ENSTA BRETAGNE

Programmation orientée objet en Java

Projet

Hiba HNAINI
Sylvain Guérin

Année scolaire 2022-2023

Indications pour bien réussir votre projet

Recommandations et consignes pour la formation des groupes et l'organisation du travail

1. Veuillez faire des groupes de 3 étudiants au plus.
2. Le mode Agile s'invite ici pour éviter l'effet tunnel et pour vous aider à livrer des versions exécutables dès le début !
3. Donnez-vous les moyens de travailler correctement. Prenez le temps de vous organiser et de travailler en utilisant les outils qu'il vous faut : un bon IDE (IntelliJ Ultimate ou Eclipse), un outil de gestion de versions et de travail collaboratif (Git et GitHub) …
4. Les **difficultés avec votre ordinateur**, **vos outils** (performances, configuration, bugs, licences, ...) **et votre équipe de travail** (horaires de travail, répartition des responsabilités, manque de confiance, manque d'engagement et de communication ...) **sont de la responsabilité de l'étudiant ou de l'équipe et NON des enseignants**.

Recommandations et consignes pour la réalisation du projet

1. Réfléchissez bien avant de coder : attention à la modélisation, au découpage, à la « duplication » de code, … Ceci est un cours de PROGRAMMATION ORIENTEE OBJET et non de programmation impérative.
2. Testez, testez et... testez encore (mieux : écrivez des tests !).
3. Pensez à rédiger une javadoc parfaite (la compiler et la relire). La javadoc ne fait que "formater" la documentation que vous avez renseignée dans le code source. Cela ne sera pas magique si vous ne documentez pas bien les classes, méthodes, paramètres, les exceptions propagées, … de votre application. Voici un tutoriel présentant l'outil Javadoc de Sun, qui permet de générer les documentations d'un code Java : https://simonandre.developpez.com/tutoriels/java/presentation-javadoc/
4. Lorsque vous êtes satisfaits de votre application, prenez le temps, à tête reposée, de relire tout le code. Interrogez-vous, étonnez-vous, vérifiez la cohérence entre les classes, les éventuelles redondances de code, et le besoin d'abstraire ou de rendre générique, réfléchissez aux éventuelles modifications futures et à leurs répercussions sur l'évolution du code, ...
5. Je vous conseille ensuite de soumettre votre code à Sonarcube (et/ou Codacy) ; ça devrait vous permettre d'encore améliorer votre code. Ces outils vous permettront de faire une analyse statique qui a pour but de mesurer la qualité du code de vos applications et de vous fournir des métriques portant sur la qualité du code et permettant d'identifier précisément les points à corriger (code mort, bugs potentiels, non-respect des standards, manque ou excès de commentaires...). L'utilisation de ce type d'outils ne remplace en aucun cas les tests unitaires, mais permet d'identifier rapidement certains défauts du code.

Recommandations pour la partie rapport et rendu du travail

Voici quelques recommandations pour préparer et rendre vos projets :

1. Vous rendrez votre travail en exportant votre projet Java en format .zip. Préciser le nom du fichier .zip en y incluant les noms des étudiants de l'équipe.
2. Préparez la documentation de votre projet : l'architecture (au moins le diagramme représentant les modules/composants et classes de votre projet) doit être faite avant de commencer à coder ; à chaque fois que vous avancez sur votre projet, préparez votre javadoc et documentez ce que vous venez de faire et la manière dont vous l'avez faite (dans un fichier partagé entre tous les membres de l'équipe).
3. Faites un rapport avec, au moins :
   a. Une page de garde avec l'information de l'équipe de travail
   b. Le cahier de charges

c. Les nouvelles fonctionnalités que vous avez ajoutées au cahier de charges de base
d. L'architecture du logiciel (au moins le diagramme représentant les modules/composants et classes de votre projet)
e. Garanties de qualité telles que les tests, le plan de couverture des tests, des algorithmes d'optimisation si vous en avez implémentés, comment vous avez réussi à améliorer la généricité et l'extensibilité de vos implémentations, comment vous avez rendu votre projet facile à évoluer…)
f. Méthode de travail (processus suivi, planning du projet, distribution des responsabilités, comment vous avez travaillé ensemble ...)

4. Déposer sous le Moodle du cours (si cette application ne fonctionne pas, veuillez envoyer vos projets dans un email aux enseignants du cours et UN LIEN POUR TÉLÉCHARGER LE PROJET ET LA DOCUMENTATION : surtout, n'incluez pas le projet en tant que pièce jointe de votre courriel, car les filtres de sécurité du serveur SMTP ne laisseront pas passer du code source) le fichier .zip et toute la documentation (ficher .docx ou PDF) de votre projet.
5. Veuillez respecter le dernier délai pour déposer votre projet (voir le planning dans les slides du cours)

Conseils et consignes pour la présentation et démonstration du projet

Vous avez ***20 minutes de présentation et 10 minutes de questions par équipe*** pendant les deux dernières séances du cours pour faire la présentation (en équipe) de vos projets. La note et les commentaires la justifiant vous seront donnés à la fin de chaque présentation/démonstration.

1. Préparez-vous pour la présentation et démonstration de votre projet (avez-vous les connecteurs qu'il faut ? avez-vous assez de batterie pour faire la présentation sans interruptions ? vous êtes-vous coordonnés pour l'utilisation de la parole (qui présente quoi) ? …)
2. Respectez le temps alloué
3. Pour la partie présentation, préparez des slides présentant de manière succincte le cahier des charges, le problème que vous voulez résoudre et en particulier les nouvelles fonctionnalités que vous avez ajoutées au cahier de charges de base, l'architecture du logiciel (au moins le diagramme représentant les modules et classes de votre projet), des éléments de qualité (si vous avez fait des options donc les algorithmes d'optimisation, de généricité, d'extensibilité, etc…) et des éléments méthodologiques (planning du projet, distribution des responsabilités, comment vous avez travaillé ensemble, refactoring du code...)
4. Pour la démonstration il faudra montrer en quoi le programme correspond au cahier de charges en déroulant des scénarios de simulation ; veuillez avoir un esprit critique et objectif de l'outil obtenu, du code produit, et veuillez présenter des perspectives d'évolution.
5. Il vous faudra répondre aux questions de manière claire, concise et avec du recul sur votre projet.

Projet : Duel Invaders

Space Invaders est connu comme un des premiers jeux d'arcade. Il a largement influencé nombre de ses successeurs. Taito Corporation a sorti Space Invaders en 1978. À la fin de l'année 1979, il était devenu un phénomène mondial et la première ère des jeux d'arcade avait commencé. Le jeu était disponible à la fois en version standard et en version table-top. Les commandes étaient à l'origine des boutons pour se déplacer à gauche, à droite et tirer. Les boutons de déplacement ont été remplacés par un joystick dans les versions ultérieures. La tâche de ce projet est d'écrire une version de Space Invaders pilotée par le clavier, pour deux joueurs, appelée Duel Invaders. Cela implique que votre jeu est principalement basé sur les mêmes mécanismes de jeu que Space Invaders.

Game Mechanics

Le jeu est représenté sur la figure 1. Il consiste en deux armadas d'aliens, orientées dans des directions opposées, et de deux canons laser, un en haut de l'écran et un en bas de l'écran. Les deux armadas d'aliens commencent dos à dos au milieu de l'écran et se déplacent vers des extrémités opposées de l'écran. Le but du jeu consiste pour chaque joueur à tirer sur les aliens avec leurs lasers tout en évitant les tirs des aliens et en empêchant une invasion. Les joueurs peuvent s'entraider pour détruire les aliens.

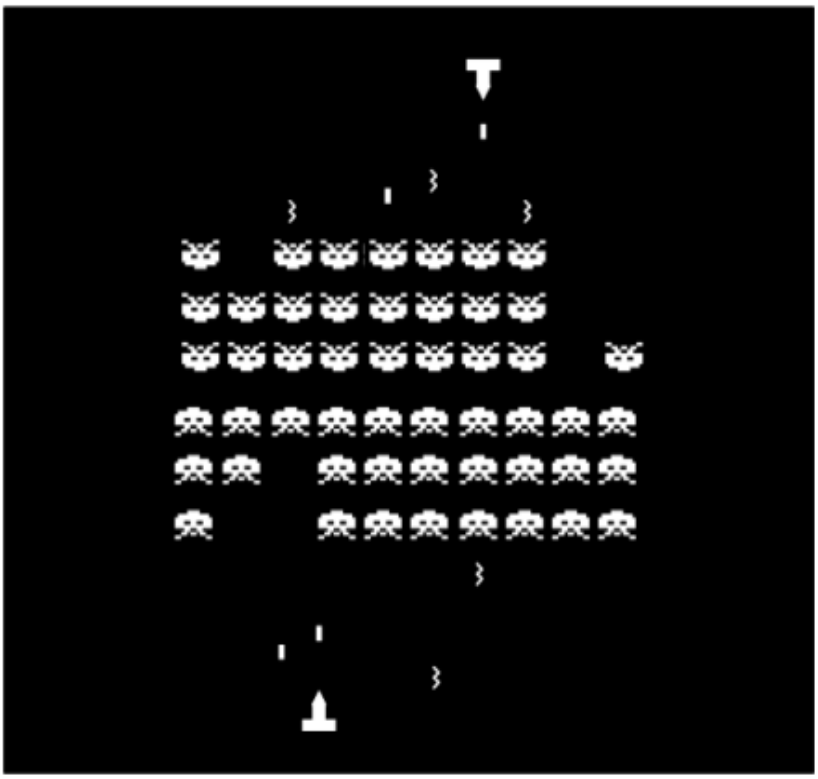


Figure 1: Duel Invaders

Déplacements

L'armada alien verticale part de la gauche de l'écran et se déplace horizontalement de gauche à droite. Lorsqu'elle atteint le côté droit de l'écran, elle descend d'une rangée et se déplace de droite à gauche. Lorsqu'elle atteint le côté gauche de l'écran, elle descend d'une rangée et se déplace à nouveau vers la droite. L'armada orientée vers le haut implante des mêmes déplacements en se dirigeant vers le haut de l'écran. Les

canons laser des joueurs ne peuvent se déplacer qu'horizontalement. Les tirs des joueurs et des aliens sont limités aux mouvements verticaux.

Tirs

Les canons laser des joueurs et les extraterrestres sont capables de tirer. Voici les règles qui régissent les tirs :

- Si le tir d'un joueur touche un alien, l'alien et le tir sont détruits. Le tir peut toucher des aliens appartenant à l'une ou l'autre armada.

- Si le tir d'un joueur touche le canon laser d'un autre joueur, le tir et le canon de l'autre joueur sont détruits.

- Si le missile d'un alien touche le canon laser d'un joueur, le canon laser et le missile sont détruits.

- Si le tir d'un joueur touche le missile d'un alien, les deux sont détruits.

Fin de partie

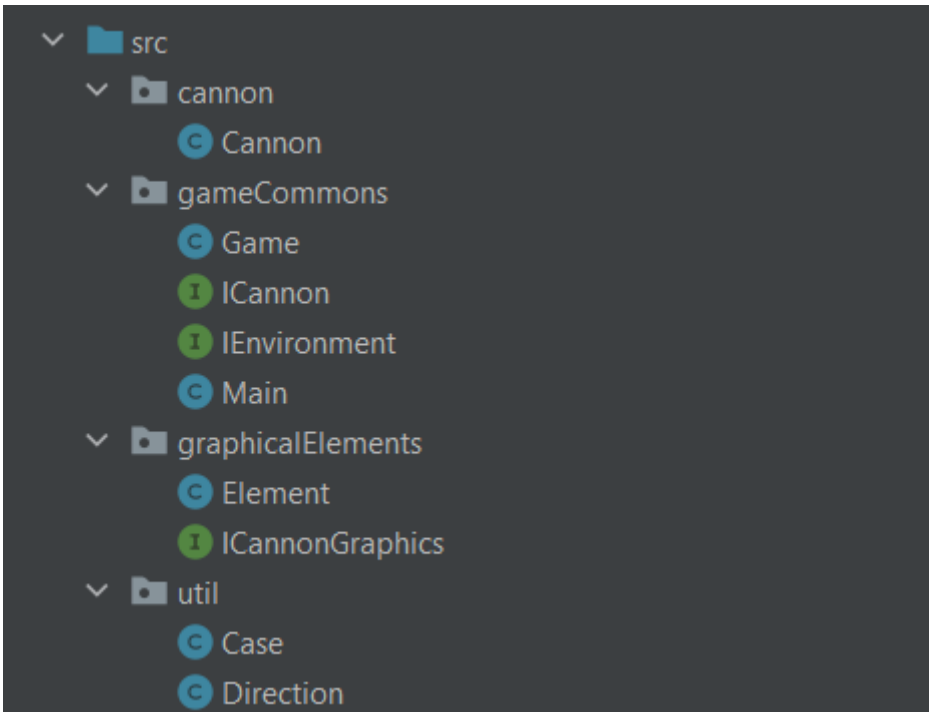
La partie se termine lorsque l'une des situations suivantes se produit :

- Les extraterrestres envahissent en atteignant l'une des rangées dans lesquelles se trouvent les lasers des joueurs.

- L'un des canons laser des joueurs est détruit.

- Les joueurs gagnent en détruisant tous les aliens.

Voici une proposition (c'est juste une proposition que vous pouvez suivre ou pas selon la portée et les améliorations que vous voulez apporter au projet) pour commencer à organiser votre projet :



- cannon : le package devra contenir la ou les classes gérant le canon. Il s'y trouve par exemple la classe **Cannon**
- graphicalElements : contient la gestion de l'interface graphique. Voici quelques fichiers qui pourront se trouver dans ce package :
    - L'interface **ICannonGraphics** avec les méthodes disponibles pour l'utilisateur

    - La classe **Element** (implémentant l'interface **ICannonGraphics)** utile, par exemple, pour faciliter la création d'une case et une couleur pour demander d'afficher un carré de cette couleur à des coordonnées données.
- gameCommons : contient les entités permettant de gérer la partie de manière générale, avec par exemple :
    - Les interfaces **ICannon** et **IEnvironment** pour décrire les actions qui doivent être disponibles pour le contrôle des tirs et de l'environnement respectivement.
    - La classe **Game** pour gérer une partie de jeu ; elle ne dépend que des interfaces modélisant le jeu et ne doit pas dépendre directement des classes concrètes que vous implémentez.
    - La classe **Main** pour de lancer le jeu avec certains paramètres,
- util : contient, par exemple, les définitions de types **Case** et **Direction** utiles à travers tout le projet. Une entité de type **Direction** contiendra par exemple une énumération de 4 valeurs pour décrire à chaque moment la direction de le canon : RIGHT et LEFT, désignées par Direction.RIGHT et Direction.LEFT.

L'environnement

Le schéma proposé se compose des éléments suivants :

— un environnement global, composé des aliens

— ligne d'aliens a un sens (de gauche à droite ou de droite à gauche) et une vitesse. Elle contient un ensemble d'aliens.

Sauvegardez une version stable de votre projet une fois cette étape finie. Vous devrez présenter une version fonctionnelle de cette partie en soutenance, et présenter vos choix et difficultés.

Les canons

Dans cette partie, vous allez devoir coder Cannon et Game.

Vous devez implémenter toutes les interfaces données.

Sauvegardez une version stable de votre projet une fois cette étape finie. Vous devrez présenter une version fonctionnelle de cette partie en soutenance, et présenter vos choix et difficultés.

Timer

Affichez le temps de jeu à la fin de la partie, en particulier en cas de victoire.

Jeu infini

Vous allez maintenant écrire des variantes des classes précédentes pour jouer à un jeu « légèrement différent » : des aliens sont ajoutés jusqu'à ce qu'un des deux canons soit détruit.

Après avoir réfléchi à comment l'implémenter, créez de nouvelles classes CannonInf et EnvInf pour gérer ce nouveau mode de jeu. Vous pouvez modifier d'autres classes. Vous êtes ici autorisés à modifier les interfaces, à bon escient et en assurant la compatibilité avec les modules précédemment développés ! Il n'est pas nécessaire

de modifier le package graphicalElements. On pourra de plus ajouter un score, affiché à la fin de partie, correspondant à la plus haute ligne jamais atteinte.

Deux joueurs

Permettre de jouer à deux, sur un clavier (un joueur se déplace avec les flèches, l'autre avec QZSD). Il faudra alors instancier deux objets différents de Cannon, et modifier la classe CannonGraphic pour écouter d'autres touches, tout en sachant lesquelles correspondent à quel canon. Vous pouvez garder un affichage en grille simple comme dans les premières parties, n'hésitez pas à en changer la résolution.

Logs (traces)

Enregistrer les résultats du jeu et les étapes dans un fichier

Tests

Ajouter les tests nécessaires pour tester le jeu

Environnement 2D libres

On cherche à généraliser le principe du jeu à d'autres types d'environnements. Le principe reste de tirer sur les aliens. Vous pouvez ajouter un type d'extraterrestres qui explosent et suppriment ceux qui les entourent.

Meilleur rendu graphique

Le rendu graphique est très basique, vous pouvez l'améliorer, minimalement ou énormément. Assurez-vous cependant avant cela d'avoir des parties 1, 2 et 3 fonctionnelles ! Attention, le débogage peut être potentiellement laborieux et plusieurs notions sortent du cadre du cours.