Programmation en Java — Projet

L3 Informatique – 1er semestre – automne 2024 – Version 3, en date du 22 novembre 2024

Sujet : Réaliser le jeu Cascadia en Java

Le but de ce projet est de réaliser une version PC offline d'un jeu de société : **Cascadia**. Il s'agit d'un jeu de stratégie où un à quatre joueurs cherchent à créer dans la région de Cascadia l'environnement le plus accueillant pour la faune locale.
Bosphore Jugé, 11 ans, qui adore ce jeu, a demandé au Père Noël d'en recevoir une version électronique ; comme il n'est pas pressé, il est prêt à attendre jusqu'au 12 janvier. Aidez-le !

Le jeu

La règle du jeu Cascadia est disponible **en ligne**. Nous vous invitons à la lire avec attention, puisqu'il s'agit du jeu que vous allez devoir programmer.

Conseils

- ▷ Lisez l'ensemble de l'énoncé avant même de commencer quoi que ce soit !
- ▷ Le but de ce projet est de nous montrer que vous savez programmer "objet". Vous serez donc pénalisés si vous n'exploitez pas suffisamment les notions vues en cours.
- ▷ Lisez bien les règles du jeu et prenez bien le temps de réfléchir à la meilleure façon de mettre en place tel ou tel mécanisme du jeu avant de commencer à coder. Si vous commencez à coder avant d'avoir suffisamment réfléchi, vous serez, en pratique, obligés de revenir en arrière, ce qui vous demandera beaucoup plus de temps et d'efforts.
- ▷ 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 faites des choix d'implantation !

Le programme à réaliser

Vous allez réaliser votre jeu en cinq phases. Chaque phase devra être terminée et fonctionnelle avant de passer à la phase suivante. Cependant, lors de la réalisation d'une phase donnée, il est bien sûr indispensable de maintenir un code modulable qui permettra d'intégrer facilement les modifications apportées lors des phases suivantes.

Phase 1 : La base

Dans un premier temps, vous devez réaliser une version simplifiée du jeu, **en ligne de commande**, dans laquelle

- ▷ au lieu d'avoir des tuiles hexagonales, ou a uniquement des tuiles carrées 1×1, deux tuiles étant considérées comme voisines si elles ont une arête commune ; ça rendra les aspects géométriques plus faciles à gérer : difficile d'afficher un hexagone en ligne de commande… ;
- ▷ comme avec des tuiles hexagonales, la tuile de départ est constituée de trois tuiles 1×1 contiguës ; on vous laisse choisir lesquelles, et quelle sera la forme (en L ou en barre) de la tuile de départ ;
- ▷ chaque tuile 1×1 contient un seul type de paysage (ce qui fait qu'on n'aura jamais besoin de les pivoter) et permet le placement d'exactement deux espèces animales : on vous laisse choisir lesquelles selon votre bon plaisir ;

- ▷ il n'y a pas de jetons Nature ;
- ▷ il y a exactement deux joueurs ;
- ▷ au lancement du jeu, on laisse à l'utilisateur le choix entre les variantes famille et intermédiaire.

Phase 2 : Affichage graphique

Une fois la phase 1 terminée, il vous est demandé de mettre en place une interface graphique simple. Vous devrez trouver une manière adaptée de laisser aux utilisateurs le choix entre l'interface graphique et l'interface en ligne de commande.
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 `zen-6.0.jar`).

  Pour ajouter un jar à un projet sous Eclipse, il faut :

  - ▷ 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) 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, ...

Phase 3 : Le jeu presque complet

Une fois la phase 2 terminée, vous devez réaliser le jeu complet, permettant de faire jouer deux à quatre joueurs humains, **à l'exception des réussites de Cascadia** (pages 12 à 14). Vous devrez donc trouver une manière adaptée de laisser aux utilisateurs le choix

- ▷ du nombre de joueurs (entre deux et quatre) ;
- ▷ entre la version en ligne de commande avec tuiles carrées, la version graphique avec tuiles carrées, et la version graphique avec tuiles hexagonales officielles ;
- ▷ des cartes de scores utilisées : variante familiale, variante intermédiaire, ou une carte (entre A et D) pour chaque espèce animale.

Phase 4 : Mode solo et réussites de Cascadia

Une fois la phase 3 terminée, vous devez inclure un mode solo, ainsi qu'un mécanisme permettant de mettre en œuvre les réussites de Cascadia d'une partie sur l'autre, et notamment d'enregistrer sa progression d'une partie à une autre. À vous de trouver une manière raisonnable de faire !

Consignes de rendu

- ▷ Ce projet est à faire en **binôme** (c'est-à-dire exactement **deux** personnes) d'un **même groupe de TP**. Vous aurez jusqu'au lundi **4 novembre** à **23:59** pour indiquer, sur e-learning, le nom de votre partenaire de projet. Si vous avez des difficultés à trouver un partenaire de projet, approchez-vous sans attendre de votre chargé de TP, pour qu'il vous aide à ce propos.

Si vous n'avez pas indiqué votre choix de partenaire à la date demandée

- ▷ et si vous n'avez pas non plus participé au TP noté du lundi 4 novembre, vous serez d'office considéré comme défaillant au projet (donc à l'année de L3) ;
- ▷ et si vous avez participé au TP noté du lundi 4 novembre, un partenaire de projet vous sera attribué d'office, de manière arbitraire.

Il vous est **très vivement recommandé** de ne pas attendre le 4 novembre pour trouver un partenaire de projet. Si vous vous êtes signalés à votre chargé de TP au plus tard le jeudi **24 octobre** à **23:59**, nous nous efforcerons de vous trouver un partenaire dès le 25 octobre, ce qui vous permettra de vous coordonner un minimum avant les vacances de la Toussaint.

- ▷ Une soutenance β sera organisée les lundi **25**, mardi **26** et mercredi **27 novembre**. Pendant cette soutenance, vous ferez une démonstration sur une machine des salles de TP et serez interrogés sur le projet. Le but de cette soutenance est, entre autres, de permettre à votre enseignant de vous encourager si vous êtes sur la bonne voie, et de vous remettre dans le droit chemin sinon.

  Il est attendu, à cette étape du projet, que la phase 1 soit terminée, et la phase 2 sérieusement entamée, si ce n'est achevée ; en effet, la phase 2 est usuellement l'une des plus difficiles pour les élèves, mais si vous n'avez rien à nous montrer, nous n'aurons rien à vous suggérer.
- ▷ Un premier travail intermédiaire, qui sera présenté lors de la soutenance β, doit être déposé sur e-learning avant le dimanche **24 novembre** à **23:59**.

  En outre, le **24 novembre** à **23:59** au plus tard, le **formulaire d'autoévaluation n°1** devra être rempli par chaque groupe.
- ▷ Une soutenance finale sera organisée le jeudi **16 janvier**. Pendant cette soutenance, vous ferez une démonstration sur une machine des salles de TP et serez interrogés sur le projet. Le but de cette soutenance est, pour vous, de nous montrer ce dont vous êtes fiers et, pour nous, de vous indiquer quels étaient les points positifs et les points négatifs de votre projet.

  Il est attendu, à cette étape du projet, que les phases 1 à 4 soient achevées.
- ▷ La date limite de rendu final est le dimanche **12 janvier** à **23:59**. Passé ce délai, la zone de rendu sera fermée, et nous ne prendrons pas en compte un éventuel rendu qui nous serait envoyé autrement (par exemple par mail). En revanche, il est possible de déposer plusieurs fois votre rendu dans le temps imparti ; seule la dernière version sera alors prise en compte.

  Enfin, le **12 janvier** à **23:59** au plus tard, le **formulaire d'autoévaluation n°2** devra être rempli par chaque groupe ; il s'agit du même formulaire que le formulaire n°1, avec deux questions supplémentaires portant sur votre prise en compte des remarques faites en soutenance β.

Rendu intermédiaire

Votre rendu intermédiaire devra consister en une archive au format `zip` : tout `rar`, `tar.gz`, `7z` ou autre ne sera pas ouvert. Cette archive contiendra :

- ▷ un répertoire `src` contenant les sources du projet ;
- ▷ un répertoire `docs` contenant un manuel de l'utilisateur (`user.pdf`) et un manuel expliquant l'architecture que vous avez choisie (`dev.pdf`) ; le manuel `user.pdf` doit être lisible par Bosphore, 11 ans, qui connaît les règles de Cascadia et essaiera d'utiliser votre programme sans avoir lu d'autre document, ni même l'énoncé du projet ;
- ▷ un répertoire `docs/doc` contenant la javadoc, écrite **en anglais** ;
- ▷ un répertoire `lib` contenant les librairies dont dépend l'application ;
- ▷ un `jar` exécutable `Cascadia.jar`, qui fonctionne avec la commande `java -jar Cascadia.jar`.

Cette archive aura pour nom `Nom1_Nom2_Cascadia.zip`, où les noms sont ceux des membres du binôme par ordre alphabétique. L'extraction de cette archive devra créer un répertoire intitulé `Nom1_Nom2_Cascadia` et contenant tous les éléments demandés ci-dessus.

Rendu final

Votre rendu final devra consister en une archive au format `zip` : tout `rar`, `tar.gz`, `7z` ou autre ne sera pas ouvert. Cette archive contiendra :

- ▷ un répertoire `src` contenant les sources du projet ;
- ▷ un répertoire `docs` contenant un manuel de l'utilisateur (`user.pdf`) et un manuel expliquant l'architecture que vous avez choisie (`dev.pdf`) ; le manuel `user.pdf` doit être lisible par Bosphore, 11 ans, qui connaît les règles de Cascadia et essaiera d'utiliser votre programme sans avoir lu d'autre document, ni même l'énoncé du projet ; le manuel `dev.pdf` doit notamment inclure une section dédiée aux améliorations et corrections apportées depuis la soutenance β ;
- ▷ un répertoire `classes`, vide dans l'archive, et qui contiendra les classes une fois compilées ;
- ▷ un répertoire `lib` contenant les librairies dont dépend l'application ;
- ▷ un `jar` exécutable `Cascadia.jar`, qui fonctionne avec la commande `java -jar Cascadia.jar`, et qui possède donc un fichier `manifest` adéquat ;
- ▷ un fichier `build.xml`, écrit à la main, qui permet de
  - ▷ compiler des sources (`target compile`) ;
  - ▷ créer le `jar` exécutable (`target jar`) ; il devra s'agir de la `target` par défaut ;
  - ▷ générer la javadoc, écrite **en anglais**, dans le répertoire `docs/doc` (`target javadoc`) ;
  - ▷ nettoyer le projet pour qu'il ne rete plus que les éléments demandés (`target clean`).

Cette archive aura pour nom `Nom1_Nom2_Cascadia.zip`, où les noms sont ceux des membres du binôme par ordre alphabétique. L'extraction de cette archive devra créer un répertoire intitulé `Nom1_Nom2_Cascadia` et contenant tous les éléments demandés ci-dessus.

Critères de notation

- ▷ Bosphore, 11 ans, doit pouvoir jouer à votre jeu et y prendre plaisir ;
- ▷ la propreté et la lisibilité du code auront un poids très important dans la note ;
- ▷ l'architecture que vous aurez définie (interfaces, classes, etc) devra être donnée dans les documents PDF et aura également un poids très important dans la note ; ainsi, votre code devra être modulable, de manière à ce qu'ajouter d'autres extensions (par exemple davantage de cartes, un niveau de plus, ...) soit aussi facile que possible ;
- ▷ votre code ne devra pas contenir de méthodes de plus de 20 lignes ;
- ▷ pas de duplication de code, et respect des principes de programmation objet ;
- ▷ pas de variable globale;
- ▷ pas de code inutile ;
- ▷ présence des différents rapports et, par conséquent, orthographe correcte !
- ▷ prise en considération des remarques faites lors de la soutenance β pour le rendu final.

De manière générale, si tout est parfait d'après votre formulaire d'auto-évaluation, et si celui-ci reflète fidèlement la réalité de votre projet, votre note devrait être excellente.

Règles à respecter impérativement – Mort subite

Voici une liste de règles qu'il vous faudra respecter impérativement. Si vous ne respectez pas ne serait-ce qu'une seule de ces règles, et en fonction de la situation et de vos explications, la conséquence pour vous pourra aller du retrait de quelques points sur votre note à un statut défaillant pour le projet, donc pour la L3.

- ▷ Vous **devez** participer à la soutenance β le **25**, **26** ou **27 novembre** ; si un seul des deux membres d'un binôme participe à la soutenance β, le membre absent sera considéré comme défaillant pour le projet.
- ▷ Vous **devez** déposer un premier travail **sur e-learning** avant le **24 novembre** à **23:59**.
- ▷ Vous **devez** participer à la soutenance finale le **16 janvier novembre** ; si un seul des deux membres d'un binôme participe à la soutenance finale, le membre absent sera considéré comme défaillant pour le projet.
- ▷ Vous **devez** déposer votre version finale **sur e-learning** avant le **12 janvier** à **23:59**.
- ▷ Dans les deux cas (version intermédiaire et version finale), votre code **doit** compiler.
- ▷ Vous **devez** inclure une javadoc écrite **en anglais** et des fichiers `user.pdf` et `dev.pdf` avec votre version finale.
- ▷ Votre archive **devra** avoir le bon nom, être une archive `.zip`, et produire un répertoire qui a le bon nom.
- ▷ Le projet ne **devra pas** utiliser ou inclure de librairie externe autre que celles indiquées dans le sujet.
- ▷ Le projet ne **devra pas** contenir de code copié-collé du net. La présence d'un tel code sera interprétée comme une tentative de tricherie, et s'accompagnera donc d'une convocation devant le conseil de discipline de l'IGM.
- ▷ Le projet **devra** gérer les entrées/sorties conformément au cours ; par exemple, il ne **devra surtout pas** utiliser `java.io.File`.
- ▷ Le projet ne **devra pas** contenir de champ avec une visibilité autre que `private`, et toute méthode de visibilité `public` devra commencer par vérifier que ses arguments sont raisonnables. Par exemple, si une fonction lance une `NullPointerException`, celle-ci doit être due à un appel à `Objects.requireNonNull` qui aura détecté que l'argument proposé était nul.

Références

1. **Ant Manual** pour la construction du fichier `build.xml` ;
2. **How to create an executable jar?**
3. La **JavaDoc** ;
4. Les **entrées/sorties sur fichier** ;
5. La bibliothèque graphique **Zen**, ses **sources** et sa **documentation** ;
6. Un **exemple de code** utilisant le modèle de développement Modèle-Vue-Contrôleur pour programmer avec Zen ;
7. Une **vidéo** indiquant comment intégrer Zen à un projet avec Eclipse ;
8. Les photographies que Bosphore a incluses dans sa lettre au Père Noël : la **liste des tuiles**, la **liste des cartes**, les cartes à utliser pour les variantes **famille** et **intermédiaire**, et un aperçu des **jetons Nature, jetons animaux, dos des tuiles, et carte de score**.