Tvorba uživatelských rozhraní

Prezentace přednášek

Ústav počítačové grafiky a multimédií

Téma přednášky

Klávesnice, myš, kursor, nástěnka (clipboard)

Motto

„Jak se zpracovávají vstupy z klávesnice a myši, jak se nastavuje kursor, jak je to s nástěnkou?“

Obsah:

- Klávesnice
- Myš
- Textový kursor
- Nástěnka

Klávesnice

Vstup z klávesnice probíhá zejména prostřednictvím zpráv generovaných na základě událostí na klávesnici. Tyto zprávy jsou doručeny vždy oknu, na které “je zaostřeno”. Existují čtyři základní druhy zpráv:

- WM_CHAR – tento typ zprávy přichází po zadání znaku na klávesnici
- WM_KEYDOWN – tento typ zprávy přichází po stisku klávesy na klávesnici (nebo při “autorepeat”)
- WM_KEYUP – tento typ zprávy přichází po uvolnění klávesy na klávesnici

Pozn.: Zprávy jsou doručovány oknu, na které je “zaostřeno” (nikoli, jak tomu bývá v některých verzích nadstaveb X-Window, nad kterým je myší kursor).

Klávesnice

Typická posloupnost zpráv – pro znak „P“ a „háček“

1. WM_KEYDOWN („klávesa Shift“)
2. WM_KEYDOWN („klávesa p“)
3. WM_CHAR (znak „P“)
4. WM_KEYUP („klávesa p“)
5. WM_KEYUP („klávesa Shift“)

1. WM_KEYDOWN („klávesa Shift“)
2. WM_KEYDOWN („klávesa =“)

...

3. WM_KEYUP („klávesa =“)
4. WM_KEYUP („klávesa Shift“)

Pozn.: Protože v některých případech aplikace „čeká“, že po WM_KEYDOWN na klávese, která generuje „normálně“ znak, tento znak taky přijde, existuje zpráva WM_DEADCHAR (která by v tomto případě přišla mezi body 2. a 3. pro „háček”) upozorňující, že znak nebude generován.

Klávesnice

Parametry předávané s WM_CHAR, WM_KEYDOWN, WM_KEYUP v LParam (ve Wparam je kód znaku)

Bit	Význam
0-15	Počet opakování stisku klávesy (autorepeat)
16-23	"Scan code" klávesy. Závisí na OEM
24	"1" jedná-li se o "extended" klávesu (funkční klávesu, nebo například klávesu na číselné klávesnici)
25, 26	Nepoužito
27, 28	Pro vnitřní účely Windows
29	"1" je-li zároveň stisknuta klávesa Alt
30	Předchozí stav klávesy (autorepeat)
31	Nový stav klávesy (směr stisku)

Klávesnice

Další zprávy, které jsou pro klávesnici definovány:

- WM_SYSCHAR – tento typ zprávy přichází po zadání systémového znaku na klávesnici
- WM_SYSKEYDOWN – tento typ zprávy přichází po stisku systémové klávesy na klávesnici
- WM_SYSKEYUP – tento typ zprávy přichází po uvolnění systémové klávesy na klávesnici

Klávesnice

Funkce pro práci s klávesnicí:

- **`int GetAsyncKeyState(int VKey)`** – zjištění stavu klávesy asynchronně – není typické
- **`int GetKeyState(int Vkey)`** – jako předchozí funkce, ale synchronně s vyčítáním bufferu zpráv z Windows API funkcí **`PeekMessage`** – používá se např. při stisku myších tlačítek

```
PeekMessage(Msg,...)
switch (Msg->Msg) {
case WM_LBUTTONDOWN:
  if (GetKeyState(VK_LCONTROL)>=0)
    { ... Jen myší tlačítko ... }
  else
  { ... Ctrl a myší tlačítko ... }
```

- **`void GetKeyboardState(BYTE FAR * KeyBuffer)`** – čte do KeyBufferu stav celé klávesnice (jako bitové pole)

Myš

S myší se (podobně jako s klávesnicí) pracuje převážně pomocí zpráv generovaných na základě událostí na myši a posílaných oknu, které se nachází “pod myší”. Základními zprávami jsou:

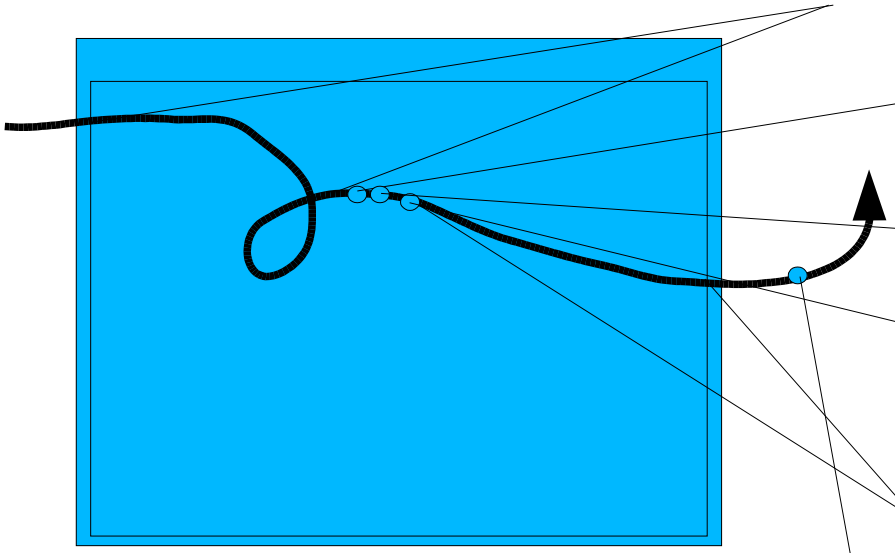
- WM_MOUSEMOVE
- WM_xBUTTONDOWN
- WM_xBUTTONUP,
- WM_xBUTTONDBLCLK

  kde, x je L, M, R

Pozn.: Pro myš platí, že zprávy jsou zasílány oknu „pod myší“ bez ohledu na „zaostření“ okna. Zprávy jsou též zasílány pouze pro klientskou oblast okna (ne pro rámeček).

Myš

Typická posloupnost zpráv o myši:



WM_MOUSEMOVE
(mnohokrát)

WM_LBUTTONDOWN
(stisk levého tlačítka)

WM_LBUTTONUP
(puštění levého tlačítka)

WM_LBUTTONDBLCLK
(další stisk levého tlačítka)

WM_MOUSEMOVE
(mnohokrát)

Puštění levého tlačítka
(WM_LBUTTONUP ale
nepřijde, resp. přijde možná
do jiného okna)

Pozn.: Posloupnost zpráv platí pro modrozelené okno (pro jiná okna bude jiná...)

Myš

Další „myší“ zprávy:

- WM_MOUSEACTIVATE
- WM_NCMOUSEMOVE
- WM_xBUTTONDBLCLK
- WM_xBUTTONDOWN
- WM_xBUTTONUP,

  kde, x je NCL, NCM nebo NCR

Pozn.: Význam „NC“ zpráv je stejný, jako pro „normální“ zprávy, ale vztahují se na „Non-Client“ oblasti okna (tedy na rámečky a nadpis).

Myš

Hlavní záludnosti práce s myší:

- Ne všechny zprávy musí dorazit – může se přeplnit fronta a nové zprávy se v tom případě zapomínají – to nevadí u WM_MOUSEMOVE, ale může to způsobit, že například zprávy WM_LBUTTONDOWN a WM_LBUTTONUP nejsou „spárovány“ - důsledek: je třeba zajistit kontrolu stavu myších tlačítek (zejména uvolnění) i mimo ...BUTTONUP (příklad viz dále)

- Při opuštění okna myš nijak okno neinformuje – nelze tedy jednoduše opuštění okna myší zjistit, resp. je k tomu třeba použít jiných postupů (viz dále)

Myš

Příklad „naivní“ obsluhy myších tlačítek (kreslení čáry):

```
int IsDown = 0;
...
switch (Msg->Msg) {
case WM_LBUTTONDOWN:
  IsDown = 1;
  break;
case WM_LBUTTONUP:
  IsDown = 0;
  break;
case WM_MOUSEMOVE:
  if (IsDown) MoveTo(Msg->LParam);
  break;
```

Myš

Příklad „bezpečné“ obsluhy myších tlačítek (kreslení čáry):

```
int IsDown = 0;
...
switch (Msg->Msg) {
case WM_LBUTTONDOWN:
  IsDown = 1;
  break;
case WM_LBUTTONUP:
  IsDown = 0;
  break;
case WM_MOUSEMOVE:
  if (!Msg->WParam&LKeyMask) IsDown = 0;
  if (IsDown) MoveTo(Msg->LParam);
  break;
```

Myš

Příklad „bezpečné“ obsluhy tlačítek – jednoduchá alternativa:

```
...
switch (Msg->Msg) {
case WM_MOUSEMOVE:
  if (Msg->WParam&LKeyMask) MoveTo(Msg->LParam);
  break;
```

Myš

„Bezpečná“ obsluha – s MouseCapture (puštění i vně okna):

```
...
switch (Msg->Msg) {
case WM_MOUSEMOVE:
   SetCapture(HWindow);
   do {
      if (PeekMessage(LoopMsg,0,0,0,PM_REMOVE)) {
         if (LoopMsg->Msg==WM_MOUSEMOVE)
           MoveTo(LoopMsg->LParam);
         TranslateMessage(LoopMsg); /* pro úplnost */
         DispatchMessage(LoopMsg);  /* pro úplnost */
         }
      } while (LoopMsg->Msg != WM_LBUTTONUP);
   ReleaseCapture();
   }
   break;
```

Myš

Detekce výjezdu myši z okna pomocí MouseCapture:

```
...
switch (Msg->Msg) {
case WM_MOUSEMOVE:
   SetCapture(HWindow);
   do {
      if (PeekMessage(LoopMsg,0,0,0,PM_REMOVE)) {
         ... /* Jakákoli obsluha myši */
         }
      } while ((LoopMsg->Msg!=WM_MOUSEMOVE)
        ||InWindow(HWindow,LoopMsg->LParam));
   /* Teď myš opustila okno, lze provést akci */
   ReleaseCapture();
   }
   break;
```

Myš

- Funkce **`SetCapture(HWND * HWindow)`** „přesměruje“ myší zprávy do zadaného okna „napořád“.
- Funkce **`ReleaseCapture(void)`** zruší přesměrování.

Pozn.: Je třeba explicitně zrušit přesměrování ihned po ukončení relevantní akce, například po vyjetí myši z okna, puštění myšího tlačítka apod. Jinak hrozí nesprávná funkce myši v celé aplikaci či systému.

Myš

Další funkce pro práci s myší:

- **`void GetCursorPos(POINT FAR * Ptr)`** – asynchronně zjistí okamžitou pozici myši
- **`void SetCursorPos(int X, int Y)`** – asynchronně nastaví polohu myšího ukazatele
- **`void ClipCursor(RECT * FAR Rect)`** – nastaví rozsah pohybu myšího ukazatele (omezení pohybu myši)
- **`void GetClipCursor(RECT * FAR Rect)`** – zjištění omezení pohybu myšího ukazatele

Pozn.: Funkce pro omezení pohybu ukazatele platí globálně a je třeba je užívat velmi opatrně.

Textový kursor

Textovým kursorem (Caret) se ve Windows rozumí kursor (blikající) v okně a NE myší ukazatel. Myší ukazatel (pointer, cursor) se vesměs nastavuje automaticky podle okna (tvar myšího ukazatele je vlastností okna, lze jej však globálně modigikovat – například na přesýpací hodiny). Pro kursor platí:

- Může být v daném okamžiku aktivní jen jeden (globálně – jen jeden a v jednom okně)
- Může být libovolného tvaru definovaného bitmapou nebo (default stav) obdélníkový zadaného rozměru
- Vždy bliká a frekvenci blikání lze nastavit
- Synchronizace blikání se změnami obsahu okna není třeba provádět na úrovní aplikace (Windows zajistí automaticky)

Funkce myšího kursoru

Pokud je třeba explicitně měnit tvar myšího ukazatele v aplikaci, lze použít následujících funkcí:

- **`HCURSOR SetCursor(HCURSOR Cursor)`** – nastaví tvar ukazatele myši – typ HCURSOR je handle datového bloku definujícího tvar ukazatele
- **`HCURSOR GetCursor()`** - získá handle na momentálně užívaný (zobrazený) tvar ukazatele
- **`HCURSOR LoadCursor(HINSTANCE Inst,LPCSTR Id)`** – získá z „resource“ záznamu tvar ukazatele
- **`void ShowCursor(BOOL Show)`** – nastaví viditelnost nebo neviditelnost myšího ukazatele

Textový kursor

Pro ovládání textového kursoru (caret) jsou k dispozici následující funkce:

- **`void CreateCaret(HWND Window,HBITMAP Bmp,int Width,int Height)`** – vytvoří ukazatel zadané velikosti, je-li zadána bitmapa, získá kursor její tvar, jinak je obdélníkový
- **`void DestroyCaret()`** - zruší kursor (caret)
- **`void SetCaretPos(int X,int Y)`**- nastaví polohu kursoru
- **`void GetCaretPos(POINT FAR * Pnt)`** – získá polohu
- **`void SetCaretBlinkTime(UINT Interval)`** – blikání (ms)
- **`UINT GetCaretBlinkTime()`** - získá blikání (ms)
- **`void HideCaret(HWND Window)`** – skryje kursor
- **`void ShowCaret(HWND Window)`** – zobrazí kursor

Pozn.: Při ztrátě „zaostření“ je třeba volat **`DestroyCaret`**

Nástěnka (clipboard)

Nástěnka (clipboard) v MS Windows je určena ke komunikaci (předávání dat) mezi programy. Komunikace je řízena uživatelem a obvyklým příkazem pro předání dat na nástěnku je "Copy", případně "Cut". Obvyklým příkazem pro získání dat z nástěnky je "Paste" nebo "Paste special".

Nástěnka by byla velmi jednoduchá, pokud by se mezi programy předávala data v jednotném formátu. Reálná situace je taková, že formátů existuje řada. Tuto situaci je třeba řešit.

Řešení uvedeného problému v MS Windows je takové, že program, který data exportuje do nástěnky, umístí na nástěnku ve všech formátech, v nichž "to umí" (nevýhodou je, že objem dat může být velký a potřebný čas také). Program importující data zvolí nejvhodnější formát z nástěnky a použije jej.

Nástěnka (clipboard)

Konvence v MS Windows je, že data jsou v nástěnce uspořádána v takovém pořadí, že formáty dat, které vykazují menší ztrátu informace předcházejí formáty, u nichž je ztráta větší. Například formátovaný text předchází text bez formátu.

Importující programy obvykle zvolí (v souladu s uvedenou konvencí) první formát, který jsou schopny importovat. Pokud tento způsob funkce není vyhovyjící, poskytují programy obvykle funkci (například "Paste special"), která uživateli umožní explicitně určit importovaný formát dat.

Existuje řada předdefinovaných formátů. Pokud předdefinovaný formát nevyhoví, lze používat uživatelsky definované formáty, které lze registrovat ve Windows.

Nástěnka (clipboard)

Výběr z předdefinovaných formátů nástěnky (clipboardu):

- CF_BITMAP Bitová mapa (HBITMAP)
- CF_DIB Bitová mapa nezávislá na zařízení (DIB)
- CF_DIF Data formátu Data Interchange Format (DIF)
- CF_METAFILEPICT Data ve formátu WMF
- CF_OWNERDISPLAY Formát zobrazovaný vlastníkem
- CF_PALETTE Barevná paleta
- CF_RIFF Data formátu RIFF (Resource Interchange)
- CF_TEXT Text, řádky končí CR, LF, ukončeno NULL
- CF_TIFF Obraz ve formátu TIFF
- CF_WAVE Zvuk ve formátu WAV, podmnožina RIFF

Nástěnka (clipboard)

Příklad zápisu do nástěnky (clipboardu):

```
...
OpenClipboard(); EmptyClipboard();
SetClipboardData(CF_TEXT,Text);
CloseClipboard();
...

...
if (OpenClipboard()) Error(); EmptyClipboard();
for (Fmt = First;Fmt<Last;Fmt = Next(Fmt))
{
   if (SetClipboardData(Fmt,D[Fmt])!=NULL) Error();
}
if (CloseClipboard()) Error();
...
```

Nástěnka (clipboard)

Příklad čtení z nástěnky (clipboardu):

```
...
if (OpenClipboard()) Error();
for (Fmt = First;Fmt<Last;Fmt = Next(Fmt)) {
  if (IsClipboardFormatAvailable[Fmt]) {
    Data = GetClipboardData(Fmt);
    }
  }
...


...
Fmt = 0; while (Fmt = EnumClipboardFormats(Fmt)) {
  if (IsKnownFormat(Fmt)) Data = GetClipboardData(Fmt);
  }
if (CloseClipboard()) Error();
...
```

Nástěnka (clipboard)

Přehled dalších funkcí pro nástěnku (clipboard):

- `BOOL OpenClipboard(HWND Window)`
- `BOOL CloseClipboard()`
- `BOOL EmptyClipboard()`
- `int CountClipboardFormats()`
- `UINT EnumClipboardFormats(UINT Format)`
- `HANDLE SetClipboardData(UINT Format,HANDLE D)`
- `HANDLE GetClipboardData(UINT Format)`
- `int GetClipboardFormatName(UINT Format,LPSTR Name,int MaxSize)`
- `BOOL IsClipboardFormatAvailable(UINT Format)`
- `UINT RegisterClipboardFormat(LPCSTR Name)`

Shrnutí

- Klávesnice a myš jako základní prostředky pro vstup údajů do počítače jsou zdrojem zpráv pro okna avšak pro jejich obsluhu existuje i řada funkcí
- Ukazatel myši se nastavuje převážně automaticky (je vlastností okna) a výjimečně jej lze explicitně nastavovat
- Textový kursor se může vyskytovat jen v jednom okně a může být obdélníkový nebo být definován bitmapou

- Do nástěnky „vysílající program“ umísťuje údaje v řadě formátů, „přijímající“ program si vybírá z dostupných formátů