Sujet : réaliser le début du jeu *Loop Hero*

Le but de ce projet est de réaliser une version simplifiée du jeu *Loop Hero*. Il s'agit d'un jeu pour un seul joueur où l'on est coincé dans une boucle que l'on aménage soi-même. C'est un jeu de type *rogue-like* avec un peu de *deck building*. Il existe sur de multiples plates-formes et avoir joué au jeu permet sans doute de mieux comprendre ses mécanismes, mais ce n'est pas indispensable. Si vous n'en n'avez jamais entendu parler, vous pouvez facilement trouver des vidéos de parties sur YouTube pour vous faire une idée du mécanisme de jeu.

Le jeu

Ce sujet fournit une présentation succincte du jeu, vous trouverez des informations beaucoup plus complètes sur le WIKI du jeu : `https://loophero.fandom.com/wiki/Loop_Hero_Wiki`. Attention, si les informations données dans le wiki sont en contradiction avec le sujet, alors c'est le sujet qui prévaut.

Le jeu commence dans une boucle vide avec quelques monstres de type *Slime*. Le personnage (un guerrier, que l'on appelle le héros) avance automatiquement dans la boucle pendant que le temps s'écoule. Lorsqu'il rencontre un monstre, un combat automatique a lieu. Si le héros meure, la partie s'arrête immédiatement, sinon, il peut éventuellement récupérer des cartes ou des objets lâchés par le monstre, ainsi que des matériaux. Les objets peuvent être équipés pour modifier les caractéristiques du héros et les cartes peuvent être jouées pour modifier la boucle et le paysage dans lequel il évolue. Le but est de gagner le plus de matériaux possibles (de maximiser leur valeur) avant de mourir ou de finir le 9e tour de boucle (c'est très simplifié par rapport au jeu réel).

Le héros, dans sa boucle La zone de jeu est constituée de la boucle et de cases de paysage que l'on peut aménager. Le héros se déplace uniquement sur la boucle.

- La zone de jeu est une grille de 21 x 12 cases carrées et la boucle compte exactement 34 cases. Sa forme est aléatoire. L'une des cases est le camps de base.
- Chaque fois que le héros passe par le camp de base, il est soigné : ses points de vie (HP) augmentent de 20% (dans la limite de son HP maximum).
- Les tuiles sont des éléments que l'on rajoute dans la zone de jeu (on les obtient grâce à des cartes, voir ci-dessous). Elles ont des règles de placement et des effets qui sont décrits en détails dans le Wiki : `https://loophero.fandom.com/wiki/Tiles`. En gros :
  - Les tuiles de paysage se placent en dehors de la boucle et apportent des bonus permanents. Elles peuvent provoquer l'apparition de monstres sur la boucle.
  - Les tuiles de bord de route et de route se placent juste à côté ou sur la boucle et génèrent principalement des monstres.
  - Quelques tuiles spéciales ont un placement spécifique décrit dans le Wiki.
  - Parmi les bonus apportés par les tuiles, il y a des points d'HP par jour (voir ci-dessous).

Les autres éléments du jeu ne font pas partie de la zone de jeu à proprement parler, mais on doit pouvoir voir les cartes en main, l'inventaire et l'équipement du héros, ainsi que le moment de la journée.

Le temps du jeu Le héros se déplace sur la boucle en continu, en commençant par le camp de base, à raison d'1.5 seconde par case.

- Certaines tuiles ont des effet qui se renouvellent chaque nouveau jour. Un jour dure 24 secondes (donc, en particulier, une boucle dure plus d'une journée).
- Dans un combat, un coup est porté chaque seconde (il y a normalement une caractéristique *Attack Speed* dans le jeu, mais on n'en tient pas compte pour ce projet). Le temps continue de s'écouler pendant le combat.

- Pour poser des tuiles et changer son équipement, il est nécessaire de "stopper" le temps. Cela s'appelle le mode "planification". Il suffit d'appuyer sur la touche espace pour passer d'un mode à l'autre.

Les *drops* Il s'agit des différents éléments lâchés par les monstres ainsi que par le placement de certaines tuiles.

- Les **matériaux** sont décrits ici avec leur valeur (*gain*) : `https://loophero.fandom.com/wiki/Materials`. Il proviennent principalement des ennemis et des tuiles ou de la destruction de certains éléments du jeu. C'est la somme de leurs valeurs qui donne le score.
- Les **cartes** permettent de placer des tuiles (sauf Oblivion qui permet de retirer des éléments). Seuls les ennemis en lâchent. Elles sont directement placées dans la main du joueur qui peut les jouer à tout moment pour placer des tuiles. La main contient au maximum 13 cartes. Si plus de cartes doivent être ajoutées, les moins récemment ajoutées sont détruites. Elles sont décrites ici : `https://loophero.fandom.com/wiki/Cards`. Le joueur a un paquet fini de cartes (le nombre de chaque type est indiqué dans les pages du Wiki) qui constitue la réserve dans laquelle elles sont piochées au hasard. Lorsqu'elles sont détruites, elles sont remises dans cette réserve.
- De l'**équipement** peut également être lâché par les ennemis, avec un probabilité propre à chaque ennemi (voir la section détaillée pour plus de précisions) :
  - il y en a de 4 types : arme, bouclier, armure et anneau ;
  - chaque item a un niveau qui dépend du nombre de tours effectués dans la boucle ;
  - ils peuvent avoir 4 niveaux de rareté : commun (gris), rare (bleu), très rare (jaune) ou exceptionnel (orange) ;
  - chaque item apporte des caractéristiques particulières au héros ;
  - le héros peut équiper un de chacun des types d'objets et en conserver 12 autres dans son inventaire. S'il n'y a plus de place dans l'inventaire, les items les moins récemment ajoutés sont détruits. Lorsque que l'on équipe un nouvel item, l'item précédemment équipé est détruit.

Les combats Les tuiles génèrent automatiquement et aléatoirement des ennemis. Chaque rencontre du héros avec un ennemi déclenche un combat automatique. Le déroulement du combat dépend uniquement des caractéristiques actuelles du héros et de l'ennemi (pour simplifier, on ne peut avoir qu'un seul ennemi par tuile). Le combat se déroule dans le temps jusqu'à ce qu'un des deux adversaires n'ait plus de points de vie (HP). Voici des précisions sur l'influence des différentes caractéristiques sur le combat :

- *Maximum HP* : indique combien de points de vie peut avoir le héros au maximum. Sa valeur est 250 au début et elle évolue au fil du temps grâce aux tuiles et à l'équipement.
- *Damage* : donne l'intervalle du nombre de points de dégâts infligés à chaque coup (de base, pour le héros, c'est $[4 : 6]$).
- *Defense* : le nombre de points de défense indique le seuil à partir duquel les dégâts comptent. Par exemple, si on fait 10 points de dégâts sur un adversaire qui a 3 points de défense, ça diminue seulement son HP de 7.
- *Counter* : le pourcentage de chance de contre-attaque (faire des dégâts au lieu d'en subir).
- *Regen per sec* : le nombre de points de vie gagnés par seconde (pendant un combat et lors du déplacement dans la boucle).
- *Evade* : le pourcentage de chance d'éviter un coup.
- *Vampirism* : le pourcentage de points de dégâts subis qui sont transformés en HP.

Il y a d'autres caractéristiques dans le jeu réel, mais on n'en tient pas compte ici. Les ennemis peuvent aussi avoir ces caractéristiques. Leur HP et leur Damage sont calculés avec la même formule :

$$base \times lvl \times 0.95 \times (1 + (lvl - 1) \times 0.02),$$

où lvl est le numéro du tour de boucle dans lequel on se trouve et *base* (*HP* pour la vie et *Strength* pour les dégâts) est la valeur indiquée dans les pages du Wiki : `https://loophero.fandom.com/wiki/Enemies`.

Les statistiques des équipements

- Les probabilités associées à la rareté sont 35% pour les gris, 30% pour les bleus, 20% pour les jaune et 15% pour les oranges.
- La probabilité pour un ennemi de lâcher un item est indiquée sur la page du Wiki (sinon, il lâche une carte).
- Le niveau d'un item (lvl) est le numéro de tour de boucle dans lequel on est.
- les items ont les caractéristiques suivantes lorsqu'ils sont gris (communs) :
  - Pour les armes : $Damage = [4 \times lvl : 6 \times lvl]$.
  - Pour les boucliers : $Defense = 4 \times lvl$.
  - Pour les armures : $MaximumHP = [80 \times lvl : 100 \times lvl]$.
  - Pour les anneaux : au choix parmi les 5 caractéristiques spéciales suivantes :
    - $Counter = 8 + (lvl - 1) \times 4$
    - $Vampirism = 8 + (lvl - 1) \times 1.5$
    - $Regen = lvl \times 0.6$
    - $Evade = 8 + (lvl - 1) \times 2$
    - $Defense = lvl \times 1.5$

  Un intervalle signifie : une valeur au hasard dans cet intervalle.
- La rareté influe sur les anneaux : chaque niveau de rareté ajoute une caractéristique spéciale au hasard en plus, dont la valeur est divisée par 2 par rapport à celle du niveau de l'anneau.
- La rareté influe aussi sur les autres items :
  - bleu : 90% de sa valeur normale mais une caractéristique spéciale en plus, dont la valeur est divisée par 3 par rapport à son niveau.
  - jaune : $80 - 100\%$ de sa valeur normale et deux caractéristiques spéciales en plus dont la valeur est divisée par 2 par rapport à son niveau.
  - orange (seulement à partir de la 3e boucle) : $80 - 100\%$ de sa valeur normale et deux caractéristiques spéciales dont la valeur est divisée par 2 par rapport à son niveau, plus une autre dont la valeur est 2 niveaux en dessous de son niveau normal.

Remarque : tous les calculs sont arrondis à l'entier inférieur.

Le programme à réaliser, en plusieurs étapes

Phase 1 : Boucle simple Durée : environ 1 mois.

À la fin de la phase 1, on doit avoir les bases du jeu avec les fonctionnalités suivantes correctement implémentées :

1. La création de la zone de jeu, avec une boucle prédéfinie (de votre choix) qui contient uniquement un camp de base et quelques *Slimes*.
2. Le héros se déplace en continu et l'interface affiche le temps qui s'écoule ainsi que le numéro de la boucle en cours.
3. Les *Slimes* apparaissent aléatoirement suivant la probabilité donnée dans le Wiki.
4. La possibilité de passer en mode "planification" pour poser des tuiles.
5. Les cartes *Grove*, *Rock* et *Meadow* (avec une de chaque en main au démarrage du jeu).
6. La possibilité de jouer des cartes pour poser des tuiles et bénéficier des effets de ces tuiles (y compris l'apparition des monstres).
7. L'affichage de la liste des ressources gagnées.
8. Pour l'instant les combats ne sont pas implémentés. On considère que chaque combat fait perdre 6 points de vie et dure 2 secondes. Si au moment d'un combat, l'HP tombe à 0, c'est perdu.
9. Les monstres lâchent uniquement des cartes (avec la probabilité indiquée dans le Wiki) et des ressources.
10. On doit pouvoir accélérer ou ralentir le temps du jeu pour pouvoir tester plus efficacement.

Phase 2 : Combats et équipement Durée : environ 1 mois.

Après avoir fini et validé la première phase, vous devez ajouter les fonctionnalités suivantes :

1. Les combats doivent se dérouler en utilisant les caractéristiques (sauf les spéciales) propres à chaque adversaire :
   - Tous les éléments pris en compte pour le combat doivent être affichés sous forme de texte : chaque coup porté, chaque HP perdu, chaque effet utilisé, etc...
   - L'affichage de ces actions peut–être fait au fil du temps du combat, mais le plus simple est de tout afficher et de décompter le temps du combat lorsque l'on stoppe l'affichage.
   - Les statistiques du héros doivent rester visibles pendant tout le temps du jeu et celles de l'ennemi pendant la durée du combat.
2. À la fin du combat, l'ennemi peut désormais lâcher des items d'équipement gris sauf les anneaux, en suivant les probabilités indiquées dans le Wiki.
3. L'inventaire est affiché en permanence. Pendant le mode planification, on peut équiper le héros avec ces items.

Phase 3 : Finitions Durée : environ 1 mois.

Lorsque la deuxième phase est complètement finie, vous devez ajouter les fonctionnalités suivantes :

1. Ajouter les cartes *Cemetery*, *Spider Cocoon*, *Vampire Mansion*, *Battlefield*, *Oblivion*, *Beacon*, *Village*, *Wheat Fields* et *Ruins*, avec toutes les fonctionnalités (tuiles, effets, ressources) associées (voir Wiki).
2. Faire en sorte que les ennemis puissent également lâcher des anneaux, avec leur caractéristiques spéciales, et les prendre en compte dans les combats.
3. Faire en sorte d'avoir des équipements des différents niveaux de rareté, avec leur formules de calcul propres.
4. Permettre la sauvegarde des données de chaque partie (nombre de boucles, nombre d'ennemis vaincus, score, ...).
5. Permettre de lire la forme de la boucle dans un fichier.
6. Avoir la possibilité de sauvegarder la partie et relancer.
7. (bonus) Implémenter les capacités spéciales (*Abilities*) des ennemis.
8. (bonus) Ne pas limiter le nombre d'ennemis à un par tuile.
9. (bonus) Générer la forme de la boucle aléatoirement.

L'interface graphique

Vous aurez besoin de réaliser une interface graphique simple. Attention, ce n'est pas la qualité de l'interface graphique qui est évaluée dans ce projet, mais vos compétences en conception et programmation objet.

- Vous **devez** utiliser la bibliothèque d'interface graphique `zen` fournie avec ce sujet (fichier `zen5.jar`).

  Pour ajouter un jar à un projet sous Eclipse :

  – Rajouter un dossier `lib` dans le répertoire du projet et y placer le fichier `.jar`.
  – Dans Eclipse, faire un clic droit sur le fichier `.jar` et choisir `Build Path > Add to Build Path`.

- On vous fournit également un mini-exemple (très incomplet) de code utilisant cette bibliothèque et suivant un modèle de développement classique appelé MVC (Modèle-Vue-Contrôleur) que vous verrez dans le cours d'IHM et que **vous devrez appliquer** dans votre code :

- Une classe `SimpleGameData` est utilisée pour gérer les données du jeu (le modèle) ainsi que toutes les actions possibles, suivant les règles du jeu.
- Une interface `GameView` implémentée par le record `SimpleGameView` permet de gérer l'affichage graphique (la vue).
- Le contrôleur implémente la boucle de jeu et la gestion des événements utilisateur (clics, touches, ...).
- On a ajouté une classe `TimeData` pour la gestion du temps.

Conseils

- Pensez à programmer "objet". Vous serez pénalisés si vous n'exploitez pas suffisamment les notions vues en cours. La hiérarchie des différentes classes de votre projet doit être soigneusement réfléchie.
- Le sujet est évolutif, respectez bien les phases de réalisation, mais gardez à l'esprit ce que vous devrez faire dans les phases suivantes lorsque vous faîtes des choix d'implémentation.
- La gestion du temps dans le jeu peut être un peu complexe, le code d'exemple fourni donne une façon de procéder simplement, n'hésitez pas à vous en servir. **Et si vous n'arrivez pas à gérer l'avancement du temps en continu, remplacez-le par le fait de devoir appuyer sur une touche pour faire avancer le héros.**

Consignes de rendu

- Ce projet est **à faire en binôme**. Si vous avez une bonne raison de demander une exception à cette règle, contactez votre enseignante (`pivoteau@univ-mlv.fr`).
- Sur e-learning, vous déposerez une archive contenant
  - **Les sources (`.java`) de votre projet** (dossier `src` sous Eclipse), correctement commentés. Il n'est pas nécessaire de fournir la javadoc de l'ensemble des classes et méthodes, mais on doit pouvoir comprendre aisément votre code grâce à son organisation, les noms de méthodes choisies et vos commentaires.
  - Un **fichier texte README.txt** qui explique
    * ce qui a été implémenté,
    * l'organisation du programme (la hiérarchie des classes implémentées),
    * les choix techniques/algorithmiques et
    * les éventuels problèmes rencontrés.

    Vous pouvez y ajouter tous les commentaires ou remarques que vous jugez nécessaires à la compréhension de votre code.
- La date limite de rendu **final** est le **lundi 6 juin 2022 à 23h55**. Passé ce délai, la note de votre projet sera 0. De plus, vous ferez deux rendus intermédiaires **obligatoires** : la phase 1 doit être terminée avant le **3 avril 2022 à 23h55** et la phase 2 complète avant le **8 mai 2022 à 23h55**. Vous déposerez, une archive du projet sur e-learning à chacune des 3 dates indiquées ci-dessus.
- Des mini-soutenances seront organisées quelques jours après le rendu de la phase 1 et le rendu final, pendant lesquelles vous ferez une démonstration sur **une machine des salles de TP** et serez interrogés sur le projet. Les deux membres du binôme doivent pouvoir répondre aux questions et leur notes peuvent être différentes en fonction de l'évaluation du travail fourni.
- Vous perdrez des points si vous ne respectez pas ces consignes.
- Si le code ne compile pas, la note de votre projet sera 0.
- Il va sans dire que si deux projets trop similaires sont rendus par 2 binômes différents, la note de ces projets sera 0.