TowerDefense

Généré par Doxygen 1.7.4

Thu Jun 2 2011 15 :53 :40

Table des matières

Chapitre 1

Index des classes

1.1 Hiérarchie des classes

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

Chapitre 2

Index des classes

2.1 Liste des classes

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

Chapitre 3

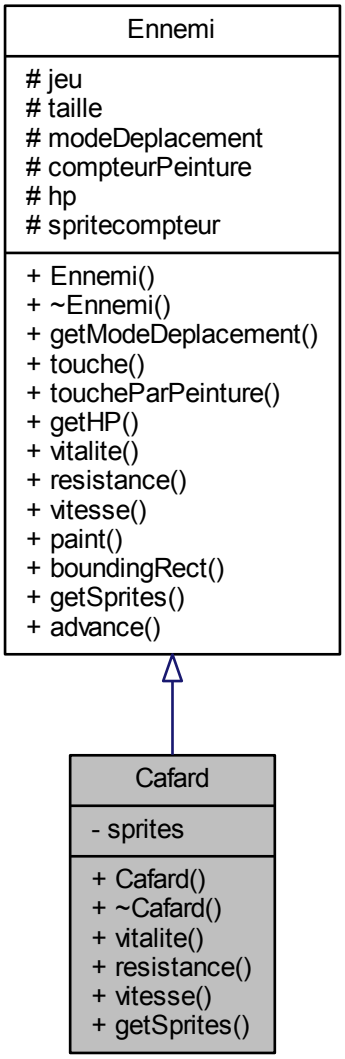
Documentation des classes

3.1 Référence de la classe Cafard

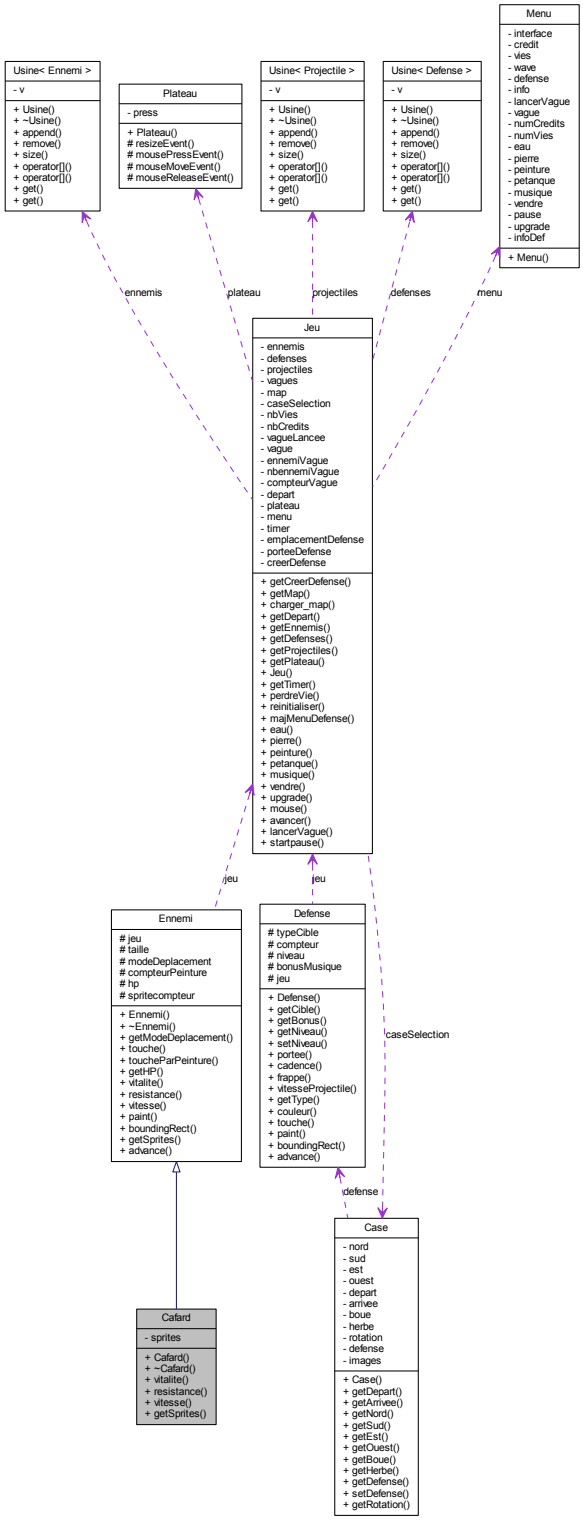
Insecte rampant avec forte résistance.

`#include <ennemi.h>`

Graphe d'héritage de Cafard :



Graphe de collaboration de Cafard :



Fonctions membres publiques

– Cafard (Jeu *jeu, float taille)

Constructeur.

– ~Cafard ()

Déstructeur spécifique lors de la mort d'un cafard.

– qreal vitalite ()

Définition de la vitalité

– qreal resistance ()

Définition de la résistance.

– qreal vitesse ()

Définition de la vitesse.

– QList< QPixmap > & getSprites ()

Définition des sprites.

3.1.1 Description détaillée

Insecte rampant avec forte résistance.

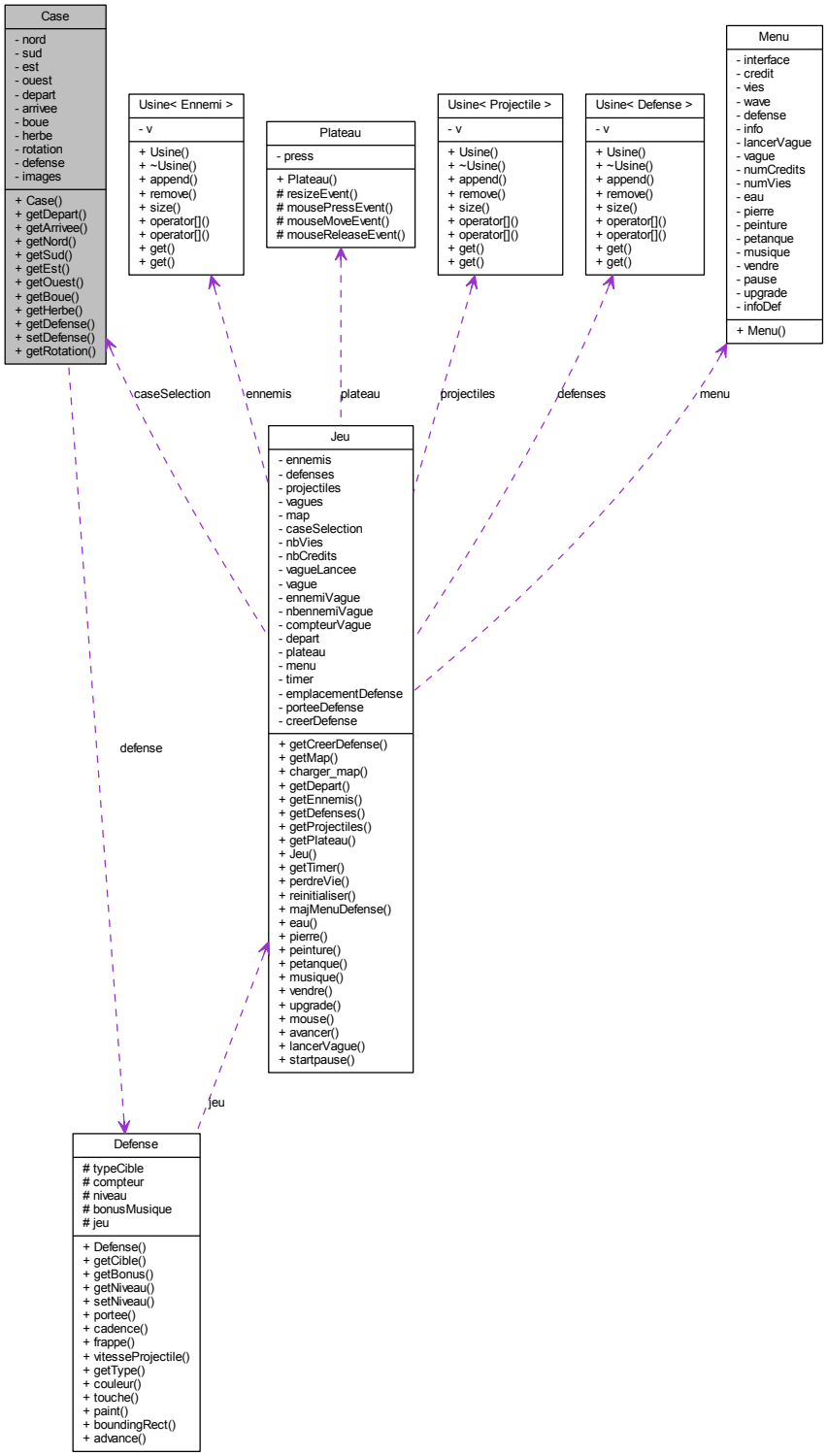
La documentation de cette classe a été générée à partir des fichiers suivants :
– ennemi.h
– cafard.cpp

3.2 Référence de la classe Case

Objet graphique représentant une case du terrain.

`#include <case.h>`

Graphe de collaboration de Case :



Fonctions membres publiques

- Case (int n)
- bool getDepart () const

  Accesseur indiquant si c'est une case de départ.
- bool getArrivee () const

  Accesseur indiquant si c'est une case d'arrivée.
- bool getNord () const

  Accesseur indiquant si c'est une case orientée vers le Nord.
- bool getSud () const

  Accesseur indiquant si c'est une case orientée vers le Sud.
- bool getEst () const

  Accesseur indiquant si c'est une case orientée vers l'Est.
- bool getOuest () const

  Accesseur indiquant si c'est une case orientée vers l'Ouest.
- bool getBoue () const

  Accesseur indiquant si c'est une case de type Boue.
- bool getHerbe () const

  Accesseur indiquant si c'est une case de type Herbe.
- Defense * getDefense () const
- void setDefense (Defense *def)
- int getRotation () const

3.2.1 Description détaillée

Objet graphique représentant une case du terrain.

3.2.2 Documentation des constructeurs et destructeur

3.2.2.1 Case : :Case (int *n*)

Construit une case en fonction du nombre chargé dans le fichier map.txt

Paramètres

n	Nombre représentant le type de la case

3.2.3 Documentation des fonctions membres

3.2.3.1 Defense* Case : :getDefense ( ) const [inline]

Accesseur indiquant l'éventuelle défense sur la case

Renvoie

Un pointeur vers l'éventuelle défense

3.2.3.2 int Case : :getRotation ( ) const [inline]

Accesseur indiquant la rotation de la case en degrés

Renvoie

degrés de rotation

3.2.3.3 void Case : :setDefense (Defense * *def*) [inline]

Met en place une défense sur la case

Paramètres

def	Pointeur sur la défense

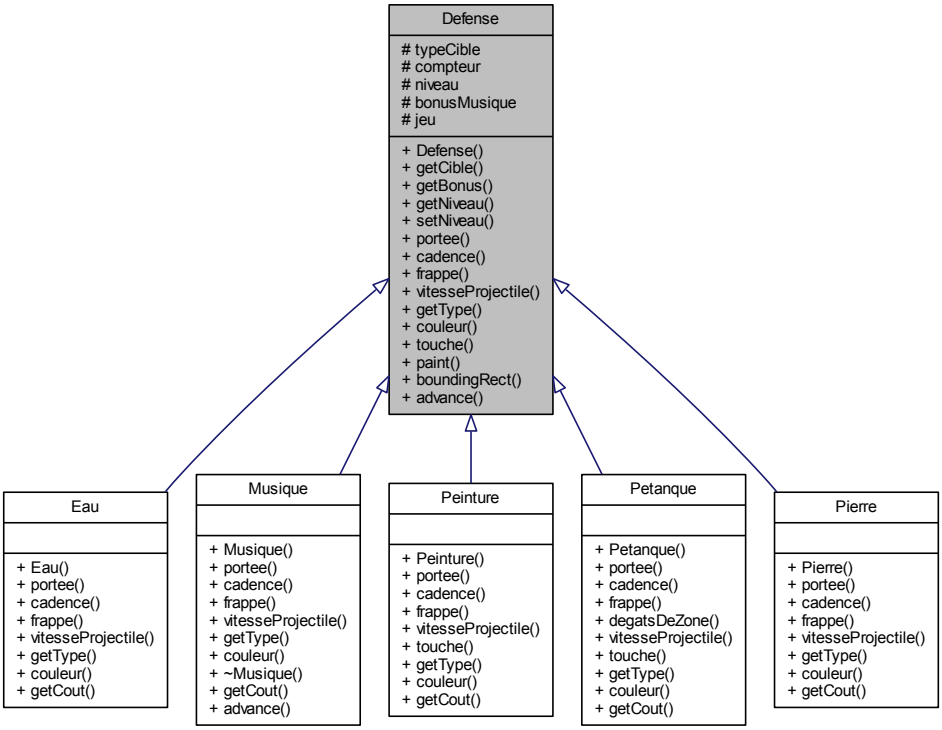
La documentation de cette classe a été générée à partir des fichiers suivants :
– case.h
– case.cpp

3.3 Référence de la classe Defense

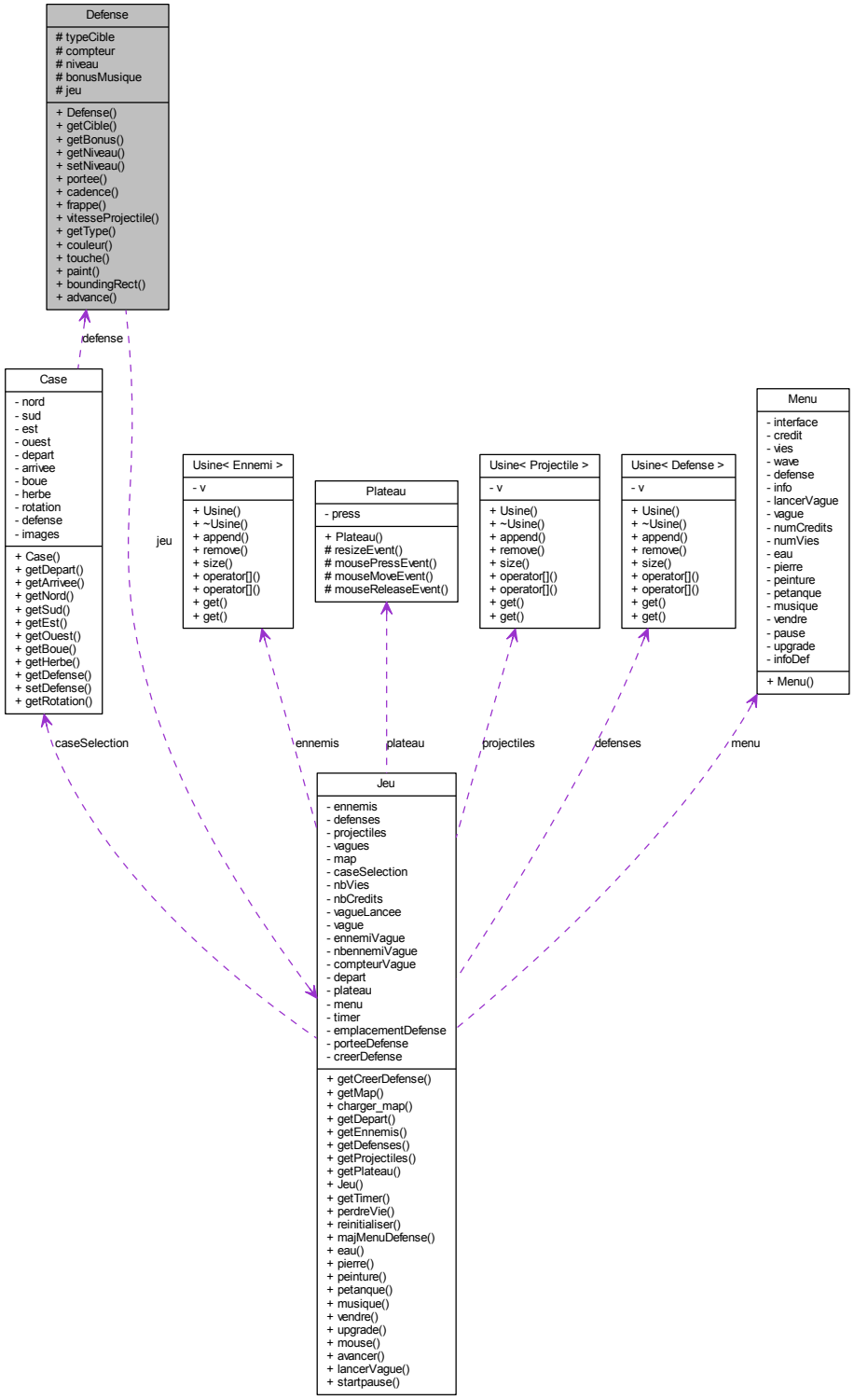
Objet graphique représentant une défense.

`#include <defense.h>`

Graphe d'héritage de Defense :



Graphe de collaboration de Defense :



Connecteurs publics

– virtual void advance (int phase)

Action à faire à chaque frame.

Fonctions membres publiques

– Defense (Jeu ∗jeu)
– QString getCible ()
– float & getBonus ()
– int getNiveau ()
– void setNiveau (int n)
– virtual qreal portee ()=0
– virtual qreal cadence ()=0
– virtual qreal frappe ()=0
– virtual qreal vitesseProjectile ()=0
– virtual QString getType ()=0
– virtual const QColor & couleur () const =0
– virtual void touche (Ennemi ∗cible)
– virtual void paint (QPainter ∗painter, const QStyleOptionGraphicsItem ∗option, QWidget ∗widget)

Affichage de la défense.

– QRectF boundingRect () const

Carré d'affichage de la défense.

Attributs protégés

– QString typeCible

Type de cible que la défense peut attaquer.

– int compteur

Compteur pour le lancement des projectiles.

– int niveau

Niveau d'upgrade.

– float bonusMusique

Bonus actuel donné par d'éventuels musiciens.

– Jeu ∗ jeu

Jeu sur lequel elle est placée.

3.3.1 Description détaillée

Objet graphique représentant une défense.

3.3.2 Documentation des constructeurs et destructeur

3.3.2.1 Defense : :Defense (Jeu ∗ *jeu*)

Construit une défense sur le jeu

Paramètres

jeu	Jeu sur lequel elle est construite

3.3.3 Documentation des fonctions membres

3.3.3.1 virtual qreal Defense : :cadence ( ) [pure virtual]

Méthode virtuelle pure

Renvoie

cadence de la défense en tirs/seconde

Implémenté dans Eau, Pierre, Petanque, Peinture, et Musique.

3.3.3.2 virtual const QColor& Defense : :couleur ( ) const [pure virtual]

Méthode virtuelle pure

Renvoie

Couleur de la défense

Implémenté dans Eau, Pierre, Petanque, Peinture, et Musique.

3.3.3.3 virtual qreal Defense : :frappe ( ) [pure virtual]

Méthode virtuelle pure

Renvoie

frappe de la défense

Implémenté dans Eau, Pierre, Petanque, Peinture, et Musique.

3.3.3.4 float& Defense : :getBonus ( ) [inline]

Accesseur indiquant le bonus par d'éventuels musiciens

Renvoie

la valeur actuelle du bonus

3.3.3.5 QString Defense : :getCible ( ) [inline]

Accesseur du type d'ennemi que la défense peut attaquer

Renvoie

type de cible

3.3.3.6 int Defense : :getNiveau ( ) [inline]

Accesseur du niveau de la défense

Renvoie

le niveau de la défense

3.3.3.7 virtual QString Defense : :getType ( ) [pure virtual]

Méthode virtuelle pure

Renvoie

QString représentant le type de la défense

Implémenté dans Eau, Pierre, Petanque, Peinture, et Musique.

3.3.3.8 virtual qreal Defense : :portee ( ) [pure virtual]

Méthode virtuelle pure

Renvoie

portée de la défense

Implémenté dans Eau, Pierre, Petanque, Peinture, et Musique.

3.3.3.9 void Defense : :setNiveau (int *n*) [inline]

Change le niveau de la défense

Paramètres

n	le nouveau niveau

3.3.3.10 void Defense : :touche (Ennemi * *cible*) [virtual]

Action à faire quand le projectile de la défense atteinte une cible

Paramètres

cible	Ennemi touché par le projectile

Réimplémentée dans Petanque, et Peinture.

3.3.3.11 virtual qreal Defense : :vitesseProjectile ( ) [pure virtual]

Méthode virtuelle pure

Renvoie

vitesse du projectile en cases/seconde

Implémenté dans Eau, Pierre, Petanque, Peinture, et Musique.

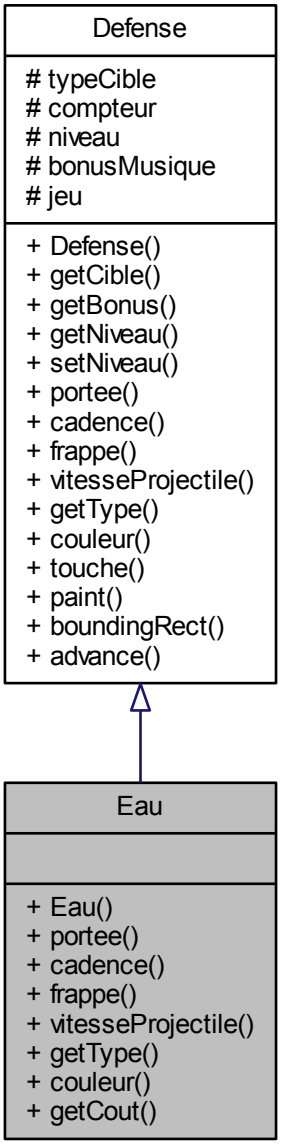
La documentation de cette classe a été générée à partir des fichiers suivants :
– defense.h
– defense.cpp

3.4 Référence de la classe Eau

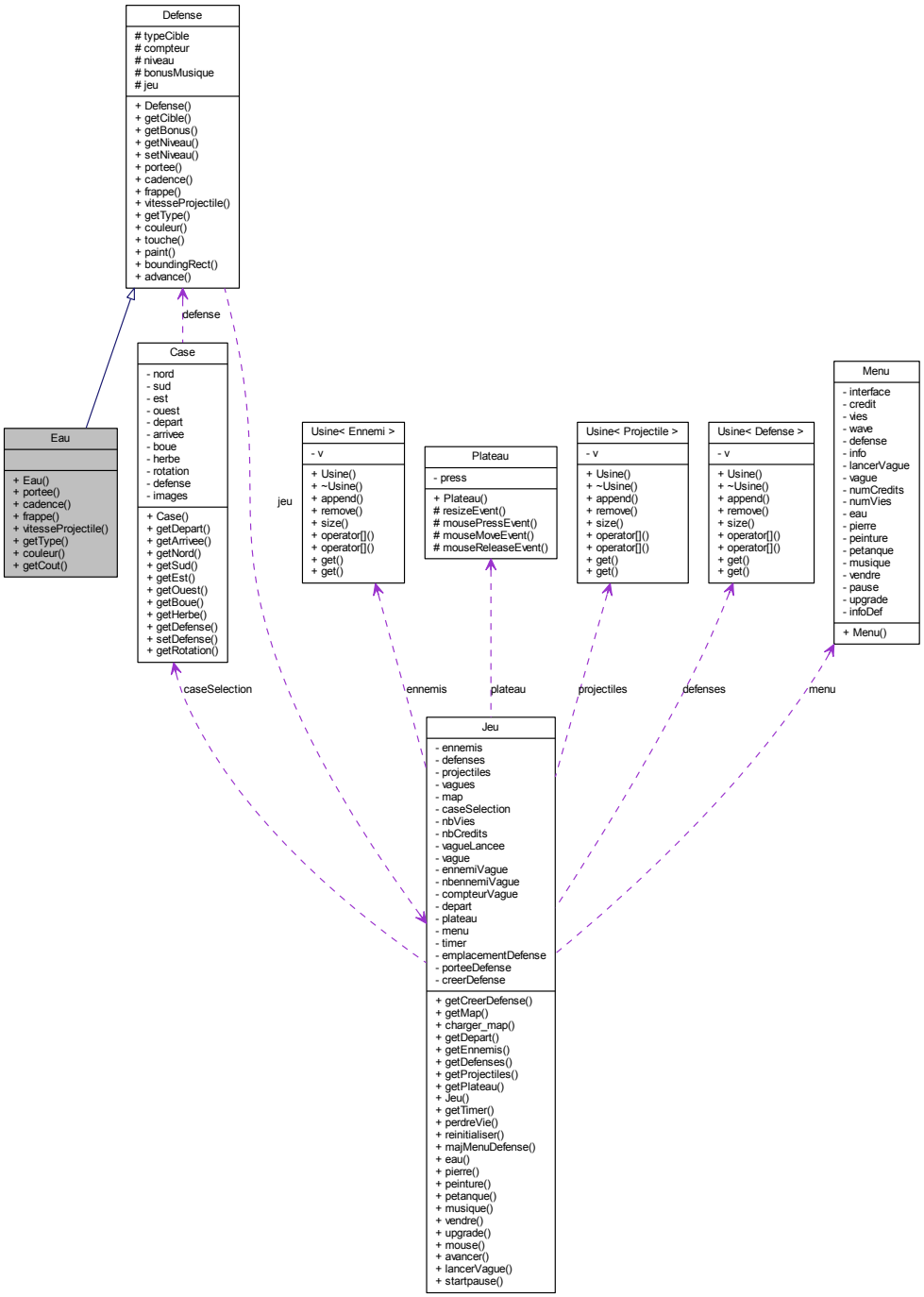
Défense peu éfficace mais rapide.

`#include <defense.h>`

Graphe d'héritage de Eau :



Graphe de collaboration de Eau :



Fonctions membres publiques

- Eau (Jeu *jeu)

  Constructeur.
- qreal portee ()

  Définition de la portée.
- qreal cadence ()

  Définition de la cadence de tir.
- qreal frappe ()

  Définition de la puissance de frappe.
- qreal vitesseProjectile ()

  Définition de la vitesse des projectiles.
- QString getType ()

  Définition du type.
- const QColor & couleur () const

  Définition de la couleur.

Fonctions membres publiques statiques

- static int getCout (int niveau)

  Définition du coût.

3.4.1 Description détaillée

Défense peu efficace mais rapide.

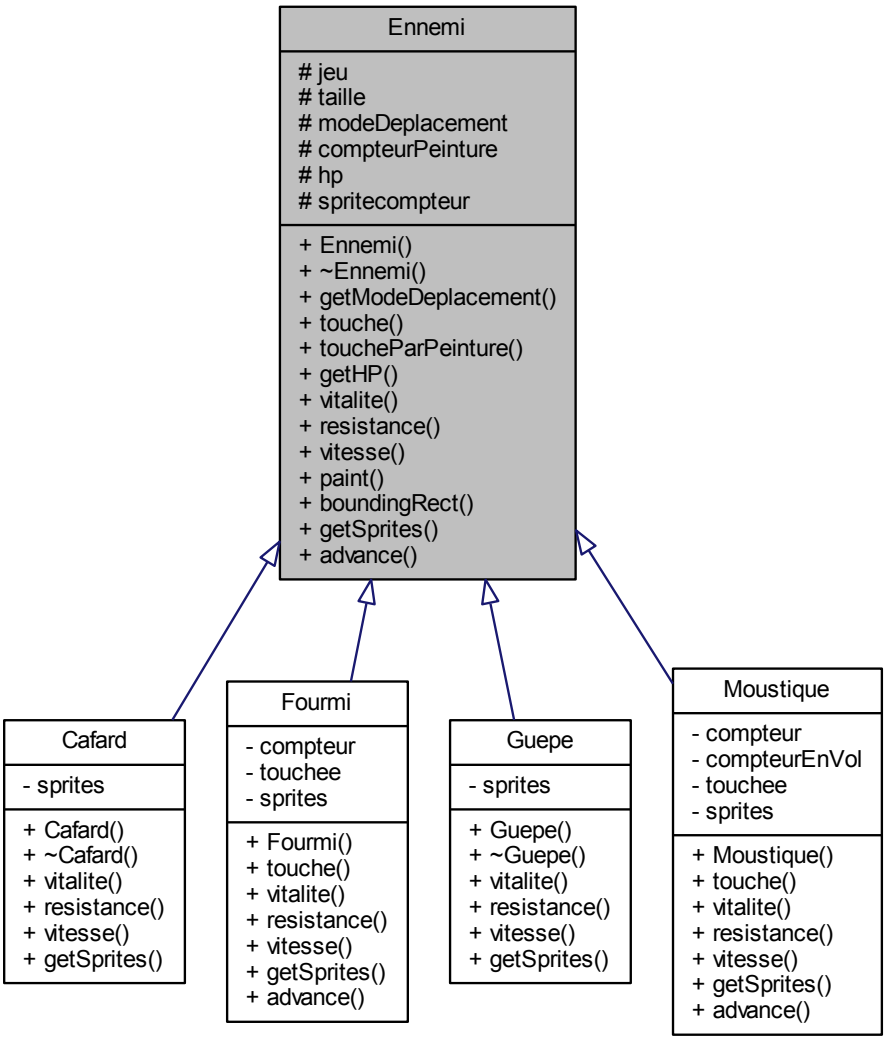
La documentation de cette classe a été générée à partir des fichiers suivants :
- defense.h
- eau.cpp

3.5 Référence de la classe Ennemi

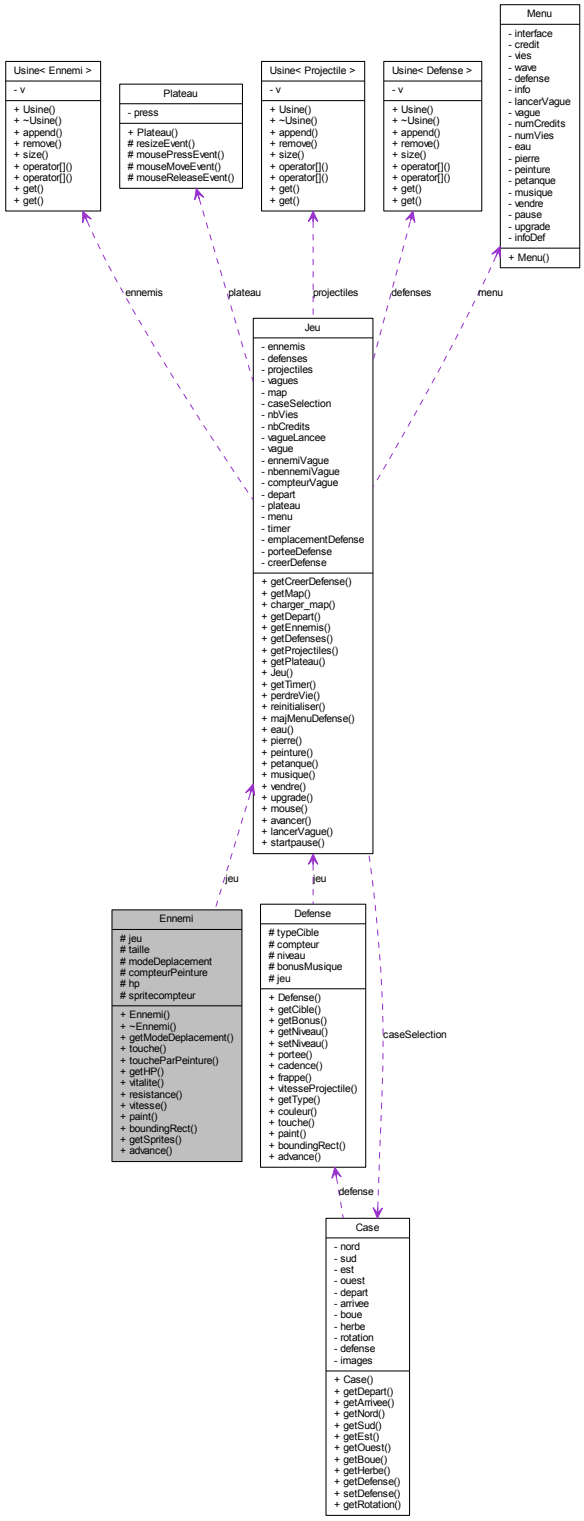
Objet graphique représentant un Ennemi abstrait.

```
#include <ennemi.h>
```

Graphe d'héritage de Ennemi :



Graphe de collaboration de Ennemi :



Connecteurs publics

- void advance (int phase)

  Déplacement de l'ennemi.

Signaux

- void arrivee ()

  Indique que l'ennemi est arrivée à la case de fin.

Fonctions membres publiques

- Ennemi (Jeu ∗jeu)
- virtual ∼Ennemi ()

  Destructeur polymorphe.
- QString getModeDeplacement ()

  Accesseur du mode de déplacement.
- virtual void touche (qreal attaque)
- virtual void toucheParPeinture (qreal frappe)
- qreal getHP ()
- virtual qreal vitalite ()=0
- virtual qreal resistance ()=0
- virtual qreal vitesse ()=0
- void paint (QPainter ∗painter, const QStyleOptionGraphicsItem ∗option, QWidget ∗widget)

  Affichage de l'ennemi.
- QRectF boundingRect () const
- virtual QList< QPixmap > & getSprites ()=0

Attributs protégés

- Jeu ∗ jeu

  Jeu dans lequel il est.
- qreal taille

  Taille.
- QString modeDeplacement

  Mode de déplacement.
- int compteurPeinture

  Compteur pour la durée de l'effet de ralentissement de la peinture.
- qreal hp

  Points de vie.
- int spritecompteur

  Compteur pour l'affichage des sprites.

3.5.1 Description détaillée

Objet graphique représentant un Ennemi abstrait.

3.5.2 Documentation des constructeurs et destructeur

3.5.2.1 Ennemi : :Ennemi (Jeu * *jeu*)

Construit un ennemi
Paramètres

jeu	Jeu dans lequel il est ajouté

3.5.3 Documentation des fonctions membres

3.5.3.1 QRectF Ennemi : :boundingRect ( ) const

Rectangle d'affichage
Renvoie

Rectangle contenant l'ennemi

3.5.3.2 qreal Ennemi : :getHP ( ) [inline]

Accesseur des points de vie
Renvoie

points de vie restant

3.5.3.3 virtual QList<QPixmap>& Ennemi : :getSprites ( ) [pure virtual]

Méthode virtuelle pure
Renvoie

Liste des sprites de l'ennemi

Implémenté dans Fourmi, Cafard, Moustique, et Guepe.

3.5.3.4 virtual qreal Ennemi : :resistance ( ) [pure virtual]

Méthode virtuelle pure
Renvoie

résistance de l'ennemi

Implémenté dans Fourmi, Cafard, Moustique, et Guepe.

3.5.3.5 void Ennemi : :touche (qreal *attaque*) [virtual]

L'Ennemi se fait toucher par le projectile d'une défense
Paramètres

attaque	Dégats du projectile

Réimplémentée dans Fourmi, et Moustique.

3.5.3.6 void Ennemi : :toucheParPeinture (qreal *frappe*) [virtual]

L'Ennemi se fait toucher par le projectile d'une défense qui le ralentit

Paramètres

frappe	Dégats du projectile

3.5.3.7 virtual qreal Ennemi : :vitalite ( ) [pure virtual]

Méthode virtuelle pure

Renvoie

vitalité de l'ennemi

Implémenté dans Fourmi, Cafard, Moustique, et Guepe.

3.5.3.8 virtual qreal Ennemi : :vitesse ( ) [pure virtual]

Méthode virtuelle pure

Renvoie

vitesse de l'ennemi en cases par seconde

Implémenté dans Fourmi, Cafard, Moustique, et Guepe.

La documentation de cette classe a été générée à partir des fichiers suivants :
– ennemi.h
– ennemi.cpp

3.6 Référence de la classe EnnemisVague

Sous-ensemble d'une vague permettant de générer plusieurs fois de mêmes ennemis.

`#include <vague.h>`

Fonctions membres publiques

– EnnemisVague ()
  Constructeur par défaut.
– EnnemisVague (QString T, float S, int N, int I)
– QString getType () const
  Type des ennemis.
– qreal getTaille () const
  Taille des ennemis.
– int getNb () const
  Nombre d'ennemis.
– int getDelai () const
  Nombre de frames entre deux ennemis.

3.6.1 Description détaillée

Sous-ensemble d'une vague permettant de générer plusieurs fois de mêmes ennemis.

3.6.2 Documentation des constructeurs et destructeur

3.6.2.1 EnnemisVague : :EnnemisVague (QString *T,* float *S,* int *N,* int *I*)

Constructeur complet

Paramètres

T	le type des ennemis
S	la taille des ennemis
N	le nombre d'ennemis
I	le delai entre deux ennemis

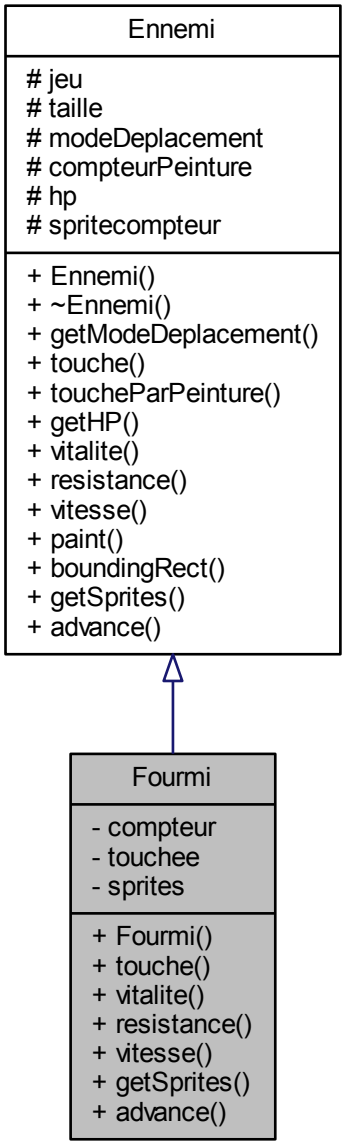
La documentation de cette classe a été générée à partir des fichiers suivants :
– vague.h
– vague.cpp

3.7 Référence de la classe Fourmi

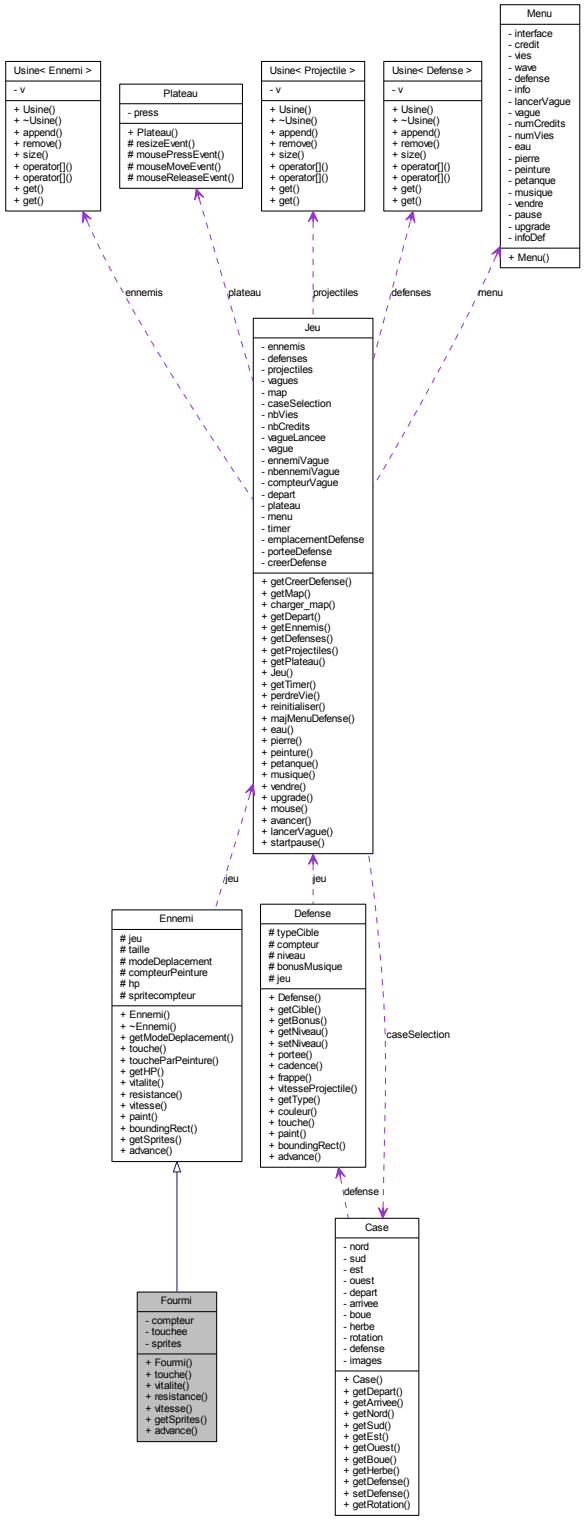
Insecte rampant de base.

```
#include <ennemi.h>
```

Graphe d'héritage de Fourmi :



Graphe de collaboration de Fourmi :



Connecteurs publics

– void advance (int phase)

Traitement spécifique au en fonction du temps.

Fonctions membres publiques

– Fourmi (Jeu *jeu, float taille)

Constructeur.

– void touche (qreal attaque)

Traitement spécifique quand les fourmis se font toucher.

– qreal vitalite ()

Définition de la vitalité

– qreal resistance ()

Définition de la résistance.

– qreal vitesse ()

Définition de la vitesse.

– QList< QPixmap > & getSprites ()

Définition des sprites.

3.7.1 Description détaillée

Insecte rampant de base.

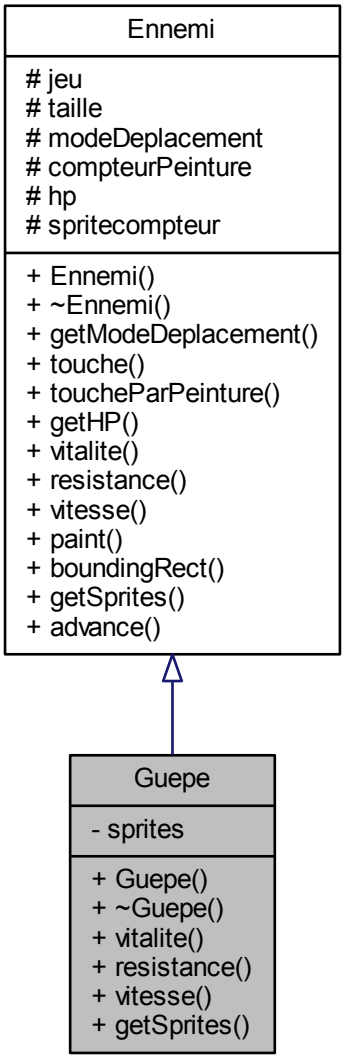
La documentation de cette classe a été générée à partir des fichiers suivants :
– ennemi.h
– fourmi.cpp

3.8 Référence de la classe Guepe

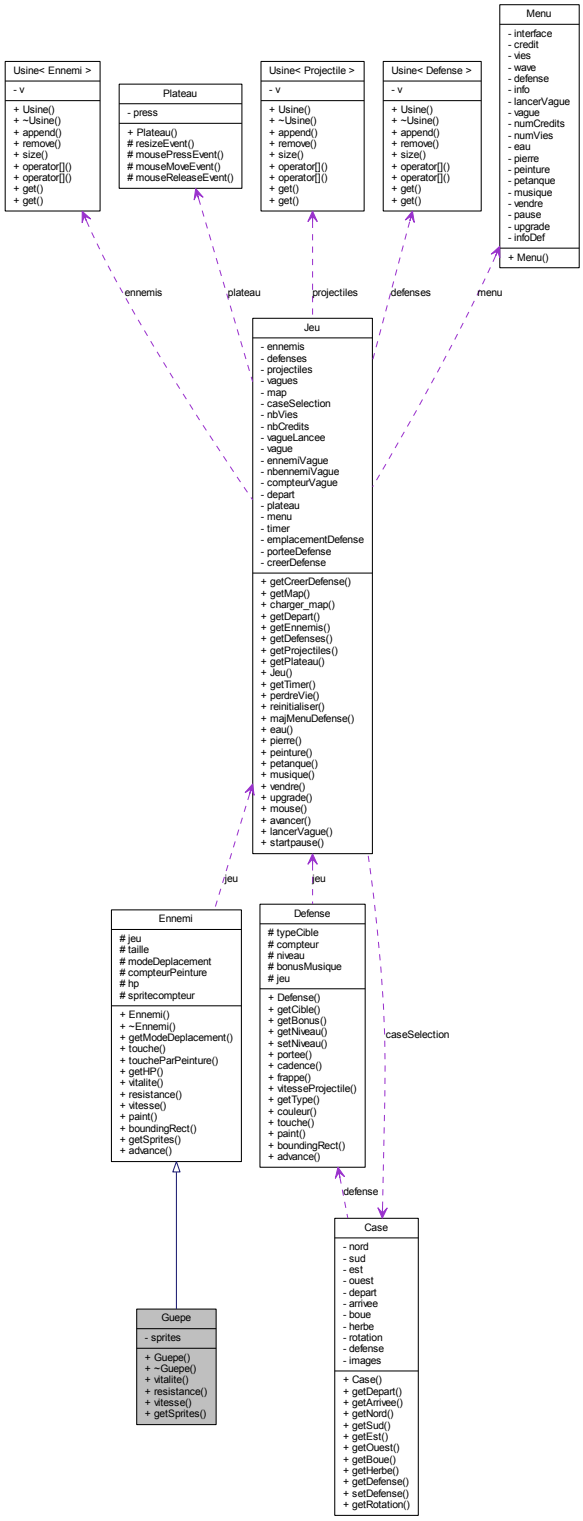
Insectes volant faisant des dégats de zone au sol à leur mort.

```
#include <ennemi.h>
```

Graphe d'héritage de Guepe :



Graphe de collaboration de Guepe :



Fonctions membres publiques

– Guepe (Jeu *jeu, float taille)

Constructeur.

– ~Guepe ()

Traitement spécifique quand les guêpes meurent.

– qreal vitalite ()

Définition de la vitalité

– qreal resistance ()

Définition de la résistance.

– qreal vitesse ()

Définition de la vitesse.

– QList< QPixmap > & getSprites ()

Définition des sprites.

3.8.1 Description détaillée

Insectes volant faisant des dégats de zone au sol à leur mort.

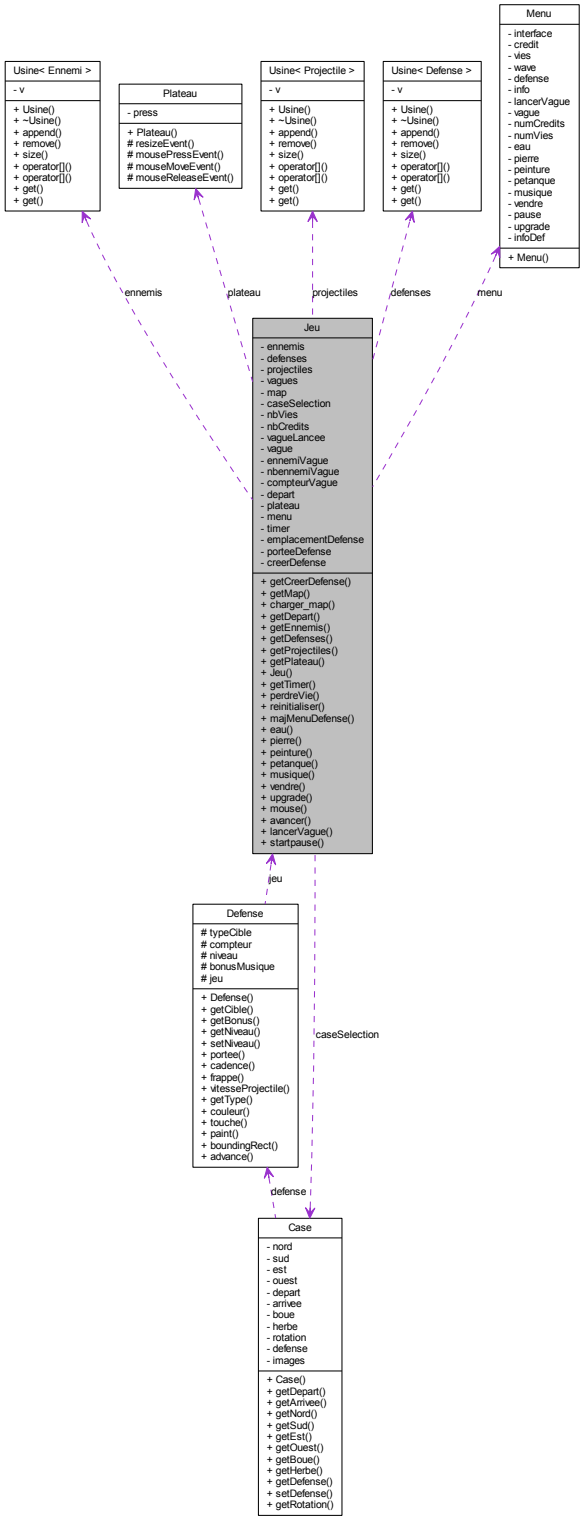
La documentation de cette classe a été générée à partir des fichiers suivants :
– ennemi.h
– guepe.cpp

3.9 Référence de la classe Jeu

Classe principale du Jeu.

```
#include <jeu.h>
```

Graphe de collaboration de Jeu :



Connecteurs publics

- void eau ()
  Clic sur le bouton d'ajout d'une défense de type Eau.
- void pierre ()
  Clic sur le bouton d'ajout d'une défense de type Pierre.
- void peinture ()
  Clic sur le bouton d'ajout d'une défense de type Peinture.
- void petanque ()
  Clic sur le bouton d'ajout d'une défense de type Pétanque.
- void musique ()
  Clic sur le bouton d'ajout d'une défense de type Musique.
- void vendre ()
  Clic sur le bouton de vente de défense.
- void upgrade ()
  Clic sur le bouton d'upgrade de défense.
- void mouse (QPointF)
  Prise en compte du mouvement de la souris sur le plateau de jeu.
- void avancer ()
  Rafraichissement du Jeu en fonction du Timer.
- void lancerVague ()
  Clic sur le bouton pour lancer la vague suivante.
- void startpause ()
  Clic sur le bouton de pause.

Fonctions membres publiques

- const QPoint & getCreerDefense () const
  Accesseur de la position pour créer une défense.
- QVector< QVector< Case * > > & getMap ()
  Accesseur de la matrice de la carte du jeu.
- void charger_map (QString map)
- const QPoint & getDepart () const
  Accesseur de la position de départ des ennemis.
- Usine< Ennemi > & getEnnemis ()
  Accesseur de la liste des ennemis présents.
- Usine< Defense > & getDefenses ()
  Accesseur de la liste des défenses présentes.
- Usine< Projectile > & getProjectiles ()
  Accesseur de la liste des projectiles présents.
- Plateau & getPlateau ()
  Accesseur du plateau de jeu.
- Jeu (QWidget *parent=0)
  Constructeur.
- QTimer & getTimer ()
  Accesseur du Timer principal.
- void perdreVie ()
  Fait perdre une vie au joueur.
- void reinitialiser ()
  Réinitialise le jeu.
- void majMenuDefense ()
  Met à jour les informations sur la défense selectionnée.

3.9.1 Description détaillée

Classe principale du Jeu.

3.9.2 Documentation des fonctions membres

3.9.2.1 void Jeu : :charger_map (QString *map*)

Charge le terrain du jeu

Paramètres

map	Emplacement du fichier map.txt

La documentation de cette classe a été générée à partir des fichiers suivants :
– jeu.h
– jeu.cpp

3.10 Référence de la classe Menu

Menu de jeu permettant au joueur d'intéragir et d'obtenir des informations.

`#include <menu.h>`

Fonctions membres publiques

– Menu ()
Constructeur par défaut.

Amis

– class Jeu

3.10.1 Description détaillée

Menu de jeu permettant au joueur d'intéragir et d'obtenir des informations.

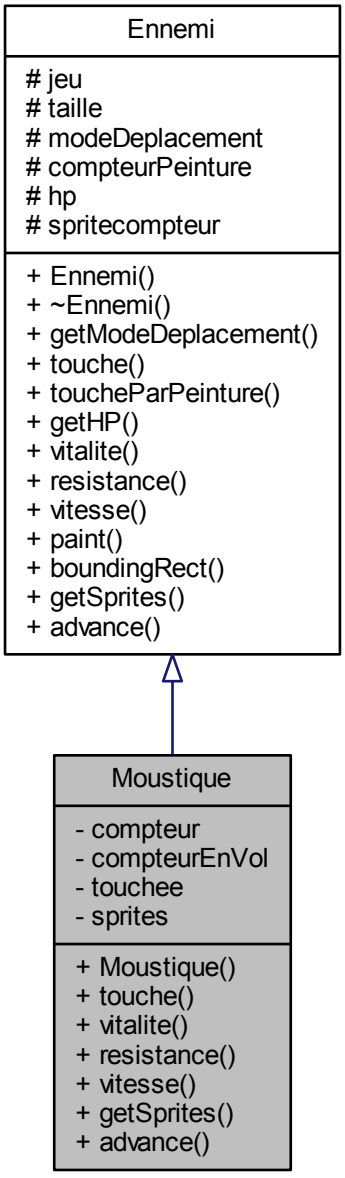
La documentation de cette classe a été générée à partir des fichiers suivants :
– menu.h
– menu.cpp

3.11 Référence de la classe Moustique

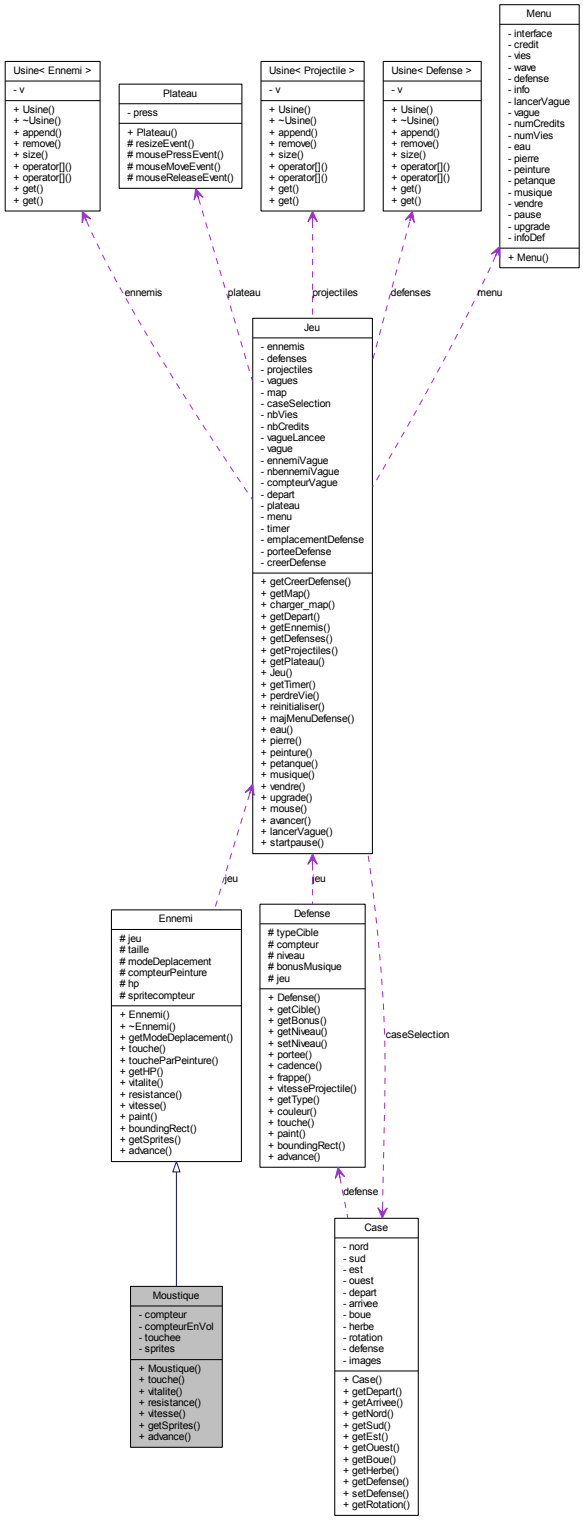
Insecte fragiles en vol pouvant se cacher et être très résistants au sol.

`#include <ennemi.h>`

Graphe d'héritage de Moustique :



Graphe de collaboration de Moustique :



Connecteurs publics

– void advance (int phase)

Traitement spécifique en fonction du temps.

Fonctions membres publiques

– Moustique (Jeu *jeu, float taille)

Constructeur.

– void touche (qreal attaque)

Traitement spécifique quand les moustiques se font toucher.

– qreal vitalite ()

Définition de la vitalité

– qreal resistance ()

Définition de la résistance.

– qreal vitesse ()

Définition de la vitesse.

– QList< QPixmap > & getSprites ()

Définition des sprites.

3.11.1 Description détaillée

Insecte fragiles en vol pouvant se cacher et être très résistants au sol.

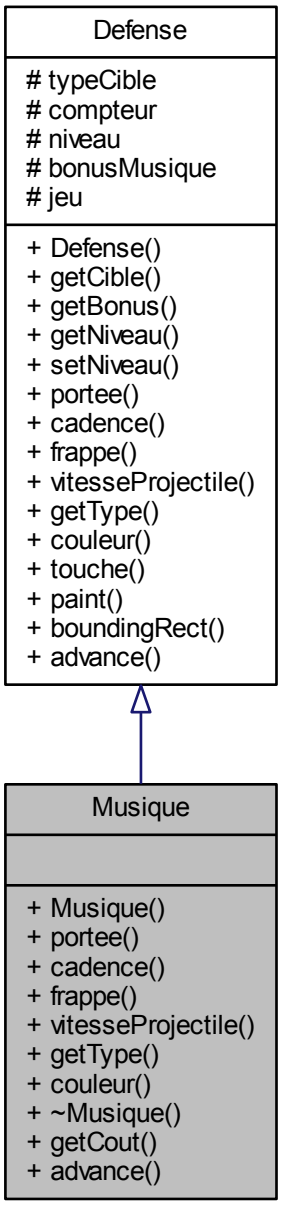
La documentation de cette classe a été générée à partir des fichiers suivants :
– ennemi.h
– moustique.cpp

3.12 Référence de la classe Musique

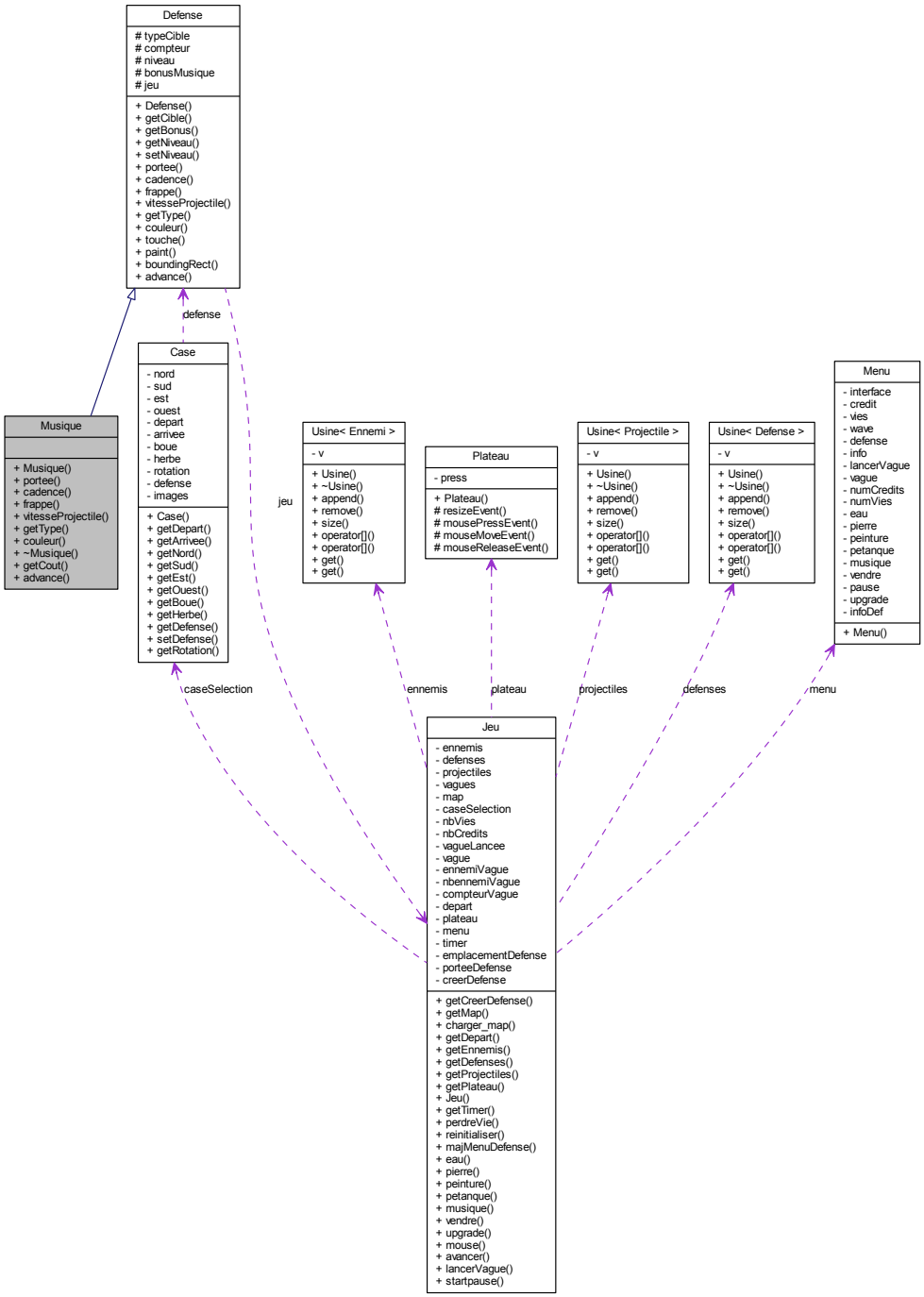
Défense permettant d'améliorer les défenses aux alentours.

```
#include <defense.h>
```

Graphe d'héritage de Musique :



Graphe de collaboration de Musique :



Connecteurs publics

– void advance (int phase)

Traitement spécifique pour chaque frame d'un musicien.

Fonctions membres publiques

– Musique (Jeu *jeu)

Constructeur.

– qreal portee ()

Définition de la portée.

– qreal cadence ()

Définition de la cadence de tir.

– qreal frappe ()

Définition de la puissance de frappe.

– qreal vitesseProjectile ()

Définition de la vitesse des projectiles.

– QString getType ()

Définition du type.

– const QColor & couleur () const

Définition de la couleur.

– ~Musique ()

Traitement spécifique lors de la vente d'un musicien.

Fonctions membres publiques statiques

– static int getCout (int niveau)

Définition du coût.

3.12.1 Description détaillée

Défense permettant d'améliorer les défenses aux alentours.

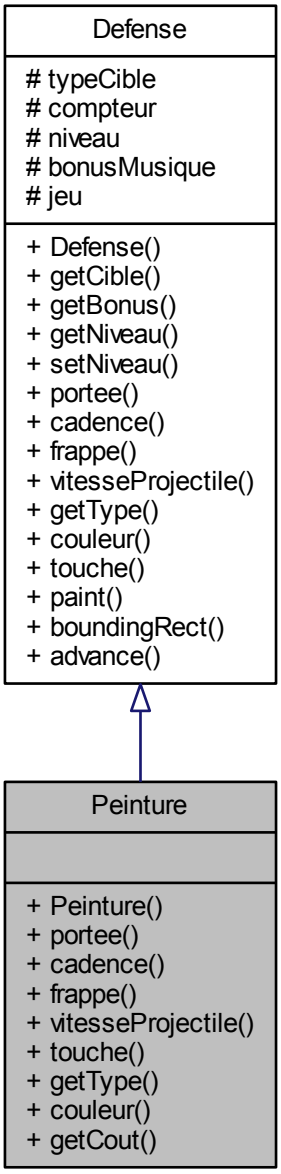
La documentation de cette classe a été générée à partir des fichiers suivants :
– defense.h
– musique.cpp

3.13 Référence de la classe Peinture

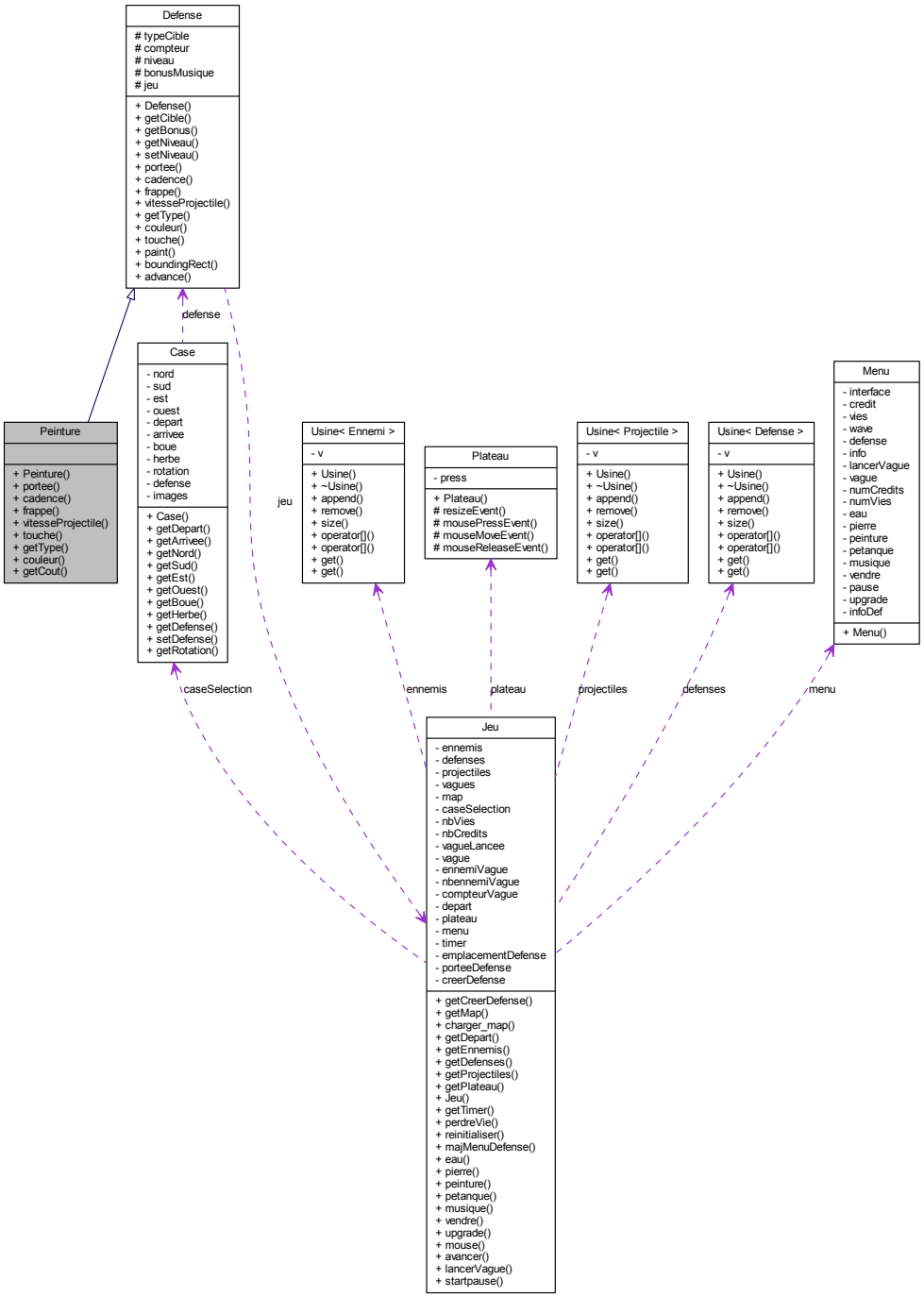
Défense permettant de ralentir les ennemis touchés.

```
#include <defense.h>
```

Graphe d'héritage de Peinture :



Graphe de collaboration de Peinture :



Fonctions membres publiques

- Peinture (Jeu *jeu)

  Constructeur.
- qreal portee ()

  Définition de la portée.
- qreal cadence ()

  Définition de la cadence de tir.
- qreal frappe ()

  Définition de la puissance de frappe.
- qreal vitesseProjectile ()

  Définition de la vitesse des projectiles.
- void touche (Ennemi *cible)

  Traitement spécifique quand le projectile atteint la cible.
- QString getType ()

  Définition du type.
- const QColor & couleur () const

  Définition de la couleur.

Fonctions membres publiques statiques

- static int getCout (int niveau)

  Définition du coût.

3.13.1 Description détaillée

Défense permettant de ralentir les ennemis touchés.

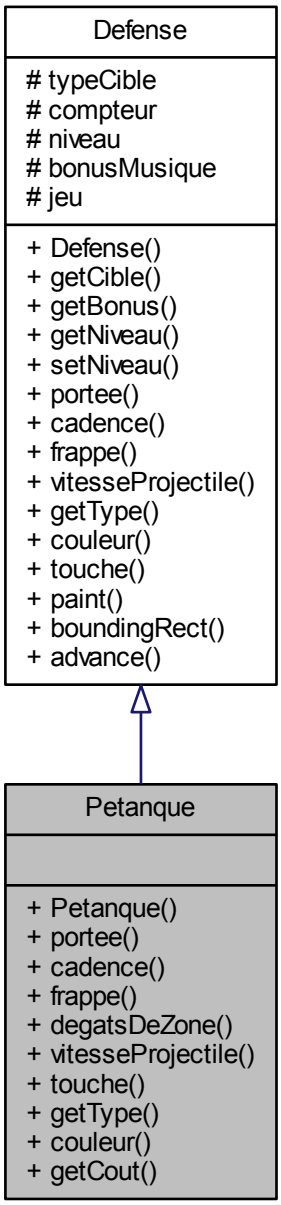
La documentation de cette classe a été générée à partir des fichiers suivants :
- defense.h
- peinture.cpp

3.14 Référence de la classe Petanque

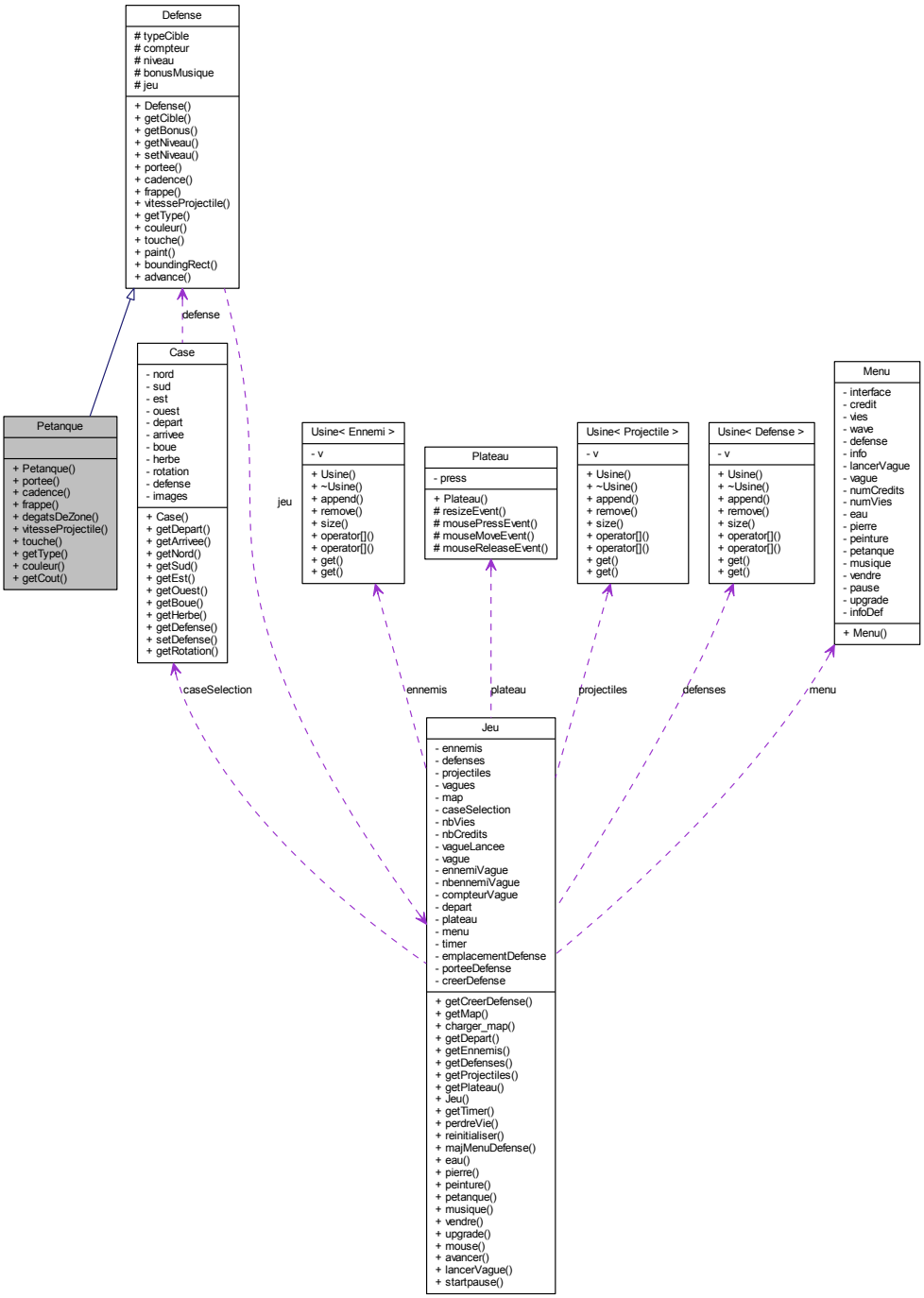
Défense lente mais faisant des dégats de zone.

```
#include <defense.h>
```

Graphe d'héritage de Petanque :



Graphe de collaboration de Petanque :



Fonctions membres publiques

- Petanque (Jeu *jeu)

  Constructeur.
- qreal portee ()

  Définition de la portée.
- qreal cadence ()

  Définition de la cadence de tir.
- qreal frappe ()

  Définition de la puissance de frappe.
- qreal degatsDeZone ()

  Définition de la puissance des dégats de zone.
- qreal vitesseProjectile ()

  Définition de la vitesse des projectiles.
- void touche (Ennemi *cible)

  Traitement spécifique quand le projectile atteint la cible.
- QString getType ()

  Définition du type.
- const QColor & couleur () const

  Définition de la couleur.

Fonctions membres publiques statiques

- static int getCout (int niveau)

  Définition du coût.

3.14.1 Description détaillée

Défense lente mais faisant des dégats de zone.

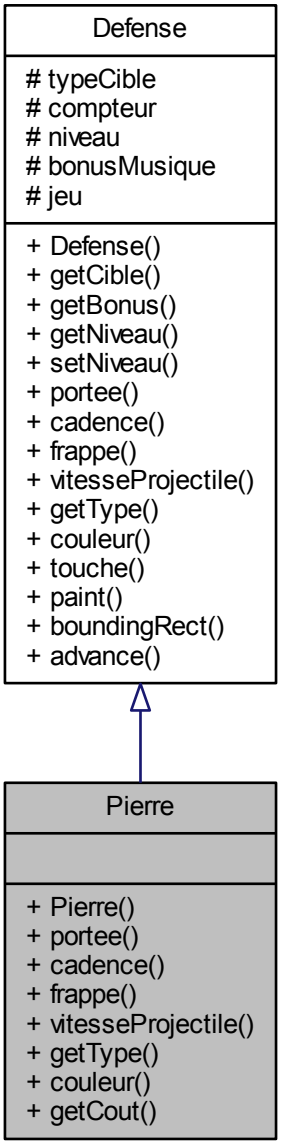
La documentation de cette classe a été générée à partir des fichiers suivants :
- defense.h
- petanque.cpp

3.15 Référence de la classe Pierre

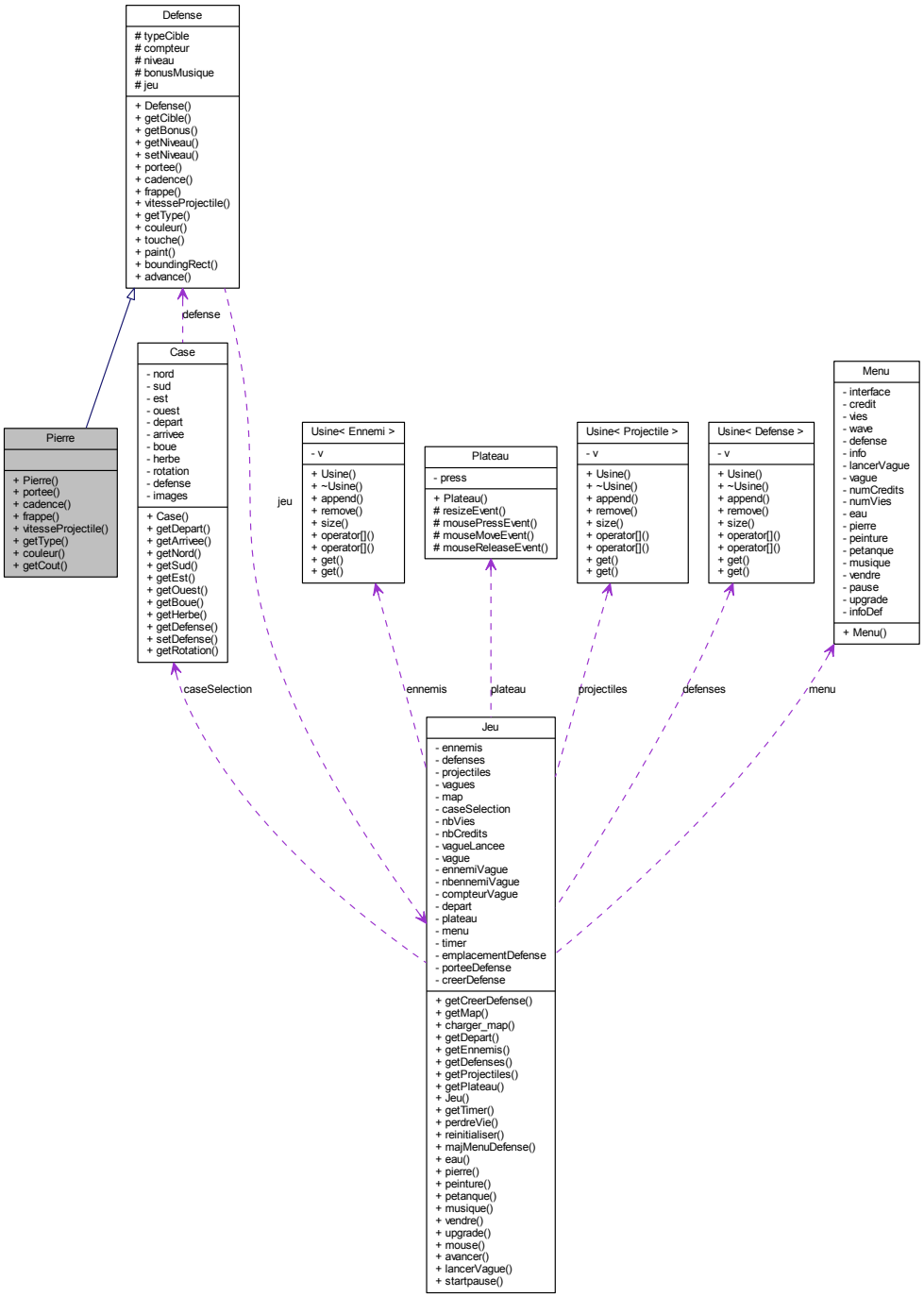
Défense éfficace contre les cibles volantes.

```
#include <defense.h>
```

Graphe d'héritage de Pierre :



Graphe de collaboration de Pierre :



Fonctions membres publiques

– Pierre (Jeu *jeu)
  Constructeur.
– qreal portee ()
  Définition de la portée.
– qreal cadence ()
  Définition de la cadence de tir.
– qreal frappe ()
  Définition de la puissance de frappe.
– qreal vitesseProjectile ()
  Définition de la vitesse des projectiles.
– QString getType ()
  Définition du type.
– const QColor & couleur () const
  Définition de la couleur.

Fonctions membres publiques statiques

– static int getCout (int niveau)
  Définition du coût.

3.15.1 Description détaillée

Défense éfficace contre les cibles volantes.

La documentation de cette classe a été générée à partir des fichiers suivants :
– defense.h
– pierre.cpp

3.16 Référence de la classe Plateau

Plateau de Jeu contenant les éléments graphiques animés.

`#include <plateau.h>`

Signaux

– void mouse (QPointF)
  Signal indiquant le mouvement de la souris.

Fonctions membres publiques

– Plateau (QWidget *parent=0)
  Constructeur.

Fonctions membres protégées

– virtual void resizeEvent (QResizeEvent ∗event)

Redimensionnement automatique du plateau de jeu en fonction de la fenêtre.

– virtual void mousePressEvent (QMouseEvent ∗event)

Détecte l'appui du bouton de la souris.

– virtual void mouseMoveEvent (QMouseEvent ∗event)

Détecte le mouvement de la souris.

– virtual void mouseReleaseEvent (QMouseEvent ∗event)

Détecte le relachement du bouton de la souris.

3.16.1 Description détaillée

Plateau de Jeu contenant les éléments graphiques animés.

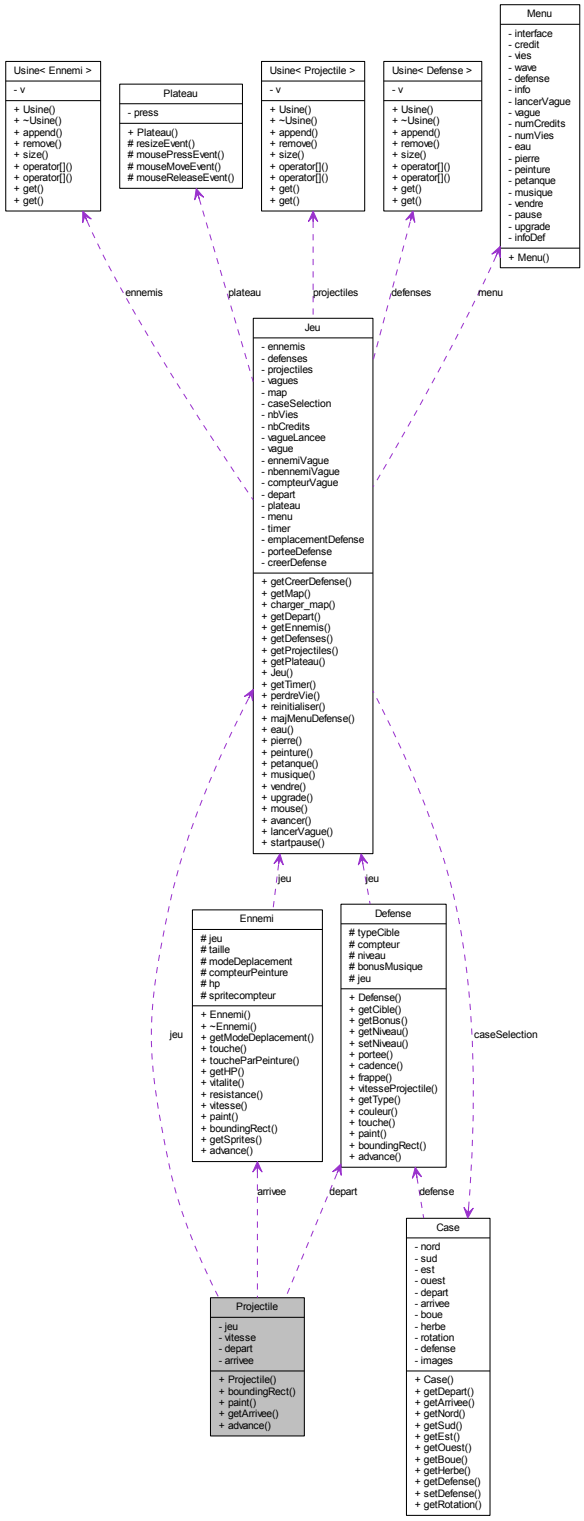
La documentation de cette classe a été générée à partir des fichiers suivants :
– plateau.h
– plateau.cpp

3.17 Référence de la classe Projectile

Objet graphique représentant un projectile allant d'une défense à un ennemi.

```
#include <projectile.h>
```

Graphe de collaboration de Projectile :



Connecteurs publics

– virtual void advance (int phase)

Déplacement du projectile.

Fonctions membres publiques

– Projectile (Jeu *jeu, Defense *def, Ennemi *ennemi)
– QRectF boundingRect () const

Taille du projectile.

– void paint (QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget)

Affichage du projectile.

– const Ennemi * getArrivee () const

Accesseur constant de l'ennemi visé

3.17.1 Description détaillée

Objet graphique représentant un projectile allant d'une défense à un ennemi.

3.17.2 Documentation des constructeurs et destructeur

3.17.2.1 Projectile : :Projectile (Jeu * *jeu,* Defense * *def,* Ennemi * *ennemi*)

Construit un projectile

Paramètres

jeu	Jeu dans lequel il est ajouté
def	Défense qui lance le projectile
ennemi	Ennemi ciblé par la défense

La documentation de cette classe a été générée à partir des fichiers suivants :
– projectile.h
– projectile.cpp

3.18 Référence de la classe Usine< T > (modèle)

Patron de classe stoquant un vecteur de pointeurs avec libération automatique de la mémoire.

`#include <usine.h>`

Fonctions membres publiques

– Usine ()

Constructeur créant un vecteur vide.

– ~Usine ()

Destructeur libérant la mémoire de tous les pointeurs.

- void append (T *x)
- void remove (int i)
- int size ()
- T * operator[] (int i)

  Acceusseur d'un élément du vecteur.
- const T * operator[] (int i) const

  Acesseur constant d'un élément du vecteur.
- QVector< T * > & get ()

  Accesseur du vecteur.
- const QVector< T * > & get () const

  Accesseur constant du vecteur.

3.18.1 Description détaillée

template<class T>class Usine< T >

Patron de classe stoquant un vecteur de pointeurs avec libération automatique de la mémoire.

3.18.2 Documentation des fonctions membres

3.18.2.1 template<class T> void Usine< T > : :append (T * *x*) [inline]

Ajoute un nouvel élément au vecteur

Paramètres

x	élément à ajouter

3.18.2.2 template<class T> void Usine< T > : :remove (int *i*) [inline]

Libère la mémoire d'un élément et l'enlève du vecteur

Paramètres

i	indice de l'élément

3.18.2.3 template<class T> int Usine< T > : :size ( ) [inline]

Récupère la taille du vecteur

Renvoie

taille du vecteur

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

- usine.h

3.19 Référence de la classe Vague

Nom et ennemis d'une vague, correspondant à une ligne du fichiers waves.txt.

`#include <vague.h>`

Fonctions membres publiques

- Vague ()
  Constructeur par défaut.
- const QVector< EnnemisVague > & getEnnemis () const
  Vecteur const des ennemis de la vague.
- QVector< EnnemisVague > & getEnnemis ()
  Vecteur des ennemis de la vague.
- QString getNom () const
  Nom de la vague.
- Vague (QString nom)

Fonctions membres publiques statiques

- static void charger_vagues (QString fichier, QVector< Vague > &vagues)

3.19.1 Description détaillée

Nom et ennemis d'une vague, correspondant à une ligne du fichiers waves.txt.

3.19.2 Documentation des constructeurs et destructeur

3.19.2.1 Vague : :Vague (QString *nom*)

Construit une vague

Paramètres

nom	Nom de la vague

3.19.3 Documentation des fonctions membres

3.19.3.1 void Vague : :charger_vagues (QString *fichier,* QVector< **Vague** > & *vagues*) [static]

Charge toutes les vagues d'un fichier donné dans un vecteur de vagues

Paramètres

fichier	Chemin du fichier waves.txt
vagues	Référence vers un vecteur de vague où enregistrer les vagues

La documentation de cette classe a été générée à partir des fichiers suivants :

– vague.h
– vague.cpp