MCI in Multimedia Programming and Multimedia Authoring

Conf.dr. Marian DÂRDALĂ
Catedra de Informatică Economică, ASE Bucureşti

Media Control Interface (MCI) is a flexible interface that can be used to develop multimedia applications. In this paper are represented two approaches concerning MCI: one for multimedia programming another for multimedia authoring. The exemplifications were done in Visual C++, Toolbook Instructor and Macromedia Director.
***Keywords**: MCI, command, notify, device, audio, video, stream, capture.*

Introducere

Interfaţa MCI asigură conexiunea şi controlul perifericelor hardware sau emulate software la un calculator, în mediul Windows, prin legături specifice descrise în regiştrii sau în fişierul SYSTEM.INI, prin secţiunea [*mci*]. Iniţial ea a fost definită pentru unitatea de CD-ROM apoi a fost extinsă şi pentru alte periferice multimedia.
MCI furnizează comenzi standard pentru lucrul cu dispozitivele şi cu resursele multimedia stocate in fişiere. Aceste comenzi formează o interfaţă generică adaptată tipurilor de dispozitive multimedia. Interfaţa MCI fiind adresabilă prin comenzi, ea poate fi invocată atât din limbaje de programare (C, C++, C#, Basic, Java) cât şi din limbaje de tip script (Open Script, Lingo), specifice produselor software de creaţie multimedia (authoring).
MCI oferă aplicaţiilor capabilităţi în ceea ce priveşte controlul audio şi vizual al perifericelor. Astfel, aplicaţiile pot utiliza MCI pentru a controla orice dispozitiv multimedia recunoscut, adică: dispozitive audio de tip *wave*, secvenţe *MIDI* şi *CD audio* precum şi dispozitive *video digitale*. In cadrul MCI, driver-ele dispozitivelor sunt clasificate ca fiind simple sau compuse. Cele simple utilizează numai numele dispozitivului (*cdaudio* sau *videodisc*) în timp ce, cele compuse necesită şi numele unui fişier ce conţine resursa multimedia.

2 MCI în programarea multimedia

În cele ce urmează se vor prezenta modalităţile de utilizare a interfeţei MCI pentru dezvoltarea de aplicaţii multimedia folosind limbajele de programare, cu exemplificare în Visual C++.
Transmiterea unei comenzi MCI se face fie cu funcţia *mciSendString* fie cu funcţia *mciSendCommand*.
În cazul folosirii funcţiei *mciSendCommand*, se transmite o comandă MCI-ului prin utilizarea unui set de mesaje care implică şi folosirea unor structuri şi constante predefinite. In general acestea au prefixul *MCI_*. Structurile mai sunt denumite şi blocuri de parametri deoarece unii membrii ai structurii conţin parametri de intrare iar alţii sunt folosiţi ca parametri de ieşire.
Alternativa pe care o oferă funcţia *mciSendString* constă în faptul că se transmite o comandă MCI numai într-o formă text. Sistemul de operare converteşte o comandă tip text într-una de tip mesaj înainte să o trimită unui driver MCI pentru a o procesa. Returnarea valorilor în această modalitate de lucru se face tot într-un şir.
Indiferent de modalitatea aleasă, scenariul de lucru este dat de următorii paşi:
- deschiderea resursei multimedia sau a perifericului ca dispozitiv MCI;
- redarea sau captarea fluxului media;
- închiderea dispozitivului MCI.

- **Lucru în mod mesaj de comandă** se face prin apelul funcţiei:

```
MCIERROR mciSendCommand(MCIDEVICEID
IDDevice, UINT uMsg,
DWORD fdwCommand, DWORD_PTR
dwParam);
```

cu parametrii:
- *IDDevice* – identificatorul dispozitivului;
- *uMsg* – mesajul trimis dispozitivului (co-

manda propriu-zisă);
- *fdwCommand* – flag-uri setate prin comandă;
- *dwParam* – blocul de parametri.

Funcția returnează 0 dacă ea s-a apelat cu succes sau diferit de 0 în caz de eşec. Când valoarea returnată e diferită de 0 atunci valoarea trebuie interpretată ca un cod de eroare. Obținerea mesajului de eroare plecând de la codul ei se face prin apelul funcției:

```
BOOL mciGetErrorString(DWORD
fdwError, LPTSTR lpszErrorText,
        UINT cchErrorText);
```

unde:
- *fdwError* – codul erorii;
- *lpszErrorText* – şirul de caractere unde se va stoca mesajul de eroare (parametru de ieşire);
- *cchErrorText* – numărul maxim de caractere folosit pentru memorarea mesajului;

Pentru exemplificare se va derula o secvență video memorată în fişierul *film.avi*.
Deschiderea dispozitivului se face cu ajutorul mesajului *MCI_OPEN*. Această comandă încarcă driver-ul în memorie în caz că el nu este deja încărcat şi apoi furnizează identificatorul dispozitivului în variabila: MCIDEVICEID vid; Identificatorul obținut prin operația de deschidere se va folosi în secvențele de comenzi MCI ce vor urma; în cazul în care acest identificator este unul valid. Membrii structurii *MCI_OPEN_PARMS* se încarcă în funcție de dispozitiv şi de operația care urmează a se efectua. In secvența:

```
MCI_OPEN_PARMS mcio;
mcio.lpstrDeviceType="avivideo";
mcio.lpstrElementName="c:\\temp\\fil
m.avi";
```

s-a încărcat tipul dispozitivului (*avivideo* – video digital) şi elementul multimedia care urmează a fi vizualizat (*c:\\temp\\film.avi*). Deschiderea propriu-zisă a dispozitivului se face prin apelul:

```
int code_err;
if(code_err=mciSendCommand(0,MCI_OPE
N,
MCI_OPEN_ELEMENT|MCI_OPEN_TYPE,
(DWORD_PTR)&mcio))
```

flag-urile sugerează membrii structurii care au fost încărcați în prealabil şi care joacă rol de parametrii de intrare. In caz de eşec, se obține mesajul corespunzător codului erorii, după care acesta se afişează:

```
{
      char sir[101];
      mciGetErrorString(code_err,
sir, 100);
      AfxMessageBox(sir);
      return;
}
```

Dacă operația s-a efectuat cu succes, se preia din membrul *wDeviceID* identificatorul dispozitivului:

```
vid=mcio.wDeviceID;
```

Derularea propriu-zisă a *secvenței video* se va face folosindu-se mesajul *MCI_PLAY*. Video-ul este o dată de tip continuu şi este util de identificat, într-o aplicație, momentul terminării derulării secvenței. Acest lucru se face prin *notificare*, adică se trimite mesajul (*MM_MCINOTIFY*) în momentul terminării unei date de tip continuu (ex. animație, sunet, video).
Derularea secvenței video cu notificarea evenimentului de terminare, se face încărcând membrul *dwCallback* al structurii *MCI_PLAY_PARMS* cu handle-ul la fereastra care va primi mesajul de notificare:

```
MCI_PLAY_PARMS mcip;
mcip.dwCallback=(DWORD)m_hWnd;
```

Derularea secvenței video se face prin apelul:

```
mciSendCommand(vid,MCI_PLAY,
MCI_NOTIFY, (DWORD_PTR)&mcip);
```

flag-ul *MCI_NOTIFY* forțează trimiterea mesajului de notificare ferestrei corespunzătoare.
Derularea unei secvențe video necesită o fereastră (spațiu ecran) în care fluxul de imagini să fie vizualizat. Comanda de derulare, utilizată în această formă, determină vizualizarea sevenței într-o fereastră MCI care este dimensionată implicit la dimensiunea cadrului video iar pe *caption bar* va apare numele fişierului *avi* care se vizualizează.
Acțiunea care urmează să se desfăşoare ca urmare a notificării trebuie precizată ca răspuns la mesajul *MM_MCINOTIFY*. Pentru a răspunde la un mesaj nestandard trebuie să se supraîncarce funcția fereastră şi să se răspundă explicit la mesajul dorit. Spre exemplificare, după terminarea derulării secvenței video, se va închide dispozitivul MCI şi se va afişa mesajul GATA!!!:

```
LRESULT
```

```
CAvimciDlg::DefWindowProc(UINT
message, WPARAM wParam,
            LPARAM lParam)
{
      if(message == MM_MCINOTIFY )
      {
Inchiderea dispozitivului MCI:
            mciSendCommand(vid,
MCI_CLOSE, 0, 0);
Afişare mesaj:

      AfxMessageBox("GATA!!!");
      }
      return
CDialog::DefWindowProc(message,
wParam, lParam);
}
```

- **Lucrul în mod şir de comandă** se face prin apelul funcției:

```
MCIERROR mciSendString(LPCTSTR
lpszCommand, LPTSTR
lpszReturnString,
      UINT cchReturn, HANDLE
hwndCallback);
```

cu parametrii:

- *lpszCommand* – comanda MCI;
- *lpszReturnString* – şir pentru returnarea diferitelor rezultate;
- *cchReturn* – numărul maxim de caractere al şirului returnat;
- *hwndCallback* – handle-rul ferestrei care primeşte mesajul de notificare.

Funcția returnează 0 dacă ea s-a apelat cu succes altfel, diferit de 0, caz în care valoarea reprezintă un cod de eroare.

În vederea exemplificării acestui mod de lucru, se va capta semnal vocal din microfon şi secvența se va stoca în fişierul *nouw.wav*. Etapele care trebuie parcurse sunt:

- *Deschiderea dispozitivului* MCI pentru a crea o nouă secvență de tip *waveaudio*:

```
char sirrez[101];
mciSendString("open new type
waveaudio alias captura",
      sirrez, 100, NULL);
```

opțiunea *alias* permite ca dispozitivului sa-i fie asociat un nume (*captura*), prin care acesta să fie adresat în celelalte operații.

- *Stabilirea* unor *parametrii* care privesc *numerizarea* semnalului *audio* se face cu comanda *set*. Ea este utilă în mod deosebit atunci cînd se doreşte schimbarea valorilor implicite asociate parametrilor. De exemplu, dacă dorim să numerizăm sunetul la o rată de eşantionare de 11025 Hz, folosind un singur canal şi standardul de 8 biți per eşantion, atunci comanda MCI este:

```
mciSendString("set captura
bitspersample 8 channels 1
samplespersec 11025",

      sirrez, 100, NULL);
```

- *Inregistrarea* propriu-zisă a secvenței se realizează cu comanda *record*:

```
mciSendString("record captura",
sirrez, 100, NULL);
```

- *Oprirea înregistrării* se face cu comanda *stop*:

```
mciSendString("stop captura",
sirrez, 100, NULL);
```

- *Stocarea* pe disc, în fişierul *nouw.wav*, a secvenței înregistrate se realizează cu comanda *save*:

```
mciSendString("save captura
c:\\temp\\nouw.wav", sirrez, 100,
NULL);
```

- *Inchiderea* dispozitivului MCI deschis şi utilizat sub numele de *captura* se face folosindu-se comanda *close*:

```
mciSendString("close captura",
sirrez, 100, NULL);
```

3 MCI în creația multimedia

Controlul dispozitivelor prin MCI se poate face şi din produse software de creație multimedia (software authoring). Produse software din această categorie cum ar fi *Toolbook Instructor* sau *Macromedia Director*, prin limbajele lor de tip script (*Open Script* respectiv *Lingo*) pot controla dispozitivele multimedia prin comenzi MCI. Comenzile se transmit numai la nivel de text (şir de comandă).

In *Open Script* o comanda MCI se transmite cu ajutorul funcției *callMCI* care primeşte ca prim parametru un şir ce reprezintă comanda MCI şi un parametru opțional ce identifică obiectul care va primi mesajul de notificare (*MCINotify)*.

Pentru a exemplifica modul de lucru cu MCI din Toolbook Instructor se va derula o secvență video stocată în fişierul *mar.avi*; la terminarea ei se va notifica un obiect care va închide dispozitivul MCI şi va afişa un mesaj de terminare. Afişarea secvenței video se va face prin apăsarea unui buton aflat în pagină; secvența de cod este ataşată scriptului buto-

nului. Pentru ca secvența video să nu fie afișată în fereastra MCI implicită s-a optat pentru afișarea ei într-o fereastră utilizator. In acest scop a fost construită o fereastră de utilizator numită *fer*, prin intermediul unei clase de obiecte specializată în acest sens (*viewer*).
Scriptul răspunde la mesajul *buttonClick* care este trimis butonului la apăsarea acestuia.

```
to handle buttonClick
```

Deschidere fereastră utilizator (ea a fost creată vizual în proiect)

```
open viewer "fer"
```

Obținerea handle-ului la fereastră

```
hf = clientHandle of viewer "fer"
```

Deschiderea dispozitivului MCI identificabil prin alias-ul *film*

```
get callMCI("open c:\temp\mar.avi
alias film")
```

Asocierea dispozitivului *film* cu fereastra de utilizator care are handle-rul stocat în variabila *hf*

```
get callMCI("window film handle" &&
hf)
```

Stabilirea poziției ferestrei *fer* în raport cu fereastra principală identificată prin *viewer ID 0*

```
item 1 of position of viewer "fer" =
item 1 of position of viewer ID 0 +
100
item 2 of position of viewer "fer" =
item 2 of position of viewer ID 0 +
100
```

Afișarea ferestrei de utilizator (*fer*)

```
show viewer "fer"
```

Derularea secvenței video în fereastra *fer*; se notifică obiectul însuși identificat prin *self*

```
get callMCI("play film", self)
end
```

Tot în scriptul butonului se mai răspunde și la mesajul *MCINotify* care este trimis de comanda de derulare a secvenței video (*play*), prin notificare:

```
to handle MCINotify
```

Se închide fereastra de vizualizare (*fer*)

```
close viewer "fer"
```

Se închide dispozitivul MCI identificat prin *film*

```
get callMCI("close film")
```

Mesaj de terminare a derulării secvenței

```
request "Secventa s-a terminat"
end
```

In *Lingo*, o comanda MCI se transmite cu ajutorul comenzii *mci*. Comanda este transmisă dispozitivului MCI tot în formă text. Produsul *Director* fiind implementat atât pentru platforma *MacIntosh* cît și pentru platforma *Windows*, comanda *mci* este valabilă doar pentru implementarea *Windows*, deoarece interfața MCI este specifică acestui sistem de operare.
Pentru exemplificare, se va audia prima melodie de pe un CD Audio aflat în unitatea de CD. Declanșarea operației se face prin apăsarea unui buton existent în scenă; scriptul atașat lui este:

```
on mouseUp me
```

Se deschide dispozitivul MCI în sistem, de tip *cdaudio* folosindu-se pentru referirea lui ulterioară alias-ul *melodie*

```
  mci "open cdaudio alias melodie"
```

Se setează formatul timpului la TMSF (Tracks, Minutes, Seconds, and Frames)

```
mci "set melodie time format TMSF"
```

Derularea primei melodii de pe CD (track–ul 1) în mod sincron (aplicația așteaptă terminarea melodiei)

```
mci "play melodie from 1 to 2 wait"
```

Inchiderea dispozitivului MCI

```
mci "close melodie"
end
```

Setarea formatului timpului la TMSF, când se folosește un dispozitiv de tip *cdaudio*, este necesară pentru ca, la comanda *play*, valorile asociate opțiunilor *from*, *to* să fie interpretate ca numere de melodii de pe CD (*Tracks*).

4 Concluzii

Dezvoltarea aplicațiilor multimedia a determinat apariția multor modalități de control a fluxului media. Interfața MCI constituie o modalitate flexibilă în utilizare prin faptul că se pot controla stream-uri media din diferite medii software de dezvoltare, cât și cuprinzătoare în sensul că, prin intermediul ei se pot manipula resurse multimedia și dispozitive dintre cele mai variate.

Bibliografie

Rimmer, S., *Multimedia Programing for Windows*, McGraw-Hill, 1994
Smeureanu, I., Drulă, G., *Multimedia concepte și practică*, CISON, București 1997
*** * ***, *Macromedia Director 8.5 Shockwave Studio – Lingo Dictionary*, Macromedia Inc., 2001
*** * *,** *Microsoft Developer Network Library,* Microsoft Press, 2004
*** * ***, *ToolBook II Instructor – User Guide*, click2learn.com, Inc., 2000