Drones

Généré par Doxygen 1.8.14

Table des matières

Chapitre 1

Index hiérarchique

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

Index des fichiers

3.1 Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

Chapitre 4

Documentation des classes

4.1 Référence de la classe Affichage

`#include <Affichage.h>`

Fonctions membres publiques

- Affichage (Environnement *env)
- virtual ~Affichage ()
- void draw (TrackBallCamera *, GLuint)

4.1.1 Description détaillée

Classe qui permet d'afficher en 3D l'Environnement. Cela inclut principalement les Drones et les Obstacles. Utilise openGL et SDL

Auteurs

Timothé, Simon

4.1.2 Documentation des constructeurs et destructeur

4.1.2.1 Affichage()

```
Affichage::Affichage (
            Environnement * env )
```

Un constructeur utilisant un environnement pour s'y lier par pointeur. initialise aussi la texture des drones directement par une valeur donnée dans le constructeur.

Paramètres

env	l'Environnement vers lequel pointer; sur lequel Affichage devra faire son travail.

4.1.2.2 ~Affichage()

```
Affichage::~Affichage ( )  [virtual]
```

Simple Destructeur de l'Affichage.

4.1.3 Documentation des fonctions membres

4.1.3.1 draw()

```
void Affichage::draw (
            TrackBallCamera * camera,
            GLuint droneText )
```

Méthode principale, affichant l'Environnement en attribut

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

- Affichage.h
- Affichage.cpp

4.2 Référence de la classe Capteur

```
#include <Capteur.h>
```

Fonctions membres publiques

- Capteur (const float &p, const VecteurR3 &dir, Environnement *environnement)
- virtual ~Capteur ()
- float getDistanceDetectee () const
- float getPortee () const
- void **updateDistanceDetecteeObstacle** ()
- void **updateDistanceDetecteeBords** ()
- void updateDistanceDetectee ()
- VecteurR3 getDirection () const
- bool detecteQQch () const
- void associerInfoDrone (const float, VecteurR3 *)

4.2.1 Description détaillée

Auteurs

Timothé

Date

20 Avril 2018

Les capteurs sont les outils nécéssaires aux drones pour detecter les obstacles alentours à une distance donnée.

4.2.2 Documentation des constructeurs et destructeur

4.2.2.1 Capteur()

```
Capteur::Capteur (
            const float & p,
            const VecteurR3 & dir,
            Environnement * environnement )
```

Constructeur de Capteur initialisant tous ses paramètres à des valeurs données en entrée.

4.2.2.2 ~Capteur()

```
Capteur::~Capteur ( )  [virtual]
```

Déstructeur de Capteur

4.2.3 Documentation des fonctions membres

4.2.3.1 associerInfoDrone()

```
void Capteur::associerInfoDrone (
            const float rayon,
            VecteurR3 * _pPositionDrone )
```

Associe les informations (utiles au Capteur) du Drone à ce Capteur

4.2.3.2 detecteQQch()

```
bool Capteur::detecteQQch ( ) const
```

Renvoie un boolean donnant si le capteur detecte un obstacle (distDetectee<portee)

4.2.3.3 getDirection()

```
VecteurR3 Capteur::getDirection ( ) const
```

Getter de la direction du capteur

4.2.3.4 getDistanceDetectee()

```
float Capteur::getDistanceDetectee ( ) const
```

Getter de la distance detectee par le capteur

4.2.3.5 getPortee()

```
float Capteur::getPortee ( ) const
```

Getter portée du capteur

4.2.3.6 updateDistanceDetectee()

```
void Capteur::updateDistanceDetectee ( )
```

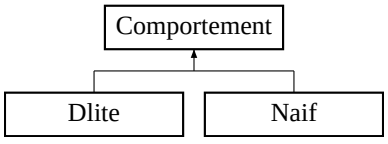
Calcul la distance entre le drone et les obstacles alentours. La fonction sera appelée par drone, de manière itérative.

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

4.3 Référence de la classe Comportement

```
#include <Comportement.h>
```

Graphe d'héritage de Comportement:



Fonctions membres publiques

- Comportement ()
- virtual ~Comportement ()
- virtual VecteurR3 allerPoint (const VecteurR3 &posActuelle, const VecteurR3 &destination, const std::vector< Capteur > vCapteurs, const VecteurR3 vitesse)=0

4.3.1 Description détaillée

Interface donnant la fonction fondamentale de comportement de chaque Drone: le choix d'un nouveau vecteur accélération.

Auteur

Louis

4.3.2 Documentation des constructeurs et destructeur

4.3.2.1 Comportement()

```
Comportement::Comportement ( )
```

Constructeur vide.

4.3.2.2 ~Comportement()

```
Comportement::~Comportement ( )  [virtual]
```

Destructeur vide.

4.3.3 Documentation des fonctions membres

4.3.3.1 allerPoint()

```
virtual VecteurR3 Comportement::allerPoint (
            const VecteurR3 & posActuelle,
            const VecteurR3 & destination,
            const std::vector< Capteur > vCapteurs,
            const VecteurR3 vitesse )  [pure virtual]
```

Méthode fondamentale de Comportement des Drones. A partir des positions du Drone et de son premier objectif, détermine (la méthode importe peu ici) le vecteur accélération pour la frame suivante.

Paramètres

posActuelle	la position du Drone au temps t.
destination	la position à atteindre.
vCapteurs,le	vecteur des Capteurs donnant l'information sensorielle du Drone.

Renvoie

le vecteur accélération

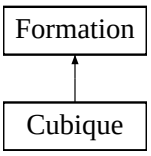
Implémenté dans Naif.

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

4.4 Référence de la classe Cubique

`#include <Cubique.h>`

Graphe d'héritage de Cubique:



Fonctions membres publiques

— Cubique (VecteurR3, float, int)
— virtual ∼Cubique ()
— virtual vector< VecteurR3 > genererMaillage ()

Attributs protégés

— float longueurCote
— VecteurR3 origine

4.4.1 Description détaillée

Classe fille de Formation, qui permet de dessiner un cube.

Auteur

Margot, Théau et Morgan

Date

13/04/18

4.4.2 Documentation des constructeurs et destructeur

4.4.2.1 Cubique()

```
Cubique::Cubique (
            VecteurR3 origine,
            float longueur,
            int nbreDrone )
```

Constructeur de la Formation.

4.4.2.2 ~Cubique()

```
Cubique::~Cubique ( )  [virtual]
```

Destructeur usuel de la Formation.

4.4.3 Documentation des fonctions membres

4.4.3.1 genererMaillage()

```
vector< VecteurR3 > Cubique::genererMaillage ( )  [virtual]
```

Méthode héritée, calcule le maillage adapté à la formation cubique

Implémente Formation.

4.4.4 Documentation des données membres

4.4.4.1 longueurCote

```
float Cubique::longueurCote  [protected]
```

Longueur du côté du cube

4.4.4.2 origine

```
VecteurR3 Cubique::origine  [protected]
```

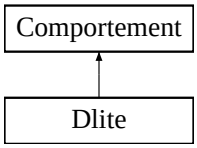
Orgine du cube

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

4.5 Référence de la classe Dlite

`#include <Dlite.h>`

Graphe d'héritage de Dlite:



Fonctions membres publiques

- Dlite ()
- virtual ∼Dlite ()
- VecteurR3 allerPoint (VecteurR3 posActuelle, VecteurR3 destination, std::vector< Capteur > vCapteurs)

4.5.1 Description détaillée

Type de Comportement: algorithme Dlite: amélioration dynamique de l'algorithme de pathfinding conventionnel A∗.

4.5.2 Documentation des constructeurs et destructeur

4.5.2.1 Dlite()

```
Dlite::Dlite ( )
```

Constructeur de l'algorithme.

4.5.2.2 ∼Dlite()

```
Dlite::~Dlite ( )  [virtual]
```

Destructeur de l'algorithme.

4.5.3 Documentation des fonctions membres

4.5.3.1 allerPoint()

```
VecteurR3 Dlite::allerPoint (
            VecteurR3 posActuelle,
            VecteurR3 destination,
            std::vector< Capteur > vCapteurs )
```

Méthode fondamentale de Comportement des Drones. A partir des positions du Drone, de son premier objectif et des capteurs, détermine le vecteur accélération pour la frame suivante.

Paramètres

posActuelle	la position du Drone au temps t.
destination	la position à atteindre.
vCapteurs,le	vecteur des Capteurs donnant l'information sensorielle du Drone.

Renvoie

le vecteur accélération

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

— Dlite.h
— Dlite.cpp

4.6 Référence de la classe Drone

`#include <Drone.h>`

Fonctions membres publiques

— Drone (const VecteurR3)
— Drone (const float, const VecteurR3, const std::vector< Capteur > &)
— Drone (const float rayon, VecteurR3 pos, std::vector< Capteur > vCap, const VecteurR3 _gravite, VecteurR3 vit=VecteurR3())
— Drone **operator++** (int)
— virtual ~Drone ()
— VecteurR3 getPremierObjectif () const
— std::queue< VecteurR3 > getVObjectifs () const
— std::vector< Capteur > getVCapteurs () const
— VecteurR3 getPosition () const
— VecteurR3 getVitesse () const
— VecteurR3 getAcceleration () const
— void setPosition (const VecteurR3)
— void setVitesse (const VecteurR3)
— void setAcceleration (const VecteurR3 &)
— float getRayon () const
— void neFonctionnePlus ()
— bool estFonctionnel () const
— bool aObjectif () const
— bool atteintObjectif ()
— bool porteUnColis () const
— void ajouterObjectif (const VecteurR3 &obj)
— void livrerColis (const VecteurR3 &retrait, const VecteurR3 &depot)

4.6.1 Description détaillée

Agent du réseau qui a pour principale fonctionnalité de pouvoir se rendre d'un point à un autre, en suivant liste d'objectifs. Il se déplace en se donnant un vecteur accélération, qui donnera sa vitesse et position en temps t+1 via la classe Environnement.

Auteur

Louis, Quentin

4.6.2 Documentation des constructeurs et destructeur

4.6.2.1 Drone() [1/3]

```
Drone::Drone (
            const VecteurR3 pos )
```

Constructeur de Drone pour tests, avec position

4.6.2.2 Drone() [2/3]

```
Drone::Drone (
            const float rayonDrone,
            const VecteurR3 pos,
            const std::vector< Capteur > & vCap )
```

Constructeur de test des Capteurs

4.6.2.3 Drone() [3/3]

```
Drone::Drone (
            const float rayon,
            VecteurR3 pos,
            std::vector< Capteur > vCap,
            const VecteurR3 _gravite,
            VecteurR3 vit = VecteurR3() )
```

Constructeur effectif de Drone. Prend une position et (éventuellement) vitesse initiales, un rayon et un vecteur de Capteurs.

4.6.2.4 ~Drone()

```
Drone::~Drone ( )  [virtual]
```

Destructeur de Drone.

4.6.3 Documentation des fonctions membres

4.6.3.1 ajouterObjectif()

```
void Drone::ajouterObjectif (
            const VecteurR3 & obj )
```

Méthode qui ajoute une destination à la liste des objectifs.

Paramètres

obj	le point de R3 à ajouter à la liste d'objectifs.

4.6.3.2 aObjectif()

```
bool Drone::aObjectif ( ) const
```

Fonction qui indique si le drone a au moins un objectif (liste non nulle)

4.6.3.3 atteintObjectif()

```
bool Drone::atteintObjectif ( )
```

Teste si le drone a atteint son objectif. Le cas échéant, l'objectif est supprimé de la liste.

4.6.3.4 estFonctionnel()

```
bool Drone::estFonctionnel ( ) const
```

Getter sur l'état du drone

4.6.3.5 getAcceleration()

```
VecteurR3 Drone::getAcceleration ( ) const
```

Getter du vecteur acceleration

4.6.3.6 getPosition()

```
VecteurR3 Drone::getPosition ( ) const
```

Getter du vecteur position

4.6.3.7 getPremierObjectif()

```
VecteurR3 Drone::getPremierObjectif ( ) const
```

Getter du premier objectif du Drone

4.6.3.8 getRayon()

```
float Drone::getRayon ( ) const
```

Getter du rayon du Drone

4.6.3.9 getVCapteurs()

```
std::vector< Capteur > Drone::getVCapteurs ( ) const
```

Getter du vecteur de capteurs

4.6.3.10 getVitesse()

```
VecteurR3 Drone::getVitesse ( ) const
```

Getter du vecteur vitesse

4.6.3.11 getVObjectifs()

```
std::queue< VecteurR3 > Drone::getVObjectifs ( ) const
```

Getter des objectifs du Drone

4.6.3.12 livrerColis()

```
void Drone::livrerColis (
            const VecteurR3 & retrait,
            const VecteurR3 & depot )
```

Ajoute à liste des objectifs le point de retrait et de dépôt du colis.

Paramètres

retrait	Le point auquel récupérer le colis.
depot	Le point auquel déposer le colis.

4.6.3.13 neFonctionnePlus()

```
void Drone::neFonctionnePlus ( )
```

affecte au bool 'fonctionne' la valeur false.

4.6.3.14 porteUnColis()

```
bool Drone::porteUnColis ( ) const
```

getter porteColis

4.6.3.15 setAcceleration()

```
void Drone::setAcceleration (
            const VecteurR3 & _acc )
```

Setter du vecteur acceleration

4.6.3.16 setPosition()

```
void Drone::setPosition (
            const VecteurR3 nvllPos )
```

Setter du vecteur position

4.6.3.17 setVitesse()

```
void Drone::setVitesse (
            const VecteurR3 nvllVit )
```

Setter du vecteur vitesse

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

4.7 Référence de la classe Environnement

```
#include <Environnement.h>
```

Fonctions membres publiques

- Environnement ()
- Environnement (const float tailleCote)
- float **getCote** () const
- VecteurR3 **getGravite** () const
- std::vector< VecteurR3 > **getVRetraits** () const
- std::vector< VecteurR3 > **getVDepots** () const
- VecteurR3 **getOrigineEnv** () const
- std::vector< Obstacle > **getVObstacles** () const
- Environnement operator++ (int)
- void ajouterObstacle (const Obstacle &)
- void ajouterColis (const VecteurR3 &, const VecteurR3 &)
- void associerEssaim (Essaim *)
- std::vector< Drone * > getEssaimDrones () const

4.7.1 Description détaillée

Classe contenant les éléments de l'environnement et leurs positions (i.e Essaim, Obstacles, colis) Gère la détection de collision et le calcul de la position des drones. C'est le moteur physique du projet.

Auteurs

Timothé, Simon

4.7.2 Documentation des constructeurs et destructeur

4.7.2.1 Environnement() [1/2]

```
Environnement::Environnement ( )
```

Constructeur vide

4.7.2.2 Environnement() [2/2]

```
Environnement::Environnement (
            const float tailleCote )
```

Constructeur d'Environnement avec la taille d'un côté. Le reste sera ajouté via les méthodes prévues à cet effet.

Auteurs

Timothé, Simon, Théau

4.7.3 Documentation des fonctions membres

4.7.3.1 ajouterColis()

```
void Environnement::ajouterColis (
            const VecteurR3 & retrait,
            const VecteurR3 & depot )
```

Ajoute un colis dans la liste à récupérer

4.7.3.2 ajouterObstacle()

```
void Environnement::ajouterObstacle (
            const Obstacle & obs )
```

Ajoute directement un objet Obstacle à la liste de l'Environnement

4.7.3.3 associerEssaim()

```
void Environnement::associerEssaim (
            Essaim * e )
```

Donne sa valeur à l'attribut essaim une fois ce dernier créé (juste un set)

4.7.3.4 getEssaimDrones()

```
std::vector< Drone * > Environnement::getEssaimDrones ( ) const
```

Getter des Drones pour simplifier l'accès lors de l'affichage

4.7.3.5 operator++()

```
Environnement Environnement::operator++ (
            int a )
```

Surchage de l'opérateur ++, afin de passer au temps+1

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

4.8 Référence de la classe Essaim

```
#include <Essaim.h>
```

Fonctions membres publiques

- Essaim ()
- Essaim (Environnement &env, int nbDrones)
- void retirerColis (VecteurR3 retrait, VecteurR3 depot)
- void formation (Formation &F)
- void ajouterDrone (Drone &)
- std::vector< Drone * > getVDrones () const

4.8.1 Description détaillée

La classe Essaim est celle qui contient l'ensemble des drones Elle a pour objectif de contrôler leurs objectifs (Attribution des colis aux drones, formations, ...)

Auteurs

Timothé, Simon

4.8.2 Documentation des constructeurs et destructeur

4.8.2.1 Essaim() [1/2]

```
Essaim::Essaim ( )
```

Constructeur vide

4.8.2.2 Essaim() [2/2]

```
Essaim::Essaim (
            Environnement & env,
            int nbDrones )
```

Constructeur principal (utilisé dans le main de l'application) prenant l'Environnement et le nombre de Drones

4.8.3 Documentation des fonctions membres

4.8.3.1 ajouterDrone()

```
void Essaim::ajouterDrone (
            Drone & drone )
```

Ajoute un drone à l'essaim (dans le vector de drone)

4.8.3.2 formation()

```
void Essaim::formation (
            Formation & F )
```

Ordre aux drones de l'essaim de réaliser une formation

4.8.3.3 getVDrones()

```
vector< Drone * > Essaim::getVDrones ( ) const
```

getter du vector de Drone

4.8.3.4 retirerColis()

```
void Essaim::retirerColis (
            VecteurR3 retrait,
            VecteurR3 depot )
```

Ordre d'aller retirer un colis. Le drone qui doit aller le colis au point B est determiné dans le corps de la fonction et non passé en entrée

Paramètres

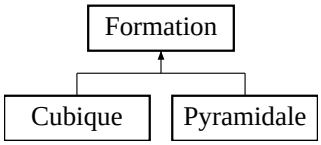
retrait	point de retrait du colis
depot	lieu où déposer le colis

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

4.9 Référence de la classe Formation

`#include <Formation.h>`

Graphe d'héritage de Formation:



Fonctions membres publiques

- Formation ()
- virtual ~Formation ()
- virtual vector< VecteurR3 > genererMaillage ()=0

Attributs protégés

- float altitude
- int nbDrones

4.9.1 Description détaillée

Classe abstraite correspondant à une figure géométrique aérienne que peut réaliser une parte ou l'ensemble de l'essaim.

Auteur

Margot, Théau et Morgan

Date

13/04/18

4.9.2 Documentation des constructeurs et destructeur

4.9.2.1 Formation()

```
Formation::Formation ( )
```

Constructeur inutilisable (classe abstraite)

4.9.2.2 ~Formation()

```
Formation::~Formation ( )  [virtual]
```

Destructeur inutilisable (classe abstraite)

4.9.3 Documentation des fonctions membres

4.9.3.1 genererMaillage()

```
virtual vector<VecteurR3> Formation::genererMaillage ( )  [pure virtual]
```

Méthode permettant de générer le maillage à partir des points et des contraintes de taille de Formation.

Renvoie

Retourne une nouvelle liste de vecteurs

Implémenté dans Cubique.

4.9.4 Documentation des données membres

4.9.4.1 altitude

```
float Formation::altitude  [protected]
```

Altitude de la formation

4.9.4.2 nbDrones

```
int Formation::nbDrones  [protected]
```

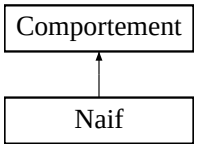
Le nombre de drones qui composent la formation

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

4.10 Référence de la classe Naif

`#include <Naif.h>`

Graphe d'héritage de Naif:



Fonctions membres publiques

- Naif (const VecteurR3, const VecteurR3)
- virtual VecteurR3 allerPoint (const VecteurR3 &posActuelle, const VecteurR3 &destination, const std::vector< Capteur > vCapteurs, const VecteurR3 vitesse) override
- bool **atteintFinal** (const VecteurR3 &posActuelle, const VecteurR3 &destination) const
- bool **atteint** (const VecteurR3 &posActuelle, const VecteurR3 &destination, const float &epsilon) const
- bool **presenceObstacles** (const VecteurR3 posActuelle, const VecteurR3 destination, const std::vector< Capteur > vCapteurs, const VecteurR3 &)

Attributs publics

- VecteurR3 depart
- VecteurR3 dest
- VecteurR3 v0

4.10.1 Description détaillée

Classe héritant de Comportement, c'est donc un algo possible de déplacement des drones. Il consiste à monter en altitude lorsque le drone rencontre un obstacle devant lui afin de passer au dessus.

Auteur

Simon

4.10.2 Documentation des constructeurs et destructeur

4.10.2.1 Naif()

```
Naif::Naif (
            const VecteurR3 _depart,
            const VecteurR3 _v0 )
```

Initialise le point duquel le Drone part et sa vitesse

4.10.3 Documentation des fonctions membres

4.10.3.1 allerPoint()

```
VecteurR3 Naif::allerPoint (
            const VecteurR3 & posActuelle,
            const VecteurR3 & destination,
            const std::vector< Capteur > vCapteurs,
            const VecteurR3 vitesse )  [override], [virtual]
```

Méthode fondamentale de Comportement des Drones. A partir des positions du Drone, de son premier objectif et des capteurs, dtermine le vecteur acclration pour la frame suivante.

Paramètres

posActuelle	la position du Drone au temps t.
destination	la position atteindre.
vCapteurs,le	vecteur des Capteurs donnant l'information sensorielle du Drone.

Renvoie

le vecteur acclration

Implémente Comportement.

4.10.4 Documentation des données membres

4.10.4.1 depart

```
VecteurR3 Naif::depart
```

Position à laquelle le Drone a commencé à aller à l'objectif. Permet de réduire sa vitesse avant d'atteindre l'objectif.

4.10.4.2 dest

```
VecteurR3 Naif::dest
```

Destination à atteindre. Permet de réinitialiser le depart en cas de changement d'objectif.

4.10.4.3 v0

```
VecteurR3 Naif::v0
```

Vitesse à laquelle le drone est parti

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

4.11 Référence de la classe Obstacle

Fonctions membres publiques

- Obstacle (VecteurR3 init, float taillex, float tailley, float taillez)
- vector< VecteurR3 > getVSommets () const
- vector< VecteurR3 > getFaceYMin () const
- vector< VecteurR3 > getFaceYMax () const
- vector< VecteurR3 > getFaceXMin () const
- vector< VecteurR3 > getFaceXMax () const
- vector< VecteurR3 > getFaceZMin () const
- vector< VecteurR3 > getFaceZMax () const
- VecteurR3 getCentre () const
- VecteurR3 getInit () const
- float getTailleX () const
- float getTailleY () const
- float getTailleZ () const

4.11.1 Documentation des constructeurs et destructeur

4.11.1.1 Obstacle()

```
Obstacle::Obstacle (
            VecteurR3 init,
            float taillex,
            float tailley,
            float taillez )
```

Constructeur de pavé droit Ordre : base en horaire; haut en horaire (sens de rotation)

Paramètres

init	la position du coin aux coordonnées x,y,z minimales

4.11.2 Documentation des fonctions membres

4.11.2.1 getCentre()

```
VecteurR3 Obstacle::getCentre ( ) const
```

Getter centre

4.11.2.2 getFaceXMax()

```
vector< VecteurR3 > Obstacle::getFaceXMax ( ) const
```

Getter face avant, X constant. (4 sommets)

4.11.2.3 getFaceXMin()

```
vector< VecteurR3 > Obstacle::getFaceXMin ( ) const
```

Getter face arriere, X constant. (4 sommets)

4.11.2.4 getFaceYMax()

```
vector< VecteurR3 > Obstacle::getFaceYMax ( ) const
```

Getter face droite, Y constant. (4 sommets)

4.11.2.5 getFaceYMin()

```
vector< VecteurR3 > Obstacle::getFaceYMin ( ) const
```

Getter face gauche, Y constant. (4 sommets)

4.11.2.6 getFaceZMax()

```
vector< VecteurR3 > Obstacle::getFaceZMax ( ) const
```

Getter face haute, Z constant. (4 sommets)

4.11.2.7 getFaceZMin()

```
vector< VecteurR3 > Obstacle::getFaceZMin ( ) const
```

Getter face basse, Z constant. (4 sommets)

4.11.2.8 getInit()

```
VecteurR3 Obstacle::getInit ( ) const
```

Getter point inital (coordonnées x,y,z minimales)

4.11.2.9 getTailleX()

```
float Obstacle::getTailleX ( ) const
```

Getter taille de cote en x

4.11.2.10 getTailleY()

```
float Obstacle::getTailleY ( ) const
```

Getter taille de cote en y

4.11.2.11 getTailleZ()

```
float Obstacle::getTailleZ ( ) const
```

Getter taille de cote en z

4.11.2.12 getVSommets()

```
vector< VecteurR3 > Obstacle::getVSommets ( ) const
```

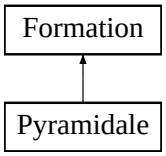
Renvoie le vector des sommets

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

4.12 Référence de la classe Pyramidale

```
#include <Pyramidale.h>
```

Graphe d'héritage de Pyramidale:



Attributs protégés

- vector< VecteurR3 > vPointsBase
- VecteurR3 sommet

Membres hérités additionnels

4.12.1 Description détaillée

Classe fille de Formation; dessine une pyramide.

Auteur

Margot, Théau et Morgan

Date

13/04/18

4.12.2 Documentation des données membres

4.12.2.1 sommet

```
VecteurR3 Pyramidale::sommet  [protected]
```

Point sommet de la pyramide

4.12.2.2 vPointsBase

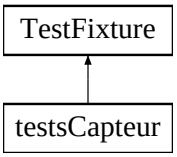
```
vector<VecteurR3> Pyramidale::vPointsBase  [protected]
```

Points formant la base de la pyramide

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

4.13 Référence de la classe testsCapteur

Graphe d'héritage de testsCapteur:



Fonctions membres publiques

- void **setUp** (void)
- void **tearDown** (void)

Fonctions membres protégées

— void testUpdateDistanceDetectee (void)

4.13.1 Documentation des fonctions membres

4.13.1.1 testUpdateDistanceDetectee()

```
void testsCapteur::testUpdateDistanceDetectee (
            void  )  [protected]
```

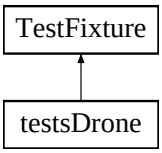
Je lance la detection

Je vérifie que mon capteur detecte bien l'obstacle

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

4.14 Référence de la classe testsDrone

Graphe d'héritage de testsDrone:



Fonctions membres publiques

— void **setUp** (void)
— void **tearDown** (void)

Fonctions membres protégées

— void testAjouterObjectif ()
— void testLivrerColis ()
— void testAtteintObjectif ()
— void testplusplus ()

4.14.1 Documentation des fonctions membres

4.14.1.1 testAjouterObjectif()

```
void testsDrone::testAjouterObjectif ( )  [protected]
```

Teste l'ajout d'un objectif dans la liste des points à atteindre du Drone. ASSERT si l'objectif a bien été ajouté; c'est-à-dire que la liste est plus grande d'un élément, qui est celui affiché. La vérification de la validité du point n'est pas du ressort du Drone (et donc de cette fonction).

4.14.1.2 testAtteintObjectif()

```
void testsDrone::testAtteintObjectif ( )  [protected]
```

Teste si l'objectif est bien considéré comme atteint et bien supprimé de la liste

4.14.1.3 testLivrerColis()

```
void testsDrone::testLivrerColis ( )  [protected]
```

Teste l'ordre de livraison de colis. Réalise globalement les mêmes tests que testAjouterObjectif, sur deux points.

Voir également

testAjouterObjectif

4.14.1.4 testplusplus()

```
void testsDrone::testplusplus ( )  [protected]
```

Teste si la position est bien actualisé

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

4.15 Référence de la classe testsEnvironnement

Fonctions membres publiques

- void **setup** (void)
- void **tearDown** (void)

Fonctions membres protégées

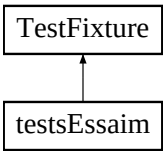
- void **testcalculerPos** (void)
- void **testcolision** (void)

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

4.16 Référence de la classe testsEssaim

`#include <testsEssaim.h>`

Graphe d'héritage de testsEssaim:



Fonctions membres publiques

- void **setUp** (void)
- void **tearDown** (void)

Fonctions membres protégées

- void **testAjouterDrone** ()
- void testRetirerColis ()
- void testAffectationDronePos ()

4.16.1 Description détaillée

classe de test pour la classe Essaim

Auteur

Simon

4.16.2 Documentation des fonctions membres

4.16.2.1 testAffectationDronePos()

`void testsEssaim::testAffectationDronePos ( ) [protected]`

Appelle la fonction et vérifie si le bon Drone est bien dans la liste

4.16.2.2 testRetirerColis()

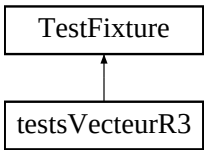
`void testsEssaim::testRetirerColis ( ) [protected]`

teste si la mthode affecterDronePos affecte au drone le noeud du maillage le plus proche de sa position C'est une mthode fastidieuse tester. On va tester un exemple simple. Etant donn 8 noeuds issus d'un maillage d'une formation cubique, on cr 8 drones lgrement dcal au 8 sommets. On vrifie que chaque drone a comme objectif d'aller au sommet cot de lui. teste la mthode retirer colis. On va faire apparaitre un colis a un endroit et on vrifie que le drone le plus proche a bien reu l'ordre de s'y rendre. Pour cela on regarde que les points de retrait et dpose ont t ajout la liste des objectifs du drone le plus proche.

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

4.17 Référence de la classe testsVecteurR3

Graphe d'héritage de testsVecteurR3:



Fonctions membres publiques

- void **setUp** (void)
- void **tearDown** (void)

Fonctions membres protégées

- void **testEgalite** (void)
- void **testAddition** (void)
- void **testSoustraction** (void)
- void **testMulti** (void)
- void **testDiv** (void)
- void **testProdScal** (void)
- void **testMultiplicationScalaire** (void)
- void **testIncrementation** (void)
- void **testNorme22** (void)
- void **testprodVec** (void)
- void **testValeurAbsolue** (void)
- void **testReflexionPlanOrtho** (void)

La documentation de cette classe a été générée à partir des fichiers suivants :
- testsVecteurR3.h
- testsVecteurR3.cpp

4.18 Référence de la classe TrackBallCamera

```
#include <trackballcamera.h>
```

Fonctions membres publiques

- virtual void **OnMouseMotion** (const SDL_MouseMotionEvent &event)
- virtual void **OnMouseButton** (const SDL_MouseButtonEvent &event)
- virtual void **OnKeyboard** (const SDL_KeyboardEvent &event)
- virtual void **look** ()
- virtual void **setMotionSensivity** (double sensivity)
- virtual void **setScrollSensivity** (double sensivity)

Attributs protégés

- double **_motionSensivity**
- double **_scrollSensivity**
- bool **_hold**
- double **_distance**
- double **_angleY**
- double **_angleZ**
- SDL_Cursor * **_hand1**
- SDL_Cursor * **_hand2**

4.18.1 Description détaillée

Class gestionnaire de tous les mouvements de la caméra; permet d'effectuer des rotations autour du centre à la souris. Récupéré d'Openclassrooms.

Auteur

Louis, Openclassrooms

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

4.19 Référence de la classe VecteurR3

`#include <VecteurR3.h>`

Fonctions membres publiques

- VecteurR3 ()
- VecteurR3 (const float &x, const float &y, const float &z)
- virtual ∼VecteurR3 ()
- float **getX** () const
- float **getY** () const
- float **getZ** () const
- void **setX** (const float &)
- void **setY** (const float &)
- void **setZ** (const float &)
- float operator[ ] (const int &) const
- bool **operator==** (const VecteurR3 &vComp) const
- bool egal (const VecteurR3 &vComp, const float &epsilon=0) const
- VecteurR3 operator+ (const VecteurR3 &) const
- VecteurR3 operator- (const VecteurR3 &) const
- VecteurR3 div (const VecteurR3 &) const
- VecteurR3 multi (const VecteurR3 &) const
- void operator= (const VecteurR3 &)
- void operator+= (const VecteurR3 &)
- void operator-= (const VecteurR3 &)
- float operator∗ (const VecteurR3 &) const
- VecteurR3 operator∗ (const float &) const
- float norme22 () const
- float norme2 () const
- VecteurR3 prodVec (const VecteurR3 &) const
- VecteurR3 valeurAbsolue () const
- VecteurR3 reflexionPlanOrtho (const VecteurR3 &) const

Amis

- std::ostream & operator<< (std::ostream &os, const VecteurR3 &)

4.19.1 Description détaillée

Classe d'un vecteur dans R3, avec trois coordonnées et les opérations classiques des ensembles vectoriels.

Auteurs

: Margot, Morgan, Théau, Louis

Version

1.0 @13 avril 2018

4.19.2 Documentation des constructeurs et destructeur

4.19.2.1 VecteurR3() [1/2]

```
VecteurR3::VecteurR3 ( )
```

Constructeur de VecteurR3 initilisant les coordonnées à l'origine.

4.19.2.2 VecteurR3() [2/2]

```
VecteurR3::VecteurR3 (
            const float & x,
            const float & y,
            const float & z )
```

Constructeur de VecteurR3 à partir de trois coordonnées données.

4.19.2.3 ~VecteurR3()

```
VecteurR3::~VecteurR3 ( )  [virtual]
```

Destructeur d'un VecteurR3.

4.19.3 Documentation des fonctions membres

4.19.3.1 div()

```
VecteurR3 VecteurR3::div (
            const VecteurR3 & v ) const
```

division de deux vecteurs composante par composante

4.19.3.2 egal()

```
bool VecteurR3::egal (
            const VecteurR3 & vComp,
            const float & epsilon = 0 ) const
```

Comparaison de deux vecteurs à un voisinage de rayon donné près

Paramètres

vComp	le VecteurR3 auquel se comparer
epsilon	la marge d'erreur que l'on se laisse

Renvoie

si le vecteur est bien le même que celui en entrée, à une précision epsilon

4.19.3.3 multi()

```
VecteurR3 VecteurR3::multi (
            const VecteurR3 & v ) const
```

multiplication de deux vecteurs composante par composante

4.19.3.4 norme2()

```
float VecteurR3::norme2 ( ) const
```

Norme (ou distance à l'origine) du vecteur. Calcule simplement la racine de norme22.

4.19.3.5 norme22()

```
float VecteurR3::norme22 ( ) const
```

Norme AU CARRE du vecteur (pour optimisation, lorsque la distance même n'est pas nécessaire)

4.19.3.6 operator∗() [1/2]

```
float VecteurR3::operator* (
            const VecteurR3 & v ) const
```

Produit scalaire de ce vecteur avec un autre

4.19.3.7 operator∗() [2/2]

```
VecteurR3 VecteurR3::operator* (
            const float & scal ) const
```

Multiplication d'un vecteur par un scalaire

4.19.3.8 operator+()

```
VecteurR3 VecteurR3::operator+ (
            const VecteurR3 & v ) const
```

Addition de deux vecteurs composante par composante

4.19.3.9 operator+=()

```
void VecteurR3::operator+= (
            const VecteurR3 & v )
```

Addition des coordonnées actuelles avec celles d'un autre (raccourci +=)

4.19.3.10 operator-()

```
VecteurR3 VecteurR3::operator- (
            const VecteurR3 & v ) const
```

Soustraction de deux vecteurs composante par composante

4.19.3.11 operator-=()

```
void VecteurR3::operator-= (
            const VecteurR3 & v )
```

Soustraction des coordonnées actuelles avec celles d'un autre (raccourci -=)

4.19.3.12 operator=()

```
void VecteurR3::operator= (
            const VecteurR3 & v )
```

Affectation d'un vecteur à partir d'un autre

4.19.3.13 operator[]()

```
float VecteurR3::operator[] (
            const int & index ) const
```

Alternative aux getters : operateur []

4.19.3.14 prodVec()

```
VecteurR3 VecteurR3::prodVec (
            const VecteurR3 & v ) const
```

Calcul du produit vectoriel. (Useful pour verifier la colinearite).

4.19.3.15 reflexionPlanOrtho()

```
VecteurR3 VecteurR3::reflexionPlanOrtho (
            const VecteurR3 & v ) const
```

Calcul le produit de la matrice de Householder générée par ce VecteurR3 (this) avec un vecteur donné en entrée. Cela donne le vecteur de reflexion par rapport au plan orthogonal à ce vecteur.

4.19.3.16 valeurAbsolue()

```
VecteurR3 VecteurR3::valeurAbsolue ( ) const
```

Ressort la valeur absolue composante par composante

4.19.4 Documentation des fonctions amies et associées

4.19.4.1 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const VecteurR3 & v )  [friend]
```

Methode d'affichage pour debug

La documentation de cette classe a été générée à partir des fichiers suivants :
- VecteurR3.h
- VecteurR3.cpp

Chapitre 5

Documentation des fichiers

5.1 Référence du fichier main.cpp

```
#include <SDL/SDL.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <cstdlib>
#include <list>
#include <cppunit/TestCase.h>
#include <cppunit/TestFixture.h>
#include <cppunit/ui/text/TextTestRunner.h>
#include <cppunit/extensions/HelperMacros.h>
#include <cppunit/extensions/TestFactoryRegistry.h>
#include <cppunit/TestResult.h>
#include <cppunit/TestResultCollector.h>
#include <cppunit/TestRunner.h>
#include <cppunit/BriefTestProgressListener.h>
#include <cppunit/CompilerOutputter.h>
#include <cppunit/XmlOutputter.h>
#include <netinet/in.h>
#include "../include/trackballcamera.h"
#include "../include/sdlglutils.h"
#include "../include/Environnement.h"
#include "../include/Essaim.h"
#include "../include/Affichage.h"
#include "../tests/testsVecteurR3.h"
#include "../tests/testsEssaim.h"
#include "../tests/testsDrone.h"
```

Macros

- #define **TEST** true
- #define **FPS** 50
- #define **LARGEUR_FENETRE** 1366
- #define **HAUTEUR_FENETRE** 700

Fonctions

- **CPPUNIT_TEST_SUITE_REGISTRATION** (testsVecteurR3)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (testsEssaim)
- **CPPUNIT_TEST_SUITE_REGISTRATION** (testsDrone)
- void **stop** ()
- int **main** (int argc, char ∗argv[ ])

Variables

- TrackBallCamera ∗ **camera**

5.1.1 Description détaillée

Fichier contenant la fonction main, à lancer pour démarrer l'application Drone