Rapport de la seconde itération

Téo Pisenti

Introduction

Notre but dans cette itération était d'enrichir les fonctionnalités du jeu et d'avoir une interface graphique. Nous nous sommes donc répartis les taches en 2 groupes, premier continuerait d'enrichir les fonctionnalités du jeu (cartes chances, case prison, enchères…) à partir de la version non graphique.

Je faisais quant-à-moi parti du secon groupe avec Léo qui s'est occupé d'ajouter une composante graphique au projet.

Réalisation

Moteur graphique

Dans cette seconde itération, mon objectif a été de permettre la création de l'interface graphique. J'ai pour cela entièrement écrit un moteur de jeu en utilisant la bibliothèque LWJGL de java. Cette bibliothèque est un wrapper l'api bas niveau Opengl qui permet d'écrire des applications graphique accélérées par le GPU.

J'ai écrit cette bibliothèque pour des projets personnels et également pour ce projet (d'où des spécificités comme le IsoDrawableInstance).

J'ai pour cela dû créer une architecture complexe qui permet de gérer de façon transparente les transferts de données entre le CPU et le GPU (Vertex Buffers Objects, Array Buffer Objects…). De recréer les concepts de mesh, modèle2D, texture, uv mapping (que l'on retrouve dans les moteurs de jeu traditionnels).

Le code de cette interface graphique se divise entre différents packages : - vecteur : de simples vecteurs de dimensions 2, 3 et 4 - drawable : les objets finaux que l'on peut afficher et qui sont utilisés par l'interfaceGraphique. - window : des classes représentant une fenêtre, un bouton, et une interface pour réagir au redimensionnement de la fenêtre - glType : des objets qui permettent de manipuler les primitives définies par l'api Opengl et de gérer les transferts de données CPU<->GPU - glThread : gestion de la boucle de rendu dans un thread séparé et création d'un garbage collector pour libérer la mémoire allouée aux primitives Opengl (puisque Opengl n'utilise pas la jvm) quand ils ne sont plus référencés par une poignée dans le code java (en utilisant des WeakReference)

Ce code est entièrement documenté au travers de la javadoc de toutes les méthodes et attributs si vous voulez en savoir davantage.

Voici ci-dessous un diagramme UML réalisé en amont de l'écriture de ce code. Il est (presque) totalement juste. Vous pouvez trouver une version plus récente

(autogénérée par eclipse, bonne chance :)) dans le dossier res/architectures du git.

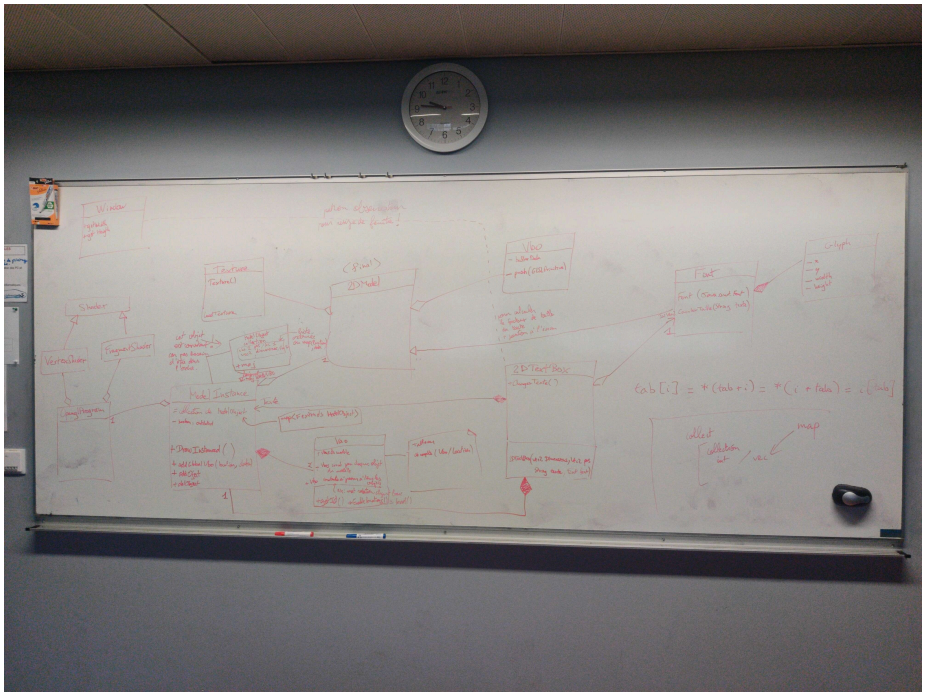

Figure 1: diagramme UML du moteur graphique en C205 le jour où le CGT à coupé le courant

Interface graphique avec Léo

J'ai également aidé Léo à réaliser une première version de l'interface graphique à partir du moteur de jeu. Sans beacoup d'interactivité pour le moment. Le clic sur les boutons écrit des messages dans le terminal.

Je lui ai expliqué le fonctionnement du moteur de jeu et lui ai fourni un petit code d'exemple.

Le but est désormais de relier cette interface avec le code de la logique du Monopoly pour rendre le jeu réellement intéractif, et basculer définitivement sur l'interface graphique du jeu.

Voici une image de l'interface actuelle (en cours de travail) :

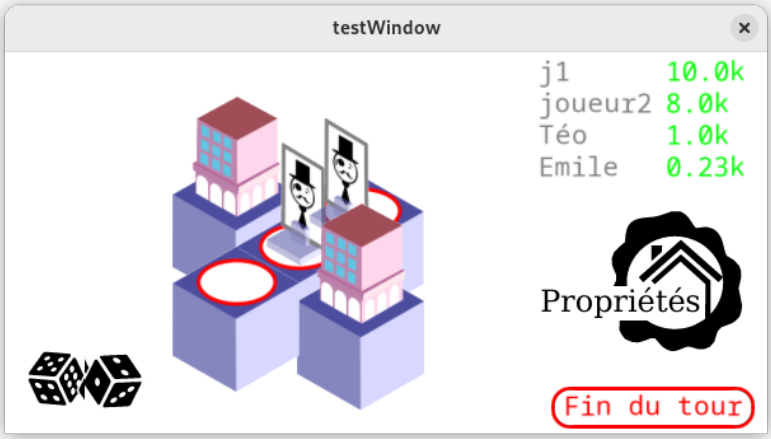

Figure 2: interface actuelle (en cours de developpement)