## INTRODUCCIÓN

El desarrollo de videojuegos ha evolucionado hacia sistemas de alta complejidad técnica, generando la idea generalizada de que su creación requiere dominar conocimientos complejos y motores comerciales especializados, como Unity o Unreal Engine, los cuales presentan curvas de aprendizaje pronunciadas que limitan la innovación en entornos académicos e iniciales.

Para buscar solución a este problema, el presente documento detalla el desarrollo de ModeX, un sistema arcade modular implementado en Python utilizando las bibliotecas Pygame y Arcade. Se optó por el uso de Python ya que es un lenguaje con una sintaxis accesible y un ecosistema especializado, lo que permite un desarrollo iterativo más eficiente que en lenguajes compilados.

La investigación se fundamenta en tres dimensiones: teórica, validando la viabilidad de lenguajes interpretados en el desarrollo de aplicaciones de tiempo real; práctica, brindando asequibilidad y accesibilidad que elimina barreras económicas y técnicas para estudiantes y desarrolladores independientes; y metodológica, aportando un conjunto de patrones arquitectónicos documentados para Python.

Se postula como hipótesis que es técnica y económicamente viable el desarrollo de videojuegos 2D profesionales basados en Python empleando las bibliotecas Pygame y Arcade y que es posible alcanzar un rendimiento de 60 FPS mediante técnicas de optimización aun cuando los lenguajes interpretados tengan limitantes en el desarrollo de aplicaciones de tiempo real.

El objetivo general consiste en desarrollar un sistema arcade modular en Python verificando su viabilidad con datos de rendimiento. Los objetivos específicos que guían el proceso son:

- 1. Diseñar una estructura modular que permita integrar varios juegos.
- 2. Implementar técnicas de optimización alcanzando los 60 FPS promedio.
- 3. Desarrollar cuatro títulos representativos de géneros arcade clásicos.
- 4. Elaborar documentación técnica detallada que facilite la replicabilidad del proyecto.

El presente documento tiene la siguiente estructura: la sección de Marco Teórico aborda los fundamentos técnicos de programación gráfica 2D; la Metodología detalla el uso de la metodología RAD en el proyecto; en Desarrollo se describe la arquitectura de cada módulo; finalmente, el Análisis de Resultados muestra una evaluación de viabilidad del enfoque propuesto.

## PLANTEAMIENTO DEL PROBLEMA

El desarrollo de videojuegos se ha convertido en una parte prominente de la economía, no sólo para los propietarios de las compañías, sino también para la economía global en general. Según el reporte Video Games in the 21st Century: The 2020 Economic Impact Report, el sector aporta más de 90.3 mil millones de dólares en producción económica anual, sostiene aproximadamente 429,000 empleos directos, indirectos e inducidos, y ofrece una compensación promedio superior a los 121,000 dólares anuales por trabajador (Entertainment Software Association [ESA], 2020). Sin embargo, persiste una barrera importante: la percepción de que para la creación de videojuegos se necesita del dominio de conocimientos complejos, como programación orientada a objetos (POO) o técnicas de renderizado avanzadas, y motores de desarrollo comerciales.

Ante este escenario, se plantea la siguiente pregunta: ¿es técnicamente viable el desarrollo de videojuegos 2D profesionales a través del uso de Python con bibliotecas de código abierto para lograr un rendimiento óptimo? El presente estudio busca validar esta viabilidad mediante el desarrollo de un sistema arcade modular con arquitectura escalable y técnicas de optimización, probando que lenguajes interpretados pueden democratizar el desarrollo sin sacrificar calidad.

Esta problemática afecta directamente la democratización del desarrollo de videojuegos, ya que, los motores profesionales presentan curvas de aprendizaje pronunciadas que demandan comprensión de ecosistemas complejos: herramientas heterogéneas, editores especializados, sistemas propietarios y flujos de trabajo que se diferencian claramente entre plataformas. El dominio de motores profesionales requiere habilidades técnicas avanzadas que limitan la participación de desarrolladores autodidactas, educadores y creadores con formación técnica básica, limitando el acceso a quienes poseen recursos especializados.

Esta situación evidencia que la dependencia actual de herramientas complejas conlleva a una situación en la cual esta restricción limita el potencial creativo y, además, sistemáticamente excluye a las partes más desfavorecidas de la población con acceso restringido al hardware especializado o capacidades técnicas en desarrollo. Para democratizar el desarrollo, se debe buscar alternativas tecnológicas que reduzcan la complejidad inicial sin sacrificar calidad, permitiendo que el factor decisivo sea la creatividad, no el dominio de ecosistemas tecnológicos especializados.

**JUSTIFICACIÓN**

El desarrollo del sistema ModeX busca democratizar el desarrollo de videojuegos eliminando barreras técnicas y económicas que limiten la participación de desarrolladores en economías dentro del entretenimiento digital. Según Naciones Unidas (2023), los países en desarrollo enfrentan "barreras para el despliegue [tecnológico], incluyendo altos costos de infraestructura, asequibilidad de dispositivos, suministro eléctrico poco confiable y restricciones regulatorias" (párr. 12, traducción propia), limitando el acceso equitativo a herramientas de desarrollo digital.

El proyecto beneficia directamente a tres sectores: instituciones educativas que requieren herramientas accesibles para planes de estudio en desarrollo de videojuegos; desarrolladores independientes que necesitan validar prototipos sin inversión inicial; y comunidades autodidactas que buscan recursos documentados de implementaciones reales.

La aplicación adaptada de RAD para desarrollo de videojuegos representa una contribución metodológica significativa, dado que la metodología tradicional fue diseñada para aplicaciones empresariales con requisitos definidos, mientras que videojuegos requieren iteración constante sobre mecánicas emergentes durante el proceso creativo.

La documentación técnica generada establece un protocolo replicable para evaluación de viabilidad técnica de Python en dominios de aplicación, tradicionalmente dominados por lenguajes de sistemas: técnicas de optimización de rendimiento, física o lógica de juego y análisis de consumo de memoria mediante trazas que evidencian patrones de asignación/liberación.

Considerando los medios proporcionados por la metodología RAD, así como los recursos humanos, medios técnicos y el tiempo disponible, el proyecto se manifiesta como totalmente pertinente y adecuado. Al realizarse en tecnologías de código abierto, como Python, con comunidades activas y más aún al trabajar con hardware convencional, podrá servir como referencia mediante la publicación de su documentación técnica, facilitando su posible replicación.

Adicionalmente el proyecto establece el precedente de que es posible crear productos comercializables utilizando exclusivamente herramientas gratuitas, desafiando la narrativa predominante que asocia calidad con costos de software.

## HIPÓTESIS

Se postula que es técnicamente viable el desarrollo de calidad profesional de videojuegos bidimensionales con Python como lenguaje base y sus bibliotecas Pygame y Arcade, alcanzando los 60 FPS sin el uso de motores comerciales, demostrando que con técnicas de optimización, como sprite batching, culling espacial, object pooling, se pueden superar los impedimentos de lenguajes interpretados para poder desarrollar un videojuego funcional y de calidad que satisfagan estándares de jugabilidad y experiencia de usuario.

## OBJETIVO GENERAL

Diseñar, implementar y validar empíricamente un sistema arcade modular compuesto por cuatro videojuegos 2D funcionales, implementados en Python mediante las bibliotecas especializadas Pygame y Arcade, aplicando principios de ingeniería de software (programación orientada a objetos, patrones de diseño, arquitectura de código modular, separación de responsabilidades) y técnicas de optimización de rendimiento, con el propósito de demostrar la viabilidad técnica, económica y metodológica de crear videojuegos comercializables sin dependencia de motores comerciales.

## OBJETIVOS ESPECÍFICOS

- Desarrollar (Aplicar) cuatro videojuegos bidimensionales con diferentes mecánicas que evidencien la versatilidad técnica de Python.
- Diseñar (Crear) una estructura modular que permita integrar varios juegos y maximice la reutilización de código.
- Optimizar (Analizar) el rendimiento del sistema para garantizar ejecución fluida mediante técnicas específicas
- Documentar (Evaluar) exhaustivamente el proceso de desarrollo, decisiones arquitectónicas y soluciones a problemas técnicos, generando conocimiento transferible que sirva como referencia para futuros proyectos de desarrollo de videojuegos.

**TIPO DE INVESTIGACIÓN**

De acuerdo con la tipología propuesta, la presente investigación es de tipo tecnológico de alcance descriptivo-correlacional y aplicada. En consecuencia, se desarrolla un producto tecnológico que soluciona los problemas de la creación de videojuegos utilizando Pygame, otorgando herramientas prácticas para su uso en entornos educativos e independientes. Asimismo, es de tipo descriptivo, ya que se detalla el rendimiento del sistema con datos técnicos de videojuegos durante las sesiones de juego tales como el framerate, uso de memoria, tiempo de respuesta y la estabilidad del sistema bajo condiciones de operación auténticas.

El diseño metodológico que se seguirá para este estudio es de naturaleza evaluativa-experimental, ya que se construirá en la práctica del playtesting y se complementará con un análisis de datos en tiempo real. En este conjunto, la validación empírica se recopila a través de sesiones de juego cuidadosamente administradas por usuarios que representan al público objetivo de manera significativa. Durante estas sesiones los usuarios prueban el juego, mientras que el sistema recopila una cantidad considerable de datos y medidas de rendimiento de forma autónoma y automática que servirán para las conclusiones de viabilidad del proyecto, sin interferir con la experiencia del usuario en el videojuego.

Metodológicamente, la investigación adopta un enfoque predominantemente cuantitativo como paradigma, con observaciones cualitativas secundarias que solo enriquecen la interpretación de los resultados sin constituir el núcleo analítico. La secuencia generalmente secuencial-deductiva ordena la investigación; es decir, se formulan hipótesis específicas y, posteriormente, estas se validan con la recolección sistemática de evidencia numérica obtenida durante las sesiones de juego.

En este sentido, resulta evidente que este diseño metodológico cuantitativo-evaluativo permite una validación empírica de la viabilidad técnica de Python para el desarrollo profesional de juegos a través de una evidencia objetiva recogida en condiciones de uso reales. De hecho, la utilización de datos en tiempo real de playtesting con usuarios y de métricas técnicas automatizadas no solo permite tomar decisiones bien fundamentadas sobre el rendimiento, la usabilidad y la aplicabilidad del enfoque propuesto en entornos educativos y de desarrollo independiente, sino que también valida empíricamente la viabilidad técnica del enfoque propuesto.

**MARCO TEÓRICO**

Python se ha establecido como un lenguaje accesible y sencillo, aunque tradicionalmente considerado inadecuado para aplicaciones de tiempo real por su naturaleza interpretada, sin embargo, versiones modernas de Python incorporan optimizaciones de rendimiento que aportan mayor fluidez en la ejecución del videojuego. El uso de Python en el desarrollo de videojuegos se ha expandido más allá de scripting, encontrando aplicaciones en automatización de pruebas mediante la combinación de Behavior-Driven Development (BDD) con aprendizaje por refuerzo, permitiendo validación de mecánicas en proyectos basados en Python (Doria et al., 2024).

Pygame y Arcade son las dos opciones más conocidas para hacer juegos en Python. Pygame nació a principios de los 2000 como una herramienta gratuita para crear aplicaciones multimedia. Es fácil de aprender, tiene mucha documentación y una comunidad activa, por eso tantos desarrolladores la usan. (Hunt, 2023). Por otro lado, Arcade ofrece una API orientada a objetos con clases base específicas para entidades de juego que simplifican el desarrollo mientras mantienen capacidades de optimización avanzadas.

Investigaciones empíricas que analizan 27 lenguajes de programación utilizando benchmarks estandarizados demuestran que lenguajes compilados como C, C++ y Rust superan consistentemente a Python en tiempo de ejecución, con diferencias que pueden alcanzar órdenes de magnitud en tareas computacionalmente intensivas (Pereira et al., 2021). Por esta razón, para lograr un rendimiento óptimo en el videojuego se implementaron optimizaciones de rendimiento, un ejemplo fue la definición de __slots__, que modifica el modelo de almacenamiento subyacente utilizando una estructura de tamaño fijo, eliminando la sobrecarga del diccionario por instancia. La selección apropiada de estructuras de datos en Python es importante para el rendimiento de aplicaciones, ya que diferentes estructuras ofrecen distintas complejidades computacionales impactando directamente en la eficiencia de programas que manejan grandes volúmenes de datos o requieren respuesta en tiempo real (Lee & Hubbard, 2024).

Técnicas de optimización de rendimiento adicionales incluyen caching y memorización mediante functools.lru_cache para eliminar trabajo almacenando resultados de llamadas a función. El decorador lru_cache usa el caché Least Recently Used que guarda resultados de llamadas a funciones y retorna el resultado en caché cuando son los mismos argumentos, reduciendo tiempos de ejecución en funciones recursivas costosas (Python Software Foundation, 2024).

La aplicación de patrones de diseño en entornos educativos ha demostrado ser efectiva para enseñar principios fundamentales del diseño de juegos, permitiendo a estudiantes y diseñadores comprender soluciones para problemas relacionados con mecánicas de juego, diseño espacial y arquitectura de niveles (Barney, 2021).

Patrones como Singleton para managers globales y Manager para el manejo de eventos de entidades proporcionan estructuras fundamentales que previenen problemas arquitectónicos comunes y mejoran mantenibilidad del código. Dado que separamos los datos del comportamiento y esto permite una mejora del rendimiento y la escalabilidad, con el tiempo ECS se puede promover cualquier tipo de videojuego bidimensional. Un mapeo sistemático sobre el patrón Entity Component System revela que ECS constituye un patrón arquitectónico utilizado principalmente en la industria de videojuegos para implementar aplicaciones en tiempo real. (Voisard et al., 2025).

Pasando al uso de la metodología RAD en el videojuego, el análisis comparativo entre metodologías ágiles y enfoques estructurados tradicionales revela que las metodologías ágiles presentan tasas de éxito del 40% comparadas con solo 15% en proyectos que utilizan el modelo en cascada. Las metodologías ágiles destacan por facilitar cambios de requisitos en cualquier etapa del proyecto, priorizar la satisfacción del cliente, promover desarrollo iterativo y mantener interacción continua entre cliente y desarrolladores, características que resultan particularmente valiosas en proyectos con requisitos cambiantes como los videojuegos (Mishra & Alzoubi, 2023).

Es por esa razón que optamos por el uso de la metodología RAD. Esta metodología estructura el desarrollo en ciclos iterativos de 2-3 semanas que producen prototipos funcionales que se pueden evaluar. Este enfoque puede descubrir o mostrar al diseñador temprano los problemas de diseño, viabilidad técnica y duración estimada de la implementación. Mientras tanto, los artículos de investigación en diseño y desarrollo de serious games ofrecen enfoques que integran buenas prácticas en ingeniería de software, prácticas en la industria de videojuegos y factores de éxito en serious games. Estos enfoques incluyen evaluación iterativa de artefactos por diseñadores, expertos del dominio y jugadores desde etapas tempranas del desarrollo, lo que permite superar desafíos de diseño, mantener control sobre la evolución de requisitos y facilitar el diseño participativo (Ben Amara et al., 2024).

Además, se empleó el playtesting, término del Games User Research que se usa para evaluar decisiones de diseño en función de la retroalimentación que proviene de los jugadores para mejorar la experiencia de estos. La investigación sobre desafíos de playtesting en estudios indie revela que, aunque las mejores prácticas de GUR son bien conocidas, su implementación requiere recursos, conocimiento y experiencia que no siempre están disponibles para desarrolladores independientes. Los desafíos principales incluyen encontrar participantes apropiados para las pruebas y manejar efectivamente los datos recopilados durante las sesiones de playtesting (Denisova et al., 2024).

Derivado de este proceso, se elaboró la investigación reproducible que puede abordarse desde múltiples perspectivas, incluyendo el desarrollo de plataformas que promuevan reproducibilidad, la compartición de código fuente y la liberación de bibliotecas que proporcionen acceso a algoritmos relevantes en las disciplinas correspondientes (Colom et al., 2023). En el contexto específico del desarrollo de videojuegos, la documentación técnica exhaustiva constituye un elemento fundamental para facilitar la replicabilidad de proyectos y democratizar el conocimiento especializado. Esta documentación debe incluir especificaciones detalladas del entorno de desarrollo, versiones exactas de motores gráficos y bibliotecas utilizadas, configuraciones de hardware necesarias, procedimientos de compilación y ejecución, así como la descripción sistemática de decisiones arquitectónicas y de diseño implementadas. La disponibilidad pública de esta información técnica estructurada permite que desarrolladores independientes, estudiantes e investigadores puedan reproducir, estudiar y extender proyectos existentes sin depender exclusivamente del conocimiento de equipos originales, contribuyendo así a la formación de una comunidad más inclusiva. Por otro lado, esta práctica de documentación estricta también corre la posibilidad de validar descubrimientos de investigación aplicada en desarrollo de juegos y establecer normas en beneficio del ámbito académico y de la industria.

En síntesis, la viabilidad técnica de Python para videojuegos bidimensionales se fundamenta en la convergencia de tres pilares: capacidades de optimización del lenguaje mediante técnicas específicas de rendimiento, implementación rigurosa de patrones de diseño probados que garantizan escalabilidad arquitectónica, y adopción de metodologías ágiles como RAD que facilitan iteración rápida y validación continua. Esta base teórica proporciona el sustento conceptual necesario para evaluar empíricamente el rendimiento, eficiencia y productividad de Python en desarrollo de videojuegos mediante el enfoque metodológico cuantitativo propuesto.

## DESCRIPCIÓN DEL DESARROLLO E IMPLEMENTACIÓN DEL PROTOTIPO

A continuación, se presentan los recursos materiales y tecnológicos utilizados, la metodología adoptada y el cronograma de desarrollo del prototipo, el cual fue estructurado bajo la metodología de Desarrollo Rápido de Aplicaciones (RAD). Se optó por esta metodología debido a la naturaleza iterativa y cambiante del diseño de videojuegos, ya que permite evaluar y ajustar continuamente los componentes del sistema mediante ciclos cortos que producen prototipos funcionales.

Para el desarrollo del proyecto, los recursos materiales y tecnológicos necesarios fueron una PC portátil con las siguientes especificaciones: procesador Intel Core i5-8600, 8 GB de RAM, tarjeta gráfica Intel UHD 630. El entorno de desarrollo se configuró con Python 3.11.4, las bibliotecas Pygame y Arcade, Tiled Map Editor para diseño de niveles, Pytest para tests, PyInstaller para empaquetado del ejecutable y SQLite para persistencia de datos. Los estudiantes conformaron el equipo de trabajo para asumir los roles de desarrolladores, diseñadores y testers, según las fases del proyecto.

La coordinación del proyecto se llevó a cabo mediante la metodología RAD, utilizando ciclos iterativos de dos semanas como organización del trabajo. Al inicio de cada ciclo se definían los objetivos del sprint, y al cierre se evaluaban los entregables funcionales obtenidos. Los asesores del proyecto realizaron el monitoreo y validación de los avances, apoyo en la toma de decisiones de arquitectura y verificación del cumplimiento de los objetivos planteados.

La estrategia metodológica para el acopio, procesamiento y análisis de la información se estructuró en tres etapas. En la etapa de acopio se recopiló información de la documentación oficial de las bibliotecas utilizadas y artículos académicos sobre optimización de rendimiento. En la etapa de procesamiento, los datos recopilados fueron datos de rendimiento, decisiones de arquitectura y resultados de playtesting. Finalmente, en la etapa de análisis, la información fue interpretada mediante la documentación técnica generada, además se logró validar la hipótesis y generar conclusiones sobre viabilidad técnica de Python para el desarrollo de videojuegos 2D.

El 11 de septiembre de 2025 se dió inicio formal al desarrollo de ModeX Versión 1.0, estableciendo como objetivo principal la creación de un prototipo que integrara al menos dos videojuegos funcionales con mecánicas diferentes. Durante este período que duró hasta el 1 de octubre de 2025, el enfoque principal fue establecer la estructura base del proyecto.

El 12 de septiembre de 2025 se implementó la metaclase Singleton que asegura la instanciación única de clases críticas. El 20 de septiembre de 2025 se implementó un sistema de logging estructurado y automatizado para rastreo de eventos durante ejecución. Consecutivamente el 21 de septiembre de 2025 se desarrolló el módulo utils.paths, módulo clave en la estructura futura del proyecto, implementando funciones helper para resolución de rutas relativas a rutas absolutas.

Entre el 23 de septiembre y 1 de octubre de 2025 nos enfocamos en el desarrollo del primer videojuego del arcade: el simulador de carreras 2D llamado "Neon Rush". El 1 de octubre de 2025 se implementaron las siguientes clases: Track basada en carga de archivo TMX exportado desde Tiled Map Editor, clase Car heredando de arcade.Sprite representando a un carro de F1.

El 3 de octubre de 2025 se inició con el desarrollo de ModeX Versión 1.1 expandiendo el sistema con el segundo videojuego de género diferente: plataformas 2D llamado "Cosmic Odyssey" con mecánicas de exploración, combate y progresión por niveles. Este período de desarrollo que duró hasta el 3 de noviembre fue más complejo, ya que hubo más sistemas que interactuaban entre sí: física plataformera, cámara, gestión de niveles, enemigos y sistema de combate. Entre el 3 y 6 de octubre de 2025 se implementaron sus clases base: Player, Wall, Level y Camera.

El 12 de octubre de 2025 se implementaron e integraron los objetos y clases de estado al game loop principal del juego de plataformas: inicialización del motor de física de Arcade, instanciación de Camera, carga de niveles, música de fondo e instanciación de enemigos y plataformas. El 13 de octubre de 2025 se integró un sistema de lectura y carga de archivos JSON para configuración de niveles en el juego de plataformas.

Trabajando en el juego de plataformas, el 14 de octubre de 2025 se implementó la clase Wall representando barreras de circuito. El 17 de octubre de 2025 se implementó la clase RaceManager que maneja lógica de carrera: estados, vueltas, checkpoints, temporizador y cruce de meta. Entre el 18 y 21 de octubre de 2025 se desarrollaron vistas complementarias: WinView (mostrada al completar el juego en su totalidad), PauseView (mostrada al pausar el juego) y GameOverView (mostrada cuando se pierde en un nivel). Del 20-22 de octubre de 2025 se actualizó el juego de carreras en la Versión 1.1 integrando todos los elementos desarrollados en el juego.

Pasando al juego de peleas entre el 22 y 24 de octubre se implementaron clases de objetos: Actor con atributos de peleador básicos, Player (primer jugador) y Skeleton (segundo jugador), Entre el 24 y 31 de octubre de 2025 se implementó su bucle principal con la integración de sistemas.

El 3 de noviembre de 2025 se inició el desarrollo de ModeX Versión 1.2 con contenido adicional y preparación específica para presentación en el Concurso Local de Prototipos programado para el 25 de noviembre. Se priorizaron funciones con mayor impacto visual, mejoras de interfaz, gráfica, robustez ante errores de usuario y búsqueda e integración de icono oficial de ModeX.

Desarrollando el juego de plataformas, el 7 de noviembre de 2025 se actualizó el menú principal añadiendo selección entre tres jugadores antes de iniciar nivel. Entre el 8 y 10 de noviembre de 2025 se desarrollaron las mismas vistas que en el juego de carreras con imágenes propias. Entre el 10 y 11 de noviembre de 2025 se completó su respectiva actualización para la Versión 1.2.

Diseñando el juego de peleas, el 12 de noviembre de 2025 se buscaron y obtuvieron nuevos escenarios y personajes en menús para assets gráficos de este juego. Entre el 12 y 13 de noviembre de 2025 se completó la respectiva actualización para la Versión 1.2 balanceando personajes ajustando sus estadísticas e integrando nuevos personajes y escenarios.

El 12 de noviembre de 2025 se realizó la búsqueda de assets para cuarto juego del sistema: shooter cooperativo de zombies con mecánicas de defensa de oleadas. Continuando con el juego de shooter, el 14 de noviembre de 2025 se implementaron objetos: clases Bullet, Camera, Particle, Player, PowerUp, PowerUpMachine y Zombie proporcionando componentes necesarios para funcionalidad. El 19 de noviembre de 2025 se implementó su bucle principal manejando sistemas de oleadas: timer de oleada y tipos de zombies introducidos progresivamente en la oleada. El 24 de noviembre de 2025 se creó la primera versión ejecutable empaquetando el proyecto como archivo .bat, ejecutando el punto de entrada principal del videojuego. Esta versión (ModeX V1.2) fue la primera versión completamente estable y presentable públicamente.

El 18 y 20 de diciembre de 2025, comenzó el desarrollo de ModeX Version 1.3, que se centró en mejoras identificadas por los jueces: las vistas de inicio se actualizaron y se implementó que la Ventana fuera resizable. La arquitectura del programa se mejoró con optimizaciones de rendimiento y se implementó la persistencia de datos estructurada con una base de datos local.

Entre el 19 y 30 de diciembre de 2025 ejecutamos una actualización total del juego de carreras implementando sistema completamente rediseñado: adición de 2 pistas, mejora de arquitectura con optimizaciones de rendimiento, implementación de escalabilidad de objetos y mejora de diseño de pista ya existente. Entre el 22 y 30 de diciembre de 2025 se implementó módulo de estado de juego utilizando base de datos SQLite para persistencia estructurada de datos.

El 1 de enero de 2026 se comenzó con el desarrollo de ModeX Versión 1.4 enfocándose en optimizaciones de rendimiento y adición de contenido de juego de plataformas. Durante toda esta versión se implementaron optimizaciones de rendimiento globales aplicando técnicas como: object pooling para balas y partículas, sprite batching para agrupación de renderizado reduciendo draw calls y viajes al GPU, culling espacial omitiendo actualización/renderizado de entidades fuera de viewport y spatial hash permitiendo evitar cálculos redundantes de objetos estáticos. El 14 de enero de 2026 agregamos botones interactivos a casi todas las vistas del juego facilitando navegación mediante el ratón complementando controles de teclado.

El 3 de febrero de 2026 iniciamos desarrollo de ModeX Versión 1.5 enfocándonos en finalización de contenido restante y preparación para concurso Estatal. Entre el 8-10 de febrero de 2026 integramos los dos juegos actualizados (peleas, shooter zombies) al arcade. Entre el 10-14 de febrero de 2026 se ejecutaron pruebas con usuarios recopilando información para documentación técnica: reclutamos 25 participantes representativos (jugadores casuales, personas sin experiencia gaming). El 22 de febrero de 2026 elaboramos la documentación técnica: arquitectura de sistema, diagramas de secuencia, análisis de rendimiento con gráficos, comparativas de métricas y análisis de complejidad de código. El 23 de febrero de 2026 creamos ejecutable .exe del arcade V1.5 empaquetando proyecto Python en ejecutable standalone mediante PyInstaller.

Este cronograma detallado documenta la evolución completa del proyecto ModeX desde su concepción inicial en septiembre de 2025 hasta su forma final en febrero de 2026, reflejando proceso iterativo de desarrollo, refinamiento basado en feedback y preparación rigurosa para competición académica con participación de equipo multidisciplinario coordinado mediante la metodología RAD (Rapid Application Development) adaptada a un entorno educativo.

**PROPUESTA DEL VALOR DEL PROTOTIPO**

La propuesta de valor de ModeX consiste en la democratización del conocimiento para desarrollar un videojuego utilizando tecnología asequible y al alcance de estudiantes en proceso de formación, desarrolladores de videojuegos principiantes y pequeños estudios.

La posibilidad de acceso y el ahorro financiero son los ejes cuantitativos del valor. Mientras motores comerciales como Unity requieren mínimo 8GB de RAM y recomiendan cantidades mayores para proyectos complejos (Unity Technologies, 2026), y Unreal Engine especifica requisitos recomendados de 32GB de RAM, procesadores quad-core de 2.5 GHz o superiores, y tarjetas gráficas con 8GB o más de VRAM (Epic Games, 2026), ModeX funciona eficientemente en hardware convencional (procesadores Intel Core i5 de octava generación, 8GB RAM, gráficos integrados), con instalación inferior a 300MB. Además, la documentación técnica de este proyecto es de fácil comprensión y entendimiento, permitiendo resultados en 4-6 semanas. Esta accesibilidad disminuye las barreras económicas y técnicas que han marginado a desarrolladores independientes y pequeños estudios del ecosistema de desarrollo de videojuegos.

Lo que diferencia a este proyecto es la innovación pedagógica y la experiencia de aprendizaje. ModeX, mediante su detallada documentación técnica, ofrece patrones de diseño y arquitectura de código, lo que posibilita el análisis, la comprensión y la duplicación de los componentes del sistema. Los patrones arquitectónicos que se pueden transferir quedan al descubierto gracias a la variedad de géneros aplicados (carreras, plataformas, shooter cooperativo y peleas).

La combinación estratégica de estos elementos presenta una oferta diferente que no busca competir directamente con motores comerciales, sino satisfacer a los segmentos desatendidos donde la accesibilidad y transparencia de conocimiento mediante documentación técnica como guía de replicación superan en valor a capacidades gráficas avanzadas o herramientas de productividad empresarial.  ModeX reconoce que el cliente requiere herramientas de aprendizaje fundamentadas en principios, no producto de producción optimizado para eficiencia comercial, posicionando al proyecto como puente hacia la comprensión profunda que posteriormente facilita el uso de herramientas profesionales.

**ESTUDIO DE VIABILIDAD PARA LA IMPLEMENTACIÓN DEL PROTOTIPO**

Este estudio determina si ModeX puede implementarse exitosamente para instituciones educativas, estudiantes en proceso de formación, desarrolladores de videojuegos principiantes y pequeños estudios. La evaluación examina si el proyecto es viable a nivel técnico, determina los riesgos que podrían surgir y sugiere medidas para reducir dichos riesgos.

La estabilidad y la experiencia del stack tecnológico escogido son la base de la viabilidad técnica de ModeX. Python es un lenguaje de programación que cuenta con una comunidad activa, ha sido desarrollado durante más de treinta años y tiene un uso extendido en centros educativos. Pygame cuenta con una historia de mantenimiento desde el año 2000 y también proporciona una documentación detallada para su uso. Mientras que Arcade ofrece una API moderna manteniendo acceso a directo a funciones primitivas de bajo nivel cuando se necesita. Esta combinación asegura que la documentación sea completa y que exista compatibilidad a largo plazo, lo que reduce el riesgo de obsolescencia. Los requisitos de hardware (procesador Intel Core i5 de octava generación, 8 GB de RAM y gráficos integrados Intel UHD 620/630) corresponden a especificaciones presentes comúnmente en equipos de gama media actuales, por lo que el hardware no representaría un impedimento para la replicabilidad del proyecto.

El análisis reconoce dos tipos de riesgo y sus correspondientes estrategias de mitigación. Los riesgos técnicos incluyen potencial incompatibilidad con configuraciones de hardware específicas (particularmente drivers gráficos obsoletos en equipos antiguos) mitigable mediante testing en hardware y documentación de prevención de problemas. Los riesgos institucionales consideran resistencia a adopción de herramientas no-comerciales por percepciones de menor prestigio profesional, mitigable mediante evidencia de transferibilidad de competencias y casos de éxito documentados de graduados tras formación con herramientas educativas similares.

La evaluación integral confirma la viabilidad técnica de ModeX para implementación en instituciones educativas, estudiantes en proceso de formación, desarrolladores de videojuegos principiantes y pequeños estudios. Con los factores de riesgo identificados y mitigables mediante estrategias preventivas documentadas, lo que fundamenta una probabilidad elevada de ejecución exitosa cumpliendo los objetivos establecidos de democratización del conocimiento en desarrollo de videojuegos.

**ESTUDIO DE FACTIBILIDAD TÉCNICA Y FINANCIERA (COSTOS) PARA SU PRODUCCIÓN E IMPLEMENTACIÓN**

El análisis de materiales tecnológicos seleccionados para ModeX evalúa propiedades fundamentales que determinan su idoneidad para el desarrollo propuesto. Python 3.11.4 presenta características técnicas favorables: gestión automática de memoria y tipado dinámico facilitando prototipado rápido. Pygame proporciona renderizado por hardware mediante OpenGL y cuenta con un sistema de eventos asíncrono, por otro lado, Arcade añade abstracciones de alto nivel.

El análisis financiero desglosa costos totales de producción en cuatro categorías principales. Costos de desarrollo humano**:** El proyecto contempla 2 estudiantes de ingeniería en sistemas prácticas profesionales durante 3 meses, sin costo directo al proyecto ya que forma parte de sus requisitos académicos de titulación. En términos de valor de mercado, 2 desarrolladores junior contratados durante 3 meses generarían un costo de $25,230-35,322 MXN considerando el salario promedio de $4,205-5,887 MXN mensuales para desarrolladores junior contratados en México, lo cual representa aproximadamente el 50-70% del salario promedio general de desarrolladores reportado por INEGI en $8,410 MXN mensuales (Secretaría de Economía de México., 2025). Costos de software y herramientas: $0 MXN dado uso exclusivo de herramientas open-source. Costos de assets: $0 MXN mediante uso de repositorios y/o creación de assets personalizados. Costos de hardware para consulta**:** un equipo de cómputo reacondicionado que cumple con los requisitos técnicos (Intel Core i5 de octava generación, 8GB RAM, Intel UHD Graphics 630) tiene un costo aproximado de $3,906 MXN en el mercado mexicano (PC ONE México, 2026), mientras que un kit básico de teclado y ratón inalámbrico tiene un costo de $363 MXN (Xataka México, 2025). Estos costos son referenciales para instituciones sin equipo, ya que ModeX funciona en hardware existente de especificaciones medias. Costo total de producción: $0 MXN con equipo existente, o $8,538 MXN si se requiere adquirir 2 equipos completos para 2 desarrolladores.

El análisis beneficio-costo muestra una relación favorable**:** los beneficios tangibles (ahorro en licencias y mayor alcance de capacitación) e intangibles (desarrollo de competencias, reducción de barreras de acceso y contribución al ecosistema educativo) superan ampliamente la inversión inicial de desarrollo. Esto confirma la viabilidad financiera del proyecto tanto como herramienta educativa gratuita como potencial producto comercial escalable de bajo costo marginal.

**IMPACTO SOCIAL, ECOLÓGICO, TECNOLÓGICO, Y/O DESARROLLO SUSTENTABLE**

ModeX es un proyecto con iniciativa tecnológica cuyo impacto va más allá de lo técnico, ya que contribuye al acceso educativo, al cuidado de recursos materiales y al desarrollo de habilidades en software. Su diseño es sostenible porque incluye responsabilidad ambiental, rendimiento técnico y beneficio para sus usuarios.

En promover el acceso al conocimiento sobre la creación de videojuegos, particularmente para alumnos, autodidactas y estudios pequeños con recursos escasos, se manifiesta el impacto social de ModeX. Su documentación técnica permite compartir conocimiento sin barreras económicas y contribuye al ámbito educativo. Además, establece un precedente de colaboración y apertura que fomenta una cultura de intercambio beneficiosa para el desarrollo tecnológico.

El impacto ambiental de ModeX es la funcionalidad del proyecto sin necesidad de adquirir nuevo hardware ni actualizar equipos. Sus requisitos técnicos (procesador de Intel Core i5-i7, 8 GB de RAM y gráficos integrados) son convencionales lo que ayuda a reducir el consumo de recursos para fabricar nuevos equipos y disminuir la generación de residuos electrónicos.

La influencia tecnológica de ModeX es que se trata de una oportunidad para compartir conocimientos técnicos a través de la documentación que amplía las optimizaciones de rendimiento, los patrones de diseño y la arquitectura del software. Incluye técnicas como detección de colisiones eficiente, uso de máquinas de estados para IA y optimizaciones de rendimiento (object pooling y sprite batching). Esto ayuda a entender los principios de rendimiento de optimización y la creación de videojuegos por completo, lo que permite un aprendizaje mucho más profundo que el uso de motores propietarios.

ModeX se ubica como un diseño sustentable de triple borde porque integra todas las dimensiones social, ambiental y tecnológica a modo de desarrollo sustentable. Permite bajar las barreras de entrada al conocimiento técnico reduciendo, así, los costos económicos. Evita nuevo hardware utilizando el existente y reduce la cantidad de basura electrónica, enseñando sobre optimización de performance, arquitectura de código y buenas prácticas. Así, el proyecto se convierte en el ejemplo de una construcción maravillosa con educación, el ecologismo y el desarrollo de habilidades colectivas en primer lugar.

**ESTRATEGIA PARA LA PROTECCIÓN INTELECTUAL DEL PROTOTIPO**

La presente sección describe las estrategias de protección de la propiedad intelectual del sistema ModeX, considerando los organismos reguladores competentes: el Instituto Mexicano de la Propiedad Industrial (IMPI), el Instituto Nacional del Derecho de Autor (INDAUTOR) y la Organización Mundial de la Propiedad Intelectual (OMPI), cuya base de datos internacional PATENTSCOPE fue consultada mediante Google Patents y para obtener acceso a las patentes.

Con el objetivo de encontrar proyectos similares y argumentar que el prototipo es original, se buscaron antecedentes en los portales oficiales recomendados. La búsqueda en el sistema MARCia del IMPI bajo el criterio "ModeX" como texto similar en la categoría Software no arrojó ningún resultado registrado que fuera similar, confirmando la disponibilidad del nombre para su futuro registro. La búsqueda en Google Patents reveló patentes relacionadas con sistemas de juego para apuestas y entretenimiento monetario multijugador, los cuales son diferentes de ModeX en propósito, arquitectura y público objetivo. Ningún resultado corresponde a un sistema arcade modular desarrollado en Python con documentación técnica de apoyo pedagógico.

ModeX es un sistema de software de entretenimiento que contiene cuatro videojuegos 2D con diferentes mecánicas, cuya diferencia con otros proyectos es la documentación técnica generada durante su desarrollo. Esta documentación, que incluye decisiones de arquitectura, patrones de diseño y análisis de rendimiento, es el componente de difusión educativa del proyecto, siendo el recurso que se pondrá a disposición de instituciones educativas, estudiantes y desarrolladores independientes. El sistema incluye código fuente propio, documentación técnica y assets gráficos originales, por lo que la vía de protección más adecuada es el registro de derechos de autor ante INDAUTOR bajo la categoría de Programa de Cómputo, complementado con el registro de la documentación técnica como Obra Científica y Tecnológica, de acuerdo con el catálogo oficial de obras registrables establecido por INDAUTOR.

En conclusión, la búsqueda de patentes confirma que ModeX es un desarrollo original sin registros similares en los sistemas consultados, lo que refuerza su viabilidad de protección intelectual. El proceso de registro formal ante INDAUTOR se llevará a cabo en la conclusión del proyecto, garantizando la autoría legal del sistema y su documentación técnica en beneficio del equipo desarrollador.

## ANÁLISIS DE RESULTADOS

El análisis de rendimiento se realizó con un sistema de medición automático integrado en el bucle principal de los 3 videojuegos con más problemas a largo plazo referentes al rendimiento. Para estas pruebas se reclutaron 25 personas que jugaron cada uno de los videojuegos mientras que el sistema de medición obtenía los datos de rendimiento. Durante cada sesión, se obtuvieron datos de desempeño cada dos segundos, los datos se almacenaron en un archivo CSV y fueron los siguientes: FPS instantáneo, FPS promedio, FPS mínimo histórico, percentil 1 y percentil 99 del FPS, desviación estándar, frame time (ms), consumo de memoria RAM, datos del GC, conteo de entidades, draw calls estimados y frecuencia de frame drops consecutivos. En total se realizaron 25 sesiones de medición por videojuego probado, con duraciones variables según el tiempo que el usuario haya jugado al videojuego.

**Cuadro 1. Estadísticas descriptivas del FPS instantáneo por videojuego**

| Juego | Media FPS | Mediana FPS | Desv. Est. | Mín. FPS | P1 (1%) | P99 (99%) | Frame Drops (%) |
|---|---|---|---|---|---|---|---|
| Platformer | 58.75 | 58.79 | 1.05 | 42.00* | 56.37 | 60.50 | 0.21% |
| Racing | 58.10 | 58.10 | 1.20 | 5.60‡ | 40.00‡ | 60.50 | 2.90% |
| Fight | 60.41 | 60.45 | 0.75 | 56.80 | 58.64 | 62.02 | 0.00% |

*Nota: * El valor mínimo de Platformer son frames capturados durante caídas de GC. ‡ Los valores extremos de Racing (FPS = 5.60 y 40.20) corresponden al momento de transición entre niveles.*

Los datos del Cuadro 1 muestran un rendimiento diferente según la complejidad de cada juego. El Fight muestra datos muy consistentes, con una media de 60.41 FPS y una desviación estándar de 0.75, estableciéndose en los 60 FPS a lo largo de las sesiones. El juego Platformer muestra un rendimiento estable en condiciones normales, con una media de 58.75 FPS y una desviación estándar de 1.05, lo que indica variabilidad moderada y controlada. El percentil 1 de 56.37 FPS confirma que el 99 % de los frames del Platformer se mantienen por encima de 56 FPS, cumpliendo con el criterio de calidad de experiencia de usuario definido en la hipótesis.

El juego Racing presenta la mayor dispersión en sus valores. Sin embargo, esta cifra es engañosa si no se tiene en cuenta el contexto: la mayor parte de la variabilidad de los datos viene de dos fuentes:

- (a) Spikes de transición de nivel: los FPS bajan momentáneamente a valores de 0.4 FPS cuando se carga el siguiente nivel, véase Anexo B para mayor entendimiento.
- (b) Sesiones con condiciones iniciales degradadas del OS: se registran FPS de arranque entre 5 y 48 FPS que se recuperan progresivamente en el videojuego.

Excluyendo estos eventos, la mediana del FPS en Racing es de 58.10, lo que nos dice que en condiciones normales el juego se desempeña dentro del rango esperado. Para mayor detalle sobre la carga computacional por juego y nivel, véase Anexo A.

**Figura 1. Distribución de FPS por videojuego (todas las sesiones)**

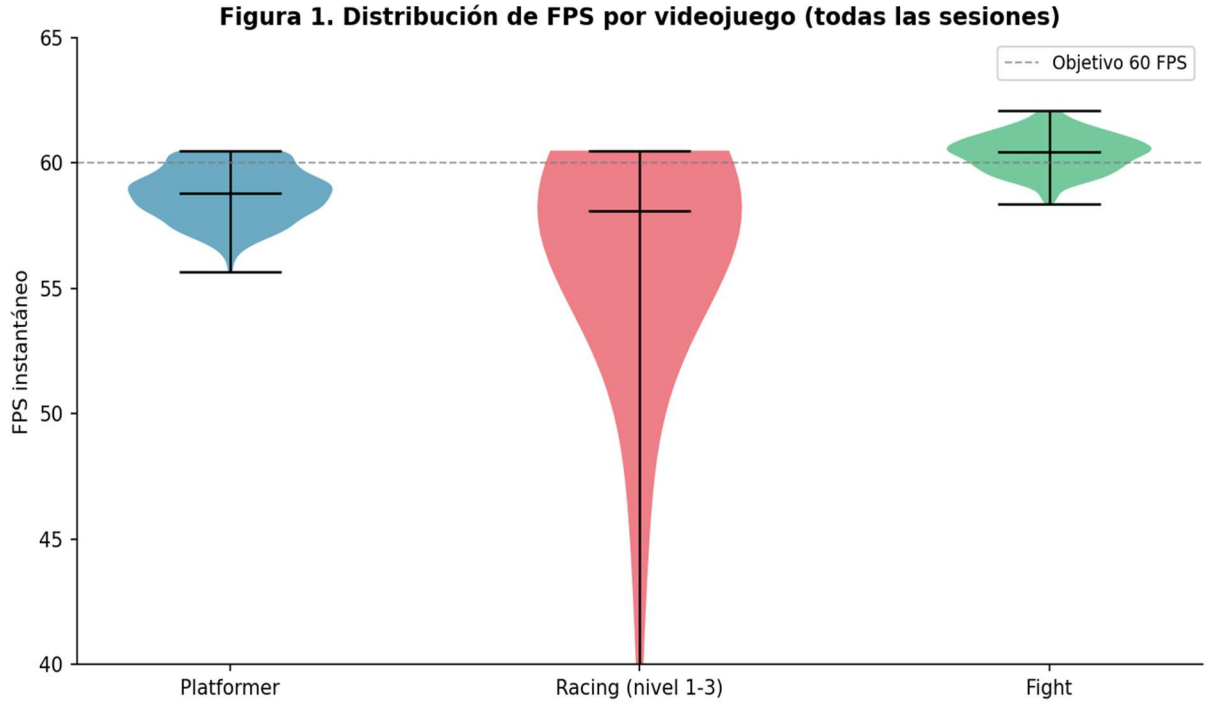


*Figura 1. Distribución FPS instantáneo por videojuego. La línea gris marca el objetivo de 60 FPS.*

La forma de cada violín en la Figura 1 nos ofrece información sobre concentración de valores y rango de variación. El del juego Fight es simétrico en torno a 60 FPS, lo que muestra la estabilidad señalada en la tabla anterior. El Platformer nos muestra un perfil igualmente concentrado, aunque con una pequeña cola inferior que muestra caídas causadas por el GC. El Racing presenta el perfil más amplio con una cola inferior extendida, mostrando los eventos atípicos ya mencionados

.

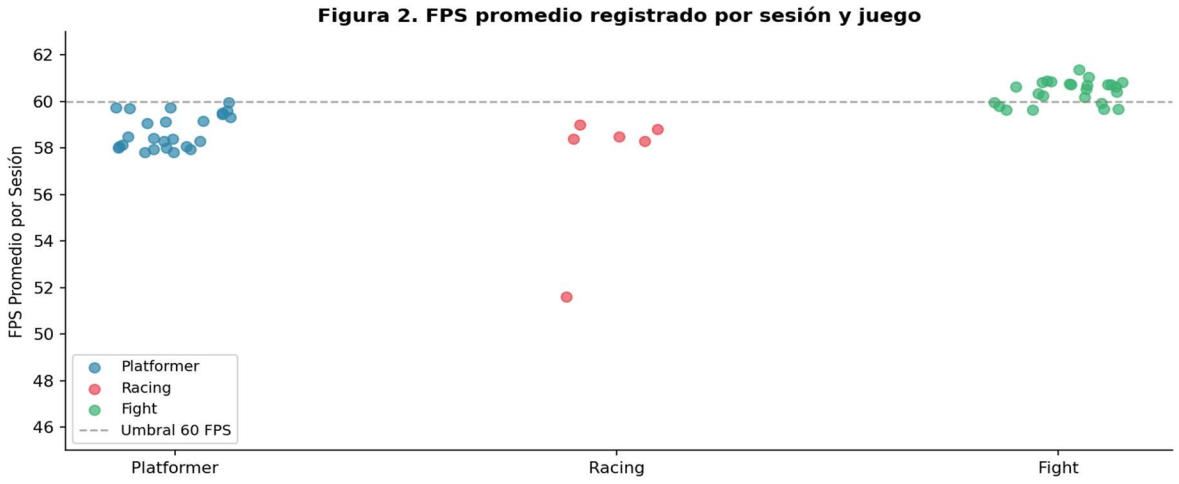


*Figura 2. FPS promedio por sesión y juego. La línea discontinua indica el umbral de 60 FPS.*

La Figura 2 muestra los FPS promedio por sesión para cada juego. En el juego Fight, las sesiones totales están cerca o por encima de los 60 FPS, sin sesiones por debajo de 59 FPS promedio. Las sesiones del Platformer se agrupan entre 57.5 y 60 FPS. Las sesiones de Racing muestran más dispersión, con promedios inferiores a 55 FPS por efecto de arranques degradados; no obstante, la concentración principal de sesiones de Racing también está en el rango 57–60 FPS. Un análisis extendido de la relación entre entidades activas y FPS promedio se presenta en el Anexo C.

Estos resultados son consistentes con lo señalado por Pereira et al. (2021), quienes documentaron las limitaciones de rendimiento de lenguajes interpretados frente a lenguajes compilados en tareas computacionalmente intensivas. Sin embargo, los datos obtenidos demuestran que con la implementación de técnicas de optimización de rendimiento es posible superar estas limitaciones en el entorno específico del desarrollo de videojuegos bidimensionales.

Los resultados del análisis de rendimiento validan la hipótesis del proyecto: los tres videojuegos evaluados alcanzan un promedio de FPS cercano o superior a 60 en hardware convencional, con un consumo de memoria RAM estable en 50 MB y ausencia de fugas de memoria en todas las sesiones. Para una comparativa visual entre estos datos véase el Anexo D.

**CONCLUSIONES**

El desarrollo de ModeX confirma que Python, con las bibliotecas Pygame y Arcade, es una plataforma técnicamente viable para la creación de videojuegos bidimensionales de calidad profesional sin el uso de motores comerciales complejos. Esto responde directamente al problema de investigación que se planteó: es posible reducir las barreras técnicas y económicas del desarrollo de videojuegos sin sacrificar la calidad del videojuego, siempre que se implementen técnicas de optimización de rendimiento adecuadas y una arquitectura de código modular bien establecida.

La hipótesis se confirma. Los datos de rendimiento recopilados durante 75 sesiones de juego (25 por título probado) demuestran que los tres videojuegos que se evaluaron tienen un promedio de FPS igual o cercano a 60 en condiciones normales de ejecución sobre hardware convencional (Intel Core i5 8.ª gen., 8 GB RAM, gráficos integrados). El juego Fight logró una media de 60.41 FPS, Platformer de 58.75 FPS y Racing de 58.10 FPS excluyendo los spikes de transición de nivel. Los únicos eventos atípicos identificados —congelamientos momentáneos cuando se carga un nivel en el juego de Racing— son limitaciones corregibles mediante carga asíncrona de recursos, sin afectar el desempeño general del sistema.

Los cuatro objetivos específicos se cumplieron satisfactoriamente. Se desarrollaron cuatro videojuegos con diferentes mecánicas (carreras, plataformas, peleas, shooter cooperativo) validando la versatilidad técnica de Python en distintos géneros. La arquitectura modular del código implementada —con clases base reutilizables (Item) y managers (RaceManager, LevelManager)— permitió compartir componentes entre todos los videojuegos, aprovechando así la reutilización de código. Las técnicas de optimización aplicadas (object pooling, sprite batching, culling espacial y spatial hashing) resultaron efectivas: el consumo de memoria RAM se mantuvo constante en todos los juegos y sesiones, sin detectar fugas de memoria. Finalmente, la documentación técnica generada —que incluye diagramas de arquitectura, decisiones de diseño y análisis de datos de rendimiento— constituye una guía replicable para proyectos similares en entornos educativos e independientes.

Más allá de los resultados cuantitativos, este proyecto tiene participación relevante en el entorno de desarrollo de videojuegos independiente y educativo. La accesibilidad de Python como lenguaje de programación, combinada con la disponibilidad gratuita de Pygame y Arcade, elimina dos de las barreras más frecuentes que enfrentan desarrolladores novatos: el costo económico de las herramientas y la curva de aprendizaje asociada a motores como Unity o Unreal Engine. ModeX demuestra que estas barreras no son obstáculos insuperables, sino que pueden ser abordadas con decisiones de diseño informadas y con un proceso de optimización sistemático. Esta conclusión es importante en entornos educativos, donde los recursos son limitados y el objetivo principal es el aprendizaje del proceso de desarrollo, no necesariamente la producción de un videojuego.

Como recomendaciones derivadas del análisis, se propone:

- (1) Implementar carga asíncrona de recursos en el juego de Racing para eliminar congelamientos en transiciones de nivel.
- (2) Controlar condiciones iniciales del sistema operativo antes de las sesiones de medición para reducir la variabilidad atribuible a estados externos y no al motor del juego.
- (3) Extender el sistema de automático de medición de datos de rendimiento para identificar cuellos de botella, permitiendo mayor fluidez en la ejecución del videojuego.
- (4) Ampliar el sistema ModeX para un quinto género, con el propósito de seguir poniendo a prueba los límites de la arquitectura modular en escenarios de mayor densidad computacional.

En conjunto, los resultados obtenidos respaldan la viabilidad de ModeX como herramienta pedagógica mediante su documentación técnica y validan el uso de Python como lenguaje base para el desarrollo de videojuegos 2D accesibles, funcionales y bien documentados. El proyecto no solo cumple con los objetivos planteados, también establece un precedente metodológico útil para futuros proyectos que busquen explorar el rendimiento de entornos de desarrollo en el ámbito del videojuego independiente.

**REFERENCIAS BIBLIOGRÁFICAS**

Barney, C. A. (2021). Application of pattern language for game design in pedagogy and design practice. *Information*, *12*(10), 393. https://doi.org/10.3390/info12100393

Ben Amara, B., Mhiri Sellami, H., & Ben Said, L. (2024). An approach for serious game design and development based on iterative evaluation. *Journal of Software: Evolution and Process*, *36*(5), e2680. https://doi.org/10.1002/smr.2680

Colom, M., Hernández, J. A., Kerautret, B., & Perret, B. (2023). Development efforts for reproducible research: Platform, library and editorial investment. En B. Kerautret et al. (Eds.), *Reproducible research in pattern recognition. RRPR 2022* (pp. 3–13). Springer. https://doi.org/10.1007/978-3-031-40773-4_1

Denisova, A., Bromley, S., Mirza-Babaei, P., & Mekler, E. D. (2024). Towards democratisation of games user research: Exploring playtesting challenges of indie video game developers. *Proceedings of the ACM on Human-Computer Interaction*, *8*(CHI PLAY), Article 343. https://doi.org/10.1145/3677108

Doria, D., Pereira, T., Bastos, A., Vale, T., & Figueiredo, E. (2024). A behavior-driven development and reinforcement learning approach for videogame automated testing. En *Proceedings of the ACM/IEEE 8th International Workshop on Games and Software Engineering* (pp. 73–79). ACM. https://doi.org/10.1145/3643658.3643919

Entertainment Software Association. (2020, 3 de diciembre). *Report: Video games contribute $90 billion+ to U.S. economy*. https://www.theesa.com/report-video-games-contribute-90-billion-to-u-s-economy/

Epic Games. (2026). *Hardware and software specifications for Unreal Engine*. Unreal Engine 5.7 Documentation. https://dev.epicgames.com/documentation/en-us/unreal-engine/hardware-and-software-specifications-for-unreal-engine

Hunt, J. (2023). Building games with Pygame. En J. Hunt (Ed.), *Advanced guide to Python 3 programming* (pp. 307–318). Springer. https://doi.org/10.1007/978-3-031-40336-1_21

Lee, K. D., & Hubbard, S. (2024). *Data structures and algorithms with Python: With an introduction to multiprocessing*. Springer. https://doi.org/10.1007/978-3-031-42209-6

Mishra, A., & Alzoubi, Y. I. (2023). Structured software development versus agile software development: A comparative analysis. *International Journal of System Assurance*

*Engineering and Management*, *14*(4), 1504–1522. https://doi.org/10.1007/s13198-023-01958-5

Naciones Unidas. (2023, 6 de octubre). *Widening digital gap between developed, developing states threatening to exclude world's poorest from next industrial revolution, speakers tell Second Committee* [Cobertura de reunión GA/EF/3587]. https://press.un.org/en/2023/gaef3587.doc.htm

PC ONE México. (2026). *PC – HP EliteDesk 800 G4 - i5 8va gen. 8 GB RAM, 240 GB SSD, SFF* [Catálogo de productos]. https://pconemexico.com.mx/products/pc-hp-elitedesk-800-g4-i5-8va-gen-8-gb-ram-480-gb-ssd-sff

Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, J. P., & Saraiva, J. (2021). Ranking programming languages by energy efficiency. *Science of Computer Programming*, *205*, 102609. https://doi.org/10.1016/j.scico.2021.102609

Python Software Foundation. (2024). *functools — Higher-order functions and operations on callable objects*. Python Documentation. https://docs.python.org/3/library/functools.html

Secretaría de Economía de México. (2025). *Desarrolladores y analistas de software y multimedia*. Data México. https://www.economia.gob.mx/datamexico/es/profile/occupation/desarrolladores-y-analistas-de-software-y-multimedia

TRBL Services. (2024). *Optimización de rendimiento en desarrollo de videojuegos: estrategias y herramientas*. https://trbl-services.eu/rendimiento-en-desarrollo-de-videojuegos/

Unity Technologies. (2026). *System requirements for Unity 6.3*. Unity Manual. https://docs.unity3d.com/Manual/system-requirements.html

Voisard, L., De Freitas Serra, H., Politowski, C., Petrillo, F., & Guéhéneuc, Y. G. (2025). A mapping study of the entity component system pattern. En *2025 IEEE/ACM 9th International Workshop on Games and Software Engineering (GAS)* (pp. 33–40). IEEE. https://doi.org/10.1109/GAS66647.2025.00010

Xataka México. (2025, 27 de agosto). *Teclado y mouse por 363 pesos: la oferta de Amazon para ahorrar en accesorios de computadora*. https://www.xataka.com.mx/seleccion/amazon-tiene-este-kit-teclado-mouse-inalambricos-400-pesos-compatible-windows-mac

## ANEXOS

Esta sección recopila visualizaciones complementarias generadas durante el análisis de rendimiento que, por razones de extensión, no fueron incorporadas en el cuerpo principal del documento. Estos gráficos amplían la evidencia presentada en el Análisis de Rendimiento. Cada figura tiene una descripción que facilita su comprensión de forma independiente al texto principal.

### *Anexo A*

*Cuadro 2. Carga computacional por juego y nivel*

| Juego / Nivel | Entidades Totales | Draw Calls | Objetos Python Activos | Complejidad Visual |
|---|---|---|---|---|
| Platformer (Nivel 1) | 40 – 80 | 45 – 85 | ~154,500 – 156,500 | Media |
| Racing (Nivel 1) | 75 | 80 | ~154,500 – 156,500 | Media-Baja |
| Racing (Nivel 2) | 140 | 145 | ~228,000 – 229,800 | Alta |
| Racing (Nivel 3) | 78 | 83 | ~301,000 | Media (acumulada) |
| Fight (Nivel 1) | 2 | 10 | ~21,265 – 21,290 | Muy Baja |

*Nota: Los valores de objetos Python activos incluyen estructuras internas del intérprete.*

### *Anexo B*

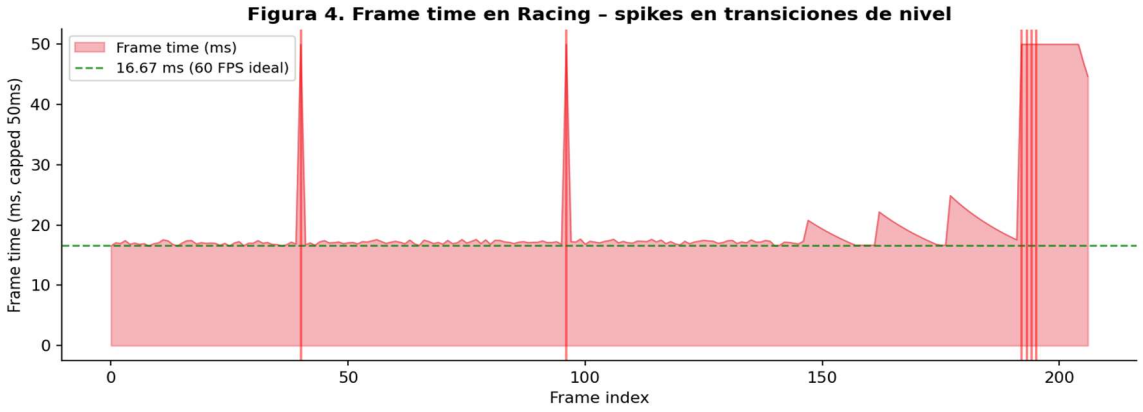


*Figura 4. Frame time registrado en Racing a lo largo de las sesiones. Las líneas verticales rojas señalan los spikes de transición de nivel. La línea discontinua verde indica los 60 FPS objetivos.*

La Figura 4 muestra con claridad el patrón de comportamiento del frame time en Racing. En condiciones normales de ejecución, el frame time está entre el rango de 16.67 ms y 17.5 ms, valores que corresponden al rango 57–60 FPS y que resultan imperceptibles para el usuario. Sin embargo, en los momentos de transición entre niveles se identifican spikes de frame time que alcanzan 2,237 ms en la transición Nivel 1→2 y 2,791 ms en la transición Nivel 2→3. Estos eventos, de duración única por transición, corresponden a la carga sincrónica de nuevos recursos gráficos y la creación de un nuevo objeto pista. A diferencia de los drops de FPS observados en el Platformer (que son puntuales y atribuibles a ciclos del GC), los spikes de transición de Racing son estructurales y predecibles, lo que abre la posibilidad de disminuirlos mediante carga asincrónica o pre-caching de niveles.

## *Anexo C*

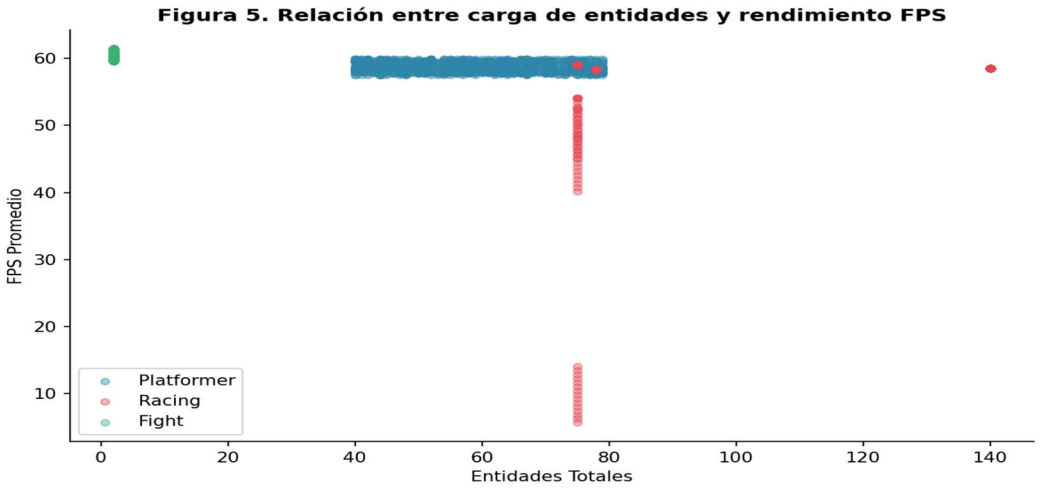


*Figura 5. Relación entre el número de entidades activas y el FPS promedio registrado.*

El Cuadro 2 y la Figura 5 muestran la relación entre carga de entidades y rendimiento de FPS. Un hallazgo importante es que el incremento de 75 a 140 entidades en Racing al pasar del Nivel 1 al Nivel 2 no produce una degradación del FPS durante el juego: la mediana del FPS resulta en el rango 58–60, lo que sugiere que la carga de dibujado asociada a 140 entidades con sus respectivos 145 draw calls es manejable para el sistema de dibujado en hardware de referencia.

## *Anexo D*

**Figura 6. Comparativa de métricas clave entre juegos**

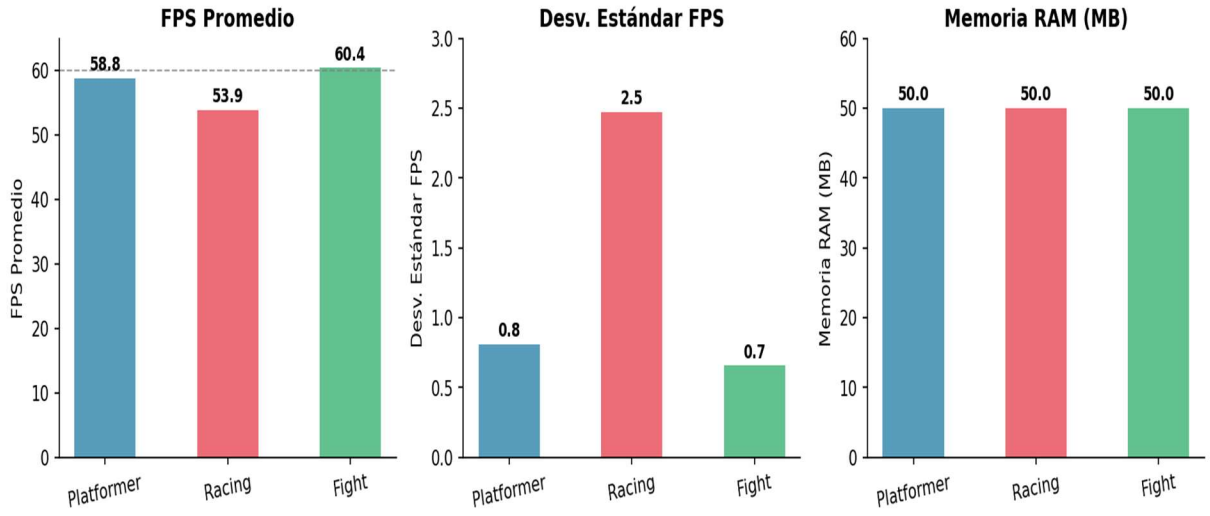


*Figura 6. Comparativa de datos clave entre los tres videojuegos: FPS promedio, desviación estándar del FPS y consumo de memoria RAM.*

La Figura 6 muestra los tres datos más relevantes en una vista comparativa. Respecto al FPS promedio, Fight lidera la cuenta con 60.41, seguido de cerca por Platformer con 58.75; Racing muestra el valor más bajo (53.9) por influencia de los frames atípicos ya discutidos —su mediana real de 58.10 lo coloca en el mismo rango que los demás juegos. En cuanto a la desviación estándar, Fight exhibe la mayor consistencia (0.75), mientras que la elevada desviación de Racing (11.90) es consecuencia directa de los spikes de transición y el consumo de memoria RAM es idéntico en los tres juegos (50 MB constantes). Estos tres indicadores confirman la viabilidad técnica de ModeX: el FPS cercano a 60 valida la fluidez del sistema, la baja desviación estándar confirma su estabilidad, y el consumo constante de 50 MB de RAM demuestra una gestión de memoria eficiente, cumpliendo así los estándares mínimos de un videojuego funcional y de calidad para el usuario en producción.

Las figuras presentadas en este apartado complementan y respaldan cuantitativamente las interpretaciones desarrolladas en la sección de Análisis de Resultados. En conjunto, las visualizaciones confirman que el sistema ModeX mantiene un rendimiento estable y predecible en condiciones normales de operación.