QCard DLL Lesson 9

Lesson 8 Lesson 10

-
Alyce

QCard DLL Lesson 9 | Dragging a Card | Mouse Click Starts Drag | InitDrag | AbortDrag | DoDrag | EndDrag | DEMO

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

Dragging a Card

QCard has features that allows us to drag a single card or a block of cards. We'll start with dragging a single card.

Mouse Click Starts Drag

We can start the dragging event when the user presses a mouse button down. We'll use the right button for this. Once the button is down, we'll activate the mouse move event for the right button and call on QCard to intialize the dragging procedure. Since InitDrag returns the index of the clicked card, we'll write that on the display so we can see what's happening. We need to send the graphicbox handle and MouseX and MouseY to the function.

```
    #1.g "when rightButtonDown [initDrag]"
wait

[initDrag]
    cardIndex=InitDrag(hBox,MouseX,MouseY)
    #1.g "place 10 420;\Drag Card Index is ";cardIndex;space$(100)
    #1.g "when rightButtonMove [doDrag]"
    wait
```

InitDrag

Here is the documentation from QCard for InitDrag.

- *Use this function in a MouseDown event to start a drag operation. The function searches through all cards to determine if the mouse cursor is over any card whose IsBlocked property is FALSE. If it finds one, it returns the number of that card. If it does not find one, it searches through all the cards in the deck to see if the mouse lies in the top 16 (or User-Defined OffSet) pixels of any card, blocked or not. If it does, it returns the number of that card.*

- *By checking the IsBlocked property of this returned card, you can tell if the user wants to carry out a single drag or a block drag. If InitDrag returns a value of 0, the mouse is not currently located over any card. The InitDrag function should always be followed by a corresponding AbortDrag, an EndDrag or an EndBlockDrag call. This is due to that fact that InitDrag "captures" all mouse input, and requires one of these corresponding calls to release the capture. The values nx and ny are the current mouse coordinates.*

Here is the InitDrag function, wrapped in a Liberty BASIC function.

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

AbortDrag

QCard DLL captures mouse events while dragging. If mouse event capture is not released, the program will crash. Mouse event capture is released by EndDrag, but if something goes amiss, we can assure a stable program by calling AbortDrag to release mouse event capture.

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

DoDrag

DoDrag is the subroutine used by QCard to move the card. It requires the handle of the graphicbox and MouseX and MouseY.

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

The documentation tells us that DoDrag, "Carries out the drag operation which was initiated by InitDrag call. DoDrag moves the current Source Card to its new location. Values nx and ny are the current mouse coordinates." It does not return a value.

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

EndDrag

We stop the dragging operation and drop the card at the current location when the right mouse button is released. The EndDrag function requires the handle of the graphicbox and MouseX and MouseY. We also cancel the "rightButtonUp" event in our code.

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

Qcard tells us this about EndDrag, "EndDrag ends a single drag operation and returns the number of the Destination card (that is, the card it is being dropped on), if any. It searches the deck for any card which overlaps the Source Card and whose IsBlocked property is FALSE. If it finds one, it returns the number of that card. The function also releases the mouse which was captured by the InitDrag call."

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

DEMO

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

The demo uses the foundation of our Memory card game. It invites the user to right click and drag a card.

```
'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
newIndex=0              'used when shuffling
tempCard=0              'temp var used when shuffling

[varSetup]
i=0         'i will be our counter var in for/next loops
design=1    'default design is circles

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 "Card Dragging" 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
    Call SetCurrentBack design

    '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."
    gosub [shuffleCards]

    'set xy location to start deal
    x=10:y=2
```

```
    for i = 1 to 24
        'set status of all cards to 0, which is face down
        ' - we won't do this yet, so we can see the results
        'of our deal
        'call SetCardStatus card(i), 0

        Call DealCard hBox,card(i),x,y
        x=x+100
        if x>510 then   'move to next row
            x=10
            y=y+100
        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)
    #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


[shuffleCards]
    playwave "shuffle.wav",async
    'now shuffle cards
    for i = 1 to 24
        newIndex=int(rnd(0)*24)+1
        tempCard=card(i)  'temp var to allow switching values
        card(i)=card(newIndex)
'this index now contains value from random index
        card(newIndex)=tempCard
'random index now contains value from other index

'now card(i) has switched values with a random card in the array
    next
    playwave "shuffle.wav",sync
    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

Function InitDrag(hndle, x, y)
```

```
    calldll #qc, "InitDrag",_
        hndle as ulong, x as long, y as long,_
        InitDrag as long
    end function

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 SetCardStatus nC,face
    'nC is number of card - 1-52 in first deck and
    '53-104 in second deck, if used
    'face:  0=facedown,1=faceup
    calldll #qc, "SetCardStatus",nC as long,_
    face as long,r as void
    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 DrawBack hndle, nV, x, y
    'nV can be 1,2,3,4,5,6 for 6 possible designs
    'draws a cardback image on screen
    calldll #qc, "DrawBack",hndle as ulong,_
    nV as long,x as long,y as long,r as void
    End Sub

Sub SetCurrentBack nV
    'nV can be 1,2,3,4,5,6 for 6 possible designs
    calldll #qc, "SetCurrentBack",nV 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 9 | Dragging a Card | Mouse Click Starts Drag | InitDrag | AbortDrag | DoDrag | EndDrag | DEMO

Lesson 8 Lesson 10