Departamento de Ciencias de la Computación
Universidad de las Fuerzas Armadas - ESPE

Tarea No. 1

Comparación de motores de juegos

Integrantes:
Yeshua Amador Chiliquinga Amaya
Andrés Alexander Espín Andrade
José Miguel Sanmartín Galán
Josué Abrahan Ferrin Lozano

Carrera / Asignatura: Ingeniería de Software / Desarrollo de Videojuegos
NRC: 28478
Docente: Hector Paúl Pinto Pachacama, Ing

Fecha de presentación: 22 de octubre de 2025

Índice

1. Introducción

Un motor de videojuegos (game engine) es un framework de software diseñado para la creación y el desarrollo de videojuegos. Proporciona a los desarrolladores un conjunto de herramientas y funcionalidades esenciales, abstrayendo la complejidad subyacente del hardware y permitiéndoles centrarse en los aspectos creativos del juego, como la jugabilidad, los gráficos y la narrativa.

Las funcionalidades clave de un motor moderno incluyen:

- **Motor de renderizado:** Para gráficos 2D y 3D.
- **Motor de físicas:** Para simular colisiones, gravedad y otras interacciones.
- **Gestión de audio:** Procesamiento y reproducción de sonido y música.
- **Scripting (Guionizado):** Lógica del juego mediante lenguajes de programación o sistemas visuales.
- **Gestión de entradas:** Manejo de teclado, ratón, mandos y pantallas táctiles.
- **Animación:** Herramientas para crear y controlar animaciones de personajes y objetos.

La elección del motor es una de las decisiones más críticas en el desarrollo de un videojuego, ya que impacta directamente en el alcance del proyecto, el flujo de trabajo del equipo y el producto final. En este documento, se comparan cuatro motores prominentes: Unity, Unreal Engine, Godot Engine y GameMaker Studio.

2. Criterios de Comparación

Para realizar un análisis equitativo y estructurado, el grupo ha definido los siguientes criterios de evaluación:

1. **Curva de Aprendizaje y Facilidad de Uso:** La dificultad y el tiempo requerido para que un nuevo desarrollador se vuelva competente en el motor.
2. **Lenguaje de Programación:** El/los lenguaje(s) de scripting utilizados (p.ej., C++, C#, GDScript) y su impacto en la productividad y el rendimiento.
3. **Capacidades Gráficas (2D/3D):** La potencia y flexibilidad del motor para renderizar gráficos, tanto en dos como en tres dimensiones.
4. **Rendimiento:** La eficiencia general del motor y qué tan bien se ejecutan los juegos compilados, especialmente en hardware de gama baja o media.
5. **Plataformas Soportadas:** La capacidad del motor para exportar el juego a diferentes sistemas (PC, consolas, móviles, web).
6. **Modelo de Licencia y Costo:** La estructura de precios, incluyendo costos iniciales, regalías (royalties) o tarifas por instalación.

7. **Comunidad y Ecosistema:** El tamaño de la comunidad de desarrolladores, la calidad de la documentación y la disponibilidad de recursos de aprendizaje (tutoriales, foros).

8. **Tienda de Activos (Asset Store):** La disponibilidad y calidad de activos prehechos (modelos 3D, sprites 2D, scripts) que pueden acelerar el desarrollo.

3. Análisis de Motores de Juego

3.1. Unity

Características: Motor versátil basado en componentes. Utiliza **C#** como lenguaje principal de scripting. Ofrece múltiples pipelines de renderizado (URP, HDRP) y un robusto ecosistema.

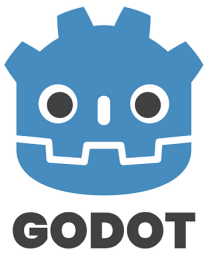
- **Ventajas:**
  - **Ecosistema Maduro:** La **Unity Asset Store** es la más grande del mercado, ahorrando incontables horas de desarrollo.
  - **Multiplataforma Real:** Es el líder indiscutible para el desarrollo móvil (iOS/Android) y tiene un excelente soporte para consolas, PC y VR/AR.
  - **Curva de Aprendizaje:** Más fácil de iniciar que Unreal. C# es un lenguaje moderno y ampliamente utilizado.
- **Desventajas:**
  - **Modelo de Licencia (Controversia):** La reciente introducción de la Runtime Fee"(tarifa por instalación) ha generado desconfianza en la comunidad, a pesar de las revisiones posteriores.
  - **Rendimiento 3D:** Aunque potente, alcanzar el fotorrealismo de alta gama de Unreal (p.ej., Nanite, Lumen) es considerablemente más difícil.
- **Limitaciones:** El pipeline de renderizado por defecto (Built-in) está obsoleto. Los proyectos 3D complejos requieren una configuración cuidadosa de URP o HDRP.
- **Riesgos:** Incertidumbre sobre futuros cambios en el modelo de precios y licenciamiento.
- **Juegos Adecuados:** Títulos móviles (p.ej., *Genshin Impact*), juegos independientes 2D y 3D (p.ej., *Hollow Knight*), VR/AR, y producciones de "doble A"(AA).

3.2. Unreal Engine (UE)

Características: El estándar de la industria para gráficos **AAA**. Utiliza **C++** para la lógica de bajo nivel y **Blueprints** (un potente sistema de scripting visual). Conocido por sus tecnologías de vanguardia como Nanite (geometría virtualizada) y Lumen (iluminación global).

- **Ventajas:**
  - **Gráficos de Vanguardia:** Insuperable en fotorrealismo. Es la elección para juegos que buscan çalidad de próxima generación".
  - **Blueprints:** Permite a artistas y diseñadores implementar lógica de juego compleja sin escribir código.
  - **Acceso al Código Fuente:** El motor es de código abierto (aunque no libre), permitiendo modificaciones profundas.
- **Desventajas:**
  - **Curva de Aprendizaje:** Extremadamente pronunciada. C++ es complejo y el motor en sí es masivo y pesado.
  - **Requisitos de Hardware:** Requiere un PC de desarrollo potente.
  - **Desarrollo 2D:** Sus herramientas 2D (Paper2D) son limitadas y no son su foco principal.
- **Limitaciones:** Excesivo (overkill) para juegos simples o móviles. El tamaño de los proyectos y los tiempos de compilación pueden ser muy grandes.
- **Riesgos:** La complejidad del proyecto puede salirse de control fácilmente. Requiere un equipo con alta especialización.
- **Juegos Adecuados:** Títulos AAA 3D (p.ej., *Star Wars Jedi: Survivor*), shooters en primera/tercera persona, juegos de mundo abierto y simulaciones arquitectónicas.

3.3. Godot Engine



Características: Un motor **completamente libre y de código abierto** (FOSS) bajo la licencia MIT. Su arquitectura se basa en una jerarquía de nodos y escenas. Utiliza **GDScript** (un lenguaje similar a Python) y también soporta C#.

- **Ventajas:**
  - **Licencia (MIT):** 100 % gratuito. Sin regalías, sin tarifas, sin condiciones. El juego es propiedad total del desarrollador.
  - **Flujo de Trabajo 2D:** Considerado uno de los mejores motores para desarrollo 2D, con un pipeline dedicado y herramientas específicas (p.ej., TileMaps).
  - **Ligero y Rápido:** El editor es extremadamente pequeño (menos de 100MB) y se abre instantáneamente. GDScript es muy fácil de aprender.
- **Desventajas:**
  - **3D de Alta Gama:** Aunque Godot 4 ha mejorado enormemente (con el backend de Vulkan), aún no compite con UE5 en fotorrealismo.
  - **Ecosistema Joven:** Su tienda de activos es pequeña y hay menos desarrolladores experimentados en el mercado laboral.
- **Limitaciones:** El soporte para consolas (PlayStation, Xbox, Switch) no es directo y generalmente requiere de servicios de terceros (como W4 Games) debido a los NDAs de las consolas.
- **Riesgos:** Riesgo bajo. El principal riesgo es encontrar limitaciones de rendimiento en proyectos 3D extremadamente ambiciosos.
- **Juegos Adecuados:** Juegos independientes 2D (p.ej., *Cassette Beasts*), pixel art, juegos para game jams, prototipos rápidos y herramientas de software.

3.4. GameMaker Studio

Características: Un motor veterano **enfocado casi exclusivamente en 2D**. Utiliza su propio lenguaje, **GML (GameMaker Language)**, y un sistema de scripting visual (GML Visual).

- **Ventajas:**
  - **Velocidad de Prototipado 2D:** Posiblemente el motor más rápido para crear un juego 2D funcional, especialmente plataformeros o shooters top-down.
  - **Curva de Aprendizaje Baja:** GML es simple y está diseñado específicamente para la lógica de juegos.
  - **Historial Comprobado:** Muchos de los juegos independientes más exitosos se hicieron en GameMaker.
- **Desventajas:**
  - **Capacidades 3D:** Prácticamente inexistentes. No está diseñado para eso.
  - **Lenguaje Propietario:** GML no es transferible a ningún otro entorno de desarrollo.
  - **Modelo de Licencia:** Se basa en suscripción para exportar a diferentes plataformas (consolas).
- **Limitaciones:** Estás .[en]cerrado.[en] un paradigma de desarrollo 2D. No es flexible para proyectos que no sean juegos.
- **Riesgos:** Alcanzar el "techo"del motor. Si el proyecto crece y requiere 3D complejo, se necesitaría una migración completa a otro motor.
- **Juegos Adecuados:** Juegos 2D de estilo retro y pixel art (p.ej., *Undertale*, *Hotline Miami*), plataformeros y juegos de arcade.

4. Matriz Comparativa

La siguiente tabla resume los puntos clave de cada motor basados en los criterios definidos.

Cuadro 1: Matriz Comparativa de Motores de Videojuegos

Criterio	**Unity**	**Unreal Engine**	**Godot Engine**	**GameMaker Studio**
Curva de Aprendizaje	Media. C# es amigable. El editor es complejo pero bien documentado.	Alta. C++ es difícil y el motor es masivo. Blueprints ayudan.	Baja. GDScript (similar a Python) es muy fácil. Arquitectura de nodos intuitiva.	Muy Baja. GML es simple y enfocado. Ideal para principiantes.
Lenguaje Principal	C#	C++ y Blueprints (visual)	GDScript y C#	GML (GameMaker Language) y GML Visual
Capacidades Gráficas	Fuerte en 3D (URP/HDRP) y 2D. Excelente para VR/AR.	Vanguardia en 3D (Nanite, Lumen). Fotorrealismo. Débil en 2D.	Excelente en 2D. 3D moderno y capaz (Vulkan), pero detrás de UE/Unity.	Excelente en 2D y pixel art. Capacidades 3D casi nulas.
Rendimiento	Bueno. Optimizado para móviles. Puede sufrir en 3D de alta gama si no se configura bien.	Excelente en 3D de alta gama. Excesivo y pesado para proyectos simples o móviles.	Excelente. Muy ligero. Optimizado para 2D y 3D de gama media.	Excelente para 2D. Muy ligero.
Plataformas	Todas (PC, Consolas, Móvil, Web, VR/AR). El rey de la multiplataforma.	Todas (PC, Consolas, Móvil). Fuerte en consolas de alta gama.	PC, Móvil, Web. Consolas requieren servicios de terceros.	PC, Consolas, Móvil, Web. Requiere suscripción para consolas.
Logotipo y Costo	Gratis (Personal). Planes Pro (suscripción). **Runtime Fee** controversial por instalación.	Gratis para empezar. **5 % de regalías** sobre ingresos brutos después del primer $1M USD.	**100 % Gratis (Licencia MIT).** Sin regalías, sin tarifas, sin condiciones.	Suscripción mensual/anual para exportar (especialmente a consolas).
Ecosistema	Masivo. La **Asset Store** más grande. Comunidad enorme.	Grande. Fuerte en AAA. "Quixel Megascans"(activos fotorrealistas) está integrado.	Creciente. Comunidad apasionada y colaborativa. Asset Library más pequeña.	Mediano. Comunidad muy fuerte y enfocada en 2D.

5. Caso de Estudio y Recomendación

Para la exposición, se plantea un proyecto específico y se selecciona el motor ideal.

5.1. Definición del Proyecto

- **Título (Provisional):** .El Eco del Relojero".
- **Género:** Juego de plataformas 2D con elementos de puzzle.
- **Estilo Gráfico:** Pixel art de alta definición (estilo *Celeste* o *Cassette Beasts*).
- **Equipo:** Un grupo pequeño (1-3 desarrolladores) con habilidades mixtas (programación y arte).
- **Plataformas Objetivo:** PC (Steam) y Móviles (iOS/Android).
- **Restricciones:** Presupuesto limitado, necesidad de prototipado rápido.

5.2. Análisis y Recomendación

Basado en los requisitos del proyecto, se evalúa la idoneidad de cada motor:

- **Unreal Engine: Descartado.** Es un motor 3D. Su pipeline 2D (Paper2D) es insuficiente y el motor es demasiado pesado (overkill) para un juego de pixel art destinado a móviles.
- **GameMaker Studio: Fuerte Contendiente.** Es excelente para pixel art 2D y prototipado rápido. Sin embargo, su modelo de suscripción para exportar a consolas (un objetivo futuro potencial) y el uso de un lenguaje propietario (GML) son desventajas.
- **Unity: Contendiente Sólido.** Tiene herramientas 2D muy robustas (Tilemaps, 2D Physics) y el mejor soporte para exportar a móviles del mercado. La Asset Store es una gran ventaja. *Riesgo Principal:* El modelo de Runtime Fee", aunque no afectaría a un equipo pequeño al inicio, genera incertidumbre a largo plazo para un juego móvil que podría volverse viral.
- **Godot Engine: Recomendación Ideal.**
  1. **Costo (Licencia MIT):** Es el factor decisivo para un equipo con presupuesto limitado. **Cero regalías y cero tarifas** significa que el 100 % de los ingresos de Steam y las tiendas móviles van al desarrollador (después del recorte de la tienda).
  2. **Flujo de Trabajo 2D:** Su pipeline 2D está diseñado desde cero y es considerado superior al de Unity, especialmente para pixel art y gestión de Tilemaps.
  3. **Curva de Aprendizaje (GDScript):** Perfecto para prototipado rápido. Un equipo pequeño puede ser productivo en días.
  4. **Ligero:** El juego final será liviano, ideal para el mercado móvil.

Conclusión del Caso de Estudio: Para el proyecto .[El] Eco del Relojero", **Godot Engine** es el motor ideal. Ofrece las mejores herramientas 2D de su clase, un lenguaje de scripting extremadamente rápido y una licencia 100 % libre que elimina por completo el riesgo financiero, lo cual es crítico para un equipo independiente pequeño.

6. Conclusiones

No existe un "mejor motor de videojuegosüniversal. La elección depende fundamentalmente de las metas del proyecto, la experiencia del equipo y las plataformas objetivo.

- **Unreal Engine** es la potencia indiscutible para juegos 3D fotorrealistas de gran presupuesto (AAA).
- **Unity** es el "todoterreno", destacando en el desarrollo móvil y 3D intermedio, con el ecosistema de activos más grande.
- **GameMaker Studio** sigue siendo el rey de la velocidad y la simplicidad para juegos 2D puros y de estilo retro.
- **Godot Engine** se ha posicionado como la opción FOSS (Libre y de Código Abierto) más fuerte, siendo una opción ideal para el desarrollo 2D independiente gracias a su excelente flujo de trabajo y su licencia sin riesgos.

7. Bibliografía

Referencias

[1] Unity Technologies. (2025). *Documentación Oficial de Unity*. Obtenido de https://docs.unity.com

[2] Epic Games, Inc. (2025). *Documentación de Unreal Engine*. Obtenido de https://docs.unrealengine.com

[3] Godot Foundation. (2025). *Documentación de Godot Engine*. Obtenido de https://docs.godotengine.org

[4] YoYo Games Ltd. (2025). *Manual de GameMaker*. Obtenido de https://manual.yoyogames.com