

Du Fondement à l'Expérience

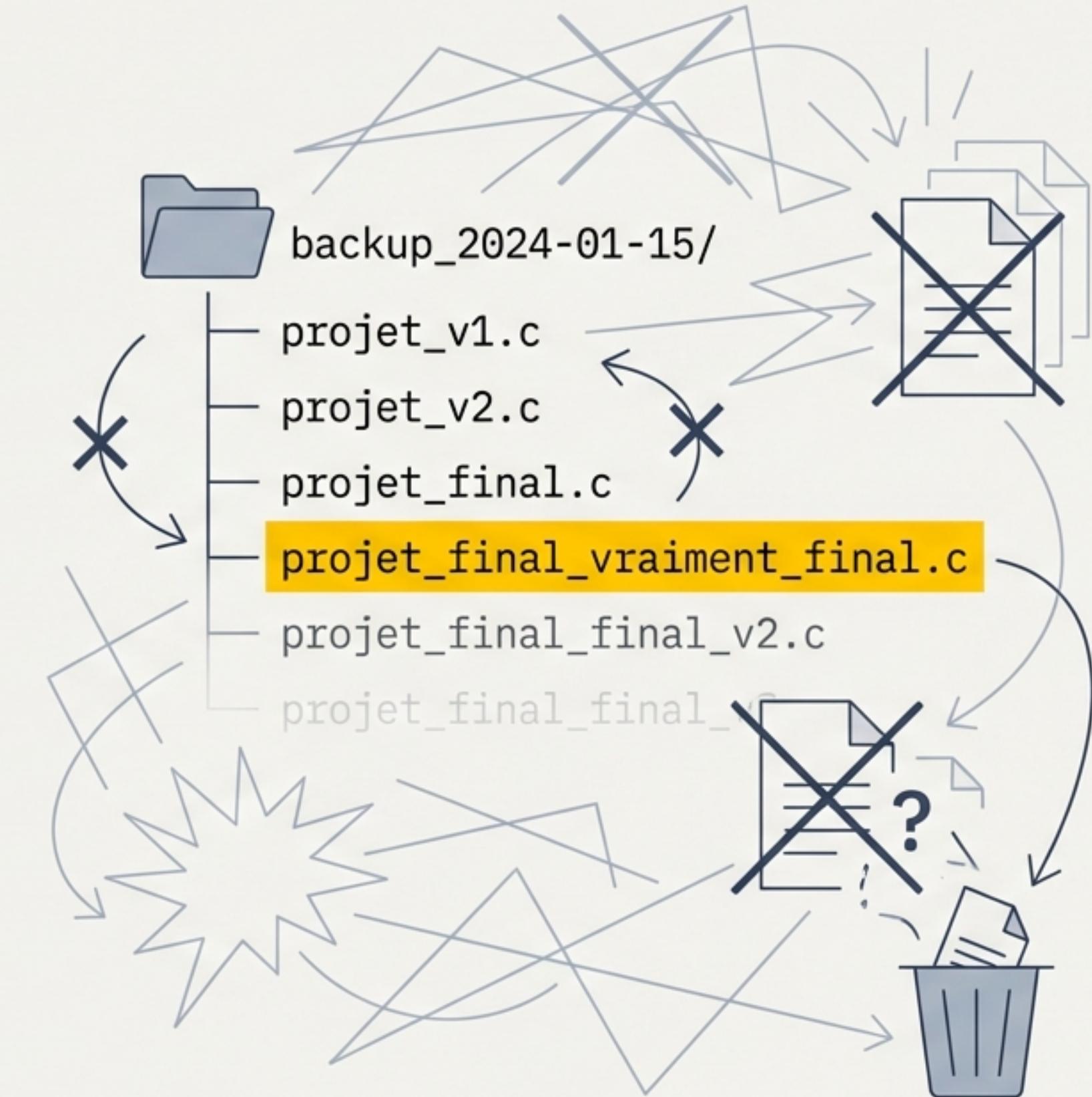
Comment la maîtrise du code (Git) et l'art de l'interaction (Jeu Vidéo) façonnent le développement moderne.



Avant le contrôle, le chaos artisanal.

Avant les VCS, la gestion de code était **manuelle** et sujette à l'erreur. La collaboration était un cauchemar.

- Aucune traçabilité des modifications
- Collaboration quasi impossible
- Perte fréquente de travail
- Conflits constants
- Gaspillage d'espace disque



Trois générations pour maîtriser la collaboration



VCS Locaux (ex: RCS, 1982)

Concept

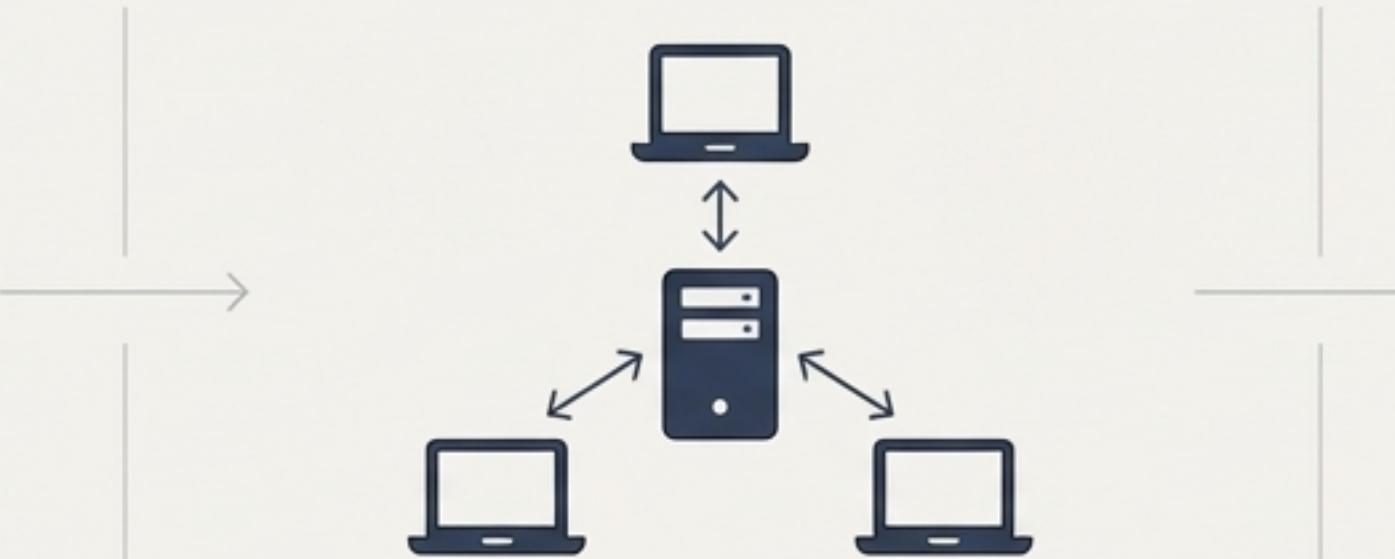
Un utilisateur à la fois, sur une machine.

Innovation

Stockage des deltas, verrouillage de fichiers.

Limite

Pas de collaboration réseau.



VCS Centralisés (ex: SVN, 2000)

Concept

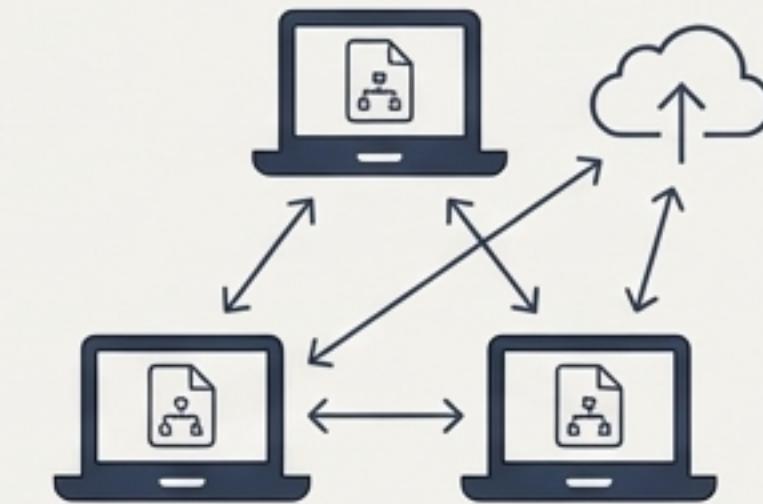
Un serveur central pour toute l'équipe.

Innovation

Travail concurrent, fusion automatique.

Limite

Point unique de défaillance, connexion réseau obligatoire, branches coûteuses.



VCS Distribués (ex: Git, 2005)

Concept

Chaque développeur a une copie complète du dépôt.

Innovation

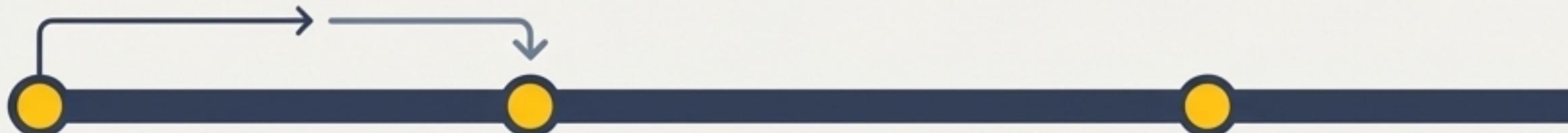
Travail hors ligne, branches légères et instantanées, workflows flexibles.

Limite

2005 : Une crise, une semaine, une révolution.

Contexte

Le noyau Linux, projet open source massif, dépendait de BitKeeper, un outil propriétaire. En avril 2005, la licence gratuite est révoquée.



3 avril : Début du développement

7 avril : Git gère son propre code source

16 juin : Git gère le noyau Linux

Le défi de Linus Torvalds

Créer un remplaçant en urgence, avec des objectifs clairs :

1. **Vitesse** quasi-instantanée A small icon of a clock with a yellow circle on the dial, representing speed.
2. **Design simple** et robuste A small icon of a gear, representing simplicity and robustness.
3. Support natif de milliers de **branches parallèles** A small icon of a USB port, representing parallel branches.
4. Architecture **complètement distribuée** A small icon of a network of nodes connected by lines, representing distributed architecture.

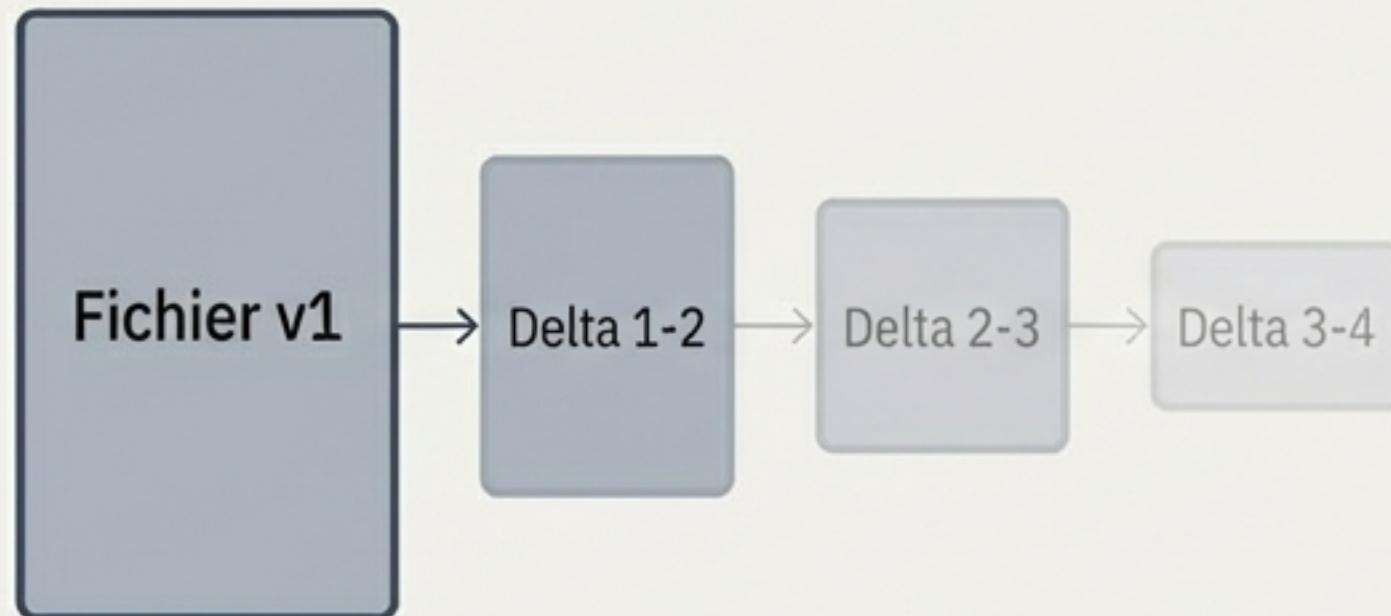
“I’m an egotistical bastard, and I name all my projects after myself. First ‘Linux’, now ‘Git’.”

– Linus Torvalds (Git signifie ‘idiot’ en argot britannique)

Le changement de paradigme : Des différences aux instantanés.

L'approche traditionnelle (Deltas)

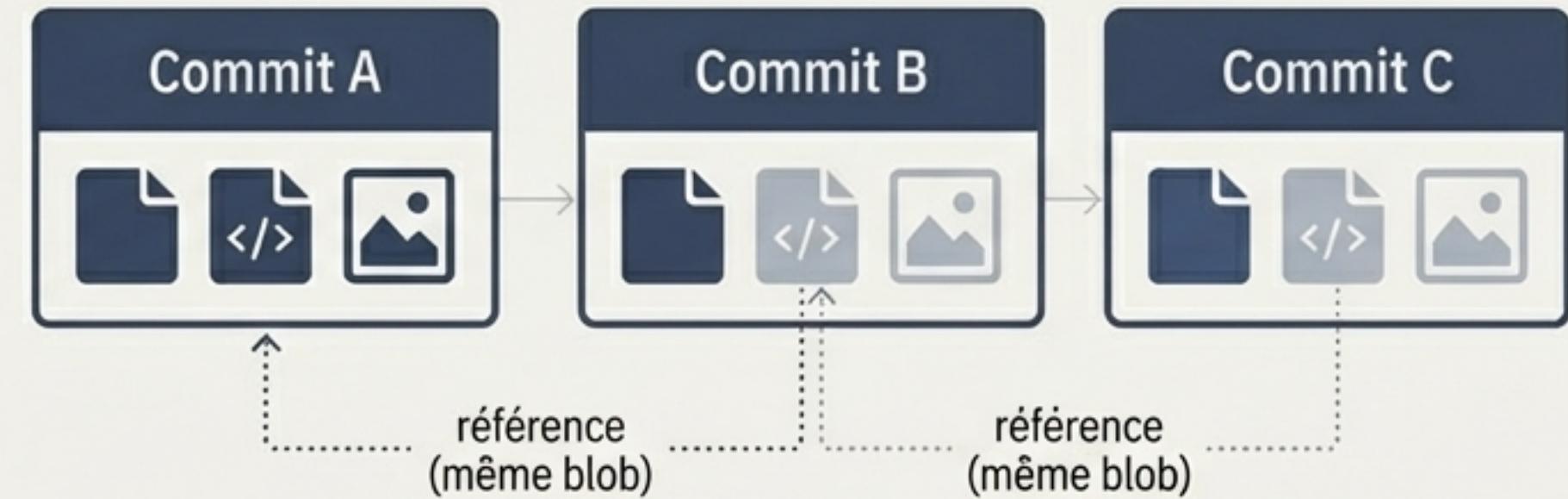
SVN / CVS



Stocke les différences entre les fichiers.
Pour reconstituer une version, il faut appliquer toute la chaîne de deltas, ce qui est lent.

L'approche Git (Sauvegardes)

Git



Git stocke un instantané complet de tous les fichiers à chaque commit. Si un fichier n'a pas changé, Git stocke juste une référence vers la version précédente.

Avantages clés

- Accès instantané à n'importe quelle version
- Branches quasi-gratuites (de simples pointeurs)
- Intégrité des données garantie par le hachage SHA-1

La fondation est stable. Quelle est la nouvelle frontière ?

Maintenant que nous maîtrisons la collaboration sur des systèmes complexes, quel est le prochain grand défi ?

Créer des expériences utilisateur fluides, performantes et unifiées sur n'importe quelle plateforme.

La solution se cachait dans l'industrie du divertissement.

Les défis des interfaces modernes – 60 FPS, gestion d'état complexe, rendu cross-platform – ont été résolus depuis des décennies par l'industrie du jeu vidéo.

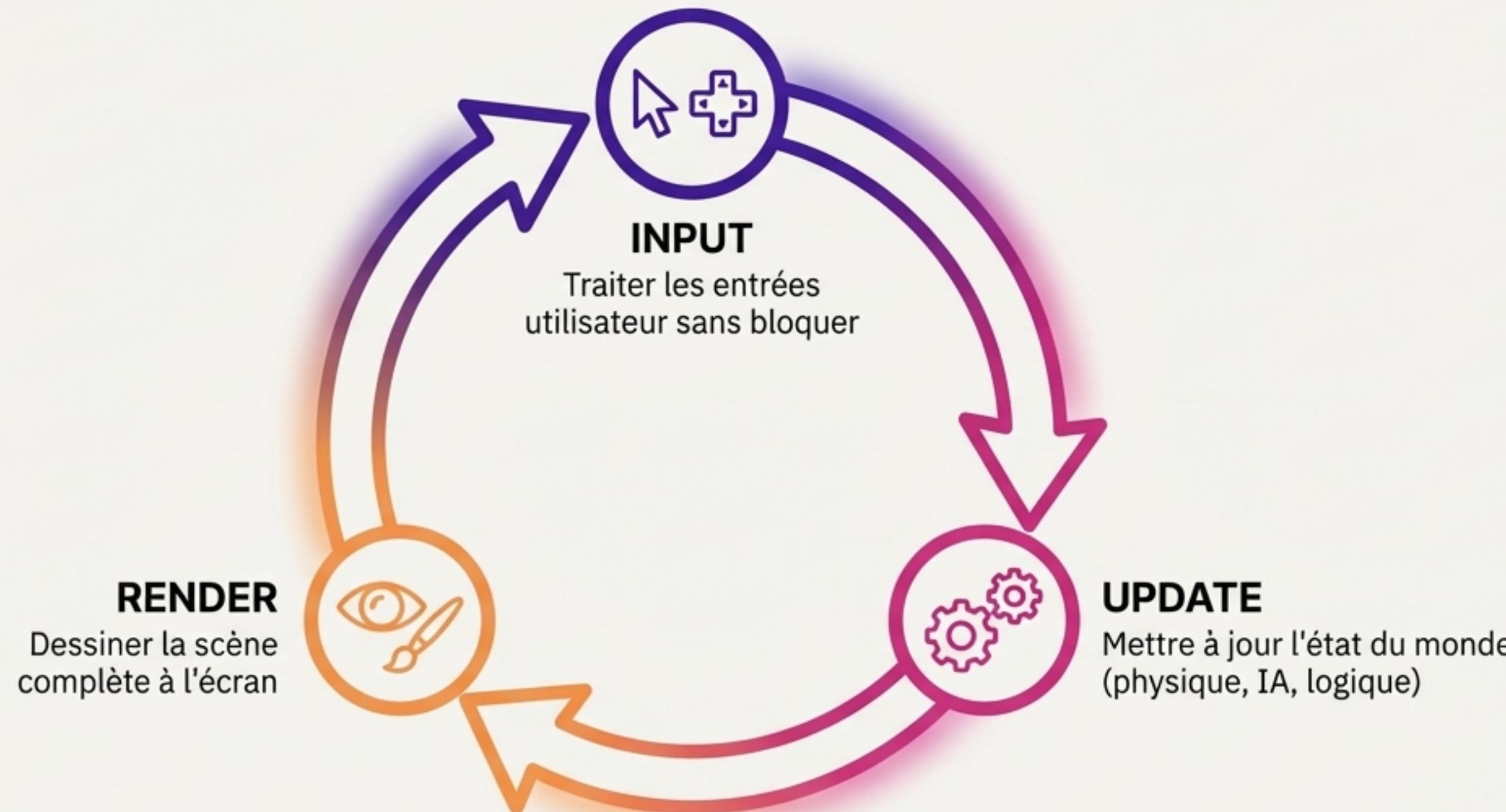
Preuve par les chiffres

- Marché mondial : **~189 milliards \$** en 2025 (prévision >530 Mds \$ en 2030)
- Principaux acteurs : Sony, Tencent, Microsoft Gaming

Métrique	Jeux Vidéo	Applications Métier
Latence	< 16ms par frame	< 200ms acceptable
GPU	Critique	Rarement utilisé
Objectif	Engagement, immersion	Productivité, efficacité

Le battement de cœur de toute expérience interactive : La Game Loop.

Le cœur d'un moteur de jeu n'est pas événementiel, mais une boucle infinie qui s'exécute idéalement 60 fois par seconde (~16.67ms par cycle).



Flutter : un moteur de jeu pour les applications

Flutter a abandonné le modèle des widgets natifs pour une philosophie radicalement différente : il dessine chaque pixel lui-même, exactement comme un moteur de jeu.



Contrôle total du rendu

Grâce à ses moteurs graphiques (Skia/Impeller), Flutter garantit un comportement identique et performant sur toutes les plateformes.



Architecture de composition

L'UI est construite en assemblant des composants (Widgets), une approche héritée des architectures de jeu modernes (Entity-Component-System).



Boucle de rendu continue

Flutter maintient une boucle de rafraîchissement à 60 FPS, similaire à une Game Loop, pour garantir des animations parfaitement fluides.

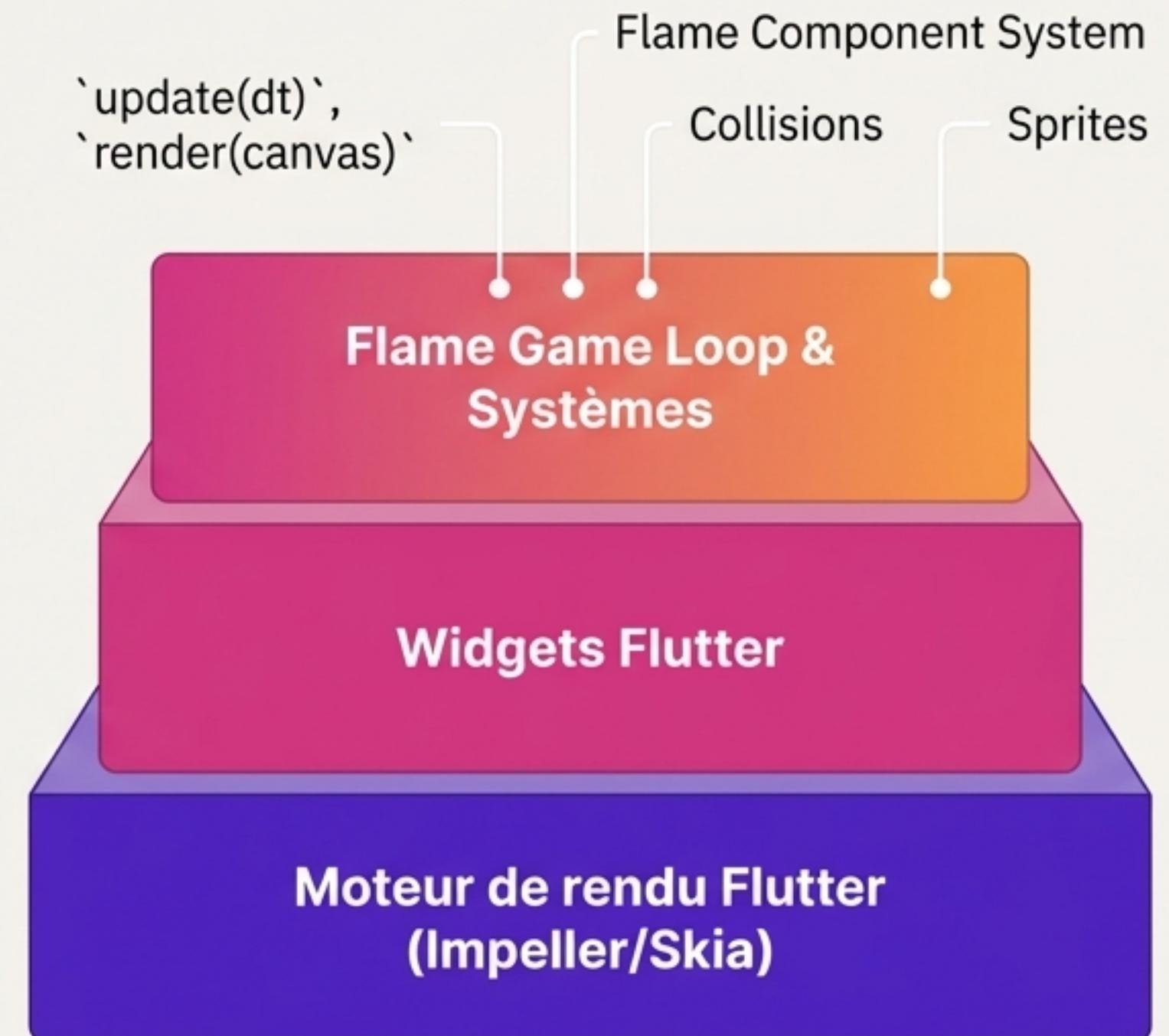
Note : Le ‘Hot Reload’, adoré des développeurs Flutter, est une technique popularisée par les moteurs de jeu pour permettre une itération créative rapide.

Flame : La fusion ultime de l'application et du jeu.

Flame est un moteur de jeu 2D minimalistique construit directement sur Flutter. Il n'essaie pas de réinventer la roue ; il étend la puissance de rendu de Flutter avec les abstractions nécessaires au développement de jeux.

Concept clé

Le Flame Component System (FCS) est une approche de composition inspirée de l'architecture Entity-Component-System (ECS) des jeux, permettant de construire des entités de jeu complexes et modulaires.

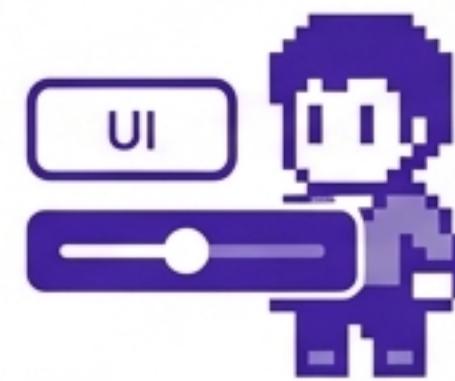


Un outil pour chaque tâche : La place de Flame dans l'écosystème.

Critère	Flame	Unity	Godot
Langage	Dart	C#	GDScript, C#
Éditeur visuel	Non (code-first)	Oui	Oui
Taille du build	Très léger	Moyen	Léger
Écosystème	Packages Flutter (pub.dev)	Asset Store (gigantesque)	Communautaire (croissant)
Courbe d'apprentissage	Facile (si Flutter connu)	Moyenne	Facile
Cas d'usage idéal	Apps gamifiées, jeux 2D avec UI complexe	Projets 2D/3D polyvalents	Jeux indie 2D/3D

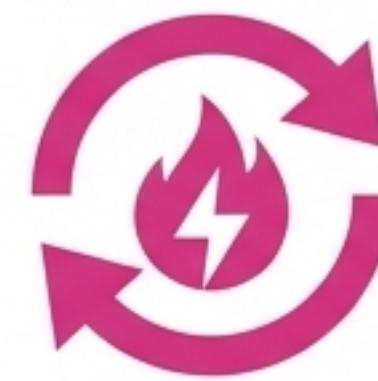
L'avantage Flutter : Les super-pouvoirs de Flame.

Développer avec Flame, c'est développer un jeu avec tous les avantages de l'écosystème d'applications le plus productif.



Intégration UI native

Intégrez de manière transparente des interfaces Flutter complexes (menus, HUDs, formulaires) directement dans votre jeu. Fini les webviews ou les frameworks UI séparés.



Hot Reload pour les jeux

Modifiez la logique, les animations ou les effets de votre jeu et voyez les résultats instantanément, sans recompiler. Une vitesse d'itération inégalée.



Accès à l'écosystème de packages

Intégrez Firebase pour le backend, Rive pour les animations, des solutions de paiement... tout l'univers de `pub.dev` est à votre disposition.



Déploiement unifié

Une seule base de code pour créer un jeu qui fonctionne nativement sur iOS, Android, Web, Windows, macOS et Linux.

Le bon outil pour le bon projet.



Quand utiliser Flame

- ✓ **Jeux casual 2D** : Puzzles, jeux de cartes, endless runners.
- ✓ **Applications gamifiées** : Intégrer des mini-jeux ou des mécaniques ludiques dans une app métier.
- ✓ **Prototypage rapide** : Tester une idée de gameplay en quelques heures grâce au Hot Reload.
- ✓ **Projets avec une UI riche** : Quand l'interface est aussi importante que le jeu lui-même.



Quand préférer un moteur traditionnel

- ✗ **Jeux 3D complexes** : Flame est principalement orienté 2D.
- ✗ **Projets AAA** : Manque d'outils avancés (cinématiques, shaders visuels...).
- ✗ **Équipes avec des designers non-codeurs** : L'absence d'éditeur visuel est un frein.
- ✗ **Besoin d'un Asset Store massif** : Pour les projets qui reposent fortement sur des assets pré-faits.

Recommandations :

Unity (polyvalence), Godot (open source), Unreal (graphismes de pointe).

Le développeur moderne : Architecte de Systèmes & Metteur en Scène d'Expériences



Les outils que nous utilisons façonnent notre pensée. En maîtrisant les fondations de la collaboration et en adoptant les principes de l'interaction temps réel, nous ne sommes plus seulement des constructeurs de logique, mais des créateurs d'expériences.

L'innovation se trouve aux frontières.

Git est né de la nécessité de gérer des systèmes distribués. Les interfaces modernes s'inspirent de décennies de recherche en divertissement interactif.

Quelle sera la prochaine discipline qui transformera notre métier ? La physique des particules ? La biologie synthétique ? L'architecture urbaine ?

Explorez.