Physique du Jeu : Les 3 Lois de Newton

Ou « Comment programmer l'univers »

Isaac Newton n'a jamais codé en JavaScript, mais sans lui, aucun jeu vidéo moderne n'existerait. Ses trois lois sont les règles fondamentales du moteur physique que nous allons construire.

1. La Première Loi : L'Inertie

Le principe de la « Flemme »

> **La Loi :** Un objet au repos reste au repos, et un objet en mouvement continue en ligne droite à vitesse constante, **sauf** si une force extérieure agit sur lui.

Ce que ça veut dire : Les objets ont une résistance naturelle au changement. Ils veulent continuer à faire ce qu'ils font déjà. Pour changer la vitesse d'un objet (accélérer, freiner, tourner), il **faut** appliquer une force.

Exemple « Vie de tous les jours » : Vous êtes debout dans le bus. Le bus freine brusquement. Votre corps continue d'avancer et vous manquez de tomber. C'est l'inertie : votre corps voulait garder sa vitesse.

Exemple « Jeu Vidéo » :

- **Space Invaders / Mario (Sol)** : Quand vous lâchez la manette, le personnage s'arrête instantanément. *C'est physiquement faux !* (C'est comme s'il y avait des frottements infinis).
- **Asteroids / Dead Space (Espace)** : Vous donnez une impulsion au vaisseau. Même si vous lâchez les commandes, le vaisseau continue d'avancer pour toujours dans la même direction. C'est ça, la vraie inertie (pas de frottements dans l'espace).
- **Niveaux de glace** : Pourquoi Mario glisse ? Parce que les frottements (la force extérieure qui freine) sont réduits. L'inertie reprend le dessus.

---

2. La Deuxième Loi : La Dynamique

L'équation du Moteur Physique

C'est la loi la plus importante pour nous, car c'est celle qu'on code littéralement dans `animate()`.

> **La Formule :** $\vec{F} = m \cdot \vec{a}$
>
> Pour le codeur, on l'écrit plutôt :
>
> $$\vec{a} = \frac{\vec{F}}{m}$$

Ce que ça veut dire : L'accélération ($\vec{a}$) dépend de deux choses :

1. La puissance de la poussée ($\vec{F}$).
2. La lourdeur de l'objet (m).

Exemple « Vie de tous les jours » : Imaginez un caddie de supermarché.
- **Vide (m petit)** : Une petite poussette le fait partir à toute vitesse.
- **Plein d'eau (m grand)** : La même poussette le fait à peine bouger.

Exemple « Jeu Vidéo » (Balancing) : Imaginez un jeu de tir (FPS) avec deux classes de personnages :
- **Le Scout (Masse = 50kg)** : Si le joueur appuie sur « Avancer » (Force = 500N), il accélère à 10m/s^2. Il est agile.
- **Le Tank (Masse = 200kg)** : Avec la même touche « Avancer » (Force = 500N), il accélère seulement à 2.5m/s^2. Il est lent et lourd.

Dans Three.js : C'est ici qu'on calcule `acceleration`. Si on veut simuler du vent, de la gravité ou des ressorts, on additionne toutes les forces, on divise par la masse, et on obtient l'accélération pour mettre à jour la vitesse.

---

3. La Troisième Loi : Action-Réaction

Le principe du Karma immédiat

La Loi : Pour toute action (force), il y a une réaction égale et opposée.

$$\vec{F}_{A \to B} = -\vec{F}_{B \to A}$$

Ce que ça veut dire : On ne peut pas toucher sans être touché. Si vous poussez un mur, le mur vous pousse en retour. Les forces vont toujours par paires.

Exemple « Vie de tous les jours » :
- **Le Skateboard :** Pour avancer, vous poussez le sol vers l'**arrière** avec votre pied. En réaction, le sol pousse votre pied (et vous) vers l'**avant**.
- **Le Ballon de baudruche :** L'air est éjecté vers l'arrière, le ballon part vers l'avant.

Exemple « Jeu Vidéo » :
- **Recul des armes (Recoil)** : Dans *Call of Duty*, quand vous tirez une balle vers l'avant (Action), votre arme et votre caméra remontent vers l'arrière (Réaction).
- **Rocket Jump (Quake / TF2)** : Vous tirez une roquette sur le sol. L'explosion exerce une force vers le bas sur le sol. En réaction, le sol (et l'explosion) exerce une force vers le haut sur le joueur, le propulsant dans les airs.
- **Collisions (Billard)** : Quand la boule blanche tape une boule rouge, la blanche s'arrête (ou recule) car la rouge lui a « rendu » le choc.

Résumé pour le développeur

Loi	Concept	Implémentation Code
1. Inertie	Les objets gardent leur vitesse.	Ne remettez pas `velocity` à 0 à chaque frame ! Laissez-la vivre entre les frames.
2. F = ma	Force → Accélération.	`acc = forces.divideScalar(mass)`
3. Action-Réaction	Les interactions sont doubles.	Collision : Si A repousse B, alors B doit repousser A avec la même force.

Tableau 1. – Aide-mémoire des lois de Newton