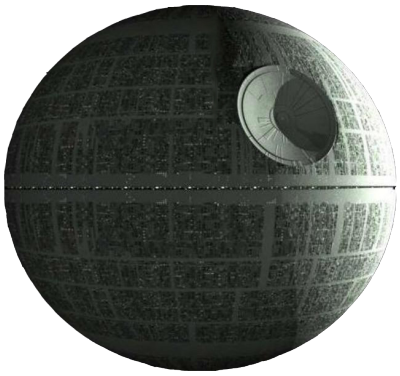
Document Technique

La guerre des étoiles



Organisé par :

Le comité exécutif des Jeux et défis informatiques

de Sherbrooke

11 juillet 2015

Table des matières

Table des matières **1**

1 Informations générales **3**

2 Node.js **3**

2.1 Organisation des fichiers . . . 3

2.1.1 app.js . . . 3

2.1.2 AI.js . . . 3

2.2 Contenu de game_objects . . . 3

2.3 Fonctions de l'objet game . . . 5

2.3.1 attack_planet . . . 5

3 Python 3.4 **6**

3.1 Organisation des fichiers . . . 6

3.1.1 server.py . . . 6

3.1.2 game_api.py . . . 6

3.1.3 AI.py . . . 6

3.2 Contenu de game_objects . . . 6

3.3 Fonctions du module game_api . . . 8

3.3.1 attack_planet . . . 8

4 C# **8**

4.1 Organisation des fichiers . . . 8

4.1.1 AIServer.cs . . . 8

4.1.2 Planet.cs . . . 9

4.1.3 Ship.cs . . . 9

4.1.4 TCPServer.cs . . . 9

4.1.5 UpdateContainer.cs . . . 9

4.1.6 AI.cs . . . 9

4.2 Fonctions publiques de la classe TCPServer . . . 9

4.2.1 AttackPlanet . . . 9

5 Game objects **10**

5.1 Ship . . . 10

5.1.1 position . . . 10

5.1.2 owner . . . 10

5.1.3 ship_count . . . 10

5.1.4 id . . . 10

5.1.5 target . . . 10

5.2 Planet . . . 10

5.2.1 position . . . 10
5.2.2 owner . . . 10
5.2.3 ship_count . . . 11
5.2.4 is_deathstar . . . 11
5.2.5 deathstar_charge . . . 11
5.2.6 size . . . 11
5.2.7 id . . . 11

6 Ajout phase 2 **11**

6.1 Python . . . 11
6.1.1 deathstar_destroy_planet . . . 11
6.1.2 redirect_ship . . . 11
6.2 Node.js . . . 12
6.2.1 deathstar_destroy_planet . . . 12
6.2.2 redirect_ship . . . 12
6.3 C# . . . 12
6.3.1 DeathstarDestroyPlanet . . . 12
6.3.2 RedirectShip . . . 12

1 Informations générales

L'API est branché au jeu via une communication TCP. Pour sélectionner le port désiré, chaque language permet de spécifier le port en "command-line argument". e.g : python server.py 8888, node app.js 8888 et AIServer.exe 8888. À chaque 30 frames, soit environ 1 secondes la fonction update de votre intelligence artificielle sera appelée avec les informations pertinantes pour faire vos actions.

2 Node.js

Le port par défaut de l'API Node.js est le port 8890. Pour partir l'AI, il suffit de lancer la commande suivante : node app.js

2.1 Organisation des fichiers

2.1.1 app.js

Vous ne devez en aucun cas modifiez ce fichier. Il contient les fonctions de communications TCP et les fonctions pour communiquer au jeu. Ces fonctions sont au bas du fichier si jamais vous voulez les consulter.

2.1.2 AI.js

La variable name doit être assignée à votre nom d'équipe sans quoi vous pourriez être disqualifié de la ronde du tournoi en question.

La variable game contient les fonctions pour communiquer au jeu.

La fonction set_game est appelée par app.js pour initialiser game. NE PAS MODIFIER OU ENLEVER !

La fonction update est appelée par le jeu à chaque 30 frames avec un objet représentant l'état de la partie en cours. Voir la section game_objects plus bas pour voir son contenu.

La fonction set_name est appelé au début du jeu pour que votre intelligence artificielle renvoie son nom au jeu. NE PAS MODIFIER OU ENLEVER !

À la fin du fichier, les fonctions principales sont exportées pour que app.js puisse les appeler. NE PAS MODIFIER OU ENLEVER !

2.2 Contenu de game_objects

Voici un exemple de game_objects que vous pourriez recevoir dans un update :

```
{
    "ships": [
        {
            "position": {
                "x": "3.422666",
                "y": "-2.913278"
            },
            "owner": "Julien",
            "ship_count": "84",
            "id": "3",
            "target": "6"
        },
        {
            "position": {
                "x": "0.8924749",
                "y": "-1.540014"
            },
            "owner": "Julien",
            "ship_count": "43",
            "id": "4",
            "target": "3"
        },
        {
            "position": {
                "x": "-2.97397",
                "y": "0.9789585"
            },
            "owner": "Julien",
            "ship_count": "78",
            "id": "5",
            "target": "5"
        }
    ],
    "planets": [
        {
            "position": {
                "x": "-2.278147",
                "y": "-1.999854"
```

```
            },
            "owner": "Julien",
            "ship_count": "0",
            "is_deathstar": "false",
            "deathstar_charge": "NaN",
            "size": "1.196134",
            "id": "0"
        },
        {
            "position": {
                "x": "2.278147",
                "y": "1.999854"
            },
            "owner": "Bot",
            "ship_count": "616",
            "is_deathstar": "false",
            "deathstar_charge": "NaN",
            "size": "1.196134",
            "id": "1"
        }
    ]
}
```

Vous remarquerez que tous les variables sont en string. Vous devrez donc faire très attention de caster correctement vos variables.

2.3 Fonctions de l'objet game

Il est à noter que cette section ne contiendra que les fonctions qui sont dans la première phase de la compétition. Les autres fonctions seront documentées dans les documents de phase.

2.3.1 attack_planet

Cette fonction est utilisée pour attaquer ou défendre une planète. Voici sa signature : attack_planet(id_origin, id_target, num_ship)

- id_origin : L'id de la planète d'origine des ships.
- id_target : L'id de la planète de destination des ships.
- num_ship : Le nombre de ships à envoyer.

3 Python 3.4

Le port par défaut de l'API python est le port 8888. Pour lancer l'AI, vous devez entrer la commande suivante : python server.py

3.1 Organisation des fichiers

3.1.1 server.py

Vous ne devez en aucun cas modifiez ce fichier. Il contient les fonctions de communications TCP et les fonctions pour communiquer au jeu.

3.1.2 game_api.py

Vous ne devez en aucun cas modifiez ce fichier. Il contient les fonctions qui vous seront utiles pour faire des actions dans le jeu.

3.1.3 AI.py

La variable name doit être assignée à votre nom d'équipe sans quoi vous pourriez être disqualifié de la ronde du tournois en question.

La module game_api contient les fonctions pour communiquer au jeu.

La fonction update est appelée par le jeu à chaque 30 frame avec un objet représentant l'état de la partie en cours. Voir la section game_objects plus bas pour voir son contenu.

La fonction set_name est appelée au début du jeu pour que votre intelligence artificielle renvoie son nom au jeu. NE PAS MODIFIER OU ENLEVER !

3.2 Contenu de game_objects

Voici un exemple de game_objects que vous pourriez recevoir dans un update :

```
{
    "ships": [
        {
            "position": {
                "x": "3.422666",
                "y": "-2.913278"
            },
            "owner": "Julien",
            "ship_count": "84",
```

```
            "id": "3",
            "target": "6"
        },
        {
            "position": {
                "x": "0.8924749",
                "y": "-1.540014"
            },
            "owner": "Julien",
            "ship_count": "43",
            "id": "4",
            "target": "3"
        },
        {
            "position": {
                "x": "-2.97397",
                "y": "0.9789585"
            },
            "owner": "Julien",
            "ship_count": "78",
            "id": "5",
            "target": "5"
        }
    ],
    "planets": [
        {
            "position": {
                "x": "-2.278147",
                "y": "-1.999854"
            },
            "owner": "Julien",
            "ship_count": "0",
            "is_deathstar": "false",
            "deathstar_charge": "NaN",
            "size": "1.196134",
            "id": "0"
        },
        {
            "position": {
```

```
                "x": "2.278147",
                "y": "1.999854"
            },
            "owner": "Bot",
            "ship_count": "616",
            "is_deathstar": "false",
            "deathstar_charge": "NaN",
            "size": "1.196134",
            "id": "1"
        }
    ]
}
```

Vous remarquerez que tous les variables sont en string. Vous devrez donc faire très attention de caster correctement vos variables.

3.3 Fonctions du module game_api

Il est à noter que cette section ne contiendra que les fonctions qui sont dans la première phase de la compétition. Les autres fonctions seront documentées dans les documents des autres phases.

3.3.1 attack_planet

Cette fonction est utilisée pour attaquer ou défendre une planète. Voici sa signature : attack_planet(id_origin, id_target, num_ship)

- id_origin : L'id de la planète d'origine des ships.
- id_target : L'id de la planète de destination des ships.
- num_ship : Le nombre de ships à envoyer.

4 C#

Le port par défaut de l'API C# est le port 8889.

4.1 Organisation des fichiers

4.1.1 AIServer.cs

Main du programme permettant de modifier le port avec les command-line arguments. NE PAS MODIFIER OU ENLEVER !

4.1.2 Planet.cs

La classe pour représenter une planète. Elle contient toutes les propriétés d'une planète.

4.1.3 Ship.cs

La classe pour représenter un vaisseau. Elle contient toutes les propriétés d'un vaisseau.

4.1.4 TCPServer.cs

La classe permettant de communiquer avec le jeu. Les fonctions publiques de cette classe permettent de faire les actions désirées dans le jeu. NE PAS MODIFIER OU ENLEVER !

4.1.5 UpdateContainer.cs

La classe qui est envoyée en paramètre à l'update. Elle contient une liste de planètes et de vaisseaux. NE PAS MODIFIER OU ENLEVER !

4.1.6 AI.cs

La variable name doit être assignée à votre nom d'équipe sans quoi vous pourriez être disqualifié de la ronde du tournois en question.

La propriété Game contient les fonctions pour communiquer au jeu.

La fonction update est appelé par le jeu à chaque 30 frame avec un UpdateContainer représentant l'état de la partie en cours.

La fonction set_name est appelée au début du jeu pour que votre intelligence artificielle renvoie son nom au jeu. NE PAS MODIFIER OU ENLEVER !

4.2 Fonctions publiques de la classe TCPServer

Il est à noter que cette section ne contiendra que les fonctions qui sont dans la première phase de la compétition. Les autres fonctions seront documentées dans les documents des autres phases.

4.2.1 AttackPlanet

Cette fonction est utilisée pour attaquer ou défendre une planète. Voici sa signature :
public void AttackPlanet(Planet origin, Planet target, int shipCount)

— origin : L'objet Planet d'où proviennent les ships.

— target : L'objet Planet de destination des ships.

— shipCount : Le nombre de ships à envoyer.

5 Game objects

5.1 Ship

Un Ship est le vaisseau qui est envoyé pour défendre ou attaquer une planète.

5.1.1 position

La position du vaisseau est en 2D, donc celle-ci est divisée en coordonnées x et y.

5.1.2 owner

Le propriétaire du Ship, cette variable correspond au nom inscrit dans votre AI ou celui de votre ennemi.

5.1.3 ship_count

Le nombre de forces que contient le vaisseau.

5.1.4 id

Un integer qui correspond à l'identifiant du vaisseau. Celui-ci est unique et permet donc de retrouver à coup sûr le ship voulu.

5.1.5 target

Un integer qui correspond au id de la planète de destination du vaisseau.

5.2 Planet

Une planète abrite un nombre x de ship et peut par le fait même envoyer ces ships vers une planète pour la défendre ou l'attaquer.

5.2.1 position

La position de la planète est 2D, donc celle-ci est divisée en coordonnées x et y.

5.2.2 owner

Le propriétaire de la Planet, cette variable correspond au nom inscrit dans votre AI ou celui de votre ennemi.

5.2.3 ship_count

Le nombre de Ship que contient la planète.

5.2.4 is_deathstar

Un booléen qui indique si la planète est une deathstar.

5.2.5 deathstar_charge

Un float qui indique la charge du laser destructeur de la deathstar lorsque c'est une deathstar. Lorsque cette variable est >= 1 cela signifie que le laser est prêt a tirer.

5.2.6 size

La taille de la planète. Cette variable influence la vitesse à laquelle la planète produit des vaisseaux. Plus elle est grosse, plus la planète génère des ships.

5.2.7 id

Un integer qui correspond à l'identifiant de la planète. Celui-ci est unique et permet donc de retrouver à coup sur la planète voulu.

6 Ajout phase 2

6.1 Python

6.1.1 deathstar_destroy_planet

Cette fonction est utilisée pour détruire une planète avec la deathstar lorsque celle-ci à un deathstar_charge >= 1. Voici sa signature :
deathstar_destroy_planet(id_deathstar, id_planet_end)

- id_deathstar : L'id de la deathstar.
- id_planet_end : L'id de la planète à détruire.

6.1.2 redirect_ship

Cette fonction change la target d'un ship. Voici sa signature :
redirect_ship(id_ship, id_new_target)

- id_ship : L'id du vaisseau à rediriger.
- id_new_target : L'id de la planète vers laquelle on redirige.

6.2 Node.js

6.2.1 deathstar_destroy_planet

Cette fonction est utilisée pour détruire une planète avec la deathstar lorsque celle-ci à un deathstar_charge >= 1. Voici sa signature :

deathstar_destroy_planet(id_deathstar, id_planet_target)

— id_deathstar : L'id de la deathstar.

— id_planet_target : L'id de la planète à détruire.

6.2.2 redirect_ship

Cette fonction change la target d'un ship. Voici sa signature :

redirect_ship(id_ship, id_target)

— id_ship : L'id du vaisseau à rediriger.

— id_target : L'id de la planète vers laquelle on redirige.

6.3 C#

6.3.1 DeathstarDestroyPlanet

Cette fonction est utilisée pour détruire une planète avec la deathstar lorsque celle-ci à un deathstar_charge >= 1. Voici sa signature :

void DeathstarDestroyPlanet(Planet deathstar, Planet target)

— deathstar : La deathstar.

— target : La planète à détruire.

6.3.2 RedirectShip

Cette fonction change la target d'un ship. Voici sa signature :

void RedirectShip(Ship ship, Planet target)

— ship : Le vaisseau à rediriger.

— target : La planète vers laquelle on redirige.