Faster Than Light

Projet de fin de semestre de l'UE PO

Introduction

Faster Than Light est un jeu de stratégie dans l'espace créé par SUBSET GAMES. Le joueur contrôle un vaisseau et son équipage pour se défendre contre des attaquants. L'intérêt du jeu consiste à exploiter intelligemment les différents systèmes de son vaisseau pour pouvoir détruire le vaisseau adversaire avant d'être détruit soi-même. En pratique, il faut bien répartir l'énergie et l'équipage entre les systèmes, et trouver quels systèmes de l'adversaire doivent être éliminés rapidement.

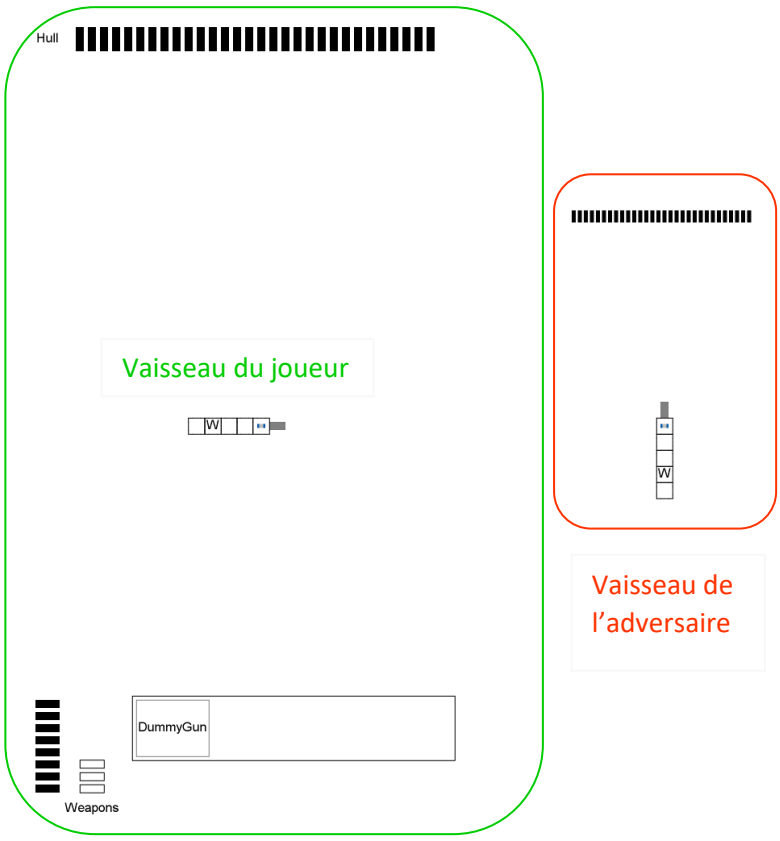
Après chaque combat, le joueur améliore son vaisseau afin de pouvoir affronter des adversaires de plus en plus puissants.

Dans ce projet, nous allons implémenter une version simplifiée du jeu Faster Than Light. Vous êtes bien sûr libres de l'améliorer après !

Règles[1]

Champ de bataille

- Le champ de bataille est composé de deux vues :
  - Le vaisseau du joueur
  - Le vaisseau de l'adversaire



[1] Plus d'info sur https://ftl.fandom.com/wiki/FTL:_Faster_Than_Light_Wiki

Les vaisseaux

Chaque vaisseau est composé :

- de quatre salles : la salle du module **moteur** , la salle du module **bouclier,** la salle du module **arsenal** et une salle qui n'est associée à aucun module
- d'un réacteur qui permet de distribuer de l'énergie aux différents modules
- d'une coque avec un niveau d'intégrité
- d'un équipage composé d'un ou plusieurs membres

Coque

- L'intégrité du vaisseau est initialement de 30, le vaisseau est détruit si l'intégrité tombe à 0
- Si un projectile touche le vaisseau alors l'intégrité de la coque baisse. Le nombre de points d'intégrité perdus dépend des dégâts subis (cf « Dégâts »)

Réacteur

- Un vaisseau commence avec un réacteur pouvant fournir 8 d'énergie
- Cette énergie peut être distribuée dans les modules (moteur, bouclier ou arsenal)
- Après chaque combat le réacteur peut être amélioré pour fournir plus d'énergie (voir «fin de combat»)
- Le niveau maximal est de 15

Modules

- Pour activer un module, on doit lui donner de l'énergie provenant du réacteur. Pendant un combat, on peut également retirer de l'énergie à un module pour la restituer au réacteur et ensuite alimenter d'autres modules.
  Par exemple, on peut vouloir donner moins d'énergie au module Arsenal pour donner plus d'énergie au module Moteur.
- Le niveau d'un module correspond au nombre maximum d'énergie que l'on peut donner à ce module. Il est toutefois possible d'alimenter partiellement le module.
  Par exemple, le module Arsenal commence avec 3 d'énergie maximum, mais on peut l'alimenter uniquement avec 1 d'énergie.
- Le niveau **actif** d'un module correspond à la quantité d'énergie qui alimente ce module (il peut être inférieur au niveau du module dans le cas d'une alimentation partielle)
- Lors du premier combat les modules sont tous de niveau 1 mais ce niveau pourra augmenter par la suite (voir «fin de combat»)
- Chaque joueur possède trois modules qui ont tous une utilité différente :

    - Moteur
        - Quand actif le vaisseau peut esquiver un projectile avec probabilité 5%
        - La probabilité augmente de 5% par niveau **actif**
        - Le niveau maximal est 8
    - Bouclier
        - Le module bouclier fournit un nombre de protection au vaisseau qui correspond au nombre de niveaux **impairs actifs**
        - Pour chaque niveau **pair actif**, la protection du niveau précédent se recharge plus rapidement (en 1.5 secondes)
        - Quand une protection bloque un projectile, elle se décharge et mettra 2 secondes à se recharger
        - Les protections se rechargent dans l'ordre.

Exemple avec 2 protections : deux projectiles viennent d'être bloqués alors la première protection se recharge et la deuxième protection se rechargera quand la première sera de nouveau opérationnelle
        - Le niveau maximal est 8
    - Arsenal
        - Quand actif, il est possible de tirer un projectile avec une arme
        - Le niveau **actif** correspond à l'énergie fournie à l'arme lorsque que le joueur décide de tirer
        - Le niveau maximal est 4

Armes

- Il existe trois types d'armes :
    - Laser :
        - endommage un module touché ainsi que la coque en leur infligeant entre 1 et 4 dégâts
        - le nombre de dégâts infligés dépend du niveau d'énergie actif du module arsenal
        - un bouclier peut bloquer un projectile laser, le vaisseau (module visé et coque) ne subit alors aucun dégât
    - Ion :
        - désactive un module ennemi touché pendant un certain temps mais n'inflige aucun dégât
        - si le projectile ion est bloqué par un bouclier alors le bouclier est désactivé pendant un certain temps
        - la durée de désactivation varie entre 1 et 4 secondes en fonction du niveau d'énergie actif du module arsenal
    - Missile :
        - endommage un module touché ainsi que la coque en leur infligeant 4 dégâts
        - n'est pas arrêté par le bouclier du vaisseau
        - il faut que le module arsenal ait au moins un niveau actif pour pouvoir tirer un missile mais cette arme n'est pas affectée par les niveaux d'énergie supplémentaires
        - le nombre de missiles est limité. Le joueur possède 3 missiles pour toute la partie mais pourra éventuellement en récupérer à la fin d'un combat (voir «fin de combat»)

- Quand au moins un niveau d'énergie de l'arsenal est actif, le joueur peut cliquer sur une arme, et choisir le module visé.

Dégâts

- Si un module est touché par un projectile, il subit le nombre de dégâts infligés par le projectile
- Quand un module subit X de dégâts, les X derniers niveaux du module sont endommagés et donc désactivés
- Les modules sont réparés par les membres d'équipage. Un membre d'équipage répare automatiquement le module de la salle où il est si ce module est endommagé.
- La réparation d'un niveau prend 2 secondes, et les niveaux sont réparés dans l'ordre
    - Le temps de réparation est divisé par le nombre de membres d'équipage réparant le module

L'Équipage

- Au début d'un combat, tous les membres de l'équipage sont dans la salle qui ne contient aucun module
- Pendant le combat, le joueur peut téléporter chaque membre de l'équipage pour les répartir dans les différentes salles
    - Une salle peut contenir plusieurs membres d'équipage
- Chaque membre d'équipage présent dans une salle qui contient un module octroie un bonus de 5 % à ce module

IA adversaire

En plus du vaisseau du joueur, vous devrez implémenter les vaisseaux adversaires. L'IA utilisée peut être naïve (ex : répartition uniforme de l'énergie disponible et utilisation aléatoire d'armes). La seule contrainte est que les vaisseaux ennemis ne doivent pas rester passifs. À chaque fin de combat un vaisseau ennemi plus puissant apparaît mais l'IA peut rester la même.

Caractéristique initiale des vaisseaux

Lors du premier combat, les deux vaisseaux (joueur et 1er ennemi) commencent avec les caractéristiques suivantes :

- Énergie disponible dans le réacteur : 3
- Armes : 1 laser, 1 Ion, 1 lanceur de missiles
- Niveau maximum des modules : 1
- Nombre de membre dans l'équipage : 1
- Intégrité du vaisseau : 30

Fin de combat

- Si le vaisseau du joueur est détruit, la partie est perdue
- Si le vaisseau ennemi est détruit, une récompense aléatoire est fournie au joueur ainsi qu'une amélioration pour le vaisseau.
- La récompense aléatoire peut être :
    - Une arme
    - Une réparation entre 1 et 5 de l'intégrité de la coque
    - Un membre d'équipage
    - Un missile supplémentaire
- Le joueur peut choisir d'améliorer un des modules du vaisseau ou d'augmenter l'énergie disponible dans le réacteur
- Quand le joueur a choisi son amélioration, un autre vaisseau aléatoire ennemi plus puissant apparait

Implémentation fournie et travail à faire

Un début d'implémentation vous est fourni. Cette implémentation contient un moteur de jeu minimal permettant de lancer le jeu et d'interagir avec quelques éléments du vaisseau. Voilà ce que vous pouvez faire avec cette implémentation :

- Gestion de l'énergie du module Arsenal : ajouter avec '1', diminuer avec 'Shift-1'
- Activer l'arme « DummyWeapon » avec 'a' (il y a un temps de chargement). Vous pouvez aussi cliquer avec la souris sur « DummyWeapon ». Le chargement n'est possible que s'il y a de l'énergie affectée au module Arsenal.
    - NB : nous avons laissé dans le code fourni un temps de chargement pour l'arme pour vous montrer comment faire, mais ne l'imposons pas dans la règle du jeu. Vous êtes libre d'instaurer ou non des temps de chargement pour vos armes.

- Tirer l'arme « DummyWeapon » avec 'Ctrl-a'
- Viser le vaisseau adverse : flèches directionnelles
- Sélectionner un membre d'équipage : 'q', désélectionner avec 'Shift-q'

Comme vous pouvez le constater, il manque beaucoup des fonctionnalités demandées dans la règle du jeu. En particulier, vous allez devoir :

- Gérer la visée : on ne peut pas encore choisir la salle du vaisseau adversaire visée
- Gérer les tirs : actuellement des projectiles sont envoyés mais n'arrivent nulle part
- Gérer les dégâts, qui dépendent de plusieurs facteurs (niveau actif du module Arsenal, présence ou non de membres d'équipages dans l'Arsenal, évitement et bouclier du vaisseau adverse)
- Implémenter les modules manquants, et la répartition d'énergie entre les modules
- Implémenter les armes manquantes
- Permettre de téléporter des membres d'équipage dans des salles particulières : pour cela, ajouter des touches permettant de choisir où téléporter le membre d'équipage sélectionné
- Gérer les réparations des modules par les membres d'équipage
- Gérer la fin du combat

Les classes qui vous sont fournies vous donnent une base solide sur laquelle construire votre code. Nous les expliquons un peu ici, elles sont également commentées en Javadoc.

- Start : la classe principale du jeu, c'est là que réside le main et la boucle principale du jeu. Normalement vous n'avez pas ou peu besoin la modifier.
- World : représente le monde du jeu, avec les deux vaisseaux (celui du joueur et celui de l'adversaire). Contient plusieurs fonctions importantes :
    - processKey() gère les appuis sur les touches par le joueur
    - step() fait tous les traitements nécessaires à faire avancer le jeu d'un petit pas de temps. Dans ces traitements vous trouverez la fonction processHit(…) (à faire), qui permet de gérer les dégâts des projectiles sur les vaisseaux
    - draw() dessine l'écran du jeu
- Bindings : contient les traitements des touches pressées par l'utilisateur. A bien regarder aussi !
- Tile : une salle d'un vaisseau (un petit carré sur la représentation graphique).
- Ship : classe abstraite représentant un vaisseau, ici on vous donne beaucoup de choses : il faut passer du temps pour bien comprendre cette classe.
- DummyShip : classe héritée de Ship, représente un petit vaisseau de 4 salles en ligne. Observez bien comment dans le constructeur, le vaisseau est construit salle par salle. Cette approche est très flexible et permet facilement de faire d'autres plans de vaisseaux si vous le désirez.
- CrewMember : un membre d'équipage.
- Module : classe abstraite pour tous les modules, contient la gestion de l'énergie et le dessin. Bien regarder ce que fait cette classe !
- Reactor : classe du module Reacteur.
- WeaponControl : classe du module Arsenal. Beaucoup de choses se passent ici car il faut gérer les différentes armes mais aussi le tir de projectiles : à bien regarder !
- Button : une représentation d'un Bouton qui peut être cliqué à la souris. Pour cela il faut faire une classe héritée : regardez la classe WeaponButton dans WeaponControl.
- StdDraw : version légèrement modifiée de la librairie graphique StdDraw que vous connaissez.
- Vector2 : représente une paire d'éléments.

Comment aborder ce projet ?

- Dans un premier temps, nous vous recommandons de lancer le code que nous vous fournissons, et de passer du temps à « jouer » avec et à bien comprendre ce qu'il fait.
- Pour cela, lisez les codes des classes fournies, et faites des petites modifications pour vérifier que vous comprenez ce qui se passe. Par exemple, comment modifier l'énergie initiale du réacteur ? De l'Arsenal ? Comment modifier la touche qui sélectionne un membre d'équipage ? Cette phase initiale de compréhension devrait vous prendre entre 1h et 2h minimum.
- A partir de là, vous avez plusieurs points de départ possible :
    - Si vous préférez la conception objet, vous pouvez commencer par déterminer les classes qui manquent pour couvrir toutes les règles du jeu, et en implémenter de premières versions.
    - Si vous préférez l'algorithmique, vous pouvez vous attaquer à la fonction processHit(…) qui gère les collisions de projectiles avec les vaisseaux.
    - Si vous préférez écrire rapidement du code, vous pouvez implémenter les fonctions de visée et de téléportation des personnages dans les salles du vaisseau
- Vous avez un certain nombre de tâches à réaliser pour que votre jeu suive toutes les règles énoncées ci-dessus. Nous vous conseillons de préparer un document partagé avec votre binôme où vous :
    - Listerez les tâches à réaliser
    - Vous répartirez les tâches à faire entre vous
    - Marquerez au fur et à mesure votre avancement

Consignes

- La date de rendu de votre travail est le 15/12/2019 à 23h59.
- Le travail est à faire en binôme
- La qualité d'expérience du jeu prime sur sa complexité : un jeu avec moins de fonctionnalités mais qui tourne sans bugs sera préféré
- Le code doit être clair et bien documenté (utilisation impérative du format javadoc)
- Aucun projet comportant du code copié d'internet ou d'autre projets ne sera accepté ->**dans ce cas la note sera 0**
- Il est obligatoire d'utiliser les Collections Java pour réaliser ce projet : cette fois on ne veut pas voir de listes chaînées faites à la main !
- Voici la grille de notation (susceptible de changer sans préavis) :
    - **Projet implémentant toutes les règles décrites plus haut, avec un code clair et commenté : 14/20**
        - Chaque proposition de la partie «Pour aller plus loin» ci-après qui sera implémentée pourra apporter de 1 à 3 points de bonus en fonction de la qualité du travail fourni et de la difficulté de la tâche réalisée

- Pour toute amélioration que vous faites : indiquez-la au moment du rendu, afin que votre chargé de TP puisse la prendre en compte et la tester !

Pour aller plus loin

Nous vous proposons ci-dessous une liste d'améliorations de votre jeu, qui vous permettront de faire un jeu plus riche et plus proche du jeu original…tout en gagnant des points !

- Dans le jeu original, les vaisseaux peuvent avoir plus de 4 salles, avec de nombreuses organisations possibles des salles donnant des formes différentes de vaisseaux. Les salles sont entourées de murs et pour y accéder les membres de l'équipage doivent passer par des portes : on ne peut pas téléporter les membres d'équipage, il faut les déplacer de salle en salle (avec les flèches haut/bas/droite/gauche). Implémentez ce système, en proposant plusieurs plans de vaisseaux pas trop simples (une dizaine de salles) qui permettront de bien tester que votre mécanique de déplacement des membres d'équipage fonctionne. Cette amélioration peut rapporter jusqu'à 3 points.
    - Pour aller encore plus loin sur cette amélioration, vous pouvez rajouter l'ouverture/fermeture des portes à la souris, la gestion de l'oxygène dans les salles, et la gestion des incendies se propageant dans les salles du vaisseau, en vous basant sur les vidéos et documentations du jeu original, et en simplifiant si nécessaire. C'est difficile à faire : ne le tentez que si vous êtes très en avance !
- Une autre option est de rajouter des armes du jeu original (https://ftl.fandom.com/wiki/Weapons). Les armes doivent se comporter, autant que possible, comme indiqué dans la documentation, et vous devez nous donner le moyen de les tester rapidement (ex : avoir un vaisseau équipé de ces armes ou pouvoir les obtenir par cheatcode). En fonction de la difficulté de programmation de l'arme ajoutée, vous gagnerez entre 1 et 2 points.
- De la même façon, vous pouvez choisir de rajouter des modules du jeu original (https://ftl.fandom.com/wiki/Systems). Regardez par exemple « Hacking » ou « Drones ». Les systèmes doivent se comporter, autant que possible, comme dans le jeu original et vous devez nous donner le moyen de les tester rapidement (ex : avoir un vaisseau équipé de ces systèmes ou pouvoir les obtenir par cheatcode). En fonction de la difficulté de programmation du module ajouté, vous gagnerez entre 1 et 3 points.
- Vous pouvez aussi créer des catégories de membres d'équipage, chaque catégorie étant spécialisée sur un des types de module du vaisseau. Par exemple, un spécialiste du Moteur doublera le bonus Moteur lorsqu'il sera dans la salle du Moteur, mais aura un bonus deux fois plus faible lorsqu'il sera dans une autre salle. Il faudra distinguer graphiquement les différents spécialistes, et nous permettre de tester rapidement leur influence (cheatcode pour avoir des membres d'équipage du bon type). Cette amélioration peut être assez simple à réaliser, et vous rapportera entre 1 et 2 points suivant la qualité de votre réalisation.
- Vous pouvez vous focaliser sur l'IA de l'adversaire afin de la rendre moins naïve : vous pouvez implémenter des fonctionnalités pour qu'elle choisisse de manière pertinente les modules du vaisseau du joueur à viser, et pour qu'elle répartisse correctement son énergie et ses membres d'équipage. Pour cette amélioration, certaines solutions simples peuvent donner de premiers résultats convaincants (mieux que le hasard), il est également possible d'aller sur des approches beaucoup plus complexes…Attention de ne pas faire « d'usine à gaz » illisible ! Cette amélioration vous permettra de gagner entre 1 et 3 points en fonction de la difficulté de votre réalisation et de la qualité de votre travail.
- Dans le jeu original, le vaisseau se déplace entre différents secteurs (https://ftl.fandom.com/wiki/Sector). Suivant les secteurs, il peut rencontrer des ennemis, des dépôts de ressources ou des magasins permettant d'acheter des armes, des réparations ou des membres d'équipage. Vous pouvez implémenter une carte de secteurs et la navigation entre secteurs, ce qui vous demandera de revoir un peu la boucle principale du jeu. Cette amélioration, de difficulté moyenne, vous rapportera entre 1 et 2 points (voire plus en fonction de votre réalisation).
- Toute autre amélioration provenant du jeu original ou de votre imagination est la bienvenue !