

Quiero ser como Will (Wright)

8 de septiembre de 2007

Escrito por Miguel Santirso González

<http://www.redroomsoftware.com/tirsoweb/>

Capítulo 1: La primera vez mola.

1

—Llegas tarde —le dije al verla bajar del autobús.

Como no tenía nada que hacer y hacía un día agradable para estar al aire, había ido a esperarla a la parada del autobús. Además, tenía bastantes ganas de empezar; mucha gente me pide de vez en cuando que les enseñe a hacer juegos, pero muy pocas muestran tantas ganas como había mostrado Olaya durante la semana pasada, cuando intentaba convencerme para quedar un día y ayudarle a empezar en esto del desarrollo de videojuegos.

—Bueno, por favor. Llego cinco minutos tarde —dijo mientras reía—. Seguro que tú llevas ya aquí desde las ocho, empollándote algún manual de programación para impresionarme.

Los dos nos reímos y fuimos directamente a la sala B-2 de la escuela donde estudiamos informática desde hace ya tres años. La sala estaba casi vacía y pudimos sentarnos juntos al fondo, donde no nos molestasen demasiado. Ella se sentó delante del ordenador y yo me puse a su lado para ayudarla a crear su primer videojuego.

— ¿Linux o Windows? — me preguntó al ver el menú del GRUB después de arrancar el ordenador.

—Para lo que vamos a hacer ahora nos valdría cualquiera de los dos, pero como supongo que querrás seguir luego en tu casa, mejor arranca Windows.

Mientras arrancaba empecé a trazar un pequeño plan de trabajo, ya que no había preparado nada concreto (aunque más o menos ya tenía alguna idea de lo que íbamos a hacer). Lo normal cuando se empieza en algo de programación es hacer un “Hola Mundo” y en el mundo de los videojuegos eso suele traducirse a crear un PONG, así que había decidido empezar por ahí.

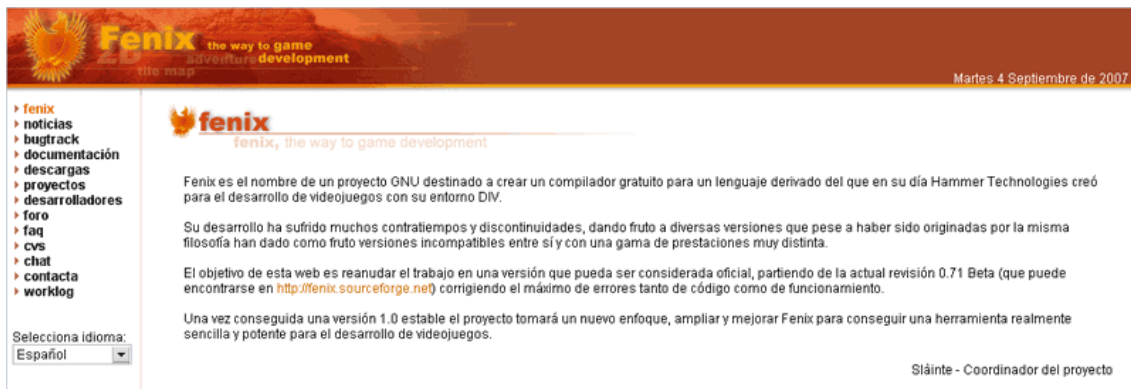
—Ya está. ¿Qué hago? —me preguntó en cuanto inició sesión.

—Bueno, si fuéramos a empezar de la forma habitual, tendrías que poner a descargar diversos programas tales como el Photoshop, el 3DSMax, el Cubase, el DirectX SDK, el XNA, el Adobe Flash CS3 y el Ogre Engine, pero creo que será mejor no empezar de la forma habitual.

— ¿No? Pues yo ya me había bajado el Photoshop y el 3DSMax por si acaso —dijo mientras se reía.

—No. Mejor he decidido que vamos a empezar de la forma correcta. Abre el Firefox anda. Y entra en <http://fenix.divsite.net/>.

Ilustración 1: Portada de la web de Fenix



— ¿Qué es esto?— Preguntó con cara extrañada.

—Bueno, es el lenguaje de programación que vamos a usar hoy. Es bastante antiguo y está un poquito abandonado pero tiene muchas ventajas, sobre todo para alguien que empieza.

—¿Por qué?

—Bueno, lo más evidente es que está orientado a videojuegos, y facilita muchas tareas propias del desarrollo de un videojuego. Además, los programas hechos en Fenix se pueden portar muy fácilmente a otros sistemas operativos y plataformas. ¡Se pueden portar incluso a consolas como Dreamcast o GP32!

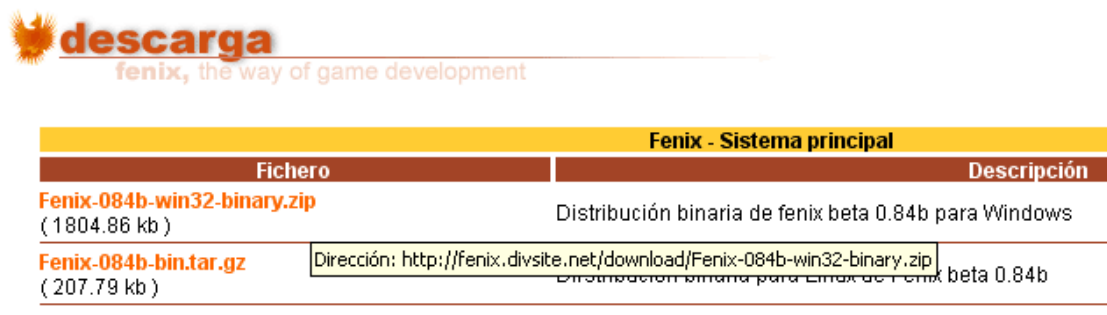
—Bueno, parece que está bien. Pero yo ya estoy acostumbrada a C++, ¿vamos a usar Fenix siempre?

—No, no. Esto es solo para el primer día. Es que para usar C++ hay que descargarse algunas librerías y es un poco más complicado de configurar todo para que funcione.

—Ah, bueno. Además siempre oí que C++ es el estándar en videojuegos.

—Y lo es, pero de momento esto es lo mejor. ¡Bueno venga, no me distraigas! Vete a [descargas](#) y elige la primera ([Fenix-084b-win32-binary.zip](#)). Si en el futuro saliera una versión más moderna, tendrías que bajar esa y si quisieras compilar el juego para otra plataforma distinta de Windows, tendrías que descargar otra distribución para esa plataforma.

Ilustración 2: Descargar los binarios de Fenix



—Miguel, atiéndeme anda que ya está descargado.

—Ah, vale. Bueno pues descomprímelo en un sitio que no se te olvide y abre esa carpeta que te voy a explicar qué es cada cosa.

—Ya.

—Vale. Ahí ves dos ejecutables. El *FXC es el Compilador de Fenix* (que sirve para compilar, lógicamente) y el *FXI es el Interpretador de Fenix* (que sirve para ejecutar los juegos). Hace falta un intérprete porque Fenix es un lenguaje interpretado, aunque eso carezca de importancia a este nivel.

—Ajá, el fxc es como el gcc que usamos en clase para c++. Pero no entiendo muy bien qué es eso de usar un intérprete para ejecutar los juegos. ¿Y esos archivos dll que hay por ahí?

—Bueno, el intérprete es necesario porque, a diferencia del gcc por ejemplo, el FXC no genera un ejecutable propiamente dicho. Lo que hace es generar un archivo DCB que luego es interpretado por el intérprete de fénix. Las dll que ves que empiezan por “SDL” son las librerías del SDL, una librería para gráficos, y la de zlib es para asuntos de compresión, aunque eso ya lo veremos otro día.

—Entendido. ¿Vamos a empezar a programar ya?

—Sí. Te estoy notando un poco ansiosa, así que vamos a empezar de una vez. *Crea una carpeta nueva con el nombre “día 1” en el escritorio (que usaremos como directorio de trabajo) y, para no complicarnos, copia en ella el compilador de fénix (FXC.exe) y el intérprete (FXI.exe). Además necesitaremos las otras librerías (SDL.dll, SDL_Mixer.dll y zlib1.dll) así que cópialas también.*

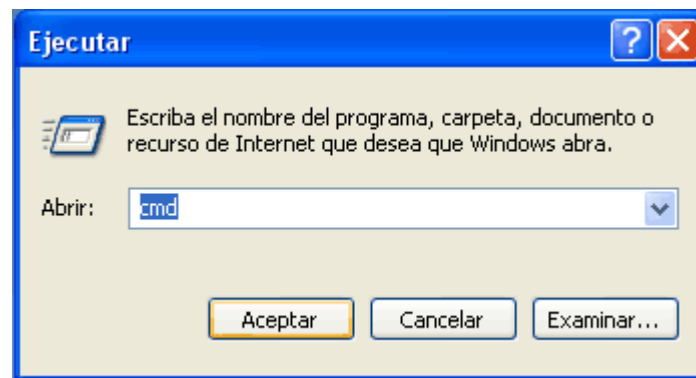
—Dices que lo hacemos así “para no complicarnos”. ¿A qué te refieres?

—Hombre, es que copiar el compilador y las librerías a la carpeta de trabajo no es la mejor forma de trabajar, pero de momento vamos a hacerlo así para no complicarnos. Más adelante te enseñaré cuál es la forma correcta de hacerlo.

—Perfecto entonces. ¿Abro el Visual Studio?

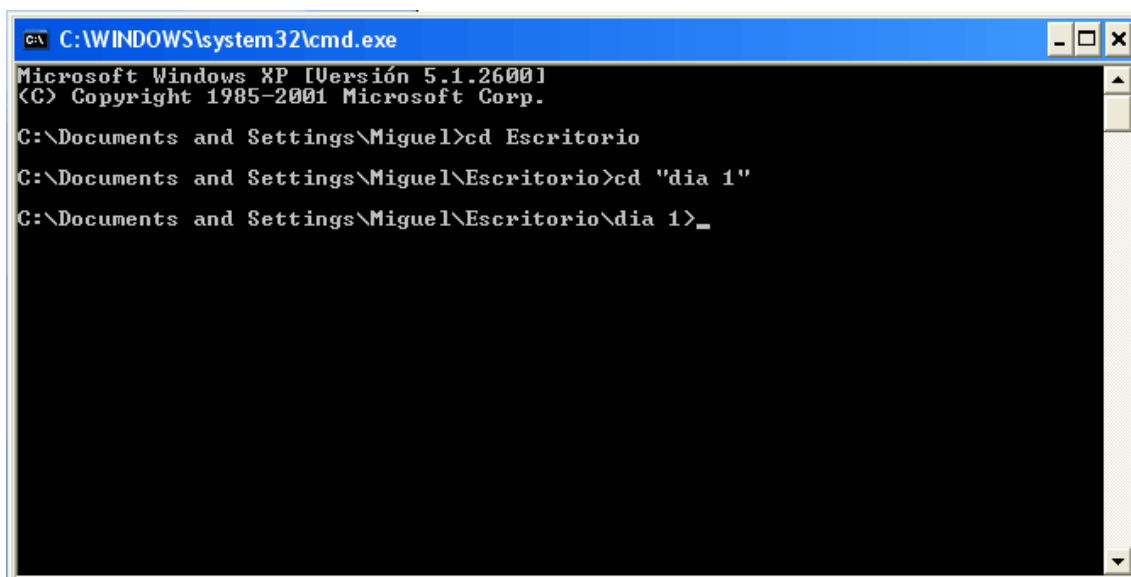
—No hombre —respondo mientras me río—, ya te dije que hoy vamos a empezar por lo básico. Sí, ya sé que te dije que lo vamos a usar, pero hoy todavía no. *A ver, deja abierta la carpeta que acabas de crear en el escritorio y abre una línea de comandos (inicio->ejecutar->cmd).*

Ilustración 3: Abrir una línea de comandos



—Vale —le dije al ver que ya está abierta la línea de comandos—, no te asustes. Ya sé que no estás acostumbrada a usar la línea de comandos, pero no vamos a hacer nada raro. *Navega hasta la carpeta de trabajo (cd c:\Documents and Settings\Olaya\Escritorio\dia 1).*

Ilustración 4: Navegar hasta la carpeta de trabajo



—De acuerdo, *ahora crea un archivo de texto de nombre "mipong.prg" y ábrelo en segundo plano con el bloc de notas (escribe "notepad mipong.prg &").*

Ilustración 5: Crear un archivo de texto y abrirlo con el notepad.



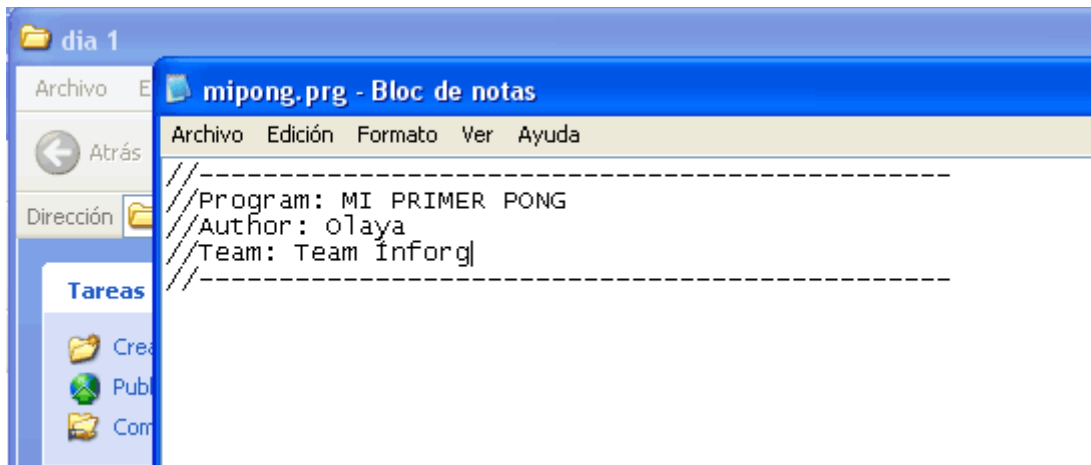
—Bueno, de esto deduzco que vamos a hacer un PONG, pero no entiendo una cosa... ¿iBloc de notas!? No me vaciles anda.

—No te estoy vacilando, Oli —le respondo entre risas—. Para lo que vamos a hacer hoy no hace falta nada más. Bueno, quizás abramos el Paint en algún momento.

—Por Dios...—responde al tiempo que se echa las manos a la cabeza.

—*Bien, ahora que tenemos abierto nuestro programa, vamos a empezar a programar en él. Escribe lo siguiente:*

Ilustración 6: Las primeras líneas de código de cualquier juego



—Ya. ¿Qué más?

—Bien, veo que has mejorado en mecanografía —risas—. ¿No ves como no era tan difícil? Ahora compívalo y envíasele a tu novio.

—Ehhh... No te rías de mí. Además, ya sabes que Isaac no es mi novio. No soy lo bastante buena para él. Venga, déjate de coñas y vamos a programar un juego.

—Bueno vale, pero esas líneas son importantes a pesar de ser comentarios que el compilador no tendrá en cuenta. No todos los programadores novatos lo hacen, pero documentar el código de tus juegos (como el de cualquier programa) es importantísimo. Sirve para que otros puedan entenderlo o incluso para poder modificar un código que tú misma hayas escrito hace mucho tiempo. Es una buena costumbre escribir siempre un encabezamiento en el código fuente de tus programas para indicar, como mínimo, quién lo hizo y otros datos sobre el programa como la fecha de la última modificación o la versión.

—Vaaale. Oye, ¿sabes que estás siendo un poco pesado?

—Vale anda. Ahora *en serio, escribe esto que tengo aquí preparado*. Yo me voy a por un refresco, porque imagino que va para largo...

—Vete a la mierda —me responde mientras salgo a la máquina que hay donde las escaleras.

Ilustración 7: Esqueleto típico de un juego

```

mipong.prg - Bloc de notas
Archivo Edición Formato Ver Ayuda
//-----
//Program: MI PRIMER PONG
//Author: oLaya
//Team: Team Inforg
//-----

////////////////////////////////////
// Función inicial (aquí empieza todo)

Program MIPONG;
Global
  int RESOLUCION_X;
  int RESOLUCION_Y;
BEGIN
  // INICIALIZAR ASPECTOS GRÁFICOS

  set_title("Mi Primer Pong"); // Establece el título de la ventana del juego
  Full_screen = false; // No queremos que se ejecute a pantalla completa
  Graph_mode = mode_16bits; // El modo gráfico será de 16 bits
  set_mode (m640x480); // Resolución de 640x480
  set_fps(60, 60/10); // Establecemos los FPS que queremos

  RESOLUCION_X = 640; // Almacenamos en dos variables la resolución
  RESOLUCION_Y = 480; // para utilizarlo después

  // ESCRIBIR INFORMACIÓN SOBRE EL JUEGO EN PANTALLA
  write(0,10,10,0,"Mi Primer Pong, v. 0.01");
  write(0,10,20,0,"Team Inforg");
  write_int(0,300,10,1,&fps);

  // HEMOS TERMINADO DE INICIALIZAR, LANZAMOS EL BUCLE PRINCIPAL
  Controlador_Juego(); // Arrancamos el controlador del juego
END

////////////////////////////////////
// controlador del juego (aquí se controla el juego)

Process Controlador_Juego()
BEGIN
  Loop // Bucle principal

  If(key(_esc)) // si se pulsa escape
    exit(0,0); // salir del juego
  End

  Frame; // Refrescar pantalla

  End // del loop
END

```

Cuando volví de la máquina ya estaba terminando, así que esperé unos instantes y me preparé para continuar.

—¿Quieres un poco?—le dije a la vez que le hacía el gesto de ofrecerle mi CocaCola.

—No, gracias. Ya terminé de copiar eso. Parece bastante fácil, creo que he entendido la mayoría.

—Es normal, es un código muy legible, pero antes de explicártelo en detalle vamos a ver si funciona (seguro que no, nunca lo hace a la primera).

—Vale, hay que compilarlo supongo...

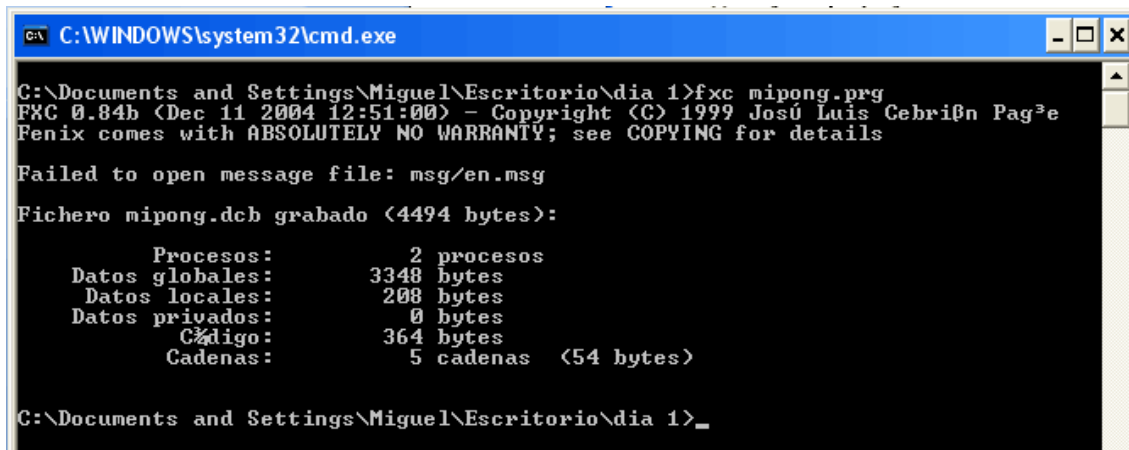
—Sí, *para compilarlo vete a la línea de comandos que ya tenías abierta y teclea lo siguiente:*

Ilustración 8: Orden para compilar el programa

```
C:\Documents and Settings\Miguel\Escritorio\dia 1>fxc mipong.prg_
```

Lo hace, y al pulsar INTRO ve aparecer el resultado de la compilación:

Ilustración 9: Resultado de la compilación



```
C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\Miguel\Escritorio\dia 1>fxc mipong.prg
FXC 0.84b (Dec 11 2004 12:51:00) - Copyright (C) 1999 Josu Luis Cebriñn Pag³e
Fenix comes with ABSOLUTELY NO WARRANTY; see COPYING for details
Failed to open message file: msg/en.msg
Fichero mipong.dcb grabado (4494 bytes):
    Procesos:          2 procesos
    Datos globales:    3348 bytes
    Datos locales:     208 bytes
    Datos privados:    0 bytes
    Código:           364 bytes
    Cadenas:           5 cadenas (54 bytes)
C:\Documents and Settings\Miguel\Escritorio\dia 1>_
```

—Genial. *No te has equivocado en nada. Si la compilación tuviera algún error, ahí se te indicaría el problema, lo corregirías en el código y volverías a compilar hasta eliminar todos los errores.* Fíjate en la carpeta de trabajo; se ha creado un nuevo fichero llamado “mipong.dcb”.

Ilustración 10: Archivo generado tras una compilación exitosa



—*Ese archivo contiene toda la información necesaria para que el intérprete pueda ejecutar tu juego. Para ejecutarlo escribe lo siguiente en la línea de comandos (“fxi mipong.dcb”):*

Ilustración 11: Instrucciones para ejecutar el juego

```
C:\Documents and Settings\Miguel\Escritorio\dia 1>fxi mipong.dcb_
```

Al pulsar INTRO, ve aparecer la ventana del “juego”.

Ilustración 12: Captura del "juego" funcionando.



—¡Et Voila! —le digo mientras me levanto y le hago una reverencia.

—¡Eh, esto mola! Llevo tres años en informática y es la primera vez que creo una ventana de Windows. Es increíble que esto me satisfaga tanto, pero estoy muy contenta —dice al tiempo que me sonrío.

—Lo sé; creo que todos sentimos lo mismo la primera vez, aunque yo era “algo” más joven cuando empecé—le digo mientras le devuelvo la sonrisa.

—Bueno, venga. ¡Quiero más!

—Bueno, bueno, querrás que te explique ese código supongo...

—Ah sí claro.

—Vale, echa un vistazo al código:

Ilustración 13: Palabras reservadas en el código de Fenix

```

mipong.prg - Bloc de notas
Archivo Edición Formato Ver Ayuda
//-----
//Program: MI PRIMER PONG
//Author: Olaya
//Team: Team inforg
//-----
////////////////////////////////////
// Función inicial (aquí empieza todo)

Program MIPONG;
Global
  int RESOLUCION_X;
  int RESOLUCION_Y;
BEGIN
  // INICIALIZAR ASPECTOS GRÁFICOS

  set_title("Mi Primer Pong"); // Establece el título de la ventana del juego
  Full_screen = false; // No queremos que se ejecute a pantalla completa
  Graph_mode = mode_16bits; // El modo gráfico será de 16 bits
  set_mode (m640x480); // Resolución de 640x480
  set_fps(60, 60/10); // Establecemos los FPS que queremos

  RESOLUCION_X = 640; // Almacenamos en dos variables la resolución
  RESOLUCION_Y = 480; // para utilizarlo después

  // ESCRIBIR INFORMACIÓN SOBRE EL JUEGO EN PANTALLA
  write(0,10,10,0,"Mi Primer Pong, v. 0.01");
  write(0,10,20,0,"Team Inforg");
  write_int(0,300,10,1,&fps);

  // HEMOS TERMINADO DE INICIALIZAR, LANZAMOS EL BUCLE PRINCIPAL
  Controlador_Juego(); // Arrancamos el controlador del juego
END

////////////////////////////////////
// Controlador del juego (aquí se controla el juego)

Process Controlador_Juego()
BEGIN
  Loop // Bucle principal

  If(key(_esc)) // si se pulsa escape
    exit(0,0); // salir del juego
  End

  Frame; // Refrescar pantalla

  End // del loop
END

```

—Bien, fíjate en esas partes que te estoy señalando. Todas esas son “palabras” reservadas del lenguaje de programación Fenix. Sirven para definir los diferentes ámbitos del código que escribas y es necesario entenderlos bien, aunque tampoco encierran mucho misterio...

—Sí, la verdad es que es bastante claro.

—Mira, esta tabla te explica un poco para que sirve cada cosa. Míratela y dime si lo entiendes todo bien.

Tabla 1: Significado de los términos reservados que aparecen en el código.

Program MIPONG;	Indica el inicio y el nombre del programa. Es algo así como la función main() en los lenguajes más tradicionales. Fíjate que lleva un “;” al final, a mi casi siempre se me olvida.
Global	Situado después del Program, permite definir variables globales, que serán visibles desde cualquier punto del programa.
BEGIN/END	Indican el comienzo y el final de un ámbito. Equivalen a las llaves en C/C++, Java y otros muchos lenguajes.
Process	Indica el comienzo de un proceso. Puedes entender un proceso como el equivalente a una función.
LOOP/END	Sirve para crear un bucle infinito. Esto parecería una tontería en la programación tradicional, pero en videojuegos es bastante habitual usar bucles “infinitos”.

—Bueno, es más o menos lo que imaginaba —dijo con aires de suficiencia.

—¿Ah, sí? Bueno pues entonces no te lo voy a explicar. Quiero que vayas a la documentación de Fenix y que lo averigües tú misma.

—Bah, oh. No te piques, explícamelo tú que seguro que lo entiendo mejor.

—No, no, si no lo digo porque me pique. Lo digo porque creo que es lo mejor; una de las mayores habilidades que debe tener un programador de videojuegos es la capacidad de aprendizaje y de investigación. Cada proyecto nuevo te exigirá nuevos y muy variados conocimientos, y tendrás que ser capaz de apañártelas y aprender por ti misma.

—Eso me gusta. Desde pequeña he tenido mucha inquietud por aprender un montón de cosas. Mi madre siempre me recuerda que ya de pequeña desmontaba todos los aparatos de la casa para intentar averiguar cómo funcionaban...

—Eso es muy bueno —le respondo mientras nos reímos—. Me parece algo fundamental para un buen desarrollador de videojuegos. Así que venga, *vete al manual on-line en el manual del Fenix (<http://fenix.divsite.net/>). Está en el menú de “documentación”; elige la primera opción “Documentación On-Line”.*

—Ya está. Esto parece como el índice de la documentación ¿no?

Ilustración 14: Índice de la documentación on-line



—Sí, esto que estás viendo es muy típico en programación cuando se presenta una API (el conjunto de funciones que te ofrece cierta aplicación para usarla). Siempre se suele poner un índice en la parte izquierda, con las diferentes categorías de la API y si haces clic en alguna de ellas, verás la lista de funciones que ofrece la API dentro de esa categoría. Este tipo de documentaciones on-line tienes que tenerlas siempre abiertas cuando estés programando, ¡sobre todo al principio!

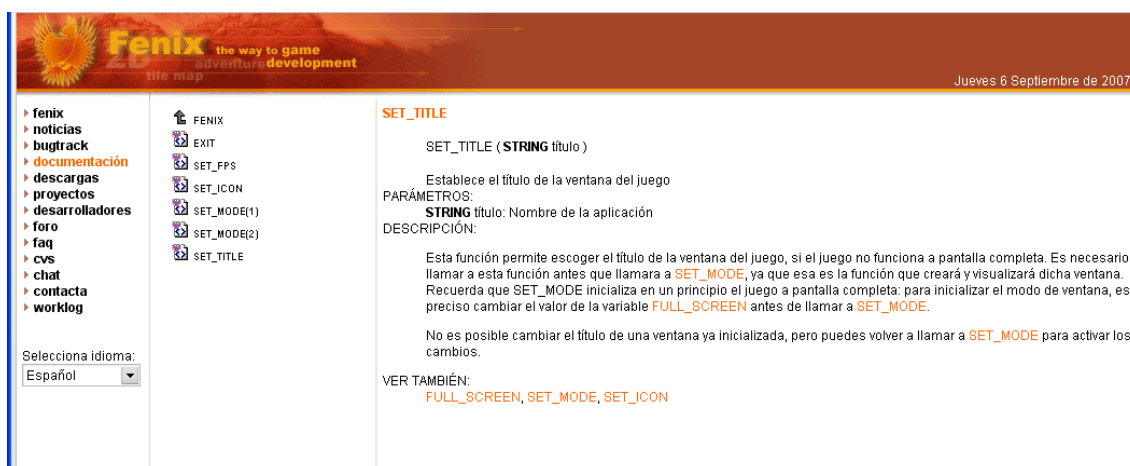
—Ya veo... Sí que parece útil para empezar, sí...

—No solo es útil, ¡es indispensable! *Venga, ahora quiero que vayas mirando todas las funciones que aparecen en el código y que las busques en la documentación.*

—Vale... Bueno, lo de “int RESOLUCION_X” y tal ya veo que es igual que en cualquier otro lenguaje de programación; simplemente declara una variable de tipo entero (integer), ¿verdad?

—Sí... Ya veo que has aprendido algo en estos tres años —le digo con un sutil tono sarcástico...

—Ja, ja, ja. Qué gracia me haces —responde en un tono ya claramente sarcástico—. Sigamos... A ver, esto de “set_title()” no sé qué es. Voy a mirar; Tiene pinta de estar en la categoría de inicialización... ¡Sí, aquí está!

Ilustración 15: Documentación de `set_title(string)`

—Sí, ahí está... Échale un vistazo que esa es bastante sencillita.

—Ah, pues sí. Además, la documentación lo explica muy claro. Bueno, así que con esto cambio el título de la ventana. ¡Ah, claro! Por eso salía “Mi Primer Pong” como el título de la ventana de antes. A ver, voy a seguir. Esto de Full_Screen no parece una función, es una variable en realidad; parece una variable reservada de Fenix. Ah, claro, debe estar en esto de “Variables Globales”.

—Efectivamente, como estás viendo en la documentación esa variable global sirve para definir si la ventana se creará a pantalla completa o en ventana. Si pruebas a cambiar ese false por un true, verás que se ejecuta a pantalla completa.

—¿Ah, sí? Voy a probar —inmediatamente cambia ese false a true, recompila el código y lo vuelve a ejecutar tal y como había hecho antes. Como es lógico, apareció lo mismo que antes, pero a pantalla completa.

—¡Mola! Empiezo a entender cómo va todo esto... Déjame seguir, que creo que podré encontrar todas las funciones en la documentación.

Y así siguió durante un rato, buscando en la documentación todas las funciones y declaraciones que no entendía. Sólo necesitó ayuda para la función `key()`, que por algún motivo no está en el índice. Se puede llegar hasta la documentación de `KEY` yendo a “Variables Globales > SCAN_CODE” y haciendo clic en el enlace que aparece en la descripción de la función.

—Genial. Ya entiendo todo. Pero no estoy segura de si sabría hacer algo por mi misma todavía— dijo al acabar de buscar todo en la documentación.

—Bueno, ahora lo que te falta es algo más de práctica, aunque teniendo la documentación delante y sabiendo lo que quieres, seguro que conseguirías algo. Aún así, como es tu primer día voy a guiarte bastante.

—De acuerdo. Supongo que ahora empezaremos con el juego de verdad.

—Sí, lo primero que quiero que hagas es que pienses en cómo es el PONG, que lo visualices y que me digas las cosas que crees que habría que programar.

—Vale... Entiendo que habrá que programar las dos paletas, y el control con el teclado. Y bueno, también habrá que programar la pelota para que rebote en las paletas y en los bordes superior e inferior de la pantalla.

—Bien, justo como esperaba (y quería) has caído en uno de los errores más típicos a la hora de desarrollar un videojuego. Has subestimado el desarrollo del juego y ahora estarás pensando que con programar esas tres cosas ya habrás terminado; de lo que no te has dado cuenta es de que hay que programar también algún sistema de puntos para controlar qué jugador gana, habrá que controlar que la pelota vuelva a aparecer cuando se salga de la pantalla, habrá que hacer una pantalla de bienvenida y que, si queremos la opción de un solo jugador, tendríamos que programar una pequeña “inteligencia artificial”, o IA, para una de las paletas.

—Bueno, hombre. Pero todo eso son tonterías, lo otro es lo más importante... Todo eso que has dicho se puede hacer en un momento.

—Sí, eso es lo que te parece ahora, pero todo eso seguramente nos lleve tanto tiempo como las tres cosas que tú has identificado. Además, ten en cuenta que estamos haciendo un PONG; si estuviéramos haciendo un proyecto más grande e ignorásemos todos esos detalles, llegaríamos a tener un problema en el desarrollo.

—¿Tan grave es?— dijo ella extrañada.

—Sí. Un ejemplo muy típico es la gente que empieza y enseguida quiere hacer un juego de rol, normalmente multijugador y masivo (reinventar el World Of Warcraft, vamos).

—¡Ey! Eso molaría— me interrumpió.

—Sí, molaría mucho. Y muchos piensan que “con programar un editor de escenarios (que no es poco), unos personajes que se puedan mover por él y un sistema cliente/servidor (que tampoco es moco de pavo) ya lo tienen hecho”. Al poco tiempo de empezar, los pocos que consiguen llegar a eso, se dan cuenta de que necesitan unas cantidades ingentes de gráficos y un montón de programación para completar todos los “detallitos” que requiere un juego así.

—Sí... Parece que tienes razón. Vale, entonces ya sabemos que tenemos que programar esas tres cosas que dije yo y los detalles que dijiste tú. ¿Por dónde empezamos?

—Pues por lo primero que has dicho: Por las paletas de los jugadores. Para no complicarnos vamos a hacerlo para dos jugadores, sin la opción de una paleta controlada por ordenador.

—Me parece bien.

—*Bueno, pues vete al código y añade este código al que te señalo.*

Ilustración 16: Variables que definen las características de la paleta del jugador 1.

```

Program MIPONG;
Global
  int RESOLUCION_X;
  int RESOLUCION_Y;

  int X_Paleta_1;
  int Y_Paleta_1;
  int Ancho_Paleta_1;
  int Alto_Paleta_1;
  int Color_Paleta_1;
BEGIN

```

—Aquí definimos cinco variables que definirán completamente las características de nuestro primer elemento de juego, que es la paleta del jugador 1. En este caso las dos paletas van a ser iguales así que podrían compartir las variables referentes a la anchura, la altura y el color (que no se van a modificar durante el juego), pero es mejor así por si quisiéramos modificarlas más adelante.

—Ya está, oye, pero me fijo que no pones comentarios a las variables. ¿No decías que era tan importante?

—Y lo es, pero fíjate también que pongo nombres muy descriptivos a las variables. Cualquiera puede entender la utilidad de esas variables. Creo que es muy buena costumbre utilizar este tipo de nombres para las variables, porque mucha gente utiliza abreviaturas o siglas que pueden resultar muy confusas para entender el código. Simplemente imagina que cambiara esas variables por cosas como XP1 o AP1; no se entendería nada.

» **Ahora añade esto:**

Ilustración 17: Código a añadir en Controlador_Juego

```

Process Controlador_Juego()
BEGIN
  Inicializar(); // Inicializar al estado inicial

  Loop // Bucle principal
    If(key(_esc)) // si se pulsa escape
      exit(0,0); // salir del juego
    End
  dibujar(); // dibujar todos los elementos en pantalla.

  Frame; // Refrescar pantalla

  End // del loop
END

```

—Aquí llamamos a dos funciones que programaremos justo ahora, aunque imagino que ya sabrás que es lo que van a hacer.

» **Por último, escribe esto al final del código:**

Ilustración 18: Código inicial de las funciones Dibujar() e Inicializar()

```

    dibujar(); // dibujar todos los elementos en pantalla.
    Frame; // Refrescar pantalla
End // del loop
END

////////////////////////////////////
// Inicializa el juego a su estado inicial
Process Inicializar()
BEGIN
    X_Paleta_1=30;
    Y_Paleta_1=240;
    Ancho_Paleta_1=20;
    Alto_Paleta_1=100;
END

////////////////////////////////////
// Dibuja los elementos del juego en pantalla
Process Dibujar()
Begin
    clear_screen(); // Limpiar la pantalla
    drawing_map(0,0); // Establecer que se dibuje en el fondo1
    // DIBUJAR PALETA 1
    drawing_color(rgb(255,255,255)); // Indicamos el color de dibujo
    draw_box(X_Paleta_1-Ancho_Paleta_1/2, Y_Paleta_1-Alto_Paleta_1/2,
            X_Paleta_1+Ancho_Paleta_1/2, Y_Paleta_1+Alto_Paleta_1/2);
End

```

—Aquí añadimos dos funciones que también son muy típicas en cualquier videojuego. Yo creo que deberían estar presentes, de una u otra forma, en todos los videojuegos.

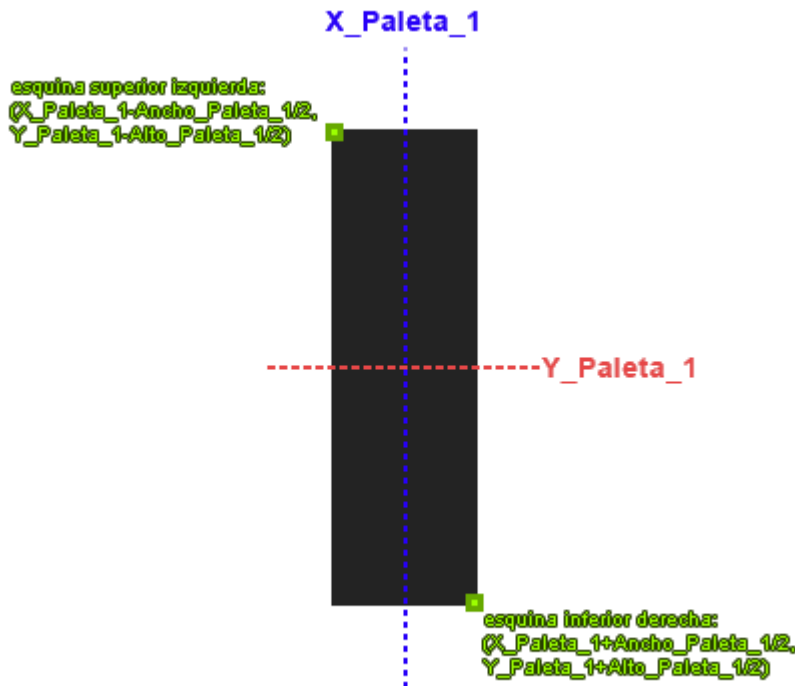
»La función Inicializar() —le seguí explicando— sirve para inicializar todas las variables del juego a un estado inicial. Lo más típico en estas funciones es reinicializar los contadores de puntuaciones o de vidas, aunque también se debe situar aquí cualquier otra tarea que sirva para poner el juego en su estado inicial.

»Finalmente, la función Dibujar() tiene como objetivo dibujar en pantalla todos los objetos del juego. Como en este caso solo estamos dibujando la paleta del jugador 1, sólo necesitamos un par de líneas.

—Entiendo... Pero estoy mirando la documentación de draw_box() y más o menos entiendo lo que hace, pero todo ese código que tienes ahí es muy complicado. ¿Puedes explicarme un poco todo eso?

—Sí, claro. En realidad no es tan complicado, simplemente estoy dando las cuatro coordenadas de un rectángulo en función de las variables que definimos antes. Aún así te voy a hacer un dibujo para que lo entiendas mejor.

Ilustración 19: Cálculo de las coordenadas del rectángulo de la paleta



—Ten en cuenta que, por convenio, el origen de coordenadas (el punto 0,0) está en la esquina superior izquierda, por lo que las coordenadas serán mayores cuanto más abajo y a la derecha estén— le dije para que pudiera entenderlo mejor.

—Ah claro, ahora lo entiendo. A la función `draw_box()` le pasas como argumentos las dos esquinas que definen el rectángulo. Y también veo que la posición de la paleta se define en el centro del rectángulo.

—Sí, eso suele ser lo habitual, aunque también sería posible tomar la posición de la paleta como la esquina superior izquierda. Aún así, en este caso es mejor así porque nos interesa que el rectángulo aparezca centrado en dichas coordenadas.

—Vale, ya lo entiendo perfectamente. ¿Lo compilo y ejecuto?

—Sí. ¿Te acuerdas de cómo era no?

—Sí, claro.

Ilustración 20: Recordatorio de cómo compilar y ejecutar el juego

```
C:\Documents and Settings\Miguel\Escritorio\dia 1>fxc mipong.prg_
```

```
C:\Documents and Settings\Miguel\Escritorio\dia 1>fxi mipong.dcb_
```

Tecleando con un aspecto bastante animado, ejecuta sin problema las órdenes para compilar y ejecutar el juego y ve la paleta ya dibujada en la pantalla.

Ilustración 21: Captura del juego mostrando la primera paleta dibujada.



—Ajá. La verdad es que es exactamente lo que esperaba... Pero hay una cosa que todavía no veo; ¿cómo se crean los gráficos de verdad? Porque no creo que todos los gráficos se hagan a base de rectángulos de colores— me preguntó con bastante curiosidad.

—Bueno, la verdad es que esperaba que me preguntases esto y no me extraña porque al principio podría parecer que esto de dibujar rectángulos es una cutrez. Sin embargo, la potencia de utilizar primitivas (se les llama así a las formas básicas como rectángulos, círculos o polígonos) es algo normalmente muy infravalorado y que incluso yo mismo descubrí hace bastante poco. Pero sí, también reconozco que lo más normal, aunque innecesario para este ejemplo, sería crear un gráfico de una paleta con un programa como el paint por ejemplo y cargarlo en el juego a través de ciertas funciones que nos permiten trabajar con gráficos.

—Ah, claro... Tiene sentido, pero tengo la sensación de que hoy todavía no vamos a entrar en eso, ¿verdad?

—Verdad. Y no porque sea más difícil, de hecho sería más fácil que lo que estamos haciendo, si no porque quiero que aprendas a aprovechar todos los recursos que tienes en tu mano y hagas las cosas siempre de la forma más flexible. Ten en cuenta que ahora, sin cambiar nada más que

unas líneas de código, podríamos cambiar el tamaño, la forma o el color de la paleta. Si hubiéramos hecho un gráfico para esto tendríamos que volver a crearlo.

—Sí, la verdad es que tiene mucho sentido. Además así queda muy retro; en parte tiene su gracia también...

—Sí. Bueno, ¿quieres hacer que se mueva?

—¡Por supuesto!

—Bien, pues va a ser muy sencillo. *Vamos a crear una nueva función llamada Actualizar() donde gestionaremos todo lo relacionado con el comportamiento de los objetos. Desde ella nos ocuparemos de actualizar la posición de la paleta según lo que recibamos desde el teclado. Venga, coge el teclado y añade este código que te pongo:*

Ilustración 22: Llamada a la función Actualizar() desde Controlador_Juego

```

////////////////////////////////////
// Controlador del juego (aquí se controla el juego)

Process Controlador_Juego()
BEGIN

  Inicializar(); // Inicializar al estado inicial

  Loop // Bucle principal

    If(key(_esc)) // si se pulsa escape
      exit(0,0); // salir del juego
    End

    Actualizar(); // Actualizar la lógica del juego
    Dibujar(); // dibujar todos los elementos en pantalla.

    Frame; // Refrescar pantalla

  End // del loop

END

```

Ilustración 23: Función que actualiza la lógica del juego

```

// DIBUJAR PALETA 1

drawing_color(rgb(255,255,255)); // Indicamos el color de dibujo
draw_box(X_Paleta_1-Ancho_Paleta_1/2, Y_Paleta_1-Alto_Paleta_1/2,
        X_Paleta_1+Ancho_Paleta_1/2, Y_Paleta_1+Alto_Paleta_1/2);
End

////////////////////////////////////
// Actualiza la lógica del juego

Process Actualizar()
Begin

  if(key(_up))
    Y_Paleta_1 -= 3;
  End

  if(key(_down))
    Y_Paleta_1 += 3;
  End

End

```

—Oye, antes de probarlo, ¿qué es eso de la “lógica del juego”?

—Ah, claro, perdón por no explicártelo antes. Por lógica del juego normalmente se entiende todo aquello que no tiene que ver con dibujar los objetos en pantalla. Es decir, la lógica define la forma en que funcionan los elementos del juego. Después, la parte de dibujado o de “render” se ocupa de representar todos esos elementos en la pantalla, aunque también hay elementos del juego que no tienen representación visual. De todas formas, todo lo relacionado con terminología no es importante ahora mismo; con el tiempo irás aprendiendo la jerga que tanto nos gusta usar, para desgracia de los que empiezan.

Inmediatamente después de mi explicación, Olaya se lanza a compilar y ejecutar (parece que lleve ya años haciéndolo) y prueba el principio de su primer videojuego. Enseguida se da cuenta del primer “detalle” en el que no había caído.

—Eh, vaya fallo. La paleta puede salirse de la pantalla.

—Sí, claro. No veo en tu código nada que lo impida... *Venga, quiero que lo soluciones tú sola; no es tan difícil. Pero acuérdate del método que utilizamos antes para calcular las esquinas de la paleta.*

—Vale... Déjame pensar.

Está probando durante un rato varias soluciones diferentes, y al rato obtiene una solución válida.

—Ya lo tengo Miguel. ¿Es así?

Ilustración 24: Código que controla que la paleta 1 no se salga de la pantalla

```
// Actualiza la lógica del juego
Process Actualizar()
Begin
    if(key(_up) AND Y_Paleta_1-Alto_Paleta_1/2 > 50)
        Y_Paleta_1 -= 3;
    End
    if(key(_down) AND Y_Paleta_1+Alto_Paleta_1/2 < 430)
        Y_Paleta_1 += 3;
    End
End
```

—Bueno, casi. Falta una cosa... ¡No has comentado el código! Y eso ya puede resultar suficientemente complicado como para que alguien, o a ti misma dentro de un mes, no lo entienda a la primera.

—Vaaale— responde al tiempo que se pone a escribir los comentarios.

Ilustración 25: Comentarios que explican la función actualizar()

```

Process Actualizar()
Begin
// si se pulsa "arriba" y la paleta no ha llegado arriba
IF(key(_up) AND Y_Paleta_1-Alto_Paleta_1/2 > 50)
  Y_Paleta_1 -= 3; // Hacemos que suba
End
// si se pulsa "abajo" y la paleta no ha llegado abajo
IF(key(_down) AND Y_Paleta_1+Alto_Paleta_1/2 < 430)
  Y_Paleta_1 += 3; // Hacemos que baje
End
End

```

—¿Mejor? —me dice al terminar de escribirlos.

—Pues sí, mucho mejor —le digo sonriendo—. Pero quiero hacer hincapié en una cosa: Escribir directamente el código de la lógica en la función Actualizar() es aceptable para este juego porque es muy sencillo, pero a poco que haya que controlar más cosas de lógica, lo normal es crear diferentes funciones como Actualizar_Teclado() o Actualizar_Paleta() y llamarlas desde la función actualizar para mantenerlo todo mucho más organizado. Todo eso es aplicable también para la función dibujar.

—Ya me imagino... Meter toda la —se toma unos instantes para encontrar la palabra—... lógica de un juego grande dentro de la misma función sería un jaleo.

—Sí, sería algo prácticamente incomprensible a primera vista. *Bueno, ya tenemos al primer jugador listo; ahora quiero que hagas tú sola, y sin copiar, al jugador dos. Pero como en los cursores no es muy cómodo para jugar los dos, haz que las paletas se controlen con teclas normales del teclado.*

—Vale, allá voy.

Para forzarla a pensar por ella misma, salgo de la sala con el pretexto de ir al baño y la dejo sola enfrentándose sola por primera vez a un problema (aunque también es verdad que es un problema muy sencillo).

Al salir de la sala, me dirijo hacia las escaleras y, como me esperaba, me encuentro con algunos de mis amigos de clase jugando a la pocha. Más concretamente, allí están Toni, Tulipán Bnistel, Tuero, Julián y Christian.

—Ey, chavales. ¿Qué tal?— les digo a modo de saludo —. ¿Y el resto?

—Parece que todavía no llegaron —responde Julián.

—Oye —interrumpe Toni—, pero eso no es lo raro... ¿Qué haces tú aquí a estas horas? Creía que no empezabas hasta por la tarde.

—Nah, es que tenía que hacer unas cosas...

—¿Cómo que unas cosas? —insiste con mucho más interés ahora.

—Joer Toni, es que no te enteras de una —dice Christian—, ¿no sabes que quedó ya temprano con Olaya para “hacer juegos”? —continuó, haciendo énfasis en el final de su frase al tiempo que dibujaba una sonrisa en su cara.

—¡Ohhhh yesssss! ¡Very Fuckin’! —gritó Toni al oírlo—. ¿Así que con Olaya eh? Vaya, vaya, y parecías tonto cuando te sacamos de la basura.

—Bueno, bueno —me defendí yo—. Lo primero, quiero aclarar que fue ella quién me lo pidió y que lo hago por hacerle un favor simplemente.

—Ya, ya —intervino Tuero—... Yo creo que lo haces más bien porque quieres que ella te devuelva el favor. ¿O no?

—Bueno, ya veo por donde va la cosa —respondí—. Adiós.

Entonces volví hacia la sala de prácticas.

La verdad es que, aunque no quería reconocerlo, no les faltaba razón; es innegable que Olaya es preciosa y además siempre ha demostrado ser muy inteligente y divertida... Definitivamente no me molestaría que se llegase a fijar en mí.

—¿Ya lo tienes? —pregunté al volver al lado de Olaya.

—Sí. Aquí está:

Ilustración 26: Nuevas variables globales

```

////////////////////////////////////
// Función inicial (aquí empieza todo)

Program MIPONG;
Global
  int RESOLUCION_X;
  int RESOLUCION_Y;

// VARIABLES GLOBALES
int Velocidad_Paletas;

// CARACTERÍSTICAS DE LA PALETA 1
int X_Paleta_1;
int Y_Paleta_1;
int Ancho_Paleta_1;
int Alto_Paleta_1;
int Color_Paleta_1;

// CARACTERÍSTICAS DE LA PALETA 2
int X_Paleta_2;
int Y_Paleta_2;
int Ancho_Paleta_2;
int Alto_Paleta_2;
int Color_Paleta_2;

BEGIN

```

—¡Guau! Veo que ya estás cogiendo el truco a esto —le dije algo sorprendido—. Me sorprende que hayas creado esa variable global “Velocidad_Paletas”. Realmente eso está muy bien.

—Sí —respondió orgullosa—. Se me ocurrió cuando tuve que ajustar por tercera vez la velocidad de las dos paletas; me di cuenta de que así tengo que modificar menos código para ajustarla.

Ilustración 27: Inicializar nuevas variables

```
////////////////////////////////////  
// Inicializa el juego a su estado inicial  
  
Process Inicializar()  
BEGIN  
  
    velocidad_Paletas = 5;  
  
    // INICIALIZAR PALETA 1  
    X_Paleta_1=30;  
    Y_Paleta_1=240;  
    Ancho_Paleta_1=20;  
    Alto_Paleta_1=100;  
  
    // INICIALIZAR PALETA 2  
    X_Paleta_2=610;  
    Y_Paleta_2=240;  
    Ancho_Paleta_2=20;  
    Alto_Paleta_2=100;  
  
END
```

—Muy bien, aquí veo que inicializas correctamente las variables que introdujiste antes.

—Sí, bueno esto no era tan complicado...

Ilustración 28: Nuevo código para actualizar y dibujar la segunda paleta

```

////////////////////////////////////
// Dibuja los elementos del juego en pantalla
Process Dibujar()
Begin

  clear_screen(); // Limpiar la pantalla
  drawing_map(0,0); // Establecer que se dibuje en el fondo

  // DIBUJAR PALETA 1

  drawing_color(rgb(255,255,255)); // Indicamos el color de dibujo
  draw_box(X_Paleta_1-Ancho_Paleta_1/2, Y_Paleta_1-Alto_Paleta_1/2,
    X_Paleta_1+Ancho_Paleta_1/2, Y_Paleta_1+Alto_Paleta_1/2);

  // DIBUJAR PALETA 2

  drawing_color(rgb(255,255,255)); // Indicamos el color de dibujo
  draw_box(X_Paleta_2-Ancho_Paleta_2/2, Y_Paleta_2-Alto_Paleta_2/2,
    X_Paleta_2+Ancho_Paleta_2/2, Y_Paleta_2+Alto_Paleta_2/2);

End

////////////////////////////////////
// Actualiza la lógica del juego
Process Actualizar()
Begin

  // ACTUALIZAR PALETA 1

  // Si se pulsa "arriba" y la paleta no ha llegado arriba
  if(key(_d) AND Y_Paleta_1-Alto_Paleta_1/2 > 50)
    Y_Paleta_1 -= velocidad_Paletas; // Hacemos que suba
  End

  // Si se pulsa "abajo" y la paleta no ha llegado abajo
  if(key(_c) AND Y_Paleta_1+Alto_Paleta_1/2 < 430)
    Y_Paleta_1 += velocidad_Paletas; // Hacemos que baje
  End

  // ACTUALIZAR PALETA 2

  // Si se pulsa "arriba" y la paleta no ha llegado arriba
  if(key(_k) AND Y_Paleta_2-Alto_Paleta_2/2 > 50)
    Y_Paleta_2 -= Velocidad_Paletas; // Hacemos que suba
  End

  // Si se pulsa "abajo" y la paleta no ha llegado abajo
  if(key(_m) AND Y_Paleta_2+Alto_Paleta_2/2 < 430)
    Y_Paleta_2 += Velocidad_Paletas; // Hacemos que baje
  End

End

```

—Bien, y veo que aquí has hecho básicamente lo mismo que con la anterior paleta. Bueno, yo creo que está todo bien... ¿Lo probamos?

—Sí, claro.

Enseguida compila y ejecuta y nos ponemos a hacer que las dos paletas suban y bajen hasta que nos cansamos...

Ilustración 29: Captura del juego con las dos paletas funcionando



—¿Mola, eh? —me dijo al cerrar la aplicación.

—Pues sí. Por aquí empezó la historia de los videojuegos y por aquí estás empezando tú... ¿No es bonito?

—Bueno, bueno, no te pases tampoco... Venga, quiero hacer la pelota de una vez ya. *Creo e inicializo las variables para sus características, ¿no?*

—Vale.

Ilustración 30: Variables globales para la pelota

```
// CARACTERÍSTICAS DE LA PALETA 2
int X_Paleta_2;
int Y_Paleta_2;
int Ancho_Paleta_2;
int Alto_Paleta_2;
int Color_Paleta_2;

// CARACTERÍSTICAS DE LA BOLA
int X_Bola;
int Y_Bola;
int Ancho_Bola;
int Alto_Bola;
int Color_Bola;

BEGIN

// INICIALIZAR ASPECTOS GRÁFICOS
```

Ilustración 31: Inicializar características de la bola

```

////////////////////////////////////
// Inicializa el juego a su estado inicial

Process Inicializar()
BEGIN

    velocidad_Paletas = 5;

    // INICIALIZAR PALETA 1
    X_Paleta_1=30;
    Y_Paleta_1=240;
    Ancho_Paleta_1=20;
    Alto_Paleta_1=100;

    // INICIALIZAR PALETA 2
    X_Paleta_2=610;
    Y_Paleta_2=240;
    Ancho_Paleta_2=20;
    Alto_Paleta_2=100;

    // INICIALIZAR BOLA

    X_Bola=320;
    Y_Bola=240;
    Ancho_Bola=15;
    Alto_Bola=15;

END

```

—*Ya está, ahora voy a hacer que la dibuje en pantalla*—dijo ella, con mucha confianza ya.

Ilustración 32: Código para dibujar la pelota

```

////////////////////////////////////
// Dibuja los elementos del juego en pantalla

Process Dibujar()
Begin

    clear_screen(); // Limpiar la pantalla
    drawing_map(0,0); // Establecer que se dibuje en el fondo

    // DIBUJAR PALETA 1

    drawing_color(rgb(255,255,255)); // Indicamos el color de dibujo
    draw_box(X_Paleta_1-Ancho_Paleta_1/2, Y_Paleta_1-Alto_Paleta_1/2,
             X_Paleta_1+Ancho_Paleta_1/2, Y_Paleta_1+Alto_Paleta_1/2);

    // DIBUJAR PALETA 2

    drawing_color(rgb(255,255,255)); // Indicamos el color de dibujo
    draw_box(X_Paleta_2-Ancho_Paleta_2/2, Y_Paleta_2-Alto_Paleta_2/2,
             X_Paleta_2+Ancho_Paleta_2/2, Y_Paleta_2+Alto_Paleta_2/2);

    // DIBUJAR BOLA

    drawing_color(rgb(255,255,255)); // Indicamos el color de dibujo
    draw_box(X_Bola-Ancho_Bola/2, Y_Bola-Alto_Bola/2,
             X_Bola+Ancho_Bola/2, Y_Bola+Alto_Bola/2);

End

```

—Está bien, Olaya. *Prueba a compilarlo y ejecutarlo antes de seguir; de momento es mejor ir con calma para no acumular errores.*

—Vale.

Al compilar y ejecutar, obtiene el resultado esperado:

Ilustración 33: Juego con la bola dibujándose en pantalla



—Bueno, parece que funciona bien —le dije.

—Sí, pero estoy pensando que va a ser muy complicado hacer que la bola se mueva y rebote y todo eso...

—Bueno. En realidad no demasiado, porque por la forma y colocación de las paletas resulta muy sencillo controlar todas las colisiones y los rebotes. Ten en cuenta que las paletas no pueden rotar ni tomar formas curvadas, eso facilita mucho los cálculos.

—Ya, aún así creo que me costaría.

—Probablemente; al principio estas cosas de matemáticas cuestan bastante, aunque sean muy básicas. Pero normalmente esto pasa porque la gente no se para a pensarlo; otro tipo de código se ve inmediatamente al verlo escrito, pero para entender código relacionado con físicas o geometría muchas veces hace falta dejar de mirar a la pantalla, coger un papel y un lápiz y dibujar el problema en cuestión. De esa forma todo se ve mucho más claro y entonces ahorrarás tiempo.

—Sí, como cuando antes me dibujaste lo de las coordenadas de la paleta.

—Sí. Aunque esta vez no te voy a hacer ningún dibujo porque esto es lo último que vamos a hacer hoy y vas a tener mucho tiempo para entenderlo hasta que volvamos a quedar otro día. Si es que tú quieres, claro...

—¡Sí, claro! Hoy estoy disfrutando muchísimo; aunque esto sea una tontería, voy a poder llegar a casa y enseñárselo a mi hermana. Creo que va a quedar muy impresionada —dijo mientras reía al imaginarse la escena.

—Me alegro de que lo estés pasando bien —le respondí, devolviéndole la sonrisa—. Vamos que ya nos queda poco. A ver, como es un código bastante grande, vamos a meter la lógica de la bola en una función (o proceso en este caso) aparte. *Copia al final este código de aquí:*

Ilustración 34: Control de colisiones de la bola

```

////////////////////////////////////
// Actualiza la lógica de la bola
Process Control_Bola()
Begin
|
  // COLISIONES CON PALETA 1
  If(Velocidad_Bola_X < 0 && X_Bola>X_Paleta_1 // si la bola está en zona de colisionar con la paleta 1
    && X_Bola-Ancho_Bola/2<=X_Paleta_1+Ancho_Paleta_1/2)

    If(abs(Y_Paleta_1-Y_Bola)<Alto_Paleta_1*0.5+Alto_Bola*0.5) // si la paleta 1 está bien colocada
      Velocidad_Bola_X *=-1;
    End

  End

  // COLISIONES CON PALETA 2
  If(Velocidad_Bola_X >0 && X_Bola<X_Paleta_2 // si la bola está en zona de colisionar con la paleta 2
    && X_Bola+Ancho_Bola/2>=X_Paleta_2-Ancho_Paleta_2/2)

    If(abs(Y_Paleta_2-Y_Bola)<Alto_Paleta_2*0.5+Alto_Bola*0.5) // si la paleta 2 está bien colocada
      Velocidad_Bola_X *=-1;
    End

  End

  // COLISIONES CON PAREDES SUPERIOR E INFERIOR
  If(Velocidad_Bola_Y<0 && Y_Bola-Ancho_Bola*0.5<=50) // si la bola colisiona con la pared superior
    Velocidad_Bola_Y *=-1;
  End
  If(Velocidad_Bola_Y>0 && Y_Bola+Ancho_Bola*0.5>=430) // si la bola colisiona con la pared inferior
    Velocidad_Bola_Y *=-1;
  End

  X_Bola+=Velocidad_Bola_X;
  Y_Bola+=Velocidad_Bola_Y;

End

```

Línea 185, columna 1

—Buf... ¿De verdad —comenzó a preguntar preocupada— que no me vas a explicar esto? Parece muy complicado. Yo no entiendo nada ahí.

—Ya lo imagino, pero recuerda lo que te acabo de decir. Este tipo de código que hace cálculos gráficos se tiene que leer a la vez que se hace un dibujo de los elementos involucrados. De otra forma es muy difícil entenderlo, sobre todo cuando no tienes costumbre. *Ahora crea e inicializa las dos variables globales que estamos usando en ese proceso:*

Ilustración 35: Variables de velocidad de la bola

```
// CARACTERÍSTICAS DE LA BOLA
int X_Bola;
int Y_Bola;
int Ancho_Bola;
int Alto_Bola;
int Color_Bola;
int velocidad_Bola_X;
int velocidad_Bola_Y;
BEGIN
// INICIALIZAR ASPECTOS GRÁFICOS
set_title("Mi Primer Pong"); // Establece el título de la ventana del juego
```

Ilustración 36: Inicializar la velocidad de la bola

```
// INICIALIZAR BOLA
X_Bola=320;
Y_Bola=240;
Ancho_Bola=15;
Alto_Bola=15;
velocidad_Bola_X = 3;
velocidad_Bola_Y = 3;
END
////////////////////////////////////
// Dibuja los elementos del juego en pantalla
Process Dibujar()
Begin
clear_screen(); // Limpiar la pantalla
drawing_map(0,0); // Establecer que se dibuje en el fondo
```

—Y ahora, como te podrás imaginar, tenemos que añadir al código de “Actualizar()” una llamada a la función que acabas de crear.

Ilustración 37: Llamada a control bola

```
// ACTUALIZAR PALETA 2
// si se pulsa "arriba" y la paleta no ha llegado arriba
if(key(_k) AND Y_Paleta_2-Alto_Paleta_2/2 > 50)
Y_Paleta_2 -= Velocidad_Paletas; // Hacemos que suba
End
// si se pulsa "abajo" y la paleta no ha llegado abajo
if(key(_m) AND Y_Paleta_2+Alto_Paleta_2/2 < 430)
Y_Paleta_2 += Velocidad_Paletas; // Hacemos que baje
End
control_Bola();
End
////////////////////////////////////
// Actualiza la lógica de la bola
Process Control_Bola()
Begin
```

—Bueno, ya está —le dije.

—¿Ya está? ¿Podemos jugar ya?

—*Sí claro, compila y ejecuta.*

Ilustración 38: El primer PONG de Olaya funcionando. ¡Mola!



Al compilar, llevamos inmediatamente las manos al teclado y empezamos a jugar. Al cabo de un rato, me despisto por un instante y se me escapa la bola por un poco.

—¡Oh, sí! Te he ganado —grita Olaya mientras se levanta de la silla con los brazos en alto—. Bueno, qué otra cosa podría haber ocurrido, después de todo este es mi PONG.

—Bueno, bueno, tampoco te pases que te he ayudado bastante —le dije, siguiendo con la broma.

—Vale, tienes razón. Pero te aseguro que voy a convertirlo en mi PONG. ¡Haré algo que lo haga único en el mundo!

—Me parece genial, porque eso es exactamente lo que te iba a pedir. Es verdad que ahora funciona, pero fíjate que no tiene ni marcador ni pantalla de bienvenida, y cuando la bola se sale de la pantalla hay que reiniciarlo para seguir jugando. Además, todavía no has entendido del todo el código de las colisiones entre la bola y las paletas... Todavía te queda mucho curro en este PONG, así que creo que dejaremos algún tiempo hasta la próxima lección para que tengas tiempo a experimentar todo lo que necesitas.

—Sí, creo que es lo mejor —respondió—. ¿Qué te parece volver a quedar en unas tres semanas, más o menos?

—Bueno, sí, creo que estaría bien. En ese tiempo podrás terminar este PONG e, incluso, hacer alguna otra cosilla. Fíjate que no hemos usado las variables del color de los objetos, me gustaría que investigases para averiguar como usarlas.

—Sí. Confía en mí que esto me divierte muchísimo.

—Vale, pero ten en cuenta que te falta bastante para conocer en profundidad el Fenix; por ejemplo, tiene un montón de características que facilitan el trabajo y no te he enseñado, principalmente para que conozcas como se trabaja con lo más básico. Seguramente te encuentres con un montón de dificultades, quizás algunas demasiado grandes.

—Ya, supongo que sí... ¡Pero para eso estás tú! Te aseguro que te voy a dar mucho la turra a partir de ahora —dijo mientras se reía.

—Lo imagino... Aunque bueno, tratándose de ti haré un esfuerzo... Pero no te preocupes, que no te de miedo preguntar o investigar en google. Recuerda que la capacidad de aprender y solucionar problemas es una de las mayores virtudes en un desarrollador de videojuegos. Tienes que empezar ya a practicar.

Cerramos todas las ventanas que teníamos abiertas, ejecutamos por última vez el código recién creado y empezamos a jugar al PONG, el primer juego creado por mi amiga. Esto podría ser el comienzo de un gran cambio en la vida de Olaya... ¡Podría convertirse en desarrolladora de videojuegos! Eso sí que molaría.

Fin del capítulo

Nota del autor:

Hola, soy Miguel Santirso, el autor de esta guía y ahora me dirijo a ti, el lector que has seguido este capítulo hasta el final. Realmente agradezco que hayas elegido esta guía de entre todas las que hay en internet y espero que hayas disfrutado y aprendido con ella.

Como cualquier cosa que se haga, esta guía puede contener errores, omisiones o partes mejorables. Por eso, os invito a que contactéis conmigo a través de los diferentes medios que tenéis a vuestra disposición para comentarme cualquier cosa que me ayude a mejorar los siguientes capítulos. También podéis consultar dudas que tengáis acerca del desarrollo de videojuegos (ya sean dudas técnicas o no), pero esto prefiero que lo preguntéis a través de los foros en los que me suelo mover (www.stratos-ad.com ó www.colectivoguma.com) para que cualquiera pueda leer vuestras preguntas y os pueda responder más gente aparte de mí.

Mi correo electrónico personal es tirso.00@gmail.com, pero os pediría que lo utilizéis solo para asuntos importantes ya que para otras cosas prefiero que me contactéis de cualquier otra forma.

Apéndice A: Listado de imágenes

Ilustración 1: Portada de la web de Fenix	3
Ilustración 2: Descargar los binarios de Fenix	3
Ilustración 3: Abrir una línea de comandos	5
Ilustración 4: Navegar hasta la carpeta de trabajo	5
Ilustración 5: Crear un archivo de texto y abrirlo con el notepad.	5
Ilustración 6: Las primeras líneas de código de cualquier juego	6
Ilustración 7: Esqueleto típico de un juego	7
Ilustración 8: Orden para compilar el programa	8
Ilustración 9: Resultado de la compilación	8
Ilustración 10: Archivo generado tras una compilación exitosa	8
Ilustración 11: Instrucciones para ejecutar el juego	8
Ilustración 12: Captura del "juego" funcionando.	9
Ilustración 13: Palabras reservadas en el código de Fenix	10
Ilustración 14: Índice de la documentación on-line.....	12
Ilustración 15: Documentación de set_title(string)	13
Ilustración 16: Variables que definen las características de la paleta del jugador 1.	15
Ilustración 17: Código a añadir en Controlador_Juego.....	15
Ilustración 18: Código inicial de las funciones Dibujar() e Inicializar()	16
Ilustración 19: Cálculo de las coordenadas del rectángulo de la paleta	17
Ilustración 20: Recordatorio de cómo compilar y ejecutar el juego	17
Ilustración 21: Captura del juego mostrando la primera paleta dibujada.	18
Ilustración 22: Llamada a la función Actualizar() desde Controlador_Juego	19
Ilustración 23: Función que actualiza la lógica del juego.....	19
Ilustración 24: Código que controla que la paleta 1 no se salga de la pantalla	20
Ilustración 25: Comentarios que explican la función actualizar()	21
Ilustración 26: Nuevas variables globales.....	22
Ilustración 27: Inicializar nuevas variables	23
Ilustración 28: Nuevo código para actualizar y dibujar la segunda paleta	24
Ilustración 29: Captura del juego con las dos paletas funcionando.....	25
Ilustración 30: Variables globales para la pelota	25
Ilustración 31: Inicializar características de la bola	26
Ilustración 32: Código para dibujar la pelota.....	26
Ilustración 33: Juego con la bola dibujándose en pantalla	27
Ilustración 34: Control de colisiones de la bola	28
Ilustración 35: Variables de velocidad de la bola	29
Ilustración 36: Inicializar la velocidad de la bola	29
Ilustración 37: Llamada a control bola	29
Ilustración 38: El primer PONG de Olaya funcionando. ¡Mola!	30