Projet de POCA — SimCity

Université Paris Diderot – Master 2

10 octobre 2013

1 Principe du projet

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 le *cahier des charges* (**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 l'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

Le projet consiste en la réalisation d'un clone du jeu SimCity [5].

Comme c'est souvent le cas dans l'industrie du jeux vidéo, vous n'avez pas une spécification précise à suivre. Vous devez donc vous documenter sur les règles et les mécanismes du jeu et sur les variantes existantes. De même, comme dans l'industrie du jeux vidéo, toutes les manières des mieux comprendre votre objectif sont les bienvenues : étudier le code des variantes libres existantes—notamment : Micropolis [3, 2], Lincity [1], et OpenCity [4]—les tester en jouant, etc.

Votre but initial est de réaliser une implémentation minimale du jeu (p.ex. ce n'est pas la peine d'implémenter tout de suite tous les types des catastrophes), en vous assurant que votre architecture permet d'ajouter plus tard très facilement ce qui manque (p.ex. en ajoutant seulement de données, ou en implantant quelques nouvelles classes).

Néanmoins, votre jeu devra permettre de générer aléatoirement le plateau de jeu, choisir parmi différentes types des bâtiments/quartiers (commercial, résidentiel, centrale nucléaire, etc.), de les connecter au réseau électrique et

routier, choisir le niveau de taxation, et de jouer en temps réel en donnant des retours (p.ex. trop de trafic, trop des taxes, trop de pollution, etc.) au maire. Réfléchissez surtout à comment implémenter et rendre extensible la liste des types bâtiments/quartiers et leurs effets sur le déroulement du jeu.

L'interface du jeu doit être minimale, pas la peine de perdre du temps à préparer des graphismes sophistiqués : si votre jeu a du succès, des artistes graphiques s'en occuperont plus tard. Pensez donc par exemple à développer une interface minimale ou l'état du jeu est représenté par des caractères textuels, ... mais rendez simple le passage ultérieur à des graphismes plus sophistiqués ! Pour contenir vos canevas, utilisez un *toolkit* graphique déjà existant, comme par exemple :

- Swing pour Scala : `http://www.scala-lang.org/api/current/scala/swing/package.html`
- Java Curses : `http://sourceforge.net/projects/javacurses/`
- Java GNOME : `http://java-gnome.sourceforge.net/`
- Qt Jambi : `http://qt-jambi.org/`

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 *dès le début* des comptes Git aux enseignants pour pouvoir suivre votre travail.

3.1 Délivrables

1. **Cahier des charges (deadline : 14 octobre 2013)**
   Dans un répertoire `deliverables/requirements/` de votre dépôt, vous devez nous soumettre une liste de propriétés (fonctionnelles et non fonctionnelles) que vous avez identifiés et que votre projet s'engage à respecter. Voir [6] et le cours de Génie logiciel avancé (M1).
2. **Architecture (deadline : 28 octobre 2013)**
   Dans un répertoire `deliverables/architecture/` de votre dépôt, vous devez nous soumettre une architecture sous la forme d'un ou plusieurs diagrammes de classes UML et d'autres diagrammes de votre choix accompagnés d'explications justifiant vos choix. Ce document sera fourni au format PDF et pourra suivre le plan suivant :

   interprétation du sujet Vous expliquerez de façon informelle ce que vous avez compris du sujet et de ces enjeux. Quels sont les problèmes techniques et conceptuels que vous avez exhibés ?

   concepts Dans cette section, vous définirez les concepts utilisés pour modéliser le problème ainsi que les invariants essentiels du système.

description de l'architecture Vous donnerez ici les diagrammes UML décrivant votre architecture et surtout sa justification.

extensions envisagées Vous énumérez ici les généralisations et les extensions que vous avez imaginées et vous expliquerez pourquoi votre architecture permet de les traiter facilement.

3. **Squelette (deadline : 4 novembre 2013)**
   (en Anglais : *walking skeleton*) Dans un répertoire `deliverables/skeleton/` vous devez nous soumettre une première ébauche de l'implémentation de votre projet. L'implémentation ne devrait pas être complète, loin de là, mais il devrait contenir toutes les abstractions prévues par votre architecture (même si beaucoup d'entre eux auront des méthodes non implémentés) et une ébauche d'interface utilisateur pour rendre claire et testable par un humain la modalité d'interaction avec l'utilisateur.

   Le squelette doit évidemment compiler à l'aide d'une commande `make` effectuée à la racine de ce répertoire. Un fichier `README` doit aussi être fourni et doit expliquer comment compiler et exécuter votre système.

4. **Implémentation (deadline : 18 novembre 2013)**
   Dans un répertoire `deliverables/version1/` 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. Enfin, une soutenance début janvier vous permettra de présenter votre projet au jury de POCA.

Références

[1] Lincity homepage. http://lincity.sourceforge.net/. retrieved October 2013.

[2] Micropolis homepage. http://www.donhopkins.com/home/micropolis/. retrieved October 2013.

[3] Wikipedia - Micropolis. https://en.wikipedia.org/wiki/Micropolis_(software). retrieved October 2013.

[4] OpenCity homepage. http://www.opencity.info/. retrieved October 2013.

[5] Wikipedia - SimCity. https://en.wikipedia.org/wiki/Sim_city. retrieved October 2013.

[6] Wikipedia - Software requirements specification. https://en.wikipedia.org/wiki/Software_Requirements_Specification. retrieved October 2013.