# Stickman Brawl — Post-mortem

> Projet individuel — Automne 2025
> Auteur : Antonin Claudel

---

## 1) Informations générales

**Titre du projet :** *Stickman Brawl*
**Nature du projet :** jeu de combat local (2 joueurs) de type *Smash-like*.
**Moteur :** Unreal Engine 5.6
**Technologies :** intégralement développées en **Blueprints** (C++ seulement exploré, puis écarté).
**Période de réalisation :** du **8 septembre 2025** (prototype Zero-G) au **26 octobre 2025**.

### Évolution du concept

Le projet est né d'une première expérimentation, *Stickman Zero G*, centrée sur la simulation physique et la gravité nulle. Cette orientation, rapidement abandonnée pour des raisons de faisabilité et de rendement pédagogique, a conduit à un pivot conceptuel vers un *Smash-like* exploitant pleinement les systèmes internes du moteur (notamment `LaunchCharacter`, `ApplyDamage`, `Jump` et `AddMovementInput`).
Les principales décisions de cadrage ont consisté à :

- Substituer la **caméra dynamique** par une **caméra latérale fixe**, réduisant la complexité d'intégration.
- Redéfinir la condition de *KO* : abandon du modèle d'éjection au profit d'une élimination à **santé nulle**, formalisée dans les versions V2 et V2.1 du One-Pager.
- Restreindre le périmètre à un **multijoueur local à 2 joueurs fixes**, l'ajout ou le retrait dynamique via C++ (détection des manettes) ayant été jugé non prioritaire dans la contrainte temporelle.

---

## 2) Objectifs initiaux

L'objectif principal consistait à produire, dans un cadre individuel, un **prototype jouable et stable** démontrant une compréhension approfondie du moteur et un réel apprentissage tout en demeurant réalisable dans les délais impartis.
Les intentions initiales portaient sur :

- L'expérimentation des **mécaniques de combat rapproché** ;
- La mise en œuvre d'un **multijoueur local robuste**.

Bien que non planifiée au départ, la **maîtrise du Player State et des Data Assets** est devenue un apprentissage central, structurant la réflexion architecturale finale.

---

## 3) Points de réussite

La rigueur du refactoring a abouti à une architecture cohérente et lisible, entièrement en Blueprint. Les classes UE fondamentales (**GameMode**, **GameState**, **PlayerState**, **PlayerController**, **Character**) ont été utilisées conformément à leur rôle, et le multijoueur 2P fonctionne de manière fiable.

Les fonctionnalités initiales — déplacements latéraux, `LaunchCharacter`, `ApplyDamage`, gestion de la santé et des stocks, détection des vainqueurs — ont été stables dès leur première implémentation.

La gestion de version (Git CLI) a suivi une méthodologie itérative : commits atomiques sur une branche `main`, intégration fréquente, exploitation avancée de **Git Bash**, **Vim**, **PowerShell** et **GH CLI**.

Les choix techniques marquants incluent l'usage des **Maps**, des **Widgets imbriqués**, des **Data Assets**, des **Event Dispatchers**, des **Anim Notifies** et des **Sockets**.

Les ajustements de gameplay post-tests — notamment l'augmentation de l'**Air Control**, la correction du *lag* post-attaque et la priorisation du tick Player Controller pour la rotation — ont renforcé la fluidité et la lisibilité du combat.

---

## 4) Problèmes rencontrés et résolutions

Les difficultés majeures ont été liées à l'ordre d'initialisation des acteurs (widgets, Player State/Controller, ID local −1) et à la nécessité d'ajuster les délais d'exécution. Le remplacement d'une macro du GameMode par une fonction a permis les appels par référence depuis un WBP.

Le module de sélection des personnages, véritable terrain d'apprentissage, s'est étendu sur une période d'environ vingt jours. Les importations d'assets ont été stabilisées grâce à la normalisation des Offsets et à l'usage du modèle « Without Skin ».

La piste Zero-G, centrée sur la simulation physique du ragdoll, s'est révélée non viable dans les délais : le pivot vers le Smash-like a permis de tirer parti de la physique moteur sans surcharge algorithmique.

Les erreurs conceptuelles identifiées concernent principalement :

- La complexité inutile de l'interface : l'utilisation de **CommonUI** aurait pu simplifier la gestion des interactions ;
- La logique de rotation « face à l'attaquant », encore tributaire du Tick ;
- Quelques collisions capsules restant à optimiser.

Aucune contrainte matérielle n'a limité le développement.

---

## 5) Compétences acquises

- **Connaissances techniques approfondies :** maîtrise des systèmes de mouvement (Air Control, Jump), des traces multi-sphériques, de `ApplyDamage/AnyDamage`, et de `LaunchCharacter`.
- **Gestion de l'état du jeu :** appropriation complète de Player State, Game State et Player Controller, et compréhension de leur cycle de vie et d'interdépendance.
- **Conception de données :** usage pertinent de Maps et Data Assets ; structuration claire du modèle de données.
- **Interface utilisateur :** implémentation d'UI réactives via Event Dispatchers et hiérarchies de Widgets.
- **Méthodologie logicielle :** refactorisation itérative, réduction de la dette technique et appropriation des abstractions moteur existantes plutôt que réimplémentation.

### Compétences transversales

La principale compétence non technique consolidée demeure la **gestion du temps et des priorités**, en adéquation avec le périmètre du One-Pager. Le renoncement réfléchi à certaines fonctionnalités ambitieuses (connexion/déconnexion dynamique) illustre une approche mature de la gestion de projet.

---

## 6) Perspectives d'amélioration

Sur le plan du gameplay, plusieurs axes de recherche subsistent :

- Intégration d'actions avancées (saisie, esquive, dash, bouclier, ledge grab, attaques directionnelles, marche différenciée) ;
- Mise en œuvre d'un **knockback progressif** et d'un système d'éjection ;
- Ajout d'un **hitstop** et d'une meilleure décorrélation entre la vélocité entrante et le launch.

Une révision de la caméra (dynamique), l'ajout de plateformes, l'enrichissement visuel des arènes et une meilleure gestion du focus après *alt-tab* figurent parmi les améliorations envisagées.
À plus long terme, des travaux sur la rotation déterministe et la refonte des collisions sont souhaitables.

En cas de prolongement de deux mois supplémentaires, la priorité serait donnée à l'optimisation du cœur du gameplay, du retour visuel et de la stabilité de l'interface.
La publication sur itch.io est envisagée à titre de vitrine technique. Les projets futurs pourraient porter sur l'**internationalisation** et la **mise en réseau**, notamment dans le cadre d'un prototype FPS.

---

## 7) Bilan

L'expérience a confirmé l'importance de l'ajustement dynamique du périmètre : les objectifs initiaux, bien qu'ajustés à la baisse, ont été atteints avec satisfaction. Cette démarche illustre une compréhension pratique du *scope management* et de l'adaptabilité agile.

Ce travail a consolidé une approche d'ingénierie : compréhension fine des mécanismes internes du moteur, anticipation des dépendances inter-systèmes et articulation équilibrée entre ambition technique et viabilité temporelle.

Le projet totalise environ **77 à 100 heures de travail effectif**, soit une moyenne de **11 à 14 heures par semaine sur 7 semaines**. Cette estimation, dérivée de l'analyse des commits et pondérée selon leur ampleur, reflète fidèlement l'intensité du développement.

**Synthèse personnelle :** la réalisation intégrale du projet, sans collaboration directe, a renforcé mon autonomie méthodologique et mon assurance technique.
**Maxime directrice :** *Keep it simple, stupid* — principe dont la rigueur d'application s'avère essentielle dans les productions interactives à contrainte forte.