
Subject: U++ & MacOS X Carbon

Posted by [mirek](#) on Wed, 18 Apr 2007 12:06:18 GMT

[View Forum Message](#) <> [Reply to Message](#)

Well, I am currently looking at Carbon as guinea pig, therefore I think it is not a bad idea to share...

What we need first to is to be able to draw... DrawRect first. This is where I got:

```
#include <Core/Core.h>
```

```
#include <Carbon/Carbon.h>
```

```
#define kWindowTop 100
```

```
#define kWindowLeft 50
```

```
#define kWindowRight 250
```

```
#define kWindowBottom 250
```

```
const EventTypeSpec  eventList[] =
{
  { kEventClassWindow, kEventWindowClose },
  { kEventClassWindow, kEventWindowActivated },
  { kEventClassWindow, kEventWindowDeactivated },
  { kEventClassWindow, kEventWindowDrawContent },
};
```

```
void MyDrawInWindow (WindowRef window)
{
  CGContextRef myContext;
  SetPortWindowPort (window);// 1
  QDBeginCGContext (GetWindowPort (window), &myContext);
  CGContextSetRGBFillColor (myContext, 1, 0, 0, 1);
  CGContextFillRect (myContext, CGRectMake (0, 0, 200, 100));
  CGContextSetRGBFillColor (myContext, 0, 0, 1, .5);
  CGContextFillRect (myContext, CGRectMake (0, 0, 100, 200));
  CGContextFlush(myContext);// 4
  QDEndCGContext (GetWindowPort(window), &myContext);// 5
}
```

```
static pascal OSStatus MyWindowEventHandler(EventHandlerCallRef nextHandler,
                                           EventRef theEvent,
                                           void *userData)
{
  OSStatus result = eventNotHandledErr;
  WindowRef theWindow = (WindowRef) userData;
  UInt32 whatHappened;
```

```

whatHappened = GetEventKind(theEvent);

switch(whatHappened) {
case kEventWindowClose:
    DisposeWindow(theWindow);
    QuitApplicationEventLoop();
    result = noErr;
    break;
case kEventWindowActivated:
    ::CallNextEventHandler(nextHandler, theEvent);
    result = noErr;
    break;

case kEventWindowDeactivated:
    ::CallNextEventHandler(nextHandler, theEvent);
    result = noErr;
    break;

case kEventWindowDrawContent:
    LOG("PAINT!");
    MyDrawInWindow(theWindow);
    ::CallNextEventHandler(nextHandler, theEvent);
    break;
    }

    return result;
}

void Initialize(void)
{
    // Do one-time-only initialization

    WindowRef          theWindow;
    WindowAttributes   windowAttrs;
    Rect                contentRect;
    EventHandlerUPP    handlerUPP;

    windowAttrs = kWindowStandardDocumentAttributes |
                  kWindowStandardHandlerAttribute;

    SetRect(&contentRect, kWindowLeft, kWindowTop, kWindowRight, kWindowBottom);

    CreateNewWindow(kDocumentWindowClass, windowAttrs,
                   &contentRect, &theWindow);

    SetWindowTitleWithCFString(theWindow, CFSTR("U++ Carbon Example"));
}

```

```

handlerUPP = NewEventHandlerUPP(MyWindowEventHandler);

InstallWindowEventHandler(theWindow, handlerUPP,
                        GetEventTypeCount(eventList), eventList,
                        theWindow, NULL);

ShowWindow(theWindow);

InitCursor();
}

void Finalize(void)
{
}

using namespace UPP;

CONSOLE_APP_MAIN
{
LOG("Hello!");
Initialize(); // Do one-time-only initialization

RunApplicationEventLoop(); //Process events until time to quit

Finalize(); // Do one-time-only finalization
}

```

Looks to do something, but it is jerky on resizing the window.

I have read something about "HIView", but all U++ needs is top-level window to draw on, so I wanted to try this approach (not using HIView) first.

I hope there is some Carbon expert in the community to comment

BTW, I have also encountered strange and funny problem - with U++ allocator (new/delete overrde), debugging facilities in U++ detect problem (most likely write past the end of block) inside Carbon libraries. Of course, something can be wrong with U++ allocator too, but that is unlikely as Win32, X11 and even MacOSX11 versions work without problem. For now, I have used "USEMALLOC" to disable U++ allocator.

Mirek

Subject: Re: U++ & MacOS X Carbon
Posted by [lundman](#) on Thu, 19 Apr 2007 03:48:53 GMT
[View Forum Message](#) <> [Reply to Message](#)

Looks good and behaves good here.

But only after I create a Bundle for it. Otherwise it's somewhat "dead".

Lund
