Tangram

Generated by Doxygen 1.8.13

Contents

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:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

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 (const Point< int > &point, const Point< int > &sizing, std::string text)
  Constructor of a Button.
- Button (const Point< int > &point, const 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 (
            const Point< int > & point,
            const 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 (
            const Point< int > & point,
            const 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:

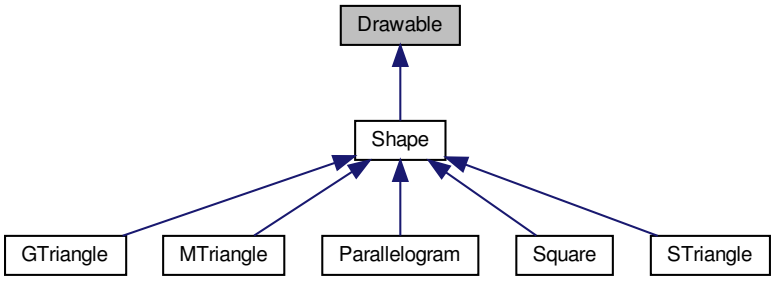
- include/drawable/Button.hpp
- src/drawable/Button.cpp

5.2 Drawable Class Reference

Drawable is everything to draw.

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

Inheritance diagram for Drawable:



Public Member Functions

- ~Drawable ()=default

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

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 ()
  Destructor 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:

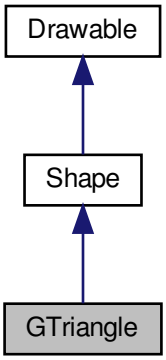
- include/game/Game.hpp
- src/game/Game.cpp

5.4 GTriangle Class Reference

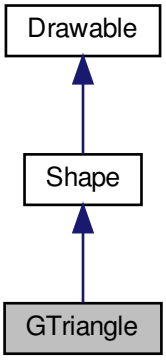
Class of the greatest triangle.

`#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.
- void **draw** (MLV_Color color) override
- 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< T > &p1, const Point< T > &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 (const 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 (
            const Button & button )
```

Add a button in the Menu.

Parameters

button	: Button to add

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

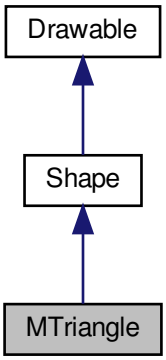
- include/drawable/Menu.hpp
- src/drawable/Menu.cpp

5.8 MTriangle Class Reference

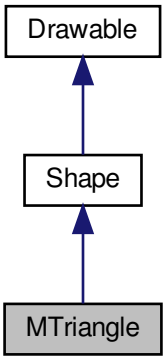
Class of the medium triangle.

```
#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.
- void **draw** (MLV_Color color) override
- 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 (std::shared_ptr< 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 (
            std::shared_ptr< 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:

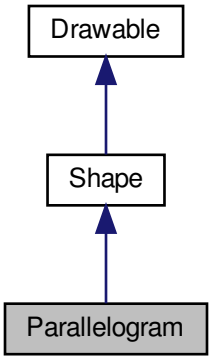
- include/game/Objective.hpp
- src/game/Objective.cpp

5.10 Parallelogram Class Reference

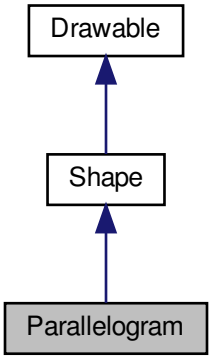
Class of the parallelogram.

`#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.
- void **draw** (MLV_Color color) override
- 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

- constexpr **Point** (const Point< T > &p)=default
- 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 aatomic 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:

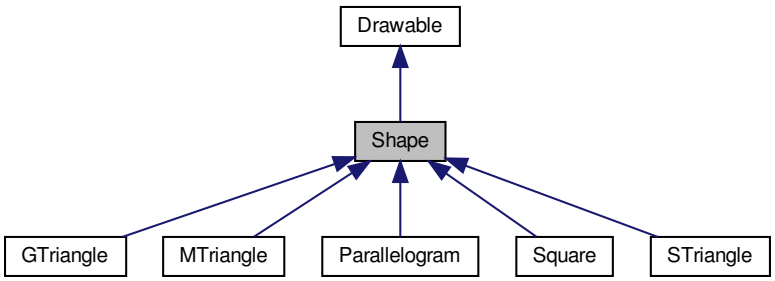
- include/parser/Save.hpp

5.13 Shape Class Reference

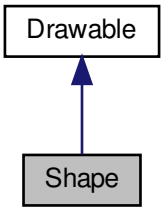
Abstract Class of every Shape.

`#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:

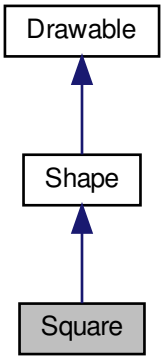
- include/drawable/Shape.hpp
- src/drawable/Shape.cpp

5.14 Square Class Reference

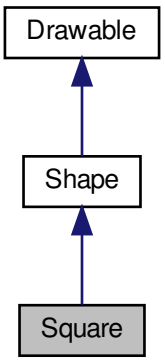
Class of the square.

`#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_PURPLE)

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

- Square (const std::vector< STriangle > &triangle, MLV_Color color=MLV_COLOR_PURPLE)

  Constructor of Square, requires a vector of STriangles.
- Square (const Point< double > &origin, double angular=0.0, MLV_Color color=MLV_COLOR_PURPLE)

  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.
- void **draw** (MLV_Color color) override
- 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_PURPLE )  [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_PURPLE )  [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_PURPLE )  [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:

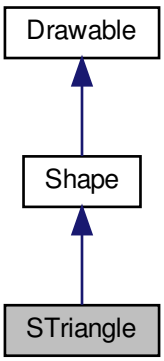
- include/shape/Square.hpp
- src/shape/Square.cpp

5.15 STriangle Class Reference

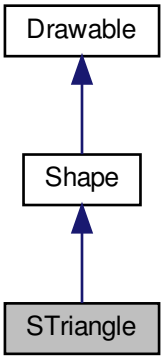
Class of the small triangle.

```
#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) override

  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 )  [override], [virtual]
```

Draw this shape on IHM with specific color.

Parameters

Color	: Color from the graphic library MLV like MLV_COLOR_XXX

Implements Drawable.

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

5.16 Struct Struct Reference

Hash a Point<T> to hash a point with Point<T>

5.16.1 Detailed Description

Hash a Point<T> to hash a point with Point<T>

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

- include/utils/Point.hpp

Chapter 6

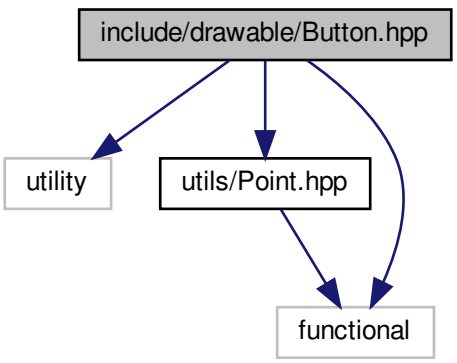
File Documentation

6.1 include/drawable/Button.hpp File Reference

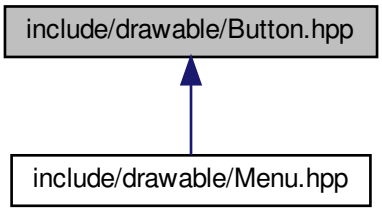
Every buttons of menu.

```
#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

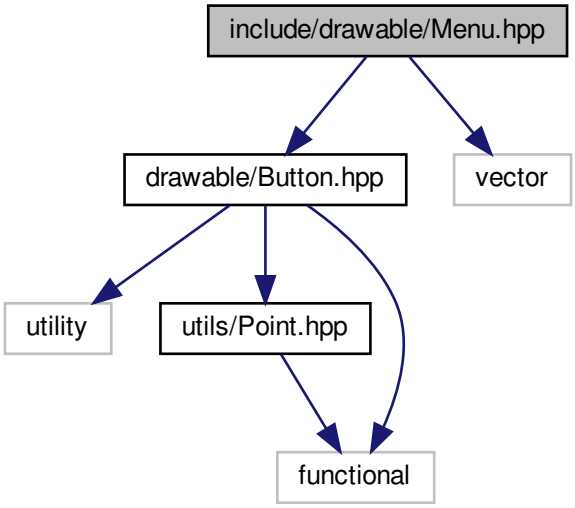
1.0

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

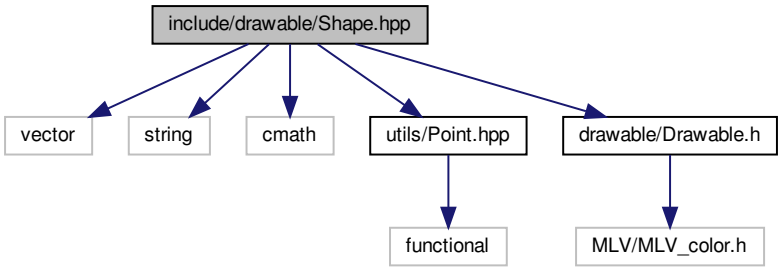
6.3 include/drawable/Shape.hpp File Reference

Abstract Class Shape of every shape in Tangram.

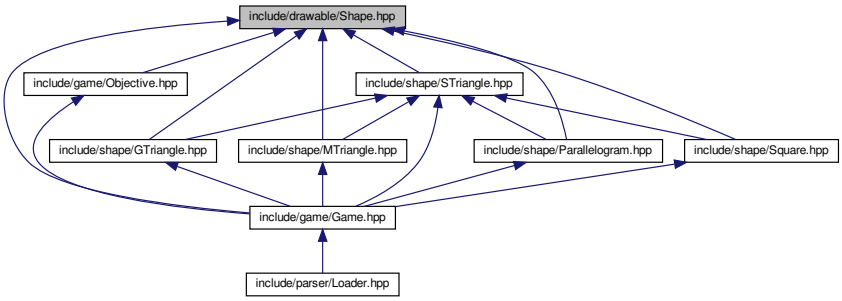
```
#include <vector>
#include <string>
#include <cmath>
#include <utils/Point.hpp>
```

```
#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

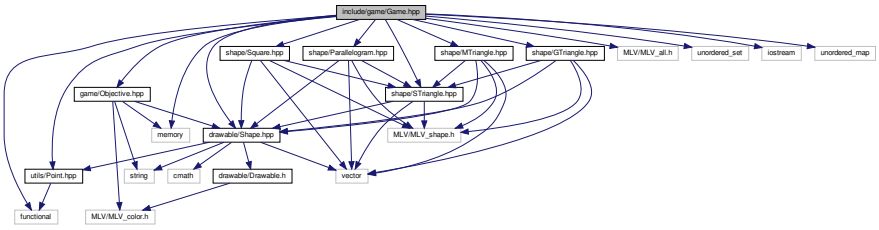
1.0

6.4 include/game/Game.hpp File Reference

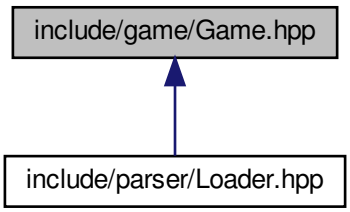
Main Game of the Tangram.

```
#include <drawable/Shape.hpp>
#include <utils/Point.hpp>
#include <game/Objective.hpp>
#include <shape/STriangle.hpp>
#include <shape/MTriangle.hpp>
#include <shape/GTriangle.hpp>
#include <shape/Parallelogram.hpp>
#include <shape/Square.hpp>
#include <MLV/MLV_all.h>
#include <functional>
#include <unordered_set>
#include <memory>
#include <iostream>
#include <unordered_map>
```

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

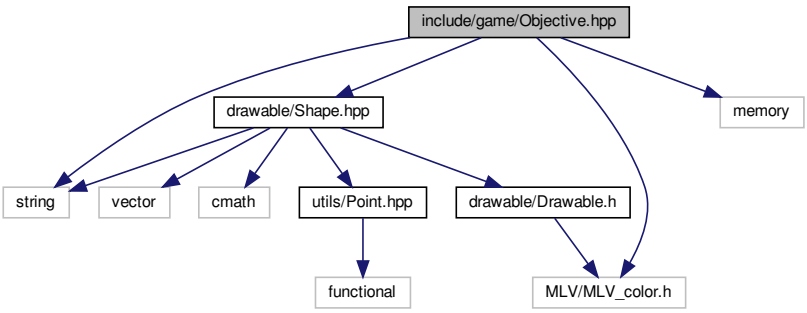
1.0

6.5 include/game/Objective.hpp File Reference

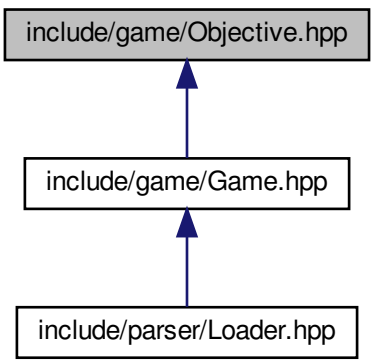
Objective of the Tangram's board.

```
#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

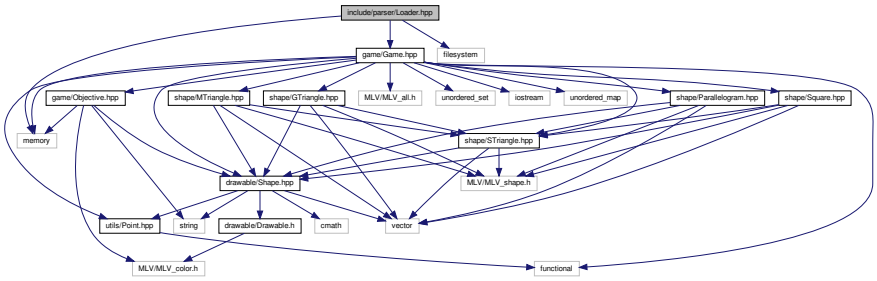
1.0

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

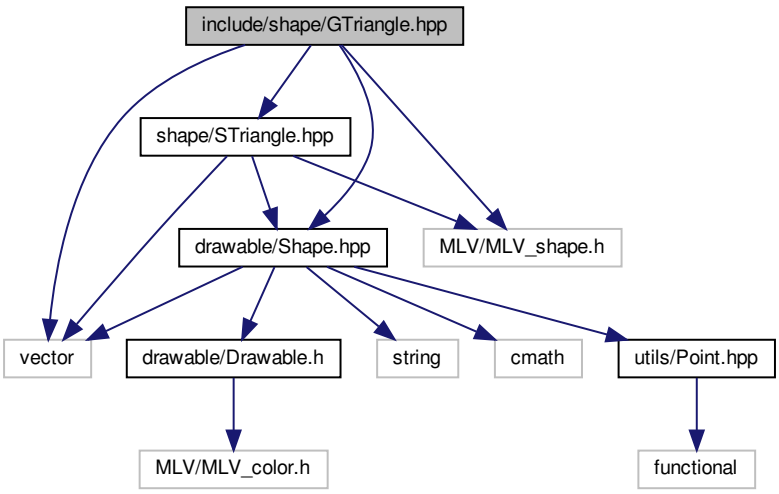
1.0

6.8 include/shape/GTriangle.hpp File Reference

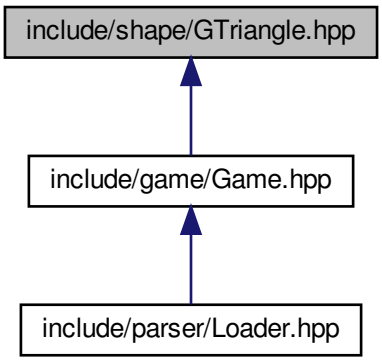
Shape of Great Triangle.

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

Include dependency graph for GTriangle.hpp:



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



Classes

- class GTriangle

  Class of the greatest triangle.

6.8.1 Detailed Description

Shape of Great Triangle.

Author

Jérémie LE BASTARD

Version

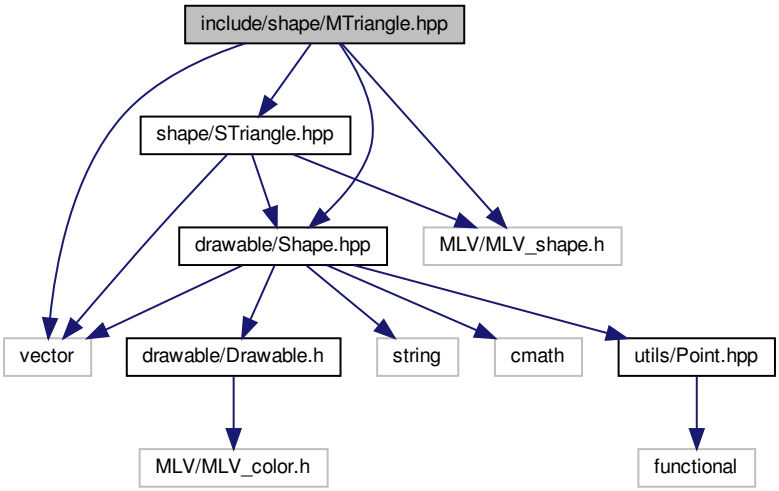
1.0

6.9 include/shape/MTriangle.hpp File Reference

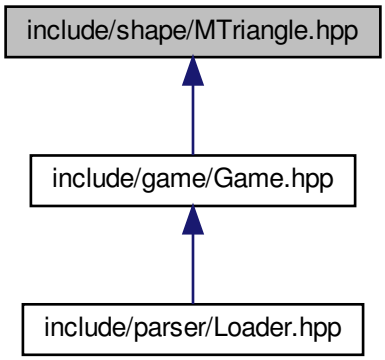
Shape of Medium Triangle.

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

`#include <MLV/MLV_shape.h>`
Include dependency graph for MTriangle.hpp:



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



Classes

- class MTriangle

  Class of the medium triangle.

6.9.1 Detailed Description

Shape of Medium Triangle.

Author

Jérémie LE BASTARD

Version

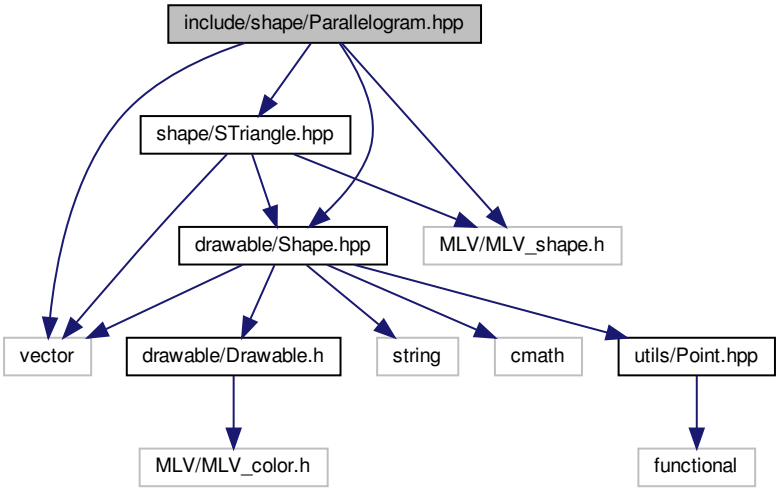
1.0

6.10 include/shape/Parallelogram.hpp File Reference

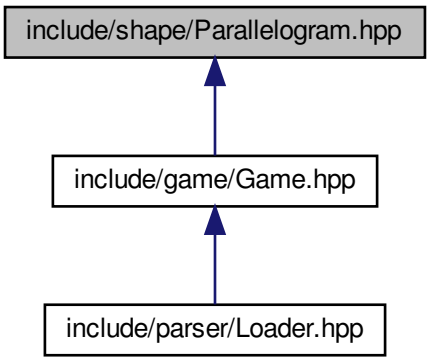
Shape of Parallelogram.

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

Include dependency graph for Parallelogram.hpp:



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



Classes

- class Parallelogram

  Class of the parallelogram.

6.10.1 Detailed Description

Shape of Parallelogram.

Author

Jérémie LE BASTARD

Version

1.0

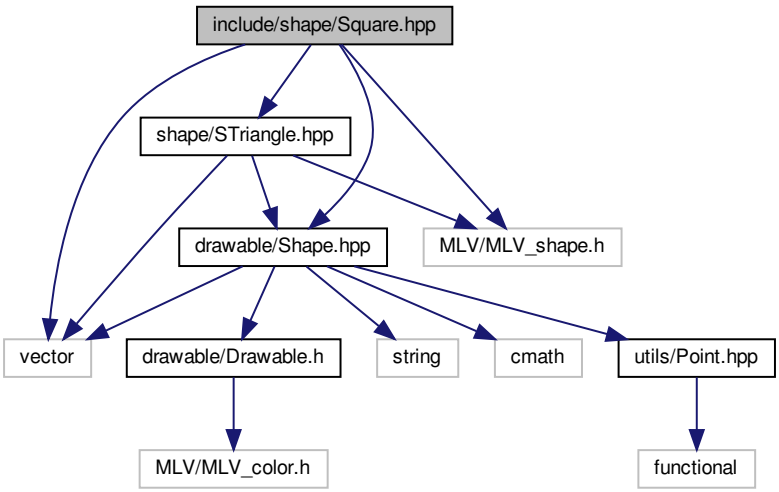
6.11 include/shape/Square.hpp File Reference

Shape of Square.

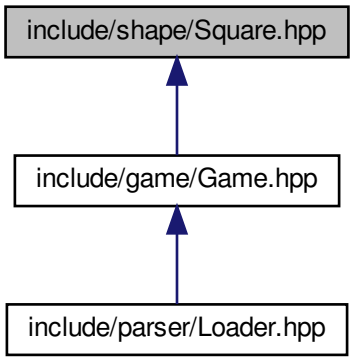
```
#include <vector>
#include <shape/STriangle.hpp>
#include <drawable/Shape.hpp>
```

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

Include dependency graph for Square.hpp:



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



Classes

- class Square

  Class of the square.

6.11.1 Detailed Description

Shape of Square.

Author

Jérémie LE BASTARD

Version

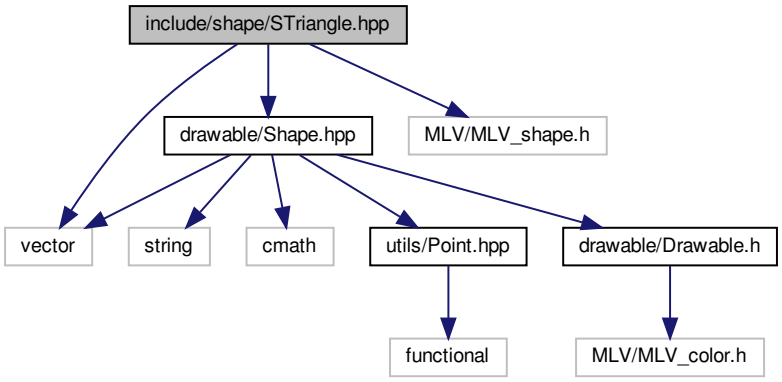
1.0

6.12 include/shape/STriangle.hpp File Reference

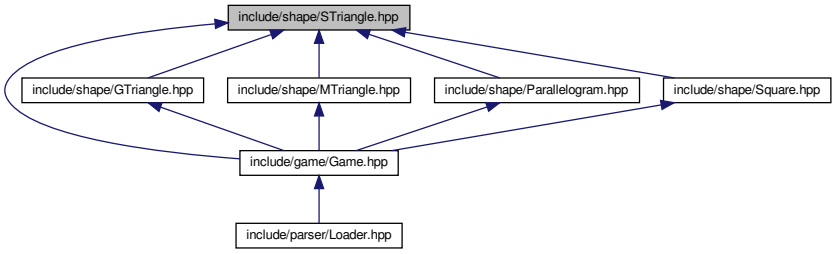
Shape of Small Triangle.

```
#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

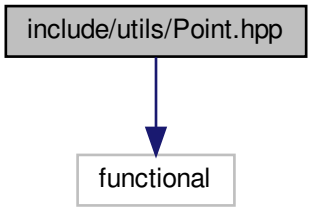
Version

1.0

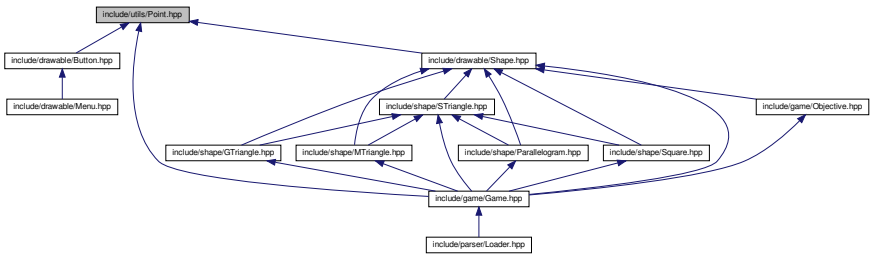
6.13 include/utils/Point.hpp File Reference

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