Fanny Monet MONF10589604 équipe 14

Allan Crucerey CRUA16099505

Louis Lepoutre LEPL14039608

Journal de développement

Première revue de sprint :

Pour commencer, nous avons dû nous familiariser avec le langage C++ via des tutoriels sur internet puisque c'est la première fois que nous utilisions ce langage.

Pour travailler en groupe nous avons créé une interface GIT (un environnement de travail). Nous avons passé un peu de temps pour configurer notre répertoire et préparer notre environnement de travail.

Nous avons appris à séparer les classes en « .h » et « .cpp » ce qui facilite la clarté du code, mais aussi son organisation.

Au début du projet, lors de la création de la classe vecteur nous voulions pouvoir choisir entre travailler avec un vecteur à deux dimensions et un vecteur à trois dimensions. Cela compliquant la réalisation de notre travail et n'étant pas très explicite nous avons choisi de le retirer et de garder un vecteur en trois dimensions.

Pour la création de la classe particule, notre problème était de savoir où définir la position, la vitesse et l'accélération de l'objet particule dans la classe main ou particule. Finalement on a défini la position et la vitesse dans particule.

Dans la classe particule, nous avons eu un problème pour créer la fonction qui met à jour la position et la vitesse de la particule. En effet, a la base nous n'avions qu'un ensemble position et vitesse qui prenait les valeurs en (x, y, z) de la position et de la vitesse. Notre fonction ne fonctionnait pas et n'était pas clair, car on devait créer une boucle for pour effectuer les opérations sur chaque argument.Nous avons passé par trois paramètres en position (posx, posy, posz) et trois paramètres en vitesse (vitx, vity, vitz) ce qui nous permet de rendre le calcul plus simple et plus facile pour comprendre les erreurs.

Nous avions mis la gravité sur les trois axes x, y et z, alors qu'il fallait mettre la gravité seulement sur un axe (ici l'axe z).

Pour les calculs réaliser à chaque frame nous avons eu des difficultés. On a essayé d'adapter le programme au réel, mais cela ne fonctionnait pas, car « visual studio » mais du temps à calculer par rapport au temps réel, donc on pouvait perdre des espaces de temps. Donc on a pris comme temps = « 1/30 * temps » pour calculer en fonction du temps machine.

Nous avons perdu plusieurs heures à essayer d'intégrer « glut » une interface graphique à notre projet sans réussir à le faire fonctionner. Nous avons donc décidé de nous contenter d'un affichage console par manque de temps.

Deuxième revue de sprint :

Pour continuer le projet nous avons d'abord repris notre class vecteur que nous avons corrigée en vecteur 2D et vecteur 3D.

Nous nous sommes ensuite penchés sur la création des class ParticuleForcesGenerator, ParticuleSpring et RegistreForce. Nous n'avons pas eu de difficultés pour ParticuleSpring et RegistreForce.

En revanche pour RegistreForce plusieurs problèmes sont survenus, sur la compréhension de l'algorithme, notamment pour savoir comment enregistrer la particule et sa force dans une liste. Le deuxième problème étant dû à une erreur de SDK sur visual studio et de versions. Lors de la compilation, le programme nous retourné plusieurs erreurs qui étaient dû à un problème de SDK sur l'un des ordinateurs.

Par la suite, on s'est concentré sur la création des class ParticuleBouyancy, ParticuleBungee, ParticuleCable, ParticuleContact, ParticuleContactGenerator, ParticuleContactResolver, ParticuleTige, DragGenerator, ParticuleStiffSpring et ParticuleLink. Nous avons aussi effectué une correction sur GeneratorForceGravite afin de mettre la bonne valeur pour la gravité.

Dans les class ParticuleCable et ParticuleLink nous avons rajoutés les méthodes pour introduire les collisions.

Nous n'avons toujours pas réussi à intégrer une interface graphique, l'intégration de la librairie graphique n'arrive pas à se faire dans notre projet, nous avons en effet un problème de versioning. Notre interface ne fonctionnant pas nous avons décidé de ne pas l'intégrer au projet afin de pouvoir le compiler.

30/10/2018 : Implémentation de la classe graphique qui fonctionne.

Troisième revue de sprint :

Nous devions créer un corp rigide pour notre objet, le premier travail était d'inclure une rotation sur un cube et le deuxième travail d'effectuer une collision entre deux rectangles.

Ainsi, nous avons commencé à créer une classe Matrice3 et Matrice4 ainsi que la classe Quaternion avec les méthodes associées tel que la conversion du quaternion en matrice. Dans quaternion quelques difficultés sur la méthode UpdateVelocityAngulaire(), surtout sur l'écriture de la formule.

Pour la création des classes RigidBody et RectangleBody nous y avons d'abord implémenter les méthodes associées et résolution le problème du Drag.

Ajout des méthodes de conversion du quaternion en Matrice 3x4, des conversions des quaternions en angle d'Euler, et du tensor d'inertie.

Ajout du damping pour les classes RigidBody et RectangleBody.

Nous avons essayé de fixer le problème de la création d'un rectangle dans l'interface graphique, mais deux problèmes se pose à nous :

- Nous n'arrivons pas à générer deux rectangles en même temps. Les deux ne peuvent pas coexister et ne sont pas indépendant l'un de l'autre.
- Le carré n'a pas de rotation, il reste fixe malgré l'update de la rotation à chaque frame.

Nous essayerons par la suite de résoudre le problème de rotation du cube.

4ème revue de sprint :

Sur la création de la détection des contacts en 3D nous sommes d'abord partis sur la détection par des sphères (« BundingSphere ») mais aillant des problèmes d'intégration nous avons finis par choisir des rectangles englobant nos objets.

Des problèmes sont survenus sur la réalisation de l'octree, notamment sur la compréhension mais surtout faire fonctionner l'octree. En effet, nous avons dû reprendre le code plusieurs fois voir le refaire pour essayer de le faire fonctionner correctement.

Sur les classes « contact » et « collisionData » nous avons eu des problèmes sur la compréhension et la réalisation des méthodes « generatecollision ».

Nous avons aussi dû corriger l'affichages des rectangles, notamment celui du grand rectangle. Les difficultés étant d'avoir un rendu visuel propre.

Nous avons fixé aussi « TransformMatrice », « INverseInertie » pour obtenir le mouvement du rectangle. Beaucoup de correction de la phase précédente ont dû être corrigé.

Nous avons des difficultés sur la « narrowphase », nous n'avons pas réussi intégrer une liste de collisions.