CS 498SL: Virtual Reality

Prof: Steve LaValle, Office: 3318 SC, 2-3pm Thu
TA: Apollo Ellis - computer graphics
Aldi Sipolins - psychology, VR

About myself:
- USC PhD in UWC ECE
- Prof since 2001 in CS, taught 373 a lot
- Robotics researcher; mechatronics, sensor, motion, 3D graphics, simulation, world
- Oculus VR: 2012-2015
- Started by Palmer Luckey
- I joined a few weeks after Kickstarter
- Head tracking, eye calibration, character percep, psychology, health, safety
- Chief Scientist Mad 2013, FB acquisto

Slides: Palmer, Finland, Cecilia Baeza, Grandma Oculus
Course Administration

http://courses.engr.illinois.edu/cs498sl/

- 2 midterms, 1 exam, no final
- Midterm problems: 1-5, work in pairs
- Final project: teams coming, faculty pairing possible

VR Lab: 4240 Siebel Center
12 PCs with Tilted Black graphics cards, 668, 2 matrix, 27”，2
Oculus Rift DK2s
VR Lab: Windows & Visual Studio
- Unity Free game engine
- Programming: "Unity scripts in C# & Java"
- C++ with Raw Oculus SDK

"Not in CS? Meet people outside!"

Textbooks:

"Why there?"
- No modern book on VR
- Wait technology invariant coverage
- PRIOR experience in other fields
- Check extra materials

Lectures: Blackboard + Pictures
Class Forum: Piazza
Social experiences

Goals:
- Learn how to build a good VR experience
- Understand how VR works (engineering & psychology)
- Learn how to critique VR
- Learn fundamentals to shape future of VR

Social: comfortable + adequate for task

Task: Game, write code, maintain relationship, relax, watch film, virtual travel
Definition of VR: (lots of bad definitions out there)

Inducing targeted behavior in an organism by using artificial sensory stimulation, while the organism has little or no awareness of the interference.

Note: Feeding the organism. Not consciously aware.
Note: Does not depend on particular technology, but "artificial" technology is something.

Common terms: immersion
presence

- gerbil, headmill, cockroach, omnitread

Other concepts

Augmented reality: Like Google Glass

Telepresence: Imagine robot + omnidirectional camera + VR headset

Teleportation: Remote control

For me, the world inside the headset does not need to be virtual.

Think also about: - reading a book - watching a movie
- talking on phone - video conferencing
- playing a videogame - listening to music
- Second Life...
Amarents: 
1) Get Ch 4 Math online 
2) Try only tutorials 
3) See dates on Piazza 
4) Project list coming in day or two
Some historical perspective:

- Oldest cave painting: El Castillo hands 40,800 years old
- String art rectangles: Medieval art, bad perspective (1470)
- Perspective fixed: Vredeman de Vries (1600)
- Old movie: L’Arrivée D’un Train En Gare De La Ciotat, 1895 Lumière
- Old movie: A trip to the moon, 1902
- Gravity clip
- Albert Pratt: Wrist gun — Like TF2 by Valve for Oculus 1916
- Sensa-camera: Hellog! 1957 stereo 3D, smells, real vibrations, stereo sound
- Made a headset in 1960; For 3D movies - Telesphere
- Ivan Sutherland - 1968: Sword of Damocles - Video
- CAVE Propstation system: U. Illi lo - Chicago 1991
- VFX1 by Forks, mid 90s
- Virtually gaming systems
- Lawn mow Man 1992
- Nintendo Virtual Boy, 1995 — red lines & headache, death of VR

Beware of hype!

Assigned: Read our Unity Instructure overview & manual

Extra material

- Read Mother, Ch 1
Forgotten from last time:

Books:
  - Mather, Foundations of Sensation and Perception, 2nd Ed., 2009

Assignments:
  - Read Mather, Ch1 by next Thu.
  - read Unity tutorials
  - How to make good VR experiences
  - comfortable & adequate for task

* Show slides and videos, to complete the history

**Bird's-eye view**

Pick a sensor on your body: eye, ear, finger, tongue, nose, to sense? deaf?

Each sensor moves through space or changes in spaceway:
- Controlled by brain (nervous system?)
- Degree of Freedom (DOFs)
- Each sensor has a configuration space: set of all configurations

Ex: Left ear

  Position - 3 DOFs \( \rightarrow \) 6 DOFs total

Oriotation - 3 DOFs

  Right eye

  More complicated: - 6 DOFs

  Rotor with head fixed

  Refocus (accommodation)
BIRDS-EYE VIEW: HARDWARE
An abstract view

control configuration

world → sensor (sense) → brain

ALTER THE WORLD

world → attitude → rendered display (HW) → brain

The brain is fooled

alternate world → sensor → brain

Displays and rendering

Audio and hearing

Display is called a speaker

Rendering produces sounds for the ear(s)
Video and seeing: Display is called a screen, or projector.

Rendering produces image on screen.

No tracking needed if...[diagram with notes]
Compare two audio systems

7.2 Surround

- Displays are far from ears
- Localized audio for fixed room

Note: Could use head tracking to simulate 7.2 using headphones

displays are close
separated by 90°, follows head rotation

Compare two VR visual systems

Cave

HMD (Rift, Morpheus)

Like "eyephones"

Need to track head to show correct images
(or the Zeiss Cinemizer OLED glasses)
Make comparison to audio:

- Head tracking was not needed for 7.2 surround and CAVE.
  - 1.5 needed for HMDs. Why?
  - Why wasn't tracking used for headphones?
    - Not as noticeable for audio, as for video.

  How: listen to music with headphones and turn your head. Does it seem like the band is rotating around you? They are fixed in 7.2. Same is true for CAVE and HMD.

- To fool the brain, we need to undo the sensor transformations caused by head and body motion.

  - Advantages to keeping display as close as possible to sensor:
    - Lower cost
    - Lower power consumption
    - Increased portability

  Hey, why not neural interfaces? Cut the optic nerve!

- Drawbacks:
  - Tracking needed
    - More work for the alternate world generator

  Example: Body tracking for HMD, not CAVE
Inside-out

Outside-in
Regarding CAVE: Tracking not needed unless want to
change viewpoint based on head motion. Includes stereo.
Multiple viewers, time multiplexing frames.

Bugs-Eye View: VR Hardware
Tracking hardware components:

- Inertial Measurement Unit (IMU):
  - Originally designed for aircraft, spacecraft
  - Mainly provide data for orientation estimation
  - Now available in MEMS thanks to smartphones
    (Invensense IMU 3000; St. Microelectronics)
  - From a few dollars to hundreds of thousands; calibration crucial
  - Measure: 3D angular velocity (rad/sec) - gyroscope
    3D linear acceleration (m/sec²) - accelerometer

- Magnetometer: 3D magnetic field (Gauss)

  - Camera: VGA 60FPS etc
    - Inside-out: On headset
    - Outside-in: In room, fixed

- Depth camera: like Kinect

What to track?
- Head, eyes
- Hands, whole body
- Entire surrounding environment
Display Components
- Including vector display (skipped plasma)
  - (CRT, LCD, OLED + others)
  - Light field display
  - Virtual retina display

Audio:
- Standard speaker/earphones
- Bone conduction

Touch:
- Haptic (force feedback) devices
- Keyboard, Xbox controller, mouse

Small Taste? Vestibular.

Visual rendering: Computer graphics (HW+SW)

Visual display:
- Resolution: # of pixels needed per second
- Frame rate: Displayed frames per second (Hz)

Audio is similar with sampling rate & quantization levels

Computer: CPU, GPU

Lens: Palms, Las, Fra Fry's

Input: Xbox controller, keyboard, mouse
Bird's-Eye View Software

input  output

head tracker

alternate world generator (AWG)

visual renderer  visual display

audio renderer  audio display

Some geometry: Superimposing attitude and physical worlds

- Stuff: walls, plants
- Collision detection
- Physical simulation
- Avatar model

mobility region in physical world

head tracker at work here!
Assessment: Read/Review Shirley Ch 2, 5, 6
Saddle Ch 3, 4
Unity 3D guide
Three kinds of motions:

1) User moves, and motion is matched in alternate world.
2) Through an interface, user locomotes its virtual self (avatar).
   - Like moving on a virtual cart.
   - May be uncomfortable: Virtual/perceived motion only.

Others:
(Alternative: Other avatars, objects in alternate world)

Alternate World Generator (AWG) Example:

1) Game engines: Unity 3D, Unreal Engine 4
   Handles avatar motion, physics, visual rendering.
   Powerful, but not optimized for VR.

2) Google Street View server + interface for navigation

3) Robot + Camera + Video processing, drone, mobile robot, humanoid

4) Write your own simulator with avatar motion controls.

Someday we should have nice VR engines!
Affinities Word Concept (AMC: Example)

Diagram:

- Organism
- VR HMD

Output

Input
Bird's-Eye View: Sensation & Perception
(Perceptual psychology; vision science)

VR Questions
- How do we perceive depth? (3D)
- What causes nausea, fatigue?
- How do we perceive an audio source?
- How much resolution is enough in a display?
- What is presence?

Our perception process appears to be simple:
Immediate, effortless, direct

Cerebral cortex: ~10 billion neurons
- outer layer, folded sheet 2.5 mm thick
- most advanced part of brain;
- most neurons devoted to vision perception

Computers can't beat cerebral cortex

Like a big directed graph:

The brain fixes things to

Human brain

Order of importance for our class:
1) Vision
2) Balance
3) Hearing
4) Smell
5) Taste/Sense

former body
### Classification of Senses

<table>
<thead>
<tr>
<th>Sense</th>
<th>Stimulus</th>
<th>Receptor</th>
<th>Sensory Structure</th>
<th>(optional)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vision</td>
<td>Electrostatic energy</td>
<td>Photoreceptors</td>
<td>Eye</td>
<td>Robot Equal. Camera</td>
</tr>
<tr>
<td>Hearing</td>
<td>Acoustic waves</td>
<td>Mechanoreceptors</td>
<td>Ear</td>
<td>microphone</td>
</tr>
<tr>
<td>Touch</td>
<td>Tissue dist.</td>
<td>Mechanoreceptors</td>
<td>Skin, muscle</td>
<td>pressure sensors, thermometers</td>
</tr>
<tr>
<td>Balance (Vestibular)</td>
<td>Gravity, acceleration</td>
<td>Mechanoreceptors</td>
<td>Vestibular organs</td>
<td>IMU</td>
</tr>
<tr>
<td>Taste/Smell</td>
<td>Chemical compounds</td>
<td>Chemoreceptors</td>
<td>Mouth / Nose</td>
<td>Ph.</td>
</tr>
</tbody>
</table>
Most important for us: Vision + Balance (Vertebral)

1)vection  →  Optical Flow disagrees with vestibular
2) Vestibulo Ocular Reflex (VOR)
   - Affects sense of stationarity
   - Eye counter-rotates with head rotation to fixate
     eyes on a target.

Some fundamental concepts in sensory and perception

Receptors: Transducers that convert stimulus energy to neural impulses
  (Much like an engineered sensor to an analog or digital signal)

(p.14 Maier)

Physical World  →  Receptor  →  Intermediate Neuron  →  Interneurons  →  Thalamus  →  Primary Cortex

Hierarchical Processing!

Sensory system selectivity
  - Each cell has a receptive field
  - Limited frequency, color, shape -- for each ("activation zone")
  - Electrodes to perform single-unit recordings
    (Display particular stimulus, determine whether cell active)
Organization of neurons (network spatial topology)

Topographic Map

Psychophysics: Scientific study of

Physical stimuli → perceptual phenomena (Conscious mental state)

What range of stimuli produce "red"? "sour"? "dark"?

Probability of detection

stimulus parameter (e.g. intensity)

Criticism: You can't tell the joke by calling C as A, but

Related: Just noticeable difference (JND)
Joke: Stigler's Law of Eponomy

No scientific law is named after its original discoverer.

The discoverer: Robert Merton.
Magnitude estimation:

Steven's power law (1961) - see Wikipedia

\[ m \rightarrow cm^x \quad p = cm^x \]

- \( m \) = magnitude or intensity of stimulus
- \( p \) = perceived magnitude
- \( x \) = exponent that depends on particular exponent
  - (could have \( x < 1 \), \( x > 1 \), \( x = 1 \))
- \( c \) = constant based on units, etc.

Examples:

- Perimeter: 0.67
- Loudness: 1
- Visual length: 1
- Electric shock: 3.5
- Sound 3000Hz tone
- Projected line
- Current through finger

Criticism: You can fit the data by choosing \( c \) and \( x \), but is the law always "correct"?
Related: Just-Noticeable Difference (JND)

Weber's law:

\[ \frac{\Delta m}{m} = c \]

(in other words, only percent change matters)

\( \Delta m \) is the JND of stimulus magnitude

Plasticity & adaptability

![Graph showing impulses per second over time](image)
May be better now.

1) Stationary models
2) Movable models

Obstacles:
1) Visual: block visibility
2) Physical: block motion
Geometric Modeling

Let $W$ be a 3D world: $W = \mathbb{R}^3$

Every point in $W$ described by coordinates $(x, y, z) \in W$

The world has two kinds of objects inside:

1) Obstacles, which never move: walls, furniture

2) Movable bodies, which have a space of possible transformations (configuration space)
   - Avatars, bullets, animals, blowing leaves, coffee cup

Coordinate frame: - Obstacles described in world frame
                 - Each rigid body described in its own body frame

Let's model obstacles, but some principles apply for bodies

Two choices: 1) Solid representation
             2) Boundary representation
Boundary representation is easier, but almost always hard to distinguish "inside" from "outside".

Collision: In collision? ?

Worst-case model: Polygon soup. Think about real world data from cameras, Kinect.

Let's use triangle primitives $(x_1, y_1, z_1)$

$(x_2, y_2, z_2)$

$(x_3, y_3, z_3)$

$\Rightarrow$ A plane patch

Graphical problem: How to render the surface?
- Compute normal - ambiguity problem

Strip - approximate any surface
- saves vertices
- might have numerical holes
- GPUs like these!

Mesh
Transforming Rigid Bodies

Transform each vertex of each triangle  \( \text{DOF} \)

3 cases:
- Easy: Translation
- Harder: Rotation
- Hardest: Rotation + Translation

<table>
<thead>
<tr>
<th>2D</th>
<th>3D</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

Translation

Shift triangle by \((x_t, y_t, z_t)\)

\[
\begin{align*}
(x_1, y_1, z_1) & \implies (x_1 + x_t, y_1 + y_t, z_1 + z_t) \\
(x_2, y_2, z_2) & \implies (x_2 + x_t, y_2 + y_t, z_2 + z_t) \\
(x_3, y_3, z_3) & \implies (x_3 + x_t, y_3 + y_t, z_3 + z_t)
\end{align*}
\]

We need to get to 3D rotation. Start with 2D linear transformations:

\[
\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
\]
basis for Cartenza coordinates
\( \hat{x}, \hat{y}, \hat{z} \)

\[
\begin{bmatrix}
M_{11} & M_{12} \\
M_{21} & M_{22}
\end{bmatrix}
\begin{bmatrix}
1 \\
0
\end{bmatrix}
=
\begin{bmatrix}
m_{11} \\
m_{21}
\end{bmatrix}
\]

\[
\begin{bmatrix}
M_{11} & M_{12} \\
M_{21} & M_{22}
\end{bmatrix}
\begin{bmatrix}
0 \\
1
\end{bmatrix}
=
\begin{bmatrix}
m_{12} \\
m_{22}
\end{bmatrix}
\]

Explain:

\[
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
2 & 0 \\
0 & 2
\end{bmatrix}
\begin{bmatrix}
-1 & 0 \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
-1 & 0 \\
0 & -1
\end{bmatrix}
\]

no effect  scale  mirror  rotate 90°

\[
\begin{bmatrix}
1 & 1 \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\]

x-shear  y-shear

A circle of rotation
\([ r \cos \theta, r \sin \theta ]
\]

A square of rotation
\([ x', y'] = [ x \cos \theta - y \sin \theta, x \sin \theta + y \cos \theta ]
\]
For rotations:
1) No stretching of axes - preserve scale
2) No shearing - orthogonal axes must result
3) No mirror images - preserve orientation

4 DOFs

1: Columns have length one

\[
\begin{bmatrix}
0 & 0 \\
\end{bmatrix}
\]

\[
m_{11}^2 + m_{31}^2 = 1
\]

\[
m_{12}^2 + m_{32}^2 = 1
\]

2: Columns are perpendicular. Inner products = 0

\[
m_{11}n_{12} + m_{31}n_{32} = 0
\]

3: \( \det((\cdot)) = 1 \) (not -1)

Which matrices remain as rotations?

\[
\begin{bmatrix}
\cos \theta & -\sin \theta \\
\sin \theta & \cos \theta
\end{bmatrix}
\]

for all \( \theta \in [0, 2\pi) \)

To rotate \((x, y)\):

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} = \begin{bmatrix}
\cos \theta & -\sin \theta \\
\sin \theta & \cos \theta
\end{bmatrix}
\]

A circle of rotation:

\[
m_{1} = \cos \theta
\]

\[
m_{2} = \sin \theta
\]
Now try 3D rotation. Which 3x3 matrices are rotations?

\[
\begin{bmatrix}
\begin{array}{c}
\vdots \\
V_1 \\
\vdots \\
V_2 \\
\vdots \\
V_3 \\
\vdots
\end{array}
\end{bmatrix}
\]

- 9 DOFs
- \[ \| V_1 \| = \| V_2 \| = \| V_3 \| = 1 \]
- \[ V_1 \cdot V_2 = 0, \quad V_2 \cdot V_3 = 0, \quad V_1 \cdot V_3 = 0 \]
- \[ \det \left( \begin{bmatrix} \vdots & \vdots & \vdots \end{bmatrix} \right) = 1 \]

A 3D "surface" (manifold)

3 Canonical Rotations: Yaw, Pitch, Roll

**Yaw**:

\[
R(\alpha) = \begin{bmatrix}
\cos \alpha & 0 & \sin \alpha \\
0 & 1 & 0 \\
-\sin \alpha & 0 & \cos \alpha
\end{bmatrix}
\]

Leave y coordinate untouched

**Pitch**:

\[
R(\beta) = \begin{bmatrix}
1 & 0 & 0 \\
0 & \cos \beta & -\sin \beta \\
0 & \sin \beta & \cos \beta
\end{bmatrix}
\]

Leave x untouched

**Roll**:

\[
R(\gamma) = \begin{bmatrix}
\cos \gamma & -\sin \gamma & 0 \\
\sin \gamma & \cos \gamma & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

Leave z untouched

Every 3D rotation \( R \) can be achieved by some \( \alpha, \beta, \gamma \) with \( R = R(\alpha)R(\beta)R(\gamma) \) with \( \alpha \in [-\pi, \pi], \beta \in [-\pi, \pi], \gamma \in [-\pi/2, \pi/2] \).
Example

\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & \cos(-\theta) & -\sin(-\theta) \\
0 & \sin(-\theta) & \cos(-\theta)
\end{bmatrix}
\begin{bmatrix}
\cos(\theta) & 0 & \sin(\theta) \\
0 & 1 & 0 \\
-\sin(\theta) & 0 & \cos(\theta)
\end{bmatrix}
\begin{bmatrix}
\phi(0) & 0 & 1 \\
0 & \phi(0) & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

After $\theta = \frac{\pi}{2}$, we have:

\[
\begin{bmatrix}
8 & 0 & 1 \\
0 & 8 & 0 \\
-1 & 0 & 0
\end{bmatrix}
\]

Result

\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & \sin(\phi) & -\cos(\phi) \\
0 & \cos(\phi) & \sin(\phi)
\end{bmatrix}
\begin{bmatrix}
\cos(\phi) & 0 & \sin(\phi) \\
0 & 1 & 0 \\
-\sin(\phi) & 0 & \cos(\phi)
\end{bmatrix}
\begin{bmatrix}
\phi(0) & 0 & 1 \\
0 & \phi(0) & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

Apply trig identity:

(Roll+Pitch) yields same rotation, regardless of Roll or Pitch angles.

\[
\begin{bmatrix}
0 & 0 & 1 \\
0 & \cos(\theta) & -\sin(\theta) \\
0 & \sin(\theta) & \cos(\theta)
\end{bmatrix}
\begin{bmatrix}
\phi(0) & 0 & 1 \\
0 & \phi(0) & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

End 3D orientation in component form may be written as:

\[
\begin{bmatrix}
0 & 0 & 1 \\
0 & \cos(\theta) & -\sin(\theta) \\
0 & \sin(\theta) & \cos(\theta)
\end{bmatrix}
\begin{bmatrix}
\phi(0) & 0 & 1 \\
0 & \phi(0) & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

With $\theta = \frac{\pi}{2}$.
Problems
- Order matters: 3D rotations not commutative!!
- Kinematic singularities at nonuniform rotation
  - Vary $\alpha, \beta, \gamma$ at constant speed, all rotate very wildly
  - Gimbal lock

(illustrate with prop) - Problem: Rotation axis keeps changing on body

Euler's rotation theorem: All 3D rotations have an axis-angle representation
(1776)

$$\mathbf{v} = (v_1, v_2, v_3)$$

$$|\mathbf{v}| = 1$$

Note

$$\mathbf{v}$$ and $$\mathbf{v} + (2\pi - \theta)$$ are the same rotation!

It would be nice to have some algebra

$$(V, \theta) \circ (V', \theta') = ?$$

so that we always know the resulting axis and angle. (Not in 2D only)

(In OpenCV, exponential coordinates used: $$(\theta, \theta_1, \theta_3)$$)

scale is the angle!

Nice! 3 parameters for 3 DOFs

$$60, 0) = \text{identity}$$
Note: 3DOFs, as needed for 3D rotation.

\[ \mathbf{V} = (\theta, \mathbf{V}) \cdot (\mathbf{V}, \theta) \]
Unit quaternions as a better representation of 3D rotation.

\[ q = (a, b, c, d) \in \mathbb{R}^4 \text{ and } a^2 + b^2 + c^2 + d^2 = 1 \Rightarrow \text{"normalized"} \]

Set of all \( q \) is a hypersphere \((S^3)\)

Unit 3D/4D game representation: \((x, y, z, w) = (b, c, d, a)\)

\[ a + bi + cj + dk \]

"Real part" "Imaginary" part like complex numbers \( a + bi \)

Most common representation:

Encoding a 3D rotation using unit quaternions:

\[ v = (v_x, v_y, v_z) \quad \text{with} \quad |v| = 1 \]

\[ (\cos \theta / 2, v_x \sin \theta / 2, v_y \sin \theta / 2, v_z \sin \theta / 2) \]

Access to recover angle. Just perform \( \sqrt{b^2 + c^2 + d^2} \) to recover \( a \) as the scalar part.
Useful examples:

\[
\begin{align*}
(1, 0, 0, 0) & \quad \text{identity rotation} \\
(0, 1, 0, 0) & \quad \text{pitch by } \pi_x \\
(0, 0, 1, 0) & \quad \text{yaw by } \pi_y \\
(0, 0, 0, 1) & \quad \text{roll by } \pi_z \\
(1, \frac{1}{2}, 0, 0) & \quad \text{roll by } \pi_x \\
(1, 0, \frac{1}{2}, 0) & \quad \text{roll by } \pi_y \\
(1, 0, 0, \frac{1}{2}) & \quad \text{roll by } \pi_z \\
\end{align*}
\]

The bigger the axis part, the bigger the rotation.

Inverses and multiple representations:

\[
(a, b, c, d) \quad \overset{\text{equivalent}}{\longleftrightarrow} \quad (-a, -b, -c, -d)
\]

\[
(a_1, b_1, c_1, d_1) \quad \overset{\text{equivalent}}{\longleftrightarrow} \quad (-a_1, b_1, c_1, d_1)
\]

Multiplication:

Let \( p = (b, c, d) \)

\[
q_1 \circ q_2 = q_3
\]

\[
q_1 \circ q_2 = (a_1 q_2 - p_1 \cdot p_2, \ p_1 \cdot p_2 + a_1 p_2 + a_2 p_1)
\]

\[
q_3 = (b_3, c_3, d_3)
\]

Can't be commutative!!!
R - useful because "active"; rotate directly

q - no singularities
  - small change in q = small change in rotation
    (just like \( \theta \) in 2D rotation)
  - normalized hemispheres

PR - intuitive
  - bad singularities

\[
R_3 \hat{R} \hat{R} \hat{R} \hat{R} R_3 R_3 = R_3 \hat{R_2} R_2 R_3 = R_3 \hat{R}_3 R_3 = I
\]
Conversion

Rotation matrices: Start with $R$

- Two-to-one: Quaternion
- One-to-one: (Axis, angle)

$R_1 \cdot R_2 = R_3$

Down, down, up

$q \circ q = q_2$

Note: Can apply $q$ directly to rotate $(x, y, z)$.

Quaternions order: Which is applied first? $R_1 \cdot R_2$

$R_1(R_2[\xi]) \rightarrow R_2$ is!

Beware of inverses:

$\left(R_1 R_2 R_3\right)^{-1} = R_3^{-1} R_2^{-1} R_1^{-1}$ (Linear algebra, group theory)
Which is first? \( p = (x, y, z) \)

\[ R \cdot Q \cdot p = Rp, \quad p'' = Qp' \rightarrow R \rightarrow Q \]

Which way together?

\[ p'' : QRp \text{ or } p'' : RQp ? \]

First make

\[
\begin{bmatrix}
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0
\end{bmatrix}
\]

Rotate using quaternion \( q \) directly

\[ p = (x, y, z, 1) \quad p' = qpq^{-1} \]

\( q^{-1} = (a, -b, -c, -d) \)
Homogeneous transformation matrices - A math hack

Rotate followed by translate: Place a rigid body anywhere

\[
\begin{bmatrix}
    x' \\
    y' \\
    z'
\end{bmatrix} =
\begin{bmatrix}
    R & x_4 \\
    0 & 1
\end{bmatrix}
\begin{bmatrix}
    x \\
    y \\
    z
\end{bmatrix}
\begin{bmatrix}
    1
\end{bmatrix}
\]

This is outside of matrix multiplication (vector addition)

Make a 4x4 matrix that is algebraically equivalent (add 4D vector)

\[
\begin{bmatrix}
    x' \\
    y' \\
    z'
\end{bmatrix} =
\begin{bmatrix}
    R & x_4 \\
    0 & 1
\end{bmatrix}
\begin{bmatrix}
    x \\
    y \\
    z
\end{bmatrix}
\begin{bmatrix}
    1
\end{bmatrix}
\]

Same result, but extra "1" in 4th position

Now you get one 4x4 matrix \( T = \begin{bmatrix} R & x_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \) that places rigid bodies

Important question: What is \( T^{-1} \)? (How to undo the placement?)
What is $R^{-1}$? $R^{-1} = R^T$ for a transposer. Only for "orthogonal matrix" orthonormal.

Opposite of $x_4, y_4, z_4$ is $-x_4, -y_4, -z_4$.

Is $T^{-1} = \begin{bmatrix} R^T \\ 0 & 0 & 0 & 1 \end{bmatrix}$? No!!

Wrong order of operations.

$$T^{-1} = \begin{bmatrix} R^T & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_4 \\ 0 & 1 & 0 & -y_4 \\ 0 & 0 & 1 & -z_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Now let's carry only about transforming points from $W$ to see.

The remainder is in the rendering pipeline.
The Chain of Transformations is cyclical.

### Viewing Transformations

![Diagram of viewing transformations]

Questions:

- How should artificial light propagate?
- How to discretize geometric model onto screen?
- How to determine what is visible or blocked?
- How to make correct perspective?
- Efficiency? Take a shortcut if flaws are not perceptible.

Now let's worry only about transforming points from W to screen (points, line segments, triangles...).

The remainder is the rendering pipeline.

- Log: becomes an inverse of a rigid body motion
- Tan: apply perspective transform and rescale, translate
- Tap: units convenient and more pixels, translate
Apply

\[ T \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} i \\ j \end{bmatrix} \]

Units

Body, free, world frame, eye frame: meters (or feet)
Sensory, view frame: none
Screen frame: pixel index
Let $T$ denote a $4 \times 4$ homogeneous transformation matrix. For viewport from body coordinates:

$$T = T_{up}T_{can}T_{eye}T_{rb}$$

Insert $T_{rb}$ for stereo rendering.

Insert $T_{dist}$ to compensate for lens distortion.

Look at each transformation:

- $T_{rb}$ - already covered (rotation + translation)
- $T_{eye}$ - becomes an inverse of a rigid body plecost
- $T_{can}$ - apply perspective transform and rescale, translate
- $T_{up}$ - units conversion and more rescale, translate
Why is \( \hat{c} \) not simply \( \vec{0} \)?

This allows slippy input! \( \vec{0} \) need not be perpendicular to image plane.
Fixed (Galilean) Eye-Transformation

Express eye's rotational translation in world coordinate position (pose)

\[ x \text{ in } \mathbb{R}^3 \]
\[ \mathbf{e} \text{ coincides with } -z \]

Consider a "lookat":

1. Position of eye: \( \mathbf{e} \)
2. Looking direction in eye's coordinate: \( \mathbf{e'} \)
3. Up direction: \( \mathbf{u} \)

Make a transform to place eye in \( W \)

Coordinate axes:
- \( \hat{x} = \hat{z} \times \hat{y} \)
- \( \hat{y} = \hat{x} \times \hat{z} \)
- \( \hat{z} = -\hat{e} \)

\[
\mathbf{R}_{ee} = \begin{bmatrix}
\hat{x}_1 & \hat{y}_1 & \hat{z}_1 \\
\hat{x}_2 & \hat{y}_2 & \hat{z}_2 \\
\hat{x}_3 & \hat{y}_3 & \hat{z}_3
\end{bmatrix}
\]

To transform world into eye, you need inverse transform.
\[
T_{\text{left eye}} = \begin{bmatrix}
1 & 0 & 0 & \frac{t}{2} \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} T_{\text{eye}}
\]

\[
T_{\text{right eye}} = \begin{bmatrix}
1 & 0 & 0 & -\frac{t}{2} \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} T_{\text{eye}}
\]

\( t = \text{inter eye distance in virtual world} \)

\( t = 0.064 \text{ m in real world} \)
Interpretations:
1) Dragging the world as if it were a rigid body
2) Tiny eye at origin (e = (0,0,0))
   - Yaw world by \( \Theta \) (\( \text{without many eyes} \))
   - Yaw eyes by \(- \Theta\) (\"world\")

Thus,
\[ R_{\text{eye}} = R^{-1}_{\text{eye}} \]

\[
T_{\text{eye}} = \begin{bmatrix}
\hat{x} & \hat{x} & \hat{y} & 0 \\
\hat{y} & \hat{y} & \hat{z} & 0 \\
\hat{z} & \hat{z} & \hat{z} & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & -e_x \\
0 & 1 & 0 & -e_y \\
0 & 0 & 1 & -e_z \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Inverse translation

Stereo Canonical View Transform

Viewing frustum

Projective geometry (space of lines through origin)

Perspective transforms
\[ h = \frac{1}{z_p} y_p \]
Dividing by \( z_p \) is not linear
The transformed corners of viewing frustum:

\((l, b, n, r, f, f)\)
Trick by using multiplication to do the algebra.

1D example (homography):

\[
\begin{bmatrix}
y' \\
h'
\end{bmatrix} =
\begin{bmatrix}
3 & 0 & 0 \\
0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
y_p \\
z_p
\end{bmatrix} =
\begin{bmatrix}
3y_p \\
z_p
\end{bmatrix}
\]

Define \( h = \frac{y'}{h'} = \frac{dy_p}{z_p} \) (same as previous figure).

\( n, f \)

\[
T_p =
\begin{bmatrix}
n & 0 & 0 & 0 \\
0 & n & 0 & 0 \\
0 & 0 & m & -f \\
0 & 0 & 1 & 0
\end{bmatrix}
\]

\( T_{st} = \frac{2}{r^2} \begin{bmatrix}
2 & 0 & 0 & -r \\
0 & \frac{1}{2r^2} & 0 & 0 \\
0 & 0 & \frac{1}{2r^2} & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \)

\( T_{can} = T_{st} \cdot T_p \)

(Tech in Shirley)

Scale and Translate

\( x' = nx \)
\( y' = ny \)
\( z' = z(nf) - fn \)

\( \frac{2f}{x''} \)

\( x' : l \leftarrow r \) (left \rightarrow right)
\( y' : b \leftarrow t \) (bottom \rightarrow top)
\( z' : n \leftarrow f \) (near \rightarrow far)
One remaining part: $T_{vp}$ converts $-1...1$ range to pixel coordinates.

- $N_x = \# \ of \ horizontal \ pixels \ (e.g. \ 1920)$
- $N_y = \# \ of \ vertical \ pixels \ (e.g. \ 1080)$

$$
T_{vp} = \begin{bmatrix}
\frac{N_x}{2} & 0 & 0 & \frac{N_x - 1}{2} \\
0 & \frac{N_y}{2} & 0 & \frac{N_y - 1}{2} \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
i \\
j
\end{bmatrix} = \begin{bmatrix}
\tilde{i} \\
\tilde{j}
\end{bmatrix} \text{ pixel indices}
$$

Point source of light
- Generates constant:
- Rays are orthogonal to point light.
Light

Three interpretations:
1) particles (photons)
   \[ f = \frac{c}{\lambda} \quad \text{frequency} \]
   \[ c \quad \text{speed of light} \]

2) rays
   computer scientists love this!
   (graphics, computational geometry)

3) waves

Point source of light:
- Generates wavefronts
- Rays are orthogonal to wavefronts

Without mirrors or lenses, rays always diverge

\[ \text{Parallel rays} \Rightarrow \text{(almost) parallel rays} \]
parallel rays → zero vergence
Collimated rays → rays to or from infinity

We use materials to bend the rays/mars

Transmission (Refraction)
Absorption
Reflection

Reflection types:
- Specular
- Diffuse

Snell's Law:
\[
\frac{n_1}{n_2} = \frac{\sin \theta_1}{\sin \theta_2}
\]

Refractive Index:
\[
\frac{\lambda_1}{\lambda_2} = \frac{c}{n_1} = \frac{c}{n_2}
\]
Refraction

Two interfaces

Simple Lenses (Spherical)

Focal point

Focal length

Wave front interpretation

Focal point

Focal length

[draw these above in different color]
Lenses' equation

\[ \frac{1}{f} = (n-1) \left[ \frac{1}{R_1} - \frac{1}{R_2} + \frac{(n-1)d}{nR_1R_2} \right] \]

Index of refraction \( n \)

Thin approximation: \( d \approx 0 \)

\[ \frac{1}{f} = (n-1) \left[ \frac{1}{R_1} - \frac{1}{R_2} \right] \]

Convenient unit: Diopeter (m\(^{-1}\))

Converging/diverging power of lens

\[ 20 \text{cm} \]

\( 5 \text{D} \) (5 diopters) or \( \left( \frac{1}{f} \right) \text{D} \)
Could have diverging lens

Lens power \( \left( \frac{1}{f} \right) \) \( \rightarrow \) negative because lens causes divergence

\[ D < 0 \quad D = 0 \quad D > 0 \]

Combining lenses or interfaces:

\[ D = D_1 + D_2 + D_3 \quad \text{---} \]

Assumes distance between lenses or surfaces \( \geq 0 \)

("Thin lens approximation")

\( \text{Optometrists must not like algebra!} \)
$\text{From p.159}$

W. J. Smith

Modern Optical Engineering

(Oblique image of the object)
Structure of the human eye

Retina acts as focal points
Non-planar, non-uniform receptors

Vitreous humour
Fovea
Optic nerve
Pupil
Ciliary

\[ n_1 = 1.009 \quad \text{air} \]
\[ n_2 = 1.376 \quad \text{cornea} \]
\[ n_3 = 1.386 \quad \text{ocular fluid (aqueous)} \]
\[ n_4 = 1.413 \quad \text{lens} \]
\[ n_5 = 1.387 \quad \text{vitreous} \]

(YP is dry tissue)

Total power of eye's optical system: 59.82 D (16.8 mm)
Imaging properties of a lens

\[ \frac{1}{S_1} + \frac{1}{S_2} = \frac{1}{f} \]

\( S_1, S_2 > f \)

If \( S_1 = \infty \), then \( S_2 = f \)
(parallel light)

The human eye lens adjusts \( f \) (or D) to bring closer objects into focus

Could have \( S_2 \) be negative: magnifying glass

Think about looking into the Rift

(real image still pictured at real focal plane by eye lenses)

\( S_2 < 0 \)
Along horizontal axis, this is like writing a stretched move from □ to □
Lens Aberrations

1. Spherical aberration

Solution: \[ \ell (\text{elliptical}) \rightarrow (\text{hyperbolic}) \]
Harder to manufacture (aspheric lens)

Related: Look into edges of Rift, less and image might not be focused

2. Optical distortion

For high field of \( \omega \) lenses, the image is distorted

\[
\begin{align*}
\text{original input} & \quad \Rightarrow \quad \text{view through lens (virtual image)}
\end{align*}
\]

Nearly all pinwheel light is a mixture of wavelengths/ frequencies (white: dark is not in spectrum)
Inverse of pin cushion is barrel distortion

\[ \Rightarrow \]

Think old CRT monitors

In RFT to compensate for pin cushion distortion, apply a barrel distortion to rendered image.
- Now easy to do in graphics hardware (1990s had warped images)
- Apply in last stage of chain of transforms

\[ r \rightarrow f(r) \]

(only a function of radius, or distance to image center)

3. Chromatic aberration

Light is composed of waves, varying between \( \approx 400 \text{nm} \) and \( \approx 700 \text{nm} \) wavelengths

\(<400\text{nm} \text{ is ultraviolet; } >700\text{nm is infrared} )

(Each wave has frequency \( f = \frac{c}{\lambda} \))

The visible spectrum consists of pure waves (single sinusoid):

- Red, Orange, Yellow, Green, Blue, Indigo, Violet

Nearly all perceived light is a mixture of wavelengths/frequencies

(Note: White is not in spectrum)
Spectral power - Like a histogram of wavelengths

Emission of source: relatively even

Incandescent lamp

Daylight

Spectral reflectance

Receiving color of an object: Source spectrum AND reflectance spectrum

Dispersion: Speed of light in medium depends on its wavelength!

White light

Full spectrum

Something happens in a lens

E.g. Blue focal length < red focal length
Solution:

1) Find lens with high Abbe number (prisms)
2) Make compound lens
3) Compound in soft wax by shifting glass on pupil wavelengths

4. Astigmatism
   - Elliptical eccentricity in lens
   - Can happen in human eye
   - Vertically and horizontally
   - Vertical vs horizontal, no common focus

5. Coma
   - Off-axis distortion of image (like "comet")
   - Airy pattern near varying lens thickness from edge to center
How does the eye's lens/image system work?

Accommodation: Lens shape changes to increase dioptric power.
Normal eye examples:

1) Note that retinal image is not planar.
   - Eye lens is not circular
   - Shallowly focused on retina
   - Eye muscle relaxed

2) Blurry on retina
   - Eye muscle relaxed

3) 
   - Eye uses muscles to accommodate
   - Dioptric power increases

4) Reading glasses: Make close objects "seem like" infinite distance
   - Lens makes rays parallel
   - No accommodation required
"The Dress"

Blue & Black
White & Gold

For me: Red & Yellow

News Flash! The dress contained thermochromic dye! When you take a picture, the color changes on keep.
A video rendering system and optical system put real image on retina (fleshy eye)

Hold eye fixed!

The image hits a spherical surface of densely packed photoreceptors: Like pixels, but input instead of output.

Two kinds: Rods and Cones

<table>
<thead>
<tr>
<th></th>
<th>Rods</th>
<th>Cones</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number (per eye)</td>
<td>120,000,000</td>
<td>6,000,000</td>
</tr>
<tr>
<td>Function</td>
<td>Low light visibility</td>
<td>Color sensitivity</td>
</tr>
</tbody>
</table>
Diodes:
Note: 1 μm is close to visible light wavelengths.

0.4 μm to 0.7 μm

Can't get much smaller without bleed-over into one pixel.

Area of retina: 1091 mm² → close to 200,000,000 photoreceptors

Density of fovea: ~200,000/mm²

If resolution too low:

Image of one pixel
At 0°: all cones  
At 2°: cones get bigger (16μm diameter)  
At 50°: mostly rods  
Rob ~ lamina

Implication: - Sharp color at retina; sharp monochrom at periphery  
- Retina/color requires high light intensity

How much resolution is enough for VR?

Rough estimate: \( \sqrt{126,000,000} \approx 11225 \)  

Perhaps screen should be 16k x 16k. Any more is a waste!
More about photoreceptors

Light intensity: **Photometric** take into account human sensitivity

\[ \text{luminance} \times \text{candela/m}^2 = \text{cd/m}^2 \]

<table>
<thead>
<tr>
<th>Material</th>
<th>Luminance (cd/m²)</th>
<th>Photons/m²-sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paper in daylight</td>
<td>0.0003</td>
<td>0.01</td>
</tr>
<tr>
<td>Paper in moonlight</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>Monitor</td>
<td>63</td>
<td>160</td>
</tr>
<tr>
<td>Room Light</td>
<td>376</td>
<td>1600</td>
</tr>
<tr>
<td>Bluesky</td>
<td>2500</td>
<td>10,000</td>
</tr>
<tr>
<td>Paper in sunlight</td>
<td>40,000</td>
<td>100,000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Spectral vision</th>
<th>Photopic vision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Photoreceptors</td>
<td>Red</td>
<td>Cone</td>
</tr>
<tr>
<td>Light level</td>
<td>&lt;0.01 cd/m²</td>
<td>&gt;10 cd/m²</td>
</tr>
<tr>
<td>Dark adaptation</td>
<td>35 min</td>
<td>10 min</td>
</tr>
<tr>
<td>Color</td>
<td>Monochromatic</td>
<td>Trichromatic</td>
</tr>
</tbody>
</table>

- **Stereoscopic vision** (night)
- **Photopic vision** (day)

- 96% of cells dedicated to this
- near day time animals
Next problem: The eye can rotate!

Right eye (nose →)
Superior rectus → Inferior oblique
Lateral rectus → Medial rectus
 Inferior rectus
 Superior Oblique

LM, MR: Eye yaw (up to 50 degrees), adduction (non-vestibular part)
SR, IO, IR, SO: Eye pitch, avoiding roll

Types of eye movements

Conjugate  Disjunctive

Voluntary  Saccade  Convergence
     Dorsiflex
Involuntary  Versions/Ocular Divergence
     Optokinetic
     Microsaccades
If VR tracking/render distorts, the world appears "swimmy."

Or real world can appear swimmy - take off glasses.
Notes

1. Saccades
   - rapid "jerks" lasting < 45 ms
   - Example: Looking over a face, reading
   - Improvement:
     - Saccadic masking hides the motion intervals from the brain

2. Smooth pursuit
   - Track a moving visual signal
   - Example: Moving tennis ball, car
   - <30°/s, otherwise saccade added
     - Reduces motion blur (slow photoreceptors)

3. Vestibulo-ocular reflex
   - By-passes brain (≤ 100 ms, fastest reflex in body)
     - Keeps inner stability
   - Corrects rotations for 6DOF head motion
     - VOR gain adaptation

4. Opto-Kinetic
   - Alternating light pass alters saccade
     - Example: Watch a passing train

5. Convergence/Divergence
   - Stereovision

Note: Accommodation happens too, but not in VR at present.
Result: Very hard to design VR set display that is "equivalent" to

2

Coordination

Note: Damn to program for part of VR set
6. Microsaccades
   - 8 to 2 degrees motion
   - Purpose: Partially debated, unknown
   - Might cause illusory motion
     - Recall slide earlier

Regarding VR and eye motion:

1. Not looking through optical axis (even if eyeball centered)

   ![Eye diagram]

   Problem: Focus distortion, maybe nausea

2. VOR gain adaptation - swaying effect in VR and real world

3. Complicated interaction between display & photoreceptors
   a. Pixels switch color, intensity at same rate (e.g., LCD, PGG, OLED)
   b. Frames might be off (black), frame rates 60, 75, 90 FPS
   c. Photoreceptors slow 0.1-0.25 response time
   d. All eye moves shift image on retina!
4. Vergence usually coupled with accommodation
   - Opposite in VR ⇒ always in fovea

   - Switch to slides on visual pathways
   - Based on Maller Ch 7

   Show
   - Retina
   - Cross section

   ON bipolar ⇒ cone and rod ⇒ connected to amacrine cells
   OFF bipolar ⇒ cone only ⇒ connected to ganglion

   Horizontal cells: lateral inhibition

   Pathways: Ganglion ⇒ LGN/Thalamus post ⇒ visual cortex

   Make alien circuit joke
- Deceptive field model → funneling compression
- Opponency → important idea
  → spectral vs spatial

**Ganglion Cells**

<table>
<thead>
<tr>
<th>Midget</th>
<th>Parvcell</th>
<th>Bipolar</th>
</tr>
</thead>
<tbody>
<tr>
<td>70-10%</td>
<td>0-10%</td>
<td>&lt;10%</td>
</tr>
<tr>
<td>Photopic</td>
<td>P&amp;Scotopic</td>
<td>Photopic</td>
</tr>
<tr>
<td>Opposed</td>
<td>Opposed</td>
<td>Nonopposed</td>
</tr>
<tr>
<td>Red-green (cortical)</td>
<td>Yellow (adjacent)</td>
<td>Blue-yellow</td>
</tr>
<tr>
<td>Transient</td>
<td>Sustained</td>
<td>transient</td>
</tr>
</tbody>
</table>

**ON-Cells**

**OFF-Cells**

- Ganglion Image
- Path to visual cortex
- Orientation Tuning

Visual Cortex

End slides
Depth Perception

Depth cues:
1) Metric - vary continually with distance
2) Ordinal - ordering: near to far

Multiplicity of depth cues - not just stereo
(think about Google Streetview)

1. Retinal image size

2. Height in visual field

3. Texture gradients
   - Disparity -> left and right eye images are shifted
   - Parallax -> images
   - Perspective: Compression
   - Harrier

*6* (L6 evening cruise TV)
4.) Image blur
Blue parts of image to appear outside of focal plane/depth

5.) Atmospheric perspective
Hazy mountains in distance

6.) Accommodation
For close objects and people under 50

7.) Motion parallax
Optical Flow

8.) Shadows/Shading

9.) Inteposition

10.) Stereo cues
-Vergence angle

-Stereoscopic disparity → left and right eye images are shifted
-Disparities - multiple images
-Thinking of IPD in VR!

-Horopter
(LG explains curved TV)
Combination of Cues

Think Bayesian

Scale perception vs depth perception
- Reach and grab something
- They acted each other in VR
- Context, IPD important

Motion Perception

Purposes:
- Segmentation/Segregation:
  Extract moving body from background
- Extract 3D structure from motion
  (Spin the chair around)
- Visual guidance of action
  - Manipulation: grab a cup
  - Hand-eye coordination

Note: Hard to do because of south plant
Neural circuitry for motion

O → (Image on retina)
  
A, B → Neighboring detector neurons
  
C neuron activated for (A timeshifted) x (B current)
  "was at A, now at B"

Possible to confuse with wagon-wheel effect (video)

Object Motion vs Observer Motion

Retinal image moves same way in either case

Object moves, observer and eye fixed

Note: Hard to do because of smooth pursuit
The brain uses more information to distinguish
- Saccadic masking/suppression - suppresses motion detectors
- Eye movement commands
- Large-scale motion - If eye moves, the whole scene moves
(Recall big swing illusion)

Optical Flow
- Track movement of features on retina (or image plane or screen)
- A vector field on image plane (or sphere)

Self Motion

Forward  Backward  Left  Right  CCW  CW
(Walk around room)
Big problem in VR: Illusion of self motion from optical flow
-vection
- Visual system & Vestibular system in disagreement

Ex: Ramp up character motion (angular)

\[
\text{Speed} \quad \text{Time} \quad \text{Speed} \quad \text{Time}
\]

Which is more comfortable?

- Small acceleration mismatch over long time
- Large acceleration mismatch over short time \( \Rightarrow \text{better} \)

Stroboscopic apparent motion

Think frames per second (FPS)

- Not exactly true in real hardware
- Pixel switching speeds
- Sequential scanout

61)
See Wikipedia: "Fringe Rate" or "Flicker-Fusion Threshold"
One ms enough for a free to register.
Up to 15 FPS, images perceived individually.

Early silent films: 16-24 FPS (hard-taxed speed)
Home movies: 16-18 FPS
Motion picture industry: 24 FPS standard

- Two-blade projectors → 48 FPS effective reduce
- Three blade → 72 FPS effective flicker
  Thomas Edison’s improvement for comfort is 48
New motion picture standard proposed: 48 (James Cameron wants)

CRT refresh rate → large monitors, peripheral vision

60 Hz - noticeable flicker
72 Hz - minimum organic reconnection
85 Hz - comfort for all

Note: Detectable vs not, but headaches vs not, and

1 2 3 Levels

Very general!

Current talk in VR industry:

60 Hz vs 75 Hz vs 90 Hz vs 120 Hz
Gear VR / DK2 / Oculus / Vive / Sony
- Google for "blu-busters"
  Everything better than 60Hz
- Abrasch blogs

[Graph: \( \text{real world "point"} \)]

### Notes

- Words because photographs need only low of exposure and hold for around 100ms
Problems with displays

Most scans cannot imagine line by line.

Motivated by original CRT - electron gun - analog

Need to keep phosphors illuminating

This causes a "wobble" when screen object moves

But not translating across retina in most cases

Recall smooth pursuit and VOR

Both supposed to keep object fixed on retina

Low persistence leads to flicker fusion problem; thus 90 Hz or higher
Another problem: LCDs have slow pixel-switching speed
   Result: Blur — see DK!

**Tracking Systems**

What do we want to track?

Think about rigid bodies

1. Head wearing HMD
2. Paws of hands
3. Eyes
4. Fingers
5. Entire body
6. Movable objects — controller, coffee cup —
   Other people in the space

For each, estimate rotation + translation
   (orientation + position)

Rotation \( R \), \( q = (a, b, c, d) \)
Translation \( (x_t, y_t, z_t) \)

\[
\begin{bmatrix}
  R & x_t \\
  y_t & 1 \\
  z_t & 0 \\
  0 & 0 & 0 & 1
\end{bmatrix}
\]
Estimating orientation (Oculus Rift DK1, Gear VR)

2D case

\[ w = \frac{d\theta}{dt} \]

Merry-go-round

At \( t = 0 \), \( \theta = \theta_0 \)
At \( t > 0 \), rotation rate is \( w \) rad/s
Thus, \( \theta(t) = \theta_0 + wt \)

What if \( w \) varies over time?

\[ \theta(t) = \theta_0 + \int_0^t w(s) ds \]

Discrete-time approximation:

\[ \Delta t = 1 \text{s} \]

\[ w_i = w(i-1)\Delta t \]

\[ \theta = \theta_0 + \sum_{i=1}^{k} \Delta \theta_i = \theta_0 + \sum_{i=1}^{k} w_i \Delta t = \hat{\theta} \] (Euler integration)

Now suppose a sensor estimates \( w_i \) every \( \Delta t \)

\[ \hat{\theta} = \theta_0 + \sum_{i=1}^{k} \hat{w}_i \Delta t \]

estimated rotation

sensor reading from 1D gyroscope

[Continue in next notebook]
Blogger Beau Cronin

Oculus BPG: "O BPG may be the most substantial thing ever written on applied sensory neuroscience"
Oculus Best Practices Guide

Goals:
1. Oculomotor comfort - avoid eye strain
2. Bodily comfort - prevent feelings of disorientation, nausea
3. Positive user exposure - fun, immersive engagement
4. Minimal aftereffects - visual-motor functioning after use

Rendering:
- Keep 1:1 correspondence between physical and virtual world
  head rotation & translations
- Don't make fixed splash screen or images w/o tracking
- L & R eye image differ only by viewpoint
- Supersampling, anti-aliasing - worst at edge pixels

Latency:
- Target ~20ms for sensor-fusion reading to rendering
- Keep latency constant as possible
- Use predictive tracking

Tracking & report:
- Never let tracking thread starve
- Don't make large rotating live (e.g. horizon)

Potential tracking:
- Mindful of user poking their heads anywhere uncontrollably
- Better to fade out when limit (e.g. wall) reached
Accelerations
- Make mismatched (physical vs. virtual) as short as possible
- Acceleration could be constant speed! Move along curve
- Have user control acceleration

Move speed
- Real world 1.4 m/s walking
- Teleporting OK, but try to preserve orientation
- Don't make them look one way while moving another

Cameras
- Don't do zoom effects - bad optical feedback
- No head bobbing

Managing Sim-Sickness
- Find fresh users, you are not good judge of your own
- Note that people vary in their tolerance
- Make warnings or options to vary intensity of problem

Degree of Stereoscopic Depth
- Don't make things too close
- Don't make virtual camera R/L distance below

User Interface
- Make UI sit 2-3 meters in front of user
- Keep in middle 1/3 of screen
- Cautious with head move to select menu
- Embed info into world objects
- Draw crosses at object distance
Avatar Control
- Acceleration
- Avoid keyboard options if possible
- Head movement for control: Dumbo the Elephant

Sound
- Differ between headphones and speakers (600 ms transform)
- NPC (non-player character) speech should not be "screaming"
- Keep position (and orientation) in mind

Contact
- Units: 1 meter in physical world = 1 meter in Unity
- Most objects between 0.75 - 3.5 m away
- Consider rendering avatar body, but beware of monstrosity (e.g., sitting vs. standing)
- Use monocular depth cues
- Avoid strafing, backstepping, spinning

Health & Safety
- Avoid 1-30 Hz flicker → photosensitive seizures
- Avoid high contrast, high-spatial-frequency patterns
3D orientation case

\[ \text{Axis angle } (\vec{v}, \theta) \]

3 axis gyroscope measures

\[ \vec{w} = (w_x, w_y, w_z) \]

If \( w \) is constant over \( \Delta t \), then rotation is

by \( \Delta \theta = \frac{\| \vec{w} \| \Delta t}{\left( w_x^2 + w_y^2 + w_z^2 \right)^{\frac{1}{2}}} \)

and axis \( \vec{v} = \frac{1}{\| \vec{w} \|} (w_x, w_y, w_z) \)

during for \( \Delta t \).

Let \( q_i \) be quaternion representation of \( (\vec{v}_i, \Delta \theta_i) \)

\[ q_i = \text{quat}(\vec{v}_i, \Delta \theta_i) \]
Recursively:

\( \theta_{\text{current}} = \Delta \theta_k + \theta_{\text{prev}} = \omega_k \Delta t + \theta_{\text{prev}} \)

\( \hat{q}_{\text{current}} = \Delta \hat{q}_k \circ \hat{q}_{\text{prev}} = \text{Quat}(\hat{v}_k, \Delta \hat{\theta}_k) \circ \hat{q}_{\text{prev}} \)
Integrating sensor readings to estimate orientation

Recall 2D: \[ \hat{\theta} = \theta_0 + \sum_{i=1}^{k} \Delta \hat{\theta}_i = \theta_0 + \Delta \hat{\theta}_1 + \Delta \hat{\theta}_2 + \cdots + \Delta \hat{\theta}_k \]

\[ \Delta \hat{\theta}_i = \hat{\omega}_i \Delta t \]

Now 3D:

\[ \hat{q} = \Delta \hat{q}_k \circ \Delta \hat{q}_{k-1} \circ \cdots \circ \Delta \hat{q}_2 \circ \Delta \hat{q}_1 \circ \hat{q}_0 \]

\[ \text{Each } \Delta \hat{q}_i = \text{Quat}(\hat{\omega}_i, \Delta \hat{\theta}_i) \]

\[ \hat{\omega}_i = \frac{\|\Delta \hat{q}_i\|}{\|\Delta \hat{q}_i\|_2} \]

Problem: Drift errors

2D: \[ d_k = \theta_k - \hat{\theta}_k \]

3D: \[ d_k = \hat{q}_k \circ \hat{q}_k^{-1} \]

The error is a 3D rotation, with yaw, pitch, roll components.
Note: Drift corrections can come from sensors at very low frequency.
Let pitch + roll error be called **tilt error**.

Correcting for drift errors:

1. Use another sensor to provide world reference
2. Gradually apply corrections
   - Fast enough to fully compensate
   - Slow enough to avoid sim. sickness

For tilt error: Need a gravity or "up" sensor
For yaw error: Need a "compass"

Use accelerometer to correct tilt error

![Coordinate frame is estimated orientation](image)

\[ \hat{\mathbf{a}} = (\hat{a}_x, \hat{a}_y, \hat{a}_z) \]

\[ \overrightarrow{\text{tilt axis in } XZ \text{ plane}} = (-\hat{a}_z, 0, \hat{a}_x) \]

Need to rotate by \( \phi \) about \( \overrightarrow{\text{tilt axis}} \) to fix.
2D case

\[ \hat{\theta} \text{ corrob} = \Theta + \alpha \phi \]

error measured in \( \Theta \)
Simple way to correct: Complementary filter

Gain coefficient $\alpha > 0 \quad \alpha \approx 0$ (example $\alpha = 0.0001$)

In each step, charge every step

\[ \hat{\theta}_{\text{corrected}} = \text{Quat}(\hat{\mathbf{V}}_{\text{tilt}}, \alpha \phi) \cdot \hat{\theta} \]

$\alpha$ needs to be large enough to cancel error, but small enough to avoid sim. sickness and perceived jitter.

Problem: Accelerator measures linear acceleration of sensor

vector sum

Use heuristics to detect when "not moving"

Example:

\[ \| \hat{\mathbf{a}} \| \approx 9.8 \]

\[ \text{Can make mistakes. Accelerate head down!} \]
Use magnetometer to correct yaw error:

Similar to tilt correction:
- Calculate reference error
- Gradually apply using complementary filter

Problem:
- Vector sum of Earth's field, building field, board field
- Calibration hard
- Field might vary over time and position
- MRI iron bias

Accuracy ≈ 5 degrees

See Mahony, 2008: Complementary filters on SO(3)

Lavalle et al., ICRA 2014
Estimating position + orientation (Rift DK2, Sony Morpheus, Valve Vive)

- IMU (gyro+accel) not enough
  - Drift is too fast from double-integrating &
  - No way to detect drift error

Solutions:
1) Generate non-contact magnetic or EM field
   - ReapHydra, STEM Sense
   - UWB Radio

2) Visibility or line-of-sight methods

Camera arrangements
- On headset
- In world

inside-out
outside-in
Pinhole camera
Perspective projection

Features in an image

Features:
1) Natural - hard computer vision
   - Extract & maintain from natural scenes
   - Remove moving objects
   - Reliability low

2) Artificial - trivial computer vision: Blob detection
   - QR tags, retroreflective markers, LEDs, laser projector
   - Can stay in IR spectrum (invisible to humans)

Retroreflective features

Mocap systems
VisaQ OptiTrack

Bright IR LEDs

Power \sim \frac{1}{d^2} \quad \text{and } d \text{ is doubled}
Determine position & orientation of triangle from features in image.

DOF analysis:

- Start with 6 DOFs (rigid body)
- Each feature setting 2 DOFs (xy coords in image)

P1P: Object can rotate freely (3) + normal for (1) = 4
P2P: Object can swing around edge between two features (1) and roll (pitch) while changing distance (1) = 2
P3P: No freedom left

Illustrate with real object
IR LED features - small, bright

PnP problem: Determine rigid-body transform from identified, observed features on a rigid body

P3P's

A system of polynomial equations can be obtained. Generally, 8 solutions, but only 4 in front of camera (See Wu, Hu, 2006 PnP Problem Revisited)

P3P, P3P, P3P — PnP

multiple solutions, coplanarity cause difficulty
\[ \hat{x}_{cor} = \hat{x} + \alpha_1 \Delta \dot{x} + \frac{1}{2} \hat{a}^2 \Delta t + \hat{v} \Delta t \]
\[ \hat{v}_{cor} = \hat{v} + \alpha_2 \Delta \dot{v} + \hat{a} \Delta t \]
Problem:
- Noise, pixel quantization error
- Features entering or leaving FOV

Incremental PnP
- Assume transform known already
- Can't travel far in < 20 ms
- Perturb relative transform to match image
- Optimization, randomized gradient descent
- Accelerometer helps improve starting point for optimization

Sensor Fusion:
- Camera can drift, correct orientation
- Accelerometer can help with position estimation
- Use complementary filter (2nd order)
  \[ \dot{x} = U \]
  Equivalent to Kalman Filter for this problem (Higgins, 1975)

- Double integrator dynamical system model
  \[ \dot{x} = U \]
  \[ \dot{U} = \alpha \]
  \( \alpha \) coefficients for both position and velocity
(Should refer to mate rotation <CW)
Pharology

**Lighthouse approach**  (Valve/HTC Vive)

Why not bypass taking an image and scanning for features?

- Computationally costly (could do in hardware)
- Limited light per pixel
- More robust, not necessarily better

Note the mathematical/visibility equivalence:

- Identification of features is easy
- Need to know rotation rate (100Hz?)
- No image processing
- Little power dissipation because laser
- Does have many parts
Need horizontal and vertical beams

Problems:
- Two beams hit same sensor at same time
- Maintaining timing to avoid angular drift

Apparent solution: Pulse a bright flash (in IR) once in a while
- Received by all sensors at same time
- Revert/sync timing clocks
Rendering

Recall chain of transformations:
Transform point in body frame to world to screen space (wrt ul) (pixel location)

\[ T = T_{rp} T_{can} T_{eye} T_{rb} \]

Next problem: How to set the RGB value of each pixel?

Two approaches:

<table>
<thead>
<tr>
<th>Image-order rendering</th>
<th>Object-order rendering</th>
</tr>
</thead>
<tbody>
<tr>
<td>pixel-by-pixel</td>
<td>triangle-by-triangle (or primitive)</td>
</tr>
<tr>
<td>ray tracing</td>
<td>rasterization</td>
</tr>
<tr>
<td>scene-to-index traversal</td>
<td>used by GPUs</td>
</tr>
<tr>
<td>usually slower</td>
<td>usually faster</td>
</tr>
</tbody>
</table>

Simplistic Lambertian - illumination proportional to cosine of
Ray tracing stages:
1) Ray generation
2) Ray intersection
3) Shading

#1 is a ray of perspective projection

#2 must find first intersection from focal point

Intersection test

#3 Shading

Simplest: Lambertian - illumination proportional to cosine of

\[ L = k_d \cdot I \cdot \max(0, n \cdot \ell) \]

Unit vector
When light visible in mirror.

\[ \theta \] is the angle of incidence, \[ \theta' \] is the angle of reflection. 

Director = \text{normal} \quad (\text{Law of Reflection})
diffuse, flat, matte surface

Handling highlights: Blinn-Phong shading

Unit vectors

\[ \mathbf{h} = \frac{\mathbf{v} + \mathbf{l}}{||\mathbf{v} + \mathbf{l}||} \]

\[ L = k_d I_d \max(0, n \cdot d) + k_s I_s \max(0, n \cdot h)^p \]

\[ p = \begin{cases} 10 & \text{eggshell} \\ 100 & \text{muddy shiny} \\ 1000 & \text{glossy} \\ 10000 & \text{nearly mirror-like} \end{cases} \]

What about ambient light?
Add \( k_a I_a \) to \( L \)

What about multiple light sources?
Add diffuse and specular for each one.

What about multiple reflections?

Global illumination – very expensive
(visibility with mirror)

Transparency? Recall absorption, refraction, reflection, picture from optics
More general model for shading: BRDF

B. directional Reflectance Distribution Function

\[ f(\theta_i, \phi_i, \theta_r, \phi_r) = \frac{\text{radiance}}{\text{irradiance}} \]

Reflection symmetry \( f(\theta_i, \theta_r, \phi_i - \phi_r) \) — for physical consistency

Lambertian: Special case with \( f \) constant

Problem with VR: Stereo eye view

Need to do twice the work for shiny surfaces
Useful adaptation:

- Z buffer from light source perspective can generate shadow map (shadow parts of object obscured by light)
Rasterization

Most common - GPUs
Object order rendering - triangle by triangle

Fill in pixels by sampling midpoint

Problem: Aliasing

Depth ordering problem: Which triangle is in front?

Painters algorithm: Sort by furthest to closest; paint far to near order

Problem: depth cycle - segment triangles

Problem: Sorting too costly

Z-buffer: Store depth value (z) at each pixel.

Render triangles in arbitrary order.

Render pixel only if new z value is less than old one at that pixel

Clipping: Remove triangles (or parts of triangles) that are behind the eye, or too close

(Vertical clipping plane)

VR problem: What?
Efficiency: Culling - Eliminate unnecessary rasterizations

Backface: Remove triangles on "back" of object (not in view)

View Frustum: Remove triangles not in viewing frustum

Propagating color, RGB, other attributes efficiently:

Barycentric coordinates

\[ p = \alpha_1 p_1 + \alpha_2 p_2 + \alpha_3 p_3 \quad \text{For: } 0 \leq \alpha_1, \alpha_2, \alpha_3 \leq 1 \quad \text{and} \quad \alpha_1 + \alpha_2 + \alpha_3 = 1 \]

\[ (\alpha_i = 0 \rightarrow \text{an edge} \quad \alpha_i, \alpha_j = 0 \rightarrow \text{on vertex}) \]

Efficient interpolation

\[ R = \alpha_1 R_1 + \alpha_2 R_2 + \alpha_3 R_3 \]
\[ G = \alpha_1 G_1 + \alpha_2 G_2 + \alpha_3 G_3 \]
\[ B = \alpha_1 B_1 + \alpha_2 B_2 + \alpha_3 B_3 \]

\[ \uparrow \quad \uparrow \quad \uparrow \]
\[ \alpha_1 p_1 \quad \alpha_2 p_2 \quad \alpha_3 p_3 \]
Regular patterns repeat a specific wavelength and phase issue at boundaries.
Other mappings over triangles:

- Texture mapping (like painting a picture onto surface)

- Look up mipmapping for scaling textures (a form of antialiasing)

- Bump mapping: Map a pattern of normals to surface, make it look rougher

- Normal mapping: Use surface normals from high-resolution surface, make plane look spherical (for example)

- Use Lambertian shading to make surface look curved.
Main differences:
1) Head motion
2) Stereo
3) Low resolution
Problems with rendering for VR:

Shading:
- Highlights (speculars) need stereo perspective = double work
  - Texture maps look like painted cardboard
  - Bump/normal maps look fake

Aliasing:
- Resolution looks low
  - Motion turns "staircase" into "escalator"

- Stereo causes mismatched stairs per edge

Render target:
- Unusual shape
- Use stencil buffer in GPU
  - Correct for optical distortion due to lenses
    pincushion-barrel annihilation
Latency - GPU pipeline optimized for triangle throughput, not latency.
- Latency compensation by post-rendering image warp (time warp)
  Visibility problems for 6 DOF case

Geometry errors - Thin objects look fake or implausible
  (leaves in Tuscany)
- Holes, isolated points more obvious
  (think about SLAM data)

- Chromatic aberration?
- Head in walls?
  (you can move your head into/through obstacles)
Antialiasing - Get rid of "jaggies"

Supersampling

![Diagram showing standard, higher resolution, and deterministic random pattern]

Average the colors for the final pixel value

Common special case (hack): MSAA - multisample antialiasing

Only supersample depth (z) and stencil values (in or out of triangle)

Compute polynomials barrel distortion:

\[ f(r) = 1 + cr^3 + c_5r^5 + c_7r^7 \]

Odd powers not needed (theory & optics)

c, c_i chosen by hand or laser rearranged

Apply \( f(r) \) to every pixel in barrel warp image

Pencil Line - barrel annihilation
Distortion Shading for High FOV

- Use polar coordinates from optical center
- Assume rotation invariance for distortion ($\theta$ does not matter)
- Radially symmetric

$$(r, \theta) \rightarrow (F(r), \theta)$$

Composite polynomial barrel distortion

$$f(r) = 1 + C_1 r^2 + C_2 r^4 + C_3 r^6 + \ldots$$

Odd powers not needed (theory of optics)

$c_1, c_2, c_3$ chosen by hand or laser measurements

Apply $f(r)$ to every pixel to barrel warp image

Pincushion-barrel annihilation!
Also, Darsa, Costa, Varshney, 1997

(\text{Consider} \; \text{boundary/particle} \; \text{interaction})

(f_{\text{rel}} = 1 + C_{\text{rel}} + C_{\text{rel}}^2 + C_{\text{rel}}^3)

(f_{\text{rel}} = 1 + C_{\text{rel}}^2 + C_{\text{rel}}^3 + C_{\text{rel}}^4)

\text{Consider \; your \; update \; (fresh \; \text{water})}

\text{C.e.p. \; choosing \; a \; point \; at \; force \; maximum}

\text{Apply \; f(t) \; to \; each \; bond \; of \; polar \; and \; non-polar}

\text{bonds. \; \text{Quadrupole \; approximation}}
Post-rendering image warp (time warp)

Mark McMillan, Bishop 1997
Carmack 2013

Four steps:
1) Read latest head pose (orientation & position)
   How long? 10-50ms?
2) Render scene into buffer using pose for viewpoint
3) Read latest head pose
4) Adjust rendered output to fake the newer viewpoint

Image shifting accounts for small pitch and yaw:
   (vertical) (horizontal)
Image rotation accounts for small roll.

What about position?
Some faking possible:

- Rescaling image to account for depth (z) change
- Shift image to account for x, y change

Problems:
- Perspective projection not quite right as z changes (small change OK)
- Visibility problems

\[ \text{No data for this!} \]

Another use: Improve frame rate by inserting faked viewpoints

\[ \text{Faked frames: Adjust viewpoint only} \]

Properly rendered frames

Note: We are not doing lenses, images
Rarefaction

Compressio

Revised on 07/19/92

Legend: 1 Red

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.

Diagram for initial flow.
Audio for VR

Real world

VR

Sound waves: Similar to light except:
- Fluctuating air pressure (instead of EM/photons) \( \frac{4kN}{m^2} \) Hz
- Frequency only 20Hz to 20000Hz (instead of 400-700 nm)
- Speed 343m/s in air (not \( c = 3 \times 10^8 \) m/s)

Interacting with other media: Just like light

\[ \frac{1}{V_1} \]

Absorption

Transmission

Reflection

Note: We are not doing lenses, images
Frequency spectrum: Just like light

\[ \text{power} \begin{array}{c}
\text{voltage} \\
\text{frequency}
\end{array} \]

Pure tone

\[ \text{power} \begin{array}{c}
\text{voltage} \\
\text{frequency}
\end{array} \]

Complex wave

Think filtering, Fourier transforms

Human auditory system

(Show slides for pictures - dr 24)

Explain components: pinna, tympanic membrane, cochlea, organ of Corti, hair cells, --
Perception of Sound

Recall Steven's power law: \( n \rightarrow cm^x \)

- \( n \): magnitude of stimulus
- \( m \): perceived magnitude

Loudness at 3000Hz: \( x \approx 0.67 \)

Loudness as function of frequency

\[ \text{Loudness} \approx \text{Fixed dB at } 1\text{kHz} \]

Fig 5.2 in Malher

Loudness adaptation

Also, loudness above noise threshold
φ and θ are known for visual because of position or retina at eye rotation.
Pitch perception (recall Weber's law and JND)

![Graph showing the relationship between pitch and threshold with frequency on the x-axis and threshold on the y-axis.]

**Auditory localization**

Where is sound coming from? (Similar to depth perception in vision)

3 Coordinates:
1. Horizontal plane - azimuth
2. Vertical plane - elevation
3. Distance

Minimum audible angle (MAA) - JND

- ~1° below 1000 Hz, straight ahead
- ~5° to the side
- Terrible around 1500-1600 Hz on side
Audio illusion:

Preference of main/first wave suppression of later wave (evidenced)

only clear one

both heard

(SMC - (AAM): shows results)

Some thoughts:
Monaural cues
1) Pinna and external ear canal shape
2) Intensity decreasing by inverse square law
3) Spectrum of sounds
   Low frequency travels further; distant thunder is only low frequency rumble
4) Direct vs. reverberation energy
   (When there are reflecting surfaces)

Binaural cues
ILD: Inter-aural Level Difference — acoustic shadow of ear ("beef face")
(ILD: Inter-aural Time Difference — arrival time; distance between ears
≈ 14 cm
In sensing: Multilateration from TDOA — time difference of arrival
(BECCA system, WWII) Modern GPS)

\[ \Delta t^2 = \frac{(d_x - d_y)^2}{s^2} \]

\[ \text{Speed of sound } \approx 340 \text{ m/s} \]

Not right (only ear speaks)

Hyperboloidal surface! Cone of Confusion
Google for "audio rendering"  
UNC talk Anish Choudak (2009)

[Handwritten notes and diagrams]

Projectivity surface: due to Cauchy
Sound rendering

Four steps

1. Modeling
   - Geometric models - walls, obstacles
   - Acoustic material properties - absorption, reflection, refraction
   - Sound sources - point source, parallel wave source, loudness
   - Lower resolution than for graphics (not like pixels)
   - Simplified

2. Propagation
   - Reflection: specular, scattering
   - Diffraction
   - Refraction
   - Doppler effect
   - Attenuation

Computational approaches

Numerical vs Combinatorial

Visibility, ray-tracing

Solutions to Helmholtz wave Eq.
PDE - PEM, etc.

Accurate but expensive

Similar to graphics
Estimate wave propagation

Fast, approximate
Kind of like applying barrel distortion, at the end for visual rendering
3. Rendering

Use head position and orientation to determine pressure signal at right and left ears.

Problems: Account for scattering due to:
- Outer ears (pinna)
- Head
- Whole body (clothing?)

HRTF (Head-Related Transfer Function)

$H(\omega) = \frac{\text{Output}(\omega)}{\text{Input}(\omega)}$  Adjust for scattering using linear transfer function (in frequency $\omega$ domain)

- Depends on individual ear, head, body shape
- Depends on $x, y, z$ of audio source
- Can estimate by recording in anechoic chamber; vary impulse responses

$H(\omega, \theta, \phi, d)$

$H(\omega, \theta, \phi) \approx$ far-field

- Quasi-angles, distance
- Could use "far field" and distance is "at infinity"
4. Display

Especially one "pixel" display per ear
Time is more important than space
Frequency, phase, amplitudes

Surround sound vs. headphones vs earbuds
(Fixed in space)

What about head position & orientation?

Like CAVE

Problems for developers:

1. How much modeling needed? How to accomplish?
   - Need good middleware

2. How to evaluate correctness or accuracy?
   - Ears not as sensitive as eyes (and eyes already play tricks)
   - HRTFs are person-dependent - like eyeglasses

3. Computation cost
3D User Interfaces
Bowen et al., 2005

Include vehicles, grappling, jumping high

Visualization of oneself:
- Torso
- Arms, legs, hands
- Face in mirror

Problems to be addressed
- Depth of wall, floor, ceiling
- Shadows
- Non-photorealistic rendering
- Predicting how people move
- Understanding objects in the real world
Interfaces for VR

Some categories:

1. Locomotion - moving oneself from place to place
   Issues: Speed, vestibular mismatch, getting lost

2. Manipulation - touching, feeling, grabbing, carrying, discarding objects
   Issues: Selection, force feedback, grasping realism, depth perception

3. System Control - menus, windows, text
   Issues: Readability, comfort, speed

4. Social - chat, testing
   Issues: Face covered, body language, spatial arrangement, uncanny valley

5. Others?? Text editing? Street Fighting? Cinematography?

Simulating ANY interface that exists in the physical world!
- Stationary bicycle
- Omnidirectional treadmill

Bruder, Lukas, Steinicke, 2015

\[
\begin{align*}
\text{Moment of force} & \quad \text{moment about axis} \\
\end{align*}
\]
**Locomotion** - Traveling in VR

All physical vs All virtual

Value Vive Focus HMD FPS limited

Walk & look around

All looking & walking done by controller

How to apply it:

Recall chain of transitions: Tip → Tran → Teye → Tb

Consider how Teye is forced from "look at" to get \( \hat{x}, \hat{y}, \hat{z}, e \)

All physical: Teye entirely determined by head tracking

All virtual: Teye, \( \hat{e} \), controller

Cold trick for all physical walking in unbounded virtual worlds:

Redirected walking (Razaqpur, Kahn, Whitton, 2001)

- Distort rotations, bad straight walks
- Keep forcing user into cave
If a lens is tilted, then the optical center ≠ mass center.
For HMD FPS:

Look at frame transformed by $T_{\text{Vehicle}}$

\[
T_{\text{eye}} = (T_{\text{vehicle}})^{-1} = T_{\text{vehicle}}^{-1} T_{\text{track}}^{-1}
\]

Particulars:
- Center of rotation important for Vehicle

For virtual walking in a fixed chair, Vehicle is equivalent to 2D rigid body transform

- $x$, $z$ offset, and yaw direction (tors)
- For free-flying ship, Vehicle is full 3D rigid body transform

#1

#2

#3

#4

#5

#6

#7

#8

#9

#10
Problems with vestibular mismatch (vection)

- Linear acceleration in VR causes perceived acceleration from visual circular motion.
- Note: Arrows should indicate the acceleration vectors.
- Vestibular organ knows there is no motion.

Two strategies for initiating a walk:

- Strategy 1: Linear increase in velocity.
- Strategy 2: Linear increase in acceleration.

Strategy 2 is more comfortable because the brain rejects the brief, large mismatch as outliers.

Idea: Do the same for rotations (Windlands).
- Discrete jumps for orientation change.
- Too large jumps $\rightarrow$ confusion about new direction.
Angular motion, even when constant, induces rotation.

Circular motion:

\[ \mathbf{v} = \omega \times \mathbf{r} \]

\[ \mathbf{a} = \omega \times \mathbf{v} = \omega \times (\omega \times \mathbf{r}) \]

Further problem: Even motion at constant velocity (straightline, constant speed) is uncomfortable. Why?

Is continuous motion important?

- Depends on application - usually not important!
- Teleporting - good for large areas, possible location confusion
- VR Cinema - pick your seat from seating chart
- Use a map or WIM (World In Miniature)

Issue: Wayfinding?
- Learning a virtual city or forest
- Visual landmarks important - like real life!
Manipulation

Factors

1) Selection: Distance and direction to object
   - Size
   - Clutter around object
   - Occlusion (may be partial)

2) Positioning: Distance and direction of initial goal
   - Positioning precision required?
   - Does goal have basin of attraction?

3) Rotation: Amount and precision of rotation
   - All 3 DOFs?

Input device categories:

1) Metric - Motions are tracked through space: consider DOFs
   - Desk mouse, air mouse (gyro, 3D), Leap Motion, gloves

2) Switch - Motion induced by pressing, pushing
   - Joystick, buttons

Even Atari 2600 (1977) had both: Metric was paddle & Switch was joystick

Which one is artificial steering wheel?

Could be either, etc.
Note that both have **scaling issues over space and/or time**

- How far to make move?
- How long to hold button?
- How much to rotate in a top of joystick?

- Perceptual issue: Learnable motor program
- Precision issue: Can a high-precision task be accomplished?

  - Note: No redundant mismatch with scaling for manipulation!

Selection idea: Do ray casting to point of objects - like laser pointer

Manipulation idea: Make a virtual hand on an extensible arm

Final advice: Reduce DOFs as much as possible in task

General issue: Haptic feedback

- Rumble
- Pressure/force

Gloves

Manipulator robot hand

3D printed objects

**Important application:** Medical simulation or tele-surgery
See Oculus B2G and Other "Shark".  

Bouma et al
System Control

Sending commands to the application

Most common: Graphical menu
(Other: Voice command, gesture command)

How to select? Head tracking, controller.

Guidelines:

- Embed menu in virtual world—like a billboard
  (Do not attach to force in first person)
- Lazily pull menu into looking direction in 6K
- Distance in virtual world should be 2-3 meters away
- Fit inside middle 1/3 of viewing area—avoid head needs to read
- Embed into the world—ammo could appear on gun

What about text entry?

- Keyboard difficult
- Split keyboard with separator
  (Or): Charcoal keyboard
Social Interaction

Virtual communities - sociology
- Book by Howard Rheingold
  - Plato, chatrooms, email lists, Usenet, MUDs
- Multiuser dungeon

Special case:
  - Virtual worlds

Science fiction: Snowcrash, Ready Player One, Neuromancer?

MMORPGs: Massively multiplayer online role-playing games
  - Teamwork

Second Life (Oasis Simulator)
  - An entire economy formed - Linden Dollars
    - GDP $664 million in 2009
  - Like Bitcoins
  - No government - damage, theft hard to trace

Face-to-face interaction

- Avatars vs Real People
- Text vs Real Audio

- Allows fantasy, attraction
- Accessible, real

Today’s virtual classroom
- Students look at every student simultaneously because
  the real world is in the middle of their HMD
Achieving realism

- Use omnidirectional video camera
- Is person in real or placed into virtual world? Extruding problem
- What if an HMD occludes their face? Camera inside?
- Effects of delays - at least synchronize audio/video

With avatars, achieving the impossible (Transformed Social Interaction)

- Change gender, "race", species
- Change height, weight, eye color, hair color
- Empathy can be taught!!
  - Police training videos
- Look at multiple people simultaneously (Bates et al. 04)

In VR, the teacher can look at every student simultaneously because each gets a unique world on their HMD.

Also, if students are not looking at you, then their avatar fades away.
Bringing realism back into the avatar:

- Eye tracking (blinks, winks?)
  (More latency allowed than for repeated rendering)
- Face modeling, animating, and tracking
  See Paul Debevec, USC
- Beware the uncanny valley!
  People get creeped out
- Hand and body tracking
  Poor tracking worse than none at all
  Legen...dary: Kinect can be poor
  Can communicate body language

Final common interface: Any interface from physical world
  can be simulated in VR.

Hence, a VR interface can be simulated in VR!
  And on and on...

(Put on a virtual HMD while already in VR)

Like the levels in Inception