Simulation Multi-Agents

Généré par Doxygen 1.8.6

Jeudi 3 Mars 2016 18 :47 :04

Table des matières

Chapitre 1

Liste des choses à faire

Membre Faction : :init ()
Vérifier si la place n'est pas déja occupée par une autre planète mère

Chapitre 2

Index hiérarchique

2.1 Hiérarchie des classes

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

Chapitre 3

Index des classes

3.1 Liste des classes

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

Chapitre 4

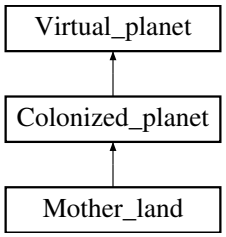
Documentation des classes

4.1 Référence de la classe Colonized_planet

Planète colonisée.

`#include <Colonized_planet.hpp>`

Graphe d'héritage de Colonized_planet :



Fonctions membres publiques

- Colonized_planet (World &, unsigned, unsigned, Faction &)
  Constructeur, prépare la planète.
- Colonized_planet (Virtual_planet ∗, Faction &)
  Création d'une planète colonisée à partir d'une autre planète.
- void update_neighbourhood (Virtual_planet ∗, Virtual_planet ∗)
  Mise à jour du voisinage.
- bool attack (Virtual_planet ∗)
  Attaque une planète.
- virtual bool is_attacked (Virtual_planet ∗)
  La planète est attaquée.
- void convert_to_free_planet ()
  Transforme la colonie courante en free planete.
- Faction & get_faction ()
  Faction à laquelle appartient la planète.
- double get_defense ()
  Défense totale de la planète.
- double get_production ()
  Production de la planète.
- Virtual_planet ∗ **get_target** ()
- void demand_to_faction (double)
  Réalise une demande de fonds auprès de la faction.
- double estimate_cost ()
  Estime le coût d'une attaque.
- void add_to_budget (double)
  Ajoute les fonds obtenus auprès de la faction.
- void reinitialisate_target ()
  Réinitialise la cible, annule la tentative d'attaque.

- char display ()
  Caractère par défaut (point)
- std : :string get_color_name ()
  Couleur par défaut (gris)
- void run ()
  Joue le tour.
- bool can_be_replaced ()
  Renvoie true si la planete n'est pas un agent.

Attributs protégés

- double colony_defense_
  Valeur de défense.
- double colony_production_
  Valeur de production de monnaie.
- Faction & faction_
  Faction à laquelle appartient la colonie.
- Virtual_planet * target_
  Planète ciblée par une éventuelle attaque.
- double budget_
  Argent disponible.
- double demand_
  Coût estimé d'une opération.

Attributs protégés statiques

- static const int MAX_COLONY_DEFENSE = 50
  Valeur de défense maximum.
- static const int MAX_COLONY_PRODUCTION = 15
  Valeur de production maximum.
- static const int COLONY_MULTIPLICATOR = 10
  Facteur de production.

Membres hérités additionnels

4.1.1 Description détaillée

Planète colonisée.

Une planète colonisée est une planète possédée par une faction

4.1.2 Documentation des constructeurs et destructeur

4.1.2.1 Colonized_planet : :Colonized_planet (World & *world,* unsigned *pos_x,* unsigned *pos_y,* Faction & *fac*)

Constructeur, prépare la planète.

Met à jour la liste des agents en attente et ajoute la planète à la liste des colonies de la faction.

Paramètres

world	plateau de jeu
pos_x	numéro de la ligne sur la grille

pos_y	numéro de la colonne sur la grille
fac	faction à laquelle doit appartenir la nouvelle planète

4.1.2.2 Colonized_planet : :Colonized_planet (**Virtual_planet** ∗ *fp,* **Faction** & *faction*)

Création d'une planète colonisée à partir d'une autre planète.

Paramètres

fp	planète d'origine
faction	faction à laquelle doit appartenir la nouvelle planète

Note

Tient compte du voisinage de l'ancienne planète et le met à jour

4.1.3 Documentation des fonctions membres

4.1.3.1 void Colonized_planet : :add_to_budget (double *given_money*)

Ajoute les fonds obtenus auprès de la faction.

Augmente le budget du montant obtenu, et diminue d'autant la demande précédemment effectuée.

Paramètres

given_money	Montant obtenu

4.1.3.2 bool Colonized_planet : :attack (**Virtual_planet** ∗ *victim*)

Attaque une planète.

Paramètres

victim	planète qui est attaquée

Renvoie

victoire de l'attaque

4.1.3.3 void Colonized_planet : :convert_to_free_planet ( )

Transforme la colonie courante en free planete.

Avertissement

Ne supprime pas la colonie et ne met pas a jour le voisinage ! !

4.1.3.4 void Colonized_planet : :demand_to_faction (double *cost*)

Réalise une demande de fonds auprès de la faction.

Paramètres

cost	Montant demandé

4.1.3.5 double Colonized_planet : :estimate_cost ( ) [virtual]

Estime le coût d'une attaque.

Renvoie

coût estimé

Réimplémentée à partir de Virtual_planet.

4.1.3.6 double Colonized_planet : :get_defense ( ) [virtual]

Défense totale de la planète.

Renvoie

somme de la défense naturelle de la planète et de celle de la colonie

Réimplémentée à partir de Virtual_planet.

4.1.3.7 Faction & Colonized_planet : :get_faction ( ) [virtual]

Faction à laquelle appartient la planète.

Renvoie

faction

Réimplémentée à partir de Virtual_planet.

4.1.3.8 double Colonized_planet : :get_production ( ) [virtual]

Production de la planète.

Renvoie

somme de la production naturelle de la planète et de celle de la colonie

Réimplémentée à partir de Virtual_planet.

4.1.3.9 bool Colonized_planet : :is_attacked (Virtual_planet * *attacker*) [virtual]

La planète est attaquée.

Lorsque la planète est attaquée, elle tente de parer l'attaque. Son score de défense est utilisé et peut être inpacté ainsi que la valeur de production.

Paramètres

attacker	planète offensive

Renvoie

victoire de l'attaque

Note

Si la planète ne parvient pas à parer l'attaque alors elle est éliminée de la liste d'attente de jeu car elle sera supprimée par la suite.

Voir également

colony_production_
colony_defense_

Réimplémentée à partir de Virtual_planet.

Réimplémentée dans Mother_land.

4.1.3.10 void Colonized_planet : :run ( )

Joue le tour.

La planète peut choisir d'attaquer ou bien de produire des richesses.

Renvoie

Booléen indiquant si la planète a réalisé une attaque

4.1.3.11 void Colonized_planet : :update_neighbourhood (Virtual_planet ∗ *old_one,* Virtual_planet ∗ *new_one*) [virtual]

Mise à jour du voisinage.

Parcourt le voisinage de la planète courante et remplace old_one par new_one dans la liste.

Paramètres

old_one	ancienne planète
new_one	nouvelle planète

Note

Si l'ancienne planète était la cible de l'un des voisins, alors cette cible est effacée.

Réimplémentée à partir de Virtual_planet.

La documentation de cette classe a été générée à partir des fichiers suivants :
— /home/pipissavy/Dropbox/Cours/ISIMA/ZZ2/simulation/simu_multi_agents/src/Colonized_planet.hpp
— /home/pipissavy/Dropbox/Cours/ISIMA/ZZ2/simulation/simu_multi_agents/src/Colonized_planet.cpp

4.2 Référence de la classe Comparator

Comparateur de planètes colonisées.

`#include <Faction.hpp>`

Fonctions membres publiques

— Comparator (Colonized_planet ∗colonized_planet)
Constructeur.
— bool operator() (std : :pair< Colonized_planet ∗, double > pair_colony)
Comparaison avec la planète de base.

Attributs privés

— Colonized_planet ∗ colonized_planet_
Planète qui doit servir de base pour la comparaison.

4.2.1 Description détaillée

Comparateur de planètes colonisées.

4.2.2 Documentation des constructeurs et destructeur

4.2.2.1 Comparator : :Comparator (Colonized_planet ∗ *colonized_planet*) [inline]

Constructeur.

Paramètres

colonized_planet	planète servant d'élément de comparaison

4.2.3 Documentation des fonctions membres

4.2.3.1 bool Comparator : :operator() (std : :pair< Colonized_planet ∗, double > *pair_colony*) [inline]

Comparaison avec la planète de base.

Paramètres

pair_colony	une paire dont le premier membre est comparé avec la planète de base

Renvoie

Booléen à vrai si la comparaison est valide

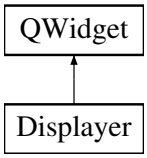
La documentation de cette classe a été générée à partir du fichier suivant :
— /home/pipissavy/Dropbox/Cours/ISIMA/ZZ2/simulation/simu_multi_agents/src/Faction.hpp

4.3 Référence de la classe Displayer

Zone d'affichage de l'application.

`#include <displayer.h>`

Graphe d'héritage de Displayer :



Fonctions membres publiques

— Displayer (QWidget ∗parent=0)
Constructeur par défaut.
— Displayer (QMainWindow ∗)
Crée l'interface visuelle de présentation de la grille de simulation.

— void display_world ()
Affiche la grille.
— bool play ()
Teste si la partie n'est pas finie.
— ~Displayer ()
Supprime et nettoie l'interface visuelle de présentation de la grille de simulation.

Connecteurs protégés

— void timerEvent ()
Evènement de mise à jour de l'affichage.
— void refresh ()
Commande de rafraichissement de l'affichage.

Fonctions membres privées

— void end ()
Affiche la faction gagnante dans une boîte à message.
— void display_planet (unsigned posX, unsigned posY)
Affiche une planète.

Attributs privés

— QGraphicsScene * m_scene
Scène de l'affichage.
— QGraphicsView * m_view
Vue.
— QPlanet ** graph_grid_
Grille des objets affichés.
— QGraphicsTextItem * text_panel_
Panneau de texte latéral.
— World world_
Plateau de jeu.
— unsigned size_planete_
Taille d'une planète à l'affichage (px)
— unsigned len_text_box_
Largeur du panneau latéral (px)
— unsigned len_canvas_
Largeur du panneau graphique (px)
— unsigned hei_canvas_
Hauteur du panneau graphique (px)
— QTimer * timer
Chronomètre pour gérer la fréquence d'affichage.
— QString winning_faction_
Nom de la faction gagnante.
— QPixmap shield_
Symbole bouclier.
— QPixmap gold_
Symbole lingot.

4.3.1 Description détaillée

Zone d'affichage de l'application.

Cette classe gère l'affichage de l'application ainsi que le lancement du jeu.

4.3.2 Documentation des constructeurs et destructeur

4.3.2.1 Displayer : :Displayer (QWidget * *parent* = 0) [explicit]

Constructeur par défaut.

Paramètres

parent	Objet parent (conteneur par exemple)

4.3.2.2 Displayer : :Displayer (QMainWindow ∗ *mw*)

Crée l'interface visuelle de présentation de la grille de simulation.

Paramètres

mw	Objet graphique Qt parent

4.3.3 Documentation des fonctions membres

4.3.3.1 void Displayer : :display_planet (unsigned *posX*, unsigned *posY*) [private]

Affiche une planète.

Paramètres

posX	Position X de la planète
posY	Position Y de la planète

Note

Position X = 0, Y = 0 en haut à gauche

4.3.3.2 bool Displayer : :play ( )

Teste si la partie n'est pas finie.

Renvoie

Etat du jeu (true -> en cours, false -> fini)

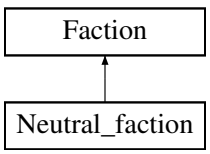
La documentation de cette classe a été générée à partir des fichiers suivants :
- displayer.h
- displayer.cpp

4.4 Référence de la classe Faction

Entité de jeu, ensemble de planètes.

`#include <Faction.hpp>`

Graphe d'héritage de Faction :



Fonctions membres publiques

- Faction (World &world, std : :string name="default", Mother_land ∗=nullptr)

Constructeur.

— Faction & operator= (Faction)

Opérateur d'affectation, sans effet.

— bool operator== (const Faction &other) const

Comparateur avec une autre faction.

— bool operator== (Faction &other)

Comparateur avec une autre faction.

— Faction * run ()

Lance le jeu.

— void init ()

Initialise la faction.

— void die ()

Tue la faction.

— std : :string get_name ()

Nom de la colonie.

— double get_money ()

Argent disponible.

— char get_motherland_symbol ()

Caractère représentant la planète mère.

— char get_colony_symbol ()

Caractère représentant une colonie.

— std : :string get_colony_color_name ()

Couleur x11 des colonies.

— std : :string get_motherland_color_name ()

Couleur x11 de la planète mère.

— Mother_land * get_motherland_ ()

Planète mère.

— std : :list< Colonized_planet * > & get_colonies ()

Liste des colonies.

— void add_colony (Colonized_planet *colony)

Ajoute une colonie à la faction.

— void remove_colony (Colonized_planet *colony)

Supprime une colonie.

— void remove_mother_land ()

Supprime la planète mère.

— void remove_demand (Colonized_planet *colony)

Supprime une demande de fonds.

— void add_to_banque (double)

Ajoute de l'argent généré à la banque.

— void set_colony_symbol (char c)

Définit le caractère représentant une colonie.

— void set_motherland_symbol (char c)

Définit le caractère représentant la planète mère.

— void set_colony_color_name (std : :string colony_color_name)

Définit la couleur des colonies.

— void set_motherland_color_name (std : :string motherland_color_name)

Définit la couleur de la planète mère.

— void add_demand (Colonized_planet *, double)

Ajoute une demande de fonds à la liste des demandes.

— double get_money_spent ()

Argent dépensé

— double get_money_produce_ ()

Argent généré

— int get_nb_successful_attack_ ()

Attaques réussies.

— int get_nb_failed_attack_ ()

Attaques échouées.

— void inc_nb_successful_attack_ ()

Augmente le compteur d'attaques réussies.

— void inc_nb_failed_attack_ ()

Augmente le compteur d'attaques échouées.

— std : :string toString ()

Texte représentant une faction.

— std : :string stats ()

Valeurs finales de la faction.

Attributs privés

- World & world_
  Monde auquel est rattachée la faction.
- std : :string name_
  Nom de la faction.
- double money_
  Argent possédé par la faction.
- Mother_land * motherland_
  Planète qui gère la faction.
- char motherland_symbol_
  Caractère représentant la planète mère.
- std : :list< Colonized_planet * > colonies_
  Liste des colonies de la faction.
- char colony_symbol_
  Caractère représentant les colonies.
- std : :string colony_color_name_
  Nom de la couleur représentant la colonie (x11names)
- std : :string motherland_color_name_
  Nom de la couleur représentant la planète mère (x11names)
- std : :list< std : :pair
  < Colonized_planet *, double > > demands_
  Liste des demandes de fonds associées à la planète demandeuse.
- double money_spent_
  Argent dépensé
- double money_produce_
  Argent généré
- int nb_successful_attack_
  Attaques réussies.
- int nb_failed_attack_
  Attaques échouées.

4.4.1 Description détaillée

Entité de jeu, ensemble de planètes.

Une faction est un groupement de colonies. Une faction est dirigée par une planète mère.

Note

La mort de la planète mère induit la mort de la faction.

4.4.2 Documentation des constructeurs et destructeur

4.4.2.1 Faction : :Faction (World & *world*, std : :string *name* = "default", Mother_land * *mother_land* = nullptr)

Constructeur.

Paramètres

world	Monde auquel appartient la faction
name	Nom de la faction
mother_land	Planète mère de la faction

4.4.3 Documentation des fonctions membres

4.4.3.1 void Faction : :add_demand (Colonized_planet * *demander*, double *cost*)

Ajoute une demande de fonds à la liste des demandes.

Paramètres

demander	La planète effectuant la demande
cost	Le montant demandé

4.4.3.2 void Faction : :add_to_banque (double *adding_money*)

Ajoute de l'argent généré à la banque.

Paramètres

adding_money	quantité d'argent à ajouter

4.4.3.3 void Faction : :die ( )

Tue la faction.

Avertissement

Provoque la suppression définitive des colonies

Note

Le voisinage et l'ordonnanceur sont mis à jour

4.4.3.4 void Faction : :init ( )

Initialise la faction.

Note

Crée la planète mère n'importe où sur la grille

A faire Vérifier si la place n'est pas déja occupée par une autre planète mère

4.4.3.5 void Faction : :remove_colony (**Colonized_planet** ∗ *colony*)

Supprime une colonie.

Paramètres

colony	colonie à supprimer

4.4.3.6 void Faction : :remove_demand (**Colonized_planet** ∗ *colony*)

Supprime une demande de fonds.

Paramètres

colony	colonie qui a initié la demande

4.4.3.7 void Faction : :remove_mother_land ( )

Supprime la planète mère.

Note

Cette méthode ne détruit pas la faction.

4.4.3.8 Faction * Faction : :run ( )

Lance le jeu.

Si la planète mère a été détruite, la faction est supprimée.

Voir également

die()

Renvoie

pointeur sur la faction qui a été détruite.

Note

En cas de non-destruction de la faction, le pointeur est nul.

4.4.3.9 string Faction : :stats ( )

Valeurs finales de la faction.

Renvoie

bloc de texte reprenant les paramètres principaux et de statistiques finales de manière formatée

4.4.3.10 string Faction : :toString ( )

Texte représentant une faction.

Renvoie

bloc de texte reprenant les paramètres principaux et de statistiques

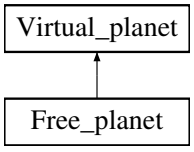
La documentation de cette classe a été générée à partir des fichiers suivants :
- /home/pipissavy/Dropbox/Cours/ISIMA/ZZ2/simulation/simu_multi_agents/src/Faction.hpp
- /home/pipissavy/Dropbox/Cours/ISIMA/ZZ2/simulation/simu_multi_agents/src/Faction.cpp

4.5 Référence de la classe Free_planet

Planète libre.

`#include <Free_planet.hpp>`

Graphe d'héritage de Free_planet :



Fonctions membres publiques

- Free_planet (World &, unsigned, unsigned)
  Constructeur.
- Free_planet (Colonized_planet ∗)
  Crée une planète libre à partir d'une colonie.
- void run ()
  Joue le tour.
- bool is_attacked (Virtual_planet ∗)
  Répond à une attaque.

Membres hérités additionnels

4.5.1 Description détaillée

Planète libre.

Planète non encore colonisée.

4.5.2 Documentation des constructeurs et destructeur

4.5.2.1 Free_planet : :Free_planet (World & *world,* unsigned *pos_x,* unsigned *pos_y*)

Constructeur.

Paramètres

world	Monde auquel est rattachée la planète
pos_x	Ligne sur la grille
pos_y	Colonne sur la grille

4.5.2.2 Free_planet : :Free_planet (Colonized_planet ∗ *other*)

Crée une planète libre à partir d'une colonie.

Voir également

Colonized_planet

Paramètres

other	planète servant de base

Note

Met à jour le voisinage en remplaçant l'ancienne par la nouvelle planète.

4.5.3 Documentation des fonctions membres

4.5.3.1 bool Free_planet : :is_attacked (Virtual_planet ∗) [virtual]

Répond à une attaque.

Renvoie

Booléen à vrai 4 fois sur 5 en moyenne

Réimplémentée à partir de Virtual_planet.

4.5.3.2 void Free_planet : :run ( )

Joue le tour.

Avertissement

Ne doit jamais être appelée car une planète libre n'est pas sensée jouer.

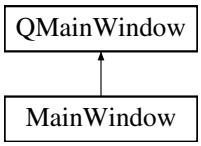
La documentation de cette classe a été générée à partir des fichiers suivants :
- /home/pipissavy/Dropbox/Cours/ISIMA/ZZ2/simulation/simu_multi_agents/src/Free_planet.hpp
- /home/pipissavy/Dropbox/Cours/ISIMA/ZZ2/simulation/simu_multi_agents/src/Free_planet.cpp

4.6 Référence de la classe MainWindow

Fenêtre principale.

`#include <mainwindow.h>`

Graphe d'héritage de MainWindow :



Fonctions membres publiques

- **MainWindow** (QWidget ∗parent=0)
- void show ()

  Affichage des objets contenus dans la fenêtre.

Attributs privés

- Ui : :MainWindow ∗ ui

  Interface utilisateur, contient les différents objets et la structure.

4.6.1 Description détaillée

Fenêtre principale.

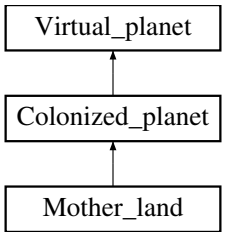
La documentation de cette classe a été générée à partir des fichiers suivants :
- mainwindow.h
- mainwindow.cpp

4.7 Référence de la classe Mother_land

Planète mère.

`#include <Mother_land.hpp>`

Graphe d'héritage de Mother_land :



Fonctions membres publiques

- Mother_land (Virtual_planet ∗, Faction &)
  Constructeur.
- char display ()
  Caractère représentant la planète mère.
- std : :string get_color_name ()
  Couleur représentant la planète mère.
- bool is_attacked (Virtual_planet ∗)
  Répond à une attaque.
- std : :string stats ()
  Statistique liées a la planete mere.

Membres hérités additionnels

4.7.1 Description détaillée

Planète mère.

Dirige une faction

Voir également

Faction

4.7.2 Documentation des constructeurs et destructeur

4.7.2.1 Mother_land : :Mother_land (Virtual_planet ∗ *fp,* Faction & *fac*)

Constructeur.

Appelle le constructeur de Colonized_planet

Voir également

Colonized_planet

Paramètres

fp	Planète présente précédemment
fac	Faction à laquelle est rattachée la planète mère

4.7.3 Documentation des fonctions membres

4.7.3.1 bool Mother_land : :is_attacked (Virtual_planet ∗ *attacker*) [virtual]

Répond à une attaque.

Paramètres

attacker	Planète attaquante

Renvoie

vrai si l'attaquant n'appartient pas à la même faction

Note

Tir ami interdit

Réimplémentée à partir de Colonized_planet.

4.7.3.2 string Mother_land : :stats ( )

Statistique liées a la planete mere.

Renvoie

une chaine de caracteres formatée

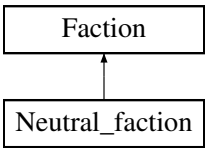
La documentation de cette classe a été générée à partir des fichiers suivants :
- /home/pipissavy/Dropbox/Cours/ISIMA/ZZ2/simulation/simu_multi_agents/src/Mother_land.hpp
- /home/pipissavy/Dropbox/Cours/ISIMA/ZZ2/simulation/simu_multi_agents/src/Mother_land.cpp

4.8 Référence de la classe Neutral_faction

Faction neutre.

`#include <Neutral_faction.hpp>`

Graphe d'héritage de Neutral_faction :



Fonctions membres publiques statiques

- static Neutral_faction * get_instance (World &world)
  Demande de la planète neutre.
- static void dispose ()
  Libère l'instance.

Fonctions membres privées

- Neutral_faction (World &world)
  Constructeur.

Attributs privés statiques

- static Neutral_faction * instance_ = nullptr
  Instance (unique)

Membres hérités additionnels

4.8.1 Description détaillée

Faction neutre.

C'est la faction à laquelle appartiennent les planètes libres

Note

Modèle singleton

4.8.2 Documentation des constructeurs et destructeur

4.8.2.1 Neutral_faction : :Neutral_faction (World & *world*) [private]

Constructeur.

Paramètres

world	Monde auquel appartient la faction neutre

4.8.3 Documentation des fonctions membres

4.8.3.1 Neutral_faction * Neutral_faction : :get_instance (World & *world*) [static]

Demande de la planète neutre.

Paramètres

world	Monde auquel la faction neutre appartient

Renvoie

pointeur sur l'instance unique de la faction neutre

La documentation de cette classe a été générée à partir des fichiers suivants :
- /home/pipissavy/Dropbox/Cours/ISIMA/ZZ2/simulation/simu_multi_agents/src/Neutral_faction.hpp
- /home/pipissavy/Dropbox/Cours/ISIMA/ZZ2/simulation/simu_multi_agents/src/Neutral_faction.cpp

4.9 Référence de la classe QPlanet

Gestion d'affichage de planète.

`#include <qplanet.h>`

Fonctions membres publiques

- QPlanet ()
  Initialise les pointeurs.
- QGraphicsEllipseItem * getEllipse () const
  Obtenir le disque représentant une planète.
- void setEllipse (QGraphicsEllipseItem *value)
  Définit le disque représentant une planète.
- QGraphicsPixmapItem * getGold () const
  Obtenir l'image des sous !
- void setGold (QGraphicsPixmapItem *value)
  Définir l'image des sous.
- QGraphicsPixmapItem * getShield () const
  Obtenir l'image du bouclier.

- void setShield (QGraphicsPixmapItem ∗value)
  Définir l'image du bouclier.
- QGraphicsTextItem ∗ getEco () const
  Obtenir la monnaie.
- void setEco (QGraphicsTextItem ∗value)
  Définir la monnaie.
- QGraphicsTextItem ∗ getDefense () const
  Obtenir la défense de la planète.
- void setDefense (QGraphicsTextItem ∗value)
  Définir la défense de la planète.

Attributs privés

- QGraphicsEllipseItem ∗ ellipse_
  Disque représentant une planète.
- QGraphicsPixmapItem ∗ gold_
  Image or.
- QGraphicsPixmapItem ∗ shield_
  Image bouclier.
- QGraphicsTextItem ∗ eco_
  Monnaie.
- QGraphicsTextItem ∗ defense_
  Défense.

4.9.1 Description détaillée

Gestion d'affichage de planète.

Conserve des pointeurs vers les planètes affichées, de manière à les remplacer quand elles ont changé.

4.9.2 Documentation des fonctions membres

4.9.2.1 QGraphicsTextItem ∗ QPlanet : :getDefense ( ) const

Obtenir la défense de la planète.

Renvoie

Texte représentant la défense

4.9.2.2 QGraphicsTextItem ∗ QPlanet : :getEco ( ) const

Obtenir la monnaie.

Renvoie

Texte représentant la monnaie

4.9.2.3 QGraphicsEllipseItem ∗ QPlanet : :getEllipse ( ) const

Obtenir le disque représentant une planète.

Renvoie

Ellipse Qt pour représenter la planète

4.9.2.4 QGraphicsPixmapItem ∗ QPlanet : :getGold ( ) const

Obtenir l'image des sous !

Renvoie

Pixmap avec des pièces

4.9.2.5 QGraphicsPixmapItem ∗ QPlanet : :getShield ( ) const

Obtenir l'image du bouclier.

Renvoie

Pixmap avec un bouclier

4.9.2.6 void QPlanet : :setDefense (QGraphicsTextItem ∗ *value*)

Définir la défense de la planète.

Paramètres

value	Texte représentant la défense généré à l'affichage

4.9.2.7 void QPlanet : :setEco (QGraphicsTextItem ∗ *value*)

Définir la monnaie.

Paramètres

value	Texte représentant la monnaie généré à l'affichage

4.9.2.8 void QPlanet : :setEllipse (QGraphicsEllipseItem ∗ *value*)

Définit le disque représentant une planète.

Paramètres

value	disque créé à l'affichage

4.9.2.9 void QPlanet : :setGold (QGraphicsPixmapItem ∗ *value*)

Définir l'image des sous.

Paramètres

value	Image des sous générée à l'affichage

4.9.2.10 void QPlanet : :setShield (QGraphicsPixmapItem ∗ *value*)

Définir l'image du bouclier.

Paramètres

value	Image du bouclier générée à l'affichage

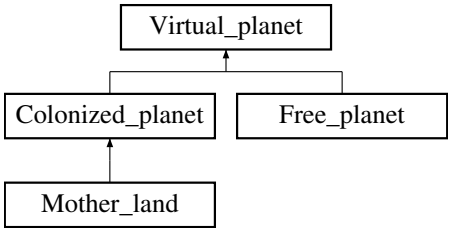
La documentation de cette classe a été générée à partir des fichiers suivants :
- qplanet.h
- qplanet.cpp

4.10 Référence de la classe Virtual_planet

Planète virtuelle.

`#include <Virtual_planet.hpp>`

Graphe d'héritage de Virtual_planet :



Fonctions membres publiques

- Virtual_planet (World &, unsigned, unsigned)
  Constructeur.
- Virtual_planet (Virtual_planet ∗)
  Constructeur par copie.
- virtual ∼Virtual_planet ()
  Destructeur à redéfinir.
- void set_neighbourhood ()
  Calcule le voisinage de la planète.
- void set_neighbourhood2 ()
  Calcule le voisinage de la planète.
- virtual void update_neighbourhood (Virtual_planet ∗, Virtual_planet ∗)
  Met à jour le voisinage.
- virtual bool is_attacked (Virtual_planet ∗)
  Réponse lorsqu'une attaque est subie (vrai par défaut)
- void run ()
  Joue sur le plateau.
- virtual void reinitialisate_target ()
  Réinitialisation de la cible, à redéfinir.
- bool has_changed ()
  Récupère un changement.
- virtual bool can_be_replaced ()
  Renvoie true si la planete n'est pas un agent.
- virtual char display ()
  Caractère par défaut (point)
- virtual double estimate_cost ()
  Estime le cout d'une attaque.
- virtual std : :string get_color_name ()
  Couleur par défaut (gris)
- virtual Faction & get_faction ()
  Obtenir la faction à laquelle appartient la planète.
- virtual unsigned pos_x ()
  Ligne dans la grille.
- virtual unsigned pos_y ()
  Colonne dans la grille.

- World & get_world ()
  Monde auquel appartient la planète.
- virtual double get_defense ()
  Défense de base.
- virtual double get_production ()
  Productivité de base.
- std : :vector< Virtual_planet * > get_neighbourhood ()
  Liste des voisins.

Fonctions membres protégées

- void change ()
  Signale un changement d'état.

Attributs protégés

- World & world_
  Monde auquel appartient la planète.
- unsigned pos_x_
  Ligne dans la grille.
- unsigned pos_y_
  Colonne dans la grille.
- std : :vector< Virtual_planet * > neighbourhood_
  Liste des voisins.
- bool changed_
  Etat changé
- double production_rate_
  Taux de production entre MIN_PRODUCTION_RATE et MAX_PRODUCTION_RATE.
- double natural_defense_
  Taux de défense naturelle entre MIN_NATURAL_DEFENSE et MAX_NATURAL_DEFENSE.

Attributs protégés statiques

- static const int MIN_PRODUCTION_RATE = 0
  Taux de production minimal.
- static const int MAX_PRODUCTION_RATE = 15
  Taux de production maximal.
- static const int MIN_NATURAL_DEFENSE = 25
  Taux de défense naturelle minimal.
- static const int MAX_NATURAL_DEFENSE = 50
  Taux de défense naturelle maximal.

4.10.1 Description détaillée

Planète virtuelle.

Planète servant de base à toutes les autres. Contient tous les paramètres communs

4.10.2 Documentation des constructeurs et destructeur

4.10.2.1 Virtual_planet : :Virtual_planet (World & *world,* unsigned *pos_x,* unsigned *pos_y*)

Constructeur.

Paramètres

world	Monde auquel la planète appartient
pos_x	Ligne dans la grille
pos_y	Colonne dans la grille

4.10.2.2 Virtual_planet : :Virtual_planet (Virtual_planet * *other*)

Constructeur par copie.

Paramètres

other	Planète d'origine

4.10.3 Documentation des fonctions membres

4.10.3.1 Faction & Virtual_planet : :get_faction ( ) [virtual]

Obtenir la faction à laquelle appartient la planète.

Renvoie

Faction neutre par défaut

Réimplémentée dans Colonized_planet.

4.10.3.2 bool Virtual_planet : :has_changed ( )

Récupère un changement.

Renvoie

l'état de changement

Note

réinitialise l'état de changement s'il était actif

4.10.3.3 void Virtual_planet : :run ( )

Joue sur le plateau.

Note

Non utilisée

4.10.3.4 void Virtual_planet : :set_neighbourhood2 ( )

Calcule le voisinage de la planète.

Note

Grille torique

4.10.3.5 void Virtual_planet : :update_neighbourhood (Virtual_planet * *old_one*, Virtual_planet * *new_one*) [virtual]

Met à jour le voisinage.

Paramètres

old_one	planète à remplacer
new_one	planète remplaçante

Réimplémentée dans Colonized_planet.

4.10.4 Documentation des données membres

4.10.4.1 double Virtual_planet : :natural_defense_ [protected]

Taux de défense naturelle entre MIN_NATURAL_DEFENSE et MAX_NATURAL_DEFENSE.

Voir également

MIN_NATURAL_DEFENSE
MAX_NATURAL_DEFENSE

4.10.4.2 double Virtual_planet : :production_rate_ [protected]

Taux de production entre MIN_PRODUCTION_RATE et MAX_PRODUCTION_RATE.

Voir également

MIN_PRODUCTION_RATE
MAX_PRODUCTION_RATE

La documentation de cette classe a été générée à partir des fichiers suivants :
— /home/pipissavy/Dropbox/Cours/ISIMA/ZZ2/simulation/simu_multi_agents/src/Virtual_planet.hpp
— /home/pipissavy/Dropbox/Cours/ISIMA/ZZ2/simulation/simu_multi_agents/src/Virtual_planet.cpp

4.11 Référence de la classe World

Monde du jeu, possédant la grille.

`#include <World.hpp>`

Fonctions membres publiques

— World (unsigned len=20, unsigned hei=20)
Constructeur.
— ∼World ()
Destructeur.
— time_h start ()
Lance la simulation.
— void scheduler ()
Ordonnanceur.
— void test2factions ()
Exemple de simulation avec 2 factions.
— void test3factions ()
Exemple de simulation avec 3 factions.
— void test4factions ()
Exemple de simulation avec 4 factions.
— void display ()
Affichage de la grille.
— void remove_faction (Faction ∗)
Suppression définitive d'une faction.
— void add_waiting_agent (Colonized_planet ∗)
Ajout d'un agent à la liste d'attente.

- void remove_waiting_agent (Colonized_planet ∗)
  Suppression d'un agent de la liste d'attente.
- Virtual_planet ∗ get_grid (unsigned x, unsigned y)
  Planète dans la grille.
- void set_grid (Virtual_planet ∗, unsigned x, unsigned y)
  Installe une planète dans la grille.
- unsigned len () const
  Nombre de places sur une ligne.
- unsigned hei () const
  Nombre de places sur une colonne.
- bool isEnded ()
  Vrai si la partie est terminée.
- unsigned get_steps () const
  Numéro du tour courant.
- std : :list< Faction > get_factions ()
  Liste des factions.
- string toString ()
  to string
- string get_winner_name ()
  Nom de la faction gagnante.
- string stats_general ()
  Statistiques.
- string stats_faction ()
  Statistiques.
- void add_stat_faction (Faction &)
  Ajout dans la liste des statistiques de faction et mise a jour des statistiques generales.

Fonctions membres publiques statiques

- static int gen_mt ()
  Générateur aléatoire (Mersenne Twister)
- static int gen_mt (int a, int b)
  Générateur aléatoire borné par un intervalle.
- static int gen_mt_shuffle (int i)
  Générateur aléatoire borné supérieurement.
- static void dispose ()
  Nettoie la faction neutre.

Attributs privés

- bool end_
  Fin de la partie.
- std : :vector< std : :vector < Virtual_planet ∗ > > grid_
  Grille de jeu.
- std : :list< Faction > factions_
  Liste des factions en jeu.
- std : :vector< Colonized_planet ∗ > waiting_agents_
  Liste des agents n'ayant pas encore joué
- std : :vector< Colonized_planet ∗ > already_run_agents_
  Liste des agents ayant déjà joué
- unsigned steps_
  Nombre de tours.
- unsigned nb_simulated_agents_
  Nombre d'agents.
- unsigned nb_attacks_success_
  Nombre d'attaques réussites dans la simulation.
- unsigned nb_attacks_failed_
  Nombre d'attaques echouées dans la simulation.
- list< string > stats_factions_
  Statistiques finales de toutes les factions.
- unsigned len_

Nombre de places par ligne.

— unsigned hei_

Nombre de places par colonne.

Attributs privés statiques

— static const bool DEBUG = false

Constante de débogage.

— static std : :mt19937 gen_mt_ = std : :mt19937()

Générateur aléatoire.

4.11.1 Description détaillée

Monde du jeu, possédant la grille.

4.11.2 Documentation des constructeurs et destructeur

4.11.2.1 World : :World (unsigned *len* = 20, unsigned *hei* = 20)

Constructeur.

Paramètres

len	Nombre de planètes par ligne
hei	Nombre de planètes par colonne

4.11.3 Documentation des fonctions membres

4.11.3.1 void World : :add_stat_faction (Faction & *faction*)

Ajout dans la liste des statistiques de faction et mise a jour des statistiques generales.

Paramètres

faction	Faction a archiver dans les statisiques

4.11.3.2 void World : :add_waiting_agent (Colonized_planet * *colonized_planet*)

Ajout d'un agent à la liste d'attente.

Paramètres

colonized_planet	planète à ajouter

4.11.3.3 void World : :display ( )

Affichage de la grille.

Note

Pour le mode console

4.11.3.4 int World : :gen_mt ( ) [static]

Générateur aléatoire (Mersenne Twister)

Renvoie

entier aléatoire

4.11.3.5 int World : :gen_mt (int *a*, int *b*) [static]

Générateur aléatoire borné par un intervalle.

Paramètres

a	borne inférieure
b	borne supérieure

Renvoie

entier aléatoire entre a et b

4.11.3.6 int World : :gen_mt_shuffle (int *i*) [static]

Générateur aléatoire borné supérieurement.

Paramètres

i	borne supérieure (intervalle ouvert)

Renvoie

entier aléatoire entre 0 et i

4.11.3.7 std : :list< Faction > World : :get_factions ( )

Liste des factions.

Renvoie

liste

4.11.3.8 Virtual_planet * World : :get_grid (unsigned *x*, unsigned *y*)

Planète dans la grille.

Paramètres

x	ligne où se trouve la planète
y	colonne où se trouve la planète

Renvoie

planète à la position (x,y)

4.11.3.9 string World : :get_winner_name ( )

Nom de la faction gagnante.

Renvoie

nom de la faction gagnante

4.11.3.10 void World : :remove_faction (Faction ∗ *faction*)

Suppression définitive d'une faction.

Paramètres

faction	faction à supprimer

4.11.3.11 void World : :remove_waiting_agent (Colonized_planet ∗ *colonized_planet*)

Suppression d'un agent de la liste d'attente.

Paramètres

colonized_planet	planète à supprimer

4.11.3.12 void World : :scheduler ( )

Ordonnanceur.

Réalise un tirage aléatoire de l'ordre de passage des agents.

Lance l'exécution de chaque faction en réponse aux attaques du tour précédent.

Supprime les factions qui n'ont pas survécu aux attaques.

Lance l'exécution de chaque planète, agent par agent selon l'ordre défini.

Détermine si le jeu est terminé (i.e. s'il ne reste plus qu'une faction en jeu)

4.11.3.13 void World : :set_grid (Virtual_planet ∗ *planet,* unsigned *x,* unsigned *y*)

Installe une planète dans la grille.

Paramètres

planet	planète à insérer
x	ligne de destination
y	colonne de destination

4.11.3.14 time_h World : :start ( )

Lance la simulation.

Renvoie

Nombre de tours joués jusqu'à la fin du jeu

4.11.3.15 string World : :stats_faction ( )

Statistiques.

Renvoie

texte formaté représentant les statistiques finales des factions

4.11.3.16 string World : :stats_general ( )

Statistiques.

Renvoie

texte formaté représentant les statistiques finales de jeu

4.11.3.17 string World : :toString ( )

to string

Renvoie

texte représentant les valeurs courante du jeu

4.11.4 Documentation des données membres

4.11.4.1 std : :vector<Colonized_planet * > World : :waiting_agents_ [private]

Liste des agents n'ayant pas encore joué

Note

Echangé avec already_run_agents_ à chaque tour

Voir également

already_run_agents_

La documentation de cette classe a été générée à partir des fichiers suivants :
- /home/pipissavy/Dropbox/Cours/ISIMA/ZZ2/simulation/simu_multi_agents/src/World.hpp
- /home/pipissavy/Dropbox/Cours/ISIMA/ZZ2/simulation/simu_multi_agents/src/World.cpp

Index