SE3VR11 – Virtual Reality

"The ultimate display would, of course, be a room within which the computer can control the existence of matter. A chair displayed in such a room would be good enough to sit in. Handcuffs displayed in such a room would be confining, and a bullet displayed in such a room would be fatal. With appropriate programming such a display could literally be the Wonderland into which Alice walked."

Ivan Sutherland (1966)

What is Virtual Reality?

"a way for humans to visualize, manipulate and interact with computers and extremely complex data"

(Aukstakalnis & Blatner, 1992)

"an immersive, interactive experience generated by a computer"

(Pimentel & Teixeira)

"[an] experience...in which the user is effectively immersed in a responsive virtual world"

(Brooks, 1999)

What is Virtual Reality?

"Virtual reality is a high-end user-computer interface that involves real-time simulation and interactions through multiple sensorial channels. These sensorial modalities are visual, auditory, tactile, smell and taste."

(Burdea, 2003)

"a medium composed of interactive computer simulations that sense the participant's position and replace or augment the feedback to one or more senses - giving the feeling of being immersed or being present in the simulation."

(Sherman & Craig)

What is Virtual Reality?

"Language serves not only to express thoughts, but to make possible thoughts which could not exist without it."

(Bertrand Russell)

"Virtual reality is about discovery, about doing things that couldn't be done before, expressing ideas that couldn't be expressed before."

(Gullichsen, in Pimentel & Teixeira)

"Enough of definitions ... they don't help"

(Sir Michael Brady, FRS)

SE3VR11 – Course Structure

Course:

- 20 lectures
  - Introduction by Paul Sharkey (this presentation)
  - 10 by Richard Mitchell on Perception and Systems
  - 8 by Paul Sharkey on 3D Graphics
- Assignment
  - Modelling using Unity, set by Richard Mitchell

SE3VR11 – Course Assessment

Assessment:

- 2 hour examination
  - Answer 3 out of 4 questions
  - Weighted 70%
- Assignment
  - Report and Demo
  - Weighted 30%

VR at Reading

- Researching VR since early 1990s
- Developed Expertise in
  - Distributed VR Systems
  - Massive Multi-user VR Systems (scalable)
  - Collaborative VR
  - VR for Rehabilitation
- Inaugurated Intl Conf. on VR and Disability (1996)
- Awarded £1m grant in 1999 to install the CAVE

Potted History of ICDVRAT

- Inaugurated in 1996
- Disparate group
  - computer geeks, engineers with tech "solutions"
  - psychologists, OTs/PTs, with applications
  - Each speaking English, but not necessarily understanding each other
- Leading to multidisciplinary team approach
- More integrated technology and properly conducted trials

Themes at ICDVRAT

- Virtual, augmented and enhanced environments
- Rehabilitation and training tools for rehabilitation
- Stroke and brain injury rehabilitation
- Cognition and cognitive processing
- Communication, speech and language
- Communication aids
- Virtual environments for special needs
- Haptic devices
- Visual Impairment
- Visual impairment through virtual simulation
- Ambisonics (3D Sound) and acoustic virtual environments
- Mobility and wheelchair navigation
- Multi-user systems for user interaction
- Input devices, sensors and actuators
- Design of virtual environments
- Product design testing and prototyping
- Tools for architectural/CAD design
- Human factors issues
- Clinical assessment Medical systems
- Computer access

Technology, Domain, Application

TECHNOLOGIES	DISABILITY	APPLICATIONS
3D virtual environments	Visual impairment	Access and interaction
Multimedia, Multi-sensory & Acoustic environments	Cognitive impairment	Training and education
Display technologies desktop, projection, HMD, CAVE	Motor impairment	Assessment
	Learning disability	Rehabilitation
	Wheelchair users	Mobility aids
Interaction methods - PC interaction methods - 3D interaction devices and navigation systems - Tangible interfaces - Gesture and eye tracking - Haptic, force-feedback and tactile devices - Augmented reality, embedded technologies		Language and communication Professional use Design/evaluation tools

VR as a Tool for Rehabilitation

- VR as real-time interactive simulation
- Technology developments in VR
- As a viable tool for rehabilitation
- Beyond a visual tool
- Breadth of applications
- VR in the hospital/clinic
- VR in the home?
- Introduction to Course

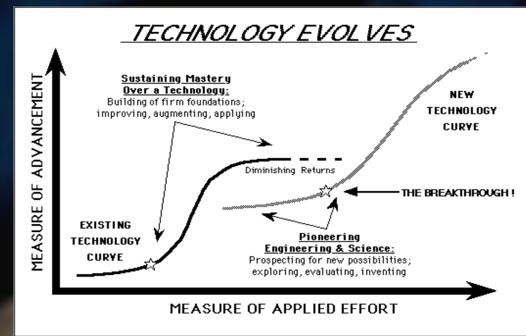
Typical views on virtual reality systems

- Visionary!
- Too Expensive!
- Just what the field needs!
- Where's the science?
- Like the Holodeck
- Need better interfaces
- Hmm…interesting…
- Can they really do that?
- How is it any better than what we already do?

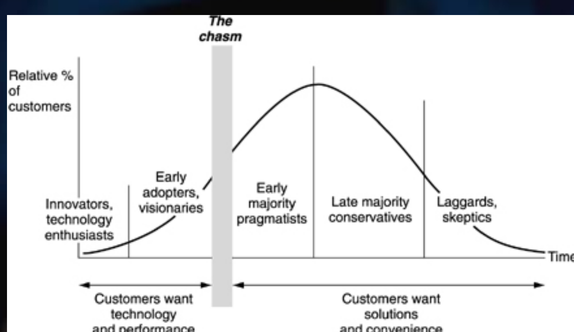
What these views reflect

- Faith in Technology to solve problems
- Fear & Distrust of Technology
- Economics
- Frustration with existing tools
- Popular media influences
- Scientific and pragmatic questions
- Curiosity
- Healthy Skepticism

Technological Evolution …

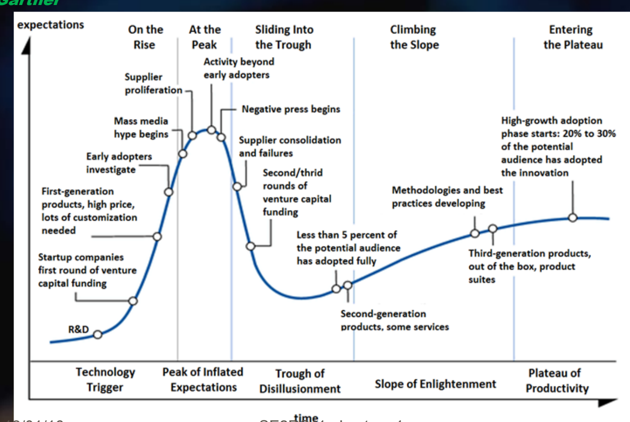


User Perspective …



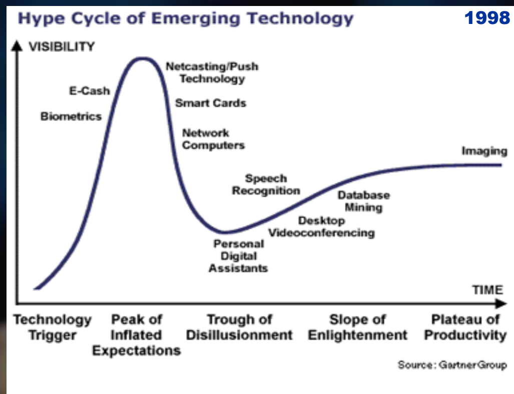
Emerging Technologies Hype Cycle

Gartner

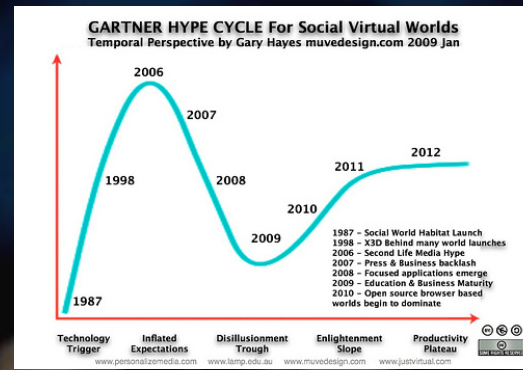


12/01/16 SE2RS11 - Lecture 1

Hype Cycle 1998

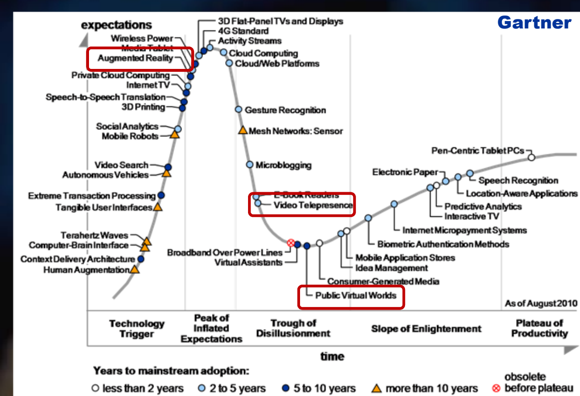


Hype Cycle for Virtual Social Worlds 2009



Hype Cycle 2005

Hype Cycle 2010



Hype Cycle 2015

VR as real-time interactive simulation

- Focus for VR simulation is (amongst others) on
  - human-in-the-loop
    - requires real time response, within msecs
  - immersion of the user
    - not just physical immersion but also emotional
    - believability (presence – immersion beyond the interface)
- The criteria above do not necessarily equate with faithful rendering of physical environments
  - "ecological validity"

Populated VR

- In addition to creating believable environments it is becoming increasingly important for many applications to have populated environments
  - Multiple users
    - represented as avatars or by video
  - Autonomous users
    - 'virtual humans' which can incorporate strong narrative support for the scenario/environment
  - Semi-autonomous Virtual Humans
    - a number of VHs operated by a single user
  - Operated over a network = distributed VR
    - how to maintain consistency in the models

Developments over a decade

- Speed of Computing and Reduced Costs
- VR Application Expansion and Refinements
- The Game Industry
  - Graphics Hardware and Software
  - Display Technology
  - Interaction Devices
- Virtual Humans
- Wider Professional Acceptance
- Library of VR Environments that can be reutilised

Believability … illustrative example

- Training for resuscitation
- "Wizard of Oz" type control
- Look for the following …
  - delays in responses of the virtual human
  - missed ordering, where the virtual human speaks on the phone before she picks it up
  - the change in the user from interacting with a computer simulation to becoming immersed in the scenario

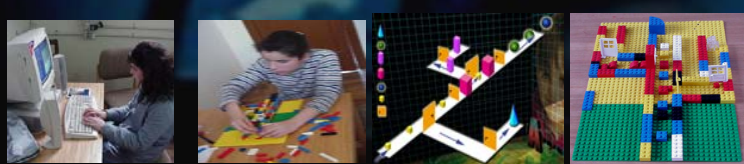
Thalman et al, EPFL, Lausanne

Is there more?

What about senses other than vision?

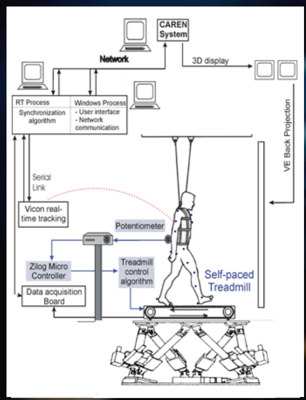
Audio VR

- VR for the blind/visually impaired?
  - Cognitive Impact & Spatial Awareness in Blind Children (Sanchez & Lumbreras)

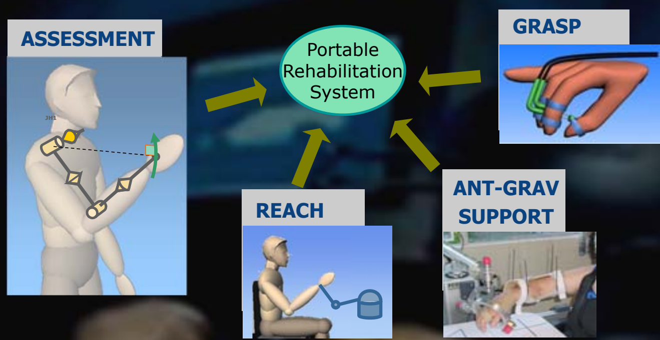


- Training for interaction with guide dogs

Active platforms for gait training



Haptic Devices



Grasp Control

SWOT Analysis for VR

Strengths

- Ecological validity
- Stimulus control and consistency
- Repetitive and hierarchical stimulus delivery possible
- Cueing stimuli for "errorless learning"
- Real time performance feedback
- Self-guided exploration and independent practice
- Stimulus and response modification contingent on user's impairments
- Complete naturalistic performance record
- Safe testing and training environment which minimizes risks due to errors
- Graduated, systematic exposure
- Distraction
- Gaming factors to enhance motivation
- Low cost functional environments that can be duplicated and distributed

Weaknesses

- "Perceived" and Actual Costs
- "Perceived" and Actual Complexity

- Platform Compatibility
- The Interface Challenge

- Display Hardware
- Side Effects

- Front End Flexibility
- Back End Data Extraction, Management

- Wires!

Opportunities

- Processing Power/Graphics/Video Integration
- Academic and Professional Acceptance
- Well-Matched VR Rehab applications also have widespread intuitive appeal to the public

- Close Knit VR Community
- Gaming and Entertainment Industry Drivers
- Game-Based Educational Drivers
- Vision-Based Tracking

- Integration with Imaging and Psychophysiological Approaches
- TeleRehabilitation
- Virtual Humans

Threats

- No Moore's Law Operating in the area of HMDs and other quality peripherals
- Need Cost/Benefit Proofs!

- After effects Lawsuit Potential
- Ethical Challenges

- The Perception that VR Tools will eliminate the need for the expert
- Limited Awareness/Unrealistic Expectations

VR for Anxiety Disorders

- Heights
- Flying
- Driving

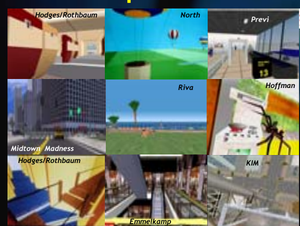
- Spiders/snakes
- Public Speaking
- Claustrophobia

- Generalized Social Phobia
- Panic Disorder with Agoraphobia
- Post Traumatic Stress Disorder

1998

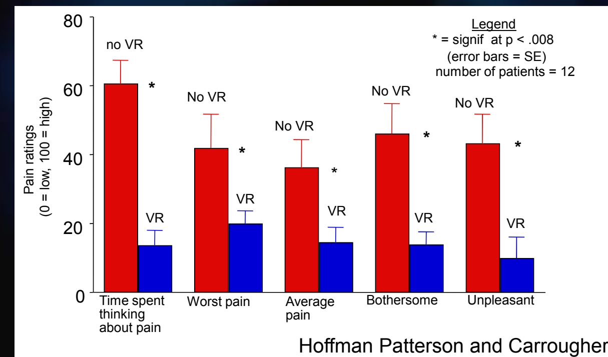
2008

From Simple Phobias to



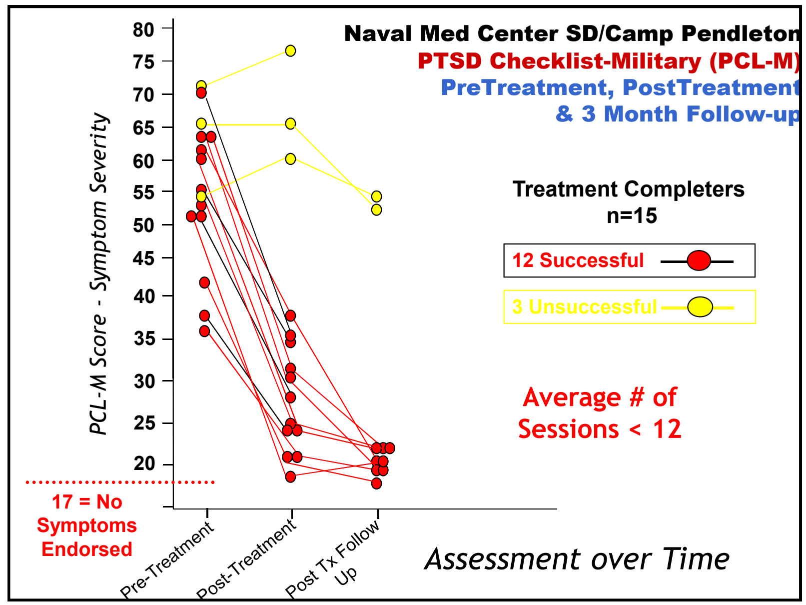
- Addiction
- ADHD
- Alzheimers
- Autism
- Balance Disorders
- Cerebral Palsy
- Neglect
- Pain Distraction
- Phantom Limb
- PTSD
- Stroke, TBI, Parkinsons
- Spinal Cord Injury
- And Many More…

Pain distraction during physical therapy



PTSD in Gulf Veterans

- Graduated Exposure Therapy for Post Traumatic Stress (PTSD)
  - AKA Combat Stress
- Developed from Full Spectrum Warrior
  - Reutilising existing software
  - US content
    - Question: Are Humvees, and not Bedfords, really a problem?
- Now undergoing clinical trials (in the US)
  - A number of positive case studies are emerging for soldiers diagnosed with severe PTSD



PTSD in the clinic … and at home?

- Total cost of equipment = $6,000
  - Patient simulation computer
  - Clinician "Wizard of Oz" controller
  - Accessories
- Capability to deliver therapy to the home
  - Can extend to family members to allow wider understanding of trauma suffered

Some Terminology

As we've established, there are no universal definitions but there are widely used terms, for clarity we'll summarise the most common definitions as:

▪Virtual Reality (VR) – "[Any combination of] artificial sensory stimuli intended to give a user the impression of a physical environment other than that which they inhabit."

▪Virtual Environment (VE) – "The content of a VR simulation irrespective of the way in which the user is made aware of it." (Also Virtual World)

▪Virtual Reality Interface (VRI) "A piece of technology which enables a human user to perceive and/or interact with a VE, usually via the sensory organs."

Note: These definitions are not more 'correct' than any others

FOUR Key Elements of VR

- A Virtual Environment (world)

  an imaginary space often (but not necessarily) manifested through a medium

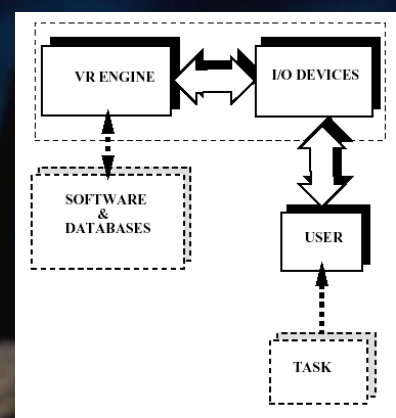
  a computer-based virtual world is the description of objects within a simulation
- Immersion/Presence

  More on this later …
- Sensory feedback

  what kinds? – Lots to choose from
- Interactivity

  respond to user actions – in what way?

Components of a VR System



Components of a VR System I

- Output Devices:
  - Display technologies: Visual, auditory, haptic (touch)
  - Also sometimes: olfactory, gustatory, inertial
- Input Devices:
  - Tracking/Sensing: head movement, body position, force.
  - Interpretation: Voice recognition, Gestures

Components of a VR System II

- VR Engine:
  - How things change with time: physical modelling, collision detection and response
  - Behaviour: AI, animation
- Databases:
  - Object representation: geometry, colour, texture, sound
  - History of events and actions
  - Characters, personalities

SE3VR11 – Topics

Richard Mitchell

2. Presence
3. Visual Perception
4. Visual Display Technologies

5. Auditory Perception
6. Auditory Display Technologies

7. Haptic Perception
8. Haptic Display Technologies

9. Dynamic Virtual Worlds
10. Interaction and Input Devices

SE3VR11 – Topics

Paul Sharkey

1. Introduction (this lecture)

2. 3D Computer Graphics , Displaying Images, 3D Scene Representation
3. Matrix Algebra, Rendering

4. Illumination of scene, light sources and reflection
5. Constructive Solid Geometry

6. Distributed Virtual Reality

SE3VR11 : Virtual Reality

Professor Paul Sharkey p.m.sharkey@reading.ac.uk

Lecture 2 – 21/01/2016

Course Outline

Split into the following topics:

- **2D/3D Graphics**
  - Object Representations
  - Mathematics involved in 3D Graphics
  - Materials and Texturing
  - Lighting and Shading
  - Rendering Methods
  - Animation
- **3D Modelling Approaches**
- **Scene Graphs and Software**
- **(Distributed Virtual Environments)**

Computer Graphics

Computer graphics deals with all aspects of producing an image using a computer

- Imaging - representing 2D images
- Modelling - representing 3D objects
- Rendering - creating an image from models
- Animation - simulating changes over time

It involves

- Hardware
- Software Libraries
- Software Applications

Computer Graphics

What hardware is needed to create this image?

- **Processing Power**
- **Storage / Memory**
- **Input / Output devices**

Computer Graphics

What software is needed to create this image?

- **Libraries to control the hardware**
- **Generating object data**
  (Modelling)
- **Mathematical models for lighting and materials**
  (Shading)
- **Conversion to something that can be displayed**
  (Rendering)

Computer Graphics

Adding animation and interaction to the graphics...

- **Collisions and intersections**
- **Object manipulation**
- **Key framing**
- **Inverse kinematics**
- **Simulations**

A Brief History of Computer Graphics
1950-1960

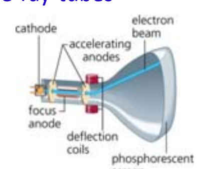
Computer graphics goes back to the earliest days of computing

- Strip charts
- Pen plotters
- Simplistic displays from arrays of lights to 'nixie-tubes'

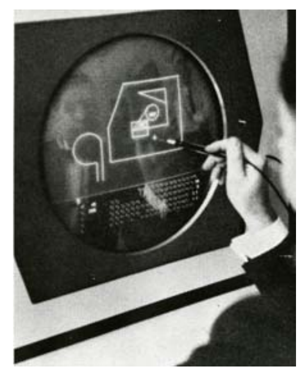
A Brief History of Computer Graphics
1960-1970

Computer graphics starts improving ...

- Wire-frame graphics
- Dedicated display processors
- Storage Tubes
- Cathode-ray-tubes



- Sketchpad – Ivan Sutherland



A Brief History of Computer Graphics
Ivan Sutherland

Ivan Sutherland's PhD thesis at MIT

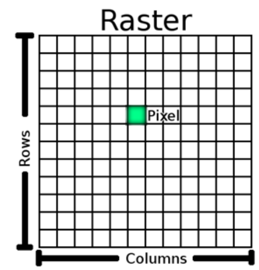
- **Recognition of man and machine interaction**
- **Sketchpad** (predecessor to the GUI)
  - Display image
  - User moves light pen
  - Computer generates new display
  - Loop…

Sutherland created many of the now common algorithms for basic computer graphics

A Brief History of Computer Graphics
1970-1980

Introduction of …

- Raster Graphics



- Graphics standards start forming
- Workstations and PCs

The acceleration of graphics technology really starts to climb

A Brief History of Computer Graphics
1980-1990

Introduction of …

- Special purpose hardware – Silicon Graphics

- Industry standards in place
- Graphics over a network – X Window System
- Human Computer Interfaces (HCI)

A Brief History of Computer Graphics
1990-2000

- **OpenGL API** (application program interface)
- **Computer generated feature length films**
- **Graphics techniques at hardware levels**
  - Texture mapping
  - Blending
  - Stencil Buffers – typically used to add shadows
- **More funky SGI**

A Brief History of Computer Graphics
2000 to present

- Photorealism
- Multi core and multi processors dedicated to graphics
- Programmable pipelines
- Real time ray tracing

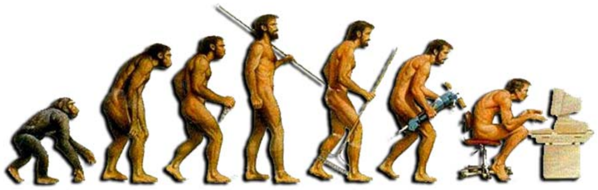
  (...nearly)
- and a LOT more

Computer Graphics … what is it good for?

- Entertainment
- Computer-aided Design (CAD)
- Scientific Visualisation
- Training
- Education
- E-Commerce
- Computer Art
- Virtual Reality
- . . .

Computer Graphics … what is it good for?

- . . .
- wherever your imagination takes you …



Components for 3D Graphics

Outline

- Displaying an Image
  - Hardware
  - Systems
  - Colour Representation
- 3D Scene Representation
  - Intro
  - Coordinate Systems
  - 3D Primatives

Components for 3D Graphics

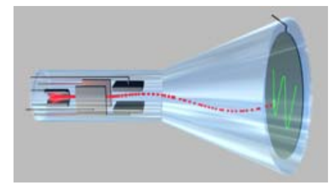
Outline

- **Displaying an Image**
  - **Hardware**
  - Systems
  - Colour Representation
- **3D Scene Representation**
  - Intro
  - Coordinate Systems
  - 3D Primitives

Components for 3D Graphics
Hardware

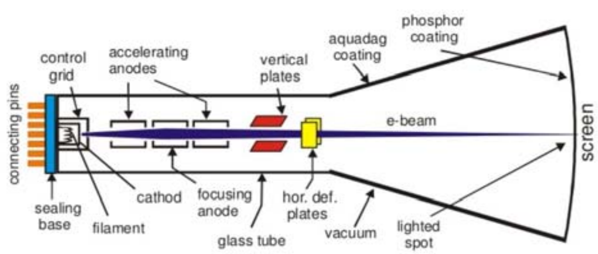
CRT – Cathode Ray Tube





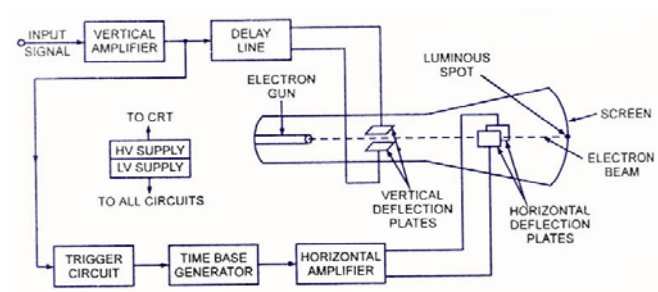
Components for 3D Graphics
Hardware

CRT – Cathode Ray Tube



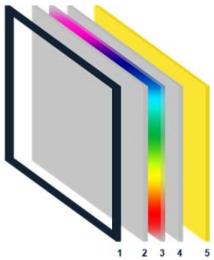
Components for 3D Graphics
Hardware

CRT – Cathode Ray Tube



Components for 3D Graphics
Hardware

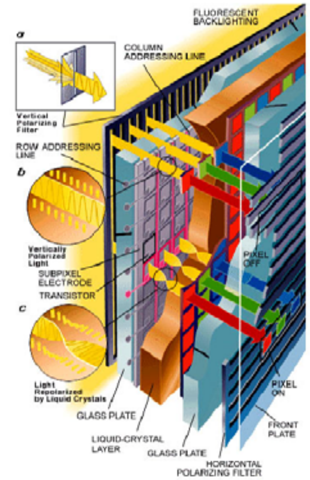
LCD – Liquid Crystal Display



Components for 3D Graphics
Hardware

LCD – Liquid Crystal Display

- Compact in depth
- Limited in size
  - size vs cost
  - getting cheaper every day
- Light is polarized
  - stereo cannot be generated using polarizing glasses



Components for 3D Graphics
Hardware

AMOLED – Active Matrix Organic Light Emitting Diode Display

- Since 2008 in smart phones
  - low power, low cost
  - Smartphones can now be used in HMDs (*see* Net Gear)
- Used increasingly in larger screens

Components for 3D Graphics
Hardware

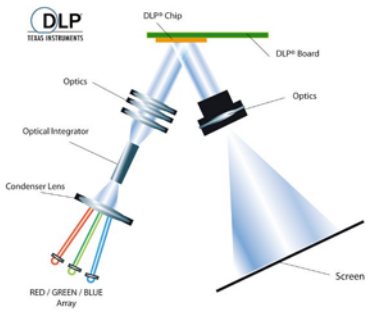
DLP – Digital Light Processing Projectors

- Used for back projected systems
  - Used in the Reading CAVE
  - Low cost (very few >£500)
  - Cost = resolution + Lumens

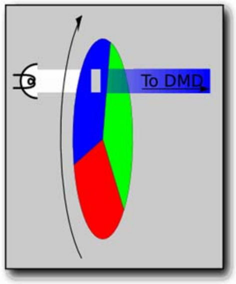
  - (LCDs can also be used in projection systems)

Components for 3D Graphics
Hardware

DLP – Digital Light Processing Projectors

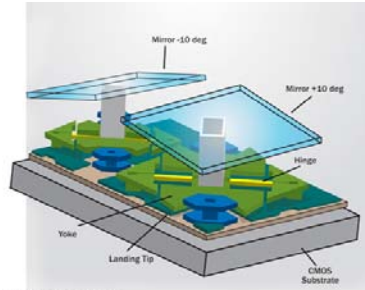


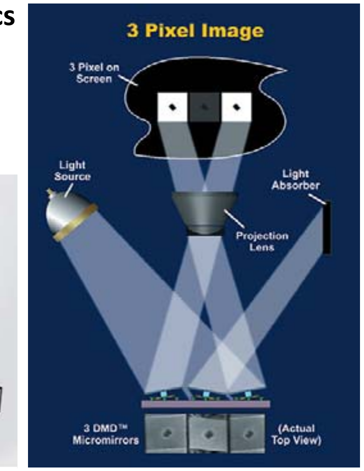
or



Components for 3D Graphics
Hardware

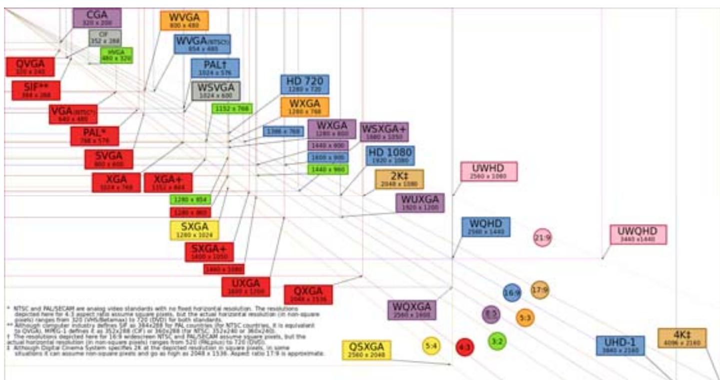
DLP – Digital Light Processing





Components for 3D Graphics
Hardware

Common Display Resolution Standards

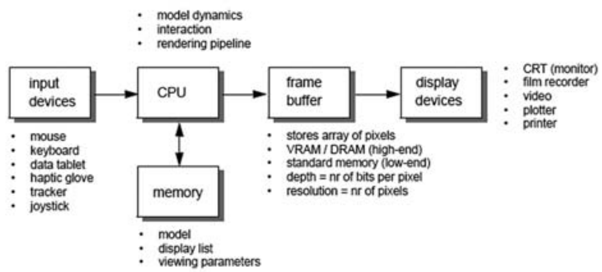


Components for 3D Graphics

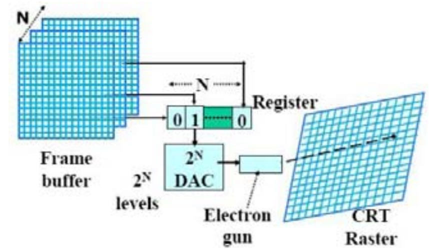
Outline

- **Displaying an Image**
  - Hardware
  - **Systems**
  - Colour Representation
- **3D Scene Representation**
  - Intro
  - Coordinate Systems
  - 3D Primitives

Components for 3D Graphics
Systems
The Frame Buffer

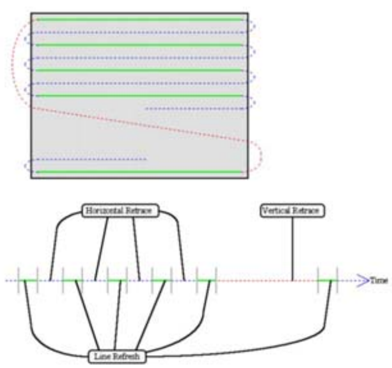


Components for 3D Graphics
Systems
The Frame Buffer

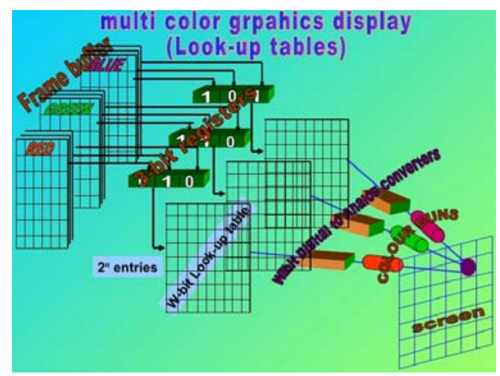


An N- bit plane gray level frame buffer

Components for 3D Graphics
Systems
Screen Refresh



Components for 3D Graphics
Systems
Colour Frame Buffer



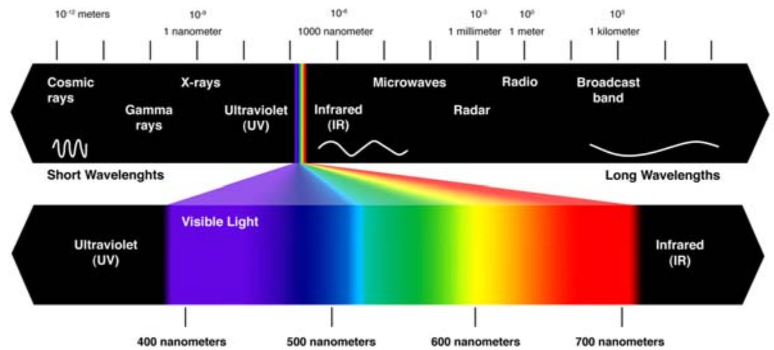
Components for 3D Graphics

Outline

- **Displaying an Image**
  - Hardware
  - Systems
  - **Colour Representation**
- 3D Scene Representation
  - Intro
  - Coordinate Systems
  - 3D Primitives

Components for 3D Graphics
Colour Representation

The Colour Spectrum

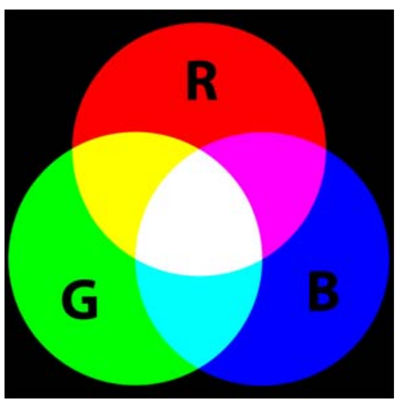


Components for 3D Graphics
Colour Representation

RGB – Red Green Blue

- **Based on Light**
- **Additive mixing**
- **Mainly for projecting**

R	G	B	Colour
0	0	0	Black
1	1	1	White
1	0	0	Red
0	1	0	Green
0	0	1	Blue
1	1	0	Yellow
0	1	1	Cyan
1	0	1	Magenta

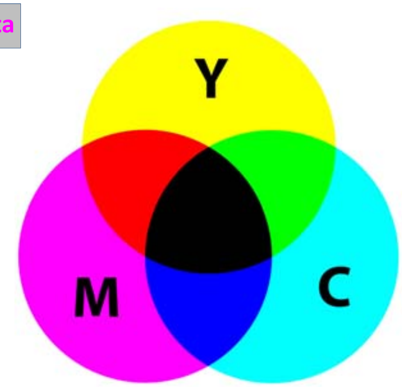


Components for 3D Graphics
Colour Representation

CYM – Cyan Yellow Magenta

- **Based on Pigment**
- **Subtractive mixing**
- **Mainly for printing**

C	M	Y	Colour
1	1	1	≈Black
0	0	0	White
0	1	1	Red
1	0	1	Green
1	1	0	Blue
0	0	1	Yellow
1	0	0	Cyan
0	1	0	Magenta



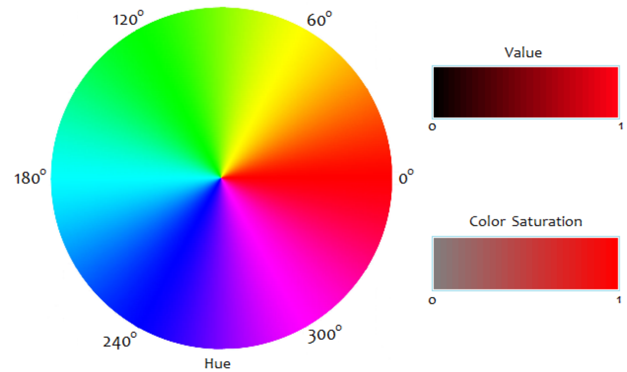
Components for 3D Graphics
Colour Representation

CYMK – Cyan Yellow Magenta Black

- K = “key”
- Based on Pigment
- Subtractive mixing
- Mainly used in printing
- Colours typically applied in the order CYMK

Components for 3D Graphics
Colour Representation

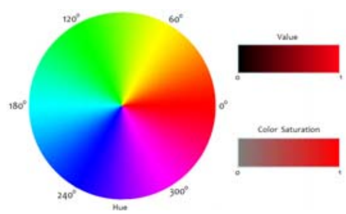
HSV – Hue Saturation Value



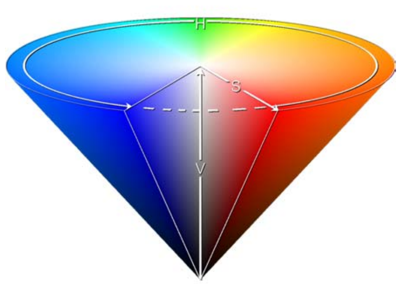
Components for 3D Graphics
Colour Representation

HSV – Hue Saturation Value

- Based on RGB
- Designed for computer graphics



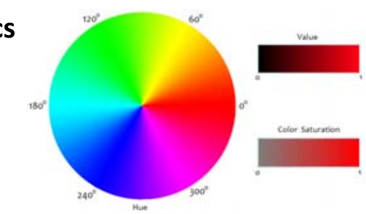
H	S	V	Colour
*	*	0	Black
*	0	1	White
0	1	1	Red
120	1	1	Green
240	1	1	Blue
*	0	0.5	Grey



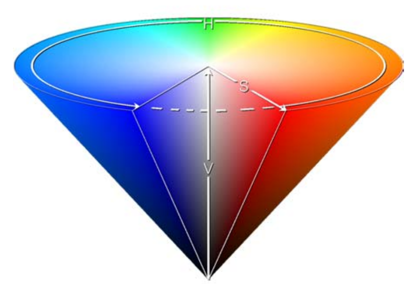
Components for 3D Graphics
Colour Representation

HSV – Hue Saturation Value

- Based on RGB
- Designed for computer graphics



H	S	V	Colour
*	*	0	Black
*	0	1	White
0	1	1	Red
120	1	1	Green
240	1	1	Blue
*	0	0.5	Grey



SE3VR11 : Virtual Reality

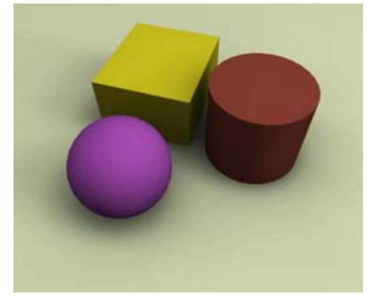
Professor Paul Sharkey
p.m.sharkey@reading.ac.uk

Lecture 3 – 28/01/2016

3D Scene Representation
Intro

Representation

- Points
- Lines or line segments
- Polygons
- Surfaces
- Solids
- Voxels
- . . .



We will look at some of these in more detail shortly …

Components for 3D Graphics

Outline

- **Displaying an Image**
  - Hardware
  - Systems
  - Colour Representation
- **3D Scene Representation**
  - Intro
  - Coordinate Systems
  - 3D Primitives

3D Scene Representation
Coordinate Systems

We need some sort of geometric base to form our computations, these come in the form of coordinate systems

- **Cartesian rectilinear system:**

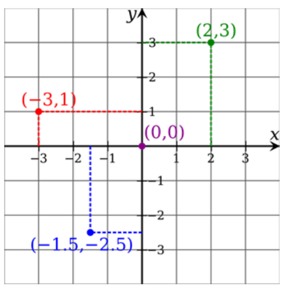
$$(x, y, z)$$

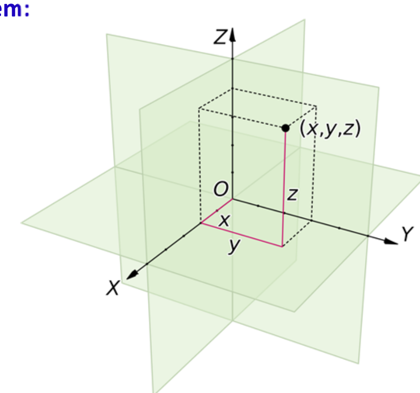
- **Spherical, Polar or Angular system:**

$$(r, \theta, \varphi)$$

3D Scene Representation
Coordinate Systems

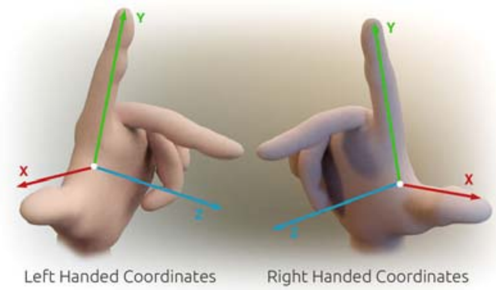
Cartesian rectilinear system:





3D Scene Representation
Coordinate Systems

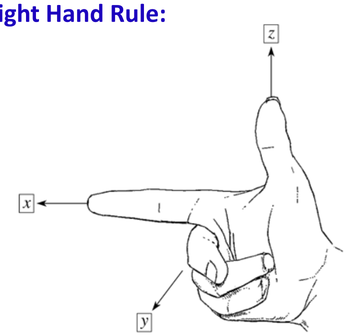
Left Hand Rule vs **Right Hand Rule**

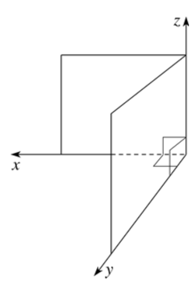


clockwise rotation vs **counter clockwise rotation**

3D Scene Representation
Coordinate Systems

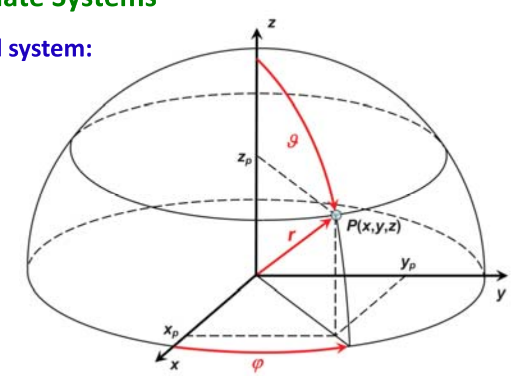
Right Hand Rule:





3D Scene Representation
Coordinate Systems

Spherical system:



Exercises: How to translate between the two systems?
How to translate between right/left hand systems?

Components for 3D Graphics

Outline

- **Displaying an Image**
  - Hardware
  - Systems
  - Colour Representation
- **3D Scene Representation**
  - Intro
  - Coordinate Systems
  - **3D Primitives**

3D Scene Representation
3D Primitives

Points

- **A point specifies a location**
- **Represented by 3 values**
- **The point has no volume (it is infinitely small)**
- **Can be expressed as**

$$\mathbf{p} = (x, y, z)$$

or

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

3D Scene Representation
3D Primitives

Vectors

- **Specifies a direction and a magnitude**
- **Represented by 3 values**
- **A point can be represented as a vector from (0, 0, 0)**
- **Commonly used with a unit length (normalised)**

$$\underline{\mathrm{V}} = \overline{\mathrm{V}} = (dx, dy, dz) = \langle dx, dy, dz \rangle$$

or

$$\underline{\mathrm{V}} = \overline{\mathrm{V}} = \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix}$$

3D Scene Representation
3D Primitives

Vector Addition

- **Adding**

$$\overline{\mathrm{C}} = \overline{\mathrm{A}} + \overline{\mathrm{B}} = (a_x + b_x, a_y + b_y, a_z + b_z)$$

- **Subtracting**

$$\overline{\mathrm{C}} = \overline{\mathrm{A}} - \overline{\mathrm{B}} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} - \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} a_x - b_x \\ a_y - b_y \\ a_z - b_z \end{bmatrix}$$

3D Scene Representation
3D Primitives

Vector Normalisation

- **Magnitude:** $\boxed{\|\overline{\mathrm{V}}\| = \sqrt{x^2 + y^2 + z^2}}$
- **When** $\|\overline{\mathrm{V}}\| = 1$ **the vector is normalised**
- **To normalise a vector:** $\boxed{\widehat{\mathrm{V}} = \dfrac{\overline{\mathrm{V}}}{\|\overline{\mathrm{V}}\|}}$
- $\widehat{\mathrm{V}}$ **is now of unit length**

3D Scene Representation
3D Primitives

Vector Dot Product (AKA **scalar** product)

- **With** $\overline{\mathbf{a}} = \langle a_1, a_2, a_3 \rangle$ **and** $\overline{\mathbf{b}} = \langle b_1, b_2, b_3 \rangle$
- **The dot product is given by**

$$\boxed{\overline{\mathbf{a}} \bullet \overline{\mathbf{b}} = a_1 b_1 + a_2 b_2 + a_3 b_3}$$

- **The result is a scalar value**

3D Scene Representation
3D Primitives

Angle between two Vectors

- **The dot product is also defined as**

$$\boxed{\overline{\mathbf{a}} \bullet \overline{\mathbf{b}} = \|\overline{\mathbf{a}}\| \cdot \|\overline{\mathbf{b}}\| \cos\theta}$$

- **Hence**

$$\boxed{\cos\theta = \frac{\overline{\mathbf{a}} \bullet \overline{\mathbf{b}}}{\|\overline{\mathbf{a}}\| \cdot \|\overline{\mathbf{b}}\|}}$$

- **The result** θ **can be found by taking the inverse** $\cos^{-1}(\ldots)$

3D Scene Representation
3D Primitives

Orthogonal Vectors

- **Two vectors are orthogonal (i.e. perpendicular) to each other when**

$$\boxed{\overline{\mathbf{a}} \bullet \overline{\mathbf{b}} = 0}$$

- **This is easily verified** $\cos\theta = \dfrac{\overline{\mathbf{a}} \bullet \overline{\mathbf{b}}}{\|\overline{\mathbf{a}}\| \cdot \|\overline{\mathbf{b}}\|} = 0$
- **Therefore** $\theta = \cos^{-1}\{0\} = 90°$

3D Scene Representation
3D Primitives

3D Line

- Given a point $\overline{\mathbf{p}}$ and a direction vector $\overline{\mathbf{V}}$
- Then $\overline{\mathbf{p}}_1$ is another point on the same line

$$\boxed{\overline{\mathbf{p}}_1 = \overline{\mathbf{p}} + t\overline{\mathbf{V}}}$$

where the parameter t is a scalar

$$(-\infty \le t \le \infty)$$

3D Scene Representation
3D Primitives

Rays

- Given a point $\overline{\mathbf{p}}$ and a direction vector $\overline{\mathbf{V}}$, and another point on the same line $\overline{\mathbf{p}}_1$

$$\boxed{\overline{\mathbf{p}}_1 = \overline{\mathbf{p}} + t\overline{\mathbf{V}}}$$

- Then, if the parameter t is restricted to the range

$$(0 \le t \le \infty)$$

- the point can be considered to a general point on a ray, originating at $\overline{\mathbf{p}}$ and shining in the direction $\overline{\mathbf{V}}$

3D Scene Representation
3D Primitives

3D Line Segment

- Given two points $\overline{\mathbf{p}}_1$ and $\overline{\mathbf{p}}_2$
- Then $\overline{\mathbf{p}}_3$ is another point on the same line between them

$$\boxed{\overline{\mathbf{p}}_3 = \overline{\mathbf{p}}_1 + t\left(\overline{\mathbf{p}}_2 - \overline{\mathbf{p}}_1\right)}$$

where $(0 \le t \le 1)$

Note that two points can be used to define a direction:

$$\boxed{\overline{\mathbf{p}}_2 - \overline{\mathbf{p}}_1 = \overline{\mathbf{V}}}$$

3D Scene Representation
3D Primitives

Matrices

2×2 matrix: $\mathbf{M} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$

3×3 matrix: $\mathbf{M} = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix}$

4×4 matrix: $\mathbf{M} = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}$

(note different styles in notation can be used)

3D Scene Representation
3D Primitives

2×2 Matrix Determinant

$$|\mathbf{M}| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

This will enable us to calculate the vector cross product between two vectors

3D Scene Representation
3D Primitives

Vector Cross Product (AKA **vector** product)

- **The cross product between two vectors is defined as**

$$\boxed{\bar{\mathbf{a}} \times \bar{\mathbf{b}} = \|\bar{\mathbf{a}}\| \cdot \|\bar{\mathbf{b}}\| \hat{\mathbf{n}} \sin\theta}$$

- **In words, it is …**
  - The magnitude of vector $\|\bar{\mathbf{a}}\|$
  - multiplied by the magnitude of vector $\|\bar{\mathbf{b}}\|$
  - multiplied by the **sine** of the angle between them $\sin\theta$
  - multiplied by the **normal** to both vectors $\hat{\mathbf{n}}$

Importantly, the cross product can be used to find the normal

3D Scene Representation
3D Primitives

Vector Cross Product $\bar{\mathbf{a}} \times \bar{\mathbf{b}} = \|\bar{\mathbf{a}}\| \cdot \|\bar{\mathbf{b}}\| \hat{\mathbf{n}} \sin\theta$

- **The cross product results in a vector**
- **The cross product can also be found using determinants as**

$$\bar{\mathbf{a}} \times \bar{\mathbf{b}} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad c_1 = \begin{vmatrix} a_2 & b_2 \\ a_3 & b_3 \end{vmatrix} \quad c_2 = -\begin{vmatrix} a_1 & b_1 \\ a_3 & b_3 \end{vmatrix} \quad c_3 = \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}$$

- **Note the negative sign in the middle term!**

- **Note that** $\boxed{\bar{\mathbf{a}} \times \bar{\mathbf{b}} = -\bar{\mathbf{b}} \times \bar{\mathbf{a}}}$

3D Scene Representation
3D Primitives

Vector Cross Product – Cover up method

- **One way to remember how to calculate cross products is using the cover up method**

$$\bar{\mathbf{a}} \times \bar{\mathbf{b}} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

- **To determine any element, cover up that row and calculate the resulting determinant (remembering to apply a negative in row 2)**

$$\bar{\mathbf{a}} \times \bar{\mathbf{b}} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

3D Scene Representation
3D Primitives

Vector Cross Product – Cover up method

- One way to remember how to calculate cross products is using the cover up method

$$\overline{\mathbf{a}} \times \overline{\mathbf{b}} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

- To determine any element, cover up that row and calculate the resulting determinant (remembering to apply a negative in row 2)

$$\overline{\mathbf{a}} \times \overline{\mathbf{b}} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} \blacksquare \\ a_2 \\ a_3 \end{bmatrix} \times \begin{bmatrix} \blacksquare \\ b_2 \\ b_3 \end{bmatrix} \quad \therefore \quad c_1 = \begin{vmatrix} a_2 & b_2 \\ a_3 & b_3 \end{vmatrix}$$

3D Scene Representation
3D Primitives

Normal of two vectors

The cross product can be used to find the normal to both vectors

A normal is found by taking the cross product

$$\mathbf{n} = \overline{\mathbf{a}} \times \overline{\mathbf{b}}$$

A unit direction normal is found by dividing by the magnitude of the resultant cross product

$$\boxed{\hat{\mathbf{n}} = \frac{\overline{\mathbf{a}} \times \overline{\mathbf{b}}}{\left\| \overline{\mathbf{a}} \times \overline{\mathbf{b}} \right\|}}$$

3D Scene Representation
3D Primitives

3D Plane

A plane can be defined in 2 ways:

- Any known point and where the normal vector to the plane is known
- Any three points that are not co-linear
  - if the three points were co-linear then the best they can be used for is to define an infinite number of planes about a common axis

3D Scene Representation
3D Primitives

3D Plane – Defined using 1 point and the normal

Given a point $\mathbf{p}_0$ and a normal vector $\hat{\mathbf{n}}$ then the general point $\mathbf{p} = (x, y, z)$ will be on the plane if the following constraint is true

$$\boxed{(\mathbf{p} - \mathbf{p}_0) \bullet \hat{\mathbf{n}} = 0}$$

Proof: If the dot product is equal to zero, then $\cos\theta = 0$ and the angle must be $90°$

Therefore the vector $(\mathbf{p} - \mathbf{p}_0)$ must lie in the plane and so the point $\mathbf{p}$ must be in the plane.

3D Scene Representation
3D Primitives

3D Plane – <u>Defined using 3 non co-linear points</u>

Given a points $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$ the general point $\mathbf{p} = (x, y, z)$ will be on the plane if the following constraint is true

$$\boxed{(\mathbf{p}-\mathbf{p}_0)\cdot\frac{(\mathbf{p}_2-\mathbf{p}_0)\times(\mathbf{p}_1-\mathbf{p}_0)}{\left\|(\mathbf{p}_2-\mathbf{p}_0)\times(\mathbf{p}_1-\mathbf{p}_0)\right\|}=0}$$

<u>Proof</u>: The quotient term is in fact the normal to the plane

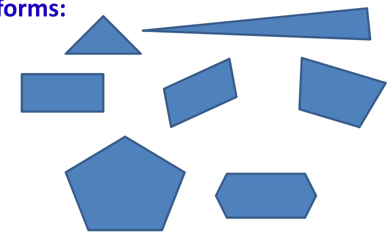
Each bracketed term is a vector in the plane and the cross product, which is then normalised, defines the <u>normal</u> $\hat{\mathbf{n}}$

(*The rest of the proof is the same as before*)

3D Scene Representation
3D Primitives

<u>Bounded Plane or Polygon</u>

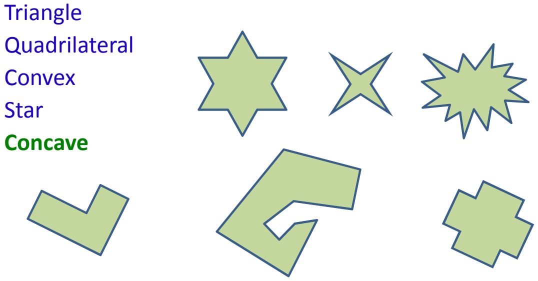
- **If a plane is bounded we get a polygon and they come in many forms:**
  - Triangle
  - Quadrilateral
  - **Convex**



3D Scene Representation
3D Primitives

<u>Bounded Plane or Polygon</u>

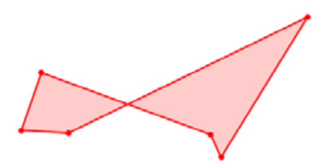
- **If a plane is bounded we get a polygon and they come in many forms:**
  - Triangle
  - Quadrilateral
  - Convex
  - Star
  - **Concave**



3D Scene Representation
3D Primitives

<u>Bounded Plane or Polygon</u>

- **If a plane is bounded we get a polygon and they come in many forms:**
  - Triangle
  - Quadrilateral
  - Convex
  - Star
  - Concave
  - **Self-intersecting**

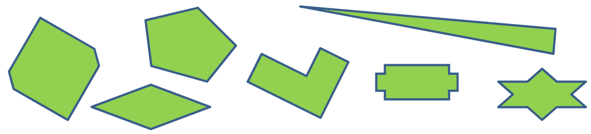


- **If a hole is required, then more than one polygon is required**

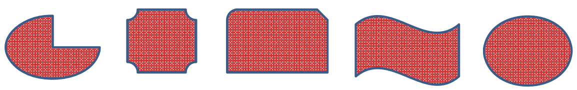
3D Scene Representation
3D Primitives

Polygons or not?

- Polygons



- Not Polygons



3D Scene Representation
3D Primitives

Matrix Multiplication

2×2 Example

$$\mathbf{A} = \begin{bmatrix} 3 & 5 \\ -1 & 2 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} -4 & 7 \\ 6 & 1 \end{bmatrix}$$

Then

$$\mathbf{AB} = \begin{bmatrix} 3 & 5 \\ -1 & 2 \end{bmatrix}\begin{bmatrix} -4 & 7 \\ 6 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} (3)(-4)+(5)(6) & (3)(7)+(5)(1) \\ (-1)(-4)+(2)(6) & (-1)(7)+(2)(1) \end{bmatrix}$$

$$= \begin{bmatrix} +18 & +26 \\ +16 & -5 \end{bmatrix}$$

Be sure you are familiar with the process

3D Scene Representation
3D Primitives

Matrix Multiplication

3×3 Example

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & -4 \\ 4 & -5 & 6 \\ 8 & -7 & 9 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 10 & -3 & -4 \\ 2 & 0 & 1 \\ 12 & 12 & 20 \end{bmatrix}$$

Show that $$\mathbf{C} = \mathbf{AB} = \begin{bmatrix} -22 & -54 & -85 \\ 102 & 60 & 99 \\ 174 & 84 & 141 \end{bmatrix}$$

3D Scene Representation
3D Primitives

Matrix Multiplication

4×4 Example

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 8 & 7 & 6 \\ 5 & 4 & 3 & 2 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 1 & -2 & 3 & -4 \\ -5 & 6 & 7 & -8 \\ 10 & 12 & 14 & 16 \\ 0 & 10 & 20 & 18 \end{bmatrix}$$

Show that $$\mathbf{C} = \mathbf{AB} = \begin{bmatrix} 21 & 86 & 139 & 100 \\ 45 & 190 & 315 & 188 \\ 39 & 174 & 301 & 120 \\ 15 & 70 & 125 & 32 \end{bmatrix}$$

SE3VR11 : Virtual Reality

Professor Paul Sharkey p.m.sharkey@reading.ac.uk

Lecture 4 – 04/02/2016

3D Scene Representation
3D Primitives

Matrix Transforms

All transformations appropriate for computer graphics can be represented with a single 4×4 matrix

$$\mathbf{M} = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}$$

If a transformation is represented by the matrix $\mathbf{T}$, the point $\mathbf{p}$ can be transformed to the new point $\mathbf{p}'$ by matrix multiplication:

$$\boxed{\mathbf{p}' = \mathbf{T}\mathbf{p}}$$

3D Scene Representation
3D Primitives

Matrix Transforms

Example: If $\mathbf{T}$ and $\mathbf{p}$ are given by $\mathbf{T} = \begin{bmatrix} 1 & 3 & -2 & 5 \\ 4 & 8 & 4 & 4 \\ -1 & 0 & 2 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ $\mathbf{p} = \begin{bmatrix} 1 \\ 5 \\ 2 \end{bmatrix}$

Find $\mathbf{p}'$

$$\mathbf{p}' = \mathbf{T} \times \mathbf{p} = \begin{bmatrix} 1 & 3 & -2 & 5 \\ 4 & 8 & 4 & 4 \\ -1 & 0 & 2 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 5 \\ 2 \end{bmatrix} \quad \text{X}$$

But, we have a problem …

The number of columns in the matrix DOES NOT equal the number of rows in the vector. The dimensions are not right!

3D Scene Representation
3D Primitives

Matrix Transforms

The solution is to augment the point into an homogenous form

$$\mathbf{p} = \begin{bmatrix} 1 \\ 5 \\ 2 \end{bmatrix} \quad \Rightarrow \quad \mathbf{p}_h = \begin{bmatrix} 1 \\ 5 \\ 2 \\ 1 \end{bmatrix}$$

Now

$$\mathbf{p}'_h = \mathbf{T}\mathbf{p}_h = \begin{bmatrix} 1 & 3 & -2 & 5 \\ 4 & 8 & 4 & 4 \\ -1 & 0 & 2 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 5 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 17 \\ 56 \\ 4 \\ 1 \end{bmatrix} \quad \Rightarrow \quad \mathbf{p}' = \begin{bmatrix} 17 \\ 56 \\ 4 \end{bmatrix}$$

where the transformed point is found by conversion back from the homogenous form

3D Scene Representation
3D Primitives

$$\mathbf{p}'_h = \mathbf{T}\mathbf{p}_h = \begin{bmatrix} 1 & 3 & -2 & 5 \\ 4 & 8 & 4 & 4 \\ -1 & 0 & 2 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 5 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 17 \\ 56 \\ 4 \\ 1 \end{bmatrix}$$

Matrix Transforms

How do such transformation matrices come about?

Transformation matrices can be used to define:

- Translations
- Rotations
- Scaling
- Skewing *or* Shearing
- Reflections
- Perspective transforms
- Any combination of the above

Combinations are realised by multiplication of matrices

3D Scene Representation
3D Primitives

Matrix Transforms – Translations

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Applied as (eg.)

$$\mathbf{p} = \begin{bmatrix} 1 \\ 5 \\ 2 \end{bmatrix} \Rightarrow \mathbf{p}'_{(h)} = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 5 \\ 2 \\ 1 \end{bmatrix} \Rightarrow \mathbf{p}' = \begin{bmatrix} 1+dx \\ 5+dy \\ 2+dz \end{bmatrix}$$

3D Scene Representation
3D Primitives

Matrix Transforms – Rotations about x

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Applied as (eg.)

$$\mathbf{p} = \begin{bmatrix} 1 \\ 5 \\ 2 \end{bmatrix} \Rightarrow \mathbf{p}'_{(h)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 60 & -\sin 60 & 0 \\ 0 & \sin 60 & \cos 60 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 5 \\ 2 \\ 1 \end{bmatrix} \Rightarrow \mathbf{p}' = \begin{bmatrix} 1 \\ 0.7680 \\ 5.3301 \end{bmatrix}$$

3D Scene Representation
3D Primitives

Matrix Transforms – Rotations about y

$$\mathbf{T} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Applied as (eg.)

$$\mathbf{p} = \begin{bmatrix} 1 \\ 5 \\ 2 \end{bmatrix} \Rightarrow \mathbf{p}'_{(h)} = \begin{bmatrix} \cos 60 & 0 & \sin 60 & 0 \\ 0 & 1 & 0 & 0 \\ -\sin 60 & 0 & \cos 60 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 5 \\ 2 \\ 1 \end{bmatrix} \Rightarrow \mathbf{p}' = \begin{bmatrix} 2.2321 \\ 5 \\ 0.1340 \end{bmatrix}$$

3D Scene Representation
3D Primitives

Matrix Transforms – Rotations about z

$$\mathbf{T}=\begin{bmatrix}\cos\theta & -\sin\theta & 0 & 0\\ \sin\theta & \cos\theta & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Applied as (eg.)

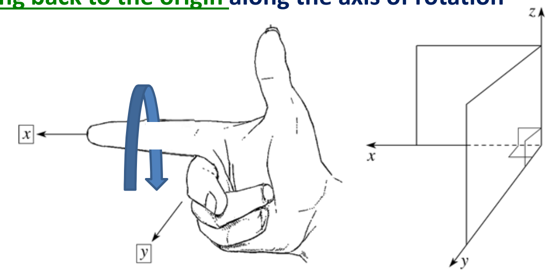
$$\mathbf{p}=\begin{bmatrix}1\\5\\2\end{bmatrix}\Rightarrow \mathbf{p}'_{(h)}=\begin{bmatrix}\cos 60 & -\sin 60 & 0 & 0\\ \sin 60 & \cos 60 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1\\5\\2\\1\end{bmatrix}\Rightarrow \mathbf{p}'=\begin{bmatrix}-3.801\\3.3660\\2\end{bmatrix}$$

3D Scene Representation
3D Primitives

Matrix Transforms – Rotations general

Rotations are always about the main coordinate axes

Rotations are positive in an anti-clockwise direction when looking back to the origin along the axis of rotation



3D Scene Representation
3D Primitives

Matrix Transforms – Scaling

$$\mathbf{T}=\begin{bmatrix}S_x & 0 & 0 & 0\\ 0 & S_y & 0 & 0\\ 0 & 0 & S_z & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Applied as (eg.)

$$\mathbf{p}=\begin{bmatrix}1\\5\\2\end{bmatrix}\Rightarrow \mathbf{p}'_{(h)}=\begin{bmatrix}S_x & 0 & 0 & 0\\ 0 & S_y & 0 & 0\\ 0 & 0 & S_z & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1\\5\\2\\1\end{bmatrix}\Rightarrow \mathbf{p}'=\begin{bmatrix}S_x\\5S_y\\2S_z\end{bmatrix}$$

3D Scene Representation
3D Primitives

Matrix Transforms – Shear

$$\mathbf{T}=\begin{bmatrix}1 & sh_x^y & sh_x^z & 0\\ sh_y^x & 1 & sh_y^z & 0\\ sh_z^x & sh_z^y & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

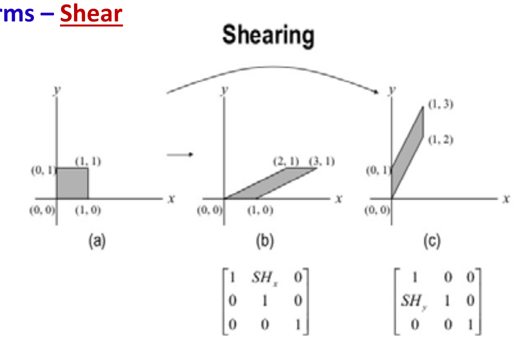
Applied as (eg.)

$$\mathbf{p}=\begin{bmatrix}1\\5\\2\end{bmatrix}\Rightarrow \mathbf{p}'_{(h)}=\begin{bmatrix}1 & sh_x^y & sh_x^z & 0\\ sh_y^x & 1 & sh_y^z & 0\\ sh_z^x & sh_z^y & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1\\5\\2\\1\end{bmatrix}\Rightarrow \mathbf{p}'=\begin{bmatrix}1+5sh_x^y+2sh_x^z\\5+sh_y^x+2sh_y^z\\2+sh_z^x+5sh_z^y\end{bmatrix}$$

3D Scene Representation
3D Primitives

Matrix Transforms – Shear

Examples



Here $sh_x^y = 2$

and $sh_y^x = 2$

The matrices shown here are the equivalent of the top left 3×3 block

3D Scene Representation
3D Primitives

Matrix Transforms – Shear

Exercise: Show that applying the following shear matrix

$$\mathbf{T} = \begin{bmatrix} 1 & sh_x^y & 0 & 0 \\ sh_y^x & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

results in a very different outcome to applying one shear after another

$$\mathbf{T} = \mathbf{T}_1\mathbf{T}_2 = \begin{bmatrix} 1 & sh_x^y & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ sh_y^x & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Scene Representation
3D Primitives

Matrix Transforms – Shear

$$\mathbf{T} = \begin{bmatrix} 1 & sh_x^y & 0 & 0 \\ sh_y^x & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Mathematically:

$$\mathbf{T} = \mathbf{T}_1\mathbf{T}_2 = \begin{bmatrix} 1 & sh_x^y & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ sh_y^x & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1+\left(sh_x^y\right)\left(sh_y^x\right) & sh_x^y & 0 & 0 \\ sh_y^x & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

What is the graphical result for the cube?

3D Scene Representation
3D Primitives

Matrix Transforms – Shear

$$\mathbf{T} = \begin{bmatrix} 1+\left(sh_x^y\right)\left(sh_y^x\right) & sh_x^y & 0 & 0 \\ sh_y^x & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

What is the graphical result for the cube?

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

Show that the points are located at:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 5 \\ 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 7 \\ 3 \\ 0 \end{bmatrix}$$

3D Scene Representation
3D Primitives

Matrix Transforms – Shear

$$\mathbf{T} = \begin{bmatrix} 1 & sh_x^y & 0 & 0 \\ sh_y^x & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

What is the graphical result for the cube?

$$\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}$$

Show that the points are located at:

$$\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}2\\1\\0\end{bmatrix}, \begin{bmatrix}1\\2\\0\end{bmatrix}, \begin{bmatrix}3\\3\\0\end{bmatrix}$$

3D Scene Representation
3D Primitives

Matrix Transforms – Reflections in x-axis

$$\mathbf{T} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Applied as (eg.)

$$\mathbf{p} = \begin{bmatrix}1\\5\\2\end{bmatrix} \Rightarrow \mathbf{p}'_{(h)} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix}1\\5\\2\\1\end{bmatrix} \Rightarrow \mathbf{p}' = \begin{bmatrix}-1\\5\\2\end{bmatrix}$$

3D Scene Representation
3D Primitives

Matrix Transforms – Reflections in y-axis

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Applied as (eg.)

$$\mathbf{p} = \begin{bmatrix}1\\5\\2\end{bmatrix} \Rightarrow \mathbf{p}'_{(h)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix}1\\5\\2\\1\end{bmatrix} \Rightarrow \mathbf{p}' = \begin{bmatrix}1\\-5\\2\end{bmatrix}$$

3D Scene Representation
3D Primitives

Matrix Transforms – Reflections in z-axis

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Applied as (eg.)

$$\mathbf{p} = \begin{bmatrix}1\\5\\2\end{bmatrix} \Rightarrow \mathbf{p}'_{(h)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix}1\\5\\2\\1\end{bmatrix} \Rightarrow \mathbf{p}' = \begin{bmatrix}1\\5\\-2\end{bmatrix}$$

3D Scene Representation
3D Primitives

Matrix Transforms – Combinations

For more complex transformations it is possible to combine multiple transformation matrices in a single matrix

Example: Move point $\mathbf{p}$ 10 units along the y-axis, then rotate it 20 degrees around the z-axis and then move it –15 units along the x-axis.

3D Scene Representation
3D Primitives

Matrix Transforms – Combinations

Example: Move point $\mathbf{p}$ 10 units along the y-axis, rotate it 20 degrees around the z-axis and then move it –15 units along the x-axis

$$\mathbf{T}_{trans,y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 10 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{T}_{rot,z} = \begin{bmatrix} \cos 20 & -\sin 20 & 0 & 0 \\ \sin 20 & \cos 20 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{T}_{trans,x} = \begin{bmatrix} 1 & 0 & 0 & -15 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.4081 & -0.9129 & 0 & 0 \\ 0.9129 & 0.4081 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Scene Representation
3D Primitives

Matrix Transforms – Combinations

Example: Move point $\mathbf{p}$ 10 units along the y-axis, rotate it 20 degrees around the z-axis and then move it –15 units along the x-axis …

$$\mathbf{p}' = \mathbf{T}_{trans,y}\mathbf{p} \qquad (\mathbf{p}' = \mathbf{T}_1\mathbf{p})$$

$$\mathbf{p}'' = \mathbf{T}_{rot,z}\mathbf{p}' \qquad (\mathbf{p}'' = \mathbf{T}_2\mathbf{p}')$$

$$\mathbf{p}''' = \mathbf{T}_{trans,x}\mathbf{p}'' \qquad (\mathbf{p}''' = \mathbf{T}_3\mathbf{p}'')$$

This is combined as

$$\boxed{\mathbf{p}''' = \mathbf{T}_{trans,x}\mathbf{T}_{rot,z}\mathbf{T}_{trans,y}\mathbf{p}} \qquad (\mathbf{p}''' = \mathbf{T}_3\mathbf{T}_2\mathbf{T}_1\mathbf{p})$$

Note the order of the transforms is from right to left

This is most important

3D Scene Representation
3D Primitives

Matrix Transforms – Combinations

The order of matrix multiplication is important!

$$\boxed{\mathbf{T} \times \mathbf{S} \neq \mathbf{S} \times \mathbf{T}}$$

3D Scene Representation
3D Primitives

Matrix Transforms – Combinations

$$\mathbf{p}''' = \mathbf{T}_{trans,x}\mathbf{T}_{rot,z}\mathbf{T}_{trans,y}\mathbf{p}$$

Example:

$$\mathbf{p}''' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 10 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.4081 & -0.9129 & 0 & 0 \\ 0.9129 & 0.4081 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -15 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{p}$$

$$\mathbf{p}''' = \begin{bmatrix} 0.4081 & -0.9129 & 0 & -6.1215 \\ 0.9129 & 0.4081 & 0 & -3.6935 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{p} \qquad \mathbf{p} = \begin{bmatrix} 1 \\ 5 \\ 2 \end{bmatrix} \Rightarrow \mathbf{p}' = \begin{bmatrix} -10.2779 \\ -0.7401 \\ 2 \end{bmatrix}$$

(the z component is unaffected in this example)

3D Scene Representation
3D Primitives

Transforming Objects

In the previous example, the transformations were applied to a single point

How is the same transform applied to a complex object?

Consider a cuboid, defined by eight coherent points

$$\mathbf{obj}_{cuboid} = \begin{bmatrix} 2 & 8 & 8 & 2 & 2 & 8 & 8 & 2 \\ 3 & 3 & 3 & 3 & 5 & 5 & 5 & 5 \\ 4 & 4 & -3 & -3 & 4 & 4 & -3 & -3 \end{bmatrix}$$

(This cuboid is aligned with the major axes – sketch it out!)

3D Scene Representation
3D Primitives

Transforming Objects

Applying the combined transform requires augmented the cuboid description to include a fourth row of '1's

$$\mathbf{obj}_{cuboid(h)} = \begin{bmatrix} 2 & 8 & 8 & 2 & 2 & 8 & 8 & 2 \\ 3 & 3 & 3 & 3 & 5 & 5 & 5 & 5 \\ 4 & 4 & -3 & -3 & 4 & 4 & -3 & -3 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The transformed object is

$$\mathbf{obj}'_{cuboid(h)} = \begin{bmatrix} 0.4081 & -0.9129 & 0 & -6.1215 \\ 0.9129 & 0.4081 & 0 & -3.6935 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 8 & 8 & 2 & 2 & 8 & 8 & 2 \\ 3 & 3 & 3 & 3 & 5 & 5 & 5 & 5 \\ 4 & 4 & -3 & -3 & 4 & 4 & -3 & -3 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

3D Scene Representation
3D Primitives

Transforming Objects

Result:

$$\mathbf{obj}'_{cuboid} = \begin{bmatrix} -8.044 & -5.595 & -5.595 & -8.044 & -9.870 & -7.421 & -7.421 & -9.870 \\ -0.643 & 4.834 & 4.834 & -0.643 & 0.173 & 5.650 & 5.650 & 0.173 \\ 4 & 4 & -3 & -3 & 4 & 4 & -3 & -3 \end{bmatrix}$$

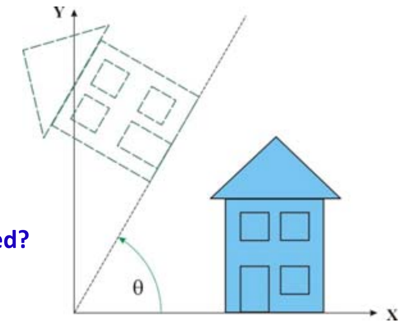
- The ordering of the points is the same as originally defined
- It is clear from the points that there is still a degree of symmetry with the main coordinate axes
- This is expected as the object was only rotated by one primary rotation
- More complex shapes may have hundreds or thousands of vertices

3D Scene Representation
3D Primitives

Rotating Objects – rotate the following object about z by 60°

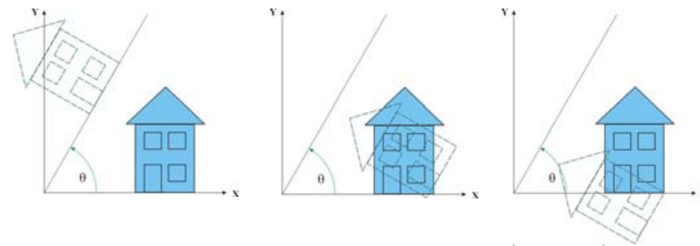
This is the result of applying the $Rot_{z,60}$ transform.

Was this what was wanted?

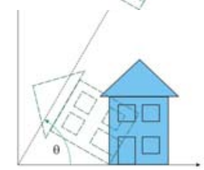


3D Scene Representation
3D Primitives

Rotating Objects – rotate the following object about z by 45°



Which is the desired rotation?



3D Scene Representation
3D Primitives

Rotating Objects

Rotations need to be defined not only by the axis of rotation but also by the point on the object that the rotation axis goes through.

If a point is identified in the object through which the axis of desired rotation is defined, the rotation can be achieved by

- translating this point to the origin of the coordinate system
- affecting the rotation
- translating back to its original location

$$\boxed{\mathbf{obj}_{rotated} = \mathbf{T}_{Tx,back}\mathbf{T}_{Rot,\theta}\mathbf{T}_{Tx,to\ origin}\mathbf{obj}_{original}}$$

3D Scene Representation
3D Primitives

Rotating Objects

- translating this point to the origin of the coordinate system
- affecting the rotation
- translating back to its original location

$$\boxed{\mathbf{obj}_{rotated} = \mathbf{T}_{Tx,back}\mathbf{T}_{Rot,\theta}\mathbf{T}_{Tx,to\ origin}\mathbf{obj}_{original}}$$

Note the order of transformation

3D Scene Representation
3D Primitives

What about Arbitrary Rotations?

A more general rotation can be achieved by defining

- A point $\mathbf{p}_1$ relative to the object (or within the object)
- A normalised direction vector, $\mathbf{k}$, defined by a second point $\mathbf{p}_2$ relative to point $\mathbf{p}_1$ and
- The angle θ about this axis to rotate

The final rotation is

$$\boxed{\mathbf{obj}_{rotated} = \mathbf{T}_{Tx,+\mathbf{p}_1}\mathbf{T}_{Rot,\mathbf{k},\theta}\mathbf{T}_{Tx,-\mathbf{p}_1}\mathbf{obj}_{original}}$$

where the rotation is defined on the following slides

3D Scene Representation
3D Primitives

$$\boxed{\mathbf{T}_{Rot,\mathbf{k},\theta}}$$

Equivalent Axis Rotations

An equivalent axes rotation is achieved by aligning the equivalent axis with one of the principal coordinate axes (e.g. the z axis), rotating about that axis, and then putting everything back in its original orientation

(Similar to translating to the origin and back again)

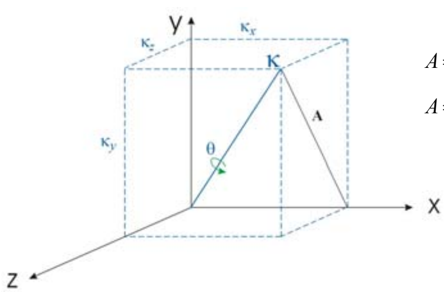
This initial alignment requires two rotations and hence two reverse rotations plus the required rotation in between

This gives FIVE rotations in total

3D Scene Representation
3D Primitives

$$\boxed{\mathbf{T}_{Rot,\mathbf{k},\theta}}$$

Equivalent Axis Rotations

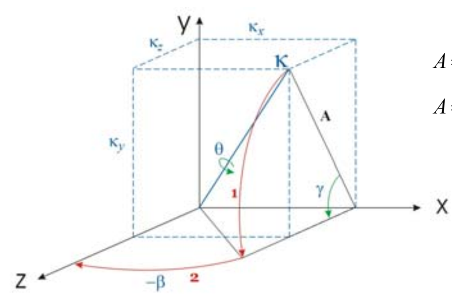


$$A = \sqrt{\kappa_y^2 + \kappa_z^2}$$
$$A = \sqrt{1 - \kappa_x^2}$$

3D Scene Representation
3D Primitives

$$\boxed{\mathbf{T}_{Rot,\mathbf{k},\theta}}$$

Equivalent Axis Rotations

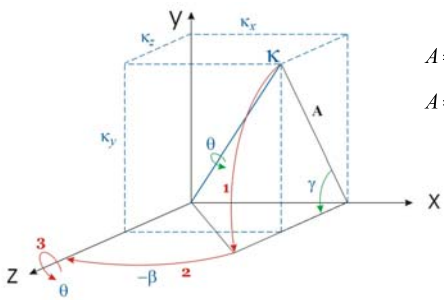


$$A = \sqrt{\kappa_y^2 + \kappa_z^2}$$
$$A = \sqrt{1 - \kappa_x^2}$$

3D Scene Representation
3D Primitives

$\mathbf{T}_{Rot,\mathbf{k},\theta}$

Equivalent Axis Rotations

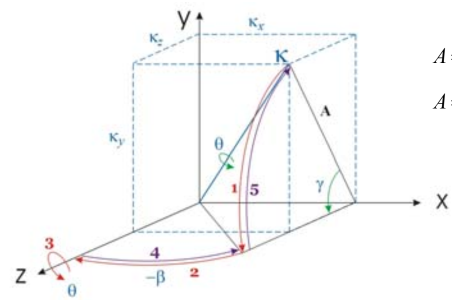


$$A=\sqrt{\kappa_y^2+\kappa_z^2}$$
$$A=\sqrt{1-\kappa_x^2}$$

3D Scene Representation
3D Primitives

$\mathbf{T}_{Rot,\mathbf{k},\theta}$

Equivalent Axis Rotations



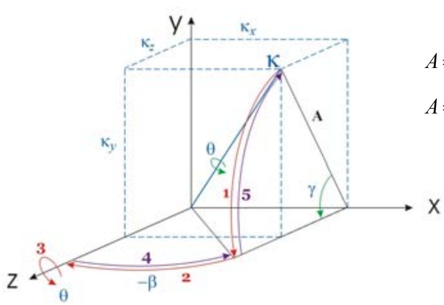
$$A=\sqrt{\kappa_y^2+\kappa_z^2}$$
$$A=\sqrt{1-\kappa_x^2}$$

$$\mathbf{T}_{Rot,\mathbf{k},\theta}=\mathbf{T}_5\mathbf{T}_4\mathbf{T}_3\mathbf{T}_2\mathbf{T}_1$$

3D Scene Representation
3D Primitives

$\mathbf{T}_{Rot,\mathbf{k},\theta}$

Equivalent Axis Rotations



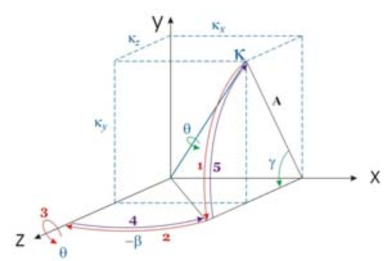
$$A=\sqrt{\kappa_y^2+\kappa_z^2}$$
$$A=\sqrt{1-\kappa_x^2}$$

$$\mathbf{T}_{Rot,\mathbf{k},\theta}=\mathbf{T}_{Rot,\mathbf{x},+\gamma}\mathbf{T}_{Rot,\mathbf{y},-\beta}\mathbf{T}_{Rot,\mathbf{z},\theta}\mathbf{T}_{Rot,\mathbf{y},+\beta}\mathbf{T}_{Rot,\mathbf{x},-\gamma}$$

3D Scene Representation
3D Primitives

$\mathbf{T}_{Rot,\mathbf{k},\theta}$

Equivalent Axis Rotations



$$A=\sqrt{\kappa_y^2+\kappa_z^2}$$
$$A=\sqrt{1-\kappa_x^2}$$

$$\boxed{\sin\beta=\kappa_x}$$
$$\boxed{\cos\beta=A}$$

$$A\cos\gamma=\kappa_z \qquad \boxed{\cos\gamma=\frac{\kappa_z}{A}}$$

$$A\sin\gamma=\kappa_y \qquad \boxed{\sin\gamma=\frac{\kappa_y}{A}}$$

$$\mathbf{T}_{Rot,\mathbf{k},\theta}=\mathbf{T}_{Rot,\mathbf{x},+\gamma}\mathbf{T}_{Rot,\mathbf{y},-\beta}\mathbf{T}_{Rot,\mathbf{z},\theta}\mathbf{T}_{Rot,\mathbf{y},+\beta}\mathbf{T}_{Rot,\mathbf{x},-\gamma}$$

3D Scene Representation
3D Primitives

$\mathbf{T}_{Rot,\mathbf{k},\theta}$

Equivalent Axis Rotations

An equivalent axes rotation transform is then found as

lots

and lots

and lots of algebra later …

3D Scene Representation
3D Primitives

$\mathbf{T}_{Rot,\mathbf{k},\theta}$

Equivalent Axis Rotations

An equivalent axes rotation transform is then found as

$$\mathbf{T}_{Rot,\mathbf{k},\theta} = \begin{bmatrix} xxv+c & xyv-zs & xzv+ys & 0 \\ yxv+zs & yyv+c & yzv-xs & 0 \\ zxv-ys & zyv+xs & zzv+c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where shorthand is used, with

$$s = \sin(\theta) \qquad c = \cos(\theta) \qquad v = 1 - c$$

and

$$\mathbf{k} = [x, y, z] \qquad \|\mathbf{k}\| = 1$$

SE3VR11 : Virtual Reality

Professor Paul Sharkey p.m.sharkey@reading.ac.uk

Lecture 5 – 25/02/2016

3D Scene Representation
3D Primitives

$\mathbf{T}_{Rot,\mathbf{k},\theta}$

Equivalent Axis Rotations

An equivalent axes rotation is achieved by aligning the equivalent axis with one of the principal coordinate axes (e.g. the z axis), rotating about that axis, and then putting everything back in its original orientation

(Similar to translating to the origin and back again)

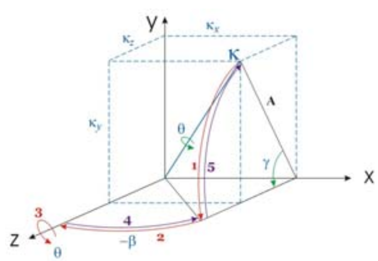
This initial alignment requires two rotations and hence two reverse rotations plus the required rotation in between

This gives FIVE rotations in total

3D Scene Representation
3D Primitives

$\mathbf{T}_{Rot,\mathbf{k},\theta}$

Equivalent Axis Rotations



$$A=\sqrt{\kappa_y^2+\kappa_z^2}$$
$$A=\sqrt{1-\kappa_x^2}$$

$$\boxed{\sin\beta=\kappa_x}\quad\boxed{\cos\beta=A}$$

$$A\cos\gamma=\kappa_z \qquad \boxed{\cos\gamma=\frac{\kappa_z}{A}}$$

$$A\sin\gamma=\kappa_y \qquad \boxed{\sin\gamma=\frac{\kappa_y}{A}}$$

$$\boxed{\mathbf{T}_{Rot,\mathbf{k},\theta}=\mathbf{T}_{Rot,\mathbf{x},+\gamma}\mathbf{T}_{Rot,\mathbf{y},-\beta}\mathbf{T}_{Rot,\mathbf{z},\theta}\mathbf{T}_{Rot,\mathbf{y},+\beta}\mathbf{T}_{Rot,\mathbf{x},-\gamma}}$$

3D Scene Representation
3D Primitives

$\mathbf{T}_{Rot,\mathbf{k},\theta}$

Equivalent Axis Rotations

An equivalent axes rotation transform is then found as

$$\mathbf{T}_{Rot,\mathbf{k},\theta}=\begin{bmatrix} xxv+c & xyv-zs & xzv+ys & 0 \\ yxv+zs & yyv+c & yzv-xs & 0 \\ zxv-ys & zyv+xs & zzv+c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where shorthand is used, with

$$s=\sin(\theta) \qquad c=\cos(\theta) \qquad v=1-c$$

and

$$\mathbf{k}=[x,y,z] \qquad \|\mathbf{k}\|=1$$

3D Scene Representation
3D Primitives

$$\mathbf{T}_{Rot,\mathbf{k},\theta} = \begin{bmatrix} xxv+c & xyv-zs & xzv+ys & 0 \\ yxv+zs & yyv+c & yzv-xs & 0 \\ zxv-ys & zyv+xs & zzv+c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Final Transform Set – RECAP

The final Transform is found as:

1. Translate to origin
2. Rotate about $\mathbf{k}$ by θ
3. Translate back to original position

$$\boxed{\mathbf{obj}_{rotated} = \mathbf{T}_{Tx,back}\mathbf{T}_{Rot,\mathbf{k},\theta}\mathbf{T}_{Tx,to\ origin}\mathbf{obj}_{original}}$$

3D Scene Representation
3D Primitives – Example

Example

The origin of an object is defined at a position [10,20,30].

It is desired to re-orient the object by rotating it by 150° about an axis defined by the vector pointing from the object origin to another point on the object at [25, 10, 10].

Find the transform that implements the above.

$$\mathbf{T}_1 = \mathbf{T}_{Tx,back}\mathbf{T}_{Rot,\mathbf{k},\theta}\mathbf{T}_{Tx,to\ origin}$$

3D Scene Representation
3D Primitives – Example

$$\mathbf{T}_1 = \mathbf{T}_{Tx,back}\mathbf{T}_{Rot,\mathbf{k},\theta}\mathbf{T}_{Tx,to\ origin}$$

Example

$$\mathbf{T}_{Tx,to\ origin} = \begin{bmatrix} 1 & 0 & 0 & -10 \\ 0 & 1 & 0 & -20 \\ 0 & 0 & 1 & -30 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{T}_{Tx,back} = \begin{bmatrix} 1 & 0 & 0 & +10 \\ 0 & 1 & 0 & +20 \\ 0 & 0 & 1 & +30 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The rotation matrix is

$$\mathbf{T}_{Rot,\mathbf{k},150} = \begin{bmatrix} xxv+c & xyv-zs & xzv+ys & 0 \\ yxv+zs & yyv+c & yzv-xs & 0 \\ zxv-ys & zyv+xs & zzv+c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Scene Representation
3D Primitives – Example

$$\mathbf{T}_1 = \mathbf{T}_{Tx,back}\mathbf{T}_{Rot,\mathbf{k},\theta}\mathbf{T}_{Tx,to\ origin}$$

Example

The direction vector is found as

$$\mathbf{V}_{01} = \begin{bmatrix} 25 \\ 10 \\ 10 \end{bmatrix} - \begin{bmatrix} 10 \\ 20 \\ 30 \end{bmatrix} = \begin{bmatrix} 15 \\ -10 \\ -20 \end{bmatrix} \qquad \mathbf{k} = \begin{bmatrix} 15 \\ -10 \\ -20 \end{bmatrix} \Big/ \sqrt{15^2+10^2+20^2} = \begin{bmatrix} 0.5571 \\ -0.3714 \\ -0.7428 \end{bmatrix}$$

Check that the direction vector is of unit length

$$|\mathbf{k}| = \sqrt{(0.5571)^2 + (-0.3714)^2 + (-0.7428)^2} = 1.000025...$$

OK to the 4DP retained in this illustration

Programmes should maintain much higher accuracy than 4DP

3D Scene Representation
3D Primitives – Example

$$\mathbf{k} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0.5571 \\ -0.3714 \\ -0.7428 \end{bmatrix}$$

Example

Equivalent axis rotation, $\theta = 150°$

$s = \sin(\theta) = 0.5 \qquad c = \cos(\theta) = -0.866 \qquad v = 1 - c = 0.1340$

$$\mathbf{T}_{Rot,\mathbf{k},150} = \begin{bmatrix} -0.2869 & -0.0147 & -0.9578 & 0 \\ -0.7575 & -0.6086 & 0.2362 & 0 \\ -0.5865 & 0.7933 & 0.1635 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Check that resulting matrix is a true rotation

$$\mathbf{R} \times \mathbf{R}^T = \mathbf{I}_3$$

(The inverse of a rotation matrix is the transpose of the 3×3 rotation matrix). This is true to ~4DP here

3D Scene Representation
3D Primitives – Example

$$\mathbf{T}_1 = \mathbf{T}_{Tx,back}\mathbf{T}_{Rot,\mathbf{k},\theta}\mathbf{T}_{Tx,to\ origin}$$

Example

The final transform is

$$\mathbf{T}_1 = \begin{bmatrix} 1 & 0 & 0 & +10 \\ 0 & 1 & 0 & +20 \\ 0 & 0 & 1 & +30 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -0.2869 & -0.0147 & -0.9578 & 0 \\ -0.7575 & -0.6086 & 0.2362 & 0 \\ -0.5865 & 0.7933 & 0.1635 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -10 \\ 0 & 1 & 0 & -20 \\ 0 & 0 & 1 & -30 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -0.2869 & -0.0147 & -0.9578 & 41.8981 \\ -0.7575 & -0.6086 & 0.2362 & 32.6608 \\ -0.5865 & 0.7933 & 0.1635 & 15.0932 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Scene Representation
3D Primitives – Example

$$\mathbf{T}_1 = \mathbf{T}_{Tx,back}\mathbf{T}_{Rot,\mathbf{k},\theta}\mathbf{T}_{Tx,to\ origin}$$

Example

The final transform is

$$= \begin{bmatrix} -0.2869 & -0.0147 & -0.9578 & 41.8981 \\ -0.7575 & -0.6086 & 0.2362 & 32.6608 \\ -0.5865 & 0.7933 & 0.1635 & 15.0932 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The top left **3×3 matrix** represents the compound rotation of the whole object

The top right **1×3 matrix** represents the additional translation of all points on the object

3D Scene Representation
3D Primitives – Example

$$\mathbf{T}_1 = \mathbf{T}_{Tx,back}\mathbf{T}_{Rot,\mathbf{k},\theta}\mathbf{T}_{Tx,to\ origin}$$

Example

Note that the two points that define the axis of rotation are

- The origin at a position [10,20,30]
- The second point at [25, 10, 10]

Neither points (or indeed any points along the axis) should be changed by application of the compound transformation

- This is easily verified (e.g. Matlab)
- This is also a good test to ensure your code is working correctly

(Also easily seen that off-axis points are transformed as required)

Remaining topics

- **Representation**
  - Planar Surfaces
  - Constructive Solid Geometry
  - Solids/Voxel modelling
- **Camera Models/Scene Views**
  - Perspective
- **Shading and Lighting**
  - Shading models
- **(Distributed VR)**
  - (Perception Filters)

Representation

An scene can contain different type of objects (clouds, trees, stones, buildings, furniture etc.). For all of them, a wide variety of representation models are available

- **Polygonal surfaces and quadrics**
- **Spline surfaces**
- **Solid modeling**
- **Volumetric models**
- **Procedural models (fractals, particle systems,…)**
- **Physic based modeling**

Representation

There are lots of methods of representing a surface and each has advantages/disadvantages

- **3D Models**
  - **Non-Polygonal Models**
    - Constructive Solid Geometry
    - Implicit Surfaces
    - Parametric Surfaces
  - **Polygonal Models**
    - Structured
      - *Convex*
      - *Non Convex*
    - Polygon Soups

Planar Surface Modelling

Polygonal Modelling

- **Polygon mesh: vertex, edges and polygon collection where each edge is shared by two polygons as maximum**
  - vertex: point with coordinates (x,y,z)
  - edge: line segment that joins two vertices
  - polygon: closed sequence of edges
- **Different type of representation that can be used at the same time in the same application**
  - Explicit
  - Pointers to list of vertices
  - Pointers to list of edges
- **Criteria to evaluate different representations:**
  - Time
  - Space
  - Topological Information

Planar Surface Modelling
Polygonal Modelling

Explicit Representation

- **Each polygon is represented by a list of vertex coordinates**

$$P_1 = \{(x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)\}$$
$$P_2 = \{(x_3, y_3, z_3), \ldots, (x_m, y_m, z_m)\}$$

- **Vertices are stored in order (clockwise or counter-clockwise)**
- **Shared vertices are duplicated**
- **There is no explicit representation for shared vertices and edges**

- **Advantages**
  - Efficient representation for individual polygons
- **Disadvantages**
  - High storage cost
  - In order to move a vertex, it is necessary to traverse all the polygons
  - If the edges are drawn, the shared ones are drawn twice

Planar Surface Modelling
Polygonal Modelling

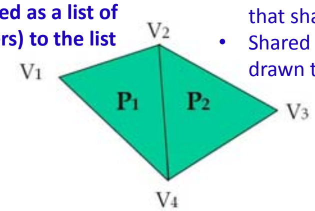
Pointers to list of vertices

- **Each vertex is stored once in a list**

$$V = \{V_1, V_2, \ldots, V_n\}$$
$$= \{(x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)\}$$

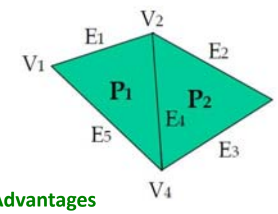
- **A polygon is defined as a list of indexes (or pointers) to the list of vertices**

$$P_1 = \{V_1, V_3, V_4\}$$
$$P_2 = \{V_4, V_2, V_3\}$$



- **Advantages**
  - Each vertex is stored just once
  - Coordinates of vertices can be easily changed
- **Disadvantages**
  - Difficult to find polygons that share an edge
  - Shared edges are still drawn twice

Planar Surface Modelling
Polygonal Modelling



Pointers to list of edges

- **Again, a list of vertices**
- **A polygon is defined as a list of indexes to the list of edges**
- **Each edge points to two vertices and to the polygons it belongs to**

$$E_1 = \{V_1, V_2, P_1, \lambda\}$$
$$E_2 = \{V_2, V_3, P_2, \lambda\}$$
$$E_3 = \{V_3, V_4, P_2, \lambda\}$$
$$E_4 = \{V_4, V_2, P_1, P_2\}$$
$$E_5 = \{V_4, V_1, P_1, \lambda\}$$

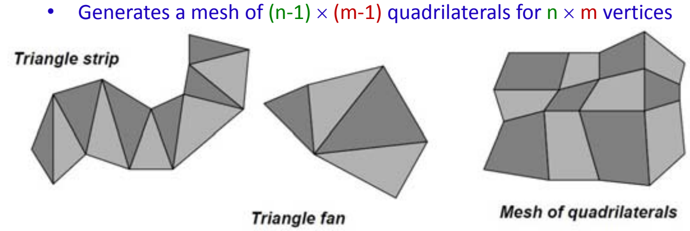
$$P_1 = \{E_1, E_4, E_5\}$$
$$P_2 = \{E_2, E_3, E_4\}$$

- **Advantages**
  - Each vertex is stored just once
  - The shared edges are drawn just once
  - Coordinates of vertices can be easily changed
- **Disadvantages**
  - Difficult to determine which edges share a vertex

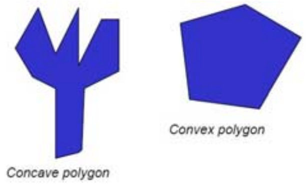
Planar Surface Modelling
Polygonal Meshes

Types of Polygonal Meshes

- **Triangle Strip**
  - For n vertices, produces (n-2) connected triangles
- **Triangle Fan**
  - For n vertices, produces (n-2) connected triangles
- **Mesh of Quadrilaterals**
  - Generates a mesh of (n-1) × (m-1) quadrilaterals for n × m vertices

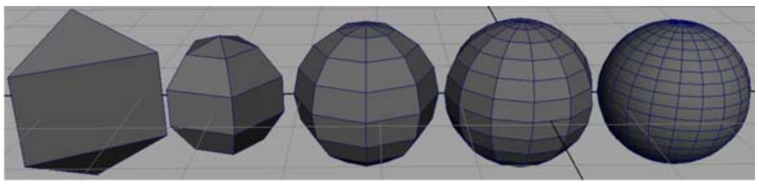


Planar Surface Modelling
Polygonal Modelling



Characteristics

- Comprise any number of polygons, usually triangles or quadrilaterals (quads)
- Concave vs Convex Shapes
- Varying levels of topological information, from none (soups) to complex connectivity graphs
- Not great for smooth curves, need huge numbers of polygons



Planar Surface Modelling
Polygonal Modelling (Convex)

Convex Polygons

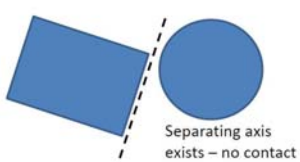
- Can speed up interference tests for Collision Detection if the topology has know properties
- Convex objects are much faster to test
- An object is convex *iff* (if and only if)

  "A line segment between any two points on an object is on or within its boundary"
- Two convex objects can only ever touch at one point or in one plane – very useful

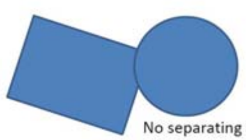
Planar Surface Modelling
Polygonal Modelling (Convex)

Convex Polygons

- If two objects are convex, testing for collision involves finding a plane where all points of object A lie on one side and all points of object B lie on the other
- Known as the Separating Axis Theorem (SAT)
- Many highly optimised algorithms for convex Collision Detection exist

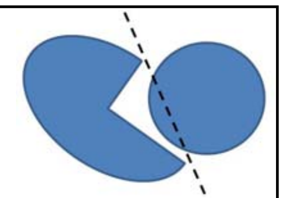


Separating axis exists – no contact



No separating axis exists – objects in contact

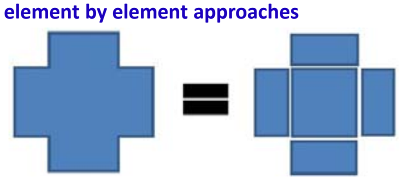
Planar Surface Modelling
Polygonal Modelling (Non-Convex)



No Separating axis exists – but still no contact

Non-Convex Polygons

- Non-convex models much harder
- Can chop up non-convex models into convex pieces (convex decomposition)
- Heuristic search methods can result in incorrect results
- 'Brute-Force' methods test every polygon against every other polygon – which is slow
- Can make use of geometric coherence to speed up element by element approaches



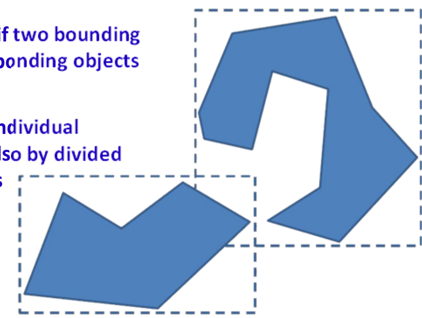
Chopping up complex objects into convex parts greatly speeds up collision detection

Geometric Coherence

- As models become populated with geometric shapes, the geometric coherence of the model needs to be maintained by avoiding overlapping of objects
  - During run-time, this coherence needs to continually tested as objects move or are moved within the environment
- To minimise the number of pair-wise tests a collision detection pass is usually divided into two phases: Broad and Narrow

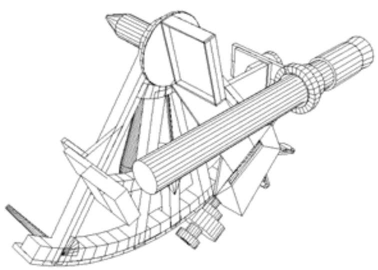
Geometric Coherence

- In a broad phase pass only the bounding boxes of objects are tested
- A bounding box is the smallest box which fits all the geometry of an object within it. Can also use other convex shapes as bounding volumes
- In a narrow-phase pass only if two bounding boxes overlap are the corresponding objects tested
- Objects comprised of many individual elements like polygons can also by divided into a tree of bounding boxes

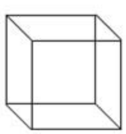


Planar Surface Modelling

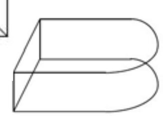
Wireframe Modelling



- **Elements:**
  - points, lines, **arcs** and **circles**, **conic** and **curves**
- **Advantages:**
  - easy to build, low memory requirements and storage
- **Disadvantages:**
  - ambiguous representation (hidden-lines removal algorithms)
  - lack of visual coherence (line-inclusion algorithms)



Ambiguity



Lack of coherence