.NET Component for Managing Web Cam Stream

Lect. Adriana REVEIU
Catedra de Informatică Economică, A.S.E. Bucureşti

A component provides a way to create and reuse graphical interfaces. The paper presents a solution for controlling an external device connected to computer, using Win32 API functions. The user control enable to preview and save multimedia stream captured from a web cam.
Keywords: *Win32 API functions, user control, unmanaged code, video stream.*

Utilizarea funcţiilor Win32 API în aplicaţii C#

.NET Framework a fost dezvoltat pentru crearea de aplicaţii pentru afaceri şi nu conţine suport pentru aplicaţii multimedia. Dezvoltarea de aplicaţii multimedia în mediul .NET se poate face fie apelând funcţii ale interfeţei de programare a aplicaţiilor Win32 API sau utilizând biblioteca DirectX. Ambele soluţii impun, în acest moment, utilizarea de cod neîntreţinut (*unmanaged code*).

Accesarea dispozitivului extern, a camerei web, prelucrarea, vizualizarea şi salvarea fluxului multimedia captat se face folosit funcţii Win32 API. Interfaţa API cuprinde funcţii ale sistemului de operare Windows folosite pentru manipularea resurselor sistemului de calcul.

Accesarea funcţiilor API se poate face doar în cod neîntreţinut (*unmanaged code*) deoarece în versiunea actuală a .NET Framework, versiunea 1.1, nu au fost dezvoltate clase în codul întreţinut (*managed code*) pentru manipularea acestui dispozitiv. Acest lucru este datorat faptului că acţiunile codului asupra ferestrelor sau asupra sistemului de operare nu pot fi determinate când se lucrează cu apeluri sau valori API.

Folosirea funcţiilor API asigură flexibilitate sportită şi viteză maximă de manipulare a dispozitivelor periferice dar are dezavantajul dificultăţii de manipulare a funcţiilor.

O soluţie alternativă la utilizarea funcţiilor API este folosirea bibliotecii DirectX. Dezavantajul acestei soluţii este dat de faptul ca versiunea actuală DirectX 9.0 are implementate în versiunea *managed code*, destul de puţine funcţionalităţi pentru manipularea fluxului multimedia preluat de la dispozitive externe limitând astfel facilităţile ce pot fi oferite aplicaţiei.

La modul general, utilizarea funcţiilor Win32 API presupune cunoaşterea unor detalii despre aceste funcţii şi anume:

- Rolul funcţiei,
- Lista argumentelor funcţiei şi valorile returnate de aceasta,
- Descrierea tuturor constantelor folosite de funcţie,
- Detalii despre structurile de date şi funcţiile necesare pentru transferul datelor,
- Valorile şi ordinea enumerărilor folosite de funcţie.

Informaţiile nu sunt necesare în cazul tuturor funcţiilor dar la fiecare apel de funcţie trebuie analizate cerinţele acesteia.

În ceea ce priveşte accesul la Win32 API, limbajul C# oferă o serie de avantaje faţă de alte limbaje .NET datorită asemănării între limbajul C# şi C.

Control de utilizator pentru captarea şi salvarea în timp real a fluxului multimedia din camera web

Pentru crearea controlului de utilizator folosit pentru captarea şi gestiunea fluxului multimedia am creat o aplicaţie Windows Control Library. Componenta va putea fi utilizată în orice aplicaţie Windows la fel ca orice altă componentă Windows.

Aplicaţia lucrează pe principiul transmiterii de mesaje sistemului de operare şi gestionarea răspunsurilor şi structurilor de date asociate acestora. Funcţiile API au fost importate din bibliotecile sistemului.

Clasele aplicaţiei control utilizator sunt:

- **APICls** – conţine metode şi structurile Win32 API necesare aplicaţiei,
- **CameraWebCls** – gestionează camera web şi fluxul multimedia în varianta cod întreţinut .NET,
- **CtrlCameraWeb** – clasa control de utilizator care asigură interfaţa cu utilizatorul.

Clasa **APICls** foloseşte metode ale bibliotecilor sistemului de operare: avicap32.dll şi user32.dll şi anume:

- metoda *capCreateCaptureWindow* din avicap32.dll pentru crearea unei ferestre de captură

```
DllImport("avicap32.dll")] public static extern IntPtr
capCreateCaptureWindowA(byte[] lpszWindowName, int dwStyle, int x,
int y, int nWidth, int nHeight, IntPtr hWndParent, int nID);
```

- metodele *SetWindowPos* din biblioteca user32.dll pentru stabilirea poziţiei ferestrei de captură

```
[DllImport("avicap32.dll")]public static extern int
capGetVideoFormat(IntPtr hWnd, IntPtr psVideoFormat, int wSize );
```

- supraîncărcări ale metodei *SendMessage* din biblioteca user32.dll pentru transmiterea de mesaje ferestrei de captură. Sunt necesare diferite supraîncărcări ale metodei *SendMessage* deoarece mesajele transmise folosesc parametrii de formate diferite.

```
[DllImport("User32.dll")] public static extern bool
SendMessage(IntPtr hWnd, int wMsg, bool wParam, int lParam);
[DllImport("User32.dll")] public static extern bool
SendMessage(IntPtr hWnd, int wMsg, short wParam, int lParam);
[DllImport("User32.dll")] public static extern bool
SendMessage(IntPtr hWnd, int wMsg, int wParam, ref BITMAPINFO
lParam);
[DllImport("User32.dll")] public static extern bool
SendMessage(IntPtr hWnd, int wMsg, int wParam, string lParam);
```

Atributul *DllImport* este folosit pentru apelul metodelor din cod unmanaged. *Extern* indică faptul că metoda este implementată în exteriorul aplicaţiei.

Constantele hexazecimale necesare pentru definirea mesajelor aplicaţiei:

```
public const int WM_USER = 0x400;
public const int WS_CHILD = 0x40000000;
public const int WS_VISIBLE = 0x10000000;
public const int SWP_NOMOVE = 0x2;
public const int SWP_NOZORDER = 0x4;
public const int WM_CAP_DRIVER_CONNECT = WM_USER + 10;
public const int WM_CAP_DRIVER_DISCONNECT = WM_USER + 11;
public const int WM_CAP_SET_CALLBACK_FRAME = WM_USER + 5;
public const int WM_CAP_SET_PREVIEW = WM_USER + 50;
public const int WM_CAP_SET_PREVIEWRATE = WM_USER + 52;
public const int WM_CAP_SET_VIDEOFORMAT = WM_USER + 45;
public const int WM_CAP_FILE_SET_CAPTURE_FILE=WM_USER +20;
public const int WM_CAP_FILE_SAVEAS = WM_USER + 23;
public const int WM_CAP_SEQUENCE= WM_USER + 62;
public const int WM_CAP_STOP=WM_USER + 68;
```

Structurile de date necesare pentru apelurile funcţiilor Windows API

Structura **VIDEOHDR** este folosită de funcţia de captură a fluxului multimedia

```
//clasa StructLayout este folosită pentru controlul formatului fizic al da-
telor structurii
//LayoutKind este constanta ce controleaza formatul unui obiect cand e ex-
portat in cod unmanaged
```

```
StructLayout(LayoutKind.Sequential)] public struct VIDEOHDR
{     [MarshalAs(UnmanagedType.I4)] public int lpData;
      [MarshalAs(UnmanagedType.I4)] public int dwBufferLength;
      [MarshalAs(UnmanagedType.I4)] public int dwBytesUsed;
      [MarshalAs(UnmanagedType.I4)] public int dwTimeCaptured;
      [MarshalAs(UnmanagedType.I4)] public int dwUser;
      [MarshalAs(UnmanagedType.I4)] public int dwFlags;
      [MarshalAs(UnmanagedType.ByValArray, SizeConst=4)] public int[]
      dwReserved;
}
```

//Structura **BITMAPINFOHEADER** conține informații despre dimensiunile şi culorile unui
// DIB (*device-independent bitmap*)

```
[StructLayout(LayoutKind.Sequential)] public struct BITMAPINFOHEADER
{     [MarshalAs(UnmanagedType.I4)] public Int32 biSize ;
      [MarshalAs(UnmanagedType.I4)] public Int32 biWidth ;
      [MarshalAs(UnmanagedType.I4)] public Int32 biHeight ;
      [MarshalAs(UnmanagedType.I2)] public short biPlanes;
      [MarshalAs(UnmanagedType.I2)] public short biBitCount ;
      [MarshalAs(UnmanagedType.I4)] public Int32 biCompression;
      [MarshalAs(UnmanagedType.I4)] public Int32 biSizeImage;
      [MarshalAs(UnmanagedType.I4)] public Int32 biXPelsPerMeter;
      [MarshalAs(UnmanagedType.I4)] public Int32 biYPelsPerMeter;
      [MarshalAs(UnmanagedType.I4)] public Int32 biClrUsed;
      [MarshalAs(UnmanagedType.I4)] public Int32 biClrImportant;
}
```

Structura **BITMAPINFO** defineşte informații despre dimensiuni şi culori pentru un DIB.

```
[StructLayout(LayoutKind.Sequential)] public struct BITMAPINFO
{     [MarshalAs(UnmanagedType.Struct, SizeConst=40)] public
      BITMAPINFOHEADER bmiHeader;
      [MarshalAs(UnmanagedType.ByValArray, SizeConst=1024)] public
      Int32[] bmiColors;
}
```

Transforma datele din blocuri de memorie unmanaged in obiecte managed se face folosind metoda **GetStructure**

```
public static object GetStructure(IntPtr ptr,ValueType structure)
{     return Marshal.PtrToStructure(ptr,structure.GetType());
}

public static object GetStructure(int ptr,ValueType structure)
{     return GetStructure(new IntPtr(ptr),structure);
}

//Returneaza marimea unei clase
public static int SizeOf(object structure)
{     return Marshal.SizeOf(structure);
}
```

Clasa **CameraWebCls**

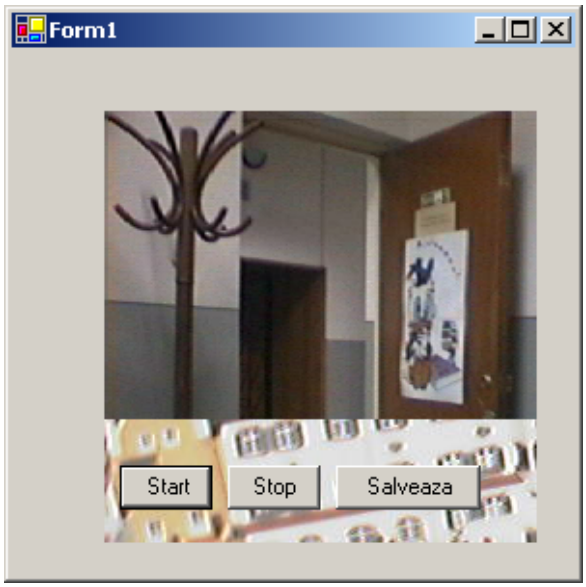
```
//Tipul de date IntPtr este folosit pentru gestiunea handlerelor şi a
// pointerilor şi este un întreg de o dimensiune specifică platformei
public CtrlCameraWebCls(IntPtr handle, int width,int height)
{     mControlPtr = handle;
      mWidth = width;
      mHeight = height;
}
```

```
private IntPtr lwndC; // pastreaza unmanaged handler al controlului
private IntPtr mControlPtr; // pastreaza managed handler  al controlului
private int mWidth;
private int mHeight;

public void OpresteCamera()
{     this.capDriverDisconnect(this.lwndC);
}

public void SalveazaFluxul()
{     capCaptureSequence(this.lwndC);
      capFileSaveAs(this.lwndC,"c:\capture.avi");
}
public void PornesteCamera()
{     byte[] lpszName = new byte[100];
      byte[] lpszVer = new byte[100];
      APICls.capGetDriverDescriptionA(0, lpszName, 100,lpszVer, 100);
      //incarca informatii despre driverul dispozitivului de captură
      // plug&play şi creaza fereastra de captura, ca fereastră copil a
      // ferestrei curente
      this.lwndC = APICls.capCreateCaptureWindowA(lpszName,
APICls.WS_VISIBLE       + APICls.WS_CHILD, 0, 0, mWidth, mHeight,
mControlPtr, 0);
      //dacă se realizează conexiunea
if (this.capDriverConnect(this.lwndC, 0))
      {this.capPreviewRate(this.lwndC, 66); //previzualizarea se face la 66
      // milisecunde
      this.capPreview(this.lwndC, true); //se declaşează previzualizarea
      APICls.BITMAPINFO bitmapinfo = new APICls.BITMAPINFO();
      //se încarcă structura de date cu informaţii despre fişierul bitmap
      // corespunzător unui cadru al secvenţei video
      bitmapinfo.bmiHeader.biSize = APICls.SizeOf(bitmapinfo.bmiHeader);
      bitmapinfo.bmiHeader.biWidth =352;
      bitmapinfo.bmiHeader.biHeight = 288;
      bitmapinfo.bmiHeader.biPlanes = 1;
      bitmapinfo.bmiHeader.biBitCount = 24;
      this.capSetVideoFormat(this.lwndC, ref bitmapinfo,
      APICls.SizeOf(bitmapinfo));
      APICls.SetWindowPos(this.lwndC, 0, 0, 0, mWidth , mHeight , 6);
      }
}
private bool capCaptureStop(IntPtr lwnd)
{     return APICls.SendMessage(lwnd,APICls.WM_CAP_STOP,0,0);
}
private bool capFileSaveAs(IntPtr lwnd, string nfis)
{     return APICls.SendMessage(lwnd,APICls.WM_CAP_FILE_SAVEAS,0,nfis);
}

private bool capCaptureSequence(IntPtr lwnd)
{     return APICls.SendMessage(lwnd,APICls.WM_CAP_SEQUENCE,0,0);
}

// functii private
// Funcţia de conectare a ferestrei de captură, creată anterior, la drive-
rul
// dispozitivului de captură
private bool capDriverConnect(IntPtr lwnd, short i)
{     return APICls.SendMessage(lwnd, APICls.WM_CAP_DRIVER_CONNECT, i, 0);
}
```

```
// deconectarea ferestrei de la dispozitivul de captură
private bool capDriverDisconnect(IntPtr lwnd)
{     return APICls.SendMessage(lwnd, APICls.WM_CAP_DRIVER_DISCONNECT, 0,
0);
}

// activarea modului Preview
private bool capPreview(IntPtr lwnd, bool f)
{     return APICls.SendMessage(lwnd, APICls.WM_CAP_SET_PREVIEW , f, 0);
}

//Stabilirea intervelului de timp la care sunt captate si afişate noi ca-
dre, în milisecunde
private bool capPreviewRate(IntPtr lwnd, short wMS)
{     return APICls.SendMessage(lwnd, APICls.WM_CAP_SET_PREVIEWRATE, wMS,
0);
}
// Procedura este apelată cand fereastra de captura previzualizeaza cadre.
private bool capSetCallbackOnFrame(IntPtr lwnd, APICls.FrameEventHandler
lpProc)
{     return APICls.SendMessage(lwnd, APICls.WM_CAP_SET_CALLBACK_FRAME, 0,
lpProc);
}

//seteaza formatul datei video captate, format dat ca parametrii ai
// structurii BITMAPINFO
private bool capSetVideoFormat(IntPtr hCapWnd, ref APICls.BITMAPINFO
BmpFormat, int CapFormatSize)
{return APICls.SendMessage(hCapWnd, APICls.WM_CAP_SET_VIDEOFORMAT,
CapFormatSize, ref BmpFormat);
}
```

Interfața controlului utilizator este o formă Windows ce conține un control **PictureBox** în care se afișează cadrele preluate din camera web. În clasa formei, **CtrlCameraWeb**, se crează o instanță a clasei **CameraWebCls:** `new CtrlCameraWebCls(pb.Handle,pb.Width,pb.Height)` şi se apelează metodele pentru

- pornirea camerei **PornesteCamera(),**
- pentru salvarea fluxului captat **SalveazaFluxul()**,
- oprirea camerei **OpresteCamera().**



Bibliografie

- Mueller J.P., *.NET Framework Solutions - In Search of the Lost Win32 API*, SYBEX Inc., 2002,
- Smeureanu I., Dârdală M., Reveiu A. – *Visual C# .NET* – CISON 2004,
- *** MSDN Library, 2004, Microsoft.