TP LO11 Example

Generated by Doxygen 1.9.5

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 compréhention 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 celà, je propose de faire une batterie de test pour chaque unité de code, c'est à dire faire des assert pour chaque fonction/fonctionalité, avant que celle-ci soit implémenté histoire qu'on soit tou.te.s sur un même pied concernant la fonctionalité des fonctions. Et celà nous permettera 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:

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

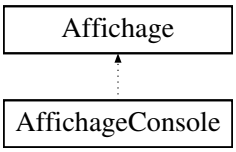
Chapter 6

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.

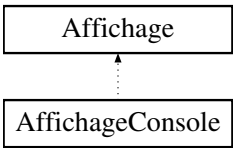
The documentation for this class was generated from the following file:

- 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:

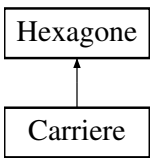
- Affichage.hpp
- Affichage.cpp

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 vons "descendre" le chantier comme un mechanisme de rivière.

Parameters

index	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:

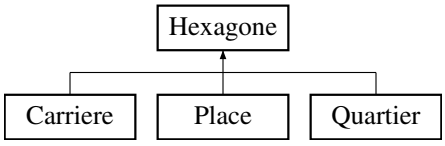
- Deck.hpp

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

map	Un pointeur vers le plateau dans lequel on va vouloir placer l'hexagone
position	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 emplacemen 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

joueur_qui_recouvre	un pointeur vers le Joueur qui recouvre la tuile. Est utilisé pour modifier des attributs du joueur. Cette fonction est appelée dès que l'hexagone est recouvert. Elle sert notemment à la classe Carrière à ajouter une pierre au joureur.

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]
```

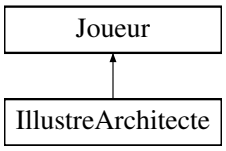
The documentation for this class was generated from the following files:

- 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 Architechte.

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

- ScoreSoloArchitechte * score

  Le Score de l'illustre architechte, il va être utilisé pour calculer son score à la fun de la partie.
- int niveau = 0

  Le niveau de l'Architechte (va changer les regles)
- bool joueToutSeul = true

  L'illustre Architechte 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

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

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

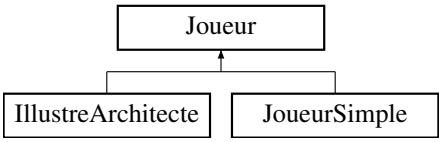
- Players.hpp

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 coordonées)

  Tente de placer une tuile aux coordonées 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'IllustreArchitechte)

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

chantier	un tableau de Tuile, représentant le chantier

6.8.2.6 place_tuile()

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

Parameters

tuile	la Tuile à placer
coordonées	le Vector2 qui donne la position de la Tuile

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:

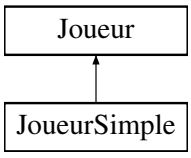
- Players.hpp

6.9 JoueurSimple Class Reference

Une impléméntation 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:

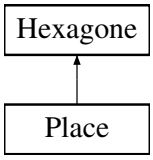
- Players.hpp

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 coordonées 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 coordonée 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 coordonées du Vector2

Parameters

coordonees	le Vector2 des coordonées de l'hexagone recherché

Returns

l'hexagone à ces coordonées

6.11.2.4 peut_placer()

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

Parameters

tuileJeu	La Tuile qu'on essaye de placer
position	La position à laquelle on essaye de placer la Tuile

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

tuile	La Tuile à placer
position	La position où l'on essaye de placer la Tuile

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:

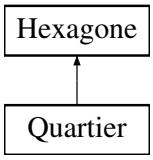
- Plateau.hpp

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:

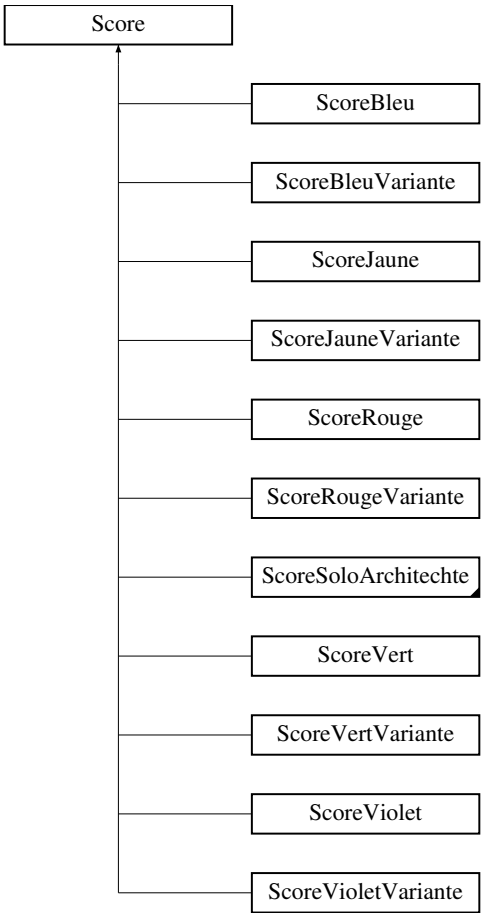
- Hexagone.hpp

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 ScoreSoloArchitechteBleu, ScoreSoloArchitechteRouge, ScoreSoloArchitechteJaune, ScoreSoloArchitechteVert, ScoreSoloArchitechteViolet, 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:

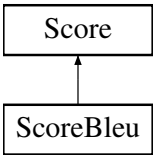
- Score.hpp

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:

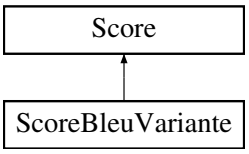
- Score.hpp

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:

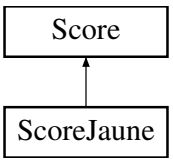
- Score.hpp

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:

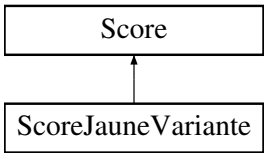
- Score.hpp

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:

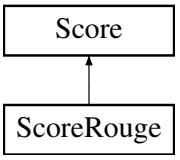
- Score.hpp

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:

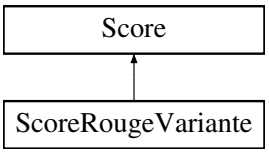
- Score.hpp

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:

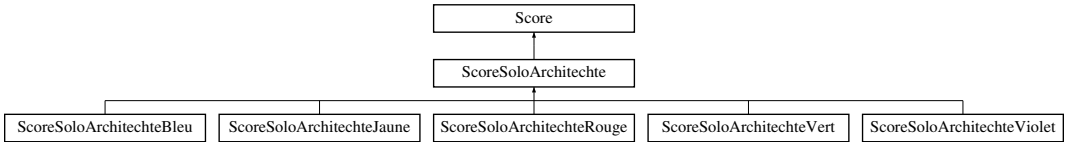
- Score.hpp

6.20 ScoreSoloArchitechte Class Reference

Classe de Score de l'Illustre Architechte, utilisée pour calculer le score de l'Illustre Architechte à partir d'un plateau.

`#include <Score.hpp>`

Inheritance diagram for ScoreSoloArchitechte:



Public Member Functions

- virtual void set_niveau (int niveau)

  Le setteur de niveau de l'Illustre Architechte.
- int get_niveau ()

  Le getteur de niveau de l'Illustre Architechte.

Protected Attributes

- int niveau = 0

  Le niveau de l'Illustre Architechte, sera utilisé pour calculer son score.

6.20.1 Detailed Description

Classe de Score de l'Illustre Architechte, utilisée pour calculer le score de l'Illustre Architechte 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 ScoreSoloArchitechte::get_niveau ( )
```

6.20.2.2 set_niveau()

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

6.20.3 Member Data Documentation

6.20.3.1 niveau

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

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

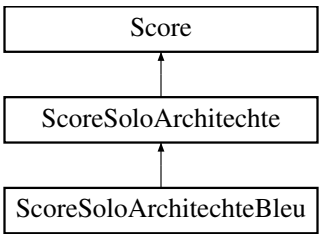
- Score.hpp

6.21 ScoreSoloArchitechteBleu Class Reference

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

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

Inheritance diagram for ScoreSoloArchitechteBleu:



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

Reimplemented from Score.

6.21.1.2 score_bleu()

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

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

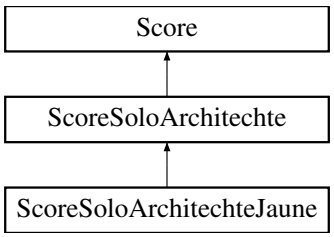
- Score.hpp

6.22 ScoreSoloArchitechteJaune Class Reference

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

`#include <Score.hpp>`

Inheritance diagram for ScoreSoloArchitechteJaune:



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

Reimplemented from Score.

6.22.1.2 score_jaune()

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

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

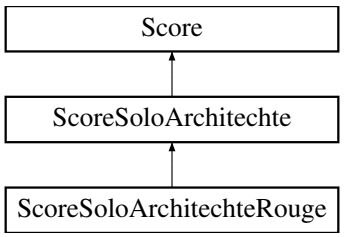
- Score.hpp

6.23 ScoreSoloArchitechteRouge Class Reference

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

`#include <Score.hpp>`

Inheritance diagram for ScoreSoloArchitechteRouge:



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

Reimplemented from Score.

6.23.1.2 score_rouge()

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

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

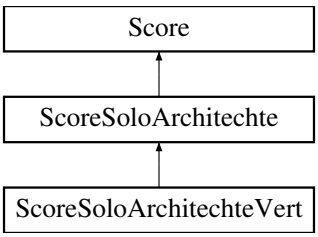
- Score.hpp

6.24 ScoreSoloArchitechteVert Class Reference

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

`#include <Score.hpp>`

Inheritance diagram for ScoreSoloArchitechteVert:



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

Reimplemented from Score.

6.24.1.2 score_vert()

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

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

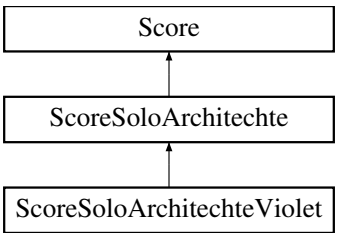
- Score.hpp

6.25 ScoreSoloArchitechteViolet Class Reference

Décorateur qui implémente le scoring pour les points Violets de l'Illustre Architechte.

`#include <Score.hpp>`

Inheritance diagram for ScoreSoloArchitechteViolet:



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

Reimplemented from Score.

6.25.1.2 score_violet()

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

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

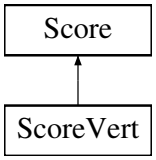
- Score.hpp

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:

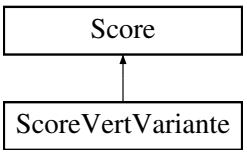
- Score.hpp

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:

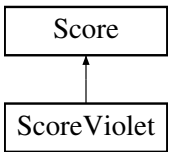
- Score.hpp

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:

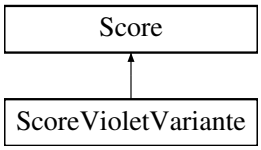
- Score.hpp

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:

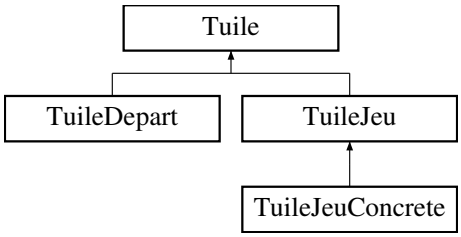
- Score.hpp

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:

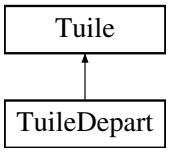
- Tuile.hpp

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:

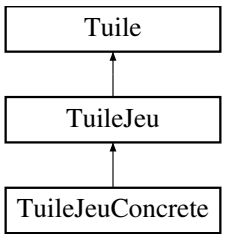
- Tuile.hpp

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:

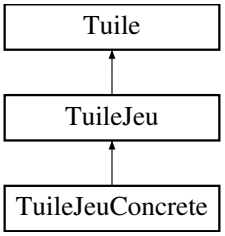
- Tuile.hpp

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 coordonées données.

Public Attributes

- float x

  La coordonée X du Vecteur.
- float y

  La coordonée 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 coordonées donées

6.34.2 Member Data Documentation

6.34.2.1 x

```
float Vector2::x
```

La coordonée X du Vecteur

6.34.2.2 y

```
float Vector2::y
```

La coordonée 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:
11     int taille;
13     int nombreTuiles;
15     Tuile tuiles[max_tuiles_dans_deck];
16 public:
18     Deck(int nombre_joueurs);
20     int get_taille();
22     int get_nombre_tuiles();
24     Tuile tirer_tuile();
26     Tuile* tirer_tuile(int nombre_tuiles);
27 };
28
29
30 #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
12 #ifndef LO21_PROJET_MAIN_HPP
13 #define LO21_PROJET_MAIN_HPP
14
15 #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;
20
22
27     Hexagone& obtenir_hexagone(const Vector2& coordonees);
29
35     bool peut_placer(const Tuile& tuileJeu, const Vector2& position);
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 impléméntation 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
9
13 class Joueur {
14 protected:
16     int pierre = 0;
18     Plateau& plateauJoueur;
20
24     bool joueToutSeul;
25
26 public:
28     int get_score();
30     int get_pierre();
32     void set_pierre(int pierre);
34     void ajouter_pierre(int pierre);
36     Plateau& get_plateau();
38
43     bool place_tuile(Tuile tuile, Vector2 coordonées);
45
49     void jouer(Tuile* chantier);
50 };
51
53
56 class JoueurSimple : public Joueur {
57 protected:
59     Score* scoreJoueur;
61     bool joueToutSeul = false;
62 public:
64     int get_score();
65 };
66
67 class IllustreArchitecte : public Joueur {
68 protected:
70     ScoreSoloArchitechte* score;
72     int niveau = 0;
74     bool joueToutSeul = true;
76     Tuile choisir_tuile(Tuile* chantier);
78     Vector2 trouver_emplacement_tuile(Tuile& tuile);
79 public:
81     void set_niveau(int niveau);
83     int get_niveau();
85     void jouer(Tuile* chantier);
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 ScoreSoloArchitechte

  Classe de Score de l'Illustre Architechte, utilisée pour calculer le score de l'Illustre Architechte à partir d'un plateau.
- class ScoreSoloArchitechteBleu

  Décorateur qui implémente le scoring pour les points Bleus de l'Illustre Architechte.
- class ScoreSoloArchitechteRouge

  Décorateur qui implémente le scoring pour les points Rouges de l'Illustre Architechte.
- class ScoreSoloArchitechteJaune

  Décorateur qui implémente le scoring pour les points Jaunes de l'Illustre Architechte.
- class ScoreSoloArchitechteVert

  Décorateur qui implémente le scoring pour les points Verts de l'Illustre Architechte.
- class ScoreSoloArchitechteViolet

  Décorateur qui implémente le scoring pour les points Violets de l'Illustre Architechte.
- 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
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;
20     virtual int score(Plateau* plateau) {return 0;}
21 };
22
24
28 class ScoreSoloArchitechte : public Score {
29 protected:
31     int niveau = 0;
32 public:
34     virtual void set_niveau(int niveau);
36     int get_niveau();
37 };
38
40 class ScoreSoloArchitechteBleu : public ScoreSoloArchitechte {
41     int score_bleu(Plateau* plateau);
42 public:
43     int score(Plateau* plateau) override;
44 };
45
47 class ScoreSoloArchitechteRouge : public ScoreSoloArchitechte {
48     int score_rouge(Plateau* plateau);
49 public:
50     int score(Plateau* plateau) override;
51 };
52
54 class ScoreSoloArchitechteJaune : public ScoreSoloArchitechte {
55     int score_jaune(Plateau* plateau);
56 public:
57     int score(Plateau* plateau) override;
58 };
59
61 class ScoreSoloArchitechteVert : public ScoreSoloArchitechte {
62     int score_vert(Plateau* plateau);
63 public:
64     int score(Plateau* plateau) override;
65 };
66
68 class ScoreSoloArchitechteViolet : public ScoreSoloArchitechte {
69     int score_violet(Plateau* plateau);
70 public:
71     int score(Plateau* plateau) override;
72 };
73
75 class ScoreBleu : public Score{
76     int score_bleu(Plateau* plateau);
77 public:
78     int score (Plateau* plateau) override;
79 };
80
82 class ScoreRouge : public Score{
83     int score_rouge(Plateau* plateau);
84 public:
85     int score (Plateau* plateau) override;
86 };
87
89 class ScoreVert : public Score{
90     int score_vert(Plateau* plateau);
91 public:
92     int score (Plateau* plateau) override;
93 };
94
96 class ScoreViolet : public Score{
97     int score_violet(Plateau* plateau);
98 public:
99     int score (Plateau* plateau) override;
100 };
101
103 class ScoreJaune : public Score{
104     int score_jaune(Plateau* plateau);
105 public:
```

```
106     int score (Plateau* plateau) override;
107 };
108
109
111 class ScoreBleuVariante : public Score{
112     int score_bleu_variante(Plateau* plateau);
113 public:
114     int score (Plateau* plateau) override;
115 };
116
118 class ScoreRougeVariante : public Score{
119     int score_rouge_variante(Plateau* plateau);
120 public:
121     int score (Plateau* plateau) override;
122 };
123
125 class ScoreVertVariante : public Score{
126     int score_vert_variante(Plateau* plateau);
127 public:
128     int score (Plateau* plateau) override;
129 };
130
132 class ScoreVioletVariante : public Score{
133     int score_violet_variante(Plateau* plateau);
134 public:
135     int score (Plateau* plateau) override;
136 };
137
139 class ScoreJauneVariante : public Score{
140     int score_jaune_variante(Plateau* plateau);
141 public:
142     int score (Plateau* plateau) override;
143 };
144
145
146 #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 NDEBUG` 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
18 void assertTests();
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
13
19 class Hexagone {
20 protected:
22     enum CouleursAkropolis couleur;
24     Tuile* tuileParent;
26     int indice_tuile;
27
28 public :
29     Hexagone() = default;
30     ~Hexagone() = default;
31
33     int get_couleur() const;
35     Tuile* get_tuile() const;
37     Vector2 get_local_position() const;
39
43     int get_hauteur() const;
44
46
55     bool peut_etre_placee(Plateau* map, Vector2 position) const;
56
58
63     void quand_recouvert(Joueur* joueur_qui_recouvre) const;
64
66
72     string get_displayed_text() const {return "default";};
73 };
74
76
79 class Place : public Hexagone{
80 protected:
82
85     int etoiles;
86
87 public:
88     Place() = default;
89     ~Place() {std::cout<<"Place détruite";};
90
92     int get_etoiles();
```

```
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
9 const int max_enfants_par_tuile = 10;
10
12
15 class Tuile {
16 protected:
18     int hauteur;
20     int nombre_enfants;
22     Hexagone enfants[max_enfants_par_tuile];
24     Vector2 positions_enfants[max_enfants_par_tuile];
26     int rotation;
27
28 public :
29     Tuile() = default;
30     ~Tuile() {std::cout«"Tuile détruite";};
31
33     void set_hauteur(int hauteur) {hauteur = max(hauteur, 0);}
35     int get_hauteur() const {return hauteur;}
37     Hexagone* get_enfants() {return enfants;}
39     Vector2* get_positions_enfants() {return positions_enfants;}
41     int get_nombre_enfant() {return nombre_enfants;}
42 };
43
45 class TuileDepart : public Tuile {
46
47 };
48
50 class TuileJeu : public Tuile {
51 };
52
54 class TuileJeuConcrete : public TuileJeu {
55 };
56
57 #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
5 enum CouleursAkropolis {
6     BLANC,
7     BLEU,
8     JAUNE,
9     ROUGE,
10     VIOLET,
11     VERT
12 };
13
15 class Vector2 {
16 public:
18
19     float x;
21
22     float y;
23
25
26     Vector2() : x(0), y(0) {};
28
29     Vector2(float x) : x(x), y(0) {};
31
32     Vector2(float x, float y) : x(x), y(y) {};
33 };
34
35 #endif
```

Index