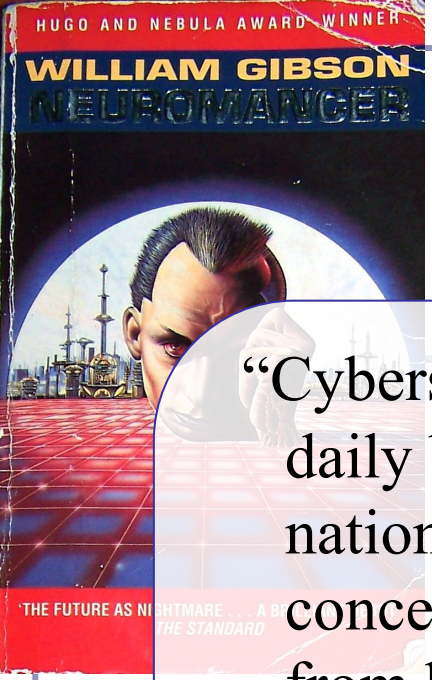
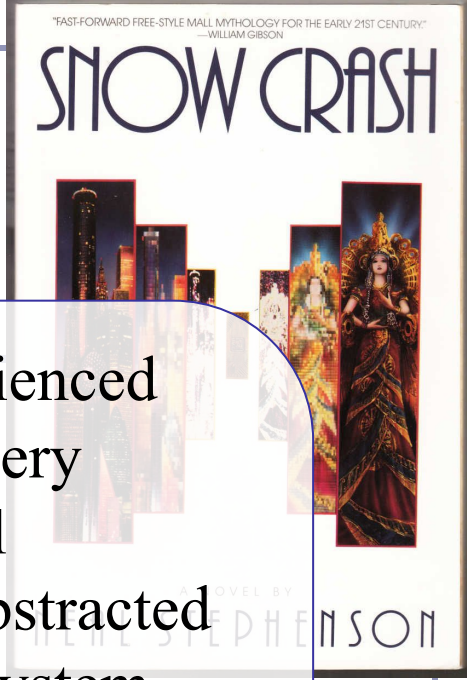
Virtual Reality

Immersion and Presence
in digital realities

Alex Benton, University of Cambridge – alex@bentonian.com

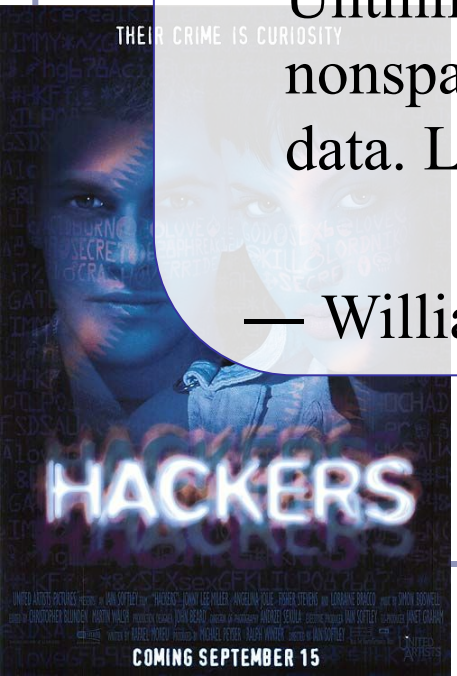
Supported in part by Google UK, Ltd





"Cyberspace. A consensual hallucination experienced daily by billions of legitimate operators, in every nation, by children being taught mathematical concepts... A graphic representation of data abstracted from banks of every computer in the human system. Unthinkable complexity. Lines of light ranged in the nonspace of the mind, clusters and constellations of data. Like city lights, receding..."

— William Gibson, Neuromancer (1984)



~~*What is … the Matrix?*~~

What is Virtual Reality?

Immersion is the art and technology of surrounding the user with a virtual context, such that there's world above, below, and all around them.

Presence is the visceral reaction to a convincing immersion experience. It's when immersion is so good that the body reacts instinctively to the virtual world as though it's the real one.

When you turn your head to look up at the attacking enemy bombers, that's immersion; when you can't stop yourself from ducking as they roar by overhead, that's presence.

Top: HTC Vive (Image creduit: *Business Insider*)
Middle: *The Matrix* (1999)
Bottom: Google Daydream View (2016)

The "Sword of Damocles" (1968)

In 1968, Harvard Professor Ivan Sutherland, working with his student Bob Sproull, invented the world's first *head-mounted display*, or *HMD*.

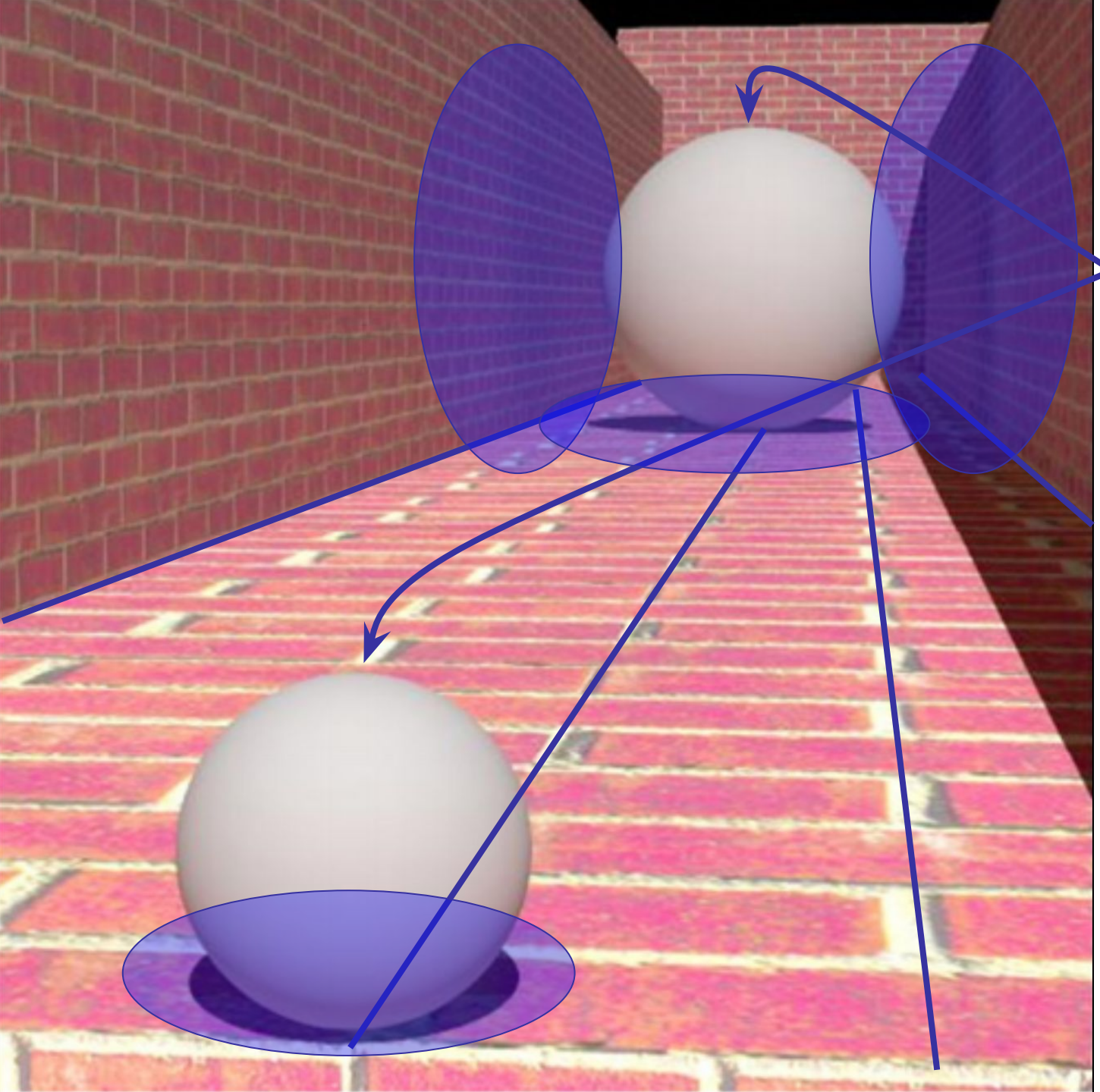
> *"The right way to think about computer graphics is that the screen is a window through which one looks into a virtual world. And the challenge is to makes the world look real, sound real, feel real and interact realistically."*
>
> *-Ivan Sutherland (1965)*

Distance and Vision

Our eyes and brain compute *depth cues* from many different signals:

- **Binocular vision ("stereopsis")**
  The brain merges two images into one with depth
  - Ocular convergence
  - Shadow stereopsis
- **Perspective**
  Distant things are smaller
- **Parallax motion** and **occlusion**
  Things moving relative to each other, or in front of each other, convey depth
- **Texture**, **lighting** and **shading**
  We see less detail far away; shade shows shape; distant objects are fainter
- **Relative size and position** and **connection to the ground**
  If we know an object's size we can derive distance, or the reverse; if an object is grounded, perspective on the ground anchors the object's distance

Image: Pere Borrell del Caso's *Escapando la Critica* ("*Escaping Criticism*") (1874)



Perspective

Occlusion

Shadows

Ambient shadows

Image credit: Scott Murray

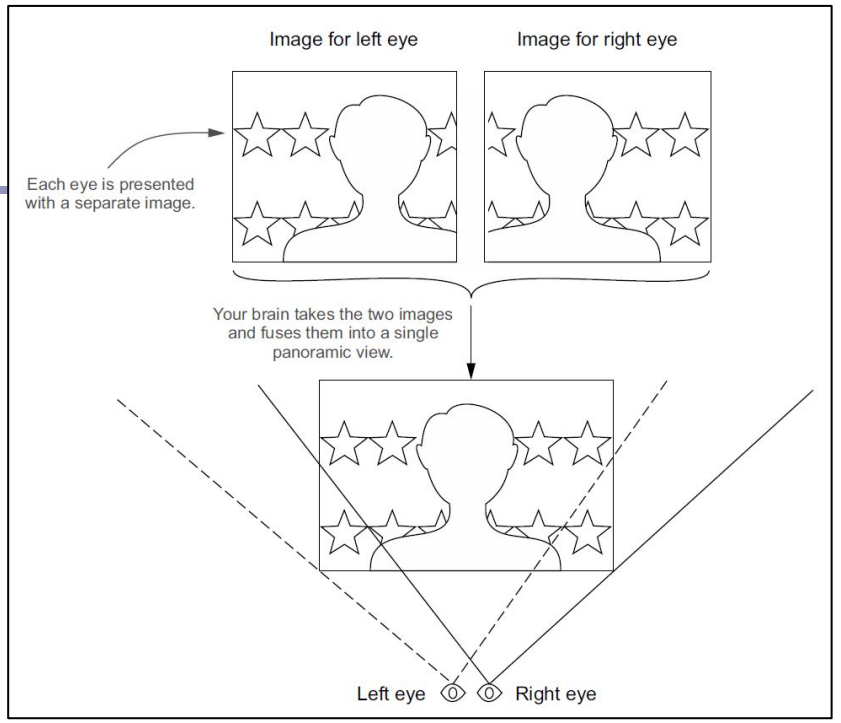
Murray, Boyaci, Kersten, *The representation of perceived angular size in human primary visual cortex*, Nature Neuroscience (2006)

Binocular display

Today's VR headsets work by presenting similar, but different, views to each eye

Each eye sees an image of the virtual scene from that eye's point of view in VR

This can be accomplished by rendering two views to one screen (Playstation VR, Google Daydream) or two dedicated displays (Oculus Rift, HTC Vive)



Top: Davis, Bryla, Benton, *Oculus Rift in Action* (2014)
Bottom: Oculus DK1 demo scene "Tuscanny"

Teardown of an Oculus Rift CV1

Teardown of an Oculus Rift CV1 showing details of lenses and displays
https://www.ifixit.com/Teardown/Oculus+Rift+CV1+Teardown/60612

Accounting for lens effects

- Lenses bend light: the lenses in the VR headset warp the image on the screen, creating a *pincushion distortion*.
- This is countered by first introducing a *barrel distortion* in the GPU shader used to render the image.
- The barrel-distorted image stretches back to full size when it's seen through the headset lenses.

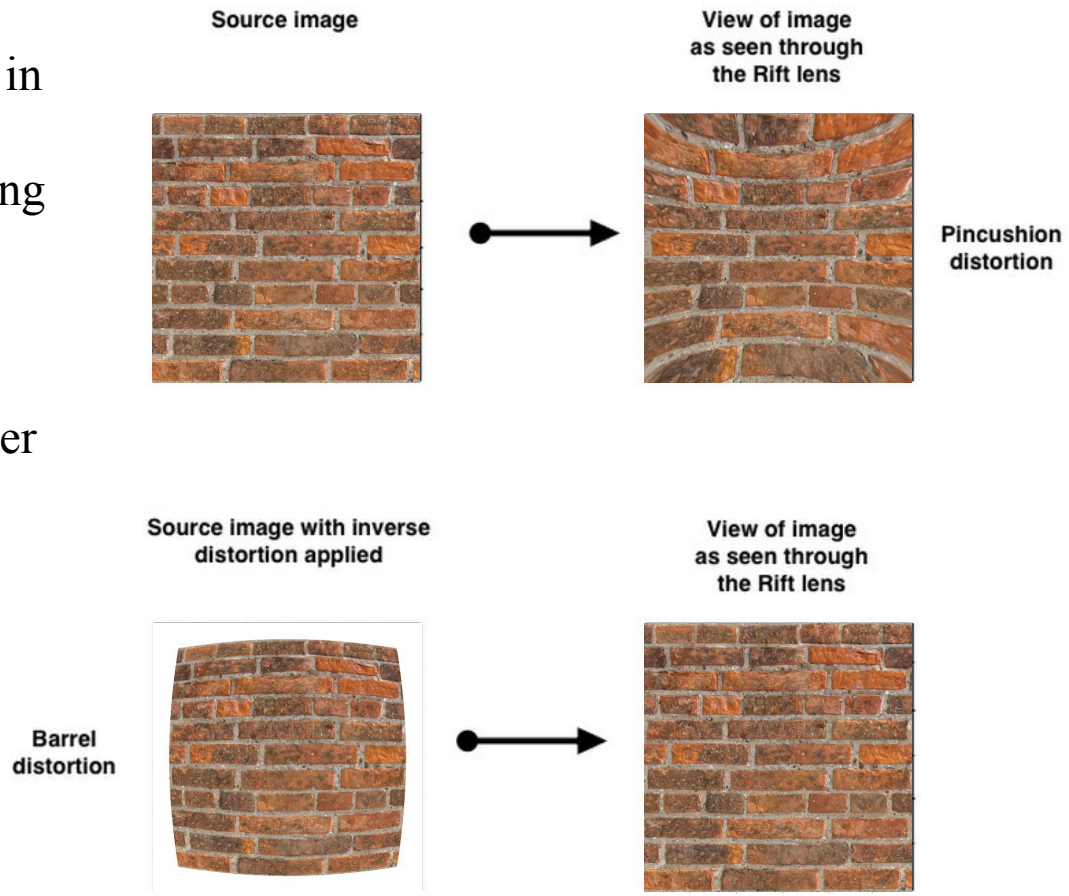
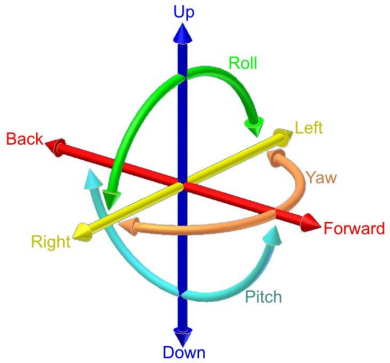


Image credit: Davis, Bryla, Benton, *Oculus Rift in Action* (2014)

Sensors

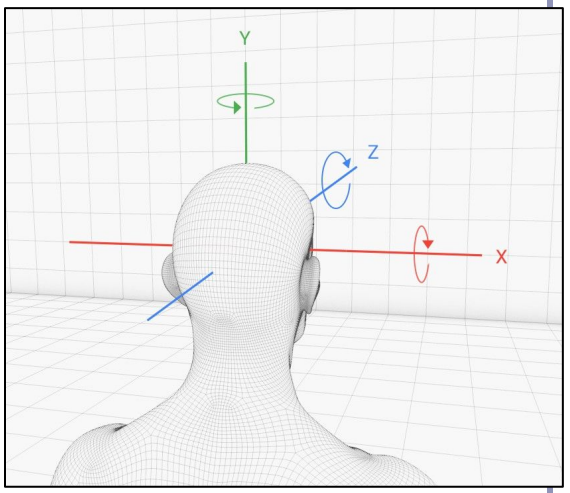
Accelerometer and electromagnetic sensors in the headset trac the user's *orientation* and *acceleration*. VR software converts these values to a basis which transforms the scene.



Ex: WebVR API:

```
interface VRPose {
  readonly attribute Float32Array? position;
  readonly attribute Float32Array? linearVelocity;
  readonly attribute Float32Array? linearAcceleration;

  readonly attribute Float32Array? orientation;
  readonly attribute Float32Array? angularVelocity;
  readonly attribute Float32Array? angularAcceleration;
};
```



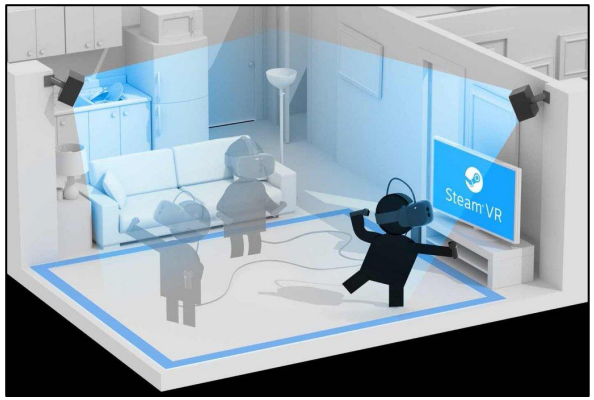
Top: 6DoF (6 degrees of freedom) - Wikipedia
Bottom: Roll (Z), Pitch (X) and Yaw (Y) - Google Design

Sensor fusion

Problem: Even the best accelerometer can't detect all motion. Over a few seconds, position will drift.

Solution: Advanced headsets also track position with separate hardware on the user's desk or walls.

- Oculus Rift: "Constellation", a desk-based IR camera, tracks a pattern of IR LEDs on the headset
- HTC Vive: "base station" units track user in room
- Playstation VR: LEDs captured by PS camera



The goal is to respond in a handful of milliseconds to any change in the user's position or orientation, to preserve presence.

Top: Constellation through an IR-enabled camera (image credit: ifixit.com)
Bottom: HTC Vive room setup showing two base stations (image credit: HTC)

Sensors - how fast is fast?

- To preserve presence, the rendered image must respond to changes in head pose faster than the user can perceive
- That's believed to be about 20ms, so no HMD can have a framerate below 50hz
- Most headset display hardware has a higher framerate
  - The Rift CV1 is locked at 90hz
  - Rift software **must** exceed that framerate
  - Failure to do so causes 'judder' as frames are lost
  - Judder leads to nausea, nausea leads to hate, hate leads to the dark side

Dealing with latency: sensor prediction

A key immersion improvement is to *predict the future basis*. This allows software to optimize rendering.

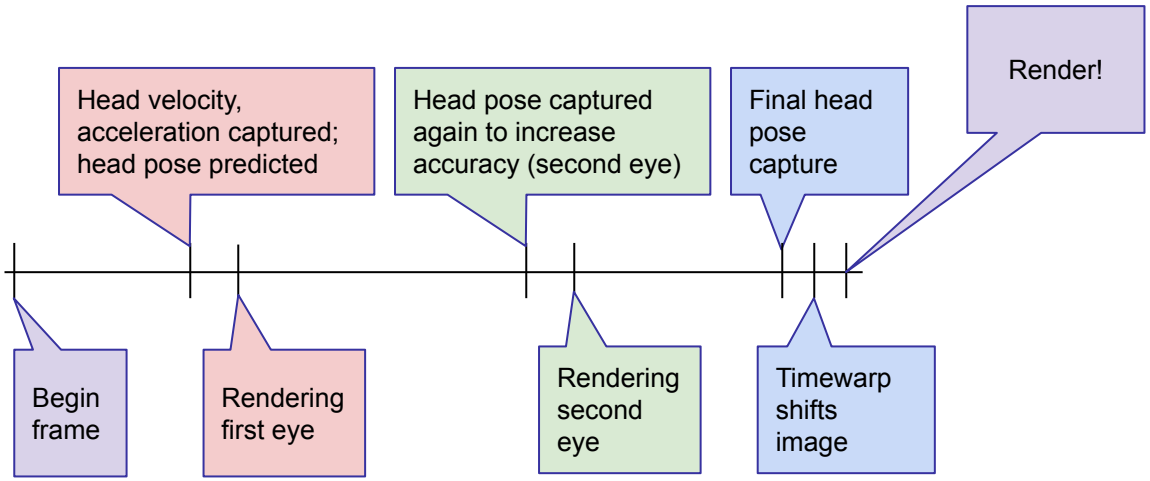
- At time t, head pos = X, head velocity = V, head acceleration = A
- Human heads do not accelerate very fast
- Rendering a single frame takes dt milliseconds
- At $t + dt$, we can predict pos = $X + V dt + \frac{1}{2} A dt^2$
- By starting to render the world from the user's predicted head position, when rendering is complete, it aligns with where there head is by then (hopefully).

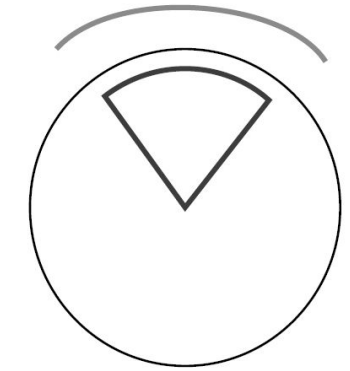
Ex: The WebVR API returns predicted pose by default

Dealing with latency: ‘timewarp’

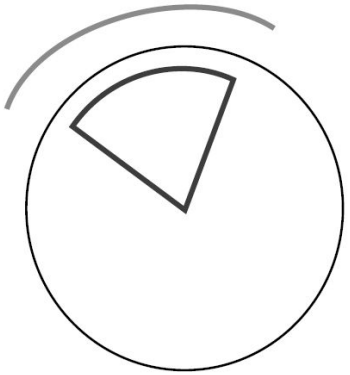
Another technique to deal with lost frames is *asynchronous timewarp*.

- Headset pose is fetched immediately before frame display and is used to shift the frame on the display to compensate for ill-predicted head motion





Original head pose



With timewarp applied

Image credit: Davis, Bryla, Benton, *Oculus Rift in Action* (2014)

Developing for VR

Dedicated SDKs

- HTC Vive
- Oculus Rift SDK
  - C++
  - Bindingsfor Python, Java
- Google Daydream SDK
  - Android, iOS and Unity
- Playstation VR
  - Playstation dev kit

General-purpose SDKs

- WebGL - three.js
- WebVR API

Higher-level game development

- Unity VR

“Sim sickness”

The Problem:

1. Your body says, “Ah, we’re sitting still.”
2. Your eyes say, “No, we’re moving! It’s exciting!”
3. Your body says, “Woah, my inputs disagree! I must have eaten some bad mushrooms. Better get rid of them!”
4. Antisocial behavior ensues

The causes of *simulation sickness* (like motion sickness, but in reverse) are many. Severity varies between individuals; underlying causes are poorly understood.

Reducing sim sickness

The cardinal rule of VR:

> The user is in control of the camera.

1. **Never** take head-tracking control away from the user
2. Head-tracking **must match the user's motion**
3. **Avoid moving the user** without direct interaction
4. If you must move the user, do so in a way that **doesn't break presence**

How can you mitigate sim sickness?

Design your UI to reduce illness

- Never mess with the field of view
- Don't use head bob
- Don't knock the user around
- Offer multiple forms of camera control
  - Look direction
  - Mouse + keyboard
  - Gamepad
- Try to match in-world character height and *IPD* (*inter-pupilary distance*) to that of the user
- Where possible, give the user a stable in-world reference frame that moves with them, like a vehicle or cockpit

Hawken, by Meteor Entertainment (2014)

Further ways to reduce sim sickness

Design your VR world to reduce illness

- Limit sidestepping, backstepping, turning; never force the user to spin
- If on foot, move at real-world speeds (1.4m/s walk, 3m/s run)
- Don’t use stairs, use ramps
- Design to scale--IPD and character height should match world scale
- Keep the horizon line consistent, static and constant
- Avoid very large moving objects which take up most of the field of view
- Use darker textures
- Avoid flickering, flashing, or high color contrasts
- Don’t put content where they have to roll their eyes to see it
- If possible, build breaks into your VR experience
- If possible, give the user an avatar; if possible, the avatar body should react to user motion, to give an illusion of proprioception

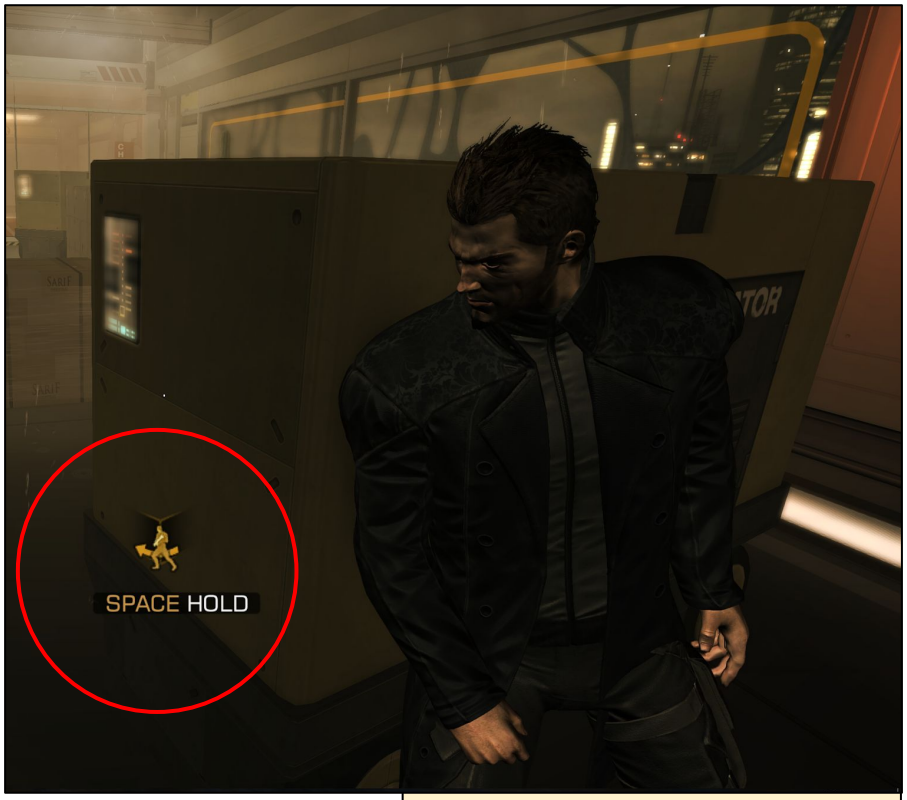
Classic user interfaces in 3D

Many classic UI paradigms *will not work* if you recreate them in VR

- UI locked to sides or corners of the screen will be distorted by lenses and harder to see
- Side and corner positions force the user to roll their eyes
- Floating 3D dialogs create a virtual plane within a virtual world, breaking presence
- Modal dialogs ‘pause’ the world
- Small text is much harder to read in VR

Top: *EVE Online* (2003)
Bottom: *Team Fortress* (2007)

In-world UIs are evolving



Deus Ex Human Revolution (2011)

Deus Ex Mankind Divided (2016)

Increasingly, UI elements are being integrated into the virtual world

The best virtual UI is in-world UI

Top left: Call of Duty: Black Ops (2010)
Bottom left: Crysis 3 (2013)

Top right: Halo 4 (2012)
Bottom right: Batman: Arkham Knight (2015)

Strike Suit Zero (2013)

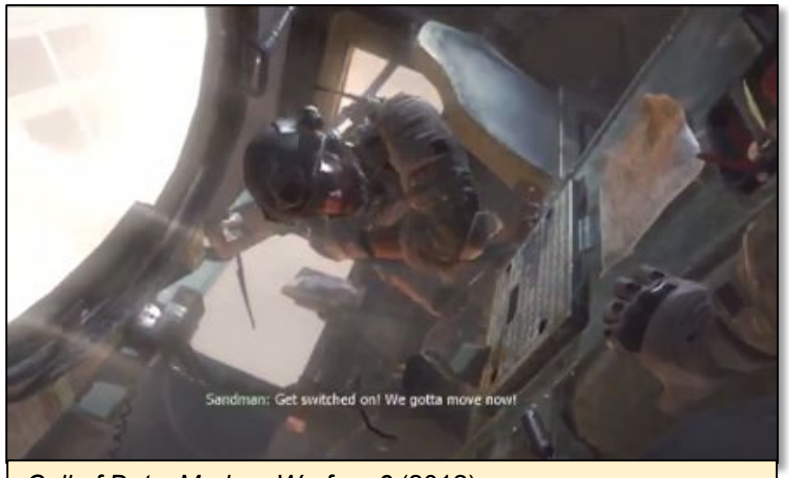
Elite: Dangerous (2014)

Storytelling in games

The visual language of games is often the language of movies

- Cutscenes
- Angle / reverse-angle conversations
- Voiceover narration
- Pans
- Dissolves
- Zooms...

In VR, storytelling by moving the camera will not work well because the user *is* the camera.



Call of Duty: Modern Warfare 3 (2012)
The player's helicopter has been shot down; they emerge into gameplay, transitioning smoothly from passive to active.

"It's a new communications medium. What is necessary is to develop a grammar and syntax. It's like film. When film was invented, no one knew how to use it. But gradually, a visual grammar was developed. Filmgoers began to understand how the grammar was used to communicate certain things. We have to do the same thing with this."

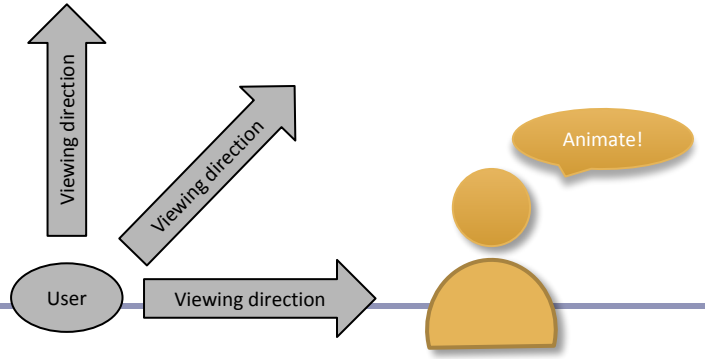
Neal Stephenson, *Interface,* 1994

Drawing the user's attention

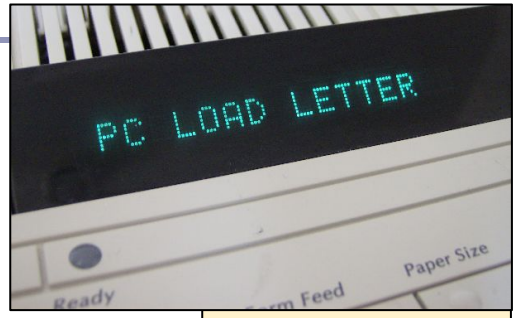
When presenting dramatic content in VR, you risk the user looking away at a key moment.

- Use audio cues, movement or changing lighting or color to draw focus
- Use other characters in the scene; when they all turn to look at something, the player will too
- Design the scene to direct the eye
- Remember that in VR, you know when key content is in the viewing frustum

The Emperor's New Groove (2000)



Advice for a good UI



An unhelpful error message

Always display relevant state—Primary application state should be visible to the user. For an FPS shoot-em-up, this means showing variables like ammo count and health. Combine audio and video for key cues such as player injury.

Use familiar context and imagery—Don't make your users learn specialized terms so they can use your app. If you're writing a surgery interface for medical training, don't force medical students to learn about virtual cameras and FOVs.

Support undo/redo—Don't penalize your users for clicking the wrong thing. Make undoing recent actions a primary user interface mode whenever feasible.

Design to prevent error—If you want users to enter a value between 1 and 10 in a box, don't ask them to type; they could type 42. Give them a slider instead.

Build shortcuts for expert users—The feeling that you're becoming an expert in a system often comes from learning its shortcuts. Make sure that you offer combos and shortcuts that your users can learn—but don't require them.

Don't require expert understanding—Visually indicate when an action can be performed, and provide useful data if the action will need context. If a jet fighter pilot can drop a bomb, then somewhere on the UI should be a little indicator of the number of bombs remaining. That tells players that bombs are an option and how many they've got. If it takes a key press to drop the bomb, show that key on the UI.

Keep it simple—Don't overwhelm your users with useless information; don't compete with yourself for space on the screen. Always keep your UI simple. "If you can't explain it to a six-year-old, you don't understand it yourself" (attributed to Albert Einstein).

Make error messages meaningful—Don't force users to look up arcane error codes. If something goes wrong, take the time to clearly say what, and more important, what the user should do about it.

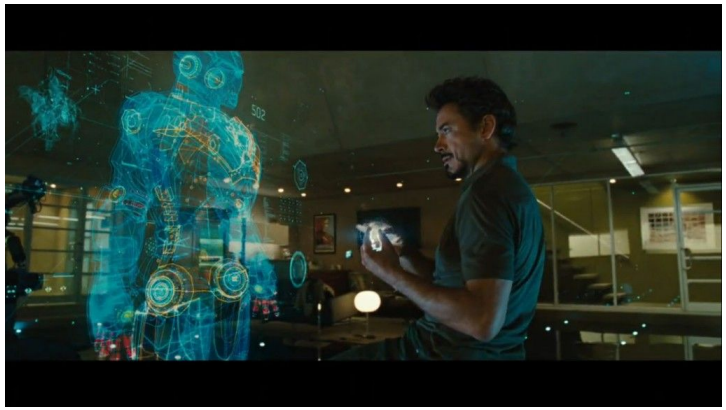
Abridged from *Usability Engineering* by Jakob Nielsen (Morgan Kaufmann, 1993)

Gestural interfaces

Hollywood has been training us for a while now to expect *gestural user interfaces*.

A gestural interface uses predetermined intuitive hand and body gestures to control virtual representations of material data.

Many hand position capture devices are in development (ex: Leap Motion)



Johnny Mnemonic (1995)

Marvel's Agents of S.H.I.E.L.D. (2013) S01 E13

References

Developing in VR

- *Fundamentals of Computer Graphics*, by P. Shirley, M. Ashikhmin, and S. Marschner (A. K. Peters/CRC Press, 2009)
- *Computer Graphics: Principles and Practice*, by J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes (Addison-Wesley Professional, 2013)
- *Oculus Rift in Action*, by Davis, Bryla and Benton (2014)
- *Oculus Best Practices Guide* - developer.oculus.com/documentation

Motion sickness/simulator sickness

- *Textbook of Maritime Medicine*, by the Norwegian Centre for Maritime Medicine (2013). See chapter 20, "Motion Sickness" (textbook.ncmm.no)
- *Validating an Efficient Method to Quantify Motion Sickness*, by B. Keshavarz and H. Hecht (2011). *Human Factors: The Journal of the Human Factors and Ergonomics Society* 53.4: 415–26.
- *Simulator Sickness Questionnaire*, by R. S. Kennedy, N. E. Lane, K. S. Berbaum, and M. G. Lilienthal (1993). *The International Journal of Aviation Psychology* 3(3): 203–20.
- Motion Sickness Susceptibility Questionnaire Revised and Its Relationship to Other Forms of Sickness, by J. F. Golding (1998). *Brain Research Bulletin*, 47(5): 507–16.

UI design for VR

- 3D User Interfaces: New Directions and Perspectives, by D. A. Bowman, S. Coquillart, B. Froehlich, M. Hirose…and W. Stuerzlinger. (2008). *IEEE Computer Graphics and Applications* 28(6): 20–36.
- Design and Evaluation of Mouse Cursors in a Stereoscopic Desktop Environment, by L. Schemali and E. Eisemann (2014). *3D User Interfaces (3DUI), 2014 IEEE Symposium* (pp. 67-70). IEEE. Recorded talk is available at vimeo.com/91489021
- Developing Virtual Reality Games and Experiences— www.gdcvault.com/play/1020714. Presented at GDC 2014.
- Egocentric Object Manipulation in Virtual Environments: Empirical Evaluation of Interaction Techniques, by I. Poupyrev, S. Weghorst, M. Billinghurst, and T. Ichikawa (1998). *Computer Graphics Forum*, 17(3): 41–52.
- Kinect Hand Detection, by G. Gallagher—video.mit.edu/watch/kinect-hand-detection-12073
- *Make It So: Interaction Design Lessons from Science Fiction*, by N. Shedroff and C. Noessel (Rosenfeld Media, 2012)
- Lessons learned porting *Team Fortress 2* to virtual reality—media.steampowered.com/apps/valve/2013/Team_Fortress_in_VR_GDC.pdf
- Pointing at 3D Target Projections with One-Eyed and Stereo Cursors, by R. J. Teather and W. Stuerzlinger. (2013). *ACM Conference on Human Factors in Computing Systems*: 159–68.
- Pointing to the future of UI, by J. Underkoffler (2010). Talk given at TED. www.ted.com/talks/john_underkoffler_drive_3d_data_with_a_gesture
- Selection Using a One-Eyed Cursor in a Fish Tank VR Environment, by C. Ware and K. Lowther. (1997). *ACM Transactions on Computer-Human Interaction Journal,* 4(4): 309–22.
- *Usability Engineering*, by J. Nielsen (Morgan Kaufmann, 1993)