Centro de Estudios de Postgrado

Máster en Profesorado de Enseñanza Secundaria Obligatoria, Bachillerato, Formación Profesional y Enseñanza de Idiomas

Universidad de Jaén

Centro de Estudios de Postgrado

Trabajo Fin de Máster

PROGRAMACIÓN DE JUEGOS. LIBRERÍAS Y ENTORNOS DE DESARROLLO ESPECÍFICOS. MOTORES DE JUEGOS. FUNCIONES Y COMPONENTES.

Alumno/a: **Barbero Rodríguez, María de la Paz**

Tutor/a: Prof. D. Pedro García González

Dpto: Departamento de Informática

Contenido

Índice de figuras

Índice de tablas

1. Resumen y palabras clave/ Abstract and keywords.

En este documento se presenta el Trabajo de Fin de Master titulado *Programación de juegos. Librerías y entornos de desarrollo específicos. Motores de juegos. Funciones y componentes* que hace referencia al tema 39.2 de la Orden EDU/3138/2011, de 15 de noviembre de 2011.

El Trabajo consta de dos partes, un estudio epistemológico y la proyección didáctica. En la primera parte se hará un repaso por la historia de los videojuegos destacando los hitos más relevantes, se nombrarán las librerías y entornos más importantes en el desarrollo de juegos, se detallará la arquitectura que compone el motor, se analizarán dos de los motores más utilizados en la actualidad para la creación de videojuegos y se incluirán las tendencias futuras.

En la segunda parte, la proyección didáctica, se va a concretar el contenido de la parte anterior en la unidad didáctica *Introducción al desarrollo de videojuegos*, perteneciente al módulo *Programación multimedia y dispositivos móviles* correspondiente al Ciclo Formativo de Grado Superior de Desarrollo de Aplicaciones Multiplataforma. En ella se incluirán objetivos, metodología, detalles del proceso de evaluación y atención a la diversidad siguiendo la normativa vigente, tanto a nivel estatal como autonómico.

Palabras clave: videojuegos, motor de videojuegos, librerías, entornos de desarrollo

---

This document presents the Master's Thesis entitled Game Programming. Specific libraries and development environments. Game engines. Functions and components that refers to the topic 39.2 of the Order EDU/3138/2011, of November 15, 2011.

The work consists of two parts, an epistemological study and the didactic projection. The first part will review the history of video games highlighting the most important milestones, the most important libraries and environments in game development will be named, the architecture that makes up the engine will be detailed, two of the most widely used engines currently used for creating video games will be analysed and future trends will be included.

In the second part, the didactic projection, the content of the previous part will be specified in the didactic unit *Introducción al desarrollo de videojuegos*, belonging to the module *Programación multimedia y dispositivos móviles* corresponding to the *Ciclo Formativo de Grado Superior de Desarrollo de Aplicaciones Multiplataforma*. It will include objectives, methodology, details of the evaluation process and attention to diversity following the current regulations, both at state and regional level.

Keywords: video game, game engine, libraries, integrated development environment.

2. Fundamentación epistemológica.

2.1. Introducción

La industria de los videojuegos está en continuo crecimiento. En España el valor de este mercado alcanza los 2600 millones de euros (Desarrollo Español de Videojuegos (DEV), 2021). Esta industria no sólo hace referencia a la idea tradicional de que una persona adquiera un videojuego para su uso particular, sino que ha ido cobrando otras perspectivas nuevas. Hace unos años irrumpieron los *esports* (competiciones profesionales de videojuegos) en este panorama, de forma que los videojuegos se trasladaban a competiciones multitudinarias que se están posicionando cada vez más cerca de otras competiciones "tradicionales" con mayor recorrido, como la NBA o la NFL (*National Football League*) (Syracuse University, 2019). Además de los *esports*, otro aspecto destacable en esta industria es la relevancia de los creadores de contenido y *streamers* del mundo de los videojuegos. Concretamente, en una de las plataformas de *streaming* más conocidas, Twitch, los creadores españoles se sitúan a la cabeza del ranking mundial en cuanto a visualizaciones y suscripciones. Estas plataformas están compitiendo con los medios tradicionales como la televisión en cuanto a la franja de edad de los más jóvenes.

Por otra parte, los videojuegos no sólo tienen una finalidad de entretenimiento, sino que también son herramientas que pueden resultar muy útiles en ámbitos como la sanidad, la educación, la psicología… Estos son los denominados *serious games*, que permiten, por ejemplo, realizar una ejecución del Test del zoo, una prueba para medir las funciones ejecutivas de una persona, de forma que se aprovechan los beneficios de la gamificación también en el ámbito de la psicología.

Además de todo ello, los videojuegos están ocupando un lugar importante en cuanto a paliar los efectos psicológicos negativos que ha causado la pandemia, y concretamente, en el periodo de cuarentena. Por una parte, los videojuegos han ayudado a las personas a evadirse de la realidad gracias a su capacidad de inmersión en el mundo virtual, por otro lado, también han ayudado a las personas a relacionarse en ese mundo virtual, haciendo que el sentimiento de soledad que se produce sea menor. Este doble efecto se ha visto ilustrado por la salida al mercado del juego *Animal Crossing: New Horizons* (Nintendo, 2020) en el comienzo de la pandemia (Zhu, 2021).

Por todos estos motivos se demuestra la relevancia que han tenido, tienen y tendrán los videojuegos en la vida de la gente. A continuación, se va a repasar el inicio y evolución del mundo de los videojuegos tanto a nivel internacional como nacional para después pasar a los detalles más técnicos de la realización de los mismos y concluir con una visión de futuro de este sector.

2.2. Antecedentes.

La evolución del sector de los videojuegos se ha visto afectada por los avances tecnológicos de los que depende, por lo que podemos decir que también se rige por la Ley de Moore (Moore, 1965).

Los antecedentes de los primeros videojuegos, o juegos digitales, eran máquinas recreativas mecánicas o electromecánicas que funcionaban con monedas. En la década de 1890 nace el primer pinball de forma primitiva. Unos años más tarde, en el contexto de la Segunda Guerra Mundial la temática de estas máquinas pasa a basarse en misiles, carreras y lucha (Williams, 2017).

Primeros juegos digitales en entornos de investigación

En cuanto a los primeros juegos digitales, comenzaron a desarrollarse en el ámbito científico, como experimentos de investigación en universidades, pocos años más tarde, se inició su comercialización en el ámbito recreativo.

En los años posteriores a la Segunda Guerra Mundial surgen las primeras minicomputadoras, que suponen avances en cuanto a tamaño, velocidad y coste. Esto permitió un mayor acceso a ellas y el desarrollo de juegos dirigidos a la investigación en universidades e institutos que sirvieron como instrumento para demostrar teorías científicas. Estos primeros juegos construyeron los cimientos de los posteriores juegos comercializados y el desarrollo de videojuegos en general.

Uno de los ámbitos en los que se desarrollaron los juegos de ordenador fue en la investigación de la inteligencia artificial mediante el ajedrez. En él destacan Leonardo Torres y Quevedo con *el Ajedrecista* en 1912 y Alan Turing y Claude Shannon, que utilizaron programas basados en el ajedrez como medio para demostrar teorías de toma de decisiones en los años 40. A finales de los 50, se consiguió que los ordenadores pudieran ejecutar partidas de ajedrez convencionales completas.

En 1952, Alexander S. Douglas crea el programa *OXO* o *Noughts and Crosses* (Figura 1. *Primeros videojuegos de la historia* [Figura]. Adaptación de Weekly Game Project, https://www.mobygames.com/game/mainframe/oxo_/promo/promoImageId,212557/, *A recreation of the original Tennis For Two*, Brookhaven National Library, https://www.bnl.gov/about/history/firstvideo.php y *SpaceWar!*, Nik Clayton, 2007, https://www.flickr.com/photos/nikclayton/1394377691., que algunas fuentes (Belli & López Raventós, 2008) consideran que es el primer videojuego de la historia. Era un tres en raya que se ejecutaba sobre el ordenador EDSAC y cuyo principal objetivo era su uso en la tesis de interacción persona-ordenador.

En 1958 nace de manos de William Higinbotham *Tennis for Two* (Figura 1. *Primeros videojuegos de la historia* [Figura]. Adaptación de Weekly Game Project, https://www.mobygames.com/game/mainframe/oxo_/promo/promoImageId,212557/, *A recreation of the original Tennis For Two*, Brookhaven National Library, https://www.bnl.gov/about/history/firstvideo.php y *SpaceWar!*, Nik Clayton, 2007, https://www.flickr.com/photos/nikclayton/1394377691. , un juego electrónico de dos jugadores cuya finalidad era el entretenimiento basado en el tenis de los jóvenes que visitaban el Laboratorio Nacional de Brookhave (Williams, 2017).

En este contexto de investigación en las universidades aparece el *hacking* en los años 50 y 60. Concretamente, en el MIT (Massachusetts Institute of Technology), durante las horas que los ordenadores estaban libres, el estudiante de informática Steve Russell desarrolló un juego con gráficos vectoriales basado en el espacio llamado *Spacewar!* en el ordenador Programmed Data Processor-1 (PDP-1) en 1962 (Figura 1. *Primeros videojuegos de la historia* [Figura]. Adaptación de Weekly Game Project, https://www.mobygames.com/game/mainframe/oxo_/promo/promoImageId,212557/, *A recreation of the original Tennis For Two*, Brookhaven National Library, https://www.bnl.gov/about/history/firstvideo.php y *SpaceWar!*, Nik Clayton, 2007, https://www.flickr.com/photos/nikclayton/1394377691.. El código de este juego se extendió por las universidades, laboratorios y en el ámbito empresarial, haciendo que surgieran numerosas versiones y adaptaciones para otros ordenadores.

Figura 1. *Primeros videojuegos de la historia* [Figura]. Adaptación de Weekly Game Project, https://www.mobygames.com/game/mainframe/oxo_/promo/promoImageId,212557/, *A recreation of the original Tennis For Two*, Brookhaven National Library, https://www.bnl.gov/about/history/firstvideo.php y *SpaceWar!*, Nik Clayton, 2007, https://www.flickr.com/photos/nikclayton/1394377691.

Primeros juegos digitales comercializados

En cuanto a los primeros juegos digitales comercializados seguían dos tendencias diferentes: adaptar juegos mecánicos tradicionales al formato digital o superar la barrera de lo convencional para así lograr nuevas experiencias de juego.

La distribución del código de *Spacewar!* promovió el nacimiento de *Computer Space* en 1971 a manos de Nolan Bushnell y Ted Dabney, uno de los juegos recreativos más populares y generalizados de la época, que utilizaba un circuito diseñado exclusivamente para su ejecución.

En 1966 Ralph Baer junto a un grupo de ingenieros comenzaron a trabajar en un sistema compuesto por una televisión que mostraba cuadros de luz interactivos. Este proyecto evolucionó hasta la creación de *Magnavox Odyssey* en 1972, la que podemos denominar como la primera consola doméstica. *Odyssey* estaba dotada de una serie de juegos pregrabados que se podían jugar mediante el uso de unas tarjetas especiales que modificaban la propia lógica de la máquina.

Mientras tanto, en ese mismo año, nace *Pong*, la primera máquina recreativa de la compañía Atari basada en el tenis de mesa. El juego tuvo un éxito inesperado, haciendo que otras compañías como Taito, Sega, Williams Electronics o Midway crearan sus primeros juegos digitales en torno a 1973, muchos de ellos basados en *Pong*. Uno de estos juegos fue *Breakout* (1976), creado por Steve Jobs y Steve Wozniak justo antes de fundar Apple.

Los primeros juegos arcade estaban basados en la lógica de transistores, mediante la cual, el comportamiento del juego estaba inscrito de forma física en los circuitos. Así, cada juego tenía un circuito totalmente diferente a cualquier otro, puesto que tenían un diseño específico. Sin embargo, a finales de los 70, con la adopción de los microprocesadores en el ámbito de la creación de videojuegos, cambió el modelo de desarrollo de los mismos, ya que, la aparición de placas más uniformes hizo aumentar la velocidad de producción, y con ella, la complejidad de los juegos. Como consecuencia, los programadores comenzaron a tomar mayor relevancia en el desarrollo de videojuegos, desplazando así a los ingenieros eléctricos que se habían encargado hasta el momento de esta tarea.

Edad de Oro de los recreativos en Norteamérica

Del 1978 al 1984 la industria del videojuego se divide en dos ámbitos: las máquinas recreativas arcade y las primeras consolas domésticas. Este período se conoce como la Edad de Oro de los recreativos en Norteamérica, debido a la revolución en cuanto a conceptos, personajes y elementos visuales de la época. Estos juegos se caracterizaron por tener niveles que aumentaban progresivamente la dificultad, a diferencia de los juegos que había hasta el momento que mantenían la misma dificultad, pero con una restricción de tiempo máximo. Los juegos comenzaron a incluir colores en los gráficos

debido a las mejoras en hardware y monitores. Fue una época de gran competitividad entre las empresas, aunque sobre todas ellas destacaba Atari. Algunos de los títulos más relevantes de este periodo son *Space Invaders* (Taito, 1978), *Asteroids* (Atari, 1979), *Pac-Man* (Namco, 1980), *Donkey Kong* (Nintendo, 1981) o *Tron* (Bally/Midway, 1982).

Primeras consolas domésticas

Por otra parte, en el mismo espacio de tiempo, las compañías de Fairchild Semiconductor y Atari comenzaron a desarrollar modelos de consolas domésticas más parecidas a la lógica de los ordenadores, donde podrían ejecutarse multitud de juegos, a diferencia del enfoque predominante en la época (unidades dedicadas a un juego exclusivamente). Posteriormente, otras empresas comenzaron a seguir esta corriente, lo que daría lugar a una competencia feroz en el mercado de consolas. Fairchild Semiconductor lanza en 1976 la primera consola doméstica basada en cartuchos, y Atari crea, un año después, la consola basada en cartuchos VCS (Video Computer System). Estos primeros juegos estaban muy influenciados por las ideas de diseño que se utilizaban para los juegos recreativos, aunque también surgieron nuevos enfoques más adecuados para jugar en casa, lo que hizo que el tiempo de juego se prolongara durante horas.

En cuanto al desarrollo de juegos de la VCS, el reto principal era mostrar la parte gráfica del juego de forma adecuada, ya que los recursos que tenía el dispositivo eran muy limitados. Esta consola se diseñó para ejecutar juegos de pelota como *Pong!* y de disparo como *Tank* (Namco, 1974), por lo que el hardware estaba ideado para ello. Los cartuchos de juegos tenían de 2 a 4 kilobytes, aunque la consola podía almacenar sólo 128 bytes. Sin embargo, gracias a técnicas gráficas como la ideada por Bob Whitehead, persianas venecianas, se podía aprovechar los recursos que la consola ofrecía.

Otro fenómeno destacable que ocurre en esta época es la aparición de compañías que desarrollaban juegos para consolas que no habían creado ellas mismas. Hasta la fecha, cada empresa diseñaba juegos exclusivamente para su máquina, pero, con el nacimiento de Activision (1979) e Imagic (1981) se comenzó a crear un perfil de compañías que creaban juegos para terceros, no sólo para consolas, sino también para ordenadores. Este hecho tuvo consecuencias, como el aumento de competencia entre empresas, mejora de la calidad y originalidad de los juegos desarrollados o el comienzo de los roles en el desarrollo de un juego.

Algunos ejemplos de consolas domésticas son Intellivison, de Mattel, en 1976 o ColecoVision, de Coleco, en 1982, con licencias de juegos creados por empresas como Nintendo, Namco, Konami y Sega. Los títulos más destacados de la época son *David Crane's Grand Prix* (1982, Activision), que fue más allá de los anteriores juegos de

carreras ofrecidos en VCS, ya que incluía vehículos multicolores con neumáticos que simulaban la rotación a diferentes velocidades o *Utopia* (1981, Mattel), un juego de rol ambientado en una isla, además de adaptaciones de juegos recreativos.

En la siguiente imagen (Figura 2. Adaptada de *The great history of video games* [Infografía], por Baron Hong, https://venngage.net/p/168706/the-great-history-of-video-games) se pueden observar estas primeras consolas domésticas.

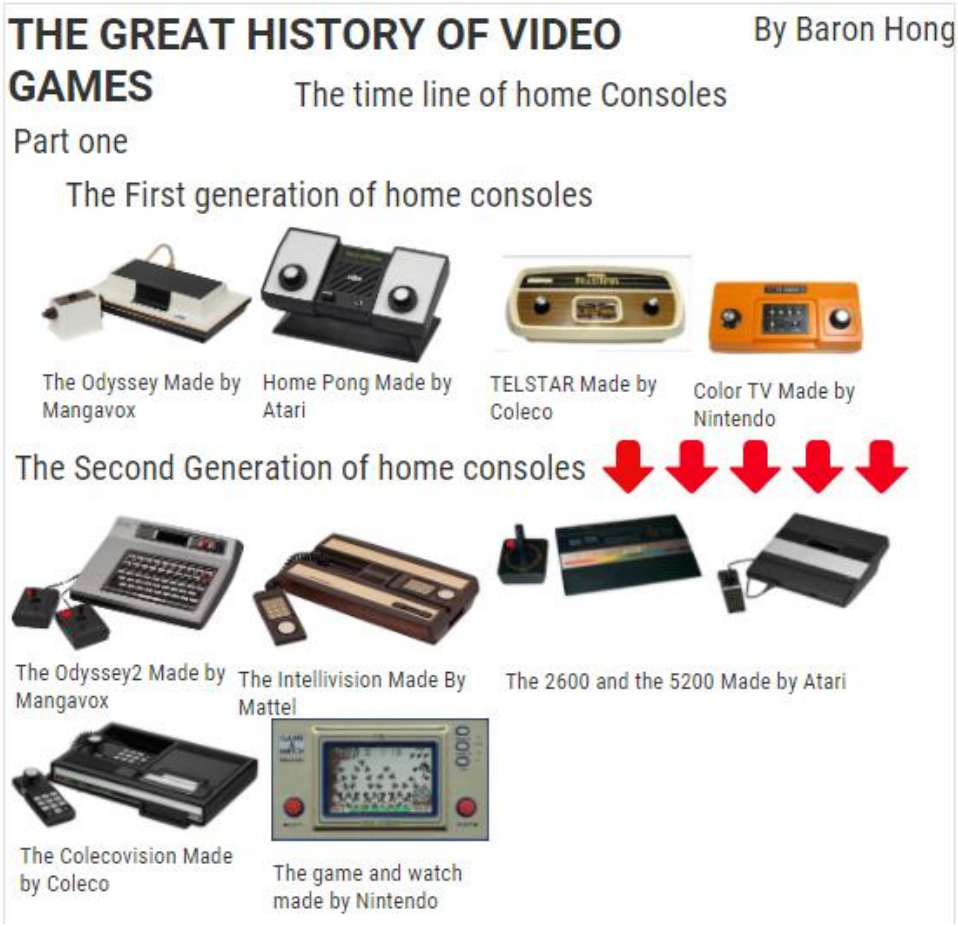

Figura 2. Adaptada de *The great history of video games* [Infografía], por Baron Hong, https://venngage.net/p/168706/the-great-history-of-video-games

Juegos en primeras computadoras domésticas

En cuanto al panorama de las computadoras, a finales de los 70 nacen los microordenadores gracias a compañías como Apple (Apple II), Commodore (Commodore PET), Tandy (TRS-80) y Atari (Atari 400 y Atari 800). Estos ordenadores gozaron de bastante popularidad en el desarrollo de juegos gracias a la gran capacidad que suponían en cuanto a gráficos y sonido por lo que comenzaron a introducirse en muchos hogares. Estos primeros ordenadores compartían características comunes con las consolas de segunda generación: procesadores de 8 bits, usaban monitores de televisión para mostrar la información…

Los géneros que se desarrollaron en los primeros juegos para ordenadores comerciales fueron: aventura, rol y simulación. Los primeros juegos de aventura no poseían gráficos, sino que usaban bloques de texto que describían la escena y el usuario tenía que introducir los comandos como órdenes de teclado. El primer juego de este tipo fue *Adventure* (Will Crowther, 1975), que ya comenzó a distribuirse por ARPAnet. Por otra parte, gracias a juegos como *Mystery House (1980,* On-Line Systems*)* o la saga *Hi-Res* Adventure, se comenzaron a introducir gráficos en los juegos y posteriormente, color en los mismos.

El género de rol, también conocido como CRPG (Computer Role-Playing Games), destacó en los primeros juegos de ordenador en los 70 y 80. Estos representaban sistemas de combate simulado contra enemigos como monstruos. Se trataba de juegos en primera persona con gráficos sencillos que representaban pasillos rectangulares. Destacan dos juegos principalmente, que tuvieron relevancia en los títulos posteriores: *Wizardry: Proving Grounds of the Mad Overlord* (SRI Tech, 1981) y *Ultima* (Richard Garriott, 1982).

En cuanto a los videojuegos de simulación de vuelo y conducción, uno de los primeros fue *Flight Simulator* (subLOGIC, 1979), en el que los jugadores podían realizar vuelos libres e incluso participar en combates de la Primera Guerra Mundial (Figura 3. *The bridge and mountains in release 3 of A2-FS1* [Figura], por Jos Grupping, 2005, https://fshistory.simflight.com/fsvault/fs1-apple.htm. La parte gráfica del juego consistía en formas creadas con líneas rectas, aunque, en sus posteriores versiones, tanto el realismo de los gráficos como el tamaño del mundo aumentaron. *Flight Simulator* estaba disponible para los ordenadores Apple II y TRS-80.

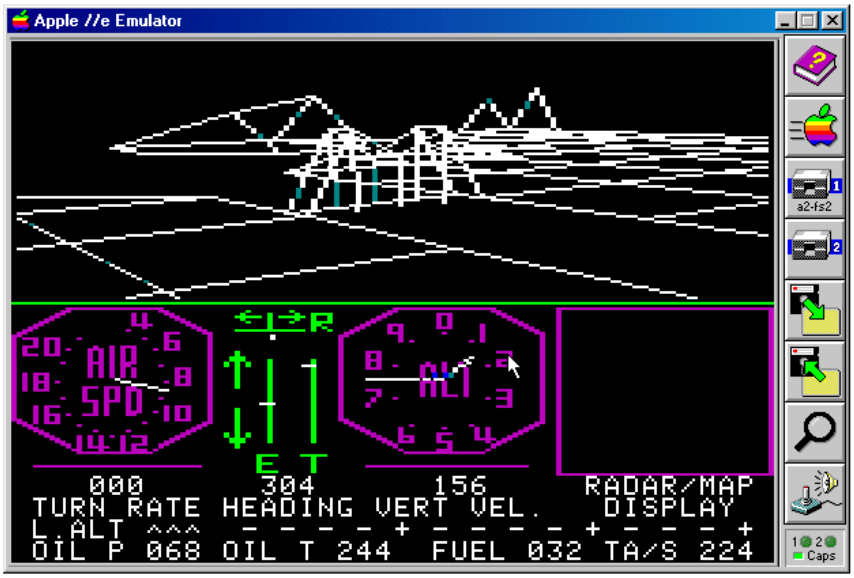

Figura 3. *The bridge and mountains in release 3 of A2-FS1* [Figura], por Jos Grupping, 2005, https://fshistory.simflight.com/fsvault/fs1-apple.htm

A medida que las prestaciones de los nuevos modelos mejoraron, la parte gráfica comenzó a tener mayor importancia. Estos nuevos modelos eran Commodore 64 (1982), ZX Spectrum (1982) y versiones más potentes del Apple II.

A finales de los años 80 surge una nueva generación de ordenadores de 16 y 32 bits con mayores capacidades gráficas y computacionales que las existentes hasta el momento. Por este motivo, se produjo un gran interés por parte de los desarrolladores para aprovechar todo el abanico de posibilidades que se abría con estas mejoras. Este fenómeno se potenció mucho más cuando Apple lanza Macintosh en 1984, que incorporaba ratón e interfaz gráfica, lo que hizo que las formas de interacción cambiaran y, con ello, el diseño de los videojuegos.

La industria del videojuego japonesa

Otro mercado fuerte en el sector de los videojuegos, además de Estados Unidos, fue el japonés. Este mercado se diversifica entre las máquinas recreativas y las consolas domésticas. En el ámbito de las recreativas, destacaron tres tendencias principales:

- La simulación del 3D, jugando con la profundidad mediante cambios de escala de los objetos (en este proceso tenía un papel esencial los procesadores de 16 bits que podían manejar el cambio de tamaño de los *sprites*). En este campo destaca Yu Suzuki, que trabajaba en Sega, y desarrolló una tecnología de hardware que permitía cambiar el tamaño de miles de sprites por segundo, lo que producía sensación de movimiento. Esta tecnología fue la base de otros muchos juegos posteriores de la compañía.
- Los juegos de desplazamiento lateral, en los que los personajes tenían que luchar contra una gran cantidad de enemigos a la vez que se iban desplazando. Algunos títulos de este género son *The Legend of Kage* (Taito, 1984) o *Rolling Thunder* (Namco, 1986).
- Los juegos de lucha cara a cara consistían en una serie de combates uno contra uno en el que cada personaje dispone de una barra de salud que hay que debilitar con la ayuda de ataques como patadas. Uno de los más conocidos fue *Street Fighter* (Capcom, 1987), que sentó las bases del diseño de los posteriores títulos de este género.

Por otra parte, en 1981 Nintendo Masayuki Uemura buscaba crear una consola que pudiera reproducir los juegos arcade, lo que desemboca en 1983 en Family Computer o Famicon, una consola de cartuchos y procesador de 8 bits que usaba bloques de 8x8 pixeles y gráficos basados en *tiles* (mosaico). También incluía unos controles más cómodos e intuitivos (flechas en forma de signo '+' para movimientos horizontales y verticales), que resultaron ideales para el diseño de juegos los que se tenga la

posibilidad de explorar el mundo. La Famicon resultó ser bastante exitosa en Japón, y también su adaptación al mercado norteamericano, la consola NES.

Nintendo comenzó a incorporar el enfoque de diseño de juegos basados en niveles más orientados a la jugabilidad en casa, en vez de adaptar los títulos recreativos a la consola doméstica. Siguiendo esta línea, nace *Super Mario Bros* (1985), uno de los juegos más conocidos e influyentes de la historia. Otro título que se desarrolló al mismo tiempo fue *The Legend of Zelda* (1986) con la misma narrativa que el primero, un héroe que maneja el jugador y se desplaza por un mundo para conseguir liberar a la princesa de un enemigo. El éxito fue tan abrumador que de estos clásicos surgieron numerosas secuelas.

Con la llegada de los procesadores de 16 bits al mundo de las consolas, la competencia entre Nintendo, Sega y NEC se acrecentó. En el caso de Nintendo, se vio favorecida por la exclusividad de sus títulos, ya que crearon juegos de nueva generación basados en otros clásicos como *Castlevania* o *Adventure Island*. Una consola que destacó sobre las demás fue la Mega Drive de Sega, cuyo objetivo era reproducir de forma fiel los títulos arcade más populares. Además de esta visión, Sega se aleja un poco de este objetivo con el lanzamiento de *Sonic the Hedgehog* (Sonic Team, 1991), que competía directamente con los títulos de Super Mario de Nintendo. A su vez, Nintendo continuaba lanzando títulos conocidos como Donkey *Kong Country* (Rare, 1994) y *Super Mario World 2: Yoshi's Island* (Nintendo, 1995) a la vez que iba madurando dirigiéndose lentamente al panorama incipiente de los juegos 3D.

Mejora de las prestaciones y comienzo del 3D

A finales de los 80 y mediados de los 90 se comienza a usar el CD-ROM en el ámbito de los ordenadores de forma masiva en las aplicaciones multimedia y los videojuegos, que pretendían obtener un mayor realismo en la parte gráfica del juego. Surgieron dos corrientes diferentes en cuanto a la consecución de realismo gráfico:

- Fotorrealismo: combinaba imágenes digitalizadas, video en movimiento y gráficos 3D prerrenderizados.
- Realismo espacial: cálculo en tiempo real de los polígonos en 3D. Sin embargo, esta técnica requería excesivo cómputo.

A su vez, también surgieron consolas basadas en CD-ROM. Incluso, los fabricantes principales (Nintendo, Sega y NEC), crearon módulos para añadir a sus consolas de cartuchos para que estas también pudieran disfrutar de esta tecnología. Al mismo tiempo, los ordenadores incorporaban cada vez más el módulo de CD-ROM como una característica básica. Debido a la gran variedad de consolas que existían, los desarrolladores (de terceros) comenzaron a desplegar los juegos en múltiples plataformas.

Mientras tanto, los simuladores en máquinas recreativas comienzan a incluir 3D mediante uso de polígonos en 3D con altas tasas de refresco para conseguir sensación de velocidad. Namco crea *Winning Run* en 1988, un juego de F1 con un hardware capaz de procesar 60000 polígonos en un segundo. Después, otras compañías siguieron este camino de desarrollo de hardware con mayor capacidad de procesamiento de polígonos, lo que hizo también que esta tecnología llegara a otros géneros, como el de *shooters* (disparos).

Las consolas también comenzaron a incorporar el 3D en sus desarrollos, haciendo que, compañías como Nintendo realizara actualizaciones de hardware en su consola. Uno de los primeros juegos para consolas en incorporar 3D fue *Star Fox* (Nintendo, 1993), que usaba polígonos 3D para la nave del jugador y los enemigos. En el desarrollo de juegos de ordenador, también comenzaron a incorporar juegos en 3D, aunque, debido a las limitaciones tecnológicas, tenían que entremezclar elementos 2D (imágenes y texturas sobre geometría de alambre) y 3D (personajes), por ejemplo, *The Terminator* (Bethesda, 1991).

Id Software y primer motor de juegos

Cabe destacar la aportación de Id Software en el enfoque del desarrollo de videojuegos. Esta fue una compañía fundada en 1991 por John Carmack, John Romero, Adrian Carmack, Tom Hall y Jay Wilbur, programadores, artistas y diseñadores que ya tenían experiencia en el mundo de los videojuegos. Esta empresa ha tenido una gran influencia en la historia de los videojuegos, gracias a que dio forma al género que conocemos como FPS *(first person shooter*). En 1993, lanza *Doom*, un juego FPS en 3D que incluía la técnica *raycasting* para conseguir una mayor eficiencia y fluidez en el juego. Este juego también alentó la creación de *mods* (modificaciones creadas por los jugadores en los juegos que cambian algún aspecto del mismo) debido a la facilidad que ofrecía para ello, pues, las diferentes partes del código se encontraban separadas según su función. Por este motivo, se considera que el nacimiento del concepto "motor de videojuegos" se debe al videojuego *Doom*. Además de ello, fue de los primeros juegos en aprovechar Internet para crear un modo de juego multijugador.

Década de los 2000

A finales de los 90, la tónica general se centraba en los juegos 3D en tiempo real, que se vieron favorecidos gracias a las mejoras en hardware para ordenadores, como los procesadores Pentium y la aparición de tarjetas gráficas. Las principales marcas Sega, Sony y Nintendo lanzaron nuevas consolas de quinta generación, concretamente, destacando PlayStation de Sony gracias a sus características tanto gráficas como a la hora del almacenamiento, que superó a la Nintendo 64. Los juegos tradicionales 2D

comenzaron a actualizarse a 3D con títulos como *Super Mario 64* (1996) y Clash *Bandicoot* (Naughty Dog, 1996).

La industria se centró en el desarrollo de videojuegos para consolas a comienzos de los 2000 debido a que la arquitectura para las consolas era fija, a diferencia del caso de los ordenadores, cuya arquitectura de componentes era variable, lo que provocaba problemas de compatibilidad que requerían mayor esfuerzo en el desarrollo. Durante los últimos años, Sony y Nintendo se hacen con el mercado de las consolas con la serie de Wii y PlayStation2 y sus consolas portátiles como Nintendo DS y PSP. A estas empresas se suma Microsoft con la creación de XBox, que intentaba acercarse al diseño de un ordenador. Empleaba DirectX, Pentium III, un disco duro y una tarjeta gráfica, además de un módem para el acceso a Internet.

En cuanto a los gráficos, la mejora y evolución fue rápida debido, en parte, a las mejoras en hardware. Estas mejoras llevan a intentar representar los modelos con el mayor realismo posible mediante la iluminación y texturas realistas y aumentando la complejidad de los mismos. En la animación de personajes surge la técnica de *rigging* basada en la representación de los modelos mediante esqueletos articulados. Además, dichos modelos también se crean a partir de escaneados 3D de actores. El realismo en las superficies se logró gracias a la adopción de los *shaders* y de técnicas como el *bump mapping*.

Con las conexiones a Internet de banda ancha, nacen nuevas formas de distribución de juegos de manera digital. En 2003 nace *Steam* de manos de Valve, creador de *Half-Life*, convirtiéndose en la mayor fuente de venta de juegos para ordenador. *Steam* también tuvo un papel importante gracias a la acogida temprana de juegos independientes o *indies*, por lo que se convirtió en uno de los lugares más relevantes para la adquisición de estos juegos.

Panorama en España

En Europa, la industria de los videojuegos nace en los años 80 con los microordenadores MSX, Spectrum, Commodore y Amstrad como protagonistas. En España, unos años más tarde, el empresario José Luis Domínguez vio una oportunidad de mercado en el sector de los videojuegos y creó la compañía Indescomp. De ella nacen en 1983 los videojuegos *La pulga*, el primer videojuego de la historia desarrollado en España, y *Fred* (Ramos, 2017), que tuvieron bastante éxito, y no solo a nivel nacional. En este contexto se dice que nace la edad de oro del software español (1983-1991), en el que José Luis Domínguez fue una de las personalidades más relevantes (blackmores, 2017). Un año más tarde, nacen las empresas Dinamics y Erbe. La primera de ellas desarrolló juegos como *Fernando Martín Basket Master* (1987), Erbe, en cambio, se encargaba de la distribución e importación de títulos. De estas

compañías surgen otras que también tuvieron bastante relevancia en el panorama de la Edad de Oro del software español, como Opera Soft, Made in Spain, Topo Soft, Rebel Act Studio y Pyro Studio. Otros juegos que fueron bastante exitosos en la época son: *La abadía del crimen* (Francisco Menéndez y Juan Delcán, 1988), *Navy Moves* (Dinamic, 1988), *PC Fútbol 1.0* (Dinamic Multimedia, 1992) o *PC Basket* (Dinamic Multimedia, 1992) (*Historia de los videojuegos en España*, 2021).

La llegada de los procesadores de 16 bits, el PC y las producciones norteamericanas afectó negativamente a las empresas de desarrollo de videojuegos de nuestro país, lo que hizo que muchas de ellas desaparecieran al no poder adaptarse al cambio. Pocos años más tarde la industria comienza a resurgir con *Commandos: Behind Enemy Lines* (Pyro Studio, 1998), uno de los títulos más exitosos de la historia del desarrollo de videojuegos en España. Durante la década de 2000-2010 surgen nuevas empresas y se crean secuelas de algunos de los juegos más conocidos de la década anterior. Otros títulos destacados de la última década son: *Castlevania Lords Of Shadow* (Mercury Steam, 2010), *Unepic* (@unepic_fran, 2011), *Metroid Samus Returns* (Mercury Steam, 2017), *Gris* (Nomada Studio, 2018), *Temtem* (Crema, 2020) (Desarrollo Español de Videojuegos (DEV), 2021).

2.3. Librerías específicas.

En el desarrollo de un videojuego hay numerosos aspectos que se deben tratar. Para ello se hace uso de diferentes librerías, SDKs y APIs que nos facilitan estas tareas. Podemos agruparlas en relación a su funcionalidad o propósito.

2.3.1. Propósito general. Estructuras de datos y algoritmos.

Los juegos, como cualquier software, necesitan de estructuras de datos y algoritmos para gestionar y manipular los propios datos. En este apartado destaca la librería estándar de C++ y la librería estándar de plantillas de C++ (STL), aunque también se añaden otras alternativas como Boost y Folly.

En primer lugar, encontramos Boost, que es un conjunto de bibliotecas de código abierto de C++. Está disponible para gran número de sistemas operativos. Incluye implementaciones de estructuras de datos, algoritmos de C++11, C++14 y C++17, manejo de hilos, bibliotecas matemáticas, gestión de entrada/salida, etc. (*Boost Library Documentation*, s. f.).

Otra librería relevante basada en componentes es Folly (Facebook Open Source Library). Pretende extender la biblioteca estándar de C++ y Boost teniendo en mente conseguir el máximo rendimiento y eficiencia. Es de uso gratuito y puede ejecutarse en Windows, Linux y macOS (*facebook/folly*, 2021).

2.3.2. Librerías específicas

A continuación se presentan librerías, SDKs y APIs ideados para la programación de aspectos concretos que forman parte de los videojuegos.

Gráficos

En primer lugar, podemos destacar OpenGL (Open Graphics Library), que es una API de gráficos 2D y 3D implementada por Khronos muy conocida y usada en el desarrollo de aplicaciones gráficas. Es independiente del sistema de ventanas y del sistema operativo (*OpenGL - The Industry's Foundation for High Performance Graphics*, 2011). OpenGL se compone de procedimientos y métodos que permiten el manejo de *shaders*, de objetos y operaciones involucradas en la generación de imágenes gráficas de alta calidad. Además de esto, controla el dibujado de formas geométricas o primitivas como puntos, líneas o polígonos (Segal & Akeley, 2019).

Por otra parte, encontramos Vulkan, una API creada por la misma compañía que OpenGL en 2017. Permite el acceso altamente eficiente y multiplataforma a las GPU actuales utilizadas en ordenadores, consolas y teléfonos inteligentes (Vulkan, 2021). Se

caracteriza por ser de más bajo nivel que OpenGL y otorgar un mayor grado de control a la aplicación. Esto hace que la carga de CPU sea mínima debido a que la capa de abstracción es bastante fina (*KhronosGroup/Vulkan-Samples*, s. f.). También permite el control de los recursos que comparten GPU y CPU como memoria. Vulkan no solo se usa para aplicaciones gráficas, sino también para aquellas que hacen uso de la GPU para cómputo.

En la Figura 4. *Vulkan: Performance, Predictability, Portability* [Figura], 2019, Github se puede observar una comparativa entre ambas APIs y las ventajas que ofrece Vulkan sobre OpenGL.

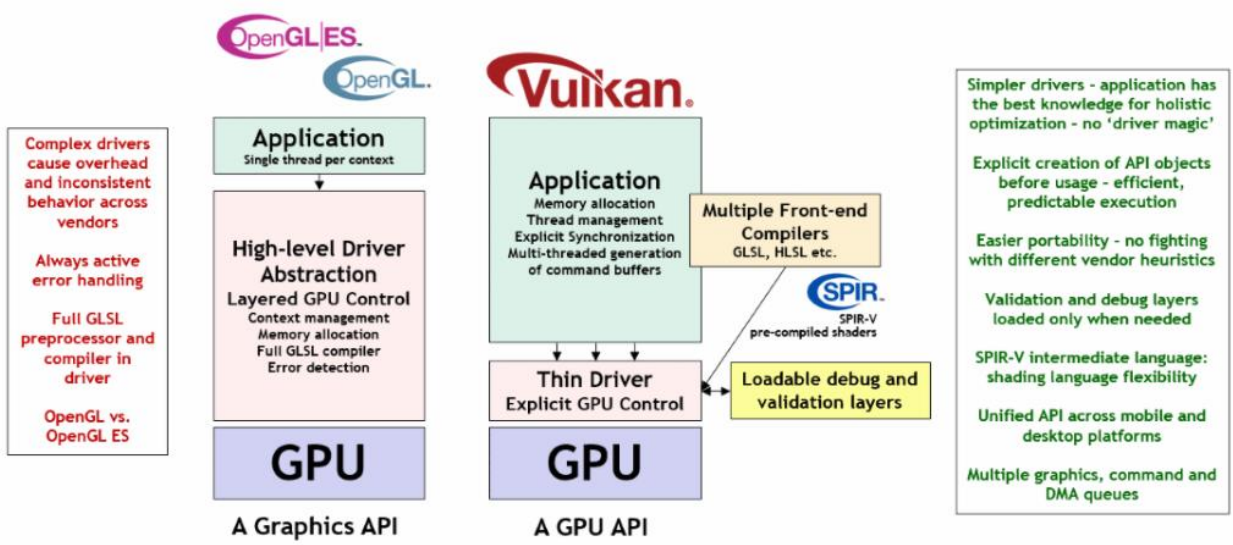

Figura 4. *Vulkan: Performance, Predictability, Portability* [Figura], 2019, Github (https://github.com/KhronosGroup/Vulkan-Guide/blob/master/images/what_is_vulkan_compared_to_gl.png)

Otra API muy conocida es Direct3D, que forma parte del SDK de Microsoft DirectX y permite el dibujado de primitivas con el pipeline de rendering o la realización de cálculos en paralelo mediante el *computer shader*. Comparte el mismo enfoque que Vulkan en cuanto la gestión de recursos, que también la delega en la aplicación (*Getting Started with Direct3D*, 2018).

Por último, encontramos Metal, la API desarrollada por Apple para sus dispositivos iOS, macOS y tvOS. Metal permite el renderizado de gráficos 3D avanzados y el cálculo en paralelo mediante el uso del procesador gráfico (*Metal | Apple Developer Documentation*, s. f.).

Colisiones y física

Dentro de este apartado destaca PhysX, que es un SDK de física escalable y ampliable, multiplataforma y de código abierto. Está desarrollado por la compañía NVIDIA y podemos encontrarlo en motores como Unity y Unreal Engine (*NVIDIA PhysX SDK 3.3.4 Documentation*, s. f.). PhysX es apropiado para aplicaciones interactivas 3D

en tiempo real donde se antepone el rendimiento a la precisión, por lo que es más usado en aplicaciones multimedia como los videojuegos que en aplicaciones del ámbito científico. PhysX se integra con otras librerías desarrolladas por NVIDIA como NvCloth, para la simulación de ropa y textil; Flex, para simulaciones mediante la representación de objetos mediante partículas; Flow, para simular fuego, humo y combustibles y Blast, para imitar el comportamiento de destrucción. Algunas características que incluye este SDK son ():

- Simulación multihilo.
- Múltiples algoritmos de detección de colisiones.
- Detección de colisiones de mallas convexas, triangulares y de formas primitivas.
- Detección de colisiones discretas y continuas.

Por otra parte, encontramos Bullet, una librería en C++ de código abierto para detección de colisiones y dinámica de cuerpos rígidos. Bullet también está disponible para Python con el nombre de PyBullet. Proporciona simulación de dinámica directa, cálculo de dinámica inversa, cinemática directa e inversa, detección de colisiones y consultas de intersección de rayos (Coumans & Bai, 2021).

Por último, se puede destacar Havok Physics, un SDK propietario que ha sido utilizado en algunas de las producciones más conocidas. Ofrece un alto rendimiento gracias a la suspensión de cuerpos rígidos inactivos, además de tener una alta calidad en la simulación (*Havok Physics for Unity*, s. f.). En la simulación de dinámica, incluye detección de contacto robusta incluso en grandes mundos de juego, adaptada para numerosos casos de uso. Además, realiza la detección de colisiones de forma continua para entornos, personajes y vehículos (*Havok Physics*, s. f.).

Audio

FMOD es una herramienta propietaria para añadir sonidos y música a cualquier videojuego. Está orientada para productores, diseñadores y programadores, además de incluir una API en C++/C# de audio 3D para Mac, iOS, Android y Windows (*FMOD Studio*, 2021). Esta API cubre los aspectos como los canales de audio, los sonidos, la grabación de audio, el procesamiento de señales digitales (DSP) o el audio 3D y la oclusión. También incluye el concepto de “sonidos virtuales”, que son los que se reproducen pero no se escuchan debido a la distancia a la que se encuentran (*FMOD - Core API Guide*, 2021).

En cuanto al sistema de audio 3D, una de las características más interesantes que encontramos en esta API, consiste en situar los sonidos en el entorno tridimensional, de forma que se “muevan” alrededor del oyente gracias a la simulación del efecto Doppler (cambio de la frecuencia de la onda con respecto a un actor en movimiento),

la atenuación del volumen o filtrados especiales. Por ejemplo, para la oclusión, se usa un filtro de paso bajo para simular cómo los sonidos atraviesan los objetos.

La herramienta FMOD se puede integrar en motores como Unreal Engine, Unity y Cry Engine.

Otra opción que podemos encontrar es Wwise, que es una herramienta multiplataforma muy similar a la anterior. También está orientada a varios perfiles dentro de la creación de videojuegos, aunque la parte más interesante es el SDK que proporciona en C++ para desarrolladores (*Wwise*, 2021). Además de las funcionalidades más básicas, Wwise incluye un módulo para la gestión del audio espacial, permitiendo el modelado de la propagación del sonido mediante algoritmos de difracción, reflexión y transmisión del mismo (*Wwise SDK 2021.1.1 - Spatial Audio*, 2021).

A diferencia de FMOD, Wwise incluye una opción para obtener la licencia gratuita para pequeñas producciones o para el ámbito académico.

Por último, podemos señalar la alternativa open source, OpenAl (Open Audio Library), una API de audio 3D multiplataforma (*OpenAL: Cross Platform 3D Audio*, s. f.). Su funcionamiento es similar al de OpenGL y se basa en tres objetos básicos: Listener (que representa la posición desde la que se escucha el audio), Source (cada fuente de audio) y Buffer (que almacena los sonidos y puede asignarse a diferentes fuentes). De las tres alternativas que se han analizado, esta es la que precisa de un mayor grado de experiencia, debido a que es de más bajo nivel que las anteriores.

Inteligencia Artificial

En el desarrollo de videojuegos, una de las cuestiones que se deben abordar es el comportamiento de los personajes que no son jugadores (NPC – *Non Player Character*). Para ello se usan algoritmos de inteligencia artificial de cálculo de trayectorias, entre otros, que forman parte de las herramientas que se van a nombrar a continuación.

Path Engine es un SDK propietario para la implementación de movimiento inteligente de los agentes. Se basa en la búsqueda de trayectorias mediante puntos de visibilidad en entorno 3D. El modelo del movimiento define cómo interactúan los agentes y los obstáculos en estas superficies y, cómo los obstáculos y los bordes de las superficies limitan el movimiento de los agentes. Además de ello, realiza estos cálculos de forma eficiente, siendo adecuado para juegos multijugador masivo en los que el mundo es complejo y de gran tamaño (*PathEngine - Intelligent agent movement*, 2021).

También encontramos herramientas multiplataforma de código abierto como Recast/Detour, que permiten obtener mallas de navegación de forma automática y rápida, mediante la voxelización y simplificación de la geometría del mundo (*Recast & Detour*, 2021).

Multijugador y red

En este apartado podemos destacar Photon, que es un framework de desarrollo para construir juegos multijugador en tiempo real para varias plataformas en diversos lenguajes de programación. Incluye un SDK de la parte del servidor y otro para la parte del cliente. Permite gestionar las partidas y asociar los clientes a dichas partidas, la persistencia de los datos de las partidas o añadir la funcionalidad de chat tanto de voz como texto al juego. Está disponible para la mayoría de plataformas y motores (*Realtime Intro - Photon*, s. f.).

2.3.3. Herramientas generales.

Además de las librerías, APIs y SDKs mencionadas anteriormente, diseñadas específicamente para una tarea, podemos encontrar herramientas genéricas que agrupan varios aspectos y que hacen más sencillo el desarrollo de juegos.

En primer lugar, tenemos la opción open source SFML (Simple and Fast Multimedia Library). Proporciona una interfaz sencilla para facilitar el desarrollo de juegos y aplicaciones multimedia. Está disponible para Windows, Linux, macOS y próximamente Android e iOS. Consta de diversos módulos que pueden utilizarse por separado o de forma conjunta para la creación de juegos (*Tutorials for SFML 2.5*, s. f.). Estos son:

- Módulo del sistema operativo, para cuestiones como la gestión de hilos o la carga de archivos.
- Gestión de ventanas, eventos y entradas de diferentes dispositivos.
- Módulo de gráficos, basado en OpenGL, que permite el dibujado 2D de formas, uso de *shaders* o el control de la cámara 2D.
- Módulo de audio, encargado de la reproducción y grabación de sonidos, además de incluir opciones para producir la espacialización del sonido 3D.
- Módulo de red, que hace posible la comunicación mediante sockets, realizar peticiones HTTP y la transferencia de archivos mediante FTP.

Otra librería bastante similar a SFML es SDL (Simple DirectMedia Layer), que también es de código abierto y multiplataforma. Permite el acceso al hardware de audio, gráficos y dispositivos de entrada (ratón o teclado). Cubre los mismos módulos que la

anterior excepto la parte de red. Además, incluye funcionalidades extra como, por ejemplo la gestión de sensores y dispositivos hápticos, y también incluye la integración con Direct3D, Vulkan y OpenGL (*SDL Wiki*, s. f.).

2.4. Entornos de desarrollo específicos.

En este apartado podemos realizar una distinción entre entornos que trabajan junto a motores de videojuegos y los que permiten crear un juego desde cero, aunque, realmente, crear un juego desde cero implica construir un pequeño motor.

Rider

En primer lugar, encontramos Rider, que es un entorno de desarrollo .NET creado por la compañía JetBrains basado en el IDE IntelliJ y en la herramienta ReSharper. Puede ser ejecutado en Windows, MacOS y Linux. Aunque el precio de la licencia excede los 100 euros, es gratuito para estudiantes y docentes. Rider puede integrarse como IDE en Unity y Unreal Engine para el desarrollo de código en C# y C++, respectivamente (*Rider*, 2021).

En el caso de la integración con Unity, algunas de las características que encontramos son (*Rider para Unity*, 2021):

- Potente, incluye 2500 inspecciones y refactorizaciones para que el proceso de escribir código sin errores sea mucho más eficiente.
- Configuración automática para enlazar Unity con Rider como editor de código C# y *shaders*.
- Contiene las opciones del editor de Unity para controlar la ejecución y pausa del juego, de forma que se puede manejar esta ejecución desde Rider.
- Aporta consejos gracias al análisis del código específico para desarrollo de videojuegos.
- Incluye opciones de depuración y, además puede descompilar bibliotecas para poder depurar el código, establecer puntos de interrupción y visualizar los valores de las variables.
- Permite ejecutar los test unitarios de la API de Unity, visualizar los resultados, filtrarlos y ver la traza de la pila.
- La pestaña de *logs* permite navegar por la pila de eventos y acceder a los archivos y métodos involucrados en estos.
- Ayuda a controlar y mejorar el rendimiento. Destaca las funciones y procedimientos más costosos dentro de los métodos que se ejecutan con mayor frecuencia, como *Update* (Figura 5. *Análisis de rendimiento de métodos en Rider* [Figura]). Además, detecta patrones de programación menos eficientes y sugiere correcciones.

- Permite buscar los usos de clases o métodos dentro de escenas o recursos, muestra su ubicación de forma jerárquica y permite acceder a ellos.
- Es compatible con los archivos que contienen los *shaders* ofreciendo herramientas para su sintaxis.
- Contiene herramientas para el control de versiones.

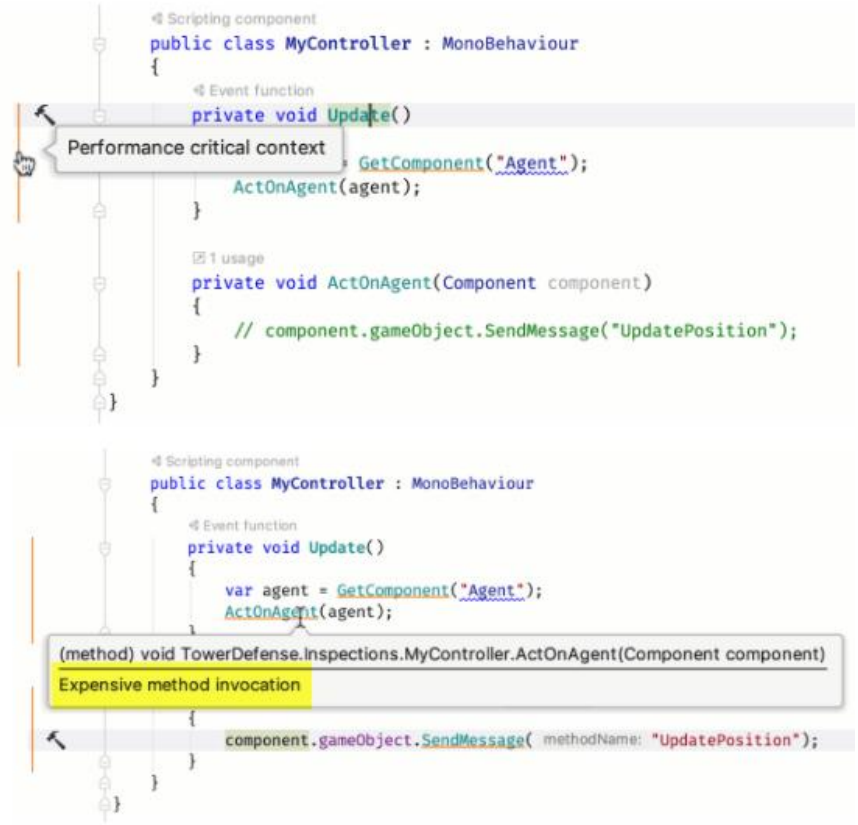

Figura 5. *Análisis de rendimiento de métodos en Rider* [Figura], 2021, JetBrains, https://www.jetbrains.com/dotnet/promo/unity/

En el caso de la integración con Unreal Engine, aún no se encuentra disponible totalmente, sino que se puede solicitar un acceso anticipado hasta su lanzamiento a finales de 2021. Rider para Unreal contiene una funcionalidad similar a la versión para Unity, con potentes herramientas para ayudar a la escritura de código sin errores y refactorización del mismo. Además, permite navegar por los Blueprints (generación de código visual) y encontrar los usos en los archivos de estos. Por otra parte, proporciona consejos para seguir la nomenclatura estándar de Unreal (*The .NET Tools Blog*, 2020).

Visual Studio

El otro entorno de desarrollo usado por excelencia es Visual Studio, de la compañía Microsoft. Puede incorporarse a múltiples motores como Unity, Unreal o Cocos, además de proporcionar herramientas para manejar las cuestiones gráficas usando la

API de DirectX. Visual Studio está disponible para Windows y macOS, pero no para distribuciones Linux. Contiene las funcionalidades más genéricas de los IDEs actuales, como el control de versiones o la refactorización y navegación por las llamadas.

Visual Studio, en su integración con los motores Unity, Unreal Engine y Cocos, incluye los asistentes de escritura de código para la escritura rápida y sin errores (Figura 6. *Asistente de escritura de código para Unity*) basados en la tecnología IntelliSense, además de poder depurar los proyectos y controlar el rendimiento del programa.

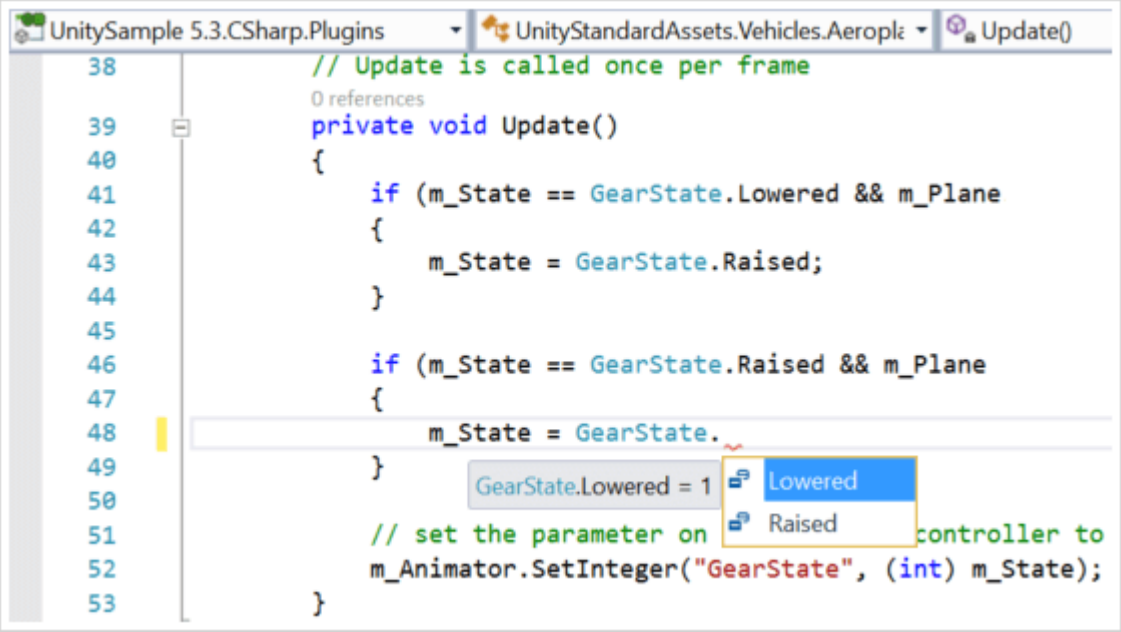

Figura 6. *Asistente de escritura de código para Unity* [Figura], 2021, Visual Studio, https://visualstudio.microsoft.com/es/vs/unity-tools/

En cuanto al desarrollo de juegos con DirectX mediante la Plataforma Universal de Windows (UWP), hay numerosos recursos para cada una de las tareas que conlleva. Concretamente, en la parte gráfica del desarrollo con DirectX, Visual Studio incluye plantillas y herramientas específicas para el desarrollo de juegos que usan Direct3D. Contiene un editor de imágenes, modelos y *shaders*. Por otra parte, incorpora funcionalidad para el control de la parte gráfica mediante la depuración de *shaders* de forma sencilla, la depuración de fotogramas y el análisis del rendimiento del dibujado de los mismos. También permite monitorizar el uso de la propia GPU a través del análisis de los fotogramas, para saber cuáles son las llamadas de dibujo menos eficientes y dónde se encuentran los cuellos de botella (*Desarrollo de juegos con Visual Studio*, 2021).

Eclipse

Por último, podemos destacar Eclipse, de parte de IBM y Eclipse Foundation, que es conocido por ser uno de los IDEs más utilizados en el desarrollo de código Java. Sin embargo, también incluye la posibilidad de trabajar en otros lenguajes como C/C++, JavaScript/TypeScript o PHP. Está disponible para Windows, macOS y Linux, además de ser gratuito y de código abierto. Eclipse provee un núcleo base en el que permite añadir nueva funcionalidad mediante plugins que pueden desarrollar terceros, como, por ejemplo, para incluir el control de versiones o herramientas para la mejora de la calidad del código. Además, proporciona las herramientas necesarias para la depuración del código (Gurr, 2020).

2.5. Motores de juegos. Funciones y componentes.

En este apartado se va a definir qué es un motor de juegos para, posteriormente, describir la forma en la que se organizan y estructuran los motores mediante componentes y qué función tiene cada uno de estos. Por último, se ha incluido una sección en la que se detallan las características de tres motores actuales bastante conocidos (CryEngine, Godot y Unreal Engine) y algunos juegos conocidos que han sido creados en estos.

2.5.1. Definición de motor de juegos

Históricamente, el concepto de motor de juegos nace en 1993 con el videojuego *Doom* de la empresa id Software, debido a que, en su diseño y creación, la motivación fue mantener separados los diferentes módulos y componentes. De esta forma, la parte software más a bajo nivel era independiente de la parte más enfocada al arte y mecánicas del juego, lo que posibilitaba la reutilización de los componentes. Este enfoque revolucionó la industria del videojuego, ya que se comenzaron a crear nuevos juegos con reglas y arte totalmente diferentes sin tener que realizar cambios en el motor del juego, además de potenciar la cultura de los *mods* (Gregory, 2018). De esta forma, el motor se encarga de proporcionar el software necesario para tareas comunes como el renderizado, la gestión de entradas de usuario, la parte de red para juegos multijugador, el audio, etc.

A finales de la década de los 90, ya se comenzó a considerar las licencias concedidas de los motores como una fuente de ingresos para las grandes empresas, mientras que, por otra parte, la reutilización de motores de videojuegos de terceros ahorraba tiempo y dinero a muchas compañías y desarrolladores, lo que hacía que fuera una opción bastante viable, tanto a nivel económico como a nivel de resultado final (Vallejo Fernández & Martín Angelina, 2015).

La distinción entre un juego y su motor no es trivial, debido a que la mayoría de las veces va a haber dependencias, como, por ejemplo, las que impone el género del propio juego. Los motores que se usan para desarrollar juegos de disparos en primera persona o FPS no siempre van a resultar tan adecuados para juegos de otros géneros, como los de carreras, debido a que, los motores se suelen desarrollar con características idóneas para los requerimientos del género en cuestión (Mercado, 2021). En el caso de los motores más genéricos, o de propósito general, se puede afirmar que resultan menos optimizados, debido a que, para que fueran más eficientes, deberían adaptarse a las necesidades que requiere cada caso o, al hardware en el que se ejecutará. Aunque, con las últimas mejoras en hardware y software, las diferencias que existe entre los motores dependiendo del género al que están orientados es cada vez menor (Gregory, 2018).

La mayoría de los motores de juegos que existen en la actualidad están desarrollados en C++ debido a que es un lenguaje que permite el paradigma de programación orientado a objetos, uno de los enfoques más utilizados en el desarrollo de motores y juegos. Otra característica de este lenguaje es la posibilidad de la gestión de memoria a bajo nivel, lo que permite obtener mayor rendimiento en la programación.

2.5.2. Componentes. Sistema de capas.

Los motores de videojuegos se construyen siguiendo una arquitectura en capas. Este enfoque se caracteriza porque cada una de estas capas tiene una función específica y cada capa depende, a su vez, las capas inferiores, pero no al revés. De esta manera, se puede modificar una capa superior sin que estos cambios afecten a las inferiores. En este caso, la parte característica del propio juego se encontrará en las últimas capas.

La imagen siguiente (Figura 7. *Visión conceptual de la arquitectura general de un motor de juegos*) representa un modelo simplificado de todo el sistema basado en capas de un motor de videojuegos, que se va a concretar a continuación (Vallejo Fernández & Martín Angelina, 2015).

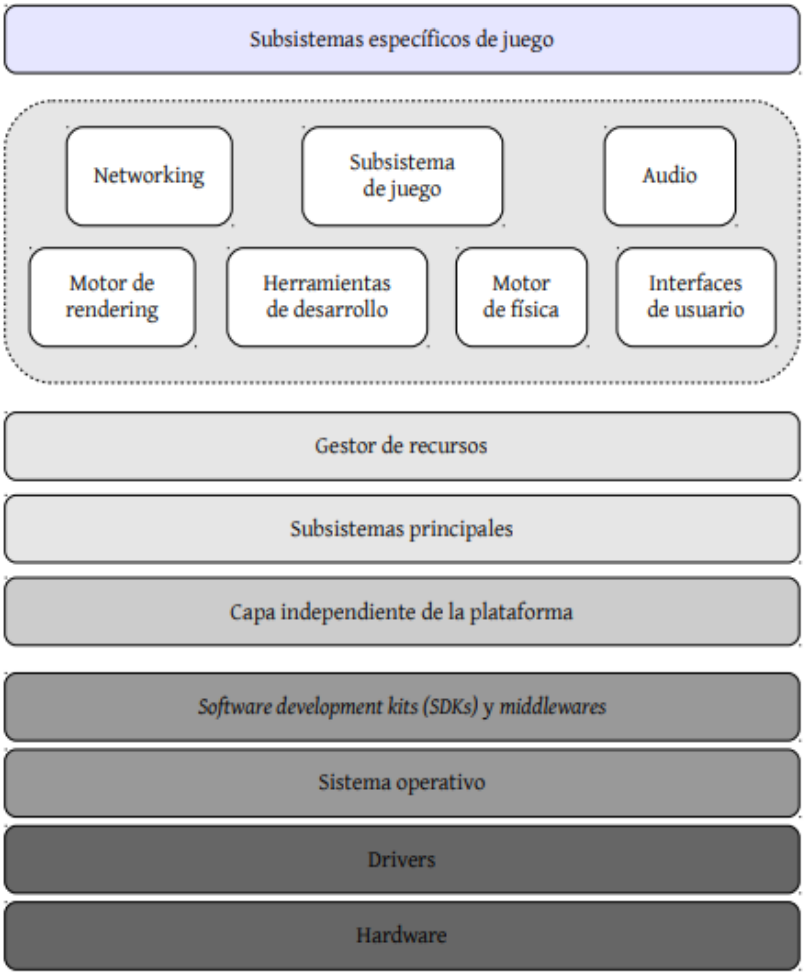

Figura 7. *Visión conceptual de la arquitectura general de un motor de juegos*, extraído de *Desarrollo de Videojuegos: Un Enfoque Práctico* (p. 15), por V. Barba Pizarro, 2015.

Hardware

En primer lugar, encontramos la capa del hardware correspondiente a la máquina en la que se va a ejecutar el juego. Puede ser un ordenador, una consola, un smartphone o una consola portátil.

Drivers

Esta capa hace de interfaz entre el hardware y el sistema operativo. Son los controladores que permiten abstraer los detalles físicos al sistema operativo. Por ejemplo, es necesario el uso de drivers para usar la tarjeta gráfica o la tarjeta de sonido.

Sistema operativo

Es el software que controla la ejecución de los programas (como los juegos) y provee servicios como la asignación de recursos, el control de la entrada y salida y la gestión de los datos. Las primeras consolas no contenían sistema operativo, sino un simple *firmware*, y el juego era el encargado de controlar y gestionar los elementos hardware. Sin embargo, las consolas más modernas sí que lo incluyen, permitiendo interrumpir la ejecución del juego o apropiarse de algunos recursos, permitiendo así acciones como la recepción de mensajes de chat en línea, la descarga de juegos o contenidos y la grabación del desarrollo de la partida, entre otras.

Kits de Desarrollo Software (SDK) y *middleware*

Como todos los sistemas, los motores de videojuegos utilizan librerías, *frameworks*, APIs y SDKs para proporcionar funcionalidades. En la siguiente imagen (Figura 8. *Capa SDK y middleware*.) aparecen algunas de las más utilizadas, aunque esta cuestión se ha tratado anteriormente, con mayor detenimiento, en el apartado de

Librerías específicas.

Figura 8. *Capa SDK y middleware*. Adaptada de *Runtime game engine architecture* [Figura], 2018, Jason Gregory, https://www.gameenginebook.com/figures.html

Capa de independencia de la plataforma

La mayoría de videojuegos que encontramos hoy en el mercado están disponibles para múltiples plataformas (ordenadores, consolas portátiles y de sobremesa…) para así obtener el mayor beneficio económico posible. Esta capa (Figura 9. *Capa de independencia de la plataforma*) posibilita esta característica, abstrayendo los elementos que dependen de la plataforma a las capas superiores. Dicha abstracción es necesaria debido a que, algunas APIs de programación, como las que proporcionan los sistemas operativos, y librerías, cambian dependiendo de cada plataforma.

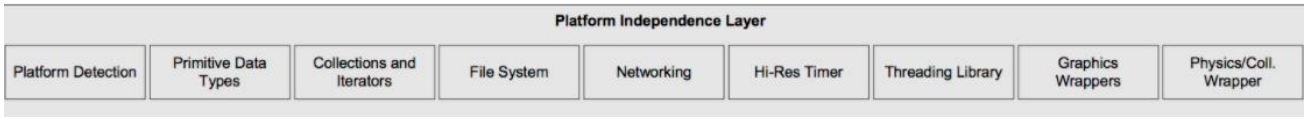

Figura 9. *Capa de independencia de la plataforma*. Adaptada de *Runtime game engine architecture* [Figura], 2018, Jason Gregory, https://www.gameenginebook.com/figures.html

Subsistemas principales

Cada motor necesita de una serie de utilidades software que le dan soporte al mismo. Como puede observarse en la Figura 10. *Capa de sistemas principales*., estas utilidades o subsistemas pueden estar enfocados al desarrollo de juegos o ser más genéricos, como los que encontramos en un proyecto software con mayor complejidad. Algunos de estos subsistemas son:

- Módulo de inicialización y finalización. Es uno de los más importantes, ya que se encarga de inicializar y finalizar de manera adecuada cada subsistema que compone el motor en un orden específico, teniendo en cuenta las dependencias que puedan existir entre ellos.
- Aserciones. Este subsistema se encarga de las expresiones que sirven para controlar los errores en la lógica y en las suposiciones del programador. Sirven en las tareas de diseño y desarrollo del programa.
- Gestión de la memoria. La gran mayoría de motores implementa este sistema para controlar la asignación y liberación de la memoria y así evitar la fragmentación de la memoria y sus consecuencias.

- Librería matemática. Debido a todos los elementos que componen un juego, se hace necesario un sistema encargado de manejar los aspectos matemáticos. Estos incluyen tratamiento de vectores, matrices, rotaciones con cuaterniones, operaciones con la geometría…
- Estructuras de datos y algoritmos personalizados.
- Lectura y escritura asíncrona de archivos. Este subsistema permite cargar datos en segundo plano mientras el juego sigue funcionando. Estos datos pueden hacer referencia a sonidos, texturas, configuración de niveles, geometría…

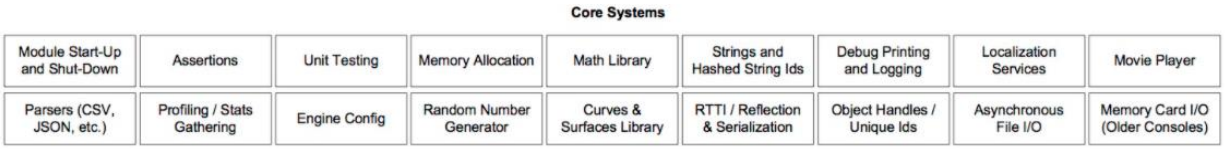

Figura 10. *Capa de sistemas principales*. Adaptada de *Runtime game engine architecture* [Figura], 2018, Jason Gregory, https://www.gameenginebook.com/figures.html

Gestor de recursos

Provee una abstracción para acceder a los *assets* o activos del juego (Figura 11. *Gestor de recursos*.), como modelos, texturas, animaciones, scripts… En este punto, hay motores que gestionan los recursos de forma centralizada y otros que delegan esta tarea en el programador.

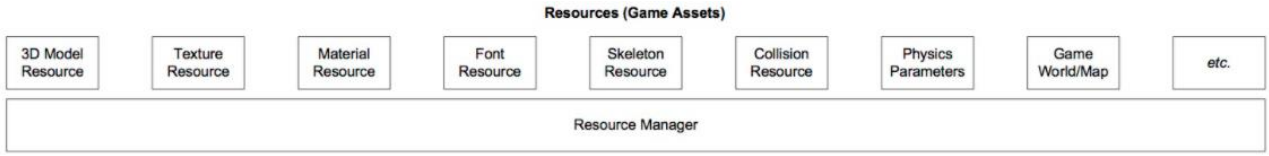

Figura 11. *Gestor de recursos*. Adaptada de *Runtime game engine architecture* [Figura], 2018, Jason Gregory, https://www.gameenginebook.com/figures.html

Motor de renderizado o *rendering*

Es una de las partes más importantes e indispensables de cualquier motor de juego debido a la importancia de la parte gráfica implícita que poseen los juegos. El motor de renderizado, a su vez, se compone de otros subsistemas, que suelen estar organizados siguiendo un enfoque arquitectónico basado en capas. Estos son los subsistemas de los que se compone:

- **Renderizador de bajo nivel** (Figura 12. *Renderizador de bajo nivel*.): es la capa inferior del motor de renderizado y se encarga de dibujar las primitivas de las que se componen los modelos de forma rápida y "en bruto", sin considerar qué partes son visibles desde la cámara y cuáles quedan ocultas. A su vez, esta capa

actúa como interfaz con los dispositivos gráficos disponibles mediante los SDKs gráficos como OpenGL, Vulkan o Direct3D. Además, podemos encontrar otros componentes que también participan en el proceso de renderizado de la escena, como las diferentes luces que componen la iluminación de la escena, las cámaras que determinan qué se visualiza de la escena, los materiales y texturas asociados a las primitivas, el dibujado de letras…

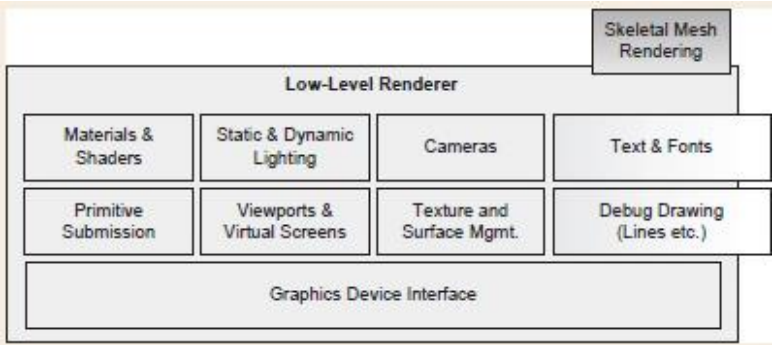

Figura 12. *Renderizador de bajo nivel*. Adaptada de *Runtime game engine architecture* [Figura], 2018, Jason Gregory, https://www.gameenginebook.com/figures.html

- **Optimizaciones y recortes de la escena** (Figura 13. *Optimizaciones y recortes de escena*.)**:** esta capa está justo por encima del renderizador de bajo nivel y se encarga de limitar las primitivas que después pasará a dibujar este. Estas optimizaciones se basan en el cálculo de la visibilidad de la escena desde el punto de vista de la cámara y son muy importantes en la mejora de la eficiencia del propio motor, ya que existen numerosos juegos, como los *shooters* en primera persona, en los que los fotogramas por segundo son críticos. En esta capa se incluyen estructuras de datos espaciales y algoritmos geométricos muy útiles para estos cálculos de visibilidad y ocultamiento como se observa en la imagen.

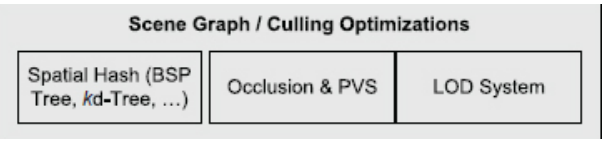

Figura 13. *Optimizaciones y recortes de escena*. Adaptada de *Runtime game engine architecture* [Figura], 2018, Jason Gregory, https://www.gameenginebook.com/figures.html

- **Efectos visuales** (Figura 14. *Efectos visuales*.)**:** los motores incluyen una amplia variedad de efectos visuales como los sistemas de partículas, iluminación HDR, mapeo de la iluminación y sombreado dinámico, iluminación PRT o efectos "a posteriori". Estos efectos son relevantes a la hora de aportar realismo a la escena y conseguir la sensación de inmersión que se intenta alcanzar. Por ejemplo, los sistemas de partículas permiten representar humo, llamas o la trayectoria de un proyectil.

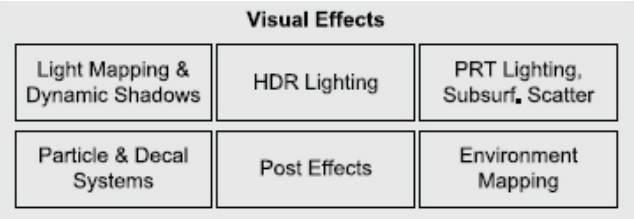

Figura 14. *Efectos visuales*. Adaptada de *Runtime game engine architecture* [Figura], 2018, Jason Gregory, https://www.gameenginebook.com/figures.html

- **Front-end** (Figura 15. *Front-end*.)**:** hace referencia a la interfaz gráfica con la que interactúa el jugador a lo largo de la ejecución del juego. Son elementos en dos dimensiones que se superponen al mundo 3D. También incluye módulos para visualizar videos o cinemáticas en el juego.

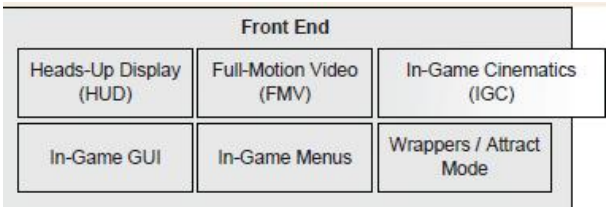

Figura 15. *Front-end*. Adaptada de *Runtime game engine architecture* [Figura], 2018, Jason Gregory, https://www.gameenginebook.com/figures.html

Herramientas de depuración y análisis del rendimiento del software

Aunque en casi cualquier proyecto software, es necesario el uso de herramientas que permitan la depuración y optimización del código, en el desarrollo de videojuegos esta característica es indispensable debido a su naturaleza. En los juegos es importante el control de los recursos, por ejemplo, de la memoria, y estas herramientas son una pieza clave en este proceso. La práctica más habitual en los motores de juegos es incluir sus propias herramientas para estas cuestiones en vez de usar las que proporcionan terceros. Algunas de las tareas que realizan son (Figura 16. *Herramientas de depuración y análisis del rendimiento de software*.):

- Monitorización de los recursos que consume cada subsistema del motor.
- Medir el tiempo que tarda en ejecutar cada parte del código.
- Visualizar las estadísticas de rendimiento durante la ejecución.
- Grabar instantes en el juego, un aspecto muy valioso para la corrección de posibles errores. Por ejemplo, en la PlayStation 4, siempre se graban los últimos 15 segundos y, cuando ocurre un fallo, se envían a los servidores para su posterior análisis.

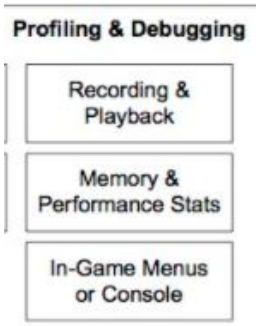

Figura 16. *Herramientas de depuración y análisis del rendimiento de software*. Adaptada de *Runtime game engine architecture* [Figura], 2018, Jason Gregory, https://www.gameenginebook.com/figures.html

Sistema de físicas y colisiones

Este sistema (Figura 17. *Sistema de física y colisiones*.) es fundamental en los juegos en los que tenemos un mundo en dos o tres dimensiones en el que los objetos se mueven, ya que proporciona un comportamiento coherente con el mundo. Con la detección de colisiones podemos lograr que los objetos no se atraviesen unos a otros, sino que exista una interacción entre ellos. Este sistema de detección de colisiones, además de detectarlas, permite conocer en qué puntos se ha producido la colisión y cómo se va a resolver esta. En estas tareas se emplean estructuras de datos espaciales como la caja envolvente AABB, y algoritmos geométricos como los test de intersección. Los cálculos implicados en estas tareas deben ser lo más óptimos posible para conseguir la mayor eficiencia posible.

Por otra parte, junto al tratamiento de colisiones, se encuentra el sistema de física, que, en realidad, es una simulación más o menos realista de la dinámica de cuerpos rígidos, de cómo afectan las fuerzas a los objetos. El sistema de física suele estar muy acoplado con el de colisiones y trabajar de forma coordinada con este.

El sistema de físicas y colisiones permite, entre otras cosas, la destrucción de edificios u objetos, cálculo de la trayectoria de un proyectil mediante lanzado de rayos, aporte de realismo a personajes con el movimiento del pelo o la ropa, simulación del comportamiento del agua, etc.

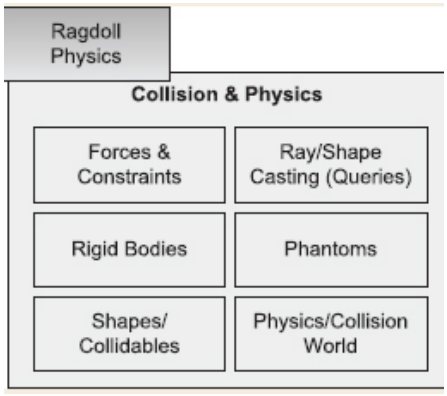

Figura 17. *Sistema de física y colisiones*. Adaptada de *Runtime game engine architecture* [Figura], 2018, Jason Gregory, https://www.gameenginebook.com/figures.html

Animación

El sistema de animación (Figura 18. *Animación*.) es indispensable para juegos que incluyen personajes humanos o humanoides, debido a que estos tienen que moverse de forma natural y fluida. Gracias a este sistema, se aporta una herramienta para que los creadores puedan "deformar" los modelos para conseguir diferentes poses. Podemos encontrar cinco tipos de animación básica en videojuegos: animación por *sprites*, jerárquica de cuerpos rígidos, de esqueleto, de los vértices y *morph* (interpolación). Normalmente se utiliza la animación mediante esqueleto, que consiste en una jerarquía de articulaciones rígidas unidas unas con otras y que pueden moverse unas con respecto a otras. Esta técnica se combina con el *skinning*, que consiste en recubrir todo el esqueleto con una malla de triángulos continua en la que cada vértice puede estar asociado a varias articulaciones, que van a producir el movimiento de los mismos. Para este proceso de *skinning*, tenemos un componente que actúa como intermediario entre el sistema de animación y el renderizador. El sistema de animación pasa las matrices que representan las poses al renderizador y este se encarga de ponerle la piel a los esqueletos.

Aunque en esta parte nos hemos referido a animación de humanoides, el sistema de animación también se puede aplicar a cuerpos articulados, que no sean completamente rígidos.

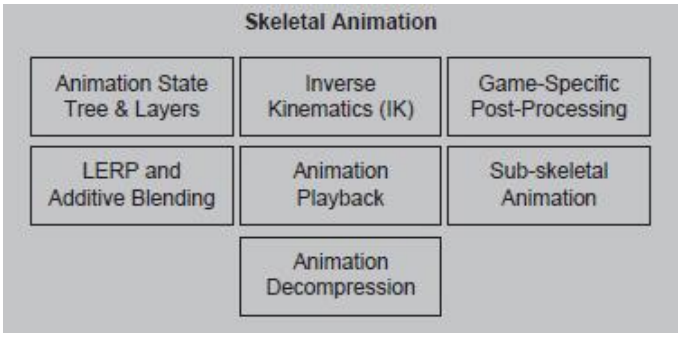

Figura 18. *Animación*. Adaptada de *Runtime game engine architecture* [Figura], 2018, Jason Gregory, https://www.gameenginebook.com/figures.html

Interfaz de usuario

Para lograr la interacción del jugador se utilizan dispositivos de interfaz humana (HID), como el ratón, el teclado, los mandos y otros dispositivos específicos, volantes, la plataforma Wii Fit, etc. Este subsistema tiene una doble función como se puede observar en la Figura 19*. Interfaz de usuario*., ya que por una parte hace de interfaz entre los controladores del hardware a más bajo nivel y los controles de más alto nivel y, por otra parte, se encarga de manejar y tratar estos eventos de entrada, y también salida, ya que a veces también se proporciona una respuesta mediante estos dispositivos. Algunas tareas de las que se encarga son: detección de eventos y secuencias de botones, filtrar las señales analógicas, manejar múltiples dispositivos para varios jugadores...

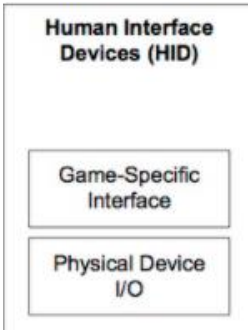

Figura 19. *Interfaz de usuario*. Adaptada de *Runtime game engine architecture* [Figura], 2018, Jason Gregory, https://www.gameenginebook.com/figures.html

Audio

Normalmente, los desarrolladores de juegos no suelen prestar tanta atención al audio como a la parte gráfica o la física, aunque, como afirma Jason Gregory, “Nonetheless, no great game is complete without a stunning audio engine” (“Ningún buen juego está completo sin un buen motor de audio”). El sonido cobra un papel

importante en muchos juegos, debido a que es un elemento que ayuda en la experiencia inmersiva del jugador. De hecho, los programadores de sonido llaman a este motor, motor de renderizado de audio.

Este componente no sólo se encarga del procesamiento de señales digitales y de la reproducción de los sonidos, sino que también suele incorporar un sistema de audio 3D (Figura 20. *Audio*.). Este sistema permite dotar de realismo un entorno en tres dimensiones, dando como resultado múltiples canales de audio que simulan la presencia en el mundo virtual. Para ello realiza tareas como la síntesis de sonido correspondiente a los eventos que ocurren, la espacialización o control del volumen para ubicar el sonido, el modelado acústico, que aplica efectos como la reverberación, etc.

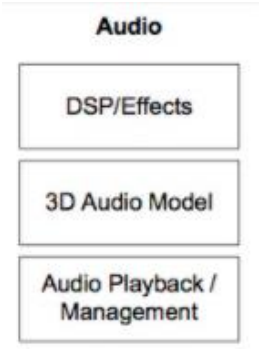

Figura 20. *Audio*. Adaptada de *Runtime game engine architecture* [Figura], 2018, Jason Gregory, https://www.gameenginebook.com/figures.html

Multijugador o juegos en red (*networking*)

Una gran parte de los juegos actuales contiene modos de juego multijugador, ya sea de forma "local" (ambos jugadores comparten espacio físico) o remota, donde varios o muchos jugadores se conectan en red para jugar juntos. El hecho de que un juego sea multijugador hace que haya que diseñar otras partes del motor de forma diferente a la manera en que se haría para un solo jugador. Este motivo influye en el desarrollo del propio motor, ya que se suelen diseñar las funciones multijugador desde el principio y realizar el modo de un jugador como un caso particular del modo multijugador.

El subsistema de red o multijugador (Figura 21. *Juegos en red*.) se encarga de manejar y controlar el modo multijugador e informar a los demás jugadores de los detalles acerca del estado de la partida y de los demás jugadores (replicación). El envío de información se suele realizar mediante *sockets* y se manda la mínima información para agilizar dichos envíos.

Figura 21. *Juegos en red*. Adaptada de *Runtime game engine architecture* [Figura], 2018, Jason Gregory, https://www.gameenginebook.com/figures.html

Subsistemas de juego

Los subsistemas anteriores proporcionan la base para el desarrollo de un juego, sin embargo, no forman un juego por sí mismos, necesitan añadir la jugabilidad, que hace referencia a la acción que tiene lugar en el juego, las reglas que rigen el mundo virtual y las mecánicas de juego. Esta jugabilidad se define en la capa más superior de todas. Los subsistemas de juego aíslan a esta última capa de los sistemas anteriores de más bajo nivel. De estos subsistemas, el principal es el modelo de objetos de juego en tiempo de ejecución, que es una implementación del modelo de objetos que se visualiza en el editor.

En esta capa (Figura 22. *Subsistema de juego*.) se tiene en cuenta los elementos estáticos y dinámicos que pertenecen al mundo del juego. Por ejemplo, la geometría estática del entorno, los personajes tanto jugadores como no jugadores, los cuerpos rígidos dinámicos... Estos elementos se suelen modelar siguiendo el paradigma orientado a objetos.

Otro componente de esta capa es el sistema de eventos, que proporciona un mecanismo de comunicación entre los objetos del juego. El emisor crea una pequeña estructura de datos denominada evento y el receptor la recibe mediante su función encargada de manejar los eventos.

También se encarga de gestionar los niveles del mundo virtual, cargando los datos necesarios para cada uno de ellos.

Para describir la lógica del juego se incluye además un sistema de scripts que permiten a los programadores implementar las reglas y mecánicas. A veces, integran un propio lenguaje más sencillo que C/C++ en el motor o, incluso, un lenguaje de programación visual.

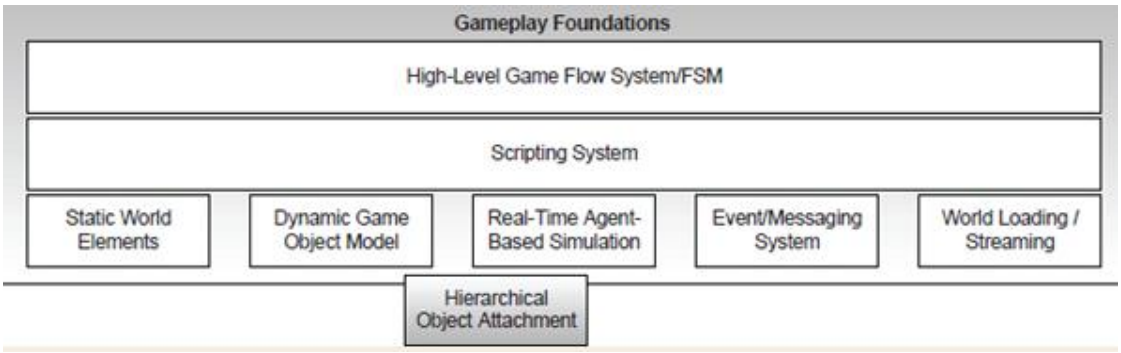

Figura 22. *Subsistema de juego*. Adaptada de *Runtime game engine architecture* [Figura], 2018, Jason Gregory, https://www.gameenginebook.com/figures.html

Subsistemas específicos del juego

La capa más superior de todas (Figura 23. *Subsistemas específicos del juego*.) es en la que los programadores trabajan para implementar las características del juego. Estos subsistemas son muy numerosos y dependen de cada juego. Algunos de ellos hacen referencia a las armas, vehículos, a las cámaras, las mecánicas y reglas que afectan al jugador...

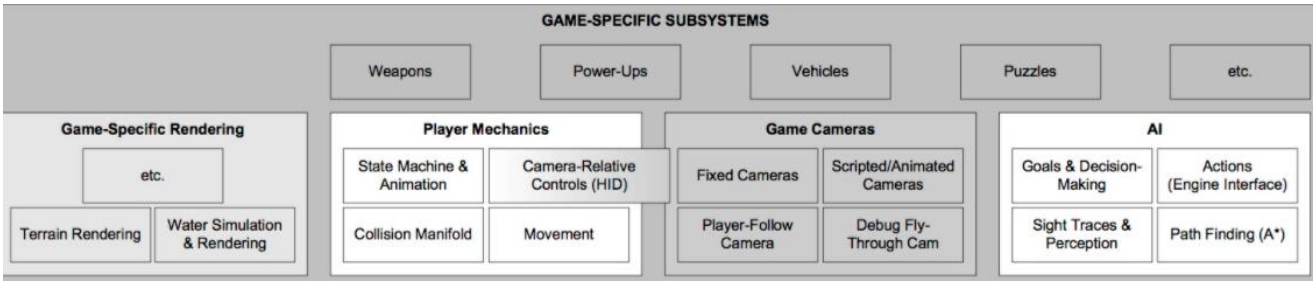

Figura 23. *Subsistemas específicos del juego*. Adaptada de *Runtime game engine architecture* [Figura], 2018, Jason Gregory, https://www.gameenginebook.com/figures.html

2.5.3. Ejemplos de motores y juegos desarrollados.

Actualmente el número de motores de videojuegos que existen es bastante amplio. Por este motivo vamos a destacar algunos de los más conocidos.

Todos los motores que se van a mencionar se basan en el sistema “lo que ves es lo que juegas” (WYSIWYP), en el que se puede editar y previsualizar en tiempo real los aspectos del juego.

En primer lugar, tenemos CryEngine, que es un motor bastante potente y de gran calidad en el desarrollo de videojuegos. Está disponible para todas las plataformas y permite la creación de juegos para Windows, Linux, PlayStation 4, Xbox One, Oculus Rift, OSVR, PlayStation VR y HTC Vive. El uso del motor es gratuito, se paga en el

momento en el que se lanza el juego y comienza a tener beneficios. Además, tanto el código del motor como el del editor están disponibles para los usuarios (*CRYENGINE Support - General*, 2021).

Algunas de las características principales de CryEngine son (*CRYENGINE Features*, 2021):

- En el apartado de gráficos, tenemos técnicas avanzadas para conseguir dotar de realismo los mundos. En cuanto a la iluminación de las escenas, destaca el renderizado basado en física, que simula el comportamiento de la luz en el mundo real o las luces de área, que se modelan como superficies o volúmenes, teniendo un mayor control sobre ellas. También incluye "volúmenes de agua", que permiten simular fielmente el efecto del agua en diferentes formas.
- En animación, el motor proporciona herramientas para dotar de movimientos de forma realista los gestos de la cara, además de incluir la animación mediante esqueleto con parámetros y algoritmos procedimentales para cinemática inversa.
- En Inteligencia Artificial, se añaden mallas de navegación multicapa para que los personajes no jugadores (NPC) avancen por el área evitando atascos. También permite dotar de cierto comportamiento a los NPCs simulando los sentidos como la vista o el oído.
- Permite la integración con middleware como FMOD, Miles Sound System o Wwise, para la parte de audio. También incluye un sistema para la espacialización del audio, simulando las características y propiedades de los sonidos del mundo real.
- En cuanto a las físicas y colisiones, incluye un módulo para resolver el comportamiento de los cuerpos rígidos y articulados. También permite simular el comportamiento de vehículos en aspectos como la suspensión o la fricción de los neumáticos. Por otra parte, implementa varios modelos de destrucción dependiendo de la naturaleza del objeto.
- Contiene un módulo para gestionar los aspectos del modo multijugador, tanto a nivel de cliente como de servidor.
- Incluye herramientas para controlar el rendimiento de CPU y GPU en tiempo de ejecución, como la de perfilado, que indica qué módulos y funciones están implicadas.

Algunos de los títulos desarrollados con este motor son *Aporia: Beyond the Valley* (Investigate North, 2017) (Figura 24. *Fotograma de Aporia: Beyond the Valley* [Figura]. Recuperado de https://www.cryengine.com/showcase/view/aporia-beyond-the-valley), *Crysis 3* (Crytek, 2013) o *Evolve* (Turtle Rock Studios, 2015).

Figura 24. *Fotograma de Aporia: Beyond the Valley* [Figura]. Recuperado de https://www.cryengine.com/showcase/view/aporia-beyond-the-valley

Por otra parte, se encuentra Godot, que es un motor gratuito de código abierto para el desarrollo de juegos 2D y 3D. Consiste en un editor ligero de unos 30 MB que se puede ejecutar en Windows, macOS, Linux y BSD y con el que se pueden desarrollar juegos para multitud de plataformas, desde ordenadores de sobremesa, Android, iOS y consolas como PlayStation o Nintendo Switch, hasta para la web mediante HTML5 y WebAssembly. También permite la creación de juegos de Realidad Virtual y Aumentada (*Godot Engine - Features*, 2021).

Algunos aspectos destacables de este motor son:

- El renderizador basado en física, siguiendo el modelo Physically Based Rendering, que simula el comportamiento de la luz en el mundo real en las superficies.
- En la parte de física incluye diferentes tipos de cuerpos: rígidos, articulados, estáticos, blandos o vehículos. Para la detección de colisiones usa formas simples como esferas o cubos y permite generar *colliders* para cualquier modelo 3D.
- Incluye sonido mono, estéreo y envolvente 5.1 y 7.1, en el apartado de audio. También permite aplicar el efecto Doppler, que, junto a la característica anterior, ayuda a conseguir el efecto de audio 3D.
- En cuanto a animación, incorpora cinemática directa e inversa, soporte para animar propiedades mediante interpolación y curvas de Bézier.
- Además de los dispositivos habituales de entrada, incluye soporte para los eventos de tabletas con posibilidad de controlar la presión.

- En el ámbito de la inteligencia artificial, incluye el algoritmo A* y las mallas de navegación con detección de obstáculos para los NPCs.
- En la parte de red, incluye una API de alto nivel basada en HTTP y SSL, aunque también soporta conexiones TCP y UDP.

The Garden Path (carrotcake.studio, 2021) (Figura 25. *Fotograma del juego The Garden Path* [Figura]. Recuperado de https://store.steampowered.com/app/1638500/The_Garden_Path/ .), *Resolutiion* (Monolith of Minds, 2020) o *Gravity Ace* (John Watson, 2020) son algunos ejemplos de los títulos desarrollados con Godot.

Figura 25. *Fotograma del juego The Garden Path* [Figura]. Recuperado de https://store.steampowered.com/app/1638500/The_Garden_Path/ .

Otro motor bastante conocido y usado es Unreal Engine, de la compañía Epic Games. Permite el desarrollo de juegos en cualquier plataforma disponible. No sólo está indicado para el ámbito de los videojuegos, sino que se usa en campos como la arquitectura, el cine o la automoción. Es de uso gratuito para el ámbito de la educación y cuando los beneficios son inferiores al millón de dólares (*Unreal Engine - Features*, 2021).

Unreal Engine se caracteriza por:

- En el apartado gráfico, permite usar la técnica de ray tracing en tiempo real para el cálculo de sombras, reflexiones, traslucidez e iluminación global. Además, incluye modelos de sombreado avanzados para conseguir efecto de transparencia, de piel o de pelo, entre otros, que

permiten lograr resultados más realistas en objetos y superficies con características especiales.

- Para las tareas de animación, tenemos herramientas como cinemática directa e inversa, animación basada en física, espacios de mezcla para controlar múltiples variables al mismo tiempo o controlar las transformaciones del esqueleto directamente.
- En cuanto a la física, contiene herramientas para simular el comportamiento de ropa y tejidos, la destrucción de objetos e, incluso, el movimiento del pelo. Además, permite la creación de efectos visuales, como el humo, el fuego, el polvo o el agua, mediante partículas.
- En el ámbito de la inteligencia artificial incluye varias características para dotar de sentido el comportamiento de los NPCs. Combina árboles de decisiones con información del entorno y con sistemas de percepción de la vista, oído o el daño. Por otra parte, también incluye las mallas de navegación dinámicas para el cálculo óptimo de la trayectoria.
- En el apartado de audio, provee numerosas herramientas para la espacialización del audio, el procesamiento y síntesis de las señales digitales para aplicar efectos, o la gestión de la concurrencia de sonidos.
- Tiene un framework integrado basado en la arquitectura cliente/servidor para facilitar la creación de juegos multijugador de forma escalable.

Gracias a todas estas características y al potencial de Unreal Engine, ha sido la base de numerosos títulos muy conocidos como *Valorant* (Riot Games, 2020), *PlayerUnknown's Battlegrounds* (PUBG Corporation, 2016), *Fortnite* (Epic Games, 2017) o *Final Fantasy VII Remake* (Square Enix, 2020) (Figura 26. *Fotograma de Final Fantasy VII Remake* [Figura]. Recuperado de https://square-enix-games.com/en_EU/news/final-fantasy-vii-remake-screenshots-wall-market-honeybee-inn.).

Figura 26. *Fotograma de Final Fantasy VII Remake* [Figura]. Recuperado de https://square-enix-games.com/en_EU/news/final-fantasy-vii-remake-screenshots-wall-market-honeybee-inn.

2.6.Tendencias futuras.

En cuanto a las tendencias actuales y futuras, podemos discernir entre el ámbito del desarrollo de juegos y los propios juegos en sí, ya que ambas van a ir ligadas.

Está claro que uno de los principales objetivos en cuanto al desarrollo de videojuegos siempre ha sido conseguir el mayor realismo posible en todos los aspectos de los mismos. Desde la parte gráfica, la iluminación, el movimiento de los personajes, hasta el comportamiento de NPCs, son factores que determinan la calidad de un título. Este planteamiento de evolución constante seguirá desarrollándose mediante nuevas técnicas para continuar mejorando todas estas características que hacen que el jugador se sienta inmerso en el mundo virtual.

Una de las tendencias actuales es el cloud gaming, en el que no es necesario tener un equipo con altas prestaciones para poder disfrutar de la experiencia de jugar de forma fluida con alta calidad en gráficos y sonidos. Plataformas como Stadia de Google o la española NWare de Cloudware ofrecen la posibilidad de jugar en streaming desde un ordenador o teléfono simplemente teniendo acceso a internet. La tendencia de las suscripciones a los servicios de música, series y películas también se traslada a este ámbito y, algunos autores como el CEO de Ubisoft apuntan que los juegos en streaming van a reemplazar a las plataformas convencionales (Cal, 2018). Gracias a la progresiva incorporación de nuevas tecnologías de redes móviles como el 5G, se potenciará este enfoque de juego.

Otro aspecto que está cobrando relevancia y que se prevé que continúe creciendo es el desarrollo de juegos para realidad virtual o realidad aumentada. La búsqueda continua de la sensación de inmersión en el mundo virtual hace que se usen estas tecnologías para lograr este objetivo. Además, la mejora en el hardware de los dispositivos móviles, permite que cualquier persona pueda experimentar con la realidad aumentada sin necesidad de adquirir otro dispositivo. Estos factores explican el éxito de juegos como Pokemon Go (Nintendo, 2016) o Jurassic World Alive (Ludia, 2018).

En lo que respecta a los motores de videojuegos, el *community manager* de Epic Games, Dazhou, afirma que los motores no sólo se van a utilizar para la creación de videojuegos, sino que su uso se va a expandir a otras áreas como la arquitectura o la industria del automóvil (COCOS, 2020). También dice que los motores van a ir adaptándose a las necesidades, tanto de los creadores como de los jugadores, y a las nuevas tecnologías, como la realidad virtual y aumentada, el *deep learning* o las nuevas formas de interacción.

Una característica que están incluyendo bastantes motores es la programación visual, que hace que sea más sencillo describir las reglas y mecánicas que rigen el

comportamiento del juego. Esto permite que personas que no son programadoras programen, debido a la facilidad y usabilidad de manejar una interfaz basada en nodos conectados, sin tener que conocer lenguajes de programación. Sin embargo, con la programación visual no se puede obtener todo el potencial que se consigue con el scripting tradicional, a pesar de las múltiples ventajas que ofrece. Algunos ejemplos de motores que incluyen esta característica son Unreal Engine con los Blueprints (Figura 27. *Blueprint en Unreal Engine* [Figura].), Unity con Bolt, CryEngine con Schematyc o Godot.

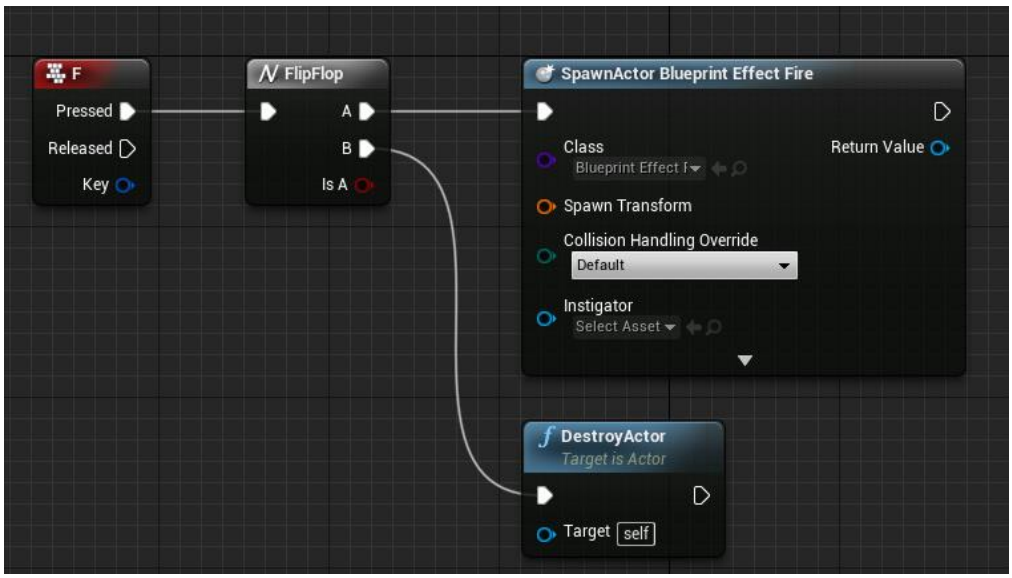

Figura 27. *Blueprint en Unreal Engine* [Figura]. Recuperado de https://docs.unrealengine.com/4.26/en-US/ProgrammingAndScripting/SpawnAndDestroyActors/

Personalmente, considero que los avances que se produzcan en cuanto a las tecnologías relacionadas con el desarrollo de videojuegos van a estar orientadas por el objetivo de conseguir cada vez más realismo, desde las mejoras en el apartado de gráficos, física y sonido, hasta la creación de nuevas formas de interacción que hagan que el usuario se introduzca en el mundo virtual de forma cada vez más intuitiva y transparente.

2.7. Conclusiones.

Estos motores hacen posible que estudios pequeños con pocos recursos o, incluso, personas a título individual, puedan desarrollar juegos sin problemas y, como consecuencia, que la industria del videojuego tenga tanta diversidad en cuanto a géneros y estilos, pues no solo abarca títulos comerciales de grandes compañías, sino que también se ve influenciada por ideas innovadoras de pequeños estudios.

Hay varias posibilidades en cuanto a la creación de videojuegos, pudiendo escribir todo el código desde cero, usando librerías y middleware de terceros que simplifique algunas tareas o utilizando motores en los que simplemente haya que programar la lógica del juego. Las grandes compañías suelen desarrollar sus propios motores diseñados para satisfacer las necesidades específicas de sus títulos, sin embargo, para equipos más pequeños con recursos económicos y temporales limitados, la mejor opción es el uso de un motor como los mencionados anteriormente que solo requieren de la programación de mecánicas y reglas específicas del juego, de esta forma, los esfuerzos van dedicados a la creación de la parte más distintiva del mismo.

En cuanto a la industria del videojuego en sí, cada vez hay más jugadores adultos debido a que las primeras generaciones que comenzaron a jugar en las primeras consolas, tanto portátiles como de sobremesa, han ido creciendo. El mundo de los videojuegos ha dejado de ser cosa de niños exclusivamente, para abrirse a un público maduro que lo considera como una forma de ocio más. Aunque el público que tiene es tan diverso en cuanto a edades, los adolescentes y jóvenes siguen siendo, sin duda, los principales consumidores de juegos, por lo que no es de extrañar que la mayor parte de los títulos estén orientados a esa franja de edad.

Por otra parte, los *serious games* son herramientas con bastante potencial en ámbitos como la educación, pues suponen una forma más cercana al alumnado de aprender, de manera divertida, con posibilidades que no permiten otras herramientas, por lo que no es extraño que cada vez más se vayan incluyendo en las aulas o centros de formación.

A pesar de ser un mercado en crecimiento, en España existen pocas oportunidades en el ámbito laboral (como apunta el Libro Blanco del Desarrollo Español de Videojuegos), tan solo hay en torno a 7000 personas trabajando en este campo en nuestro país, mientras que muchas otras personas, sin posibilidad de acceder al mercado laboral español, optan por trabajar en el extranjero donde hay un mayor número de oportunidades y las condiciones laborales (sueldos y dedicación), son mejores. En este sentido, el papel de la administración es clave en cuanto al fomento del emprendimiento, el asesoramiento y la dotación de ayudas para quienes comienzan su vida laboral. El principal reto al que se enfrenta la industria en nuestro

país es la consolidación de este mercado mediante la profesionalización del sector y la evitación de la fuga de cerebros fuera de España para trabajar como creadores de videojuegos (Desarrollo Español de Videojuegos (DEV), 2021).

3. Proyección didáctica

Como se ha mencionado en el apartado anterior *Fundamentación epistemológica*., el mercado de los videojuegos está en constante evolución debido al impacto que estos tienen en la sociedad, tanto los diseñados únicamente para fines lúdicos como aquellos otros diseñados para aspectos más formales.

A pesar de que nuestro país es un gran consumidor de la industria del videojuego, no existe una industria del desarrollo de videojuegos consolidada, como cabría de esperar debido a la gran demanda del mismo. Por esta razón, se pretende motivar a los estudiantes a emprender en este ámbito.

En los siguientes epígrafes se va a detallar la programación de la unidad didáctica *Introducción al desarrollo de videojuegos* correspondiente al módulo de segundo curso *Programación multimedia y dispositivos móviles* del Ciclo Formativo de Grado Superior de Desarrollo de Aplicaciones Multiplataforma. En dicha programación, se incluyen diversos aspectos relacionados con la normativa, organización del aula, agrupamientos, recursos, metodologías, mecanismos e instrumentos de evaluación y recuperación y atención a la diversidad, contemplados todos ellos en el currículo de la materia.

3.1. Contextualización del centro

El IES Miguel Sánchez López (Figura 28. *Fachada del IES Miguel Sánchez López*.) se encuentra en la localidad de Torredelcampo, situada a escasos kilómetros de Jaén capital y bien comunicada con otras provincias como Córdoba y Granada mediante la autovía A-316. Este municipio cuenta con una población de 14000 habitantes y una extensión de 182. Este centro se ubica en la Avenida de la Constitución (Figura 29. *Localización del IES Miguel Sánchez López*. Extraído de https://www.google.com/maps/place/I.E.S.+Miguel+S%C3%A1nchez+L%C3%B3pez/@37.7681599,-3.9040464,17z/data=!3m1!4b1!4m5!3m4!1s0xd6dd0249fe140a5:0xecaaf92e8b7ba4b0!8m2!3d37.7682492!4d-3.9019104), una localización cercana al centro de la ciudad y a la entrada desde la autovía, lo cual hace que su situación sea estratégica. Además de este, se encuentra el IES Torre Olvidada, en el que sólo se imparte la etapa de ESO, por lo que los alumnos de esta localidad que desean cursar Bachillerato o ciclo formativo son derivados al IES Miguel Sánchez López. La consecuencia de todo esto es elevado el número de alumnos de 1º y 2º de Bachillerato y la alta demanda que posee el ciclo formativo. En el centro solo se imparte el Ciclo Formativo de Grado Medio de Sistemas Microinformáticos y Redes, a los que acceden alumnos tanto de la localidad como de municipios cercanos como Torredonjimeno, Jaén y Jamilena. La programación de la

unidad didáctica se orienta al caso de que en un futuro se implante el Ciclo Formativo de Grado Superior de Desarrollo de Aplicaciones Multiplataforma.

Figura 28. *Fachada del IES Miguel Sánchez López*.

Figura 29. *Localización del IES Miguel Sánchez López*. Extraído de https://www.google.com/maps/place/I.E.S.+Miguel+S%C3%A1nchez+L%C3%B3pez/@37.7681599,-3.9040464,17z/data=!3m1!4b1!4m5!3m4!1s0xd6dd0249fe140a5:0xecaaf92e8b7ba4b0!8m2!3d37.7682492!4d-3.9019104

La situación económica de Torredelcampo es bastante próspera y dinámica en relación a los municipios aledaños. Dentro de las actividades económicas destacan la agricultura (cultivo del olivar), la industria y el sector terciario. En cuanto a la situación

sociocultural de las familias, podemos encontrar, desde aquellas con estudios básicos, hasta aquellas con estudios medios y superiores (en torno al 40%).

Según datos de cursos anteriores y encuestas realizadas, la motivación para continuar los estudios en Bachillerato y ciclos formativos, es muy alta, alrededor del 80%.

Al centro pertenecen sesenta profesores, de los cuales seis son del Departamento de Informática. Tres de ellos pertenecen al cuerpo de Profesores de Enseñanza Secundaria y tres son Profesores Técnicos de Formación Profesional.

Características del supuesto alumnado

El número de alumnos pertenecientes al CFGS de Desarrollo de Aplicaciones Multiplataforma, se eleva a veintinueve personas, distribuidas entre primero y segundo.

Centrándonos en el curso al que pertenece el módulo de *Programación multimedia y dispositivos móviles*, los catorce alumnos de segundo no disponen de módulos pendientes, todos han promocionado desde primero con buenas calificaciones, siendo la media de la clase un 7,2. Dentro de este grupo se encuentra un alumno con hipoacusia leve en el oído izquierdo, NEAE.

3.2. Marco legislativo

La normativa en la que se basa la programación de dicha unidad didáctica, tanto a nivel estatal como a nivel autonómico, es:

1. Real Decreto 450/2010, de 16 de abril, por el que se establece el título de Técnico Superior en Desarrollo de Aplicaciones Multiplataforma y se fijan sus enseñanzas mínimas.
2. Orden de 16 de junio de 2011, por la que se desarrolla el currículo correspondiente al título de Técnico Superior en Desarrollo de Aplicaciones Multiplataforma.

3.3. Información general

De acuerdo con la normativa mencionada en el apartado anterior, se establece lo siguiente respecto al desarrollo de la programación de la unidad didáctica:

Título: Introducción al desarrollo de videojuegos

Etapa: Ciclo Formativo Grado Superior Desarrollo de Aplicaciones Multiplataforma.

Curso: Segundo

Módulo al que pertenece: Programación multimedia y dispositivos móviles. Código: 0489. (4 horas/semana, 84 horas totales)

Trimestre: Primero

Temporalización número de horas: 9 h (del 21 de septiembre al 5 de octubre de 2021)

Número de sesiones: 4 sesiones de 2 horas y la mitad de la quinta sesión

Competencia general:

El artículo 4 del Real Decreto 450/2010 establece que:

"La competencia general de este título consiste en desarrollar, implantar, documentar y mantener aplicaciones informáticas multiplataforma, utilizando tecnologías y entornos de desarrollo específicos, garantizando el acceso a los datos de forma segura y cumpliendo los criterios de «usabilidad» y calidad exigidas en los estándares establecidos."

Orientaciones pedagógicas:

El módulo profesional al que pertenece la unidad didáctica contiene la formación necesaria para desempeñar la función de desarrollo de aplicaciones multimedia, juegos y aplicaciones adaptadas para su explotación en dispositivos móviles y de entre los aspectos que incluye, destaca el apartado c). Estos aspectos son:

a) La creación de aplicaciones que incluyen contenidos multimedia basadas en la inclusión de librerías específicas en función de la tecnología utilizada.

b) La creación de aplicaciones para dispositivos móviles que garantizan la persistencia de los datos y establecen conexiones para permitir su intercambio.

c) El desarrollo de juegos 2D y 3D utilizando las funcionalidades que ofrecen los motores de juegos, así como su puesta a punto e implantación en dispositivos móviles.

Competencias profesionales, personales y sociales:

La totalidad de las competencias profesionales, personales y sociales del título son las que se relacionan a continuación:

a) Configurar y explotar sistemas informáticos, adaptando la configuración lógica del sistema según las necesidades de uso y los criterios establecidos.

b) Aplicar técnicas y procedimientos relacionados con la seguridad en sistemas, servicios y aplicaciones, cumpliendo el plan de seguridad.

c) Gestionar bases de datos, interpretando su diseño lógico y verificando integridad, consistencia, seguridad y accesibilidad de los datos.

d) Gestionar entornos de desarrollo adaptando su configuración en cada caso para permitir el desarrollo y despliegue de aplicaciones.

e) Desarrollar aplicaciones multiplataforma con acceso a bases de datos utilizando lenguajes, librerías y herramientas adecuados a las especificaciones.

f) Desarrollar aplicaciones implementando un sistema completo de formularios e informes que permitan gestionar de forma integral la información almacenada.

g) Integrar contenidos gráficos y componentes multimedia en aplicaciones multiplataforma, empleando herramientas específicas y cumpliendo los requerimientos establecidos.

h) Desarrollar interfaces gráficos de usuario interactivos y con la usabilidad adecuada, empleando componentes visuales estándar o implementando componentes visuales específicos.

i) Participar en el desarrollo de juegos y aplicaciones en el ámbito del entretenimiento y la educación empleando técnicas, motores y entornos de desarrollo específicos.

j) Desarrollar aplicaciones para teléfonos, PDA y otros dispositivos móviles empleando técnicas y entornos de desarrollo específicos.

k) Crear ayudas generales y sensibles al contexto, empleando herramientas específicas e integrándolas en sus correspondientes aplicaciones.

l) Crear tutoriales, manuales de usuario, de instalación, de configuración y de administración, empleando herramientas específicas.

m) Empaquetar aplicaciones para su distribución preparando paquetes auto instalables con asistentes incorporados.

n) Desarrollar aplicaciones multiproceso y multihilo empleando librerías y técnicas de programación específicas.

ñ) Desarrollar aplicaciones capaces de ofrecer servicios en red empleando mecanismos de comunicación.

o) Participar en la implantación de sistemas ERP-CRM evaluando la utilidad de cada uno de sus módulos.

p) Gestionar la información almacenada en sistemas ERP-CRM garantizando su integridad.

q) Desarrollar componentes personalizados para un sistema ERP-CRM atendiendo a los requerimientos.

r) Realizar planes de pruebas verificando el funcionamiento de los componentes software desarrollados, según las especificaciones.

s) Desplegar y distribuir aplicaciones en distintos ámbitos de implantación verificando su comportamiento y realizando las modificaciones necesarias.

t) Establecer vías eficaces de relación profesional y comunicación con sus superiores, compañeros y subordinados, respetando la autonomía y competencias de las distintas personas.

u) Liderar situaciones colectivas que se puedan producir, mediando en conflictos personales y laborales, contribuyendo al establecimiento de un ambiente de trabajo agradable, actuando en todo momento de forma respetuosa y tolerante.

v) Gestionar su carrera profesional, analizando las oportunidades de empleo, autoempleo y de aprendizaje.

w) Mantener el espíritu de innovación y actualización en el ámbito de su trabajo para adaptarse a los cambios tecnológicos y organizativos de su entorno profesional.

x) Crear y gestionar una pequeña empresa, realizando un estudio de viabilidad de productos, de planificación de la producción y de comercialización.

y) Participar de forma activa en la vida económica, social y cultural, con una actitud crítica y responsable.

Las competencias profesionales propias del módulo *Programación multimedia y dispositivos móviles* corresponden a las que aparecen en los apartados d), e), g), h), i), j), l), m), n), ñ), s), t), w), y, de ellas, se alcanzarían en la unidad didáctica las de los apartados d), i), w).

Líneas de actuación:

Las líneas de actuación en el proceso de enseñanza aprendizaje que permiten alcanzar los objetivos del módulo versarán sobre:

a) El análisis de las tecnologías disponibles para dispositivos móviles, sus características y funcionalidad.
b) La utilización de emuladores para evaluar el funcionamiento tanto de las aplicaciones para dispositivos móviles desarrolladas como de las modificaciones introducidas en aplicaciones existentes.

c) El desarrollo de aplicaciones para dispositivos móviles que garantizan la persistencia de los datos y permiten el establecimiento de conexiones con otros dispositivos y el intercambio de datos.

d) El desarrollo de aplicaciones que integran objetos multimedia.

e) El análisis de motores de juegos, sus características y funcionalidades.

f) El desarrollo de juegos 2D y 3D aplicando técnicas específicas y utilizando instrucciones gráficas para establecer efectos sobre objetos o imágenes.

Concretamente, en la unidad didáctica, se encuadran las líneas de los apartados e) y f).

Objetivos Generales:

A continuación, se muestran los objetivos generales del título de Técnico Superior en Desarrollo de Aplicaciones Multiplataforma, de los cuáles, d), e), f), g), h), i), j), l), m), n), r), s), w) se alcanzan en el módulo *Programación multimedia y dispositivos móviles*. De estos objetivos, la unidad *Introducción a la programación de videojuegos* contribuye a la consecución de los que se muestran en negrita.

a) Ajustar la configuración lógica del sistema analizando las necesidades y criterios establecidos para configurar y explotar sistemas informáticos.

b) Identificar las necesidades de seguridad analizando vulnerabilidades y verificando el plan preestablecido para aplicar técnicas y procedimientos relacionados con la seguridad en el sistema.

c) Interpretar el diseño lógico de bases de datos, analizando y cumpliendo las especificaciones relativas a su aplicación, para gestionar bases de datos.

d) Instalar y configurar módulos y complementos, evaluando su funcionalidad, para gestionar entornos de desarrollo.

e) Seleccionar y emplear lenguajes, herramientas y librerías, interpretando las especificaciones para desarrollar aplicaciones multiplataforma con acceso a bases de datos.

f) Gestionar la información almacenada, planificando e implementando sistemas de formularios e informes para desarrollar aplicaciones de gestión.

g) Seleccionar y utilizar herramientas específicas, lenguajes y librerías, evaluando sus posibilidades y siguiendo un manual de estilo, para manipular e integrar en aplicaciones multiplataforma contenidos gráficos y componentes multimedia.

h) Emplear herramientas de desarrollo, lenguajes y componentes visuales, siguiendo las especificaciones y verificando interactividad y usabilidad, para desarrollar interfaces gráficos de usuario en aplicaciones multiplataforma.

i) Seleccionar y emplear técnicas, motores y entornos de desarrollo, evaluando sus posibilidades, para participar en el desarrollo de juegos y aplicaciones en el ámbito del entretenimiento.

j) Seleccionar y emplear técnicas, lenguajes y entornos de desarrollo, evaluando sus posibilidades, para desarrollar aplicaciones en teléfonos, PDA y otros dispositivos móviles.

k) Valorar y emplear herramientas específicas, atendiendo a la estructura de los contenidos, para crear ayudas generales y sensibles al contexto.

l) Valorar y emplear herramientas específicas, atendiendo a la estructura de los contenidos, para crear tutoriales, manuales de usuario y otros documentos asociados a una aplicación.

m) Seleccionar y emplear técnicas y herramientas, evaluando la utilidad de los asistentes de instalación generados, para empaquetar aplicaciones.

n) Analizar y aplicar técnicas y librerías específicas, simulando diferentes escenarios, para desarrollar aplicaciones capaces de ofrecer servicios en red.

ñ) Analizar y aplicar técnicas y librerías de programación, evaluando su funcionalidad para desarrollar aplicaciones multiproceso y multihilo.

o) Reconocer la estructura de los sistemas ERP-CRM, identificando la utilidad de cada uno de sus módulos, para participar en su implantación.

p) Realizar consultas, analizando y evaluando su alcance, para gestionar la información almacenada en sistemas ERP-CRM.

q) Seleccionar y emplear lenguajes y herramientas, atendiendo a los requerimientos, para desarrollar componentes personalizados en sistemas ERP-CRM.

r) Verificar los componentes software desarrollados, analizando las especificaciones, para completar un plan de pruebas.

s) Establecer procedimientos, verificando su funcionalidad, para desplegar y distribuir aplicaciones.

t) Describir los roles de cada uno de los componentes del grupo de trabajo, identificando en cada caso la responsabilidad asociada, para establecer las relaciones profesionales más convenientes.

u) Identificar formas de intervención ante conflictos de tipo personal y laboral, teniendo en cuenta las decisiones más convenientes, para garantizar un entorno de trabajo satisfactorio.

v) Identificar y valorar las oportunidades de promoción profesional y de aprendizaje, analizando el contexto del sector, para elegir el itinerario laboral y formativo más conveniente.

w) Identificar los cambios tecnológicos, organizativos, económicos y laborales en su actividad, analizando sus implicaciones en el ámbito de trabajo, para mantener el espíritu de innovación.

x) Reconocer las oportunidades de negocio, identificando y analizando demandas del mercado para crear y gestionar una pequeña empresa.

y) Reconocer sus derechos y deberes como agente activo en la sociedad, analizando el marco legal que regula las condiciones sociales y laborales para participar como ciudadano democrático.

Resultados de aprendizaje y criterios de evaluación:

En la unidad Introducción a la programación de videojuegos se pretende lograr el resultado de aprendizaje 4. Selecciona y prueba motores de juegos analizando la arquitectura de juegos 2D y 3D, concretamente, a través de los criterios de evaluación resaltados en negrita:

a) Se han analizado los componentes de un motor de juegos.
b) Se han identificado los elementos que componen la arquitectura de un juego 2D y 3D.
c) Se han analizado entornos de desarrollo de juegos.
d) Se han analizado diferentes motores de juegos, sus características y funcionalidades.
e) Se han identificado los bloques funcionales de un juego existente.
f) Se han definido y ejecutado procesos de render.
g) Se ha reconocido la representación lógica y espacial de una escena gráfica sobre un juego existente.

Los criterios de evaluación f y g se aplicarán en las unidades posteriores.

Contenidos Conceptuales:

Los contenidos conceptuales básicos asociados al resultado de aprendizaje 4 y, concretamente a los criterios de evaluación a), b), c), d), y e) corresponden al apartado *Análisis de motores de juegos*, y son los siguientes:

- Animación 2D y 3D.
- Arquitectura del juego. Componentes.
- Motores de juegos. Tipos y utilización.

- Áreas de especialización, librerías utilizadas y lenguajes de programación.
- Componentes de un motor de juegos.
- Librerías que proporcionan las funciones básicas de un Motor 2D/3D.

Temas transversales:

Como tema transversal se va a tratar el emprendimiento en el mundo de los videojuegos. Para ello se realizará un aprendizaje por descubrimiento basado en cómo comercializar y desplegar un videojuego desde diferentes puntos de vista (publicidad, plataformas de distribución, legislación…).

En la siguiente tabla (*Tabla 1. Resumen información general.*) se resume la información anterior y la normativa en la que se basa esta unidad didáctica.

Tabla 1. *Resumen información general.*

<table>
<tr><th colspan="8">UD 1. Introducción al desarrollo de videojuegos</th></tr>
<tr><th>Resultado de aprendizaje</th><th>Objetivos Generales Ciclo Formativo</th><th>Líneas de actuación</th><th>Competencias profesionales, personales y sociales</th><th>Curso</th><th>Trimestre y ponderación</th><th>Ponderación global</th><th>Número de horas y sesiones</th></tr>
<tr><td>4</td><td>i, w</td><td>e, f</td><td>d, i, w</td><td>2º</td><td>1º (30% global)</td><td>7.5% (25% trimestre)</td><td>9 horas (5,5 sesiones)</td></tr>
<tr><th colspan="8">Normativa</th></tr>
<tr><th>Estatal</th><td colspan="3">Real Decreto 450/2010, de 16 de abril, por el que se establece el título de Técnico Superior en Desarrollo de Aplicaciones Multiplataforma y se fijan sus enseñanzas mínimas.</td><th>Autonómica</th><td colspan="3">Orden de 16 de junio de 2011, por la que se desarrolla el currículo correspondiente al título de Técnico Superior en Desarrollo de Aplicaciones Multiplataforma.</td></tr>
<tr><th>Criterios de evaluación</th><td colspan="7">a, b, c, d, e</td></tr>
<tr><th colspan="8">Objetivos didácticos</th></tr>
<tr><td colspan="8">• Analizar motores de juegos. Conocer diferentes motores de juegos, los tipos de motores y su utilización.</td></tr>
</table>

<table>
<tr><td colspan="3">

- Identificar los componentes de un motor de juegos.
- Conocer las librerías que proporcionan las funciones básicas de un Motor 2D/3D.
- Diferenciar las áreas de especialización, librerías utilizadas y lenguajes de programación.
- Conocer la arquitectura del juego y sus componentes. Arquitectura sistema-entidad-componente.
- Identificar y aplicar diferentes técnicas de animaciones 2D y 3D.

</td></tr>
<tr><td colspan="3">Contenidos conceptuales</td></tr>
<tr><td colspan="3">

- Animación 2D y 3D.
- Arquitectura del juego. Componentes.
- Motores de juegos. Tipos y utilización.
- Áreas de especialización, librerías utilizadas y lenguajes de programación.
- Componentes de un motor de juegos
- Librerías que proporcionan las funciones básicas de un Motor 2D/3D.

</td></tr>
<tr><td>Tema transversal</td><td>Emprendimiento en el mundo de los videojuegos</td><td>Actividad de investigación. ¿Qué hacer para vender un videojuego? (Plataformas para la distribución, marketing, leyes…)</td></tr>
</table>

3.4. Metodología

Los aspectos metodológicos para impartir la unidad didáctica versan sobre los siguientes apartados:

3.4.1. Organización del aula

Las clases se van a impartir en el laboratorio de Informática destinado a la docencia del segundo curso del CFGS de Desarrollo de Aplicaciones Multiplataforma. Las mesas en las que se ubican los ordenadores estarán en los laterales de la clase, y, en el centro, habrá mesas grandes en las que se podrán sentar el grupo de alumnos para seguir la exposición teórica del profesor y tomar notas. En cuanto a los equipos colocados en los laterales, cada alumno dispondrá de un ordenador en el que poder trabajar de forma individual.

3.4.2. Agrupamientos

Se van a realizar tres tipos de agrupamientos:

- Trabajo individual: Se usa para los momentos de realización de los ejercicios prácticos que se van a desarrollar a lo largo de la unidad, además de en la prueba escrita.
- Trabajo en parejas: Para la actividad de ampliación, los alumnos tendrán que trabajar en esta modalidad, con el fin de desarrollar habilidades como el trabajo en equipo, la cooperación, comunicación, etc. En casos excepcionales, podrán ser grupo de tres.
- Grupo-clase: Es el que conforma la totalidad de los alumnos de la clase y el que se va a dar en los momentos de explicación de los conceptos por parte de la profesora.

3.4.3. Materiales y recursos didácticos.

Para el desarrollo de las actividades de enseñanza-aprendizaje se dispone de:

- Hardware: un ordenador para cada alumno, con 16 GB de memoria RAM, GPU NVIDIA RTX 2080 Ti, procesador Intel i7 de novena generación y disco duro SSD Samsung Evo 1TB. Características adecuadas para el desarrollo de aplicaciones multimedia de forma fluida.

  Además de los equipos en los que se va a trabajar, se dispone de *gamepads* o mandos de juegos.

- Sistema operativo Windows 10.

- Software: Unity, Unreal Engine (motores de juegos), Visual Studio (entorno de desarrollo).
- Moodle como entorno virtual de aprendizaje (VLE) para la realización de actividades online, descargar guiones, foros, entrega las actividades…
- Pizarra digital.
- Cámara para la modalidad semipresencial.
- Diapositivas para guiar la explicación.
- Videotutoriales.
- Libros de consulta y apuntes de clase, disponibles en formato electrónico y físico.

3.4.4. Metodología y actividades de enseñanza-aprendizaje.

Para conseguir que el alumno interiorice los conceptos clave, se van a estructurar las sesiones de la siguiente forma:

En la primera parte de la clase, se va a hacer un pequeño repaso de los contenidos vistos en la última sesión y se van a relacionar con un adelanto de los contenidos que se van a trabajar en la sesión actual, como forma de presentación de estos. En el caso de la primera sesión, se introducirá y contextualizará la unidad.

Seguidamente, se va a proceder a la explicación de los contenidos. Durante esta explicación, se animará a los alumnos a exponer sus ideas y dudas para generar dinámicas de debate. En esta parte, la profesora irá realizando unos ejemplos prácticos cuando considere que la relevancia del contenido lo requiere.

Una vez se realiza la explicación, los alumnos tendrán que trabajar de forma práctica los conceptos que se han visto. En esta parte también se podrán consultar las dudas al profesor.

Aunque las actividades están orientadas para su realización en clase, excepcionalmente, podrá realizarse una entrega en otro momento que determine la profesora.

Además de lo anterior, se va a proponer a los alumnos la realización de una actividad de ampliación en la que se trata el tema transversal del emprendimiento.

Las actividades de enseñanza-aprendizaje que se van a realizar para desarrollar esta metodología son:

- Actividad de repaso-presentación, en la que se van a resaltar los conceptos más relevantes de la sesión anterior (excepto en la primera sesión) y se van a relacionar con los contenidos que se van a trabajar en la sesión actual. Esta actividad permite incidir en los aspectos que no hayan quedado tan claros

teniendo en cuenta los resultados de las actividades de consolidación de la sesión anterior.

- Explicación de los contenidos, que van a permitir sentar las bases del conocimiento de los alumnos. En esta actividad, los alumnos pueden intervenir con dudas e ideas relacionadas con los contenidos. También tienen una componente práctica, ya que el profesor entrelaza los conceptos con la aplicación de los mismos mediante ejemplos.
- Actividades de consolidación individuales, en las que se van a trabajar unos ejercicios prácticos relacionados con los contenidos vistos en la sesión. Estas actividades deberán entregarse al final de cada sesión.
- Prueba teórico-práctica escrita, en la que se evaluará la asimilación de conceptos teóricos y puesta en práctica de los conceptos trabajados en la unidad didáctica. La prueba constará de diez preguntas tipo test, dos de respuesta corta y un supuesto práctico de identificación de entidades y componentes que el alumno deberá resolver.
- Trabajo de investigación en casa (actividad de ampliación), en el que las parejas de alumnos deberán buscar información acerca del proceso de comercialización de un videojuego. Los alumnos deberán coordinarse y comunicarse para realizar cada una de las partes. La fecha límite de entrega será el día del examen.

3.4.5. Temporalización.

La siguiente tabla (Tabla 2. *Temporalización de las sesiones*) muestra la organización de las actividades de enseñanza-aprendizaje en sesiones.

Tabla 2. *Temporalización de las sesiones*

Metodología				
Sesión	**Actividad**	**Tiempo**	**Agrupamiento**	**Recursos**
1	**Actividad de repaso-presentación** Introducción de la unidad didáctica. Contextualización del tema. Relevancia de los videojuegos.	10 min	Grupo-clase	Diapositivas Pizarra digital
1	**Explicación de los contenidos** Conceptos básicos de los videojuegos: • Definición videojuego y motor de videojuegos • Características videojuegos: mecánicas, reglas, géneros, modos de juego Realización de ejemplos • Arquitectura de un videojuego: Sistema-Entidad-Componente Realización de ejemplos • Ciclo de vida de un juego Introducción a Unity y Unreal Engine: partes del editor Presentación de la actividad de investigación	50 min	Grupo-clase	Diapositivas Pizarra digital Apuntes de clase Libro de texto

1	**Actividades de consolidación** • Dado el título de un juego, realiza una descripción del mismo teniendo en cuenta las mecánicas, reglas, géneros, modos de juego • Dado un enunciado con la lógica de un juego, realizar un análisis de la arquitectura y determinar las entidades, componentes y métodos involucrados	60 min	Individual	Ordenadores Moodle Apuntes de clase Libro de texto
2	**Actividad de repaso-presentación**	10 min	Grupo-clase	Diapositivas Pizarra digital
2	**Explicación de los contenidos** Componentes del motor I: • Arquitectura en capas y subsistemas del motor • Motor de renderizado: fundamentos, subsistemas, características de Vulkan y Direct3D Opciones en Unreal Engine y Unity para seleccionar una API u otra • Motor de física y colisiones: fundamentos, concepto de *rigidbody*, librería PhysX Gestión de físicas y colisiones en Unity y Unreal	45 min	Grupo-clase	Diapositivas Pizarra digital Apuntes de clase Libro de texto
2	**Actividades de consolidación** Guion de actividades para realizar con Unity (mediante scripting) y Unreal (mediante Blueprints): gestionar colisiones y fuerzas entre los objetos de la escena. Documentar el proceso y las diferencias observadas entre ambos motores en cuanto a	65 min	Individual	Ordenadores Moodle Apuntes de clase

	la realización.			Libro de texto
3	**Actividad de repaso-presentación**	10 min	Grupo-clase	Diapositivas Pizarra digital
3	**Explicación de los contenidos** Componentes del motor II: • Sistema de animación: *rigging*, cinemática inversa, máquina de estados, animación 2D mediante fotogramas clave (*keyframes*) • Audio: espacialización, efectos de sonido, librería FMOD	40 min	Grupo-clase	Diapositivas Pizarra digital Apuntes de clase Libro de texto
3	**Actividades de consolidación** Guion de actividades para realizar con Unity (mediante scripting) y Unreal (mediante Blueprints): realizar animación de humanoides mediante poses y cinemática inversa. Realizar una animación de sprites en 2D Documentar el proceso y las diferencias observadas entre ambos motores en cuanto a la realización.	70 min	Individual	Ordenadores Moodle Apuntes de clase Libro de texto
4	**Actividad de repaso-presentación**	10 min	Grupo-clase	Diapositivas Pizarra digital
4	Componentes del motor III: • Multijugador, *networking*: conceptos básicos (modelo cliente-servidor, replicación), SDK Photon.	45 min	Grupo-clase	Diapositivas Pizarra digital Apuntes de clase

	• Interfaces de usuario, dispositivos: gestión de eventos de teclado, ratón y *gamepad*			Libro de texto *Gamepad*
4	**Actividades de consolidación** Guion de actividades para realizar con Unity (mediante scripting): • Modificar los controles de usuario incluyendo los específicos del *gamepad.* • Realizar una prueba multijugador local haciendo uso del asset Photon 2 • Documentar el proceso.	65 min	Individual	Ordenadores Moodle Apuntes de clase Libro de texto *Gamepad*
5	**Prueba teórico-práctica escrita** Test + respuesta corta + casos prácticos	60 min (15+15+30)	Individual	Examen escrito

3.5. Evaluación

La evaluación de la unidad didáctica se llevará a cabo atendiendo a los siguientes apartados:

3.5.1. Criterios de evaluación

A continuación (Tabla 3. *Resultados de aprendizaje, criterios de calificación e instrumentos.*), se muestran los resultados de aprendizaje, criterios de evaluación e instrumentos básicos asociados a los contenidos de la UD.

Tabla 3. *Resultados de aprendizaje, criterios de calificación e instrumentos.*

Resultados de aprendizaje	Criterios de evaluación	Instrumentos
4. Selecciona y prueba motores de juegos analizando la arquitectura de juegos 2D y 3D	a) Se han analizado los componentes de un motor de juegos.	- Rúbrica actividades prácticas individuales - Examen escrito - Cuaderno del profesor - Rúbrica de observación en el aula
	b) Se han identificado los elementos que componen la arquitectura de un juego 2D y 3D.	- Rúbrica actividades prácticas individuales - Examen escrito - Cuaderno del profesor - Rúbrica de observación en el aula
	c) Se han analizado entornos de desarrollo de juegos.	- Rúbrica actividades prácticas individuales - Cuaderno del profesor - Rúbrica de observación en el aula
	d) Se han analizado diferentes motores de juegos, sus características y funcionalidades.	- Rúbrica actividades prácticas individuales - Cuaderno del profesor - Rúbrica de observación en el aula

	e) Se han identificado los bloques funcionales de un juego existente.	- Rúbrica actividades prácticas individuales - Examen escrito - Cuaderno del profesor - Rúbrica de observación en el aula
Tema transversal		- Rúbrica trabajo de investigación

3.5.2. Técnicas e instrumentos de evaluación.

Las técnicas de evaluación presentes en esta UD son:

● **Observación:**

Se prestará atención, sobre todo, a las actuaciones y comportamientos del alumnado durante el desarrollo de las actividades de enseñanza aprendizaje, así como la actitud, la participación en clase, las intervenciones en clase, la forma en la que interactúa con los compañeros y el material del aula. Con el objetivo de fomentar el respeto hacia los demás y de cara al futuro laboral, se incidirá en estos dos últimos aspectos.

Por supuesto, mediante la observación se controla la ejecución de los ejercicios prácticos de forma individual.

● **Actividades prácticas individuales:**

Las actividades relacionadas con el contenido visto en cada sesión servirán para que el alumno interiorice e incorpore nuevos conocimientos de forma práctica y gradual. Las relaciones de ejercicios se activarán en la plataforma Moodle justo antes de su ejecución para asegurar que se realizan dentro del horario de la clase. En caso de que un alumno, justificadamente, no asista a clase, lo podrá realizar simultáneamente de forma remota o, con posterioridad, acordando la fecha con la profesora.

Estas actividades se realizarán usando los motores de Unity y Unreal Engine para que los alumnos puedan comparar las características de ambos. Al final, también deberán plasmar el proceso seguido y las principales diferencias encontradas.

● **Trabajo de investigación (tema transversal):**

El alumnado buscará información acerca del proceso, normativa y otras cuestiones relativas a la comercialización de videojuegos para alentar el espíritu emprendedor y dotarle de herramientas, con proyección para su futuro laboral.

Esta tarea diseñada para llevarse a cabo en parejas, también persigue la finalidad de concebir la idea de concebir diversas modalidades empresariales entre los compañeros de clase.

● **Prueba teórico-práctica escrita:**

En la última sesión se realizará una prueba escrita en la que se demuestre la asimilación de contenidos procedentes de las sesiones anteriores. Dicha prueba constará de diferentes tipos de cuestiones que se especificarán en el apartado de instrumentos.

Estas técnicas van a hacer uso de los siguientes instrumentos de evaluación:

● **Cuaderno del profesor:**

Se anotará en cada sesión los aspectos más relevantes reflejados en la técnica de observación: actitud, realización de tareas…

● **Rúbrica observación en el aula:**

Se evalúa el comportamiento del alumnado atendiendo a las observaciones del cuaderno del profesor. Se valorará la participación, la actitud y el material. Los ítems se calificarán entre 3 y 10 puntos.

● **Rúbrica actividades prácticas individuales:**

Esta rúbrica recoge los puntos clave del desarrollo de las actividades prácticas. Se evalúa el trabajo individual realizado en el tiempo de la sesión. Tendrá una calificación de entre 0 y 10. Se valorarán aspectos como el grado de consecución de los objetivos de los ejercicios, la originalidad, la comprensión, el análisis de los motores, etc.

● **Rúbrica trabajo de investigación:**

Se evaluará el grado de detalle con el que se expongan la información, la iniciativa para ampliar más allá de lo requerido, la bibliografía consultada y la innovación.

● **Examen práctico escrito:**

Compuesto de preguntas de diferente tipología para tratar tanto los aspectos más teóricos y los más prácticos. Constará de:

- Diez preguntas tipo test de respuesta múltiple.

- Dos preguntas de respuesta corta.
- Un caso práctico, en el que se identifiquen los bloques funcionales y se definan las entidades y componentes de un videojuego presentado.

Se limitará el tiempo de ejecución de estas partes.

En la siguiente tabla (Tabla 4. *Resumen de técnicas e instrumentos de evaluación.*) se detallan las técnicas utilizadas y los instrumentos asociados a cada una de ellas, así como la ponderación de cada una:

Tabla 4. *Resumen de técnicas e instrumentos de evaluación.*

Tabla resumen de técnicas e instrumentos de evaluación				
Procedimientos de evaluación (métodos)	Actividades prácticas individuales	Trabajo investigación	Prueba teórico-práctica escrita	Observación
Porcentaje	50%	-	35%	15%
Instrumento de evaluación (medios)	Rúbrica actividades individuales	Rúbrica trabajo investigación	Examen	- Cuaderno del profesor - Rúbrica observación
Momento	En las sesiones 1, 2, 3, 4	Desde la 1ª sesión hasta la 5ª en casa	Última sesión	Durante el desarrollo de la UD

A continuación, se exponen las rúbricas mencionadas:

Tabla 5. *Rúbrica actividades prácticas individuales.*

Rúbrica actividades prácticas individuales						
Ítem	**Ponderación**	**Mal (0 ptos)**	**Bajo (20% ptos)**	**Medio (50% ptos)**	**Alto (85% ptos)**	**Excelente (100% ptos)**
Asimilación o análisis del problema	20%	No entiende el problema	Apenas comprende el problema	Comprende el problema, pero no en su totalidad	-	Comprende el problema en su totalidad
Resolución del ejercicio	45%	No se ha realizado ninguna resolución	Apenas resuelve el problema	Resuelve el problema de forma parcial	Resuelve el problema en su totalidad, pero no de manera óptima	Resuelve el problema en su totalidad de manera eficiente e incorpora funcionalidades extra.
Realización de documentación	35%	No incluye la documentación requerida	La documentación aportada es insuficiente	Incluye los puntos básicos requeridos de forma escueta	Incluye una descripción detallada de los aspectos requeridos	Además de lo anterior, incluye otros aspectos más avanzados que no se le han pedido

Tabla 6. *Rúbrica trabajo de investigación.*

Rúbrica trabajo de investigación					
Ítem	**Ponderación**	**Bajo (20% ptos)**	**Medio (50% ptos)**	**Alto (85% ptos)**	**Excelente (100% ptos)**
Contenido	50%	El contenido es insuficiente, no ha tratado los puntos clave.	Se han incluido los aspectos mínimos del tema asignado. No ha entrado en detalles.	Se han incluido los aspectos clave que se pedían de forma correcta.	Además de recoger la información que se pedía, ha añadido puntos extra que conllevan una mayor reflexión acerca del tema. Relaciona conceptos con otras disciplinas.
Expresión	15%	Es difícil entender las ideas que expone.	-	-	Se expresa de forma adecuada haciendo que el contenido sea entendible.
Bibliografía	10%	Añade una sola fuente sin contrastar la información.	Recoge la información de diversas fuentes no contrastadas.	-	Consulta una cantidad adecuada de fuentes contrastadas.
Parte de conclusiones	25%	No se ha realizado reflexión o ha sido demasiado pobre.	Incluyen una pequeña reflexión sin profundizar demasiado.	-	La reflexión es adecuada, madura y realista.

Tabla 7. *Rúbrica observación en el aula.*

Rúbrica observación en el aula				
Ítem	**Ponderación**	**Bajo (30% ptos)**	**Medio (70% ptos)**	**Excelente (100% ptos)**
Participación en clase	60%	Sus aportaciones son mínimas y se limitan a las preguntas que le hace la profesora directamente.	Participa cuando lo requiere la profesora y sus aportaciones son correctas.	Participa voluntariamente y las aportaciones que realiza son enriquecedoras para la clase.
Actitud hacia los demás	20%	Apenas se relaciona con los compañeros, se relaciona de forma poco adecuada con los demás.	Se relaciona bien con los compañeros colaborando solo cuando así lo requiere la profesora.	Ayuda siempre a los compañeros intentando resolver sus dudas, sin que suponga una copia del trabajo.
Material propio y del aula	20%	Usa el material de forma inadecuada o bien se le olvida a menudo el material que debe traer.	En líneas generales suele traer el material necesario y suele respetar el material del aula.	Siempre trae el material necesario, cuida las infraestructuras y el material del aula.

3.5.3. Criterios de calificación

El mecanismo de evaluación de esta unidad didáctica se va a realizar mediante el modelo clásico. A continuación, se detalla la información presente en la tabla resumen de técnicas e instrumentos de evaluación (*Tabla 3. Resultados de aprendizaje, criterios de calificación e instrumentos.*), donde se relacionan los instrumentos y técnicas de evaluación empleados.

Actividades prácticas individuales

Esta parte constituye el 50% de la calificación de la unidad didáctica. En ella se evalúan los conceptos teórico-prácticos, así como el manejo y uso de los motores de videojuegos y entornos de desarrollo, mediante una serie de ejercicios prácticos en los que se ponen de manifiesto los conceptos explicados en las sesiones. Los ejercicios se adaptan al contenido visto en cada sesión.

Estas actividades se realizarán en horario de clase, de forma individual, en el aula. Es importante destacar que se valorará el aprovechamiento del tiempo. Al final de cada sesión, los alumnos enviarán el resultado del trabajo a la profesora a través de la entrega habilitada para ello. Se tendrá en cuenta la ayuda entre compañeros siempre que no suponga una copia del trabajo, en cuyo caso tendrán que repetir la actividad.

Las actividades se evaluarán con la ayuda de la Rúbrica de actividades individuales (Tabla 5*. Rúbrica actividades prácticas individuales.*) mencionada anteriormente. Así la nota oscilará entre 0 y 10 puntos. Las actividades de todas las sesiones tendrán el mismo peso.

El alumno debe conseguir al menos una puntuación de al menos 4 puntos en este apartado para superar esta unidad didáctica.

La calificación mediante esta técnica se calcula como:

Nota prácticas individuales: 0,5 * nota media actividades prácticas individuales

Prueba teórico-práctica escrita

Esta parte le corresponde el 35% del peso de la unidad didáctica. Consiste en la realización de un examen escrito individual compuesto por diferentes tipos de preguntas relacionadas con los conceptos trabajados en clase con el fin de que el alumno demuestre los conocimientos teórico-prácticos aprendidos. La ponderación de cada parte del examen es la siguiente:

- Diez preguntas tipo test de opción múltiple, cada pregunta bien contestada suma 0,5 puntos. Cada tres preguntas mal respondidas se resta el valor de una acertada.
- Dos preguntas de respuesta corta, que tendrán un valor de 1,25 puntos cada una.
- Un caso práctico que supondrá 2,5 puntos en el examen.

El alumno debe conseguir al menos una puntuación de al menos 4 puntos en este apartado para superar esta unidad didáctica.

La puntuación del examen oscilará entre 0 y 10 puntos. La nota final de esta parte de la evaluación se calcula como:

Nota prueba teórico-práctica = nota examen * 0,35

Observación

Esta parte corresponde al 15% de la calificación de la UD. Los aspectos que se van a tener en cuenta son los siguientes:

- Actitud positiva y participativa
- Aportaciones constructivas
- Colaboración con los compañeros. Respeto por los compañeros y profesora.
- Respeto por material, tanto de la clase como propio.

La observación se tendrá en cuenta durante todas las sesiones, en las que la profesora deberá tomar nota de los aspectos mencionados anteriormente. Además, se hará uso de la Rúbrica de observación en el aula (Tabla 7. *Rúbrica observación en el aula.*) para obtener la calificación de cada uno de los ítems implicados.

Trabajo investigación parejas

El trabajo de investigación parejas no pondera en la nota de la unidad, sino que puede suponer un incremento de la misma de hasta un punto. Se trata de una actividad de ampliación.

Este trabajo se realizará en horario extraescolar, o en horario de clase siempre que el resto de actividades se hayan completado.

Para evaluar esta parte, se utilizará la Rúbrica de trabajo de investigación (Tabla 6. *Rúbrica trabajo de investigación.*), que comprende una calificación de entre 2 y 10 puntos. En esta parte, los dos componentes del grupo obtendrán la misma calificación.

Así, la calificación de esta parte se calcula como: nota trabajo investigación * 0,1

La nota final de esta unidad será la suma de:

+ 50% de la nota de las actividades prácticas individuales
+ 35% de la nota de la prueba escrita
+ 15% de la nota de la observación del profesor a lo largo de la unidad
+ (subir nota) 10% de la nota del trabajo de investigación

Observación: Es imprescindible que el alumno obtenga un mínimo de 4 puntos en las partes de actividades prácticas individuales y prueba escrita para poder ponderar dichas calificaciones en la nota global.

Una vez obtenida la nota final, si esta es menor que 5 puntos, el alumno tendrá que recuperar la evaluación de la unidad. En caso de que la nota final sea mayor o igual a 5 puntos, el alumno habrá superado la unidad.

3.5.4. Recuperación

Para recuperar la unidad didáctica el alumno, dependiendo de la casuística en la que se encuentre deberá:

- En caso de que la nota de las actividades prácticas sea inferior a 4, realizar aquellas actividades prácticas que no haya superado o realizado en su momento. Estas actividades se evaluarán siguiendo la Tabla 5*. Rúbrica actividades prácticas individuales.* y la nota final de esta parte es la media de las calificaciones de las actividades.
- En caso de que la nota de la prueba escrita sea inferior a 4, hacer un examen de características similares al realizado al finalizar la unidad didáctica.

En el momento de recuperación, ya no se tiene en cuenta la nota de observación ni la nota de la actividad de ampliación, y se ponderará la nota de cada parte como el 50%.

El alumno habrá superado la evaluación si la nota de la prueba es mayor o igual a 5.

La recuperación de la unidad didáctica se realizará en 3 momentos, al final del trimestre, al final de curso (en la convocatoria ordinaria) y en la convocatoria extraordinaria.

3.6. Atención a la diversidad

En el segundo curso de Programación de Aplicaciones Multiplataforma se encuentra escolarizado un alumno que padece hipoacusia leve en el oído izquierdo, que se caracteriza por tener dificultad para escuchar sonidos en entornos con ruido, lo cual merma la capacidad para escuchar las explicaciones de la profesora, así como las intervenciones de los compañeros.

La hipoacusia es un caso de atención a la diversidad que, como se señala desde la Consejería de Educación de la Junta de Andalucía, requiere la aplicación de medidas tendentes a la adquisición de competencias clave y la consecución de los objetivos de la materia en cuestión. Por tanto, es un caso de Necesidades Específicas de Apoyo a la Educación (NEAE). Las medidas propuestas para paliar dicha situación serían las siguientes:

- Ubicación estratégica en la clase, sentándose en primera fila en el lateral izquierdo de la clase, para poder escuchar mejor las explicaciones.
- En cuanto a las explicaciones orales por parte de la profesora, irán acompañadas de diapositivas o material visual para el apoyo en el contenido.
- Se procurará que los debates realizados en clase se realicen de forma ordenada para no generar ruido y de esta manera alcanzar la comprensión de estos.

Estas medidas siempre irán acompañadas de las instrucciones que facilite tanto el Departamento de Orientación como el especialista médico que trate al alumno.

Por otra parte, para aquellos alumnos capaces de finalizar las tareas antes que el resto porque su ritmo de aprendizaje sea mayor, se propondrán actividades de ampliación o profundización de las ya realizadas dependiendo de la casuística.

3.7. Situación COVID-19

Debido a las circunstancias especiales derivadas de la situación sociosanitaria provocada por el COVID-19, se hace necesaria la implementación de medidas en la programación de la unidad didáctica con el objetivo de abordar todas las situaciones posibles. A continuación, se detalla la normativa de referencia:

- *INSTRUCCIONES de 6 de julio de 2020, de la Viceconsejería de Educación y Deporte, relativas a la organización de los centros docentes para el curso escolar 2020/2021, motivada por la crisis sanitaria del COVID19.*
- *CIRCULAR de 3 de septiembre de 2020, de la Viceconsejería de Educación y Deporte, relativa a medidas de flexibilización curricular y organizativas para el curso escolar 2020/2021.*
- *Real Decreto-ley 31/2020, de 29 de septiembre, por el que se adoptan medidas urgentes en el ámbito de la educación no universitaria.*

La programación se atiene al modelo diseñado de forma autónoma por el centro, bajo el amparo de la normativa citada anteriormente. Se contemplan varios escenarios posibles:

- Presencialidad. La Consejería de Educación estipula que este será el modelo prioritario si las circunstancias lo permiten, en cuyo caso se adoptarán las siguientes medidas:
    - Mantenimiento del llamado grupo de convivencia escolar, por el que se evitará el contacto del alumnado del grupo con alumnado de otros grupos diferentes.
    - Medidas de distanciamiento social entre alumnado, así como espaciado entre mesas, sillas y ordenadores.
    - Medidas higiénicas, como uso obligatorio de mascarilla en todo momento, desinfección de mobiliario y material antes y después de cada uso, limpieza permanente de manos (el centro proporciona los productos desinfectantes).
    - Asignación fija de puestos a cada alumno dentro del aula.
    - Actividades realizadas en pareja únicamente a través de plataformas virtuales.
    - En caso de la aparición de síntomas relacionados con el virus, se actuará según lo indicado en el protocolo del centro.

- Semipresencialidad (síncrona). Este caso se daría según el porcentaje reflejado en el Plan de Centro y divide la clase en dos grupos que alternan la presencialidad. En este caso las medidas serían:
    - Distanciamiento y medidas de higiene y medidas de desinfección.
    - Asignación fija de puestos.
    - La mitad del grupo recibe las clases presencialmente, mientras que la otra mitad las recibe a través de Google Meet en la misma sesión de forma síncrona.
    - Las actividades prácticas individuales se realizarán todas a través de Moodle, entregando los resultados a la finalización de cada sesión.
    - Al inicio de cada sesión, se ampliará el tiempo de repaso en la actividad de resumen-presentación (20 minutos) según consta en las Instrucciones del 6 de julio de 2020.
    - La prueba teórico-práctica escrita se realizará en la misma sesión para todo el alumnado, el grupo presencial la realizará en formato papel y el grupo online mediante un cuestionario en Moodle y deberá tener la cámara activa.
- Online. Clase mediante Google Meet, entrega de actividades y prueba teórico-práctica en Moodle con cámara activa.

En las situaciones de semipresencialidad y docencia online, se tendrá en cuenta la siguiente casuística:

A. Alumnado que no tiene acceso a las TIC, en cuyo caso el centro proporcionará los medios tecnológicos pertinentes.

B. Atendiendo al Artículo 10 del Real Decreto 31/2020, los contenidos y los criterios de evaluación se establecerán en función de la presencialidad. En este caso, los criterios serán los establecidos en el currículo. En el caso de las otras dos modalidades, se establecerán unos mínimos para superar positivamente la unidad didáctica. En el caso de la unidad didáctica *Introducción al desarrollo de videojuegos*, los criterios básicos mínimos son:

    a) Se han analizado los componentes de un motor de juegos.
    b) Se han identificado los elementos que componen la arquitectura de un juego 2D y 3D.
    c) Se han analizado entornos de desarrollo de juegos.

4. Bibliografía

Belli, S., & López Raventós, C. (2008). Breve historia de los videojuegos: Simone Belli, Cristian López. *Athenea Digital: revista de pensamiento e investigación social, Nº. 14*, 159-179.

blackmores. (2017, junio 26). *Entrevista: José Luis Domínguez*. http://videojuegosretro-upm.blogspot.com/2017/06/entrevista-jose-luis-dominguez.html

Boost Library Documentation. (s. f.). Recuperado 2 de junio de 2021, de https://www.boost.org/doc/libs/?view=condensed

Cal, J. (2018, junio 7). *Ubisoft CEO Yves Guillemot: «Game streaming will replace all gaming platforms»*. TechSpot. https://www.techspot.com/news/74986-ubisoft-ceo-yves-guillemot-game-streaming-replace-all.html

COCOS. (2020, febrero 11). What Will A Game Engine Look Like In Ten Years? *Cocos Blog*. https://www.cocos.com/en/what-will-a-game-engine-look-like-in-ten-years

Coumans, E., & Bai, Y. (2021). *PyBullet Quickstart Guide*. PyBullet Quickstart Guide. https://docs.google.com/document/u/1/d/10sXEhzFRSnvFcl3XxNGhnD4N2SedqwdAvK3dsihxVUA/edit?usp=embed_facebook

CRYENGINE Features. (2021). CRYENGINE. https://www.cryengine.com/features

CRYENGINE Support—General. (2021). CRYENGINE. https://www.cryengine.com/support/view/general

Desarrollo de juegos con Visual Studio. (2021). Visual Studio. https://visualstudio.microsoft.com/es/vs/features/game-development/

Desarrollo Español de Videojuegos (DEV). (2021). *Libro Blanco del Desarrollo Español de Videojuegos 2020* (p. 118). Asociación Española de Empresas Productoras y Desarrolladoras de Videojuegos y Software de Entretenimiento. https://dev.org.es/libroblancodev2020

Facebook/folly. (2021). [C++]. Facebook. https://github.com/facebook/folly (Original work published 2012)

FMOD - Core API Guide. (2021, mayo 25). https://fmod.com/resources/documentation-api?version=2.00&page=core-guide.html

FMOD Studio. (2021). FMOD. https://www.fmod.com/studio

Getting started with Direct3D. (2018). https://docs.microsoft.com/en-us/windows/win32/getting-started-with-direct3d

Godot Engine—Features. (2021). Godot Engine. https://godotengine.org/features

Gregory, J. (2018). *Game Engine Architecture, Third Edition* (Tercera). A K Peters/CRC Press.

Gurr, F. (2020, septiembre 17). *Eclipse Wiki*. Eclipse Foundation. https://wiki.eclipse.org/Main_Page

Havok Physics. (s. f.). Havok. Recuperado 2 de junio de 2021, de https://www.havok.com/havok-physics/

Havok Physics for Unity. (s. f.). Unity Manual. Recuperado 2 de junio de 2021, de https://docs.unity3d.com/Packages/com.havok.physics@0.1/manual/index.html

Historia de los videojuegos en España. (2021). elotrolado. https://www.elotrolado.net/wiki/Historia_de_los_videojuegos_en_Espa%c3%b1a#1988_-_Un_crimen_en_la_abad.C3.ADa

KhronosGroup/Vulkan-Samples. (s. f.). GitHub. Recuperado 26 de mayo de 2021, de https://github.com/KhronosGroup/Vulkan-Samples

Mercado, P. (2021, abril 5). *Los 25 Motores de Videojuegos Gratis y de Pago*. IndustriaAnimacion.com. https://www.industriaanimacion.com/2021/04/los-25-motores-de-videojuegos-gratis-y-de-paga/

Metal | Apple Developer Documentation. (s. f.). Recuperado 1 de junio de 2021, de https://developer.apple.com/documentation/metal

Moore, G. E. (1965). Cramming more components onto integrated circuits. *Electronics*, *38*(8), 114-117.

NVIDIA PhysX SDK 3.3.4 Documentation. (s. f.). Recuperado 1 de junio de 2021, de https://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/guide/3.3.4/Manual/Introduction.html#physics-vs-physx

OpenAL: Cross Platform 3D Audio. (s. f.). OpenAL. Recuperado 1 de junio de 2021, de https://openal.org/

OpenGL - The Industry's Foundation for High Performance Graphics. (2011, julio 19). The Khronos Group. https://www.khronos.org/opengl/

PathEngine—Intelligent agent movement. (2021). PathEngine. https://www.pathengine.com/overview

Ramos, D. (2017, mayo 19). Historia de los videojuegos en España | Década de los 80s y 90s. *Into the games*. https://www.intothegames.com/la-pulga-commandos-historia-videojuego-espanol/

Realtime Intro—Photon. (s. f.). Photon Engine. Recuperado 2 de junio de 2021, de https://doc.photonengine.com/en-us/realtime/current/getting-started/realtime-intro

Recast & Detour. (2021). GitHub. https://github.com/recastnavigation/recastnavigation

Rider. (2021). JetBrains. https://www.jetbrains.com/es-es/rider/

Rider para Unity. (2021). JetBrains.

SDL Wiki. (s. f.). SDL. Recuperado 3 de junio de 2021, de https://wiki.libsdl.org

Segal, M., & Akeley, K. (2019). *OpenGL 4.6 (Core Profile)*.

Syracuse University. (2019, enero 17). With Viewership and Revenue Booming, Esports Set to Compete with Traditional Sports [Syracuse University]. *Syracuse University Blog*. https://onlinegrad.syracuse.edu/blog/esports-to-with-traditional-sports/

The .NET Tools Blog. (2020). JetBrains Blog. https://blog.jetbrains.com/dotnet/2020/04/22/rider-unreal-engine-eap/

Tutorials for SFML 2.5. (s. f.). SFML. Recuperado 3 de junio de 2021, de https://www.sfml-dev.org/tutorials/2.5/

Unreal Engine—Features. (2021). Unreal Engine. http://www.unrealengine.com/features

Vallejo Fernández, D., & Martín Angelina, C. (2015). *Desarrollo de Videojuegos: Un Enfoque Práctico.: Vol. Arquitectura del Motor*. OpenLibra. https://openlibra.com/es/book/desarrollo-de-videojuegos-un-enfoque-practico-vol-1-arquitectura-del-motor

Vulkan. (2021, mayo 10). *Home | Vulkan | Cross platform 3D Graphics*. https://www.vulkan.org/

Williams, A. (2017). *History of Digital Games*. Taylor & Francis Group. https://learning.oreilly.com/library/view/history-of-digital/9781317503804/

Wwise. (2021). Audiokinetic. https://www.audiokinetic.com/products/wwise/

Wwise SDK 2021.1.1—Spatial Audio. (2021). Audiokinetic. https://www.audiokinetic.com/library/edge/?source=SDK&id=spatial_audio.html

Zhu, L. (2021). The psychology behind video games during COVID-19 pandemic: A case study of Animal Crossing: New Horizons. *Human Behavior and Emerging Technologies*, *3*(1), 157-159. https://doi.org/10.1002/hbe2.221