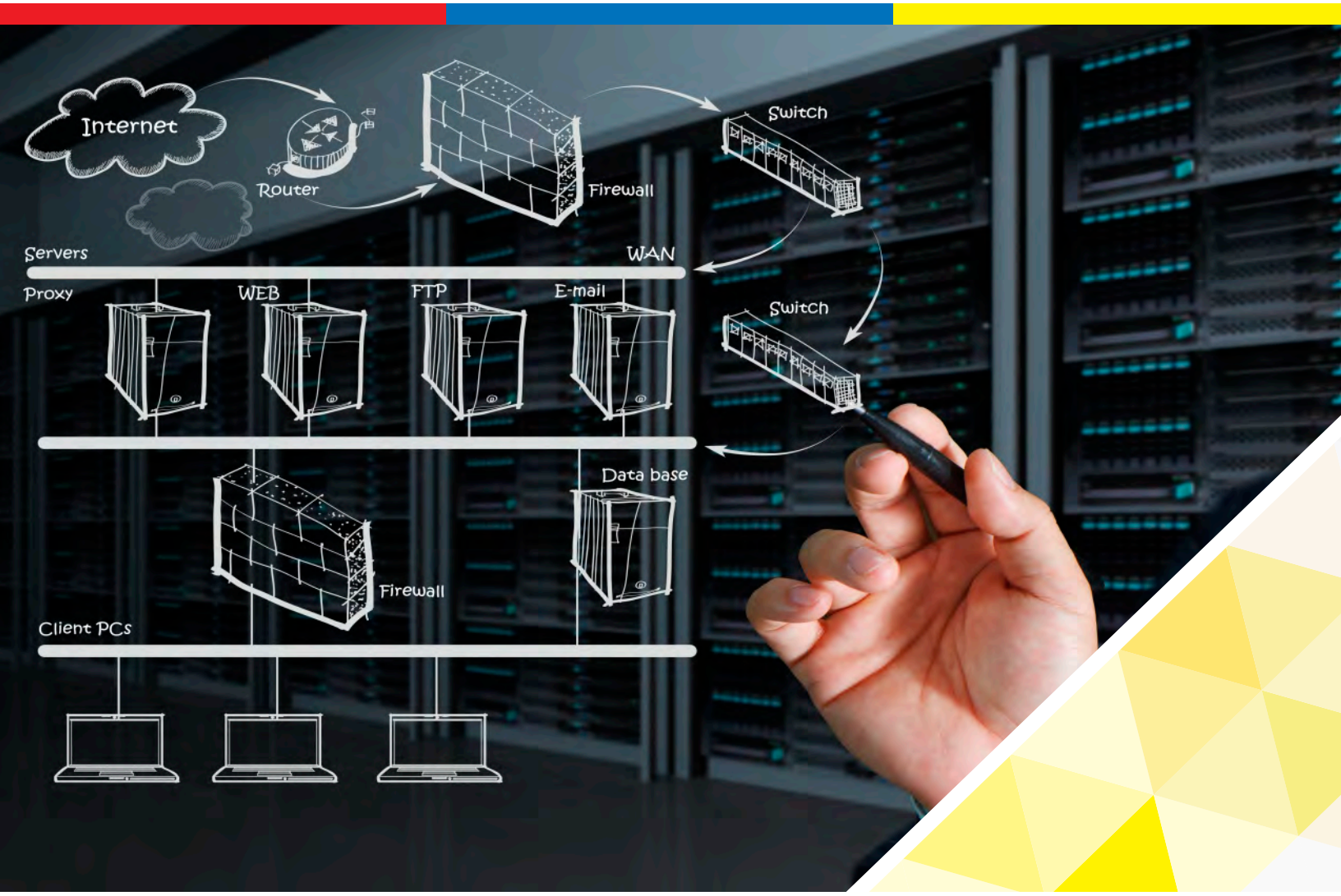
ACL Digital
an ALTEN group company

Innovate, Digitize, Transform

PORTING MFC/WIN32 APPLICATIONS TO MAC OS X

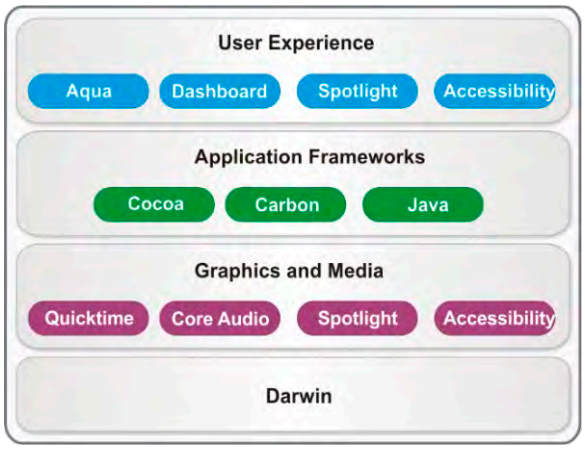


INTRODUCTION TO MAC OS X

Mac OS X supports multiple development technologies like Cocoa object-oriented framework, to compensate J2SE 1.3 and 1.4.2 implementations, of common UNIX tools and scripting languages. Apple provides an advanced XCode Tools, is a software development environment. We have VC++, VB, Javascript and VB script in Visual studio in Windows, similarly XCode in MAC.

All the information about the installed applications are stored in a configuration files, like the registry information in Windows. Mac uses CUPS (Common Unix Printing System) as a default Printer driver. Darwin Kernel environment, BSD libraries and BSD command environment which performs the operations of Kernel, similar to User and GDI in Windows OS. Quartz in Mac OS X is like GDI and GDI+ in Windows which is used for graphics and windowing environment. The OpenGL framework (OpenGL framework) in Mac OS X includes a highly optimized implementation of the OpenGL libraries that provides high-quality graphics which is also available in Windows.

MAC OS X ARCHITECTURE



MAC OS X RUNTIME ARCHITECTURE

A runtime environment is a set of conventions that determines how code and data are loaded into memory and managed. Like MSVCRT, Libc in windows. Mac OS X supports two modern runtime environments: dyld (dynamic loader) and CFM (Code Fragment Manager).Windows provide dlls and libraries (Static & Dynamic) perform the similar operations.

DYLD RUNTIME ENVIRONMENT

The dyld runtime environment is the preferred environment for any development because it is the native environment used by all Mac OS X system libraries. To support this environment, we should build our code modules using the Mach-O executable file format. The dyld library manager is the program responsible for loading your Mach-O code modules, resolving library dependencies and beginning the execution of our code.

CFM Runtime Environment

The CFMruntime environment is the legacy environment inherited from Mac OS 9. The CFM runtime environment expects code modules to be built using the Preferred Executable Format (PEF).

In Windows we can achieve these things using single threading, multiple threading, COM, ATL, dll's etc,

DEVELOPMENT ENVIRONMENT

Xcode

Xcode is the engine that powers Apple's integrated development environment (IDE) for Mac OS X as Visual studio in Windows. XCode provides a flexible tree listing, a code editor with syntax highlighting and Microsoft-like IntelliSense to automatically insert method definitions for easy coding. XCode doesn't have tabbed window like visual studio instead it should be opened in new window.

Using XCode we can build projects containing source code written in C, C++, Objective-C, Objective-C++, and Java. It generates executables of all supported types on Mac OS X, including command-line tool's, frameworks, plug-ins, kernel extensions, bundles, and applications.

CARBON

Carbon application environment contains set of C APIs used to create full-featured applications for all types of users. The Carbon environment includes support for all the standard Aqua user interface elements such as Windows, controls, and menus, which we can design using Interface Builder. It also provides an extensive infrastructure for handling events, managing data, and using system resources. Like MFC (Microsoft Foundation Classes) in Windows environment.

The Carbon application environment comprises of several key frameworks, Carbon framework (Carbon. Framework), CoreServices framework (CoreServices. framework), Application Services framework (Application Services. framework). Three types of projects can be created using carbon they are Carbon Application, Carbon C++ Application, Carbon C++ Standard Application and Carbon Dynamic Library.

GENERAL CARBON API'S AND ITS EQUIVALENT WIN32 or MFC API'S

Rendering API's in Carbon

CGContextAddArc, CGContextAddArcToPoint, CGContextAddCurveToPoint, CGContextAddLines, CGContextAddLineToPoint, CGContextAddPath, CGContextAddQuadCurveToPoint, CGContextAddRect, CGContextAddRects, CGContextBegin Path, CGContextClosePath, CGContextMoveToPoint, CGContextAddEllipseInRect, CGContextSaveGState, CGContextRestoreGState

MFC equivalent API's

CDC::Moveto, CDC::Lineto, CDC::Rectangle, CDC::Ellipse, CDC::Arc, CDC::ArtTo, CDC::FillRect, CDC::Polygon, CDC::FloodFill, CDC::Pie etc. CDC::SaveDC, CDC::RestoreDC

View

HIScrollViewCreate() - Creates a scroll view. OSStatus HIScrollViewCreate (OptionBits inOptions,HIViewRef outView);*

Parameters

inOptions - Options for our scroll view. You must specify either a horizontal or a vertical scroll bar. If neither is passed, an error is returned. For possible values, see "Scroll View Constants". outView - The new scroll view.

Return Value

An operating system result code.

Other Related Scroll View Functions

```
//Obtains current setting of a scroll view's scroll bar
//auto-hide setting.
HIScrollViewGetScrollBarAutoHide
//Sets a scroll view's auto-hide setting.
HIScrollViewSetScrollBarAutoHide
//Determines whether it is possible to navigate in a scroll view.
HIScrollViewCanNavigate
HIScrollViewNavigate //Changes the portion of a view's target.
```

B. Rendering Images

```
CGContextDrawImage - Draws an image into a graphics context.
void CGContextDrawImage (CGContextRef context, CGRect rect, CGImageRef image);
```

Parameters

Context

The graphics context in which to draw the image.

Rect

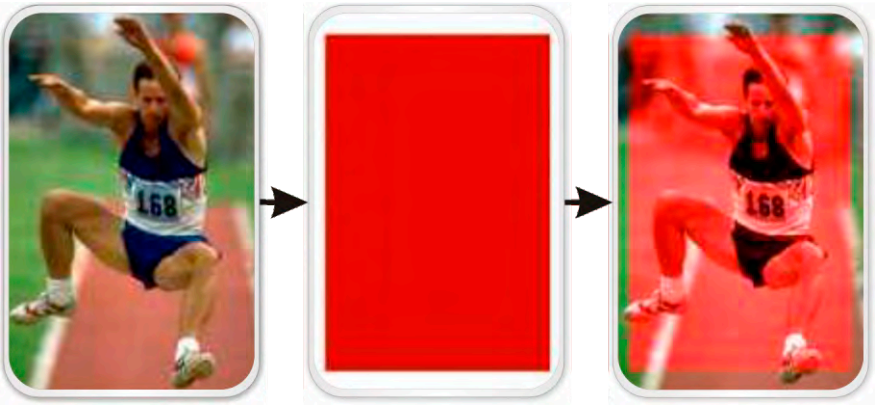
The location and dimensions in user space of the bounding box in which to draw the image.

Alpha Blending Colorizing an Image

One way that you can use the color blend mode is to colorize an image. Draw the image you want to colorize. Then set the blend mode by passing the constant kCGBlendModeColor to the function CGContextSetBlendMode. Draw and fill a rectangle (or other shape) using the color you want to use for colorizing the image. The code in Listing draws a fully opaque red rectangle over the image of the jumper,to achieve the result shown on the right side of the figure. Note that the entire image of the jumper is not colorized because the red rectangle is smaller than the image.

The Following code will render the image with blending option.

```
CGContextSaveGState (context);
CGContextDrawImage(context, myRect1, image);
CGContextSetBlendMode(context, kCGBlendModeColor);
CGContextSetRGBFillColor (context, 0.8, 0.0, 0.0, 1.0);
CGContextFillRect (conte.xt, myRect2);
CGContextSaveGState (context);
```



Similar Alpha Blending in Visual C++

```
Cbitmap bmpRed;
bmpRed.LoadBitmap(IDB_RED);
mdcPicture.SelectObject(&bmpRed);
BITMAP bm;   bmpRed.GetBitmap(&bm);
BLENDFUNCTION bf;
bf.AlphaFormat =0;
bf.BlendFlags =0;
bf.BlendOp =0;
bf.SourceConstantAlpha = 40;
AlphaBlend(pDC->m_hDC, rectBitMAP.left, rectBitMAP.top,
rectBitMAP.Width(), rectBitMAP.Height(), mdcPicture.m_hDC, 0, 0,
bm.bmWidth, bm.bmHeight, bf);
pDC->SelectObject(&rPen);
pDC->Rectangle(rectBitMAP);
```

This sample will show the first page after Alpha Blending in windows and next page with normal display.

Function for **Alpha Blending** is available in CImage::AlphaBlend which is present in Windows GDI platform SDK used in vc++ application on windows environment. The **BLENDFUNCTION** structure controls blending by specifying the blending functions for source and destination bitmaps.

Saving and Restoring the Current Graphics State

//Pushes a copy of the current graphics state on the

//top of a context's graphics state stack.

CGContextSaveGState

//Sets the current graphics state to the state

//most recently saved

CGContextRestoreGState

VC++ equivalent Function

//Saves the current state of the device context by copying

//state information (such as clipping region, selected objects,

and mapping mode) to a context stack maintained by Windows.

CDC::SaveDC

Restores the device context to the previous state identified DC

CDC::RestoreDC -

Converting Between Device Space and User Space

// Transforms a point from user space coordinates to

// device space coordinates.

CGContextConvertPointToDeviceSpace

// Transforms a point from device space coordinates

// to user space coordinates.

CGContextConvertPointToUserSpace

// Transforms a rectangle from user space coordinate to

// device space coordinates

// Transforms a rectangle from user space coordinate

// to user space coordinates

CGContextConvertRectToUserSpace

CGContextConvertSizeToDeviceSpace

CGContextConvertSizeToUserSpace

Other Related Rendering Functions

Functions related with Setting Color, Color Space, and Shadow Values

```
// Sets the opacity level for objects drawn in a graphics context.
CGContextSetAlpha
// Sets the current fill color to a value in the DeviceCMYK
// color space.
CGContextSetCMYKFillColor
CGContextSetFillColor // Sets the current fill color.
// Sets the current stroke color to a value in the DeviceCMYK
// color space
CGContextSetCMYKStrokeColor
CGContextSetFillColorSpace,CGContextSetFillColorWithColor,
CGContextSetGrayFillColor,CGContextSetGrayStrokeColor,
CGContextSetRGBFillColor,CGContextSetRGBStrokeColor,
CGContextSetShadow,CGContextSetShadowWithColor,
CGContextSetStrokeColor,CGContextSetStrokeColorSpace,
CGContextSetStrokeColorWithColor
```

Other Related Rendering Functions

```
CDC::BitBlt,CDC::StretchBlt,CDC::LoadBitmap,CDC::TransparentBlt
// Converts the client coordinates of a given point or rectangle
// on the display to screen coordinates.
CWnd::ClientToScreen();
// Converts the screen coordinates of a given point or rectangle
// on the display to client coordinates.
CWnd::ScreenToClient();
```

Example

```
Crect myRect;
GetClientRect(&myRect);
ClientToScreen(myRect);
MoveWindow(myRect.left, myRect.top,myRect.Width(), myRect.Height());
```

ToolBar Functions in VC++

```
CToolBarCtrl::Create,CToolBarCtrl::LoadToolbar,
CToolBarCtrl::GetItemID,CToolBarCtrl::SetBitmap,
CToolBarCtrl::GetButtonStyle,CToolBarCtrl::SetButtons,
CToolBarCtrl::LoadImages
```

ToolBar Functions in MAC

HIToolbarCreate,HIToolbarItemCreate,SetWindowToolbar,ShowHideWindowToolbar,ChangeWindowAttributes,CreateToolbarItemForIdetifier,HIToolbarItemSetCommandID,HIToolbarItemSetIconRef

Menu Creation Functions in VC++

CMenu::CreateMenu,CMenu::CreatePopupMenu,
CMenu::LoadMenu,CMenu::Des

Similar function for creating Menu in Carbon

CreateNewMenu, DuplicateMenu, RetainMenu, ReleaseMenu, DisposeMenu,
CreateCustomMenu, RegisterMenuDefinition, SetMenuDefinition,
GetMenuDefinition, CreateStandardFontMenu, UpdateStandardFontMenu

Menu Creation Functions in VC++

CWnd::Create,CWnd::ShowWindow,CWnd::BringWindowToTop,
CWnd::MoveWndow CWnd::DestroyWindow

MAC equivalent Window Function

CreateNewWindow,CreateCustomWindow,DisposeWindow,CreateWindowFromResource, ActivateWindow

File Operation in MAC

FSCreateFileUnicode(), FSCreateFork(), FSOpenFork(), FSReadFork() , FSWriteFork()

C equivalent functions

FILE, fopen, fclose, fwrite, fread (stdio.h)

Datatypes in MAC

Sint32,UInt32,Str255,float,OSType,CFRange,Boolean,CFString,UString,

Equivalent Datatype in VC++

Int,float,char,double,CString,Bool*

FileHandling functions can be used commonly both vc++ and in carbon if it is based on C language as given

malloc, calloc, realloc, free(alloc.h) memcpy, memset, strcpy,
strcat, strcmp, strstr, strlen atoi, itoa, sprintf

COCOA

Cocoa is an advanced object-oriented development platform in Mac OS X. Cocoa is a set of frameworks used for the development of full-featured applications in the Objective-C language. It is based on the integration of OpenStep, Apple technologies, and Java. Cocoa consists of two faces, Runtime Aspect and Development Aspect. The Cocoa application environment consists of two object-oriented frameworks:

Cocoa is an advanced object-oriented development platform in Mac OS X. Cocoa is a set of frameworks used for the development of full-featured applications in the Objective-C language. It is based on the integration of OpenStep, Apple technologies, and Java. Cocoa consists of two faces, Runtime Aspect and Development Aspect. The Cocoa application environment consists of two object-oriented frameworks:

- **Foundation Framework Classes** : Implements data management, file access, process notification, memory management, network communication, and other low-level features.
- **Application Kit framework Classes** : Implements the user interface layer of an application, including windows, dialogs, controls, menus, and event handling.
- **Cocoa framework** : Imports both Foundation and the Application Kit. If we are writing a Cocoa program that does not have a graphical user interface (a background server, for example), we can link the program with the Foundation framework.

Using Cocoa we can create different types of applications like Cocoa Application, Cocoa Document-based Application, Cocoa-Java Application and Cocoa-Java Document-based Application like win32 application, Dialog based application, Document-view Application in VC++ & MFC.

Objective C

An object-oriented programming language based on standard C and a runtime system that implements the dynamic functions of the language. Objective-C is extension to the C language which is based on Smalltalk, one of the first object-oriented programming languages. Objective-C is available in the Cocoa application environment. The Objective-C language is a superset of ANSI C with special syntax and run-time extensions that make object-oriented programming possible.

CREATING AN EXECUTABLE IN MAC

In Windows the Executable files are .EXE file and for the Mac it is a .DMG file. DMG Files can be created with Disk Copy, burnt to CD or mounted as a normal volume. If we have a .DMG file on a Windows PC it won't be able to use. But in Windows the default output in will be *.exe format.

1. Create a New Folder in MAC and place the files that should be created in disk image into this new folder.
2. Right click (or CTRL-Click) the folder and select "Get Info" and note the size of its contents.
3. Open Disk Utility (Applications > Utilities > Disk Utility)
4. Click the "New Image" icon to create a new disk image. Enter a name for the Image, and select a size adequate for the size of your folder you created in Step 2. Set the encryption to "none" and Format to "read/write disk image".
5. Place the contents of the folder from Step 2 into the newly mounted disk image
6. Unmount the Disk Image.

For creating setup we are using Install shield in Windows. Install Shield itself contain separate version for MAC OS X and some other software's like InstallJammer is also available to create setup in MAC OS X.

CONCLUSION

There is no direct porting facility between this operating systems. For example C,C++ developers will like to work on Carbon, Java programmers or Visual Basic developers in Java or Cocoa, Unix based programmers in BSD or X11 and Applescript for scripting language. Though they are using one common developing environment called Xcode.

References

» http://www.apple.com/macosx/

» http://www.kernelthread.com/mac/osx/

» http://www.cocoadevcentral.com/

» http://www.developers.apple.com/documentation