GEN - Projet Balloon
Rapport Sprint 1

Patrick Lachaize

Le Projet Balloon est élaboré pour servir de guide aux projets des étudiants dans le cadre du cours GEN-C. Il sera géré avec une approche agile et selon la méthode Scrum.

Ce document est un exemple de rapport intermédiaire demandé à l'issue de la phase d'élaboration.

Dans cette phase, les étudiants mettront également en place l'architecture logicielle et implémenteront une première story.

*Consultez les transparents du cours « **Cycle de vie.pdf »** et* « **Scrum.pdf »** *en référence.*

Equipe de projet

Lister ici le nom d'un projet IceScrum, le repository du projet, les pseudos IceScrum et les noms des membres du groupe, leurs rôles, leurs emails. Le Product Owner est imposé : Po Genc.

Projet IceScrum : ICEGEN

Repository : https://github.com/patricklac/balloon.git

ScrumMaster : Po Genc (PG), pogenc@gmail.com (Patrick Lachaize)

Product Owner : Po Genc (PG), pogenc@gmail.com (Patrick Lachaize)

Team members :

- Patrick Lachaize (PL), patrick.lachaize@heig-vd.ch

Vision

Reprenez la proposition du sujet validée en phase initialisation, éventuellement mise à jour si elle n'a pas été validée.

Il s'agit de simuler un jeu de lancer de ballon entre joueurs. Le ballon est lancé de joueur en joueur. Lorsqu'il le possède, un joueur peut modifier la taille du ballon ! Les joueurs peuvent rejoindre ou quitter le jeu en cours de partie. Le ballon ne doit jamais être perdu.

Fonctionnalités, limites et pistes d'extension

Les fonctionnalités seront décrites sous la forme de cas d'utilisation ou de stories. Toutes les fonctionnalités du projet n'ont pas à être identifiées dans ce document, elles le seront en cours de projet. Il en faut au moins une qui sera implémentée et livrée dans ce sprint, plus quelques autres en préparation du suivant.

Des limites sont posées pour l'implémentation initiale mais des pistes peuvent être évoquées pour l'évolution du projet, sans être encore assez précises pour en faire des stories fonctionnelles (on parle alors de stories epic).

La conception de l'application devra être de type client-serveur mais dans un premier temps, l'application sera exécutée sur un seul PC dans un seul processus d'exécution. Chaque intervenant

disposera de sa fenêtre graphique au sein du même environnement d'exécution et s'exécutera dans son propre thread. Le serveur disposera également de son propre thread.

L'interface graphique sera de type client lourd (pas web) mais doit rester simple (pas d'open GL …). Elle sera conçue avec une approche Modèle-Vue-Contrôleur pour permettre la réalisation de tests automatisés (de type JUnit) sur le modèle.

Dans un deuxième temps, il est envisagé de convertir l'application avec une technologie de type RPC (Remote Procedure Call) pour que chaque intervenant puisse utiliser son propre PC.

Premières stories (la première sera réalisée dans ce sprint d'élaboration) :

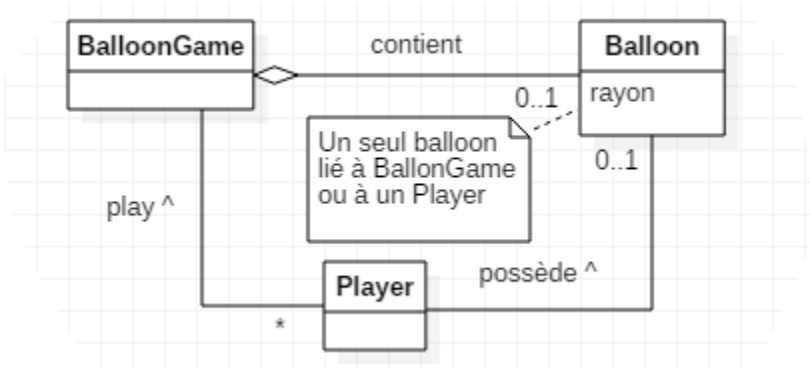
- Le jeu (BalloonGame) est lancé. Un premier joueur (Player) arrive : il obtient le ballon (Balloon) disponible dans BalloonGame et peut modifier sa taille.
- D'autres joueurs arrivent, ils sont disposés en cercle. Chaque joueur lance le ballon à son voisin de droite.
- Des joueurs peuvent rejoindre ou quitter le cercle en cours de jeu. Le ballon ne doit jamais être perdu. Si un joueur quitte le jeu, le ballon est lancé à son voisin. S'il n'en a pas (c'était le dernier joueur), le ballon est rangé dans BalloonGame.

Limites et pistes pour des extensions :

- Dans un premier temps, le cercle lui-même ne sera pas représenté géographiquement, mais c'est une extension envisageable.
- Réalisation initiale en multi-thread dans un seul processus. La conversion de l'application en mode réseau en utilisant des RPC est envisagée en extension.
- Interface en ligne de commande et interface graphique simple en 2 dimensions.

Modèle du domaine

Les concepts principaux du sujet sont modélisés sous la forme d'un diagramme de classe schématique. L'une d'elle sera un singleton correspondant au sujet lui-même.



Interface homme-machine

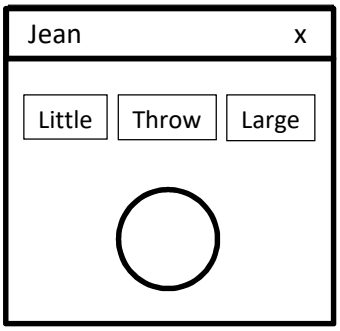
Représentation schématisée et commentée de l'IHM choisie pour l'application. Il ne s'agit que de sa version initiale, correspondant aux premières stories.

BalloonGame (interface console) :

```
> Entrez le nom d'un joueur ou quitter :
> Jean
> Entrez le nom d'un joueur ou quitter :
> quitter
```

Lorsqu'on entre « Jean », une fenêtre pour le joueur Jean est créée.

Player (fenêtre graphique) :



- Le joueur part en fermant la fenêtre
  - Le balloon est lancé ou rangé
- Little et Large diminuent ou agrandissent le balloon
- Throw lance le balloon au voisin de droite
  - Le balloon est effacé de cette fenêtre
  - Il apparait dans la fenêtre du voisin
- Throw est sans effet s'il n'y a pas d'autre joueur dans le jeu

Environnement de développement et conception de l'application

Le langage de développement et les framework nécessaires sont choisis et justifiés. La simplicité (Keep It Simple) et la connaissance partagée (par le groupe, les enseignants) doit être un critère principal. D'autres choix que ceux mentionnés ici sont possibles mais devront être validés par l'enseignant et l'assistant en début de ce sprint d'élaboration.

D'autre part, les choix de conception globaux pour l'application sont précisés ici s'il y a lieu.

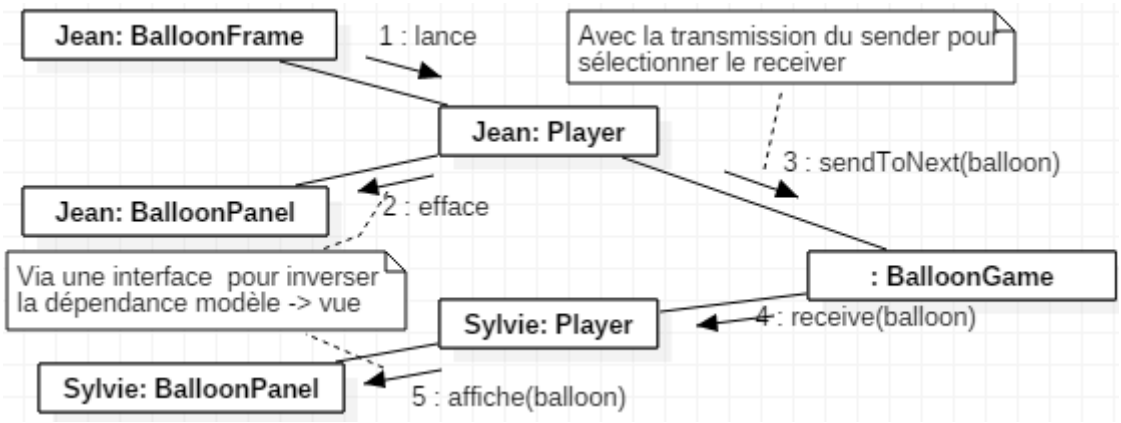
L'application sera développée en Java avec la bibliothèque graphique Swing, le framework de test JUnit et les threads de java.lang. On la structurera selon le pattern Modèle-Vue-Contrôleur.

Ce choix est justifié par la volonté de mettre en pratique les connaissances acquises dans les cours logiciel du semestre précédent et de celui en cours. Il ne nécessite pas l'acquisition de technologies supplémentaires, l'objectif du projet devant être l'apprentissage du génie logiciel.

Pour le lancer du balloon, on fait le choix de passer par BalloonGame qui officiera comme serveur. La transmission du balloon directement entre joueurs rendrait la gestion de la non-perte du balloon difficile à réaliser en cas de départ de joueurs.

Interaction principale

Interaction principale représentée dans un diagramme de séquence ou de communication commenté. Le format du diagramme de communication est souvent plus approprié pour une interaction globale schématique. On peut réaliser plusieurs diagrammes si nécessaires.



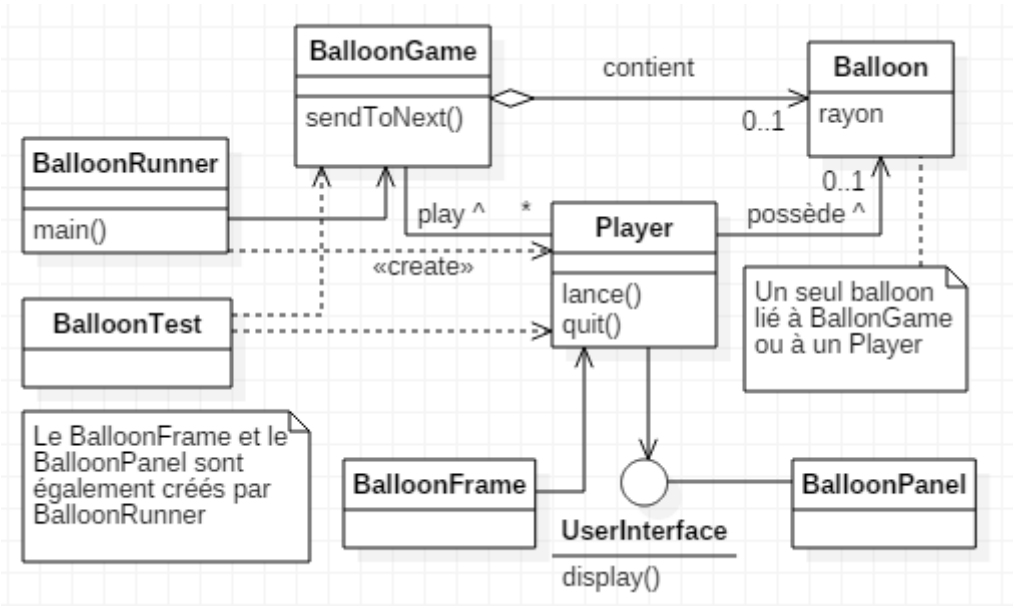
Mise en œuvre MVC : BalloonFrame (Contrôleur), BalloonPanel(Vue), Player (Modèle). Mise en œuvre client/serveur: BalloonGame comme serveur devra contenir une liste de ses clients (Players) pour pouvoir transmettre le ballon (message receive). Le lien BalloonGame / Player sera bidirectionnel.

Conception globale des classes

Le modèle de domaine est modifié, complété et commenté. Il devient un diagramme de conception. On lui ajoutera en particulier les classes nécessaires à l'interaction homme-machine.

Il n'est toujours pas demandé d'être exhaustif. L'essentiel est de montrer l'architecture initiale du système et les choix de conception générale.

Il devra également mettre en évidence une classe pour héberger les tests d'acceptation des stories qui peuvent être automatisés, c'est à dire ceux qui ne demandent pas d'interaction utilisateur.



La classe BalloonRunner est la classe de lancement de l'application (main). Elle crée le BalloonGame et les Player . Elle utilise le mode console pour son interaction.

L'interface UserInterface permet de garder le modèle (Player) indépendant de la vue, selon le pattern Observateur.

La classe BalloonTest est conçue pour permettre de tester l'ensemble du modèle.