Dźwięk

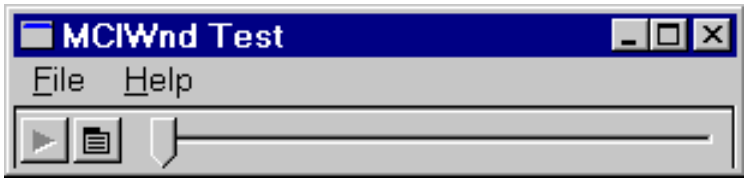
- **`MessageBeep()`**
  - zdefiniowany w systemie dźwięk określonego typu
- MIDI *(Musical Instrument Digital Interface)*
  - z wykorzystaniem MCI, *stream buffers* lub serwisów MIDI
- WAVE
  - **`PlaySound()`** - dźwięk z pliku, zasobów lub ustawień systemowych
- MCI *(Media Control Interface)*
- DirectX
  - DirectSound
  - DirectMusic

Media Control Interface (MCI)

- Zbiór uniwersalnych funkcji sterujących urządzeniami multimedialnymi (wyjście audio, MIDI, CD, video)
- Urządzenia MCI - dowolne urządzenia obsługujące zestaw poleceń MCI
- Polecenia MCI
  - systemowe:
    break, sysinfo
  - wymagane od każdego urządzenia:
    capability, close, info, open, status
  - podstawowe:
    load, pause, play, record, resume, save, seek, set, status, stop
  - rozszerzone, m.in.:
    configure, setaudio, setvideo, freeze, unfreeze

MCIWnd

- Klasa okna z wbudowaną kontrolą urządzeń multimedialnych



- Użycie:
  - tworzenie/niszczenie - **`MCIWndCreate()`**, **`MCIWndDestroy()`**
  - otwarcie/zamknięcie urządzenia lub pliku - **`MCIWndOpen()`**, **`MCIWndClose()`**
  - odtwarzanie - **`MCIWndPlay()`**, **`MCIWndPause()`**, **`MDIWndResume()`**
  - ograniczenie czasu odtwarzania - **`MCIWndPlayFrom()`**, **`MCIWndPlayTo()`**, **`MCIWndPlayFromTo()`**
  - nagrywanie - **`MCIWndNew()`**, **`MCIWndSaveDialog()`**, flaga **`MCIWNDF_RECORD`** w **`MCIWndCreate()`**

Dźwięk w .NET

- Przestrzeń nazw **`System.Media`** [2.0]
  - **`SoundPlayer`**
    - tylko pliki .wav
    - **`SoundLocation`** – ścieżka do pliku lub adres WWW
      `Stream` – strumień wejściowy
    - synchroniczne lub asynchroniczne wczytywanie i odgrywanie
    - możliwość odgrywania z automatycznym powtarzaniem
  - **`SystemSounds`**
    - statyczne metody zwracające obiekt **`SystemSound`** dla systemowych dźwięków (*Asterisk, Beep, Exclamation, Hand, Question*)
  - **`SystemSound`**
    - metoda **`Play()`**

```
SystemSounds.Asterisk.Play();
```

MP3 w .NET

- Brak

- Rozwiązaniem jest użycie funkcji MCI importowanych przy pomocy atrybutu `DllImport`
  - gotowe, darmowe klasy z Mentalis.org: **http://www.mentalis.org/soft/class.qpx?id=1**

Kontrolka Windows Media Player

- Windows Media Player ActiveX
  - HTML w przeglądarkach (IE, Firefox, Netscape Navigator)
  - Microsoft Visual C++
  - Microsoft Visual Basic 6.0
  - .NET Framework
  - Microsoft Office
- Windows Media Player Mobile ActiveX
  - HTML w Microsoft Pocket IE
  - eMbedded Visual C++
  - Active Template Library
  - możliwe także do wykorzystania z poziomu .NET Compact Framework

Składowe Windows Media Player SDK

- Windows Media Player Skins
- Windows Media Player Plug-ins
  - Windows Media Player Custom Visualizations
  - Windows Media Player User Interface Plug-ins
  - Windows Media Player DSP Plug-ins
  - Windows Media Player Conversion Plug-ins
- Windows Media Metafiles
- Windows Media Player Online Stores

DirectX

- Grupa technologii multimedialnych dla Windows
- Część składowa Windows 98/Me/2000/XP/2003/Vista
- API *(Application Programming Interface)*
  - obsługa sprzętu (karta graficzna, dźwiękowa, klawiatura, mysz, joystick)
  - wykorzystanie nowoczesnych możliwości sprzętowych
  - sprawdzenie możliwości sprzętu i dostosowanie wykorzystywanych procedur
  - wysoka wydajność
  - HAL *(Hardware Abstraction Layer)* - niezależność od urządzenia, korzystanie ze sterowników producentów
- DirectX SDK – do tworzenia programów
  DirectX Redistributable – do rozpowszechniania z programami

Wersje i składowe DirectX

- Wersje DirectX
  - 10 – dołączona do Windows Vista (wyłącznie dla tego systemu)
  - 9 – dołączona do Windows Server 2003
  - 8.1 – dołączona do Windows XP
- Składowe
  - DirectX Graphics
  - DirectInput
  - DirectSound
- Stare składowe
  - DirectDraw – obecnie składowa Direct3D
  - DirectPlay – nie używać (zalecenia: używać *Windows Sockets* i *Windows Firewall API*)
  - DirectShow – przeniesiona z DirectX SDK do Platform SDK
  - DirectMusic – nierozwijana, bez dokumentacji

Kompilacja projektów DirectX

- Pliki nagłówkowe [.h]
  - dodać **<DXSDK>\Include** do listy katalogów z plikami nagłówkowymi projektu
  - musi być pierwsza na liście
- Biblioteki [.lib]
  - dodać **<DXSDK>\Lib\[x64,x86]** do listy katalogów z bibliotekami dołączanymi do projektu
- Jeśli w projekcie ma być użyta wcześniejsza wersja DirectX, zadeklarować stałą preprocesora
  - np. **#define DIRECT3D_VERSION 0x0700**
- Wykorzystanie GUID w wersji 8.0 i późniejszych
  - dodać do projektu bibliotekę Dxguid.lib
  - usunąć **#define INITGUID** ze wszystkich plików źródłowych

DirectX Graphics - inicjalizacja

```
// utworzenie obiektu Direct3D
IDirect3D9 *pD3D = NULL;
pD3D = Direct3DCreate9(D3D_SDK_VERSION);
// pobranie trybu (rozdzielczosc, kolory)
D3DDISPLAYMODE d3ddm;
pD3D->GetAdapterDisplayMode(D3DADAPTER_DEFAULT,
                            &d3ddm);
// utworzenie urzadzenia Direct3D
IDirect3DDevice9 *pDev = NULL;
D3DPRESENT_PARAMETERS d3dpp;
ZeroMemory(&d3dpp, sizeof(d3dpp));
d3dpp.Windowed = TRUE;
d3dpp.SwapEffect = D3DSWAPEFFECT_DISCARD;
d3dpp.BackBufferFormat = d3ddm.Format;
g_pD3D->CreateDevice(D3DADAPTER_DEFAULT,
          D3DDEVTYPE_HAL, hWnd,
          D3DCREATE_SOFTWARE_VERTEXPROCESSING,
               &d3dpp, &pDev);
```

DirectX Graphics - rysowanie

```
// wyczyszczenie bufora
pDev->Clear(0, NULL, D3DCLEAR_TARGET,
    D3DCOLOR_XRGB(0,0,255), 1.0f, 0);

// rozpoczęcie rysowania sceny
pDev->BeginScene();

... // kod rysujący

// zakończenie rysowania sceny
pDev->EndScene();

// wyświetlenie narysowanej sceny
pDev->Present(NULL, NULL, NULL, NULL);
```

DirectX Graphics - zwalnianie obiektów

- Zwolnić wszystkie użyte obiekty DirectX

```
if (pDev != NULL)
    pDev->Release();
if (pD3D != NULL)
    pD3D->Release();
```

- Uwzględnić konieczność wielokrotnego tworzenia i niszczenia obiektów DirectX
  - np. zmiana rozdzielczości lub liczby kolorów

DirectX Graphics - wierzchołki

- Tworzenie

```
CUSTOMVERTEX arV[] = { ... };
IDirect3DVertexBuffer9  *pVB;
pDev->CreateVertexBuffer(..., &pVB)

VOID* pV;
pVB->Lock(0, sizeof(arV), (BYTE**)&pV, 0);
memcpy(pV, arVertices, sizeof(arVertices));
pVB->Unlock();
```

- Wyświetlanie

```
pDev->SetStreamSource(0, pVB,
    sizeof(CUSTOMVERTEX);
pDev->SetVertexShader(D3DFVF_CUSTOMVERTEX);
pDev->DrawPrimitive(D3DPT_TRANGLELIST, 0,1 );
```

DirectX Graphics - macierze

- Przekszałcenie świata *(world transformation)*

```
D3DXMATRIX matWorld;
D3DXMatrixRotationY( &matWorld,
    timeGetTime()/150.0f );
pDev->SetTransform( D3DTS_WORLD, &matWorld );
```

- Przekształcenie widoku *(view transformation)*

```
D3DXMATRIX matView;
D3DXMatrixLookAtLH( &matView, &D3DXVECTOR3(..),
    &D3DXVECTOR3(..), &D3DXVECTOR3(..));
pDev->SetTransform( D3DTS_VIEW, &matView );
```

- Przekształcenie projekcji *(projection transformation)*

```
D3DXMATRIX matProj;
D3DXMatrixPerspectiveFovLH( &matProj,
    D3DX_PI/4, 1.0f, 1.0f, 100.0f );
pDev->SetTransform(D3DTS_PROJECTION,&matProj);
```

DirectX Graphics - światła

- W wierzchołkach musi być informacja o wektorze normalnym

```
#define D3DFVF_CUSTOMVERTEX (D3DFVF_XYZ|D3DFVF_NORMAL)
```

- Materiał

```
D3DMATERIAL9 mtrl;
...
pDev->SetMaterial( &mtrl );
```

- Światło *(ambient, point light, directional light, spotlight)*

```
D3DLIGHT9 light;
...
pDev->SetLight(0, &light);
pDev->LightEnable(0, TRUE);
pDev->SetRenderState(D3DRS_LIGHTING, TRUE);
pDev->SetRenderState(D3DRS_AMBIENT,0x00202020);
```

DirectX Graphics - tekstury

- W wierzchołkach musi być informacja o współrzędnych tekstury

```
#define D3DFVF_CUSTOMVERTEX \
      (D3DFVF_XYZ|D3DFVF_DIFFUSE|D3DFVF_TEX1)
```

- Inicjowanie

```
IDirect3DTexture8 *pTexture;
D3DXCreateTextureFromFile(pDev, "texture.bmp",
    &pTexture );
```

- Rysowanie

```
pDev->SetTexture(0, pTexture);
pDev->SetTextureStageState(0, D3DTSS_..., ...);
```

DirectX Graphics - siatki *(meshes)*

- Pozwalają na wczytanie obiektów z plików **.x**
- Wczytanie

```
ID3DXMesh *pMesh;
ID3DXBuffer *pBuffer;
D3DXLoadMeshFromX("tiger.x",
      D3DXMESH_SYSTEMMEM, pDev, NULL,
      &pBuffer, &dwNumMaterials, &pMesh);
// dalej wczytanie informacji o materiałach
```

- Rysowanie

```
// dla każdego wczytanego podzbioru obiektu
pDev->SetMaterial(&g_pMeshMaterials[i]);
pDev->SetTexture(0, g_pMeshTextures[i]);
pMesh->DrawSubset(i);
```

DirectX Graphics - wybrane elementy

- *Viewport* - prostokąt, na który rzucana jest scena 3D
- Obcinanie - według współrzędnych
- Mgła
- Przezroczystość
- Z-bufor
- *Antialiasing* - rozmywanie brzegowych pikseli
- *Bump mapping* - symulacja rzeźby
- *Environment mapping* - symulacja efektu odbicia obrazu
- *Geometry blending* - gładkie połączenia segmentów
- *HLSL (high-level shader language) shaders*

DirectSound

- Cechy:
  - odgrywanie dźwięków z plików lub zasobów w formacie AVI
  - równoczesne odgrywanie wielu dźwięków
  - możliwość wykorzystania sprzętowych buforów do ogrywania dźwięków o wysokich priorytetach
  - trójwymiarowa przestrzeń dźwięków
  - specjalne efekty, np. echo, chór
    - możliwość dynamicznej zmiany efektów
  - przechwycenie dźwięków z mikrofonu lub innego wejścia

DirectInput

- API dla urządzeń sterującymi grami (mysz, klawiatura, joystick, *force-feedback*)
- Cechy:
  - obsługa urządzeń nieobsługiwanych w Win32 API
  - szybsza komunikacja bezpośrednio ze sprzętem
  - ignorowanie niektórych ustawień z panelu sterowania (pracuje bezpośrednio ze sterownikami urządzeń)
  - otrzymywanie danych od urządzenia w działającej w tle aplikacji
  - otrzymywanie danych bez wiedzy na temat generującego je urządzenia

DirectInput - użycie

1. Utworzenie obiektu *DirectInput*
2. Wyliczenie urządzeń
3. Utworzenie obiektu *DirectInputDevice* dla każdego używanego urządzenia
4. Ustawienia urządzenia
   - poziom współpracy
   - format danych
   - bufor i jego rozmiar
5. Otrzymanie informacji o gotowości odbioru danych
6. Otrzymywanie danych z urządzenia
7. Możliwość modyfikacji stanu urządzenia
8. Zwolnienie urządzeń, zamknięcie *DirectInput*

DirectSetup

- Proste API pozwalające tworzyć instalatory komponentów DirectX
- Funkcja `DirectXSetup()`
  - nadpisuje komponenty zainstalowane w systemie, ale zgodność wersji gwarantowana przez COM daje możliwość wykorzystywania nadpisanej wersji
  - wykorzystuje dane systemowe sterowników graficznych i muzycznych
  - parametry:
    - uchwyt do okna rodzica dla dialogów instalatora
    - ścieżka do plików DirectX
    - flagi określające, które komponenty zostaną zainstalowane

DirectX dla kodu zarządzanego (.NET)

- Cechy:
  - brak warstwy pośredniej z COM (Component Object Model) – większa wydajność
  - bardziej intuicyjny interfejs (wzorowany na klasach ze standardowej biblioteki .NET)
  - automatyczne zwalnianie i niszczenie obiektów

- Większość przykładów zawartych w DirectX SDK została ponownie napisana w języku C#

Przestrzeń nazw `Microsoft.DirectX`

- **`Microsoft.DirectX`**
  - pomocnicze operacje, obsługa wyjątków, macierze, kwaterniony, wektory, płaszczyzna obcinania
- **`Microsoft.DirectX.AudioVideoPlayback`**
  - ogrywanie i prosta kontrola nad plikami audio i video
- **`Microsoft.DirectX.Diagnostics`**
  - użycie *DirectX Diagnostics Tool*
- **`Microsoft.DirectX.Direct3D`**
- **`Microsoft.DirectX.DirectInput`**
- **`Microsoft.DirectX.DirectSound`**

Przestrzeń nazw `Microsoft.DirectX` c.d.

- **`Microsoft.DirectX.PrivateImplementationDetails`**
  - dostęp do niezarządzanych elementów API DirectX
- **`Microsoft.DirectX.Security`**
  - kontrola uprawnień dla obiektów Direct3D, DirectInput, DirectPlay, and DirectSound

- Wycofane:
  - **`Microsoft.DirectX.DirectDraw`**
  - **`Microsoft.DirectX.DirectPlay`**

Managed Direct3D – użycie w formularzu

```
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;

class MyDirectXForm : Form{
  Device device = null;
  private bool InitializeGraphics() { ... }
  private void Render() { ... }

  static void Main() {
    using (MyDirectXForm frm = new MyDirectXForm) {
    if (!frm.InitializeGraphics()) {
      MessageBox.Show("Could not initialize Direct3D");
      return;
    }
    frm.Show();
    while (frm.Created) {
      frm.Render();
      Application.DoEvents();
    }
  }
}
```

Managed Direct3D - inicjalizacja

```
public bool InitializeGraphics() {
  try {
    PresentParameters presentParams = new
                                PresentParameters();
    presentParams.Windowed = true;
    presentParams.SwapEffect = SwapEffect.Discard;
    // i wiele innych parametrów

    device = new Device(0, DeviceType.Hardware, this,
                  CreateFlags.SoftwareVertexProcessing,
                  presentParams);
    return true;
  }
  catch (DirectXException) {
    return false;
  }
}
```

Managed Direct 3D - zdarzenia

- Dwa rodzaje zdarzeń:
  - **`DeviceLost`** – np. po uruchomienie drugiej aplikacji
  - **`Reset`** – np. podczas zmiany trybu (okno <-> pełny ekran)
- Reakcja na zderzenie: zwykle wywołanie **`device.Reset()`**
  - najpierw zwolnić wszystkie poprzednio używane zasoby (tekstury, obiekty, bufory itp.)

```
public void OnResetDevice(object sender,EventArgs e) {
    // ...
}
public void OnLostDevice(object sender, EventArgs e ) {
    // ...
}

device.DeviceReset +=
    new System.EventHandler(this.OnResetDevice);
device.DeviceLost +=
    new System.EventHandler(this.OnLostDevice);
```

Managed Direct3D - rysowanie

```
private void Render() {
  if (device == null) {
    return;
  }

  device.Clear(ClearFlags.Target | ClearFlags.ZBuffer,
               System.Drawing.Color.Blue, 1.0f, 0);
  device.BeginScene();

  // narysowanie sceny

  device.EndScene();
  device.Present();
}

protected override void OnPaint(PaintEventArgs e) {
  Render();
}
```

Managed Direct3D – Przykłady: wierzchołki

```
VertexElements = new VertexElement[] {
  new VertexElement(0, 0,
    DeclarationType.Float3,
    DeclarationMethod.Default,
    DeclarationUsage.Position,
    0),
  new VertexElement(0, 12,
    DeclarationType.Float3,
    DeclarationMethod.Default,
    DeclarationUsage.Normal,
    0),
  new VertexElement(0, 24,
    DeclarationType.Float2,
    DeclarationMethod.Default,
    DeclarationUsage.TextureCoordinate,
    0),
  VertexElement.VertexDeclarationEnd
};
VertexDeclarator = new VertexDeclaration(device,
VertexElements);
Vertices = new VertexBuffer(device, NumberOfVertices * 32,
  Usage.WriteOnly, VertexFormats.None, Pool.Managed);
```

Managed Direct3D – Przykłady: wierzchołki c.d.

```
VertexDeclarator = new VertexDeclaration(
  device, VertexElements);
Vertices = new VertexBuffer(device,
  NumberOfVertices * 32, Usage.WriteOnly,
  VertexFormats.None, Pool.Managed);
GraphicsStream stream = null;
stream = Vertices.Lock(0, 0, LockFlags.None);
stream.Write(vertices);
Vertices.Unlock();
Indices = new IndexBuffer(device,
  NumberOfFaces * 3 * 16,
  Usage.WriteOnly, Pool.Managed, true );
Indices.SetData(faces, 0, LockFlags.None);
```

```
// rysowanie:
device.VertexDeclaration = VertexDeclaration;
device.SetStreamSource(0, Vertices, 0, 32);
device.Indices = Indices;
device.DrawIndexedPrimitives(
  PrimitiveType.TriangleList, 0, 0,
  NumberOfVertices, 0, NumberOfFaces);
```

Managed Direct3D – Przykłady: światła

```
// konieczne włączenie obsługi świateł
Device.RenderState.Lighting = true;

// opcjonalnie światło, które jest wszędzie
Device.RenderState.Ambient = System.Drawing.Color.Gray;

device.Lights[Id].Enabled = true;
device.Lights[Id].Ambient = Light.Ambient;
device.Lights[Id].Attenuation0 = Light.Attenuation0;
device.Lights[Id].Attenuation1 = Light.Attenuation1;
device.Lights[Id].Attenuation2 = Light.Attenuation2;
device.Lights[Id].Diffuse = Light.Diffuse;
device.Lights[Id].Direction = Light.Direction;
device.Lights[Id].Falloff = Light.Falloff;
device.Lights[Id].InnerConeAngle = Light.InnerConeAngle;
device.Lights[Id].OuterConeAngle = Light.OuterConeAngle;
device.Lights[Id].Position = Light.Position;
device.Lights[Id].Range = Light.Range;
device.Lights[Id].Specular = Light.Specular;
device.Lights[Id].Type = Light.Type;
```

OpenGL

- Programowalny interfejs dla urządzeń graficznych
  - renderuje dwu- i trzywymiarowe obiekty opisane zestawem wierzchołków lub pikselami
- Przenośność sprzętowa i systemowa
- Windows: 95/NT
- Składowe
  - *core OpenGL functions*
    #include <GL/gl.h>
    opengl32.lib
  - *OpenGL Utility library*
    #include <GL/glu.h>
    glu32.lib
  - *OpenGL Programming Guide auxiliary library*
    #include <GL/glaux.h>
    glaux.lib