DirectPlay

Vishnu Kotrajaras, PhD

What is DirectPlay?

- It is a multiplayer component of DirectX
- It abstracts communication methods away
  - Same function call for communication by TCP/IP, serial cable and modem to modem
  - The communication methods are called “service providers”
  - Still, there are different elements that need to be checked, such as TCP/IP can guarantee packets, but other protocols may not be able to

Finding Service Providers

- We must enumerate them to find which service providers are available
- Enumerate = cycle through it and create a list
  - Then we can select what to use from this list
- An alternative is to
  - Attempt to use a certain service provider that you want, and see if the call returns with no error

Game Sessions

- It is an instance of a multiplayer game running simultaneously on two or more computers
- People on the same sessions can see one another (theoretically, because the game area may be very large)

Peer-to-Peer Game Sessions

- Players communicate directly with all other players
- Traffic increases dramatically with each additional player

Peer Communication Object

- To start peer-to-peer, first you must create a CLSID_DirectPlay8Peer object

```
IDirectPlay8Peer        *g_pDP;
CoCreateInstance(CLSID_DirectPlay8Peer ,
        NULL,
        CLSCTX_INPROC_SERVER,
        IID_ IDirectPlay8Peer,
        (LPVOID*) & g_pDP);
```

Initialize the object

Peer Hosting

- There is a host (server) that is used for accepting new players
- Host migration
  - If the host quits, another player is automatically selected as a host
- To connect to a host
  - Specify the host you want to connect
  - If you are using TCP/IP, you also need IP address and port number

Peer-To-Peer Host duty

- Keeping track of who is in the game
- Informing everyone when a new player joins the game
- Informing everyone when a player leaves the game
- Accepting or denying a new player to the game

Client/Server Sessions

- Clients only talk to server
- The server manages everything

Server Communication Object

- Must create CLSID_DirectPlay8Server object
- The interface that accompanies it is IID_IDirectPlay8Server

Server Hosting

- Usually, server is a stand-alone console application
- No graphics
  - Allows a player to host a game and play it on the same machine (without too much resource wasted in graphics)
- Server needs to set up port for players to connect
  - If you are behind a firewall, you need to open up some ports
  - Once you decide which port to use, call the host function of DirectPlay

DirectPlay Addresses

- To establish a connection, you must have an address to connect
  - Socket: URL + port number
  - DirectPlay: special address that holds info
    - Ip adress, baud rate etc

The Address Object

- To create an address, you must use IDirectPlay8Address interface. You must create CLSID_DirectPLayAddress com object

```
hReturn = CoCreateInstance(CLSID_DirectPLay8Address,
                    NULL, CLSCTX_INPROC_SERVER,
                    IID_IDirectPlay8Address,
                    (LPVOID*) &g_pHostAddress);
if (FAILED(hReturn)){
    MessageBox(hWindow, “Fail to Create Host Address()”,
          “Invalid Param”, MB_ICONERROR);
    return –1;
}
```

The Address Object (cont.)

- Two types of address objects
  - One for the computer you are running the application = device address
  - One for the computer you are connecting to = host address
- Creating an address is not enough, we must set it up for the data it needs to hold, that is, setting its service provider

Setting Address Object's Service Provider

```
//set the host address to TCP/IP
if (FAILED(hReturn = g_pHostAddress ->
    SetSP(&CLSID_DB8SP_TCPIP))){
    MessageBox(hWindow, "Fail to SetSP for Host Address",
        "Invalid Param", MB_ICONERROR);
    return –1;
}
```

CLSID_DB8SP_IPX
CLSID_DB8SP_MODEM Modem to modem
CLSID_DB8SP_SERIAL

Contents of DirectPlay Address

- Is a string that contains value pairs
- First part contains
  - x-directplay:/ // this is the address, before setting //Service provider
- Once SetSP() is called, it fills in the string with the provider's GUID (unique identifier)

Service provider's GUID

x-directplay:/provider=%7BEBFE7BA0-628D-11D2-AE0F-006097B01411%7D;
device=%7BIP ADAPTER GUID%7D;

GUID of the device on your computer the connection will use

Address Strings

- Provider
- Device

We saw in the previous page

- hostName //used mainly for TCP/IP
- Port // port on the host, used in TCP/IP or IPX
- ApplicationInstance //GUID of the game you //want to connect
- Baud // baud rate for modem or serial connection
- StopBits //the stop bits for modem or serial connection

Address Strings (cont.)

- Parity // the parity of modem or serial connection
- PhoneNumber //the phone number to call for //modem connection
- FlowControl // the flow control for modem or //serial connection
- Program // an optional program GUID

To use TCP/IP to connect to IP 192.168.0.2 on port 6000

x-directplay:/provider=%7BEBFE7BA0-628D-11D2-AE0F-006097B01411%7D;
device=%7BIP ADAPTER GUID%7D;
hostname= 192.168.0.2;
port=6000;

Each of these is called a "component"

You can put # at the end. After #, you can put any strings

IDirectPlay8Address Functions

- We have two ways to work with the address
  - The first is, access the address string info directly
  - The second is, use the interface's functions to modify the string
  - The interface functions are shown next page

IDirectPlay8Address Functions (cont.)

- BuildFromURLW builds a Dplay address from a wide string
- BuildFromURLA builds a Dplay address from a string
- Duplicate Duplicates the Dplay address
- SetEqual Sets the address to be equal to another address
- IsEqual checks whether two addresses are equal
- Clear clears out the address string
- GetURLW retrieves the address in wide string format
- GetURLA retrieves the address in string format
- GetSP retrieves the service provider GUID from the address
- GetUserData retrieves any user-specified data
- SetSP sets the service provider GUID in the address

IDirectPlay8Address Functions (cont2.)

- SetUserData sets user data in the address
- GetNumComponents gets the number of components in the address
- GetComponentByName gets the contents of a component by name
- GetComponentByIndex gets the contents of a component by position
- AddComponent adds a component to the address
- GetDevice gets the device in the address
- SetDevice sets the device in the address
- BuildFromDPADDRESS sets the address to be equal to the address used in Dplay 4

AddComponent()

```
HRESULT AddComponent(
    const WCHAR *const pwszName, // component name
    const void *const lpvData, //data to be used
    const DWORD dwDataSize, //size of the data to be used
    const DWORD dwDataType //type of data of which this
                           //component consists
)
```

พวกตัวแดงนี่เดี๋ยวมีอธิบายเพิ่ม

Component Name

- DPNA_KEY_APPLICATION_INSTANCE the application instance
- DPNA_KEY_BAUD modem baud rate
- DPNA_KEY_DEVICE connection device
- DPNA_KEY_FLOWCONTROL modem flow control
- DPNA_KEY_HOSTNAME IP or host name
- DPNA_KEY_PARITY modem parity
- DPNA_KEY_PHONENUMBER modem phone number
- DPNA_KEY_PORT IP port
- DPNA_KEY_PROGRAM program GUID
- DPNA_KEY_PROVIDER service provider
- DPNA_KEY_STOPBITS modem stop bits

Size of the data to be used

- 3 possibles size
  - sizeof(DWORD)
  - sizeof(GUID)
  - (wcslen(wszYourString)+1)*sizeof(WCHAR)

Length of wide character string

Account for NULL terminator

Total string size

Type of Data

- DPNA_DATATYPE_STRING
- DPNA_DATATYPE_DWORD
- DPNA_DATATYPE_GUID
- DPNA_DATATYPE_BINARY

AddComponent() Example

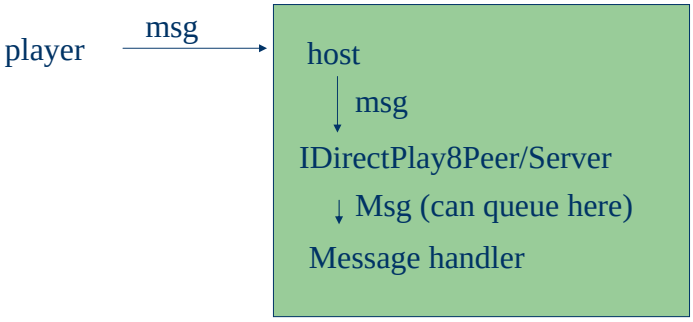
```
hReturn = g_pHostAddress ->
    AddComponent(DPNA_KEY_HOSTNAME,
    wszHostName,
    wcslen(wszHostName)+1)*sizeof(WCHAR),
    DPNA_DATATYPE_STRING);

if(hReturn != S_OK){
    MessageBox(hWnd, “Fail to AddComponent()”,
          “hrJoinGame()”, MB_ICONERROR);
    return –1;
}
```

All that is done so far is preparing the connection

DirectPlay Message Handler

- When messages are received, they are sent to message handler



Creating Message Handler

- Example: setting handler in peer-to-peer

```
if(FAILED(hReturn=g_pDP -> Initialize(
    NULL,
    DirectPlayMessageHandler,
    0)))
{
    MessageBox(hWnd, “Fail to create message handler”,
          “DirectPlay Error”, MB_ICONERROR);
    return –1;
}
```

IDirectPlay8Peer interface variable

This is the function to use

Initialize()

```
HRESULT Initialize(
    PVOID const pvUserContext, // who sent the message
    const PFNDPNMESSAGEHANDLER pfn, //name of the message
                  //handler function to receive DirectPlay messages
    const DWORD dwFlags //normally NULL
)
```

See next page

Message Handler Function

```
HRESULT WINAPI MessageHandler(
    PVOID pvUserContext, // user's context
    DWORD dwMessageId //type of message passed
    PVOID pMsgBuffer //actual message contents
)
```

The message handler must be thread safe because it can be called from multiple locations

Type of Message passed (some are only for certain interfaces)

- DPN_MSGID_ADD_PLAYER_TO_GROUP
- DPN_MSGID_ASYNC_OP_COMPLETE
- DPN_MSGID_CLIENT_INFO
- DPN_MSGID_CONNECT_COMPLETE
- DPN_MSGID_CREATE_GROUP
- DPN_MSGID_CREATE_PLAYER
- DPN_MSGID_DESTROY_GROUP
- DPN_MSGID_DESTROY_PLAYER
- DPN_MSGID_ENUM_HOSTS_QUERY
- DPN_MSGID_ ENUM_HOSTS_RESPONSE
- DPN_MSGID_GROUP_INFO

Type of Message passed (cont.)

- DPN_MSGID_HOST_MIGRATE
- DPN_MSGID_INDICATE_CONNECT
- DPN_MSGID_INDICATE_CONNECT_ABORTED
- DPN_MSGID_PEER_INFO
- DPN_MSGID_RECEIVE
- DPN_MSGID_REMOVE_PLAYER_FROM_GROUP
- DPN_MSGID_RETURN_BUFFER
- DPN_MSGID_SEND_COMPLETE
- DPN_MSGID_SERVER_INFO
- DPN_MSGID_TERMINATE_SESSION
- DPL_MSGID_CONNECT

Type of Message passed (cont2.)

- DPL_MSGID_CONNECTION_SETTINGS
- DPL_MSGID_DISCONNECT
- DPL_MSGID_RECEIVE
- DPL_MSGID_SESSION_STATUS

Example: Inside Message Handler Function

```
Switch(dwMessageId){
    case DPN_MSGID_HOST_MIGRATE:
    {       vShowText(hLB_Output, “Migrate Host”);
            PDPNMSG_HOST_MIGRATE pHostMigrateMsg;
            pHostMigrateMsg =
            (PDPNMSG_HOST_MIGRATE)pMsgBuffer;

            //check to see if we are the new host
            if(pHostMigrateMsg -> dpnidNewHost == g_dpnidLocalPlayer){
                    vShowText(h_LB_Output, “(HOSTING)”);
            }
            break;
    }
}
```

Create the right message structure pointer

Example2: Processing Chat Packet

```
    case DPN_MSGID_RECEIVE:
    {
            PDPNMSG_ RECEIVE pReceiveMsg;
            pReceiveMsg =
            (PDPNMSG_ RECEIVE)pMsgBuffer;

            vShowText(hLB_Output, (char*) pReceiveMsg  ->
    pReceiveData );
            break;
    }
```

Initializing Peer-To-Peer Session

1. Initialize COM
2. Create an IDirectPlay8Peer interface
3. Initialize the message handler
4. Create a device address
5. Set the service provider
6. Create a host address

Initializing COM

Do it once per thread

```
hReturn = CoInitialize(NULL);
if (FAILED(hReturn)){
    MessageBox(hWindow, “Error initializing COM”,
        “DirectPlay Error”, MB_ICONERROR);
    return hReturn;
}
```

When exit game, do CoUninitialize()

Create an IDirectPlay8Peer interface (only once)

```
IDirectPlay8Peer          *g_pDP;
If (FAILED(hReturn =CoCreateInstance(CLSID_DirectPlay8Peer ,
        NULL,
        CLSCTX_INPROC_SERVER,
        IID_ IDirectPlay8Peer,
        (LPVOID*) & g_pDP))){
    MessageBox(hWnd, "Can't create DPlayPeer",
        "DirectPlay Error", MB_ICONERROR);
}
```

นี่คือโค้ดที่กล่าวถึงตั้งแต่ตอนแรกที่พูดถึง peer-to-peer ไง

CoCreateInstance() is used to create a local copy of COM object. This is the only way a COM object can be used

CoCreateInstance()

Class's unique id.

```
CoCreateInstance(
    REFCLSID rclsid , //CLSID that matches the object you are trying
                //to retrieve  i.e. use the interface's
    LPUNKNOWN pUnkOuter, //is the object part of an aggregate?
    DWORD dwClsContext,//how the object's code will be processed
                //e.g. CLSCTX_INPROC_SERVER means
                //the object will run in the process that calls it
    REFIID riid, //reference to the interface of the object (always IID)
    LPVOID* & ppv //address of variable to receive interface pointer
)
```

Initializing Message Handler (yes, we already talked about it)

```
if(FAILED(hReturn=g_pDP -> Initialize(
    NULL,
    DirectPlayMessageHandler, //this is our message handler function
    0)))
{
    MessageBox(hWnd, “Fail to create message handler”,
        “DirectPlay Error”, MB_ICONERROR);
    return –1;
}
```

Yes, we must create the handler function

DirectPlayMessageHandler()

```
HRESULT WINAPI DirectPlayMessageHandler(
    PVOID pvUserContext, DWORD dwMessageId,
    PVOID pMsgBuffer)
{
    HRESULT    hReturn = S_OK;
    Switch(dwMessageId){
        ….
    }
    return hReturn;
}
```

Creating the Device Address

- It represents your machine's method for communication, we already talked about it in "Address Object"

```
hReturn = CoCreateInstance(CLSID_DirectPLay8Address,
                    NULL, CLSCTX_INPROC_SERVER,
                    IID_IDirectPlay8Address,
                    (LPVOID*) &g_pDeviceAddress);
if (FAILED(hReturn)){
    MessageBox(hWindow, "Fail to Create Device",
          "CoCreateInstance()", MB_ICONERROR);
    return –1;
}
```

Set the Service Provider (we already talked about this too)

```
if (FAILED(hReturn = g_pDeviceAddress ->
    SetSP(&CLSID_DB8SP_TCPIP))){
    MessageBox(hWindow, "Fail to SetSP() for device address",
          "Invalid Param", MB_ICONERROR);
    return –1;
}
```

Creating the Host Address and Set it's Service Provider

- Similar to what we did for device address

```
hReturn = CoCreateInstance(CLSID_DirectPLay8Address,
                           NULL, CLSCTX_INPROC_SERVER,
                           IID_IDirectPlay8Address,
                           (LPVOID*) &g_pHostAddress);
if (FAILED(hReturn)){
    MessageBox(hWindow, "Fail to Create Host Address",
          "CoCreateInstance()", MB_ICONERROR);
    return –1;
} if (FAILED(hReturn = g_pHostAddress ->
    SetSP(&CLSID_DB8SP_TCPIP))){
    MessageBox(hWindow, "Fail to SetSP() for host address",
          "Invalid Param", MB_ICONERROR);
    return –1;
}
```

Hosting a Peer-to-Peer Session

```
hReturn = g_pDP -> Host(
    &dnAppDesc, // application description
    g_pDeviceAddress, //IDirectPlay8Address of our device address
    1, //how many addresses are in the second parameter
    NULL, //for future
    NULL, //for future
    NULL, //player context info
    NULL //flag  e.g. DPNHOST_OKTOQUERYFORADDRESSING
);
```

This device address now becomes host

Pop-up window can come up to ask for additional connection info

Application Description

- Is a structure

```
typedef struct _DPN_APPLICATION_DESC{
    DWORD       dwSize; // size of DPN_APPLICATION_DESC structure
    DWORD       dwFlags;
    GUID        guidInstance; //GUID for the instance of the app.
    GUID        guidApplication;
    DWORD       dwMaxPlayers; // if 0, infinite players can join
    DWORD       dwCurrentPlayers; //only filled when we use
                                  //GetApplicationDescription()
    WCHAR*      pwszSessionName; // in unicode format
```

Application Description (CONT.)

```
    WCHAR*      pwszPassword; // in unicode
    PVOID       pvReservedData; //reserved to NULL
    DWORD       dwReservedDataSize; //reserved to NULL
    PVOID       pvApplicationReservedData;
    DWORD       dwApplicationReservedDataSize;
}DPN_APPLICATION_DESC, *PDPN_APPLICATION_DESC;
```

Before using it in hosting, you must first clear and initialize it

```
DPN_APPLICATION_DESC dnAppDesc;
ZeroMemory(&dnAppDesc, sizeof(DPN_APPLICATION_DESC));
dnAppDesc.dwSize = sizeof(DPN_APPLICATION_DESC);
```

Flags for Application Description

- DPNSESSION_CLIENT_SERVER
- DPNSESSION_MIGRATE_HOST
- DPNSESSION_NODPNSVR
- DPNSESSION_REQUIREPASSWORD
- We can mix flags with (|) operator
- For peer sessions, you can leave it to NULL unless you allow host migration
- If DPNSESSION_REQUIREPASSWORD is used, then we must set the value of pwszPassword

Setting Peer Information

- This is done before calling Host()
- Set your player info so that other players can see you

```
typedef struct _DPN_PLAYER_INFO{
    DWORD       dwSize;  //size of DPN_PLAYER_INFO structure
    DWORD       dwInfoFlags; //type of info. stored in the structure
    PWSTR       pwszName; //player name in unicode
    PVOID       pvData; //player description
    DWORD       dwDataSize; //size of player description
    DWORD       dwPlayerFlags; // DPN_PLAYER_LOCAL or
                               // DPN_PLAYER_HOST
} DPN_PLAYER_INFO, * PDPN_PLAYER_INFO;
```

DPNINFO_NAME

Setting Peer Information (cont.)

Before using it, you must first clear and initialize it

```
//create local player info
DPN_PLAYER_INFO    dpPlayerInfo;
//clear out the structure
ZeroMemory(& dpPlayerInfo, sizeof(DPN_ PLAYER_INFO));
//set the size
dpPlayerInfo.dwSize = sizeof(DPN_ PLAYER_INFO);
//this structure contains the player name
dpPlayerInfo.dwInfoFlags = DPNINFO_NAME;
//set the name
dpPlayerInfo. pwszName = wszPeerName;
```

Set Peer Info in g_pDP

- Done after the player information structure is created

```
if (FAILED(hReturn = g_pDP -> SetPeerInfo(
    &dpPlayerInfo,
    NULL,
    NULL,
    DPNSETPEERINFO_SYNC))){
    MessageBox(hWindow, “Fail to SetPeerInfo()”,
          “DirectPlay Error”, MB_ICONERROR);
    return –1;
}
```

SetPeerInfo()

```
HRESULT SetPeerInfo(
    //the player info that we set up
    const DPN_ PLAYER_INFO *const pdpnPlayerInfo,

    //user-specified context, usually NULL
    PVOID const pvAsyncContext,
    // don't use
    DPHABDLE *const phAsynHandle,

    //can only use DPNSETPEERINFO_SYNC
    const DWORD dwFlags
);
```

Hosting Review

1. Initialize and set up peer info, using DPN_ PLAYER_INFO and SetPeerInfo()
2. Set up application description with DPN_APPLICATION_DESC data structure
3. Optionally add the port number to the device address
4. Use Host()

Joining a Peer-To-Peer Session (4 steps in total)

1. Set up peer info //เหมือนเดิมเลย
2. Set up application description
   - Create the description and load it with the size and GUID for the application

```
ZeroMemory(&dpnAppDesc, sizeof(DPN_APPLICATION_DESC));
dpnAppDesc.dwSize = sizeof(DPN_APPLICATION_DESC);
dpnAppDesc.guidApplication = DP_CHAT;
```

3. Set the host name and optional port number
   - Use AddComPonent()

↓ The code for this is in the next page

Joining a Peer-To-Peer Session (cont.)

```
//add host name to address
hReturn = g_pHostAddress ->
      AddComponent(DPNA_KEY_HOSTNAME,
      wszHostName,
      wcslen(wszHostName)+1)*sizeof(WCHAR),
      DPNA_DATATYPE_STRING);
if (hReturn != S_OK){
      MessageBox(hWnd, “Fail to AddComponent()”,
          “hrJoinGame()”, MB_ICONERROR);
      return –1;
}
// then add port number to address
```

Joining a Peer-To-Peer Session (cont2.)

```
hReturn = g_pHostAddress -> AddComponent(DPNA_KEY_PORT,
        &dwPort,
        sizeof(DWORD),
        DPNA_DATATYPE_DWORD);
if (hReturn != S_OK){
        MessageBox(hWnd, “Fail to AddComponent()”,
            “hrJoinGame()”, MB_ICONERROR);
        return –1;
}
```

4. Connect to the host using the Connect() function

The Connect() function

```
HRESULT Connect(
    const DPN_ APPLICATION_DESC *const pdnAppDesc, //app.desc.
    IDirectPlay8Address *const pHostAddr, //host address
    IDirectPlay8Address *const pDeviceInfo, //device address
    const DPN_SECURITY_DESC *const pdnSecurity, //NULL
    const DPN_SECURITY_CREDENTIALS *const pdnCredentials, //NULL
    const void  *const pvUserConnectData, //app. specific data, or NULL
    const DWORD dwUserConnectDataSize, //size of the previous, 0 normally
    void *const pvPlayerContext, //user context, can be NULL
    void *const pvAsyncContext, //user-supplied context data, return when
                                //DPN_MSGID_CONNECT_COMPLETE is received
    DPNHANDLE *const phAsyncHandle, //NULL for synchronous connection
    const DWORD dwFlags //one or more flags to set
);
```

Once you call Connect()

- When the host accepts the connection, you get DPN_MSGID_CONNECT_COMPLETE message
  - Once this message is received, the connection is complete

```
hReturn = g_pDP -> Connect(
    & pdnAppDesc,
    g_pHostAddress,
    g_pDeviceAddress,
    NULL,NULL,
    NULL,0,NULL,NULL
    &g_hConnectAsyncOp,
    NULL
);
```

End