

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

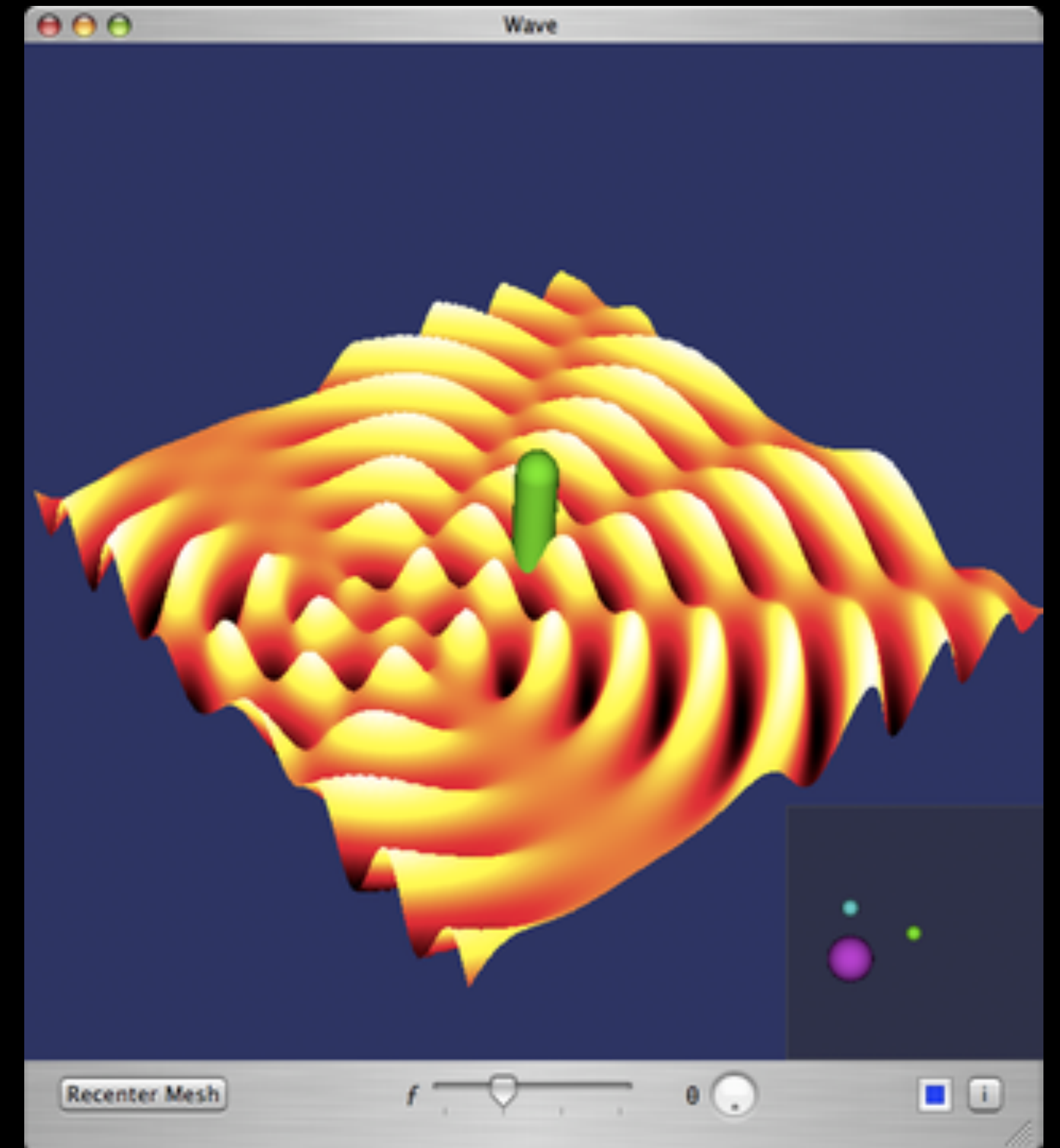
Globalstar: Satellites & Rockets



- 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



Beginning iPhone Games Development

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

Apress®

Commercial Game Engines

Corona SDK
(Lua)



Platino
(JavaScript)



appcelerator®

- Primary platforms: iOS & Android
- Also: Mac & Windows

BLURRR SDK

A MODULAR, CUSTOMIZABLE NATIVE CROSS-PLATFORM
SDK FOR 2D GAMES & APPS MADE SIMPLE

HOME WHY BLURRR? BLOG VIDEOS
LOGIN/DOWNLOAD DOCUMENTATION CONTACT

20px MAIN CONTENT AREA 640px 20px

BLURRR SDK BETA NOW OPEN

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

SOFTWARE—PRACTICE AND EXPERIENCE, VOL. 0(0), 1–27 (? 1995)

IUP/LED: A Portable User Interface Development Tool

C. H. LEVY, L. H. DE FIGUEIREDO, M. GATTASS, C. J. P. LUCENA

Departamento de Informática, PUC-Rio

Rua Marquês de São Vicente 225, 22453-900 Rio de Janeiro, RJ, Brazil

levy,lhf,gattass,lucena@icad.puc-rio.br

AND

D. D. COWAN

*Computer Science Department & Computer Systems Group
University of Waterloo, Waterloo, Ontario, Canada N2L 3G1*

dcowan@csg.uwaterloo.ca

SUMMARY

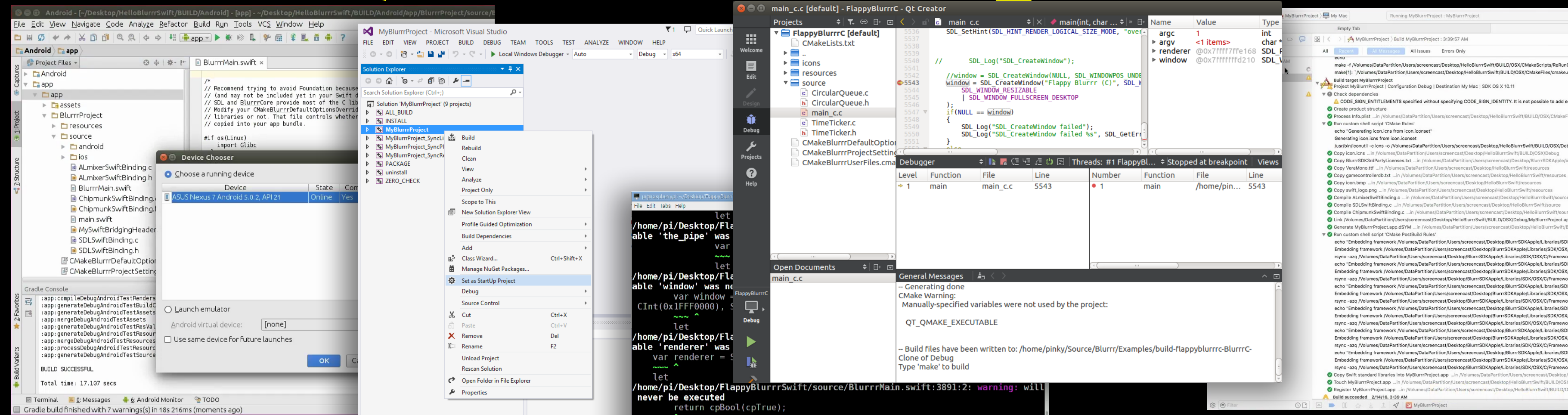
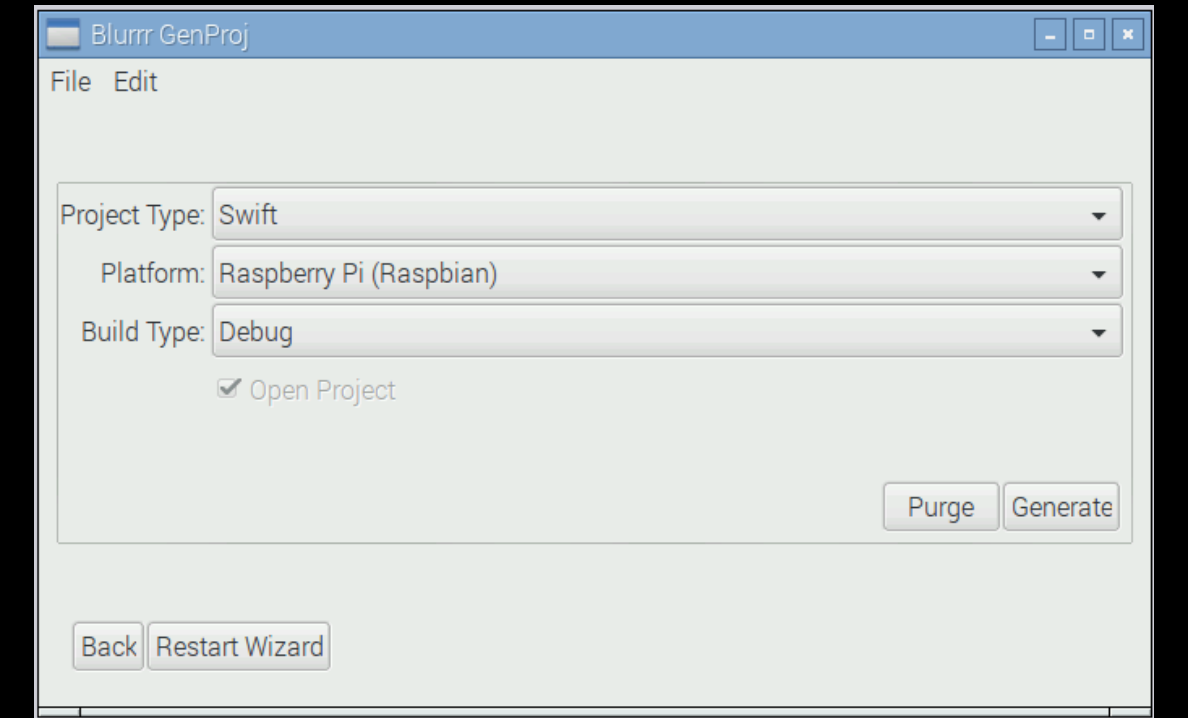
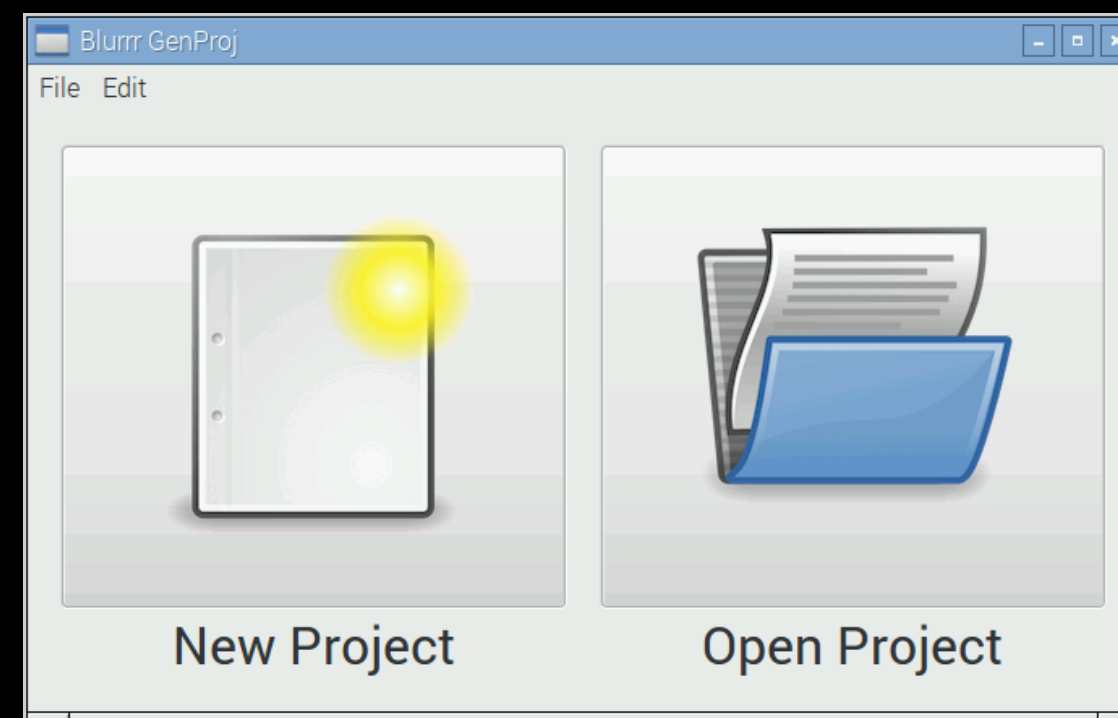
Minimizing the amount of code that must be written and maintained is particularly critical in the development of the user interface for a highly interactive system, since the code for the user interface represents a substantial part of the application. This is especially important where the interactive system is available on a number of distinct platforms. Providing a single user interface abstraction requiring only one set of source code that can be mapped automatically into specific interface systems appears to be the preferred approach; but the underlying model must be designed carefully in order to keep the system relatively simple, easy to use and maintain, and allow ease of experimentation as user interfaces are produced. We describe the design and implementation of IUP/LED, a portable user interface toolkit that we believe has these properties. The toolkit is designed for rapid prototyping and modification, to provide a look and feel

Why IUP?

Let me try to paint a picture with my story

- Needed cross-platform native-ish GUI tools for Blurrr SDK (blurrrsdk.com)

- Launcher to generate native IDE projects



Android Studio

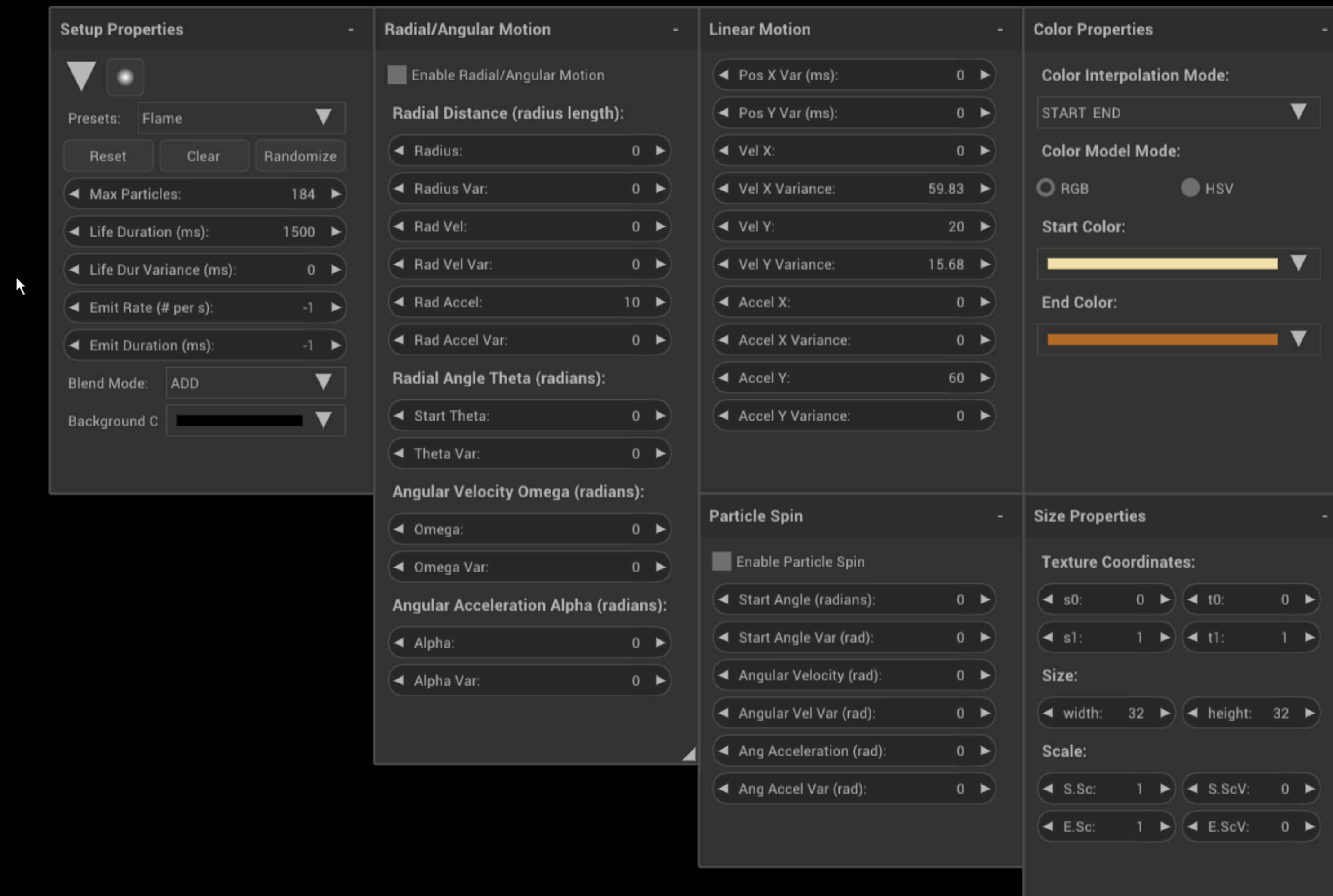
Visual Studio

Makefiles

Qt Creator

Xcode

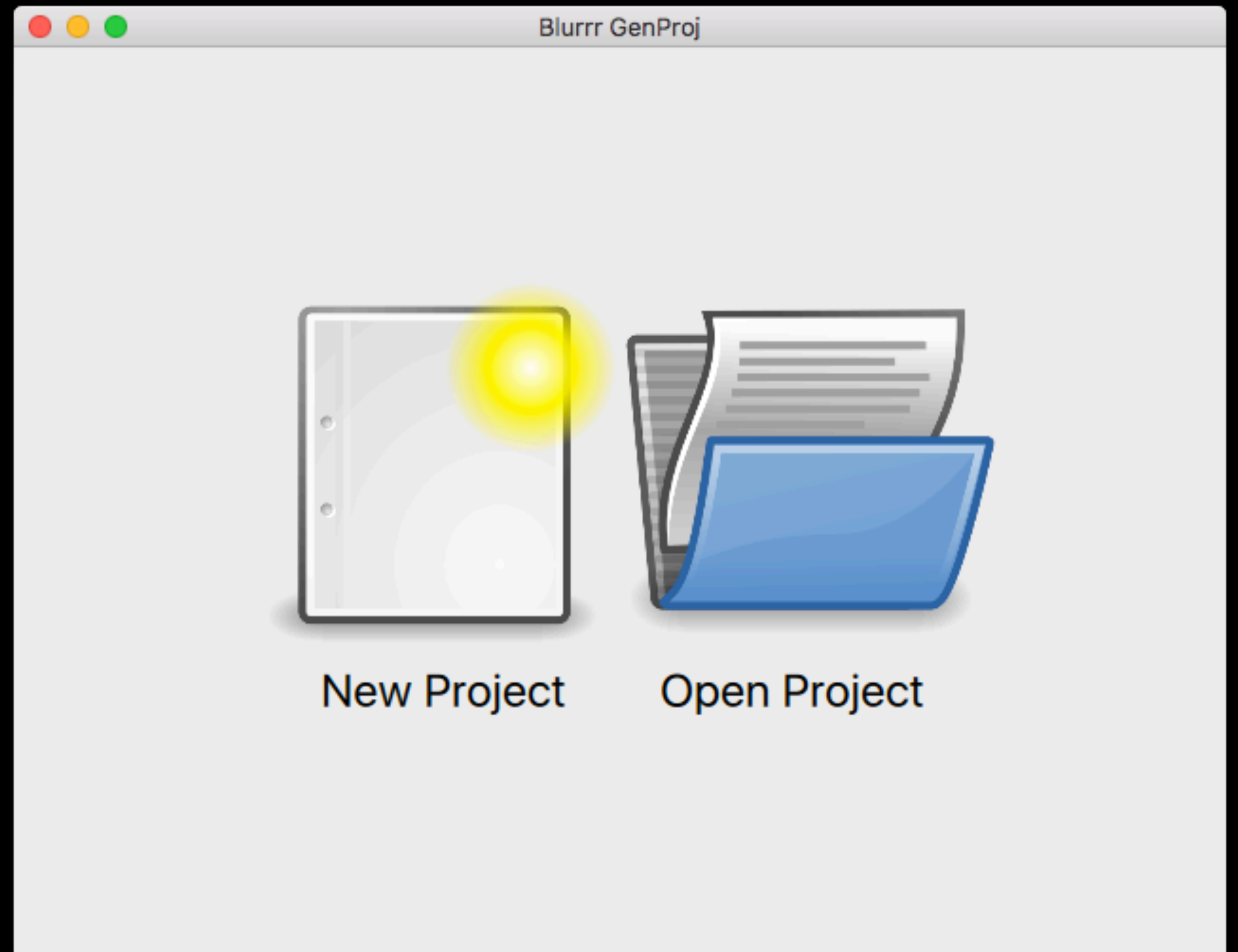
Game GUIs possible, but not ideal for requirements



Blurr 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

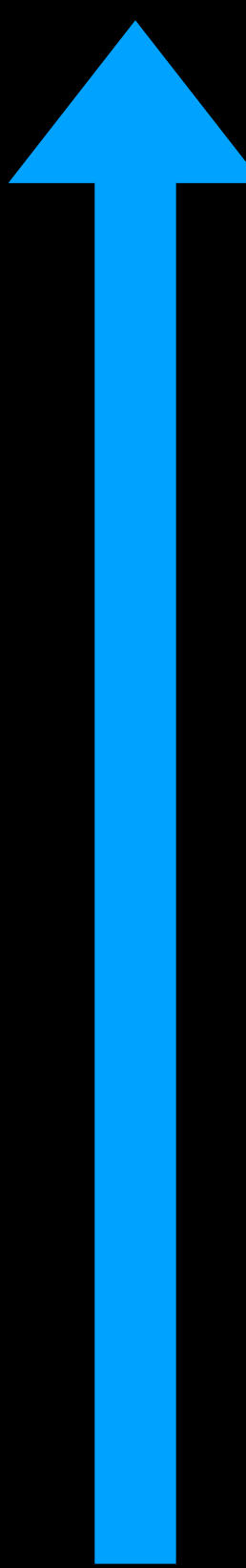


RAM Usage Comparison Mac OS X 10.12

Process Name	Threads	Compressed M...	Ports	PID	User	Memory	Real Mem	Private Mem	Shared Mem	Purgeable Mem
BlurrrGenProj	8	19.2 MB	250	45976	ewing	43.2 MB	65.3 MB	20.3 MB	20.2 MB	60 KB
Dock	3	47.6 MB	430	379	ewing	75.1 MB	85.1 MB	22.0 MB	26.0 MB	24 KB
com.apple.dt.SKAgent	3	88.1 MB	68	32336	ewing	90.2 MB	64.2 MB	24.1 MB	21.3 MB	0 bytes
loginwindow	2	26.0 MB	695	110	ewing	53.9 MB	64.0 MB	26.6 MB	14.2 MB	8 KB
com.apple.appkit.xpc....	8	17.2 MB	329	50877	ewing	33.3 MB	60.8 MB	14.5 MB	22.9 MB	0 bytes
SystemUIServer	4	16.8 MB	454	381	ewing	32.6 MB	59.3 MB	13.8 MB	19.5 MB	12 KB
storeassetd	3	14.5 MB	182	452	ewing	49.9 MB	58.2 MB	26.3 MB	6.0 MB	8 KB
com.apple.appkit.xpc....	9	17.4 MB	343	28745	ewing	31.0 MB	57.0 MB	11.2 MB	38.0 MB	0 bytes
com.apple.appkit.xpc....	9	18.3 MB	320	46172	ewing	30.8 MB	54.4 MB	10.5 MB	19.3 MB	0 bytes
VTDecoderXPCService	2	0 bytes	42	53295	ewing	17.7 MB	53.5 MB	17.0 MB	6.0 MB	0 bytes
accountsd	4	52.8 MB	255	346	ewing	65.4 MB	52.1 MB	13.2 MB	6.8 MB	1.1 MB
mdworker	4	0 bytes	55	53264	ewing	21.6 MB	49.6 MB	21.0 MB	10.0 MB	0 bytes
cloudd	10	14.2 MB	514	473	ewing	38.3 MB	49.6 MB	24.1 MB	6.0 MB	1.3 MB
Calculator	4	0 bytes	205	53328	ewing	20.4 MB	48.8 MB	16.7 MB	23.2 MB	132 KB
mdworker	4	0 bytes	55	53264	ewing	19.1 MB	47.8 MB	18.5 MB	18.0 MB	0 bytes
mdworker	4	0 bytes	55	53263	ewing	20.0 MB	46.4 MB	19.4 MB	10.0 MB	0 bytes
com.apple.appkit.xpc....	4	24.6 MB	259	7932	ewing	28.8 MB	46.0 MB	3.4 MB	42.8 MB	0 bytes
mdworker	4	0 bytes	55	53258	ewing	23.2 MB	45.4 MB	22.6 MB	10.0 MB	0 bytes
identityservicesd	6	10.5 MB	533	359	ewing	22.5 MB	44.8 MB	10.8 MB	6.8 MB	84 KB
parsecd	3	17.0 MB	121	450	ewing	28.4 MB	43.2 MB	11.1 MB	25.3 MB	316 KB
IMDPersistenceAgent	2	6.4 MB	97	390	ewing	7.6 MB	42.9 MB	1.1 MB	35.0 MB	0 bytes
CalNCService	4	4.3 MB	164	341	ewing	13.3 MB	42.6 MB	6.5 MB	8.7 MB	0 bytes
callservicesd	2	14.9 MB	305	524	ewing	19.3 MB	40.6 MB	4.0 MB	7.2 MB	0 bytes
photoanalysisd	2	32.0 MB	80	330	ewing	36.6 MB	40.4 MB	3.8 MB	7.1 MB	0 bytes
BlurrrGenProj	4	0 bytes	193	53316	ewing	13.5 MB	39.8 MB	11.7 MB	16.3 MB	732 KB

Qt Version

More RAM



Apple
Calculator
for reference

IUP Version

Less RAM

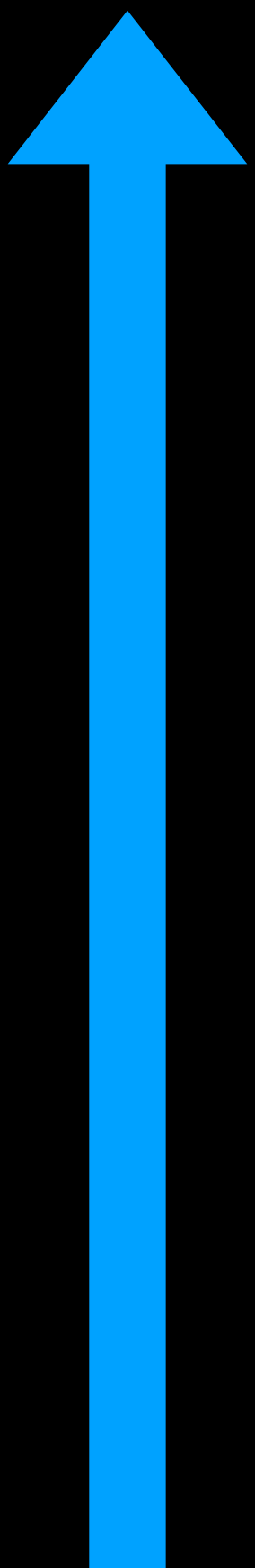
RAM Usage Comparison

Ubuntu 12.04LTS

Qt Version

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

More RAM



IUP Version

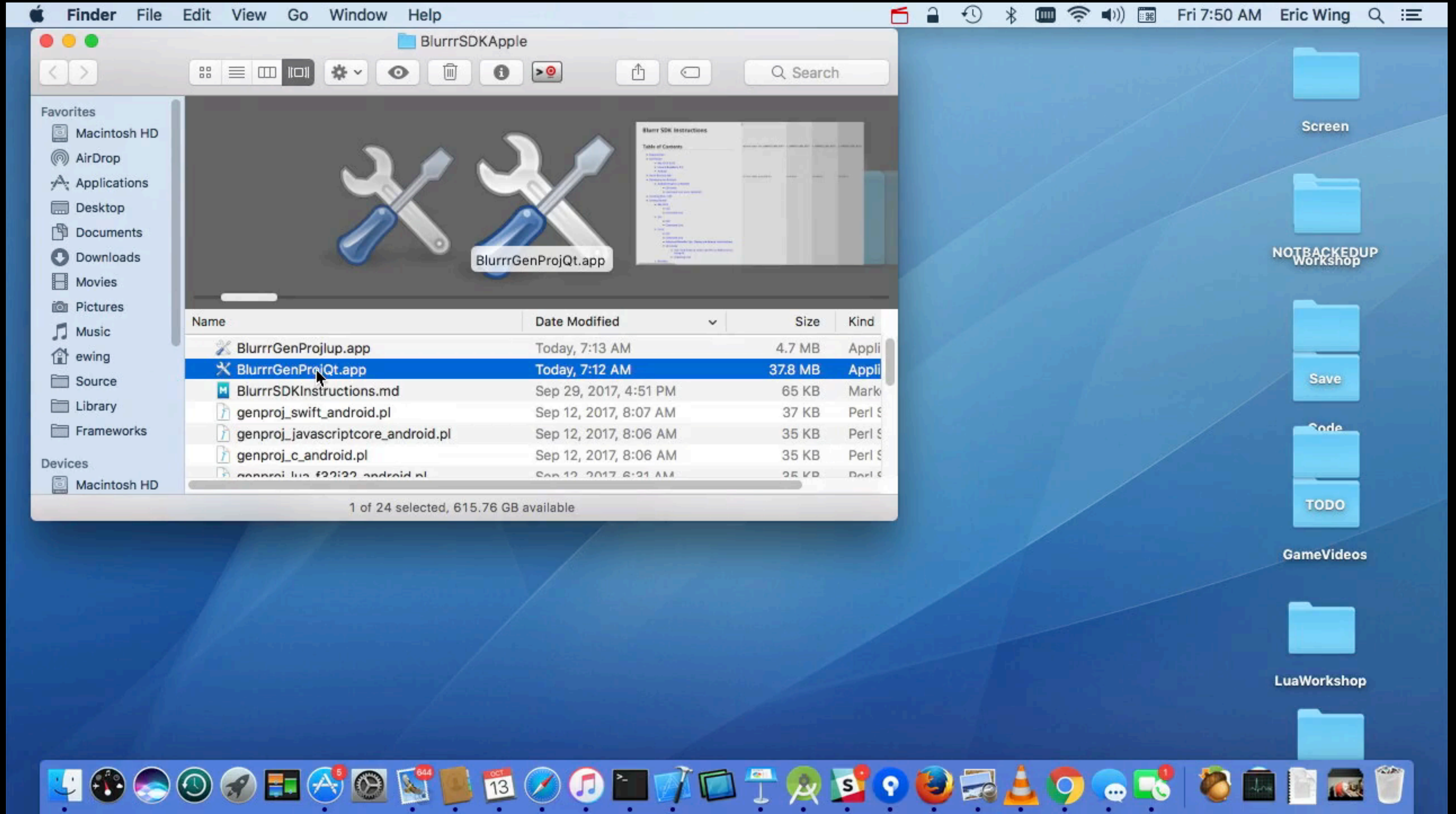
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1058	root	20	0	345m	125m	22m	S	1	3.2	7:56.43	Xorg
11641	pinky	20	0	694m	20m	11m	S	0	0.5	39:31.88	BlurrrGenProj

gcalctool
for reference

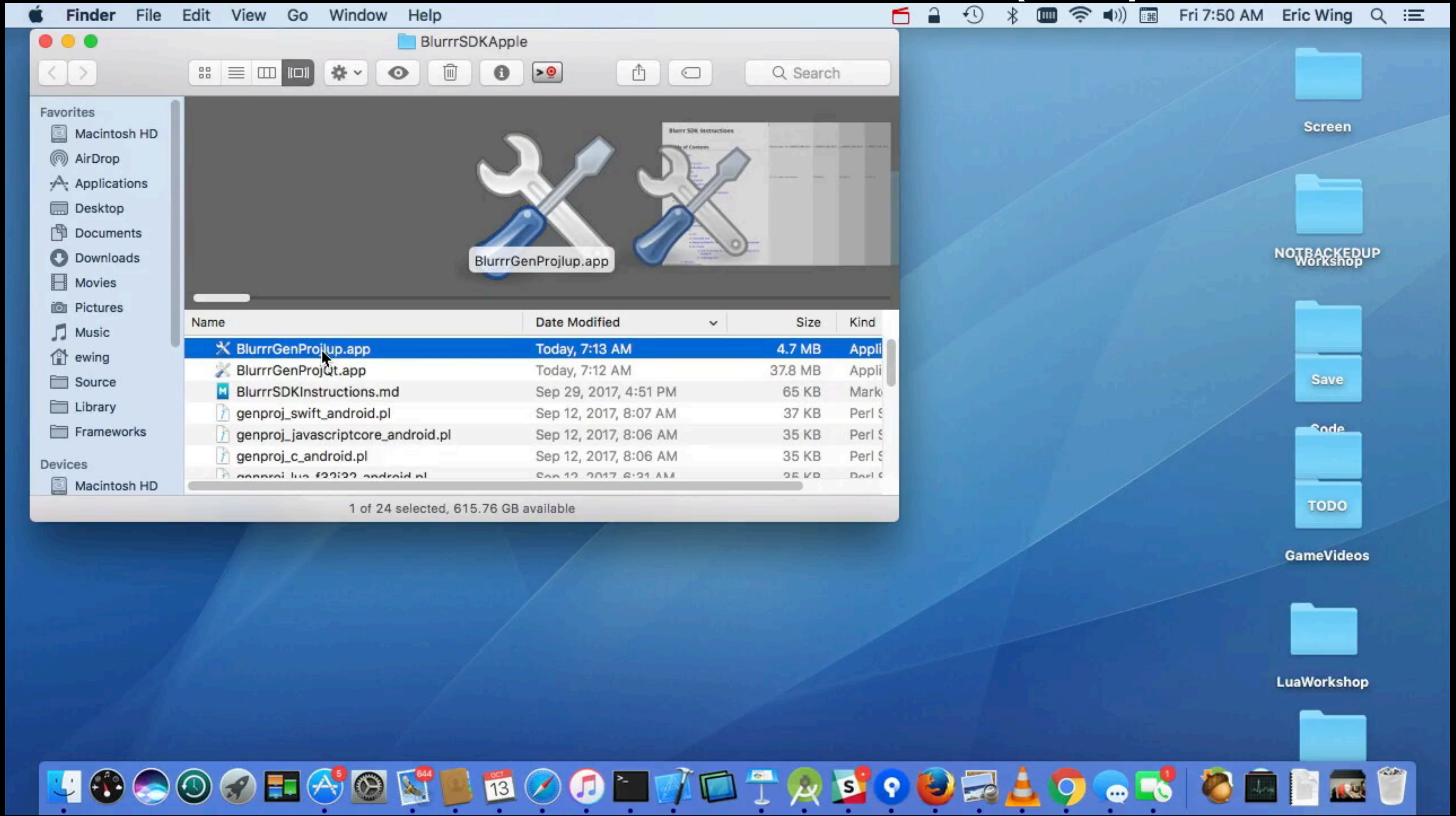
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1058	root	20	0	352m	132m	22m	S	4	3.4	8:11.67	Xorg
32511	pinky	20	0	400m	16m	10m	S	2	0.4	0:00.30	gcalctool

Less RAM

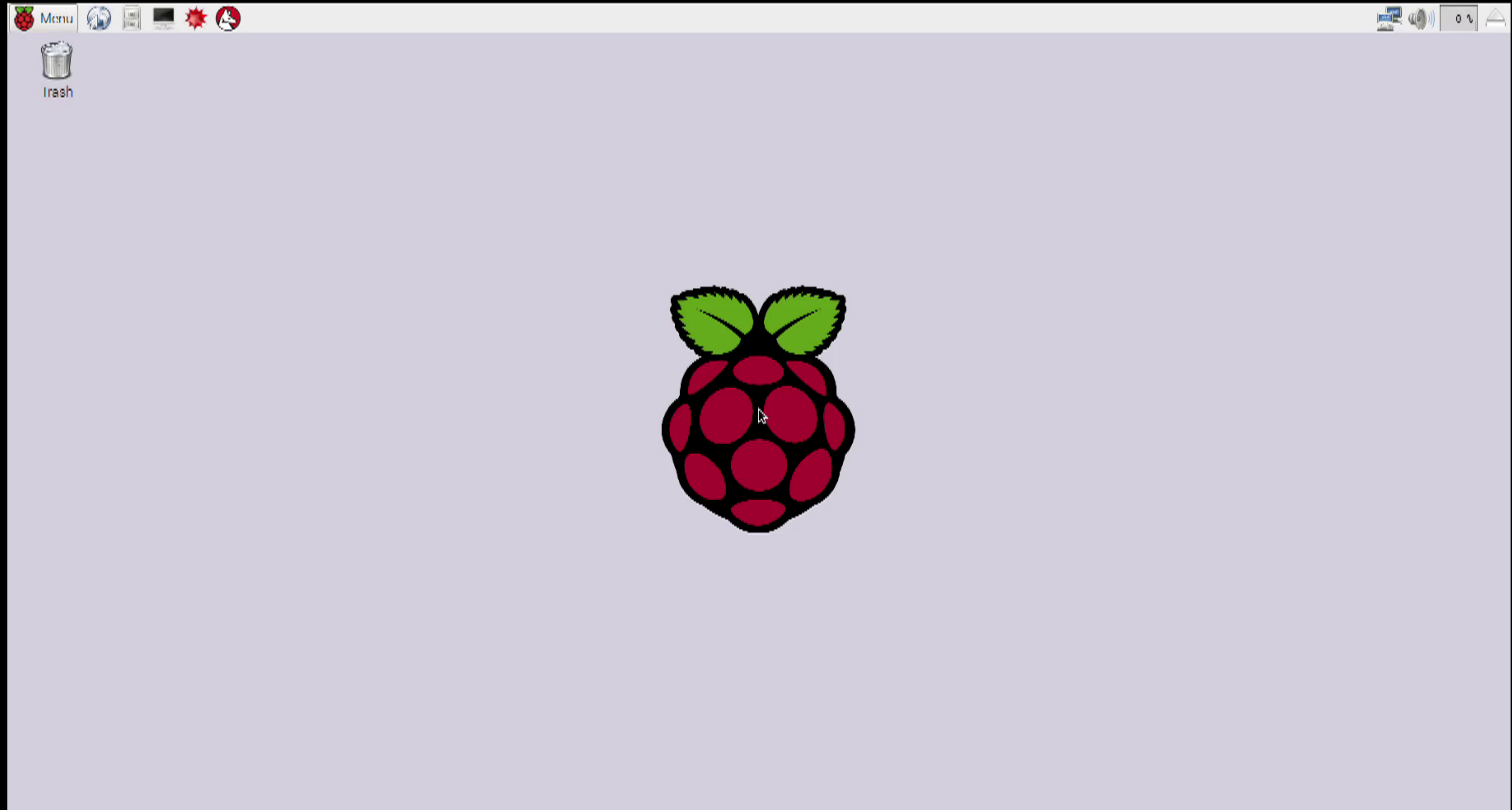
Qt Launch Time (slow)



Native/IUP Launch Time (fast)



Then Raspberry Pi happened



Oh, yeah. IUP.

IUP

- Product
 - Overview
 - Availability
 - Support
 - Credits
 - Documentation
 - Publications
 - Copyright/License
- Download
 - SVN
- History
 - To Do
 - Comparing
- Gallery
 - Screenshots
- Guide
- Tutorial
- System
- Attributes
- Events and Callbacks
- Dialogs
- Layout Composition

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

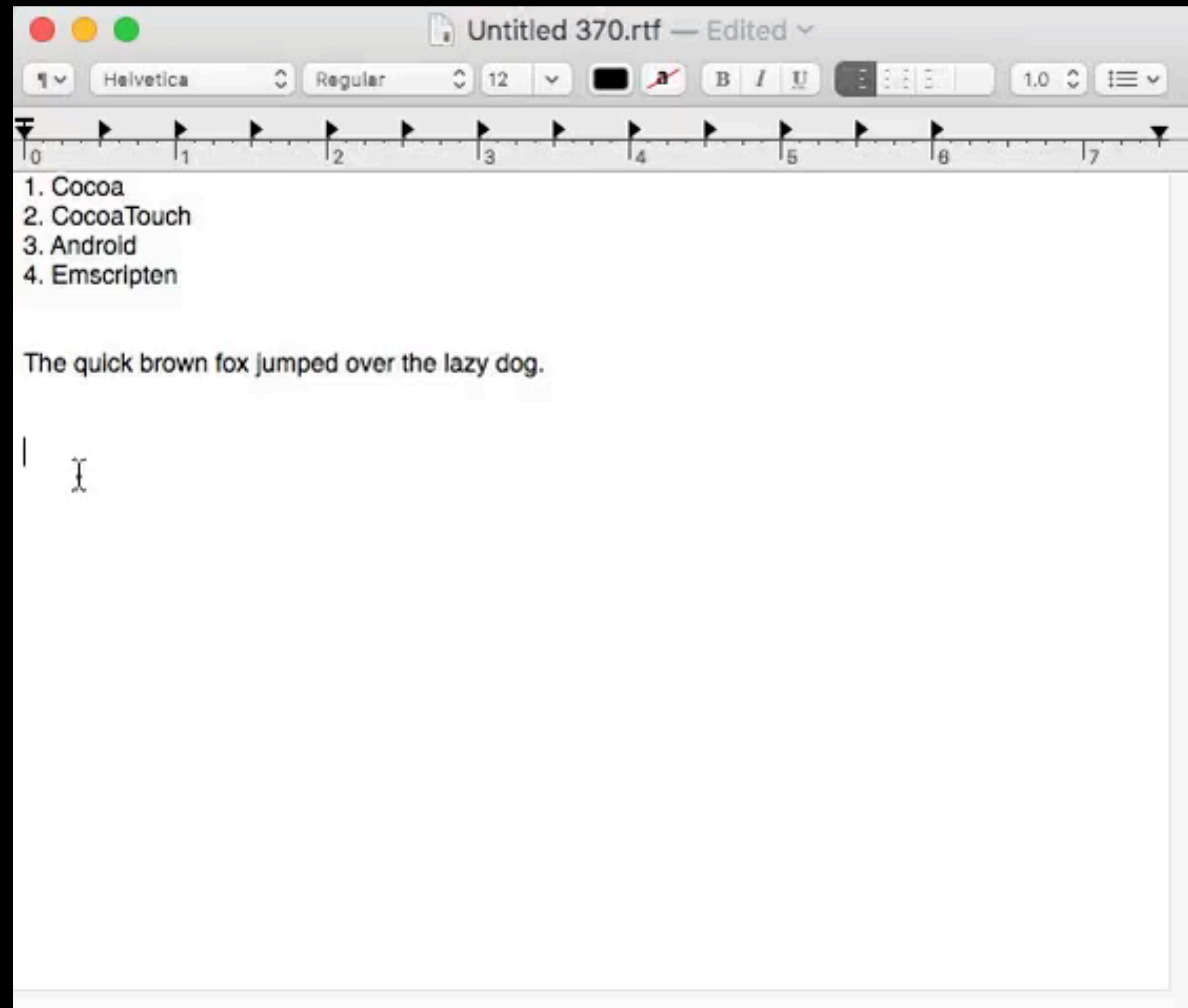
<http://www.tecgraf.puc-rio.br/iup>

Also available at <http://iup.sourceforge.net/>

Why does native UI matter?

- Already mentioned RAM & Performance
- Also usability conventions

Cocoa Discontinuous Text Selection

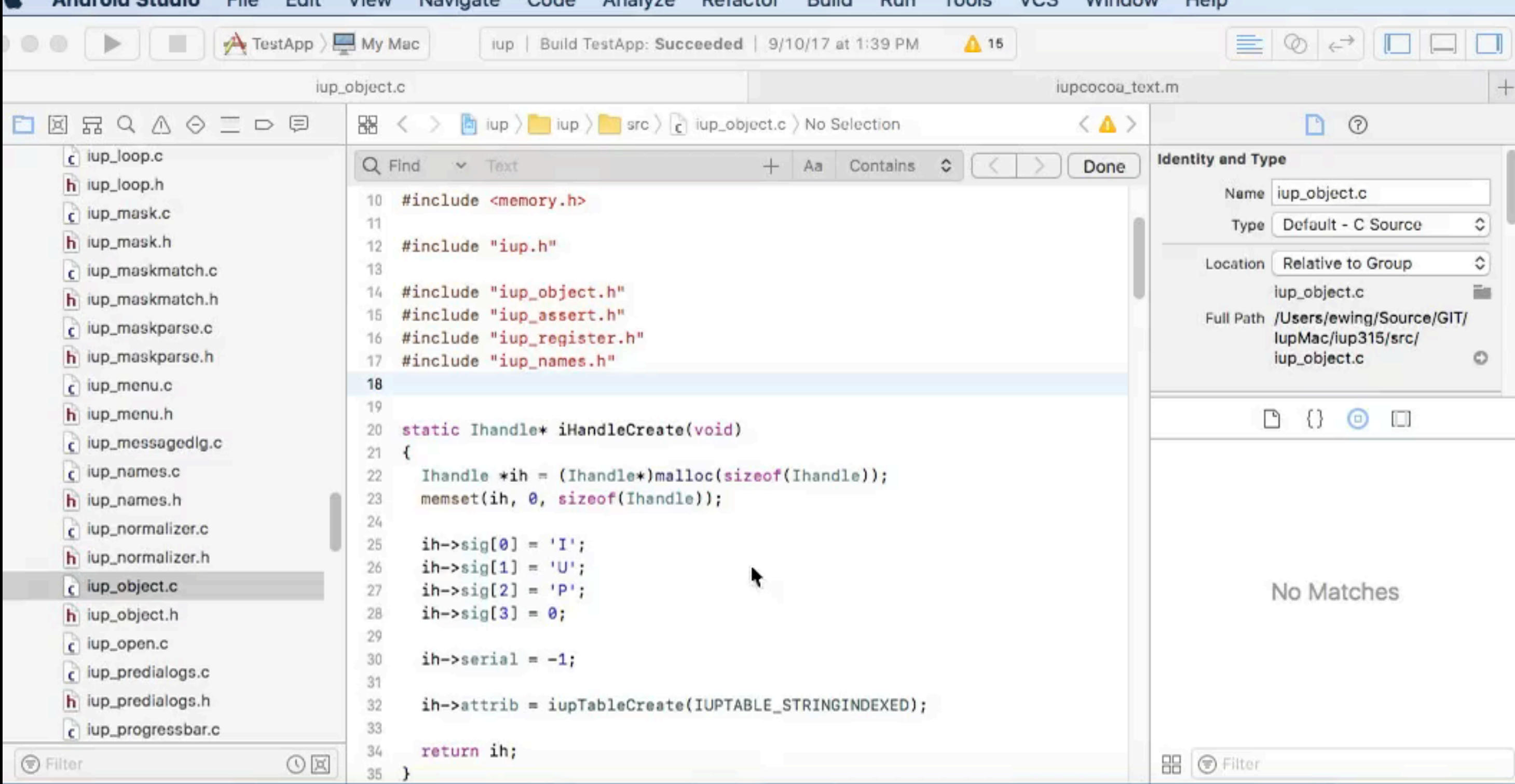


1. Can select text vertically

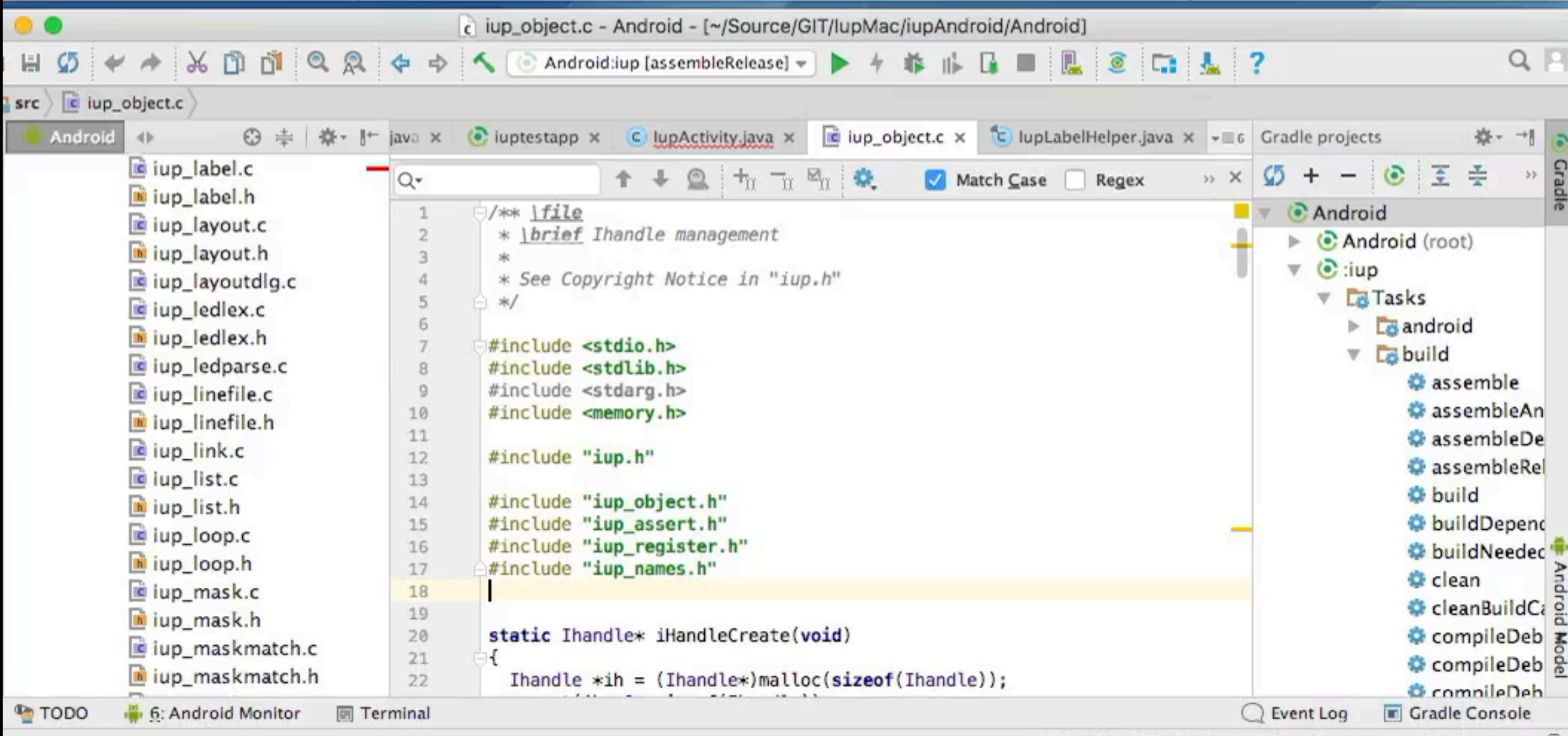
1. Can cut & paste

2. Can select multiple discontinuous sections

1. Can cut & paste



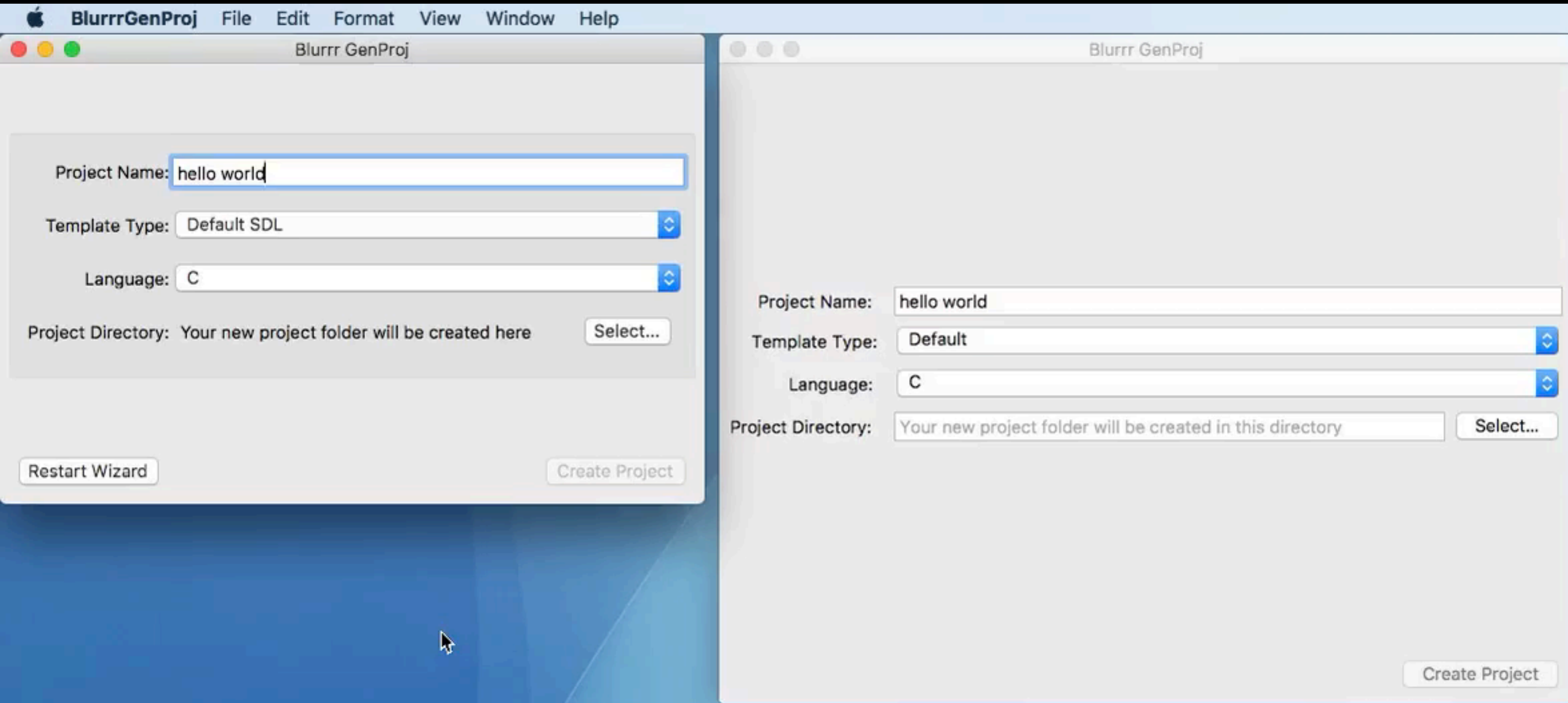
Find Buffer:
Cmd-E, Cmd-G
Native (top)
vs.
Java Android Studio (bottom)



1. Cmd-E to put into Find Buffer
2. Cmd-G to find next

Services & Menu Built-ins

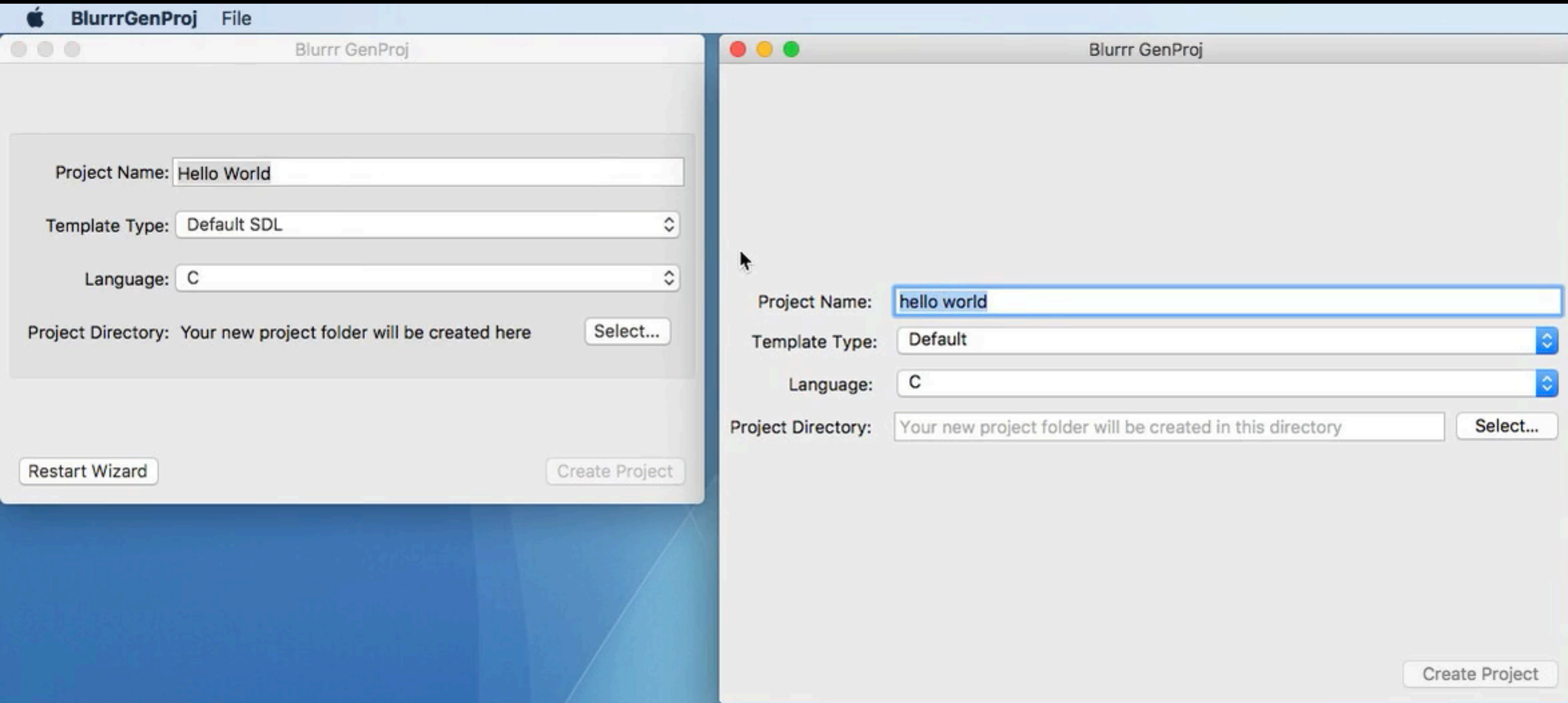
Native (left) vs. Qt (right)



1. Make Upper Case

Services & Menu Built-ins

Native (left) vs. Qt (right)



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

The image is a collage of several screenshots from a Mac OS desktop, illustrating the transition from accessibility to display settings. The central focus is the 'System Preferences' window, which is open to the 'Displays' pane. The 'Accessibility' pane is also visible, showing various options like 'VoiceOver', 'Zoom', and 'Reduce Motion'. Other windows include the 'Blurr GenProj' project wizard, the 'Android Studio' IDE with a project structure view and an event log, and a Slack chat window showing a discussion about installing an application on a Debian system.

System Preferences - Displays

- Resolution: 1440 x 900
- Refresh Rate: 60 Hz
- Color: Default
- Resolution Scaling: Default
- Resolution Scaling: 100%
- Resolution Scaling: 125%
- Resolution Scaling: 150%
- Resolution Scaling: 175%
- Resolution Scaling: 200%
- Resolution Scaling: 250%
- Resolution Scaling: 300%
- Resolution Scaling: 400%
- Resolution Scaling: 500%
- Resolution Scaling: 600%
- Resolution Scaling: 800%
- Resolution Scaling: 1000%

System Preferences - Accessibility

- VoiceOver: Off
- Zoom: Off
- Reduce Motion: Off
- Reduce Transparency: Off
- Reduce Visual Feedback: Off
- Reduce System Sounds: Off
- Reduce Animations: Off
- Reduce Motion: Off
- Reduce Transparency: Off
- Reduce Visual Feedback: Off
- Reduce System Sounds: Off
- Reduce Animations: Off

Blurr GenProj

Project Name: For maximum portability, avoid spaces & special characters
Template Type: Default SDL
Language: C
Project Directory: Your new project folder will be created here [Select...]
[Restart Wizard] [Create Project]

Android Studio

build.gradle - Android - [~/Source/GIT/...]
Android:iup [assembleRelease]

Project Structure:

- Android
- Project Files
- Problems
- iupFontHelper.java
- iup
 - manifests
 - java
 - br.pucrio.tecgraf.iup
 - br.pucrio.tecgraf.iup (androidTest)
 - br.pucrio.tecgraf.iup (test)
 - cpp
 - iup (Shared Library, ~/Source/GIT/iupMac/iupAndroi
 - iupimglib (Shared Library, ~/Source/GIT/iupMac/iup
 - res
 - iuptestapp
 - iuptestappwebbrowser
 - iupweb
 - manifests
 - java

Event Log:

- 10/12/17 9:58 PM Platform and Plugin Updates: The following components are ready to [update](#): Android Emulator, NDK, Android SDK Platform-Tools, Android SDK Tools
- 9:58 PM Gradle sync started
- 9:59 PM Gradle sync completed

Slack - #general

40+ new messages since 12:46 PM [Mark as read]

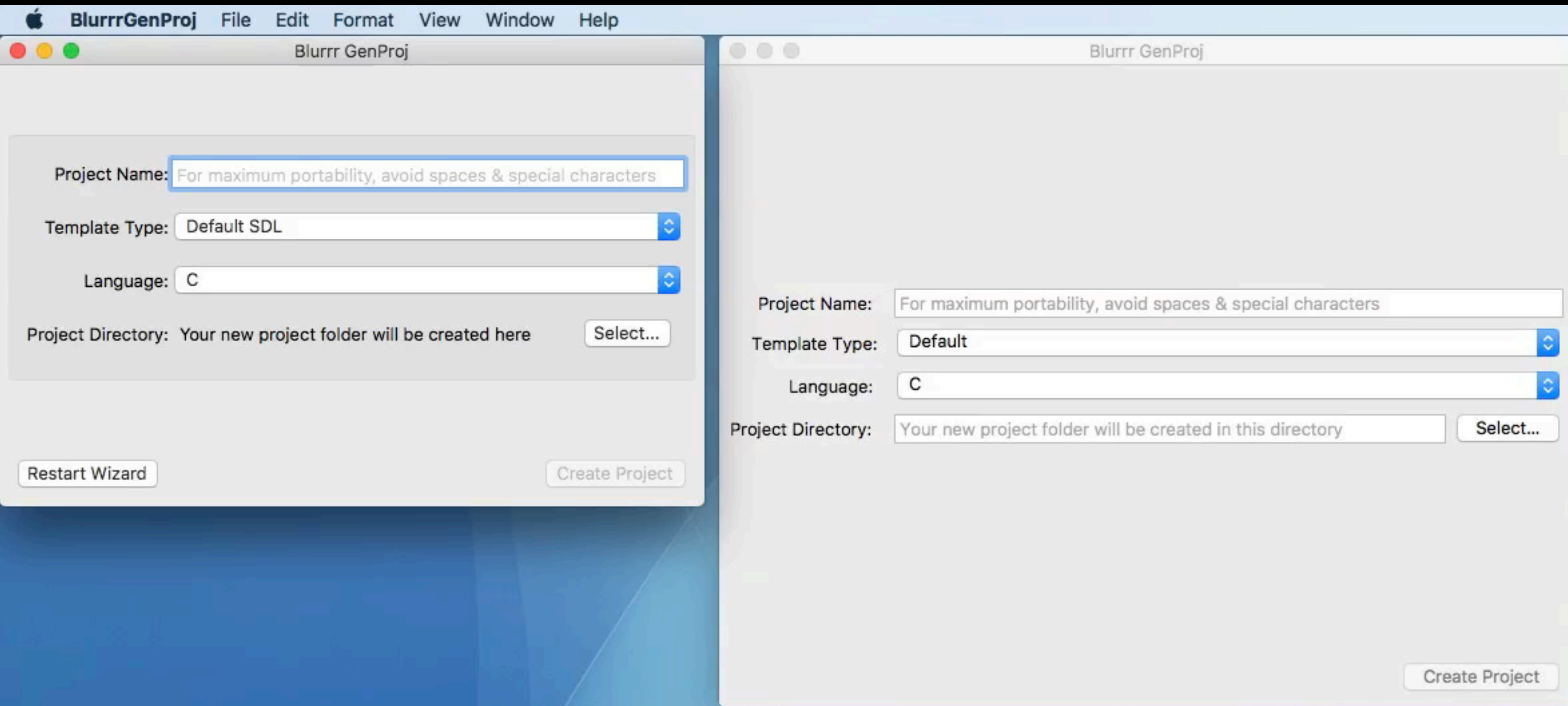
easy to convert

hpux735 5:58 PM cool!

RNeese 5:58 PM you apt-get install alien rename the swift-3.1.1.tgz to .tar.gz alien -v swift-3.1.1.tar.gz and it does the work so then you can push ut debs

hpux735 6:04 PM If I were to make them longer-term I'd probably make a DEBIAN control file the right way. I just don't have the time at the

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 Solutions

- 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:

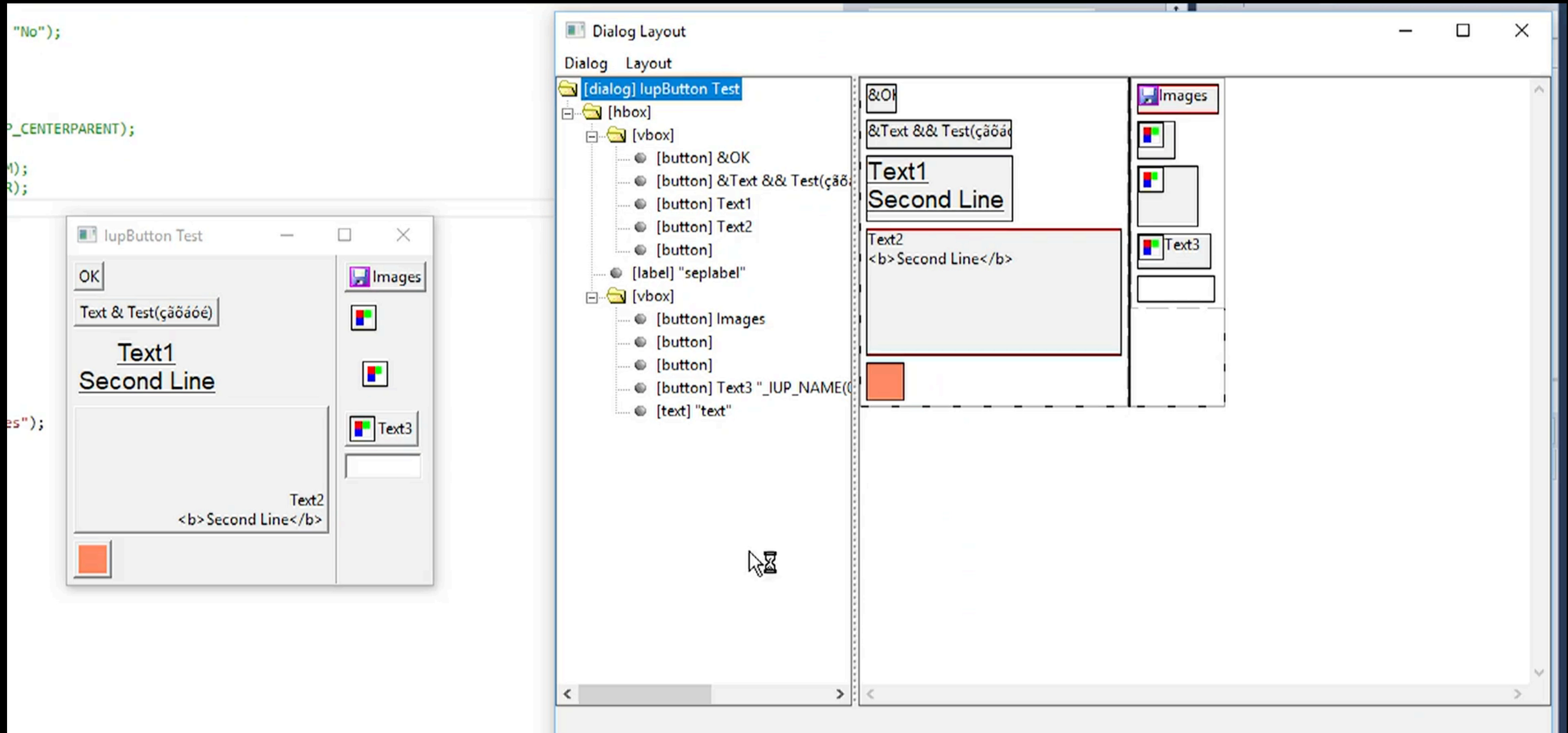
```
btn = button[ACTIVE=NO] ("OK", action_ok)
```

```
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

IupCocoa

- 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

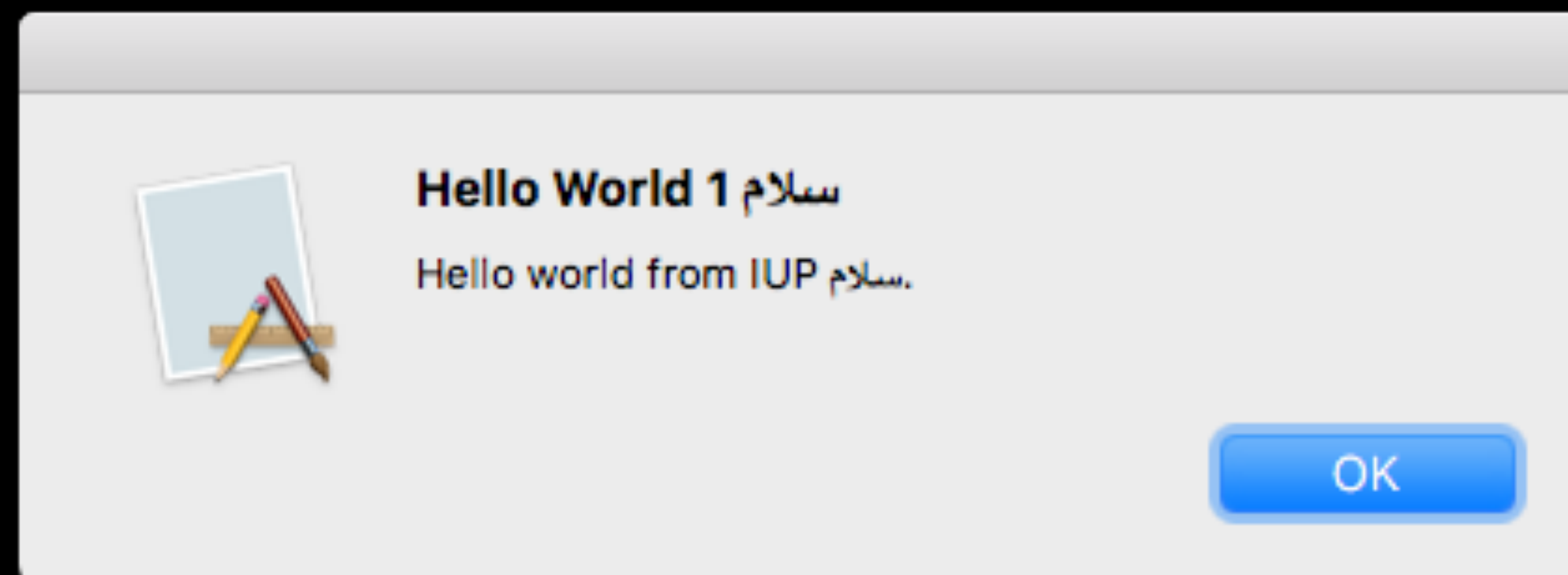
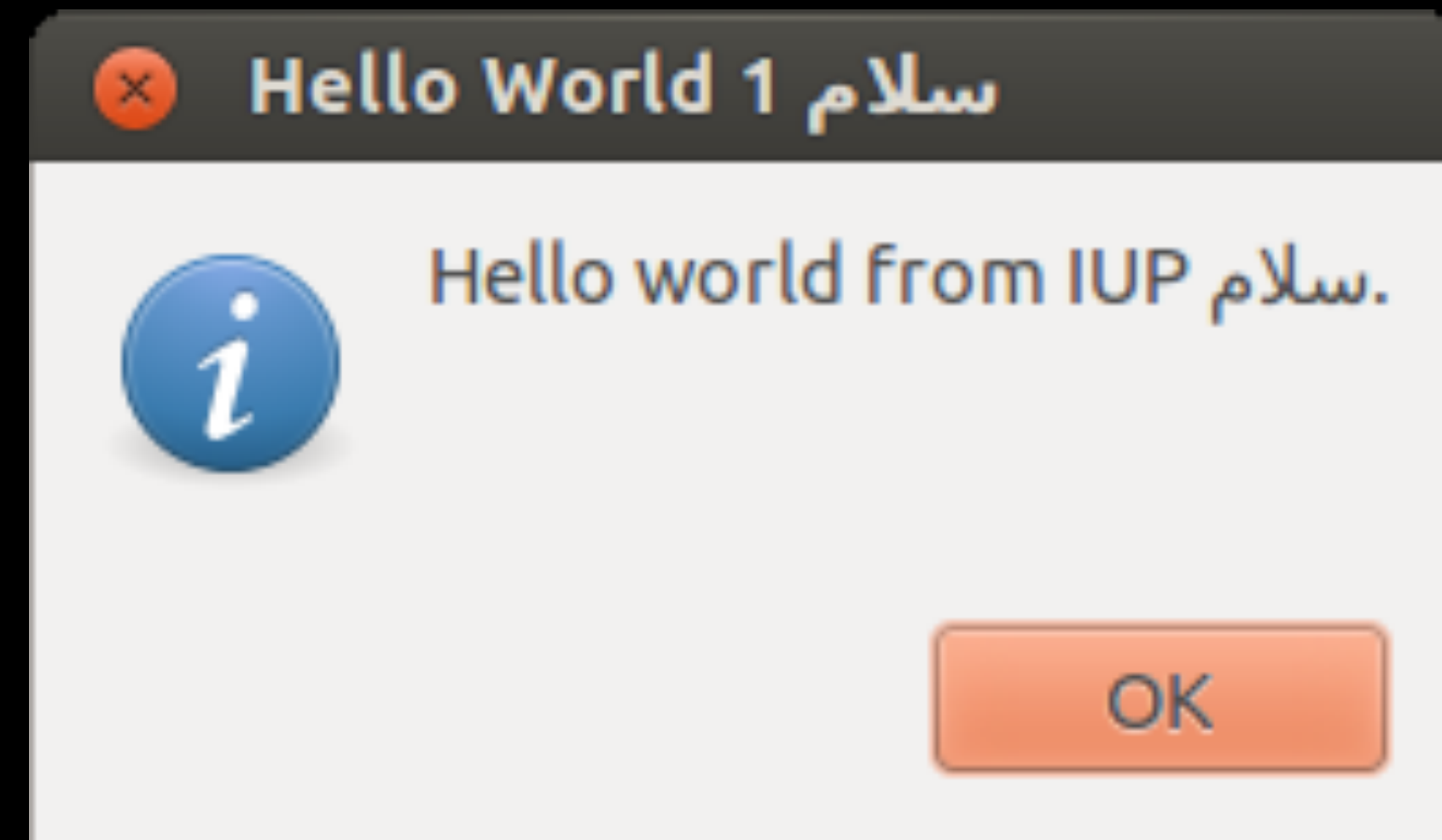
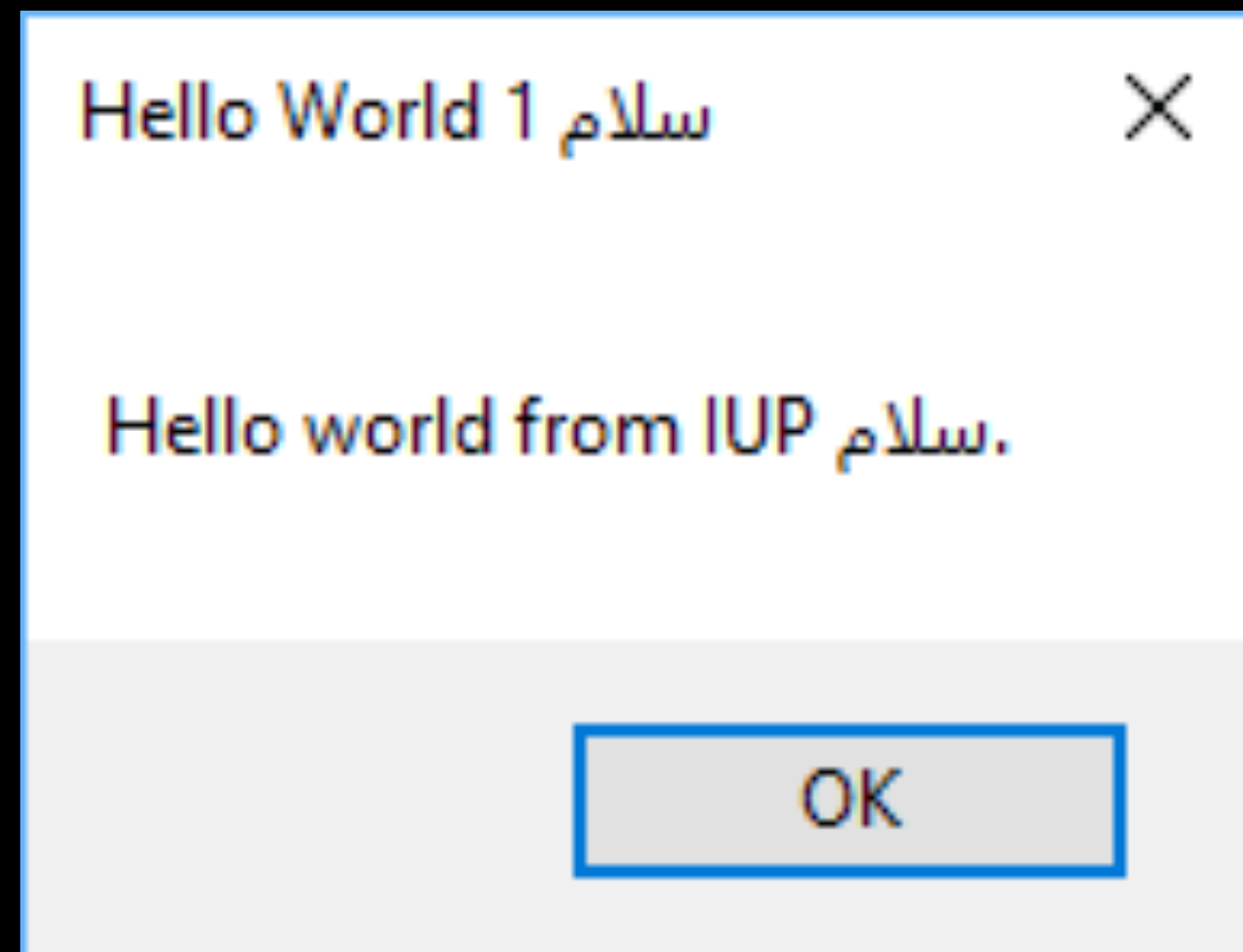
lupCocoa

- 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

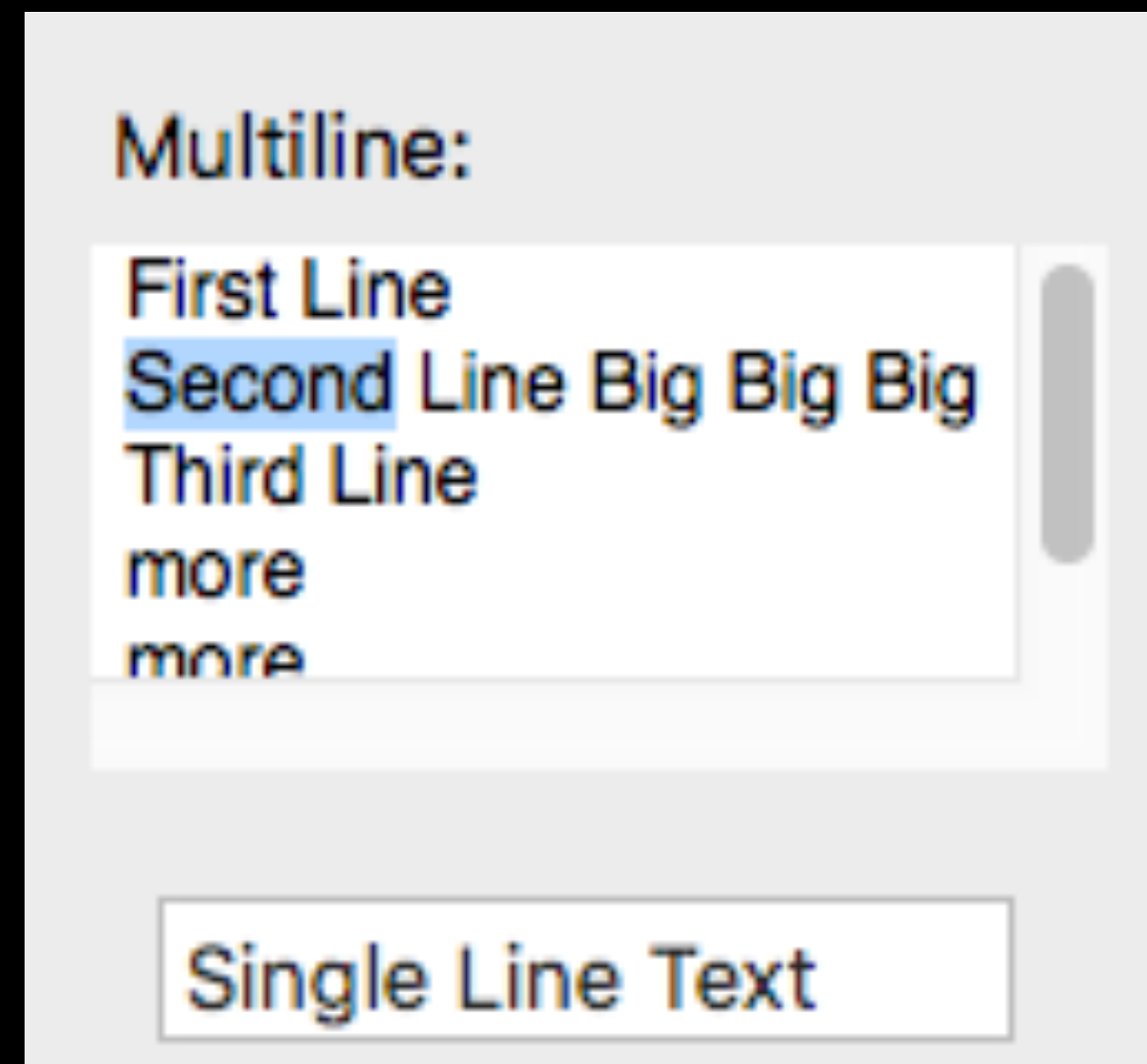
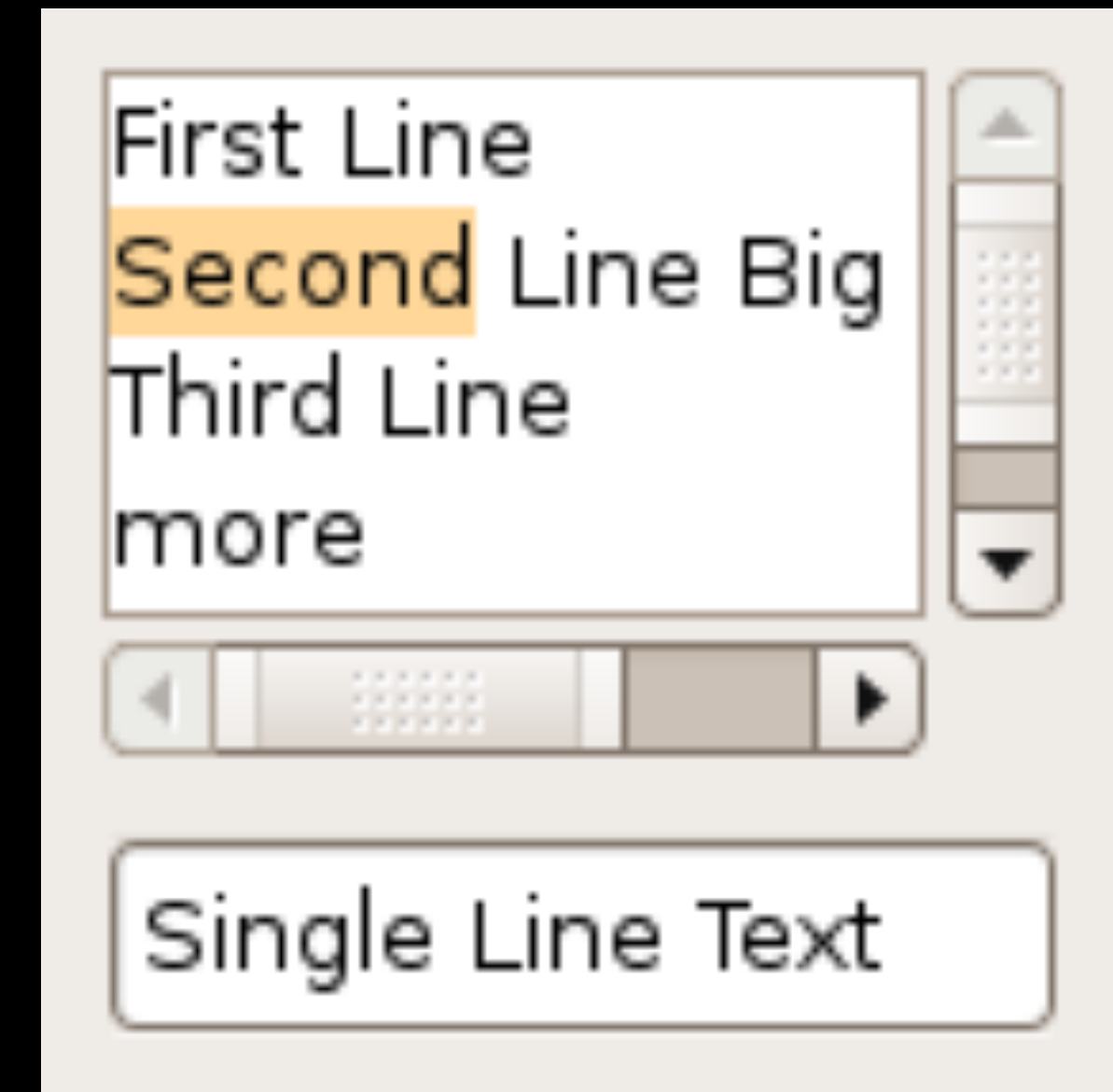
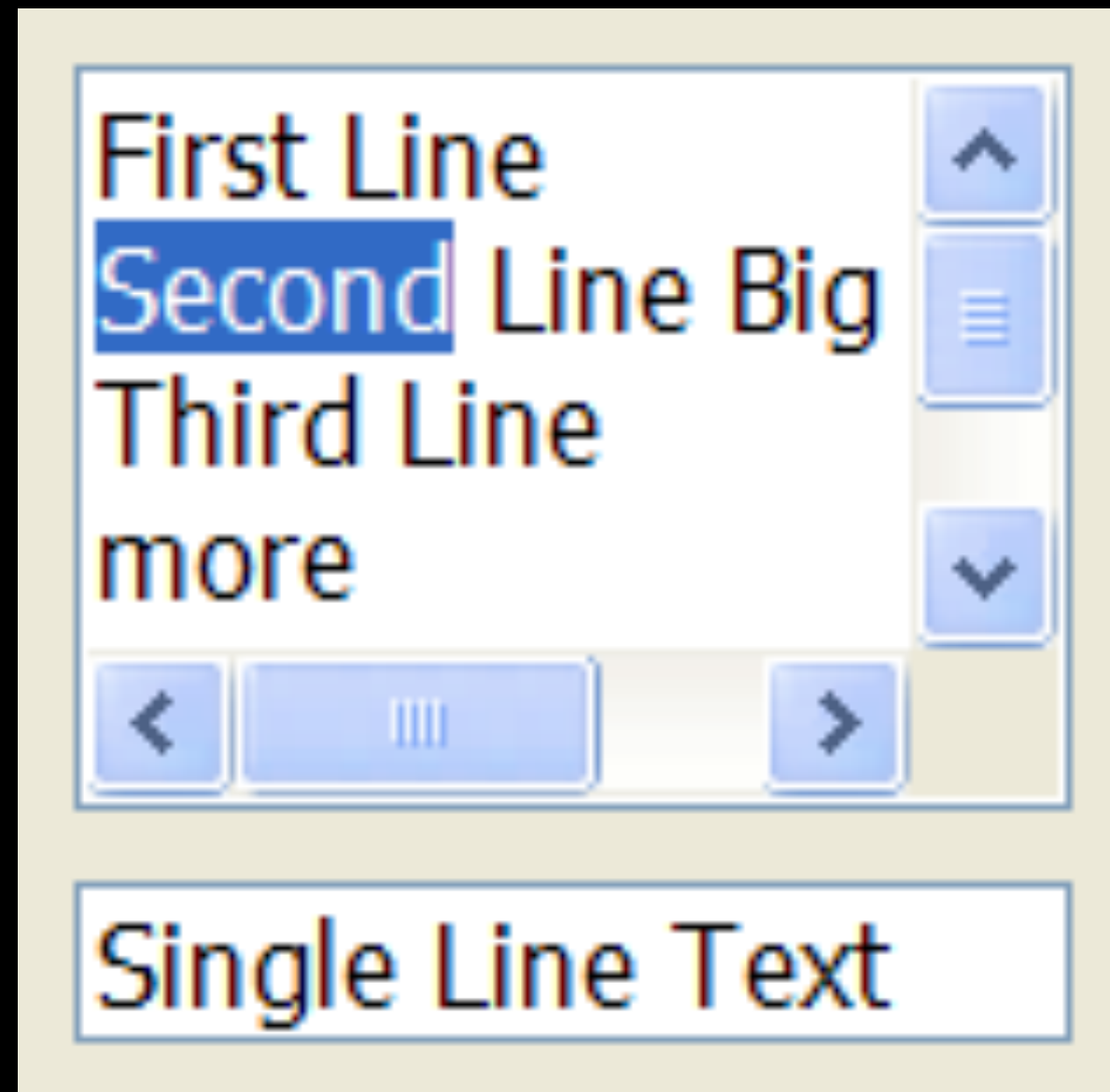
Quick Overview on Implemented things

- 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.)

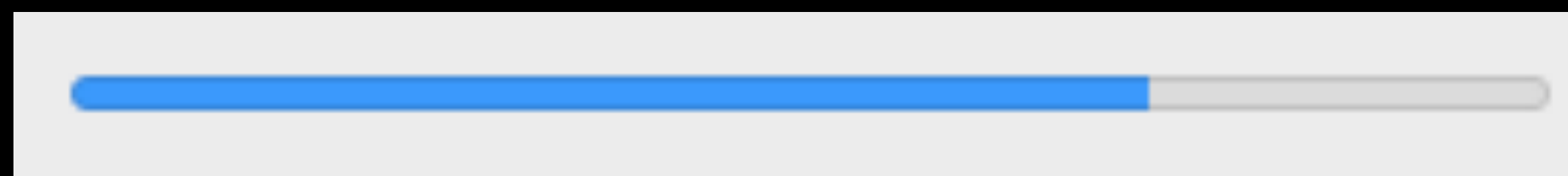
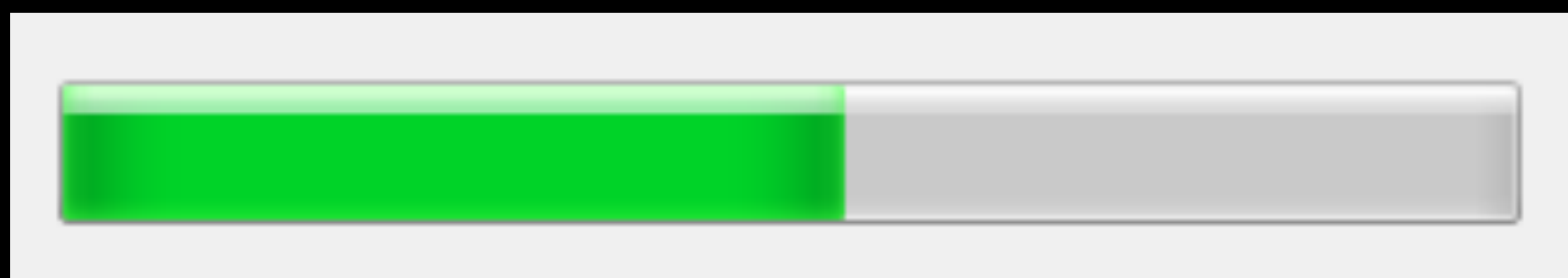
Dialogs, Labels, & Buttons



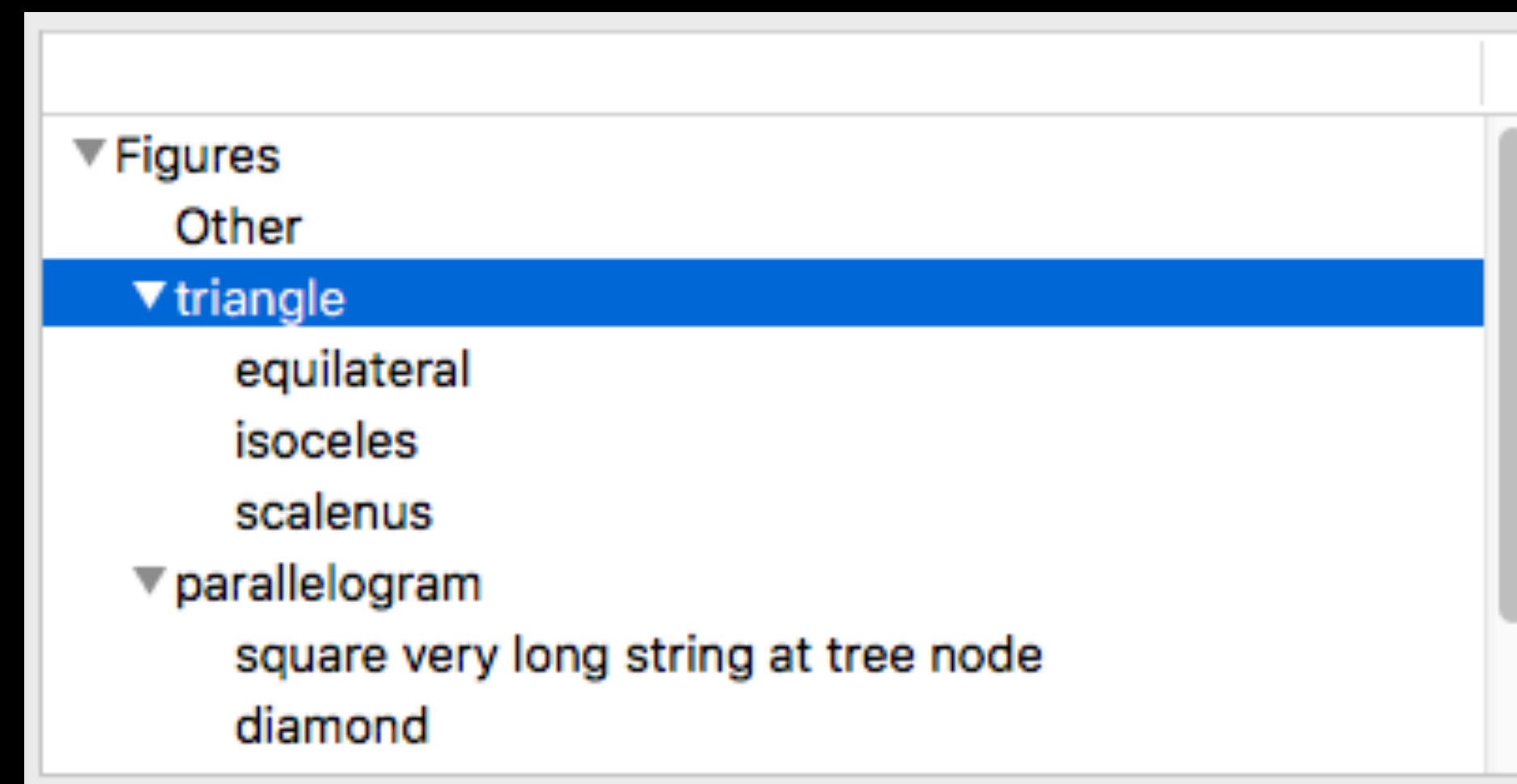
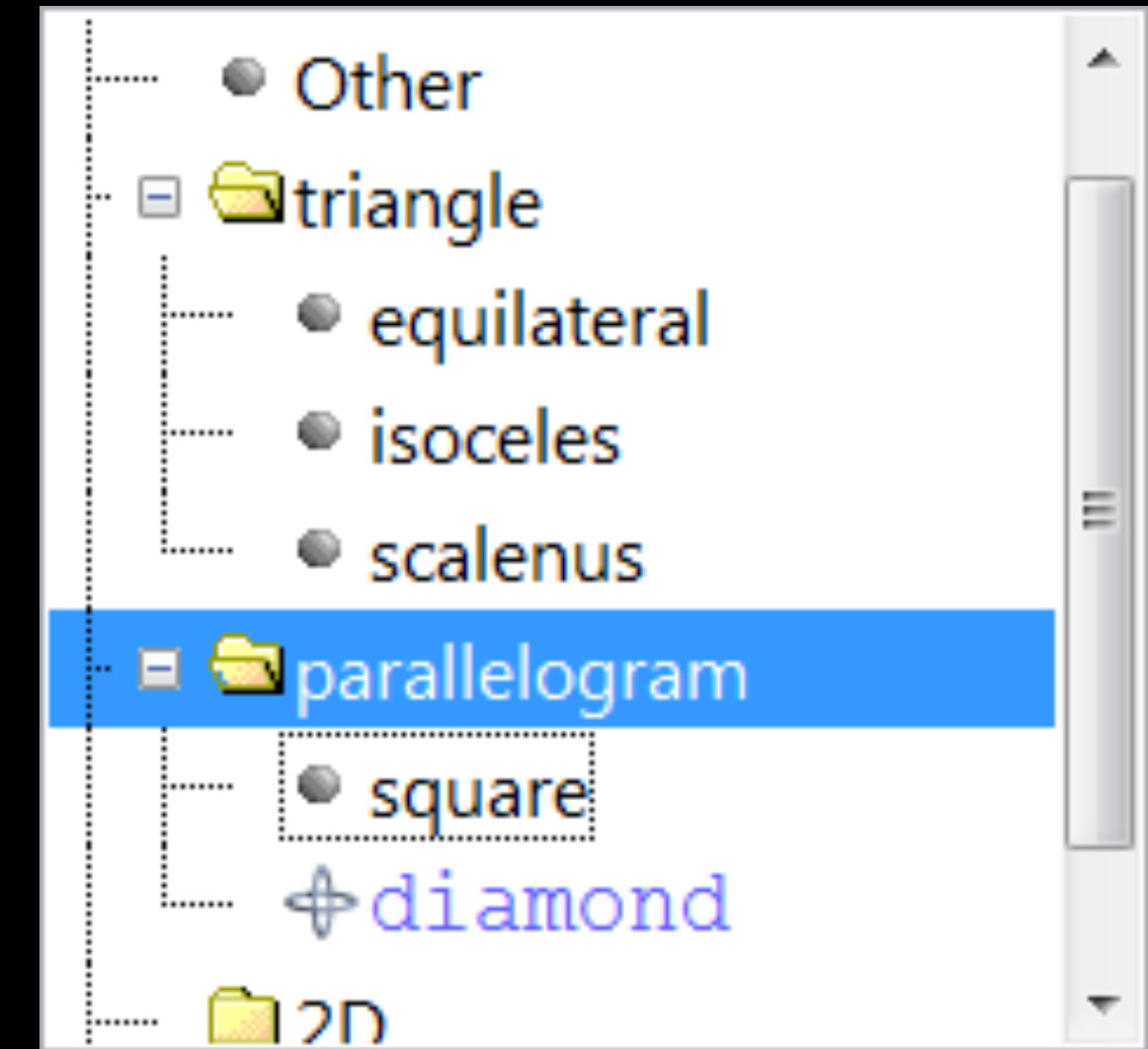
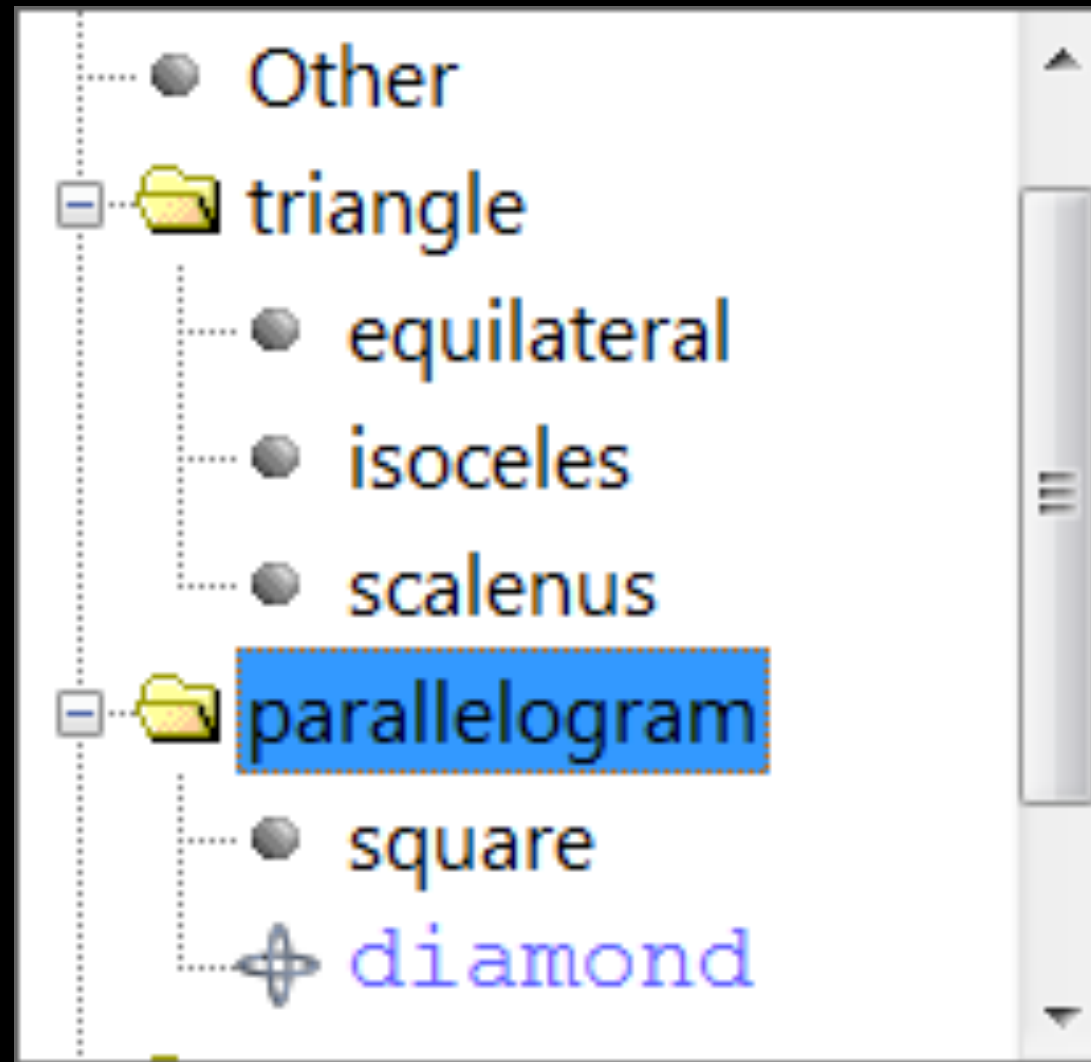
(Input) Text



Progress Bar



Tree



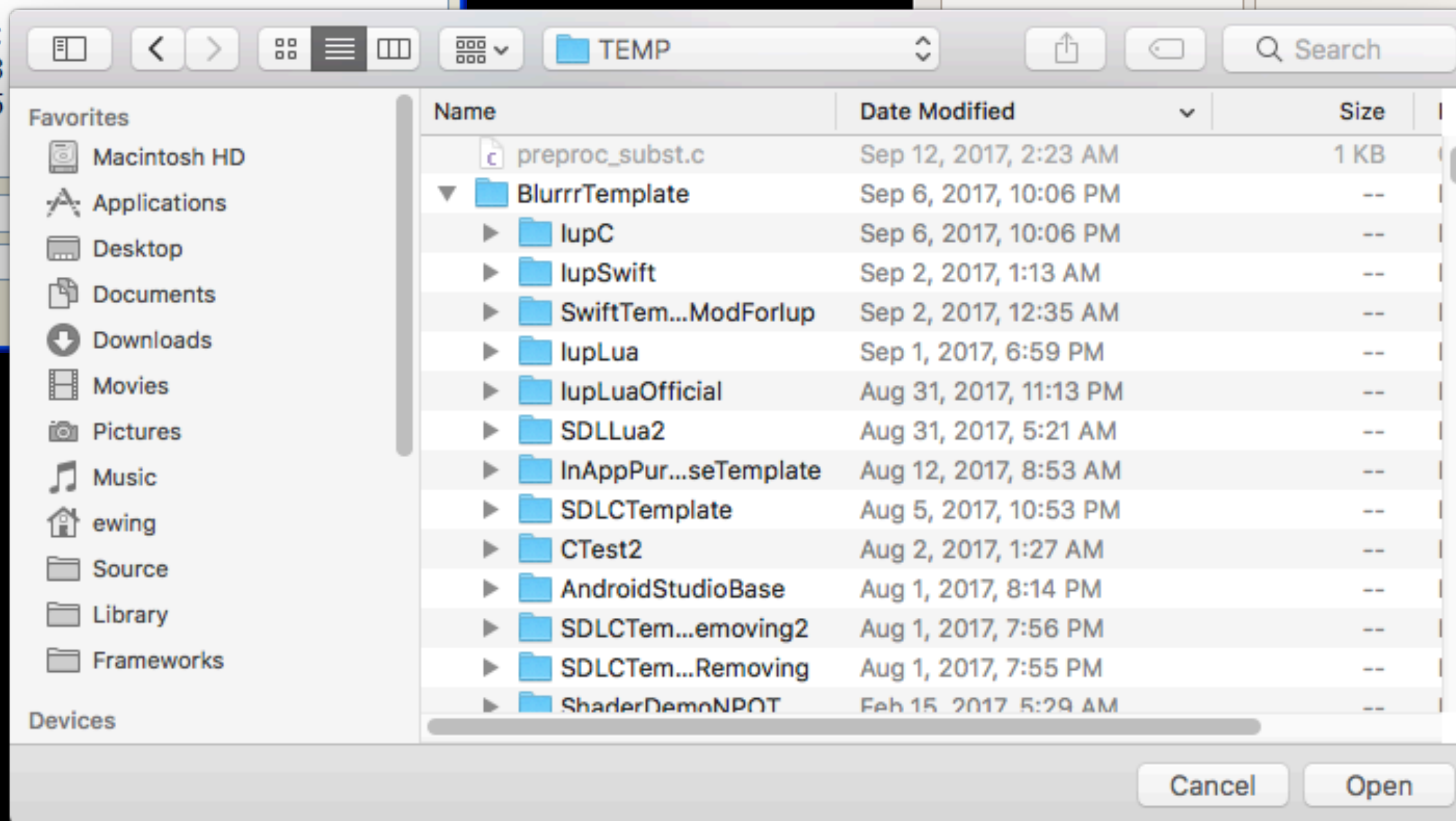
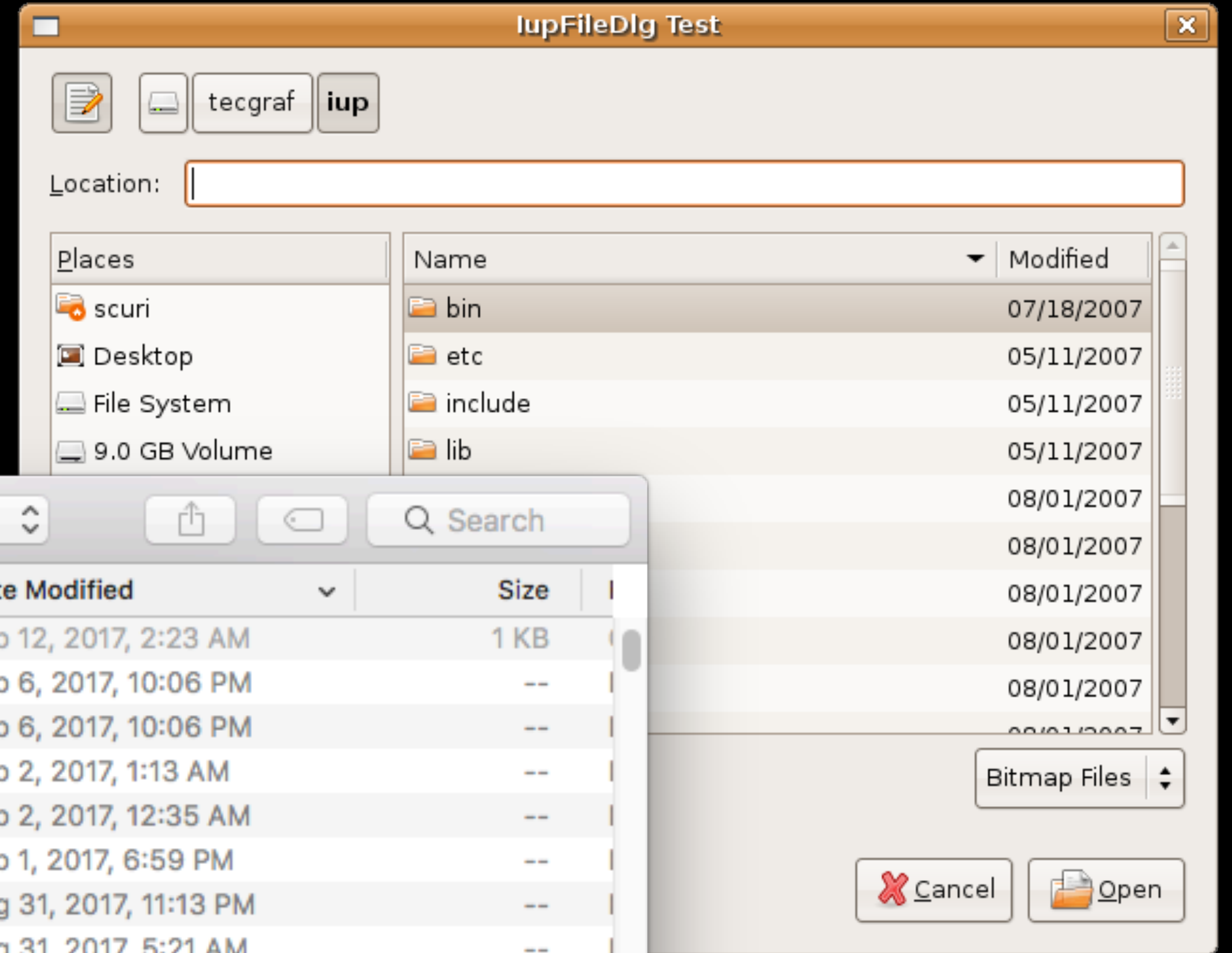
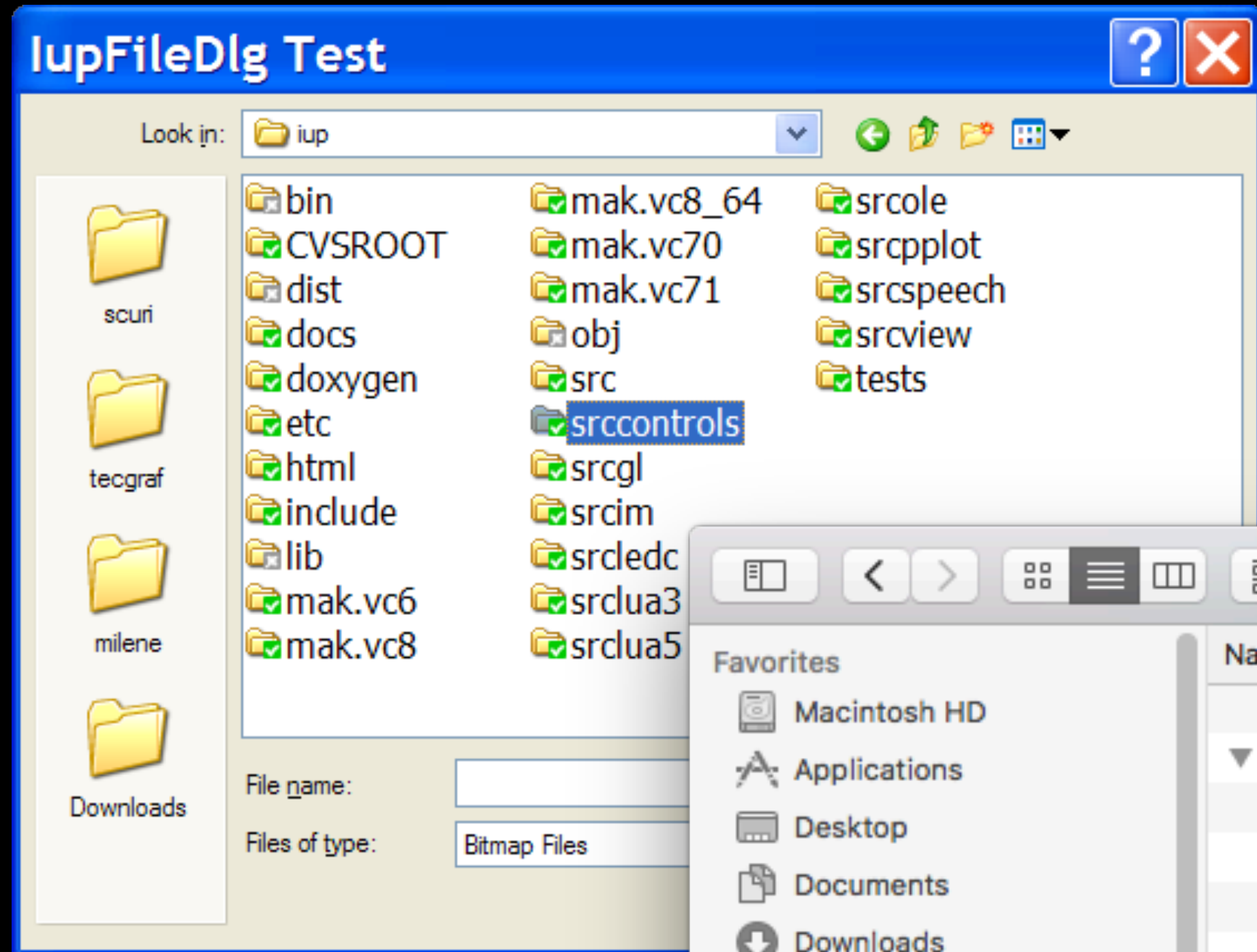
Calendar

November 2015						
Mo	Tu	We	Th	Fr	Sa	Su
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

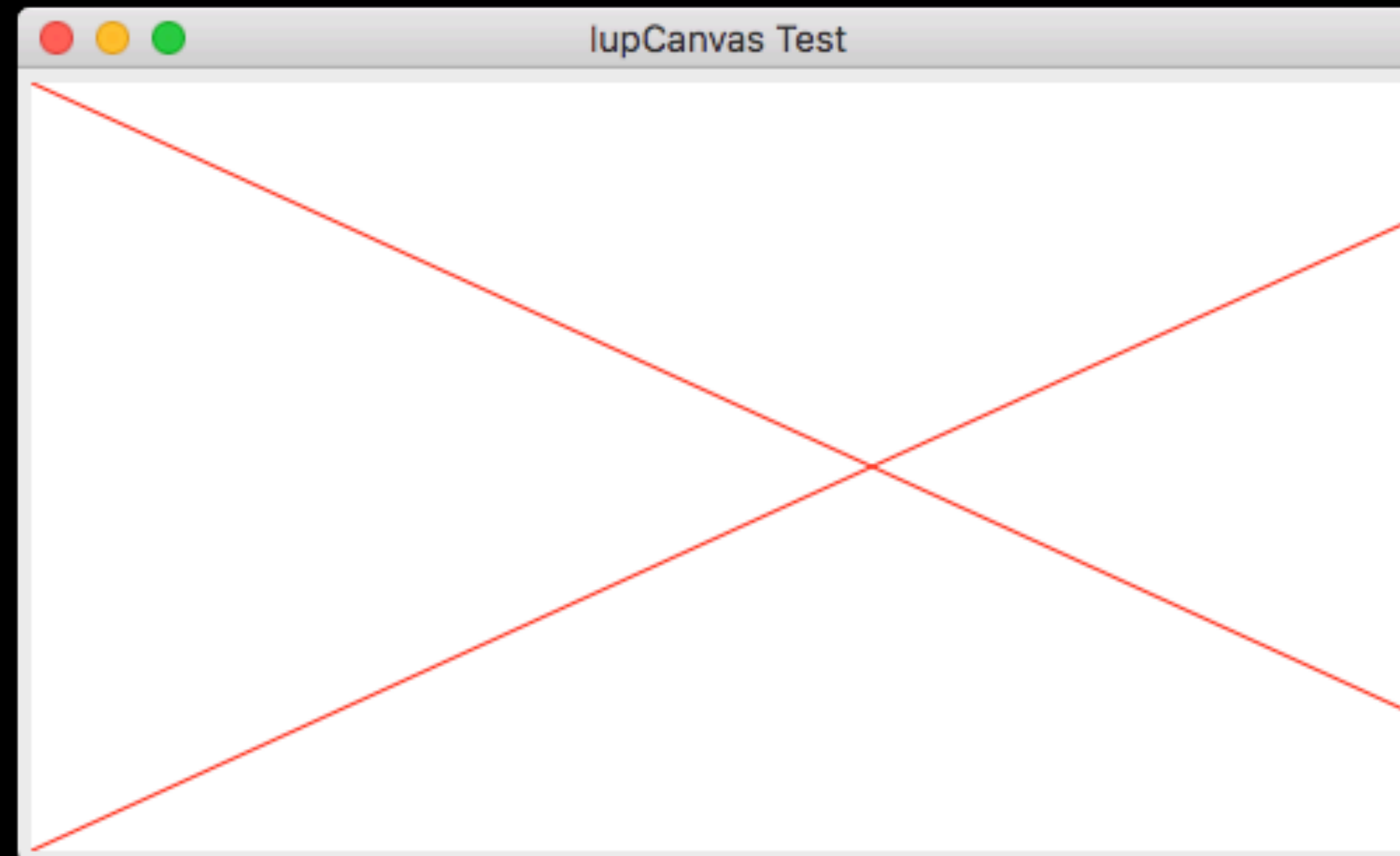
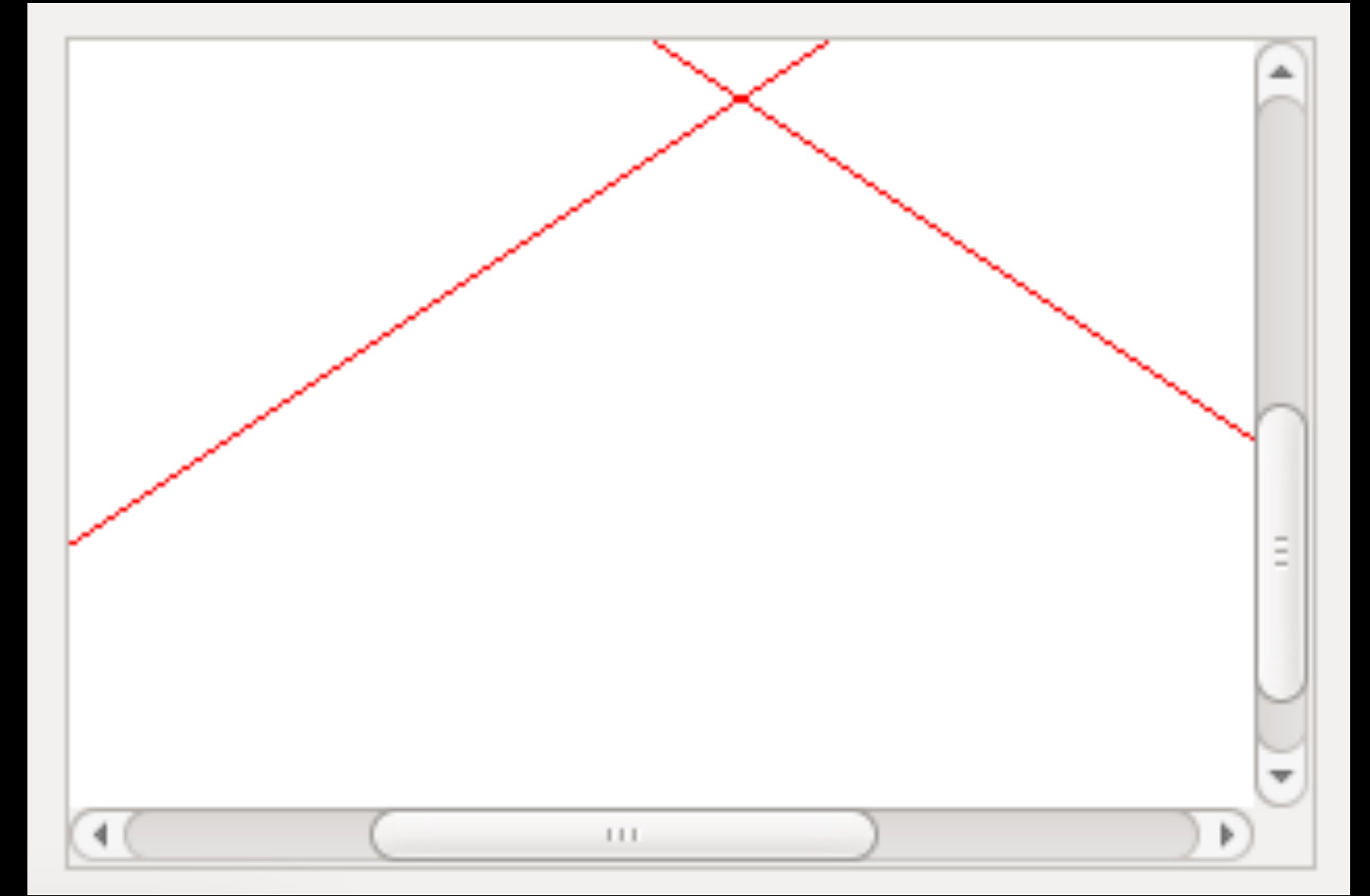
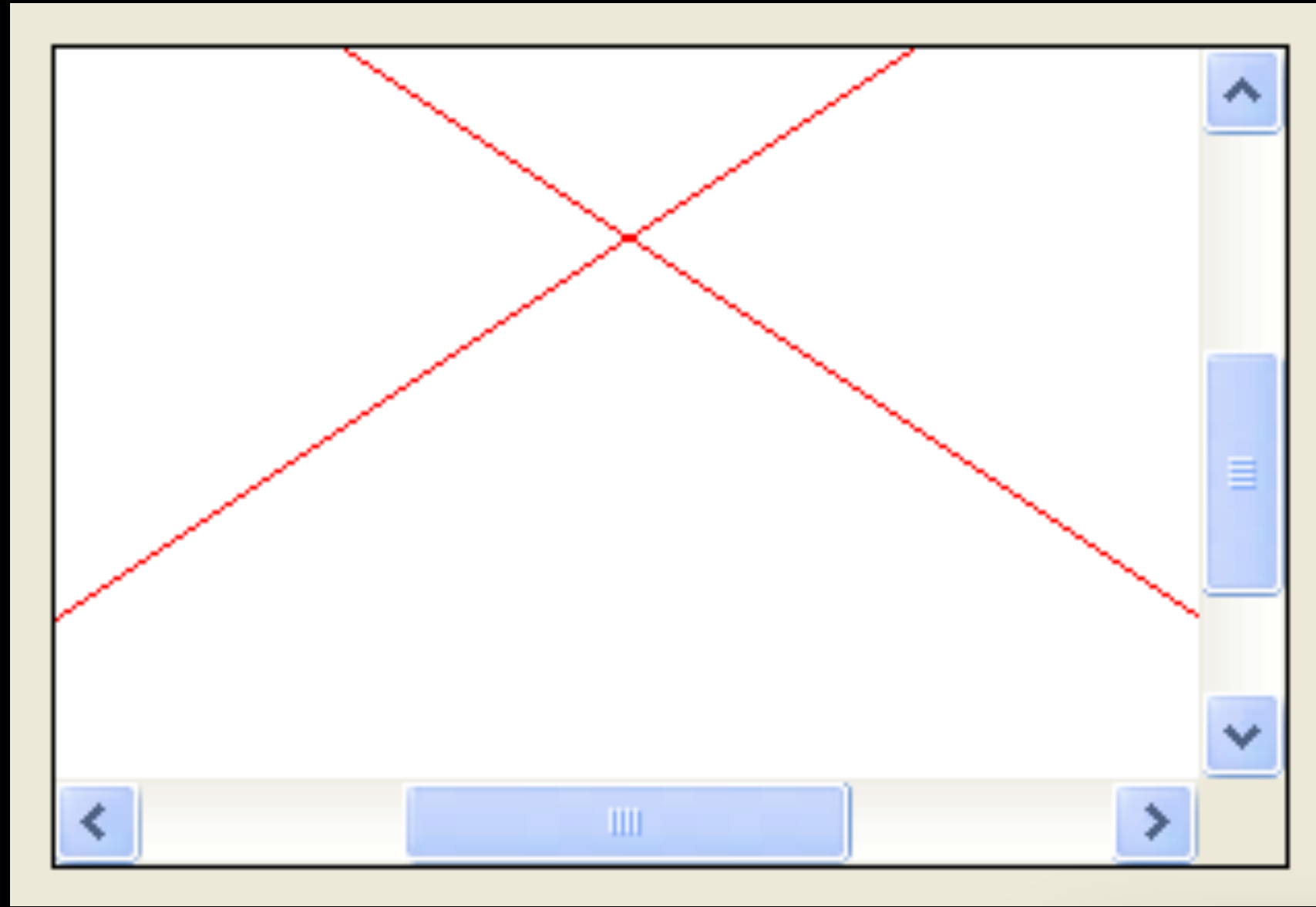
November						2015
Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5
6	7	8	9	10	11	12

Oct 2017						
Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

File Dialogs



Canvas



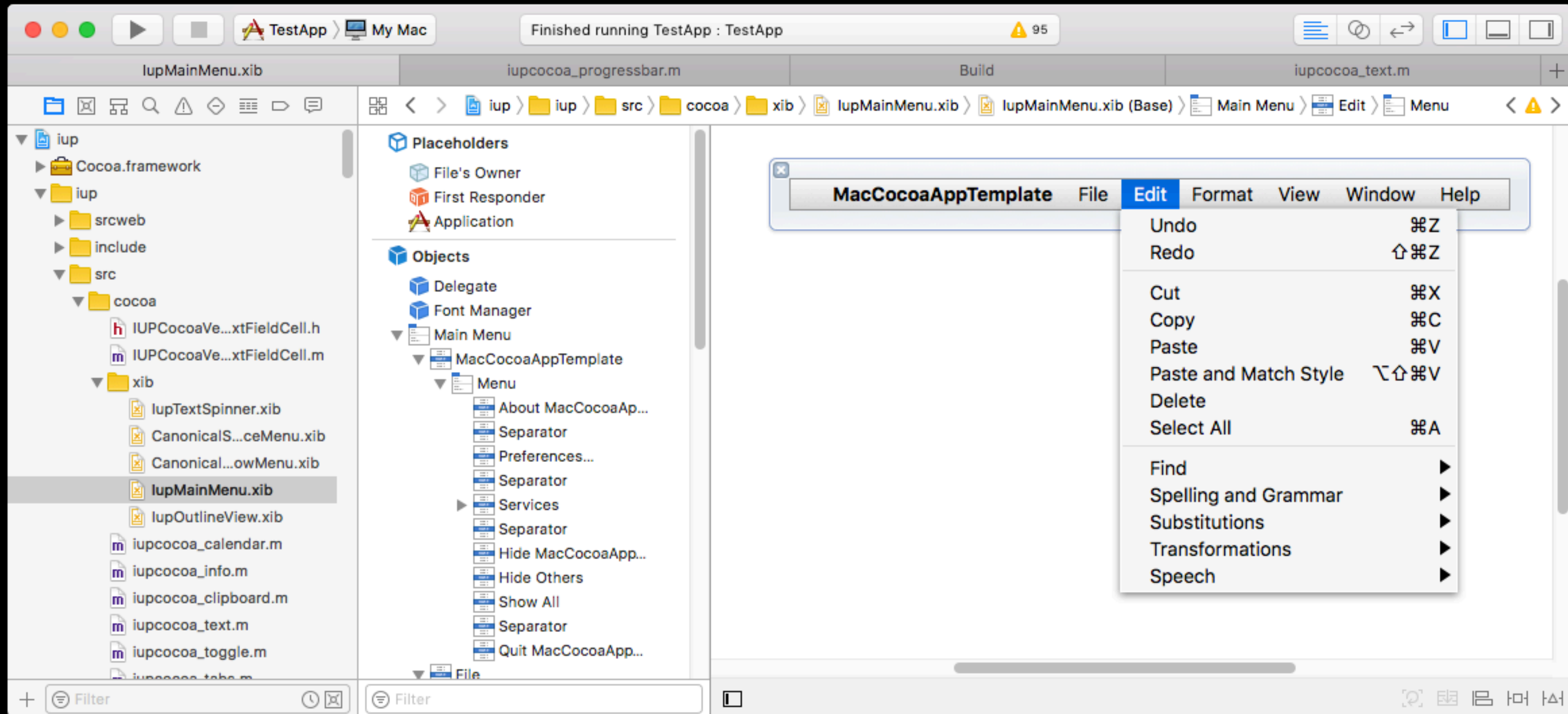
Some Impedance Mismatches

- 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

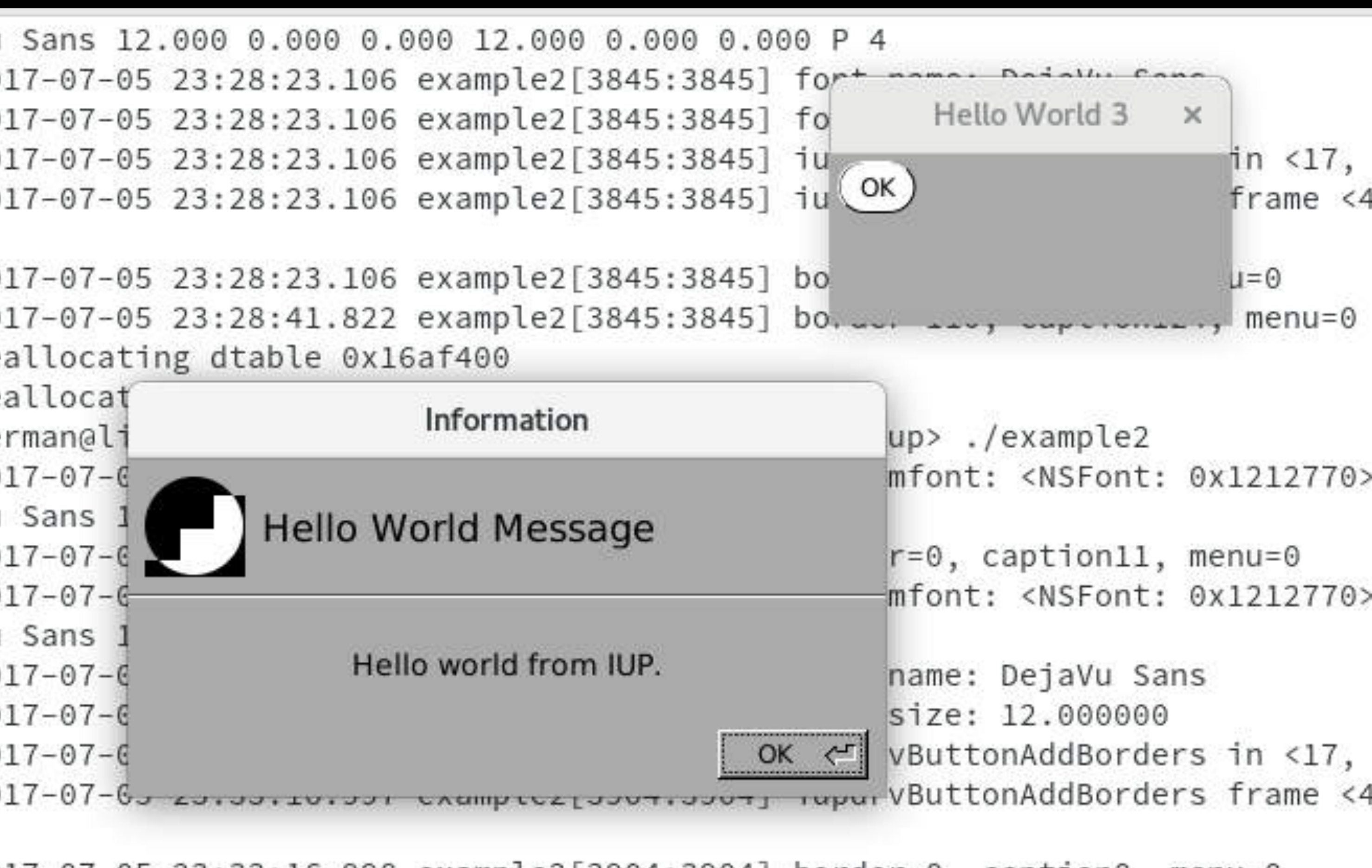
Application Menu



- Using: `IupSetGlobal("MENU", (const char*)main_menu);`
- Instead of per-window: `IupSetAttributeHandle(dialog, "MENU", main_menu);`

One More Thing...GNUStep

- Courtesy: Germán Arias



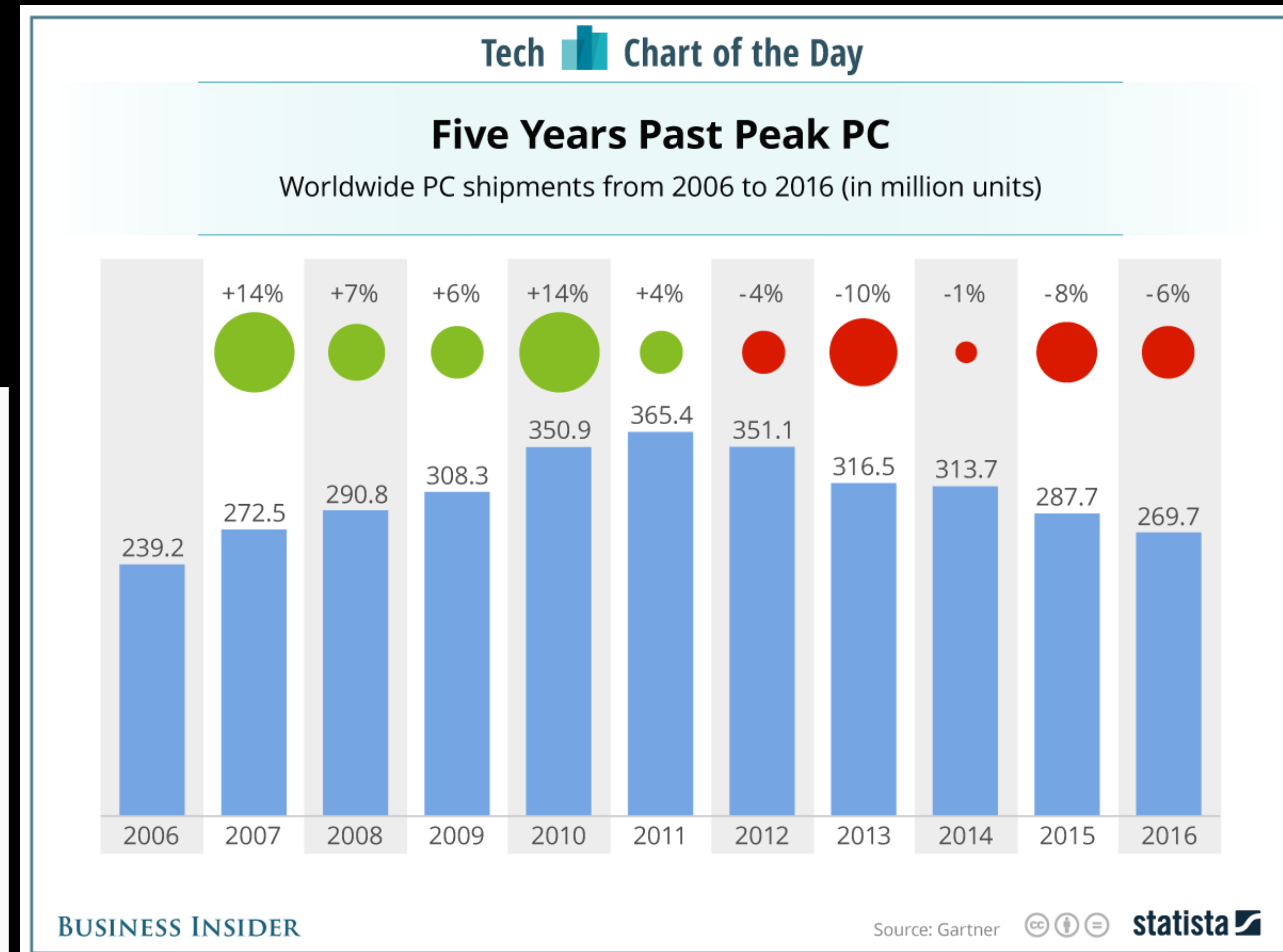
That's lupCocoa

- 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

October 10, 2017 04:41 PM Eastern Daylight Time

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

 Webmaster Central Blog

Official news on crawling and indexing sites for the Google index

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

GAMES

Mobile game revenue finally surpasses PC and consoles

STEPHANIE CHAN @SWEIJUCHAN JULY 13, 2017 11:45 AM

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.

IUP for iOS & Android

- 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?

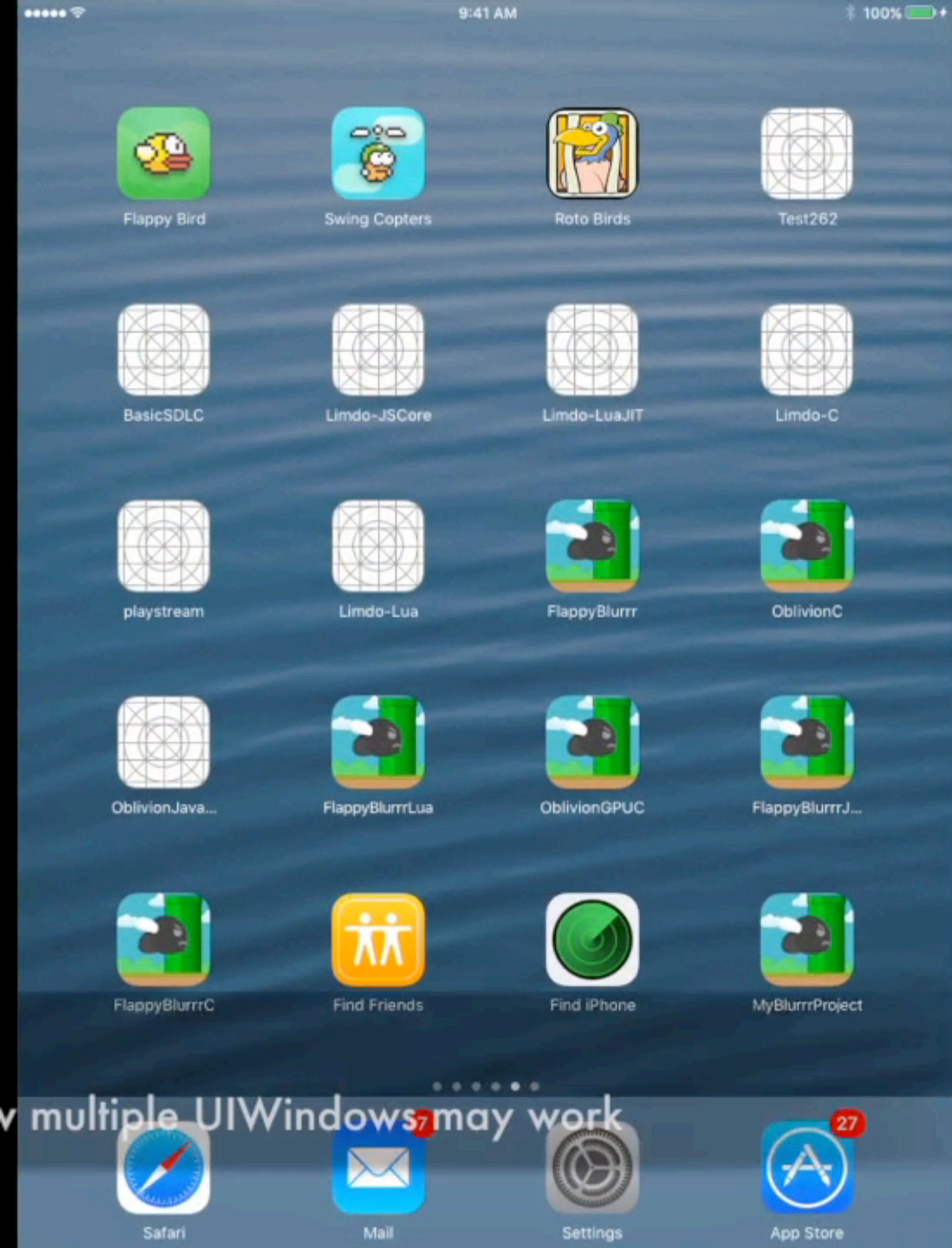
- “Dialog” is a “Window” on Desktop
- iOS: While a UIWindow exists, the paradigm isn’t good for multiple dialogs
 - The more common and useful construct is the UIViewController
- Android: The corresponding construct is an Activity

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 Safari

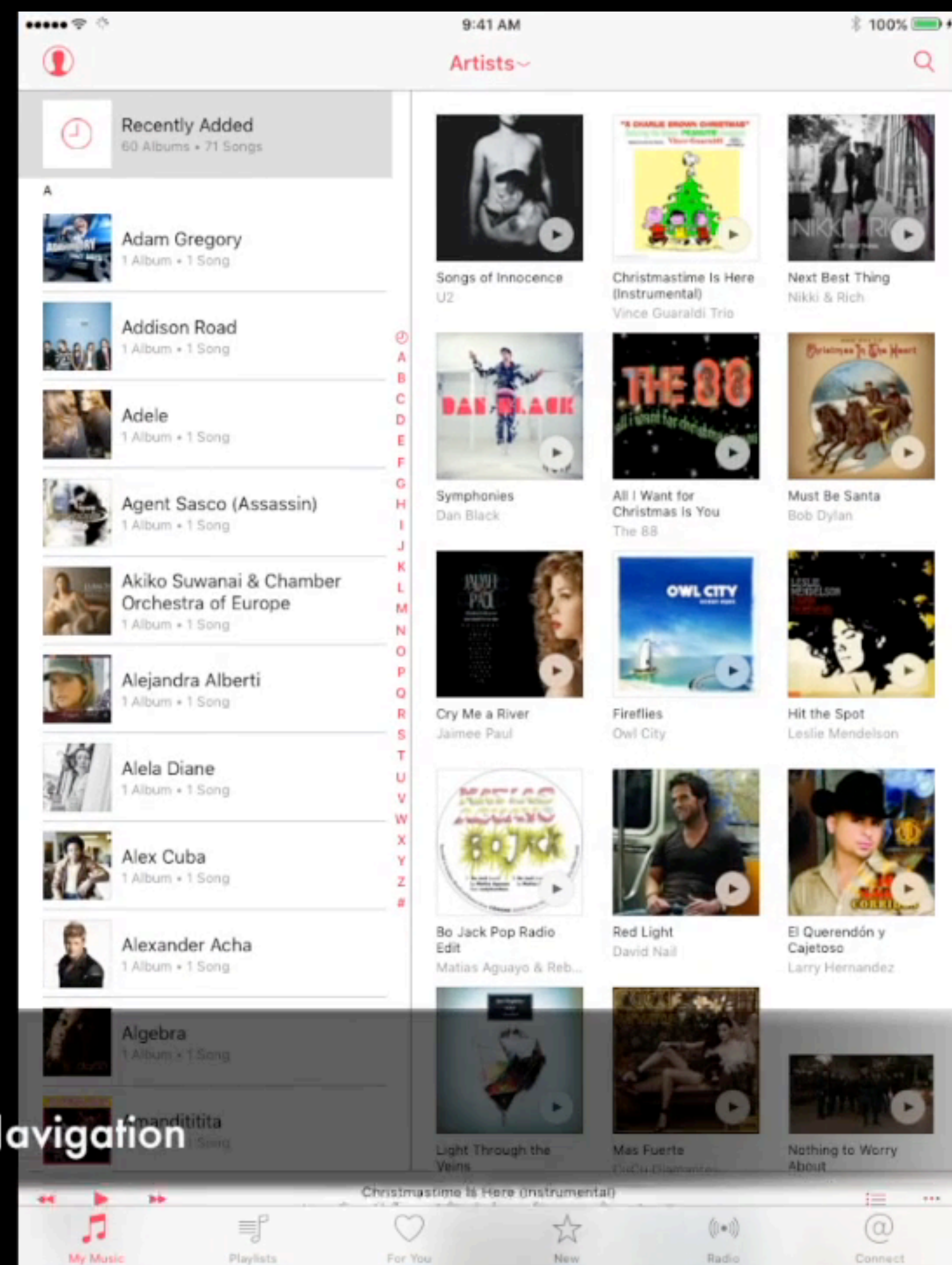
Stand-in for showing how multiple UIWindows may work



iOS UINavigationController & Android Activity

- Both UINavigationController & Activity drill down & back through a stack of views
- Users expect this behavior
- Behavior is built-in

iOS Music app
Example of Drill-down Navigation



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)

*Fun Fact: Written in Swift using bindings to IUP

```
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
}
```

Ubuntu Linux (amd64)

The image shows a screenshot of an Ubuntu Linux desktop environment. The desktop background is a solid orange color. On the left side, there is a vertical dock with several application icons: Dash, Home, Firefox, LibreOffice, and a terminal icon. The top of the screen features a dark grey panel with system status icons (network, volume, power) and the time '7:38 AM'. Two windows are open:

- Terminal Window:** The title bar reads 'Terminal'. The terminal output shows the following commands and progress:

```
pinky@Ubuntu16vb: ~/TEMP/IupSwift/BUILD/Linux
pinky@Ubuntu16vb:~/TEMP/IupSwift/BUILD/Linux$ make
[ 12%] Built target IupSwift_SyncIcons
[ 12%] Built target IupSwift_SyncLibraries
[ 62%] Built target IupSwift_SyncResources
[ 62%] Built target IupSwift_SyncPlugins
[100%] Built target IupSwift
pinky@Ubuntu16vb:~/TEMP/IupSwift/BUILD/Linux$ make clean
pinky@Ubuntu16vb:~/TEMP/IupSwift/BUILD/Linux$ make
[ 12%] Embedding icon /home/pinky/TEMP/IupSwift/icons/linux/icon.png
[ 12%] Built target IupSwift_SyncIcons
[ 12%] Built target IupSwift_SyncLibraries
[ 25%] Embedding resource /home/pinky/TEMP/IupSwift/resources/icon.bmp
[ 37%] Embedding resource /home/pinky/TEMP/IupSwift/resources/VeraMono.ttf
[ 50%] Embedding resource /home/pinky/Source/Blurrr/Release/BlurrrSDKLinux/boots
trap/SharedCMake/C/resources/BlurrrSDK3rdPartyLicenses.txt
[ 62%] Embedding resource /home/pinky/TEMP/IupSwift/resources/gamecontrollerdb.t
xt
[ 62%] Built target IupSwift_SyncResources
[ 62%] Built target IupSwift_SyncPlugins
[ 75%] Building Swift object CMakeFiles/IupSwift.dir/source/main.swift.o
[ 87%] Building Swift object CMakeFiles/IupSwift.dir/source/BlurrrMain.swift.o
[100%] Linking Swift executable IupSwift
[100%] Built target IupSwift
pinky@Ubuntu16vb:~/TEMP/IupSwift/BUILD/Linux$ make
```
- Blurr GenProj Dialog Box:** The title bar reads 'Blurr GenProj'. It contains the following settings:
 - Project Type: Swift
 - Platform: Linux
 - Build Type: Debug
 - Open Project
 - Swift path must be setup in Preferences
 - Buttons: Purge, Generate
 - Bottom buttons: Back, Restart Wizard

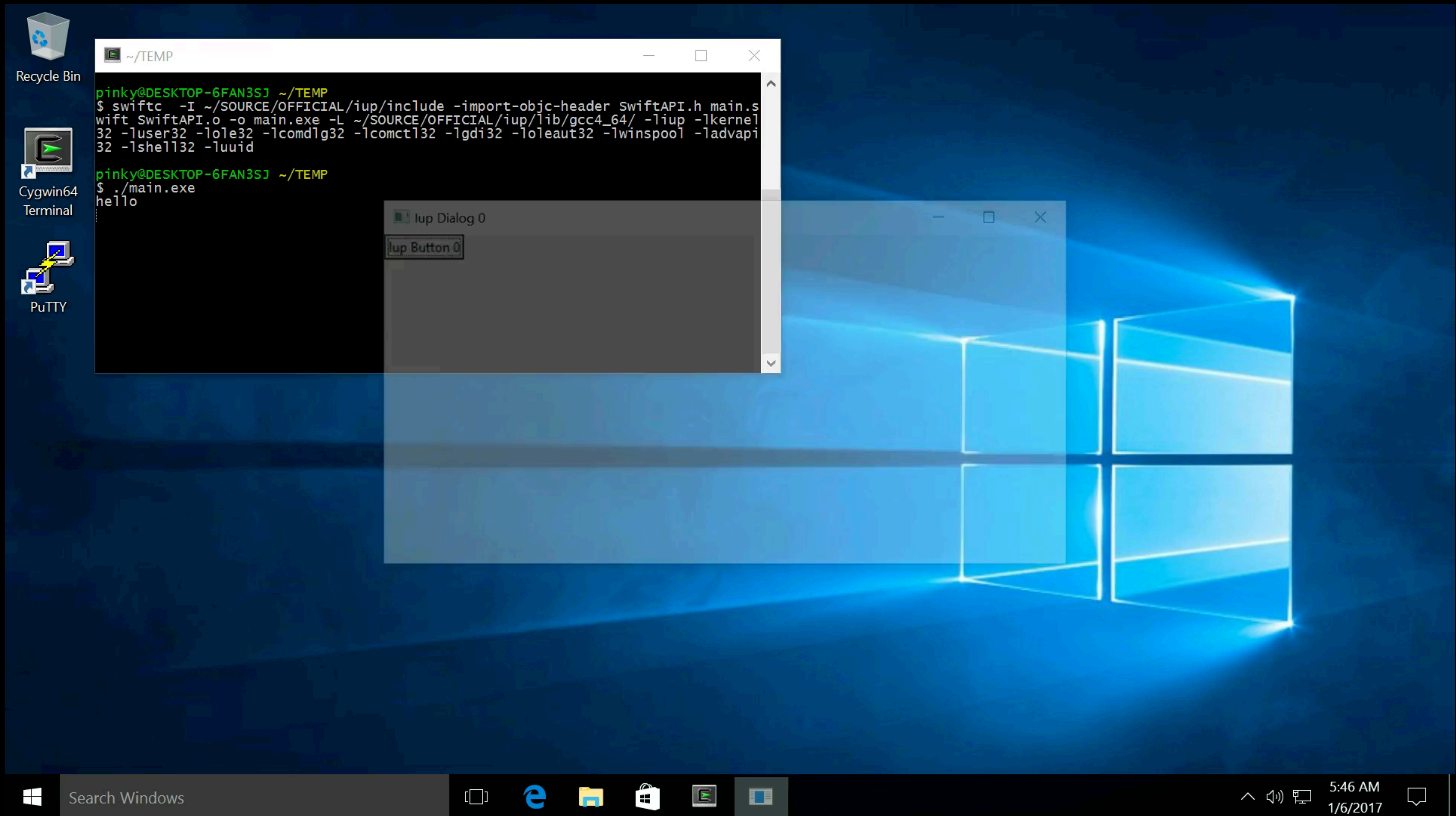
Raspberry Pi (Raspbian)

The screenshot displays the Raspbian desktop environment. At the top, the taskbar includes a menu icon, system icons (network, volume, battery), and the system tray showing the time as 06:14 and 0% battery. The desktop background is light blue and features a large Raspberry Pi logo in the center. On the left side, there are icons for 'Trash' and 'New'. In the bottom-left corner, a terminal window is open, showing the following output:

```
pi@raspberrypi: ~/TEMP/iupswift/BUILD/Raspbian
File Edit Tabs Help
CPackSourceConfig.cmake
pi@raspberrypi:~/TEMP/iupswift/BUILD/Raspbian $ make
[ 0%] Built target IupSwift_SyncPlugins
[ 12%] Built target IupSwift_SyncIcons
[ 62%] Built target IupSwift_SyncResources
[ 62%] Built target IupSwift_SyncLibraries
[100%] Built target IupSwift
pi@raspberrypi:~/TEMP/iupswift/BUILD/Raspbian $ ./IupSwift
```

In the bottom-right corner, a Swift IDE window titled 'Blurr GenProj' is open. It shows the Swift logo and the project directory path: `/home/pi/TEMP/iupswift/`. A 'Restart Wizard' button is visible at the bottom of the window.

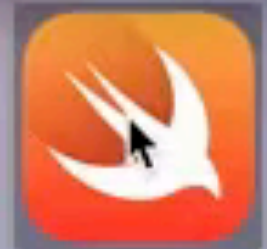
Windows



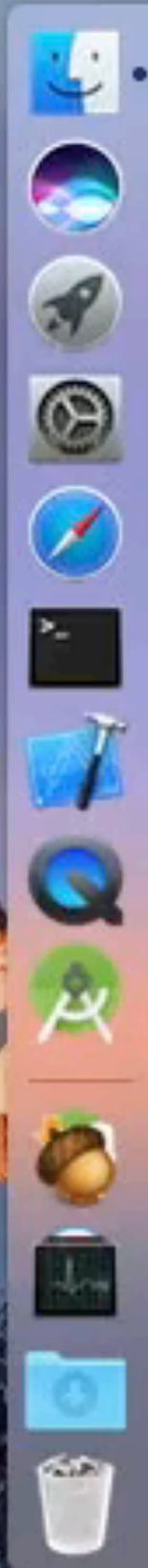
*Fun fact: Swift on Windows is still extremely experimental

Mac

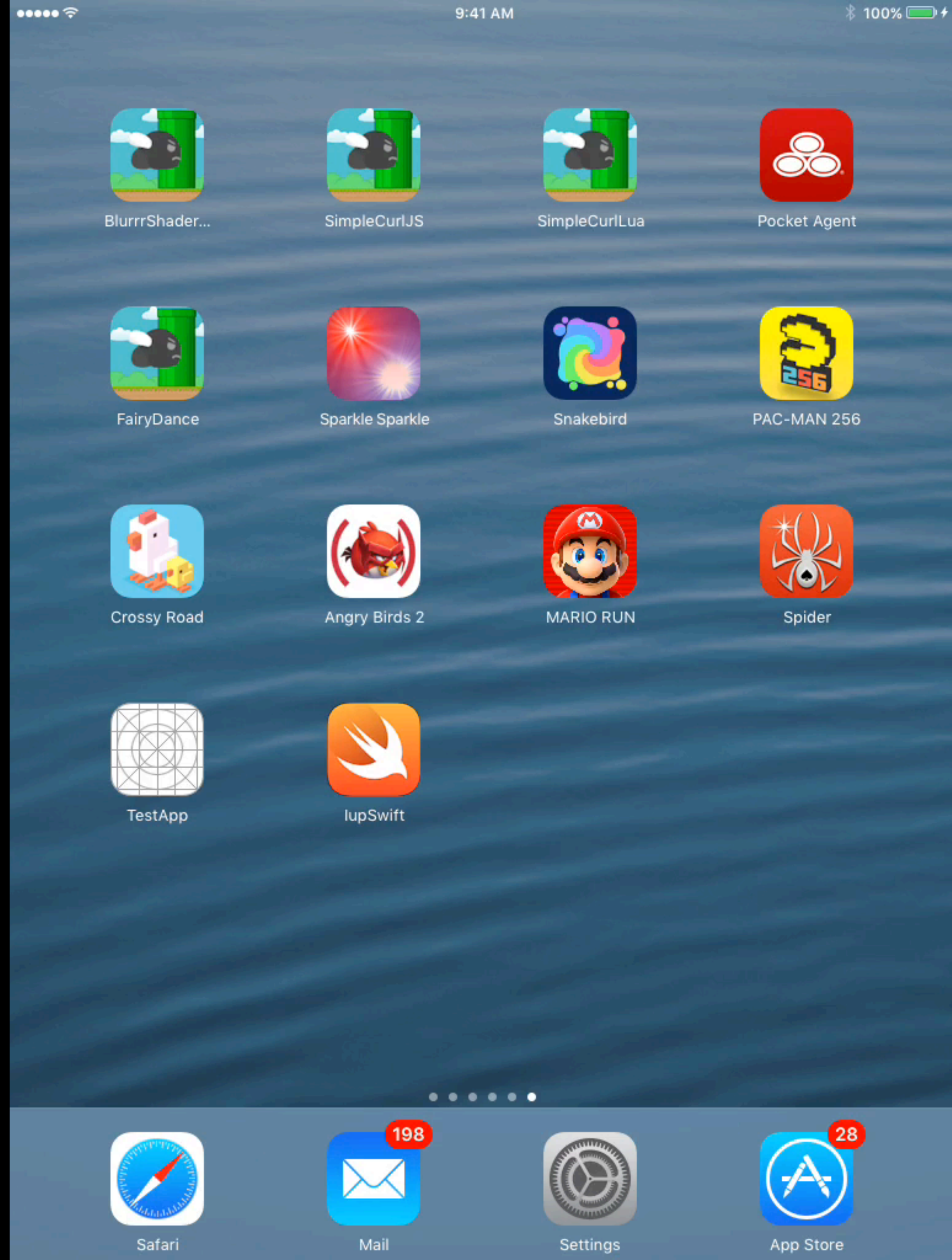
Finder File Edit View Go Window Help Fri 1:47 AM screencast



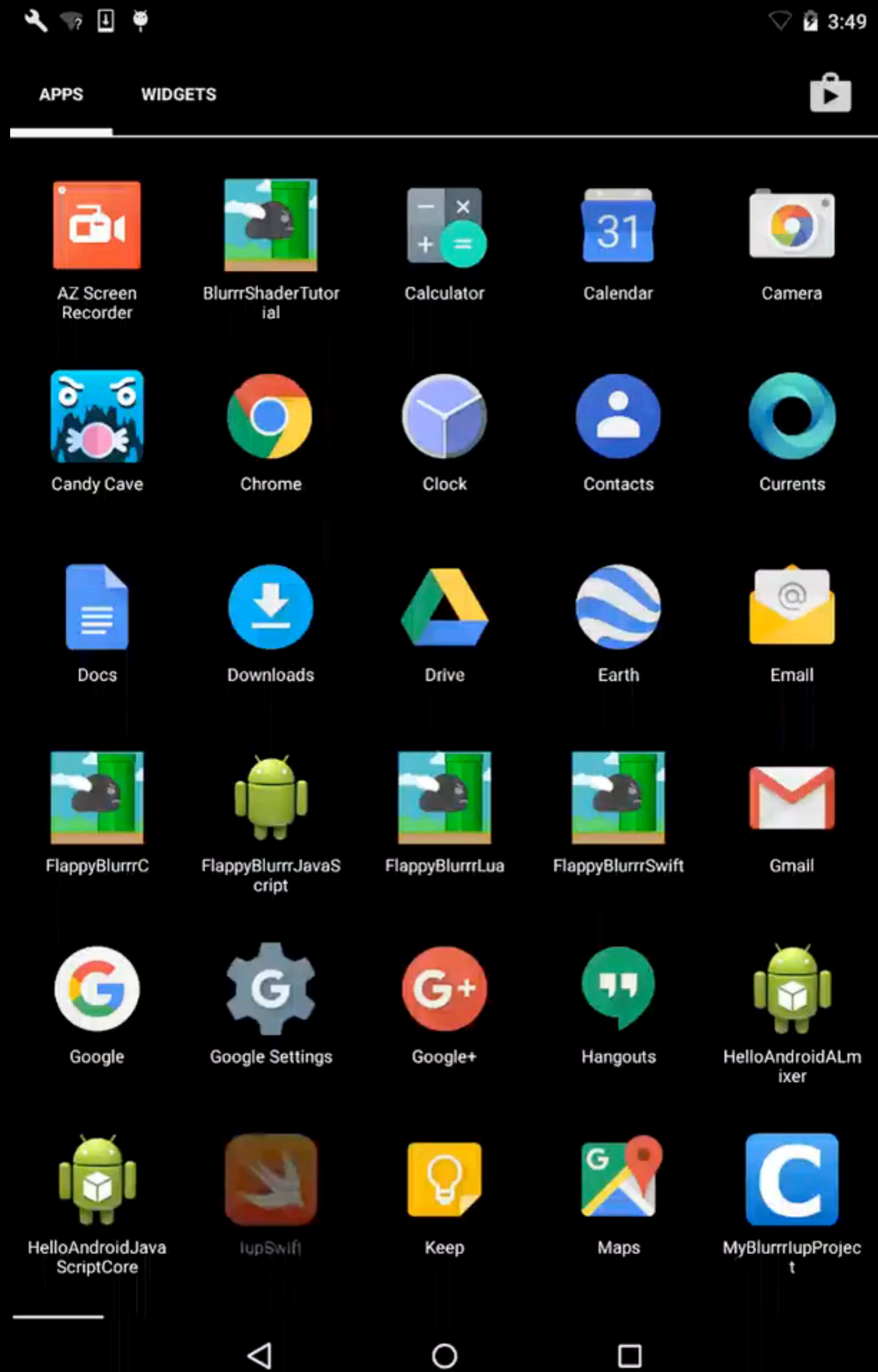
lupSwift



iOS



Android



lupCocoaTouch

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

iOS Event Loop

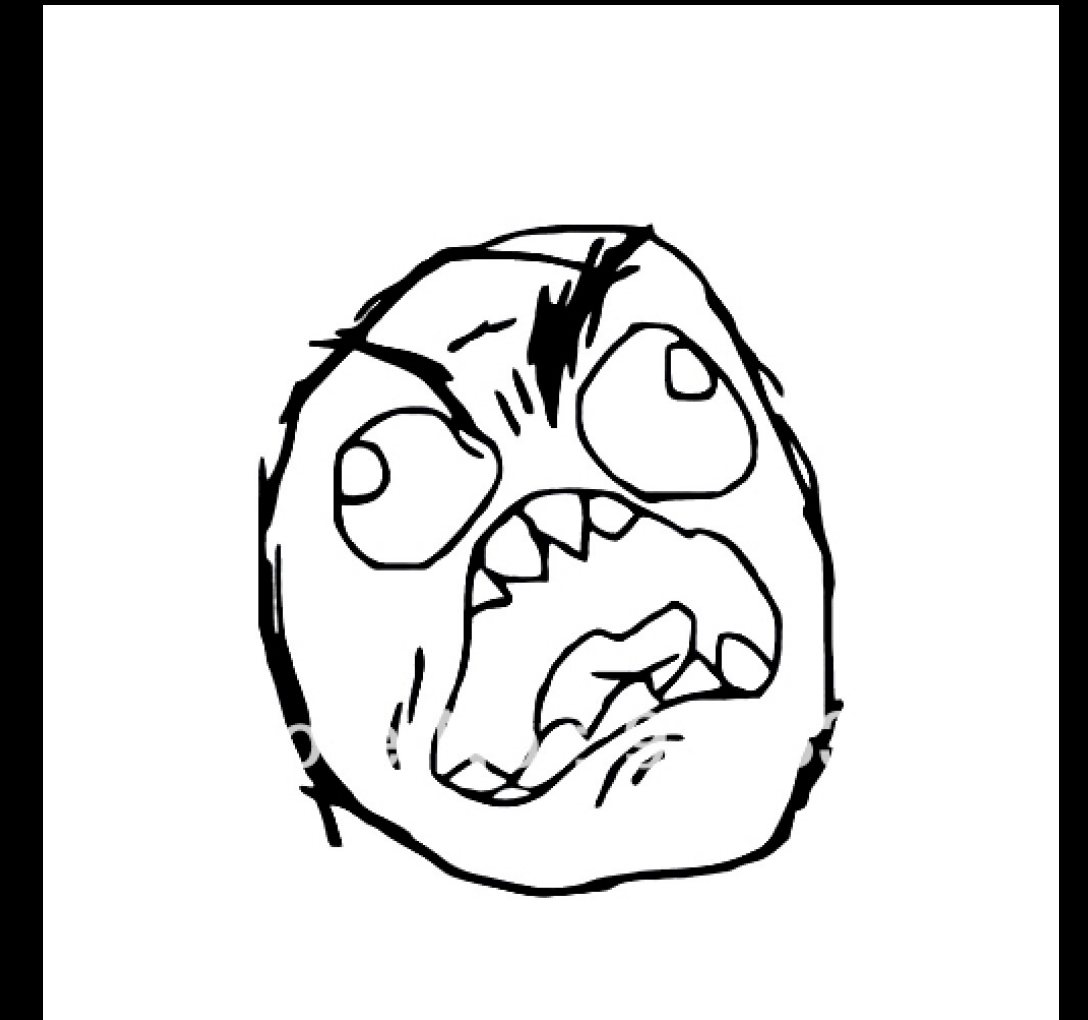
- Apple controls the event loop (same as Mac)
 - More rigid than Mac

lupAndroid

- *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

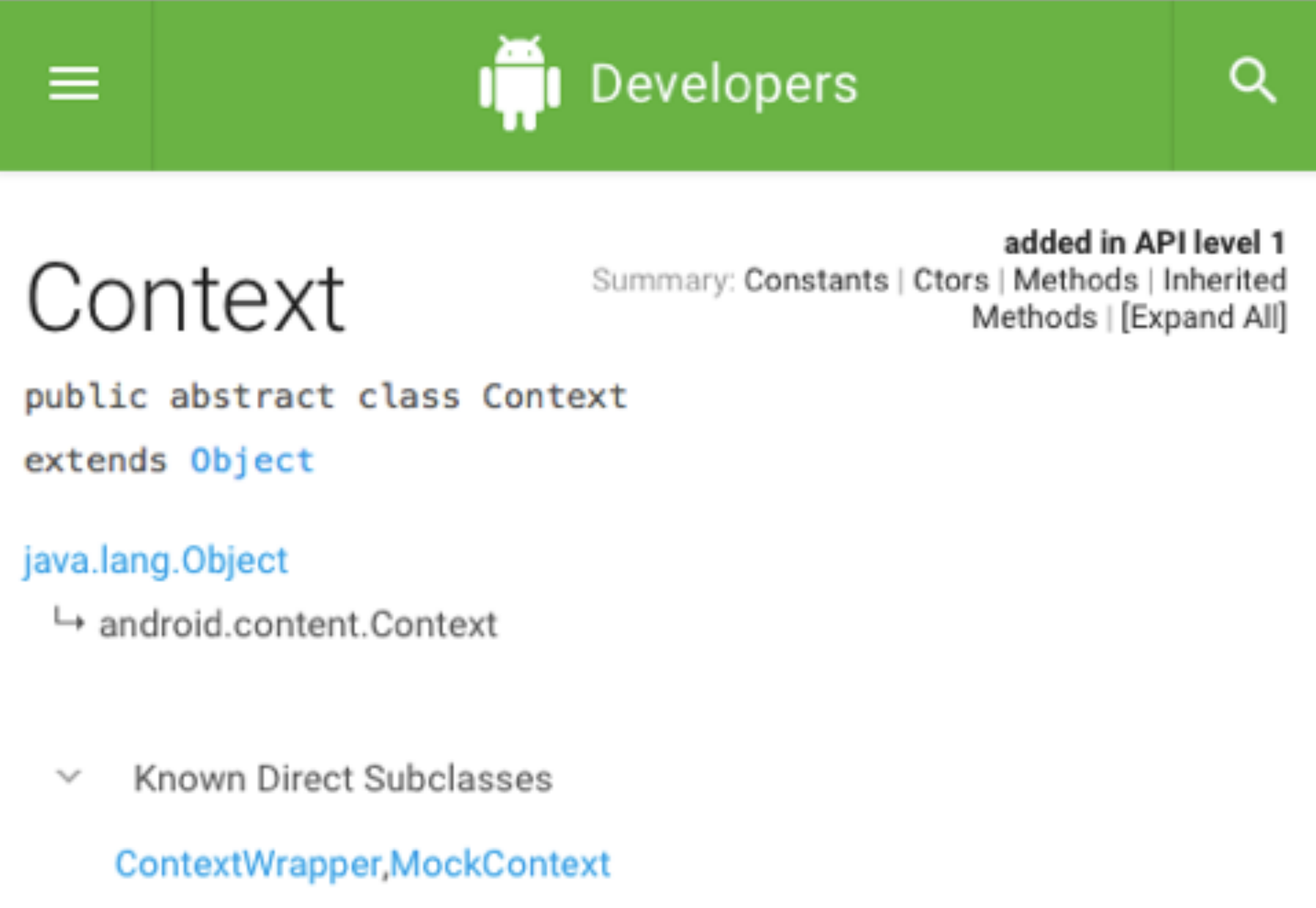
The Android NDK “Really Does Suck”

- 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



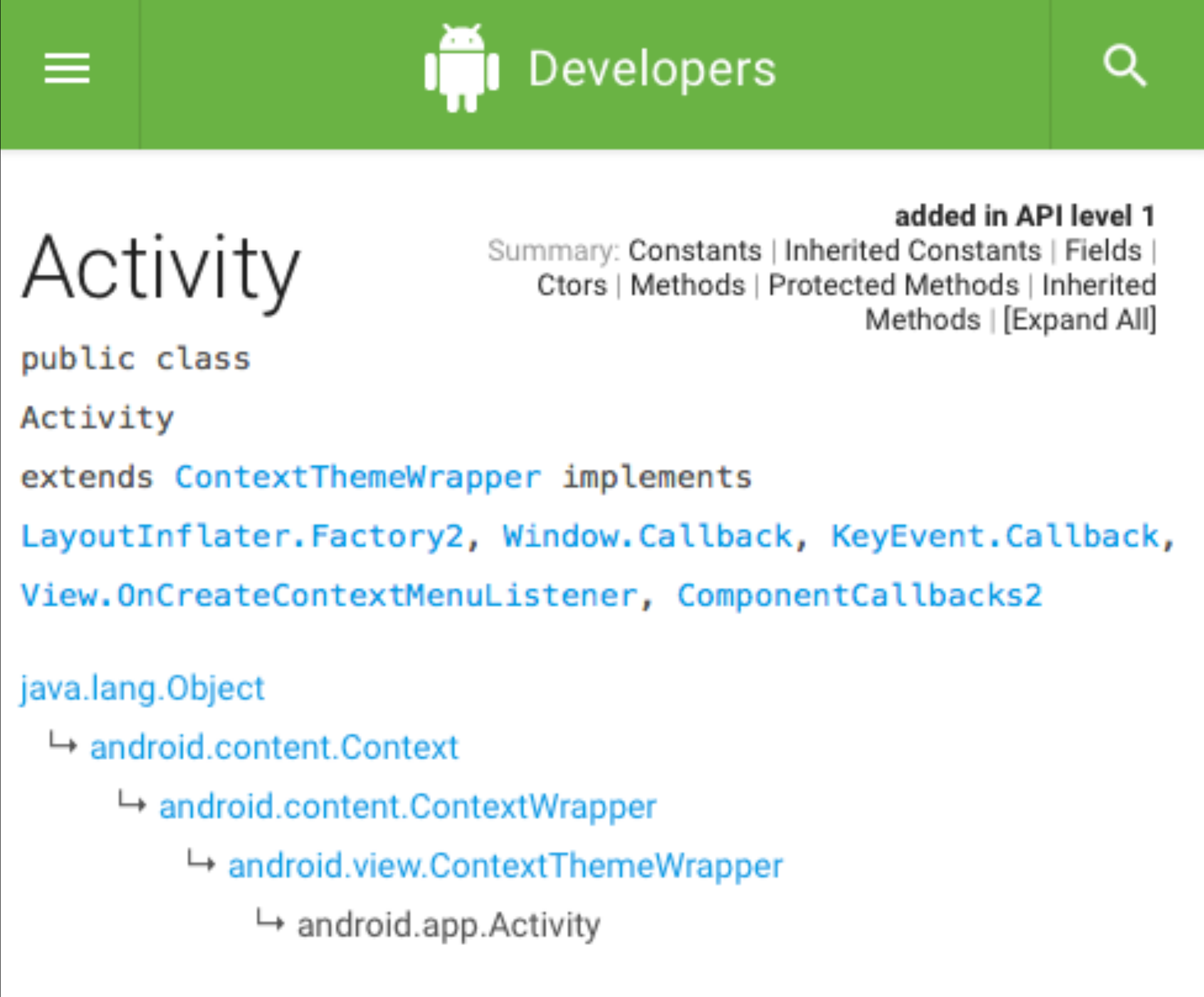
The screenshot shows the Android Developers website interface. At the top, there is a green header with a hamburger menu icon on the left, the Android logo and the word "Developers" in the center, and a search icon on the right. Below the header, the main content area displays the documentation for the `Context` class. The title "Context" is prominently displayed. To its right, it says "added in API level 1" and provides navigation links: "Summary: Constants | Ctors | Methods | Inherited Methods | [Expand All]". Below the title, the code snippet shows `public abstract class Context` and `extends Object`. A link to `java.lang.Object` is provided, with a sub-link for `android.content.Context`. At the bottom, there is a section for "Known Direct Subclasses" which lists `ContextWrapper` and `MockContext`.

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



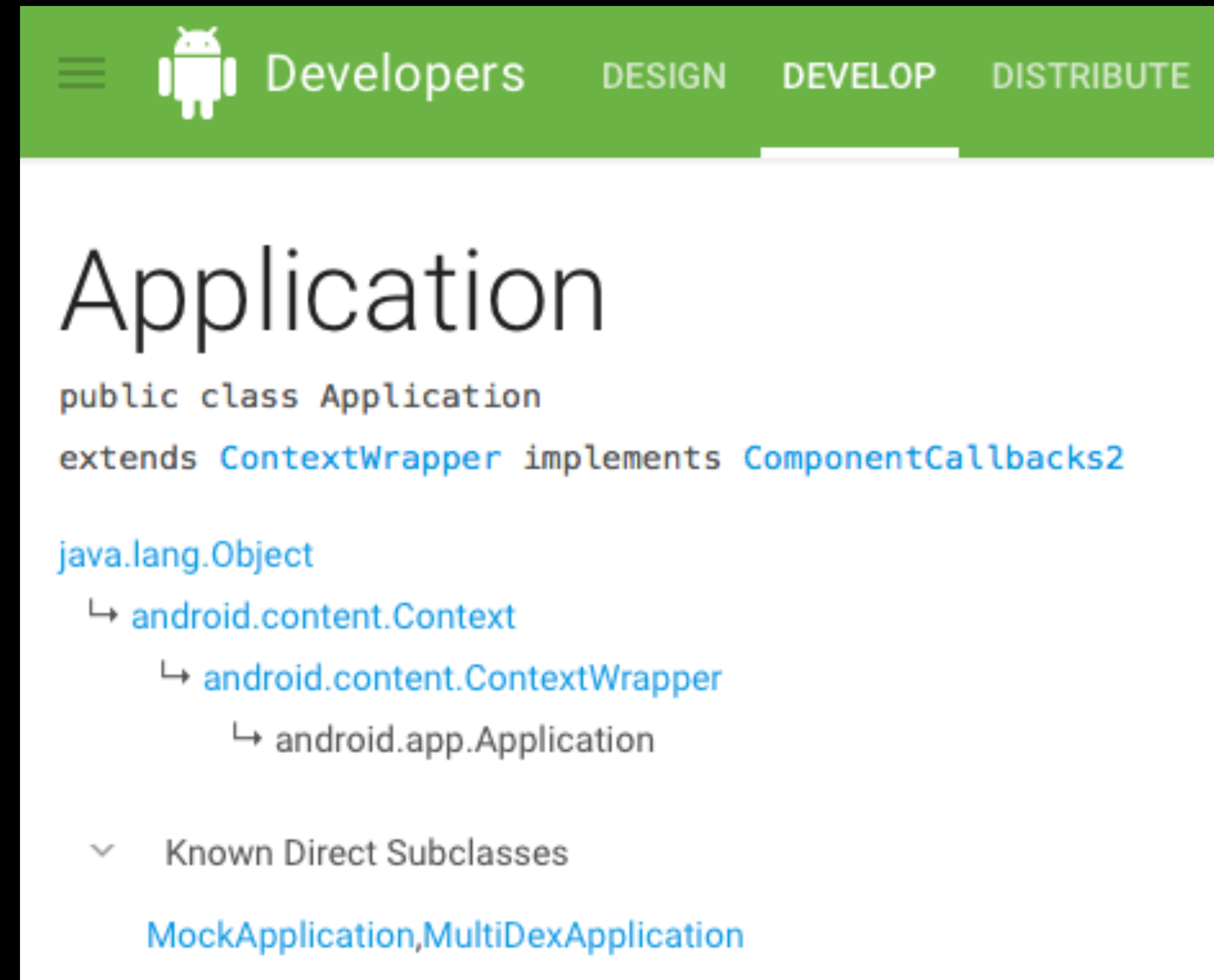
The screenshot shows the Android Studio interface with the 'Activity' class documentation. The top bar is green with a hamburger menu icon, an Android robot icon, and the text 'Developers'. A search icon is in the top right. The main content area has a white background. The title 'Activity' is in a large font. To the right of the title, it says 'added in API level 1'. Below the title, there are links for 'Summary: Constants | Inherited Constants | Fields | Ctors | Methods | Protected Methods | Inherited Methods | [Expand All]'. The class signature is 'public class Activity'. Below that, it says 'extends ContextThemeWrapper implements LayoutInflater.Factory2, Window.Callback, KeyEvent.Callback, View.OnCreateContextMenuListener, ComponentCallbacks2'. Underneath, it shows the inheritance hierarchy: 'java.lang.Object' with a right-pointing arrow, '↳ android.content.Context' with a right-pointing arrow, '↳ android.content.ContextWrapper' with a right-pointing arrow, '↳ android.view.ContextThemeWrapper' with a right-pointing arrow, and '↳ android.app.Activity' with a right-pointing arrow.

```
public class Activity
 extends ContextThemeWrapper implements
 LayoutInflater.Factory2, Window.Callback, KeyEvent.Callback,
 View.OnCreateContextMenuListener, ComponentCallbacks2

java.lang.Object
↳ android.content.Context
 ↳ android.content.ContextWrapper
 ↳ android.view.ContextThemeWrapper
 ↳ android.app.Activity
```

# 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`



The screenshot shows the Android Studio interface with a green header bar containing a hamburger menu, the Android logo, and the text 'Developers'. To the right of the header are the tabs 'DESIGN', 'DEVELOP', and 'DISTRIBUTE'. The main content area displays the 'Application' class documentation. The title 'Application' is in a large font. Below it, the class signature is shown: `public class Application` extends `ContextWrapper` implements `ComponentCallbacks2`. The superclass `java.lang.Object` is listed below. A tree view shows the inheritance path: `↳ android.content.Context`, `↳ android.content.ContextWrapper`, and `↳ android.app.Application`. A section titled 'Known Direct Subclasses' is expanded, showing `MockApplication, MultiDexApplication`.

# 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) UIThread

- All GUI APIs must be called from the UIThread
- 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

# lupEmscripten (Web Browser)

## Chris Matzenbach

- One other major platform to discuss
- Native vs. Web fight is not over

# lupEmscripten

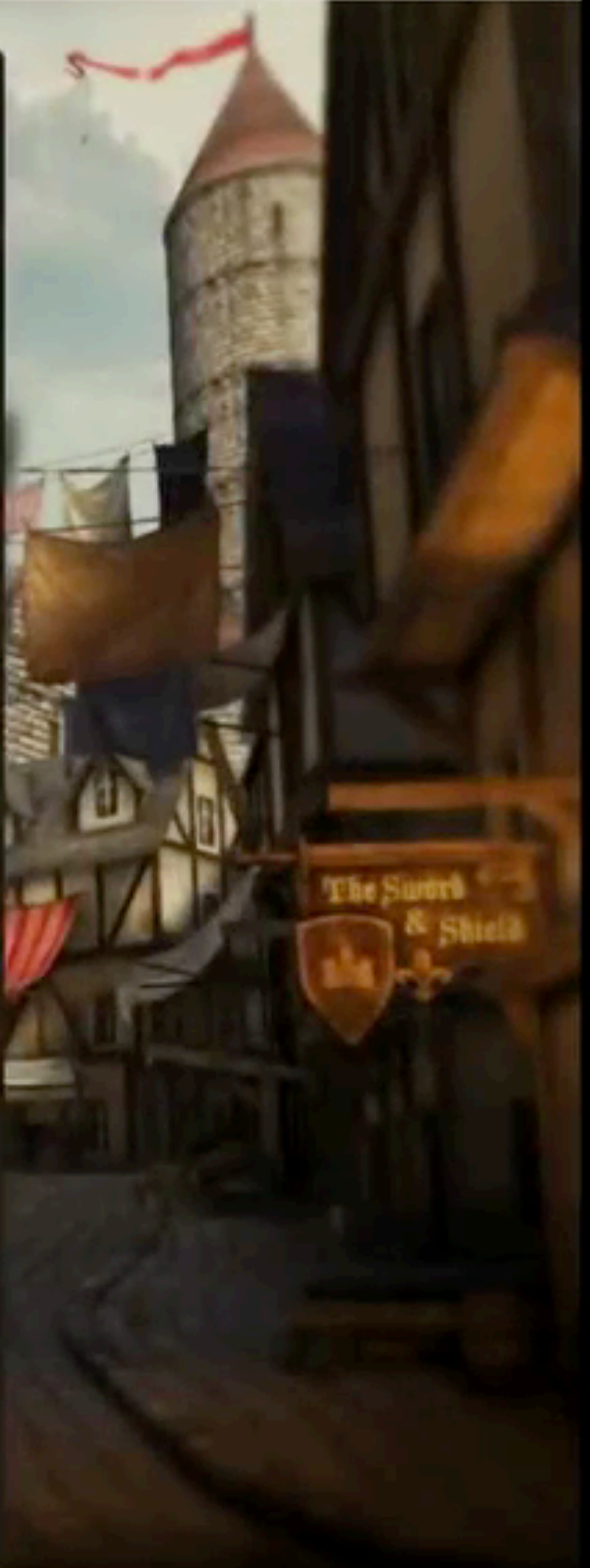
Chris Matzenbach

# 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**

# JavaScript - the problem child

- 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
- Solution? The Emscripten compiler, released in 2014



# 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
- lwpEmscripten - 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



# External Function Example

```
3 extern int emjsTest_GetInt(int widget_id);
4
5 // here is our C function, which will make use of our external JavaScript function
6 int emscriptenExample(int widgetID) {
7
8 // here we call our external function, as if it were any other C function
9 int val = emjsTest_GetInt(widgetID);
10
11 return val;
12 }
```

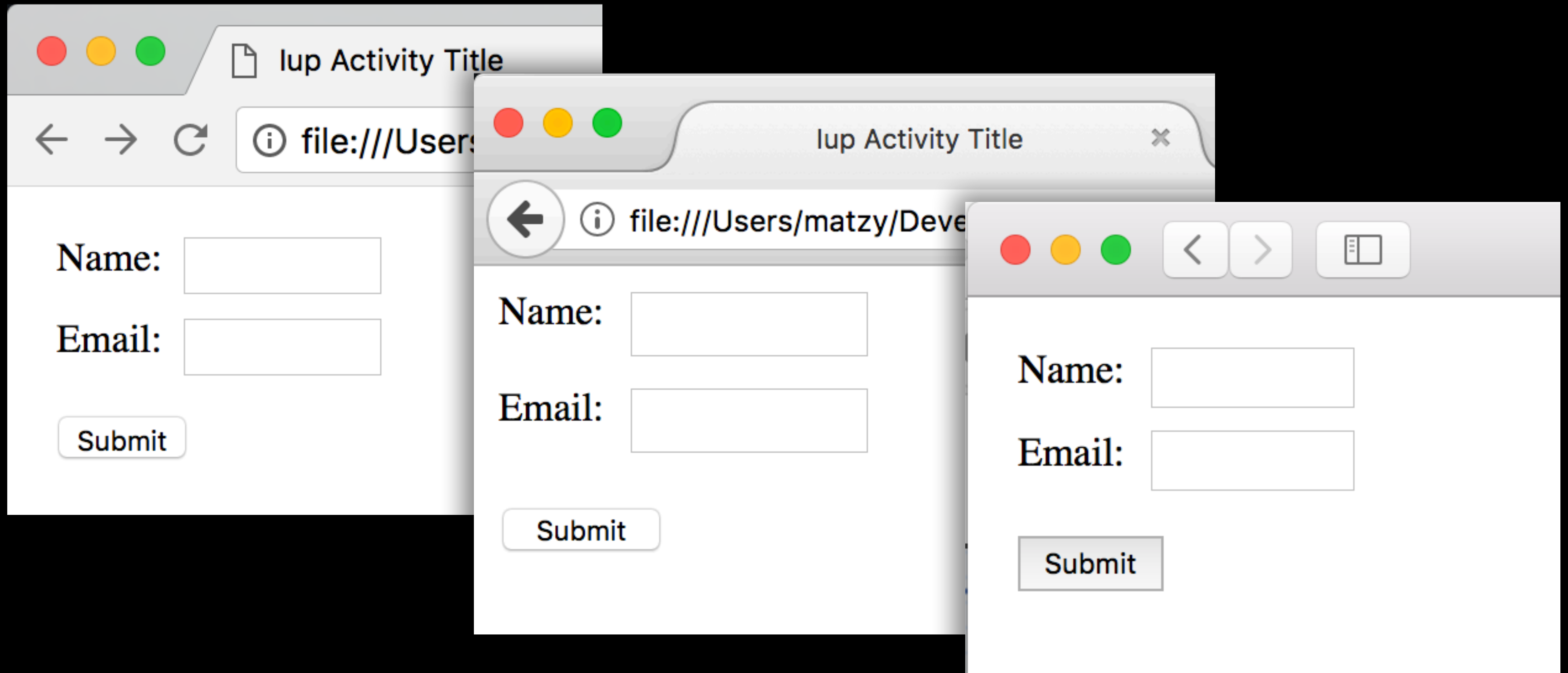
```
1 // library which holds our external JavaScript functions
2 var Library = {
3
4 emjsTest_GetInt: function(widget_id) {
5 // pretending this returns an int (really returns a string)
6 return document.getElementById(widget_id).innerHTML;
7 }
8
9 };
```

# Widget Creation

- User defines the widget they want in lup, 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

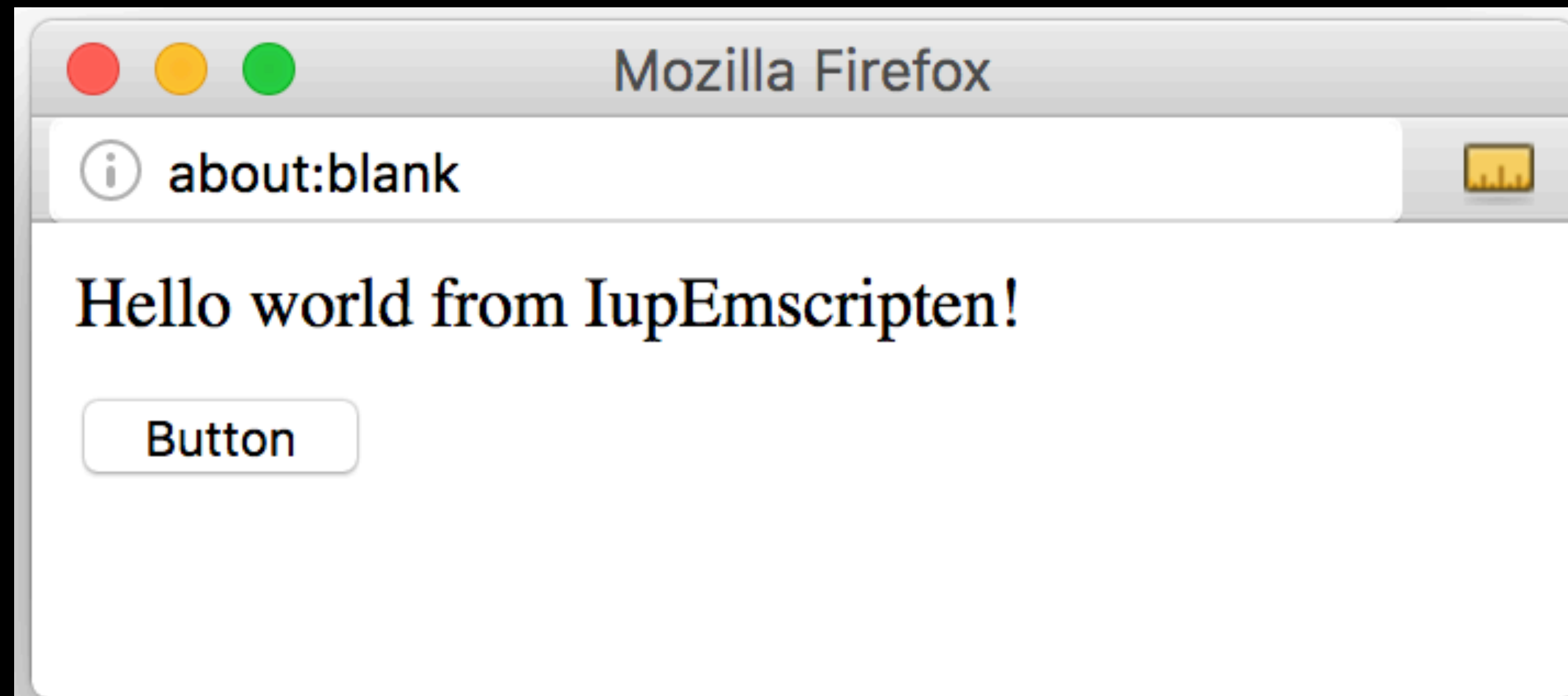
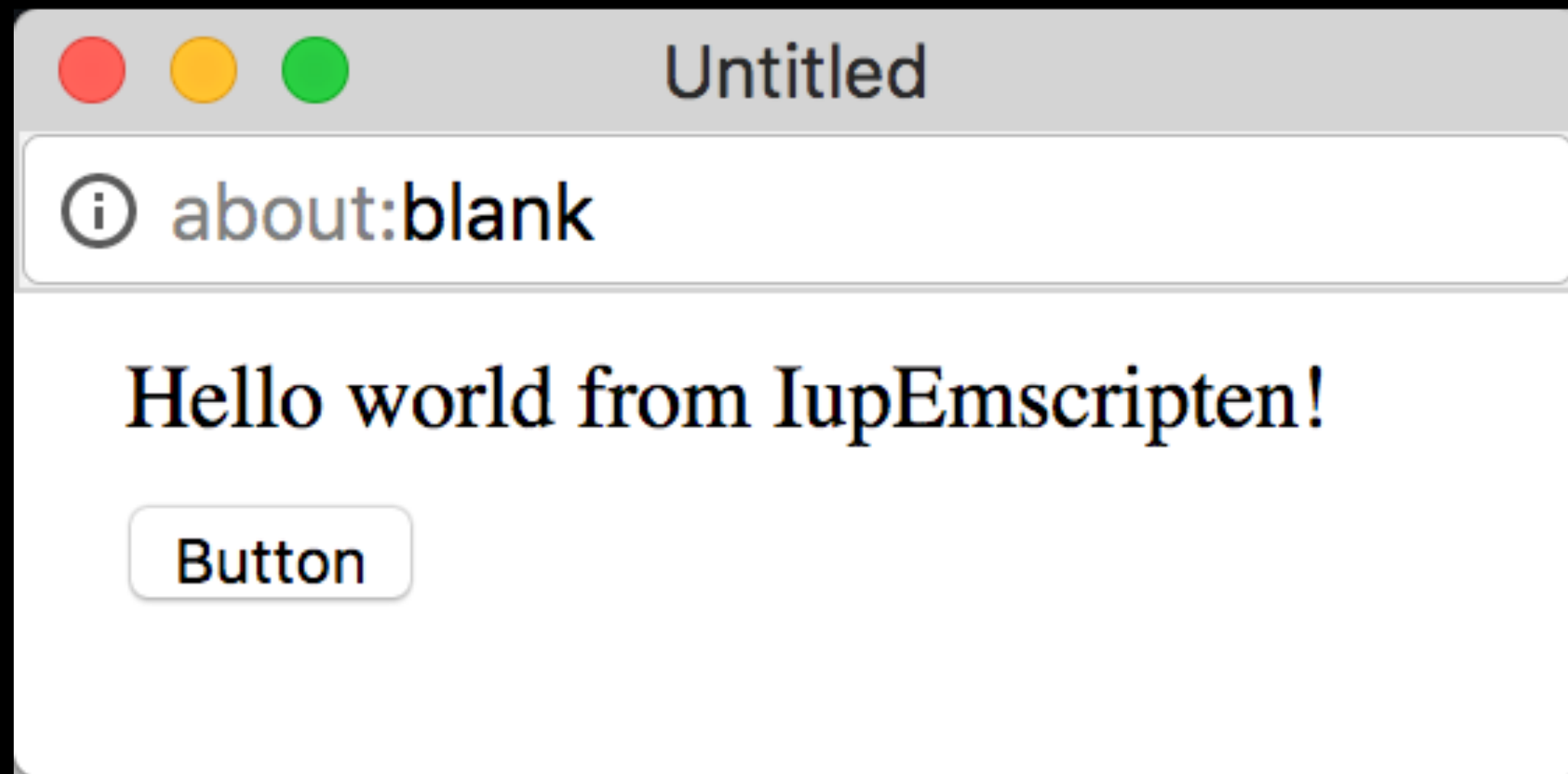
# Label, Input Text and Button

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



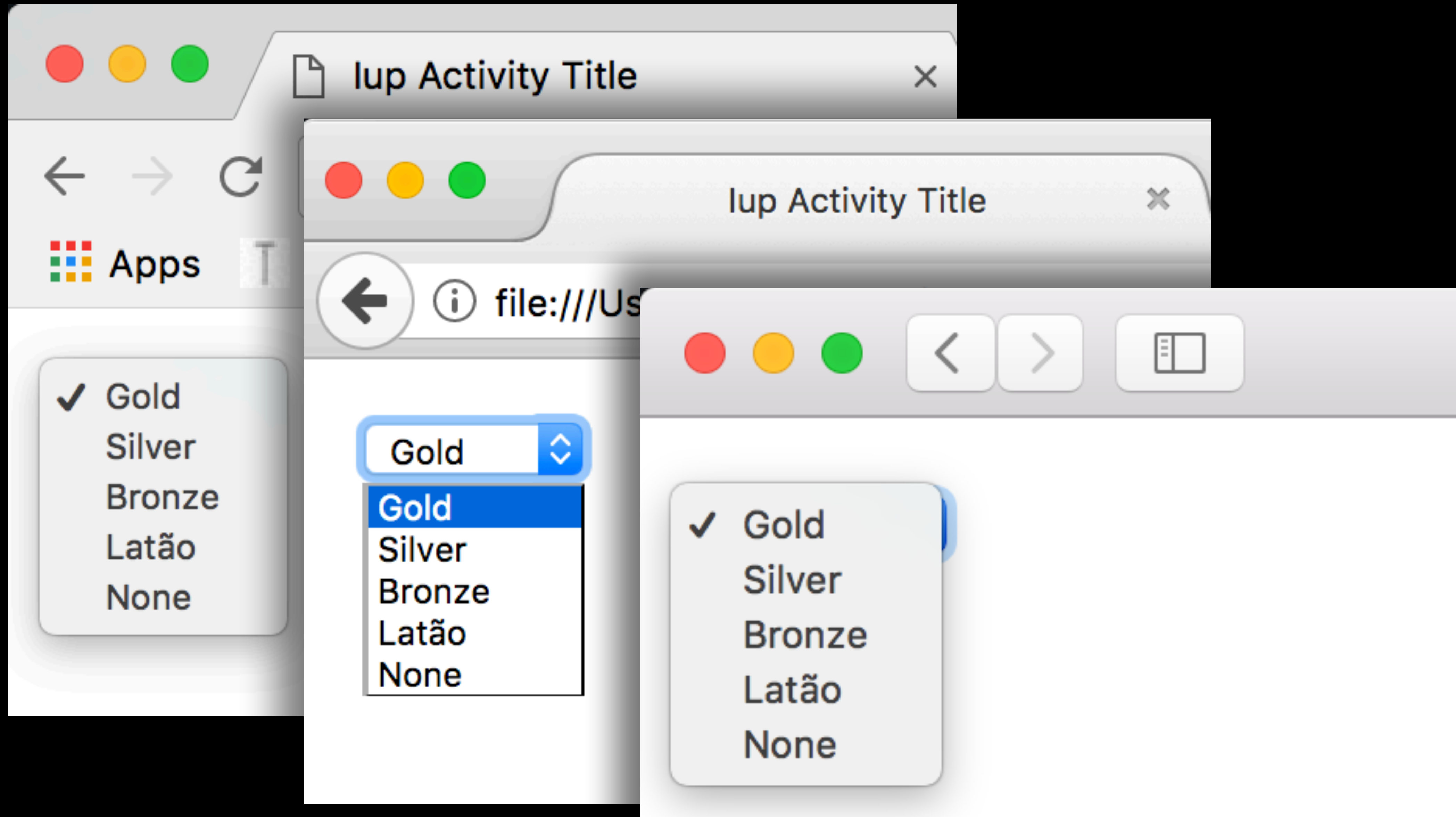
# Dialog, Label and Button

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



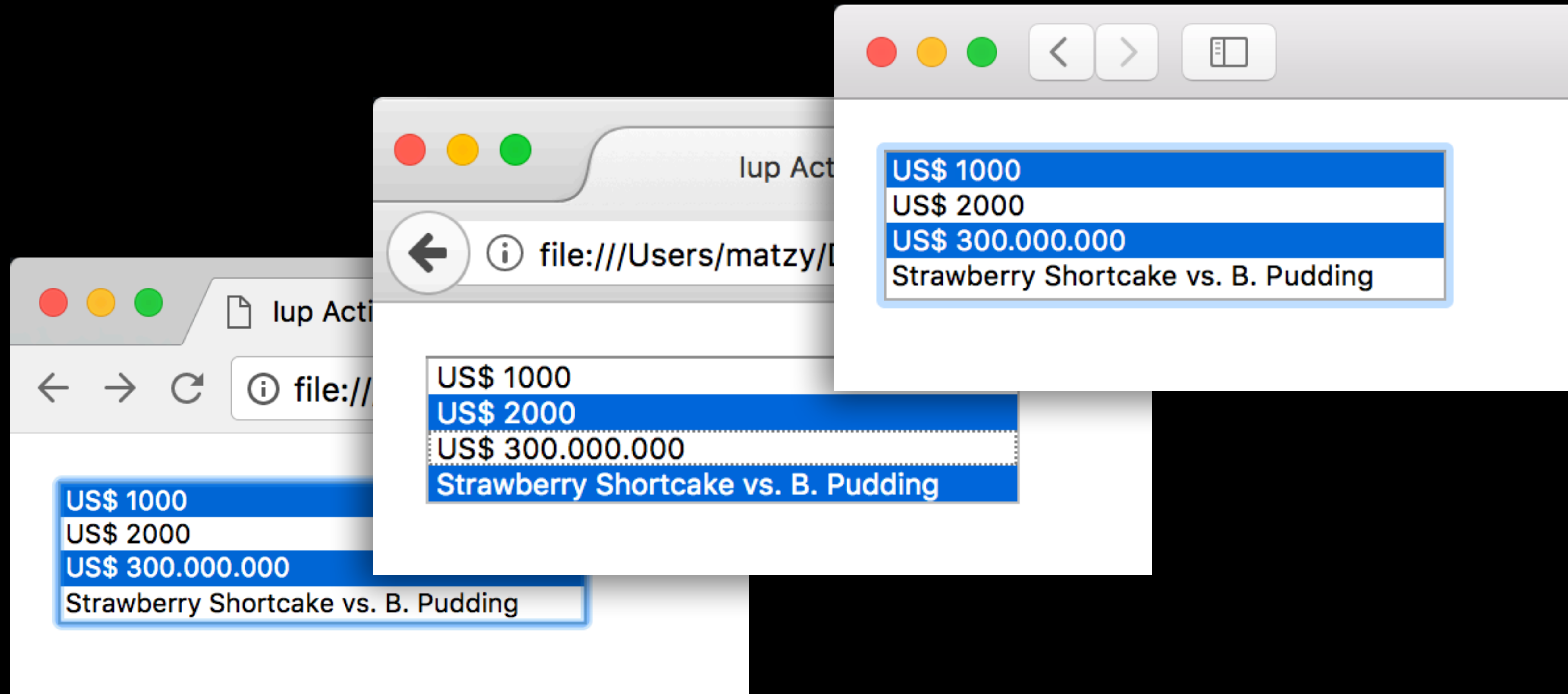
# List - Dropdown

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



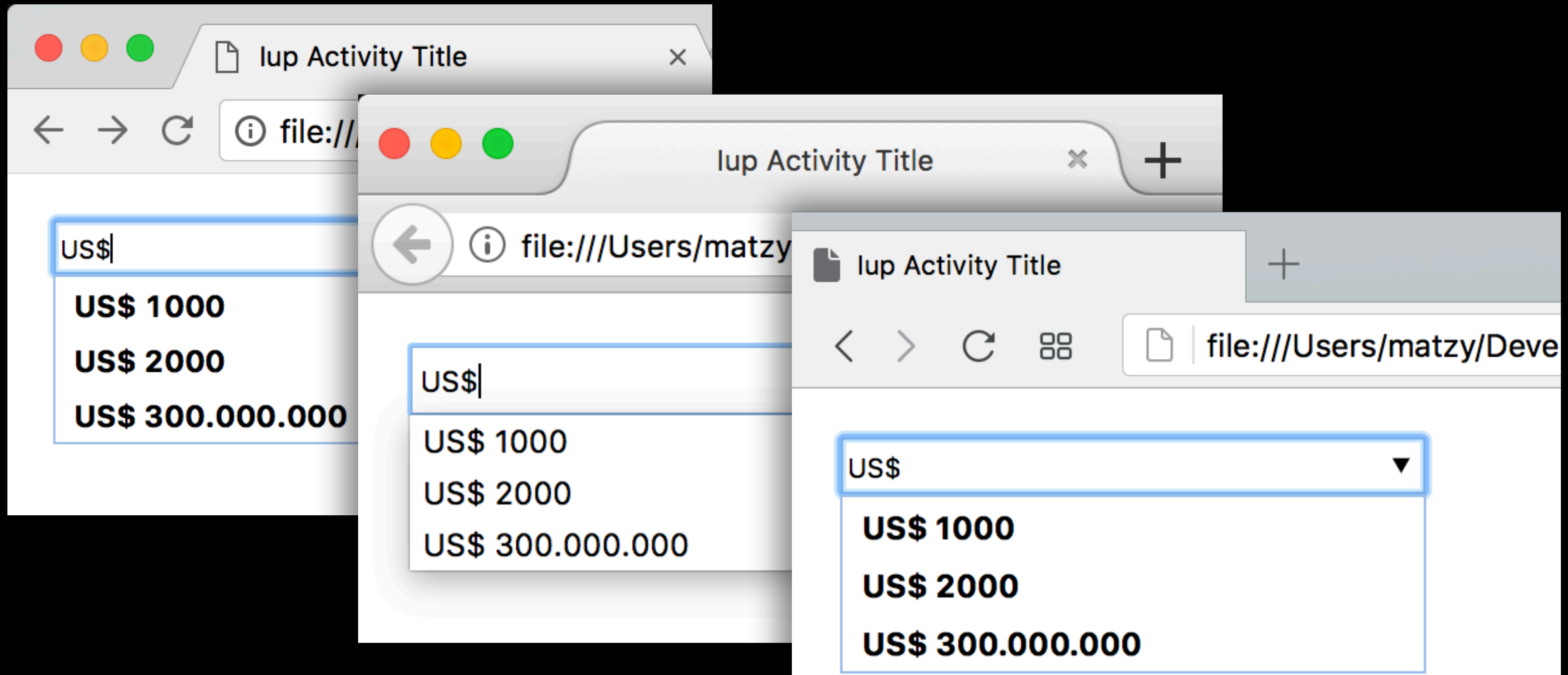
# List - Multiple

- 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

# Event Loop

- 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)

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

# IUP Init (Old vs. New)

```
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();
 return 0;
}
```

# IUP Init (Old vs. New)

```
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();
 return 0;
}
```

Acts as explicit Opt-in for new behavior



# IUP Init (Old vs. New)

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

```
void IupExitPoint()
{
 TupClose();
}

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;
}
```

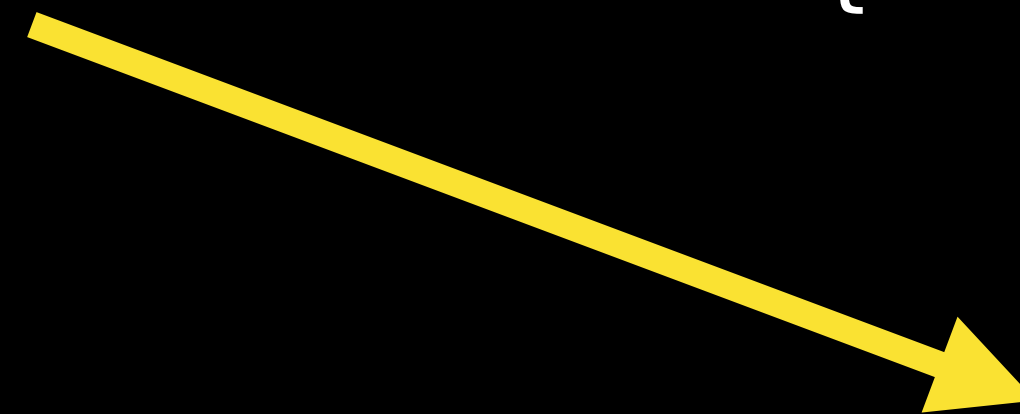
# IUP Init on Cocoa/CocoaTouch

```
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

Designated EntryPoint  
for Android

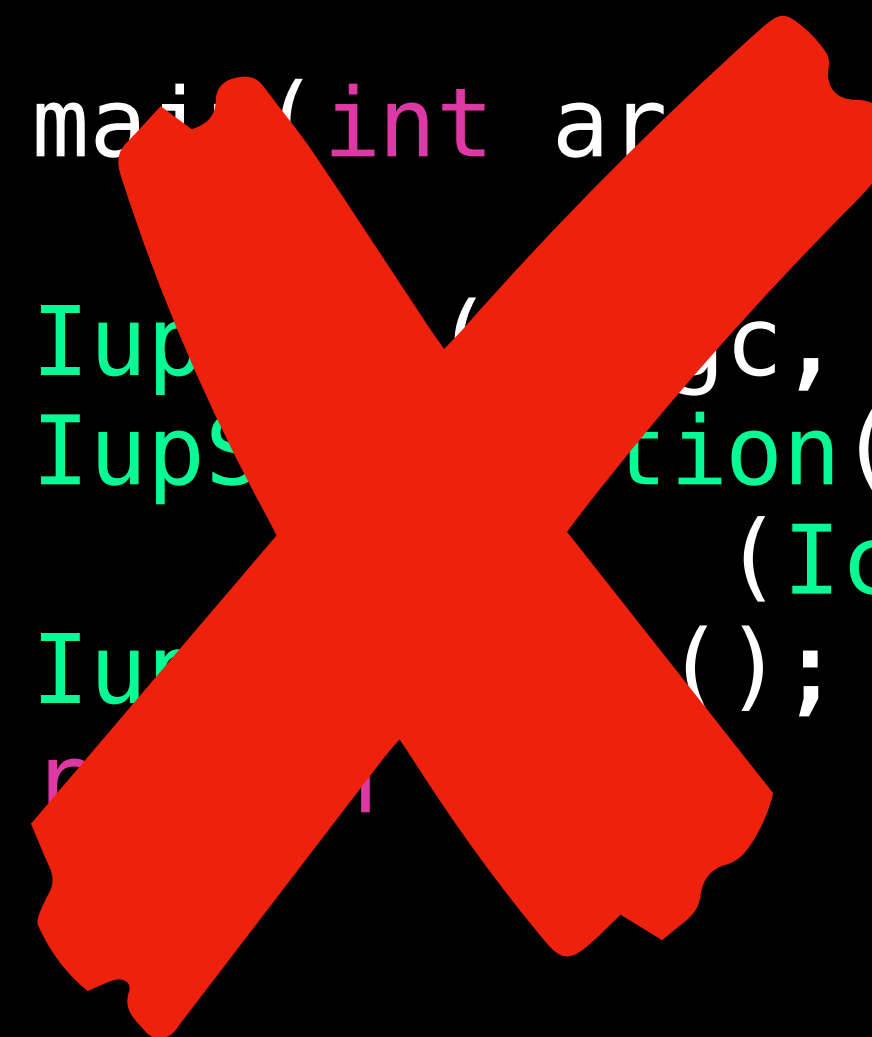


```
void IupExitPoint()
{
 IupClose();
}

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

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

Never gets called



# IUP Init on Emscripten

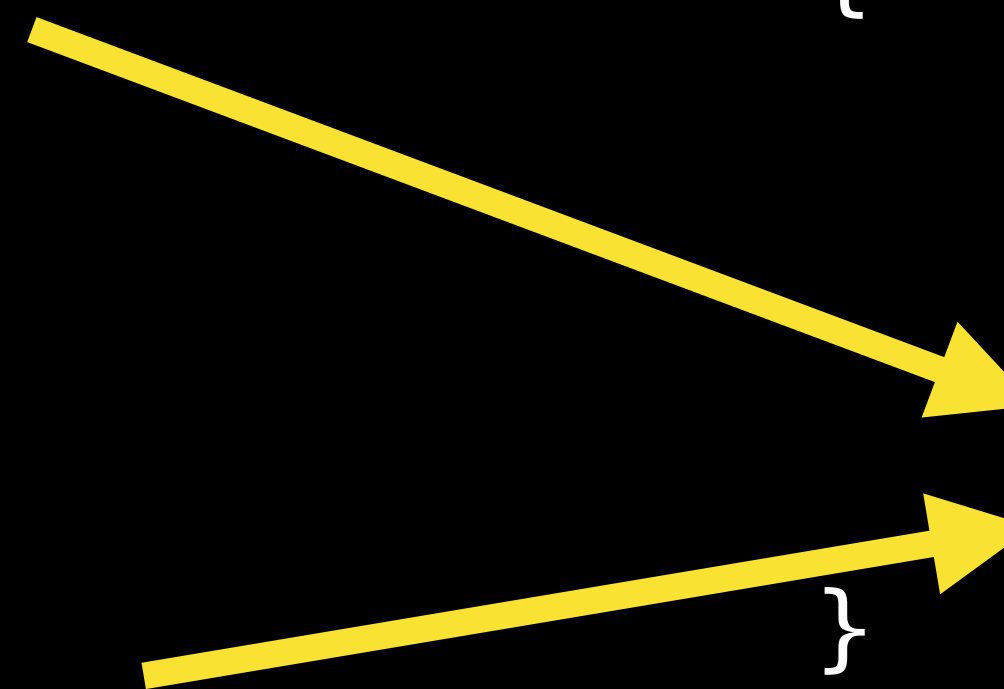
```
void IupExitPoint()
{
 IupClose();
}

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

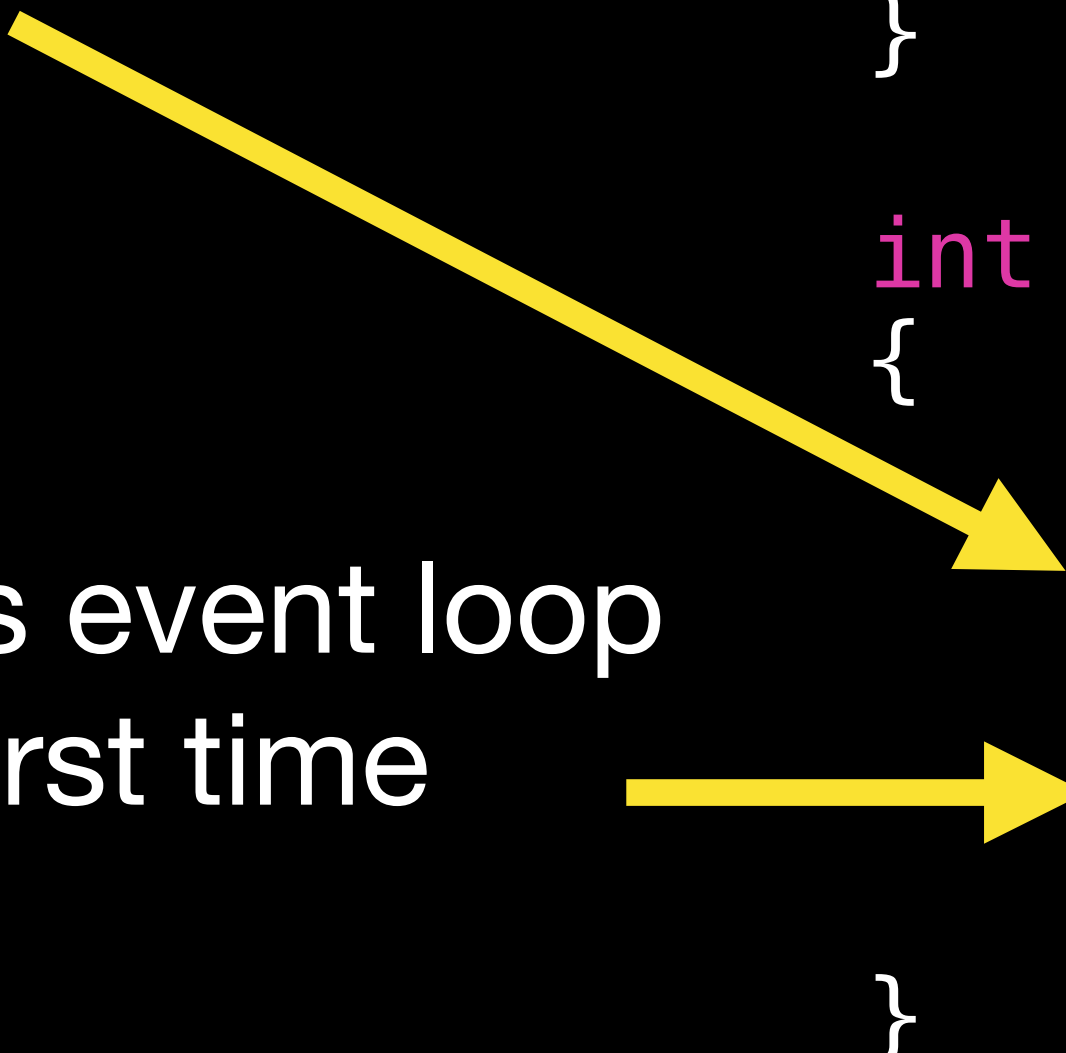
Used to detect opt-in to activate new behavior

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

```
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;
}
```



# 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

```
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;
}
```

# 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

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

 ExemptWord <- 'bork' / 'Bork'

 EndOfParagraphPunctuation <- [.!?]%n1
 -> 'Bork Bork Bork!'

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

# lupBork: Windows

## Uses ISpVoice via C++

The screenshot shows the Microsoft Visual Studio IDE with the following components:

- Solution Explorer:** Shows the project structure for 'lupBork', including source files like 'main.lua.c' and 'SpeechSynthWindows.cpp'.
- Code Editor:** Displays the C++ code for 'SpeechSynthWindows.cpp'. The code includes a thread function 'RunSpawnedThread' that initializes 'ISpVoice' and calls 'Speak' to output a message.
- Diagnostic Tools:** Shows a diagnostics session of 7 seconds with graphs for Process Memory (MB) and CPU usage.
- Output Window:** Displays the text 'Welcome to the wonderful world of the Swedish Chef!'.
- Error List:** Shows 0 Errors, 0 Warnings, and 0 Messages.

```
34
35
36 static int RunSpawnedThread(void* user_data)
37 {
38 HRESULT hr;
39 struct MyThreadUserData* my_thread_user_data = (struct MyThreadUserData*)user_data;
40 ISpVoice* speech_synthesizer = NULL;
41
42 ::CoInitialize(NULL);
43 hr = CoCreateInstance(CLSID_SpVoice, NULL, CLSCTX_ALL, IID_ISpVoice, (void **)&speech_synthesizer);
44 if (!SUCCEEDED(hr))
45 {
46 OutputDebugStringA("CoCreateInstance failed\n");
47 }
48
49 // Microsoft blocks on this API.
50 // HRESULT hr = s_speechSynthesizer->Speak(my_thread_user_data->input_string);
51 hr = speech_synthesizer->Speak(my_thread_user_data->input_string);
52
53 speech_synthesizer->Release();
54 speech_synthesizer = NULL;
55 ::CoUninitialize();
56
57 BlurrCore_Free(my_thread_user_data->input_string);
58 free(my_thread_user_data);
59
60 // TODO: How to signal main thread playt
61
62 if(NULL != s_speechDidFinishCallback)
63 {
64 s_speechDidFinishCallback();
65 }
66
67 return 0;
68 }
69
70 void Speech_SayString(const char* input_string)
71 {
72 struct MyThreadUserData* my_thread_user_data = (struct MyThreadUserData*)calloc(1, sizeof(struct MyThreadUserData));
73 my_thread_user_data->input_string = input_string;
74 }
```

# IupBork: Linux

## Fork/exec to external process 'spd-say'

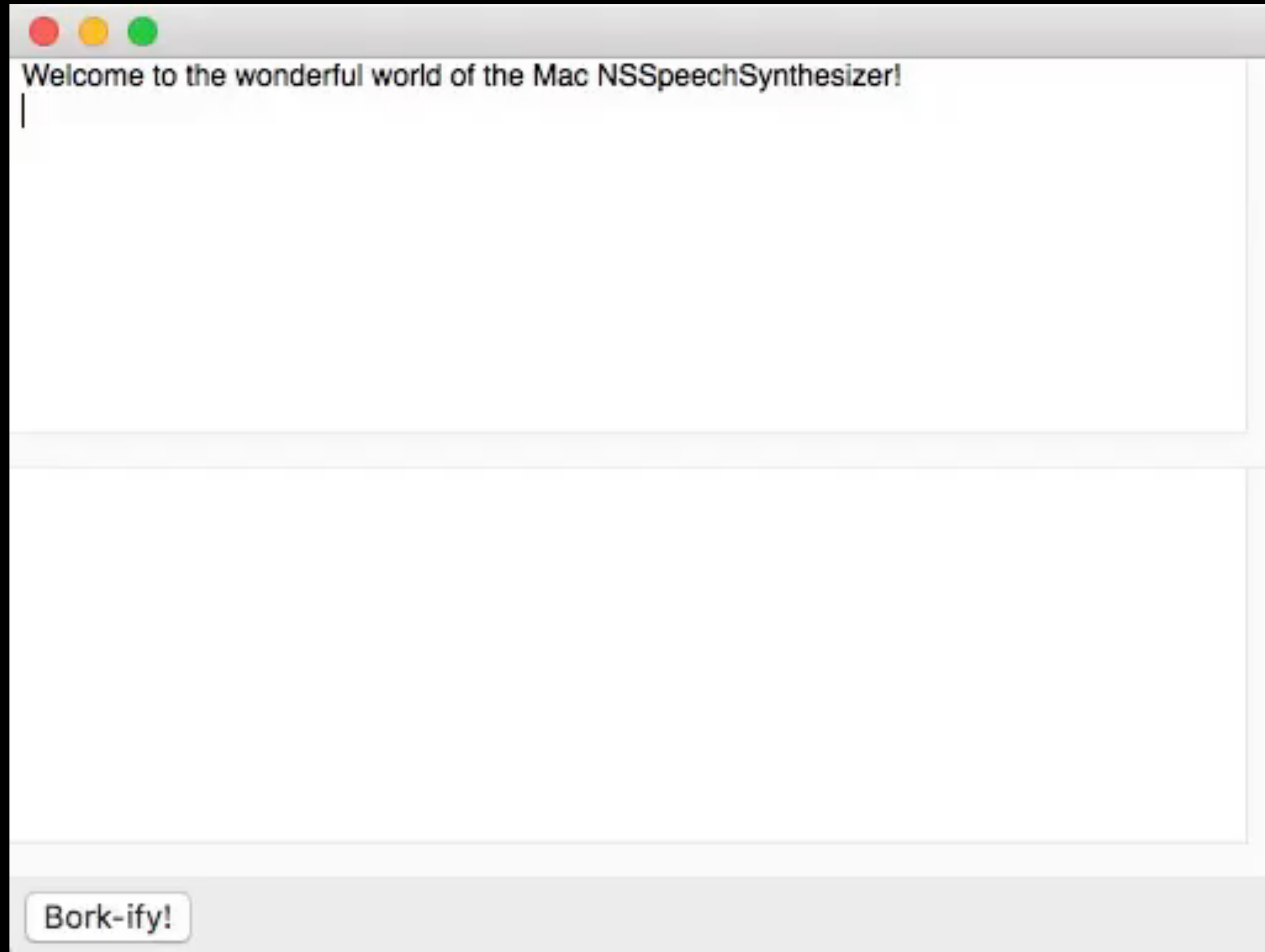
The image shows a Linux desktop environment with a terminal window and a Gvim editor window.

**Terminal Window:** The terminal shows the user 'pinky' at 'Ubuntu16vb' performing a commit and push to a Bitbucket repository. The commit message is 'Velcume-a tu zee vunderffool vurld ooff zee Sveddeesh Cheff! Included in thees ercheefe-a ere-a fuoor feele...'. The user then runs './IupBork', which displays a window titled 'Welcome to the wonderful world of the Swedish Chef!' with a 'Bork-ify!' button. The terminal also shows the compilation of 'cheff.x' and the use of 'cheff' to redirect text into 'README.bork'.

**Gvim Editor:** The editor shows the source code for 'SpeechSynthExternalProcess.c'. The code defines a thread function 'RunSpawnedThread' that forks a process to run 'spd-say' with the argument '--wait'. The code uses 'SDL\_Log' for logging and 'SDL\_Delay' to be nice to the CPU. A 'TODO' comment asks for a way to signal the main thread that playback is done. The code also shows the creation of a 'submit\_button' with the text 'Bork-ify!' and the 'ACTION' attribute.

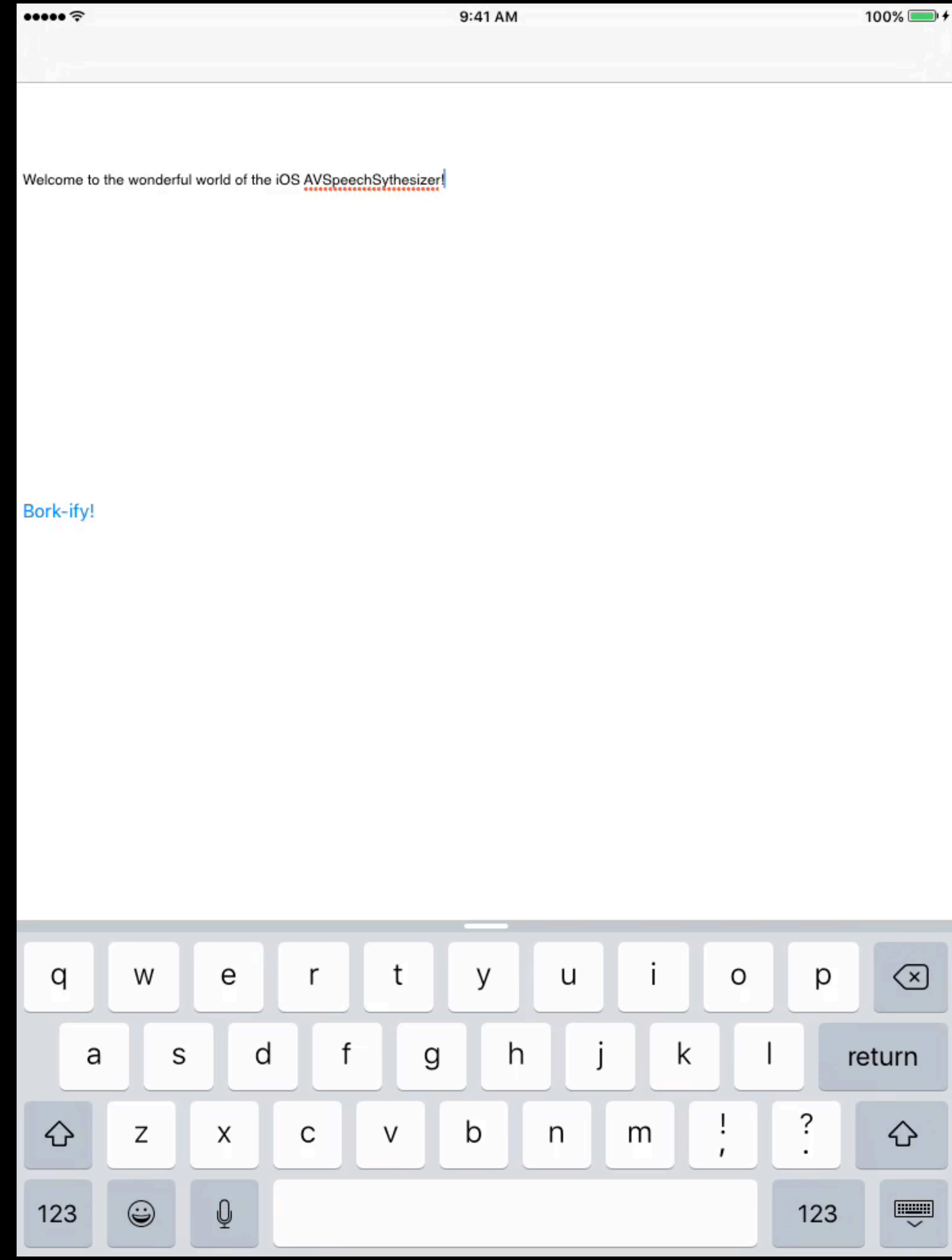
# IupBork: Mac

Uses NSSpeechSynthesizer via Objective-C or Swift



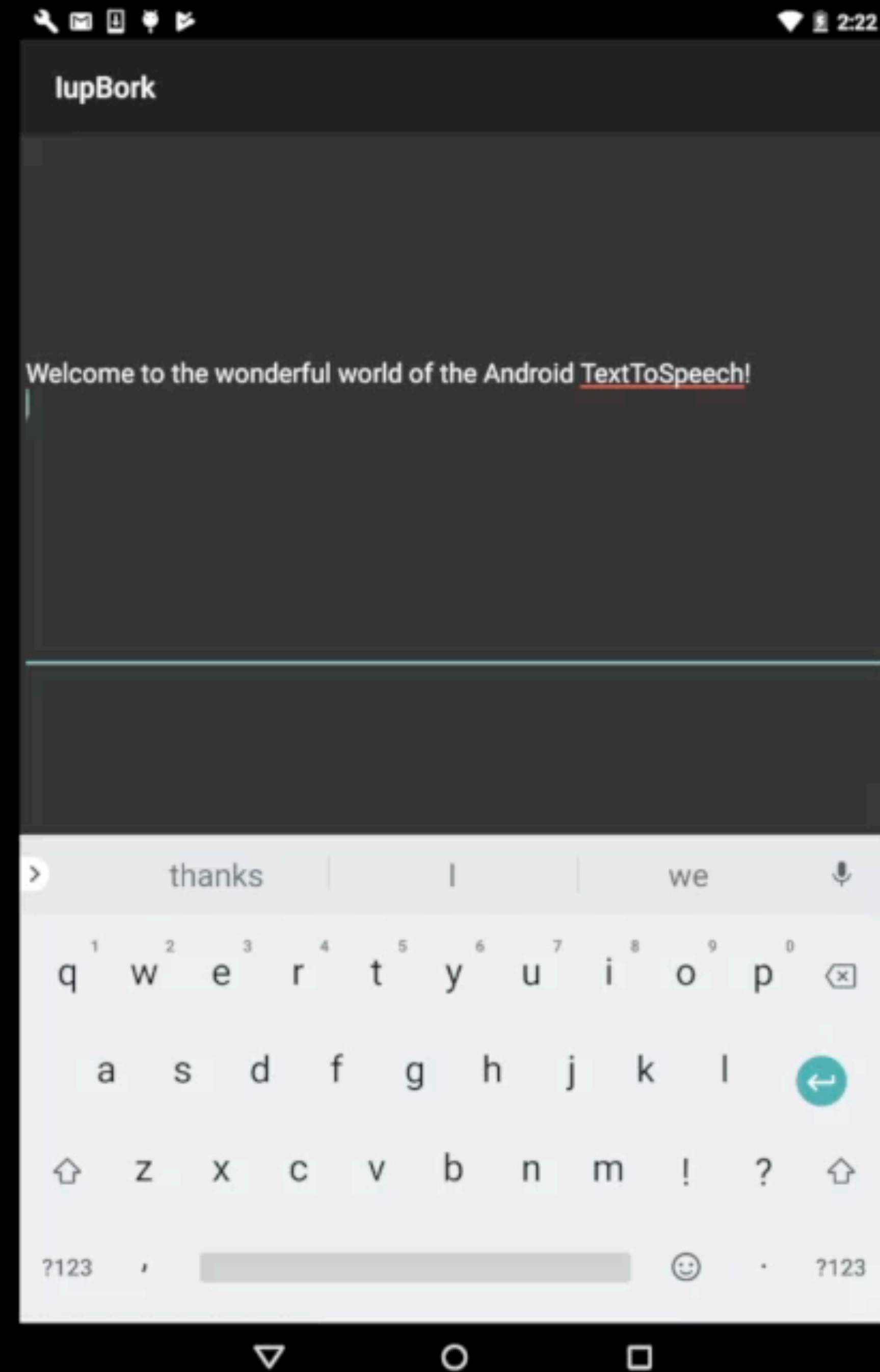
# IupBork: iOS

Uses AVSpeechSynthesizer via  
Objective-C or Swift



# IupBork: Android

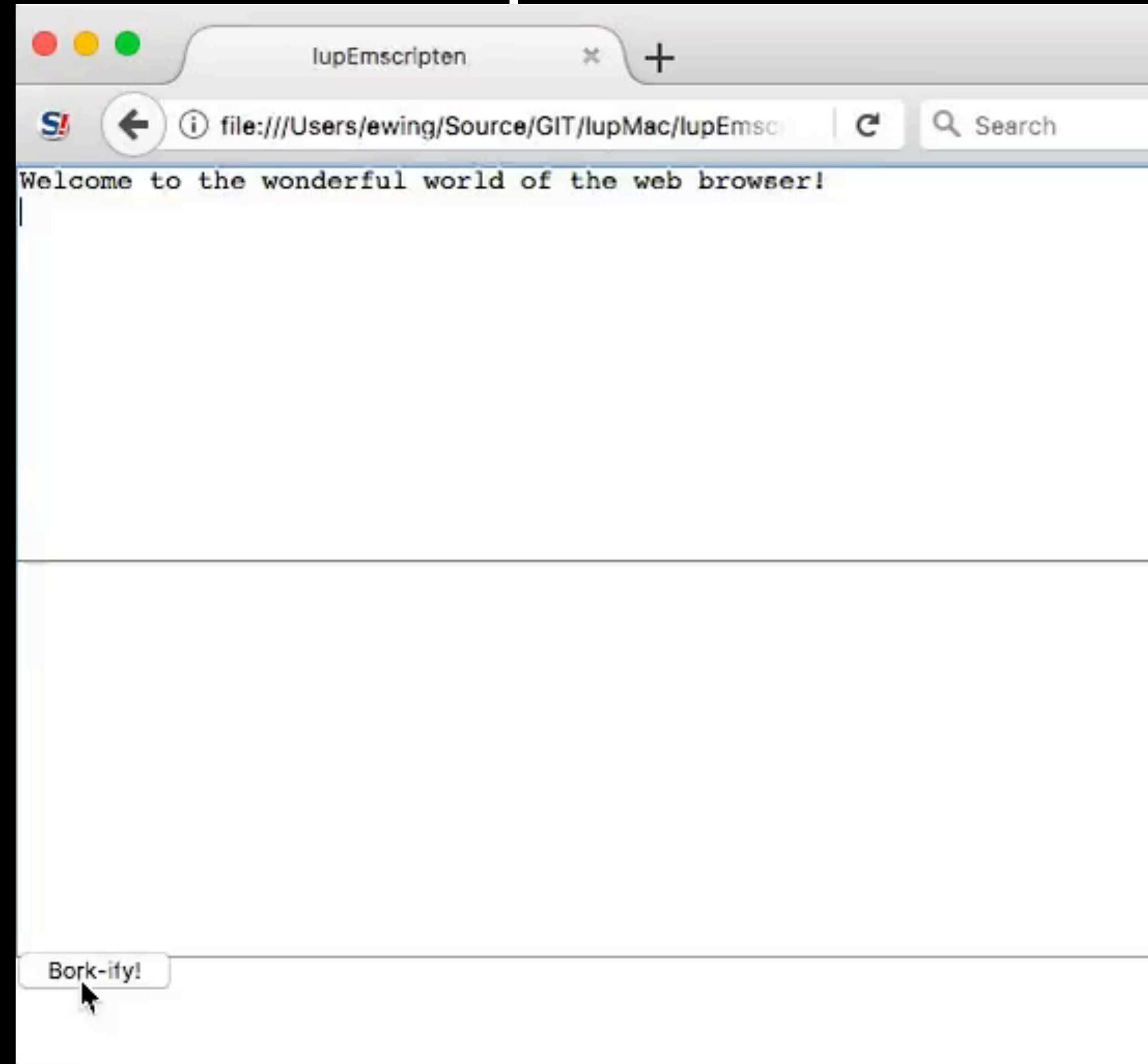
Uses TextToSpeech via Java



# 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?

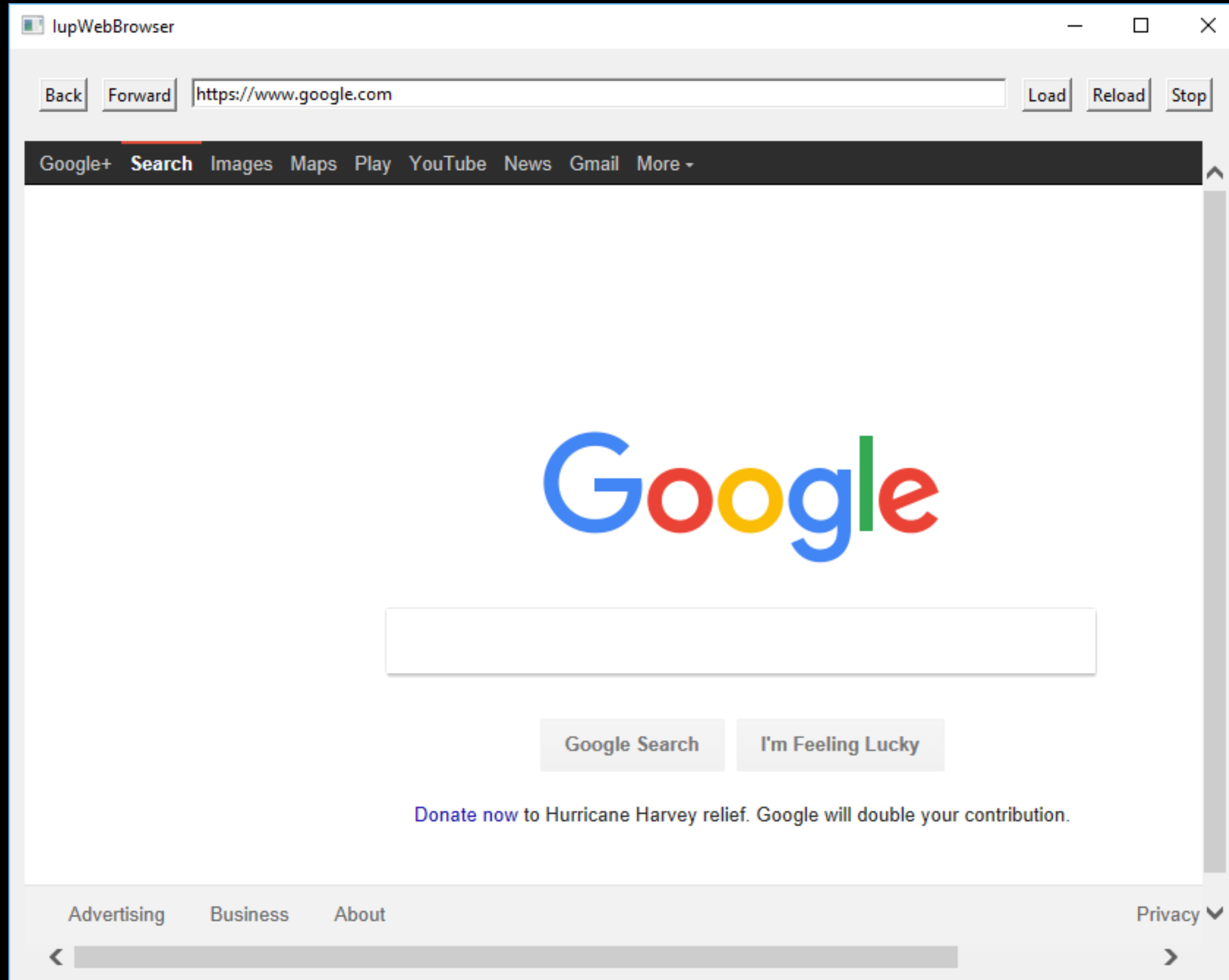
- Yes!
  - (In fact, that's exactly how we implement IUP in the first place)

# Example: IupWeb

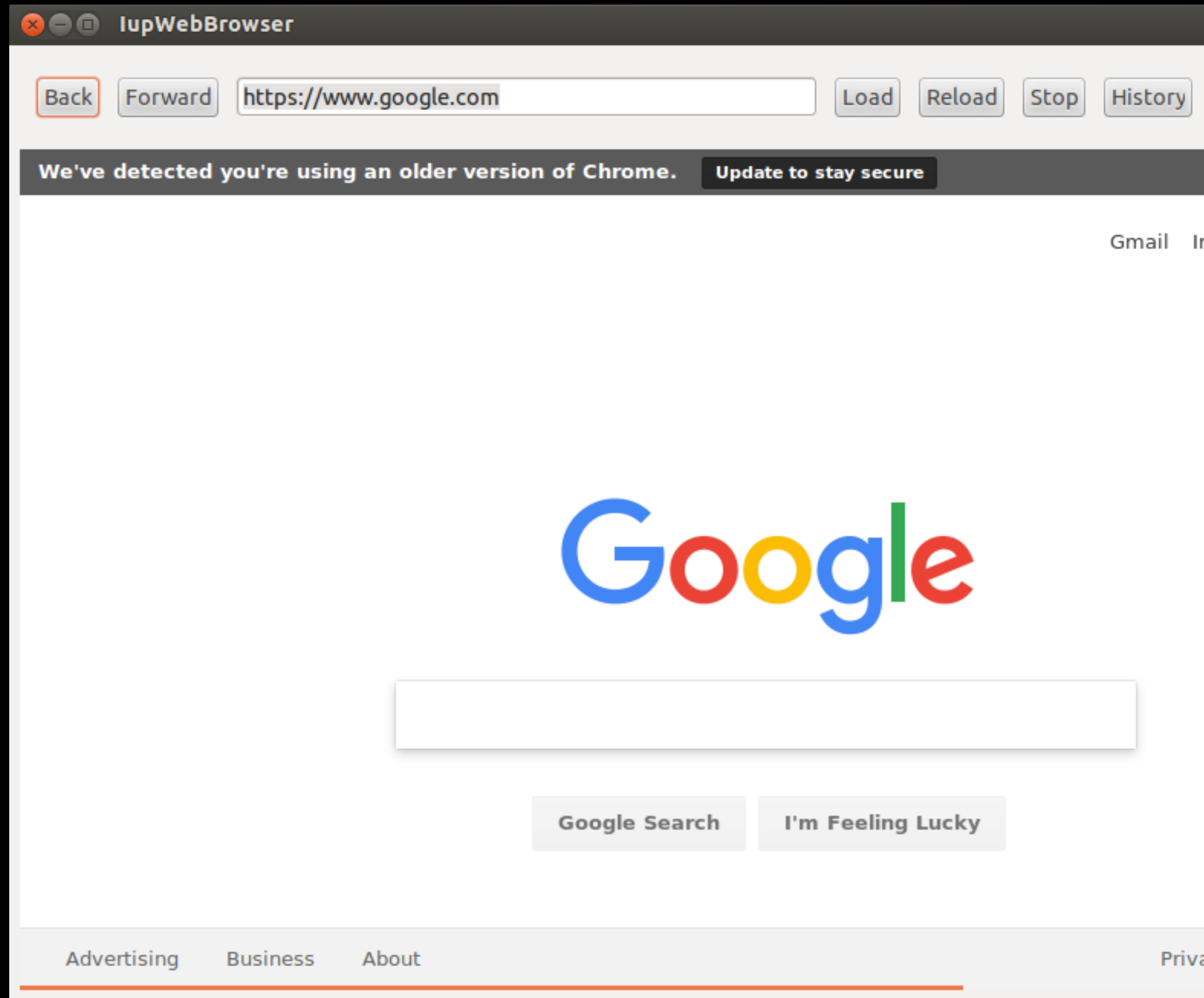
- Separate, not part of the main IUP library
  - Each platform has a different native web widget
    - Native => tiny profile

```
web = IupWebBrowser();
IupSetAttribute(web, "VALUE", "https://www.blurrrsdk.com");
dlg = IupDialog(web);
IupShow(dlg);
```

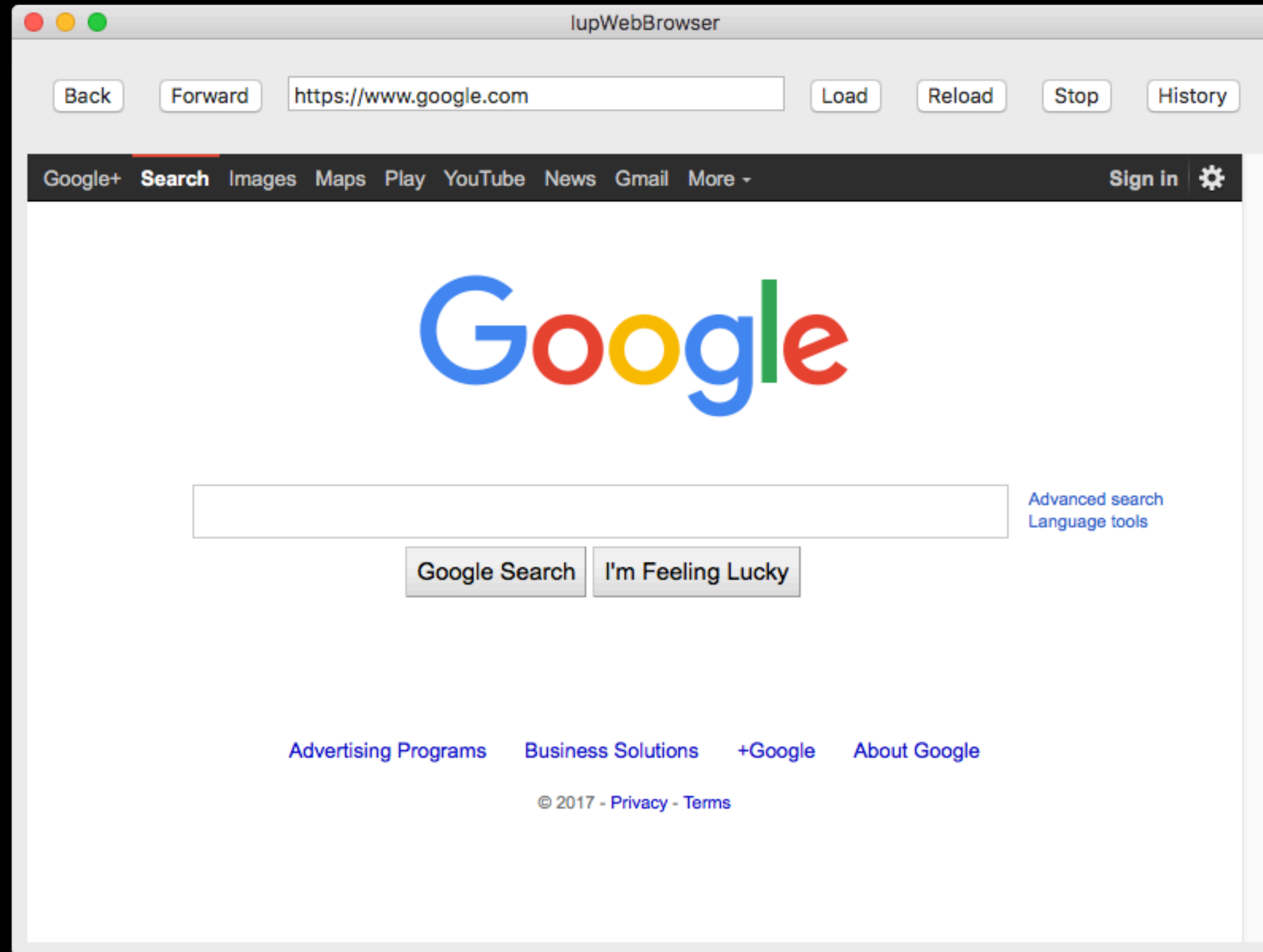
# IupWeb: Windows



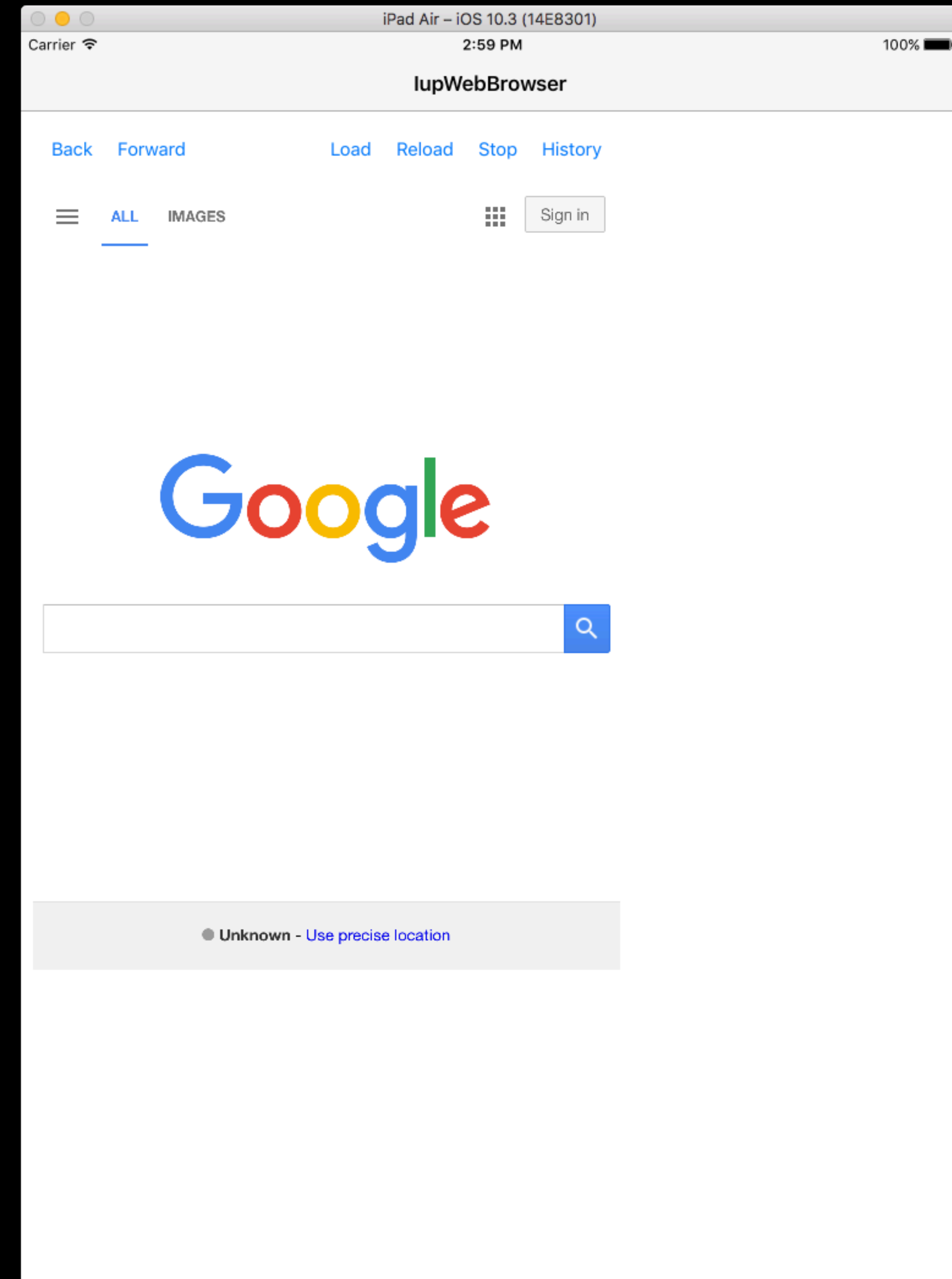
# IupWeb: Linux GTK



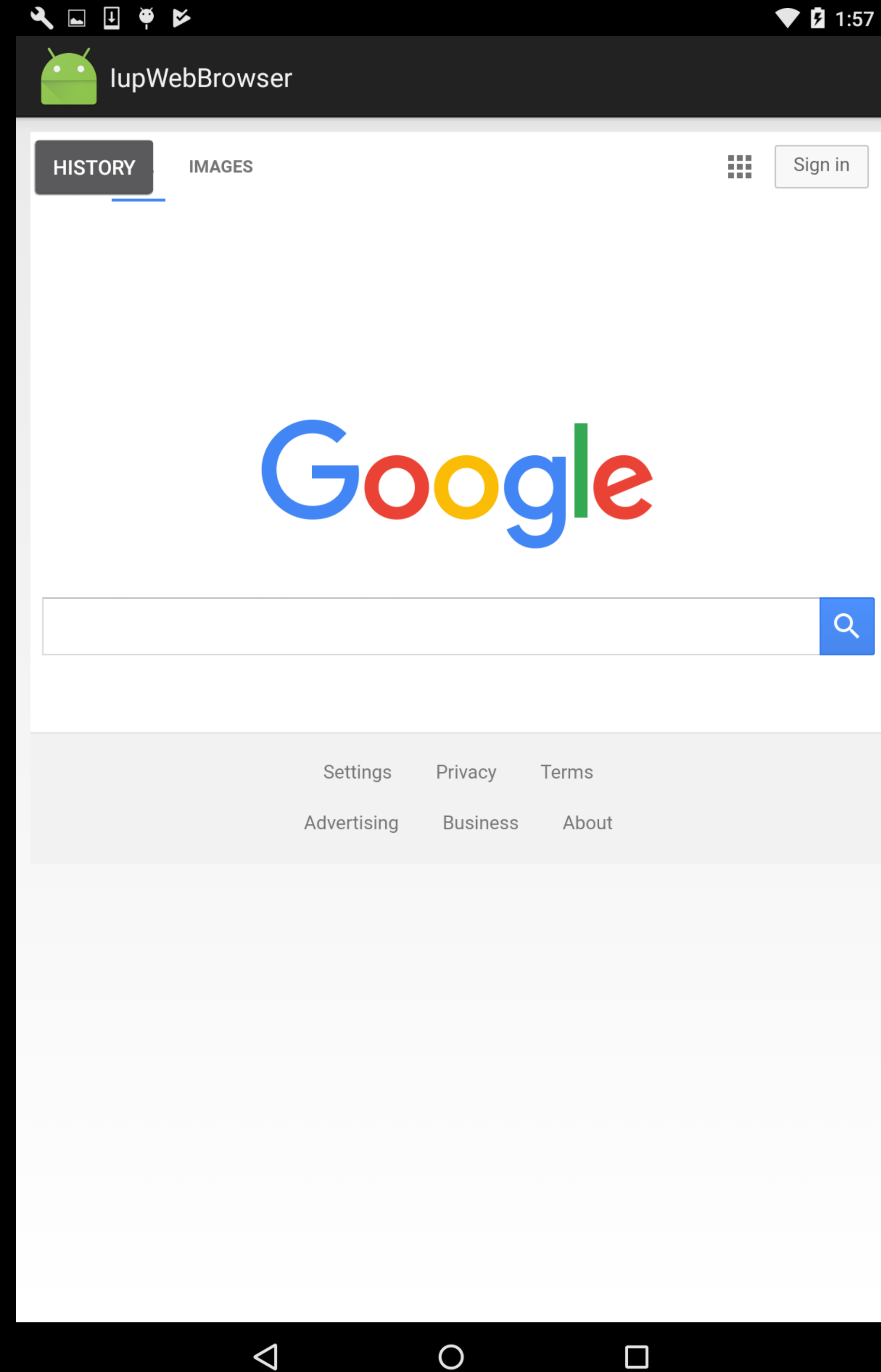
# IupWeb: Mac



# IupWeb: iOS



# IupWeb: Android



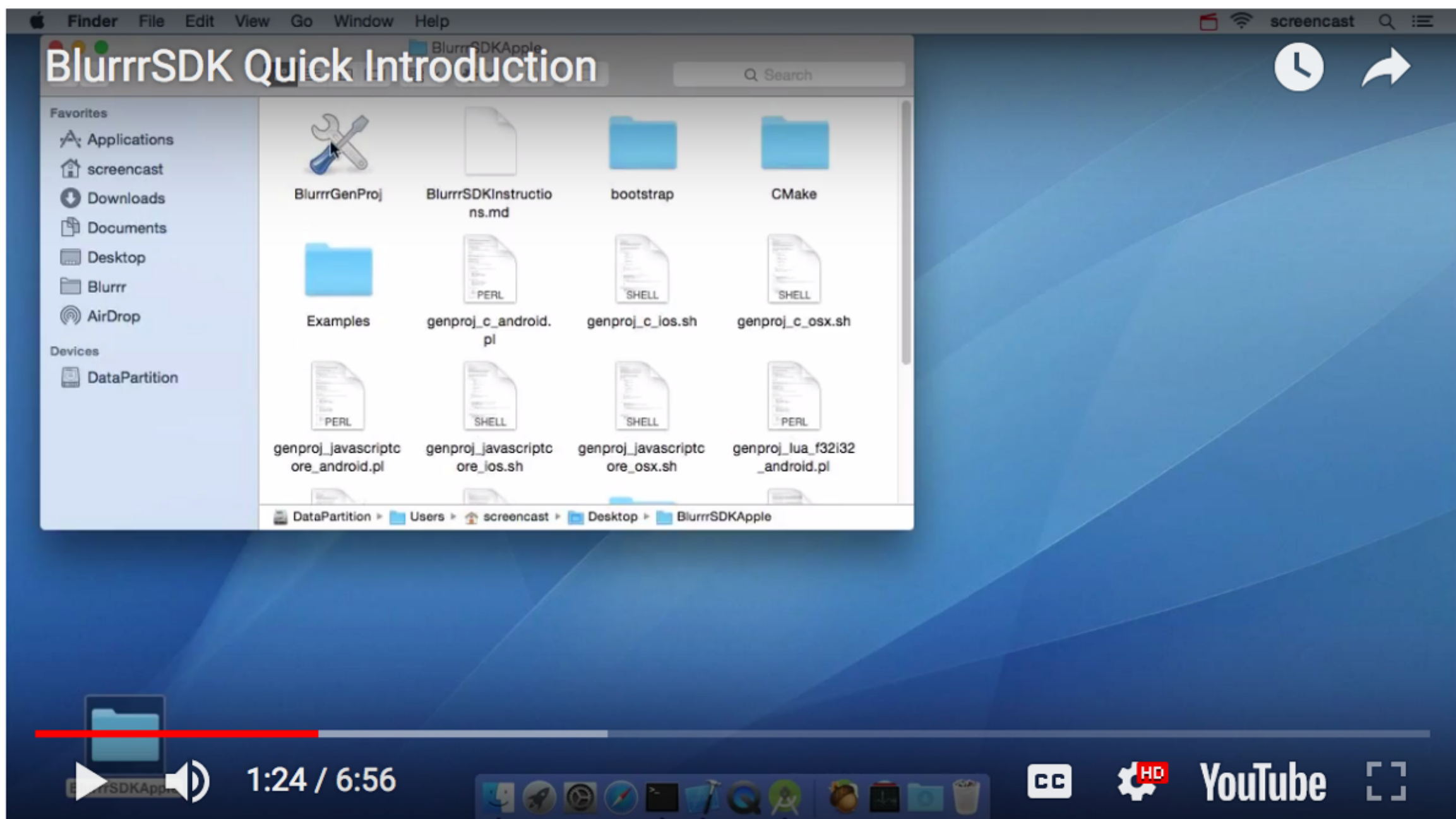
# lupWeb: Emscripten

lupWebBrowser

file:///Users/matzy/Development/lupEmscripten/BUILD/emscripten/webbrowser.html

Back Forward  Load Reload Stop History

## BlurrrSDK Quick Introduction



The screenshot shows a Mac desktop with a Finder window titled "BlurrrSDK Quick Introduction". The window displays a directory structure with the following items:

- BlurrGenProj (Folder)
- BlurrrSDKInstructions.md (Text File)
- bootstrap (Folder)
- CMake (Folder)
- Examples (Folder)
- genproj\_c\_android.pl (Perl Script)
- genproj\_c\_ios.sh (Shell Script)
- genproj\_c\_osx.sh (Shell Script)
- genproj\_javascriptc\_ore\_android.pl (Perl Script)
- genproj\_javascriptc\_ore\_ios.sh (Shell Script)
- genproj\_javascriptc\_ore\_osx.sh (Shell Script)
- genproj\_lua\_f32i32\_android.pl (Perl Script)

1:24 / 6:56

[BlurrrSDK Quick Introduction Part 1 \(view on YouTube\)](#)

[Part 2](#)

[Part 3](#)



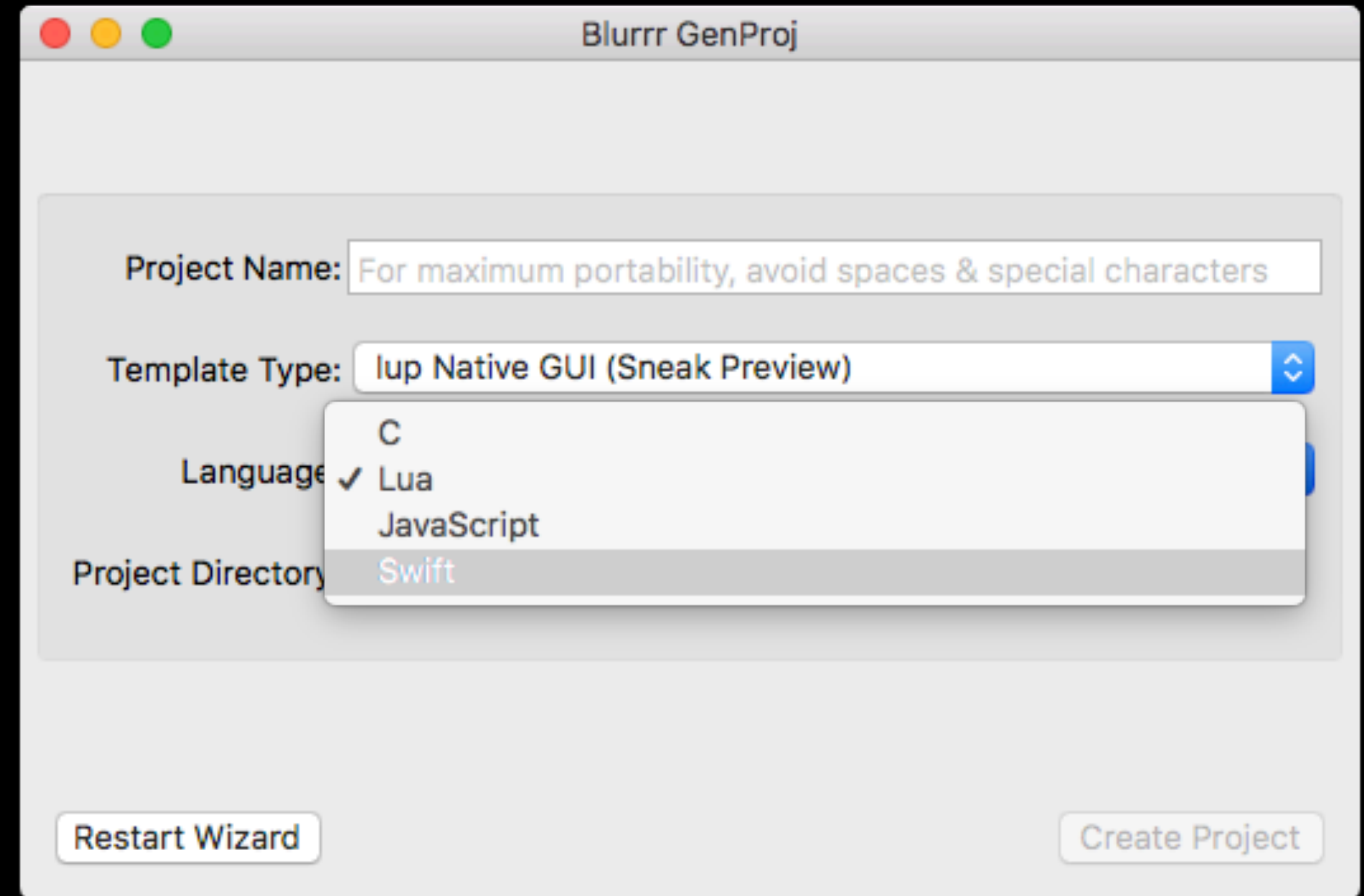
# Size is small too

The screenshot shows a file manager window titled "Linux" with a sidebar on the left and a main pane on the right. The sidebar lists "Computer" and "Network" sections with various system folders. The main pane shows a breadcrumb path: Home > Source > GIT > IupCocoa > BUILD > Linux. Below the path is a table of files and folders.

| Name                | Size     | Type                | Date Modified                   |
|---------------------|----------|---------------------|---------------------------------|
| ▶ CMakeFiles        | 28 items | folder              | Thu 21 Sep 2017 05:38:25 PM PDT |
| ▶ CMakeModules      | 1 item   | folder              | Fri 01 Sep 2017 06:26:12 PM PDT |
| CMakeCache.txt      | 33.6 kB  | plain text document | Fri 01 Sep 2017 06:26:12 PM PDT |
| cmake_install.cmake | 8.5 kB   | CMake source code   | Fri 01 Sep 2017 06:26:12 PM PDT |
| libiup.so           | 769.6 kB | shared library      | Sat 02 Sep 2017 01:27:07 AM PDT |
| libiupweb.so        | 20.4 kB  | shared library      | Sat 02 Sep 2017 01:27:08 AM PDT |
| Makefile            | 185.4 kB | Makefile            | Fri 01 Sep 2017 06:26:12 PM PDT |
| webbrowser          | 13.9 kB  | executable          | Thu 21 Sep 2017 05:38:25 PM PDT |

# Links

- Blurrr SDK now shipping with “Sneak Preview” IUP
  - <https://blurrrsdk.com>
  - Contains templates & examples seen today (e.g lupBork, lupWeb)
- Repos:
  - <https://github.com/ewmailing/lupCocoa>
  - <https://github.com/ewmailing/lupCocoaTouch>
  - <https://github.com/ewmailing/lupAndroid>
  - <https://github.com/ewmailing/lupEmscripten>



# Please Support Us

- Purchase Blurrr SDK ([blurrrsdk.com](http://blurrrsdk.com))
- Donation link at [blurrrsdk.com](http://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



Adobe



Macromedia



Corona SDK



Platino



@codinginswift



Thank you

**Eric Wing**

blurrrsdk.com

@ewingfighter / @BlurrrSDK

**Chris Matzenbach**

cmatzenbach@gmail.com