

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
5.4	GTriangle Class Reference	13
5.4.1	Detailed Description	15
5.4.2	Constructor & Destructor Documentation	15
5.4.2.1	GTriangle() [1/2]	15
5.4.2.2	GTriangle() [2/2]	15
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()	16
5.4.3.4	rotate()	17
5.4.3.5	toString()	17
5.5	Loader Class Reference	17
5.6	Menu Class Reference	18
5.6.1	Detailed Description	18
5.6.2	Member Function Documentation	18
5.6.2.1	add_button()	18
5.7	MTriangle Class Reference	18
5.7.1	Detailed Description	20
5.7.2	Constructor & Destructor Documentation	20
5.7.2.1	MTriangle() [1/2]	20
5.7.2.2	MTriangle() [2/2]	20
5.7.3	Member Function Documentation	21
5.7.3.1	get_Points()	21
5.7.3.2	is_in_shape()	21
5.7.3.3	move()	21
5.7.3.4	rotate()	22
5.7.3.5	toString()	22
5.8	Objective Class Reference	22

5.8.1	Detailed Description	23
5.8.2	Member Function Documentation	23
5.8.2.1	boardCompleted()	23
5.9	Parallelogram Class Reference	23
5.9.1	Detailed Description	25
5.9.2	Constructor & Destructor Documentation	25
5.9.2.1	Parallelogram() [1/2]	25
5.9.2.2	Parallelogram() [2/2]	25
5.9.3	Member Function Documentation	26
5.9.3.1	get_Points()	26
5.9.3.2	is_in_shape()	26
5.9.3.3	move()	26
5.9.3.4	rotate()	27
5.9.3.5	toString()	27
5.10	Point< T > Class Template Reference	27
5.10.1	Detailed Description	28
5.10.2	Constructor & Destructor Documentation	28
5.10.2.1	Point()	28
5.10.3	Member Function Documentation	29
5.10.3.1	operator!=(())	29
5.10.3.2	operator<()	29
5.10.3.3	operator=()	30
5.10.3.4	operator==(())	30
5.10.3.5	operator>()	30
5.10.4	Member Data Documentation	31
5.10.4.1	x	31
5.10.4.2	y	31
5.11	Save Class Reference	31
5.12	Shape Class Reference	31
5.12.1	Detailed Description	33

5.12.2	Member Function Documentation	33
5.12.2.1	get_Points()	33
5.12.2.2	is_in_shape()	33
5.12.2.3	move()	33
5.12.2.4	rotate()	34
5.12.2.5	toString()	34
5.13	Square Class Reference	35
5.13.1	Detailed Description	36
5.13.2	Constructor & Destructor Documentation	36
5.13.2.1	Square() [1/2]	36
5.13.2.2	Square() [2/2]	36
5.13.3	Member Function Documentation	37
5.13.3.1	get_Points()	37
5.13.3.2	is_in_shape()	37
5.13.3.3	move()	38
5.13.3.4	rotate()	39
5.13.3.5	toString()	39
5.14	STriangle Class Reference	40
5.14.1	Detailed Description	41
5.14.2	Constructor & Destructor Documentation	41
5.14.2.1	STriangle() [1/3]	41
5.14.2.2	STriangle() [2/3]	42
5.14.2.3	STriangle() [3/3]	42
5.14.3	Member Function Documentation	42
5.14.3.1	center_point()	42
5.14.3.2	computeDistance()	43
5.14.3.3	draw()	43
5.14.3.4	get_center_point()	43
5.14.3.5	get_Points()	44
5.14.3.6	is_in_shape()	44
5.14.3.7	is_in_triangle()	44
5.14.3.8	move()	45
5.14.3.9	rotate()	45
5.14.3.10	toString()	45

6 File Documentation	47
6.1 include/drawable/Button.hpp File Reference	47
6.1.1 Detailed Description	48
6.2 include/drawable/Menu.hpp File Reference	48
6.2.1 Detailed Description	49
6.3 include/drawable/Shape.hpp File Reference	49
6.3.1 Detailed Description	50
6.4 include/game/Game.hpp File Reference	50
6.4.1 Detailed Description	51
6.5 include/game/Objective.hpp File Reference	51
6.5.1 Detailed Description	52
6.6 include/shape/GTriangle.hpp File Reference	52
6.6.1 Detailed Description	53
6.7 include/shape/MTriangle.hpp File Reference	53
6.7.1 Detailed Description	54
6.8 include/shape/Parallelogram.hpp File Reference	54
6.8.1 Detailed Description	55
6.9 include/shape/Square.hpp File Reference	55
6.9.1 Detailed Description	56
6.10 include/utils/Point.hpp File Reference	56
6.10.1 Detailed Description	56
Index	57

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	31
GTriangle	13
MTriangle	18
Parallelogram	23
Square	35
STriangle	40
Game	12
Loader	17
Menu	18
Objective	22
Point< T >	27
Point< double >	27
Point< int >	27
Save	31

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	13
Loader		
Loader	17
Menu		
Menu	Menu of the game	18
MTriangle		
MTriangle	Class of the medium triangle	18
Objective		
Objective	Class of the board Objective	22
Parallelogram		
Parallelogram	Class of the parallelogram	23
Point< T >		
Point< T >	Class of a Point	27
Save		
Save	31
Shape		
Shape	Abstract Class of every Shape	31
Square		
Square	Class of the square	35
STriangle		
STriangle	Class of the small triangle	40

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	47
include/drawable/ Drawable.h	??
include/drawable/ Menu.hpp	
Menu of the Tangram's Game	48
include/drawable/ Shape.hpp	
Abstract Class Shape of every shape in Tangram	49
include/game/ Game.hpp	
Main Game of the Tangram	50
include/game/ Objective.hpp	
Objective of the Tangram's board	51
include/parser/ Loader.hpp	??
include/parser/ Save.hpp	??
include/shape/ GTriangle.hpp	
Shape of Great Triangle	52
include/shape/ MTriangle.hpp	
Shape of Medium Triangle	53
include/shape/ Parallelogram.hpp	
Shape of Parallelogram	54
include/shape/ Square.hpp	
Shape of Square	55
include/shape/ STriangle.hpp	??
include/utlis/ Point.hpp	
Point for every shape and menu	56

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:

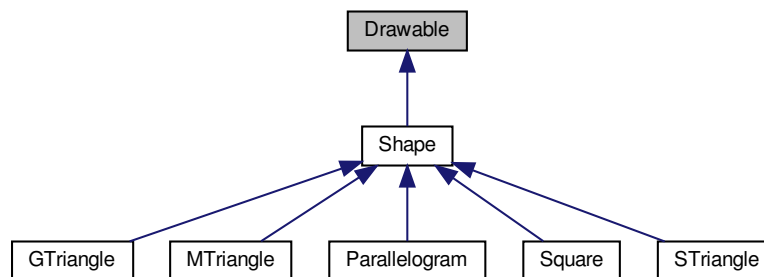
- include/drawable/[Button.hpp](#)
- src/drawable/Button.cpp

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)
Initialize the game.

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 )
```

Initialize the game.

Parameters

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

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

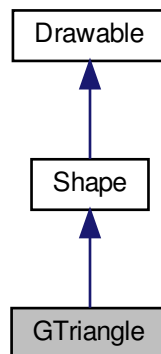
- include/game/[Game.hpp](#)
- src/game/Game.cpp

5.4 GTriangle Class Reference

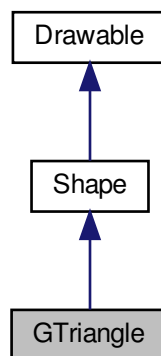
Class of the greatest triangle.

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

Inheritance diagram for GTriangle:



Collaboration diagram for GTriangle:



Public Member Functions

- `~GTriangle ()` override
Destructor of `GTriangle`.
- `GTriangle ()`
Constructor by default of `GTriangle`, make a triangle as default.
- `GTriangle (const std::vector< STriangle > &triangle)`
Constructor of `GTriangle`, requires a vector of triangles.
- `GTriangle (Point< double > origin, double angular=0.0)`
Constructor of `GTriangle`, calls the delegate Default Constructor.
- void `move (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](#) ([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() [1/2]

```
GTriangle::GTriangle (
    const std::vector< STriangle > & triangle ) [explicit]
```

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

Parameters

<i>triangle</i>	: The GTriangle will created with a vector of STriangle (4)
-----------------	---

5.4.2.2 GTriangle() [2/2]

```
GTriangle::GTriangle (
    Point< double > origin,
    double angular = 0.0 ) [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

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 (
    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 (
    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

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

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

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:

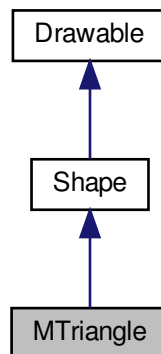
- include/drawable/[Menu.hpp](#)
- src/drawable/Menu.cpp

5.7 MTriangle Class Reference

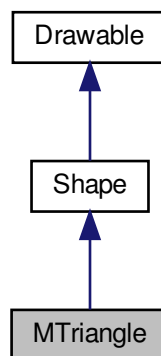
Class of the medium triangle.

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

Inheritance diagram for MTriangle:



Collaboration diagram for MTriangle:



Public Member Functions

- `~MTriangle ()` override
Destructor of `MTriangle`.
- `MTriangle ()`
Constructor by default of `MTriangle`, make a `MTriangle` as default.
- `MTriangle (const std::vector< STriangle > &triangle)`
Constructor of `MTriangle`, requires a vector of `STriangles`.
- `MTriangle (Point< double > origin, double angular=0.0)`
Constructor of `MTriangle`, calls the delegate Default Constructor.
- `void move (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* (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() [1/2]

```
MTriangle::MTriangle (
    const std::vector< STriangle > & triangle ) [explicit]
```

Constructor of *MTriangle*, requires a vector of *STriangles*.

Parameters

<i>triangle</i>	: The <i>MTriangle</i> will created with a vector of <i>STriangle</i> (4)
-----------------	---

5.7.2.2 MTriangle() [2/2]

```
MTriangle::MTriangle (
    Point< double > origin,
    double angular = 0.0 ) [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

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 (
    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 (
    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

- bool [boardCompleted](#) (std::vector< [Shape](#) *> objective, 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 (
    std::vector< Shape *> objective,
    std::vector< Shape *> game )
```

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

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

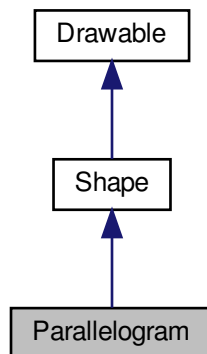
- include/game/[Objective.hpp](#)
- src/game/Objective.cpp

5.9 Parallelogram Class Reference

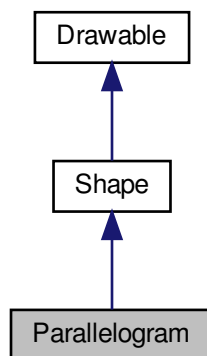
Class of the parallelogram.

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

Inheritance diagram for Parallelogram:



Collaboration diagram for Parallelogram:



Public Member Functions

- [~Parallelogram](#) () override
Destructor of [Parallelogram](#).
- [Parallelogram](#) ()
Constructor by default of [Parallelogram](#), make a [Parallelogram](#) as default.
- [Parallelogram](#) (const std::vector< [STriangle](#) > &triangle)
Constructor of [Parallelogram](#), requires a vector of [STriangles](#).
- [Parallelogram](#) ([Point](#)< double > origin, double angular=0.0)
Constructor of [Parallelogram](#), calls the delegate Default Constructor.
- void [move](#) ([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](#) ([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](#)() [1/2]

```
Parallelogram::Parallelogram (
    const std::vector< STriangle > & triangle ) [explicit]
```

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

Parameters

<i>triangle</i>	: The Parallelogram will created with a vector of STriangle (4)
-----------------	---

5.9.2.2 [Parallelogram](#)() [2/2]

```
Parallelogram::Parallelogram (
    Point< double > origin,
    double angular = 0.0 ) [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

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 (
    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 (
    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

p	: 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

p	: 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

p	: Point to comapre
-----	------------------------------------

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/utills/[Point.hpp](#)

5.11 Save Class Reference

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

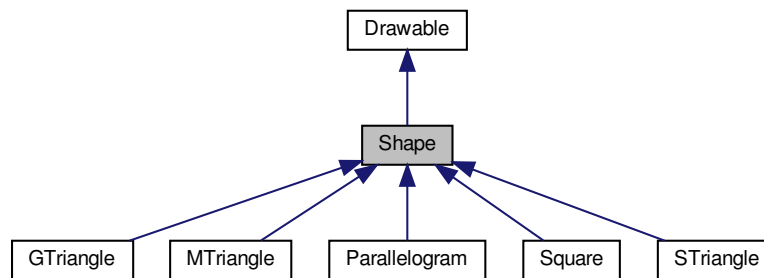
- include/parser/Save.hpp

5.12 Shape Class Reference

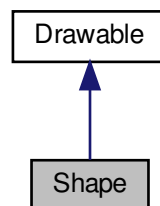
Abstract Class of every [Shape](#).

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

Inheritance diagram for Shape:



Collaboration diagram for Shape:



Public Member Functions

- virtual `~Shape()`=0
Destructor of Abstract [Shape](#).
- virtual void `move(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(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 (
    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 (
    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:

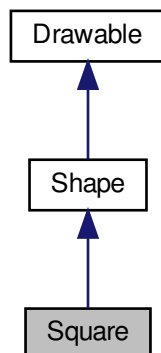
- [include/drawable/Shape.hpp](#)
- [src/drawable/Shape.cpp](#)

5.13 Square Class Reference

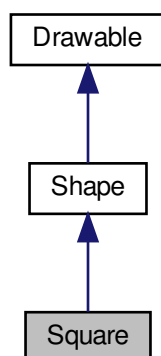
Class of the square.

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

Inheritance diagram for Square:



Collaboration diagram for Square:



Public Member Functions

- [~Square](#) () override
Destructor of [Square](#).
- [Square](#) ()

- Constructor by default of [Square](#), make a [Square](#) as default.
- [Square](#) (const std::vector< [STriangle](#) > &triangle)

Constructor of [Square](#), requires a vector of STriangles.
- [Square](#) ([Point](#)< double > origin, double angular=0.0)

Constructor of [Square](#), calls the delegate Default Constructor.
- void [move](#) ([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](#) ([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\(\)](#) [1/2]

```
Square::Square (
    const std::vector< STriangle > & triangle ) [explicit]
```

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

Parameters

<i>triangle</i>	: The Square will created with a vector of STriangle (4)
-----------------	--

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

```
Square::Square (
    Point< double > origin,
    double angular = 0.0 ) [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

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 (
    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 (
    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:

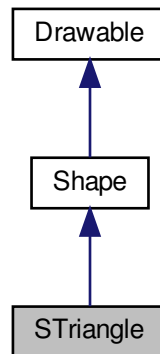
- [include/shape/Square.hpp](#)
- [src/shape/Square.cpp](#)

5.14 STriangle Class Reference

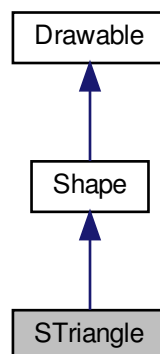
Class of the small triangle.

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

Inheritance diagram for STriangle:



Collaboration diagram for STriangle:



Public Member Functions

- [~STriangle](#) () override
Destructor of [STriangle](#).
- [STriangle](#) ()

- Constructor by default of *MTriangle*, make a *STriangle* as default.

 - *STriangle* (*Point*< double > p1, *Point*< double > p2, *Point*< double > p3)

Constructor of *STriangle*, requires 3 points.
- *STriangle* (const std::vector< *Point*< double >> &points)

Constructor of *STriangle*, requires a vector of 3 points.
- *STriangle* (*Point*< double > origin, double angular=0.0)

Constructor of *STriangle*, calls the delegate Default Constructor.
- void *move* (*Point*< double > translation) override

Move the *MTriangle* by point translation.
- void *rotate* (double angular, *Point*< double > center_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* (*Point*< double > click) override

Check if a point is in this shape.
- bool *is_in_triangle* (*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* (*Point*< double > point1, *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() [1/3]

```
STriangle::STriangle (
    Point< double > p1,
    Point< double > p2,
    Point< double > p3 )
```

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

5.14.2.2 [STriangle\(\)](#) [2/3]

```
STriangle::STriangle (
    const std::vector< Point< double >> & points ) [explicit]
```

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

Parameters

<i>points</i>	: vector of 3 points
---------------	----------------------

5.14.2.3 [STriangle\(\)](#) [3/3]

```
STriangle::STriangle (
    Point< double > origin,
    double angular = 0.0 ) [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

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 (
    Point< double > point1,
    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 (
    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 (
    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 (
    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,
    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

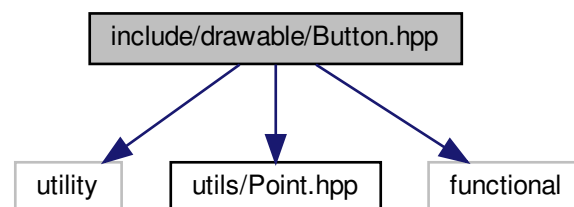
Every buttons of menu.

```
#include <utility>
```

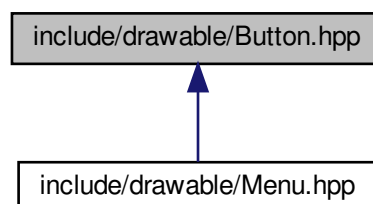
```
#include <utils/Point.hpp>
```

```
#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

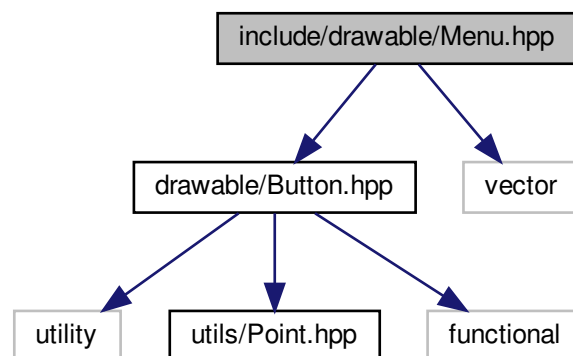
6.2 include/drawable/Menu.hpp File Reference

[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

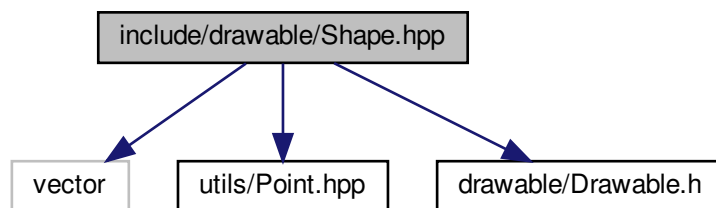
Version

1.0

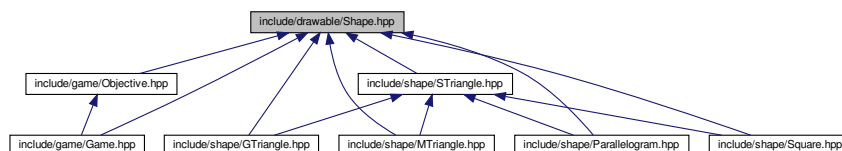
6.3 include/drawable/Shape.hpp File Reference

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

```
#include <vector>
#include <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

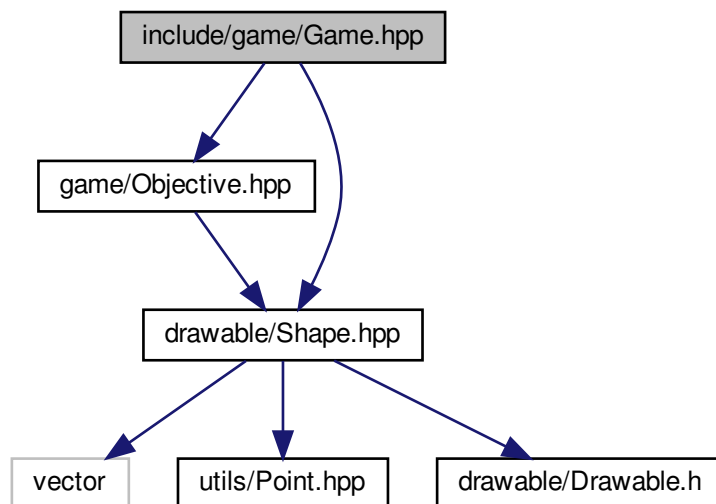
6.4 include/game/Game.hpp File Reference

Main [Game](#) of the Tangram.

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

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

Include dependency graph for Game.hpp:



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

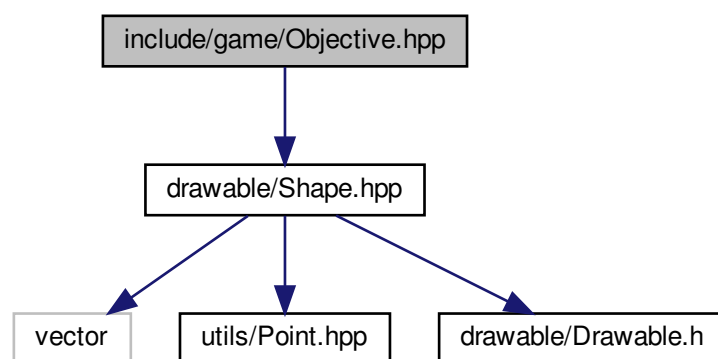
1.0

6.5 include/game/Objective.hpp File Reference

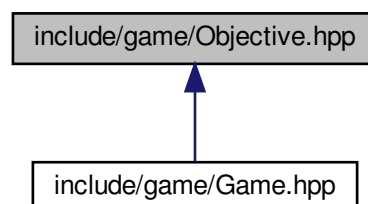
[Objective](#) of the Tangram's board.

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

Include 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

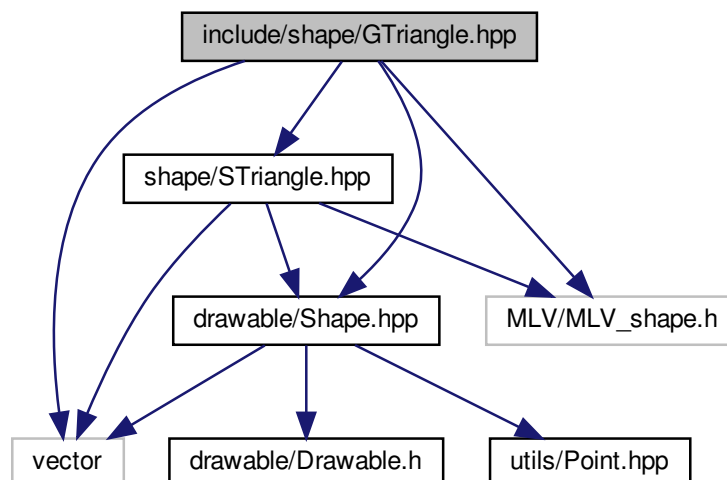
1.0

6.6 include/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.

6.6.1 Detailed Description

[Shape](#) of Great Triangle.

Author

Jérémie LE BASTARD

Version

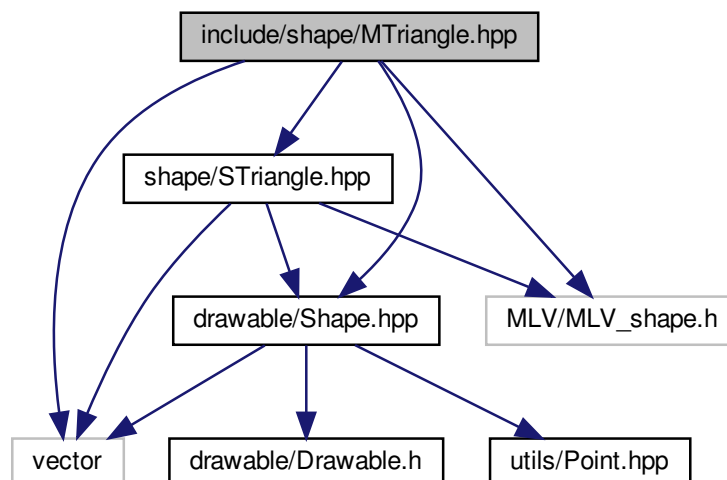
1.0

6.7 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.

6.7.1 Detailed Description

[Shape](#) of Medium Triangle.

[Shape](#) of Small Triangle.

Author

Jérémie LE BASTARD

Version

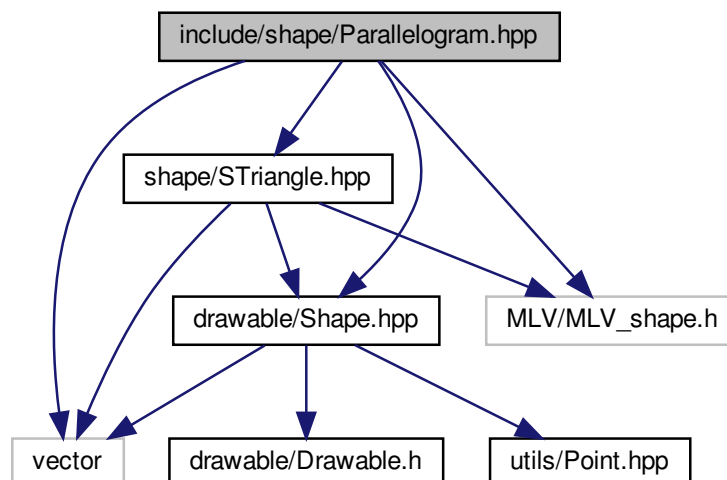
1.0

6.8 `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.

6.8.1 Detailed Description

Shape of [Parallelogram](#).

Author

Jérémie LE BASTARD

Version

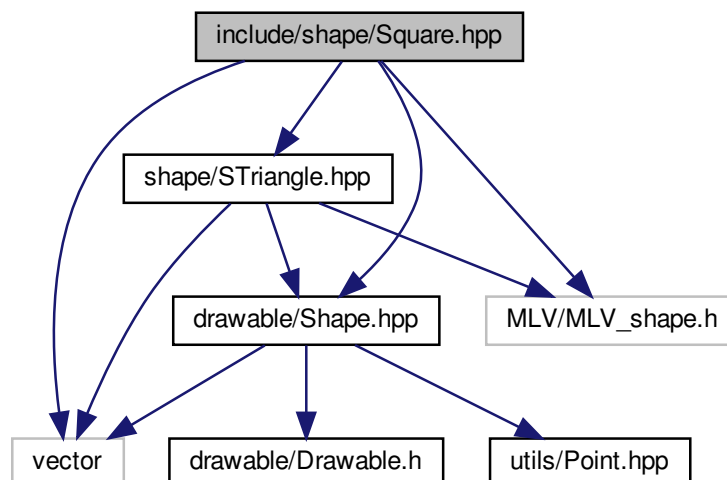
1.0

6.9 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.

6.9.1 Detailed Description

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

Author

Jérémie LE BASTARD

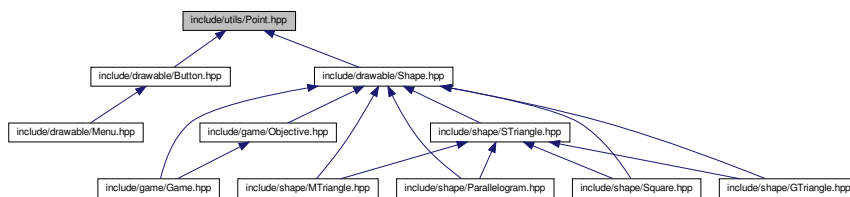
Version

1.0

6.10 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](#).

6.10.1 Detailed Description

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

Author

Jérémie LE BASTARD

Version

1.0

Index

- add_button
 - Menu, [18](#)
- boardCompleted
 - Objective, [23](#)
- Button, [9](#)
 - Button, [9](#), [10](#)
 - click, [10](#)
 - click_in_button, [10](#)
 - set_callback, [11](#)
- center_point
 - STriangle, [42](#)
- click
 - Button, [10](#)
- click_in_button
 - Button, [10](#)
- computeDistance
 - STriangle, [43](#)
- draw
 - STriangle, [43](#)
- Drawable, [11](#)
- GTriangle, [13](#)
 - GTriangle, [15](#)
 - get_Points, [16](#)
 - is_in_shape, [16](#)
 - move, [16](#)
 - rotate, [17](#)
 - toString, [17](#)
- Game, [12](#)
 - Game, [13](#)
- get_Points
 - GTriangle, [16](#)
 - MTriangle, [21](#)
 - Parallelogram, [26](#)
 - STriangle, [44](#)
 - Shape, [33](#)
 - Square, [37](#)
- get_center_point
 - STriangle, [43](#)
- include/drawable/Button.hpp, [47](#)
- include/drawable/Menu.hpp, [48](#)
- include/drawable/Shape.hpp, [49](#)
- include/game/Game.hpp, [50](#)
- include/game/Objective.hpp, [51](#)
- include/shape/GTriangle.hpp, [52](#)
- include/shape/MTriangle.hpp, [53](#)
- include/shape/Parallelogram.hpp, [54](#)
- include/shape/Square.hpp, [55](#)
- include/utils/Point.hpp, [56](#)
- is_in_shape
 - GTriangle, [16](#)
 - MTriangle, [21](#)
 - Parallelogram, [26](#)
 - STriangle, [44](#)
 - Shape, [33](#)
 - Square, [37](#)
- is_in_triangle
 - STriangle, [44](#)
- Loader, [17](#)
- MTriangle, [18](#)
 - get_Points, [21](#)
 - is_in_shape, [21](#)
 - MTriangle, [20](#)
 - move, [21](#)
 - rotate, [22](#)
 - toString, [22](#)
- Menu, [18](#)
 - add_button, [18](#)
- move
 - GTriangle, [16](#)
 - MTriangle, [21](#)
 - Parallelogram, [26](#)
 - STriangle, [45](#)
 - Shape, [33](#)
 - Square, [37](#)
- Objective, [22](#)
 - boardCompleted, [23](#)
- operator!=
 - Point, [29](#)
- operator<
 - Point, [29](#)
- operator>
 - Point, [30](#)
- operator=
 - Point, [29](#)
- operator==
 - Point, [30](#)
- Parallelogram, [23](#)
 - get_Points, [26](#)
 - is_in_shape, [26](#)
 - move, [26](#)
 - Parallelogram, [25](#)
 - rotate, [27](#)

- toString, [27](#)
- Point
 - operator!=, [29](#)
 - operator<, [29](#)
 - operator>, [30](#)
 - operator=, [29](#)
 - operator==, [30](#)
 - Point, [28](#)
 - x, [31](#)
 - y, [31](#)
- Point< T >, [27](#)
- rotate
 - GTriangle, [17](#)
 - MTriangle, [22](#)
 - Parallelogram, [27](#)
 - STriangle, [45](#)
 - Shape, [34](#)
 - Square, [39](#)
- STriangle, [40](#)
 - center_point, [42](#)
 - computeDistance, [43](#)
 - draw, [43](#)
 - get_Points, [44](#)
 - get_center_point, [43](#)
 - is_in_shape, [44](#)
 - is_in_triangle, [44](#)
 - move, [45](#)
 - rotate, [45](#)
 - STriangle, [41](#), [42](#)
 - toString, [45](#)
- Save, [31](#)
- set_callback
 - Button, [11](#)
- Shape, [31](#)
 - get_Points, [33](#)
 - is_in_shape, [33](#)
 - move, [33](#)
 - rotate, [34](#)
 - toString, [34](#)
- Square, [35](#)
 - get_Points, [37](#)
 - is_in_shape, [37](#)
 - move, [37](#)
 - rotate, [39](#)
 - Square, [36](#)
 - toString, [39](#)
- toString
 - GTriangle, [17](#)
 - MTriangle, [22](#)
 - Parallelogram, [27](#)
 - STriangle, [45](#)
 - Shape, [34](#)
 - Square, [39](#)
- x
 - Point, [31](#)
- y
 - Point, [31](#)