Projet Java III - 2024-2025

HELBArmy Seconde Session

Introduction :

Dans le cadre du cours de Java III, il vous est demander d'implémenter une simulation de batailles militaires : HELBArmy.

Dans un monde en guerre économique, deux armées s'affrontent pour dominer un territoire stratégique. Chaque camp envoie ses troupes — collecteurs, semeurs et assassins pour prendre l'avantage.

Description des éléments de la simulation :

L’espace de jeu :

La simulation se déroule sur une carte contenant un certain nombre de lignes et de colonnes. Chaque élément de la simulation possède une position sur la carte. Sauf exception, il ne peut jamais y avoir deux éléments à la même position sur la carte. Les bords de la carte sont des murs que les unités ne peuvent pas franchir.

Les arbres :

La carte contient un certain nombre d'arbres placés aléatoirement. Les arbres permettent de collecter du bois. Chaque arbre initialement présent sur la carte possède une quantité de bois de 50 unités. Un arbre peut en contenir au maximum 100 unités de bois. Lorsqu'un arbre est entièrement récolté, il disparaît de la carte. Tant qu'il est présent, il constitue un obstacle infranchissable.

Les rochers :

La carte contient un certain nombre de rochers placés aléatoirement. Ils permettent de collecter du minerai. Chaque rocher possède une quantité initiale de 100 unités de minerai et peut en contenir au maximum 200. Lorsqu'un rocher est entièrement collecté, il disparaît de la carte. Tant qu'il est présent, il constitue un obstacle infranchissable. Contrairement aux autres éléments du jeu, les rochers occupent plusieurs cases, chacun couvrant une zone de 2x2 cases.

Les villes :

Les villes sont des éléments placés sur la carte et capables de générer des unités. Initialement, deux villes adversaires sont présentes sur la carte, situées respectivement en (0,0) et (n,m), où n représente le nombre de lignes et m le nombre de colonnes de la carte.

Les unités :

Les unités sont générées par les villes. Chaque unité occupe une case sur l’espace de jeu.

Les collecteurs :

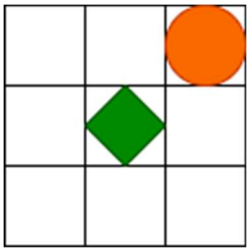
Les collecteurs apparaissent en orange sur la carte. Ils permettent de collecter des ressources. Une fois générés, ils se dirigent vers la ressource la plus proche pour la récolter.

La collecte s'effectue de la manière suivante : le collecteur doit se placer sur une des cases adjacentes à la ressource, et une unité de ressource est collectée par seconde. Au fur et à mesure de la collecte, la quantité de ressources diminue.

Quand une ressource a été collectée par le collecteur, elle s’ajoute automatiquement aux ressources disponibles dans la ville à laquelle le collecteur

appartient. Il n’est donc pas nécessaire que le collecteur retourne en ville pour déposer ses ressources.

Le schéma suivant illustre une situation de collecte du bois d’un arbre par un collecteur. Sept autres positions de collecte sont encore disponibles pour d’autres collecteurs sur le même arbre.



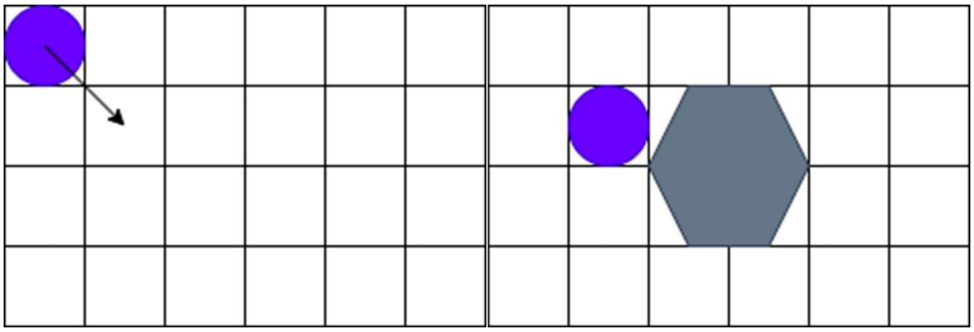
Il existe deux types de collecteurs : les bûcherons et les piocheurs. Le bûcheron possède un bonus de récolte sur les arbres : il peut collecter 2 unités de bois par seconde lorsqu'il exploite un arbre. Le piocheur, quant à lui, bénéficie d’un bonus de récolte sur les rochers et peut collecter 3 unités de minerai par seconde lorsqu'il exploite un rocher. Cette caractéristique n’est pas visible visuellement : il est impossible de distinguer un bûcheron d’un piocheur visuellement.

Les semeurs :

Les semeurs vaillants du rêve (abrégé semeurs) apparaissent en mauve sur la carte. Ils permettent de créer de nouvelles ressources à collecter. Le semis s'effectue de la manière suivante : le semeur doit se placer sur une case adjacente à un ensemble de cases pouvant accueillir la ressource, et donc libres de tout autre élément. Une fois en position, le processus de semis commence. La ressource apparaît alors sur la carte, empêchant toute autre unité de circuler sur cet emplacement. Chaque seconde écoulée pendant le processus de semis ajoute deux unités de ressource à la ressource en cours de création, jusqu'à atteindre sa capacité maximale.

Si le processus est interrompu, par exemple si le semeur est attaqué pendant le processus de semis, la ressource n’est plus alimentée. Ainsi, si le processus de création d’un rocher est arrêté après 10 secondes, celui-ci ne contiendra que 20 unités de minerai.

Ci-dessous, l'illustration du semis d'un rocher. Dans un premier temps, le semeur se déplace jusqu'à la position de semis. Une fois celle-ci atteinte, le processus de semis commence et un rocher apparaît sur la carte, à une position adjacente à la position de semis, sur laquelle aucun autre élément n'est présent.



Les assassins :

Les assassins apparaissent en bleu sur la carte. Ils chassent les ennemis adverses les plus proches, en commençant par les assassins adverses, puis les collecteurs, et enfin les semeurs.

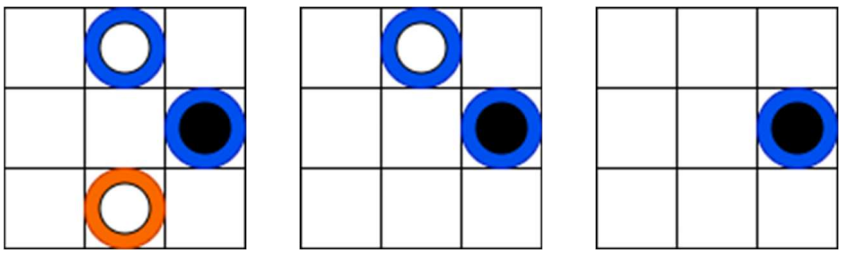
Système de combat :

Lorsqu'une unité d'armée rivale se trouve sur une position adjacente à une autre, et qu'au moins l'une de ces unités est un assassin, un combat s'engage entre ces unités.

Le combat entre deux unités adverses se situant sur des cases adjacentes s’effectue de la manière suivante : un dé à deux faces est lancé, chaque face étant associée à l’une des unités en combat. L’unité correspondant au résultat du jet de dé est éliminée et disparaît alors du terrain.

Plusieurs unités peuvent être impliquées dans un combat. Une unité peut avoir autant d'adversaires que de cases adjacentes. Dans ce cas, un dé contenant autant de faces que le nombre d’unités impliquées est lancé, chaque face étant associée à l’une des unités. L’unité correspondant au résultat du jet de dé est éliminée et disparaît alors du terrain. Ce procédé est répété jusqu’à ce qu’il n’y ait plus d’unités en combat.

Dans le schéma suivant, on assiste au combat entre trois unités, suite au déplacement de l’unité de la ville sud (affichée avec un disque central de couleur noire). Tout d’abord, le collecteur du nord est vaincu. Le combat n’est donc pas terminé, car deux unités ennemies sont encore situées sur des positions adjacentes. Le combat se termine enfin par la défaite de l’assassin du nord.

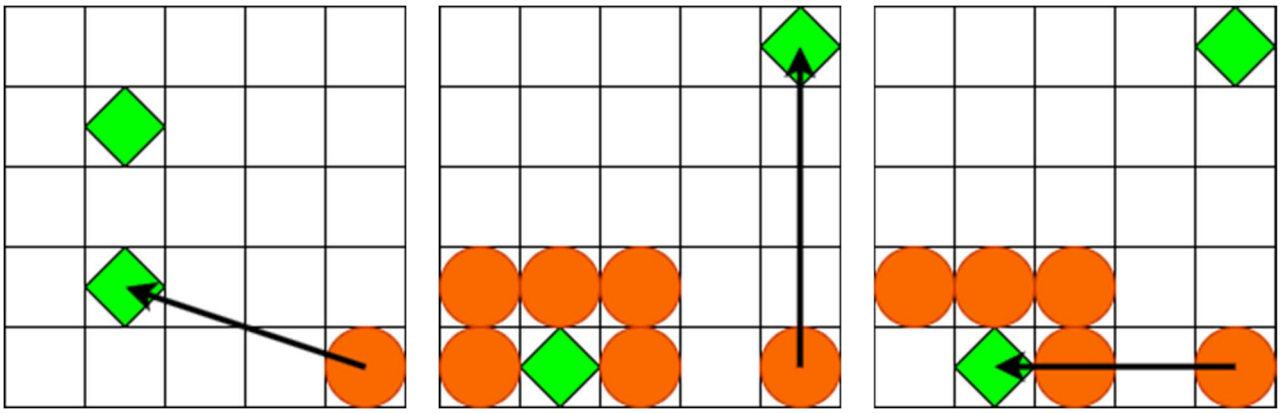


Système de déplacement :

Lorsqu'elles ne sont pas engagées dans un combat, les unités se déplacent en fonction de diverses influences.

Collecteurs : Ils se déplacent vers la ressource collectable la plus proche, c'est-à-dire une ressource dont une position de collecte est disponible.

Dans le schéma de gauche, l'arbre le plus proche dispose de positions de collecte disponibles. Le collecteur s'y dirige naturellement. Dans le schéma central, l'arbre le plus proche ne possède plus de positions de collecte libres. Le collecteur se déplace donc vers le second arbre le plus proche, qui a une position de collecte disponible. Le schéma de droite présente une situation problématique qu'il ne vous est pas demandé de résoudre : l'arbre possède une position de collecte libre, mais elle n'est pas accessible. Le collecteur se dirigera néanmoins vers cet arbre, même s'il est incapable d'y accéder.



Si aucune ressource collectable n’est disponible sur le carte, le collecteur ne se déplace pas.

Semeurs : Les semeurs décident aléatoirement de la ressource à générer.

Si cette ressource est un arbre, ils choisissent aléatoirement parmi tous les arbres présents sur la carte un arbre possédant une case adjacente libre, puis se déplacent afin de semer un arbre sur cette case. Les arbres sont donc toujours plantés à côté d’un autre arbre. Si aucun arbre n’est présent sur la carte, le semeur place alors l’arbre à une position libre aléatoire.

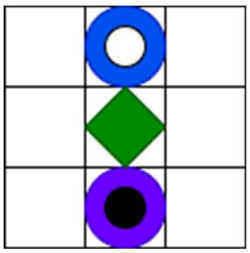
Si cette ressource est un rocher, ils choisissent la position la plus éloignée de tous les autres rochers présents sur la carte et permettant le placement d’un rocher. Ils s’y déplacent ensuite afin de semer le rocher à cette position. Une fois arrivé à la position déterminée, si le placement de la ressource n’est plus possible (par exemple, si l’emplacement n’est plus libre), le semeur décide d’une nouvelle ressource et d’une nouvelle position pour le semis.

Assassins : Les assassins pourchassent les ennemis adverses les plus proches, en commençant par les assassins adverses, puis les collecteurs adverses, et enfin les semeurs adverses. Concrètement, si des assassins adverses sont présents sur

la carte, l'assassin se dirige vers celui qui est le plus proche. Si aucun assassin adverse n'est présent, l'assassin se dirige vers le collecteur adverse le plus proche. Si aucun collecteur adverse n'est présent non plus, il se dirige vers le semeur adverse le plus proche. Si aucun ennemi n'est présent, l'assassin choisit une position aléatoire parmi toutes les positions libres sur la carte et s'y dirige pour s'y arrêter jusqu'à ce que l'apparition d'un ennemi le pousse à reprendre son déplacement.

Déplacement et gestion des obstacles :

Chaque unité peut se déplacer horizontalement, verticalement, ou en diagonale, à condition que la case du déplacement soit une case ne contenant pas un autre élément du jeu. Il ne vous est pas demandé de gérer les contournements d'obstacles. Par exemple, dans cette situation, l'assassin pourchassant le semeur reste bloqué par l'arbre.



Vous êtes toutefois libres si vous le désirez d'implémenter le code permettant le contournement des obstacles.

Vitesses et gestion du temps :

Toutes les unités se déplacent à la même vitesse. Par défaut, tous les éléments du jeu effectuent une action par seconde. On peut généraliser en disant que le jeu rafraichi les éléments une fois par seconde. Il doit être possible de modifier cette vitesse de rafraichissement. Par exemple, il doit être possible, depuis le code, de rafraichir le jeu une fois toutes les 0,5 secondes afin d'accélérer le déroulement de la simulation. Le drapeau est le seul élément dont le temps fonctionnel ne peut pas être impacté par le changement de la vitesse de rafraîchissement.

Génération des unités :

Les unités sont générées de manière aléatoire par les villes. Toutefois, ce choix aléatoire est soumis aux influences suivantes :

- Plus il y a d'arbres présents sur la carte, plus les chances de générer un collecteur bûcheron sont grandes.
- Plus il y a de rochers présents sur la carte, plus les chances de générer un collecteur piocheur sont grandes.
- Plus il y a d'assassins adverses sur la carte, plus les chances de générer un assassin sont grandes.

Toutes les deux secondes, une unité est générée par la ville. La génération des unités ne coûte pas de ressources.

Les unités sont générées à partir des villes, aléatoirement, sur l'une des cases libres les plus proches de la ville. Dès lors, si l'unité ne peut pas être générée sur une des cases situées à une distance de 1, elle sera générée sur une des cases situées à une distance de 2, et ainsi de suite jusqu'à une distance de 3, etc.

Collectables :

Le Drapeau : Toutes les deux minutes, un drapeau apparaît à une position aléatoire sur la carte et reste présent pendant 10 secondes avant de disparaître automatiquement. Lorsque le drapeau est présent, toutes les unités modifient leur comportement pour se déplacer aléatoirement sur la carte. Si le drapeau disparaît ou est fortuitement collecté par une unité, toutes les unités reprennent leur comportement normal.

La Pierre Philosophale (la quoi ?) : La pierre philosophale est une pierre philosophale dotée de pouvoirs philosophales qui agit comme un portail. Lorsqu'une unité entre en contact avec la pierre philosophale, l'unité est téléportée à une position aléatoire sur la carte. La pierre philosophale ne disparaît pas quand elle est collectée, elle reste à son emplacement. Initialement, il n'y a pas de pierres philosophales présentes sur le terrain.

Fin de partie :

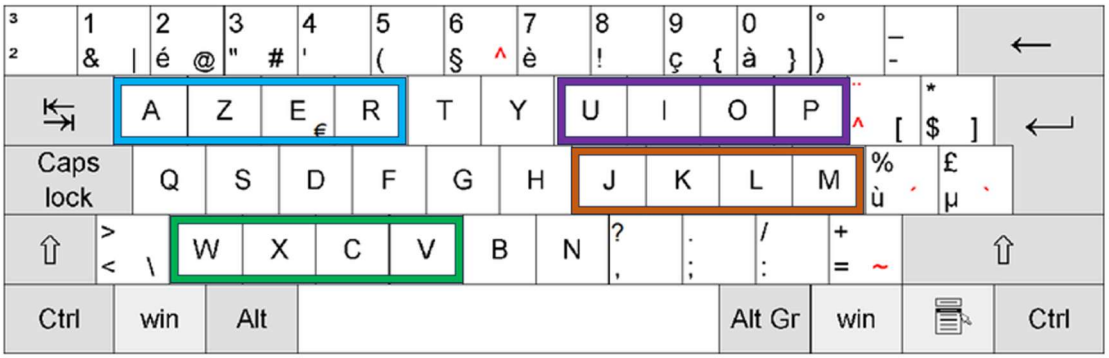
À l'issue de la partie, lorsque le programme est terminé, soit lors de la fermeture de la fenêtre de jeu, soit par l'utilisation d'un cheat code associé, les ressources en bois et en minerai des deux équipes sont affichées en console. Si une des équipes possède à la fois l'avantage sur la quantité de minerai et de bois, celle-ci est explicitement désignée comme vainqueur. Sinon il s'agit d'une égalité.

Commandes interactives – Cheat codes :

Afin de permettre des interactions avec la simulation, il existe un système de commandes permettant d'influencer la partie. Quand l'utilisateur appuie sur une touche, cela déclenche un évènement précis :

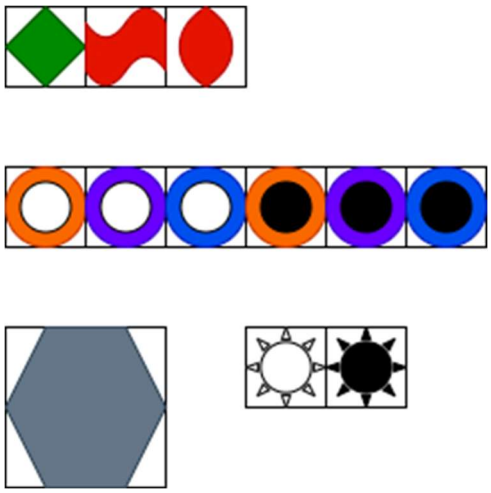
a	Génère instantanément un collecteur dans la ville nord.
z	Génère instantanément un semeur dans la ville nord.
e	Génère instantanément un assassin dans la ville nord.
r	Génère instantanément une unité de type aléatoire dans la ville nord.
w	Génère instantanément un collecteur dans la ville sud.
x	Génère instantanément un semeur dans la ville sud.
c	Génère instantanément un assassin dans la ville sud.
v	Génère instantanément une unité de type aléatoire dans la ville sud.
j	Tue instantanément tous les collecteurs.
k	Tue instantanément tous les semeurs.
l	Tue instantanément tous les assassins.
m	Tue instantanément toutes les unités présentes sur la carte.
u	Retire instantanément de la carte, tous les éléments à l'exception des unités.

i	S'il n'y en a pas déjà un, un drapeau apparait sur la carte.
o	Arrête la simulation. Fin de la partie et affichage du résultat dans la console.
p	Fait apparaitre une pierre philosophale à une position aléatoire.



Aspect graphique :

Il vous est demandé de respecter la charte graphique suivante pour l'affichage des villes, ressources, unités et des collectables (disponible sur eCampus et éditable sur draw.io) :



Ci-dessus, la première ligne représente, de gauche à droite, un arbre, un drapeau et une pierre philosophale. La seconde ligne représente les unités des équipes du nord et du sud. Finalement sur la dernière ligne, un rocher, suivit des villes du nord et du sud.

Notes Importantes :

Les contraintes et fonctionnalités du projet sont **susceptibles d'évoluer au cours du temps**. Pensez donc à adapter une stratégie de développement adéquate.

Certains points de la description ne sont pas précisés ou sont laissés volontairement vagues. Il revient à vous de faire certains choix d'interprétations. Veillez toutefois à ce que votre approche soit logique et justifiée.

Contrainte de développement :

- Votre programme devra être compilable et exécutable dans un environnement Linux Ubuntu tel qu'une des machines virtuelles utilisées en cours et disponible sur le SharePoint d'eCampus. Un script bash nommé « run.sh » devra permettre la compilation et l'exécution de l'application en utilisant seulement la commande suivante «bash run.sh » en terminale. Si le fichier n'est pas fourni, ou si la compilation et l'exécution ne fonctionnent pas dans l'environnement spécifié, le projet ne sera pas corrigé et sanctionné d'un zéro. **Testez donc avant que tout fonctionne !**

- Votre code java devra être **compatible avec la version de l'openjdk** disponible sur cette même machine virtuelle.

- Votre code devra respecter les principes de designs orientés objet comme vu au cours. Pensez donc à faire des choix logiques de design de classes afin de produire un **code propre et maintenable**.

- Votre code devra présenter **une structure correcte et maintenable**. Notamment :
  - Evitez la duplication de code.
  - Evitez les constantes magiques.
  - Evitez le code mort.
  - Nommez correctement vos variables, méthodes et classes et organisez correctement votre code.

  Il est fortement conseillé de consulter la partie du cours sur les bonnes pratiques de développement. Un code dont la qualité sera jugée insuffisante **sera sanctionné d'un zéro**.

- Votre jeu devra être développé en utilisant **exclusivement la librairie graphique JavaFX** comme vu en cours **et à partir du jeu du SnakeFX** qui devra être utilisé comme base fonctionnelle. Toutes les ressources disponibles sur eCampus restent également mobilisables.

- A l'exception des commentaires, l'entièreté de votre programme **devra être codé en anglais** (nom de variables/fonction/classes/etc…).

- Votre code doit être suffisamment commenté. Pensez donc, pour chaque méthode « complexe », à au minimum, expliquer ce que représentent les paramètres, ce que fait la méthode et ce qui est retourné. Un code insuffisamment commenté ne sera pas évalué et **sera sanctionné d'un zéro**.

Rapport :

Il vous est demandé de rédiger un rapport décrivant votre projet. Votre rapport devra contenir au minimum les sections suivantes :

Introduction : Cette section devra introduire votre projet. Décrire ce qui a été réalisé et présenter brièvement la structure de votre rapport. (max 1 page)
Fonctionnalités de base : Cette section devra expliquer les fonctionnalités offertes par votre application d'un point de vue à la fois fonctionnel et technique. (max 5 pages)
Analyse : Cette section devra expliquer la structure de votre implémentation en utilisant les outils d'analyses déjà vus durant votre parcours. Si aucun outil n'a été vu pour l'instant, considérez qu'il vous est demander d'élaborer des schémas explicatifs sur la structure de votre code afin d'en expliquer les choix de designs logiciels. Attention : Tous les diagrammes doivent être commentés ! (max 3 pages)
Limitations : Les limites de votre application, par exemple : dans quels cas d'utilisation votre application pourrait ne pas fonctionner comme prévu ? Y a-t-il des aspects techniques qui n'ont pas été traité ? Si vous aviez plus de temps pour le projet, qu'auriez-vous amélioré ? Plusieurs points de vue sont possibles, il revient à l'étudiant de choisir les points qu'il considère les plus pertinents pour réaliser son autocritique. (max 2 pages)
Conclusion : Votre conclusion sur le projet. Ce que vous avez réussi à faire ou non durant le projet et les apprentissages que vous en tirez. (max 1 page)

Il vous est demandé de respecter le nombre maximum de pages par sections. La taille de la police d'écriture devra obligatoirement être supérieur ou égale à 12 et inférieur ou égale à 14 pour le contenu et les titres. Une grille d'auto-évaluation permettant de mieux comprendre les attendus de l'enseignant concernant votre rapport vous sera possiblement communiquée.

Maitrise des productions :

L'évaluation du projet vise à attester de la bonne maitrise, des concepts liés au cours et au développement du projet, par l'étudiant. Toute réalisation pour laquelle l'étudiant ne peut pas démontrer une maîtrise suffisante lors de l'évaluation orale ne sera pas prise en considération.

Deadline et remise :

La date limite pour la remise du projet est le **dimanche 17 aout à 23h59**. Le projet devra être déposé sur eCampus à l'intérieur d'un fichier **.zip** contenant toutes les sources de votre projet ainsi que le rapport au format **.DOCX**.
Attention certaines **activités obligatoires** autours du projet pourraient être annoncées **hors session**.

Développement et Triche (1/2) :

- Tout acte de triche sera sanctionné par **une note de fraude au bulletin et sera notifié à la direction qui pourra possiblement décider de sanctions supplémentaires**.

- Des parties de code réutilisés d'un projet existant (d'un autre étudiant ou disponible sur le net) sans références dans votre rapport et sans mention de l'utilité du code utilisé est toujours considéré comme une fraude.

- Pour ce projet, **vous ne pouvez pas reprendre des parties du code d'un autre étudiant, de cette année, ou d'une année précédente**.

- Pour ce projet **vous ne pouvez pas vous inspirer/servir d'un jeu/code disponible sur internet.**

- Si vous avez un doute, contactez l'enseignant le plus tôt possible afin d'éviter du refactoring inutile, ou pire, **une note de zéro/fraude**.

Développement et Triche (2/2) – Utilisation de LLMs :

- Pour ce projet, le seul LLM qu'il vous est permis d'utiliser est ChatGPT dans sa version gratuite. Son utilisation reste toutefois soumise à la condition suivante : vous devez créer un compte dédié au projet avec l'adresse électronique suivante, à créer également : nom.prenom.java.q3.2425@gmail.com et dont le mot de passe sera fourni à l'enseignant. Tous les prompts effectués en rapport avec le projet (implémentation et rapport) devront être fait sur ce compte. À tout moment l'enseignant doit pouvoir avoir une vue **rapide** sur **l'historique des prompts réalisés**. En particulier le jour de l'évaluation finale du projet. Il revient à l'étudiant d'assurer l'accès à cet historique. Tout manquement à cette consigne pourra entrainer **une note de zéro/fraude**. Ces prompts devront rester disponibles pour toute la durée de la session, jusqu'à la délibération de celle-ci. Attention, bien que ChatGPT soit autorisé, il ne vous est pas permis

d'inclure dans votre projet des éléments autre que les briques fonctionnelles déjà fournies par le SnakeFX et les codes ressources disponible sur eCampus. Si vous incluez dans votre implémentation des éléments proposés par ChatGPT qui vous feraient sortir du cadre fixé par les contraintes de cet énoncé **vous serez pénalisé**. Notez que pour votre propre apprentissage, l'enseignant vous recommande de ne pas y recourir, ou du moins, d'essayer dans un premier lieu de vous en passer. Il est extrêmement déconseillé d'intégrer un résultat donné par ChatGPT dans votre travail sans en avoir fait une **relecture critique**.

Conseil pratique :

Voici quelques conseils qui j'espère pourront vous aider.

- Veillez à ce que votre code ne contienne pas de constantes magique et/ou de duplication qui serait facilement évitable avec l'utilisation de méthodes.
- Veillez à effectivement implémenter les différents comportements demandés.
- Réfléchissez en terme d'« attribution de responsabilités » en accord avec les principes vu au cours (segmentation logique en classes).
- Ne négligez pas la théorie du cours (vous serez interrogés dessus).
- Ne négligez pas votre rapport. Tachez d'y expliquer/justifier explicitement vos choix d'implémentation. (Exemple : pourquoi un héritage ici ? pourquoi l'implémentation comme ceci ? quel avantage en termes de structure ? …)
- Prenez le temps de bien comprendre tout l'énoncé avant de vous lancer (et lisez la FAQ).
- Vérifiez que votre code compile et run effectivement via le script bash prévu sur la machine Ubuntu présente dans le SharePoint du cours.

FAQ :

- **Puis je ajouter d'autres sections ou sous-sections dans le rapport ?**

Oui. La partie rapport de ce document donne seulement la structure minimum.

- **Puis je coder ou rendre mon rapport en anglais ?**

Oui. Pour ce qui est du code vous devez toutefois respecter les usages corrects des conventions de nommage. Codez en anglais.

- **Puis je programmer sur Windows avec Eclipse ?**

Oui vous pouvez programmer comme vous le désirez mais vous devez respecter les contraintes de ce document, notamment : votre code doit être exécutable sur un environnement linux Ubuntu via un script run.sh que vous devez fournir en même temps que vos sources (voir les contraintes de développement). Une machine préconfigurée sera disponible sur le SharePoint. **C'est cette machine qui sera utilisée pour l'évaluation de votre projet.**

- **Le rapport est-il important ?**

Oui. Le rapport est une **pièce centrale de votre projet** et c'est le premier outil de communication qui me servira à juger de la bonne réalisation du projet, pas seulement du point de vue du code mais également de la méthodologie utilisée.

- **Que voulez-vous dire par « tous les diagrammes doivent être commentés ».**

Les diagrammes doivent servir à illustrer et appuyer vos explications sur la structure de votre implémentation. Ils ne remplacent aucunement un texte explicatif revenant sur les points d'attention.

- **Je n'ai pas réussi à tout réaliser. Est-ce que ça vaut la peine de vous rendre le projet ?**

Oui veillez toutefois à être claire sur les parties non implémentées. Il est très déconseiller de dissimuler ou d'« oublier » de mentionner qu'une partie n'a pas été réalisée. Veillez toutefois à bien respecter les consignes. Par exemple, votre code doit pouvoir compiler avec le script bash demandé, la qualité du code doit être suffisante, etc…

- **Puis je réaliser le projet en groupe ?**

Non le projet doit être réalisé individuellement.

- **Que voulez-vous dire par « Votre code devra respecter les principes de designs orientés objet comme vu au cours. »**

L'orienté objet fait intervenir certains principes comme l'héritage ou les méthodes statiques. Il revient à vous de décider quand les mettre en œuvre ou non. Votre approche devra toutefois être logique et justifiée. Cela implique notamment, de faire apparaitre de l'héritage quand cela a du sens, de rendre une classe abstraite quand cela a du sens, d'utiliser intelligemment l'encapsulation etc…

- **Dois-je vraiment faire de l'orienté objet ? Mon programme peut fonctionner sans.**

Vous devez absolument mettre en œuvre l'orienté objet pour ce projet. Cela fait partie des contraintes du projet. Un non-respect de ces contraintes sera pénalisé. L'utilisation de l'orienté objet n'est pas une contrainte faible. Veillez donc à la respecter.

- **Je ne peux vraiment pas utiliser de code venant d'internet ?**

Non appart pour ce qui peut être considéré comme des briques fonctionnelles. (Par exemple le code permettant de lire/écrire un fichier, le code relatif à l'utilisation des listes ou autres structures de données). Dans tous les cas ne prenez aucun risque et contactez l'enseignant le plus tôt possible si vous avez un doute !

- **L’aspect graphique du jeu et la jouabilité sont-ils des critères importants ?**

Ces critères peuvent être pris en compte dans l’évaluation mais sont nettement moins importants que l’implémentation des fonctionnalités et le respect des contraintes. Dis grossièrement : Mieux vaut un jeu moche mais avec toutes les fonctionnalités implémentées qu’un jeu magnifique mais avec des fonctionnalités manquantes.

- **Puis je utiliser un outil comme Scene Builder pour la création d’interfaces graphique ?**

Non, vous ne pouvez utiliser que les bases fonctionnelles du SnakeFX partagées par l’enseignant sur eCampus. Tout projet dont le code fait usage d’un fichier .fxml sera sanctionné d’un zéro.

- **Puis-je modifier le projet après la remise finale ?**

Non, il ne vous est pas permis d’apporter des modifications, même mineures, à votre projet après la remise finale. Le projet présenté devant l’enseignant doit être exactement le même que celui qui a été remis sur eCampus lors de la remise finale. Tout manquement à cette consigne, consistera en **un cas de fraude**.

- **Je n’ai pas pu participer à une des activités obligatoires organisées dans le cadre du projet pour raison justifiée.**

C’est compréhensible. Il vous est toutefois demandé de vous remettre en ordre concernant les taches non réalisées dès votre retour. Attention, il revient à vous de faire les démarches nécessaires. Il convient donc de prévenir l’enseignant que l’activité n’a pas pu être réalisée et demander les étapes à suivre pour possiblement y remédier.

- **Que voulez-vous dire par : *Il est extrêmement déconseillé d’intégrer un résultat donné par ChatGPT dans votre travail sans en avoir fait une relecture critique.***

Comme précisé dans la fiche DUE, le projet vise aussi à évaluer votre capacité à faire preuve d’esprit critique au sens large. Le projet proposera d’ailleurs un contexte complet propice à la mobilisation de ces capacités. Intégrer du contenu sans relecture, en toute confiance avec le résultat retourné par un outil de type LLM, sujet aux erreurs et aux approximations d’interprétations, est à l’opposé d’une application correcte d’un esprit critique.

- **Comment m'assurer que mes choix d'interprétation sont logiques et justifiés ?**

En demandant à l'enseignant, qui joue le rôle de client/chef de projet. Cela est d'ailleurs représentatif d'un processus de développement réaliste.

- **Puis je utiliser la notation template ou encore les java.util.stream?**

Non, la notation template n’est pas autorisée à l’exception de son usage dans la structure de données vues au cours de Java Q3. Les java.util.stream ne sont également pas autorisés. Cela rentre dans le cadre de la consigné énoncée plus haut : il ne vous est pas permis d’inclure dans votre projet des éléments autre que les briques fonctionnelles déjà fournies par le SnakeFX et les codes ressources disponible sur eCampus.