

Journal de développement - Phase 3

Mathématiques et physiques pour le jeu vidéo – Groupe Sonic

Chronologie du projet

- **30 octobre:** Implémentation des classes de base (matrices 3 et 4, quaternions)
- **31 octobre :** Implémentation des tests pour ces classes. Nettoyage du dépôt, suppression des éléments uniques à la phase 2.
- **2 novembre:** Utilisation de la caméra 'EasyCamera' fournie par OpenFrameworks, déplacements dans l'espace avec les touches directionnelles.
- **4 novembre:** Classe de base pour les RigidBody.
- **7 novembre:** Implémentation des fonctions majeures (update et draw) des RigidBody. Ajout de quelques opérateurs manquants.
- **12 novembre:** Premier test de la rotation d'un RigidBody. Implémentation d'un raycast afin de viser un point pour lui appliquer une force plus tard.
- **14 novembre:** Implémentation des points d'application des forces ainsi que du torque.
- **16 novembre:** Liaison de toutes les fonctionnalités entre elles dans le GameContext, notamment l'application d'une force au point de contact du raycast. Ajout de la possibilité de faire apparaître des RigidBody à densité variable devant la caméra. Quelques bugs ont été corrigés, et le README a été mis à jour avec l'ensemble des nouveaux contrôles.

Difficultés rencontrées

La phase 3 du projet a été plus facile à mettre en place que la seconde. Nous nous sommes mieux organisés suite à l'échec de la phase précédente, avec l'instauration d'une séance de travail en commun par semaine. Le travail était cependant compliqué à répartir de manière uniforme pour chaque membre de l'équipe. En effet, certaines des tâches auraient été plus idéales à faire par une seule personne, afin d'éviter les dépendances de deux personnes ne travaillant pas au même moment.

Astuces de programmation apprises

Comme il s'agissait de notre première phase avec un environnement 3D, nous avons dû mettre à jour les contrôles afin d'avoir un déplacement dans l'espace, contrairement à l'écran fixe des deux premières phases. Le raycast a également été implémenté pour cette phase du projet. Autrement, cette phase ressemblant beaucoup à la première, nous connaissions donc déjà les concepts de programmation nécessaires pour venir à bout de celle-ci.

Choix de développement

De la même manière que pour la masse des particules, nous avons choisi de stocker le tenseur d'inertie inverse afin d'éviter des calculs redondants. En effet, celui-ci n'est utilisé que pour mettre à jour l'accélération angulaire, nous l'utilisons donc seulement sous sa forme inverse. De plus, comme nous n'utilisons que des RigidBody de forme parallélépipède rectangle, le tenseur d'inertie est une matrice diagonale et peut être facilement inversée.

Nous avons également choisi de modéliser le RigidBody avec 3 axes représentant les demi-dimensions de la boîte. Les axes initiaux (avant toute rotation) sont également stockés dans l'objet, premièrement pour avoir accès aux dimensions de celles-ci sans calcul, mais surtout pour appliquer la rotation de manière absolue à partir des axes de départ. En effet, un calcul itératif résulterait en une perte de précision rapide des flottants, déformant la boîte initiale plus le temps passe. Les axes après la rotation sont stockés dans la classe afin de les recalculer entre l'update et le draw.

Enfin, l'ajout de force a dû être modifié pour supporter les points d'applications de force, contrairement à ce que l'on se contentait de faire jusqu'à présent avec les particules.