
bool Document::saveFile ( )
```

7.2.3.13 saveFile() [2/2]

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

save to a file

Here is the caller graph for this function:



7.2.3.14 setBuffInfo()

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

Here is the caller graph for this function:

7.2.4 Member Data Documentation

7.2.4.1 absPath

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

absolute path

7.2.4.2 buffInfo

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

7.2.4.3 docChanged

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

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

7.2.4.4 relPath

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

relative path

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

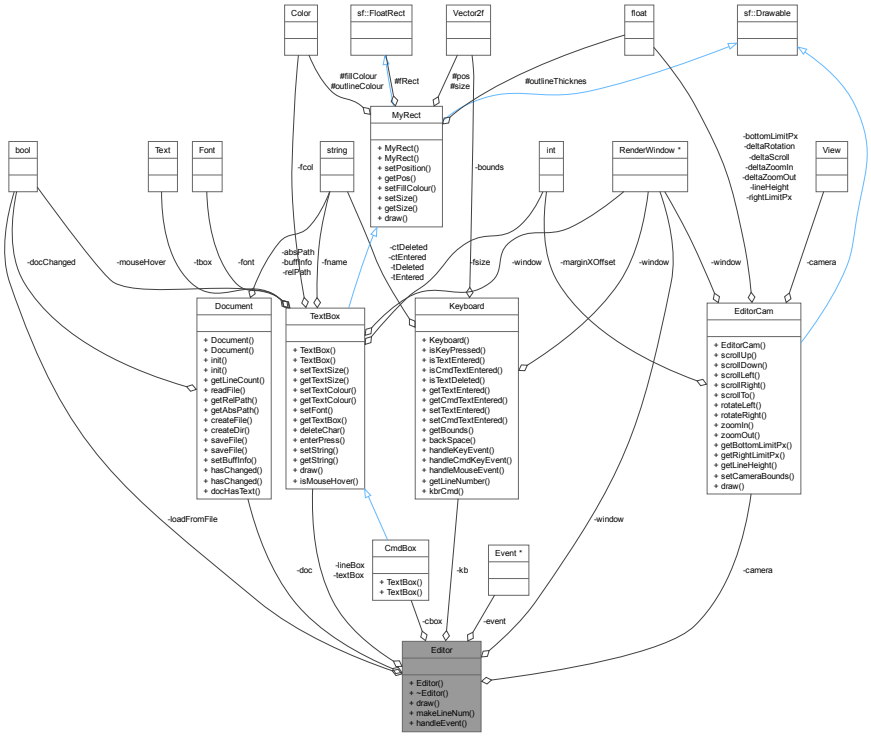
- include/Kamil/Document.h
- src/Document.cpp

7.3 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 ()
- 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.

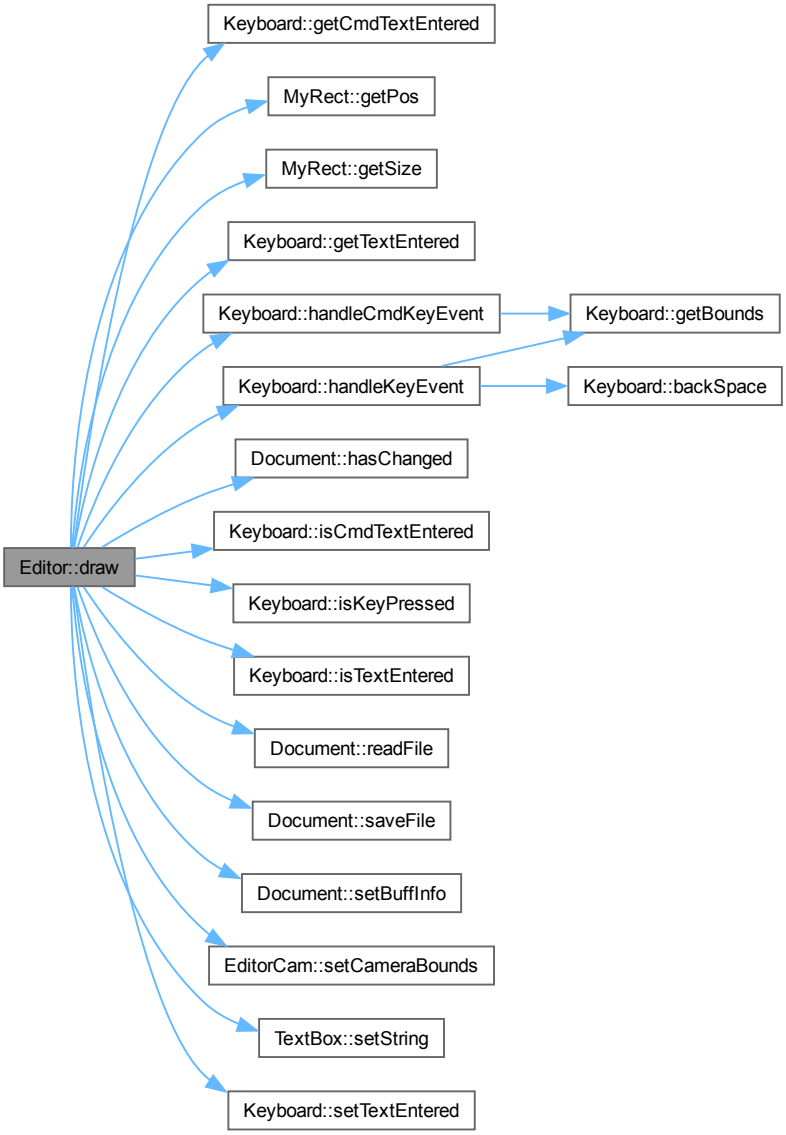
7.3.3 Member Function Documentation

7.3.3.1 draw()

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

function that draws everything to RenderWindow

SOON DEPRECATED Here is the call graph for this function:

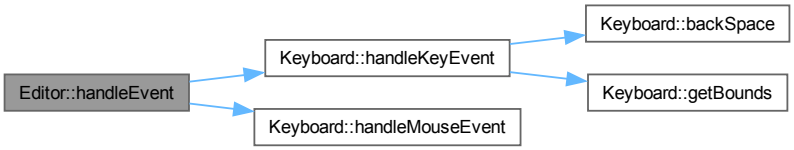


7.3.3.2 handleEvent()

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

handle the events for the Editor

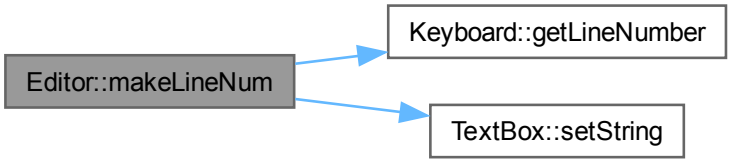
where all event handles are called when interacting with other classes e.g. kb.handleEvent(); kb.handleMouse↩ Events(); Here is the call graph for this function:



7.3.3.3 makeLineNum()

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

Here is the call graph for this function:



7.3.4 Member Data Documentation

7.3.4.1 camera

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

7.3.4.2 cbox

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

reference to command box that we draw

7.3.4.3 doc

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

pointer to the working document

7.3.4.4 event

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

refernce to event

7.3.4.5 kb

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

handles keyboard events

7.3.4.6 lineBox

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

7.3.4.7 loadFromFile

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

7.3.4.8 textBox

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

reference to textbox that we draw

7.3.4.9 window

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

refernce to RenderWindow

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

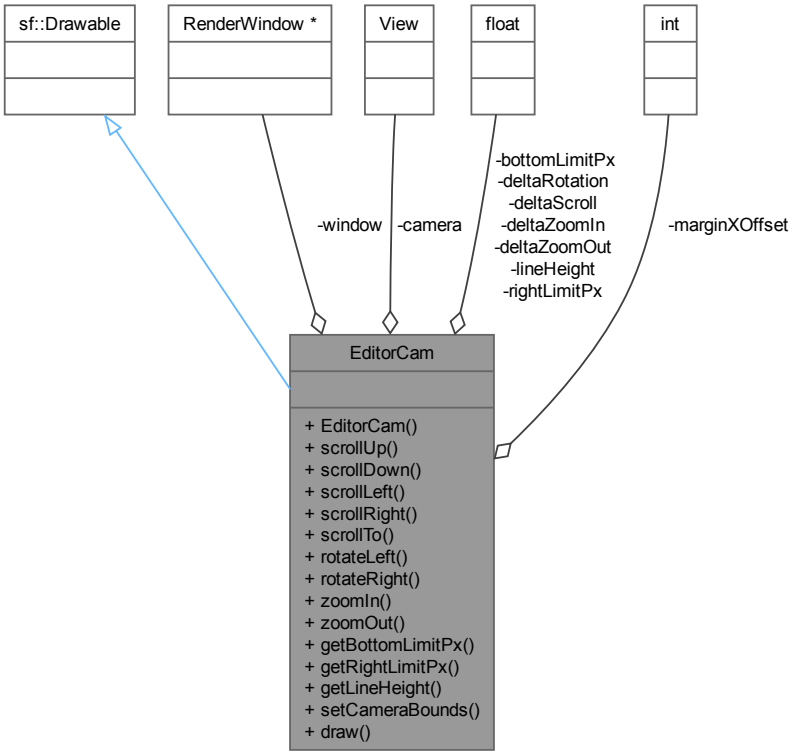
- include/Kamil/Editor.h
- src/Editor.cpp

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)
- void scrollUp ()
- void scrollDown ()
- void scrollLeft ()
- void scrollRight ()
- void scrollTo (float x, float y)
- void rotateLeft ()
- void rotateRight ()
- void zoomIn ()
- void zoomOut ()
- float getBottomLimitPx ()
- float getRightLimitPx ()
- int getLineHeight ()
- void setCameraBounds (int width, int height)
- void draw (sf::RenderTarget &target, sf::RenderStates states) const override

Private Attributes

- sf::RenderWindow * window
- sf::View camera

- float deltaScroll
- float deltaRotation
- float deltaZoomIn
- float deltaZoomOut
- float rightLimitPx
- float bottomLimitPx
- float lineHeight
- int marginXOffset

7.4.1 Constructor & Destructor Documentation

7.4.1.1 EditorCam()

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

7.4.2 Member Function Documentation

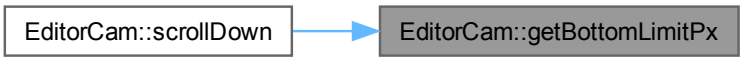
7.4.2.1 draw()

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

7.4.2.2 getBottomLimitPx()

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

Here is the caller graph for this function:



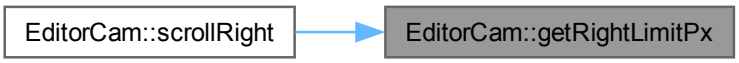
7.4.2.3 getLineHeight()

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

7.4.2.4 getRightLimitPx()

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

Here is the caller graph for this function:



7.4.2.5 rotateLeft()

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

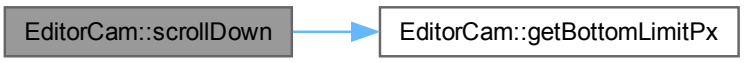
7.4.2.6 rotateRight()

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

7.4.2.7 scrollDown()

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

Here is the call graph for this function:



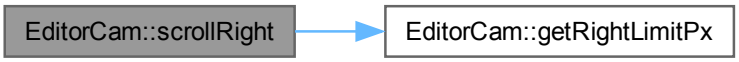
7.4.2.8 scrollLeft()

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

7.4.2.9 scrollRight()

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

Here is the call graph for this function:



7.4.2.10 scrollTo()

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

7.4.2.11 scrollUp()

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

7.4.2.12 setCameraBounds()

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

Here is the caller graph for this function:

7.4.2.13 zoomIn()

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

7.4.2.14 zoomOut()

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

7.4.3 Member Data Documentation

7.4.3.1 bottomLimitPx

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

7.4.3.2 camera

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

handles camera manipulation

7.4.3.3 deltaRotation

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

delta time for rotation

7.4.3.4 deltaScroll

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

delta 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]
```

7.4.3.8 marginXOffset

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

7.4.3.9 rightLimitPx

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

delta time for zoomin/out

7.4.3.10 window

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

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

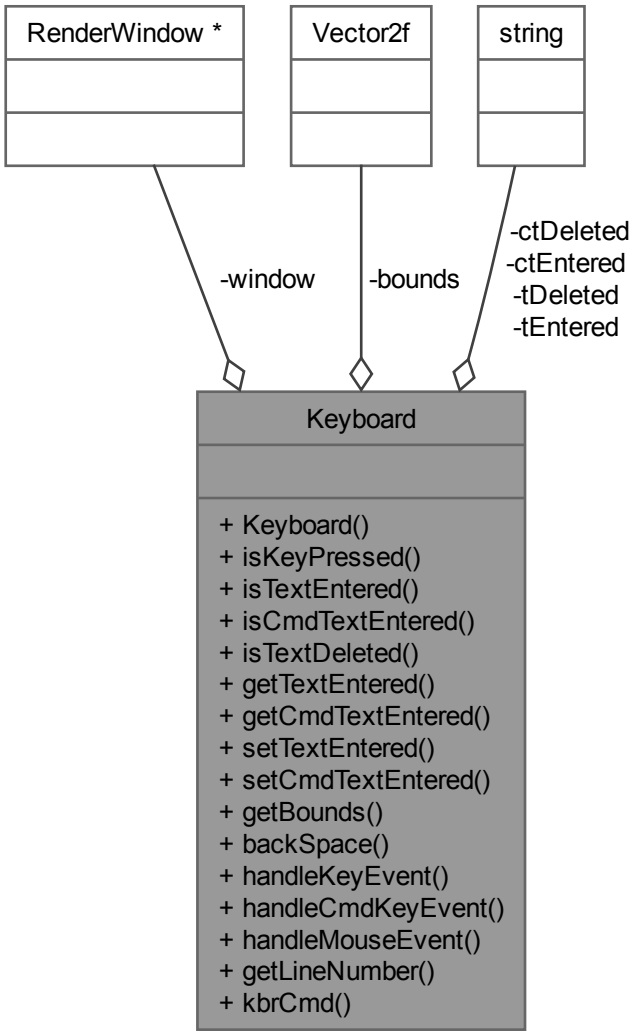
- include/Kamil/EditorCam.h
- src/EditorCam.cpp

7.5 Keyboard Class Reference

A class to handle Keyboard input.

`#include <Keyboard.h>`

Collaboration diagram for Keyboard:



Public Member Functions

- Keyboard (sf::RenderWindow *win, Document *doc, sf::Vector2f bounds)

  Constructor for Keyboard class.
- bool isKeyPressed (sf::Keyboard::Key)

  checks if a key is pressed
- bool isTextEntered ()

checks if a text is entered

- bool isCmdTextEntered ()

  checks if text is entered to the command box
- bool isTextDeleted ()

  check if text is being deleted
- std::string getTextEntered ()

  returns text entered
- std::string getCmdTextEntered ()

  returns text entered
- void setTextEntered (std::string)

  sets text
- void setCmdTextEntered (std::string)

  sets text
- sf::Vector2f getBounds () const

  get the bounds of the area we are in
- void backSpace ()

  when we backspace on teh text
- void handleKeyEvent (sf::Event &event)

  handle keyboard events
- void handleCmdKeyEvent ()

  handle keyboard events
- void handleMouseEvent (sf::Event &event)

  mouse keyboard events
- int getLineNumber ()
- template<typename T , size_t N, typename... Args>
  void kbrCmd (Args... args)

Private Attributes

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

7.5.1 Detailed Description

A class to handle Keyboard input.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 Keyboard()

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

Constructor for Keyboard class.

Parameters

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

Here is the caller graph for this function:

7.5.3.2 getBounds()

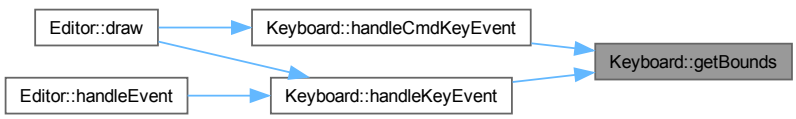
```
sf::Vector2f Keyboard::getBounds ( ) const
```

get the bounds of the area we are in

Returns

sf::Vector2f bounded area

Here is the caller graph for this function:



7.5.3.3 getCmdTextEntered()

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

returns text entered

Returns

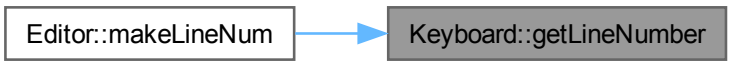
std::string text entered

Here is the caller graph for this function:

7.5.3.4 getLineNumber()

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

Here is the caller graph for this function:



7.5.3.5 getTextEntered()

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

returns text entered

Returns

std::string text entered

Here is the caller graph for this function:

7.5.3.6 handleCmdKeyEvent()

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

handle keyboard events

Parameters

event	- to get text entered from events

Here is the call graph for this function:

Here is the caller graph for this function:

7.5.3.7 handleKeyEvent()

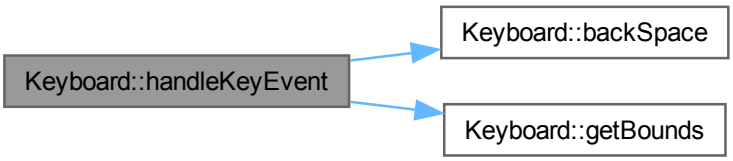
```
void Keyboard::handleKeyEvent (
            sf::Event & event )
```

handle keyboard events

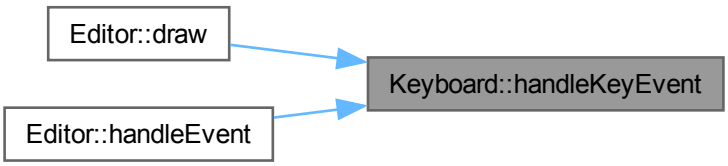
Parameters

event	- to get text entered from events

Here is the call graph for this function:



Here is the caller graph for this function:



7.5.3.8 handleMouseEvent()

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

mouse keyboard events

Parameters

event	- to get text entered from events

Here is the caller graph for this function:

7.5.3.9 isCmdTextEntered()

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

checks if text is entered to the command box

Returns

bool tru eif key is pressed false if not

Here is the caller graph for this function:

7.5.3.10 isKeyPressed()

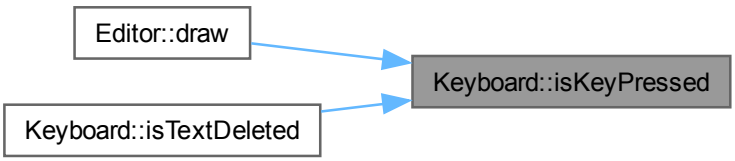
```
bool Keyboard::isKeyPressed (
            sf::Keyboard::Key key )
```

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()

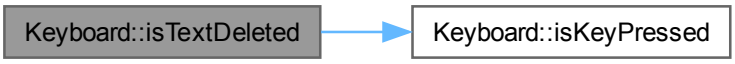
```
bool Keyboard::isTextDeleted ( )
```

check if text is being deleted

Returns

bool true if text is being deleted

Here is the call graph for this function:



7.5.3.12 isTextEntered()

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

checks if a text is entered

Returns

bool true if key is pressed false if not

Here is the caller graph for this function:

7.5.3.13 kbrCmd()

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

7.5.3.14 setCmdTextEntered()

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

sets text

Parameters

nstring	- new string

7.5.3.15 setTextEntered()

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

sets text

Parameters

nstring	- new string

Here is the caller graph for this function:

7.5.4 Member Data Documentation

7.5.4.1 bounds

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

store the bounded area

7.5.4.2 ctDeleted

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

tmp for text deleted to cmd

7.5.4.3 ctEntered

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

tmp for text 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:

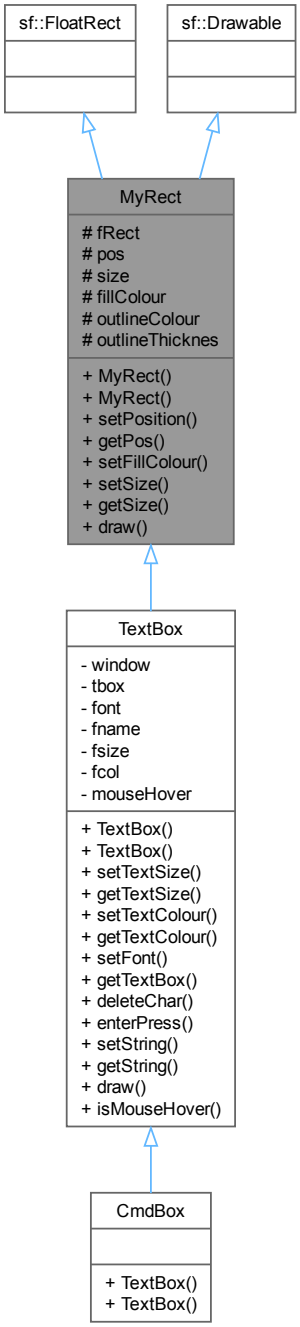
- include/Kamil/Keyboard.h
- src/Keyboard.cpp

7.6 MyRect Class Reference

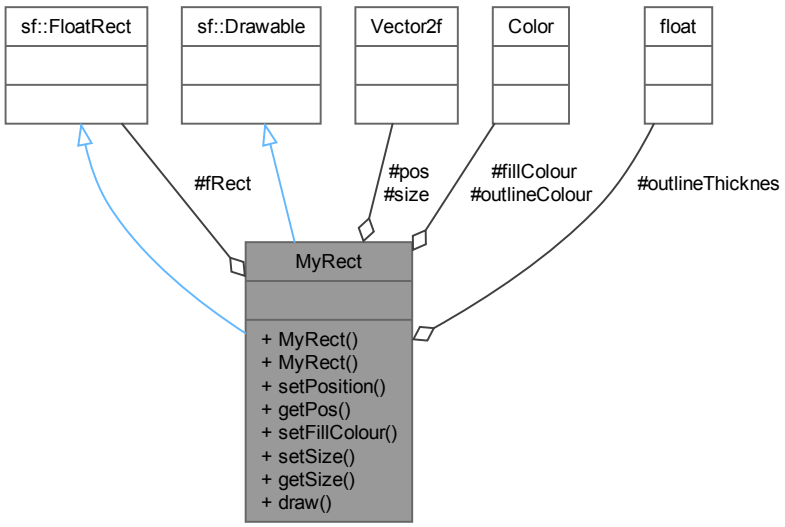
gives extra functionality to FloatRect

```
#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

7.6.3.2 getPos()

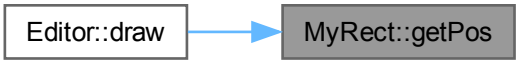
```
sf::Vector2f MyRect::getPos ( ) const
```

get the position of rect

Returns

sf::Vector2f pos

Here is the caller graph for this function:



7.6.3.3 getSize()

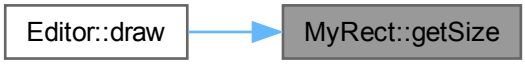
```
sf::Vector2f MyRect::getSize ( ) const
```

get the size of the rect

Returns

sf::Vector2f size

Here is the caller graph for this function:



7.6.3.4 setFillColour()

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

set the fill colour of the rect

Parameters

sf::Color	colour

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

7.6.4 Member Data Documentation

7.6.4.1 fillColour

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

colour of rect

7.6.4.2 fRect

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

for collision checking

7.6.4.3 outlineColour

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

outline colour of rect

7.6.4.4 outlineThicknes

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

outline thickness of rect

7.6.4.5 pos

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

position of rect

7.6.4.6 size

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

size of rect

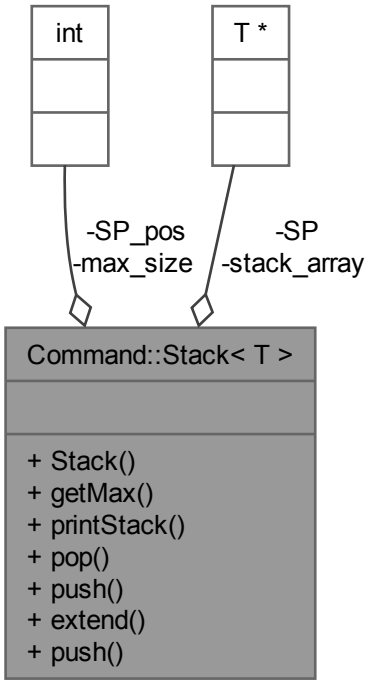
The documentation for this class was generated from the following files:

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

7.7 Command::Stack< T > Class Template Reference

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

Collaboration diagram for Command::Stack< T >:



Public Member Functions

- Stack (int)
- int getMax () const
- void printStack () const
- int pop ()
- int push (T)
- void extend (int)
- int push (std::string value)

Private Attributes

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

7.7.1 Constructor & Destructor Documentation

7.7.1.1 Stack()

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

7.7.2 Member Function Documentation

7.7.2.1 extend()

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

7.7.2.2 getMax()

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

7.7.2.3 pop()

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

7.7.2.4 printStack()

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

7.7.2.5 push() [1/2]

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

7.7.2.6 push() [2/2]

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

Here is the caller graph for this function:

7.7.3 Member Data Documentation

7.7.3.1 max_size

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

7.7.3.2 SP

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

7.7.3.3 SP_pos

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

7.7.3.4 stack_array

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

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

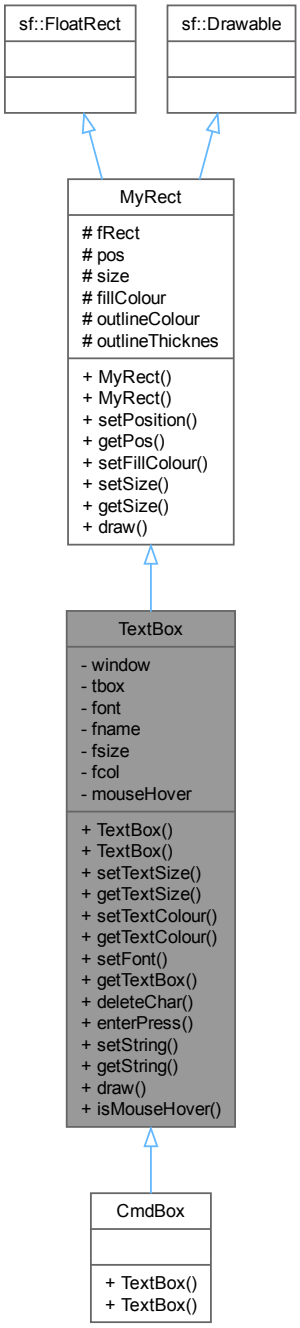
- include/Kamil/Commands.h
- include/Kamil/Utils/Stack.h
- src/Utils/Stack.cpp

7.8 TextBox Class Reference

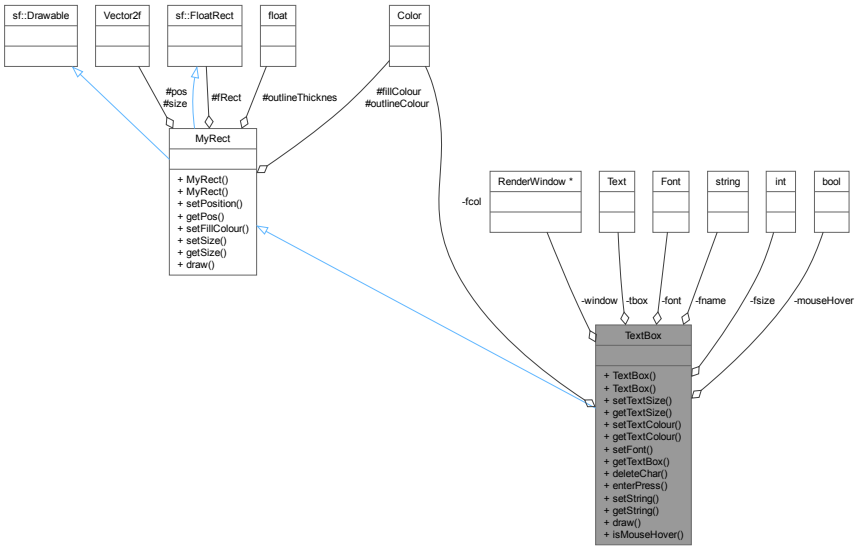
A class that makes a Textbox in SFML.

`#include <TextBox.h>`

Inheritance diagram for TextBox:



Collaboration diagram for TextBox:



Public Member Functions

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

  Constructor for TextBox.
- TextBox ()
- void setTextSize (int size)

  Set the size of the text.
- int getTextSize () const

  Get the size of the text.
- void setTextColour (sf::Color colour)

  Set the colour of the text.
- sf::Color getTextColour () const

  Get the colour of the text.
- void setFont (sf::Font &font)

  set what font you use
- sf::Text getTextBox () const

  Get both the main textbox and the cmd textbox in a vector.
- void deleteChar ()

  Delete last character entered.
- void enterPress ()

  Handles Enter key press.
- void setString (std::string nstring)

  Sets the string.
- std::string getString () const

  returns the text in tbox
- void draw (sf::RenderTarget &target, sf::RenderStates states) const override

  used to draw to the screen virutal method inherited from MyRect -> sf::Drawable thats overrided here is what allows us to draw to window using window.draw(TextBox)
- bool isMouseHover ()

  check if mouse is hovering over current textbox

Public Member Functions inherited from MyRect

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

Private Attributes

- sf::RenderWindow * window
- sf::Text tbox {}
- sf::Font font {}
- std::string fname {}
- int fsize {}
- sf::Color fcol {}
- bool mouseHover

Additional Inherited Members

Protected Attributes inherited from MyRect

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

7.8.1 Detailed Description

A class that makes a Textbox in SFML.

The class creates a textbox for inputting and handling text and Keyboard commands and allows the use of commands in the secondary textbox cmdbox

7.8.2 Constructor & Destructor Documentation

7.8.2.1 TextBox() [1/2]

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

Constructor for TextBox.

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.

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.

7.8.3.4 getString()

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

returns the text in tbox

Returns

type std::string

7.8.3.5 getTextBox()

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

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

Returns

type Boxv2 that contains textbox and cmdbox

7.8.3.6 getTextColour()

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

Get the colour of the text.

Returns

sf::Colour textColour

7.8.3.7 getTextSize()

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

Get the size of the text.

Returns

an int of the text size

7.8.3.8 isMouseHover()

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

check if mouse is hovering over current textbox

Returns

bool - yes if hovering

7.8.3.9 setFont()

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

set what font you use

Parameters

font	file dir of font

7.8.3.10 setString()

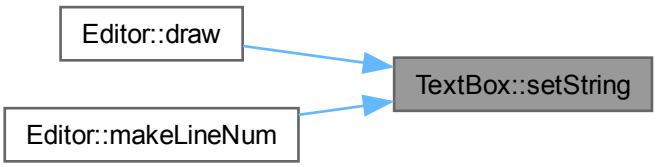
```
void TextBox::setString (
            std::string nstring )
```

Sets the string.

Parameters

nstring	- new string placed on tbox

Here is the caller graph for this function:



7.8.3.11 setTextColour()

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

Set the colour of the text.

Parameters

fill	font colour

7.8.3.12 setTextSize()

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

Set the size of the text.

Parameters

size	text size

7.8.4 Member Data Documentation

7.8.4.1 fcol

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

the font colour

7.8.4.2 fname

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

the name of the font used

7.8.4.3 font

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

the font that the TextBox uses

7.8.4.4 fsize

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

the font size

7.8.4.5 mouseHover

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

if the mouse is hovering over

7.8.4.6 tbox

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

the text that everything is written onto

7.8.4.7 window

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

pointer to the main RenderWindow variable

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

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

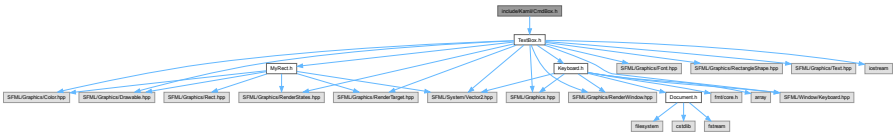
Chapter 8

File Documentation

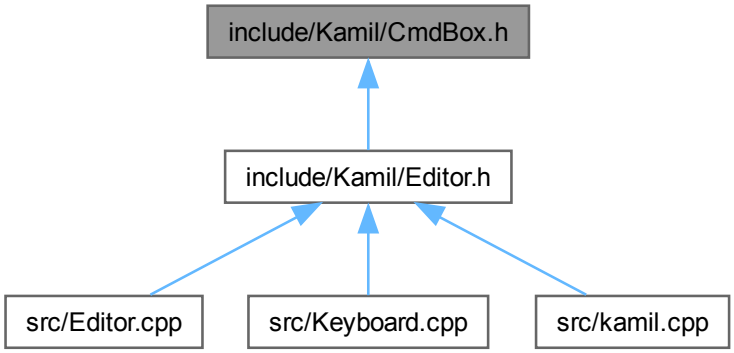
8.1 include/Kamil/CmdBox.h File Reference

```
#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

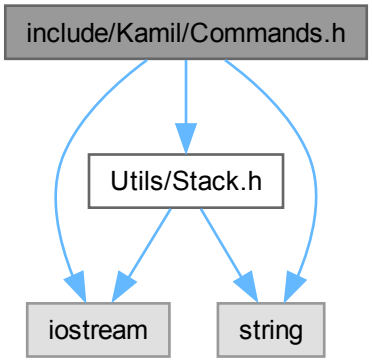
Go to the documentation of this file.

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

8.3 include/Kamil/Commands.h File Reference

```
#include <iostream>
#include <string>
#include "Utils/Stack.h"
```

Include dependency graph for Commands.h:



Namespaces

- namespace Command

  A stack in the Command namespace.

8.4 Commands.h

Go to the documentation of this file.

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

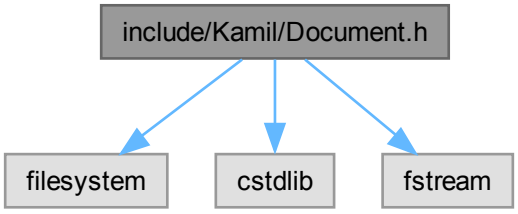
```
00008
00009 namespace Command{
00010     // template <typename T>
00011     // class Node{ // used for LinkedList
00012     //     public:
00013     //         Node();
00014     //         Node(T);
00015     //         T data;
00016     //         Node* next;
00017     // };
00018
00019     // template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

8.5 include/Kamil/Document.h File Reference

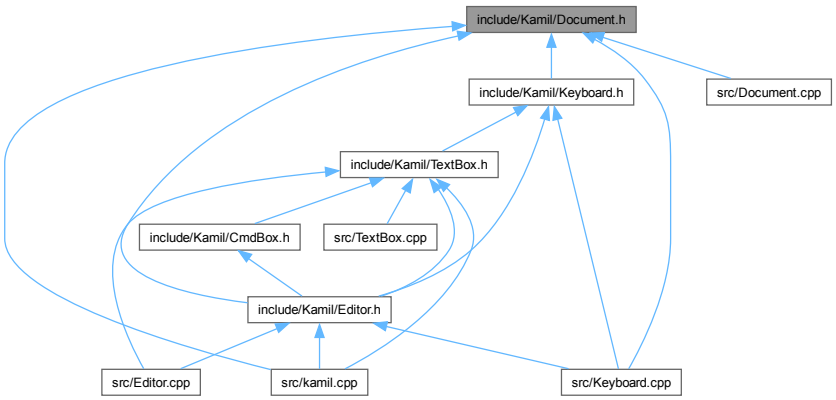
Interface file for the Document class.

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

Include dependency graph for Document.h:



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



Classes

- class Document

  Document class.

8.5.1 Detailed Description

Interface file for the Document class.

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

8.6 Document.h

Go to the documentation of this file.

```
00001 #ifndef KAMIL_DOCUMENT_H
00002 #define KAMIL_DOCUMENT_H
00003
00004
00014 #include <filesystem>
00015 #include <cstdlib>
00016 #include <fstream>
00017
00021 class Document{
00022 public:
00026     Document();
00027
00032     Document(std::string fileP);
00033
00038     void init();
00039
00040     void init(std::string inF);
00041
00042
00043     int getLineCount();
00044
00049     std::string readFile();
00050
00054     std::string getRelPath();
00055
00059     std::string getAbsPath();
00060
```

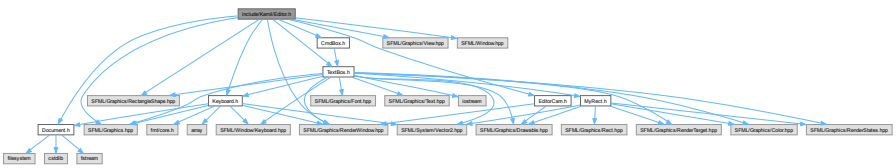
```
00064     void createFile();
00065
00069     void createDir();
00070
00074     bool saveFile(const std::string& filename);
00075
00076     bool saveFile();
00077
00078
00079     void setBuffInfo(std::string info);
00080
00085     bool hasChanged();
00086
00087     bool hasChanged(bool val);
00088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

8.7 include/Kamil/Editor.h File Reference

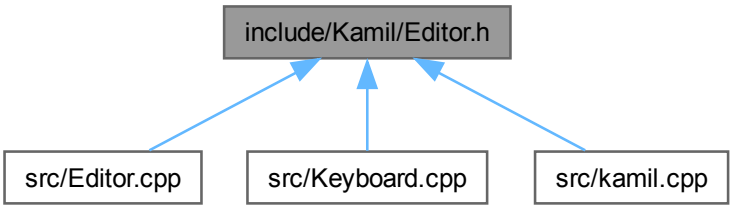
Interface file for the Editor class.

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

Include dependency graph for Editor.h:



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



Classes

- class Editor

  Class that handles and draws everything in the Editor.

8.7.1 Detailed Description

Interface file for the Editor class.

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

8.8 Editor.h

Go to the documentation of this file.

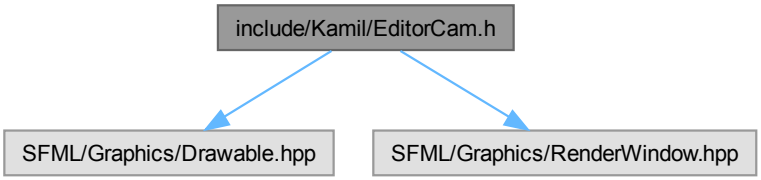
```
00001 #ifndef KAMIL_EDITOR_WINDOW_HPP
00002 #define KAMIL_EDITOR_WINDOW_HPP
00003
00016 #include <SFML/Graphics.hpp>
00017 #include <SFML/Graphics/RectangleShape.hpp>
00018 #include <SFML/Graphics/RenderWindow.hpp>
00019 #include <SFML/Graphics/View.hpp>
00020 #include <SFML/Window.hpp>
00021
00022 #include "TextBox.h"
00023 #include "Keyboard.h"
00024 #include "CmdBox.h"
00025 #include "Document.h"
00026 #include "EditorCam.h"
00027
00028
00032 class Editor{
00033     public:
00040         Editor(sf::RenderWindow* window, sf::Event* event, Document* doc);
00041
00045         ~Editor();
00046
00052         void draw();
00053
00054         void makeLineNum();
00055
00062         void handleEvent();
00063
00064     private:
00065         Document* doc;
00066         TextBox* textBox;
```

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

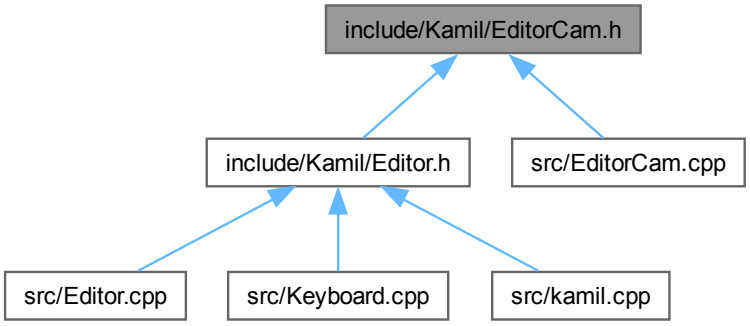
8.9 include/Kamil/EditorCam.h File Reference

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

Include dependency graph for EditorCam.h:



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



Classes

- class EditorCam

8.10 EditorCam.h

Go to the documentation of this file.

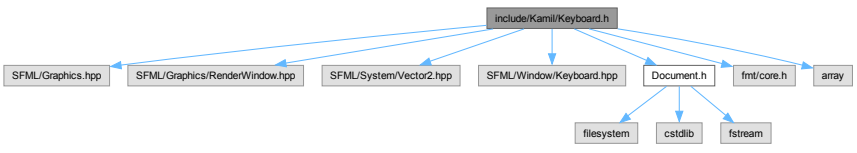
```
00001 #ifndef KAMIL_EDITOR_CAM_H
00002 #define KAMIL_EDITOR_CAM_H
00003
00004
00005
00006 #include <SFML/Graphics/Drawable.hpp>
00007 #include <SFML/Graphics/RenderWindow.hpp>
00008
00009
00010 class EditorCam : public sf::Drawable
00011 {
00012 public:
00013     EditorCam(sf::RenderWindow* window,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

8.11 include/Kamil/Keyboard.h File Reference

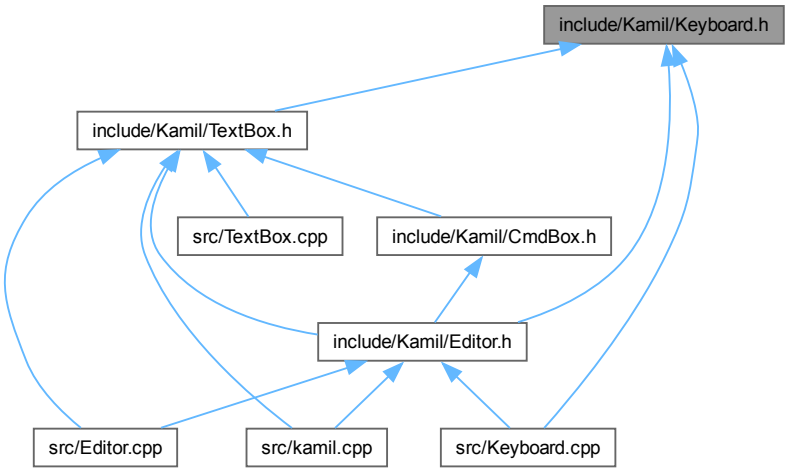
Interface file for Keyboard.h.

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

Include dependency graph for Keyboard.h:



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



Classes

- class Keyboard

  A class to handle Keyboard input.

Namespaces

- namespace KEYS

  An enum for Keyboard characters in hex form.

Enumerations

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

8.11.1 Detailed Description

Interface file for Keyboard.h.

A class that handles all keyboard and mouse events for the editor is responsible for 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
00004
00015 #include <SFML/Graphics.hpp>
00016 #include <SFML/Graphics/RenderWindow.hpp>
00017 #include <SFML/System/Vector2.hpp>
00018 #include <SFML/Window/Keyboard.hpp>
00019
00020 #include "Document.h"
00021 #include <fmt/core.h>
00022
00023 #include <array>
00024
00028 namespace KEYS{
00029     enum {
00030         ESCAPE = 0x1B,
00031         ENTER = 0xD,
00032         BS = 0x8,
00033         Shift_A = 0x41,
00034         CTRL = 0x11,
00035         DELETE = 0x7f,
00036     };
00037 }
00038
00039
00040 #ifdef USE_KEYS
00041
00042 #define "LControl" sf::Keyboard::KEYS::LControl
00043
00044 #endif
00045
00046
00047
00051 class Keyboard{
00052     public:
00058         Keyboard(sf::RenderWindow* win, Document* doc, sf::Vector2f bounds);
00059
00060
00065         bool isKeyPressed(sf::Keyboard::Key);
00066
00071         bool isTextEntered();
00072
00077         bool isCmdTextEntered();
00078
00083         bool isTextDeleted();
00084
00089         std::string getTextEntered();
00090
00091
00096         std::string getCmdTextEntered();
00097
00098
00103         void setTextEntered(std::string);
00104
00105
00110         void setCmdTextEntered(std::string);
00111
00112
00117         sf::Vector2f getBounds() const;
00118
00122         void backSpace();
00123
00128         void handleKeyEvent(sf::Event& event);
00129
00130
00135         void handleCmdKeyEvent();
00136
00141         void handleMouseEvent(sf::Event& event); // not implemented yet
00142
00143
00144         int getLineNumber();
00145
00146         template<typename T, size_t N, typename... Args>
00147         void kbrCmd(Args... args){
00148             std::array<T,N> val{args...};
00149             for(const auto& element: val){
00150                 fmt::print("{}", element);
00151             }
00152         }
00153
00154
```

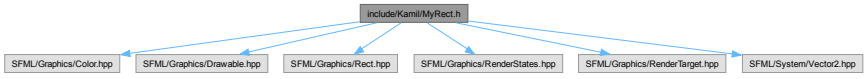
```
00155 // get position in text
00156
00157     private:
00158         sf::RenderWindow* window;
00159         // Document* doc;
00160         sf::Vector2f bounds;
00161         std::string tEntered;
00162         std::string tDeleted;
00164         std::string ctEntered;
00165         std::string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

8.13 include/Kamil/MyRect.h File Reference

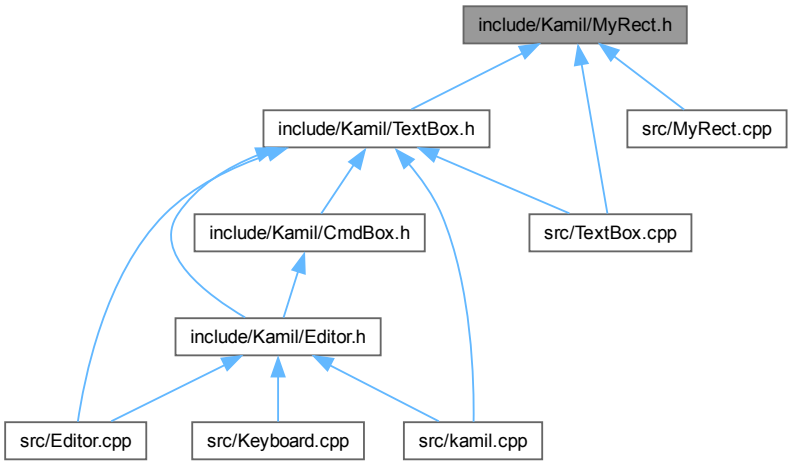
Interface file for the MyRect class.

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

Include dependency graph for MyRect.h:



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



Classes

- class MyRect

  gives extra functionality to FloatRect

8.13.1 Detailed Description

Interface file for the MyRect class.

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

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

8.14 MyRect.h

Go to the documentation of this file.

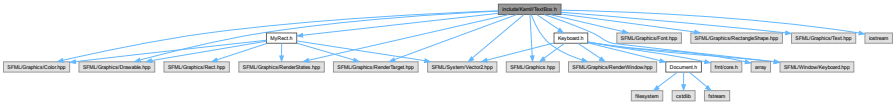
```
00001 #ifndef KAMIL_MYRECT_H
00002 #define KAMIL_MYRECT_H
00003
00004
00025 #include <SFML/Graphics/Color.hpp>
00026 #include <SFML/Graphics/Drawable.hpp>
00027 #include <SFML/Graphics/Rect.hpp>
00028 #include <SFML/Graphics/RenderStates.hpp>
00029 #include <SFML/Graphics/RenderTarget.hpp>
00030 #include <SFML/System/Vector2.hpp>
00031
00032
00039 class MyRect : public sf::FloatRect
00040              , public sf::Drawable
00041 {
00042     public:
00051         MyRect(sf::Vector2f pos, sf::Vector2f size, sf::Color fillColour, sf::Color outlineColour,
      float outlineThicknes);
00052         MyRect();
00053
00058         void setPosition(sf::Vector2f pos);
00059
00064         sf::Vector2f getPos()const;
00065
00070         void setFillColour(sf::Color colour);
00071
00076         void setSize(sf::Vector2f size);
00077
00082         sf::Vector2f getSize()const;
00083
00094         void draw(sf::RenderTarget& target, sf::RenderStates states)const override;
00095
00096     protected:
00097         sf::FloatRect fRect;
00098         sf::Vector2f pos;
00099         sf::Vector2f size;
00100         sf::Color fillColour;
00101         sf::Color outlineColour;
00102         float outlineThicknes;
00104 };
00105
00106 #endif // KAMIL_MYRECT_H
00107
```

8.15 include/Kamil/TextBox.h File Reference

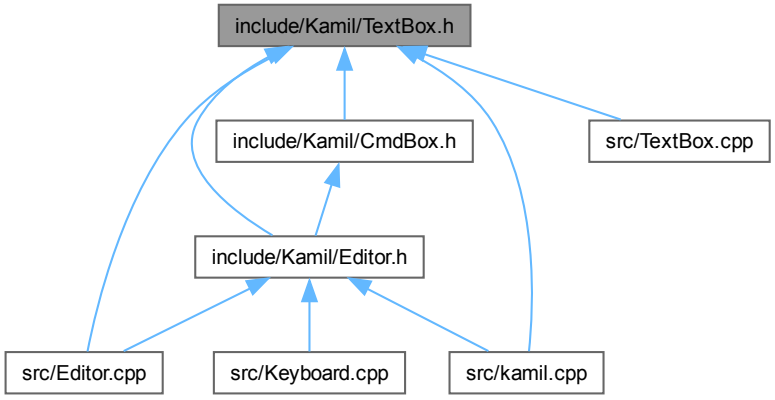
```
#include <SFML/Graphics.hpp>
#include <SFML/Graphics/Color.hpp>
#include <SFML/Graphics/Drawable.hpp>
```

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

Include dependency graph for TextBox.h:



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



Classes

- class TextBox

  A class that makes a Textbox in SFML.

8.16 TextBox.h

Go to the documentation of this file.

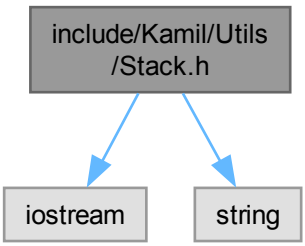
```
00001 #ifndef KAMIL_TEXTBOX_HPP
00002 #define KAMIL_TEXTBOX_HPP
00003
00014 #include <SFML/Graphics.hpp>
00015 #include <SFML/Graphics/Color.hpp>
00016 #include <SFML/Graphics/Drawable.hpp>
00017 #include <SFML/Graphics/Font.hpp>
```

```
00018 #include <SFML/Graphics/RectangleShape.hpp>
00019 #include <SFML/Graphics/RenderStates.hpp>
00020 #include <SFML/Graphics/RenderTarget.hpp>
00021 #include <SFML/Graphics/RenderWindow.hpp>
00022 #include <SFML/Graphics/Text.hpp>
00023 #include <SFML/System/Vector2.hpp>
00024 #include <SFML/Window/Keyboard.hpp>
00025 #include <iostream>
00026
00027 #include "Keyboard.h"
00028 #include "MyRect.h"
00029
00030
00031 /*
00032  *
00033  * TODO:
00034  *       Make a RectangleShape that acts as the bounds of the TextBox
00035  *       then add limits to the textbox so it stays in the limits
00036  *
00037  *       Add the Keybord manager class here and use its methods
00038  *       to handle the key events
00039  */
00040
00041
00048 class TextBox : public MyRect
00049 {
00050     public:
00062         TextBox(sf::RenderWindow* win, sf::Vector2f pos, sf::Vector2f size, std::string sfont, int
      fsize, sf::Color fcol, sf::Color background, float thicc);
00063         TextBox();
00064
00069         void setTextSize(int size);
00070
00075         int getTextSize()const;
00076
00081         void setTextColour(sf::Color colour);
00082
00087         sf::Color getTextColour()const;
00088
00093         void setFont(sf::Font& font);
00094
00100         sf::Text getTextBox()const;
00101
00105         void deleteChar();
00106
00110         void enterPress();
00111
00116         void setString(std::string nstring);
00117
00122         std::string getString()const;
00123
00131         void draw(sf::RenderTarget& target, sf::RenderStates states)const override;
00132
00137         bool isMouseHover();
00138
00139     private:
00140         sf::RenderWindow* window;
00141         sf::Text tbox{};
00142         sf::Font font{};
00143         std::string fname{};
00144         int fsize{};
00145         sf::Color fcol{};
00146         bool mouseHover;
00147 };
00148 #endif // KAMIL_TEXTBOX_HPP
```

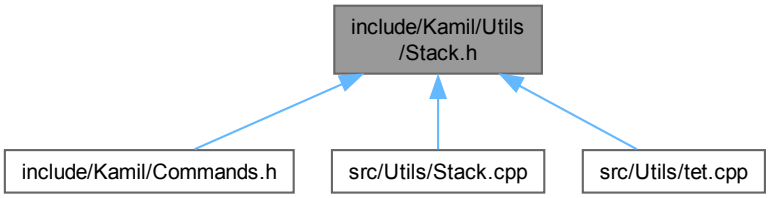
8.17 include/Kamil/Utils/Stack.h File Reference

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

Include dependency graph for Stack.h:



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



Classes

- class Command::Stack< T >

Namespaces

- namespace Command

  A stack in the Command namespace.

8.18 Stack.h

Go to the documentation of this file.

```
00001 #ifndef KAMIL_STACK_H
00002 #define KAMIL_STACK_H
00003
00004 #include <iostream>
00005 #include <string>
00006
00010 namespace Command{
00011
00012 // Dynamic stack array
00013 template<typename T>
```

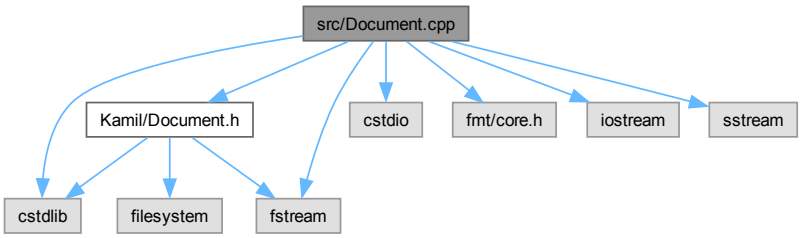
```
00014 class Stack{
00015 public:
00016     Stack(int);
00017     int getMax()const;
00018     void printStack()const;
00019     int pop();
00020     int push(T);
00021     void extend(int);
00022 private:
00023     int max_size{};
00024     T* stack_array{new T[max_size]};
00025     T* SP = &stack_array[max_size];
00026     int SP_pos = max_size;
00027 };
00028
00029 #endif
00030
00031 } // Command
```

8.19 README.md File Reference

8.20 src/Document.cpp File Reference

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

Include dependency graph for Document.cpp:

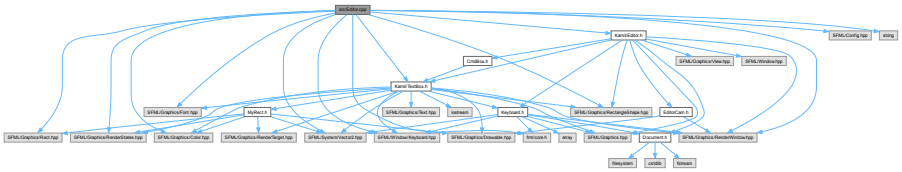


8.21 src/Editor.cpp File Reference

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

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

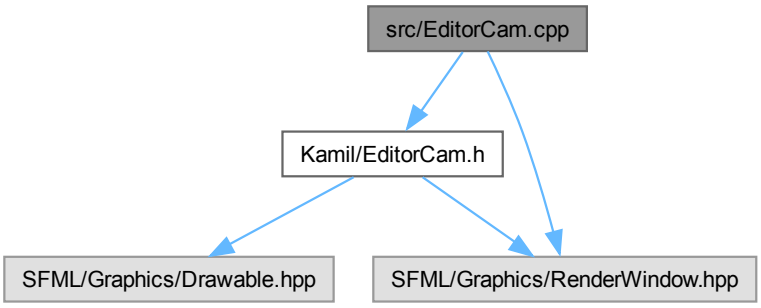
Include dependency graph for Editor.cpp:



8.22 src/EditorCam.cpp File Reference

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

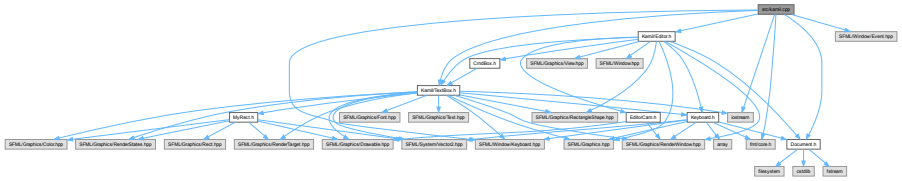
Include dependency graph for EditorCam.cpp:



8.23 src/kamil.cpp File Reference

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

Include dependency graph for kamil.cpp:



Functions

- int main (int argc, char ∗argv[ ])

8.23.1 Function Documentation

8.23.1.1 main()

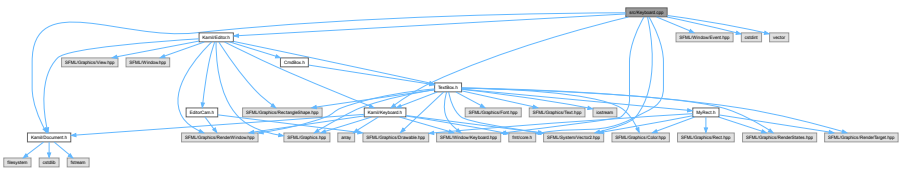
```
int main (
            int argc,
            char * argv[] )
```

Here is the call graph for this function:

8.24 src/Keyboard.cpp File Reference

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

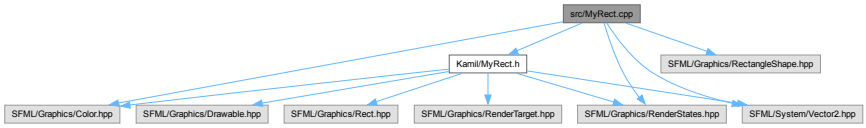
Include dependency graph for Keyboard.cpp:



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>
```

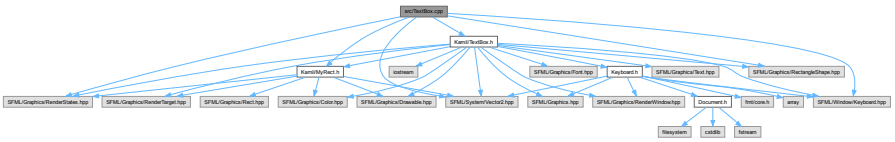
Include dependency graph for MyRect.cpp:



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>
```

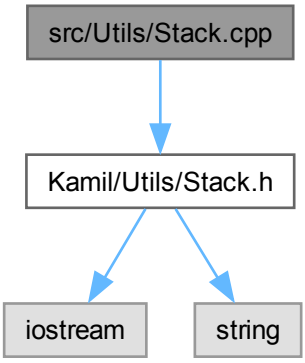
Include dependency graph for TextBox.cpp:



8.27 src/Utils/Stack.cpp File Reference

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

Include dependency graph for Stack.cpp:



Namespaces

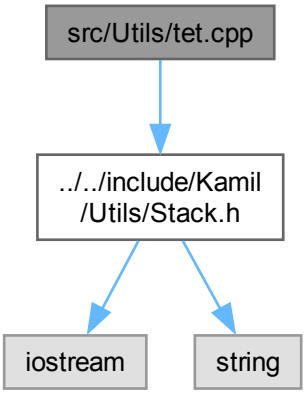
- namespace Command

  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