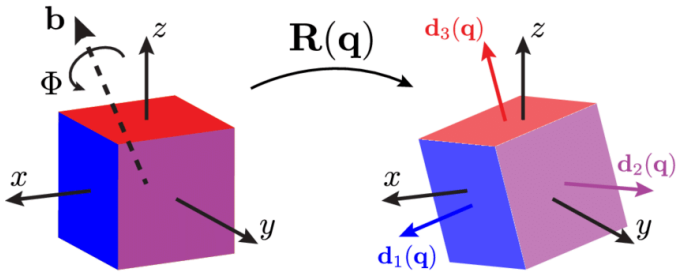
R5.DWeb-DI.07 - TP4

Vous avez vu dans un TP précédent la physique des corps déformables. Nous allons maintenant revenir à ce qu'on appelle les "corps rigides", qui peuvent être représentés en général par deux variables :

- la position du centre de gravité de l'objet
- l'orientation de l'objet (qu'on appelle aussi **quaternion**)



Ceci permet de conserver au cours du temps une orientation cohérente, surtout dans le cas de collisions entre objets. Différentes librairies JS sont disponibles, notamment Ammo et Rapier, mais nous allons utiliser **Cannon** (un peu plus ancienne mais qui nous suffira largement). Le principe est souvent le même : la lib gère les objets physiques, qui sont ensuite liés à des objets géométriques par leurs propriétés **position** et **quaternion**.

Exercice 1

Commencez par reprendre l'exemple de base sur cannon-es. Vous aurez besoin de télécharger et d'importer la lib dans votre code :
https://raw.githubusercontent.com/pmndrs/cannon-es/master/dist/cannon-es.js

1. A la fin du tutoriel vous devez obtenir une simple sphère qui tombe sur le sol, ajoutez la possibilité de faire sauter la sphère dans une direction aléatoire avec la barre Espace : https://mediaserver.unilim.fr/videos/18082024-112620/
2. Remplacez la sphère par un cube en l'empêchant de dépasser les limites du plan : https://mediaserver.unilim.fr/videos/18082024-112953/
3. Les collisions entre objets sont gérées de façon minimale, surtout pour éviter les interpénétrations. Mais on peut rajouter des objets de type **Material** et **ContactMaterial** pour introduire des coefficients de friction et de restitution comme sur cet exemple : https://pmndrs.github.io/cannon-es/examples/bounce
   Utilisez ce principe pour gérer plusieurs cubes qui rebondissent sur le sol et entre eux : https://mediaserver.unilim.fr/videos/18082024-113310/

Exercice 2

Reprenez ces principes pour implémenter un petit jeu de chamboule-tout avec six cylindres et une sphère. Commencez par bien placer les boîtes et modifiez les paramètres pour utiliser un

SplitSolver qui réduira les instabilités :
https://pmndrs.github.io/cannon-es/examples/split_solver

La balle est lancée par un clic de souris sur l'une des boîtes :
https://mediaserver.unilim.fr/videos/18082024-125540/

Exercice 3

Le but est maintenant de coder un élément qui pourrait servir par exemple pour un menu dans une page web : https://mediaserver.unilim.fr/videos/20082024-171855/

Vous avez plusieurs problèmes à résoudre, qu'il vaut mieux traiter dans l'ordre :

1. Votre code devrait pouvoir afficher n'importe quelle chaîne de caractères (sans espace)
2. La gravité va dans le sens -Z et chaque lettre correspond à une sphère CANNON dans le sens +X. Elles sont initialisées à une position Z=10, un plan (invisible) permettant de les faire rebondir et de les empêcher de "tomber" indéfiniment
3. Un clic de souris sur une lettre permet de la faire "sauter" dans le sens +Z
4. Les lettres sont liées entre elles (la 1ère avec la 2ème, la 2ème avec la 3ème, etc) par des contraintes, qui sont similaires aux ressorts vus avec les objets déformables. Voici le code que vous pouvez utiliser pour lier une lettre "textBody" à la précédente :

```
            const constraint = new CANNON.HingeConstraint(previous,
textBody, {
                pivotA: new CANNON.Vec3(textBody.position.x -
previous.position.x, 0, 0),
                pivotB: new CANNON.Vec3(0, 0, 0),
                axisA: new CANNON.Vec3(1, 0, 0),
                axisB: new CANNON.Vec3(1, 0, 0),
            });
            constraint.collideConnected = true;
            world.addConstraint(constraint);
```

   Essayez d'expliquer à quoi correspondent les différents paramètres.
5. Pour éviter que le texte ne se déplace trop après plusieurs clics sur les lettres, on peut ajouter "manuellement" une petite force qui ramène la première lettre vers sa position X d'origine.

Exercice 4

Et maintenant du basket, avec des modèles de https://poly.pizza/

Vidéo : https://mediaserver.unilim.fr/videos/22082024-193347/

Plusieurs éléments peuvent être repris des TP précédents, notamment la caméra qui “tracke” le ballon et se déplace avec les flèches du clavier en restant toujours à la même hauteur. La boîte de dialogue permet d’afficher (via des objets ThreeJS) les objets CANNON statiques ajoutés sur le cercle, le panneau et le poteau. Il sera sûrement nécessaire de tâtonner pour bien les positionner, et d’ajuster les paramètres physiques pour la friction et la restitution lors des collisions avec le ballon associé à une sphère CANNON dynamique.
Le shoot est déclenché avec la touche Espace, et tient compte des paramètres de puissance et d’inclinaison qui peuvent être réglés dans la boîte de dialogue. Vous pouvez essayer de détecter ensuite si le ballon passe dans le cercle, et d’améliorer un peu les collisions montrées sur la vidéo...

Pour aller encore plus loin on pourrait animer le filet (mais il faudrait modifier le modèle 3D), CANNON offrant un modèle de type “masse-ressorts” qui permet de simuler des objets de type “tissu” :
https://tympanus.net/codrops/2020/02/11/how-to-create-a-physics-based-3d-cloth-with-cannon-js-and-three-js/