YATD

0.9

Généré par Doxygen 1.7.4

Wed Jun 8 2011 23 :37 :49

Table des matières

Chapitre 1

Index des structures de données

1.1 Hiérarchie des classes

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

Chapitre 2

Index des structures de données

2.1 Structures de données

Liste des structures de données avec une brève description :

Chapitre 3

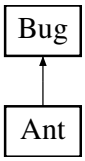
Documentation des structures de données

3.1 Référence de la classe Ant

Classe représentant les fourmis.

`#include <Ant.h>`

Graphe d'héritage de Ant :



Signaux

- void dead (Bug ∗bug)
  Signal de mort de l'insecte.
- void goalReached (Bug ∗bug)
  Signal d'arrivé.

Fonctions membres publiques

- Ant (double x, double y, double s, double start_angle)
  Constructeur de Ant.
- ∼Ant ()
  Destructeur de Ant.
- void paint (QPainter ∗painter, const QStyleOptionGraphicsItem ∗option, QWidget ∗widget)

  Surcharge de la méthode virtuelle pure paint() héritée de QGraphicsObject.
- void hit (double dmg)

Infliger des dégats à la fourmi et activer sa capacité spéciale.

- QRectF boundingRect () const
  Surcharge de la fonction boudingRect() héritée de QGraphicsObject.
- QPainterPath shape () const
  Surcharge de la méthode virtuelle pure shape() héritée de QGraphicsObject.
- short int getMoveType ()
  Permet d'accéder à la valeur moveType de l'insecte.

Champs de données

- Render ∗ parent
  L'objet Render parent de l'insecte.

Fonctions membres protégées

- void advance (int step)
  Implémentation de la fonction virtuelle pure advance() héritée de QGGaphicsObject.

Attributs protégés

- double size
  Taille.
- int frame
  Compteur de frame.
- double speed
  Vitesse de base.

Connecteurs privés

- void slowDown ()
  Slot appelé pour ralentir la fourmi Après 5 secondes, le timer émet un signal timeout qui appelle la fonction slowDown() de la fourmi pour reprendre sa vitesse initiale ?

Attributs privés

- QImage ∗ image [3]
  Cache d'images.
- QTimer ∗ timer
  Timer de ralentissement.

3.1.1 Description détaillée

Définie les caractéristiques spécifiques aux fourmis, comme leur graphisme, leur type de déplacement (CRAWL)...

3.1.2 Documentation des constructeurs et destructeur

3.1.2.1 Ant : :Ant (double *x*, double *y*, double *s*, double *start_angle*)

Ce constructeur initialise les images en cache de la fourmi et appelle le constructeur parent (Bug) avec les bons paramètres correspondant aux caractéristique d'une fourmi.

Paramètres

x	Un double correspondant à l'abscisse du point de départ des insectes.
y	Un double correspondant à l'ordonnée du point de départ des insectes.
s	La taille de la fourmi, influe sur la taille de la représentation graphique et sur ses autres caractéristiques.
start_angle	L'angle vers lequel fait face le départ des insectes.

3.1.2.2 Ant : :~Ant ( )

Désalloue la mémoire des images de la fourmi en cache.

3.1.3 Documentation des fonctions membres

3.1.3.1 void Bug : :advance (int *step*) [protected, inherited]

Implémentation de la fonction virtuelle pure advance() héritée de QGGaphicsObject.

Paramètres

step	La fonction est appelé deux fois automatiquement par QT à chaque update(), une fois avec 0 comme paramètre puis une fois avec 1.

3.1.3.2 QRectF Bug : :boundingRect ( ) const [inherited]

Fonction appelé automatiquement par QT pour savoir s'il doit ou non réafficher l'insecte.

Renvoie

Un objet QRectF correspondant au rectangle englobant l'ensemble du dessin de l'insecte.

3.1.3.3 void Bug : :dead (Bug * *bug*) [signal, inherited]

Signal émit lorsque l'insecte est tué.

Paramètres

bug	Un pointeur vers l'insecte qui vient de mourir.

3.1.3.4 short int Bug : :getMoveType ( ) [inherited]

Renvoie

la valeur prédéfinie FLY ou CRAWL.

3.1.3.5 void Bug : :goalReached (Bug ∗ *bug*) [signal, inherited]

Signal émit par l'insecte lorsqu'il atteint son but.

Paramètres

bug	Un pointeur vers l'insecte.

3.1.3.6 void Ant : :hit (double *dmg*)

Appelle la méthode Bug : :hit() pour infliger les dégats, et lance le timer pour accélérer la fourmi pendant 5 secondes.

Paramètres

dmg	Un double correspondant au montant de dégats à infliger avec réduction.

Réimplémentée à partir de Bug.

3.1.3.7 void Ant : :paint (QPainter ∗ *painter,* const QStyleOptionGraphicsItem ∗ *option,* QWidget ∗ *widget*)

Est appelé automatiquement par Qt pour redessiner la fourmi.

3.1.3.8 QPainterPath Bug : :shape ( ) const [inherited]

Fonction utilisé par QT pour traiter les collisions entre objets graphiques.

Renvoie

Un object QPainterPath correspondant au contour de collision de l'insecte.

3.1.4 Documentation des champs

3.1.4.1 int Bug : :frame [protected, inherited]

Compteur d'image utilisé pour afficher successivement chaque image des animations.

3.1.4.2 QImage∗ Ant : :image[3] [private]

Les images de la fourmis à chaque position, redimensionnées en fonction de sa taille et mises en cache pour un affichage plus rapide.

3.1.4.3 Render∗ Bug : :parent [inherited]

Quand on ajoute un insecte à l'objet Render par la méthode addBug(), cet attribut est automatiquement initialisé.

3.1.4.4 double Bug : :size [protected, inherited]

La taille de l'insecte, influe à la fois sur la taille de la représentation graphique et sur les caractéristiques de l'insecte.'

3.1.4.5 double Bug : :speed [protected, inherited]

La vitesse en case/seconde à laquelle se déplace l'insecte.

3.1.4.6 QTimer∗ Ant : :timer [private]

Quand une fourmis est blessée, elle accélère pendant 5 secondes, ce timer permet de faire ce décompte.

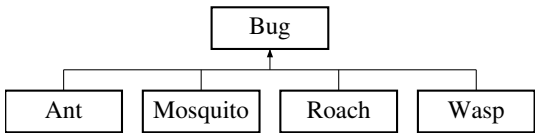
La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Ant.h
– src/Ant.cpp

3.2 Référence de la classe Bug

Classe abstraite dont héritent les ennemis.

`#include <Bug.h>`

Graphe d'héritage de Bug :



Signaux

– void dead (Bug ∗bug)
Signal de mort de l'insecte.
– void goalReached (Bug ∗bug)
Signal d'arrivé.

Fonctions membres publiques

– Bug (double x, double y, double s, double health, double res, double start_angle, double init_speed, short int type)
Constructeur de Bug.
– QRectF boundingRect () const
Surcharge de la fonction boudingRect() héritée de QGraphicsObject.
– QPainterPath shape () const

Surcharge de la méthode virtuelle pure shape() héritée de QGraphicsObject.

– void hit (double dmg)
Infliger des dégats à l'insecte.
– short int getMoveType ()
Permet d'accéder à la valeur moveType de l'insecte.

Champs de données

– Render ∗ parent
L'objet Render parent de l'insecte.

Fonctions membres protégées

– void advance (int step)
Implémentation de la fonction virtuelle pure advance() héritée de QGGaphicsObject.

Attributs protégés

– double size
Taille.
– int frame
Compteur de frame.
– double speed
Vitesse de base.

Attributs privés

– double hp
Point de vie.
– double resist
Résistance.
– double angle
Angle des déplacements.
– short int moveType
Type de déplacement.
– QPoint lastSquare
Case d'origine.

3.2.1 Description détaillée

Cette classe définie les actions de base commune à tous les insectes (l'affichage, la vie et les déplacements entre autres).

3.2.2 Documentation des constructeurs et destructeur

3.2.2.1 Bug : :Bug (double *x*, double *y*, double *s*, double *health*, double *res*, double *start_angle*, double *init_speed*, short int *type*)

La classe Bug étant abstraite, le constructeur sera toujours appelé par ses classes filles (Wasp, Ant, Mosquito, Roach) avec les bons paramètres.

Paramètres

x	Un double correspondant à l'abscisse du point de départ des insectes.
y	Un double correspondant à l'ordonnée du point de départ des insectes.
s	La taille de l'insecte à créer.
health	Un double donnant la vitalité initiale de l'insecte.
res	Un double donnant la résistance aux dégats de l'insecte.
start_angle	L'angle vers lequel fait face le départ des insectes.
init_speed	La vitesse initiale de l'insecte.
type	Prends une des valeurs prédéfinies FLY ou CRAWL correspondant au type de déplacement de l'insecte.

3.2.3 Documentation des fonctions membres

3.2.3.1 void Bug : :advance (int *step*) [protected]

Implémentation de la fonction virtuelle pure advance() héritée de QGGaphicsObject.

Paramètres

step	La fonction est appelé deux fois automatiquement par QT à chaque update(), une fois avec 0 comme paramètre puis une fois avec 1.

3.2.3.2 QRectF Bug : :boundingRect ( ) const

Fonction appelé automatiquement par QT pour savoir s'il doit ou non réafficher l'insecte.

Renvoie

Un objet QRectF correspondant au rectangle englobant l'ensemble du dessin de l'insecte.

3.2.3.3 void Bug : :dead (Bug * *bug*) [signal]

Signal émit lorsque l'insecte est tué.

Paramètres

bug	Un pointeur vers l'insecte qui vient de mourir.

3.2.3.4 short int Bug : :getMoveType ( )

Renvoie

la valeur prédéfinie FLY ou CRAWL.

3.2.3.5 void Bug : :goalReached (Bug * *bug*) [signal]

Signal émit par l'insecte lorsqu'il atteint son but.

Paramètres

bug	Un pointeur vers l'insecte.

3.2.3.6 void Bug : :hit (double *dmg*)

Fonction pouvant être appelée pour infliger des dégats à l'insecte (par exemple par les projectiles des tours).

Paramètres

dmg	Un double correspondant au point de dégat à infliger (avant réduction par la résistance de l'insecte).

Réimplémentée dans Ant.

3.2.3.7 QPainterPath Bug : :shape ( ) const

Fonction utilisé par QT pour traiter les collisions entre objets graphiques.

Renvoie

Un object QPainterPath correspondant au contour de collision de l'insecte.

3.2.4 Documentation des champs

3.2.4.1 double Bug : :angle [private]

Angle dans lequel l'insecte se déplace.

3.2.4.2 int Bug : :frame [protected]

Compteur d'image utilisé pour afficher successivement chaque image des animations.

3.2.4.3 double Bug : :hp [private]

La vitalité de l'insecte, quand elle atteint 0, l'insecte meurt et le joueur gagne 1 crédit.

3.2.4.4 QPoint Bug : :lastSquare [private]

Dernière case visité par l'insecte. Elle est utilisé dans la détection de changement de case.

3.2.4.5 short int Bug : :moveType [private]

Peut prendre les valeurs prédéfinies FLY ou CRAWL.

3.2.4.6 Render∗ Bug : :parent

Quand on ajoute un insecte à l'objet Render par la méthode addBug(), cet attribut est automatiquement initialisé.

3.2.4.7 double Bug : :resist [private]

La résistance aux dégats de l'insecte.

3.2.4.8 double Bug : :size [protected]

La taille de l'insecte, influe à la fois sur la taille de la représentation graphique et sur les caractéristiques de l'insecte.'

3.2.4.9 double Bug : :speed [protected]

La vitesse en case/seconde à laquelle se déplace l'insecte.

La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Bug.h
– src/Bug.cpp
– src/moc_Bug.cpp

3.3 Référence de la classe Hatchery

Classe implémentation du design pattern Factory.

`#include <Hatchery.h>`

Fonctions membres publiques

– Hatchery (double start_x, double start_y, double start_angle)
  Constructeur d'Hatchery.
– Bug ∗ spawnBug (QString race, double size)
  Crée un nouvel insecte de la race et de la taille données.

Attributs privés

– double x
  Abscisse du spawn des insectes.
– double y
  Ordonnée du spawn des insectes.

– double angle
Angle vers lequel fait face le spawn des insectes.

3.3.1 Description détaillée

Design pattern Factory utilisé pour créer un nouvel insecte au niveau du spawn.

3.3.2 Documentation des constructeurs et destructeur

3.3.2.1 Hatchery : :Hatchery (double *start_x,* double *start_y,* double *start_angle*)

Permet d'instancier un objet Hatchery pour initialiser la position et l'angle du spawn des insectes.

Paramètres

start_x	Un double correspondant à l'abscisse du point de départ des insectes.
start_y	Un double correspondant à l'ordonnée du point de départ des insectes.
start_angle	Un double correspondant à l'angle de départ des insectes (en degré).

3.3.3 Documentation des fonctions membres

3.3.3.1 Bug * Hatchery : :spawnBug (QString *race,* double *size*)

Paramètres

race	Un QString donnant la race de l'insecte a créer.
size	Un double donnant la taille de l'insecte.

Renvoie

Un pointeur vers l'insecte créé.

3.3.4 Documentation des champs

3.3.4.1 double Hatchery : :angle [private]

3.3.4.2 double Hatchery : :x [private]

3.3.4.3 double Hatchery : :y [private]

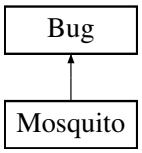
La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Hatchery.h
– src/Hatchery.cpp

3.4 Référence de la classe Mosquito

Classe représentant les moustiques.

`#include <Mosquito.h>`

Graphe d'héritage de Mosquito :



Signaux

- void dead (Bug *bug)
  Signal de mort de l'insecte.
- void goalReached (Bug *bug)
  Signal d'arrivé.

Fonctions membres publiques

- Mosquito (double x, double y, double s, double start_angle)
  Constructeur de Mosquito.
- ~Mosquito ()
  Destructeur de Mosquito.
- void paint (QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget)

  Surcharge de la fonction virtuelle pure paint() hérité de QGraphicsObject.
- QRectF boundingRect () const
  Surcharge de la fonction boudingRect() héritée de QGraphicsObject.
- QPainterPath shape () const
  Surcharge de la méthode virtuelle pure shape() héritée de QGraphicsObject.
- void hit (double dmg)
  Infliger des dégats à l'insecte.
- short int getMoveType ()
  Permet d'accéder à la valeur moveType de l'insecte.

Champs de données

- Render * parent
  L'objet Render parent de l'insecte.

Fonctions membres protégées

- void advance (int step)
  Implémentation de la fonction virtuelle pure advance() héritée de QGGaphicsObject.

Attributs protégés

– double size
  Taille.
– int frame
  Compteur de frame.
– double speed
  Vitesse de base.

Attributs privés

– QImage ∗ image [2]
  Cache d'image.

3.4.1 Description détaillée

Définie les caractéristiques spécifiques aux moustiques, comme leur graphisme, leur type de déplacement (FLY)...

3.4.2 Documentation des constructeurs et destructeur

3.4.2.1 Mosquito : :Mosquito (double *x,* double *y,* double *s,* double *start_angle*)

Ce constructeur charge les images du moustique et appelle le constructeur parent de Bug() avec les paramètres correspondants aux caractéristiques d'un moustique.

Paramètres

x	Un double correspondant à l'abscisse du point de départ des insectes.
y	Un double correspondant à l'ordonnée du point de départ des insectes.
s	La taille du moustique, influe sur la taille de sa représentation graphique et sur les autres caractéristiques.
start_angle	Angle de départ de l'insecte.

3.4.2.2 Mosquito : :∼Mosquito ( )

Désalloue la mémoire des images du moustique mises en cache.

3.4.3 Documentation des fonctions membres

3.4.3.1 void Bug : :advance (int *step*) [protected, inherited]

Implémentation de la fonction virtuelle pure advance() héritée de QGGaphicsObject.

Paramètres

step	La fonction est appelé deux fois automatiquement par QT à chaque update(), une fois avec 0 comme paramètre puis une fois avec 1.

3.4.3.2 QRectF Bug : :boundingRect ( ) const [inherited]

Fonction appelé automatiquement par QT pour savoir s'il doit ou non réafficher l'insecte.

Renvoie

Un objet QRectF correspondant au rectangle englobant l'ensemble du dessin de l'insecte.

3.4.3.3 void Bug : :dead (Bug ∗ *bug*) [signal, inherited]

Signal émit lorsque l'insecte est tué.

Paramètres

bug	Un pointeur vers l'insecte qui vient de mourir.

3.4.3.4 short int Bug : :getMoveType ( ) [inherited]

Renvoie

la valeur prédéfinie FLY ou CRAWL.

3.4.3.5 void Bug : :goalReached (Bug ∗ *bug*) [signal, inherited]

Signal émit par l'insecte lorsqu'il atteint son but.

Paramètres

bug	Un pointeur vers l'insecte.

3.4.3.6 void Bug : :hit (double *dmg*) [inherited]

Fonction pouvant être appelée pour infliger des dégats à l'insecte (par exemple par les projectiles des tours).

Paramètres

dmg	Un double correspondant au point de dégat à infliger (avant réduction par la résistance de l'insecte).

Réimplémentée dans Ant.

3.4.3.7 void Mosquito : :paint (QPainter ∗ *painter,* const QStyleOptionGraphicsItem ∗ *option,* QWidget ∗ *widget*)

Est appelé automatiquement par Qt pour redessiner le moustique.

3.4.3.8 QPainterPath Bug : :shape ( ) const [inherited]

Fonction utilisé par QT pour traiter les collisions entre objets graphiques.

Renvoie

Un object QPainterPath correspondant au contour de collision de l'insecte.

3.4.4 Documentation des champs

3.4.4.1 int Bug : :frame [protected, inherited]

Compteur d'image utilisé pour afficher successivement chaque image des animations.

3.4.4.2 QImage* Mosquito : :image[2] [private]

Les images du moustique sont redimensionnées puis mises en cache pour un affichage plus rapide.

3.4.4.3 Render* Bug : :parent [inherited]

Quand on ajoute un insecte à l'objet Render par la méthode addBug(), cet attribut est automatiquement initialisé.

3.4.4.4 double Bug : :size [protected, inherited]

La taille de l'insecte, influe à la fois sur la taille de la représentation graphique et sur les caractéristiques de l'insecte.'

3.4.4.5 double Bug : :speed [protected, inherited]

La vitesse en case/seconde à laquelle se déplace l'insecte.

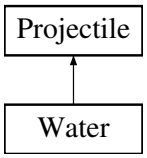
La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Mosquito.h
– src/Mosquito.cpp

3.5 Référence de la classe Projectile

Classe représentant les projectiles des tours.

`#include <Projectile.h>`

Graphe d'héritage de Projectile :



Signaux

– void explode (Projectile ∗missile)
Signal d'explosion du projectile.

Fonctions membres publiques

– Projectile (QPointF start, Bug ∗trgt, double init_speed, double damage)
Constructeur de Projectile Initialise les caractéristiques du projectile.
– QRectF boundingRect () const
Surcharge de la methode virtuelle pure boundingRect() héritée de QGraphicsObject.
– void paint (QPainter ∗painter, QColor color)
Surcharge de la méthode virtuelle pure paint() héritée de QGraphicsObject.

Fonctions membres protégées

– void advance (int step)
Surcharge de la méthode virtuelle pure héritée de QGraphicsObject.

Attributs privés

– QPointF dest
La destination du projectile.
– Bug ∗ target
L'insecte ciblée par le projectile.
– double speed
La vitesse en case par seconde du projectile.
– double dmg
Les dégats infligé à l'impact par le projectile.
– double maxrange
La portée maximale du projectile (s'il n'atteint personne avant d'arriver à se portée max, il disparait).
– double travelled
La distance parcourue par le projectile.

3.5.1 Description détaillée

Définie les méthodes des projectiles ainsi que leur caractéristiques de base.

3.5.2 Documentation des constructeurs et destructeur

3.5.2.1 Projectile : :Projectile (QPointF *start,* Bug ∗ *trgt,* double *init_speed,* double *damage*)

Paramètres

start	Point de départ du projectile.
trgt	Insecte cible du projectile.
init_speed	Vitesse de vol du projectile.
damage	Dégats infligés à l'impact par le projectile.

3.5.3 Documentation des fonctions membres

3.5.3.1 void Projectile : :advance (int *step*) [protected]

Est appellé automatiquement par Qt deux fois par appelle au slot advance de la QGraphicsScene.

Paramètres

step	appelé une fois avant de redessiner avec step = 0 et une fois avec step = 1.

3.5.3.2 QRectF Projectile : :boundingRect ( ) const

Fonction appelé automatiquement par QT pour savoir s'il doit redessiner ou non l'objet graphique.

Renvoie

Un object QRectF correspondant au rectangle englobant le dessin du projectile.

3.5.3.3 void Projectile : :explode (Projectile ∗ *missile*) [signal]

Signale émit lorsque le projectile explose pour qu'il soit retiré de la scène.

Paramètres

missile	Un pointeur vers le projectile pour que la scène puisse le détruire correctement.

3.5.3.4 void Projectile : :paint (QPainter ∗ *painter,* QColor *color*)

Est appelé automatiquement par Qt pour redessiner le projectile.

3.5.4 Documentation des champs

3.5.4.1 QPointF Projectile : :dest [private]

3.5.4.2 double Projectile : :dmg [private]

3.5.4.3 double Projectile : :maxrange [private]

3.5.4.4 double Projectile : :speed [private]

3.5.4.5 Bug∗ Projectile : :target [private]

3.5.4.6 double Projectile : :travelled [private]

La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Projectile.h
– src/moc_Projectile.cpp

– src/Projectile.cpp

3.6 Référence de la classe Render

Classe gérant le rendu graphique du jeu.

`#include <Render.h>`

Connecteurs publics

– void nextWave ()
  Lance la prochaine Vague.
– void nextBug ()
  Crée le prochaine insecte de la vague.
– void towerBought (QString type)
  Message indiquant l'achat d'une tour à la scène.
– void bugFinish (Bug ∗bug)
  Arrivée d'un insecte.
– void bugKilled (Bug ∗bug)
  Mort d'un insecte.
– void addProjectile (Projectile ∗missile)
  Ajout d'un projectile.
– void destroyTower (Tower ∗tower)
  Vente d'une tour.
– void explodingProjectile (Projectile ∗missile)
  explosion d'un projectile.

Signaux

– void newWaveName (QString name)
  Nouvelle vaque.
– void loseLife ()
  Perte d'une vie.
– void getCred ()
  Gain d'un crédit.
– void selectTower (Tower ∗tower)
  Sélection d'une tour.

Fonctions membres publiques

– Render ()
  Constructeur du rendu.
– ∼Render ()
  Destructeur de Render.
– void drawBackground (QPainter ∗painter, const QRectF &rect)
  Méthode surchargée de QGraphicsScene.
– QPoint square (QGraphicsItem &item)
  Donne les coordonnées de la case où se trouve l'objet graphique donnée.
– QPoint goal ()
  Acesseur.
– double getAngle (QPoint &current)
  Permet à un insecte de se diriger.

- void mousePressEvent (QGraphicsSceneMouseEvent ∗mouseEvent)
  Catch les événements souris.
- Bug ∗ getTarget (QPointF pos, double range, short int targetType)
  Permet aux tours de choisir leur prochaine cible.

Fonctions membres privées

- void addBug (Bug ∗bug)
  Interface de addItem() spécialisé.

Attributs privés

- QTimer waveTimer
  Timer appelant la méthode nextBug() a intervalle régulier lors d'une vague.
- QPoint ∗ start
  La position du départ des insectes.
- double start_angle
  L'angle de départ des insectes.
- int waveNumber
  Le numéro de la vague en cours.
- QStringList ∗ wave
  La liste des insectes restant a spawn dans la vague en cours.
- int map [ROW][COLUMN]
  Un tableau à deux dimensions des cases de la map avec leur contenu (chemin, boue, herbe).
- Tower ∗ towers [ROW][COLUMN]
  Un tableau à deux dimensions de pointeurs vers les tours présentent sur la map.
- QPoint goalSquare
  La position de l'arrivée des insectes.
- QHash< int, QPixmap > tiles
  La liste des images des différentes tuiles présentent sur la map.
- Hatchery ∗ b1
  Un pointeur vers la fabrique a utiliser pour spawn les insectes.
- QString tower2build
  La tour que l'on vient d'acheter et que l'on doit placer.
- QList< Bug ∗ > bugs
  Une liste de pointeur vers tous les insectes présent dans la Scene.

3.6.1 Description détaillée

Classe héritant de QGraphicsScene et gérant les interactions entre les éléments du jeu, le placement des tours...

3.6.2 Documentation des constructeurs et destructeur

3.6.2.1 Render : :Render ( )

Crée un nouvelle objet pour gérer le rendu.

3.6.2.2 Render : :∼Render ( )

Désalloue correctement la mémoire puis quitte le jeu.

3.6.3 Documentation des fonctions membres

3.6.3.1 void Render : :addBug (Bug ∗ *bug*) [private]

Permet d'ajouter un insecte en initialisant correctement son parent et an l'ajoutant à la liste spécifique aux insectes.

3.6.3.2 void Render : :addProjectile (Projectile ∗ *missile*) [slot]

Lorsqu'un projectile est tirée, Render l'ajoute à la scène.

Paramètres

missile	Un pointeur vers le projectile qui vient d'être tiré.

3.6.3.3 void Render : :bugFinish (Bug ∗ *bug*) [slot]

Quand un insecte arrive la scène le détruit et fais perdre une vie (emit loseLife()) au joueur.

Paramètres

bug	Un pointeur vers l'insecte qui vient d'arriver à la fin.

3.6.3.4 void Render : :bugKilled (Bug ∗ *bug*) [slot]

Quand un insecte est tué, la scène le détruit, appelant ses caractéristiques spéciales et ajoute un crédit au joueur.

Paramètres

bug	Un pointeur vers l'insecte qui vient d'arriver à la fin.

3.6.3.5 void Render : :destroyTower (Tower ∗ *tower*) [slot]

Lorsqu'une défense est vendu, la scène la détruit.

Paramètres

tower	Un pointeur vers la tour à détruire.

3.6.3.6 void Render : :drawBackground (QPainter ∗ *painter,* const QRectF & *rect*)

Appelé automatiquement par Qt, permet de redessiner le fond de la scène.

3.6.3.7 void Render : :explodingProjectile (Projectile ∗ *missile*) [slot]

Lorsqu'un projectile explose, la scène le détruit.

Paramètres

missile	Un pointeur vers le projectile à faire exploser.

3.6.3.8 double Render : :getAngle (QPoint & *current*)

Donne à un insecte l'angle vers lequel se tourner pour suivre le chemin.

Paramètres

current	La position actuelle de l'insecte.

Renvoie

L'angle vers lequel se diriger.

3.6.3.9 void Render : :getCred ( ) [signal]

Indique à l'UI de créditer le joueur pour un kill d'insecte.

3.6.3.10 Bug ∗ Render : :getTarget (QPointF *pos,* double *range,* short int *targetType*)

Donne l'insecte le plus avancé sur le chemin à portée de la tour et d'un type de déplacement permettant d'être touché par la tour.

Paramètres

pos	La position de la tour.
range	La portée en case de la tour.
targetType	Le(s) type(s) de cibles que la tour peut attaquer.

Renvoie

L'insecte à abattre.

3.6.3.11 QPoint Render : :goal ( )

Retourne la position de l'arrivée.

Renvoie

les coordonnées de l'arrivée sous forme d'un QPoint.

3.6.3.12 void Render : :loseLife ( ) [signal]

Indique à l'UI que le joueur viens de perdre une vie.

3.6.3.13 void Render : :mousePressEvent (QGraphicsSceneMouseEvent ∗ *mouseEvent*)

Surcharge de la méthode héritée de QGraphicsScene pour gérer les entrées souris.

Paramètres

mouseEvent	L'événement souris générée par Qt.

3.6.3.14 void Render : :newWaveName (QString *name*) [signal]

Indique à l'UI de changer le nom de la vague en cours.

Paramètres

name	le nom de la nouvelle vague.

3.6.3.15 void Render : :nextBug ( ) [slot]

Appelle la fabrique pour spawn l'insecte suivant de la vague donnée par la liste wave.

3.6.3.16 void Render : :nextWave ( ) [slot]

Lance la prochaine Vague en lisant le fichier de conguration dans map/.

3.6.3.17 void Render : :selectTower (Tower ∗ *tower*) [signal]

Indique à l'UI que le joueur vient de sélectionner une défense.

Paramètres

tower	Un pointeur vers la défense sélectionnée.

3.6.3.18 QPoint Render : :square (QGraphicsItem & *item*)

Donne les coordonnées de la case où se trouve l'objet graphique donnée.

Paramètres

item	Un pointeur vers l'objet dont on veut connaitre la position.

Renvoie

Les coordonnée de la case sous forme d'un QPoint.

3.6.3.19 void Render : :towerBought (QString *type*) [slot]

Message indiquant l'achat d'une tour à la scène.

Paramètres

type	Le type de tour à placer.

3.6.4 Documentation des champs

3.6.4.1 Hatchery* Render : :b1 [private]

3.6.4.2 QList<Bug *> Render : :bugs [private]

3.6.4.3 QPoint Render : :goalSquare [private]

3.6.4.4 int Render : :map[ROW][COLUMN] [private]

3.6.4.5 QPoint* Render : :start [private]

3.6.4.6 double Render : :start_angle [private]

3.6.4.7 QHash<int, QPixmap> Render : :tiles [private]

3.6.4.8 QString Render : :tower2build [private]

3.6.4.9 Tower* Render : :towers[ROW][COLUMN] [private]

3.6.4.10 QStringList* Render : :wave [private]

3.6.4.11 int Render : :waveNumber [private]

3.6.4.12 QTimer Render : :waveTimer [private]

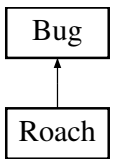
La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Render.h
– src/moc_Render.cpp
– src/Render.cpp

3.7 Référence de la classe Roach

Classe représentant un cafard.

`#include <Roach.h>`

Graphe d'héritage de Roach :



Signaux

– void dead (Bug *bug)

Signal de mort de l'insecte.
- void goalReached (Bug *bug)
  Signal d'arrivé.

Fonctions membres publiques

- Roach (double x, double y, double s, double start_angle)
  Constructeur de Roach.
- ~Roach ()
  Destructeur de Roach.
- void paint (QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget)

  Surcharge de la méthode virtuelle pure paint() héritée de QGraphicsObject.
- QRectF boundingRect () const
  Surcharge de la fonction boudingRect() héritée de QGraphicsObject.
- QPainterPath shape () const
  Surcharge de la méthode virtuelle pure shape() héritée de QGraphicsObject.
- void hit (double dmg)
  Infliger des dégats à l'insecte.
- short int getMoveType ()
  Permet d'accéder à la valeur moveType de l'insecte.

Champs de données

- Render * parent
  L'objet Render parent de l'insecte.

Fonctions membres protégées

- void advance (int step)
  Implémentation de la fonction virtuelle pure advance() héritée de QGGaphicsObject.

Attributs protégés

- double size
  Taille.
- int frame
  Compteur de frame.
- double speed
  Vitesse de base.

Attributs privés

- QImage * image [3]
  Cache d'images.

3.7.1 Description détaillée

Définie les caractéristiques spécifiques aux cafards, comme leur graphisme, leur type de déplacement (CRAWL)...

3.7.2 Documentation des constructeurs et destructeur

3.7.2.1 Roach : :Roach (double *x,* double *y,* double *s,* double *start_angle*)

Appelle le constructeur de Bug avec les paramètres correspondants aux caractéristiques d'un cafard.

3.7.2.2 Roach : :~Roach ( )

Désalloue la mémoire correctement.

3.7.3 Documentation des fonctions membres

3.7.3.1 void Bug : :advance (int *step*) [protected, inherited]

Implémentation de la fonction virtuelle pure advance() héritée de QGGaphicsObject.

Paramètres

step	La fonction est appelé deux fois automatiquement par QT à chaque update(), une fois avec 0 comme paramètre puis une fois avec 1.

3.7.3.2 QRectF Bug : :boundingRect ( ) const [inherited]

Fonction appelé automatiquement par QT pour savoir s'il doit ou non réafficher l'insecte.

Renvoie

Un objet QRectF correspondant au rectangle englobant l'ensemble du dessin de l'insecte.

3.7.3.3 void Bug : :dead (Bug * *bug*) [signal, inherited]

Signal émit lorsque l'insecte est tué.

Paramètres

bug	Un pointeur vers l'insecte qui vient de mourir.

3.7.3.4 short int Bug : :getMoveType ( ) [inherited]

Renvoie

la valeur prédéfinie FLY ou CRAWL.

3.7.3.5 void Bug : :goalReached (Bug * *bug*) [signal, inherited]

Signal émit par l'insecte lorsqu'il atteint son but.

Paramètres

bug	Un pointeur vers l'insecte.

3.7.3.6 void Bug : :hit (double *dmg*) [inherited]

Fonction pouvant être appelée pour infliger des dégats à l'insecte (par exemple par les projectiles des tours).

Paramètres

dmg	Un double correspondant au point de dégat à infliger (avant réduction par la résistance de l'insecte).

Réimplémentée dans Ant.

3.7.3.7 void Roach : :paint (QPainter ∗ *painter,* const QStyleOptionGraphicsItem ∗ *option,* QWidget ∗ *widget*)

Est appelé automatiquement par Qt pour redessiner le cafard.

3.7.3.8 QPainterPath Bug : :shape ( ) const [inherited]

Fonction utilisé par QT pour traiter les collisions entre objets graphiques.

Renvoie

Un object QPainterPath correspondant au contour de collision de l'insecte.

3.7.4 Documentation des champs

3.7.4.1 int Bug : :frame [protected, inherited]

Compteur d'image utilisé pour afficher successivement chaque image des animations.

3.7.4.2 QImage∗ Roach : :image[3] [private]

Les images du cafard à chaque position, redimensionnées en fonction de sa taille.

3.7.4.3 Render∗ Bug : :parent [inherited]

Quand on ajoute un insecte à l'objet Render par la méthode addBug(), cet attribut est automatiquement initialisé.

3.7.4.4 double Bug : :size [protected, inherited]

La taille de l'insecte, influe à la fois sur la taille de la représentation graphique et sur les caractéristiques de l'insecte.'

3.7.4.5 double Bug : :speed [protected, inherited]

La vitesse en case/seconde à laquelle se déplace l'insecte.

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

3.8 Référence de la classe Tower

Classe représentant toutes les défenses.

`#include <Tower.h>`

Connecteurs publics

– void fire ()
Signal de tir reçu.

Signaux

– void projectile (Projectile ∗missile)
Création de projectile.

Fonctions membres publiques

– Tower (QPointF buildPos, QString typeTower)
Constructeur de Tower.
– QRectF boundingRect () const
Surcharge de la méthode virtuelle pure boudingRect() héritée de QGraphicsObject.
– void paint (QPainter ∗painter, const QStyleOptionGraphicsItem ∗option, QWidget ∗widget)

Surcharge de la méthode virtuelle pure paint() héritée de QGraphicsObject.
– QString getType ()
Assesseur.
– double getRange ()
Assesseur.
– short int getLvl ()
Assesseur.
– double getFirerate ()
Assesseur.
– float getPrice ()
Assesseur.
– float getUpgCost ()
Assesseur.
– void upgrade ()
Amélioration de la défense.

Champs de données

– Render ∗ parent

L'objet Render parent de la défense.

Attributs privés

- short int level
  Le niveau actuel de la défense.
- float price
  Le prix de revent de la défense.
- float upg1
  Le coût pour améliorer la défense du niveau 1 au 2.
- float upg2
  Le coût pour améliorer la défense du niveau 2 au 3.
- QColor color
  La couleur de la défense.
- short int targetType
  Le type de cible que peut atteindre la défense.
- double range
  La portée de la défense.
- double firerate
  La cadence de tir de la défense.
- QTimer * timer
  Timer utilisé pour la cadence de tir de la défense.
- QPointF pos
  La position sur la map de la défense.
- QString type
  Le type spécifique de la défense.

3.8.1 Description détaillée

Implémente les méthodes communes à toutes les défenses (la différenciation se fait surtout au niveau des projectiles et des stats des défenses).

3.8.2 Documentation des constructeurs et destructeur

3.8.2.1 Tower : :Tower (QPointF *buildPos*, QString *typeTower*)

Initialise les caractéristiques de la tour selon son type.

Paramètres

buildPos	La position où placer la défense.
typeTower	Le type de défense à créer.

3.8.3 Documentation des fonctions membres

3.8.3.1 QRectF Tower : :boundingRect ( ) const

Appelé automatiquement par Qt pour savoir s'il doit ou non redessiner l'objet.

Renvoie

Le rectangle englobant tous les dessins de la défense.

3.8.3.2 void Tower : :fire ( ) [slot]

Méthode appelé par le timer de cadence pour tirer.

3.8.3.3 double Tower : :getFirerate ( )

Assesseur vers firerate.

Renvoie

Le contenu de l'attribut firerate.

3.8.3.4 short int Tower : :getLvl ( )

Assesseur vers level.

Renvoie

Le contenu de l'attribut level.

3.8.3.5 float Tower : :getPrice ( )

Assesseur vers price.

Renvoie

Le contenu de l'attribut price.

3.8.3.6 double Tower : :getRange ( )

Assesseur vers range.

Renvoie

Le contenu de l'attribut range.

3.8.3.7 QString Tower : :getType ( )

Assesseur vers type.

Renvoie

Le contenu de l'attribut type.

3.8.3.8 float Tower : :getUpgCost ( )

Assesseur vers upg.

Renvoie

Le prix pour améliorer la défense vers le niveau suivant.

3.8.3.9 void Tower : :paint (QPainter ∗ *painter,* const QStyleOptionGraphicsItem ∗ *option,* QWidget ∗ *widget*)

Appelé automatiquement par Qt pour dessiner la défense.

3.8.3.10 void Tower : :projectile (Projectile ∗ *missile*) [signal]

Envoie le projectile créer à la scène.

Paramètres

missile	Un pointeur vers le projectile tirée.

3.8.3.11 void Tower : :upgrade ( )

Améliore la défense (et donc ses caractéristiques) vers le niveau suivant.

3.8.4 Documentation des champs

3.8.4.1 QColor Tower : :color [private]

3.8.4.2 double Tower : :firerate [private]

3.8.4.3 short int Tower : :level [private]

3.8.4.4 Render∗ Tower : :parent

3.8.4.5 QPointF Tower : :pos [private]

3.8.4.6 float Tower : :price [private]

3.8.4.7 double Tower : :range [private]

3.8.4.8 short int Tower : :targetType [private]

3.8.4.9 QTimer∗ Tower : :timer [private]

3.8.4.10 QString Tower : :type [private]

3.8.4.11 float Tower : :upg1 [private]

3.8.4.12 float Tower : :upg2 [private]

La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Tower.h
– src/moc_Tower.cpp
– src/Tower.cpp

3.9 Référence de la classe UI

Classe de l'interface graphique.

`#include <UI.h>`

Connecteurs publics

- void setWaveName (QString name)
  Changement de vague.
- void addCred ()
  Gain d'un crédit.
- void loseLife ()
  Perte de vie.
- void startWave ()
  Vague suivante.
- void buyWaterTower ()
  Achat d'une défense "pistolet à eau".
- void buySlingshotTower ()
  Achat d'une défense "lance-pierres".
- void buyPaintballTower ()
  Achat d'une défense "paintball".
- void buyBowlingTower ()
  Achat d'une défense "pétanque".
- void selectTower (Tower *tower)
  Sélection d'une tour.
- void sellSelectedTower ()
  Vente de la tour sélectionnée.
- void upgradeSelectedTower ()
  Amélioration de la défense sélectionnée.

Signaux

- void buyTower (QString type)
  Achat d'une tour.
- void nextWave ()
  Vague suivante.
- void towerSold (Tower *tower)
  Vente d'une défense.
- void defeat ()
  Signal de défaite.

Fonctions membres publiques

- UI (QWidget *parent)
  Constructeur de UI.

Champs de données

- QPushButton * start
  Le bouton pour lancer la vague suivante.

Attributs privés

- QGroupBox * tower
  Regroupe les éléments du magasin de défenses.
- QGroupBox * stats
  Regroupe les éléments ayant attrait à la défense sélectionnée.
- QPushButton * sell
  Le bouton pour vendre la tour sélectionnée.
- QPushButton * upg
  Le bouton pour améliorer la tour sélectionnée.
- QPushButton * water
  Le bouton pour acheter un pistolet à eau.
- QPushButton * sling
  Le bouton pour acheter un lance-pierre.
- QPushButton * bowling
  Le bouton pour engager un joueur de pétanque.
- QPushButton * paintball
  Le bouton pour acheter un Paintball.
- QLabel * name
  Le texte 'Type : '.
- QLabel * lvl
  Le texte 'Niveau : '.
- QLabel * range
  Le texte 'portée : '.
- QLabel * firerate
  Le texte 'cadence : '.
- QLabel * t_name
  Le type de la tour sélectionnée.
- QLabel * t_lvl
  Le niveau de la tour sélectionnée.
- QLabel * t_range
  La portée de la tour sélectionnée.
- QLabel * t_firerate
  La cadence de tir de la tour sélectionnée.
- QLabel * wave
  le nom de la vague en cours.
- QLabel * cred_txt
  Le texte 'crédits : '.
- QLabel * life_txt
  Le texte 'vie : '.
- QLCDNumber * cred
  L'affciheur de crédits restants.
- QLCDNumber * life
  L'afficheur de vie restantes.
- Tower * selected
  Un pointeur vers la tour sélectionnée actuellement.

3.9.1 Description détaillée

Gère l'ensemble des éléments de l'interface graphique affichée à droite.

3.9.2 Documentation des constructeurs et destructeur

3.9.2.1 UI : :UI (QWidget * *parent*)

initialise l'interface utilisateur.

Paramètres

parent	Le widge parent de l'UI.

3.9.3 Documentation des fonctions membres

3.9.3.1 void UI : :addCred ( ) [slot]

Incrémente de 1 les crédit du joueur.

3.9.3.2 void UI : :buyBowlingTower ( ) [slot]

Vérifie si le joueur possède suffisament de crédit puis décompte le prix.

3.9.3.3 void UI : :buyPaintballTower ( ) [slot]

Vérifie si le joueur possède suffisament de crédit puis décompte le prix.

3.9.3.4 void UI : :buySlingshotTower ( ) [slot]

Vérifie si le joueur possède suffisament de crédit puis décompte le prix.

3.9.3.5 void UI : :buyTower (QString *type*) [signal]

Indique au Render de placer la défense achetée.

Paramètres

type	Le type de la défense achetée.

3.9.3.6 void UI : :buyWaterTower ( ) [slot]

Vérifie si le joueur possède suffisament de crédit puis décompte le prix.

3.9.3.7 void UI : :defeat ( ) [signal]

Indique au programme de stopper le jeu.

3.9.3.8 void UI : :loseLife ( ) [slot]

Décrémente de 1 le total de vies du joueur.

3.9.3.9 void UI : :nextWave ( ) [signal]

Indique au Render de lancer la vague suivante.

3.9.3.10 void UI : :selectTower (Tower * *tower*) [slot]

Affiche les caractérisque de la défense sélectionnée.

Paramètres

tower	Un pointeur vers la nouvelle défense sélectionnée.

3.9.3.11 void UI : :sellSelectedTower ( ) [slot]

Reçoit le signal du bouton "Vendre" et recrédite le joueur du moitié du prix de la défense.

3.9.3.12 void UI : :setWaveName (QString *name*) [slot]

Change le nom de la vague courante.

Paramètres

name	Le nom de la nouvelle vague.

3.9.3.13 void UI : :startWave ( ) [slot]

Reçoit le signal du bouton "lancer la vague suivante".

3.9.3.14 void UI : :towerSold (Tower * *tower*) [signal]

Indique au Render de détruire une défense vendue.

Paramètres

tower	Un pointeur vers la défense à détruire.

3.9.3.15 void UI : :upgradeSelectedTower ( ) [slot]

Reçoit le signal du bouton "améliorer" et améliore la tour si les crédits sont suffisants.

3.9.4 Documentation des champs

3.9.4.1 QPushButton* UI : :bowling [private]

3.9.4.2 QLCDNumber* UI : :cred [private]

3.9.4.3 QLabel* UI : :cred_txt [private]

3.9.4.4 QLabel* UI : :firerate [private]

3.9.4.5 QLCDNumber* UI : :life [private]

3.9.4.6 QLabel* UI : :life_txt [private]

3.9.4.7 QLabel* UI : :lvl [private]

3.9.4.8 QLabel* UI : :name [private]

3.9.4.9 QPushButton* UI : :paintball [private]

3.9.4.10 QLabel* UI : :range [private]

3.9.4.11 Tower* UI : :selected [private]

3.9.4.12 QPushButton* UI : :sell [private]

3.9.4.13 QPushButton* UI : :sling [private]

3.9.4.14 QPushButton* UI : :start

3.9.4.15 QGroupBox* UI : :stats [private]

3.9.4.16 QLabel* UI : :t_firerate [private]

3.9.4.17 QLabel* UI : :t_lvl [private]

3.9.4.18 QLabel* UI : :t_name [private]

3.9.4.19 QLabel* UI : :t_range [private]

3.9.4.20 QGroupBox* UI : :tower [private]

3.9.4.21 QPushButton* UI : :upg [private]

3.9.4.22 QPushButton* UI : :water [private]

3.9.4.23 QLabel* UI : :wave [private]

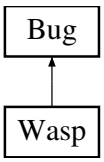
La documentation de cette classe a été générée à partir des fichiers suivants :
– src/UI.h
– src/moc_UI.cpp
– src/UI.cpp

3.10 Référence de la classe Wasp

Classe représentant les guêpes.

`#include <Wasp.h>`

Graphe d'héritage de Wasp :



Signaux

- void dead (Bug *bug)
  Signal de mort de l'insecte.
- void goalReached (Bug *bug)
  Signal d'arrivé.

Fonctions membres publiques

- Wasp (double x, double y, double s, double start_angle)
  Constructeur de Wasp.
- ~Wasp ()
  Destructeur de Wasp.
- void paint (QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget)

  Surcharge de la méthode virtuelle pure paint() héritée de QGraphicsObject.
- QRectF boundingRect () const
  Surcharge de la fonction boudingRect() héritée de QGraphicsObject.
- QPainterPath shape () const
  Surcharge de la méthode virtuelle pure shape() héritée de QGraphicsObject.
- void hit (double dmg)
  Infliger des dégats à l'insecte.
- short int getMoveType ()
  Permet d'accéder à la valeur moveType de l'insecte.

Champs de données

- Render * parent
  L'objet Render parent de l'insecte.

Fonctions membres protégées

- void advance (int step)
  Implémentation de la fonction virtuelle pure advance() héritée de QGGaphicsObject.

Attributs protégés

– double size
  Taille.
– int frame
  Compteur de frame.
– double speed
  Vitesse de base.

Attributs privés

– QImage ∗ image [2]
  Cache d'images.

3.10.1 Description détaillée

Définie les caractéristiques propres aux guêpes.

3.10.2 Documentation des constructeurs et destructeur

3.10.2.1 Wasp : :Wasp (double *x,* double *y,* double *s,* double *start_angle*)

Appelle le constructeur de Bug avec les paramètres spécifiques aux guêpes.

Paramètres

x	L'abscisse du départ des insectes.
y	L'ordonnée du départ des insectes.
s	La taille de la guêpe.
start_angle	l'angle de départ des insectes.

3.10.2.2 Wasp : :∼Wasp ( )

Désalloue la mémoire des images.

3.10.3 Documentation des fonctions membres

3.10.3.1 void Bug : :advance (int *step*) [protected, inherited]

Implémentation de la fonction virtuelle pure advance() héritée de QGGaphicsObject.

Paramètres

step	La fonction est appelé deux fois automatiquement par QT à chaque update(), une fois avec 0 comme paramètre puis une fois avec 1.

3.10.3.2 QRectF Bug : :boundingRect ( ) const [inherited]

Fonction appelé automatiquement par QT pour savoir s'il doit ou non réafficher l'insecte.

Renvoie

Un objet QRectF correspondant au rectangle englobant l'ensemble du dessin de l'insecte.

3.10.3.3 void Bug : :dead (Bug * *bug*) [signal, inherited]

Signal émit lorsque l'insecte est tué.

Paramètres

bug	Un pointeur vers l'insecte qui vient de mourir.

3.10.3.4 short int Bug : :getMoveType ( ) [inherited]

Renvoie

la valeur prédéfinie FLY ou CRAWL.

3.10.3.5 void Bug : :goalReached (Bug * *bug*) [signal, inherited]

Signal émit par l'insecte lorsqu'il atteint son but.

Paramètres

bug	Un pointeur vers l'insecte.

3.10.3.6 void Bug : :hit (double *dmg*) [inherited]

Fonction pouvant être appelée pour infliger des dégats à l'insecte (par exemple par les projectiles des tours).

Paramètres

dmg	Un double correspondant au point de dégat à infliger (avant réduction par la résistance de l'insecte).

Réimplémentée dans Ant.

3.10.3.7 void Wasp : :paint (QPainter * *painter,* const QStyleOptionGraphicsItem * *option,* QWidget * *widget*)

Est appelé automatiquement par Qt pour dessiner la guêpe.

3.10.3.8 QPainterPath Bug : :shape ( ) const [inherited]

Fonction utilisé par QT pour traiter les collisions entre objets graphiques.

Renvoie

Un object QPainterPath correspondant au contour de collision de l'insecte.

3.10.4 Documentation des champs

3.10.4.1 int Bug : :frame [protected, inherited]

Compteur d'image utilisé pour afficher successivement chaque image des animations.

3.10.4.2 QImage∗ Wasp : :image[2] [private]

Les images de la guêpes redimensionnées.

3.10.4.3 Render∗ Bug : :parent [inherited]

Quand on ajoute un insecte à l'objet Render par la méthode addBug(), cet attribut est automatiquement initialisé.

3.10.4.4 double Bug : :size [protected, inherited]

La taille de l'insecte, influe à la fois sur la taille de la représentation graphique et sur les caractéristiques de l'insecte.'

3.10.4.5 double Bug : :speed [protected, inherited]

La vitesse en case/seconde à laquelle se déplace l'insecte.

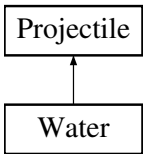
La documentation de cette classe a été générée à partir des fichiers suivants :
– src/Wasp.h
– src/Wasp.cpp

3.11 Référence de la classe Water

Classe représentant les projectiles du pistolet à eau.

`#include <Water.h>`

Graphe d'héritage de Water :



Signaux

– void explode (Projectile ∗missile)
Signal d'explosion du projectile.

Fonctions membres publiques

– Water (QPointF start, Bug ∗target, short int lvl)
Constructeur de Ware.
– void paint (QPainter ∗painter, const QStyleOptionGraphicsItem ∗option, QWidget ∗widget)

Surcharge de la méthode virtuelle pure paint() héritée de QGraphicsObject.
– QRectF boundingRect () const
Surcharge de la methode virtuelle pure boundingRect() héritée de QGraphicsObject.
– void paint (QPainter ∗painter, QColor color)
Surcharge de la méthode virtuelle pure paint() héritée de QGraphicsObject.

Fonctions membres protégées

– void advance (int step)
Surcharge de la méthode virtuelle pure héritée de QGraphicsObject.

3.11.1 Description détaillée

Classe représentant les projectiles du pistolet à eau, rien de spéciale.

3.11.2 Documentation des constructeurs et destructeur

3.11.2.1 Water : :Water (QPointF *start,* Bug ∗ *target,* short int *lvl*)

Appelle le constructeur de Projectile avec les caractéristiques du pistolet à eau.

Paramètres

start	Position de départ.
target	Pointeur vers l'insecte ciblé.
lvl	Le niveau du projectile.

3.11.3 Documentation des fonctions membres

3.11.3.1 void Projectile : :advance (int *step*) [protected, inherited]

Est appellé automatiquement par Qt deux fois par appelle au slot advance de la QGraphicsScene.

Paramètres

step	appelé une fois avant de redessiner avec step = 0 et une fois avec step = 1.

3.11.3.2 QRectF Projectile : :boundingRect ( ) const [inherited]

Fonction appelé automatiquement par QT pour savoir s'il doit redessiner ou non l'objet graphique.

Renvoie

Un object QRectF correspondant au rectangle englobant le dessin du projectile.

3.11.3.3 void Projectile : :explode (Projectile ∗ *missile*) [signal, inherited]

Signale émit lorsque le projectile explose pour qu'il soit retiré de la scène.

Paramètres

missile	Un pointeur vers le projectile pour que la scène puisse le détruire correctement.

3.11.3.4 void Water : :paint (QPainter ∗ *painter,* const QStyleOptionGraphicsItem ∗ *option,* QWidget ∗ *widget*)

Appelé automatiquement par Qt pour redessiner l'objet.

3.11.3.5 void Projectile : :paint (QPainter ∗ *painter,* QColor *color*) [inherited]

Est appelé automatiquement par Qt pour redessiner le projectile.

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

Index