1. Ce este programare controlata de evenimente?
Ce este programare pilotata de evenimente?

Programarea orientată eveniment este o paradigmă a programării calculatoarelor. Spre deosebire de programele tradiționale, care-și urmează propria execuție, schimbându-și câteodata doar cursul în puncte de ramificație (instrucțiuni test, etc), cursul execuției unui program orientat eveniment este condus în mare parte de e programe mici numite handlere de eveniment, care sunt apelate ca răspuns la evenimente externe și dintr-un coordonator. În multe cazuri, handlerele de evenimente pot declanșa ele însele evenimente, ducând la o cascadă de evenimente.

2. PPE:Scurt istoric

Înainte event-handlerele erau implementate ca subrutine în cadrul programului procedural. Fluxul de execuție a programului era determinat de către programator și controlat de firul principal al aplicației. Programul trebuia foarte bine structurat. Într-un program controlat de evenimente nu există un flux de control. Rutina principală conține o buclă de dispecerizare a evenimentelor (dispecer de evenimente), care atunci când apare un eveniment apelează procedura adecvată de tratare a acestuia. Deoarece codul pentru bucla de dispecerizare a evenimentelor este furnizat de obicei de mediul sau cadrul de dezvoltare bazat pe evenimente și în mare parte este invizibil pentru programator, percepția programatorului asupra aplicației este cea a unei colecții de rutine de tratare a evenimentelor.

3. Handlerul de eveniment și coordonatorul (dispatcher)

Evenimentele sunt gestionate de operatorul central evenimente(dispatcher) – ciclu care ruleaza in fundal si asteapta sa se produca un eveniment. Cand este un eveniment produs, dispecerul trebuie sa determine tipul evenimentului si sa apeleze handlerul corespunzator. Handlerul de evenimente este un mic bloc de cod procedural care trateaza un eveniment. Acesta produce un raspuns visual pentru a informa sau directiona utilizatorul. Un handler de evenimente poate declansa un alt eveniment care va solicita un alt handler.

4. PPE: domenii de aplicare

4.1 in cazul cand sunt facute interfete utilizator(inclusive cele grafice pentru aplicatii desktop)
4.2 la crearea aplicatiilor server in cazul in care generarea proceselor de serviciu este exclusa.
4.3 la crearea aplicatiilor care constituie un joc si trebuiesc controlate o multime de evenimente.

5. PPE pentru aplicatii server

PPE este utilizată pentru a rezolva problema scalării atunci când există mai mult de 10.000 de conexiuni simultane. În serverele construite pe modelul "un fir pe o conexiune", problemele de scalabilitate apar din următoarele motive:

• cheltuieli excesive pentru structurile de date ale sistemului de operare necesare pentru a descrie o sarcină;

• cheltuieli excesive pentru comutarea contextelor.

Aplicația server prin PPE este implementată într-un apel de sistem, care primește evenimentele simultan din mai mulți descriptori (multiplexare). La procesarea evenimentelor utilizate doar operațiile de I/O fără blocare: nici un descriptor nu împiedică procesarea evenimentelor din alți descriptori.

Servere Web:

•Node.js

•Nginx

6. PPE pentru aplicatii desktop

Evenimentele și gestionarea lor sunt esențiale pentru implementarea interfeței grafice de utilizator. Exemplu: interacțiunea programului cu evenimente de la mouse. Apăsarea butonului stâng al mouse-ului determină o întrerupere a sistemului care declanșează o procedură specifică în interiorul sistemului de operare. În această procedură, fereastra de sub cursorul mouse-ului este căutată. Dacă se găsește o fereastră, atunci acest eveniment este trimis la coada de așteptare a mesajelor din această fereastră.

În funcție de tipul ferestrei, pot fi generate evenimente suplimentare.

În C# codul de tratare a evenimentului poate arăta astfel:

```
private void button1_Click(object sender, EventArgs e)

{
MessageBox.Show("Butonul a fost apăsat");
```

7. PPE limbaje de programare

Diferite limbaje de programare suportă PPE în grad diferit. Cel mai complet nivel de susținere a evenimentelor îl au următoarele limbaje(lista este incompletă):

• Perl (evenimente și daemoni DAEMON cu prioritățile lor PRIO),

• Delphi (limbaj de programare),

• ActionScript 3.0,
• C# (evenimente event),

• JavaScript (acțiunile utilizatorului).

În C# evenimentele sunt implementate ca element al limbajului. Exemplu de declarație a unui eveniment:

```
public class MyClass{

public event EventHandler MyEvent;}
```

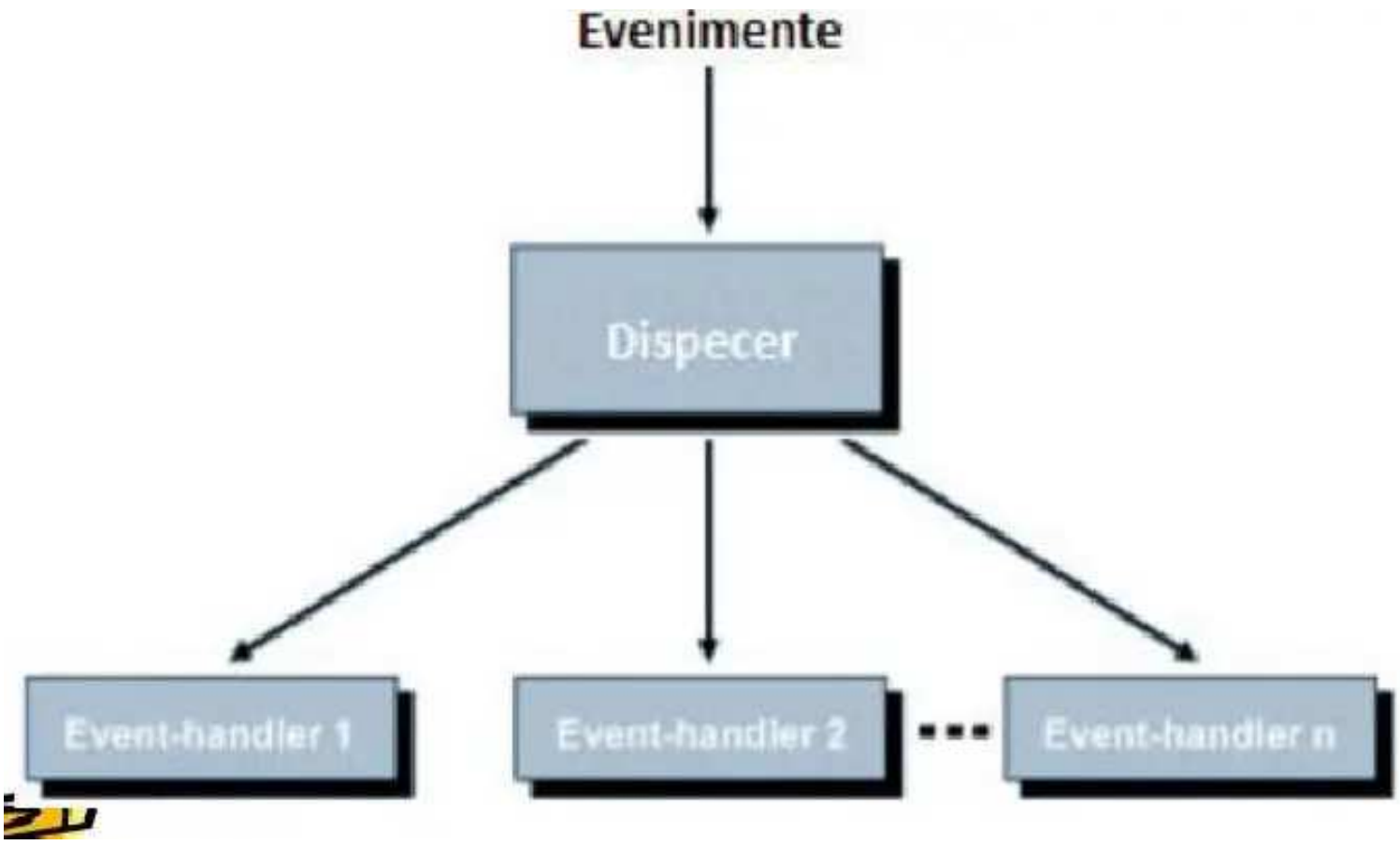
Aici, EventHandler este un delegat care definește tipul procedurii de tratare a evenimentului. Abonarea la eveniment este realizată astfel: myClass.MyEvent += new EventHandler (Handler);

Aici myClass este o instanță a clasei MyClass, Handler este procedura de tratare a evenimentului.

8. Cum functioneaza PPE

Elementul central al unei aplicații pilotate de evenimente este partea programului care recepționează evenimentele (dispecerul) și transmite fiecare eveniment manipulatorului (handler, procedură de tratare) propriu. Dispecerul rămâne activ până când va întâlni un eveniment (de exemplu, "End_Program") care îl va determina să închidă aplicația. În anumite circumstanțe, dispecerul poate întâlni un eveniment pentru care nu există un handler adecvat. În funcție de natura evenimentului, dispecerul poate fie să îl ignore, fie să genereze o excepție. Într-un mediu de programare pilotat de evenimente, evenimentele standard sunt de obicei identificate utilizând ID-ul obiectului afectat de eveniment (ex., numele unui buton de comandă dintr-o formă) și ID-ul propriu al evenimentului (ex., "clic-stânga").

9. Reprezentarea grafica a paradigmei PPE. Pseudo-cod planificator



Do forever:

Get event from input stream

If event type == endprogram: quit

Else if event type == event_01:

Call event-handler for event_01 with params

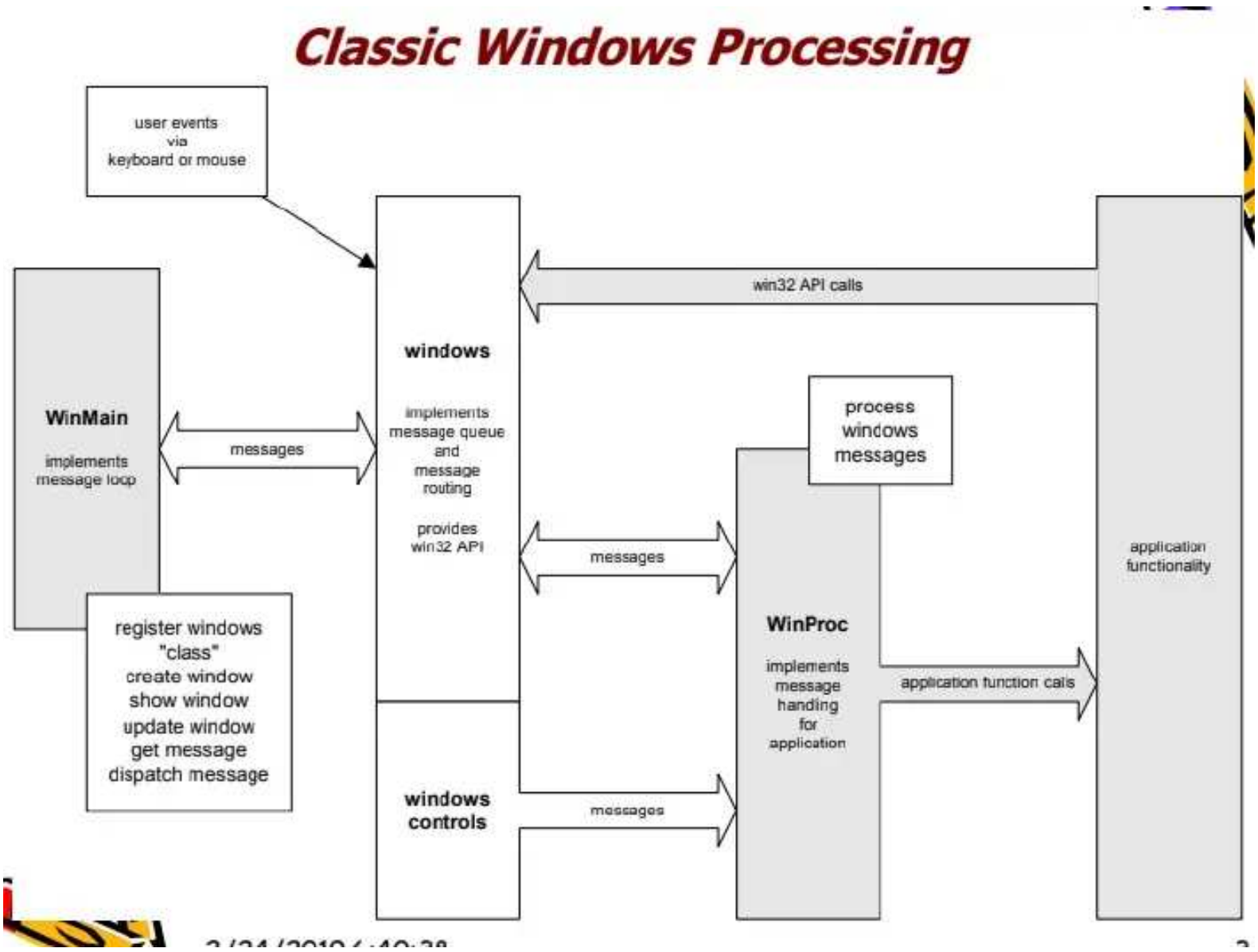
Else handle unrecognized event

End loop

10. Coada de evenimente. Programarea interfetei grafice a utilizatorului

Într-un sistem bazat pe evenimente, frecvența apariției evenimentelor poate fi mare. Dispecerul și event-handlerele s-ar putea să nu poată face față tuturor evenimentelor imediat ce acestea apar. Soluția este de a plasa evenimentele neprocesate într-un fir de așteptare până când acestea vor putea fi preluate pentru a fi tratate. Firul de așteptare garantează că toate evenimentele vor fi tratate la un moment dat și într-o anumită ordine. Lungimea firului și timpul necesar procesării evenimentelor depind de viteza procesorului, capacitatea memoriei operative și numărul de alte aplicații executate în același timp.

Conexiunea dintre user si program este de doua tipuri(linia de comanda si GUI (graphic user interface).



GUI ^

11. Istoria Windows

Windows este o interfaţă grafică cu utilizatorul (GUI - graphical user interface). Elementele grafice asigură o utilizare mai eficientă a spaţiului de afişare, un mediu bogat din punct de vedere vizual pentru transferul de informaţii şi posibilitatea de afişare a rezultatelor aşa cum vor arăta în realitate pe hârtie . Windows reprezintă un SO complet , integrat pe 32/64 biţi şi care se distinge prin: • facilităţi de conectare în reţea; • facilităţi de protected-mode; • facilităţi de multitasking şi multithreading; **AVANTAJUL OFERIT DE MULTITASKING** Prin multitasking noi putem intelege executarea simultana a mai multor taskuri.Traducerea corecta spune doar ca se executa mai multe taskuri fara a se specifica simultaneitatea lor. Avantajul multitaskingului ar fi ca el ne permite sa executam mai multe operatii de care avem nevoie in acelasi timp, de aceea noi putem sa ne permitem comutarea rapida de la un context la altul :

Lucruri noi:

Interfaţa grafică cu utilizatorul (GUI)

Apelurile de funcţii

Programarea orientată pe obiecte

Arhitectura bazată pe mesaje

Procedura de fereastră

12. Istoria Windows

Windows 95 asigură suportul pentru mai mult de o mie de apeluri de funcţii pe care le pot folosi aplicaţiile. Toate funcţiile Windows importante sunt declarate în fişiere antet. Principalul fişier antet se numeşte WINDOWS.H şi include multe alte fişiere antet.

Arhitectura bazată pe mesaje

În Windows, atunci când utilizatorul redimensionează o fereastră, sistemul de operare trimite programului un mesaj prin care îi comunică noile dimensiuni ale ferestrei. Programul poate apoi să modifice conţinutul ferestrei, astfel încât acesta să se adapteze la noile dimensiuni.

„sistemul de operare trimite programului un mesaj" - adica Windows apelează o funcţie din program. Parametrii acestei funcţii descriu mesajul respectiv. Această funcţie din programul Windows este cunoscută sub numele de „procedură de fereastră" („window procedure").

Procedura de fereastră

Orice fereastră creată de un program are asociată o procedură de fereastră. Procedura de fereastră este e funcţie care se poate afla chiar în program sau într-o bibliotecă cu legături dinamice (DLL). Windows trimite un mesaj către o fereastră prin apelarea procedurii de fereastră. Procedura de fereastră prelucrează anumite informaţii pe baza mesajului primit, apoi returnează controlul sistemului de operare.

WinMain şi WndProc

WndProc - trateaza mesajele care vin de la sistemul de operare.Ea mai determina continutul care va fi afisat in zona client al ferestrei si modul cum fereastra va raspunde la ceea ce va face utilizatorul.
Funcţia WinMain reprezintă punctul de intrare în program. Aceasta este echivalentul funcţiei main din programele scrise în limbajul C. Orice program pentru Windows trebuie să aibă o funcţie WinMain.

13. Atributele contextului de dispozitiv. Salvarea contextului de dispozitiv. Funcţiile LineTo, Polyline şi PolylineTo, PolyPolyline, Arc, PolyBezier şi PolyBezierTo, Rectangle, Ellipse, RoundRect, Chord şi Pie.

a) **Contextul de dispozitiv** (prescurtat DC - device context) - este o structură de date întreţinută intern de interfaţa GDI. Fiecare context de dispozitiv este asociat unui anumit dispozitiv de afişare, cum ar fi imprimanta, plotterul sau monitorul video.O parte dintre valorile din contextul de dispozitiv sunt atribute grafice. Atributele contextului de dispozitiv controlează caracteristicile

textului. afişat. De exemplu, între atributele contextului de dispozitiv stabileşte culoarea prestabilită este negru. De asemenea, contextul prestabilit şte dispozitiv stabileşte ca fondul să fie alb. Atunci când un program afişează un text pe ecran, Windows foloseşte această culoare de fond ca să umple spaţiul dreptunghiular care înconjoară fiecare caracter, spaţiu numit „casetă caracter" („character box")

b) Salvarea contextului de dispozitiv - Variabila handle a contextului de dispozitiv este eliberata de functia BeginPaint. In general aceasta este salvata intr-o variabila numita hdc. In procedura de fereastra se defineste aceasta variabila astfel : HDC hdc; Tipul de date HDC este definit ca un întreg fără semn, pe 32 de biţi. Apelul funcţiei EndPaint eliberează variabila handle a contextului de dispozitiv.

c) Functiile : Windows poate să deseneze linii drepte, linii eliptice (linii curbe, pe circumferinţa unei elipse) şi curbe Bezier.

Functia Polyline – foloseste ca punct de plecare pozitia curenta, si stabileste ca pozitie curenta sfirsitul ultimei linii desenate.

Funcţia LineTo este una dintre puţinele funcţii GDI care nu are nevoie de dimensiunile complete ale obiectului ce urmează să fie desenat. Funcţia LineTo desenează o linie de la „poziţia curentă" definită în contextul de dispozitiv pană la punctul specificat la apelarea funcţiei (exclusiv acest punct). Poziţia curentă este folosită ca punct de plecare şi de alte funcţii GDI. Dacă apelaţi

funcţia LineTo fără să stabiliţi mai întâi poziţia curentă, funcţia desenează o linie pornind din colţul din stânga-sus al zonei client.
Rectangle(hdc,xLeft,yTop,xRight,yBottom)(desenarea unui dreptunghi)
Ellipse (hdc, xLeft, yTop, xRight, yBottom) ;(ne permite desenarea elipsei)
RoundRect (hdc, xLeft, yTop, xRight, yBottom, xCornerEllipse, yCornerEllipse) ; (Funcţia pentru desenarea unui dreptunghi cu colţurile rotunjite)
Functiile PolyBezier si PoluBezierTo se folosesc pentru trasarea uneia sau a mai multor curbe Bezier conexe: PolyBezier (hdc, pt, iCount) ; sau instrucţiunea: PolyBezierTo (hide, pt, iCount) ;

14. Bazele utilizarii cronometrului

Cronometrul Windows este un dispozitiv de intrare, ce comunică periodic unei aplicaţii trecerea unui anumit interval de timp. Utilizari ale cronometrului:

■ Multitaskingul - Dacă programul trebuie să execute o operaţie de durată, o poate împărţi în operaţii mai mici, pe care să le prelucreze la primirea unui mesaj WMJTIMER.

■ Actualizarea unui raport de stare - Un program poate să utilizeze cronometrul pentru afişarea „în timp real" a unor informaţii care se schimbă continuu, în funcţie de resursele sistemului sau de evoluţia unei anumite operaţii.

■ Implementarea unei caracteristici de „autosalvare" - Cronometrul poate să „amintească" unui program Windows să salveze pe hard-disc documentul la care lucrează utilizatorul, după trecerea unui anumit interval de timp.(Microsoft Word are aceasta functie incorporata)

■ Închiderea versiunilor demo ale unui program - Unele versiuni demonstrative ale programelor sunt proiectate astfel încât să-şi încheie execuţia, de exemplu, după trecerea a 30 de minute de la lansare, precum s-a epuizat timpul pentru trial version.

15. Care este punctul de intrare intr-un program windows

Funcţia WinMain reprezintă punctul de intrare în program. Aceasta este echivalentul funcţiei main din programele scrise în limbajul C. Orice program pentru Windows trebuie să aibă o funcţie WinMain

16. Ce este notatia ungara

Notatia ungara consta in aceea ca fiecare nume de variabilă începe cu una sau mai multe litere mici care specifică tipul de date al variabilei. De exemplu, prefixul sz al variabilei szCmdLine semnifică „şir de caractere terminat cu zero". Prefixul h al variabilelor hInstance şi hPrevInstance înseamnă „variabilă handle";

17. Ce este variabila handle si destinatia ei

Variabila handle este un nr pe 32 biti, care face trimitere la un anumit obiect. Valoarea reală a variabilei handle nu este importantă pentru program, dar modulul Windows care o furnizează programului ştie cum să îl manipuleze pentru trimiterea la obiect

18. Ce este device context

Device Context – structura de date intretinuta intern de interfata GDI. O parte din valorile din DC sunt atributele grafice , ce definesc careva particularitati referitor la modul de lucru a unor functii de desenare din interfata GDI

19. Ce este multitasking controlat

Multitasking-ul controlat este procesul in care pentru executarea mai multor task-uri simultan se foloseste ceasul, care aloca timpi de procesare a diferitor programe care ruleaza in sistem. Unele din programe cedeaza controlul pentru ca altele sa isi poate continua executia. El a aparut incepand cu windwos 95, si programele se impart in mai multe fire de executie, care ruleaza in acelasi timp.

20. Ce facem daca cronometrul nu este accesibil

Cronometrul Windows este un dispozitiv de intrare, ce comunică periodic unei aplicaţii trecerea unui anumit interval de timp. Programul specifică sistemului de operare acest interval de timp, iar acesta la randul sau ii trimite un raspuns ca au trecut de exemplu 10000ms. Dacă nu este disponibil nici un cronometru, funcţia SetTimer returnează valoarea NULL.

Secvenţa de cod din exemplul următor afişează o casetă de mesaje atunci când funcţia SetTimer nu poate aloca programului un cronometru:

```
 if (!SetTimer (hwnd, 1, 1000, NULL))
{ MessageBox (hwnd, "Too many clocks or timers!", "Program Name", MB_ICONEXCLAMATION ]
MB_OK);
 return FALSE ; }
```

21. Ce prezinta procedura unei ferestre

Fiecare din ferestrele create intr-un program contine o procedura de fereastra. Windowsul trimite un mesaj catre fereastra prin apelarea procedurii de fereastra cu anumite argumente(continutul mesajului). La fel aceasta se poate numi can un mod de incapsulare a codului care raspunde intrarilor si afiseaza elemente grafice pe ecran(la fel in baza mesajelor trimise).

22. Ce prezinta procedura unei ferestre

Clasa butoanelor este clasa predefinita care permite crearea controalelor ferestre descendente sub forma de butoane.Aceasta are o procedura de fereastra(aflata in dll) care raspunde de prelucrarea mesajelor trimise catre ferestrele butoanelor, ca mai apoi acestea sa reactioneze intr-un mod sau altul la anumite actiuni ale userului. Butonul transmite mesajul catre procedura ferestrel prin WM_COMMAND.

23. Coduri virtuale de taste. Starea tastelor de modificare. Utilizarea mesajelor de actionare a tastelor.

Fiecare tasta de pe claviatura are un numar care este numit codul virtual si se contine in wParam ale mesajelor WM_KEYUP/KEYDOWN. **Starea** poate fi obtinuta cu functia GetKeyState. Daca avem tasta shift apasata, atunci functia getkeystate(vk_shift) returneaza o valoare negative. **Utilizarea.** wm_keydown este utilizat pentru tastele care nu returneaza caractere, adica pentru tasta de deplasare, shift, capslock, etc.

24. Crearea ferestrelor child.

Fereastra descendenta prelucrează mesajele primite de la mouse şi de la tastatură şi înştiinţează fereastra părinte atunci când starea proprie se modifică. În acest fel, fereastra descendent devine un dispozitiv de introducere a datelor pentru fereastra părinte. Fiecare fereastră descendent este creată printr-un apel al funcţiei CreateWindow. Putem crea atat ferestre descendente noi, cat si cele predefinite ca button, scrollbar, listbox, checkbox. Sistemul de operare Windows conţine deja procedurile de fereastră pentru prelucrarea mesajelor către fereastra descendent pe baza acestor clase. Procedura ferestrei părinte trimite mesaje către controlul de tip fereastră descendent, iar acesta trimite mesaje către procedura ferestrei părinte.

25. Cum poate fi obtinuta variabil handle a unui device context.

Variabila handle poate fi obtinuta in mai multe cazuri. In primul caz se utilizeaza tratarea mesajelor WM_PAINT. (hdc=beginpaint(hwnd,&ps);. Valoarea returnata de functia beginpaint si este variabila handle a contextului de dispozitiv. Un alt mod este utilizand functia GetDC. De exemplu hdc=GetDC(hwnd);

26. Curbe bezier. Functia PolyBezierTo.

O curbă Bezier este definită prin patru puncte - două capete şi două puncte de control. Capetele curbei sunt ancorate în cele două puncte finale. Punctele de control acţionează ca nişte „magneţi" care deformează linia dreaptă dintre cele două puncte finale. Funcţia PolyBezierTo foloseşte poziţia curentă ca punct de început pentru prima curbă Bezier. Fiecare curbă desenată are nevoie doar de trei puncte. La returnarea funcţiei, poziţia curentă este punctul final al ultimei curbe desenate. Sintaxa: PolyBezierTo (hdc,pt, iCount) // pt-matrice de structure point, iCount – numarul de puncte din matrice, 1+3*nr curbelor.

27. Cursorul de editare(CARET).

Există cinci funcţii principale pentru cursorul de editare:

♣ CreateCaret - creează un cursor de editare asociat unei ferestre.
♣ SetCaretPos - stabileşte poziţia cursorului de editare în fereastră.
♣ ShowCaret - afişează cursorul de editare.
♣ HideCaret - maschează cursorul de editare.
♣ DestroyCaret - distruge cursorul de editare creat

Cursorul de editare este, de obicei, o linie ori un bloc orizontal de dimensiunea unui caracter, sau o linie verticală. Cursorul de editare este afişat în fereastra care deţine cursorul de intrare (input focus). La un moment dat, o singură fereastră poate detine cursorul de intrare, asa că existenta unui singur cursor de editare este logică. Principala regulă de folosire a cursorului de editare este simplă. O procedură de fereastră apelează funcţia CreateCaret în timpul prelucrării mesajului WM_SETFOCUS şi funcţia DestroyCaret în timpul prelucrării mesajului WM_KILLFOCUS. Dupa creare e nevoie sa-l afiseze cu functia ShowCaret, si sa-l mascheze cand nu e nevoie cu HideCaret.

28. Desenarea suprafetelor pline. Functia Polygon si modul de umplere. SetPolyFillMode.

Pentru **desenarea suprafeţe pline** sunt utilizate 7 funcţii windows:

Funcţie	Figura
Rectangle	Dreptunghicucolţuridrepte Elipsă
Ellipse	Dreptunghi cu colţuri rotunjite
RoundRect	Arcpecircumferinţauneielipse,avândcapeteleuniteprintr-o coardă
Polygon	Figurăgeometricăavândmaimultelaturi
PolyPolygon	Maimultefiguri geometrice cu mai multe laturi

FuncţiaPolygonesteutlizatăpentruadesenadiversepoligoanepebazaunuisistemdepuncte.Eaaredrept parametric variabilahandleacontextuluidedispozitiv,unpointerla omatricedestructurePOINT,şinumăruldepuncte din matrice: Polygon (hdc, pt, iCount) .

SetPolyFillMode (hdc, iMode)

Pentru a colora este nevoie de a utiliza pensula, care se creaza si se utilizeaza in modul urmator: hBrush = CreateSolidBrush (rgbColor); selectobject(hdc,hbrush);

29. Dimensiunea unui caracter, GETTEXTMETRICS, TEXTMETRIC .

Dimensiunea unui caracter este necesară în cazul utilizării funcţiei TextOut pentru afişarea mai multor linii. Spaţiul dintre liniile succesive de text poate fi stabilit pe baza înălţimii unui caracter iar spaţiul dintre coloane pe baza lăţimii medii a caracterelor din font. Dimensiunile caracterelor sunt obtinute prin apelarea funcţiei GetTextMetrics. Funcţia GetTextMetrics are ca parametru o variabilă handle a contextului de dispozitiv, deoarece returnează informaţii despre fontul selectat în contextul de dispozitiv. Windows copiază valorile referitoare la dimensiunile caracterelor într-o structură de tip TEXTMETRIC. Valorile sunt exprimate în unităţi de măsură care depind de modul de mapare selectat în contextul de dispozitiv. În contextul prestabilit de dispozitiv, modul de mapare este MM_TEXT, aşa că dimensiunile sunt date în pixeli.

TEXTMETRIC tm; hdc = GetDC(hwnd); GetTextMetrics(hdc, &tm) ; ReleaseDC(hwnd, hdc);

30. Dimensiunea zonei client. LOWORD HIGHWORD. Barele de derulare. Domeniu si pozitia unei bare de derulare.

O metodă obişnuită de determinare a dimensiunilor zonei client a unei ferestre este prelucrarea mesajului WM_SIZE în procedura de fereastră. Windows trimite un mesaj WM_SIZE către procedura de fereastră, de fiecare dată când se modifică dimensiunile ferestrei. Parametrul lParam transmis procedurii de fereastră conţine lăţimea zonei client în cuvântul mai puţin semnificativ (LOWORD) şi înălţimea zonei client în cuvântul mai semnificativ (HIWORD). Codul pentru prelucrarea acestui mesaj arată astfel: static int cxClient, cyClient ; [alte linii de program] case WM_SIZE : cxClient = LOWORD (lParam) ; cyClient = HIWORD (lParam) ; return 0 ;

Fiecare **bară de derulare** are asociate un „domeniu" (definit printr-o pereche de numere întregi care reprezintă valorile maximă şi minimă) şi o „poziţie" (punctul în care se află caseta de derulare în domeniul asociat barei de derulare). In mod prestabilit, domeniul unei bare de derulare este de la 0 (sus sau în stânga) la 100 (jos sau în dreapta) dar poate fi uşor modificat astfel încât să aibă o formă mai convenabilă pentru program: SetScrollRange (hwnd, iBar, iMin, iMax, bRedraw) ;

31. Elemente de baza despre mouse

Windows 95 permite folosirea mouse-ului cu un buton, cu două butoane sau cu trei butoane, precum şi folosirea unui joystick sau a unui creion optic, pentru simularea unui mouse cu un buton. Totuşi, cel mai utlizat este mouse-ul cu trei butoane, devenit un standard de facto. Al doilea buton al mouse-ului este recomandat pentru apelarea „meniurilor de context" - meniuri care apar în fereastră în afara barei de meniu - sau pentru operaţii speciale de tragere. Pentru a determina dacă mouse-ul este prezent se foloseşte funcţia GetSystemMetrics. fMouse = GetSystemMetrics (SM_MOUSEPRESENT). Valoarea returnata va fi 1 in cazul in care mousul este present.

32. Ferestre child de control si culoare

Pentru a îmbunătăţi aspectul acestor butoane trebuie ori să schimbăm culoarea zonei client, aşa încât aceasta să se asorteze cu fondul butoanelor, ori să schimbăm culoarea folosită pentru fondul butoanelor. Culorile de sistem pentru butoane: COLOR_BTNFACE este folosită pentru culoarea suprafeţei principale a butoanelor de apăsare şi pentru fondul celorlalte butoane. (De asemenea, aceasta este culoarea de sistem folosită pentru casete de dialog şi casete de mesaje.) COLOR_BTNSHADOW este folosită pentru sugerarea unei umbre la marginile din dreapta şi de jos ale

butoanelor de apăsare, precum şi în interiorul pătratelor din casetele de validare şi a cercurilor din butoanele radio. COLOR_BTNTEXT- culoarea textului în cazul butoanelor de apăsare COLOR_WINDOWTEXT - culoarea textului pentru pentru celelalte butoane.

33. Ferestre child definite de programator

Crearea unui buton cu stilul BS_OWNERDRAW oferă controlul complet asupra aspectului vizual al butonului. Un buton creat cu stilul BS_OWNERDRAW trimite ferestrei părinte un mesaj WM_DRAWITEM de fiecare dată când butonul trebuie să fie redesenat. Aceasta se întâmplă la crearea butonului, de fiecare dată când este apăsat sau eliberat, când obţine sau pierde cursorul de intrare şi în orice altă situaţie când trebuie să fie redesenat.

34. Ferestre child. Butoane si input focus

Butoanele de apăsare, butoanele radio şi butoanele desenate de proprietar primesc cursorul de intrare atunci când se execută clic pe acestea. Controlul indică faptul că deţine cursorul de intrare printr-o linie punctată care înconjoară textul. Atunci când o fereastră descendent primeşte cursorul de intrare, fereastra părinte îl pierde; toate intrările de la tastatură sunt direcţionate către control, nu către fereastra părinte. O fereastră părinte poate împiedica un control de tip fereastră descendent să primească cursorul de intrare prelucrând mesajul WM_KILLFOCUS.

29.Ferestre child. Clasa barelor de derulare.

Spre deosebire de butoane, barele de derulare nu trimit mesaje WM_COMMAND către fereastra părinte. în schimb, trimit mesaje WM_VSCROLL şi WM_HSCROLL, ca şi barele de derulare ale ferestrelor. Deşi barele de derulare ale ferestrelor au lăţime fixă, în cazul controalelor de tip bare de derulare Windows foloseşte dimensiunile dreptunghiului specificate la apelarea funcţiei *CreateWindow* sau a funcţiei *MoveWindow.*

Putem folosi funcţia *GetSystemMetrics* ca să obţinem înălţimea unei bare de derulare orizontale:

GetSystemMetrics (SH_CYHSCROLL) ;

sau lăţimea unei bare de derulare verticale:

GetSystemMetrics (SM_CXVSCROLL) ;

Putem să stabili intervalele şi poziţia unui control de tip bară de derulare cu aceleaşi apeluri pe care le folosim şi pentru barele de derulare ale ferestrelor:

SetScrollRange (hwndScroll , SB CTL, iMin, iMax, bRedraw) ; SetScrollPos (hwndScroll , SB CTL, iPos, bRedraw) ; SetScrollInfo (hwndScroll , SB_CTL, &si, bRedraw) ;

30.Ferestre child. Clasa de redactare.

Atunci cand creati o fereastra descendent a folosind numele de clasa ,,edit", definiti un dreptunghi pe baza coordonatelor *x* si *y* si a parametrilor inaltime si latime - specificati la apelarea functiei *CreateWindow.* Acest dreptunghi contine text care poate fi editat. Atunci cand fereastra descendent detine cursorul de intrare, puteti sa introduced text de la tastatura, sa deplasati cursorul, sa selectati portiuni din text cu ajutorul mouse-ului sau al tastei Shift si al tastelor de deplasare, sa stergeti textul selectat si sa pastrati in memoria

temporara (Clipboard) apasand tastele Ctrl+X, sa il copiati cu ajutorul tastelor Ctrl+C sau sa inserati in controlul de editare textul din memoria temporara apasand tastele Ctrl+V.

31.Ferestre child. Clasa static.

Clasa statica nu acceptă intrări de la mouse sau de la tastatură şi nu trimitwe mesaje WM_COMMAND către fereastra părinte. (Atunci când executaţi clic pe o fereastră descendent statică, aceasta interceptează mesajul WM_NCHITTEST şi returnează o valoare HTTRANSPARENT către Windows. Ca rezultat, Windows trimite acelaşi mesaj WM_NCHITTEST către fereastra aflată dedesubt, care de obicei este fereastra părinte. în general, fereastra părinte transmite mesajul primit către procedura *DefWindowProc,* unde acesta este convertit într-un mesaj de mouse din zona client.)

Clasele statice includ şi trei stiluri de text: SS_LEFT, SS_RIGHT şi SS_CENTER. Acestea creează rubrici de text aliniate la stânga, la dreapta sau centrate.