

# TP : Le Laboratoire de Chocs 1D

## 🔗 Objectifs du TP

Nous allons implémenter la réponse physique d'une collision entre deux boules de masses différentes.

Pour simplifier le problème et isoler les mathématiques de l'impulsion, nous travaillons dans un « Laboratoire 1D » :

- Pas de gravité.
- Mouvement uniquement sur l'axe X.
- Pas de friction pour l'instant.

## 1. Analyse de la Scène

Le code de démarrage (Template.js) contient :

- Une **Boule Rouge (A)** : Lourde ( $m = 5$ ), lancée vers la droite.
- Une **Boule Verte (B)** : Légère ( $m = 1$ ), immobile (actuellement désactivée).
- Une interface **GUI** : Pour modifier les masses, la vitesse et le temps en direct.

**État actuel** : La boule rouge traverse le vide et rebondit bêtement sur les murs (inversion simple de vitesse).

## 💡 2. Rappel Théorique : L'Impulsion

Lors d'un choc, nous devons calculer une impulsion  $j$  (scalaire) et l'appliquer aux deux objets.

**Formule de l'Impulsion (avec Masse Réduite) :**

$$j = -(1 + e) \cdot v_{\text{rel}} \cdot \left( \frac{m_A m_B}{m_A + m_B} \right)$$

- $e$  : Coefficient de restitution (1 = Élastique, 0 = Mou).
- $v_{\text{rel}}$  : Vitesse relative de rapprochement (

$$(\vec{v}_A - \vec{v}_B) \cdot \vec{n}$$

).

**Application (Action-Réaction) :**

$$\vec{v}'_A = \vec{v}_A + \left( \frac{j}{m_A} \right) \vec{n}$$

$$\vec{v}'_B = \vec{v}_B - \left( \frac{j}{m_B} \right) \vec{n}$$

### 3. Vos Missions (Code)

**Étape A : Activer la Boule B** Dans le fichier `init()`, décommentez les lignes créant la deuxième boule. Relancez : les boules devraient se traverser comme des fantômes.

**Étape B : Implémenter le « Solver » (Fonction `resolveBallCollision`)** Repérez la zone `TODO ÉTUDIANT`. Vous devez traduire la physique en JavaScript (Three.js).

1. Calculez la Normale  $\vec{n}$  (Vecteur unitaire de B vers A).
2. Calculez la Vitesse Relative  $v_{\text{rel}}$  (Produit scalaire).
3. **Sécurité** : Si  $v_{\text{rel}} > 0$  (elles s'éloignent), on arrête (return).
4. Calculez la Masse Réduite et l'Impulsion  $j$ .
5. Appliquez le changement aux vitesses `velA` et `velB`.

*Indice : Utilisez `addScaledVector(vector, scale)`.*

### 4. Expérimentations (Observations)

Une fois votre code fonctionnel, utilisez la GUI pour tester ces scénarios physiques classiques.

#### Scénario 1 : Le Pendule de Newton

- Réglez  $m_A = 5$  et  $m_B = 5$  (Masses égales).
- Restitution  $e = 1$ .
- **Observation** : La boule rouge doit s'arrêter net. La verte doit partir avec toute la vitesse.

#### Scénario 2 : Le Mur

- Réglez  $m_A = 1$  (Légère) et  $m_B = 100$  (Très lourde).
- **Observation** : La rouge rebondit en arrière. La verte bouge à peine.

#### Scénario 3 : Le Bulldozer (Effet Catapulte)

- Réglez  $m_A = 100$  (Très lourde) et  $m_B = 1$  (Légère).
- **Observation** : La rouge continue presque sans ralentir. À quelle vitesse part la verte ? (Indice :  $2 \times v_A$ ).

#### Scénario 4 : La Pâte à modeler

- Mettez la restitution  $e = 0$ .
- **Observation** : Les boules doivent se coller et avancer ensemble à une vitesse moyenne.

### 💡 Astuce Debug

Si vos boules disparaissent ou accélèrent à l'infini :

1. Vérifiez le signe dans l'application de l'impulsion ( $-j$  pour B).
2. Vérifiez la condition `if (vRel > 0) return;`.
3. Utilisez le slider « Time Scale » pour mettre le temps au ralenti (0.1) et voir exactement ce qui se passe au moment de l'impact.