Vision on the Android

- Shyam S Kumar
Outline

Android Environment

• Software on the Android
• User Interface
• Computation

Application to perform Object Detection

• Overview of the vision based components
• Overview of the client-server framework
• Discussion of the Android Components
Based on Customised Java

- Number of Android-specific implementations
- Also provides support for non-standard features

Uses a Dalvik VM for managing code

Multi-Threaded Applications

XML for GUI and Application Manifest
Largely re-implemented Java Framework

Redefines some standard Java interfaces

Extends to provide direct support for:

- Internet Access
- 3D graphics using OpenGL
- Database storage (SQLite)
- Image I/O and compression
- Map Access
- Peripheral Access (GPS, Accelerometer, )
- plenty more …
Java vs C/C++

Java

• Platform Independent
• Class-Based
  • Member Functions
  • Inheritance
  • Enum.s
  • Member Variables ...
• Virtual Machine Interprets Java bytecode and controls execution, memory etc
• VM controlled Memory Management
• Compiler Philosophy: Strict guidelines
• Libraries: .jar

C/C++

• Platform Dependent
• Global Functions
• Global Variables
• Structures, Enums and Classes
  • Member Functions
  • Member Variables
  • Inheritance
  • ...
• User controlled Memory Management
• Execution controlled by self with OS interaction
• Compiler Philosophy: User knows what its doing
• Libraries: platform dependent (so,dll ...)
Dalvik VM (Virtual Machine)

- Designed by Dan Bornstein (@ Google)
- Register based-VM
  - Optimised for low memory performance
  - Designed for embedded devices and mobile platforms
- Just-in-time Compiler (since 2.2)
- Different from standard Java VM in the way instructions are loaded
- More compact packaging compared to JARs
Main Components of an App

User Interface and ‘Views’

• Menus
• Lists
• Buttons
• Images

Organisation and ‘Activities’

• Reading / writing to memory
• Calculations
• Network I/O
Like an itinerary for the OS to follow and verify
- Lists Activities
- Explicitly requests for hardware access
- State need for special security needs
- Library Links
- Minimum API level used
- Written in XML
- Also has a interface to avoid use of XML
Various types of ‘Views’

- Map View
- Web View ...

Defining Views

- in xml, or (Encouraged)
- explicit code (less preferred)

Adapters to build-upon
Sample Views

Default Views

Custom View
The core of an Application

- Providing functionality
- Reacting to inputs

Organised as discrete Parts of an Application

- Handlers to provide discrete functionality
- Example
  - Network I/O
  - Computation
  - Animation
Activity State Machine

Create
  - Initialise members
  - Set state variables

Start
  - Load persistent information
  - Initialise Views

Resume
  - Start Computation
  - Network I/O ...

Pause
  - Begin ending animations, calculations etc
  - Prepare to stop
  - Set state variables

Stop
  - Save Persistent variables

Destroy
  - Ensure all variables removed.
  - Close running activities
The Droid

* CPU - 550 MHz (Arm® Cortex™ A8 processor)
* RAM - 256 MB
* App Memory - 16MB (Android Limit)
* Network I/O - 3G and 802.11[b,g]
* Camera - 5MP
* Sensors -
  * Inertial Measurement Unit – IMU (using Accelerometers)
  * Ambient Light
  * Proximity Sensor
  * Compass
* and on and on ...
A Vision App on the Cloud
A Vision App on the cloud ...

The Premise:
Vision-based components

Object Detector
- Depth-Encoded Hough Voting based detector *
- Implemented on the cloud
- Implemented using OpenCV in C++

*Depth-Encoded Hough Voting for Joint Object Detection and Shape Recovery. Authors: Sun et al ECCV 2010
The Android Components

* Camera
* Internet Access – (WiFi or 3G)
* Activities to organise the server interaction
  * A few pages of Java for Android
  * OpenGL
* Time – to search google for solutions!
What’s on the cloud?

* The cloud / server is currently based off of an AMD machine:
  * Ubuntu-10.04
  * 3-core (2.8GHz, 4GB) AMD powerhouse
* It runs an instance of apache-tomcat 6.0.26
  * Java based standard server module
  * It interacts with the vision modules using system calls or (optionally) using the java-native-interface (jni).
* Native C-Code to perform object Detection
1) Call system code (run non-java c-code binary) for DEHV
2) Detect Objects (c-code)
3) Save copies on cloud
4) Send Locations back to client
5) Conclude

Choices
1) Receive images
2) Set-up view to display choices
3) Display
4) Wait for user interaction
5) Conclude

Click!
1) Click!
2) Scale image to size
3) Send to Cloud

Process Flow
~1-3 seconds

Client
~4-5 seconds

Cloud
JNIs – Java Native Interface

- Java Native Interface
- Utility available on java to use C/C++ in Java
- Created to reuse existing C/C++ code in Java
- Very useful tool to use existing, powerful libraries
- (however) Strongly discouraged to simply port any C/C++ code
Why C/C++ on Java?

* Well ...
* Lot of Vision toolsets and code available in C/C++
* Lot of non-vision tools also available in C/C++ for:
  * graphics rendering
  * Image Processing
  * Abundant Mathematics / Physics support
* It is FASTER!
C/C++ (native code) on Android

- Can Port using Java Native Interface (JNI)
- NDK - Native Development Kit (Android Support)
- No default support in NDK for STL (vectors, lists ...)
- 3rd Party tools to help with this (crystax NDK)
Please Bring your laptops

- Will have a code-along demo

Topics

- Cover basics of writing a simple app on the android
- Cover the essentials of JNI on the Android

Don’t Forget your Androids!
Fin!

Fin!