Android and VR

from Cardboard to Tango and beyond

AnDevCon
The Android Developer Conference

google.com/+DarioLaverde
HTC Sr Developer Evangelist

Mixed Reality

aka Reality-Virtuality Continuum *

* Paul Milgram, 1994

http://etclab.mie.utoronto.ca/publication/1994/Milgram_Takemura_SPIE1994.pdf

it’s all about presence

(and not feeling nausea)

- at least 90 Hz (fps) per screen
- at least 1200x1080 (per eye)
- accurate controllers (see and use your “hands” + more)
- ability to track your location (ideally 360° room scale VR)

GIZMODO

HTC Vive: Virtual Reality That's So Damn Real I Can't Even Handle It

Carlos Rebato

Filed to: VIRTUAL REALITY 3/04/15 1:00pm

149,884 36 ★

VIDEO

Hands-on: HTC's Vive headset made this virtual reality noob a virtual reality believer

By Florence Ion | Follow
Greenbot | Mar 4, 2015 10:12 AM PT

RELATED TOPICS

"Wow!"

More like this


Reality-checking HTC Vive and SteamVR: 4 crucial questions for the virtual...


Move over, Oculus: HTC announces SteamVR-powered Vive VR headset


HTC One M9 review: A disappointingly good phone

the three tiers

- PC - HTC Vive, Oculus
- Console - Morpheus
- Mobile - Cardboard, Gear VR

note: Google describes Cardboard as a “snack-able experience”

Designing for VR

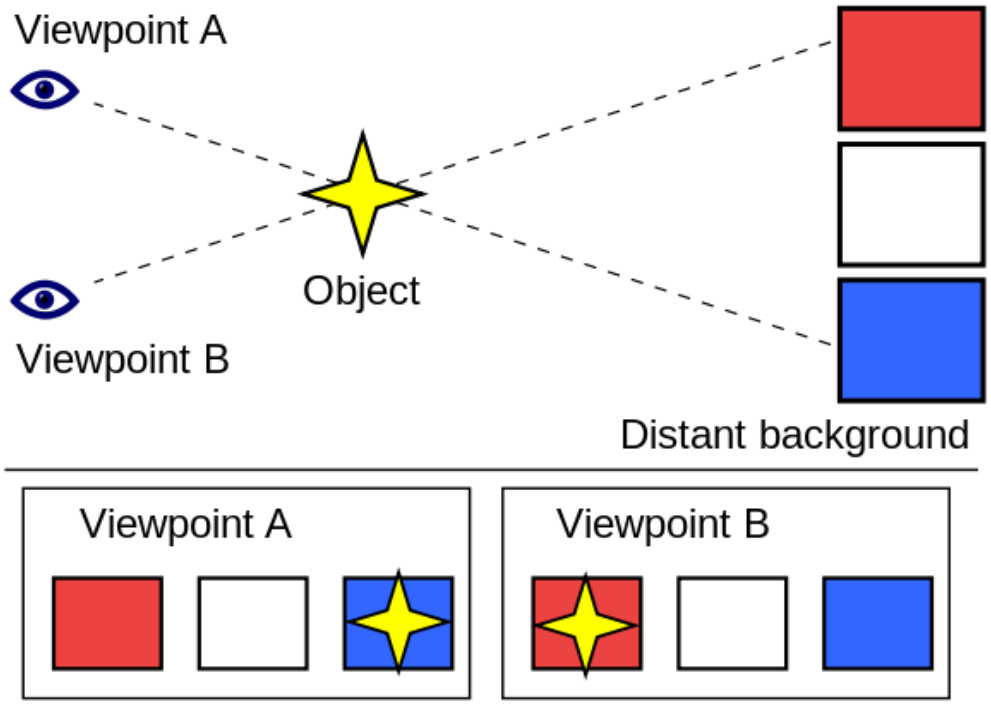
Rules for maintaining presence

Maintain framerate!

ideally 90 fps minimum

60 on mobile (can still make you ill)

some basics first: Stereoscopy



What’s IPD?

IPD = interpupillary distance

Ideally you should be able to adjust it to your specific IPD – otherwise default to a avg (mean) number

What’s IPD?

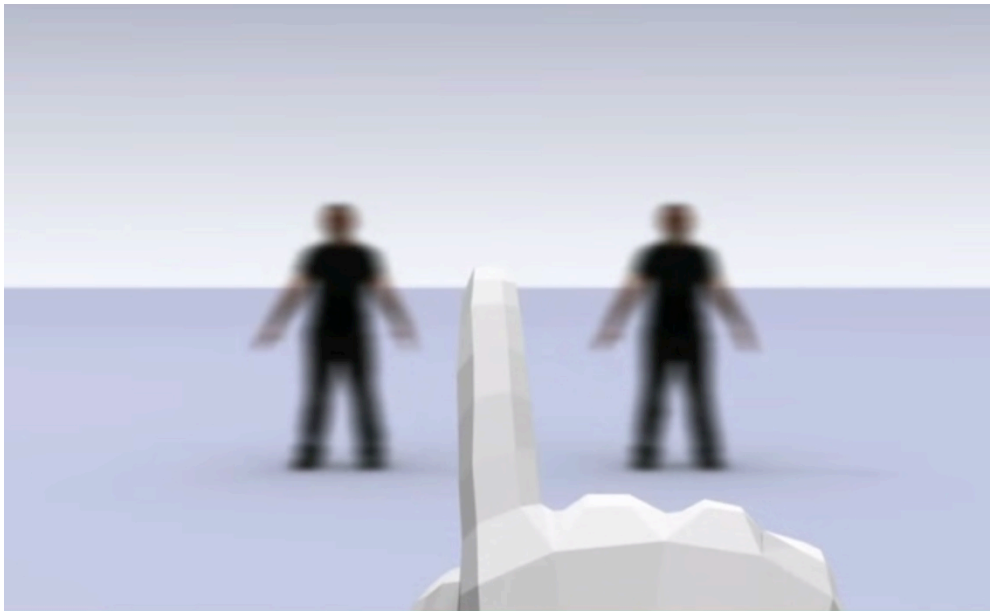
Mean IPD is an important and oft-quoted measure in stereoscopic work. However, there is startlingly little agreement on what it should be. Mean IPD has been quoted in the stereoscopic literature as being anything from 58 mm to 70 mm. It is known to vary with respect to age, gender and race.

What's *Your* IPD?

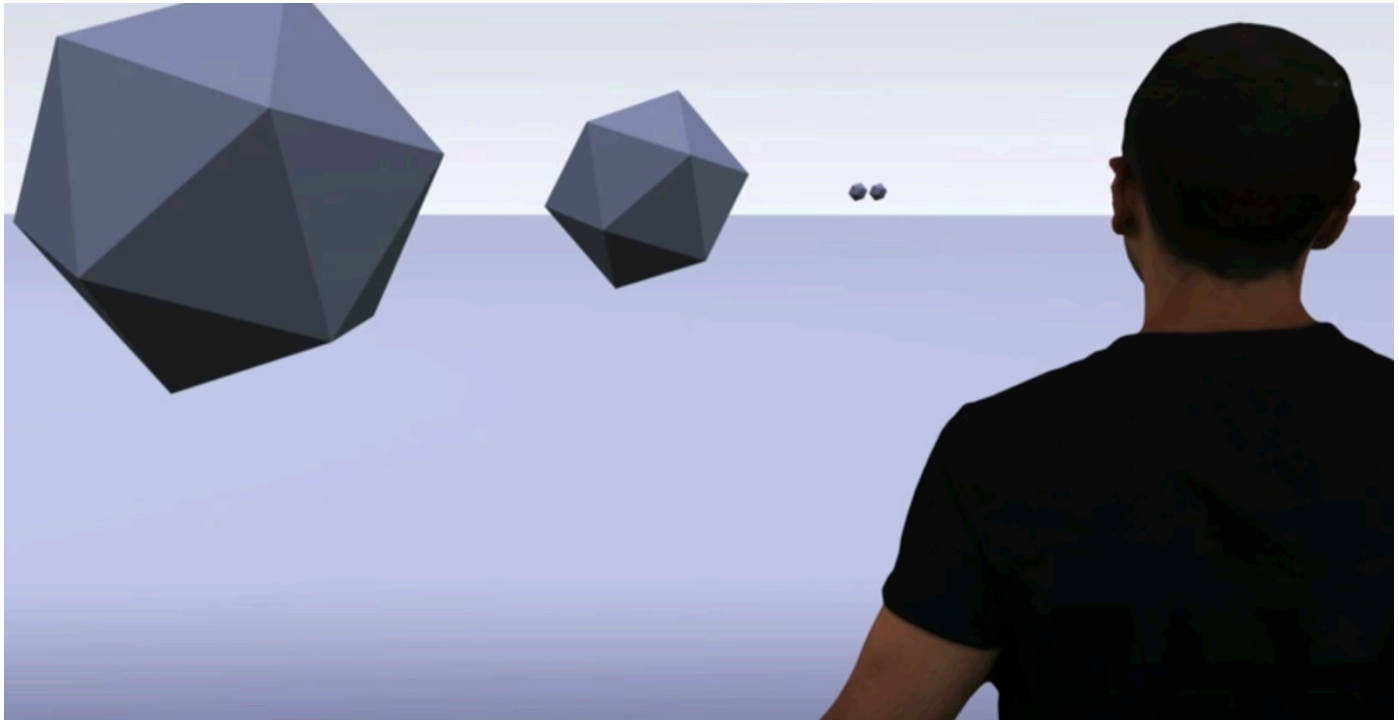
Use a IPD tool to measure the distance between your pupils:

https://pd.warbyparker.com

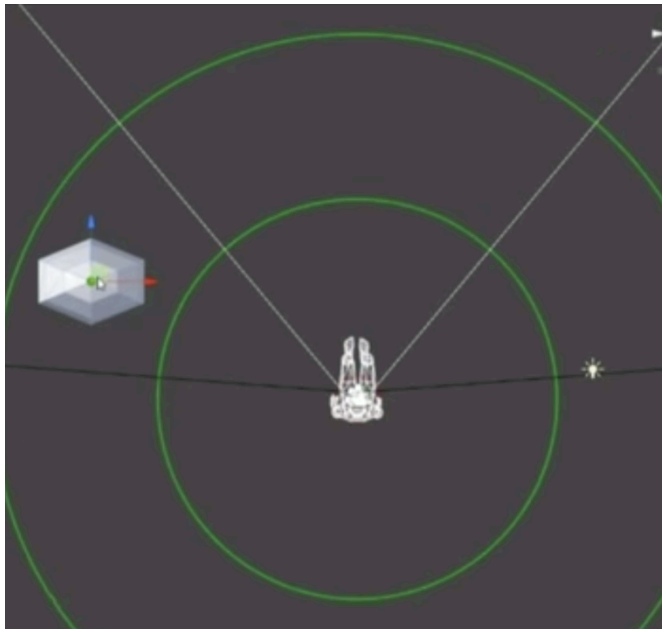
Convergence and Focus

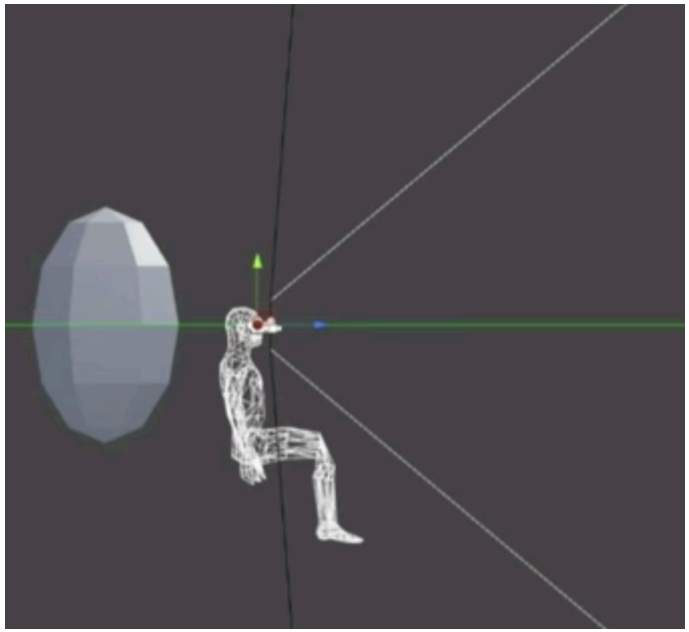


Convergence and Focus

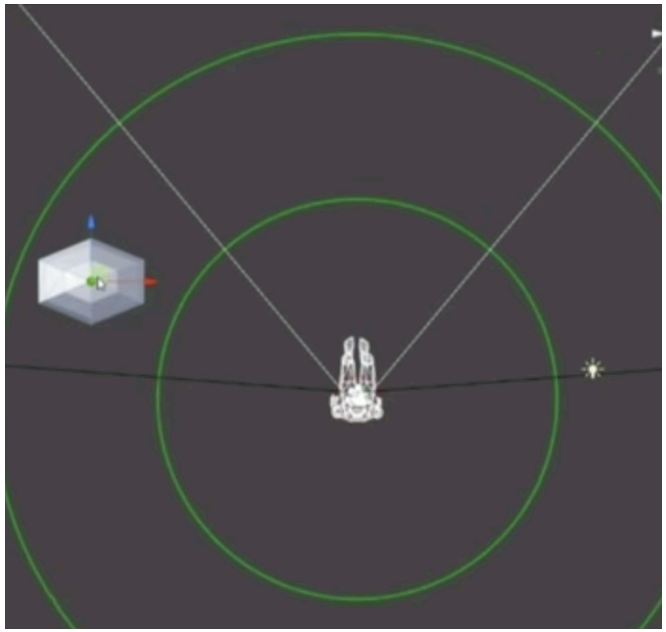


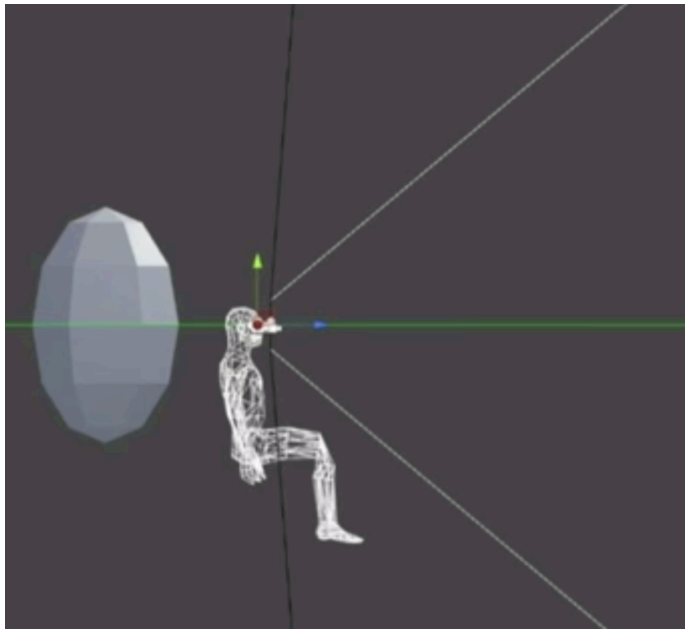
Field of View (FOV)





Field of View (FOV)

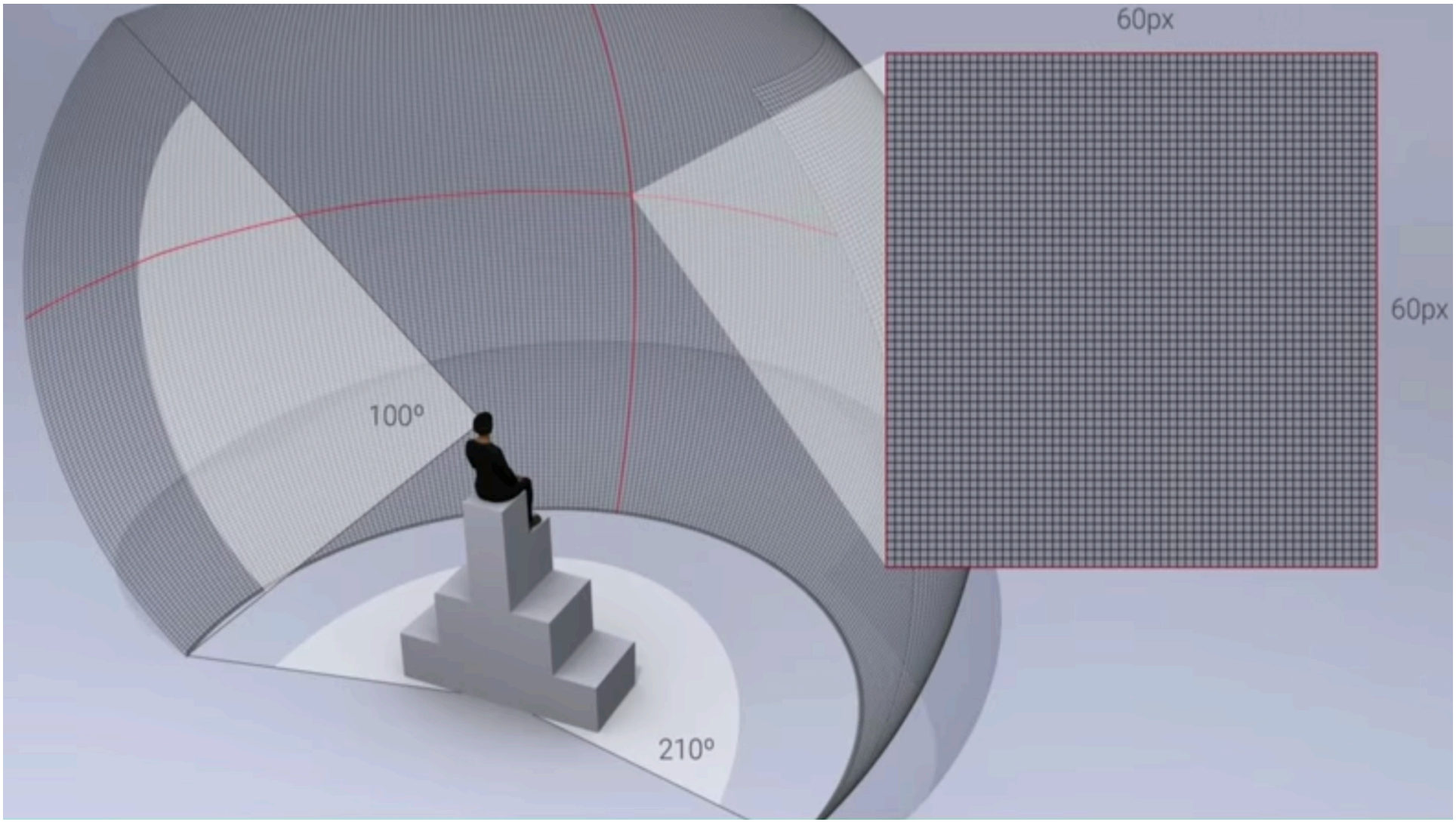




Ideally: 210 x 100

So what’s the minimum res?

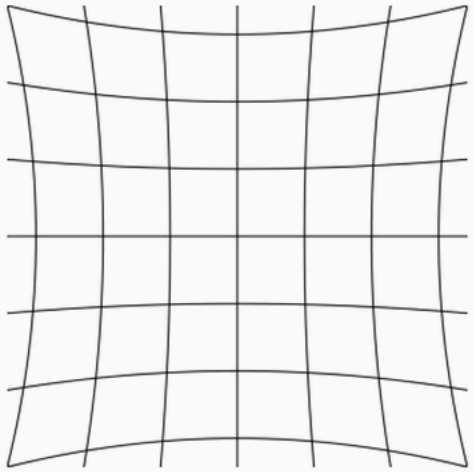
As Google suggests, think of it in terms of degrees of FOV and pixels per angle and not screen resolution.


60px
60px
100°
210°

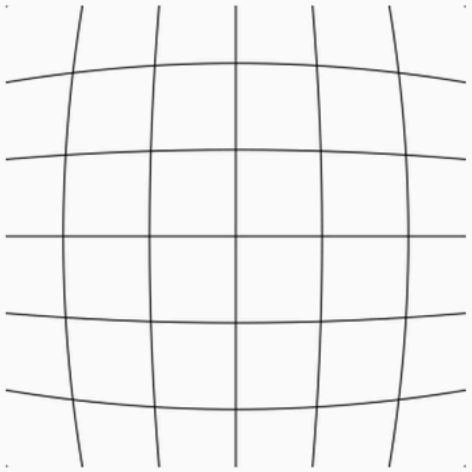
So what’s the minimum res?

that’s 12600 x 6000

Lens distortion



Pincushion Distortion



Barrel Distortion

Designing for VR

Avoiding simulator sickness

motion sickness is a defensive mechanism

- so again don’t drop frames!

fade to black if you can’t - use audio cues

Designing for VR

Don’t do anything disconcerting:

e.g. start in the middle of a desk

hyperspace with visual effects (fade first)

Keep a stable horizon!

Try to avoid inducing phobias (unless treating them):
acrophobia, agoraphobia, claustrophobia

Designing for VR

- Use fixed velocity and grounding with objects
- avoid sudden brightness changes
- use the world for indicators – debug logging too

So consider physiological and environment factors and “be intentional”

Designing for VR

- the power of teleportation

- impossible experience “cardboard” falling, flying

When adopting 2D to 3D:

e.g. real-estate app: adds sense and scale and layout - instead of photo gallery and need to physically travel to locations

Designing for VR

leverage whole canvas use hints:
rotation on scene change (fade to black!)
spatial audio cues
gaze cue (use a reticle!)

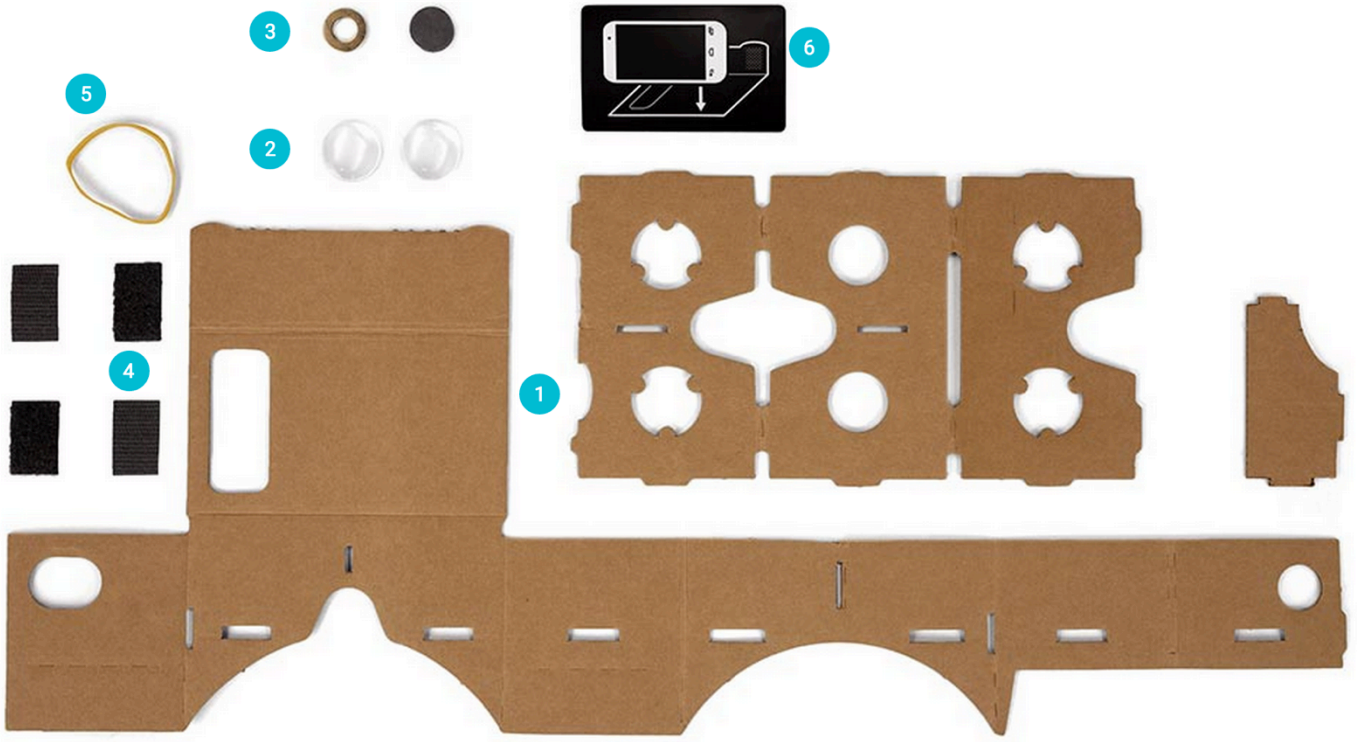
Designing for VR

using your mobile phone instead

(and of course using Android instead)

so let’s start talk about Google’s Cardboard

DIY VR



or buy a viewer

DODOcase

Ready-made Cardboard viewer kits with an alternative capacitive switch that works with most phones.

BUY IT

I Am Cardboard

Pick from two sizes and five colors to fit your taste and phone. Check out their EVA foam Cardboard.

BUY IT

Knox Labs

Offers standard Cardboard, and for something a little different, an Aluminum Cardboard viewer.

BUY IT

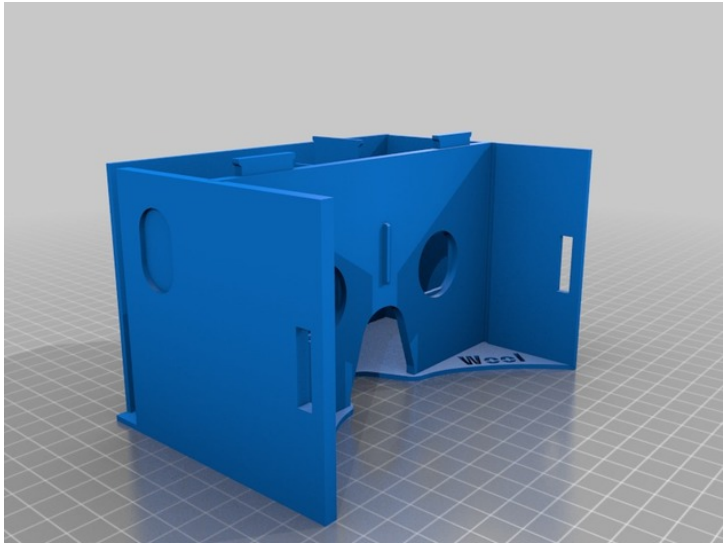
Unofficial Cardboard

Buy fully assembled Cardboard, or just the parts to build your own. Two sizes to fit most phones.

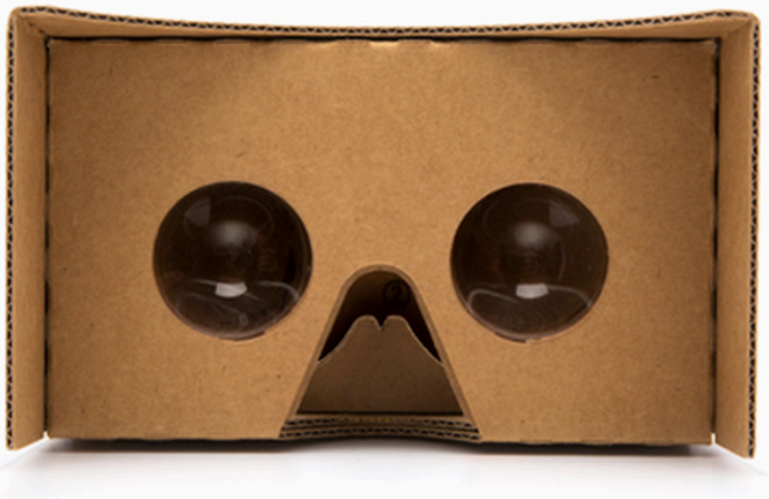
BUY IT

or 3d print it

http://www.yeggi.com/q/google+cardboard/



Cardboard V2



THE NEW CARDBOARD

The new Cardboard was unveiled at Google I/O 2015. It supports larger phones with screens up to 6 inches. It has a new button that works with any phone. And it assembles (and disassembles) in just 3 steps. Like the first Cardboard, it's still about VR for everyone.

Designing for Cardboard

Design Lab VR app:

https://play.google.com/store/apps/details?id=com.google.vr.cardboard.apps.designlab

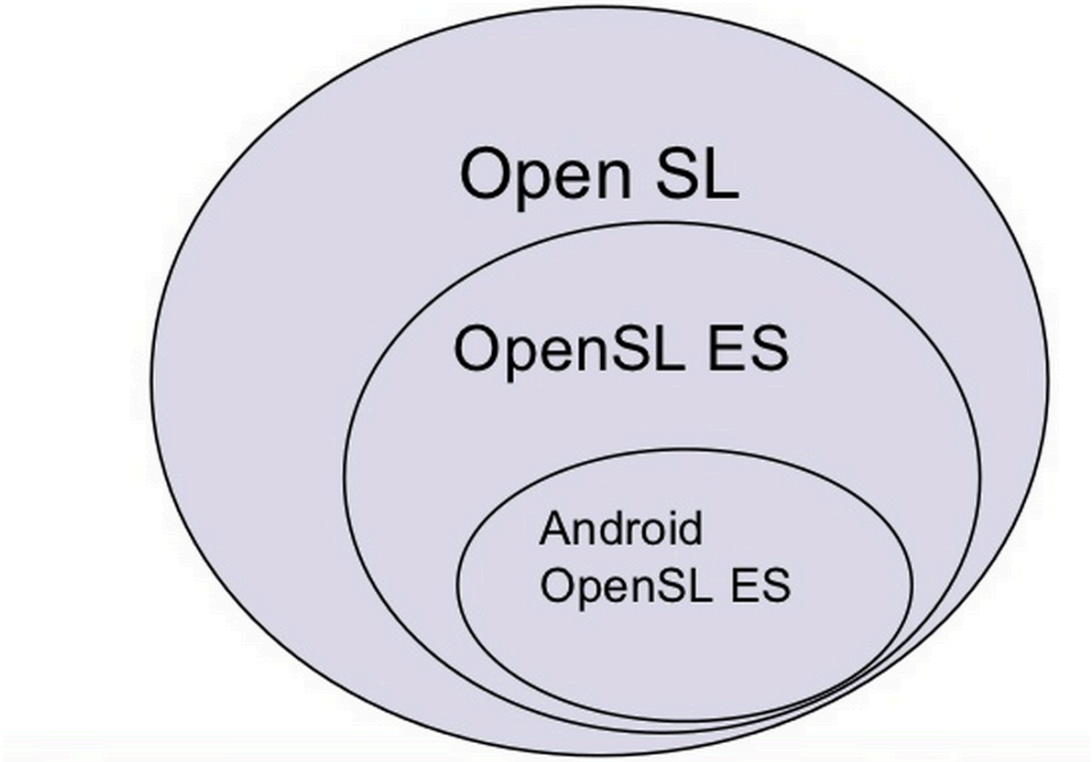
VR Design Principles Covered

1. Using a Reticle
2. UI Depth & Eye Strain
3. Using Constant Velocity
4. Keeping the User Grounded
5. Maintaining Head Tracking
6. Guiding with Light
7. Leveraging Scale
8. Spatial Audio
9. Gaze Cues
10. Make it Beautiful

Google’s Cardboard Best Practices

- icon silhouette (material design)
- use “Cardboard” instead of Virtual Reality
- “get cardboard” if they don’t have it

3D positional audio



What Cardboard SDK provides

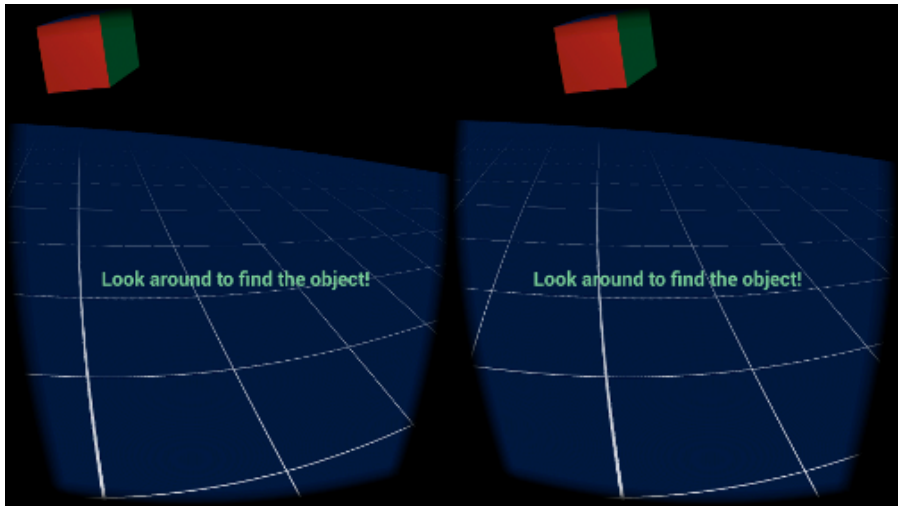
- Lens distortion correction.
- Head tracking.
- 3D calibration.
- Side-by-side rendering.
- Stereo geometry configuration.
- User input event handling.

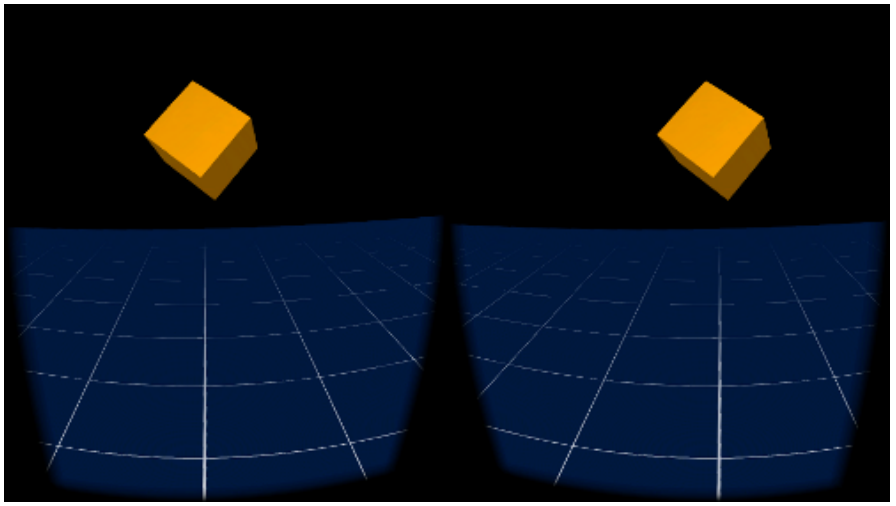
CardboardView.StereoRenderer

Public Methods

abstract void	onDrawEye(Eye eye)
abstract void	onFinishFrame(Viewport viewport)
abstract void	onNewFrame(HeadTransform headTransform)
abstract void	onRendererShutdown()
abstract void	onSurfaceChanged(int width, int height)
abstract void	onSurfaceCreated(EGLConfig config)

Cardboard SDK Sample





```java
/**
 * Draws a frame for an eye.
 *
 * @param eye The eye to render. Includes all required transformations.
 */
@Override
public void onDrawEye(Eye eye) {
    GLES20.glClear(GLES20.GL_COLOR_BUFFER_BIT | GLES20.GL_DEPTH_BUFFER_BIT);

    checkGLError("mColorParam");

    // Apply the eye transformation to the camera.
    Matrix.multiplyMM(mView, 0, eye.getEyeView(), 0, mCamera, 0);

    // Set the position of the light
    Matrix.multiplyMV(mLightPosInEyeSpace, 0, mView, 0, LIGHT_POS_IN_WORLD_SPACE, 0);

    // Build the ModelView and ModelViewProjection matrices
    // for calculating cube position and light.
    float[] perspective = eye.getPerspective(Z_NEAR, Z_FAR);
    Matrix.multiplyMM(mModelView, 0, mView, 0, mModelCube, 0);
    Matrix.multiplyMM(mModelViewProjection, 0, perspective, 0, mModelView, 0);
    drawCube();

    // Set mModelView for the floor, so we draw floor in the correct location
    Matrix.multiplyMM(mModelView, 0, mView, 0, mModelFloor, 0);
    Matrix.multiplyMM(mModelViewProjection, 0, perspective, 0,
        mModelView, 0);
    drawFloor();
}
```

Cardboard SDK checklist

- minSdkVersion="16"
- <uses-feature android:glEsVersion="0x00020000" android:required="true" />
- intent filter: com.google.intent.category.CARDBOARD
- Extend CardboardActivity
- Implement CardboardView.StereoRenderer
- Use CardboxView (which extends GLSurfaceView)

Cardboard SDK setup

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.common_ui);
    CardboardView cardboardView = (CardboardView) findViewById(R.id.cardboard_view);
    // Associate a CardboardView.StereoRenderer with cardboardView.
    cardboardView.setRenderer(this);
    // Associate the cardboardView with this activity.
    setCardboardView(cardboardView);
```

transforms

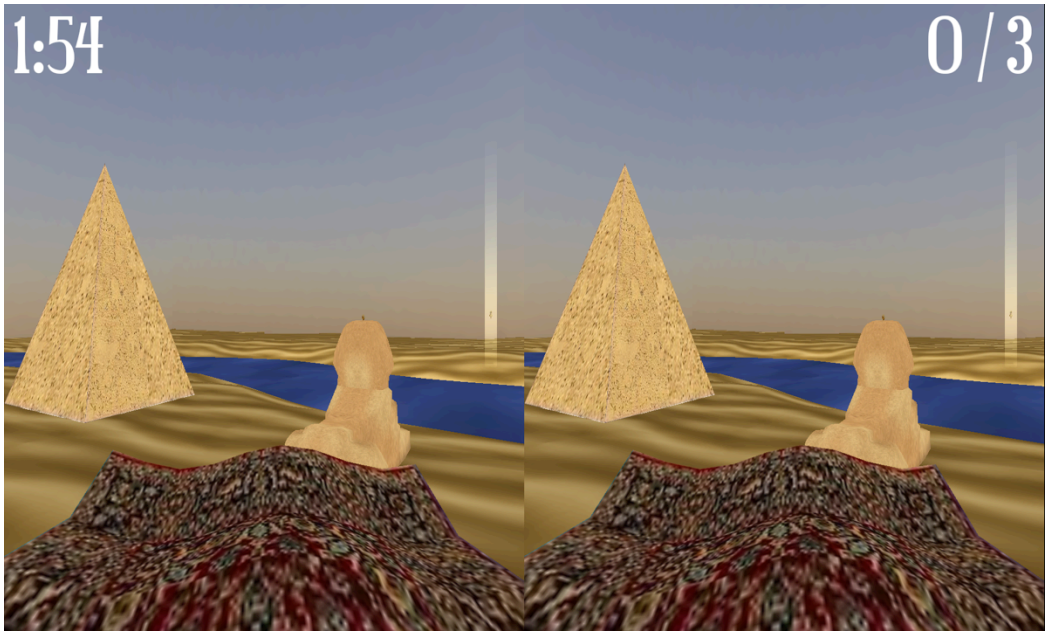
```
@Override
public void onNewFrame(HeadTransform headTransform) {
...
   headTransform.getHeadView(mHeadView, 0);
...
}
```

inputs

```
@Override
public void onCardboardTrigger() {
    if (isLookingAtObject()) {
        mScore++;
        mOverlayView.show3DToast("Found it! Look around for another one.\nScore = " + mScore);
        ...
    } else {
        mOverlayView.show3DToast("Look around to find the object!");
    }
    // Always give user feedback
    mVibrator.vibrate(50);
}
```

porting existing games

case study: open source game “Seven Wonders”



http://github.com/skylight1

porting existing games to use Cardboard SDK

- Provides 2nd camera view (adjustable distance between eyes)
- Provides 3D Text
- Handles input (e.g. d-pad)
- reconsider positional/acceleration based navigation using main camera view i.e. consider VR head tracking

first attempt

- add stereoscopy
  - i.e. add second camera using your IPD

Very quick prototype but not accounting for:

- lenses
- adjustable IPD
- UI tools for text, toasts etc.
- head tracking

Project Tango

Durovis’s Dive Headset

Using Project Tango for VR

https://youtu.be/tPR9EFE20Aw?t=400

Project Tango helps you let you know where you are:

- indoor navigation - Google using it for visually impaired project
- robotics, AR, VR - 3d tracking capabilities

Project Tango Components

1 Motion Tracking - like computer mouse… relative motion - but in 3D

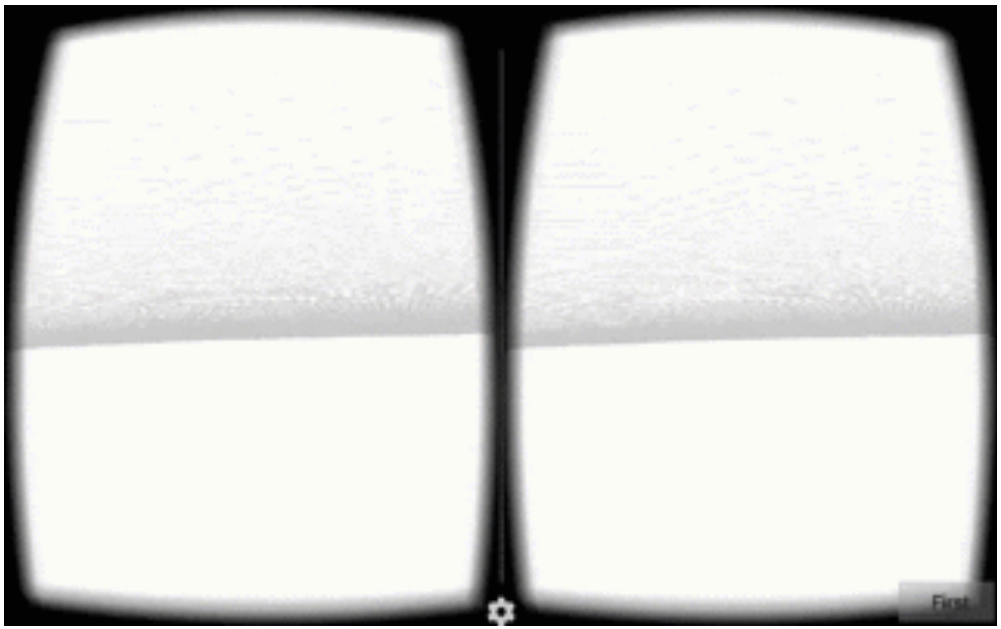
2 Area Learning - can localize to previously seen maps

3 Depth Perception - infrared signals to measure floor, wall, furniture

new reference design

Announced partnership with Qualcomm:

demos



Unity3D / Unreal3D vs Java

Cardboard, Tango include samples for both

Caveat: trend points to supporting via Unity (mostly) vs Android/Java code

New to OpenGL? Time to rethink tools….

Adding Additional Inputs

- If in Android, just use standard APIs BLE, BT, Wifi
- If in Unity, write a plugin to wrap the APIs (not that difficult to wrap Android see sample code)
- For both watch for:
  - Mapping coordinate systems
  - Rotation calculations – may require quaternions!

Quaternions vs Euler Angles

To address the problem of angular movement in 3D space (when using controllers etc) you may need to use Quaternions (an extension of the complex number. Instead of just i, we have three numbers that are all square roots of -1, denoted by i, j, and k.) to avoid Gimbal lock (as rotations in the Euler representation are done with respect to the global axis, a rotation in one axis could 'override' a rotation in another, making you lose a degree of freedom)

required reading:

http://www.gamedev.net/page/resources/_/technical/math-and-physics/quaternion-powers-r1095

OpenVR, OSVR

Open resources for VR development:
(how to port that Java/Android 3D engine)
With Valve's OpenVR and Razer's OSVR

https://github.com/ValveSoftware/openvr
http://www.razerzone.com/osvr

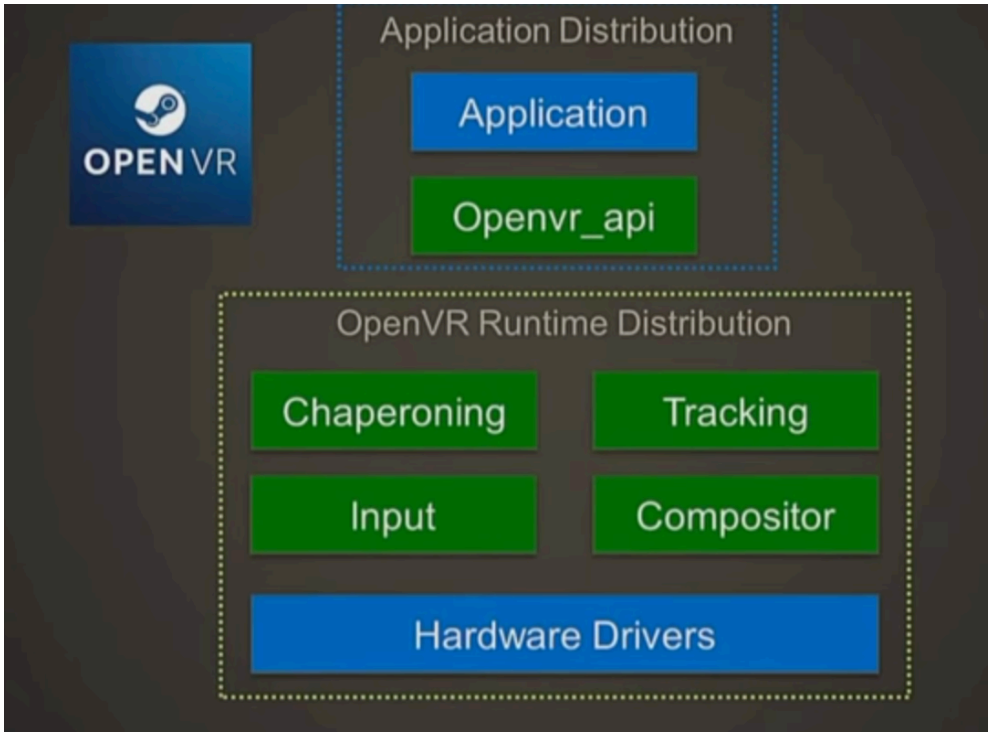
OSVR

Full open source stack:

- Software and HW!

- Targets most headsets
  (including theirs of course)

OpenVR

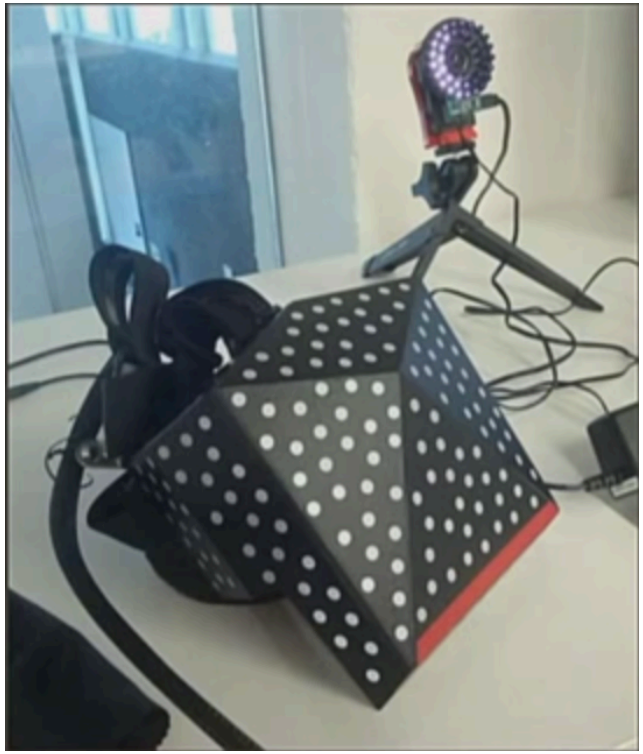


- any headset, includes support for HTC Vive, Oculus, Lighthouse tracking system, open to 3rd party drivers

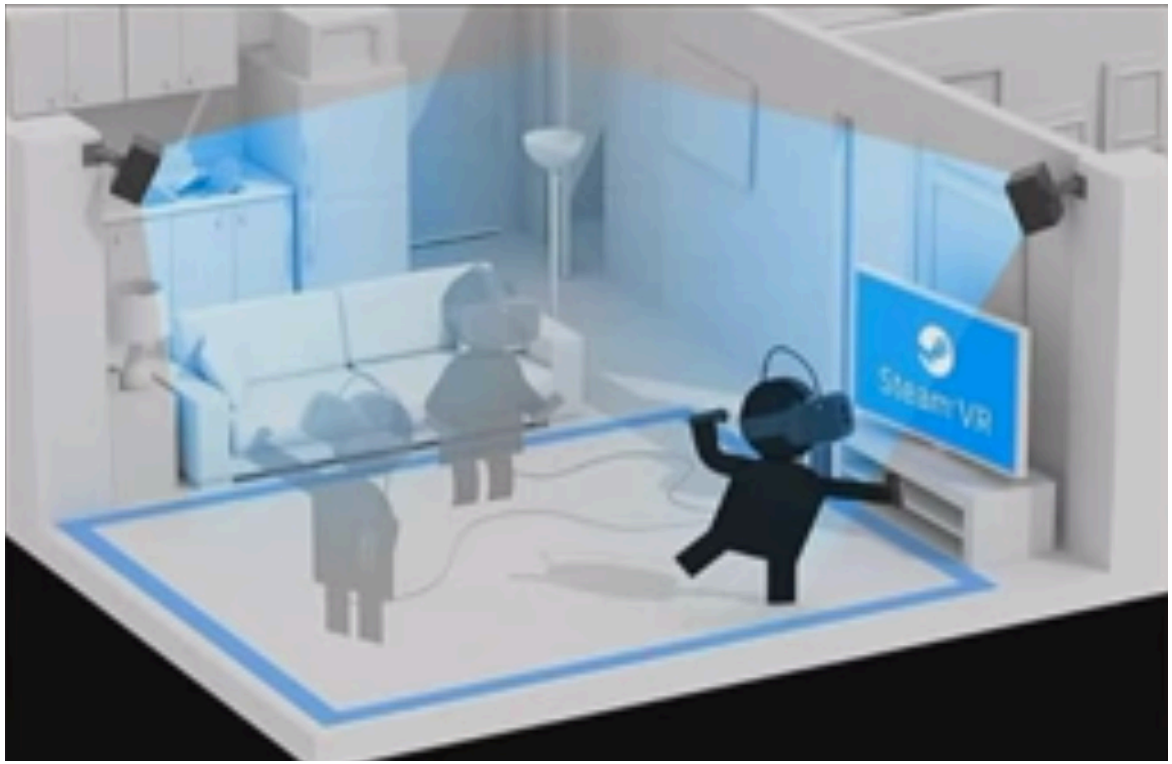
- OSVR announced support via OpenVR

Lighthouse tracking system

previous attempts:



Lighthouse tracking system



Jump

360 “VR” Video

Currently not immersive enough

- waiting for positional tracking capability which would require real time CGI
- For most 360 video, seating/standing experience ok
  - should be stereoscopic (or at least use depth mapping to imitate it)

additional VR resources

Senses Override (Cardboard) - Shanee Nishry, Developer Advocate, Google
https://www.parleys.com/tutorial/5471cbd1e4b0e15e672384bf

Advanced VR Rendering - Alex Vlachos, Valve Software
http://media.steampowered.com/apps/valve/2015/Alex_Vlachos_Advanced_VR_Rendering_GDC2015.pdf

Google I/O 2015 : Designing for Virtual Reality
https://www.youtube.com/watch?v=Qwh1LBzz3AU

Featured Cardboard apps at the Playstore
http://g.co/cardboardapps

Tango VR with Cardboard SDK:
https://medium.com/@dariony/developing-vr-for-tango-with-java-7cd1d252a7ea

Q & A

Today at the Exhibit Hall: 2:45 pm - 3:15 pm

Office Hours:
Ask Anything! with Dario Laverde

contact: http://google.com/+DarioLaverde