

## QCard DLL Lesson 8

[Lesson 7](#) [Lesson 9](#)

---

[Alyce](#)

[QCard DLL Lesson 8](#) | [Finishing the Memory Card Game](#) | [Duplicate Click Error](#) | [End of Game](#) | [Play Again?](#) | [DEMO](#)

---

See [Lesson 1](#) for QCard DLL and WAV files needed for the demo code.

## Finishing the Memory Card Game

It is time to fix some errors, and the game is done!

## Duplicate Click Error

In the code for lesson 7, if the user turns over a card and clicks it again while it is turned up the program registers a match! We can fix that simply by checking to see if cardTwo is in the same spot as cardOne, and if it is, stop and wait for more user events.

```
'Check to see if the user has already exposed this card.  
if clickCard=cardOne then wait
```

That was easy!

## End of Game

There is an additional error in the code from lesson 7. Nothing happens when all cards are removed from the table. The user needs to get a message and have the ability to begin a new game. We've added a flag to indicate that the game is over. The gameWon variable is 0 when a game is in progress.

```
gameWon=0    'flag that is set when all pairs are removed
```

When the game is over we set the flag to 1.

```
if score=12 then gameWon=1 'flag that all pairs are removed
```

## Play Again?

When the game is over, we ask the user if he wants to play again. If he does, we go to the [new] routine. If he doesn't, we disable mouse events and wait. 'We've moved the mouse-button trapping code so that it appears after the [new] label rather than before it 'so a new game automatically restores the trapping of the leftButtonUp event.

```
'If all pairs have been removed, ask user if he
'wants to play again.
if gameWon=1 then
    msg$="You have won in ";turns;" turns. Play again?"
    confirm msg$;answer$
    if answer$="yes" then
        'start a new game
        goto [new]
    else
        'disable mouse event trapping and wait
        #1.g "when leftButtonUp"
    end if
end if
wait
```

## DEMO

See [Lesson 1](#) for QCard DLL and WAV files needed for the demo code.

```
'new this time
gameWon=0 'flag that is set when all pairs are removed

[varSetup]
i=0 'i will be our counter var in for/next loops
design=1 'default design is circles
newIndex=0 'used when shuffling
tempCard=0 'temp var used when shuffling
clickCard=0 'index of current card clicked by user
dim card(24,2) 'array to hold card info
    'card(n,1)=index of card in deck
    'card(n,2)=visible on table? 1=yes, 0=no
```

```
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]"

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

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

[new] 'reset variables and shuffle cards for next try
    turns=0      : score=0
    clickCard=0  : gameWon=0
    cardOne=0    : cardTwo=0
    cardOneX=0   : cardTwoX=0
    cardOneY=0   : cardTwoY=0

    Call SetDefaultValues
    Call SetCurrentBack design

    'draw a nice background
    #1.g "down; fill 10 190 225"
    #1.g "backcolor 10 190 225"
    'trap mouse clicks:
    #1.g "setfocus; when leftButtonUp [checkIndex]"

    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,1), 0
```

```
'deal cards
Call DealCard hBox,card(i,1),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

'Check to see if the user has already exposed this card.
if clickCard=cardOne then wait

'if card is not visible (has been removed), then wait
if card(clickCard,2)=0 then wait

'remove card to restore tabletop
call RemoveCard hBox, card(clickCard,1)

'set status of cards to 1, which is face up
Call SetCardStatus card(clickCard,1), 1

'deal card face up
Call DealCard hBox,card(clickCard,1),x,y

gosub [readValue]

'If all pairs have been removed, ask user if he
'wants to play again.
if gameWon=1 then
    msg$="You have won in ";turns;" turns.  Play again?"
    confirm msg$;answer$
    if answer$="yes" then
        'start a new game
        goto [new]
    else
```

```
        'disable mouse event trapping and wait
        #1.g "when leftButtonUp"
    end if
end if
wait
```

```
[readValue]
```

```
    'check whether this is first or second card
    if cardOne=0 then
        cardOne=clickCard
        cardOneX=x
        cardOneY=y
        return 'leave first card up and return
    else
        cardTwo=clickCard
        cardTwoX=x
        cardTwoY=y
    end if
```

```
#1.g "when leftButtonUp" 'turn off mouse event while pausing
call Pause 2000          '2 second pause to view cards
#1.g "setfocus; when leftButtonUp [checkIndex]"
```

```
oneVal = GetCardValue(card(cardOne,1))
twoVal = GetCardValue(card(cardTwo,1))
'ace=1,deuce=2....jack=11,queen=12,king=13
oneSuit = GetCardSuit(card(cardOne,1))
twoSuit = GetCardSuit(card(cardTwo,1))
'returns 1=Clubs, 2=Diamonds, 3=Hearts, 4=Spades.
```

```
'Remove cards from table --
'they will be redealt if they don't match.
call RemoveCard hBox, card(cardOne,1)
call RemoveCard hBox, card(cardTwo,1)
turns=turns+1
```

```
'See if cards match each other in suit and value.
'If they don't match, turn them face down and redeal them.
if (oneVal<>twoVal) or (oneSuit<>twoSuit) then
    'set status of cards to 0, which is face down
    Call SetCardStatus card(cardOne,1), 0
    Call SetCardStatus card(cardTwo,1), 0

    'deal card face down
    Call DealCard hBox,card(cardOne,1),cardOneX,cardOneY
```

```
        Call DealCard hBox,card(cardTwo,1),cardTwoX,cardTwoY
    else
        'If cards match, increment score and don't
        'replace them on the table.
        'Set visible to 'off'
        card(cardOne,2)=0
        card(cardTwo,2)=0
        score=score+1
    end if

    cardOne=0      : cardTwo=0
    cardOneX=0     : cardTwoX=0
    cardOneY=0     : cardTwoY=0 'reset for next try

    msg$="Turns: ";turns;"          Score: ";score
    #l.g "place 10 420"
    #l.g "\" ; msg$; space$(400)
    if score=12 then gameWon=1 'flag that all pairs are removed
    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(1,1)=11 'jack of clubs
    card(2,1)=12 'queen
    card(3,1)=13 'king
    card(4,1)=24 'jack of diamonds
    card(5,1)=25 'queen
```

```
card(6,1)=26  'king
card(7,1)=37  'jack of hearts
card(8,1)=38  'queen
card(9,1)=39  'king
card(10,1)=50 'jack of spades
card(11,1)=51 'queen
card(12,1)=52 'king

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

```
[shuffleCards]
'first set all cards as visible, card(n,2)=1
for i = 1 to 24
    card(i,2)=1
next

playwave "shuffle.wav",async

'now shuffle cards
for i = 1 to 24
    newIndex=int(rnd(0)*24)+1
    tempCard=card(i,1) 'temp var to allow switching values
    card(i,1)=card(newIndex,1)
'random index now contains value from other index
'random index now contains value from other index
'now card(i,1) 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
```

```
Sub RemoveCard hndle,nC
```

```
'removes a card from screen that was  
'drawn with DealCard, replacing screen background  
callDll #qc, "RemoveCard",hndle as ulong,_  
nC as long,r as void  
End Sub
```

---

[QCard DLL Lesson 8](#) | [Finishing the Memory Card Game](#) | [Duplicate Click Error](#) | [End of Game](#) | [Play Again?](#) | [DEMO](#)

---

[Lesson 7](#) [Lesson 9](#)