MiniVibes

"Premium Is Scarce. Skill Is Not." (Claude) - v1.1

Introduction

L'évolution récente (c'est-à-dire des dernières *semaines*, pas années !) des agents IA et du *vibe coding* sont un **bouleversement sans précédent dans l'histoire du code**.

Nous entrons dans une ère où savoir *vibe coder* va devenir aussi important que savoir coder. Aucune formation n'existe encore, c'est un continent inconnu. À vous de le défricher. Ce projet en est une occasion.

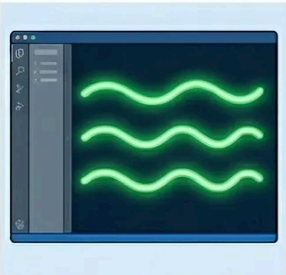

Figure 1

Figure 2

Figure 3

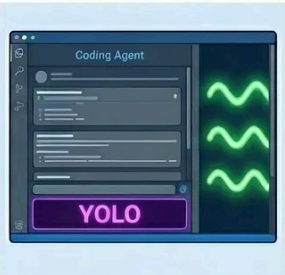

Figure 4

Figure 5

Figure 6

Figure 7

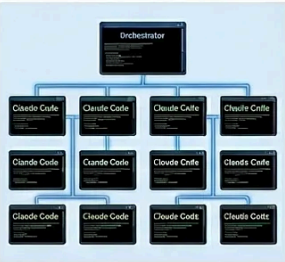

Figure 8

Le projet

Vous avez déjà utilisé un agent IA pour **coder depuis votre terminal**. Si vous savez maîtriser un tel agent, bravo : vous voilà maintenant au stage 5 dans l'échelle du dev (ou "Developer-Agent Evolution Model", cf. image au-dessus) !

Le challenge sera de poursuivre cette lancée et, avec des ressources limitées, de produire les projets les plus aboutis possibles :

- **limitées en temps**, car chaque projet vous prendra une journée seulement,
- **limitées en tokens**, car vous serez toujours sur Copilot, avec un output de tokens limité,
- **limitées en intelligence**, car vous n'aurez pas le luxe d'utiliser des modèles "lourds".

Votre objectif est de parvenir à reproduire des projets de plus en plus ambitieux, **en n'utilisant qu'une requête "premium" par projet/jour**. Le reste du temps, vous utiliserez un modèle "mini", avec un multiplicateur de 0.33x (GPT-5.1-Codex-Mini, Claude Haiku 4.5, Gemini 3 Flash…).

En fin de journée, un comparatif des meilleurs projets sera présenté. À chaque fin de journée, un **projet du jour** sera élu par les équipes comme étant le plus abouti.

J01 - ne lâchez pas les chevaux

Le projet du premier jour sera de reproduire le jeu https://enclose.horse.

Vous pouvez utiliser la stack de votre choix. Le site doit pouvoir tourner sur un navigateur Web. **Soyez très attentifs à toutes les fonctionnalités proposées**, elles ne sont pas toutes évidentes ! Passez déjà du temps à bien l'observer, comprendre son fonctionnement et ses finesses (graphiques, autres).

Une fois le site reproduit, vous êtes invités, bien sûr, à l'améliorer.

Les autres projets vous seront communiqués à chaque début de journée.

Contraintes

Vous fonctionnerez en deux temps : d'abord, vous ferez rédiger une **spécification complète** par votre agent. Vous devrez **vérifier la cohérence et la faisabilité** de cette spécification avant de la valider. Une fois validée, cette spécification constituera le **contexte principal** pour l'agent tout au long de la production du projet.

Éléments attendus dans la spécification :

1. **Découpage fonctionnel et technique** (front-end/back-end, modules ou composants principaux : UI, moteur de jeu, API, stockage…)
2. **Stack technique** (langages/frameworks/libs, DB et schéma simplifié, gestionnaire de packages, outils de build, de mise en forme, de test..)
3. **Contraintes et ressources** (limite des requêtes premium / mini utilisées par jour, éventuellement restrictions de performance…)
4. **Règles de rendu et de fidélité** (objectif visuel ou comportemental précis à atteindre, critères de succès minimal pour considérer un module fonctionnel, tests/vérifications automatisables
5. **Organisation du projet** (structure des fichiers et conventions de nommage)
6. **Stratégie de développement** (étapes à générer en premier, étapes de validation, gestion du contexte)

Voici les contraintes :

- Vous devez avoir un fichier COPILOT.md dans votre projet
- Vous devez pouvoir montrer tous vos prompts
- Aucun code ne doit provenir d'ailleurs que de votre agent
- Vous devez versionner intelligemment votre code avec Git
- Idéalement, votre projet est dockerisé (mais ça n'est pas une obligation absolue, notamment si vous ne faites pas de Web)
- À la fin de votre session, vous devez créer un fichier SESSION.md contenant le résultat de la commande /session

Bonus :

- Vous utilisez plusieurs agents, avec des Git worktrees
- Votre projet a des tests unitaires
- Votre projet est non-injectable et non-attaquable trivialement (injection SQL, XSS, input validation…)
- Votre code est formaté correctement et de façon homogène sur tout le projet. Vous pouvez montrer qu'un système de formatage est en place
- Vous utilisez une base de données quand cela est pertinent (relationnelle, NoSQL…)

Compétences visées

- Vibe coding
- Prompting

Rendu

Le projet est à rendre sur votre github :

https://github.com/prenom-nom/minivibes

Base de connaissances

- https://justin.abrah.ms/blog/2026-01-08-yegge-s-developer-agent-evolution-model.html

- https://x.com/bcherny/status/2015979257038831967
- https://git-scm.com/docs/git-worktree