Projet Java III - 2024-2025

HELBArmy

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, deux armées s'affrontent pour dominer un territoire stratégique. Chaque camp envoie ses troupes — collecteurs, cavaliers, piquiers et déserteurs — pour prendre l'avantage. Gagner la bataille dépend de la gestion des ressources et du contrôle d'objets clés comme les drapeaux et les pierres magiques. La bataille est lancée : quelle armée l'emportera ?

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. 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 :

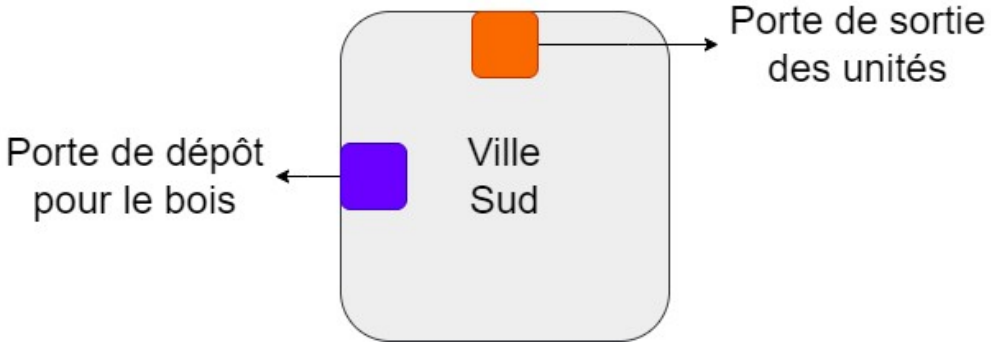
La carte contient un certain nombre d'arbres placés aléatoirement sur la carte. Les arbres permettent de collecter du bois. Un arbre dispose d'une certaine quantité de bois pouvant varier entre 0 et 100. Visuellement, les arbres contenant le plus de bois apparaissent plus verts sur la carte. Quand l'intégralité du bois de l'arbre a été collecté, celui-ci disparaît de la carte pendant 30 secondes avant de réapparaître à la même position avec la quantité initiale de bois dont il disposait. Quand un arbre est présent sur la carte, celui-ci est un obstacle qui ne peut pas être traversé. Toutefois, la case occupée par l'arbre peut être traversée durant sa période de disparition.

Les villes :

Les villes sont des éléments ayant une position sur la carte. Elles génèrent des unités qui sortent par une porte unique orientée vers le centre de la carte. Elles disposent également d'un point de dépôt pour la collecte du bois. Deux villes adversaires sont présentes sur la même carte, situées respectivement au milieu du bord supérieur et au milieu du bord inférieur.

Les villes sont les seuls éléments du jeu à occuper plus d'une case dans l'espace de jeu, chacune occupant un espace de 5x5 cases.

Le schéma suivant illustre la ville du sud, où l'on peut voir deux portes : la porte de sortie des unités et la porte de dépôt pour la collecte du bois par les collecteurs.



Les unités :

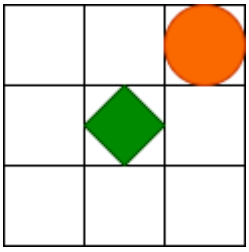
Les unités sont générées par les villes. Pour créer une unité, il est nécessaire de dépenser une certaine quantité de ressources, laquelle dépend du type d'unité à produire. De plus, un temps d'attente est requis, également variable en fonction de l'unité à produire. Il existe quatre types d'unités générées par les villes : les collecteurs, les déserteurs, les cavaliers et les piquiers. Chaque unité occupe une case sur l’espace de jeu.

Les collecteurs :

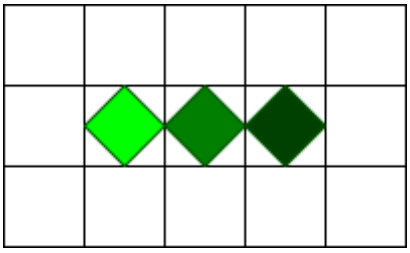
Les collecteurs apparaissent en orange sur la carte. Ils permettent de collecter des ressources. Une fois générés, ils se dirigent vers l'arbre le plus proche pour y récolter du bois. Les collecteurs peuvent accumuler jusqu'à 25 unités de bois. Lorsqu'ils sont pleins, ils retournent à la ville pour déposer leurs ressources avant de repartir. Si, à l'issue d'une collecte, ils ne sont pas pleins, ils se dirigent vers l'arbre suivant le plus proche.

La collecte s'effectue de la manière suivante : le collecteur doit se placer sur une des cases adjacentes à l'arbre (8 positions possibles), et une unité de bois est collectée par seconde. Au fur et à mesure de la collecte, la quantité de bois de l'arbre diminue, et celui-ci apparaît de moins en moins vert sur la carte.

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.



Le schéma suivant illustre trois arbres présentant différents niveaux de bois. L’arbre tout à gauche dispose de la quantité maximale de bois possible (100) et affiche donc la couleur la plus verte (0, 255, 0). L’arbre du milieu possède deux fois moins de bois, ce qui lui confère la couleur (0, 127, 0). L’arbre tout à droite possède 25 unités de bois, ce qui lui donne la couleur (0, 63, 0).



Les déserteurs :

Les déserteurs apparaissent en mauve sur la carte. Ils chassent les collecteurs de l'armée adverse et fuient toutes les autres unités ennemies. De plus, ils sont avantagés au combat face aux piquiers.

Les cavaliers :

Les cavaliers apparaissent en bleu sur la carte. Ils chassent les déserteurs de l'armée adverse et se déplacent en groupes alliés, en maintenant une distance de sécurité entre eux. Ils sont avantagés au combat face aux déserteurs.

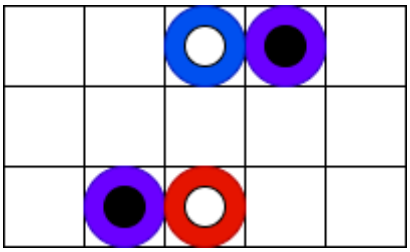
Les piquiers :

Les piquiers apparaissent en rouge sur la carte. Ils tiennent leur position et attaquent à vue. Ils sont avantagés au combat face aux cavaliers.

Système de combat :

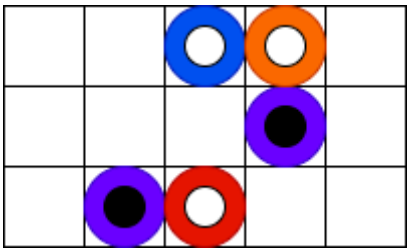
Lorsqu'une unité d'armée rivale se trouve sur une position adjacente à une autre, un combat s'engage entre ces unités. Chaque unité inflige alors, l'une après l'autre, des coups à l'unité adverse, causant des dégâts calculés en fonction de ses points d'attaque et des bonus éventuels, jusqu'à ce que les points de vie de l'une des unités tombent à zéro. Cette unité disparaît alors de la carte, permettant à l'autre unité de reprendre son déplacement.

Le schéma suivant illustre deux affrontements, un cavalier contre un déserteur ainsi qu'un déserteur contre un piquier. Les couleurs noir et blanches inclues à l'affichage de l'unité permettent de différencier les deux équipes.



Plusieurs unités peuvent être impliquées dans un combat. Une unité peut avoir autant d'adversaires que de cases adjacentes. Dans ce cas, l'unité porte un coup à un adversaire choisi au hasard parmi ceux à sa portée. Le combat se poursuit jusqu'à ce qu'il n'y ait plus d'adversaires.

Dans le schéma suivant, le déserteur sur la ligne du milieu peut attaquer aléatoirement le cavalier, le collecteur, ou le piquier adverse.



Le tableau suivant résume les points de vie, d'attaque, ainsi que les bonus :

Collecteur	PV 50	Attaque 5	
Déserteur	PV 125	Attaque 10	x1,5 contre les piquiers
Cavalier	PV 200	Attaque 10	x2 contre les déserteurs
Piquier	PV 175	Attaque 15	x3 contre les cavaliers

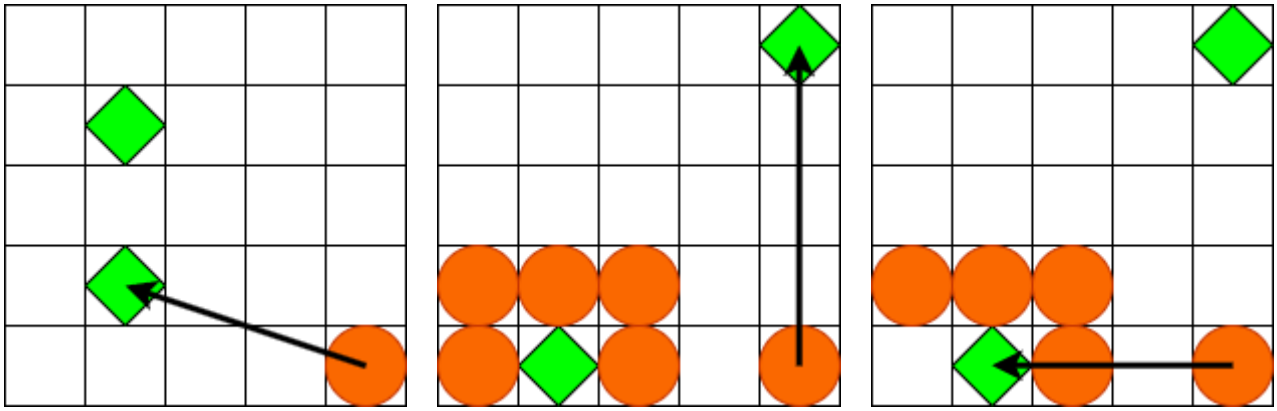
Chaque coup porté par un déserteur à un collecteur ou à un cavalier inflige donc 10 points de dégâts. Et chaque coup porté par un déserteur à un piquier inflige donc 15 points de dégâts (=10*1,5).

Système de déplacement :

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

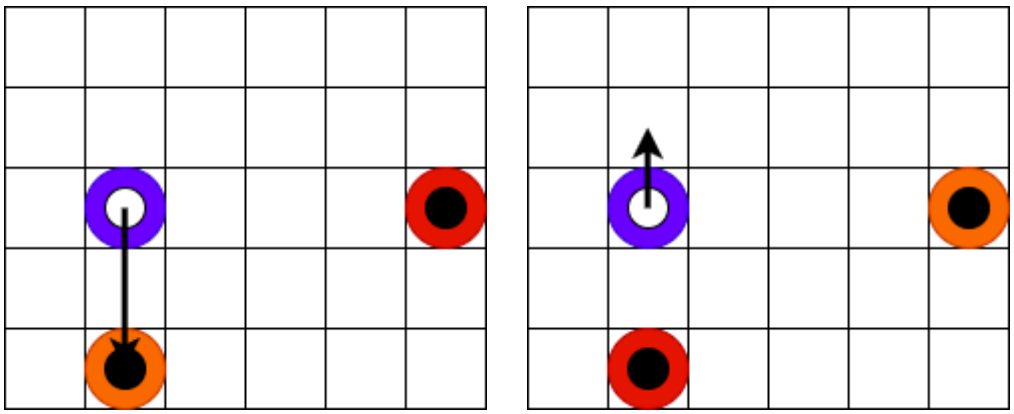
Collecteurs : Ils se déplacent vers l'arbre collectable le plus proche, c'est-à-dire un arbre dont une position de collecte est disponible.

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.

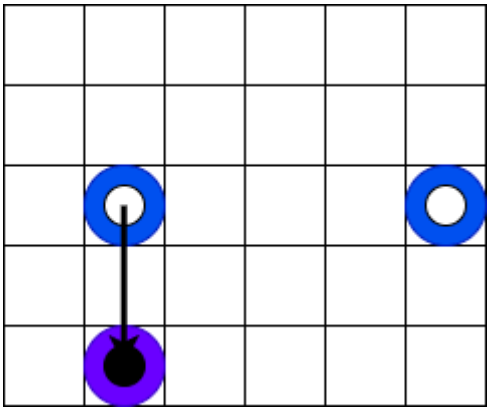


Déserteurs : Ils chassent les collecteurs ennemis et fuient les autres unités adverses. Ils comparent la distance entre le collecteur attaquable le plus proche et l'unité ennemie non collecteur la plus proche. Si le collecteur est plus proche, ils se dirigent vers lui. Si une unité ennemie non collecteur est plus proche, ils se dirigent dans la direction opposée.

Le schéma suivant illustre le déplacement du déserteur dans deux situations, chacune menant à un choix de direction différent pour le déserteur.

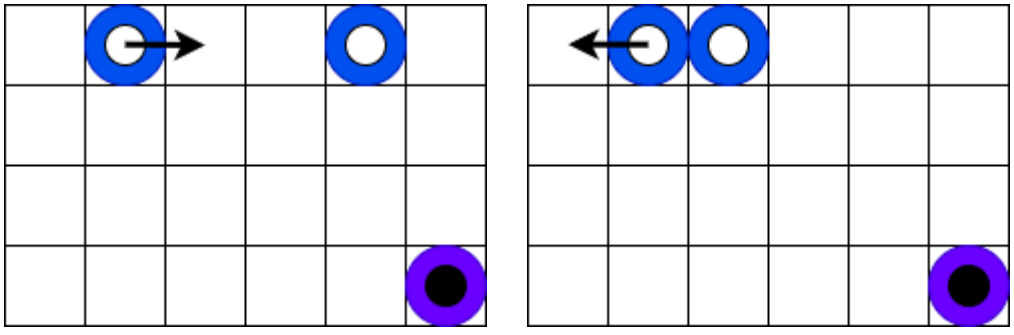


Cavaliers : Ils chassent les déserteurs et se déplacent en groupes alliés. Les cavaliers comparent la distance entre le déserteur le plus proche et l'allié cavalier. le plus proche. Si le déserteur est plus proche, il se dirigent vers lui comme le montre le schéma suivant où la situation est similaire au déserteur.



Si l'allié cavalier est plus proche, deux cas sont possibles :

Si la distance avec cet allié est inférieure à la distance de sécurité établie entre les cavaliers, le cavalier se déplace dans la direction opposée pour rétablir cette distance. Si la distance est égale ou supérieure à cette distance de sécurité, il se dirige vers cet allié. Les schémas suivants illustrent les deux situations possibles pour une distance de sécurité de « 1 ».

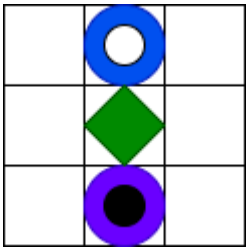


Au fur et à mesure de la bataille, la distance de sécurité entre tous les cavaliers d'une même équipe peut évoluer. Lorsque n'importe quel cavalier engage un combat, cette distance passe à 1 pour tous les cavaliers de l'équipe. Si, après que tous les cavaliers d'une équipe ont agi, aucun n'a attaqué, cette distance de sécurité est incrémentée de 1.

Piquiers : Les piquiers tiennent leur position et attaquent à vue. À leur sortie de la ville, ils reçoivent une position aléatoire sur la carte qui leur est assignée. Tous les piquiers d'une même équipe partagent un champ de vision collectif qui est égal au nombre total de piquiers présents sur la carte. Lorsqu'ils se déplacent, ils vérifient si un ennemi est présent à une distance inférieure à ce champ de vision. Si un ennemi est repéré, ils se dirigent vers lui. Sinon, ils se rendent à leur position assignée.

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, le cavalier 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. Attention, cela ne doit toutefois pas impacter les temps fonctionnels spécifiés dans l'énonce : exemple : même avec un rafraichissement du jeu accéléré x2, les arbres disparus mettent toujours 30 secondes à réapparaitre.

Génération des unités :

Les unités sont générées de manière aléatoire par les villes. Le choix de l'unité à générer se fait de la façon suivante : on sélectionne aléatoirement parmi les unités pour lesquelles les ressources nécessaires sont disponibles. Voici le coût en ressources et le temps de génération par unité :

- Collecteur : 0 bois - 5 secondes
- Déserteur : 50 bois - 10 secondes
- Cavalier : 100 bois - 15 secondes
- Piquier : 75 bois - 5 secondes

Initialement, les villes ne possèdent aucune ressource.

Collectables :

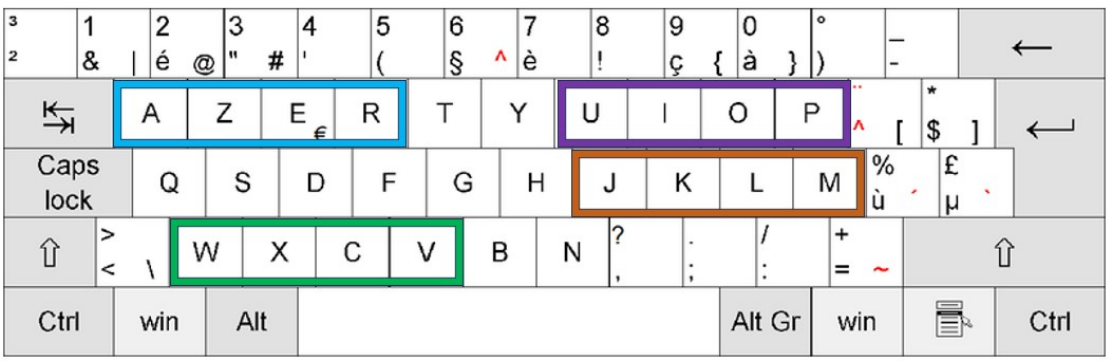
Le Drapeau : Toutes les deux minutes, un drapeau apparaît à une position aléatoire sur la carte. Lorsque le drapeau est présent, toutes les unités modifient leur comportement pour se diriger vers lui afin de le collecter. Une fois le drapeau collecté par une unité, il disparaît et toutes les unités reprennent leur comportement normal. Au moment de la collecte, toutes les unités de l'équipe ayant récupéré le drapeau bénéficient de 50 % de points de vie supplémentaires. Ce bonus ne s'applique qu'aux unités déjà présentes sur la carte au moment de la collecte. Un seul drapeau peut être présent sur la carte à la fois.

La Pierre Philosophale (la quoi ?) : Initialement, deux pierres philosophales sont placées aléatoirement sur la carte. Lorsqu'une unité entre en contact avec l'une d'elles, elle a une chance sur deux de mourir ou de devenir invincible, avec des points de vie illimités.

Commandes interactives – Cheat codes :

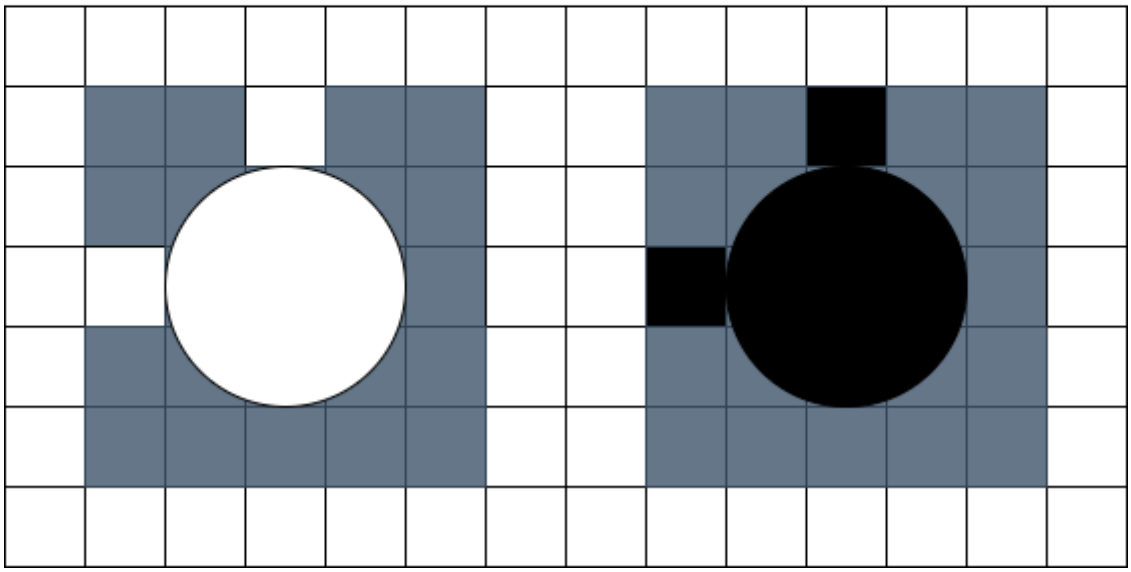
Afin de permettre des interactions avec la simulation, il existe un système de commandes permettant d'influencer la partie. Quand l'utilisateur appuie sur une touche, cela déclenche un évènement précis :

a	Génère instantanément un collecteur dans la ville nord.
z	Génère instantanément un déserteur dans la ville nord.
e	Génère instantanément un cavalier dans la ville nord.
r	Génère instantanément un piquier dans la ville nord.
w	Génère instantanément un collecteur dans la ville sud.
x	Génère instantanément un déserteur dans la ville sud.
c	Génère instantanément un cavalier dans la ville sud.
v	Génère instantanément un piquier dans la ville sud.
j	Stopper/activer le déplacement de tous les collecteurs.
k	Stopper/activer le déplacement de tous les déserteurs.
l	Stopper/activer le déplacement de tous les cavaliers.
m	Stopper/activer le déplacement de tous les piquiers.
u	Met instantanément à 0 les PV de toutes les unités présentes sur la carte.
i	S'il n'y en a pas déjà un, un drapeau apparait sur la carte.
o	Reset de la simulation
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 arbres, unités et des collectables (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 équipe nord et sud. Finalement, les villes du nord et du sud avec leur portes respectives.

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. Au minimum un commentaire est attendu pour toutes les méthodes contenant plus de cinq lignes de code. Ce commentaire doit expliquer ce que représentent les paramètres, ce que fait la méthode et ce qui est retourné. **Un code ne respectant pas cette règle 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 demandé 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 communiquée ultérieurement.

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 29 décembre 2024 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'au vendredi 31 janvier inclus). 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.

- **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.