QCard DLL Lesson 10

Lesson 9 Lesson 11

- Alyce

See Lesson 1 for QCard DLL and WAV files needed for the demo code.

More on Card Dragging

In the previous demo we allowed the user to right-click on a card and drag it around. When the user released the button the card was dropped in a new position. We paid no attention to the card's status or to the location it was dropped. In a real game we'd need to monitor all such activity.

Blocked Cards

QCard has two functions to set and query the blocked status of a card. A card can be marked as blocked if it is covered or partially covered by another card. QCard lets us set the "blocked" status with AdjustCardBlocked. We can query the "blocked" status of a card with GetCardBlocked. The API calls look like this:

```
calldll #qc, "AdjustCardBlocked",_
    nC as long,_     'index of card to block/unblock
    bValue as long,_ '1=blocked, 0=unblocked
    re as void       'no return

calldll #qc, "GetCardBlocked",_
    nC as long,_       'index of card to query
    isBlocked as long  '1=blocked, 0=unblocked
```

Blocking Some Cards

In our demo, we'll change the display so that the rows of cards overlap. Each row is 20 pixels below the previous row. When we deal the first three rows we'll also set the "blocked" status of each card in that row to "blocked". We will leave the cards in the fourth row with the default "unblocked" status. We've introduced a variable called "row" to keep track of the row being dealt.

```
    x=10:y=2:row=1
    for i = 1 to 24
        Call DealCard hBox,card(i),x,y
        'When creating a row or pile of cards,
        'only the topmost card should have an
        'IsBlocked value of FALSE.
        if row<4 then call AdjustCardBlocked card(i),1

        x=x+100
        if x>510 then   'move to next row
            x=10
            y=y+offset
            row=row+1
        end if
        playwave "card.wav",sync

       'pause 100 milliseconds between cards
        call Pause 100
        scan
    next
```

Aborting the Dragging Operation

QCard DLL captures mouse events from the time InitDrag is called until EndDrag is called. If for any reason we need to abort the dragging procedure, we must call AbortDrag so that the DLL releases the capture of mouse events.

```
    calldll #qc, "AbortDrag", re as void
```

Preventing the Drag of Blocked Cards

When the user right-clicks the mouse we'll use InitDrag to discover which card in our card array was clicked. InitDrag returns the index of the card at the given x,y location. We then query its blocked status.

If the card is blocked we document that at the bottom of the display and halt the routine with AbortDrag. If it is not blocked, we begin the dragging procedure, as we did in the last lesson.

```
[initDrag]
    cardIndex=InitDrag(hBox,MouseX,MouseY)
```

```
    isBlocked=GetCardBlocked(cardIndex)    'see if card is blocked
    if isBlocked<>0 then
        'if card is blocked, inform user and abort drag operation
        call AbortDrag
        #1.g
"place 10 420;\BLOCKED Card Index is ";cardIndex;space$(100)
        wait
    end if
    #1.g "place 10 420;\Drag Card Index is ";cardIndex;space$(100)
    #1.g "when rightButtonMove [doDrag]"
    wait
```

In a real game you'd want to set the blocked status of the newly uncovered card to 0. You'd also need to set the card that was just covered by the dragged card to "blocked". We did not complicate the code with these routines in order to keep the demo program short and understandable.

Offset Value

QCard has a default offset value of 16 pixels. The cards are 70x95 pixels and if one card is displayed 16 pixels lower thena the card beneath both card values will be visible. **This offset is important to the DLL when it discovers which card is at MouseX and MouseY.**

We can change the offset with SetOffSet. We change the offset to 20 pixels for this demo.

```
    calldll #qc, "SetOffSet",_
'change pixel offset for vertical colums of cards
        offset as long,_        'number of pixels to offset
        re as void
```

DEMO

See Lesson 1 for QCard DLL and WAV files needed for the demo code.

The demo deals the cards in four overlapped rows. It sets the blocked status of cards in the first three rows. It checks the blocked status of the card clicked by the user. It allows the user to drag the card if the designated card status is "unblocked". We've eliminated the shuffling routine to keep things simple for this demo.

```
'An open project card game, begun by Alyce Watson, May 27, 2003.
```

```
'Uses Qcard32.dll, a freeware library of playing card images.
'DLL by Stephen Murphy.  Qcard32.DLL website:
'http://www.telusplanet.net/public/stevem/

dim card(24)            'array to hold cards
gosub [fillCardArray]   'fill array with card values

[varSetup]
i=0         'i will be our counter var in for/next loops
offset=20   'offset for card vertical spacing for dragging purposes

nomainwin
    WindowWidth=640:WindowHeight=480
    UpperLeftX=1:UpperLeftY=1

    menu #1, "&File", "&New",[new],"E&xit", [quit]
    graphicbox #1.g, 0, 0, 640, 440
    open "Dragging Cards" for window_nf as #1
    #1 "trapclose [quit]"

    'get graphicbox handle
    hBox=hwnd(#1.g)

    'open the dll
    open "qcard32.dll" for dll as #qc
    'initialize the deck
    Call InitializeDeck hBox

[new]
    Call SetDefaultValues

    'draw a nice background
    #1.g "down; fill 10 190 225"
    #1.g "backcolor 10 190 225"
    'temp message for this demo only
    #1.g "place 10 420"
    #1.g "\Right-
click on a card, then move mouse to drag and drop it."

    offset=20
    call SetOffSet offset    'set offset to 20 pixels - default is 16

    'set xy location to start deal
    x=10:y=2:row=1
    for i = 1 to 24
        Call DealCard hBox,card(i),x,y
```

```
        'When creating a row or pile of cards,
        'only the topmost card should have an
        'IsBlocked value of FALSE.
        if row<4 then call AdjustCardBlocked card(i),1

        x=x+100
        if x>510 then   'move to next row
            x=10
            y=y+offset
            row=row+1
        end if
        playwave "card.wav",sync

       'pause 100 milliseconds between cards
        call Pause 100
        scan
    next
    #1.g "when rightButtonDown [initDrag]"
wait

[initDrag]
    cardIndex=InitDrag(hBox,MouseX,MouseY)
    isBlocked=GetCardBlocked(cardIndex)    'see if card is blocked
    if isBlocked<>0 then
        'if card is blocked, inform user and abort drag operation
        call AbortDrag
        #1.g
"place 10 420;\BLOCKED Card Index is ";cardIndex;space$(100)
        wait
    end if
    #1.g "place 10 420;\Drag Card Index is ";cardIndex;space$(100)
    #1.g "when rightButtonMove [doDrag]"
    wait

[doDrag]
    call DoDrag hBox, MouseX, MouseY
    #1.g "when rightButtonUp [endDrag]"
    wait

[endDrag]
    #1.g "when rightButtonUp"
    cardIndex=EndDrag(hBox,MouseX,MouseY)
    #1.g
"place 10 420;\Destination Card Index is ";cardIndex;space$(100)
    wait
```

```
[fillCardArray]
    'fill card array
    'cards 1 to 52 are in the first deck
    'cards 53 to 104 are in the second deck
    'use cards Jack through King in each suit, first deck
    card(1)=11  'jack of clubs
    card(2)=12  'queen
    card(3)=13  'king
    card(4)=24  'jack of diamonds
    card(5)=25  'queen
    card(6)=26  'king
    card(7)=37  'jack of hearts
    card(8)=38  'queen
    card(9)=39  'king
    card(10)=50  'jack of spades
    card(11)=51  'queen
    card(12)=52  'king

    'now use second deck, to fill second half of array
    for i = 1 to 12
        card(i+12)=card(i)+52
    next
    RETURN

[quit] close #qc:close #1:end


''''''''''''''''''''
'subs and functions:
Sub Pause ms
    'pause ms number of milliseconds
    calldll #kernel32,"Sleep",_
    ms as long, re as void
    End Sub

Sub SetOffSet offset
    calldll #qc, "SetOffSet",offset as long,_
        re as void
    end sub

Sub AdjustCardBlocked nC, bValue
    calldll #qc, "AdjustCardBlocked",_
        nC as long, bValue as long, re as void
    end sub
```

```
Function GetCardBlocked(nC)
    calldll #qc, "GetCardBlocked",nC as long,_
        GetCardBlocked as long
    end function

Function InitDrag(hndle, x, y)
    calldll #qc, "InitDrag",_
        hndle as ulong, x as long, y as long,_
        InitDrag as long
    end function

Sub AbortDrag
    calldll #qc, "AbortDrag",re as void
    end sub

Sub DoDrag hndle,x,y
    calldll #qc, "DoDrag",hndle as ulong,_
    x as long, y as long, r as void
    end sub

Function EndDrag(hndle,x,y)
    calldll #qc, "EndDrag",hndle as ulong,_
    x as long, y as long, EndDrag as long
    end function

Sub InitializeDeck hndle
    calldll #qc, "InitializeDeck",_
    hndle as ulong,r as long
    End Sub

Sub DealCard hndle,nC,x,y
    'places card on window whose handle is hndle at x,y
    'nC is number of card - 1-52 in first deck and
    '53-104 in second deck, if used
    calldll #qc, "DealCard",hndle as ulong,nC as long,_
    x as long,y as long,r as void
    End Sub

Sub SetDefaultValues
    'reset all card properties back to their default values.
    calldll #qc, "SetDefaultValues",r as void
    End Sub
```

QCard DLL Lesson 10 | More on Card Dragging | Blocked Cards | Blocking Some Cards | Aborting the Dragging Operation | Preventing the Drag of Blocked Cards | Offset Value | DEMO

Lesson 9 Lesson 11