Drones

Generated by Doxygen 1.8.14

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 Affichage Class Reference

`#include <Affichage.h>`

Public Member Functions

- Affichage (const Environnement &env)
- virtual ~Affichage ()
- void draw ()

4.1.1 Detailed Description

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

Authors

Timothé, Simon

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Affichage()

```
Affichage::Affichage (
            const Environnement & env )
```

Un constructeur utilisant un environnement pour s'y lier par pointeur.

Parameters

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 Member Function Documentation

4.1.3.1 draw()

```
void Affichage::draw ( )
```

Méthode principale, affichant l'Environnement en attribut

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

- Affichage.h
- Affichage.cpp

4.2 Capteur Class Reference

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

Public Member Functions

- Capteur (const float &p, const VecteurR3 &dir, const Environnement &environnement)
- virtual ~Capteur ()
- float updateDistanceDetectee ()

4.2.1 Detailed Description

Authors

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 Constructor & Destructor Documentation

4.2.2.1 Capteur()

```
Capteur::Capteur (
            const float & p,
            const VecteurR3 & dir,
            const 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 Member Function Documentation

4.2.3.1 updateDistanceDetectee()

```
float Capteur::updateDistanceDetectee ( )
```

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

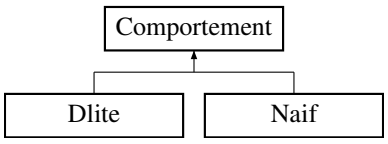
The documentation for this class was generated from the following files:

- Capteur.h
- Capteur.cpp

4.3 Comportement Class Reference

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

Inheritance diagram for Comportement:



Public Member Functions

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

4.3.1 Detailed Description

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

Author

Louis

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Comportement()

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

Constructeur vide.

4.3.2.2 ~Comportement()

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

Destructeur vide.

4.3.3 Member Function Documentation

4.3.3.1 allerPoint()

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

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.

Parameters

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

Returns

le vecteur accélération

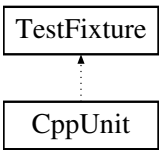
The documentation for this class was generated from the following files:

- Comportement.h
- Comportement.cpp

4.4 CppUnit Class Reference

`#include <testsCubique.h>`

Inheritance diagram for CppUnit:



Public Member Functions

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

Protected Member Functions

- void testsGenererMaillage ()

4.4.1 Detailed Description

classe de test de la classe Cubique

Author

Simon

4.4.2 Member Function Documentation

4.4.2.1 testsGenererMaillage()

```
void CppUnit::testsGenererMaillage ( )  [protected]
```

A partir des informations relatives formation cubique (nbre drones, longeur cot origine On verifie que le maillage est celui attendu. Plus prsement on va cr une formation origine du rep, de longueur 1 pour 8 drones. On vfie alors que le retour de la fonction est les 8 sommets du cube.

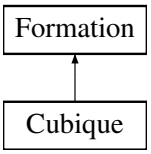
The documentation for this class was generated from the following file:

- testsCubique.h

4.5 Cubique Class Reference

```
#include <Cubique.h>
```

Inheritance diagram for Cubique:



Public Member Functions

- Cubique (VecteurR3, float, int)
- virtual ~Cubique ()
- virtual vector< VecteurR3 > genererMaillage ()

Protected Attributes

- float longueurCote
- VecteurR3 origine

4.5.1 Detailed Description

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

Author

Margot, Théau et Morgan

Date

13/04/18

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Cubique()

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

Constructeur de la Formation.

4.5.2.2 ~Cubique()

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

Destructeur usuel de la Formation.

4.5.3 Member Function Documentation

4.5.3.1 genererMaillage()

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

Méthode héritée, calcule le maillage adapté à la formation cubique

Implements Formation.

4.5.4 Member Data Documentation

4.5.4.1 longueurCote

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

Longueur du côté du cube

4.5.4.2 origine

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

Orgine du cube

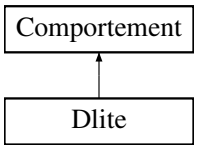
The documentation for this class was generated from the following files:

- Cubique.h
- Cubique.cpp

4.6 Dlite Class Reference

```
#include <Dlite.h>
```

Inheritance diagram for Dlite:



Public Member Functions

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

4.6.1 Detailed Description

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

4.6.2 Constructor & Destructor Documentation

4.6.2.1 Dlite()

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

Constructeur de l'algorithme.

4.6.2.2 ~Dlite()

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

Destructeur de l'algorithme.

4.6.3 Member Function Documentation

4.6.3.1 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.

Parameters

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

Returns

le vecteur accélération

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

- Dlite.h
- Dlite.cpp

4.7 Drone Class Reference

```
#include <Drone.h>
```

Public Member Functions

- Drone (std::vector< Capteur >)
- Drone (std::vector< Capteur >, const VecteurR3)
- Drone (std::vector< Capteur >, const std::vector< VecteurR3 >)
- Drone (std::vector< Capteur >, const float &, const Comportement &, const std::vector< VecteurR3 >)

- Drone (std::vector< Capteur >, const float &, const Comportement &, const std::vector< VecteurR3 >, const VecteurR3)
- Drone (std::vector< Capteur >, const float &, const Comportement &, const std::vector< VecteurR3 >, const VecteurR3, const VecteurR3)
- virtual ~Drone ()
- std::vector< VecteurR3 > getObjectifs () const
- std::vector< Capteur > getvCapteurs () const
- void ajouterObjectif (const VecteurR3 &obj)
- void livrerColis (const VecteurR3 &retrait, const VecteurR3 &depot)

4.7.1 Detailed Description

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.

Author

Louis, Quentin

4.7.2 Constructor & Destructor Documentation

4.7.2.1 Drone() [1/6]

```
Drone::Drone (
            std::vector< Capteur > _vCapteurs )
```

Constructeur de Drone pour tests

4.7.2.2 Drone() [2/6]

```
Drone::Drone (
            std::vector< Capteur > _vCapteurs,
            const VecteurR3 pos )
```

Constructeur avec simplement la position initiale

4.7.2.3 Drone() [3/6]

```
Drone::Drone (
            std::vector< Capteur > ,
            const std::vector< VecteurR3 >  )
```

Constructeur de Drone pour tests : avec direction Capteurs

4.7.2.4 Drone() [4/6]

```
Drone::Drone (
            std::vector< Capteur > ,
            const float & ,
            const Comportement & ,
            const std::vector< VecteurR3 >  )
```

Constructeur de Drone, initialisant les attributs spatio-temporels à 0. Nécessite un Comportement, une taille, un nombre de capteurs

4.7.2.5 Drone() [5/6]

```
Drone::Drone (
            std::vector< Capteur > ,
            const float & ,
            const Comportement & ,
            const std::vector< VecteurR3 > ,
            const VecteurR3  )
```

Constructeur de Drone, initialisant la position à celle demandée.

4.7.2.6 Drone() [6/6]

```
Drone::Drone (
            std::vector< Capteur > ,
            const float & ,
            const Comportement & ,
            const std::vector< VecteurR3 > ,
            const VecteurR3 ,
            const VecteurR3  )
```

Constructeur de Drone, initialisant la position et la vitesse à celles demandées.

4.7.2.7 ~Drone()

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

Destructeur de Drone.

4.7.3 Member Function Documentation

4.7.3.1 ajouterObjectif()

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

Méthode qui ajoute une destination à la liste des objectifs.

Parameters

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

4.7.3.2 getObjectifs()

```
std::vector<VecteurR3> Drone::getObjectifs ( ) const
```

Getter des objectifs du Drone

4.7.3.3 getvCapteurs()

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

Getter du vecteur de capteurs

4.7.3.4 livrerColis()

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

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

Parameters

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

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

- Drone.h
- Drone.cpp

4.8 Environnement Class Reference

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

Public Member Functions

- Environnement (const float)
- Environnement (const std::vector< Obstacle > &, const Essaim &, const float &)
- void operator++ ()
- void **ajouterObstacle** (std::vector< VecteurR3 > &positionObstacle)

4.8.1 Detailed Description

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.

Authors

Timothé, Simon

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Environnement() [1/2]

```
Environnement::Environnement (
            const float _g )
```

Constructeur vide

Authors

Timothé, Simon

4.8.2.2 Environnement() [2/2]

```
Environnement::Environnement (
            const std::vector< Obstacle > & ,
            const Essaim & ,
            const float &  )
```

Constructeur principal de l'Environnement

4.8.3 Member Function Documentation

4.8.3.1 operator++()

```
void Environnement::operator++ ( )
```

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

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

- Environnement.h
- Environnement.cpp

4.9 Essaim Class Reference

`#include <Essaim.h>`

Public Member Functions

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

4.9.1 Detailed Description

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, ...)

Authors

Timothé, Simon

4.9.2 Member Function Documentation

4.9.2.1 ajouterDrone()

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

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

4.9.2.2 formation()

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

Ordre aux drones de l'essaim de réaliser une formation

4.9.2.3 getVDrones()

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

getter du vector de Drone

4.9.2.4 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

Parameters

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

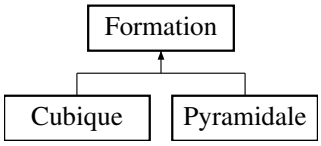
The documentation for this class was generated from the following files:

- Essaim.h
- Essaim.cpp

4.10 Formation Class Reference

`#include <Formation.h>`

Inheritance diagram for Formation:



Public Member Functions

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

Protected Attributes

- float altitude
- int nbDrones

4.10.1 Detailed Description

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

Author

Margot, Théau et Morgan

Date

13/04/18

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Formation()

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

Constructeur inutilisable (classe abstraite)

4.10.2.2 ~Formation()

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

Destructeur inutilisable (classe abstraite)

4.10.3 Member Function Documentation

4.10.3.1 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.

Returns

Retourne une nouvelle liste de vecteurs

Implemented in Cubique.

4.10.4 Member Data Documentation

4.10.4.1 altitude

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

Altitude de la formation

4.10.4.2 nbDrones

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

Le nombre de drones qui composent la formation

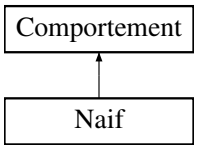
The documentation for this class was generated from the following files:

- Formation.h
- Formation.cpp

4.11 Naif Class Reference

```
#include <Naif.h>
```

Inheritance diagram for Naif:



Public Member Functions

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

4.11.1 Detailed Description

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

Author

Simon

4.11.2 Member Function Documentation

4.11.2.1 allerPoint()

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

Mode fondamentale de Comportement des Drones. A partir des positions du Drone, de son premier objectif et des capteurs, drmine le vecteur accration pour la frame suivante.

Parameters

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

Returns

le vecteur accration

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

- Naif.h
- Naif.cpp

4.12 Obstacle Class Reference

Public Member Functions

- **Obstacle** (vector< VecteurR3 >)
- vector< VecteurR3 > getVSommets ()

4.12.1 Member Function Documentation

4.12.1.1 getVSommets()

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

Renvoie le vector des sommets

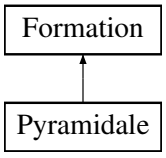
The documentation for this class was generated from the following files:

- Obstacle.h
- Obstacle.cpp

4.13 Pyramidale Class Reference

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

Inheritance diagram for Pyramidale:



Protected Attributes

- vector< VecteurR3 > vPointsBase
- VecteurR3 sommet

Additional Inherited Members

4.13.1 Detailed Description

Classe fille de Formation; dessine une pyramide.

Author

Margot, Théau et Morgan

Date

13/04/18

4.13.2 Member Data Documentation

4.13.2.1 sommet

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

Point sommet de la pyramide

4.13.2.2 vPointsBase

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

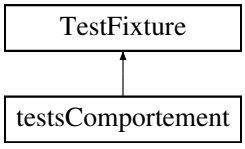
Points formant la base de la pyramide

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

- Pyramidale.h
- Pyramidale.cpp

4.14 testsComportement Class Reference

Inheritance diagram for testsComportement:



Public Member Functions

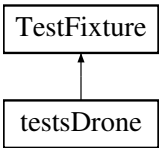
- void **setUp** (void)
- void **testAllerPoint** (void)
- void **tearDown** (void)

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

- testsComportement.h
- testsComportement.cpp

4.15 testsDrone Class Reference

Inheritance diagram for testsDrone:



Public Member Functions

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

Protected Member Functions

- void testAjouterObjectif (void)
- void testLivrerColis (void)
- void testUpdateCapteurs (void)

4.15.1 Member Function Documentation

4.15.1.1 testAjouterObjectif()

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

Teste l'ajout d'un objectif

4.15.1.2 testLivrerColis()

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

Teste l'ordre de livraison de colis

4.15.1.3 testUpdateCapteurs()

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

Teste la modification des capteurs

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

- testsDrone.h
- testsDrone.cpp

4.16 testsEnvironnement Class Reference

Public Member Functions

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

Protected Member Functions

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

4.16.1 Member Function Documentation

4.16.1.1 testcalculerPos()

```
void testsEnvironnement::testcalculerPos (
            void  )  [protected]
```

On va d'abord crée un environnement vide. Dans lequel on va créée un essaim composé de 1 drone seul. On va donner pour ordre au drone d'aller à une position puis une fois arrivé, on calculera la position du drone et vérifiera qu'elle coincide avec celle demandée.

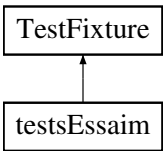
The documentation for this class was generated from the following files:

- testsEnvironnement.h
- testsEnvironnement.cpp

4.17 testsEssaim Class Reference

`#include <testsEssaim.h>`

Inheritance diagram for testsEssaim:



Public Member Functions

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

Protected Member Functions

- void testRetirerColis ()
- void testAffectationDronePos (vector< VecteurR3 >) vector< int >

4.17.1 Detailed Description

classe de test pour la classe Essaim

Author

Simon

4.17.2 Member Function Documentation

4.17.2.1 testAffectationDronePos()

```
void testsEssaim::testAffectationDronePos (
            vector< VecteurR3 > ) [protected]
```

teste si la mode affecterDronePos affecte au drone le noeud du maillage le plus proche de sa position C'est une mode fastidieuse ster. On va tester un exemple simple. Etant donnoeuds issus d'un maillage d'une formation cubique, on cr drones lrement dl 8 sommets. On vfie que chaque drone a comme objectif d'aller au sommet t lui.

4.17.2.2 testRetirerColis()

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

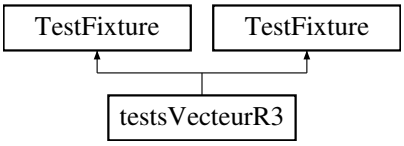
teste la mode retirer colis. On va faire apparaitre un colis a un endroit et on vfie que le drone le plus proche a bien re'ordre de s'y rendre. Pour cela on regarde que les points de retrait et dse ont ajouta liste des objectifs du drone le plus proche.

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

- testsEssaim.h
- testsEssaim.cpp

4.18 testsVecteurR3 Class Reference

Inheritance diagram for testsVecteurR3:



Public Member Functions

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

Protected Member Functions

- void **testUpdateDistanceDetectee** (void)
- void **testAddition** (void)
- void **testSoustraction** (void)
- void **testAffectation** (void)
- void **testMultiplication** (void)
- void **testMultiplicationScalaire** (void)
- void **testIncrementation** (void)
- void **testNorme22** (void)
- void **testprodVec** (void)

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

- testsCapteur.h
- testsVecteurR3.h
- testsVecteurR3.cpp

4.19 TrackBallCamera Class Reference

`#include <trackballcamera.h>`

Public Member Functions

- 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)

Protected Attributes

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

4.19.1 Detailed Description

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

Author

Louis, Openclassrooms

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

- trackballcamera.h
- trackballcamera.cpp

4.20 VecteurR3 Class Reference

`#include <VecteurR3.h>`

Public Member Functions

- VecteurR3 ()
- VecteurR3 (const float &x, const float &y, const float &z)
- virtual ~VecteurR3 ()
- float **getX** () const
- float **getY** () const
- float **getZ** () const
- 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
- 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

4.20.1 Detailed Description

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

Authors

: Margot, Morgan, Théau, Louis

Version

1.0 @13 avril 2018

4.20.2 Constructor & Destructor Documentation

4.20.2.1 VecteurR3() [1/2]

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

Constructeur de VecteurR3 initilisant les coordonnées à l'origine.

4.20.2.2 VecteurR3() [2/2]

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

Constructeur de VecteurR3 à partir de trois coordonnées données.

4.20.2.3 ~VecteurR3()

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

Destructeur d'un VecteurR3.

4.20.3 Member Function Documentation

4.20.3.1 egal()

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

Comparaison de deux vecteurs à un voisinage de rayon donné près

Parameters

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

Returns

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

4.20.3.2 norme2()

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

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

4.20.3.3 norme22()

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

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

4.20.3.4 operator*() [1/2]

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

Produit scalaire de ce vecteur avec un autre

4.20.3.5 operator*() [2/2]

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

Multiplication d'un vecteur par un scalaire

4.20.3.6 operator+()

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

Addition de deux vecteurs composante par composante

4.20.3.7 operator+=()

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

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

4.20.3.8 operator-()

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

Soustraction de deux vecteurs composante par composante

4.20.3.9 operator=()

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

Affectation d'un vecteur à partir d'un autre

4.20.3.10 operator[]()

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

Alternative aux getters : operateur []

4.20.3.11 prodVec()

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

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

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

- VecteurR3.h
- VecteurR3.cpp

Chapter 5

File Documentation

5.1 main.cpp File Reference

```
#include <SDL/SDL.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <cstdlib>
#include "../include/sdlglutils.h"
#include "../include/trackballcamera.h"
```

Macros

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

Functions

- void **DrawGL** ()
- void **stop** ()
- int **main** (int argc, char ∗argv[ ])

Variables

- GLuint **droneText**
- TrackBallCamera ∗ **camera**

5.1.1 Detailed Description

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