

# Tangram

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>Tangram</b>	<b>1</b>
<b>2</b>	<b>Hierarchical Index</b>	<b>5</b>
2.1	Class Hierarchy . . . . .	5
<b>3</b>	<b>Class Index</b>	<b>7</b>
3.1	Class List . . . . .	7
<b>4</b>	<b>File Index</b>	<b>9</b>
4.1	File List . . . . .	9
<b>5</b>	<b>Class Documentation</b>	<b>11</b>
5.1	A_Shape Class Reference . . . . .	11
5.1.1	Detailed Description . . . . .	13
5.1.2	Constructor & Destructor Documentation . . . . .	13
5.1.2.1	~A_Shape() . . . . .	13
5.1.3	Member Function Documentation . . . . .	13
5.1.3.1	aCurrentAngular() . . . . .	14
5.1.3.2	aGetArea() . . . . .	14
5.1.3.3	aGetColor() . . . . .	14
5.1.3.4	aGetPoints() . . . . .	14
5.1.3.5	aGetShape() . . . . .	15
5.1.3.6	aGetStatusReverse() . . . . .	15
5.1.3.7	alsInShape() . . . . .	15
5.1.3.8	aLeftCorner() . . . . .	16

5.1.3.9	<a href="#">aLeftFlip()</a>	16
5.1.3.10	<a href="#">aMove()</a>	16
5.1.3.11	<a href="#">aReverse()</a>	16
5.1.3.12	<a href="#">aRightFlip()</a>	17
5.1.3.13	<a href="#">aRotate()</a>	17
5.1.3.14	<a href="#">aSetPoints()</a>	17
5.1.3.15	<a href="#">aToString()</a>	17
5.1.3.16	<a href="#">computeDistance()</a>	18
5.2	<a href="#">C_Button Class Reference</a>	18
5.2.1	<a href="#">Detailed Description</a>	19
5.2.2	<a href="#">Constructor &amp; Destructor Documentation</a>	19
5.2.2.1	<a href="#">~C_Button()</a>	19
5.2.2.2	<a href="#">C_Button()</a> [1/2]	19
5.2.2.3	<a href="#">C_Button()</a> [2/2]	20
5.2.3	<a href="#">Member Function Documentation</a>	20
5.2.3.1	<a href="#">Click()</a>	20
5.2.3.2	<a href="#">ClickInButton()</a>	20
5.2.3.3	<a href="#">Draw()</a> [1/2]	21
5.2.3.4	<a href="#">Draw()</a> [2/2]	21
5.2.3.5	<a href="#">SetCallBack()</a>	21
5.3	<a href="#">C_Game Class Reference</a>	22
5.3.1	<a href="#">Detailed Description</a>	22
5.3.2	<a href="#">Constructor &amp; Destructor Documentation</a>	23
5.3.2.1	<a href="#">~C_Game()</a>	23
5.3.2.2	<a href="#">C_Game()</a>	23
5.3.3	<a href="#">Member Function Documentation</a>	23
5.3.3.1	<a href="#">addShape()</a>	23
5.3.3.2	<a href="#">Clear()</a>	23
5.3.3.3	<a href="#">GetObjectiveColor()</a>	24
5.3.3.4	<a href="#">MainLoop()</a>	24

5.3.3.5	SetObjective()	24
5.4	C_GTriangle Class Reference	24
5.4.1	Detailed Description	27
5.4.2	Constructor & Destructor Documentation	27
5.4.2.1	~C_GTriangle()	28
5.4.2.2	C_GTriangle() [1/3]	28
5.4.2.3	C_GTriangle() [2/3]	28
5.4.2.4	C_GTriangle() [3/3]	28
5.4.3	Member Function Documentation	29
5.4.3.1	aCurrentAngular()	29
5.4.3.2	aGetArea()	29
5.4.3.3	aGetColor()	29
5.4.3.4	aGetPoints()	30
5.4.3.5	aGetShape()	30
5.4.3.6	aGetStatusReverse()	30
5.4.3.7	alsInShape()	30
5.4.3.8	aLeftCorner()	31
5.4.3.9	aLeftFlip()	31
5.4.3.10	aMove()	31
5.4.3.11	aReverse()	32
5.4.3.12	aRightFlip()	32
5.4.3.13	aRotate()	32
5.4.3.14	aSetPoints()	32
5.4.3.15	aToString()	33
5.4.3.16	iDraw() [1/2]	33
5.4.3.17	iDraw() [2/2]	33
5.5	C_Loader Class Reference	33
5.5.1	Detailed Description	34
5.5.2	Member Function Documentation	34
5.5.2.1	ParseFile()	34

5.6	C_Menu Class Reference . . . . .	35
5.6.1	Detailed Description . . . . .	35
5.6.2	Constructor & Destructor Documentation . . . . .	35
5.6.2.1	~C_Menu() . . . . .	35
5.6.3	Member Function Documentation . . . . .	36
5.6.3.1	AddButton() . . . . .	36
5.6.3.2	MainLoop() . . . . .	36
5.7	C_MTriangle Class Reference . . . . .	36
5.7.1	Detailed Description . . . . .	39
5.7.2	Constructor & Destructor Documentation . . . . .	39
5.7.2.1	~C_MTriangle() . . . . .	40
5.7.2.2	C_MTriangle() [1/3] . . . . .	40
5.7.2.3	C_MTriangle() [2/3] . . . . .	40
5.7.2.4	C_MTriangle() [3/3] . . . . .	40
5.7.3	Member Function Documentation . . . . .	41
5.7.3.1	aCurrentAngular() . . . . .	41
5.7.3.2	aGetArea() . . . . .	41
5.7.3.3	aGetColor() . . . . .	41
5.7.3.4	aGetPoints() . . . . .	42
5.7.3.5	aGetShape() . . . . .	42
5.7.3.6	aGetStatusReverse() . . . . .	42
5.7.3.7	alsInShape() . . . . .	42
5.7.3.8	aLeftCorner() . . . . .	43
5.7.3.9	aLeftFlip() . . . . .	43
5.7.3.10	aMove() . . . . .	43
5.7.3.11	aReverse() . . . . .	44
5.7.3.12	aRightFlip() . . . . .	44
5.7.3.13	aRotate() . . . . .	44
5.7.3.14	aSetPoints() . . . . .	44
5.7.3.15	aToString() . . . . .	45

5.7.3.16	iDraw() [1/2]	45
5.7.3.17	iDraw() [2/2]	45
5.8	C_Objective Class Reference	46
5.8.1	Detailed Description	47
5.8.2	Constructor & Destructor Documentation	47
5.8.2.1	~C_Objective()	47
5.8.2.2	C_Objective() [1/2]	47
5.8.2.3	C_Objective() [2/2]	47
5.8.3	Member Function Documentation	48
5.8.3.1	BoardCompleted()	48
5.8.3.2	Clear()	48
5.8.3.3	GetColor()	48
5.8.3.4	GetCompleted()	48
5.8.3.5	GetObjective()	49
5.8.3.6	SetObjective()	49
5.9	C_Parallelogram Class Reference	49
5.9.1	Detailed Description	52
5.9.2	Constructor & Destructor Documentation	52
5.9.2.1	~C_Parallelogram()	53
5.9.2.2	C_Parallelogram() [1/3]	53
5.9.2.3	C_Parallelogram() [2/3]	53
5.9.2.4	C_Parallelogram() [3/3]	53
5.9.3	Member Function Documentation	54
5.9.3.1	aCurrentAngular()	54
5.9.3.2	aGetArea()	54
5.9.3.3	aGetColor()	54
5.9.3.4	aGetPoints()	55
5.9.3.5	aGetShape()	55
5.9.3.6	aGetStatusReverse()	55
5.9.3.7	alsInShape()	55

5.9.3.8	<a href="#">aLeftCorner()</a>	56
5.9.3.9	<a href="#">aLeftFlip()</a>	56
5.9.3.10	<a href="#">aMove()</a>	56
5.9.3.11	<a href="#">aReverse()</a>	57
5.9.3.12	<a href="#">aRightFlip()</a>	57
5.9.3.13	<a href="#">aRotate()</a>	57
5.9.3.14	<a href="#">aSetPoints()</a>	57
5.9.3.15	<a href="#">aToString()</a>	58
5.9.3.16	<a href="#">iDraw()</a> [1/2]	58
5.9.3.17	<a href="#">iDraw()</a> [2/2]	58
5.10	<a href="#">C_Save Class Reference</a>	59
5.10.1	<a href="#">Detailed Description</a>	59
5.10.2	<a href="#">Constructor &amp; Destructor Documentation</a>	59
5.10.2.1	<a href="#">C_Save()</a>	59
5.10.3	<a href="#">Member Function Documentation</a>	59
5.10.3.1	<a href="#">Save()</a>	59
5.11	<a href="#">C_Square Class Reference</a>	60
5.11.1	<a href="#">Detailed Description</a>	63
5.11.2	<a href="#">Constructor &amp; Destructor Documentation</a>	63
5.11.2.1	<a href="#">~C_Square()</a>	64
5.11.2.2	<a href="#">C_Square()</a> [1/3]	64
5.11.2.3	<a href="#">C_Square()</a> [2/3]	64
5.11.2.4	<a href="#">C_Square()</a> [3/3]	64
5.11.3	<a href="#">Member Function Documentation</a>	65
5.11.3.1	<a href="#">aCurrentAngular()</a>	65
5.11.3.2	<a href="#">aGetArea()</a>	65
5.11.3.3	<a href="#">aGetColor()</a>	65
5.11.3.4	<a href="#">aGetPoints()</a>	66
5.11.3.5	<a href="#">aGetShape()</a>	66
5.11.3.6	<a href="#">aGetStatusReverse()</a>	66



5.11.3.7	<a href="#">alsInShape()</a> . . . . .	66
5.11.3.8	<a href="#">aLeftCorner()</a> . . . . .	67
5.11.3.9	<a href="#">aLeftFlip()</a> . . . . .	67
5.11.3.10	<a href="#">aMove()</a> . . . . .	67
5.11.3.11	<a href="#">aReverse()</a> . . . . .	68
5.11.3.12	<a href="#">aRightFlip()</a> . . . . .	68
5.11.3.13	<a href="#">aRotate()</a> . . . . .	68
5.11.3.14	<a href="#">aSetPoints()</a> . . . . .	68
5.11.3.15	<a href="#">aToString()</a> . . . . .	69
5.11.3.16	<a href="#">iDraw()</a> [1/2] . . . . .	69
5.11.3.17	<a href="#">iDraw()</a> [2/2] . . . . .	69
5.12	<a href="#">C_STriangle Class Reference</a> . . . . .	70
5.12.1	<a href="#">Detailed Description</a> . . . . .	73
5.12.2	<a href="#">Constructor &amp; Destructor Documentation</a> . . . . .	73
5.12.2.1	<a href="#">~C_STriangle()</a> . . . . .	73
5.12.2.2	<a href="#">C_STriangle()</a> [1/4] . . . . .	73
5.12.2.3	<a href="#">C_STriangle()</a> [2/4] . . . . .	73
5.12.2.4	<a href="#">C_STriangle()</a> [3/4] . . . . .	74
5.12.2.5	<a href="#">C_STriangle()</a> [4/4] . . . . .	74
5.12.3	<a href="#">Member Function Documentation</a> . . . . .	74
5.12.3.1	<a href="#">aCurrentAngular()</a> . . . . .	75
5.12.3.2	<a href="#">aGetArea()</a> . . . . .	75
5.12.3.3	<a href="#">aGetColor()</a> . . . . .	75
5.12.3.4	<a href="#">aGetPoints()</a> . . . . .	75
5.12.3.5	<a href="#">aGetShape()</a> . . . . .	76
5.12.3.6	<a href="#">aGetStatusReverse()</a> . . . . .	76
5.12.3.7	<a href="#">alsInShape()</a> . . . . .	76
5.12.3.8	<a href="#">aLeftCorner()</a> . . . . .	77
5.12.3.9	<a href="#">aLeftFlip()</a> . . . . .	77
5.12.3.10	<a href="#">aMove()</a> . . . . .	77

5.12.3.11 aReverse()	77
5.12.3.12 aRightFlip()	78
5.12.3.13 aRotate()	78
5.12.3.14 aSetPoints()	78
5.12.3.15 aToString()	78
5.12.3.16 CenterPoint()	79
5.12.3.17 GetCenterPoint()	79
5.12.3.18 GetFlip()	79
5.12.3.19 iDraw() [1/2]	80
5.12.3.20 iDraw() [2/2]	80
5.12.3.21 IsInSTriangle()	80
5.12.3.22 LeftFlip()	80
5.12.3.23 Reverse()	81
5.12.3.24 RightFlip()	81
5.12.3.25 Rotate()	81
5.13 T_Point< T >::hash_point Struct Reference	82
5.13.1 Member Function Documentation	82
5.13.1.1 operator>() [1/2]	82
5.13.1.2 operator>() [2/2]	83
5.14 I_Drawable Class Reference	83
5.14.1 Detailed Description	85
5.14.2 Constructor & Destructor Documentation	85
5.14.2.1 ~I_Drawable()	85
5.14.3 Member Function Documentation	85
5.14.3.1 iDraw() [1/2]	85
5.14.3.2 iDraw() [2/2]	85
5.15 Struct Struct Reference	86
5.15.1 Detailed Description	86
5.16 T_Point< T > Class Template Reference	86
5.16.1 Detailed Description	89

5.16.2	Constructor & Destructor Documentation . . . . .	89
5.16.2.1	T_Point() [1/4] . . . . .	89
5.16.2.2	T_Point() [2/4] . . . . .	89
5.16.2.3	T_Point() [3/4] . . . . .	90
5.16.2.4	T_Point() [4/4] . . . . .	90
5.16.2.5	~T_Point() . . . . .	90
5.16.3	Member Function Documentation . . . . .	90
5.16.3.1	operator!=(()) . . . . .	90
5.16.3.2	operator+() . . . . .	91
5.16.3.3	operator+=() . . . . .	91
5.16.3.4	operator-() . . . . .	92
5.16.3.5	operator-=() . . . . .	92
5.16.3.6	operator<() . . . . .	92
5.16.3.7	operator=() . . . . .	93
5.16.3.8	operator==(()) . . . . .	93
5.16.3.9	operator>() . . . . .	93
5.16.4	Member Data Documentation . . . . .	94
5.16.4.1	x . . . . .	94
5.16.4.2	y . . . . .	94
<b>6</b>	<b>File Documentation</b>	<b>95</b>
6.1	include/drawable/A_Shape.hpp File Reference . . . . .	95
6.1.1	Detailed Description . . . . .	96
6.2	include/drawable/C_Button.hpp File Reference . . . . .	96
6.2.1	Detailed Description . . . . .	97
6.3	include/drawable/C_Menu.hpp File Reference . . . . .	97
6.3.1	Detailed Description . . . . .	98
6.4	include/drawable/I_Drawable.h File Reference . . . . .	99
6.5	include/game/C_Game.hpp File Reference . . . . .	100
6.5.1	Detailed Description . . . . .	101
6.6	include/game/C_Objective.hpp File Reference . . . . .	101

6.6.1 Detailed Description . . . . .	102
6.7 include/parser/C_Loader.hpp File Reference . . . . .	102
6.7.1 Detailed Description . . . . .	103
6.8 include/parser/C_Save.hpp File Reference . . . . .	103
6.8.1 Detailed Description . . . . .	104
6.9 include/shape/C_GTriangle.hpp File Reference . . . . .	104
6.9.1 Detailed Description . . . . .	105
6.10 include/shape/C_MTriangle.hpp File Reference . . . . .	106
6.10.1 Detailed Description . . . . .	107
6.11 include/shape/C_Parallelogram.hpp File Reference . . . . .	107
6.11.1 Detailed Description . . . . .	108
6.12 include/shape/C_Square.hpp File Reference . . . . .	108
6.12.1 Detailed Description . . . . .	109
6.13 include/shape/C_STriangle.hpp File Reference . . . . .	109
6.13.1 Detailed Description . . . . .	110
6.14 include/utils/T_Point.hpp File Reference . . . . .	110
6.14.1 Detailed Description . . . . .	111
6.15 README.md File Reference . . . . .	111
6.16 src/drawable/A_Shape.cpp File Reference . . . . .	111
6.17 src/drawable/C_Button.cpp File Reference . . . . .	112
6.18 src/drawable/C_Menu.cpp File Reference . . . . .	112
6.19 src/drawable/I_Drawable.cpp File Reference . . . . .	113
6.20 src/game/C_Game.cpp File Reference . . . . .	113
6.21 src/game/C_Objective.cpp File Reference . . . . .	113
6.22 src/Main.cpp File Reference . . . . .	113
6.22.1 Function Documentation . . . . .	114
6.22.1.1 main() . . . . .	114
6.22.2 Variable Documentation . . . . .	114
6.22.2.1 page . . . . .	114
6.23 src/parser/C_Loader.cpp File Reference . . . . .	115
6.24 src/parser/C_Save.cpp File Reference . . . . .	115
6.25 src/shape/C_GTriangle.cpp File Reference . . . . .	116
6.26 src/shape/C_MTriangle.cpp File Reference . . . . .	116
6.27 src/shape/C_Parallelogram.cpp File Reference . . . . .	117
6.28 src/shape/C_Square.cpp File Reference . . . . .	118
6.29 src/shape/C_STriangle.cpp File Reference . . . . .	118
6.30 src/utils/T_Point.cpp File Reference . . . . .	119

# Chapter 1

## Tangram

A student project about the Tangram game made in C++

### Getting started

When you're in the root directory of this project, follow the next steps :

#### CMake

First, If you have not did it already, you can build the game by executing the following command line : `>$ cmake ./cmake-build-debug`

#### Make

Second, If you have not did it already, you can make the executable's game by executing the following command line : `>$ cd cmake-build-debug`

`>$ make`

#### Run

Before using the run command line, you have to use the two aforementioned command in the right order. If you have already did it, you can run the game by executing the following command line : `>$ ./tangram`

### How to play

Run the game with the following command line in the cmake-debug-build directory : `>$ ./tangram`

You can play now.

### Launch Button

You can create a new puzzle board if you click on the `Launch` button and use the following commands : `>mouse click left` on a shape and drag to move it.

```
>mouse click right on a shape and drag to rotate it.
>press 'Esc' to exit this mode.
press 's' to save the current board as puzzle.
>press 'd' on a shape mouseovered to rotate it 45° anti clockwise.
>press 'f' on a shape mouseovered to rotate it 45° clockwise.
>press 'r' to symmetrically reverse the shape.
>Note that last command rotates every shape to 180° except parallelogram which is
```

overturned (in a mirror fashion)

### Load Button

If you click on the `Load` button, you can load a puzzle file and try to resolve it. You can use the following commands : `>mouse click left` on a shape and drag to move it.

```
>mouse click right on a shape and drag to rotate it.
>press 'Esc' to exit this mode.
>press 'd' on a shape mouseovered to rotate it 45° anti clockwise.
>press 'f' on a shape mouseovered to rotate it 45° clockwise.
>press 'r' to symmetrically reverse the shape.
>Note that last command rotates every shape to 180° except parallelogram which is
```

overturned (in a mirror fashion)

### End Game

The game will stop when you put the last shape at the right place. You will return to the main menu. When you solve a puzzle, the last shape dropped will be displayed in white and the game will freeze a for few seconds before you return to the main menu.

### Documentation

Here you can find HTML files, LaTeX files and PDF.

#### HTML

Open with your browser

```
>$ cd doc/html
```

```
>index.html
```

## LaTeX

```
>$ cd doc/latex
```

## PDF

Open with a PDF reader

```
>$ cd doc/latex
```

refman.pdf

## Regenerate Documentation

You can generate this document as needed. If you're updating the code and the documentation, you should do execute in the root directory of this project : 

```
>$ doxygen config-file
```

If you want customize the documentation generated, you could also configurate the following file : 

```
>$ gedit config-file
```

## Regenerate LaTeX Documentation

To generate the PDF documentation, execute the following commands : 

```
>$ cd doc/latex
```

```
>$ make
```





## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

C_Button . . . . .	18
C_Game . . . . .	22
C_Loader . . . . .	33
C_Menu . . . . .	35
C_Objective . . . . .	46
C_Save . . . . .	59
T_Point< T >::hash_point . . . . .	82
I_Drawable . . . . .	83
A_Shape . . . . .	11
C_GTriangle . . . . .	24
C_MTriangle . . . . .	36
C_Parallelogram . . . . .	49
C_Square . . . . .	60
C_STriangle . . . . .	70
Struct . . . . .	86
T_Point< T > . . . . .	86
T_Point< double > . . . . .	86
T_Point< int > . . . . .	86



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">A_Shape</a>	Abstract Class of every <a href="#">A_Shape</a> . . . . .	11
<a href="#">C_Button</a>	<a href="#">C_Button</a> of the <a href="#">C_Menu</a> . . . . .	18
<a href="#">C_Game</a>	Class of the main <a href="#">C_Game</a> . . . . .	22
<a href="#">C_GTriangle</a>	Class of the greatest <a href="#">C_GTriangle</a> . . . . .	24
<a href="#">C_Loader</a>	Class of the main <a href="#">C_Loader</a> . . . . .	33
<a href="#">C_Menu</a>	<a href="#">C_Menu</a> of the game . . . . .	35
<a href="#">C_MTriangle</a>	Class of the medium <a href="#">C_MTriangle</a> . . . . .	36
<a href="#">C_Objective</a>	Class of the board <a href="#">C_Objective</a> . . . . .	46
<a href="#">C_Parallelogram</a>	Class of the parallelogram . . . . .	49
<a href="#">C_Save</a>	Class of the main Saver . . . . .	59
<a href="#">C_Square</a>	Class of the square . . . . .	60
<a href="#">C_STriangle</a>	Class of the small <a href="#">C_STriangle</a> . . . . .	70
<a href="#">T_Point&lt; T &gt;::hash_point</a>	. . . . .	82
<a href="#">I_Drawable</a>	<a href="#">I_Drawable</a> is everything to iDraw . . . . .	83
Struct	Hash a <a href="#">T_Point&lt;T&gt;</a> to hash a point with <a href="#">T_Point&lt;T&gt;</a> . . . . .	86
<a href="#">T_Point&lt; T &gt;</a>	Class of a <a href="#">T_Point</a> . . . . .	86



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

include/drawable/A_Shape.hpp	
Abstract Class <a href="#">A_Shape</a> of every shape in Tangram . . . . .	95
include/drawable/C_Button.hpp	
Every mButtons of menu . . . . .	96
include/drawable/C_Menu.hpp	
<a href="#">C_Menu</a> of the Tangram's <a href="#">C_Game</a> . . . . .	97
include/drawable/I_Drawable.h	
include/game/C_Game.hpp	
Main <a href="#">C_Game</a> of the Tangram . . . . .	100
include/game/C_Objective.hpp	
<a href="#">C_Objective</a> of the Tangram's board . . . . .	101
include/parser/C_Loader.hpp	
Load a board of Tangram . . . . .	102
include/parser/C_Save.hpp	
<a href="#">C_Save</a> a board of Tangram . . . . .	103
include/shape/C_GTriangle.hpp	
<a href="#">A_Shape</a> of Great Triangle . . . . .	104
include/shape/C_MTriangle.hpp	
<a href="#">A_Shape</a> of Medium Triangle . . . . .	106
include/shape/C_Parallelogram.hpp	
<a href="#">A_Shape</a> of <a href="#">C_Parallelogram</a> . . . . .	107
include/shape/C_Square.hpp	
<a href="#">A_Shape</a> of <a href="#">C_Square</a> . . . . .	108
include/shape/C_STriangle.hpp	
<a href="#">A_Shape</a> of Small Triangle . . . . .	109
include/utils/T_Point.hpp	
<a href="#">T_Point</a> for every shape and menu . . . . .	110
src/Main.cpp	
src/drawable/A_Shape.cpp	
src/drawable/C_Button.cpp	
src/drawable/C_Menu.cpp	
src/drawable/I_Drawable.cpp	
src/game/C_Game.cpp	
src/game/C_Objective.cpp	
src/parser/C_Loader.cpp	

<a href="#">src/parser/C_Save.cpp</a>	115
<a href="#">src/shape/C_GTriangle.cpp</a>	116
<a href="#">src/shape/C_MTriangle.cpp</a>	116
<a href="#">src/shape/C_Parallelogram.cpp</a>	117
<a href="#">src/shape/C_Square.cpp</a>	118
<a href="#">src/shape/C_STriangle.cpp</a>	118
<a href="#">src/utls/T_Point.cpp</a>	119

## Chapter 5

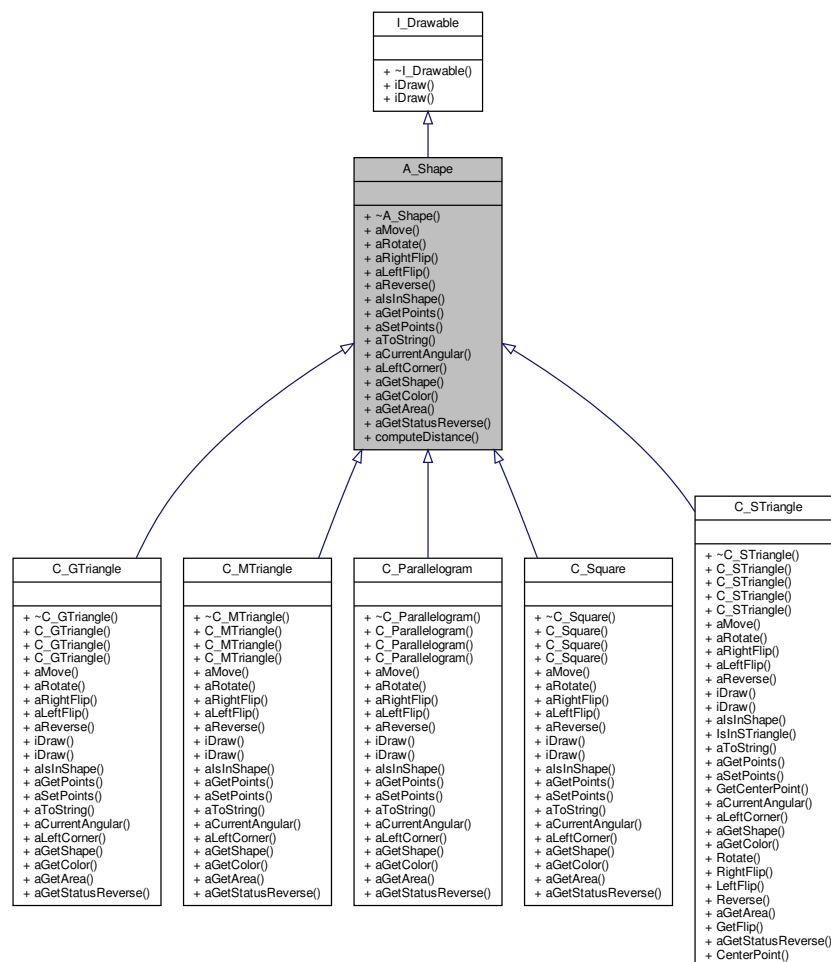
# Class Documentation

### 5.1 A\_Shape Class Reference

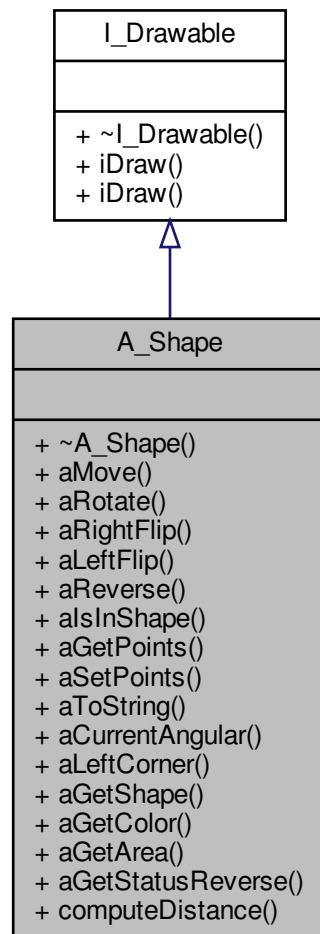
Abstract Class of every [A\\_Shape](#).

```
#include <A_Shape.hpp>
```

Inheritance diagram for A\_Shape:



Collaboration diagram for A\_Shape:



## Public Member Functions

- virtual `~A_Shape ()=0`  
*Destructor of Abstract `A_Shape`.*
- virtual void `aMove (const T_Point< double > &translation)=0`  
*Pure virtual function. Move the `A_Shape` by point translation.*
- virtual void `aRotate (double angular)=0`  
*Pure virtual function. Rotate the `A_Shape` with specified angular.*
- virtual void `aRightFlip ()=0`  
*Pure virtual function. Flip the figure as 45° clock (Pi/4)*
- virtual void `aLeftFlip ()=0`  
*Pure virtual function. Flip the figure as 45° anti clock (Pi/4)*
- virtual void `aReverse ()=0`  
*Pure virtual function. Reverse the shape as symmetry.*
- virtual bool `aIsInShape (const T_Point< double > &point)=0`



- Pure virtual function. Check if a point is in this shape.*
- virtual std::vector< T\_Point< double > > aGetPoints ()=0
- Pure virtual function. Get all mPoints of this shape.*
- virtual bool aSetPoints (const T\_Point< double > &ref, const T\_Point< double > &changed)=0
- Pure virtual function. Get all mPoints of this shape.*
- virtual std::string aToString ()=0
- Pure virtual function. Convert all data of A\_Shape in a string.*
- virtual double aCurrentAngular ()=0
- Pure virtual function. Get the current angular of a A\_Shape.*
- virtual T\_Point< double > aLeftCorner ()=0
- Pure virtual function. Take the point at left top corner of a A\_Shape.*
- virtual std::string aGetShape ()=0
- Pure virtual function. Get the A\_Shape type.*
- virtual MLV\_Color aGetColor ()=0
- Pure virtual function. Get the color of a A\_Shape.*
- virtual double aGetArea ()=0
- Pure virtual function. Get the area of a A\_Shape.*
- virtual bool aGetStatusReverse () const =0
- Get the status of shape reversed or not.*

## Static Public Member Functions

- static double computeDistance (const T\_Point< double > &point1, const T\_Point< double > &point2)
- Compute distance between 2 mPoints.*

### 5.1.1 Detailed Description

Abstract Class of every A\_Shape.

This class manage everything other shape (C\_STriangle, C\_MTriangle, C\_GTriangle, C\_Square, C\_Parallelogram)

### 5.1.2 Constructor & Destructor Documentation

#### 5.1.2.1 ~A\_Shape()

```
A_Shape::~A_Shape ( ) [pure virtual], [default]
```

Destructor of Abstract A\_Shape.

### 5.1.3 Member Function Documentation

#### 5.1.3.1 aCurrentAngular()

```
virtual double A_Shape::aCurrentAngular ( ) [pure virtual]
```

Pure virtual function. Get the current angular of a [A\\_Shape](#).

##### Returns

Return the current angular of a [A\\_Shape](#) as double

Implemented in [C\\_STriangle](#), [C\\_Parallelogram](#), [C\\_MTriangle](#), [C\\_Square](#), and [C\\_GTriangle](#).

#### 5.1.3.2 aGetArea()

```
virtual double A_Shape::aGetArea ( ) [pure virtual]
```

Pure virtual function. Get the area of a [A\\_Shape](#).

##### Returns

Return the area of a [A\\_Shape](#)

Implemented in [C\\_STriangle](#), [C\\_Parallelogram](#), [C\\_MTriangle](#), [C\\_Square](#), and [C\\_GTriangle](#).

#### 5.1.3.3 aGetColor()

```
virtual MLV_Color A_Shape::aGetColor ( ) [pure virtual]
```

Pure virtual function. Get the color of a [A\\_Shape](#).

##### Returns

Return the MLV\_Color of a [A\\_Shape](#)

Implemented in [C\\_STriangle](#), [C\\_Parallelogram](#), [C\\_MTriangle](#), [C\\_Square](#), and [C\\_GTriangle](#).

#### 5.1.3.4 aGetPoints()

```
virtual std::vector<T_Point<double>> A_Shape::aGetPoints ( ) [pure virtual]
```

Pure virtual function. Get all mPoints of this shape.

##### Returns

Return a vector of mPoints of this shape

Implemented in [C\\_STriangle](#), [C\\_Parallelogram](#), [C\\_MTriangle](#), [C\\_Square](#), and [C\\_GTriangle](#).

#### 5.1.3.5 aGetShape()

```
virtual std::string A_Shape::aGetShape ( ) [pure virtual]
```

Pure virtual function. Get the [A\\_Shape](#) type.

##### Returns

Return as string a [A\\_Shape](#) type

Implemented in [C\\_STriangle](#), [C\\_Parallelogram](#), [C\\_MTriangle](#), [C\\_Square](#), and [C\\_GTriangle](#).

#### 5.1.3.6 aGetStatusReverse()

```
virtual bool A_Shape::aGetStatusReverse ( ) const [pure virtual]
```

Get the status of shape reversed or not.

##### Returns

Return true if the shape got reversed, false otherwise

Implemented in [C\\_STriangle](#), [C\\_Parallelogram](#), [C\\_MTriangle](#), [C\\_Square](#), and [C\\_GTriangle](#).

#### 5.1.3.7 aIsInShape()

```
virtual bool A_Shape::aIsInShape (
    const T_Point< double > & point ) [pure virtual]
```

Pure virtual function. Check if a point is in this shape.

##### Parameters

<i>point</i>	: <a href="#">T_Point</a> to check
--------------	------------------------------------

##### Returns

true if Click is in this shape, false if not

Implemented in [C\\_STriangle](#), [C\\_Parallelogram](#), [C\\_MTriangle](#), [C\\_Square](#), and [C\\_GTriangle](#).

#### 5.1.3.8 aLeftCorner()

```
virtual T_Point<double> A_Shape::aLeftCorner ( ) [pure virtual]
```

Pure virtual function. Take the point at left top corner of a [A\\_Shape](#).

##### Returns

Return the point at left top corner

Implemented in [C\\_STriangle](#), [C\\_Parallelogram](#), [C\\_MTriangle](#), [C\\_Square](#), and [C\\_GTriangle](#).

#### 5.1.3.9 aLeftFlip()

```
virtual void A_Shape::aLeftFlip ( ) [pure virtual]
```

Pure virtual function. Flip the figure as 45° anti clock (Pi/4)

Implemented in [C\\_STriangle](#), [C\\_Parallelogram](#), [C\\_MTriangle](#), [C\\_Square](#), and [C\\_GTriangle](#).

#### 5.1.3.10 aMove()

```
virtual void A_Shape::aMove (
    const T_Point< double > & translation ) [pure virtual]
```

Pure virtual function. Move the [A\\_Shape](#) by point translation.

##### Parameters

<i>translation</i>	: Every mPoints of this shape will be translate by this __Parameter
--------------------	---

Implemented in [C\\_STriangle](#), [C\\_Parallelogram](#), [C\\_MTriangle](#), [C\\_Square](#), and [C\\_GTriangle](#).

#### 5.1.3.11 aReverse()

```
virtual void A_Shape::aReverse ( ) [pure virtual]
```

Pure virtual function. Reverse the shape as symmetry.

Implemented in [C\\_STriangle](#), [C\\_Parallelogram](#), [C\\_MTriangle](#), [C\\_Square](#), and [C\\_GTriangle](#).

## 5.1.3.12 aRightFlip()

```
virtual void A_Shape::aRightFlip ( ) [pure virtual]
```

Pure virtual function. Flip the figure as 45° clock (Pi/4)

Implemented in [C\\_STriangle](#), [C\\_Parallelogram](#), [C\\_MTriangle](#), [C\\_Square](#), and [C\\_GTriangle](#).

## 5.1.3.13 aRotate()

```
virtual void A_Shape::aRotate (
    double angular ) [pure virtual]
```

Pure virtual function. Rotate the [A\\_Shape](#) with specified angular.

## Parameters

<i>angular</i>	: This angular should be between (0, 2PI)
----------------	---

Implemented in [C\\_STriangle](#), [C\\_Parallelogram](#), [C\\_MTriangle](#), [C\\_Square](#), and [C\\_GTriangle](#).

## 5.1.3.14 aSetPoints()

```
virtual bool A_Shape::aSetPoints (
    const T_Point< double > & ref,
    const T_Point< double > & changed ) [pure virtual]
```

Pure virtual function. Get all mPoints of this shape.

## Returns

Return a vector of mPoints of this shape

Implemented in [C\\_STriangle](#), [C\\_Parallelogram](#), [C\\_MTriangle](#), [C\\_Square](#), and [C\\_GTriangle](#).

## 5.1.3.15 aToString()

```
virtual std::string A_Shape::aToString ( ) [pure virtual]
```

Pure virtual function. Convert all data of [A\\_Shape](#) in a string.

## Returns

Return a string which contains every mPoints of this shape

Implemented in [C\\_STriangle](#), [C\\_Parallelogram](#), [C\\_MTriangle](#), [C\\_Square](#), and [C\\_GTriangle](#).

### 5.1.3.16 computeDistance()

```
static double A_Shape::computeDistance (
    const T_Point< double > & point1,
    const T_Point< double > & point2 ) [inline], [static]
```

Compute distance between 2 mPoints.

#### Parameters

<i>point1</i>	: First point
<i>point2</i>	: Second point

#### Returns

Return the distance between these two mPoints

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

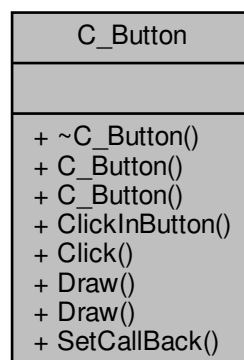
- [include/drawable/A\\_Shape.hpp](#)
- [src/drawable/A\\_Shape.cpp](#)

## 5.2 C\_Button Class Reference

[C\\_Button](#) of the [C\\_Menu](#).

```
#include <C_Button.hpp>
```

Collaboration diagram for C\_Button:



## Public Member Functions

- [~C\\_Button](#) ()  
*Class methods.*
- [C\\_Button](#) (const [T\\_Point](#)< int > &point, const [T\\_Point](#)< int > &sizing, std::string text)  
*Constructor of a [C\\_Button](#).*
- [C\\_Button](#) (const [T\\_Point](#)< int > &point, const [T\\_Point](#)< int > &sizing, std::string text, std::function< int(int)> callback)  
*Constructor of a [C\\_Button](#).*
- bool [ClickInButton](#) (const [T\\_Point](#)< int > &click)  
*Check if a Click is in the button.*
- int [Click](#) (int)  
*Define a value about a Click.*
- void [Draw](#) ()  
*Draw the button.*
- void [Draw](#) (MLV\_Color color)  
*Draw the button with specific color.*
- void [SetCallBack](#) (std::function< int(int)> callback)  
*Set a callback for a button.*

### 5.2.1 Detailed Description

[C\\_Button](#) of the [C\\_Menu](#).

This class manage all mButtons of the menu

### 5.2.2 Constructor & Destructor Documentation

#### 5.2.2.1 ~C\_Button()

```
C_Button::~~C_Button ( ) [default]
```

Class methods.

Destructor of the [C\\_Button](#)

#### 5.2.2.2 C\_Button() [1/2]

```
C_Button::C_Button (
    const T\_Point< int > & point,
    const T\_Point< int > & sizing,
    std::string text )
```

Constructor of a [C\\_Button](#).

**Parameters**

<i>point</i>	: Top left point position of the button
<i>sizing</i>	: Sizing of the button, (width , height)
<i>text</i>	: Text of the button

**5.2.2.3 C\_Button()** [2/2]

```

C_Button::C_Button (
    const T_Point< int > & point,
    const T_Point< int > & sizing,
    std::string text,
    std::function< int(int)> callback )

```

Constructor of a [C\\_Button](#).

**Parameters**

<i>point</i>	: Top left point position of the button
<i>sizing</i>	: Sizing of the button, (width , height)
<i>text</i>	: Text of the button
<i>callback</i>	: Pointer of function for callback

**5.2.3 Member Function Documentation****5.2.3.1 Click()**

```

int C_Button::Click (
    int val )

```

Define a value about a Click.

**Returns**

Return a value about a Click

**5.2.3.2 ClickInButton()**

```

bool C_Button::ClickInButton (
    const T_Point< int > & click )

```

Check if a Click is in the button.



**Parameters**

<i>click</i>	: <a href="#">T_Point</a> to check
--------------	------------------------------------

**Returns**

True if the Click is in this button, false if not

**5.2.3.3 Draw()** [1/2]

```
void C_Button::Draw ( )
```

Draw the button.

**5.2.3.4 Draw()** [2/2]

```
void C_Button::Draw (
    MLV_Color color )
```

Draw the button with specific color.

**Parameters**

<i>color</i>	: MLV_Color needed to draw the button
--------------	---------------------------------------

**5.2.3.5 SetCallBack()**

```
void C_Button::SetCallBack (
    std::function< int(int)> callback )
```

Set a callback for a button.

**Parameters**

<i>callback</i>	: Requires a pointer of function for set the callback
-----------------	---

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

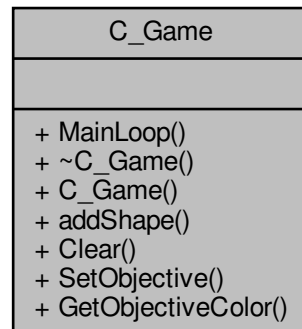
- [include/drawable/C\\_Button.hpp](#)
- [src/drawable/C\\_Button.cpp](#)

## 5.3 C\_Game Class Reference

Class of the main [C\\_Game](#).

```
#include <C_Game.hpp>
```

Collaboration diagram for C\_Game:



### Public Member Functions

- void [MainLoop](#) ()  
*Main loop of the game.*
- [~C\\_Game](#) ()  
*Destructor of the game.*
- [C\\_Game](#) (int w, int h)  
*Constructor of the game, initialize a game with an sizing.*
- void [addShape](#) (std::shared\_ptr< [A\\_Shape](#) > s)  
*Add a shape in the game.*
- void [Clear](#) ()  
*Clear the game / the board and the mObjective.*
- void [SetObjective](#) (const std::vector< std::shared\_ptr< [A\\_Shape](#) >> &vec\_objective)  
*Set the mObjective of the game.*
- MLV\_Color [GetObjectiveColor](#) ()  
*Get the mColor of the mObjective of the game.*

#### 5.3.1 Detailed Description

Class of the main [C\\_Game](#).

This class manage everything about the main game

## 5.3.2 Constructor & Destructor Documentation

### 5.3.2.1 ~C\_Game()

```
C_Game::~~C_Game ( )
```

Destructor of the game.

### 5.3.2.2 C\_Game()

```
C_Game::C_Game (
    int w,
    int h )
```

Constructor of the game, initialize a game with an sizing.

#### Parameters

<i>w</i>	: Width of the window
<i>h</i>	: Height of the window

## 5.3.3 Member Function Documentation

### 5.3.3.1 addShape()

```
void C_Game::addShape (
    std::shared_ptr< A_Shape > s )
```

Add a shape in the game.

#### Parameters

<i>s</i>	: <a href="#">A_Shape</a> to add
----------	----------------------------------

### 5.3.3.2 Clear()

```
void C_Game::Clear ( )
```

Clear the game / the board and the mObjective.

#### 5.3.3.3 GetObjectiveColor()

```
MLV_Color C_Game::GetObjectiveColor ( )
```

Get the mColor of the mObjective of the game.

#### Returns

Return the mColor of the mObjective of the game

#### 5.3.3.4 MainLoop()

```
void C_Game::MainLoop ( )
```

Main loop of the game.

#### 5.3.3.5 SetObjective()

```
void C_Game::SetObjective (
    const std::vector< std::shared_ptr< A_Shape >> & vec_objective )
```

Set the mObjective of the game.

#### Parameters

<code>vec_objective</code>	: Vector of <a href="#">C_Objective</a> for new game;
----------------------------	---

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

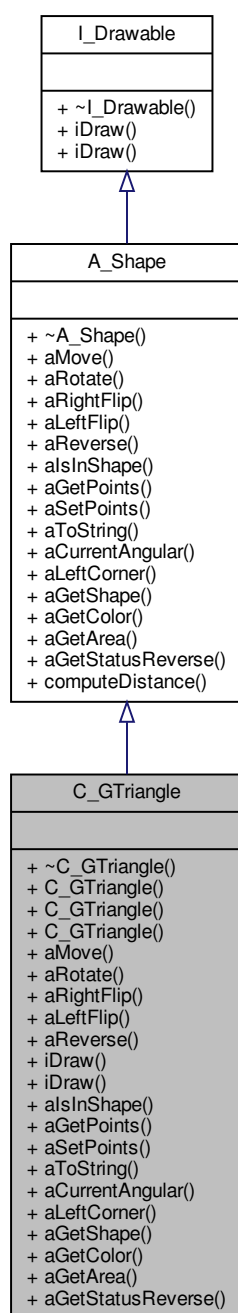
- [include/game/C\\_Game.hpp](#)
- [src/game/C\\_Game.cpp](#)

## 5.4 C\_GTriangle Class Reference

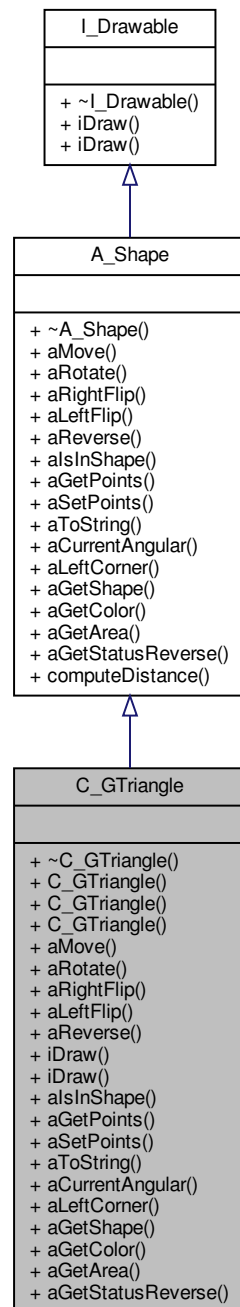
Class of the greatest [C\\_GTriangle](#).

```
#include <C_GTriangle.hpp>
```

Inheritance diagram for C\_GTriangle:



Collaboration diagram for C\_GTriangle:



## Public Member Functions

- `~C_GTriangle()` override  
*Destructor of `C_GTriangle`.*
- `C_GTriangle` (MLV\_Color color=MLV\_COLOR\_RED)  
*Constructor by default of `C_GTriangle`, make a `C_GTriangle` as default.*
- `C_GTriangle` (const std::vector< `C_STriangle` > &triangle, MLV\_Color color=MLV\_COLOR\_RED)

- Constructor of *C\_GTriangle*, requires a vector of triangles.

  - *C\_GTriangle* (const *T\_Point*< double > &origin, double angular=0.0, MLV\_Color color=MLV\_COLOR\_RED)

Constructor of *C\_GTriangle*, calls the delegate Default Constructor.
- void *aMove* (const *T\_Point*< double > &translation) override

Move the *C\_GTriangle* by point translation.
- void *aRotate* (double angular) override

Rotate the *C\_GTriangle* with specified angular.
- void *aRightFlip* () override

Flip the figure as 45° clock.
- void *aLeftFlip* () override

Flip the figure as 45° anti clock.
- void *aReverse* () override

Reverse the figure as symmetry.
- void *iDraw* () override

Draw this shape on IHM.
- void *iDraw* (MLV\_Color color) override

Draw this shape on IHM with a specific mColor.
- bool *alsInShape* (const *T\_Point*< double > &click) override

Check if a point is in this shape.
- std::vector< *T\_Point*< double > > *aGetPoints* () override

Get mPoints of this shape.
- bool *aSetPoints* (const *T\_Point*< double > &ref, const *T\_Point*< double > &changed) override

Set mPoints of this shape.
- std::string *aToString* () override

Convert all data of *C\_GTriangle* in a string.
- double *aCurrentAngular* () override

Get the current angular of this shape.
- *T\_Point*< double > *aLeftCorner* () override

Take the point at left top corner.
- std::string *aGetShape* () override

Get the shape type.
- MLV\_Color *aGetColor* () override

Get the color of the shape.
- double *aGetArea* () override

Get the area of the shape.
- bool *aGetStatusReverse* () const override

Get the status of shape reversed or not.

## Additional Inherited Members

### 5.4.1 Detailed Description

Class of the greatest *C\_GTriangle*.

This class manage everything about the greatest G\_GTriangle

### 5.4.2 Constructor & Destructor Documentation

#### 5.4.2.1 ~C\_GTriangle()

```
C_GTriangle::~C_GTriangle ( ) [override]
```

Destructor of [C\\_GTriangle](#).

#### 5.4.2.2 C\_GTriangle() [1/3]

```
C_GTriangle::C_GTriangle (
    MLV_Color color = MLV_COLOR_RED ) [explicit]
```

Constructor by default of [C\\_GTriangle](#), make a [C\\_GTriangle](#) as default.

##### Parameters

<i>color</i>	: Optional __Parameter, mColor of this shape
--------------	--

#### 5.4.2.3 C\_GTriangle() [2/3]

```
C_GTriangle::C_GTriangle (
    const std::vector< C_STriangle > & triangle,
    MLV_Color color = MLV_COLOR_RED ) [explicit]
```

Constructor of [C\\_GTriangle](#), requires a vector of triangles.

##### Parameters

<i>triangle</i>	: The <a href="#">C_GTriangle</a> will created with a vector of <a href="#">C_STriangle</a> (4)
<i>color</i>	: Optional __Parameter, mColor of this shape

#### 5.4.2.4 C\_GTriangle() [3/3]

```
C_GTriangle::C_GTriangle (
    const T_Point< double > & origin,
    double angular = 0.0,
    MLV_Color color = MLV_COLOR_RED ) [explicit]
```

Constructor of [C\\_GTriangle](#), calls the delegate Default Constructor.

##### Parameters

<i>origin</i>	: shifts the figure of a translation of the origin
<i>angular</i>	: Optional __Parameter (angular=0.0 as default), aRotate the figure with an angular
<i>color</i>	: Optional __Parameter, mColor of this shape



### 5.4.3 Member Function Documentation

#### 5.4.3.1 aCurrentAngular()

```
double C_GTriangle::aCurrentAngular ( ) [override], [virtual]
```

Get the current angular of this shape.

**Returns**

Implements [A\\_Shape](#).

#### 5.4.3.2 aGetArea()

```
double C_GTriangle::aGetArea ( ) [override], [virtual]
```

Get the area of the shape.

**Returns**

Return the area of the shape

Implements [A\\_Shape](#).

#### 5.4.3.3 aGetColor()

```
MLV_Color C_GTriangle::aGetColor ( ) [override], [virtual]
```

Get the color of the shape.

**Returns**

Return the MLV\_Color of the shape

Implements [A\\_Shape](#).

#### 5.4.3.4 aGetPoints()

```
std::vector< T_Point< double > > C_GTriangle::aGetPoints ( ) [override], [virtual]
```

Get mPoints of this shape.

##### Returns

Return a vector of mPoints of this shape

Implements [A\\_Shape](#).

#### 5.4.3.5 aGetShape()

```
std::string C_GTriangle::aGetShape ( ) [override], [virtual]
```

Get the shape type.

##### Returns

Return as string the shape type

Implements [A\\_Shape](#).

#### 5.4.3.6 aGetStatusReverse()

```
bool C_GTriangle::aGetStatusReverse ( ) const [override], [virtual]
```

Get the status of shape reversed or not.

##### Returns

Return true if the shape got reversed, false otherwise

Implements [A\\_Shape](#).

#### 5.4.3.7 aIsInShape()

```
bool C_GTriangle::aIsInShape (
    const T_Point< double > & click ) [override], [virtual]
```

Check if a point is in this shape.

## Parameters

<i>click</i>	: <a href="#">T_Point</a> to check
--------------	------------------------------------

## Returns

true if Click is in this shape, false if not

Implements [A\\_Shape](#).

## 5.4.3.8 aLeftCorner()

```
T\_Point< double > C_GTriangle::aLeftCorner ( ) [override], [virtual]
```

Take the point at left top corner.

## Returns

Return the point at left top corner

Implements [A\\_Shape](#).

## 5.4.3.9 aLeftFlip()

```
void C_GTriangle::aLeftFlip ( ) [override], [virtual]
```

Flip the figure as 45° anti clock.

Implements [A\\_Shape](#).

## 5.4.3.10 aMove()

```
void C_GTriangle::aMove (
    const T\_Point< double > & translation ) [override], [virtual]
```

Move the [C\\_GTriangle](#) by point translation.

## Parameters

<i>translation</i>	: Every mPoints of this shape will be translate by this __Parameter
--------------------	---

Implements [A\\_Shape](#).

#### 5.4.3.11 aReverse()

```
void C_GTriangle::aReverse ( ) [override], [virtual]
```

Reverse the figure as symmetry.

Implements [A\\_Shape](#).

#### 5.4.3.12 aRightFlip()

```
void C_GTriangle::aRightFlip ( ) [override], [virtual]
```

Flip the figure as 45° clock.

Implements [A\\_Shape](#).

#### 5.4.3.13 aRotate()

```
void C_GTriangle::aRotate (
    double angular ) [override], [virtual]
```

Rotate the [C\\_GTriangle](#) with specified angular.

##### Parameters

<i>angular</i>	: This angular should be between (0, 2PI)
----------------	---

Implements [A\\_Shape](#).

#### 5.4.3.14 aSetPoints()

```
bool C_GTriangle::aSetPoints (
    const T\_Point< double > & ref,
    const T\_Point< double > & changed ) [override], [virtual]
```

Set mPoints of this shape.

##### Returns

Return a true if something has been changed, false either

Implements [A\\_Shape](#).

## 5.4.3.15 aToString()

```
std::string C_GTriangle::aToString ( ) [override], [virtual]
```

Convert all data of [C\\_GTriangle](#) in a string.

## Returns

Return a string which contains every mPoints of this shape

Implements [A\\_Shape](#).

## 5.4.3.16 iDraw() [1/2]

```
void C_GTriangle::iDraw ( ) [override], [virtual]
```

Draw this shape on IHM.

Implements [I\\_Drawable](#).

## 5.4.3.17 iDraw() [2/2]

```
void C_GTriangle::iDraw (
    MLV_Color color ) [override], [virtual]
```

Draw this shape on IHM with a specific mColor.

## Parameters

<i>color</i>	Color used to __Draw the shape
--------------	--------------------------------

Implements [I\\_Drawable](#).

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

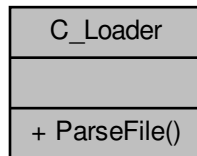
- include/shape/[C\\_GTriangle.hpp](#)
- src/shape/[C\\_GTriangle.cpp](#)

## 5.5 C\_Loader Class Reference

Class of the main [C\\_Loader](#).

```
#include <C_Loader.hpp>
```

Collaboration diagram for C\_Loader:



## Static Public Member Functions

- static bool [ParseFile](#) (const std::string &filename, [C\\_Game](#) &game)  
*Parse a file to make a board.*

### 5.5.1 Detailed Description

Class of the main [C\\_Loader](#).

This class manage everything about the loader

### 5.5.2 Member Function Documentation

#### 5.5.2.1 ParseFile()

```
bool C_Loader::ParseFile (
    const std::string & filename,
    C\_Game & game ) [static]
```

Parse a file to make a board.

#### Parameters

<i>filename</i>	: name of the file, this file should be located in this directory ./Tangram/extern/board/
<i>game</i>	: The current game / board

#### Returns

True if the game has been created, false if not

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

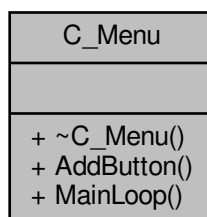
- [include/parser/C\\_Loader.hpp](#)
- [src/parser/C\\_Loader.cpp](#)

## 5.6 C\_Menu Class Reference

[C\\_Menu](#) of the game.

```
#include <C_Menu.hpp>
```

Collaboration diagram for [C\\_Menu](#):



### Public Member Functions

- [~C\\_Menu](#) ()
- void [AddButton](#) (const [C\\_Button](#) &button)  
*Add a button in the [C\\_Menu](#).*
- void [MainLoop](#) ()  
*Main loop of the [C\\_Menu](#).*

#### 5.6.1 Detailed Description

[C\\_Menu](#) of the game.

This class manage everything about Tangram's menu

#### 5.6.2 Constructor & Destructor Documentation

##### 5.6.2.1 ~C\_Menu()

```
C_Menu::~~C_Menu ( )
```

### 5.6.3 Member Function Documentation

#### 5.6.3.1 AddButton()

```
void C_Menu::AddButton (
    const C_Button & button )
```

Add a button in the [C\\_Menu](#).

##### Parameters

<i>button</i>	: <a href="#">C_Button</a> to add
---------------	-----------------------------------

#### 5.6.3.2 MainLoop()

```
void C_Menu::MainLoop ( )
```

Main loop of the [C\\_Menu](#).

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

- [include/drawable/C\\_Menu.hpp](#)
- [src/drawable/C\\_Menu.cpp](#)

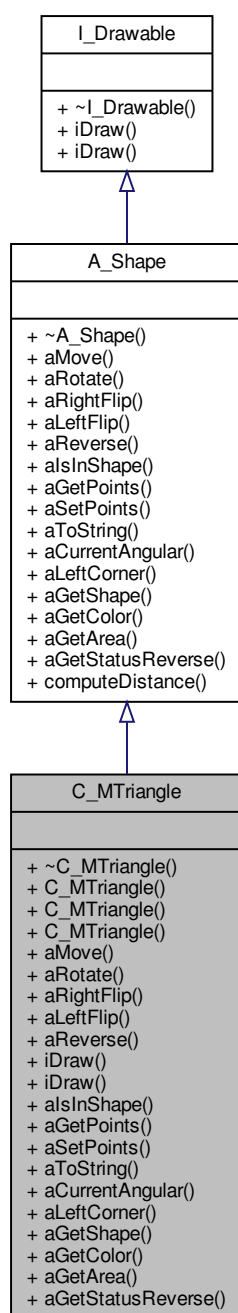
## 5.7 C\_MTriangle Class Reference

Class of the medium [C\\_MTriangle](#).

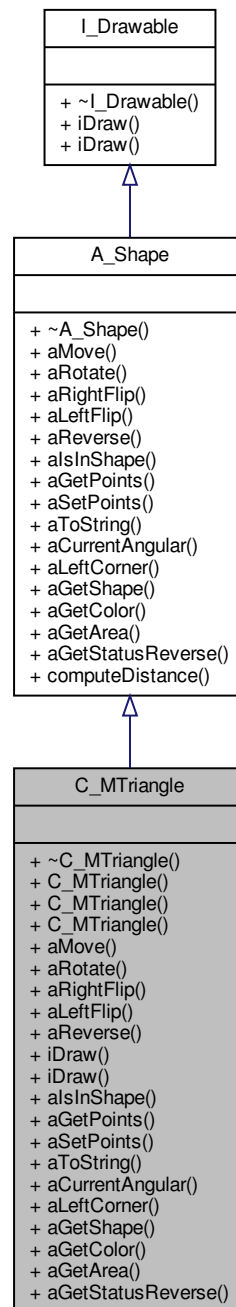
```
#include <C_MTriangle.hpp>
```



Inheritance diagram for C\_MTriangle:



Collaboration diagram for C\_MTriangle:



## Public Member Functions

- `~C_MTriangle()` override  
*Destructor of `C_MTriangle`.*
- `C_MTriangle` (MLV\_Color color=MLV\_COLOR\_ORANGE)  
*Constructor by default of `C_MTriangle`, make a `C_MTriangle` as default.*
- `C_MTriangle` (const std::vector< `C_STriangle` > &triangle, MLV\_Color color=MLV\_COLOR\_ORANGE)

Constructor of [C\\_MTriangle](#), requires a vector of [STriangles](#).

- [C\\_MTriangle](#) (const [T\\_Point](#)< double > &origin, double angular=0.0, MLV\_Color color=MLV\_COLOR\_ORANGE)

Constructor of [C\\_MTriangle](#), calls the delegate Default Constructor.

- void [aMove](#) (const [T\\_Point](#)< double > &translation) override

Move the [C\\_MTriangle](#) by point translation.

- void [aRotate](#) (double angular) override

Rotate the [C\\_MTriangle](#) with specified angular.

- void [aRightFlip](#) () override

Flip the figure as 45 ° clock.

- void [aLeftFlip](#) () override

Flip the figure as 45 ° anti clock.

- void [aReverse](#) () override

Reverse the figure as symmetry.

- void [iDraw](#) () override

Draw this shape on IHM.

- void [iDraw](#) (MLV\_Color color) override

Draw this shape on IHM with specific color.

- bool [aIsInShape](#) (const [T\\_Point](#)< double > &click) override

Check if a point is in this shape.

- std::vector< [T\\_Point](#)< double > > [aGetPoints](#) () override

Get mPoints of this shape.

- bool [aSetPoints](#) (const [T\\_Point](#)< double > &ref, const [T\\_Point](#)< double > &changed) override

Set a point to another one.

- std::string [aToString](#) () override

Convert all data of [C\\_MTriangle](#) in a string.

- double [aCurrentAngular](#) () override

Get the current angular of this shape.

- [T\\_Point](#)< double > [aLeftCorner](#) () override

Take the point at left top corner.

- std::string [aGetShape](#) () override

Get the shape type.

- MLV\_Color [aGetColor](#) () override

Get the color of the shape.

- double [aGetArea](#) () override

Get the area of the shape.

- bool [aGetStatusReverse](#) () const override

Get the status of shape reversed or not.

## Additional Inherited Members

### 5.7.1 Detailed Description

Class of the medium [C\\_MTriangle](#).

This class manage everything about the medium [C\\_MTriangle](#)

### 5.7.2 Constructor & Destructor Documentation

### 5.7.2.1 ~C\_MTriangle()

```
C_MTriangle::~C_MTriangle ( ) [override]
```

Destructor of [C\\_MTriangle](#).

### 5.7.2.2 C\_MTriangle() [1/3]

```
C_MTriangle::C_MTriangle (
    MLV_Color color = MLV_COLOR_ORANGE ) [explicit]
```

Constructor by default of [C\\_MTriangle](#), make a [C\\_MTriangle](#) as default.

#### Parameters

<i>color</i>	: Optional __Parameter, mColor of this shape
--------------	--

### 5.7.2.3 C\_MTriangle() [2/3]

```
C_MTriangle::C_MTriangle (
    const std::vector< C_STriangle > & triangle,
    MLV_Color color = MLV_COLOR_ORANGE ) [explicit]
```

Constructor of [C\\_MTriangle](#), requires a vector of STriangles.

#### Parameters

<i>triangle</i>	: The <a href="#">C_MTriangle</a> will created with a vector of <a href="#">C_STriangle</a> (4)
<i>color</i>	: Optional __Parameter, mColor of this shape

### 5.7.2.4 C\_MTriangle() [3/3]

```
C_MTriangle::C_MTriangle (
    const T_Point< double > & origin,
    double angular = 0.0,
    MLV_Color color = MLV_COLOR_ORANGE ) [explicit]
```

Constructor of [C\\_MTriangle](#), calls the delegate Default Constructor.

#### Parameters

<i>origin</i>	: shifts the figure of a translation of the origin
<i>angular</i>	: Optional __Parameter (angular=0.0 as default), aRotate the figure with an angular
<i>color</i>	: Optional __Parameter, mColor of this shape

### 5.7.3 Member Function Documentation

#### 5.7.3.1 aCurrentAngular()

```
double C_MTriangle::aCurrentAngular ( ) [override], [virtual]
```

Get the current angular of this shape.

**Returns**

Implements [A\\_Shape](#).

#### 5.7.3.2 aGetArea()

```
double C_MTriangle::aGetArea ( ) [override], [virtual]
```

Get the area of the shape.

**Returns**

Return the area of the shape

Implements [A\\_Shape](#).

#### 5.7.3.3 aGetColor()

```
MLV_Color C_MTriangle::aGetColor ( ) [override], [virtual]
```

Get the color of the shape.

**Returns**

Return the MLV\_Color of the shape

Implements [A\\_Shape](#).

#### 5.7.3.4 aGetPoints()

```
std::vector< T_Point< double > > C_MTriangle::aGetPoints ( ) [override], [virtual]
```

Get mPoints of this shape.

##### Returns

Return a vector of mPoints of this shape

Implements [A\\_Shape](#).

#### 5.7.3.5 aGetShape()

```
std::string C_MTriangle::aGetShape ( ) [override], [virtual]
```

Get the shape type.

##### Returns

Return as string the shape type

Implements [A\\_Shape](#).

#### 5.7.3.6 aGetStatusReverse()

```
bool C_MTriangle::aGetStatusReverse ( ) const [override], [virtual]
```

Get the status of shape reversed or not.

##### Returns

Return true if the shape got reversed, false otherwise

Implements [A\\_Shape](#).

#### 5.7.3.7 aIsInShape()

```
bool C_MTriangle::aIsInShape (
    const T_Point< double > & click ) [override], [virtual]
```

Check if a point is in this shape.

## Parameters

<i>click</i>	: <a href="#">T_Point</a> to check
--------------	------------------------------------

## Returns

true if Click is in this shape, false if not

Implements [A\\_Shape](#).

## 5.7.3.8 aLeftCorner()

```
T\_Point< double > C_MTriangle::aLeftCorner ( ) [override], [virtual]
```

Take the point at left top corner.

## Returns

Return the point at left top corner

Implements [A\\_Shape](#).

## 5.7.3.9 aLeftFlip()

```
void C_MTriangle::aLeftFlip ( ) [override], [virtual]
```

Flip the figure as 45° anti clock.

Implements [A\\_Shape](#).

## 5.7.3.10 aMove()

```
void C_MTriangle::aMove (
    const T\_Point< double > & translation ) [override], [virtual]
```

Move the [C\\_MTriangle](#) by point translation.

## Parameters

<i>translation</i>	: Every mPoints of this shape will be translate by this __Parameter
--------------------	---

Implements [A\\_Shape](#).

**5.7.3.11 aReverse()**

```
void C_MTriangle::aReverse ( ) [override], [virtual]
```

Reverse the figure as symmetry.

Implements [A\\_Shape](#).

**5.7.3.12 aRightFlip()**

```
void C_MTriangle::aRightFlip ( ) [override], [virtual]
```

Flip the figure as 45° clock.

Implements [A\\_Shape](#).

**5.7.3.13 aRotate()**

```
void C_MTriangle::aRotate (
    double angular ) [override], [virtual]
```

Rotate the [C\\_MTriangle](#) with specified angular.

**Parameters**

<i>angular</i>	: This angular should be between (0, 2PI)
----------------	---

Implements [A\\_Shape](#).

**5.7.3.14 aSetPoints()**

```
bool C_MTriangle::aSetPoints (
    const T_Point< double > & ref,
    const T_Point< double > & changed ) [override], [virtual]
```

Set a point to another one.

**Parameters**

<i>ref</i>	: Point to change
<i>changed</i>	: New value of the point



**Returns**

True if the ref point exists, false otherwise

Implements [A\\_Shape](#).

**5.7.3.15 aToString()**

```
std::string C_MTriangle::aToString ( ) [override], [virtual]
```

Convert all data of [C\\_MTriangle](#) in a string.

**Returns**

Return a string which contains every mPoints of this shape

Implements [A\\_Shape](#).

**5.7.3.16 iDraw()** [1/2]

```
void C_MTriangle::iDraw ( ) [override], [virtual]
```

Draw this shape on IHM.

Implements [I\\_Drawable](#).

**5.7.3.17 iDraw()** [2/2]

```
void C_MTriangle::iDraw (
    MLV_Color color ) [override], [virtual]
```

Draw this shape on IHM with specific color.

**Parameters**

<i>color</i>	: Color of the shape will be draw
--------------	-----------------------------------

Implements [I\\_Drawable](#).

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

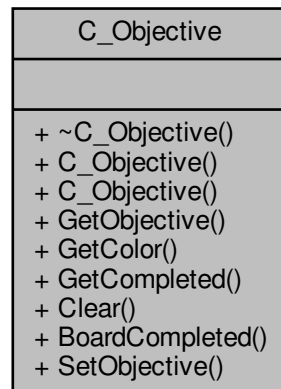
- include/shape/C\_MTriangle.hpp
- src/shape/C\_MTriangle.cpp

## 5.8 C\_Objective Class Reference

Class of the board [C\\_Objective](#).

```
#include <C_Objective.hpp>
```

Collaboration diagram for C\_Objective:



### Public Member Functions

- [~C\\_Objective](#) ()  
*Class methods.*
- [C\\_Objective](#) (MLV\_Color color=MLV\_COLOR\_GRAY70)  
*Constructor of an mObjective, default constructor.*
- [C\\_Objective](#) (const std::vector< std::shared\_ptr< [A\\_Shape](#) >> &objective, MLV\_Color color=MLV\_COLOR\_GRAY70)  
*Constructor of an mObjective.*
- std::vector< std::shared\_ptr< [A\\_Shape](#) >> [GetObjective](#) ()  
*Get all shape of the mObjective.*
- MLV\_Color [GetColor](#) ()  
*Get the mColor of an [C\\_Objective](#).*
- double [GetCompleted](#) (const std::vector< std::shared\_ptr< [A\\_Shape](#) >> &objective, const std::vector< std::shared\_ptr< [A\\_Shape](#) >> &game)  
*Give the progress of the puzzle.*
- void [Clear](#) ()  
*Clear the objective.*

### Static Public Member Functions

- static bool [BoardCompleted](#) (const std::vector< std::shared\_ptr< [A\\_Shape](#) >> &objective, const std::vector< std::shared\_ptr< [A\\_Shape](#) >> &game)  
*Check if the board is mCompleted.*
- static void [SetObjective](#) (std::shared\_ptr< [C\\_Objective](#) > objective, const std::vector< std::shared\_ptr< [A\\_Shape](#) >> &vec\_objective)  
*Set an [C\\_Objective](#) for a new game.*

### 5.8.1 Detailed Description

Class of the board [C\\_Objective](#).

This class manage everything about the mObjective

### 5.8.2 Constructor & Destructor Documentation

#### 5.8.2.1 ~C\_Objective()

```
C_Objective::~~C_Objective ( )
```

Class methods.

#### 5.8.2.2 C\_Objective() [1/2]

```
C_Objective::C_Objective (
    MLV_Color color = MLV_COLOR_GRAY70 ) [explicit]
```

Constructor of an mObjective, default constructor.

##### Parameters

<i>color</i>	: mColor of the mObjective shape
--------------	----------------------------------

#### 5.8.2.3 C\_Objective() [2/2]

```
C_Objective::C_Objective (
    const std::vector< std::shared_ptr< A\_Shape >> & objective,
    MLV_Color color = MLV_COLOR_GRAY70 ) [explicit]
```

Constructor of an mObjective.

##### Parameters

<i>objective</i>	: <a href="#">C_Objective</a> requires a vector of <a href="#">A_Shape</a>
<i>color</i>	: mColor of the mObjective shape

### 5.8.3 Member Function Documentation

#### 5.8.3.1 BoardCompleted()

```
bool C_Objective::BoardCompleted (
    const std::vector< std::shared_ptr< A_Shape >> & objective,
    const std::vector< std::shared_ptr< A_Shape >> & game ) [static]
```

Check if the board is mCompleted.

##### Parameters

<i>objective</i>	: Vector of mObjective's shape
<i>game</i>	: Vector of current game's shape

##### Returns

True if the board is mCompleted, false if not

#### 5.8.3.2 Clear()

```
void C_Objective::Clear ( )
```

Clear the objective.

#### 5.8.3.3 GetColor()

```
MLV_Color C_Objective::GetColor ( )
```

Get the mColor of an [C\\_Objective](#).

##### Returns

Return the mColor of an [C\\_Objective](#)

#### 5.8.3.4 GetCompleted()

```
double C_Objective::GetCompleted (
    const std::vector< std::shared_ptr< A_Shape >> & objective,
    const std::vector< std::shared_ptr< A_Shape >> & game )
```

Give the progress of the puzzle.

## Parameters

<i>objective</i>	: Shapes of objective
<i>game</i>	: Shape of the game

## Returns

Return the %100 of the progress

## 5.8.3.5 GetObjective()

```
std::vector< std::shared_ptr< A_Shape > > C_Objective::GetObjective ( )
```

Get all shape of the mObjective.

## Returns

Return a vector of shape of the mObjective

## 5.8.3.6 SetObjective()

```
void C_Objective::SetObjective (
    std::shared_ptr< C_Objective > objective,
    const std::vector< std::shared_ptr< A_Shape >> & vec_objective ) [static]
```

Set an [C\\_Objective](#) for a new game.

## Parameters

<i>objective</i>	: <a href="#">C_Objective</a> to mUpdate
<i>vec_objective</i>	:Vector of new <a href="#">A_Shape</a> for the new <a href="#">C_Objective</a>

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

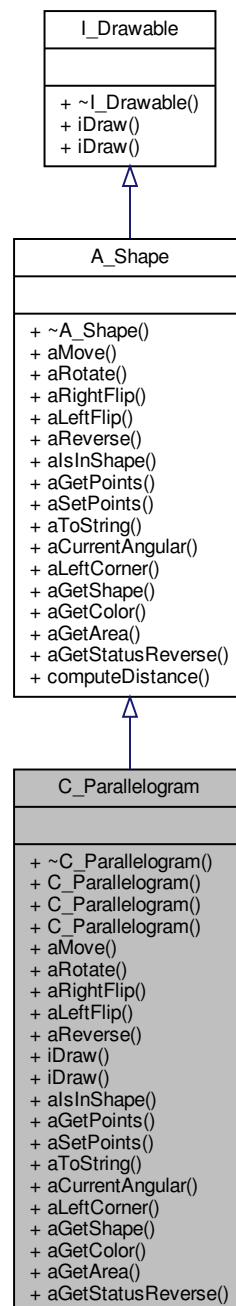
- [include/game/C\\_Objective.hpp](#)
- [src/game/C\\_Objective.cpp](#)

## 5.9 C\_Parallelogram Class Reference

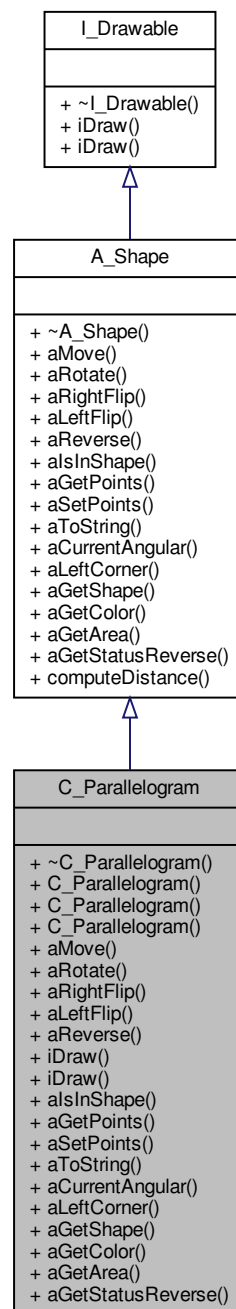
Class of the parallelogram.

```
#include <C_Parallelogram.hpp>
```

Inheritance diagram for C\_Parallelogram:



Collaboration diagram for C\_Parallelogram:



## Public Member Functions

- `~C_Parallelogram()` override  
Destructor of `C_Parallelogram`.
- `C_Parallelogram` (MLV\_Color color=MLV\_COLOR\_BLUE)  
Constructor by default of `C_Parallelogram`, make a `C_Parallelogram` as default.
- `C_Parallelogram` (const std::vector< `C_STriangle` > &triangle, MLV\_Color color=MLV\_COLOR\_BLUE)

- Constructor of *C\_Parallelogram*, requires a vector of *STriangles*.

  - *C\_Parallelogram* (const *T\_Point*< double > &origin, double angular=0.0, MLV\_Color color=MLV\_COLOR←\_BLUE, bool reverse=false)

Constructor of *C\_Parallelogram*, calls the delegate Default Constructor.
- void *aMove* (const *T\_Point*< double > &translation) override
 

Move the *C\_Parallelogram* by point translation.
- void *aRotate* (double angular) override
 

Rotate the *C\_Parallelogram* with specified angular.
- void *aRightFlip* () override
 

Flip the figure as 45 ° clock.
- void *aLeftFlip* () override
 

Flip the figure as 45 ° anti clock.
- void *aReverse* () override
 

Reverse the figure as symmetry.
- void *iDraw* () override
 

Draw this shape on IHM.
- void *iDraw* (MLV\_Color color) override
 

Draw this shape on IHM with specific color.
- bool *alsInShape* (const *T\_Point*< double > &click) override
 

Check if a point is in this shape.
- std::vector< *T\_Point*< double > > *aGetPoints* () override
 

Get mPoints of this shape.
- bool *aSetPoints* (const *T\_Point*< double > &ref, const *T\_Point*< double > &changed) override
 

Set a point to another one.
- std::string *aToString* () override
 

Convert all data of *C\_Parallelogram* in a string.
- double *aCurrentAngular* () override
 

Get the current angular of this shape.
- *T\_Point*< double > *aLeftCorner* () override
 

Take the point at left top corner.
- std::string *aGetShape* () override
 

Get the shape type.
- MLV\_Color *aGetColor* () override
 

Get the color of the shape.
- double *aGetArea* () override
 

Get the area of the shape.
- bool *aGetStatusReverse* () const override
 

Get the status of shape reversed or not.

## Additional Inherited Members

### 5.9.1 Detailed Description

Class of the parallelogram.

This class manage everything about the *C\_Parallelogram*

### 5.9.2 Constructor & Destructor Documentation



## 5.9.2.1 ~C\_Parallelogram()

```
C_Parallelogram::~C_Parallelogram ( ) [override]
```

Destructor of [C\\_Parallelogram](#).

## 5.9.2.2 C\_Parallelogram() [1/3]

```
C_Parallelogram::C_Parallelogram (
    MLV_Color color = MLV_COLOR_BLUE ) [explicit]
```

Constructor by default of [C\\_Parallelogram](#), make a [C\\_Parallelogram](#) as default.

## Parameters

<i>color</i>	: Optional __Parameter, mColor of this shape
--------------	--

## 5.9.2.3 C\_Parallelogram() [2/3]

```
C_Parallelogram::C_Parallelogram (
    const std::vector< C_STriangle > & triangle,
    MLV_Color color = MLV_COLOR_BLUE ) [explicit]
```

Constructor of [C\\_Parallelogram](#), requires a vector of STriangles.

## Parameters

<i>triangle</i>	: The <a href="#">C_Parallelogram</a> will created with a vector of <a href="#">C_STriangle</a> (4)
<i>color</i>	: Optional __Parameter, mColor of this shape

## 5.9.2.4 C\_Parallelogram() [3/3]

```
C_Parallelogram::C_Parallelogram (
    const T_Point< double > & origin,
    double angular = 0.0,
    MLV_Color color = MLV_COLOR_BLUE,
    bool reverse = false ) [explicit]
```

Constructor of [C\\_Parallelogram](#), calls the delegate Default Constructor.

## Parameters

<i>origin</i>	: shifts the figure of a translation of the origin
<i>angular</i>	: Optional __Parameter (angular=0.0 as default), aRotate the figure with an angular
<i>color</i>	: Optional __Parameter, mColor of this shape

### 5.9.3 Member Function Documentation

#### 5.9.3.1 aCurrentAngular()

```
double C_Parallelogram::aCurrentAngular ( ) [override], [virtual]
```

Get the current angular of this shape.

##### Returns

Implements [A\\_Shape](#).

#### 5.9.3.2 aGetArea()

```
double C_Parallelogram::aGetArea ( ) [override], [virtual]
```

Get the area of the shape.

##### Returns

Return the area of the shape

Implements [A\\_Shape](#).

#### 5.9.3.3 aGetColor()

```
MLV_Color C_Parallelogram::aGetColor ( ) [override], [virtual]
```

Get the color of the shape.

##### Returns

Return the MLV\_Color of the shape

Implements [A\\_Shape](#).

#### 5.9.3.4 aGetPoints()

```
std::vector< T_Point< double > > C_Parallelogram::aGetPoints ( ) [override], [virtual]
```

Get mPoints of this shape.

##### Returns

Return a vector of mPoints of this shape

Implements [A\\_Shape](#).

#### 5.9.3.5 aGetShape()

```
std::string C_Parallelogram::aGetShape ( ) [override], [virtual]
```

Get the shape type.

##### Returns

Return as string the shape type

Implements [A\\_Shape](#).

#### 5.9.3.6 aGetStatusReverse()

```
bool C_Parallelogram::aGetStatusReverse ( ) const [override], [virtual]
```

Get the status of shape reversed or not.

##### Returns

Return true if the shape got reversed, false otherwise

Implements [A\\_Shape](#).

#### 5.9.3.7 aIsInShape()

```
bool C_Parallelogram::aIsInShape (
    const T_Point< double > & click ) [override], [virtual]
```

Check if a point is in this shape.

**Parameters**

<i>click</i>	: <a href="#">T_Point</a> to check
--------------	------------------------------------

**Returns**

true if Click is in this shape, false if not

Implements [A\\_Shape](#).

**5.9.3.8 aLeftCorner()**

```
T\_Point< double > C_Parallelogram::aLeftCorner ( ) [override], [virtual]
```

Take the point at left top corner.

**Returns**

Return the point at left top corner

Implements [A\\_Shape](#).

**5.9.3.9 aLeftFlip()**

```
void C_Parallelogram::aLeftFlip ( ) [override], [virtual]
```

Flip the figure as 45° anti clock.

Implements [A\\_Shape](#).

**5.9.3.10 aMove()**

```
void C_Parallelogram::aMove (
    const T\_Point< double > & translation ) [override], [virtual]
```

Move the [C\\_Parallelogram](#) by point translation.

**Parameters**

<i>translation</i>	: Every mPoints of this shape will be translate by this __Parameter
--------------------	---

Implements [A\\_Shape](#).

#### 5.9.3.11 aReverse()

```
void C_Parallelogram::aReverse ( ) [override], [virtual]
```

Reverse the figure as symmetry.

Implements [A\\_Shape](#).

#### 5.9.3.12 aRightFlip()

```
void C_Parallelogram::aRightFlip ( ) [override], [virtual]
```

Flip the figure as 45° clock.

Implements [A\\_Shape](#).

#### 5.9.3.13 aRotate()

```
void C_Parallelogram::aRotate (
    double angular ) [override], [virtual]
```

Rotate the [C\\_Parallelogram](#) with specified angular.

Parameters

<i>angular</i>	: This angular should be between (0, 2PI)
----------------	---

Implements [A\\_Shape](#).

#### 5.9.3.14 aSetPoints()

```
bool C_Parallelogram::aSetPoints (
    const T\_Point< double > & ref,
    const T\_Point< double > & changed ) [override], [virtual]
```

Set a point to another one.

Parameters

<i>ref</i>	: Point to change
<i>changed</i>	: New value of the point

**Returns**

True if the ref point exists, false otherwise

Implements [A\\_Shape](#).

**5.9.3.15 aToString()**

```
std::string C_Parallelogram::aToString ( ) [override], [virtual]
```

Convert all data of [C\\_Parallelogram](#) in a string.

**Returns**

Return a string which contains every mPoints of this shape

Implements [A\\_Shape](#).

**5.9.3.16 iDraw()** [1/2]

```
void C_Parallelogram::iDraw ( ) [override], [virtual]
```

Draw this shape on IHM.

Implements [I\\_Drawable](#).

**5.9.3.17 iDraw()** [2/2]

```
void C_Parallelogram::iDraw (
    MLV_Color color ) [override], [virtual]
```

Draw this shape on IHM with specific color.

**Parameters**

<i>color</i>	: Color of the shape will be draw
--------------	-----------------------------------

Implements [I\\_Drawable](#).

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

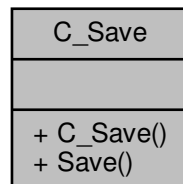
- include/shape/[C\\_Parallelogram.hpp](#)
- src/shape/[C\\_Parallelogram.cpp](#)

## 5.10 C\_Save Class Reference

Class of the main Saver.

```
#include <C_Save.hpp>
```

Collaboration diagram for C\_Save:



### Public Member Functions

- [C\\_Save](#) ()
- bool [Save](#) (const std::vector< std::shared\_ptr< [A\\_Shape](#) >> &Game)  
*Save the current board as puzzle file in a page which contains less than 12 files.*

#### 5.10.1 Detailed Description

Class of the main Saver.

This class manage everything about the save

#### 5.10.2 Constructor & Destructor Documentation

##### 5.10.2.1 C\_Save()

```
C_Save::C_Save ( )
```

Construct an instance of a saver

#### 5.10.3 Member Function Documentation

##### 5.10.3.1 Save()

```
bool C_Save::Save (
    const std::vector< std::shared_ptr< A\_Shape >> & Game )
```

Save the current board as puzzle file in a page which contains less than 12 files.

**Parameters**

<i>Game</i>	: Current game
-------------	----------------

**Returns**

Return true if the board has been saved, false otherwise

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

- [include/parser/C\\_Save.hpp](#)
- [src/parser/C\\_Save.cpp](#)

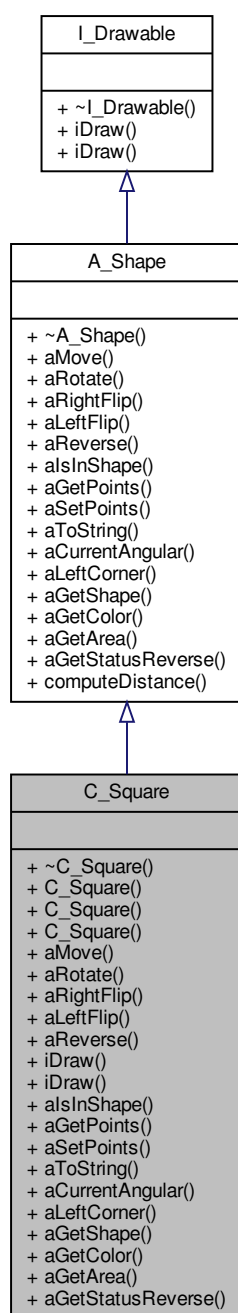
## 5.11 C\_Square Class Reference

Class of the square.

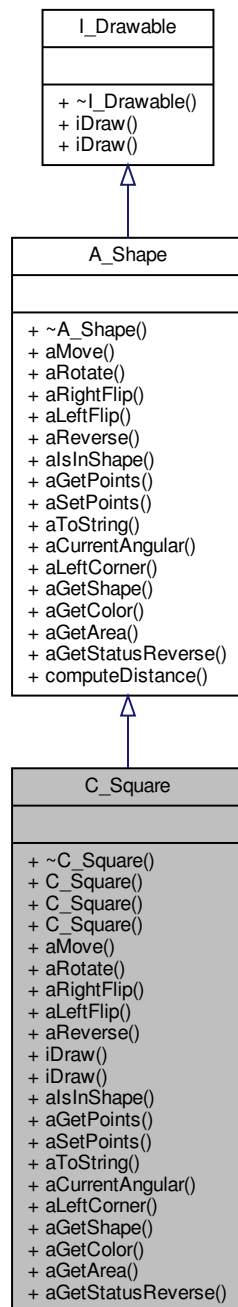
```
#include <C_Square.hpp>
```



Inheritance diagram for C\_Square:



Collaboration diagram for C\_Square:



## Public Member Functions

- `~C_Square ()` override  
*Destructor of `C_Square`.*
- `C_Square (MLV_Color color=MLV_COLOR_PURPLE)`  
*Constructor by default of `C_Square`, make a `C_Square` as default.*
- `C_Square (const std::vector< C_STriangle > &triangle, MLV_Color color=MLV_COLOR_PURPLE)`

Constructor of [C\\_Square](#), requires a vector of [STriangles](#).

- [C\\_Square](#) (const [T\\_Point](#)< double > &origin, double angular=0.0, MLV\_Color color=MLV\_COLOR\_Purple) (PLE)

Constructor of [C\\_Square](#), calls the delegate Default Constructor.

- void [aMove](#) (const [T\\_Point](#)< double > &translation) override

Move the [C\\_Square](#) by point translation.

- void [aRotate](#) (double angular) override

Rotate the [C\\_Square](#) with specified angular.

- void [aRightFlip](#) () override

Flip the figure as 45° clock.

- void [aLeftFlip](#) () override

Flip the figure as 45° anti clock.

- void [aReverse](#) () override

Reverse the figure as symmetry.

- void [iDraw](#) () override

Draw this shape on IHM.

- void [iDraw](#) (MLV\_Color color) override

Draw this shape on IHM with specific color.

- bool [aIsInShape](#) (const [T\\_Point](#)< double > &click) override

Check if a point is in this shape.

- std::vector< [T\\_Point](#)< double > > [aGetPoints](#) () override

Get mPoints of this shape.

- bool [aSetPoints](#) (const [T\\_Point](#)< double > &ref, const [T\\_Point](#)< double > &changed) override

Set a point to another one.

- std::string [aToString](#) () override

Convert all data of [C\\_Square](#) in a string.

- double [aCurrentAngular](#) () override

Get the current angular of this shape.

- [T\\_Point](#)< double > [aLeftCorner](#) () override

Take the point at left top corner.

- std::string [aGetShape](#) () override

Get the shape type.

- MLV\_Color [aGetColor](#) () override

Get the color of the shape.

- double [aGetArea](#) () override

Get the area of the shape.

- bool [aGetStatusReverse](#) () const override

Get the status of shape reversed or not.

## Additional Inherited Members

### 5.11.1 Detailed Description

Class of the square.

This class manage everything about the [C\\_Square](#)

### 5.11.2 Constructor & Destructor Documentation

5.11.2.1 `~C_Square()`

```
C_Square::~C_Square ( ) [override]
```

Destructor of [C\\_Square](#).

5.11.2.2 `C_Square()` [1/3]

```
C_Square::C_Square (
    MLV_Color color = MLV_COLOR_PURPLE ) [explicit]
```

Constructor by default of [C\\_Square](#), make a [C\\_Square](#) as default.

## Parameters

<i>color</i>	: Optional __Parameter, mColor of this shape
--------------	--

5.11.2.3 `C_Square()` [2/3]

```
C_Square::C_Square (
    const std::vector< C_STriangle > & triangle,
    MLV_Color color = MLV_COLOR_PURPLE ) [explicit]
```

Constructor of [C\\_Square](#), requires a vector of STriangles.

## Parameters

<i>triangle</i>	: The <a href="#">C_Square</a> will created with a vector of <a href="#">C_STriangle</a> (4)
<i>color</i>	: Optional __Parameter, mColor of this shape

5.11.2.4 `C_Square()` [3/3]

```
C_Square::C_Square (
    const T_Point< double > & origin,
    double angular = 0.0,
    MLV_Color color = MLV_COLOR_PURPLE ) [explicit]
```

Constructor of [C\\_Square](#), calls the delegate Default Constructor.

## Parameters

<i>origin</i>	: shifts the figure of a translation of the origin
<i>angular</i>	: Optional __Parameter (angular=0.0 as default), aRotate the figure with an angular
<i>color</i>	: Optional __Parameter, mColor of this shape

### 5.11.3 Member Function Documentation

#### 5.11.3.1 aCurrentAngular()

```
double C_Square::aCurrentAngular ( ) [override], [virtual]
```

Get the current angular of this shape.

**Returns**

Implements [A\\_Shape](#).

#### 5.11.3.2 aGetArea()

```
double C_Square::aGetArea ( ) [override], [virtual]
```

Get the area of the shape.

**Returns**

Return the area of the shape

Implements [A\\_Shape](#).

#### 5.11.3.3 aGetColor()

```
MLV_Color C_Square::aGetColor ( ) [override], [virtual]
```

Get the color of the shape.

**Returns**

Return the MLV\_Color of the shape

Implements [A\\_Shape](#).

#### 5.11.3.4 aGetPoints()

```
std::vector< T_Point< double > > C_Square::aGetPoints ( ) [override], [virtual]
```

Get mPoints of this shape.

##### Returns

Return a vector of mPoints of this shape

Implements [A\\_Shape](#).

#### 5.11.3.5 aGetShape()

```
std::string C_Square::aGetShape ( ) [override], [virtual]
```

Get the shape type.

##### Returns

Return as string the shape type

Implements [A\\_Shape](#).

#### 5.11.3.6 aGetStatusReverse()

```
bool C_Square::aGetStatusReverse ( ) const [override], [virtual]
```

Get the status of shape reversed or not.

##### Returns

Return true if the shape got reversed, false otherwise

Implements [A\\_Shape](#).

#### 5.11.3.7 aIsInShape()

```
bool C_Square::aIsInShape (
    const T_Point< double > & click ) [override], [virtual]
```

Check if a point is in this shape.

## Parameters

<i>click</i>	: <a href="#">T_Point</a> to check
--------------	------------------------------------

## Returns

true if Click is in this shape, false if not

Implements [A\\_Shape](#).

## 5.11.3.8 aLeftCorner()

```
T\_Point< double > C_Square::aLeftCorner ( ) [override], [virtual]
```

Take the point at left top corner.

## Returns

Return the point at left top corner

Implements [A\\_Shape](#).

## 5.11.3.9 aLeftFlip()

```
void C_Square::aLeftFlip ( ) [override], [virtual]
```

Flip the figure as 45° anti clock.

Implements [A\\_Shape](#).

## 5.11.3.10 aMove()

```
void C_Square::aMove (
    const T\_Point< double > & translation ) [override], [virtual]
```

Move the [C\\_Square](#) by point translation.

## Parameters

<i>translation</i>	: Every mPoints of this shape will be translate by this __Parameter
--------------------	---

Implements [A\\_Shape](#).

#### 5.11.3.11 aReverse()

```
void C_Square::aReverse ( ) [override], [virtual]
```

Reverse the figure as symmetry.

Implements [A\\_Shape](#).

#### 5.11.3.12 aRightFlip()

```
void C_Square::aRightFlip ( ) [override], [virtual]
```

Flip the figure as 45° clock.

Implements [A\\_Shape](#).

#### 5.11.3.13 aRotate()

```
void C_Square::aRotate (
    double angular ) [override], [virtual]
```

Rotate the [C\\_Square](#) with specified angular.

##### Parameters

<i>angular</i>	: This angular should be between (0, 2PI)
----------------	---

Implements [A\\_Shape](#).

#### 5.11.3.14 aSetPoints()

```
bool C_Square::aSetPoints (
    const T\_Point< double > & ref,
    const T\_Point< double > & changed ) [override], [virtual]
```

Set a point to another one.

##### Parameters

<i>ref</i>	: Point to change
<i>changed</i>	: New value of the point



**Returns**

True if the ref point exists, false otherwise

Implements [A\\_Shape](#).

**5.11.3.15 aToString()**

```
std::string C_Square::aToString ( ) [override], [virtual]
```

Convert all data of [C\\_Square](#) in a string.

**Returns**

Return a string which contains every mPoints of this shape

Implements [A\\_Shape](#).

**5.11.3.16 iDraw() [1/2]**

```
void C_Square::iDraw ( ) [override], [virtual]
```

Draw this shape on IHM.

Implements [I\\_Drawable](#).

**5.11.3.17 iDraw() [2/2]**

```
void C_Square::iDraw (
    MLV_Color color ) [override], [virtual]
```

Draw this shape on IHM with specific color.

**Parameters**

<i>color</i>	: color of the shape will be draw
--------------	-----------------------------------

Implements [I\\_Drawable](#).

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

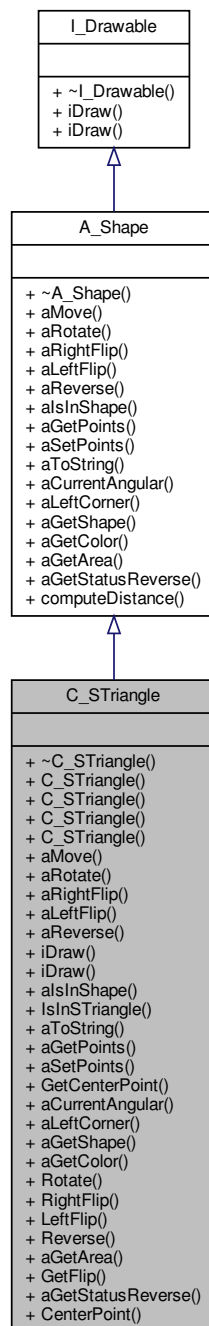
- include/shape/C\_Square.hpp
- src/shape/C\_Square.cpp

## 5.12 C\_STriangle Class Reference

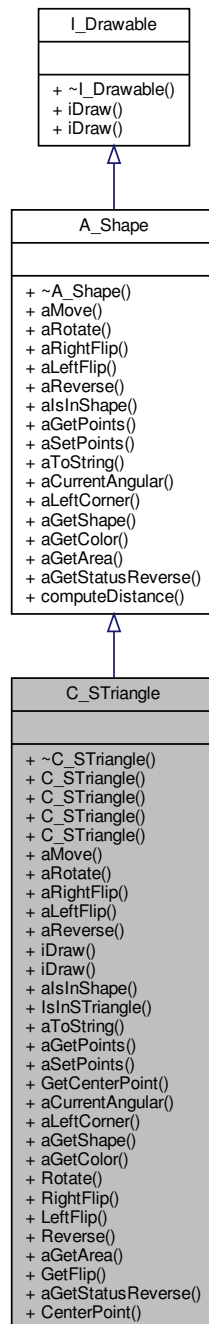
Class of the small [C\\_STriangle](#).

```
#include <C_STriangle.hpp>
```

Inheritance diagram for C\_STriangle:



Collaboration diagram for C\_STriangle:



## Public Member Functions

- [~C\\_STriangle](#) () override  
*Destructor of [C\\_STriangle](#).*
- [C\\_STriangle](#) (MLV\_Color color=MLV\_COLOR\_GREEN)  
*Constructor by default of [C\\_MTriangle](#), make a [C\\_STriangle](#) as default.*

- `C_STriangle` (const `T_Point`< double > &p1, const `T_Point`< double > &p2, const `T_Point`< double > &p3, `MLV_Color` color=`MLV_COLOR_GREEN`)  
*Constructor of `C_STriangle`, requires 3 mPoints.*
- `C_STriangle` (const std::vector< `T_Point`< double > > &points, `MLV_Color` color=`MLV_COLOR_GREEN`)  
*Constructor of `C_STriangle`, requires a vector of 3 mPoints.*
- `C_STriangle` (const `T_Point`< double > &origin, double angular=0.0, `MLV_Color` color=`MLV_COLOR_GREEN`)  
*Constructor of `C_STriangle`, calls the delegate Default Constructor.*
- void `aMove` (const `T_Point`< double > &translation) override  
*Move the `C_MTriangle` by point translation.*
- void `aRotate` (double angular) override  
*Rotate the `C_STriangle` with specified angular.*
- void `aRightFlip` () override  
*Flip the figure as 45° clock.*
- void `aLeftFlip` () override  
*Flip the figure as 45° anti clock.*
- void `aReverse` () override  
*Reverse the figure as symmetry.*
- void `iDraw` () override  
*Draw this shape on IHM.*
- void `iDraw` (`MLV_Color` color) override  
*Draw this shape on IHM with specific mColor.*
- bool `alsInShape` (const `T_Point`< double > &click) override  
*Check if a point is in this shape.*
- bool `IsInSTriangle` (const `T_Point`< double > &click)  
*Check if a point is in this `C_STriangle`.*
- std::string `aToString` () override  
*Convert all data of `C_MTriangle` in a string.*
- std::vector< `T_Point`< double > > `aGetPoints` () override  
*Get every mPoints of this `C_STriangle`.*
- bool `aSetPoints` (const `T_Point`< double > &ref, const `T_Point`< double > &changed) override  
*Set a point as same value that another point given in parameter.*
- `T_Point`< double > `GetCenterPoint` () const  
*Get the current center point of this `C_STriangle`.*
- double `aCurrentAngular` () override  
*Get the current angular of this shape.*
- `T_Point`< double > `aLeftCorner` () override  
*Take the point at left top corner.*
- std::string `aGetShape` () override  
*Get the type of shape is it.*
- `MLV_Color` `aGetColor` () override  
*Get the color of the shape.*
- void `Rotate` (double angular, const `T_Point`< double > &center\_point)  
*Rotate an `C_STriangle` with specified angular, used only for an other shape.*
- void `RightFlip` (const `T_Point`< double > &centerPoint)  
*Right flip as 45° clock.*
- void `LeftFlip` (const `T_Point`< double > &centerPoint)  
*Right flip as 45° anti clock.*
- void `Reverse` (const `T_Point`< double > &centerPoint)  
*Reverse the figure as symmetry.*
- double `aGetArea` () override

*Get the area of the shape.*

- `std::vector< T_Point< double > > GetFlip ()`

*Get a vector of "flip" needed to flip the figure.*

- `bool aGetStatusReverse ()` const override

*Get the status of shape reversed or not.*

## Static Public Member Functions

- `static T_Point< double > CenterPoint (const std::vector< T_Point< double > > &list_points)`

*Compute the center point of N mPoints.*

## 5.12.1 Detailed Description

Class of the small [C\\_STriangle](#).

This class manage everything about the small [C\\_STriangle](#)

## 5.12.2 Constructor & Destructor Documentation

### 5.12.2.1 ~C\_STriangle()

```
C_STriangle::~C_STriangle ( ) [override]
```

Destructor of [C\\_STriangle](#).

### 5.12.2.2 C\_STriangle() [1/4]

```
C_STriangle::C_STriangle (
    MLV_Color color = MLV_COLOR_GREEN ) [explicit]
```

Constructor by default of [C\\_MTriangle](#), make a [C\\_STriangle](#) as default.

#### Parameters

<i>color</i>	: Optional __Parameter, mColor of this shape
--------------	--

### 5.12.2.3 C\_STriangle() [2/4]

```
C_STriangle::C_STriangle (
    const T_Point< double > & p1,
```

```

const T_Point< double > & p2,
const T_Point< double > & p3,
MLV_Color color = MLV_COLOR_GREEN )

```

Constructor of [C\\_STriangle](#), requires 3 mPoints.

#### Parameters

<i>p1</i>	: First point of the <a href="#">C_STriangle</a>
<i>p2</i>	: Second point of the <a href="#">C_STriangle</a>
<i>p3</i>	: Third point of the <a href="#">C_STriangle</a>
<i>color</i>	: Optional __Parameter, mColor of this shape

#### 5.12.2.4 C\_STriangle() [3/4]

```

C_STriangle::C_STriangle (
    const std::vector< T_Point< double >> & points,
    MLV_Color color = MLV_COLOR_GREEN ) [explicit]

```

Constructor of [C\\_STriangle](#), requires a vector of 3 mPoints.

#### Parameters

<i>points</i>	: vector of 3 mPoints
<i>color</i>	: Optional __Parameter, mColor of this shape

#### 5.12.2.5 C\_STriangle() [4/4]

```

C_STriangle::C_STriangle (
    const T_Point< double > & origin,
    double angular = 0.0,
    MLV_Color color = MLV_COLOR_GREEN ) [explicit]

```

Constructor of [C\\_STriangle](#), calls the delegate Default Constructor.

#### Parameters

<i>origin</i>	: shifts the figure of a translation of the origin
<i>angular</i>	: Optional __Parameter (angular=0.0 as default), aRotate the figure with an angular
<i>color</i>	: Optional __Parameter, mColor of this shape

### 5.12.3 Member Function Documentation

#### 5.12.3.1 aCurrentAngular()

```
double C_STriangle::aCurrentAngular ( ) [override], [virtual]
```

Get the current angular of this shape.

##### Returns

Return the current angular in double

Implements [A\\_Shape](#).

#### 5.12.3.2 aGetArea()

```
double C_STriangle::aGetArea ( ) [override], [virtual]
```

Get the area of the shape.

##### Returns

Return the area of this shape as a double

Implements [A\\_Shape](#).

#### 5.12.3.3 aGetColor()

```
MLV_Color C_STriangle::aGetColor ( ) [override], [virtual]
```

Get the color of the shape.

##### Returns

Return the MLV\_Color of the shape

Implements [A\\_Shape](#).

#### 5.12.3.4 aGetPoints()

```
std::vector< T_Point< double > > C_STriangle::aGetPoints ( ) [override], [virtual]
```

Get every mPoints of this [C\\_STriangle](#).

##### Returns

Return a vector of these mPoints

Implements [A\\_Shape](#).

#### 5.12.3.5 aGetShape()

```
std::string C_STriangle::aGetShape ( ) [override], [virtual]
```

Get the type of shape is it.

##### Returns

Return as string the type of shape is it

Implements [A\\_Shape](#).

#### 5.12.3.6 aGetStatusReverse()

```
bool C_STriangle::aGetStatusReverse ( ) const [override], [virtual]
```

Get the status of shape reversed or not.

##### Returns

Return true if the shape got reversed, false otherwise

Implements [A\\_Shape](#).

#### 5.12.3.7 aIsInShape()

```
bool C_STriangle::aIsInShape (
    const T_Point< double > & click ) [override], [virtual]
```

Check if a point is in this shape.

##### Parameters

<i>click</i>	: <a href="#">T_Point</a> to check
--------------	------------------------------------

##### Returns

true if Click is in this shape, false if not

Implements [A\\_Shape](#).



#### 5.12.3.8 aLeftCorner()

```
T_Point< double > C_STriangle::aLeftCorner ( ) [override], [virtual]
```

Take the point at left top corner.

##### Returns

Return the point at left top corner

Implements [A\\_Shape](#).

#### 5.12.3.9 aLeftFlip()

```
void C_STriangle::aLeftFlip ( ) [override], [virtual]
```

Flip the figure as 45° anti clock.

Implements [A\\_Shape](#).

#### 5.12.3.10 aMove()

```
void C_STriangle::aMove (
    const T_Point< double > & translation ) [override], [virtual]
```

Move the [C\\_MTriangle](#) by point translation.

##### Parameters

<i>translation</i>	: Every mPoints of this shape will be translate by this __Parameter
--------------------	---

Implements [A\\_Shape](#).

#### 5.12.3.11 aReverse()

```
void C_STriangle::aReverse ( ) [override], [virtual]
```

Reverse the figure as symmetry.

Implements [A\\_Shape](#).

#### 5.12.3.12 aRightFlip()

```
void C_STriangle::aRightFlip ( ) [override], [virtual]
```

Flip the figure as 45° clock.

Implements [A\\_Shape](#).

#### 5.12.3.13 aRotate()

```
void C_STriangle::aRotate (
    double angular ) [override], [virtual]
```

Rotate the [C\\_STriangle](#) with specified angular.

##### Parameters

<i>angular</i>	: This angular should be between (0, 2PI)
----------------	---

Implements [A\\_Shape](#).

#### 5.12.3.14 aSetPoints()

```
bool C_STriangle::aSetPoints (
    const T\_Point< double > & ref,
    const T\_Point< double > & changed ) [override], [virtual]
```

Set a point as same value that another point given in parameter.

##### Parameters

<i>ref</i>	Point we want to set
<i>changed</i>	The ref Point will take same value as this one

##### Returns

Return true if the ref point exists and has benn changed, false otherwise

Implements [A\\_Shape](#).

#### 5.12.3.15 aToString()

```
std::string C_STriangle::aToString ( ) [override], [virtual]
```

Convert all data of [C\\_MTriangle](#) in a string.

**Returns**

Return a string which contains every mPoints of this shape

Implements [A\\_Shape](#).

**5.12.3.16 CenterPoint()**

```
T_Point< double > C_STriangle::CenterPoint (
    const std::vector< T_Point< double >> & list_points ) [static]
```

Compute the center point of N mPoints.

**Parameters**

<i>list_points</i>	: vector of N mPoints
--------------------	-----------------------

**Returns**

Return the center point of these N mPoints

**5.12.3.17 GetCenterPoint()**

```
T_Point< double > C_STriangle::GetCenterPoint ( ) const
```

Get the current center point of this [C\\_STriangle](#).

**Returns**

Return the current center point of this [C\\_STriangle](#)

**5.12.3.18 GetFlip()**

```
std::vector< T_Point< double > > C_STriangle::GetFlip ( )
```

Get a vector of "flip" needed to flip the figure.

**Returns**

Return a vector of point needed to flip the figure

**5.12.3.19 iDraw()** [1/2]

```
void C_STriangle::iDraw ( ) [override], [virtual]
```

Draw this shape on IHM.

Implements [I\\_Drawable](#).

**5.12.3.20 iDraw()** [2/2]

```
void C_STriangle::iDraw (
    MLV_Color color ) [override], [virtual]
```

Draw this shape on IHM with specific mColor.

**Parameters**

<i>Color</i>	: Color from the graphic library MLV like MLV_COLOR_XXX
--------------	---

Implements [I\\_Drawable](#).

**5.12.3.21 IsInSTriangle()**

```
bool C_STriangle::IsInSTriangle (
    const T_Point< double > & click )
```

Check if a point is in this [C\\_STriangle](#).

**Parameters**

<i>click</i>	: <a href="#">T_Point</a> to check
--------------	------------------------------------

**Returns**

true if Click is in this shape, false if not

**5.12.3.22 LeftFlip()**

```
void C_STriangle::LeftFlip (
    const T_Point< double > & centerPoint )
```

Right flip as 45° anti clock.

## Parameters

<i>centerPoint</i>	: flip the figure about this center point
--------------------	---

## 5.12.3.23 Reverse()

```
void C_STriangle::Reverse (
    const T_Point< double > & centerPoint )
```

Reverse the figure as symmetry.

## Parameters

<i>centerPoint</i>	: Reverse the figure as symmetry about this center point
--------------------	--

## 5.12.3.24 RightFlip()

```
void C_STriangle::RightFlip (
    const T_Point< double > & centerPoint )
```

Right flip as 45° clock.

## Parameters

<i>centerPoint</i>	: flip the figure about this center point
--------------------	---

## 5.12.3.25 Rotate()

```
void C_STriangle::Rotate (
    double angular,
    const T_Point< double > & center_point )
```

Rotate an [C\\_STriangle](#) with specified angular, used only for an other shape.

## Parameters

<i>angular</i>	: This angular should be between (0, 2PI)
<i>center_point</i>	: Rotate an <a href="#">C_STriangle</a> around this point

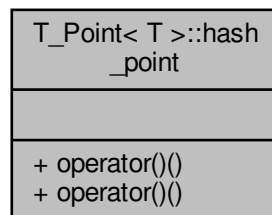
The documentation for this class was generated from the following files:

- [include/shape/C\\_STriangle.hpp](#)
- [src/shape/C\\_STriangle.cpp](#)

### 5.13 T\_Point< T >::hash\_point Struct Reference

```
#include <T_Point.hpp>
```

Collaboration diagram for T\_Point< T >::hash\_point:



#### Public Member Functions

- `std::size_t operator()` (const [T\\_Point< T >](#) &p) const  
*Operator to hash a point.*
- `bool operator()` (const [T\\_Point< T >](#) &p1, const [T\\_Point< T >](#) &p2) const  
*Operator equal need to hash a point.*

#### 5.13.1 Member Function Documentation

##### 5.13.1.1 operator() [1/2]

```
template<typename T>
std::size_t T\_Point< T >::hash\_point::operator\(\) (
    const T\_Point< T > & p ) const [inline]
```

Operator to hash a point.

#### Parameters

<i>p</i>	: point to hash
----------	-----------------

**Returns**

Return the hash of the point

**5.13.1.2 operator>() [2/2]**

```
template<typename T>
bool T_Point< T >::hash_point::operator() (
    const T_Point< T > & p1,
    const T_Point< T > & p2 ) const [inline]
```

Operator equal need to hash a point.

**Parameters**

<i>p1</i>	: Point 1
<i>p2</i>	: Point 2

**Returns**

Return true if p1 and p2 are equals, false otherwise

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

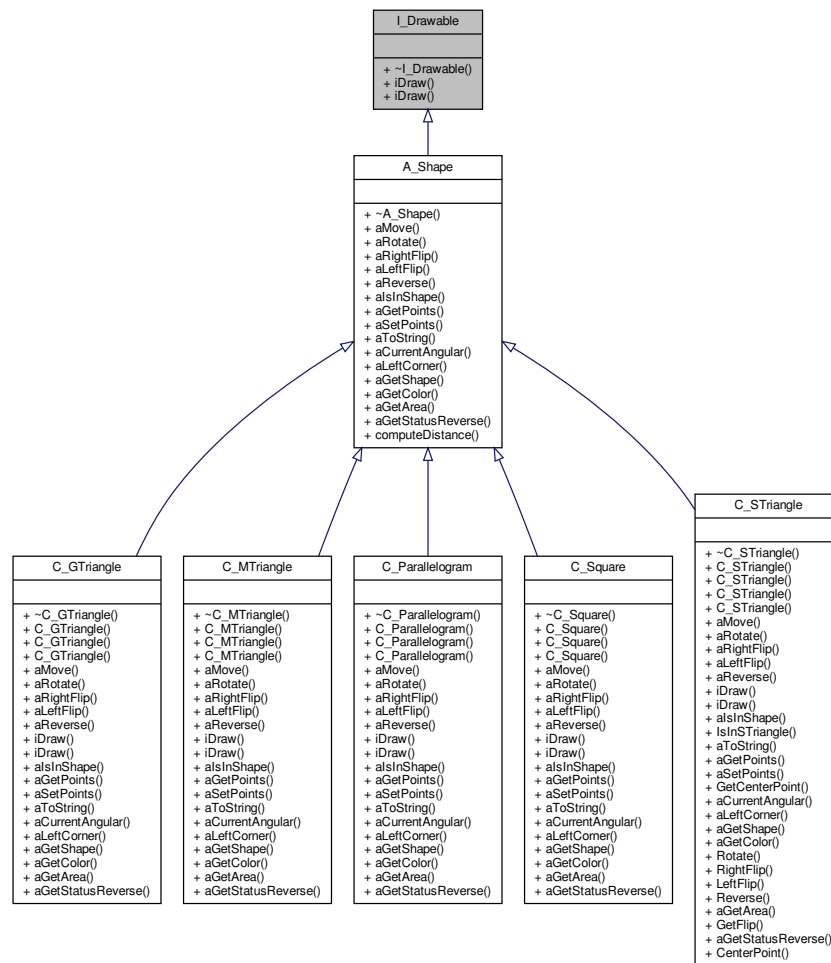
- [include/utlis/T\\_Point.hpp](#)

**5.14 I\_Drawable Class Reference**

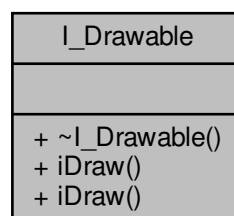
[I\\_Drawable](#) is everything to iDraw.

```
#include <I_Drawable.h>
```

Inheritance diagram for I\_Drawable:



Collaboration diagram for I\_Drawable:



## Public Member Functions

- `~I_Drawable()` = default



*Pure virtual function. Draw everything which needs to be iDraw.*

- virtual void [iDraw](#) ()=0
- virtual void [iDraw](#) (MLV\_Color color)=0

### 5.14.1 Detailed Description

[I\\_Drawable](#) is everything to iDraw.

This class manage everything drawing

### 5.14.2 Constructor & Destructor Documentation

#### 5.14.2.1 ~I\_Drawable()

```
I_Drawable::~~I_Drawable ( ) [default]
```

Pure virtual function. Draw everything which needs to be iDraw.

### 5.14.3 Member Function Documentation

#### 5.14.3.1 iDraw() [1/2]

```
virtual void I_Drawable::iDraw ( ) [pure virtual]
```

Implemented in [C\\_STriangle](#), [C\\_Parallelogram](#), [C\\_MTriangle](#), [C\\_Square](#), and [C\\_GTriangle](#).

#### 5.14.3.2 iDraw() [2/2]

```
virtual void I_Drawable::iDraw (
    MLV_Color color ) [pure virtual]
```

Implemented in [C\\_STriangle](#), [C\\_Parallelogram](#), [C\\_MTriangle](#), [C\\_Square](#), and [C\\_GTriangle](#).

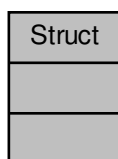
The documentation for this class was generated from the following file:

- [include/drawable/I\\_Drawable.h](#)

## 5.15 Struct Struct Reference

Hash a `T_Point<T>` to hash a point with `T_Point<T>`

Collaboration diagram for Struct:



### 5.15.1 Detailed Description

Hash a `T_Point<T>` to hash a point with `T_Point<T>`

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

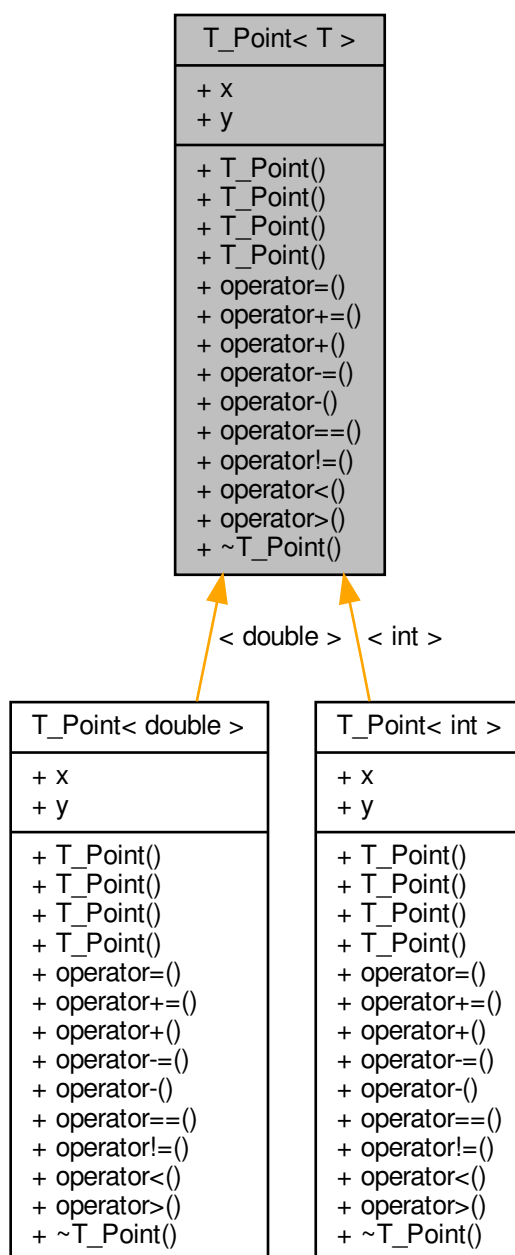
- [include/utlis/T\\_Point.hpp](#)

## 5.16 `T_Point< T >` Class Template Reference

Class of a [T\\_Point](#).

```
#include <T_Point.hpp>
```

Inheritance diagram for T\_Point< T >:



Collaboration diagram for `T_Point< T >`:

<code>T_Point&lt; T &gt;</code>
<code>+ x</code> <code>+ y</code>
<code>+ T_Point()</code> <code>+ T_Point()</code> <code>+ T_Point()</code> <code>+ T_Point()</code> <code>+ operator=()</code> <code>+ operator+=()</code> <code>+ operator+()</code> <code>+ operator-=()</code> <code>+ operator-()</code> <code>+ operator==()</code> <code>+ operator!=()</code> <code>+ operator&lt;()</code> <code>+ operator&gt;()</code> <code>+ ~T_Point()</code>

## Classes

- struct [hash\\_point](#)

## Public Member Functions

- constexpr [T\\_Point](#) (const [T\\_Point< T >](#) &p)=default
- [T\\_Point](#) ()  
*Constructor for a point with initialisation list.*
- [T\\_Point](#) (const [T\\_Point< T >](#) &&p) noexcept  
*Constructor of a point with move semantic.*
- [T\\_Point](#) (const T &\_x, const T &\_y)  
*Constructor for a point. Requires a X and a Y coordinate.*
- [T\\_Point](#) & [operator=](#) (const [T\\_Point< T >](#) &p)  
*Operator = of a point.*
- [T\\_Point](#) & [operator+=](#) (const [T\\_Point< T >](#) &p)  
*Operator +=.*
- [T\\_Point](#) [operator+](#) (const [T\\_Point< T >](#) &p)  
*Operator +.*
- [T\\_Point](#) & [operator-=](#) (const [T\\_Point< T >](#) &p)  
*Operator -=.*
- [T\\_Point](#) [operator-](#) (const [T\\_Point< T >](#) &p)  
*Operator -.*
- bool [operator==](#) (const [T\\_Point< T >](#) &p) const

- Operator == of a point.*
- bool `operator!=` (const `T_Point< T >` &p) const
- Operator != of a point.*
- bool `operator<` (const `T_Point< T >` &p) const
- Operator < of a point.*
- bool `operator>` (const `T_Point< T >` &p) const
- Operator > of a point.*
- `~T_Point` ()=default

## Public Attributes

- `T x`
- `T y`

### 5.16.1 Detailed Description

```
template<typename T>
class T_Point< T >
```

Class of a `T_Point`.

#### Template Parameters

<code>T</code>	: Template parameter This class manage everything about a point
----------------	---

### 5.16.2 Constructor & Destructor Documentation

#### 5.16.2.1 T\_Point() [1/4]

```
template<typename T>
constexpr T_Point< T >::T_Point (
    const T_Point< T > & p ) [default]
```

#### 5.16.2.2 T\_Point() [2/4]

```
template<typename T>
T_Point< T >::T_Point ( ) [inline]
```

Constructor for a point with initialisation list.

### 5.16.2.3 T\_Point() [3/4]

```
template<typename T>
T_Point< T >::T_Point (
    const T_Point< T > && p ) [inline], [noexcept]
```

Constructor of a point with move semantic.

#### Parameters

<i>p</i>	: Point to move
----------	-----------------

### 5.16.2.4 T\_Point() [4/4]

```
template<typename T>
T_Point< T >::T_Point (
    const T & _x,
    const T & _y ) [inline]
```

Constructor for a point. Requires a X and a Y coordinate.

#### Parameters

$\leftrightarrow$	: Template X coordinate
$\_ \leftrightarrow$	
<i>x</i>	
$\leftrightarrow$	: Template Y coordinate
$\_ \leftrightarrow$	
<i>y</i>	

### 5.16.2.5 ~T\_Point()

```
template<typename T>
T_Point< T >::~~T_Point ( ) [default]
```

## 5.16.3 Member Function Documentation

### 5.16.3.1 operator!=(())

```
template<typename T>
bool T_Point< T >::operator!= (
    const T_Point< T > & p ) const [inline]
```

Operator != of a point.

**Parameters**

$p$	: T_Point to compare
-----	----------------------

**Returns**

Return True if the point is different, false if not

**5.16.3.2 operator+()**

```
template<typename T>
T_Point T_Point< T >::operator+ (
    const T_Point< T > & p ) [inline]
```

Operator +.

**Parameters**

$p$	: Point
-----	---------

**Returns**

Return the behavior when a point is add by another one

**5.16.3.3 operator+=()**

```
template<typename T>
T_Point& T_Point< T >::operator+= (
    const T_Point< T > & p ) [inline]
```

Operator +=.

**Parameters**

$p$	: Point
-----	---------

**Returns**

Return the behavior when a point is affected and add by another one

#### 5.16.3.4 operator-()

```
template<typename T>
T_Point T_Point< T >::operator- (
    const T_Point< T > & p ) [inline]
```

Operator -.

##### Parameters

$p$	: Point
-----	---------

##### Returns

Return the behavior when a point is subtract by another one

#### 5.16.3.5 operator-=()

```
template<typename T>
T_Point& T_Point< T >::operator-= (
    const T_Point< T > & p ) [inline]
```

Operator -=.

##### Parameters

$p$	: Point
-----	---------

##### Returns

Return the behavior when a point is affected and subtract by another one

#### 5.16.3.6 operator<()

```
template<typename T>
bool T_Point< T >::operator< (
    const T_Point< T > & p ) const [inline]
```

Operator < of a point.

##### Parameters

$p$	: T_Point to compare
-----	----------------------



**Returns**

Return True if the point is strictly weaker, false if not

**5.16.3.7 operator=()**

```
template<typename T>
T_Point& T_Point< T >::operator= (
    const T_Point< T > & p ) [inline]
```

Operator = of a point.

**Parameters**

<i>p</i>	: T_Point to "copy"
----------	---------------------

**Returns**

Return a reference to an atomic point

**5.16.3.8 operator==()**

```
template<typename T>
bool T_Point< T >::operator== (
    const T_Point< T > & p ) const [inline]
```

Operator == of a point.

**Parameters**

<i>p</i>	: T_Point to compare
----------	----------------------

**Returns**

Return True if the point is the same, false if not

**5.16.3.9 operator>()**

```
template<typename T>
bool T_Point< T >::operator> (
    const T_Point< T > & p ) const [inline]
```

Operator > of a point.

**Parameters**

$p$	: <a href="#">T_Point</a> to compare
-----	--------------------------------------

**Returns**

Return True if the point is strictly greater, false if not

**5.16.4 Member Data Documentation****5.16.4.1 x**

```
template<typename T>  
T T\_Point< T >::x
```

Template x for a point

**5.16.4.2 y**

```
template<typename T>  
T T\_Point< T >::y
```

Template y for a point

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

- [include/utis/T\\_Point.hpp](#)

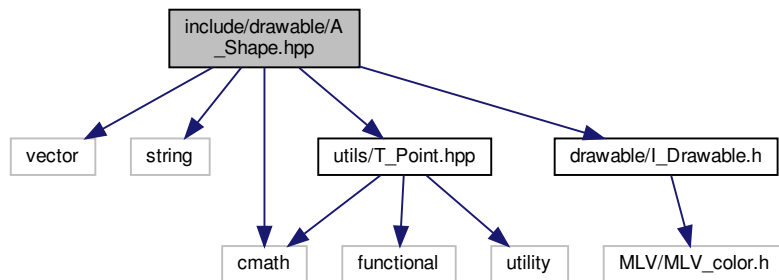
## Chapter 6

# File Documentation

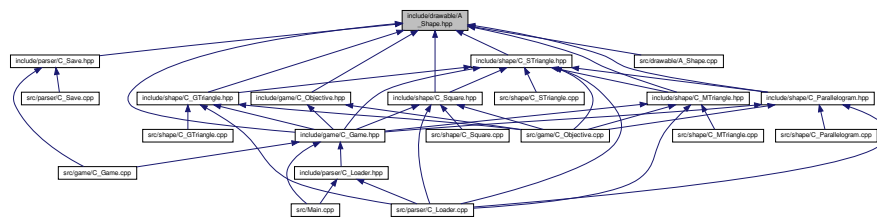
### 6.1 include/drawable/A\_Shape.hpp File Reference

Abstract Class [A\\_Shape](#) of every shape in Tangram.

```
#include <vector>
#include <string>
#include <cmath>
#include <utils/T_Point.hpp>
#include <drawable/I_Drawable.h>
Include dependency graph for A_Shape.hpp:
```



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



## Classes

- class [A\\_Shape](#)

*Abstract Class of every [A\\_Shape](#).*

### 6.1.1 Detailed Description

Abstract Class [A\\_Shape](#) of every shape in Tangram.

#### Author

J  r  mie LE BASTARD

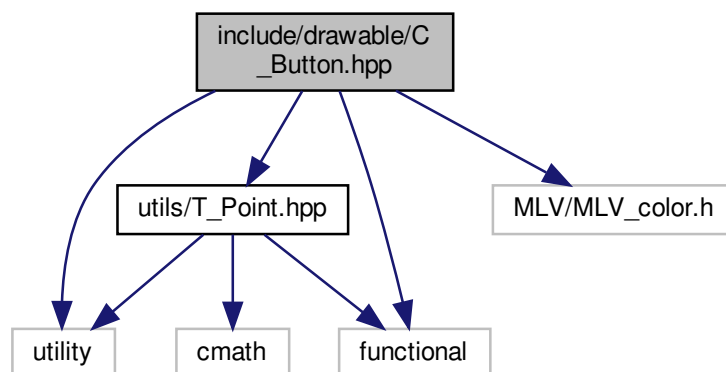
#### Version

1.0

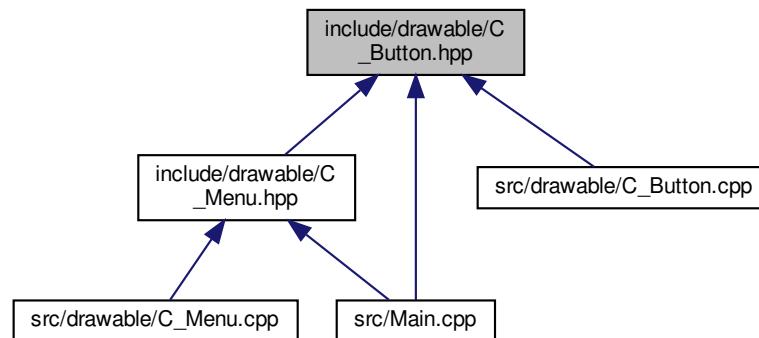
## 6.2 include/drawable/C\_Button.hpp File Reference

Every mButtons of menu.

```
#include <utility>
#include <utils/T_Point.hpp>
#include <functional>
#include <MLV/MLV_color.h>
Include dependency graph for C_Button.hpp:
```



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



## Classes

- class [C\\_Button](#)  
*[C\\_Button](#) of the [C\\_Menu](#).*

### 6.2.1 Detailed Description

Every mButtons of menu.

#### Author

J  r  mie LE BASTARD

#### Version

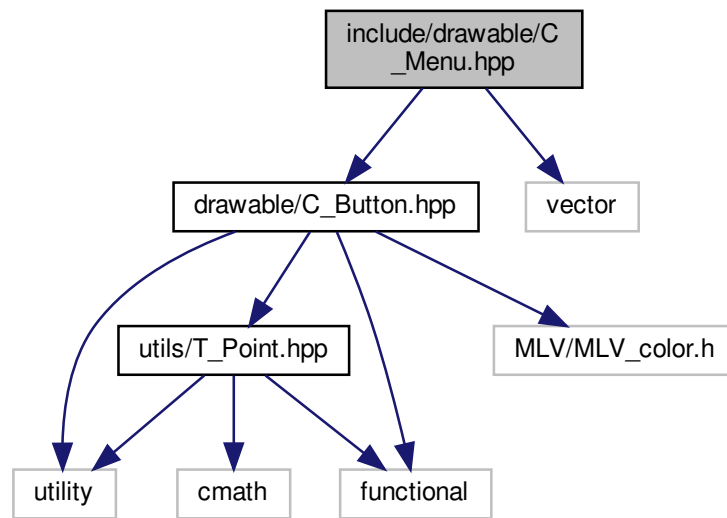
1.0

## 6.3 include/drawable/C\_Menu.hpp File Reference

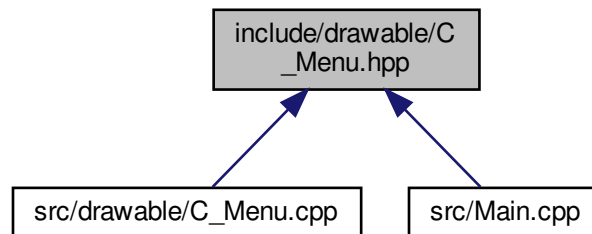
[C\\_Menu](#) of the Tangram's [C\\_Game](#).

```
#include <drawable/C_Button.hpp>  
#include <vector>
```

Include dependency graph for C\_Menu.hpp:



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



## Classes

- class [C\\_Menu](#)  
*C\_Menu of the game.*

### 6.3.1 Detailed Description

[C\\_Menu](#) of the Tangram's [C\\_Game](#).

**Author**

Jérémie LE BASTARD

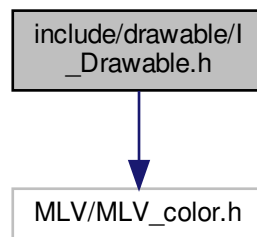
## Version

1.0

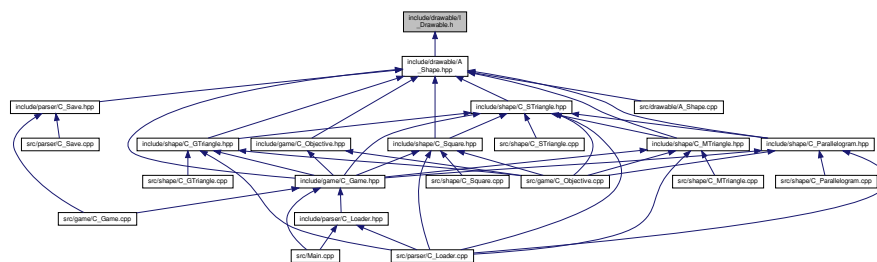
## 6.4 include/drawable/I\_Drawable.h File Reference

```
#include <MLV/MLV_color.h>
```

Include dependency graph for I\_Drawable.h:



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



## Classes

- class | Drawable

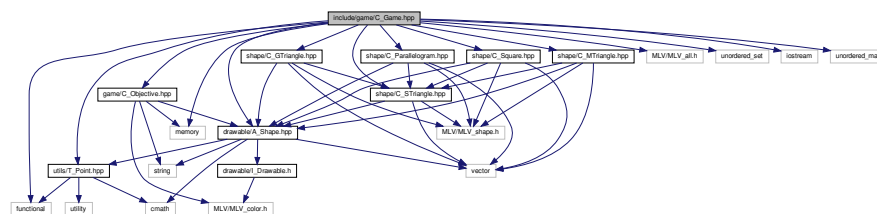
*I\_Drawable* is everything to iDraw.

## 6.5 include/game/C\_Game.hpp File Reference

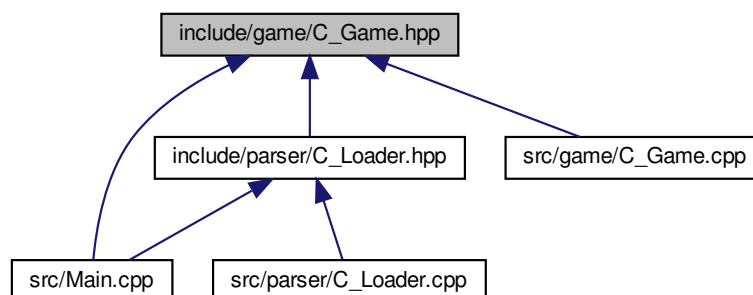
Main [C\\_Game](#) of the Tangram.

```
#include <drawable/A_Shape.hpp>
#include <utils/T_Point.hpp>
#include <game/C_Objective.hpp>
#include <shape/C_STriangle.hpp>
#include <shape/C_MTriangle.hpp>
#include <shape/C_GTriangle.hpp>
#include <shape/C_Parallelogram.hpp>
#include <shape/C_Square.hpp>
#include <MLV/MLV_all.h>
#include <functional>
#include <unordered_set>
#include <memory>
#include <iostream>
#include <unordered_map>
```

Include dependency graph for [C\\_Game.hpp](#):



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



## Classes

- class [C\\_Game](#)

*Class of the main [C\\_Game](#).*



### 6.5.1 Detailed Description

Main [C\\_Game](#) of the Tangram.

Author

J  r  mie LE BASTARD

Version

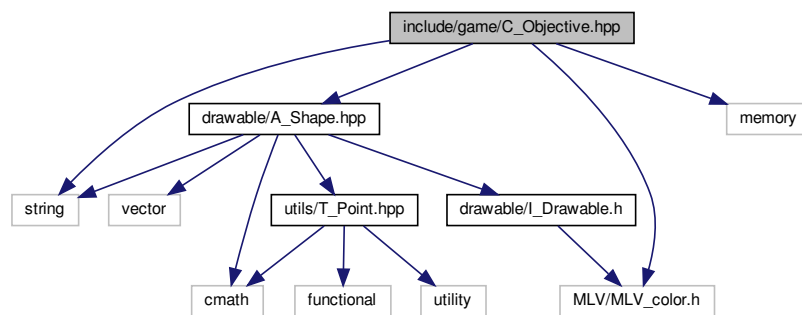
1.0

## 6.6 include/game/C\_Objective.hpp File Reference

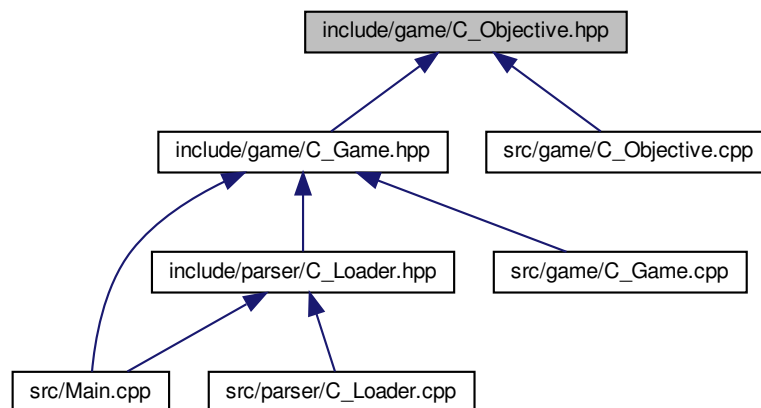
[C\\_Objective](#) of the Tangram's board.

```
#include <drawable/A_Shape.hpp>
#include <string>
#include <memory>
#include <MLV/MLV_color.h>
```

Include dependency graph for [C\\_Objective.hpp](#):



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



## Classes

- class [C\\_Objective](#)  
Class of the board [C\\_Objective](#).

### 6.6.1 Detailed Description

[C\\_Objective](#) of the Tangram's board.

#### Author

J  r  mie LE BASTARD

#### Version

1.0

## 6.7 include/parser/C\_Loader.hpp File Reference

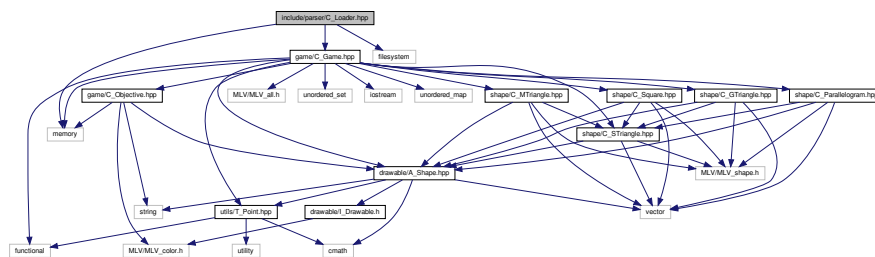
Load a board of Tangram.

```
#include <game/C_Game.hpp>
```

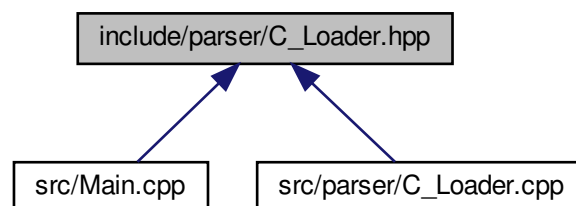
```
#include <filesystem>
```

```
#include <memory>
```

Include dependency graph for C\_Loader.hpp:



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



## Classes

- class [C\\_Loader](#)

Class of the main [C\\_Loader](#).

### 6.7.1 Detailed Description

Load a board of Tangram.

#### Author

J  r  mie LE BASTARD

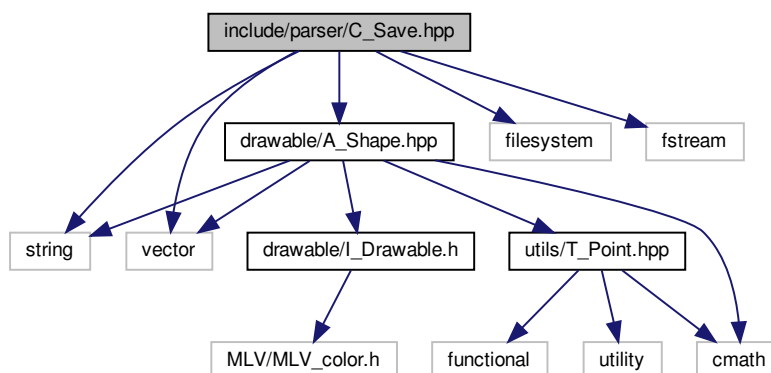
#### Version

1.0

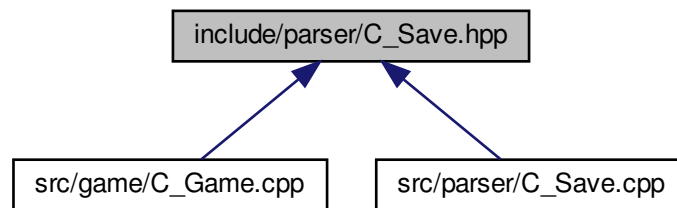
## 6.8 include/parser/C\_Save.hpp File Reference

[C\\_Save](#) a board of Tangram.

```
#include <string>
#include <filesystem>
#include <vector>
#include <fstream>
#include <drawable/A_Shape.hpp>
Include dependency graph for C_Save.hpp:
```



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



## Classes

- class [C\\_Save](#)

*Class of the main Saver.*

### 6.8.1 Detailed Description

[C\\_Save](#) a board of Tangram.

#### Author

J  r  mie LE BASTARD

#### Version

1.0

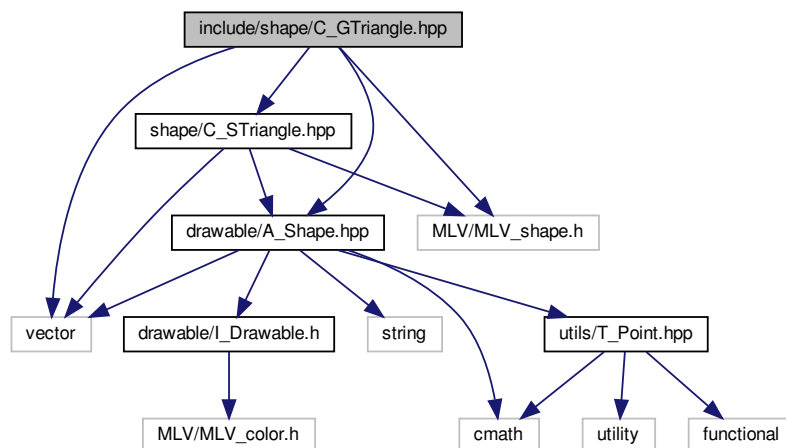
## 6.9 include/shape/C\_GTriangle.hpp File Reference

[A\\_Shape](#) of Great Triangle.

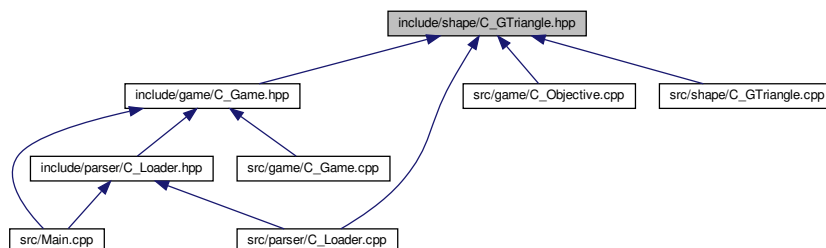
```
#include <vector>
#include <shape/C_STriangle.hpp>
#include <drawable/A_Shape.hpp>
```

```
#include <MLV/MLV_shape.h>
```

Include dependency graph for C\_GTriangle.hpp:



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



## Classes

- class [C\\_GTriangle](#)  
Class of the greatest [C\\_GTriangle](#).

### 6.9.1 Detailed Description

[A\\_Shape](#) of Great Triangle.

#### Author

Jérémie LE BASTARD

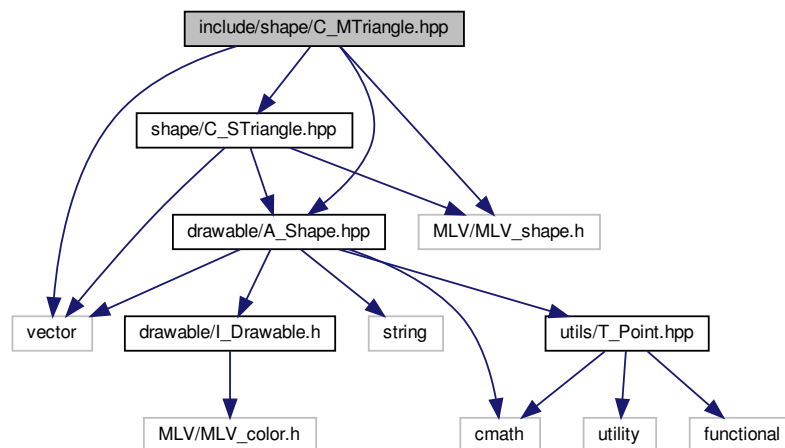
#### Version

1.0

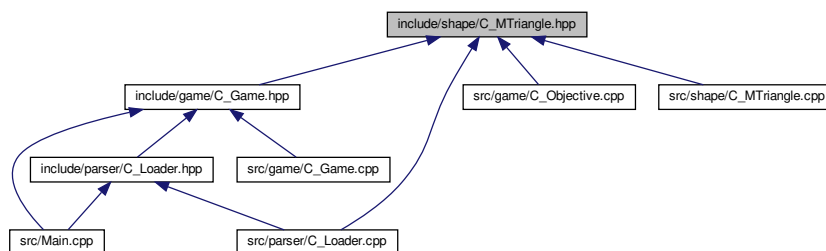
## 6.10 include/shape/C\_MTriangle.hpp File Reference

[A\\_Shape](#) of Medium Triangle.

```
#include <vector>
#include <shape/C_STriangle.hpp>
#include <drawable/A_Shape.hpp>
#include <MLV/MLV_shape.h>
Include dependency graph for C_MTriangle.hpp:
```



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



## Classes

- class [C\\_MTriangle](#)

*Class of the medium [C\\_MTriangle](#).*



## Classes

- class [C\\_Parallelogram](#)  
*Class of the parallelogram.*

### 6.11.1 Detailed Description

[A\\_Shape](#) of [C\\_Parallelogram](#).

#### Author

J  r  mie LE BASTARD

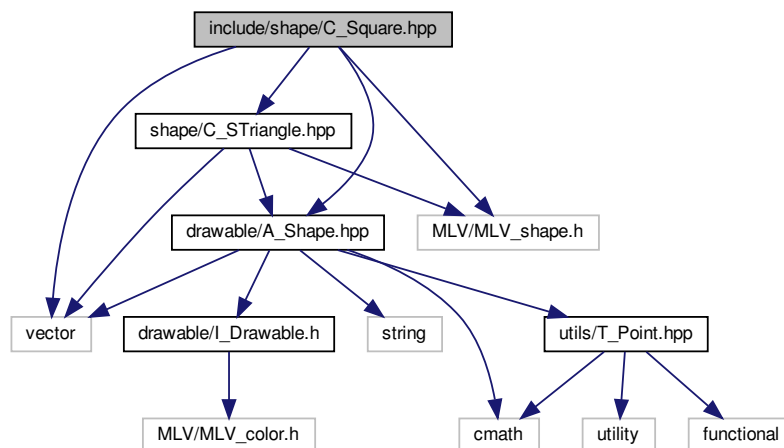
#### Version

1.0

## 6.12 include/shape/C\_Square.hpp File Reference

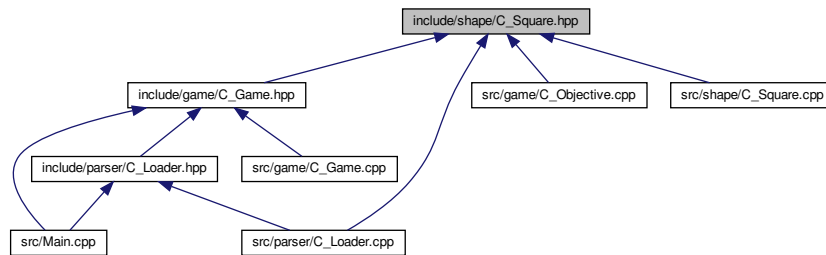
[A\\_Shape](#) of [C\\_Square](#).

```
#include <vector>
#include <shape/C_STriangle.hpp>
#include <drawable/A_Shape.hpp>
#include <MLV/MLV_shape.h>
Include dependency graph for C_Square.hpp:
```





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



## Classes

- class [C\\_Square](#)  
*Class of the square.*

### 6.12.1 Detailed Description

[A\\_Shape](#) of [C\\_Square](#).

Author

J  r  mie LE BASTARD

Version

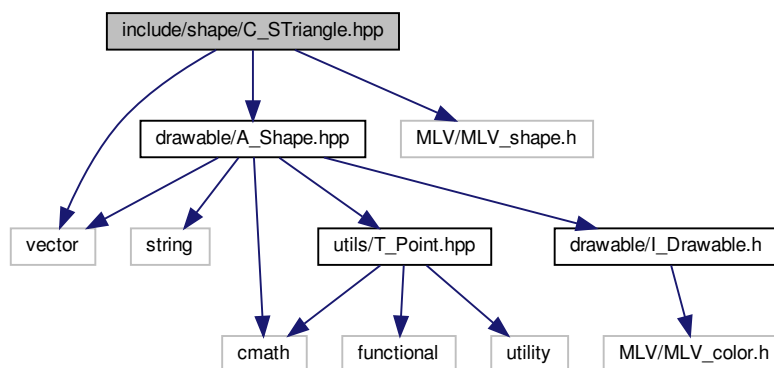
1.0

## 6.13 include/shape/C\_STriangle.hpp File Reference

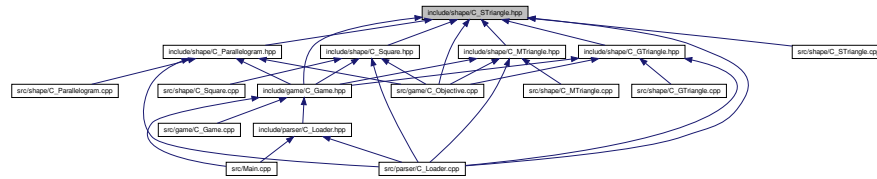
[A\\_Shape](#) of Small Triangle.

```
#include <vector>
#include <drawable/A_Shape.hpp>
#include <MLV/MLV_shape.h>
```

Include dependency graph for C\_STriangle.hpp:



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



## Classes

- class [C\\_STriangle](#)  
Class of the small [C\\_STriangle](#).

### 6.13.1 Detailed Description

[A\\_Shape](#) of Small Triangle.

#### Author

Jérémie LE BASTARD

#### Version

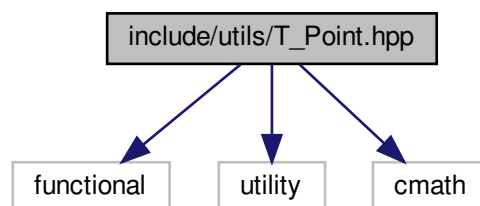
1.0

## 6.14 include/utils/T\_Point.hpp File Reference

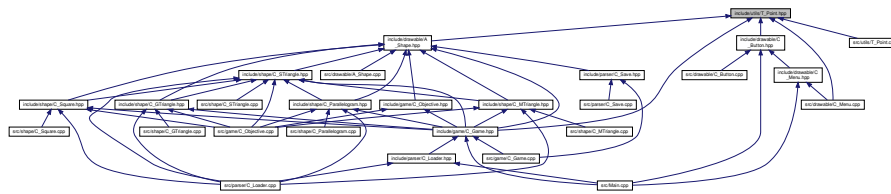
[T\\_Point](#) for every shape and menu.

```
#include <functional>
#include <utility>
#include <cmath>
```

Include dependency graph for [T\\_Point.hpp](#):



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



## Classes

- class `T_Point< T >`  
Class of a `T_Point`.
- struct `T_Point< T >::hash_point`

### 6.14.1 Detailed Description

`T_Point` for every shape and menu.

#### Author

J  r  mie LE BASTARD

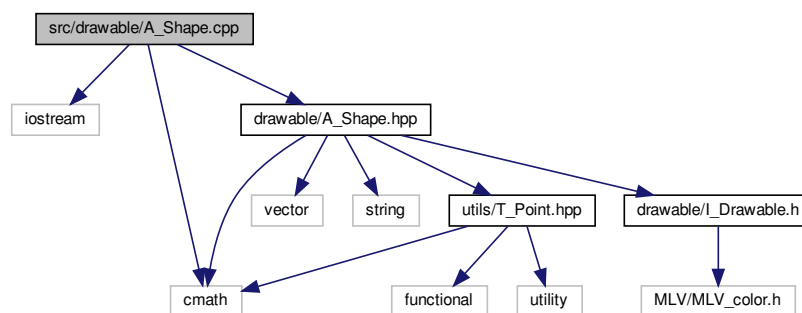
#### Version

1.0

## 6.15 README.md File Reference

### 6.16 src/drawable/A\_Shape.cpp File Reference

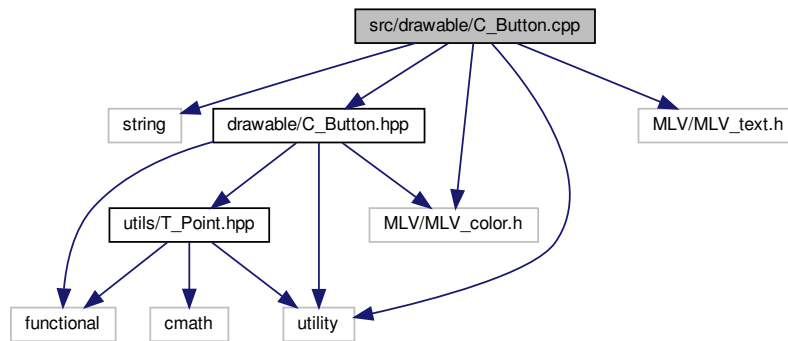
```
#include <iostream>
#include <cmath>
#include <drawable/A_Shape.hpp>
Include dependency graph for A_Shape.cpp:
```



## 6.17 src/drawable/C\_Button.cpp File Reference

```
#include <string>
#include <drawable/C_Button.hpp>
#include <utility>
#include <MLV/MLV_color.h>
#include <MLV/MLV_text.h>
```

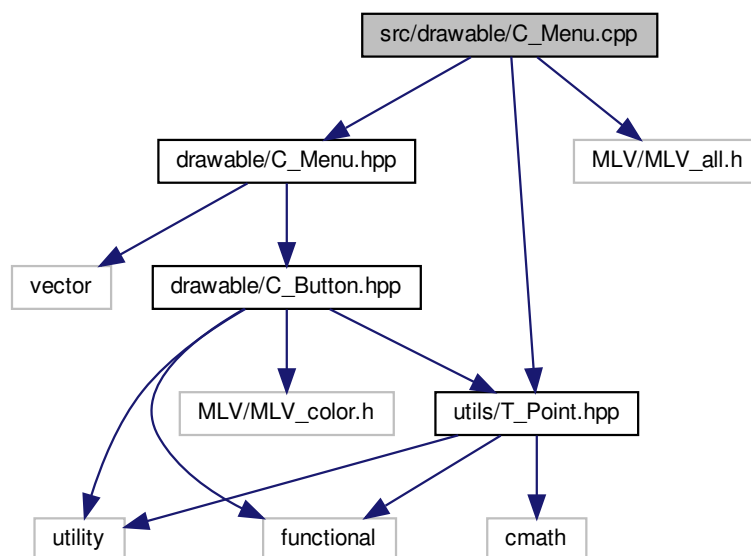
Include dependency graph for C\_Button.cpp:



## 6.18 src/drawable/C\_Menu.cpp File Reference

```
#include <drawable/C_Menu.hpp>
#include <utils/T_Point.hpp>
#include <MLV/MLV_all.h>
```

Include dependency graph for C\_Menu.cpp:

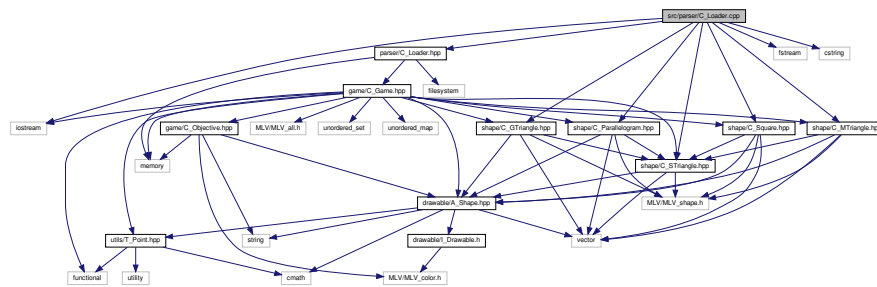






## 6.23 src/parser/C\_Loader.cpp File Reference

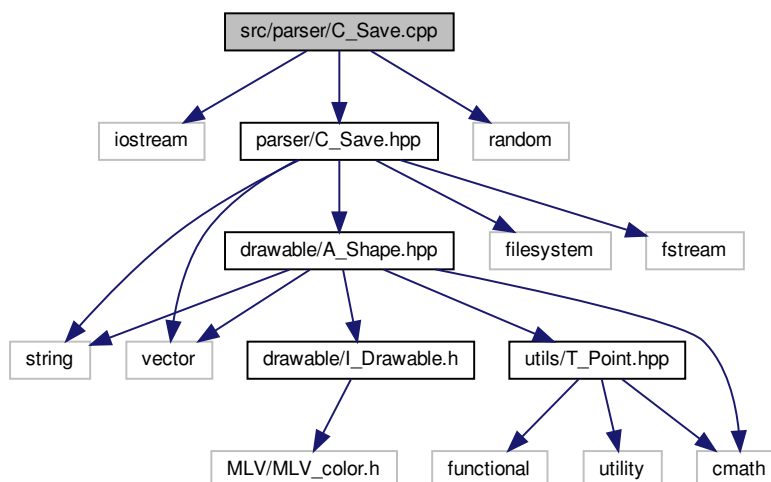
```
#include <iostream>
#include <parser/C_Loader.hpp>
#include <fstream>
#include <cstring>
#include <shape/C_STriangle.hpp>
#include <shape/C_MTriangle.hpp>
#include <shape/C_GTriangle.hpp>
#include <shape/C_Square.hpp>
#include <shape/C_Parallelogram.hpp>
```



## 6.24 src/parser/C\_Save.cpp File Reference

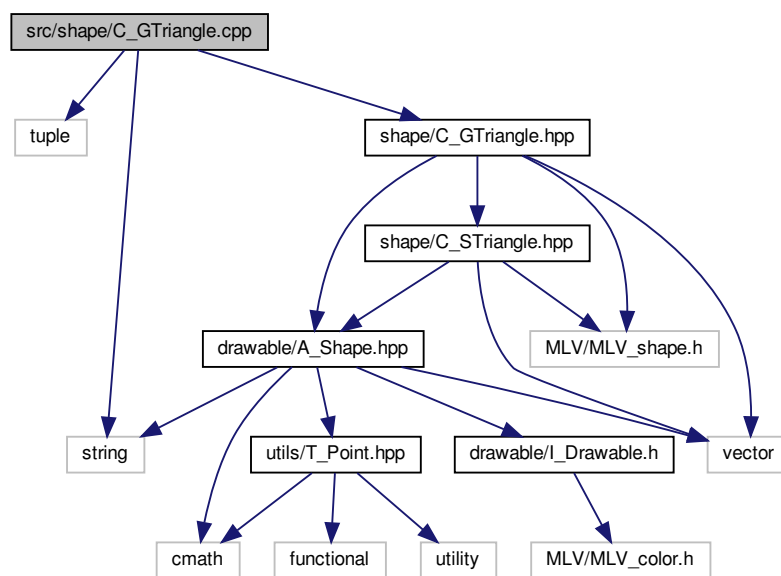
```
#include <iostream>
#include <parser/C_Save.hpp>
#include <random>
```

Include dependency graph for C\_Save.cpp:



## 6.25 src/shape/C\_GTriangle.cpp File Reference

```
#include <tuple>
#include <string>
#include <shape/C_GTriangle.hpp>
Include dependency graph for C_GTriangle.cpp:
```

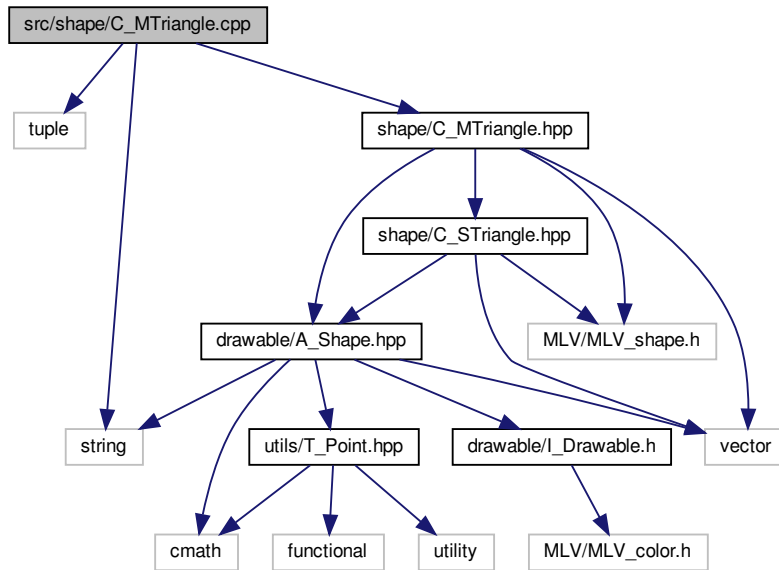


## 6.26 src/shape/C\_MTriangle.cpp File Reference

```
#include <tuple>
#include <string>
#include <shape/C_MTriangle.hpp>
```



Include dependency graph for C\_MTriangle.cpp:



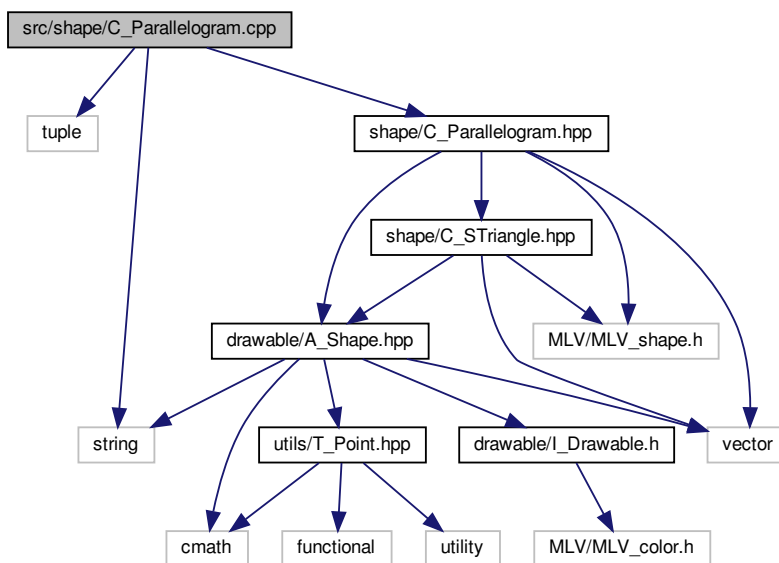
## 6.27 src/shape/C\_Parallelogram.cpp File Reference

```

#include <tuple>
#include <string>
#include <shape/C_Parallelogram.hpp>

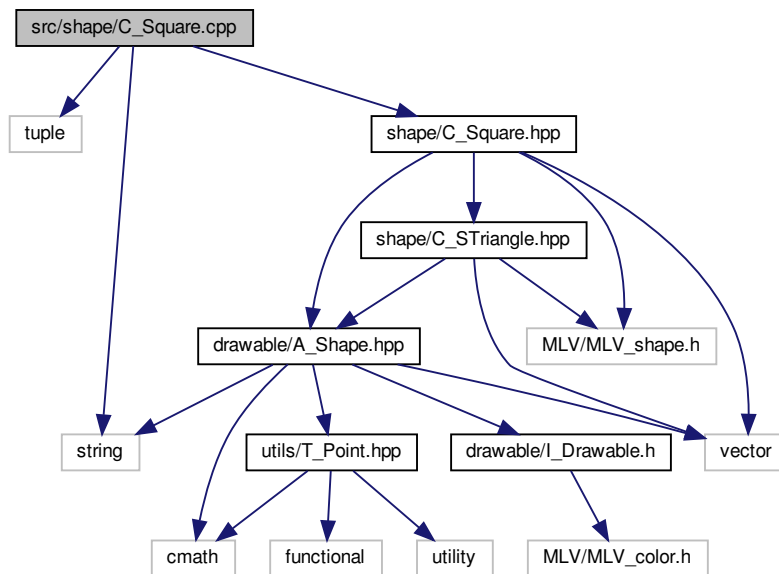
```

Include dependency graph for C\_Parallelogram.cpp:



## 6.28 src/shape/C\_Square.cpp File Reference

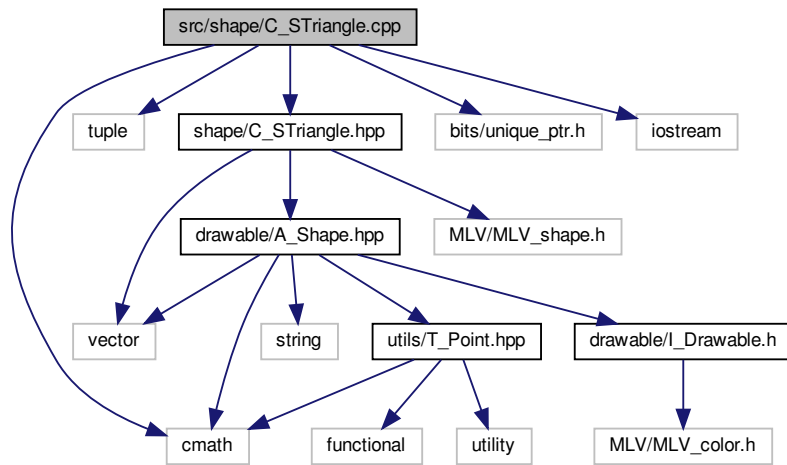
```
#include <tuple>
#include <string>
#include <shape/C_Square.hpp>
Include dependency graph for C_Square.cpp:
```



## 6.29 src/shape/C\_STriangle.cpp File Reference

```
#include <cmath>
#include <tuple>
#include <shape/C_STriangle.hpp>
#include <bits/unique_ptr.h>
#include <iostream>
```

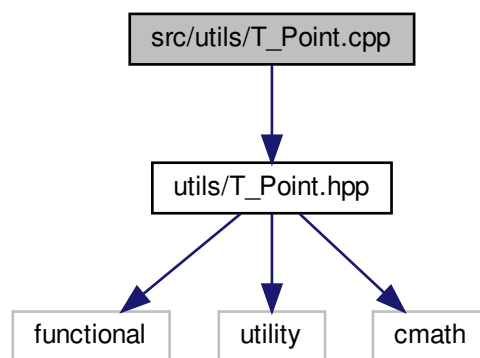
Include dependency graph for C\_STriangle.cpp:



## 6.30 src/shape/C\_STriangle.cpp File Reference

```
#include <utils/T_Point.hpp>
```

Include dependency graph for T\_Point.cpp:





# Index

- ~A\_Shape
  - A\_Shape, [13](#)
- ~C\_Button
  - C\_Button, [19](#)
- ~C\_GTriangle
  - C\_GTriangle, [27](#)
- ~C\_Game
  - C\_Game, [23](#)
- ~C\_MTriangle
  - C\_MTriangle, [39](#)
- ~C\_Menu
  - C\_Menu, [35](#)
- ~C\_Objective
  - C\_Objective, [47](#)
- ~C\_Parallelogram
  - C\_Parallelogram, [52](#)
- ~C\_STriangle
  - C\_STriangle, [73](#)
- ~C\_Square
  - C\_Square, [63](#)
- ~I\_Drawable
  - I\_Drawable, [85](#)
- ~T\_Point
  - T\_Point, [90](#)
- A\_Shape, [11](#)
  - ~A\_Shape, [13](#)
  - aCurrentAngular, [13](#)
  - aGetArea, [14](#)
  - aGetColor, [14](#)
  - aGetPoints, [14](#)
  - aGetShape, [14](#)
  - aGetStatusReverse, [15](#)
  - alsInShape, [15](#)
  - aLeftCorner, [15](#)
  - aLeftFlip, [16](#)
  - aMove, [16](#)
  - aReverse, [16](#)
  - aRightFlip, [16](#)
  - aRotate, [17](#)
  - aSetPoints, [17](#)
  - aToString, [17](#)
  - computeDistance, [17](#)
- aCurrentAngular
  - A\_Shape, [13](#)
  - C\_GTriangle, [29](#)
  - C\_MTriangle, [41](#)
  - C\_Parallelogram, [54](#)
  - C\_STriangle, [74](#)
  - C\_Square, [65](#)
- aGetArea
  - A\_Shape, [14](#)
  - C\_GTriangle, [29](#)
  - C\_MTriangle, [41](#)
  - C\_Parallelogram, [54](#)
  - C\_STriangle, [75](#)
  - C\_Square, [65](#)
- aGetColor
  - A\_Shape, [14](#)
  - C\_GTriangle, [29](#)
  - C\_MTriangle, [41](#)
  - C\_Parallelogram, [54](#)
  - C\_STriangle, [75](#)
  - C\_Square, [65](#)
- aGetPoints
  - A\_Shape, [14](#)
  - C\_GTriangle, [29](#)
  - C\_MTriangle, [41](#)
  - C\_Parallelogram, [54](#)
  - C\_STriangle, [75](#)
  - C\_Square, [65](#)
- aGetShape
  - A\_Shape, [14](#)
  - C\_GTriangle, [30](#)
  - C\_MTriangle, [42](#)
  - C\_Parallelogram, [55](#)
  - C\_STriangle, [75](#)
  - C\_Square, [66](#)
- aGetStatusReverse
  - A\_Shape, [15](#)
  - C\_GTriangle, [30](#)
  - C\_MTriangle, [42](#)
  - C\_Parallelogram, [55](#)
  - C\_STriangle, [76](#)
  - C\_Square, [66](#)
- alsInShape
  - A\_Shape, [15](#)
  - C\_GTriangle, [30](#)
  - C\_MTriangle, [42](#)
  - C\_Parallelogram, [55](#)
  - C\_STriangle, [76](#)
  - C\_Square, [66](#)
- aLeftCorner
  - A\_Shape, [15](#)
  - C\_GTriangle, [31](#)
  - C\_MTriangle, [43](#)
  - C\_Parallelogram, [56](#)
  - C\_STriangle, [76](#)
  - C\_Square, [67](#)

- aLeftFlip
  - A\_Shape, 16
  - C\_GTriangle, 31
  - C\_MTriangle, 43
  - C\_Parallelogram, 56
  - C\_STriangle, 77
  - C\_Square, 67
- aMove
  - A\_Shape, 16
  - C\_GTriangle, 31
  - C\_MTriangle, 43
  - C\_Parallelogram, 56
  - C\_STriangle, 77
  - C\_Square, 67
- aReverse
  - A\_Shape, 16
  - C\_GTriangle, 32
  - C\_MTriangle, 44
  - C\_Parallelogram, 57
  - C\_STriangle, 77
  - C\_Square, 68
- aRightFlip
  - A\_Shape, 16
  - C\_GTriangle, 32
  - C\_MTriangle, 44
  - C\_Parallelogram, 57
  - C\_STriangle, 77
  - C\_Square, 68
- aRotate
  - A\_Shape, 17
  - C\_GTriangle, 32
  - C\_MTriangle, 44
  - C\_Parallelogram, 57
  - C\_STriangle, 78
  - C\_Square, 68
- aSetPoints
  - A\_Shape, 17
  - C\_GTriangle, 32
  - C\_MTriangle, 44
  - C\_Parallelogram, 57
  - C\_STriangle, 78
  - C\_Square, 68
- aToString
  - A\_Shape, 17
  - C\_GTriangle, 32
  - C\_MTriangle, 45
  - C\_Parallelogram, 58
  - C\_STriangle, 78
  - C\_Square, 69
- AddButton
  - C\_Menu, 36
- addShape
  - C\_Game, 23
- BoardCompleted
  - C\_Objective, 48
- C\_Button, 18
  - ~C\_Button, 19
- C\_Button, 19, 20
  - Click, 20
  - ClickInButton, 20
  - Draw, 21
  - SetCallBack, 21
- C\_GTriangle, 24
  - ~C\_GTriangle, 27
  - aCurrentAngular, 29
  - aGetArea, 29
  - aGetColor, 29
  - aGetPoints, 29
  - aGetShape, 30
  - aGetStatusReverse, 30
  - alsInShape, 30
  - aLeftCorner, 31
  - aLeftFlip, 31
  - aMove, 31
  - aReverse, 32
  - aRightFlip, 32
  - aRotate, 32
  - aSetPoints, 32
  - aToString, 32
  - C\_GTriangle, 28
  - iDraw, 33
- C\_Game, 22
  - ~C\_Game, 23
  - addShape, 23
  - C\_Game, 23
  - Clear, 23
  - GetObjectiveColor, 24
  - MainLoop, 24
  - SetObjective, 24
- C\_Loader, 33
  - ParseFile, 34
- C\_MTriangle, 36
  - ~C\_MTriangle, 39
  - aCurrentAngular, 41
  - aGetArea, 41
  - aGetColor, 41
  - aGetPoints, 41
  - aGetShape, 42
  - aGetStatusReverse, 42
  - alsInShape, 42
  - aLeftCorner, 43
  - aLeftFlip, 43
  - aMove, 43
  - aReverse, 44
  - aRightFlip, 44
  - aRotate, 44
  - aSetPoints, 44
  - aToString, 45
  - C\_MTriangle, 40
  - iDraw, 45
- C\_Menu, 35
  - ~C\_Menu, 35
  - AddButton, 36
  - MainLoop, 36
- C\_Objective, 46
  - ~C\_Objective, 46

- ~C\_Objective, 47
- BoardCompleted, 48
- C\_Objective, 47
- Clear, 48
- GetColor, 48
- GetCompleted, 48
- GetObjective, 49
- SetObjective, 49
- C\_Parallelogram, 49
  - ~C\_Parallelogram, 52
  - aCurrentAngular, 54
  - aGetArea, 54
  - aGetColor, 54
  - aGetPoints, 54
  - aGetShape, 55
  - aGetStatusReverse, 55
  - alsInShape, 55
  - aLeftCorner, 56
  - aLeftFlip, 56
  - aMove, 56
  - aReverse, 57
  - aRightFlip, 57
  - aRotate, 57
  - aSetPoints, 57
  - aToString, 58
  - C\_Parallelogram, 53
  - iDraw, 58
- C\_STriangle, 70
  - ~C\_STriangle, 73
  - aCurrentAngular, 74
  - aGetArea, 75
  - aGetColor, 75
  - aGetPoints, 75
  - aGetShape, 75
  - aGetStatusReverse, 76
  - alsInShape, 76
  - aLeftCorner, 76
  - aLeftFlip, 77
  - aMove, 77
  - aReverse, 77
  - aRightFlip, 77
  - aRotate, 78
  - aSetPoints, 78
  - aToString, 78
  - C\_STriangle, 73, 74
  - CenterPoint, 79
  - GetCenterPoint, 79
  - GetFlip, 79
  - iDraw, 79, 80
  - IsInSTriangle, 80
  - LeftFlip, 80
  - Reverse, 81
  - RightFlip, 81
  - Rotate, 81
- C\_Save, 59
  - C\_Save, 59
  - Save, 59
- C\_Square, 60
  - ~C\_Square, 63
  - aCurrentAngular, 65
  - aGetArea, 65
  - aGetColor, 65
  - aGetPoints, 65
  - aGetShape, 66
  - aGetStatusReverse, 66
  - alsInShape, 66
  - aLeftCorner, 67
  - aLeftFlip, 67
  - aMove, 67
  - aReverse, 68
  - aRightFlip, 68
  - aRotate, 68
  - aSetPoints, 68
  - aToString, 69
  - C\_Square, 64
  - iDraw, 69
- CenterPoint
  - C\_STriangle, 79
- Clear
  - C\_Game, 23
  - C\_Objective, 48
- Click
  - C\_Button, 20
- ClickInButton
  - C\_Button, 20
- computeDistance
  - A\_Shape, 17
- Draw
  - C\_Button, 21
- GetCenterPoint
  - C\_STriangle, 79
- GetColor
  - C\_Objective, 48
- GetCompleted
  - C\_Objective, 48
- GetFlip
  - C\_STriangle, 79
- GetObjective
  - C\_Objective, 49
- GetObjectiveColor
  - C\_Game, 24
- I\_Drawable, 83
  - ~I\_Drawable, 85
  - iDraw, 85
- iDraw
  - C\_GTriangle, 33
  - C\_MTriangle, 45
  - C\_Parallelogram, 58
  - C\_STriangle, 79, 80
  - C\_Square, 69
  - I\_Drawable, 85
- include/drawable/A\_Shape.hpp, 95
- include/drawable/C\_Button.hpp, 96
- include/drawable/C\_Menu.hpp, 97

- include/drawable/I\_Drawable.h, 99
- include/game/C\_Game.hpp, 100
- include/game/C\_Objective.hpp, 101
- include/parser/C\_Loader.hpp, 102
- include/parser/C\_Save.hpp, 103
- include/shape/C\_GTriangle.hpp, 104
- include/shape/C\_MTriangle.hpp, 106
- include/shape/C\_Parallelogram.hpp, 107
- include/shape/C\_STriangle.hpp, 109
- include/shape/C\_Square.hpp, 108
- include/utils/T\_Point.hpp, 110
- IsInSTriangle
  - C\_STriangle, 80
- LeftFlip
  - C\_STriangle, 80
- main
  - Main.cpp, 114
- Main.cpp
  - main, 114
  - page, 114
- MainLoop
  - C\_Game, 24
  - C\_Menu, 36
- operator!=
  - T\_Point, 90
- operator<
  - T\_Point, 92
- operator>
  - T\_Point, 93
- operator()
  - T\_Point::hash\_point, 82, 83
- operator+
  - T\_Point, 91
- operator+=
  - T\_Point, 91
- operator-
  - T\_Point, 91
- operator-=
  - T\_Point, 92
- operator=
  - T\_Point, 93
- operator==
  - T\_Point, 93
- page
  - Main.cpp, 114
- ParseFile
  - C\_Loader, 34
- README.md, 111
- Reverse
  - C\_STriangle, 81
- RightFlip
  - C\_STriangle, 81
- Rotate
  - C\_STriangle, 81
- Save
  - C\_Save, 59
- SetCallBack
  - C\_Button, 21
- SetObjective
  - C\_Game, 24
  - C\_Objective, 49
- src/Main.cpp, 113
- src/drawable/A\_Shape.cpp, 111
- src/drawable/C\_Button.cpp, 112
- src/drawable/C\_Menu.cpp, 112
- src/drawable/I\_Drawable.cpp, 113
- src/game/C\_Game.cpp, 113
- src/game/C\_Objective.cpp, 113
- src/parser/C\_Loader.cpp, 115
- src/parser/C\_Save.cpp, 115
- src/shape/C\_GTriangle.cpp, 116
- src/shape/C\_MTriangle.cpp, 116
- src/shape/C\_Parallelogram.cpp, 117
- src/shape/C\_STriangle.cpp, 118
- src/shape/C\_Square.cpp, 118
- src/utils/T\_Point.cpp, 119
- Struct, 86
- T\_Point
  - ~T\_Point, 90
  - operator!=, 90
  - operator<, 92
  - operator>, 93
  - operator+, 91
  - operator+=, 91
  - operator-, 91
  - operator-=, 92
  - operator=, 93
  - operator==, 93
  - T\_Point, 89, 90
  - x, 94
  - y, 94
- T\_Point< T >, 86
- T\_Point< T >::hash\_point, 82
- T\_Point::hash\_point
  - operator(), 82, 83
- x
  - T\_Point, 94
- y
  - T\_Point, 94