# Intro

* My Self
* Candlelight Software Consulting and Contracting
* Java, Android, Embedded, Augmented Reality
* This Presentation:
  + Anatomy of an real world AR Application
  + Not cover Rubik Cube Solution Logic
  + Probably markets
  + Probably hardware

# Enterprise Augmented Reality

* Smart Glasses
* Fully Immersed (i.e., Full Graphics Overlay)
* Project Primary Markets will be Enterprise (Business Needs)
* Certain Jundra
* Probably not “App Store” and general consumer markets
* Possible vendors: IBM, Oracle, SAP, etc…

# AR Rubik Cube App

* Iconic
* Next to impossible for a mortal human being
* Clear example of Synergetic AR
* Relatively Simple
* Representative of real world AR App
* Intended Purpose:
  + Party Gadget, or Conference Gadget
  + Reference Standard for Smart Glasses
  + Base Line Software Technology
  + Open Source Domain
* Source Code: github.com/AndroidSteve/Rubik-Solver
* Potentially will become the Angry Bird of Smart Glasses.

# Ecosystem

* Android
* Eclipse
* OpenCV
* OpenGL
* Source code on GitHub
* Developed on conventional Smart Phone
* All in Java
* See <http://www.cl-sw.com/blog/1> for installation

# Basic Architecture

* Model-View-Controller
* Controller:
  + Image Recognition
  + Gesture Recognition
  + Model Recognition
* Model:
  + Cube and Application State
* View:
  + Overlay Graphics

# Controller - Overview

* Output is User and State Change Events
* Lion Share of code and complexity
* OpenCV
  + Large Collection of Facilities
  + Developer must have strategy
* Many processing steps

# Controller – Edge Detection

* Greyscale edge detection
* Result is binary image
* << photo >>

# Controller – Image Dilation

* Simply expansion of white areas
* Result is binary image
* << photo >>

# Controller – Contour Recognition

* Canny Edge Detection
* Result is a set of closed contours
* << photo >>

# Controller – Polygon Generation

* Obtain N sided (where N is small) Polygon for each Contour
* Result is a set of polygons
* << photo >>

# Controller – Rhombi Generation

* Technically a Parallelogram
* Result is a set of Features
* “Features” with characteristics
  + Location
  + Edge Sizes
  + Orientation
* << photo >>

# Controller – Rubik Face Recognition

* 2D Reconstruction
* Determine a 3x3 grid from available Rhombi
* Result is location, size and orientation of grid
* Available Rhombi are a subset
* << table >>
* << photo >>

# Controller – 3D Reconstruction

* Pose Estimator
* Provide set of Image Features and Model Features
* Result is location and orientation of cube in 3D space
* << photo >>

# Controller – Gesture Recognition

* Stable Face
* Stable New Face (i.e., different from previous)

# Controller – Application State Machine

* Observation phase cube rotation
* Solution phase cube edge rotation

# Model

* State of cube: location, orientation
* Face content of each side
* Application State
* Solution State

# View

* All OpenGL
* Pilot Cube
* Invisible Cube Overlay
* User Direction Arrows

# Summary of Rubik Cube App

* Nicely demonstrates a well know problem
* Nicely demonstrates power of AR
* However, this app not ready for broad use:
  + Variation of Rubik Cubes (Shape and Colors)
  + Variation of Luminous
  + Improved Scaling
  + Variation of Devices : libraries, cpus, gpus, cameras, displays
  + Variation of User Gestures
* Real target is of course Smart Glasses (i.e., Hands Free)
* Android (iOS) excellent ecosystems
* High Power Consumption!
* Camera Calibration Tricky!

# CPU and Power Consumption

* CPU Breakdown
* Frame Rate vs. Latency
* Typical Power Consumption Rate

# Android (iOS) Ecosystem Pros and Cons

* Existing infrastructure well suited for AR Apps
* Databases, WiFi, Bluetooth, Internet, Accounts, etc...
* Existing Smart Phone UI very useful

# IDE : Eclipse verse Android Studio

* Open Source (no formal support) vs. Close Source (formal support)
* But is a tool, not source code! Thus, no legal issues in adopting.
* AS support for C and C++: specifically build Linux and Android Framework
* AS support for OpenCL and OpenGL (shader language)
* Like WordPress verse SquareSpace

# Third Party Contributions: OpenXX

* Very Impressive Functionality
* Open GL already in place
* Open CV Installation required
* However:
  + APIs are very historic
  + APIs are not very Java like
  + APIs are not strongly typed
  + Many Incompatibilities

# OpenGL Comments

# OpenCV Comments

* Deep background in clever algorithms

# Nature of Enterprise AR Apps

* Architecture is really simple Model-View-Controller
* Image Recognition is lion-share and challenging
* Much more challenging to author than smart phone apps

# Technology Timing

* Image processing technology (in development for past 40 years)
* Portable hand held platforms (from Smart Phone)
* Heavy GPU processing power (from Game Industry)
* Android (iOS) existing ecosystem

# Business Models

* Information: email, movies
* Games
* Enterprise
* Consumer vs. Corporate
* App Store vs. Xxxxxxxxxxxx

# Early Adoption Markets

* Enterprise / Business
  + Production Assembly
  + Military Mechanics Maintenance
  + Inventory
  + Instructional Training
  + Christmas Tree Bicycle Assembly
* Predict firms like IBM, Oracle, SAP, etc…

# Smart Glasses Hardware Convergence

* AR on Smart Phone / Table interesting, but powerful on Smart Glasses
* Many different architecture and designs
* Predict “Minimal Smart Glasses” will emerge as dominate
  + Camera, Displays, some IMU Sensors, maybe Eye Tracking
  + Light Weight: no batteries
  + USB 3.0 to host Smart Phone: Power & Connectivity
  + Android or Apple are ideal hosts
  + Auxiliary Power Pack

# Software Convergence: AR API

* Write Once, works on all Smart Glasses
* Necessary for AR to mature
* Blending of Interests:
  + App Authors (what is needed)
  + Device Manf (driver implementation)
* Probably Open-Source (light licensing)
* Dual Android / iOS support (Unity?)
* Additional AR Functionality
* Predict several device vendors to collaborate and foster
* Refinement of OpenCV
* Subset of OpenGL

Purpose

* Self-Education
* Broader Educational

Status

* Describe Functionality

Nature

* Image Recognition is lion-share
* Rather easy compared to more real world situations.
* However, this app not ready for broad use:
  + Variation of Rubik Cubes (Shape and Colors)
  + Variation of Luminous
  + Variation of Devices : libraries, cpu’s, gpu’s
  + Variation of User Guesters
* Real target is Smart Glasses: i.e., Hands Free

Product Possibilities

* Party Gadget, or Conference Gadget
* Really serves as demo
* However, truly demonstrates potential

Bigger Markets

* Production Assembly
* Military equipment repair
* Emergency Medical Care
* Ikea Furniture Assembly
* Christmas Bicycle Assembly

This Presentation

* Walk Through the anatomy of an AR app.
* Discuss deficiencies

Design

* Model View Controller
* Image Recognition
* Gesture Recognition

Core Technologies:

* OpenCV (Khrnoous Java API)
* OpenGL (Google API wrapper)
* OpenCL (no official java wrapper, JNI to C++ used).
* Android

Comments on Technologies

* OpenCV
  + Several language APIs
  + org.opencv
  + Class Mat everywhere
  + Auto generated Java API ??
  + Matrix storage Row-Column layout ??
  + Seems like roots are in experimental and Mat Lab.
* OpenGL
  + Seems like written for C++ (using operator overloading)
  + javax.microedition.khronos.opengles
  + All matrices represented by float[]
  + Matrix storage Column-Row layout.
  + Auto generated Java API?a
  + Roots are in gaming industry (performance is a must)
* Java APIs for both reflect history
* Neither is very strong typed
* Some overlap between the two
* Substantial “impedance matching”
  + Float vs Double
  + Degrees vs radians
  + X-Y-Z Coordinate definitions
  + OpenGL and OpenCV not “super polished:” i.e., as Android Studio is to Eclipse.
* Opportunity for a strong-java unification layer, but is there yet sufficient wisdom?
  + Look for convergence platform to be a battleground of the Android AR vendors.
  + Like Unity, software platforms that unifies and absorbs hardware differences among vendors.
  + Since this would be an API and code, resistance to a commercial solution.
* IDE : Eclipse verse Android Studio
  + Open Source (no formal support) vs. Close Source (formal support)
  + But is a tool, not source code! Thus, no legal issues.
  + AS support for C and C++: specifically build Linux and Android Framework
  + Like WordPress verse SquareSpace.

Hardware

* Google Cardboard AR demonstrates how hardware agnostic AR is.
* My Fortunetelling is:
  + Smart Glasses are effectively auxiliary cameras and display to smartphone.
  + Expect to simple Smart Glasses with USB3 to smartphone.
  + Leverage users existing smartphone.
  + Hardware race to the bottom: features, quality, cost.
* Extra battery pack for eight hour day probably needed.

Timing

* Image processing technology (in development for past 40 years)
* Portable hand held platforms (from Smart Phone)
* Heavy GPU processing power (from Game Industry)
* Android (bring all together)

# Chapter

Basic Environment Setup and Hello World App

Nature

* Ecosystem defined for a point in time.

Packages

* Android ATD (Eclipse + Android SDK)
* Android NDK
* OpenCV Library
* Appcompat?
* OpenCV (obtain libOpenCL.so from device)

Tools

* ADB
* Eclipse load,run,logcat,debug,ddms

App

* Basic Activity

Soapbox

* Eclipse vs. Intellij
* Complexity

# Chapter

Menu, Parameter, and Diagnostic Modes infrastructure

* Preparation for all to come
* Files: Constants, Activity, and Menu.

# Basis Image Processor

Camera, Display and Frame

* Frame (Camera) Listener
* Return passed in frame arg.
* Represents “segment” approach to IP

## Grey Scale

* What about all that color?

## Gaussian Filtering

* Sigma size?

## Canny Edge Detection

* Threshold coefficients

## Dilation

## Contour Detection

## Polygon Detection

# Parallelogram Detection

* Term Rhombus (singular) Rhombi (plural) used in code.
* Steps:
  + Filter out holes (or keep holes)
  + Filter according to minimum size.
  + Filter xxxxxx?
  + Determine angles
  + Filter w.r.t. mean-set
* Typically, not all nine tiles recognized

# Rubik Face Detection

* What we see verse what the camera sees.
* Objective is 2D 3x3 skewed grid.
* Obtain skew angles from Rhombus set
* Obtain center of each Rhombi
* Normalize to rectangular grid
* 2D Sort
* Least Means Square test and acceptance
* Tile color recording

# CPU Time Consumption

# Projections

* AR will be a heavy CPU and GPU consuming field
  + Algorithms are extensive
  + Lots of search and AI
  + Always on