

Implementación de Aplicaciones Interactivas

Desarrollo de Sistemas Interactivos

Contenido

1. Programación de Aplicaciones Interactivas
 1. Programación Orientada a Eventos
 2. Gestión de la Entrada/Salida
2. Teclado
3. Ratón
4. Abstracción de la Entrada/Salida
5. Separación de Diseño y Programación
 1. Programación del GUI
 2. Lenguajes de etiquetas
 3. Lenguajes de estilo de diseño
6. Patrones de diseño de aplicaciones interactivas
 1. MVC
 2. MVVM

1 Programación de Aplicaciones Interactivas

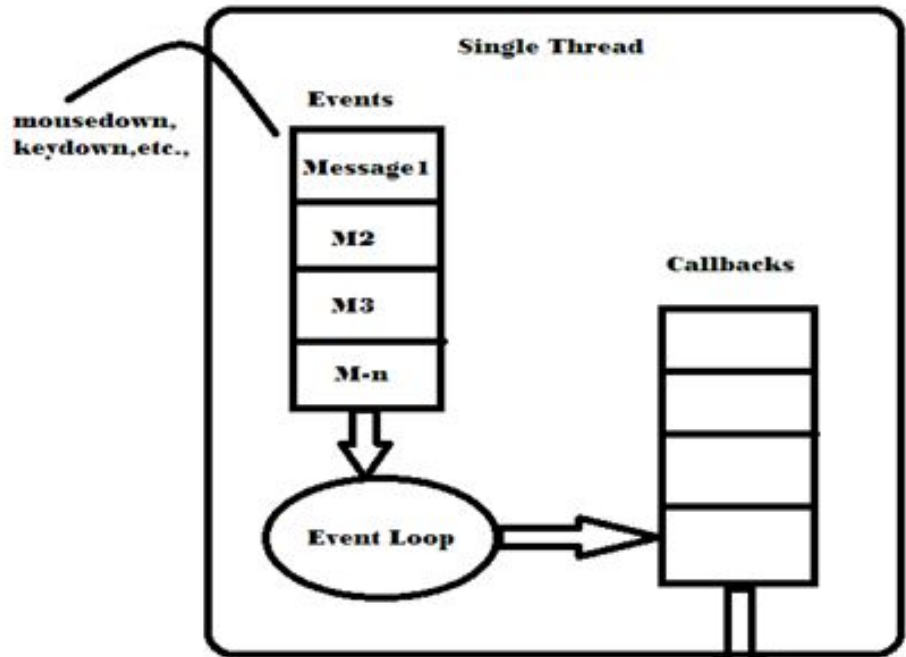
Paradigma de la programación orientada a eventos: Los programas atienden a **sucesos (eventos)** que ocurren, y dependiendo de cuáles sean, se ejecutan diferentes funciones.

La programación orientada a eventos **es la forma natural** de programar un sistema interactivo, pues **los usuarios generan eventos** a través de los dispositivos de entrada.

Supone una complicación añadida con respecto a otros paradigmas de programación, debido a que el **flujo de ejecución del software escapa al control del programador**.

1.1 Modelo de Eventos

- **Evento** : El evento (por ejemplo, eventos de ratón)
- **Mensaje**: El gestor de eventos (normalmente el S.O) genera un mensaje que contiene propiedades asociadas al evento.
- **Bucle de mensajes**: Encargado de atender los mensajes y distribuirlos.
- **Callback**: La función encargada de responder al mensaje (maneja el evento).



1.1 Tipos de Eventos

- **Externos:** Producidos por el usuario a través de dispositivos de **Entrada/Salida (I/O)**.

Ejemplos:

- Eventos de pulsaciones de teclado o ratón.
- Gestos de Usuario (“Hola Cortana”).

- **Internos:** Producidos por el sistema o la aplicación.

Ejemplos:

- Vencimiento de un temporizador.
- Orden de Suspende el Sistema.

1.2 Mapeo de la Entrada/Salida (I/O)

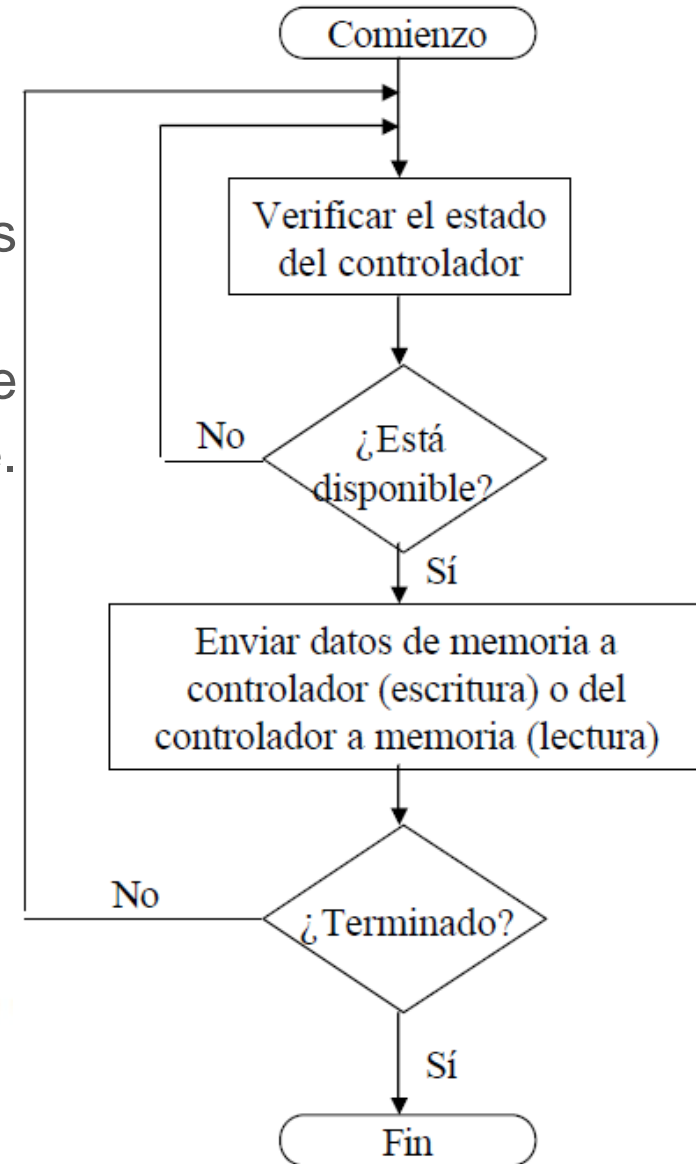
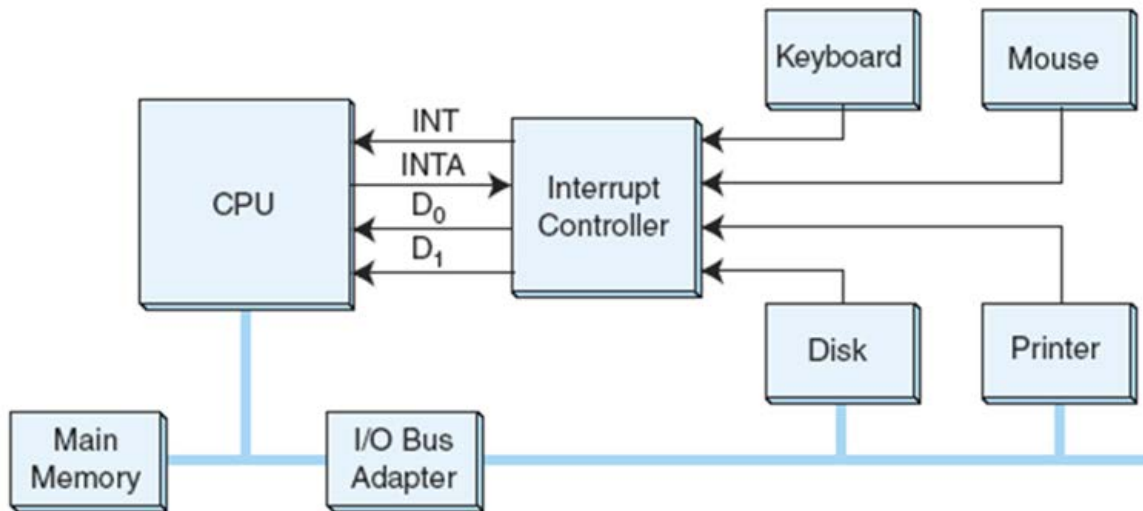
- **En memoria:** acceso con instrucciones de lectura/escritura en memoria.
- **En puertos de E/S:** acceso con instrucciones especiales (IN y OUT).

Tree	Address Range	Device	Status
System Information	0x03F6-0x03F6	Primary IDE Channel	OK
System Summary	0x0170-0x0177	Secondary IDE Channel	OK
Hardware Resources	0x0376-0x0376	Secondary IDE Channel	OK
Conflicts/Sharing	0xE400-0xE43F	Intel(R) PRO/100+ Management Ad...	OK
DMA	0xE800-0xE83F	Creative AudioPCI (ES1371,ES1373)...	OK
Forced Hardware	0x0200-0x0207	Game Port for Creative	OK
I/O	0x0020-0x0021	Programmable interrupt controller	OK
IRQs	0x00A0-0x00A1	Programmable interrupt controller	OK
Memory	0x0040-0x0043	System timer	OK
Components	0x0000-0x000F	Direct memory access controller	OK
Software Environment	0x0081-0x0083	Direct memory access controller	OK
Internet Explorer 5	0x0087-0x0087	Direct memory access controller	OK
Applications	0x0089-0x008B	Direct memory access controller	OK
	0x008F-0x0091	Direct memory access controller	OK
	0x00C0-0x00DF	Direct memory access controller	OK
	0x0060-0x0060	PC/AT Enhanced PS/2 Keyboard (10...	OK
	0x0064-0x0064	PC/AT Enhanced PS/2 Keyboard (10...	OK
	0x0378-0x037F	Printer Port (LPT1)	OK
	0x03F8-0x03FF	Communications Port (COM1)	OK
	0x02F8-0x02FF	Communications Port (COM2)	OK
	0x03F2-0x03F5	Standard floppy disk controller	OK

1.2 Gestión de la Entrada/Salida (I/O)

La I/O se puede manejar por:

- **Encuesta (Polling):** el dispositivo de I/O es consultado periódicamente por la CPU.
- **Interrupciones:** el dispositivo I/O interrumpe la CPU cuando quiere comunicarse.

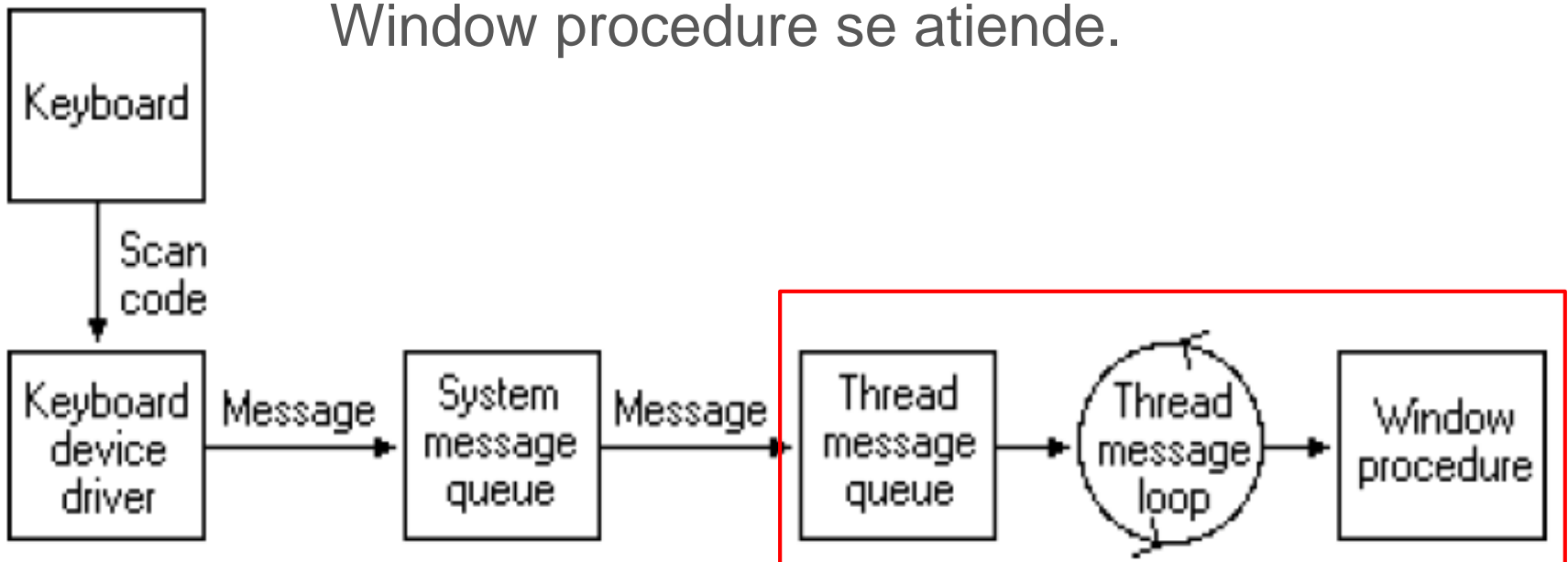


1.3 Captura y Gestión de los Eventos

El **Sistema Operativo** y los **marcos de desarrollo de aplicación (.NET, JAVA, WIN32, ...)**, se encargan de:

- Capturar los eventos y generar mensajes.
- Distribuir los mensajes hasta la aplicación

El bucle de la aplicación se lo envía a la ventana y en Window procedure se atiende.



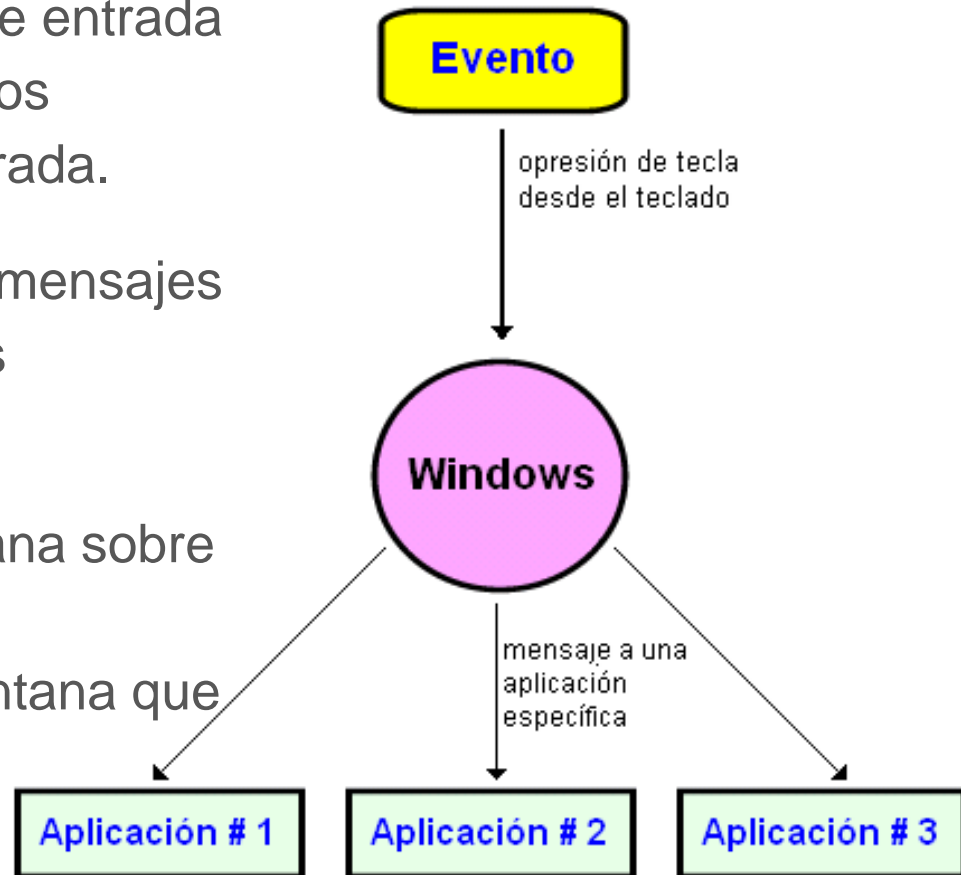
1.3 Entrada en Windows

Existe una hebra del sistema, hilo de entrada en bruto (RIT), que recibe los eventos hardware de los dispositivos de entrada.

La hebra del sistema construye los mensajes adecuados, y los envía a las hebras correspondientes:

- Los eventos del ratón a la ventana sobre la que está el puntero
- Los eventos del teclado a la ventana que tiene el foco

El procesamiento de la entrada no necesariamente se hace de forma ordenada.



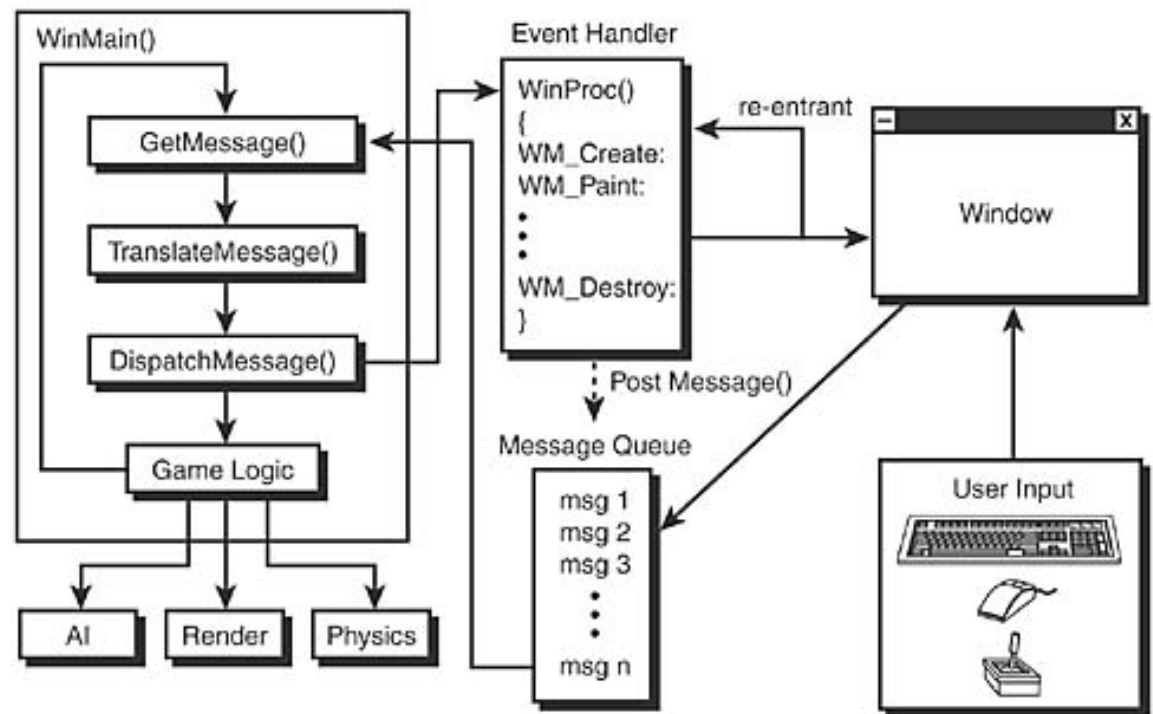
1.3 Cola de mensajes en Windows

Todas las aplicaciones tienen una cola de mensajes.

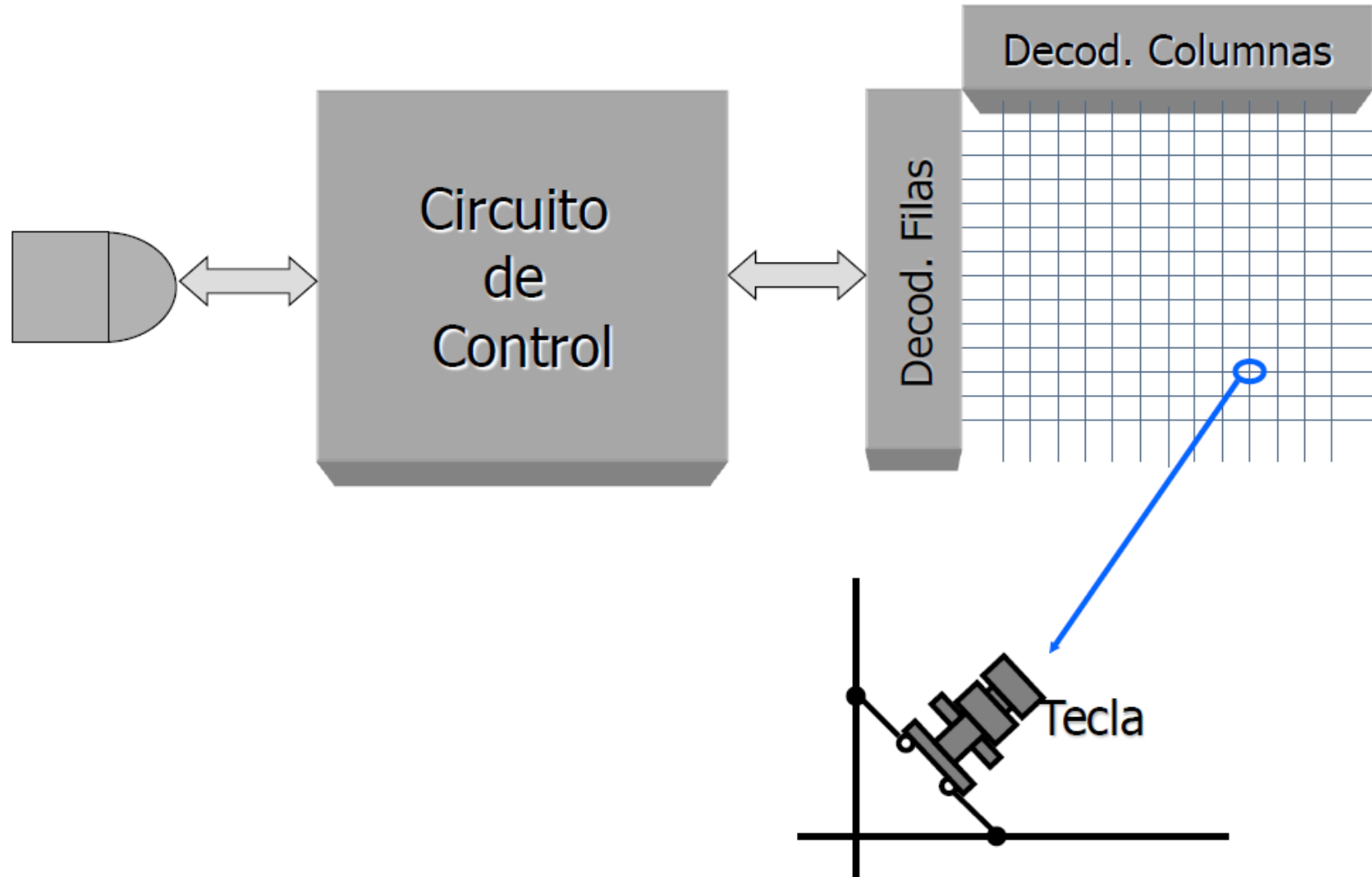
- En Win16 estaba limitada a 8 mensajes.
- En Win32 es una lista enlazada.

Si una aplicación crea varias ventanas, todos sus mensajes van a la misma cola de mensajes.

Nuestra aplicación tendrá un bucle de procesamiento de mensajes.



2. Teclado. Funcionamiento básico



2.1 Teclado. Funcionamiento básico

Se utiliza un microcontrolador (como el Intel 8048) conocido como keyboard controller.

- Rastrea la rejilla, activando las líneas para ver si hay alguna tecla pulsada.
- El microcontrolador es capaz de eliminar el “efecto rebote” de la pulsación de teclas, discriminándolo para no interpretarlo como una doble pulsación.

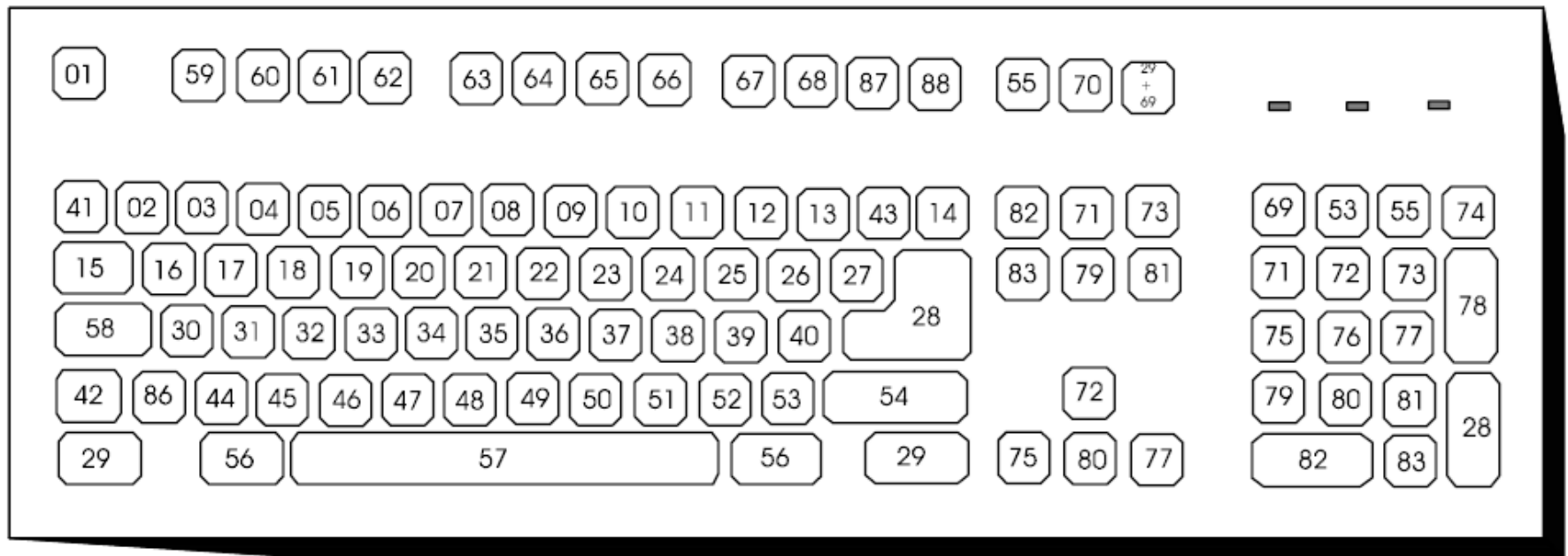
El microcontrolador se comunica con el controlador del teclado por medio de una línea serie.

- En los antiguos XT la comunicación era unidireccional
- LEDs de los XT podían no estar actualizados

2.1 Teclado. Funcionamiento básico

Existen varias disposiciones (“físicas”) de teclas comunes en todos los teclados, independientemente de los símbolos que haya en ellas.

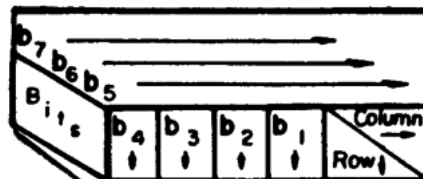
Cada tecla tiene un código distinto. Algunas tienen el mismo código, y son indistinguibles por software. El código representa una tecla, y no el carácter escrito en ella.



2.1 Teclado. No son códigos ASCII

No hay una relación definida entre código de tecla y código ASCII.
Es responsabilidad del “host” (ordenador) hacer esa traducción.

USASCII code chart



					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
					0	1	2	3	4	5	6	7
Row	b4	b3	b2	b1	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	\	p
1	0	0	0	1	SOH	DC1	!	1	A	Q	a	q
2	0	0	1	0	STX	DC2	"	2	B	R	b	r
3	0	0	1	1	ETX	DC3	#	3	C	S	c	s
4	0	1	0	0	EOT	DC4	\$	4	D	T	d	t
5	0	1	0	1	ENQ	NAK	%	5	E	U	e	u
6	0	1	1	0	ACK	SYN	&	6	F	V	f	v
7	0	1	1	1	BEL	ETB	'	7	G	W	g	w
8	1	0	0	0	BS	CAN	(8	H	X	h	x
9	1	0	0	1	HT	EM)	9	I	Y	i	y
10	1	0	1	0	LF	SUB	*	:	J	Z	j	z
11	1	0	1	1	VT	ESC	+	;	K	[k	{
12	1	1	0	0	FF	FS	,	<	L	\	l	
13	1	1	0	1	CR	GS	-	=	M]	m	}
14	1	1	1	0	SO	RS	.	>	N	^	n	~
15	1	1	1	1	SI	US	/	?	O	_	o	DEL

2.1 Teclado. Funcionamiento básico

El teclado genera dos tipos de eventos distintos:

- Pulsación de tecla
- Liberación de tecla

Para pulsación continuada reenvía el evento pulsación.

La repetición tiene dos parámetros distintos:

- Retraso: 0.25, 0.5, 0.75 o 1 segundo
- Frecuencia: entre 2 y 30 caracteres por segundo

2.2 Teclado. En MS-DOS y BIOS

El chip controlador del teclado genera los códigos y una interrupción hardware (IRQ1, la de mayor prioridad después del reloj del sistema).

El software controlador del teclado traduce los códigos a letras.

Mantiene en memoria la pulsación de teclas especiales (mayúsculas, bloqueo de mayúsculas, etc.) para traducir las pulsaciones de las teclas.

La BIOS mete el código ASCII de la tecla en el buffer del teclado.

- Si la tecla es especial (teclas de función, etc.), ocupa dos entradas en el búffer.
- Es una cola circular con 16 entradas.
- Si no son procesadas, y llega otra pulsación, la BIOS pita[ba].

2.2 Teclado. En MS-DOS y BIOS

Los programas pueden:

- Leer las teclas directamente del búffer
- Preguntar a la BIOS (también el estado de las teclas especiales)
- Pedir al sistema operativo (MS-DOS) la entrada
 - Algunas llamadas (entrada “cocida”) puede quedarse esperando hasta que se pulse INTRO, manejar automáticamente el borrar o mover flechas...

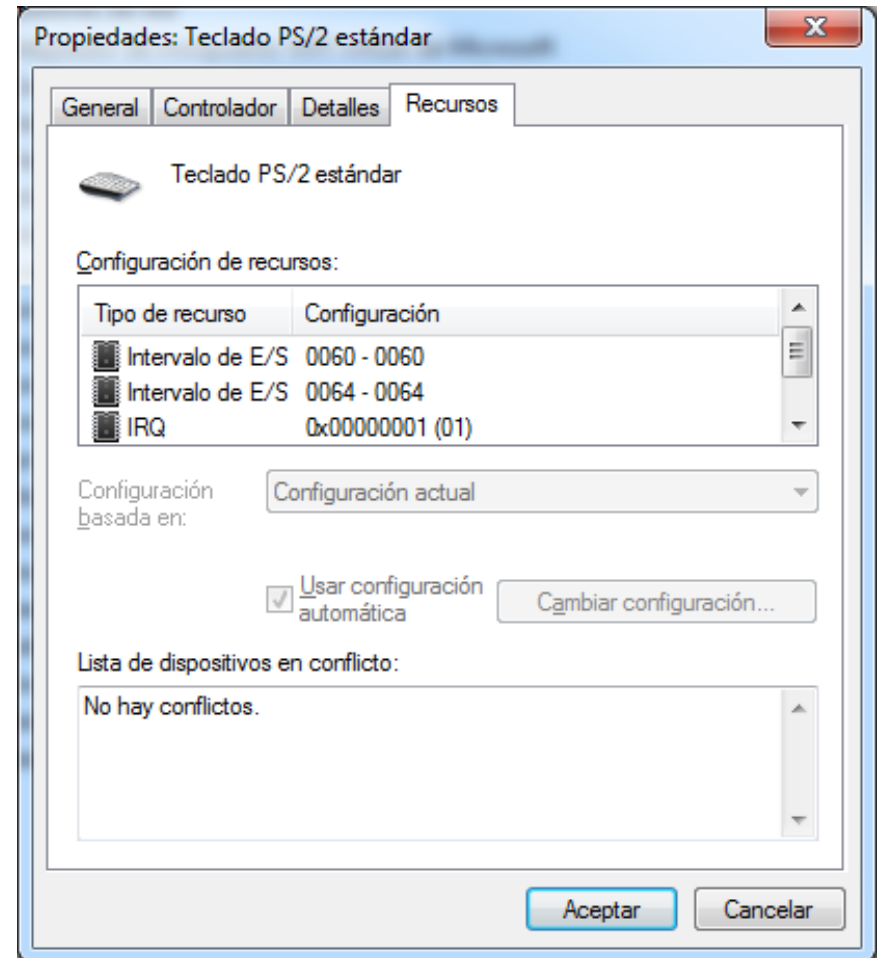
Para juegos la pulsación de teclas proporcionada por la BIOS era insuficiente:

- La repetición de las teclas era demasiado lenta
- Interesaba cuando se pulsaba y despulsaba
- Instalaban su propia rutina de gestión de la IRQ

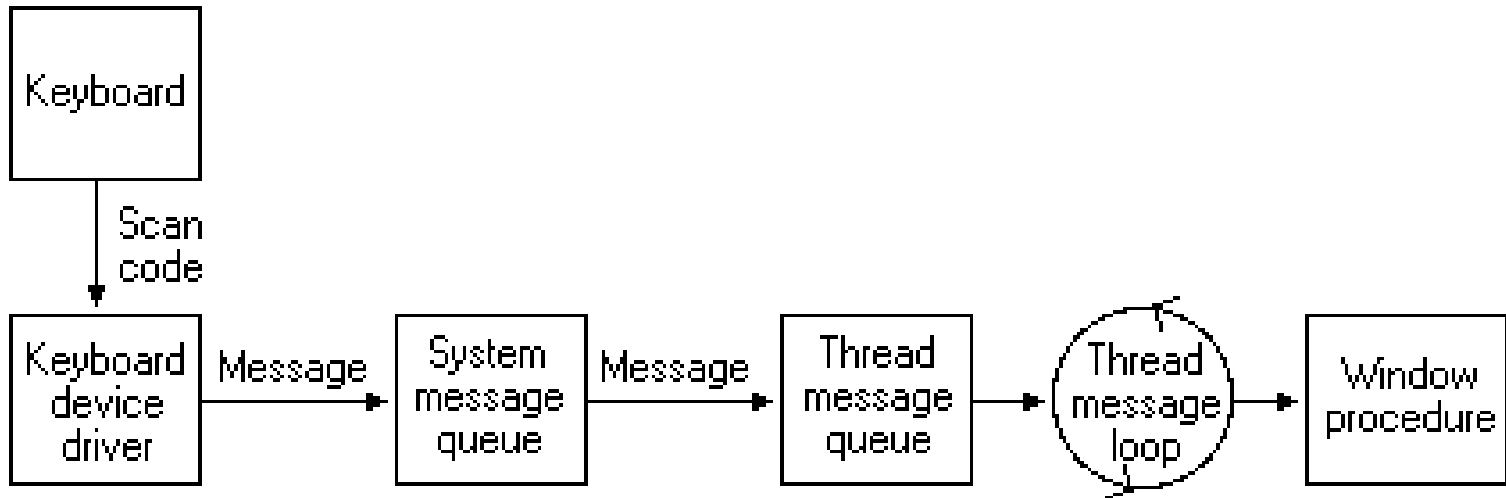
2.3 Teclado en Windows 7

Sigue utilizando la IRQ1.

El S.O. configura el mapeo de caracteres, permitiendo cambiar el idioma del teclado.



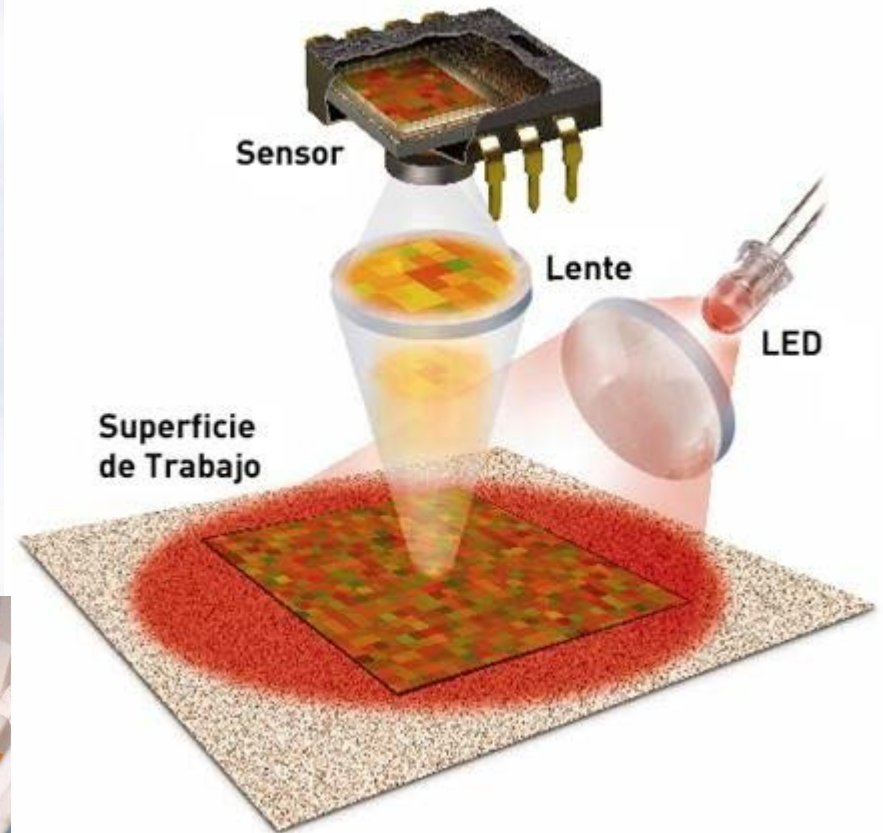
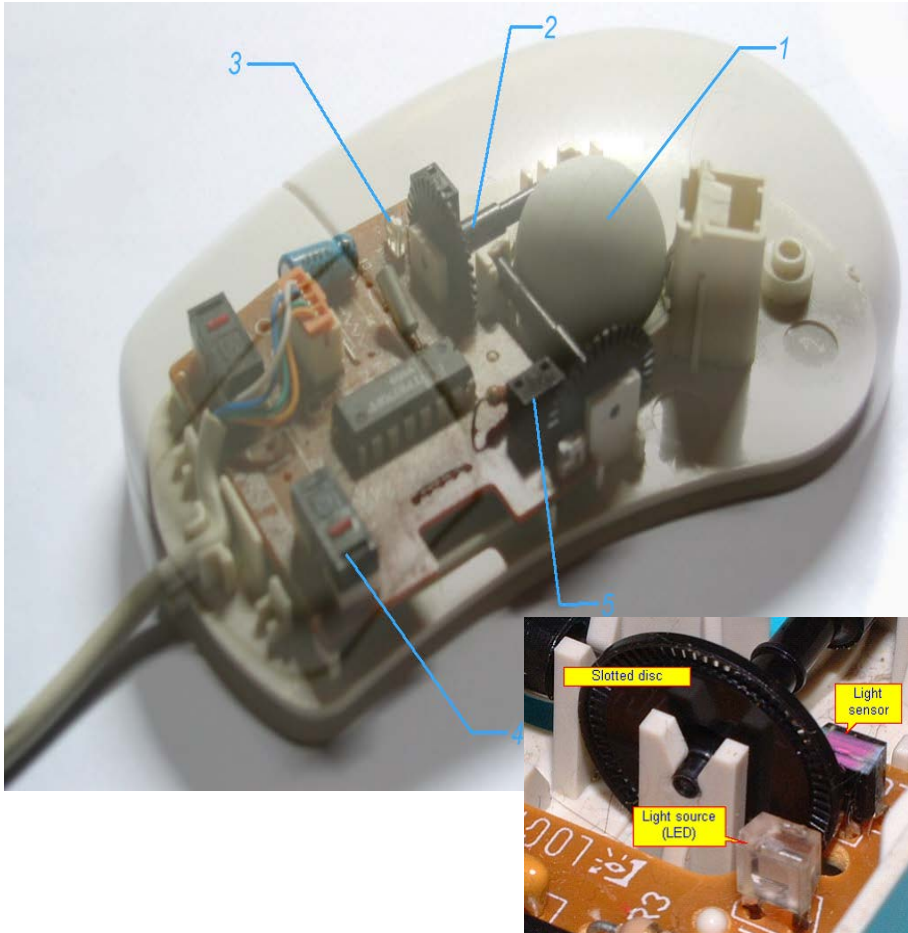
2..3 Teclado en Windows



TranslateMessage, llamado antes de procesar un mensaje, comprueba si es una pulsación de tecla y la traduce a mensajes del estilo *WM_CHAR*.

DispatchMessage, llama a la función encargada de procesar los mensajes, normalmente *WindowsProc*.

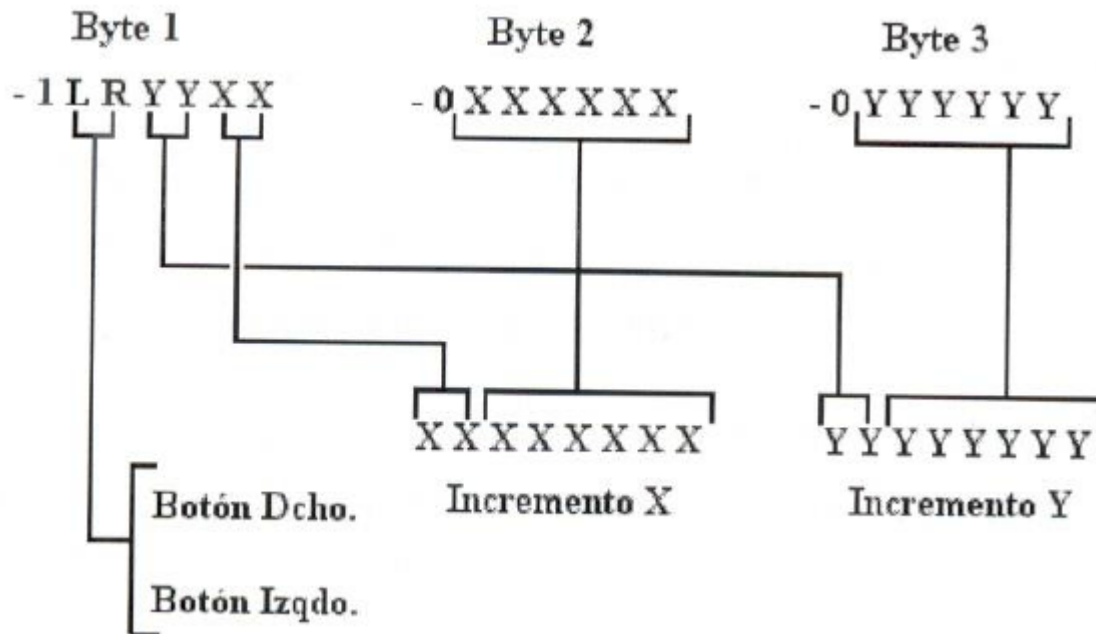
3.1 Ratón. Funcionamiento básico



3.1 Ratón. Funcionamiento básico

Cada vez que cambia el estado, el ratón manda por la línea serie un conjunto de datos.

- Desplazamiento vertical y horizontal (medido en mickies)
- Estado de los botones (pulsado / despulsado)



**Protocolo de Microsoft
para los primeros ratones
de dos botones**

3.2 Ratón. En MS-DOS

Cuando el ratón envía datos por la línea serie, se produce una interrupción hardware (IRQ3 o IRQ4), capturada por el controlador del ratón.

Permitía establecer la sensibilidad (relación entre mickeys y píxeles)

El controlador del ratón es más complicado que el del teclado.

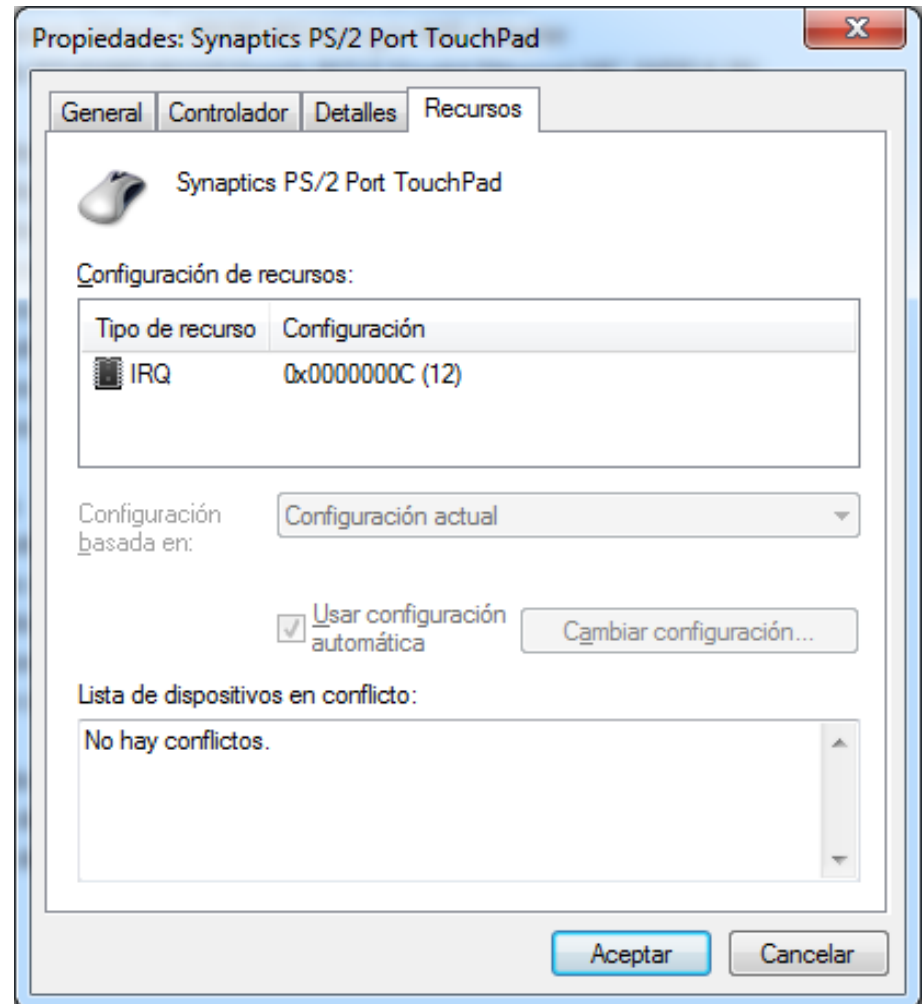
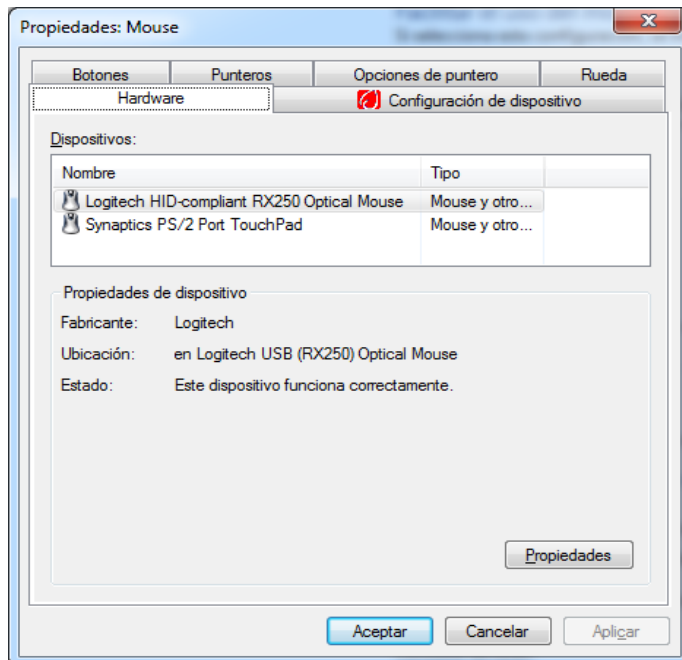
- En modo texto, hace video inverso
- En modo gráfico muestra un gráfico de puntero
- Hay que cuidarse de los posibles cambios debajo del cursor

Se podía registrar una función para ser llamada cada vez que ocurría un evento.

3.3 Ratón en Windows 7

Puede utilizar otras puertos y otras interrupciones.

El driver permite configurar el comportamiento.



3.3 Ratón en Windows

Una ventana recibe todos los eventos del ratón que ocurren encima de ella, independientemente de si tiene el foco.

Inicialmente los eventos posibles son:

- Movimiento del ratón
- Pulsación o liberación de alguno de los tres botones del ratón

Además, el mensaje generado tiene un campo con flags que indican el estado de las teclas de control y mayúsculas, y el estado de los botones.

Se distingue entre eventos que suceden en el área cliente o fuera de ella.

- Nosotros procesaremos los eventos del área cliente.
- Los otros serán procesados por el S.O.

3.3 Ratón en Windows. Rueda del Ratón

La rueda del ratón en Windows 95 no estaba integrada con el gestor de ventanas (mensaje específico del driver).

En Windows 98 ya se integró con el mensaje *WM_MOUSEWHEEL*.

La sensibilidad de la rueda depende del ratón.

- Algunos drivers devuelven el movimiento en bloques grandes, mientras otros lo hacen en bloques pequeños

En juegos, el diseño del interfaz no debe exigir el uso de la rueda:

- En portátiles no está disponible
- Se pueden asignar teclas con la misma acción

Aún así, el juego debería soportar la rueda. No es vital, pero el jugador medio lo espera

5.3 Ratón en Windows. Captura del Ratón

A veces interesa capturar el movimiento cuando está fuera de la ventana

- Botones pulsados
- Drag and Drop

Es posible hacerlo con *SetCapture*

La hebra sigue recibiendo mensajes hasta que:

- Llama a *RemoveCapture*
- El ratón es capturado por otra ventana
- El usuario pulsa sobre otra ventana

Solo una ventana activa puede capturar el ratón

5.3 Ratón en Windows. Captura del Ratón

La programación del Drag and Drop requiere:

- Detectar la pulsación del ratón => Activa la captura
- Detectar si el movimiento del ratón sobrepasa un umbral => Activación del Drag and Drop
- Manejo del evento de movimiento => Dibujando los objetos correspondientes
- Detectar la liberación del botón => Terminar el desplazamiento, liberar la captura del ratón y realizar la acción deseada dependiendo de la posición final.

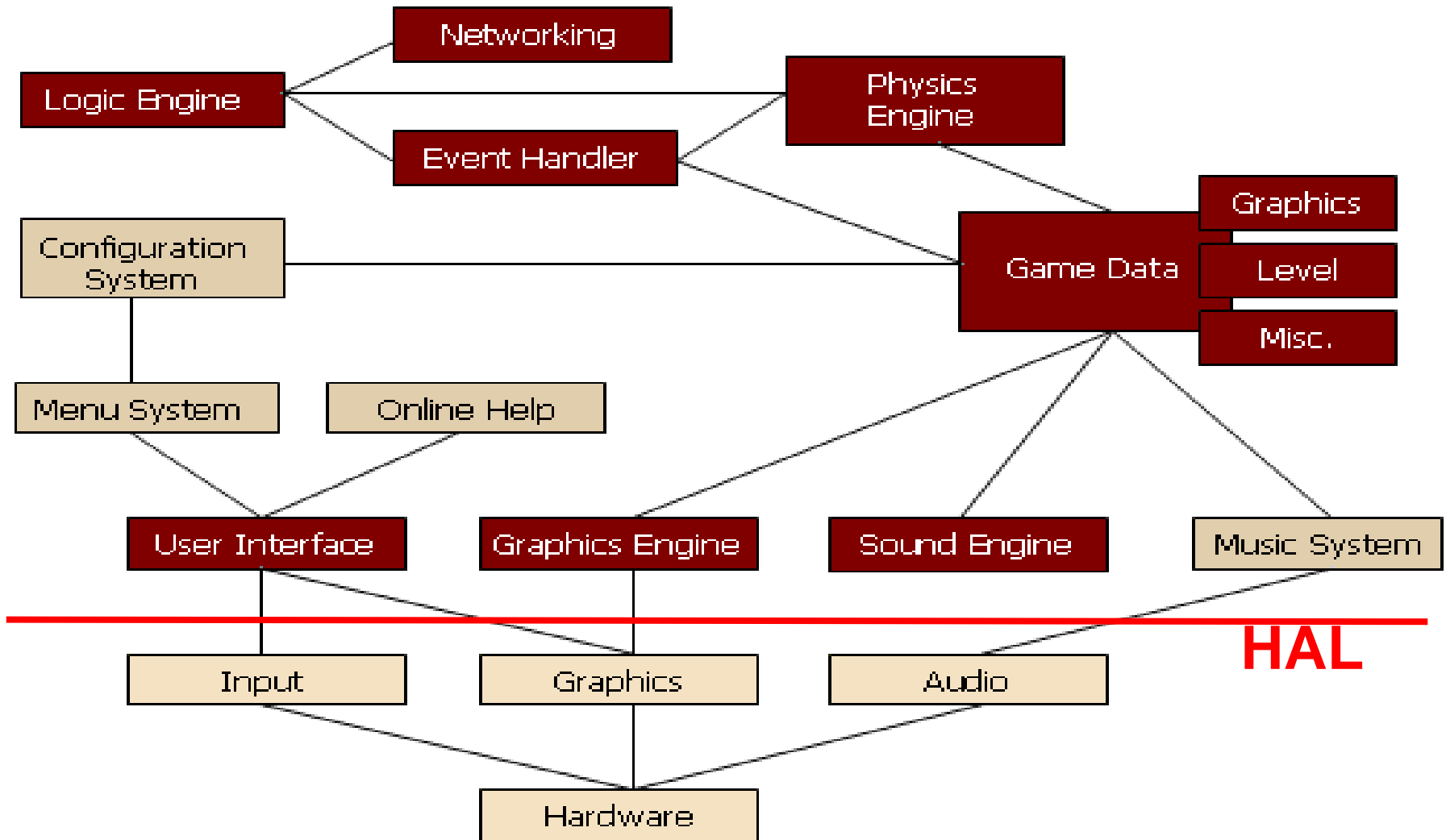
4 Capa de Abstracción del Hardware (HAL) de E/S

La capa de abstracción de hardware (Hardware Abstraction Layer o HAL) es un elemento del sistema operativo que funciona como **una interfaz entre el software y el hardware del sistema**, proveyendo una plataforma de hardware consistente sobre la cual corren las aplicaciones.

- Cuando se emplea una HAL, **las aplicaciones no acceden directamente al hardware sino que lo hacen a la capa abstracta provista por la HAL.**
- Del mismo modo que las API, las HAL permiten que las **aplicaciones sean independientes del hardware** porque abstraen información acerca de tales sistemas, como lo son las cachés, los buses de E/S y las interrupciones, y usan estos datos para darle al software una forma de interactuar con los requerimientos específicos del hardware sobre el que deba correr.

4 Capa de Abstracción del Hardware (HAL) de E/S

Los motores de videojuegos crean sus propias HAL para portabilidad.



5 Separación de Diseño y Programación

Hoy en día se separan incluso los lenguajes de programación:

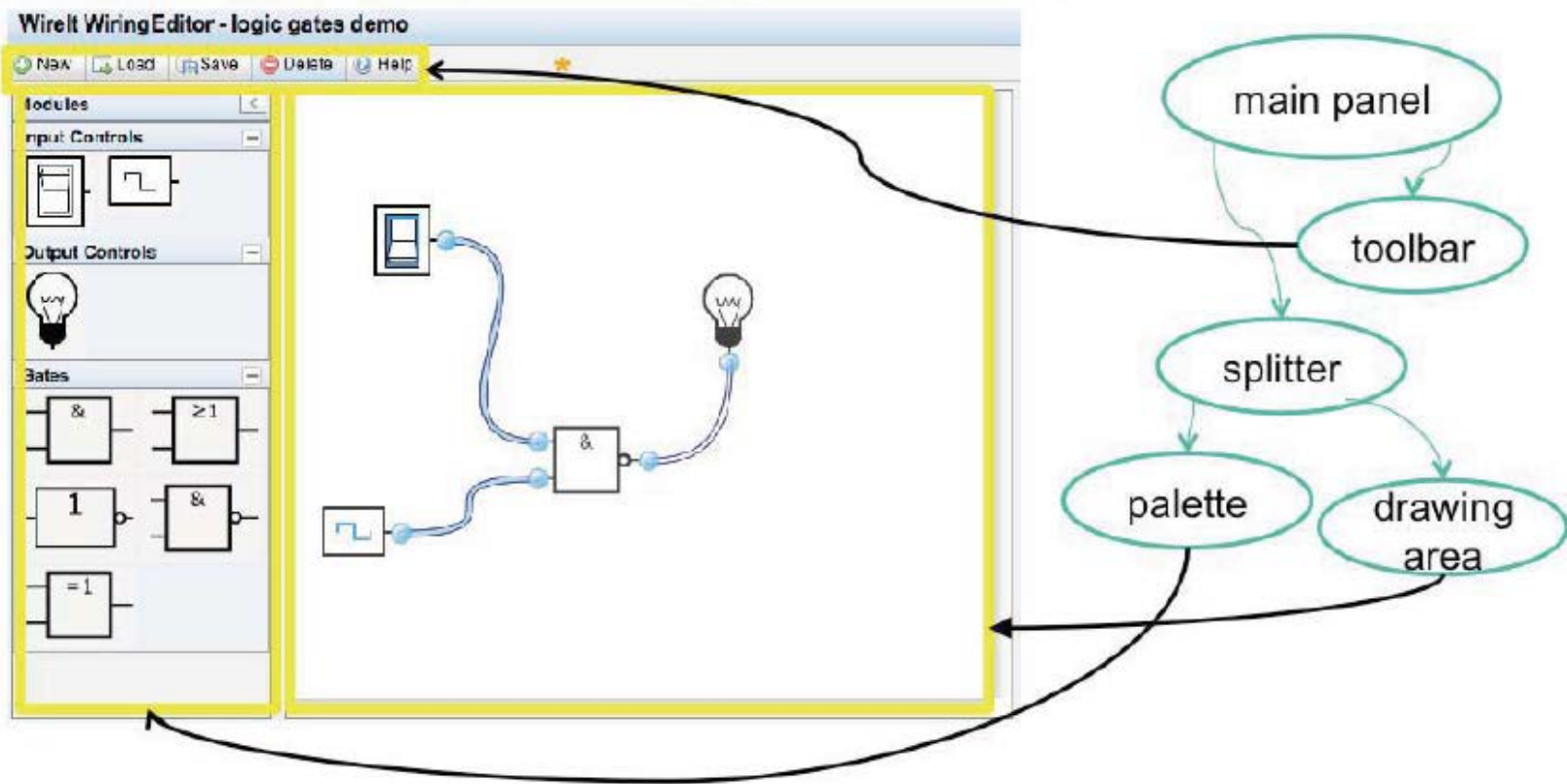
- El Diseñador utiliza un lenguaje marcas (p.e. HTML, XML, XMLA, ...) y estilos de diseño.
- El Programador utilizar un lenguaje más flexible y potente (p.e. JavaScript, C++, C#, Java,...).

Esta separación de lenguajes es habitual en las tecnologías actuales:

- WPF
- Universal Windows Platform (UWP)
- Android Studio

5.1 Programación del GUI. View Tree

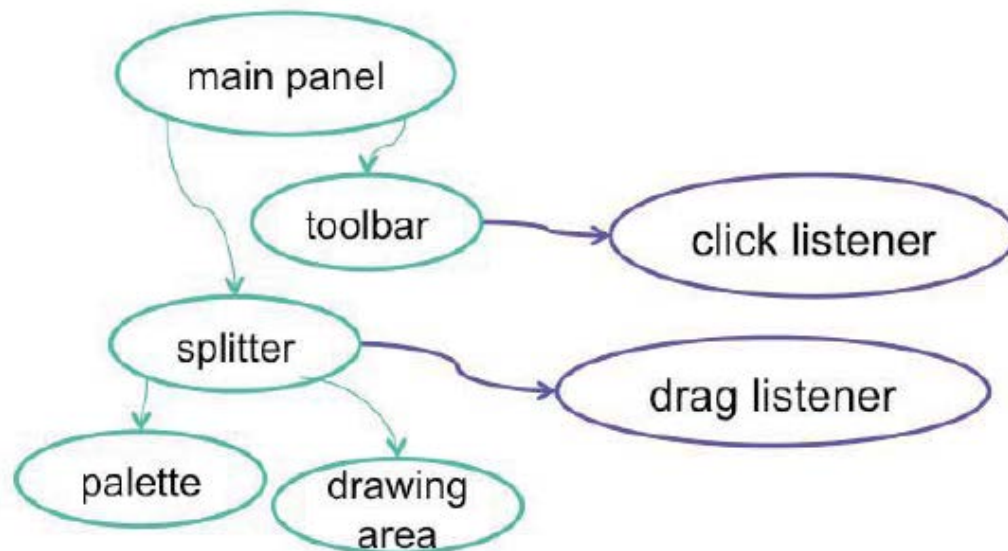
Un GUI se puede estructurar como un árbol de Vistas.



Cada *Vista* es un objeto que se muestra por sí solo en una región.

5.1 Programación GUI. Recorridos del *View Tree*

- *Layout (Diseño)*: Recorre el árbol calculando el diseño de las vistas (posiciones, tamaños, ...) en función de las propiedades de la ventana/pantalla.
- *Salida (Representación)*: La jerarquía del árbol dirige el proceso de representación, pero cada *Vista* es responsable de redibujarse.
- *Entrada (Captura)*: El GUI recibe la entrada de teclado, ratón ... Las vistas del árbol tienen asociado manejadores de eventos.



5.2 Lenguajes de marca

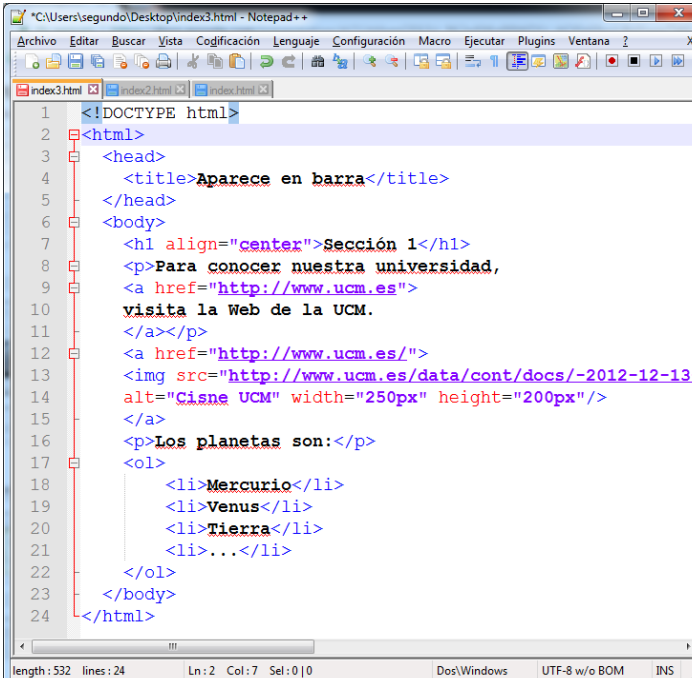
Los lenguajes de marcas, como HTML, XML, ..., constan de:

- **texto**: que define los contenidos reales
- **marcas (etiquetas o tags)**: que permiten dar significado al texto o aplicar algún tratamiento especial sobre este.

Permiten la agrupación en forma de árbol.

Los navegadores/frame-works recorren el árbol interpretando las marcas:

- Construyendo/actualizando los objetos gráficos
- Registrando los “callbacks” de respuesta a eventos



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Aparece en barra</title>
5   </head>
6   <body>
7     <h1 align="center">Sección 1</h1>
8     <p>Para conocer nuestra universidad,
9     <a href="http://www.ucm.es">
10      visita la Web de la UCM.
11    </a></p>
12    <a href="http://www.ucm.es/">
13    
15    </a>
16    <p>Los planetas son:</p>
17    <ol>
18      <li>Mercurio</li>
19      <li>Venus</li>
20      <li>Tierra</li>
21      <li>...</li>
22    </ol>
23  </body>
24 </html>
```

5.2 Lenguajes de marca

*C:\Users\segundo\Desktop\index3.html - Notepad++

Archivo Editar Buscar Vista Codificación Lenguaje Configuración Macro Ejecutar Plugins Ventanas

index3.html index2.html index.html

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Aparece en barra</title>
5   </head>
6   <body>
7     <h1 align="center">Sección 1</h1>
8     <p>Para conocer nuestra universidad,
9     <a href="http://www.ucm.es">
10    visita la Web de la UCM.
11    </a></p>
12    <a href="http://www.ucm.es/">
13    
16    <p>Los planetas son:</p>
17    <ol>
18      <li>Mercurio</li>
19      <li>Venus</li>
20      <li>Tierra</li>
21      <li>...</li>
22    </ol>
23  </body>
24 </html>
```

length: 532 lines: 24 Ln: 2 Col: 7 Sel: 0 | 0 Dos\Windows UTF-8 w/o BOM INS

SEGUNDO


Aparece en barra

file:///C:/Users/segundo/Desktop/index3.html

16 Calendario Mi unidad sesteban segu222G Otros marcadores

Sección 1

Para conocer nuestra universidad, [visita la Web de la UCM](http://www.ucm.es).



UNIVERSIDAD COMPLUTENSE
MADRID

Los planetas son:

1. Mercurio
2. Venus
3. Tierra
4. ...

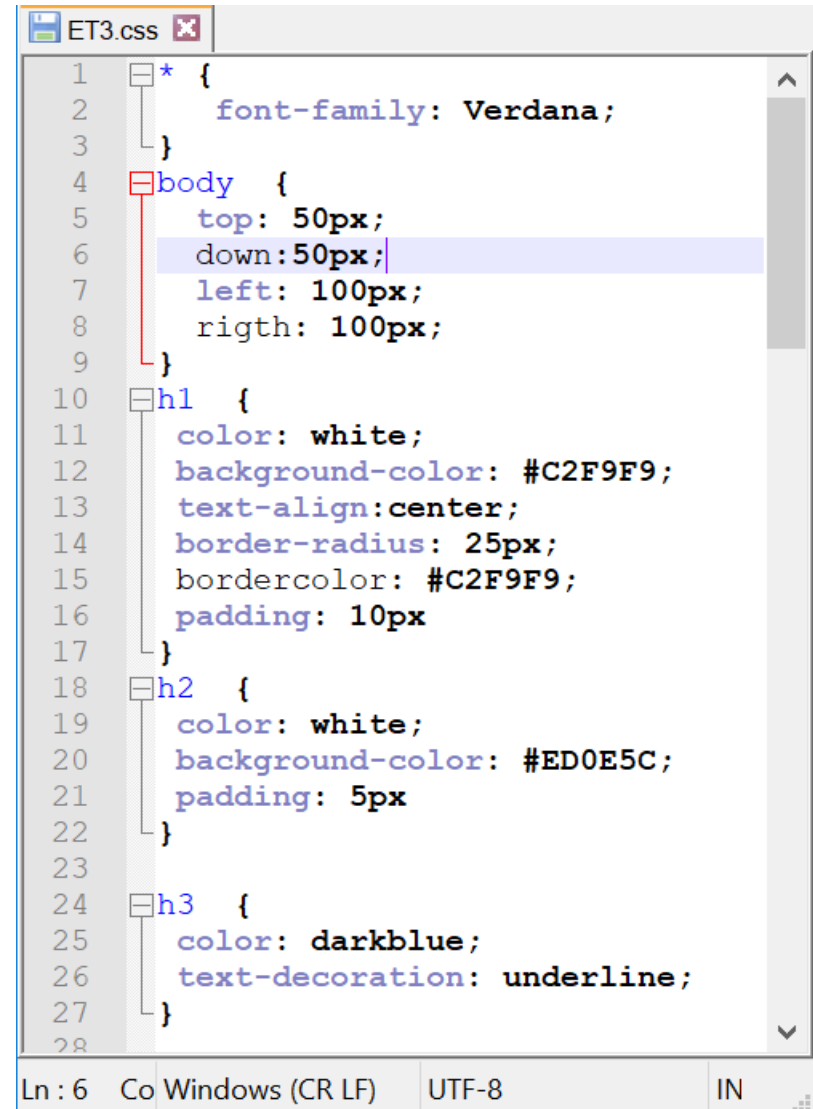
5.3 Lenguajes de estilos

Los lenguaje de marcas permiten utilizar estilos de diseño, que define la presentación de los documentos, p.e. CSS para HTML.

CSS es un lenguaje de estilo que define la presentación de los documentos HTML.

Cada uno tiene su función:

- HTML se usa para estructurar el contenido;
- CSS se usa para formatear el contenido previamente estructurado.



```
1  * {
2      font-family: Verdana;
3  }
4  body {
5      top: 50px;
6      down: 50px;
7      left: 100px;
8      righth: 100px;
9  }
10 h1 {
11     color: white;
12     background-color: #C2F9F9;
13     text-align: center;
14     border-radius: 25px;
15     bordercolor: #C2F9F9;
16     padding: 10px
17 }
18 h2 {
19     color: white;
20     background-color: #ED0E5C;
21     padding: 5px
22 }
23
24 h3 {
25     color: darkblue;
26     text-decoration: underline;
27 }
28
```

Ln : 6 Co Windows (CR LF) UTF-8 IN

6 Patrones de Arquitectura software de GUI

Problema: ¿Cómo modularizar la funcionalidad de la interfaz de usuario de una aplicación de tal forma que usted pueda modificar fácilmente sus partes individuales?

Patrones de arquitectura software:

- MVC (*Model-View-Controller*)
- MVVM (*Model-View-ViewModel*)

Son patrones de arquitectura del software que sirven para separar la funcionalidad de la aplicación del interfaz de usuario.

El objetivo es facilitar el desarrollo por separado de ambos aspectos, incrementando la reutilización y flexibilidad.

6.1 MVC (Model-View-Controller)

Descrito por primera vez en 1979 para Smalltalk

<http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html>

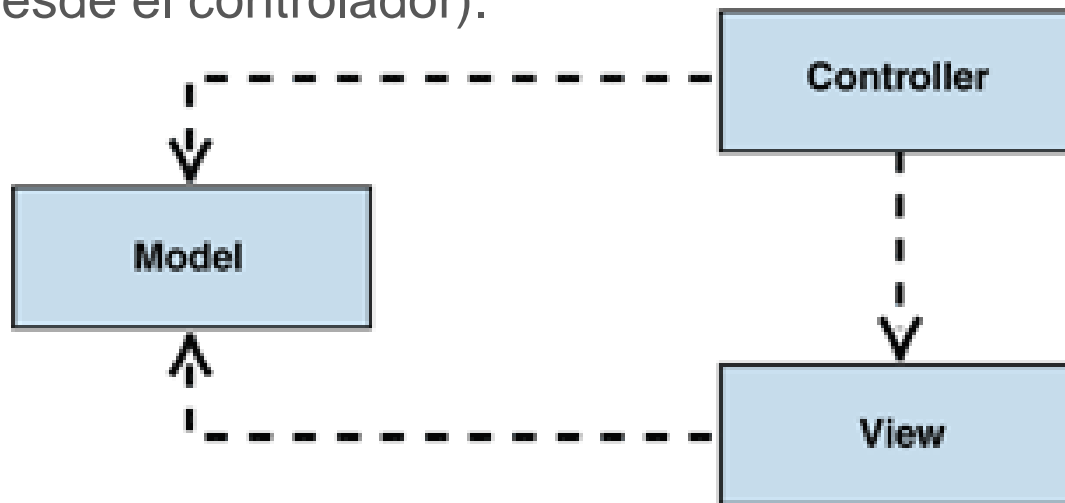
Utilizado en múltiples frameworks:

- Java Enterprise Edition (J2EE),
- XForms (Formato XML estándar del W3C)
- GTK+ (toolkit creado por Gnome para X Window)
- **ASP.NET MVC Framework (Microsoft)**
- Google Web Toolkit (GWT)
- Apache Struts (framework para aplicaciones web J2EE)
- Ruby on Rails (framework para aplicaciones web con Ruby)
- ...

6.1 MVC (Model-View-Controller)

El Modelo-Vista-Controlador (MVC) patrón separa el modelo de la aplicación, la presentación y la entrada del usuario en tres clases separadas [[Burbeck92](#)]:

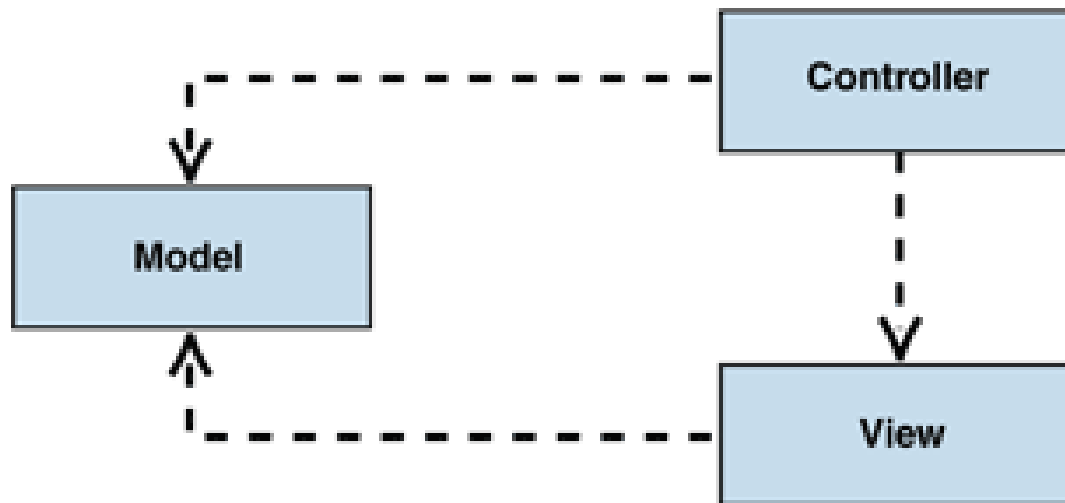
- El modelo maneja el comportamiento y los datos del dominio de aplicación, responde a las solicitudes de información sobre su estado (por lo general desde el punto de vista), y responde a las instrucciones para cambiar el estado (por lo general desde el controlador).



6.1 MVC (Model-View-Controller)

Las vistas y los controladores suelen estar muy relacionados. Los controladores tratan los eventos que se producen en la interfaz gráfica (vista).

- La vista gestiona la visualización de la información.
- El controlador interpreta las entradas de teclado y ratón por parte del usuario, informando al modelo y / o la vista para cambiar según sea apropiado.



6.1 MVC (Model-View-Controller)

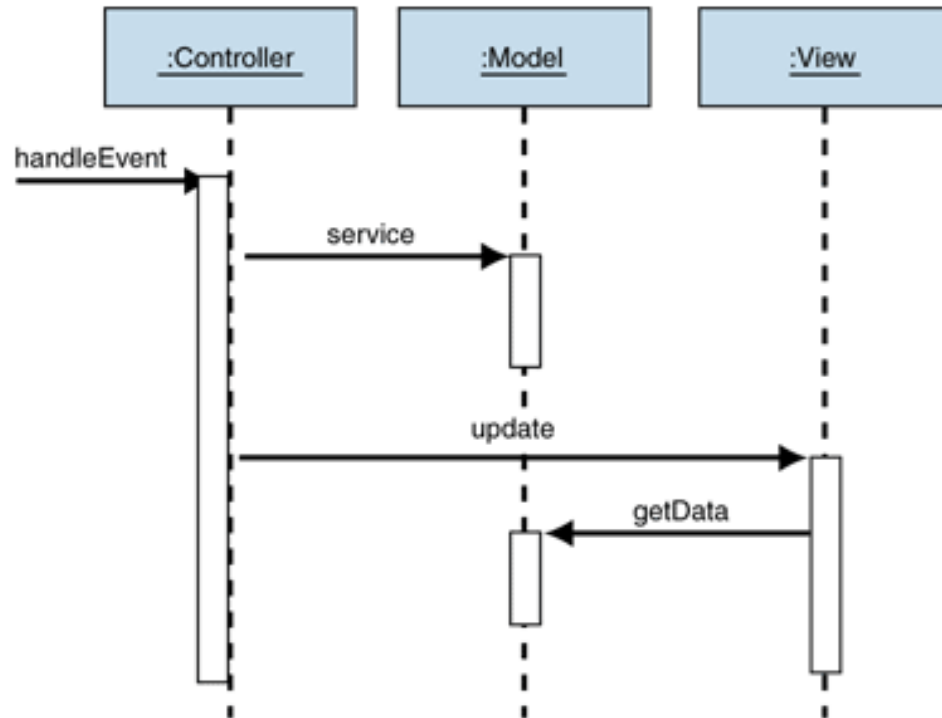
Es importante tener en cuenta que tanto la vista y el controlador dependen del modelo. Sin embargo, el modelo no depende de ni la vista ni el controlador. Esta es una de las principales ventajas de la separación. Esta separación permite que el modelo que se construya y pruebe independiente de la presentación visual.

La separación entre la vista y el controlador es secundaria en muchas aplicaciones cliente, y, de hecho, muchos marcos de la interfaz de usuario implementan las funciones como un objeto.

En las aplicaciones Web, por otra parte, la separación entre la vista (el navegador) y el controlador (los componentes de servidor que controlan la solicitud de HTTP) está muy bien definida.

6.1 MVC (Pasivo)

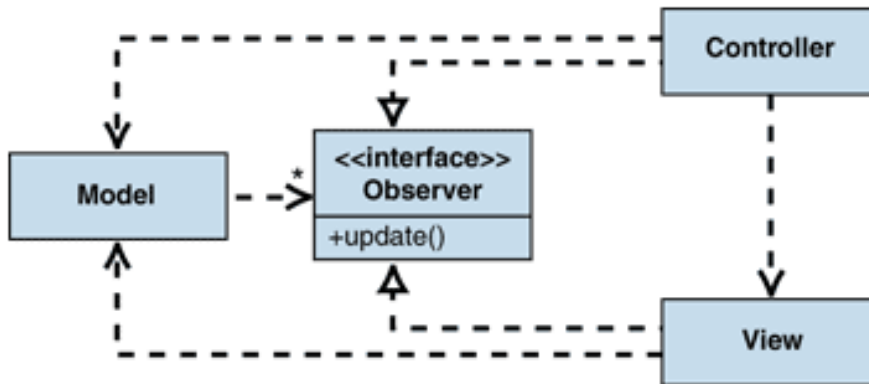
Se emplea el modelo pasivo cuando un controlador manipula el modelo exclusivamente. El controlador modifica el modelo y luego informa a la vista de que el modelo ha cambiado y debe ser renovado.



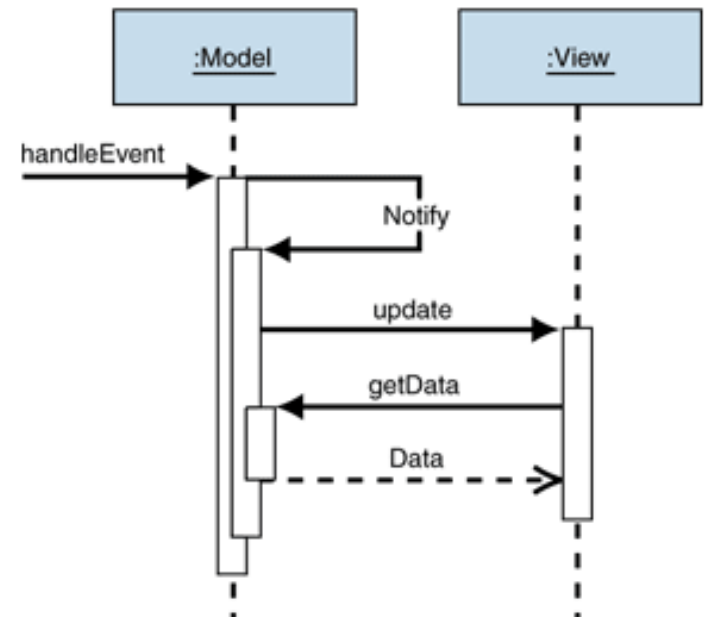
6.1 MVC (Activo)

Se utiliza cuando el modelo cambia de estado sin la participación del controlador.

Esto puede ocurrir cuando otras fuentes están cambiando los datos y los cambios deben reflejarse en las vistas. Los puntos de vista individuales implementan el Observador interfaz y se registran con el modelo. El modelo de seguimiento de la lista de todos los observadores que se suscriben a los cambios. Cuando un modelo cambia, se repite el modelo a través de todos los observadores registrados y les notifica del cambio.



Este enfoque es a menudo llamado "publicación-suscripción."



6.1 MVC (Ventajas)

- **Es compatible con múltiples puntos de vista.** Debido a que la vista se separa del modelo y no hay dependencia directa del modelo a la vista, la interfaz de usuario puede mostrar varias vistas de los mismos datos al mismo tiempo.
- **Tiene capacidad para el cambio.** Requisitos de la interfaz de usuario tienden a cambiar más rápidamente que los modelos de negocio. Los usuarios pueden preferir diferentes colores, fuentes, diseños de pantalla, y los niveles de apoyo a nuevos dispositivos, como teléfonos móviles o PDAs. Debido a que el modelo no depende de los puntos de vista, la adición de nuevos puntos de vista para el sistema en general, no afecta el modelo.

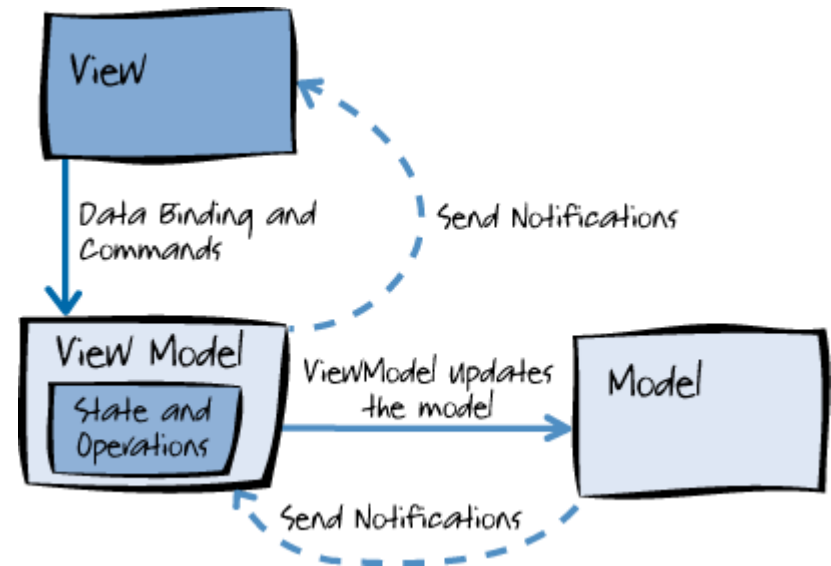
6.1 MVC (Desventajas)

- **Complejidad.** El MVC patrón introduce nuevos niveles de indirección y por lo tanto aumenta la complejidad de la solución ligeramente. También aumenta la naturaleza orientada a eventos del código de interfaz de usuario, que puede ser más difícil de depurar.
- **Costes de actualizaciones frecuentes.** Desacoplar el modelo desde el punto de vista no significa que los desarrolladores del modelo puedan ignorar la naturaleza de las vistas. Por ejemplo, si el modelo se somete a cambios frecuentes, podría inundar los puntos de vista con las solicitudes de actualización.

6.2 MVVM (Model-View-ViewModel)

La Arquitectura Model–View–View Model es una evolución del MVC. Facilita la separación del desarrollo de la interfaz gráfica de usuario frente al desarrollo de la lógica de la aplicación o del modelo de datos.

MVVM fue desarrollado por Ken Cooper y Ted Peters (Microsoft) en 2005 para simplificar la programación orientada a eventos de las interfaces de usuario, explotando las características de Windows Presentation Foundation (WPF).

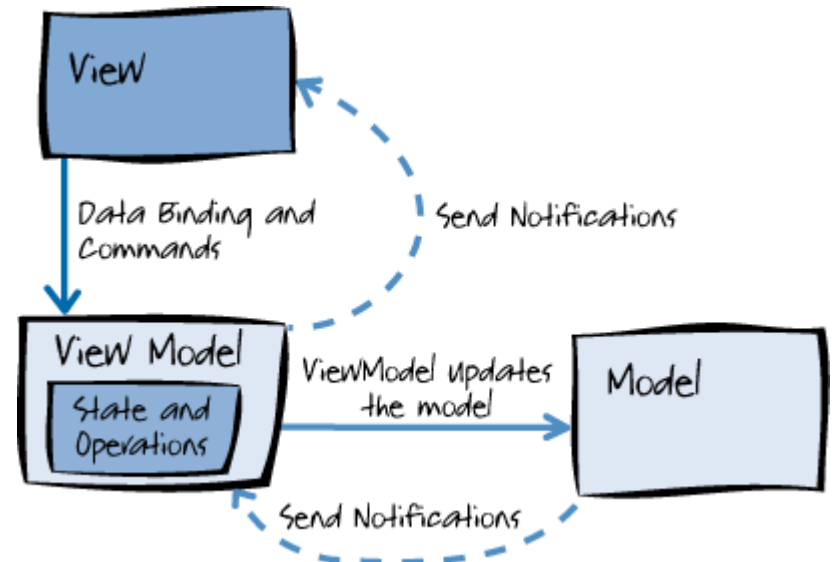


6.2 MVVM (Model-View-ViewModel)

En el **Modelo** (Model): Se establece la Lógica de la aplicación y los Datos. Provee eventos y notificaciones.

En la **Vista** (View): se realizará únicamente lo relacionado con Ventanas, controles UI, layouts y estilos.

La **VistaModelo** (ViewModel) es un convertidor de valores; que significa que el modelo de vista es responsable de la exposición (conversión) de los objetos de datos del modelo, de tal manera que los objetos se presenten y y actualicen fácilmente.



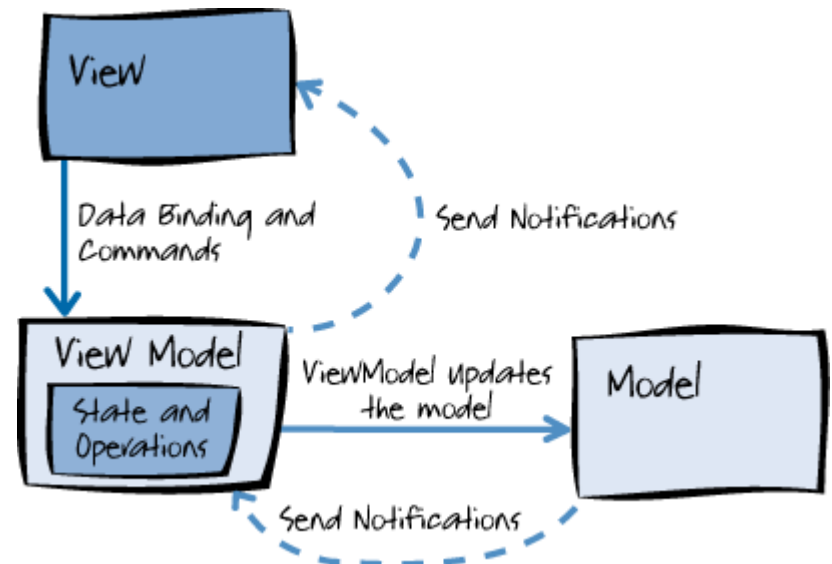
6.2 MVVM (Model-View-ViewModel)

El modelo de vista actúa como intermediario entre la vista y el modelo, y es responsable de manejar la lógica de la vista.

Por lo general, el modelo de vista interactúa con el modelo mediante la invocación de métodos en las clases del modelo.

El modelo de vista a continuación proporciona datos del modelo en una forma que la vista se puede utilizar fácilmente.

Por ejemplo, cuando un usuario hace clic en un botón en la interfaz de usuario se puede desencadenar la acción de un comando en el modelo de vista.



6.2 MVVM (Ventajas)

- **Portabilidad:** Tan solo hay que generar una Vista diferente para cada plataforma. Algunas herramientas generan las vistas automáticamente: Unity, Xamarin, ...
- **Mantenibilidad:** Una separación limpia de diferentes tipos de código debería hacer más fácil hacer cambios. Esto significa que puede mantenerse de forma ágil y actualizarse a nuevas versiones de forma rápida.
- **Pruebas unitarias:** Con MVVM cada pieza de código es más granular. Eso hace que sea mucho más fácil de escribir pruebas unitarias.
- **Extensibilidad:** Se pueden sustituir o añadir nuevas piezas de código que pueden hacer cosas similares en los lugares correctos en la arquitectura.

6.2 MVVM (Desventajas)

- Para interfaces de usuario sencilla, MVVM puede ser un **trabajo excesivo**.
- En aplicaciones grandes puede ser **difícil de diseñar** los modelo de vista.
- La **depuración puede ser difícil** cuando tenemos enlaces de datos complejos en ViewModel.
- **Convertir los datos** de forma automática es un trabajo no trivial y supone una sobrecarga de la CPU.

6.2 Ejemplo MVVM (Xamarain)

- MVVM es un patrón de diseño idóneo para el desarrollo de aplicaciones multiplataforma, pues separa claramente el Modelo (independiente de la plataforma) de la Vista (que es dependiente de la plataforma).

