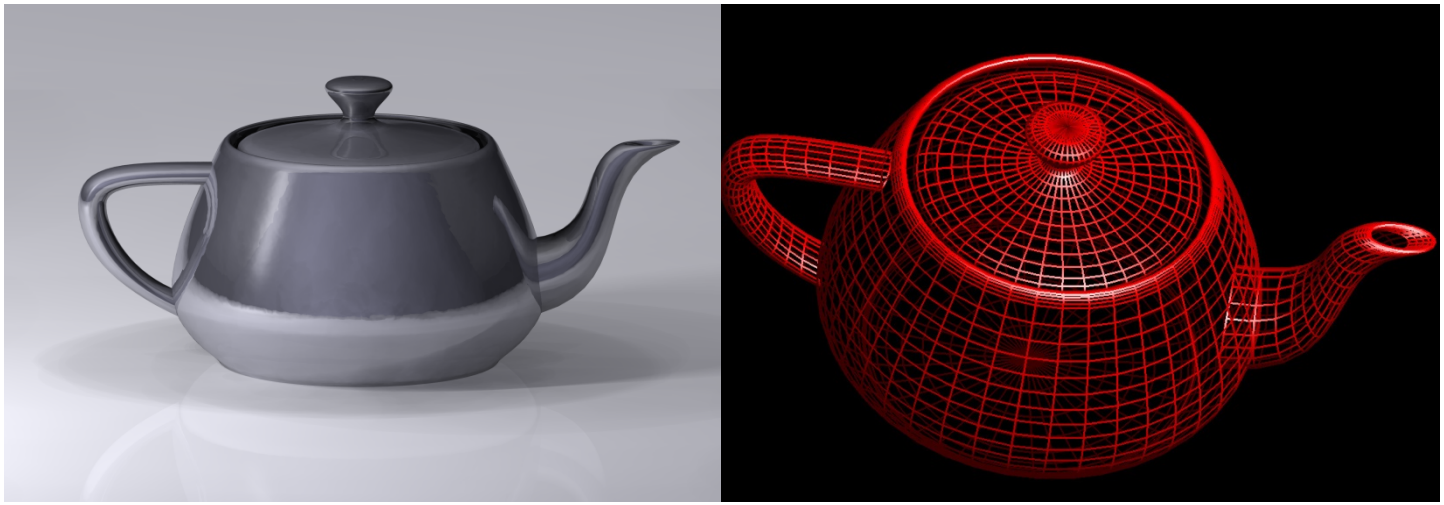


Foundations of Computer Graphics

Saurabh Ray

Reading

Reading for Lectures 6 and 7 : Sections 4.3, 4.7-4.10.

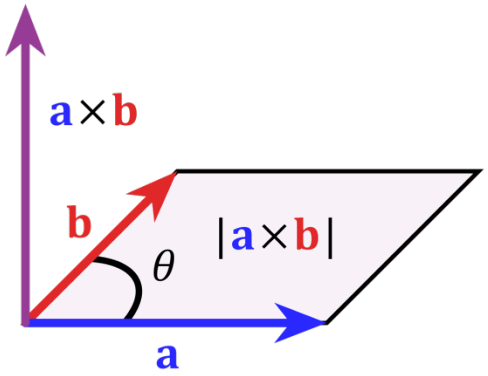
Reading for Lecture 8 : Sections 5.1 - 5.7.

Code on github: https://github.abudhabi.nyu.edu/sr194/CG-2020

Cross Product

$\vec{a}, \vec{b}$: two vectors in *three* dimensions

$\vec{a} \times \vec{b}$ is a **vector**.

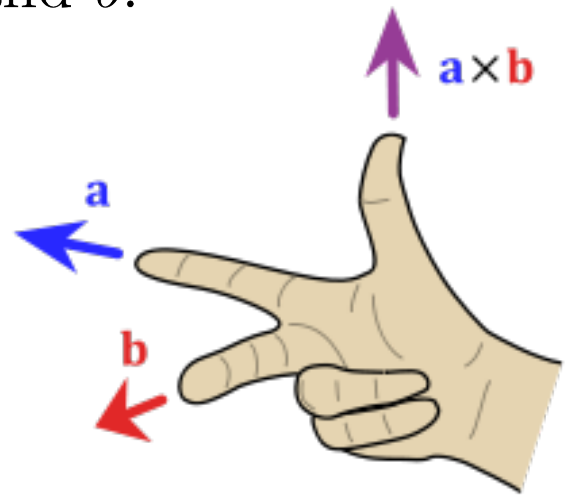


- *magnitude:* $\|\vec{a}\| \, \|\vec{b}\| \, sin\theta$

  (θ: angle between $\vec{a}$ and $\vec{b}$)

- *direction*: $\perp$ to the plane containing $\vec{a}$ and $\vec{b}$.

  given by the right hand rule



Note: $\vec{a} \times \vec{b} = -\vec{b} \times \vec{a}$

cross product is anti-commutative

Cross Product

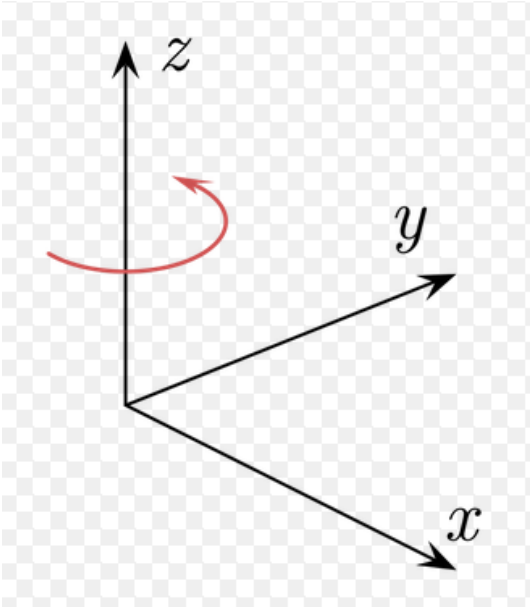
Properties:

- $\vec{a} \times (\alpha\vec{b}) = \alpha(\vec{a} \times \vec{b})$ for any $\alpha \in \mathbb{R}$.
- $\vec{a} \times (\vec{b} + \vec{c}) = \vec{a} \times \vec{b} + \vec{a} \times \vec{c}$ *cross product distributes over addition*

  See `http://www.math.oregonstate.edu/bridge/papers/dot+cross.pdf`

Cross Product

Suppose that we have a right-handed coordinate system.



$\hat{i}$: *unit vector in the positive x direction*

$\hat{j}$: *unit vector in the positive y direction*

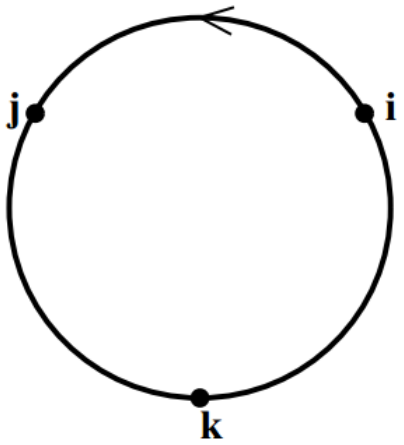
$\hat{k}$: *unit vector in the positive z direction*

Let $\vec{u} = (u_1, u_2, u_3)$. Then, $\vec{u} = u_1\hat{i} + u_2\hat{j} + u_3\hat{k}$

$$\hat{i} \times \hat{j} = \hat{k} \qquad \hat{j} \times \hat{k} = \hat{i} \qquad \hat{k} \times \hat{i} = \hat{j}$$

$$\hat{j} \times \hat{i} = -\hat{k} \qquad \hat{k} \times \hat{j} = -\hat{i} \qquad \hat{i} \times \hat{k} = -\hat{j}$$

$$\hat{i} \times \hat{i} = \hat{j} \times \hat{j} = \hat{k} \times \hat{k} = 0$$



Cross Product

Let $\vec{u} = (u_1, u_2, u_3), \quad \vec{v} = (v_1, v_2, v_3).$

$$\begin{aligned}
\text{Then, } \vec{u} \times \vec{v} = & \ (u_1\hat{i} + u_2\hat{j} + u_3\hat{k}) \times (v_1\hat{i} + v_2\hat{j} + v_3\hat{k}) \\
= & \ (u_2v_3 - u_3v_2)\hat{i} + (u_3v_1 - u_1v_3)\hat{j} + (u_1v_2 - u_2v_1)\hat{k} \\
= & \ (u_2v_3 - u_3v_2, \ u_3v_1 - u_1v_3, \ u_1v_2 - u_2v_1)
\end{aligned}$$

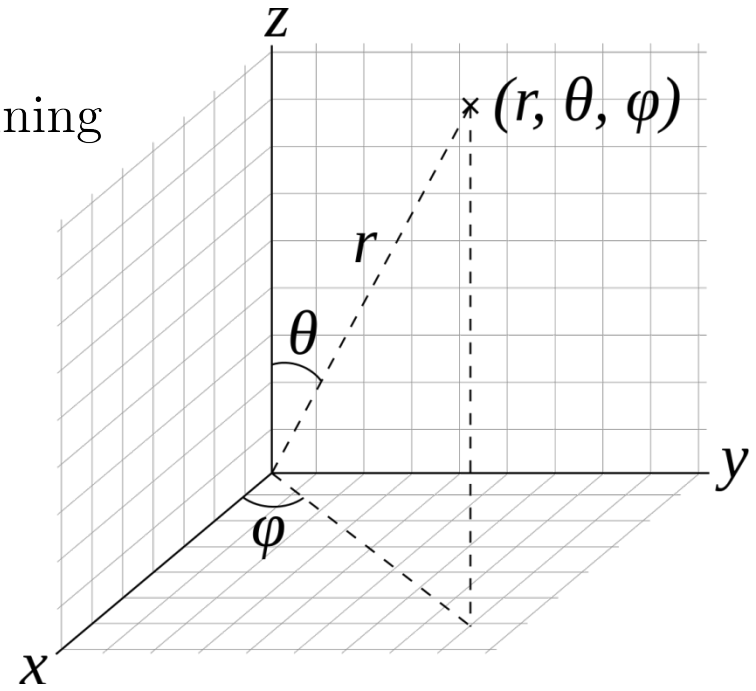
Easy exercise: Find a vector of length 1 that is perpendicular to both the vectors $(1, 1, 1)$ and $(-1, 0, 1)$.

Easy exercise: For vectors $\vec{u} = (2, -1, 7)$ and $v = (-2, 9, 3)$, compute $(\vec{u} + \vec{v}) \cdot (\vec{u} \times \vec{v})$.

Rotation about an arbitrary axis in 3D

Suppose that we want rotate about an axis joining the origin o to the point $p = (r, \theta, \phi)$ (spherical coordinates).

We want counterclockwise rotation by ψ as seen from p.



Idea:

1. Do rotations so that p lies on z-axis. — $Rotate_z(-\phi)$ followed by $Rotate_y(-\theta)$
2. Do rotation about the z-axis. — $Rotate_z(\psi)$
3. Do reverse of Step 1. — $Rotate_y(\theta)$ followed by $Rotate_z(\phi)$

$$Rotate_{\vec{p}}(\psi) = Rotate_z(\phi) \cdot Rotate_y(\theta) \cdot Rotate_z(\psi) \cdot Rotate_y(-\theta) \cdot Rotate_z(-\phi)$$

Coding: Rotation about an axis

Suppose that a point p has cartesian coordinates: (a, b, c)

How do we compute its spherical coordinates?

$$r = \sqrt{a^2 + b^2 + c^2}$$

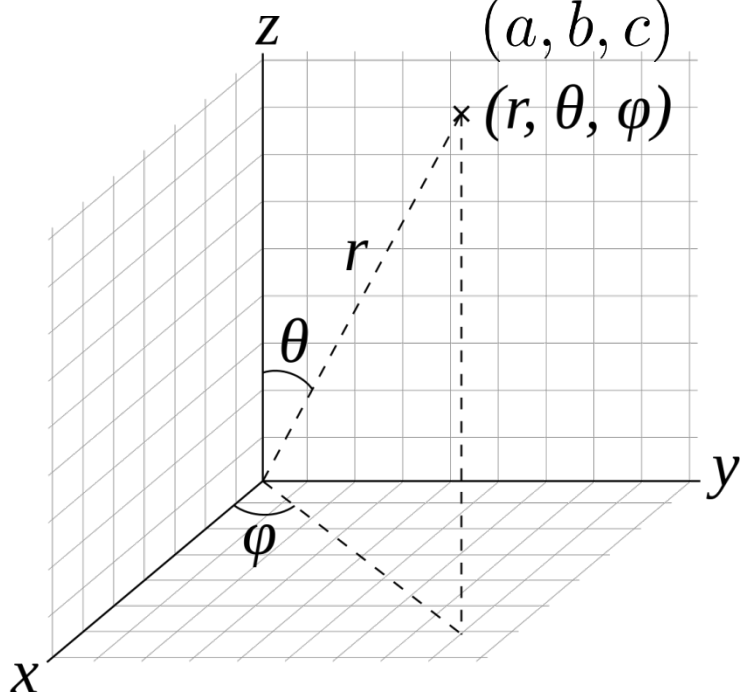
How do we compute θ and ϕ?

$$tan\ \theta = \sqrt{a^2 + b^2}/c \qquad tan\ \phi = b/a$$

Just taking $tan^{-1}(b/a)$ won't give us the right ϕ since we would get the same answer for the point $(-a, -b, c)$ which has a different ϕ.

We need to look at the signs of a and b. `atan2(y,x)` does this.

$$\phi = atan2(b, a) \qquad \theta = atan2(\sqrt{a^2 + b^2}, c)$$



Coding: Rotation about an axis

Plan:

1. Input a vector $\vec{v} = (v_x, v_y, v_z)$ and angle ψ from the user.

2. Compute a 4×4 homogenous transformation matrix M for rotation by angle ψ about the line $o + t\vec{v}$ (o: origin)

3. Send the matrix M to the vertex shaders as a uniform variable and transform each vertex x to Mx

Coding: Rotation about an axis

Input from the user:

In the html file:

```
Vx: <input type="number" id="vx" value=1> </input><br>
Vy: <input type="number" id="vy" value=0> </input><br>
Vz: <input type="number" id="vz" value=0> </input><br>
Angle: <input type = "range" id ="angle"  min=-360 max=360 step=1></input>
```

In the js file:

```
// set rotation axis and angle of rotation
var vx = document.getElementById("vx");
var vy = document.getElementById("vy");
var vz = document.getElementById("vz");
var angle = document.getElementById("angle");
vx.onchange = vy.onchange = vz.onchange = angle.oninput = handler;
```

triggered whenever we move the slider

Coding: Rotation about an axis

Processing the input:

```
function handler(){
    // Get v and psi
    var v = vec3(parseFloat(vx.value), parseFloat(vy.value), parseFloat(vz.value));
    var psi = radians(parseFloat(angle.value));

    // Compute the rotation matrix
    var M = rot(v, psi);

    // Send matrix to the shaders
    gl.uniformMatrix4fv(uM, gl.FALSE, flatten(M));

    // Modify data for drawing the axis
    gl.bindBuffer(gl.ARRAY_BUFFER, vBuffer);
    var w = [-v[0], -v[1], -v[2]];
    gl.bufferSubData(gl.ARRAY_BUFFER,18*sizeof['vec3'],flatten([v,w]));
}
```

Don't transpose. Must be **false**.

Coding: Rotation about an axis

Constructing the rotation matrix:

```
function Rx(theta) {
  var c = Math.cos(theta);
  var s = Math.sin(theta);
  var rx = mat4( 1.0,  0.0,  0.0, 0.0,
      0.0,  c,  -s, 0.0,
      0.0, s,  c, 0.0,
      0.0,  0.0,  0.0, 1.0 );
  return rx;
}
```

```
function Ry(theta) {
  var c = Math.cos(theta);
  var s = Math.sin(theta);
  var ry = mat4( c, 0.0, s, 0.0,
      0.0, 1.0,  0.0, 0.0,
      -s, 0.0,  c, 0.0,
      0.0, 0.0,  0.0, 1.0 );
  return ry;
}
```

```
function Rz(theta) {
  var c = Math.cos(theta);
  var s = Math.sin(theta);
  var rz = mat4( c, -s, 0.0, 0.0,
      s,  c, 0.0, 0.0,
      0.0,  0.0, 1.0, 0.0,
      0.0,  0.0, 0.0, 1.0 );
  return rz;
}
```

```
function matProd(){
    var M = arguments[0];
    for(var i=1; i<arguments.length; ++i){
        M = mult(M,arguments[i]);
    }
    return M;
}
```

```
function rot(v, psi) {
    // psi is in radians
    var a = v[0], b = v[1], c = v[2];

    // compute theta and phi components
    // of the spherical coordinate of v
    var s = Math.sqrt(a * a + b * b);
    var theta = Math.atan2(s, c);
    var phi = Math.atan2(b, a);

    return matProd(Rz(phi),Ry(theta),
                   Rz(psi),
                   Ry(-theta),Rz(-phi));
}
```

Note: *the rotateX() and rotateY() functions in MV.js are incorrect. We therefore write our own functions.*

Coding: Rotation about an axis

render function:

```
function render(now){
    requestAnimationFrame(render);
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

    //draw pyramid
    gl.uniform1i(uID,1);
    gl.drawArrays(gl.TRIANGLES,0,18);

    //draw rotation axis
    gl.uniform1i(uID,2);
    gl.drawArrays(gl.LINES,18,2);
}
```

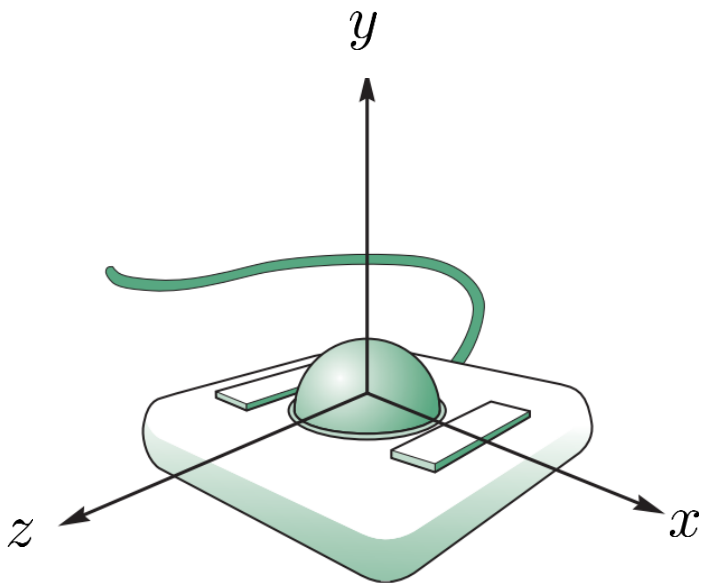
Coding: Rotation about an axis

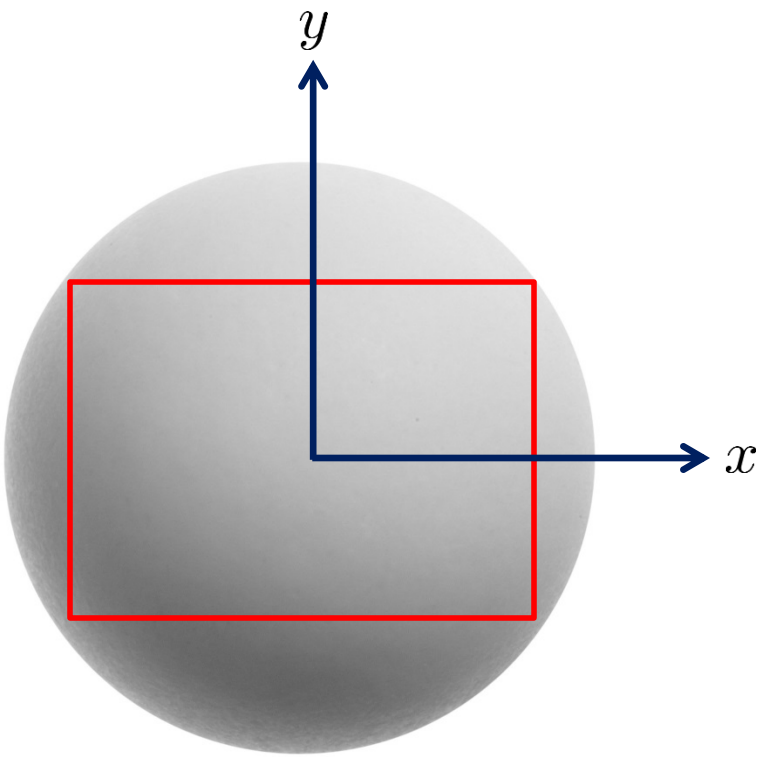
Vertex Shader:

```
attribute vec4 vPosition;
attribute vec4 vColor;
varying vec4 fColor;

uniform mat4 M;
uniform int ID;

void main(){
        fColor = vColor;
        if(ID == 1){
         gl_Position = M*vPosition;   for the pyramid
        }
        else{
         gl_Position = vPosition;     for the axis of rotation
        }
        gl_Position.z *= -1.0; // since webGL uses a left
                               // handed coordinate system
}
```

Virtual Trackball

Sphere equation: $x^2 + y^2 + z^2 = r^2$





$(x, y, 0)$ on the xy-plane corresponds to (x, y, z) on the hemisphere,

where $z = \sqrt{r^2 - x^2 - y^2}$

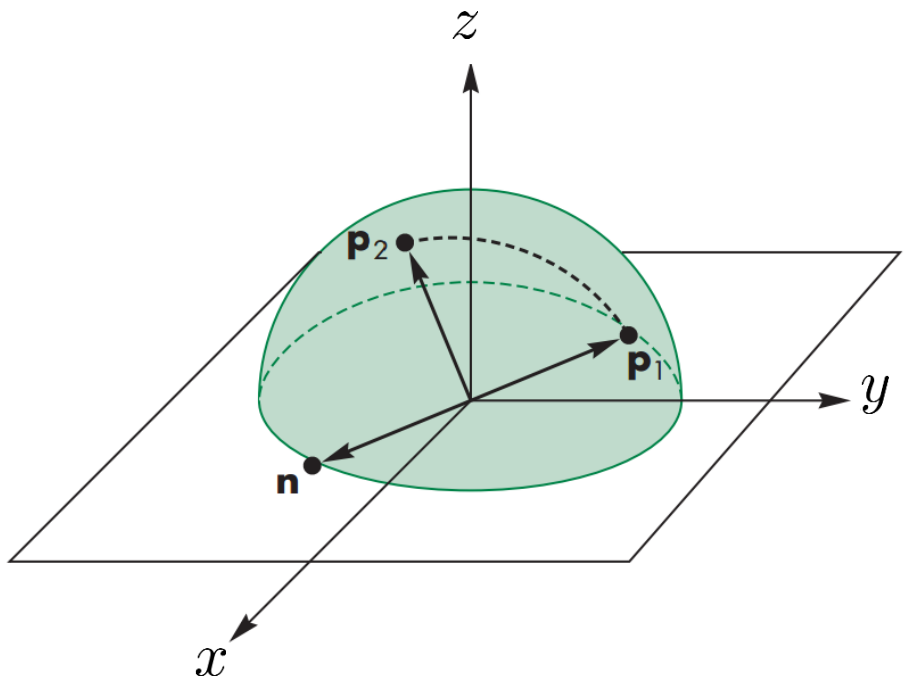
Virtual Trackball

A mouse click a point (x, y) on the xy-plane, can be treated as a click on the point $(x, y, \sqrt{r^2 - x^2 - y^2})$ on the hemisphere.

Suppose that we click at a point p_1 and we drag the mouse and release it at a point p_2.

Let $\vec{p}_1$: unit vector along op_1

$\vec{p}_2$: unit vector along op_2



What is the axis of rotation?

unit vector along: $\vec{p}_1 \times \vec{p}_2$

Angle of rotation θ?

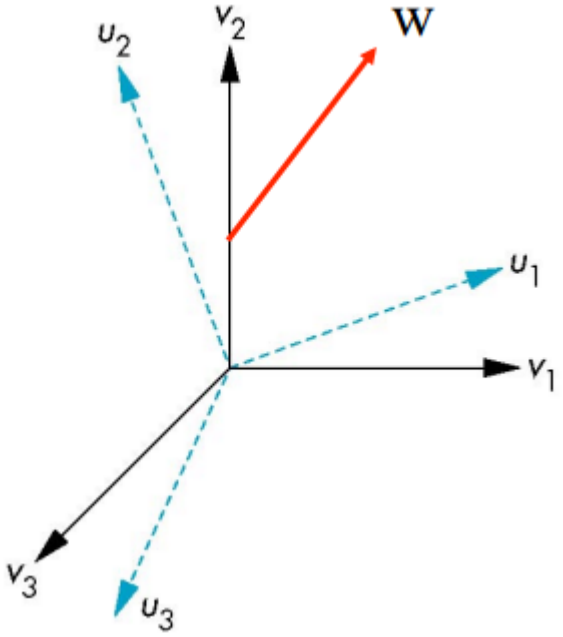
$sin\ \theta = \|\vec{p}_1 \times \vec{p}_2\|$

Change of Coordinate Systems

Let $A : (\vec{v}_1, \vec{v}_2, \vec{v}_3)$ and $B : (\vec{u}_1, \vec{u}_2, \vec{u}_3)$ be two coordinate systems.

Any vector w has one set of coordinates w.r.t. the first system and another set of coordinates w.r.t. the second system.

Given its coordinates in the first system we would like to find its coordinates in the second system.



Let $\vec{w} = \alpha_1\vec{v}_1 + \alpha_2\vec{v}_2 + \alpha_3\vec{v}_3$.

We want to find β_1, β_2 and β_3 s.t.
$\vec{w} = \beta_1\vec{u}_1 + \beta_2\vec{u}_2 + \beta_3\vec{u}_3$.

Idea: Express each of the new basis vectors $\vec{u}_1, \vec{u}_2$ and $\vec{u}_3$ in terms of the old basis vectors $\vec{v}_1, \vec{v}_2$ and $\vec{v}_3$.

Change of Coordinate Systems

Let the coordinates of $\vec{u}_1$ in system A be $(\gamma_{11}, \gamma_{12}, \gamma_{13})$.

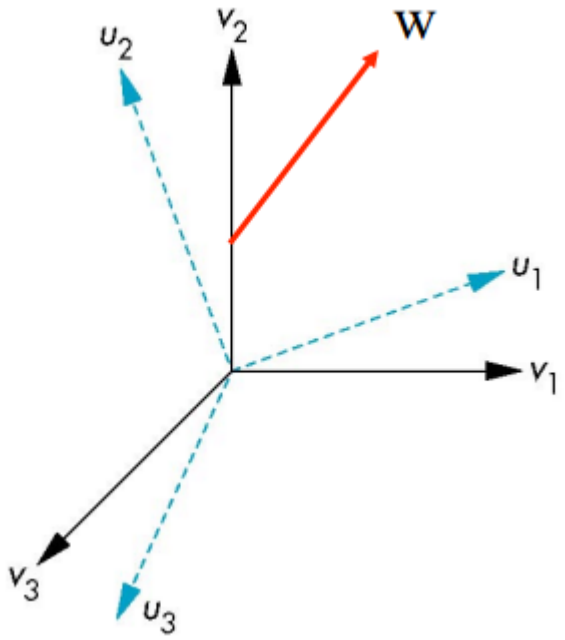
Then,

$$\vec{u}_1 = \gamma_{11}\vec{v}_1 + \gamma_{12}\vec{v}_2 + \gamma_{13}\vec{v}_3$$

Similarly,

$$\vec{u}_2 = \gamma_{21}\vec{v}_1 + \gamma_{22}\vec{v}_2 + \gamma_{23}\vec{v}_3$$

$$\vec{u}_3 = \gamma_{31}\vec{v}_1 + \gamma_{32}\vec{v}_2 + \gamma_{33}\vec{v}_3$$



$$\begin{bmatrix} \vec{u}_1 \\ \vec{u}_2 \\ \vec{u}_3 \end{bmatrix} = \underbrace{\begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} \\ \gamma_{31} & \gamma_{32} & \gamma_{33} \end{bmatrix}}_{M} \begin{bmatrix} \vec{v}_1 \\ \vec{v}_2 \\ \vec{v}_3 \end{bmatrix}$$

$\mathbf{u}$ $\qquad$ $\mathbf{v}$

$$\mathbf{u} = M\mathbf{v}$$

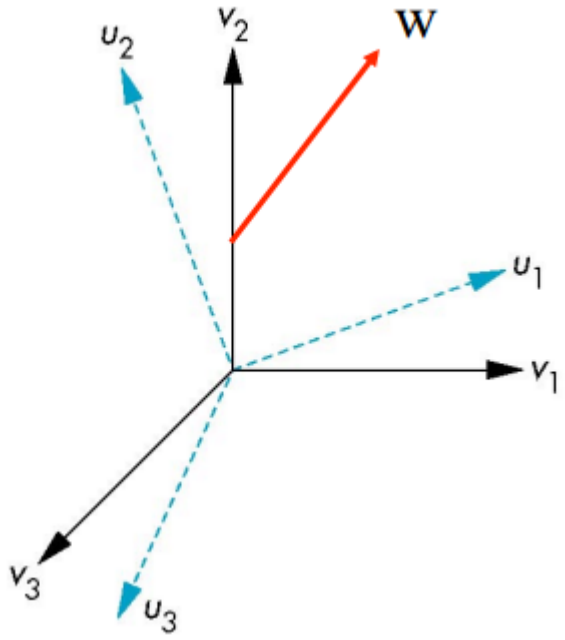
Change of Coordinate Systems

$$\text{Let } \vec{w} = \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 \end{bmatrix} \begin{bmatrix} \vec{u}_1 \\ \vec{u}_2 \\ \vec{u}_3 \end{bmatrix}$$

$$= \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 \end{bmatrix} \mathbf{u}$$

$$= \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 \end{bmatrix} M\mathbf{v}$$

$$(\text{since } \mathbf{u} = M\mathbf{v})$$



Recall, $\vec{w} = \alpha_1\vec{v}_1 + \alpha_2\vec{v}_2 + \alpha_3\vec{v}_3 = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \end{bmatrix}\mathbf{v}$

$$\Longrightarrow \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \end{bmatrix} = \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 \end{bmatrix} M$$

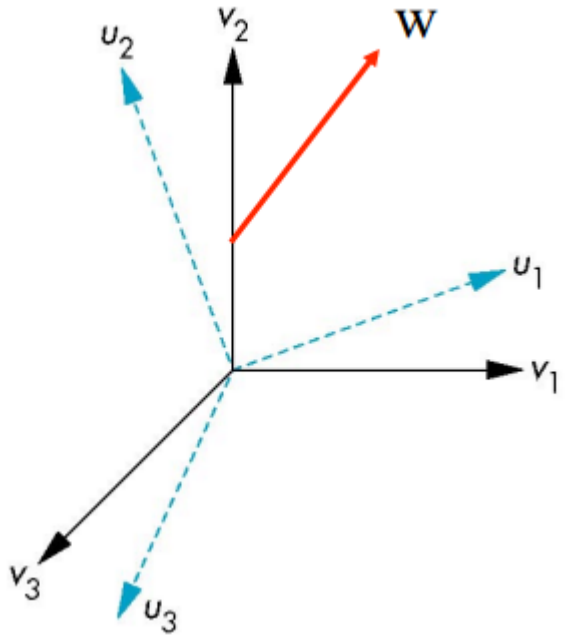
Taking transpose on both sides: $\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = M^T \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}$

Change of Coordinate Systems

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = M^T \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}$$

$\boldsymbol{\alpha}$ — *old coordinate vector*

$\boldsymbol{\beta}$ — *new coordinate vector*



$$\boldsymbol{\alpha} = M^T\boldsymbol{\beta} \quad \Longrightarrow \quad \boxed{\boldsymbol{\beta} = (M^T)^{-1}\boldsymbol{\alpha}}$$

Change of Coordinate Systems

Summary:

old coordinate system $\mathbf{v} = \begin{bmatrix} \vec{v}_1 \\ \vec{v}_2 \\ \vec{v}_3 \end{bmatrix}$

new coordinate system $\mathbf{u} = \begin{bmatrix} \vec{u}_1 \\ \vec{u}_2 \\ \vec{u}_3 \end{bmatrix}$

old coordinate vector $\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}$

new coordinate vector $\boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}$

M: matrix s.t. $\mathbf{u} = M\mathbf{v}$

express new basis vectors in terms of old basis vectors

Then, $\boldsymbol{\beta} = (M^T)^{-1}\boldsymbol{\alpha}$

Change of Coordinate Systems - Example

Let $\vec{v_1}, \vec{v}_2, \vec{v}_3$ be three basis vectors,

Let $\vec{w} = \vec{v}_1 + 2\vec{v}_2 + 3\vec{v}_3$. $\qquad \alpha = [\ 1 \quad 2 \quad 3\ ]^T$

New basis vectors: $\qquad \vec{u}_1 = \vec{v}_1$

$$\vec{u}_2 = \vec{v}_1 + \vec{v}_2$$

$$\vec{u}_3 = \vec{v}_1 + \vec{v}_2 + \vec{v}_3$$

$$\begin{bmatrix} \vec{u}_1 \\ \vec{u}_2 \\ \vec{u}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \vec{v}_1 \\ \vec{v}_2 \\ \vec{v}_3 \end{bmatrix}$$

$$\mathbf{u} \qquad\qquad M \qquad\qquad \mathbf{v}$$

Change of Coordinate Systems - Example

$$\mathbf{u} = M\mathbf{v}.$$

$$\boldsymbol{\beta} = (M^T)^{-1}\boldsymbol{\alpha} \quad = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ 3 \end{bmatrix}$$

$$\vec{w} = -\vec{u}_1 - \vec{u}_2 + 3\vec{u}_3$$

Change of Frames

Coordinate Frame A: $(\vec{v}_1, \vec{v}_2, \vec{v}_3, P_0)$.

A point p is represented as: $\alpha_1\vec{v}_1 + \alpha_2\vec{v}_2 + \alpha_3\vec{v}_3 + P_0$.

$$p = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & 1 \end{bmatrix} \begin{bmatrix} \vec{v}_1 \\ \vec{v}_2 \\ \vec{v}_3 \\ P_0 \end{bmatrix} \qquad \textit{Homogeneous coordinate vector:} \quad \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ 1 \end{bmatrix}$$

A vector w is represented as: $\beta_1\vec{v}_1 + \beta_2\vec{v}_2 + \beta_3\vec{v}_3$.

$$w = \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 & 0 \end{bmatrix} \begin{bmatrix} \vec{v}_1 \\ \vec{v}_2 \\ \vec{v}_3 \\ P_0 \end{bmatrix} \qquad \textit{Homogeneous coordinate vector:} \quad \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ 0 \end{bmatrix}$$

Note: $0.P = 0, \quad 1.P = P$

Change of Frames

Another Coordinate Frame B: $(\vec{u}_1, \vec{u}_2, \vec{u}_3, Q_0)$.

$$\vec{u}_1 = \gamma_{11}\vec{v}_1 + \gamma_{12}\vec{v}_2 + \gamma_{13}\vec{v}_3$$

$$\vec{u}_2 = \gamma_{21}\vec{v}_1 + \gamma_{22}\vec{v}_2 + \gamma_{23}\vec{v}_3$$

$$\vec{u}_3 = \gamma_{31}\vec{v}_1 + \gamma_{32}\vec{v}_2 + \gamma_{33}\vec{v}_3$$

$$Q_0 = \gamma_{41}\vec{v}_1 + \gamma_{42}\vec{v}_2 + \gamma_{43}\vec{v}_3 + P_0$$

$$\begin{bmatrix} \vec{u}_1 \\ \vec{u}_2 \\ \vec{u}_3 \\ Q_0 \end{bmatrix} = M \begin{bmatrix} \vec{v}_1 \\ \vec{v}_2 \\ \vec{v}_3 \\ P_0 \end{bmatrix} \qquad M = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & 0 \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & 0 \\ \gamma_{31} & \gamma_{32} & \gamma_{33} & 0 \\ \gamma_{41} & \gamma_{42} & \gamma_{43} & 1 \end{bmatrix}$$

$$\boxed{\boldsymbol{\beta} = (M^T)^{-1}\boldsymbol{\alpha}}$$

$\boldsymbol{\beta}$ = *new coord. vector* $\qquad$ $\boldsymbol{\alpha}$ = *old coord. vector*

Change of Frames - Example

Frame: $(\vec{v_1}, \vec{v}_2, \vec{v}_3, P_0)$

New frame:

$$\vec{u}_1 = \vec{v}_1$$

$$\vec{u}_2 = \vec{v}_1 + \vec{v}_2$$

$$\vec{u}_3 = \vec{v}_1 + \vec{v}_2 + \vec{v}_3$$

$$Q_0 = v_1 + 2v_2 + 3v_3 + P_0$$

$$\begin{bmatrix} \vec{u}_1 \\ \vec{u}_2 \\ \vec{u}_3 \\ Q_0 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 2 & 3 & 1 \end{bmatrix}}_{M} \begin{bmatrix} \vec{v}_1 \\ \vec{v}_2 \\ \vec{v}_3 \\ P_0 \end{bmatrix}$$

$$M^T = \begin{bmatrix} \overbrace{\begin{matrix} 1 \\ 0 \\ 0 \\ 0 \end{matrix}}^{\vec{u}_1} & \overbrace{\begin{matrix} 1 \\ 1 \\ 0 \\ 0 \end{matrix}}^{\vec{u}_2} & \overbrace{\begin{matrix} 1 \\ 1 \\ 1 \\ 0 \end{matrix}}^{\vec{u}_3} & \overbrace{\begin{matrix} 1 \\ 2 \\ 3 \\ 1 \end{matrix}}^{\vec{Q}_0} \end{bmatrix}$$

Change of Frames - Example

$$(M^T)^{-1} = \begin{bmatrix} 1 & -1 & 0 & 1 \\ 0 & -1 & -1 & 1 \\ 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Point p: $\begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \end{bmatrix}$ New coordinates: $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

Vector $\mathbf{a} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 0 \end{bmatrix}$ New coordinates: $\begin{bmatrix} -1 \\ -1 \\ 3 \\ 0 \end{bmatrix}$

Typical Frames used in Computer Graphics

- Object (or model) coordinates
- World coordinates
- Eye (or camera) coordinates

The transformations from model to world coordinates and from world to eye coordinates are usually combined into a **Model-View matrix**.

- Clip coordinates

4D Homogeneous coordinates

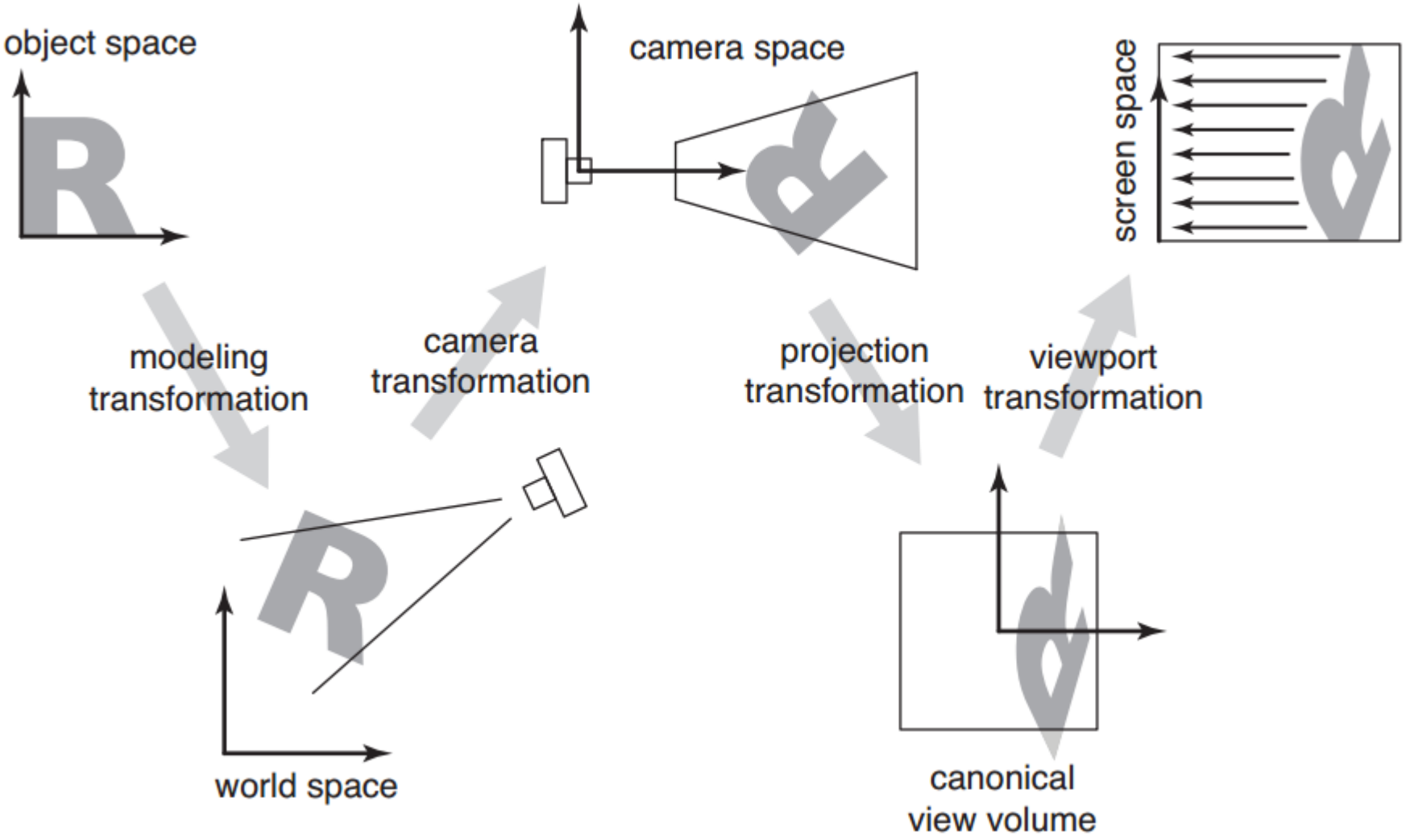
- Normalized device coordinates

3D coordinates

- Window (or screen) coordinates

2D coordinates

Viewing Transformations



Viewing Transformations

Modelling Transformation: One for each object to put it in the world space.

Camera (eye) transformation: Converts from world coordinates to camera coordinates. Just one for the whole scene.

Projection Transformation: Ensures that the view volume is mapped to the canonical view volume which is $[-1,1] \times [-1,1] \times [-1,1]$. Depends on desired projection i.e. "type of lens" used.

Viewport(windowing) Transformation: maps from canonical view volume to screen space - pixel coordinates.