le cnam Aquitaine

TP Noté 1

Année 2020/2021

CODE UE : USSI2T

INTITULE : Programmation avancée

DATE : 10 septembre 2020

DEADLINE : 5 octobre 2020

CONSIGNES :
- Devoir à la maison : l'internet est autorisé et nécessaire, l'échange d'idées entre les élèves est encouragé, mais le plagiat est formellement interdit. Le code sera soumis à plusieurs vérificateurs de plagiat, et tout plagiat confirmé sera pénalisé.

Procédure de remise du code source :
Le code produit sera transmis au professeur via un dépôt *Git* dédié à chaque élève, dont l'adresse lui sera préalablement fournie (par *GitHub Classroom*). Afin de ne pas perdre son travail, l'élève effectuera régulièrement des *commits + push* sur ce dépôt Git (après chaque question par exemple). Les *commits* pushés avant la deadline seront pris en compte pour la notation ; les *commits* pushés après ne seront pas visibles par le professeur ! En cas de problème technique avéré, le téléchargement du projet sur l'ENF pourra être envisagé.

Notation :
- Pour chaque exercice, la conformité de l'application aux exigences décrites dans les questions ainsi que la qualité du code produit seront prises en compte.
Vous devrez également apporter des explications (concises) de votre raisonnement et de l'implémentation dans les commentaires de code et dans le fichier « README.md ».

Devoir de Programmation Avancée (Java)

Initialisation du projet avec *Git* et *IntelliJ*

Pour commencer, suivre la procédure suivante :

1. Taper dans le navigateur le lien permettant de générer le dépôt Git sur GitHub : https://classroom.github.com/g/mU5oBPEW
2. Dans *IntelliJ*, créer un nouveau projet en clonant le dépôt Git précédemment généré : Menu principal → « Get from Version Control » → sélectionner « Git » → copier l'URL du dépôt Git → « Clone »
3. Un projet *IntelliJ* est maintenant créé et synchronisé avec un dépôt Git dédié. Si besoin, paramétrer dans *IntelliJ* la version de Java à utiliser en suivant la procédure en annexe.

Vous remarquerez que le projet généré est (quasiment) vide. Ce sera à vous de développer l'application « from scratch » !

Ne pas oublier d'effectuer un *git push* avant la deadline afin de soumettre le code source au professeur.

Projet de programmation : créer un mini jeu d'aventure en 2D

L'objectif de ce TP est d'utiliser la librairie Java **Swing** pour implémenter un prototype de jeu d'aventure en 2D.
Le sujet est volontairement conçu de manière à laisser libre cours à votre imagination : des lignes directrices sont données, mais les choix d'implémentation et de certaines fonctionnalités, ainsi que les choix de conception de l'interface graphique sont laissés libres.
Les parties **I**, **II** et **III** ci-dessous décrivent les lignes directrices à respecter impérativement.
La partie **IV** décrit de façon non-exhaustive des pistes d'amélioration possibles que vous pourrez optionnellement implémenter.
A priori, la notation sera la suivante : 16 points sur 20 maximum pour les parties **I**, **II** et **III**, ainsi que des points supplémentaires pour toute fonctionnalité « bonus » (qu'elle soit décrite ou non dans la partie **IV**). En théorie, il sera donc possible d'obtenir une note supérieure à 20/20. À noter que les points seront partagés entre la programmation (code) et les explications (commentaires et fichier « README.md ») ; une fonctionnalité correctement implémentée mais mal documentée ne permettra donc pas d'obtenir le maximum de points.

I. Concevoir et implémenter l'interface graphique

Implémenter l'ensemble des composants de l'interface graphique du jeu. Cet ensemble doit au minimum comporter les composants **1**, **2** et **3** suivants :

1) Le menu principal du jeu : il s'agit de la fenêtre s'affichant au lancement du jeu. Cette fenêtre affiche le titre du jeu, et permet de :
   a) Lancer une nouvelle partie ; cela fait disparaître la fenêtre et fait apparaître le composant **2**
   b) Quitter le programme
2) La fenêtre de sélection du héros : il s'agit de la fenêtre s'affichant au lancement d'une nouvelle partie. Elle permet de nommer le héros qui sera contrôlé par le joueur, et de sélectionner le type de personnage qui le représentera (appelé « classe ») parmi un ensemble donné à définir (pouvant comprendre des animaux et des humanoïdes). La sélection du héros par le joueur fait disparaître la fenêtre et fait apparaître le composant **3**
3) La fenêtre de jeu : il s'agit de la fenêtre principale du jeu, constituée au minimum des composants **3.a** et **3.b** suivants :
   a) Au centre, apparaît la carte du jeu en 2D, sur laquelle se trouvent le héros (contrôlé par le joueur) et les personnages non-jouables
   b) À côté de ou sous la carte, apparaît un panel contenant à la fois l'état actuel du héros (par exemple : points de vie, énergie, position sur la carte…) et les actions possibles (par exemple : attaquer, se reposer, boire, manger…). D'autre part, le joueur peut basculer sur le menu principal du jeu (composant **1**) à tout moment, en pressant une touche du clavier. Enfin, lorsque la partie est terminée, le joueur retourne sur le menu principal du jeu (composant **1**).

II. Initialiser la carte de jeu en 2D

L'un des éléments centraux du jeu est la carte de jeu en 2D représentant le monde avec lequel le joueur peut interagir par le biais du héros dont il a le contrôle. Notre prototype de jeu utilisera une simple carte en deux dimensions, représentée en vue « aérienne » (vue du dessus, à l'instar du jeu « Stardew Valley » par exemple). Cette carte est prédéfinie (« en dur » dans le code ou encodée dans un fichier) et ses caractéristiques sont les suivantes :

1) Sa taille est égale à : <taille de la fenêtre de jeu> – <taille du composant **3.b**>. Elle n'est donc pas « scrollable », c'est-à-dire que le joueur voit l'intégralité de la carte à tout moment
2) La carte est composée de « tuiles » de couleur, posées sur une grille en 2D (comme dans un jeu d'échecs). Cette grille permet de gérer plus facilement le positionnement et le déplacement des différents éléments mobiles sur la carte
3) Les tuiles représentent au minimum deux types de terrain : la terre ferme et l'eau
4) Le héros ainsi que les personnages non-jouables (animaux ou humanoïdes) sont initialisés à des positions aléatoires sur la carte.

III. Gérer le héros

Le héros est le personnage principal du jeu, créé et contrôlé par le joueur. À noter que par défaut dans ce prototype de jeu, les personnages non-jouables n'effectuent aucune action ou déplacement. Les caractéristiques du héros sont les suivantes :

1) La classe choisie affecte les attributs du héros, qui comprennent au minimum :
   a) La vélocité
   b) La force
   c) Capacité à voler (booléen)
   d) Capacité à nager (booléen)
   e) Points de vie
   f) Points d'énergie
2) Le héros peut se déplacer sur la carte. Sa position initiale correspond à une tuile (valide) aléatoire, et il peut à tout moment se déplacer sur les tuiles adjacentes (pas de déplacement en diagonale), à condition que la tuile ne contienne pas d'obstacle infranchissable (par exemple : de l'eau, un rocher…). La classe choisie affecte la façon dont le héros se déplace, notamment au niveau de la vitesse de déplacement et de la capacité ou pas à nager ou à voler
3) Dans ce prototype de jeu, l'objectif pour le joueur est de gérer l'énergie du héros : quelque soit la classe, le héros dispose d'un certain nombre de points d'énergie, qui sont dépensés lorsqu'il réalise des actions (par exemple : déplacement, attaque…), et qui de plus baisse graduellement (lentement) au cours du temps. Lorsque l'énergie tombe à 0, le héros est KO et la partie est perdue. Pour éviter l'inexorable KO, le héros peut entreprendre des actions pour augmenter son énergie (par exemple : se reposer, manger un animal chassé, boire de l'eau trouvée sur la carte…)
4) La gestion du déplacement et des actions du héros est faite via la souris, en utilisant le panel d'actions (composant **3.b**). L'ensemble des actions et déplacements possibles est mis à jour après chaque déplacement du héros sur la carte, en fonction des tuiles adjacentes à la nouvelle position du héros. En effet, en fonction de cette position, certaines actions ou déplacements sont possibles et d'autres impossibles (par exemple : se déplacer à gauche est impossible si la tuile de gauche contient de l'eau et que le héros ne sait pas nager…)
5) En plus du déplacement, le héros peut interagir avec l'environnement à travers au minimum les actions suivantes :
   a) Boire de l'eau (l'eau du jeu est bien sûr tout à fait propre à la consommation)
   b) Une action quelconque (de votre choix) avec un personnage non-joueur (par exemple : attaquer, saluer…)

IV. [Optionnel / Points bonus] Améliorer le programme

Ci-après une liste non-exhaustive et non-ordonnée de pistes d'amélioration qu'il est possible d'implémenter pour améliorer le prototype de jeu (qui ressemblera alors davantage à un véritable jeu !). À noter que vous pouvez tout à fait faire le choix d'implémenter des fonctionnalités qui ne se trouvent pas dans cette liste.

1) Générer la carte de jeu aléatoirement à l'aide d'un algorithme de génération procédurale, afin que chaque nouvelle partie soit une nouvelle expérience pour le joueur

2) Ajouter d'autres éléments (interactifs ou pas) sur la carte de jeu, afin de la rendre plus vivante et plus diversifiée (par exemple : génération aléatoire d'arbres et forêts, rochers, villages…)
3) Agrandir la carte au-delà des limites de la taille de l'écran, ce qui implique qu'elle devienne scrollable (et ainsi potentiellement zoomable)
4) Rendre le déplacement et les actions du héros réalisables via des touches du clavier (par exemple : flèches directionnelles pour se déplacer…)
5) Ajouter des animations pour les déplacements et les actions afin de rendre le jeu plus agréable
6) Ajouter la possibilité pour le joueur de paramétrer la carte de jeu (par exemple : type d'environnement (désert, île tropicale, neige), taille de la carte…)
7) Modifier la gestion des déplacements du héros pour qu'il soit en mode « point and click », c'est-à-dire que le joueur clique avec la souris sur la position de destination au lieu de diriger le héros tuile par tuile. Cela implique l'utilisation d'un algorithme de « pathfinding » pour calculer l'itinéraire du héros pour qu'il puisse se rendre automatiquement jusqu'à sa destination
8) Rendre les personnages non-jouables plus vivants (par exemple : déplacements et actions (aléatoires ou basés sur une intelligence artificielle basique)…)
9) Implémenter un système de combat basique se basant sur les attributs existants du héros tels que la force et les points de vie
10) Implémenter un système de progression basique pour le héros : en réalisant certaines actions (par exemple un combat) le héros gagne de l'expérience ce qui lui permet d'améliorer progressivement ses attributs (par exemple : force, énergie…)
11) Ajout d'une musique de fond et d'effets sonores
12) Implémenter la notion de compagnons de route : le héros peut recruter des compagnons de route (personnages non-jouables) pour le suivre et l'assister dans sa quête. Le joueur contrôle un groupe de personnages au lieu d'un unique héros.

Annexes

Quelques liens utiles :

- https://docs.oracle.com/javase/8/docs/api/ (API Java 8)
- https://docs.oracle.com/en/java/javase/11/docs/api/ (API Java 11)
- https://docs.oracle.com/en/java/javase/13/docs/api/ (API Java 13)

En cas de message d'erreur lié à la version de Java, suivre la procédure suivante :

1. Menu « File » → « Project Structure » → « Project » → modifier « Project SDK » et « Project Language Level »
2. Menu « File » → « Project Structure » → « Modules » → modifier « Language Level »
3. Menu « File » → « Settings » → taper dans la barre de recherche « Java Compiler » → modifier « Target bytecode version »

Activation des assertions au niveau de la JVM :

Menu « Run » → « Run… » → « Edit Configurations » → sélectionner la configuration à gauche → taper « -ea » dans le champ « VM options »