ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ
ΙΔΡΥΜΑ ΠΑΤΡΩΝ
ΣΧΟΛΗ Σ.Δ.Ο.
ΤΜΗΜΑ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ

Πτυχιακή Εργασία

ΘΕΜΑ

ΠΡΟΓΡΑΜΜΑ VIDEO CLUB ΓΙΑ WINDOWS «W.V.C.»

ΕΙΣΗΓΗΤΗΣ:
ΚΑΡΟΥΣΟΥ ΒΙΚΤΩΡΙΑ

ΣΠΟΥΔΑΣΤΕΣ:
ΑΛΕΞΑΝΔΡΙΔΗΣ ΓΕΩΡΓΙΟΣ
ΒΑΛΙΔΑΣ ΜΙΧΑΛΗΣ
ΖΟΡΜΠΑΣ ΒΑΣΙΛΕΙΟΣ

ΠΑΤΡΑ 1995

W.V.C.

Πρόγραμμα για VIDEO CLUB μέσω windows

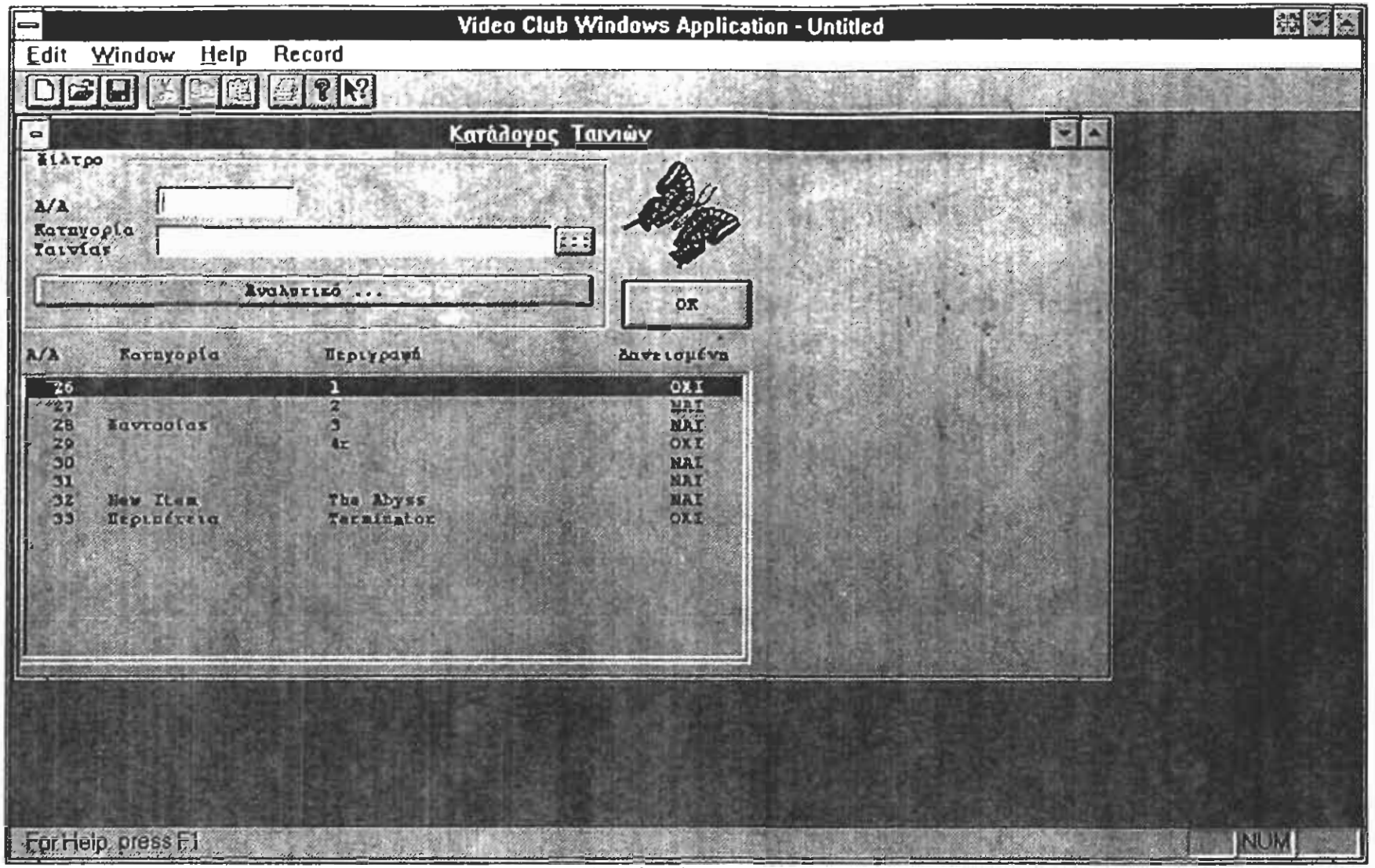
Πρόκειται για ένα πρόγραμμα το οποίο είναι ειδικά διαμορφωμένο για να δουλεύεται μέσα από τα windows ενός P.C.

Η επιλογή δημιουργίας του προγράμματος σε windows έγινε διότι τα διάφορα δεδομένα, αποτελέσματα και λοιπά στοιχεία, μπορούν να παρουσιαστούν στην οθόνη παράλληλα σε μορφή παραθύρων έστω και αν είναι ανόμοια.

Αυτή η ιδιότητα των windows είναι άνετα εφαρμόσιμη σε πρόγραμμα όπου απαιτείται άμεση απεικόνιση στοιχείων σύγκριση και παραλληλισμός, ένταξη νέων δεδομένων κ.τ.λ., ώστε να είναι δυνατή μία πρακτική και ρεαλιστική καταγραφή όπου ο χρήστης του p.c. να νιώθει πως το πρόγραμμα είναι πράγματι ένα χρήσιμο εργαλείο για την περάτωση των εργασιών του.

Ένα ακόμη πλεονέκτημα είναι το γεγονός ότι τα windows είναι ενσωματωμένα στους περισσότερους p.c., κάτι το οποίο διευκολύνει το χρήστη, διότι δεν είναι αναγκασμένος στην (πολυέξοδη πολλές φορές) αγορά ενός ξεχωριστού προγράμματος για να μπορέσει να οργανώσει τη δουλειά του.

Δεν πρέπει να παραλείψουμε επίσης πως τα windows είναι πρόγραμμα, που όπως διαφαίνεται από τις εξελίξεις στο χώρο της πληροφορικής, θα είναι σε εφαρμογή και στο απώτερο μέλλον, και πως θα εξελίσσεται χρόνο με το χρόνο καταστώντας δυναμικές και σύγχρονες τις εφαρμογές (utillites) που θα προκύπτουν από την


Video Club Windows Application - Untitled
Edit Window Help Record
Κατάλογος Ταινιών
Φίλτρο
Α/Α
Κατηγορία Ταινίας
Αναλυτικό ...
OK
Α/Α Κατηγορία Περιγραφή Διαθεσιμότητα
26 1 ΟΧΙ
27 2 ΝΑΙ
28 Καντσόλας 3 ΝΑΙ
29 4r ΟΧΙ
30 ΝΑΙ
31 ΝΑΙ
32 New Item The Abyss ΝΑΙ
33 Περιπέτεια Terminator ΟΧΙ
For Help, press F1
NUM

ΣΧΗΜΑ 1

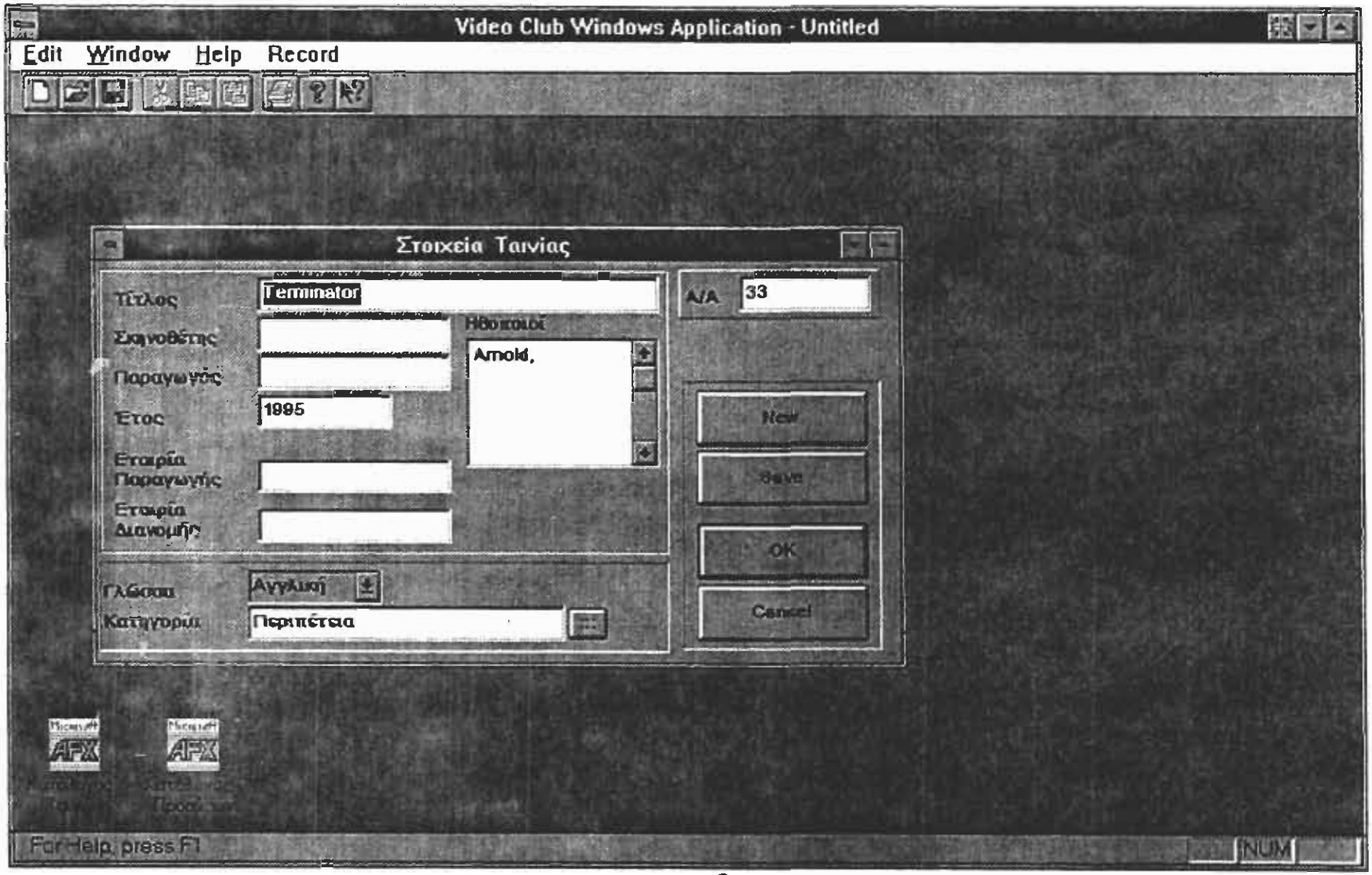
χρησιμοποίησή του. Τα windows έχουν το ιδίωμα να είναι εύχρηστα από οποιονδήποτε χρήστη p.c. λόγω της απλότητας και της εύκολης εμπέδωσης και μάθησης, εκμηδενίζοντας έτσι τα αισθήματα αποξένωσης που δημιουργούν τα πολύπλοκα προγράμματα στους χρήστες p.c., οι οποίοι δεν έχουν το εύρος γνώσεων για να ξεπεράσουν τα προβλήματα εφαρμογής δύσχρηστων προγραμμάτων.

Βάσει αυτού του γενικού σκεπτικού και σταθμίζοντας τα δεδομένα του αναλύσαμε πιό πάνω θεωρήσαμε πως η δημιουργία ενός προγράμματος για VIDEO CLUB θα ήταν πιο σωστό και λειτουργικό να χρησιμοποιείται μέσω των windows.

Βασική διάρθρωση του W.V.C.

Το πρόγραμμα, βασισμένο στην αρχή της απλότητας των windows, αποτελείται από τέσσερα «παράθυρα», που αντιπροσωπεύουν τις τέσσερις κύριες κατηγορίες του προγράμματος. Το πρώτο παράθυρο αναφέρεται στην κατηγορία «κατάλογος ταινιών» (σχήμα 1). Όπως και στα υπόλοιπα 3 παράθυρα έτσι και σ΄αυτό, έχουμε στο επάνω μέρος ένα «φίλτρο», το οποίο διερευνά και εμφανίζει στην οθόνη (στο κάτω κωδικό της είτε την κατηγορία στην οποία ανήκει η ζητούμενη ταινία.

Το κάτω μέρος εμφανίζει τα αποτελέσματα της διερεύνησης του φίλτρου. Εάν στην θέση του αύξοντα αριθμού του καταλόγου ταινιών τοποθετήσουμε τον αριθμό μηδέν, αυτόματα παρουσιάζονται όλες οι ταινίες της κατηγορίας που έχουμε ήδη επιλέξει. Επίσης στο κάτω μέρος εμφανίζονται παράλληλα οι κωδικοί των κασετών, οι κατηγορίες που ανήκουν, οι τίτλοι και αν είναι δανεισμένες ή όχι.

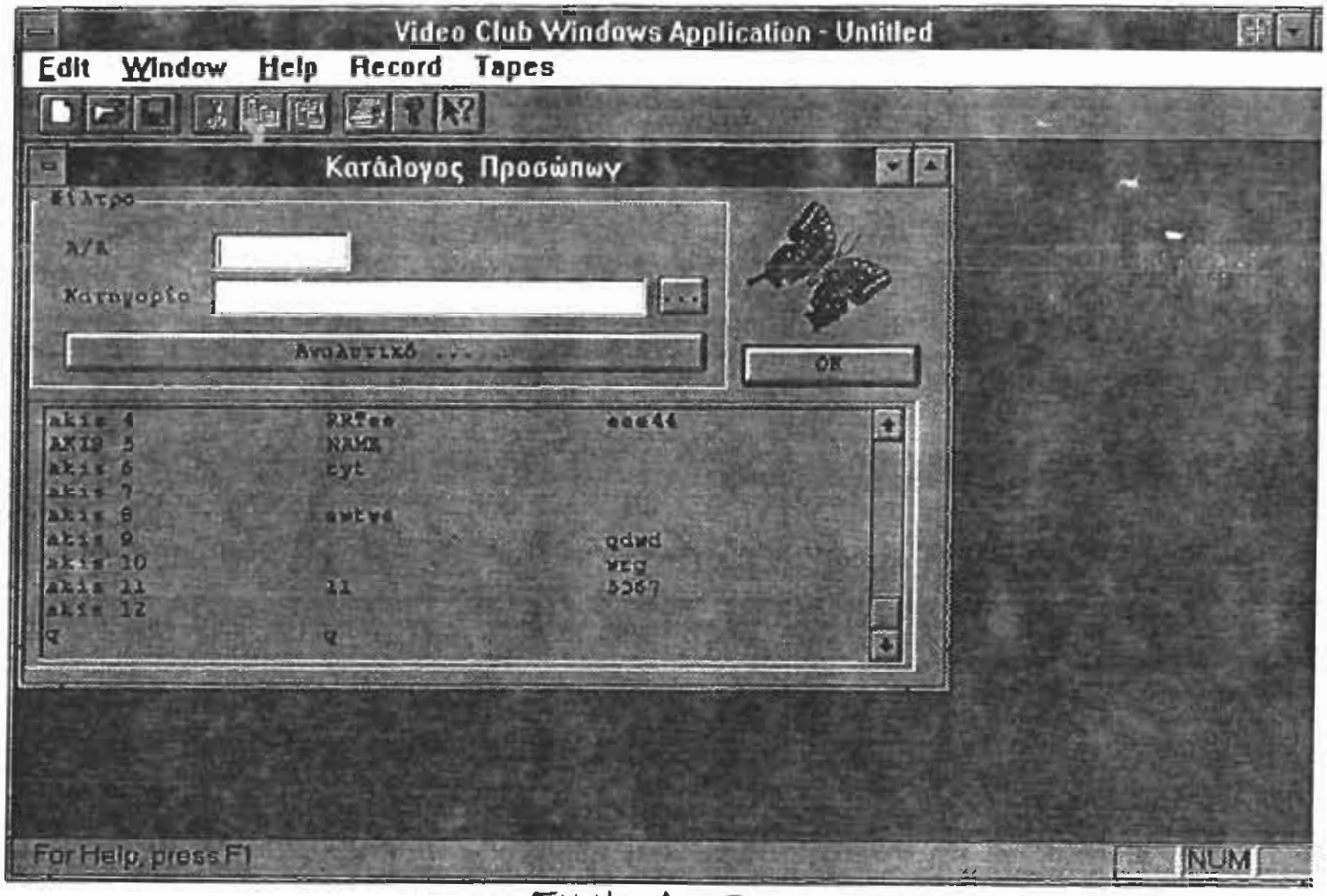

Video Club Windows Application - Untitled
Edit Window Help Record
Στοιχεία Ταινίας
Τίτλος
Terminator
Σκηνοθέτης
Παραγωγός
Έτος
1995
Εταιρία Παραγωγής
Εταιρία Διανομής
Ηθοποιοί
Arnold,
Α/Α
33
New
Save
OK
Cancel
Γλώσσα
Αγγλική
Κατηγορία
Περιπέτεια
For Help, press F1
NUM

ΣΧΗΜΑ 2

Αν φυσικά τοποθετήσουμε έναν αύξοντα αριθμό διάφορο του μηδενός εξυπακούεται πως θα εμφανιστούν τα παραπάνω στοιχεία μόνο της κασέτας που αντιστοιχεί στον πληκτρολογούμενο κωδικό.

Το επόμενο «παράθυρο» που σχετίζεται άμεσα και αλληλένδετα με το προηγούμενο είναι αυτό που αναφέρεται στα στοιχεία της εκάστοτε ταινίας (σχήμα 2). Σ΄αυτό αναγράφονται τα εξής στοιχεί: τίτλος ταινίας, σκηνοθέτης, παραγωγός, έτος παραγωγής, εταιρία παραγωγής, εταιρεία διανομής, γλώσσα, κατηγορία, ηθοποιοί και φυσικά ο κωδικός αριθμός. Και αυτό το παράθυρο περιέχει ένα «φίλτρο» το οποίο διερευνά την ζητούμενη ταινία σε περίπτωση που δεν είναι γνωστός ο κωδικός της, έχουμε τη δυνατότητα να ενεργοποιήσουμε το «φίλτρο» πληκτρολογώντας μέρος του τίτλου οπότε έχουμε παρουσίαση όλων των ταινιών που εμπεριέχουν το ίδιο μέρος τίτλου, είτε πληκτρολογώντας το όνομα του ηθοποιού, οπότε έχουμε επίσης παρουσίαση του συνόλου των ταινιών στις οποίες πρωταγωνιστεί ο προαναφερόμενος ηθοποιός.

Το τρίτο «παράθυρο» είναι ο κατάλογος προσώπων (σχήμα 3). Η λειτουργία προσομοιάζει τη λειτουργία του «παραθύρου» κατάλογος ταινιών (σχ.1). Έχουμε και εδώ ένα «φίλτρο» το οποίο διερευνά το όνομα του πελάτη και σε ποιά κατηγορία ανήκει. Αυτό επιτυγχάνεται είτε πληκτρολογώντας τον κωδικό του πελάτη, είτε τον αναζητούμε μέσω της κατηγορίας που ανήκει (νέος, παλιός, τακτικός κτλ). Μέσω αυτού του «παραθύρου» περνάμε στο τέταρτο «παράθυρο» που καλείται στοιχεία προσώπου (σχήμα 4) όπου είναι ουσιαστικά η καρτέλα πελάτου στην οποία αναφέρονται τα

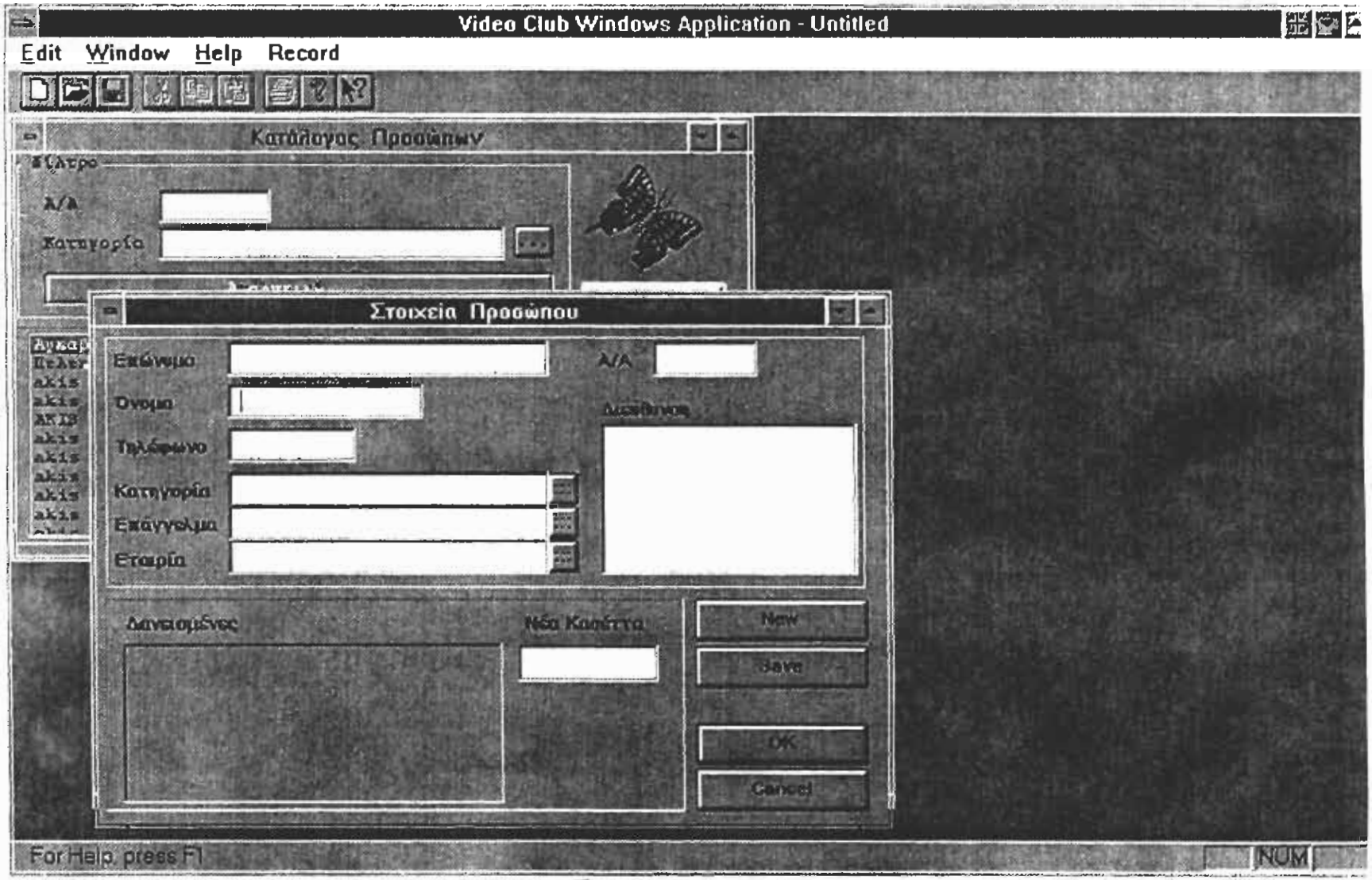

Video Club Windows Application - Untitled
Edit Window Help Record Tapes
Κατάλογος Προσώπων
Φίλτρο
Α/Α
Κατηγορία
Αναλυτικό ...
OK
For Help, press F1
NUM

ΣΧΗΜΑ 3

επιμέρους στοιχεία του πελάτη: όνομα, επώνυμο, τηλέφωνο, κατηγορία, επάγγελμα, εταιρία που εργάζεται, διεύθυνση και φυσικά ο κωδικός του.

Επίσης μία πολύ χρήσιμη πληροφορία που παρέχει το «παράθυρο» είναι οι ταινίες που έχει ενοικιάσει ο πελάτης και το χρόνο ενοικίασής του (για υπολογισμό τυχόν καθυστέρησης κτλ). Μέσω αυτού του «παραθύρου» χρεώνουμε στον πελάτη την κασέτα που έχει επιλέξει ή σβήνουμε την κασέτα που έχει επιστρέψει. Φυσικά το πρόγραμμα μέσω των «παραθύρων» προσφέρει στον χρήστη και πολλές επιπλέον υπολειτουργίες όπως: α) δυνατότητες δημιουργίας ειδικευμένων υποκατηγοριών σε μία προϋπάρχουσα κατηγορία πελάτου ή ταινίας, β) δυνατότητα αυτόματης πληροφόρησης του πελάτη για το αν έχει ξανανοικιάσει την κασέτα, γ) δυνατότητα πληροφόρησης του χρήστη για το που βρίσκεται κάποια κασέτα και για το πόσο διάστημα έχει κρατηθεί από τον πελάτη, δ) δυνατότητα επέκτασης του προγράμματος για λειτουργία αποθήκης διαφορετικού αντικειμένου από το ήδη υπάρχον (δισκοθήκη, C.D., κασέτα, P.C.C. κτλ).

Το πρόγραμμα ικανοποιεί πλήρως τις ανάγκες ενός σύγχρονου VICEO CLUB συνδυάζοντας τα χαρακτηριστικά της απλότητας και ευχρηστίας με τα αντίστοιχα χαρακτηριστικά της σύγχρονης μορφής του πλουραλισμού ικανοτήτων και εφαρμογών, της εξέλιξης και μακροβιότητας καθιστώντας το αναπόσπαστο τμήμα της οργάνωσης και λειτουργία ενός καταστήματος που επιθυμεί τα καλύτερα δυνατά αποτελέσματα για το ίδιο και την πελατεία του.


Video Club Windows Application - Untitled
Edit Window Help Record
Κατάλογος Προσώπων
Φίλτρο
Α/Α
Κατηγορία
Στοιχεία Προσώπου
Επώνυμο
Α/Α
Όνομα
Διεύθυνση
Τηλέφωνο
Κατηγορία
Επάγγελμα
Εταιρία
Δανεισμένες
Νέα Κασέττα
New
Save
OK
Cancel
For Help, press F1
NUM

ΣΧΗΜΑ 4

```
// fpersvw.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// fPersonAnalysisView form view


#ifndef __AFXEXT_H__
#include <afxext.h>
#endif

#ifndef __tform_h_OK__
#include "tform.h"
#endif

#ifndef __tformana_h_OK__
#include "tformana.h"
#endif

#include "tpsnrs.h"


#ifndef __trstree_h_OK__
#include"trstree.h"
#endif
#ifndef __rsrelpt_h_OK__
#include "rsrelpt.h"
#endif



#ifndef __s_i_hpp_OK__
#include "s_i.hpp"
#include "s_i.inl"
#endif
```

```
class fPersonAnalysisView : public fFormAnalysisView
{
   DECLARE_DYNCREATE(fPersonAnalysisView)
protected:
   fPersonAnalysisView();          // protected constructor used by dynamic creation

// Form Data
public:
s_i       m_siPIN;

Rset_RelPersonType_t     m_rsRel_PersonType;
tPersonRset      m_rsPerson;
trsTree          m_rsPersonType, m_rsInstituteType, m_rsJobType;
//CDatabase m_db;
//Overides
public:
   void  OnInitialUpdate    ();
   void  OnUpdate           (CView *pSender, LPARAM  lHint, CObject *pHint);

virtual     int          PerformRecordSave                ();
virtual     int          PerformMoveRecordsetToCurrent    ();

int       m_iFlag_FirstTime;
long      m_lPIN_CurrentPerson;


   //{{AFX_DATA(fPersonAnalysisView)
   enum { IDD = IDD_ANALYSIS_FORM };
   CListBox m_list_Types;
   CEdit m_edit_lPIN_NewType;
   //}}AFX_DATA

   tTreeEdit m_editPersonType;
   tTreeEdit m_editInstituteType;
   tTreeEdit m_editJobType;

// Attributes
public:
```

```
// fpsnana.cpp : implementation file
//

#include "stdafx.h"

#include "zdoc.h"

#ifndef __tform_h_OK__
#include "tform.h"
#endif

#ifndef __tformana_h_OK__
#include "tformana.h"
#endif
#include "resource.h"

#include "fpsnana.h"
#include "zdcmnt.h"


#ifndef __cpp2bas_h_OK__
#include "cpp2bas.h"
#endif


#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// fPersonAnalysisView

IMPLEMENT_DYNCREATE(fPersonAnalysisView, tFormAnalysisView)

fPersonAnalysisView::fPersonAnalysisView()
    : tFormAnalysisView(fPersonAnalysisView::IDD),
      m_rsPerson("File_Person"),
      m_rsRel_PersonTape     ("Rel_PersonTape"),
      m_rsPersonType      ("Tree_Person"),
      m_rsInstituteType   ("Tree_Institute"),
      m_rsJobType         ("Tree_Job"),
      m_editPersonType(&m_rsPersonType, 1), m_editInstituteType(&m_rsInstituteType, 1), m_editJobType(&m_rsJobType, 1
,
      m_siPIN()

    //{{AFX_DATA_INIT(fPersonAnalysisView)
      // NOTE: the ClassWizard will add member initialization here
    //}}AFX_DATA_INIT

m_pset            =
spRs.app (&m_rsPerson);


_fNewRecordSaved = 0;
_iFlag_FirstTime=1;

_lPIN_CurrentPerson = 0;




oid fPersonAnalysisView::DoDataExchange(CDataExchange* pDX){
////////////////////////////////////////////////////////////////////////////
  tFormAnalysisView::DoDataExchange(pDX);
////////////////////////////////////////////////////////////////////////////
String  str;
char    azs[256];
////////////////////////////////////////////////////////////////////////////
  //{{AFX_DATA_MAP(fPersonAnalysisView)
  //}}AFX_DATA_MAP

f (pDX->m_bSaveAndValidate==FALSE && m_iFlag_FirstTime) {
  DDX_Control(pDX, IDC_LIST_Tapes, m_list_Tapes);
  DDX_Control(pDX, IDC_EDIT_lPIN_NewTape, m_edit_lPIN_NewTape);

  m_iFlag_FirstTime 0;
  }

f (pDX->m_bSaveAndValidate==FALSE && !m_iFlag_FirstTime) {
  DDX_tFieldText(pDX, IDC_EDIT_lPIN,  m_rsPerson.m_uiId,  &m_rsPerson);

  m_list_Tapes.ResetContent();
  m_siPIN.erase_all();

  if (!m_fEditingNewRecord) {
     m_rsRel_PersonTape.m_strFilter.Empty();   m_rsRel_PersonTape.m_strSort = "dt_Rental";
     MakeFilter ("lPIN_Person",  m_rsRel_PersonTape.m_strFilter, m_rsPerson.m_uiId);
     MakeFilter ("iFlag_Return", m_rsRel_PersonTape.m_strFilter, 0, TRUE);
```

```
      m_rsRel_PersonTape.Requery();
      while (m_rsRel_PersonTape.IsValid()) {
         if (m_rsRel_PersonTape.m_dt_Rental.year>1900)  sprintf(azs, "%5ld\t%02d/%02d/%02d %02d:%02d", m_rsRel_Perso
nTape.m_lPIN_Tape, m_rsRel_PersonTape.m_dt_Rental.day, m_rsRel_PersonTape.m_dt_Rental.month, m_rsRel_PersonTape.m_dt
_Rental.year-1900,

                  m_rsRel_PersonTape.m_dt_Rental.hour, m_rsRel_PersonTape.m_dt_Rental.minute);
         else                                          sprintf(azs, "%5ld\t%8s %5s ", m_rsRel_PersonTape.m_lPIN_Tap
e, "", "");
         m_list_Tapes.AddString(azs);
         m_siPIN.add((int)m_rsRel_PersonTape.m_lPIN_Tape);
         if (!m_rsRel_PersonTape.GetNext()) break;
         }
      m_rsRel_PersonTape.Close();
      }


   }


   DDX_tControl(pDX, IDC_EDIT_PERSONTYPE,    m_rsPerson.m_ulIdPersonType,     m_editPersonType,     this, IDC_BUTTON_
PERSONTYPE);
   DDX_tControl(pDX, IDC_EDIT_INSTITUTETYPE, m_rsPerson.m_ulIdInstituteType,  m_editInstituteType, this, IDC_BUTTON_
INSTITUTETYPE);
   DDX_tControl(pDX, IDC_EDIT_JOBTYPE,       m_rsPerson.m_ulIdJobType,        m_editJobType,        this, IDC_BUTTON_
JOBTYPE);

   DDX_tFieldText(pDX, IDC_EDIT_ADDRESS,     m_rsPerson.m_strAddress,       &m_rsPerson);
//   DDX_tFieldText(pDX, IDC_EDIT_COMMENT,     m_rsPerson.m_strComment,       &m_rsPerson);
   DDX_tFieldText(pDX, IDC_EDIT_FNAME,       m_rsPerson.m_strFName,         &m_rsPerson);
   DDX_tFieldText(pDX, IDC_EDIT_LNAME,       m_rsPerson.m_strLName,         &m_rsPerson);
   DDX_tFieldText(pDX, IDC_EDIT_TELEPHONE,   m_rsPerson.m_strTelephone,     &m_rsPerson);

//   DDX_tFieldText(pDX, IDC_EDIT_FILENUMBER,  m_rsPerson.m_ulIdFileNumber,  &m_rsPerson);


if (pDX->m_bSaveAndValidate == TRUE) {
      //MessageBeep(MB_ICONEXCLAMATION);  AfxMessageBox("");
      //pDX->Fail();
   }

}



fPersonAnalysisView::~fPersonAnalysisView(){
////////////////////////////////////////////////////////////////////////////
}



BEGIN_MESSAGE_MAP(fPersonAnalysisView, tFormAnalysisView)
   //{{AFX_MSG_MAP(fPersonAnalysisView)
   ON_COMMAND(ID_RECORD_SAVE,     OnRecordSave)
   ON_BN_CLICKED(IDC_BUTTON_OK,     OnButtonOk)
   ON_BN_CLICKED(IDC_BUTTON_CANCEL, OnButtonCancel)
   ON_BN_CLICKED(IDC_BUTTON_SAVE,   OnButtonSave)
   ON_BN_CLICKED(IDC_BUTTON_NEW,    OnButtonNew)
   ON_LBN_DBLCLK(IDC_LIST_Tapes, OnDblclkListTapes)
   //}}AFX_MSG_MAP
END_MESSAGE_MAP()


////////////////////////////////////////////////////////////////////////////
// fPersonAnalysisView message handlers




void fPersonAnalysisView::OnInitialUpdate(){
////////////////////////////////////////////////////////////////////////////////////////
char          azs[50];
zdcmnt       *pDoc = (zdcmnt *)GetDocument();
////////////////////////////////////////////////////////////////////////////////////////
int   iMapMode;
CSize sizeArea, sizePage, sizeLine;
GetDeviceScrollSizes(iMapMode, sizeArea, sizePage, sizeLine);
m_sizeDefault = CSize(sizeArea)+CSize(15,30);
int   iEveryUnits=80;
m_list_Tapes.SetTabStops(iEveryUnits);
////////////////////////////////////////////////////////////////////////////////////////
m_lPIN_CurrentPerson    = g_lPIN_CurrentPerson;
////////////////////////////////////////////////////////////////////////////////////////
tFormAnalysisView::OnInitialUpdate();
}
```

```
void  fPersonAnalysisView::OnUpdate(CView *pSender, LPARAM  lHint, CObject *pHint) {
//////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////////
tFormAnalysisView::OnUpdate(pSender, lHint, pHint);
//////////////////////////////////////////////////////////////////////////////////////
PerformViewUpdate (pSender, lHint, pHint);
}
```

```
void fPersonAnalysisView::OnRecordSave(){
////////////////////////////////////////////////////////////////////////////////
tFormAnalysisView::OnRecordSave();
}
```

```
void fPersonAnalysisView::OnButtonOk() {
//////////////////////////////////////////////////////////////////////////////////////
CString       str;
CTime         tm;
long          lPIN_NewTape;
char          azs[256];
//////////////////////////////////////////////////////////////////////////////////////
m_edit_lPIN_NewTape.GetWindowText(str);    m_edit_lPIN_NewTape.SetWindowText("");
lPIN_NewTape = atol(str);
//////////////////////////////////////////////////////////////////////////////////////
if (!lPIN_NewTape) {
   OnRecordSave();
   GetParent()->DestroyWindow();
   return;
   }
//////////////////////////////////////////////////////////////////////////////////////

   tm = CTime::GetCurrentTime();

   m_rsRel_PersonTape.m_strFilter.Empty();    m_rsRel_PersonTape.m_strSort = "dt_Rental";
   MakeFilter ("lPIN_Tape",     m_rsRel_PersonTape.m_strFilter, lPIN_NewTape);
   MakeFilter ("iFlag_Return", m_rsRel_PersonTape.m_strFilter, 0, TRUE);
   m_rsRel_PersonTape.Requery();
   if (m_rsRel_PersonTape.IsValid())    {
      MessageBeep(MB_ICONEXCLAMATION);
      AfxMessageBox("Áõôþ ç êáóóÝôá äåí áé ðåç ãíïÜéóíÞíç");
      m_rsRel_PersonTape.Close();
      return;
      }

   m_rsRel_PersonTape.m_strFilter.Empty();    m_rsRel_PersonTape.m_strSort = "dt_Rental";
   MakeFilter ("lPIN_Tape",       m_rsRel_PersonTape.m_strFilter, lPIN_NewTape);
   MakeFilter ("lPIN_Person",     m_rsRel_PersonTape.m_strFilter, m_lPIN_CurrentPerson);
   m_rsRel_PersonTape.Requery();
   if (m_rsRel_PersonTape.IsValid())    {
      MessageBeep(MB_ICONEXCLAMATION);
      if (AfxMessageBox("Ÿ Ðåëåûóçò Ý÷åé ÍáÌáâåÍáõóåì áõôÞí ôçí óáéíßá\nÈá ôçí óâÞóå óÚèø;", MB_YESNO)==IDNO) {
         m_rsRel_PersonTape.Close();
         return;
         }
      m_rsRel_PersonTape.Delete();
      }
   //m_rsRel_PersonTape.Open();
   m_rsRel_PersonTape.AddNew();
   m_rsRel_PersonTape.m_lPIN_Person     = m_lPIN_CurrentPerson;
   m_rsRel_PersonTape.m_lPIN_Tape       = lPIN_NewTape;
     m_rsRel_PersonTape.m_dt_Rental.year        = tm.GetYear();
     m_rsRel_PersonTape.m_dt_Rental.month       = tm.GetMonth();
     m_rsRel_PersonTape.m_dt_Rental.day         = tm.GetDay();
     m_rsRel_PersonTape.m_dt_Rental.hour        = tm.GetHour();
     m_rsRel_PersonTape.m_dt_Rental.minute      = tm.GetMinute();
     m_rsRel_PersonTape.m_dt_Rental.fraction    = 0;
   m_rsRel_PersonTape.m_iFlag_Return            = 0;

   m_rsRel_PersonTape.Update();
   m_rsRel_PersonTape.Close();
```

```
   sprintf(azs, "%5ld\t%02d/%02d/%02d %02d:%02d", lPIN_NewTape, tm.GetDay(), tm.GetMonth(), tm.GetYear()-1900, tm.G
tHour(), tm.GetMinute());
   m_list_Tapes.AddString(azs);
   m_siPIN.app((int)lPIN_NewTape);


}


void fPersonAnalysisView::OnButtonCancel() {
////////////////////////////////////////////////////////////////////////////////////////////////
GetParent()->DestroyWindow();
}



void fPersonAnalysisView::OnButtonSave() {
////////////////////////////////////////////////////////////////////////////////////////////////
OnRecordSave();
}



void fPersonAnalysisView::OnButtonNew() {
////////////////////////////////////////////////////////////////////////////////////////////////

OnRecordNew();
}
```

```
int    fPersonAnalysisView::PerformRecordSave(){
////////////////////////////////////////////////////////////////////////////
int      iFlagDataSaved;
////////////////////////////////////////////////////////////////////////////
iFlagDataSaved = fFormAnalysisView::PerformRecordSave();

if (m_fNewRecordSaved) {
   m_rsPerson.m_strFilter="";    m_rsPerson.m_strSort="uiId";
   m_rsPerson.Requery();         m_rsPerson.MoveLast();

   m_lPIN_CurrentPerson   = m_rsPerson.m_uiId;
   g_lPIN_CurrentPerson   = m_rsPerson.m_uiId;
```

```
    UpdateData(FALSE);
    m_iNewRecordSaved = 0;
    }

return iFlagDataSaved;
}
```

```
int        fPersonAnalysisView::PerformMoveRecordsetToCurrent    ()   {
/////////////////////////////////////////////////////////////////////////////////
char       azs[128];
/////////////////////////////////////////////////////////////////////////////////
if (!m_lPIN_CurrentPerson)  return 0;
/////////////////////////////////////////////////////////////////////////////////
sprintf(azs, "uiId=%ld", m_lPIN_CurrentPerson);
m_rsPerson.m_strFilter=azs;           m_rsPerson.m_strSort = "";
m_rsPerson.Requery();
if (!(m_rsPerson.IsValid())) return 0;

m_lPIN_CurrentPerson    = m_rsPerson.m_uiId;
g_lPIN_CurrentPerson    = m_rsPerson.m_uiId;

return m_lPIN_CurrentPerson;
}
```

```
void fPersonAnalysisView::OnDblclkListTapes() {
/////////////////////////////////////////////////////////////////////////////////
int          iPos, iPIN;
/////////////////////////////////////////////////////////////////////////////////////////////
if ((iPos = m_list_Tapes.GetCurSel()) == LB_ERR)   return;
/////////////////////////////////////////////////////////////////////////////////////////////
m_siPIN.get_item(++iPos, iPIN);

m_rsRel_PersonTape.m_strFilter.Empty();   m_rsRel_PersonTape.m_strSort = "dt_Rental";

MakeFilter ("lPIN_Tape",     m_rsRel_PersonTape.m_strFilter, (long) iPIN);
MakeFilter ("iFlag_Return", m_rsRel_PersonTape.m_strFilter, 0, TRUE);
m_rsRel_PersonTape.Requery();
if (!m_rsRel_PersonTape.IsValid())   {
   MessageBeep(MB_ICONEXCLAMATION);
   AfxMessageBox("Áõôþ ç ôáéíßá äåí åßíáé äáíåéóìÝíç.");
   m_rsRel_PersonTape.Close();
   return;
   }
m_rsRel_PersonTape.Edit();
m_rsRel_PersonTape.m_iFlag_Return = 1;
m_rsRel_PersonTape.Update();
m_rsRel_PersonTape.Close();

PerformViewUpdate (NULL, 0, NULL);
MessageBeep(MB_ICONEXCLAMATION);
}
```

```
// fpersvw.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// fPersonSelectView form view


#ifndef __AFXEXT_H__
#include <afxext.h>
#endif

#ifndef __tform_h_OK__
#include "tform.h"
#endif

#ifndef __tformsel_h_OK__
#include "tformsel.h"
#endif

#include "tpsnrs.h"
#ifndef __w_disbtn_hpp_OK__
#include "w_disbtn.hpp"
#endif

#ifndef __a_pvoid_hpp_OK__
#include "a_pvoid.hpp"
#include "a_pvoid.inl"
#endif


#ifndef __trstree_h_OK__
#include"trstree.h"
#endif


class fPersonSelectView : public tFormSelectionView
{
   DECLARE_DYNCREATE(fPersonSelectView)
protected:
   fPersonSelectView();           // protected constructor used by dynamic creation

// Form Data
public:
tPersonRset     m_rsPerson;
trsTree         m_rsPersonType;
trsTree         m_rsInstituteType;
trsTree         m_rsJobType;
tTreeEdit       m_tredPersonType;

long    m_lPIN_CurrentPerson;

   void  OnInitialUpdate();
virtual  int       ConvertCurrentData          (CString& str);
virtual  void      PerformSearch               ();
virtual  void      OpenAnalysisView            ();   // Opens associated view for the current record

   //{{AFX_DATA(fPersonSelectView)
   enum { IDD = IDD_SELECTION_FORM };
      // NOTE: the ClassWizard will add data members here
   //}}AFX_DATA

// Attributes
public:

// Operations
public:

// Implementation
protected:
   virtual ~fPersonSelectView();
   virtual void DoDataExchange(CDataExchange* pDX);   // DDX/DDV support
   // Generated message map functions
   //{{AFX_MSG(fPersonSelectView)
   afx_msg void OnTimer(UINT nIDEvent);
   afx_msg void OnButton1();
   afx_msg void OnButtonFilter();
   afx_msg void OnViewFormTapes();
   //}}AFX_MSG
   DECLARE_MESSAGE_MAP()


w_disbtn    m_dbtn;
int         m_fTimes;
WORD        m_wTimer;
int         m_iFlag_FirstTime;
};

/////////////////////////////////////////////////////////////////////////////
```

```
// fpsnvw.cpp : implementation file
//

#include "stdafx.h"
#include "zdoc.h"

#ifndef __tform_h_OK__
#include "tform.h"
#endif

#include "fpsnvw.h"
#include "fpsnana.h"
#include "zdcmnt.h"

#include "dtreesel.h"



#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// fPersonSelectView

IMPLEMENT_DYNCREATE(fPersonSelectView, tFormSelectionView)

fPersonSelectView::fPersonSelectView()
   : tFormSelectionView(fPersonSelectView::IDD),
     m_rsPerson          ("File_Person"),
     m_rsPersonType      ("Tree_Person"),
     m_rsInstituteType   ("Tree_Institute"),
     m_rsJobType         ("Tree_Job"),
     m_tredPersonType(&m_rsPersonType, 1),
     m_fTimes(0), m_wTimer(0), m_dbtn()
{
   //{{AFX_DATA_INIT(fPersonSelectView)
      // NOTE: the ClassWizard will add member initialization here
   //}}AFX_DATA_INIT
m_prset = &m_rsPerson;
m_lPIN_CurrentPerson = 0;
m_iFlag_FirstTime =1;
}



fPersonSelectView::~fPersonSelectView(){
/////////////////////////////////////////////////////////////////////////////
m_rsPerson.Close();
}






void fPersonSelectView::DoDataExchange(CDataExchange* pDX){
/////////////////////////////////////////////////////////////////////////////
   tFormSelectionView::DoDataExchange(pDX);
   //{{AFX_DATA_MAP(fPersonSelectView)
      // NOTE: the ClassWizard will add DDX and DDV calls here
   //}}AFX_DATA_MAP
   CString str;

if (pDX->m_bSaveAndValidate==FALSE && m_iFlag_FirstTime) {
   m_iFlag_FirstTime=0;
   }

if (pDX->m_bSaveAndValidate==FALSE) {
   m_lPIN_CurrentPerson = 0;
   }

   DDX_Text(pDX, IDC_EDIT_lPIN,    str);
   int iDump=0;
   DDX_tControl(pDX, IDC_EDIT_PERSONTYPE,     iDump,     m_tredPersonType,     this, IDC_BUTTON_PERSONTYPE);

if (pDX->m_bSaveAndValidate==TRUE ) {
   m_lPIN_CurrentPerson = atol(str);
   }
}




BEGIN_MESSAGE_MAP(fPersonSelectView, tFormSelectionView)
   //{{AFX_MSG_MAP(fPersonSelectView)
   ON_WM_TIMER()
```

```
    ON_BN_CLICKED(IDC_BUTTON_FILTER, OnButtonFilter)
    ON_COMMAND(ID_VIEWFORM_TAPES,        OnViewFormTapes)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////////
// fPersonSelectView message handlers




void fPersonSelectView::OnInitialUpdate(){
////////////////////////////////////////////////////////////////////////////////////////
zdcmnt      *pDoc = (zdcmnt *)GetDocument();
////////////////////////////////////////////////////////////////////////////
m_dbtn.SubclassDlgItem(IDC_BTN_MOVIE, this);
////////////////////////////////////////////////////////////////////////////
g_pDB->SetSynchronousMode(TRUE);
////////////////////////////////////////////////////////////////////////////
int   iMapMode;
CSize sizeArea, sizePage, sizeLine;
GetDeviceScrollSizes(iMapMode, sizeArea, sizePage, sizeLine);
m_sizeDefault = CSize(sizeArea)+CSize(15,30);
////////////////////////////////////////////////////////////////////////////
tFormSelectionView::OnInitialUpdate();



}




void fPersonSelectView::PerformSearch() {
////////////////////////////////////////////////////////////////////////////////////
a_pvoid         *paChilds;      paChilds = new a_pvoid(20);
char            azs[64];
int             ic=0;
////////////////////////////////////////////////////////////////////////////////////
m_fTimes = 0;       m_wTimer = SetTimer(1, 200, NULL);  //butterfly

m_rsPerson.m_strFilter = "";
m_rsPerson.m_strSort ="";

UpdateData(TRUE);
if (m_lPIN_CurrentPerson) {
    MakeFilter ( "ulId", m_rsPerson.m_strFilter,  m_lPIN_CurrentPerson);
    }
else {
    MakeFilter("ulIdPersonType",      m_rsPerson.m_strFilter,  &m_rsPersonType, m_treedPersonType.m_lItemId);
    }
m_rsPerson.Requery();    FillListBox();
m_rsPerson.Close();
UpdateData(FALSE);
}




void fPersonSelectView::OnTimer(UINT nIDEvent){
////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////
if (m_fTimes%2==0) {
    m_dbtn.InitObject ("bfly1.bmp");
    }
else {
    m_dbtn.InitObject ("bfly2.bmp");
    }
m_dbtn.Invalidate(FALSE);

if (m_fTimes++ != 4) return;
if (m_wTimer) {
    KillTimer(m_wTimer);m_wTimer= 0;
    }
}




int   fPersonSelectView::ConvertCurrentData(CString& str) {
```

```
//////////////////////////////////////////////////////////////////////////////
str  = m_rsPerson.m_strLName + "\t" + m_rsPerson.m_strFName + "\t" + m_rsPerson.m_strTelephone;

return 1;
}




void fPersonSelectView::OpenAnalysisView ()  {
//////////////////////////////////////////////////////////////////////////////////////////////
zdcmnt      *pDoc = (zdcmnt *)GetDocument();
int          i;
WORD         w, wInitPos;
//////////////////////////////////////////////////////////////////////////////////////////////
if ((i = m_lb.GetCurSel()) == LB_ERR)   return;
wInitPos = m_prset->m_swPool.get_pos();
w = m_prset->GetTheId(i);
m_prset->m_swPool.set_pos(wInitPos);
//////////////////////////////////////////////////////////////////////////////////////////////
g_pExternalRset = NULL;
//g_CurrentPersonId = m_rsPerson.m_uiId;
g_lPIN_CurrentPerson = w;

CMDIChildWnd *pNewFrame
   = (CMDIChildWnd *)(g_pTemplateEditForm_Person->CreateNewFrame(pDoc, NULL));
g_pTemplateEditForm_Person->InitialUpdateFrame(pNewFrame,pDoc);

}
```

```
#include "frpsn.h"
void fPersonSelectView::OnButtonFilter() {
//////////////////////////////////////////////////////////////////////////////////////////////
int   i;
//////////////////////////////////////////////////////////////////////////////////////////////
m_rsPerson.Open();
//m_rsPerson.InitialiseRecordDataValues ();
m_rsPerson.AddNew();  // << <<
g_pExternalRset = &m_rsPerson;
dPersonFilter  dlg;
i = dlg.DoModal();
if (i!=IDOK) {
   m_rsPerson.Close();
   return;
   }
//////////////////////////////////////////////////////////////////////////////
m_tredPersonType.m_lItemId =0;m_tredPersonType.m_iFlagFullPath=0;
m_tredPersonType.SetWindowText("Detailed filter active ...");
//////////////////////////////////////////////////////////////////////////////
a_pvoid        *paChilds;     paChilds   new a_pvoid[20];
char           azs[64];
int            ic=0;
//////////////////////////////////////////////////////////////////////////////
m_fTimes = 0;       m_wTimer   SetTimer(1, 200, NULL);  //butterfly

m_rsPerson.m_strFilter="";
m_rsPerson.m_strSort="";


MakeFilter("uiIdPersonType",      m_rsPerson.m_strFilter,  &m_rsPersonType,       m_rsPerson.m_uiIdPersonType     );
MakeFilter("uiIdInstituteType",   m_rsPerson.m_strFilter,  &m_rsInstituteType,    m_rsPerson.m_uiIdInstituteType );
MakeFilter("uiIdJobType",         m_rsPerson.m_strFilter,  &m_rsJobType,          m_rsPerson.m_uiIdJobType    );
MakeFilter("strAddress",          m_rsPerson.m_strFilter,  m_rsPerson.m_strAddress);
MakeFilter("strFName",            m_rsPerson.m_strFilter,  m_rsPerson.m_strFName);
MakeFilter("strLName",            m_rsPerson.m_strFilter,  m_rsPerson.m_strLName);
MakeFilter("strTelephone",        m_rsPerson.m_strFilter,  m_rsPerson.m_strTelephone);
//MakeFilter("uiIdFileNumber",      m_rsPerson.m_strFilter,  m_rsPerson.m_uiIdFileNumber);
//MakeFilter("strComment",          m_rsPerson.m_strFilter,  m_rsPerson.m_strComment);

m_rsPerson.Move(AFX_MOVE_REFRESH);  // << << <<
```

```
m_rsPerson.Requery();   m_lb.ResetContent(); /////// ..... .....
FillListBox();
////////////////////////////////////////////////////////////////////////////////
m_rsPerson.Close();
}
```

```
void fPersonSelectView::OnViewFormTapes ()  {
////////////////////////////////////////////////////////////////////////////////
zdcmnt      *pDoc = (zdcmnt *)GetDocument();
int          i;
WORD         w, wInitPos;
////////////////////////////////////////////////////////////////////////////////
if ((i = m_lb.GetCurSel()) == LB_ERR)   return;
wInitPos = m_pset->m_swPool.get_pos();
w = m_pset->GetTheId(i);
m_pset->m_swPool.set_pos(wInitPos);
////////////////////////////////////////////////////////////////////////////////
g_pExternalRset = NULL;
//g_CurrentPersonId = m_rsPerson.m_uiId;
g_lPIN_CurrentPerson = 0;

CMDIChildWnd *pNewFrame
   = (CMDIChildWnd *)( g_pTemplateViewForm_Tape->CreateNewFrame(pDoc, NULL));
g_pTemplateViewForm_Tape->InitialUpdateFrame(pNewFrame,pDoc);

}
```

```
#ifndef __efsoldr_h_OK__
#define __efsoldr_h_OK__
//
// fpsnvw.h : header file
//
/////////////////////////////////////////////////////////////////////////////
// EditForm_Tape_t form view


#ifndef __AFXEXT_H__
#include "afxext.h"
#endif

#ifndef __tform_h_OK__
#include "tform.h"
#endif

#ifndef __tformana_h_OK__
#include "tformana.h"
#endif

#ifndef __rssoldr_h_OK__
#include "rssoldr.h"
#endif
#ifndef __trstree_h_OK__
#include"trstree.h"
#endif


class EditForm_Tape_t : public tFormAnalysisView
{
   DECLARE_DYNCREATE(EditForm_Tape_t)
protected:
   EditForm_Tape_t();          // protected constructor used by dynamic creation

// Form Data
public:
//CDatabase m_db;
//Overides
public:
   void  OnInitialUpdate    ();
   void  OnUpdate           (CView *pSender, LPARAM  lHint, CObject *pHint);

virtual     int          PerformRecordSave                ();
virtual     int          PerformMoveRecordsetToCurrent    ();


   //{{AFX_DATA(EditForm_Tape_t)
   enum { IDD = IDD_EDITFORM_TAPE };
   //}}AFX_DATA

/////////////////////////////////////////////////////////////////////////////
long                        m_lPIN_CurrentTape;
/////////////////////////////////////////////////////////////////////////////
Rset_FileTape_t             m_rsFile_Tape;
trsTree                     m_rsTree_FilmType;
RsetList_t                  m_rsList_Language;
/////////////////////////////////////////////////////////////////////////////
CEdit          m_edit_lPIN;
tTreeEdit      m_tred_l_FilmType;
tComboBox      m_combo_l_Language;

CEdit          m_edit_str_Title;
CEdit          m_edit_str_Director;
CEdit          m_edit_str_Producer;
CEdit          m_edit_str_Actors;
CEdit          m_edit_str_Company;
CEdit          m_edit_str_DistributionCompany;

CEdit          m_edit_i_Year;
/////////////////////////////////////////////////////////////////////////////
int   m_iFlagFirstTime;
/////////////////////////////////////////////////////////////////////////////
// Attributes
public:

// Operations
public:

// Implementation
protected:
   virtual ~EditForm_Tape_t();
   virtual void DoDataExchange(CDataExchange* pDX);   // DDX/DDV support
   // Generated message map functions
   //{{AFX_MSG(EditForm_Tape_t)
   afx_msg void OnRecordSave();
   afx_msg void OnRecordNew();
   afx_msg void OnButtonTapeCancel();
   afx_msg void OnButtonTapeNew();
   afx_msg void OnButtonTapeOk();
   afx_msg void OnButtonTapeSave();
```

```
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
```

```
// fpsnana.cpp : implementation file
//

#include "stdafx.h"

#include "zdoc.h"

#ifndef __tform_h_OK__
#include "tform.h"
#endif

#ifndef __tformana_h_OK__
#include "tformana.h"
#endif
#include "resource.h"

#include "efsoldr.h"
#include "zdcmnt.h"

#ifndef __cpp2bas_h_OK__
#include "cpp2bas.h"
#endif
/////////////////////////////////////////////////
///
///
#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

#ifndef __physical_h_OK__
#include "physical.h"
#endif



/////////////////////////////////////////////////////////////////////////////
// EditForm_Tape_t

IMPLEMENT_DYNCREATE(EditForm_Tape_t, tFormAnalysisView)

EditForm_Tape_t::EditForm_Tape_t()
    : tFormAnalysisView(EditForm_Tape_t::IDD),
      m_rsFile_Tape          {"File_Tape"},
      m_rsTree_FilmType      {"Tree_FilmType"},
      m_rsList_Language      {"List_Language"},

      m_tred_l_FilmType(&m_rsTree_FilmType, 1),
      m_combo_l_Language(&m_rsList_Language)
{
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
    //{{AFX_DATA_INIT(EditForm_Tape_t)
       // NOTE: the ClassWizard will add member initialization here
    //}}AFX_DATA_INIT

m_sprs.app (&m_rsFile_Tape);
//
m_fNewRecordSaved = 0;
m_iFlagFirstTime=1;

m_lPIN_CurrentTape   0;
}



void EditForm_Tape_t::DoDataExchange(CDataExchange* pDX){
/////////////////////////////////////////////////////////////////////////////
    tFormAnalysisView::DoDataExchange(pDX);
/////////////////////////////////////////////////////////////////////////////
    //{{AFX_DATA_MAP(EditForm_Tape_t)
    //}}AFX_DATA_MAP
    //DDX_Control(pDX, IDC_EDIT_PATIENTFULLNAME,    m_editPatientName);
if (pDX->m_bSaveAndValidate==FALSE && !m_iFlagFirstTime) {
    DDX_tFieldText (pDX, IDC_EDIT_lPIN,      m_rsFile_Tape.m_lPIN,         &m_rsFile_Tape);
    }


    DDX_tControl(pDX, IDC_EDIT_l_FilmType,          m_rsFile_Tape.m_l_FilmType,          m_tred_l_FilmType,
    this, IDC_BUTTON_l_FilmType);

    DDX_tControl(pDX, IDC_COMBO_l_Language,         m_rsFile_Tape.m_l_Language,          m_combo_l_Language,        th
is);

    DDX_tFieldText (pDX, IDC_EDIT_i_Year,       m_rsFile_Tape.m_i_Year,         &m_rsFile_Tape);

    DDX_tFieldText (pDX, IDC_EDIT_str_Title,      m_rsFile_Tape.m_str_Title,         &m_rsFile_Tape);
    DDV_MaxChars    (pDX, m_rsFile_Tape.m_str_Title, 50);
    DDX_tFieldText (pDX, IDC_EDIT_str_Director,      m_rsFile_Tape.m_str_Director,         &m_rsFile_Tape);
    DDV_MaxChars    (pDX, m_rsFile_Tape.m_str_Director, 30);
```

```
    DDX_tFieldText (pDX, IDC_EDIT_str_Producer,     m_rsFile_Tape.m_str_Producer,        &m_rsFile_Tape);
    DDV_MaxChars   (pDX, m_rsFile_Tape.m_str_Producer, 30);
    DDX_tFieldText (pDX, IDC_EDIT_str_Actors,     m_rsFile_Tape.m_str_Actors,        &m_rsFile_Tape);
    DDV_MaxChars   (pDX, m_rsFile_Tape.m_str_Actors, 255);
    DDX_tFieldText (pDX, IDC_EDIT_str_Company,     m_rsFile_Tape.m_str_Company,        &m_rsFile_Tape);
    DDV_MaxChars   (pDX, m_rsFile_Tape.m_str_Company, 50);
    DDX_tFieldText (pDX, IDC_EDIT_str_DistributionCompany,     m_rsFile_Tape.m_str_DistributionCompany,        &m_rsF
ile_Tape);
    DDV_MaxChars   (pDX, m_rsFile_Tape.m_str_DistributionCompany, 50);

    //CString str;
    //DDX_tFieldDateTime  (pDX, IDC_EDIT_dtEntry,     str,    (TimeStamp_t&) m_rsFile_Tape.m_dtEntry,      &m_rsFile_Ta
pe, 0);
    //DDX_tFieldDateTime  (pDX, IDC_EDIT_dtExit,      str,    (TimeStamp_t&) m_rsFile_Tape.m_dtExit,       &m_rsFile_Ta
pe, 0);



if (pDX->m_bSaveAndValidate==FALSE && m_iFlagFirstTime) {
    m_iFlagFirstTime=0;
    }

if (pDX->m_bSaveAndValidate == TRUE) {
       //MessageBeep(MB_ICONEXCLAMATION);  AfxMessageBox("");
       //pDX->Fail();
    }


}



EditForm_Tape_t::~EditForm_Tape_t(){
////////////////////////////////////////////////////////////////////////////
}



BEGIN_MESSAGE_MAP(EditForm_Tape_t, tFormAnalysisView)
    //{{AFX_MSG_MAP(EditForm_Tape_t)
    ON_COMMAND(ID_RECORD_SAVE,          OnRecordSave)
    ON_COMMAND(ID_RECORD_NEW,           OnRecordNew)
    ON_COMMAND(ID_EDIT_UNDO,            OnEditUndo)
    ON_COMMAND(ID_RECORD_DELETE,        OnRecordDelete)
    ON_BN_CLICKED(IDC_BUTTON_TapeCANCEL, OnButtonTapeCancel)
    ON_BN_CLICKED(IDC_BUTTON_TapeNEW, OnButtonTapeNew)
    ON_BN_CLICKED(IDC_BUTTON_TapeOK, OnButtonTapeOk)
    ON_BN_CLICKED(IDC_BUTTON_TapeSAVE, OnButtonTapeSave)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
//    ON_BN_CLICKED(IDC_BTN_OK,                 OnBtnOk)
//    ON_BN_CLICKED(IDC_BTN_CANCEL,             OnBtnCancel)


////////////////////////////////////////////////////////////////////////////
// EditForm_Tape_t message handlers









void EditForm_Tape_t::OnInitialUpdate(){
////////////////////////////////////////////////////////////////////////////
char          azs[50];
zdcmnt       *pDoc = (zdcmnt *)GetDocument();
////////////////////////////////////////////////////////////////////////////
int    iMapMode;
CSize sizeArea, sizePage, sizeLine;
GetDeviceScrollSizes(iMapMode, sizeArea, sizePage, sizeLine);
m_sizeDefault = CSize(sizeArea)+CSize(15,30);
////////////////////////////////////////////////////////////////////////////
m_lPIN_CurrentTape  = g_lPIN_CurrentTape;
////////////////////////////////////////////////////////////////////////////
tFormAnalysisView::OnInitialUpdate();
}
```

```
void  EditForm_Tape_t::OnUpdate(CView *pSender, LPARAM  lHint, CObject *pHint) {
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
tFormAnalysisView::OnUpdate(pSender, lHint, pHint);
////////////////////////////////////////////////////////////////////////////////
PerformViewUpdate (pSender, lHint, pHint);
}
```

```
void EditForm_Tape_t::OnRecordNew(){
////////////////////////////////////////////////////////////////////////////////
tFormAnalysisView::OnRecordNew();
}
```

```
void EditForm_Tape_t::OnRecordSave(){
////////////////////////////////////////////////////////////////////////////////
tFormAnalysisView::OnRecordSave();
}
```

```
void EditForm_Tape_t::OnButtonTapeCancel() {
////////////////////////////////////////////////////////////////////////////////
GetParent()->DestroyWindow();
}


void EditForm_Tape_t::OnButtonTapeNew() {
////////////////////////////////////////////////////////////////////////////////
OnRecordNew();
}



void EditForm_Tape_t::OnButtonTapeOk() {
////////////////////////////////////////////////////////////////////////////////
OnRecordSave();
GetParent()->DestroyWindow();
}


void EditForm_Tape_t::OnButtonTapeSave() {
////////////////////////////////////////////////////////////////////////////////
OnRecordSave();
}









int   EditForm_Tape_t::PerformRecordSave(){
////////////////////////////////////////////////////////////////////////////////
int      iFlagDataSaved;
////////////////////////////////////////////////////////////////////////////////
iFlagDataSaved = tFormAnalysisView::PerformRecordSave();

if (m_fNewRecordSaved) {
   m_rsFile_Tape.m_strFilter="";   m_rsFile_Tape.m_strSort="lPIN";
   m_rsFile_Tape.Requery();        m_rsFile_Tape.MoveLast();

   m_lPIN_CurrentTape    = m_rsFile_Tape.m_lPIN;
   g_lPIN_CurrentTape    = m_rsFile_Tape.m_lPIN;

   UpdateData(FALSE);
   m_fNewRecordSaved    0;
   }

return iFlagDataSaved;
}
```

```
int          EditForm_Tape_t::PerformMoveRecordsetToCurrent     ()    {
////////////////////////////////////////////////////////////////////////////////
char      azs[128];
////////////////////////////////////////////////////////////////////////////////
if (!m_lPIN_CurrentTape)  return 0;
////////////////////////////////////////////////////////////////////////////////
sprintf(azs, "lPIN=%ld", m_lPIN_CurrentTape);
m_rsFile_Tape.m_strFilter=azs;          m_rsFile_Tape.m_strSort "";
m_rsFile_Tape.Requery();
if (!(m_rsFile_Tape.IsValid())) return 0;

m_lPIN_CurrentTape    = m_rsFile_Tape.m_lPIN;
g_lPIN_CurrentTape    = m_rsFile_Tape.m_lPIN;

return m_lPIN_CurrentTape;
}
```

```
#ifndef __vfsoldr_h_OK__
#define __vfsoldr_h_OK__
#endif

// fpsnvw.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// ViewForm_Tape_t form view


#ifndef __AFXEXT_H__
#include <afxext.h>
#endif

#ifndef __tform_h_OK__
#include "tform.h"
#endif

#ifndef __tformsel_h_OK__
#include "tformsel.h"
#endif

#ifndef __w_disbtn_hpp_OK__
#include "w_disbtn.hpp"
#endif

#ifndef __a_pvoid_hpp_OK__
#include "a_pvoid.hpp"
#include "a_pvoid.inl"
#endif

#ifndef __trstree_h_OK__
#include"trstree.h"
#endif

#ifndef __rssoldr_h_OK__
#include "rssoldr.h"
#endif


#ifndef __rsrelpt_h_OK__
#include "rsrelpt.h"
#endif
////////////////////////////////


class ViewForm_Tape_t     : public tFormSelectionView
{
   DECLARE_DYNCREATE(ViewForm_Tape_t)
protected:
   ViewForm_Tape_t();         // protected constructor used by dynamic creation

// Form Data
public:


long                        m_lPIN_CurrentTape;
Rset_FileTape_t          m_rsFile_Tape;


trsTree                  m_rsTree_FilmType;
RsetList_t               m_rsList_Language;
Rset_RelPersonTape_t     m_rsRel_PersonTape;

long           m_lPIN_FilmType;
tTreeEdit      m_tred_FilmType;

        void     OnInitialUpdate();

virtual  int       ConvertCurrentData          (CString& str);
virtual  void      PerformSearch               ();
virtual  void      OpenAnalysisView            ();   // Opens associated view for the current record

   //{{AFX_DATA(ViewForm_Tape_t)
   enum { IDD = IDD_VIEWFORM_TAPE };
```

```
    //}}AFX_DATA

// Attributes
public:

// Operations
public:

// Implementation
protected:
    virtual ~ViewForm_Tape_t();
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    // Generated message map functions
    //{{AFX_MSG(ViewForm_Tape_t)
    afx_msg void OnTimer(UINT nIDEvent);
    afx_msg void OnLButtonDblClk(UINT nFlags, CPoint point);
    afx_msg  void  OnRecordNew();
    afx_msg  void  OnButtonFilter();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()


W_disbtn    m_dBtn;
int         m_fTimes;
WORD        m_wTimer;

int    m_iFlagFirstTime;
};

/////////////////////////////////////////////////////////////////////////////
```

```
// fpsnvw.cpp : implementation file
//

#include "stdafx.h"
#include "zdoc.h"

#ifndef __tform_h_OK__
#include "tform.h"
#endif

#include "vfsoldr.h"


#include "zdcmnt.h"

#ifndef __dtreesel_h_OK__
#include "dtreesel.h"
#endif



#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// ViewForm_Tape_t

IMPLEMENT_DYNCREATE(ViewForm_Tape_t, tFormSelectionView)

ViewForm_Tape_t::ViewForm_Tape_t()
    : tFormSelectionView(ViewForm_Tape_t::IDD),
      m_rsFile_Tape          {"File_Tape"},
      m_rsTree_FilmType      {"Tree_FilmType"},
      m_rsList_Language      {"List_Language"},
      m_rsRel_PersonTape     {"Rel_PersonTape"},
      m_tred_FilmType(&m_rsTree_FilmType, 1),
      m_fTimes(0), m_wTimer(0), m_dbtn(){
/////////////////////////////////////////////////////////////////////////////////////////////
    //{{AFX_DATA_INIT(ViewForm_Tape_t)
       // NOTE: the ClassWizard will add member initialization here
    //}}AFX_DATA_INIT

//m_sprs.app (&
m_prset = &m_rsFile_Tape;                       // <<  <<  <<  <<
m_lPIN_CurrentTape = 0;
m_lPIN_FilmType = 0;

m_iFlagFirstTime=1;
}




ViewForm_Tape_t::~ViewForm_Tape_t(){
/////////////////////////////////////////////////////////////////////////////
}




void ViewForm_Tape_t::DoDataExchange(CDataExchange* pDX){
/////////////////////////////////////////////////////////////////////////////
    tFormSelectionView::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(ViewForm_Tape_t)
    //}}AFX_DATA_MAP

    CString str;

if (pDX->m_bSaveAndValidate==FALSE && m_iFlagFirstTime) {
    m_iFlagFirstTime=0;
    }

if (pDX->m_bSaveAndValidate==FALSE ) {
    m_lPIN_CurrentTape = 0;
    }
DDX_Text(pDX, IDC_EDIT_lPIN,    str);
DDX_tControl(pDX, IDC_EDIT_l_FilmType,     m_lPIN_FilmType,      m_tred_FilmType,     this, IDC_BUTTON_l_FilmType);

if (pDX->m_bSaveAndValidate==TRUE ) {
    m_lPIN_CurrentTape = atol(str);
    }

}
```

```
BEGIN_MESSAGE_MAP(ViewForm_Tape_t, tFormSelectionView)
    //{{AFX_MSG_MAP(ViewForm_Tape_t)
    ON_WM_TIMER()
    ON_WM_LBUTTONDBLCLK()
    ON_COMMAND(ID_RECORD_NEW,          OnRecordNew)
    ON_BN_CLICKED(IDC_BUTTON_FILTER, OnButtonFilter)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////////
// ViewForm_Tape_t message handlers

void ViewForm_Tape_t::OnTimer(UINT nIDEvent){
/////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////////
if (m_fTimes%2==0) {
    m_dbtn.InitObject("butterfly.tmp");
    }
else {
    m_dbtn.InitObject("butterfly.tmp");
    }
m_dbtn.Invalidate(FALSE);

if (m_fTimes++ != 21) return;
if (m_wTimer) {
    KillTimer(m_wTimer);m_wTimer=0;
    }
}




void ViewForm_Tape_t::OnInitialUpdate(){
/////////////////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////////////////////
m_dbtn.SubclassDlgItem(IDC_BTN_MOVIE, this);
/////////////////////////////////////////////////////////////////////////////////////////
g_pDB->SetSynchronousMode(TRUE);
/////////////////////////////////////////////////////////////////////////////////////////
int    iMapMode;
CSize sizeArea, sizePage, sizeLine;
GetDeviceScrollSizes(iMapMode, sizeArea, sizePage, sizeLine);
//m_sizeDefault = CSize(sizeArea)+CSize(15,30);
m_sizeDefault = CSize(700,360);
/////////////////////////////////////////////////////////////////////////////////////////
tFormSelectionView::OnInitialUpdate();
}




void ViewForm_Tape_t::PerformSearch() {
/////////////////////////////////////////////////////////////////////////////////////////
char           azs[64];
int            ic=0, i;
CString        *pstr, str;
/////////////////////////////////////////////////////////////////////////////////////////
m_fTimes = 0;       m_wTimer = SetTimer(1, 200, NULL);  //butterfly
m_rsTree_FilmType.ReadAllDescriptions(20);
//.ReadAllDescriptions(20);
//
//m_rsList_Bathmos.ReadAllDescriptions(5);

m_rsFile_Tape.m_strFilter="";    m_rsFile_Tape.m_strSort="";



//TrimString(m_pset->m_strFilter);
//if (m_pset->m_strFilter.IsEmpty()) {
```

```
    //AfxMessageBox ("Please give a VHS");
//    return;
//    }

UpdateData(TRUE);
if (m_lPIN_CurrentTape) {
    MakeFilter ( "lPIN", m_rsFile_Tape.m_strFilter,  m_lPIN_CurrentTape);
    }
else {
    MakeFilter("l_FilmType",      m_rsFile_Tape.m_strFilter,  &m_rsTree_FilmType, m_lPIN_FilmType);
    }


m_rsFile_Tape.Requery();
FillListBox();
m_rsFile_Tape.Close();
m_rsRel_PersonTape.Close();
//.DiscardDescriptions();
m_rsTree_FilmType.DiscardDescriptions();
//
UpdateData(FALSE);
}
```

```
int   ViewForm_Tape_1::ConvertCurrentData(CString& str) {
////////////////////////////////////////////////////////////////////////////////
char     azs[128];
CString  *pstr, strHelp;
////////////////////////////////////////////////////////////////////////////////
sprintf(azs, "%4ld ", m_rsFile_Tape.m_lPIN);
str +=  azs;

pstr= m_rsTree_FilmType.GetDescription(m_rsFile_Tape.m_l_FilmType);
if (pstr)    { sprintf(azs, "  %-20s", *pstr);      }
else         { sprintf(azs, "  %-20s", "");         }
str += azs;



sprintf(azs, "%-30s ", m_rsFile_Tape.m_str_Title.Left(30));
str += azs;


m_rsRel_PersonTape.m_strFilter.Empty();
m_rsRel_PersonTape.m_strSort.Empty();
MakeFilter ("lPIN_Tape",      m_rsRel_PersonTape.m_strFilter, m_rsFile_Tape.m_lPIN);
MakeFilter ("iFlag_Return",   m_rsRel_PersonTape.m_strFilter, 0, TRUE);
m_rsRel_PersonTape.Requery();
//if(m_rsFile_Tape.m_dtEntry.year>1900)  sprintf(azs, "%02d/%02d/%02d ", m_rsFile_Tape.m_dtEntry.day, m_rsFile_Tape.
m_dtEntry.month, m_rsFile_Tape.m_dtEntry.year-1900);
//else                                   sprintf(azs, "%8s ", "");
//str +=  azs;
if (m_rsRel_PersonTape.IsValid()) {
   strcpy (azs, "NAI");
   }
else {
   strcpy (azs, "OXI");
   }
str +=  azs;



return 1;
}
```

```
void ViewForm_Tape_t::OpenAnalysisView ()  {
////////////////////////////////////////////////////////////////////////////////////////////
zdcmnt      *pDoc = (zdcmnt *)GetDocument();
int          i;
WORD         w, wInitPos;
////////////////////////////////////////////////////////////////////////////////////////////
if ((i = m_lb.GetCurSel()) == LB_ERR)   return;
wInitPos = m_prset->m_swPool.get_pos();
w = m_prset->GetTheId(i);
m_prset->m_swPool.set_pos(wInitPos);
////////////////////////////////////////////////////////////////////////////////////////////
g_pExternalRset          = NULL;
g_lPIN_CurrentTape    = w;

//if (!CheckBiosLock()) {
//   g_lPIN_CurrentSoldier      = m_lIdPatient / w;
//   }

CMDIChildWnd *pNewFrame
   = (CMDIChildWnd *)( g_pTemplateEditForm_Tape->CreateNewFrame(pDoc, NULL));
g_pTemplateEditForm_Tape->InitialUpdateFrame(pNewFrame,pDoc);

}
```

```
void ViewForm_Tape_t::OnLButtonDblClk(UINT nFlags, CPoint point) {
////////////////////////////////////////////////////////////////////////////////////////////
tFormSelectionView::OnLButtonDblClk(nFlags, point);
}
```

```
void ViewForm_Tape_t::OnRecordNew ()  {
////////////////////////////////////////////////////////////////////////////////////////////
zdcmnt      *pDoc = (zdcmnt *)GetDocument();
////////////////////////////////////////////////////////////////////////////////////////////
g_pExternalRset          = NULL;
g_lPIN_CurrentTape       = 0;
g_iFlag_NewRecordInitialising = 1;


CMDIChildWnd *pNewFrame
   = (CMDIChildWnd *)( g_pTemplateEditForm_Tape->CreateNewFrame(pDoc, NULL));
g_pTemplateEditForm_Tape->InitialUpdateFrame(pNewFrame,pDoc);

}
```

```
#include "fdtape.h"
void ViewForm_Tape_t::OnButtonFilter() {
////////////////////////////////////////////////////////////////////////////////////////////
int   i;
////////////////////////////////////////////////////////////////////////////////////////////
m_rsFile_Tape.Open();
//m_rsPerson.InitialiseRecordDataValues ();
m_rsFile_Tape.AddNew();  // << <<
g_pExternalRset = &m_rsFile_Tape;
FilterDialog_Tape_t  dlg;
i = dlg.DoModal();
if (i!=IDOK) {
   m_rsFile_Tape.Close();
   return;
   }
////////////////////////////////////////////////////////////////////////////////
m_tred_FilmType.m_lItemId=0;m_tred_FilmType.m_iFlagFullPath 0;
m_tred_FilmType.SetWindowText("Detailed filter active ...");
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
m_iTimes = 0;       m_wTimer = SetTimer(1, 200, NULL);  //butterfly

m_rsFile_Tape.m_strFilter="";
m_rsFile_Tape.m_strSort="";

MakeFilter ("l_Language", m_rsFile_Tape.m_strFilter,    m_rsFile_Tape.m_l_Language);
MakeFilter ("l_FilmType", m_rsFile_Tape.m_strFilter,    &m_rsTree_FilmType, m_rsFile_Tape.m_l_FilmType);

MakeFilter ("i_Year",     m_rsFile_Tape.m_strFilter,    m_rsFile_Tape.m_i_Year);

MakeFilter ("str_Title",    m_rsFile_Tape.m_strFilter,    m_rsFile_Tape.m_str_Title);
MakeFilter ("str_Director",m_rsFile_Tape.m_strFilter,   m_rsFile_Tape.m_str_Director);
MakeFilter ("str_Producer",m_rsFile_Tape.m_strFilter,   m_rsFile_Tape.m_str_Producer);
MakeFilter ("str_Company", m_rsFile_Tape.m_strFilter,   m_rsFile_Tape.m_str_Company);
MakeFilter ("str_DistributionCompany",  m_rsFile_Tape.m_strFilter,    m_rsFile_Tape.m_str_DistributionCompany);
MakeFilter ("str_Actors",    m_rsFile_Tape.m_strFilter, m_rsFile_Tape.m_str_Actors);


m_rsFile_Tape.Move(AFX_MOVE_REFRESH);  // << << <<
```

```
m_rsTree_FilmType.ReadAllDescriptions(20);
m_rsFile_Tape.Requery();    m_lb.ResetContent(); ////// ..... .....
FillListBox();
////////////////////////////////////////////////////////////////////////////////
m_rsTree_FilmType.DiscardDescriptions();
m_rsFile_Tape.Close();
}
```