IUP Next

New Modern Backends for the Cross-Platform Native GUI Library

Mac/Cocoa       iOS/CocoaTouch       Android       Web/DOM

Eric Wing
blurrrsdk.com
@ewingfighter / @BlurrrSDK

Chris Matzenbach
cmatzenbach@gmail.com

Lua Workshop 2017
My Background:
Worn lots of hats
Globalstar: Satellites & Rockets

• Global communication system based on constellation of satellites and ground stations

• Launch satellites into space with rockets!

• (Not relevant for this talk, but I’m told it sounds cool)
From Cross-platform to Native Cocoa

• Cross-platform, Scientific Visualization

• End of the Unix Wars => Microsoft Windows domination

• Mac OS X: A Unix with a user-friendly UI

• Meld cross-platform OpenGL sci-viz with native Cocoa UI for best experience
LuaCocoa

- Wrote world’s first **full-featured** bridge between Lua & Cocoa
  - Obj-C runtime + libffi + Mac OS X 10.5 BridgeSupport
  - Complete API coverage including C APIs
  - Dual mode: Obj-C garbage collection & traditional
  - PowerPC/Intel, 32-bit/64-bit Universal Binaries
Beginning iPhone Games Development

Essential Guide for all iPhone and iPod touch Game Developers

Peter Bakhirev | PJ Cabrera | Ian Marsh | Scott Penberthy
Ben Britten Smith | Eric Wing

Apress
Commercial Game Engines

Corona SDK (Lua)

• Primary platforms: iOS & Android

• Also: Mac & Windows

Platino (JavaScript)
Cross-platform Native App Dev Made Simple

Native application development is harder than it should be. Let’s fix that.
IUP (Portable User Interface)

• Cross-platform **Native** GUI library

• GUI-only (not bloated kitchen sink)

• From PUC-Rio (same as Lua)

• MIT License

• Current Active Backends:
  • Windows
  • GTK2 & GTK3
  • Motif
  • Haiku
Why IUP?
Let me try to paint a picture with my story

- Needed cross-platform native-ish GUI tools for Blurrrr SDK (blurrrsdk.com)
- Launcher to generate native IDE projects
Game GUIs possible, but not ideal for requirements

Blurrr SDK Particle Editor made with Nuklear (game) GUI
The Usual Suspects

- Blurrr SDK supports dev on Windows, Mac, Linux, Raspberry Pi
  - Make apps for Windows, Mac, Linux, Pi, iOS, Android
  - wxWidgets, Qt, Java, Tk, NodeWebKit, etc.
- Decided to try QtQuick
## RAM Usage Comparison

### Mac OS X 10.12

<table>
<thead>
<tr>
<th>Process Name</th>
<th>Threads</th>
<th>Compressed Mem</th>
<th>Ports</th>
<th>PID</th>
<th>User</th>
<th>Memory</th>
<th>Real Mem</th>
<th>Private Mem</th>
<th>Shared Mem</th>
<th>Purgeable Mem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dock</td>
<td>3</td>
<td>96.2 MB</td>
<td>56</td>
<td>445976</td>
<td>ewing</td>
<td>76.1 MB</td>
<td>22.6 MB</td>
<td>22.6 MB</td>
<td>0 bytes</td>
<td>0 bytes</td>
</tr>
<tr>
<td>loginwindow</td>
<td>2</td>
<td>26.0 MB</td>
<td>695</td>
<td>110</td>
<td>ewing</td>
<td>35.3 MB</td>
<td>26.6 MB</td>
<td>20.3 MB</td>
<td>14.2 MB</td>
<td>0 bytes</td>
</tr>
<tr>
<td>SystemUIServer</td>
<td>1</td>
<td>14.5 MB</td>
<td>182</td>
<td>452</td>
<td>ewing</td>
<td>49.9 MB</td>
<td>15.8 MB</td>
<td>8.0 MB</td>
<td>7.0 MB</td>
<td>0 bytes</td>
</tr>
<tr>
<td>notessend</td>
<td>1</td>
<td>17.2 MB</td>
<td>329</td>
<td>50877</td>
<td>ewing</td>
<td>33.3 MB</td>
<td>14.8 MB</td>
<td>22.9 MB</td>
<td>0 bytes</td>
<td>0 bytes</td>
</tr>
<tr>
<td>SystemUIServer.xpc</td>
<td>1</td>
<td>16.8 MB</td>
<td>454</td>
<td>381</td>
<td>ewing</td>
<td>32.8 MB</td>
<td>13.8 MB</td>
<td>19.5 MB</td>
<td>12 KB</td>
<td>0 bytes</td>
</tr>
<tr>
<td>SystemUIServer.xpc</td>
<td>1</td>
<td>17.4 MB</td>
<td>343</td>
<td>28746</td>
<td>ewing</td>
<td>31.0 MB</td>
<td>11.2 MB</td>
<td>38.0 MB</td>
<td>0 bytes</td>
<td>0 bytes</td>
</tr>
<tr>
<td>SystemUIServer.xpc</td>
<td>1</td>
<td>18.3 MB</td>
<td>320</td>
<td>48172</td>
<td>ewing</td>
<td>30.8 MB</td>
<td>10.3 MB</td>
<td>19.3 MB</td>
<td>12 KB</td>
<td>0 bytes</td>
</tr>
<tr>
<td>VTDecoderXPCService</td>
<td>2</td>
<td>0 bytes</td>
<td>42</td>
<td>53295</td>
<td>ewing</td>
<td>17.7 MB</td>
<td>6.0 MB</td>
<td>6.0 MB</td>
<td>0 bytes</td>
<td>0 bytes</td>
</tr>
<tr>
<td>accountsd</td>
<td>5</td>
<td>58.2 MB</td>
<td>55</td>
<td>368</td>
<td>ewing</td>
<td>56.2 MB</td>
<td>24.7 MB</td>
<td>21.5 MB</td>
<td>0 bytes</td>
<td>0 bytes</td>
</tr>
<tr>
<td>ndworker</td>
<td>3</td>
<td>14.2 MB</td>
<td>154</td>
<td>472</td>
<td>ewing</td>
<td>36.9 MB</td>
<td>24.1 MB</td>
<td>13.8 MB</td>
<td>1.3 MB</td>
<td>0 bytes</td>
</tr>
<tr>
<td>ndworker</td>
<td>4</td>
<td>0 bytes</td>
<td>205</td>
<td>53328</td>
<td>ewing</td>
<td>20.4 MB</td>
<td>48.8 MB</td>
<td>16.7 MB</td>
<td>23.2 MB</td>
<td>132 KB</td>
</tr>
<tr>
<td>com.apple.AppKit.xpc</td>
<td>4</td>
<td>0 bytes</td>
<td>205</td>
<td>53328</td>
<td>ewing</td>
<td>20.4 MB</td>
<td>48.8 MB</td>
<td>16.7 MB</td>
<td>23.2 MB</td>
<td>132 KB</td>
</tr>
<tr>
<td>com.apple.AppKit.xpc</td>
<td>4</td>
<td>0 bytes</td>
<td>205</td>
<td>53328</td>
<td>ewing</td>
<td>20.4 MB</td>
<td>48.8 MB</td>
<td>16.7 MB</td>
<td>23.2 MB</td>
<td>132 KB</td>
</tr>
<tr>
<td>com.apple.AppKit.xpc</td>
<td>4</td>
<td>24.6 MB</td>
<td>259</td>
<td>7932</td>
<td>ewing</td>
<td>28.3 MB</td>
<td>40.6 MB</td>
<td>3.4 MB</td>
<td>42.8 MB</td>
<td>0 bytes</td>
</tr>
<tr>
<td>com.apple.AppKit.xpc</td>
<td>4</td>
<td>0 bytes</td>
<td>205</td>
<td>53328</td>
<td>ewing</td>
<td>20.4 MB</td>
<td>48.8 MB</td>
<td>16.7 MB</td>
<td>23.2 MB</td>
<td>132 KB</td>
</tr>
<tr>
<td>com.apple.AppKit.xpc</td>
<td>4</td>
<td>0 bytes</td>
<td>205</td>
<td>53328</td>
<td>ewing</td>
<td>20.4 MB</td>
<td>48.8 MB</td>
<td>16.7 MB</td>
<td>23.2 MB</td>
<td>132 KB</td>
</tr>
<tr>
<td>com.apple.AppKit.xpc</td>
<td>4</td>
<td>0 bytes</td>
<td>205</td>
<td>53328</td>
<td>ewing</td>
<td>20.4 MB</td>
<td>48.8 MB</td>
<td>16.7 MB</td>
<td>23.2 MB</td>
<td>132 KB</td>
</tr>
<tr>
<td>identitiservicesd</td>
<td>6</td>
<td>10.6 MB</td>
<td>533</td>
<td>360</td>
<td>ewing</td>
<td>22.5 MB</td>
<td>44.8 MB</td>
<td>10.6 MB</td>
<td>8.8 MB</td>
<td>0 bytes</td>
</tr>
<tr>
<td>parsecd</td>
<td>1</td>
<td>17.1 MB</td>
<td>112</td>
<td>451</td>
<td>ewing</td>
<td>28.4 MB</td>
<td>43.1 MB</td>
<td>12.2 MB</td>
<td>25.3 MB</td>
<td>316 KB</td>
</tr>
<tr>
<td>IMDPersistanceAgent</td>
<td>2</td>
<td>6.4 MB</td>
<td>97</td>
<td>390</td>
<td>ewing</td>
<td>7.6 MB</td>
<td>42.9 MB</td>
<td>1.9 MB</td>
<td>35.0 MB</td>
<td>0 bytes</td>
</tr>
<tr>
<td>CallNCService</td>
<td>4</td>
<td>4.3 MB</td>
<td>164</td>
<td>341</td>
<td>ewing</td>
<td>13.3 MB</td>
<td>42.6 MB</td>
<td>6.5 MB</td>
<td>8.7 MB</td>
<td>0 bytes</td>
</tr>
<tr>
<td>callNCservice</td>
<td>2</td>
<td>14.8 MB</td>
<td>305</td>
<td>524</td>
<td>ewing</td>
<td>19.3 MB</td>
<td>40.6 MB</td>
<td>4.0 MB</td>
<td>7.2 MB</td>
<td>0 bytes</td>
</tr>
<tr>
<td>callNCservice</td>
<td>2</td>
<td>23.0 MB</td>
<td>169</td>
<td>793</td>
<td>ewing</td>
<td>30.8 MB</td>
<td>42.6 MB</td>
<td>11.7 MB</td>
<td>7.2 MB</td>
<td>0 bytes</td>
</tr>
<tr>
<td>BlurriGenProj</td>
<td>4</td>
<td>0 bytes</td>
<td>193</td>
<td>53316</td>
<td>ewing</td>
<td>13.8 MB</td>
<td>39.8 MB</td>
<td>11.7 MB</td>
<td>15.3 MB</td>
<td>732 KB</td>
</tr>
</tbody>
</table>

**Qt Version**

- Apple Calculator (for reference)

**IUP Version**

- BlurriGenProj

---

**More RAM**

- Calculator

**Less RAM**

- BlurriGenProj
## RAM Usage Comparison

### Ubuntu 12.04LTS

#### Qt Version

<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PR</th>
<th>NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHR</th>
<th>S</th>
<th>%CPU</th>
<th>%MEM</th>
<th>TIME+</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>1058</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>353m</td>
<td>133m</td>
<td>22m</td>
<td>S</td>
<td>7</td>
<td>3.4</td>
<td>8:04.91</td>
<td>Xorg</td>
</tr>
<tr>
<td>32384</td>
<td>pinky</td>
<td>20</td>
<td>0</td>
<td>897m</td>
<td>73m</td>
<td>38m</td>
<td>S</td>
<td>6</td>
<td>1.9</td>
<td>0:01.08</td>
<td>BlurrrGenProj</td>
</tr>
</tbody>
</table>

#### IUP Version

<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PR</th>
<th>NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHR</th>
<th>S</th>
<th>%CPU</th>
<th>%MEM</th>
<th>TIME+</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>1058</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>345m</td>
<td>125m</td>
<td>22m</td>
<td>S</td>
<td>1</td>
<td>3.2</td>
<td>7:56.43</td>
<td>Xorg</td>
</tr>
<tr>
<td>11641</td>
<td>pinky</td>
<td>20</td>
<td>0</td>
<td>694m</td>
<td>20m</td>
<td>11m</td>
<td>S</td>
<td>0</td>
<td>0.5</td>
<td>39:31.88</td>
<td>BlurrrGenProj</td>
</tr>
</tbody>
</table>

#### gcalctool

<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PR</th>
<th>NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHR</th>
<th>S</th>
<th>%CPU</th>
<th>%MEM</th>
<th>TIME+</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>1058</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>352m</td>
<td>132m</td>
<td>22m</td>
<td>S</td>
<td>4</td>
<td>3.4</td>
<td>8:11.67</td>
<td>Xorg</td>
</tr>
<tr>
<td>32511</td>
<td>pinky</td>
<td>20</td>
<td>0</td>
<td>400m</td>
<td>16m</td>
<td>10m</td>
<td>S</td>
<td>2</td>
<td>0.4</td>
<td>0:00.30</td>
<td>gcalctool</td>
</tr>
</tbody>
</table>

More RAM: Qt Version

Less RAM: IUP Version and gcalctool for reference
Qt Launch Time (slow)
Native/IUP Launch Time (fast)
Then Raspberry Pi happened
IUP
Portable User Interface
Version 3.23

IUP is a multi-platform toolkit for building graphical user interfaces. It offers a simple API in three basic languages: C, Lua and LED. IUP’s purpose is to allow a program source code to be compiled in different systems without any modification. Its main advantages are:

• high performance, due to the fact that it uses native interface elements.
• fast learning by the user, due to the simplicity of its API.

This work was developed at Tecgraf/PUC-Rio by means of the partnership with PETROBRAS/CENPES.

Project Management:

Antonio Escaño Scuri

Tecgraf - Computer Graphics Technology Group, PUC-Rio, Brazil
http://www.tecgraf.puc-rio.br/iup
Also available at: http://iup.sourceforge.net/
Why does native UI matter?

• Already mentioned RAM & Performance

• Also usability conventions
Cocoa Discontinuous Text Selection

1. Can select text vertically
   1. Can cut & paste

2. Can select multiple discontinuous sections
   1. Can cut & paste
Find Buffer: 
Cmd-E, Cmd-G

Native (top) 
vs.
Java Android Studio (bottom)

1. Cmd-E to put into Find Buffer
2. Cmd-G to find next
1. Make Upper Case
2. Ctrl-Cmd-D (or Services Menu) to activate Dictionary
Accessibility

- Microsoft, Apple, Google spend enormous effort to make their platforms accessible to people with special needs
- Built-in behaviors are automatic if you use their stuff
- Many non-native app miss this
- Selling to the government usually requires apps to be Accessible
Accessibility -> Display
Microphone Speech-to-Text
Native (left) vs. Qt (right)

- Fn-Fn to activate
IUP the research side

• IUP started as a research project for good reason

  • How can you make a cross-platform interface when the native interfaces for every platform are so drastically different?

  • How do you provide access to features that not all platforms may support?

  • And how do you do this without constantly changing/breaking the API, especially when new platforms are introduced?

  • How do you deal with different sizes for widgets

• Since every platform uses a different programming language for their native development, how do you deal with this in a flexible and cross-platform way?
• IUP’s core and public API are implemented in pure C, because C is the one language that every language can talk to

• Platform backends are implemented in each platform’s native language using the native widget set

• IUP does not employ language subclassing since that can’t be expected to work across all the platforms.

• Instead IUP uses attributes to set properties
IUP Attribute Solution

IupSetAttribute(button, "TITLE", "OK");
IupSetAttribute(button, "ACTIVE", "NO"); // disables button

• Can scale to cover entire native API features
• Does not require breaking/changing the API
• Unsupported properties on platforms can simply ignore the feature request
• Makes IUP simple to learn/use
IUP Designed for Language Bindings

- Recognized most people don’t want to write in C, so API was designed for easy binding
- Small API
  - Attributes help keep it small
- Lua bindings first class citizens
- Lots of other language bindings
IUP LED: Textual Layout Description Format

- Think: Windows XAML, Android XML layout, Apple Interface Builder XIB
- Optional: (can do everything programmatically)
- Normally runtime, but optional compiler to convert to compile time
- Optional use case: Can have different LED files for different platforms
- Example:

```
btn = button[ACTIVE=NO]('"OK", action_ok)

dlg = dialog[SIZE = FULLxFULL TITLE = "Test"](btn)
```
IupLayoutDialog: (Live) Run-time Layout Editor

Real app/GUI on left

Layout Editor on Right
Other official IUP accessory libraries

- IupCD (Canvas Draw library)
  - Uses the native 2D drawing API on each system to implement “non-native” widgets (like Qt)
  - Easy way to create new, cross-platform widgets
- IupPlot: Plot/graphing library built on IupCD
- IupGL: OpenGL
Definition: Cocoa

- The framework (library) you write native Mac applications in
- Provides lots of widgets, e.g. Text, Buttons, Windows, etc.
- Name is pun on “Java”
  - Favorite coffee vs. Favorite hot beverage
IupCocoa

• Definition: Objective-C
  • The native programming language you write Cocoa applications in
  • It is a 100% pure superset of C (which C++ can’t even claim)
  • Obj-C adds an object system and powerful runtime inspired by Smalltalk.
  • (Strange) Syntax was designed to avoid conflicting with C/C++, which allows intermixing all 3 languages in the same file
Quick Overview on Implemented things

• Cocoa is pretty straight-forward and matches well with Windows & GTK

• IupCocoa is not finished but...
  • It is also further along than most people think
    • (I’m now shipping the Mac IUP version of BlurrrGenProj seen earlier.)
Dialogs, Labels, & Buttons

Hello World 1

Hello world from IUP

سلام.

Hello World 1

سلام.

Hello world from IUP

سلام.

OK

OK
(Input) Text

First Line
Second Line Big
Third Line
more

Single Line Text

First Line
Second Line Big
Third Line
more
more

Multiline:
First Line
Second Line Big Big Big
Third Line
more
more

Single Line Text
Calendar
File Dialogs
Canvas
Some Impedance Mismatches

- Event Loop
- Application Menu system
Cocoa Event Loop

- Cocoa wants to control the event loop
- You are not supposed to pump it yourself
- Yes, there are ways around this, but has been known to break things
  - Modal windows
  - menu bar behavior
  - Game Center
• Using: `IupSetGlobal("MENU", (const char*)main_menu);`;

• Instead of per-window: `IupSetAttributeHandle(dialog, "MENU", main_menu);`;
One More Thing... GNUStep

• Courtesy: Germán Arias
That’s IupCocoa

• But...
But it’s 2017…
Mobile Revolution started in 2007

- 270 million PCs shipped in 2016 (Gartner)
- 1.5 billion smartphones shipped in 2016 (Gartner)
- PC sales declining

Gartner Says Worldwide PC Shipments Declined 3.6 Percent in Third Quarter of 2017

Traditional Promotions, Such as Back-to-School Sales, No Longer an Effective Growth Driver

STAMFORD, Conn.--(BUSINESS WIRE)--Worldwide PC shipments totaled 67 million units in the third quarter of 2017, a 3.6 percent decline from the third quarter of 2016, according to preliminary results by Gartner, Inc. This is the 12th consecutive quarter of declining PC shipments.

“While there were signs of stabilization in the PC industry in key regions, including EMEA, Japan and Latin America, the relatively stable results were offset by the U.S. market, which saw a 10 percent year-over-year decline in part because of a very weak back-to-school sales season.”
Google switching to Mobile-first indexing

- Mobile versions of sites will be used for ranking

Mobile-first Indexing
Friday, November 04, 2016

Today, most people are searching on Google using a mobile device. However, our ranking systems still typically look at the desktop version of a page's content to evaluate its relevance to the user. This can cause issues when the mobile page has less content than the desktop page because our algorithms are not evaluating the actual page that is seen by a mobile searcher.

To make our results more useful, we've begun experiments to make our index mobile-first. Although our search index will continue to be a single index of websites and apps, our algorithms will eventually primarily use the mobile version of a site's content to rank pages from that site, to
Two-thirds of the world now has a mobile phone in their pocket, and apparently a lot of them are spending money on games. A new report from industry analyst DFC Intelligence found that mobile games revenue exceeded PC and console revenue for the first time ever in 2016.

The mobile games market grew 32 percent to reach $38 billion last year, and according to market analyst Newzoo, will reach $65 billion in 2020. In China, tech giants Tencent and NetEase both doubled their revenue. NetEase overtook Tencent as the No. 1 mobile publisher last year, but Tencent is still the largest gaming company in the world and has a huge presence in not just mobile but in the PC gaming world as well with subsidiaries like League of Legends studio Riot. It reported a 58 percent increase in Q1 this year and is currently valued at $316 billion.
Good News: IUP's original design seems flexible enough to handle this

- “Attributes” instead of too many hardcoded APIs
- We have “IupDialog” and not “IupWindow”
- But… requires some careful design/thought on how things should map/ work on mobile
Example “Thought-Exercise”:
What does “Dialog” mean for mobile?

- “Dialog” is a “Window” on Desktop

- iOS: While a UIWindow exists, the paradigm isn’t good for multiple dialogs
  - The more common and useful construct is the UIViewController

- Android: The corresponding construct is an Activity
iOS: UIWindow is not the best mapping for Dialog

- Not obvious how to deal with multiple dialogs
- Use Safari as an example of multiple UIWindows
- Safari “switching” behavior is not built-in
iOS UIVViewController & Android Activity

• Both UINavigationController & Activity drill down & back through a stack of views

• Users expect this behavior

• Behavior is built-in
So let’s implement it and put it all together

- Demo: Show a singular IUP program
- Runs native on all platforms
- Feels natural for every platform
Demo Program: Create Dialog & Button (recursive)

```swift
var g_buttonCounter = 0;

func BlurrrMain() -> Int32 {
    IupOpen(nil, nil)
    IupSetFunction("ENTRY_POINT", IupEntryPoint);
    IupMainLoop()
    return 0;
}

func IupEntryPoint(_ ih:OpaquePointer?) -> Int32 {
    g_buttonCounter = 0
    return ShowNewDialogCallback(nil)
}

func ShowNewDialogCallback(_ ih:OpaquePointer?) -> Int32 {
    let button = IupButton(nil, nil)
    IupSetStrAttribute(button, "TITLE", "Iup Button \(g_buttonCounter)")
    IupSetCallback(button, "ACTION", ShowNewDialogCallback)
    let dialog = IupDialog(button)
    IupSetAttribute(dialog, "SIZE", "QUARTERxQUARTER")
    IupSetStrAttribute(dialog, "TITLE", "Iup Dialog \(g_buttonCounter)")
    IupShow(dialog)
    g_buttonCounter += 1
    return IUP_DEFAULT
}
```

*Fun Fact: Written in Swift using bindings to IUP*
Ubuntu Linux (amd64)
Raspberry Pi (Raspbian)
*Fun fact: Swift on Windows is still extremely experimental*
Mac
Android
IupCocoaTouch

- Very similar to Mac
  - APIs are a little different (UIView instead of NSView), but semantically similar
iOS Event Loop

• Apple controls the event loop (same as Mac)

• More rigid than Mac
IupAndroid

- **All** Android apps *must* use the Android **SDK** which is in Java
  - You cannot escape this

- Android GUI APIs are completely in Java

- Android **NDK** was later added to allow for C & C++ development
The Android NDK “Really Does Suck”

• John Carmack - “Half-baked”, “Really does suck”

• Second class citizen on Android

• Almost no Android libraries are provided in the NDK

• Lots of things are broken, slow to get fixed, if ever

• Word on the street (few years ago): Only 2 full-time Google engineers + a few part time

  • Consistent with number of Google employees on NDK mailing list

  • No slight intended on those 2 engineers. Valiant effort. Google treats them as the black sheep.

  • Google: Among the richest, powerful companies in the world with #1 dominance in mobile, and this is the best effort Google chooses to put in. Shameful.

• Android is 9 years old. Our pleas are ignored. Public ridicule is the only tool we have left.
Android’s obsession with (Java) God Objects

- Context class
- Activity class
- Application class
Android file system and the .apk

- Files that ship with your app are inside the “.apk” (think .zip)
- Can’t use standard C file family (fopen, fread)
- Needs a “God” object from a Context class to get an AAssetManager
  
  ```
  AAsset* AAssetManager_open(AAssetManager *mgr, const char *filename, int mode);
  ```
- Existing cross-platform (ANSI) C/C++ libraries won’t work without modification
An Activity is a Context

- Common mistake is to try to keep a reference to an Activity for the life of a program
- They typically come-and-go in most apps
An Application is a Context

- Created my own IupApplication class
- Will provide a public way to retrieve it to help other libraries in your app
- Does mean that any other library that uses this same approach is incompatible with IupAndroid
Android Event Loop

- Android controls the event-loop. PERIOD.
- **Never** block the event loop
- You cannot manually pump the event loop
- Will impact lupMainLoop and start up sequence
- Also, there is no "int main" because we are Java, not C
Android (main) UI Thread

- All GUI APIs must be called from the UI Thread
- Working around the event-loop design with a background thread is usually a mistake
  - Makes you second-class citizen on the platform (can’t directly call APIs)
    - Road-block when using single-threaded languages
  - Must understand threading model of your app, the library, and the OS
    - Callbacks must be redirected back to the proper thread of the handler
- Performance usually suffers because of so many locks, context switches
IupEmscripten (Web Browser)
Chris Matzenbach

• One other major platform to discuss

• Native vs. Web fight is not over
IupEmscripten

Chris Matzenbach
Core Idea

• Do I write a native application or a web application? Must make a choice

• Why not treat the web browser like any other platform?

• This is our core idea: let’s take our native programs and deploy them to the web browser - **as if it were any other platform**
While many options exist for native development, when it comes to the web, there’s only one choice - JavaScript.

In the past this has always required a re-write for native apps.

Emscripten and the Birth of the Idea

- What is Emscripten? C/C++ to JavaScript compiler, released in 2013
- The Unreal team ported Unreal Engine 3 to the browser using Emscripten *in just four days*
- While impressive, everything is drawn onscreen. We need something better - native web widgets
- IupEmscripten - the first cross-platform library using the browser’s own native widgets
How does this work?

- In order to render native web widgets, we need to call into JavaScript to access the DOM APIs

- Emscripten didn’t intend for us to modify the JavaScript side

- We do have the ability to call into external JavaScript functions - from there, we can access the necessary APIs to draw native widgets

- This is what differentiates us from other cross-platform libraries that also compile for the web
```c
extern int emjsTest_GetInt(int widget_id);

// here is our C function, which will make use of our external JavaScript function
int emscriptenExample(int widgetID) {

    // here we call our external function, as if it were any other C function
    int val = emjsTest_GetInt(widgetID);

    return val;
}
```

```javascript
// library which holds our external JavaScript functions
var Library = {

    emjsTest_GetInt: function(widget_id) {
        // pretending this returns an int (really returns a string)
        return document.getElementById(widget_id).innerHTML;
    }
};
```
Widget Creation

- User defines the widget they want in Iup, which we can access on the C-side. How do we get this over to JavaScript?
  - Emscripten does not allow us to pass objects over the C/JavaScript bridge
  - No stack API like in Lua
  - We can, however, pass integers across
  - We utilize a global ID map that maps integers to objects, serving as a proxy and allowing us object access on both sides of the bridge
  - Each side references the same ID along with their own ‘interpretation’ of the widget
Label, Input Text and Button

- Here we have a super simple form, showing the label, input text and button widgets.
Dialog, Label and Button

- Here we have an external dialog (aka “pop-up”) with a simple label and button.

![External dialog with label and button](image1.png)

Hello world from IupEmscripten!

Button
List - Dropdown

- Standard dropdown list; sizes automatically based off of largest item
List - Multiple

- Multiple selection list; user can hold down command/control to select/deselect multiple items within list.
List - Editbox+Dropdown

- List type that functions as a dropdown, but also allows user to type in the input box, narrowing down the selections.
Memory Management

- Emscripten follows a C/C++ paradigm and assumes we manage memory ourselves - no garbage collector

- Iup manages the memory for us through the use of Map and UnMap functions

- However, any objects we create on the JavaScript side will be garbage collected once we return from the external function call

- How do we prevent this from happening?
The Global ID Map!!

- The answer is our global ID map - because a reference to the object exists in the ID map, it prevents the object from being garbage collected by JavaScript.

- Likewise, by calling into JavaScript from Iup’s UnMap function, we can remove the object from the ID map, ensuring it is garbage collected by JavaScript.
• As Eric mentioned, Iup wants to control the Event Loop. We cannot let that happen!

• We need to let the web browser and JavaScript control the Event Loop. There is no other option.
What have we learned?

• The native experience Iup promises can be brought to the web

• The web backend allows your applications to be more portable than ever

• If your user’s device can run a web browser, it can run your application

• You no longer have to make the choice between native vs web!
Bringing Everything Together

• A few changes to IUP are needed
IUP needed changes
(for all platforms)

• Rules:

• (Unchanged) Legacy code must continue to still run as before

• But those who want the new platforms must opt-in by conforming to the new (slight) changes

• Existing platforms are updated to work with these new changes
IUP Init (Legacy)

```c
int main(int argc, char* argv[])
{
    IupOpen(&argc, &argv);
    CreateYourGui(); // your stuff here
    IupMainLoop();
    IupClose();
    return 0;
}
```
IUP Init (Old vs. New)

```c
int main(int argc, char* argv[]) {
    IupOpen(&argc, &argv);
    CreateYourGui(); // your stuff here
    IupMainLoop();
    IupClose();
    return 0;
}

void IupExitPoint() {
    IupClose();
}

void IupEntryPoint() {
    IupSetFunction("EXIT_CB",
                   (Icallback)IupExitPoint);
    CreateYourGui(); // your stuff here
}
```

```c
int main(int argc, char* argv[]) {
    IupOpen(&argc, &argv);
    IupSetFunction("ENTRY_POINT",
                   (Icallback)IupEntryPoint);
    IupMainLoop();
    return 0;
}
```
IUP Init (Old vs. New)

```c
int main(int argc, char* argv[]) {
    IupOpen(&argc, &argv);
    CreateYourGui(); // your stuff here
    IupMainLoop();
    IupClose();
    return 0;
}
```

```c
int main(int argc, char* argv[]) {
    IupOpen(&argc, &argv);
    CreateYourGui(); // your stuff here
    IupMainLoop();
    return 0;
}
```

Acts as explicit Opt-in for new behavior
int main(int argc, char* argv[]) {
    IupOpen(&argc, &argv);
    CreateYourGui(); // your stuff here
    IupMainLoop();
    IupClose();
    return 0;
}

void IupExitPoint() {
    IupClose();
}

void IupEntryPoint() {
    IupSetFunction("EXIT_CB",
                   (Icallback)IupExitPoint);
    CreateYourGui(); // your stuff here
}

int main(int argc, char* argv[]) {
    IupOpen(&argc, &argv);
    IupSetFunction("ENTRY_POINT",
                   (Icallback)IupEntryPoint);
    IupMainLoop();
    IupClose();
    return 0;
}
IUP Init on Cocoa/CocoaTouch

void IupExitPoint()
{
    IupClose();
}

void IupEntryPoint()
{
    IupSetFunction("EXIT_CB",
                   (Icallback)IupExitPoint);
    CreateYourGui(); // your stuff here
}

int main(int argc, char* argv[])
{
    IupOpen(&argc, &argv);
    IupSetFunction("ENTRY_POINT",
                   (Icallback)IupEntryPoint);
    IupMainLoop();
    return 0;
}

1. Starts native event loop
2. Calls IupEntryPoint
3. May never return
**IUP Init on Android**

```
void IupEntryPoint()
{
    IupSetFunction("EXIT_CB",
                   (Icallback)IupExitPoint);
    CreateYourGui(); // your stuff here
}

int main(int argc, char* argv[])
{
    IupOpen(&argc, &argv);
    IupSetFunction("ENTRY_POINT",
                   (Icallback)IupEntryPoint);
    IupClose();
    return 0;
}
```
IUP Init on Emscripten

void IupExitPoint()
{
    IupClose();
}

void IupEntryPoint()
{
    IupSetFunction("EXIT_CB",
        (Icallback)IupExitPoint);
    CreateYourGui(); // your stuff here
}

int main(int argc, char* argv[])
{
    IupOpen(&argc, &argv);
    IupSetFunction("ENTRY_POINT",
        (Icallback)IupEntryPoint);
    IupMainLoop();
    return 0;
}

1. Must not block
2. Calls IupEntryPoint
3. Returns immediately

main finishes, but our application continues to run
IUP Init on Updated Existing Platforms

void IupExitPoint()
{
    IupClose();
}

void IupEntryPoint()
{
    IupSetFunction("EXIT_CB",
                   (Icallback)IupExitPoint);
    CreateYourGui(); // your stuff here
}

int main(int argc, char* argv[])
{
    IupOpen(&argc, &argv);
    IupSetFunction("ENTRY_POINT",
                   (Icallback)IupEntryPoint);
    IupMainLoop();
    return 0;
}

1. Blocks, manually pumps event loop
2. Calls IupEntryPoint on first time
3. Returns on Quit

Used to detect opt-in to activate new behavior
Iup Init is cross-platform again

• Legacy Apps:
  • Don’t design around manually pumping the event loop (nested IupMainLoop())
  • If you need to poll, use IupTimer to poll with periodic callbacks

```c
void IupEntryPoint()
{
    IupSetFunction("EXIT_CB",
                   (Icallback)IupExitPoint);
    CreateYourGui(); // your stuff here
}

int main(int argc, char* argv[])
{
    IupOpen(&argc, &argv);
    IupSetFunction("ENTRY_POINT",
                   (Icallback)IupEntryPoint);
    IupMainLoop();
    return 0;
}
```
Threading Model becomes more rigorous

- Cocoa, CocoaTouch, Android, Emscripten all must be on the main thread
- IUP was ambiguous about threads
  - That needs to be formalized to require people to write on the main thread
Putting it all together: Final questions & demos

• “How far can I take it?”
Can I use custom or platform-specific code in my IUP app?

• Yes!
Example: IupBork

- Last year demo: LuaCocoa (Muppet) Swedish Chef translator
- Written with LPeg / Lua
- This year: Ported to IUP
- Single cross-platform UI written in IUP
- Native speech synthesizer implementation for each platform

```python
bork = re.compile[
    text <- {~ item* ~}
    WordChar <- [A-Za-z']
    NotWord <- [^A-Za-z']
    item <- ProcessedWord / NotWord

    ExemptWord <- 'bork' / 'Bork'

    EndOfParagraphPunctuation <- [!.?]%nl
        -> 'Bork Bork Bork!

    AccentSyllable <- 'an' -> 'un'
        / 'An' -> 'Un'
        / 'au' -> 'oo'
        / 'Au' -> 'Oo'
        / 'the' -> 'zee'
        / 'The' -> 'Zee'
        / 'v' -> 'f'
        / 'V' -> 'F'
        / 'w' -> 'v'
        / 'W' -> 'V'
...
```
IupBork: Windows

Uses ISpVoice via C++
IupBork: Linux

Fork/exec to external process ‘spd-say’
IupBork: Mac

Uses NSSpeechSynthesizer via Objective-C or Swift
IupBork: iOS

Uses AVSpeechSynthesizer via Objective-C or Swift
IupBork: Android

Uses TextToSpeech via Java
IupBork: Emscripten

Uses SpeechSynthesisUtterance via JavaScript/Web

- Yes, convoluted/crazy…

1. C code including IUP, Lua VM, LPeg all compiled to JavaScript via Emscripten

2. Loading all compiled JS in a web browser

3. Running Lua script inside JS-compiled LuaVM inside JS web browser VM

4. Calling out to native JavaScript/Web APIs for speech (and GUI)
Can I integrate my own custom/native views with IUP?

• Yes!

• (In fact, that’s exactly how we implement IUP in the first place)
Example: IupWeb

- Separate, not part of the main IUP library

- Each platform has a different native web widget

- Native => tiny profile

```c
web = IupWebBrowser();
IupSetAttribute(web, "VALUE", "https://www.blurrrsdk.com");
dlg = IupDialog(web);
IupShow(dlg);
```
IupWeb: Linux GTK
IupWeb: iOS
IupWeb: Android
IupWeb: Emscripten
Size is small too
• Blurrr SDK now shipping with “Sneak Preview” IUP
  • https://blurrrsdk.com
  • Contains templates & examples seen today (e.g. IupBork, IupWeb)
• Repos:
  • https://github.com/ewmailing/IupCocoa
  • https://github.com/ewmailing/IupCocoaTouch
  • https://github.com/ewmailing/IupAndroid
  • https://github.com/ewmailing/IupEmscripten
Please Support Us

- Purchase Blurrr SDK (blurrrsdk.com)
- Donation link at blurrrsdk.com
- Corporate Sponsorships
- Consulting work?
- Volunteers/Contributions
- Google Summer of Code organization
  - Maybe LuaLab can help?
- Please spread the word about IUP Next. (Friends, Social Media)
GIST Walk for a Cure

http://www.gistwalksanjose.org

GIST Walk

Walk for a Cure

Sunday, October 22, 2017

Almaden Lake Park
San Jose, CA
Carlos M. Icaza
June 5, 1966 - May 17, 2016
Thank you

Eric Wing
blurrrsdk.com
@ewingfighter / @BlurrrSDK

Chris Matzenbach
cmatzenbach@gmail.com