[1. Intro 3](#_Toc411162360)

[2. Enterprise Augmented Reality 3](#_Toc411162361)

[3. AR Rubik Cube App 3](#_Toc411162362)

[4. Ecosystem 3](#_Toc411162363)

[5. Basic Architecture 4](#_Toc411162364)

[6. Controller - Overview 4](#_Toc411162365)

[7. Controller – Edge Detection 4](#_Toc411162366)

[8. Controller – Image Dilation 4](#_Toc411162367)

[9. Controller – Contour Recognition 4](#_Toc411162368)

[10. Controller – Polygon Generation 4](#_Toc411162369)

[11. Controller – Rhombi Generation 4](#_Toc411162370)

[12. Controller – Rubik Face Recognition 5](#_Toc411162371)

[13. Controller – 3D Reconstruction 5](#_Toc411162372)

[14. Controller – Gesture Recognition 5](#_Toc411162373)

[15. Controller – Application State Machine 5](#_Toc411162374)

[16. Model 5](#_Toc411162375)

[17. View 5](#_Toc411162376)

[18. Summary of Rubik Cube App 5](#_Toc411162377)

[19. CPU and Power Consumption 6](#_Toc411162378)

[20. Android (iOS) Ecosystem Pros and Cons 6](#_Toc411162379)

[21. IDE : Eclipse verse Android Studio 6](#_Toc411162380)

[22. Third Party Contributions: OpenXX 6](#_Toc411162381)

[23. OpenGL Comments 6](#_Toc411162382)

[24. OpenCV Comments 7](#_Toc411162383)

[25. Nature of Enterprise AR Apps 7](#_Toc411162384)

[26. Technology Timing 7](#_Toc411162385)

[27. Business Models 7](#_Toc411162386)

[28. Early Adoption Markets 7](#_Toc411162387)

[29. Smart Glasses Hardware Convergence 8](#_Toc411162388)

[30. Software Convergence: AR API 8](#_Toc411162389)

[31. Market Predictions 8](#_Toc411162390)

[32. Predictions 8](#_Toc411162391)

[33. Main Players / Forces 8](#_Toc411162392)

[34. Closing 9](#_Toc411162393)

# 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

* Achieves improved productivity
* Smart Glasses:
  + Hands Free
  + See-Through Rendering
  + Fully Immersed (i.e., Full Graphics Overlay)
* Predict Primary Markets will be Enterprise (Business Needs)
* 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 (Khrnoous Java API)
* OpenGL (Google API wrapper)
* OpenCL (no official java wrapper, JNI to C++ used).
* 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
* Typically, not all nine Rhombi are recognized
* Sometimes, extra Rhombi are provided
* Result is location, size and orientation of grid
* << 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

* Has roots in gaming industry
* Java API:
  + Very close (exact) 1:1 mapping with ‘C’ version
  + More like a ‘C’ interface
  + Not really OO
  + Weak Enums
  + No Comments in docs!
* Very Efficient! Already on GPUs
* Already installed on android devices

# OpenCV Comments

* Experimental / Exploratory strong connections to MatLab
* No overall theory, rather a collection of techniques
* Deep background in clever algorithms
* Java API:
  + Very close (exact) 1:1 mapping with ‘C’ version
  + Matrix class everywhere
  + Not strongly typed
  + Weak Enums
* Not installed on android devices, but not too hard
* Exciting times: possible explosion of interest
* Nature to move to GPUs

# 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
* Always on
* CPU and GPU consuming => Power Consuming

# 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
  + Leverage of low-cost Smart Phone devices
* Power Consumption an Issue :=> 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/Metaio)? Big Layer!
* Additional AR Functionality
* Predict several device vendors to collaborate and foster
* Refinement of OpenCV
* Subset of OpenGL
* Examples:
  + Metaio SDK
  + ARPA SDK
* <http://socialcompare.com/en/comparison/augmented-reality-sdks>

# Market Predictions

* Not as large a Market as Smart Phones
* Hardware will become a commodity
* Enterprise application will be primary markets
* Wide range of secondary markets
* Open Source API?

# Predictions

* 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

# Main Players / Forces

* Smart Glasses Manufactures
* AR SDK Vendors
* Application Developers

# Closing

* Probably will NOT be a demand for apps that run across a wide variety of devices: i.e., industrial uses where device can be specified. Thus, not significant needs for cross-platform SDKs like Unity. Therefore SDKs will not be prominate. Also, early hardware will have uniquiniques.