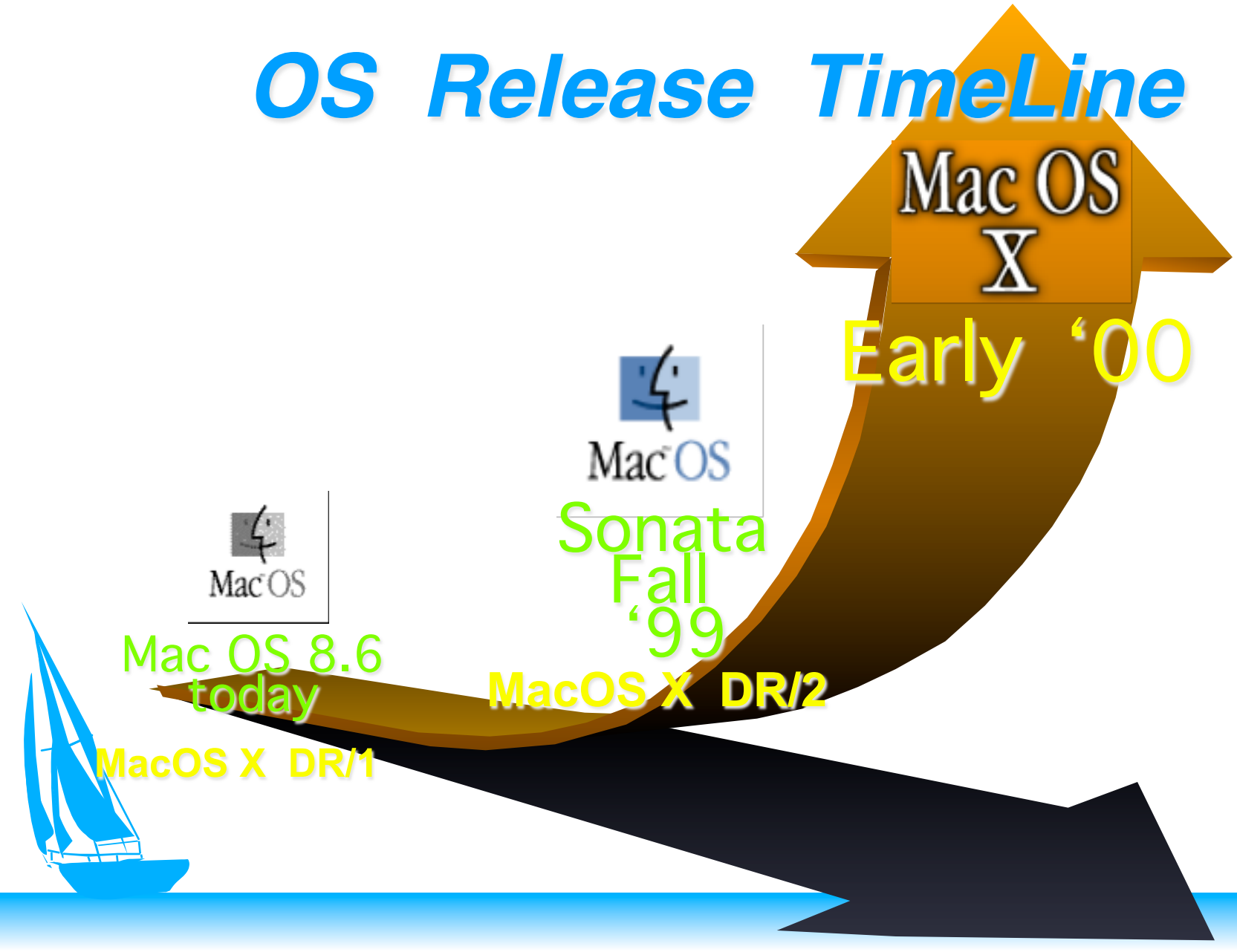
Porting to the Carbon API

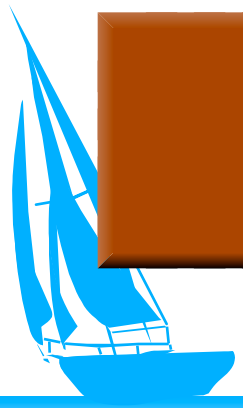
for Macintosh Developers

Jonathan Hoyle
Eastman Kodak
5/20/99

OS Release TimeLine



Mac OS X Architecture



Classic
(formerly Blue Box)

Carbon

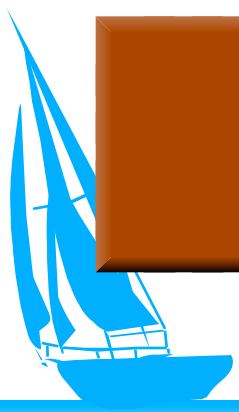
Cocoa
(formerly
Yellow Box)

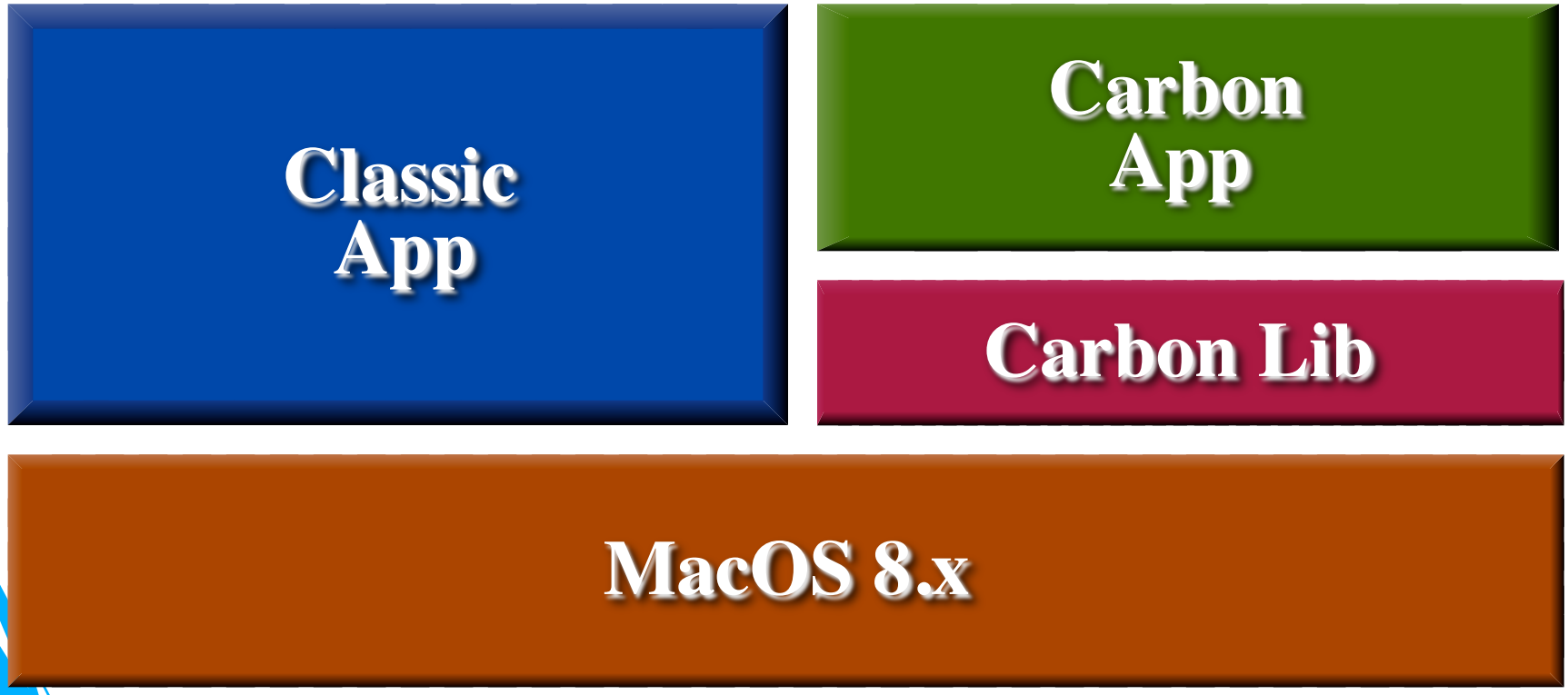
Java

Common Services

Core OS

Mac OS 8.x Architecture





Carbon Delivery

- Built into MacOS X
- Built into Sonata
- CarbonLib shared library for MacOS 8.1+
- PowerPC only
- Carbon apps will look like any Classic app on MacOS 8.x
- “Lite CarbonLib” available now
- CodeWarrior 5 supports Carbon (June ‘99)

WWDC '99 Tools

MacOS X DR/1

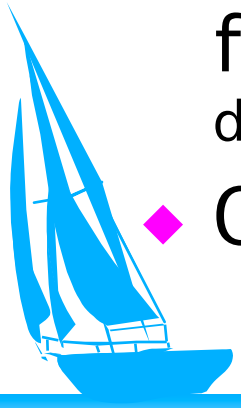
- Carbon SDK
- LiteCarbonLib
- Sample code & utilities

CW for MacOS X & Carbon

- CW 4->4.1 Updater
- Carbonated PP & MSL
- CFM & Mach-O Carbon compilers
- samples & other goodies

Carbon Technical Issues

- Carbon apps use new exe format
- Two flavors of apps: CFM and Mach-O
- PowerPlant apps must use Carbonized PP
- Carbon apps can assume 8.1 as baseline
- Carbon apps still must check OS version for later calls. (For example, CarbonLib on 8.1 doesn't support new 8.5 calls.)
- Carbon apps can print to Classic drivers



Carbon Technical Issues

All Carbon apps have access to:

- Everything in 8.1
- Navigation Services
- CoreFoundation “classes”:

  CFString, CFBoolean, CFArray, CFSet, CFPreferences, CFPlugin, etc.
- Carbon Events

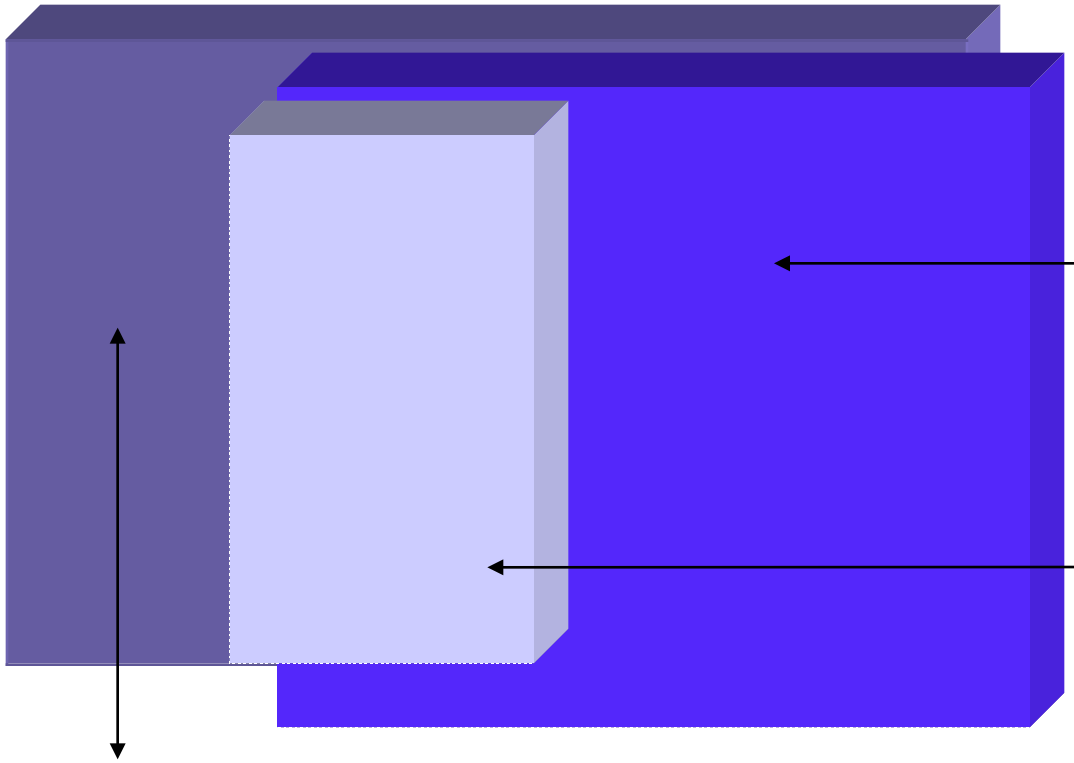
Classic Applications

- No longer inside a “box”
- Receive little benefit from MacOS X
- Not preemptive, share same memory
- All 68K apps are classic
- Apps compiled in the older binary format are classic, even if the source code is Carbon compliant.
- A “fat” format (if any) is TBD.

What About Cocoa?

- Mac OS X only
- Not available for Sonata
- No CocoaLib for MacOS 8.x
- No longer committing to Windows support
- OpenStep API
- No CFM, Mach-O executable type only
- Requires Objective C or Java

Carbon API's



Carbon:
3/4 of the ToolBox

A typical app uses > 90% Carbon API's

8200 Mac ToolBox Calls

Gone for Carbon

- Standard File (use Nav Services)
- Appletalk (exists only at the Core OS)
- QuickDraw GX
- QuickDraw 3D (use OpenGL)
- Balloon Help (use new Carbon Help)
- MFS File System, Working Directories
- Edition Manager, themes, sundry others
- Low Memory, Segment Loader, 68K-isms

Support in Carbon

- Common Managers: QuickDraw, Window, etc.
- Apple Events
- Open Transport
- Printing API (modified)
- Game Sprockets
- Dictionary Manager (partially supported)
- SCSI Manager (partially supported)
- Other HW: Use Dev Mgr on 8.x, IOKit for Mac OS X

Not Available in DR/1

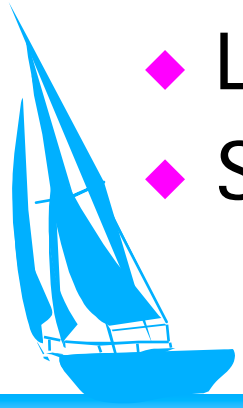
- QuickTime
- Sound Manager
- Speech Manager
- Display Manager
- AppleScript
- ColorSync
- TextServices Manager
- Text Encoding

Steps in Porting

- *Carbon Overview* video from WWDC '99
- Use CarbonDater on current PPC app
- Use Carbonized Universal Headers
- `#define TARGET_CARBON 1`
- Link with CarbonLib
- Make sure all other lib's are Carbonized
- Replace Low Memory accesses
- Handle Opaque Data Structures

What structs are opaque?

- WindowRecords, DialogRecords, etc.
- Menus
- Controls
- Ports
- Regions
- Lists
- See Table 2-2 in *Carbon Porting Guide*



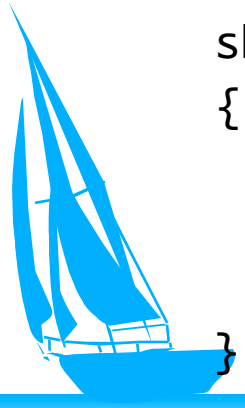
Opaque Data Structures

In MacOS 8, Regions are transparent:

```
short getLeftCorner(RgnHandle inRgnHdl)
{
     return (**inRgnHdl).rgnBBox.left;
}
```

In MacOS X, Regions are opaque:

```
short getLeftCorner(RgnHandle inRgnHdl)
{
     Rect     rgnRect;
     GetRegionBounds(inRgnHdl, &rgnRect);
     return rgnRect.left;
}
```



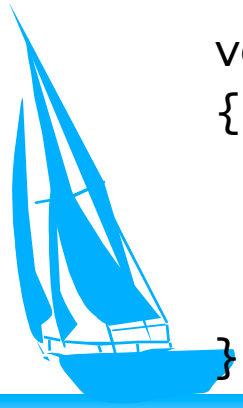
Opaque Data Structures

In MacOS 8, GrafPtr = WindowPtr = DialogPtr:

```
void DrawLine(WindowPtr inWindowPtr)
{
     SetPort(inWindowPtr);
     LineTo(100, 100);
}
```

In MacOS X, you must use accessors:

```
void DrawLine(WindowPtr inWindowPtr)
{
     GrafPtr  theGrafPtr = GetWindowPort(inWindowPtr);
     SetPort(theGrafPtr);
     LineTo(100, 100);
}
```



Think different.

99 Worldwide Developers Conference

Q & A