Projet de POCA — SimCity

Version 2.0

Université Paris Diderot – Master 2 SRI/LC/LP

16 décembre 2013

1 Principe du projet — rappel

Le projet de POCA vise à vous faire progresser dans votre capacité *à concevoir un système objet extensible*.

Il se déroule en *deux étapes*. Dans la première étape, un sujet vous est fourni. Ce sujet est *volontairement* décrit de façon informelle, sans vous donner d'indication pour le réaliser. Vous devez donc définir vous-même les *requis* (**délivrable 1**) et l'*architecture* (**délivrable 2**) de votre projet.

Une *implémentation* écrite en Scala de cette première version constitue les **délivrable 3** (squelette) et **délivrable 4**.

Vous recevrez ensuite un *autre sujet* qui est une extension du premier. Cette extension sera objet du **délivrable 5** (plus de détails seront fournis avec le deuxième sujet) et mettra à l'épreuve votre architecture initiale : un projet bien pensé *n'aura pas besoin de modifier* le code de la première version du projet pour traiter cette extension mais seulement de *rajouter de nouveaux composants*.

2 Sujet

- une première généralisation consiste à **ajouter les événements, types des terrains, et bâtiments** de SimCity qui vous n'aviez pas inclus dans la première version. Le but ici est de convaincre (vos enseignants et vous-mêmes) de la possibilité de les ajouter sans changer la structure de classes, mais simplement en ajoutant de nouvelles classes.
  Quels sont les événements, terrains, et bâtiments les plus difficiles à ajouter ?
- une deuxième généralisation est plus technique. Vous avez développé votre jeu avec un *toolkit* graphique (p.ex. Swing, Java Curses, Java GNOME, Qt Jambi, etc.). Maintenant, vous devez **supporter un autre toolkit graphique** et permettre, au moment de l'exécution, de choisir entre les deux.
  Combien de classes avez-vous dû toucher pour réaliser cette modification ?
- une troisième généralisation, plus simple, consiste à ajouter des **réseaux non contiguës** à vos villes, p.ex. des rues avec des ponts qui lient ensemble

des cases distantes, ou un opérateur Internet qui ne connecte pas toutes les cases le long de son parcours.

- une quatrième généralisation consiste à **géolocaliser l'origine des paramètres** d'une zone de la ville (pollution, criminalité, traffic, etc.) et à permettre leur propagation. P.ex. la pollution devrait avoir comme origine des centrales à charbon et y être très haute, être moyenne dans les cases voisines, faible dans les cases plus lointaines, etc. Comment combiner les effets polluants de plusieurs centrales ? Qu'est-ce que vous allez considérer comme l'origine des paramètres intrinsèquement globaux comme le traffic ?
  Pensez aussi à ajouter d'autres paramètres, comme p.ex. la densité d'espace vert, la vie nocturne et à codifier leur effets.
- une cinquième généralisation est le support pour **plusieurs villes**. Chaque ville aura un maire diffèrent (joué par la même personne ou par des joueurs différents). Les villes seront connectées par des nouveaux types de rues, comme p.ex. des autoroutes. Les effets des constructions dans une ville peuvent se répercuter sur d'autres : p.ex. la pollution, le trafic, etc.
  Est-ce que vos modèles des données, des effets, et vos interfaces pour les villes sont suffisants pour capturer tout cela ?
- **déploiement *client-server*** : une fois réalisée la possibilité de jouer à plusieurs, il est naturel de distribuer le jeu en réseau, avec un serveur de jeu qui garde l'état et plusieurs clients qui offrent une interface utilisateur et communiquent avec le serveur.
  Est-il possible et simple d'utiliser vos classes pour compiler plusieurs exécutables, un pour le serveur, et un pour le client ?

On vous propose donc plusieurs extensions, vous devrez réfléchir à la réalisation de toutes ces extensions, et en réaliser le plus possible sous forme de code Scala.

3 Travail demandé

L'utilisation du Git est obligatoire. L'historique de ce dernier permettra de déterminer la contribution des membres de l'équipe et la gestion du temps dont vous avez fait preuve. Vous êtes libre de choisir l'hébergement Git que vous souhaitez—l'UFR offre hébergement Git mais vous pouvez aller ailleurs si vous le souhaitez. La seule contrainte et de donner *de le début* des comptes Git aux enseignants pour pouvoir suivre votre travail.

3.1 Délivrables

1. **Version 2 (deadline : 21 janvier 2014)**
   Dans un répertoire `deliverables/version2` vous devez nous soumettre la première version complète de votre projet.
   Votre projet doit aussi être testé. La qualité du code—c'est-à-dire sa correction, sa robustesse et son élégance—sera prise en compte dans la notation.