Projet 2 GL : Jeu vidéo

L'objectif de ce projet est de réaliser un logiciel de qualité en utilisant les méthodes et techniques issues du génie logiciel. L'une des spécificités du domaine du jeu vidéo concerne les changements continuels de besoins et de spécifications dans des délais extrêmement courts. Ces changements rapides et soudains de besoins et de spécifications rendent très délicate l'application des méthodes de développement traditionnelle. Les méthodes itératives, telles que *Scrum* ou *eXtrem Programming* (XP), bien que semblant convenir à ce type de logiciel, peinent à être efficaces si elles ne sont pas appliquées avec une grande rigueur et une grande maîtrise. Même si les exigences sont moindres que celles requises par le développement de systèmes critiques, réaliser un jeu vidéo dans un délai court (6 semaines) est donc un bon entraînement pour la mise en pratique des méthodes itératives, tout en couvrant un large spectre de domaines et de connaissances en informatique (programmation graphique, programmation sonore, programmation physique, réseau, IA (Intelligence Artificielle) programmation parallèle, etc.).

1. Contexte

L'objectif générale consiste à réaliser en 7 semaines un prototype de jeu vidéo que les étudiants pourront comparer avec les produits existants sur le marché, leur donnant ainsi une première idée du travail et des contraintes nécessaires pour réaliser un produit commercial de ce type.

2. Architecture traditionnelle

Les jeux vidéo s'appuient en grande majorité, traditionnellement, sur une architecture en 3 couches :

- les bibliothèques bas niveau (développées par les constructeurs) fournissent une interface entre le matériel et les composants moteurs ;
- le moteur de jeu (*engine*), divisé en plusieurs modules moteurs, fournit les routines génériques élémentaires (core-kernel, gestion de l'affichage et des effets graphiques, gestion des sons, gestion du réseau, etc.) ; le moteur de jeu se veut en général générique et réutilisable ;
- le *gameplay* implémente les règles et mécaniques de jeu définies dans le *Game Design Document (GDD)* – un document très proche d'un dossier de spécifications (ce document n'est pas demandé pour ce projet).

La démocratisation du développement de jeu vidéo au niveau amateur et indépendant a mené à la création de bibliothèques de fonctions dans divers langages qui reprennent plus ou moins les deux couches les plus basses, ramenant ainsi le développement du jeu vidéo plus ou moins à l'implémentation de la troisième couche. Il est toutefois intéressant de se confronter à la réalité et de de s'approprier la couche *engine* (en s'appuyant sur des librairies multimédia existantes telles que

Qt, JavaFX ou swing), et également la couche *gameplay*.

3. Enoncé du besoin

Les jeux vidéo de type arcade très populaires dans les années 80, continue d'attirer la curiosité et la convoitise des utilisateurs (joueurs), des développeurs et même des chercheurs.

Le client de ce jeu est une entreprise spécialisée dans les jeux vidéo, représentée par l'équipe pédagogique de l'UE GL. Il demande aux étudiants d'AMU de développer (1) une librairie de moteurs génériques et (2) développer un jeu vidéo (qui vous sera dévoilé au cours du développement) en s'appuyant sur votre librairie générique.

4. Moteurs

Comme dit précédemment, la couche moteur est subdivisée en sous-modules moteurs fournissant chacun des fonctionnalités génériques utilisables par la couche *gameplay*. La majeure partie de ces modules seront simplement une ré-encapsulation des méthodes fournies par les librairies utilisées. Il est peut être intéressant de découvrir et découper le moteur, tout en étant contraignant dans les techniques employées. Par exemple, le moteur d'IA peut fournir tout un tas de fonctionnalités basées sur les moteurs d'inférence, les réseaux de neurones, etc. Afin de pouvoir réaliser le projet sur la durée imparti, il est donc nécessaire de limiter le champ des algorithmes pouvant être implémentés. A titre indicatif, voilà une liste des modules moteurs traditionnels :

1. Le *core-kernel* : il s'occupe de la synchronisation des modules moteurs (qu'ils soient séquentiels ou parallèles) selon les événements reçus de la couche *gameplay* et la sortie des moteurs.
2. Le moteur graphique : se charge d'afficher, animer et gérer une scène. Habituellement, une scène de jeu vidéo est partitionnée dans une structure spatiale (graphe de scène, arbres, etc.) permettant d'optimiser le rendu et de simplifier un certain nombre de calculs.
3. Le moteur physique : fournit les objets et méthodes qui permettent de gérer le déplacement des objets du jeu dans le temps (position, collision, etc.).
4. Le moteur IA : fournit les routines de gestions d'agents intelligents (*pathfinding*, moteur d'inférence logique, etc.).
5. Le moteur sonore : fournit les routines de gestion et de synchronisation du son.

D'autres moteurs peuvent être évoqués mais qui ne font pas l'objet de ce projet, comme le moteur réseau ou le moteur HUD *(Head Up Display)* qui permettent, respectivement, de jouer en réseau et d'avoir un environnement utilisateur sophistiqué (joystick, touches de clavier, etc.).

5. Assets

De nombreux *assets* libres de droit peuvent être trouvés sur le net et utilisés pour le développement du jeu vidéo. Il est possible, également, de construire ces propres *assets* en utilisant des outils spécifiques (https://www.piskelapp.com/p/create/sprite)