ROYAL INSTITUTE OF TECHNOLOGY

DH2323 DGI15

INTRODUCTION TO

COMPUTER GRAPHICS AND INTERACTION

MATHEMATICS

For Computer Graphics

Christopher Peters

HPCViz, KTH Royal Institute of Technology, Sweden

chpeters@kth.se

http://kth.academia.edu/ChristopherEdwardPeters

Based on DGI12 notes by Carl Henrik Ek

ROYAL INSTITUTE OF TECHNOLOGY

Broad Area

- We will just ‘scratch the surface’ today
- Example of a good reference text book:

Mathematics for 3D Game Programming and Computer Graphics (3rd ed.)

Eric Lengyel

- There are quite a few others…

ROYAL INSTITUTE OF TECHNOLOGY

What is a 'Light Ray'?

- Concept
  - Idealised narrow beam of light (optics)
  - Discrete, particles
- Geometrically speaking:
  - Similar in some ways to a straight line
  - Has a starting point and direction
  - But extends infinitely in defined direction
- Mathematically:

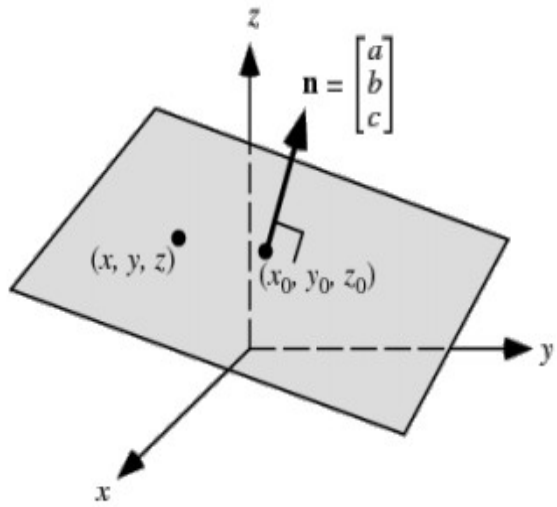
$$\mathbf{r}_0 = [x_0, y_0, z_0]^{\mathrm{T}}$$
$$\mathbf{r}_d = [x_d, y_d, z_d]^{\mathrm{T}} \ , \ ||\mathbf{r}_d|| = 1$$
$$\mathbf{r}_t = \mathbf{r}_0 + t \cdot \mathbf{r}_d$$

One degree-of-freedom

Ray-plane Intersection

- Plane defined as:

- Plane normal $\mathbf{n} = [a, b, c]$
- Unit normal $||n||_2 = 1$
- d offset to origin
- Equation $a \cdot x + b \cdot y + c \cdot z + d = 0$
- *Two degrees-of-freedom*



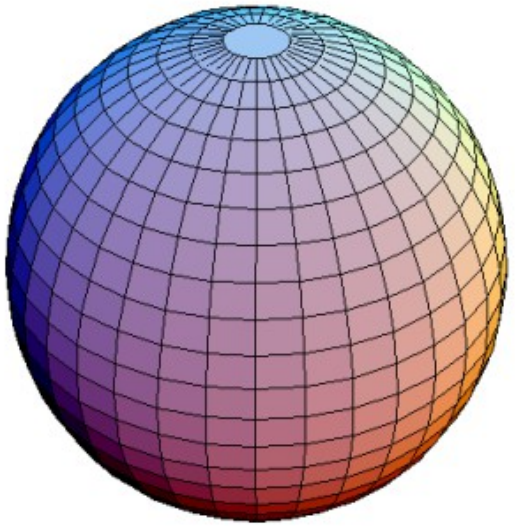
- Intersection point:

$$\mathbf{p}_i = \mathbf{r}_0 - \frac{\mathbf{n}^T\mathbf{r}_0 + d}{\mathbf{n}^T\mathbf{r}_d} \cdot \mathbf{r}_d$$

ROYAL INSTITUTE OF TECHNOLOGY

Ray-sphere Intersection

- Sphere defined as:



- Center of sphere $\mathbf{x}_c = [x_c, y_c, z_c]^{\mathrm{T}}$
- Radius r
- $(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 = r^2$

- Intersections:

$$\mathbf{p}_i = -\mathbf{r}_d^{\mathrm{T}}(\mathbf{r}_0 - \mathbf{r}_c) \pm \sqrt{\mathbf{r}_d^{\mathrm{T}}(\mathbf{r}_0 - \mathbf{r})_c - (\mathbf{r}_0 - \mathbf{x}_c)^{\mathrm{T}}(\mathbf{r}_0 - \mathbf{x}_c) + \mathbf{r}^2}$$

ROYAL INSTITUTE OF TECHNOLOGY

Worked Example

(recommend to work this out later using a pen and paper)

Find the intersections, if any, between the Ray with $r_0 = (0,2,0)^T$, $r_d = (0,-1,0)^T$ and the

Sphere with $x_c = (0,0,0)^T$, $r = 1$

Apply the quadratic formula $t = -b +/- \sqrt{(b^2-4ac)}/(2a)$ to find two solutions, where:

$a = r_d.r_d$

$b = 2r_d.(r_0-x_c)$

$c = (r_0-x_c).(r_0-x_c)-r^2$

The value of b^2-4ac indicates how many roots the equation has, where negative number indicates no intersections between the ray and sphere, a zero indicates a single intersection on the edge of the sphere and a positive number indicates two intersections where the ray enters and exits the sphere. In this example, b^2-4ac is positive indicating **two intersections**.

Apply formula; $r_d.r_d t^2+2r_d.(r_0-x_c)t+(r_0-x_c).(r_0-x_c)-r^2=0$

Entering the above value gives **$t^2-4t+3=0$**

=> t = 3 and t = 1

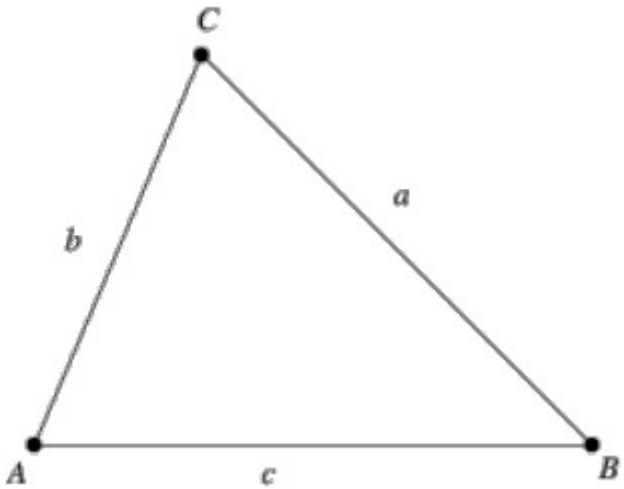
Recalling ray equation: $r_0+t.r_d$

t=1: (0,2,0)+(0,-1,0) = **(0,1,0) … first intersection point**

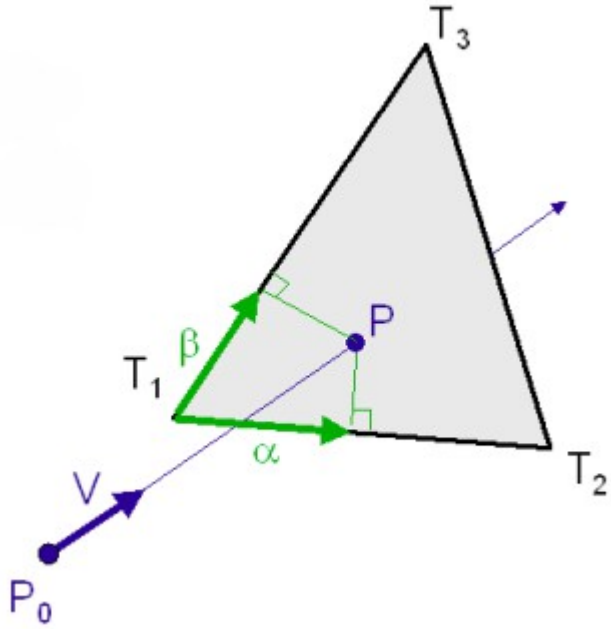
t=3: (0,2,0)+(0,-3,0) = **(0,-1,0) … second intersection point**

Ray-triangle Intersection

- Triangle defined as:
  - Three vertices
    - $\mathbf{t}_i = [x, y, z]^{\mathrm{T}}, i = 1 \ldots 3$



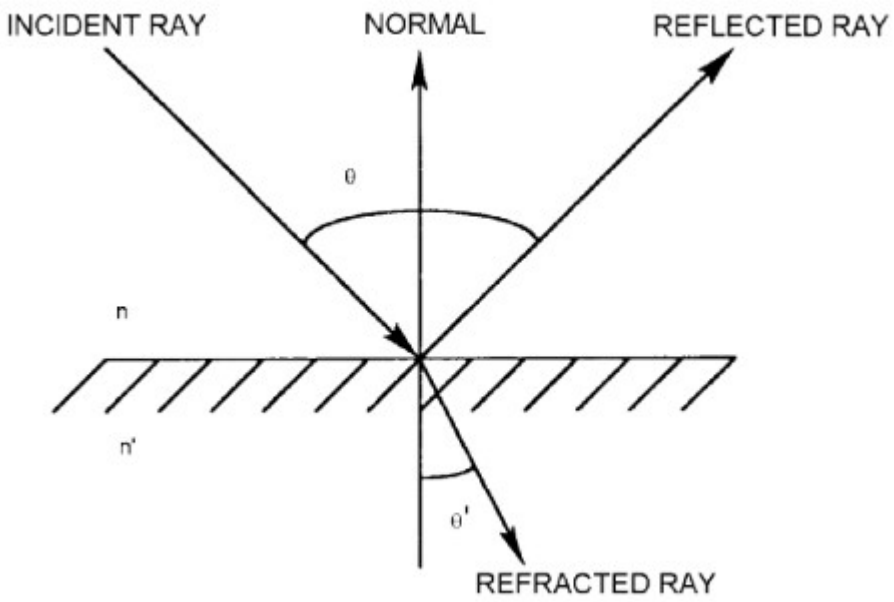
- Intersection:
  1. Check collision with plane
  2. Check if inside triangle
     - $0 \leq \alpha, \beta \leq 1$
     - $\alpha + \beta \leq 1$



Surfaces

Must also consider rays hitting and bouncing off surfaces

1. Incoming ray hits surface
2. Divided into,
   - reflected component
   - refracted component



ROYAL INSTITUTE OF TECHNOLOGY

Nice Results

"Pebbles" by Jonathan Hunt

"Bonsai Life" by Jeremy M. Praay

"Glasses" by Gilles Tran

ROYAL INSTITUTE OF TECHNOLOGY

Basis and Coordinate Systems

In any scene, we need a way to be able to position and orientate points, vectors, objects, etc: We do this be defining a *basis*

The basis is defined by an o*rigin* and a number of *basis vectors*

Can think of the basis as a ‘starting point’

We employ a *Cartesian* basis

The basis vectors are *mutually orthogonal* and *unit length*

- *Unit length: have a length of 1*
- *Mutually Orthogonal: each vector is at a right angle to the others*

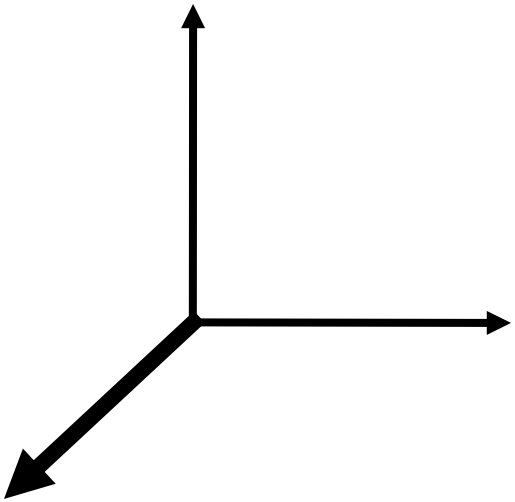
Basis vectors for 3 dimensions

- Named x,y and z

Position in *Cartesian coordinates:* tuple (x,y,z)

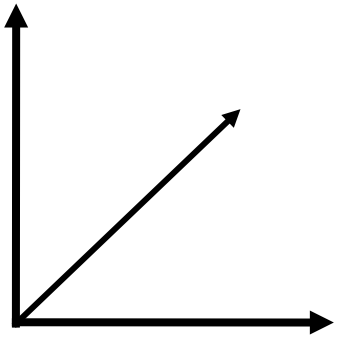
ROYAL INSTITUTE OF TECHNOLOGY

Coordinate Systems



Right handed system

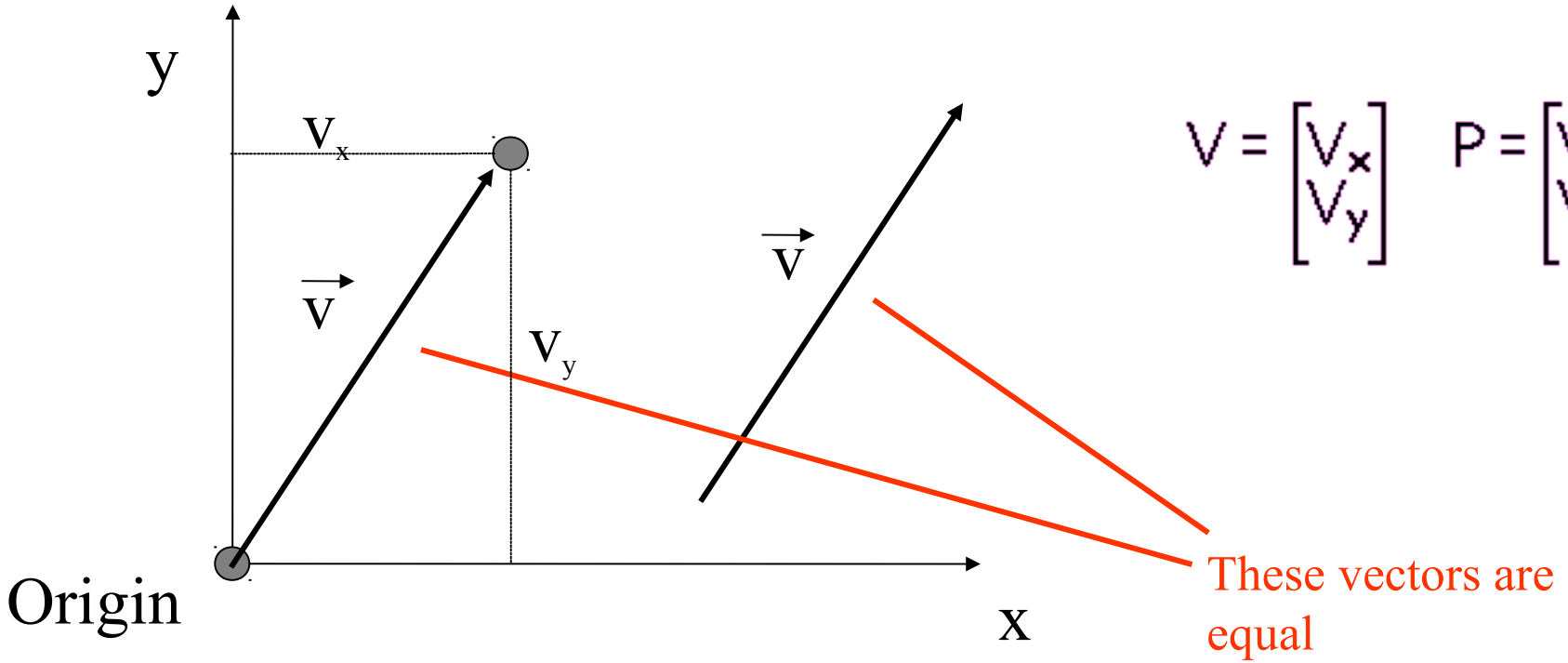
Z comes out of the page



Left handed system

Z goes into the page

Vectors are not the same as positions !



$$V = \begin{bmatrix} V_x \\ V_y \end{bmatrix} \quad P = \begin{bmatrix} V_x \\ V_y \end{bmatrix}$$

Here, x and y are our *basis vectors*

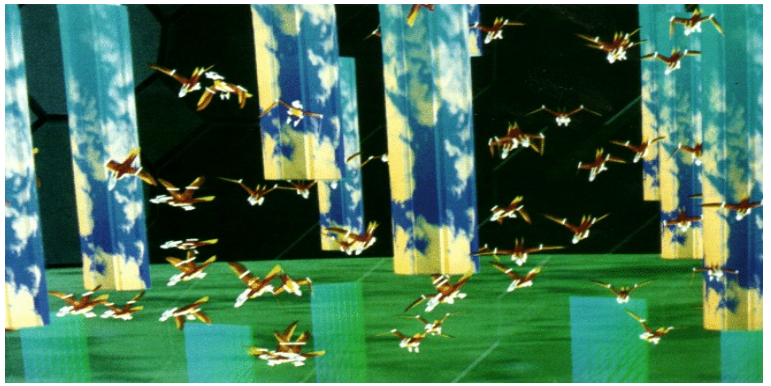
ROYAL INSTITUTE OF TECHNOLOGY

Vector Operations

- Addition
- Subtraction
- Scaling
- Magnitude
- Normalisation
- Dot Product
- Cross Product
- Vital for graphics programming concepts (including flocks and crowds, lighting)

Fish, © Tu and Terzopoulos, 1994

Pedestrians, © Shao and Terzopoulos 2005



Boids, © Reynolds, 1987

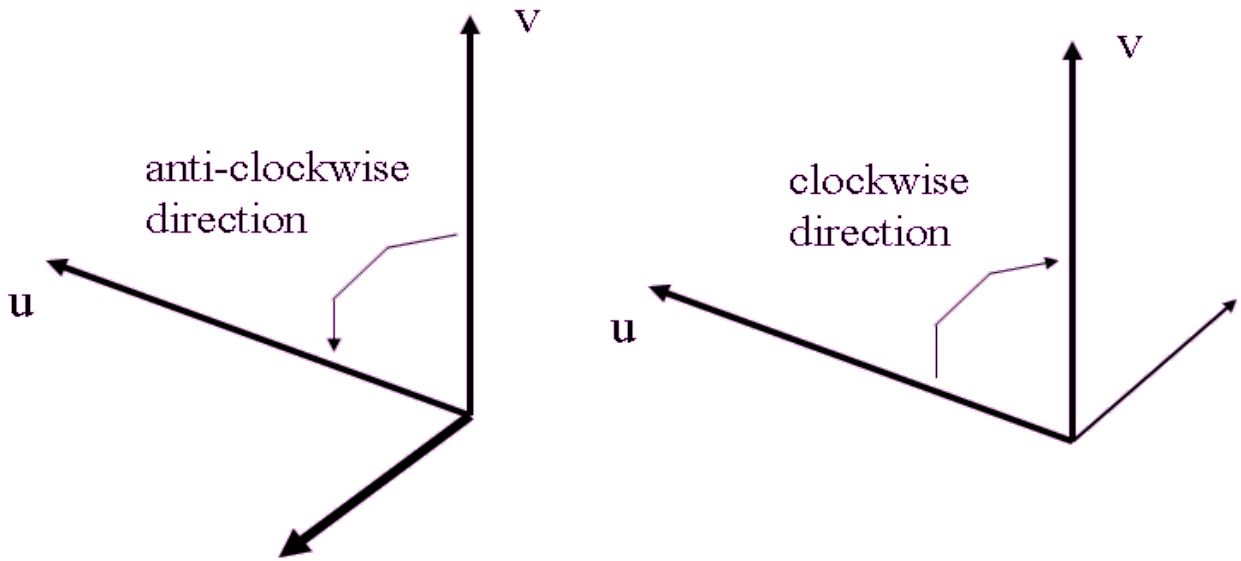
Cross Product

Calculated as:

$$U \times V = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} \times \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} U_2V_3 - U_3V_2 \\ U_3V_1 - U_1V_3 \\ U_1V_2 - U_2V_1 \end{bmatrix}$$

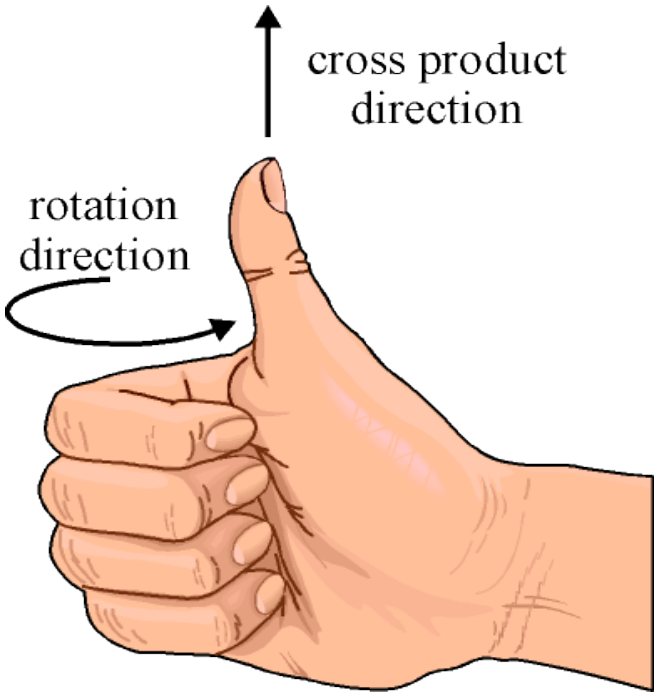
Uses the ‘x’ operator

Result of v x u is a vector, perpendicular to the plane defined by v and u



ROYAL INSTITUTE OF TECHNOLOGY

A *handy* way to remember…



ROYAL INSTITUTE OF TECHNOLOGY

Transformations

Allow us to move, orientate and change the primitives in our scene

- Move, Rotate, Stretch, Squash, Shear

Represented as *matrices:*

"*A matrix is a rectangular array of numbers. The numbers in the array are called the entries in the matrix*"

Change of basis: transformation matrix allows us to transform a vector from one basis to another

Be careful: row vs. column format matrices!!

ROYAL INSTITUTE OF TECHNOLOGY

Row Vs. Column Format

Remember this?

$$\mathbf{r}_0 = [x_0, y_0, z_0]^T$$

$$\mathbf{r}_d = [x_d, y_d, z_d]^T, \; ||\mathbf{r}_d|| = 1$$

$$\mathbf{r}_t = \mathbf{r}_0 + t \cdot \mathbf{r}_d$$

One degree-of-freedom

Row Vs. Column Format

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \neq \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} \quad \left(= \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}^T \right)$$

column format

$$\mathbf{Mv} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

row format

$$\mathbf{v}^T\mathbf{M}^T = \begin{bmatrix} u & v & w \end{bmatrix} \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix}$$

transposed

$$\mathbf{Mv} = (\mathbf{v}^T\mathbf{M}^T)^T$$

ROYAL INSTITUTE OF TECHNOLOGY

Transformations

Example:

Store a translation and a rotation in a matrix

When we apply this matrix to an object, it will be translated and rotated as specified by the matrix

So:

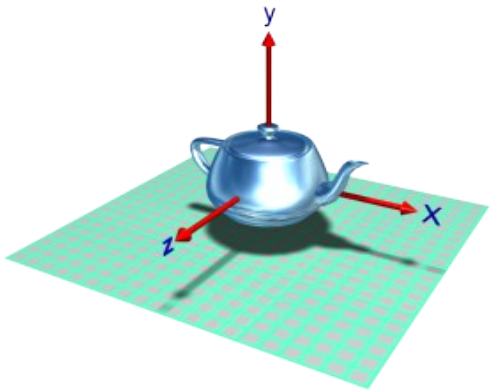
Moving and orientating primitives is a matter of creating the proper matrices and applying them at the correct time

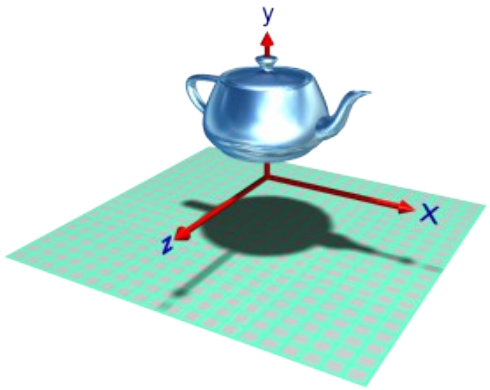
Translation

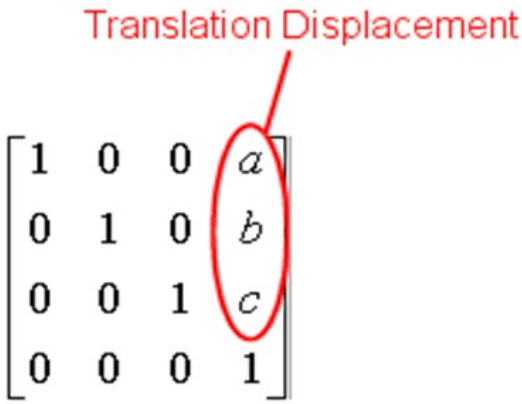
Think of translations as ‘moving’ without rotating

Translation only applies to points

Doesn’t apply to vectors, since vectors are just directions







These correspond to the displacements in the x, y and z directions

So 12th position is the translation in the x direction

ROYAL INSTITUTE OF TECHNOLOGY

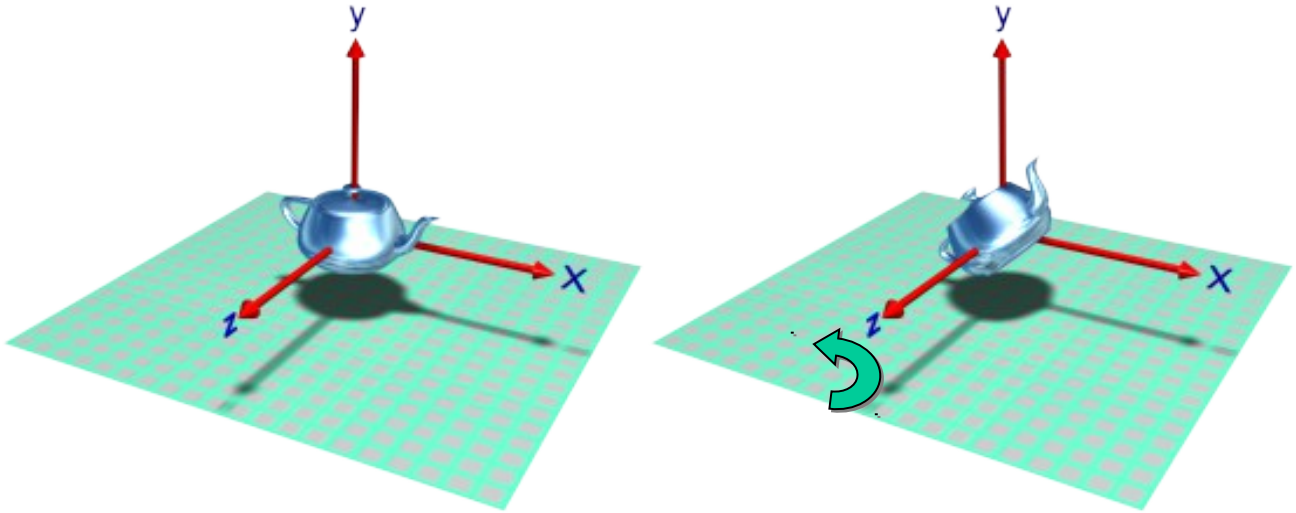
Rotation

Change the orientation of a primitive, without affecting its position

Rotation applies to both points and vectors

Rotating a vector will change its direction

Rotations are conducted *anti-clockwise* about the origin

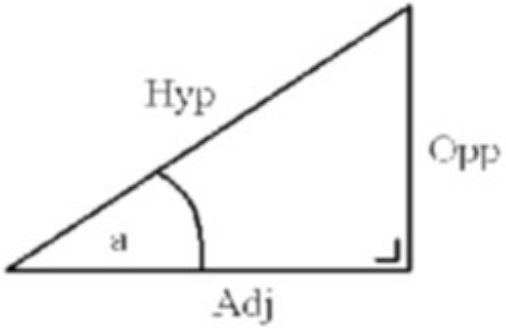


ROYAL INSTITUTE OF TECHNOLOGY

Rotation

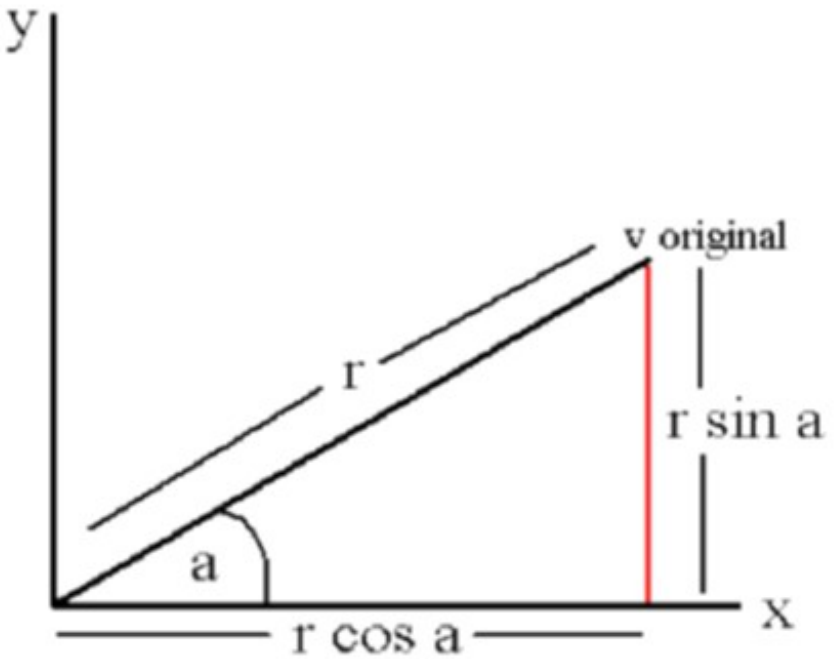
Remember:

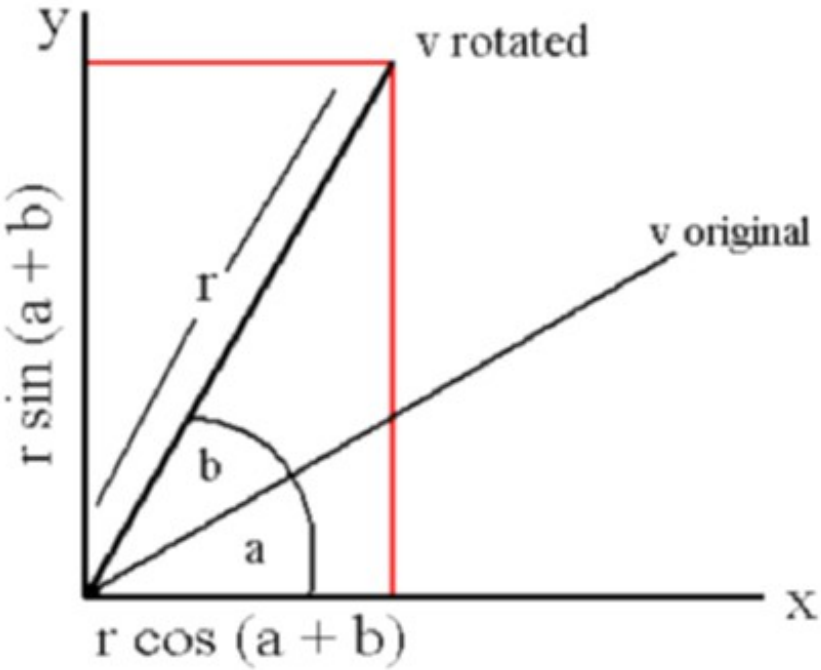
Sin a = Opp / Hyp
Cos a = Adj / Hyp
Tan a = Opp / Adj



$$\text{original } v = \begin{bmatrix} r\cos a \\ r\sin a \end{bmatrix}$$

$$\text{rotated } v = \begin{bmatrix} r\cos(a+b) \\ r\sin(a+b) \end{bmatrix}$$





Rotation

Derivation:

Expanding (a + b) from log tables:

Rotated x = r cos a cos b – r sin a sin b

Rotated y = r cos a sin b + r sin a cos b

But:

Original x = r cos a

Original y = r sin a

So:

Rotated x = original x cos b – original y sin b

Rotated y = original x sin b + original y cos b

Elements 0, 1, 2, 4, 5, 6, 8, 9, 10 define any rotations in our transformation matrix

More on all this later...

ROYAL INSTITUTE OF TECHNOLOGY

The Rotation Matrix

Rotations around the: x-axis (Rx), y-axis (Ry) and z-axis (Rz)

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{R}_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

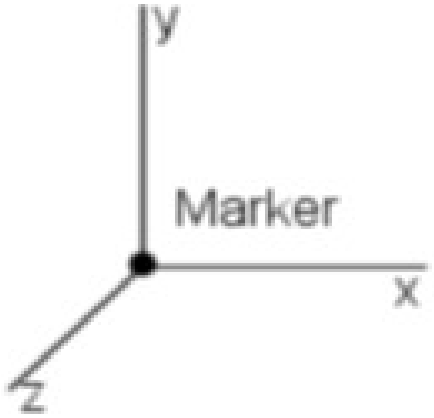
ROYAL INSTITUTE OF TECHNOLOGY

Thinking about Transformations

- Transformation matrix is used for both the camera transformation and model transformation
  - Rotating a model in one direction is equivalent to rotating the camera in the opposite direction
  - Applying a translation along the x axis to this matrix could be interpreted as:
    1. Moving the scene to the right
    2. Moving the camera to the left
- Two ways of looking at transformations: *grand, fixed coordinate system*, and *local coordinate system*

Local Coordinate Marker

Helps us visualise how objects will be placed



Imagine such a marker existing in 3d space:

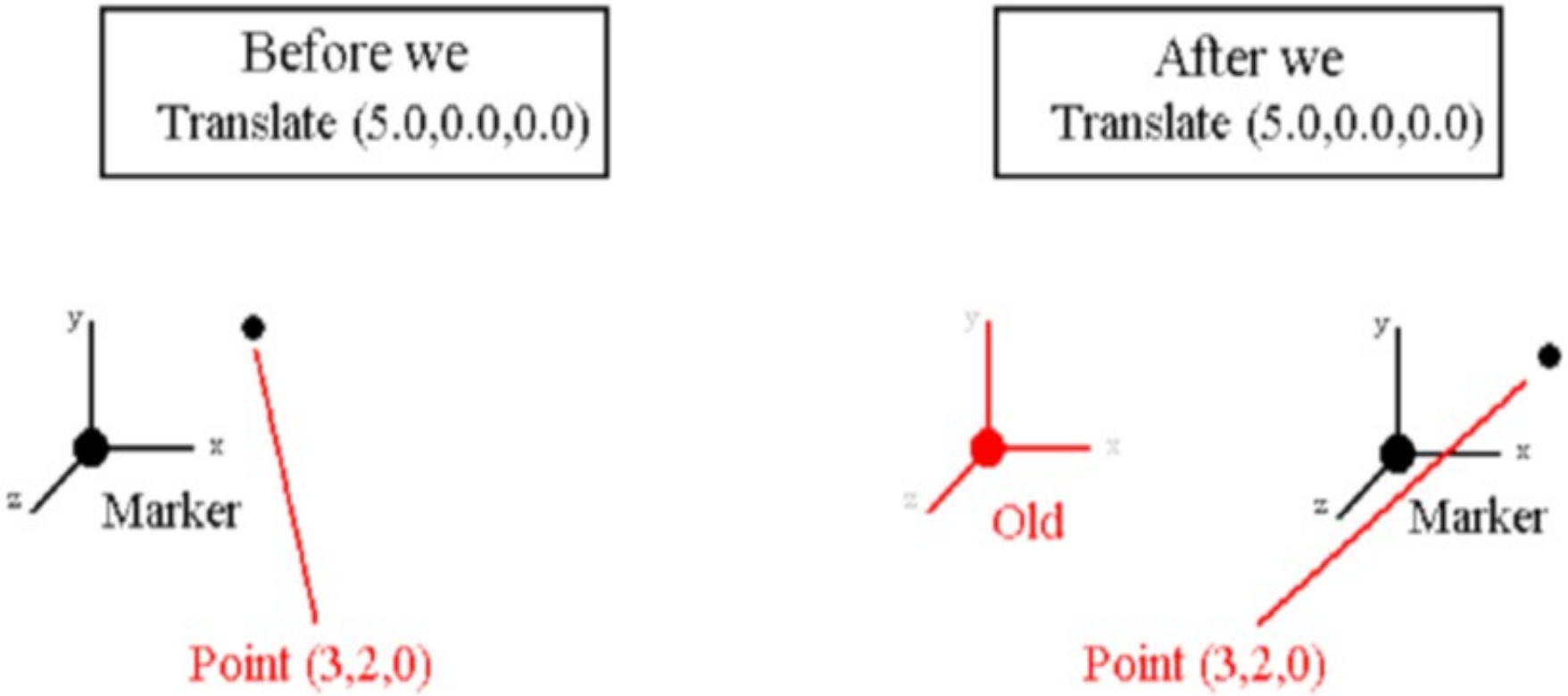
Marker starts in the same position/orientation as the **worldspace** coordinate system origin/basis vectors

All transformations are applied to the marker

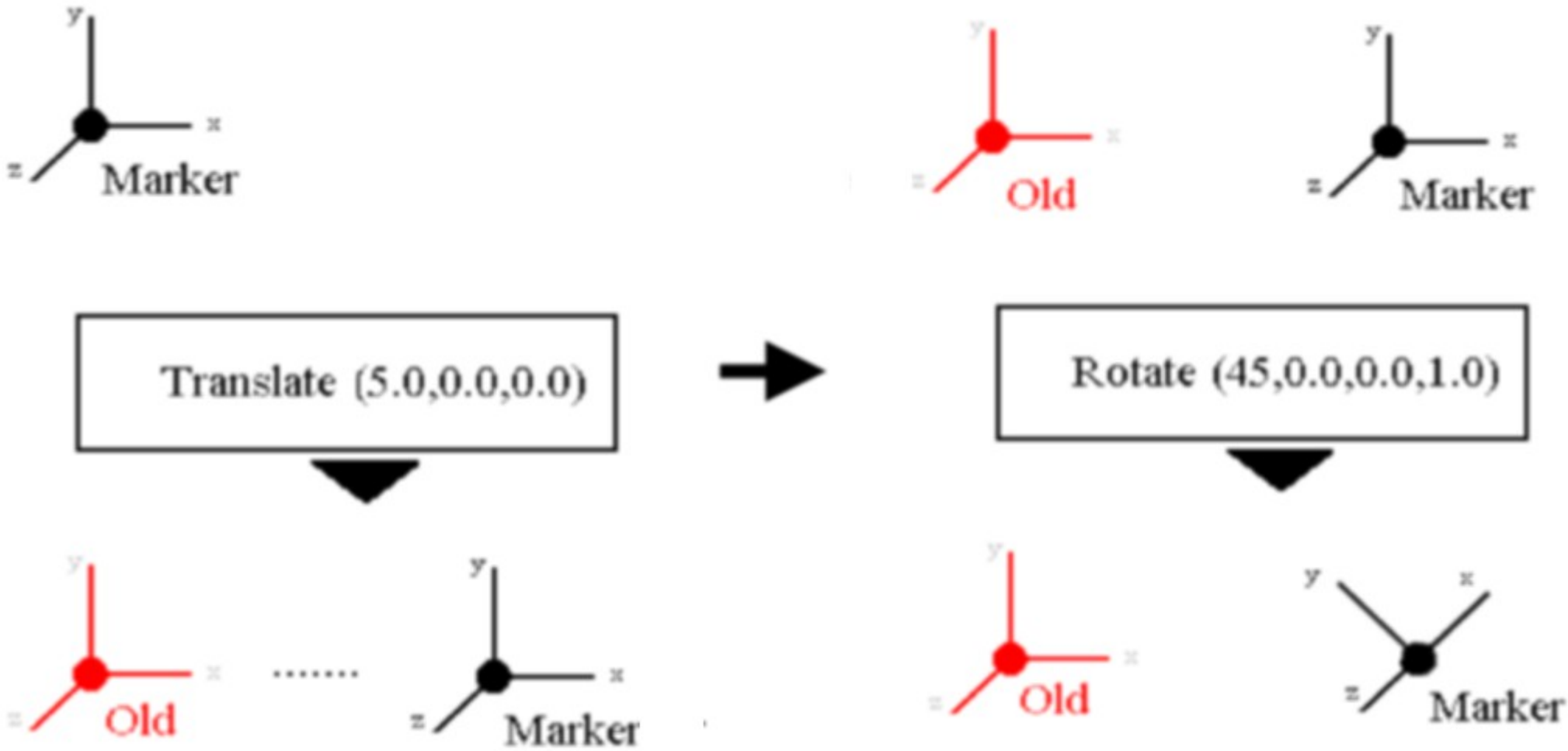
- Translate: we move this marker
- Rotate: we rotate the marker

Local Coordinate Marker

This shows the different between the world-space coordinates of a point before and after the model-view matrix has undergone a translation of 5,0,0



Local Coordinate Marker



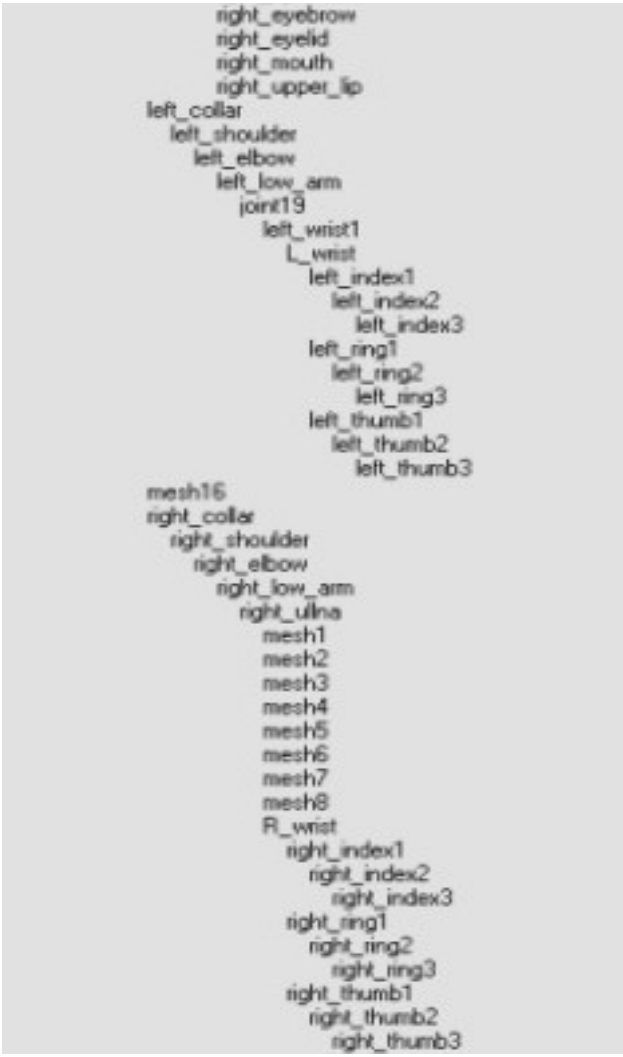
ROYAL INSTITUTE OF TECHNOLOGY

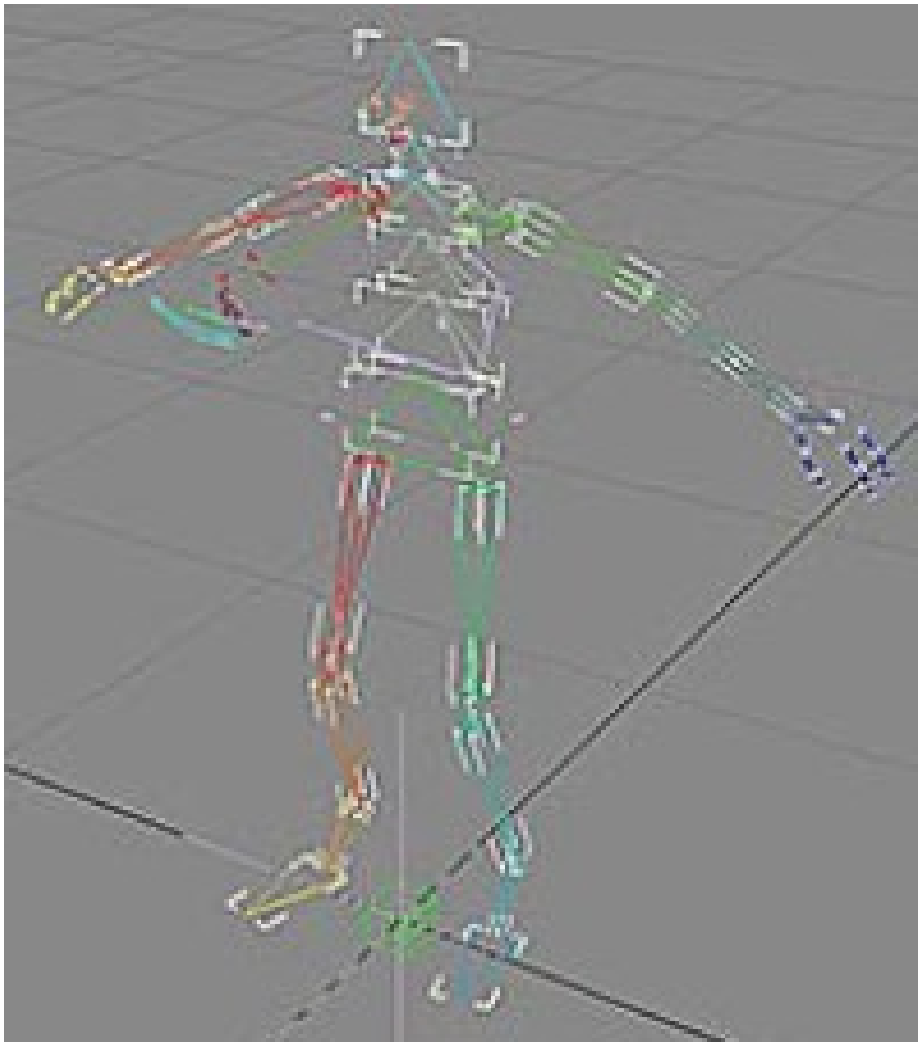
A Note on Hierarchies

- Tree of objects
  - The positions and orientations of objects further down the tree are dependant on those higher up
  - Parent and child objects
- Examples:
  - A human arm
    - the position and orientation of the hand depends on the position and orientation of the elbow, which in turn depends on that of the shoulder, and so on…
  - The Solar system
    - Solar bodies rotate about their own axes as well as orbiting around the Sun, which itself travels relative to our galaxy

ROYAL INSTITUTE OF TECHNOLOGY

A Note on Hierarchies





ROYAL INSTITUTE OF TECHNOLOGY

A Note on Hierarchies

ROYAL INSTITUTE OF TECHNOLOGY

How do we do this?

In order to position an object, the marker should first be:

- positioned at the world-space origin
- orientated to match the world-space basis vectors

Once the marker is in this position, any coordinates we give correspond to world-space coordinates

- WS coordinates are relative to the 3D scene

Things are different in a hierarchy

Objects depend on another object for their position (parent object)

- These objects need to be placed relative to their parent objects' coordinates, rather than relative to the 3D scene
- In practice, applying a sequence of transformations

ROYAL INSTITUTE OF TECHNOLOGY

Links

OpenGL tutorials

More on matrices (with OpenGL and GLM source)

GLM

C++ maths library for graphics programming

Swift3D

Hierarchical animation example

ROYAL INSTITUTE OF TECHNOLOGY

Next lecture

- Lighting and Shading
- This week (15th April)
- 15:00 – 17:00 L1

- Lab support session
- Thursday 16th April
- 10:00 – 12:00 VIC Studio