הצד המתוק של ה-Bitter Lesson

Thariq Shihipar טוען שהמודלים לא מכתיבים לנו איך לעבוד, ושאחרי שנה של תיעוש התוכנה מגיע תורו של הרנסנס

Thariq Shihipar · AI Engineer (CODE Summit 2026) · 2026-11-19
משך: כ-19 דקות · YouTube
סיכום אוטומטי — AI Briefing (aibriefing.dev)

על המסמך הזה

זה keynote קצר, כ-19 דקות, מ-CODE Summit 2026 (שהוצג בחסות Google DeepMind וערוץ AI Engineer). Thariq Shihipar עזב לאחרונה את Anthropic ועובד היום באופן עצמאי. הוא מסתכל אחורה על השנה החולפת: מה ניבא נכון, ובעיקר איפה טעה. מכאן הוא בונה מסגרת שהוא קורא לה "הצד המתוק של ה-Bitter Lesson".

הטענה המרכזית: החשש שהמודלים יכפו עלינו דרך עבודה אחת, "זרה", התברר כלא נכון. רק Bash, רק React, רק diffusion לאמנות. ככל שהמודלים משתפרים הם טובים בהמון דרכים במקביל, וזה ה-smooth interior. לכן האחריות עוברת אלינו: להחליט איך תוכנה איכותית צריכה להיראות. כדי להמחיש, הוא מציג דמו ממשחק לחימה שהוא בונה, ובו הוא מפריד בין קוד engine שהסוכן כותב לבין שכבת spec קריאה, כמעט שפת סקריפט, שהוא עצמו קורא ועורך.

איך לקרוא את המסמך

- **1. תקציר מנהלים:** הטענות המרכזיות עם חותמות זמן, וגם מה לא נאמר.
- **2. במה טעיתי:** Bash, React ו-diffusion models, שלוש תחזיות שלא התממשו.
- **3. מהפכה תעשייתית ורנסנס:** שתי מגמות שקורות במקביל בתוכנה.
- **4. Jagged frontier מול smooth interior:** איך למדוד מודל לפי רוחב ולא רק לפי שיא.
- **5. כש-vibe coding מאבד את המפתח:** הבעיה שעלתה במשחק.
- **6. הפרדת engine מ-spec ודמו:** הפתרון המעשי, תיקיית קוד שנכתבה לקריאה אנושית.
- **7. לתכנן רנסנס של תוכנה:** הרשימה שהוא משאיר לקהל.
- **8–9. מסקנות, שאלות פתוחות ומילון מונחים.**

הערת מקור המסמך נוצר אוטומטית מהתמלול ומהשקפים של ההקלטה. ייתכן ששמות ומונחים שובשו בתמלול האוטומטי (ASR). המקרים הבולטים מסומנים בטקסט. הציטוטים מועתקים מילה במילה מהתמלול ונבדקו מול חותמת הזמן שלהם. לא נוסף מידע ממקורות חיצוניים.

1. תקציר מנהלים

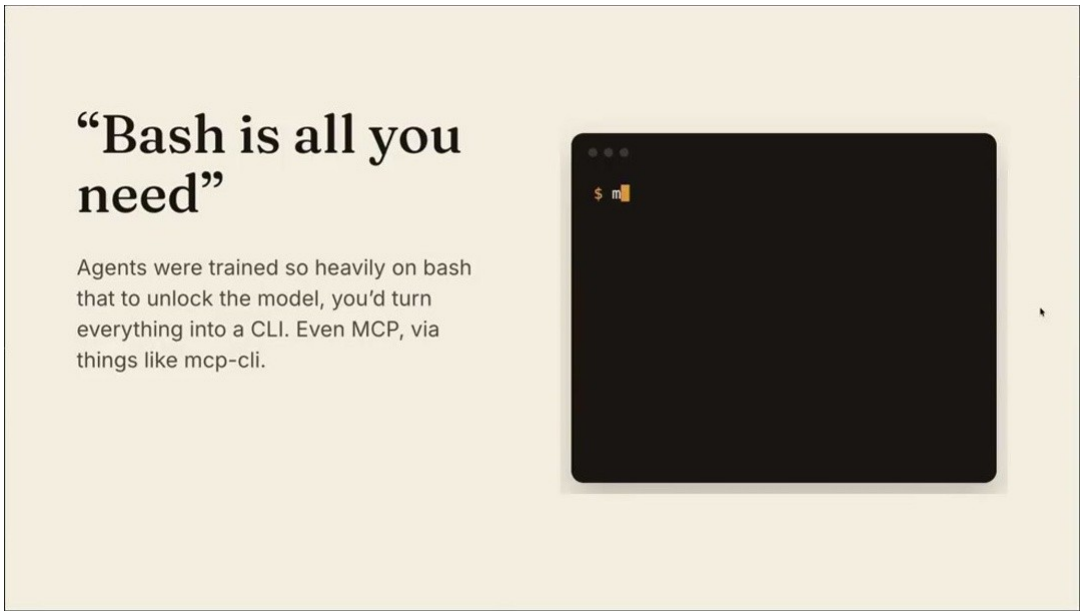
זו הרצאה של רעיונות ולא של מספרים. Shihipar מנסה להזיז את נקודת המבט ממה שהמודל עוד לא יודע לעשות למה שאנחנו רוצים לבנות איתו. הדמו קצר, אבל הוא נותן דפוס עבודה קונקרטי שאפשר להעתיק.

- **הנחת "Bash is all you need" לא החזיקה:** היום הוא מעדיף MCP על פני CLI, ויש גם code mode, skills ו-progressive disclosure. לדבריו, השימוש ב-Bash היה "weird remnant" של אופן האימון.
- **גם React לא נהיה "השפה האחרונה":** מודלים כותבים HTML, React, Swift ואפילו את פורמט ה-JSON הקנייני של Figma, "And they'll all look roughly the same".
- **אמנות לא נבלעה על ידי diffusion:** LLMs יוצרים אמנות דרך קוד ושליטה בכלים כמו Blender, Photoshop ו-Figma.
- **מהנדסי תוכנה הם החוד של האימוץ:** "Anything that is happening to software will happen to knowledge work and other parts of the economy".
- **תיעוש ורנסנס קורים בו-זמנית:** 2026 הייתה שנת ה-software factories, אבל במקביל עולה הדרישה לתוכנה איכותית יותר, וכל תוספת של יצירתיות שווה כסף.
- **ה-jagged frontier:** "If it can write React, it can write Figma, it can write JSON". מה שמעניין אותו הוא רוחב היכולת, לא עוד benchmark.
- **harness, QA ו-code review הם שלב זמני:** "the perfect model... won't need to really verify its work or scan for security vulnerabilities".
- **vibe coding טהור פוגע בשליטה:** במשחק שלו המודל "makes a bunch of choices for me", והוא כבר לא יודע מה כל דמות עושה.
- **הפתרון שלו: שכבת spec קריאה:** תיקייה נפרדת עם linting וכללים ל-Claude, שנקראת כמו אנגלית ומתפקדת כשפת סקריפט פרטית למשחק.

מה לא נאמר

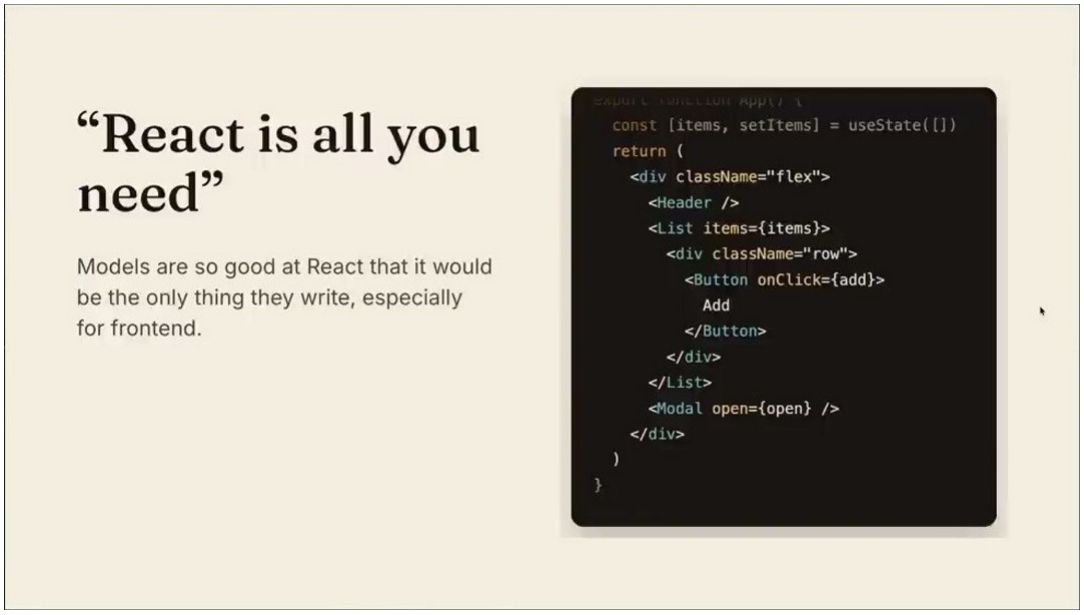
- **אין מדדים:** לא נמסרו מספרים על פרודוקטיביות, זמני פיתוח או איכות, לא לפני ההפרדה ל-spec ולא אחריה.
- **הטענה ש-harness ו-review יתייתרו לא מבוססת:** היא מוצגת כאינטואיציה על "המודל המושלם", בלי לוח זמנים ובלי ראיות.
- **"they'll all look roughly the same":** זו התרשמות אישית. לא הוצגה השוואה שיטתית בין HTML, React ו-Swift.
- **גבולות הגישה:** הוא מודה שהוא "still quite early on" ושלדברים מורכבים הוא עדיין צריך את Claude. לא נאמר איך זה מחזיק בקוד בסיס גדול או בצוות.
- **בחירת כלים:** הוא מזכיר MCP, code mode ו-skills, אבל לא מסביר מתי לבחור בכל אחד.

2. במה טעיתי: Bash, React ו-diffusion



"Bash is all you need": לפי השקף, הסוכנים אומנו כל כך הרבה על bash שהגישה הייתה להפוך הכול ל-CLI, כולל MCP דרך mcp-cli

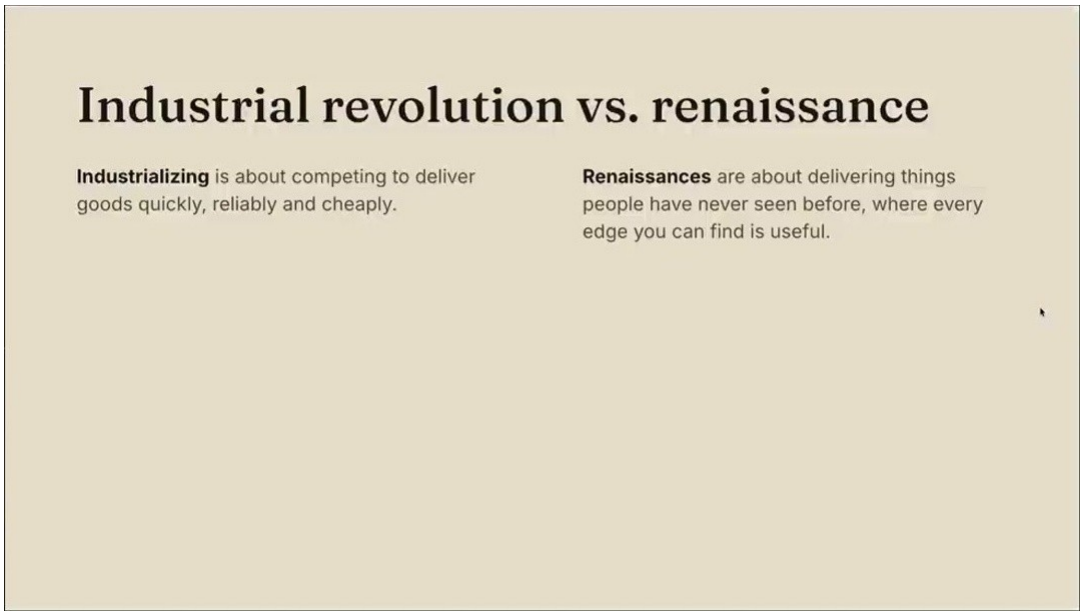
הוא פותח במה שצדק בו השנה: מודלים שעורכים ומייצרים וידאו, מודלים שמשתמשים ב-HTML כדי להבין את העבודה שלהם, ועזרה בתכנון ובלמידה. מיד אחר כך הוא עובר למה שהוא קורא לו memory hole, הנטייה לשכוח את התחזיות שהתבדו. הראשונה: Claude Code נבנה סביב כלי Bash, ומכאן הסיקו ש"There is just going to be Bash and CLIs". זה לא מה שקרה: "I actually prefer MCP to CLIs now, but now there's also code mode and skills and progressive disclosure".



"React is all you need": השקף טוען שהמודלים טובים כל כך ב-React שזה יהיה הדבר היחיד שיכתבו, בעיקר ב-frontend, ולצדו קטע קוד JSX

התחזית השנייה הייתה ש-React יהפוך ל"שפה האחרונה" כי המודלים אומנו עליו. בפועל הם כותבים בכל פורמט ברמה דומה, ו-"I think a lot of people are writing their own languages now". השלישית: diffusion models "יאכלו" את האמנות ולא תהיה לנו שליטה. בפועל LLMs עובדים דרך קוד ודרך computer use בכלים מקצועיים. את כל החששות האלה הוא מכנס תחת המושגים bitter lesson, alien intelligence ו-shoggoth, הפחד שנצטרך להיכנע לאינטליגנציה זרה: "You just have to submit to this foreign intelligence. But that's just like, I don't think that's what happened". כאן מופיעה שאלת המפתח של ההרצאה: "how can we be AGI-pilled and human-centered?"

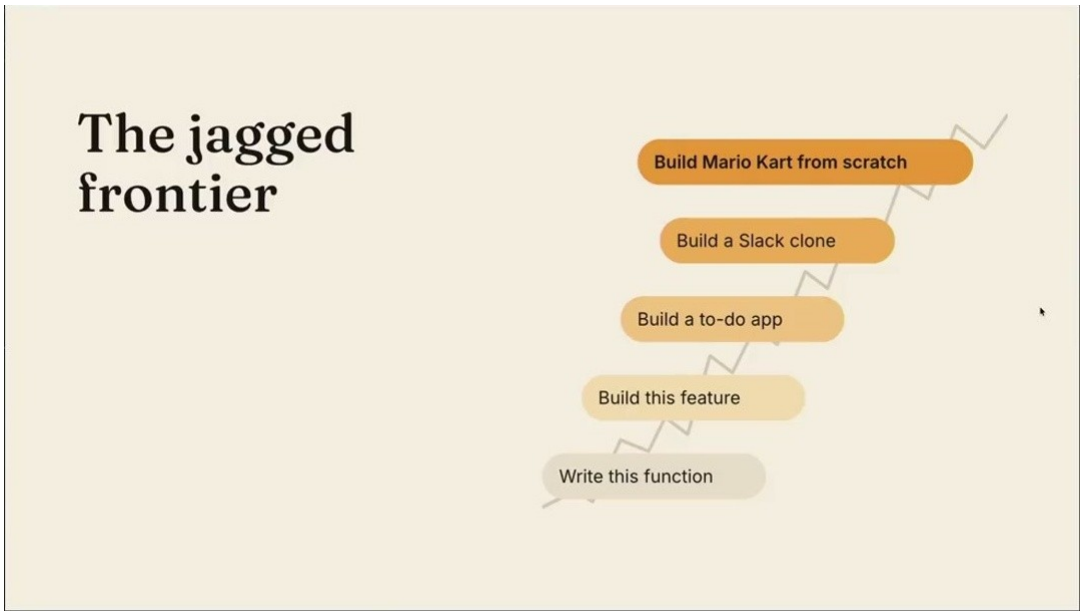
3. מהפכה תעשייתית ורנסנס



Industrial revolution vs. renaissance: תיעוש הוא תחרות על אספקת סחורה מהר, באופן אמין ובזול. רנסנס הוא אספקת דברים שלא נראו קודם, שבהם כל יתרון קטן שווה

לדבריו, מהנדסי תוכנה הם החוד של אימוץ ה-AI, ולכן הדרך שבה אנחנו עובדים עם AI היא מתכון לשאר הכלכלה. הוא מציע להחזיק בראש שתי מגמות במקביל. הראשונה: "I think this year was really about the industrial revolution", שנת ה-software factories, שבה תוכנה הפסיקה להיות מלאכה לא אמינה ונעשתה משהו שכבר לא מגביל אותנו. השנייה: רנסנס, שבו הדרישה לאיכות עולה ו-"every marginal bit of creativity, utility, spirit, like soul that you put into it is useful and economically valuable". הוא מודה שקשה להחזיק את שתיהן בראש בבת אחת, אבל לדעתו שתיהן נכונות.

4. Jagged frontier מול smooth interior



The jagged frontier: מדרגות יכולת עולות, מ-Write this function דרך Build this feature, Build a to-do app ו-Build a Slack clone ועד Build Mario Kart from scratch

ה-jagged frontier הוא מה שה-benchmarks מראים: לפני שנתיים "כתוב פונקציה", היום "בנה Mario Kart מאפס". מולו הוא מציב את ה-smooth interior (בשקף: "Models have also grow horizontally"), המעגל של כל מה שהמודל כבר עושה היטב, ושם היכולת כבר לא משוננת: "If it can write React, it can write Figma, it can

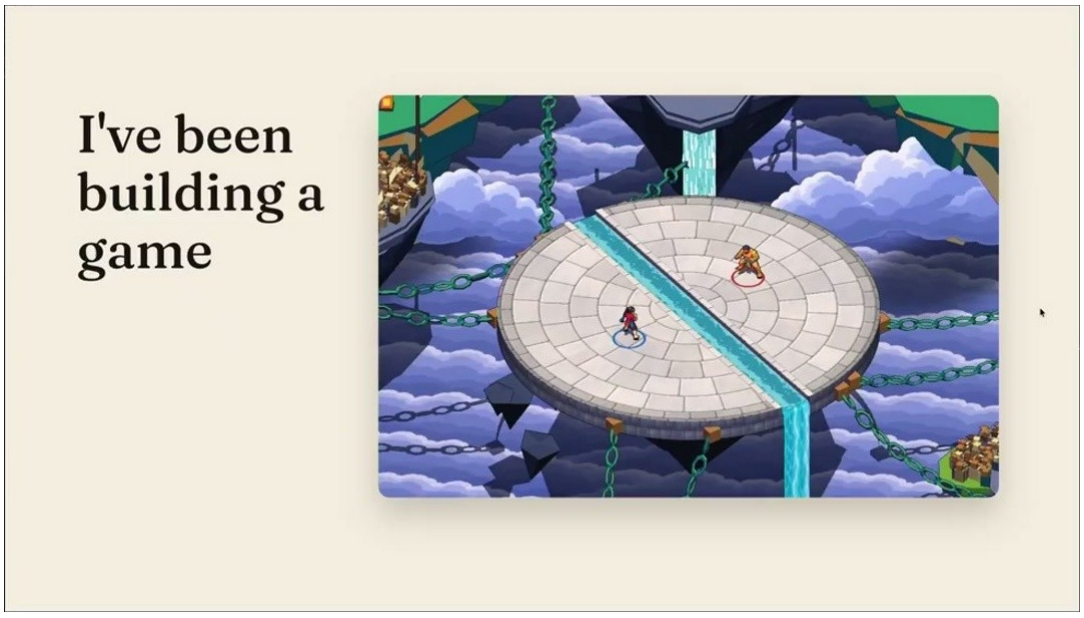
I don't really" :מבחינתו זה החלק שמשנה. "write JSON, it can do all of these capabilities all together
care that they can build Mario Kart from scratch anymore". השאלה שמעניינת אותו היא אם התוכנה
איכותית. שקף נוסף מוסיף שכל מה שדורש היום harness כנראה לא יידרש: המודל המושלם ייכתוב קוד נכון לפי מה שרצית, בלי לבדוק את עצמו ובלי לסרוק פגיעויות.



How much each model has memorized from pretraining: רשת מפות עולם שכל אחת מהן נבנתה משאלת "יבשה או מים?" לכל קואורדינטה, דור אחר דור של מודלי Claude

כדי להמחיש את רוחב הידע הוא מציג eval שבו נותנים למודל קו רוחב וקו אורך ושואלים אם הנקודה ביבשה או במים, ומהתשובות משרטטים מפת עולם. לדבריו מישהו בשם "Kaparthi" פרסם את זה (כנראה Karpathy, השם לא ודאי בתמלול). המודל המוקדם מצייר מפה מטושטשת, והמאוחר כמעט מושלם. שמות הדגמים בתמלול הם "opus 4.6" ו-"Opus 5.5", וגם הם עלולים להיות משובשים. המסקנה שלו: "we can use intelligence to understand and shape how we use intelligence", ולדעתו זה יהיה הנושא של השנה הבאה.

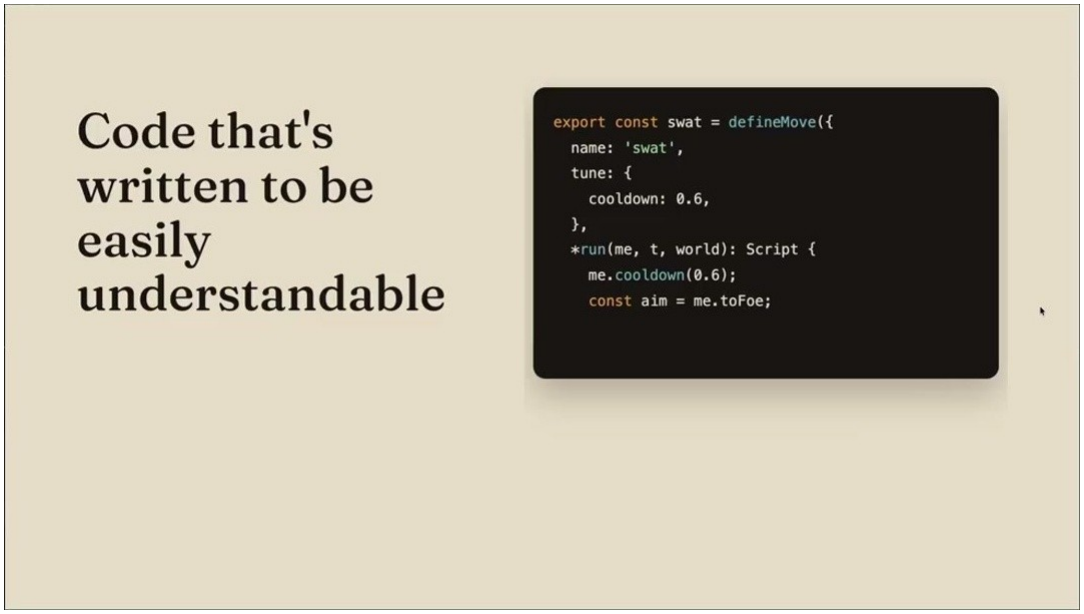
5. כש-vibe coding מאבד את המפתח



"I've been building a game": צילום מסך של משחק לחימה איזומטרי דו-ממדי, זירה עגולה מעל עננים

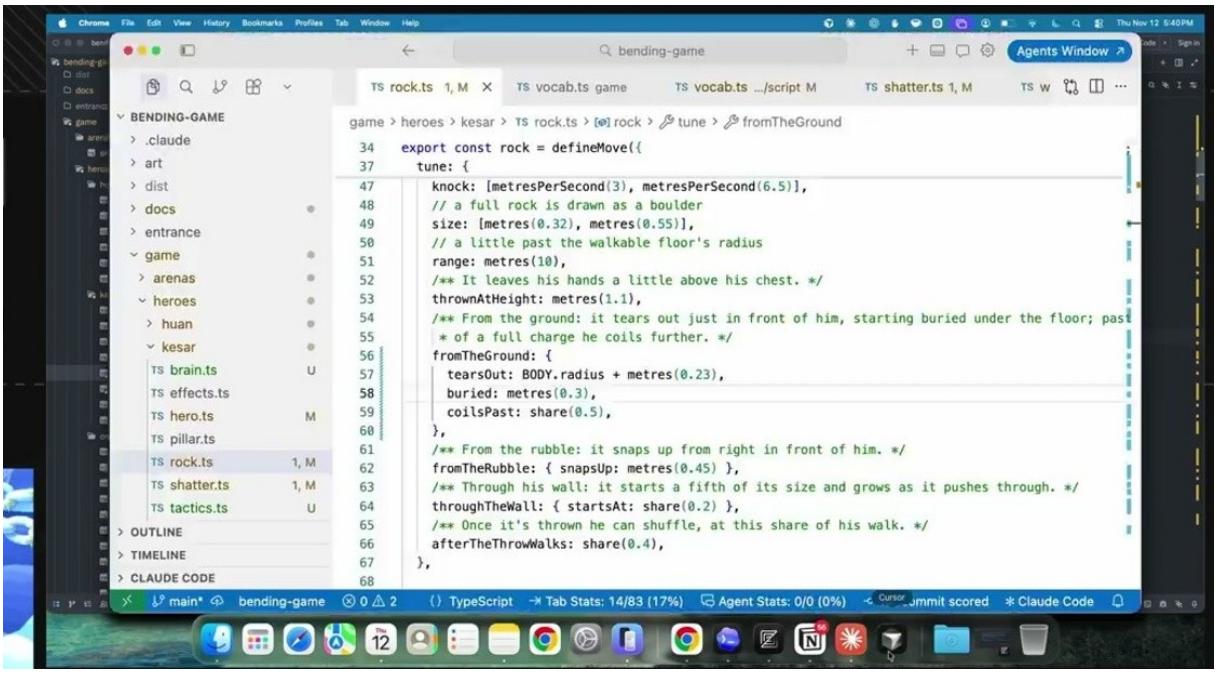
המקרה המעשי הוא משחק לחימה איזומטרי דו-ממדי בהשראת Avatar ו-Smash Brothers. הוא ביקש מ-Claude דמות שזורקת סלע, וגילה שיש עשרות החלטות סמויות: האם הסלע עובר דרך קיר שדמות אחרת מזמנת? באיזו מהירות? מה הגודל? איך עובד ה-charge? "And I find that the model just makes a bunch of choices for me". התוצאה: הוא לא יודע מה הדמויות שלו עושות, הוא לא נכנס ל-zone, הוא שולח prompt ועובר לדבר אחר, ובסוף "I honestly make less progress because of these things". השקפים מסכמים: "I don't think this is acceptable", "Great work is about the designs explored and the choices made (יש יותר מידע במה שהושמט מאשר במה שנכלל), ו-"You simply have to be an expert". צריך להכיר את התחום לעומק ולדעת "את הספרות". בעל פה: "You have to really know your stuff in order to make something great".

6. הפרדת engine מ-spec ודמו חי



Code that's written to be easily understandable: הגדרת move בסגנון defineMove עם שם, פרמטרי tuning כמו cooldown ופונקציית run שמפעילה את המהלך

הפתרון שלו, שהוא מדגיש שהוא עדיין מוקדם: לחלק את קוד הבסיס לשניים. חלק אחד הוא engine שהסוכן כותב והוא לא מתיימר להבין, למשל מצלמות ו-rendering. החלק השני הוא תיקייה שהוא קורא לה spec, ושוקל לשנות את שמה ל-human. בתיקייה הזו יש linting וכללים ל-Claude שמגדירים איך הקוד צריך להיראות: "It's supposed to feel like English". הצפיפות מכוונת כך שקריאה אחת מספיקה כדי להבין מה קורה, והקוד משמש כ-ground truth של המשחק. בדמו הוא פותח את ה-move של זריקת הסלע. הוא מתועד היטב, משתמש ביחידות כמו מטרים במקום "0.5", ומראה שלוש דרכים לשלוף סלע: מהקרקע, מהקיר ועוד אחת. "I've essentially created this almost a custom scripting language for my game".

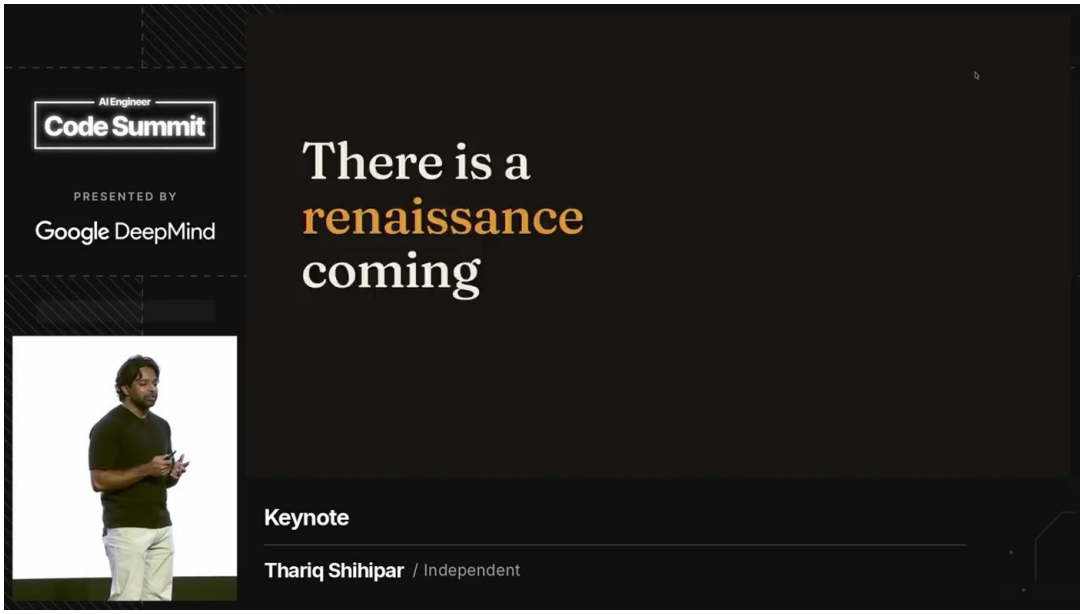


מתוך הדמו: קובץ ה-move של הסלע ב-IDE, עם פרמטרים כמו radius ו-speed במטרים והערות שמתארות כל וריאציה של שליפת הסלע

> **Thariq Shihipar** — And this is kind of like spec-driven development, but the spec is this scripting language, basically.

בדמו הוא משנה את גודל הסלע ל-1.5, מריץ את המשחק ורואה את הסלע נטען וגדל. לעריכה הוא חזר ל-IDE ממשי: הוא משתמש יותר ב-Cursor ובמודל ה-tab, למשל כדי להשלים בהערה כלל כמו "אם מזמנים קיר באמצע, הכנס אותו לתור כפעולה". הוא מסייג: "when I need to do something complicated, I still need to ask Claude to do it", ו-Claude הוא שבונה את ה-primitives. הרווח לדבריו: "And I feel much more in control", ואפשר לעבור על מה ש-Claude עושה בזמן שהוא עובד.

7. לתכנן רנסנס של תוכנה



"There is a renaissance coming": שקף הסיום שלפני השאלה "?What do we want the future of work to look like"

המסקנה שלו: כבר אין סיבה להיות מוגבלים ביכולות המודל או במה שהוא "אוהב" לכתוב. השאלה היחידה היא איך מספקים תוכנה מעולה. מכאן רשימת כיוונים: שפות חדשות (יעילות כמו Rust, אבל גם מודולריות, hot reloadable וקריאות), primitives חדשים (סוגי מסדי נתונים, שרתים ואינטראקציות עם מערכת ההפעלה), פרדיגמות

אינטראקציה חדשות שבזכותן כדאי לחזור למחקרי HCI ישנים, ואסתטיקה חדשה. הוא מודה ש-"I don't think any of this is quite new" ושרוב הדוברים בכנס אמרו דברים דומים. הסיום: "Because the models are not this alien intelligence that's trying to force us into a specific way of working. They can kind of do everything, and they will. But our job is to bring about the Renaissance."

8. מסקנות ופעולות

- **ליצור שכבת spec קריאה:** בפרויקט שנבנה עם סוכן, לייחד תיקייה ללוגיקה העסקית או לחוקי המשחק. לכתוב לה כללי lint והנחיות ל-Claude שדורשים קוד "שנקרא כמו אנגלית", ולהשאיר ל-agent את ה-engine.
- **להשתמש ביחידות של התחום ולא במספרי קסם:** מטרים, שניות ושמות של פעולות במקום 0.5. זה מה שמאפשר לערוך את הקוד ביד בלי להבין את ה-engine.
- **להחזיר את ה-IDE:** לכיוונונים קטנים, עריכה ידנית ו-tab completion משאירים אותך בתוך העבודה. לבקש מהסוכן רק primitives מורכבים.
- **לא לבחור stack לפי מה שהמודל "אוהב":** לפי Shihipar, ה-smooth interior מאפשר לבחור פורמט וכלי לפי איכות התוצאה.
- **לבדוק אם אתה עוד מבין מה בנית:** אם אחרי vibe coding אתה לא יודע מה המערכת עושה, זה סימן לבנות ground truth קריא.

שאלה פתוחה	למה זה חשוב
האם שכבת spec עובדת גם בצוות ובקוד בסיס גדול?	הדמו הוא פרויקט אישי של אדם אחד בשלב מוקדם.
מי שומר על ההתאמה בין ה-spec ל-engine?	כש-Claude בונה primitives חדשים, ה-spec עלול לא לשקף את ההתנהגות בפועל.
מתי באמת אפשר לוותר על review ו-QA?	זו טענה על "המודל המושלם", ולא הוצגו ראיות או לוח זמנים.
מתי לבחור MCP, CLI, code mode או skills?	הוא מעדיף MCP, אבל לא נתן קריטריונים.
איך מודדים "איכות של רנסנס"?	אין מדד מוצע, רק תחושת שליטה ומומחיות.

9. מילון מונחים

מונח	פירוש
Bitter Lesson	הטענה של Rich Sutton ששיטות כלליות שמנצלות כוח חישוב מנצחות לאורך זמן ידע אנושי מקודד. כאן במובן של חשש שנצטרך להיכנע לדרך העבודה של המודל
Jagged frontier	גבול היכולות המשונן של המודלים, כפי שהוא נמדד ב-benchmarks
Smooth interior	מושג של הדובר: התחום הרחב והאחיד שבו המודל כבר עושה היטב מגוון משימות
Vibe coding	פיתוח שבו נותנים לסוכן לכתוב את הקוד כמעט בלי לקרוא אותו
Spec-driven development	פיתוח שמונע ממפרט. כאן המפרט הוא קוד קריא בשפת סקריפט פנימית
Harness	תשתית שעוטפת את המודל: בדיקות, אימות וכלים
Software factory	קו ייצור אוטומטי של תוכנה באמצעות סוכנים
MCP	Model Context Protocol, פרוטוקול לחיבור כלים ונתונים לסוכנים
Code mode / Skills / Progressive disclosure	דרכים נוספות שבהן agents פועלים בעולם: כתיבת קוד שקורא לכלים, חבילות יכולת, וחשיפה הדרגתית של הקשר
AGI-pilled	מי שמשוכנע ש-AGI קרוב ופועל בהתאם
Shoggoth	מֶם שמתאר LLM כיצור זר שמסתתר מאחורי פנים ידידותיות (בתמלול: shoguts)
Tab model	מודל ההשלמה האוטומטית של Cursor
Primitives	אבני בניין בסיסיות שה-engine חושף לשכבת ה-spec