

Projet LO21 - UTC - P22

Projet de l'UV LO21 réalisé par DU Sylvain, BOUZAR Massil, OUEDRAOGO Taoufiq,
THIBAUT Ambroise et TAVERNY Kelyan

| | | |
|----------|---|-----------|
| 1 | Projet Carcassonne - Conception et développement : d'une application pour jouer au jeu Carcassonne | 1 |
| 2 | Index des espaces de nommage | 3 |
| 2.1 | Liste des espaces de nommage | 3 |
| 3 | Index hiérarchique | 5 |
| 3.1 | Hiérarchie des classes | 5 |
| 4 | Index des classes | 7 |
| 4.1 | Liste des classes | 7 |
| 5 | Index des fichiers | 9 |
| 5.1 | Liste des fichiers | 9 |
| 6 | Documentation des espaces de nommage | 11 |
| 6.1 | Référence de l'espace de nommage std | 11 |
| 6.1.1 | Description détaillée | 11 |
| 7 | Documentation des classes | 13 |
| 7.1 | Référence de la classe Abbe | 13 |
| 7.1.1 | Documentation des fonctions membres | 13 |
| 7.1.1.1 | affichage() | 13 |
| 7.1.1.2 | validationPlacementM() | 13 |
| 7.1.1.3 | validationPlacementT() | 14 |
| 7.2 | Référence de la classe AubergesEtCathedrales | 14 |
| 7.2.1 | Documentation des fonctions membres | 15 |
| 7.2.1.1 | affichage() | 15 |
| 7.2.1.2 | validationPlacementM() | 15 |
| 7.2.1.3 | validationPlacementT() | 15 |
| 7.3 | Référence de la classe ContenanceTuile | 16 |
| 7.3.1 | Documentation des constructeurs et destructeur | 16 |
| 7.3.1.1 | ContenanceTuile() | 16 |
| 7.4 | Référence de la classe Controller | 17 |
| 7.4.1 | Documentation des constructeurs et destructeur | 18 |
| 7.4.1.1 | Controller() | 18 |
| 7.4.2 | Documentation des fonctions membres | 18 |
| 7.4.2.1 | creerEspace() | 18 |
| 7.4.2.2 | estCompatible() | 18 |
| 7.4.2.3 | fusionVoisin() | 19 |
| 7.4.2.4 | getEspace() | 19 |
| 7.4.2.5 | getEspaceTuile() | 19 |
| 7.4.2.6 | placementMeeple() | 20 |
| 7.4.2.7 | placementTuile() | 20 |
| 7.4.2.8 | placementTuileAutorise() | 21 |

| | |
|---|----|
| 7.5 Référence de la classe Espace | 21 |
| 7.5.1 Description détaillée | 22 |
| 7.5.2 Documentation des constructeurs et destructeur | 22 |
| 7.5.2.1 Espace() | 22 |
| 7.5.3 Documentation des fonctions membres | 22 |
| 7.5.3.1 addContenance() | 22 |
| 7.5.3.2 addMeeple() | 23 |
| 7.5.3.3 getContenus() | 23 |
| 7.5.3.4 getMeeples() | 23 |
| 7.5.3.5 getNbrBouclier() | 23 |
| 7.5.3.6 getNbrContenanceTuile() | 24 |
| 7.5.3.7 getNbrMeeple() | 24 |
| 7.5.3.8 getType() | 24 |
| 7.5.3.9 isComplete() | 24 |
| 7.5.3.10 isFree() | 25 |
| 7.6 Référence de la classe Joueur | 25 |
| 7.6.1 Documentation des constructeurs et destructeur | 25 |
| 7.6.1.1 Joueur() | 25 |
| 7.6.2 Documentation des fonctions membres | 26 |
| 7.6.2.1 addMeeple() | 26 |
| 7.6.2.2 addName() | 26 |
| 7.6.2.3 addScore() | 26 |
| 7.6.2.4 setNbMeeple() | 27 |
| 7.7 Référence de la classe Meeple | 27 |
| 7.7.1 Description détaillée | 27 |
| 7.7.2 Documentation des constructeurs et destructeur | 27 |
| 7.7.2.1 Meeple() | 28 |
| 7.8 Référence de la classe ModeJeu | 28 |
| 7.8.1 Description détaillée | 28 |
| 7.8.2 Documentation des fonctions membres | 28 |
| 7.8.2.1 validationPlacementM() | 28 |
| 7.8.2.2 validationPlacementT() | 29 |
| 7.9 Référence de la classe Paysan | 29 |
| 7.9.1 Documentation des fonctions membres | 30 |
| 7.9.1.1 affichage() | 30 |
| 7.9.1.2 validationPlacementM() | 30 |
| 7.9.1.3 validationPlacementT() | 30 |
| 7.10 Référence de la classe Pioche | 31 |
| 7.10.1 Documentation des constructeurs et destructeur | 31 |
| 7.10.1.1 Pioche() | 31 |
| 7.10.2 Documentation des fonctions membres | 32 |
| 7.10.2.1 estVide() | 32 |

| | |
|---|----|
| 7.10.2.2 piocher() | 32 |
| 7.11 Référence de la classe Plateau | 32 |
| 7.11.1 Description détaillée | 33 |
| 7.11.2 Documentation des constructeurs et destructeur | 33 |
| 7.11.2.1 Plateau() | 33 |
| 7.11.3 Documentation des fonctions membres | 33 |
| 7.11.3.1 ajouterEspace() | 33 |
| 7.11.3.2 ajouterTuiles() | 34 |
| 7.11.3.3 existeTuile() | 34 |
| 7.11.3.4 supprimerEspace() | 34 |
| 7.12 Référence de la classe Riviere | 35 |
| 7.12.1 Documentation des fonctions membres | 35 |
| 7.12.1.1 affichage() | 35 |
| 7.12.1.2 validationPlacementM() | 35 |
| 7.12.1.3 validationPlacementT() | 36 |
| 7.13 Référence de la classe Standard | 36 |
| 7.13.1 Documentation des fonctions membres | 37 |
| 7.13.1.1 affichage() | 37 |
| 7.13.1.2 validationPlacementM() | 37 |
| 7.13.1.3 validationPlacementT() | 37 |
| 7.14 Référence de la classe Tuile | 38 |
| 7.14.1 Documentation des constructeurs et destructeur | 38 |
| 7.14.1.1 Tuile() | 38 |
| 7.14.2 Documentation des fonctions membres | 39 |
| 7.14.2.1 getContenance() | 39 |
| 7.14.2.2 getContenu() | 39 |
| 7.15 Référence de la classe TypeMeeple | 39 |
| 7.15.1 Description détaillée | 40 |
| 7.15.2 Documentation des constructeurs et destructeur | 40 |
| 7.15.2.1 TypeMeeple() | 40 |
| 7.16 Référence de la classe VueAccueil | 40 |
| 7.16.1 Description détaillée | 41 |
| 7.16.2 Documentation des constructeurs et destructeur | 41 |
| 7.16.2.1 VueAccueil() | 41 |
| 7.16.3 Documentation des fonctions membres | 41 |
| 7.16.3.1 getNbrJoueur() | 41 |
| 7.17 Référence de la classe VueContenuTuile | 42 |
| 7.17.1 Description détaillée | 42 |
| 7.17.2 Documentation des constructeurs et destructeur | 42 |
| 7.17.2.1 VueContenuTuile() [1/2] | 42 |
| 7.17.2.2 VueContenuTuile() [2/2] | 43 |
| 7.17.3 Documentation des fonctions membres | 43 |

| | |
|---|----|
| 7.17.3.1 setNomCouleurAvecM() | 43 |
| 7.18 Référence de la classe VueFormNom | 44 |
| 7.18.1 Documentation des constructeurs et destructeur | 44 |
| 7.18.1.1 VueFormNom() | 44 |
| 7.19 Référence de la classe VueJouerTour | 44 |
| 7.19.1 Documentation des constructeurs et destructeur | 45 |
| 7.19.1.1 VueJouerTour() | 45 |
| 7.20 Référence de la classe vueOuPlacerMeeple | 45 |
| 7.20.1 Documentation des constructeurs et destructeur | 45 |
| 7.20.1.1 vueOuPlacerMeeple() | 46 |
| 7.21 Référence de la classe VueOuRetirerMeeple | 46 |
| 7.21.1 Documentation des constructeurs et destructeur | 46 |
| 7.21.1.1 VueOuRetirerMeeple() | 46 |
| 7.22 Référence de la classe VuePartie | 47 |
| 7.22.1 Description détaillée | 48 |
| 7.22.2 Documentation des constructeurs et destructeur | 48 |
| 7.22.2.1 VuePartie() | 48 |
| 7.22.3 Documentation des fonctions membres | 48 |
| 7.22.3.1 placerMeeple() | 48 |
| 7.22.3.2 placerTuile() | 49 |
| 7.23 Référence de la classe vuePlacementMeeple | 49 |
| 7.23.1 Documentation des constructeurs et destructeur | 50 |
| 7.23.1.1 vuePlacementMeeple() | 50 |
| 7.23.2 Documentation des fonctions membres | 50 |
| 7.23.2.1 getNCol() | 50 |
| 7.23.2.2 getNligne() | 50 |
| 7.24 Référence de la classe VuePlateau | 51 |
| 7.24.1 Description détaillée | 51 |
| 7.24.2 Documentation des constructeurs et destructeur | 51 |
| 7.24.2.1 VuePlateau() | 51 |
| 7.25 Référence de la classe VueRetirerMeeple | 52 |
| 7.25.1 Documentation des constructeurs et destructeur | 52 |
| 7.25.1.1 VueRetirerMeeple() | 52 |
| 7.26 Référence de la classe VueScore1Joueur | 52 |
| 7.26.1 Description détaillée | 53 |
| 7.26.2 Documentation des constructeurs et destructeur | 53 |
| 7.26.2.1 VueScore1Joueur() | 53 |
| 7.27 Référence de la classe VueTuile | 53 |
| 7.27.1 Documentation des constructeurs et destructeur | 54 |
| 7.27.1.1 VueTuile() [1/2] | 54 |
| 7.27.1.2 VueTuile() [2/2] | 54 |
| 7.27.2 Documentation des fonctions membres | 55 |

| | |
|---|-----------|
| 7.27.2.1 addMeeple() | 55 |
| 7.27.2.2 getVueContenuT() | 55 |
| 7.27.2.3 setContenuTuile() | 55 |
| 8 Documentation des fichiers | 57 |
| 8.1 Référence du fichier controller.h | 57 |
| 8.2 controller.h | 57 |
| 8.3 Référence du fichier espace.h | 59 |
| 8.3.1 Description détaillée | 59 |
| 8.4 espace.h | 59 |
| 8.5 Référence du fichier joueur.h | 60 |
| 8.5.1 Description détaillée | 61 |
| 8.6 joueur.h | 61 |
| 8.7 meeple.h | 61 |
| 8.8 modeJeu.h | 62 |
| 8.9 Référence du fichier pioche.h | 67 |
| 8.10 pioche.h | 68 |
| 8.11 Référence du fichier plateau.h | 68 |
| 8.12 plateau.h | 68 |
| 8.13 Référence du fichier tuile.h | 69 |
| 8.13.1 Documentation du type de l'énumération | 70 |
| 8.13.1.1 TypesTuiles | 70 |
| 8.14 tuile.h | 70 |
| 8.15 vueAccueil.h | 72 |
| 8.16 vueContenuTuile.h | 72 |
| 8.17 Référence du fichier vueFormNom.h | 73 |
| 8.17.1 Description détaillée | 73 |
| 8.18 vueFormNom.h | 73 |
| 8.19 vueJouerTour.h | 74 |
| 8.20 vueOuPlacerMeeple.h | 74 |
| 8.21 vueOuRetirerMeeple.h | 75 |
| 8.22 vuePartie.h | 75 |
| 8.23 vuePlacementMeeple.h | 76 |
| 8.24 vuePlateau.h | 77 |
| 8.25 vueRetirerMeeple.h | 77 |
| 8.26 vueScore1Joueur.h | 78 |
| 8.27 vueTuile.h | 78 |
| Index | 81 |

Chapitre 1

Projet Carcassonne - Conception et développement : d'une application pour jouer au jeu Carcassonne

Projet de l'UV LO21 réalisé par DU Sylvain, BOUZAR Massil, OUEDRAOGO Taoufiq, THIBAUT Ambroise et TAVERNY Kelyan

Chapitre 2

Index des espaces de nommage

2.1 Liste des espaces de nommage

Liste de tous les espaces de nommage documentés avec une brève description:

| | |
|-------------------------------|----|
| std | 11 |
|-------------------------------|----|

Chapitre 3

Index hiérarchique

3.1 Hiérarchie des classes

Cette liste d'héritage est classée approximativement par ordre alphabétique :

| | |
|---------------------------------|----|
| ContenanceTuile | 16 |
| Controller | 17 |
| Espace | 21 |
| Joueur | 25 |
| Meeple | 27 |
| ModeJeu | 28 |
| Abbe | 13 |
| AubergesEtCathedrales | 14 |
| Paysan | 29 |
| Riviere | 35 |
| Standard | 36 |
| Pioche | 31 |
| Plateau | 32 |
| QDialog | |
| VueOuRetirerMeeple | 46 |
| VueRetirerMeeple | 52 |
| vueOuPlacerMeeple | 45 |
| vuePlacementMeeple | 49 |
| QMainWindow | |
| VueAccueil | 40 |
| VueFormNom | 44 |
| VuePartie | 47 |
| QWidget | |
| VueContenuTuile | 42 |
| VueJouerTour | 44 |
| VuePlateau | 51 |
| VueScore1Joueur | 52 |
| VueTuile | 53 |
| Tuile | 38 |
| TypeMeeple | 39 |

Chapitre 4

Index des classes

4.1 Liste des classes

Liste des classes, structures, unions et interfaces avec une brève description :

| | |
|---|----|
| Abbe | 13 |
| AubergesEtCathedrales | 14 |
| ContenanceTuile | 16 |
| Controller | 17 |
| Espace | |
| La classe Espace contient les différents éléments d'un Espace (ensemble de ContenuTuile de même type sur plusieurs Tuiles différentes tel qu'une ville ou champ) | 21 |
| Joueur | 25 |
| Meeple | |
| La classe fille | 27 |
| ModeJeu | |
| La classe ModeJeu implémentation du design pattern Template Method | 28 |
| Paysan | 29 |
| Pioche | 31 |
| Plateau | |
| La classe Plateau permet de gérer le plateau, avec les tuiles, les espaces | 32 |
| Riviere | 35 |
| Standard | 36 |
| Tuile | 38 |
| TypeMeeple | |
| Classe mère | 39 |
| VueAccueil | |
| La classe VueAccueil est de type QMainWindow | 40 |
| VueContenuTuile | |
| La classe VueContenuTuile est un QWidget affichant un type de tuile avec couleur associe et 2 premiere lettre du type de tuile | 42 |
| VueFormNom | 44 |
| VueJouerTour | 44 |
| vueOuPlacerMeeple | 45 |
| VueOuRetirerMeeple | 46 |
| VuePartie | |
| La classe VuePartie Est decomposer en plusieurs partie : un vertical layout contenant des VueScore1Joueur . 1 layout pour y mettre la Vuetuile a placer. 1 horizontal layout contenant un label et un QLCDNumber designant numero du tour actuel. 1 horizontal layout contenant les QPushButton (valider, fin Partie etc). 1 QWidget contenant un QLabel poru designer le nom du joueur devant jouer ainsi qu'un QLCDNumber designant le nombre de meeple lui restant | 47 |

| | |
|--|----|
| vuePlacementMeeple | 49 |
| VuePlateau | |
| La classe VuePlateau est un QWidget contenant un QWidgetTable | 51 |
| VueRetirerMeeple | 52 |
| VueScore1Joueur | |
| La classe VueScore1Joueur est un QWidget constitué d'un vertical layout contenant 2 QLabel pour désigner le nom du joueur et le score, 1 QLCDNumber pour afficher le score et 2 bouton dans un vertical layout pour changer le score | 52 |
| VueTuile | 53 |

Chapitre 5

Index des fichiers

5.1 Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

| | |
|--|----|
| controller.h | 57 |
| espace.h | |
| Details | 59 |
| joueur.h | |
| Details | 60 |
| meeple.h | ?? |
| modeJeu.h | ?? |
| pioche.h | 67 |
| plateau.h | 68 |
| tuile.h | 69 |
| vueAccueil.h | ?? |
| vueContenuTuile.h | ?? |
| vueFormNom.h | |
| Espace dédié a la création de label et zone de texte en fonction du nombre de joueur | 73 |
| vueJouerTour.h | ?? |
| vueOuPlacerMeeple.h | ?? |
| vueOuRetirerMeeple.h | ?? |
| vuePartie.h | ?? |
| vuePlacementMeeple.h | ?? |
| vuePlateau.h | ?? |
| vueRetirerMeeple.h | ?? |
| vueScore1Joueur.h | ?? |
| vueTuile.h | ?? |

Chapitre 6

Documentation des espaces de nommage

6.1 Référence de l'espace de nommage std

6.1.1 Description détaillée

Pour le placement des tuiles, seul l'extension rivière est différente Pour le placement des meeples: -[Standard](#): on peut placer partout sauf sur dans les champs (faite)

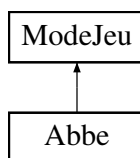
Pour le placement des meeples: -[Standard](#): on peut placer partout sauf sur dans les champs (faite) -Auberge et cathédrales : on ne peut pas placer sur une auberge ou une cathédrale (faite) -[Riviere](#) : On ne peut pas placer de meeples sur une tuile de type rivière (faite) -[Paysan](#) : couche (récupérer que à la fin de la partie) ou pas couche -[Abbe](#) :L'abbé ne peut être posé que sur une abbaye ou un jardin, -tandis que le meeples selon les règles habituelles peut être posé sur n'importe quelle section de tuile (mais pas sur le jardin).

Chapitre 7

Documentation des classes

7.1 Référence de la classe Abbe

Graphe d'héritage de Abbe:



Fonctions membres publiques

- bool [validationPlacementT](#) ([Tuile](#) newTuile, int x, int y, [Plateau](#) *plateau)
Vérifie le placement d'une tuile selon les règles de l'extensions.
- bool [validationPlacementM](#) (const [TypesTuiles](#) &tp, [Meeple](#) *m, [Espace](#) *e)
Vérifie le placement d'un meeple selon les règles de l'extension.
- void [affichage](#) ()

7.1.1 Documentation des fonctions membres

7.1.1.1 affichage()

```
void Abbe::affichage ( ) [inline], [virtual]
```

Implémente [ModeJeu](#).

7.1.1.2 validationPlacementM()

```
bool Abbe::validationPlacementM (  
    const TypesTuiles & tp,  
    Meeple * m,  
    Espace * e ) [inline], [virtual]
```

Vérifie le placement d'un meeple selon les règles de l'extension.

Paramètres

| | |
|-----------|--|
| <i>tp</i> | |
| <i>m</i> | |
| <i>e</i> | |

Renvoie

Implémente [ModeJeu](#).

7.1.1.3 validationPlacementT()

```
bool Abbe::validationPlacementT (
    Tuile newTuile,
    int x,
    int y,
    Plateau * plateau ) [inline], [virtual]
```

Vérifie le placement d'une tuile selon les règles de l'extensions.

Paramètres

| | |
|-----------------|--|
| <i>newTuile</i> | |
| <i>x</i> | |
| <i>y</i> | |
| <i>plateau</i> | |

Renvoie

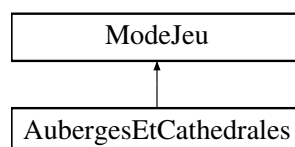
Implémente [ModeJeu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

— modeJeu.h

7.2 Référence de la classe AubergesEtCathedrales

Graphique d'héritage de AubergesEtCathedrales:



Fonctions membres publiques

- bool [validationPlacementT](#) ([Tuile](#) newTuile, int x, int y, [Plateau](#) *plateau)
Vérifie le placement d'une tuile selon les règles de l'extensions.
- bool [validationPlacementM](#) (const [TypesTuiles](#) &tp, [Meeple](#) *m, [Espace](#) *e)
Vérifie le placement d'un meeple selon les règles de l'extension.
- void [affichage](#) ()

7.2.1 Documentation des fonctions membres

7.2.1.1 affichage()

```
void AubergesEtCathedrales::affichage ( ) [inline], [virtual]
```

Implémente [ModeJeu](#).

7.2.1.2 validationPlacementM()

```
bool AubergesEtCathedrales::validationPlacementM (
    const TypesTuiles & tp,
    Meeple * m,
    Espace * e ) [inline], [virtual]
```

Vérifie le placement d'un meeple selon les règles de l'extension.

Paramètres

| | |
|-----------|--|
| <i>tp</i> | |
| <i>m</i> | |
| <i>e</i> | |

Renvoie

Implémente [ModeJeu](#).

7.2.1.3 validationPlacementT()

```
bool AubergesEtCathedrales::validationPlacementT (
    Tuile newTuile,
    int x,
    int y,
    Plateau * plateau ) [inline], [virtual]
```

Vérifie le placement d'une tuile selon les règles de l'extensions.

Paramètres

| | |
|-----------------|--|
| <i>newTuile</i> | |
| <i>x</i> | |
| <i>y</i> | |
| <i>plateau</i> | |

Renvoi

Implémente [ModeJeu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

— modeJeu.h

7.3 Référence de la classe ContenanceTuile

Fonctions membres publiques

- [ContenanceTuile](#) (const [TypesTuiles](#) t, const int n)
il prend en paramètre nbT.
- const [TypesTuiles](#) & **getType** () const
- const int **getNumPlacement** () const
- const bool **getBouclier** () const
- void **setBouclier** ()
Si le ContenuTuile est une ville, permet de lui attribuer un bouclier.
- void **setType** (const [TypesTuiles](#) t)
- bool **operator==** (const [ContenanceTuile](#) &c) const
- [ContenanceTuile](#) (const [ContenanceTuile](#) &c)
- [ContenanceTuile](#) & **operator=** (const [ContenanceTuile](#) &c)

Amis

- class **Pioche**
- class **Tuile**

7.3.1 Documentation des constructeurs et destructeur

7.3.1.1 ContenanceTuile()

```
ContenanceTuile::ContenanceTuile (
    const TypesTuiles t,
    const int n ) [inline]
```

il prend en paramètre *nbT*.

Construteur

Paramètres

| | |
|----------|--------------------------------|
| <i>t</i> | est de type TypesTuiles const. |
| <i>n</i> | est un entier const. |

La documentation de cette classe a été générée à partir du fichier suivant :

— [tuile.h](#)

7.4 Référence de la classe Controller

Fonctions membres publiques

- **Controller** (int nj)
Le controller contient les principaux algorithmes permettant de gerer une partie.
- **Controller** (int nj, vector< string > listeNomJoueur, vector< int > listeNumExtensions)
- **Controller** (const **Controller** &)=delete
- **Controller** & **operator=** (const **Controller** &)=delete
- bool **getPaysansActive** () const
- bool **getAbbeActive** () const
- bool **getRiviereActive** () const
- bool **getAubergeActive** () const
- **Espace** * **getEspace** (const **ContenanceTuile** *c)
Permet de récupérer l'espace d'une contenue de tuile.
- **Espace** * **getEspaceTuile** (const **ContenanceTuile** &c, const vector< **Espace** * > e)
Permet de récupérer l'espace d'une contenue de tuile.
- void **creerEspace** (const **Tuile** *T)
Permet de regrouper les différents espaces d'une même tuile en prenant tous les cas possibles selon les règles et extensions.
- void **fusionVoisin** (const **Tuile** *tuile)
Permet de fusionner les différents espaces de deux tuiles différentes et de regrouper les mêmes espaces.
- void **placementTuile** (**Tuile** *newTuile, int x, int y)
Permet de placer la tuile sur le plateau en attribuant tous ses voisins s'il en a et ses coordonnées sur le plateau.
- void **placementMeeple** (**Joueur** *j, **Meeple** *m, **TypesTuiles** tm, int i, int x, int y)
Permet de placer un meeples sur une contenu de tuile en initialisant son type, le contenu où il est, le joueur à qui appartient le meeples, ses coordonnées, et l'ajout dans son espace, de plus on retire le meeples au joueur.
- bool **placementTuileAutorise** (**Tuile** newTuile)
Vérifie si la tuile donnée peut être autorisée n'importe où sur le plateau, à côté d'autres tuiles et selon les règles.
- bool **estCompatible** (**Tuile** newTuile, int x, int y)
Vérifie si la tuile donnée peut être placée à la position x, y sur le plateau.
- void **nextTour** ()
Permet de passer au joueur suivant, d'incrémenter le nombre de tour et de terminer la partie si elle est finie.
- void **AffichageJoueurs** ()
la méthode AffichageJoueurs permet d'afficher tous les joueurs de la partie.
- vector< **Joueur** * > **getJoueurs** ()
- bool **getFin** () const
- int **getNbJoueur** () const
- int **getTour** () const
- **Pioche** * **getPioche** () const
- int **getNumJoueurActu** () const
- **Plateau** * **getPlateau** () const
- vector< int > **getExtensions** () const
- **ModeJeu** * **getModeJeu** (int i) const
- bool **placementTuileAutorise** (**Tuile** newTuile, **Plateau** *plateau)
- bool **estCompatible** (**Tuile** newTuile, int x, int y, **Plateau** *plateau)
- bool **validationPlacementRiviere** (**Tuile** newTuile, int x, int y, **Plateau** *plateau)
- void **placementMeeple** (**Joueur** *j, **Meeple** *m, **TypeMeeple** tm, int i, int x, int y, **Plateau** *plateau)
- void **placementTuile** (**Tuile** *newTuile, int x, int y, **Plateau** *plateau)

7.4.1 Documentation des constructeurs et destructeur

7.4.1.1 Controller()

```
Controller::Controller (
    int nj )
```

Le controller contient les principaux algorithmes permettant de gerer une partie.

Constructeur

Paramètres

| | |
|---------------------------|--------------------------------|
| <i>nj</i> | un entier du nombre de joueur. |
| <i>listeNomJoueur</i> | un vector de string. |
| <i>listeNumExtensions</i> | un vector d'entier. |

7.4.2 Documentation des fonctions membres

7.4.2.1 creerEspace()

```
void Controller::creerEspace (
    const Tuile * T )
```

Permet de regrouper les différents espaces d'une même tuile en prenant tous les cas possibles selon les règles et extensions.

Paramètres

| | |
|----------|--|
| <i>T</i> | |
|----------|--|

7.4.2.2 estCompatible()

```
bool Controller::estCompatible (
    Tuile newTuile,
    int x,
    int y )
```

Vérifie si la tuile donnée peut être placée à la position x, y sur le plateau.

Paramètres

| | |
|-----------------|---|
| <i>newTuile</i> | est de type Tuile . |
| <i>newTuile</i> | est de type Tuile . |
| <i>x</i> | pour la position horizontale de la tuile. |
| <i>y</i> | pour la position verticale de la tuile. |

7.4.2.3 fusionVoisin()

```
void Controller::fusionVoisin (
    const Tuile * tuile )
```

Permet de fusionner les différents espaces de deux tuiles différentes et de regrouper les mêmes espaces.

Paramètres

| | |
|--------------|--|
| <i>tuile</i> | |
|--------------|--|

7.4.2.4 getEspace()

```
Espace * Controller::getEspace (
    const ContenanceTuile * c )
```

Permet de récupérer l'espace d'une contenue de tuile.

Paramètres

| | |
|----------|--|
| <i>c</i> | |
|----------|--|

Renvoie

7.4.2.5 getEspaceTuile()

```
Espace * Controller::getEspaceTuile (
    const ContenanceTuile & c,
    const vector< Espace * > e )
```

Permet de récupérer l'espace d'une contenue de tuile.

Paramètres

| | |
|----------|--|
| <i>c</i> | |
| <i>e</i> | |

Renvoie

7.4.2.6 placementMeeple()

```
void Controller::placementMeeple (
    Joueur * j,
    Meeple * m,
    TypesTuiles tm,
    int i,
    int x,
    int y )
```

Permet de placer un meeples sur une contenu de tuile en initialisant son type, le contenu où il est, le joueur à qui appartient le meeples, ses coordonnées, et l'ajout dans son espace, de plus on retire le meeples au joueur.

Paramètres

| | |
|-----------|---|
| <i>j</i> | est un pointeur de Joueur . |
| <i>m</i> | est un pointeur de Meeple . |
| <i>tm</i> | est un TypeMeeple . |
| <i>i</i> | est un entier. |
| <i>x</i> | pour la position. |
| <i>y</i> | pour la position. |

7.4.2.7 placementTuile()

```
void Controller::placementTuile (
    Tuile * newTuile,
    int x,
    int y )
```

Permet de placer la tuile sur le plateau en attribuant tous ses voisins s'il en a et ses coordonnées sur le plateau.

Paramètres

| | |
|-----------------|--|
| <i>newTuile</i> | est un pointeur de Tuile . |
| <i>x</i> | est la position horizontal. |
| <i>y</i> | est la position vertical. |

7.4.2.8 placementTuileAutorise()

```
bool Controller::placementTuileAutorise (
    Tuile newTuile )
```

Vérifie si la tuile donnée peut être autorisée n'importe où sur le plateau, à côté d'autres tuiles et selon les règles.

Paramètres

| | |
|-----------------|-------------------------------------|
| <i>newTuile</i> | est de type Tuile . |
|-----------------|-------------------------------------|

La documentation de cette classe a été générée à partir du fichier suivant :

— [controller.h](#)

7.5 Référence de la classe Espace

La classe [Espace](#) contient les différents éléments d'un [Espace](#) (ensemble de ContenuTuile de même type sur plusieurs Tuiles différentes tel qu'une ville ou champ)

```
#include <espace.h>
```

Fonctions membres publiques

- [Espace](#) (const [TypesTuiles](#) &Tt)
il prend en paramètre Tt.
- const int [getNbrMeeple](#) () const
récupérer le nombre de meeple
- const int [getNbrBouclier](#) () const
récupérer le nombre de bouclier
- const [TypesTuiles](#) & [getType](#) () const
récupérer le type
- const vector< const [ContenanceTuile](#) * > [getContenus](#) () const
récupérer l'extention
- const [ContenanceTuile](#) * [getContenus](#) (int i) const
- const vector< const [Meeple](#) * > [getMeeples](#) () const
getMeeples
- const [Meeple](#) * [getMeeples](#) (int i) const
- const size_t [getNbrContenanceTuile](#) () const
récupérer le nombre de contenance de la tuile.
- void [addBouclier](#) ()
ajouter un bouclier
- void [addContenance](#) (const [ContenanceTuile](#) *C)
ajouter une contenance
- void [addMeeple](#) (const [Meeple](#) *M)
ajouter un meeple.
- [Espace](#) * [fusionEspace](#) ([Espace](#) *e)
- bool [isComplete](#) ()
vérifie si c'est complet.
- const bool [isFree](#) () const
vérifie si la place est libre.
- void [calculScore](#) ()
calcul le score.

Amis

— class **Controller**

7.5.1 Description détaillée

La classe [Espace](#) contient les différents éléments d'un [Espace](#) (ensemble de ContenuTuile de même type sur plusieurs Tuiles différentes tel qu'une ville ou champ)

7.5.2 Documentation des constructeurs et destructeur

7.5.2.1 Espace()

```
Espace::Espace (
    const TypesTuiles & Tt ) [inline]
```

il prend en paramètre *Tt*.

Constructeur

Paramètres

| | |
|-----------|---|
| <i>Tt</i> | est de type TypesTuiles . |
|-----------|---|

7.5.3 Documentation des fonctions membres

7.5.3.1 addContenance()

```
void Espace::addContenance (
    const ContenanceTuile * C )
```

ajouter une contenance

Paramètres

| | |
|----------|---|
| <i>C</i> | est de type ContenanceTuile . |
|----------|---|

7.5.3.2 addMeeple()

```
void Espace::addMeeple (
    const Meeple * M )
```

ajouter un meeple.

Paramètres

| | |
|----------|--------------------|
| <i>M</i> | est de type meeple |
|----------|--------------------|

7.5.3.3 getContenus()

```
const vector< const ContenanceTuile * > Espace::getContenus ( ) const [inline]
```

recupérer l'extention

Renvoie

le contenus

7.5.3.4 getMeeples()

```
const vector< const Meeple * > Espace::getMeeples ( ) const [inline]
```

getMeeples

Renvoie

7.5.3.5 getNbrBouclier()

```
const int Espace::getNbrBouclier ( ) const [inline]
```

recupérer le nombre de bouclier

Renvoie

nombre de bouclier

7.5.3.6 getNbrContenanceTuile()

```
const size_t Espace::getNbrContenanceTuile ( ) const [inline]
```

récupérer le nombre de contenance de la tuile.

Renvoie

la taille de contenus

7.5.3.7 getNbrMeeple()

```
const int Espace::getNbrMeeple ( ) const [inline]
```

récupérer le nombre de meeple

Renvoie

nombre de [Meeple](#)

7.5.3.8 getType()

```
const TypesTuiles & Espace::getType ( ) const [inline]
```

récupérer le type

Renvoie

le type

7.5.3.9 isComplete()

```
bool Espace::isComplete ( )
```

vérifie si c'est complet.

Renvoie

vrai ou faux.

7.5.3.10 isFree()

```
const bool Espace::isFree ( ) const [inline]
```

vérifie si la place est libre.

Renvoie

vrai ou faux

La documentation de cette classe a été générée à partir du fichier suivant :

— [espace.h](#)

7.6 Référence de la classe Joueur

Fonctions membres publiques

- [Joueur](#) (int uid, int nb, string str)
il prend en paramètre uid, nb et str
- **Joueur** (int uid)
- const int **getId** () const
- const int **getScore** () const
- const size_t **getNbrMeeplesUsed** () const
- const int **getNbrMeeples** () const
- const string **getName** () const
- const vector< const [Meeple](#) * > **getMeeples** () const
- void [addScore](#) (int pts)
ajouter des points au score.
- void [addMeeple](#) (const [Meeple](#) &m)
Attribue un meeple lorsque le joueur en question décide de placer un meeple sur une tuile, en tenant compte du nombre maximum de meeple défini par les extensions.
- void [addName](#) (const string &str)
Permet de nommer un joueur de la partie.
- void **removeMeeple** ()
déplacer un meeple
- void [setNbMeeple](#) (int i)
modifier le nombre de meeple

7.6.1 Documentation des constructeurs et destructeur

7.6.1.1 Joueur()

```
Joueur::Joueur (
    int uid,
    int nb,
    string str ) [inline]
```

il prend en paramètre *uid*, *nb* et *str*

Construteur

Paramètres

| | |
|------------|--------------------|
| <i>uid</i> | est un entier |
| <i>nb</i> | est un entier |
| <i>str</i> | est de type string |

7.6.2 Documentation des fonctions membres

7.6.2.1 addMeeple()

```
void Joueur::addMeeple (
    const Meeple & m )
```

Attribue un meeple lorsque le joueur en question décide de placer un meeple sur une tuile, en tenant compte du nombre maximum de meeple défini par les extensions.

Paramètres

| | |
|----------|--|
| <i>m</i> | |
|----------|--|

7.6.2.2 addName()

```
void Joueur::addName (
    const string & str )
```

Permet de nommer un joueur de la partie.

Paramètres

| | |
|------------|--|
| <i>str</i> | |
|------------|--|

7.6.2.3 addScore()

```
void Joueur::addScore (
    int pts ) [inline]
```

ajouter des points au score.

Paramètres

| | |
|------------|--|
| <i>pts</i> | |
|------------|--|

7.6.2.4 setNbMeeple()

```
void Joueur::setNbMeeple (
    int i ) [inline]
```

modifier le nombre de meeple

Paramètres

| | |
|----------|--|
| <i>i</i> | |
|----------|--|

La documentation de cette classe a été générée à partir du fichier suivant :

— [joueur.h](#)

7.7 Référence de la classe Meeple

la classe fille

```
#include <meeple.h>
```

Fonctions membres publiques

- [Meeple](#) ()
Meeple.
- **Meeple** (const [Meeple](#) &)=delete
- [Meeple](#) & **operator=** (const [Meeple](#) &)=delete
- bool **isUsed** () const
- const [TypeMeeple](#) & **getType** () const
- const int **getIdJoueur** () const
- const [ContenanceTuile](#) * **getContenanceTuile** () const
- void **setType** (const [TypeMeeple](#) &ty)
- void **setType** (const NomMeeple &name)
- void **setIdJoueur** (int i)
- void **setContenance** ([ContenanceTuile](#) &ct)

7.7.1 Description détaillée

la classe fille

7.7.2 Documentation des constructeurs et destructeur

7.7.2.1 Meeple()

```
Meeple::Meeple ( ) [inline]
```

[Meeple](#).

Constructeur

La documentation de cette classe a été générée à partir du fichier suivant :

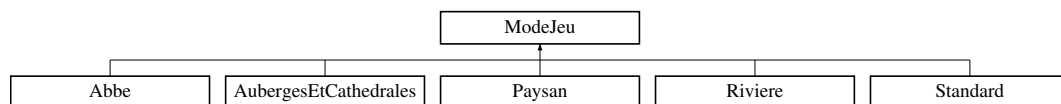
— meeple.h

7.8 Référence de la classe ModeJeu

La classe [ModeJeu](#) implémentation du design pattern Template Method.

```
#include <modeJeu.h>
```

Graphe d'héritage de ModeJeu:



Fonctions membres publiques

- virtual bool [validationPlacementT](#) ([Tuile](#) newTuile, int x, int y, [Plateau](#) *plateau)=0
Vérifie le placement d'une tuile selon les règles de l'extensions.
- virtual bool [validationPlacementM](#) (const [TypesTuiles](#) &tp, [Meeple](#) *m, [Espace](#) *e)=0
Vérifie le placement d'un meeple selon les règles de l'extension.
- virtual void [affichage](#) ()=0

7.8.1 Description détaillée

La classe [ModeJeu](#) implémentation du design pattern Template Method.

7.8.2 Documentation des fonctions membres

7.8.2.1 validationPlacementM()

```
virtual bool ModeJeu::validationPlacementM (
    const TypesTuiles & tp,
    Meeple * m,
    Espace * e ) [pure virtual]
```

Vérifie le placement d'un meeple selon les règles de l'extension.

Paramètres

| | |
|-----------|--|
| <i>tp</i> | |
| <i>m</i> | |
| <i>e</i> | |

Renvoie

Implémenté dans [Standard](#), [Riviere](#), [AubergesEtCathedrales](#), [Paysan](#), et [Abbe](#).

7.8.2.2 validationPlacementT()

```
virtual bool ModeJeu::validationPlacementT (
    Tuile newTuile,
    int x,
    int y,
    Plateau * plateau ) [pure virtual]
```

Vérifie le placement d'une tuile selon les règles de l'extensions.

Paramètres

| | |
|-----------------|--|
| <i>newTuile</i> | |
| <i>x</i> | |
| <i>y</i> | |
| <i>plateau</i> | |

Renvoie

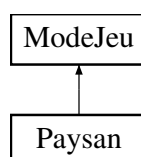
Implémenté dans [Standard](#), [Riviere](#), [AubergesEtCathedrales](#), [Paysan](#), et [Abbe](#).

La documentation de cette classe a été générée à partir du fichier suivant :

— modeJeu.h

7.9 Référence de la classe Paysan

Graphe d'héritage de Paysan:



Fonctions membres publiques

- bool [validationPlacementT](#) ([Tuile](#) newTuile, int x, int y, [Plateau](#) *plateau)
Vérifie le placement d'une tuile selon les règles de l'extensions.
- bool [validationPlacementM](#) (const [TypesTuiles](#) &tp, [Meeple](#) *m, [Espace](#) *e)
Vérifie le placement d'un meeple selon les règles de l'extension.
- void [affichage](#) ()

7.9.1 Documentation des fonctions membres

7.9.1.1 affichage()

```
void Paysan::affichage ( ) [inline], [virtual]
```

Implémente [ModeJeu](#).

7.9.1.2 validationPlacementM()

```
bool Paysan::validationPlacementM (
    const TypesTuiles & tp,
    Meeple * m,
    Espace * e ) [inline], [virtual]
```

Vérifie le placement d'un meeple selon les règles de l'extension.

Paramètres

| | |
|-----------|--|
| <i>tp</i> | |
| <i>m</i> | |
| <i>e</i> | |

Renvoie

Implémente [ModeJeu](#).

7.9.1.3 validationPlacementT()

```
bool Paysan::validationPlacementT (
    Tuile newTuile,
    int x,
    int y,
    Plateau * plateau ) [inline], [virtual]
```

Vérifie le placement d'une tuile selon les règles de l'extensions.

Paramètres

| | |
|-----------------|--|
| <i>newTuile</i> | |
| <i>x</i> | |
| <i>y</i> | |
| <i>plateau</i> | |

Renvoie

Implémente [ModeJeu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

— `modeJeu.h`

7.10 Référence de la classe Pioche

Fonctions membres publiques

- [Pioche](#) (vector< int > mode)
il prend en paramètre mode.
- const [Tuile](#) & **getTuile** (int i) const
- size_t **getNbTuiles** () const
- [Tuile](#) * **piocher** (int nbTour, bool riviereActive)
c'est la fonction qui relie l'action de piocher
- bool **estVide** () const
vérifie si l'emplacement est vide
- **Pioche** (const [Pioche](#) &)=delete
- [Pioche](#) & **operator=** (const [Pioche](#) &)=delete
- bool **getBouclier** ([ContenanceTuile](#) &c)

7.10.1 Documentation des constructeurs et destructeur

7.10.1.1 Pioche()

```
Pioche::Pioche (
    vector< int > mode )
```

il prend en paramètre *mode*.

Constructeur

Paramètres

| | |
|-------------|-------------------------|
| <i>mode</i> | est un vector d'entier. |
|-------------|-------------------------|

7.10.2 Documentation des fonctions membres

7.10.2.1 estVide()

```
bool Pioche::estVide ( ) const [inline]
```

vérifie si l'emplacement est vide

Renvoie

7.10.2.2 piocher()

```
Tuile * Pioche::piocher (
    int nbTour,
    bool riviereActive )
```

c'est la fonction qui relie l'action de piocher

Paramètres

| | |
|----------------------|--|
| <i>nbTour</i> | |
| <i>riviereActive</i> | |

Renvoie

La documentation de cette classe a été générée à partir du fichier suivant :

— [pioche.h](#)

7.11 Référence de la classe Plateau

La classe [Plateau](#) permet de gérer le plateau, avec les tuiles, les espaces.

```
#include <plateau.h>
```


Fonctions membres publiques

- `Plateau` (int nbT)
il prend en paramètre nbT.
- int `getNbrTuiles` () const
- int `getNbrEspaces` () const
- void `setNbrTuiles` (const int &nbT)
- void `ajouterTuiles` (Tuile *tuile)
Permet d'ajouter les tuiles dans le plateau .
- void `ajouterEspace` (Espace *espace)
Permet d'ajouter un espace dans le plateau.
- void `supprimerEspace` (Espace *espace)
Permet de supprimer un espace sur le plateau.
- std::vector< Tuile * > `getTuiles` () const
- std::vector< Espace * > `getEspaces` () const
- Tuile * `existeTuile` (int x, int y)
Vérifie si il existe une tuile a cette emplacement.

7.11.1 Description détaillée

La classe `Plateau` permet de gérer le plateau, avec les tuiles, les espaces.

7.11.2 Documentation des constructeurs et destructeur

7.11.2.1 Plateau()

```
Plateau::Plateau (
    int nbT ) [inline]
```

il prend en paramètre *nbT*.

Constructeur

Paramètres

| | |
|------------|----------------|
| <i>nbT</i> | est un entier. |
|------------|----------------|

7.11.3 Documentation des fonctions membres

7.11.3.1 ajouterEspace()

```
void Plateau::ajouterEspace (
    Espace * espace )
```

Permet d'ajouter un espace dans le plateau.

Paramètres

| | |
|---------------|--|
| <i>espace</i> | |
|---------------|--|

7.11.3.2 ajouterTuiles()

```
void Plateau::ajouterTuiles (
    Tuile * tuile )
```

Permet d'ajouter les tuiles dans le plateau .

Paramètres

| | |
|--------------|--|
| <i>tuile</i> | |
|--------------|--|

7.11.3.3 existeTuile()

```
Tuile * Plateau::existeTuile (
    int x,
    int y ) [inline]
```

Vérifie si il existe une tuile a cette emplacement.

Paramètres

| | |
|----------|--|
| <i>x</i> | |
| <i>y</i> | |

Renvoie

7.11.3.4 supprimerEspace()

```
void Plateau::supprimerEspace (
    Espace * espace )
```

Permet de supprimer un espace sur le plateau.

Paramètres

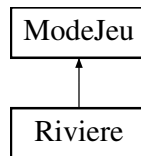
| | |
|---------------|--|
| <i>espace</i> | |
|---------------|--|

La documentation de cette classe a été générée à partir du fichier suivant :

— [plateau.h](#)

7.12 Référence de la classe Riviere

Graphe d'héritage de Riviere:



Fonctions membres publiques

- bool [validationPlacementT](#) ([Tuile](#) newTuile, int x, int y, [Plateau](#) *plateau)
Vérifie le placement d'une tuile selon les règles de l'extensions.
- bool [validationPlacementM](#) (const [TypesTuiles](#) &tp, [Meeple](#) *m, [Espace](#) *e)
Vérifie le placement d'un meeple selon les règles de l'extension.
- void [affichage](#) ()

7.12.1 Documentation des fonctions membres

7.12.1.1 affichage()

```
void Riviere::affichage ( ) [inline], [virtual]
```

Implémente [ModeJeu](#).

7.12.1.2 validationPlacementM()

```
bool Riviere::validationPlacementM (
    const TypesTuiles & tp,
    Meeple * m,
    Espace * e ) [inline], [virtual]
```

Vérifie le placement d'un meeple selon les règles de l'extension.

Paramètres

| | |
|-----------|--|
| <i>tp</i> | |
| <i>m</i> | |
| <i>e</i> | |

Renvoie

Implémente [ModeJeu](#).

7.12.1.3 validationPlacementT()

```
bool Riviere::validationPlacementT (
    Tuile newTuile,
    int x,
    int y,
    Plateau * plateau ) [inline], [virtual]
```

Vérifie le placement d'une tuile selon les règles de l'extensions.

Paramètres

| | |
|-----------------|--|
| <i>newTuile</i> | |
| <i>x</i> | |
| <i>y</i> | |
| <i>plateau</i> | |

Renvoie

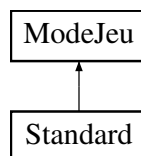
Implémente [ModeJeu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

— modeJeu.h

7.13 Référence de la classe Standard

Graphe d'héritage de Standard:



Fonctions membres publiques

- bool [validationPlacementT](#) ([Tuile](#) newTuile, int x, int y, [Plateau](#) *plateau)
Vérifie le placement d'une tuile selon les règles de l'extensions.
- bool [validationPlacementM](#) (const [TypesTuiles](#) &tp, [Meeple](#) *m, [Espace](#) *e)
Vérifie le placement d'un meeples selon les règles de l'extension.
- void [affichage](#) ()

7.13.1 Documentation des fonctions membres

7.13.1.1 affichage()

```
void Standard::affichage ( ) [inline], [virtual]
```

Implémente [ModeJeu](#).

7.13.1.2 validationPlacementM()

```
bool Standard::validationPlacementM (
    const TypesTuiles & tp,
    Meeples * m,
    Espace * e ) [inline], [virtual]
```

Vérifie le placement d'un meeples selon les règles de l'extension.

Paramètres

| | |
|-----------|--|
| <i>tp</i> | |
| <i>m</i> | |
| <i>e</i> | |

Renvoie

Implémente [ModeJeu](#).

7.13.1.3 validationPlacementT()

```
bool Standard::validationPlacementT (
    Tuile newTuile,
    int x,
    int y,
    Plateau * plateau ) [inline], [virtual]
```

Vérifie le placement d'une tuile selon les règles de l'extensions.

Paramètres

| | |
|-----------------|--|
| <i>newTuile</i> | |
| <i>x</i> | |
| <i>y</i> | |
| <i>plateau</i> | |

Renvoie

Implémente [ModeJeu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

— modeJeu.h

7.14 Référence de la classe Tuile

Fonctions membres publiques

- [Tuile](#) (const vector< [ContenanceTuile](#) > &c)
il prend en paramètre c.
- const size_t **getSize** () const
- const int **getX** () const
- const int **getY** () const
- const [Tuile](#) * **getVoisinHaut** () const
- const [Tuile](#) * **getVoisinBas** () const
- const [Tuile](#) * **getVoisinGauche** () const
- const [Tuile](#) * **getVoisinDroite** () const
- const vector< [ContenanceTuile](#) > **getContenance** () const
- const [ContenanceTuile](#) & **getContenance** (int i) const
Renvoie le ContenuTuile situé au ième emplacement.
- const [ContenanceTuile](#) * **getContenancePointeur** (int i) const
- [ContenanceTuile](#) * **getContenancePointeur** (int i)
- const [TypesTuiles](#) & **getContenu** (int i) const
Renvoie le type du ContenuTuile situé au ième emplacement.
- void **changerOrientation** ()
- [Tuile](#) (const [Tuile](#) &T)
- [Tuile](#) & **operator=** (const [Tuile](#) &T)
- bool **operator==** (const [Tuile](#) &T) const
- void **setContenu** (int i, [TypesTuiles](#) c)
- void **ReplaceParChamps** ()
- void **setVoisinHaut** ([Tuile](#) *t)
- void **setVoisinBas** ([Tuile](#) *t)
- void **setVoisinGauche** ([Tuile](#) *t)
- void **setVoisinDroite** ([Tuile](#) *t)
- void **setPosX** (int x)
- void **setPosY** (int y)
- void **setX** (int x)
- void **setY** (int y)

7.14.1 Documentation des constructeurs et destructeur

7.14.1.1 Tuile()

```
Tuile::Tuile (
    const vector< ContenanceTuile > & c )
```

il prend en paramètre c.

Construteur

Paramètres

| | |
|----------|-----------------------------------|
| <i>c</i> | est un vector de contenanceTuile. |
|----------|-----------------------------------|

7.14.2 Documentation des fonctions membres

7.14.2.1 getContenance()

```
const ContenanceTuile & Tuile::getContenance (
    int i ) const [inline]
```

Renvoie le ContenuTuile situé au ième emplacement.

Paramètres

| | |
|----------|--|
| <i>i</i> | |
|----------|--|

Renvoie

7.14.2.2 getContenu()

```
const TypesTuiles & Tuile::getContenu (
    int i ) const [inline]
```

Renvoie le type du ContenuTuile situé au ième emplacement.

Paramètres

| | |
|----------|--|
| <i>i</i> | |
|----------|--|

Renvoie

La documentation de cette classe a été générée à partir du fichier suivant :
— [tuile.h](#)

7.15 Référence de la classe TypeMeeple

Classe mère.

```
#include <meeple.h>
```

Fonctions membres publiques

- [TypeMeeple](#) (const NomMeeple &name=rien)
il prend en paramètre name et est initialisé à rien.
- const NomMeeple & **getNom** () const
- void **setNom** (const NomMeeple &name)

7.15.1 Description détaillée

Classe mère.

7.15.2 Documentation des constructeurs et destructeur

7.15.2.1 TypeMeeple()

```
TypeMeeple::TypeMeeple (
    const NomMeeple & name = rien ) [inline]
```

il prend en paramètre *name* et est initialisé à rien.

Construteur

Paramètres

| | |
|-------------|------------------------|
| <i>name</i> | est de type NomMeeple& |
|-------------|------------------------|

La documentation de cette classe a été générée à partir du fichier suivant :

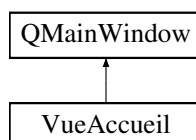
- meeples.h

7.16 Référence de la classe VueAccueil

La classe [VueAccueil](#) est de type QMainWindow.

```
#include <vueAccueil.h>
```

Graphe d'héritage de VueAccueil:



Fonctions membres publiques

- [VueAccueil](#) (QWidget *parent=nullptr)
créer l'espace QSpinBox pour entrer le nombre de joueur, et, grâce a des checkBox, demande les extensions choisi pour le jeu.
- void **setNbrJoueur** ()
Cette methode permet de récupérer le nombre entré dans le QSpinBox dans [VueAccueil](#).
- int **getNbrJoueur** () const
Cette fonction permet de transformer la valeur récupée par setNbJoueur et la tranformer en int()

7.16.1 Description détaillée

La classe [VueAccueil](#) est de type QMainWindow.

7.16.2 Documentation des constructeurs et destructeur

7.16.2.1 VueAccueil()

```
VueAccueil::VueAccueil (
    QWidget * parent = nullptr ) [explicit]
```

créer l'espace QSpinBox pour entrer le nombre de joueur, et, grâce a des checkBox, demande les extensions choisi pour le jeu.

Construteur

Paramètres

| | |
|---------------|--|
| <i>parent</i> | |
|---------------|--|

7.16.3 Documentation des fonctions membres

7.16.3.1 getNbrJoueur()

```
int VueAccueil::getNbrJoueur ( ) const [inline]
```

Cette fonction permet de transformer la valeur récupée par setNbJoueur et la tranformer en int()

Renvoie

La documentation de cette classe a été générée à partir du fichier suivant :

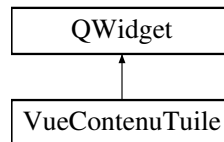
- vueAccueil.h

7.17 Référence de la classe VueContenuTuile

La classe `VueContenuTuile` est un `QWidget` affichant un type de tuile avec couleur associée et 2 première lettre du type de tuile.

```
#include <vueContenuTuile.h>
```

Graphe d'héritage de `VueContenuTuile`:



Fonctions membres publiques

- `VueContenuTuile` (`TypesTuiles` type, bool bouc, `QWidget *parent=nullptr`)
Premier constructeur sans placement de meeple. Appel de `setNomCouleurSansM`.
- `VueContenuTuile` (`TypesTuiles` type, bool bouc, int id, bool meeple, `QWidget *parent=nullptr`)
Second constructeur pour placement `Meeple`, appel de `setNomCouleurAvecM`.
- void `setNomCouleurSansM` ()
En fonction de type, définit la couleur du widget et y ajoute le bon texte.
- void `setNomCouleurAvecM` (const int &id)
Même fonctionnement que `setNomCouleurSansM`. rajoute le texte à l'affichage classique des 2 premières lettres pour indiquer la présence de meeple.
- `TypesTuiles` `getTypeTuile` () const
- bool `getBouclier` () const
- bool `getMeeple` () const
- `VueContenuTuile` & `operator=` (const `VueContenuTuile` &c)

7.17.1 Description détaillée

La classe `VueContenuTuile` est un `QWidget` affichant un type de tuile avec couleur associée et 2 première lettre du type de tuile.

7.17.2 Documentation des constructeurs et destructeur

7.17.2.1 `VueContenuTuile()` [1/2]

```
VueContenuTuile::VueContenuTuile (
    TypesTuiles type,
    bool bouc,
    QWidget * parent = nullptr ) [explicit]
```

Premier constructeur sans placement de meeple. Appel de `setNomCouleurSansM`.

Constructeur n°1

Paramètres

| | |
|---------------|--|
| <i>type</i> | |
| <i>bouc</i> | |
| <i>parent</i> | |

7.17.2.2 VueContenuTuile() [2/2]

```
VueContenuTuile::VueContenuTuile (
    TypesTuiles type,
    bool bouc,
    int id,
    bool meeple,
    QWidget * parent = nullptr ) [explicit]
```

Second constructeur pour placement [Meeple](#), appel de [setNomCouleurAvecM](#).

Constructeur n°2

Paramètres

| | |
|---------------|--|
| <i>type</i> | |
| <i>bouc</i> | |
| <i>id</i> | |
| <i>meeple</i> | |
| <i>parent</i> | |

7.17.3 Documentation des fonctions membres

7.17.3.1 setNomCouleurAvecM()

```
void VueContenuTuile::setNomCouleurAvecM (
    const int & id )
```

Meme fonctionnement que [setNomCouleurSansM](#). rajoute le texte à l'affichage classique des 2 premières lettre pour indiquer la présence de meeple.

Paramètres

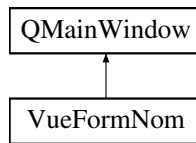
| | |
|-----------|--|
| <i>id</i> | |
|-----------|--|

La documentation de cette classe a été générée à partir du fichier suivant :

— [vueContenuTuile.h](#)

7.18 Référence de la classe VueFormNom

Graphe d'héritage de VueFormNom:



Fonctions membres publiques

- [VueFormNom](#) (int &nJoueur, vector< int > listeNumExt, QWidget *parent=nullptr)
constructeur où sont créé les label et LineEdit, mis par la suite dans des QList pour les récupérer pour [VuePartie](#)
- int [getNbrJoueur](#) () const

7.18.1 Documentation des constructeurs et destructeur

7.18.1.1 VueFormNom()

```

VueFormNom::VueFormNom (
    int & nJoueur,
    vector< int > listeNumExt,
    QWidget * parent = nullptr ) [explicit]
  
```

constructeur où sont créé les label et LineEdit, mis par la suite dans des QList pour les récupérer pour [VuePartie](#)

Constructeur

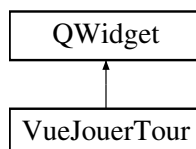
Paramètres

| | |
|--------------------|--|
| <i>nJoueur</i> | |
| <i>listeNumExt</i> | |
| <i>parent</i> | |

La documentation de cette classe a été générée à partir du fichier suivant :
 — [vueFormNom.h](#)

7.19 Référence de la classe VueJouerTour

Graphe d'héritage de VueJouerTour:



Fonctions membres publiques

- [VueJouerTour](#) (QWidget *parent=nullptr)

QWidget contenant un label pour designer le nom du joueur et un QLCDNumber pour designer le nombre de meeple restants.

7.19.1 Documentation des constructeurs et destructeur

7.19.1.1 VueJouerTour()

```
VueJouerTour::VueJouerTour (
    QWidget * parent = nullptr ) [explicit]
```

QWidget contenant un label pour designer le nom du joueur et un QLCDNumber pour designer le nombre de meeple restants.

Constructeur

Paramètres

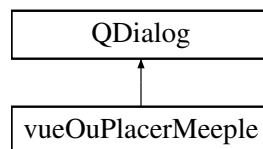
| | |
|---------------|--|
| <i>parent</i> | |
|---------------|--|

La documentation de cette classe a été générée à partir du fichier suivant :

- vueJouerTour.h

7.20 Référence de la classe vueOuPlacerMeeple

Graphe d'héritage de vueOuPlacerMeeple:



Fonctions membres publiques

- [vueOuPlacerMeeple](#) (int l, int c, QWidget *parent=nullptr, [VuePartie](#) *part=nullptr, [Tuile](#) *t=nullptr, [Controller](#) *con=nullptr)
- [vueOuPlacerMeeple](#)
- void [setTuile](#) ([Tuile](#) &)

7.20.1 Documentation des constructeurs et destructeur

7.20.1.1 vueOuPlacerMeeple()

```
vueOuPlacerMeeple::vueOuPlacerMeeple (
    int l,
    int c,
    QWidget * parent = nullptr,
    VuePartie * part = nullptr,
    Tuile * t = nullptr,
    Controller * con = nullptr ) [explicit]
```

vueOuPlacerMeeple

Constructeur

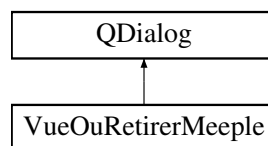
Paramètres

| | |
|---------------|--|
| <i>l</i> | |
| <i>c</i> | |
| <i>parent</i> | |
| <i>part</i> | |
| <i>t</i> | |

La documentation de cette classe a été générée à partir du fichier suivant :
 — vueOuPlacerMeeple.h

7.21 Référence de la classe VueOuRetirerMeeple

Graphe d'héritage de VueOuRetirerMeeple:



Fonctions membres publiques

- [VueOuRetirerMeeple](#) ([VuePartie](#) *partie=nullptr, QWidget *parent=nullptr)
[VueOuRetirerMeeple](#).
- int **containMeeple** ([VueTuile](#) &vt)
- void **setTuile** ([Tuile](#) &tuile, [VueTuile](#) &vt)

7.21.1 Documentation des constructeurs et destructeur

7.21.1.1 VueOuRetirerMeeple()

```
VueOuRetirerMeeple::VueOuRetirerMeeple (
    VuePartie * partie = nullptr,
    QWidget * parent = nullptr ) [explicit]
```

[VueOuRetirerMeeple](#).

Constructeur

Paramètres

| | |
|---------------|--|
| <i>parent</i> | |
| <i>partie</i> | |

La documentation de cette classe a été générée à partir du fichier suivant :

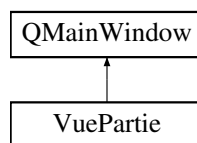
— vueOuRetirerMeeple.h

7.22 Référence de la classe VuePartie

La classe [VuePartie](#) Est decomposer en plusieurs partie : un vertical layout contenant des [VueScore1Joueur](#). 1 layout pour y mettre la Vuetuile a placer. 1 horizontal layout contenant un label et un QLCDNumber designant numero du tour actuel. 1 horizontal layout contenant les QPushButton (valider, fin Partie etc). 1 QWidget contenant un QLabel poru designer le nom du joueur devant jouer ainsi qu'un QLCDNumber designant le nombre de meeple lui restant.

```
#include <vuePartie.h>
```

Graphe d'héritage de VuePartie:



Fonctions membres publiques

- [VuePartie](#) ([Controller](#) *c, QWidget *parent=nullptr)
constructeur appelant setAffichageScore, setAffichageTuile, setJoueurActu, setPlateau
- void **setAffichageScore** ()
Initialise le vertical layout dedie au informations des joueurs : converti le vecteur de [Joueur](#) fourni par le controleur en vecteur de [VueScore1Joueur](#). Affiche le numero du tour actuel.
- void **setAffichageTuile** ()
affiche la tuile a placer en bas a droite de la fenetre.
- void **setJoueurActu** ()
Récupère la string du joueur devant jouer via le controller, la transforme en QString via fromStdString et l'ajoute au label. recupere egalement le nombre de meeple restant au joueur et l'attribut au QLCDNumber.
- void **setPlateau** ()
initialise le plateau : centrage, affichage de la premiere tuile piochée
- void **piocherCarte** ()
- void [placerMeeple](#) (const int Nligne, const int NCol, [Tuile](#) *tuile)
placerMeeple
- void **retirerMeeple** ()
- void [placerTuile](#) (const int Nligne, const int NCol, [Tuile](#) *tuile)
Contruit et ajoute la [VueTuile](#) à partir d'une tuile du modele, a la position donnee en parametre de la fonction, sur le plateau.
- void **updateVueTuileAddM** (int l, int c, int p, [Tuile](#) *T)
- void **updateVueTuileRemoveM** (int l, int c)
- void **partieFinie** ()
supprime l'ensemble des widget non nécessaire pour le décompte des points

Attributs publics

- [Tuile](#) * **tuilePlace**
- [VueTuile](#) * **vueTuilePlace**

7.22.1 Description détaillée

La classe [VuePartie](#) Est decomposer en plusieurs partie : un vertical layout contenant des [VueScore1Joueur](#). 1 layout pour y mettre la Vuetuile a placer. 1 horizontal layout contenant un label et un QLCDNumber designant numero du tour actuel. 1 horizontal layout contenant les QPushButton (valider, fin Partie etc). 1 QWidget contenant un QLabel poru designer le nom du joueur devant jouer ainsi qu'un QLCDNumber designant le nombre de meeple lui restant.

7.22.2 Documentation des constructeurs et destructeur

7.22.2.1 VuePartie()

```
VuePartie::VuePartie (
    Controller * c,
    QWidget * parent = nullptr ) [explicit]
```

constructeur appelant setAffichageScore, setAffichageTuile, setJoueurActu, setPlateau

Constructeur

Paramètres

| | |
|---------------|--|
| <i>c</i> | |
| <i>parent</i> | |

7.22.3 Documentation des fonctions membres

7.22.3.1 placerMeeple()

```
void VuePartie::placerMeeple (
    const int Nligne,
    const int NCol,
    Tuile * tuile )
```

placerMeeple

Paramètres

| | |
|---------------|--|
| <i>Nligne</i> | |
| <i>NCol</i> | |
| <i>tuile</i> | |

7.22.3.2 placerTuile()

```
void VuePartie::placerTuile (
    const int Nligne,
    const int NCol,
    Tuile * tuile )
```

Construit et ajoute la [VueTuile](#) à partir d'une tuile du modele, a la position donnee en parametre de la fonction, sur le plateau.

Paramètres

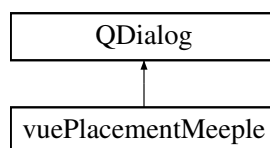
| | |
|---------------|--|
| <i>Nligne</i> | |
| <i>NCol</i> | |
| <i>tuile</i> | |

La documentation de cette classe a été générée à partir du fichier suivant :

— vuePartie.h

7.23 Référence de la classe vuePlacementMeeple

Graphe d'héritage de vuePlacementMeeple:



Fonctions membres publiques

- [vuePlacementMeeple](#) (QWidget *parent=nullptr, [VuePartie](#) *partie=nullptr, const int l=0, const int c=0, [Tuile](#) *tuile=nullptr, [Controller](#) *con=nullptr)
[vuePlacementMeeple](#)
- int [getNligne](#) () const
[getNligne](#)
- int [getNCol](#) () const
[getNCol](#)

7.23.1 Documentation des constructeurs et destructeur

7.23.1.1 vuePlacementMeeple()

```
vuePlacementMeeple::vuePlacementMeeple (
    QWidget * parent = nullptr,
    VuePartie * partie = nullptr,
    const int l = 0,
    const int c = 0,
    Tuile * tuile = nullptr,
    Controller * con = nullptr ) [explicit]
```

vuePlacementMeeple

Constructeur

Paramètres

| | |
|---------------|--|
| <i>parent</i> | |
| <i>partie</i> | |
| <i>l</i> | |
| <i>c</i> | |
| <i>tuile</i> | |

7.23.2 Documentation des fonctions membres

7.23.2.1 getNCol()

```
int vuePlacementMeeple::getNCol ( ) const [inline]
```

getNCol

Renvoie

Ncol

7.23.2.2 getNligne()

```
int vuePlacementMeeple::getNligne ( ) const [inline]
```

getNligne

Renvoie

Nligne

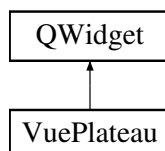
La documentation de cette classe a été générée à partir du fichier suivant :
 — vuePlacementMeeple.h

7.24 Référence de la classe VuePlateau

La classe `VuePlateau` est un `QWidget` contenant un `QWidgetTable`.

```
#include <vuePlateau.h>
```

Graphe d'héritage de `VuePlateau`:



Fonctions membres publiques

- `VuePlateau` (`QWidget *parent=nullptr`)
Appel de `setTable`.
- `void setTable ()`
Permet de centrer la plateau sur la case 72, 72.

7.24.1 Description détaillée

La classe `VuePlateau` est un `QWidget` contenant un `QWidgetTable`.

7.24.2 Documentation des constructeurs et destructeur

7.24.2.1 `VuePlateau()`

```
VuePlateau::VuePlateau (
    QWidget * parent = nullptr ) [explicit]
```

Appel de `setTable`.

Constructeur

Paramètres

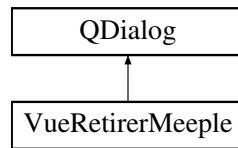
| | |
|---------------------|--|
| <code>parent</code> | |
|---------------------|--|

La documentation de cette classe a été générée à partir du fichier suivant :

- `vuePlateau.h`

7.25 Référence de la classe VueRetirerMeeple

Graphe d'héritage de VueRetirerMeeple:



Fonctions membres publiques

- [VueRetirerMeeple](#) ([VuePartie](#) *partie=nullptr, QWidget *parent=nullptr)
VueRetirerMeeple.

7.25.1 Documentation des constructeurs et destructeur

7.25.1.1 VueRetirerMeeple()

```

VueRetirerMeeple::VueRetirerMeeple (
    VuePartie * partie = nullptr,
    QWidget * parent = nullptr ) [explicit]
  
```

[VueRetirerMeeple.](#)

Constructeur

Paramètres

| | |
|---------------|--|
| <i>parent</i> | |
| <i>partie</i> | |

La documentation de cette classe a été générée à partir du fichier suivant :

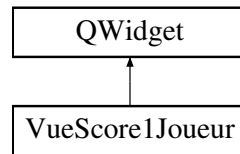
- `vueRetirerMeeple.h`

7.26 Référence de la classe VueScore1Joueur

La classe [VueScore1Joueur](#) est un QWidget constitué d'un vertical layout contenant 2 QLabel pour désigner le nom du joueur et le score, 1 QLCDNumber pour afficher le score et 2 bouton dans un vertical layout pour changer le score.

```
#include <vueScore1Joueur.h>
```

Graphe d'héritage de VueScore1Joueur:



Fonctions membres publiques

- [VueScore1Joueur](#) (std::string nomJ, int id, QWidget *parent=nullptr)

Transforme l'id du joueur en std::string et transforme cette string en QString afin de l'ajouter au label.

7.26.1 Description détaillée

La classe [VueScore1Joueur](#) est un QWidget constitué d'un vertical layout contenant 2 QLabel pour désigner le nom du joueur et le score, 1 QLCDNumber pour afficher le score et 2 bouton dans un vertical layout pour changer le score.

7.26.2 Documentation des constructeurs et destructeur

7.26.2.1 VueScore1Joueur()

```

VueScore1Joueur::VueScore1Joueur (
    std::string nomJ,
    int id,
    QWidget * parent = nullptr ) [explicit]
  
```

Transforme l'id du joueur en std::string et transforme cette string en QString afin de l'ajouter au label.

Constructeur

Paramètres

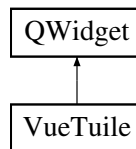
| | |
|---------------|--|
| <i>nomJ</i> | |
| <i>id</i> | |
| <i>parent</i> | |

La documentation de cette classe a été générée à partir du fichier suivant :

- `vueScore1Joueur.h`

7.27 Référence de la classe VueTuile

Graphe d'héritage de VueTuile:



Fonctions membres publiques

- [VueTuile](#) ([Tuile](#) *tuile, QWidget *parent=nullptr)
Construit une Vuetuile à partir d'une tuile du modele. Utilise setContenuTuile.
- [VueTuile](#) ([Tuile](#) *tuile, int p, QWidget *parent=nullptr)
Meme fonction que le constructeur 1 mais on ajoute le placement d'un meeple. Utilise ajoutMeeple.
- void [setContenuTuile](#) ([Tuile](#) &tuile)
Pour chaque case de la tuile, creer la vueContenuTuile associé et l'ajoute au bonne emplacement sur la QGrid.
- void [addMeeple](#) ([Tuile](#) *tuile, int p, int id)
ajoute un meeple
- vector< [VueContenuTuile](#) * > & [getVueContenuT](#) ()
getVueContenuT

7.27.1 Documentation des constructeurs et destructeur

7.27.1.1 [VueTuile\(\)](#) [1/2]

```

VueTuile::VueTuile (
    Tuile * tuile,
    QWidget * parent = nullptr ) [explicit]
  
```

Construit une Vuetuile à partir d'une tuile du modele. Utilise setContenuTuile.

Constructeur n°1

Paramètres

| | |
|---------------|--|
| <i>tuile</i> | |
| <i>parent</i> | |

7.27.1.2 [VueTuile\(\)](#) [2/2]

```

VueTuile::VueTuile (
    Tuile * tuile,
    int p,
    QWidget * parent = nullptr ) [explicit]
  
```

Meme fonction que le constructeur 1 mais on ajoute le placement d'un meeple. Utilise ajoutMeeple.

Paramètres

| | |
|---------------|--|
| <i>tuile</i> | |
| <i>p</i> | |
| <i>parent</i> | |

7.27.2 Documentation des fonctions membres

7.27.2.1 addMeeple()

```
void VueTuile::addMeeple (
    Tuile * tuile,
    int p,
    int id )
```

ajoute un meeple

Paramètres

| | |
|--------------|--|
| <i>tuile</i> | |
| <i>p</i> | |

7.27.2.2 getVueContenuT()

```
vector< VueContenuTuile * > & VueTuile::getVueContenuT ( ) [inline]
```

getVueContenuT

Renvoie

7.27.2.3 setContenuTuile()

```
void VueTuile::setContenuTuile (
    Tuile & tuile )
```

Pour chaque case de la tuile, creer la vueContenuTuile associé et l'ajoute au bonne emplacement sur la QGrid.

Paramètres

| | |
|--------------|--|
| <i>tuile</i> | |
|--------------|--|

La documentation de cette classe a été générée à partir du fichier suivant :

— `vueTuile.h`

Chapitre 8

Documentation des fichiers

8.1 Référence du fichier controller.h

```
#include <iostream>
#include <string>
#include <vector>
#include "pioche.h"
#include "joueur.h"
#include "plateau.h"
#include "tuile.h"
#include "meeple.h"
#include "modeJeu.h"
```

Classes

— class [Controller](#)

Espaces de nommage

— namespace [std](#)

8.2 controller.h

[Aller à la documentation de ce fichier.](#)

```
1 #ifndef _CONTROLLER_H
2 #define _CONTROLLER_H
3 #include <iostream>
4 #include <string>
5 #include <vector>
6 #include "pioche.h"
7 #include "joueur.h"
8 #include "plateau.h"
9 #include "tuile.h"
10 #include "meeple.h"
11 #include "modeJeu.h"
12
28 using namespace std;
29
30 class Controller{
```

```

31
32
33     private:
34
35         bool fini;
36         vector<Joueur*> listeJoueurs;
37         vector<int> extensions;
38         ModeJeu* modeJeu[5];
39         Plateau *plateau;
40         Pioche *pioche;
41         bool riviereActive = false;
42         bool paysansActive = false;
43         bool aubergeActive = false;
44         bool abbeActive = false;
45         int tour;
46         int nbJoueurs;
47         int numJoueurActu; // Va de 0 à nbrJoueur - 1
48
49     public:
50
51     Controller() = default;
52
53     Controller(int nj);
54     Controller(int nj, vector<string> listeNomJoueur, vector<int> listeNumExtensions);
55     //Destructeur
56     ~Controller() = default;
57     Controller(const Controller&) = delete;
58     Controller& operator=(const Controller&) = delete ;
59
60     inline bool getPaysansActive() const {return paysansActive;}
61     inline bool getAbbeActive() const {return abbeActive;}
62     inline bool getRiviereActive() const {return riviereActive;}
63     inline bool getAubergeActive() const {return aubergeActive;}
64
65     Espace* getEspace(const ContenanceTuile* c);
66
67     Espace* getEspaceTuile(const ContenanceTuile& c, const vector<Espace*> e);
68
69     void creerEspace(const Tuile* T);
70
71     void fusionVoisin(const Tuile* tuile);
72
73
74     void placementTuile(Tuile *newTuile, int x, int y);
75
76     void placementMeeple(Joueur* j, Meeple* m, TypesTuiles tm, int i, int x, int y);
77
78     bool placementTuileAutorise(Tuile newTuile);
79
80     //Vérifie si la tuile donnée peut être placée à la position x, y sur le plateau
81     bool estCompatible(Tuile newTuile, int x, int y);
82     // void compteScore(State s);
83     // void compteScore(TypesTuiles t, State s);
84
85     void nextTour();
86     //void compteScoreAbbaye(State s);
87
88     void AffichageJoueurs(){
89         for(int i=0; i<listeJoueurs.size(); i++){
90             cout << "Joueur " << i << " : " << listeJoueurs.at(i)->getNbrMeeples() << " meeples" << endl;
91         }
92     }
93
94     vector<Joueur*> getJoueurs(){
95         return listeJoueurs;
96     }
97
98     //Getters
99     inline bool getFini() const{
100         return this->fini;
101     }
102
103     inline int getNbJoueur() const{
104         return this->nbJoueurs;
105     }
106
107     inline int getTour() const{
108         return this->tour;
109     }
110
111     inline Pioche* getPioche() const{return this->pioche;}
112
113     inline int getNumJoueurActu() const{return this->numJoueurActu;}
114     inline Plateau* getPlateau() const{

```

```

176         return this->plateau;
177     }
178
179     inline vector<int> getExtensions() const{
180         return this->extensions;
181     }
182
183     //retourner le modedejeu
184     inline ModeJeu* getModeJeu(int i) const{
185         return this->modeJeu[i];
186     }
187
188     //Fonction test
189     bool placementTuileAutorise(Tuile newTuile,Plateau* plateau);
190     bool estCompatible(Tuile newTuile,int x,int y,Plateau *plateau);
191     bool validationPlacementRiviere(Tuile newTuile,int x,int y,Plateau *plateau);
192     void placementMeeple(Joueur* j,Meeple* m,TypeMeeple tm,int i,int x,int y,Plateau *plateau);
193     void placementTuile(Tuile *newTuile,int x,int y,Plateau *plateau);
194 };
195
196 #endif

```

8.3 Référence du fichier espace.h

Details.

```

#include <stdio.h>
#include "meeple.h"
#include <vector>

```

Classes

— class [Espace](#)

La classe [Espace](#) contient les différents éléments d'un [Espace](#) (ensemble de ContenuTuile de même type sur plusieurs Tuiles différentes tel qu'une ville ou champ)

Fonctions

— ostream & **operator**<< (ostream &f, const [Espace](#) &E)

8.3.1 Description détaillée

Details.

8.4 espace.h

[Aller à la documentation de ce fichier.](#)

```

1
2
3 #ifndef espace_h
4 #define espace_h
5
6 #include <stdio.h>
7 #include "meeple.h"
8 #include <vector>
9
19 class Espace {
20

```

```

21     friend class Controller;
22
23 private :
24
25     TypesTuiles type;
26     int nbrBouclier;
27     int nbrMeeple;
28     vector<const ContenanceTuile*> contenus;
29     vector<const Meeple*> meeples;
30
31 public :
32
33     Espace(const TypesTuiles& Tt): nbrBouclier(0), nbrMeeple(0), type(Tt){}
34
35
36
37     inline const int getNbrMeeple() const { return nbrMeeple; }
38
39     inline const int getNbrBouclier() const { return nbrBouclier; }
40
41     inline const TypesTuiles& getType() const { return type; }
42
43     inline const vector<const ContenanceTuile*> getContenus() const { return contenus; }
44
45     inline const ContenanceTuile* getContenus(int i) const { return contenus[i]; }
46
47     inline const vector<const Meeple*> getMeeples() const { return meeples; }
48
49     inline const Meeple* getMeeples(int i) const { return meeples[i]; }
50
51     inline const size_t getNbrContenanceTuile() const { return contenus.size(); }
52
53     inline void addBouclier() { nbrBouclier += 1; }
54
55     void addContenance(const ContenanceTuile* C);
56
57
58     void addMeeple(const Meeple* M);
59
60     Espace* fusionEspace(Espace* e);
61
62     bool isComplete();
63
64     inline const bool isFree() const {return meeples.size() == 0;}
65
66     void calculScore();
67 };
68
69 ostream& operator<<(ostream& f, const Espace& E);
70
71 #endif

```

8.5 Référence du fichier joueur.h

Details.

```

#include <stdio.h>
#include <iostream>
#include <vector>
#include <string>
#include "meeple.h"

```

Classes

— class [Joueur](#)

Fonctions

— ostream & **operator**<< (ostream &f, const [Joueur](#) &J)

8.5.1 Description détaillée

Details.

8.6 joueur.h

[Aller à la documentation de ce fichier.](#)

```

1
2 #ifndef joueur_h
3 #define joueur_h
4
5
6 #include <stdio.h>
7 #include <iostream>
8 #include <vector>
9 #include <string>
10 #include "meeple.h"
11
12
13 class Joueur {
14 private:
15     int id;
16     int score;
17     int nbrMeeples;
18     string name;
19     vector<const Meeple*> meeples;
20
21 public:
22     Joueur(int uid, int nb, string str) : id(uid), nbrMeeples(nb), score(0), name(str), meeples(NULL) {}
23     Joueur(int uid) : id(uid){}
24
25     inline const int getId() const { return id; }
26     inline const int getScore() const { return score; }
27     inline const size_t getNbrMeeplesUsed() const { return meeples.size(); }
28     inline const int getNbrMeeples() const { return nbrMeeples; }
29     inline const string getName() const { return name; }
30     inline const vector<const Meeple*> getMeeples() const { return meeples; }
31
32     inline void addScore(int pts) { score += pts; }
33
34     void addMeeple(const Meeple& m);
35
36     void addName(const string& str);
37
38     void removeMeeple() {nbrMeeples--;}
39
40     void setNbMeeple(int i){
41         nbrMeeples = i;
42     }
43 };
44
45 ostream& operator<<(ostream& f, const Joueur& J);
46
47 #endif

```

8.7 meeple.h

```

1
2 #ifndef meeple_h
3 #define meeple_h
4
5
6 #include <stdio.h>
7 #include <iostream>
8 #include "tuile.h"
9
10
11 using namespace std;
12
13

```

```

14
15
20 enum NomMeeple {
21     chevalier,
22     paysan,
23     abbe,
24     voleur,
25     rien
26 };
27
28
29
34 static const char* NomMeeple_str[] = { "chevalier", "paysan", "abbe", "voleur" };
35
36
37
38 class TypeMeeple {
39
40     private:
41
42         NomMeeple nom;
43
44     public:
45
46         TypeMeeple(const NomMeeple& name=rien): nom(name){} // constructeur
47
48         inline const NomMeeple& getNom() const { return nom; }
49         inline void setNom(const NomMeeple& name) { nom = name; }
50 };
51
52 ostream& operator<<(ostream& f, const TypeMeeple& Tm);
53
54
55 class Meeple {
56
57     private:
58
59         ContenanceTuile* contenanceTuile;
60         TypeMeeple type;
61         int id_joueur;
62
63     public:
64
65         Meeple(): contenanceTuile(nullptr) {}
66
67         Meeple(const Meeple&) = delete;
68         Meeple& operator=(const Meeple&) = delete;
69
70         inline bool isUsed() const { return contenanceTuile != nullptr; } // 1 si nullptr 0 sinon
71         inline const TypeMeeple& getType() const { return type; }
72         inline const int getIdJoueur() const { return id_joueur; }
73         inline const ContenanceTuile* getContenanceTuile() const { return contenanceTuile; }
74
75         inline void setType(const TypeMeeple& ty) { type = ty; };
76         inline void setType(const NomMeeple& name) { type.setNom(name); };
77         inline void setIdJoueur(int i) { id_joueur = i; }
78         inline void setContenance(ContenanceTuile& ct) { contenanceTuile = &ct; }
79 };
80
81 ostream& operator<<(ostream& f, const Meeple& M);
82
83 #endif
84

```

8.8 modeJeu.h

```

1 #ifndef _MODEJEU_H
2 #define _MODEJEU_H
3
4 #include <iostream>
5 #include <string>
6 #include <vector>
7 #include "tuile.h"
8 #include "meeple.h"
9 #include "espace.h"

```

```

10 #include "plateau.h"
11
12
30 using namespace std;
31
32
36 class ModeJeu{
37
38     public:
39
48         virtual bool validationPlacementT(Tuile newTuile,int x,int y,Plateau *plateau) = 0;
49
57         virtual bool validationPlacementM(const TypesTuiles& tp,Meeple *m,Espace *e) = 0;
58         virtual void affichage() = 0;
59
60 };
61
62 // ----- EXTENSION STANDARD -----
63
64 class Standard : public ModeJeu{
65     public:
66
67         bool validationPlacementT(Tuile newTuile,int x,int y,Plateau *plateau){
68
69             if(!plateau->existeTuile(x,y-1) && !plateau->existeTuile(x,y+1) && !plateau->existeTuile(x-1,y)
&& !plateau->existeTuile(x+1,y)){
70                 return false;
71             }
72
73             if(plateau->existeTuile(x,y-1)){
74                 if(newTuile.getContenu(1)!=plateau->existeTuile(x,y-1)->getContenu(5)){
75                     return false;
76                 }
77             }
78
79             //Voisin du bas
80
81             if(plateau->existeTuile(x,y+1)){
82                 if(newTuile.getContenu(5)!=plateau->existeTuile(x,y+1)->getContenu(1)){
83                     return false;
84                 }
85             }
86
87             //Voisin de gauche
88             if(plateau->existeTuile(x-1,y)){
89                 if(newTuile.getContenu(7)!=plateau->existeTuile(x-1,y)->getContenu(3)){
90                     return false;
91                 }
92             }
93
94             //Voisin de droite
95             if(plateau->existeTuile(x+1,y)){
96                 if(newTuile.getContenu(3)!=plateau->existeTuile(x+1,y)->getContenu(7)){
97                     return false;
98                 }
99             }
100         }
101
102         return true;
103
104     }
105
106     bool validationPlacementM(const TypesTuiles& tp,Meeple *m ,Espace *e){
107
108         if(e==NULL){
109             return false;
110         }
111
112         if(!e->isFree() || tp==TypesTuiles::champs){
113             return false;
114         }
115         return true;
116     }
117
118     void affichage(){
119         cout<<"*****Standard*****"<<endl;
120     }
121 };
122
123
124 // ----- EXTENSION RIVIERE -----
125
126 class Riviere : public ModeJeu{
127
128     public:
129
130     bool validationPlacementT(Tuile newTuile,int x,int y,Plateau *plateau){

```

```

131
132         if((newTuile.getContenu(0) != TypesTuiles::riviere) &&
133            (newTuile.getContenu(1) != TypesTuiles::riviere) &&
134            (newTuile.getContenu(2) != TypesTuiles::riviere) &&
135            (newTuile.getContenu(3) != TypesTuiles::riviere) &&
136            (newTuile.getContenu(4) != TypesTuiles::riviere) &&
137            (newTuile.getContenu(5) != TypesTuiles::riviere) &&
138            (newTuile.getContenu(6) != TypesTuiles::riviere) &&
139            (newTuile.getContenu(7) != TypesTuiles::riviere) &&
140            (newTuile.getContenu(8) != TypesTuiles::riviere)){ return true; }
141
142
143         if(!plateau->existeTuile(x,y-1) && !plateau->existeTuile(x,y+1) && !plateau->existeTuile(x-1,y)
&& !plateau->existeTuile(x+1,y)){
144             return false;
145         }
146
147         //Voisin du haut
148         if(plateau->existeTuile(x,y-1)) {
149             if(newTuile.getContenu(1) != plateau->existeTuile(x,y-1)->getContenu(5)
150                || (newTuile.getContenu(1) != TypesTuiles::riviere)
151                || (plateau->existeTuile(x,y-1)->getContenu(5) != TypesTuiles::riviere)
152                || (newTuile.getContenu(1) == plateau->existeTuile(x,y-1)->getContenu(5)
153                    && newTuile.getContenu(3) == plateau->existeTuile(x,y-1)->getContenu(3)
154                    && newTuile.getContenu(3) == TypesTuiles::riviere
155                    && newTuile.getContenu(1) == TypesTuiles::riviere)
156                || (newTuile.getContenu(1) == plateau->existeTuile(x,y-1)->getContenu(5)
157                    && newTuile.getContenu(7) == plateau->existeTuile(x,y-1)->getContenu(7)
158                    && newTuile.getContenu(7) == TypesTuiles::riviere
159                    && newTuile.getContenu(1) == TypesTuiles::riviere)) {
160                 return false;
161             }
162         }
163     }
164
165     //Voisin du bas
166
167     if(plateau->existeTuile(x,y+1)) {
168         if(newTuile.getContenu(5) != plateau->existeTuile(x,y+1)->getContenu(1)
169            || (newTuile.getContenu(5) != TypesTuiles::riviere)
170            || (plateau->existeTuile(x,y+1)->getContenu(1) != TypesTuiles::riviere)
171            || (newTuile.getContenu(5) == plateau->existeTuile(x,y+1)->getContenu(1)
172                && newTuile.getContenu(3) == plateau->existeTuile(x,y+1)->getContenu(3)
173                && newTuile.getContenu(5) == TypesTuiles::riviere
174                && newTuile.getContenu(7) == TypesTuiles::riviere)
175            || (newTuile.getContenu(5) == plateau->existeTuile(x,y+1)->getContenu(1)
176                && newTuile.getContenu(7) == plateau->existeTuile(x,y+1)->getContenu(7)
177                && newTuile.getContenu(5) == TypesTuiles::riviere
178                && newTuile.getContenu(3) == TypesTuiles::riviere)) {
179                 return false;
180             }
181         }
182     }
183
184     //Voisin de gauche
185     if(plateau->existeTuile(x-1,y)) {
186         if(newTuile.getContenu(7) != plateau->existeTuile(x-1,y)->getContenu(3)
187            || (newTuile.getContenu(7) != TypesTuiles::riviere)
188            || (plateau->existeTuile(x-1,y)->getContenu(3) != TypesTuiles::riviere)
189            || (newTuile.getContenu(7) == plateau->existeTuile(x-1,y)->getContenu(3)
190                && newTuile.getContenu(1) == plateau->existeTuile(x-1,y)->getContenu(1)
191                && newTuile.getContenu(7) == TypesTuiles::riviere
192                && newTuile.getContenu(1) == TypesTuiles::riviere)
193            || (newTuile.getContenu(7) == plateau->existeTuile(x-1,y)->getContenu(3)
194                && newTuile.getContenu(5) == plateau->existeTuile(x-1,y)->getContenu(5)
195                && newTuile.getContenu(5) == TypesTuiles::riviere
196                && newTuile.getContenu(7) == TypesTuiles::riviere)) {
197                 return false;
198             }
199         }
200     }
201
202     //Voisin de droite
203     if(plateau->existeTuile(x+1,y)) {
204         if(newTuile.getContenu(3) != plateau->existeTuile(x+1,y)->getContenu(7)
205            || (newTuile.getContenu(3) != TypesTuiles::riviere)
206            || (plateau->existeTuile(x+1,y)->getContenu(7) != TypesTuiles::riviere)
207            || (newTuile.getContenu(3) == plateau->existeTuile(x+1,y)->getContenu(7)
208                && newTuile.getContenu(1) == plateau->existeTuile(x+1,y)->getContenu(1)
209                && newTuile.getContenu(3) == TypesTuiles::riviere
210                && newTuile.getContenu(1) == TypesTuiles::riviere)
211            || (newTuile.getContenu(3) == plateau->existeTuile(x+1,y)->getContenu(7)
212                && newTuile.getContenu(5) == plateau->existeTuile(x+1,y)->getContenu(5)
213                && newTuile.getContenu(3) == TypesTuiles::riviere
214                && newTuile.getContenu(5) == TypesTuiles::riviere)) {
215                 return false;
216             }
217         }
218     }

```



```

217
218     }
219
220     return true;
221
222 }
223
224 bool validationPlacementM(const TypesTuiles& tp, Meeple *m, Espace *e){
225     if(e==NULL){
226         return false;
227     }
228     if(!e->isFree() || tp==TypesTuiles::riviere){
229
230         return false;
231     }
232     return true;
233 }
234
235 void affichage(){
236     cout << " Riviere " << endl;
237 }
238 };
239
240 //EXTENSION AUBERGES ET CATHEDRALES
241
242 class AubergesEtCathedrales:public ModeJeu{
243 public:
244     bool validationPlacementT(Tuile newTuile,int x,int y,Plateau *plateau){
245
246
247
248         if(!plateau->existeTuile(x,y-1) && !plateau->existeTuile(x,y+1) && !plateau->existeTuile(x-1,y)
&& !plateau->existeTuile(x+1,y)){
249             return false;
250         }
251
252         if(plateau->existeTuile(x,y-1)){
253             if(newTuile.getContenu(1)!=plateau->existeTuile(x,y-1)->getContenu(5)){
254                 return false;
255             }
256         }
257
258         //Voisin du bas
259
260         if(plateau->existeTuile(x,y+1)){
261             if(newTuile.getContenu(5)!=plateau->existeTuile(x,y+1)->getContenu(1)){
262                 return false;
263             }
264         }
265
266         //Voisin de gauche
267         if(plateau->existeTuile(x-1,y)){
268             if(newTuile.getContenu(7)!=plateau->existeTuile(x-1,y)->getContenu(3)){
269                 return false;
270             }
271         }
272
273         //Voisin de droite
274         if(plateau->existeTuile(x+1,y)){
275             if(newTuile.getContenu(3)!=plateau->existeTuile(x+1,y)->getContenu(7)){
276                 return false;
277             }
278         }
279
280         return true;
281
282     }
283
284     // un meeple ne peut pas occuper une auberge ou une cathédrale
285
286     bool validationPlacementM(const TypesTuiles& tp, Meeple *m, Espace *e){
287         if(e==NULL){
288             return false;
289         }
290
291         if(!e->isFree() || tp==TypesTuiles::auberge || tp==TypesTuiles::cathedrale){
292             return false;
293         }
294         return true;
295     }
296
297     void affichage(){
298         cout<<"AubergesEtCathedrales"<<endl;
299     }
300 };
301
302 // ----- EXTENSION PAYSAN -----

```

```

303
304 class Paysan: public ModeJeu{
305     public:
306
307         bool validationPlacementT(Tuile newTuile,int x,int y,Plateau *plateau){
308
309             if(!plateau->existeTuile(x,y-1) && !plateau->existeTuile(x,y+1) && !plateau->existeTuile(x-1,y)
310             && !plateau->existeTuile(x+1,y)){
311                 return false;
312             }
313
314             if(plateau->existeTuile(x,y-1)){
315                 if(newTuile.getContenu(1)!=plateau->existeTuile(x,y-1)->getContenu(5)){
316                     return false;
317                 }
318             }
319
320
321             //Voisin du bas
322
323             if(plateau->existeTuile(x,y+1)){
324                 if(newTuile.getContenu(5)!=plateau->existeTuile(x,y+1)->getContenu(1)){
325                     return false;
326                 }
327             }
328
329             //Voisin de gauche
330             if(plateau->existeTuile(x-1,y)){
331                 if(newTuile.getContenu(7)!=plateau->existeTuile(x-1,y)->getContenu(3)){
332                     return false;
333                 }
334             }
335
336             //Voisin de droite
337             if(plateau->existeTuile(x+1,y)){
338                 if(newTuile.getContenu(3)!=plateau->existeTuile(x+1,y)->getContenu(7)){
339                     return false;
340                 }
341             }
342
343             return true;
344
345         }
346
347         bool validationPlacementM(const TypesTuiles& tp,Meeple *m ,Espace *e){
348             if(e==NULL){
349                 return false;
350             }
351             if(!e->isFree()) {
352                 return false;
353             }
354             return true;
355         }
356
357         void affichage(){
358             cout<<"Paysan"<<endl;
359         }
360     };
361 };
362
363
364
365 // ----- EXTENSION ABBE -----
366
367
368 class Abbe : public ModeJeu{
369     public:
370
371         bool validationPlacementT(Tuile newTuile,int x,int y,Plateau *plateau){
372
373             if(!plateau->existeTuile(x,y-1) && !plateau->existeTuile(x,y+1) && !plateau->existeTuile(x-1,y)
374             && !plateau->existeTuile(x+1,y)){
375                 return false;
376             }
377
378             if(plateau->existeTuile(x,y-1)){
379                 if(newTuile.getContenu(1)!=plateau->existeTuile(x,y-1)->getContenu(5)){
380                     return false;
381                 }
382             }
383
384             //Voisin du bas
385
386             if(plateau->existeTuile(x,y+1)){

```

```

388         if(newTuile.getContenu(5)!=plateau->existeTuile(x,y+1)->getContenu(1)){
389             return false;
390         }
391     }
392
393     //Voisin de gauche
394     if(plateau->existeTuile(x-1,y)){
395         if(newTuile.getContenu(7)!=plateau->existeTuile(x-1,y)->getContenu(3)){
396             return false;
397         }
398     }
399
400     //Voisin de droite
401     if(plateau->existeTuile(x+1,y)){
402         if(newTuile.getContenu(3)!=plateau->existeTuile(x+1,y)->getContenu(7)){
403             return false;
404         }
405     }
406
407     return true;
408 }
409
410
411 bool validationPlacementM(const TypesTuiles& tp,Meeple *m ,Espace *e){
412     if(e==NULL){
413         return false;
414     }
415
416     //Si un meeple est déjà présent sur l'espace
417     if(!e->isFree()){
418         return false;
419     }
420
421     //L'abbé ne peut être posé que sur une abbaye ou un jardin
422     if(m->getType().getNom()==NomMeeple::abbe){
423         if(tp==TypesTuiles::abbaye || tp==TypesTuiles::jardin){
424             return true;
425         }
426         else{
427             return false;
428         }
429     }
430     //le meeple peut être posé sur n'importe ou sauf sur le jardin
431     else{
432         if(tp==TypesTuiles::jardin){
433             return false;
434         }
435         else{
436             return true;
437         }
438     }
439     return true;
440 }
441 void affichage(){
442     cout<<"Abbe"<<endl;
443 }
444 };
445
446 #endif

```

8.9 Référence du fichier pioche.h

```

#include <stdio.h>
#include "tuile.h"

```

Classes

— class [Pioche](#)

Fonctions

— ostream & **operator**<< (ostream &f, const [Pioche](#) &T)

8.10 pioche.h

[Aller à la documentation de ce fichier.](#)

```

1
2 #ifndef pioche_h
3 #define pioche_h
4
5
6 #include <stdio.h>
7 #include "tuile.h"
8
9
15 class Pioche {
16
17 private :
18     vector<Tuile> tuiles;
19
20
21 public :
22
28     Pioche(vector<int> mode);
29
30     inline const Tuile& getTuile(int i) const {return tuiles[i];}
31     inline size_t getNbTuiles() const {return tuiles.size();}
32
33     Tuile* piocher(int nbTour, bool riviereActive);
34
35     bool estVide() const {return tuiles.size() == 0;}
36
37     Pioche(const Pioche&) = delete;
38     Pioche& operator=(const Pioche&) = delete;
39     inline bool getBouclier(ContenanceTuile& c) { return c.bouclier; };
40
41 };
42
43 ostream& operator<<(ostream& f, const Pioche& T);
44
45 #endif

```

8.11 Référence du fichier plateau.h

```

#include <vector>
#include <iostream>
#include "tuile.h"
#include "espace.h"

```

Classes

— class [Plateau](#)

La classe [Plateau](#) permet de gérer le plateau, avec les tuiles, les espaces.

8.12 plateau.h

[Aller à la documentation de ce fichier.](#)

```

1 #ifndef PROJET_LO21_CARCASSONNE_PLATEAU_H
2 #define PROJET_LO21_CARCASSONNE_PLATEAU_H
3 #include<vector>
4 #include <iostream>
5 #include "tuile.h"
6 #include "espace.h"
7
16 class Plateau{

```

```

17 private:
18     std::vector<Tuile*> tuiles;
19     std::vector<Espace*> espaces;
20     int nbrEspaces;
21     int nbrTuiles;
22
23 public:
24
25     Plateau(int nbT) : tuiles(nbT), nbrTuiles(nbT){};
26     Plateau() = default;
27     ~Plateau();
28     int getNbrTuiles() const{return nbrTuiles;}
29     int getNbrEspaces() const{return nbrEspaces;}
30     void setNbrTuiles(const int &nbrT) {this->nbrTuiles = nbrT;}
31
32     void ajouterTuiles(Tuile* tuile);
33
34     void ajouterEspace(Espace* espace);
35
36     void supprimerEspace(Espace* espace);
37     std::vector<Tuile*> getTuiles() const{return tuiles;}
38     std::vector<Espace*> getEspaces() const {return espaces;}
39
40     Tuile* existeTuile(int x,int y){
41         for(Tuile* tuile : tuiles){
42             if(tuile->getX() == x && tuile->getY() == y){
43                 return tuile;
44             }
45         }
46         return NULL;
47     }
48 };
49 #endif //PROJET_LO21_CARCASSONNE_PLATEAU_H

```

8.13 Référence du fichier tuile.h

```

#include <stdio.h>
#include <vector>
#include <string>
#include <iostream>

```

Classes

- class [ContenanceTuile](#)
- class [Tuile](#)

Espaces de nommage

- namespace [std](#)

Énumérations

- enum [TypesTuiles](#) {
[route](#) , [abbaye](#) , [ville](#) , [champs](#) ,
[riviere](#) , [auberge](#) , [cathedrale](#) , [jardin](#) ,
[droute](#) }

Fonctions

- ostream & **operator**<< (ostream &f, const [Tuile](#) &T)
- ostream & **operator**<< (ostream &f, const [ContenanceTuile](#) &T)

8.13.1 Documentation du type de l'énumération

8.13.1.1 TypesTuiles

enum `TypesTuiles`

Valeurs énumérées

| | |
|------------|---------|
| route | Type 1. |
| abbaye | Type 2. |
| ville | Type 3. |
| champs | Type 4. |
| riviere | Type 5. |
| auberge | Type 6. |
| cathedrale | Type 7. |
| jardin | Type 8. |
| droute | Type 9. |

8.14 tuile.h

[Aller à la documentation de ce fichier.](#)

```
1
2 #ifndef tuile_h
3 #define tuile_h
4
5
6 #include <stdio.h>
7 #include <vector>
8 #include<string>
9 #include<iostream>
10
11 using namespace std;
12
13
14 // Definition enum TypesTuiles
15 enum TypesTuiles {
16     route,
17     abbaye,
18     ville,
19     champs,
20     riviere,
21     auberge,
22     cathedrale,
23     jardin,
24     droute
25 };
26
27 // Convertir Enum en String
28 static const char* TypesTuiles_str[] = { "Route", "Abbaye", "Ville", "Champs", "Riviere", "Auberge",
29     "Cathedrale", "Jardin", "DoubleRoute" };
30
31
32 class ContenanceTuile{
33 private :
34     friend class Pioche;
35     friend class Tuile;
36
37 }
```

```

58     int numPlacement;
59     TypesTuiles type;
60     bool bouclier;
61
62 public:
63
64     ContenanceTuile(const TypesTuiles t, const int n): type(t), numPlacement(n), bouclier(false) {}
65     ContenanceTuile() = default;
66
67     inline const TypesTuiles& getType() const {return type;}
68     inline const int getNumPlacement() const {return numPlacement;}
69     const bool getBouclier() const { return bouclier; }
70
71     void setBouclier();
72     void setType(const TypesTuiles t){
73         type = t;
74     }
75
76     inline bool operator==(const ContenanceTuile& c) const {return this->type == c.getType(); };
77
78     ContenanceTuile(const ContenanceTuile& c):numPlacement(c.numPlacement), type(c.type),
79     bouclier(c.bouclier) {}
80     ContenanceTuile& operator=(const ContenanceTuile& c);
81 };
82
83 class Tuile {
84 private:
85     Tuile* voisin_haut;
86     Tuile* voisin_bas;
87     Tuile* voisin_gauche;
88     Tuile* voisin_droite;
89     int posX;
90     int posY;
91     vector<ContenanceTuile> contenance;
92
93 public :
94     Tuile(const vector<ContenanceTuile>& c);
95     Tuile() = default;
96
97     inline const size_t getSize() const {return contenance.size();}
98     inline const int getX() const {return posX;}
99     inline const int getY() const {return posY;}
100     inline const Tuile* getVoisinHaut() const {return voisin_haut;}
101     inline const Tuile* getVoisinBas() const {return voisin_bas;}
102     inline const Tuile* getVoisinGauche() const {return voisin_gauche;}
103     inline const Tuile* getVoisinDroite() const {return voisin_droite;}
104
105     const vector<ContenanceTuile> getContenance() const {return contenance; }
106
107     const ContenanceTuile& getContenance(int i) const {return contenance[i]; }
108     const ContenanceTuile* getContenancePointeur(int i) const {return &contenance[i]; }
109     ContenanceTuile* getContenancePointeur(int i) {return &contenance[i]; }
110
111     const TypesTuiles& getContenu(int i) const { return contenance[i].getType() ; }
112     void changerOrientation();
113
114     Tuile(const Tuile& T);
115     Tuile& operator=(const Tuile& T);
116     bool operator==(const Tuile& T) const{return this->getContenance() == T.getContenance();}
117
118     void setContenu(int i, TypesTuiles c){
119         contenance[i].setType(c);
120     }
121
122     void ReplaceParChamps(){
123         for (int i = 0; i < 9; i++){
124             if((getContenu(i) == TypesTuiles::auberge) || (getContenu(i)==TypesTuiles::jardin)){
125                 setContenu(i, TypesTuiles::champs);
126             }
127         }
128     }
129
130     void setVoisinHaut(Tuile* t){
131         voisin_haut = t;
132     }
133     void setVoisinBas(Tuile* t){
134         voisin_bas = t;
135     }
136     void setVoisinGauche(Tuile* t){
137         voisin_gauche = t;
138     }

```

```

172     }
173     void setVoisinDroite(Tuile* t){
174         voisin_droite = t;
175     }
176     void setPosX(int x){
177         this->posX = x;
178     }
179     void setPosY(int y){
180         posY = y;
181     }
182
183         //setter x et y
184     void setX(int x){
185         posX = x;
186     }
187     void setY(int y){
188         posY = y;
189     }
190
191 };
192
193
194 ostream& operator<<(ostream& f, const Tuile& T);
195 ostream& operator<<(ostream& f, const ContenanceTuile& T);
196
197
198 #endif /* tuile_h */
199

```

8.15 vueAccueil.h

```

1 #ifndef VUEACCUEIL_H
2 #define VUEACCUEIL_H
3
4 #include <QMainWindow>
5
6 namespace Ui {
7 class VueAccueil;
8 }
9
10 class VueAccueil : public QMainWindow
11 {
12     Q_OBJECT
13
14 public:
15
16     explicit VueAccueil(QWidget *parent = nullptr);
17     ~VueAccueil();
18
19     void setNbrJoueur();
20
21     int getNbrJoueur() const {return this->nbJoueur;}
22 private:
23     Ui::VueAccueil *ui;
24     int nbJoueur;
25
26 private slots:
27     void on_pushButton_clicked();
28 };
29
30 #endif // VUEACCUEIL_H

```

8.16 vueContenuTuile.h

```

1 #ifndef VUECONTENUTUILE_H
2 #define VUECONTENUTUILE_H
3
4 #include <QWidget>
5 #include "../modele/tuile.h"
6
7 namespace Ui {
8 class VueContenuTuile;
9 }
10
11 class VueContenuTuile : public QWidget
12 {
13     Q_OBJECT
14
15 private:

```



```

21     Ui::VueContenuTuile *ui;
22     TypesTuiles typeTuile;
23     bool bouclier;
24     int idJoueurMeeple;
25     bool meeple;
26
27
28 public:
29
30     explicit VueContenuTuile(TypesTuiles type, bool bouc, QWidget *parent = nullptr);
31
32     explicit VueContenuTuile(TypesTuiles type, bool bouc, int id, bool meeple, QWidget *parent = nullptr);
33     ~VueContenuTuile();
34
35     void setNomCouleurSansM();
36
37     void setNomCouleurAvecM(const int& id);
38     inline TypesTuiles getTypeTuile() const {return this->typeTuile;}
39     inline bool getBouclier() const {return this->bouclier;}
40     inline bool getMeeple() const {return this->meeple;}
41
42     VueContenuTuile& operator=(const VueContenuTuile& c);
43 };
44
45 #endif // VUECONTENUTUILE_H

```

8.17 Référence du fichier vueFormNom.h

Espace dédié a la création de label et zone de texte en fonction du nombre de joueur.

```

#include <QMainWindow>
#include <QLabel>
#include <QLineEdit>
#include <QFormLayout>
#include <vector>

```

Classes

— class [VueFormNom](#)

8.17.1 Description détaillée

Espace dédié a la création de label et zone de texte en fonction du nombre de joueur.

8.18 vueFormNom.h

[Aller à la documentation de ce fichier.](#)

```

1 #ifndef VUEFORMNOM_H
2 #define VUEFORMNOM_H
3
4 #include <QMainWindow>
5 #include <QLabel>
6 #include <QLineEdit>
7 #include <QFormLayout>
8 #include <vector>
9 using namespace std;
10
11
12
13 namespace Ui {
14 class VueFormNom;

```

```

19 }
20
21 class VueFormNom : public QMainWindow
22 {
23     Q_OBJECT
24
25 public:
26
27     explicit VueFormNom(int& nJoueur, vector<int> listeNumExt, QWidget *parent = nullptr);
28     ~VueFormNom();
29     int getNbrJoueur() const {return this->nbrJoueur;}
30
31 private slots:
32
33     void on_pushButton_2_clicked();
34
35 private:
36     Ui::VueFormNom *ui;
37     QWidget* m_window;
38     int nbrJoueur;
39     QList<QLineEdit*> m_listLineEdit;
40     QList<QLabel*> m_listLabel;
41     QLabel* L_prenom;
42     QLineEdit* m_prenom;
43     vector<int> listeNumE;
44 };
45
46 #endif // VUEFORMNOM_H

```

8.19 vueJouerTour.h

```

1 #ifndef VUEJOUERTOUR_H
2 #define VUEJOUERTOUR_H
3
4 #include <QWidget>
5
6 namespace Ui {
7 class VueJouerTour;
8 }
9
10 class VueJouerTour : public QWidget
11 {
12     Q_OBJECT
13
14 public:
15
16     explicit VueJouerTour(QWidget *parent = nullptr);
17     ~VueJouerTour();
18
19 private:
20     Ui::VueJouerTour *ui;
21 };
22
23 #endif // VUEJOUERTOUR_H

```

8.20 vueOuPlacerMeeple.h

```

1 #ifndef VUEOUPLACERMEEPLE_H
2 #define VUEOUPLACERMEEPLE_H
3
4 #include <QDialog>
5 #include "vuePartie.h"
6 #include "modele/tuile.h"
7 #include <QAbstractButton>
8
9
10 namespace Ui {
11 class vueOuPlacerMeeple;
12 }
13
14 class vueOuPlacerMeeple : public QDialog
15 {
16     Q_OBJECT
17
18

```

```

19 public:
20
21     explicit vueOuPlacerMeeple(int l,int c, QWidget *parent = nullptr, VuePartie* part = nullptr, Tuile*
22     t = nullptr, Controller *con=nullptr);
23     ~vueOuPlacerMeeple();
24     void setTuile(Tuile&);
25
26 private slots:
27     void on_c0_clicked();
28     void on_c1_clicked();
29     void on_c2_clicked();
30     void on_c3_clicked();
31     void on_c4_clicked();
32     void on_c5_clicked();
33     void on_c6_clicked();
34     void on_c7_clicked();
35     void on_c8_clicked();
36
37 private:
38     Ui::vueOuPlacerMeeple *ui;
39     VuePartie* partie;
40     int Nligne;
41     int Ncol;
42     Tuile* tuile;
43     Controller *controller;
44 };
45
46 #endif // VUEOUPLACERMEEPLE_H

```

8.21 vueOuRetirerMeeple.h

```

1 #ifndef VUEOURETIRERMEEPLE_H
2 #define VUEOURETIRERMEEPLE_H
3
4 #include <QDialog>
5 #include "vuePartie.h"
6 #include "modele/tuile.h"
7 #include <QAbstractButton>
8
9
10
11 namespace Ui {
12 class VueOuRetirerMeeple;
13 }
14
15
16 class VueOuRetirerMeeple : public QDialog
17 {
18     Q_OBJECT
19
20 public:
21
22     explicit VueOuRetirerMeeple(VuePartie* partie=nullptr,QWidget *parent=nullptr);
23     ~VueOuRetirerMeeple();
24
25     int containMeeple(VueTuile& vt);
26     void setTuile(Tuile& tuile,VueTuile& vt);
27
28 private slots:
29     void on_OK_clicked();
30
31 private:
32     Ui::VueOuRetirerMeeple *ui;
33     VuePartie* partie;
34 };
35
36 #endif // VUEOURETIRERMEEPLE_H

```

8.22 vuePartie.h

```

1 #ifndef VUEPARTIE_H
2 #define VUEPARTIE_H
3
4 #include <QMainWindow>
5 #include "vueTuile.h"
6 #include "../modele/controller.h"
7
8

```

```

9 namespace Ui {
10 class VuePartie;
11 }
12
16 class VuePartie : public QMainWindow
17 {
18     Q_OBJECT
19
20 public:
21
28     explicit VuePartie(Controller* c, QWidget *parent = nullptr);
29     ~VuePartie();
30
34     void setAffichageScore();
35
39     void setAffichageTuile();
40
44     void setJoueurActu();
45
49     void setPlateau();
50     void piocherCarte();
51
58     void placerMeeple(const int Nligne, const int NCol, Tuile* tuile);
59     void retirerMeeple();
60
67     void placerTuile(const int Nligne, const int NCol, Tuile* tuile);
68
69     void updateVueTuileAddM(int l, int c, int p, Tuile* T);
70     void updateVueTuileRemoveM(int l, int c);
71
75     void partieFinie();
76
77     Tuile *tuilePlace;
78     VueTuile* vueTuilePlace ;
79
80
81 private slots:
82
86     void on_zoomIn_clicked();
87
91     void on_zoomOut_clicked();
92
96     void on_bouttonValiderTuile_clicked();
97
101    void on_rotationTuile_clicked();
102
103
104
105
106    void on_bouttonFinPartie_clicked();
107
108 private:
109     Ui::VuePartie *ui;
110     Controller* controller;
111
112 };
113
114 #endif // VUEPARTIE_H

```

8.23 vuePlacementMeeple.h

```

1 #ifndef VUEPLACEMENTMEEPLE_H
2 #define VUEPLACEMENTMEEPLE_H
3
4 #include "modele/tuile.h"
5 #include <QDialog>
6 #include <QAbstractButton>
7 #include "vuePartie.h"
8 #include "vueOuPlacerMeeple.h"
9
10
11 namespace Ui {
12 class vuePlacementMeeple;
13 }
14
15 class vuePlacementMeeple : public QDialog
16 {
17     Q_OBJECT
18
19 public:
20
30     explicit vuePlacementMeeple(QWidget *parent = nullptr, VuePartie* partie = nullptr, const int l = 0,
        const int c = 0, Tuile* tuile = nullptr, Controller* con=nullptr);

```

```

31     ~vuePlacementMeeple();
32
33     inline int getNlign()const {return Nlign;}
34
35     inline int getNCol()const {return NCol;}
36
37 private slots:
38
39     void on_Non_clicked();
40     void on_Oui_clicked();
41
42 private:
43     Ui::vuePlacementMeeple *ui;
44     VuePartie* partie;
45     const int Nlign;
46     const int NCol;
47     Tuile* tuile;
48     Controller* controller;
49 };
50
51 #endif // VUEPLACEMENTMEEPLE_H

```

8.24 vuePlateau.h

```

1 #ifndef VUEPLATEAU_H
2 #define VUEPLATEAU_H
3
4 #include <QWidget>
5
6 namespace Ui {
7 class VuePlateau;
8 }
9
10 class VuePlateau : public QWidget
11 {
12     Q_OBJECT
13
14 public:
15
16     explicit VuePlateau(QWidget *parent = nullptr);
17
18     void setTable();
19     ~VuePlateau();
20
21 private:
22     Ui::VuePlateau *ui;
23 };
24
25 #endif // VUEPLATEAU_H

```

8.25 vueRetirerMeeple.h

```

1 #ifndef VUERETIRERMEEPLE_H
2 #define VUERETIRERMEEPLE_H
3
4 #include <QDialog>
5 #include "vuePartie.h"
6 #include "modele/tuile.h"
7 #include <QAbstractButton>
8 #include "vueOuRetirerMeeple.h"
9
10
11 namespace Ui {
12 class VueRetirerMeeple;
13 }
14
15 class VueRetirerMeeple : public QDialog
16 {
17     Q_OBJECT
18
19 public:
20
21     explicit VueRetirerMeeple(VuePartie* partie=nullptr, QWidget *parent=nullptr);
22     ~VueRetirerMeeple();
23
24 private slots:
25     void on_oui_clicked();

```

```

33     void on_non_clicked();
34
35
36 private:
37     Ui::VueRetirerMeeple *ui;
38     VuePartie* partie;
39 };
40
41 #endif // VUERETIRERMEEPLE_H

```

8.26 vueScore1Joueur.h

```

1 #ifndef VUESCORE1JOUEUR_H
2 #define VUESCORE1JOUEUR_H
3
4 #include <QWidget>
5
6 namespace Ui {
7 class VueScore1Joueur;
8 }
9
13 class VueScore1Joueur : public QWidget
14 {
15     Q_OBJECT
16
17 public:
18
19     explicit VueScore1Joueur(std::string nomJ,int id, QWidget *parent = nullptr);
20     ~VueScore1Joueur();
21
22 private:
23     Ui::VueScore1Joueur *ui;
24     std::string nom;
25     int id;
26
27 private slots:
28     void on_ajouterPoints_clicked();
29     void on_retirerPoints_clicked();
30 };
31
32 #endif // VUESCORE1JOUEUR_H

```

8.27 vueTuile.h

```

1 #ifndef VUETUILE_H
2 #define VUETUILE_H
3
4 #include <QWidget>
5 #include "../modele/tuile.h"
6 #include <vector>
7 #include "vueContenuTuile.h"
8
9
10 namespace Ui {
11 class VueTuile;
12 }
13
14 class VueTuile : public QWidget
15 {
16     Q_OBJECT
17
18 public:
19
20     explicit VueTuile(Tuile* tuile ,QWidget *parent = nullptr);
21
22     explicit VueTuile(Tuile* tuile, int p, QWidget *parent = nullptr);
23     ~VueTuile();
24
25     void setContenuTuile(Tuile& tuile);
26
27     void addMeeple(Tuile* tuile, int p,int id);
28
29     inline vector<VueContenuTuile*>& getVueContenuT() {return tabContenuTuile;}
30
31 private:
32     Ui::VueTuile *ui;
33     vector<VueContenuTuile*> tabContenuTuile;

```

```
59 };  
60  
61 #endif // VUETUILE_H
```


Index

- abbaye
 - tuile.h, [70](#)
- Abbe, [13](#)
 - affichage, [13](#)
 - validationPlacementM, [13](#)
 - validationPlacementT, [14](#)
- addContenance
 - Espace, [22](#)
- addMeeple
 - Espace, [22](#)
 - Joueur, [26](#)
 - VueTuile, [55](#)
- addName
 - Joueur, [26](#)
- addScore
 - Joueur, [26](#)
- affichage
 - Abbe, [13](#)
 - AubergesEtCathedrales, [15](#)
 - Paysan, [30](#)
 - Riviere, [35](#)
 - Standard, [37](#)
- ajouterEspace
 - Plateau, [33](#)
- ajouterTuiles
 - Plateau, [34](#)
- auberge
 - tuile.h, [70](#)
- AubergesEtCathedrales, [14](#)
 - affichage, [15](#)
 - validationPlacementM, [15](#)
 - validationPlacementT, [15](#)
- cathedrale
 - tuile.h, [70](#)
- champs
 - tuile.h, [70](#)
- ContenanceTuile, [16](#)
 - ContenanceTuile, [16](#)
- Controller, [17](#)
 - Controller, [18](#)
 - creerEspace, [18](#)
 - estCompatible, [18](#)
 - fusionVoisin, [19](#)
 - getEspace, [19](#)
 - getEspaceTuile, [19](#)
 - placementMeeple, [20](#)
 - placementTuile, [20](#)
 - placementTuileAutorise, [21](#)
- controller.h, [57](#)
- creerEspace
 - Controller, [18](#)
- droute
 - tuile.h, [70](#)
- Espace, [21](#)
 - addContenance, [22](#)
 - addMeeple, [22](#)
 - Espace, [22](#)
 - getContenus, [23](#)
 - getMeeples, [23](#)
 - getNbrBouclier, [23](#)
 - getNbrContenanceTuile, [23](#)
 - getNbrMeeple, [24](#)
 - getType, [24](#)
 - isComplete, [24](#)
 - isFree, [24](#)
- espace.h, [59](#)
- estCompatible
 - Controller, [18](#)
- estVide
 - Pioche, [32](#)
- existeTuile
 - Plateau, [34](#)
- fusionVoisin
 - Controller, [19](#)
- getContenance
 - Tuile, [39](#)
- getContenu
 - Tuile, [39](#)
- getContenus
 - Espace, [23](#)
- getEspace
 - Controller, [19](#)
- getEspaceTuile
 - Controller, [19](#)
- getMeeples
 - Espace, [23](#)
- getNbrBouclier
 - Espace, [23](#)
- getNbrContenanceTuile
 - Espace, [23](#)
- getNbrJoueur
 - VueAccueil, [41](#)
- getNbrMeeple
 - Espace, [24](#)
- getNCol

- vuePlacementMeeple, 50
- getNligne
 - vuePlacementMeeple, 50
- getType
 - Espace, 24
- getVueContenuT
 - VueTuile, 55
- isComplete
 - Espace, 24
- isFree
 - Espace, 24
- jardin
 - tuile.h, 70
- Joueur, 25
 - addMeeple, 26
 - addName, 26
 - addScore, 26
 - Joueur, 25
 - setNbMeeple, 27
- jeueur.h, 60
- Meeple, 27
 - Meeple, 27
- ModeJeu, 28
 - validationPlacementM, 28
 - validationPlacementT, 29
- Paysan, 29
 - affichage, 30
 - validationPlacementM, 30
 - validationPlacementT, 30
- Pioche, 31
 - estVide, 32
 - Pioche, 31
 - piocher, 32
- pioche.h, 67
- piocher
 - Pioche, 32
- placementMeeple
 - Controller, 20
- placementTuile
 - Controller, 20
- placementTuileAutorise
 - Controller, 21
- placerMeeple
 - VuePartie, 48
- placerTuile
 - VuePartie, 49
- Plateau, 32
 - ajouterEspace, 33
 - ajouterTuiles, 34
 - existeTuile, 34
 - Plateau, 33
 - supprimerEspace, 34
- plateau.h, 68
- Riviere, 35
 - affichage, 35
 - validationPlacementM, 35
 - validationPlacementT, 36
- riviere
 - tuile.h, 70
- route
 - tuile.h, 70
- setContentTuile
 - VueTuile, 55
- setNbMeeple
 - Joueur, 27
- setNomCouleurAvecM
 - VueContenuTuile, 43
- Standard, 36
 - affichage, 37
 - validationPlacementM, 37
 - validationPlacementT, 37
- std, 11
- supprimerEspace
 - Plateau, 34
- Tuile, 38
 - getContenance, 39
 - getContenu, 39
 - Tuile, 38
- tuile.h, 69
 - abbaye, 70
 - auberge, 70
 - cathedrale, 70
 - champs, 70
 - droute, 70
 - jardin, 70
 - riviere, 70
 - route, 70
 - TypesTuiles, 70
 - ville, 70
- TypeMeeple, 39
 - TypeMeeple, 40
- TypesTuiles
 - tuile.h, 70
- validationPlacementM
 - Abbe, 13
 - AubergesEtCathedrales, 15
 - ModeJeu, 28
 - Paysan, 30
 - Riviere, 35
 - Standard, 37
- validationPlacementT
 - Abbe, 14
 - AubergesEtCathedrales, 15
 - ModeJeu, 29
 - Paysan, 30
 - Riviere, 36
 - Standard, 37
- ville
 - tuile.h, 70
- VueAccueil, 40

- getNbrJoueur, [41](#)
- VueAccueil, [41](#)
- VueContenuTuile, [42](#)
 - setNomCouleurAvecM, [43](#)
 - VueContenuTuile, [42](#), [43](#)
- VueFormNom, [44](#)
 - VueFormNom, [44](#)
- vueFormNom.h, [73](#)
- VueJouerTour, [44](#)
 - VueJouerTour, [45](#)
- vueOuPlacerMeeple, [45](#)
 - vueOuPlacerMeeple, [45](#)
- VueOuRetirerMeeple, [46](#)
 - VueOuRetirerMeeple, [46](#)
- VuePartie, [47](#)
 - placerMeeple, [48](#)
 - placerTuile, [49](#)
 - VuePartie, [48](#)
- vuePlacementMeeple, [49](#)
 - getNCol, [50](#)
 - getNligne, [50](#)
 - vuePlacementMeeple, [50](#)
- VuePlateau, [51](#)
 - VuePlateau, [51](#)
- VueRetirerMeeple, [52](#)
 - VueRetirerMeeple, [52](#)
- VueScore1Joueur, [52](#)
 - VueScore1Joueur, [53](#)
- VueTuile, [53](#)
 - addMeeple, [55](#)
 - getVueContenuT, [55](#)
 - setContenuTuile, [55](#)
 - VueTuile, [54](#)