Optimización de la CPU para Videojuegos en Equipos de Bajo Rendimiento mediante el Cambio de Prioridad y Afinidad

John Kyle Freeman Escalona[1], Alejandra Lizbeth Pérez Daniel[2],
Walter Hernández Badillo[3], Jair Vázquez Enriquez[4]
Carlos Daniel Moreno Navarrete[5], MC. Mónica López Hernández[6]

***Resumen*—** En este artículo se muestra una propuesta de optimización de rendimiento en el videojuego Tonatiuh: Sangre al amanecer (desarrollado en Easy FPS Editor) mediante el cambio de prioridad y afinidad del proceso del juego gracias, a un script Batch en conjunto con la herramienta Prioaff. Esta automatización busca evitar saturación de recursos y estabilizar la generación de FPS, así como minimizar la latencia, en equipos de bajo rendimiento. Los resultados obtenidos muestran mayor eficiencia en el uso de la CPU y estabilidad del juego, tras la aplicación de los procesos realizados, la solución es una alternativa accesible y funcional para optimizar videojuegos sin necesidad de contar con equipos de gama alta.

***Palabras clave*—** automatización, videojuego, CPU, afinidad de procesos, FPS

Introducción

En la actualidad, los videojuegos requieren una optimización adecuada en el uso de los recursos de hardware y software para garantizar una experiencia fluida y estable durante su ejecución. Trabajos como los de Geris et al. (2024) y Koulaxidis et al. (2025) demostraron una correlación significativa entre los FPS (fotogramas por segundo) y la latencia en entornos de realidad virtual, donde un mayor número de fotogramas por segundo se asocia con menor latencia, lo que resulta en una mayor experiencia natural para el usuario. Por lo tanto, la automatización en el uso de los recursos de equipos de cómputo es importante para garantizar una buena experiencia de juego (Qi et al., 2014). Dentro de los videojuegos, los procesos gráficos y la lógica exigen una mayor gestión efectiva de la CPU (Central Processing Unit) y la GPU (Graphics Processing Unit), generando una mayor compatibilidad con los equipos de baja gama (Chen et al., 2019).

Distintos trabajos han abordado la mejora del rendimiento dentro de los videojuegos en diferentes enfoques. Aragón- Jurado et al. (2024) utilizaron un algoritmo genético para optimizar automáticamente el motor de Doom Legacy por medio de transformaciones de compilación con LLVM (Low Level Virtual Machine), logrando incrementar los FPS en un 6.21% al ejecutarlo en un Steam Deck. Por su parte, Bastidas García et al. (2023) realizaron un análisis comparativo entre Windows y Linux, comprobando que la elección de sistema operativo llega a influir en el consumo de recursos y estabilidad de los videojuegos.

De igual manera, dentro del proyecto Mi Bosque 3D, López y Villamar (2021) lograron optimizar el rendimiento de los FPS un 55% en tiempo real por medio de un sistema de administración de objetos basados en la proximidad del usuario, logrando liberar recursos y sosteniendo un bajo grado de latencia. Todos estos estudios reflejan la importancia de la combinación de técnicas de optimización automática, el ajustar los entornos operativos y una mejor gestión de recursos para minimizar el rendimiento en hardware limitado.

En este contexto, se propone una solución de optimización del rendimiento aplicada al videojuego Tonatiuh: Sangre al amanecer, desarrollado sobre el motor gráfico Easy FPS Editor, creado por JessicoChan, versión 1.6, y Clark subsecuentemente en la versión actual. Utilizando un script Batch (.bat) el cual contiene una serie de comandos de DOS, que se ejecutan de forma parcialmente automatizada, modificando la afinidad del procesador, logrando que el usuario decida la ejecución del juego en un núcleo, o sea multinúcleo, respecto a las características de su dispositivo. Favoreciendo a computadoras de bajos recursos, por medio de la reducción del uso intensivo de la CPU, evitando latencia al momento de ejecutar el videojuego y mejorando la generación y estabilidad de FPS.

[1] John Kyle Freeman Escalona es estudiante de Ingeniería en Sistemas Computacionales en la Universidad Interamericana, Puebla, México. freemanjohnkyle@gmail.com
[2] Alejandra Lizbeth Pérez Daniel es estudiante de Ingeniería en Sistemas Computacionales en la Universidad Interamericana, Puebla, México perezdanielalejandralizbeth@gmail.com
[3] Walter Hernández Badillo es estudiante de Ingeniería en Sistemas Computacionales en la Universidad Interamericana, Puebla, México whernandezbadillo@gmail.com
[4] Jair Vázquez Enríquez es estudiante de Ingeniería en Sistemas Computacionales en la Universidad Interamericana, Puebla, México jairvazenz14@gmail.com
[5] Carlos Daniel Moreno Navarrete es estudiante de Ingeniería en Sistemas Computacionales en la Universidad Interamericana, Puebla, México morenomorcarlos.4@gmail.com
[6] MC. Mónica López Hernández es profesora de Ingeniería en la Universidad Interamericana, Puebla, México. m.lopez@lainter.edu.mx

Metodología

Funcionamiento del script Batch y la aplicación Prioaff

Los bloques diseñados en la Figura 1 describen la cronología del desarrollo del videojuego, como el de su optimización, esto con el fin de obtener el mejor rendimiento en el entorno de desarrollo. El motor gráfico utilizado se llama Easy FPS Editor, para la creación de juegos de género shooter 3D en primera persona.

Prioaff permite hacer un cambio de afinidad y prioridad del videojuego, así dándole la opción al usuario de decidir con cuántos núcleos ejecutar el programa. La aplicación Prioaff funciona como complemento de la aplicación del videojuego, realizando un cambio de afinidad, como también de la prioridad.

Como se observa en la Figura 1, el bloque de la aplicación Prioaff está constituida por dos componentes: uno que es parsing de argumentos, el cual permitirá conocer la aplicación a optimizar, como los parámetros a implementar (afinidad y prioridad); y el otro es el de creación de procesos, el cual, con la ayuda del parsing, obtendrá los valores que ahora tendrá la aplicación del videojuego en cuanto a afinidad y prioridad. Para entender mejor la lógica de la aplicación de Prioaff en la Figura 2, se muestra un diagrama de flujo que ayudará con la explicación.

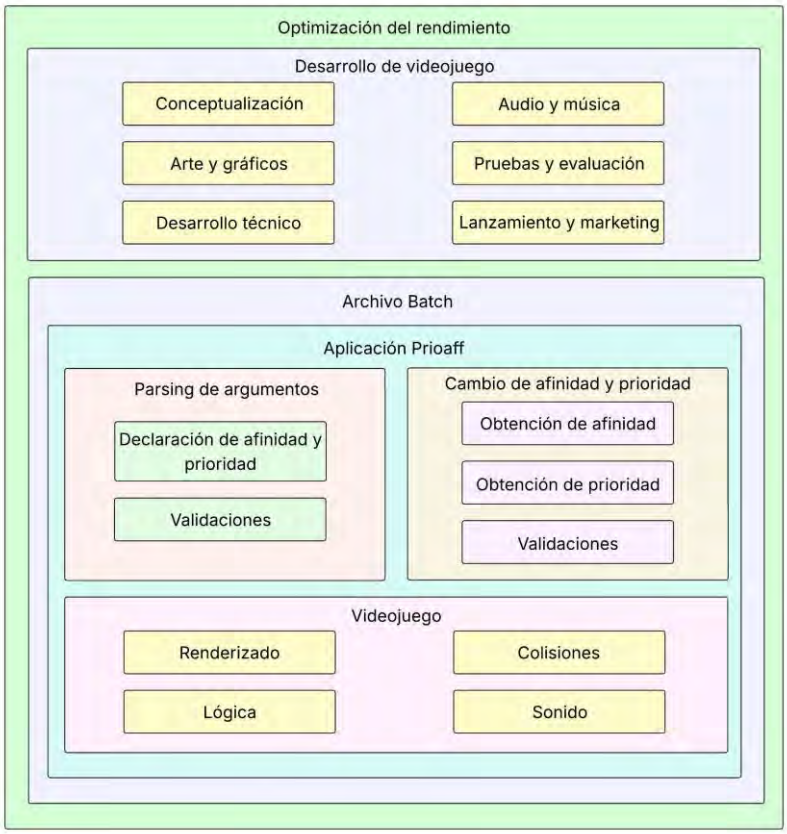

Figura 1. Diagrama de bloques sobre metodología del desarrollo del videojuego y del Script Batch, elaboración propia.

Entonces, se ejecuta el archivo con los tres parámetros correspondientes: ruta del videojuego, número de prioridad y número de afinidad. Se hace la declaración de variables, según la cantidad que se especificó en su momento; no se pueden usar 6 núcleos si la PC solo cuenta con 4.

Pasando las validaciones, se crea el proceso para cambiar la prioridad; en caso de no hacerse el cambio, el archivo no procederá con el siguiente paso; por el contrario, si se hace el cambio, ya creará el proceso para cambiar la afinidad.

El diagrama de la Figura 2 representa el proceso lógico que se lleva a cabo en la aplicación Prioaff para ejecutar el videojuego con una afinidad y prioridad determinada; es decir, ahora el videojuego forma parte del proceso de la aplicación Prioaff, siendo este el último paso.

Es importante notar cómo al principio del diagrama se inicia con un bloque denominado “Ejecución de archivo batch”. Este bloque tiene la función de facilitar todo el proceso de ejecución, debido a que dicho archivo

establece los parámetros necesarios para llevar a cabo la ejecución de la aplicación Prioaff. En realidad, el procedimiento es sencillo, pero resulta ser altamente funcional.

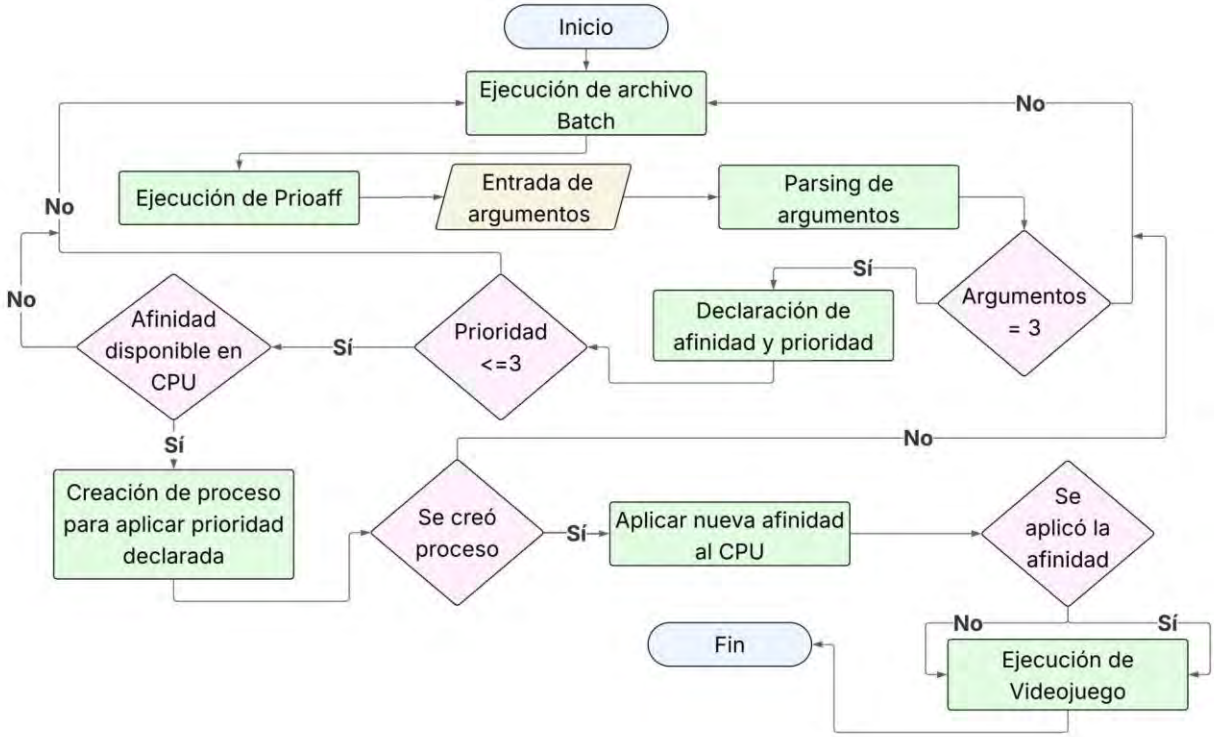

Figura 2. Diagrama de flujo de aplicación Prioaff, elaboración propia.

Entorno de prueba y especificaciones de hardware

Las pruebas de la ejecución del juego fueron realizadas en los siguientes equipos:
(Equipo A) Equipo gama baja Laptop Thinkpad: Sistema operativo: Windows 10, procesador Intel(R) Core(TM) i5-4300U (1.90 GHz min) (2.49 GHz max), cantidad y tipo de memoria RAM (8 GB, DDR3 4800 MT/s)
(Equipo B) Equipo gama baja/media Laptop Lenovo Yoga: Sistema operativo: Windows 11, Procesador: 12th Gen Intel(R) Core(TM) i7-1260P (2.10 GHz), GPU integrada Intel(R) Iris(R) Xe Graphics (1.40 GHz), Cantidad y tipo de memoria RAM (16 GB, DDR5 4800 MT/s)
(Equipo C) Equipo gama alta Escritorio: Sistema operativo: Windows 11, procesador: 14th Gen Intel(R) Core(TM) i5-14600K (3.50 GHz), GPU discreto NVIDIA 5070 Ti, cantidad y tipo de memoria RAM (32 GB, DDR5 4800 MT/s)

Configuración de las pruebas realizadas

Las corridas fueron divididas en cuatro partes, donde la primera es sin afinidad ni configuración de los núcleos a utilizar (multinúcleo). Posteriormente, con ayuda del archivo Batch, se cambió la afinidad para correr el videojuego con uno, dos y tres núcleos.

La afinidad se refiere a la relación que existe entre una aplicación y procesadores. Mientras que la prioridad es la importancia de una aplicación dentro de un sistema o entorno.

Herramientas e índices de desempeño

El archivo Batch trabajará junto con la aplicación Prioaff y el videojuego para realizar un cambio de afinidad y prioridad a la hora de ejecución. La intención es optimizar el rendimiento del videojuego; buscando que la carga sea en menos hilos para mejorar la eficiencia, ayudando con el aumento de FPS, porque es mejor contar con dos núcleos estables que con cuatro saturados (Alseqyani, 2023).

Se analiza el impacto de estos ajustes en parámetros como latencia, estabilidad y uso de GPU, verificando que no se generen cuellos de botella ni conflictos con la CPU (Wang et al., 2016).

La generación estándar de fotogramas son 60 FPS (Gràcia Andreu, G., 2023), lo cual asegura que los movimientos del jugador y los escenarios resulten más naturales. Esto puede depender de las capacidades del equipo,

como es el procesador, memoria RAM y cómo aprovecha los recursos, así como equipos modernos que tienen GPUs o tarjetas gráficas integradas que permiten renderizar las imágenes del videojuego para suavizar los movimientos.

Por otro lado, el frametime va de la mano con los FPS; este es el tiempo en el que se generan los fotogramas. Este, a pesar de fluctuar, debe de ser lo más estable posible para mantener un estable flujo de los FPS para desplegar en el monitor; si se introduce inestabilidad, como lo puede ser saltos entre núcleos o una saturación sistémica del CPU, va a haber una cantidad no equilibrada de FPS mostrados al usuario, lo cual muestra una imagen irregular y poco estable (Liu, S., Kuwahara, A., Scovell, J. J., & Claypool, M., 2023).

Resultados

Los resultados obtenidos en tiempo real con MSI Afterburner son mostrados en el Cuadro 1 y Cuadro 2.

Parámetro de desempeño	Nivel 1 - Equipo A	Nivel 2 - Equipo A	Nivel 1 - Equipo B	Nivel 2 - Equipo B	Nivel 1 - Equipo C	Nivel 2 - Equipo C
Gama	Baja	Baja	Baja/Media	Baja/Media	Alta	Alta
Utilización GPU	53%	59%	35%	57%	2%	3%
Memoria utilizada de la GPU	210 mb	232 mb8	1544 mb	1777 mb	975 mb	1114 mb
CPU 1	79%	88%	62%	63%	30%	28%
CPU 2	36%	28%	18%	5%	0%	0%
CPU 3	-	-	42%	11%	1%	14%
CPU 4	-	-	4%	7%	0%	0%
Utilización total de la CPU	57%	58%	12%	15%	5%	5%
Utilización de memoria RAM	3496 mb	3747 mb	9008 mb	12290 mb	5713 mb	6119 mb
FPS	53 fps	27 fps	58 fps	56 fps	59 fps	59 fps
Frametime	39.8 ms	78.1 ms	33.2 ms	67.3 ms	18.7 ms	20.8 ms

Cuadro 1. Comparación de datos extraídos vía MSI Afterburner, 1 núcleo asignado al proceso.

Parámetro de desempeño	Nivel 1 - Equipo A	Nivel 2 - Equipo A	Nivel 1 - Equipo B	Nivel 2 - Equipo B	Nivel 1 - Equipo C	Nivel 2 - Equipo C
Gama	Baja	Baja	Baja/Media	Baja/Media	Alta	Alta
Utilización GPU	81%	90%	28%	43%	2%	5%
Memoria utilizada de la GPU	469 mb	468 mb	1443 mb	1528 mb	1071 mb	981 mb
CPU 1	62%	83%	46%	58%	1%	6%
CPU 2	47%	77%	0%	0%	0%	27%
CPU 3	-	-	0%	68%	0%	23%
CPU 4	-	-	54%	0%	0%	1%
Utilización total del CPU	55%	80%	13%	26%	3%	4%
Utilización de	3984 mb	4180 mb	9193 mb	10378 mb	5930 mb	6087 mb

memoria RAM						
FPS	41 fps	24 fps	59 fps	54 fps	60 fps	60 fps

Cuadro 2. Comparación de datos extraídos vía MSI Afterburner, con funcionalidad multinúcleos asignada al proceso.

En el Cuadro 1, del Equipo A se puede apreciar que la no contención ayuda a generar alrededor de 20-30% más fotogramas por segundo, comparado con el Cuadro 2, esta generación adicional ayuda a mejorar el frametime y fluidez del movimiento. Sin embargo, la latencia presenciada por el jugador aumenta en un 0.5% proporcional al porcentaje de ganancia en FPS. En el caso del Equipo B se visualiza que los beneficios son menores en el Cuadro 1 en comparación con el Cuadro 2, la arquitectura de la CPU facilita la generación de FPS en conjunto con la GPU, pero se reduce la carga total de la misma, en ocasiones en un casi 40%, no solo mejorando la estabilidad del frametime, sino también el consumo de energía. En el equipo C vemos en ambos cuadros que los componentes dan una estabilidad plena en FPS, frametime y también en la latencia presenciada por el jugador, esto gracias a la velocidad base de los núcleos y su cantidad.

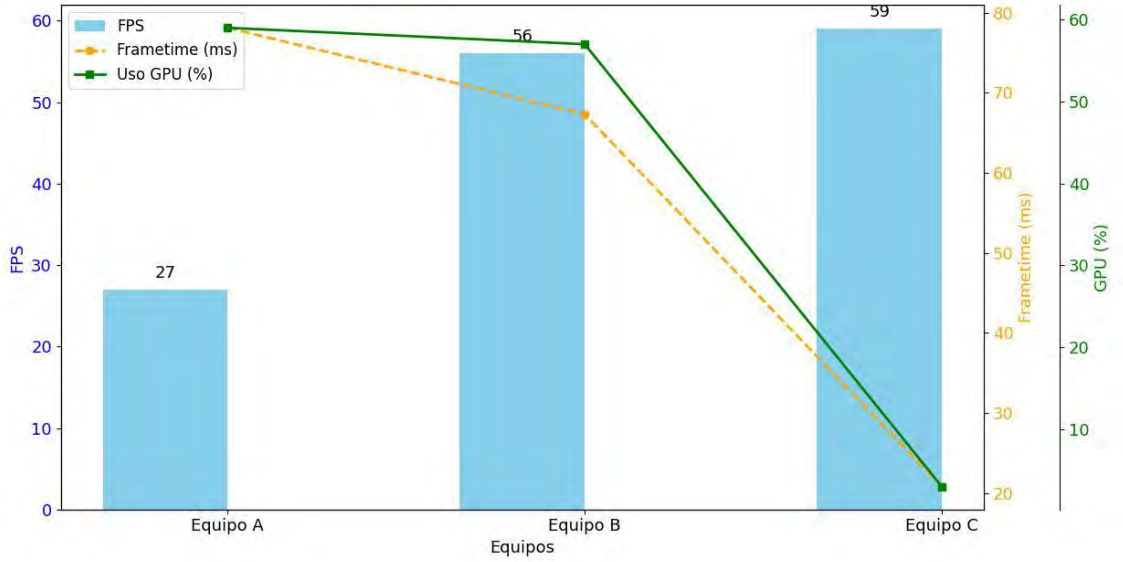

Figura 3. Rendimiento con 1 núcleo asignado (Nivel 2). Elaboración propia.

Análisis

A partir de los resultados obtenidos, se puede identificar que hay estabilidad de FPS, por medio de la afinidad de núcleos en CPUs que cuentan con 1 a 2 núcleos únicamente (1 o 2 núcleos físicos y 2 o 4 hilos lógicos). Esto se debe a la omisión de contención en el proceso, a lo que se alude con no hacer uso de contención, que es deshabilitar la asignación dinámica por decisión del planificador (sistema que se encarga de distribuir cargas en el sistema operativo) o de la aplicación, para que el proceso se albergue en uno u otro núcleo de la CPU, dependiendo de qué tan demandante es el programa.

En procesadores con pocos núcleos, es más beneficioso asignar afinidad fija para evitar saltos entre núcleos. Esto, sin embargo, no viene sin un costo asociado, a pesar de que se presentaron generaciones más abundantes de FPS, debido a que ya no hay saltos entre núcleos que pierdan la construcción de fotograma; el frametime (tiempo en el que se genera un fotograma) se incrementó en cantidad la latencia, esto a pesar de que no es deseable en casos en los cuales se tiene una latencia muy elevada. Por otro lado, para el programa de Tonatiuh, se refleja un beneficio en la estabilidad de generación de FPS.

Conclusiones

En conclusión, el desarrollo e implementación del archivo .bat es un recurso importante para CPUs que cuenten con 1 a 2 núcleos físicos únicamente; este está desarrollado en DOS y permite realizar cambios en el hardware

a través de la interfaz de línea de comandos. Se afirmó por medio de la extracción de porcentajes de utilización de los recursos computacionales, en equipos de gama baja, que hay una mejoría importante, aunque haya una compensación que debe tomarse en cuenta. Se encontró similitud con los procesos de postprocessing que se han hecho tan populares y un gran producto en el mercado de los videojuegos, como DLSS, FSR, XeSS o PSSR, los cuales facilitan la generación de FPS a costa de latencia presenciada en la respuesta de acciones efectuadas por el jugador.

Los resultados presentados en la figura 3, confirman que un nivel de afinidad a un único núcleo hace que se reduzcan los FPS y aumente el frametime en equipos de baja gama (así como es el caso del equipo A), jugando en contra de la experiencia de juego. A su vez, esta técnica permite, en equipos con hardware muy limitado, realizar un mejor manejo de los recursos y no gestionar mal estos últimos, evitando saturaciones y mejorando la estabilidad con respecto a aquellas configuraciones en las que se utilizan indiscriminadamente múltiples núcleos. En cambio, en los equipos de alta gama, así como el equipo C, el impacto es mínimo, por lo que la afinidad es completamente aplicable con independencia de las características de la máquina, lo que permite confirmar parcialmente la hipótesis formulada.

Limitaciones

El presente trabajo se enfocó exclusivamente en la automatización y asignación de núcleos junto con la prioridad a los procesos del videojuego mediante un archivo Batch, dejando fuera el desarrollo de una posible interfaz gráfica de usuario que facilite aún más su uso. La decisión actual permite centrarse en validar la efectividad técnica, sin embargo, se reconoce que la ausencia de una interfaz gráfica limita su accesibilidad para los usuarios que no tengan experiencia en entornos de consola; por ello se deja abierta la posibilidad de en un futuro implementar una GUI que simplifique la interacción con la herramienta.

Recomendaciones

Cabe destacar que el archivo Batch se puede incorporar a funcionalidades adicionales como la finalización automática de tareas en segundo plano o limpieza de los archivos temporales antes de iniciar el juego; estas acciones podrían contribuir a tener una mayor estabilidad y rendimiento del sistema. Las extensiones no se abordaron en este estudio, pero se representan líneas de trabajo a futuro, las cuales complementarán de forma útil para que funcionen en el enfoque actual de automatización.

Referencias

Aragón-Jurado, J. M., de la Torre, J. C., Ruiz, P., & Dorronsoro, B. (2024). *Optimización automática del videojuego Doom para un rendimiento óptimo en Steam Deck*. Escuela Superior de Ingeniería, Universidad de Cádiz.

Alseqyani, A. N. (2023). History and Future Trends of Multicore Computer Architecture. *International Journal of Computer Graphics and Animation*, *13*(2), 01–08. https://doi.org/10.5121/ijcga.2023.13201

García, J. M. B., Moreno, L. F. V., & Cerecer, E. R. O. (2023). ANÁLISIS DE RENDIMIENTO ENTRE LINUX Y WINDOWS EN LA EJECUCIÓN DE VIDEOJUEGOS UTILIZANDO DIFERENTES NIVELES DE HARDWARE. *Revista Digital de Tecnologías Informáticas y Sistemas*, *7*(1), 9-14.

Chen, W.-M., Cheng, S.-W., & Hsiu, P.-C. (2019). A User-Centric CPU-GPU Governing Framework for 3-D Mobile Games. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *38*(5), 961–974. https://doi.org/10.1109/TCAD.2018.2834404

Geris, A., Cukurbasi, B., Kilinc, M., & Teke, O. (2024). Balancing performance and comfort in virtual reality: A study of FPS, latency, and batch values. Software: Practice and Experience, 54(12), 2336-2348.

Gràcia Andreu, G. (2023). *Uso de metodologías de diseño actuales en la elaboración de un videojuego para la primera generación de consolas portátiles de 32 bits (Game Boy Advance)* [Trabajo de fin de grado, Universitat Politècnica de Catalunya]. Facultat d'Informàtica de Barcelona.

Koulaxidis, G., & Xinogalos, S. (2022). Improving mobile game performance with basic optimization techniques in unity. *Modelling*, *3*(2), 201-223.

Liu, S., Kuwahara, A., Scovell, J. J., & Claypool, M. (2023). The Effects of Frame Rate Variation on Game Player Quality of Experience. *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 1–10. https://doi.org/10.1145/3544548.3580665

López Vélez, M. J., & Villamar Saltos, L. A. (2022). *Optimización de rendimiento del videojuego Mi Bosque 3D* [Proyecto integrador, Escuela Superior Politécnica del Litoral]. Facultad de Ingeniería en Electricidad y Computación, ESPOL.

Qi, Z., Yao, J., Zhang, C., Yu, M., Yang, Z., & Guan, H. (2014). VGRIS: Virtualized GPU resource isolation and scheduling in cloud gaming. *ACM Transactions on Architecture and Code Optimization (TACO)*, *11*(2), 1-25.

Wang, P.-H., Li, C.-H., & Yang, C.-L. (2016). Latency sensitivity-based cache partitioning for heterogeneous multi-core architecture. *Proceedings of the 53rd Annual Design Automation Conference*, 1–6. https://doi.org/10.1145/2897937.2898036