

Journal : C'est historique, le développement logiciel est entièrement automatisable

Posté par Andréas Livet le 14 octobre 2026 à 18:18. Licence CC BY-SA.

Étiquettes : grand, remplacement, libre, adobe, vibe, coding



Sommaire

- [Ces applications ne sont pas "mal codées"](#)
- [Ces applications ne sont pas issue d'un travail de décompilation](#)
- [Ces applications ne sont pas non plus des copies des logiciels libres existants](#)
- [Ces applications sont fonctionnelles](#)
- [Ces applications sont libres et multiplateformes](#)
- [Ces applications sont codées en Rust](#)
- [Ces applications sont codées de manière \(quasi\) automatisée](#)
- [Nous sommes comme les mathématiciens](#)
- [Comment cela est-ce possible ?](#)
- [La naissance d'une nouvelle forme de logiciel libre](#)
- [La fin du code propriétaire et la fin du logiciel libre tel que nous le connaissons ?](#)
- [La fin d'un métier](#)
- [Conclusion](#)

Depuis le début de la semaine je suis sous le choc suite à [ce tweet](#) de Brandon Thomas qui date du 03 octobre 2026 :

I've vibe coded replacements to 7 of Adobe's apps in Rust. They're open source (links below)

On peut retrouver toutes ces applications (il y en a maintenant 13) [sur le site d'ArtCraft](#).

J'aimerais développer en quoi je trouve cet instant historique pour tout le secteur du développement logiciel qu'il soit propriétaire ou libre.

Ces applications ne sont pas "mal codées"

Note : Je base mon analyse sur PhotoCraft, le clone de Photoshop, mais à ce que j'ai vu tous les projets suivent peu ou prou la même architecture.

A priori on pourrait se dire : "bah c'est encore un énième gars qui a vibe codé un truc tout pourri comme tout ce qui est vibe code", mais je crois qu'on est en train de voir la naissance d'une toute nouvelle manière de faire des logiciels : une manière entièrement automatisée.

A regarder [tous ces dépôts](#) d'un peu plus près, les projets ont l'air cohérents, bien organisés, bien écrit (je n'ai pas trop d'expérience en Rust mais au moins ça n'a pas l'air d'être d'un code totalement immaintenable) et, au moins en parti, testé.

Il y a tout un set de [librairies sous-jacentes](#) qui sont partagées entre les applications.

Les applications ont l'air bien [architecturées](#), avec notamment une séparation très claire entre le cœur des fonctionnalités et l'interface utilisateur, ce qui est une très bonne pratique.

L'[historique des commits](#) est propre, on ne voit pas des commits genre "dev 1", "fix truc" etc. et ça ne donne pas l'impression que les nouvelles fonctionnalités rajoutent du chaos comme on pourrait se l'imaginer sur un projet de cette taille développé aussi rapidement.

Les fonctions sont bien nommées, pas trop grosses, pas plein de commentaires inutiles (comme on voit souvent avec le code fait par LLM), bien découpées en fichiers eux mêmes pas trop gros, mis dans [des dossiers qui font sens](#).

Bref, je ne suis pas expert Rust, je n'ai pas passé des heures à scruter les différents dépôts, mais ce que je peux dire c'est que ça ne *sont* pas le truc coder à l'arrache juste pour que "ça marche".

Voilà pour la première objection que l'on pourrait faire à ces projets

Ces applications ne sont pas issue d'un travail de décompilation

Beaucoup de personnes pensent que cette suite d'application est le fruit d'une décompilation des produits Adobe (et maintenant [Microsoft](#), [AutoDesk](#) et [Figma](#)) et d'une simple *retranscription* en Rust.

Il est très clair que ce n'est pas du tout le cas : les choix architecturaux et de librairies sont sans aucun doute extrêmement différents de comment ont été codés ces différents logiciels. De plus, comme je l'ai dit ils partagent tous le même socle là où une décompilation/transcription porterai sa propre architecture et ses propres librairies pour chaque projet.

Autre "indice" si je puis dire, le fichier [AGENTS.md](#) (sans doute la pièce maîtresse de ce projet vu que tout a été fait par un agent de codage) précise bien que les fonctionnalités codées [ne doivent pas être issues d'un processus de décompilation](#) (Point "3. Clean Room").

Donc il est très peu probable que ce projet soit juste une sorte de clone illégale de la suite Adobe et autres.

Ces applications ne sont pas non plus des copies des logiciels libres existants

J'ai lu a plusieurs endroits que vu qu'il existait beaucoup de logiciels libres comparables à ces logiciels (Gimp, Krita, LibreOffice, Inkscape etc.), il était facile pour une IA de les refaire en se basant sur ce qu'elle a appris.

Mais, je crois que ce raisonnement est tout aussi erroné. Il s'agit réellement d'une nouvelle contribution. Alors, bien sûr certains algorithmes doivent être repris (peut-être entièrement) d'implémentations existantes, mais qui ferait autrement ?

Honnêtement, vous devez recoder une librairie qui fait du dessin vectoriel, vous allez réinventer les algo efficaces et autre ?

Bon, en fait vous allez sûrement utiliser une librairie optimisée qui le fait déjà, car c'est la chose la plus naturelle à faire.

Mais par exemple Google ont bien recoder leur [propre librairie de rendu 2D](#) pour faire flutter, Chrome, etc.

Ici, j'ai l'impression (je n'ai pas encore vérifié) que beaucoup de choses assez bas niveau sont recodées : codecs, rasterisation etc.

C'est visiblement une volonté des auteurs qui précisent par exemple que [FFmpeg n'est pas utilisé](#) pour leur logiciel d'édition vidéo et que leur implémentation se base uniquement sur les spécifications publiques (et bien sûr les données d'entraînement des LLM...).

Il faudrait comparer sérieusement les implémentations de FFMpeg et celles de FilmCraft pour trancher, mais quand bien même les grands principes de ces librairies seraient repris, l'agencement de tous les composants est inédit. Rien que le fait que ces logiciels libres sont souvent écrit en C ou C++ avec des librairies différent fait que le résultat est très différent.

Donc, même s'il y a forcément énormément d'inspiration venant des autres logiciels libres, la suite Artcraft n'en est pas une pâle copie, mais bien quelque chose de nouveau et *original* au moins dans le langage et les librairies sous-jacentes utilisées.

Ces applications sont fonctionnelles

C'est la plus grosse critique qu'on peut faire à ces applications : ça à l'air de marcher, mais en fait c'est du flan, une sorte de chateau de carte qui s'effondre dès que l'on touche quelque chose.

Et bien, peut-être qu'aujourd'hui elles sont encore un peu bugguées, peut-être que certaines fonctionnalités ne fonctionnent pas comme elles devraient, mais je suis prêt à parier que d'ici quelques semaines, tout le monde sera d'accord pour dire que ces applications fonctionnent très bien et sont sans doute meilleures que leur alternatives libres ou propriétaires.

Je n'ai jamais utilisé la suite Adobe, j'ai vu quelques tests et discuté avec des pros qui ont testé et qui sont déjà très étonné du résultat, même si bien sûr ils constatent quelques fonctionnalités manquantes, quelques bugs etc.

Des milliers de personnes sont déjà en train de s'emparer de ces outils et réalisent des choses avec. Si c'était juste une sorte de *façade* ou de coquille vide, ça se verrait tout de suite.

Ces applications sont libres et multiplateformes

Un point important est le côté multi plateforme de cette suite : dès le premier jour il y avait des builds Windows, Mac, Linux, FreeBSD mais aussi Web (via WebAssembly) et des ports vers Android et iOS sont en cours. Bref, du rarement vu pour des logiciels propriétaires et pas toujours le cas pour des logiciels libres (je crois me souvenir que le port Windows de Gimp a pas toujours été à la hauteur faute de mainteneur, un exemple parmi d'autres).

Tous les ingrédients sont là pour une adoption massive.

Ces applications sont codées en Rust

On voit tellement souvent pointer ce nom de langage récemment que beaucoup voit cela uniquement comme un argument marketing (coder en Rust = bien).

Mais, il y a tout de même une réalité : les programmes écrit en Rust sont exempt d'une bonne partie des problèmes de mémoire des programmes écrit en C/C++.

De plus, et c'est un élément important : ils bénéficient aussi d'un éco système moderne et riche de plein de librairies efficaces en terme de mémoire et de ressources CPU/GPU.

L'aspect *moderne* n'est pas juste pour dire "c'est récent donc c'est mieux", c'est plutôt lié au fait que ces librairies tirent profit de la parallélisation et de la généralisation des GPU sur nos machines du quotidien.

Aussi, en ce qui concerne les interfaces utilisateurs, les librairies sont encore jeunes mais sont parties sur des bases neuves en terme de conception.

Qt est très complet et a su évoluer mais est tout de même basé sur du C++ maison (ça a peut-être changé), GTK est un librairie C avec tout ce que ça implique en terme de conception etc. Pas forcément bien adapté à Rust et surtout c'est une couche qui ne bénéficie pas de la protection mémoire de Rust.

La suite ArtCraft [utilise](#) la librairie [egui](#) qui offre une accélération matérielle et un côté multiplateforme avec un rendu identique de partout (à ce que j'ai compris). Ce n'est peut-être pas la librairie la plus complète, mais vu ce que les logiciels ArtCraft font en terme d'interface utilisateur, ça ne les a pas trop arrêté.

Ils [ont une librairie](#) codée par dessus egui qui s'appelle [craft-ui](#) et qui fait partie des [librairies partagées](#) entre les applications. C'est un ensemble de composants réutilisables à ce que j'ai compris.

Ces applications sont codées de manière (quasi) automatisée

Maintenant que l'on sait tout ça, l'aspect le plus *choquant* dans cette histoire est que toute cette activité de génie logiciel, de construction de briques fondamentales, puis d'applications extrêmement variées et complexes, s'est faite en quelques semaines ! Oui pas quelques mois, pas quelques années, quelques semaines (pour certains logiciels [visiblement 2 semaines](#)) !

Et beaucoup des commits sont crédités de echelon (Brandon Thomas, le gars qui a eu l'idée au départ), cela nous indique quoi ?

Il a forcément automatisé la quasi totalité de son flux de travail.

Et je ne parle pas d'avoir automatisé la boucle de analyse=>code=>review=>commit, non ça c'est *so septembre 2026*, non je veux dire qu'il a tout automatisé ! Le découpage en étape, l'analyse de l'existant (comment fonctionne l'interface, mais aussi comment font les autres), les mesures de performance, les tests de non regression, la gestion des PR (quand on fait une demande de fusion de branche dans GitHub), les réorganisations de code nécessaire (refacto) etc.

Je suis persuadé que ce qu'il a fait, un non informaticien aurait quasiment pu le faire. Au sens où, il y a évidemment tellement peu de travail humain face à la quantité astronomique (on parle de plusieurs centaines de milliers de lignes de code pondu en même pas un mois) de code qu'il est évident que l'IA utilisée à jouer le rôle principale dans toute cette affaire.

Attention, je ne veux pas dire que Brandon Thomas n'est pas doué ou qu'il n'a pas fourni des efforts pour en arriver là, juste qu'il y a une telle disproportion entre les efforts fournis et le résultat qu'il est inconcevable qu'un humain soit derrière chaque décision architecturales prisent. Puis il est de toute manière inconcevable que quelqu'un puisse s'y connaître dans l'ensemble des domaines traités par ces logiciels (bon peut-être un de mes anciens collègues - bien le bonjour Daniel au passage - mais c'est une autre histoire).

Donc au mieux, il a donner les grandes lignes (les contraintes si on veut) : codé en Rust, multi plateforme, pas de décompilation, fidélité avec l'interface originale etc. et il a laissé l'IA s'auto organiser. J'exagère à peine...

Nous sommes comme les mathématiciens

Aujourd'hui, je me sens comme les mathématiciens qui voient déferler une montagne de papiers prouvant des théorèmes ou trouvant des solutions à des problèmes vieux de plusieurs siècles pour certains.

A quoi sert mon expérience, ma *petite* maîtrise technique si une IA, même pas forcément dirigée peut produire 100 à 1000 fois plus rapidement que moi des logiciels qui seront d'une plus grande qualité ?

Comment cela est-ce possible ?

J'ai beau utiliser intensivement des LLM et des outils agentiques depuis des mois, je ne voyais toujours pas comment une telle productivité puisse être possible.

Les modèles que j'ai testé sont bons, rapides, mais pas bon à ce point et pas rapides à ce point.

Or, il semblerait qu'il y a eu une sorte de "saut quantique" avec les très récentes versions de Claude et de ChatGPT. Précisément Opus 5.5 ou Fable et GPT Astra.

J'ai arrêté d'utiliser les IA américaines depuis quelques semaines alors je n'avais pas vu ce changement. J'avais bien vu ces vidéos qui vous disent que "tout à basculer", que cette dernière IA est un truc de ouf etc. mais comme ça fait 3 ans qu'on nous dit ça toutes les semaines, je n'y avais pas prêter plus attention que ça.

Sauf, que là où le crois qu'on a atteint un point particulier : le point où le métier d'analyse/développement ou d'ingénieur en développement *est entièrement* automatisable.

Et on voit que c'est visiblement reproductible, car plein d'autres personnes contribuent, très probablement toutes avec une IA (comment suivre humainement le développement de ces logiciels qui n'existaient pas il y a 2 semaines ?), de manière effrénée.

Les commits s'enchaînent à la vitesse de l'éclair sur tous les logiciels de la suite.

Chaque jour, c'est l'équivalent de mois de travail qui sont commits, vraiment je n'en reviens pas.

J'aimerais moi tromper et me rassurer en me disant que ça va se casser la gueule, que c'est un effet de mode, mais que bientôt nous verrons la triste réalité qui est que ces logiciels sont inmaintenables, qu'ils ne font rien comme il faut et qu'il faut tout jeter à la poubelle. Mais, je crois que ça n'arrivera pas.

La naissance d'une nouvelle forme de logiciel libre

Comme je le disais dans [mon commentaire](#) sur l'article parlant de la fin du logiciel propriétaire.

Je crois qu'on est à l'aube d'une nouvelle façon de faire du logiciel libre. Une façon entièrement pilotée par des IA qui va accélérer le développement de solutions d'un facteur 100 ou 1000.

Encore une fois, je fais le pari que si ça continue comme ça (c'est à dire si les IA d'Antropic et d'OpenAI restent relativement abordables pendant quelques mois), on va voir arriver des OS entièrement fait de zéro. Je ne serai pas étonné de voir un vrai clone de MacOS par exemple, mais libre et non verrouillé.

Peut-être même que cet OS pourra être compatible avec les drivers Linux et Android et pourra être porté sur n'importe quelle machine. Cela, au prix de quelques euros et de quelques heures/jours d'interaction avec ces LLMs.

Si vous ne pensez pas que c'est possible, regardez les [dernières expérimentations](#) de Linus Torvalds au sujet du dev assisté par IA sur le noyau Linux.

Vous allez me dire : "mais c'est parce qu'il a la maîtrise, il comprend ce qu'il fait et l'IA ne fait que l'aider". Non, en fait, j'aurai dit la même chose il y a 2 mois peut-être. Mais, là je crois que n'importe qui qui aurait dit à l'IA : "Analyse moi ce patch, optimise le ou trouve des bugs" aurait eu les mêmes résultats.

En gros, ces outils se suffisent à eux-mêmes.

On peut maintenant créer une sorte de boucle, sans humain au milieu, qui dirait en essence : "refait moi Linux mais en mieux et en Rust et je veux que les drivers Linux soient compatibles".

L'humain n'aurait qu'à gérer peut-être qu'une étape dans la boucle : merger le code et relancer une nouvelle itération, peut-être répondre à quelques questions si des choix sont à faire, mais on pourrait entièrement automatiser le truc.

En fait, la taille du contexte gérer par ces LLM est maintenant assez grande pour avoir la vision sur toute une base de code énorme.

Les outils (harnais) tels que Claude Code, Opex, DSH et compagnie sont assez matures pour optimiser l'usage de ces LLMs, les orchestrer, limiter le contexte etc. ce qui les rend beaucoup plus performants et autonomes.

Et si vous pensez qu'un LLM ne peut pas pondre du code *nouveau* au sens original, qu'il ne fait que recraché ce qu'il a vu, c'est que vous n'avez pas suivi ce qu'il se passe dans le domaine.

Une IA du niveau d'Astra ou de Fable est parfaitement capable de lire les spécifications d'un codec et de produire un code qui sera potentiellement plus optimisé que le code écrit par un humain au prix de journées passées à optimiser telle ou telle partie.

Alors peut-être qu'on a pas encore des IA complètement "génies" comme Fabrice Bellard qui fait toute sortes de prouesses sur des programmes extrêmement rapides, petits et performants ? Peut-être que certaines idées vraiment novatrices n'émergeraient pas d'un LLM ? Par exemple, est-ce qu'une IA aurait pu inventer l'[opérateur pipe d'unix](#) ? Ou l'idée bien connue que ["tout est fichier"](#) ?

On ne peut plus rien affirmer, l'histoire récente nous a montré que ces IA étaient capable d'énormément de choses que l'on ne soupçonnait pas.

La fin du code propriétaire et la fin du logiciel libre tel que nous le connaissons ?

Comme [le remarque](#) très judicieusement NicolasP, si ArtCraft marque la fin du logiciel propriétaire et que l'IA aussi sonne le glas du logiciel libre car il n'y aura plus personne pour le financer et que l'IA détruit aussi les communautés autour de ces logiciels. Que reste-t-il ?

Il va rester ces nouveaux types de logiciels. Des sortes de "monstres" qu'on fera évoluer à coup d'abonnement ou de frais d'API.

La nouvelle façon de contribuer à ces nouveaux logiciels sera de payer du temps de computation pour créer ce nouveau type de commun (oui car cela reste du commun selon moi). Ou alors remplir un rapport de bug ou une demande de fonctionnalité qui sera par la suite analysée et implémentée ou refusée par une IA.

Ces outils seront très certainement plus performants, plus portable et plus sécurisés que leurs équivalent libres ou propriétaires.

Ils vont rapidement s'imposer chez les particuliers, peut-être un peu plus tarder chez les entreprises. Tout dépend du nombre de *fautes* et d'attaques que le secteur privé va subir *faute* de ne pas utiliser des systèmes robustes créé entièrement par IA.

Je ne crois pas que ce que je vis d'écrire est de la science fiction, ce que je dis n'est pas basé sur une évolution probable de ces systèmes, mais sur le constat de leur fonctionnement et capacités actuelles.

Vous avez le droit de ne pas souhaiter un tel futur et peut-être qu'il n'arrivera pas, mais ça ne sera pas à cause d'une limitation des IA. Le risque que cela n'arrive pas est selon moi extérieur à la technique elle même, il est plutôt économique et géo politique.

On le sait tous, l'IA est une bulle économique qui risque d'éclater à tout moment, si cela arrive, déjà l'économie réelle va en subir de grave conséquence et ensuite l'accès à ces LLM va sans doute devenir bien plus cher. Ce qui limiterai leur usages à de rare privilégiés. Et donc construire de tels édifices pourrait simplement devenir trop coûteux.

La fin d'un métier

J'ai l'impression d'assister, impuissant, à la fin de mon métier.

Ok, on aura toujours besoin de personnes qui sont plus versées dans la technique et qui veulent "mettre les mains dans le cambouis". Mais en gros, n'importe qu'elle personne qui aime bidouiller pourra coder (ou plutôt assister la production de code) en très peu de temps des logiciels d'une très grande complexité et d'une grande qualité aussi.

Peut-être que si on regardera les détails, certaines choses ne seront pas "bien" pensées au sens ou un

humain n'aurait pas passer autant d'énergie à tester telle fonctionnalité ou que telle donnée n'était pas utile à stocker, ou encore que la technologie utilisée était peut-être pas la plus adaptée (si c'est une personne qui n'y connaît rien qui à fait le demande c'est un cas probable). Mais bon qu'est ce que ça changera ? Combien de projets libre ou propriétaire sont des exemples de clarté, de bons choix architecturaux, de bons choix techniques ? La plupart des logiciels que je connaisse et sur lesquels j'ai travaillé souffraient tous d'une dette technique plus ou moins importante.

Quand tu transmets à un nouveau collègue les connaissances sur une base de code c'est extrêmement fréquent de dire des choses comme : "bon là on a fait comme ça car on avait pas le temps" ou encore "on a commencer le projet avec cette architecture/dépendance/librairie/langage mais c'était une erreur mais c'est trop couteux de l'enlever".

Les projets entièrement pilotés par IA ne seront pas exempts de ces problématiques de conception initiale, juste il sera tellement plus aisé de tout refaire, d'enlever une librairie, de changer de langage que la notion même de dette technique ne sera plus pertinente.

Prenons l'exemple de PhotoCraft, le clone de PhotoShop. Il suffira de lancer de manière hebdomadaire une IA qui analysera l'ensemble du code et proposera des modifications structurelles (refacto) du code pour l'améliorer et éviter la dette technique. Et c'est ce qu'ils font déjà...

Conclusion

Je ne sais vraiment pas quoi en penser. Je suis abasourdi par la vitesse à laquelle vont les choses. L'année dernière je me disais : "l'ia c'est pas mal, ça aide beaucoup mais bon y a encore besoin de relire ce qu'elle produit, elle comprend rien parfois etc.", il y a quelques mois j'ai commencé à utiliser les outils agentiques et j'ai arrêté de coder manuellement, mais j'ai passé énormément d'énergie à tenter de maîtriser ces systèmes, rester en maîtrise malgré la vitesse, faire des revues de code, mettre en place des flux de travail vérifiables etc. Je sentais que mon expertise améliorait de manière significative la qualité de ce que produit l'IA.

Aujourd'hui, cette expertise est entièrement intégrée à ces outils et dans les données d'entraînement de LLM qui se comportent maintenant "comme il faudrait". La boucle est bouclée et je n'ai plus besoin d'en faire partie.

Ça c'est pour ma petite personne. Concernant l'avenir du développement logiciel et du logiciel libre, je crois qu'on est pas prêt et qu'il va se passer des choses que l'on pensait impossibles ou inatignables il y a seulement quelques années et d'une certaine manière c'est enthousiasmant.

(12 commentaires).

Markdown

Epub

Bravo et merci

Posté par piratebab2 le 14 octobre 2026 à 19:09.

Évalué à -3 (+0/-4).

[A] # Re: Bravo et merci

Posté par fork_bomb le 14 octobre 2026 à 19:54.

Évalué à 1 (+0/-0).

Tu n'es pas sur un forum mais un journal.



Désolé d'avoir réagi sur ce point de détail, mais comme je n'ai pas vu d'argument je ne pouvais réagir que sur ce qu'on voyait.

Répondre

Bof

Posté par fearan le 14 octobre 2026 à 19:16. Évalué à 4 (+1/-0).

Dernière modification le 14 octobre 2026 à 19:16.

Tu réagis à chauds sur une activité récente, tu paniques et tu ponds un putain de pavé pour exprimer ton désarroi.



Le développement logiciel c'est 80/20, tu passe 20% du temps à faire 80% de l'appli, et 80% à faire les 20% qui manques.

Des commentaires d'utilisateur de Photoshop on pointé moult détails de ce qui ne marche pas ou mal. Tu survoles le code pour voir que c'est biens structuré; encore heureux que ça l'est ça fait des 20 aines qu'on théorise sur comment architecturer un projet, et le gars qui gère la swarm d'agent s'y connaît un minimum.

typiquement avec <https://github.com/github/spec-kit> tu peux faire des trucs assez propre et assez poussé, mais ça continue de nécessiter une supervision.

Donc mon conseil, avant de paniquer attends de voir la fin du dev ;)

car là il n'en est même pas au 80%.

--

Il ne faut pas décorner les bœufs avant d'avoir semé le vent

Répondre

[A] # Re: Bof

Posté par orfenor le 14 octobre 2026 à 19:28.

Évalué à 2 (+2/-0). Dernière modification le 14 octobre 2026 à 19:30.

Sans vouloir crier au loup, ce qui se passe est quand même assez dingue.

Et vu que

le gars qui gère la swarm d'agent s'y connaît un minimum

même si

il n'en est même pas au 80%.

on peut penser que le projet va progresser.

Et vu qu'il y a beaucoup de PR proposées indépendamment sur Photocraft, le projet fédère, ça devrait donc pas mal progresser sur les 20% les plus durs.

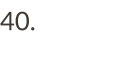
Répondre

Sceptique

Posté par biblicric05 le 14 octobre 2026 à 19:25.

Évalué à 2 (+1/-0). Dernière modification le 14 octobre 2026 à 19:26.

Je suis sceptique quand même...



Un exemple concret : le logiciel [RapidRaw](#) est développé (en Rust)

depuis un peu plus d'un an de façon extrêmement rapide avec de l'IA.

RapidRaw est un peu comme Lightroom d'Adobe (ou DarkTable dans le monde libre) et permet de développer ses photos "brut de capteur".

Je teste régulièrement les nouvelles versions de RapidRaw : au premier abord, c'est bluffant, on a l'impression que c'est aussi bien que Lightroom. Il y a même des fonctionnalités de masquage par IA...

Mais dès qu'on veut s'en servir sérieusement, ça crashe, ça merdouille, c'est la cata. En gros, c'est inutilisable au quotidien...

Développer des logiciels avec l'aide de l'IA pourquoi pas, mais à mon avis il faudra quand même des humains pour guider, tester, valider tout ça...

Pour le moment, une IA n'est pas intelligente, c'est juste une machine qui pioche ce qui est le plus probable pour le problème posé dans l'énorme base de données de code disponibles sur Internet.

Autre exemple : j'ai testé ChatGPT sur des questions un peu pointues de Traitement du Signal. Au début, j'ai été bluffé par la quantité d'équations produites et les résultats fournis étaient justes. Mais quand j'ai essayé de suivre les démonstrations, j'ai vu qu'il prétend utiliser certains théorèmes alors que ce ne sont pas ceux qui correspondent aux équations. Les équations ne se suivent pas vraiment logiquement. Les éléments sont là mais un peu dans le désordre. Il y a une apparence de logique et de raisonnement mais pas plus...

Bref, je pense que le soufflet va retomber assez vite.

Après j'avais aussi prédit en 1992 que le téléphone portable n'avait aucun avenir !

Répondre

Texte généré par IA ?

Posté par Fernando le 14 octobre 2026 à 19:30.

Évalué à 2 (+2/-0).

Je me suis forcé à lire ton texte et cela sent le texte généré par IA à plein nez.

Au final j'ai l'impression que c'est une pub pour l'IA du genre pondue par OpenAI et consorts.

Bref ce n'est pas intéressant.

Répondre

my 0.1 cent

Posté par passant-le le 14 octobre 2026 à 19:40.

Évalué à 2 (+0/-0).

ArtCraft a été forké plus de de 4300 fois. Au final il va y avoir plus de vibecodeurs que de potentiels utilisateurs et plus généralement pourquoi l'utilisateur aurait besoin d'un vibecodeur? Du coup, je prédis que Github va accumuler une quantité monstrueuse de code mort.



Je sentais que mon expertise améliorait de manière significative la qualité de ce que produit l'IA.

T'as quand même conscience que quand (pas si) que les LLM seront en mesure de répondre ton expertise ta valeur ajoutée va passer à zéro.

--

Je trolle dès quand ça parle business, sécurité et sciences sociales

Répondre

[A] # Re: my 0.1 cent

Posté par fork_bomb le 14 octobre 2026 à 19:50.

Évalué à 2 (+1/-0).

Du coup, je prédis que Github va accumuler une quantité monstrueuse de code mort.



Même pas besoin de prédire, GitHub a déjà expliqué ses problèmes de fiabilité par la croissance exponentielle du nombre de commits depuis 2 ans.

Par contre je ne vois pas de croissance exponentielle du nombre de logiciels utiles. Dans tous les domaines où j'utilise des logiciels on utilise peu ou prou la même chose qu'il y a 2 ans.

Répondre

[A] # Re: my 0.1 cent

Posté par Fernando le 14 octobre 2026 à 19:56.

Évalué à 1 (+0/-0).

Et les IA vont s'entraîner dessus



Répondre

Ces applications semble "mal codées"

Posté par GwenLg le 14 octobre 2026 à 19:45. Évalué à 3 (+2/-0).

Bref, je ne suis pas expert Rust, je n'ai pas passé des heures à scruter les différents dépôts, mais ce que je peux dire c'est que ça ne sent pas le truc coder à l'arrache juste pour que "ça marche".

J'ai jeté un oeil deux minutes, la première crate interne que je suis allé voir, c'est `artcraft/crates/lib/errors`, crate minimal qui sert juste à faire des `re-export` (de `anyhow::anyhow` et `anyhow::bail`) et redéfinir des nouveaux noms/types (`AnyhowResult<T> = anyhow::Result<T>` et `AnyhowError = anyhow::Error`) l'intérêt me semble limité, mais pourquoi pas si c'est utilisé partout, ça peut permettre de faire évoluer le type en une fois en cas de besoin.

Ensuite je suis allé voir le fichier `Cargo.toml` de `artcraft/crates/lib/files/images`, et là dans les `dev-dependencies` je vois un `import` de `anyhow` directement

Dans le `Cargo.toml` racine il y a un `import` de `thiserror` qui permet de définir facilement des nouveaux types avec moins de code écrit à la main, mais dans `crates/lib/files/images/src/error.rs` `thiserror` n'est pas utilisé, alors que ça permettrait de ne pas avoir à écrire à la main plus que la définition de l'enum (avec les attributs qui vont bien) et en plus en gérant la `stack d'erreur` ce qui n'est pas le cas ici.

Je n'ai pas envie d'aller regarder plus, mais ça ne me donne pas trop confiance dans la qualité du code.

Répondre

[A] # Re: Ces applications semble "mal codées"

Posté par Fernando le 14 octobre 2026 à 19:59.

Évalué à 1 (+0/-0).

Merci, CQFD.

Répondre

Monsieur c'est un Wendy's ici

Posté par fork_bomb le 14 octobre 2026 à 19:47.

Évalué à 3 (+2/-0).

Quel est l'intérêt de ce journal alors que le même sujet a largement été débattu dans [un journal précédent](#).



posté il n'y a même pas deux jours ?

Je comprends bien que le sujet te tienne à coeur, mais il faut aussi respecter les lecteurs qui manifestent clairement un ras-le-bol sur ce débat.

Répondre

Envoyer un commentaire

Suivre le flux des commentaires

Note : les commentaires appartiennent à celles et ceux qui les ont postés. Nous n'en sommes pas responsables.

Revenir en haut de page

Derniers commentaires

Étiquettes (tags) populaires

- Re: Ces applications...
- Re: my 0.1 cent
- Re: Bravo et merci
- Re: my 0.1 cent
- Monsieur c'est un W...
- Ces applications se...
- Redox et haiku
- my 0.1 cent
- Re: Ivan Illich
- Texte généré par IA ?
- Re: Bof
- Sceptique

Sites amis

- Agenda du Libre
- April
- Éditions D-BookeR
- Éditions Diamond
- Éditions Eyrolles
- Éditions ENI
- En Vente Libre
- Framasoft
- La Quadrature du Net
- Lea-Linux
- Open Source Initiative
- Imprimerie Grafik Plus

À propos de LinuxFr.org

- Mentions légales
- Faire un don
- L'équipe de LinuxFr...
- Informations sur le s...
- Aide / Foire aux que...
- Suivi des suggestion...
- Wiki du site
- Règles de modération
- Statistiques
- API pour le développ...
- Code source du site
- Plan du site