# QCard DLL Lesson 5

Lesson 4 Lesson 6

-
Alyce

QCard DLL Lesson 5 | Getting Card Info | Suit? | Value? | Color? | Status? | DEMO

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

## Getting Card Info

In earlier lessons, we discussed dealing the cards, showing their fronts or their backs, and changing the card back design. To create a playable game, we must ascertain the suit and value of a card selected by the user. In Lesson 4, we discovered the card that was clicked by the user and found its index in our card array. With that index, we can find out more about the card.

## Suit?

Remember, we have filled an array called card() with a set of shuffled cards. When we know which card from the array was selected by the user, we check the value of that card and place it into a variable called clickCard . See Lesson 4 to refresh your memory on this.

There is a function in the Qcard DLL to get the suit of a card, called GetCardSuit() It requires the index number of the card, and returns the suit as a number. A return of 1 means the card is a club, 2 means it is a diamond, 3 means it is a heart and 4 means that it is a spade.

```
calldll #qc, "GetCardSuit",_
  nIndex as long,_    'index of card to query
  suit as long
'returns 1=Clubs, 2=Diamonds, 3=Hearts, 4=Spades.
```

Here is the API call, wrapped in a Liberty BASIC function:

```
Function GetCardSuit(nC)
    'returns 1=Clubs, 2=Diamonds, 3=Hearts, 4=Spades.
    calldll #qc, "GetCardSuit",nC as long,_
    GetCardSuit as long
```

```
    End Function
```

# Value?

In a very similar way, we can get the value of the card. The function from the DLL is GetCardValue . The first argument is the index of the card to query. The function returns the value of the card, where an ace is 1, a deuce is 2, and so on, up to 11 for jack, 12 for queen, and 13 for king.

```
calldll #qc, "GetCardValue",_
  nIndex as long,_    'index of card to query
  value as long       'ace=1,deuce=2....jack=11,queen=12,king=13
```

Here is the API call, wrapped in a Liberty BASIC function:

```
Function GetCardValue(nC)
    'ace=1,deuce=2....jack=11,queen=12,king=13
    calldll #qc, "GetCardValue",nC as long,_
    GetCardValue as long
    End Function
```

# Color?

We don't need to know the color of a card in our demo, but there is a function for this in the Qcard DLL. The first argument is the index of the card, and it returns the color. A return of 1 means the card is black, while 2 means that it is red.

```
calldll #qc, "GetCardColor",_
  nIndex as long,_    'index of card to query
  color as long       '1=black, 2=red
```

# Status?

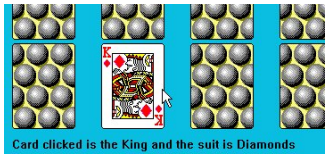
In our demo we've used the function SetCardStatus to cause either the back or the front of a card to be displayed. There is also a function to GetCardStatus in the DLL. The first argument is the index of the card to query, and the return is the card's status. A status of 1 means that the card is face up, while 0 means that it is face down.

```
calldll #qc, "GetCardStatus",_
  nIndex as long,_    'index of card to query
  status as long      '1=face up, 0=face down
```

# DEMO

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

The demo program that accompanies this lesson allows the user to click on one of the cards, and the program ascertains the suit and value of the card. It then gives the user a message stating which suit and value were chosen.




```
'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/

'new in cards5.bas:
'This demo determines the suits and values of cards clicked,
'and displays a message about them.



[varSetup]
i=0         'i will be our counter var in for/next loops
design=1    'default design is circles
dim card(24)'array to hold cards
newIndex=0  'used when shuffling
tempCard=0  'temp var used when shuffling
clickCard=0 'index of current card clicked by user
gosub [fillCardArray]   'fill array with card values


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

    menu #1, "&File", "&New",[new],"E&xit", [quit]
```

```
    menu #1, "&Card Back Design","&Circles",[circles],"&Blue",[blue],_
    "&Red",[red],"&Mountain",[mountain],"&Purple",[purple],
"M&usic",[music]
    graphicbox #1.g, 0, 0, 640, 440
    open "Memory Card Game" for window_nf as #1
    #1 "trapclose [quit]"

    'trap mouse clicks:
    #1.g "setfocus; when leftButtonUp [checkIndex]"

    '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 "\Try Menu File -> New Game to shuffle deck."
    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
        Call SetCardStatus card(i), 0

        'deal cards
        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
    wait


[checkIndex]
    clickCard=0:x=0:y=0 'reset values
    mx=MouseX   : my=MouseY 'mouse x and y location
    'Cards are placed in a grid that is 100x100,
    'so it is easy to determine which card is clicked
    'by checking mouse position. Card height is about
    '100, and width is about 80.
    'Index of clicked card is placed in var called clickCard
    'and x,y locations are placed in vars called x and y.
    'MouseY determines row, and MouseX determines column.
    select case
    case my<=102                    'first row
        y=2
        if mx<=90 then clickCard=1:x=10
        if (mx>=110) and (mx<=190) then clickCard=2:x=110
        if (mx>=210) and (mx<=290) then clickCard=3:x=210
        if (mx>=310) and (mx<=390) then clickCard=4:x=310
        if (mx>=410) and (mx<=490) then clickCard=5:x=410
        if (mx>=510) and (mx<=590) then clickCard=6:x=510
    case (my>=102) and (my<202)     'second row
        y=102
        if mx<=90 then clickCard=7:x=10
        if (mx>=110) and (mx<=190) then clickCard=8:x=110
        if (mx>=210) and (mx<=290) then clickCard=9:x=210
        if (mx>=310) and (mx<=390) then clickCard=10:x=310
        if (mx>=410) and (mx<=490) then clickCard=11:x=410
        if (mx>=510) and (mx<=590) then clickCard=12:x=510
    case (my>=202) and (my<302)     'third row
        y=202
        if mx<=90 then clickCard=13:x=10
        if (mx>=110) and (mx<=190) then clickCard=14:x=110
        if (mx>=210) and (mx<=290) then clickCard=15:x=210
        if (mx>=310) and (mx<=390) then clickCard=16:x=310
        if (mx>=410) and (mx<=490) then clickCard=17:x=410
        if (mx>=510) and (mx<=590) then clickCard=18:x=510
    case (my>=302) and (my<402)     'fourth row
        y=302
        if mx<=90 then clickCard=19:x=10
        if (mx>=110) and (mx<=190) then clickCard=20:x=110
```

```
        if (mx>=210) and (mx<=290) then clickCard=21:x=210
        if (mx>=310) and (mx<=390) then clickCard=22:x=310
        if (mx>=410) and (mx<=490) then clickCard=23:x=410
        if (mx>=510) and (mx<=590) then clickCard=24:x=510
    case else
        clickCard=0
    end select

    if clickCard=0 then wait
    'turn card so that it is face up
    Call SetCardStatus card(clickCard), 1
    'deal card again so that it displays face up
    Call DealCard hBox, card(clickCard), x, y

    gosub [readValue]
    wait


[readValue]
    thisVal = GetCardValue(card(clickCard))
    'ace=1,deuce=2....jack=11,queen=12,king=13
    value$=
"Ace Deuce Trey Four Five Six Seven Eight Nine Ten Jack Queen King"

    thisSuit = GetCardSuit(card(clickCard))
    'returns 1=Clubs, 2=Diamonds, 3=Hearts, 4=Spades.
    suit$="Clubs Diamonds Hearts Spades"

    'This routine is for this demo, just for fun.
    'Values are 1-13, so placing the words that correspond
    'to the values in a string, we can extract the word that
    'matches the numeric value with the word$ function.
    'In the same way, suits are 1-4, so placing the suit names
    'in a string in order so that the names correspond with the
    'values allows us to extract the word from the string with
    'the word$ function.
    thisVal$=word$(value$,thisVal)
    thisSuit$=word$(suit$,thisSuit)

    msg$="Card clicked is the ";thisVal$;" and the suit is ";thisSuit$

    #1.g "place 10 420"
    #1.g "\" ; msg$; space$(400)
    RETURN
```

```
'setting new card back doesn't restart game,
'so new back won't show until new game is started:
[circles] design=1:goto [setDesign]
[blue] design=2:goto [setDesign]
[red] design=3:goto [setDesign]
[mountain] design=4:goto [setDesign]
[purple] design=5:goto [setDesign]
[music] design=6:goto [setDesign]

[setDesign]
    Call SetCurrentBack design
    'design can be 1,2,3,4,5,6 for 6 possible designs
    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 GetCardSuit(nC)
    'returns 1=Clubs, 2=Diamonds, 3=Hearts, 4=Spades.
    calldll #qc, "GetCardSuit",nC as long,_
    GetCardSuit as long
    End Function

Function GetCardValue(nC)
    'ace=1,deuce=2....jack=11,queen=12,king=13
    calldll #qc, "GetCardValue",nC as long,_
    GetCardValue 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 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 5 | Getting Card Info | Suit? | Value? | Color? | Status? | DEMO

Lesson 4 Lesson 6