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
5.3.3.2 get_Objective_Color() . . . 13
5.3.3.3 set_Objective() . . . 14
5.3.3.4 stick() . . . 14
5.4 GTriangle Class Reference . . . 14
5.4.1 Detailed Description . . . 16
5.4.2 Constructor & Destructor Documentation . . . 16
5.4.2.1 GTriangle() [1/3] . . . 16
5.4.2.2 GTriangle() [2/3] . . . 16
5.4.2.3 GTriangle() [3/3] . . . 18
5.4.3 Member Function Documentation . . . 18
5.4.3.1 get_Points() . . . 18
5.4.3.2 is_in_shape() . . . 18
5.4.3.3 move() . . . 19
5.4.3.4 rotate() . . . 19
5.4.3.5 set_Points() . . . 19
5.4.3.6 toString() . . . 20
5.5 Point< T >::hash_point Struct Reference . . . 20
5.6 Loader Class Reference . . . 20
5.6.1 Detailed Description . . . 21
5.6.2 Member Function Documentation . . . 21
5.6.2.1 parse_file() . . . 21
5.7 Menu Class Reference . . . 21
5.7.1 Detailed Description . . . 22
5.7.2 Member Function Documentation . . . 22
5.7.2.1 add_button() . . . 22
5.8 MTriangle Class Reference . . . 22

5.8.1 Detailed Description . . . 23
5.8.2 Constructor & Destructor Documentation . . . 24
5.8.2.1 MTriangle() [1/3] . . . 24
5.8.2.2 MTriangle() [2/3] . . . 24
5.8.2.3 MTriangle() [3/3] . . . 24
5.8.3 Member Function Documentation . . . 25
5.8.3.1 get_Points() . . . 25
5.8.3.2 is_in_shape() . . . 25
5.8.3.3 move() . . . 25
5.8.3.4 rotate() . . . 26
5.8.3.5 set_Points() . . . 26
5.8.3.6 toString() . . . 26
5.9 Objective Class Reference . . . 27
5.9.1 Detailed Description . . . 27
5.9.2 Constructor & Destructor Documentation . . . 27
5.9.2.1 Objective() [1/2] . . . 27
5.9.2.2 Objective() [2/2] . . . 28
5.9.3 Member Function Documentation . . . 28
5.9.3.1 boardCompleted() . . . 28
5.9.3.2 get_Color() . . . 28
5.9.3.3 get_Objective() . . . 29
5.9.3.4 set_Objective() . . . 29
5.10 Parallelogram Class Reference . . . 29
5.10.1 Detailed Description . . . 31
5.10.2 Constructor & Destructor Documentation . . . 31
5.10.2.1 Parallelogram() [1/3] . . . 31
5.10.2.2 Parallelogram() [2/3] . . . 31
5.10.2.3 Parallelogram() [3/3] . . . 33
5.10.3 Member Function Documentation . . . 33
5.10.3.1 get_Points() . . . 33

5.10.3.2 is_in_shape() . . . 33
5.10.3.3 move() . . . 34
5.10.3.4 rotate() . . . 34
5.10.3.5 set_Points() . . . 34
5.10.3.6 toString() . . . 35
5.11 Point< T > Class Template Reference . . . 35
5.11.1 Detailed Description . . . 36
5.11.2 Constructor & Destructor Documentation . . . 36
5.11.2.1 Point() . . . 36
5.11.3 Member Function Documentation . . . 36
5.11.3.1 operator"!=() . . . 36
5.11.3.2 operator<() . . . 37
5.11.3.3 operator=() . . . 37
5.11.3.4 operator==() . . . 38
5.11.3.5 operator>() . . . 38
5.11.4 Member Data Documentation . . . 38
5.11.4.1 x . . . 38
5.11.4.2 y . . . 39
5.12 Save Class Reference . . . 39
5.12.1 Detailed Description . . . 39
5.13 Shape Class Reference . . . 39
5.13.1 Detailed Description . . . 40
5.13.2 Member Function Documentation . . . 40
5.13.2.1 computeDistance() . . . 41
5.13.2.2 get_Points() . . . 41
5.13.2.3 is_in_shape() . . . 41
5.13.2.4 move() . . . 42
5.13.2.5 rotate() . . . 42
5.13.2.6 set_Points() . . . 42
5.13.2.7 toString() . . . 43

5.14 Square Class Reference . . . 43
5.14.1 Detailed Description . . . 44
5.14.2 Constructor & Destructor Documentation . . . 45
5.14.2.1 Square() [1/3] . . . 45
5.14.2.2 Square() [2/3] . . . 45
5.14.2.3 Square() [3/3] . . . 45
5.14.3 Member Function Documentation . . . 46
5.14.3.1 get_Points() . . . 46
5.14.3.2 is_in_shape() . . . 46
5.14.3.3 move() . . . 46
5.14.3.4 rotate() . . . 47
5.14.3.5 set_Points() . . . 47
5.14.3.6 toString() . . . 47
5.15 STriangle Class Reference . . . 48
5.15.1 Detailed Description . . . 49
5.15.2 Constructor & Destructor Documentation . . . 49
5.15.2.1 STriangle() [1/4] . . . 49
5.15.2.2 STriangle() [2/4] . . . 50
5.15.2.3 STriangle() [3/4] . . . 50
5.15.2.4 STriangle() [4/4] . . . 50
5.15.3 Member Function Documentation . . . 51
5.15.3.1 center_point() . . . 51
5.15.3.2 draw() . . . 51
5.15.3.3 get_center_point() . . . 51
5.15.3.4 get_Points() . . . 52
5.15.3.5 is_in_shape() . . . 52
5.15.3.6 is_in_triangle() . . . 52
5.15.3.7 move() . . . 53
5.15.3.8 rotate() . . . 53
5.15.3.9 set_Points() . . . 53
5.15.3.10 toString() . . . 54

6 File Documentation **55**

6.1 include/drawable/Button.hpp File Reference . . . 55

6.1.1 Detailed Description . . . 56

6.2 include/drawable/Menu.hpp File Reference . . . 56

6.2.1 Detailed Description . . . 57

6.3 include/drawable/Shape.hpp File Reference . . . 57

6.3.1 Detailed Description . . . 58

6.4 include/game/Game.hpp File Reference . . . 59

6.4.1 Detailed Description . . . 59

6.5 include/game/Objective.hpp File Reference . . . 60

6.5.1 Detailed Description . . . 61

6.6 include/parser/Loader.hpp File Reference . . . 61

6.6.1 Detailed Description . . . 62

6.7 include/parser/Save.hpp File Reference . . . 62

6.7.1 Detailed Description . . . 62

6.8 include/shape/GTriangle.hpp File Reference . . . 62

6.8.1 Detailed Description . . . 63

6.9 include/shape/MTriangle.hpp File Reference . . . 63

6.9.1 Detailed Description . . . 64

6.10 include/shape/Parallelogram.hpp File Reference . . . 64

6.10.1 Detailed Description . . . 65

6.11 include/shape/Square.hpp File Reference . . . 65

6.11.1 Detailed Description . . . 66

6.12 include/shape/STriangle.hpp File Reference . . . 66

6.12.1 Detailed Description . . . 67

6.13 include/utils/Point.hpp File Reference . . . 68

6.13.1 Detailed Description . . . 68

Index **69**

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 . . . 39
    GTriangle . . . 14
    MTriangle . . . 22
    Parallelogram . . . 29
    Square . . . 43
    STriangle . . . 48
Game . . . 12
Point< T >::hash_point . . . 20
Loader . . . 20
Menu . . . 21
Objective . . . 27
Point< T > . . . 35
Point< double > . . . 35
Point< int > . . . 35
Save . . . 39

Chapter 3

Class Index

3.1 Class List

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

Button
: Button of the Menu . . . 9

Drawable
: Drawable is everything to draw . . . 11

Game
: Class of the main Game . . . 12

GTriangle
: Class of the greatest triangle . . . 14

Point< T >::hash_point . . . 20

Loader
: Class of the main Loader . . . 20

Menu
: Menu of the game . . . 21

MTriangle
: Class of the medium triangle . . . 22

Objective
: Class of the board Objective . . . 27

Parallelogram
: Class of the parallelogram . . . 29

Point< T >
: Class of a Point . . . 35

Save
: Class of the main Saver . . . 39

Shape
: Abstract Class of every Shape . . . 39

Square
: Class of the square . . . 43

STriangle
: Class of the small triangle . . . 48

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 . . . 55

include/drawable/**Drawable.h** . . . **??**

include/drawable/Menu.hpp
: Menu of the Tangram's Game . . . 56

include/drawable/Shape.hpp
: Abstract Class Shape of every shape in Tangram . . . 57

include/game/Game.hpp
: Main Game of the Tangram . . . 59

include/game/Objective.hpp
: Objective of the Tangram's board . . . 60

include/parser/Loader.hpp
: Load a board of Tangram . . . 61

include/parser/Save.hpp
: Save a board of Tangram . . . 62

include/shape/GTriangle.hpp
: Shape of Great Triangle . . . 62

include/shape/MTriangle.hpp
: Shape of Medium Triangle . . . 63

include/shape/Parallelogram.hpp
: Shape of Parallelogram . . . 64

include/shape/Square.hpp
: Shape of Square . . . 65

include/shape/STriangle.hpp
: Shape of Small Triangle . . . 66

include/utils/Point.hpp
: Point for every shape and menu . . . 68

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

point	: Top left point position of the button
sizing	: Sizing of the button, (width , height)
text	: 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

point	: Top left point position of the button
sizing	: Sizing of the button, (width , height)
text	: Text of the button
callback	: 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

click	: 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

callback	: Requires a pointer of function for set the callback

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

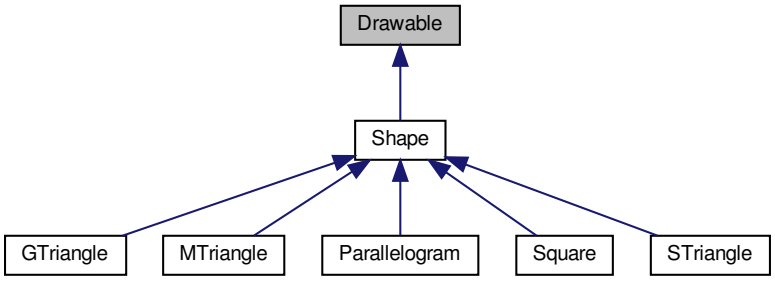
- include/drawable/Button.hpp
- src/drawable/Button.cpp

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 (std::shared_ptr< Shape > s)
  Add a shape in the game.
- void clear ()
  Clear the game / the board and the objective.
- void stick (const std::shared_ptr< Shape > &shape)
  Stick the shape to nearest objective points.
- void set_Objective (const std::vector< std::shared_ptr< Shape >> &vec_objective)
  Set the objective of the game.
- MLV_Color get_Objective_Color ()
  Get the color of the objective of 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 )
```

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

Parameters

w	: Width of the window
h	: Height of the window

5.3.3 Member Function Documentation

5.3.3.1 add_shape()

```
void Game::add_shape (
            std::shared_ptr< Shape > s )
```

Add a shape in the game.

Parameters

s	: Shape to add

5.3.3.2 get_Objective_Color()

```
MLV_Color Game::get_Objective_Color ( )
```

Get the color of the objective of the game.

Returns

Return the color of the objective of the game

5.3.3.3 set_Objective()

```
void Game::set_Objective (
            const std::vector< std::shared_ptr< Shape >> & vec_objective )
```

Set the objective of the game.

Parameters

vec_objective	: Vector of Objective for new game;

5.3.3.4 stick()

```
void Game::stick (
            const std::shared_ptr< Shape > & shape )
```

Stick the shape to nearest objective points.

Parameters

shape	: Last shape rotated or moved

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

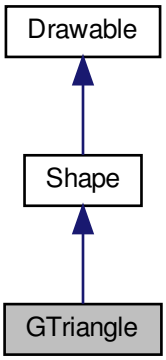
- include/game/Game.hpp
- src/game/Game.cpp

5.4 GTriangle Class Reference

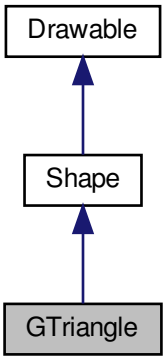
Class of the greatest triangle.

```
#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 deleguate 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.

- bool set_Points (const Point< double > &ref, const Point< double > &changed) override

  Pure virtual function. Get all points of this shape.

- std::string toString () override

  Convert all data of GTriangle in a string.

Additional Inherited Members

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

color	: 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

triangle	: The GTriangle will created with a vector of STriangle (4)
color	: 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 deleguate Default Constructor.

Parameters

origin	: shifts the figure of a translation of the origin
angular	: Optional parameter (angular=0.0 as default), rotate the figure with an angular
color	: 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

click	: 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

translation	: 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

angular	: This angular should be between (0, 2PI)

Implements Shape.

5.4.3.5 set_Points()

```
bool GTriangle::set_Points (
            const Point< double > & ref,
            const Point< double > & changed )  [override], [virtual]
```

Pure virtual function. Get all points of this shape.

Returns

Return a vector of points of this shape

Implements Shape.

5.4.3.6 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 Point< T >::hash_point Struct Reference

Public Member Functions

- std::size_t **operator()** (const Point< double > &p) const
- bool **operator()** (const Point< double > &p1, const Point< double > &p2) const

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

- include/utils/Point.hpp

5.6 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.

5.6.1 Detailed Description

Class of the main Loader.

This class manage everything about the loader

5.6.2 Member Function Documentation

5.6.2.1 parse_file()

```
bool Loader::parse_file (
            const std::string & filename,
            Game & game )  [static]
```

Parse a file to make a board.

Parameters

filename	: name of the file, this file should be located in this directory ./Tangram/extern/board/
game	: 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

5.7 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.

5.7.1 Detailed Description

Menu of the game.

This class manage everything about Tangram's menu

5.7.2 Member Function Documentation

5.7.2.1 add_button()

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

Add a button in the Menu.

Parameters

button	: Button to add

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

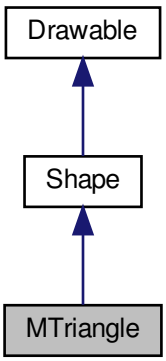
- include/drawable/Menu.hpp
- src/drawable/Menu.cpp

5.8 MTriangle Class Reference

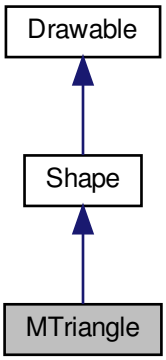
Class of the medium triangle.

`#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 deleguate 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.
- bool set_Points (const Point< double > &ref, const Point< double > &changed) override
  Pure virtual function. Get all points of this shape.
- std::string toString () override
  Convert all data of MTriangle in a string.

Additional Inherited Members

5.8.1 Detailed Description

Class of the medium triangle.

This class manage everything about the medium triangle

5.8.2 Constructor & Destructor Documentation

5.8.2.1 MTriangle() [1/3]

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

Constructor by default of MTriangle, make a MTriangle as default.

Parameters

color	: Optional parameter, color of this shape

5.8.2.2 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

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

5.8.2.3 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 deleguate Default Constructor.

Parameters

origin	: shifts the figure of a translation of the origin
angular	: Optional parameter (angular=0.0 as default), rotate the figure with an angular
color	: Optional parameter, color of this shape

5.8.3 Member Function Documentation

5.8.3.1 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.

5.8.3.2 is_in_shape()

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

Check if a point is in this shape.

Parameters

click	: Point to check

Returns

true if click is in this shape, false if not

Implements Shape.

5.8.3.3 move()

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

Move the MTriangle by point translation.

Parameters

translation	: Every points of this shape will be translate by this parameter

Implements Shape.

5.8.3.4 rotate()

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

Rotate the MTriangle with specified angular.

Parameters

angular	: This angular should be between (0, 2PI)

Implements Shape.

5.8.3.5 set_Points()

```
bool MTriangle::set_Points (
            const Point< double > & ref,
            const Point< double > & changed )  [override], [virtual]
```

Pure virtual function. Get all points of this shape.

Returns

Return a vector of points of this shape

Implements Shape.

5.8.3.6 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

5.9 Objective Class Reference

Class of the board Objective.

`#include <Objective.hpp>`

Public Member Functions

- Objective (MLV_Color color=MLV_COLOR_GRAY70)

  Constructor of an objective, default constructor.
- Objective (const std::vector< std::shared_ptr< Shape >> &objective, MLV_Color color=MLV_COLOR_G↩ RAY70)

  Constructor of an objective.
- std::vector< std::shared_ptr< Shape > > get_Objective ()

  Get all shape of the objective.
- MLV_Color get_Color ()

  Get the color of an Objective.

Static Public Member Functions

- static bool boardCompleted (const std::vector< std::shared_ptr< Shape >> &objective, const std::vector< std::shared_ptr< Shape >> &game)

  Check if the board is completed.
- static void set_Objective (Objective *objective, const std::vector< std::shared_ptr< Shape >> &vec_↩ objective)

  Set an Objective for a new game.

5.9.1 Detailed Description

Class of the board Objective.

This class manage everything about the objective

5.9.2 Constructor & Destructor Documentation

5.9.2.1 Objective() [1/2]

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

Constructor of an objective, default constructor.

Parameters

color	: color of the objective shape

5.9.2.2 Objective() [2/2]

```
Objective::Objective (
            const std::vector< std::shared_ptr< Shape >> & objective,
            MLV_Color color = MLV_COLOR_GRAY70 )  [explicit]
```

Constructor of an objective.

Parameters

objective	: Objective requires a vector of Shape
color	: color of the objective shape

5.9.3 Member Function Documentation

5.9.3.1 boardCompleted()

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

Check if the board is completed.

Parameters

objective	: Vector of objective's shape
game	: Vector of current game's shape

Returns

True if the board is completed, false if not

5.9.3.2 get_Color()

```
MLV_Color Objective::get_Color ( )
```

Get the color of an Objective.

Returns

Return the color of an Objective

5.9.3.3 get_Objective()

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

Get all shape of the objective.

Returns

Return a vector of shape of the objective

5.9.3.4 set_Objective()

```
void Objective::set_Objective (
            Objective * objective,
            const std::vector< std::shared_ptr< Shape >> & vec_objective ) [static]
```

Set an Objective for a new game.

Parameters

objective	: Objective to update
vec_objective	:Vector of new Shape for the new Objective

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

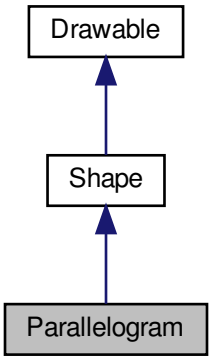
- include/game/Objective.hpp
- src/game/Objective.cpp

5.10 Parallelogram Class Reference

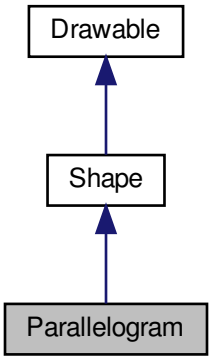
Class of the parallelogram.

```
#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 deleguate 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.

- bool set_Points (const Point< double > &ref, const Point< double > &changed) override

  Pure virtual function. Get all points of this shape.

- std::string toString () override

  Convert all data of Parallelogram in a string.

Additional Inherited Members

5.10.1 Detailed Description

Class of the parallelogram.

This class manage everything about the Parallelogram

5.10.2 Constructor & Destructor Documentation

5.10.2.1 Parallelogram() [1/3]

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

Constructor by default of Parallelogram, make a Parallelogram as default.

Parameters

color	: Optional parameter, color of this shape

5.10.2.2 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

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

5.10.2.3 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 deleguate Default Constructor.

Parameters

origin	: shifts the figure of a translation of the origin
angular	: Optional parameter (angular=0.0 as default), rotate the figure with an angular
color	: Optional parameter, color of this shape

5.10.3 Member Function Documentation

5.10.3.1 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.

5.10.3.2 is_in_shape()

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

Check if a point is in this shape.

Parameters

click	: Point to check

Returns

true if click is in this shape, false if not

Implements Shape.

5.10.3.3 move()

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

Move the Parallelogram by point translation.

Parameters

translation	: Every points of this shape will be translate by this parameter

Implements Shape.

5.10.3.4 rotate()

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

Rotate the Parallelogram with specified angular.

Parameters

angular	: This angular should be between (0, 2PI)

Implements Shape.

5.10.3.5 set_Points()

```
bool Parallelogram::set_Points (
            const Point< double > & ref,
            const Point< double > & changed )  [override], [virtual]
```

Pure virtual function. Get all points of this shape.

Returns

Return a vector of points of this shape

Implements Shape.

5.10.3.6 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

5.11 Point< T > Class Template Reference

Class of a Point.

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

Classes

- struct hash_point

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

5.11.1 Detailed Description

template<typename T>
class Point< T >

Class of a Point.

Template Parameters

T	: Template parameter This class manage everything about a point

5.11.2 Constructor & Destructor Documentation

5.11.2.1 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

↩ _↩ *x*	: Template X coordinate
↩ _↩ *y*	: Template Y coordinate

5.11.3 Member Function Documentation

5.11.3.1 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 different, false if not

5.11.3.2 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 strictly weaker, false if not

5.11.3.3 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

5.11.3.4 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

5.11.3.5 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

5.11.4 Member Data Documentation

5.11.4.1 x

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

Template x for a point

5.11.4.2 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/utils/Point.hpp

5.12 Save Class Reference

Class of the main Saver.

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

5.12.1 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:

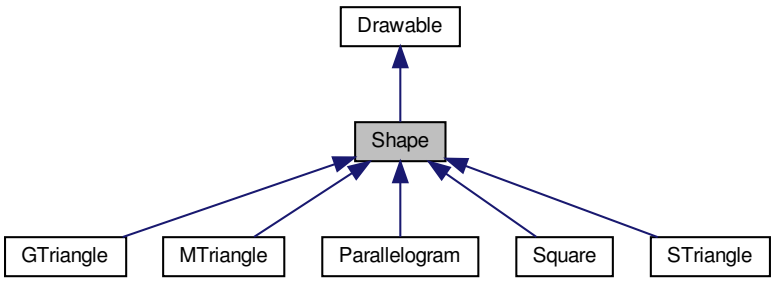
- include/parser/Save.hpp

5.13 Shape Class Reference

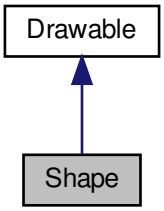
Abstract Class of every Shape.

```
#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 bool set_Points (const Point< double > &ref, const Point< double > &changed)=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.

Static Public Member Functions

- static double computeDistance (const Point< double > &point1, const Point< double > &point2)

  Compute distance between 2 points.

5.13.1 Detailed Description

Abstract Class of every Shape.

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

5.13.2 Member Function Documentation

5.13.2.1 computeDistance()

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

Compute distance between 2 points.

Parameters

point1	: First point
point2	: Second point

Returns

Return the distance between these two points

5.13.2.2 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.

5.13.2.3 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

point	: Point to check

Returns

true if click is in this shape, false if not

Implemented in STriangle, GTriangle, MTriangle, Parallelogram, and Square.

5.13.2.4 move()

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

Pure virtual function. Move the Shape by point translation.

Parameters

translation	: Every points of this shape will be translate by this parameter

Implemented in STriangle, GTriangle, MTriangle, Parallelogram, and Square.

5.13.2.5 rotate()

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

Pure virtual function. Rotate the GTriangle with specified angular.

Parameters

angular	: This angular should be between (0, 2PI)

Implemented in GTriangle, MTriangle, Parallelogram, and Square.

5.13.2.6 set_Points()

```
virtual bool Shape::set_Points (
            const Point< double > & ref,
            const Point< double > & changed )  [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.

5.13.2.7 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:

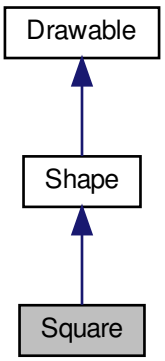
- include/drawable/Shape.hpp
- src/drawable/Shape.cpp

5.14 Square Class Reference

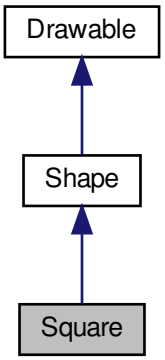
Class of the square.

```
#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 deleguate 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.
- bool set_Points (const Point< double > &ref, const Point< double > &changed) override
  Pure virtual function. Get all points of this shape.
- std::string toString () override
  Convert all data of Square in a string.

Additional Inherited Members

5.14.1 Detailed Description

Class of the square.

This class manage everything about the Square

5.14.2 Constructor & Destructor Documentation

5.14.2.1 Square() [1/3]

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

Constructor by default of Square, make a Square as default.

Parameters

color	: Optional parameter, color of this shape

5.14.2.2 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

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

5.14.2.3 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 deleguate Default Constructor.

Parameters

origin	: shifts the figure of a translation of the origin
angular	: Optional parameter (angular=0.0 as default), rotate the figure with an angular
color	: Optional parameter, color of this shape

5.14.3 Member Function Documentation

5.14.3.1 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.

5.14.3.2 is_in_shape()

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

Check if a point is in this shape.

Parameters

click	: Point to check

Returns

true if click is in this shape, false if not

Implements Shape.

5.14.3.3 move()

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

Move the Square by point translation.

Parameters

translation	: Every points of this shape will be translate by this parameter

Implements Shape.

5.14.3.4 rotate()

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

Rotate the Square with specified angular.

Parameters

angular	: This angular should be between (0, 2PI)

Implements Shape.

5.14.3.5 set_Points()

```
bool Square::set_Points (
            const Point< double > & ref,
            const Point< double > & changed )  [override], [virtual]
```

Pure virtual function. Get all points of this shape.

Returns

Return a vector of points of this shape

Implements Shape.

5.14.3.6 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:

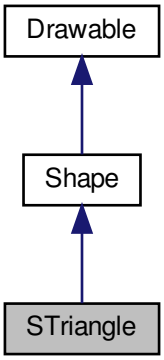
- include/shape/Square.hpp
- src/shape/Square.cpp

5.15 STriangle Class Reference

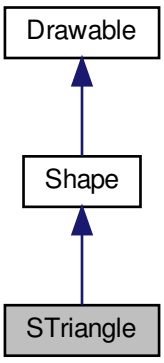
Class of the small triangle.

`#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 deleguate Default Constructor.
- void move (const Point< double > &translation) override

  Move the MTriangle by point translation.
- void rotate (double angular, const 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 (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.
- std::vector< Point< double > > get_Points () override

  Get every points of this STriangle.
- bool set_Points (const Point< double > &ref, const Point< double > &changed) override

  Pure virtual function. Get all points of this shape.
- 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.

5.15.1 Detailed Description

Class of the small triangle.

This class manage everything about the small triangle

5.15.2 Constructor & Destructor Documentation

5.15.2.1 STriangle() [1/4]

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

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

Parameters

color	: Optional parameter, color of this shape

5.15.2.2 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

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

5.15.2.3 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

points	: vector of 3 points
color	: Optional parameter, color of this shape

5.15.2.4 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 deleguate Default Constructor.

Parameters

origin	: shifts the figure of a translation of the origin
angular	: Optional parameter (angular=0.0 as default), rotate the figure with an angular
color	: Optional parameter, color of this shape

5.15.3 Member Function Documentation

5.15.3.1 center_point()

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

Compute the center point of N points.

Parameters

list_points	: vector of N points

Returns

Return the center point of these N points

5.15.3.2 draw()

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

Draw this shape on IHM with specific color.

Parameters

Color	: Color from the graphic library MLV like MLV_COLOR_XXX

5.15.3.3 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

5.15.3.4 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.

5.15.3.5 is_in_shape()

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

Check if a point is in this shape.

Parameters

click	: Point to check

Returns

true if click is in this shape, false if not

Implements Shape.

5.15.3.6 is_in_triangle()

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

Check if a point is in this STriangle.

Parameters

click	: Point to check

Returns

true if click is in this shape, false if not

5.15.3.7 move()

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

Move the MTriangle by point translation.

Parameters

translation	: Every points of this shape will be translate by this parameter

Implements Shape.

5.15.3.8 rotate()

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

Rotate an STriangle with specified angular, used only for an other shape.

Parameters

angular	: This angular should be between (0, 2PI)
center_point	: Rotate an STriangle around this point

5.15.3.9 set_Points()

```
bool STriangle::set_Points (
            const Point< double > & ref,
            const Point< double > & changed )  [override], [virtual]
```

Pure virtual function. Get all points of this shape.

Returns

Return a vector of points of this shape

Implements Shape.

5.15.3.10 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

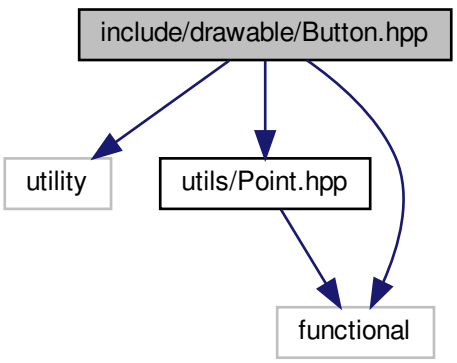
File Documentation

6.1 include/drawable/Button.hpp File Reference

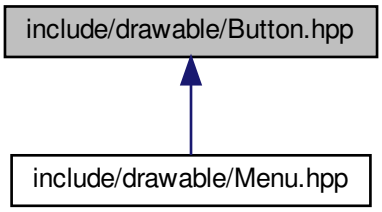
Every buttons of menu.

```
#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

Jérémie LE BASTARD

Version

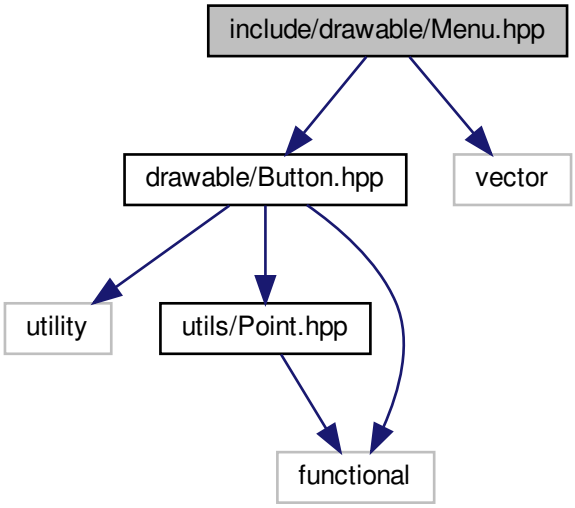
1.0

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

Jérémie LE BASTARD

Version

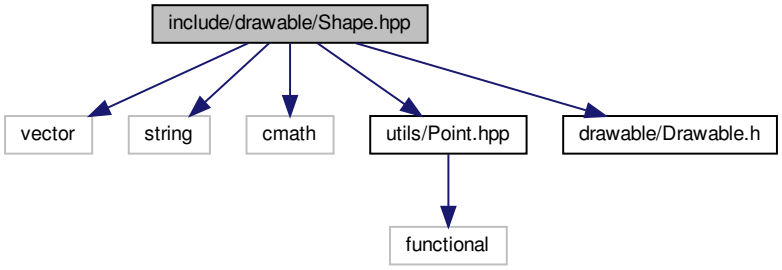
1.0

6.3 include/drawable/Shape.hpp File Reference

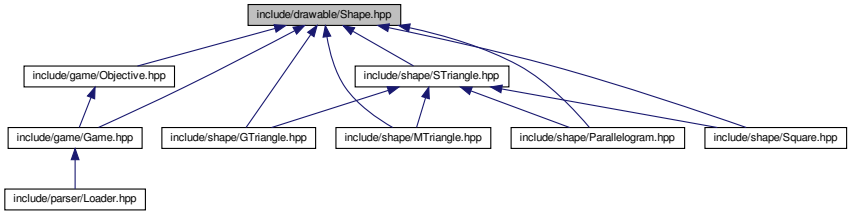
Abstract Class Shape of every shape in Tangram.

```
#include <vector>
#include <string>
#include <cmath>
#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

Jérémie LE BASTARD

Version

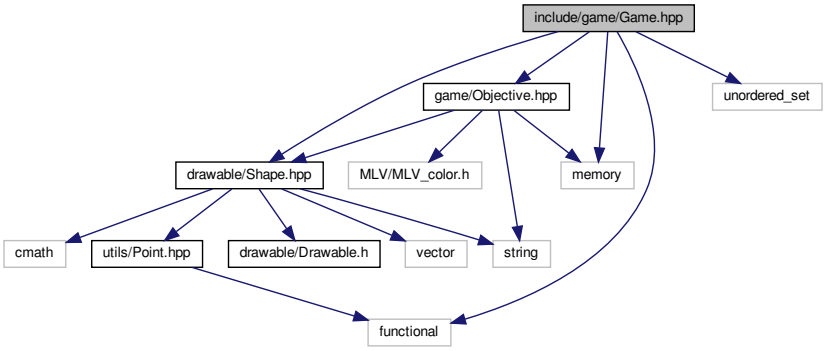
1.0

6.4 include/game/Game.hpp File Reference

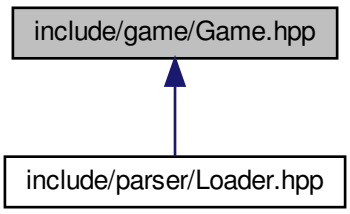
Main Game of the Tangram.

```
#include <game/Objective.hpp>
#include <drawable/Shape.hpp>
#include <functional>
#include <unordered_set>
#include <memory>
```

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

Jérémie LE BASTARD

Version

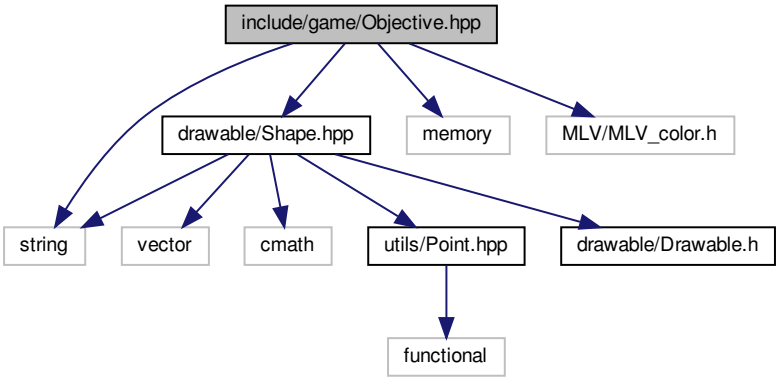
1.0

6.5 include/game/Objective.hpp File Reference

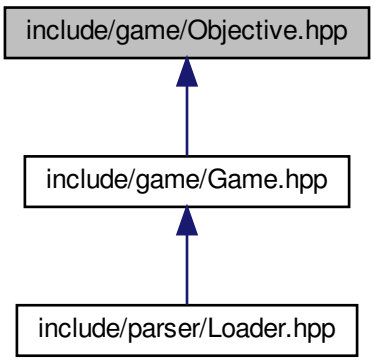
Objective of the Tangram's board.

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

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

Jérémie LE BASTARD

Version

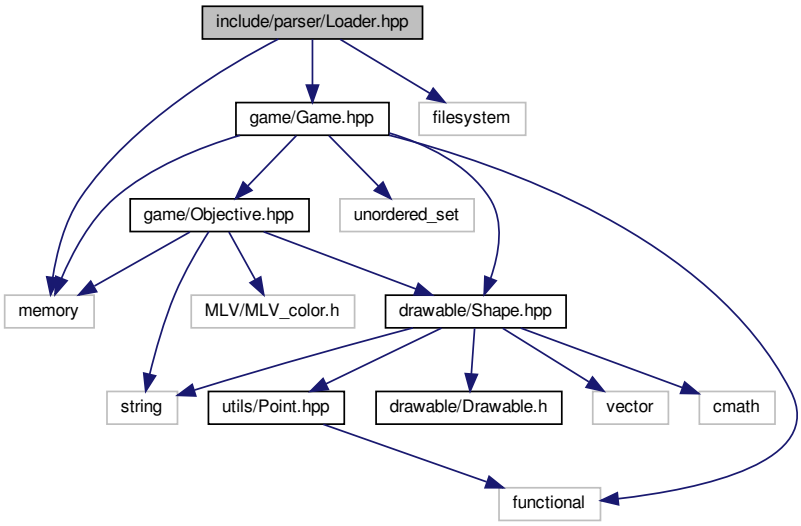
1.0

6.6 include/parser/Loader.hpp File Reference

Load a board of Tangram.

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

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

Jérémie LE BASTARD

Version

1.0

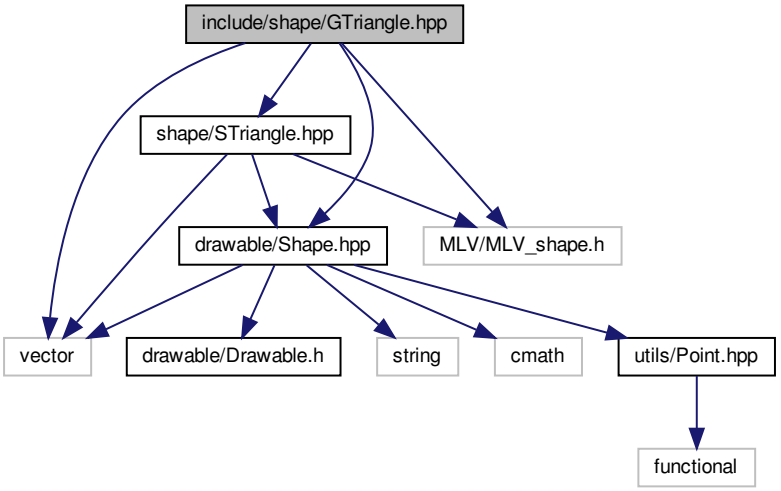
6.8 include/shape/GTriangle.hpp File Reference

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

Jérémie LE BASTARD

Version

1.0

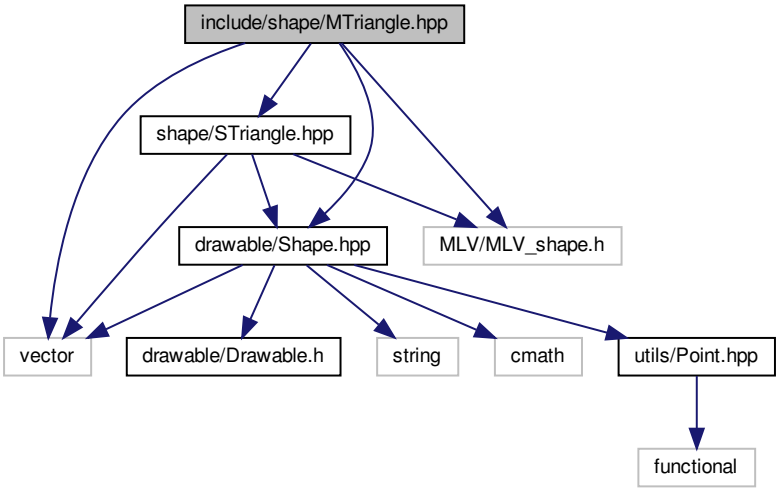
6.9 include/shape/MTriangle.hpp File Reference

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

Jérémie LE BASTARD

Version

1.0

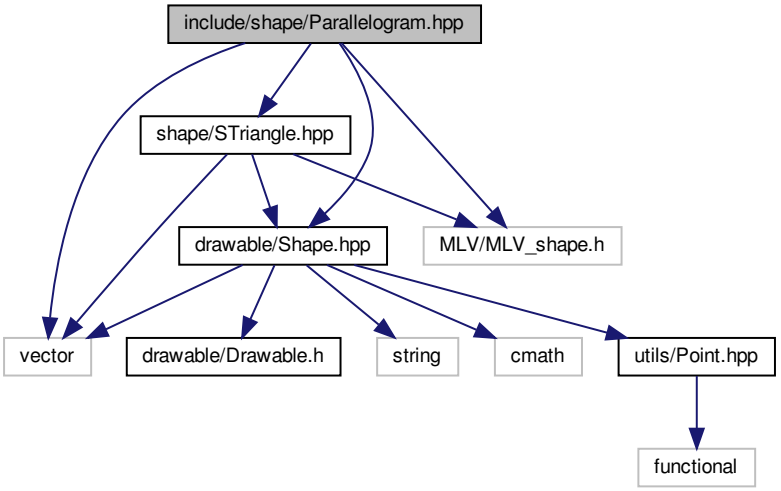
6.10 include/shape/Parallelogram.hpp File Reference

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

Jérémie LE BASTARD

Version

1.0

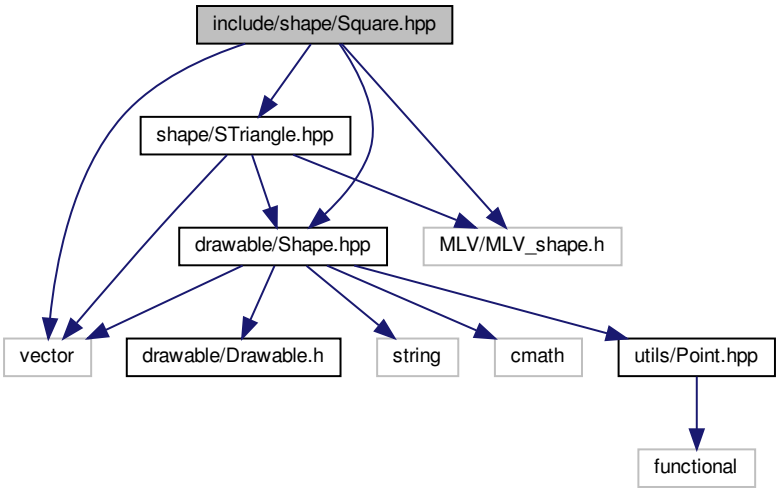
6.11 include/shape/Square.hpp File Reference

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

Jérémie LE BASTARD

Version

1.0

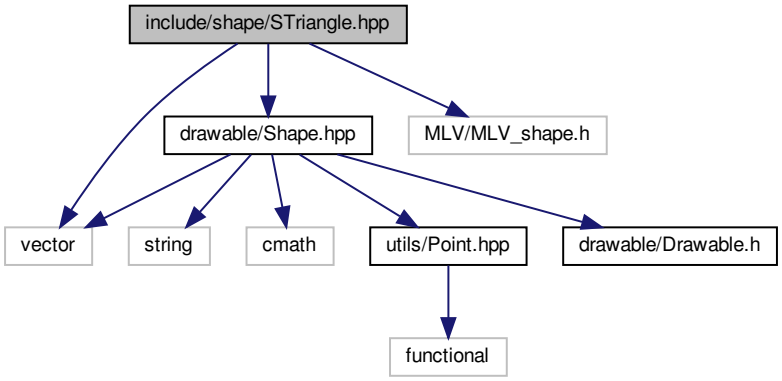
6.12 include/shape/STriangle.hpp File Reference

Shape of Small Triangle.

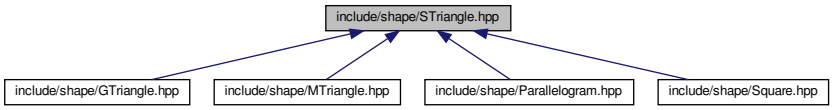
```
#include <vector>
#include <drawable/Shape.hpp>
```

```
#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

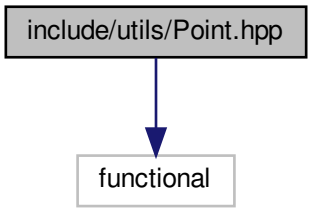
Version

1.0

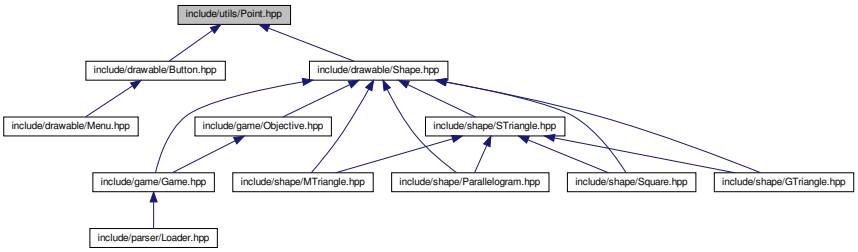
6.13 include/utils/Point.hpp File Reference

Point for every shape and menu.

`#include <functional>`
Include dependency graph for Point.hpp:



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



Classes

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

6.13.1 Detailed Description

Point for every shape and menu.

Author

Jérémie LE BASTARD

Version

1.0

Index

add_button
    Menu, 22
add_shape
    Game, 13

boardCompleted
    Objective, 28
Button, 9
    Button, 9, 10
    click, 10
    click_in_button, 10
    set_callback, 11

center_point
    STriangle, 51
click
    Button, 10
click_in_button
    Button, 10
computeDistance
    Shape, 40

draw
    STriangle, 51
Drawable, 11

GTriangle, 14
    GTriangle, 16, 18
    get_Points, 18
    is_in_shape, 18
    move, 19
    rotate, 19
    set_Points, 19
    toString, 20
Game, 12
    add_shape, 13
    Game, 13
    get_Objective_Color, 13
    set_Objective, 14
    stick, 14
get_Color
    Objective, 28
get_Objective
    Objective, 28
get_Objective_Color
    Game, 13
get_Points
    GTriangle, 18
    MTriangle, 25
    Parallelogram, 33
    STriangle, 52
    Shape, 41
    Square, 46
get_center_point
    STriangle, 51

include/drawable/Button.hpp, 55
include/drawable/Menu.hpp, 56
include/drawable/Shape.hpp, 57
include/game/Game.hpp, 59
include/game/Objective.hpp, 60
include/parser/Loader.hpp, 61
include/parser/Save.hpp, 62
include/shape/GTriangle.hpp, 62
include/shape/MTriangle.hpp, 63
include/shape/Parallelogram.hpp, 64
include/shape/STriangle.hpp, 66
include/shape/Square.hpp, 65
include/utils/Point.hpp, 68
is_in_shape
    GTriangle, 18
    MTriangle, 25
    Parallelogram, 33
    STriangle, 52
    Shape, 41
    Square, 46
is_in_triangle
    STriangle, 52

Loader, 20
    parse_file, 21

MTriangle, 22
    get_Points, 25
    is_in_shape, 25
    MTriangle, 24
    move, 25
    rotate, 26
    set_Points, 26
    toString, 26
Menu, 21
    add_button, 22
move
    GTriangle, 19
    MTriangle, 25
    Parallelogram, 34
    STriangle, 53
    Shape, 41
    Square, 46

Objective, 27

boardCompleted, 28
get_Color, 28
get_Objective, 28
Objective, 27, 28
set_Objective, 29
operator!=
Point, 36
operator<
Point, 37
operator>
Point, 38
operator=
Point, 37
operator==
Point, 37

Parallelogram, 29
get_Points, 33
is_in_shape, 33
move, 34
Parallelogram, 31, 33
rotate, 34
set_Points, 34
toString, 35
parse_file
Loader, 21
Point
operator!=, 36
operator<, 37
operator>, 38
operator=, 37
operator==, 37
Point, 36
x, 38
y, 38
Point< T >, 35
Point< T >::hash_point, 20

rotate
GTriangle, 19
MTriangle, 26
Parallelogram, 34
STriangle, 53
Shape, 42
Square, 47

STriangle, 48
center_point, 51
draw, 51
get_Points, 52
get_center_point, 51
is_in_shape, 52
is_in_triangle, 52
move, 53
rotate, 53
STriangle, 49, 50
set_Points, 53
toString, 53
Save, 39
set_Objective
Game, 14
Objective, 29
set_Points
GTriangle, 19
MTriangle, 26
Parallelogram, 34
STriangle, 53
Shape, 42
Square, 47
set_callback
Button, 11
Shape, 39
computeDistance, 40
get_Points, 41
is_in_shape, 41
move, 41
rotate, 42
set_Points, 42
toString, 42
Square, 43
get_Points, 46
is_in_shape, 46
move, 46
rotate, 47
set_Points, 47
Square, 45
toString, 47
stick
Game, 14

toString
GTriangle, 20
MTriangle, 26
Parallelogram, 35
STriangle, 53
Shape, 42
Square, 47

x
Point, 38

y
Point, 38