Square Enix prueba un LLM local de NVIDIA para hacer juegos

13 de noviembre de 2026, Robotstart (ロボスタ). Selección: Miguel Acevedo

modelos, ia, japon, nvidia, square-enix



El 10 de noviembre de 2026, en el “Agentic AI Day Japan 2026” que NVIDIA organizó en The Prince Park Tower Tokyo, Shota Fujii, de Square Enix, dio una charla titulada “Implementación de IA generativa en una plataforma de desarrollo de juegos”.

Fujii entró a la empresa como recién graduado en abril de 2024 y trabaja en la división de IA y desarrollo de motores, en los equipos de IA generativa y de artista técnico (TA) a la vez.

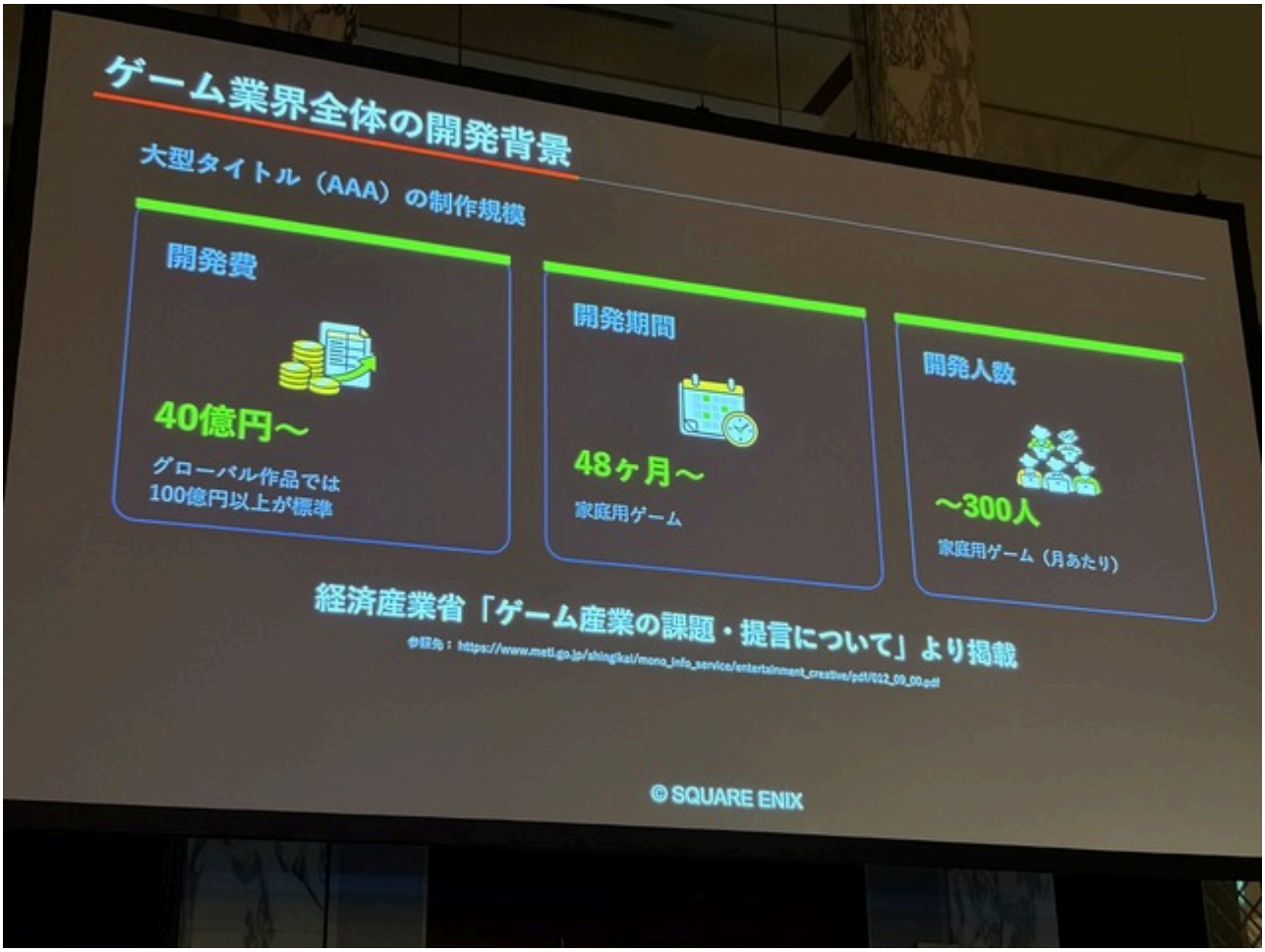
Fujii verificó con “NVIDIA Nemotron 3.5 Super VL” hasta dónde un modelo de lenguaje grande corriendo en la GPU local (LLM local) puede hacerse cargo del trabajo de desarrollo de juegos, y presentó los resultados. El contenido no es un uso en la operación de la empresa, sino una verificación personal de Fujii.



Shota Fujii durante la charla

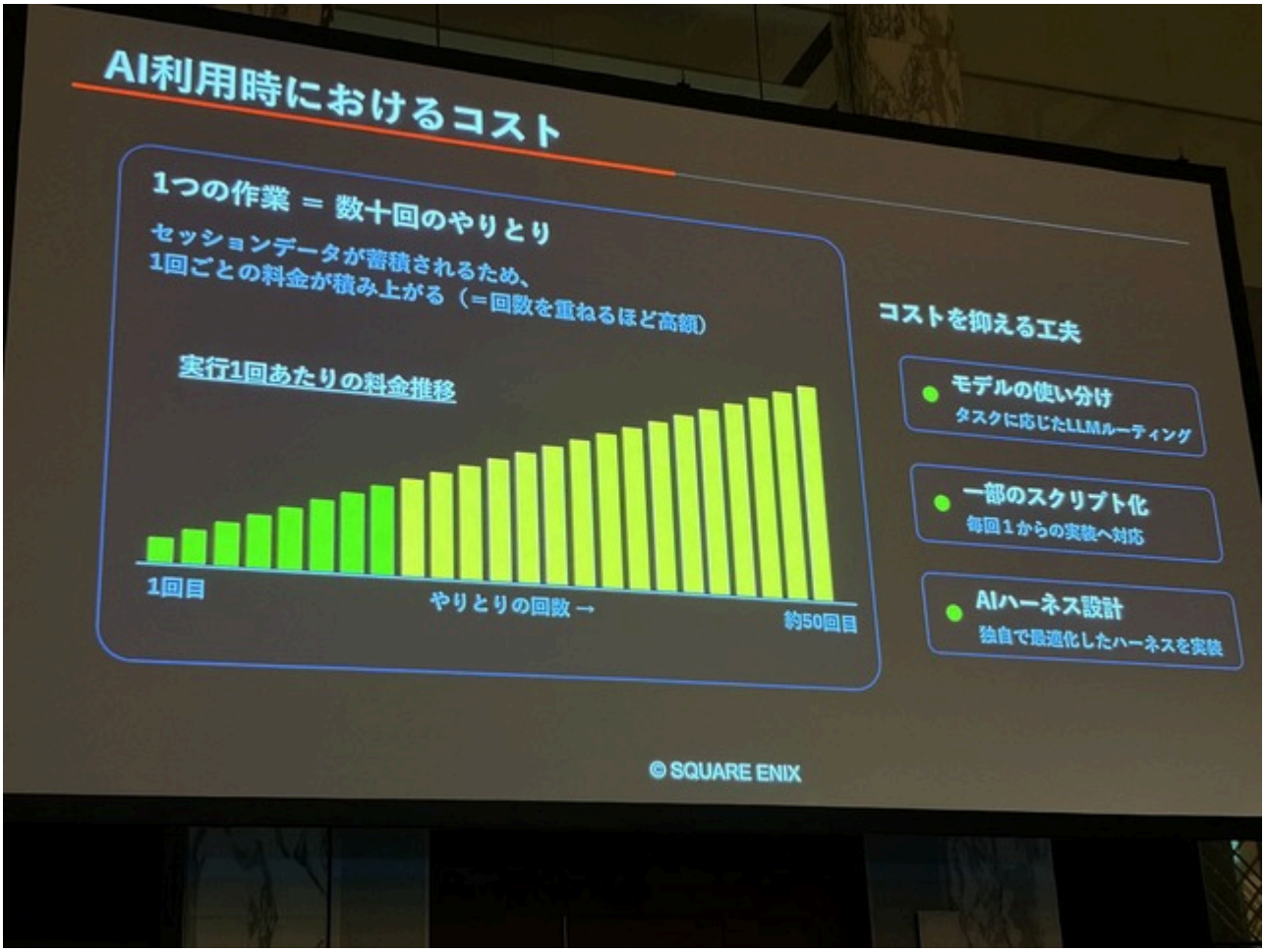
Costos de desarrollo, salarios y tarifas de IA: el punto de partida para mirar al LLM local

Fujii primero mostró la escala del desarrollo en la industria del juego. Según el documento del METI “Problemas y propuestas de la industria del juego”, el costo de un título grande (AAA) va de 4.000 millones de yenes, y un título global supera como estándar los 10.000 millones. El plazo de desarrollo de un juego de consola es de 48 meses o más, y el equipo llega a 300 personas por mes.



Diapositiva con costos, plazos y personal de los títulos grandes, según el documento del METI

A los salarios en alza se suma la preocupación por las tarifas de IA. Una tarea se compone de decenas de intercambios, y en cada intercambio se acumulan los datos de sesión (el historial de la conversación), así que el costo no para de crecer. En el desarrollo de juegos, probar una y otra vez ideas sin respuesta correcta y convertir imágenes y videos de alta resolución en muchos tokens empuja todavía más el gasto.

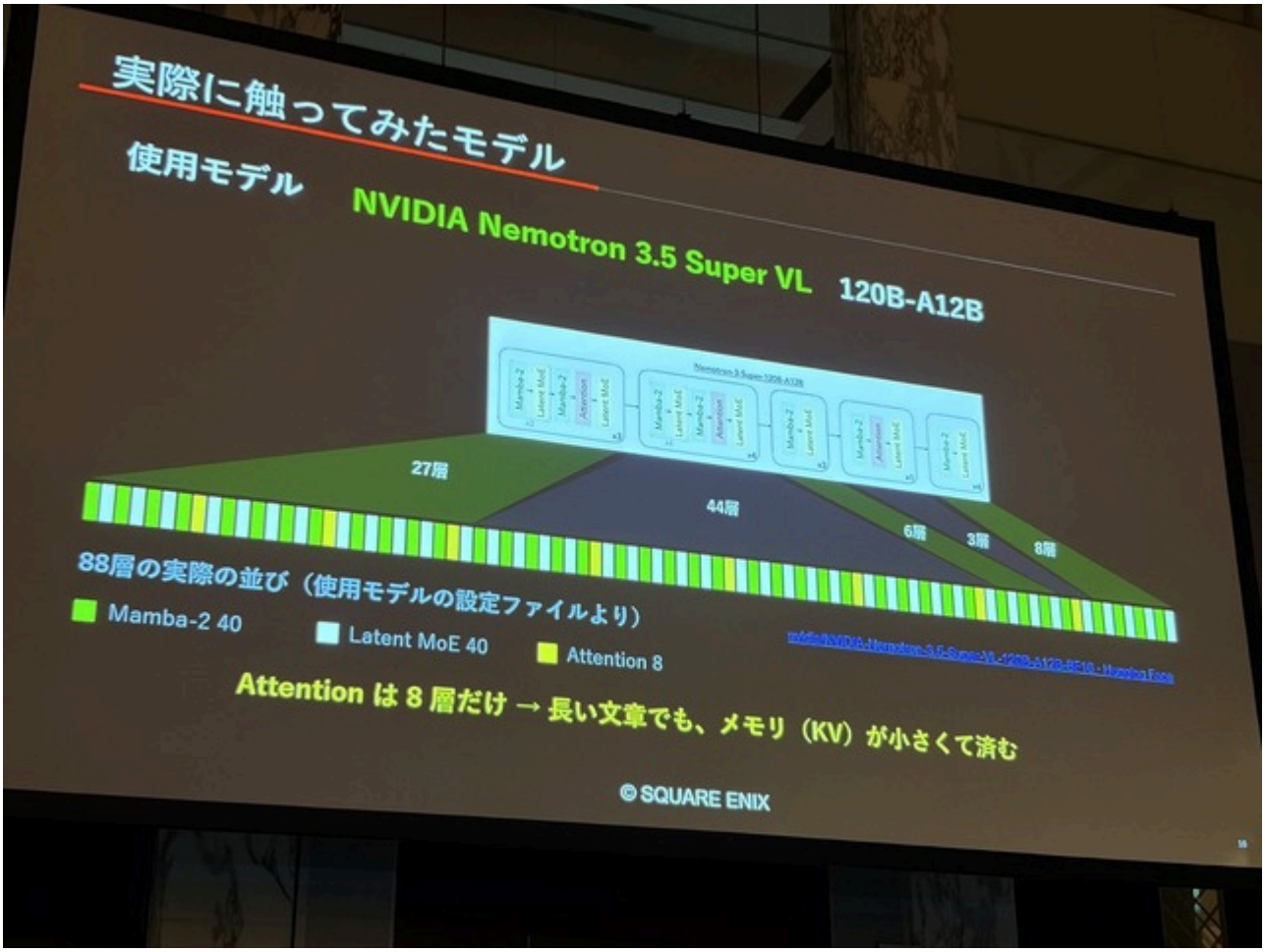


Diapositiva que muestra cómo el costo por llamada crece con la cantidad de intercambios, y tres ideas para bajarlo

Como contramedidas, Fujii nombró tres: enrutamiento de LLM según la dificultad de la tarea, convertir en scripts una parte del procesamiento, y diseño de harness de IA, que optimiza el entorno que hace funcionar a la IA. La opción siguiente es el LLM local. Sus ventajas, explicó, son tres: seguridad, porque los datos confidenciales se procesan dentro de la empresa; consistencia, porque se evita que una actualización del modelo en la nube cambie la calidad de salida; y disponibilidad, porque baja la dependencia de APIs externas.

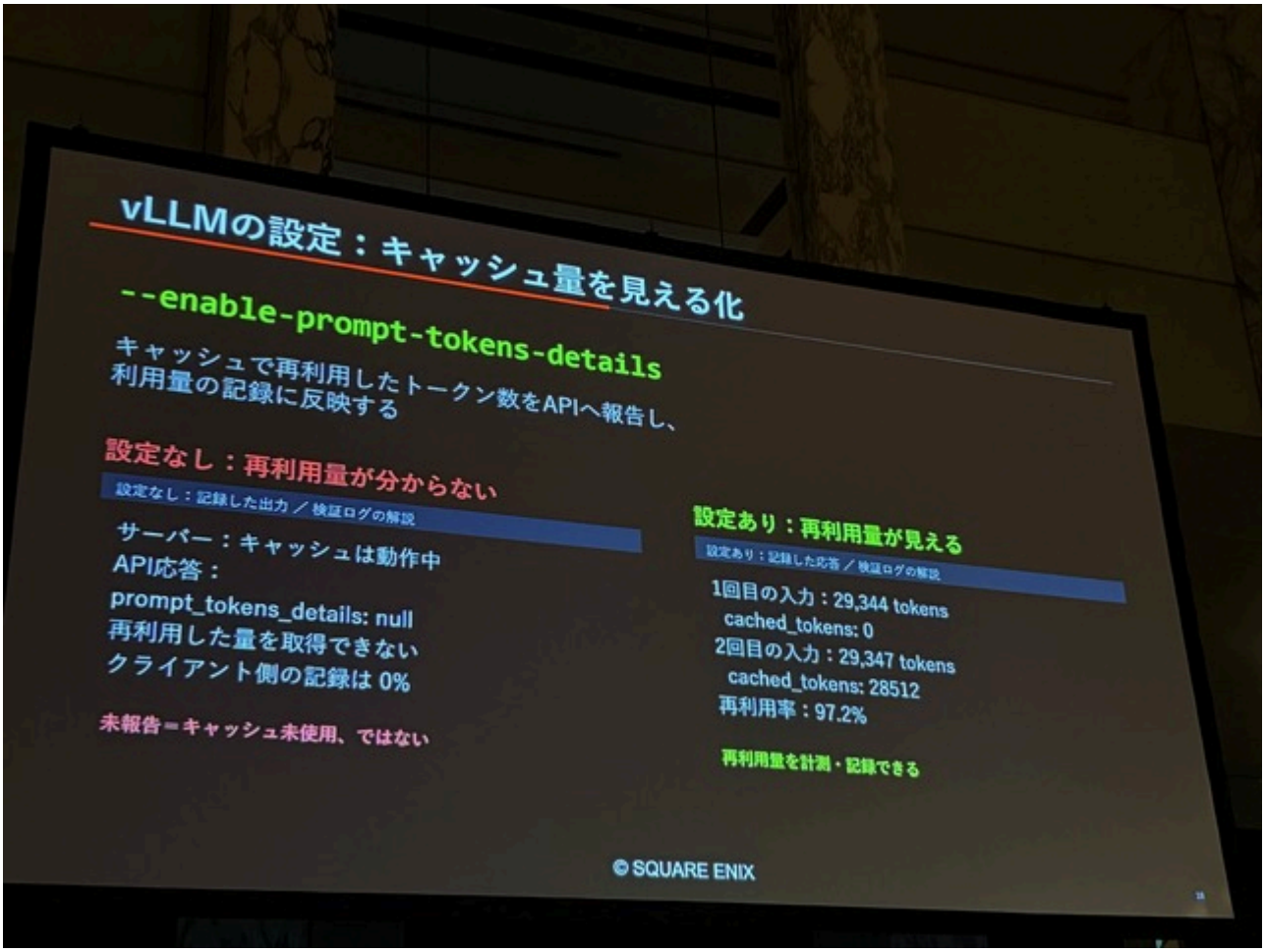
El Nemotron de la verificación achica la atención a 8 capas

El “NVIDIA Nemotron 3.5 Super VL” usado en la verificación es un modelo de visión y lenguaje que también procesa imágenes y video, con el número de modelo 120B-A12B. De sus 88 capas, 40 son Mamba-2, 40 son Latent MoE y 8 son Attention. Fujii destacó que, al tener solo 8 capas de Attention, la memoria (KV) necesaria es chica incluso con textos largos.



Diapositiva con la composición de capas del modelo usado

En la operación, presentó la configuración de agregar “–enable-prompt-tokens-details” al framework de inferencia vLLM. Sin esa opción, aunque el servidor esté usando caché, la respuesta de la API no muestra cuánto se reutilizó. Con la opción activa, en el ejemplo la primera entrada fue de 29.344 tokens y en la segunda, de 29.347 tokens, se reutilizaron 28.512: una tasa de reutilización del 97,2%. En agentes de IA que trabajan acumulando registros, esta medición sirve, dijo Fujii.

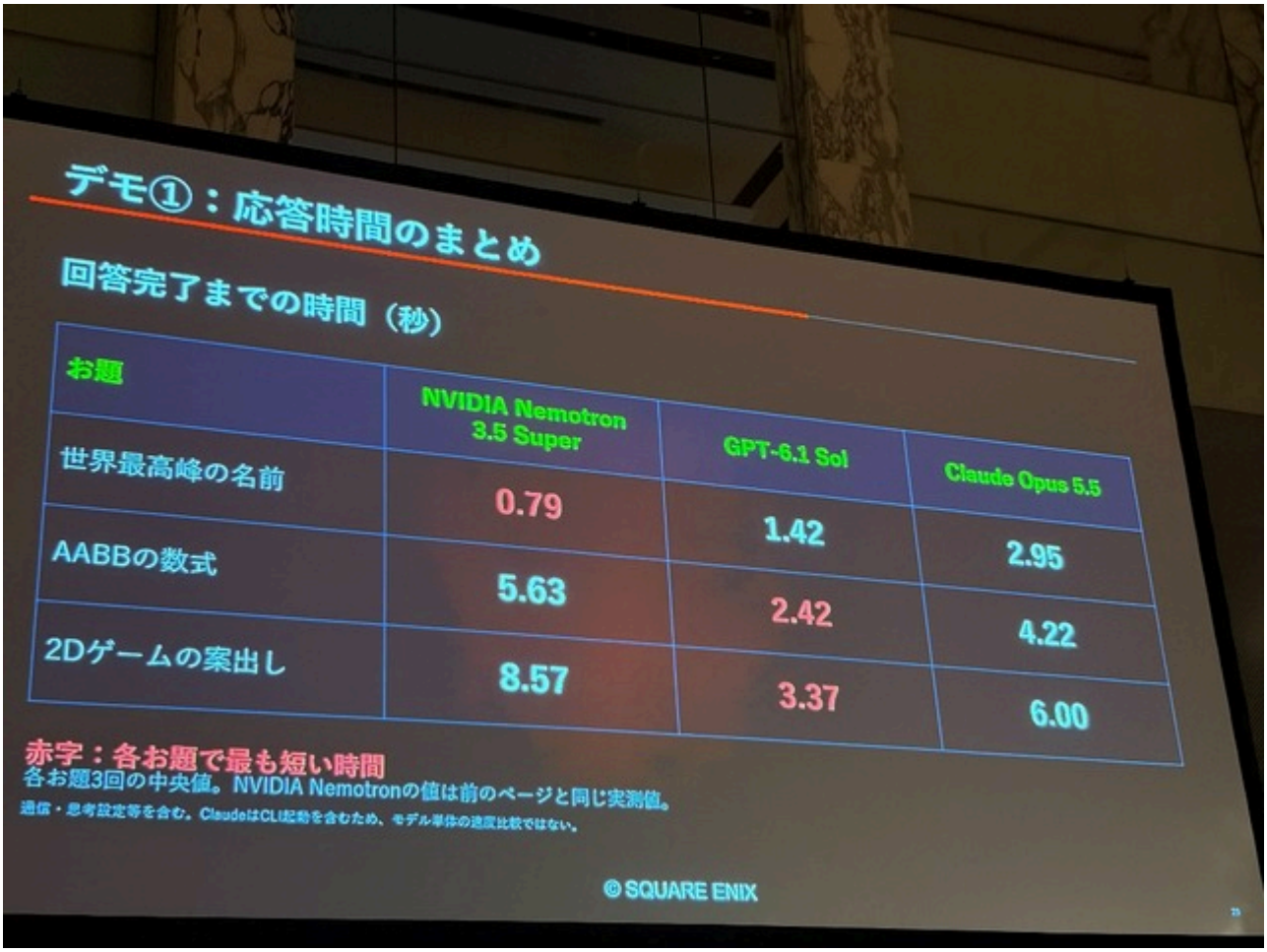


Diapositiva que explica la configuración para ver la cantidad de caché reutilizada

En preguntas simples Nemotron fue el más rápido; la diferencia se achicó al subir la dificultad

La verificación se hizo en cuatro niveles de dificultad creciente: chat, generación de scripts, generación de movimiento e integración MCP.

En el primer chat, se compararon los tiempos de respuesta de tres consignas entre NVIDIA Nemotron 3.5 Super, GPT-6.1 Sol y Claude Opus 5.5. La mediana de tres intentos por consigna fue la siguiente. En “el nombre de la montaña más alta del mundo”, Nemotron tardó 0,79 segundos, GPT-6.1 Sol 1,42 y Claude Opus 5.5 2,95. En “la fórmula de AABB”, fueron 5,63, 2,42 y 4,22 segundos, y en “idear un juego 2D”, 8,57, 3,37 y 6,00 segundos. Nemotron fue el más corto solo en la pregunta más fácil. La diapositiva aclara que Claude incluye el tiempo de arranque de la CLI, así que no es una comparación de velocidad del modelo solo.



Diapositiva con la tabla que compara los tiempos de respuesta de los tres modelos

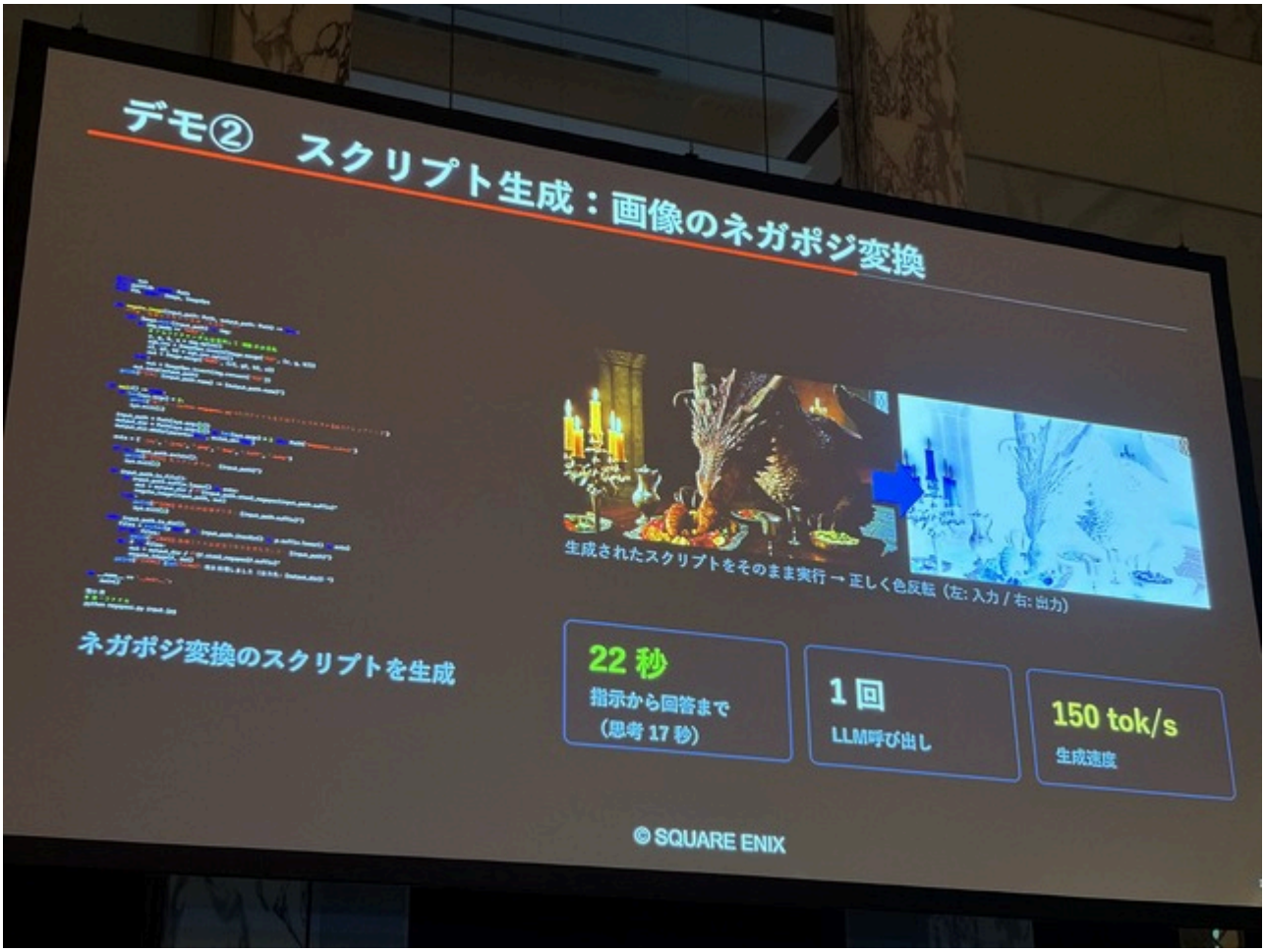
Como la diferencia de velocidad cambia según la dificultad de la pregunta, Fujii concluyó que en chats simples es fácil incorporar un modelo local.

También probó la comprensión de video. Al cargar un video de 5,2 segundos y 864×480 píxeles, “la comida del dragón”, el modelo describió con precisión una escena donde un dragón enorme come en una habitación oscura y la cámara hace zoom a su cabeza. La salida fue de 780 tokens, en 10,6 segundos con razonamiento y 3,4 segundos con un resumen de tres frases.



Diapositiva con el resultado de Nemotron describiendo la escena del video de la comida del dragón

En la generación de scripts, generó en una sola llamada de LLM un script de Python que invierte los colores de una imagen. De la instrucción a la respuesta pasaron 22 segundos (17 de razonamiento), con una velocidad de generación de 150 tokens por segundo. El script generado se ejecutó tal cual y la imagen se invirtió correctamente.



Diapositiva con el tiempo y la velocidad de generación del script de inversión de colores

También generó de una vez un juego simple en HTML y JavaScript con movimiento, salto y detección de ataque.



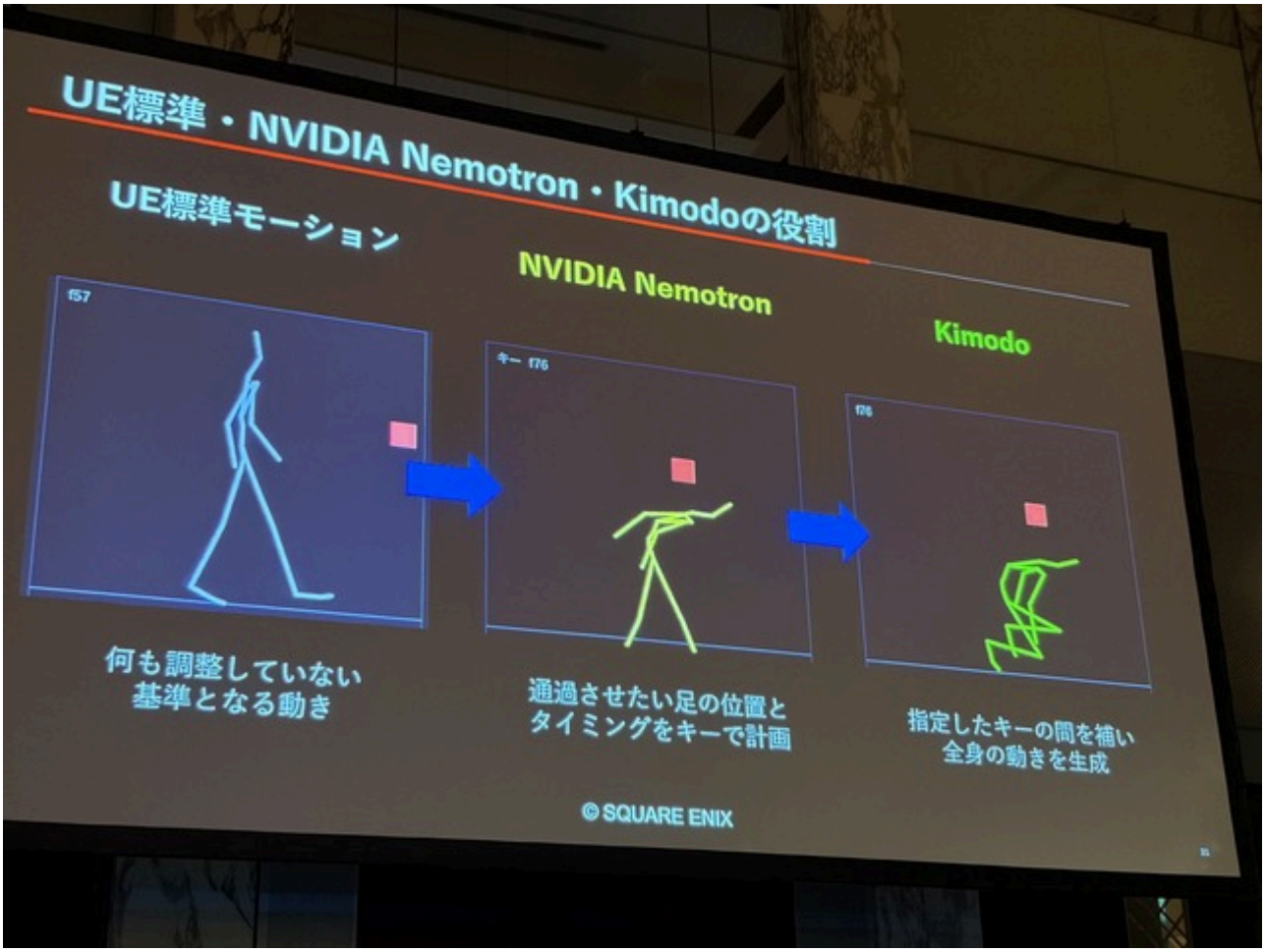
Diapositiva que muestra un personaje moviéndose en el juego simple

Fujii aclaró que eran pruebas de baja dificultad y que en velocidad de salida ChatGPT y Claude estuvieron por encima.

En la generación de movimiento, la ubicación de las claves definió la calidad

La primera verificación aplicada fue la generación de movimiento. En el entorno de Unreal Engine creó movimientos como un personaje que pasa por debajo de una barra, sube a una plataforma o salta una zanja.

El trabajo se reparte en tres etapas. Primero se toma como base el movimiento de caminata estándar de Unreal Engine. Después Nemotron planifica como fotogramas clave la posición y el momento por donde debe pasar el pie. Por último, Kimodo, el modelo de generación de movimiento de NVIDIA, completa los intervalos entre las claves y genera el movimiento de cuerpo entero.



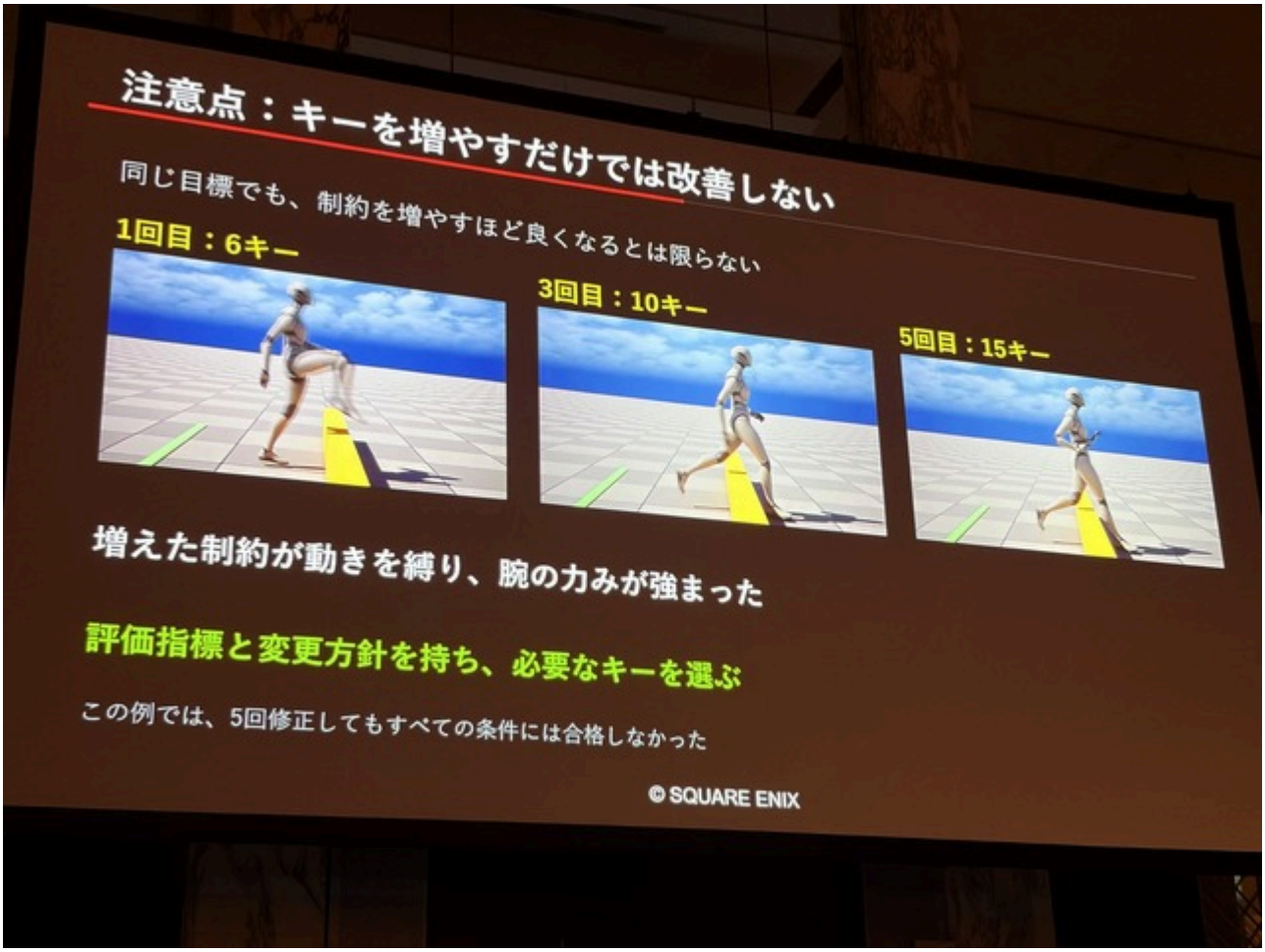
Diapositiva con el reparto de roles entre el movimiento estándar de Unreal Engine, Nemotron y Kimodo

Al movimiento resultante le quedaba deslizamiento de pies. Entonces Fujii giró el ciclo PDCA y corrigió la ubicación de las claves: definir las condiciones de paso, generar el movimiento a partir de las claves, medir penetración y deslizamiento, y reducir las claves a los puntos necesarios. El deslizamiento máximo en apoyo bajó de 16,4 cm a 1,7 cm.



Diapositiva con el resultado: mejorar la ubicación de las claves con PDCA bajó el deslizamiento de 16,4 cm a 1,7 cm

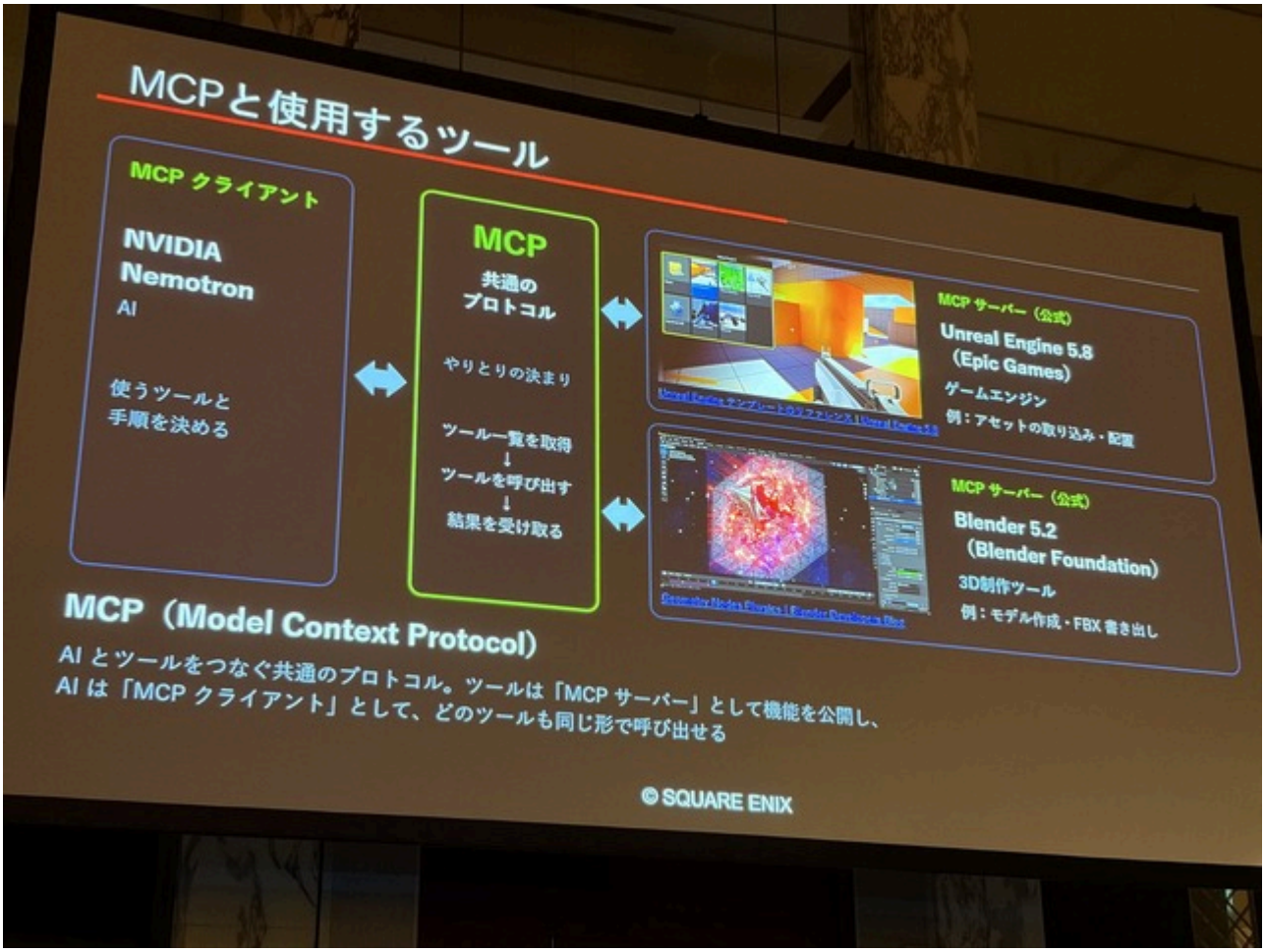
En cambio, más claves no siempre mejoraron el resultado. Con el mismo objetivo, al pasar de 6 claves en el primer intento a 10 en el tercero y 15 en el quinto, las restricciones extra ataron el movimiento y los brazos se tensaron más. En este caso, ni con cinco correcciones pasó todas las condiciones. Fujii dijo que hay que elegir las claves necesarias con un indicador de evaluación y una política de cambio definidos.



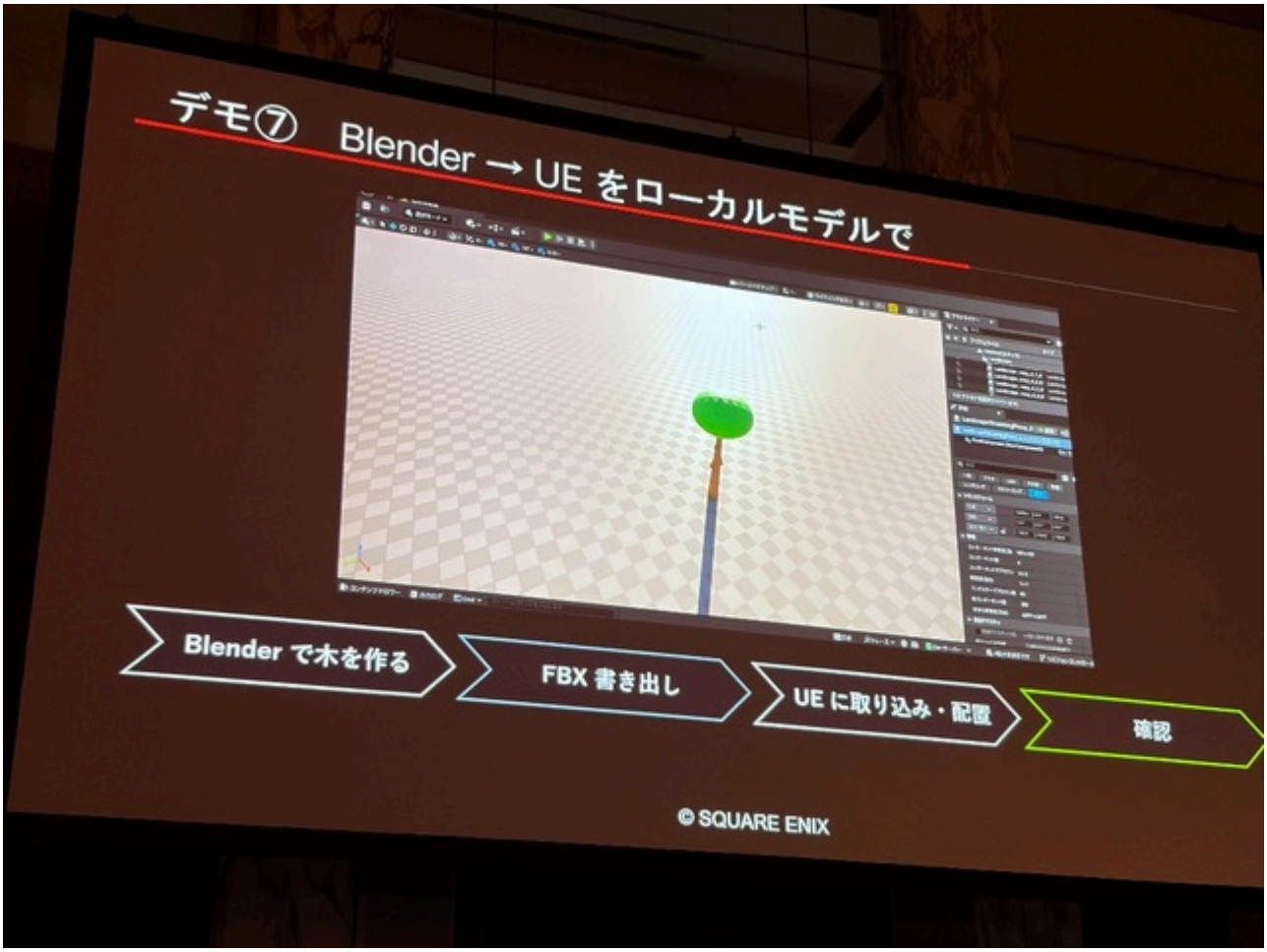
Diapositiva con un ejemplo en el que subir las claves de 6 a 10 y 15 no mejoró el movimiento

Con MCP y enrutamiento, lo local se hizo cargo del 28% del trabajo

La segunda verificación aplicada fue MCP (Model Context Protocol), el protocolo común que conecta IA con herramientas: la herramienta publica funciones como servidor MCP y la IA las llama como cliente. En la verificación se usaron los servidores MCP oficiales de Unreal Engine 5.8 y Blender 5.2. Con el modelo local se pudo ejecutar la tarea de crear un árbol en Blender, exportarlo en FBX e importarlo y ubicarlo en Unreal Engine.

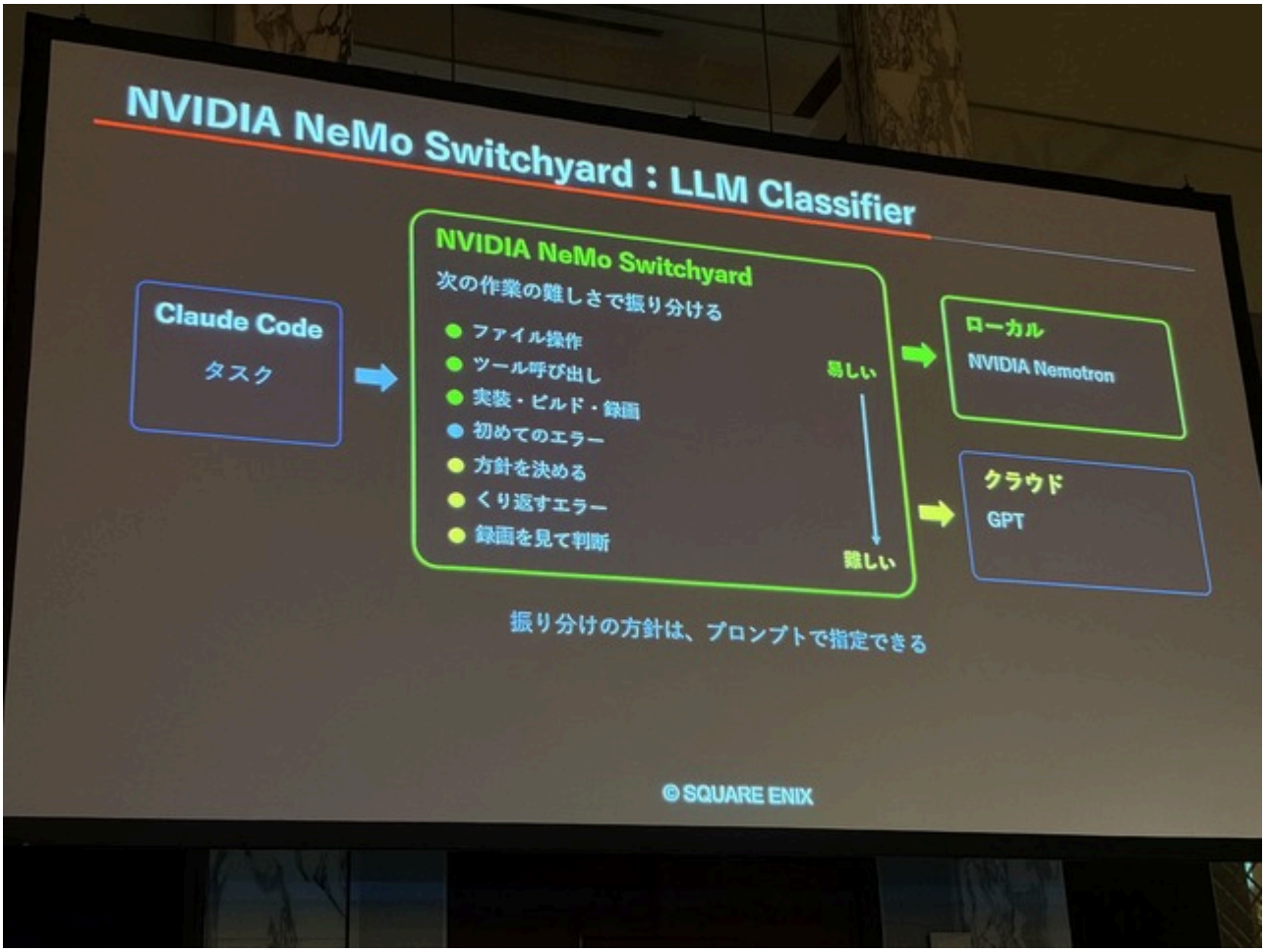


Diapositiva con la configuración que usa a Nemotron como cliente MCP para operar Unreal Engine 5.8 y Blender 5.2



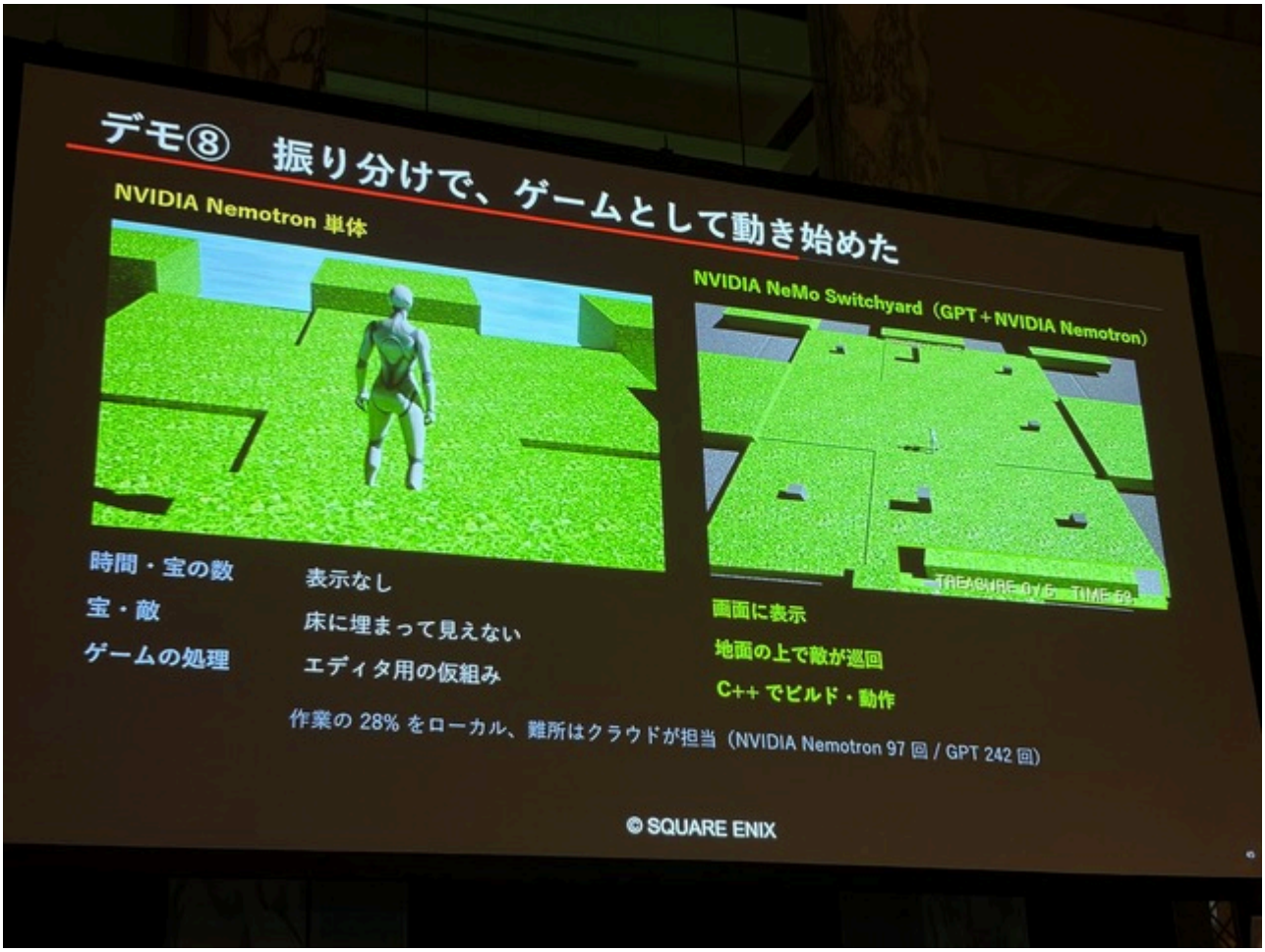
Diapositiva con la demo de Blender a Unreal Engine

El intento de combinar herramientas para crear un juego completo era de dificultad alta y no se logró solo con el modelo local. Entonces Fujii usó NVIDIA NeMo Switchyard, un enrutador que reparte por dificultad las tareas que llegan desde Claude Code. Las tareas fáciles (operación de archivos, llamadas a herramientas, implementación, compilación y grabación) van al Nemotron local; las difíciles (decidir la estrategia, resolver errores que se repiten, juzgar viendo la grabación) van al GPT en la nube. La política de reparto se define en el prompt.



Diapositiva del mecanismo que reparte el trabajo entre el Nemotron local y el GPT en la nube según la dificultad

Con Nemotron solo, la hora y la cantidad de tesoros no aparecían en pantalla, y los tesoros y enemigos quedaban enterrados en el piso y no se veían. El procesamiento del juego era un armado provisional para el editor. Con el reparto, la pantalla mostró la información, los enemigos patrullaron sobre el suelo y el juego compiló en C++ y empezó a funcionar como tal. Lo local se hizo cargo del 28% del trabajo, con 97 llamadas a Nemotron y 242 a GPT.

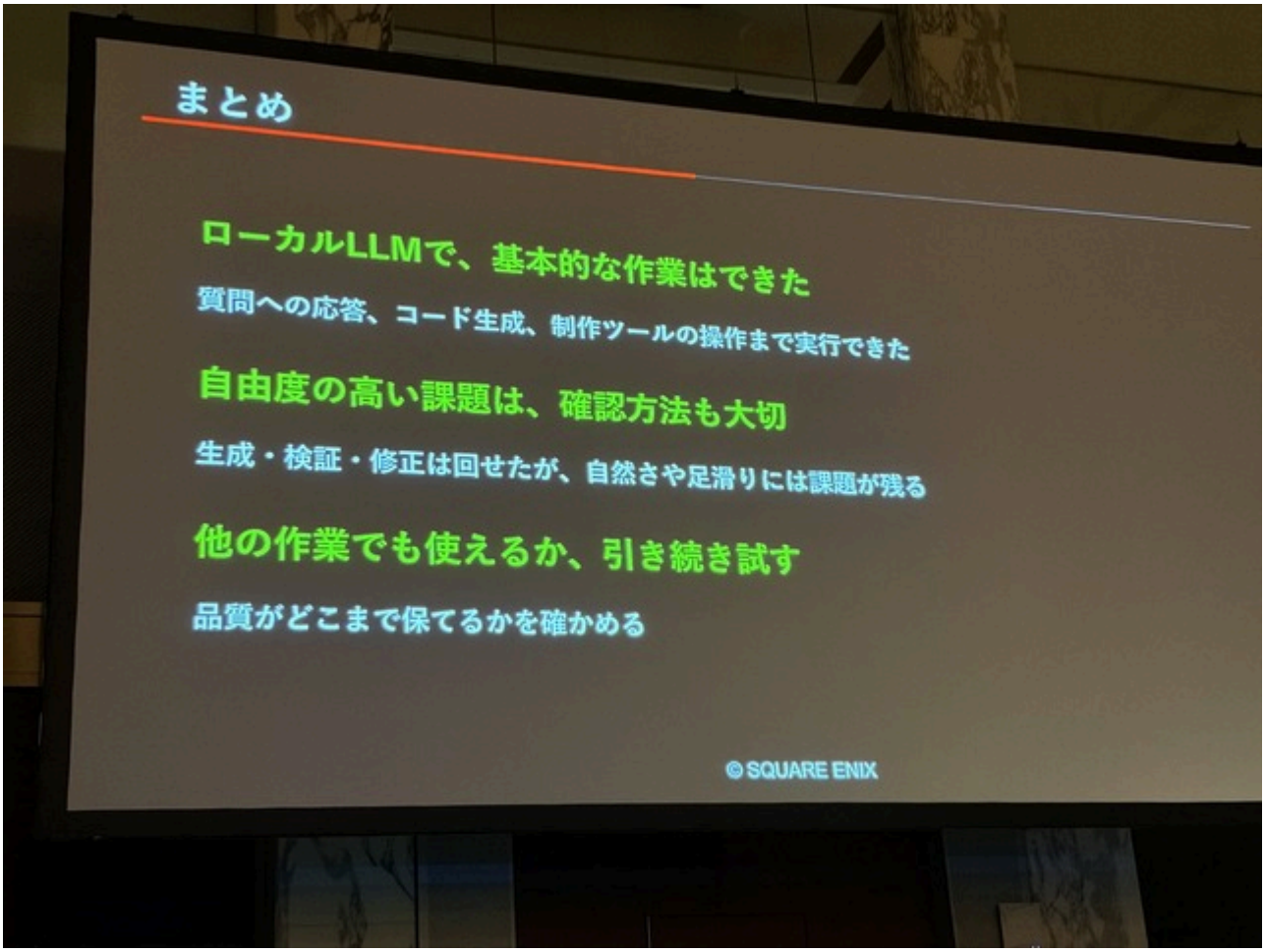


Diapositiva que compara la pantalla del juego con Nemotron solo y con el reparto de NeMo Switchyard

Las tareas básicas funcionan en local; los problemas de alta libertad necesitan verificación

Como cierre de la charla, Fujii nombró tres puntos:

- Responder preguntas, generar código y operar herramientas de producción se pudo hacer con un LLM local.
- El ciclo de generar, verificar y corregir se pudo girar, pero quedan problemas en la naturalidad del movimiento y el deslizamiento de pies.
- Seguirá probando si sirve para otras tareas y hasta dónde se mantiene la calidad.



Diapositiva con el resumen de la charla

En los problemas de alta libertad, hay que preparar primero la forma de verificar el resultado. Fujii también comentó que en las tareas difíciles del desarrollo de juegos va a hacer falta combinar el modelo local con modelos de frontera. Y agregó que la generación de movimiento está muy relacionada con áreas industriales como robots y fábricas, e invitó a aprovecharla en esos sitios.

Fuente: Robotstart (ロボスタ)

roboticaasialatam.com/2026-11-13-nvidia-nemotron-4e43de/