A little code goes a long way
Cross-platform game development with Lua

Ivan Beliy, Software Engineer
A bit of History

- Rich Lua support in Marmalade SDK
- Lua is used as a main scripting language in AAA titles
- Big interest in rapid app development and game prototyping
Marmalade for C++

- Pure C platform abstraction layer
- Unique custom build system powers everything
- Bundled toolchain - users need no platform SDK or code
- Deploy to: iOS, Android, BlackBerry 10, Windows Phone 8, LG TV, Roku TV, Mac and Windows desktop
- Plugins for Xcode (Mac) and Visual Studio (PC) for debugging
- Easily package and push to devices or store with the Hub
- Middleware, engines and extensions
2D development: the problem

- Developers were asking for rapid dev tools
- C++ runs fast but is slow to write!
- Many developers don’t want to learn C++
- Existing 2D RAD tools:
  - Very high level – nice editors but limited code support
  - Or lower level but not very extendable
  - Closed source
  - iOS and Android only
  - Lack of professional support
  - Rely on cloud-based building
A solution: Marmalade Quick
Marmalade… quick!

- Write apps in Lua: the fastest scripting language, simple but powerful
- No need to know or use C++
- Do more with less code…
- …but open source and extendable if needed
- Uses popular frameworks like Cocos2d under the hood
- Utilise Marmalades robust cross platform foundations, MKB project system and deployment options
- Full Lua IDE for debugging
Where to?

- OS
- Android
- Windows Phone 8
- Windows Desktop
- Mac Desktop
- Tizen
- BlackBerry 10
- BlackBerry Playbook
- Roku
How Quick works

- Quick’s APIs and your game code are entirely in Lua
- Quick has bindings from Lua APIs to C++ implementations
- A precompiled Marmalade C++ app implements the performance critical engine parts using Cocos2d-x, provides the bindings and loads & executes your Lua code
- Lua code runs via a C++ implementation of Lua that is built into the app
When the app is launched…

- The C++ part launches like a regular Marmalade app, including desktop Simulator support
- It does initialisation: system, memory, GL, event handlers…
- It creates a Lua runtime to run scripts
- It initialises the Quick engine code
- It loads your app’s main.lua file and executes “your app”

- You can simply edit Lua code and reload updates live in the desktop simulator
A simple example

Display a label and a textured button that will change the label’s color:

```lua
local label = director:createLabel(0, 0, "Hi there!")
local button = director:createSprite(0, 100, "textures/button.png")

function button:touch(event)
    if (event.phase == "began") then
        label.color = color.blue
    end
end

button:addEventListener("touch", button)
```
function systemEvents:touch(event)
    if event.phase == "began" then
        local b = director:createSprite(event.x, event.y, "ball.png")
        physics:addNode(b, {radius=40})
        b:addEventListener("collision", bodyCollision)
    end
end

function bodyCollision(event, target)
    if event.phase == "began" then
        local c = director:createSprite(event.x, event.y, "cross.png")
        tween:to(c, {time=0.25, xScale=0, yScale=0})
    end
end
Open source components

- **OpenQuick (free)** contains:
  - C++ source for the Quick frameworks, including integration of Cocos2d, Box2d, etc.
  - Open source Lua wrappings and additional APIs to provide a super easy to use high level interface

- You can get this from GitHub and compile as platform-specific projects without Marmalade

- **Marmalade Quick (licensed)** adds:
  - Non-standards-based (but super useful) features like accelerometer, location and in-app billing, via the abstraction APIs in the regular Marmalade C++ product
  - Support for Hub, deployment and debugging tools
  - Robust internals to hide device and GPU fragmentation issues
Extending Quick

- The C++ and Lua parts of Quick are open source: you can extend and improve them
- Quick includes **toLua**, which allows you to wrap any Marmalade C++ API with Lua code and support it in Quick

- Marmalade C++ supports loads of C++ extensions and middleware to take advantage of
- Marmalade 1st party C++ APIs are also extendable:
  - Many are provided as open source extensions
  - An extensions kit allows you to create new C APIs to give access to additional platform features

- Using Cocos2d and Lua means that various sprite and scene editors could be easily integrated…
Do more with Marmalade

Lua

Lua engine components

Lua wrappers for some S3E APIs

New Lua wrappers for S3E & C++ APIs

New Lua wrappers for Extensions

Quick’s C++ engine components

Marmalade Quick out of the box

Platform abstraction

Marmalade S3E Core APIs
(file, memory, compass, accelerometer, video…)

C++ Middleware

Extensions
Cases

- Shoot Me!
- Dream Candy Planet
- Signal to The Stars
- Coins and Stuff