

Tangram

Generated by Doxygen 1.8.13

Contents

1	Tangram	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Class Documentation	9
5.1	Button Class Reference	9
5.1.1	Detailed Description	9
5.1.2	Constructor & Destructor Documentation	9
5.1.2.1	Button() [1/2]	10
5.1.2.2	Button() [2/2]	10
5.1.3	Member Function Documentation	10
5.1.3.1	click()	10
5.1.3.2	click_in_button()	11
5.1.3.3	set_callback()	11
5.2	Drawable Class Reference	11
5.2.1	Detailed Description	12
5.3	Game Class Reference	12
5.3.1	Detailed Description	13

5.3.2	Constructor & Destructor Documentation	13
5.3.2.1	Game()	13
5.3.3	Member Function Documentation	13
5.3.3.1	add_shape()	13
5.4	GTriangle Class Reference	14
5.4.1	Detailed Description	15
5.4.2	Constructor & Destructor Documentation	15
5.4.2.1	GTriangle() [1/3]	15
5.4.2.2	GTriangle() [2/3]	15
5.4.2.3	GTriangle() [3/3]	16
5.4.3	Member Function Documentation	16
5.4.3.1	get_Points()	16
5.4.3.2	is_in_shape()	16
5.4.3.3	move()	17
5.4.3.4	rotate()	17
5.4.3.5	toString()	17
5.5	Loader Class Reference	18
5.5.1	Detailed Description	18
5.5.2	Member Function Documentation	18
5.5.2.1	parse_file()	18
5.6	Menu Class Reference	19
5.6.1	Detailed Description	19
5.6.2	Member Function Documentation	19
5.6.2.1	add_button()	19
5.7	MTriangle Class Reference	20
5.7.1	Detailed Description	21
5.7.2	Constructor & Destructor Documentation	21
5.7.2.1	MTriangle() [1/3]	21
5.7.2.2	MTriangle() [2/3]	22
5.7.2.3	MTriangle() [3/3]	22

5.7.3	Member Function Documentation	22
5.7.3.1	get_Points()	22
5.7.3.2	is_in_shape()	23
5.7.3.3	move()	24
5.7.3.4	rotate()	24
5.7.3.5	toString()	24
5.8	Objective Class Reference	25
5.8.1	Detailed Description	25
5.8.2	Member Function Documentation	25
5.8.2.1	boardCompleted()	25
5.8.2.2	get_Objective()	26
5.9	Parallelogram Class Reference	26
5.9.1	Detailed Description	27
5.9.2	Constructor & Destructor Documentation	28
5.9.2.1	Parallelogram() [1/3]	28
5.9.2.2	Parallelogram() [2/3]	28
5.9.2.3	Parallelogram() [3/3]	28
5.9.3	Member Function Documentation	29
5.9.3.1	get_Points()	29
5.9.3.2	is_in_shape()	29
5.9.3.3	move()	29
5.9.3.4	rotate()	30
5.9.3.5	toString()	30
5.10	Point< T > Class Template Reference	30
5.10.1	Detailed Description	31
5.10.2	Constructor & Destructor Documentation	31
5.10.2.1	Point()	31
5.10.3	Member Function Documentation	32
5.10.3.1	operator!=(())	32
5.10.3.2	operator<()	32

5.10.3.3	operator=()	33
5.10.3.4	operator==()	33
5.10.3.5	operator>()	33
5.10.4	Member Data Documentation	34
5.10.4.1	x	34
5.10.4.2	y	34
5.11	Save Class Reference	34
5.11.1	Detailed Description	34
5.12	Shape Class Reference	35
5.12.1	Detailed Description	36
5.12.2	Member Function Documentation	36
5.12.2.1	get_Points()	36
5.12.2.2	is_in_shape()	36
5.12.2.3	move()	36
5.12.2.4	rotate()	37
5.12.2.5	toString()	37
5.13	Square Class Reference	38
5.13.1	Detailed Description	39
5.13.2	Constructor & Destructor Documentation	39
5.13.2.1	Square() [1/3]	39
5.13.2.2	Square() [2/3]	39
5.13.2.3	Square() [3/3]	40
5.13.3	Member Function Documentation	40
5.13.3.1	get_Points()	40
5.13.3.2	is_in_shape()	40
5.13.3.3	move()	41
5.13.3.4	rotate()	41
5.13.3.5	toString()	41
5.14	STriangle Class Reference	42
5.14.1	Detailed Description	43

5.14.2	Constructor & Destructor Documentation	44
5.14.2.1	STriangle() [1/4]	44
5.14.2.2	STriangle() [2/4]	44
5.14.2.3	STriangle() [3/4]	44
5.14.2.4	STriangle() [4/4]	45
5.14.3	Member Function Documentation	45
5.14.3.1	center_point()	45
5.14.3.2	computeDistance()	45
5.14.3.3	draw()	46
5.14.3.4	get_center_point()	46
5.14.3.5	get_Points()	46
5.14.3.6	is_in_shape()	46
5.14.3.7	is_in_triangle()	47
5.14.3.8	move()	47
5.14.3.9	rotate()	47
5.14.3.10	toString()	48
6	File Documentation	49
6.1	include/drawable/Button.hpp File Reference	49
6.1.1	Detailed Description	50
6.2	include/drawable/Menu.hpp File Reference	50
6.2.1	Detailed Description	51
6.3	include/drawable/Shape.hpp File Reference	51
6.3.1	Detailed Description	52
6.4	include/game/Game.hpp File Reference	52
6.4.1	Detailed Description	53
6.5	include/game/Objective.hpp File Reference	53
6.5.1	Detailed Description	54
6.6	include/parser/Loader.hpp File Reference	54
6.6.1	Detailed Description	55
6.7	include/parser/Save.hpp File Reference	55

6.7.1	Detailed Description	56
6.8	include/shape/GTriangle.hpp File Reference	56
6.8.1	Detailed Description	57
6.9	include/shape/MTriangle.hpp File Reference	57
6.9.1	Detailed Description	58
6.10	include/shape/Parallelogram.hpp File Reference	58
6.10.1	Detailed Description	59
6.11	include/shape/Square.hpp File Reference	59
6.11.1	Detailed Description	60
6.12	include/shape/STriangle.hpp File Reference	60
6.12.1	Detailed Description	61
6.13	include/utils/Point.hpp File Reference	61
6.13.1	Detailed Description	61
Index		63

Chapter 1

Tangram

A student project about the tangram's game

How to run

When you're in the repository

```
cd cmake-build-debug
make
./tangram
```

Documentation

Here there is HTML files, LaTeX files and PDF.

HTML

```
cd doc/html
```

LaTeX

```
cd doc/latex
```

PDF

```
cd doc/latex
./refman.pdf
```

Regenerate Documentation

You can generate this document as you wish. If you're updating the code and the doc, you should do :

In the root directory of this project :

```
doxygen config-file
cd doc/latex
make
```


Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Button	9
Drawable	11
Shape	35
GTriangle	14
MTriangle	20
Parallelogram	26
Square	38
STriangle	42
Game	12
Loader	18
Menu	19
Objective	25
Point< T >	30
Point< double >	30
Point< int >	30
Save	34

Chapter 3

Class Index

3.1 Class List

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

Button		
Button	Button of the Menu	9
Drawable		
Drawable	Drawable is everything to draw	11
Game		
Game	Class of the main Game	12
GTriangle		
GTriangle	Class of the greatest triangle	14
Loader		
Loader	Class of the main Loader	18
Menu		
Menu	Menu of the game	19
MTriangle		
MTriangle	Class of the medium triangle	20
Objective		
Objective	Class of the board Objective	25
Parallelogram		
Parallelogram	Class of the parallelogram	26
Point< T >		
Point< T >	Class of a Point	30
Save		
Save	Class of the main Saver	34
Shape		
Shape	Abstract Class of every Shape	35
Square		
Square	Class of the square	38
STriangle		
STriangle	Class of the small triangle	42

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

include/drawable/ Button.hpp	
Every buttons of menu	49
include/drawable/ Drawable.h	??
include/drawable/ Menu.hpp	
Menu of the Tangram's Game	50
include/drawable/ Shape.hpp	
Abstract Class Shape of every shape in Tangram	51
include/game/ Game.hpp	
Main Game of the Tangram	52
include/game/ Objective.hpp	
Objective of the Tangram's board	53
include/parser/ Loader.hpp	
Load a board of Tangram	54
include/parser/ Save.hpp	
Save a board of Tangram	55
include/shape/ GTriangle.hpp	
Shape of Great Triangle	56
include/shape/ MTriangle.hpp	
Shape of Medium Triangle	57
include/shape/ Parallelogram.hpp	
Shape of Parallelogram	58
include/shape/ Square.hpp	
Shape of Square	59
include/shape/ STriangle.hpp	
Shape of Small Triangle	60
include/utils/ Point.hpp	
Point for every shape and menu	61

Chapter 5

Class Documentation

5.1 Button Class Reference

[Button](#) of the [Menu](#).

```
#include <Button.hpp>
```

Public Member Functions

- [~Button](#) ()
Destructor of the [Button](#).
- [Button](#) ([Point](#)< int > point, [Point](#)< int > sizing, std::string text)
Constructor of a [Button](#).
- [Button](#) ([Point](#)< int > point, [Point](#)< int > sizing, std::string text, std::function< int(int)> callback)
Constructor of a [Button](#).
- bool [click_in_button](#) (const [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 [set_callback](#) (std::function< int(int)> callback)
Set a callback for a button.

5.1.1 Detailed Description

[Button](#) of the [Menu](#).

This class manage all buttons of the menu

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Button() [1/2]

```
Button::Button (
    Point< int > point,
    Point< int > sizing,
    std::string text )
```

Constructor of a [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

5.1.2.2 Button() [2/2]

```
Button::Button (
    Point< int > point,
    Point< int > sizing,
    std::string text,
    std::function< int(int)> callback )
```

Constructor of a [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

5.1.3 Member Function Documentation

5.1.3.1 click()

```
int Button::click (
    int val )
```

Define a value about a click.

Returns

Return a value about a click

5.1.3.2 click_in_button()

```
bool Button::click_in_button (
    const Point< int > & click )
```

Check if a click is in the button.

Parameters

<i>click</i>	: Point to check
--------------	----------------------------------

Returns

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

5.1.3.3 set_callback()

```
void Button::set_callback (
    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:

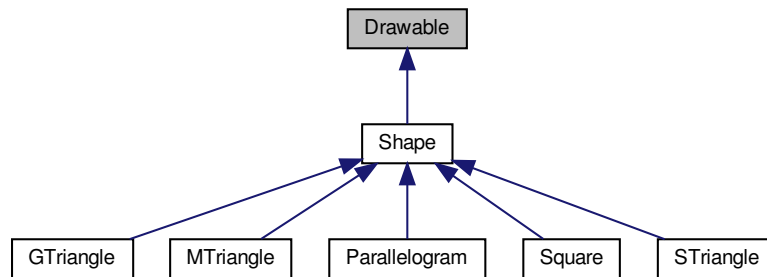
- include/drawable/[Button.hpp](#)
- src/drawable/Button.cpp

5.2 Drawable Class Reference

[Drawable](#) is everything to draw.

```
#include <Drawable.h>
```

Inheritance diagram for Drawable:



Public Member Functions

- virtual void `draw` ()=0
Pure virtual function. Draw everything which needs to be draw.

5.2.1 Detailed Description

`Drawable` is everything to draw.

This class manage everything drawing

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

- include/drawable/Drawable.h

5.3 Game Class Reference

Class of the main `Game`.

```
#include <Game.hpp>
```

Public Member Functions

- void `main_loop` ()
Main loop of the game.
- `Game` (int w, int h)
Constructor of the game, initialize a game with an sizing.
- void `add_shape` (`Shape` *s)
Add a shape in the game.
- void `clear` ()
Clear the game / the board.

5.3.1 Detailed Description

Class of the main [Game](#).

This class manage everything about the main game

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Game()

```
Game::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 add_shape()

```
void Game::add_shape (
    Shape * s )
```

Add a shape in the game.

Parameters

<i>s</i>	: Shape to add
----------	--------------------------------

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

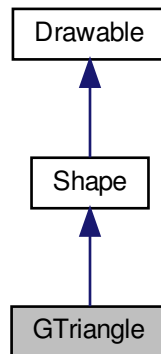
- include/game/[Game.hpp](#)
- src/game/Game.cpp

5.4 GTriangle Class Reference

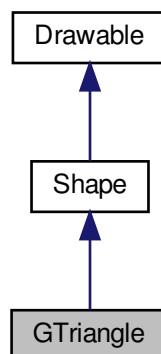
Class of the greatest triangle.

```
#include <GTriangle.hpp>
```

Inheritance diagram for GTriangle:



Collaboration diagram for GTriangle:



Public Member Functions

- [~GTriangle](#) () override
Destructor of [GTriangle](#).
- [GTriangle](#) (MLV_Color color=MLV_COLOR_RED)

- Constructor by default of [GTriangle](#), make a triangle as default.*
- [GTriangle](#) (const std::vector< [STriangle](#) > &triangle, MLV_Color color=MLV_COLOR_RED)
- Constructor of [GTriangle](#), requires a vector of triangles.*
- [GTriangle](#) (const [Point](#)< double > &origin, double angular=0.0, MLV_Color color=MLV_COLOR_RED)
- Constructor of [GTriangle](#), calls the delegate Default Constructor.*
- void [move](#) (const [Point](#)< double > &translation) override
- Move the [GTriangle](#) by point translation.*
- void [rotate](#) (double angular) override
- Rotate the [GTriangle](#) with specified angular.*
- void [flip](#) () override
- Flip the figure as symmetry.*
- void [draw](#) () override
- Draw this shape on IHM.*
- bool [is_in_shape](#) (const [Point](#)< double > &click) override
- Check if a point is in this shape.*
- std::vector< [Point](#)< double > > [get_Points](#) () override
- Get points of this shape.*
- std::string [toString](#) () override
- Convert all data of [GTriangle](#) in a string.*

5.4.1 Detailed Description

Class of the greatest triangle.

This class manage everything about the greatest triangle

5.4.2 Constructor & Destructor Documentation

5.4.2.1 GTriangle() [1/3]

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

Constructor by default of [GTriangle](#), make a triangle as default.

Parameters

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

5.4.2.2 GTriangle() [2/3]

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

Constructor of [GTriangle](#), requires a vector of triangles.

Parameters

<i>triangle</i>	: The GTriangle will created with a vector of STriangle (4)
<i>color</i>	: Optional parameter, color of this shape

5.4.2.3 [GTriangle\(\)](#) [3/3]

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

Constructor of [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), rotate the figure with an angular
<i>color</i>	: Optional parameter, color of this shape

5.4.3 Member Function Documentation

5.4.3.1 [get_Points\(\)](#)

```
std::vector< Point< double > > GTriangle::get_Points ( ) [override], [virtual]
```

Get points of this shape.

Returns

Return a vector of points of this shape

Implements [Shape](#).

5.4.3.2 [is_in_shape\(\)](#)

```
bool GTriangle::is_in_shape (
    const Point< double > & click ) [override], [virtual]
```

Check if a point is in this shape.

Parameters

<i>click</i>	: Point to check
--------------	----------------------------------

Returns

true if click is in this shape, false if not

Implements [Shape](#).

5.4.3.3 move()

```
void GTriangle::move (
    const Point< double > & translation ) [override], [virtual]
```

Move the [GTriangle](#) by point translation.

Parameters

<i>translation</i>	: Every points of this shape will be translate by this parameter
--------------------	--

Implements [Shape](#).

5.4.3.4 rotate()

```
void GTriangle::rotate (
    double angular ) [override], [virtual]
```

Rotate the [GTriangle](#) with specified angular.

Parameters

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

Implements [Shape](#).

5.4.3.5 toString()

```
std::string GTriangle::toString ( ) [override], [virtual]
```

Convert all data of [GTriangle](#) in a string.

Returns

Return a string which contains every points of this shape

Implements [Shape](#).

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

- include/shape/GTriangle.hpp
- src/shape/GTriangle.cpp

5.5 Loader Class Reference

Class of the main [Loader](#).

```
#include <Loader.hpp>
```

Static Public Member Functions

- static bool [parse_file](#) (const std::string &filename, [Game](#) &game)
Parse a file to make a board.

5.5.1 Detailed Description

Class of the main [Loader](#).

This class manage everything about the loader

5.5.2 Member Function Documentation

5.5.2.1 [parse_file\(\)](#)

```
bool Loader::parse_file (
    const std::string & filename,
    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/[Loader.hpp](#)
- src/parser/Loader.cpp

5.6 Menu Class Reference

[Menu](#) of the game.

```
#include <Menu.hpp>
```

Public Member Functions

- void [add_button](#) ([Button](#) button)
Add a button in the [Menu](#).
- void [main_loop](#) ()
Main loop of the [Menu](#).

5.6.1 Detailed Description

[Menu](#) of the game.

This class manage everything about Tangram's menu

5.6.2 Member Function Documentation

5.6.2.1 [add_button\(\)](#)

```
void Menu::add_button (  
    Button button )
```

Add a button in the [Menu](#).

Parameters

<i>button</i>	: Button to add
---------------	---------------------------------

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

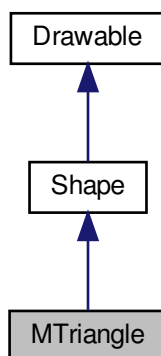
- [include/drawable/Menu.hpp](#)
- [src/drawable/Menu.cpp](#)

5.7 MTriangle Class Reference

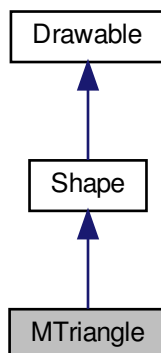
Class of the medium triangle.

```
#include <MTriangle.hpp>
```

Inheritance diagram for MTriangle:



Collaboration diagram for MTriangle:



Public Member Functions

- `~MTriangle ()` override
Destructor of [MTriangle](#).
- `MTriangle (MLV_Color color=MLV_COLOR_ORANGE)`
Constructor by default of [MTriangle](#), make a [MTriangle](#) as default.
- `MTriangle (const std::vector< STriangle > &triangle, MLV_Color color=MLV_COLOR_ORANGE)`
Constructor of [MTriangle](#), requires a vector of [STriangles](#).
- `MTriangle (const Point< double > &origin, double angular=0.0, MLV_Color color=MLV_COLOR_ORANGE)`
Constructor of [MTriangle](#), calls the delegate Default Constructor.
- `void move (const Point< double > &translation)` override
Move the [MTriangle](#) by point translation.
- `void rotate (double angular)` override
Rotate the [MTriangle](#) with specified angular.
- `void flip ()` override
Flip the figure as symmetry.
- `void draw ()` override
Draw this shape on IHM.
- `bool is_in_shape (const Point< double > &click)` override
Check if a point is in this shape.
- `std::vector< Point< double > > get_Points ()` override
Get points of this shape.
- `std::string toString ()` override
Convert all data of [MTriangle](#) in a string.

5.7.1 Detailed Description

Class of the medium triangle.

This class manage everything about the medium triangle

5.7.2 Constructor & Destructor Documentation

5.7.2.1 MTriangle() [1/3]

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

Constructor by default of [MTriangle](#), make a [MTriangle](#) as default.

Parameters

<code>color</code>	: Optional parameter, color of this shape
--------------------	---

5.7.2.2 MTriangle() [2/3]

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

Constructor of [MTriangle](#), requires a vector of STriangles.

Parameters

<i>triangle</i>	: The MTriangle will created with a vector of STriangle (4)
<i>color</i>	: Optional parameter, color of this shape

5.7.2.3 MTriangle() [3/3]

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

Constructor of [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), rotate the figure with an angular
<i>color</i>	: Optional parameter, color of this shape

5.7.3 Member Function Documentation

5.7.3.1 get_Points()

```
std::vector< Point< double > > MTriangle::get_Points ( ) [override], [virtual]
```

Get points of this shape.

Returns

Return a vector of points of this shape

Implements [Shape](#).

5.7.3.2 is_in_shape()

```
bool MTriangle::is_in_shape (
    const Point< double > & click ) [override], [virtual]
```

Check if a point is in this shape.

Parameters

<i>click</i>	: Point to check
--------------	----------------------------------

Returns

true if click is in this shape, false if not

Implements [Shape](#).

5.7.3.3 move()

```
void MTriangle::move (  
    const Point< double > & translation ) [override], [virtual]
```

Move the [MTriangle](#) by point translation.

Parameters

<i>translation</i>	: Every points of this shape will be translate by this parameter
--------------------	--

Implements [Shape](#).

5.7.3.4 rotate()

```
void MTriangle::rotate (  
    double angular ) [override], [virtual]
```

Rotate the [MTriangle](#) with specified angular.

Parameters

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

Implements [Shape](#).

5.7.3.5 toString()

```
std::string MTriangle::toString ( ) [override], [virtual]
```

Convert all data of [MTriangle](#) in a string.

Returns

Return a string which contains every points of this shape

Implements [Shape](#).

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

- include/shape/[MTriangle.hpp](#)
- src/shape/[MTriangle.cpp](#)

5.8 Objective Class Reference

Class of the board [Objective](#).

```
#include <Objective.hpp>
```

Public Member Functions

- [Objective](#) ()
Constructor of an objective.
- `std::vector< Shape * > get_Objective ()`
Get all shape of the objective.

Static Public Member Functions

- `static bool boardCompleted (const std::vector< Shape *> &objective, const std::vector< Shape *> &game)`
Check if the board is completed.

5.8.1 Detailed Description

Class of the board [Objective](#).

This class manage everything about the objective

5.8.2 Member Function Documentation

5.8.2.1 [boardCompleted\(\)](#)

```
bool Objective::boardCompleted (
    const std::vector< Shape *> & objective,
    const std::vector< Shape *> & game ) [static]
```

Check if the board is completed.

Parameters

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

Returns

True if the board is completed, false if not

5.8.2.2 get_Objective()

```
std::vector< Shape * > Objective::get_Objective ( )
```

Get all shape of the objective.

Returns

Return a vector of shape of the objective

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

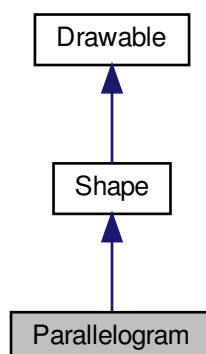
- include/game/[Objective.hpp](#)
- src/game/Objective.cpp

5.9 Parallelogram Class Reference

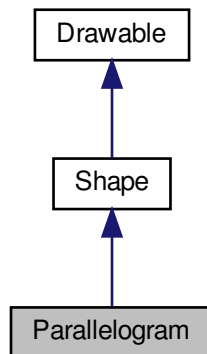
Class of the parallelogram.

```
#include <Parallelogram.hpp>
```

Inheritance diagram for Parallelogram:



Collaboration diagram for Parallelogram:



Public Member Functions

- [~Parallelogram](#) () override
Destructor of [Parallelogram](#).
- [Parallelogram](#) (MLV_Color color=MLV_COLOR_BLUE)
Constructor by default of [Parallelogram](#), make a [Parallelogram](#) as default.
- [Parallelogram](#) (const std::vector< [STriangle](#) > &triangle, MLV_Color color=MLV_COLOR_BLUE)
Constructor of [Parallelogram](#), requires a vector of [STriangles](#).
- [Parallelogram](#) (const [Point](#)< double > &origin, double angular=0.0, MLV_Color color=MLV_COLOR_BLUE)
Constructor of [Parallelogram](#), calls the delegate Default Constructor.
- void [move](#) (const [Point](#)< double > &translation) override
Move the [Parallelogram](#) by point translation.
- void [rotate](#) (double angular) override
Rotate the [Parallelogram](#) with specified angular.
- void [flip](#) () override
Flip the figure as symmetry.
- void [draw](#) () override
Draw this shape on IHM.
- bool [is_in_shape](#) (const [Point](#)< double > &click) override
Check if a point is in this shape.
- std::vector< [Point](#)< double > > [get_Points](#) () override
Get points of this shape.
- std::string [toString](#) () override
Convert all data of [Parallelogram](#) in a string.

5.9.1 Detailed Description

Class of the parallelogram.

This class manage everything about the [Parallelogram](#)

5.9.2 Constructor & Destructor Documentation

5.9.2.1 Parallelogram() [1/3]

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

Constructor by default of [Parallelogram](#), make a [Parallelogram](#) as default.

Parameters

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

5.9.2.2 Parallelogram() [2/3]

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

Constructor of [Parallelogram](#), requires a vector of STriangles.

Parameters

<i>triangle</i>	: The Parallelogram will created with a vector of STriangle (4)
<i>color</i>	: Optional parameter, color of this shape

5.9.2.3 Parallelogram() [3/3]

```
Parallelogram::Parallelogram (
    const Point< double > & origin,
    double angular = 0.0,
    MLV_Color color = MLV_COLOR_BLUE ) [explicit]
```

Constructor of [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), rotate the figure with an angular
<i>color</i>	: Optional parameter, color of this shape

5.9.3 Member Function Documentation

5.9.3.1 get_Points()

```
std::vector< Point< double > > Parallelogram::get_Points ( ) [override], [virtual]
```

Get points of this shape.

Returns

Return a vector of points of this shape

Implements [Shape](#).

5.9.3.2 is_in_shape()

```
bool Parallelogram::is_in_shape (
    const Point< double > & click ) [override], [virtual]
```

Check if a point is in this shape.

Parameters

<i>click</i>	: Point to check
--------------	----------------------------------

Returns

true if click is in this shape, false if not

Implements [Shape](#).

5.9.3.3 move()

```
void Parallelogram::move (
    const Point< double > & translation ) [override], [virtual]
```

Move the [Parallelogram](#) by point translation.

Parameters

<i>translation</i>	: Every points of this shape will be translate by this parameter
--------------------	--

Implements [Shape](#).

5.9.3.4 rotate()

```
void Parallelogram::rotate (
    double angular ) [override], [virtual]
```

Rotate the [Parallelogram](#) with specified angular.

Parameters

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

Implements [Shape](#).

5.9.3.5 toString()

```
std::string Parallelogram::toString ( ) [override], [virtual]
```

Convert all data of [Parallelogram](#) in a string.

Returns

Return a string which contains every points of this shape

Implements [Shape](#).

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

- include/shape/[Parallelogram.hpp](#)
- src/shape/Parallelogram.cpp

5.10 Point< T > Class Template Reference

Class of a [Point](#).

```
#include <Point.hpp>
```

Public Member Functions

- [Point](#) ()
Constructor for a point with initialisation list.
- [Point](#) (const T [x](#), const T [y](#))
Constructor for a point. Requires a X and a Y coordinate.
- [Point](#) & [operator=](#) (const [Point](#)< T > p)
Operator = of a point.
- bool [operator==](#) (const [Point](#)< T > p) const
Operator == of a point.
- bool [operator!=](#) (const [Point](#)< T > p) const
Operator != of a point.
- bool [operator<](#) (const [Point](#)< T > p) const
Operator < of a point.
- bool [operator>](#) (const [Point](#)< T > p) const
Operator > of a point.

Public Attributes

- T [x](#)
- T [y](#)

5.10.1 Detailed Description

```
template<typename T>
class Point< T >
```

Class of a [Point](#).

Template Parameters

T	: Template parameter This class manage everything about a point
-------------------	---

5.10.2 Constructor & Destructor Documentation

5.10.2.1 Point()

```
template<typename T>
Point< T >::Point (
    const T x,
    const T y ) [inline]
```

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

Parameters

<i>x</i>	: Template X coordinate
<i>y</i>	: Template Y coordinate

5.10.3 Member Function Documentation**5.10.3.1 operator!=()**

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

Operator != of a point.

Parameters

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

Returns

Return True if the point is different, false if not

5.10.3.2 operator<()

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

Operator < of a point.

Parameters

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

Returns

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

5.10.3.3 operator=()

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

Operator = of a point.

Parameters

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

Returns

Return a reference to a point

5.10.3.4 operator==()

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

Operator == of a point.

Parameters

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

Returns

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

5.10.3.5 operator>()

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

Operator > of a point.

Parameters

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

Returns

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

5.10.4 Member Data Documentation**5.10.4.1 x**

```
template<typename T>  
T Point< T >::x
```

Template x for a point

5.10.4.2 y

```
template<typename T>  
T Point< T >::y
```

Template y for a point

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

- include/utlis/[Point.hpp](#)

5.11 Save Class Reference

Class of the main Saver.

```
#include <Save.hpp>
```

5.11.1 Detailed Description

Class of the main Saver.

This class manage everything about the save

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

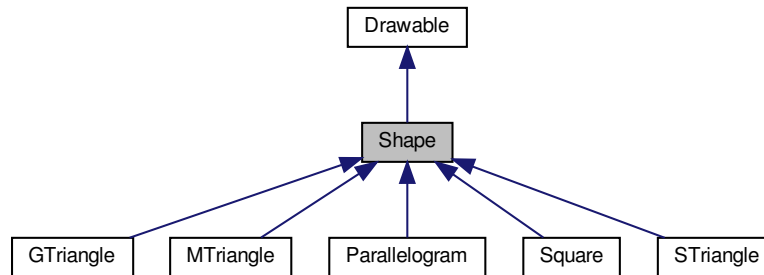
- include/parser/[Save.hpp](#)

5.12 Shape Class Reference

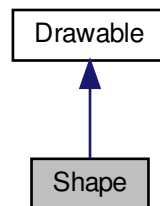
Abstract Class of every [Shape](#).

```
#include <Shape.hpp>
```

Inheritance diagram for Shape:



Collaboration diagram for Shape:



Public Member Functions

- virtual `~Shape()`=0
Destructor of Abstract [Shape](#).
- virtual void `move` (const [Point](#)< double > &translation)=0
Pure virtual function. Move the [Shape](#) by point translation.
- virtual void `rotate` (double angular)=0
Pure virtual function. Rotate the [GTriangle](#) with specified angular.
- virtual void `flip` ()=0
Pure virtual function. Flip the figure as symmetry.
- virtual bool `is_in_shape` (const [Point](#)< double > &point)=0
Pure virtual function. Check if a point is in this shape.
- virtual std::vector< [Point](#)< double > > `get_Points` ()=0
Pure virtual function. Get all points of this shape.
- virtual std::string `toString` ()=0
Pure virtual function. Convert all data of [GTriangle](#) in a string.

5.12.1 Detailed Description

Abstract Class of every [Shape](#).

This class manage everything other shape ([STriangle](#), [MTriangle](#), [GTriangle](#), [Square](#), [Parallelogram](#))

5.12.2 Member Function Documentation

5.12.2.1 `get_Points()`

```
virtual std::vector<Point<double> > Shape::get_Points ( ) [pure virtual]
```

Pure virtual function. Get all points of this shape.

Returns

Return a vector of points of this shape

Implemented in [STriangle](#), [GTriangle](#), [MTriangle](#), [Parallelogram](#), and [Square](#).

5.12.2.2 `is_in_shape()`

```
virtual bool Shape::is_in_shape (
    const Point< double > & point ) [pure virtual]
```

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

Parameters

<i>point</i>	: Point to check
--------------	----------------------------------

Returns

true if click is in this shape, false if not

Implemented in [STriangle](#), [GTriangle](#), [MTriangle](#), [Parallelogram](#), and [Square](#).

5.12.2.3 `move()`

```
virtual void Shape::move (
    const Point< double > & translation ) [pure virtual]
```

Pure virtual function. Move the [Shape](#) by point translation.

Parameters

<i>translation</i>	: Every points of this shape will be translate by this parameter
--------------------	--

Implemented in [STriangle](#), [GTriangle](#), [MTriangle](#), [Parallelogram](#), and [Square](#).

5.12.2.4 rotate()

```
virtual void Shape::rotate (
    double angular ) [pure virtual]
```

Pure virtual function. Rotate the [GTriangle](#) with specified angular.

Parameters

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

Implemented in [GTriangle](#), [MTriangle](#), [Parallelogram](#), and [Square](#).

5.12.2.5 toString()

```
virtual std::string Shape::toString ( ) [pure virtual]
```

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

Returns

Return a string which contains every points of this shape

Implemented in [STriangle](#), [GTriangle](#), [MTriangle](#), [Parallelogram](#), and [Square](#).

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

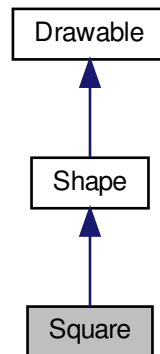
- include/drawable/[Shape.hpp](#)
- src/drawable/[Shape.cpp](#)

5.13 Square Class Reference

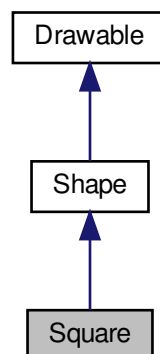
Class of the square.

```
#include <Square.hpp>
```

Inheritance diagram for Square:



Collaboration diagram for Square:



Public Member Functions

- [~Square](#) () override
Destructor of [Square](#).
- [Square](#) (MLV_Color color=MLV_COLOR_PINK)

- Constructor by default of [Square](#), make a [Square](#) as default.*
- [Square](#) (const std::vector< [STriangle](#) > &triangle, MLV_Color color=MLV_COLOR_PINK)
Constructor of [Square](#), requires a vector of STriangles.
- [Square](#) (const [Point](#)< double > &origin, double angular=0.0, MLV_Color color=MLV_COLOR_PINK)
Constructor of [Square](#), calls the delegate Default Constructor.
- void [move](#) (const [Point](#)< double > &translation) override
Move the [Square](#) by point translation.
- void [rotate](#) (double angular) override
Rotate the [Square](#) with specified angular.
- void [flip](#) () override
Flip the figure as symmetry.
- void [draw](#) () override
Draw this shape on IHM.
- bool [is_in_shape](#) (const [Point](#)< double > &click) override
Check if a point is in this shape.
- std::vector< [Point](#)< double > > [get_Points](#) () override
Get points of this shape.
- std::string [toString](#) () override
Convert all data of [Square](#) in a string.

5.13.1 Detailed Description

Class of the square.

This class manage everything about the [Square](#)

5.13.2 Constructor & Destructor Documentation

5.13.2.1 [Square\(\)](#) [1/3]

```
Square::Square (
    MLV_Color color = MLV_COLOR_PINK ) [explicit]
```

Constructor by default of [Square](#), make a [Square](#) as default.

Parameters

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

5.13.2.2 [Square\(\)](#) [2/3]

```
Square::Square (
    const std::vector< STriangle > & triangle,
    MLV_Color color = MLV_COLOR_PINK ) [explicit]
```

Constructor of [Square](#), requires a vector of [STriangles](#).

Parameters

<i>triangle</i>	: The Square will created with a vector of STriangle (4)
<i>color</i>	: Optional parameter, color of this shape

5.13.2.3 Square() [3/3]

```
Square::Square (
    const Point< double > & origin,
    double angular = 0.0,
    MLV_Color color = MLV_COLOR_PINK ) [explicit]
```

Constructor of [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), rotate the figure with an angular
<i>color</i>	: Optional parameter, color of this shape

5.13.3 Member Function Documentation

5.13.3.1 get_Points()

```
std::vector< Point< double > > Square::get_Points ( ) [override], [virtual]
```

Get points of this shape.

Returns

Return a vector of points of this shape

Implements [Shape](#).

5.13.3.2 is_in_shape()

```
bool Square::is_in_shape (
    const Point< double > & click ) [override], [virtual]
```

Check if a point is in this shape.

Parameters

<i>click</i>	: Point to check
--------------	----------------------------------

Returns

true if click is in this shape, false if not

Implements [Shape](#).

5.13.3.3 move()

```
void Square::move (
    const Point< double > & translation ) [override], [virtual]
```

Move the [Square](#) by point translation.

Parameters

<i>translation</i>	: Every points of this shape will be translate by this parameter
--------------------	--

Implements [Shape](#).

5.13.3.4 rotate()

```
void Square::rotate (
    double angular ) [override], [virtual]
```

Rotate the [Square](#) with specified angular.

Parameters

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

Implements [Shape](#).

5.13.3.5 toString()

```
std::string Square::toString ( ) [override], [virtual]
```

Convert all data of [Square](#) in a string.

Returns

Return a string which contains every points of this shape

Implements [Shape](#).

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

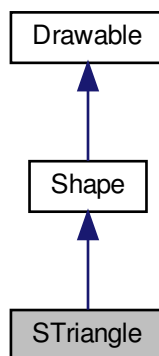
- include/shape/[Square.hpp](#)
- src/shape/Square.cpp

5.14 STriangle Class Reference

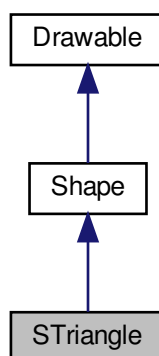
Class of the small triangle.

```
#include <STriangle.hpp>
```

Inheritance diagram for STriangle:



Collaboration diagram for STriangle:



Public Member Functions

- `~STriangle ()` override
Destructor of [STriangle](#).
- `STriangle (MLV_Color color=MLV_COLOR_GREEN)`
Constructor by default of [MTriangle](#), make a [STriangle](#) as default.
- `STriangle (const Point< double > &p1, const Point< double > &p2, const Point< double > &p3, MLV_Color color=MLV_COLOR_GREEN)`
Constructor of [STriangle](#), requires 3 points.
- `STriangle (const std::vector< Point< double > > &points, MLV_Color color=MLV_COLOR_GREEN)`
Constructor of [STriangle](#), requires a vector of 3 points.
- `STriangle (const Point< double > &origin, double angular=0.0, MLV_Color color=MLV_COLOR_GREEN)`
Constructor of [STriangle](#), calls the delegate Default Constructor.
- `void move (const Point< double > &translation)` override
Move the [MTriangle](#) by point translation.
- `void rotate (double angular, const Point< double > ¢er_point)`
Rotate an [STriangle](#) with specified angular, used only for an other shape.
- `void flip ()` override
Flip the figure as symmetry.
- `void draw ()` override
Draw this shape on IHM.
- `void draw (MLV_Color Color)`
Draw this shape on IHM with specific color.
- `bool is_in_shape (const Point< double > &click)` override
Check if a point is in this shape.
- `bool is_in_triangle (const Point< double > &click)`
Check if a point is in this [STriangle](#).
- `std::string toString ()` override
Convert all data of [MTriangle](#) in a string.
- `double computeDistance (const Point< double > &point1, const Point< double > &point2)`
Compute distance between 2 points.
- `std::vector< Point< double > > get_Points ()` override
Get every points of this [STriangle](#).
- `Point< double > get_center_point ()`
Get the current center point of this [STriangle](#).

Static Public Member Functions

- `static Point< double > center_point (const std::vector< Point< double > > &list_points)`
Compute the center point of N points.

5.14.1 Detailed Description

Class of the small triangle.

This class manage everything about the small triangle

5.14.2 Constructor & Destructor Documentation

5.14.2.1 STriangle() [1/4]

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

Constructor by default of [MTriangle](#), make a [STriangle](#) as default.

Parameters

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

5.14.2.2 STriangle() [2/4]

```
STriangle::STriangle (
    const Point< double > & p1,
    const Point< double > & p2,
    const Point< double > & p3,
    MLV_Color color = MLV_COLOR_GREEN )
```

Constructor of [STriangle](#), requires 3 points.

Parameters

<i>p1</i>	: First point of the STriangle
<i>p2</i>	: Second point of the STriangle
<i>p3</i>	: Third point of the STriangle
<i>color</i>	: Optional parameter, color of this shape

5.14.2.3 STriangle() [3/4]

```
STriangle::STriangle (
    const std::vector< Point< double >> & points,
    MLV_Color color = MLV_COLOR_GREEN ) [explicit]
```

Constructor of [STriangle](#), requires a vector of 3 points.

Parameters

<i>points</i>	: vector of 3 points
<i>color</i>	: Optional parameter, color of this shape

5.14.2.4 STriangle() [4/4]

```
STriangle::STriangle (
    const Point< double > & origin,
    double angular = 0.0,
    MLV_Color color = MLV_COLOR_GREEN ) [explicit]
```

Constructor of [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), rotate the figure with an angular
<i>color</i>	: Optional parameter, color of this shape

5.14.3 Member Function Documentation

5.14.3.1 center_point()

```
Point< double > STriangle::center_point (
    const std::vector< Point< double >> & list_points ) [static]
```

Compute the center point of N points.

Parameters

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

Returns

Return the center point of these N points

5.14.3.2 computeDistance()

```
double STriangle::computeDistance (
    const Point< double > & point1,
    const Point< double > & point2 )
```

Compute distance between 2 points.

Parameters

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

Returns

Return the distance between these two points

5.14.3.3 draw()

```
void STriangle::draw (
    MLV_Color Color )
```

Draw this shape on IHM with specific color.

Parameters

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

5.14.3.4 get_center_point()

```
Point< double > STriangle::get_center_point ( )
```

Get the current center point of this [STriangle](#).

Returns

Return the current center point of this [STriangle](#)

5.14.3.5 get_Points()

```
std::vector< Point< double > > STriangle::get_Points ( ) [override], [virtual]
```

Get every points of this [STriangle](#).

Returns

Return a vector of these points

Implements [Shape](#).

5.14.3.6 is_in_shape()

```
bool STriangle::is_in_shape (
    const Point< double > & click ) [override], [virtual]
```

Check if a point is in this shape.

Parameters

<i>click</i>	: Point to check
--------------	----------------------------------

Returns

true if click is in this shape, false if not

Implements [Shape](#).

5.14.3.7 is_in_triangle()

```
bool STriangle::is_in_triangle (
    const Point< double > & click )
```

Check if a point is in this [STriangle](#).

Parameters

<i>click</i>	: Point to check
--------------	----------------------------------

Returns

true if click is in this shape, false if not

5.14.3.8 move()

```
void STriangle::move (
    const Point< double > & translation ) [override], [virtual]
```

Move the [MTriangle](#) by point translation.

Parameters

<i>translation</i>	: Every points of this shape will be translate by this parameter
--------------------	--

Implements [Shape](#).

5.14.3.9 rotate()

```
void STriangle::rotate (
    double angular,
    const Point< double > & center_point )
```

Rotate an [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 STriangle around this point

5.14.3.10 toString()

```
std::string STriangle::toString ( ) [override], [virtual]
```

Convert all data of [MTriangle](#) in a string.

Returns

Return a string which contains every points of this shape

Implements [Shape](#).

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

- include/shape/[STriangle.hpp](#)
- src/shape/[STriangle.cpp](#)

Chapter 6

File Documentation

6.1 include/drawable/Button.hpp File Reference

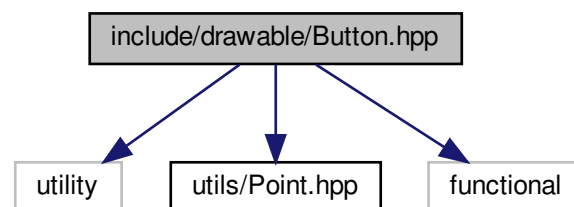
Every buttons of menu.

```
#include <utility>
```

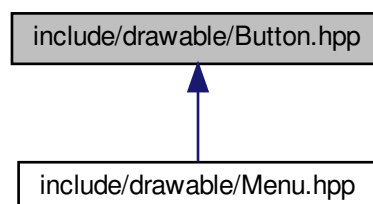
```
#include <utils/Point.hpp>
```

```
#include <functional>
```

Include dependency graph for Button.hpp:



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



Classes

- class [Button](#)
Button of the [Menu](#).

6.1.1 Detailed Description

Every buttons of menu.

Author

J  r  mie LE BASTARD

Version

1.0

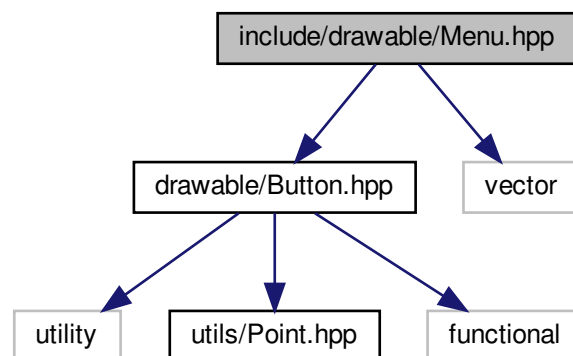
6.2 include/drawable/Menu.hpp File Reference

[Menu](#) of the Tangram's [Game](#).

```
#include <drawable/Button.hpp>
```

```
#include <vector>
```

Include dependency graph for Menu.hpp:



Classes

- class [Menu](#)
Menu of the game.

6.2.1 Detailed Description

[Menu](#) of the Tangram's [Game](#).

Author

J  r  mie LE BASTARD

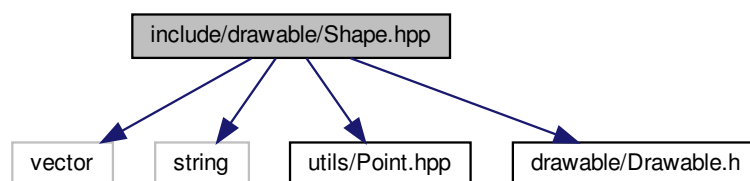
Version

1.0

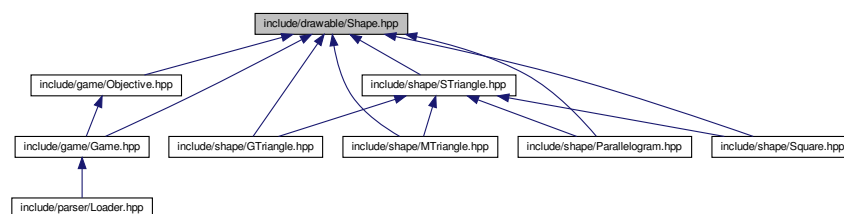
6.3 include/drawable/Shape.hpp File Reference

Abstract Class [Shape](#) of every shape in Tangram.

```
#include <vector>
#include <string>
#include <utils/Point.hpp>
#include <drawable/Drawable.h>
Include dependency graph for Shape.hpp:
```



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



Classes

- class [Shape](#)
Abstract Class of every [Shape](#).

6.3.1 Detailed Description

Abstract Class [Shape](#) of every shape in Tangram.

Author

J  r  mie LE BASTARD

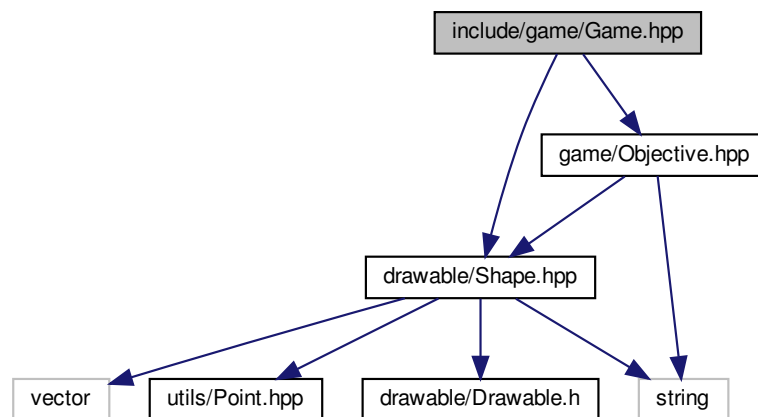
Version

1.0

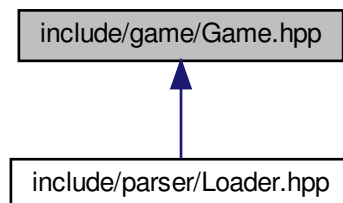
6.4 include/game/Game.hpp File Reference

Main [Game](#) of the Tangram.

```
#include <game/Objective.hpp>
#include <drawable/Shape.hpp>
Include dependency graph for Game.hpp:
```



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



Classes

- class [Game](#)

Class of the main [Game](#).

6.4.1 Detailed Description

Main [Game](#) of the Tangram.

Author

J  r  mie LE BASTARD

Version

1.0

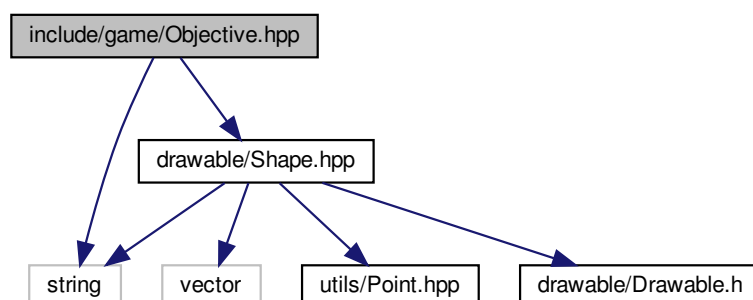
6.5 include/game/Objective.hpp File Reference

[Objective](#) of the Tangram's board.

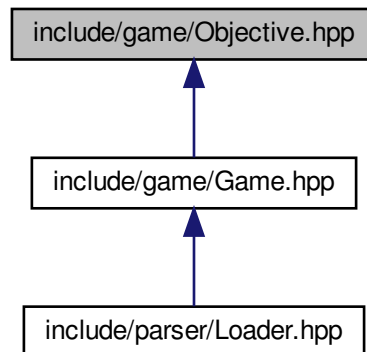
```
#include <drawable/Shape.hpp>
```

```
#include <string>
```

Include dependency graph for Objective.hpp:



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



Classes

- class [Objective](#)

Class of the board [Objective](#).

6.5.1 Detailed Description

[Objective](#) of the Tangram's board.

Author

J  r  mie LE BASTARD

Version

1.0

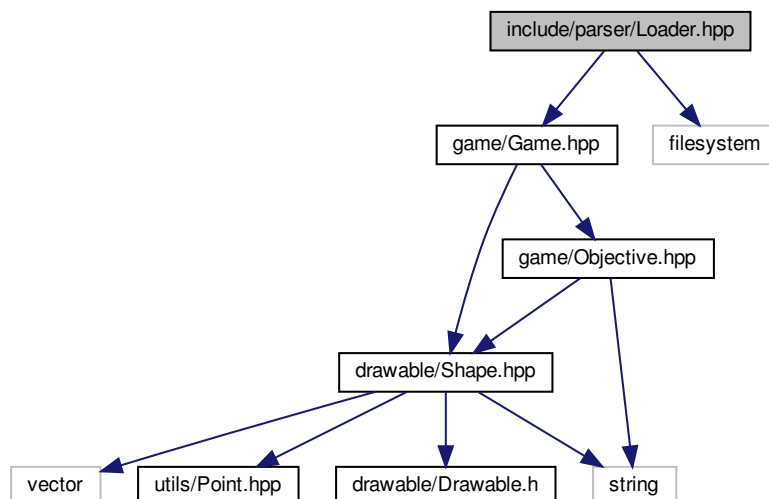
6.6 include/parser/Loader.hpp File Reference

Load a board of Tangram.

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

```
#include <filesystem>
```

Include dependency graph for Loader.hpp:



Classes

- class [Loader](#)
Class of the main [Loader](#).

6.6.1 Detailed Description

Load a board of Tangram.

Author

Jérémie LE BASTARD

Version

1.0

6.7 include/parser/Save.hpp File Reference

[Save](#) a board of Tangram.

Classes

- class [Save](#)
Class of the main Saver.

6.7.1 Detailed Description

[Save](#) a board of Tangram.

Author

J  r  mie LE BASTARD

Version

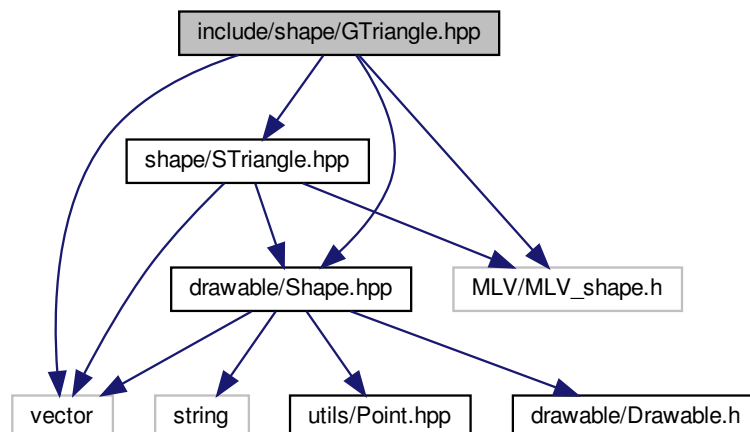
1.0

6.8 include/shape/GTriangle.hpp File Reference

[Shape](#) of Great Triangle.

```
#include <vector>
#include <shape/STriangle.hpp>
#include <drawable/Shape.hpp>
#include <MLV/MLV_shape.h>
```

Include dependency graph for GTriangle.hpp:



Classes

- class [GTriangle](#)

Class of the greatest triangle.

6.8.1 Detailed Description

[Shape](#) of Great Triangle.

Author

J  r  mie LE BASTARD

Version

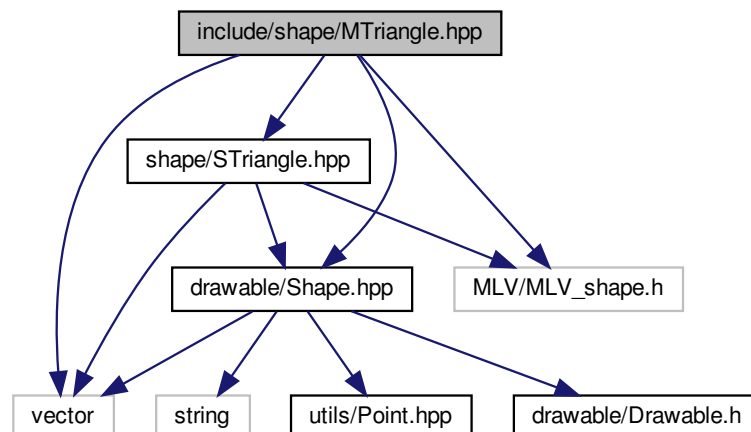
1.0

6.9 include/shape/MTriangle.hpp File Reference

[Shape](#) of Medium Triangle.

```
#include <vector>
#include <shape/STriangle.hpp>
#include <drawable/Shape.hpp>
#include <MLV/MLV_shape.h>
```

Include dependency graph for MTriangle.hpp:



Classes

- class [MTriangle](#)

Class of the medium triangle.

6.9.1 Detailed Description

[Shape](#) of Medium Triangle.

Author

J  r  mie LE BASTARD

Version

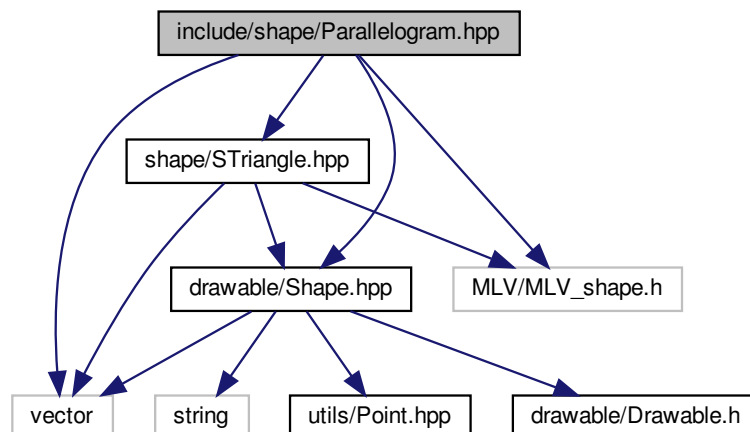
1.0

6.10 include/shape/Parallelogram.hpp File Reference

[Shape](#) of [Parallelogram](#).

```
#include <vector>
#include <shape/STriangle.hpp>
#include <drawable/Shape.hpp>
#include <MLV/MLV_shape.h>
```

Include dependency graph for Parallelogram.hpp:



Classes

- class [Parallelogram](#)

Class of the parallelogram.

6.10.1 Detailed Description

Shape of [Parallelogram](#).

Author

J  r  mie LE BASTARD

Version

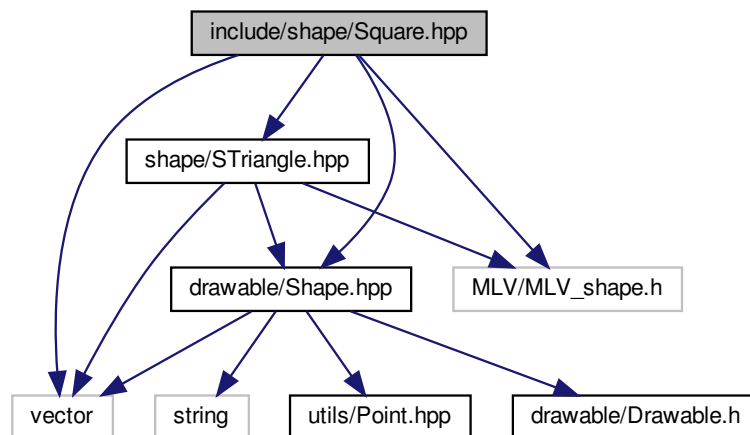
1.0

6.11 include/shape/Square.hpp File Reference

Shape of [Square](#).

```
#include <vector>
#include <shape/STriangle.hpp>
#include <drawable/Shape.hpp>
#include <MLV/MLV_shape.h>
```

Include dependency graph for Square.hpp:



Classes

- class [Square](#)

Class of the square.

6.11.1 Detailed Description

Shape of [Square](#).

Author

J  r  mie LE BASTARD

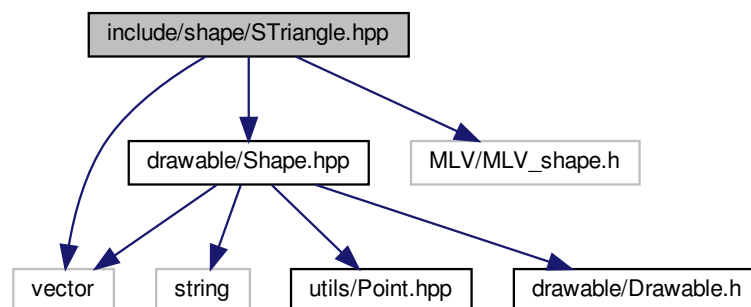
Version

1.0

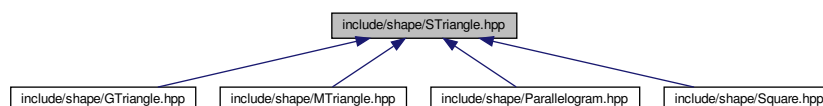
6.12 include/shape/STriangle.hpp File Reference

Shape of Small Triangle.

```
#include <vector>
#include <drawable/Shape.hpp>
#include <MLV/MLV_shape.h>
Include dependency graph for STriangle.hpp:
```



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



Classes

- class [STriangle](#)

Class of the small triangle.

6.12.1 Detailed Description

[Shape](#) of Small Triangle.

Author

Jérémie LE BASTARD

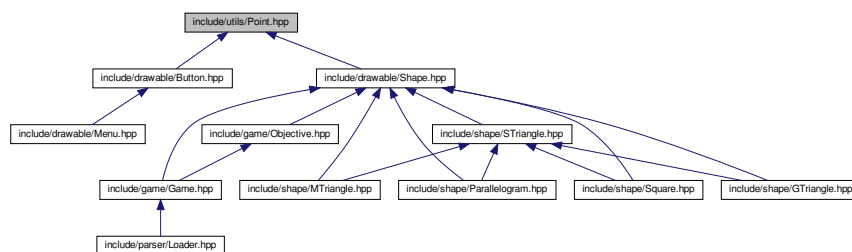
Version

1.0

6.13 include/utils/Point.hpp File Reference

[Point](#) for every shape and menu.

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



Classes

- class [Point< T >](#)
Class of a [Point](#).

6.13.1 Detailed Description

[Point](#) for every shape and menu.

Author

Jérémie LE BASTARD

Version

1.0

Index

- add_button
 - Menu, [19](#)
- add_shape
 - Game, [13](#)
- boardCompleted
 - Objective, [25](#)
- Button, [9](#)
 - Button, [9](#), [10](#)
 - click, [10](#)
 - click_in_button, [10](#)
 - set_callback, [11](#)
- center_point
 - STriangle, [45](#)
- click
 - Button, [10](#)
- click_in_button
 - Button, [10](#)
- computeDistance
 - STriangle, [45](#)
- draw
 - STriangle, [46](#)
- Drawable, [11](#)
- GTriangle, [14](#)
 - GTriangle, [15](#), [16](#)
 - get_Points, [16](#)
 - is_in_shape, [16](#)
 - move, [17](#)
 - rotate, [17](#)
 - toString, [17](#)
- Game, [12](#)
 - add_shape, [13](#)
 - Game, [13](#)
- get_Objective
 - Objective, [26](#)
- get_Points
 - GTriangle, [16](#)
 - MTriangle, [22](#)
 - Parallelogram, [29](#)
 - STriangle, [46](#)
 - Shape, [36](#)
 - Square, [40](#)
- get_center_point
 - STriangle, [46](#)
- include/drawable/Button.hpp, [49](#)
- include/drawable/Menu.hpp, [50](#)
- include/drawable/Shape.hpp, [51](#)
- include/game/Game.hpp, [52](#)
- include/game/Objective.hpp, [53](#)
- include/parser/Loader.hpp, [54](#)
- include/parser/Save.hpp, [55](#)
- include/shape/GTriangle.hpp, [56](#)
- include/shape/MTriangle.hpp, [57](#)
- include/shape/Parallelogram.hpp, [58](#)
- include/shape/STriangle.hpp, [60](#)
- include/shape/Square.hpp, [59](#)
- include/utils/Point.hpp, [61](#)
- is_in_shape
 - GTriangle, [16](#)
 - MTriangle, [22](#)
 - Parallelogram, [29](#)
 - STriangle, [46](#)
 - Shape, [36](#)
 - Square, [40](#)
- is_in_triangle
 - STriangle, [47](#)
- Loader, [18](#)
 - parse_file, [18](#)
- MTriangle, [20](#)
 - get_Points, [22](#)
 - is_in_shape, [22](#)
 - MTriangle, [21](#), [22](#)
 - move, [24](#)
 - rotate, [24](#)
 - toString, [24](#)
- Menu, [19](#)
 - add_button, [19](#)
- move
 - GTriangle, [17](#)
 - MTriangle, [24](#)
 - Parallelogram, [29](#)
 - STriangle, [47](#)
 - Shape, [36](#)
 - Square, [41](#)
- Objective, [25](#)
 - boardCompleted, [25](#)
 - get_Objective, [26](#)
- operator!=
 - Point, [32](#)
- operator<
 - Point, [32](#)
- operator>
 - Point, [33](#)
- operator=

- Point, [32](#)
- operator==
 - Point, [33](#)
- Parallelogram, [26](#)
 - get_Points, [29](#)
 - is_in_shape, [29](#)
 - move, [29](#)
 - Parallelogram, [28](#)
 - rotate, [30](#)
 - toString, [30](#)
- parse_file
 - Loader, [18](#)
- Point
 - operator!=, [32](#)
 - operator<, [32](#)
 - operator>, [33](#)
 - operator=, [32](#)
 - operator==, [33](#)
 - Point, [31](#)
 - x, [34](#)
 - y, [34](#)
- Point< T >, [30](#)
- rotate
 - GTriangle, [17](#)
 - MTriangle, [24](#)
 - Parallelogram, [30](#)
 - STriangle, [47](#)
 - Shape, [37](#)
 - Square, [41](#)
- STriangle, [42](#)
 - center_point, [45](#)
 - computeDistance, [45](#)
 - draw, [46](#)
 - get_Points, [46](#)
 - get_center_point, [46](#)
 - is_in_shape, [46](#)
 - is_in_triangle, [47](#)
 - move, [47](#)
 - rotate, [47](#)
 - STriangle, [44, 45](#)
 - toString, [48](#)
- Save, [34](#)
- set_callback
 - Button, [11](#)
- Shape, [35](#)
 - get_Points, [36](#)
 - is_in_shape, [36](#)
 - move, [36](#)
 - rotate, [37](#)
 - toString, [37](#)
- Square, [38](#)
 - get_Points, [40](#)
 - is_in_shape, [40](#)
 - move, [41](#)
 - rotate, [41](#)
 - Square, [39, 40](#)
 - toString, [41](#)
- toString
 - GTriangle, [17](#)
 - MTriangle, [24](#)
 - Parallelogram, [30](#)
 - STriangle, [48](#)
 - Shape, [37](#)
 - Square, [41](#)
- x
 - Point, [34](#)
- y
 - Point, [34](#)