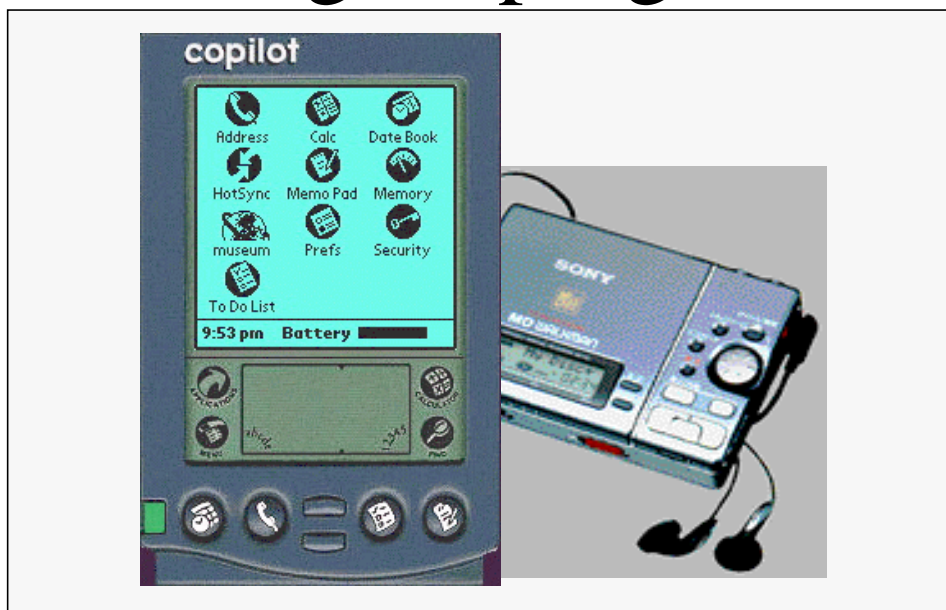


---

# Guide interactif pour visite de musées et expositions sur pen computer

## Listing du programme



**Niklaus Hirt, Informatique**

Travail de 8ème semestre

---

---

---

```

*****
*
* PROJECT: Virtual Museum for Pilot
* FILE: museum.cp
* AUTHOR: Niklaus Hirt: June 29, 1997
* DECLARER: VirtualMuseum
* DESCRIPTION:
*****
#include <Pilot.h>
#include <SerialMgr.h>

#include "muse1.h"
#include "museum.h"

#define maxTrackNumber 3 //secure number of tracks to rewind
#define buttonWidth 36 //button parameters
#define buttonHeight 12
#define buttonYpos 144
#define upButtonX 4
#define downButtonX 42
#define playButtonX 79
#define stopButtonX 115
#define XSIZE 160 //size of the drawable window
#define YSIZE 126
#define trackDelay 45 //delay for track operations

static Word StartApplication (void);
static void StopApplication (void);
static void EventLoop (void);
static Boolean ApplicationHandleEvent(EventPtr event);
static void goup();
static void godown();
static void hear(POA *actPOAptr);
static void stopTrack();
static void selected();
static void playTrack(int trackNumber);
static void gotoTrackOne();
static void selected(int x, int y);
static void drawRect(short upperX, short upperY, short extentX, short extentY);
static void drawPOA(POA *actPOA, Boolean type);
static void drawRoom(Room *actRoom);
static void drawFloor(Floor *actFloor);
static POA *getNearestPOA(Floor *actFloorPtr, int x, int y);
static int distance(int x1, int y1, int x2, int y2);
static void eraseWin();
static void stopPlay();

Boolean drawbegin=true; //draw the museum a first time

UInt SerialRef; //pointers to the serial port settings
static SerSettingsType SerSettings;

static Building *MyMuseumPtr; //pointer to the museum
POA *myPOAptr; //pointer to the selected POA

int floor =0; //actual floor number
int actTrack=1; //actual track number
int actDuration=0; //actual duration of track

Booleanplaying= false; //track playing?
ULong startTime; //starting time of track playing

/*****
* FUNCTION: MainHandleEvent
* DESCRIPTION: This routine treats the application events
* PARAMETERS: event
* RETURNED: true if there is an error
* REVISION HISTORY:
* Name Date Description
* --- --- ---
* nik 29.6.97 Initial Revision
*****/

static Boolean MainHandleEvent (EventPtr e)
{
    if (e->eType == penDownEvent) {
        int x = e->screenX;
        int y = e->screenY;
        if (x >= 10 && x <= 150 && y > 15 && y <= 140) {
            selected(x,y); // pen down SELECTION
        }
        if ((x >= upButtonX) && (x <= upButtonX+buttonWidth) && y > buttonYpos
            && y <= buttonYpos+buttonHeight) {
            goup(); // pen down UP
        }
        if (x >= downButtonX && x <= downButtonX+buttonWidth && y > buttonYpos
            && y <= buttonYpos+buttonHeight) {
            godown(); // pen down DOWN
        }
    }
}

```

```

*****
*
* PROJECT: Virtual Museum for Pilot
* FILE: museum.cp
* AUTHOR: Niklaus Hirt: June 29, 1997
* DECLARER: VirtualMuseum
* DESCRIPTION:
*****
#include <Pilot.h>
#include <SerialMgr.h>

#include "muse1.h"
#include "museum.h"

#define maxTrackNumber 3 //secure number of tracks to rewind
#define buttonWidth 36 //button parameters
#define buttonHeight 12
#define buttonYpos 144
#define upButtonX 4
#define downButtonX 42
#define playButtonX 79
#define stopButtonX 115
#define XSIZE 160 //size of the drawable window
#define YSIZE 126
#define trackDelay 45 //delay for track operations

static Word StartApplication (void);
static void StopApplication (void);
static void EventLoop (void);
static Boolean ApplicationHandleEvent(EventPtr event);
static void goup();
static void godown();
static void hear(POA *actPOAptr);
static void stopTrack();
static void selected();
static void playTrack(int trackNumber);
static void gotoTrackOne();
static void selected(int x, int y);
static void drawRect(short upperX, short upperY, short extentX, short extentY);
static void drawPOA(POA *actPOA, Boolean type);
static void drawRoom(Room *actRoom);
static void drawFloor(Floor *actFloor);
static POA *getNearestPOA(Floor *actFloorPtr, int x, int y);
static int distance(int x1, int y1, int x2, int y2);
static void eraseWin();
static void stopPlay();

```

```

    }
    if (x >= playButtonX && x <= playButtonX+buttonWidth && y >buttonYpos
        && y <= buttonYpos+buttonHeight) {
        if (!playing){
            hear(myPOAPtr); // pen down PLAY
        }
        if (x >= stopButtonX && x <= stopButtonX+buttonWidth && y >buttonYpos
            && y <= buttonYpos+buttonHeight) {
            stopplay(); // pen down STOP
        }
        return 1;
    }
}
/*****
* * FUNCTION: StartApplication
* * DESCRIPTION: This routine sets up the application.
* * PARAMETERS: nothing
* * RETURNED: true if there is an error starting the application.
* * REVISION HISTORY:
* * NameDateDescription
* * -----
* * nik29.6.97Initial Revision
* *
* *
* *
static void StopApplication (void)
{
    SerClose(SerialRef);

    SysReset();
}
/*****
* * FUNCTION: ApplicationHandleEvent
* * DESCRIPTION: Dispatches the events into the EventQueue
* * PARAMETERS: nothing
* * RETURNED: nothing
* * REVISION HISTORY:
* * NameDateDescription
* * -----
* * nik29.6.97Initial Revision
* *
* *
Boolean ApplicationHandleEvent(EventPtr e)
{
    if (e->eType == frmLoadEvent) {
        FrmPtr frm = FrmInitForm(NewForm);
        FrmSetActiveForm(frm);
        FrmSetEventHandler (frm, MainHandleEvent);

        return true;
    }
    else if (e->eType == frmOpenEvent) {

```





```

MoreStairs0[0]=&Stair0;
MoreStairs1[0]=&Stair1;
MoreStairs1[1]=&Stair2;
MoreStairs2[0]=&Stair3;

Floor Floor0(MoreRooms0Ptr,2,0, MoreStairs0,1);
Floor Floor1(MoreRooms1Ptr,2,1, MoreStairs1,2);
Floor Floor2(MoreRooms2Ptr,1,2, MoreStairs2,1);

MoreFloorsPtr[0]=&Floor0;
MoreFloorsPtr[1]=&Floor1;
MoreFloorsPtr[2]=&Floor2;

Building myBuilding (MoreFloorsPtr,3);

MyMuseumPtr = &myBuilding;

EventLoop ();

}
StopApplication ();
}
return 0;
}

/*****
 *
 * FUNCTION: MiscFunctions
 *
 * DESCRIPTION:
 *
 * PARAMETERS:
 *
 * RETURNED:
 *
 * REVISION HISTORY:
 *
 * NameDateDescription
 * -----
 * nik29.6.97Initial Revision
 *
 *****/
/*****
 *
 * Track switching operations
 *
 *****/

void gotoTrackOne() { // draws one floor
    int i;
    for (i = 0; i < actTrack+maxTrackNumber; i++)
    {
        char *s;
        WinDrawChars("
        WinDrawChars("Rewind:",7,1,120);
        s=StrIToA (s,(long)actTrack+maxTrackNumber-i);
        WinDrawChars(s,StrLen(s),34,120);
        SerSend(SerialRef,"5",1); // sends trackNumber Rewinds
        SysTaskDelay(trackDelay);
    }
    WinDrawChars("
    ",20,1,120);
}

void playTrack(int trackNumber){ // plays the track number trackNumber
    int i;
    SerSend(SerialRef,"5",1);
    SysTaskDelay(trackDelay+50);
    WinDrawChars("Track Search
    ",18,1,130);
    for (i = 1; i < trackNumber; i++)
    {
        char *s;
        WinDrawChars("
        WinDrawChars("Track:",6,1,120);
        s=StrIToA (s,(long)i);
        WinDrawChars(s,StrLen(s),30,120);
        SerSend(SerialRef,"4",1); // sends trackNumber Forwards
        SysTaskDelay(trackDelay);
    }
}

```

```

void hear(POA *actPOAptr)//play the track concerning th actual POA
{
    playing=true;

    playTrack(actTrack);
    startime=TimGetSeconds();
    eraseWin();
    drawFloor(MyMuseumPtr->GetFloor(floor+1));
}

void stopplay() //stops playing a track and goes to track one
{
    if(playing){
        WinDrawChars("Stopped
        ",18,1,130);
        gotoTrackOne();
    }
    stopTrack();
    playing=false;
}
/*****
*
*
*
***** Drawing routines for the building
*****/

void drawPOA(POA *actPOA, Boolean type){
    PointType point,ext;
    RectangleType r;
    RectanglePtr rPtr;

    if (type){
        char *s;
        s=StrITOA (s,(long)actPOA->tracknum);
        WinDrawChars(s,StrLen(s),actPOA->posX-2,actPOA->posY-4);
    }
    else
    {
        point.x=actPOA->posX-3;
        point.y=actPOA->posY-2;
        ext.x=10;
        ext.y=8;
        r.topLeft=point;
        r.extent=ext;
        rPtr=&r;
        WinFillRectangle(rPtr,1);
    }
}

```

```

}
WinDrawChars("
",20,1,120);
}

void stopTrack(){
    SerSend(SerialRef,"8",1);
    SysTaskDelay(trackDelay);
}

/*****
*
*
*
***** Routines to treat the Button functions
*****/

void selected(int x, int y){ // fixes the selection of one POA
    eraseWin();
    drawFloor(MyMuseumPtr->GetFloor(floor+1));
    myPOAptr=getNearestPOA(MyMuseumPtr->GetFloor(floor+1),x,y);
    drawbegin=false;
}

void goup() //go up one floor
{
    if (floor<MyMuseumPtr->GetFloorNum()-1){
        floor=floor+1;
        eraseWin();
        drawFloor(MyMuseumPtr->GetFloor(floor+1));
    }
}

void godown() //go down one floor
{
    if (floor>0){
        floor=floor-1;
        eraseWin();
        drawFloor(MyMuseumPtr->GetFloor(floor+1));
    }
}

```

```

void drawRoom(Room *actRoom){
    PointType point,ext;
    RectangleType r;
    RectanglePtr rPtr;

    point.x=actRoom->upperX;
    point.y=actRoom->upperY;
    ext.x=actRoom->lowerX-actRoom->upperX;
    ext.y=actRoom->lowerY-actRoom->upperY;
    r.topLeft=point;
    r.extent=ext;
    rPtr=&r;

    WinDrawRectangleFrame(l,rPtr);
    for (int j = 0; j <= actRoom->GetPOANum()-1; j++){
        drawPOA(actRoom->GetPOA(j),true);
    }
}

void drawStairs(Stair *actStair){
    PointType point,ext;
    RectangleType r;
    RectanglePtr rPtr;

    if (actStair->GetDirection()){
        for (int i=actStair->upperY;i<actStair->lowerY;i=i+4){
            WinDrawLine(i,actStair->upperY,i,actStair->lowerY);
        }
    }else
    {
        for (int i=actStair->upperY;i<actStair->lowerY;i=i+4){
            WinDrawLine(actStair->upperX,i,actStair->lowerX,i);
        }
    }

    point.x=actStair->upperX;
    point.y=actStair->upperY;
    ext.x=actStair->lowerX-actStair->upperX;
    ext.y=actStair->lowerY-actStair->upperY;
    r.topLeft=point;
    r.extent=ext;
    rPtr=&r;
}

void drawRoom(Frame l,rPtr){
    PointType point,ext;
    RectangleType r;
    RectanglePtr rPtr;

    point.x=actRoom->upperX;
    point.y=actRoom->upperY;
    ext.x=actRoom->lowerX-actRoom->upperX;
    ext.y=actRoom->lowerY-actRoom->upperY;
    r.topLeft=point;
    r.extent=ext;
    rPtr=&r;

    WinDrawRectangleFrame(l,rPtr);
    for (int j = 0; j <= actRoom->GetPOANum()-1; j++){
        drawPOA(actRoom->GetPOA(j),true);
    }
}

void drawStairs(Stair *actStair){
    PointType point,ext;
    RectangleType r;
    RectanglePtr rPtr;

    if (actStair->GetDirection()){
        for (int i=actStair->upperY;i<actStair->lowerY;i=i+4){
            WinDrawLine(i,actStair->upperY,i,actStair->lowerY);
        }
    }else
    {
        for (int i=actStair->upperY;i<actStair->lowerY;i=i+4){
            WinDrawLine(actStair->upperX,i,actStair->lowerX,i);
        }
    }

    point.x=actStair->upperX;
    point.y=actStair->upperY;
    ext.x=actStair->lowerX-actStair->upperX;
    ext.y=actStair->lowerY-actStair->upperY;
    r.topLeft=point;
    r.extent=ext;
    rPtr=&r;
}

void drawFloor(Floor *actFloor){
    char *s;
    for (int i = 0; i < actFloor->GetRoomNum(); i++){
        drawRoom(actFloor->GetRoom(i));
    }

    for (int j = 0; j < actFloor->stairsCount; j++){
        drawStairs(actFloor->Stairs[j]);
    }
}

WinDrawChars("Floor:",6,100,1);
s=StrIToA (s,(long)Floor);

WinDrawChars(s,1,130,1);

void eraseWin(){
    PointType point,ext;
    RectangleType r;
    RectanglePtr rPtr;

    point.x=0;
    point.y=15;
    ext.x=XSIZE;
    ext.y=YSIZE;
    r.topLeft=point;
    r.extent=ext;
    rPtr=&r;

    WinEraseRectangle(rPtr,0);
}

```

```

/*****
*
*           Get the neares POA from the pen selection
*
*****/
int distance(int x1, int y1, int x2, int y2){
    return (int)((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
}

POA *getNearestPOA(Floor *actFloorPtr,int x, int y){
    Room *actRoomPtr;
    POA *actPOAPtr;
    POA *nearestPOAPtr;
    int actDist=1000;
    nearestPOAPtr=actFloorPtr->Rooms[0]->GetPOA(1);
    for (int j = 0; j <= actFloorPtr->GetRoomNum()-1; j++){
        actRoomPtr=actFloorPtr->GetRoom(j);
        for (int i = 0; i <= actRoomPtr->GetPOANum()-1; i++){
            actPOAPtr=actRoomPtr->GetPOA(i);
            if (distance(x,y,actPOAPtr->posX,actPOAPtr->posY)< actDist){
                actDist=distance(x,y,actPOAPtr->posX,actPOAPtr->posY);
                nearestPOAPtr=actPOAPtr;
            }
        }
    }
    drawPOA(nearestPOAPtr,false);
    actTrack=nearestPOAPtr->tracknum;
    actDuration=nearestPOAPtr->duration;
    return nearestPOAPtr;
}

```