SEEING STARS

Using technology to deliver an engaging app to capture meteorite sightings, on Android and iOS
ThoughtWorks

Or search App Store or Google Play for “Fireballs in the Sky”
HELLO

DAVID COLLIS
Maths nerd
@davidcolls
AR MATHS

BRAD WARD
Developer
IOS

NATHAN JONES
Developer
@the_nathanjones
ANDROID
FIVE* WHYS

* Actual number may vary
1. WHY NATIVE?

To describe a fireball
Words & numbers fall short, so animated recreations were MVP.

With particle systems
Demanded performance beyond the reach of mobile web for the majority of devices.

Meant 2 native apps
Developed in parallel.
2. WHY AUGMENTED REALITY?

AR not MVP, but delightful
And improved reporting

Option for Release 1

Implemented in Release 2
3. WHY BESPOKE AR?

A unique context
No desire to license technology
Based on sensors not camera image
Camera view just black at night
Very simple interaction
Google Sky only Android
Google Sky won’t subordinate

And we had a Processing prototype
4. WHY PROTOTYPE IN PROCESSING?

Fastest way to start
Dave knew Processing (visualisation IDE)
Rapidly iterate and demonstrate we could do star maps (highest risk)
No dependencies

And finish
Porting together would be low risk
AR MATHS*

* Guaranteed to contain NO equations
Where are the stars?

How do we draw this (in a virtual window)?
Where are the stars?

How do we draw this (in a virtual window)?

Where are you looking?

Where are you standing?

Where in the universe?

Where in the sky?
WHERE IN THE UNIVERSE?

8 mins

4 years

α Centauri
WHERE IN THE UNIVERSE?

8 mins

4 years

circle to scale
WHERE ARE THE STARS?

Infinitely distant

“Fixed Stars”

HYG Database

Celestial sphere
WHERE ARE THE STARS?

How do the stars look from the earth’s surface?

Infinitely distant

“Fixed Stars”

HYG Database

How much has the sphere rotated?

What part of the sphere is directly overhead?

Celestial sphere
WHERE IN THE SKY?

Celestial sphere

Time + Date
Siderial Time + Longitude
LOCAL SIDERIAL TIME
WHERE IN THE SKY?

Celestial sphere

Time + Date

Siderial Time + Longitude

LOCAL SIDERIAL TIME

LATITUDE
WHERE IN THE SKY?

Azimuth
Elevation

Terrestrial observer

Celestial sphere

Time + Date
Siderial Time + Longitude
LOCAL SIDERIAL TIME

LATITUDE
DRAWING IN A VIRTUAL WINDOW
DRAWING IN A VIRTUAL WINDOW

- **Location**
- **Screen**
- **Known positions**
- **Perspective Projection**

Find where the line-of-sight hits the screen.
DRAWING IN A VIRTUAL WINDOW

Choose an eye-screen distance

Location

Known positions

Screen

View direction

Find where the line-of-sight hits the screen

Perspective Projection
WHERE ARE YOU STANDING?

API

GPS satellites

Cell towers

WiFi access points

CLLocationManager

LocationManager
DEFINES LOCAL REFERENCE FRAME

X - North
Y - West
Z - Up

X - East
Y - North
Z - Up
WHERE ARE YOU LOOKING?

...with respect to local reference frame
WHERE ARE YOU LOOKING?

API

SensorManager
  Register for
  updates &
  getOrientation()

WindowManager
  Device default
  orientation

CLLocationManager
  For heading

CMMotionManager
  RefFrameXTrue
  NorthZVertical

...with respect
  to local
  reference
  frame
WHERE ARE YOU LOOKING?

**API**

- **SensorManager**
  - Register for updates & getOrientation()
- **WindowManager**
  - Device default orientation
- **CLLocationManager**
  - For heading
- **CMMotionManager**
  - RefFrameXTrue
  - NorthZVertical

**Device rotation matrix**...

...with respect to local reference frame

**Magnetometers**

**Accelerometers**

**Gyrosopes**
WHERE ARE YOU LOOKING?

**API**

- **SensorManager**
  - Register for updates & getOrientation()
- **WindowManager**
  - Device default orientation

**Device rotation matrix...**

**...with respect to local reference frame**

**Magneto-meters**

**Accelerometers**

**Gyrosopes**

**CLLocationManager**
- For heading
  - CMMotionManager
  - RefFrameXTrue NorthZVertical
WHERE ARE YOU LOOKING?

Azimuth
Elevation
Tilt

Obtained from device rotation matrix

\[
\begin{bmatrix}
ax & bx & cx \\
ay & by & cy \\
az & bz & cz
\end{bmatrix}
\]
**REVIEW**

- **Once per universe***
  - [RA & DECL]
  - (Fixed stars)

- **Once per session**
  - [Azimuth & Elevation]
  - (LST & Lat)

- **Every frame**
  - Device rotation matrix
  - (APIs)

- **Once per session**
  - Device location
  - (APIs)

---

**Where in the universe?**

**Where in the sky?**

**Where are you looking?**

**Where are you standing?**
Sweet, fruity, and objective knowledge
IOS INGREDIENTS

Core Location & CoreMotion

CLLocationManager latitude - longitude

CMMotionManager azimuth - elevation - tilt

- Using CMAAttitudeReferenceFrameXTrueNorthZVertical reference frame

- The API-provided pitch, roll and yaw were not used (pitch and yaw don’t compensate for roll). Used deviceMotion.attitude.rotationMatrix directly instead.

- Reference frame ‘drifts’ over time, periodic resets resolve this.
### IOS INGREDIENTS

**Accelerate/vecLib library** hardware-accelerated vector maths

- VecLib uses the *Advanced SIMD* instruction set implemented by NEON on ARMv7 devices (iPhone 4 and above, iOS 4.3 and above), with fallbacks for older devices
- 2-10x performance bump over standard

**Objective C**

- Avoided overhead of classes/GC in calculation code
- Work well with Accelerate library’s C interface
- Rendering code is Objective-C
CoreGraphics

☐ CPU-based 2d rendering

☐ Minimal development effort with reasonable flexibility

☐ Was an expected (and realised) performance bottleneck
  ☐ OpenGLES would provide dramatically improved performance, at higher development cost.

☐ Final performance was good on iPhone5 devices

*Graceful degradation was added for iPhone4 - stars are rendered as squares rather than circles, star labels are not rendered and the horizon image is thinner.*
So many options to improve performance...

- Optimise use of Accelerate library via bulk calculations
- OpenGLES (e.g., Cocos2D or SpriteKit) for rendering
- Multithreading
- Full GPU implementation of star-positioning calculations
Android

Developing for the bot with the lot
Which API?

`SensorManager` is the home for all sensors in Android

Lots of change in this area of the API

Reference examples use deprecated `ORIENTATION_SENSOR`

Hand-rolled sensor fusion of accelerometer and compass

Take a look at the `ROTATION_VECTOR` Sensor

Adjust resulting vector for current and default orientation
How do I draw them?

We use a regular `SurfaceView`.

We use a `Timer` targeting 60FPS instead of an explicit thread.

Draw on a regular 2D `Canvas`.

Not hardware accelerated.

Code split into set of renderers.

Toggled FPS renderer for performance testing.

Room for improvement.
Coding Style

Optimised vector math libraries not as mature

Embrace some functional paradigms, separate state and behaviour

Multi-threading is an option

Beware the garbage collector

Profile all the things, these are limited resources

Many ways to skin a cat with vastly different performance
Fragmentation exists
Expect it and deal with it
Eligible for installation on 3606 devices
Pick a baseline and work out what you are in for
Don’t expect the API to be consistent

Get some real devices
Lowest and highest target OS versions
Lowest and highest screen sizes - resolution and physical size
Lowest performance - slow single-core phones
Fall back to emulator only for sanity check on look and feel
BUILDING APPS
On both Android and iOS
Use real devices

Performance differs from emulators

Sensor data not available in emulators

Test in the real world

The acid test for an AR app
Think about usage
No wifi or cell towers in the outback
Don’t block the user
While you find their location
Enough is enough
Know the required accuracy, and stop when you have it
Stop wasting their battery (Android)
Turn off location services on app hide
Differences in APIs
Android vs iOS

Frame of reference
True North or Magnetic North?
(Does it matter in Perth?)

Smoothing
How to filter noise while preserving a responsive signal
THANK YOU

For questions or suggestions:

dcolls@thoughtworks.com
njones@thoughtworks.com
brward@thoughtworks.com