

Session 9 : L'Intégration de Verlet

Qu'est-ce que l'Intégration de Verlet ?

Contrairement à l'intégration d'Euler, l'algorithme de Verlet ne stocke pas explicitement la vitesse d'un objet.

La vitesse est déduite de la différence entre la position actuelle et la position à l'instant précédent. C'est un schéma d'intégration symplectique, ce qui signifie qu'il conserve l'énergie de manière bien plus stable qu'Euler.

💡 Le Concept de Base

Imaginez que vous savez où vous êtes (P_n) et où vous étiez il y a un instant ($P_{\{n-1\}}$). La direction et la distance entre ces deux points représentent votre élan (inertie).

💡 La Formule Mathématique

La position future ($x_{\{n+1\}}$) est calculée à partir de la position actuelle (x_n) et de la précédente ($x_{\{n-1\}}$) :

$$x_{\{n+1\}} = x_n + (x_n - x_{\{n-1\}}) + a \cdot dt^2$$

Où :

- $(x_n - x_{\{n-1\}})$ représente la vitesse (pseudo-vitesse).
- $a \cdot dt^2$ représente l'accélération (comme la gravité).

L'Algorithme en 3 Étapes

Pour chaque objet dans la simulation, on suit cet ordre strict :

1. Calcul de la Vitesse : `vitesse = pos - old_pos`
2. Mise à jour :
 - On sauve la position actuelle : `temp = pos`
 - On calcule la nouvelle : `pos = pos + vitesse + (accel * dt * dt)`
 - On met à jour l'ancienne : `old_pos = temp`
3. Contraintes : On ajuste `pos` directement (murs, collisions).

Comparaison : Euler vs Verlet. Euler Explicite :

- Avantage : Très simple.
- Inconvénient : « Explode » vite. Si on modifie la position d'un objet sans changer sa vitesse, le moteur devient incohérent.

Verlet :

- Avantage : Inconditionnellement stable. Si on déplace un objet manuellement (contrainte), sa vitesse s'ajuste d'elle-même à la frame suivante.
- Idéal pour : Les tas de billes, les cordes, les tissus et les chevelures.

Gestion des Collisions

En Verlet, on ne calcule pas de forces de réaction complexes. On utilise la Relaxation de Contrainte.

Si une bille pénètre dans un mur :

- On la téléporte à la surface du mur (`pos.y = sol`).
- À la prochaine frame, le calcul `pos - old_pos` donnera une vitesse nulle ou réduite.
- Le système s'auto-équilibre.

$$v_{\text{new}} = \frac{\text{pos}_{\text{corrigée}} - \text{old}_{\text{pos}}}{dt}$$

```
// Intégration
Vector2 velocity = Vector2Subtract(ball.pos, ball.old_pos);
ball.old_pos = ball.pos;
ball.pos = Vector2Add(Vector2Add(ball.pos, velocity),
                      Vector2Scale(ball.accel, dt * dt));

// Contrainte (Collision mur)
if (ball.pos.y > 800) ball.pos.y = 800;
```