

## TP LO11 Example

Generated by Doxygen 1.9.5



<b>1 LO21 - Akropolis Project</b>	<b>1</b>
1.1 Introduction	1
1.2 – Documentation –	1
<b>2 LO21-Projet</b>	<b>3</b>
2.1 Méthodologie	3
<b>3 Hierarchical Index</b>	<b>5</b>
3.1 Class Hierarchy	5
<b>4 Class Index</b>	<b>7</b>
4.1 Class List	7
<b>5 File Index</b>	<b>9</b>
5.1 File List	9
<b>6 Class Documentation</b>	<b>11</b>
6.1 Affichage Class Reference	11
6.1.1 Member Function Documentation	11
6.1.1.1 affiche_plateau_actuel()	11
6.2 AffichageConsole Class Reference	12
6.2.1 Constructor & Destructor Documentation	12
6.2.1.1 AffichageConsole()	12
6.2.2 Member Function Documentation	12
6.2.2.1 affiche_plateau_actuel()	13
6.2.2.2 getInstance()	13
6.2.3 Member Data Documentation	13
6.2.3.1 hexH	13
6.2.3.2 hexW	13
6.2.3.3 instance	13
6.3 Carriere Class Reference	13
6.3.1 Detailed Description	14
6.4 Chantier Class Reference	14
6.4.1 Member Function Documentation	14
6.4.1.1 ajouter_tuile() [1/2]	15
6.4.1.2 ajouter_tuile() [2/2]	15
6.4.1.3 get_nombre_tuiles()	15
6.4.1.4 get_taille()	15
6.4.1.5 get_tuiles()	15
6.4.1.6 prendre_tuile()	15
6.4.1.7 set_nombre_joueurs()	16
6.4.1.8 set_taille()	16
6.4.2 Member Data Documentation	16
6.4.2.1 nombreTuiles	16

6.4.2.2 taille	16
6.4.2.3 tuiles	16
6.5 Deck Class Reference	16
6.5.1 Constructor & Destructor Documentation	17
6.5.1.1 Deck()	17
6.5.2 Member Function Documentation	17
6.5.2.1 get_nombre_tuiles()	17
6.5.2.2 get_taille()	17
6.5.2.3 tirer_tuile() [1/2]	18
6.5.2.4 tirer_tuile() [2/2]	18
6.5.3 Member Data Documentation	18
6.5.3.1 nombreTuiles	18
6.5.3.2 taille	18
6.5.3.3 tuiles	18
6.6 Hexagone Class Reference	18
6.6.1 Detailed Description	19
6.6.2 Constructor & Destructor Documentation	19
6.6.2.1 Hexagone()	19
6.6.2.2 ~Hexagone()	19
6.6.3 Member Function Documentation	20
6.6.3.1 get_couleur()	20
6.6.3.2 get_displayed_text()	20
6.6.3.3 get_hauteur()	20
6.6.3.4 get_local_position()	20
6.6.3.5 get_tuile()	20
6.6.3.6 peut_etre_placee()	20
6.6.3.7 quand_recouvert()	21
6.6.4 Member Data Documentation	21
6.6.4.1 couleur	21
6.6.4.2 indice_tuile	21
6.6.4.3 tuileParent	21
6.7 IllustreArchitecte Class Reference	22
6.7.1 Member Function Documentation	22
6.7.1.1 choisir_tuile()	22
6.7.1.2 get_niveau()	23
6.7.1.3 jouer()	23
6.7.1.4 set_niveau()	23
6.7.1.5 trouver_emplacement_tuile()	23
6.7.2 Member Data Documentation	23
6.7.2.1 joueToutSeul	23
6.7.2.2 niveau	23
6.7.2.3 score	23

6.8 Joueur Class Reference . . . . .	24
6.8.1 Detailed Description . . . . .	24
6.8.2 Member Function Documentation . . . . .	24
6.8.2.1 ajouter_pierre() . . . . .	25
6.8.2.2 get_pierre() . . . . .	25
6.8.2.3 get_plateau() . . . . .	25
6.8.2.4 get_score() . . . . .	25
6.8.2.5 jouer() . . . . .	25
6.8.2.6 place_tuile() . . . . .	25
6.8.2.7 set_pierre() . . . . .	26
6.8.3 Member Data Documentation . . . . .	26
6.8.3.1 joueToutSeul . . . . .	26
6.8.3.2 pierre . . . . .	26
6.8.3.3 plateauJoueur . . . . .	26
6.9 JoueurSimple Class Reference . . . . .	26
6.9.1 Detailed Description . . . . .	27
6.9.2 Member Function Documentation . . . . .	27
6.9.2.1 get_score() . . . . .	27
6.9.3 Member Data Documentation . . . . .	27
6.9.3.1 joueToutSeul . . . . .	27
6.9.3.2 scoreJoueur . . . . .	27
6.10 Place Class Reference . . . . .	28
6.10.1 Detailed Description . . . . .	28
6.10.2 Constructor & Destructor Documentation . . . . .	28
6.10.2.1 Place() . . . . .	28
6.10.2.2 ~Place() . . . . .	28
6.10.3 Member Function Documentation . . . . .	29
6.10.3.1 get_etoiles() . . . . .	29
6.10.4 Member Data Documentation . . . . .	29
6.10.4.1 etoiles . . . . .	29
6.11 Plateau Class Reference . . . . .	29
6.11.1 Constructor & Destructor Documentation . . . . .	30
6.11.1.1 Plateau() . . . . .	30
6.11.1.2 ~Plateau() . . . . .	30
6.11.2 Member Function Documentation . . . . .	30
6.11.2.1 get_iterateur_debut() . . . . .	30
6.11.2.2 get_iterateur_fin() . . . . .	30
6.11.2.3 obtenir_hexagone() . . . . .	30
6.11.2.4 peut_placer() . . . . .	31
6.11.2.5 placer() . . . . .	31
6.11.3 Member Data Documentation . . . . .	31
6.11.3.1 plateau . . . . .	32

6.12 Quartier Class Reference	32
6.12.1 Detailed Description	32
6.13 Score Class Reference	32
6.13.1 Detailed Description	33
6.13.2 Constructor & Destructor Documentation	33
6.13.2.1 Score() [1/2]	34
6.13.2.2 Score() [2/2]	34
6.13.2.3 ~Score()	34
6.13.3 Member Function Documentation	34
6.13.3.1 score()	34
6.13.4 Member Data Documentation	34
6.13.4.1 scoreDecore	34
6.14 ScoreBleu Class Reference	35
6.14.1 Member Function Documentation	35
6.14.1.1 score()	35
6.14.1.2 score_bleu()	35
6.15 ScoreBleuVariante Class Reference	36
6.15.1 Member Function Documentation	36
6.15.1.1 score()	36
6.15.1.2 score_bleu_variante()	36
6.16 ScoreJaune Class Reference	37
6.16.1 Member Function Documentation	37
6.16.1.1 score()	37
6.16.1.2 score_jaune()	37
6.17 ScoreJauneVariante Class Reference	38
6.17.1 Member Function Documentation	38
6.17.1.1 score()	38
6.17.1.2 score_jaune_variante()	38
6.18 ScoreRouge Class Reference	39
6.18.1 Member Function Documentation	39
6.18.1.1 score()	39
6.18.1.2 score_rouge()	39
6.19 ScoreRougeVariante Class Reference	40
6.19.1 Member Function Documentation	40
6.19.1.1 score()	40
6.19.1.2 score_rouge_variante()	40
6.20 ScoreSoloArchitecte Class Reference	41
6.20.1 Detailed Description	41
6.20.2 Member Function Documentation	41
6.20.2.1 get_niveau()	41
6.20.2.2 set_niveau()	41
6.20.3 Member Data Documentation	42

6.20.3.1 niveau	42
6.21 ScoreSoloArchitecteBleu Class Reference	42
6.21.1 Member Function Documentation	42
6.21.1.1 score()	43
6.21.1.2 score_bleu()	43
6.22 ScoreSoloArchitecteJaune Class Reference	43
6.22.1 Member Function Documentation	43
6.22.1.1 score()	44
6.22.1.2 score_jaune()	44
6.23 ScoreSoloArchitecteRouge Class Reference	44
6.23.1 Member Function Documentation	44
6.23.1.1 score()	45
6.23.1.2 score_rouge()	45
6.24 ScoreSoloArchitecteVert Class Reference	45
6.24.1 Member Function Documentation	45
6.24.1.1 score()	46
6.24.1.2 score_vert()	46
6.25 ScoreSoloArchitecteViolet Class Reference	46
6.25.1 Member Function Documentation	46
6.25.1.1 score()	47
6.25.1.2 score_violet()	47
6.26 ScoreVert Class Reference	47
6.26.1 Member Function Documentation	47
6.26.1.1 score()	48
6.26.1.2 score_vert()	48
6.27 ScoreVertVariante Class Reference	48
6.27.1 Member Function Documentation	48
6.27.1.1 score()	49
6.27.1.2 score_vert_variante()	49
6.28 ScoreViolet Class Reference	49
6.28.1 Member Function Documentation	49
6.28.1.1 score()	50
6.28.1.2 score_violet()	50
6.29 ScoreVioletVariante Class Reference	50
6.29.1 Member Function Documentation	50
6.29.1.1 score()	51
6.29.1.2 score_violet_variante()	51
6.30 Tuile Class Reference	51
6.30.1 Detailed Description	52
6.30.2 Constructor & Destructor Documentation	52
6.30.2.1 Tuile()	52
6.30.2.2 ~Tuile()	52

6.30.3 Member Function Documentation	52
6.30.3.1 get_enfants()	52
6.30.3.2 get_hauteur()	52
6.30.3.3 get_nombre_enfant()	53
6.30.3.4 get_positions_enfants()	53
6.30.3.5 set_hauteur()	53
6.30.4 Member Data Documentation	53
6.30.4.1 enfants	53
6.30.4.2 hauteur	53
6.30.4.3 nombre_enfants	53
6.30.4.4 positions_enfants	53
6.30.4.5 rotation	54
6.31 TuileDepart Class Reference	54
6.32 TuileJeu Class Reference	54
6.33 TuileJeuConcrete Class Reference	55
6.34 Vector2 Class Reference	55
6.34.1 Constructor & Destructor Documentation	56
6.34.1.1 Vector2() [1/3]	56
6.34.1.2 Vector2() [2/3]	56
6.34.1.3 Vector2() [3/3]	56
6.34.2 Member Data Documentation	56
6.34.2.1 x	56
6.34.2.2 y	56
<b>7 File Documentation</b>	<b>57</b>
7.1 Affichage.cpp File Reference	57
7.2 Affichage.hpp File Reference	57
7.3 Affichage.hpp	57
7.4 Chantier.cpp File Reference	58
7.5 Chantier.hpp File Reference	58
7.5.1 Variable Documentation	58
7.5.1.1 max_tuiles_par_chantier	58
7.6 Chantier.hpp	59
7.7 Deck.cpp File Reference	59
7.8 Deck.hpp File Reference	59
7.8.1 Variable Documentation	59
7.8.1.1 max_tuiles_dans_deck	59
7.9 Deck.hpp	60
7.10 main.cpp File Reference	60
7.10.1 Function Documentation	60
7.10.1.1 main()	60
7.11 main.hpp File Reference	60



7.12 main.hpp . . . . .	60
7.13 Plateau.cpp File Reference . . . . .	61
7.14 Plateau.hpp File Reference . . . . .	61
7.15 Plateau.hpp . . . . .	61
7.16 Players.hpp File Reference . . . . .	61
7.17 Players.hpp . . . . .	62
7.18 README.md File Reference . . . . .	62
7.19 Score.cpp File Reference . . . . .	62
7.20 Score.hpp File Reference . . . . .	63
7.21 Score.hpp . . . . .	64
7.22 Tests.cpp File Reference . . . . .	65
7.22.1 Function Documentation . . . . .	65
7.22.1.1 assertTests() . . . . .	65
7.23 Tests.hpp File Reference . . . . .	65
7.23.1 Detailed Description . . . . .	66
7.23.2 Function Documentation . . . . .	66
7.23.2.1 assertTests() . . . . .	66
7.24 Tests.hpp . . . . .	66
7.25 Hexagone.cpp File Reference . . . . .	66
7.26 Hexagone.hpp File Reference . . . . .	67
7.27 Hexagone.hpp . . . . .	67
7.28 Tuile.cpp File Reference . . . . .	68
7.29 Tuile.hpp File Reference . . . . .	68
7.29.1 Variable Documentation . . . . .	68
7.29.1.1 max_enfants_par_tuile . . . . .	68
7.30 Tuile.hpp . . . . .	69
7.31 Utils.hpp File Reference . . . . .	69
7.31.1 Enumeration Type Documentation . . . . .	69
7.31.1.1 CouleursAkropolis . . . . .	69
7.32 Utils.hpp . . . . .	70
<b>Index</b>	<b>71</b>



# Chapter 1

## LO21 - Akropolis Project

### 1.1 Introduction

This is a LO21 project, aiming to recreate the board game Akropolis.

### 1.2 – Documentation –

All documentation about Doxygen can be found at :

See also

<https://www.doxygen.nl/manual/index.html>



## Chapter 2

# LO21-Projet

Akropolis en C++, full orienté objet

### 2.1 Méthodologie

Pour garantir une bonne compréhension du projet, je propose de maintenir une documentation du projet, qui regrouper chaque classe, chaque attribut et chaque méthode, expliquant leur but.

Pour ce faire, je propose d'utiliser l'outil proposé dans le cours ( [doxygen](#) ).

En plus de cela, je propose de faire une batterie de test pour chaque unité de code, c'est à dire faire des assert pour chaque fonction/fonctionnalité, avant que celle-ci soit implémenté histoire qu'on soit tou.te.s sur un même pied concernant la fonctionnalité des fonctions. Et cela nous permettra de vérifier si, au cours de modifications, une fonction ne fonctionne plus.



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

Affichage	11
AffichageConsole	12
Chantier	14
Deck	16
Hexagone	18
Carriere	13
Place	28
Quartier	32
Joueur	24
IllustreArchitecte	22
JoueurSimple	26
Plateau	29
Score	32
ScoreBleu	35
ScoreBleuVariante	36
ScoreJaune	37
ScoreJauneVariante	38
ScoreRouge	39
ScoreRougeVariante	40
ScoreSoloArchitecte	41
ScoreSoloArchitecteBleu	42
ScoreSoloArchitecteJaune	43
ScoreSoloArchitecteRouge	44
ScoreSoloArchitecteVert	45
ScoreSoloArchitecteViolet	46
ScoreVert	47
ScoreVertVariante	48
ScoreViolet	49
ScoreVioletVariante	50
Tuile	51
TuileDepart	54
TuileJeu	54
TuileJeuConcrete	55
Vector2	55





## Chapter 4

# Class Index

### 4.1 Class List

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

<a href="#">Affichage</a>	11
<a href="#">AffichageConsole</a>	12
<a href="#">Carriere</a>	
La classe <a href="#">Carriere</a>	13
<a href="#">Chantier</a>	
La classe implémentant le <a href="#">Chantier</a> , permettant aux <a href="#">Joueur</a> de piocher des <a href="#">Tuile</a>	14
<a href="#">Deck</a>	16
<a href="#">Hexagone</a>	
La classe <a href="#">Hexagone</a> : le bloc de base de Akropolis	18
<a href="#">IllustreArchitecte</a>	22
<a href="#">Joueur</a>	
La classe abstraite <a href="#">Joueur</a> , qui implémente n'importe quel <a href="#">Joueur</a>	24
<a href="#">JoueurSimple</a>	
Une implémentation concrète d'un <a href="#">Joueur</a> humain	26
<a href="#">Place</a>	
La classe <a href="#">Place</a>	28
<a href="#">Plateau</a>	
La classe implémentant le <a href="#">Plateau</a> de jeu d'un joueur	29
<a href="#">Quartier</a>	
La classe <a href="#">Quartier</a>	32
<a href="#">Score</a>	
Classe de <a href="#">Score</a> , utilisée pour calculer le score d'un joueur à partir d'un plateau	32
<a href="#">ScoreBleu</a>	
Décorateur qui implémente le scoring pour les points Bleus d'un joueur normal	35
<a href="#">ScoreBleuVariante</a>	
Décorateur qui implémente le scoring pour les points Bleus d'un joueur avec variante	36
<a href="#">ScoreJaune</a>	
Décorateur qui implémente le scoring pour les points Jaunes d'un joueur normal	37
<a href="#">ScoreJauneVariante</a>	
Décorateur qui implémente le scoring pour les points Jaunes d'un joueur avec variante	38
<a href="#">ScoreRouge</a>	
Décorateur qui implémente le scoring pour les points Rouges d'un joueur normal	39
<a href="#">ScoreRougeVariante</a>	
Décorateur qui implémente le scoring pour les points Rouges d'un joueur avec variante	40

<a href="#">ScoreSoloArchitecte</a>	Classe de <a href="#">Score</a> de l'Illustre Architecte, utilisée pour calculer le score de l'Illustre Architecte à partir d'un plateau . . . . .	41
<a href="#">ScoreSoloArchitecteBleu</a>	Décorateur qui implémente le scoring pour les points Bleus de l'Illustre Architecte . . . . .	42
<a href="#">ScoreSoloArchitecteJaune</a>	Décorateur qui implémente le scoring pour les points Jaunes de l'Illustre Architecte . . . . .	43
<a href="#">ScoreSoloArchitecteRouge</a>	Décorateur qui implémente le scoring pour les points Rouges de l'Illustre Architecte . . . . .	44
<a href="#">ScoreSoloArchitecteVert</a>	Décorateur qui implémente le scoring pour les points Verts de l'Illustre Architecte . . . . .	45
<a href="#">ScoreSoloArchitecteViolet</a>	Décorateur qui implémente le scoring pour les points Violets de l'Illustre Architecte . . . . .	46
<a href="#">ScoreVert</a>	Décorateur qui implémente le scoring pour les points Verts d'un joueur normal . . . . .	47
<a href="#">ScoreVertVariante</a>	Décorateur qui implémente le scoring pour les points Verts d'un joueur avec variante . . . . .	48
<a href="#">ScoreViolet</a>	Décorateur qui implémente le scoring pour les points Violets d'un joueur normal . . . . .	49
<a href="#">ScoreVioletVariante</a>	Décorateur qui implémente le scoring pour les points Violets d'un joueur avec variante . . . . .	50
<a href="#">Tuile</a>	La classe <a href="#">Tuile</a> , qui définit les tuiles de jeu . . . . .	51
<a href="#">TuileDepart</a>	Représente une <a href="#">Tuile</a> de Départ de jeu, ne peut pas être placée lors d'une partie . . . . .	54
<a href="#">TuileJeu</a>	Représente une <a href="#">Tuile</a> de jeu classique . . . . .	54
<a href="#">TuileJeuConcrete</a>	Implémentation concrète de la tuile de Jeu de base . . . . .	55
<a href="#">Vector2</a>	Représente un Vecteur 2D . . . . .	55

## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

<a href="#">Affichage.cpp</a>	57
<a href="#">Affichage.hpp</a>	57
<a href="#">Chantier.cpp</a>	58
<a href="#">Chantier.hpp</a>	58
<a href="#">Deck.cpp</a>	59
<a href="#">Deck.hpp</a>	59
<a href="#">main.cpp</a>	60
<a href="#">main.hpp</a>	60
<a href="#">Plateau.cpp</a>	61
<a href="#">Plateau.hpp</a>	61
<a href="#">Players.hpp</a>	61
<a href="#">Score.cpp</a>	62
<a href="#">Score.hpp</a>	63
<a href="#">Tests.cpp</a>	65
<a href="#">Tests.hpp</a>	
Test & Assertion function definition	65
<a href="#">Hexagone.cpp</a>	66
<a href="#">Hexagone.hpp</a>	67
<a href="#">Tuile.cpp</a>	68
<a href="#">Tuile.hpp</a>	68
<a href="#">Utils.hpp</a>	69



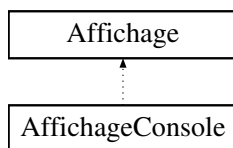
## Chapter 6

# Class Documentation

### 6.1 Affichage Class Reference

```
#include <Affichage.hpp>
```

Inheritance diagram for Affichage:



#### Public Member Functions

- virtual void [affiche\\_plateau\\_actuel](#) ([Joueur](#) &joueur)

#### 6.1.1 Member Function Documentation

##### 6.1.1.1 affiche\_plateau\_actuel()

```
virtual void Affichage::affiche_plateau_actuel (  
    Joueur & joueur ) [virtual]
```

Reimplemented in [AffichageConsole](#).

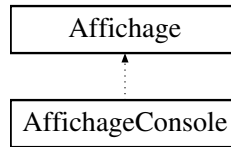
The documentation for this class was generated from the following file:

- [Affichage.hpp](#)

## 6.2 AffichageConsole Class Reference

```
#include <Affichage.hpp>
```

Inheritance diagram for AffichageConsole:



### Public Member Functions

- void [affiche\\_plateau\\_actuel](#) ([Joueur](#) &joueur)

### Static Public Member Functions

- static [AffichageConsole](#) \* [getInstance](#) ()

### Private Member Functions

- [AffichageConsole](#) ()=default

### Private Attributes

- const int [hexH](#) = 5
- const int [hexW](#) = 9

### Static Private Attributes

- static [AffichageConsole](#) \* [instance](#)

### 6.2.1 Constructor & Destructor Documentation

#### 6.2.1.1 AffichageConsole()

```
AffichageConsole::AffichageConsole ( ) [private], [default]
```

### 6.2.2 Member Function Documentation

### 6.2.2.1 affiche\_plateau\_actuel()

```
void AffichageConsole::affiche_plateau_actuel (
    Joueur & joueur ) [virtual]
```

Reimplemented from [Affichage](#).

### 6.2.2.2 getInstance()

```
static AffichageConsole * AffichageConsole::getInstance ( ) [inline], [static]
```

## 6.2.3 Member Data Documentation

### 6.2.3.1 hexH

```
const int AffichageConsole::hexH = 5 [private]
```

### 6.2.3.2 hexW

```
const int AffichageConsole::hexW = 9 [private]
```

### 6.2.3.3 instance

```
AffichageConsole* AffichageConsole::instance [static], [private]
```

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

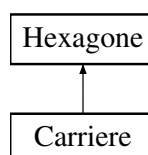
- [Affichage.hpp](#)
- [Affichage.cpp](#)

## 6.3 Carriere Class Reference

La classe [Carriere](#).

```
#include <Hexagone.hpp>
```

Inheritance diagram for Carriere:



## Additional Inherited Members

### 6.3.1 Detailed Description

Un [Hexagone](#), qui lorsque recouvert, donne une pierre au joueur actuel

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

- [Hexagone.hpp](#)

## 6.4 Chantier Class Reference

La classe implémentant le [Chantier](#), permettant aux [Joueur](#) de piocher des [Tuile](#).

```
#include <Chantier.hpp>
```

### Public Member Functions

- void [set\\_nombre\\_joueurs](#) (int nombre\_joueurs)  
*Calcule la taille du chantier selon le nombre de joueurs.*
- void [set\\_taille](#) (int taille)  
*Setteur de la taille du chantier.*
- int [get\\_taille](#) ()  
*Getteur de la taille du chantier.*
- int [get\\_nombre\\_tuiles](#) ()  
*Getteur du nombres de tuiles dans le chantier.*
- [Tuile](#) \* [get\\_tuiles](#) ()  
*Getteur des tuiles du chantier.*
- [Tuile](#) [prendre\\_tuile](#) (int index)  
*Retire la tuile, à l'index indiqué, du chantier.*
- void [ajouter\\_tuile](#) ([Tuile](#) \*tuile)  
*Ajoute une [Tuile](#) à la fin du chantier.*
- void [ajouter\\_tuile](#) ([Tuile](#) \*tuile, int nombre)  
*Ajoute des [Tuile](#) à la fin du chantier.*

### Private Attributes

- int [taille](#)  
*La taille maximale du [Chantier](#) (est dépendante du nombre de joueurs)*
- int [nombreTuiles](#) = 0  
*le nombre de tuiles actuellement dans le [Chantier](#)*
- [Tuile](#) [tuiles](#) [[max\\_tuiles\\_par\\_chantier](#)]  
*Le tableau enregistrant toutes les tuiles sur tout le [Chantier](#).*

### 6.4.1 Member Function Documentation



#### 6.4.1.1 ajouter\_tuile() [1/2]

```
void Chantier::ajouter_tuile (
    Tuile * tuile )
```

#### 6.4.1.2 ajouter\_tuile() [2/2]

```
void Chantier::ajouter_tuile (
    Tuile * tuile,
    int nombre )
```

#### 6.4.1.3 get\_nombre\_tuiles()

```
int Chantier::get_nombre_tuiles ( ) [inline]
```

#### 6.4.1.4 get\_taille()

```
int Chantier::get_taille ( ) [inline]
```

#### 6.4.1.5 get\_tuiles()

```
Tuile * Chantier::get_tuiles ( ) [inline]
```

#### 6.4.1.6 prendre\_tuile()

```
Tuile Chantier::prendre_tuile (
    int index )
```

Retire la tuile, à l'index indiqué, du chantier Les tuiles d'index supérieur vont "descendre" le chantier comme un mécanisme de rivière.

##### Parameters

<i>index</i>	l'index de la tuile à prendre
--------------	-------------------------------

**Returns**

La [Tuile](#) selectionée

**6.4.1.7 set\_nombre\_joueurs()**

```
void Chantier::set_nombre_joueurs (
    int nombre_joueurs )
```

**6.4.1.8 set\_taille()**

```
void Chantier::set_taille (
    int taille ) [inline]
```

**6.4.2 Member Data Documentation****6.4.2.1 nombreTuiles**

```
int Chantier::nombreTuiles = 0 [private]
```

**6.4.2.2 taille**

```
int Chantier::taille [private]
```

**6.4.2.3 tuiles**

```
Tuile Chantier::tuiles[max\_tuiles\_par\_chantier] [private]
```

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

- [Chantier.hpp](#)

**6.5 Deck Class Reference**

```
#include <Deck.hpp>
```

## Public Member Functions

- [Deck](#) (int nombre\_joueurs)  
*Initialise le [Deck](#) selon le nombre de joueurs.*
- int [get\\_taille](#) ()  
*Getteur de la taille maximale du [Deck](#).*
- int [get\\_nombre\\_tuiles](#) ()  
*Getteur du nombre de tuiles.*
- [Tuile tirer\\_tuile](#) ()  
*Tire la tuile la plus en haut du deck.*
- [Tuile](#) \* [tirer\\_tuile](#) (int nombre\_tuiles)  
*Tire les x tuiles les plus en haut du deck.*

## Private Attributes

- int [taille](#)  
*La taille maximale du [Deck](#) (est dépendante du nombre de joueurs)*
- int [nombreTuiles](#)  
*le nombre de tuiles actuellement dans le [Deck](#)*
- [Tuile](#) [tuiles](#) [[max\\_tuiles\\_dans\\_deck](#)]  
*Le tableau enregistrant toutes les tuiles sur tout le [Deck](#).*

## 6.5.1 Constructor & Destructor Documentation

### 6.5.1.1 Deck()

```
Deck::Deck (
    int nombre_joueurs )
```

## 6.5.2 Member Function Documentation

### 6.5.2.1 get\_nombre\_tuiles()

```
int Deck::get_nombre_tuiles ( )
```

### 6.5.2.2 get\_taille()

```
int Deck::get_taille ( )
```

### 6.5.2.3 tirer\_tuile() [1/2]

```
Tuile Deck::tirer_tuile ( )
```

### 6.5.2.4 tirer\_tuile() [2/2]

```
Tuile * Deck::tirer_tuile (
    int nombre_tuiles )
```

## 6.5.3 Member Data Documentation

### 6.5.3.1 nombreTuiles

```
int Deck::nombreTuiles [private]
```

### 6.5.3.2 taille

```
int Deck::taille [private]
```

### 6.5.3.3 tuiles

```
Tuile Deck::tuiles[max_tuiles_dans_deck] [private]
```

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

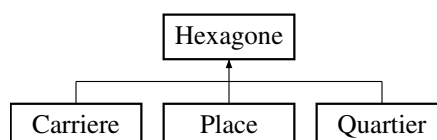
- [Deck.hpp](#)

## 6.6 Hexagone Class Reference

La classe [Hexagone](#) : le bloc de base de Akropolis.

```
#include <Hexagone.hpp>
```

Inheritance diagram for Hexagone:



## Public Member Functions

- [Hexagone](#) ()=default
- [~Hexagone](#) ()=default
- int [get\\_couleur](#) () const  
*Retourne la couleur de cet hexagone.*
- [Tuile](#) \* [get\\_tuile](#) () const  
*Retourne la tuile qui contient cet hexagone.*
- [Vector2](#) [get\\_local\\_position](#) () const  
*Retourne la position locale de cet hexagone dans la tuile.*
- int [get\\_hauteur](#) () const  
*Retourne la hauteur de cet hexagone.*
- bool [peut\\_etre\\_placee](#) ([Plateau](#) \*map, [Vector2](#) position) const  
*Retourne si l'hexagone peut être placé à cet endroit.*
- void [quand\\_recouvert](#) ([Joueur](#) \*joueur\_qui\_recouvre) const  
*Fonction appelée dès que l'hexagone est recouvert.*
- string [get\\_displayed\\_text](#) () const  
*Retourne le texte affichée au centre de l'hexagone affichée dans la Console.*

## Protected Attributes

- enum [CouleursAkropolis](#) couleur  
*La couleur de cet hexagone.*
- [Tuile](#) \* [tuileParent](#)  
*La tuile qui contient l'hexagone.*
- int [indice\\_tuile](#)  
*L'indice de cet hexagone, dans tuileParent.*

### 6.6.1 Detailed Description

La classe [Hexagone](#) modelise un hexagone, qui pourra être placé par un [Joueur](#) sur un [Plateau](#). Il s'agit d'une classe abstraite, qui est héritée par d'autres classes Comme [Place](#), [Carriere](#) ou [Quartier](#)

### 6.6.2 Constructor & Destructor Documentation

#### 6.6.2.1 Hexagone()

```
Hexagone::Hexagone ( ) [default]
```

#### 6.6.2.2 ~Hexagone()

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

## 6.6.3 Member Function Documentation

### 6.6.3.1 `get_couleur()`

```
int Hexagone::get_couleur ( ) const
```

### 6.6.3.2 `get_displayed_text()`

```
string Hexagone::get_displayed_text ( ) const [inline]
```

Retourne le texte affichée au centre de l'hexagone affichée dans la Console, est utilisée dans [AffichageConsole](#). Et est surchargée à chaque implémentation concrète de l'[Hexagone](#).

#### Returns

Le texte à afficher au centre de l'hexagone dans l'affichage console

### 6.6.3.3 `get_hauteur()`

```
int Hexagone::get_hauteur ( ) const
```

#### Returns

La hauteur de cet hexagone, qui est obtenu à partir de la hauteur de la tuile qui le contient

### 6.6.3.4 `get_local_position()`

```
Vector2 Hexagone::get_local_position ( ) const
```

### 6.6.3.5 `get_tuile()`

```
Tuile * Hexagone::get_tuile ( ) const
```

### 6.6.3.6 `peut_etre_placee()`

```
bool Hexagone::peut_etre_placee (
    Plateau * map,
    Vector2 position ) const
```

## Parameters

<i>map</i>	Un pointeur vers le plateau dans lequel on va vouloir placer l'hexagone
<i>position</i>	Un vecteur 2d qui donne la position à laquelle on va vouloir placer l'hexagone /\ Attention /\, cette fonction ne dit pas si l'emplacement est un emplacements valide, mais elle va dire (à partir d'un emplacement génériquement valide), si on peut placer cet hexagone spécifique, à cet endroit.

## Returns

Si l'hexagone peut être placé à cet endroit.

## 6.6.3.7 quand\_recouvert()

```
void Hexagone::quand_recouvert (
    Joueur * joueur_qui_recouvre ) const
```

## Parameters

<i>joueur_qui_recouvre</i>	un pointeur vers le <a href="#">Joueur</a> qui recouvre la tuile. Est utilisé pour modifier des attributs du joueur. Cette fonction est appelée dès que l'hexagone est recouvert. Elle sert notamment à la classe Carrière à ajouter une pierre au joueur.
----------------------------	--

## 6.6.4 Member Data Documentation

## 6.6.4.1 couleur

```
enum CouleursAkropolis Hexagone::couleur [protected]
```

## 6.6.4.2 indice\_tuile

```
int Hexagone::indice_tuile [protected]
```

## 6.6.4.3 tuileParent

```
Tuile* Hexagone::tuileParent [protected]
```

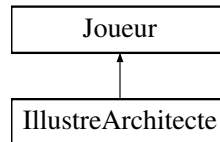
The documentation for this class was generated from the following files:

- [Hexagone.hpp](#)
- [Hexagone.cpp](#)

## 6.7 IllustreArchitecte Class Reference

```
#include <Players.hpp>
```

Inheritance diagram for IllustreArchitecte:



### Public Member Functions

- void `set_niveau` (int `niveau`)  
*Le setteur du niveau.*
- int `get_niveau` ()  
*Le getteur du niveau.*
- void `jouer` (`Tuile` \*chantier)  
*La fonction appelée pour faire jouer l'Illustre Architecte.*

### Protected Member Functions

- `Tuile` `choisir_tuile` (`Tuile` \*chantier)  
*Choisit une tuile à partir du chantier.*
- `Vector2` `trouver_emplacement_tuile` (`Tuile` &tuile)  
*Trouve un emplacement vide et valide pour placer une tuile.*

### Protected Attributes

- `ScoreSoloArchitecte` \* `score`  
*Le `Score` de l'illustre architecte, il va être utilisé pour calculer son score à la fin de la partie.*
- int `niveau` = 0  
*Le niveau de l'Architecte (va changer les regles)*
- bool `joueToutSeul` = true  
*L'illustre Architecte étant un automate, il joue tout seul.*

### 6.7.1 Member Function Documentation

#### 6.7.1.1 `choisir_tuile()`

```
Tuile IllustreArchitecte::choisir_tuile (  
    Tuile * chantier ) [protected]
```



### 6.7.1.2 get\_niveau()

```
int IllustreArchitecte::get_niveau ( )
```

### 6.7.1.3 jouer()

```
void IllustreArchitecte::jouer (
    Tuile * chantier )
```

### 6.7.1.4 set\_niveau()

```
void IllustreArchitecte::set_niveau (
    int niveau )
```

### 6.7.1.5 trouver\_emplacement\_tuile()

```
Vector2 IllustreArchitecte::trouver_emplacement_tuile (
    Tuile & tuile ) [protected]
```

## 6.7.2 Member Data Documentation

### 6.7.2.1 joueToutSeul

```
bool IllustreArchitecte::joueToutSeul = true [protected]
```

### 6.7.2.2 niveau

```
int IllustreArchitecte::niveau = 0 [protected]
```

### 6.7.2.3 score

```
ScoreSoloArchitecte* IllustreArchitecte::score [protected]
```

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

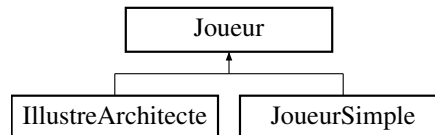
- [Players.hpp](#)

## 6.8 Joueur Class Reference

La classe abstraite `Joueur`, qui implémente n'importe quel `Joueur`.

```
#include <Players.hpp>
```

Inheritance diagram for `Joueur`:



### Public Member Functions

- `int get_score ()`  
*Retourne le `Score` calculé du `Joueur`.*
- `int get_pierre ()`  
*Retourne la quantité de pierres du `Joueur`.*
- `void set_pierre (int pierre)`  
*Set la quantité de pierres du `Joueur`.*
- `void ajouter_pierre (int pierre)`  
*Ajoute des pierres au `Joueur`.*
- `Plateau & get_plateau ()`  
*Retourne le plateau du `Joueur`.*
- `bool place_tuile (Tuile tuile, Vector2 coordonnées)`  
*Tente de placer une tuile aux coordonnées indiquées.*
- `void jouer (Tuile *chantier)`  
*La fonction appelée lorsque le joueur joueToutSeul.*

### Protected Attributes

- `int pierre = 0`  
*Le nombre de pierres du `Joueur`.*
- `Plateau & plateauJoueur`  
*Le `Plateau` du joueur.*
- `bool joueToutSeul`  
*Si le joueur joue tout seul.*

#### 6.8.1 Detailed Description

La classe qui va représenter à la fois chaque joueur humain, mais à la fois chaque joueur non humain (dont l'IllustreArchitecte)

#### 6.8.2 Member Function Documentation

### 6.8.2.1 ajouter\_pierre()

```
void Joueur::ajouter_pierre (
    int pierre )
```

### 6.8.2.2 get\_pierre()

```
int Joueur::get_pierre ( )
```

### 6.8.2.3 get\_plateau()

```
Plateau & Joueur::get_plateau ( )
```

### 6.8.2.4 get\_score()

```
int Joueur::get_score ( )
```

### 6.8.2.5 jouer()

```
void Joueur::jouer (
    Tuile * chantier )
```

#### Parameters

<i>chantier</i>	un tableau de <a href="#">Tuile</a> , représentant le chantier
-----------------	--

### 6.8.2.6 place\_tuile()

```
bool Joueur::place_tuile (
    Tuile tuile,
    Vector2 coordonées )
```

#### Parameters

<i>tuile</i>	la <a href="#">Tuile</a> à placer
<i>coordonées</i>	le <a href="#">Vector2</a> qui donne la position de la <a href="#">Tuile</a>

### Returns

Si la tuile as été placé ou non

#### 6.8.2.7 set\_pierre()

```
void Joueur::set_pierre (
    int pierre )
```

### 6.8.3 Member Data Documentation

#### 6.8.3.1 joueToutSeul

```
bool Joueur::joueToutSeul [protected]
```

Si le joueur joue tout seul.

Cet attribut indique au moteur de jeu, si il as à attendre des inputs de la part du joueur

#### 6.8.3.2 pierre

```
int Joueur::pierre = 0 [protected]
```

#### 6.8.3.3 plateauJoueur

```
Plateau& Joueur::plateauJoueur [protected]
```

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

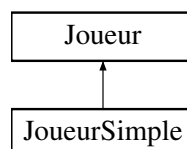
- [Players.hpp](#)

## 6.9 JoueurSimple Class Reference

Une implémentation concrète d'un [Joueur](#) humain.

```
#include <Players.hpp>
```

Inheritance diagram for JoueurSimple:



## Public Member Functions

- `int get_score ()`  
*Retourne le [Score](#) calculé du [Joueur](#).*

## Protected Attributes

- `Score * scoreJoueur`  
*Le [Score](#) du joueur, va être utilisé pour calculer le score du joueur à la fin de la partie.*
- `bool joueToutSeul = false`  
*Le joueur, étant humain, ne joue pas tout seul.*

### 6.9.1 Detailed Description

Représente un [Joueur](#) humain

### 6.9.2 Member Function Documentation

#### 6.9.2.1 get\_score()

```
int JoueurSimple::get_score ( )
```

### 6.9.3 Member Data Documentation

#### 6.9.3.1 joueToutSeul

```
bool JoueurSimple::joueToutSeul = false [protected]
```

#### 6.9.3.2 scoreJoueur

```
Score* JoueurSimple::scoreJoueur [protected]
```

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

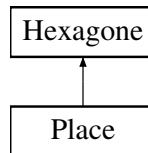
- [Players.hpp](#)

## 6.10 Place Class Reference

La classe [Place](#).

```
#include <Hexagone.hpp>
```

Inheritance diagram for Place:



### Public Member Functions

- [Place](#) ()=default
- [~Place](#) ()
- int [get\\_etoiles](#) ()

*Retourne le nombre d'étoiles.*

### Protected Attributes

- int [etoiles](#)

*Le nombre d'etoiles (le multiplicateur) de cette place.*

#### 6.10.1 Detailed Description

Un [Hexagone](#) qui permet d'ajouter un multiplicateur à une couleur.

#### 6.10.2 Constructor & Destructor Documentation

##### 6.10.2.1 Place()

```
Place::Place ( ) [default]
```

##### 6.10.2.2 ~Place()

```
Place::~~Place ( ) [inline]
```

### 6.10.3 Member Function Documentation

#### 6.10.3.1 get\_etoiles()

```
int Place::get_etoiles ( )
```

### 6.10.4 Member Data Documentation

#### 6.10.4.1 etoiles

```
int Place::etoiles [protected]
```

Permet de savoir le multiplicateur à appliquer à la fin de la partie pour le score

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

- [Hexagone.hpp](#)

## 6.11 Plateau Class Reference

La classe implémentant le [Plateau](#) de jeu d'un joueur.

```
#include <Plateau.hpp>
```

### Public Member Functions

- [Plateau](#) ()=default
- [~Plateau](#) ()=default
- [Hexagone](#) & [obtenir\\_hexagone](#) (const [Vector2](#) &coordonees)  
*TODO: implémenter le destructeur de [Plateau](#).*
- bool [peut\\_placer](#) (const [Tuile](#) &tuileJeu, const [Vector2](#) &position)  
*Retourne true si il est possible de placer tuileJeu à la position du [Vector2](#).*
- bool [placer](#) (const [Tuile](#) &tuile, const [Vector2](#) &position)  
*Place une tuile à des coordonnées spécifiques.*
- map< [Vector2](#), [Hexagone](#) & >::iterator [get\\_iterateur\\_debut](#) ()  
*Retourne un itérateur sur le plateau, mis au début de celui-ci.*
- map< [Vector2](#), [Hexagone](#) & >::iterator [get\\_iterateur\\_fin](#) ()  
*Retourne un itérateur sur le plateau, mis à la fin de celui-ci.*

## Private Attributes

- `map< Vector2, Hexagone & > plateau`

*La map associant une coordonnée vectorielle à un hexagone.*

## 6.11.1 Constructor & Destructor Documentation

### 6.11.1.1 Plateau()

```
Plateau::Plateau ( ) [default]
```

### 6.11.1.2 ~Plateau()

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

## 6.11.2 Member Function Documentation

### 6.11.2.1 get\_iterateur\_debut()

```
map< Vector2, Hexagone & >::iterator Plateau::get_iterateur_debut ( ) [inline]
```

### 6.11.2.2 get\_iterateur\_fin()

```
map< Vector2, Hexagone & >::iterator Plateau::get_iterateur_fin ( ) [inline]
```

### 6.11.2.3 obtenir\_hexagone()

```
Hexagone & Plateau::obtenir_hexagone (
    const Vector2 & coordonees )
```

Retourne l'hexagone aux coordonnées du `Vector2`



## Parameters

<i>coordonees</i>	le <a href="#">Vector2</a> des coordonnées de l'hexagone recherché
-------------------	--

## Returns

l'hexagone à ces coordonnées

### 6.11.2.4 peut\_placer()

```
bool Plateau::peut_placer (
    const Tuile & tuileJeu,
    const Vector2 & position )
```

## Parameters

<i>tuileJeu</i>	La <a href="#">Tuile</a> qu'on essaye de placer
<i>position</i>	La position à laquelle on essaye de placer la <a href="#">Tuile</a>

## Returns

si l'emplacement est valide (check également si les condition de placement des Hexagones sont validés)

### 6.11.2.5 placer()

```
bool Plateau::placer (
    const Tuile & tuile,
    const Vector2 & position )
```

## Parameters

<i>tuile</i>	La <a href="#">Tuile</a> à placer
<i>position</i>	La position où l'on essaye de placer la <a href="#">Tuile</a>

## Returns

si le placement s'est bien effectué ou non

## 6.11.3 Member Data Documentation

### 6.11.3.1 plateau

```
map<Vector2, Hexagone> Plateau::plateau [private]
```

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

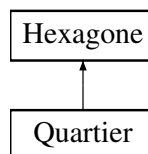
- [Plateau.hpp](#)

## 6.12 Quartier Class Reference

La classe [Quartier](#).

```
#include <Hexagone.hpp>
```

Inheritance diagram for Quartier:



### Additional Inherited Members

#### 6.12.1 Detailed Description

Un [Hexagone](#), qui est utilisé pour scorer des points

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

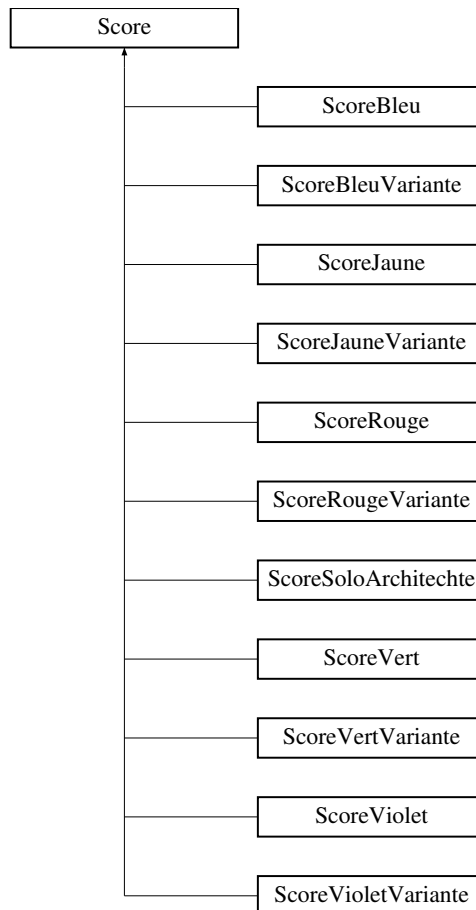
- [Hexagone.hpp](#)

## 6.13 Score Class Reference

Classe de [Score](#), utilisée pour calculer le score d'un joueur à partir d'un plateau.

```
#include <Score.hpp>
```

Inheritance diagram for Score:



## Public Member Functions

- `Score()`=default
- `Score(Score *scoreDecore)`
- `virtual ~Score()`=default
- `virtual int score(Plateau *plateau)`

*La fonction de calcul de `Score`.*

## Protected Attributes

- `Score * scoreDecore = nullptr`

*Le score décoré*

### 6.13.1 Detailed Description

Classe de `Score`, utilisée pour calculer le score d'un joueur à partir d'un plateau Est une classe abstraite, mais également un décorateur.

### 6.13.2 Constructor & Destructor Documentation

#### 6.13.2.1 Score() [1/2]

```
Score::Score ( ) [default]
```

#### 6.13.2.2 Score() [2/2]

```
Score::Score (
    Score * scoreDecore ) [inline]
```

#### 6.13.2.3 ~Score()

```
virtual Score::~~Score ( ) [virtual], [default]
```

### 6.13.3 Member Function Documentation

#### 6.13.3.1 score()

```
virtual int Score::score (
    Plateau * plateau ) [inline], [virtual]
```

Reimplemented in [ScoreSoloArchitecteBleu](#), [ScoreSoloArchitecteRouge](#), [ScoreSoloArchitecteJaune](#), [ScoreSoloArchitecteVert](#), [ScoreSoloArchitecteViolet](#), [ScoreBleu](#), [ScoreRouge](#), [ScoreVert](#), [ScoreViolet](#), [ScoreJaune](#), [ScoreBleuVariante](#), [ScoreRougeVariante](#), [ScoreVertVariante](#), [ScoreVioletVariante](#), and [ScoreJauneVariante](#).

### 6.13.4 Member Data Documentation

#### 6.13.4.1 scoreDecore

```
Score* Score::scoreDecore = nullptr [protected]
```

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

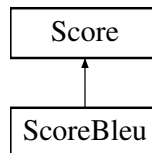
- [Score.hpp](#)

## 6.14 ScoreBleu Class Reference

Décorateur qui implémente le scoring pour les points Bleus d'un joueur normal.

```
#include <Score.hpp>
```

Inheritance diagram for ScoreBleu:



### Public Member Functions

- int [score](#) ([Plateau](#) \*plateau) override  
*La fonction de calcul de [Score](#).*

### Private Member Functions

- int [score\\_bleu](#) ([Plateau](#) \*plateau)

### Additional Inherited Members

#### 6.14.1 Member Function Documentation

##### 6.14.1.1 [score\(\)](#)

```
int ScoreBleu::score (  
    Plateau * plateau ) [override], [virtual]
```

Reimplemented from [Score](#).

##### 6.14.1.2 [score\\_bleu\(\)](#)

```
int ScoreBleu::score_bleu (  
    Plateau * plateau ) [private]
```

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

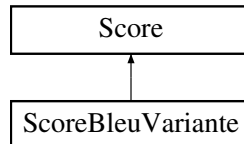
- [Score.hpp](#)

## 6.15 ScoreBleuVariante Class Reference

Décorateur qui implémente le scoring pour les points Bleus d'un joueur avec variante.

```
#include <Score.hpp>
```

Inheritance diagram for ScoreBleuVariante:



### Public Member Functions

- int [score](#) ([Plateau](#) \*plateau) override  
*La fonction de calcul de [Score](#).*

### Private Member Functions

- int [score\\_bleu\\_variante](#) ([Plateau](#) \*plateau)

### Additional Inherited Members

#### 6.15.1 Member Function Documentation

##### 6.15.1.1 [score\(\)](#)

```
int ScoreBleuVariante::score (
    Plateau * plateau ) [override], [virtual]
```

Reimplemented from [Score](#).

##### 6.15.1.2 [score\\_bleu\\_variante\(\)](#)

```
int ScoreBleuVariante::score_bleu_variante (
    Plateau * plateau ) [private]
```

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

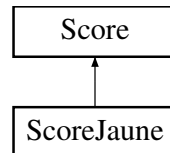
- [Score.hpp](#)

## 6.16 ScoreJaune Class Reference

Décorateur qui implémente le scoring pour les points Jaunes d'un joueur normal.

```
#include <Score.hpp>
```

Inheritance diagram for ScoreJaune:



### Public Member Functions

- int [score](#) ([Plateau](#) \*plateau) override  
*La fonction de calcul de [Score](#).*

### Private Member Functions

- int [score\\_jaune](#) ([Plateau](#) \*plateau)

### Additional Inherited Members

#### 6.16.1 Member Function Documentation

##### 6.16.1.1 [score\(\)](#)

```
int ScoreJaune::score (  
    Plateau * plateau ) [override], [virtual]
```

Reimplemented from [Score](#).

##### 6.16.1.2 [score\\_jaune\(\)](#)

```
int ScoreJaune::score_jaune (  
    Plateau * plateau ) [private]
```

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

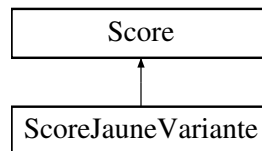
- [Score.hpp](#)

## 6.17 ScoreJauneVariante Class Reference

Décorateur qui implémente le scoring pour les points Jaunes d'un joueur avec variante.

```
#include <Score.hpp>
```

Inheritance diagram for ScoreJauneVariante:



### Public Member Functions

- int [score](#) ([Plateau](#) \*plateau) override  
*La fonction de calcul de [Score](#).*

### Private Member Functions

- int [score\\_jaune\\_variante](#) ([Plateau](#) \*plateau)

### Additional Inherited Members

#### 6.17.1 Member Function Documentation

##### 6.17.1.1 [score\(\)](#)

```
int ScoreJauneVariante::score (  
    Plateau * plateau ) [override], [virtual]
```

Reimplemented from [Score](#).

##### 6.17.1.2 [score\\_jaune\\_variante\(\)](#)

```
int ScoreJauneVariante::score_jaune_variante (  
    Plateau * plateau ) [private]
```

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

- [Score.hpp](#)

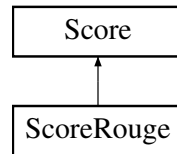


## 6.18 ScoreRouge Class Reference

Décorateur qui implémente le scoring pour les points Rouges d'un joueur normal.

```
#include <Score.hpp>
```

Inheritance diagram for ScoreRouge:



### Public Member Functions

- int [score](#) ([Plateau](#) \*plateau) override  
*La fonction de calcul de [Score](#).*

### Private Member Functions

- int [score\\_rouge](#) ([Plateau](#) \*plateau)

### Additional Inherited Members

#### 6.18.1 Member Function Documentation

##### 6.18.1.1 [score\(\)](#)

```
int ScoreRouge::score (  
    Plateau * plateau ) [override], [virtual]
```

Reimplemented from [Score](#).

##### 6.18.1.2 [score\\_rouge\(\)](#)

```
int ScoreRouge::score_rouge (  
    Plateau * plateau ) [private]
```

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

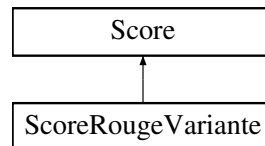
- [Score.hpp](#)

## 6.19 ScoreRougeVariante Class Reference

Décorateur qui implémente le scoring pour les points Rouges d'un joueur avec variante.

```
#include <Score.hpp>
```

Inheritance diagram for ScoreRougeVariante:



### Public Member Functions

- int [score](#) ([Plateau](#) \*plateau) override  
*La fonction de calcul de [Score](#).*

### Private Member Functions

- int [score\\_rouge\\_variante](#) ([Plateau](#) \*plateau)

### Additional Inherited Members

#### 6.19.1 Member Function Documentation

##### 6.19.1.1 [score\(\)](#)

```
int ScoreRougeVariante::score (  
    Plateau * plateau ) [override], [virtual]
```

Reimplemented from [Score](#).

##### 6.19.1.2 [score\\_rouge\\_variante\(\)](#)

```
int ScoreRougeVariante::score_rouge_variante (  
    Plateau * plateau ) [private]
```

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

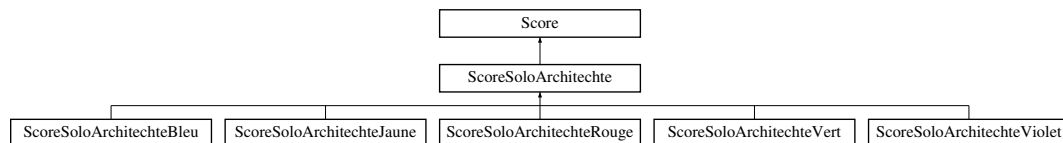
- [Score.hpp](#)

## 6.20 ScoreSoloArchitecte Class Reference

Classe de [Score](#) de l'Illustre Architecte, utilisée pour calculer le score de l'Illustre Architecte à partir d'un plateau.

```
#include <Score.hpp>
```

Inheritance diagram for ScoreSoloArchitecte:



### Public Member Functions

- virtual void [set\\_niveau](#) (int [niveau](#))  
*Le setteur de niveau de l'Illustre Architecte.*
- int [get\\_niveau](#) ()  
*Le getteur de niveau de l'Illustre Architecte.*

### Protected Attributes

- int [niveau](#) = 0  
*Le niveau de l'Illustre Architecte, sera utilisé pour calculer son score.*

### 6.20.1 Detailed Description

Classe de [Score](#) de l'Illustre Architecte, utilisée pour calculer le score de l'Illustre Architecte à partir d'un plateau. Est une classe abstraite, mais également un décorateur.

### 6.20.2 Member Function Documentation

#### 6.20.2.1 get\_niveau()

```
int ScoreSoloArchitecte::get_niveau ( )
```

#### 6.20.2.2 set\_niveau()

```
virtual void ScoreSoloArchitecte::set_niveau (
    int niveau ) [virtual]
```

### 6.20.3 Member Data Documentation

#### 6.20.3.1 niveau

```
int ScoreSoloArchitecte::niveau = 0 [protected]
```

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

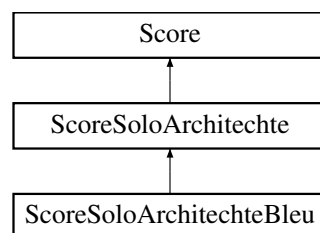
- [Score.hpp](#)

## 6.21 ScoreSoloArchitecteBleu Class Reference

Décorateur qui implémente le scoring pour les points Bleus de l'Illustre Architecte.

```
#include <Score.hpp>
```

Inheritance diagram for ScoreSoloArchitecteBleu:



### Public Member Functions

- int [score](#) ([Plateau](#) \*plateau) override  
*La fonction de calcul de [Score](#).*

### Private Member Functions

- int [score\\_bleu](#) ([Plateau](#) \*plateau)

### Additional Inherited Members

#### 6.21.1 Member Function Documentation

### 6.21.1.1 score()

```
int ScoreSoloArchitecteBleu::score (
    Plateau * plateau ) [override], [virtual]
```

Reimplemented from [Score](#).

### 6.21.1.2 score\_bleu()

```
int ScoreSoloArchitecteBleu::score_bleu (
    Plateau * plateau ) [private]
```

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

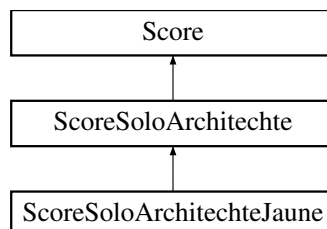
- [Score.hpp](#)

## 6.22 ScoreSoloArchitecteJaune Class Reference

Décorateur qui implémente le scoring pour les points Jaunes de l'Illustre Architecte.

```
#include <Score.hpp>
```

Inheritance diagram for ScoreSoloArchitecteJaune:



### Public Member Functions

- int [score](#) ([Plateau](#) \*plateau) override  
*La fonction de calcul de [Score](#).*

### Private Member Functions

- int [score\\_jaune](#) ([Plateau](#) \*plateau)

### Additional Inherited Members

#### 6.22.1 Member Function Documentation

### 6.22.1.1 score()

```
int ScoreSoloArchitecteJaune::score (
    Plateau * plateau ) [override], [virtual]
```

Reimplemented from [Score](#).

### 6.22.1.2 score\_jaune()

```
int ScoreSoloArchitecteJaune::score_jaune (
    Plateau * plateau ) [private]
```

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

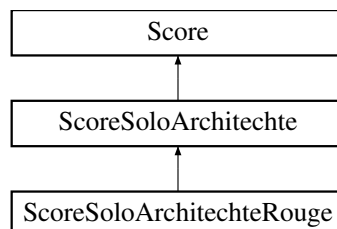
- [Score.hpp](#)

## 6.23 ScoreSoloArchitecteRouge Class Reference

Décorateur qui implémente le scoring pour les points Rouges de l'Illustre Architecte.

```
#include <Score.hpp>
```

Inheritance diagram for ScoreSoloArchitecteRouge:



### Public Member Functions

- int [score](#) ([Plateau](#) \*plateau) override  
*La fonction de calcul de [Score](#).*

### Private Member Functions

- int [score\\_rouge](#) ([Plateau](#) \*plateau)

### Additional Inherited Members

#### 6.23.1 Member Function Documentation

### 6.23.1.1 score()

```
int ScoreSoloArchitecteRouge::score (
    Plateau * plateau ) [override], [virtual]
```

Reimplemented from [Score](#).

### 6.23.1.2 score\_rouge()

```
int ScoreSoloArchitecteRouge::score_rouge (
    Plateau * plateau ) [private]
```

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

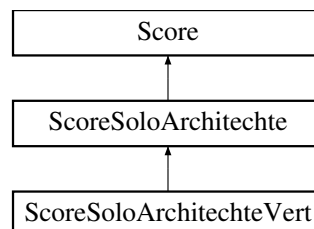
- [Score.hpp](#)

## 6.24 ScoreSoloArchitecteVert Class Reference

Décorateur qui implémente le scoring pour les points Verts de l'Illustre Architecte.

```
#include <Score.hpp>
```

Inheritance diagram for ScoreSoloArchitecteVert:



### Public Member Functions

- int [score](#) ([Plateau](#) \*plateau) override  
*La fonction de calcul de [Score](#).*

### Private Member Functions

- int [score\\_vert](#) ([Plateau](#) \*plateau)

### Additional Inherited Members

#### 6.24.1 Member Function Documentation

#### 6.24.1.1 score()

```
int ScoreSoloArchitecteVert::score (
    Plateau * plateau ) [override], [virtual]
```

Reimplemented from [Score](#).

#### 6.24.1.2 score\_vert()

```
int ScoreSoloArchitecteVert::score_vert (
    Plateau * plateau ) [private]
```

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

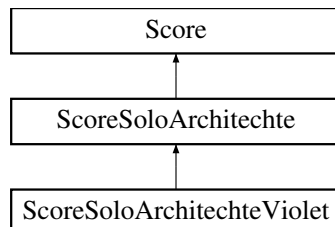
- [Score.hpp](#)

## 6.25 ScoreSoloArchitecteViolet Class Reference

Décorateur qui implémente le scoring pour les points Violet de l'Illustre Architecte.

```
#include <Score.hpp>
```

Inheritance diagram for ScoreSoloArchitecteViolet:



### Public Member Functions

- int [score](#) ([Plateau](#) \*plateau) override  
*La fonction de calcul de [Score](#).*

### Private Member Functions

- int [score\\_violet](#) ([Plateau](#) \*plateau)

### Additional Inherited Members

#### 6.25.1 Member Function Documentation



### 6.25.1.1 score()

```
int ScoreSoloArchitecteViolet::score (
    Plateau * plateau ) [override], [virtual]
```

Reimplemented from [Score](#).

### 6.25.1.2 score\_violet()

```
int ScoreSoloArchitecteViolet::score_violet (
    Plateau * plateau ) [private]
```

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

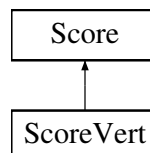
- [Score.hpp](#)

## 6.26 ScoreVert Class Reference

Décorateur qui implémente le scoring pour les points Verts d'un joueur normal.

```
#include <Score.hpp>
```

Inheritance diagram for ScoreVert:



### Public Member Functions

- int [score](#) ([Plateau](#) \*plateau) override  
*La fonction de calcul de [Score](#).*

### Private Member Functions

- int [score\\_vert](#) ([Plateau](#) \*plateau)

### Additional Inherited Members

#### 6.26.1 Member Function Documentation

### 6.26.1.1 score()

```
int ScoreVert::score (
    Plateau * plateau ) [override], [virtual]
```

Reimplemented from [Score](#).

### 6.26.1.2 score\_vert()

```
int ScoreVert::score_vert (
    Plateau * plateau ) [private]
```

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

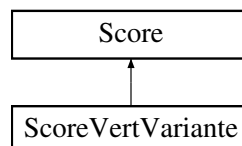
- [Score.hpp](#)

## 6.27 ScoreVertVariante Class Reference

Décorateur qui implémente le scoring pour les points Verts d'un joueur avec variante.

```
#include <Score.hpp>
```

Inheritance diagram for ScoreVertVariante:



### Public Member Functions

- int [score](#) ([Plateau](#) \*plateau) override  
*La fonction de calcul de [Score](#).*

### Private Member Functions

- int [score\\_vert\\_variante](#) ([Plateau](#) \*plateau)

### Additional Inherited Members

#### 6.27.1 Member Function Documentation

### 6.27.1.1 score()

```
int ScoreVertVariante::score (
    Plateau * plateau ) [override], [virtual]
```

Reimplemented from [Score](#).

### 6.27.1.2 score\_vert\_variante()

```
int ScoreVertVariante::score_vert_variante (
    Plateau * plateau ) [private]
```

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

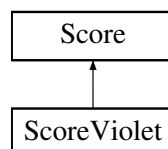
- [Score.hpp](#)

## 6.28 ScoreViolet Class Reference

Décorateur qui implémente le scoring pour les points Violets d'un joueur normal.

```
#include <Score.hpp>
```

Inheritance diagram for ScoreViolet:



### Public Member Functions

- int [score](#) ([Plateau](#) \*plateau) override  
*La fonction de calcul de [Score](#).*

### Private Member Functions

- int [score\\_violet](#) ([Plateau](#) \*plateau)

### Additional Inherited Members

## 6.28.1 Member Function Documentation

### 6.28.1.1 score()

```
int ScoreViolet::score (
    Plateau * plateau ) [override], [virtual]
```

Reimplemented from [Score](#).

### 6.28.1.2 score\_violet()

```
int ScoreViolet::score_violet (
    Plateau * plateau ) [private]
```

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

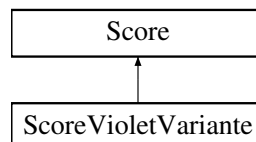
- [Score.hpp](#)

## 6.29 ScoreVioletVariante Class Reference

Décorateur qui implémente le scoring pour les points Violets d'un joueur avec variante.

```
#include <Score.hpp>
```

Inheritance diagram for ScoreVioletVariante:



### Public Member Functions

- int [score](#) ([Plateau](#) \*plateau) override  
*La fonction de calcul de [Score](#).*

### Private Member Functions

- int [score\\_violet\\_variante](#) ([Plateau](#) \*plateau)

### Additional Inherited Members

#### 6.29.1 Member Function Documentation

### 6.29.1.1 score()

```
int ScoreVioletVariante::score (
    Plateau * plateau ) [override], [virtual]
```

Reimplemented from [Score](#).

### 6.29.1.2 score\_violet\_variante()

```
int ScoreVioletVariante::score_violet_variante (
    Plateau * plateau ) [private]
```

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

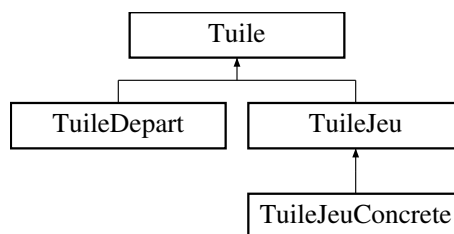
- [Score.hpp](#)

## 6.30 Tuile Class Reference

La classe [Tuile](#), qui définit les tuiles de jeu.

```
#include <Tuile.hpp>
```

Inheritance diagram for Tuile:



### Public Member Functions

- [Tuile](#) ()=default
- [~Tuile](#) ()
- void [set\\_hauteur](#) (int hauteur)  
*Setteur de la Hauteur. À appeler lors de la pose sur le terrain.*
- int [get\\_hauteur](#) () const  
*Getteur de la Hauteur.*
- [Hexagone](#) \* [get\\_enfants](#) ()  
*Getteur des enfants.*
- [Vector2](#) \* [get\\_positions\\_enfants](#) ()  
*Getteur de la position des enfants.*
- int [get\\_nombre\\_enfant](#) ()  
*Getteur du nombre d'enfants de la tuile.*

## Protected Attributes

- int [hauteur](#)  
*La hauteur de la tuile, une fois placée sur le terrain.*
- int [nombre\\_enfants](#)  
*Le nombre d'enfants ([Hexagone](#)) par [Tuile](#).*
- [Hexagone](#) enfants [[max\\_enfants\\_par\\_tuile](#)]  
*La liste des enfants ([Hexagone](#)) de cette [Tuile](#).*
- [Vector2](#) positions\_enfants [[max\\_enfants\\_par\\_tuile](#)]  
*Les positions relatives des enfants ([Hexagone](#)) de cette tuile par rapport à la tuile.*
- int [rotation](#)  
*La rotation de la tuile.*

### 6.30.1 Detailed Description

La classe [Tuile](#), qui définit les tuiles de jeu.

### 6.30.2 Constructor & Destructor Documentation

#### 6.30.2.1 [Tuile\(\)](#)

```
Tuile::Tuile ( ) [default]
```

#### 6.30.2.2 [~Tuile\(\)](#)

```
Tuile::~~Tuile ( ) [inline]
```

### 6.30.3 Member Function Documentation

#### 6.30.3.1 [get\\_enfants\(\)](#)

```
Hexagone * Tuile::get_enfants ( ) [inline]
```

#### 6.30.3.2 [get\\_hauteur\(\)](#)

```
int Tuile::get_hauteur ( ) const [inline]
```

### 6.30.3.3 get\_nombre\_enfant()

```
int Tuile::get_nombre_enfant ( ) [inline]
```

### 6.30.3.4 get\_positions\_enfants()

```
Vector2 * Tuile::get_positions_enfants ( ) [inline]
```

### 6.30.3.5 set\_hauteur()

```
void Tuile::set_hauteur (
    int hauteur ) [inline]
```

## 6.30.4 Member Data Documentation

### 6.30.4.1 enfants

```
Hexagone Tuile::enfants[max_enfants_par_tuile] [protected]
```

### 6.30.4.2 hauteur

```
int Tuile::hauteur [protected]
```

### 6.30.4.3 nombre\_enfants

```
int Tuile::nombre_enfants [protected]
```

### 6.30.4.4 positions\_enfants

```
Vector2 Tuile::positions_enfants[max_enfants_par_tuile] [protected]
```

#### 6.30.4.5 rotation

```
int Tuile::rotation [protected]
```

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

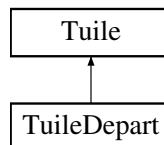
- [Tuile.hpp](#)

## 6.31 TuileDepart Class Reference

Représente une [Tuile](#) de Départ de jeu, ne peut pas être placée lors d'une partie.

```
#include <Tuile.hpp>
```

Inheritance diagram for TuileDepart:



### Additional Inherited Members

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

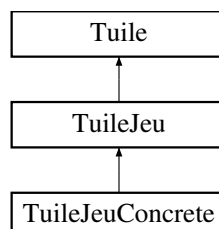
- [Tuile.hpp](#)

## 6.32 TuileJeu Class Reference

Représente une [Tuile](#) de jeu classique.

```
#include <Tuile.hpp>
```

Inheritance diagram for TuileJeu:





## Additional Inherited Members

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

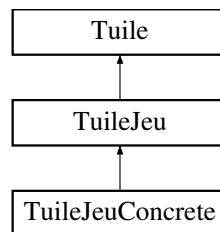
- [Tuile.hpp](#)

## 6.33 TuileJeuConcrete Class Reference

Implémentation concrète de la tuile de Jeu de base.

```
#include <Tuile.hpp>
```

Inheritance diagram for TuileJeuConcrete:



## Additional Inherited Members

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

- [Tuile.hpp](#)

## 6.34 Vector2 Class Reference

Représente un Vecteur 2D.

```
#include <Utils.hpp>
```

## Public Member Functions

- [Vector2](#) ()  
*Initialize un Vecteur nul.*
- [Vector2](#) (float x)  
*Initialize un Vecteur sur l'axe des abscisses.*
- [Vector2](#) (float x, float y)  
*Initialize un Vecteur avec les coordonnées données.*

## Public Attributes

- float [x](#)  
*La coordonnée X du Vecteur.*
- float [y](#)  
*La coordonnée Y du Vecteur.*

## 6.34.1 Constructor & Destructor Documentation

### 6.34.1.1 Vector2() [1/3]

```
Vector2::Vector2 ( ) [inline]
```

Initialize un Vecteur nul

### 6.34.1.2 Vector2() [2/3]

```
Vector2::Vector2 (
    float x ) [inline]
```

Initialize un Vecteur sur l'axe des abscisses

### 6.34.1.3 Vector2() [3/3]

```
Vector2::Vector2 (
    float x,
    float y ) [inline]
```

Initialize un Vecteur avec les coordonnées données

## 6.34.2 Member Data Documentation

### 6.34.2.1 x

```
float Vector2::x
```

La coordonnée X du Vecteur

### 6.34.2.2 y

```
float Vector2::y
```

La coordonnée Y du Vecteur

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

- [Utils.hpp](#)

## Chapter 7

# File Documentation

### 7.1 Affichage.cpp File Reference

```
#include <vector>
#include "Affichage.hpp"
```

### 7.2 Affichage.hpp File Reference

```
#include "../Utils.hpp"
#include "../Players/Players.hpp"
```

#### Classes

- class [Affichage](#)
- class [AffichageConsole](#)

### 7.3 Affichage.hpp

[Go to the documentation of this file.](#)

```
1 #ifndef LO21_PROJET_AFFICHAGE_HPP
2 #define LO21_PROJET_AFFICHAGE_HPP
3
4 #include "../Utils.hpp"
5 #include "../Players/Players.hpp"
6
7
8 class Affichage {
9
10 public:
11     virtual void affiche_plateau_actuel(Joueur& joueur);
12 };
13
14 class AffichageConsole : Affichage{
15     static AffichageConsole* instance;
16     const int hexH = 5; // lignes
17     const int hexW = 9; // colonnes
18
19     AffichageConsole() = default;
```

```
20
21 public:
22
23     static AffichageConsole* getInstance() {
24         if (instance == nullptr) instance = new AffichageConsole();
25         return instance;
26     }
27
28     void affiche_plateau_actuel(Joueur &joueur);
29 };
30
31
32 #endif //LO21_PROJET_AFFICHAGE_HPP
```

## 7.4 Chantier.cpp File Reference

```
#include "Chantier.hpp"
```

## 7.5 Chantier.hpp File Reference

```
#include "../Tuile/Tuile.hpp"
```

### Classes

- class [Chantier](#)

*La classe implémentant le [Chantier](#), permettant aux [Joueur](#) de piocher des [Tuile](#).*

### Variables

- const int [max\\_tuiles\\_par\\_chantier](#) = 5

### 7.5.1 Variable Documentation

#### 7.5.1.1 max\_tuiles\_par\_chantier

```
const int max_tuiles_par_chantier = 5
```

## 7.6 Chantier.hpp

[Go to the documentation of this file.](#)

```
1 #ifndef LO21_PROJET_CHANTIER_HPP
2 #define LO21_PROJET_CHANTIER_HPP
3
4 #include "../Tuile/Tuile.hpp"
5
6 const int max_tuiles_par_chantier = 5;
7
9 class Chantier {
10 private:
12     int taille;
14     int nombreTuiles = 0;
16     Tuile tuiles[max_tuiles_par_chantier];
17 public:
19     void set_nombre_joueurs(int nombre_joueurs);
21     void set_taille(int taille) {this->taille = min(max(taille, 0), max_tuiles_par_chantier);} // Clamp
    entre 0 et max_tuiles_par_chantier
23     int get_taille() {return taille;}
25     int get_nombre_tuiles() {return nombreTuiles;}
27     Tuile* get_tuiles() {return tuiles;}
29
36     Tuile prendre_tuile(int index);
38     void ajouter_tuile(Tuile* tuile);
40     void ajouter_tuile(Tuile* tuile, int nombre);
41 };
42
43
44 #endif //LO21_PROJET_CHANTIER_HPP
```

## 7.7 Deck.cpp File Reference

```
#include "Deck.hpp"
```

## 7.8 Deck.hpp File Reference

```
#include "../Tuile/Tuile.hpp"
```

### Classes

- class [Deck](#)

### Variables

- const int [max\\_tuiles\\_dans\\_deck](#) = 61

### 7.8.1 Variable Documentation

#### 7.8.1.1 max\_tuiles\_dans\_deck

```
const int max_tuiles_dans_deck = 61
```

## 7.9 Deck.hpp

[Go to the documentation of this file.](#)

```
1 #ifndef LO21_PROJET_DECK_HPP
2 #define LO21_PROJET_DECK_HPP
3
4 #include "../Tuile/Tuile.hpp"
5
6 const int max_tuiles_dans_deck = 61;
7
8 class Deck {
9 private:
10     int taille;
11     int nombreTuiles;
12     Tuile tuiles[max_tuiles_dans_deck];
13 public:
14     Deck(int nombre_joueurs);
15     int get_taille();
16     int get_nombre_tuiles();
17     Tuile tirer_tuile();
18     Tuile* tirer_tuile(int nombre_tuiles);
19 };
20
21 #endif
```

## 7.10 main.cpp File Reference

```
#include <iostream>
#include "main.hpp"
#include "Tests/Tests.hpp"
```

### Functions

- int [main](#) ()

### 7.10.1 Function Documentation

#### 7.10.1.1 main()

```
int main ( )
```

## 7.11 main.hpp File Reference

## 7.12 main.hpp

[Go to the documentation of this file.](#)

```
1
2 #ifndef LO21_PROJET_MAIN_HPP
3 #define LO21_PROJET_MAIN_HPP
4
5 #endif //LO21_PROJET_MAIN_HPP
```

## 7.13 Plateau.cpp File Reference

```
#include "Plateau.hpp"
```

## 7.14 Plateau.hpp File Reference

```
#include "../Utils.hpp"
#include "../Tuile/Hexagone.hpp"
#include <iostream>
#include <map>
```

### Classes

- class [Plateau](#)

*La classe implémentant le [Plateau](#) de jeu d'un joueur.*

## 7.15 Plateau.hpp

[Go to the documentation of this file.](#)

```
1 #ifndef LO21_PROJET_MAP_HPP
2 #define LO21_PROJET_MAP_HPP
3
4 #include "../Utils.hpp"
5 #include "../Tuile/Hexagone.hpp"
6 #include <iostream>
7 #include <map>
8 using namespace std;
9
11 class Plateau {
12 private:
14     map<Vector2, Hexagone&> plateau;
15
16 public:
17     Plateau() = default;
18     ~Plateau() = default;
19
20
21
22
27     Hexagone& obtenir_hexagone(const Vector2& coordonnees);
28
29
35     bool peut_placer(const Tuile& tuileJeu, const Vector2& position);
36
37
43     bool placer(const Tuile& tuile, const Vector2& position);
44
46     map<Vector2, Hexagone&>::iterator get_iterateur_debut() {return plateau.begin();};
48     map<Vector2, Hexagone&>::iterator get_iterateur_fin() {return plateau.end();};
49 };
50
51
52 #endif //LO21_PROJET_MAP_HPP
```

## 7.16 Players.hpp File Reference

```
#include "../Score/Score.hpp"
#include "../Tuile/Tuile.hpp"
#include "../Utils.hpp"
```

## Classes

- class [Joueur](#)  
*La classe abstraite [Joueur](#), qui implémente n'importe quel [Joueur](#).*
- class [JoueurSimple](#)  
*Une implémentation concrète d'un [Joueur](#) humain.*
- class [IllustreArchitecte](#)

## 7.17 Players.hpp

[Go to the documentation of this file.](#)

```

1 #ifndef LO21_PROJET_PLAYERS_HPP
2 #define LO21_PROJET_PLAYERS_HPP
3
4 #include "../Score/Score.hpp"
5 #include "../Tuile/Tuile.hpp"
6 #include "../Utils.hpp"
7
8
9
10
11
12
13 class Joueur {
14 protected:
15     int pierre = 0;
16     Plateau& plateauJoueur;
17
18
19
20
21
22
23     bool joueToutSeul;
24
25
26 public:
27     int get_score();
28     int get_pierre();
29     void set_pierre(int pierre);
30     void ajouter_pierre(int pierre);
31     Plateau& get_plateau();
32
33
34     bool place_tuile(Tuile tuile, Vector2 coordonnées);
35
36
37     void jouer(Tuile* chantier);
38 };
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56 class JoueurSimple : public Joueur {
57 protected:
58     Score* scoreJoueur;
59     bool joueToutSeul = false;
60
61 public:
62     int get_score();
63 };
64
65
66
67 class IllustreArchitecte : public Joueur {
68 protected:
69     ScoreSoloArchitecte* score;
70     int niveau = 0;
71     bool joueToutSeul = true;
72     Tuile choisir_tuile(Tuile* chantier);
73     Vector2 trouver_emplacement_tuile(Tuile& tuile);
74
75 public:
76     void set_niveau(int niveau);
77     int get_niveau();
78     void jouer(Tuile* chantier);
79 };
80
81
82
83
84
85
86
87
88 #endif //LO21_PROJET_PLAYERS_HPP

```

## 7.18 README.md File Reference

## 7.19 Score.cpp File Reference

```
#include "Score.hpp"
```



## 7.20 Score.hpp File Reference

```
#include "../Players/Plateau.hpp"
```

### Classes

- class [Score](#)  
*Classe de [Score](#), utilisée pour calculer le score d'un joueur à partir d'un plateau.*
- class [ScoreSoloArchitecte](#)  
*Classe de [Score](#) de l'Illustre Architecte, utilisée pour calculer le score de l'Illustre Architecte à partir d'un plateau.*
- class [ScoreSoloArchitecteBleu](#)  
*Décorateur qui implémente le scoring pour les points Bleus de l'Illustre Architecte.*
- class [ScoreSoloArchitecteRouge](#)  
*Décorateur qui implémente le scoring pour les points Rouges de l'Illustre Architecte.*
- class [ScoreSoloArchitecteJaune](#)  
*Décorateur qui implémente le scoring pour les points Jaunes de l'Illustre Architecte.*
- class [ScoreSoloArchitecteVert](#)  
*Décorateur qui implémente le scoring pour les points Verts de l'Illustre Architecte.*
- class [ScoreSoloArchitecteViolet](#)  
*Décorateur qui implémente le scoring pour les points Violets de l'Illustre Architecte.*
- class [ScoreBleu](#)  
*Décorateur qui implémente le scoring pour les points Bleus d'un joueur normal.*
- class [ScoreRouge](#)  
*Décorateur qui implémente le scoring pour les points Rouges d'un joueur normal.*
- class [ScoreVert](#)  
*Décorateur qui implémente le scoring pour les points Verts d'un joueur normal.*
- class [ScoreViolet](#)  
*Décorateur qui implémente le scoring pour les points Violets d'un joueur normal.*
- class [ScoreJaune](#)  
*Décorateur qui implémente le scoring pour les points Jaunes d'un joueur normal.*
- class [ScoreBleuVariante](#)  
*Décorateur qui implémente le scoring pour les points Bleus d'un joueur avec variante.*
- class [ScoreRougeVariante](#)  
*Décorateur qui implémente le scoring pour les points Rouges d'un joueur avec variante.*
- class [ScoreVertVariante](#)  
*Décorateur qui implémente le scoring pour les points Verts d'un joueur avec variante.*
- class [ScoreVioletVariante](#)  
*Décorateur qui implémente le scoring pour les points Violets d'un joueur avec variante.*
- class [ScoreJauneVariante](#)  
*Décorateur qui implémente le scoring pour les points Jaunes d'un joueur avec variante.*

## 7.21 Score.hpp

[Go to the documentation of this file.](#)

```

1  #ifndef LO21_PROJET_SCORE_HPP
2  #define LO21_PROJET_SCORE_HPP
3
4  #include "../Players/Plateau.hpp"
5
6
7
11 class Score {
12 protected:
14     Score * scoreDecore = nullptr;
15 public:
16     Score() = default;
17     Score(Score* scoreDecore) : scoreDecore(scoreDecore) {}
18     virtual ~Score() = default;
19     virtual int score(Plateau* plateau) {return 0;}
20 };
21
22
23
24
28 class ScoreSoloArchitecte : public Score {
29 protected:
31     int niveau = 0;
32 public:
34     virtual void set_niveau(int niveau);
36     int get_niveau();
37 };
38
39
40 class ScoreSoloArchitecteBleu : public ScoreSoloArchitecte {
41     int score_bleu(Plateau* plateau);
42 public:
43     int score(Plateau* plateau) override;
44 };
45
46
47 class ScoreSoloArchitecteRouge : public ScoreSoloArchitecte {
48     int score_rouge(Plateau* plateau);
49 public:
50     int score(Plateau* plateau) override;
51 };
52
53
54 class ScoreSoloArchitecteJaune : public ScoreSoloArchitecte {
55     int score_jaune(Plateau* plateau);
56 public:
57     int score(Plateau* plateau) override;
58 };
59
60
61 class ScoreSoloArchitecteVert : public ScoreSoloArchitecte {
62     int score_vert(Plateau* plateau);
63 public:
64     int score(Plateau* plateau) override;
65 };
66
67
68 class ScoreSoloArchitecteViolet : public ScoreSoloArchitecte {
69     int score_violet(Plateau* plateau);
70 public:
71     int score(Plateau* plateau) override;
72 };
73
74
75 class ScoreBleu : public Score{
76     int score_bleu(Plateau* plateau);
77 public:
78     int score (Plateau* plateau) override;
79 };
80
81
82 class ScoreRouge : public Score{
83     int score_rouge(Plateau* plateau);
84 public:
85     int score (Plateau* plateau) override;
86 };
87
88
89 class ScoreVert : public Score{
90     int score_vert(Plateau* plateau);
91 public:
92     int score (Plateau* plateau) override;
93 };
94
95
96 class ScoreViolet : public Score{
97     int score_violet(Plateau* plateau);
98 public:
99     int score (Plateau* plateau) override;
100 };
101
102
103 class ScoreJaune : public Score{
104     int score_jaune(Plateau* plateau);
105 public:

```

```

106     int score (Plateau* plateau) override;
107 };
108
109
110 class ScoreBleuVariante : public Score{
111     int score_bleu_variante(Plateau* plateau);
112 public:
113     int score (Plateau* plateau) override;
114 };
115
116 class ScoreRougeVariante : public Score{
117     int score_rouge_variante(Plateau* plateau);
118 public:
119     int score (Plateau* plateau) override;
120 };
121
122 class ScoreVertVariante : public Score{
123     int score_vert_variante(Plateau* plateau);
124 public:
125     int score (Plateau* plateau) override;
126 };
127
128 class ScoreVioletVariante : public Score{
129     int score_violet_variante(Plateau* plateau);
130 public:
131     int score (Plateau* plateau) override;
132 };
133
134 class ScoreJauneVariante : public Score{
135     int score_jaune_variante(Plateau* plateau);
136 public:
137     int score (Plateau* plateau) override;
138 };
139
140 #endif //LO21_PROJET_SCORE_HPP

```

## 7.22 Tests.cpp File Reference

```
#include "Tests.hpp"
```

### Functions

- void `assertTests` ()  
*The main test function.*

### 7.22.1 Function Documentation

#### 7.22.1.1 `assertTests()`

```
void assertTests ( )
```

## 7.23 Tests.hpp File Reference

Test & Assertion function definition.

```
#include <cassert>
```

## Functions

- void `assertTests` ()  
*The main test function.*

### 7.23.1 Detailed Description

#### Date

17/09/2025

#### Author

Dimitri Maréchal

#### Version

1

This file defines the different test/assertion functions.

To disable those tests & assertions, uncomment the `// #define NDEBUB` at the beginning of the `main.cpp` file

### 7.23.2 Function Documentation

#### 7.23.2.1 `assertTests()`

```
void assertTests ( )
```

## 7.24 Tests.hpp

[Go to the documentation of this file.](#)

```
1
12 #ifndef LO21_PROJET_TESTS_HPP
13 #define LO21_PROJET_TESTS_HPP
14
15 #include <cassert>
16
17 void assertTests();
18
19
20
21 #endif
```

## 7.25 Hexagone.cpp File Reference

```
#include "Tuile.hpp"
```

## 7.26 Hexagone.hpp File Reference

```
#include <iostream>
#include "../Utils.hpp"
```

### Classes

- class [Hexagone](#)  
*La classe [Hexagone](#) : le bloc de base de Akropolis.*
- class [Place](#)  
*La classe [Place](#).*
- class [Carriere](#)  
*La classe [Carriere](#).*
- class [Quartier](#)  
*La classe [Quartier](#).*

## 7.27 Hexagone.hpp

[Go to the documentation of this file.](#)

```
1 #ifndef LO21_PROJET_HEXAGONE_HPP
2 #define LO21_PROJET_HEXAGONE_HPP
3
4 #include <iostream>
5 #include "../Utils.hpp"
6
7 // Sont définis dans d'autres fichiers en-tête, qui importent Hexagone
8 // on les définit de façon inline pour ne pas causer d'erreurs
9 class Tuile; class Joueur; class Plateau;
10 using namespace std;
11
12
13
14
15
16
17
18
19 class Hexagone {
20 protected:
21     enum CouleursAkropolis couleur;
22     Tuile* tuileParent;
23     int indice_tuile;
24
25 public :
26     Hexagone() = default;
27     ~Hexagone() = default;
28
29     int get_couleur() const;
30     Tuile* get_tuile() const;
31     Vector2 get_local_position() const;
32
33     int get_hauteur() const;
34
35     bool peut_etre_placee(Plateau* map, Vector2 position) const;
36
37     void quand_recouvert(Joueur* joueur_qui_recouvre) const;
38
39     string get_displayed_text() const {return "default";};
40 };
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79 class Place : public Hexagone{
80 protected:
81
82     int etoiles;
83
84 public:
85     Place() = default;
86     ~Place() {std::cout<<"Place détruite";};
87
88     int get_etoiles();
89
90
91
92 }
```

```
93 };
94
96
99 class Carriere : public Hexagone{
100
101 };
102
104
107 class Quartier : public Hexagone{
108
109 };
110
111
112 #endif //LO21_PROJET_HEXAGONE_HPP
```

## 7.28 Tuile.cpp File Reference

```
#include <iostream>
#include "Tuile.hpp"
```

## 7.29 Tuile.hpp File Reference

```
#include <iostream>
#include "Hexagone.hpp"
#include "../Utils.hpp"
```

### Classes

- class [Tuile](#)  
*La classe [Tuile](#), qui définit les tuiles de jeu.*
- class [TuileDepart](#)  
*Représente une [Tuile](#) de Départ de jeu, ne peut pas être placée lors d'une partie.*
- class [TuileJeu](#)  
*Représente une [Tuile](#) de jeu classique.*
- class [TuileJeuConcrete](#)  
*Implémentation concrète de la tuile de Jeu de base.*

### Variables

- const int [max\\_enfants\\_par\\_tuile](#) = 10  
*Le nombre maximal d'enfants par [Tuile](#), est utilisé pour définir la taille des Tableaux de [Tuile](#).*

### 7.29.1 Variable Documentation

#### 7.29.1.1 max\_enfants\_par\_tuile

```
const int max_enfants_par_tuile = 10
```

## 7.30 Tuile.hpp

[Go to the documentation of this file.](#)

```
1 #ifndef LO21_PROJET_TUILE_HPP
2 #define LO21_PROJET_TUILE_HPP
3
4 #include <iostream>
5 #include "Hexagone.hpp"
6 #include "../Utils.hpp"
7
8
9 const int max_enfants_par_tuile = 10;
10
11
12
13
14
15 class Tuile {
16 protected:
17     int hauteur;
18     int nombre_enfants;
19     Hexagone enfants[max_enfants_par_tuile];
20     Vector2 positions_enfants[max_enfants_par_tuile];
21     int rotation;
22
23 public :
24     Tuile() = default;
25     ~Tuile() {std::cout<<"Tuile détruite";};
26
27     void set_hauteur(int hauteur) {hauteur = max(hauteur, 0);}
28     int get_hauteur() const {return hauteur;}
29     Hexagone* get_enfants() {return enfants;}
30     Vector2* get_positions_enfants() {return positions_enfants;}
31     int get_nombre_enfant() {return nombre_enfants;}
32 };
33
34
35 class TuileDepart : public Tuile {
36
37 };
38
39 class TuileJeu : public Tuile {
40
41 };
42
43 class TuileJeuConcrete : public TuileJeu {
44
45 };
46
47 #endif
```

## 7.31 Utils.hpp File Reference

### Classes

- class [Vector2](#)  
*Représente un Vecteur 2D.*

### Enumerations

- enum [CouleursAkropolis](#) {  
BLANC , BLEU , JAUNE , ROUGE ,  
VIOLET , VERT }  
*Définit les couleurs des tuiles d'Akropolis.*

### 7.31.1 Enumeration Type Documentation

#### 7.31.1.1 CouleursAkropolis

```
enum CouleursAkropolis
```

## Enumerator

BLANC	
BLEU	
JAUNE	
ROUGE	
VIOLET	
VERT	

## 7.32 Utils.hpp

[Go to the documentation of this file.](#)

```
1 #ifndef LO21_PROJET_UTILS_HPP
2 #define LO21_PROJET_UTILS_HPP
3
4
5 enum CouleursAkropolis {
6     BLANC,
7     BLEU,
8     JAUNE,
9     ROUGE,
10    VIOLET,
11    VERT
12 };
13
14
15 class Vector2 {
16 public:
17
18     float x;
19
20     float y;
21
22
23
24
25     Vector2() : x(0), y(0) {};
26
27     Vector2(float x) : x(x), y(0) {};
28
29     Vector2(float x, float y) : x(x), y(y) {};
30 };
31
32 #endif
```



# Index

- ~Hexagone
  - Hexagone, [19](#)
- ~Place
  - Place, [28](#)
- ~Plateau
  - Plateau, [30](#)
- ~Score
  - Score, [34](#)
- ~Tuile
  - Tuile, [52](#)
- Affichage, [11](#)
  - affiche\_plateau\_actuel, [11](#)
- Affichage.cpp, [57](#)
- Affichage.hpp, [57](#)
- AffichageConsole, [12](#)
  - AffichageConsole, [12](#)
  - affiche\_plateau\_actuel, [12](#)
  - getInstance, [13](#)
  - hexH, [13](#)
  - hexW, [13](#)
  - instance, [13](#)
- affiche\_plateau\_actuel
  - Affichage, [11](#)
  - AffichageConsole, [12](#)
- ajouter\_pierre
  - Joueur, [24](#)
- ajouter\_tuile
  - Chantier, [14](#), [15](#)
- assertTests
  - Tests.cpp, [65](#)
  - Tests.hpp, [66](#)
- BLANC
  - Utils.hpp, [70](#)
- BLEU
  - Utils.hpp, [70](#)
- Carriere, [13](#)
- Chantier, [14](#)
  - ajouter\_tuile, [14](#), [15](#)
  - get\_nombre\_tuiles, [15](#)
  - get\_taille, [15](#)
  - get\_tuiles, [15](#)
  - nombreTuiles, [16](#)
  - prendre\_tuile, [15](#)
  - set\_nombre\_joueurs, [16](#)
  - set\_taille, [16](#)
  - taille, [16](#)
  - tuiles, [16](#)
- Chantier.cpp, [58](#)
- Chantier.hpp, [58](#), [59](#)
  - max\_tuiles\_par\_chantier, [58](#)
- choisir\_tuile
  - IllustreArchitecte, [22](#)
- couleur
  - Hexagone, [21](#)
- CouleursAkropolis
  - Utils.hpp, [69](#)
- Deck, [16](#)
  - Deck, [17](#)
  - get\_nombre\_tuiles, [17](#)
  - get\_taille, [17](#)
  - nombreTuiles, [18](#)
  - taille, [18](#)
  - tirer\_tuile, [17](#), [18](#)
  - tuiles, [18](#)
- Deck.cpp, [59](#)
- Deck.hpp, [59](#), [60](#)
  - max\_tuiles\_dans\_deck, [59](#)
- enfants
  - Tuile, [53](#)
- etoiles
  - Place, [29](#)
- get\_couleur
  - Hexagone, [20](#)
- get\_displayed\_text
  - Hexagone, [20](#)
- get\_enfants
  - Tuile, [52](#)
- get\_etoiles
  - Place, [29](#)
- get\_hauteur
  - Hexagone, [20](#)
  - Tuile, [52](#)
- get\_iterateur\_debut
  - Plateau, [30](#)
- get\_iterateur\_fin
  - Plateau, [30](#)
- get\_local\_position
  - Hexagone, [20](#)
- get\_niveau
  - IllustreArchitecte, [22](#)
  - ScoreSoloArchitecte, [41](#)
- get\_nombre\_enfant
  - Tuile, [52](#)
- get\_nombre\_tuiles

- Chantier, 15
- Deck, 17
- get\_pierre
  - Joueur, 25
- get\_plateau
  - Joueur, 25
- get\_positions\_enfants
  - Tuile, 53
- get\_score
  - Joueur, 25
  - JoueurSimple, 27
- get\_taille
  - Chantier, 15
  - Deck, 17
- get\_tuile
  - Hexagone, 20
- get\_tuiles
  - Chantier, 15
- getInstance
  - AffichageConsole, 13
- hauteur
  - Tuile, 53
- Hexagone, 18
  - ~Hexagone, 19
  - couleur, 21
  - get\_couleur, 20
  - get\_displayed\_text, 20
  - get\_hauteur, 20
  - get\_local\_position, 20
  - get\_tuile, 20
  - Hexagone, 19
  - indice\_tuile, 21
  - peut\_etre\_placee, 20
  - quand\_recouvert, 21
  - tuileParent, 21
- Hexagone.cpp, 66
- Hexagone.hpp, 67
- hexH
  - AffichageConsole, 13
- hexW
  - AffichageConsole, 13
- IllustreArchitecte, 22
  - choisir\_tuile, 22
  - get\_niveau, 22
  - jouer, 23
  - joueToutSeul, 23
  - niveau, 23
  - score, 23
  - set\_niveau, 23
  - trouver\_emplacement\_tuile, 23
- indice\_tuile
  - Hexagone, 21
- instance
  - AffichageConsole, 13
- JAUNE
  - Utils.hpp, 70
- jouer
  - IllustreArchitecte, 23
  - Joueur, 25
- joueToutSeul
  - IllustreArchitecte, 23
  - Joueur, 26
  - JoueurSimple, 27
- Joueur, 24
  - ajouter\_pierre, 24
  - get\_pierre, 25
  - get\_plateau, 25
  - get\_score, 25
  - jouer, 25
  - joueToutSeul, 26
  - pierre, 26
  - place\_tuile, 25
  - plateauJoueur, 26
  - set\_pierre, 26
- JoueurSimple, 26
  - get\_score, 27
  - joueToutSeul, 27
  - scoreJoueur, 27
- main
  - main.cpp, 60
- main.cpp, 60
  - main, 60
- main.hpp, 60
- max\_enfants\_par\_tuile
  - Tuile.hpp, 68
- max\_tuiles\_dans\_deck
  - Deck.hpp, 59
- max\_tuiles\_par\_chantier
  - Chantier.hpp, 58
- niveau
  - IllustreArchitecte, 23
  - ScoreSoloArchitecte, 42
- nombre\_enfants
  - Tuile, 53
- nombreTuiles
  - Chantier, 16
  - Deck, 18
- obtenir\_hexagone
  - Plateau, 30
- peut\_etre\_placee
  - Hexagone, 20
- peut\_placer
  - Plateau, 31
- pierre
  - Joueur, 26
- Place, 28
  - ~Place, 28
  - etoiles, 29
  - get\_etoiles, 29
  - Place, 28
- place\_tuile

- Joueur, [25](#)
- placer
  - Plateau, [31](#)
- Plateau, [29](#)
  - ~Plateau, [30](#)
  - get\_iterateur\_debut, [30](#)
  - get\_iterateur\_fin, [30](#)
  - obtenir\_hexagone, [30](#)
  - peut\_placer, [31](#)
  - placer, [31](#)
  - Plateau, [30](#)
  - plateau, [31](#)
- plateau
  - Plateau, [31](#)
- Plateau.cpp, [61](#)
- Plateau.hpp, [61](#)
- plateauJoueur
  - Joueur, [26](#)
- Players.hpp, [61](#), [62](#)
- positions\_enfants
  - Tuile, [53](#)
- prendre\_tuile
  - Chantier, [15](#)
- quand\_recouvert
  - Hexagone, [21](#)
- Quartier, [32](#)
- README.md, [62](#)
- rotation
  - Tuile, [53](#)
- ROUGE
  - Utils.hpp, [70](#)
- Score, [32](#)
  - ~Score, [34](#)
  - Score, [33](#), [34](#)
  - score, [34](#)
  - scoreDecore, [34](#)
- score
  - IllustreArchitecte, [23](#)
  - Score, [34](#)
  - ScoreBleu, [35](#)
  - ScoreBleuVariante, [36](#)
  - ScoreJaune, [37](#)
  - ScoreJauneVariante, [38](#)
  - ScoreRouge, [39](#)
  - ScoreRougeVariante, [40](#)
  - ScoreSoloArchitecteBleu, [42](#)
  - ScoreSoloArchitecteJaune, [43](#)
  - ScoreSoloArchitecteRouge, [44](#)
  - ScoreSoloArchitecteVert, [45](#)
  - ScoreSoloArchitecteViolet, [46](#)
  - ScoreVert, [47](#)
  - ScoreVertVariante, [48](#)
  - ScoreViolet, [49](#)
  - ScoreVioletVariante, [50](#)
- score\_bleu
  - ScoreBleu, [35](#)
  - ScoreSoloArchitecteBleu, [43](#)
- score\_bleu\_variante
  - ScoreBleuVariante, [36](#)
- score\_jaune
  - ScoreJaune, [37](#)
  - ScoreSoloArchitecteJaune, [44](#)
- score\_jaune\_variante
  - ScoreJauneVariante, [38](#)
- score\_rouge
  - ScoreRouge, [39](#)
  - ScoreSoloArchitecteRouge, [45](#)
- score\_rouge\_variante
  - ScoreRougeVariante, [40](#)
- score\_vert
  - ScoreSoloArchitecteVert, [46](#)
  - ScoreVert, [48](#)
- score\_vert\_variante
  - ScoreVertVariante, [49](#)
- score\_violet
  - ScoreSoloArchitecteViolet, [47](#)
  - ScoreViolet, [50](#)
- score\_violet\_variante
  - ScoreVioletVariante, [51](#)
- ScoreBleu, [35](#)
  - score, [35](#)
  - score\_bleu, [35](#)
- ScoreBleuVariante, [36](#)
  - score, [36](#)
  - score\_bleu\_variante, [36](#)
- scoreDecore
  - Score, [34](#)
- ScoreJaune, [37](#)
  - score, [37](#)
  - score\_jaune, [37](#)
- ScoreJauneVariante, [38](#)
  - score, [38](#)
  - score\_jaune\_variante, [38](#)
- scoreJoueur
  - JoueurSimple, [27](#)
- ScoreRouge, [39](#)
  - score, [39](#)
  - score\_rouge, [39](#)
- ScoreRougeVariante, [40](#)
  - score, [40](#)
  - score\_rouge\_variante, [40](#)
- ScoreSoloArchitecte, [41](#)
  - get\_niveau, [41](#)
  - niveau, [42](#)
  - set\_niveau, [41](#)
- ScoreSoloArchitecteBleu, [42](#)
  - score, [42](#)
  - score\_bleu, [43](#)
- ScoreSoloArchitecteJaune, [43](#)
  - score, [43](#)
  - score\_jaune, [44](#)
- ScoreSoloArchitecteRouge, [44](#)

- score, [44](#)
- score\_rouge, [45](#)
- ScoreSoloArchitechteVert, [45](#)
  - score, [45](#)
  - score\_vert, [46](#)
- ScoreSoloArchitechteViolet, [46](#)
  - score, [46](#)
  - score\_violet, [47](#)
- ScoreVert, [47](#)
  - score, [47](#)
  - score\_vert, [48](#)
- ScoreVertVariante, [48](#)
  - score, [48](#)
  - score\_vert\_variante, [49](#)
- ScoreViolet, [49](#)
  - score, [49](#)
  - score\_violet, [50](#)
- ScoreVioletVariante, [50](#)
  - score, [50](#)
  - score\_violet\_variante, [51](#)
- set\_hauteur
  - Tuile, [53](#)
- set\_niveau
  - IllustreArchitecte, [23](#)
  - ScoreSoloArchitechte, [41](#)
- set\_nombre\_joueurs
  - Chantier, [16](#)
- set\_pierre
  - Joueur, [26](#)
- set\_taille
  - Chantier, [16](#)
- taille
  - Chantier, [16](#)
  - Deck, [18](#)
- Tests.cpp, [65](#)
  - assertTests, [65](#)
- Tests.hpp, [65](#), [66](#)
  - assertTests, [66](#)
- tirer\_tuile
  - Deck, [17](#), [18](#)
- trouver\_emplacement\_tuile
  - IllustreArchitecte, [23](#)
- Tuile, [51](#)
  - ~Tuile, [52](#)
  - enfants, [53](#)
  - get\_enfants, [52](#)
  - get\_hauteur, [52](#)
  - get\_nombre\_enfant, [52](#)
  - get\_positions\_enfants, [53](#)
  - hauteur, [53](#)
  - nombre\_enfants, [53](#)
  - positions\_enfants, [53](#)
  - rotation, [53](#)
  - set\_hauteur, [53](#)
  - Tuile, [52](#)
- Tuile.cpp, [68](#)
- Tuile.hpp, [68](#), [69](#)
  - max\_enfants\_par\_tuile, [68](#)
- TuileDepart, [54](#)
- TuileJeu, [54](#)
- TuileJeuConcrete, [55](#)
- tuileParent
  - Hexagone, [21](#)
- tuiles
  - Chantier, [16](#)
  - Deck, [18](#)
- Utils.hpp, [69](#)
  - BLANC, [70](#)
  - BLEU, [70](#)
  - CouleursAkropolis, [69](#)
  - JAUNE, [70](#)
  - ROUGE, [70](#)
  - VERT, [70](#)
  - VIOLET, [70](#)
- Vector2, [55](#)
  - Vector2, [56](#)
    - x, [56](#)
    - y, [56](#)
- VERT
  - Utils.hpp, [70](#)
- VIOLET
  - Utils.hpp, [70](#)
- x
  - Vector2, [56](#)
- y
  - Vector2, [56](#)