Projet de POOIG

Ce projet est basé sur le jeu `Pet Rescue Saga` que vous pouvez essayer gratuitement sur votre téléphone ou comme un jeu facebook [1]

Vous avez probablement déjà passé du temps sur des jeux de ce type, et l'intérêt ici est de vous rendre compte du travail à fournir pour en réaliser les bases. Bien entendu il ne s'agit pas de produire un résultat final de qualité professionnelle, mais d'aborder le problème de sa conception en compartimentant bien les différents aspects, quitte à les améliorer ensuite.

I) Généralités

- Le projet est à faire en binôme. Les monômes ne seront pas acceptés sauf pour des questions de parité. Vous pouvez éventuellement vous associer à quelqu'un qui n'est pas de votre groupe de TD (utilisez le forum dédié sur Moodle pour entrer en contact). Il n'y aura absolument pas de trinômes, et il est entendu que si des travaux se ressemblent trop nous prendrons les sanctions qui s'imposent.
- Il vous faut déclarer la constitution de votre binôme avant le 1er novembre, et idéalement pour vous avant les vacances de la Toussaint (voir Moodle - section projet).
- La note de soutenance pourra être individualisée, chacun doit donc maîtriser l'ensemble du travail présenté, y compris la partie développée par votre camarade.
- La soutenance aura lieu durant la période des examens, et votre travail sera à rendre quelques jours avant. Nous vous donnerons les dates exactes lorsque les réservations seront confirmées.
- Un conseil pratique : sauvegardez régulièrement votre travail, et lorsque vous envisagez une modification importante conservez bien la version antérieure. Ces précautions permettent d'éviter des catastrophes.

II) Présentation du jeu

Sur l'image ci-dessous vous pouvez observer un plateau du jeu que vous aller développer. On y distingue différents blocs de couleurs, et parfois à leur sommet des animaux y sont perchés. L'objectif est de les ramener sains et saufs sur le plancher des vaches. Pour y arriver le joueur va sélectionner l'une des cases de couleurs et, dès lors que le nombre de cases contigües de même couleur est suffisant, celles-ci disparaitront. Le reste du plateau se réorganisera pour proposer une nouvelle configuration, et le jeu continuera ainsi de suite.

1. vous pouvez trouver des versions en flash mais les navigateurs semblent être réticents à faire fonctionner ce langage qui est amené à disparaître

L'environnement général du jeu accompagne le joueur dans sa progression en lui proposant une suite de niveaux. Les premiers sont faciles puis, petit à petit, des contraintes ou des options sont introduites. Cet environnement fait partie du projet, nous vous demandons de le prévoir. En particulier nous souhaitons que les niveaux soient stockés sur le disque.

Le plus simple pour comprendre la mécanique du jeu et ses règles est d'en essayer les premiers niveaux. Vous rencontrez rapidement différentes options d'actions, et quelques variantes dans les configurations proposées ou dans les objectifs. Il vous faudra intégrer tout cela dans votre modèle.

En vous basant sur vos observations il vous faudra répondre précisément à ces questions : comment se passe la suppression, comment se passe la réorganisation, comment interviennent les obstacles fixes, quelles phases distinguer etc ... Ce sont ces analyses qui vous aideront à écrire les algorithmes correspondant.

III) Conseils pour l'implémentation

Pour pouvoir maîtriser la complexité de votre travail, il vous faut absolument bien le modéliser en le découpant en objets et méthodes significatifs. Il est très important que vous compartimentiez bien votre code, et qu'il soit documenté très clairement. Vous avez une grande liberté dans la façon dont vous ferez votre développement, mais cependant veillez à distinguer les choses conceptuellement. Voici quelques éléments pour alimenter votre réflexion :

Environnement de jeu

C'est l'endroit qui manipule les niveaux et la progression des joueurs. Vous avez l'occasion d'y faire l'accueil du joueur, d'y présenter les règles du jeu, ou de passer directement à des démonstrations précises pour gagner du temps lors de la soutenance.

Idéalement à cette étape certains éléments seront sauvegardés ou importés du disque. En java les sauvegardes se font assez simplement en faisant en sorte que vos objets implémentent l'interface `Serialisable`. Référez vous à la documentation de cette interface.

Le plateau - Les joueurs etc

Une partie de votre modèle sera constitué d'un plateau de jeu. La nature de ce jeu fait que vous y aurez certainement défini un membre de type tableau ainsi que d'autres caractéristiques à définir. Pensez par exemple que le jeu se déroule dans une section considérée visible de ce plateau. La réorganisation d'un plateau est propre aux plateaux, c'est dans cette classe qu'il vous faudra écrire les méthodes correspondantes.

La façon dont les joueurs interagissent avec le plateau est à déterminer en anticipant la signature des méthodes que vous souhaitez utiliser.

Les phases du jeu en lui même

Il est toujours préférable d'avoir plusieurs petites méthodes à écrire plutôt qu'une seule grande méthode qui ferait un travail compliqué à déchiffrer.

Si vous avez quelque part dans votre code un bloc qui fait plus d'une vingtaine de lignes, alors il est quasi certain que vous devriez vous relire pour introduire une phase intermédiaire.

Nature des joueurs

Si de prime abord on pense naturellement à un joueur humain, il n'est pas très difficile de le substituer par un joueur robot. Celui ci n'a pas besoin d'être très intelligent, mais il vous faut prévoir cette possibilité et l'illustrer. Toute l'intelligence se juge dans le choix des coups, mais vous pouvez vous contenter de prendre n'importe quel coup possible. (Dans une version plus évoluée vous pourriez prendre le coup qui fait disparaître le plus de cubes ou, dans une version vraiment plus aboutie, chercher à anticiper quelle serait la situation à 3 ou 4 coups d'avance. Toute idée est ici la bienvenue)

Vue et Modèle

Un point important est de bien séparer la vue du modèle. La *vue* désigne l'ensemble des aspects liés au rendu graphique, et le *modèle* regroupe l'ensemble des concepts qui sont sous-jacent, vraiment propres au jeu, et qui sont largement indépendant de la vue.

Ainsi, si vous souhaitez faire évoluer votre programme pour en changer la présentation, par exemple pour l'adapter à l'écran d'un téléphone ou d'une tablette, alors l'essentiel du jeu sera tout de même préservé. Les changements à faire relèvent tous de ce qu'on appelle *la vue*. De la même façon, si on souhaite changer un peu les règles, ajouter des variantes, celles ci concernent essentiellement *le modèle*.

Dans votre cas, de toutes façons, vous allez dans un premier temps présenter les choses en mode texte puisque les aspects graphiques seront abordés progressivement en cours de semestre. Puis, lorsque vous serez plus avancé dans votre réflexion, que vous aurez davantage de connaissances sur les interfaces graphiques, et que vous aurez mieux cerné ce que vous souhaitez apporter au projet, alors il sera temps de redéfinir les vues. Vous pouvez **prévoir cela en définissant assez tôt une interface ou une classes abstraites pour les vues attendues**, il suffira ensuite de l'instancier par des sous-classes plus ou moins évoluée (textuelle ou graphique). L'association entre les modèles et les vues se fera en introduisant un champs typé `VueGenerale` dans le modèle de votre jeu, et réciproquement si besoin. Pour rendre compte d'une modification, graphique ou textuelle, le jeu s'adressera à sa vue. La vue se chargera aussi de transmettre les actions choisies par le joueur au modèle.

Options

Voici quelques propositions qui peuvent vous inspirer en particulier si au cours du développement ou de la répartition des taches l'un d'entre vous a un peu plus de temps que l'autre.

Il est possible de travailler à un éditeur de niveau (sommaire), de proposer la possibilité de revenir en arrière en annulant le dernier coup, de demander de l'aide en obtenant en retour une zone *intéressante à jouer* (un concept algorithmique)

Règle universelle

Organisez-vous pour avancer petit à petit, régulièrement, sans vous perdre complètement sur les aspects graphiques.

IV) Travail demandé, rapport, soutenance

Vous devez réaliser un moteur permettant de jouer à Pet Rescue. La bonne utilisation des connaissances vues en cours constituera une part importante de l'évaluation. Même s'il était fonctionnel, un code qui ne comporterait qu'une seule classe serait un cas extrême qui sera inévitablement fortement pénalisé.

Pour illustrer la séparation de la vue et du modèle le jeu doit pouvoir être joué soit en mode graphique soit en mode texte, mais les deux options doivent être implémentées. La vue textuelle n'a pas vocation a être compliquée, elle pourra être purement descriptive.

Les travaux sont à rendre sur `Moodle` sous la forme d'une archive nommée *`nom1-nom2`* selon la constitution de votre binôme, et qui s'extraira dans un répertoire *`nom1-nom2`*. Elle devra contenir :

- les sources et ressources (images ...) de votre programme ; ne polluez pas votre dépôts avec des fichiers `.class` inutiles ...
- un fichier nommé `README` qui indique comment on se sert de votre programme (compilation, exécution et utilisation) ; faites en sorte que son utilisation soit la plus simple possible. Ne pensez pas que nous utilisons le même environnement de développement que celui que vous avez choisi. Le correcteur ne doit avoir que deux choses à faire pour tester votre travail [2] :
  1. décompresser votre dépot dans une console unix
  2. lire votre fichier README et y trouver la ligne de commande qui lance le programme
- un rapport au format *PDF* d'au moins cinq pages rédigées expliquant les parties traitées, les problèmes connus, et les pistes d'extensions que vous n'auriez pas encore implémentées. Il devra contenir impérativement une représentation graphique du modèle des classes (le plus simple est qu'elle soit manuscrite, puis scannée). Le rapport n'est en aucun cas une impression de votre code.
- toutes choses utiles pour rendre la soutenance fluide.

La soutenance devra pouvoir se dérouler sur une machine du script à partir des sources que vous avez déposées. Si tout se passe bien elle se déroulera devant un ou deux enseignants dans un mélange de questions et de tests. Il pourra vous être demandé de modifier votre code pour répondre à une question spécifique.

2. testez vous même votre rendu sur une autre machine ou pour le moins dans un dossier séparé de celui où vous avez travaillé. Si nous n'arrivons pas à exécuter votre code vous serez évidement fortement pénalisé.