Tangram

Generated by Doxygen 1.8.13

Contents

Chapter 1

Tangram

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

Getting started

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

CMake

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

Make

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

>$ make

Run

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

How to play

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

You can play now.

Launch Button

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

`>mouse click right` on a shape and drag to rotate it.

`>press 'Esc'` to exit this mode.

press 's' to save the current board as puzzle.

`>press 'd'` on a shape mouseovered to rotate it 45° anti clockwise.

`>press 'f` on a shape mouseovered to rotate it 45° clockwise.

`>press 'r'` to symmetrically reverse the shape.

>Note that last command rotates every shape to 180° except parallelogram which is

overturned (in a mirror fashion)

Load Button

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

`>mouse click right` on a shape and drag to rotate it.

`>press 'Esc'` to exit this mode.

`>press 'd'` on a shape mouseovered to rotate it 45° anti clockwise.

`>press 'f` on a shape mouseovered to rotate it 45° clockwise.

`>press 'r'` to symmetrically reverse the shape.

>Note that last command rotates every shape to 180° except parallelogram which is

overturned (in a mirror fashion)

End Game

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

Documentation

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

HTML

Open with your browser
>$ cd doc/html

>index.html

LaTeX

>$ cd doc/latex

PDF

Open with a PDF reader
>$ cd doc/latex

refman.pdf

Regenerate Documentation

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

If you want customize the documentation generated, you could also configurate the following file : >$ gedit config-file

Regenerate LaTeX Documentation

To generate the PDF documentation, execute the following commands : >$ cd doc/latex

>$ make

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

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 files with brief descriptions:

Chapter 5

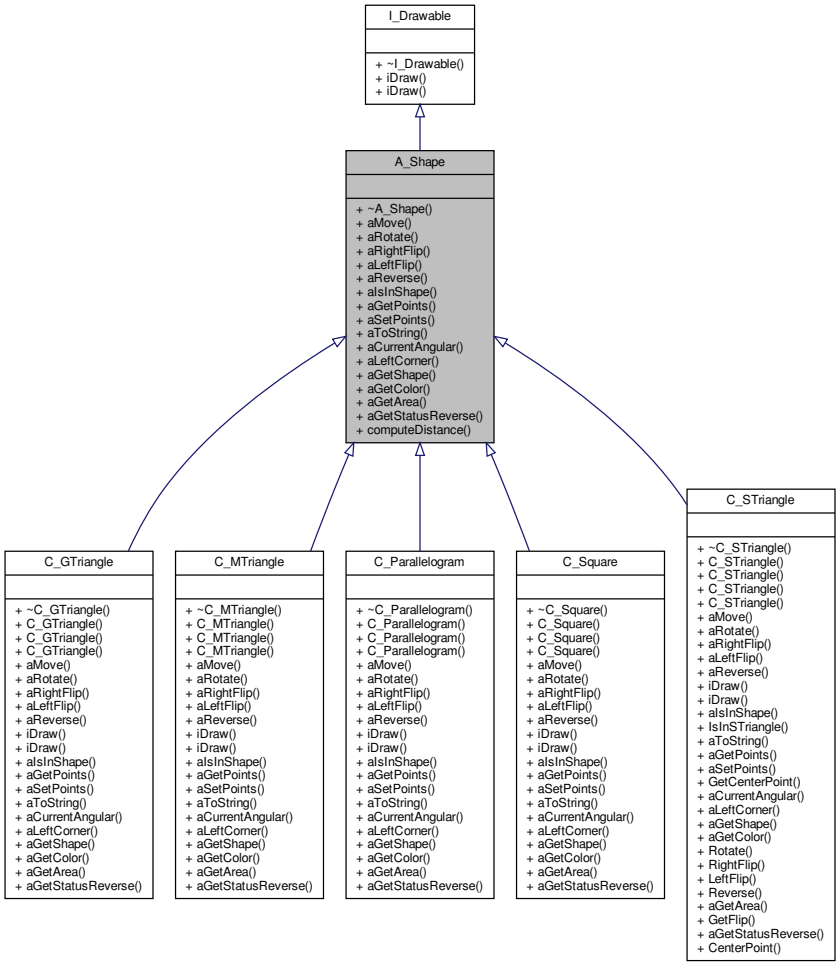
Class Documentation

5.1 A_Shape Class Reference

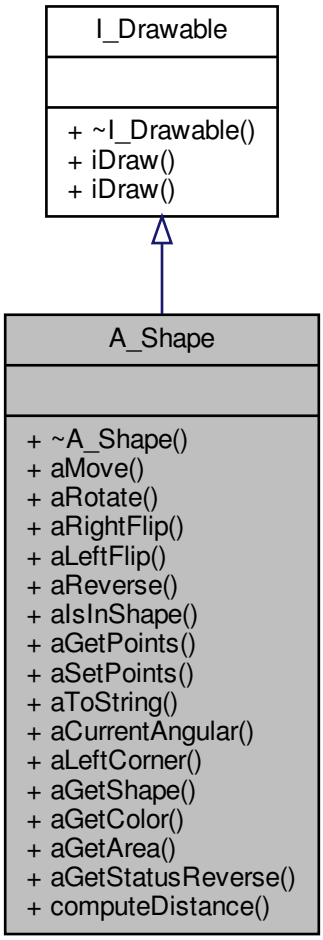
Abstract Class of every A_Shape.

`#include <A_Shape.hpp>`

Inheritance diagram for A_Shape:



Collaboration diagram for A_Shape:



Public Member Functions

- virtual ~A_Shape ()=0

  Destructor of Abstract A_Shape.
- virtual void aMove (const T_Point< double > &translation)=0

  Pure virtual function. Move the A_Shape by point translation.
- virtual void aRotate (double angular)=0

  Pure virtual function. Rotate the A_Shape with specified angular.
- virtual void aRightFlip ()=0

  Pure virtual function. Flip the figure as 45° clock (Pi/4)
- virtual void aLeftFlip ()=0

  Pure virtual function. Flip the figure as 45° anti clock (Pi/4)
- virtual void aReverse ()=0

  Pure virtual function. Reverse the shape as symmetry.
- virtual bool aIsInShape (const T_Point< double > &point)=0

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

- virtual std::vector< T_Point< double > > aGetPoints ()=0

  Pure virtual function. Get all mPoints of this shape.
- virtual bool aSetPoints (const T_Point< double > &ref, const T_Point< double > &changed)=0

  Pure virtual function. Get all mPoints of this shape.
- virtual std::string aToString ()=0

  Pure virtual function. Convert all data of A_Shape in a string.
- virtual double aCurrentAngular ()=0

  Pure virtual function. Get the current angular of a A_Shape.
- virtual T_Point< double > aLeftCorner ()=0

  Pure virtual function. Take the point at left top corner of a A_Shape.
- virtual std::string aGetShape ()=0

  Pure virtual function. Get the A_Shape type.
- virtual MLV_Color aGetColor ()=0

  Pure virtual function. Get the color of a A_Shape.
- virtual double aGetArea ()=0

  Pure virtual function. Get the area of a A_Shape.
- virtual bool aGetStatusReverse () const =0

  Get the status of shape reversed or not.

Static Public Member Functions

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

  Compute distance between 2 mPoints.

5.1.1 Detailed Description

Abstract Class of every A_Shape.

This class manage everything other shape (C_STriangle, C_MTriangle, C_GTriangle, C_Square, C_Parallelogram)

5.1.2 Constructor & Destructor Documentation

5.1.2.1 ~A_Shape()

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

Destructor of Abstract A_Shape.

5.1.3 Member Function Documentation

5.1.3.1 aCurrentAngular()

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

Pure virtual function. Get the current angular of a A_Shape.

Returns

Return the current angular of a A_Shape as double

Implemented in C_STriangle, C_Parallelogram, C_MTriangle, C_Square, and C_GTriangle.

5.1.3.2 aGetArea()

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

Pure virtual function. Get the area of a A_Shape.

Returns

Return the area of a A_Shape

Implemented in C_STriangle, C_Parallelogram, C_MTriangle, C_Square, and C_GTriangle.

5.1.3.3 aGetColor()

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

Pure virtual function. Get the color of a A_Shape.

Returns

Return the MLV_Color of a A_Shape

Implemented in C_STriangle, C_Parallelogram, C_MTriangle, C_Square, and C_GTriangle.

5.1.3.4 aGetPoints()

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

Pure virtual function. Get all mPoints of this shape.

Returns

Return a vector of mPoints of this shape

Implemented in C_STriangle, C_Parallelogram, C_MTriangle, C_Square, and C_GTriangle.

5.1.3.5 aGetShape()

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

Pure virtual function. Get the A_Shape type.

Returns

Return as string a A_Shape type

Implemented in C_STriangle, C_Parallelogram, C_MTriangle, C_Square, and C_GTriangle.

5.1.3.6 aGetStatusReverse()

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

Get the status of shape reversed or not.

Returns

Return true if the shape got reversed, false otherwise

Implemented in C_STriangle, C_Parallelogram, C_MTriangle, C_Square, and C_GTriangle.

5.1.3.7 aIsInShape()

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

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

Parameters

point	: T_Point to check

Returns

true if Click is in this shape, false if not

Implemented in C_STriangle, C_Parallelogram, C_MTriangle, C_Square, and C_GTriangle.

5.1.3.8 aLeftCorner()

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

Pure virtual function. Take the point at left top corner of a A_Shape.

Returns

Return the point at left top corner

Implemented in C_STriangle, C_Parallelogram, C_MTriangle, C_Square, and C_GTriangle.

5.1.3.9 aLeftFlip()

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

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

Implemented in C_STriangle, C_Parallelogram, C_MTriangle, C_Square, and C_GTriangle.

5.1.3.10 aMove()

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

Pure virtual function. Move the A_Shape by point translation.

Parameters

translation	: Every mPoints of this shape will be translate by this __Parameter

Implemented in C_STriangle, C_Parallelogram, C_MTriangle, C_Square, and C_GTriangle.

5.1.3.11 aReverse()

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

Pure virtual function. Reverse the shape as symmetry.

Implemented in C_STriangle, C_Parallelogram, C_MTriangle, C_Square, and C_GTriangle.

5.1.3.12 aRightFlip()

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

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

Implemented in C_STriangle, C_Parallelogram, C_MTriangle, C_Square, and C_GTriangle.

5.1.3.13 aRotate()

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

Pure virtual function. Rotate the A_Shape with specified angular.

Parameters

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

Implemented in C_STriangle, C_Parallelogram, C_MTriangle, C_Square, and C_GTriangle.

5.1.3.14 aSetPoints()

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

Pure virtual function. Get all mPoints of this shape.

Returns

Return a vector of mPoints of this shape

Implemented in C_STriangle, C_Parallelogram, C_MTriangle, C_Square, and C_GTriangle.

5.1.3.15 aToString()

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

Pure virtual function. Convert all data of A_Shape in a string.

Returns

Return a string which contains every mPoints of this shape

Implemented in C_STriangle, C_Parallelogram, C_MTriangle, C_Square, and C_GTriangle.

5.1.3.16 computeDistance()

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

Compute distance between 2 mPoints.

Parameters

point1	: First point
point2	: Second point

Returns

Return the distance between these two mPoints

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

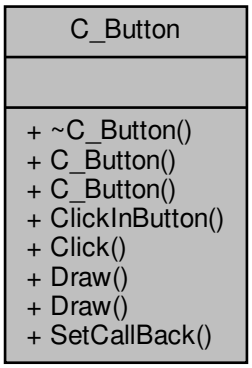
- include/drawable/A_Shape.hpp
- src/drawable/A_Shape.cpp

5.2 C_Button Class Reference

C_Button of the C_Menu.

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

Collaboration diagram for C_Button:



Public Member Functions

- ~C_Button ()

  Class methods.
- C_Button (const T_Point< int > &point, const T_Point< int > &sizing, std::string text)

  Constructor of a C_Button.
- C_Button (const T_Point< int > &point, const T_Point< int > &sizing, std::string text, std::function< int(int)> callback)

  Constructor of a C_Button.
- bool ClickInButton (const T_Point< int > &click)

  Check if a Click is in the button.
- int Click (int)

  Define a value about a Click.
- void Draw ()

  Draw the button.
- void Draw (MLV_Color color)

  Draw the button with specific color.
- void SetCallBack (std::function< int(int)> callback)

  Set a callback for a button.

5.2.1 Detailed Description

C_Button of the C_Menu.

This class manage all mButtons of the menu

5.2.2 Constructor & Destructor Documentation

5.2.2.1 ~C_Button()

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

Class methods.

Destructor of the C_Button

5.2.2.2 C_Button() [1/2]

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

Constructor of a C_Button.

Parameters

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

5.2.2.3 C_Button() [2/2]

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

Constructor of a C_Button.

Parameters

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

5.2.3 Member Function Documentation

5.2.3.1 Click()

```
int C_Button::Click (
            int val )
```

Define a value about a Click.

Returns

Return a value about a Click

5.2.3.2 ClickInButton()

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

Check if a Click is in the button.

Parameters

click	: T_Point to check

Returns

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

5.2.3.3 Draw() [1/2]

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

Draw the button.

5.2.3.4 Draw() [2/2]

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

Draw the button with specific color.

Parameters

color	: MLV_Color needed to draw the button

5.2.3.5 SetCallBack()

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

Set a callback for a button.

Parameters

callback	: Requires a pointer of function for set the callback

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

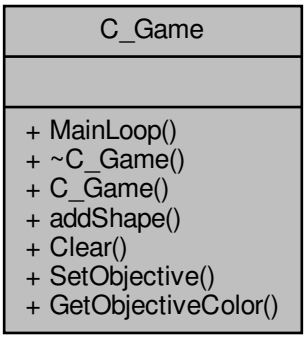
- include/drawable/C_Button.hpp
- src/drawable/C_Button.cpp

5.3 C_Game Class Reference

Class of the main C_Game.

`#include <C_Game.hpp>`

Collaboration diagram for C_Game:



Public Member Functions

- void MainLoop ()

  Main loop of the game.
- ~C_Game ()

  Destructor of the game.
- C_Game (int w, int h)

  Constructor of the game, initialize a game with an sizing.
- void addShape (std::shared_ptr< A_Shape > s)

  Add a shape in the game.
- void Clear ()

  Clear the game / the board and the mObjective.
- void SetObjective (const std::vector< std::shared_ptr< A_Shape >> &vec_objective)

  Set the mObjective of the game.
- MLV_Color GetObjectiveColor ()

  Get the mColor of the mObjective of the game.

5.3.1 Detailed Description

Class of the main C_Game.

This class manage everything about the main game

5.3.2 Constructor & Destructor Documentation

5.3.2.1 ~C_Game()

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

Destructor of the game.

5.3.2.2 C_Game()

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

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

Parameters

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

5.3.3 Member Function Documentation

5.3.3.1 addShape()

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

Add a shape in the game.

Parameters

s	: A_Shape to add

5.3.3.2 Clear()

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

Clear the game / the board and the mObjective.

5.3.3.3 GetObjectiveColor()

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

Get the mColor of the mObjective of the game.

Returns

Return the mColor of the mObjective of the game

5.3.3.4 MainLoop()

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

Main loop of the game.

5.3.3.5 SetObjective()

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

Set the mObjective of the game.

Parameters

vec_objective	: Vector of C_Objective for new game;

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

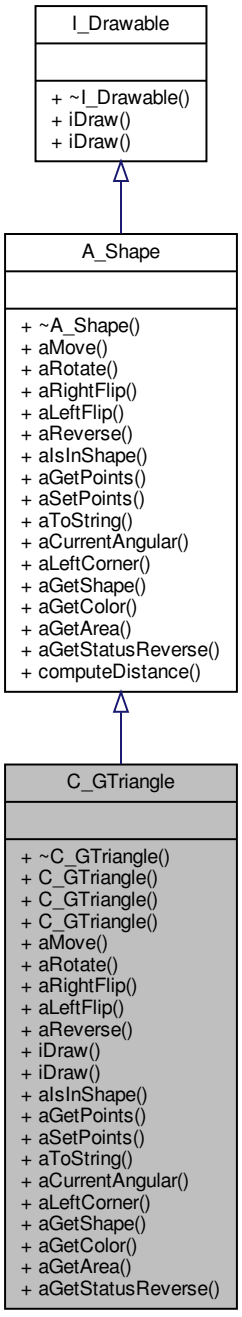
- include/game/C_Game.hpp
- src/game/C_Game.cpp

5.4 C_GTriangle Class Reference

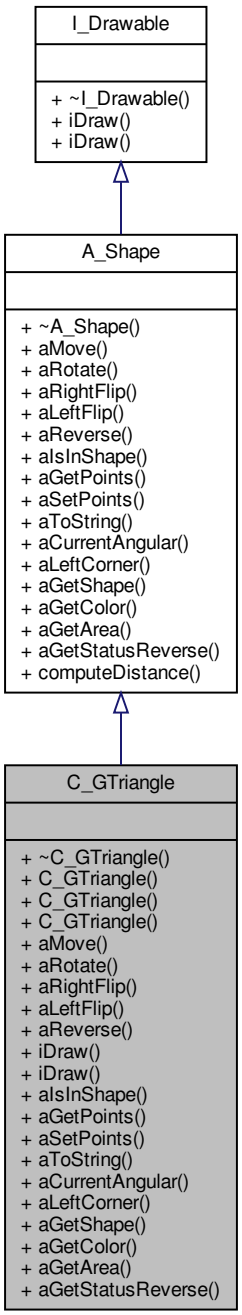
Class of the greatest C_GTriangle.

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

Inheritance diagram for C_GTriangle:



Collaboration diagram for C_GTriangle:



Public Member Functions

- ~C_GTriangle () override

  Destructor of C_GTriangle.
- C_GTriangle (MLV_Color color=MLV_COLOR_RED)

  Constructor by default of C_GTriangle, make a C_GTriangle as default.
- C_GTriangle (const std::vector< C_STriangle > &triangle, MLV_Color color=MLV_COLOR_RED)

Constructor of C_GTriangle, requires a vector of triangles.

- C_GTriangle (const T_Point< double > &origin, double angular=0.0, MLV_Color color=MLV_COLOR_RED)

  Constructor of C_GTriangle, calls the deleguate Default Constructor.
- void aMove (const T_Point< double > &translation) override

  Move the C_GTriangle by point translation.
- void aRotate (double angular) override

  Rotate the C_GTriangle with specified angular.
- void aRightFlip () override

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

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

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

  Draw this shape on IHM.
- void iDraw (MLV_Color color) override

  Draw this shape on IHM with a specific mColor.
- bool aIsInShape (const T_Point< double > &click) override

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

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

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

  Convert all data of C_GTriangle in a string.
- double aCurrentAngular () override

  Get the current angular of this shape.
- T_Point< double > aLeftCorner () override

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

  Get the shape type.
- MLV_Color aGetColor () override

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

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

  Get the status of shape reversed or not.

Additional Inherited Members

5.4.1 Detailed Description

Class of the greatest C_GTriangle.

This class manage everything about the greatest G_GTriangle

5.4.2 Constructor & Destructor Documentation

5.4.2.1 ~C_GTriangle()

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

Destructor of C_GTriangle.

5.4.2.2 C_GTriangle() [1/3]

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

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

Parameters

color	: Optional __Parameter, mColor of this shape

5.4.2.3 C_GTriangle() [2/3]

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

Constructor of C_GTriangle, requires a vector of triangles.

Parameters

triangle	: The C_GTriangle will created with a vector of C_STriangle (4)
color	: Optional __Parameter, mColor of this shape

5.4.2.4 C_GTriangle() [3/3]

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

Constructor of C_GTriangle, calls the deleguate Default Constructor.

Parameters

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

5.4.3 Member Function Documentation

5.4.3.1 aCurrentAngular()

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

Get the current angular of this shape.

Returns

Implements A_Shape.

5.4.3.2 aGetArea()

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

Get the area of the shape.

Returns

Return the area of the shape

Implements A_Shape.

5.4.3.3 aGetColor()

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

Get the color of the shape.

Returns

Return the MLV_Color of the shape

Implements A_Shape.

5.4.3.4 aGetPoints()

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

Get mPoints of this shape.

Returns

Return a vector of mPoints of this shape

Implements A_Shape.

5.4.3.5 aGetShape()

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

Get the shape type.

Returns

Return as string the shape type

Implements A_Shape.

5.4.3.6 aGetStatusReverse()

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

Get the status of shape reversed or not.

Returns

Return true if the shape got reversed, false otherwise

Implements A_Shape.

5.4.3.7 aIsInShape()

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

Check if a point is in this shape.

Parameters

click	: T_Point to check

Returns

true if Click is in this shape, false if not

Implements A_Shape.

5.4.3.8 aLeftCorner()

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

Take the point at left top corner.

Returns

Return the point at left top corner

Implements A_Shape.

5.4.3.9 aLeftFlip()

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

Flip the figure as 45° anti clock.

Implements A_Shape.

5.4.3.10 aMove()

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

Move the C_GTriangle by point translation.

Parameters

translation	: Every mPoints of this shape will be translate by this __Parameter

Implements A_Shape.

5.4.3.11 aReverse()

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

Reverse the figure as symmetry.

Implements A_Shape.

5.4.3.12 aRightFlip()

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

Flip the figure as 45° clock.

Implements A_Shape.

5.4.3.13 aRotate()

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

Rotate the C_GTriangle with specified angular.

Parameters

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

Implements A_Shape.

5.4.3.14 aSetPoints()

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

Set mPoints of this shape.

Returns

Return a true if something has been changed, false either

Implements A_Shape.

5.4.3.15 aToString()

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

Convert all data of C_GTriangle in a string.

Returns

Return a string which contains every mPoints of this shape

Implements A_Shape.

5.4.3.16 iDraw() [1/2]

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

Draw this shape on IHM.

Implements I_Drawable.

5.4.3.17 iDraw() [2/2]

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

Draw this shape on IHM with a specific mColor.

Parameters

color	Color used to __Draw the shape

Implements I_Drawable.

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

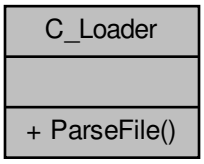
- include/shape/C_GTriangle.hpp
- src/shape/C_GTriangle.cpp

5.5 C_Loader Class Reference

Class of the main C_Loader.

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

Collaboration diagram for C_Loader:



Static Public Member Functions

- static bool ParseFile (const std::string &filename, C_Game &game)

  Parse a file to make a board.

5.5.1 Detailed Description

Class of the main C_Loader.

This class manage everything about the loader

5.5.2 Member Function Documentation

5.5.2.1 ParseFile()

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

Parse a file to make a board.

Parameters

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:

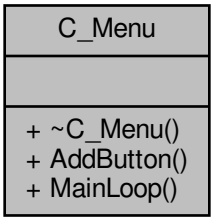
- include/parser/C_Loader.hpp
- src/parser/C_Loader.cpp

5.6 C_Menu Class Reference

C_Menu of the game.

`#include <C_Menu.hpp>`

Collaboration diagram for C_Menu:



Public Member Functions

- ~C_Menu ()
- void AddButton (const C_Button &button)

  Add a button in the C_Menu.
- void MainLoop ()

  Main loop of the C_Menu.

5.6.1 Detailed Description

C_Menu of the game.

This class manage everything about Tangram's menu

5.6.2 Constructor & Destructor Documentation

5.6.2.1 ~C_Menu()

`C_Menu::~C_Menu ( )`

5.6.3 Member Function Documentation

5.6.3.1 AddButton()

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

Add a button in the C_Menu.

Parameters

button	: C_Button to add

5.6.3.2 MainLoop()

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

Main loop of the C_Menu.

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

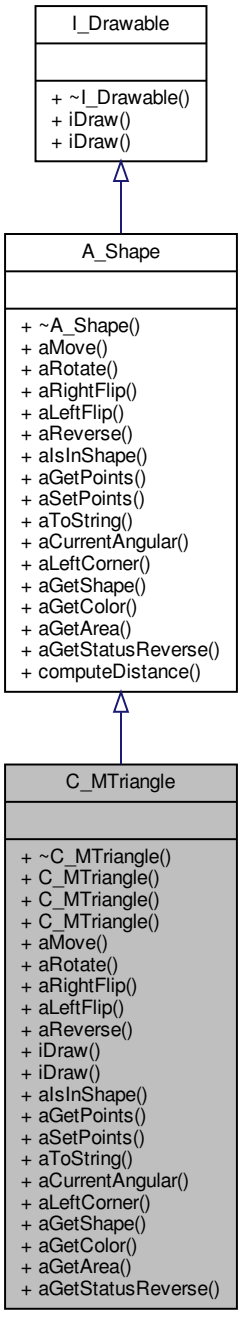
- include/drawable/C_Menu.hpp
- src/drawable/C_Menu.cpp

5.7 C_MTriangle Class Reference

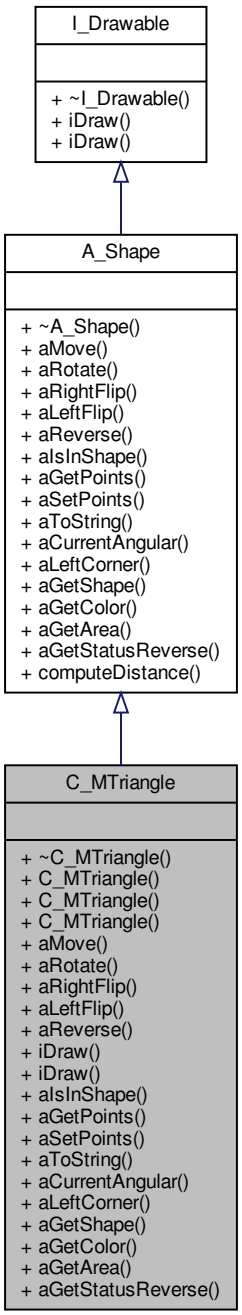
Class of the medium C_MTriangle.

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

Inheritance diagram for C_MTriangle:



Collaboration diagram for C_MTriangle:



Public Member Functions

- ~C_MTriangle () override

  Destructor of C_MTriangle.
- C_MTriangle (MLV_Color color=MLV_COLOR_ORANGE)

  Constructor by default of C_MTriangle, make a C_MTriangle as default.
- C_MTriangle (const std::vector< C_STriangle > &triangle, MLV_Color color=MLV_COLOR_ORANGE)

Constructor of C_MTriangle, requires a vector of STriangles.

- C_MTriangle (const T_Point< double > &origin, double angular=0.0, MLV_Color color=MLV_COLOR_OR↩ ANGE)

  Constructor of C_MTriangle, calls the deleguate Default Constructor.
- void aMove (const T_Point< double > &translation) override

  Move the C_MTriangle by point translation.
- void aRotate (double angular) override

  Rotate the C_MTriangle with specified angular.
- void aRightFlip () override

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

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

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

  Draw this shape on IHM.
- void iDraw (MLV_Color color) override

  Draw this shape on IHM with specific color.
- bool aIsInShape (const T_Point< double > &click) override

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

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

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

  Convert all data of C_MTriangle in a string.
- double aCurrentAngular () override

  Get the current angular of this shape.
- T_Point< double > aLeftCorner () override

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

  Get the shape type.
- MLV_Color aGetColor () override

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

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

  Get the status of shape reversed or not.

Additional Inherited Members

5.7.1 Detailed Description

Class of the medium C_MTriangle.

This class manage everything about the medium C_MTriangle

5.7.2 Constructor & Destructor Documentation

5.7.2.1 ~C_MTriangle()

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

Destructor of C_MTriangle.

5.7.2.2 C_MTriangle() [1/3]

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

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

Parameters

color	: Optional __Parameter, mColor of this shape

5.7.2.3 C_MTriangle() [2/3]

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

Constructor of C_MTriangle, requires a vector of STriangles.

Parameters

triangle	: The C_MTriangle will created with a vector of C_STriangle (4)
color	: Optional __Parameter, mColor of this shape

5.7.2.4 C_MTriangle() [3/3]

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

Constructor of C_MTriangle, calls the deleguate Default Constructor.

Parameters

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

5.7.3 Member Function Documentation

5.7.3.1 aCurrentAngular()

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

Get the current angular of this shape.

Returns

Implements A_Shape.

5.7.3.2 aGetArea()

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

Get the area of the shape.

Returns

Return the area of the shape

Implements A_Shape.

5.7.3.3 aGetColor()

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

Get the color of the shape.

Returns

Return the MLV_Color of the shape

Implements A_Shape.

5.7.3.4 aGetPoints()

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

Get mPoints of this shape.

Returns

Return a vector of mPoints of this shape

Implements A_Shape.

5.7.3.5 aGetShape()

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

Get the shape type.

Returns

Return as string the shape type

Implements A_Shape.

5.7.3.6 aGetStatusReverse()

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

Get the status of shape reversed or not.

Returns

Return true if the shape got reversed, false otherwise

Implements A_Shape.

5.7.3.7 aIsInShape()

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

Check if a point is in this shape.

Parameters

click	: T_Point to check

Returns

true if Click is in this shape, false if not

Implements A_Shape.

5.7.3.8 aLeftCorner()

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

Take the point at left top corner.

Returns

Return the point at left top corner

Implements A_Shape.

5.7.3.9 aLeftFlip()

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

Flip the figure as 45° anti clock.

Implements A_Shape.

5.7.3.10 aMove()

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

Move the C_MTriangle by point translation.

Parameters

translation	: Every mPoints of this shape will be translate by this __Parameter

Implements A_Shape.

5.7.3.11 aReverse()

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

Reverse the figure as symmetry.

Implements A_Shape.

5.7.3.12 aRightFlip()

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

Flip the figure as 45° clock.

Implements A_Shape.

5.7.3.13 aRotate()

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

Rotate the C_MTriangle with specified angular.

Parameters

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

Implements A_Shape.

5.7.3.14 aSetPoints()

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

Set a point to another one.

Parameters

ref	: Point to change
changed	: New value of the point

Returns

True if the ref point exists, false otherwise

Implements A_Shape.

5.7.3.15 aToString()

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

Convert all data of C_MTriangle in a string.

Returns

Return a string which contains every mPoints of this shape

Implements A_Shape.

5.7.3.16 iDraw() [1/2]

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

Draw this shape on IHM.

Implements I_Drawable.

5.7.3.17 iDraw() [2/2]

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

Draw this shape on IHM with specific color.

Parameters

color	: Color of the shape will be draw

Implements I_Drawable.

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

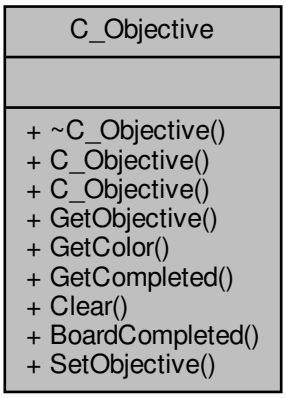
- include/shape/C_MTriangle.hpp
- src/shape/C_MTriangle.cpp

5.8 C_Objective Class Reference

Class of the board C_Objective.

`#include <C_Objective.hpp>`

Collaboration diagram for C_Objective:



Public Member Functions

- ~C_Objective ()
  Class methods.
- C_Objective (MLV_Color color=MLV_COLOR_GRAY70)
  Constructor of an mObjective, default constructor.
- C_Objective (const std::vector< std::shared_ptr< A_Shape >> &objective, MLV_Color color=MLV_COL↩ OR_GRAY70)
  Constructor of an mObjective.
- std::vector< std::shared_ptr< A_Shape > > GetObjective ()
  Get all shape of the mObjective.
- MLV_Color GetColor ()
  Get the mColor of an C_Objective.
- double GetCompleted (const std::vector< std::shared_ptr< A_Shape >> &objective, const std::vector< std::shared_ptr< A_Shape >> &game)
  Give the progress of the puzzle.
- void Clear ()
  Clear the objective.

Static Public Member Functions

- static bool BoardCompleted (const std::vector< std::shared_ptr< A_Shape >> &objective, const std↩ ::vector< std::shared_ptr< A_Shape >> &game)
  Check if the board is mCompleted.
- static void SetObjective (std::shared_ptr< C_Objective > objective, const std::vector< std::shared_ptr< A↩ _Shape >> &vec_objective)
  Set an C_Objective for a new game.

5.8.1 Detailed Description

Class of the board C_Objective.

This class manage everything about the mObjective

5.8.2 Constructor & Destructor Documentation

5.8.2.1 ~C_Objective()

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

Class methods.

5.8.2.2 C_Objective() [1/2]

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

Constructor of an mObjective, default constructor.

Parameters

color	: mColor of the mObjective shape

5.8.2.3 C_Objective() [2/2]

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

Constructor of an mObjective.

Parameters

objective	: C_Objective requires a vector of A_Shape
color	: mColor of the mObjective shape

5.8.3 Member Function Documentation

5.8.3.1 BoardCompleted()

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

Check if the board is mCompleted.

Parameters

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

Returns

True if the board is mCompleted, false if not

5.8.3.2 Clear()

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

Clear the objective.

5.8.3.3 GetColor()

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

Get the mColor of an C_Objective.

Returns

Return the mColor of an C_Objective

5.8.3.4 GetCompleted()

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

Give the progress of the puzzle.

Parameters

objective	: Shapes of objective
game	: Shape of the game

Returns

Return the %100 of the progress

5.8.3.5 GetObjective()

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

Get all shape of the mObjective.

Returns

Return a vector of shape of the mObjective

5.8.3.6 SetObjective()

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

Set an C_Objective for a new game.

Parameters

objective	: C_Objective to mUpdate
vec_objective	:Vector of new A_Shape for the new C_Objective

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

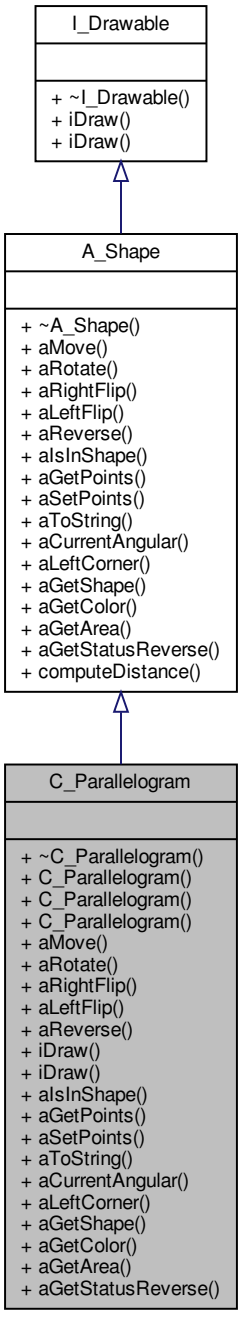
- include/game/C_Objective.hpp
- src/game/C_Objective.cpp

5.9 C_Parallelogram Class Reference

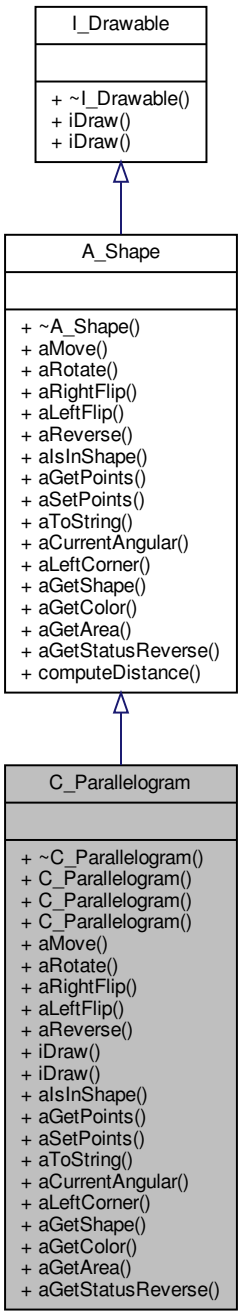
Class of the parallelogram.

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

Inheritance diagram for C_Parallelogram:



Collaboration diagram for C_Parallelogram:



Public Member Functions

- ~C_Parallelogram () override

  Destructor of C_Parallelogram.
- C_Parallelogram (MLV_Color color=MLV_COLOR_BLUE)

  Constructor by default of C_Parallelogram, make a C_Parallelogram as default.
- C_Parallelogram (const std::vector< C_STriangle > &triangle, MLV_Color color=MLV_COLOR_BLUE)

Constructor of C_Parallelogram, requires a vector of STriangles.

- C_Parallelogram (const T_Point< double > &origin, double angular=0.0, MLV_Color color=MLV_COLOR↩ _BLUE, bool reverse=false)

  Constructor of C_Parallelogram, calls the deleguate Default Constructor.
- void aMove (const T_Point< double > &translation) override

  Move the C_Parallelogram by point translation.
- void aRotate (double angular) override

  Rotate the C_Parallelogram with specified angular.
- void aRightFlip () override

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

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

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

  Draw this shape on IHM.
- void iDraw (MLV_Color color) override

  Draw this shape on IHM with specific color.
- bool aIsInShape (const T_Point< double > &click) override

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

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

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

  Convert all data of C_Parallelogram in a string.
- double aCurrentAngular () override

  Get the current angular of this shape.
- T_Point< double > aLeftCorner () override

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

  Get the shape type.
- MLV_Color aGetColor () override

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

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

  Get the status of shape reversed or not.

Additional Inherited Members

5.9.1 Detailed Description

Class of the parallelogram.

This class manage everything about the C_Parallelogram

5.9.2 Constructor & Destructor Documentation

5.9.2.1 ~C_Parallelogram()

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

Destructor of C_Parallelogram.

5.9.2.2 C_Parallelogram() [1/3]

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

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

Parameters

color	: Optional __Parameter, mColor of this shape

5.9.2.3 C_Parallelogram() [2/3]

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

Constructor of C_Parallelogram, requires a vector of STriangles.

Parameters

triangle	: The C_Parallelogram will created with a vector of C_STriangle (4)
color	: Optional __Parameter, mColor of this shape

5.9.2.4 C_Parallelogram() [3/3]

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

Constructor of C_Parallelogram, calls the deleguate Default Constructor.

Parameters

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

5.9.3 Member Function Documentation

5.9.3.1 aCurrentAngular()

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

Get the current angular of this shape.

Returns

Implements A_Shape.

5.9.3.2 aGetArea()

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

Get the area of the shape.

Returns

Return the area of the shape

Implements A_Shape.

5.9.3.3 aGetColor()

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

Get the color of the shape.

Returns

Return the MLV_Color of the shape

Implements A_Shape.

5.9.3.4 aGetPoints()

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

Get mPoints of this shape.

Returns

Return a vector of mPoints of this shape

Implements A_Shape.

5.9.3.5 aGetShape()

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

Get the shape type.

Returns

Return as string the shape type

Implements A_Shape.

5.9.3.6 aGetStatusReverse()

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

Get the status of shape reversed or not.

Returns

Return true if the shape got reversed, false otherwise

Implements A_Shape.

5.9.3.7 aIsInShape()

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

Check if a point is in this shape.

Parameters

click	: T_Point to check

Returns

true if Click is in this shape, false if not

Implements A_Shape.

5.9.3.8 aLeftCorner()

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

Take the point at left top corner.

Returns

Return the point at left top corner

Implements A_Shape.

5.9.3.9 aLeftFlip()

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

Flip the figure as 45° anti clock.

Implements A_Shape.

5.9.3.10 aMove()

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

Move the C_Parallelogram by point translation.

Parameters

translation	: Every mPoints of this shape will be translate by this __Parameter

Implements A_Shape.

5.9.3.11 aReverse()

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

Reverse the figure as symmetry.

Implements A_Shape.

5.9.3.12 aRightFlip()

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

Flip the figure as 45° clock.

Implements A_Shape.

5.9.3.13 aRotate()

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

Rotate the C_Parallelogram with specified angular.

Parameters

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

Implements A_Shape.

5.9.3.14 aSetPoints()

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

Set a point to another one.

Parameters

ref	: Point to change
changed	: New value of the point

Returns

True if the ref point exists, false otherwise

Implements A_Shape.

5.9.3.15 aToString()

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

Convert all data of C_Parallelogram in a string.

Returns

Return a string which contains every mPoints of this shape

Implements A_Shape.

5.9.3.16 iDraw() [1/2]

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

Draw this shape on IHM.

Implements I_Drawable.

5.9.3.17 iDraw() [2/2]

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

Draw this shape on IHM with specific color.

Parameters

color	: Color of the shape will be draw

Implements I_Drawable.

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

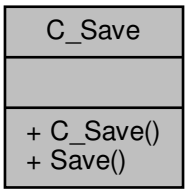
- include/shape/C_Parallelogram.hpp
- src/shape/C_Parallelogram.cpp

5.10 C_Save Class Reference

Class of the main Saver.

`#include <C_Save.hpp>`

Collaboration diagram for C_Save:



Public Member Functions

- C_Save ()
- bool Save (const std::vector< std::shared_ptr< A_Shape >> &Game)
  Save the current board as puzzle file in a page which contains less than 12 files.

5.10.1 Detailed Description

Class of the main Saver.

This class manage everything about the save

5.10.2 Constructor & Destructor Documentation

5.10.2.1 C_Save()

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

Construct an instance of a saver

5.10.3 Member Function Documentation

5.10.3.1 Save()

```
bool C_Save::Save (
            const std::vector< std::shared_ptr< A_Shape >> & Game )
```

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

Parameters

Game	: Current game

Returns

Return true if the board has been saved, false otherwise

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

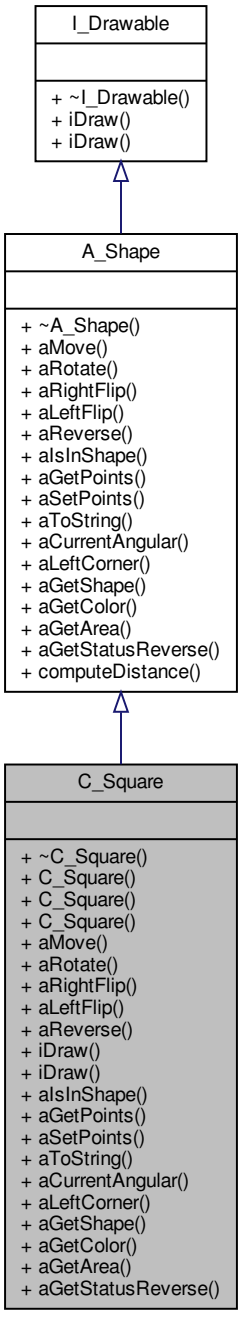
- include/parser/C_Save.hpp
- src/parser/C_Save.cpp

5.11 C_Square Class Reference

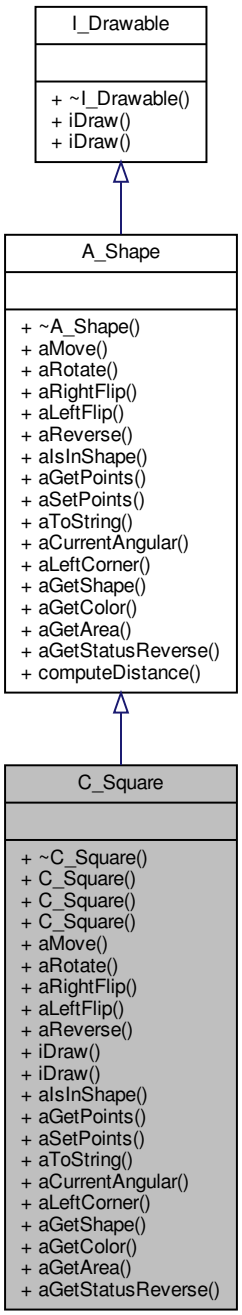
Class of the square.

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

Inheritance diagram for C_Square:



Collaboration diagram for C_Square:



Public Member Functions

- ~C_Square () override

  Destructor of C_Square.
- C_Square (MLV_Color color=MLV_COLOR_PURPLE)

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

Constructor of C_Square, requires a vector of STriangles.

- C_Square (const T_Point< double > &origin, double angular=0.0, MLV_Color color=MLV_COLOR_PUR↩ PLE)

  Constructor of C_Square, calls the deleguate Default Constructor.
- void aMove (const T_Point< double > &translation) override

  Move the C_Square by point translation.
- void aRotate (double angular) override

  Rotate the C_Square with specified angular.
- void aRightFlip () override

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

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

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

  Draw this shape on IHM.
- void iDraw (MLV_Color color) override

  Draw this shape on IHM with specific color.
- bool aIsInShape (const T_Point< double > &click) override

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

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

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

  Convert all data of C_Square in a string.
- double aCurrentAngular () override

  Get the current angular of this shape.
- T_Point< double > aLeftCorner () override

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

  Get the shape type.
- MLV_Color aGetColor () override

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

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

  Get the status of shape reversed or not.

Additional Inherited Members

5.11.1 Detailed Description

Class of the square.

This class manage everything about the C_Square

5.11.2 Constructor & Destructor Documentation

5.11.2.1 ~C_Square()

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

Destructor of C_Square.

5.11.2.2 C_Square() [1/3]

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

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

Parameters

color	: Optional __Parameter, mColor of this shape

5.11.2.3 C_Square() [2/3]

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

Constructor of C_Square, requires a vector of STriangles.

Parameters

triangle	: The C_Square will created with a vector of C_STriangle (4)
color	: Optional __Parameter, mColor of this shape

5.11.2.4 C_Square() [3/3]

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

Constructor of C_Square, calls the deleguate Default Constructor.

Parameters

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

5.11.3 Member Function Documentation

5.11.3.1 aCurrentAngular()

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

Get the current angular of this shape.

Returns

Implements A_Shape.

5.11.3.2 aGetArea()

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

Get the area of the shape.

Returns

Return the area of the shape

Implements A_Shape.

5.11.3.3 aGetColor()

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

Get the color of the shape.

Returns

Return the MLV_Color of the shape

Implements A_Shape.

5.11.3.4 aGetPoints()

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

Get mPoints of this shape.

Returns

Return a vector of mPoints of this shape

Implements A_Shape.

5.11.3.5 aGetShape()

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

Get the shape type.

Returns

Return as string the shape type

Implements A_Shape.

5.11.3.6 aGetStatusReverse()

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

Get the status of shape reversed or not.

Returns

Return true if the shape got reversed, false otherwise

Implements A_Shape.

5.11.3.7 aIsInShape()

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

Check if a point is in this shape.

Parameters

click	: T_Point to check

Returns

true if Click is in this shape, false if not

Implements A_Shape.

5.11.3.8 aLeftCorner()

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

Take the point at left top corner.

Returns

Return the point at left top corner

Implements A_Shape.

5.11.3.9 aLeftFlip()

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

Flip the figure as 45° anti clock.

Implements A_Shape.

5.11.3.10 aMove()

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

Move the C_Square by point translation.

Parameters

translation	: Every mPoints of this shape will be translate by this __Parameter

Implements A_Shape.

5.11.3.11 aReverse()

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

Reverse the figure as symmetry.

Implements A_Shape.

5.11.3.12 aRightFlip()

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

Flip the figure as 45° clock.

Implements A_Shape.

5.11.3.13 aRotate()

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

Rotate the C_Square with specified angular.

Parameters

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

Implements A_Shape.

5.11.3.14 aSetPoints()

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

Set a point to another one.

Parameters

ref	: Point to change
changed	: New value of the point

Returns

True if the ref point exists, false otherwise

Implements A_Shape.

5.11.3.15 aToString()

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

Convert all data of C_Square in a string.

Returns

Return a string which contains every mPoints of this shape

Implements A_Shape.

5.11.3.16 iDraw() [1/2]

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

Draw this shape on IHM.

Implements I_Drawable.

5.11.3.17 iDraw() [2/2]

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

Draw this shape on IHM with specific color.

Parameters

color	: color of the shape will be draw

Implements I_Drawable.

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

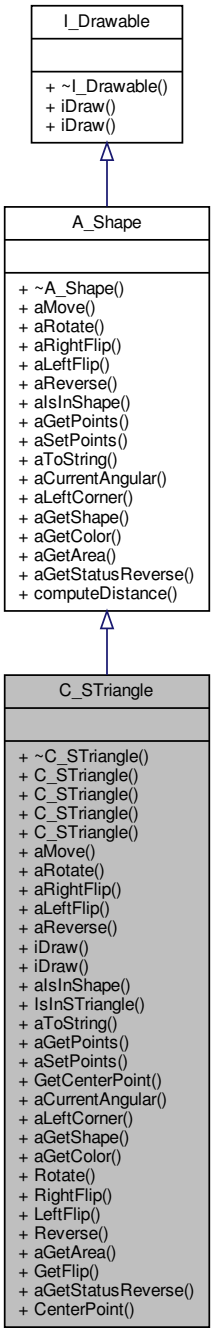
- include/shape/C_Square.hpp
- src/shape/C_Square.cpp

5.12 C_STriangle Class Reference

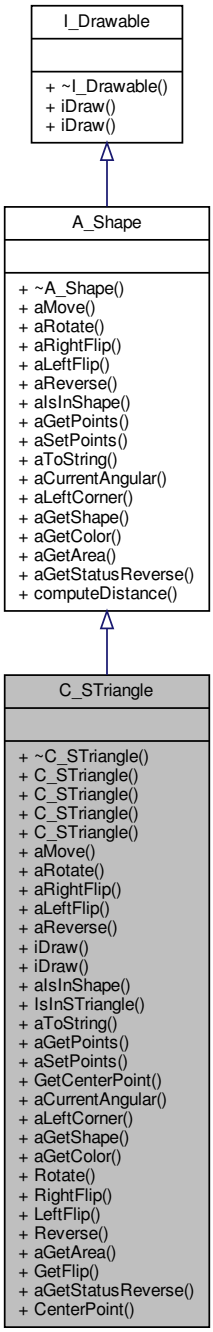
Class of the small C_STriangle.

`#include <C_STriangle.hpp>`

Inheritance diagram for C_STriangle:



Collaboration diagram for C_STriangle:



Public Member Functions

- ~C_STriangle () override

  Destructor of C_STriangle.
- C_STriangle (MLV_Color color=MLV_COLOR_GREEN)

  Constructor by default of C_MTriangle, make a C_STriangle as default.

- C_STriangle (const T_Point< double > &p1, const T_Point< double > &p2, const T_Point< double > &p3, MLV_Color color=MLV_COLOR_GREEN)

  Constructor of C_STriangle, requires 3 mPoints.
- C_STriangle (const std::vector< T_Point< double >> &points, MLV_Color color=MLV_COLOR_GREEN)

  Constructor of C_STriangle, requires a vector of 3 mPoints.
- C_STriangle (const T_Point< double > &origin, double angular=0.0, MLV_Color color=MLV_COLOR_GR↩ EEN)

  Constructor of C_STriangle, calls the deleguate Default Constructor.
- void aMove (const T_Point< double > &translation) override

  Move the C_MTriangle by point translation.
- void aRotate (double angular) override

  Rotate the C_STriangle with specified angular.
- void aRightFlip () override

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

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

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

  Draw this shape on IHM.
- void iDraw (MLV_Color color) override

  Draw this shape on IHM with specific mColor.
- bool aIsInShape (const T_Point< double > &click) override

  Check if a point is in this shape.
- bool IsInSTriangle (const T_Point< double > &click)

  Check if a point is in this C_STriangle.
- std::string aToString () override

  Convert all data of C_MTriangle in a string.
- std::vector< T_Point< double > > aGetPoints () override

  Get every mPoints of this C_STriangle.
- bool aSetPoints (const T_Point< double > &ref, const T_Point< double > &changed) override

  Set a point as same value that another point given in parameter.
- T_Point< double > GetCenterPoint () const

  Get the current center point of this C_STriangle.
- double aCurrentAngular () override

  Get the current angular of this shape.
- T_Point< double > aLeftCorner () override

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

  Get the type of shape is it.
- MLV_Color aGetColor () override

  Get the color of the shape.
- void Rotate (double angular, const T_Point< double > &center_point)

  Rotate an C_STriangle with specified angular, used only for an other shape.
- void RightFlip (const T_Point< double > &centerPoint)

  Right flip as 45°clock.
- void LeftFlip (const T_Point< double > &centerPoint)

  Right flip as 45°anti clock.
- void Reverse (const T_Point< double > &centerPoint)

  Reverse the figure as symmetry.
- double aGetArea () override

Get the area of the shape.

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

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

- bool aGetStatusReverse () const override

  Get the status of shape reversed or not.

Static Public Member Functions

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

  Compute the center point of N mPoints.

5.12.1 Detailed Description

Class of the small C_STriangle.

This class manage everything about the small C_STriangle

5.12.2 Constructor & Destructor Documentation

5.12.2.1 ~C_STriangle()

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

Destructor of C_STriangle.

5.12.2.2 C_STriangle() [1/4]

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

Constructor by default of C_MTriangle, make a C_STriangle as default.

Parameters

color	: Optional __Parameter, mColor of this shape

5.12.2.3 C_STriangle() [2/4]

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

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

Constructor of C_STriangle, requires 3 mPoints.

Parameters

p1	: First point of the C_STriangle
p2	: Second point of the C_STriangle
p3	: Third point of the C_STriangle
color	: Optional __Parameter, mColor of this shape

5.12.2.4 C_STriangle() [3/4]

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

Constructor of C_STriangle, requires a vector of 3 mPoints.

Parameters

points	: vector of 3 mPoints
color	: Optional __Parameter, mColor of this shape

5.12.2.5 C_STriangle() [4/4]

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

Constructor of C_STriangle, calls the deleguate Default Constructor.

Parameters

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

5.12.3 Member Function Documentation

5.12.3.1 aCurrentAngular()

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

Get the current angular of this shape.

Returns

Return the current angular in double

Implements A_Shape.

5.12.3.2 aGetArea()

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

Get the area of the shape.

Returns

Return the area of this shape as a double

Implements A_Shape.

5.12.3.3 aGetColor()

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

Get the color of the shape.

Returns

Return the MLV_Color of the shape

Implements A_Shape.

5.12.3.4 aGetPoints()

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

Get every mPoints of this C_STriangle.

Returns

Return a vector of these mPoints

Implements A_Shape.

5.12.3.5 aGetShape()

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

Get the type of shape is it.

Returns

Return as string the type of shape is it

Implements A_Shape.

5.12.3.6 aGetStatusReverse()

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

Get the status of shape reversed or not.

Returns

Return true if the shape got reversed, false otherwise

Implements A_Shape.

5.12.3.7 aIsInShape()

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

Check if a point is in this shape.

Parameters

click	: T_Point to check

Returns

true if Click is in this shape, false if not

Implements A_Shape.

5.12.3.8 aLeftCorner()

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

Take the point at left top corner.

Returns

Return the point at left top corner

Implements A_Shape.

5.12.3.9 aLeftFlip()

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

Flip the figure as 45° anti clock.

Implements A_Shape.

5.12.3.10 aMove()

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

Move the C_MTriangle by point translation.

Parameters

translation	: Every mPoints of this shape will be translate by this __Parameter

Implements A_Shape.

5.12.3.11 aReverse()

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

Reverse the figure as symmetry.

Implements A_Shape.

5.12.3.12 aRightFlip()

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

Flip the figure as 45° clock.

Implements A_Shape.

5.12.3.13 aRotate()

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

Rotate the C_STriangle with specified angular.

Parameters

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

Implements A_Shape.

5.12.3.14 aSetPoints()

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

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

Parameters

ref	Point we want to set
changed	The ref Point will take same value as this one

Returns

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

Implements A_Shape.

5.12.3.15 aToString()

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

Convert all data of C_MTriangle in a string.

Returns

Return a string which contains every mPoints of this shape

Implements A_Shape.

5.12.3.16 CenterPoint()

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

Compute the center point of N mPoints.

Parameters

list_points	: vector of N mPoints

Returns

Return the center point of these N mPoints

5.12.3.17 GetCenterPoint()

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

Get the current center point of this C_STriangle.

Returns

Return the current center point of this C_STriangle

5.12.3.18 GetFlip()

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

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

Returns

Return a vector of point needed to flip the figure

5.12.3.19 iDraw() [1/2]

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

Draw this shape on IHM.

Implements I_Drawable.

5.12.3.20 iDraw() [2/2]

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

Draw this shape on IHM with specific mColor.

Parameters

Color	: Color from the graphic library MLV like MLV_COLOR_XXX

Implements I_Drawable.

5.12.3.21 IsInSTriangle()

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

Check if a point is in this C_STriangle.

Parameters

click	: T_Point to check

Returns

true if Click is in this shape, false if not

5.12.3.22 LeftFlip()

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

Right flip as 45° anti clock.

Parameters

centerPoint	: flip the figure about this center point

5.12.3.23 Reverse()

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

Reverse the figure as symmetry.

Parameters

centerPoint	: Reverse the figure as symmetry about this center point

5.12.3.24 RightFlip()

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

Right flip as 45° clock.

Parameters

centerPoint	: flip the figure about this center point

5.12.3.25 Rotate()

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

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

Parameters

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

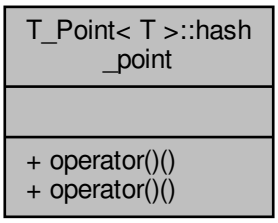
The documentation for this class was generated from the following files:

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

5.13 T_Point< T >::hash_point Struct Reference

`#include <T_Point.hpp>`

Collaboration diagram for T_Point< T >::hash_point:



Public Member Functions

- std::size_t operator() (const T_Point< T > &p) const

  Operator to hash a point.
- bool operator() (const T_Point< T > &p1, const T_Point< T > &p2) const

  Operator equal need to hash a point.

5.13.1 Member Function Documentation

5.13.1.1 operator()() [1/2]

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

Operator to hash a point.

Parameters

p	: point to hash

Returns

Return the hash of the point

5.13.1.2 operator()() [2/2]

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

Operator equal need to hash a point.

Parameters

p1	: Point 1
p2	: Point 2

Returns

Return true if p1 and p2 are equals, false otherwise

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

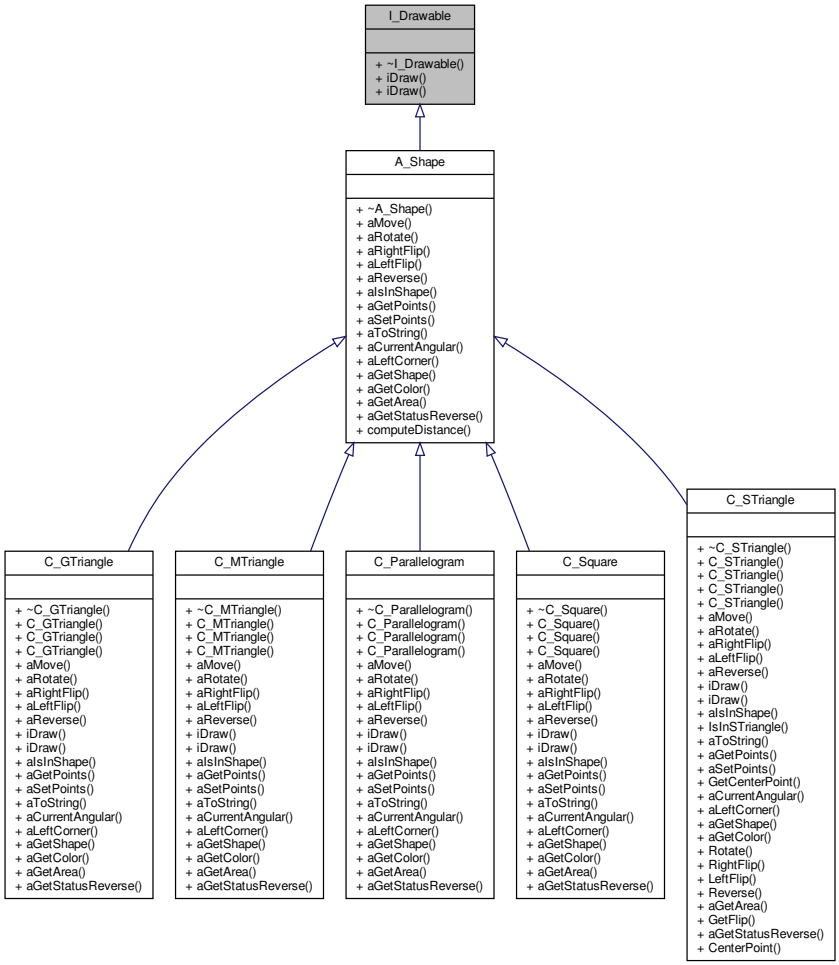
- include/utils/T_Point.hpp

5.14 I_Drawable Class Reference

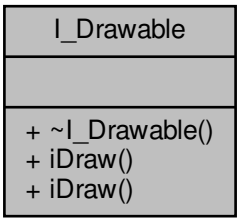
I_Drawable is everything to iDraw.

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

Inheritance diagram for I_Drawable:



Collaboration diagram for I_Drawable:



Public Member Functions

- ~I_Drawable ()=default

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

- virtual void iDraw ()=0
- virtual void iDraw (MLV_Color color)=0

5.14.1 Detailed Description

I_Drawable is everything to iDraw.

This class manage everything drawing

5.14.2 Constructor & Destructor Documentation

5.14.2.1 ~I_Drawable()

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

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

5.14.3 Member Function Documentation

5.14.3.1 iDraw() [1/2]

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

Implemented in C_STriangle, C_Parallelogram, C_MTriangle, C_Square, and C_GTriangle.

5.14.3.2 iDraw() [2/2]

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

Implemented in C_STriangle, C_Parallelogram, C_MTriangle, C_Square, and C_GTriangle.

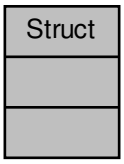
The documentation for this class was generated from the following file:

- include/drawable/I_Drawable.h

5.15 Struct Struct Reference

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

Collaboration diagram for Struct:



5.15.1 Detailed Description

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

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

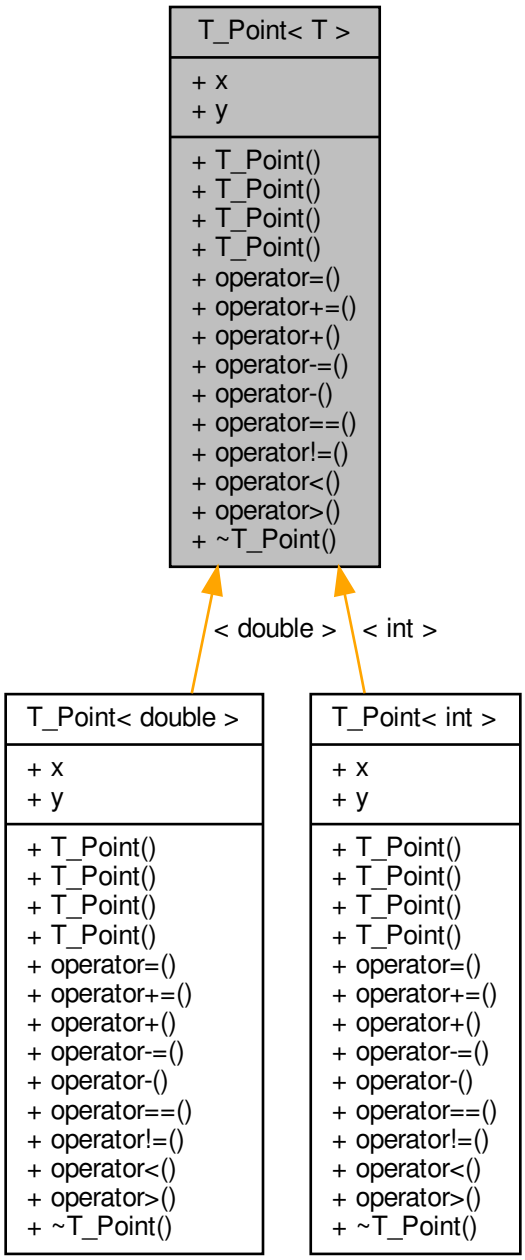
- include/utils/T_Point.hpp

5.16 T_Point< T > Class Template Reference

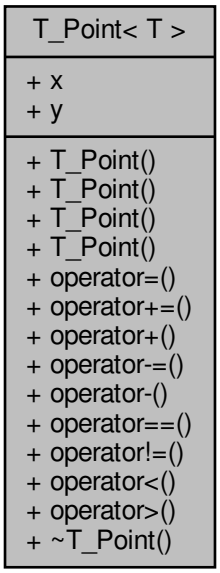
Class of a T_Point.

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

Inheritance diagram for T_Point< T >:



Collaboration diagram for T_Point< T >:



Classes

- struct hash_point

Public Member Functions

- constexpr T_Point (const T_Point< T > &p)=default
- T_Point ()

  Constructor for a point with initialisation list.
- T_Point (const T_Point< T > &&p) noexcept

  Constructor of a point with move semantic.
- T_Point (const T &_x, const T &_y)

  Constructor for a point. Requires a X and a Y coordinate.
- T_Point & operator= (const T_Point< T > &p)

  Operator = of a point.
- T_Point & operator+= (const T_Point< T > &p)

  Operator +=.
- T_Point operator+ (const T_Point< T > &p)

  Operator +.
- T_Point & operator-= (const T_Point< T > &p)

  Operator -=.
- T_Point operator- (const T_Point< T > &p)

  Operator -.
- bool operator== (const T_Point< T > &p) const

Operator == of a point.

- bool operator!= (const T_Point< T > &p) const

  Operator != of a point.

- bool operator< (const T_Point< T > &p) const

  Operator < of a point.

- bool operator> (const T_Point< T > &p) const

  Operator > of a point.

- ~T_Point ()=default

Public Attributes

- T x
- T y

5.16.1 Detailed Description

template<typename T>
class T_Point< T >

Class of a T_Point.

Template Parameters

T	: Template parameter This class manage everything about a point

5.16.2 Constructor & Destructor Documentation

5.16.2.1 T_Point() [1/4]

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

5.16.2.2 T_Point() [2/4]

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

Constructor for a point with initialisation list.

5.16.2.3 T_Point() [3/4]

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

Constructor of a point with move semantic.

Parameters

p	: Point to move

5.16.2.4 T_Point() [4/4]

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

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

Parameters

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

5.16.2.5 ~T_Point()

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

5.16.3 Member Function Documentation

5.16.3.1 operator"!=()

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

Operator != of a point.

Parameters

p	: T_Point to compare

Returns

Return True if the point is different, false if not

5.16.3.2 operator+()

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

Operator +.

Parameters

p	: Point

Returns

Return the behavior when a point is add by another one

5.16.3.3 operator+=()

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

Operator +=.

Parameters

p	: Point

Returns

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

5.16.3.4 operator-()

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

Operator -.

Parameters

p	: Point

Returns

Return the behavior when a point is subtract by another one

5.16.3.5 operator-=()

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

Operator -=.

Parameters

p	: Point

Returns

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

5.16.3.6 operator<()

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

Operator < of a point.

Parameters

p	: T_Point to compare

Returns

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

5.16.3.7 operator=()

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

Operator = of a point.

Parameters

p	: T_Point to "copy"

Returns

Return a reference to an atomic point

5.16.3.8 operator==()

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

Operator == of a point.

Parameters

p	: T_Point to compare

Returns

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

5.16.3.9 operator>()

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

Operator > of a point.

Parameters

p	: T_Point to comapre

Returns

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

5.16.4 Member Data Documentation

5.16.4.1 x

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

Template x for a point

5.16.4.2 y

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

Template y for a point

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

- include/utils/T_Point.hpp

Chapter 6

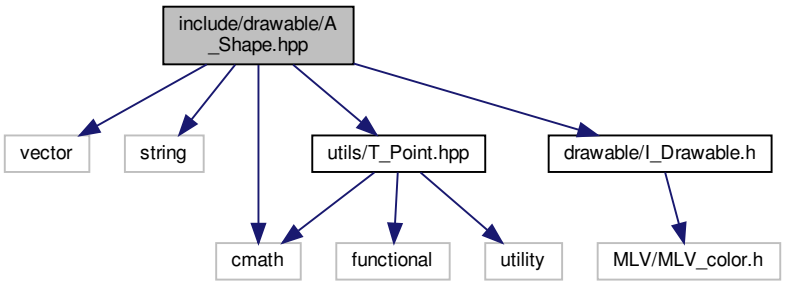
File Documentation

6.1 include/drawable/A_Shape.hpp File Reference

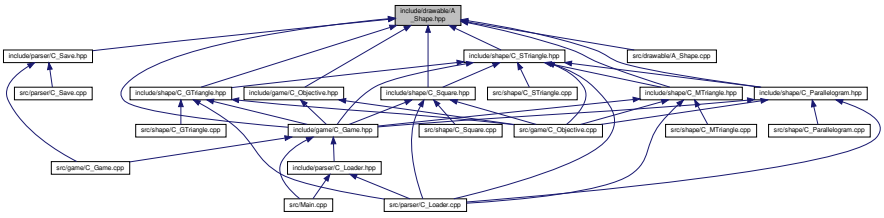
Abstract Class A_Shape of every shape in Tangram.

```
#include <vector>
#include <string>
#include <cmath>
#include <utils/T_Point.hpp>
#include <drawable/I_Drawable.h>
```

Include dependency graph for A_Shape.hpp:



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



Classes

- class A_Shape

  Abstract Class of every A_Shape.

6.1.1 Detailed Description

Abstract Class A_Shape of every shape in Tangram.

Author

Jérémie LE BASTARD

Version

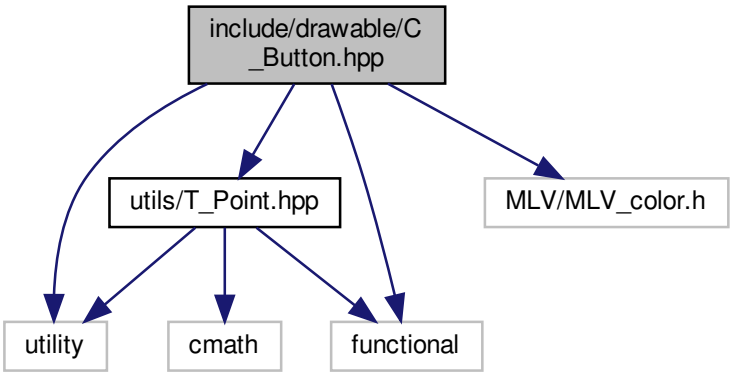
1.0

6.2 include/drawable/C_Button.hpp File Reference

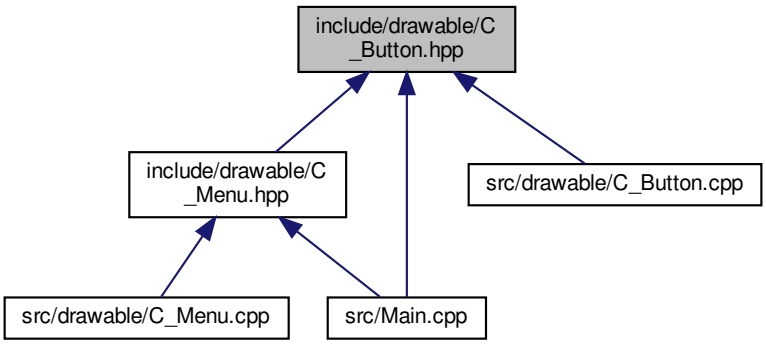
Every mButtons of menu.

```
#include <utility>
#include <utils/T_Point.hpp>
#include <functional>
#include <MLV/MLV_color.h>
```

Include dependency graph for C_Button.hpp:



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



Classes

- class C_Button

  C_Button of the C_Menu.

6.2.1 Detailed Description

Every mButtons of menu.

Author

Jérémie LE BASTARD

Version

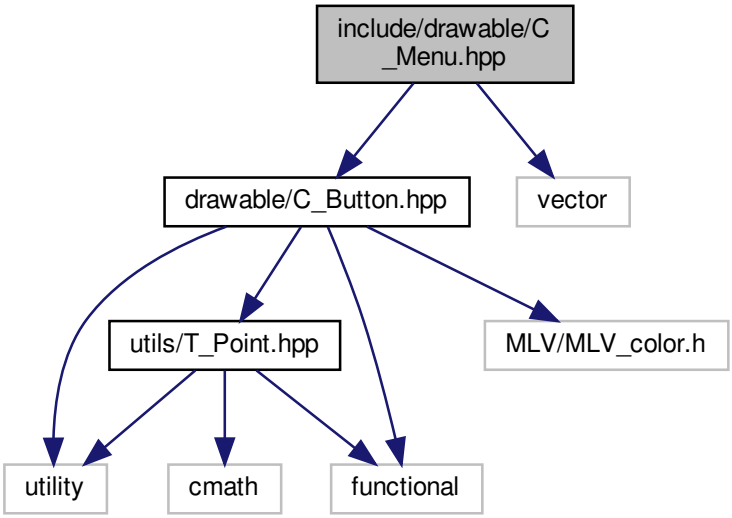
1.0

6.3 include/drawable/C_Menu.hpp File Reference

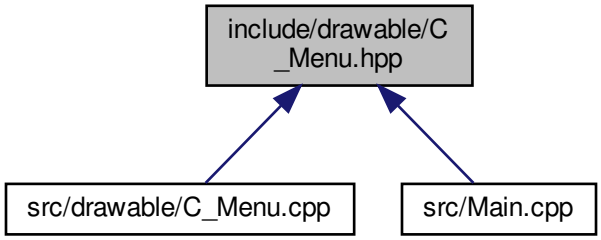
C_Menu of the Tangram's C_Game.

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

Include dependency graph for C_Menu.hpp:



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



Classes

- class C_Menu

  C_Menu of the game.

6.3.1 Detailed Description

C_Menu of the Tangram's C_Game.

Author

Jérémie LE BASTARD

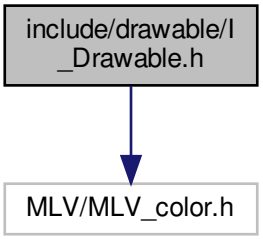
Version

1.0

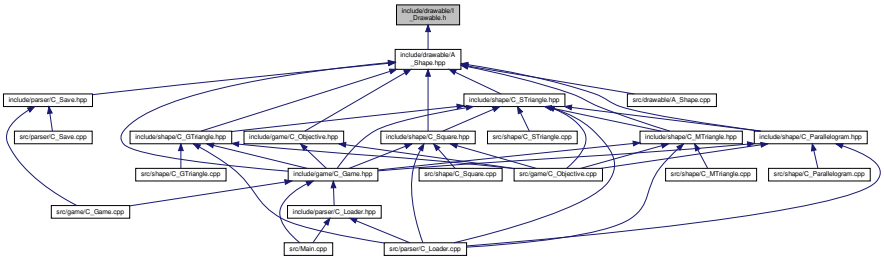
6.4 include/drawable/I_Drawable.h File Reference

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

Include dependency graph for I_Drawable.h:



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



Classes

- class I_Drawable

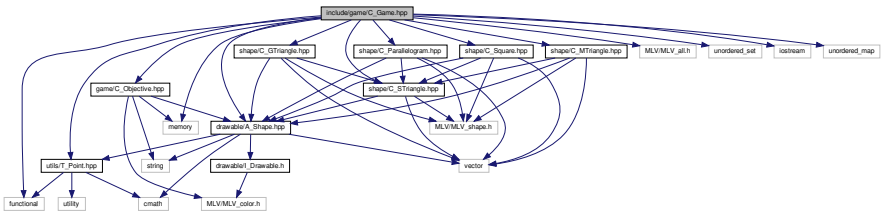
  I_Drawable is everything to iDraw.

6.5 include/game/C_Game.hpp File Reference

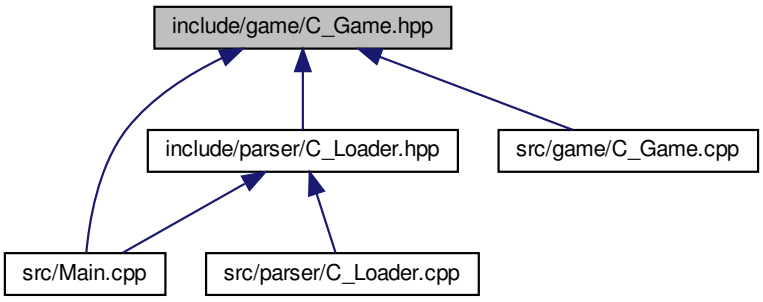
Main C_Game of the Tangram.

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

Include dependency graph for C_Game.hpp:



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



Classes

- class C_Game

  Class of the main C_Game.

6.5.1 Detailed Description

Main C_Game of the Tangram.

Author

Jérémie LE BASTARD

Version

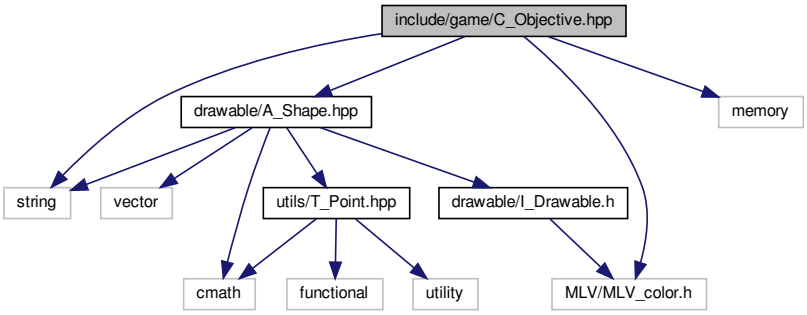
1.0

6.6 include/game/C_Objective.hpp File Reference

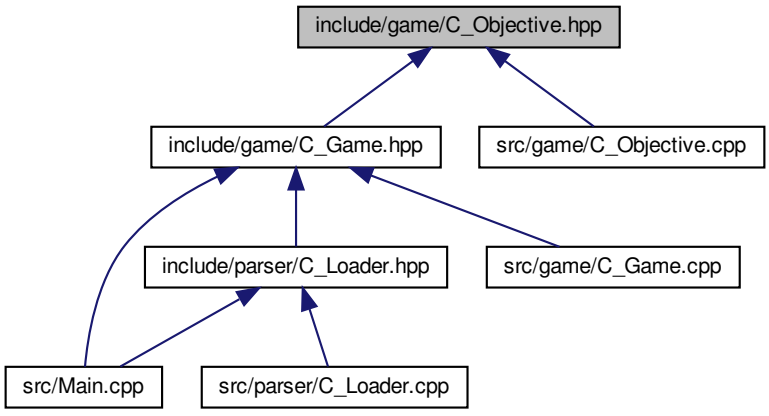
C_Objective of the Tangram's board.

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

Include dependency graph for C_Objective.hpp:



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



Classes

- class C_Objective

  Class of the board C_Objective.

6.6.1 Detailed Description

C_Objective of the Tangram's board.

Author

Jérémie LE BASTARD

Version

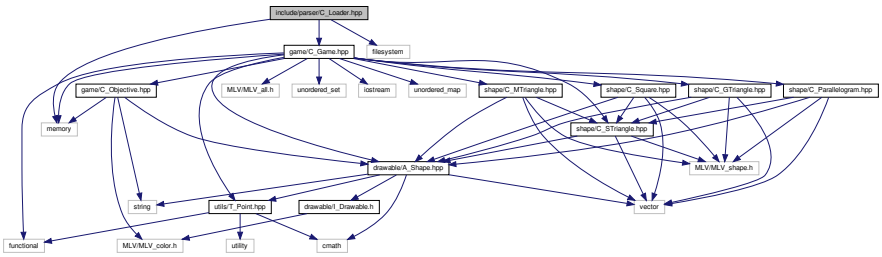
1.0

6.7 include/parser/C_Loader.hpp File Reference

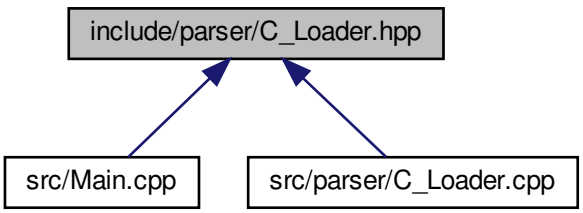
Load a board of Tangram.

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

Include dependency graph for C_Loader.hpp:



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



Classes

- class C_Loader

  Class of the main C_Loader.

6.7.1 Detailed Description

Load a board of Tangram.

Author

Jérémie LE BASTARD

Version

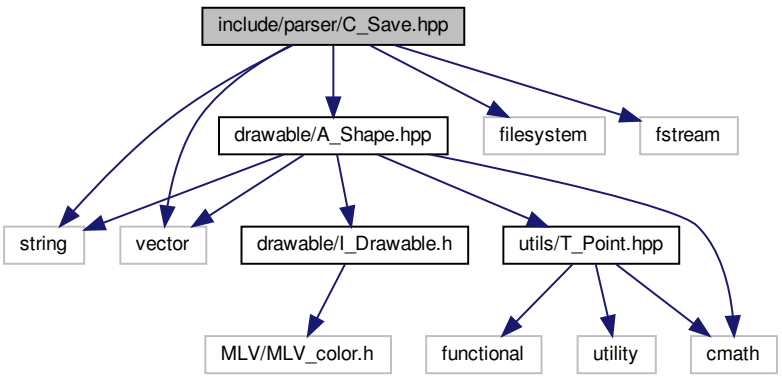
1.0

6.8 include/parser/C_Save.hpp File Reference

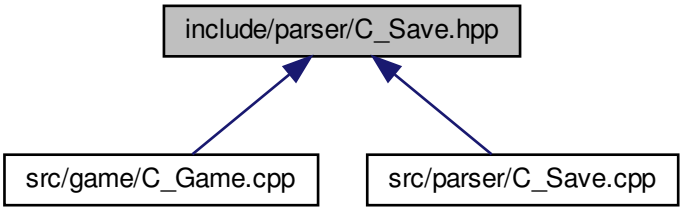
C_Save a board of Tangram.

```
#include <string>
#include <filesystem>
#include <vector>
#include <fstream>
#include <drawable/A_Shape.hpp>
```

Include dependency graph for C_Save.hpp:



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



Classes

- class C_Save

  Class of the main Saver.

6.8.1 Detailed Description

C_Save a board of Tangram.

Author

Jérémie LE BASTARD

Version

1.0

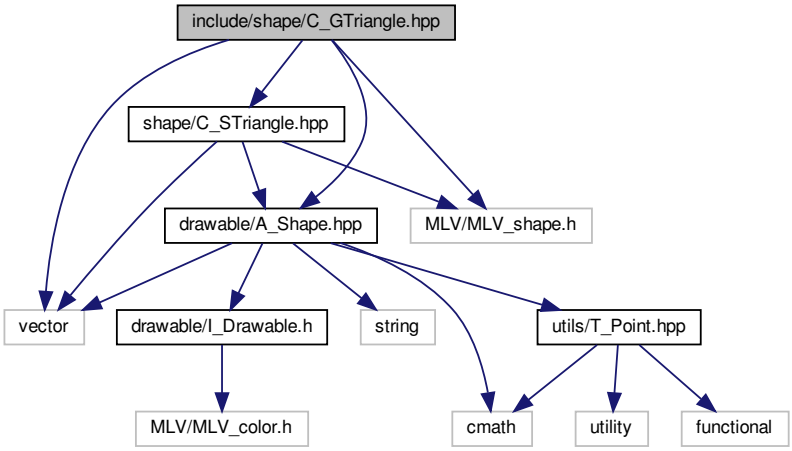
6.9 include/shape/C_GTriangle.hpp File Reference

A_Shape of Great Triangle.

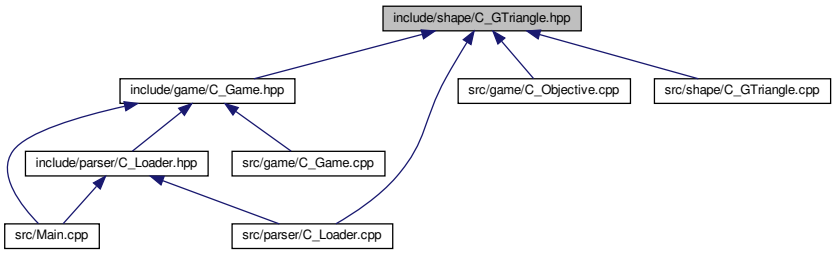
```
#include <vector>
#include <shape/C_STriangle.hpp>
#include <drawable/A_Shape.hpp>
```

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

Include dependency graph for C_GTriangle.hpp:



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



Classes

- class C_GTriangle

  Class of the greatest C_GTriangle.

6.9.1 Detailed Description

A_Shape of Great Triangle.

Author

Jérémie LE BASTARD

Version

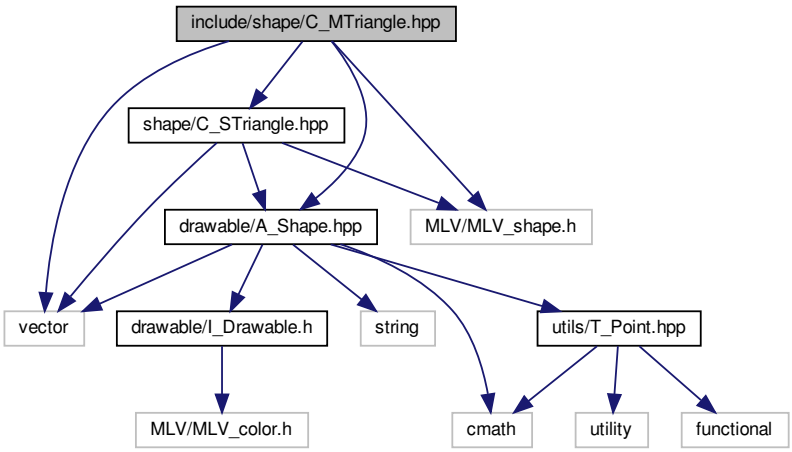
1.0

6.10 include/shape/C_MTriangle.hpp File Reference

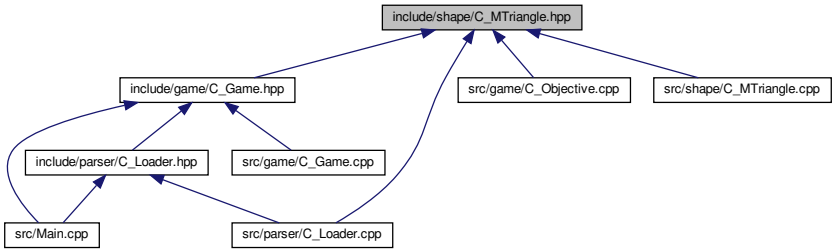
A_Shape of Medium Triangle.

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

Include dependency graph for C_MTriangle.hpp:



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



Classes

- class C_MTriangle

  Class of the medium C_MTriangle.

6.10.1 Detailed Description

A_Shape of Medium Triangle.

Author

Jérémie LE BASTARD

Version

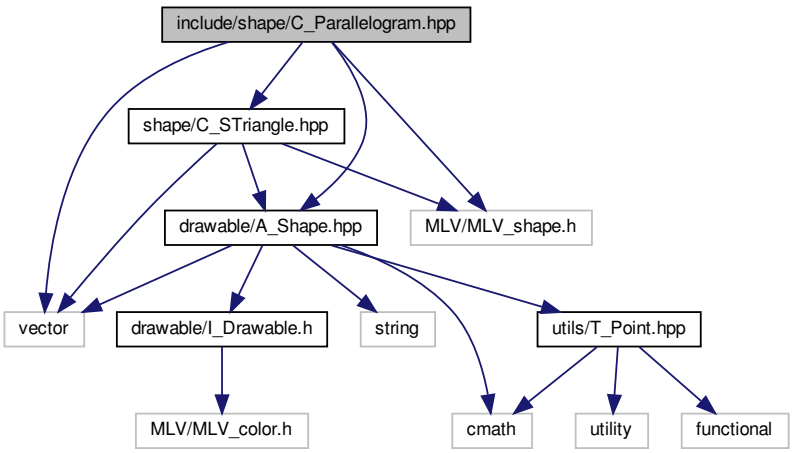
1.0

6.11 include/shape/C_Parallelogram.hpp File Reference

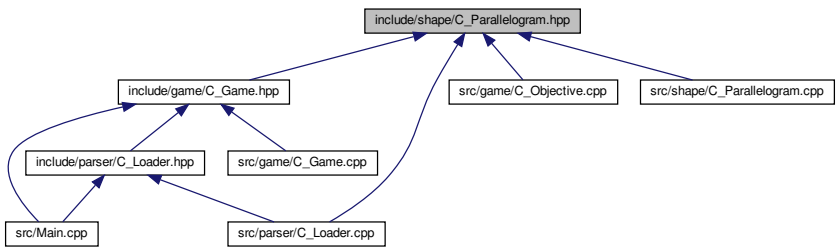
A_Shape of C_Parallelogram.

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

Include dependency graph for C_Parallelogram.hpp:



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



Classes

- class C_Parallelogram

  Class of the parallelogram.

6.11.1 Detailed Description

A_Shape of C_Parallelogram.

Author

Jérémie LE BASTARD

Version

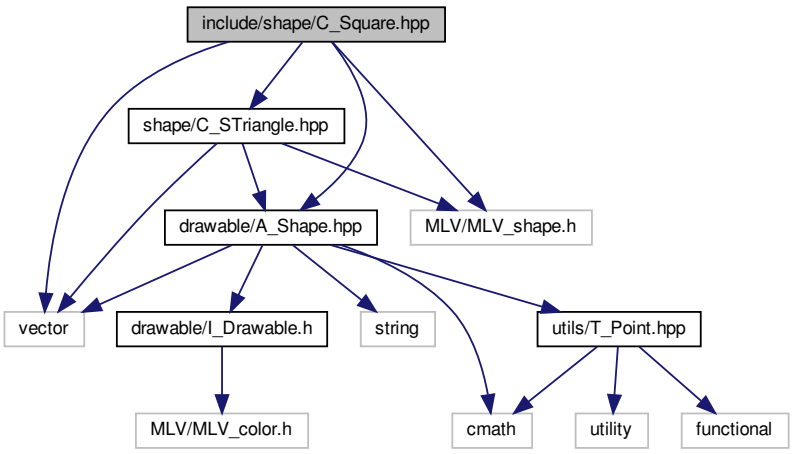
1.0

6.12 include/shape/C_Square.hpp File Reference

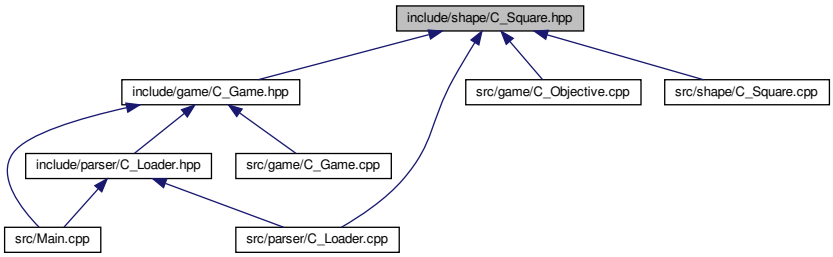
A_Shape of C_Square.

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

Include dependency graph for C_Square.hpp:



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



Classes

- class C_Square

  Class of the square.

6.12.1 Detailed Description

A_Shape of C_Square.

Author

Jérémie LE BASTARD

Version

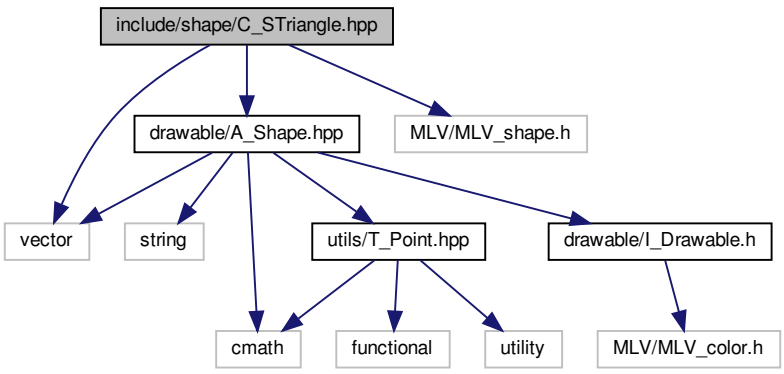
1.0

6.13 include/shape/C_STriangle.hpp File Reference

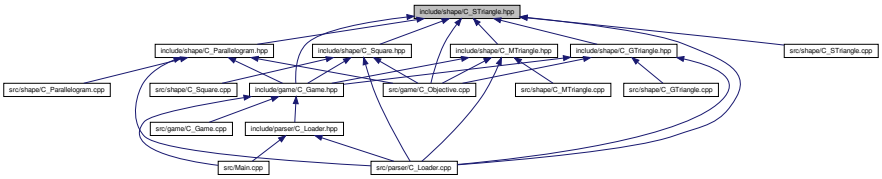
A_Shape of Small Triangle.

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

Include dependency graph for C_STriangle.hpp:



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



Classes

- class C_STriangle

  Class of the small C_STriangle.

6.13.1 Detailed Description

A_Shape of Small Triangle.

Author

Jérémie LE BASTARD

Version

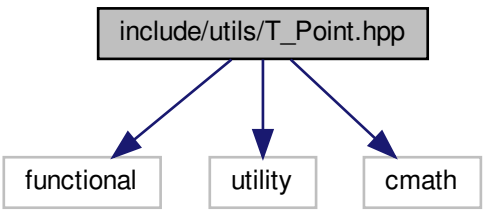
1.0

6.14 include/utils/T_Point.hpp File Reference

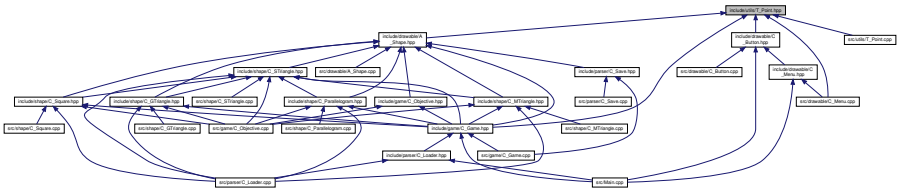
T_Point for every shape and menu.

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

Include dependency graph for T_Point.hpp:



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



Classes

- class T_Point< T >

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

6.14.1 Detailed Description

T_Point for every shape and menu.

Author

Jérémie LE BASTARD

Version

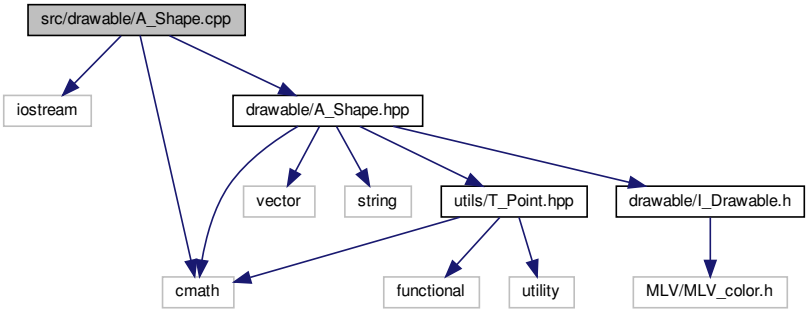
1.0

6.15 README.md File Reference

6.16 src/drawable/A_Shape.cpp File Reference

```
#include <iostream>
#include <cmath>
#include <drawable/A_Shape.hpp>
```

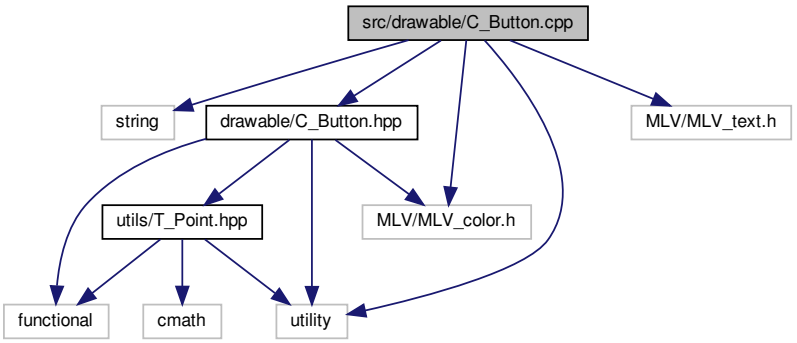
Include dependency graph for A_Shape.cpp:



6.17 src/drawable/C_Button.cpp File Reference

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

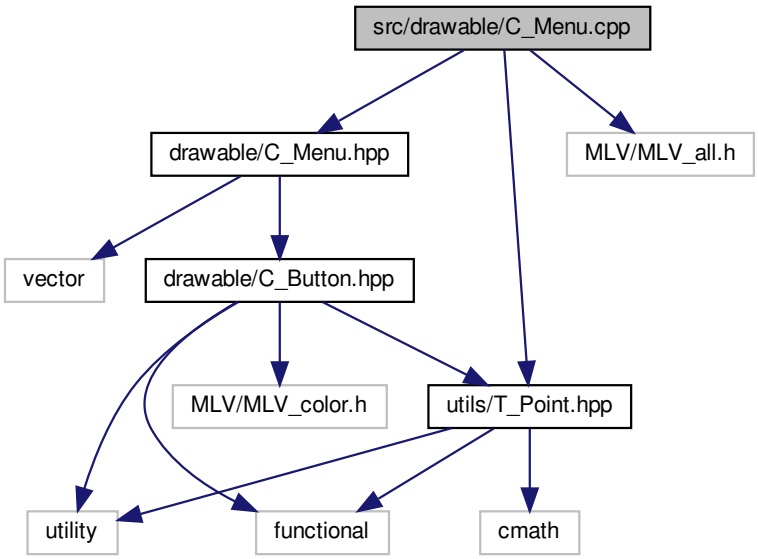
Include dependency graph for C_Button.cpp:



6.18 src/drawable/C_Menu.cpp File Reference

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

Include dependency graph for C_Menu.cpp:

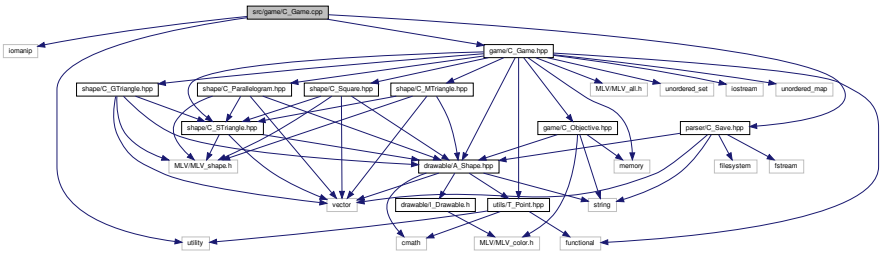


6.19 src/drawable/I_Drawable.cpp File Reference

6.20 src/game/C_Game.cpp File Reference

```
#include <iomanip>
#include <utility>
#include <game/C_Game.hpp>
#include <parser/C_Save.hpp>
```

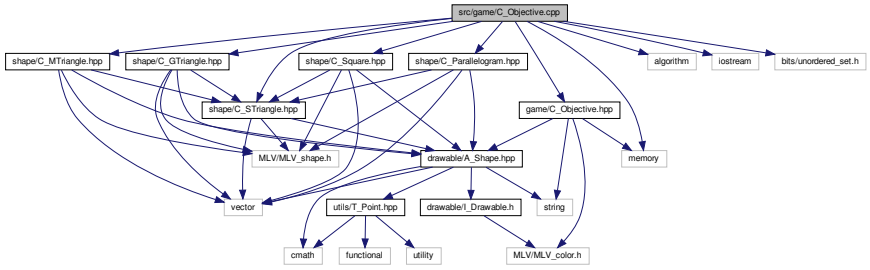
Include dependency graph for C_Game.cpp:



6.21 src/game/C_Objective.cpp File Reference

```
#include <game/C_Objective.hpp>
#include <shape/C_STriangle.hpp>
#include <shape/C_MTriangle.hpp>
#include <shape/C_GTriangle.hpp>
#include <shape/C_Parallelogram.hpp>
#include <shape/C_Square.hpp>
#include <algorithm>
#include <iostream>
#include <memory>
#include <bits/unordered_set.h>
```

Include dependency graph for C_Objective.cpp:

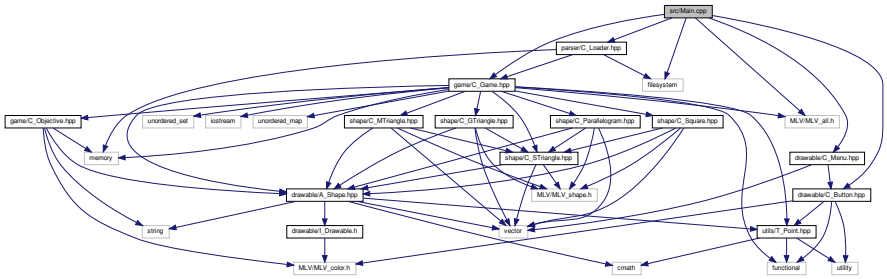


6.22 src/Main.cpp File Reference

```
#include <parser/C_Loader.hpp>
#include <drawable/C_Menu.hpp>
```

```
#include <drawable/C_Button.hpp>
#include <MLV/MLV_all.h>
#include <game/C_Game.hpp>
#include <filesystem>
```

Include dependency graph for Main.cpp:



Functions

- int main (int argc, char ∗argv[ ])

Variables

- int page = 1

6.22.1 Function Documentation

6.22.1.1 main()

```
int main (
              int argc,
              char * argv[] )
```

6.22.2 Variable Documentation

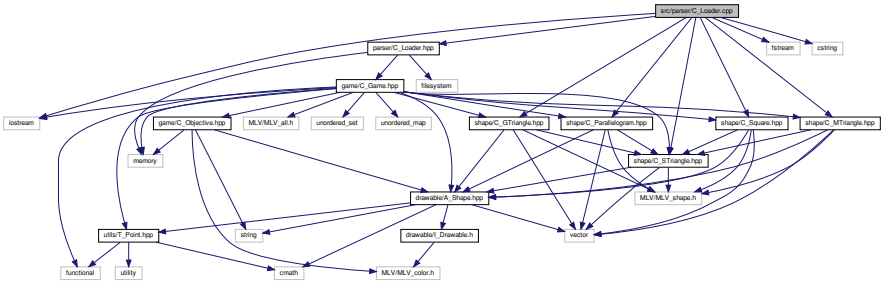
6.22.2.1 page

```
int page = 1
```

6.23 src/parser/C_Loader.cpp File Reference

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

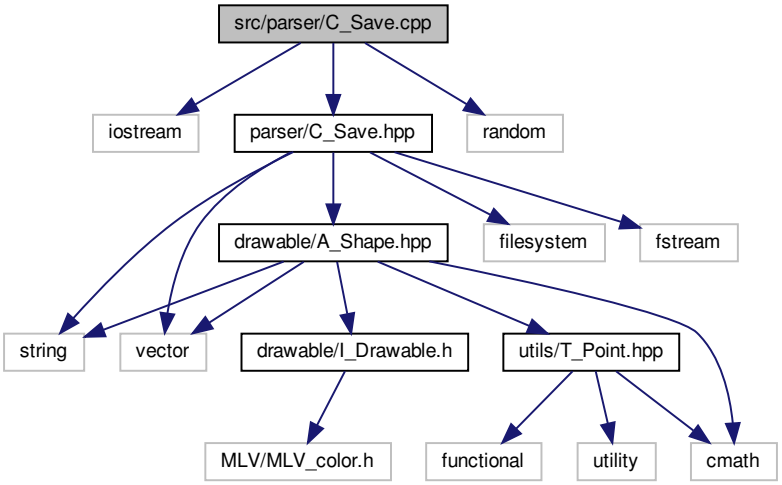
Include dependency graph for C_Loader.cpp:



6.24 src/parser/C_Save.cpp File Reference

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

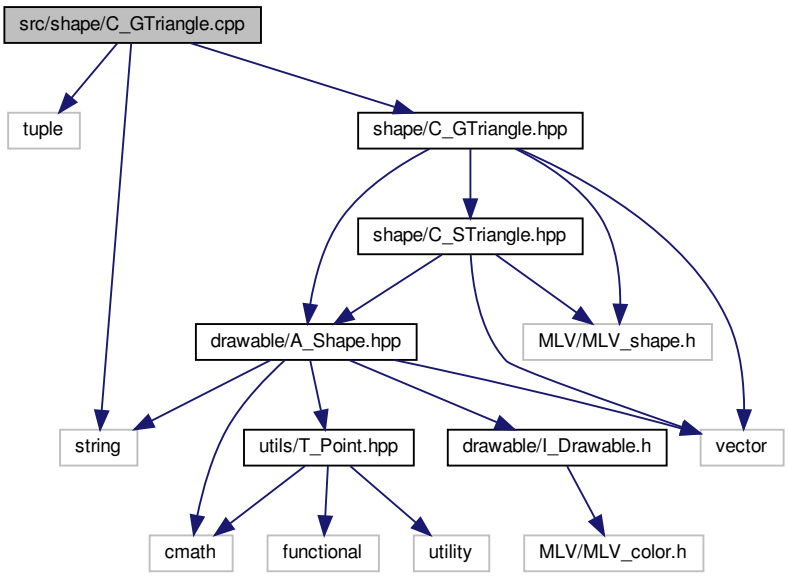
Include dependency graph for C_Save.cpp:



6.25 src/shape/C_GTriangle.cpp File Reference

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

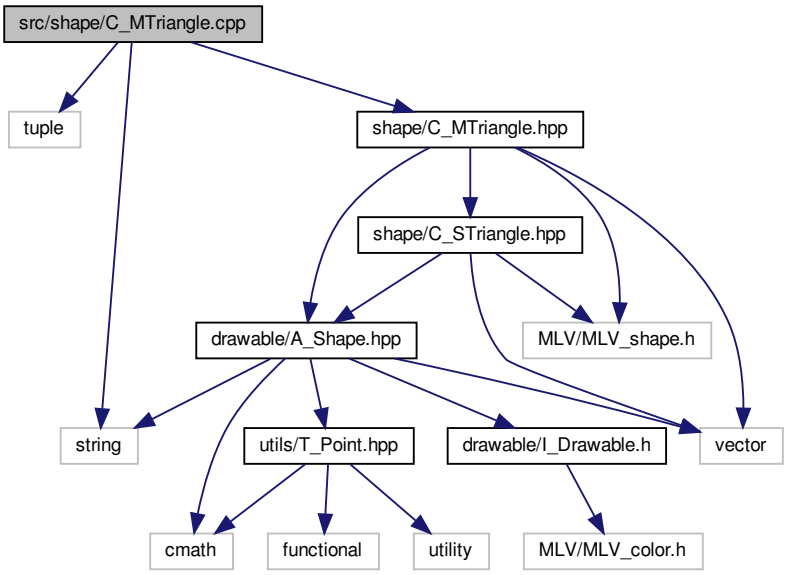
Include dependency graph for C_GTriangle.cpp:



6.26 src/shape/C_MTriangle.cpp File Reference

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

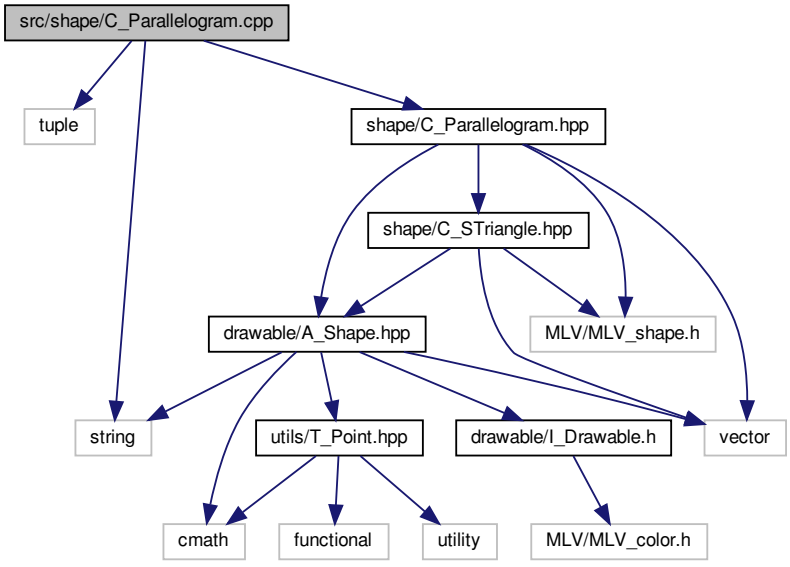
Include dependency graph for C_MTriangle.cpp:



6.27 src/shape/C_Parallelogram.cpp File Reference

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

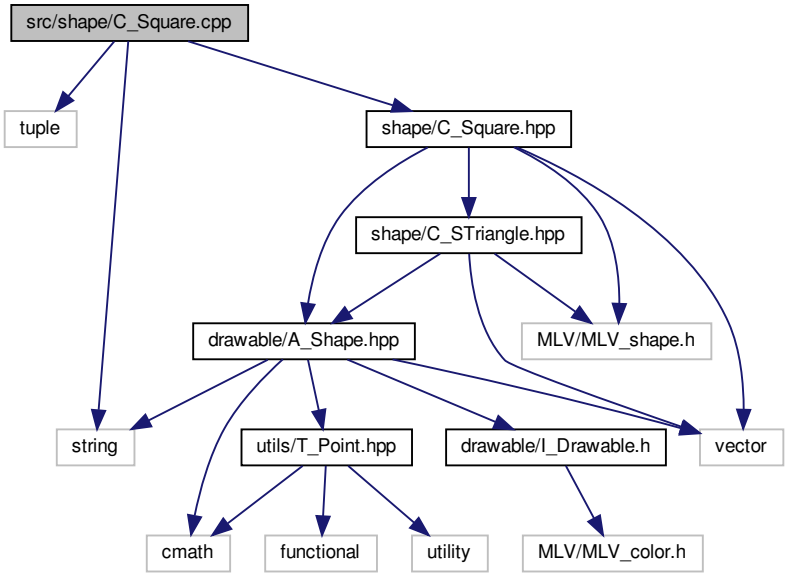
Include dependency graph for C_Parallelogram.cpp:



6.28 src/shape/C_Square.cpp File Reference

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

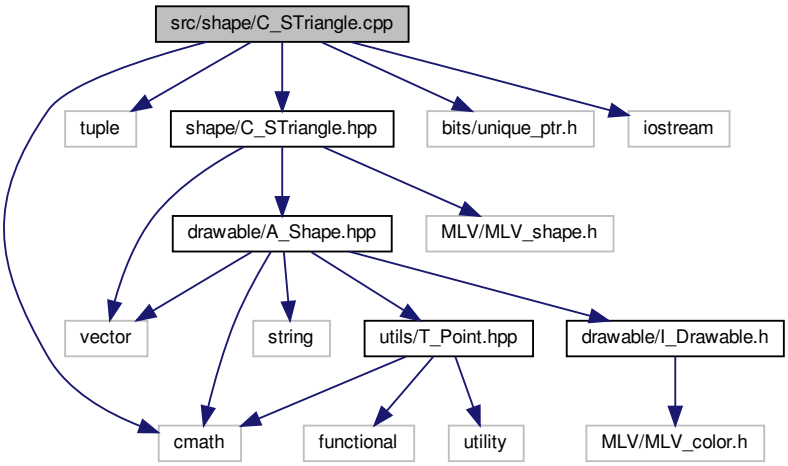
Include dependency graph for C_Square.cpp:



6.29 src/shape/C_STriangle.cpp File Reference

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

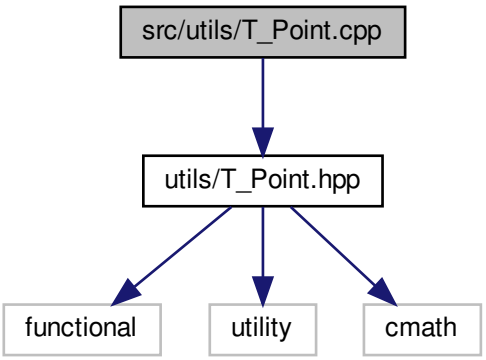
Include dependency graph for C_STriangle.cpp:



6.30 src/utils/T_Point.cpp File Reference

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

Include dependency graph for T_Point.cpp:



Index