Rapport du projet : Une ligne de produits logiciels pour la réalisation de jeux d'arcades

4 février 2010

Table des matières

1 Présentation du sujet

Ce projet est réalisé dans le cadre de l'unité d'enseignement TER, en première année de Master Informatique à l'université des Sciences de Montpellier.

Il consiste à créer un ensemble de classes qui permettra de programmer plus facilement des jeux de type "arcade". Ces jeux pourront être utilisés sur Internet, quelque soit le navigateur et le système d'exploitation de l'utilisateur. En outre ils pourront également tourner directement dans le système de l'utilisateur.

2 Définitions

2.1 Bibliothèque Logicielle

Egalement appelé librairie, une bibliothèque est constitué d'un ensemble de fonctions qui pourront être réutilisées sans avoir à les réécrire. Ces fonctions sont la plus part du temps regroupées par thémes.

2.2 Ligne de produits

Une ligne de produits est un ensemble de produits mis à disposition par une entreprise pour répondre à un même besoin, ou ayant des caractèristiques communes. Elle permet également d'augmenter la productivité tout en diminuant le temps et donc le coût de réalisation du produit.

Ligne de produits logiciels

Ensemble de logiciels appartenant à un domaine particulier, ici la création de jeux d'arcades. En informatique une ligne de produit est également appelée un framework. Généralement, un framework est codé dans un langage objet, par conséquent il est composé d'une classe mère de laquelle découle plusieurs classes filles. Ainsi un programmeur doit réimplenter les classes qui l'intéressent pour créer son logiciel. Une ligne de produits logiciels est une surcouche des bibliothèques et permet donc, non seulement de pouvoir réutiliser du code mais aussi et surtout de donner une architecture précise aux logiciels qui l'utilisent. De plus un framework doit être nécessairement extensible pour s'adapter à un plus grand nombre de logiciels.

2.3 Jeux d'Arcades

Les jeux d'arcades sont principalement des jeux à deux dimensions. La jouabilité est très simple ce qui rend ce type de jeux très populaire. Il n'y a généralement pas d'intelligence artificielle (ou très peu évolué) et encore moins de partie réseaux (les joueurs s'affrontant sur la même machine). A l'origine ce type de jeux à été créé pour les salles de jeux ou certain bar, ces pour cela que le niveau du jeux est exponentiel, afin que le joueur remette en permanence de l'argent afin de faire vivre son personnage. Cela explique aussi le fait qu'il n'y est pas

de sauvegarde des parties. Quelques exemples de jeux bien connus : Pac-Man, Casse-Briques, Ping-pong...

3 Choix d'implémentation

3.1 Choix du langage

Ce projet impose de pouvoir utiliser cette ligne de produits pour créer nos propres jeux d'arcades. Ces jeux doivent être jouable sur Internet. De nos jours il n'existe pas beaucoup de langages de programmation permettant cela. Les deux plus répandus sont JavaScript et Flex. Flex étant un tout nouveau langage développé par Adobe, il nous à parru mieux de le découvrir et d'exploiter sa richesse plutôt que de réutilisé un langage connus de tous : JavaScript.

3.1.1 Flex

Ce nouveau langage basé sur ActionScript permet de créer des clients internet riches avec une syntaxe à balises. Ainsi les scripts écrit en Flex sont compilés en SWF (format propriétaire d'Adobe également appelé Flash). Ainsi tout navigateur équipé de flash peut lire les applications Flex. Nous crérons donc notre framework en ActionScript, car c'est un langage objet, actuellement en version 3 (donc stable et évolué). Puis les utilisateurs de notre framework auront le choix de réimplenter nos classes en ActionScript ou de dévelloper directement leurs applications en Flex. L'inconvénient de ce procédé est que toute la technologie est propiétaire. Il faudra donc que l'utilisateur final ait tous les outils nécessaires à la programmation et surtout compilation de Flex.

3.1.2 ActionScript

L'action script est le langage propriétaire d'Adobe, développé dans le but de contrer JavaScript. La syntaxe est très proche de JavaScript et permet de faire des applications orientées objet. Le langage Flex combine donc la facilité d'un langage à balisage mais aussi et surtout la puissance d'un langage objet.

3.2 IDE

Adobe à fait le choix de ne pas sortir d'IDE spécialisé pour son langage préférant utiliser Eclipse par le biais d'un plugin. Eclipse est un IDE très populaire car libre de droit et très performant puisqu'en version 3. Nous ne sommes donc pas perdu puisque nous avons l'habitude de développer avec cette IDE.

3.3 Syntaxe

Nous avons fait le choix de programmer en français. Toutes nos classes et interfaces seront préfixés par un M, afin d'éviter les collisions de noms avec d'autres framework. Le M représente notre nom de groupe (MUS-D). De plus nous utiliserons la syntaxe suivante :

- MClasse : majuscule puis minuscule et majuscule pour séparer les mots.
- MIInterface : même syntaxe que les classes, mais précédée d'un I.
- méthodeDeClasse : minuscule puis majuscule pour séparer les mots.

– attribut_de_classe : minuscule puis underscore pour séparer les mots.

3.4 Commentaires

Comme nous créons une ligne de produits logiciels, nous nous devons d'être rigoureux sur les commentaires de notre code afin que les futurs utilisateurs comprennent notre façon de programmer et donc la manière dont il devront programmer leurs applications. Pour se faire nous utiliserons asDoc, une application permettant de créer des pages HTML à partir de balises placées dans notre code. Ce logiciel est distribué de base avec Flex Builder, le plugin eclipse permettant de programmer facilement en flex et donc en action script.

3.4.1 Syntaxe

4 Choix des logiciels

4.1 Eclipse

La société Adobe, ayant décidé de développer un plugin Eclipse plutôt qu'un IDE à part entière, nous sommes donc obligé de l'utiliser.

4.2 Google Code

Pour pouvoir travailler en groupe nous avons besoins d'un site internet pour regrouper nos codes et nos idées. Google Code est une plate forme nous offrant un Wiki pour échanger nos idées, un serveur SVN pour partager nos sources, un espace de stockage de fichiers et enfin un controleur de bug afin d'être prévenue d'éventuels problèmes. De plus il est entièrement gratuit et est hébergé chez Google, ce qui nous assure une plus grande sérénité quand à la sauvegarde de nos données.

4.3 SVN

Puisque Google nous offre généreusement un serveur SVN nous l'utiliserons pour garder la même version pour chacun des développeurs. Sous linux nous utiliserons le client RapidSvn pour sa simplicité (un tutorial sur son utilisation est disponible sur le site du projet). Sous window nous utiliserons Tortoise pour les même raisons.

5 Organisation

5.1 Répartition des taches

La répartition des tâches c'est éffectuée celon les envies de chacuns.
- Maneschi Romain (chef de groupe) : Coeur du projet (partie graphique) + aide sur le créateur de jeux
- Maillet Laurent : Coeur du projet (partie interne)
- Novak Audrey : Créateur de jeux
- Fhal Jonathan : Créations de 2 ou 3 jeux utilisant le framework
- König Mélanie : Coeur du projet (partie algorithmique)

5.2 Diagramme de Gantt

6 Schéma UML