

Crear un servicio de mensajería con las librerías DirectPlay de DirectX8

2.004, Jose Luis Monte Galiano, (MoGa)

jlmogasa@hotmail.com

Índice.

Introducción.....	2
Requisitos de software.	3
Implementar el servidor.	3
q Inicialización / Puesta en marcha.....	3
q Recepción y tratamiento de mensajes de los clientes.....	4
q Finalización.....	5
q Vinculación del servidor con el formulario de la aplicación. Tratamiento de los eventos del servidor.....	5
Implementar el cliente.	7
q Inicializar el cliente.	7
q Conexión con el servidor.	7
q Envío y recepción de mensajes.	8
q Desconexión.....	8
Como instalar/Usar los ejemplos.....	9
q Arranque del servidor.....	9
q Arranque del cliente MicroXat.	10
q Arranque del cliente FM Messenger.	12
Bibliografía.....	15

Introducción.

En este artículo explicaremos como utilizar las funciones **DirectPlay** de las librerías de **DirectX8**. DX8 son un conjunto de librerías de Microsoft especialmente creadas para la generación de gráficos y animación. Posteriormente añadieron funciones para el tratamiento de juegos multijugador. Con esto nació **DirectPlay**.

Con DirectPlay ahora era posible crear juegos que obedecieran comandos remotos dados por otros jugadores. Usando como pasarela Internet.

Pero podemos enfocar la potencia que nos ofrece DirectPlay para otros propósitos. Aprovechando el uso que hace de Internet para establecer comunicaciones entre diferentes máquinas, podemos crear aplicaciones profesionales de muchos tipos. No únicamente juegos. Aquí proponemos crear un servicio de mensajería instantánea. Lo mas parecido a **Messenger** de Microsoft; pero creado por nosotros y hecho a nuestra medida.

En las fuentes de ejemplo adjuntas a este artículo encontraremos dos tipos de servicios de mensajería implementados con DirectPlay.

FMMessenger : Es lo mas parecido al Messenger de Microsoft. Obviamente no es igual ni tiene las mismas características ni funciones. Pero sirva como ejemplo.

MicroXat : Implementación de un servicio de chat con DirectPlay.

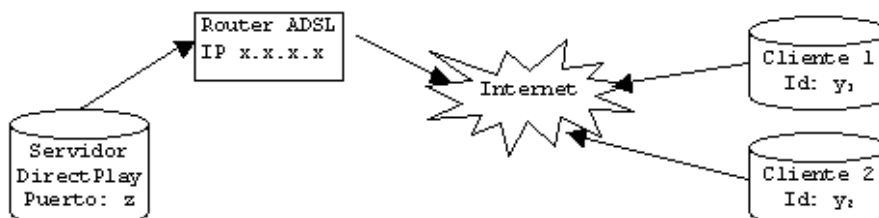
Y un servidor de DirectPlay:

Servidor PDX8_Chat : Las comunicaciones con DirectPlay 8 se basan en que un “servidor del servicio” centraliza las comunicaciones. El servidor no es mas que un pequeño programa que “escucha” un puerto. El puerto a escuchar es prácticamente indiferente. Únicamente hay que tener la precaución de que :

- No interfiera con otras aplicaciones.
- Si vuestras comunicaciones pasan por un modem-router o proxy. Vuestro sistema debe permitir el tratamiento a través de dicho puerto.
- Las aplicaciones clientes deben “escribir” en ese puerto.

En las pruebas que se han realizado con estas fuentes se usa el puerto 4000.

Un ordenador con IP conocida debe ejecutar el servidor. Con el servidor en marcha y escuchando un puerto dado; los clientes deben acceder a la IP del servidor, y emitir a través de dicho puerto. Gráficamente sería :



Los clientes acceden al servidor usando la IP x.x.x.x, y a través del puerto z. El servidor “escucha” el puerto z y, al recibir datos, actúa en consecuencia.

¿Que es lo que tiene que hacer el servidor al recibir un mensaje de un cliente?. Eso lo determina el propio mensaje, que tiene un formato determinado y conocido por el servidor. Ese formato no está establecido, y lo determinaremos nosotros en la implementación tanto del cliente como del servidor. El mensaje en si es una orden para el servidor.

DirectPlay identifica a cada cliente en ejecución asignándole una ID, (y_n), que es única para cada cliente mientras dure la ejecución del servidor; que es quien controla la asignación de dichas ID. El servidor también viene identificado con una ID.

Requisitos de software.

Para crear nuestros programas con DirectX8, necesitaremos obviamente del entorno de DirectX8. Desde la web de Microsoft podemos bajarnos la instalación y actualizar nuestro sistema. Especial atención al archivo **DX8VB.DLL**. El entorno de Visual Basic 6 necesita de este archivo, así como la instalación de nuestros programas en otros ordenadores. En nuestros programas con DirectPlay necesitaremos agregar como referencia este archivo, que usualmente se encuentra en Windows\System

Implementar el servidor.

Un servidor es un programa que usa las librerías de DirectPlay de servidor. El servidor mantiene en “escucha” un puerto determinado. Recibe mensajes de los clientes que usan sus servicios. Paralelamente informa a dichos clientes sobre el estado de cada uno de los participantes que hacen uso del servicio.

Podemos describir las funcionalidades básicas del servidor de DirectPlay en tres fundamentales :

- Inicialización / Puesta en marcha.
- Recepción y tratamiento de mensajes de los clientes.
- Finalización.

□ Inicialización / Puesta en marcha.

Para inicializar el servidor hay que indicarle algunos datos a la librería de DirectPlay. Estos son :

- El GUID de la aplicación.
Para DirectX8 : {5726CF1F-702B-5008-28BC-EF9C95D9E288}
- El número máximo de participantes, (clientes), que permitirá el servidor.
Cualquier número, teniendo en cuenta que el propio servidor ocupa una posición como participante.
- El titulo para la sesión.
Un nombre descriptivo para la sesión, y que todos los participantes de la misma verán.

Estos datos se almacenan en la estructura de datos : `DxvBLibA.DPN_APPLICATION_DESC`, que será usado posteriormente en la inicialización del servidor.

Ejemplo de inicialización:

```
' Prepara la descripción del servidor.
Dim AppDesc As DxVLibA.DPN_APPLICATION_DESC      ' Descripción de la aplicación.
With AppDesc
    .guidApplication = AppGuidDPDX8              ' Identificador del recurso.
    .lMaxPlayers = MaxParticipantes              ' Máximo de usuarios, (+1 para el servidor).
    .SessionName = TituloSesion                  ' Título de la sesión.
    .lFlags = DPNSSESSION_CLIENT_SERVER
End With

' Prepara los objetos de DirectPlay.
Set objDX = New DirectX8                          ' Crear instancia de DX8
Set objDPSServer = objDX.DirectPlayServerCreate    ' Crear objeto de servidor.
Set objDPSServerAddress = objDX.DirectPlayAddressCreate ' Configurador de DirectPlay

objDPSServer.RegisterMessageHandler pFormularioEventos ' Vincular con el formulario que centraliza
las operaciones de servidor.

objDPSServerAddress.SetSP DP8SP_TCPIP ' Tipo de conexión : TCP/IP
objDPSServerAddress.AddComponentLong DPN_KEY_PORT, 4000 ' Vincula el puerto.

objDPSServer.Host AppDesc, objDPSServerAddress ' Arranca la sesión.
```

Básicamente lo que hacemos en la inicialización es :

- Describir el servidor mediante la estructura `DxVLibA.DPN_APPLICATION_DESC` anteriormente citada.
- Inicializar los objetos `DirectX8`, `DirectPlayServerCreate` y `DirectPlayAddressCreate`.
- Indicar el formulario que recibirá los eventos. Entre otros, los mensajes provenientes de los clientes, (participantes).
- Indicar el protocolo para la comunicación con los clientes. Entre otros podemos usar TCP/IP, IPX, modem o serie. El que vamos a utilizar va a ser **TCP/IP**.
- Indicar el puerto que debe escuchar el servidor.
- Finalmente, arrancar la sesión.

□ Recepción y tratamiento de mensajes de los clientes.

La recepción de los mensajes se basa en la implementación de las funciones de eventos de **DirectPlay8Event**.

Entre los eventos que propone `DirectPlay8Event` encontramos el evento `Receive`. Para cada mensaje que llega a través del puerto que el servidor mantiene en escucha, se activa el evento `Receive`.

Un ejemplo de recepción y tratamiento de mensajes de los clientes:

```
' Recepción de señal.
Private Sub DirectPlay8Event_Receive(dpnotify As DxVLibA.DPNMSG_RECEIVE, fRejectMsg As Boolean)
    Dim lngOffset As Long
    Dim bMessageType As Byte

    Call GetDataFromBuffer(dpnotify.ReceivedData, bMessageType, SIZE_BYTE, lngOffset)
    Select Case bMessageType
        ' Se recibe un mensaje para publicar en el chat.
        Case xSrvChat.bComChatMensaje
            xSrvChat.srvEscribirEnChat dpnotify, lngOffset, tecEnviarPublico

        ' Se recibe un susurro para transmitir al destinatario.
        Case xSrvChat.bComChatSusurro
            xSrvChat.srvEscribirEnChat dpnotify, lngOffset, tecSusurrar
```

```
' Se recibe el mandato de un supervisor de eliminar a un participante.
Case xSrvChat.bComChatMandarEliminar
    xSrvChat.srvEliminarParticipantePorOrden dpnotify, lngOffset
    ActualizaListaParticipantes

' Se recibe el mandato de un supervisor de cambiar el nombre a un participante.
Case xSrvChat.bComChatMandarCambioNombre
    xSrvChat.srvCambiarNombreParticipantePorOrden dpnotify, lngOffset
    ActualizaListaParticipantes
End Select
End Sub
```

Mas adelante veremos como vincular los eventos del servidor con el formulario de la aplicación y los objetos de DirectPlay.

Finalización.

Al finalizar el servidor se deshabilitan las Ids dadas a cada cliente. Si quedan clientes abiertos, estos no advierten que el servidor se cerró hasta que no intenten enviar un mensaje. Al hacer esto recibirán un error.

Ejemplo de cierre del servidor :

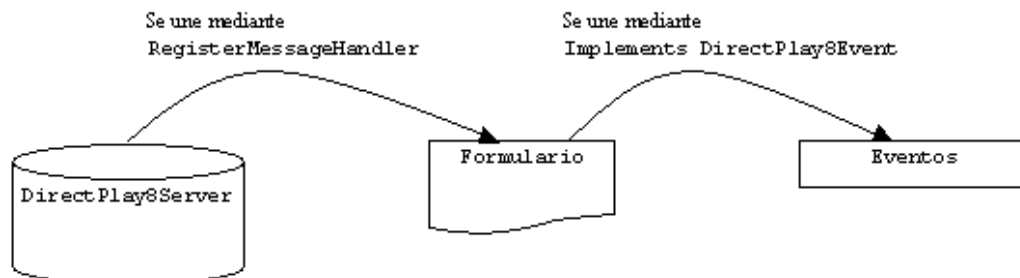
```
' Cierra los objetos de DirectPlay.
On Error Resume Next
    objDPServer.CancelAsyncOperation 0, DPNCANCEL_ALL_OPERATIONS
    objDPServer.UnRegisterMessageHandler
    objDPServer.Close
    Set objDPServer = Nothing
    Set objDPServerAddress = Nothing
    Set objDX = Nothing
On Error GoTo 0
```

Vinculación del servidor con el formulario de la aplicación. Tratamiento de los eventos del servidor.

En la implementación del servidor tenemos 3 puntos importantes que hemos de ensamblar.

- Un objeto DirectPlay8Server
- Un formulario para presentar la información.
- Los eventos que se generan en el uso del servidor y la comunicación con los clientes.

Gráficamente sería :



Ejemplo de vinculación de los elementos del servidor:

En la inicialización del servidor :

`objDPServer.RegisterMessageHandler pFormularioEventos` ' Vincular con el formulario que centraliza las operaciones de servidor.

Donde `pFormularioEventos` es el formulario principal del programa, (en nuestro caso de ejemplo, porque en realidad puede ser cualquier formulario).

En el formulario de la aplicación:

En primer lugar se define la biblioteca de eventos de DirectPlay en el formulario que tratará dichos eventos:

```
' Implementación de los eventos del control de DirectPlay.  
Implements DirectPlay8Event
```

Seguidamente, se definen las funciones de eventos de dicha biblioteca. Se han de definir todas las funciones, aunque no se utilicen todas.

```
' Quitar participante de un grupo.  
Private Sub DirectPlay8Event_AddRemovePlayerGroup(ByVal lMsgID As Long, ByVal lPlayerID As Long,  
ByVal lGroupID As Long, fRejectMsg As Boolean)  
  
' Desconexión del servidor.  
Private Sub DirectPlay8Event_AppDesc(fRejectMsg As Boolean)  
  
' Operación asíncrona completada.  
Private Sub DirectPlay8Event_AsyncOpComplete(dpnotify As DxVBLibA.DPNMSG_ASYNC_OP_COMPLETE,  
fRejectMsg As Boolean)  
  
' Conexión completa.  
Private Sub DirectPlay8Event_ConnectComplete(dpnotify As DxVBLibA.DPNMSG_CONNECT_COMPLETE,  
fRejectMsg As Boolean)  
  
' Crear nuevo grupo.  
Private Sub DirectPlay8Event_CreateGroup(ByVal lGroupID As Long, ByVal lOwnerID As Long, fRejectMsg  
As Boolean)  
  
' Destruir grupo.  
Private Sub DirectPlay8Event_DestroyGroup(ByVal lGroupID As Long, ByVal lReason As Long, fRejectMsg  
As Boolean)  
  
' Destruir participante.  
Private Sub DirectPlay8Event_DestroyPlayer(ByVal lPlayerID As Long, ByVal lReason As Long,  
fRejectMsg As Boolean)  
  
' Lista de hosts.  
Private Sub DirectPlay8Event_EnumHostsQuery(dpnotify As DxVBLibA.DPNMSG_ENUM_HOSTS_QUERY, fRejectMsg  
As Boolean)  
  
' Lista de hosts.  
Private Sub DirectPlay8Event_EnumHostsResponse(dpnotify As DxVBLibA.DPNMSG_ENUM_HOSTS_RESPONSE,  
fRejectMsg As Boolean)  
  
' Ha habido un cambio en la propiedad Hosts del servidor.  
Private Sub DirectPlay8Event_HostMigrate(ByVal lNewHostID As Long, fRejectMsg As Boolean)  
  
' La conexión de un participante se ha hecho efectiva.  
Private Sub DirectPlay8Event_IndicateConnect(dpnotify As DxVBLibA.DPNMSG_INDICATE_CONNECT,  
fRejectMsg As Boolean)  
  
' La conexión de un participante ha fallado.  
Private Sub DirectPlay8Event_IndicatedConnectAborted(fRejectMsg As Boolean)
```

```
' Al cambiar la información sobre un cliente o grupo.
Private Sub DirectPlay8Event_InfoNotify(ByVal lMsgID As Long, ByVal lNotifyID As Long, fRejectMsg As Boolean)

' Recepción de señal.
Private Sub DirectPlay8Event_Receive(dpnotify As DxVBLibA.DPNMSG_RECEIVE, fRejectMsg As Boolean)

' Envío de mensaje.
Private Sub DirectPlay8Event_SendComplete(dpnotify As DxVBLibA.DPNMSG_SEND_COMPLETE, fRejectMsg As Boolean)

' Se cierra la sesión.
Private Sub DirectPlay8Event_TerminateSession(dpnotify As DxVBLibA.DPNMSG_TERMINATE_SESSION, fRejectMsg As Boolean)

' Crear nuevo participante.
Private Sub DirectPlay8Event_CreatePlayer(ByVal lPlayerID As Long, fRejectMsg As Boolean)
```

Implementar el cliente.

Un cliente es un programa que hace uso de las librerías de DirectPlay de cliente. Su misión es la de enviar mensajes al servidor. Esos mensajes son ordenes que indican al servidor lo que debe hacer. A cada orden recibida por el servidor, este reacciona, hace lo que deba hacer y, si es necesario, envía un mensaje de confirmación al cliente con el resultado de la operación.

Las principales funcionalidades de un cliente pueden relatarse en 4 fases :

- Inicializar el cliente.
- Conexión con el servidor.
- Envío y recepción de mensajes.
- Desconexión.

¶ Inicializar el cliente.

Prepara los objetos de DirectPlay.

Ejemplo de inicialización:

```
' Inicializa objetos DX8
Set ObjDX8 = New DirectX8
Set ObjDPClient = ObjDX8.DirectPlayClientCreate ' Cliente de DirectPlay
Set ObjDPClientAddress = ObjDX8.DirectPlayAddressCreate ' Configurator de DirectPlay
Set objDPServerAddress = ObjDX8.DirectPlayAddressCreate ' Vinculo con el servidor
ObjDPClient.RegisterMessageHandler cfgFormularioEventos ' Vincula con el formulario principal.
ObjDPClientAddress.SetSP DP8SP_TCPIP ' Conexión del cliente por TCP/IP
objDPServerAddress.SetSP DP8SP_TCPIP ' Conexión del servidor por TCP/IP
objDPServerAddress.AddComponentLong DPN_KEY_PORT, 4000 ' Puerto de comunicaciones.
```

Entre otras cosas, aquí se prepara un enlace entre los objetos de cliente y el servidor, (que obviamente no tiene porqué estar ejecutándose en la misma máquina). Se declaran aquí el protocolo de comunicaciones, (TCP/IP) y el puerto por el que recibe los mensajes el servidor, (4000).

¶ Conexión con el servidor.

El cliente se conecta con el servidor del servicio. Para ello es necesario, entre otras cosas, identificar al nuevo participante que va a unirse a la sesión abierta por el servidor. Para ello usamos una estructura de datos concreta, (DPN_PLAYER_INFO). En el que daremos un nombre descriptivo al nuevo

cliente. Seguidamente adjuntamos dicha información a un nuevo objeto gestor de cliente, (DirectPlayClientCreate). Y por último activamos la conexión, (Connect).

Ejemplo de conexión:

```
' Conecta con el servidor.
Dim AppDesc As DPN_APPLICATION_DESC ' Descripción de la aplicación.

objDPServerAddress.AddComponentString DPN_KEY_HOSTNAME, cfgIP
AppDesc.guidApplication = AppGuidDPDX8 ' Se identifica el recurso.

' Se pide el nombre del nuevo participante.
Dim PlayerInfo As DPN_PLAYER_INFO
PlayerInfo.Name = mvarcfgNombreParticipante
PlayerInfo.lInfoFlags = DPNINFO_NAME

' Se vincula al objeto DirectPlay cliente, la información sobre el nuevo participante.
ObjDPClient.SetClientInfo PlayerInfo, DPNOP_SYNC

' Se intenta la conexión con el servidor dado.
ObjDPClient.Connect AppDesc, objDPServerAddress, ObjDPClientAddress, DPNCONNECT_SYNC,
cfgNombreParticipante, Len(cfgNombreParticipante)
```

¶ Envío y recepción de mensajes.

Procedimientos por los que un participante envía mensajes al resto de usuarios, y recibe mensajes del resto de usuarios.

Ejemplo de envío de mensajes:

```
' Crea un nuevo paquete.
lngOffset = NewBuffer(bByteBuffer)

' Añade la cabecera del paquete.
Call AddDataToBuffer(bByteBuffer, bComChatMensaje, SIZE_BYTE, lngOffset)

' Añade el identificador del participante..
Call AddDataToBuffer(bByteBuffer, cfgIDParticipante, SIZE_LONG, lngOffset)

' Añade el mensaje al paquete.
Call AddStringToBuffer(bByteBuffer, stMensaje, lngOffset)

' Envía el paquete.
ObjDPClient.Send bByteBuffer, 0, DPNSSEND_GUARANTEED Or DPNSSEND_NOLOOPBACK
```

Ejemplo de recepción de mensajes:

```
' Recibir el mensaje de chat.
Call GetDataFromBuffer(dpnotify.ReceivedData, IDParticipante, SIZE_LONG, lngOffset)
stMensaje = GetStringFromBuffer(dpnotify.ReceivedData, lngOffset)
```

¶ Desconexión.

Mediante esta función, el cliente notifica al servidor que abandona la sesión. Seguidamente cierra y destruye los objetos de gestión de dicho cliente.

Ejemplo de desconexión:

```
' Cancela todas las operaciones del cliente.
ObjDPClient.CancelAsyncOperation 0, DPNCANCEL_ALL_OPERATIONS

' Quitar al participante de la lista de participantes.
ObjDPClient.UnRegisterMessageHandler

' Cerrar la conexión.
DoEvents
ObjDPClient.Close

' Cerrar los objetos.
Set ObjDPClient = Nothing
Set ObjDPClientAddress = Nothing
Set ObjDX8 = Nothing
```

Como instalar/Usar los ejemplos.

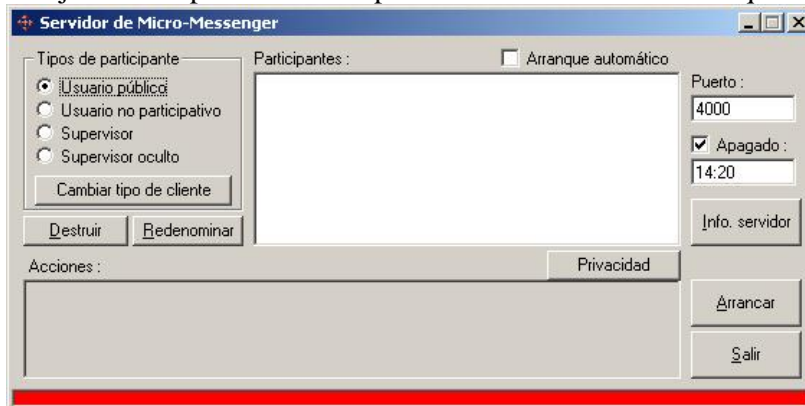
Para poder ver en marcha todo lo explicado anteriormente vamos a usar los ejemplos que se adjuntan con este artículo. Vamos a suponer que tenemos una sola máquina, y arrancaremos todos los ejemplos en la misma máquina, usando para la comunicación la IP local por defecto, (127.0.0.1).

De todas formas tener en cuenta que, si el servidor se arranca desde una máquina con IP conocida y accesible desde el exterior; para los clientes basta con cambiar la IP local por esa IP del servidor. De esta forma desde cualquier máquina, arrancando el cliente, podemos acceder al servicio.

¶ Arranque del servidor.

Para arrancar el servidor, abrimos una nueva instancia de Visual Basic 6. Seguidamente cargamos el proyecto `pDX8_Chat.vbp`.

Lo ejecutamos pulsando F5. Aparecerá una ventana como la que sigue:



Esta es la ventana del servidor del servicio de mensajería que hemos creado. Y, entre otras opciones, contiene los pasos de inicialización, recepción de mensajes y finalización explicados en el punto [Implementar el servidor](#).

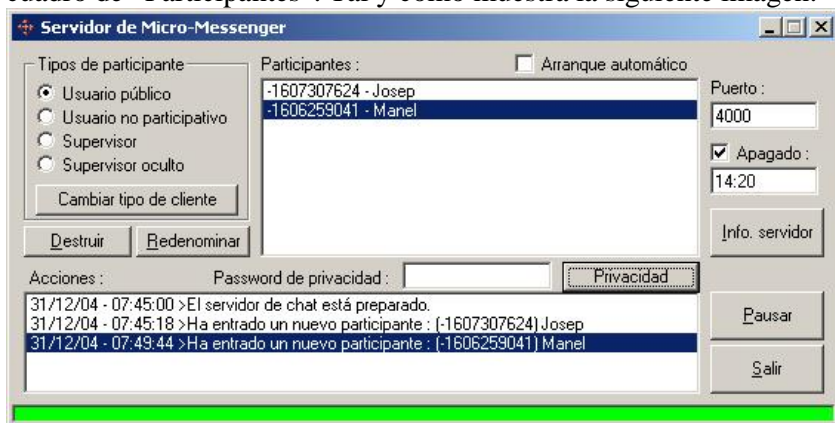
Por defecto, al arrancar el programa el servicio se encuentra desactivado. Esto lo indica la barra de color que hay en la parte inferior de la ventana. Si la barra es roja indica que el servicio está

actualmente parado. Si es verde indica que el servicio está en marcha. Para activarlo o pararlo hay que pulsar el botón “Arrancar/Parar”.

Otro dato importante es el puerto de escucha. Por defecto nos ofrece el 4000. Ese valor puede cambiarse para escuchar otro puerto. Los clientes deben “emitir” a través de ese puerto. Eso lo veremos mas adelante.

¿Como funciona?

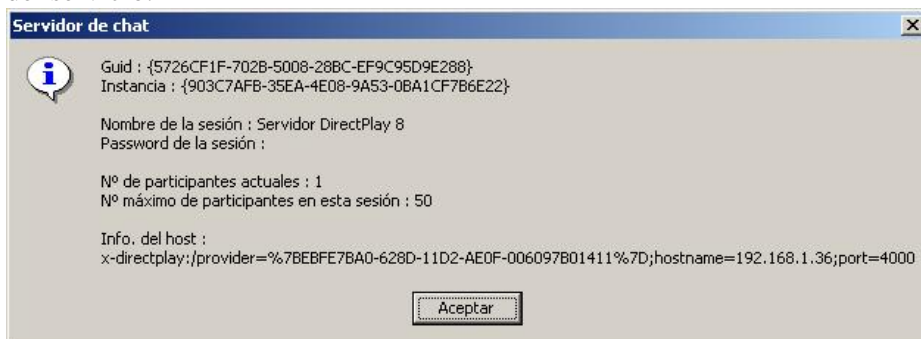
Para cada cliente que acceda al servicio proporcionado por este servidor aparecerá una entrada en el cuadro de “Participantes”. Tal y como muestra la siguiente imagen:



Haciendo clic sobre cualquiera de los participantes de la lista podemos cambiar su estatus, (cuadro “tipos de participante”), o redenominarlo, o destruirlo, (echarlo del servicio).

Pulsando sobre “Privacidad” podemos ver los movimientos que cada participante y el propio servidor han generado a lo largo de su ejecución; a modo de LOG. Al pulsar sobre “Privacidad” el programa nos pide contraseña. Inicialmente esa contraseña está en blanco. Para asignarle una hay que introducir un texto en el cuadro “Password de privacidad”.

Pulsando sobre “Info.Servidor” aparecerá una ventana con la información de estado y configuración del servicio:



Arranque del cliente MicroXat.

Para arrancar este cliente abrimos una nueva instancia de Visual Basic 6. Seguidamente cargamos el proyecto `pCliente_Chat_DX8.vbp`

Al arrancar el programa, (F5), aparece una ventana en la que nos pide el inicio de sesión del servicio de chat :

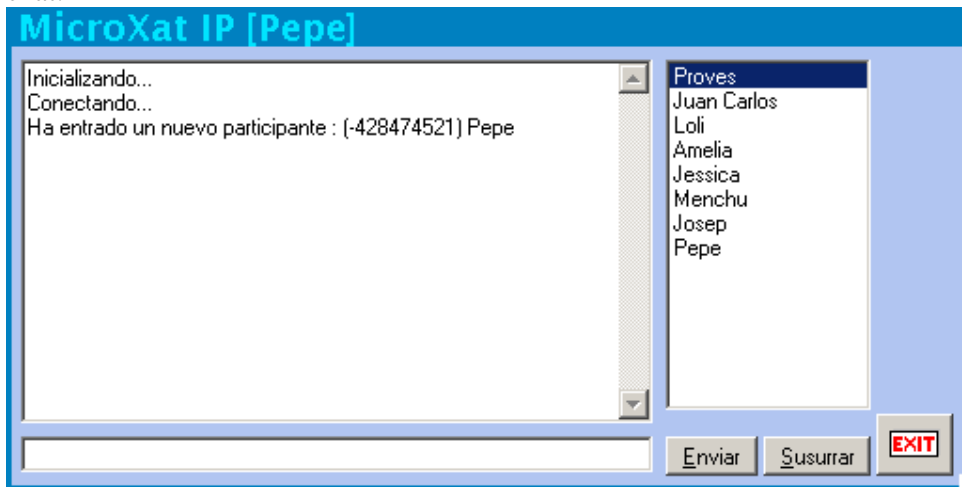


Seleccionamos la IP local, (127.0.0.1), y el puerto 4000 para establecer el contacto con el servidor de chat. También introducimos un nombre que nos identifique en el cuadro “Nombre”.



Acto seguido pulsamos sobre el botón de iniciar sesión de chat:

Si todo va bien, y el servidor del servicio está arrancado, (ver punto anterior), se abrirá la sesión de chat:



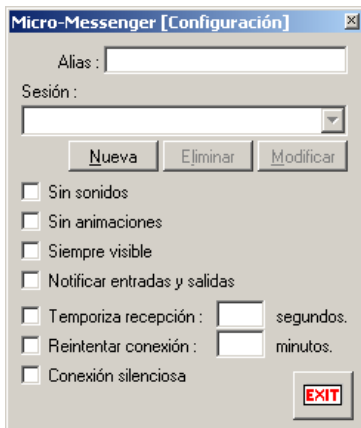
El uso de esta ventana es el típico de cualquier servidor de chat. Introducimos los mensajes y pulsamos sobre “Enviar”.

Si queremos enviar un mensaje privado a un usuario de la lista; lo seleccionamos y pulsamos sobre “Susurrar”.

Arranque del cliente FM Messenger.

Para arrancar este cliente abrimos una nueva instancia de Visual Basic 6. Y cargamos el proyecto pFMessenger.vbp.

Al arrancar el programa, (F5), nos dirá que, al ser la primera vez que se ejecuta, hemos de configurar la sesión. Se trata de decirle al sistema donde se encuentra el servidor y como queremos identificarnos ante el. Para ello aparece una ventana como la siguiente :



Lo mas importante aquí es:

- Darnos un nombre, (alias).
- Crear una nueva sesión, (pulsando sobre nueva).

Así pues, damos un nombre en el cuadro "Alias" i, seguidamente, pulsamos sobre "Nueva". Aparece entonces la siguiente ventana:



Aquí hemos de introducir:

- Un nombre descriptivo para esta sesión. Por ejemplo "local"
- Una IP. Por ejemplo : 127.0.0.1
- Y un puerto. Ha de ser el mismo que el que "escucha" el servidor. Que por defecto es el 4000

Una vez entrados los valores, la ventana queda así:



Respecto al resto de variables de la ventana de configuración :

- **Sin sonidos:** Desactiva los sonidos.
- **Sin animaciones:** Desactiva las animaciones.
- **Siempre visible:** Los mensajes que reciba este cliente se mostraran siempre por encima del resto de ventanas abiertas.
- **Notificar entradas y salidas:** A cada entrada de nuevo participante o abandono de la sesión de alguno, aparecerá una ventana de aviso.
- **Temporiza recepción:** Cuando alguien envía un mensaje a este cliente. El mensaje se cierra automáticamente en el intervalo de segundos indicado aquí.
- **Reintentar conexión:** Si no puede conectarse lo reintentará en el intervalo de minutos marcado aquí.
- **Conexión silenciosa:** No muestra errores si no puede realizar la conexión con el servidor, o si la conexión falla.

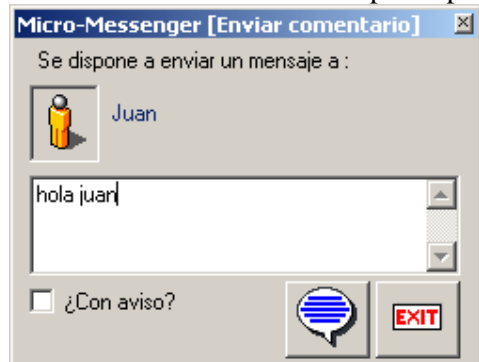
Finalmente aceptamos la configuración y, si los datos son correctos, y el servidor está en marcha, se abrirá una sesión para este nuevo cliente.

Si todo ha ido bien aparecerá un nuevo icono en la barra de iconos residentes de la barra de Windows, (al lado del reloj), con forma de “muñequito”. Pulsando sobre el aparecerá la ventana de control de ese cliente al “estilo” Messenger:



En ella veremos a todos los participantes que actualmente han abierto una sesión en el servidor. La lista es la misma que la que aparece en el cuadro de “Participantes” del servidor.

Haciendo doble-clic sobre un participante le enviaremos un mensaje:

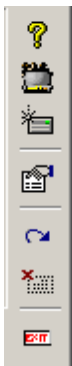


Cuando nos envíen un mensaje aparecerá una ventana en la parte superior de la pantalla con el contenido de dicho mensaje:



Si pulsamos **Alt+F2** nuestro perfil cambia, y nos convertimos en *superusuarios*. Aparecen nuevas opciones que nos permiten destruir o cambiar las propiedades del resto de participantes.

Descripción de los controles del cliente:



- **Acerca de:** Muestra una pantalla de presentación.
- **Cambiar a modo Chat / Messenger:** Esta opción presenta o bien el cuadro de usuarios a los que enviar mensajes, (por defecto), o un cuadro estilo chat donde poder enviar mensajes a todos los usuarios.
- **Ocultar la ventana de control:** Oculta esta ventana y muestra el icono de programas residentes, en la barra de tareas, al lado del reloj.
- **Propiedades:** Muestra las propiedades de configuración explicadas anteriormente.
- **Reconectar:** Cierra la sesión y vuelve a abrirla inmediatamente después.
- **Cerrar sesión:** Cierra la sesión.
- **Salir:** Cierra el programa.

Descripción de los controles de superusuario:



- **Información sobre el participante / Redenominar participante:** Muestra la información de conexión sobre un participante previamente seleccionado, (pulsando sobre él con el ratón). Seguidamente ofrece la posibilidad de cambiar el nombre público, (el que se muestra en todos los clientes), de dicho participante.
- **Destruir participante:** Destruye, (cierra la sesión), del participante seleccionado.
- **Enmudecer participante:** Cambia el estatus del participante seleccionado, prohibiéndole enviar mensajes.
- **Convertirse en superusuario oculto:** Se convierte en un usuario con características de superusuario, pero con el añadido de que no aparece en la lista de participantes activos de ningún cliente. Ese usuario únicamente aparecerá en la lista de participantes activos del servidor.

Bibliografía.

Para la creación de este artículo y los ejemplos de código me he basado en las siguientes fuentes de información:

- (Almar Joling, www.quadrantwars.com). DirectXPlay: Sending messages.
- (Almar Joling, www.quadrantwars.com). DirectXPlay: Client-Server.
- FAQs de Digitlife, (<http://www.digit-life.com/articles/dx8faq>)
- DirectX4VB, (<http://216.5.163.53/DirectX4Vb/index.asp>).
- Documentación de DirectX oficial,
(<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndrive/html/directx112000.asp>).