Projet de Mathématiques et Physique pour le Jeu Vidéo – Phase 1

Laurianna FERRA , Clément BOURLET, Mathis RONZON, Xavier PALIX, Léna HERAU
Equipe 3 – Zelda

Déroulement de la Phase 1

Dès la création du groupe, les tâches ont été réparties entre les différents membres avec l'intention de procéder par sprints d'une semaine. Ainsi, le travail effectué, le travail en cours et les projets pour la semaine suivante étaient réactualisés à chaque séance de cours après discussion entre les membres.

Lors de la première séance, il a été décidé de partir sur les technologies OpenGL avec Qt, et d'utiliser Git comme gestionnaire de version. Ce choix a été fait au vu de l'aisance d'une partie du groupe avec les logiciels, et l'autre partie a pu se mettre au niveau rapidement.

Résumé des implémentations

En premier lieu, les classes Vector3D et Matrix ont été implémentées tôt dans le projet avec leur méthodes respectives, car elles ont nécessité plus de tests que le reste en tant que base du projet. Plus tard, les rotations par matrice 3x3 ont été implémentées en tant que méthode statique de la seconde classe pour accélérer l'écriture du code.

En parallèle, une gestion de la fréquence de rafraîchissement a été ajoutée, fondée sur l'utilisation d'une méthode de mise à jour dynamique et utilisant le temps total de calcul et affichage de l'itération précédente.

Ensuite, la classe Particle a été ajoutée au projet, avec un attribut « type » pour différencier les différents types de projectiles demandés dans cette phase. On a ici assimilé les projectiles à des particules uniques dont le rendu et les caractéristiques physiques seront modifiés.

Pour l'interface de jeu, un canon a été implémenté dans la classe Gun pour pouvoir projeter les balles et boulets avec une orientation choisie et une hauteur de base. Ce canon bénéficie d'une animation, et de contrôles au clavier affichés sur l'interface utilisateur. Enfin, des règles ont été ajoutées dans le jeu, pour donner un objectif au joueur : atteindre la cible. De ce fait, des points sont calculés en fonction de la position du projectile à l'impact.

Particularités rencontrées

La gestion du framerate variable utilise un objet appelé QTimer (propre aux bibliothèques de Qt), qui appelle régulièrement une méthode « update ». Celle-ci utilise un intégrateur d'Euler pour obtenir la nouvelle position d'une particule. Cette méthode calcule son temps d'exécution et règle la période du QTimer sur ce temps. Ainsi, la méthode update est appelée dès qu'elle est finie, imitant le fonctionnement d'une boucle while(true) sous QT. Ce choix nous permet de toujours rafraichir la

simulation physique aussi vite que la machine le peut, quitte à mobiliser toutes les ressources de celle-ci en permanence.

QT nous impose de séparer le Widget OpenGL de la MainWindow du projet. Le premier sert uniquement de rendu graphique pur pour l'affichage de la simulation en temps réel. La seconde est la base calculatoire effectuant toutes les opérations logiques telles que les entrées utilisateur, le calcul du framerate, et la boucle de jeu elle-même.

La phase 1 du projet a été réalisée de prime abord en utilisant directement les classes de base, mais l'architecture permettant d'utiliser plutôt leur héritage, afin de ne pas interférer avec les phases suivantes, a déjà été pensée et ne nécessitera qu'une conversion au moment du passage à la phase 2.

Bien que de nombreuses méthodes aient été implémentées pour les matrices (déterminants, etc…), toutes ne sont pas utilisées dès la phase 1. Néanmoins, elles sont déjà testées, et leur utilisation sera plus aisée dans les phases suivantes.

Finalement, la prise en main des technologies a posé des soucis qui ont retardé le début du développement, mais la fin de la phase a pu se passer sans problème.