

Vibe Coding

Over de kracht en het gevaar van programmeren met AI

Willem L. Middelkoop

15 apr. 2025



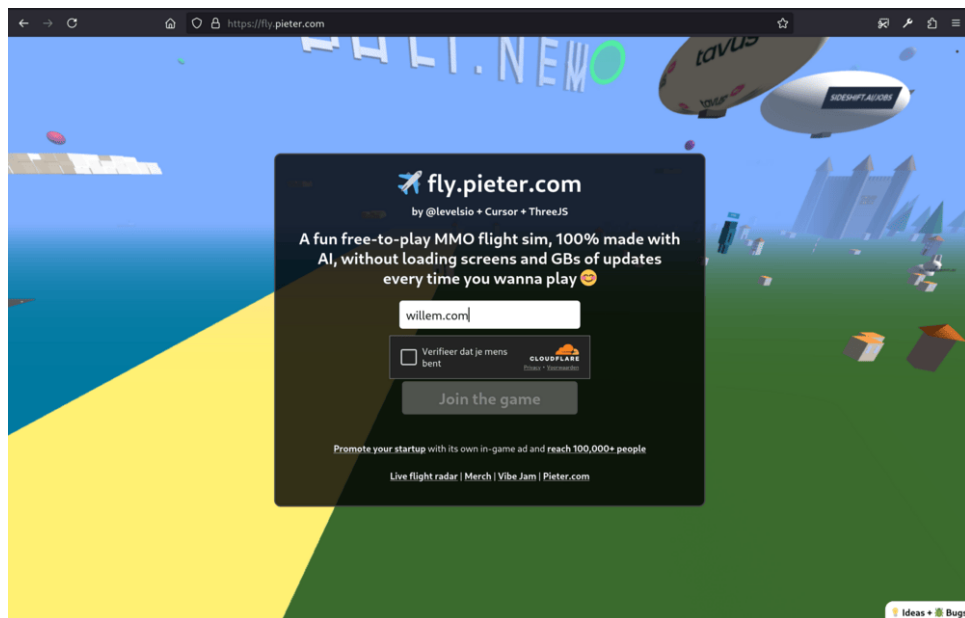
Met een omzet van meer dan \$48.000 per maand, veroorzaakte de vlucht-simulator game van Pieter Levels een hele vlag van innovatie. Zonder gedetailleerde kennis van 3D game engine technologie, 'vibe codeerde' hij zijn game met behulp van AI. Critici van zijn werk wezen op beveiligings- en schaalbaarheidsproblemen, terwijl voorstanders het verbluffende resultaat prezen. Wat kunnen we hiervan leren?

Vibe Coding

Vibe coding is een AI-gedreven programmeermethode waarbij je een probleem in een paar zinnen beschrijft als een prompt voor een taalmodel dat is afgestemd op coderen. Het model genereert software, waardoor de rol van de programmeur verschuift naar het begeleiden, testen en verfijnen van de output. Voorstanders beweren dat het beginnende

codeurs in staat stelt programma's te maken zonder uitgebreide training of software engineering vaardigheden.

Pieter Levels creëerde de game fly.pieter.com. Op sociale media hield hij een open loop van zijn inspanningen bij: van zijn eerste idee tot aan de publicatie. Hij trok sponsors aan voor in-game advertenties en hij verkoopt premium upgrades (zoals speciale vliegtuigen) in zijn game, waarmee hij duizenden dollars per maand verdient.

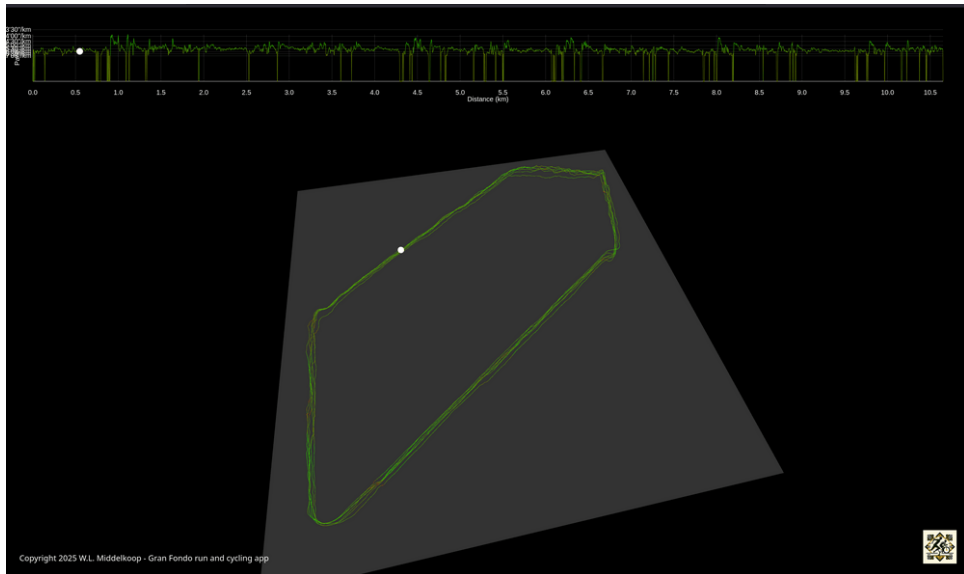


AI gegenereerde Vluchtsimulator: fly.pieter.com

Zijn succesverhaal riep vragen op over de toekomst van gameprogrammering, of zelfs programmeren in het algemeen. Als één persoon met een laptop een smash hit als deze kan maken, wat is dan de noodzaak van big budget studio's? Of, omgekeerd gevraagd, wat zou er gebeuren met big budget studio's als ze een vergelijkbare 'vibe coding' benadering van productie zouden aannemen?

Het zelf proberen

Met enige ervaring in het ontwikkelen van [een game](#) zelf (Snake 97, 40+ miljoen downloads), inspireerde Pieter's verhaal me om te kijken naar de 3D-technologie die hij gebruikte voor zijn game: [ThreeJS](#). Het bouwen van een game is moeilijk, het in 3D doen is nog moeilijker. De wiskunde die een wereld in drie dimensies ondersteunt, voegt een hele extra uitdaging toe. In veel opzichten is het anders dan gewone programma's of eenvoudige tweedimensionale games. Hoewel ik eerder wat heb geëxperimenteerd, vond ik de leercurve van 3D-wiskunde vrij steil. Dit is waar de AI kan helpen.



3D visualisatie van trainingsgegevens

Voor de [Gran Fondo app](#) gebruikte ik de AI om een [driedimensionale presentatie van een opgenomen loopafstand](#) te genereren. Ik formuleerde mijn doel in gewoon Engels en het AI-model genereerde de code die nodig was om ThreeJS te gebruiken om mijn workout op een roterend vlak weer te geven. De eerste resultaten waren niet goed, maar ik begreep genoeg van de gegenereerde resultaten om te blijven vragen naar specifieke verfijningen. Een paar uur later had ik iets wat ik leuk vond, maar bovendien had ik het gevoel dat ik veel had geleerd over de haalbaarheid van mijn oorspronkelijke concept. De AI stelde me in staat om sneller te leren.

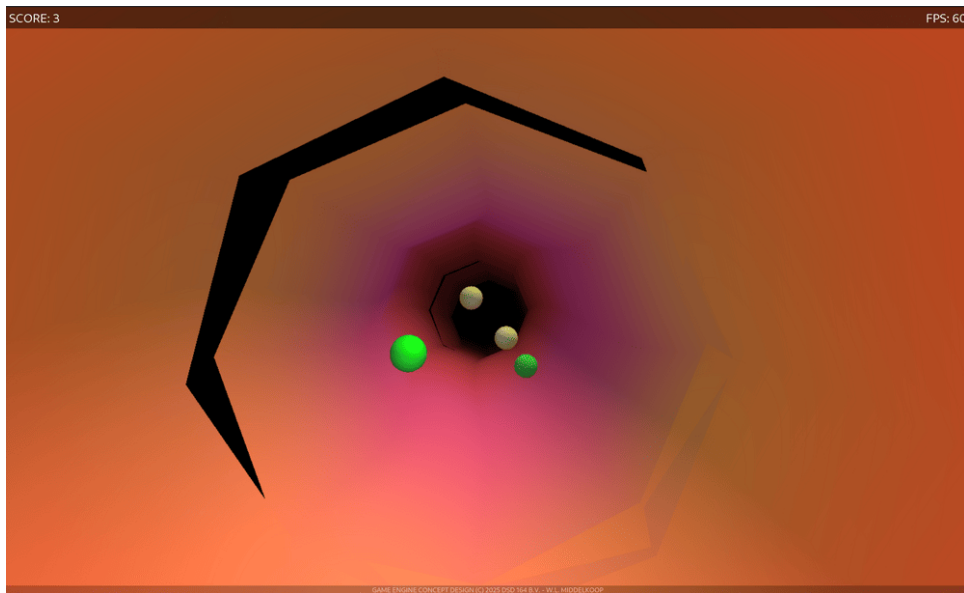


Game Concept artwork uit 2013: HDL deeltjes in de bloedbaan

Ik besloot om een ander idee te verkennen dat ik al een tijdje had geparkeerd: een game over slechte en goede cholesteroldeeltjes in de bloedbaan van een persoon. Met aspecten

van een [endless runner-style game](#), dacht ik dat dit een goede match zou zijn voor wat meer experimenten, aangezien deze games computergegenereerde levels bevatten. Het bouwen van een geautomatiseerde levelgenerator lijkt iets waarmee de AI me zou kunnen helpen.

Gebruikmakend van de nieuw gevonden 3D-mogelijkheden, bedacht ik dat de game zich in de aderen zou moeten bewegen in plaats van de laterale beweging die gebruikelijk is voor klassieke 2D-games. Dit introduceert uitdagingen zoals visuele clipping, vervormde graphics als gevolg van de positie van de camera buiten de 3D-scène. Een andere uitdaging is de toegenomen complexiteit van de botsingsdetectie van in-game elementen, een vereiste voor gameplay-mechanismen. De gegenereerde code van het AI-model groeide en groeide met elke iteratie.



3D Game Engine in actie, let op de grafische glitches in de aderwand

Na ongeveer 800 regels code begon ik de beperkingen van het AI-model te ervaren, omdat het fouten en bugs begon te maken die het in eerdere iteraties had opgelost. Elke nieuwe iteratie introduceerde nieuwe problemen en het model bleef verfijningen en oplossingen vergeten die het eerder had toegepast. Dit komt door de limieten op de contextmogelijkheden (input tokens) die worden geïmporteerd om de grote taalmodellen te laten werken. Simpel gezegd: het kan zich niet genoeg herinneren om door te gaan met itereren op het geheel tegelijk.



Productiviteitsparadox: wanneer het bedienen van de machine je meer moeite kost dan de waarde van de output

Ik stopte met het ontwikkelen van de game engine met behulp van AI na een paar dagen intensief experimenteren. Ik bereikte een punt waarop ik meer moeite stak in het beheren van het AI-model dan inspanning gericht op de eigenlijke game engine. Ik dacht dat het gewoon beter zou zijn om wat boeken over de technologieën te halen, ze te lezen, zelf vaardiger te worden en dan de game engine met de hand te bouwen. Wanneer dingen complex worden, lijken onze menselijke hersenen in staat te zijn zich te concentreren op specifieke delen van een complexe taak met respect voor het grotere geheel zoals het is.

Over de dwaasheid van "Natural Language Programming"

Naast de beperkingen van de context die een AI-model kan begrijpen, denk ik dat er het probleem van ambiguïteit is. Ik raad je aan de woorden van [Edsger Wybe Dijkstra](#) te lezen, die reflecteren op de dwaasheid van "natural language programming" in zijn [EWD667](#). Een ondubbelzinnige beschrijving, vrij van verwarde gedachten, vormt de basis van betrouwbare code. Toch laat onze moedertaal te veel ruimte voor allerlei onzin en fouten. In plaats daarvan zouden we onze woorden moeten beheersen, scherp moeten denken en ondubbelzinnig moeten specificeren.

Een formele taal, zoals een programmeertaal of een wiskundig model of een uitputtende in- en outputspecificatie, kan een verbazingwekkend effectief hulpmiddel zijn om ambiguïteit en fouten uit te sluiten. Simpel gezegd: je krijgt de beste output als je de beste input geeft.

De AI prompts met code

Om te testen of de AI me betere code zou geven, besloot ik mijn vraagtaal te veranderen. In plaats van te beschrijven wat ik wilde in natuurlijke taal, uploadde ik een stuk code

dat ik eerder had geschreven, met enkele zeer specifieke kenmerken:

- de code kwam uit een productiesysteem, ik wist dat het goed werkte
- de code bevatte verschillende foutmeldingen, wat bijdroeg aan de robuustheid
- de code bevatte geen syntaxisfouten en gebruikte logisch benoemde variabelen en functies
- de code had een zeer specifieke scope, met duidelijk gedefinieerde interfaces naar de buitenwereld

Naast de code vroeg ik de AI specifiek om één bepaald aspect van de betreffende code te wijzigen. Het moest het gebruik van een bepaald extern programma (de NodeJS-module 'lwip') verwijderen en vervangen door een ander extern programma (ImageMagick). Omdat zowel [lwip](#) als [ImageMagick](#) open source code zijn, is het AI-model goed op de hoogte van deze programma's. Ik gaf het het perfecte recept voor het koken.

De code die ik van de AI kreeg, was een 99% perfecte drop-in-vervanging voor mijn eigen code. De gegenereerde code had één bug waardoor sommige parameters niet goed werden escaped, wat leidde tot een crash. Ik kon de fout handmatig opsporen en verhelpen, omdat ik volledig vertrouwd en deskundig was binnen de context van dit specifieke stuk software. Na wat meer testen besloot ik de door AI gegenereerde code in productie te nemen, omdat de nieuwe bibliotheek veel beter presteerde dan de bibliotheek die hij verving.

Conclusie

Vibe coding met AI maakt snelle prototyping en leren mogelijk, zoals te zien is in het succes van Pieter Levels, maar faalt bij complexe projecten die precisie vereisen. Het roept een vraag op: vergroot AI ons potentieel of beperkt het ons tot zijn beperkingen? Meesterschap combineert menselijk denken met de kracht van AI. Denk na over vakmanschap: wanneer scherpen tools je vaardigheden aan en wanneer kunnen ze die bot maken?