

# BREW Display Framework

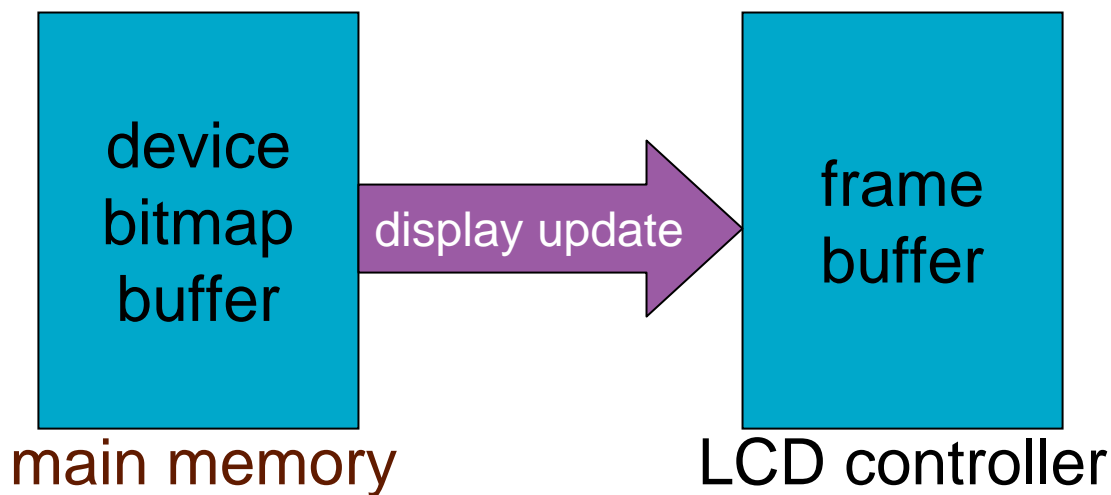
Mark Baysinger, Staff Engineer

*QUALCOMM Incorporated*



# Device Bitmap Buffer

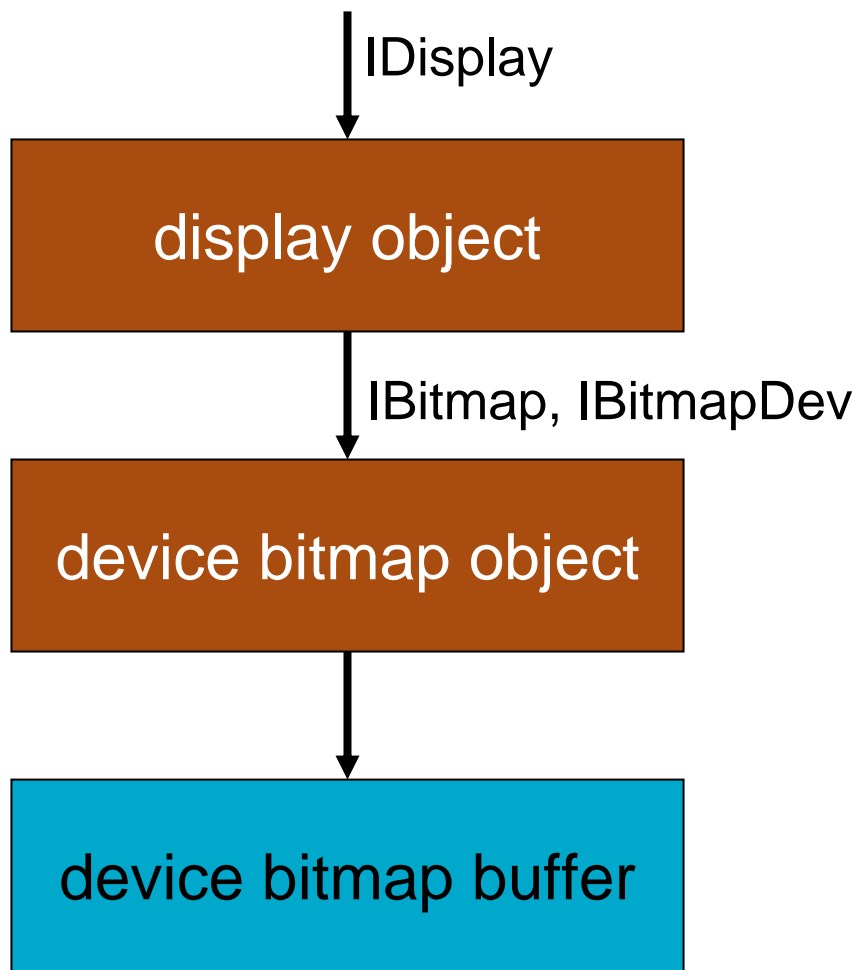
- One device bitmap buffer per display
- Apps render into the device bitmap buffer
- A display update copies the contents of the device bitmap buffer to the frame buffer



# Bitmap Objects and Display Objects

- **Bitmap objects**
  - Low level drawing operations
  - Abstraction of device bitmap and off screen bitmaps
- **Display objects**
  - Higher level drawing operations
  - Implemented in terms of calls to a bitmap object
  - Device bitmap objects are obtained via display objects

# Bitmap Objects and Display Objects



# Display Objects

- **IDisplay is the interface to a display object**
- **There may be many display objects**
- **Each app has a default display object created for it, and this display object is associated with the primary display**

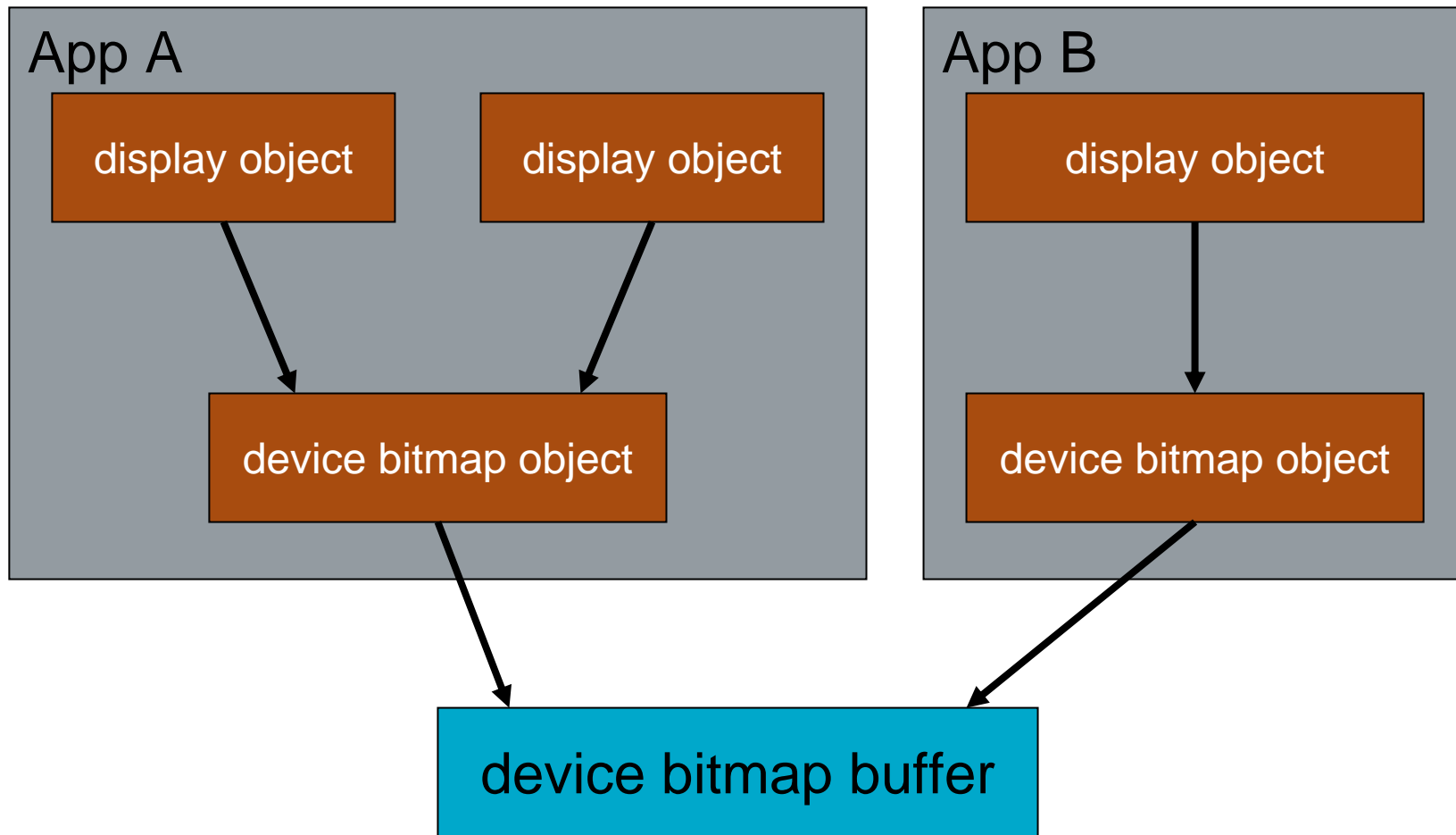
# IDisplay Example

```
// Get the default display object  
  
ISHELL_CreateInstance(piShell, AEECLSID_DISPLAY,  
(void **)&piDisplay);  
  
  
// Draw a rectangle to the device bitmap buffer  
  
SETAEERECT(&rc, 10, 10, 10, 10);  
  
IDISPLAY_FillRect(piDisplay, &rc, RGB_BLACK);  
  
  
// Update the frame buffer  
  
IDISPLAY_Update(piDisplay);
```

## Display Objects (cont.)

- **Create new display objects with**
  - `IDISPLAY_Clone()`
  - `AEECLSID_DISPLAYCLONE`
  - `AEECLSID_DISPLAY1, 2, etc. (2.1)`
  - `AEECLSID_DISPLAY_NULL (3.1)`
- **Get the destination bitmap with `IDISPLAY_GetDestination()`**
- **Change the destination bitmap with `IDISPLAY_SetDestination()`**

# Multiple Apps / Multiple Display Objects



# Bitmap Objects

- **Bitmap objects implement the IBitmap interface**
- **Any display object can be associated with any bitmap object via IDISPLAY\_SetDestination()**
- **Relatively little state, compared to display objects**

# Device Bitmap Objects

- **Additionally implements IBitmapDev interface**
- **Permanently associated with a display**
- **Renders to device bitmap buffer**
- **One per app per display**

# Using IBitmap and IBitmapDev Directly

```
// Obtain the device bitmap  
  
piBitmap = IDISPLAY_GetDestination(piDisplay);  
  
  
// Draw a rectangle  
  
c = IBITMAP_RGBToNative(piBitmap, RGB_BLACK);  
SETAEEERECT(&rc, 10, 10, 10, 10);  
IBITMAP_FillRect(piBitmap, &rc, c, AEE_RO_COPY);  
  
  
// Continued...
```

# Using IBitmap and IBitmapDev Directly

```
// Get the IBitmapDev interface (BREW 2.1)
IBITMAP_QueryInterface(piBitmap, AEEIID_BITMAPDEV,
(void **)&piBitmapDev);
```

```
// Update the display

IBITMAPDEV_Update(piBitmapDev);
```

```
// Clean up

IBITMAP_Release(piBitmap);

IBITMAPDEV_Release(piBitmapDev);
```

# Classes of Bitmap Objects

- **Device compatible bitmaps**
  - Device bitmap object
  - Bitmaps created from other device compatible bitmaps
- **Simple device independent bitmaps (simple DIBs)**
  - `IDISPLAY_CreateDIBitmap()` and `CreateDIBitmapEx()`
  - `ISHELL_LoadBitmap()` and `LoadResBitmap()`
  - Bitmaps created from other simple DIBs

# Tradeoffs Between Bitmaps Classes

- **Device compatible bitmap objects**
  - Implement all IBitmap drawing methods
  - Limited to display's native format
  - Bitmap properties (width, height, etc.) cannot be modified
- **Simple DIB objects**
  - Do not implement any IBitmap drawing methods
  - Can represent a wide range of bitmap formats
  - Bitmap properties can be modified via IDIB

# IBitmap Drawing Methods

- **IBITMAP\_DrawPixel()**
- **IBITMAP\_SetPixels()**
- **IBITMAP\_DrawHScanline()**
- **IBITMAP\_FillRect()**
- **IBITMAP\_BltIn()**
  
- **Most drawing methods take a NativeColor parameter**
- **All drawing methods take an AEERasterOp**

## Other IBitmap Methods

- **IBITMAP\_RGBToNative()**
- **IBITMAP\_NativeToRGB()**
- **IBITMAP\_GetPixel()**
- **IBITMAP\_BltOut()**
- **IBITMAP\_GetInfo()**
- **IBITMAP\_CreateCompatibleBitmap()**
- **IBITMAP\_SetTransparencyColor()**
- **IBITMAP\_GetTransparencyColor()**

# Two Blit Methods

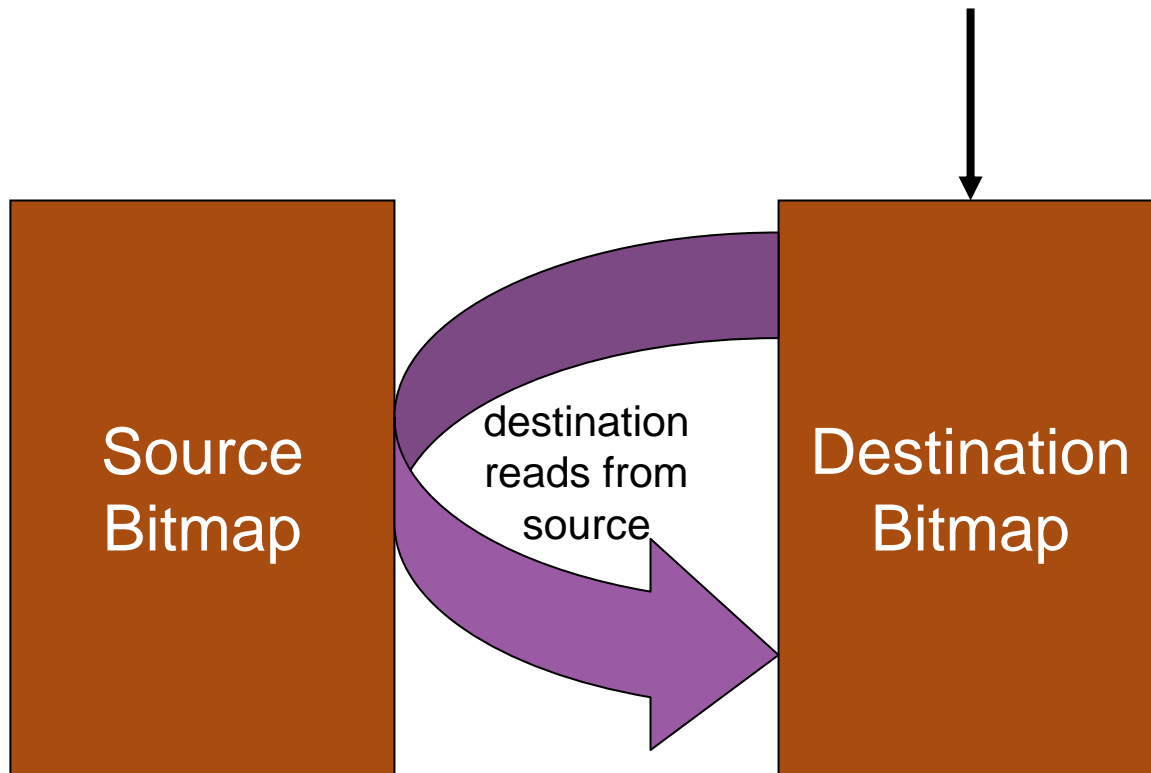
```
int IBITMAP_BlItIn(IBitmap *po, int xDst, int yDst,  
                  int dx, int dy,  
                  IBitmap *pSrc, int xSrc, int ySrc,  
                  AEERasterOp rop);
```

```
int IBITMAP_BlItOut(IBitmap *po, int xDst, int yDst,  
                   int dx, int dy,  
                   IBitmap *pDst, int xSrc, int ySrc,  
                   AEERasterOp rop);
```

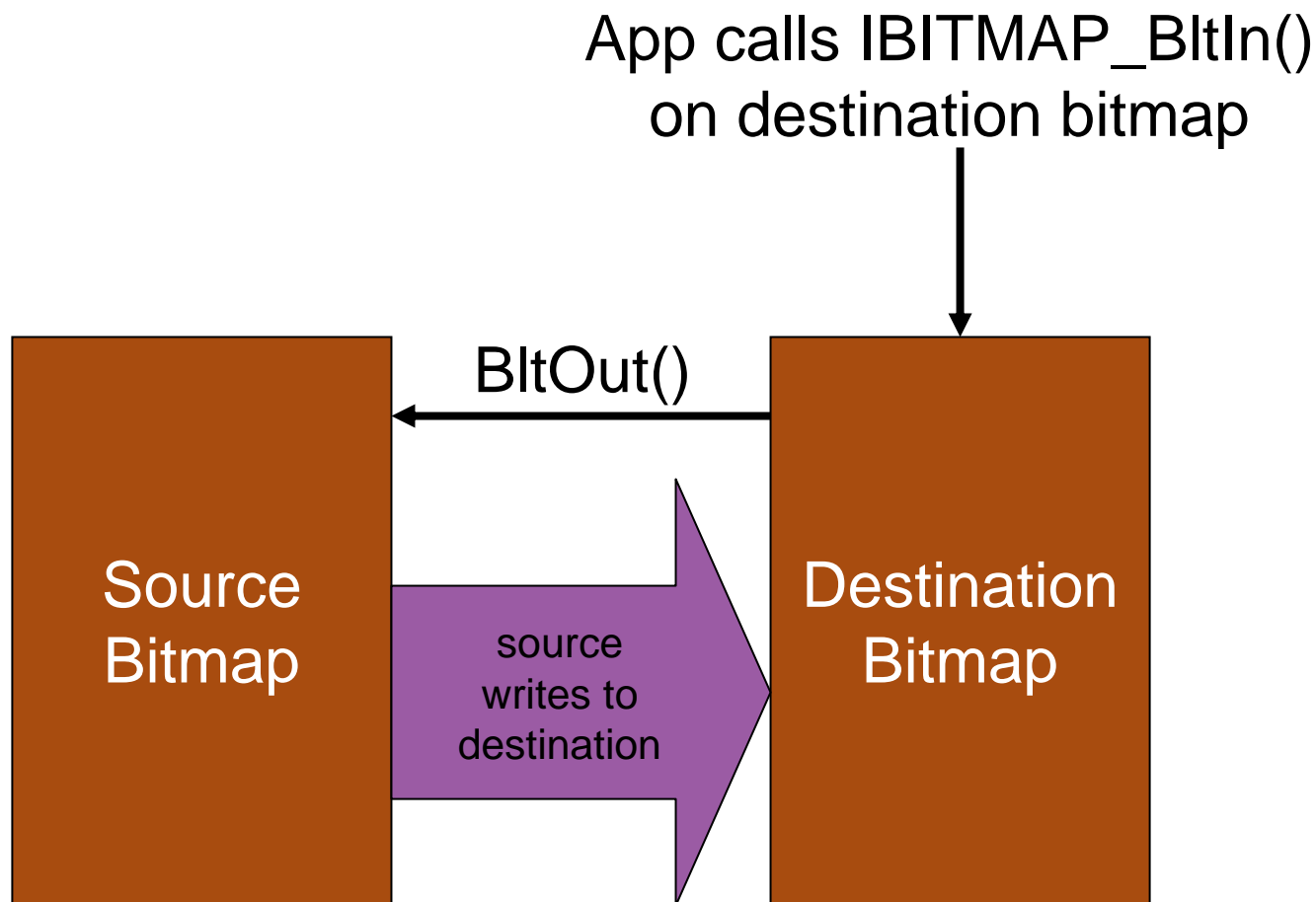
Always call IBITMAP\_BlItIn()!

# Destination Implements Blit

App calls `IBITMAP_BltIn()`  
on destination bitmap



# Source Implements Blit



# IDIB

- **Interface is obtained by calling QueryInterface() on a bitmap object**
- **All simple DIBs implement IDIB**
- **Device compatible bitmaps implement IDIB on almost all devices**

## IDIB (cont.)

- **IDIB is a strange interface**
  - Contains public data members
  - Implements IBitmap vtable

```
struct IDIB {  
    AEEVTBL(IBitmap) *pvt;  
    IQueryInterface * pPaletteMap;  
    byte *           pBmp;  
    uint32 *        pRGB;  
    NativeColor     ncTransparent;  
    uint16          cx;  
    uint16          cy;  
    int16           nPitch;  
    uint16          cntRGB;  
    uint8           nDepth;  
    uint8           nColorScheme;  
    uint8           reserved[6];  
};
```

# IDIB Tips

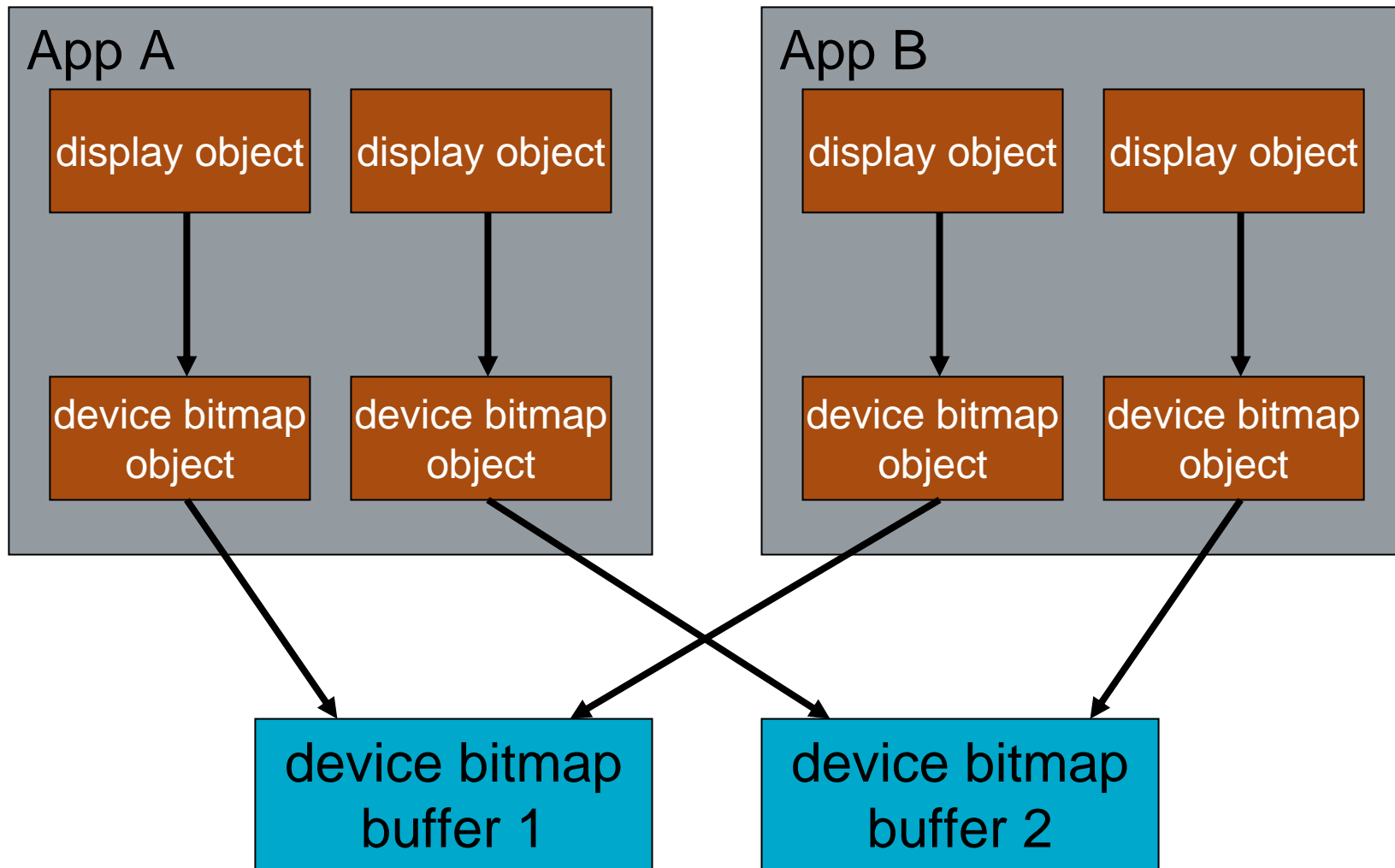
- **IDIB\* may be cast to IBitmap\*:**

```
IBITMAP_BltIn(piBitmap, ..., IDIB_TO_IBITMAP(pidIB), ...);
```

- **Don't change the fields in the IDIB struct**
  - Except for simple DIBs
- **Always invalidate after changing bitmap memory directly:**

```
IBITMAP_Invalidate(IDIB_TO_IBITMAP(pidIB), &rc);
```

# Multiple Displays



## Multiple Displays

- **Secondary displays work just like the primary display (for the most part)**
- **A secondary display has its own device bitmap buffer**
- **Create a new display object associated with a particular display via AEECLSID\_DISPLAY1, AEECLSID\_DISPLAY2, etc.**
- **Change the default display with IDISPLAY\_MakeDefault()**

# Display Access Control

- **Check whether you have access to a display via `IDISPLAY_IsEnabled()` or `IBITMAPDEV_IsEnabled()`**
- **Ask for notification via `IDISPLAY_NotifyEnable()` or `IBITMAPDEV_NotifyEnable()`**
- **Drawing operations and display updates become no-ops when you do not have access to a display**

## Display Access Control (cont.)

- **Primary display access always goes to the top visible app**
- **Secondary display access must be requested**
  - Request by creating a display object for that display
  - Display stays requested until device bitmap object is destroyed
  - Request is granted based on priority
    - Top visible app has highest priority
    - Background apps have priority behind top visible app, in order of when they entered the background state
    - Recently backgrounded apps win over less recently backgrounded apps

**Thank you!**