



Columna de Desarrollo de Video Juegos

Por Roberto S. Hernández J.

Introducción

En 1984, aproximadamente, cuando apenas era un joven de 13 años, recibí un regalo de mi padre (QEPD) que cambiaría por completo mi vida. Una preciosa y flamante computadora Commodore 16. Y realmente que me cambio la vida, pues aquel incipiente joven que tenía en mente ser algún día un arquitecto, empezó a voltear la vista hacia el fascinante mundo de las computadoras. Algunos años después, mi padre me consiguió un equipo Commodore 64, y ahí fue donde realmente sucumbí al olor de los circuitos integrados y me entregue de lleno a esta profesión. Y en honor a la verdad, que no me arrepiento. En aquellos años, lejos de pertenecer al tipo de desarrolladores que se desvivían por construir el mejor sistema administrativo, la mejor nómina, o el mejor sistema de facturación, tuve la suerte de conocer este fascinante mundo por el lado más glamoroso y atractivo. Los videojuegos en computadora.

Una de las primeras cosas que llamaron mi atención, fue sin duda alguna, descubrir la magia que se utilizaba para hacer que los personajes de los juegos se desplazaran sobre la pantalla de mi televisor. ¿Cómo es que me responden a lo que les ordeno? Realmente estas preguntas me hicieron pasar noches en vela, pensando que tipo de “súper hombre” con “súper poderes” había tenido la gracia de poder realizar algo tan increíble. Yo tenía que saber como hacer lo mismo, así que puse manos a la obra. En aquellos años, era impensable hacer un desarrollo de un videojuego que se pudiera ejecutar dignamente utilizando un lenguaje dirigido a las masas, como Basic. Si quería comprender que estaba sucediendo dentro de ese videojuego que tanto me llamaba la atención, tenía forzosamente que aprender un lenguaje bastante críptico. Lenguaje Ensamblador. Otra de las cosas con las que me tope en aquel entonces, fueron las carencias tecnológicas de los equipos. La Commodore 64 tenía un reloj de 1.2 Mhz. y solo 64 Kb. de memoria (Si, si leyeron bien. Esos equipos existían y funcionaban mucho mejor que muchos equipos que conozco actuales). Este tipo de retos hicieron a la larga, que apreciara el valor de ahorrarme un ciclo de reloj en la ejecución de mi aplicación, o que respirara profundamente y con satisfacción cuando podía reescribir alguna función que me ahorrara algunos bytes de preciosa memoria.

Han pasado muchos años desde aquellos tiempos, y muchas cosas han cambiado. Ahora la velocidad de los procesadores se mide en gigahertz. La memoria se mide en megabytes o en gigabytes. La gente ya no se preocupa por ocupar varios cientos de ciclos de reloj más, o bien desperdiciar memoria que pudiera ser cuidada. Existe una gama mucho más amplia de lenguajes de programación con características especiales para hacer desarrollos mucho más rápido y casi sin saber programar. Pero lo que también es cierto, es que aun no he visto por mas que he buscado, recursos que le enseñen a todos los principiantes que estén interesados, cuales son los principios básicos de desarrollos de videojuegos. He visitado paginas

y leído varios libros, en donde se cree que todo aquel que desarrollara un juego ya es un Guru, y te hablan con mucha naturalidad de temas tan complejos como DirectX, GDI, o Mensajes de Windows. No enseñan a la gente desde el inicio hasta el final. Nadie te dice que desarrollar un juego es uno de los tipos de desarrollo más difíciles de concretar.

La finalidad de esta columna, es guiarlos a través del fascinante mundo del desarrollo de video juegos, explicando paso a paso lo que necesitan saber y como se puede hacer. Existen muchas cosas que debemos definir antes de sentarnos a escribir código, y aquí trataré de explicarlas de manera que ustedes tengan la sensibilización de todo el proceso. Y como creo que ya fue demasiada historia e introducción, dispongámonos a comenzar. Bienvenidos y espero que disfruten este paseo por el maravilloso mundo del desarrollo de videojuegos, tanto o más que yo.

Las primeras preguntas.

En mi experiencia como docente, y como resultado de algunas de las charlas sobre desarrollo de video juegos que he dictado, inmediatamente se identifican 2 preguntas que me han hecho en repetidas ocasiones, y que me gustaría contestar de una buena vez, para que todos estemos enterados por igual de las respuestas a tan sonadas preguntas.

Pregunta # 1: Quiero programar un video juego. ¿Qué necesito saber?

Creo que la respuesta a esta pregunta es bastante abierta, pero vamos a tratar de darle una respuesta satisfactoria. Como primer punto, necesitamos definir *Que tipo de videojuego* vamos a desarrollar. De acuerdo a esto podemos deducir las habilidades y conocimientos que debemos tener para llevar a cabo dicho desarrollo. Por esta razón, hablemos un poco de los tipos de videojuegos que existen para estar un poco mas en sintonía.

Estrategia: Como el Starcraft o el Age of Empires. Este tipo de juego demanda mucha programación del lado de la Inteligencia Artificial (IA), y los gráficos no son tan importantes. El sonido tampoco es un factor determinante.

Shooter's: En este tipo de juegos existes 2 variantes. Los llamados "de primera persona" (como el DOOM o el Quake) donde el jugador tiene la sensación de ser parte de la acción, y los llamados "de tercera persona" (Como el Famoso Tomb Raider), en el cual visualizamos completamente a un personaje que es una representación del jugador dentro del juego. Para este tipo de juegos, los gráficos en una parte crucial, pues se necesita de algoritmos realmente eficaces para dar una buena impresión al jugador.

Arcade: Como el Tetris, Pac Man, Arkanoid, etc. En este tipo de juegos, la finalidad es conseguir el máximo número de puntos posibles, a través de niveles que aumentan la dificultad para seguir jugando. Una variante de este tipo de juegos es también aquellos llamados *MataMarcianos* como el famoso Space Invaders.

RPG: Por ejemplo el Ultima. Este tipo de juegos es una mezcla de acción, estrategia, combinados con mucho juego de aventuras basado en texto. Este tipo de juegos, al igual que los tipos Shooters, tienen la peculiaridad de tener un Final del juego.

SideScrollers: Como por ejemplo Mario Bros. Este tipo de juego también tiene un final, y lo difícil es programar el movimiento suave de los gráficos y la interacción del personaje principal con el resto del entorno. Existe una variante de este tipo de juegos, donde solamente se toma en cuenta la interacción de los personajes con su medio ambiente sin deslizar la pantalla. A este tipo de juegos se les llama "Juego de Niveles".

Simulación: Este tipo de juegos, tratan de representar acciones reales dentro de nuestra PC. Pueden ir desde conducir un auto (Need for Speed, Test Drive) hasta conducir un Avión (Fligh Simulator). El grado de complejidad en este tipo de juegos es demasiado alto. Se tiene que tener en consideración muchísimas cosas y la física y las matemáticas aparecen por todos lados.

Escritorio: Este tipo de juegos son los utilizados por la mayoría de la gente cuando esta ociosa frente a una computadora. Los más conocidos son los juegos de cartas (Freecell, Solitario), aunque existen algunas otras variantes (como por ejemplo el busca minas o el JawBreaker). Para desarrollar este tipo de juegos, solamente es necesario conocer las reglas del juego y poner manos a la obra, pues los gráficos normalmente son sencillos y carecen de algún tipo de música o sonidos.

En realidad son pocos el número de conocimientos necesarios, pero dependiendo de la complejidad del tipo de juego seleccionado, es que tan profundo tenemos que conocer cada uno de estos puntos:

- Conocer un lenguaje de programación (y claro, tener lógica para programar).
- Conocimientos de matemáticas y física.
- Conocimientos de manejo de gráficos
- Conocimientos de manejo de sonido.

Pregunta # 2: ¿Cuál lenguaje debo elegir?

Normalmente, cualquier lenguaje de programación debe servir para el propósito. Como les mencione en un principio, el Lenguaje Ensamblador era lo único que podía ser utilizado en los primeros tiempos. Después la gente empezó a utilizar C y C++, por su facilidad para obtener el control de los dispositivos de las computadoras. En estos momentos, muchos de los juegos comerciales están escritos en C++. Los juegos que normalmente tenemos en nuestros teléfonos celulares están escritos en Java. Como la finalidad de esta columna, además de enseñarlos a desarrollar videojuegos, es que estos se puedan ejecutar dentro de la plataforma .NET, tenemos 2 opciones. Visual Basic o C#. Realmente aquí, la elección de VB o C# ya no tiene relevancia pues a fin de cuentas los 2 generan código intermedio (MSIL). Para hacer más divertida esta columna, utilizaremos los 2 lenguajes, y hacer las conversiones de uno a otro servirá como ejercicio para todos.

Habiendo respondido esas preguntas, tenemos argumentos suficientes para continuar. Pero todavía estamos algo retirados de la parte donde empezaremos a escribir código. Quisiera compartir algunos consejos con todos ustedes antes de iniciar en forma la construcción de nuestro primer video juego. Quizás al leer estas recomendaciones, cambiaremos algunas de las decisiones que hemos tomado hasta ahora.

Consejo # 1: Empieza por el principio.

Se que hay muchos de ustedes que están ansiando escribir el próximo DOOM, o desarrollar el programa que será el próximo hit a nivel mundial. La verdad es que tenemos que ser realistas y empezar por el principio. Debemos tomar en cuenta, que los juegos que impresionan (como el Doom, por ejemplo), son el resultado del trabajo de un equipo de gente, especialista cada quien en su área, dedicados al 100% a esta tarea y que ya tienen muchísima experiencia. A saber, un equipo de desarrollo en una empresa de software de video juegos, normalmente se compone de:

- Programadores (Coders).
- Diseñadores gráficos y artistas.
- Guionistas.
- Diseñadores de niveles y personajes
- Músicos
- Sonoristas

Esto sin contar al personal administrativo y todos aquellos que están dedicados a la parte de promoción y venta del juego, a la gente que hace los manuales, a la gente que actúa como "Beta Tester", etc.

Un buen comienzo, es iniciar con un desarrollo básico e irlo creciendo. Esto nos dará seguridad y más conocimiento conforme avanzamos en la complejidad de nuestros proyectos. A esto se le llama "Camino crítico de aprendizaje". Un ejemplo, que bien podríamos tomar como inicio para nuestra columna, es el siguiente camino crítico de aprendizaje.

Nivel	Juego	Descripción
1	Juego de Bloques tipo Tetris (Fig. 1).	<ul style="list-style-type: none">• Contiene todos los elementos que aparecen prácticamente en todo tipo de juegos.• Tiene un "Game Loop" (Estructura general del juego).• No se necesita ser un artista para dibujar cuadros.• Lee y Procesa entradas, tiene animación y sonido.



Fig. 1. Tetris.

Nivel	Juego	Descripción
2	Juego Tipo Pong o Arkanoid. (Fig. 2).	Contiene los mismo elementos que el Tetris, pero incorpora: <ul style="list-style-type: none"> • Detección de colisiones. • Física de reflexión, velocidad y aceleración. • Manejo de niveles, por lo que ahora se tiene que aprender a guardar y cargar información desde algún tipo de recurso.

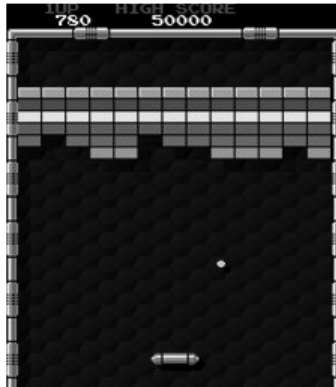


Fig. 2. Arkanoid.

Nivel	Juego	Descripción
3	Arcade Tipo Pac Man. (Fig. 3).	<ul style="list-style-type: none"> • Animación detallada de los caracteres. • Contiene un módulo de inteligencia artificial más o menos avanzado, por que por ejemplo, en el juego original, ¡los 4 fantasmas “pensaban” diferente!

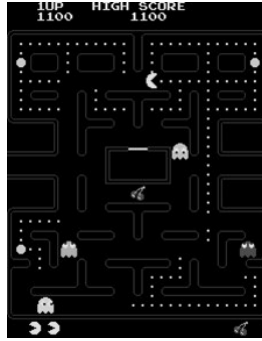


Fig. 3. Pacman.

Nivel	Juego	Descripción
4	SideScroller (Tipo Mario Bros.) (Fig. 4).	<ul style="list-style-type: none"> • Animación e interacción de los caracteres. • Multiplataforma. • Física de saltos (tiro parabólico), aceleración, gravedad, disparos, etc. • Screen Clipping. • ¡Tiene un Final!



Fig. 4. Mario Bros.

Después del 4 nivel, se han aprendido y aplicado una buena cantidad de conocimientos que nos dará la posibilidad de entrar a cosas más complejas como un Shooters de primera persona, etc.

Por esta razón, y siguiendo este mismo camino crítico de aprendizaje, nuestro primer desarrollo será un “clon” de Tetris, que llamaremos “Tetrak”.

Consejo # 2: Termina lo que empieces.

Esta recomendación es muy importante. Es necesario terminar un juego antes de iniciar el desarrollo de otros. Si no seguimos esta regla de oro, terminaremos con varios juegos TODOS desarrollados a la mitad y sin funcionalidad. Tenemos que acostumbrarnos a cumplir el objetivo que nos hemos propuesto y para esto puede servir muy bien el siguiente consejo.

Consejo # 3: Genera un documento de expectativas del juego.

Cuando se inicia en el mundo de desarrollo de videojuegos, es difícil conceptualizar un juego desde un inicio y siempre se harán cambios y agregados en el proceso de desarrollo. Pero conforme avancemos en nuestro aprendizaje será necesario utilizar una herramienta que nos ahorrará muchos dolores de cabeza y muchas líneas de código desperdiciadas. Este es el llamado “Documento de expectativas del juego”. Este documento definirá los alcances que

tenemos pensado para nuestro juego. Prácticamente es el análisis de nuestro desarrollo. Debemos de utilizar algo de tiempo para plasmar en este documento los requerimientos y expectativas de mi juego. Cuales deben ser los esquemas de puntaje, Colores, Niveles, etc. etc. Este documento cobra mayor importancia si lo visualizamos de la siguiente manera. Imagínense que están desarrollando un juego tipo Mario Bros. y han pasado casi todo el fin de semana y varios cientos de líneas escritas para lograr que los personajes se muevan. Y el lunes se les ocurre dotar al personaje con la capacidad de volar. Es casi un hecho que TODO lo que hicieron el fin de semana se vaya a la basura o tenga que resentir un cambio radical. Esto no hubiera pasado si desde un principio hubiéramos decidido hacer que nuestro personaje volara. Con esto, tampoco quiero decir que si en un momento a medio desarrollo se nos ocurre un buen agregado que no habíamos considerado, no lo pongamos en el juego. Pero les aseguro que divagar por un momento sobre lo que quiero de mi juego, como se debe de controlar, y ese tipo de cosas nos van a evitar muchos dolores de cabeza y dejaremos nuestro código más limpio.

Consejo # 4: Ten un libro de referencia del lenguaje que estés usando a la mano.

Esto permitirá resolver dudas que no son referentes al video juego en si, sino al lenguaje de programación que hayamos elegido.

Consejo # 5: No quieras ser “Exacto y Correcto” a la primera.

Mucha gente se pasa la vida tratando de hacer una función lo mejor posible a la primera vez, y para esto gasta mucho tiempo. Además, la mayoría de las veces terminan con un código inentendible hasta para ellos mismos. La idea que tengo sobre como enseñar estas técnicas de desarrollo es la siguiente. Empezaremos a desarrollar las funciones o procedimientos que se necesiten en nuestros videojuegos para que a primera impresión sean funcionales y entendibles. Después podremos empezar a optimizar dichos procedimientos de manera de terminar con un código limpio, entendible y lo mas funcional posible.

Consejo # 6: Define el tipo de estructura general de tu juego.

La estructura general del juego, dependerá del tipo de desarrollo que vayamos a utilizar. Normalmente los juegos siguen 2 grandes vertientes en lo que a estructura general del juego se refiere.

Loop Infinito.- Este tipo de estructura es que se utilizo en un principio por casi todos los juegos, y aún ahora es utilizada en muchos juegos de complejidad media a baja. La ventaja de utilizar este tipo de estructura, es que tenemos el control de todo el programa, pero es más difícil de programar. Normalmente todos los componentes tienen que ser desarrollados por nosotros (Timers, Eventos, Manejo de Teclas, etc.). Para nuestro primer desarrollo (Tetrak) utilizaremos este tipo de estructura. (Fig. 5).

```
Do While .t.  
    LeerEntradas()  
    ProcesarEntradas()  
    ActualizarDatosenMemoria()  
    ActualizarGraficos()  
    ActualizarSonido()  
Loop
```

Fig. 5. Loop Infinito.

Engine.- Otra forma de realizar los ciclos de los juegos, es la utilización de un “Engine” y valernos de herramientas programadas (por nosotros o no) que le den datos a este “Engine” y a su vez reciban instrucciones de ese “Engine”. Este tipo de estructura es usada por programas

más complejos. Y algunos (Como el DOOM) tienen “Sub-Engines” que controlan funciones particulares del juego, por ejemplo “Engine” de gráficos, “Engine” de sonido, etc. (Fig. 6).

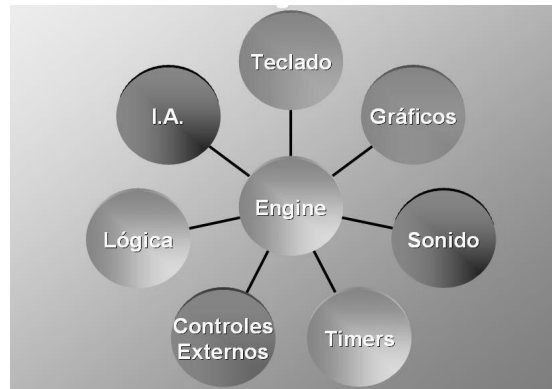


Fig. 6. Engine

Consejo # 7: Enriquece tu juego.

Por último pero no menos importante, es que además de que el juego sea “jugable”, debería de incluir una serie de características que le ayudaran a “lucir”. A esto último se le llama el “Polish” o “Acabado” del juego. Entre las características que deberíamos considerar están:

- Pantalla de Inicio y término.
- Menú de opciones (si es aplicable).
- Instrucciones de cómo jugar y como iniciar el juego.
- Pantallas introductorias.
- Pantallas de recompensas (si es aplicable).
- Marcador, indicadores de juego y puntajes más altos (High score).
- Demo Mode.
- Editores que permitan la expansibilidad del juego (por ejemplo editores de niveles, generador de caracteres, etc.).

Conclusión.

Para esta primera entrega y antes de comenzar en forma con el desarrollo de nuestro primer videojuego, han leído información que a mi parecer, ha sido de mucha ayuda para desmitificar todo lo referente al desarrollo de videojuegos. Ahora ya tenemos las bases en firme de como construiremos nuestro primer videojuego y he dado algunas pistas acerca de que usaremos para desarrollarlo. También es importante hacer notar, que necesitamos tener el Visual Studio .NET para iniciar el desarrollo. En la próxima entrega iniciaremos el desarrollo del Tetrak y explicaremos a fondo cada uno de los conceptos que utilizaremos. Agradeceré todos sus comentarios e ideas en mi correo electrónico (rshj@itesm.mx). Así que nos vemos en la próxima entrega. Ah! Y por último: Gracias Papá.

Acerca del Autor



Roberto S. Hernández Juárez, vive en Monterrey, Nuevo León, México. Es Licenciado en Ciencias Computacionales de la Facultad de Ciencias Físico Matemáticas de la UANL. Gerente de Sistemas y Telecomunicaciones del Grupo Ladrillera Mecanizada. Es profesor de cátedra de materias relacionadas con .NET en el Tecnológico de Monterrey (ITESM). Es Microsoft Certified Professional. Además desde hace 2 años es Microsoft MVP (Most Valuable Professional) en el área de Visual Basic. Líder y fundador de la ComunidadNETMonterrey y miembro del buró de directores de INETA para Latam. Puedes contactarlo en su dirección electrónica rshj@itesm.mx ó bien en robertoh@ladrillramecanizada.com.