

# Projet de POCA — NOT POKÉMON GO

version 1.0

Université Paris Diderot – Master 2

10 octobre 2016

## 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 les *requis* (**délivrable 1**), l'*architecture* (**délivrable 2**), et une squelette exécutable de votre projet (**délivrable 3**). Une *implémentation* écrite en Scala de cette première version constitue le **délivrable 4** (squelette).

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

Ce projet consiste à la réalisation d'un moteur pour un "jeu en ligne massivement multi-joueur" (Massively Multipler Online 'MMO' game). Pour ce projet on s'inspirera du jeu POKEMON GO [1, 2], mais vous avez la liberté de changer la thématique du jeu, par exemple, en substituant les Pokemons.

Le système doit contenir au moins les composantes suivantes :

- Des joueurs.
- Des personnages de la réalité augmentée (cf. Pokemons).
- Une représentation d'une carte pour modéliser la géolocalisation. Une implémentation simple est demandée, mais pensez à la possibilité d'une future extension avec de vraies cartes à la Google Maps.
- Des représentations des objets de la vraie vie ou de la réalité augmentée (cf. des poke balls, poke stations, points d'intérêts, etc.). Pour chacun de ses objets, considérez les extensions possibles pour prochaines évolutions du logiciel.
- Des interactions entre plusieurs joueurs, et entre des joueurs et des objets, etc. Pensez à utiliser des patrons de conception objet.

Dans une première version, on considère un seul joueur, laissant l'extension multijoueur pour la version finale.

## 3 Travail demandé

Le projet est à traiter en groupes de minimum 5, maximum 6 personnes. Compte-tenu de la quantité importante de travail à effectuer, il est essentiel de vous organiser en répartissant le travail entre les différents membres de l'équipe. Vous devez suivre une méthode de conduite de projet et avoir un planning rigoureux vous permettant de tenir les deadlines. Les retards seront sanctionnés dans la notation.

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 un hébergement Git<sup>1</sup> mais vous pouvez aller ailleurs si vous le souhaitez. La seule contrainte est de donner *dès le début* des comptes Git aux enseignants pour pouvoir suivre votre travail.

### 3.1 Délivrables

Sauf si on précise le contraire, tous les documents délivrés seront au format PDF.

#### 1. Manuel d'utilisateur et cahier des charges (deadline : 18/10)

Dans un fichier `deliverables/requirements.pdf` sur votre dépôt, vous produirez une liste de propriétés (fonctionnelles et non fonctionnelles) que vous avez identifiées et que votre projet s'engage à respecter. Voir [3] et le cours de Génie logiciel avancé (M1).

#### 2. Architecture et modèle logique (deadline : 25/10)

Dans un fichier `deliverables/architecture.pdf` sur votre dépôt, vous devez nous soumettre une architecture sous la forme de plusieurs diagrammes de classes UML et d'autres diagrammes de votre choix accompagnés d'explications justificatives. Ce document sera fourni au format PDF et pourra suivre le plan suivant :

- Dans un fichier `deliverables/architecture.pdf` sur votre dépôt, vous devez nous soumettre une architecture sous la forme de plusieurs diagrammes de classes UML et d'autres diagrammes de votre choix accompagnés d'explications justificatives. 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. Attention à fournir différents points de vue à différents niveaux de détails sur votre architecture. Le but de cette partie est d'avoir une compréhension complète et global de votre application : cette explication doit passer en revue la totalité des fonctionnalités de votre logiciel sans rentrer dans les détails sans intérêt. Choisissez bien le niveau d'abstraction de vos explications !

**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.

- **Attention :** Ce travail d'analyse et de rédaction est essentielle dans le projet. Vous ne devez pas le négliger et soigner la façon dont vous communiquez avec nous à travers les documents produits.

---

1. [http://moule.informatique.univ-paris-diderot.fr:8080/users/sign\\_in](http://moule.informatique.univ-paris-diderot.fr:8080/users/sign_in)

### 3. Version 0 (deadline : 08/11) (*walking skeleton*)

Dans un répertoire `deliverables/skeleton/` vous devez nous soumettre une première ébauche de l'implémentation de votre projet. L'implémentation ne devra pas être nécessairement complète, loin de là, mais devra contenir toutes les abstractions prévues par votre architecture (même si beaucoup d'entre elles auront des méthodes non implémentées) et un jeu de tests exécutable pour le code déjà implémentés.

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

### 4. Implémentation (deadline : 22/11)

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.

## Références

- [1] Pokemon go – official website. <http://www.pokemongo.com/>. Retrieved October 2016.
- [2] Wikipedia - Pokemon GO. [https://en.wikipedia.org/wiki/Pokemon\\_Go](https://en.wikipedia.org/wiki/Pokemon_Go). Retrieved October 2016.
- [3] Wikipedia - Software requirements specification. [https://en.wikipedia.org/wiki/Software\\_Requirements\\_Specification](https://en.wikipedia.org/wiki/Software_Requirements_Specification). Retrieved October 2016.