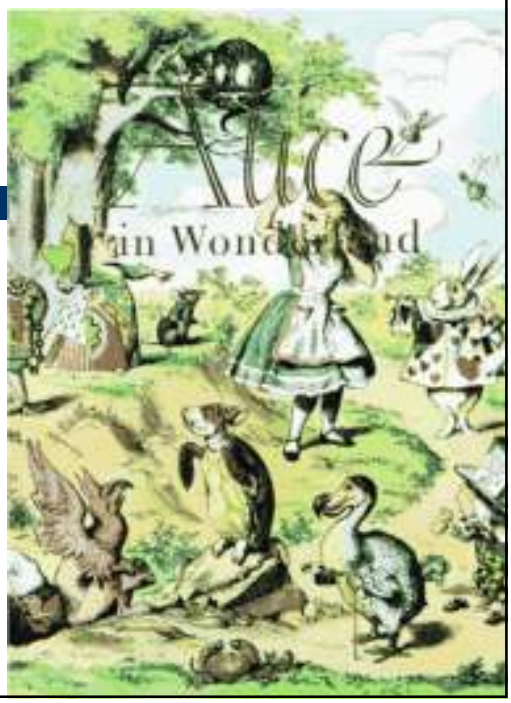
# ENSF 545
# Introduction to Virtual Reality

## Introduction

## What is Virtual Reality?




2

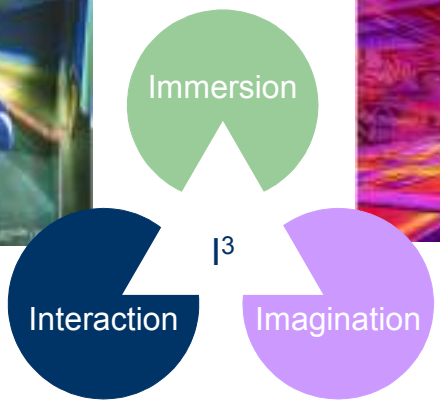
## What is VR?

- Advanced computer technologies
- High-end human-computer interface
- Real-time simulation and interactions through multi-sensory modalities
- VR == VE

3

Dr. Y. Hu

## What is VR?




* Burdea, 1993

4

Dr. Y. Hu

## Response Domain

- Reading a novel
- Movie
- 3D movie
- Game

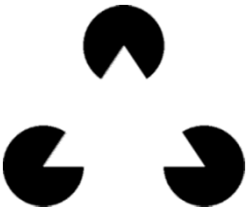
- **Immersive VR** -> + total body movement

5

Dr. Y. Hu

## Presence and Hypotheses

- Perceptual theories:
  - (Gregory:) Perceptual system selects between competing hypotheses.
  - (Stark:) Perceptual system is top-down driven.
- Hypotheses relating to the fundamental question:
  - Where am I?



## Presence ←→ Immersiveness

- A ‘good’ immersive VR (*e.g.* CAVE -- Computer Automated Virtual Environment)
- What we ‘see’ is
  where we are …
  and
  where we act

7

Dr. Y. Hu

## Perceptual Augmentation

- (Stark:) “Virtual reality works because reality is virtual.”
- Very simple cues required to trigger presence

8

Dr. Y. Hu

## Why Learning VR Technologies?

- Successes in industry
  ➔ a need for VR professionals
  - Examples – military simulations, medical rehabilitation, oil industry, games
  - Advantages – significant cost savings, saving lives, and fun ...
- Help VR development efforts in other industries
- Understand what VR can and cannot do

12

Dr. Y. Hu

## Example - Military Simulations

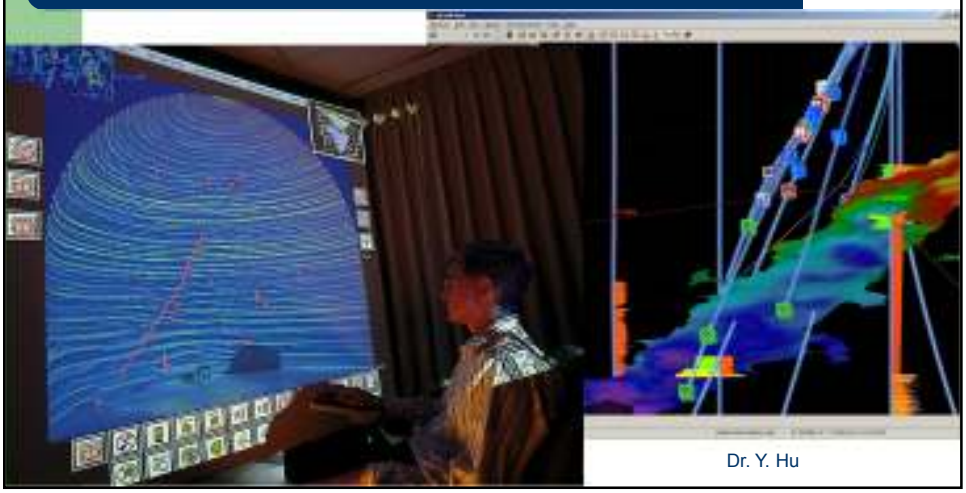
13

## Example – Medical Rehabilitation


14

Dr. Y. Hu

## Example – Oil Industry



Dr. Y. Hu

## Example – Games with WII

16

## VR History

- 1962, Morton Heilig, Sensorama
- 1981, NASA, LCD-based HMD (VIVED)
- 1985, NASA (Scott Fisher, Thomas Zimmerman, Jaron Lanier), Sensing glove
- **1989**, Jaron Lanier coined the term "**Virtual Reality**"
- 1993, IEEE organized the first VR conference in Seattle

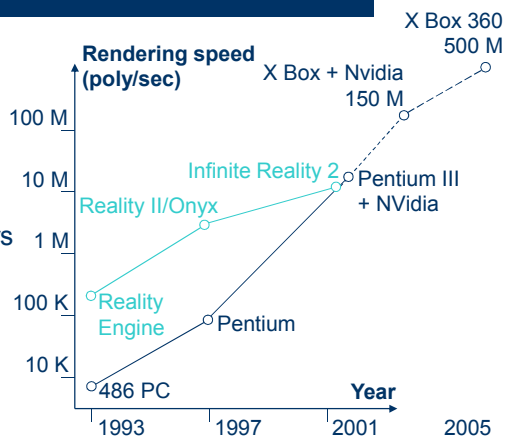
➔ VR is part of the scientific and engineering community.





## VR Industry

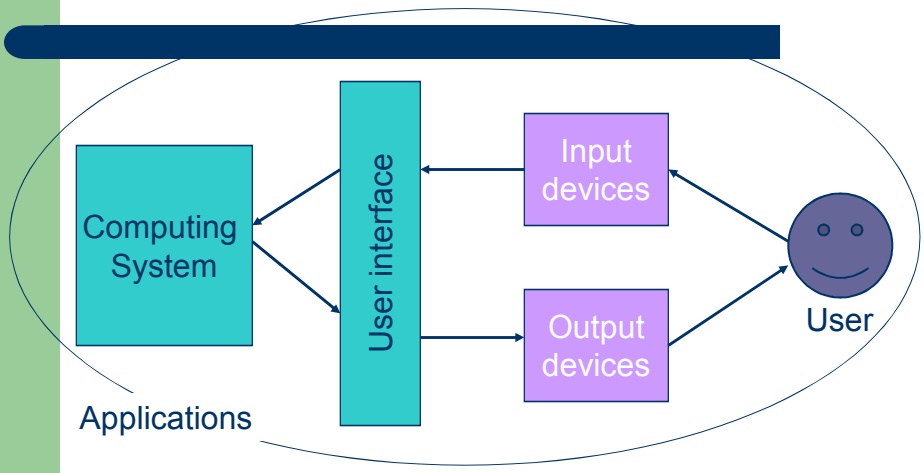
- Silicon Graphics Inc. (SGI) – Reality Engine, 1993
- Late 1990s, rebirth of VR
  - Large volume displays
  - PC graphics
  - VR I/O interfaces
- VR market:
  - $50 million, 1993
  - $3.4 billion, 2005



# ENSF 545
# Introduction to Virtual Reality

## VR Systems

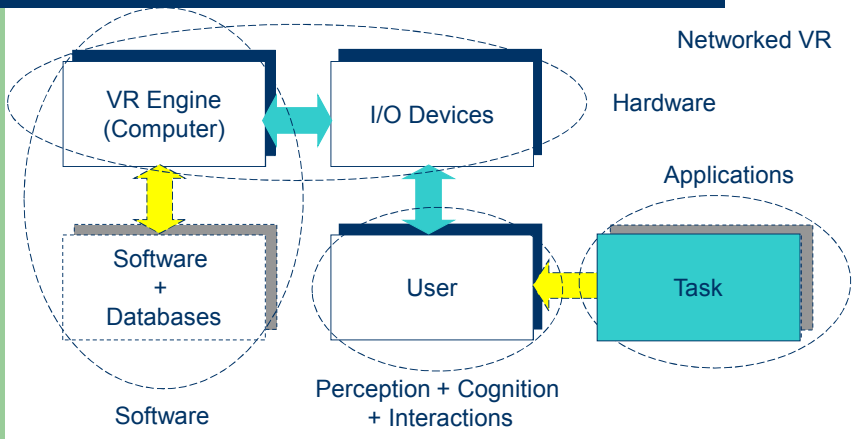
## Situation






Dr. Y. Hu

## Key Components of a VR System






Dr. Y. Hu

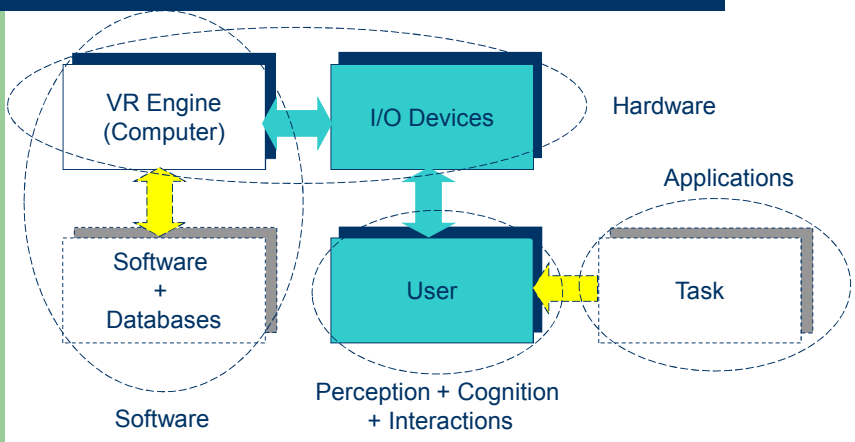
## Applications - Define

- What are the users of a specific application?
  (experiences, aptitude, motivations, needs…)
- What is the task and what is required to do it?
- What is the environment for the application?
- What are the basic perceptual, cognitive, motor, and affective capabilities of humans?





## Key Components of a VR System



## Senses vs. Devices

| | Humans | VR Systems |
|---|---|---|
| ***Vision*** | Eyes | 3D stereo graphics |
| ***Touch*** | Hand | Force feedback |
| ***Hearing*** | Ears | 3D speaker system |
| ***Smell*** | Nose | Fragrance |
| ***Taste*** | Tongue | N/A |





## Senses vs. Devices (cont'd)

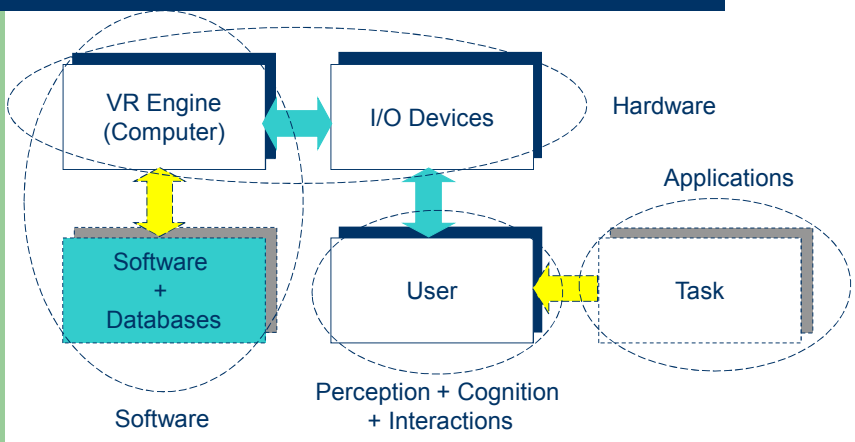
| | Humans | VR Systems |
|---|---|---|
| ***Location*** | Cues + Landmarks | Tracking device |
| ***Interaction*** | Language + Gesture | Wand, Gloves, etc. |
| ***Thermo*** | Skins | Temperature feedback gloves |

## What do You Recognize?

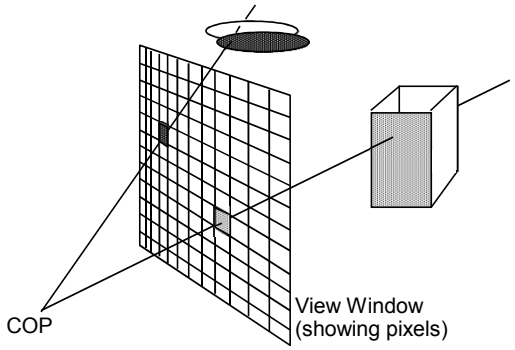




## Five Components of a VR System



## Painting Through a Window



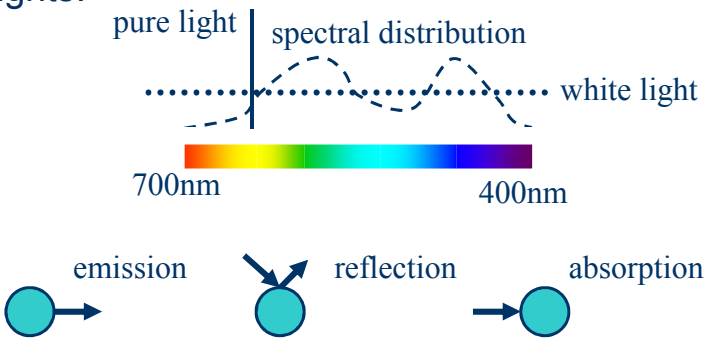

COP = Centre of Projection

34

Dr. Y. Hu

## Light in an Environment

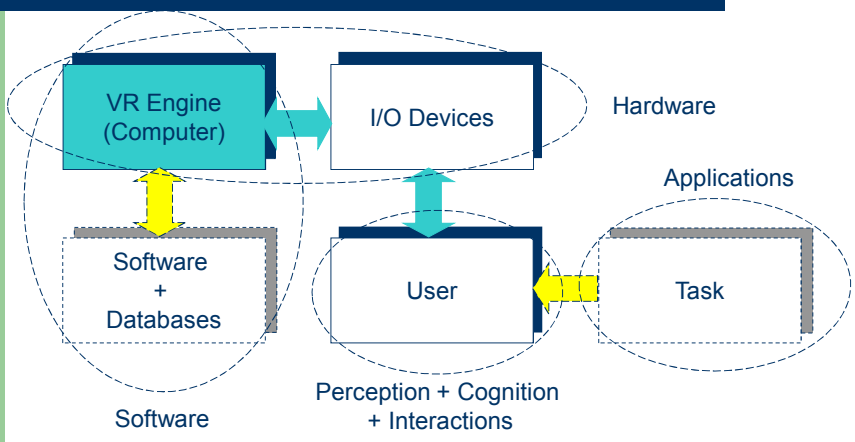
- Lights:




35

Dr. Y. Hu

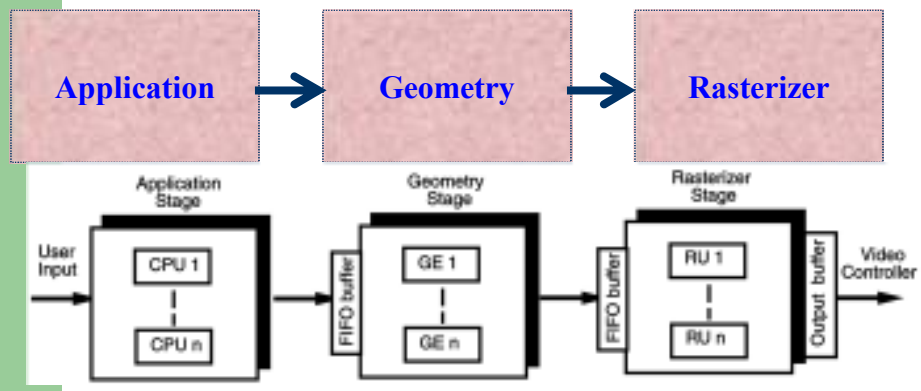
## Key Components of a VR System




36

Dr. Y. Hu

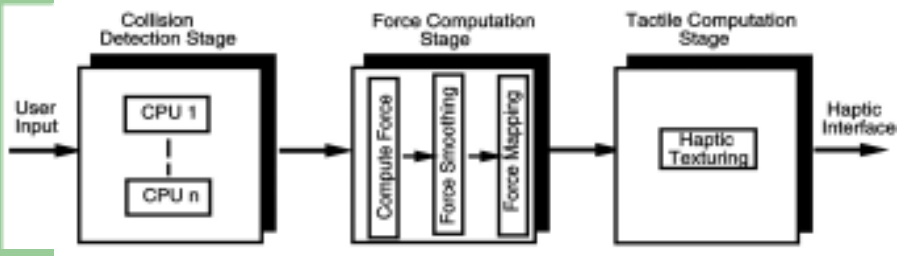
## The Graphics Rendering Pipeline




38

Dr. Y. Hu

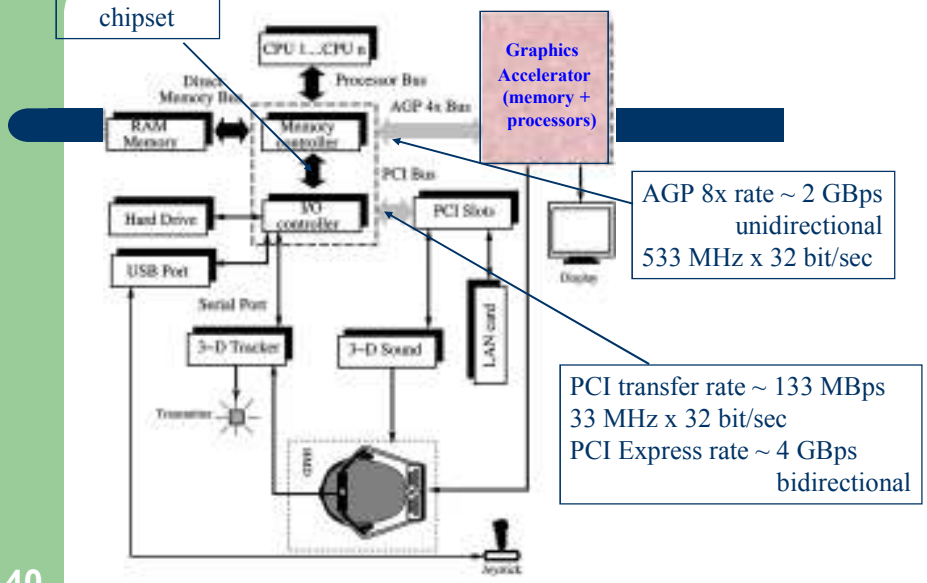
## The Haptic Rendering Pipeline




39

Dr. Y. Hu

# PC Architecture








# Workstation - SGI InfiniteReality

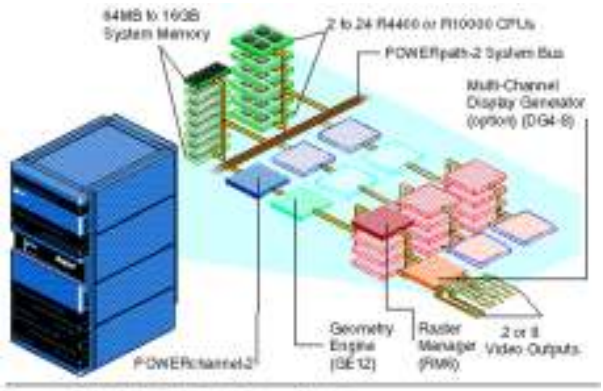
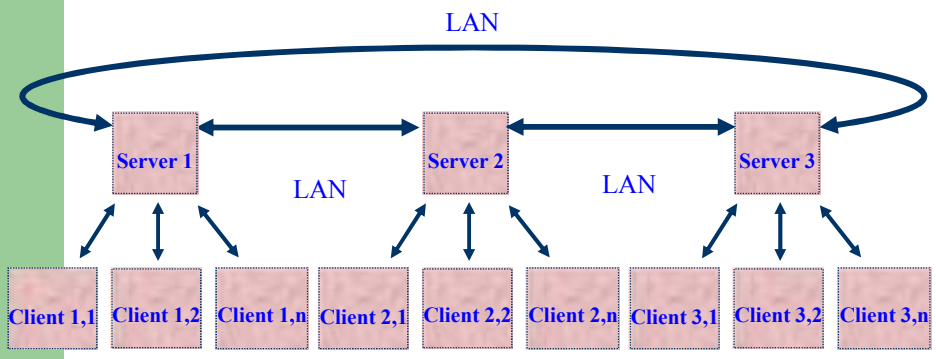



**Figure 2.** InfiniteReality Rack System Block Diagram

## Networked VR








## Recap

- Five key components of a VR system
  - VR Engine (computer)
  - Software and database
  - Input / output devices
  - User
  - Task (application)

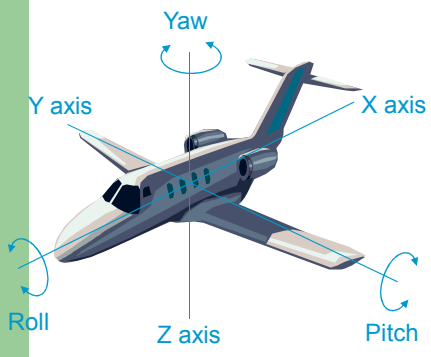
# ENSF 545
# Introduction to Virtual Reality

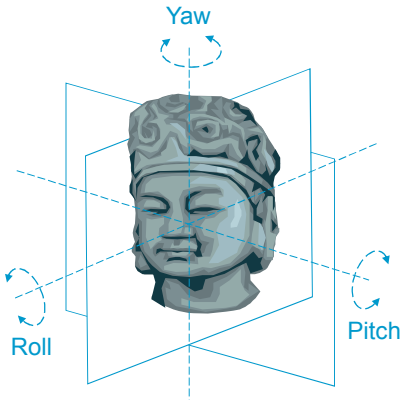
## Basic Mathematics

## Why Needs Mathematics in VR?

- Mathematics
  - Describing a scene
  - Performing operations on the scene
- Method
  - Coordinate systems (right- and left-handed) with 3 axes labelled x, y, z at right angles.

## Coordinate Frame

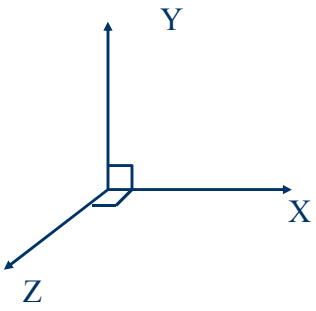


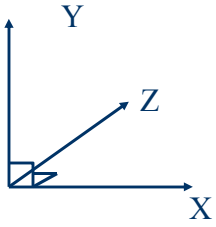




3

Dr. Y. Hu

## Co-ordinate Systems



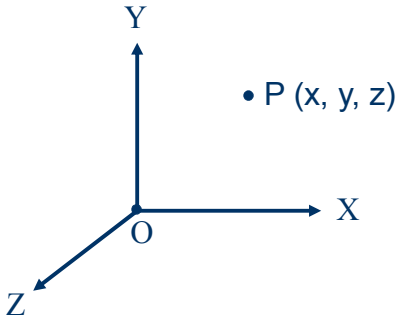




4

Dr. Y. Hu

## Points, P (x, y, z)

- Gives a position in relation to the origin (O) of a coordinate system

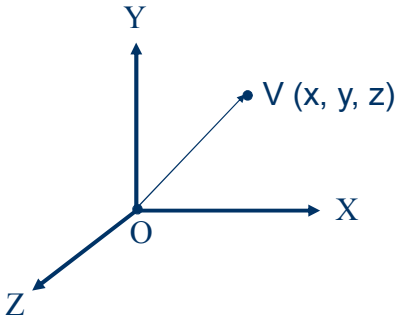



5

Dr. Y. Hu

## Vectors, V (x, y, z)

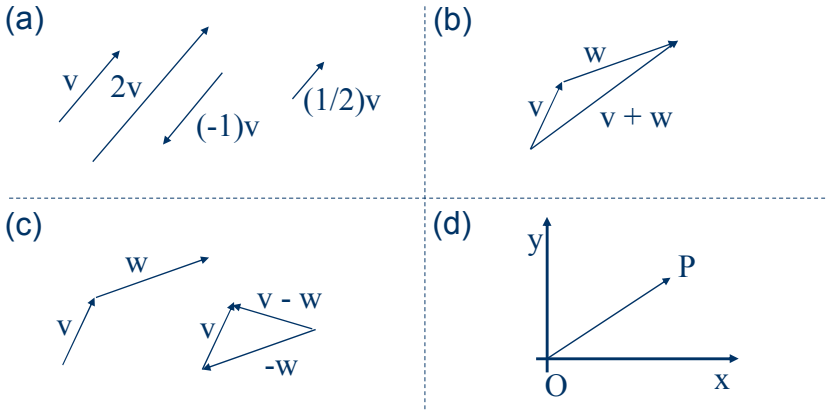
- Represent a *direction (*and magnitude) in 3D space
- Points != Vectors




6

Dr. Y. Hu

## Vectors, V (x, y, z)






Dr. Y. Hu

## Vectors V

- Length (modulus) of a vector **V** (x, y, z)
- A unit vector:

  a vector can be *normalised* such that it retains its direction, but is scaled to have unit length.



Dr. Y. Hu

## Dot Product

$\mathbf{u} \cdot \mathbf{v} = x_u \cdot x_v + y_u \cdot y_v + z_u \cdot z_v$

$\mathbf{u} \cdot \mathbf{v} = |u|\,|v| \cos\theta$

$\therefore \cos\theta = \mathbf{u} \cdot \mathbf{v} / |\mathbf{u}|\,|\mathbf{v}|$

Important: $(\mathbf{u} \cdot \mathbf{v})$ is a scalar number, not a vector





## Cross Product

- Important:
  $(\mathbf{u} \times \mathbf{v})$ is not a scalar but a vector, which is normal to the plane of the vectors **u** and **v**.
- Can be computed using the determinant of:

$$\begin{vmatrix} i & j & k \\ x_u & y_u & z_u \\ x_v & y_v & z_v \end{vmatrix} \quad \begin{vmatrix} i & j & k \\ x_u & y_u & z_u \\ x_v & y_v & z_v \end{vmatrix} \quad \begin{vmatrix} i & j & k \\ x_u & y_u & z_u \\ x_v & y_v & z_v \end{vmatrix}$$

$\mathbf{u} \times \mathbf{v} = \mathbf{i}(y_u z_v - z_u y_v), -\mathbf{j}(x_u z_v - z_u x_v), \mathbf{k}(x_u y_v - y_u x_v)$

v

- $|\mathbf{u} \times \mathbf{v}| = |\mathbf{u}||\mathbf{v}|\sin\theta$

## Parametric Equation of a Line (Ray)

Given two points $P_0 = (x_0, y_0, z_0)$ and $P_1 = (x_1, y_1, z_1)$, the line passing through them can be expressed as:

$$P(t) = P_0 + t(P_1 - P_0) = \begin{cases} x(t) = x_0 + t(x_1 - x_0) \\ y(t) = y_0 + t(y_1 - y_0) \\ z(t) = z_0 + t(z_1 - z_0) \end{cases}$$

With $-\infty < t < \infty$





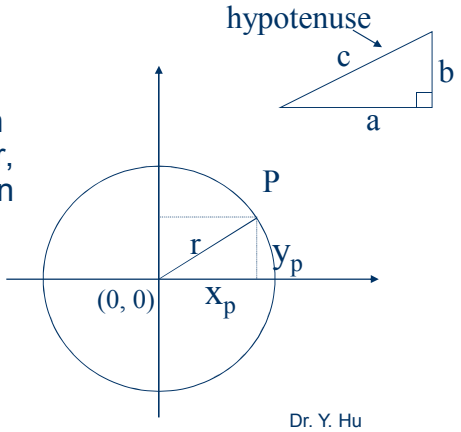
## Equation of a Sphere

- Pythagoras Theorem:

$$a^2 + b^2 = c^2$$

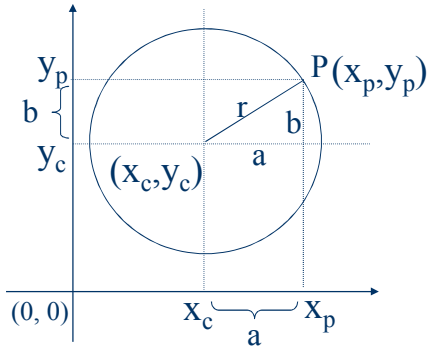
- Given a circle through the origin with radius r, then for any point P on it we have:

$$x^2 + y^2 = r^2$$



## Equation of a Sphere (cont'd)

* If the circle is not centred on the origin:




We still have

$a^2 + b^2 = r^2$

but

$a = x_p - x_c$

$b = y_p - y_c$

So for the general case ????





## Equation of a Sphere (cont'd)

* Pythagoras theorem generalises to 3D giving

$a^2 + b^2 + c^2 = d^2$

* The general equation of a sphere is:

???

## Vectors and Matrices

- Matrix is an array of numbers with dimensions M (rows) by N (columns)

$$\begin{pmatrix} 3 & 0 & 0 & -2 & 1 & -2 \\ 1 & 1 & 3 & 4 & 1 & -1 \\ -5 & 2 & 0 & 0 & 0 & 1 \end{pmatrix}$$

- Vector can be considered a 1 x M matrix

$$v = \begin{pmatrix} x & y & z \end{pmatrix}$$

15

Dr. Y. Hu

## Types of Matrix

- Identity - I

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Symmetric

$$\begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix}$$

- Diagonal

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -4 \end{pmatrix}$$

16

Dr. Y. Hu

## Operation on Matrices

- Addition

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} + \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} a+e & b+f \\ c+g & d+h \end{pmatrix}$$

- Transpose

$$\begin{pmatrix} 1 & 4 & 9 \\ 5 & 2 & 8 \\ 6 & 7 & 3 \end{pmatrix}^T = \begin{pmatrix} 1 & 5 & 6 \\ 4 & 2 & 7 \\ 9 & 8 & 3 \end{pmatrix}$$



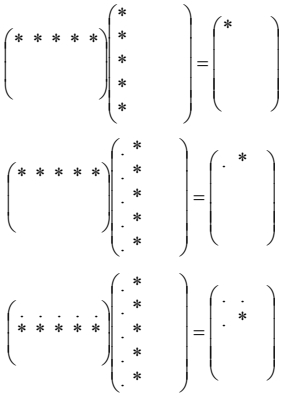


## Operations on Matrices

- Multiplication
  - A is n by k , B is k by m; A x B is n by m
  - C = A x B defined by

$$c_{ij} = \sum_{l=1}^{k} a_{il} b_{lj}$$

  - B x A not necessarily equal to A x B



## Example Multiplications

$$\begin{pmatrix} 2 & 3 & 1 \\ -1 & 0 & 1 \end{pmatrix}\begin{pmatrix} 2 & 1 \\ 1 & -1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \_\_ & \_\_ \\ \_\_ & \_\_ \end{pmatrix}$$

$$\begin{pmatrix} 2 & -2 & 3 \\ -3 & 1 & 0 \\ 1 & -1 & -1 \end{pmatrix}\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} \_\_ & \_\_ & \_\_ \\ \_\_ & \_\_ & \_\_ \\ \_\_ & \_\_ & \_\_ \end{pmatrix}$$





## Inverse

- If A x B = I and B x A = I then
  $A = B^{-1}$ and $B = A^{-1}$

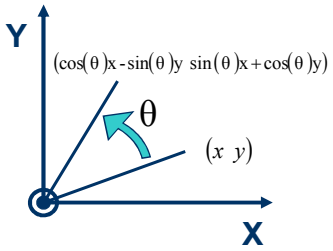
## 3D Transforms – Using 3 by 3 Matrices

- Scale
  - Use a diagonal matrix
  - Example:

$$\begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{pmatrix} \begin{pmatrix} 3 \\ 4 \\ 5 \end{pmatrix} = \begin{pmatrix} 6 \\ 4 \\ -10 \end{pmatrix}$$

- Rotation
  - Example: rotation about z axis

$$\begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$








## Rotation X, Y; Scale; Translation

- About X

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix}$$

- About Y
- Scale (should look familiar)
- Translation

## Homogenous Points

- Add 1D, but constrain that to be equal to 1
- Homogeneity means that any point in 3D space can be represented by an infinite variety of homogenous 4D points
- Why?
  - 4D allows as to include 3D translation in matrix form

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 2 \\ 3 \\ 4 \\ 1 \end{pmatrix} = \begin{pmatrix} 4 \\ 6 \\ 8 \\ 2 \end{pmatrix} = \begin{pmatrix} 3 \\ 4.5 \\ 6 \\ 1.5 \end{pmatrix}$$





## Homogenous Vectors

- Vectors != Points
- Remember points can not be added
- If A and B are points A-B is a vector

- Vectors have form $(x\ y\ z\ 1)^T$
- Addition makes sense

## Translation in Homogenous Form

$$\begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x+a \\ y+b \\ z+c \\ 1 \end{pmatrix}$$





## Putting it Together

$$\begin{pmatrix} R_1 & R_2 & R_3 & T_1 \\ R_4 & R_5 & R_6 & T_2 \\ R_7 & R_8 & R_9 & T_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} = R.T$$

- R is rotation and scale components
- T is translation component

## Order Matters

- Composition order of transforms matters

$$\begin{pmatrix}1&0&0&0\\0&0&-1&0\\0&1&0&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}1&0&0&2\\0&1&0&3\\0&0&1&4\\0&0&0&1\end{pmatrix}\begin{pmatrix}X\\Y\\Z\\1\end{pmatrix}=\begin{pmatrix}1&0&0&2\\0&0&-1&-4\\0&1&0&3\\0&0&0&1\end{pmatrix}\begin{pmatrix}X\\Y\\Z\\1\end{pmatrix}=\begin{pmatrix}X+2\\-Z-4\\Y+3\\1\end{pmatrix}$$

$$\begin{pmatrix}1&0&0&2\\0&1&0&3\\0&0&1&4\\0&0&0&1\end{pmatrix}\begin{pmatrix}1&0&0&0\\0&0&-1&0\\0&1&0&0\\0&0&0&1\end{pmatrix}\begin{pmatrix}X\\Y\\Z\\1\end{pmatrix}=\begin{pmatrix}1&0&0&2\\0&0&-1&3\\0&1&0&4\\0&0&0&1\end{pmatrix}\begin{pmatrix}X\\Y\\Z\\1\end{pmatrix}=\begin{pmatrix}X+2\\-Z+3\\Y+4\\1\end{pmatrix}$$





## Exercises

1. Calculate the following matrix: $\pi/2$ about X, then $\pi/2$ about Y, then $\pi/2$ about Z (remember "then" means multiply on the left). What is a simpler form of this matrix?
2. Compose the following matrix: translate 2 along X, then rotate $\pi/2$ about Y, then translate -2 along X. Draw a figure with a few points (you will only need 2D) and then its translation under this transformation.

## Summary

- Rotation, Scale, Translation
- Composition of transforms
- The homogenous form

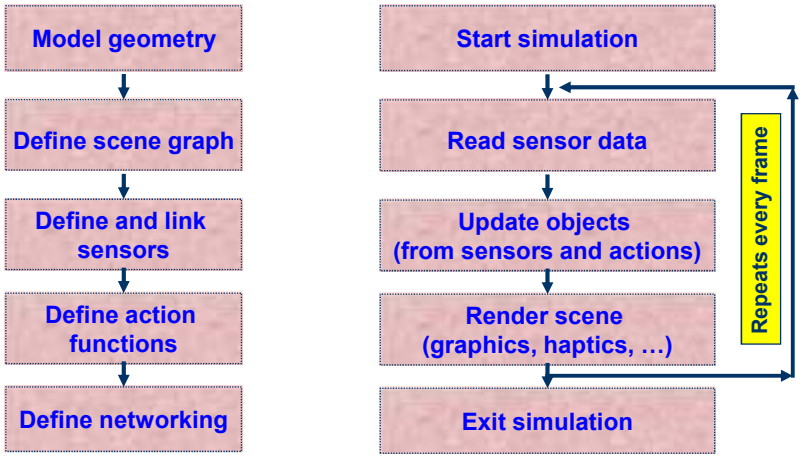
# ENSF 545
# Introduction to Virtual Reality

## VR Programming

## How to Build a VR Simulation?


2

## Authoring Stage + Run-time Loop




3

## Major Concepts of Graphics

- Separation of
  - Scene → Scene is independent of any view
  - Viewing → Views are unconstrained
  - Rendering → There are many possible rendering methods given a scene and a view

4

Dr. Y. Hu

## (Partial) Scene

## What Does Graphics API's Do?

- Objects ← **primitives**
- Primitives ← **attributes** (color, materials etc.)
- **Transformations** ← translations, rotations
- **Viewing** ← camera position, orientation, etc.
- **Input** mechanisms ← user interface
- **Control** ← communicating with operating systems, initializing programs, etc.



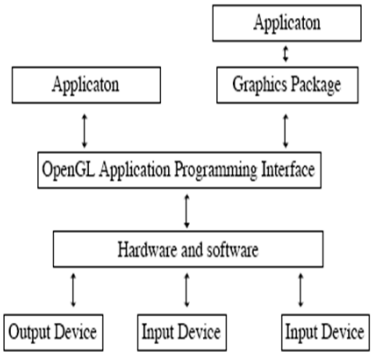


## VR Graphics and Utility Libraries

- Low level
  - **OpenGL**, Direct 3D
- User interface libraries
  - **Glut**, Tweek
- Specific purposes
  - 3DGame Studio
  - PeopleShop
  - **OpenHaptics**
  - DirectSound 3D
- High level
  - Open Inventor
  - Coin 3D
  - Open Performer
  - Open SG
  - VR Juggler
  - Java 3D
  - VRML
  - WorldToolKit ( WTK)
  - VTK

## OpenGL – www.opengl.org

- Developed in early 90s
- Standardized
  - Managed by Architectural Review Board (ARB)
- Procedural model
- Independent of operating systems








## OpenGL Libraries

- **Gl & Glu:** The OpenGL utilities.
  - Containing code for OpenGL functions, definitions.
- **Glut:** The OpenGL utility toolkit.
  - For using OpenGL within a windowing environment (e.g. windows or Mac OS).
  - Operating system dependent.

## OpenGL Model and Process

- Model:
  - Define 3D objects in space.
  - Specify camera properties (position, orientation, projection system, etc).
- Process:
  - Transformation
  - Clipping
  - Projection
  - Rasterization

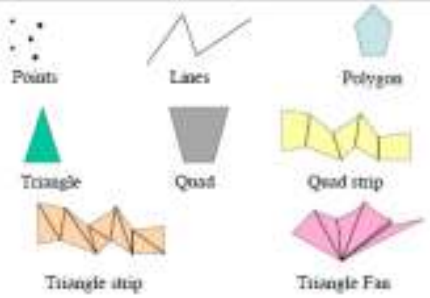


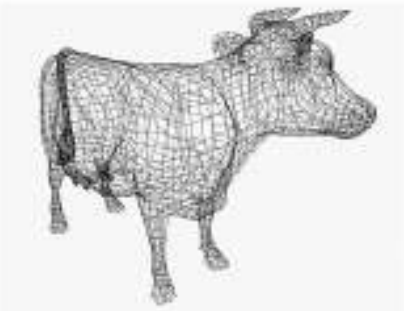

## OpenGL Basics

- Draw primitives (lines, polygons)
- Draw 3D objects
- Use a synthetic camera to form images
- Convention:

```
glFunction(para…);
GLtype
GL_TRIANGLES
```

## Primitives, Polygonal Meshes



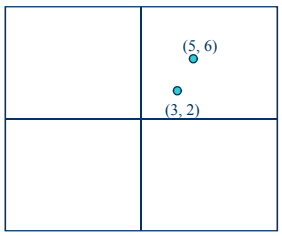



13

## OpenGL primitives: Vertices

- OpenGL defines objects in terms of vertices (points).
  - 2D shapes: polygons defined by vertices.
  - 3D shapes: groups of polygons.
- Example:

```
glBegin(GL_POINTS);
   glVertex2f(3.0, 2.0);
   glVertex2f(5.0, 6.0);
 glEnd();
```




2D plane

Dr. Y. Hu

14

## Defining a Vertex

- A 2D vertex:

```
glVertex2f(GLfloat x, GLfloat y);
```

2D vertex  floating point  openGL parameter type

- A 3D vertex:

```
glVertex3f(GLfloat x, GLfloat y, GLfloat z);
```

- **Example**:
  - A 3D vertex using integer values ??





## Arrays → Define Points

```
GLfloat myVertex[3];
myVertex[0] = 4.0;            //x value
myVertex[1] = 2.0;            //y value
myVertex[2] = 1.0;            //z value

  . . .
glVertex3fv(myVertex);
```

using array

pointer to array
containing point values

## typedef → Define Points

Array of points as 2D array:

```
GLfloat myPoints[5][2]; //array of 5 2D points
```

Using typedef:

```
//a new type, point2, as an array of 2 GLfloats
typedef GLfloat point2[2];

point2 myPoint;        //a 2 element array
myPoint[0] = 1.0;
myPoint[1] = 2.7;
```





## An Array of Points

```
//a new type, point2, as an array of 2 GLfloats
typedef GLfloat point2[2];
point2 myPoints[4] =
  {{1.0, 3.5}, {4.2, 6.7}, {3.1, 2.2}, {7.2, 1.1}};
```

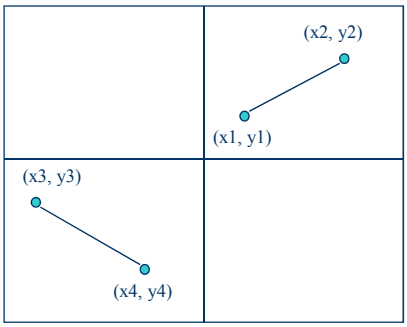
What is the value of:

- `myPoints[0]`
- `myPoints[2][1]`

## Connect Points

GL_LINES connects **pairs of points**.

```
glBegin(GL_LINES);
  glVertex2f(x1, y1);
  glVertex2f(x2, y2);
  glVertex2f(x3, y3);
  glVertex2f(x4, y4);
glEnd();
```

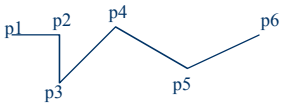




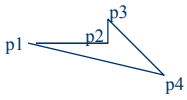



## Other Drawing Functions

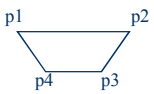
GL_LINE_STRIP:



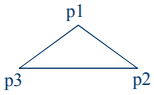

GL_LINE_LOOP:




GL_POLYGON:

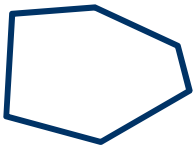



GL_TRIANGLES:

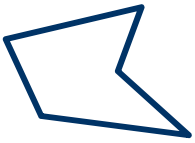


## Convex, Concave

- Convex



- Concave





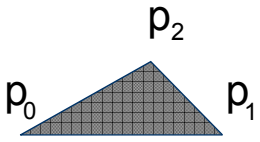


## Normal: Order of Vertices

- The cross product

$$n = (p_1 - p_0) \times (p_2 - p_0)$$

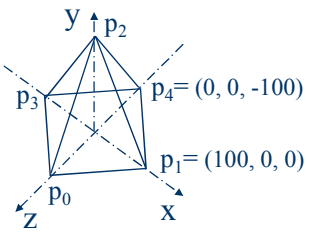
defines a **normal** to the plane




- **Normal**: A vector perpendicular to a plane
  - Pointing outward of the positive face of the plane
  - Important for lighting and shading of objects

## Object in 3D

- Use 3D points instead of 2D points.
- Example: 3D pyramid.

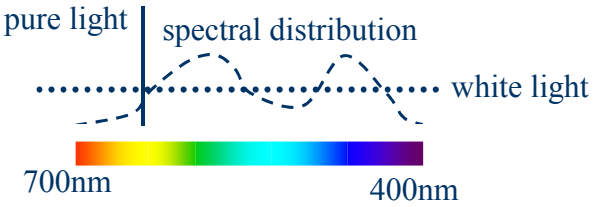





Dr. Y. Hu

## Light in an Environment

- Light:




- Reflection:




Dr. Y. Hu

## The Frame Buffer

- The **frame buffer** stores the value of each pixel in the viewing window.
- Each pixel has a given number of bits to encode the color. The number of bits is the **bit depth**.
  - 8-Bit depth → 256 possible colors ($2^8$)
  - 32-Bit depth →millions of possible colors ($2^{32}$)
  - 64-Bit depth → ???

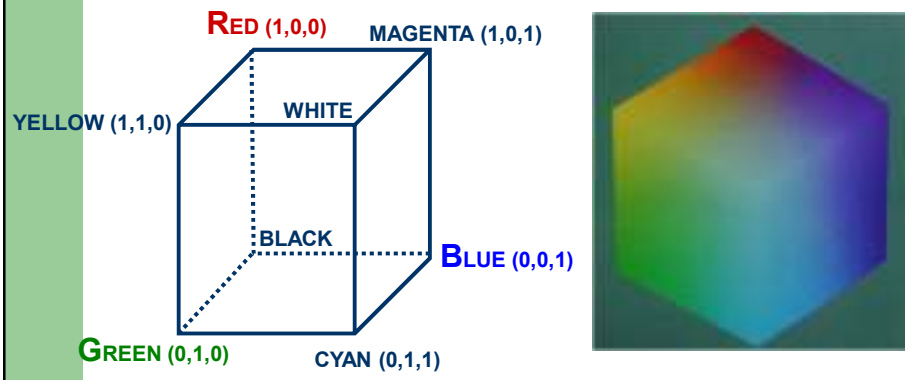




## Indexed Color

OpenGL:
Don't depend on particular hardware or number of bits per pixel.
Use generic color scale between 0 and 1.0 for each R, G, B value.

# RGB Color Model








# OpenGL Shading and Color

- Shading properties

```
glShadeModel(GL_SMOOTH | GL_FLAT)
```

- Color

```
glColorNT{V}(r,g,b,{α})
```

  - N = 3, 4
  - T = f, b, s, i, ub, ui, us
  - v implies passing a pointer to array of color

## OpenGL Color Specification

```
glDisable (GL_LIGHTING);
glColor3f(r, g, b); //r, g, b value btw.0.0 and 1.0
   … …
glEnable (GL_LIGHTING);

glColor3f(1.0, 0.0, 0.0);      //What color is this?
glColor3f(1.0, 0.0, 1.0);
glColor3f(0.0, 1.0, 0.0);
```

**α** Channel: - a 4th color parameter
- opacity (1.0), transparency (0.0)

```
glClearColor(1.0, 1.0, 1.0, 1.0);
```

RGB white

α opaque





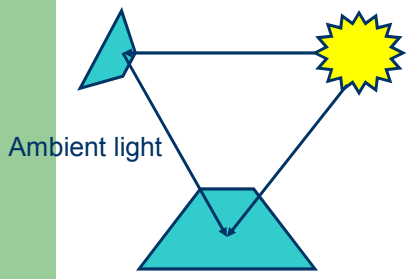
## OpenGL Materials

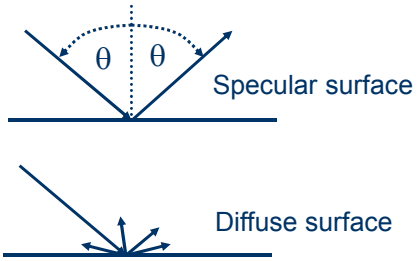
- Many lighting parameters
- Specify a material as
  - ambient, shininess, diffuse, specular

```
GLfloat mat_spec = { 0.5, 0.5, 1.0, 1.0};
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_spec)
glColorMaterial(GL_FRONT, GL_DIFFUSE)
```

## Lighting Model

- Any point in the environment receives light from all around

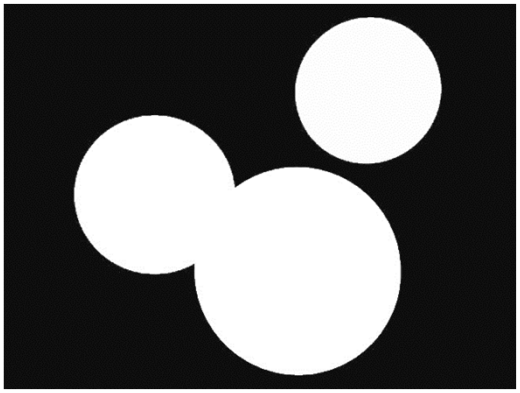



32

Dr. Y. Hu

## The Image - Detection



33

Dr. Y. Hu

## Ambient Light

- Approximation to global illumination
  - Illuminates each object to a certain extent
  - Be constant across a whole object
- Usually set for whole scene ($I_a$)
- Each object reflects only a proportion ($k_a$)
- So far then

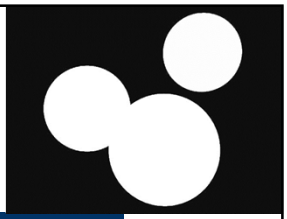
$$I_r = k_a I_a$$

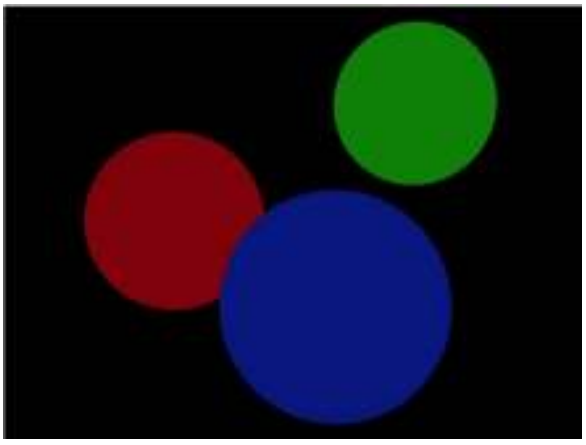
- But using RGB, so .... ?





## The Image - Ambient

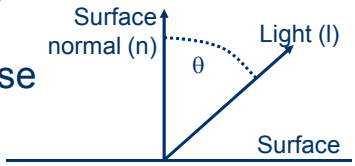




## Diffuse Light - Lambert's Law

- Lambert's law: Reflected intensity is proportional to *cosθ*
- The proportion of light reflected due to Lambert's law rather than absorbed ($k_d \rightarrow$ as $k_{d,red}$ $k_{d,green}$ and $k_{d,\ blue}$)
- Light with ambient and diffuse components

$$I_r = k_a I_a + k_d I_i (n.l)$$








## Multiple Lights?

- Add the diffuse terms

$$I_r = k_a I_a + \sum_{j=1}^{m} k_d I_{i,j} (n.l_j)$$

- $I_{i,j}$ is the incoming intensity of light j
- $l_j$ is the vector to light j

## The Image - Diffuse



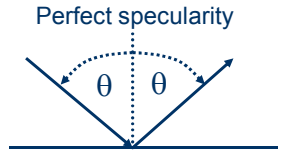


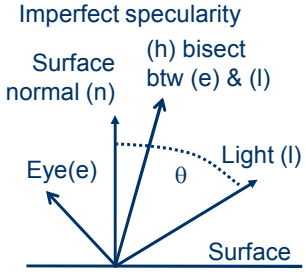
## Imperfect Specularity (Phong)

- Specular component:

$$k_s I_i (h.n)^m$$

- m is the power of the light
  - High m implies smaller specular highlight
  - Low m makes the highlight more blurred



## Specular Light

- Ambient, diffuse and specular components

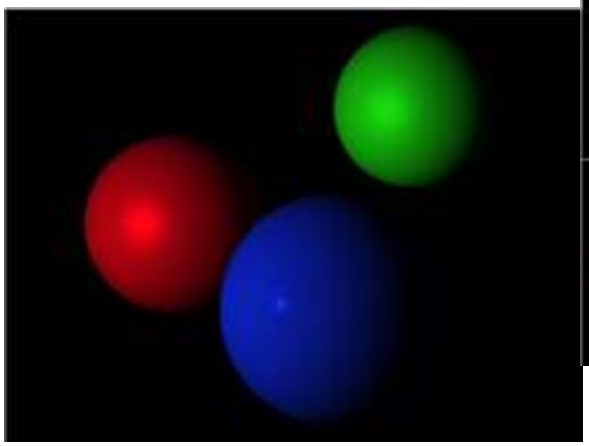
$$I_r = k_a I_a + I_i (k_d (n.l) + k_s (h.n)^m)$$

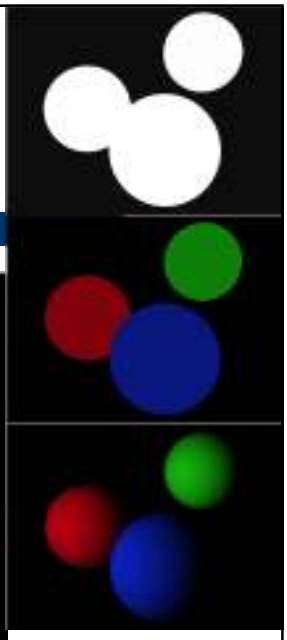
- If multiple lights ➔ ????





## The Image - Specular





## OpenGL Light Models

- Light Models:

```
GL_LIGHT_MODEL_AMBIENT          (Ambient RGBα intensity)
GL_LIGHT_MODEL_Local_VIEWER     (Specular reflection)
GL_LIGHT_MODEL_TWO_SIDE         (Lighting 1-sided or 2-sided)
```

- Example:

```
GLfloat lmodel_ambient [] = {0.2, 0.2, 0.2, 1.0}
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,lmodel_ambient)
GLfloat light_pos ={ 1.0, 2.0, 1.0, 0.0}
glLightfv(GL_LIGHT0, GL_POSITION, light_pos)
glEnable(GL_LIGHTING)
glEnable(GL_LIGHT0)
```




## Concatenation of Transformations

- Suppose you want to
  - Scale an object: P1 = SP
  - Rotate the object: P2 = RP1 = RSP
  - Translate the object: P3 = TP2 = TRSP
- With a new matrix M = TRS, P3 = MP

## Transformations in OpenGL

- Creating your own transformation:
  - Computing transformation in advance
  - Entering matrix into array
  - Loading the matrix
  - Drawing object
- Use the OpenGL transformation functions

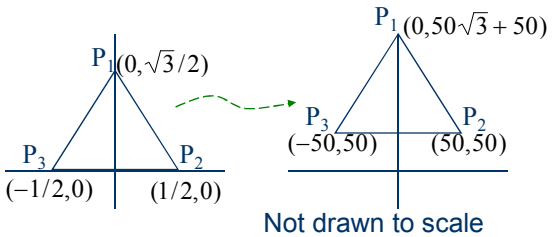




## Computing Transformation

**Example**:

Scale by 100 in both the x and y directions, then translate vertically by 50.




Not drawn to scale

## Entering Matrix into Array

In OpenGL, a transforming matrix is stored as a 16 element array in **column major order.**

$$\begin{pmatrix} 0 & 4 & 8 & 12 \\ 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \end{pmatrix}$$

```
scaleMatrix[0] = 100.0;
scaleMatrix[5] = 100.0;
scaleMatrix[10] = 1.0;
scaleMatrix[13] = 50.0;
scaleMatrix[15] = 1.0;
```

$$M = \begin{pmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 50 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

47

Dr. Y. Hu

## Loading Matrix + Drawing Object

1. Make sure the correct matrix mode
   ```
   glMatrixMode(GL_MODELVIEW);
   ```
2. Push the current model matrix onto the matrix stack
   ```
   glPushMatrix();
   ```
3. Load in the transformation matrix
   ```
   glLoadMatrixf(scaleMatrix);
   ```
4. Draw the object for transform
   ```
   glBegin(GL_LINE_LOOP);
         glVertex2f(x, y);
         ...
   glEnd();
   ```
5. Pop the old model matrix off the stack
   ```
   glPopMatrix();
   ```

48

Dr. Y. Hu

## Example Code

```
for(i = 0; i<16; i++){      //Set matrix to zero
    scaleMatrix[i] = 0;
}
scaleMatrix[0] = 100.0;     //Scale x by 100 fold
scaleMatrix[5] = 100.0;     //Scale y by 100 fold
scaleMatrix[10] = 1.0;      //Keep Z constant
scaleMatrix[13] = 50.0;     //Translate in y by 50.0
scaleMatrix[15] = 1.0;      //This element is always 1

glPushMatrix();             //Store the current matrix
glLoadMatrixf(scaleMatrix); //Load the scale matrix
glBegin(GL_LINE_LOOP);         //Draw the triangle
    for (i=0; i<3; i++){
       glVertex2fv(triVerts[i]);
}
glEnd();
glPopMatrix();      //Get original matrix back
```





## OpenGL Transform Functions

- Translate by vector (dx, dy, dz)

```
glTranslatef(dx, dy, dz);
```

- Rotate by angle about axis by (vx, vy, vz)

```
glRotatef(angle, vx, vy, vz);
```

- Scale by factor given by (sx, sy, sz)

```
glScalef(factor, sx, sy, sz);
```

## Order of Transformations

- Suppose: transformations in the order of:
  1. Scale  2. Rotate  3. Translate
- Recall concatenation of matrix multiplications:
  P' = M*P;  M = ??
- Start with the identity matrix (I):
  1. Call translatef()  M = I*T
  2. Call rotatef()  M = I*T*R
  3. Call scalef()  M = I*T*R*S
  4. Draw P  P' = MP = T*R*S*P





## Example Code

```
   glPushMatrix();            //Save model matrix
//Scale by 100 in x and y, done last
   glScalef(100.0, 100.0, 1.0);
//Translate in x direction, done 3rd
   glTranslatef(-sqrt(3.0), 0.0, 0.0);
//Rotate by 90 deg around z axis, done 2nd
   glRotatef(-90.0, 0.0, 0.0, 1.0);
//Scale along y axis only, done 1st
   glScalef(1.0, 2.0, 1.0);
   glBegin(GL_LINE_LOOP); //Draw a triangle
      for (i=0; i<3; i++){
         glVertex2fv(triVerts[i]);
      }
   glEnd();
   glPopMatrix(); //get original model matrix back
```

## Concatenating Transformation Matrix

- Concatenate a transformation matrix with other transformations, use the openGL function:

```
glMultMatrixf(const GLfloat *m);
```

- Example: Concatenating the `scaleMatrix` with a rotation of 90 degrees (assume `scaleMatrix` has been initialized to desired values):

```
glRotatef(-90.0, 0.0, 0.0, 1.0);
glMultMatrixf(scaleMatrix);
glBegin(GL_LINE_LOOP);
       ...
```

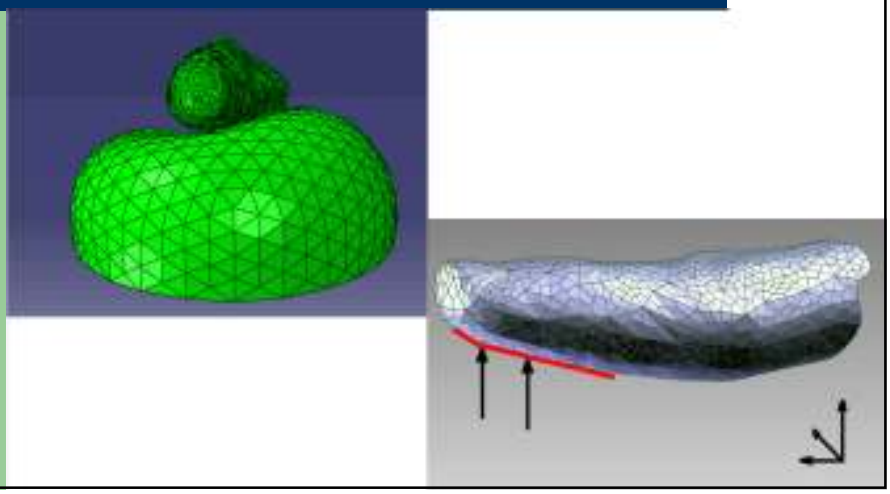




## Recap

- Using OpenGL library
  - Geometric primitives + 3D objects
  - Color + Lighting
  - Transformation

## Polygon Partitioning Object



55

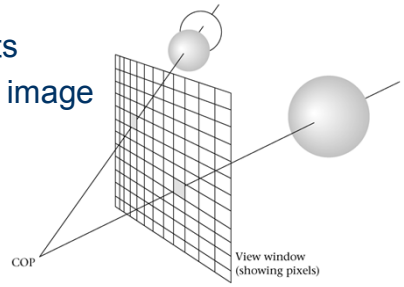
# ENSF 545
# Introduction to Virtual Reality

## Advanced VR Programming

## Camera, Projection, and View

## Camera (Eye)

- Simple camera is limiting and it is necessary to model a camera that can be moved.
- Parameters of a camera
  - Location (x, y, z)
  - Direction the camera points
  - Direction to be “up” on the image

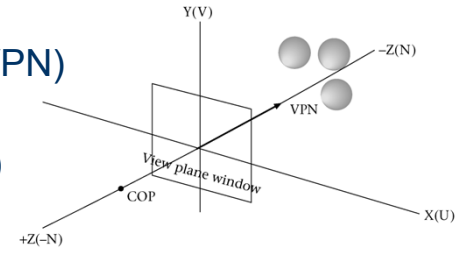



COP = centre of projection = view point

3

## Camera Coordinate System

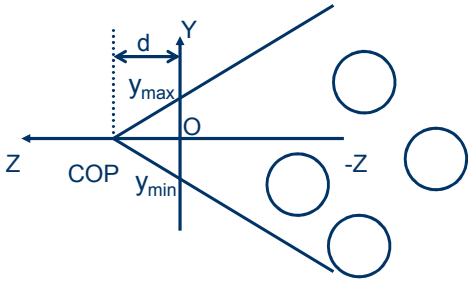
- Coordinate systems
  - Word (**XYZ**) – Right-handed
  - Camera (**UVN**) – Left-handed
- View reference point (VRP)
  - camera location
- View plane normal (VPN)
  - camera points to
- View up vector (VUV)
  - Camera up direction

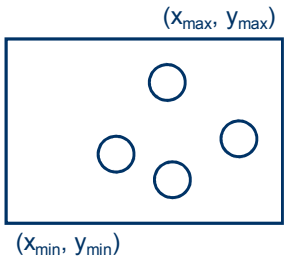



4

## Simple Camera (cross section)

- The objects must be in front of a camera to be seen.




X axis points out the slide.

COP = A simple camera

Camera view plane

5

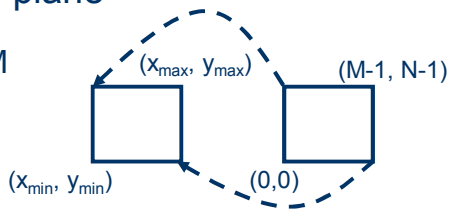


## Mapping btw. View and Window

- Mapping screen pixels (M by N window) to points in camera view plane

width = (xmax-xmin)/M

height = (ymax-ymin)/N



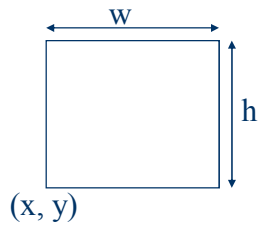

- Consider pixel i,j as a point

(xmin + width*(i+0.5), ymin + height*(j+0.5), 0.0)

6

## Defining a Viewport

- Clipping window ← **world coordinates**.
- OpenGL rendering ← **screen coordinates**.
- The drawing region on a screen → **`ViewPort`.**



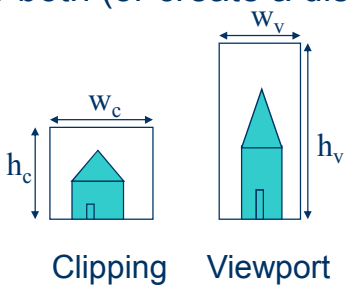

```
void glViewPort(GLint x, GLint y,
                GLsizei w, GLsizei h);
```



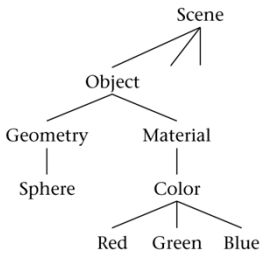


## Mapping from World to Screen

- The clipping region ←→ The entire viewport.
- Make the height/width (aspect ratio) the same for both (or create a distorted image).

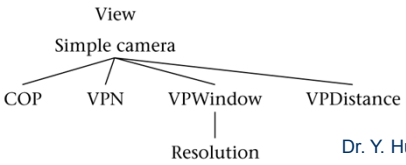


## Structure of a Scene and a View




A scene consists of multiple objects

A view consists of a camera with settings controlled by parameters

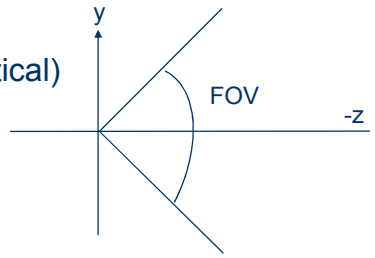







## Alternative Forms of the Camera

- Simple "Look At"
  - Give a VRP and a target point (TP)
  - VPN = TP-VRP
  - VUV = (0, 1, 0) (i.e. "up" in World Coordinates)
- Field of View (FOV)
  - Give FOV (horizontal or vertical)
  - An aspect ratio
  - Calculate viewport

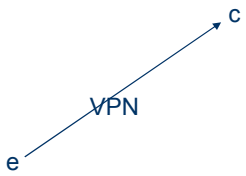


# OpenGL - glu Viewing

- Constructing an 'M' matrix

```
gluLookAt(ex,ey,ez,    //eye = COP (world coord.)
          cx,cy,cz,    //point of interest
          upx,upy,upz //up vector
          )
```

- Matrix that maps
  - `(cx,cy,cz)` to –Z axis
  - `(ex,ey,ez)` becomes the origin
  - `(upx,upy,upz)` becomes the y-axis
- Pre-multiplies current matrix

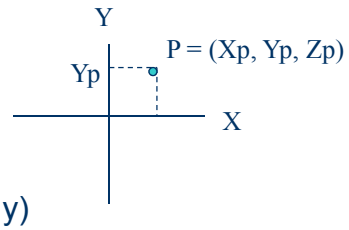


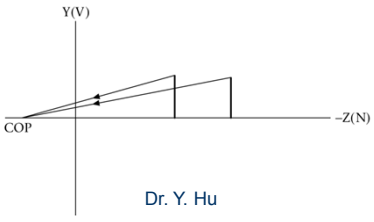





# Projections

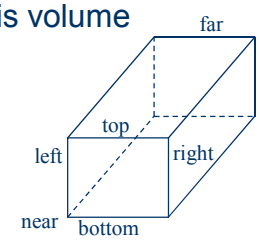
- Orthographic projection
  - COP at infinite far from the view plane
  - Z = 0 and point P = (X, Y, Z) projects to image point p = (x, y) where x = X and y = Y
- Perspective projection

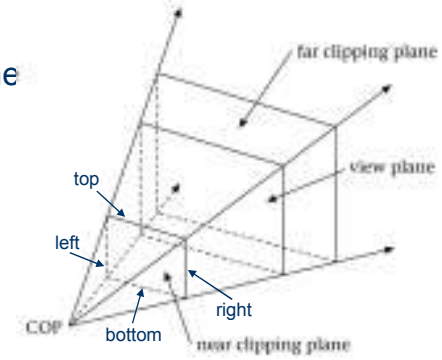


## View Volume

- View volume = Clipping volume
- **Clipping volume**
  - The region of the scene
  - Not rendering outside this volume



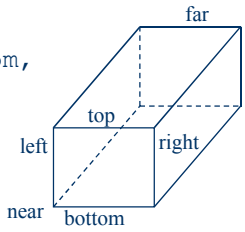



Dr. Y. Hu

## The Clipping Volume - Ortho

- In 3D

```
void glOrtho(left, right, bottom,
             top, near, far);
```




- In 2D

```
void glOrtho2D(left, right, bottom, top);
```

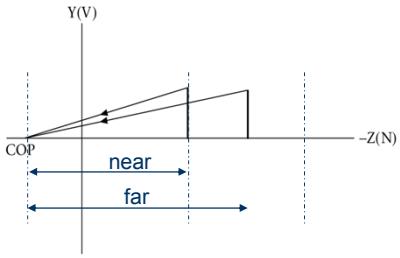
(glOrtho2D sets near and far to -1.0 and 1.0 respectively)

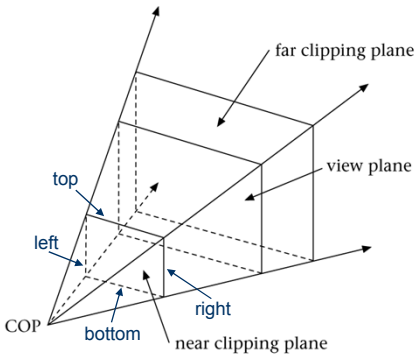


Dr. Y. Hu

# The Clipping Volume - Perspective

- **Frustrum**




```
glFrustrum(GLdouble left, GLdouble right, GLdouble
bottom, GLdouble top, GLdouble near, GLdouble far)
```

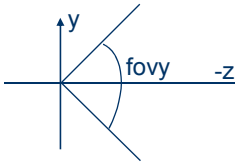


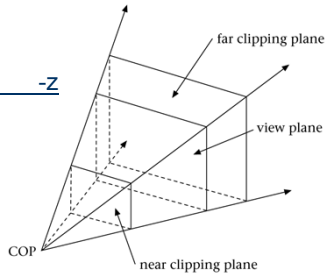


# OpenGL - glu Perspective

- glu projection matrix:

```
gluPerspective(fovy,    //field of view degrees
               aspect,  //xwidth/yheight
               zNear,   //near clipping plane
               zFar     //far clipping plane
              )
```




- Same viewing volume as `glFrustrum()`

## Revisit: Viewport

- Clipping volume ← **world coordinates**.
- OpenGL rendering ← **screen coordinates**.

- The drawing region within the clipping volume on a screen → `ViewPort`.



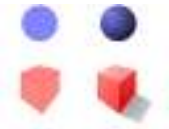


## Stereo Viewing

## Human Depth Cues

Relative size

Lighting, shadows



Linear perspective

Camera focus, depth of field

Overlaying

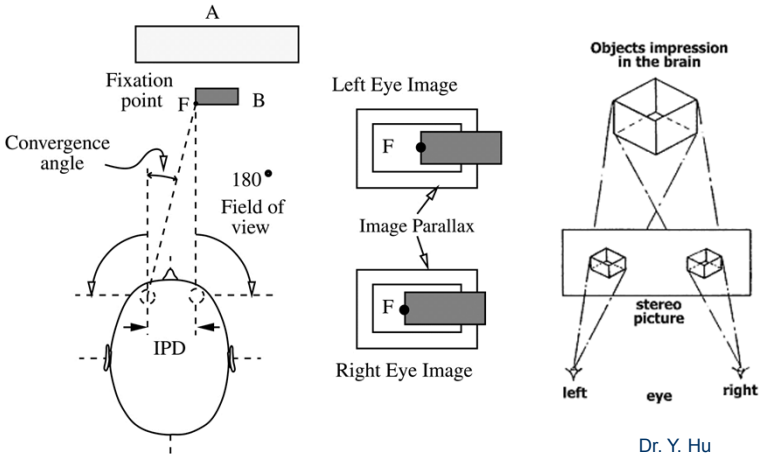
Speed





IPD = inter-pupillary distance

## Human Visual System



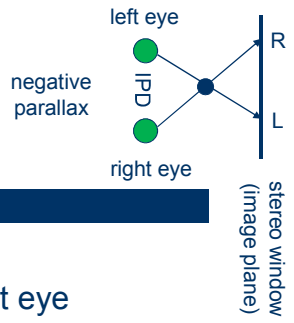
## Terminology

- Accommodation
  - → Adjustment of the focal length of the eyes
- Convergence
  - → Eye rotation inwards and parallel
- Binocular disparity
  - → Image differences produced by left and right eyes
- Motion parallax
  - → Relative movement of points with respect to head

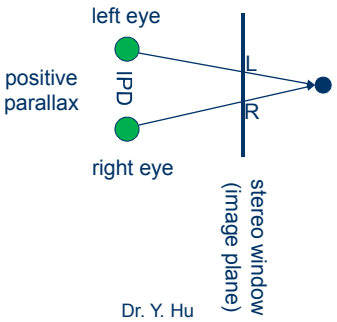




## Stereoscopy

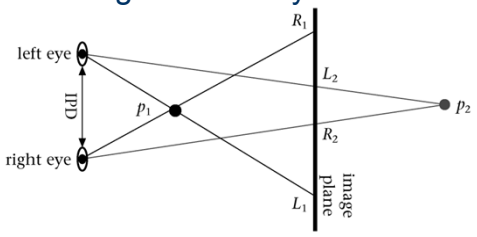



- Stereo pairs
  - → Two projections, left and right eye on flat display
- Horizontal parallax (R – L)
  - – R-L < 0; negative (pop-up)
  - – R-L > 0; positive
- Similar term for vertical parallax



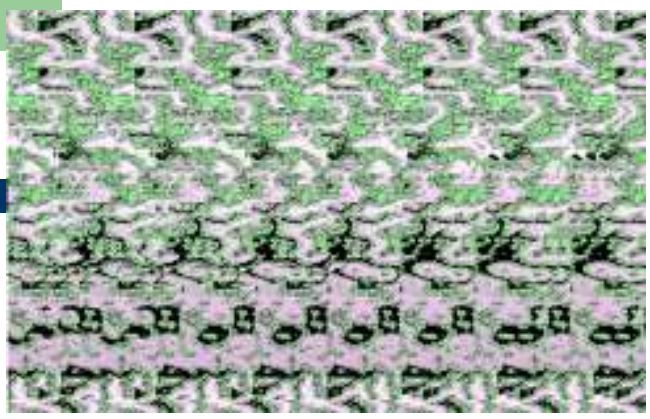
## Viewing Stereo Pairs

- Uncrossed/parallel setup
  → when right eye sees right image and left eye the left image
- Crossed setup
  → when right eye sees left image and left eye sees right image
- How to reverse the sense of depth?




25

## Try!!

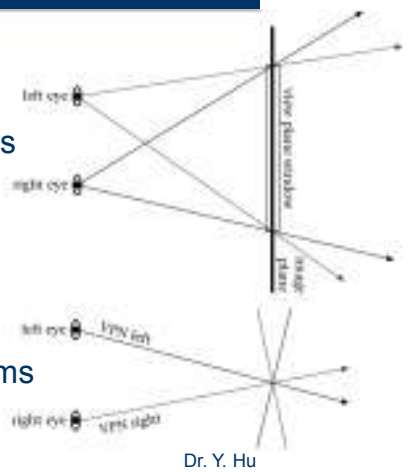


http://www.3dartist.com/3dao/stereo.htm

26

## Stereo Viewing - Programming

- Two cameras
  - → Each eye = one camera
  - → Set left and right cameras
- One camera
  - → Each eye = one frustum
  - → Draw left and right frustums






Dr. Y. Hu

## Ideals - I

- Congruence for left and right images
  - → colour, geometry, brightness
- Avoidance of vertical parallax (should be zero)
- Wide parallax (large separation of the eyes)
  - → good depth, but discomfort
- Maximum depth, but lowest parallax
- Further distance of viewer from display, the greater the parallax that can be tolerated



Dr. Y. Hu

## Ideals - II

- Cross-talk: left images reach right eye, and right images reach left eye
- Impacts of accommodation and convergence breakdown
  - Use lowest possible parallax for required depth
  - The closer homologous points, the less the disparity btw. accommodation and convergence
- The parallax less than or equal to IPD
- Other cues!!

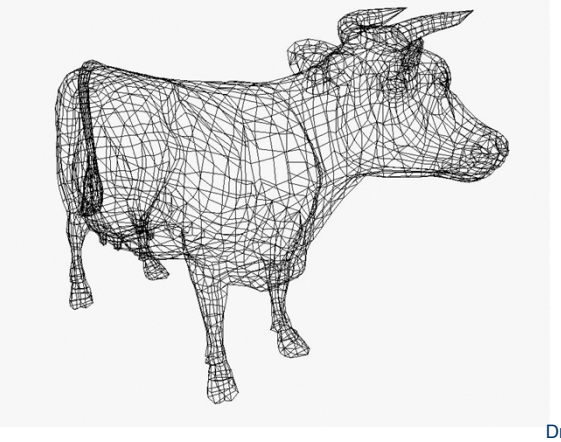




## Robinett's Discussion - Problems

- Incorrect convergence
  - optical axes not parallel
  - optical axes do not pass through centre of screens
- Accommodation and convergence not linked
  - not much can be done about this
- FOV incorrect
  - physical FOV and geometric FOV don't match
- Geometric COP doesn't match optical COP
  - need off-centre COPs

## Complex 3D Object







## Display Lists

- Until now, draw objects as define them.
  ➔ Once drawn, no way to refer to them again.
- To define a single object and refer to it later.
  ➔ In OpenGL, use a **display list.**

## Code for a Display List

```
#define BOX 1  //Give the display list an ID
. . .
void initBox(void){
    GLfloat side = 50.0;
    glNewList(BOX, GL_COMPILE);
        glBegin(GL_LINE_LOOP);
            glColor3f(0.0, 1.0, 0.0);
            glVertex2f(-side/2.0, -side/2.0);
            glVertex2f(-side/2.0, side/2.0);
            glVertex2f(side/2.0, side/2.0);
            glVertex2f(side/2.0, -side/2.0);
        glEnd();
    glEndList();
}
```

Define display list BOX





## Using a Display List

```
void display(void){
    int i;
    glClear(GL_COLOR_BUFFER_BIT);
    for(i = 0; i < 6; i++){
       glPushMatrix(); //Save old model matrix
          //Save other attributes
       glPushAttrib(GL_ALL_ATTRIB_BITS);
          glTranslatef(25.0*i, 0.0, 0.0); //Translate
          glRotatef(45.0, 0.0, 0.0, 1.0); //Rotate
          glCallList(BOX);   //Draw the Box
       glPopAttrib();
       glPopMatrix();

    }
    glFlush();
}
```

## Animation

- To animate a scene, we need to change something about the drawing with each clock tick.
- We can accomplish this with the glut library idle function:

In `main()`:

```
//Function called during idle periods
glutIdleFunc(idle);
```





## The `idle( )` Function

```
void idle(void) {
    theta = theta + 0.1;
    glutPostRedisplay();
}

void display(void){
    glClear(GL_COLOR_BUFFER_BIT);
       glPushMatrix();
       glPushAttrib(GL_ALL_ATTRIB_BITS);
              glRotatef(theta, 0.0, 0.0, 1.0);
              glTranslatef(100.0, 0.0, 0.0);
              glCallList(BOX);
       glPopAttrib();
        glPopMatrix();

    glFlush();
    glutSwapBuffers(); //Double buffering
}
```

`theta` is changed by `idle( )` with each clock tick

## Double Buffering

- Draw into one buffer while displaying the other, then swap the two.
- To guarantee that a scene is displayed only after the drawing is finished.

In `main()`:

```
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
```

In `display()`:

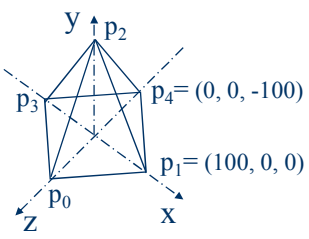
```
glutSwapBuffers(); //Double buffering
```





## Drawing in 3D

- Use 3D points instead of 2D points.
- Example: 3D pyramid.



## A 3D Pyramid

```
void display(void){ //Set up array of 3D vertices
    typedef GLfloat point3[3];
    point3 vertices[5] = { {0.0, 0.0, 100.0},
       {100.0, 0.0, 0.0},  {0.0, 100.0, 0.0},
       {-100.0, 0.0, 0.0}, {0.0, 0.0, -100.0} };
    int i;
    glBegin(GL_TRIANGLES);        //Draw pyramid
    glColor3f(1.0, 0.0, 0.0);
       for (i=0; i<3; i++){       //First face
          glVertex3fv(vertices[i]);
       }
    glColor3f(0.0, 1.0, 0.0);
       glVertex3fv(vertices[0]); //Second face
       glVertex3fv(vertices[2]);
       glVertex3fv(vertices[3]);
       . . .
```





## Hidden Surface Removal

- To remove hidden surfaces:

In `main()`:

```
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB |
                                    GLUT_DEPTH);
glEnable(GL_DEPTH_TEST);
```

In `display()`:

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

## Vertex Arrays

- A vertex array specifies the order of vertices to be drawn in a given set of drawing commands.
- To access it, use a pointer to the index of the first vertex to be drawn.

In `main()`:

```
glEnableClientState(GL_VERTEX_ARRAY);
glVertexPointer(3, GL_FLOAT, 0, vertices);
```





## Setting Up a Vertex Array

- Create an array of all vertices needed:

```
GLfloat vertices[]={0.0,0.0,100.0, 100.0,0.0,0.0,
0.0,100.0,0.0, -100.0,0.0,0.0, 0.0,0.0,-100.0};
```

($p_0$ = 0.0,0.0,100.0; $p_1$ = 100.0,0.0,0.0; $p_2$ = 0.0,100.0,0.0; $p_3$ = -100.0,0.0,0.0; $p_4$ = 0.0,0.0,-100.0)

- Provide an ordered list of vertices by index in the array that specifies each polygon to be drawn. A 3D pyramid, requires 16 vertices altogether.

```
GLubyte pyramidIndices[16] = {0, 1, 2, 0, 2, 3, 3,
  2, 4, 4, 2, 1, 0, 3, 4, 1}
```

## Drawing the Vertices

- To draw using a vertex array

```
glDrawElements(type, n, format, pointer);
```

In `display()`:

```
glDrawElements(GL_TRIANGLES, 12,
               GL_UNSIGNED_BYTE,pyramidIndices);
glDrawElements(GL_QUADS, 4, GL_UNSIGNED_BYTE,
               pyramidIndices[12]);
```





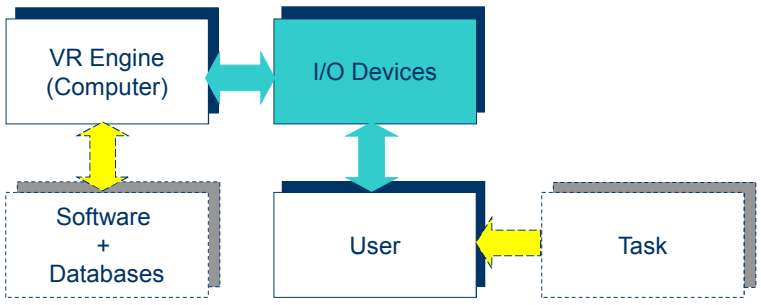
## Recap

- Using OpenGL library
  - Camera, projection, view
  - Stereoscopy
  - Display lists + 3D objects

# ENSF 545
# Introduction to Virtual Reality

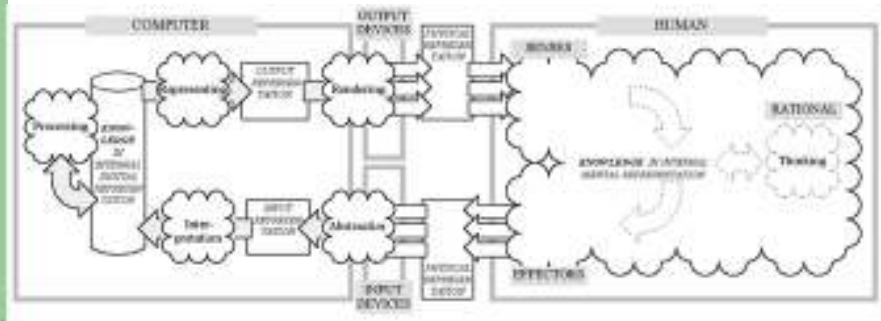
## Output Devices

## A VR System



## Human-Computer Interaction

- Computer outputs ➔ human input (Senses)
- Computer inputs ➔ human output




3

Dr. Y. Hu

## Output Devices for VR

- Graphics displays
  - visual feedback
- Haptic interfaces
  - force and touch feedback
- 3D audio hardware
  - localized sound
- Smell and taste feedback ???

4

Dr. Y. Hu

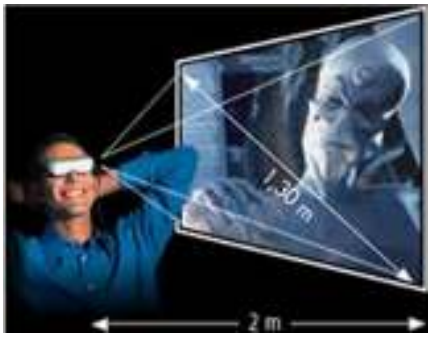
## Graphics Displays





## Graphics Displays – Definition

- A computer interface presents synthetic world images to one or several users interacting with the virtual world.

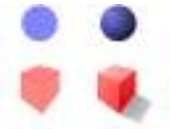



Olympus Eye Trek Face Mounted Display Optics

## Depth Cues in Human Vision

Relative size

Lighting, shadows



Linear perspective

Camera focus, depth of field

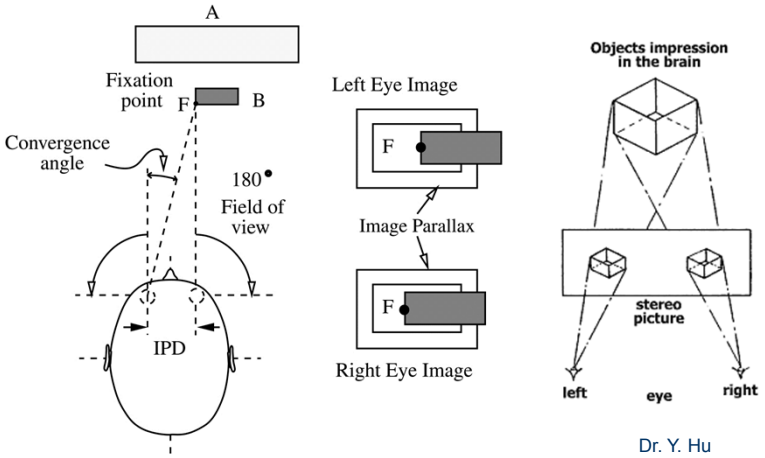
Overlaying

Speed

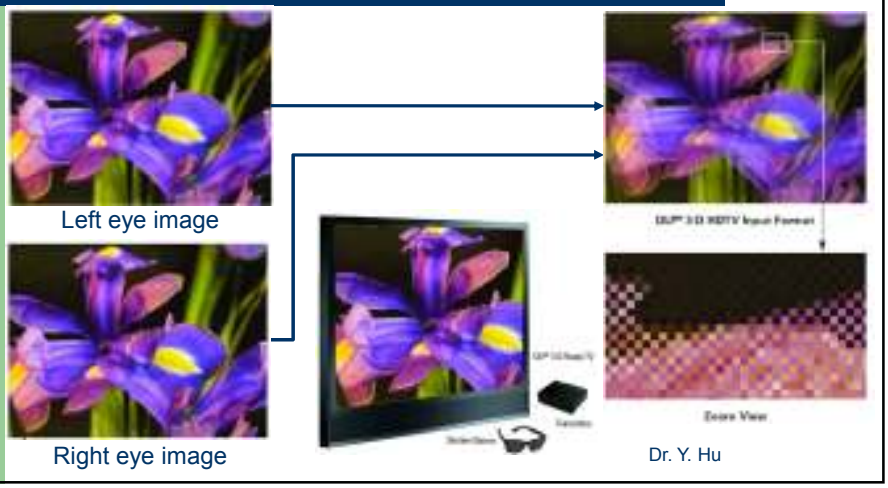




## Human Visual System



## Stereoscopy ←→ 3D HDTV




9

Dr. Y. Hu

## Consideration - Stereo Displays

- For what purposes is a stereoscopic display suitable?
- How to present 2 images of the same VR environment?
- How to deal with image discontinuity?
- What are the cost and availability?
- How does a stereo display differ from another in quality and performance?

10

Dr. Y. Hu

## Graphics Displays – Types

- Images
  - Stereoscopic, monoscopic
- Display technology
  - LCD- and CRT-based, projector-based
- Volume
  - Personal displays
  - Large volume displays





## Personal Displays

- Definition:
  - A graphics display that outputs a virtual scene destined to be viewed by a single user
- Types (stereoscopic):
  - Head-mounted displays (HMD)
  - Hand-supported displays (HSD)
  - Floor-supported displays
  - Autostereoscopic displays

# Head Mounted Displays (HMD)

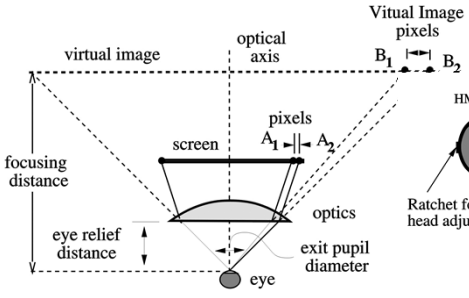
- Project images floating some 1~ 5 m in front of the user (one)
- Display technology
  - LCD
  - CRT
  - Organic LEDs
- Resolution
  - 800 x 600, 2400x 1729

Dr. Y. Hu

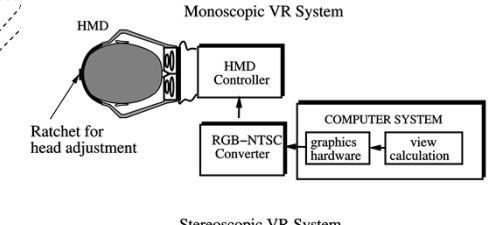
13

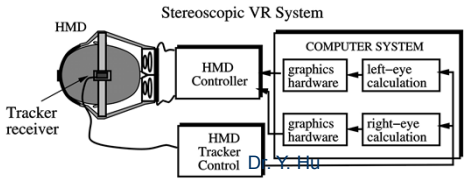
HMD: How to present 2 images for stereoscopy ??

# HMD in a VR System




Simplified HMD Optics Model



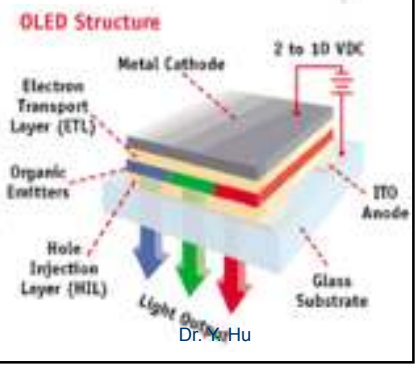




Dr. Y. Hu

14

## Organic LEDs (OLED)

- Light emitting polymer
- No need of a backlight
- Pixel-wise on/off
- No intrinsic limitations to the pixel resolution
- Low power consumption
- Paper thin and bendable
- Effective manufacturing
- Degradable




15

Dr. X. Hu

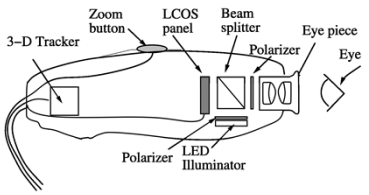
## OLED HMDs

- 5DT
  - 800x600, stereo
  - $40^0$ diagonal view
  - 600 grams, $4k
- Samsung Emagin z800
  - 800x600, stereo
  - $360^0$ pan + $60^0$ pitch (tracking)
  - 8 oz, $1.2k
- Sensics *piSight* panoramic
  - 2400x1729, binocular overlap $82^0$
  - $179^0$ horizontal + $58^0$ vertical
  - 1 kg, $???


17

Dr. X. Hu

## Supported Displays




- Virtual binocular
- Floor supported displays

18

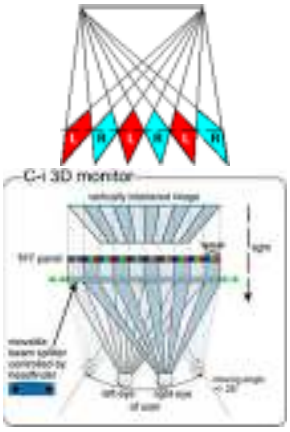
## Autostereoscopic Displays

- Not need of special glasses or other viewing aids
  - Passive displays
  - Active displays
- Display technologies
  - Parallax barrier
  - Lenticular sheets
- Resolution
  - 640x1024 to 1600x1200

19

# Types - Autostereoscopic Displays

- Passive display
  - Not track the user's head
  - The user's eyes in a small area to perceive 3D effect
- Active display
  - Track the user's head and performs a real-time adaptation of the column interlaced images

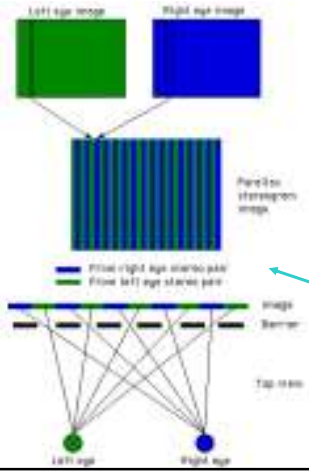



Dr. Y. Hu

20

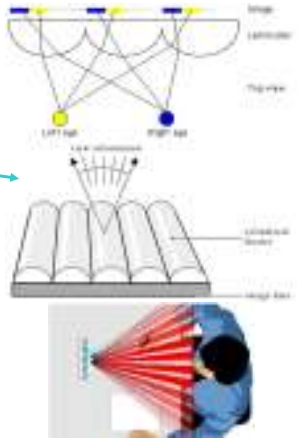
Autostereoscopic displays:
How to present 2 images for stereoscopy ??

# Parallax Barrier + Lenticular Sheets



Lenticular Sheets

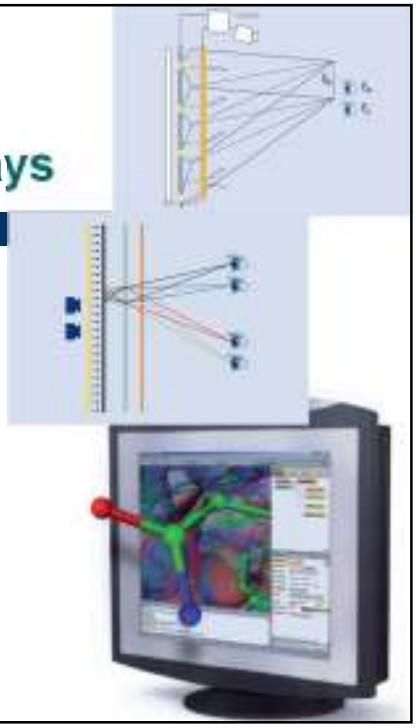
Parallax Barrier



21

## Multi-user Auto-stereoscopic Displays

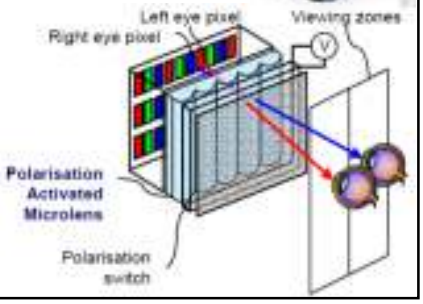
- Tracking pupils of multiple users
- Good resolution but showing some flicker
  - OLED might help to eliminate flicker
  - Higher display frequency (100 ~ 120 Hz)



22

## Autostereoscopic Cell Phone!

- TTPCom Technologies → InTouch
  - 2.1” transflective 2D/3D TFT-LCD
  - RGB displaying 132x176 pixel
  - Automatic switching 2D/3D
  - Running TTPCom WGE 3D stereo game demos




23

## Holographic Displays

- Objects appears to float in space via a 9 optical layer glass panel
- Bare-hand 3D interaction
- Incorporation of IR cameras and image processing board


24

## Comparison of Displays

*Table 3.1* Performance comparison of various personal graphics displays

| Display name | Type | Resolution (pixels) | FOV (H × V) | Weight (grams) | Price $10^3$ |
|---|---|---|---|---|---|
| Olympus "Eye-treck" | AMLCD FMD200 | 267×225 | 30°× 23° | 100 | .5 |
| Daeyang "Cy-visor" | LCOS LCD FMD | 800×600 | 60°× 43° | 160 | 1 |
| Keiser "ProView XL35" | AMLCD HMD | 1024×768 | 28°×21° | 992 | 20 |
| n-vision "Datavisor" | CRT HMD | 1280×1024 | 78°×39° | 1,587 | 35 |
| n-vision "V. Binoculars" | CRT HSD | 1280×1024 | 42° diagonal | 907 | 13.5 |
| Fakespace Labs "Boom3C" | CRT FSD | 1280×1024 | 85°× | N/A | up to 100 |
| Virtual Research "WindowVR" | Flat panel FSD | 1280×1024 | 21" diagonal | N/A | 13.9 |
| DTI "Virtual Window" | TFT LCD autostereo | 1280×1024 2D 640×1024 3D | 18.1" diagonal | 11,250 | 7 |
| Elsa "Ecomo4D" | TFT LCD autostereo | 1280×1024 2D 640×1024 3D | 18" diagonal | 17,000 | 15 |

25

## Large Volume Displays

- Definition
  - Graphics displays that allow several users located in close proximity to simultaneously view an image of the virtual world
- Active or passive glasses
- Classifications
  - Monitor-based
  - Projector-based (predominant)

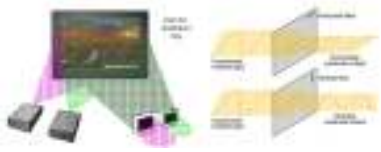




## 3D Glasses – Passive and Active

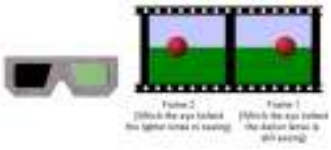
- Anaglyph glasses

- Polarized 3D glasses



- Shutter glasses (active)

- Pulfrich 3D Glasses

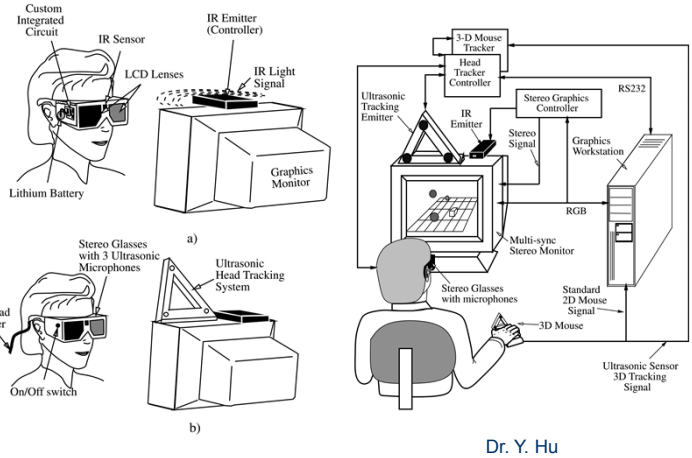


Displays using active glasses:
How to present 2 images for stereoscopy ??

# Monitor-based Displays

Untracked Wireless

Tracked Wireless








# Side-by-Side Monitors

Panoram PV 290

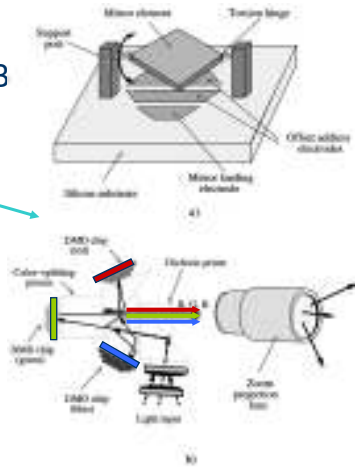
Resolution:
3840 x 1024
Dimension:
1,11 m x 0.29 m

## Image Synchronization


32

## Projector-based Displays

- Old technology
  - ➔ CRT-based 3 tubes (R, G, B
- Recent technology
  - ➔ *Digital Micro-mirror Device*
    - Workbench-type displays
    - Cave-type display
    - Wall-type displays
    - Domes



33

# Workbench-type Displays

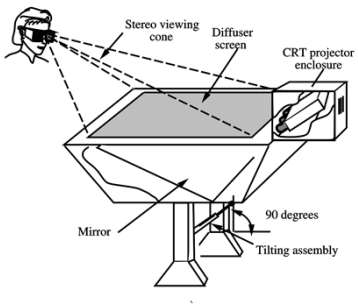
Fakespace “ImmersaDesk”

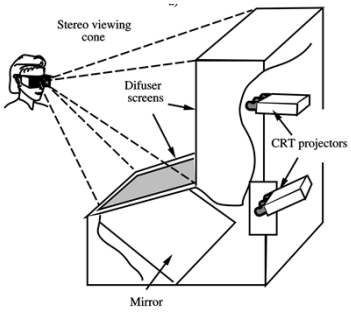
BARCO “Baron”





# Workbench Geometries



## New Types of Displays

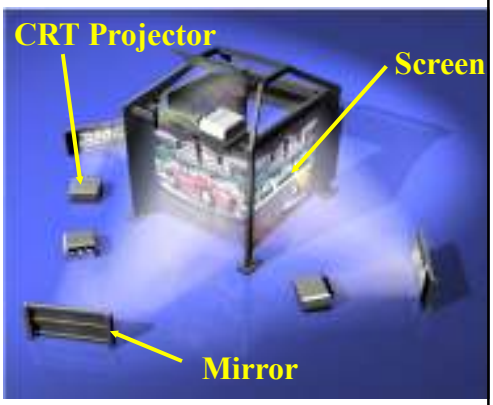
BARCO trace

Microsoft SURFACE

36

## Cave-type Displays - CAVE

- CAVE (Computer-automated Virtual Environment)
- Invented at the Electronic Visualization Laboratory, the University of Illinois

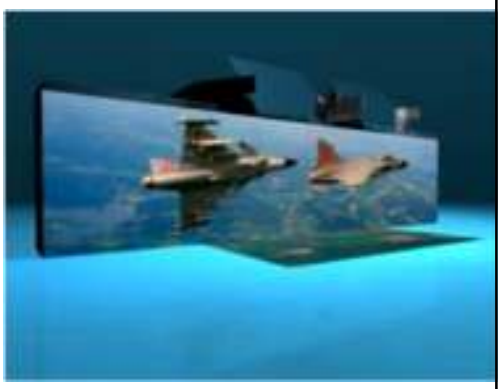



Dr. Y. Hu

37

## Cave-type Displays - RAVE

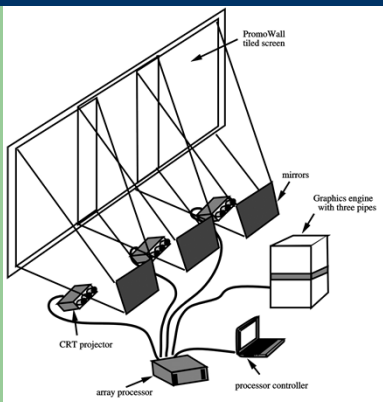
- RAVE (Re-configurable Virtual Environment)
- Various viewing
  - flat wall
  - angled theater
  - CAVE
- Several minutes to reconfigure



38

Dr. Y. Hu

## Wall-type Display




39

Dr. Y. Hu

## Pros and Cons of Wall-type Displays

- Advantages:
  - Accommodate more users
  - Give users more freedom of motion
- Disadvantages:
  - Large cost
  - Much lower resolution than for CRTs
  - More projects for more numbers of pixels/unit





## Comparison of Displays

Table 3.2 Comparison of various large-volume graphics displays

| Display name | Type | Resolution ($10^6$ pixels/m$^2$) | Image size (m$^2$) | Number of users | Price $\times 10^3$$ |
|---|---|---|---|---|---|
| StereoGraphics "CrystalEyes" | active glasses | 18.2 | 0.36× 0.2 | 4 approx. | 2.6 |
| Panoram PV290 | 3-panel monitor | 12.2 | 1.11× 0.29 | 3 approx. | 25 |
| Barco Baron | tilt workbench | 1.9 | 1.36× 0.71 | 4 approx. | 80 |
| Trimension L-shaped V-Desks | L-shaped workbench | 1.0 | 1.36× 1.73 | 4 approx. | 173 |
| Fakespace Workroom | 4-wall CAVE | 0.1 | 3.0 × 3.0×4 | 12 | 300 |
| Fakespace RAVE | modular CAVE | 0.2 | 2.3× 2.4×4 | var. | 500 |
| Panoram PanoWall | Wall (3 proj.) | 0.2 | 7.11× 2.13 | var. | 300 |
| Trimension V-Dome | Dome (7 proj.) | 0.009 | 21 diameter | 400 | 2,172 |

Note: Price does not include the computer driving the simulation

## Consideration - Stereo Displays

- How to present 2 images of the same VR environment?
- How to deal with image discontinuity?
- For what purposes is a stereoscopic display suitable?
- What are the cost and availability?
- How does a stereo display differ from another in quality and performance?





## Haptic Displays

## Haptic Displays

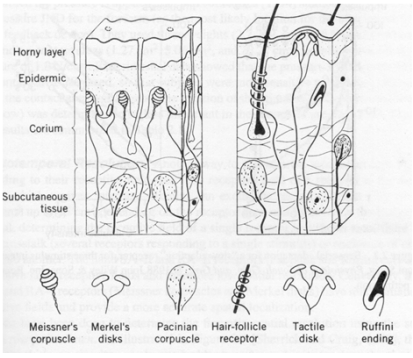
- Haptics:
  - the sense of touch (from Greek Hapthai)
- Tactile feedback:
  - conveys real-time information on surface geometry, surface roughness, slippage, and temperature
- Force feedback:
  - provides real-time information on surface compliance, object weight, and inertia





## Human Touch

- The hand:
  - Most touch sensors
- Four primary sensors:
  - Meissner's corpuscles
  - Merkel's disks
  - Pacinian corpuscles
  - Rufini corpuscles



# Skin Sensors

**Table 3.3** Comparison of various skin mechanoreceptorss

| Receptor Type | Rate of Adaptation | Stimulus frequency (Hz) | Receptive Field | Function |
|---|---|---|---|---|
| Merkel Disks | SA-I | 0–10 | Small, well defined | Edges, intensity |
| Ruffini Corpuscles | SA-II | 0–10 | Large, indistinct | Static force, skin stretch |
| Meissner Corpuscles | FA-I | 20–50 | Small, well defined | Velocity, edges |
| Pacinian Corpuscles | FA-II | 100–300 | Large, indistinct | Acceleration, vibration |

Based on Seow [1988], Cholewiak and Collins [1991], and Kalawsky [1993]

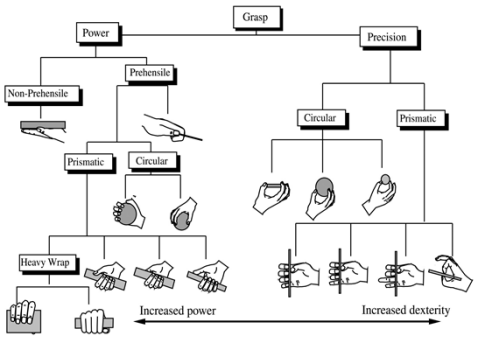
SA-I: Slow adaptation, high spatial resolution; SA-II: Slow adaptation, low spatial resolution
FA-I: Fast adaptation, high spatial resolution; FA-II: Fast adaptation, low spatial resolution





# Maximum and Sustained Force

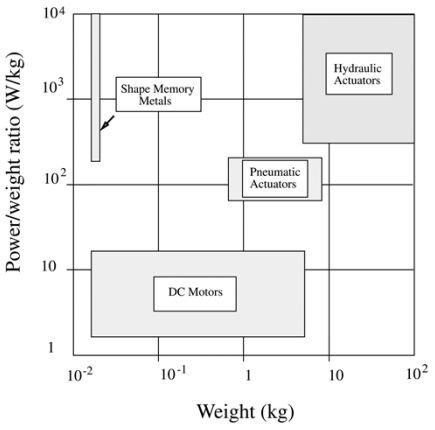
- Maximum force
  - in “power” grasp
  - 400 N (male), 225 N (female)
  - 50 N (finger joint), 100 N (shoulder)
- Sustained force
  - much smaller than maximum



## Haptic Feedback Actuators

- Good power/weight ratio;
- High power/volume ratio;
- High bandwidth;
- High dynamic range (fidelity);
- Safe for the user

➔ None actuator technology satisfies all requirements








## Consideration – Haptic Displays

- How to differentiate tactile feedback from force feedback?
- For what purposes is a haptic display suitable?
- What are the cost and availability?
- How does a haptic display differ from another in quality and performance?

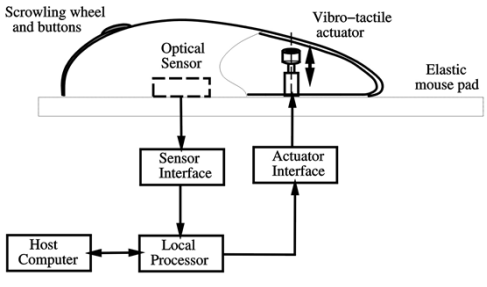
## Tactile Feedback Interfaces

- Desk-top or wearable (gloves);
  - Touch feedback mouse;
  - CyberTouch glove;
  - Temperature feedback actuators;

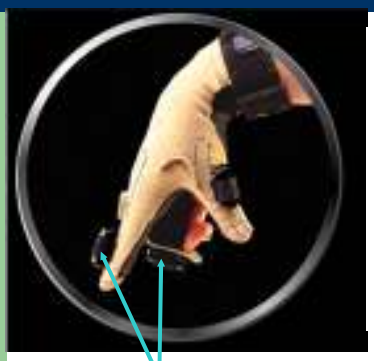


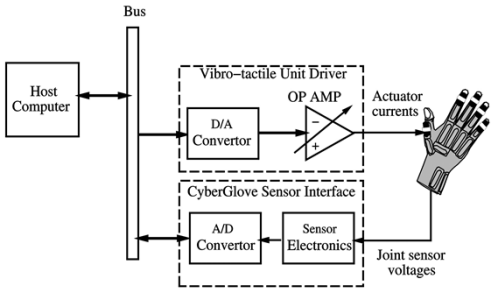


## Tactile Feedback - iFeel Mouse



## Tactile Feedback - CyberTouch Glove





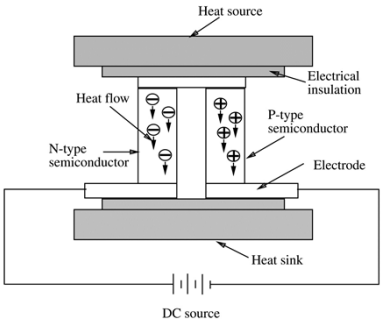

Vibrotactile actuators, 0-125 Hz frequency, 1.2 N at 125 Hz

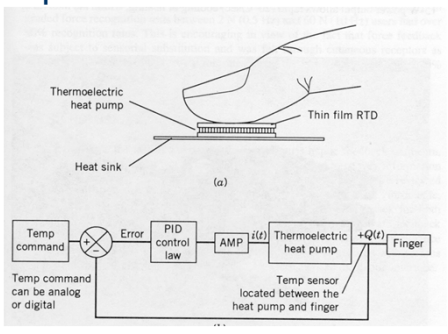




## Tactile Feedback - Temperature

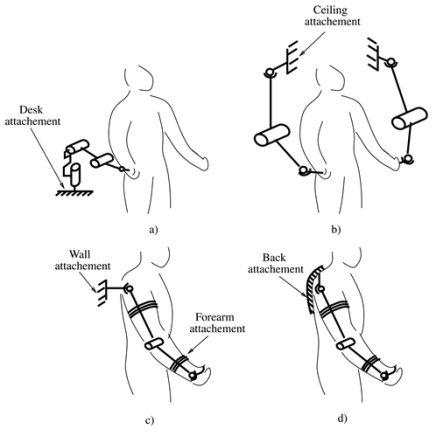
- Simulate surface thermal “feel”
- Use thermoelectric pumps



# Force Feedback Interfaces

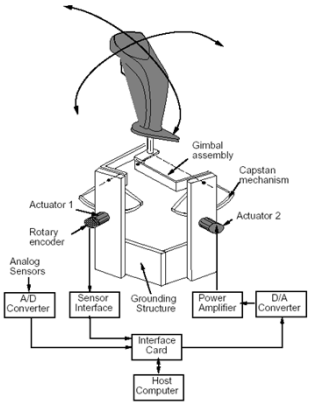
- Mechanical grounding to resist user motion
  - Force joystick
  - Stylus-style
  - Exoskeleton






Dr. Y. Hu

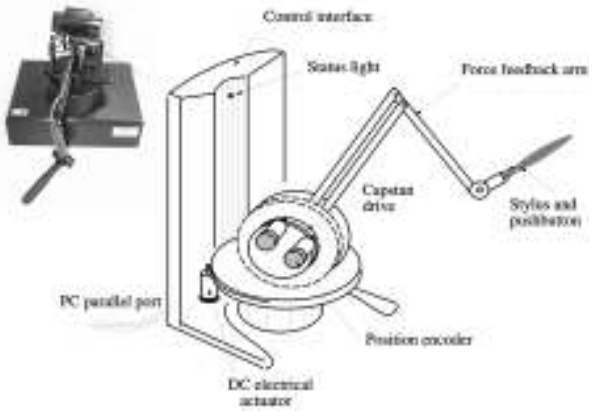
# Force Feedback - Joystick






Dr. Y. Hu

## Force Feedback - Stylus-style (Serial)




SensAble PHANToM

Dr. Y. Hu

56

## Comparison of PHANToMs

| Model | The PHANTOM Desktop Device | The PHANTOM Omni Device |
|---|---|---|
| |  |  |
| Force feedback workspace | ~6.4 W x 4.8 H x 4.8 D in<br>~ 160 W x 120 H x 120 D mm | ~6.4 W x 4.8 H x 2.8 D in<br>~ 160 W x 120 H x 70 D mm |
| Footprint (Physical area the base of device occupies on the desk) | 9 5/8 W x 7 1/4 D in<br>~245 W x 184 D mm | 8 5/8 W x 8 D in<br>~168 W x 203 D mm |
| Weight (device only) | 6 lb 5oz | 3 lb 15 oz |
| Range of motion | Hand movement pivoting at wrist | Hand movement pivoting at wrist |
| Nominal position resolution | > 1100 dpi<br>~ 0.023 mm | > 450 dpi<br>~ 0.055 mm |
| Backdrive friction | < 0.23 oz (0.06 N) | <1 oz (0.26 N) |
| Maximum available force at nominal (orthogonal arms) position | 1.8 lbf. (7.9 N) | 0.75 lbf. (3.3 N) |
| Continuous available force (24 hrs.) | 0.4 lbf. (1.75 N) | > 0.2 lbf. (0.88 N) |
| Stiffness | X axis > 10.8 lb/in (1.86 N/mm)<br>Y axis > 13.6 lb/in (2.35 N/mm)<br>Z axis > 8.6 lb/in (1.48 N/mm) | X axis > 7.3 lb/in (1.26 N/mm)<br>Y axis > 13.4 lb/in (2.31 N/mm)<br>Z axis > 5.9 lb/in (1.02 N/mm) |
| Inertia (apparent mass at tip) | ~0.101 lbm. (45 g) | ~0.101 lbm. (45 g) |
| Force feedback | x, y, z | x, y, z |

57

## Force Feedback - Stylus-style (Parallel)

Omega 3DOF

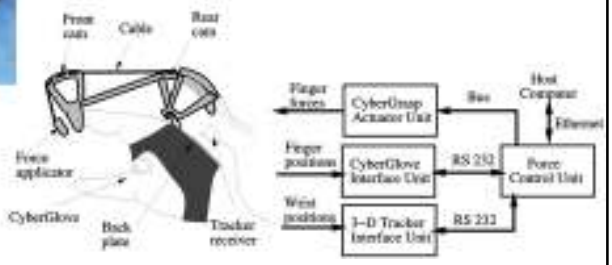
Delta 6DOF

Novint Falcon 3DOF

58

Dr. Y. Hu

## Force Feedback - Exoskeleton

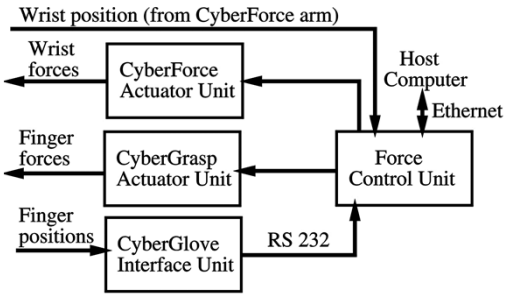




CyberGrasp

59

Dr. Y. Hu

# Force Feedback - Exoskeleton




CyberForce





# Comparison of Haptic Displays

Table 3.5 Haptic interfaces for the hand

| Product Name | Type of feedback | Number actuators | Maximum Force (N) | Weight (grams) | Bandw. (Hz) | Price ($10^3$$) |
|---|---|---|---|---|---|---|
| iFeel Mouse | vibro-tactile | one | 1.1N @30 Hz | 132 | 0–500 | 0.04 |
| CyberTouch glove | vibro-tactile | six | 1.2N @125 Hz | 142 | 0–125 | 15 |
| DTSS X10 | tempe-rature | up to eight | NA | 340 | na | 20 |
| WingMan 3D joystick | force | two | 3.3 | na | 0–333 | 0.06 |
| PHANToM Desktop | force | three | 6.4 | 75 | 7 | 16 |
| PHANToM 1.5/6.0 | force | six | 8.5 | 90–188 | 15 (est) | 57 |
| Haptic Master | force | three | 250 | na | 10 | 34 |
| CyberGrasp glove | force | five | 16 | 539 | 40 | 39 |
| CyberForce arm | force | eight | 8.8 (transl.) | na | 7 | 56 |

## Consideration – Haptic Displays

- How to differentiate tactile feedback from force feedback?
- For what purposes is a haptic display suitable?
- What are the cost and availability?
- How does a haptic display differ from another in quality and performance?





## Sound Displays

## Sound Displays

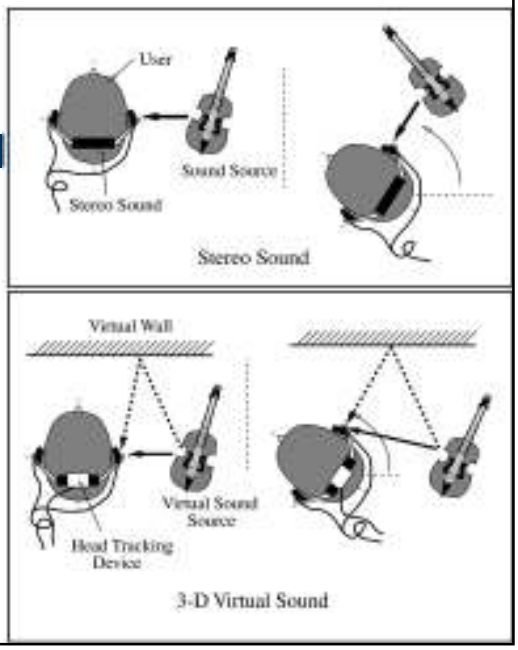
- Synthetic sound feedback to users interacting with the virtual world
- Sound types
  - Monoraural
  - Binaural → Stereo
- Increase the simulation realism

➔ Reading

64

Dr. Y. Hu

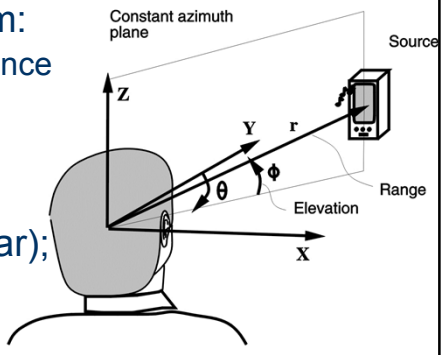
## Stereo vs. 3D sound




65

## Human Hearing Model

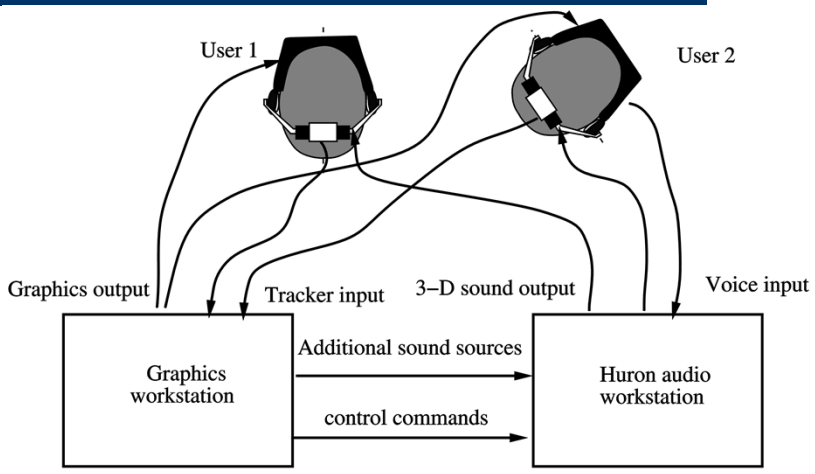
- Polar coordinate system:
  - azimuth, elevation, distance (range);
- Azimuth cues;
- Elevation cues;
- Effect of pinna (outer ear);
- HRTFs (head related transfer functions)



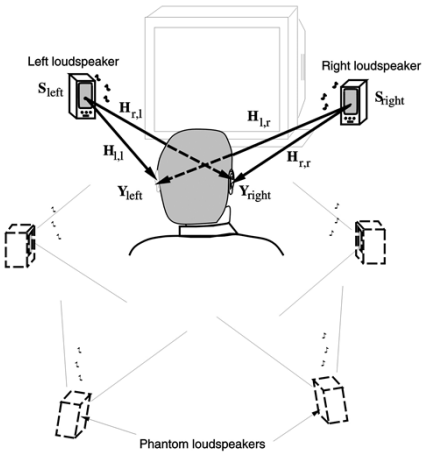





## The Huron Workstation



# 3-D Audio Displays


68

# Commercial 3D Sound Cards

- What they have to offer:
  - Digital output
  - Multi-speaker compatibility → 7.1 channel format allows for 8 speakers
  - Positional audio → offers 3D dimensions of sound
- Two main audio APIs
  - DirectSound 3D (DS3D) → Microsoft ‘s DirectX component
  - Aureal 3D (A3D) →An extension of DS3D

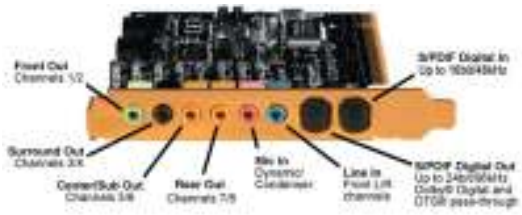
69

Dr. Y. Hu

# Creative Labs Sound Blaster Audigy 4 Pro

- Creative Labs Sound Blaster Audigy 4 Pro

- Turtle Beach Montego DDL 7.1 Sound Card







# Comparison of 3D Sound Cards

| Name | Chip/3Dsound engine/API | In/Out | SR |
|---|---|---|---|
| Creative Sound Blaster Audigy 4 Pro | CA10200 ICT DSP/CreativeWare/A3D 1.0, EAX Advanced HD 4 | 7.1-analog out; 5.1-digital out (DIN) ; 2-digital in/out (coaxial); 2-digital in/out optical ac3/dts pass-thru | $299 |
| Philips Acoustic Edge | ThunderBird Avenger/QSound/ A3D 1.0/EAX 2.0 | 5.1-analog out; 2-digital in/out (coaxial); ac3/dtz pass-thru | $100 |
| Turtle Beach Montego DDL 7.1 | EAX 1 and 2, A3D, I3DL2 and DirectSound 3D | 7.1-analog out; Optical S/PDIF In/Out; audio resolutions 24 bit (out) 16 (in); sample rates 96kHz (out) and 48kHz (in). | $80 |

## Conclusion + Recap

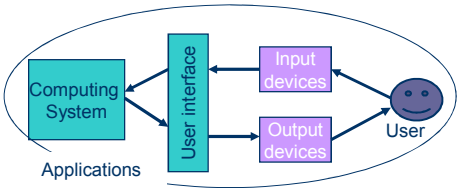
- All output devices aim at stimulating the user's senses in real time.

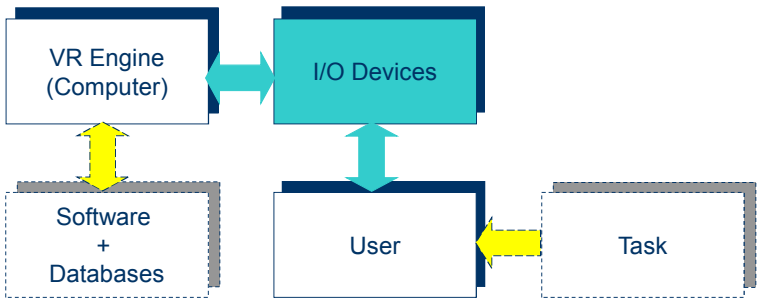
- Graphics displays
- Haptic feedback
- 3D audio feedback (reading)
- No smell and taste feedback

# ENSF 545
# Introduction to Virtual Reality

## Input Devices

## A VR System

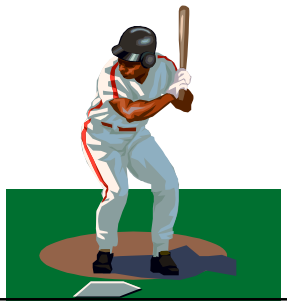


## What is *VR*?

- A high-end user interface that involves real-time simulation and interaction through multiple sensorial channels.
  - Vision
  - Sound
  - Touch
  - Smell
  - Taste

**Object**



3

## Input Devices for VR

- Trackers
  - mechanical, magnetic, optical, ultrasound, hybrid inertial, vision-based
- Navigation interfaces
  - cubic mouse, trackball, 3D probe
- Gesture interfaces
  - pinch glove, data glove, cyberGlove

4

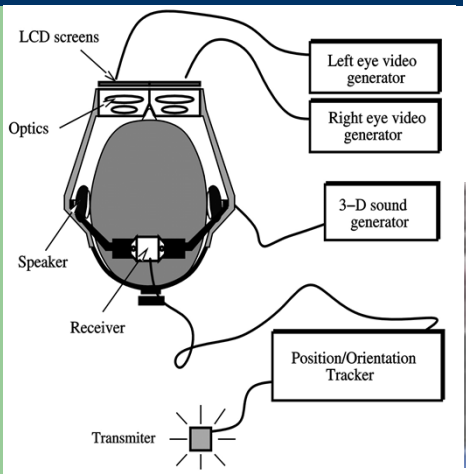
Dr. Y. Hu

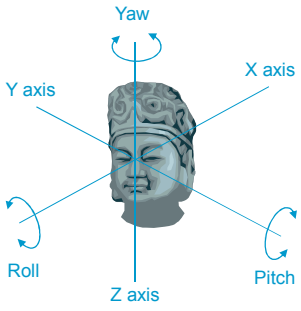
# Trackers



Dr. Y. Hu

# Trackers



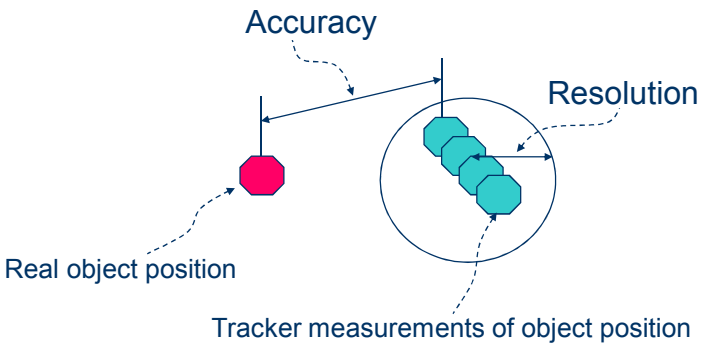

## Tracker Characteristics

- Measurement accuracy (errors)
- Sensor noise and drift
- Sensing latency
- Update rate – Readings/sec
- Measurement repeatability
- Tethered or wireless
- Work envelope
- Sensing degradation

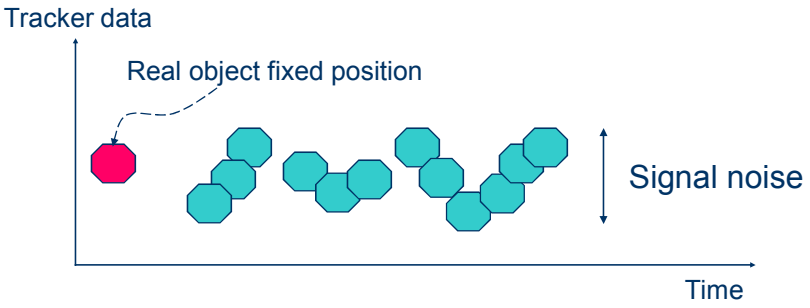




## Tracker Accuracy



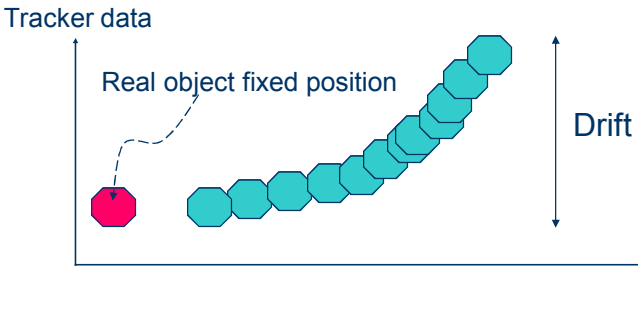
## Tracker Noise




10

Dr. Y. Hu

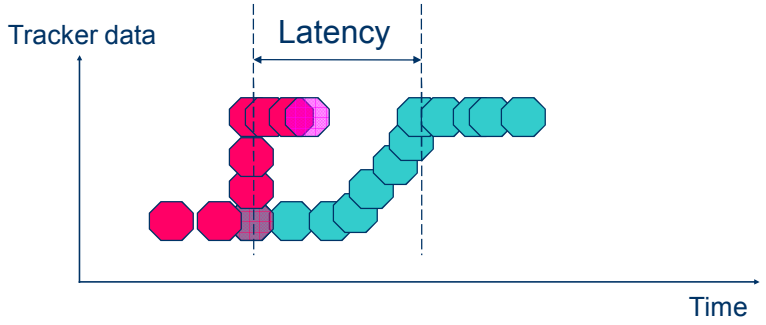
## Tracker Drift




11

Dr. Y. Hu

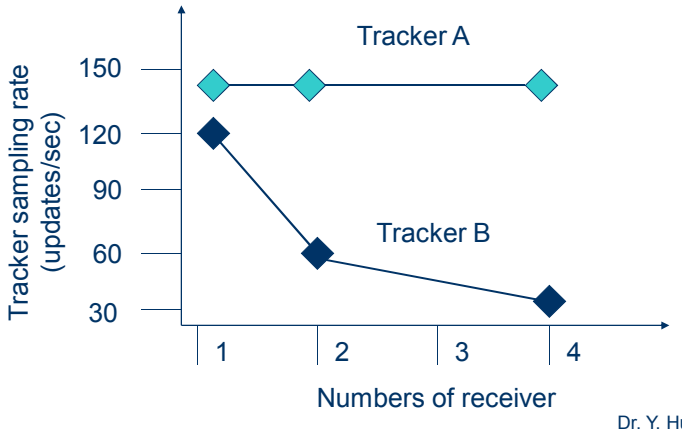
## Tracker Latency






Dr. Y. Hu

## Tracker Update Rate

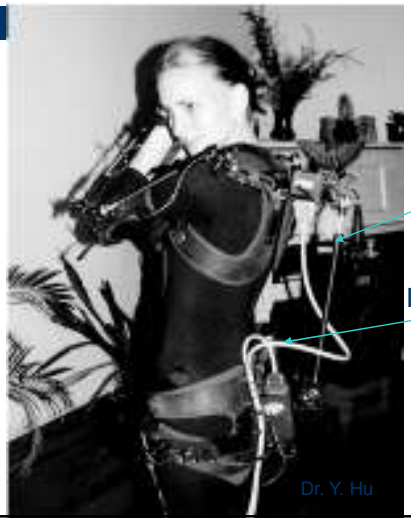





Dr. Y. Hu

## Mechanical Trackers

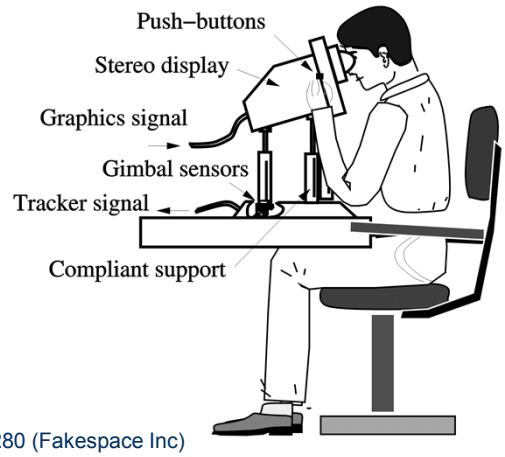
A serial or parallel *kinematic structure* composed of links interconnected using *sensorized* joints.




14

Dr. Y. Hu

## Mechanical Trackers




15

Push 1280 (Fakespace Inc)

Dr. Y. Hu

## Mechanical Trackers - Pros + Cons

- Measure position using imbedded sensors
- Have extremely low latency
- Be immune to interference from magnetic fields

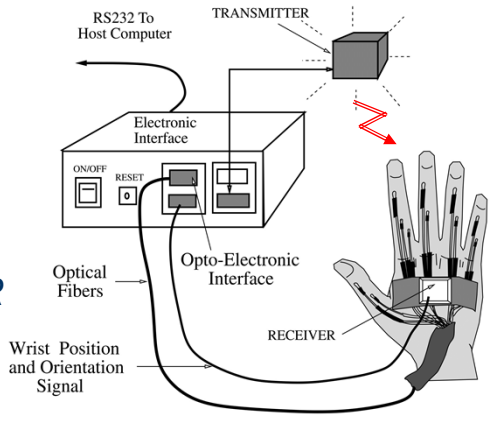
- Constrain the user's freedom of motion
- Can be heavy if worn on the body





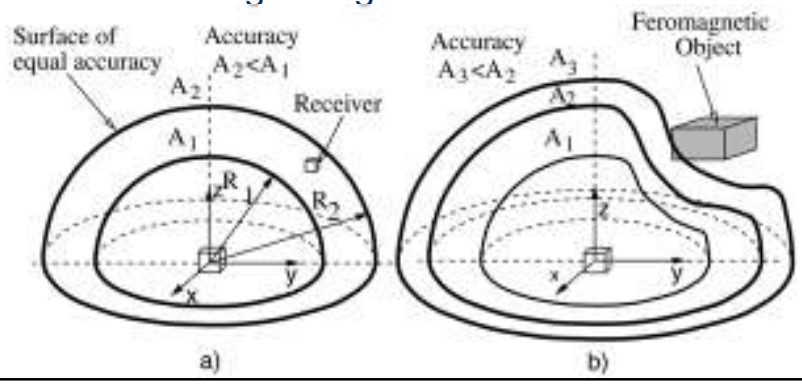
## Magnetic Trackers

- A magnetic field produced by a stationary *TRANSMITTER* to determine the real-time position of a moving *RECEIVER* element.
- AC + DC trackers.



## Accuracy Degradation

- Errors in position and orientation
- Size of errors growing from source outwards




18

## Magnetic Tracker Errors

- Due to ambient noise:

$$e_{ambient} = K_n \, (d_{transmitter\text{-}receiver})^4$$

- Due to metal:

$$e_{metal} = \frac{K_r \, (d_{transmitter\text{-}receiver})^4}{(d_{transmitter\text{-}metal})^3 \times (d_{metal\text{-}receiver})^3}$$

19

Dr. Y. Hu

## Magnetic Trackers - Pros + Cons

- Measurement in 6 DOF ← low-frequency magnetic fields
- Fields ← a fixed transmitter
- A tracked object ← the receiver

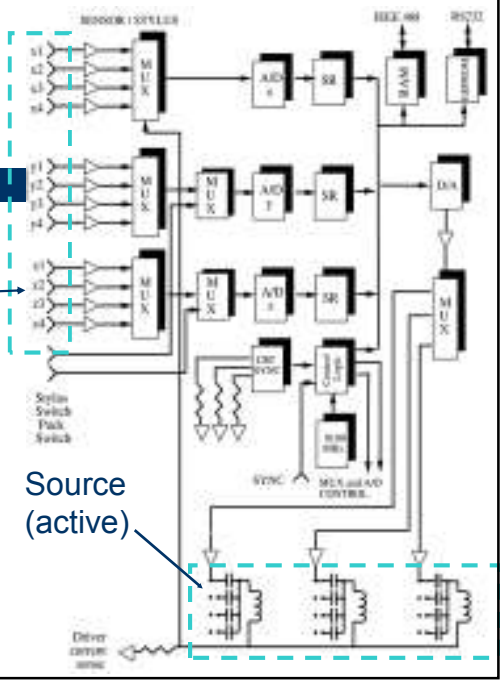
- Larger work envelop → larger transmitter
- Distance → the voltages induced in the receiver (coils) → calibration…

20

Dr. Y. Hu

## AC Tracker - Fastrack

Sensors (passive)



Source (active)

21

# AC Tracker - Wireless

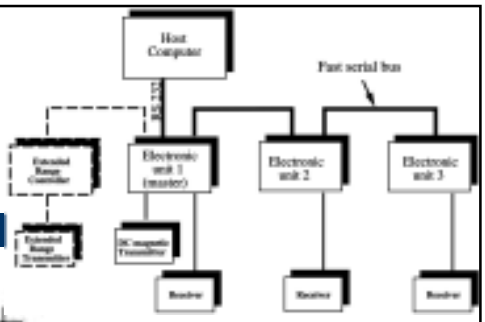
Polhemus - PATRIOT

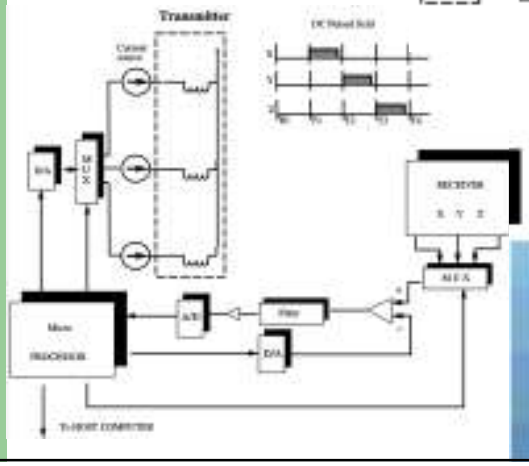
Polhemus - Liberty LATUS

Dr. Y. Hu

22

# DC Tracker - Flock of Birds

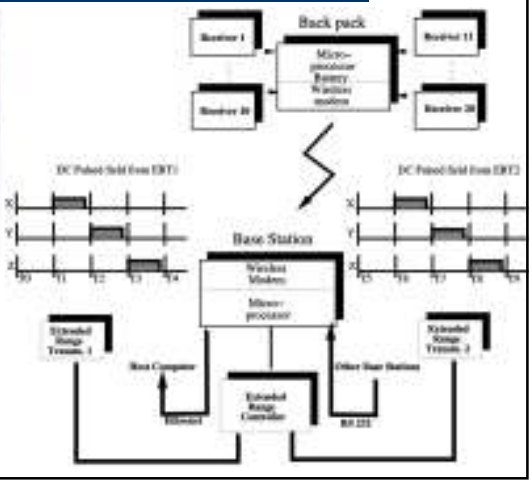



Tracker @ iCentre


23

## DC Tracker - Wireless




Ascension Technology

24

## Comparison - Trackers: AC vs. DC

| | AC | DC |
|---|---|---|
| Non-ferromagnetic metals (aluminum, brass, stainless steel) | affected | immune |
| Ferromagnetic metals (mild steel, ferrite) and copper | affected | affected |
| Resolution and accuracy | better | OK |
| Work envelop | smaller | OK |

Dr. Y. Hu

25

# Trackers: AC vs. DC

Table 2.1 Performance comparison: Fastrack vs. Flock of Birds

| SPECIFICATION | FASTRACK | FLOCK OF BIRDS |
|---|---|---|
| Operation radius: | | |
| normal | 0.75 m (30") | 1.2 m (48") |
| extended | 2.25 m (90") | 3 m (120") |
| Angular range | all-attitudes | ±180° Azimuth & Roll<br>±90° Elevation |
| Transl. accuracy | 0.03"RMS | 0.1"RMS |
| Transl. resolution | 0.0002"/inch range | 0.03"RMS |
| Angular accuracy | 0.15°RMS | 0.5°RMS |
| Angular resolution | 0.022°RMS | 0.1°RMS at 12" |
| Update rate (measurements/sec) | 120 (one receiver)<br>60 (two receivers)<br>30 (four receivers) | 144<br>(up to 30 receivers) |
| Latency (msec) (single receiver) | 8.5 (no filtering) | 7.5 (no filtering) |
| Metal Interference | Ferrite<br>Mild Steel<br>Copper<br>Stainless steel<br>Brass<br>Aluminum | Ferrite<br>Mild Steel<br>Copper |
| Interface | RS-232 (selec. baud rates to 115,200 or IEEE-488 up to 100 kbaud/sec) | RS-232 (selec. baud rates to 115,200) or RS-422/485 (selec. baud rate to 500,000) |
| Data format | ASCII or Binary | Binary |
| Modes | Point or stream | Point or stream |

26

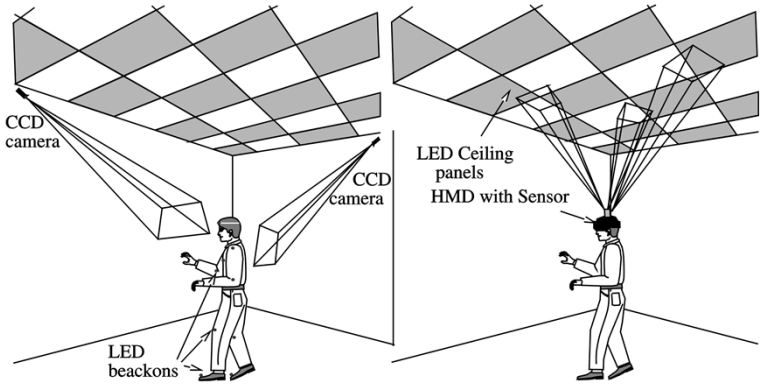
# Optical Trackers

- Optical sensing to determine the position and orientation of a moving object in real time

Dr. Y. Hu

27

## Two Major Configurations



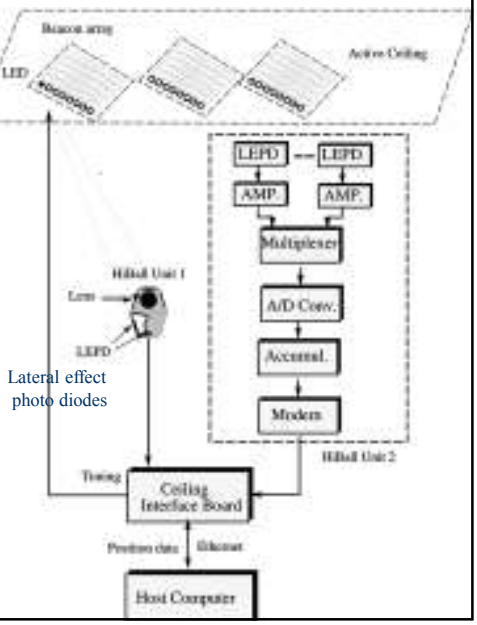

28

Dr. Y. Hu

## Outside Look-in - Vicon MX

29

## Inside Look-out




HiBall 3000

30

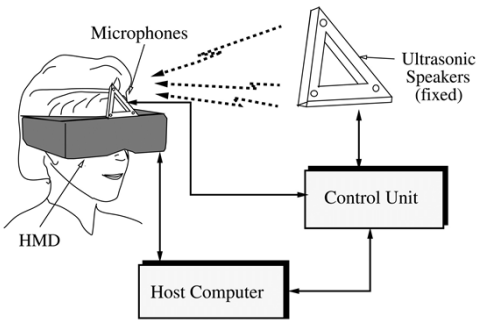
## Magnetic and Optical Trackers

| | Magnetic | Optical |
|---|---|---|
| Degrees of freedom | 6 | 3,3(interpret) |
| Metal effects | yes | no |
| “Light-in-sight” | no | yes |
| Sampling rate and accuracy | <150sets/s ~1.0 mm | >500sets/s ~0.5 mm |
| Work envelop | small | large |

31

## Ultrasonic Trackers

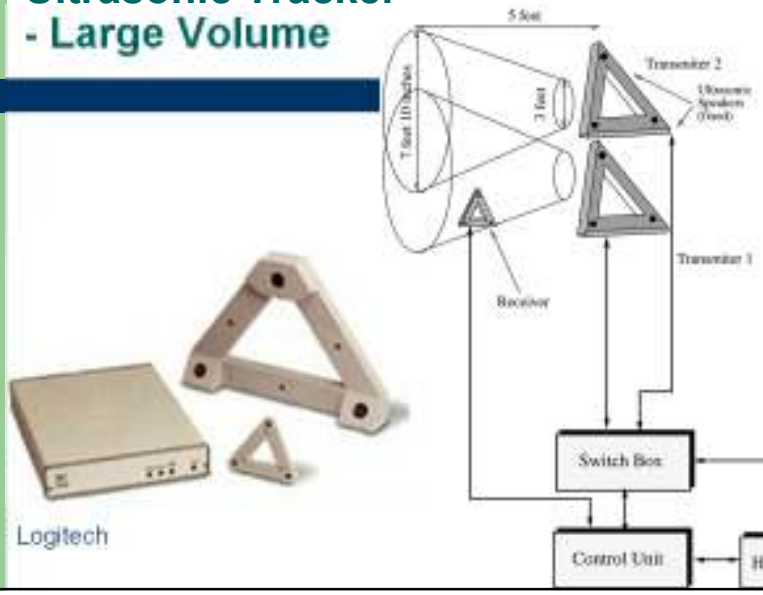
- An ultrasonic signal produced by a stationary transmitter to determine the real-time position/orientation of a moving receiver.








## Ultrasonic Tracker - Large Volume




Logitech

## Ultrasonic Trackers (Pros + Cons)

- Position ← low-frequency ultrasound
- Sound ← a fixed triangular source (speakers)
- A tracked object ← a triangular receiver

- Larger work envelope → more sources
- Distance → the sound time of flight
- Accuracy → air temperature and noise sources
- Tracking → "direct line of sight"





## Hybrid Trackers

- Combining 2 or more technologies to track object better than a single technology would allow
- Inertial / ultrasonic
- Inertial / vision

- **Inertial trackers**: self-contained sensors that measure the rate of angular change in an object orientation
  - Gyros

# Hybrid Tracker – IS 900

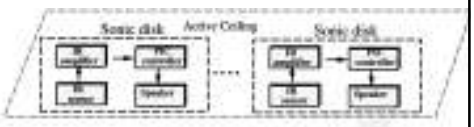


Intersense Co.

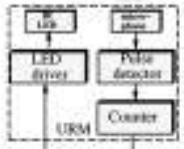
36

Dr. Y. Hu

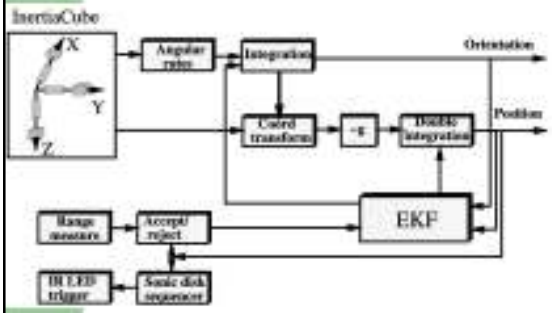
# Hybrid Tracker - IS 900

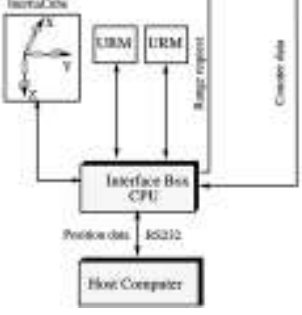


Block Diagram




Software




Intersense Co.

37

Dr. Y. Hu

## Inertial / Ultrasonic Trackers (Pros + Cons)

- Metallic objects + magnetic fields ← no interference
- Tracking ← large-volume + full room
- Orientation tracking ← "source-less"

- Accelerometer errors → decreased accuracy
- Time elapses → errors grow
- Gyroscope errors + position errors → worse, calibration

38

Dr. Y. Hu

## Hybrid Tracker – Inertial / Vision

IS 1200

Fiducials

InertiaCAM fixed-focus lens

Intersense Co.

39

Dr. Y. Hu

# Performance Comparison

40

Dr. Y. Hu

# Navigation Inputs

41

Dr. Y. Hu

## Navigation Input Devices

- Allow the interactive change of the view to the virtual environment and the exploration through the selection and manipulation of a virtual object of interest.
  - Position/velocity control of virtual objects
  - Either absolute or relatives coordinates
  - Cubic Mouse, Trackball, 3D probe

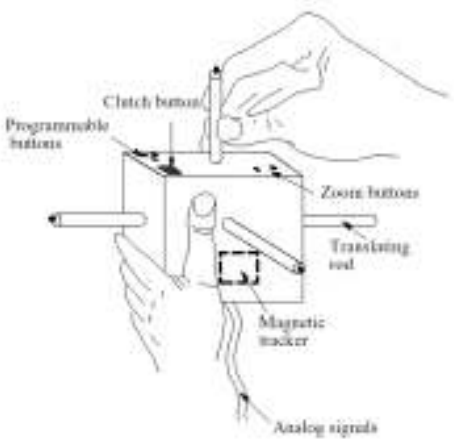




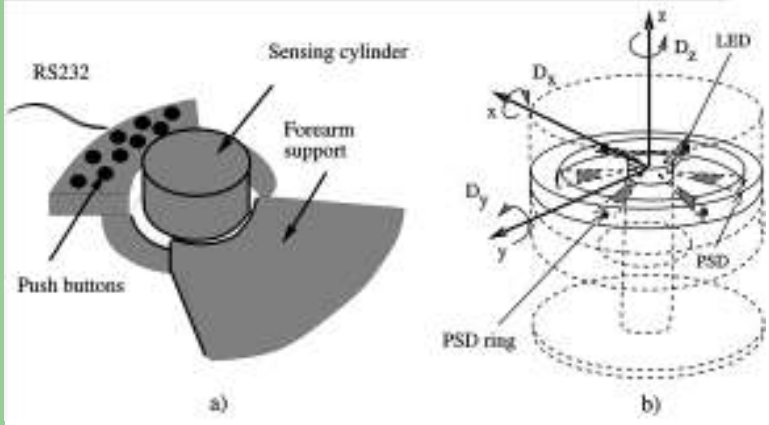
## 3D Cube

Wand @ iCentre

Magnetic tracker



# Trackball

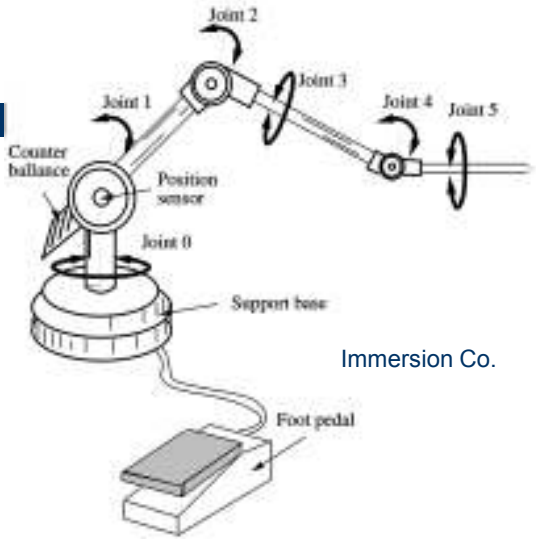





Dr. Y. Hu

# 3D Probe

SensAble PHANToM




Immersion Co.



Dr. Y. Hu

## Gesture Inputs



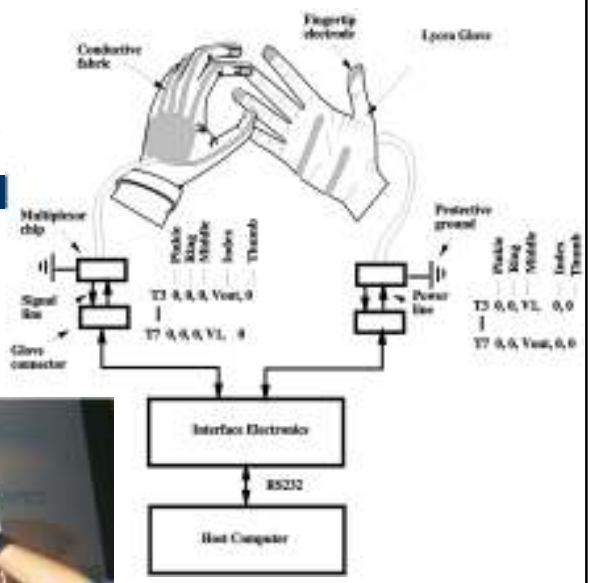


## Gesture Input Devices

- Allow dextrous control of virtual objects and interaction in real time through gesture recognition
  - Larger work envelope than trackballs, 3D probes
  - Calibration for user's hand needed
  - Sensing gloves: Pinch Glove, 5DT Data Glove, DG5 glove, CyberGlove

# Pinch Glove

Fakespace



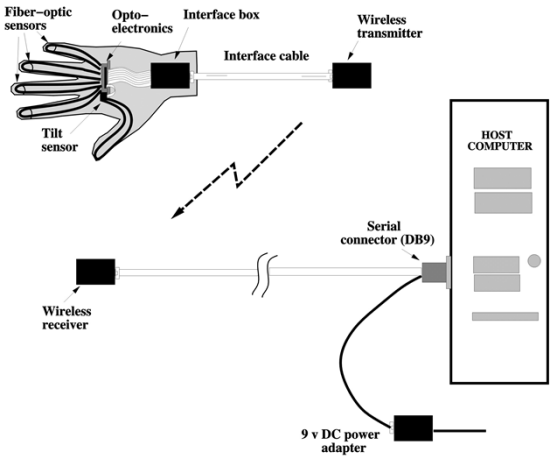



Dr. Y. Hu

# 5DT Data Glove

5 sensors

14 sensors



## DG5 Glove + DG5 VHand Glove 2.0

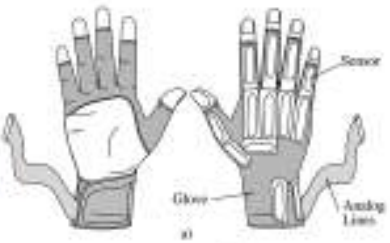
- Angles ← Voltages
- Sensor resolution 10 bit
- Needs calibration
- Less expensive (< $1000)

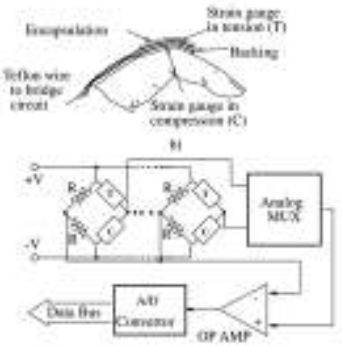
DG5 Vhand Glove 2.0

Dr. Y. Hu

50

## Cyber Glove

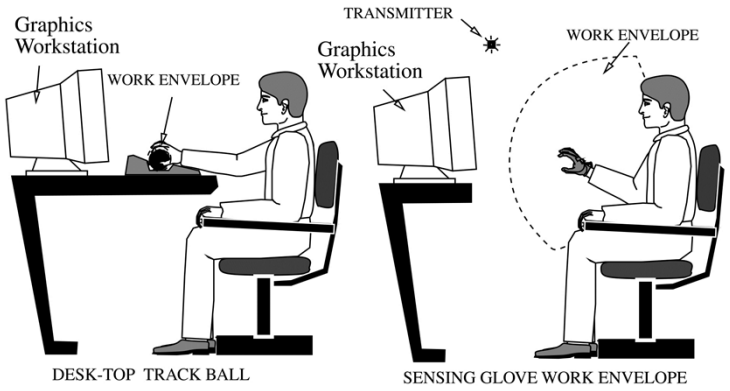





Dr. Y. Hu

51

# Comparison of Work Envelop








# Comparison of Gloves

| Specification | **Pinch Glove** | **5DT Data Glove** | **DG5 Glove DG5 VHand** | **CyberGlove** |
|---|---|---|---|---|
| Number of sensors | 7 / glove (2 gloves) | 5 or 14 / glove (1 glove) | 5 / glove (1 glove) | 18 or 22 / glove (1 glove) |
| Sensor type | Electrical | Fiber-optic | Ink film | Strain gauge |
| Record/sec | N/A | 100(5DT 5W), 200(5DT 5) | 100<br>25 (VHand) | 150(unfiltered)<br>112(filtered) |
| Sensor resolution | 1 bit (2 points) | 8 bits (256 points) | 10 bits (1024 points) | 0.5º |
| Communication rates | Wired (19.2 kb) | Wired(19.2kb), Wireless(9.6kb) | Wired(19.2kb)<br>Wireless (VHand) | Wired (115kb) |
| Wrist sensors | None | Pitch (5DT 5) | Accelerometers (VHand) | Pitch and yaw |

## Conclusion + Recap

- All input devices aim at capturing the user's input in real time and transmitting the input to the host computer running simulation.

- Trackers
- Navigation interfaces
- Gesture interfaces