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

Centro de Estudios de Postgrado

Universidad de Jaén
Centro de Estudios de Postgrado

PERIFÉRICOS, SOPORTES Y DISPOSITIVOS DE ALMACENAMIENTO Y CARACTERÍSTICAS Y FUNCIONAMIENTO DEL SISTEMA GRÁFICO

Alumno/a: Garrido Peragón, Pedro

Tutor/a: Pedro González García y
Carmen Martínez Cruz

Dpto: Informática

Junio, 2019

Tabla de contenido

0. Resumen / abstract

Este TFM tiene el nombre de “Periféricos, soportes y dispositivos de almacenamiento. Características y funcionamiento del sistema gráfico”. Por tanto, justo vamos a hablar de estos temas, ligados a mi experiencia como programador de videojuegos.

La idea es, hacer un estudio epistemológico de todas estas secciones, y además posteriormente crear una unidad didáctica. Dicha unidad, en mi caso, versará sobre videojuegos y las funcionalidades de las tarjetas gráficas (construcción de vértices y polígonos) en OpenGl, aplicada a la asignatura “Programación multimedia y dispositivos móviles”, del ciclo superior “Desarrollo de Aplicaciones Multiplataforma”.

En el estudio epistemológico, se hará un seguimiento de los periféricos, dispositivos de almacenamiento y tarjetas gráficas, desde sus antecedentes, estado actual y tendencias futuras, todo acompañado de la bibliografía correspondiente.

Respecto al desarrollo de la unidad didáctica, la centraremos en la asignatura “Programación Multimedia y Dispositivos Móviles”, asignatura sobre la cuál, pudimos acceder a una de sus horas lectivas. Dado que en ella se dan conceptos de *OpenGl* avanzados, nuestra idea es aplicar conceptos más elementales de *OpenGL* a los videojuegos que he realizado, tales como definir un Sprite, moverlo por pantalla, usarlo, hacer gestión de tiles, etc.

This TFM has the name of "Peripherals, supports and storage devices. Characteristics and operation of the graphic system ". Therefore, we are just going to talk about these issues, linked to my experience as a video game programmer.

The idea is to do an epistemological study of all these sections, and also later to create a didactic unit. This unit, in my case, will deal with videogames and the functionalities of graphic cards (construction of vertices and polygons) in OpenGl, applied to the subject "Multimedia programming and mobile devices", of the higher cycle "Development of Multiplatform Applications".

In the epistemological study, peripherals, storage devices and graphic cards will be monitored, from their background, current status and future trends, all accompanied by the corresponding bibliography.

Regarding the development of the didactic unit, we will focus on the subject "Multimedia Programming and Mobile Devices", a subject on which we could access one of their teaching hours. Given that OpenGl concepts are advanced, our idea is to apply more elementary concepts of OpenGL to the videogames I have created, such as defining a Sprite, moving it around the screen, using it, managing tiles, etc.

1. Introducción

Para empezar, vamos a realizar un estudio epistemológico de los periféricos, soportes y dispositivos de almacenamiento y características y funcionamiento del sistema gráfico. Para los 3 apartados, haremos un estudio de sus antecedentes, presente y futuro.

Indicamos la normativa del módulo que vamos a seguir.

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.

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.

En la asignatura "Programación Multimedia", como hemos comentado, se tratan conceptos programación de videojuegos en 2D y 3D, su aplicación a los ordenadores y también los móviles, se ven conceptos de OpenGL, como definir mallas de polígonos y darles color, forma, textura.

En los videojuegos que he programado, todos estos conceptos (que son avanzados) aparecen en algunas secciones del código, pero sobre todo, hacen hincapié en la manipulación de sprites y tiles, que no dejan de ser fotografías.

Sin embargo, gracias a que para desarrollar un juego en 2D no se requiere de mucho código, a los alumnos se les proporcionará uno que deberán editar y así crear un videojuego final, en concreto, el "Comecocos".

Una vez que les haya sido explicado la manera de programar otro videojuego que se le parece mucho (el *Bubble Burst*), los alumnos serán capaces de hacer el que se le ha pedido.

Esto será una puesta en contacto de los alumnos con los videojuegos y gráficos, les serán enseñados los conceptos de programación y teoría básicos, y ya ellos, podrán decidir si continuar su camino por aquí o centrarlos en otras disciplinas.

Sin más, pasamos a continuación a nuestro estudio epistemológico.

A. ESTUDIO ESPISTEMOLÓGICO

1. Introducción

En este TFM y en concreto, este estudio epistemológico, voy a tratar el siguiente contenido:

- Periféricos.
- Soportes.
- Dispositivos de almacenamiento.
- Características y funcionamiento del sistema gráfico.

Dado que mi especialidad es la programación de gráficos (videojuegos), a lo largo del estudio epistemológico y del TFM, haré la primera parte respecto a periféricos y soportes y dispositivos de almacenamiento. Sin embargo, el apartado donde más me extenderé, tanto en antecedentes, estado actual de la cuestión y tendencias futuras, será de las tarjetas gráficas y cómo se puede programar en ellas (relacionándolas con los videojuegos). Debe ser un punto de partida, para finalmente introducir en la unidad didáctica estas tarjetas gráficas, su programación (en OpenGL) y el vínculo que tiene respecto a los videojuegos que he programado estos años.

Dicha extensión es así, porque las tarjetas gráficas están muy ligadas a los videojuegos. Antiguamente, cuando alguien compraba una tarjeta gráfica puntera, a la hora de ejecutar videojuegos, se liberaba al procesador de la carga que consistía en crear los gráficos, etc. Y los programadores, podían hacer uso de las librerías de las tarjetas gráficas para programar sus videojuegos. Las consolas, por ejemplo, se les proveía de un sistema gráfico muy avanzado, que se iría quedando obsoleto conforme pasaran los años.

Dado que las tarjetas gráficas nos permiten trabajar con *sprites*, *tiles*, polígonos y vértices, extenderemos la explicación de las mismas, y veremos posteriormente su relación con los videojuegos que he programado.

2. Antecedentes

Procedemos, en este punto, a hacer un estudio sobre periféricos, soportes y dispositivos de almacenamiento y características y funcionamiento del sistema gráfico.

2.1 Periféricos

Empezaremos hablando de los periféricos. ¿Por qué son necesarios y cómo surgieron?

Antiguamente, prácticamente el único periférico que tenían los ordenadores era el monitor. Como había máquinas de escribir, se usaron las mismas para definir los teclados que conocemos hoy en día. ¿Pero eran suficientes periféricos? Claramente

no, por poner un ejemplo, el ratón se fue introduciendo poco a poco, y como todas las ideas novedosas, en un principio fue rechazado. Procedemos ahora a la definición de periférico.

Se considera periférico, a las unidades o los dispositivos a través de los cuáles el ordenador es capaz de comunicarse con el mundo exterior, (Arnau, 2011). También se les suele llamar a los sistemas que o bien almacenan y guardan información, como a los que sirve de memoria auxiliar a la memoria principal. Es decir, estos periféricos no pertenecen al grupo fundamental de un ordenador (considerando que es CPU más Memoria central), pero puede realizar operaciones de entrada y salida, ayudando y complementando al proceso de datos del que se encarga de realizar la CPU, como comenta en su libro (Todorov, 2013).

Es evidente, pues, que estos dispositivos surgieron por la necesidad de aumentar la memoria y la funcionalidad de un ordenador, sin tener que estar comprando y cambiando su CPU, procesador y unidades de memoria. Estos periféricos ayudan a potenciar la funcionalidad de un ordenador.

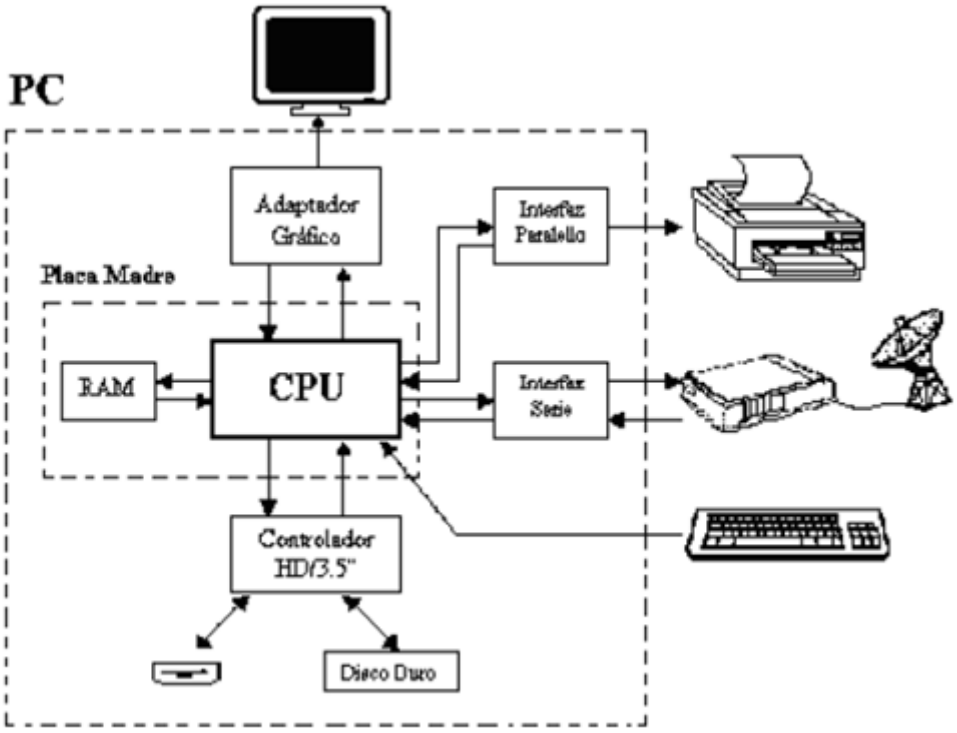


Figura 1 : Principales periféricos.[1]

El procesador y la memoria principal intercambian mensajes entre sí. Dicha memoria, alberga las siguientes propiedades:

1. Su capacidad para almacenar datos es bastante reducida.

[1] Fuente: Apuntes de Vicente Arnau Llombart.

2. Una vez que apagamos el ordenador, desaparece. Es decir, no se guarda (es volátil).

Es por este motivo, que surgieron los periféricos, (Arnau, 2011). Los periféricos pueden ser unidades de entrada, salida o memoria masiva auxiliar.

Los más comunes son:

Entrada: monitor, *keyboard*, *mouse*, dispositivos de captura directa de datos (lectora de banda magnética, detectores ópticos (cámaras digitales, caracteres impresos, barras impresas, escáneres, de marcas, etc.), detectores de caracteres impresos (OCR), unidades de reconocimiento de voz, lápiz óptico, pantallas sensibles al tacto, joystick, tableta gráfica.

Salida: monitores de visualización (tubo de electrones y Tft), impresoras, visualizadores, sintonizador de voz, registrador gráfico (plotter).

Memoria masiva auxiliar: discos y cintas magnéticas, discos ópticos y magnetoópticos, pendrives.

Mixtas: terminal teletipo, pantalla sensible al tacto y terminal interactivo teclado pantalla.

Las memorias son diferentes respecto a los demás tipos de unidades. La información que en ella guardan, normalmente está codificada, y de esta manera, los usuarios no la comprenden. Por tanto, el intercambio de mensajes entre el usuario y la memoria es distinta de los demás periféricos de entrada y salida.

Los dispositivos de entrada/salida, se encargan de transformar la información externa en impulsos electrónicos, que una vez codificados se envían al procesador para que sean interpretados, procesados y almacenados de forma automática. Normalmente, las señales se codifican en código ASCII Y la CPU la recibe en formato binario.

Con otros dispositivos, ocurre justamente lo contrario. La información almacenada en bits unos y ceros, se transforman en un conjunto de marcas visibles en el papel, con el objetivo de que el usuario pueda leer y comprobar la información que está escrita y verificar si es correcta.

Figura 2. Algunos periféricos.[2]

2.2 Sistemas de almacenamiento

En cuanto a "soportes de información y dispositivos de almacenamiento" se refiere (segundo punto a tratar en nuestro TFM), hablamos de aquellos dispositivos, (Tenología Informática, 2018) que nos permiten almacenar información y que, en principio, no son fáciles de ser llevados de un lugar a otro. En el libro de (Poulton, 2014) se hace un estudio exhaustivo de todos ellos, a lo largo de 602 páginas.

Hay muchos soportes que pueden almacenar información, y que a medida que era más más necesitamos almacenarla, fueron surgiendo nuevos soportes, y todos ellos han estado presentes en los últimos años.

Los dispositivos que más se han usado a lo largo del tiempo (algunos ya obsoletos) son estos:

- Almacenamiento con cintas magnéticas (discos duros y disquetes).
- Almacenamiento óptico (CD, DVD, *Blu-Ray*).
- Almacenamiento electrónico (tarjetas MicroSD y lápices USB extraíbles).

Al igual que ocurrió con los diskettes, como comenta (Dona, 2011) a medida que sus capacidades de almacenamiento se van menguando (en estos dispositivos) suelen caer

[2] Fuente: https://tecnologia-facil.com/wp-content/uploads/2015/10/comprar-perifericos-pc-1.jpg

en desuso. Así pues, escribimos brevemente algunos de ellos, si bien es cierto que todo esto lo explicaremos a fondo posteriormente.

• Discos duros internos y discos duros externos.
• Copias de seguridad mediante cintas y discos
• *Pendrives* (o vulgarmente llamado USB).
• Tarjetas de memoria (SD, microSD, etc.
• *CDs* o *DVDs* (discos ópticos).

En el libro de (Dona, 2011) podemos encontrar información detallada y específica de estos dispositivos.

Nos ceñiremos a algunos de los siguientes conceptos:

- Discos duros, tanto externos como internos, interfaces de usuario, manejar la BIOS, recrearnos con los disquetes y dispositivos más ancianos, y las tarjetas SD (también MicroSD).

2.3 Tarjetas gráficas

Para terminar el apartado de los antecedentes, en este TFM incluiremos todo lo relativo a "Características y funcionamiento del sistema gráfico". Brevemente, una tarjeta gráfica lleva al monitor la información gráfica que debe ser presentado por el mismo.

Dado que mi especialidad son los videojuegos, en esta sección aplicaremos en conjunto tanto las tarjetas gráficas como los videojuegos, y las relaciones entre ambos, como he comentado en la introducción.

Por último ¿para qué surgieron las tarjetas gráficas? Antiguamente, los ordenadores trabajaban mediante tarjetas perforadas (todas las entradas y salidas se valían de las mismas), más teclado e impresoras primitivas. Un día se pensó que lo ideal era añadir un monitor a la información que manejaba el ordenador. De esta forma, se tendría una forma de hacer "*Debug*" a todos los programas. Estas televisiones pasaron a ser los monitores, y para obtener la información que se debía pintar en ellos, apareció el hardware específico que también trataremos en este TFM, las tarjetas de vídeo.

Las tarjetas gráficas, (Raya, 2012) y las GPUs (*Graphic Proccesing Unit*) son la piedra angular a la hora de realizar programas basados en programación gráfica, como pueden ser la programación de videojuegos, o cualquier profesión a la que esté atribuída usar gráficos para modular entornos. El objetivo por el que fueron creadas es otorgar más velocidad en los cálculos, ofrecer realismo para las películas gráficas de

animación (algo que ya casi se consigue en las videoconsolas con la *Playstation* 4 Pro, por ejemplo), de tal forma que liberan al procesador central (CPU) para que pueda realizar otros cálculos.

Aunque estamos bastante acostumbrados a hablar de tarjetas gráficas y la guerra entre ellas (Nvidia VS Ati), la realidad es que poco conocemos de su funcionamiento (diseño a nivel de arquitectura). Por eso en esta memoria vamos a tratar de hacer una introducción a ellas y a posteriori destacar su funcionamiento y propiedades.

Las tarjetas gráficas son en sí una unidad que expande la funcionalidad de nuestro ordenador. En sí, el ordenador dispone de un procesador, pero al instalarle una tarjeta gráfica, lo libera de trabajo, y esta tarjeta permitirá transformar los datos que son impulsos eléctricos en código, que será utilizado y permitirá ver en el monitor lo que en el código se le indica que dibuje. Están dividas en:

- La GPU (*Graphic Proccesing Unit*)
- *Video memory*: es un tipo de memoria que ha avanzado muy rápido en los últimos años y que permite que la tarjeta pueda hacerse cargo de todo el contenido que le manda a la CPU el sistema. Su tamaño es variable, y es más grande o pequeña según el tipo de tarjeta. Los últimos avances tratan memoria DDR, destacando DDR2,
- RAMDAC: es un transformador de señales. En nuestro caso, se encarga de convertir la señal digital que tienen los PCs en analógica, que el monitor podrá entender.
- Disipador: Es un aparato que está hecho de un material conductor, y que se usa para evitar que las tarjetas se calienten demasiado. Si es en sí eficiente o no, dependerá de la superficie y de la estructura total. Normalmente, suelen ser muy grandes.
- Ventilador: es un aparato que sirve para mover el aire cercano y que intenta evitar que se caliente la tarjeta. Suele hacer ruido al tener partes móviles y es menos eficiente que un disipador.
- Alimentación: hasta hace unos años, la alimentación no había supuesto un problema para las tarjetas gráficas pero, las tarjetas gráficas cada vez consumen más energía. Y a pesar de que las fuentes de alimentación cada vez son más grandes, la velocidad del puerto PCI está limitada porque tiene que dar energía a cada sección. Entonces, en este tipo de tarjetas gráficas, se conectan de una forma directa la tarjeta y la fuente de alimentación y así no es necesario dar un rodeo y pasar obligatoriamente por la placa base.

Por tanto, como hemos visto en el desglose anterior, que las tarjetas gráficas tengan alimentación individual es algo bastante normal (y en modelos de tarjetas actuales).

En el siguiente punto, describiremos a fondo las características, funcionamiento y componentes del sistema gráfico.

4. Estado actual

En este apartado vamos a tratar la actualidad de nuestros elementos de estudio, primero de los periféricos, después de los sistemas de almacenamiento y por último las tarjetas gráficas.

4.1 Actualidad en periféricos

Como hemos comentado antes, los periféricos han ido evolucionando a lo largo del tiempo. En esta sección, veremos sus tendencias más actuales.

Secciones de los periféricos.

Los periféricos están divididos en dos secciones que albergan una gran diferencia entre ellas.

- La sección mecánica está compuesta por diferentes dispositivos electromecánicos, que están siempre controlados por elementos electrónicos. Algunos de ellos son: conmutadores manuales, electroimanes, motores, etc.

- La otra sección, la electrónica, gestiona todas las órdenes que vienen a la CPU, bien para la transmisión, bien para la recepción de información, y crea las señales de control para manejar adecuadamente la parte del periférico que es mecánica.

  En este apartado, es bastante normal el uso de dispositivos optoelectrónicos, que generan y detectan la información de entrada y salida. Por otro lado, son usados igualmente como detector de posición de los elementos que hemos comentado antes, mecánicos, que están incluidos en el periférico. Dicho cometido en los conversores analógico y digital es importantísimo.

Los dispositivos de entrada y salida tienen como cometido el transformar la información que llega del exterior en señales codificadas, permitiendo que sea haga realidad la detección, procesamiento, transmisión y almacenamiento de una forma asequible y automática.

Conexión de los periféricos al ordenador

Los periféricos son capaces de conectarse al ordenador gracias a unos hilos llamados buses. Pueden ser de diverso tipo, entre ellos:

- *Bus general:* transmite los datos al ordenador en paralelo.
- *Bus local:* se encarga de interconectar la CPU con otras secciones del ordenador. Es muy rápido y entre sus funciones está conectar dicha CPU con las tarjetas de la *motherboard* aparte de los controladores de dispositivo externo.
- *Bus del sistema:* conecta tanto los mencionados controladores, o los periféricos o bien las tarjetas de la placa base. Algunas veces además conectan ampliaciones de memoria.

Pero hay un problema. Algunas veces es necesario adaptar la velocidad de transferencia de los periféricos, la naturaleza de las señales de control, los niveles de tensión y otros requisitos. Para ellos existen los *buses de expansión* o *buses de entrada y salida.* En los ordenadores grandes (*PCs* de sobremesa), como tienen muchos tipos de buses, se necesitan estos adaptadores, para que todo funcione correctamente.

Los procesadores, en estos casos, tienen unas ranuras de expansión en la placa base (*motherboard*), y gracias a ellas se conectan a los buses del sistema y permiten dar soporte a distintos dispositivos a la CPU, como aceleradoras gráficas con FPGA, tarjetas digitalizadoras, etc.

Partiendo de esta base, todos los buses siguen unos estándares comunes:

- Protocolos de intercambio de información
- Estudio de la velocidad y el tempo del paso de información Anchuras de los sub buses.
- Sistema de conexión, que es físico.

Con la explicación de los buses, terminamos esta sección. Pasamos a estudiar el siguiente punto de nuestro TFM.

4.2 Soportes y dispositivos de almacenamiento

Hoy en día, como vemos en este artículo escrito por (Delgado., 2014), prácticamente la totalidad de los ordenadores que tenemos en casa tienen dos dispositivos de guardado de información por norma, los discos duros (internos y externos) y las memorias SD.

. Los disquetes y las disqueteras han ya desaparecido, y sólo se usan en ordenadores retro donde aún se usan como soporte de videojuegos, programas, etc. (ejemplo el ordenador Amiga 500 que utilizaba disquetes, ya estando en su época obsoleto). Y antes de esto, los ordenadores carecían de disco duro, y para cagar el sistema operativo se usaban disquetes, lo cuál hacía el proceso de arranque muy pesado. Las grabadoras de *DVD*, aunque se siguen incluyendo en los ordenadores, han quedado desfasadas, en el sentido de que cuando queremos guardar algo, ya no lo hacemos en *DVDs* o *CDs*, sino en tarjetas de memoria o discos duros externos.

Figura 3. Evolución de los sistemas de almacenamiento.[3]

Por tanto, los HDD (*Hard Disk Drive*), son el sistema base más importante de un ordenador que nos permite el guardado de la información. No obstante, no debemos olvidar a las memorias SD que cada vez pueden acumular gran cantidad de datos y son muy rápidas.

También formando parte de la memoria tenemos la memoria RAM, que es volátil y una vez que se apaga el ordenador se pierde toda la información que se había almacenado. Además, no suele ser muy rápida y sus accesos a memoria principal son lentos.

Partiendo de esta base, procederemos a explicar los principales tipos de almacenamiento de datos, hablando de manera general y no entrando en detalles. Queda a disposición del lector profundizar más sobre el tema que considere oportuno. Estos sistemas han sido clave en que la tecnología haya podido seguir avanzando. Ningún proyecto, grande o pequeño, hubiera podido tener éxito si en esta parte de la informática no se hubiese intentado mejorar.

[3]Fuente:https://image.slidesharecdn.com/ejercicio01-josmara3-150701112014-lva1-app6892/95/evolucin-sistemas-de-almacenamiento-1-638.jpg?cb=1435749725

Guardado de información gracias a cintas y dispositivos magnéticos.

Son los dispositivos más antiguos y desde siempre se han utilizado mundialmente. Ellos permiten guardar muchísima información en aparatos que suelen tener un volumen pequeño. Funcionan gracias a la utilización de dipolos magnéticos, que suelen estar ubicados en su superficie.

Figura 4: dispositivos de almacenamiento magnético[4]

Los que más repercusión han tenido son los discos duros externos.

Estos, como hemos comentado antes, son los dispositivos más utilizados, principalmente para guardar grandes cantidades de información y que puedan ser usados en otros ordenadores.

[4]Fuente:https://userscontent2.emaze.com/images/f553816c-d124-49fb-8249-3fcd7c70af8a/991f3efbc40fffc012e65256bba46f45.jpg

A la misma vez, tenemos la existencia de los *HDD* internos, instalados dentro de los ordenadores, y a diferencia de los externos, su capacidad de almacenamiento es mayor. Hemos de decir también que los *HDD* externos pueden tener más tendencia a romperse, pues los conectamos y desconectamos constantemente.

Por tanto, a pesar de llevar tantos años de existencia, los dispositivos magnéticos son los más usados, porque además, como hemos dicho, proporcionan un gran espacio en forma de Gigas.

Para poder escribir y leer en los soportes de almacenamiento magnético, se utilizan las partículas, que, presentes en su superficie, son magnéticas.

En lo que concierne a escribir, se crea un campo magnético, gracias al cabezal de lectura y escritura, que da propiedades de magnetización a las partículas magnéticas, obteniendo dígitos binarios según la polaridad que se utiliza.

Por otro lado, en lo que concierte a leer, se crea otro campo magnético, gracias de nuevo al mismo cabezal, y en cuanto entra en contacto con las partículas magnéticas, o bien atrae o bien repele dicho campo magnético. Así, podemos discernir si el polo está cargado positiva o negativamente.

El ejemplo de tecnología obsoleta de medio magnético por antonomasia es el disquete, que dio soporte durante muchos años a usuarios que querían guardar sus datos (videouegos, etc.) De todas formas, no olvidemos a la comunidad retro que hoy en día sigue usando disquetes para, por ejemplo, jugar a videojuegos de PC, Amiga, etc. (aunque encontrar disquetes en las tiendas sea ya imposible, pero se puede recurrir a internet). Aunque sea un tema transversal, comentar como anécdota que antiguamente, se pedía en los manuales de instrucciones que los usuarios que habían comprado un videojuego en disquetes, se hicieran una copia de seguridad del mismo.

Guardado por medio óptico.

Su mayor finalidad es guardar archivos multimedia, como vídeos, fotos y música. También podemos guardar películas, documentales, videojuegos, *comics*, *CDs* de música. Para proceder a su grabado, se utiliza un rayo láser de muy alta precisión.

A continuación ponemos algunos ejemplos: *CD*, *DVD* y *Blu-Ray* aparte de sus grabadoras y lectoras correspondientes de *CD-ROM, CD-RW, DVD-ROM y DVD-RW*. Estos soportes que hemos comentado, pueden almacenar grandes cantidades de datos. Se suelen utilizar bastante, sobre todo en los reproductores *DVD* y *Blu-Ray*, radios, computadoras, videojuegos. Este formato es muy accesible. Por poder, puede ser encontrado en papelerías,

supermercados, y por supuesto en tiendas de informática, de fotografía y demás contenido afín.

Figura 5: dispositivos de almacenamiento por medio óptico

[5]

La lectura en estos sistemas, se realiza como hemos dicho en el primer párrafo, usando un rayo láser de alta precisión. Este láser lo proyectamos en su superficie, y accederá a la información contenida en el disco. El láser, por otro lado, será desviado a distintas zonas del *CD* o *DVD*, leyendo los datos. Dichos datos estarán guardados en formato binario.

Por otro lado, para hacer uso de *CDs y DVDs* regrabables, necesitamos de un material específico, que está pegado a la capa del dispositivo y permite, que su superficie sea quemada, creando así información nueva. Creará, por tanto, los surcos que posteriormente servirán para identificar los dígitos en binario.

Guardado por medio electrónico

Es una forma muy prometedora, además de joven de guardado de información. Recurre a utilizar circuitos electrónicos para guardar dicha información. Estos circuitos, no tienen que moverse obligatoriamente. Dentro de estos dispositivos, encontramos los *pens usb* y las memorias SD, muy utilizadas actualmente. Ya que es muy fácil manejarlos, estos dispositivos fueron creciendo rápidamente en el mercado de ventas.

[5]Fuente:https://tecnologia-informatica.com/wp-content/uploads/2018/04/dispositivos-almacenamiento-informacion-5.jpg

Figura 6: Dispositivos de almacenamiento electrónico.[6]

No obstante, no son capaces de guardar grandes cantidades de datos e información. Debido a lo intrínseco a esta tecnología, de momento no es posible que almacenen muchos Gigas, aunque cada vez este tamaño va creciendo. Por otro lado, son pequeñas y portables y cada vez se usan más en videoconsolas (Nintendo 3DS, *Ps Vita*, *Switch*), cámaras de fotos, teléfonos móviles, etc. Vulgarmente son llamadas memorias *SSD* del inglés *Solid State Drive*.

Es posible el uso de estos dispositivos, y que se puedan grabar en ellos datos, gracias a los materiales con los que están construidos, altamente conductores, y que son usados a la hora de realizar los chips que almacenan dicha información.

Por tanto, cada bit (dígito binario) puede ser almacenado en estos dispositivos, en los que existen dos puertas que están realizadas usando material semiconductor. Reciben el nombre de la puerta flotante y la puerta de control). Entre ellas, tiene lugar una pequeña capa de óxido, que si tiene carga negativa (electrones), significa que el bit es 1, y si está descargada pone de manifiesto el bit 0.

De esta manera, este método de almacenamiento es igual que las usada en las memorias RAM de tipo dinámica, con la diferencia de que estas últimas no pueden retener la información durante mucho tiempo, por tanto, las tarjetas SSD no son consideradas memoria RAM en sí.

Para terminar con este apartado, y esta sección, resumimos que los dispositivos de almacenamiento magnético poseen un tiempo de acceso mucho mayor que los dispositivos de medio electrónico, pues contienen partes móviles. En cambio, el principal problema de la tecnología electrónica es su precio. Si buscamos almacenamiento sencillo, por ejemplo 4 u 8 megas, no nos sale cara una tarjeta, pero si queremos comprar de 64 o 128 bits, el precio se

[6]Fuente:http://4.bp.blogspot.com/-jL945ApSFeE/VKm0BFFW6II/AAAAAAAAAD8/6aB7Y763XRg/s1600/memorias-usb-chollos.jpg

dispara. Sumado eso a que no pueden guardar gran cantidad de información, conforman estos dos puntos el único "pero" a estos dispositivos.

4.3 Características y funcionamiento del sistema gráfico

¿Qué es una GPU?

De la misma manera que el ordenador tiene un cerebro, que es la CPU (que se encarga de todo el control del ordenador), las tarjetas gráficas poseen su propia unidad especial, que se llama GPU.

Esta *Graphic Processing Unit* (*GPU*) se encarga de liberar de trabajo a la *CPU*, ya que está desarrollada exclusivamente para ser utilizado por aplicaciones gráficas. De esta forma, mientras la CPU se encarga de cálculos como la IA, motores de física o aplicaciones normales, la GPU se utiliza para videojuegos, aplicaciones que hacen uso exclusivo de gráficos. Así, por un lado tenemos las operaciones de las que se encarga la *CPU*, y por otro lado, las de la *GPU*.

Las GPUs modernas, como trata (Rodríguez, 2010) en sus apuntes de la Universidad de Gran Canaria, tienen una estructura paralela que es más efectiva que un procesador normal, y está diseñada especialmente para el tratamiento de gráficos tales como manejo de matrices, dibujado de polígonos, transformación de vértices en sistemas de coordenadas, etc.

Las desarrolladoras de tarjetas gráficas son *ATI* y Nvidia. Existe una gran lucha entre ellas por hacerse con el mercado de dichas tarjetas gráficas. Aquí, veremos parte de esa lucha en la que llevan rivalizando más de 10 años (o incluso más).

Por otro lado, la arquitectura de las tarjetas gráficas, es muy diferente de las de la CPU. En las primeras, dicha arquitectura recibe el nombre de *streaming*, mientras que en la CPU tiene el nombre de *Von Neumann*. El objetivo de estas últimas es conseguir velocidades grandes, pero tienen a veces un pero, como las transimisiones con la ALU (Aritmethic Logic Unit) y las unidades en coma flotante (*FPU*) que reducen su velocidad (pues hay demasiados accesos a memoria.

Las *GPUs*, en cambio, tienen una estructura que hace que el gasto energético empleadas en ella para las transmisiones y demás operaciones, sea reducido, y en

cambio, la CPU se beneficia de esta circunstancia y aumenta su velocidad, y pueden, dedicarse a tareas como las que trabajan en conjunto con la *ALU*.

En la actualidad, las tarjetas gráficas han llegado a tal grado de desarrollo, que se consideran que algunas pueden ser incluso tres veces más rápidas que los procesadores generales. Así, como hemos visto antes, se dispone de velocidad "extra" que aprovecha la CPU para realizar sus otros cálculos (en coma flotante, *ALU*, inteligencia artificial, etc.) Hablando claro, del caso de aplicaciones que la necesitan.

Este auge se debe a que el hardware gráfico no suele ser muy caro, pues está la guerra y el auge de los videojuegos, que presionando en el mercado, hace que los jugadores compren continuamente consolas y videojuegos, potenciando la venta de tarjetas gráficas. Como están continuamente en evolución, tanto consolas como ordenadores, estas tarjetas no caen en desuso y se siguen vendiendo en grandes cantidades.

Estas tarjetas gráficas, traen su propio lenguaje de programación, y hoy en día, dichos lenguajes pueden compararse con los lenguajes de la CPU, como C, Java, C++ etc., ya que son de alto nivel.

El procesador gráfico en su interior.

La GPU tiene la característica de que está segmentada. De esta manera, puede trabajar haciendo diversos cálculos en paralelo. Así, es capaz de trabajar a altas velocidades y tiene una gran capacidad de procesado, ya que los programadores pueden hacer uso de estos códigos de forma inteligente y optimizándolos para que la ejecución sea la más alta posible.

Sin embargo, hay que diferenciar entre lo siguiente. Mientras que en una *CPU* los núcleos pueden trabajar de manera independiente, dichos núcleos en la *GPU* son linealmente dependientes unos de los demás. No obstante, cuando hablamos de paralelismo en ordenadores actuales, esta es una funcionalidad que sólo tiene la *GPU*.

Los procesadores, respecto de los que hemos hablado, como podemos comprobar en la publicación de (Raya, 2012) se diferencian en 2:

- Los especializados en procesar vértices.
- Los especializados en procesar píxeles.

Por tanto, anotamos como principales componentes que maneja una CPU son los vértices y los píxeles (conceptos fundamentales cuando hablamos de tarjetas gráficas).

Como veremos en la foto de *Diddy Kong* situada en la parte inferior, las tarjetas gráficas primero añaden un esqueleto, que está compuesto por vértices y polígonos. Más tarde, añaden relleno, color (haciendo uso de un bit interno) y hasta textura. Esto es algo que se hace desde la época de los *Silicon Graphics* cuando fue programado el videojuego *Donkey Kong Country*. Por aquella época, hicieron uso del ordenador *Challenge*, que eran como 20 superordenadores procesando conjuntamente. Hoy en día, con un ordenador más o menos potente, podemos obtener los mismos resultados.

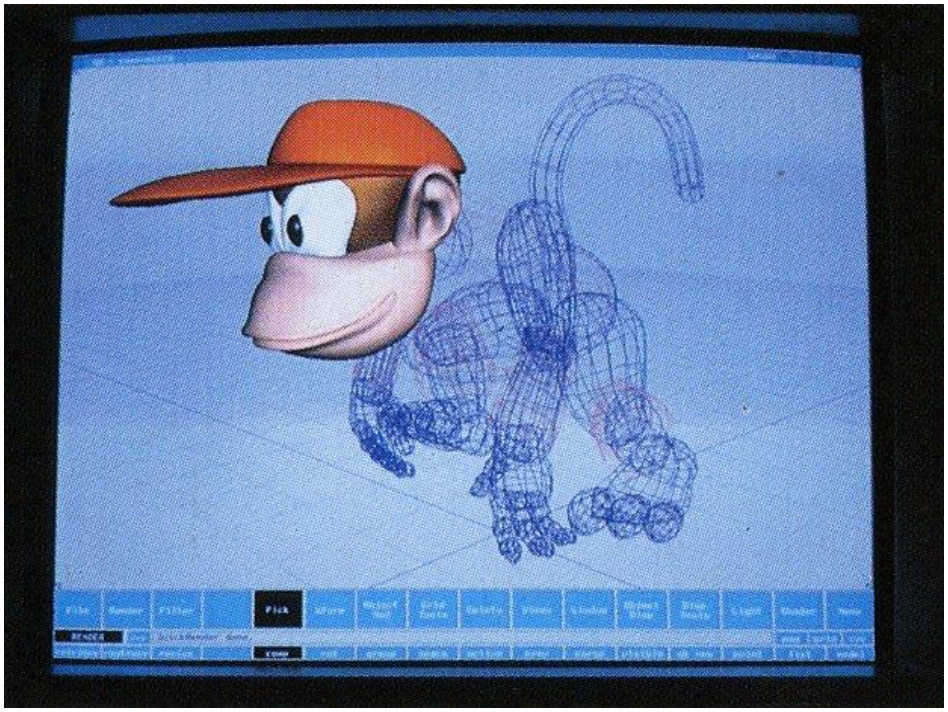


Figura 7: Ejemplo antiguo de Diddy Kong hecho con polígonos.[7]

Procesador de vértices

Un vértice es un concepto de geometría, partiendo de la base de dibujar un triángulo, un triángulo tiene 3 vértices. Si según la posición en el espacio en la que están, vamos dibujando más y más triángulos, y polígonos de diferentes lados y vértices, podremos

[7] Fuente: https://pbs.twimg.com/media/DvOLf96VYAAcIvh.jpg

crear estructuras como la mostrada en la *figura 7*. Todo esto es realizado por el procesador de vértices.

Una vez ya tenemos definido nuestro esqueleto, podemos hacer rotaciones y traslaciones sobre él. Estas son operaciones básicas en programación 3D, y también pueden ser útiles en 2D (por ejemplo, si hacemos uso de traslaciones, podemos simular la cámara de un videojuego, que se desplazará a izquierda o derecha según el personaje se mueva). Hoy en día las tarjetas gráficas pueden mover y programar un número determinado de vértices. Cuantos más "*vertex shaders"* tengamos, más potencia albergará.

Antes de la introducción de los *vertex shaders* (a partir del año 2000/2001), siempre se programaba de la misma manera, con unas pautas muy fijas y no se conseguían muy buenos resultados. Sin embargo, con la introducción de los últimos y el cambio de modelado en la tarjeta gráfica por parte de los procesadores de vértices, el programa se transforma en un microcódigo, que tiene la habilidad de realizar operaciones tales como *morphing* o *waves*. Así, se pueden conseguir gráficos que son usados en videojuegos de última generación y películas.

Rasterizado

Como hace hincapié (Raya, 2012) una vez manipulados los vértices, se realizan las etapas de *clipping*, es decir, que los polígonos que no estén en el punto de mira de la cámara, sean ocultados, y también se dejan de pintar y procesar los triángulos o polígonos que no estén en la parte visible del gráfico (técnica llamada *culling*).

Todo esto tiene lugar en la etapa de rasterización. Así, como hemos visto antes, al esqueleto de vértices y polígonos, se le añade forma, color (bit interno), textura, profundidad, y nos queda un resultado de lo más realista y espectacular.

Comentar también, que estas ideas llevaban ya tiempo en el mercado antes de los años 2000, siendo el videojuego *Alone in the Dark* el pionero en el uso de estas técnicas (*clipping* y *culling)*, más renderizado de escenario con el que interactuaba nuestro personaje hecho con polígonos.

Figura 8: Videojuego Alone in the Dark (1992, Infogrames).[8]

Procesado de fragmentos

Tras la etapa de *rasterización* llega la del *procesado de fragmentos*. Todos los datos almacenados sobre lo que se hace en la etapa de rasterización (malla, vértices, triángulos, color, textura, puntos) son lo que se llama *procesado de fragmentos*. Esto, está comentado de manera más técnica en la publicación de (Raya, 2012).

Por tanto aquí, se procesa todo lo que se le ha indicado anteriormente, y se hacen las transformaciones pertinentes. En esta sección ya se hacen uso de los *vertex shaders* que hemos comentado antes y generan su código (micro) asociado.

Para concluir esta sección, en esta etapa se hacen las fases de ROP y *blending*, que nos van dibujando la escena. Se pasa además al código diversos test para eliminar los triángulos y polígonos que quedan ocultos por los objetos, y que no se queden dentro de lo que la cámara nos ofrece.

Programación en la *GPU*.

[8] Fuente: https://noespaisparafrikis.com/wp-content/uploads/2017/12/Noticias-Analisis-Repaso-Alone-In-The-Dark-1.png

A lo largo del tiempo, las *GPUs* han avanzado tanto que hoy en día se puede programar en ellas ciertos efectos, aumentado el rendimiento del proceso en cuestión. Le daremos el nombre de *shaders*, a los programas creados por programadores de tarjetas gráficas que son ejecutados en la GPU. Por un lado, tendremos *shaders* para vértices y *shaders* para fragmentos. Los primeros, serán ejecutados en el procesador de vértices y los segundos, en el correspondiente de fragmentos.

Un *vertex shader* es una función que recibe como parámetro un vértice, y sólo trabaja con un vértice a la misma vez. Sin embargo, puede transformarlo y modificar las características del mismo, que suelen repercutir en su geometría. De esta forma, se pueden conseguir efectos gráficos tales como deformar en tiempo real un objeto (movimiento en vaivén de una ola). En el videojuego *Xenogears* se hace uso de ese ejemplo, y podemos verlo en esta foto.

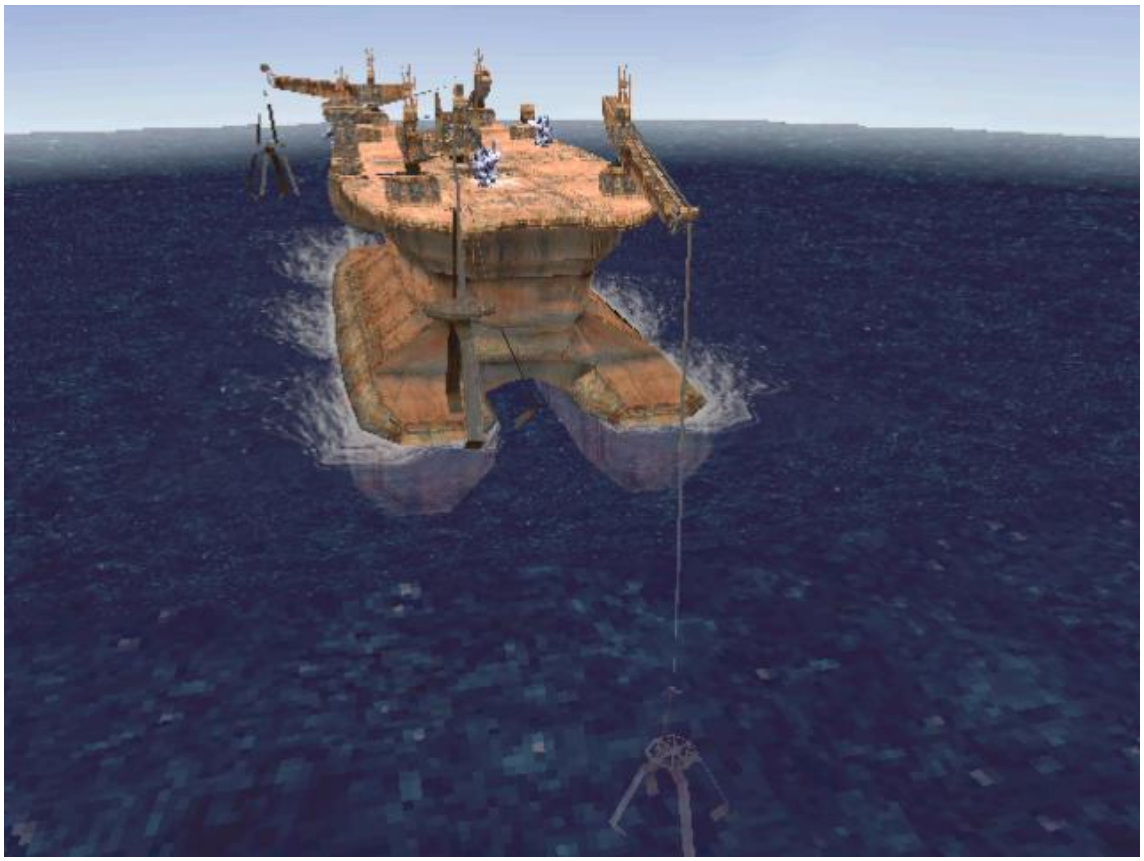


Figura 9: Xenogears y el efecto del mar.[9]

[9] Fuente: http://legendsk.com/wp-content/uploads/2014/02/xeno16.jpg?x88039

Por otro lado, tenemos los *fragment shaders*, que sirven para añadir el color y las texturas de los píxeles, haciéndolo de forma individualizada para cada uno de ellos. Por tanto, no interviene en nada relativo al esqueleto definido de triángulos y polígonos. Como es de imaginar, cuanto más se libera de carga un *fragment* o un *vertex shader*¸ esos cálculos se pueden emplear en otras funciones.

Para programar una GPU, existen lenguajes tanto a bajo nivel, como a medio y alto. Algunos ejemplos son, en Nvidia, GLSL, HLSL (*High Level Shading Language*), por parte de Microsoft y *Opengl Shader* y *CUDA* de Nvidia.

CG

Su nombre tiene su origen en "*C for Graphics*". Nvidia fue la que se encargó de gestionarlo. Está programado sobre lenguaje C, pero su sintaxis es muy diferente a lo que un programador de CPU está acostumbrado. Esto es así, porque está hecho con una estructura totalmente diferente, y antes de poder crear y ejecutar código, es necesario construir unos cimientos básicos sobre los que luego se va a trabajar.

Gracias a *Cg,* se pueden programar animaciones, geometrías, sombras, aparte de los mencionados *vertex* y *fragment* shaders.

Pero para programar, no sólo necesitamos el lenguaje, también tenemos que disponer de la API correspondiente (*Application Programming Interface*) que, como hemos comentado en diversas secciones de este trabajo, comunican la CPU con la GPU.

- OpenGL: Es un conjunto de librerías que nos permiten crear gráficos en 2D y 3D. Muchas videoconsolas han hecho uso de él. Normalmente se utiliza con el lenguaje de programación C++, que tiene rápida respuesta de programación y compilación. Hay diversas distribuciones, las que yo más he usado son Glut (que está descatalogada pero ofrece funcionalidad completa) y SFML (*Simple and Fast Media Library)*. En concreto, la he usado para programar videojuegos, y en este TFM desarrollaremos un apartado sobre el mismo poniendo como ejemplo mi experiencia real con OpenGL.
- El principal competidor de OpenGL es DirectX, creado por Microsoft. Es utilizado por la mayoría de videojuegos en los que se usa Windows como sistema operativo.

CUDA.

Figura 10: tarjeta gráfica Nvidia 8800 GTX[10]

Para terminar este apartado vamos a hablar de *CUDA*, usado en las tarjetas Nvidia desde el año 2006, y las tarjetas que la soportan van del dígito primero 8 en adelante.

Su principal característica es que une los procesadores de vértices y fragmentos, en uno solo, y el trabajo no está dividido en dos, sino que pueden usarse todos los núcleos que uno quiera, tanto de los vértices, como de los fragmentos.

Así, se consigue mucha mayor eficiencia y potencia a la hora de programar en la GPU. El lenguaje de programación que utiliza es muy parecido a *C*, y así, no pilla de sorpresa a lo que los programadores estaban acostumbrados a hacer.

5. Tendencias futuras.

5.1 Periféricos

Prácticamente todos los periféricos que utilizamos hoy en día son simples repeticiones de periféricos similares. Por supuesto, hay excepciones. A continuación vamos a explicar algunos de ellos, que vienen destacados en el artículo de revista de (20 minutos, 2014)

[10] Fuente: https://images10.newegg.com/NeweggImage/productimage/14-130-072-30.jpg

Debo antes comentar, que a pesar de ser un artículo de revista antiguo, en los últimos años no se ha avanzado tanto en estas tecnologías, por lo que en el año 2019 la situación sigue siendo parecida (o al menos, pese a que leo foros de informática, no veo que estos periféricos modernos estén ya siendo usados).

- Teclado táctil: las teclas físicas están a punto de ser sustituidas. Con un panel de vidrio curvo que emite luz, utilizando cámaras y luz infrarroja para leer los gestos y los golpes que damos con las manos. Para el ratón, se utiliza la superficie del teclado conocida como *trackpad*. Se pueden elegir diseños personalizados y hacer de este teclado esté en conjunto con nuestro PC.

- Reconocedor de la vista: si somos muy perezosos y no queremos usar el ratón y teclado para navegar por nuestro PC, existirá un dispositivo capaz de reconocer el movimiento de nuestros ojos, y así ordenar al ordenador lo que debe de hacer. Este descubrimiento ya lo fue como tal en la Wii normal de Nintendo, donde moviendo el *WiiMote* y con una barra negra se hacía las veces de ratón en pantalla. Pues es el mismo concepto, sólo que leyendo nuestros ojos.

- Guante que hace de ratón: si con un ratón y teclado no tenemos suficiente, existe y será comercializado un guante que sustituye los otros dos, con más de 30 órdenes distintas (18 puntos de contacto y 3 cojines). Originalmente estaba diseñado para control de juegos, pero hoy en día es también útil para *PCs*.

Como podemos ver, se sigue investigando y avanzando, y aunque quizás estos productos son sustitutos un poco raros de los originales, poco a poco se irán incorporando a nuestras casas y Pcs.

Figura 11: teclado táctil[11]

[11] Fuente: https://st-listas.20minutos.es/images/2014-07/384369/list_640px.jpg?1405443032

5.2 Soportes y dispositivos de almacenamiento

Para poner punto y final a las tendencias futuras de los puntos de los que trata nuestro proyecto, vamos a hablar de sistemas ya implantados, pero que se seguirán utilizando en el futuro, de una u otra forma. Son los RAID, los NAS y el almacenamiento en la nube.

Los RAID es un sistema de guardado de datos, que se basa en acoplar múltiples unidades de almacenamiento juntas, que bien pueden ser discos duros o tarjetas SSD generalmente. Aquí se replican o se redistribuyen los datos, ofreciendo una seria de ventajas.

Gracias a las múltiples conexiones, se consigue mayor capacidad, velocidad y tolerancia a fallos. Por otro lado, el uso de un RAID supone menor coste para sistemas que son más complejos.

La virtud que tienen, es que los datos se encuentran en ellos duplicados, por eso, si una unidad falla no hay problema porque los datos están copiados en las demás. Por otro lado, como la carga está dividida entre un número grande de estos discos, los accesos suelen ser mucho más rápidos.

Es pues, un sistema de duplicado de datos.

Se suelen utilizar en servidores, tanto en su versión hardware como software, actualmente.

Suelen usarse más en las empresas, donde hay una gran cantidad de información acumulada y hay que realizar copias de seguridad. Con ellos se consigue una mejora en los costes para sistemas complejos, a pesar que podría extenderse su uso en el mercado del usuario. Su competidor actual es la nube, que en principio, parece que es capaz de reducir costes, aportar mayor flexibilidad y espacio físico de cara a los usuarios finales. Veremos después de dos apartados.

Figura 12. Raid[12]

Nas

Es otra tecnología más de almacenamiento, dedicado a compartir la posibilidad de carga que ofrecen los servidores con ordenadores personales o servidores clientes a través de la red. Es, pues, una tecnología aplicada también (al igual que los RAID) al entorno empresarial y al hogar.

Su gran ventaja, es que los datos pueden ser accedidos rápidamente en los ordenadores clientes. A pesar de que supuestamente ofrecen esta gran ventaja, no han conseguido instaurarse en los hogares, quizás porque su precio es aún elevado.

Hoy en día, es el sustituto ideal del almacenamiento en la nube, que nos brinda la posibilidad de ampliar nuestra red más allá de lo local, y alcanzar así la red mundial, dando por supuesto, mayor disponibilidad. Juega a favor su velocidad, que aún no ofrece la red. De todos modos, NAS también puede ser conectado a internet.

Almacenamiento en la nube

[12] Fuente: https://ibericamultimedia.com/wp-content/uploads/2016/12/recuperar-datos-discos-raid-5-10-nas-servidores-mantenimiento-informatico-madrid-ds1515-synology-supermicro-hp-intel.jpg

Para almacenar información en la nube, tenemos que subir nuestros datos a servidores que están en internet. Suelen ser accesibles y remotos desde cualquier dispositivo y en cualquier momento o lugar, y presentan gran tolerancia a fallos.

Sistemas como *Dropbox, OneDrive*, *Google Drive* o el extinto *Mega*, nos permiten almacenar los datos que queramos en nuestra nube a nivel de usuario, y si es necesario, a nivel empresarial. Existen muchas web que tienen toda su información guardada en la nube, como Ebay, Amazon, Aliexpress, etc.

Gracias a la reducción de costes (precio bajo), sumado a la disponibilidad de la nube, hace que esta tecnología se esté expandiendo rápidamente. También suma el hecho de la reducción de espacio. Por otro lado, la nube se aprovecha de los RAID y las tecnologías de virtualización.

En dicha nube, aparte de datos, podemos alojar aplicaciones, archivos multimedia, videojuegos (*Steam*), bases de datos e incluso sistemas completos. Sus posibilidades son ilimitadas.

Quizás la única pega y desventaja, es que en la nube puede darse el caso de falta de privacidad o disponibilidad de conexiones a Internet de alta velocidad en algunos sitios. Aún así, los datos permanecen seguros ante fallos en los sistemas físicos (ordenadores finales).

Por otro lado, en la nube podemos sincronizar los datos entre sistemas, asimismo como versionar los archivos con diferentes ediciones para tener mayor disponibilidad.

Como es lógico, se pueden compartir usuarios y centros de datos fácilmente, teniendo todos los datos y pudiendo editar en tiempo real el espacio en la nube. Otro punto a favor, son los dispositivos móviles, que también pueden acceder a dicha nube y ejecutar toda la potencia de ella en ellos, aparte de más posibilidades.

Para concluir, la tendencia actual de datos en internet es la nube por sus ventajas mencionadas. Es pues, la tendencia futura que más posibilidades tiene de triunfar. No obstante, los soportes físicos siempre serán necesarios, que hagan de espejo y operen con los datos en ambos lados de las conexiones facilitadas por la nube.

Para que finalmente el almacenamiento en nube triunfe en el futuro, será condición necesaria y suficiente, el hecho de que nuestras conexiones sean rápidas, pues muchas veces llegaremos a ejecutar programas, videojuegos o ver películas que estarán almacenadas allí y se necesitará velocidad de aplicaciones en tiempo real.

Y en cuanto a soportes físicos, los discos duros irán desapareciendo, dejando su lugar a las tarjetas SSD.

Figura 13. Dibujo de la Nube.[13]

Moléculas

Lo que sí se quiere conseguir, que queda muy lejos de las tecnologías mostradas en este apartado, es guardar la información en moléculas a muy baja temperatura, como se comenta en la publicación de internet de (El grupo informático, 2018)

Figura 14: Molécula de ADN[14]

Por ejemplo, en el caso del ADN, un solo gramo de ADN podría guardar información relativa a millones de datos. Si utilizásemos de forma inteligente estas moléculas, se

[13]Fuente:https://cdn.computerhoy.com/sites/navi.axelspringer.es/public/styles/480/public/media/image/2015/04/94789-nube.jpg?itok=t3sz-QFT

[14] Fuente: https://cdn.elgrupoinformatico.com/Noticias/2018/06/adn-moleculas-almacenamiento-720x360.jpg

podría disponer de información acumulada durante mucho tiempo. Sin embargo, aún queda lejos esta opción y las 3 tendencias explicadas anteriormente serán parte de nuestro futuro.

5.3 Tarjetas gráficas

Con la continua evolución de las tarjetas gráficas, y después de poder utilizar la funcionalidad de vértices y polígonos de forma conjunta y no individual como antaño antes de los CUDA, vamos a ver en este apartado del futuro de las tarjetas, los siguientes puntos: "La nueva arquitectura unificada", "Capacidad de cómputo" y "Presente y futuro".

La nueva arquitectura unificada.

La llegada de CUDA supuso un cambio radical en la forma de programar *GPUs* que se tenía con Nvidia. A partir de ahí, y en las revisiones de sus tarjetas a partir del número 8, se usaron los procesadores de vértices y fragmentos en uno solo, consiguiendo resultados espectaculares.

Esto es así, ya que antes de introducir este modelo, cuando se procesaban vértices, el procesador de fragmentos estaba ocioso, y viceversa. Cuando se juntaron para trabajar en conjunto, se consiguieron tiempos de respuesta bastante más elevados.

Cuda surgió por las necesidades de muchas aplicaciones a las que les hacía falta mucha iluminación, o mucha geometría. Para concluir, añadir también que las *FPUs*, que son donde se realizan diversos cálculos, están mejor implementadas que en tarjetas anteriores.

Así pues, si hoy en día vamos a una tienda de informática a comprar una tarjeta gráfica, posiblemente usará la tecnología última explicada, mientras que si compramos una Nvidia de la gama 7 tendrá los procesadores separados en vértices y fragmentos y estará desactualizada.

Capacidad de cómputo

Con la introducción de estas tarjetas, año a año se van incrementando los números de GFLOPS y TFLOPs (velocidades a las que trabajan) y se van distanciando cada vez más de las velocidades que tienen las CPU. Se han invertido muchos millones en la creación de centros de supercomputación, aunque dado que cuesta mucho implantarlas (cuestiones monetarias) su número aún es pequeño.

Veamos la gráfica que refleja estos datos:

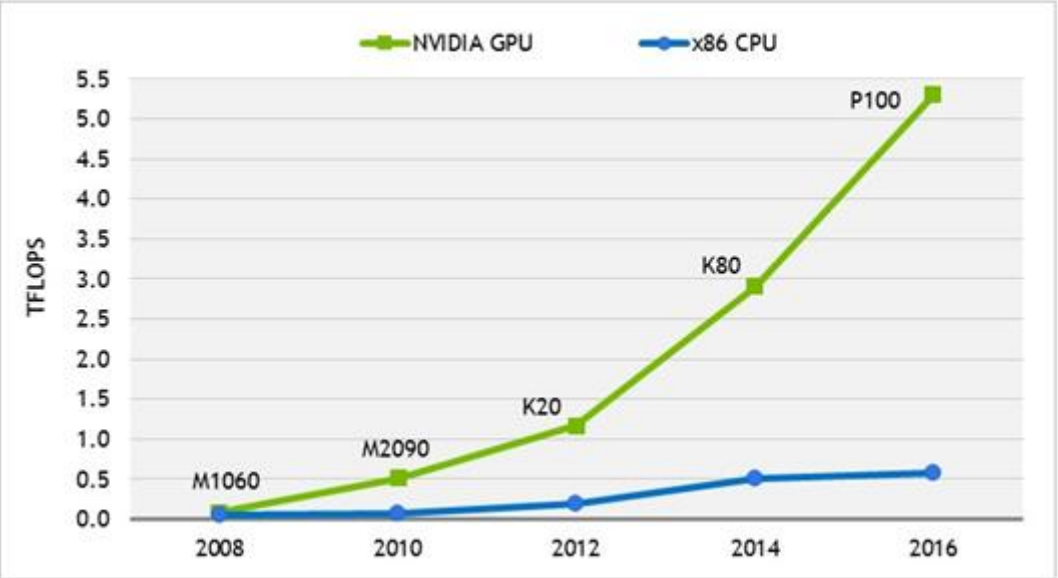


Figura 15: Comparativa GPU vs CPU (hasta el año 2016).[15]

Presente y futuro

Lo he hemos introducido anteriormente, es la evolución de las tarjetas gráficas en los años de su desarrollo y más cambiantes. Llegamos, pues, al punto de la tecnología CUDA, y pondremos el ejemplo más los datos técnicos de la última tarjeta más actualizada de Nvidia.

La última generación de *GPUs* GeForce de Nvidia es la GTX serie 400, alias FG100, que con gráficos DX 11 (Microsoft), ofrecen imágenes 3D con un realismo sobrecogedor, e inmersión total en los videojuegosgracias a la tecnología Nvidia 3D visión.

Esta tarjeta, que es la más puntera en la actualidad, dispone de 3000 millones de transistores y hace uso de todas las técnicas que hemos comentado anteriormente, algunas adaptadas y mejoradas, como el *antialiasing*. Los videojuegos que están por venir (y aplicaciones gráficas) harán uso de motores de física más avanzados, modelado 3D de personajes y escenarios más realista, y seguir con el uso de la arquitectura CUDA.

Conclusión

Como hemos podido verificar por nosotros mismos, una tarjeta gráfica contiene en sí un millón de conceptos y funciones clave para generar videojuegos, simuladores de vuelo, películas animadas, agencias espaciales, etc. A pesar de que cada ordenador posee una, no somos conscientes de la herramienta tan potente que está anclada a la placa base de nuestro PC. Es un tema interesante, pero también complicado, que

[15]Fuente:https://2s7gjr373w3x22jf92z99mgm5w-wpengine.netdna-ssl.com/wp-content/uploads/2017/07/kinetica_GPU_1.png

seguirá estando en auge, debido a la importancia que tienen hoy en día videojuegos, películas de animación, efectos especiales. Y en este apartado, hemos detallado sólo conceptos generales, en realidad, hay mucho más.

6. Conclusiones

Como hemos podido comprobar, el tratamiento de los periféricos, sistemas de almacenamiento y tarjetas gráficas, es algo muy interesante, que nos permite acercarnos a los orígenes de la informática, y ver, cómo en apenas 80 años, se ha crecido a gran escala.

Hablando de estos últimos, los disquetes dejaron de comercializarse alrededor del año 2005-2006, en detrimento de los CDs. Pero es tal la evolución de todos estos dispositivos, que se ha seguido mejorando y se han evitado, por un lado, la gran fragilidad de las cintas magnéticas que tenían los disquetes, y aunque parecían un medio muy seguro, tanto los Cds como los DVDs y Blue-Rays, no son resistentes a golpes y una leve rayita en su superficie puede significar que se rompan. Pero haciendo uso de las memorias SD, que cada vez tienen más capacidad de almacenamiento, se están consiguiendo, por un lado, más seguridad en estos dispositivos, y por otro, velocidad de acceso. Tanto es así, que consolas de última generación como la Nintendo Switch, han optado por este sistema y han ganado audiencia y métodos más fáciles y asequibles de programación. También, por otro lado, tenemos la posibilidad de guardar nuestros videojuegos, música, películas etc, en la nube. Si alguien por ejemplo quiere ahorrar espacio, esta opción es la más adecuada. Y para quien quiera coleccionar, dispone la posibilidad de toda la vida de comprar y guardar sus artículos.

En lo que a los periféricos se trata, siguen saliendo nuevas formas de orientar nuestro ocio, como por ejemplo las gafas 3D, que simulan juegos en 3D por medio de un casco. No obstante, debo decir que somos ligeramente reacios a estas nuevas tecnologías, quizá es normal porque de toda la vida hemos estado acostumbrados a teclado, ratón y pantalla.

En cuanto a las tarjetas gráficas se refiere, vemos que ha existido una tendencia a ir superando los números y estadísticas que nos ofrecían cada una, y en apenas 25 años, hemos pasado de videojuegos muy limitados (aunque con sus virtudes que no residían en los gráficos, por ejemplo el argumento, la jugabilidad, la música) a auténticas obras de arte que se confunden ya con las películas de animación, que, partiendo de la primera de ellas más conocida "*Toy Story*", es hoy ya una rutina, en la cual estos dibujos disponen de gráficos, diseñadores, y ordenadores avanzados que hacen que podamos mostrarlos en pantalla.

Antiguamente, cuando se compraba una tarjeta gráfica, daba un soporte reducido de un par de años, pero actualmente, con cada vez procesadores más potentes, las tarjetas gráficas han superado cada día más sus números de eficiencia (entre polígonos y vértices) y con una de rango mediano, se da soporte de sobra a los juegos más eficientes, con los motores más novedosos (como *Unreal*, Unity, etc.).

Para concluir y dar paso a la proyección didáctica, comentar nuevamente, que hemos tenido la posibilidad de vivir en una época, donde en apenas unos 50 años, se ha pasado de programar juegos como el *Pong*, a auténticas maravillas gráficas. ¿Cuál será nuestro tope? Cada día los programadores y desarrolladores de hardware y software nos sorprenden con cosas nuevas. ¿Qué inventarán ahora?

B. PROYECCIÓN DIDÁCTICA

1. Introducción

Antes de empezar a desarrollar esta unidad didáctica, hemos de comentar que la temporización de actividades coincide con la explicación de conceptos de OpenGL, algo que está muy relacionado con las tarjetas gráficas (haremos uso de la librería SFML: *Simple and Fast Media Library*). Dichas tarjetas en sí, nos ofrecen una serie de librerías y nos permiten trabajar con funciones manejando vértices y polígonos. No obstante, antes de poder entrar en conceptos relativos a dichos vértices y polígonos en OpenGL, hemos de tener un conocimiento medio/alto de las librerías de dicho motor. Sin embargo, nuestra unidad está más orientada a principiantes y a intentar manejar funciones que daremos prácticamente hechas a nuestros estudiantes para que hagan un juego en 2D simple. Por eso, aunque están ligados los conceptos de videojuegos y tarjetas gráficas, aquí sólo veremos brevemente de pasada los conceptos de polígonos y vértices indicados, y haremos más enfoque en hacer uso de tiles, sprites, y las funciones *glTexCoord*, *glVertex2D*, *glScale*, más algunos conceptos de colisiones e introducción de sonidos gracias a la librería SFML. Todo orientado a juegos en dos dimensiones, y no en 3D. Aun así, en el aspecto de explicación de cómo hacer un videojuego, el mismo basado en coches de carreras, hace uso de polígonos, y servirá como punto de partida a las funciones que usan a nivel avanzado las tarjetas gráficas.

La asignatura que nos servirá de base para esta unidad didáctica es “Programación Multimedia y Dispositivos Móviles”. Obviaremos la parte de dispositivos móviles, tanto como la de 3D, y centraremos nuestros esfuerzos en iniciarnos (como he comentado) en OpenGL e intentar hacer un juego en 2D sencillo como es el Comecocos.

Como último comentario, si bien es cierto que todo mi conocimiento se lo debo en parte al Máster de la Universidad Oberta de Cataluña, esta parte del TFM ha sido desarrollada exclusivamente por mí.

2. Contextualización en el centro

El centro en el que vamos a dar esta unidad didáctica es el IES Virgen del Carmen. Dicho centro se haya localizado en la ciudad de Jaén, en una zona estratégica al lado del antiguo campo de la Victoria, en lo que hoy es el gimnasio “La Victoria” y cerca del

Corte Inglés. Concretamente en la calle “Paseo de la Estación 44 con código postal 23008”.

En la siguiente imagen podemos ver el lugar donde está situado:

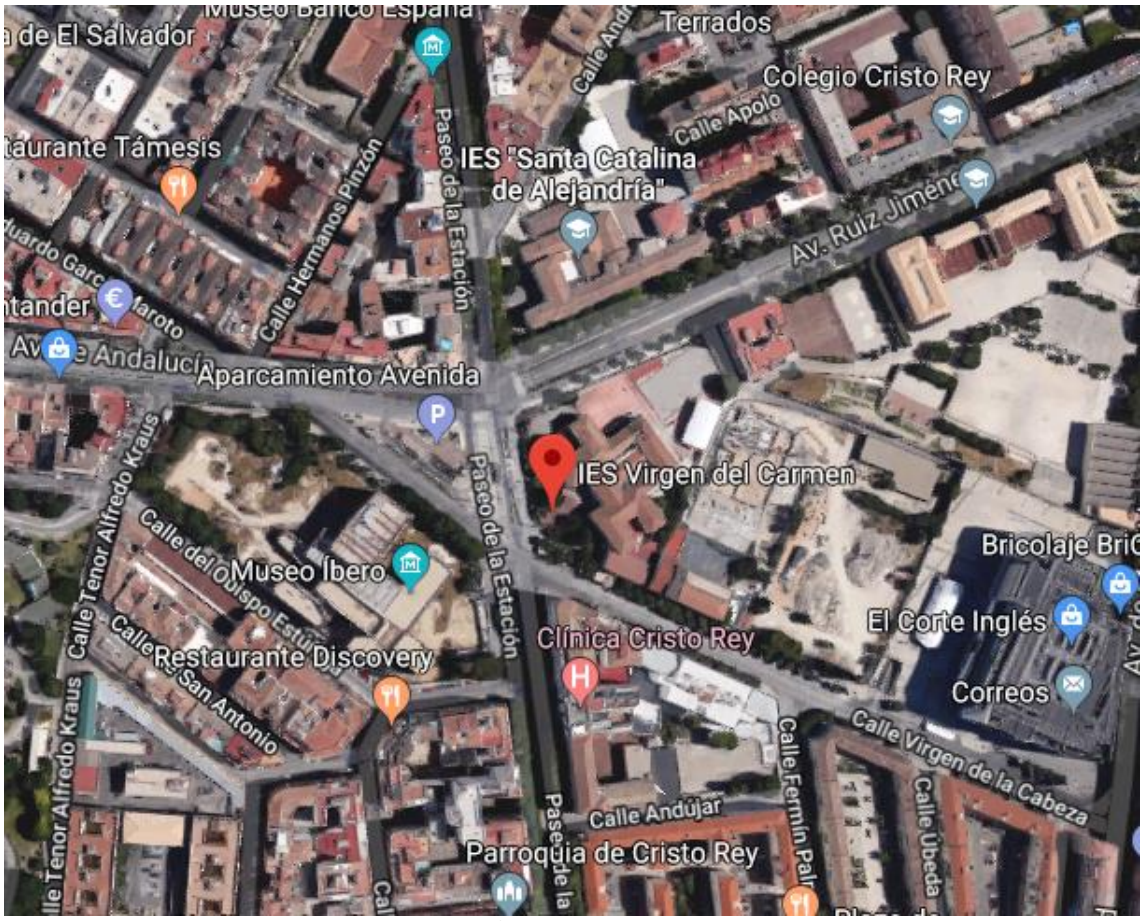


Figura 16: IES Virgen del Carmen por satélite[16]

Uso una imagen de satélite, teniendo en cuenta que no está actualizada dicha zona (por ejemplo no se ve construido el gimnasio adyacente).

Aunque el centro dispone de una gran cantidad de aulas (algunas con pizarras digitales y elementos muy actualizados) daremos nuestra clase en el aula que tiene una disposición que permite alojar a un gran número de alumnos.

3. Unidad Didáctica

[16] Fuente: http://maps.google.com

ÍNDICE UD EN CFGM DESARROLLO DE APLICACIONES MULTIPLATAFORMA.

1. Normativa

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.

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.

2. Título

a. **Etapa: CFGS**
b. **Módulo: Programación Multimedia y dispositivos móviles**

c. **Curso: Segundo Curso (*ir a ANEXO II: distribución horaria semanal, …*)**

d. **Trimestre: Primero**
e. **Número de sesiones: (10 sesiones de 1 hora cada una, y sesión doble de 2 horas al filnal)**
f. **Temporalización número de horas: 12 horas**

3. Competencia general

Artículo 4. *Competencia general.*

La competencia general de este título (ORDEN de 16 de junio de 2011) 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.

4. Orientaciones pedagógicas

Este módulo profesional (ORDEN de 16 de junio de 2011)contiene la formación necesaria para desempeñar la función de desarrollo de aplicaciones multimedia, juegos y aplicaciones adaptadas para su explotación en dispositivos móviles.

La función de desarrollo de aplicaciones multimedia, juegos y aplicaciones adaptadas para su explotación en dispositivos móviles incluye aspectos como:

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.

Las actividades profesionales asociadas a esta función se aplican en el desarrollo de software multiplataforma en empresas especializadas en la elaboración de contenidos multimedia, software de entretenimiento y juegos.

5. Competencias profesionales, personales y sociales

La formación del módulo (ORDEN de 16 de junio de 2011) contribuye a alcanzar las competencias profesionales, personales y sociales de este título que se relacionan a continuación:

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.

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.

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.

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.

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.

6. Líneas de actuación

Las líneas de actuación en el proceso de enseñanza-aprendizaje (ORDEN de 16 de junio de 2011) que permiten alcanzar los objetivos del módulo versarán sobre:

- El análisis de las tecnologías disponibles para dispositivos móviles, sus características y funcionalidad.

- La utilización de emuladores para evaluar el funcionamiento tanto de las aplicaciones para dispositivos móviles desarrolladas como de las modificaciones introducidas en aplicaciones existentes.

- 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.
- El desarrollo de aplicaciones que integran objetos multimedia.

- El análisis de motores de juegos, sus características y funcionalidades.

- El desarrollo de juegos 2D y 3D aplicando técnicas específicas y utilizando instrucciones.

7. Objetivos Generales

La formación del módulo (ORDEN de 16 de junio de 2011) contribuye a alcanzar los objetivos generales de este ciclo formativo que se relacionan a continuación:

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.

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.

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.

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.

8. Resultados de aprendizaje

Siguiente el (ORDEN de 16 de junio de 2011) tenemos:

1. Aplica tecnologías de desarrollo para dispositivos móviles evaluando sus características y capacidades.

2. Desarrolla aplicaciones para dispositivos móviles analizando y empleando las tecnologías y librerías específicas.

3. Desarrolla programas que integran contenidos multimedia analizando y empleando las tecnologías y librerías específicas.

4. Selecciona y prueba motores de juegos analizando la arquitectura de juegos 2D y 3D.

5. Desarrolla juegos 2D y 3D sencillos utilizando motores de juegos.

9. Objetivos didácticos

Para el punto 4: Selecciona y prueba motores de juegos analizando la arquitectura de juegos 2D y 3D.

a) Analizar los componentes de un motor de juegos.

b) Identificar los elementos que componen la arquitectura de un juego 2D y 3D.

c) Analizar los entornos de desarrollo de juegos.

d) Analizar diferentes motores de juegos, sus características y funcionalidades.

e) Identificar los bloques funcionales de un juego existente.

f) Definir y ejecutar procesos de render.

g) Reconocer la representación lógica y espacial de una escena gráfica sobre un juego existente.

Para el punto 5: Desarrolla juegos 2D y 3D sencillos utilizando motores de juegos.

a) Establecer la lógica de un nuevo juego.

b) Crear objetos y definido los fondos.

c) Instalar y utilizar extensiones para el manejo de escenas.

d) Utilizar instrucciones gráficas para determinar las propiedades finales de la superficie de un objeto o imagen.

e) Incorporar sonido a los diferentes eventos del juego.

f) Desarrollar e implantar juegos para dispositivos móviles.

g) Realizar pruebas de funcionamiento y optimización de los juegos desarrollados.

h) Documentar las fases de diseño y desarrollo de los juegos creados.

Contenidos Conceptuales

Según la (ORDEN de 19 de julio de 2010) tenemos lo siguiente.

Utilización de librerías multimedia integradas:

- Conceptos sobre aplicaciones multimedia.
- Arquitectura del API utilizado.
- Fuentes de datos multimedia. Clases.
- Datos basados en el tiempo.
- Procesamiento de objetos multimedia. Clases. Estados, métodos y eventos.
- Reproducción de objetos multimedia. Clases. Estados, métodos y eventos.

Análisis de motores de juegos:

- 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.
- APIs gráficos 3D.
- Estudio de juegos existentes.
- Aplicación de modificaciones sobre juegos existentes.

Desarrollo de juegos 2D y 3D:
- Entornos de desarrollo para juegos.
- Integración del motor de juegos en entornos de desarrollo.
- Conceptos avanzados de programación 3D.
- Fases de desarrollo.
- Propiedades de los objetos, luz, texturas, reflejos, sombras.
- Aplicación de las funciones del motor gráfico. Renderización.
- Aplicación de las funciones del grafo de escena. Tipos de nodos y su utilización.
- Análisis de ejecución. Optimización del código.

11. Aplicación práctica. Mis videojuegos

No son un número muy elevado. En total he programado 7 u 8 videojuegos. Que tengan aplicación aquí vamos a hablar de:

- *Bubble Burst*
- *Mario Land 2*
- Juego Open GL en 3D
- Fuocball
- *Racing Game*

Pero por otro lado, comentar que antes de pasar a conocer las funciones de vértices y polígonos, más tecnología 3D existente en las tarjetas gráficas, son necesarios unos conocimientos previos de OpenGL. Por ellos, intentaremos hacer una aproximación a estos conceptos más asequibles, y los alumnos que lo deseen, después, podrán profundizar más en el conocimiento de tarjetas gráficas y videojuegos.

Porque todos ellos tienen una aplicación gráfica en cuanto a OpenGL, (Gomila, 2019) se trata. Para cada uno de ellos mostraremos un vídeo. Inicialmente, expongo este tema, y después en nuestra unidad didáctica, todos estos videojuegos serán explicados en una clase teórica, que dará los conocimientos básicos a los alumnos (en realidad se les ofrece desglosados cómo se programan estos juegos, y no tienen necesidad de pensar).

De todas formas, antes de pasar a profundizar, debo decir que mis primeros juegos están hechos en Glut, un conjunto de librerías gráficas, que tiene la pega de que, a pesar de proporcionar funcionalidad total, ya no se van actualizando. También además, hay que añadir por otro lado librerías para tratar *sprites*, el sonido (corona y fmod, por ejemplo, respectivamente), la conexión en red, etc. Recientemente, aunque ya me lo recomendaron antes, descubrí SFML (*Simple and Fast Multimedia Library*), que es una interfaz que se conecta al ordenador, y que facilita la programación de videojuegos y aplicaciones multimedia. Está compuesto por 5 grupos de librerías (módulos): sistema, ventanas, gráficos, sonido y red. El último juego del que aquí vamos a hablar, sí está hecho con este sistema que además, cuenta con revisiones frecuentes cada año.

Pasemos pues, a describir las conexiones que existen entre mis videojuegos, con respecto a OpenGL.

Bubble Burst y Mario Land 2

Aquí vamos a hablar de las funciones GltexCoord2f y Glvertex2i. Estas dos funciones nos permiten guardar en una foto (por ejemplo), todos los *sprites* o *tiles* correspondientes a un videojuego. Con *GltexCoord* seleccionamos la posición en concreto y el Sprite o tile correspondiente, y con *Glvertex*, lo pintamos en pantalla en la posición que le indicamos.

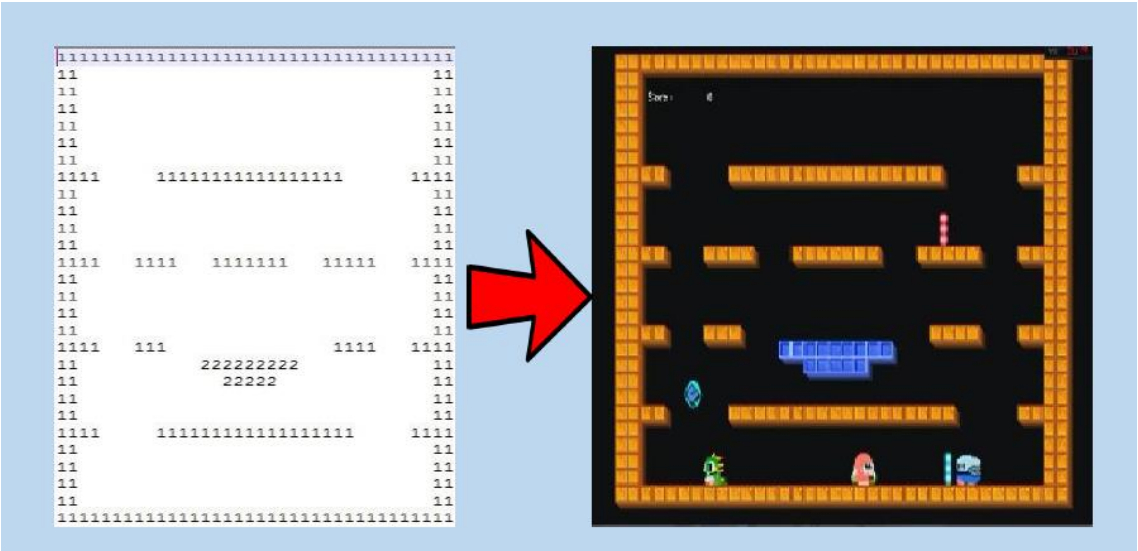


Figura 17: Transformación del mapa en *sprites* y *tiles*.[17]

Esto se consigue, haciendo uso, además, de dos programas hechos en Python, que nos permiten guardar información precisa sobre los *sprites* o *tiles*, y de otro programa para Linux llamado *Darkfunction* que comprime los *sprites* del personaje o tile en cuestión.

Por ejemplo, en Mario Land 2 tenemos:

[17] Fuente: a partir de aquí, todas las fotos las he realizado yo sobre mis videojuegos.

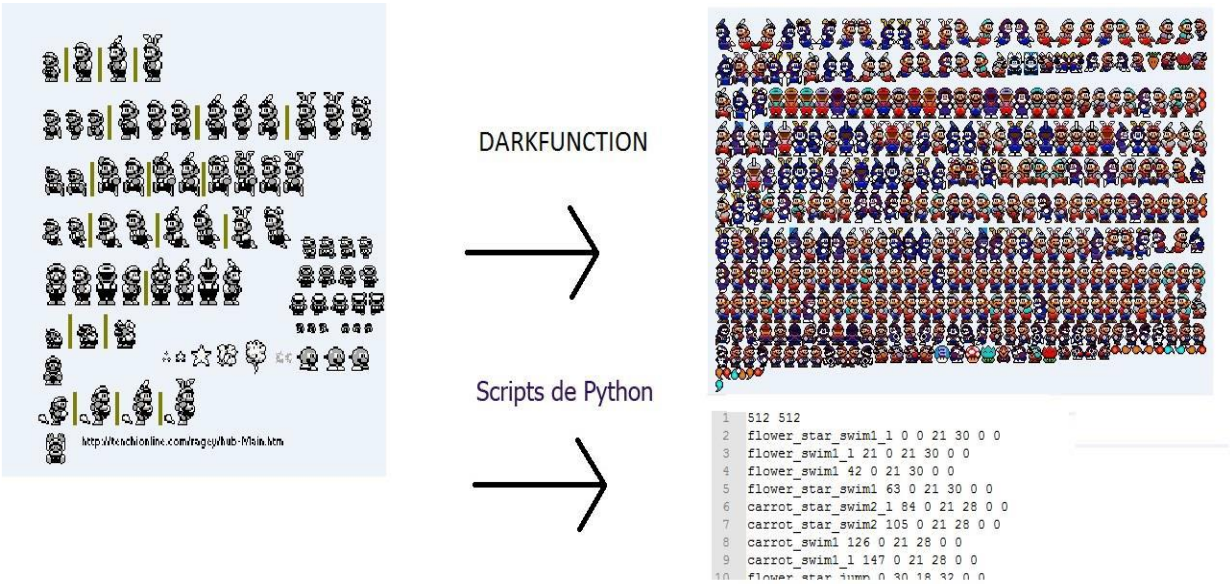


Figura 18: Scripts de Python y Darkfuntion.

Por un lado, disponemos del conjunto de los *sprites* comprimidos en un archivo de imagen, y debajo, una tablita con información de cada uno de ellos. Por ejemplo, si vamos a dibujar cuando Mario anda de derecha a izquierda, son en total 3 *sprites*. En este mapa, los localizamos, y después, en un archivo de texto especial, hacemos que entre en bucle dibujando cada una de las 3 posiciones, según pulsamos nosotros el botón de moverse a la izquierda. Para desarrollar toda esta actividad, como hemos explicado en el párrafo anterior, haremos uso del programa *Darkfunction* más unos programitas en *Python* que nos permiten organizar todas las *tiles* y *sprites*, en fotos con dichos elementos ordenados. Estos programas deben ser usados en *Linux.*

```
begin flower_left_run
fps 10
loop
frame flower_move1_l
frame flower_move2_l
frame flower_move3_l
end
```

Figura 19: Loop para cuando Mario se mueve de derecha a izquierda

No debemos olvidar, que la cámara está presente en este videojuego. Y se posiciona en la cabeza del Sprite de Mario. De esta forma, hacemos que la cámara se mueva como en un juego 2D clásico.

Juego OpenGl en 3D.

Este juego es algo básico para ser en 3D, pero al menos simula el cielo (no sé qué pasó pero no me funcionaba bien el *terrain*, es decir, el suelo).

Usando unos modelos del *Quake 2*, conseguí hacer en OpenGL esta pequeña simulación 3D. Con colisiones, basadas en un modelo tridimensional, cuando el disparo alcanzaba a los enemigos, iban muriendo.

Está hecho en Glut, al igual que los dos anteriores, y hacía uso de fmod para introducir el sonido.

Fuocball.

Un buen ejemplo de un videojuego sencillo usando *sprites* y tiles puede ser mi Fuocball. En él, se pinta un Sprite de fondo, que es el campo de fútbol, y los *sprites* en movimiento se multiplican por 11, para hacer todos los equipos. Aparte de estos 12 elementos, tendremos el último que es la pelota, que va botando de un lado a otro del campo.

Racing Game.

Y cómo engañar al ojo, haciendo un juego que 2D y convertirlo a 2.5D o 3D.

Primeramente, hemos de imaginar el mundo de un videojuego de coches. Así, viendo la siguiente imagen:

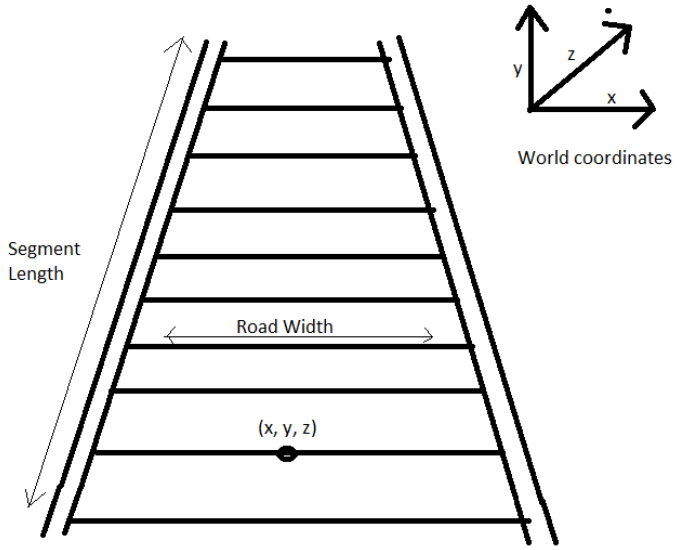


Figura 20: mundo en un juego de coches

Tenemos los siguientes puntos que hemos de anotar:

- Dibujamos 3 rectángulos. Uno para la carretera, otro para las líneas que delimitan la carretera, y otro para el césped de los lados.
- Estos rectángulos, van haciéndose más pequeños, pero engañando al ojo, lo que conseguimos es una sensación de profundidad.
- Si vamos a dibujar farolas al lado, o señales de “gire a la izquierda”, u otros coches, hacemos uso de la función *GLScale*, que adapta el tamaño a la profundidad. (antiguamente esto no se podía hacer así y había un tile para X tamaño o Y tamaño, que conforme se avanzaba en profundidad, se dibujaba el más grande o más pequeño), pero hoy en día, con la potencia de los ordenadores, hacemos uso de *GlScale*.
- Usamos una imagen de fondo, es fija, inalcanzable, pero hace más vistoso el videojuego.
- Fijamos la cámara ligeramente por encima del circuito para propiciar esa sensación de profundidad.

Ahora, hablemos del circuito:

El circuito puede ser un concepto muy abstracto de tratar, pero en realidad es también sencillo. En la siguiente figura:

```
for (int i = 0; i < 1600; i++){
    Line line;
    line.z = i*segL;


    if (i > 100 && i < 300)
        line.curve = -0.5;

    if (i > 300 && i < 700)
        line.curve = 0.5;

    if (i>750)
        line.y = sin(i / 30.0) * 1500;

    if (i % 20 == 0){
        line.spriteX =- 2.5;
        line.sprite = sFarola;

    }

    lines.push_back(line);
}
```

Figura 21: Circuito programado.

Vemos como hay un conjunto de líneas. Según la profundidad, veremos un límite (que es lo que se nos muestra en pantalla). Pero en realidad, en este bucle, todo el circuito está creado, y nos indica que de la línea 0 (que es una estructura de datos con información concreta, que veremos en la siguiente imagen) a la 1600, hay una curva a la izquierda de las líneas 100 a 300, una a la derecha de la 300 a 700. Que a partir de la línea 750, el terreno se inclina, haciendo uso de una función seno (según la altura de la función seno, la carretera se dibuja más arriba o más abajo, simulando pues, el desnivel del terreno). Y también, para cada 20 líneas, se dibuja una farola.

Aquí vemos pues cómo se vería el circuito en 2D:

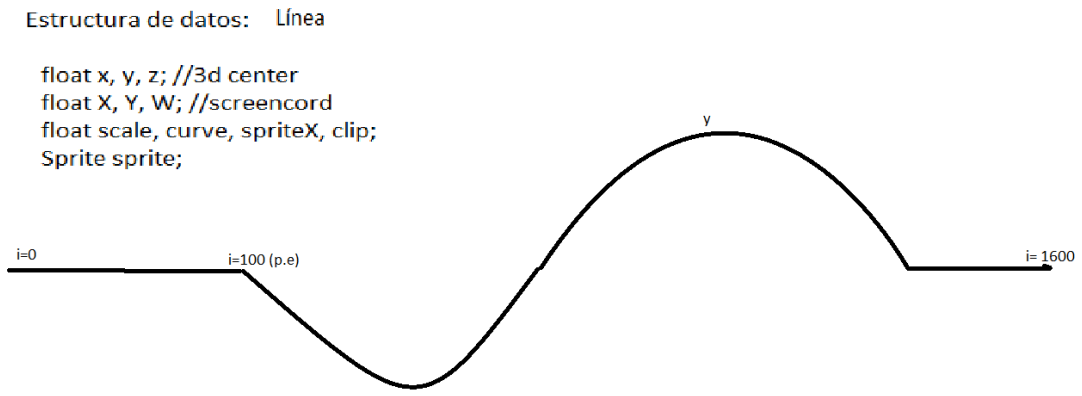


Figura 22: Circuito y estructura de datos “Línea”.

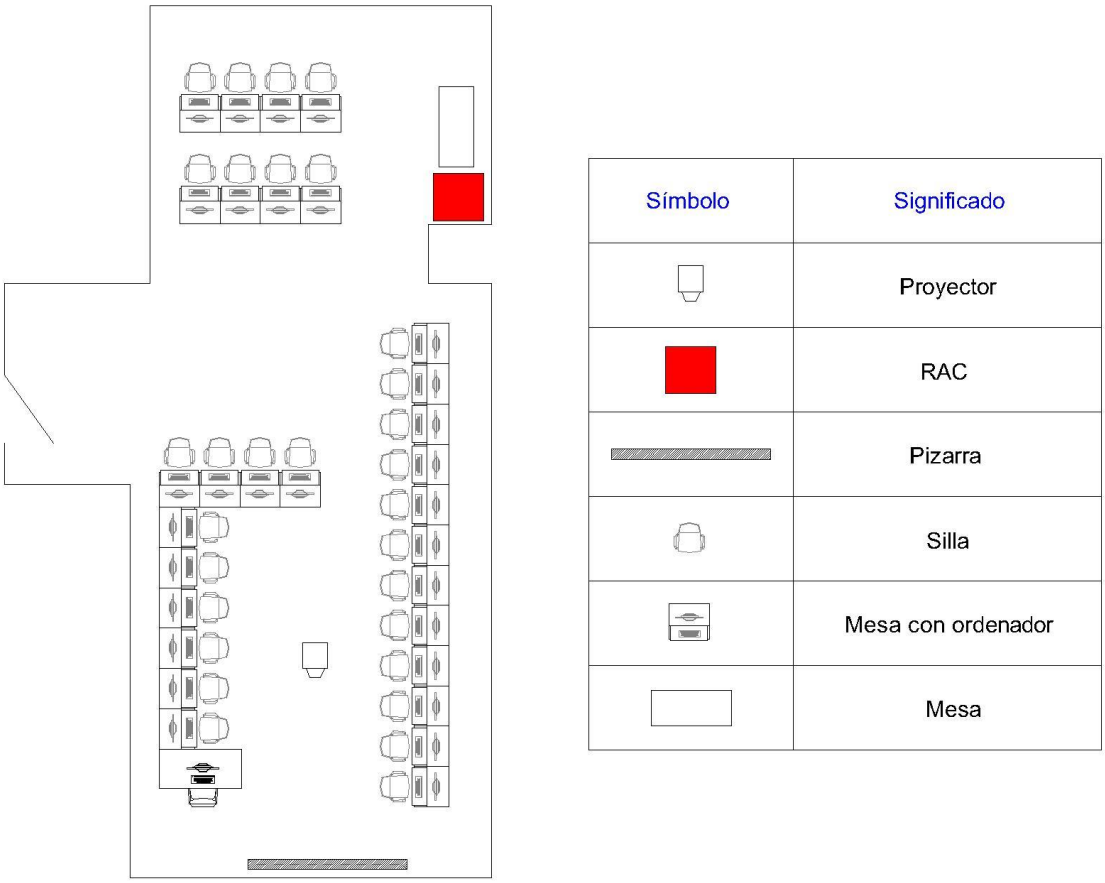
De una forma muy sencilla, creamos un juego de coches en 3D con truquillos de OpenGl.

12. Metodología

La Metodología didáctica responde a la cuestión ¿cómo enseñar? y hace referencia al conjunto de decisiones que deben ser tomadas a fin de orientar el desarrollo en el aula de los procesos de enseñanza y aprendizaje. Las opciones metodológicas estarán orientadas al aprendizaje significativo, constructivista y funcional de los diferentes contenidos.
Los aspectos metodológicos que se pretenden aplicar en la UD descansan en la idea de que el alumno/a se considere parte activa de la actividad docente, de manera que se pretende involucrarlo en el proceso de asimilación de nuevos conceptos y adquisición de capacidades, no como un mero contenedor de éstas, sino como un productor directo de estos conocimientos y habilidades en sí mismo.
El docente también debe orientar el trabajo escolar de sus alumnos, proporcionando las
indicaciones necesarias para que los alumnos puedan resolver los problemas que el estudio les plantea. A lo largo de las unidades didácticas que desarrollamos en el módulo, fomentaremos los hábitos de tenacidad constancia, laboriosidad, etc. Un aspecto importante de esta función orientadora es decidir qué actitudes se debe conseguir en los estudiantes, cuáles deben modificarse y cómo reforzar las positivas.

12.1 Organización del aula

Las clases serán impartidas en el aula 0.5 del IES Virgen del Carmen, ya que hay muchos ordenadores y su distribución en V hace que nadie tape a nadie. Esto es un laboratorio de prácticas y vendrá muy bien para desarrollar la unidad didáctica.



Símbolo	Significado
	Proyector
	RAC
	Pizarra
	Silla
	Mesa con ordenador
	Mesa

Figura 23: Aula 0.5 del IES Virgen del Carmen

Los alumnos harán uso de apuntes para tomar nota de lo que se diga en la pizarra.

El proyector, superpuesto a la pizarra blanca, podrá desplazarse hacia arriba y hacia abajo, en caso de que se quieran "poner" ejemplos de programación de juegos y haga falta escribir con rotulador.

12.2 Tipos de agrupamiento

Para poder aprender bien e iniciarse adecuadamente en videojuegos es necesario hacer los trabajos individualmente. No obstante si tienen alguna duda pueden consultar tanto al profesor como a otro alumno.

12.3 Materiales y recursos didácticos

El alumnado dispondrá:

- Hardware: Ordenadores del aula, impresora, scanner y cañón
- Pizarra
- Software:

o SO Windows (es necesario al menos, en los ordenadores del aula 0.5. Se intentará que al menos en esta aula, haya distribuciones Windows, para poder ejecutar correctamente el *Visual Studio*)
o Paquete Open Office
o Visual Studio 2013 (gratuito)
o Librerías SFML (Simple and Fast Media Library) (gratuitas).

- Libros de consulta
- Material didáctico: Presentaciones, documentos o ficha de prácticas elaboradas por el profesorado.

12.4 Metodología y tipos de actividades

Las actividades de enseñanza-aprendizaje son la manera activa y ordenada de llevar a cabo las propuestas metodológicas o experiencias de aprendizaje. La metodología seguida para para conseguir un aprendizaje significativo por parte del alumno/a seguirá la siguiente estructura. Para hacer interactivas nuestras clases, tendrán siempre la misma estructura, salvo las 2 horas de la última sesión que serán usadas para exponer el videojuego planteado.

- Clase teórica de 1 hora: explicando los conceptos correspondientes de programación de videojuegos.
- Clase práctica o aplicación de la teoría de 1 hora: aquí trabajan los alumnos y se espera que asimilen los conceptos explicados en la hora anterior.

Como puede observarse, por otro lado, la organización sigue las siguientes estrategias metodológicas:

- Estrategia expositiva: donde las lecciones magistrales son necesarias para introducir en los nuevos contenidos al alumnado

- Estrategia de indagación: donde el alumnado tiene que descubrir a través de diferentes actividades cómo desarrollar muchas de sus competencias. (prácticas o aplicación de los contenidos teóricos).

Las actividades hacen referencia a las tareas realizadas por los alumnos y alumnas con la finalidad de adquirir determinados aprendizajes. A continuación detallamos el tipo de actividades que tendrán lugar en esta U.D:
a) **Desarrollo de Contenidos**: que son dirigidas por el profesorado para que el alumnado descubra progresivamente la estructura de los contenidos de la materia
b) **Aplicación teórica**: investigar sobre un tema en concreto, que ha sido explicado en la clase anterior y no es necesario programar (por ejemplo, hacer un resumen de la historia de los videojuegos).

c) **Prácticas**: aprendizaje de manera práctica de los contenidos vistos en clase. Comprenderá desde configurar el Visual Studio a programar el videojuego Comecocos.
d) **Evaluación**: basta con asistir a clase y realizar el videojuego para aprobar esta sección de la asignatura.
e) **Exposición**: videojuego realizado por cada alumno, bien en horas de clase o en casa, con el fin de hacer una exposición y explicar cómo lo han programado.

12.5 Temporalización de actividades

Metodología. Temporización de actividades.				
Actividad	***Tiempo***	***Sesión***	***Agrupamiento***	***Recursos***
				-
Desarrollo de contenidos - Animación 2D. - Arquitectura del juego. Componentes.	1 hora	1	Individual	Ordenador Visual Studio (su versión 2013 que es gratuita) Acceso a internet
Aplicación teórica: Desarrollo diagrama de flujo. Memoria con resumen historia de videojuegos.	1 hora	1	Individual	Ordenador Visual Studio Acceso a internet Microsoft Office o similar gratuito.
Refuerzo: repaso de la clase anterior	15 minutos	2	Individual	Ordenador
Desarrollo de contenidos - Motores de juegos. Tipos y utilización. - Áreas de especialización, librerías utilizadas y lenguajes de programación.	45 minutos	2	Individual	Ordenador Visual Studio Acceso a internet
Aplicación práctica: Configurar Visual Studio en C++ y con SFML.	1 hora	2	Individual	Ordenador Visual Studio Acceso a internet
Refuerzo: repaso de la clase anterior	15 minutos	3	Individual	Ordenador Acceso a internet

Desarrollo de contenidos - Componentes de un motor de juegos. - Librerías que proporcionan las funciones básicas de un Motor 2D.	45 minutos	3	Individual	Ordenador Visual Studio Acceso a internet
Aplicación práctica Pruebas básicas con SFML (OpenGL)	1 hora	3	Individual	Ordenador Visual Studio Acceso a internet
Refuerzo: repaso de la clase anterior	10 minutos	4	Individual	Ordenador Acceso a internet
Desarrollo de contenidos - APIs gráficos 3D. - Estudio de juegos existentes.	50 minutos	4	Individual	Ordenador Visual Studio Acceso a internet
Aplicación práctica: Familiarizarse con las funciones y códigos de videojuegos que se le proporcionan.	1 hora	4	Individual	Ordenador Visual Studio Acceso a internet
Refuerzo: repaso de la clase anterior	10 minutos	5	Individual	Ordenador Acceso a internet

Desarrollo de contenidos - Aplicación de modificaciones sobre juegos existentes. - Entornos de desarrollo para videojuegos.	50 minutos	5	Individual	Ordenador Visual Studio Acceso a internet
Aplicación práctica: Mas conceptos de programación e inicio del "Comecocos".	1 hora	5	Individual	Ordenador Visual Studio Acceso a internet
Evaluación: Presentación de los videojuegos "Comecocos" hechos por los alumnos.	2 hora	6	Individual y grupal	Ordenador Visual Studio Acceso a internet

12.6 Explicación de las sesiones

- Sesión 1: hacemos una pequeña puesta de contacto de los alumnos con los videojuegos. Se les explica el bucle de un videojuego y cómo se programa la animación 2D. La aplicación teórica de esta sesión, consiste en hacer un esbozo de un videojuego que nos hayamos imaginado. También se entregará un resumen con un documento referente a la historia de los videojuegos.

- Sesión 2: esta sesión empieza con un repaso de la unidad anterior. Aquí veremos los motores más famosos de los videojuegos. En cuáles se programan dichos juegos y cómo de útiles son, o cómo trabajan dando soporte a los programadores más útil de cómo se hacía antiguamente. Como aplicación práctica, los alumnos deberán configurar el Visual Studio para montarlo con las librerías SFML (algo que no es sencillo de hacer) en C++, lenguaje por antonomasia de los videojuegos por sus bajos tiempos de ejecución y compilación.

- Sesión 3: entramos en profundidad un poco sobre el tema de los programas que se usan para programar videojuegos, las librerías de SFML y también los motores de juegos (qué es en sí un motor), que no deja de ser una arquitectura gracias a la cuál se programa un juego, y si se aprovecha bien, sus secuelas. Los alumnos harán pruebas con las librerías ya instaladas, por ejemplo, crear un *sprite*, hacer que se anime, que se mueva por la pantalla, añadirle efectos visuales gracias a funciones de SFML, etc.

- Sesión 4: veremos algunos de los APIs gráficos en 3D, y se estudiarán los juegos ya existentes (cómo se programa un plataformas, un beat em up, un juego de coches, un Tetris, etc.). Al alumno se le pedirá hacer un juego “Comecocos”, y usarán secciones de código claves para hacerlo, algo que no le debería resultar complicado, y en la segunda hora de esta sesión se dedicará a ir probando las funciones que se le han proporcionado.

- Sesión 5: se seguirá con el visionado/aprendizaje sobre cómo se programan los videojuegos, utilizando otros conceptos de programación. En concreto se creará el mapa en un fichero de texto, se usarán las *tiles*, con las funciones *GltexCoord* y *GlVertex*, y se animará el Sprite, con el objetivo de encarrilar el videojuego “Comecocos”.

- Sesión 6: cada alumno mostrará su videojuego “Comecocos” a los demás, viendo las triquiñuelas de programación que han usado, aprendiendo así todos de forma conjunta.

13. Evaluación y Recuperación. Criterios de evaluación

Para cada resultado de aprendizaje, hay una serie de criterios de evaluación.

3. Desarrolla programas que integran contenidos multimedia analizando y empleando las tecnologías y librerías específicas.

Criterios de evaluación:

a) Se han analizado entornos de desarrollo multimedia.

b) Se han reconocido las clases que permiten la captura, procesamiento y almacenamiento de datos multimedia.

c) Se han utilizado clases para la conversión de datos multimedia de un formato a otro.

d) Se han utilizado clases para construir procesadores para la transformación de las fuentes de datos multimedia.

e) Se han utilizado clases para el control de eventos, tipos de media y excepciones, entre otros.

f) Se han utilizado clases para la creación y control de animaciones.

g) Se han utilizado clases para construir reproductores de contenidos multimedia.

h) Se han depurado y documentado los programas desarrollados.

4. Selecciona y prueba motores de juegos analizando la arquitectura de juegos 2D y 3D.

Criterios de evaluación:

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.

5. Desarrolla juegos 2D y 3D sencillos utilizando motores de juegos.
Criterios de evaluación:

a) Se ha establecido la lógica de un nuevo juego.

b) Se han creado objetos y definido los fondos.

c) Se han instalado y utilizado extensiones para el manejo de escenas.

d) Se han utilizado instrucciones gráficas para determinar las propiedades finales de la superficie de un objeto o imagen.

e) Se ha incorporado sonido a los diferentes eventos del juego.

f) Se han desarrollado e implantado juegos para dispositivos móviles.

g) Se han realizado pruebas de funcionamiento y optimización de los juegos desarrollados.

h) Se han documentado las fases de diseño y desarrollo de los juegos creados.

Evaluación de la unidad didáctica con una pequeña rúbrica

Como en esta UD sólo vamos a evaluar el videojuego Comecocos (y suponemos que todos los alumnos han tenido un porcentaje de asistencia superior al 80%), detallamos a continuación su rúbrica. Si la asistencia es inferior a la comentada y no está justificada, no se puede aprobar la asignatura. Dicha rúbrica aparece un par de páginas más adelante, pero antes debemos terminar este apartado y exponer el enunciado del ejercicio de "Comococos" propuesto.

La evaluación nos sirve para conocer el grado de progreso alcanzado por los alumnos en relación a los objetivos propuestos, así como para determinar si la enseñanza ha sido adecuada o no para alcanzar dichos objetivos.

Enunciado del ejercicio propuesto.

Resolución de un videojuego basado en el clásico Comecocos. Deben tenerse en cuenta los siguientes criterios.

1. Dada una función *Scene*, adaptarla para que pinte en pantalla los tiles del videojuego. No es suficiente con poner una foto de fondo, pues habrá que hacer un estudio de las colisiones. El alumno puede, si quiere, hacer uso de las funciones *GlTexCoord* y *GlVertex2D*.
2. El videojuego debe respetar una estructura básica y lógica en:
   Init
   Update
   Logic
   Sound
   Draw

3. Dibujo de los *sprites*: tanto del Comecocos como de los fantasmas. Ambos se mueven por el mapa.
4. Colisiones: en caso de que el Comecocos toque una pared, deja de avanzar. Si se chocan fantasma y Comecocos, y este no tiene inmunidad, muere.
5. El videojuego consta de música.
6. Hay un menú básico, y si termina el juego, vuelve a él. Se da la posibilidad de salir del juego pulsando, por ejemplo, la tecla "ESC".

.
Al alumno se le proporciona un código base, y sólo tendrá que modificarlo. Antes será tratado ampliamente en clase. Todos estos trabajos, forman parte de mi aportación personal programando videojuegos.

Procedimientos e instrumentos

Asistencia (más del 80% obligatoria).
Realización del videojuego Comecocos.

Procedimiento	Asistencia	Realización del videojuego
Peso	20%	80%
Instrumento	Debe ser superior al 80% para que el alumno apruebe esta UD.	Hacer un juego sencillo Comecocos, proporcionando al alumno la función *Scene* casi hecha, para que sólo tenga que adaptar y jugar un poco con los *sprites* y las *tiles* y algunas funciones básicas de OpenGL.
Momento	Durante la UD	Durante todo el curso, más si quiere hacerlo en casa.

Para obtener la calificacion de esta materia se seguirá el siguiente proceso:

a) Asistencia: el alumno debe asistir a más del 80% de las clases. Ya que se facilitan tantas herramientas a los alumnos, es de proceder que hayan venido a clase lo suficiente. El porcentaje de esta parte suma 2 puntos al total. Si es una asistencia del 100% obtiene 2 puntos, y si va bajando, se resta con una regla de tres a esos 2 puntos.

b) Parte practica: como se ha expuesto antes, el alumno debe realizar el juego que se le ha pedido. En clase, puede consultar tanto al profesor como a otros alumnos. Si no lo llega a hacer, en casa puede terminarlo.

Rúbrica detallada del videojuego Comecocos.

Indicador	*Excelente* (10 puntos)	*Muy bien* (8 puntos)	*Bien* (6 puntos)	*Regular* (5 puntos)	*Mal* (0 puntos)
Asimilación y análisis del problema.	Las secciones de código están perfectamente definidas y podemos discernir claramente entre *Init,*	El código está ordenado, pero no diferencia todas las distintas secciones de código del videojuego.	El código es algo confuso, pero se pueden diferenciar las secciones clave de un videojuego. Hace uso de	No está bien organizado. Algunas secciones clave están incluídas en otras. No hay ni colisiones ni IA	No se entiende nada. No hay un orden en las funciones. El código está sucio. No hay ni

	Update, Draw, IA, Logic y Sound. Hace uso de colisiones e IA.	Hace uso de colisiones e IA.	colisiones pero no de IA.		colisiones ni IA.
Desarrollo de la solución	Buen uso y adaptación de la función *Scene* y *Draw*. (hace uso de *GltexCoord* y *GlVertex*). El Comecocos y los fantasmas están programados. Se ha programado así la inmunidad y la muerte de los sprites (bien Comecocos, bien fantasmas).	Función *Scene* y *Draw* bien adaptadas aunque algo difusas. (hace uso de *GltexCoord* y *GlVertex*). *Pacman* y fantasmas programados. Se ha programado así la inmunidad y la muerte de los sprites (bien Comecocos, bien fantasmas).	Función *Scene* completa pero faltan enemigos en pantalla. Dibuja el Comecocos, pero faltan elementos como la inmunidad. (no hace uso de *GltexCoord* y *GlVertex*).	Función *Scene* no muy clara, pero al menos pinta por pantalla. No hay enemigos ni *Pacman*.	No ha adaptado la función *Scene*, por tanto no ha comprendido lo que se pedía en este videojuego.
Interacción con los jugadores	Buen uso del teclado. Además permite editar las teclas. El juego va fluido, por lo tanto, se ha usado sincronización. Hay *Game Over* y vuelva al menú (que está programado).	Uso correcto de las teclas. No permite editarlas pero son sencillas. Hace uso de sincronización por lo que el juego va fluido. Hay *Game Over* y vuelva al menú (que está programado).	Mala elección de las teclas, pero el juego va fluído, por lo que se ha hecho buen uso de la sincronización. Hay *Game Over* y vuelva al menú (que está programado).	Buen uso de las teclas, pero no hay sincronización, por lo que no nos permite que nuestro Comecocos tenga portabilidad. Hay Menú o *Game Over*, pero no los dos	No se puede manejar el juego, pues va demasiado rápido o demasiado lento. No ha hecho uso de sincronización. No tiene ni menú ni *Game Over*.
Compilación y ejecución	El juego va fluído 100% y la compilación y ejecución es rápida. El juego, pues, es portable.	El juego va fluido al 100% y la ejecución y compilación da algunos *warnings*. Sin embargo, es portable.	El juego va a saltos, por lo que algún error habrá en la sincronización. Sin embargo, se puede jugar.	El juego tarda mucho en compilar ejecutar, y cuando llega el momento, va o bien demasiado rápido o lento. No se ha usado la sincronización.	El juego no compila y no se ejecuta.

Examen y rúbrica Septiembre.

La **recuperación** de la UD, está dirigida a aquellos alumnos que no hayan alcanzado los objetivos mínimos y constará de:

-Examen escrito: se pedirá la realización de un apartado breve de alguno de los videojuegos que se hayan explicado, y el alumno desarrollará la solución en papel. Vale el uso de pseudocódigo, pero debe ser claro. Se usará la rúbrica detallada del examen de Septiembre.
Dicha recuperación se realizará:
- En septiembre, en la convocatoria extraordinaria

La pérdida de una evaluación supondrá que el alumno/a deberá presentarse a la siguiente evaluación posible y su calificación en la UD será la que pertenezca a esta parte de la evaluación de la asignatura.

La rúbrica de Septiembre no es tan detallada, pues el ejercicio se hace en papel. Así, sólo tendremos algunas secciones para evaluar.

Antes, podemos poner un par de ejemplos de examen de Septiembre:

"Escribe el código que nos dibuja un Sprite en pantalla, y haz que se mueva a izquierda o derecha. No hace falta hacer un estudio de colisiones. Elección del videojuego y Sprite libre".

"Haz un esbozo de la sección de código '*Scene*' que permitirá dibujar un mapa en pantalla. Elección del videojuego y las *tiles* personalizadas".

Indicador	*Excelente* (10 puntos)	*Muy bien* (8 puntos)	*Bien* (6 puntos)	*Regular* (5 puntos)	*Mal* (0 puntos)
Desarrollo de la solución pedida.	El código es correcto al 100%	Código correcto con algunos errores.	Código no tan claro, pero el alumno comprende el problema.	Código difícil de entender, pero el alumno comprende el problema.	Mala codificación. El alumno no comprende el problema.

4. Atención a la diversidad

Se atribuye el término de Necesidades Educativas Especiales (NEE) a aquellos alumnos/as que presentan mayores dificultades que el resto de su mismo grupo y nivel educativo a la hora de alcanzar, por vía ordinaria, los objetivos y contenidos curriculares propuesto. Se engloba dentro del término de NEE a

tres tipos de colectivos:

• Alumnos con incorporación tardía y con dificultades de integración.
• Alumnos superdotados intelectualmente.
• Alumnos con necesidades educativas que presentan discapacidades físicas, psíquicas, sensoriales o graves trastornos de personalidad o conducta.

Atendiendo a lo dispuesto en el RD 1538/2006, los principios de actuación con estos alumnos son la no discriminación y la normalización educativa, a fin de lograr la igualdad de oportunidades para todos.

Sin embargo, y es algo que debemos mejorar, en los ciclos formativos de grado medio y grado superior, estas necesidades especiales no se tienen en cuenta, por lo que un alumno con discapacidad no podrá optar a realizar estos cursos.

Pero en mi opinión, deberíamos darle alguna sección de los trabajos, donde el alumno pueda ser útil, realizando tareas que no sean un impedimento por su discapacidad.

5. Conclusiones

La impresión que saco de esta unidad didáctica es muy positiva, pues a los alumnos les ha encantado por el tema de los videojuegos, que es algo que quizás no sea tan fácil de programar, y dándole unas pautas se han sentido capaces y quizás por su propia cuenta sigan programándolos en sus ratos libres, o si bien desean dedicarse a esto profesionalmente.

Además, para que no estuvieran presionados, se les pide solamente asistir a clase y realizar un juego sencillo. Damos por hecho que al haberles explicado el videojuego que tiene un Scene sencillo (el Bubble Burst), en el juego del Comecocos el mapa y los sprites son semejantes, sólo que cambiando las tiles y los sprites prácticamente. Bien es cierto que las colisiones no serán fáciles de programar, pero para eso está el profesor, para tratar y ayudar en los puntos más complicados.

En el futuro, mi deseo es también, aparte de programar estos juegos en el ordenador, lo haga en teléfonos móviles, aparte de aprender bien Unity, que es el futuro de los juegos en 3D.

Sin más, muchas gracias por haber leído este proyecto.

6. Bibliografía

20 minutos, L. 2.->. (2014). 10 innovadores periféricos que están por lanzarse. *Listas 20 minutos*.

Arnau, V. (2011). Ampliación de Estructura de Computadores. Curso 2010-11. Periféricos. 3º de Ingeniería Informática. *https://www.uv.es/varnau/AEC_2011-12.htm*.

Delgado., P. (2014). Un repaso a las tecnologías de almacenamiento actuales.

Dona, D. (2011). *Dispositivos de almacenamiento.* Libro de instituto. Editorial desconocida.

El grupo informático. (2018). Soportes físicos o la nube, ¿cuál es el almacenamiento del futuro? *El grupo informático*.

Gomila, L. (2019). Simple and Fast Multimedia Library https://www.sfml-dev.org/index.php.

ORDEN de 16 de junio de 2011, p. l. (2011).

Poulton, N. (2014). *Data Storage Networking.* Editorial Sybex.

Raya, L. (2012). *El mundo de las tarjetas gráficas. https://www.acta.es/medios/articulos/informatica_y_computacion/050031.pdf.*

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. (2010).

Rodríguez, R. (2010). GPUs, comparación entre Nvidia y Ati. *Autores cientifico-técnicos y académicos*, 16.

Tenología Informática, W. h.-i.-d.-a.-i. (2018). *Dispositivos de almacenamiento de información.*

Todorov, R. (2013). *Computer Peripherals (Revised Second Edition).*