

Física Virtual

Iker Iturralde Tejido

Tutor: Santiago Alba Carrillo

Cotutora: María Isabel Anaya Bernabé

IES Villa de Valdemoro

25/10/2019

ÍNDICE

1.-	Introducción.....	1
2.-	Conceptos previos físicos.....	3
2.1.-	Mecánica clásica y cuántica.....	3
2.2.-	Mecánica clásica y sus partes	5
2.2.1.-	<i>Cinemática</i>	9
2.2.2.-	<i>Estática</i>	9
2.2.3.-	<i>Dinámica</i>	9
3.-	Motores	10
3.1.-	Motores gráficos	11
3.2.-	Motores de física	13
3.3.-	Motores de videojuegos	14
3.3.1.-	<i>Historia</i>	16
3.3.2.-	<i>Motores de videojuegos más conocidos</i>	18
3.4.-	PHYSX.....	19
4.-	Hardware y tarjetas gráficas.....	20
4.1.-	Estructura de las tarjetas gráficas.....	22
4.2.-	Funciones de las tarjetas gráficas.....	25
5.-	Unreal Engine	27
5.1.-	Características.....	28
5.2.-	Funcionamiento	29
5.3.-	Realidad virtual	33
6.-	Simulaciones.....	34
6.1.-	Lanzamiento vertical.....	34
6.1.1.-	<i>Fundamentos físicos</i>	34
6.1.2.-	<i>Simulación</i>	35
6.1.3.-	<i>Comparación</i>	35

6.2.-	Lanzamiento oblicuo.....	37
6.2.1.-	<i>Fundamentos físicos</i>	37
6.2.2.-	<i>Simulación</i>	38
6.2.3.-	<i>Comparación</i>	38
6.3.-	Plano inclinado	39
6.3.1.-	<i>Fundamentos físicos</i>	39
6.3.2.-	<i>Simulación</i>	39
6.3.3.-	<i>Comparación</i>	40
6.4.-	Engranajes.....	41
6.4.1.-	<i>Fundamentos físicos</i>	41
6.4.2.-	<i>Simulación</i>	43
6.4.3.-	<i>Comparación</i>	43
6.5.-	Centro de masas y de gravedad	44
6.5.1.-	<i>Fundamentos físicos</i>	44
6.5.2.-	<i>Simulación</i>	45
6.5.3.-	<i>Comparación</i>	45
7.-	Conclusiones.....	46
8.-	Referencias	47
9.-	Bibliografía de imágenes	49

ÍNDICE DE LAS IMÁGENES

Ilustración 1 "Vector"	5
Ilustración 2 "Vector desplazamiento y velocidad media"	6
Ilustración 3 "Desplazamiento y trayectoria"	6
Ilustración 4 "Fuerzas"	8
Ilustración 5 "Suma de fuerzas"	8
Ilustración 6 "Resta de fuerzas"	8
Ilustración 7 "Fuerzas perpendiculares"	8
Ilustración 8 "Descomposición de fuerzas"	9
Ilustración 9 "Comparación de realidad con CGI"	11
Ilustración 10 "Pong"	14
Ilustración 11 "Ultima Underworld"	16
Ilustración 12 "Doom"	17
Ilustración 13 "Half-life"	18
Ilustración 14 "Requisitos del sistema"	20
Ilustración 15 "Salidas tarjeta gráfica"	23
Ilustración 16 "Refrigeración líquida"	24
Ilustración 17 "Funcionamiento de una tarjeta gráfica"	26
Ilustración 18 "Blueprints"	30
Ilustración 19 "Transformación manual"	31
Ilustración 20 "Transformación interactiva"	31
Ilustración 21 "Características de los actores"	32
Ilustración 22 "Lanzamiento vertical"	34
Ilustración 23 "Circunferencia primitiva"	42
Ilustración 24 "Prueba engranajes"	43
Ilustración 25 "Centro de masas"	44
Ilustración 26 "Prueba centro de masas"	45

1.- INTRODUCCIÓN

La informática está llamada a ser una de las grandes bases de nuestra futura sociedad, ya sea mediante la implementación de robots en los ámbitos más comunes de la vida de una ciudad, como en el posible estudio de cualquier ámbito científico gracias a la simulación en un ordenador. La simulación de los diferentes ámbitos está en progreso, pero destaca mayoritariamente el estudio físico y la rama de la mecánica.

Para conocer el estado actual de la simulación, se habrán explicado anteriormente diferentes conceptos como la física, los motores de videojuegos y la herramienta necesaria para su uso. A continuación, se explicará el motor al cual nos referiremos para comprobar la cuestión y por último se realizarán diferentes experimentos con el propósito de verificar esta.

De este modo, sabremos si es posible realizar o no los diferentes experimentos en un ordenador facilitando el estudio físico desde el punto de vista académico, pero también profesional. Además, se podrá lanzar una hipótesis bastante cercana a lo que sucederá de aquí a unos años.

La principal motivación para la realización de este proyecto se basa principalmente en la unión de dos temas que resultan bastante atractivos puesto que hablan de cosas reales y visibles. Asimismo, supone un reto tratar de demostrar gráficamente diferentes conceptos que, en teoría, pueden llegar a ser algo confusos, y esto es algo que siempre ayuda en el estudio de cualquier materia.

Para finalizar, me gustaría dar las gracias a todos los profesores y personas que han estado brindándome su apoyo durante la realización del proyecto; sin ellos y su constante apoyo muy probablemente no me hubiera sido posible la realización a tiempo del trabajo. Entre ellos destacar a mi tutor Santiago Alba Carrillo, mi cotutora Maribel Anaya Bernabé y a Beatriz del Río Aguirre, por su paciencia y ayuda, que, sin la cual, acabar este proyecto hubiera resultado imposible.

Abstract

Computing is called to be one of the great bases of our future society, thanks to the possibility of studying any scientific field by simulating it on a computer.

To know the current state of the simulation, different concepts such as physics, video game engines and the necessary tool for its use will have been explained previously. Next, the engine you which we will refer to when verifying the issue, will be explained. Finally, different experiments will be carried out in order to check this.

Through this project, it will be possible to know the possibility of carrying out the different experiments on a computer or not, greatly facilitating the study and a hypothesis quite close to what will happen in a few years can be launched.

2.- CONCEPTOS PREVIOS FÍSICOS

Para entender el trabajo de investigación es necesario poseer unos conocimientos previos. Comencemos por el principio, ¿qué es la física? La Física, como explica la RAE, es una ciencia que estudia las propiedades de la materia y la energía, y las relaciones entre ambas.

El objetivo principal de esta ciencia es descubrir el funcionamiento del universo, tanto a nivel molecular como a nivel universal. Para cumplir esta tarea, la Física estudia diferentes ámbitos como luz, sonido, calor, trabajo y fuerza; aunque en este proyecto lo que más interesa es la mecánica, que trata la velocidad, la aceleración y el desplazamiento.

Como bien se ha comentado anteriormente, la Física es una ciencia, y como ciencia se basa en el método científico para poder avanzar y evolucionar. Aquí es donde se sitúa el tema de este proyecto de investigación, si realmente el programa funciona ajustándose a la realidad, las fases de experimentación se podrían solventar con mayor facilidad y lo que probablemente sea más importante, a un coste más bajo; aunque de todo este programa y su funcionamiento se hablará más adelante.

2.1.- MECÁNICA CLÁSICA Y CUÁNTICA

A lo largo de la fase experimental de este trabajo, se realizarán diferentes proyectos relacionados con el tema de la mecánica, pero es necesario diferenciar los dos tipos que existen de esta rama física.

El primero y el más reciente es la mecánica cuántica, se encarga del estudio de las partículas subatómicas. Dada su complejidad su estudio es bastante limitado y el *software*¹ informático, o al menos el que se ha utilizado en el proyecto, no alcanza el nivel suficiente como para simularlo adecuadamente.

El segundo, la mecánica clásica, gracias a la cual se experimentará en la parte final de este trabajo, esta rama de la mecánica es una formulación utilizada para el estudio de los movimientos tanto de objetos muy pequeños (sin llegar al nivel cuántico) como a nivel astrónomo, dando lugar a resultados muy precisos.

¹ *Software*: conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.

Dependiendo de la formulación utilizada podemos encontrar:

- *Mecánica analítica.* Puesto que esta rama de la mecánica se basa en una formulación más abstracta y general que la vectorial, no será tratada en este trabajo. Simplemente comentar que su base se encuentra en la formulación lagrangiana y la formulación hamiltoniana; ambas presentan diferentes problemas físicos principalmente relacionados con el sistema de referencia y, por tanto, se dificultaría su estudio.
- *Mecánica vectorial.* También conocida como mecánica newtoniana, es la principal y más utilizada. Se basa en las leyes que Newton enunció en 1642; previamente la Física mecánica solo trataba de estudiar el movimiento realizado por los cuerpos y cómo se originaba, sin llegar en ningún momento a estudiar la causa del mismo.

Newton, tras su estudio realizado sobre los trabajos de Galileo y Kepler sobre el movimiento de los planetas, quiso dar una explicación más ajustada al movimiento elíptico. Para ello, su primer paso fue otorgar la idea de masa de un cuerpo considerándola como prioritaria. La masa actualmente está definida como propiedad intrínseca del cuerpo que mide su resistencia al ser acelerado.

Posterior a este estudio, Newton enunció tres leyes que relacionarían la mencionada masa con la fuerza:

1. *“Ley I. Un cuerpo permanece en su estado inicial de reposo o movimiento con velocidad uniforme a menos que sobre él actúe una fuerza exterior no equilibrada.*
2. *Ley II. La aceleración de un cuerpo es inversamente proporcional a su masa y directamente proporcional a la fuerza exterior resultante que sobre él actúa.*
3. *“Ley III. A toda acción se opone siempre una reacción igual; o sea, las acciones mutuas de dos cuerpos uno sobre otro están siempre dirigidas hacia las partes contrarias.”*

Tras la redacción estas leyes, la mecánica clásica ha evolucionado hasta el día de hoy, en la cual, pueden diferenciarse tres partes explicadas en el siguiente punto.

2.2.- MECÁNICA CLÁSICA Y SUS PARTES

Este apartado explica las diferentes partes en las que se divide la mecánica, donde hay ciertos conceptos que comparten entre ellas y que a continuación serán explicados:

- Sistema de referencia: es el punto de partida para cualquier estudio físico, que se organiza a través de este punto y un sistema de ejes (x e y).
- Movimiento: está definido como todo cambio de posición que experimentan los cuerpos en el espacio respecto al tiempo y la distancia recorrida. Existen varios tipos de movimiento entre los que destacan: los rectilíneos cuya trayectoria es una recta; y curvilíneos, cuya trayectoria es una línea curva.
- Magnitud vectorial: magnitud que además de requerir un valor numérico, requiere la especificación de su sentido y dirección; por tanto, se puede confirmar que posee las características propias de un vector, que son:
 - Módulo: valor numérico que indica el tamaño del vector.
 - Dirección: orientación del segmento de recta en el espacio que la contiene.
 - Sentido: indicado con la punta de la flecha en uno de los extremos del vector.

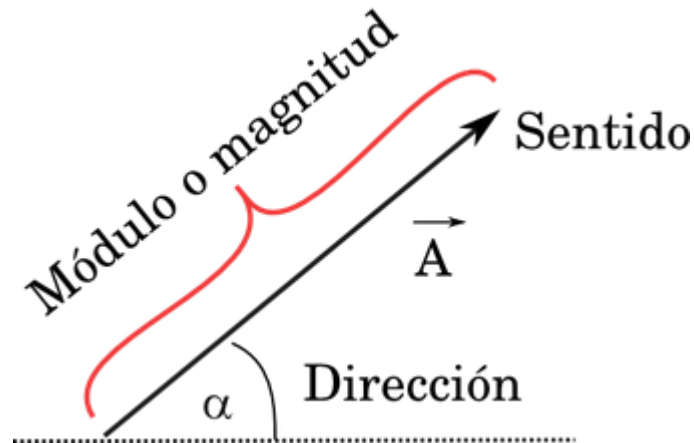


Ilustración 1 "Vector"

- Desplazamiento: magnitud vectorial que define la longitud de la trayectoria comprendida entre la posición inicial de un objeto y la final. En el sistema internacional su unidad es el metro.

$$\vec{\Delta r} = \vec{r}_f - \vec{r}_o$$

Siendo:

$\vec{\Delta r}$: vector desplazamiento.

\vec{r}_f : posición final.

\vec{r}_o : posición inicial.

- Velocidad media: es la magnitud vectorial de un cuerpo que se mueve entre dos puntos en un tiempo determinado. En el sistema internacional su unidad es el metro/segundo.

$$\vec{v} = \frac{\vec{\Delta r}}{\Delta t}$$

Siendo:

\vec{v} : velocidad media

$\vec{\Delta r}$: vector desplazamiento

Δt : variación del tiempo

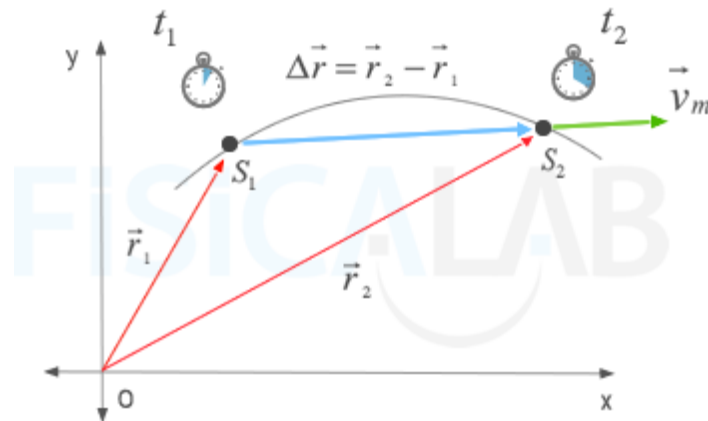


Ilustración 2 "Vector desplazamiento y velocidad media"

- Trayectoria: como la define la RAE: "línea descrita en un plano por un cuerpo en movimiento".

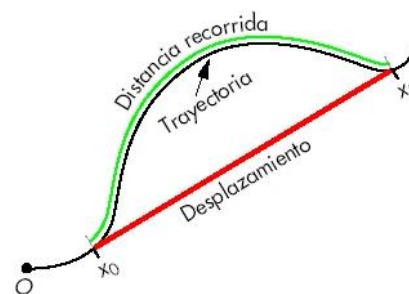


Ilustración 3 "Desplazamiento y trayectoria"

- Aceleración: magnitud vectorial física caracterizada por la rapidez con que varía la velocidad de un cuerpo. Su unidad en el sistema internacional es el metro/segundo²

$$\vec{a} = \frac{\vec{\Delta v}}{\Delta t}$$

Siendo:

\vec{a} : aceleración.

$\vec{\Delta v}$: variación de velocidad.

Δt : variación de tiempo.

- Fuerza: toda acción capaz de producir cambios en el movimiento o estructura del cuerpo. Es una magnitud vectorial capaz de explicar los cambios producidos. Su unidad en el sistema internacional es el newton.

$$F = m * \vec{a}$$

Siendo:

F: fuerza

m: masa en kilogramos.

\vec{a} : aceleración

Existen varios tipos de fuerzas:

- Peso: fuerza que ejerce la gravedad sobre un objeto. En el caso de una caída libre, esta será la única fuerza que se ejerza sobre el cuerpo.

$$P = m * \vec{g}$$

Siendo:

P: peso

m: masa en kilogramos.

\vec{g} : gravedad, siempre valdrá 9.8 m/s².

- Normal: fuerza de contacto que es ejercida por la superficie sobre la que está apoyada un cuerpo.
- Tensión: fuerza que se produce cuando se aplica a un cuerpo elástico como una cuerda.
- Rozamiento: fuerza que se origina como oposición al movimiento en dos superficies en contacto.

$$F_{roz} = \mu * N$$

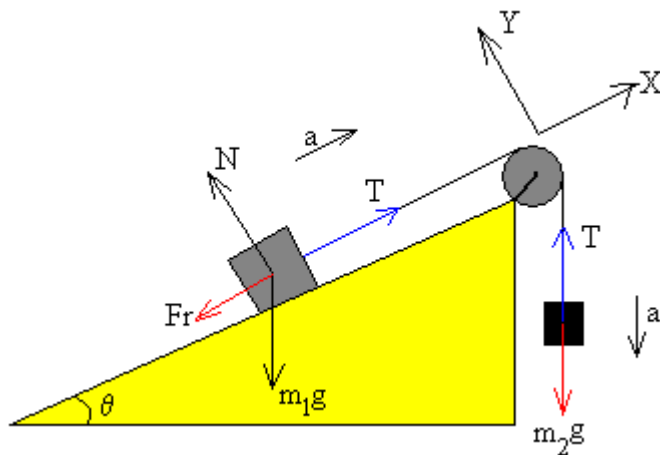
Siendo:

F_{roz} : fuerza de rozamiento.

μ : coeficiente de rozamiento, variará según el material de la superficie.

N: fuerza normal.

A continuación se expone un caso donde se muestran todas las fuerzas explicadas:



N: fuerza normal.

Fr: fuerza de rozamiento.

m_1g : fuerza de peso del cuerpo 1.

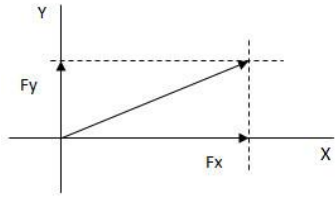
T: fuerza de tensión.

m_2g : fuerza de peso del cuerpo 2.

Ilustración 4 "Fuerzas"

Dependiendo del sentido y la dirección de las fuerzas existen diferentes tipos de operaciones:

Misma dirección y sentido	La fuerza resultante será la suma de las fuerzas presentes. Ejemplo: $F_r = F_1 + F_2$	<p>Ilustración 5 "Suma de fuerzas"</p>
Misma dirección, pero sentidos contrarios	La fuerza resultante será la diferencia de las fuerzas presentes. Ejemplo: $F_r = F_1 - F_2$	<p>Ilustración 6 "Resta de fuerzas"</p>
Dirección y sentido perpendiculares	El módulo y la fuerza resultante son diferentes. La fuerza resultante será igual a la suma de las componentes en el eje x y en el y por separado: $\vec{F}_r = F_2(\vec{i}) + F_1(\vec{j})$ El módulo será igual a la hipotenusa de la suma de cada componente: $F_r = \sqrt{F_1^2 + F_2^2}$	<p>Ilustración 7 "Fuerzas perpendiculares"</p>

<p>Diferente dirección y sentido y sin formar un ángulo de 90°</p>	<p>Se aplica la misma teoría que en el caso anterior, pero es necesario conocer, que en caso de que el vector no se sitúe ni a 90° ni a 0°, este vector se descompondrá en el dos, según el eje x o el eje y.</p> <p>Para ello, se multiplicará el valor de la fuerza por el coseno del ángulo que forme para el eje x; y lo mismo, pero con el seno para el eje y.</p>	 <p><i>Ilustración 8 "Descomposición de fuerzas"</i></p>
--	---	---

2.2.1.- Cinemática

Parte de la mecánica que estudia el movimiento de los cuerpos, sin tener en cuenta las causas que lo originan. Para ello es necesario conocer la posición, la velocidad y la aceleración del cuerpo en cada instante. (1)

2.2.2.- Estática

Parte de la mecánica que estudia los cuerpos sobre los que actúan fuerzas y momentos cuyas resultantes son nulas, de forma que permanecen en reposo o en movimiento uniforme. (2)

2.2.3.- Dinámica

Parte de la mecánica que describe la evolución en el tiempo de un sistema físico en relación con las causas que provocan cambios de movimiento. (3)

3.- MOTORES

Un motor hace referencia al *software* que ejecuta una determinada tarea en muchas aplicaciones de *software*; por ejemplo, en un motor gráfico, de videojuegos o de física, de los cuales se hablará a continuación. Aunque hay que mencionar a los motores de bases de datos (servicio principal que protege, almacena y procesa los datos), de búsqueda (sistemas informáticos que trabajan buscando datos en internet) y los de transcripción (sistema que permite convertir cualquier algoritmo en un listado de acciones), muy importantes, aunque no reseñables durante este trabajo. Un motor posibilita obtener mejor rendimiento para el programador y mejores resultados al usuario; sin embargo, no es necesario usar un motor gráfico para la elaboración de un videojuego, pues con conocimientos básicos de programación pueden crearse grandes cosas gracias a lenguajes de programación como C++, pero el uso del motor facilita y logra juegos más complejos, elaborados y atractivos para el usuario. Un símil válido sería el de un tractor que ayuda a la producción agraria.

Fue en 1989 cuando surgió el primer motor de videojuegos que se conoce y este no tiene un nombre como tal; está reconocido por ser el motor que evolucionó del usado en *Space Rogue* y ser usado en *Ultima Underworld*, de los que se hablará más adelante.

La creación de estos motores se da gracias a los kit de desarrollo de *software*, *software development kit (SDK)*, que están definidos como un conjunto de herramientas, *librerías*² y *API's* que ayudan a la creación y programación de aplicaciones para un entorno tecnológico concreto. Estos SDK permiten al programador contar con las herramientas adecuadas y así ganar tiempo y esfuerzo; por ello, los SDK aportan los siguientes recursos (4):

- Los API: interfaz de programación de aplicaciones que proporcionan un conjunto de funciones, rutinas, estructuras de datos, clases y variables, con las cuales se puede manipular el sistema sin tener un conocimiento pleno sobre él, destaca el OpenGL.
- Los IDE: entorno de desarrollo integrado, ayudan a escribir con mayor facilidad sobre el lenguaje.
- Código de ejemplo: nos proporciona un punto de partida sobre el cual nos será más fácil poder empezar a programar.
- Emulador del entorno: donde poder simular el resultado hacía donde se dirija el programa a realizar.

² *Librería: conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, que ofrece una interfaz bien definida para la funcionalidad que se invoca.*

Es importante destacar dentro de los SDK's dos elementos muy utilizados en multitud de aplicaciones: el *DirectX*, o sus versiones anteriores, que facilitan las tareas relacionadas con la multimedia; y el STL, una librería de plantillas estándar con multitud de estructuras comunes y algoritmos para poder procesarlas.

3.1.- MOTORES GRÁFICOS

Comúnmente, se asocia un motor gráfico de forma directa a un motor de videojuego. De esta manera, existe la creencia popular pero errónea, de que ambas cosas son lo mismo. El motor de un videojuego cuenta con un motor gráfico, pero un motor gráfico no cuenta con un motor de videojuego. Para poner un ejemplo, todo ser humano debe poseer un corazón, pero no por poseer un corazón debes ser un ser humano.

La principal función de un motor gráfico es *renderizar* gráficos tanto en 2 como en 3 dimensiones. Por eso anteriormente se comentó que un motor gráfico no debe ser especialmente el de un videojuego; puesto que estos también se pueden servir para diferentes funciones como la creación de las famosas *computer-generated imagery (CGI)*, imágenes creadas artificialmente muy utilizadas en películas como "*Fast and Furious 7*", tras la muerte del actor Paul Walker.



Ilustración 9 "Comparación de realidad con CGI"

Pero ¿qué es un gráfico? La palabra gráfico tiene multitud de definiciones, pero hablando de informática sería: "cualquier imagen generada por ordenador", que mediante el proceso denominado *renderizar*, permite que se puedan ver imágenes creadas mediante el lenguaje binario. Y con esto se llega a la base de la informática, el lenguaje o sistema binario hace alusión a la forma de "comunicarse" entre ordenadores. Al igual que las personas utilizan el sistema decimal, números del 0 al 9, en los ordenadores se creó un sistema mediante el uso de un simple 1 o un 0. Usar el sistema binario tiene una sencilla explicación, los ordenadores internamente trabajan siempre con dos niveles: hay corriente eléctrica o no la hay, está activado o no está activado. (5)

El 0 y el 1, adoptaron el nombre de bit y, al igual que se puede convertir números del sistema decimal al sexagesimal, también se puede convertir números del sistema binario al decimal y, con la inclusión del código *ASCII*³, los ordenadores pueden leer y transformar caracteres y texto a forma digital. Y de la misma manera, usando solo cierta cantidad de bits para definir el color de un pixel⁴, se puede obtener una imagen y posteriormente un video totalmente renderizado.

Retomando el tema de los motores; es cierto que la confusión de llamar al motor de videojuegos motor gráfico tiene una base muy potente; la pieza clave actual para la creación de un videojuego es, sin duda, el motor gráfico. Por ello, están muy relacionados entre sí, tanto que durante su historia muchas veces se les llega a mencionar juntos. Esto se debe a los motores han ido evolucionando principalmente según las necesidades en los videojuegos; y por ello, su historia será comentada en el apartado 3.3.1

Motores gráficos más conocidos

Los motores gráficos más conocidos son (6):

- id Tech 1: conocido como motor de Doom, creado por John Carmack y que fue revolucionario en su época.
- OGRE: uno de los motores gráficos más prominentes. Se desarrolló en C++ y permite la integración de motores de sonido, física, colisiones...
- Irrlicht Engine: motor de gráficos 3D también escrito en C++. Es gratuito y es usado para la creación de juegos *RPG*⁵ de carácter *indie*⁶.
- Aleph One: motor creado para shooters⁷ 3D desarrollado por Bungie antes de ser comprada por Microsoft.
- Axiom Engine: motor de visualización de gráficos 3D que provee una abstracción completa del API y soporta DirectX. Contiene un modelo *scene graph*⁸ y soporta *shaders*⁹ complejos.

³ *ASCII*: código estándar americano para el intercambio de información.

⁴ Color de pixel: está definido por la cantidad de luz roja, verde y azul.

⁵ *RPG*: videojuego en el usuario asimila la vida de un personaje.

⁶ *Indie*: videojuego independiente creado por individuos sin apoyo financiero

⁷ *Shooter*: videojuego en el cual se usan armas de fuego.

⁸ *Scene graph*: organiza la situación lógica y espacial de una escena.

⁹ *Shaders*: programa que realiza los cálculos para colocar correctamente las sombras.

3.2.- MOTORES DE FÍSICA

La principal función de un motor de física es simular todas las acciones que posean cierta relación con sistemas físicos, como la dinámica de objetos rígidos, de cuerpos deformables, de fluidos... Y de gran importancia un sistema de detección de colisiones.

Los motores de física son, junto a los ya mencionados motores gráficos, una parte fundamental de un motor de videojuegos, dado que le otorgan al videojuego un entorno más real, destacando la destrucción del entorno. Sin embargo, aunque sean una parte fundamental, un motor no es sencillo de realizar y su mejora y evolución continúan en investigación. Debido a esto, las compañías de videojuegos no crean sus propios motores de físicas, sino que implementan los ya creados por otras empresas.

Su uso no solo está dirigido hacia los videojuegos, también se utiliza para la simulación con fines científicos, puesto que puede procesar modelos complejos con gran cantidad de cálculos y una alta precisión numérica.

La base matemática sobre la que se construye un motor de físicas es la ecuación del plano. Con esto se construye gran parte de la funcionalidad que debe tener un motor de físicas, las colisiones con los objetos. Si se aplica la función a un punto cualquiera y el resultado es mayor que cero, quiere decir que el punto está encima del plano, y si es negativo, es que está debajo. (7)

Como ya se ha comentado, los motores de física incluyen un sistema de detección de colisiones que, como su propio nombre indica, permiten al *software* conocer cuando un objeto colisiona con otro o con cualquier otra parte del mundo; sin embargo, implementar este sistema puede no ser del todo rentable computacionalmente hablando, porque es necesario otorgar una *hitbox*¹⁰ precisa a cada cuerpo, es decir, que implica añadir un gran número de formas complejas y comprobaciones que pueden ralentizar o empeorar la jugabilidad. Para reducir este coste se pueden elegir los objetos que colisionaran de forma eficiente y a los que no lo hagan, se les podrá simplificar su hitbox.

¹⁰ *Hitbox: también conocida como volumen delimitador para un conjunto de objetos, es un volumen cerrado que contiene completamente la unión de los vértices de un objeto y así facilitar las colisiones u operaciones geométricas.*

Motores de física más conocidos

Los motores de física más conocidos son (6):

- *Havok physics*: dependiendo del producto que se vaya a realizar, existe una versión gratuita y otra comercial. Es conocida por utilizarse en el juego *Assasins Creed* y ser compatible con multitud de plataformas entre las que destacan PlayStation 3 y Xbox 360. Es de los más conocidos a nivel mundial debido a la fama que obtuvo el título mencionado.
- *PhysX*: explicado con mayor profundidad en el punto 3.4 debido a una característica especial, además de ser el motor de físicas que se usará en la fase experimental de este proyecto.
- *ODE: Open Dynamics Engine*, simula cuerpos rígidos y colisiones, y alcanzó gran popularidad en 2005-2006. Actualmente ya no se encuentra en desarrollo.
- *Bullet*: motor de física para objetos 3D, utilizado en juegos como *Grand Theft Auto IV* e incluso en películas como *Cómo entrenar a tu dragón*
- *Newton Game Dynamics*: motor muy específico para detectar la colisión, pero que sacrifica la velocidad de procesamiento.
- *Box2D*: motor de física más conocido en cuanto a objetos 2D. Destaca su uso en el famoso juego *Angry Birds*.

3.3.- MOTORES DE VIDEOJUEGOS

Un estudio realizado por la AEVI (Asociación Española de Videojuegos) estimó que, solo en España, en 2013 se gastaron 762 millones de euros en el ámbito de los videojuegos, tanto en los propios juegos como en las consolas o plataformas donde jugar.

La creación de los primeros videojuegos se realizó enteramente mediante lenguajes de programación, como es el caso del famoso *Pong*, creado en 1972. Aunque este no sea el primer videojuego de la historia, si es el más reseñable, conocido y marcador de una etapa en la industria.

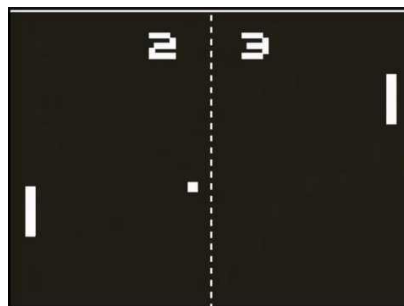


Ilustración 10 "Pong"

Tras largos años de creación de videojuegos sin un motor de videojuegos, desde 1958 con *Tennis for two*, la inclusión de este en el sector otorgó un salto de calidad.

Un motor de juegos es un conjunto de sistemas de *software* que facilitan a un creador el desarrollo de un juego sin la necesidad de escribir mucho código. Aunque desafortunadamente no existe un motor con el cual se pueda crear cualquier tipo de juego, porque se requiere de optimizaciones necesarias para las diferentes plataformas de *hardware* y los géneros, sí existen motores con un propósito general que abarcan una gran gama de videojuegos, aunque poseen una mayor complejidad. (6)

También se encuentran motores de videojuegos con un propósito específico, que se desarrollan para un único género y son más sencillos de realizar.

La creación de estos motores no es sencilla, de hecho, es un proceso bastante largo y complicado, y en muchos casos las compañías que los desarrollan venden sus licencias para permitir crear juegos con este *software*. (6)

Para la creación de un motor bien diseñado es necesario tener presentes las siguientes características (6):

- **Mantenibilidad:** debe diseñarse con la posibilidad de evolucionar acorde a las necesidades de los creadores que vayan a utilizar el motor. Este cambio siempre será inevitable, debido al continuo avance en las tecnologías y la invención de nuevas.
- **Portabilidad:** otorga la facilidad de usar el motor en diferentes dispositivos.
- **Confiabilidad:** incluyendo la fiabilidad, la protección y la seguridad. No sería agradable para el creador que el uso de un motor pueda causar daños en el ordenador y el sistema.
- **Eficiencia:** malgastar recursos del sistema como la memoria o el procesamiento, hará que solo los usuarios con una calidad de *hardware* superior puedan utilizar este programa, por tanto, cuanto mayor rendimiento y menos recursos utilice, mejor será el motor de juegos.
- **Usabilidad:** un motor debe ser sencillo de utilizar, por tanto, poseer una *interfaz*¹¹ apropiada y una documentación adecuada será indispensable.

¹¹ *Interfaz: todo tipo de dispositivo y de información que permiten al usuario tomar el control del software, logrando que el jugador interactúe con este.*

Además, es necesario definir los géneros de juegos para los cuales se desarrollará el motor, saber qué características funcionales tendrá y las áreas del motor, entre las cuales se encuentran los gráficos y las físicas, mencionados anteriormente, y otras relevantes como el audio, la conectividad a la red...

Una vez diseñada la estructura que tendrá el motor, solo quedará desarrollarlo y poner en marcha el proyecto.

3.3.1.- Historia

Muchos esperarían ver en primer lugar al motor de Doom o id Tech 1, dado que gracias a este se "acuñó" el término de motor; pero la historia no dice lo mismo.

En 1989 la compañía *Origin Systems* lanza al mercado el videojuego *Space Rogue*, con gran impacto en ordenadores como *Commodore 64* y *Apple II*. Fruto de este beneficioso resultado, la empresa decidió contratar más desarrolladores para crear *Ultima Underworld*, un juego que vendría con su propio motor gráfico, el cual permitió decorar paredes, pisos, techos y objetos varios. Además, permitió la creación de superficies inclinadas y generar un efecto 3D con la visualización de alturas. Sin embargo, la principal desventaja que poseía fue su gran lentitud para los ordenadores de la época, dado que también utilizaba una física para crear movimiento, por tanto, quedó rápidamente obsoleto. (8)



Ilustración 11 "Ultima Underworld"

Tras este primer motor, se encuentra el *Voxel Space*, un juego de simulación que posteriormente daría su nombre al motor que mezclaría dos palabras claves en los motores, volumétrico y pixel. Ambas dan el nombre a *voxel* que es la forma de representar objetos tridimensionales sin usar vectores. Esto permitía una mayor fluidez en los títulos creados en este motor e incluso se llegó a usar en algunos objetos y personajes de la película *Blade Runner*.

En 1993, salió al mercado el famoso videojuego *Doom*, un videojuego que permitía al jugador subir y bajar en los diferentes mapas. Aunque esto no es del todo cierto, el motor con el cual se desarrolló, el *id Tech*, no era precisamente un motor gráfico 3D, sino más bien la consecuencia de niveles bidimensionales que no permitían jugar un nivel encima de otro y, por ende, las alturas eran simuladas utilizando transportadores, jugando con las diferencias de altura matemática, la separación del ambiente y la posibilidad de mirar hacia arriba y abajo; cuando en realidad había pasado a una habitación continua en el mismo plano. (8)



Ilustración 12 "*Doom*"

Ken Silverman, fue un adolescente que tras ver jugar a su hermano a un videojuego decidió empezar en la industria hasta lograr ser contratado por 3D Realms y construir las bases de Build, un motor 3D que permitía paredes en ángulo y con una fluidez bastante aceptable. Tras contactar con *id Software* (creadores de *Doom*), reescribió el motor renderizando mundos 2D que otorgaban la sensación 3D. El motor siguió en desarrollo durante años otorgando diferentes mejoras, pero para su uso era necesario un procesador Pentium y una tarjeta gráfica VESA, que en 1995 era bastante costosa.

Con el motor *Quake* en 1996, llegó el 3D real y la industria tomó el rumbo que lleva hoy. Para hacer esto posible, el motor *Quake* utilizaba mapas preprocesados que se cargaban según el jugador pasaba por ellos, solventando los problemas de fluidez y velocidad que los anteriores motores fueron incapaces de solucionar.

En 1998 surgen dos motores importantes: el primero, el *Renderware*, fue un intento de motor retrocompatible con diferentes plataformas que decidió ignorar el uso de la tarjeta gráfica, y fue esto lo que le condenó; y el segundo, el "Quake Engine 2" o "id Tech 2", el cual, con la inclusión del soporte nativo de OpenGL y gracias al uso de la librería DLL, permitió la alta modificación por la comunidad aunque parte del código no fue compartido al resultar muy robusto.

En diciembre de 1999, el código de Quake es liberado y difundido rápidamente por internet. Los usuarios comenzaron a modificar el motor o crear nuevos basados en este.

Destaca el *GoldSource Engine*, que significó la introducción de Valve Corporation al mercado y sirvió para la creación del famoso juego *Half-Life* y como base del motor *Source*. (8)



Ilustración 13 "Half-life"

Por último, es necesario mencionar la aparición del *Unreal Engine* en 1998. Este motor pudo haberse presentado únicamente como un motor de *FPS*¹²; sin embargo, también se convertiría en la base de títulos RPG. Con un lenguaje de programación propio, un editor de código, un programa de modificación y un editor de mapas, logró competir con el grande de la época "id Tech 2"; además de la adición de un sistema de colisiones, una versión de filtrado de texturas y la iluminación. Posteriormente fue evolucionando y en la actualidad se encuentra en la versión 4.

3.3.2.- Motores de videojuegos más conocidos

Los motores de videojuegos más conocidos son (6):

- *Unreal Engine*: motor que usará durante mi fase experimental y que será explicado con profundidad en el punto 5.
- *Panda 3D*: motor gratuito que incluye gráficos, audio, detección de colisiones y otras herramientas.
- *RAGE Engine (Rockstar Advanced Game Engine)*: desarrollado por la empresa Rockstar famosa por la creación de títulos mundialmente conocidos como *Grand Theft Auto V* y *Red Dead Redemption*. Permite la creación de mundos extensos y utiliza el mencionado motor de físicas *Bullet*.
- *CryENGINE*: motor comercial que maneja su propia física, audio y animaciones. Es conocido por ser utilizado en juegos como *Far Cry* y *Crysis*.
- *Naughty Dog Game Engine*: motor desarrollado específicamente para PlayStation 3. Maneja una infinidad de objetos dinámicos de físicas independientes, interacción ambiente-animación y efectos de iluminación. Destaca en la creación de la saga *Uncharted*, el cual es considerado prácticamente una película al tener una transición entre las cinemáticas y la parte jugable casi inidentificable. Estas cinemáticas también están creadas con el motor de videojuegos.

¹² *FPS*: juegos de disparos en primera persona.

3.4.- PHYSX

Se le da la misma denominación al motor de físicas (*software*) y al *hardware* que incorpora algunas tarjetas gráficas Nvidia. La misión del segundo es liberar de los cálculos matemático-físicos al procesador o CPU del ordenador. Las tarjetas gráficas con Physx hacen que los videojuegos funcionen de manera más fluida, de hecho, generalmente en las configuraciones gráficas de los videojuegos se permite activar o desactivar según las necesidades del jugador; si se prefiere una mejor jugabilidad y fluidez la desactivará, pero, sin embargo, si prefiere mejores gráficos, la activará.

Este *software* está actualmente implementado en multitud de juegos y es utilizado por más de 10000 desarrolladores, entre los que encontramos a *Epic Games*, empresa encargada del desarrollo del motor *Unreal Engine*.

Physx de Nvidia es un SDK de motor de física en tiempo real patentado. Está realmente avanzado y permite realizar las físicas de cualquier juego a tiempo real. (9)

Gracias a la tecnología de Physx, se puede decir que los juegos “cobran” vida: las paredes se pueden derrumbar, la destrucción de cristales es posible, los agentes naturales como el viento y el agua son semejantes a la realidad...

Unreal Engine 4 utiliza el motor de física PhysX 3.3 para conducir sus cálculos de simulación física y realizar todos los cálculos de colisión. PhysX proporciona la capacidad de realizar una detección de colisión precisa, así como simular interacciones físicas entre objetos dentro del mundo. Tener física en un juego ayudará a mejorar el valor de inmersión de cada escena, ya que ayuda a los jugadores a creer que están interactuando con la escena y que la escena está respondiendo de una forma u otra. (10)

En la reciente actualización de la versión de Unreal Engine a 4.23, se ha implementado el motor de físicas “Chaos” que ha sustituido a Physx, y que presenta notables mejoras en el campo de la destrucción masiva. Al ser tan reciente no se ha incluido una explicación del mismo en el proyecto.

4.- HARDWARE Y TARJETAS GRÁFICAS

Se ha hablado de que un motor gráfico es el *software* que se utiliza para la renderización de diferentes programas y para mejorar sus resultados y facilitar su elaboración; una tarjeta gráfica sería la parte correspondiente de *hardware*. No es necesaria en un ordenador para su uso, pero sin ella la computadora sería incapaz de ofrecer imagen. Además, si se trata de tarjetas gráficas de mayor calibre, la calidad de esta será mayor e incluso permitirá otro tipo de funciones, aunque la principal es la de recoger la información producida por el *software* en la placa base y procesarla para dar imagen.

El *hardware*, es la parte física de un ordenador, aquello que se puede ver y tocar. En los inicios era muy básico. Con el paso de los años se pueden observar multitud de mejoras ligadas siempre a las necesidades de los *software* informáticos. Es importante saber que la parte de *hardware* de un ordenador siempre está relacionada con la parte de *software*, y también es necesario saber que si los componentes de *hardware* no son lo suficientemente potentes los programas se verán recortados. Esto se puede demostrar en cualquier videojuego, en el apartado “requisitos del sistema”. Si el sistema es igual o superior a estos requisitos, se podrá utilizar dicha aplicación:

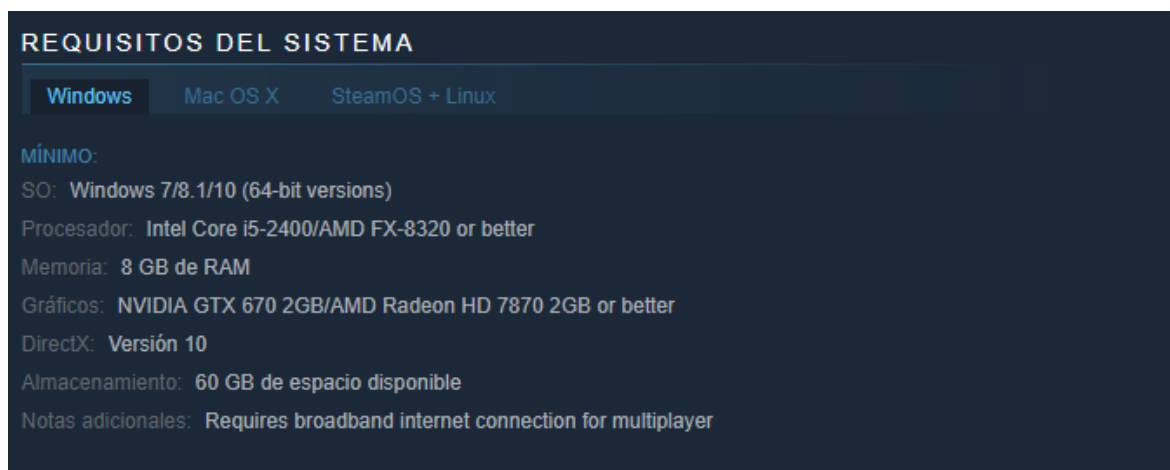


Ilustración 14 "Requisitos del sistema"

Como se puede observar, no solo se encuentra la tarjeta gráfica (en la imagen: “gráficos”) como requisito del sistema; si bien es cierto que la tarjeta proporcionará la imagen de lo que se quiere, el resto de los componentes, proporcionarán al sistema diferentes funciones totalmente necesarias para el uso del *software*, incluido el que se utilizará durante la parte experimental.

Entre estos componentes se encuentran:

- La placa base, también llamada placa madre, es la parte principal de todo ordenador sobre la cual se añadirán el resto de los componentes
- El procesador, que es el encargado de realizar las operaciones.
- La memoria *RAM*¹³, donde se cargan las instrucciones que realizará el procesador
- La fuente de alimentación, que suministrará la energía necesaria al ordenador para su funcionamiento.

Sin estos componentes un ordenador sería incapaz de realizar su función y, en consecuencia, una tarjeta gráfica sería completamente inútil; por ello, aunque haya sido de forma leve, es necesario conocer cada uno de los componentes que harán posibles este estudio.

Actualmente existen dos tipos de tarjetas gráficas: las que están integradas en el procesador, que cuentan con una menor potencia y cogen parte de la memoria RAM, pero son de un precio más reducido que las dedicadas, las cuales tienen una estructura completamente propia de la cual se hablará en el punto 4.1, y aumentan su potencia, al igual que el precio, pero si es cierto que es posible cambiarla por otra sin necesidad de cambiar el procesador. (11)

Relacionado con las tarjetas gráficas, surgió el primer problema a la hora de realizar el proyecto. Como ya se ha comentado, sin una buena tarjeta gráfica y el resto de los componentes acorde a su calidad, existen multitud de programas y juegos que no podrían funcionar correctamente o incluso no se podrían usar.

El *software* sobre el cual se experimentará, el *Unreal Engine*, no tiene unos requisitos muy altos para su uso, pero para su uso óptimo sí; y por ende, en los ordenadores del instituto IES Villa de Valdemoro fue imposible realizar dicha parte experimental. En el primer ordenador personal donde se probó la aplicación, no funcionaba de forma correcta ya que se originaban multitud de parones durante las pruebas o funcionaba en unos *fps*¹⁴ demasiados bajos, dificultando la parte experimental pero no impidiéndola.

¹³ *RAM: memorias de acceso aleatorio.*

¹⁴ *Fps: "frames per second" o frames por segundo, hace referencia a la cantidad de imágenes o frames se reproducen en 1 segundo, formando así un video, lo óptimo sería uno 30/60 fps.*

4.1.- ESTRUCTURA DE LAS TARJETAS GRÁFICAS

Reducido a lo básico, una tarjeta gráfica es muy simple: una parte de entrada de información, una *GPU*¹⁵, donde procesar la información, y una salida de vídeo que, por la conexión de un cable hacia el monitor, proporcionaría la imagen. Esto no es erróneo, pero si se especifica más, encontramos componentes que lo complican (12):

- El procesador, chipset o GPU: su principal función es la de aligerar la carga de trabajo del procesador principal del ordenador; por ello está optimizada para el cálculo en coma flotante o en *notación científica*¹⁶, predominante en funciones 3D. Es la parte más importante de una tarjeta gráfica. Gran parte de la información que se proporcione sobre una estará generalmente relacionada con la GPU. De esta información destacan tres características: la frecuencia del reloj del núcleo, velocidad en ciclos por segundo (medidas en hertzios) con la cual el GPU realiza las operaciones más básicas; el número de núcleos, se encargan de ejecutar las instrucciones del procesador (en este caso pasar del lenguaje binario a video); y el número de pipelines o hilos, encargados de traducir una imagen 3D compuesta por vértices y líneas en una imagen 2D compuesta por píxeles.
- Memoria RAM gráfica: en caso de estar integrada será la misma que la de la CPU; en caso contrario, la tarjeta incorporará una RAM específica para ella.
- RAMDAC: conversor de señal digital a analógico de memoria RAM; es decir, es capaz de transformar las señales digitales producidas por un ordenador en una señal analógica reproducible en un monitor. Aunque con la creciente popularidad de los monitores digitales, el RAMDAC está quedando obsoleto puesto que la conversión analógica ya no será necesaria.
- Conexiones con la placa base: como se mencionó al inicio, la tarjeta gráfica es la encargada de renderizar un programa y ofrecer una imagen visible al usuario; para ello, necesita recoger la información desde la placa base y procesarla; esta conexión ha ido evolucionando y cambiando de nombre con el paso del tiempo y las mejoras realizadas. Por ello se realizará una breve descripción cronológica de estos conectores:
 - Slot MSX: primer conector que permitía el paso de 8 bits simultáneos.
 - ISA: predominante durante los 80' y permitía el paso de hasta 16 bits.
 - NuBus: primer conector de los primeros ordenadores Apple.

¹⁵ GPU: "graphic processing unit" o unidad de procesamiento gráfico, procesador dedicado a los gráficos.

¹⁶ Notación científica: forma de escritura que acomoda números con valores demasiado grandes en números convencionales multiplicados en base 10.

- MCA y EISA: Disponían de 32 bits y la principal diferencia fue que MCA no era compatible con los anteriores mientras que EISA sí.
- PCI: superó claramente a los anteriores en 1993, con el mismo tamaño de bits, pero con una velocidad muy superior que permitía una configuración dinámica de los dispositivos conectados. PCI-X fue una versión que duplicó el paso de bits y aumentó en 100MHz su velocidad.
- PCIe: surgió en 2004 y es la actual interconexión entre placa base y tarjeta, cuenta con diversas versiones llegando actualmente a la PCIe 5. No se debe confundir con PCI-X, versión de PCI.
- Salidas: permiten la conexión entre la tarjeta gráfica y el monitor. Hoy podemos encontrar cuatro, los cuales se enumeran del más antiguo al más moderno (13):
 - VGA: o también conocida como SVGA (siendo esta una versión más moderna), es un conector completamente analógico, por ende, si se quiere usar un adaptador al resto de conexiones deberá poder transformar la señal analógica en digital. Actualmente solo lo encontramos en tarjetas gráficas antiguas o de gama baja porque ha sido reemplazado por el HDMI.
 - DVI: primer modelo de conector de datos completamente digital que se empleó para transmitir la imagen desde la gráfica hasta el monitor, superando al conector VGA. Podemos encontrar el DVI-I, parte digital y parte analógico, y el DVI-D completamente digital.
 - HDMI: actual principal conector de todas las tarjetas gráficas y conectores del mercado, introducido en el momento exacto de la fiebre de la alta definición (HD) por ser capaz de retransmitir tanto imagen como audio. Podemos encontrar hasta siete versiones de este conector, a cada cual mejor.
 - DisplayPort: pretende ser el conector sustituto del HDMI y llegó recientemente al mercado con este objetivo. Si se desea disponer de uno, serán necesarios un monitor y una tarjeta de alta gama, puesto que en la gama media y baja rara vez se encuentra uno.

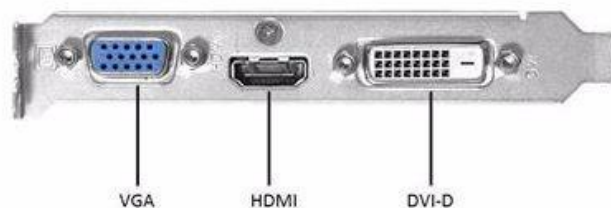


Ilustración 15 "Salidas tarjeta gráfica"

- Refrigeración: una parte poco conocida pero indispensable en el uso óptimo de una tarjeta gráfica. Al igual que en la refrigeración de un ordenador, se encarga de controlar la temperatura a la que se somete una tarjeta debido a las altas cargas de trabajo. Anteriormente podíamos únicamente encontrar dos tipos de refrigeración, pero recientemente está en auge la refrigeración líquida siendo el tercer tipo de refrigeración de los cuales se hablará a continuación:
 - Disipador: dispositivo pasivo que, gracias a estar compuesto por un material conductor de calor, extrae este de la tarjeta.
 - Ventilador: dispositivo activo con partes móviles que aleja el calor gracias a mover aire cercano, es menos eficiente que el disipador.
 - Refrigeración líquida: técnica que utiliza agua para refrigerar la tarjeta al tener el agua mayor capacidad térmica que el aire. Es el más efectivo y llamativo de los tres.

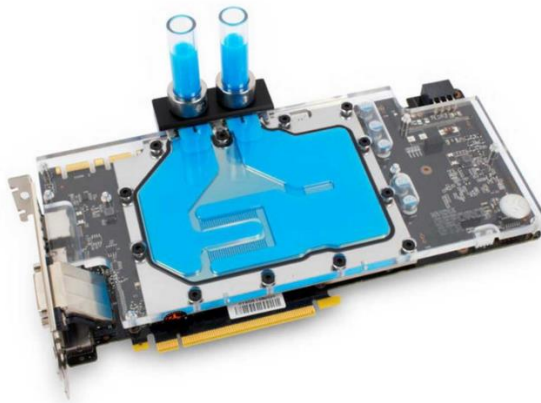


Ilustración 16 "Refrigeración líquida"

- Alimentación: hasta hace unos años, la alimentación de las tarjetas gráficas no era ningún problema; pero, al aumentar cada vez más su potencia considerablemente, también aumenta en consecuencia su necesidad de vatios y, mediante el puerto PCIe, solo se le puede suministrar la cantidad limitada de 150W; por ende, las nuevas y potentes tarjetas gráficas incluyen un conector extra denominado PCIe *power connector* que mediante una conexión directa con la fuente de alimentación y así alcanzar la cantidad de vatios suficientes para el uso óptimo de este.

4.2.- FUNCIONES DE LAS TARJETAS GRÁFICAS

Como ya se ha comentado, la principal función de las tarjetas gráficas es, sin lugar a duda, procesar la información de la CPU para ofrecernos imagen (14):

El primer paso, es recoger la información que la CPU ofrece acerca de la imagen que se quiere ofrecer, así como una empresa facilita a un equipo de publicidad la información necesaria para hacer un anuncio. Esta información es recogida por el puerto *AGP*¹⁷ o el *PCI* y conducida al GPU o procesador de la tarjeta.

En el segundo paso, el procesador realiza unos complejos cálculos matemáticos y geométricos para la elaboración de gráficos. Esto es posible porque, a diferencia de los procesadores centrales, las GPU poseen gran cantidad de núcleos a bajas *frecuencias de reloj*¹⁸; y estos núcleos están principalmente dirigidos al procesamiento de vértices y de píxeles.

El procesamiento de vértices es relativamente sencillo para GPU modernas; en este caso se trata de ordenar los vértices en el espacio según su rotación y determinar qué vértice será gráficamente visible. Por ejemplo, si una persona se encuentra escondida detrás de una farola, la tarjeta gráfica será capaz de procesar qué parte de la persona es visible y cuál no.

Posterior al procesamiento de vértices, se lleva a cabo el procesamiento de píxeles, es decir, los gráficos observables. Este es un proceso mucho más complejo y requiere una mayor carga, puesto que se usan más capas y texturas para dar un resultado más realista. Siguiendo el anterior objeto, en este proceso se determinará la textura de la farola y se logrará diferenciar de plano a la persona de la farola e incluso distinguir su tipo de ropa o su piel.

Tras esto, se logra obtener una señal digital de video adecuada para su almacenamiento, para este tercer paso, esta señal se dirigirá a la RAM, cuya finalidad será almacenar esta señal para poder repetirla tantas veces como sea necesario.

El último paso antes de mostrar la señal será llegar al *RAMDAC*, el cual, según comenta Manuel Ujaldón Martínez en su libro *Procesadores gráficos para PC*, se encarga de traducir la información digital en el *frame buffer*¹⁹ en las señales analógicas que receptiona el monitor. Es una pieza clave en los aspectos relacionados con la

¹⁷ *AGP*: “accelerated graphic port” o puerto de gráfico acelerado, semejante al explicado *PCI*.

¹⁸ *Frecuencias de reloj*: mide la velocidad con la que una computadora realiza las operaciones.

¹⁹ *Frame buffer*: categoría de dispositivos gráficos que representan cada uno de los píxeles de la pantalla.

compatibilidad del monitor que camina a su extinción tras la adopción de las señales digitales por parte de las tarjetas gráficas y la evolución hacia el panel digital de los monitores estándar.

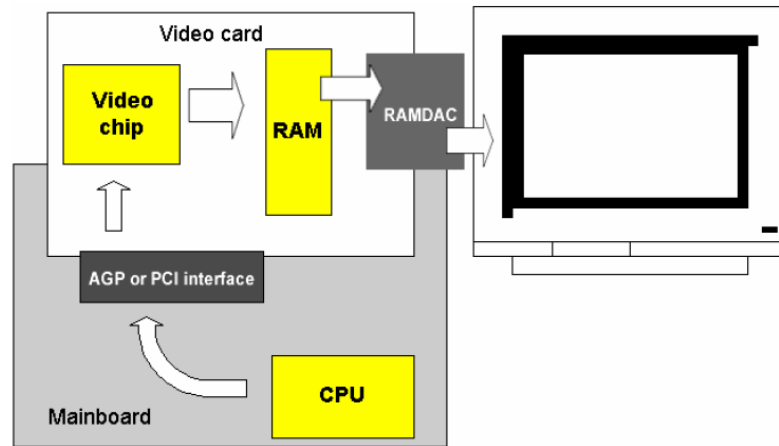


Ilustración 17 "Funcionamiento de una tarjeta gráfica"

Por otra parte, una tarjeta gráfica también puede dar multitud de funciones y utilidades extra, de no ser así, el ser uno de los componentes de más elevado coste no tendría justificación (15)

Entre estas otras funciones se encuentran la sintonización a la televisión, lo cual permitiría el visionado en el monitor usando el ordenador como una televisión. Esto es posible gracias a una tarjeta sintonizadora de video, aunque es cierto que hay algunas que no están integradas en la tarjeta gráfica e incluso se encuentran externas al ordenador. En adición, este tipo de tarjetas permiten la captura tanto de imágenes como de video.

Por último, algunas tarjetas gráficas permiten la decodificación acelerada por GPU a fin de brindar asistencia a las aplicaciones de gran procesamiento. Un claro ejemplo de esto es el *software* OBS²⁰, un famoso programa gratis que permite la grabación de *streamings*²¹ y con la implantación de una nueva extensión facilitará la codificación y el procesamiento de estos.

²⁰ OBS: open broadcaster software, aplicación libre de código abierto para la grabación.

²¹ Streaming: retransmisión en directo.

5.- UNREAL ENGINE

Unreal Engine es un motor de videojuegos creado por la empresa Epic Games en 1998. Este motor ha pasado por tres versiones hasta llegar a la actual, la cuarta. Durante estas versiones se integró el detector de colisiones y las físicas de personajes entre otros aspectos físicos. La actual cuarta versión está en continuo desarrollo, pero como gran implemento se introdujo la realidad virtual de la cual se hablará más adelante.

Muchos redactores que han escrito sobre el *Unreal Engine 4*, lo consideran algo más que un solo motor de videojuegos, de hecho, como dice la Universidad Internacional de Valencia: “Unreal Engine 4 es más que un motor de juego, también se ha utilizado en el desarrollo de películas y vídeos. La cuarta versión de este middleware permite crear elementos y composiciones para generar cinemáticas y vídeos que se pueden usar sin problemas en el séptimo arte.” Y esto no es para menos, el *Unreal Engine* ha sido utilizado en multitud de ámbitos como (16):

- Relacionado a la realidad virtual, la NASA, utiliza actualmente el *Unreal Engine* en el entrenamiento de sus astronautas, y así lo explica el ingeniero de software de la NASA, Mathew Noyes: “La NASA siempre está interesada en integrar tecnología puntera en nuestros sistemas. En un nivel fundamental, entrenar a nuestros astronautas para explorar el espacio es muy parecido a un juego. Sumergimos al usuario en un entorno tridimensional fabricado y los hacemos completar objetivos bajo varias restricciones. Un motor de juego como *Unreal Engine 4* contiene muchos elementos para facilitar el cumplimiento de nuestros objetivos de entrenamiento”
- Compañías automovilísticas como BMW o McLaren utilizan actualmente este motor para facilitar el diseño y creación de sus nuevos coches, y permitir al comprador poder personalizar el interior de su futuro automóvil de manera mucho más sencilla.
- El motor gráfico que posee, también ha sido utilizado para la creación de cinematografías a una velocidad realmente alta si se compara con *Avatar*, película en la cual, para crear una escena no muy larga se podían tardar semanas.
- Por último, la propia empresa ofrece a estudiantes y profesores para el estudio, y yo me encargaré de comprobar la facilidad y funcionalidad de esta opción.

La cantidad y variedad de recursos y posibilidades que ofrece Epic Games en su motor, por únicamente el 5% de los beneficios a partir de 3000€ de ganancia, han hecho de este motor todo un referente en la industria; por ello se espera que con la próxima generación de consolas se libere la versión 5 con grandes novedades. Mientras tanto, la versión 4.22, con la cual se ha experimentado en este proyecto, es suficiente.

5.1.- CARACTERÍSTICAS

La principal característica por la que multitud de empresas y desarrolladores se decantan por *Unreal Engine* a la hora de crear un videojuego es, sin lugar a duda su funcionamiento, el cual es explicado en el próximo punto.

Otra de las características por la que la gente se decanta por este motor, es el ya mencionado coste. Comenzar en esta industria era demasiado caro, ya sea por tener que desarrollar un motor propio o pagar una licencia de gran precio; sin embargo, la llegada de *Unreal Engine* supuso una revolución e impulso de los juegos *Indie*.

Es un motor que presenta gran robustez como *software*, con prácticamente cero fallos y con multitud de posibilidades con las crear tu videojuego, esto se puede demostrar al saber que es utilizado en los famosos juegos *Rocket League*, *Gears of Wars* y *Player Unknown Battlegrounds*. Como dato interesante, este último juego mencionado es la competencia directa del videojuego *Fortnite*, juego emblema de la empresa Epic Games, desarrolladora de Unreal.

Epic Games agregó un detalle que haría de Unreal un motor único: el acceso a su código fuente es abierto, permitiendo copiar, aprender y desarrollar por encima del código del motor.

Por último, Unreal Engine es capaz de desarrollar juegos para todas las plataformas del mercado, desde teléfonos móviles, ofreciendo herramientas que facilitan las pruebas, hasta la realidad virtual, plataforma sobre la cual hablaremos en el punto 5.3, dado a su actual y próxima relevancia en el sector.

5.2.- FUNCIONAMIENTO

La primera vez que alguien se adentra en la creación de un mundo en Unreal Engine es muy probable que no se sepa por dónde empezar. Pero, una vez que se practica un poco, se observa la cantidad de facilidades que ofrece gracias al sistema de los denominados *blueprints*.

Este sistema es extremadamente flexible y potente, ya que brinda a los diseñadores la capacidad de utilizar prácticamente toda la gama de conceptos y herramientas que generalmente solo están disponibles para los programadores. (10)

De forma simple, los *Blueprints* son como “cajas” que ahorran el esfuerzo de aprender un complejo lenguaje, esto se debe a su simpleza. El hecho de unir dos *Blueprints* con flechas que, además, solo es permitido en caso de que sea posible, es, sin lugar a duda, algo de gran calidad.

Los *blueprints* se organizan entre sí para permitir crear, implementar o modificar cualquier elemento, material o personaje y la interacción de estos con el entorno.

Existen varios tipos de *blueprints* como bien recopila Eduardo Pardos (17):

- *LevelBlueprint*: permite manejar la programación de los gráficos en eventos globales. Cada nivel tiene su propio *LevelBlueprint*, aunque a veces son prescindibles. A parte, permiten activar secuencias y generar funcionalidades específicas del juego.
- *Blueprint class*: permite a los creadores añadir contenido y funcionalidades encima de las clases de juego ya existentes. Estas clases son denominadas clase padre y el nuevo *blueprint* heredará las propiedades de este y podrá desarrollar las acciones asignadas. Entre estas clases padre destacan:
 - Actor: objeto que se puede colocar en el mundo.
 - Peón: actor que puede ser controlado por el jugador.
 - Personaje: peón que incluye habilidades como correr y saltar.
 - Modo de juego: define el juego que se está jugando.
- *Blueprint interface*: conjunto de una o más funciones que se pueden integrar en otros *blueprints*. Permite compartir datos entre otros *blueprints*.
- *Animation blueprints*: permite controlar la animación de una malla esquelética, unos gráficos que se editan dentro del editor de *blueprints* donde se realiza la mezcla de animación, el control directo de los huesos de un esqueleto o la actitud de animación final para que una malla se mueva por fotogramas. En resumen, permiten el movimiento.
- *Widget blueprints*: permite la creación de menús dentro del juego.

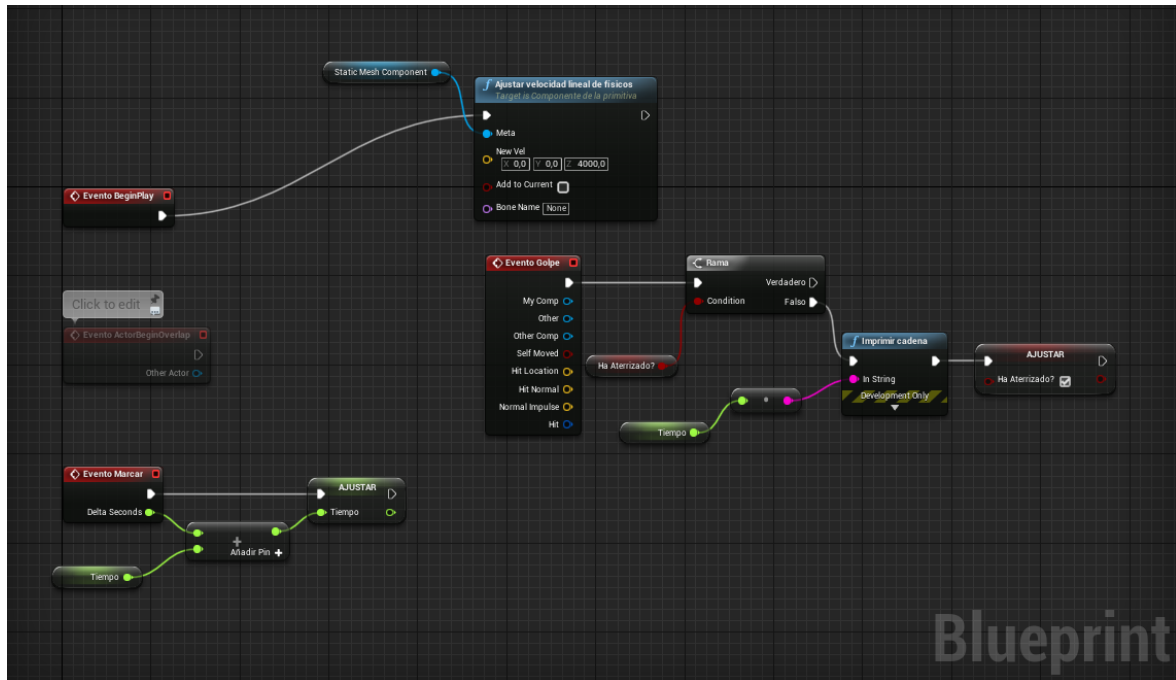


Ilustración 18 "Blueprints"

Desde este punto de vista, el uso de *Unreal Engine* parece demasiado complejo, puesto que el tema de los *blueprints* es la parte más técnica del programa, junto a la programación con lenguaje. Pero, nada más lejos de la realidad, el uso de este *software* es bastante sencillo.

Se comenzará dando una explicación sobre el entorno donde se realizarán los experimentos para posteriormente hablar de los actores y concluir con todos los elementos extras que afectarán a la hora de realizar un experimento.

El entorno en el cual se realizan todos los experimentos, juegos o programas, que se realizan con este programa, se denomina nivel. Un nivel se compone de una colección de mallas estáticas, volúmenes, luces, planos y más, todos trabajando juntos para brindar la experiencia deseada al jugador. Los niveles en *Unreal Engine* pueden variar en tamaño, desde mundos masivos basados en el terreno hasta niveles muy pequeños que contienen algunos actores. (10) Lo más importante del entorno es la configuración mundial, en la cual se puede seleccionar el modo de juego que se quiere y cambiar diferentes aspectos físicos como la gravedad del nivel.

En cualquier nivel creado se pueden colocar objetos denominados actores. Todos los actores son transformables según su posición, rotación o tamaño y tienen campos de entrada numéricos para los ejes X, Y Z. Se puede escribir valores precisos directamente en estos campos para ajustar los actores seleccionados.

Cuando se selecciona más de un actor y los valores de las propiedades difieren, los campos mostrarán *valores* múltiples. En tales casos, ingresar un número hará que se ingrese ese valor para todos los actores seleccionados (10). Existen dos métodos para transformar los actores:

- Manual: consiste en escribir manualmente el número requerido.

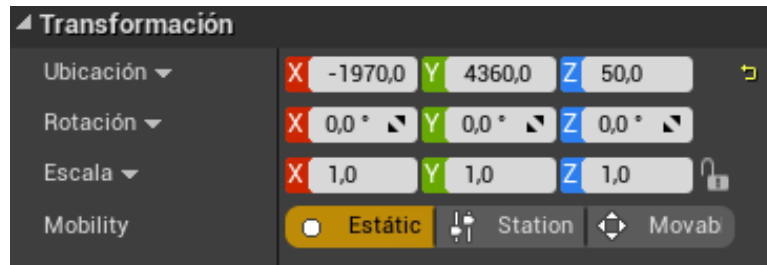


Ilustración 19 "Transformación manual"

- Interactivo: se utiliza el ratón del ordenador para las tres transformaciones. Es mucho más intuitivo pero menos preciso.

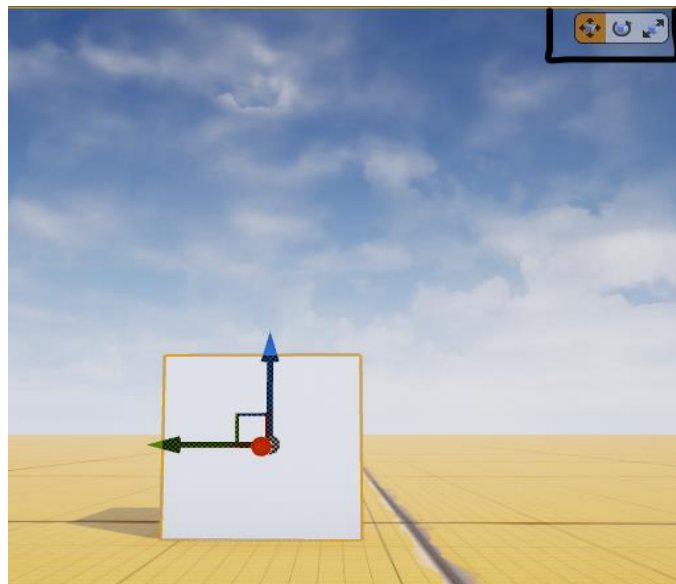


Ilustración 20 "Transformación interactiva"

Como se observa en la imagen, el actor posee tres flechas que salen de su centro hasta el exterior, con las cuales modificar el objeto en los ejes X Y Z. He señalado en negro sobre la imagen dónde se encuentran las tres transformaciones posibles de un actor: la primera opción permite modificar la posición del objeto, la segunda rotarlo y la tercera modificar su tamaño.

Lo más importante relacionado con los actores es la posibilidad que el programa ofrece de activar o modificar diferentes aspectos relevantes del actor, como pueden ser: la *hitbox* o malla estática, el material con el que está creado el actor, y otro tipo de características físicas, como la gravedad, la amortiguación y la masa.

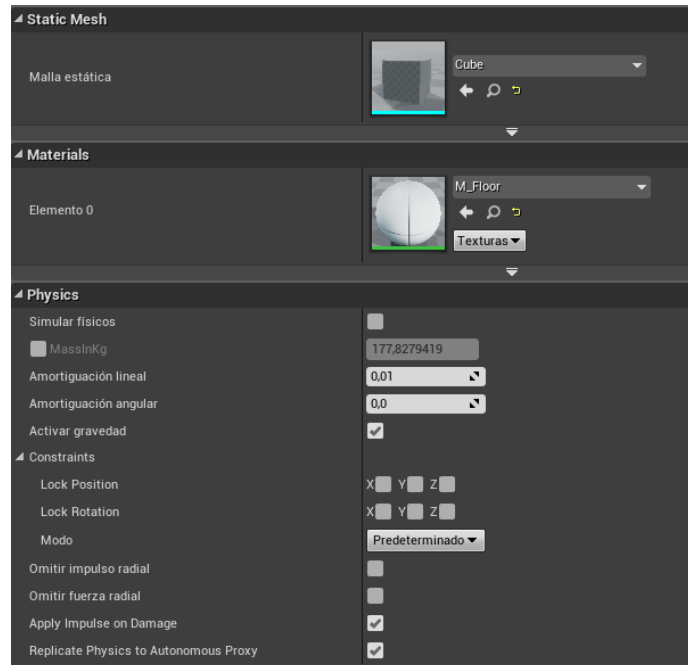


Ilustración 21 "Características de los actores"

Por último, comentar la existencia de los cuatro tipos de actores:

- **StaticMeshActor:** El actor de malla estática es un tipo simple de actor que muestra una malla en la escena. Aunque el nombre implica que el actor es estático, o no puede moverse, la parte estática del nombre se refiere al tipo de malla, una *StaticMesh*, que se utiliza. Estas mallas son estáticas porque su geometría no cambia. Sin embargo, el actor real puede moverse o modificarse de otras formas durante el juego. Estos actores generalmente se usan como geometría mundial y mallas decorativas para crear los entornos del nivel. (10)
- **Brush:** tipo básico de actor que muestra la geometría 3D simple en la escena. Los BrushActors se usan comúnmente para crear entornos de prototipos rápidamente y bloquear niveles para probar el juego. (10)
- **SkeletalMeshActor:** tipo de actor que muestra una malla animada, o SkeletalMesh, cuya geometría se puede deformar, generalmente mediante el uso de secuencias de animación creadas y exportadas desde aplicaciones de animación 3D externas. Estos actores generalmente se usan para cosas como personajes u otras criaturas vivientes, así como maquinaria compleja; cualquier cosa que necesite deformar o mostrar otro movimiento complejo. (10)
- **GameplayPlayStartActor:** un jugador de inicio es un actor que se coloca en el nivel para designar dónde debe comenzar el jugador cuando comienza el juego. (10)

5.3.- REALIDAD VIRTUAL

El término realidad virtual hace referencia a la posibilidad de simular cualquier acción producida en un sistema informático y poder verla y sentirla como si fuera real.

Dicho de otra forma, la realidad virtual permite la inmersión sensorial en un nuevo mundo, basado en entornos reales o no, que ha sido generado de forma artificial, y que podemos percibir gracias a unas gafas de realidad virtual y sus accesorios (cascos de audio, guantes, etc...). (18)

El principal objetivo de esta tecnología es la posibilidad de formar parte de un mundo real o ficticio en el cual se puede ser protagonista, por esto, está llamada a ser la próxima revolución en la industria de los videojuegos y, aunque sus orígenes se remontan a 1957, la realidad virtual tal y como la conocemos, surge en 2010 con las *Oculus Rift*, proyecto comprado posteriormente por Facebook. En 2016 esta tecnología fue denominada como la mejor del año, y multitud de empresas sacaron sus modelos, por lo que la gente pudo tener más acceso a la realidad virtual y, actualmente, a expensas de mejoras no se sabe si se quedará en el olvido o se renovará y cumplirá las expectativas del público, el cual espera una inmersión más realista y completa.

Esta inmersión se está tratando de conseguir con la adición de efectos externos como olores, aire... pero sin lograr un gran impacto. Sin embargo, la posibilidad de que el jugador quede totalmente inmerso en un videojuego sí lograría este impacto necesario.

Pero ¿para qué emplea Unreal Engine la realidad virtual?

Epic Games desarrolló una extensión en el propio *Unreal Engine* llamada *Unreal Engine VR*, en la cual podremos crear escenarios y mundos desde la realidad virtual en tiempo real buscando que el desarrollador sea más eficiente con los cambios, dado que al fin y al cabo, los juegos desarrollados estarán dirigidos a la realidad virtual y, por tanto, el desarrollador ganará tiempo al no tener que estar quitándose las gafas para probar lo que va creando.

6.- SIMULACIONES

Tras conocer los fundamentos principales sobre los cuales se realizará la parte experimental, se dará paso a esta.

El planteamiento de los experimentos será simple, primero se explicarán los fundamentos físicos específicos en los que se basan los experimentos; posterior a esto se simulará el mismo experimento en el programa y, por último, se compararán y se verificarán los resultados.

6.1.- LANZAMIENTO VERTICAL

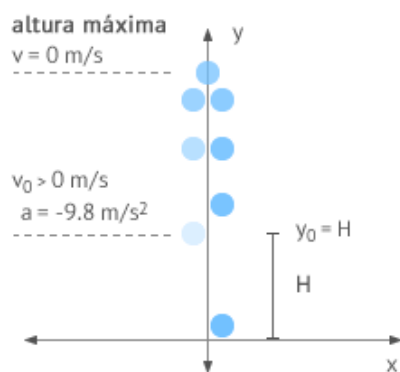
6.1.1.- Fundamentos físicos

Se intenta verificar si el tiempo de caída tras aplicar la misma velocidad inicial al cuerpo, es el mismo tanto en la parte teórica como en la experimental. Para esto nos basaremos en las fórmulas de los movimientos rectilíneos uniformemente acelerados:

$$y = y_0 + v_0 t + \frac{1}{2} \vec{g} t^2$$

$$v = v_0 - \vec{g} t$$

$$t = \frac{v_0}{\vec{g}}$$



Siendo:

y: altura máxima.

y_0 : altura inicial.

v_0 : velocidad inicial.

t: tiempo de subida*.

\vec{g} : gravedad.

Ilustración 22 "Lanzamiento vertical"

Durante el experimento calcularemos solo el tiempo que transcurre desde el lanzamiento hasta que cae de nuevo al suelo, el cual, es el doble al del resultado.

Experimento:

Un cuerpo situado en el suelo es propulsado verticalmente con una velocidad inicial de 40m/s; siguiendo la fórmula mencionada anteriormente, el tiempo que tardará el cuerpo en regresar al suelo debería ser de 8.16s

$$\begin{cases} y = 0 + \frac{40m}{s} * t + \frac{1}{2} * \frac{9.8m}{s^2} * t^2 \\ 0 = \frac{40m}{s} - \frac{9.8m}{s^2} * t \end{cases}$$

$$t = \frac{40m/s}{9.8m/s^2} = 4.08s$$

El objeto deberá tardar 4.08 segundos en alcanzar la altura máxima, y por tanto, 8.16 segundos en llegar al suelo de nuevo.

6.1.2.- Simulación

Tras aplicar la misma velocidad inicial al cuerpo que en la parte teórica se observa un tiempo de 0.16s, prácticamente el cuerpo no se levanta del suelo.

Posteriormente se realizan diferentes incrementos en la velocidad inicial hasta llegar a 1000 “m/s”, y se observa que el cuerpo se levanta y cae en 2.14 segundos, siendo un cuarto del resultado esperado.

Por último, se le da una velocidad inicial a la masa de 4000 “m/s” y los tiempos coinciden. Por tanto, se observa que las magnitudes del programa no son las mismas que en el sistema internacional.

6.1.3.- Comparación

En la primera situación, tras aplicar en ambos casos la velocidad de 40 m/s se puede observar que el programa no va acorde con la teoría, o al menos con el tema de unidades.

Tras esto, y un periodo de pruebas, se descubre que, aplicando 100 veces más la velocidad inicial en el programa, es decir, 4000 “m/s”, el programa coincide con una gran exactitud respecto a la parte teórica; aunque hay variaciones, dado que el programa si aplica fuerzas de rozamiento mientras que en la teoría son situaciones ideales.

Tras saber que en el programa el valor de velocidad inicial de propulsión es 100 veces mayor que en la teoría, se puede afirmar que el programa en vez de usar metros/segundo utiliza los centímetros como medida de distancia.

Por tanto, podemos confirmar que, exceptuando el fallo tras el desconocimiento de las magnitudes, el programa es bastante exacto e incluso ofrece la posibilidad de aplicar fuerzas de rozamiento del aire a un nivel simple, siendo esto bastante complejo en la parte teórica.

6.2.- LANZAMIENTO OBLICUO

6.2.1.- Fundamentos físicos

Se intentará verificar si el tiempo de vuelo es el mismo tanto en la parte teórica como en la experimental. Para el primer caso, nos basaremos en la simplificación de la fórmula de los lanzamientos oblicuos para hallar el tiempo:

$$t = \frac{2v_o * \text{sen}\alpha}{\vec{g}}$$

Siendo:

t : tiempo de vuelo total.

v_o : velocidad inicial.

$\text{sen}\alpha$: seno del ángulo del lanzamiento.

\vec{g} : gravedad.

El alcance máximo del lanzamiento es idéntico en ambas partes, para ello, en la parte teórica nos basaremos en la simplificación de la fórmula pero en este caso para hallar el máximo alcance:

$$x_{max} = \frac{v_o^2 * \text{sen}2\alpha}{\vec{g}}$$

Siendo:

x_{max} : distancia máxima.

v_o : velocidad inicial.

$\text{sen}\alpha$: seno del ángulo de lanzamiento.

\vec{g} : gravedad.

Experimento:

Se tratará de comprobar con diferentes ángulos de lanzamiento pero misma velocidad inicial, si es cierto que con 45° se alcanza la distancia máxima. Para esto:

Velocidad inicial	Ángulo de lanzamiento	Alcance	Tiempo de vuelo
10 metros/segundo	45°	10.19 metros	1.44 segundos
	30°	8.83 metros	1.01 segundos
	75°	5.09 metros	1.96 segundos

6.2.2.- Simulación

Durante la simulación encontramos un gran problema, ya que solo podemos aplicar velocidad inicial en los ejes, y para que la velocidad de los ejes sea igual que en la teoría, deberemos hacer la siguiente operación:

$$\text{Eje x: } v_x = v * \cos\alpha$$

$$\text{Eje y: } v_y = v * \sin\alpha$$

$$\begin{aligned} \text{Cuando el ángulo es } 45^\circ: \quad & v_x = 10\text{m/s} \cdot \cos 45^\circ = 7.07\text{m/s} \\ & v_y = 10\text{m/s} \cdot \sin 45^\circ = 7.07\text{m/s} \end{aligned}$$

$$\begin{aligned} \text{Cuando el ángulo es } 30^\circ: \quad & v_x = 10\text{m/s} \cdot \cos 30^\circ = 8.66\text{m/s} \\ & v_y = 10\text{m/s} \cdot \sin 30^\circ = 5\text{m/s} \end{aligned}$$

$$\begin{aligned} \text{Cuando el ángulo es } 75^\circ: \quad & v_x = 10\text{m/s} \cdot \cos 75^\circ = 2.58\text{m/s} \\ & v_y = 10\text{m/s} \cdot \sin 75^\circ = 9.65\text{m/s} \end{aligned}$$

Tras solventar este problema, nos dirigimos al programa y realizamos los tres experimentos.

Velocidad inicial	Ángulo de lanzamiento	Alcance	Tiempo de vuelo
10 metros/segundo	45°	10.08 metros	1.56 segundos
	30°	8.69 metros	1.13 segundos
	75°	5.07 metros	2.16 segundos

Se observa que los resultados no son siempre iguales, por tanto, el resultado escrito es una media de 10 pruebas realizadas.

Por último, se decide modificar la masa para verificar que los resultados no varían, y aunque se ponga la masa inicial multiplicada 1200 veces, en efecto, los tiempos de vuelo y la distancia no varían.

6.2.3.- Comparación

Tras realizar los tres experimentos, se observa que, al igual que en el primer experimento, los resultados son muy semejantes a la teoría y la variación de ambos casos se debe a que el simulador si aplica la fuerza de rozamiento del aire.

También se observa una variación en los resultados propios del simulador y, aunque sean mínimos, se deben también a esa fuerza de rozamiento aplicada por el aire.

6.3.- PLANO INCLINADO

6.3.1.- Fundamentos físicos

El tercer experimento tiene relación con la ilustración 4, que se mostró sobre las fuerzas casi al inicio del trabajo. En una pendiente inclinada se situará un cuerpo y se trabajará con todas las posibilidades que ofrece el programa, cambiar la inclinación de la pendiente, la masa del cuerpo, aplicar fuerzas...

Se intentará verificar el grado de realismo del programa a la hora de aplicar el rozamiento, de cambiar el ángulo e incluso, modificando la masa del cuerpo. Para ello, y basándonos en la imagen nombrada anteriormente, usaré las fórmulas de los diferentes tipos fuerzas y, en el caso de que la fuerza resultante sea inferior a la suma total que provocaría el movimiento (se debe recordar que solo existe fuerza de rozamiento en caso de que exista movimiento), el objeto continuará su movimiento.

El movimiento que ejercerá o no el cuerpo en el eje X se deberá a una comparación entre las sumas de las fuerzas que se ejerzan en cada sentido. Dado a esto, se realizan tres experimentos

- Si la componente X de la fuerza del peso es superior a la fuerza de rozamiento el cuerpo bajará por el plano.
- Si la componente X de la fuerza del peso es inferior a las fuerzas ejercidas en sentido contrario, el cuerpo subirá por la superficie. Sin embargo, en los experimentos solo existirá fuerza de rozamiento y esta no podrá ejercer una fuerza superior a la del peso.
- Si la componente X de fuerza del peso es igual a la ejercida por el rozamiento, el cuerpo se mantendrá quieto.

6.3.2.- Simulación

Tras conocer la composición de fuerzas comprobamos que el programa nos permite realizar multitud de experimentos. Nos centraremos en los tres campos mencionados para comprobar la exactitud del programa:

1. Cambio de ángulo de pendiente.
2. Cambio en la masa del cuerpo.
3. Oprimir la fuerza de rozamiento.

Se comprueba que, a mayor ángulo con una masa y fuerza de rozamiento constantes, el objeto se desliza con mayor rapidez y al contrario, si se disminuye el ángulo se observa como el objeto cada vez se desliza con una velocidad inferior.

Si se deja una fuerza de rozamiento constante y la masa se aumenta en grandes cantidades se puede observar que el objeto cae de forma más rápida según vamos disminuyendo esta masa, hasta que llega un punto en el cual el objeto no se mueve.

En el tercer caso, se puede observar perfectamente como el objeto siempre se desliza hacia abajo, aunque la masa sea muy pequeña, obviamente sin llegar a ser cero.

Por concluir, se realiza un experimento irreal otorgando una gran pendiente y un rozamiento alto. Se comprueba que, si la fuerza de rozamiento es superior a la fuerza ejercida por el peso, el objeto se quedará estático, pero algo está claro, no existe un rozamiento entre dos materiales que pueda alcanzar tal coeficiente de forma natural.

6.3.3.- Comparación

Por tanto, tras observar teóricamente qué debía suceder y, después simular exactamente lo mismo en el programa; se observa que ambos casos concuerdan. Por consiguiente, el programa en estos ámbitos físicos funciona a la perfección.

6.4.- ENGRANAJES

6.4.1.- Fundamentos físicos

Debido a la gran carga teórica de este apartado, se comenzará con una explicación de los conceptos.

Los engranajes son juegos de ruedas que disponen de unos elementos salientes denominados dientes, que encajan entre sí, de manera que unas ruedas (las motrices) arrastran a las otras (las conducidas o arrastradas).

La longitud del arco entre cada diente y el hueco a continuación se denomina paso, siguiendo la fórmula:

$$p = m * \pi$$

Siendo:

p: paso

m: longitud del diente y el hueco contiguo a él.

La única condición que existe para que dos ruedas engranen y puedan acoplarse y transmitir movimiento correctamente es que posean el mismo módulo.

El módulo es la relación entre la medida de la circunferencia primitiva y el número de dientes, siendo la circunferencia la que envuelve los puntos de contacto entre los dientes, cumpliéndose:

$$\emptyset * \pi = z * p$$

$$\emptyset * \pi = z * m * \pi$$

$$\emptyset = z * m$$

Siendo:

\emptyset : longitud de la circunferencia primitiva.

z: número de dientes.

p: paso.

m: longitud del diente y el hueco contiguo a él.

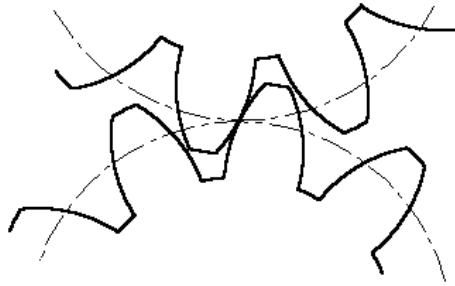


Ilustración 23 "Circunferencia primitiva"

Todos los engranajes deben cumplir la ecuación general de los engranajes que establece que la relación de velocidad angular (al ser un movimiento circular uniforme) entre los engranajes permanece constante mientras estos permanezcan engranados. Esta relación viene dada por:

$$Z_1 * w_1 = Z_2 * w_2$$

Siendo:

Z= número de dientes

w= velocidad angular

Otra forma de ver esta fórmula, aunque de forma más generalizada es con la aplicación del par motor, un momento de fuerza que ejerce un motor con sobre el eje de transmisión de potencia:

$$P = T * w$$

$$T = F * R$$

$$F_1 * R_1 = F_2 * R_2$$

Siendo:

P: potencia

T: par

w: velocidad angular

F: fuerza

R: radio

Tras esto, se puede comprobar que, si la fuerza de un engranaje disminuye, su radio deberá ser mayor para igualar al otro o viceversa, si su fuerza aumenta, el radio disminuirá. Y exactamente lo mismo con la cantidad de dientes y la velocidad angular de los engranajes.

Experimento:

Se colocan dos engranajes, uno con el doble de dientes que el otro. Se comprueba teóricamente que si el de menor número de dientes aplica una velocidad angular, el de mayor tamaño tendrá exactamente la mitad de esa velocidad.

6.4.2.- Simulación

El primer paso que se realizó fue la creación del engranaje, para esto, se utilizó un programa externo con el que era posible la creación de figuras 3D importables a Unreal Engine.

Tras esto, se le aplica un “torque”, par en la teoría, de 50 al engranaje de menor radio y se observa como el de mayor número de dientes posee una velocidad similar a la mitad del pequeño pero en dirección contraria. No es exactamente la mitad, pero esto se debe a que el programa aplica distintas fuerzas de rozamiento.

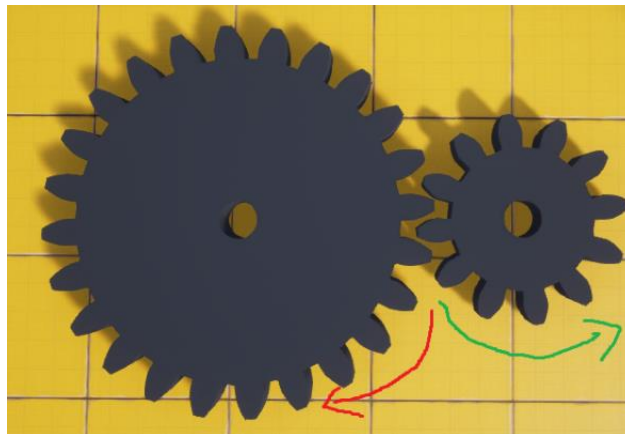


Ilustración 24 "Prueba engranajes"

6.4.3.- Comparación

Tras conocer teóricamente que, si un engranaje con el doble de dientes que otro, engrana con uno que posee el doble de velocidad, el primer engranaje tendrá una velocidad igual a la mitad de la del segundo. Y comprobar exactamente lo mismo pero en la simulación. Se puede afirmar que el programa ejecuta a la perfección la comparación de dos magnitudes que se igualan y la ejecución de velocidades angulares

6.5.- CENTRO DE MASAS Y DE GRAVEDAD

6.5.1.- Fundamentos físicos

El centro de masas representa el punto en el que suponemos que se concentra toda la masa del sistema para su estudio. Es el centro de simetría de distribución de un sistema de partículas. (19)



Ilustración 25 "Centro de masas"

Como se puede observar el centro de masas de, en este caso, una clava de malabar, se mueve de forma parabólica semejante a como lo haría una bola lanzada con la misma velocidad inicial.

Si el campo gravitatorio es uniforme, el centro de masas será igual al centro gravitatorio, el cual estudiaremos. Para esto, hay que saber que el centro de gravedad es el punto de aplicación de la resultante de todas las fuerzas de gravedad que actúan sobre las distintas masas materiales de un cuerpo, de tal forma que el momento respecto a cualquier punto de esta resultante aplicada en el centro de gravedad es el mismo que el producido por los pesos de todas las masas materiales que constituyen dicho cuerpo.

Y se intentará verificar que, a partir de las propiedades de un centro gravitatorio, un objeto simétrico, el cual posee este centro fuera de su cuerpo y por debajo de la base sobre la cual está situado, permanecerá siempre en un equilibrio estable.

6.5.2.- Simulación

Para realizar esta simulación se han creado los objetos que se ven a continuación:

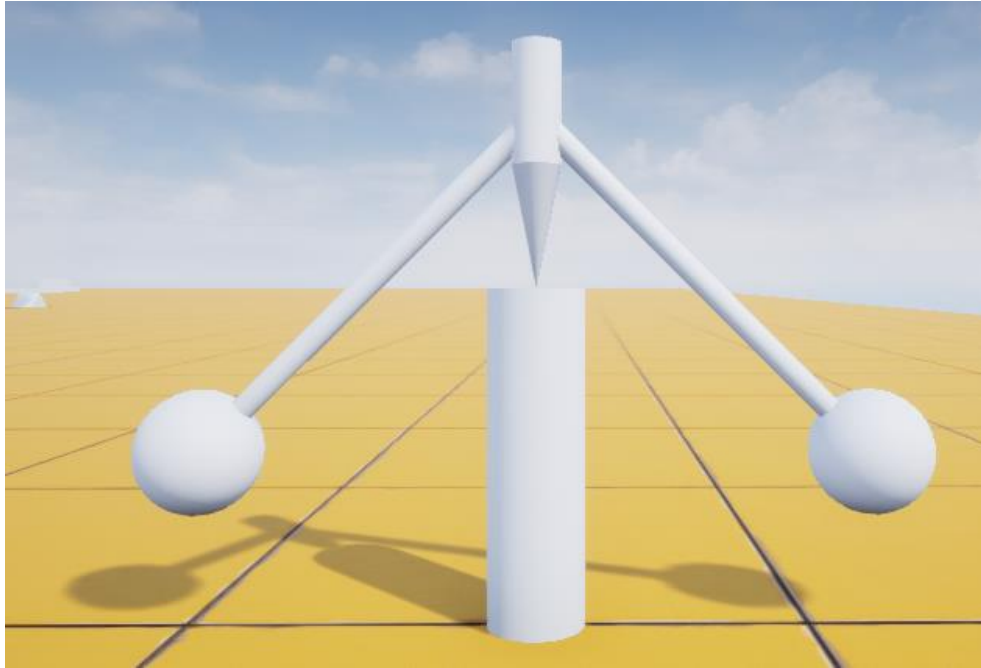


Ilustración 26 "Prueba centro de masas"

De esta forma nos aseguramos de que el centro de las masas del objeto esté por debajo de la superficie de la base creada sobre la que se apoya. En adición, el propio programa permite modificar el centro de masas, para poder simular así objetos con distintas densidades que por geometría no cumplieran que el centro de masas estuviera por debajo del punto de apoyo.

Tras esto se comienza el experimento.

Al principio, y tras colisionar el objeto con el personaje, es muy probable que lo primero que pase sea que se desplace la base y esto haga que el objeto caiga. Sin embargo, si logramos mover el objeto sin desplazarlo, o por lo menos sin que se caiga de esta, se puede observar que por mucho que se incline el objeto, este nunca se cae.

6.5.3.- Comparación

Tras conocer teóricamente que un objeto que permanezca apoyado sobre la base, con su centro de masas permanece por debajo de esta, y después de comprobar en Unreal Engine exactamente lo mismo, podemos afirmar que este experimento ha salido a la perfección, dando gran validez a las físicas utilizadas en el programa para poder realizar este experimento.

7.- CONCLUSIONES

Los motores están presentes prácticamente en casi cualquier programa informático que necesite una mejor elaboración. Es por esto por lo que conocer el estado en el que se encuentran actualmente, nos ayudará a entender que sucederá muy probablemente dentro de unos años.

Durante este trabajo de investigación he podido conocer que es un motor de videojuegos y cuál es su estado actual desarrollo centrándome de forma concreta en *Unreal Engine*. Tras realizar diversos experimentos, puedo afirmar que el programa funciona correctamente, inclusive, ofrece la posibilidad de añadir diferentes aspectos como la fuerza de rozamiento del aire, que lo hace más completo.

Creo que sería posible añadir el uso de estas tecnologías en la educación para que los alumnos pudieran comprobar visualmente sus ejercicios y así favorecer la comprensión de los mismos; de hecho, la propia compañía *Epic Games*, posee un apartado en su página para que profesores y estudiantes utilicen su programa.

Con gran certeza puedo afirmar, que al igual que lo ha hecho durante estos años atrás, esta industria evolucionará a pasos agigantados y, probablemente, las limitaciones que se pudieron encontrar durante este trabajo hayan desaparecido y surgirán otras nuevas más desafiantes.

Creo que el primer paso que se dará será el de afianzar la realidad virtual de aquí a unos años, y que no suceda como la gran multitud de innovaciones tecnológicas que han quedado en el olvido. Tras esto, se intentará dar un paso a algo más innovador con la inclusión de una posible inmersión total, algo que de aquí a corto plazo es prácticamente inimaginable, pero todo apunta a que este es el próximo gran objetivo.

Respecto a *Unreal Engine*, se espera, por parte de la comunidad, una próxima actualización a la versión 5 junto a la nueva generación de videoconsolas, la cual está siendo usada en diferentes juegos por parte de *Epic Games*, por lo que su inclusión está cada vez más cerca. Esta versión probablemente ofrezca una mejoría respecto al apartado físico, una mayor cantidad de facilidades para usuarios principiantes y multitud de recursos para los desarrolladores más expertos.

8.- REFERENCIAS

1. Estudio Física 16. [Online]. [cited 2019 10 1. Available from:
<https://estudiofisica16.wordpress.com/2016/10/26/cinematica-magnitudes-que-describen-el-movimiento/>.
2. EcuRed. [Online]. [cited 2019 10 02. Available from:
<https://www.ecured.cu/Est%C3%A1tica>.
3. Física Landia. [Online]. [cited 2019 10 02. Available from:
<https://sites.google.com/site/ultrafisica/indice-de-contenido/dinamica-fisica>.
4. 4rsoluciones. [Online]. [cited 2019 09 26. Available from:
<https://www.4rsoluciones.com/blog/que-es-un-kit-de-desarrollo-de-software-sdk-2/>.
5. Area Tecnología. [Online]. [cited 2019 9 28. Available from:
<https://www.areatecnologia.com/sistema-binario.htm>.
6. [Online]. [cited 2019 09 16. Available from:
<https://docplayer.es/45980558-3-motores-de-juegos.html>.
7. Game It. [Online]. [cited 2019 10 22. Available from:
<https://www.gameit.es/gamedev-4-motor-de-fisicas/>.
8. Varonas N. Neoteo. [Online].; 2012 [cited 2019 09 22. Available from:
<https://www.neoteo.com/los-motores-graficos-mas-importantes-de-la-historia/>.
9. Gomar J. Profesional Review. [Online].; 2018 [cited 2019 09 26. Available from:
<https://www.profesionalreview.com/2018/10/09/que-es-nvidia-physx/>.
10. Epic Games. Unreal engine documentation. [Online]. [cited 2019 10 03. Available from: <https://docs.unrealengine.com/en-US/Engine/>.
11. Alonso R. Hardzone. [Online].; 2018 [cited 2019 09 18. Available from:
<https://hardzone.es/tarjeta-grafica/>.
12. Culturación. [Online]. [cited 2019 09 18. Available from:
<https://culturacion.com/partes-que-integran-una-placa-de-video/>.

13. Usera JDd. Hardzone. [Online].; 2018 [cited 2019 10 18. Available from:
<https://hardzone.es/2018/02/18/puertos-tarjeta-grafica-diferencias/>.
14. Cadiz JD. Tarjetasgraficasjdc blogspot. [Online]. [cited 2019 10 19. Available from:
<http://tarjetasgraficasjdc.blogspot.com/p/funcionamiento.html>.
15. Compuline. [Online].; 2017 [cited 2019 09 19. Available from:
<https://compuline.com.mx/blog/para-que-sirve-una-tarjeta-grafica/>.
16. Pardos E. BaboonLab. [Online]. [cited 2019 10 08. Available from:
<https://baboonlab.odoo.com/blog/noticias-de-marketing-inmobiliario-y-tecnologia-1/post/unreal-engine-4-el-motor-grafico-que-ofrece-realismo-al-maximo-23>.
17. Pardos E. BaboonLab. [Online]. [cited 2019 10 08. Available from:
<https://baboonlab.odoo.com/blog/noticias-de-marketing-inmobiliario-y-tecnologia-1/post/blueprints-en-unreal-engine-4-funciones-tipos-y-otras-caracteristicas-21>.
18. Mundo Virtual. [Online]. [cited 2019 10 09. Available from:
<http://mundo-virtual.com/que-es-la-realidad-virtual/>.
19. Fernández JL. Física Lab. [Online]. [cited 2019 10 05. Available from:
<https://www.fisicalab.com/apartado/centro-de-masas#contenidos>.

9.- BIBLIOGRAFÍA DE IMÁGENES

Ilustración 1 "Vector":

<https://image.jimcdn.com/app/cms/image/transf/dimension=338x10000:format=png/path/sce1fe2a743ab0a00/image/i8e7406a06dd139ba/version/1374038153/vector.png>

Ilustración 2 "Vector desplazamiento y velocidad media":

<https://www.fiscalab.com/sites/all/files/contenidos/intromov/velocidad-media.png>

Ilustración 3 "Desplazamiento y trayectoria":

<https://vignette.wikia.nocookie.net/trabajo2inf/images/e/e4/Fisica.gif/revision/latest?cb=20171014210324&path-prefix=es>

Ilustración 4 "Fuerzas":

http://www.sc.ehu.es/sbweb/fisica/dinamica/rozamiento/plano_inclinado/inclinado2.gif

Ilustración 5 "Suma de fuerzas":

http://aulas.uruguayeduca.edu.uy/pluginfile.php/101102/mod_resource/content/1/imagen1Act7.png

Ilustración 6 "Resta de fuerzas":

<https://www.escolares.net/wp-content/uploads/img5.jpg>

Ilustración 7 "Fuerzas perpendiculares":

http://www.alonsoformula.com/fqeso/images/fuerzas_descomposicion.gif

Ilustración 8 "Descomposición de fuerzas":

<https://www.fisicapractica.com/imagenes/estatica-dinamica/composicion-y-descomposicion-2.jpg>

Ilustración 9 "Comparación de realidad con CGI":

<https://www.scienceabc.com/wp-content/uploads/2016/05/Real-1.jpg>

Ilustración 10 "Pong":

<https://i.ytimg.com/vi/ncB0ov5hT48/hqdefault.jpg>

Ilustración 11 "Ultima Underworld":

https://images.gog.com/db51b907d5372c42a0de68de4bf34bdc693484356176422bb9315e9cfbba6d55_product_card_v2_mobile_slider_639.jpg

Ilustración 12 "Doom":

https://http2.mlstatic.com/juego-para-pc-doom-1-y-2-para-pc-D_NQ_NP_955713-MLV31441672554_072019-F.webp

Ilustración 13 "Half-life":

<https://steamcdn.akamaihd.net/steam/apps/50/0000000156.1920x1080.jpg?t=1568754421>

Ilustración 14 "Requisitos del sistema": elaboración propia.

Ilustración 15 "Salidas tarjeta gráfica":

<https://www.calytel.com/tarjeta-grafica-vga-asus-r7240-2gd3-l-2ddr3-dviahdmi-c2x19248342>

Ilustración 16 "Refrigeración líquida":

<https://hardzone.es/2018/03/18/componentes-refrigeracion-liquida-circuito-abierto/>

Ilustración 17 "Funcionamiento de una tarjeta gráfica":

<http://tarjetasgraficasjdc.blogspot.com/p/funcionamiento.html>

Ilustración 18 "Blueprints": elaboración propia.

Ilustración 19 "Transformación manual": elaboración propia.

Ilustración 20 "Transformación interactiva": elaboración propia.

Ilustración 21 "Características de los actores": elaboración propia.

Ilustración 22 "Lanzamiento vertical":

https://sites.google.com/site/aprendiendocinematica/_/rsrc/1468756643320/lanzamiento-vertical/lanzamiento-vertical-hacia-arriba/Captura.PNG

Ilustración 23 "Circunferencia primitiva":

http://polamalu.50webs.com/OF1/mecanica/imagenes/engra_2.gif

Ilustración 24 "Prueba engranajes": elaboración propia.

Ilustración 25 "Centro de masas":

<https://www.fisicalab.com/sites/all/files/contenidos/gravitacion/centro-masas.png>

Ilustración 26 "Prueba centro de masas": elaboración propia
