Session 13 : Joints, Moteurs & Raycasts

> **Introduction**
>
> Pour le moment, nous avons des briques qui tombent. C'est amusant, mais pour faire un vrai jeu (ou une machine complexe), nous devons relier les objets entre eux.
>
> C'est le rôle des **Joints** (Articulations) et des **Constraints**.

1. Les Joints (Articulations)

Un Joint relie deux RigidBodies (A et B) et limite leurs mouvements relatifs.

> **Les Types Principaux**
>
> - **Fixed Joint :** Colle deux objets ensemble (ex: souder deux pièces).
> - **Revolute Joint (Hinge) :** Une charnière. Rotation libre autour d'un axe (ex: Roue de voiture, Porte, Genou).
> - **Prismatic Joint (Slider) :** Glissière. Translation libre sur un axe (ex: Ascenseur, Piston).
> - **Spherical Joint (Ball & Socket) :** Rotation libre sur tous les axes (ex: Épaule, Attelage de remorque).

> **Ancres (Anchors)**
>
> Pour définir un joint, il faut préciser **où** il s'accroche sur chaque objet.
> - Anchor_A : Point d'attache dans le repère local de A.
> - Anchor_B : Point d'attache dans le repère local de B.

Créer une Charnière (Rapier).

```
// On veut attacher une roue (bodyB) à une voiture (bodyA)
// L'axe de rotation est l'axe X (1, 0, 0)

let jointDesc = RAPIER.JointData.revolute(
    { x: 0.0, y: -0.5, z: 0.0 }, // Anchor A (sous la voiture)
    { x: 0.0, y: 0.0, z: 0.0 },  // Anchor B (centre de la roue)
    { x: 1.0, y: 0.0, z: 0.0 }   // Axe de rotation
);

world.createImpulseJoint(jointDesc, bodyA, bodyB);
```

2. Moteurs & Limites

Les joints ne sont pas forcément passifs. On peut les motoriser ou restreindre leur mouvement.

> **Contrôle actif**
>
> - **Limits :** Empêcher une porte de s'ouvrir à plus de 90 degrés. `joint.setLimits(-PI/2, PI/2)`

- **Motor (Velocity) :** Faire tourner une roue à vitesse constante.
  `joint.configureMotorVelocity(10.0, 50.0)` (Vitesse cible, Force max)
- **Motor (Position) :** Servo-moteur qui cherche à atteindre un angle précis.
  `joint.configureMotorPosition(targetAngle, stiffness, damping)`

3. Raycasting (Interaction)

Comment cliquer sur un objet 3D ? Comment savoir si mon personnage voit l'ennemi ? On lance un rayon invisible (**Ray**) et on demande au moteur physique ce qu'il touche.

../images/raycast.png

Fig. 1. – Le Raycast part de la caméra et traverse le monde

Tirer un rayon.

```
let ray = new RAPIER.Ray(origin, direction);
let maxToi = 100.0; // Distance max
let solid = true;   // Considérer les objets pleins comme obstacles

let hit = world.castRay(ray, maxToi, solid);

if (hit != null) {
    // On a touché quelque chose !
    let collider = hit.collider;
    let point = ray.pointAt(hit.timeOfImpact); // Position précise

    console.log("Touché : ", collider.handle);
}
```

4. Travaux Pratiques

Voir le sujet « Siege Engine - Partie B ». Vous allez construire une catapulte fonctionnelle avec des joints et interagir avec la souris.