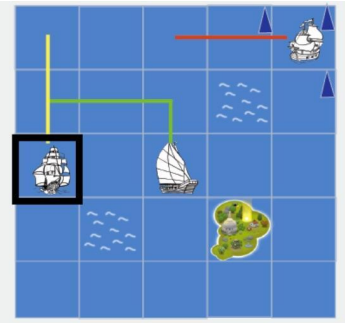
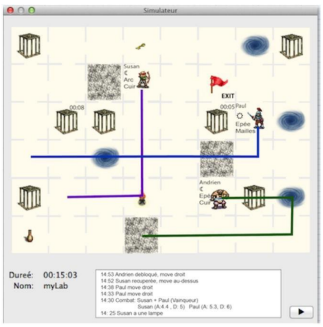
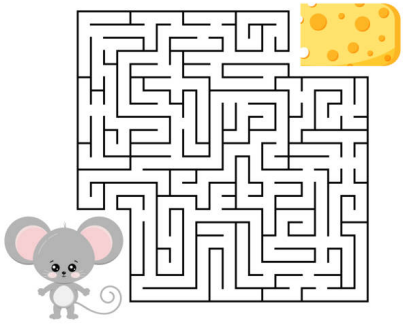
Projet IHM - ET3 : Jeu de Navigation

Contexte : Votre objectif dans ce projet est de créer un jeu de navigation interactif où des personnages contrôlés par l'utilisateur (et des personnages automatisés) se déplacent. La raison pour laquelle ils se déplacent dépend de vous : pour sortir d'un labyrinthe le plus rapidement possible, pour trouver les clés et accéder au trésor dans un château, pour gagner une compétition de bateau avec des personnages automatisés … Ce qui existe à l'intérieur du jeu dépend également de vous : s'agit-il de points d'accès ouverts et verrouillés, de pièges qui retardent les personnages, de trésors qui leur donnent des pouvoirs supplémentaires, …

Vous devez également décider de la manière dont l'utilisateur interagit avec son personnage : le déplace-t-il pas-à-pas en utilisant le clavier ou la souris, trace-t-il un chemin à suivre, etc.

L'interaction entre les personnages est également ouverte : si un personnage se trouve dans une pièce d'un labyrinthe, est-ce qu'il la bloque pour les autres ? Lorsque deux personnages se rencontrent dans un château, se battent-ils ou s'échangent-ils des informations sur les salles, etc. ? Se déplacent-ils à tour de rôle ou se déplacent-ils tous en même temps ? Á vous de définir votre jeu.

(Vous pouvez vous inspirer des exemples à la fin de l'annonce).

Déroulement du projet: L'application sera réalisée en Java, tandis que l'interface graphique utilisera **JavaFx**. L'environnement de développement conseillé est soit Eclipse soit IntelliJ, mais vous pouvez utiliser celui avec lequel vous êtes le plus familier. L'utilisation d'un outil de gestion de versions (**Git** de préférence) est fortement conseillé pour faciliter le partage entre binômes. Vous pouvez tester votre partie Back-end / algorithmique à l'aide des tests JUnit (http://junit.org/).

Le travail est à réaliser en binôme et il est donc important de s'assurer que (i) vous divisez vos tâches de façon équitable; et (ii) les différentes parties développées par les deux collègues communiquent entre elles (même si elles sont codées dans différentes versions de Java ou JavaFX. Il faut donc décider à votre plateforme d'intégration càd la machine / version ou vous mettez ensemble votre code et vous testez que cela (un de vous aura le rôle de l'intégrateur).

Cahier des charges : Bon, il s'agit quand même d'un projet d'ingénierie des interfaces graphiques, donc il y a des attentes.

i) Composants visuels

Le jeux de navigation aura :

1. Au minimum une interaction drag&drop (Priorité : 1)
2. Utilisation des images pour les personnages (Priorité : 1), mais pas seulement (plateau, ressources, pièges, etc.)
3. Une manière de créer de nouveaux personnages (contrôlés par l'utilisateur ou automatisés)
    a. création aléatoire (Priorité : 1)
    b. fenêtre pour la création graphique faite par l'utilisateur (Priorité : 2)
4. Un moyen de créer un plateau du jeu et ses propriétés
    a. création aléatoire (Priorité : 1)
    b. fenêtre pour la création graphique faite par l'utilisateur (Priorité : 2)
5. Une fenêtre pour jouer au jeu de navigation (Priorité : 1)
6. Un moyen pour l'utilisateur d'interagir/contrôler son personnage, par exemple, le déplacer, ajuster sa direction, … (Priorité : 1). Pensez à des interactions avancées (comme dessiner des chemins, faire glisser, traverser) et pas seulement à cliquer (Priorité : 2).
7. Fournissez des retours clairs à l'utilisateur sur l'état de leur personnage (en mouvement, en combat, blessé, etc.), ainsi que sur l'état du plateau du jeu (Priorité : 1)

ii) Composants applicatifs

Le jeu de navigation aura :

8. Une représentation interne du plateau du jeu (Priorité : 1). La représentation doit avoir des conditions de jeu qui ne consistent pas seulement à traverser le plateau du jeu pour gagner. Par exemple, collecter des éléments dans un ordre spécifique, remporter des combats, ajouter des événements aléatoires, brouillard de guerre ("fog of war")
9. Créer un plateau du jeu aléatoire qui peut être résolu (Priorité : 1). On note ici qu'un jeu type "labyrinthe" peut devenir très intéressant (vu complexe) au niveau algorithmique à générer aléatoirement, de façon qu'on peut le résoudre. Si vous visez un jeu qui a un algorithme complexe, ce serait bien d'y discuter avec votre enseignant. On vous encourage à commencer avec une version simplifiée avec un remplaçant "fait main" au début ou un algo très simplifié pour tester le reste des fonctionnalités.
10. Traiter les entrées de l'utilisateur pour déplacer le personnage (Priorité : 1)
11. Déterminer les conditions de victoire ou de défaite du jeu (Priorité : 1)
12. Sauvegarder la disposition du plateau du jeu (Priorité : 2)
13. Sauvegarder le jeu (plateau, positions, …) pour continuer le jeu plus tard (Priorité : 2)
14. Compter le temps et le nombre d'actions (Priorité : 3 sauf si important pour votre jeu)
15. Sauvegarder les personnages pour les réutiliser (Priorité : 3)

Quelques conseils :

- Réfléchissez à la meilleure interaction pour ce que vous souhaitez faire (glisser-déposer, widgets, palettes, menus). Visez une interface cohérente qui fonctionne bien dans son ensemble.
- Vous êtes encouragé à être créatif, mais assurez-vous que ce que vous décidez de faire correspond à votre niveau de compétence et à vos capacités. Si certains aspects vous semblent trop difficiles, essayez de simplifier les choses.
- Définissez les limites de votre projet avant de commencer. Développer plus de fonctionnalités ne vous donnera pas nécessairement une meilleure note (ou un meilleur produit). Ce projet peut aller de très simple à très complexe. Et il sera toujours possible de rajouter des fonctionnalités plus tard si vous en avez vraiment le temps. Discutez avec votre partenaire avant de décider exactement de ce que vous voulez faire. Si vous avez des doutes sur la difficulté de votre choix, demandez l'opinion de votre enseignant.
- Nous n'attendons pas de mécanismes de jeu complexes ou de personnages non-joueurs avec une IA sophistiquée.
- Votre implémentation doit être utilisable. Vous devez empêcher les utilisateurs d'atteindre des états inattendus dans le système et fournir des mécanismes de rétroaction pour éviter les erreurs de l'utilisateur.

Exemples : Voici trois exemples simples des jeux de navigation pour vous inspirer :

i) Une course en bateau (Image 1) : un plateau grillagé, avec des bateaux qui se promènent (un seul bateau est contrôlé par l'utilisateur). La météo dans différentes zones du plateau maritime change de manière imprévisible. Elle pousse les bateaux qui ont une vitesse et une direction initiales dans des directions différentes, leur causant des dommages qui les retardent. En fonction de leur taille et de leur poids, les bateaux sont affectés différemment par les conditions météorologiques. L'utilisateur effectue des gestes de balayage pour ajuster et corriger la direction de déplacement de son bateau lorsque les conditions météorologiques le dévient de sa trajectoire. L'utilisateur et les bateaux non-joueurs essaient d'atteindre un île en sécurité.

Le défi du guerrier (Image 2) : dans un château (type labyrinthe) des personnages de races différentes (orc, humaine, elf) et équipés différemment (armures, armes) se déplacent dans un "brouillard de guerre", c'est-à-dire qu'ils ne peuvent pas voir les zones du château qu'ils n'ont pas encore visitées. L'utilisateur clique vers la zone du labyrinthe où il souhaite que son personnage se déplace. Lorsque les personnages rencontrent d'autres personnages de leur race, ils partagent des informations sur ce qu'ils connaissent du château et de ses pièges, tandis que s'ils rencontrent des personnages d'autres races, ils se battent, gagnent ou perdent des armes et des trésors. Si l'utilisateur agit vite (ex avec du balayage) il peut echapper une bataille. L'objectif est de trouver autant de trésors que possible dans le labyrinthe avant la fin d'un temps prédéfini.

Labyrinthe programmé (Image 3) : une petite souris veut échapper d'un labyrinthe ou il y a des cul-de-sac ou des parties dangereuses (une chat qui se promène). L'utilisateur trace dans un

labyrinthe le trajet à suivre en laissant des morceaux de fromage et/ou en bloquant des trajets avec des pierres. La souris se déplace toujours droit, sauf s'il y a morceau de fromage autour de lui, là il tourne vers le fromage le plus près. L'objectif est d'aider le souris d'échapper avec le nombre minimum des ressources (fromage, pierres) utilisées.

Évaluation du projet:

L'évaluation du projet Java - Graphique - IHM portera sur plusieurs livrables : (i) le prototype papier ; (ii) le diagram UML ; (iii) le code du projet (l'essentiel du projet) ; un (iv) mini-rapport avec des images / retours utilisateurs / reflection ; et (v) une vidéo qui démontre les fonctionnalités de votre jeux.

Á rendre via eCampus au plus tard le dimanche **25 juin 2023 @23:59**.

Répartition des notes sur 20

Prototype papier / Storyboard	Les interactions individuelles sont affichées. Tous les écrans de l'interfaces sont présents. **à rendre au 2e session** au plus tard	1
UML	Classes bien organisées. Utilisation de MVC. **à faire et rendre au 2e session** au plus tard	1
Projet (Code + Fonctionnalités)	Livrable: Projet compilé et tourne sans (ou avec peu de) bugs. Sinon, une pénalité sera appliquée. Qualité du Code : - Organisation du code en MVC - Code bien structuré - Documentation (commentaires, descriptions classes, entrées/sorties, référence à des ressources utilisées). Suivi du cahier des charges : - Répond aux exigences au niveau visuel (vues demandés, drag&drop, utilisation images, etc) et au niveau applicatif - C'est utilisable et cohérent. Ambition / Complexité	 2 11 2
Rapport	Respecte la limite de pages (4 pages max, police 11pt) Explique (brièvement) la division de travail / responsabilités de chaque membre de l'équipe Comprend une paragraphe qui explique le jeu (comme les exemples)	2

	Comprends une image par fenêtre / vue. Comprend le cahier des charges (liste) et explique : - ce qui a été réalisé et ce qui ne l'a pas été Comprend les retours des utilisateurs, et ce qui a été corrigé ensuite Reflection sur le déroulement du projet (ce qui est bien marché et ce que vous feriez autrement) Comprend une liste des ressources utilisés (tutorials, libraries, algorithmes, …) avec une brief description on comment vous l'avez intégrez dans votre projet	
Vidéo	Soumis et montre les fonctionnalités demandés	1