3 Controlul evenimentelor

3.1 Scopul lucrării

În acest laborator se vor se vor studia moduri de preluare a evenimentelor generate de către interfaţa grafică (GUI). Temele de laborator se bazează pe aplicaţia realizată în laboratorul precedent şi de asemenea aplicaţia rezultată la finele acestui laborator va fi utilizată în lucrările viitoare (se recomandă aşadar salvarea aplicaţiei).

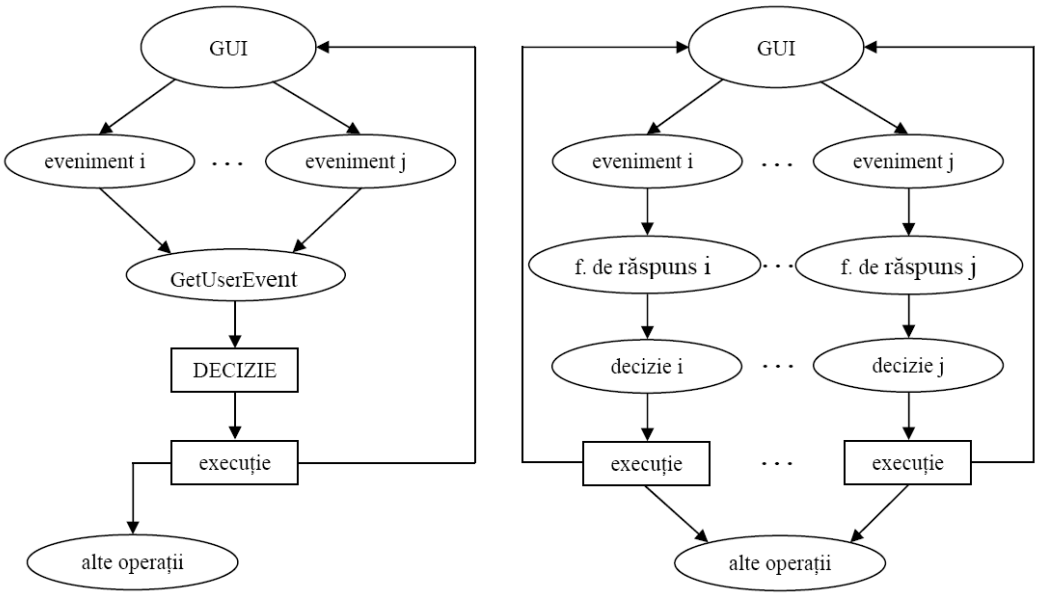
3.2 Introducere

Odată ce s-a generat o interfaţă cu utilizatorul, trebuie dezvoltat un program C care va procesa evenimentele generate de aceasta şi va controla execuţia. CVI ne oferă două metode de bază pentru proiectarea unei aplicaţii:

- funcţii de răspuns (callback functions);
- bucle pe evenimente (event-loops).

Atunci când se folosesc funcţii de răspuns, acestea vor fi descrise de către programator şi vor fi apelate automat de controalele UI. Ca exemplu să luăm o funcţie din program numită *ManualAcquisitionCB* ce este ataşată la un buton numit *Achiziţie*. Oricând se apasă butonul *Achiziţie* toate evenimentele generate în UI la apăsarea lui vor fi transmise direct în această funcţie (ca parametru) unde va trebui să fie scris cod pentru interpretarea lor.

În cazul folosirii buclelor pe evenimente, toate evenimentele produse de UI sunt transmise funcţiei **GetUserEvent** care identifică ce control a generat evenimentul şi funcţie de rezultat se trece la procesarea lor.



a) buclă de evenimente b) funcţii de callback

Figura 5.1 Metode de control ale UI

Ambele metode au avantaje şi dezavantaje. În cadrul proiectării unei aplicaţii se poate aborda fie o metodă, fie alta, fie o mixtură. Această decizie se ia funcţie de necesităţile aplicaţiei şi de experienţa programatorului.

3.3 Controlul unei interfeţe grafice

Anumite operaţii pe care le face un utilizator, cum ar fi selectarea unui element de meniu, apăsarea unui buton sau introducerea unei valori se numesc *evenimente*. Biblioteca de interfaţă cu utilizatorul furnizează legătura dintre evenimente şi codul sursă.

Astfel programul poate recunoaşte evenimente şi răspunde la acestea prin executarea unui anumit cod. Mai jos se vor prezenta evenimentele ce pot fi generate de către o interfaţă grafică CVI.

Eveniment pentru	**Eveniment**	**Informaţii pasate codului**
- controale - meniuri	EVENT_COMMIT	ID panou sau bară de meniuri ID control sau meniu
- controale	EVENT_VAL_CHANGED	ID panou ID control
- controale - panouri	EVENT_LEFT_CLICK	ID panou, ID control, coordonate mouse
	EVENT_DOUBLE_CLICK	ID panou, ID control, coordonate mouse
	EVENT_RIGHT_CLICK	ID panou, ID control, coordonate mouse
	EVENT_RIGHT_DOUBLE_CLICK	ID panou, ID control, coordonate mouse
	EVENT_KEYPRESS	ID panou, ID control, cod tastă, pointer la cod tastă
	EVENT_GOT_FOCUS	ID panou, ID control
	EVENT_LOST_FOCUS	ID panou, ID control
	EVENT_DISCARD	ID panou, ID control
- timere	EVENT_TIMER_TICK	Pointer la timpul curent, pointer la timpul scurs de când funcţia de răspuns a primit ultimul EVENT_TIMER_TICK
- panouri	EVENT_CLOSE EVENT_PANEL_SIZE EVENT_PANEL_MOVE	ID panou ID panou ID panou
- funcţia principală de răspuns	EVENT_IDLE EVENT_END_TASK	Învechit. Se vor folosi timere. Primit când Windows face o cerere de închidere a aplicaţiei.

Figura 5.2 Tabel cu evenimente

3.4 Folosirea funcţiilor de răspuns pentru a trata evenimentele generate de interfaţa cu utilizatorul

Funcţiile de răspuns pot trata oricare din evenimentele prezentate anterior. Aceste funcţii au prototipuri (declarate în fişierul header generat automat de CVI la salvarea interfeţei) pentru panouri, bare de meniu, controale.

Când utilizatorul generează un eveniment pe un obiect al interfeţei se va executa funcţia ataşată acestuia. Următorul exemplu de sursă ilustrează conceptul de funcţie de răspuns pentru un panou, două controale şi un meniu.

```
//în funcția main:
panelHandle = LoadPanel(...); // încarc panoul
DisplayPanel(panelHandle, ...); // îl afișez
menubarHandle = LoadMenuBar(...); // încarc bara de meniuri
RunUserInterface() // lansez în execuție

// interfața utilizator
int CVICALLBACK OnMainPanelCB(int handle, int event, void*callbackdata, int eventdata1, int
eventdata2)
{
    switch (event)
    {
        case EVENT_PANEL_SIZE :
             .. //codul ce va trata modificarea dimensiunii panoului
        break;

        case EVENT_PANEL_MOVE :
             .. // codul ce va trata mișcarea panoului
        break;

        case EVENT_KEYPRESS :
            .. // codul ce tratează tasta apăsată; eventdata1 și
            .. // eventdata2 conțin informații despre tastă
         break;
    }
return(0);
}

int CVICALLBACK ControlResponse (int handle, int control, int event, void *callbackdata, int
eventdata1, int eventdata2)
{
    if (control == PANEL_CONTROL1)
    {
        switch (event)
        {

            case EVENT_RIGHT_CLICK :
                .. // se tratează apăsarea pe butonul din dreapta al
                .. // mouse-ului în zona controlului 1
            break;

            case EVENT_VAL_CHANGED :
                .. // se tratează modificarea valorii controlului 1
             break;

             case EVENT_COMMIT :
                 .. // se tratează un eveniment aplicat acestui control
             break;
         }
    }
    if (control == PANEL_CONTROL2)
    {
        switch (event)
        {
```

```
            case EVENT_RIGHT_CLICK :
                .. // se tratează apăsarea pe butonul din dreapta al
                .. // mouse-ului în zona controlului 2
            break;
            case EVENT_COMMIT :
                .. // se tratează un eveniment aplicat acestui control
            break;
        }
    }
    return(0);
}

int CVICALLBACK OnMenuBarCB (int menubar, int menuitem, void *callbackdata, int panel)
{
    switch (menuitem)
    {
        case MENUBAR_MENU1_ITEM1:
            .. // codul corespunzător activării ITEM1 din cadrul
            .. // MENU1 ce aparţine barei de meniuri MENUBAR
        break;
        case MENUBAR_MENU1_ITEM2:
            .. // codul corespunzător activării ITEM2 din cadrul
            .. // MENU1 ce aparţine barei de meniuri MENUBAR
        break;
    }
    return(0);
}
```

Notă: Dacă ataşaţi funcţii de răspuns în editorul de interfeţe cu utilizatorul, acestea vor fi automat instalate prin apelul funcţiilor ***LoadPanel*** şi ***LoadMenuBar***, altfel trebuie să se instaleze prin program (folosind funcţia ***InstallPanelCallback***, ***InstallCtrlCallback***, etc.).

3.5 Folosirea funcţiei *GetUserEvent* pentru a trata evenimentele interfeţei cu utilizatorul

Funcţia ***GetUserEvent*** returnează numai evenimente de tipul *EVENT_COMMIT* (cu excepţia evenimentelor definite de programator prin ***QueueUserEvent***. Următorul exemplu de sursă ilustrează conceptul de buclă de tratare a evenimentelor:

```
#define USER_EVENT      1000
#define MYEVENT_1       (USER_EVENT+1)
#define MYEVENT_CLOSE   (USER_EVENT+2)

//...

//în funcţia main:
int event = 0; //evenimentul prins de functia GetUserEvent. Poate fi un eveniment postat de
               //programator sau EVENT_COMMIT
int val1 = 0; //prima valoare postată odată cu evenimentul (QueueUserEvent). In cazul unui
              //eveniment EVENT_COMMIT val1 va conţine handle-ul la panel
int val2 = 0; //a doua valoare valoare postată odată cu evenimentul (QueueUserEvent). In cazul
              //unui eveniment EVENT_COMMIT val2 va conţine id-ul controlului

BOOL quitFlag = FALSE; //flag pentru ieşirea din bucla de tratare a evenimentelor

panelHandle = LoadPanel(...); // încarc panoul
DisplayPanel(panelHandle,...); // îl afişez
menubarHandle = LoadMenuBar(...); // încarc bara de meniuri

while (!quitFlag)
{
```

```
    event = GetUserEvent(1, &val1, &val2)
    switch (event)
    {
        case MYEVENT_1:
            .. // tratarea evenimentului 1 postat de programator
        break;

        case MYEVENT_CLOSE :
            .. // tratarea evenimentului 1 postat de programator
            quitFlag = TRUE;
        break;
    }
}

//...

//in funcţia de callback a panelului
switch(event)
{
    case EVENT_CLOSE:
        QueueUserEvent(MYEVENT_CLOSE, panel, MAIN_PANEL);
        break;
}

//...

//in oricare altă funcţie
int v1 = 10;
int v2 = 5;
QueueUserEvent(MYEVENT_1, v1, v2);
```

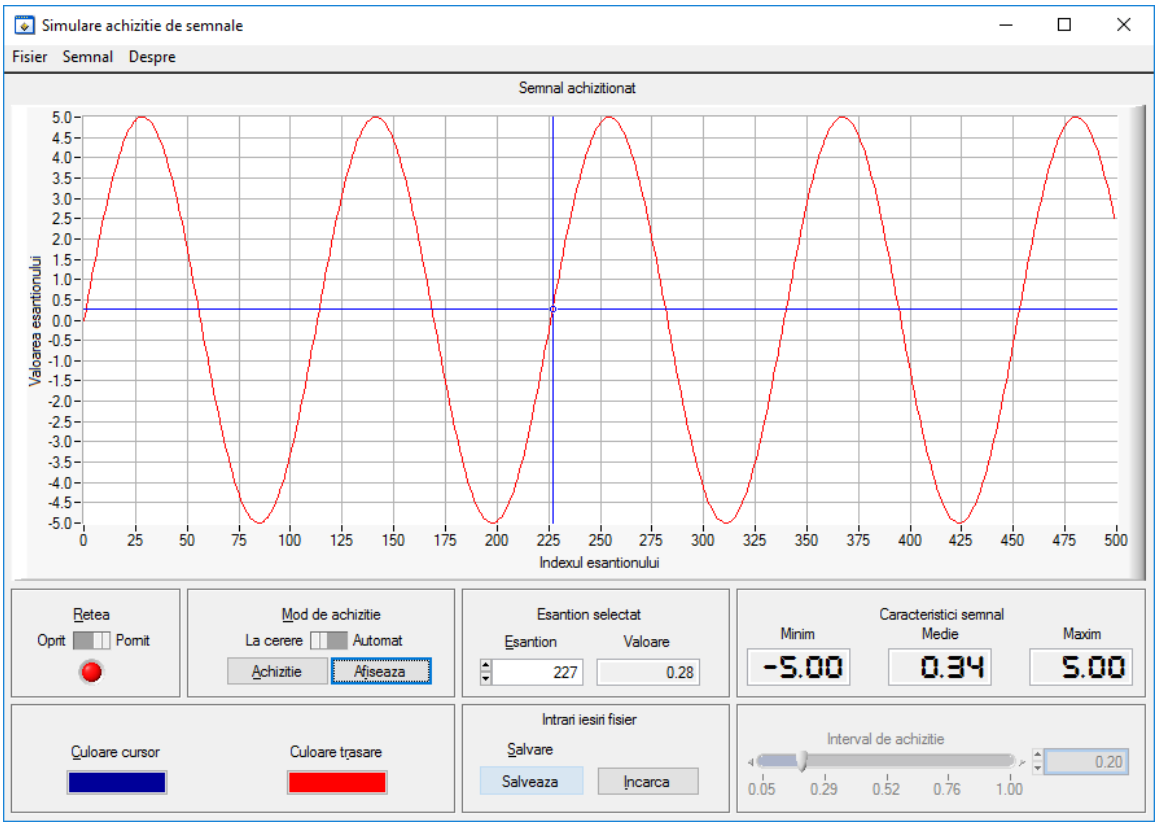
Această buclă de captare a evenimentelor va înlocui bucla standard implementată de mediul CVI prin apelul funcţiei *RunUserInterface* din funcţia *main* (se va comenta sau se va şterge apelul funcţiei *RunUserInterface*).

Funcţia *QuitUserInterface*, utilizată până acum pentru părăsirea aplicaţiei nu mai poate fi utilizată în acest scop deoarece ea practic întrerupea bucla generată de funcţia *RunUserInterface*. Din acest motiv, atunci când se utilizează o buclă de evenimente definită de programator (cum este cea de sus, de exemplu) va trebui să se implementeze un mecanism de părăsire a aplicaţiei.

O idee de implementare a părăsirii aplicaţiei atunci când se foloseşte buclă de evenimente definită de programator este de a capta evenimentele ce trebuie să cauzeze ieşirea din aplicaţie (apăsarea pe un buton de exit sau apăsarea pe butonul X al interfeţei) şi de a întrerupe bucla *while* printr-un *break*.

Notă: Se observă că nici evenimentul de apăsare pe X al panoului (*EVENT_CLOSE*) şi nici funcţia de callback al itemului *EXIT* al meniului nu generează evenimente de tip *EVENT_COMMIT*, deci ele nu vor putea fi captate direct de către funcţia *GetUserEvent*. Din acest motiv, în callback-ul acestor elemente va trebui generat un eveniment nou, prin apelul funcţiei *QueueUserEvent*. Acest eveniment va fi captat în bucla de evenimente definită de programator.

Teme



Creaţi un proiect şi includeţi fişierele *AcquisitionSimulation.c*, *AcquisitionSimulation.h*, *AcquisitionSimulation.uir*, *Signals.c*, *Signals.h*. Implementaţi cerinţele de mai jos:

1. Definiţi un eveniment *MYEVENT_CLOSE* şi închideţi aplicaţia postând acest eveniment. Atenţie apelul funcţiei *QuitUserInterface* nu mai are efect deoarece funcţia *RunUserInterface* este înlocuită de o bucla *while*. (*//To do: 01*)

2. În funcţia de callback a barei de meniu codificaţi selecţia tipului de semnal (sinus, triunghi, dreptunghi sau zgomot) folosind o variabile globală şi tipuri de semnal definite. Completaţi funcţia de callback a butonului de *Achiziţie* pentru a genera semnalul corespunzător. Observaţie: pentru generarea semnalelor se vor folosi funcţii specializate ale CVI (a se vedea fereastra stânga jos, la categoria *Libraries -> Advanced Analysis Library: SineWave*, *TriangleWave*, *SquareWave*, *WhiteNoise*). (*//To do: 02*)

3. Completaţi funcţia de callback a butonului de *Afisare* pentru a afişa semnalul generat (*PlotY*). Orice plotare va fi precedată de o ştergere (*DeleteGraphPlot*) utilizând handle-ul la plotul anterior. Calculaţi şi

afişaţi minimul, maximul şi media semnalului (*MaxMin1D*, *Mean*) (*//To do: 03*)

4. În callback-ul switch-ului *Mod de Achizitie* activaţi şi dezactivaţi timerul de simulare achiziţie după cum poziţia butonului va fi: automat/la cerere. Modificaţi perioada acestui timer în callback-ul controlului *Interval de achizitie*. Acest timer va realiza aceleaşi operaţii ca la cerinţele 2 şi 3. (*//To do: 04*)

5. Actualizaţi culorile cursorului şi a formei de undă semnalului conform cu setările controalelor *Culoare cursor* şi *Culoare trasare*. (*SetCursorAttribute, SetPlotAttribute, PlotY*) (*//To do: 05*)

6. Actualizaţi controalele *Eşantion* (index) şi *Valoare* în funcţie de poziţia curentă a cursorului. Conţinutul celor două controale vor fi actualizate ori de câte ori cursorul îşi modifică poziţia. (*GetGraphCursor*) (*//To do: 06*)

7. La modificarea controlului *Eşantion* poziţia cursorului va trebui să se modifice si se va indica noua valoare corespunzătoare cursorului. (*//To do: 07*)

8. La apăsarea butonului *Salvare*se va:
   a. construi un nume de fişier de forma *RowSignal_aaaa.ll.zz_hh-mm-ss.dat*
   b. va seta un flag care va memora starea butonului de salvare (1 – apasat, 0 – ridicat)
   c. dezactiva/activa butonul *Incarca*
   d. se va scrie în fişier semnalul afişat

   (*ArrayToFile, GetSystemDate, GetSystemTime*) (*//To do: 08*)
   În cazul în care nu se selectează nici un fişier, se va asigura continuarea în siguranţă a aplicaţiei

9. La apăsarea butonului *Incarcare* se va citi si afisa un semnal dintr-un fişier selectat printr-un dialog *FileSelectPopup*. (*FileToArray*) (*//To do: 09*) În cazul în care nu se selectează nici un fişier, se va asigura continuarea în siguranţă a aplicaţiei. Atenţie şi la minim, maxim, medie, cursor, culori

10. Creaţi un nou panel (*About*), încărcaţi-l în memorie şi afişaţi-l/îndepartaţi-l pe/de pe ecran. (*LoadPanel, InstallPopup, RemovePopup*) (*//To do: 10*)

Cuprins