Aw2D-Android

Ferdinand Majerech

Univerzita Pavla Jozefa Šafárika v Košiciach
UPJŠ

May 16, 2013

NDK

- Native development kit for Android
- Allows development in compiled languages - C, C++ by default
- Reason: speed
- Designed to <u>extend</u> apps written for Dalvik
- .so accessed through JNI from Java

C/C++ in NDK

- By default, no C++ exceptions, no RTTI, no STL
  - Can be enabled (with multiple STL implementations).
- Dynamic linking sucks
  - Static ctors called twice, static dtors never called
- Very few libs guaranteed to be present
  - Zlib, dl, EGL+GLES1/2, SLES, jnigraphics, OpenMAX
  - Native activity stuff (with a C API as readable as libPNG)
    - Allows “fully native” code, i.e. JNI is needed for anything useful
- Need JNI to access most APIs.
  - Some frameworks exist, better and worse

OpenGL ES 2.0 in practice

- Cleaned up spec - no immediate mode, etc.
- Drivers suck (everywhere), not in line with spec
- NPOT textures silently fail (Tegra)
- onPause wastes GL context, or <u>maybe</u> wastes it

GLSL ES

- No const arrays
- Arrays in spec, might or might not work on a device
- Float precision specifiers, affect performance:
  - highp: 16bit spec, 31bit common - vertices
  - mediump: 10bit spec, 15bit common - bounded distances
  - lowp: 8bit spec and used in practice - colors, normals

```
lowp vec3 pointLighting(in int light,
                        in lowp vec3 pixelDiffuse,
                        in lowp vec3 pixelNormal,
                        in highp vec3 pixelPosition);
```

ABI

- armeabi - ARMv5+, no native floating point
  - useless for games
- armeabi-v7a - ARMv7, has hardfp
- SIMD not guaranteed to be present
  - Must check for NEON at runtime
  - Need separate binaries for devices with and without NEON

TLDR

- Need separate binaries for ABIs and ABI extensions
- Native is a royal PITA compared to Linux/Windows
- Do not want to use NDK directly

Necessitas

- Qt for Android
- Beta - no stable yet(To be stable in Qt 5.2)
- Contains Qt creator modified for Android
- Similar to ADT - fast build/debug cycle
- Generates all Java wrapper code
- Development in C++ or QML/Javascript (or both)
- Qt project file takes role of manifest

Qt

- C++ framework similar to .NET/Java libraries
- Low overhead mid-level OpenGL wrappers
- QML for declarative apps with Javascript
- Supports some phone features
  - Qt was built for Maemo/Meego/Meltemi/Mer/jolla
- Cross-platform (Android, BB10, Jolla, Win/Mac/Linux ...)
  - iOS in works (well, Android, too)

Necessitas is not stable yet

- Some stuff must be hand-edited in manifest
  - Will get overwritten when generated... URRGH
- Android lifecycle is handled on background
  - Qt event loop stops
  - If timing outside the Qt event loop, no onPause/onStop for you
  - Qt ensures GL context does not get murdered
  - No way to onDestroy yet (Qt has an API for it - not implemented yet)
- If a feature is not supported, need to use JNI
  - Clashing with generated code

Necessitas apps

- Ministro
- At first launch, asks user to download the needed subset of Qt
- Gets correct builds for Android version and ABI on device
- Libs shared between Necessitas apps
- Other libs - must pack with app, built for correct ABI

Aw2D-Android

- Awesome2D for Android
- Proof-of-concept of 3D lighting for 2D graphics
- In NDK/C++/Necessitas, written specifically for Android/ARM
- Eventually to be cannibalized into an even more cross-platform Awesome2D in D

Awesome2D

- Dynamic “3D” lighting in 2D graphics
- Using colors/normals/positions pre-rendered from 3D data
- Reconstructing 3D data per pixel
- Using (modified) Blinn-Phong lighting
- Graphics comparable to 3D isometric games (Diablo 3)
- Low vertex, medium-high fragment overhead

Current state

- Basic 3D lighting demo/benchmark.
- The only interactivity is the ability to move two lights.
- Brute force approach at the moment (no texture packing, etc.)
- Surprisingly good performance (considering brute-force)
    - But more benchmarking work is needed
        - Esp. map loading

Implementation

- Basic sprites
  - No facings yet (need a YAML parser (yaml-cpp?))
- GLSL ES shaders doing the lighting
- Directional, point light sources
- Uniform wrapper to avoid excessive reuploading
  - More readable uniform management; to be backported
- Simple 2D camera

TODO

- Full sprite parsing
- Map loading
- Optimizations from desktop Awesome2D
- More ARM-specific optimizations
  - 16-bit texcoords, vertex & buffer alignment,
- Test more devices/use device specific GL extensions

Thank you for your attention!