Computer Graphics and Programming

Lecture 7
Object Picking

Jeong-Yean Yang

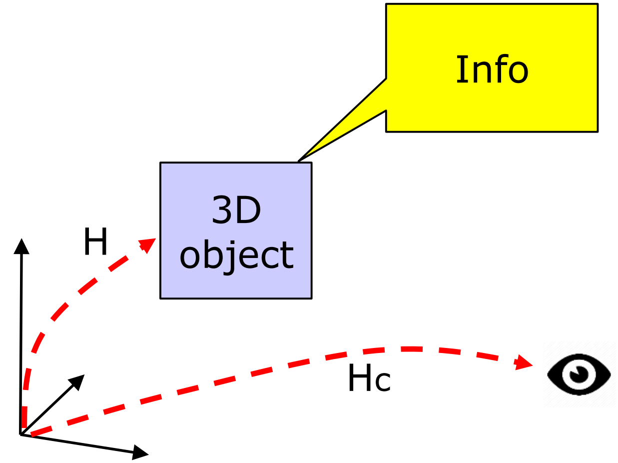
2020/12/8

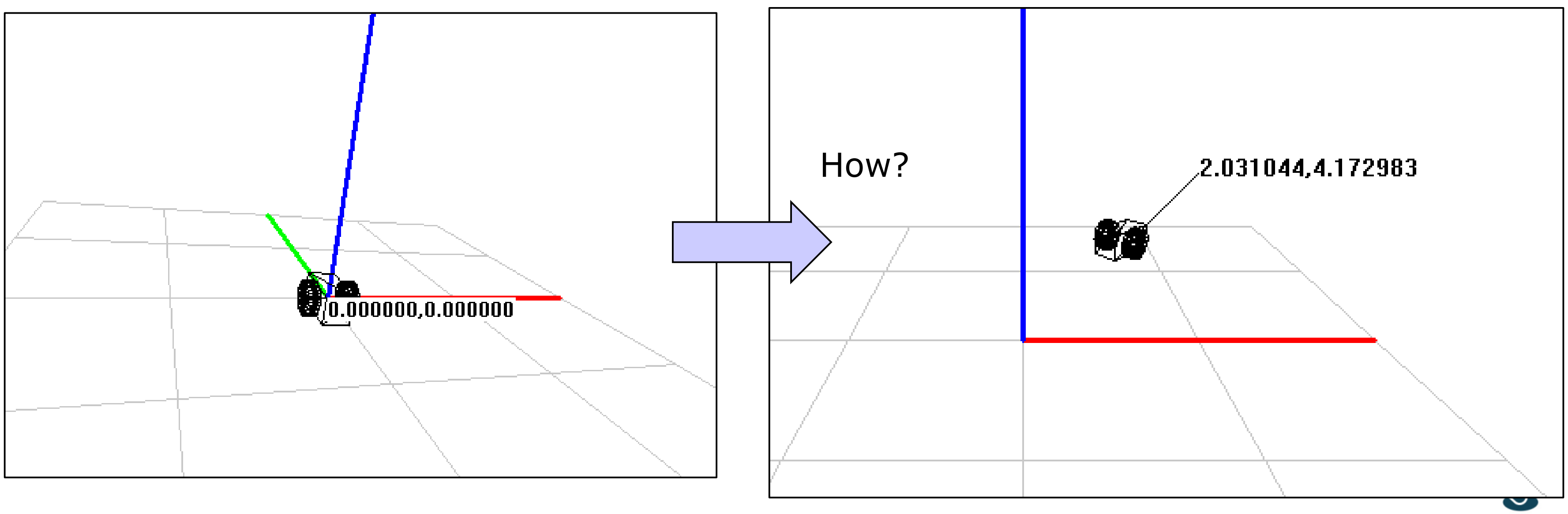
1

Display information(2D) on 3D Objects

Display in 2D Space

```
void uCar::Draw(CDC *pDC)
{
    box.Draw(pDC);
    wheel[0].Draw(pDC);
    wheel[1].Draw(pDC);

    CString buf;
    uVector o   = H.O();
    buf.Format(L"%f,%f",o.x,o.y);

    pDC->TextOut(0,0,buf);
}
```



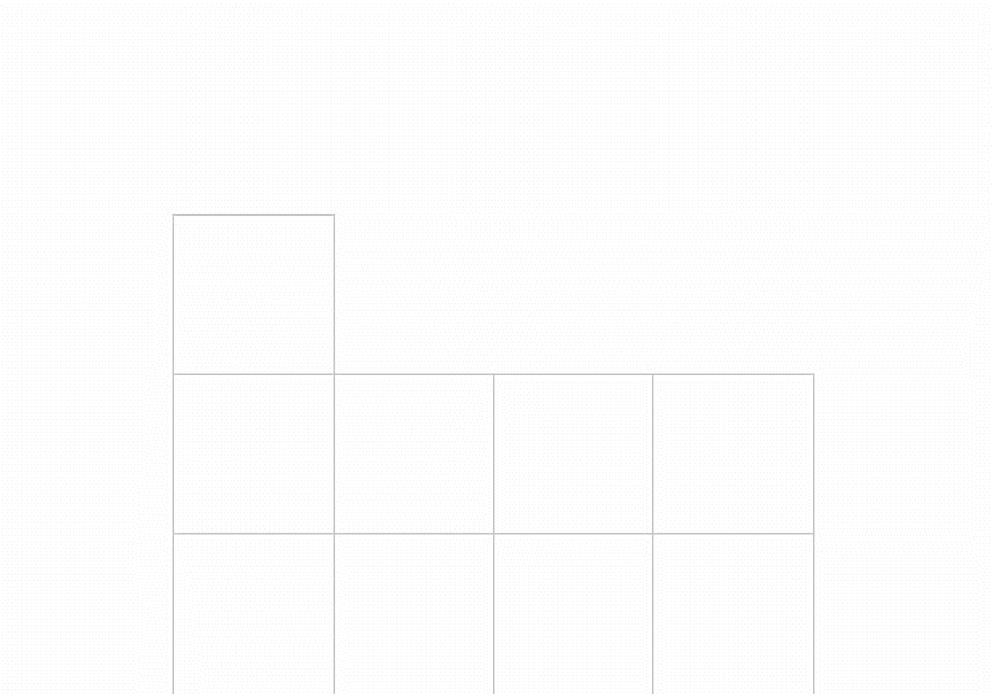


2D Projection Vector is Useful for Information Display

uWnd-51-Car-Info

```
void uCar::Draw(CDC *pDC)
{
    box.Draw(pDC);
    wheel[0].Draw(pDC);
    wheel[1].Draw(pDC);

    CString buf;
    uVector o   = H.O();
    buf.Format(L"%f,%f",o.x,o.y);

    uVector pt = box.pTemp[6];

    pDC->MoveTo(pt.x,pt.y);
    pDC->LineTo(pt.x+30,pt.y+30);
    pDC->TextOut(pt.x+30,pt.y+30+10,buf);
}
```

2D vertex



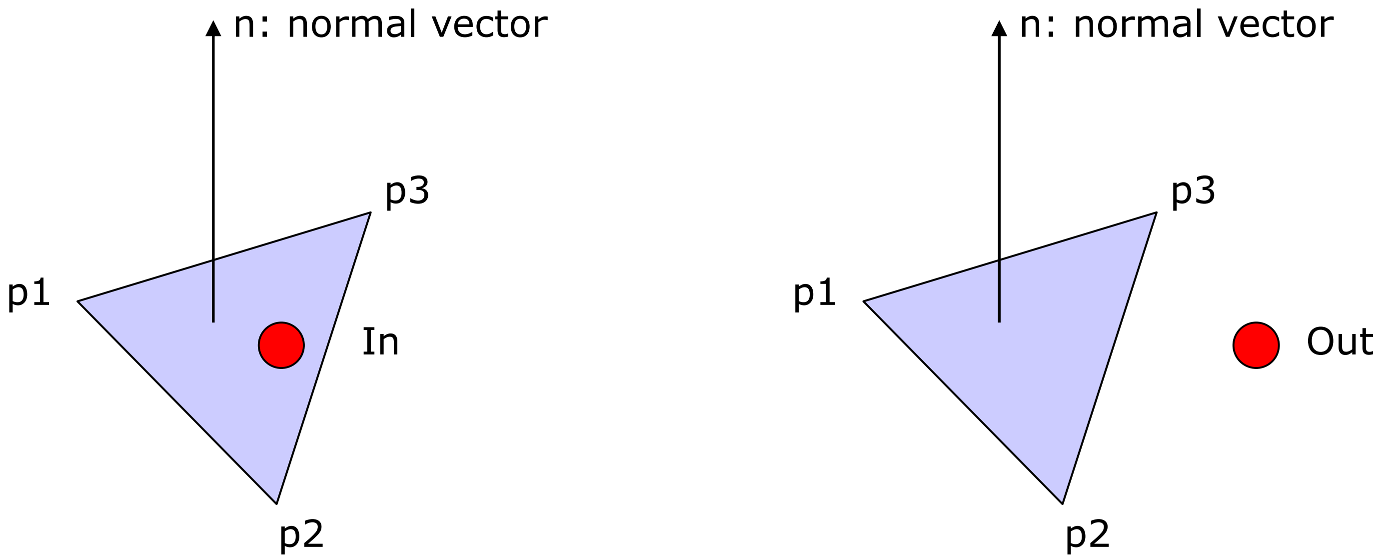
- One vertex at 3D Object is chosen
- Projection of the vertex is used for display 2D information

3D Object Picking by a Click in 2D

2

Introduction

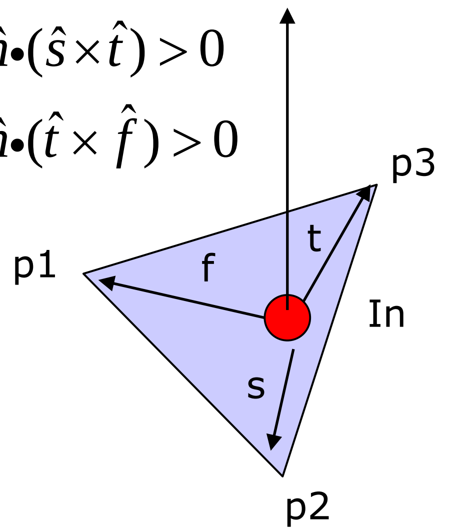
Click IN or Out for 2D Polygon



- How to Discriminate If Red point is **In** or **Out** of polygon
- Our polygon is USING Counter Clockwise.

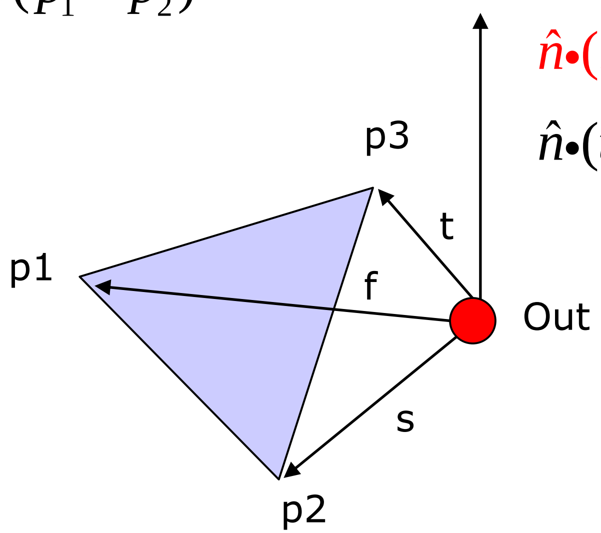
Condition for a Point in a Triangle or Not

$$\hat{n}\bullet(\hat{f}\times\hat{s})>0$$
$$\hat{n}\bullet(\hat{s}\times\hat{t})>0$$
$$\hat{n}\bullet(\hat{t}\times\hat{f})>0$$



$$\hat{n}=(p_3-p_2)\times(p_1-p_2)$$

$$\hat{n}\bullet(\hat{f}\times\hat{s})>0$$
$$\hat{n}\bullet(\hat{s}\times\hat{t})<0$$
$$\hat{n}\bullet(\hat{t}\times\hat{f})>0$$



- All Normal vectors have Positive Z element → In
- All Normal vectors don’t have Positive Z → Out.

Ex)Object Picking in uGroundPick object

uWnd-52-OP-Pick

```
uGroundPick::uGroundPick()
{
    nSelect = -1;
}
```

nSelect for selected polygon index

- uGroundPick
  - is inherited from uGround
- uGround has 16x2 polygons
  - nSelect= -1 :No selection
  - nSelect = 0~31
  - Draw a Selected polygon with RED line

```
void uGroundPick::Draw(CDC *pDC)
{
    int i;

    //black for Non-selected Triangle polygon
    for (i=0;i<nPoly;i++)
    {
        int f,s,t;
        f   = pPoly[i].f;
        s   = pPoly[i].s;
        t   = pPoly[i].t;

        pDC->MoveTo( pTemp[f].x, pTemp[f].y);
        pDC->LineTo( pTemp[s].x, pTemp[s].y);
        pDC->LineTo( pTemp[t].x, pTemp[t].y);
        pDC->LineTo( pTemp[f].x, pTemp[f].y);
    }

    // RED for selected Triangle Polygon
    if (nSelect<0)  return;

    CPen pen,*pold;
    pen.CreatePen(PS_SOLID,2,RGB(255,0,0));
    pold    = pDC->SelectObject(&pen);
    {
        int f,s,t;
        f   = pPoly[nSelect].f;
        s   = pPoly[nSelect].s;
        t   = pPoly[nSelect].t;

        pDC->MoveTo( pTemp[f].x, pTemp[f].y);
        pDC->LineTo( pTemp[s].x, pTemp[s].y);
        pDC->LineTo( pTemp[t].x, pTemp[t].y);
        pDC->LineTo( pTemp[f].x, pTemp[f].y);
    }
```

Draw Black

Draw RED

Ex)Object Picking in uGroundPick object

uWnd-52-OP-Pick

```
void uWnd::OnLButtonUp(UINT nFlags,CPoint point)
{
    ReleaseCapture();

    point.x = point.x-320;
    point.y = -(point.y-240);
    ground.Click(point);
    Redraw();

    CWnd::OnLButtonUp(nFlags,point);
}
```

Screen Display (0,0-640x480)

→ Our Display (-320,240,320,-240)

```
void uGroundPick::Click(CPoint pt)
{
    nSelect= -1;

    for (int i=0;i<nPoly;i++)
    if (pPoly[i].Click(pTemp,pt))
    {
        nSelect = i;
        break;
    }
}
```

Check all Polygons

```
BOOL uPolygon::Click(uVector *pTemp, CPoint pt)
{
    uVector o(pt.x,pt.y,0);

    uVector vf,vs,vt,n;
    vf   = pTemp[f]-o;
    vs   = pTemp[s]-o;
    vt   = pTemp[t]-o;
    vf.z     = 0;
    vs.z     = 0;
    vt.z     = 0;

    float sgn,sgn2,sgn3;

    // first
    n    = vf*vs;
    if (n.z>0)    sgn = 1;
    else          sgn =-1;

    // second
    n    = vs*vt;
    if (n.z>0)    sgn2 = 1;
    else          sgn2 =-1;
    if ( sgn*sgn2<0)      return FALSE;

    // third
    n    = vt*vf;
    if (n.z>0)    sgn3 = 1;
    else          sgn3 =-1;
    if ( sgn*sgn3<0)      return FALSE;
    return TRUE;
}
```

In Triangle

$\hat{z}\bullet(\hat{f}\times\hat{s})$,

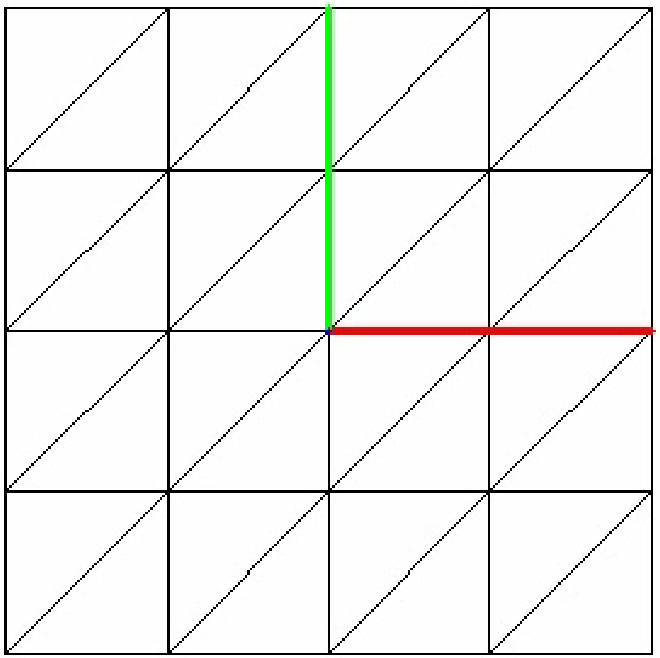
$\hat{z}\bullet(\hat{s}\times\hat{t})$,

$\hat{z}\bullet(\hat{t}\times\hat{f})$,

have SAME sign.

Ex)Object Picking in uGroundPick object

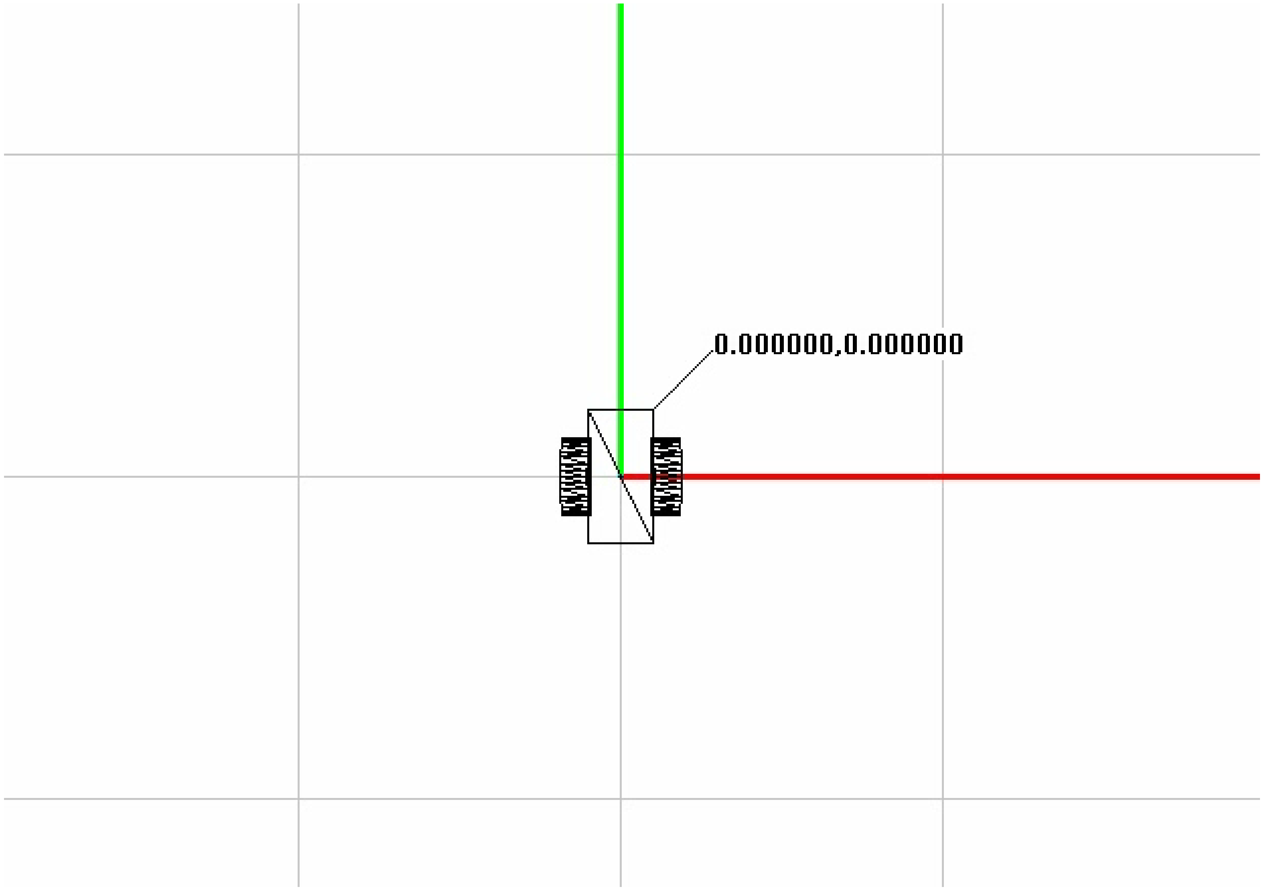
uWnd-52-OP-Pick



Extend Picking into Multiple Object

ex)uWnd-54-Car-Pick-Problem

- CAD like program supports for Object Picking



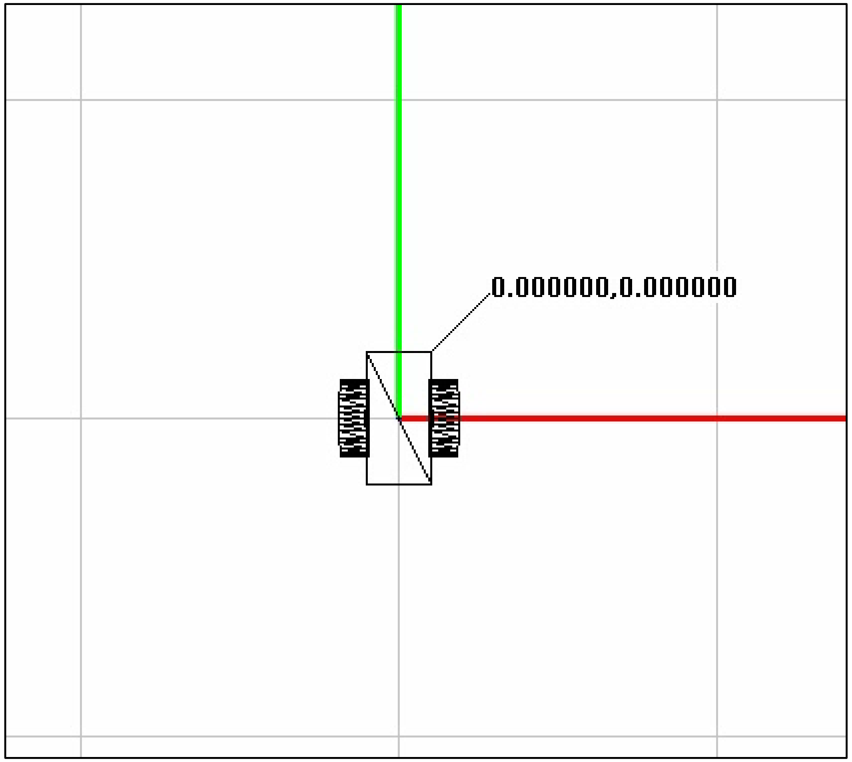
BOOL uObj::Click(point)

```
BOOL uCar::Click(CPoint pt)
{
    // reset color
    box.color          = RGB(0,0,0);
    wheel[0].color     = RGB(0,0,0);
    wheel[1].color     = RGB(0,0,0);

    // find click
    if (box.Click(pt))
    {
        box.color      = RGB(255,0,0);
        return TRUE;
    }
    if (wheel[0].Click(pt))
    {
        wheel[0].color = RGB(255,0,0);
        return TRUE;
    }
    if (wheel[1].Click(pt))
    {
        wheel[1].color = RGB(255,0,0);
        return TRUE;
    }

    return FALSE;
}
```

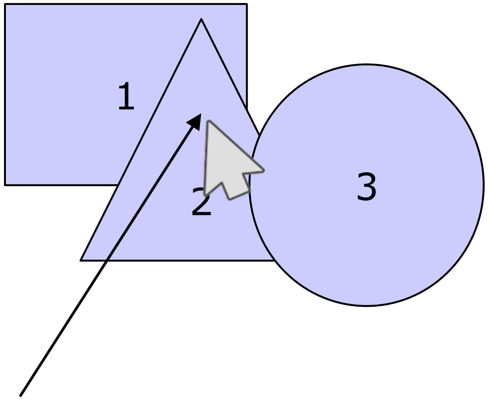
```
BOOL uObj::Click(CPoint pt)
{
    for (int i=0;i<nPoly;i++)
    if (pPoly[i].Click(pTemp,pt))    return TRUE;
    return FALSE;
}
```

- Car has
  - box, wheel[2] object
- Strategy
  - If box is clicked, return it
  - If wheel[0] is clicked, return it
  - If wheel[1] is clicked, return it.

Problem Occurs: Depth Z order is required…



- What is Z order?
  - Think next three objects



If box(1) is clicked, return box
If triangle(2) is clicked, return triangle
If circle(3) is clicked return circle.

Thus, box is clicked..

- Z ordering is required.

2

3D Object Picking by a Click in 2D

Z-Ordering

1. Z value is considered for Picking.

- We need Z-value
- Modify uCam::Projection

```
uVector uCam::Projection
{
    // Camera Framework
    t   = R*t;
    t   = T*t;

    // Projection
    t   = P*t;
    t   = t*(1./t.z);
    t   = S*t;
    return t;
}
```

EX) t = (1,2,3)
t= t/t.z
→ (1/3, 2/3, **1**)
Z is always One.

```
uVector uCam::Projection(
{
    // Camera Framework
    t   = R*t;
    t   = T*t;

    // Projection
    t   = P*t;
    t.x = t.x/t.z;
    t.y = t.y/t.z;
    t   = S*t;
    return t;
}
```

EX) t = (1,2,3)
t.x= t.x/t.z
t.y= t.y/t.z
→ t= (1/3, 2/3, **3**)

After Projection, uVector(x_{2d}, y_{2d}, z_{depth})

uVector 3D

(x,y,z)

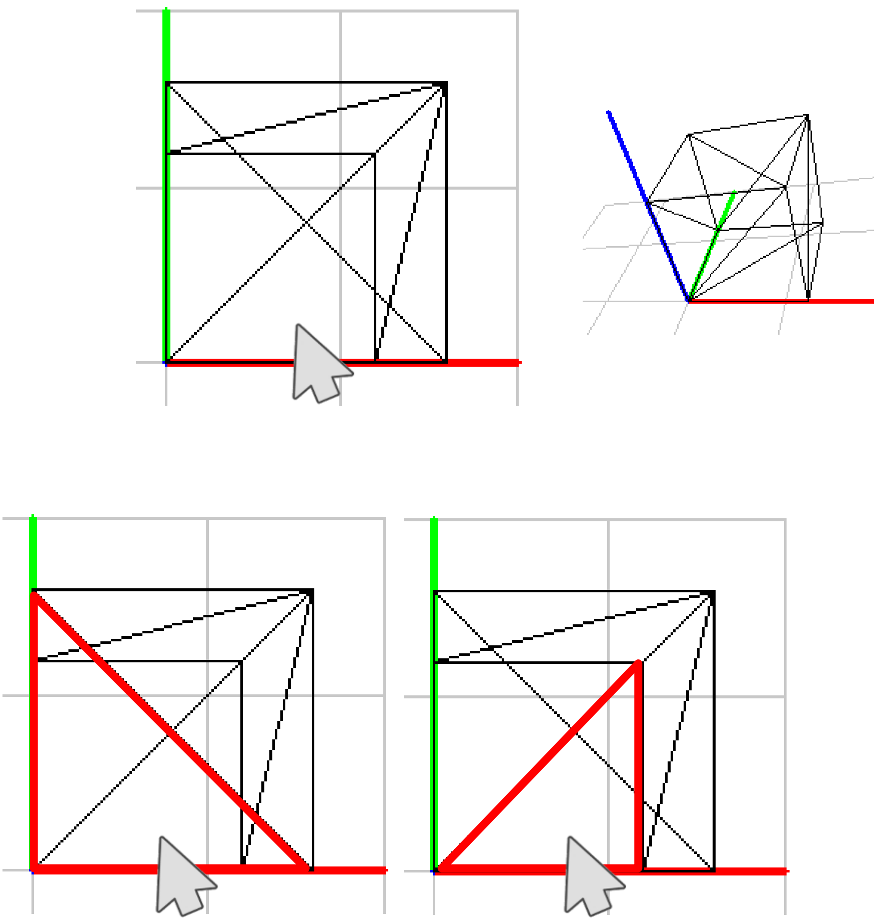
```
uVector uCam::Projection(
{
    // Camera Framework
    t   = R*t;
    t   = T*t;

    // Projection
    t   = P*t;
    t.x = t.x/t.z;
    t.y = t.y/t.z;
    t   = S*t;
    return t;
}
```

uVector 2D

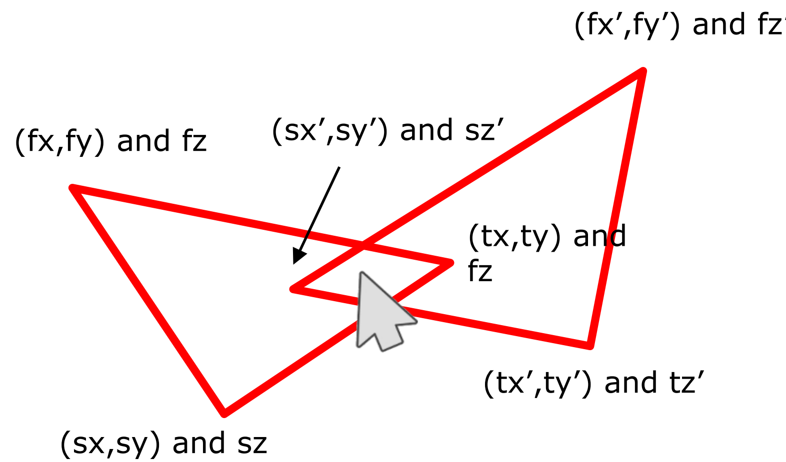
(x,y) in 2D +

Z for Depth

Multiple Polygons are Clicked



- Two triangles are clicked.



- Storing Each vector into a NEW buffer

- ○

Name	Value	Type
pTemp,8	0x00000012b7c43750 {{x=0.000000000 y=0.000000000 z=3840.10742 }, {x=53.33184...	uVector[8]
[0]	{x=0.000000000 y=0.000000000 z=3840.10742 }	uVector
[1]	{x=53.3318405 y=0.000000000 z=3840.10742 }	uVector
[2]	{x=63.9982414 y=0.000000000 z=3200.08789 }	uVector
[3]	{x=0.000000000 y=0.000000000 z=3200.08789 }	uVector
[4]	{x=0.000000000 y=53.3318405 z=3840.10742 }	uVector
[5]	{x=53.3318405 y=53.3318405 z=3840.10742 }	uVector
[6]	{x=63.9982414 y=63.9982414 z=3200.08789 }	uVector
[7]	{x=0.000000000 y=63.9982414 z=3200.08789 }	uVector

2. Create Variable(Dynamic) Buffer for Storing Vectors

- vArray.h : template library (compatible CArray in MFC)
  - Dynamic buffer: Buffer size is changed dynamically.

- What is a Template?
  - vArray<int,int> is same with int []
  - vArray<uVector,uVector> is same with uVector []

- C++ provides Template
  - Sometimes, template makes a programming to be complex.
  - But, variable buffer is good for dynamic programming.

Example) Basics of vArray

uWnd-55-OP-vArray

- vArray with int array

```
int i;
vArray<int,int> buf;

for (i=0;i<10;i++)
buf.Add(i);

int max = buf.GetSize();
int n;
for (i=0;i<max;i++)
{
    n = buf[i];
}

buf.RemoveAll();
```

- vArray with uVector array

```
int i;
vArray<uVector,uVector> buf;

for (i=0;i<10;i++)
{
    uVector tmp(i,0,0);
    buf.Add(tmp);
}

int max = buf.GetSize();
uVector t;
for (i=0;i<max;i++)
    t = buf[i];

buf.RemoveAll();
```

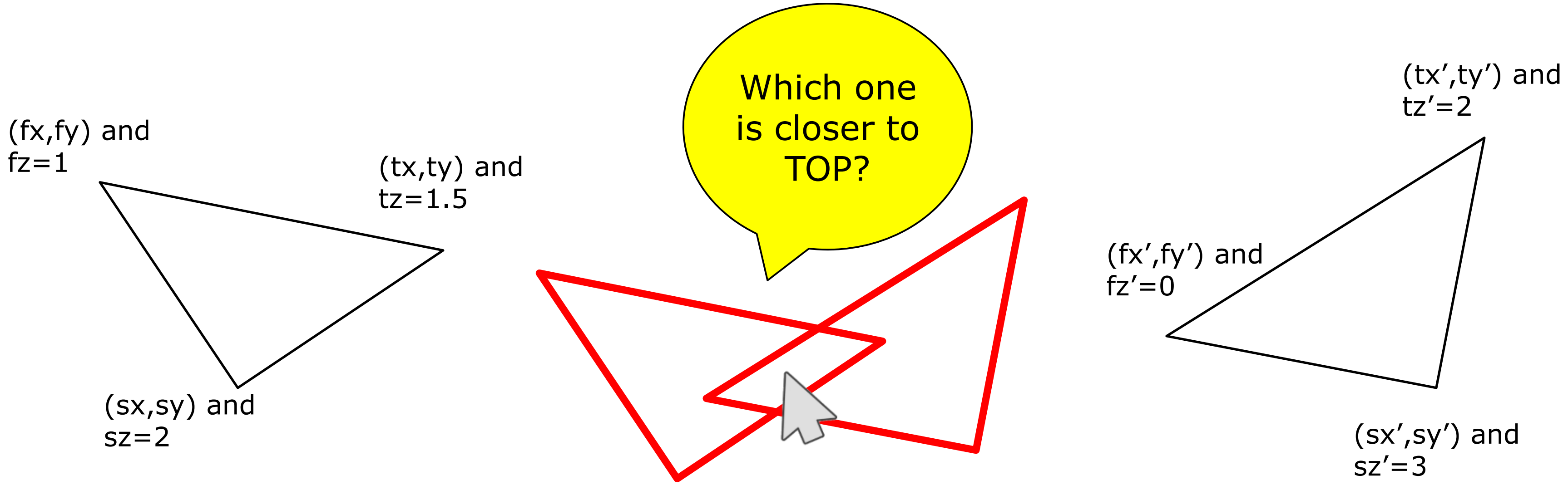
Add new Element
int, uVector, etc

Get Maximum size

Use vArray
like an array

Destroy vArray

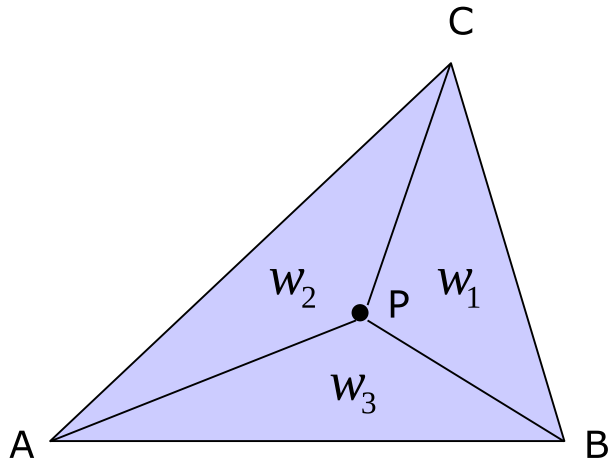
3. Baricentric Interpolation



- The Most Important method for Picking Problem.
- **Without Baricentric Interpolation**,

  We cannot find **which one is top or not**

Baricentric Interpolation



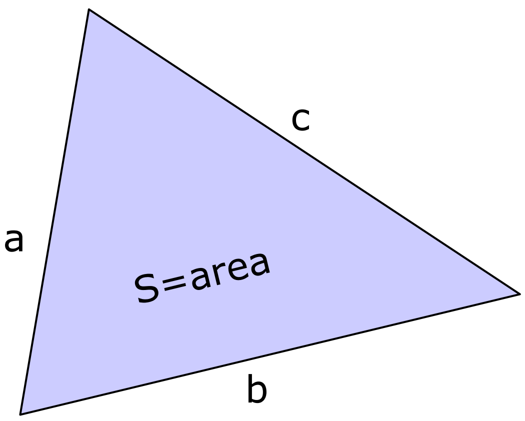
$$S = \Delta PBC + \Delta PCA + \Delta PAB = \Delta ABC$$

$$w_1 = \frac{\Delta PBC}{\Delta ABC}, w_2 = \frac{\Delta PCA}{\Delta ABC}, w_3 = \frac{\Delta PAB}{\Delta ABC} \qquad w_1 + w_2 + w_3 = 1$$

$$\therefore P = w_1 A + w_2 B + w_3 C$$

- Weight, w is calculated by Heron's formula

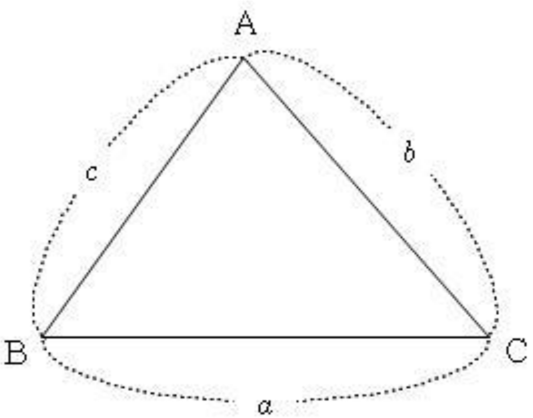
Heron’s Formula for Calculating a Triangle Area



Heron's formula

$$s = \frac{a+b+c}{2}$$

$$S \triangleq \sqrt{s(s-a)(s-b)(s-c)}$$



$$\triangle ABC = \sqrt{p(p-a)(p-b)(p-c)}$$

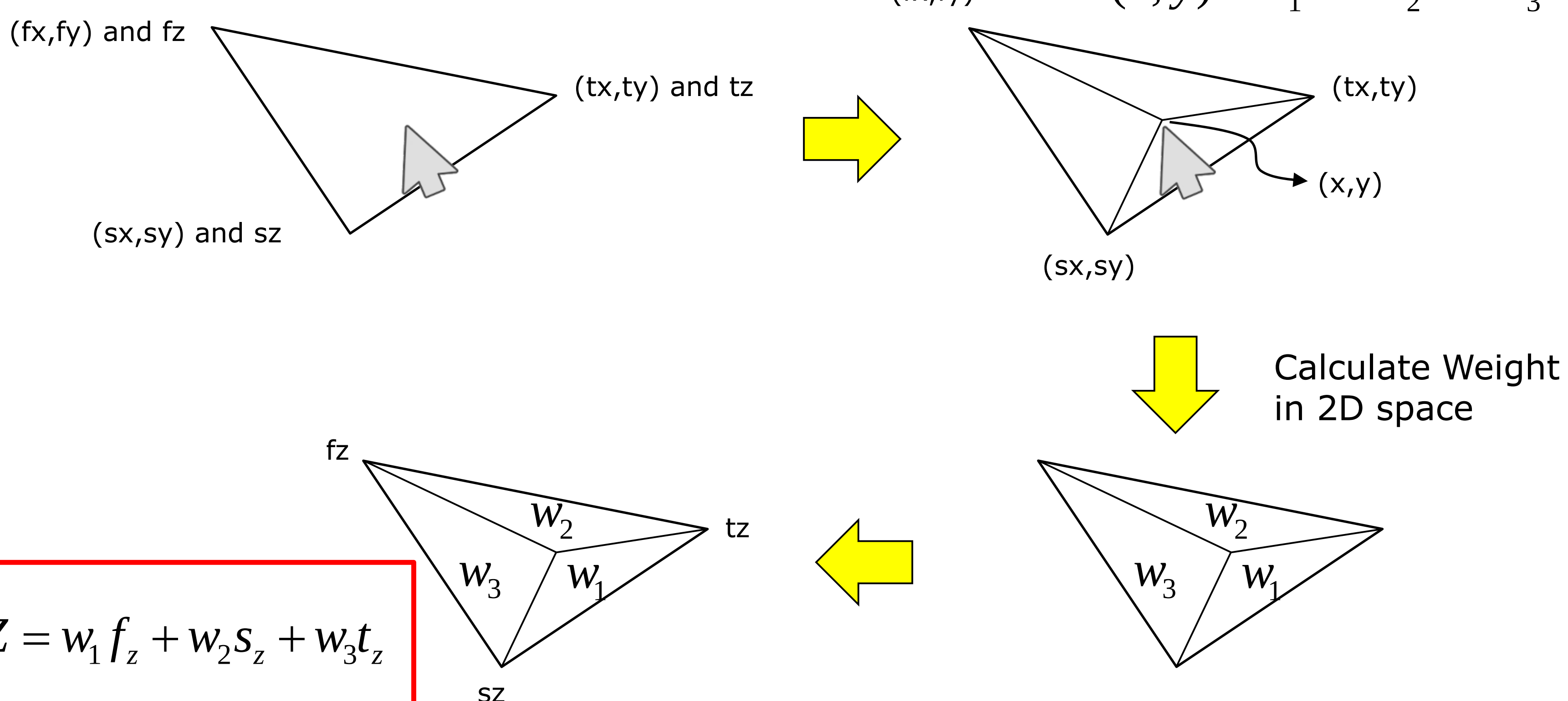
단, $p = \frac{a+b+c}{2}$

You Learned this at High school

Baricentric Interpolation for Z ordering

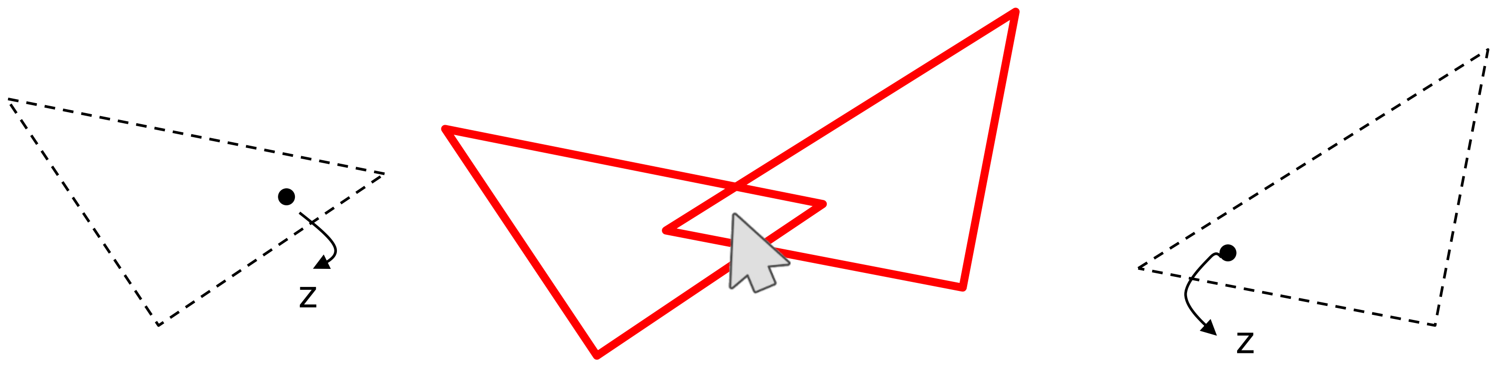
1. Do Baricentric interpolation with (x,y) in **2D**
2. Calculate Z with 2D weights

$$(x, y) = w_1 A + w_2 B + w_3 C$$



$$\therefore Z = w_1 f_z + w_2 s_z + w_3 t_z$$

4. Sort Z values for Clicked Polygons



- Which Z value is Smaller?
  - The farther object is in z direction, the larger Z value is.

- We need sorting.
  - Fast Sorting Method → Quick sort → qsort function

qsort in C programming

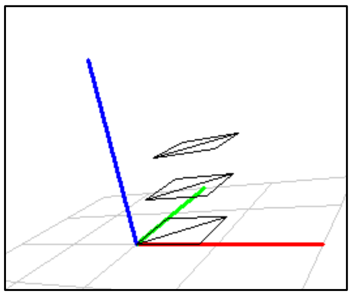
- qsort(buffer, number, size, function)

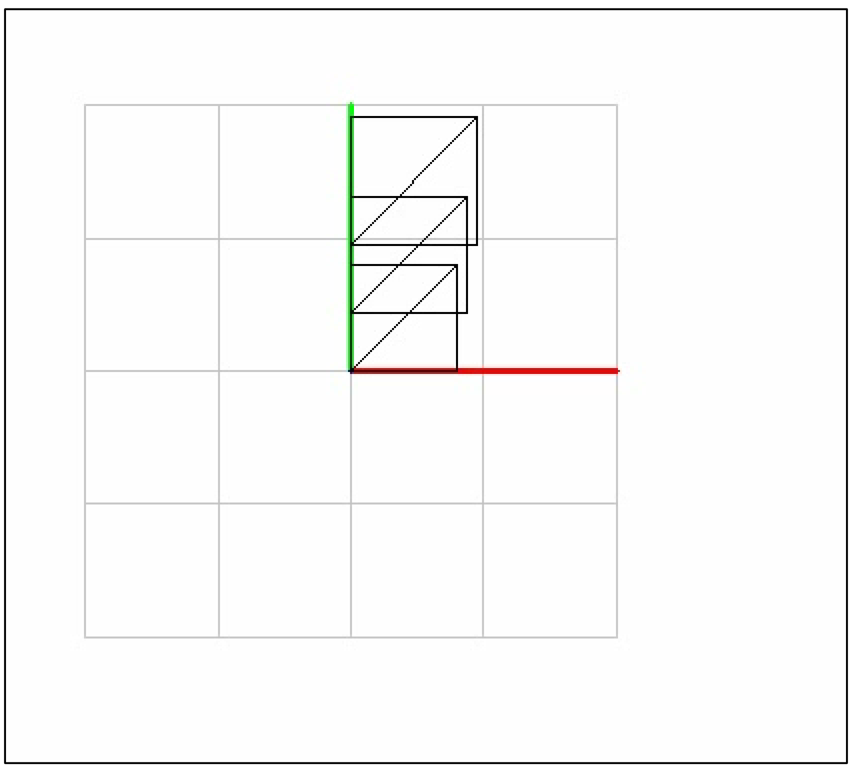
```c
int a[]={1,5,3,10,8,9};
qsort(a,6,sizeof(int),ucomp);
```

```c
int ucomp(const void *pf,const void *ps)
{
    int *f = (int*)pf;
    int *s = (int*)ps;

    if (*f<*s)  return -1;
    if (*f>*s)  return 1;
    return 0;
}
```

→a={1,3,5,8,9,10}

Ex) uWnd-55-OP-Stair Pick Object



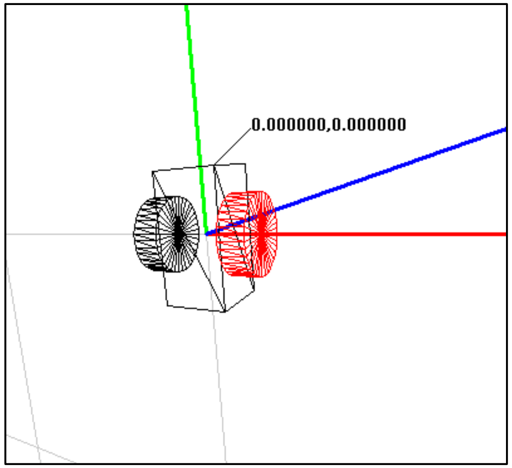


- Pick Object(clicked pt)

1. 2D projection
2. Search all triangles in which the clicked position is in or not.
3. Do Baricentric Interpolation for clicked Polygons
4. Sort Z value of Polygons.
5. Return the top polygon.

Back to Multiple Object
How to Pick an object?

- **Ex) uWnd-56-Car-Pick(only exe)**



- There are three objects.
- Click(box) → find z
- Click(wheel[0]) → find z
- Click(wheel[1]) → find z

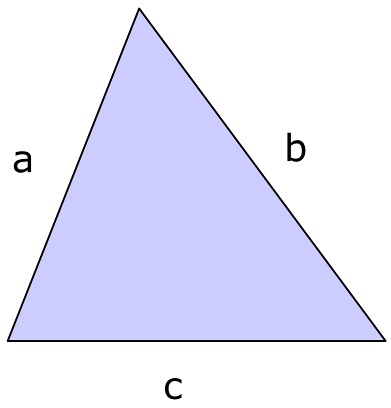
- Find the minimum z from three objects. → H.W.
- Check projection carefully
  - 1. modify projection at uCam.

```
// Projection
t    = P*t;
t.x /=t.z;
t.y /=t.z;
t    = S*t;
```

3 Mathematics about Triangle Polygon

Mathematics of Triangle

- 1. Euclidean Distance

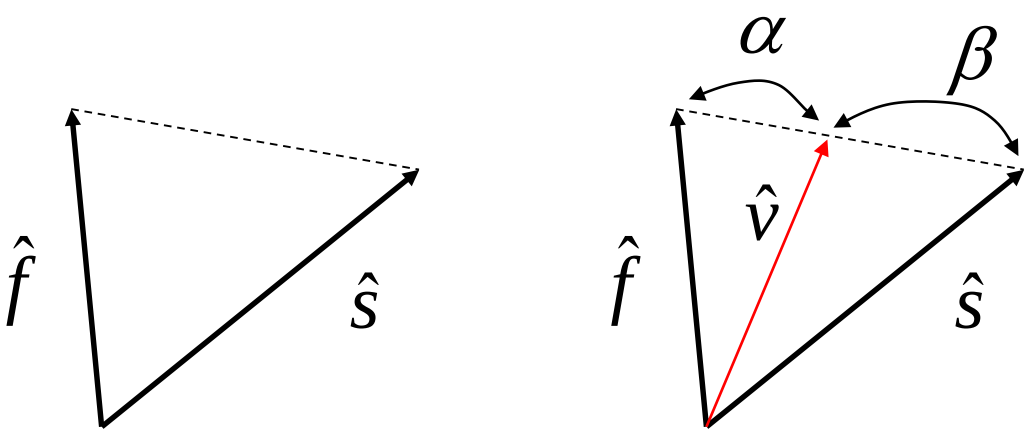


$$a < b + c$$
$$b < a + c$$
$$c < a + b$$

- What you feel in an Actual Environment is an Euclidean Space
- What you see **through your eyes** is Not an Euclidean Space.
- Definition of Euclidean Distance

$$\hat{v} = (x, y, z) \in \mathrm{R}^3 \qquad \|\hat{v}\|^2 = \sqrt{x^2 + y^2 + z^2}$$

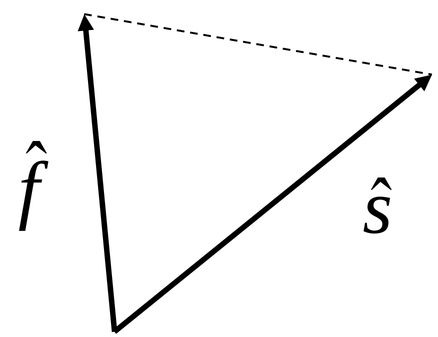
2. Definition of Triangle Polygon by Linear Interpolation

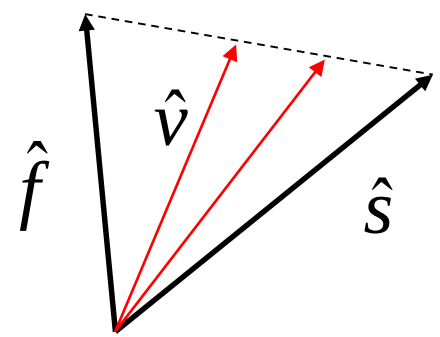


$$\therefore \hat{v} = \frac{\beta \hat{f} + \alpha \hat{s}}{\alpha + \beta}$$

- When $\alpha + \beta = 1$, it comes to be a Triangle Polygon.
- When $0 \le \alpha, \beta \le 1, \alpha + \beta \le 1,$ a vector v is inside polygon.

2. Polygon Definition in Graphics

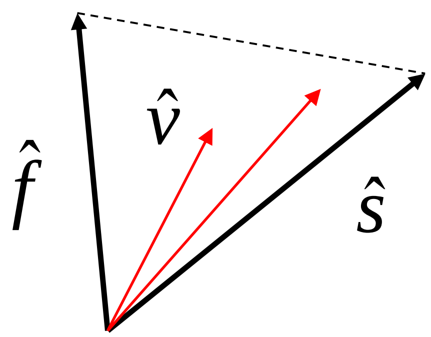




$$\therefore \hat{v} = \lambda \hat{f} + (1-\lambda)\hat{s}$$
$$0 \le \lambda \le 1$$

$if \;\; \lambda = 0:$ $\quad \therefore \hat{v} = \hat{s}$

$if \;\; \lambda = 1:$ $\quad \therefore \hat{v} = \hat{f}$



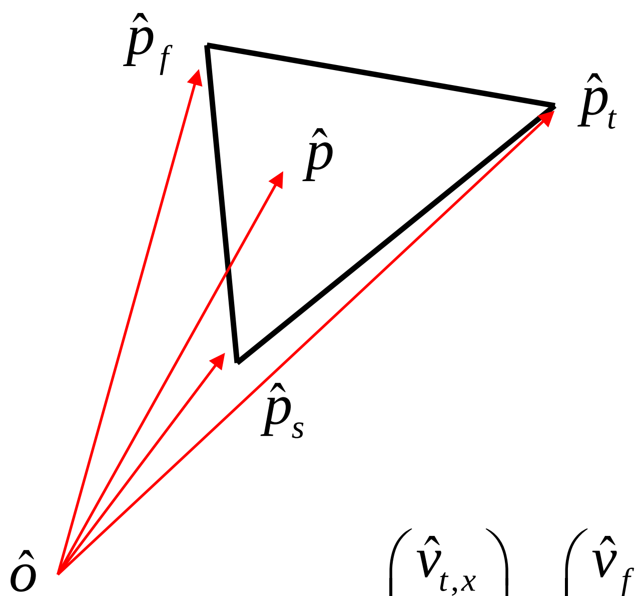
- Area of Polygon

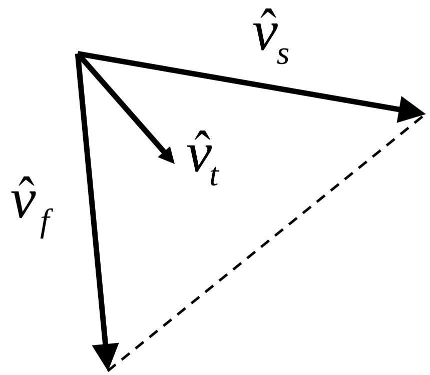
$$\hat{v} = \lambda\alpha \hat{f} + (1-\lambda)\beta\hat{s} \qquad 0 \le \lambda, \alpha, \beta \le 1$$

$$\rightarrow \hat{v} = \lambda_1 \hat{f} + \lambda_2 \hat{s} \qquad 0 \le \lambda_1, \lambda_2 \le 1 \text{ and } \lambda_1 + \lambda_2 \le 1$$

Alternative Method: Point in a Triangle or Not

- Instead of using Cross Product in pp. 7.





$$\hat{v}_f = \hat{p}_s - \hat{p}_f$$
$$\hat{v}_s = \hat{p}_t - \hat{p}_f$$
$$\hat{v}_t = \hat{p} - \hat{p}_f$$
$$= \lambda_1 \hat{v}_f + \lambda_2 \hat{v}_s$$

How to find λ_1, λ_2?

$$\begin{pmatrix} \hat{v}_{t,x} \\ \hat{v}_{t,y} \end{pmatrix} = \begin{pmatrix} \hat{v}_{f,x} & \hat{v}_{s,x} \\ \hat{v}_{f,y} & \hat{v}_{s,y} \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix}$$

$$\therefore \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} \hat{v}_{f,x} & \hat{v}_{s,x} \\ \hat{v}_{f,y} & \hat{v}_{s,y} \end{pmatrix}^{-1} \begin{pmatrix} \hat{v}_{t,x} \\ \hat{v}_{t,y} \end{pmatrix}$$

Check if

$$0 \le \lambda_1, \lambda_2 \le 1 \ \ and \ \ \lambda_1 + \lambda_2 \le 1$$

Ex)uWnd-53-OP-Pick2

```
BOOL uPolygon::Click(uVector *pTemp, CPoint pt)
{
    uVector vf,vs,vt,p;
    p   = uVector(pt.x,pt.y,0);
    vf  = pTemp[s]-pTemp[f];
    vs  = pTemp[t]-pTemp[f];
    vt  = p-pTemp[f];

    // vt = l1*vf + l2*vs;
    // vt.x = [ vf.x vs.x] [l1]
    // vt.y = [ vf.y vs.y] [l2]
    //
    // l1 = 1/det* [ vs.y -vs.x] [vt.x]
    // l2 = 1/det* [-vf.y  vf.x] [vt.y]

    float l1,l2;
    float det = vf.x*vs.y-vs.x*vf.y;
    if (fabs(det)<1e-5) return FALSE;

    l1  = (vs.y*vt.x-vs.x*vt.y)/det;
    l2  = (-vf.y*vt.x+vf.x*vt.y)/det;

    if (l1<0)    return FALSE;
    if (l2<0)    return FALSE;
    if (l1>1)    return FALSE;
    if (l2>1)    return FALSE;
    if (l1+l2>1) return FALSE;

    return TRUE;
}
```

$$\hat{v}_f = \hat{p}_s - \hat{p}_f$$

$$\hat{v}_s = \hat{p}_t - \hat{p}_f$$

$$\hat{v}_t = \hat{p} - \hat{p}_f = \lambda_1 \hat{v}_f + \lambda_2 \hat{v}_s$$

$$\begin{pmatrix}\lambda_1 \\ \lambda_2\end{pmatrix} = \begin{pmatrix}\hat{v}_{f,x} & \hat{v}_{s,x} \\ \hat{v}_{f,y} & \hat{v}_{s,y}\end{pmatrix}^{-1} \begin{pmatrix}\hat{v}_{t,x} \\ \hat{v}_{t,y}\end{pmatrix}$$

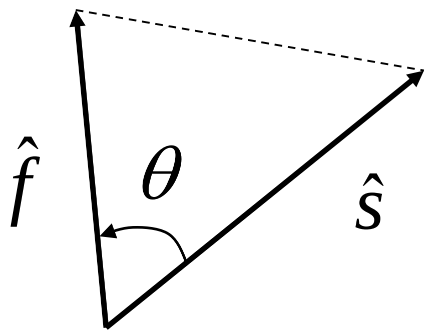
$$= \frac{1}{\hat{v}_{f,x}\hat{v}_{s,y} - \hat{v}_{s,x}\hat{v}_{f,y}} \begin{pmatrix}\hat{v}_{s,y} & -\hat{v}_{s,x} \\ -\hat{v}_{f,y} & \hat{v}_{f,x}\end{pmatrix} \begin{pmatrix}\hat{v}_{t,x} \\ \hat{v}_{t,y}\end{pmatrix}$$

If Determinant is zero, then $\hat{v}_f \parallel \hat{v}_s$.

Check if

$$0 \le \lambda_1, \lambda_2 \le 1 \;\; and \;\; \lambda_1 + \lambda_2 \le 1$$

3. Polygon Area



1.*Triangle* Area by Vector Calculus

$$S = \frac{1}{2} | \hat{f} | \cdot | \hat{s} | \sin\theta$$

$$= \frac{1}{2} | \hat{s} \times \hat{f} | = \frac{1}{2} | \hat{f} \times \hat{s} |$$

2.*Heron's* formula

$$s = \frac{a+b+c}{2}$$

$$S \triangleq \sqrt{s(s-a)(s-b)(s-c)}$$

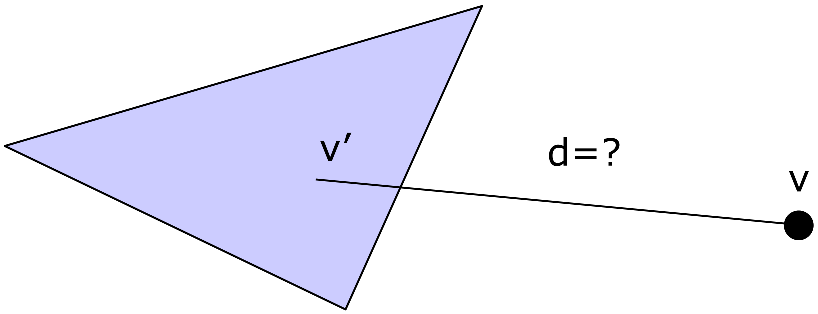
3.*Triangle Area by Vector Calculus*

$$S = \frac{1}{2} | \hat{f} | \cdot | \hat{s} | \sin\theta = \frac{1}{2} | \hat{f} | \cdot | \hat{s} | \sqrt{1 - \cos^2\theta}$$

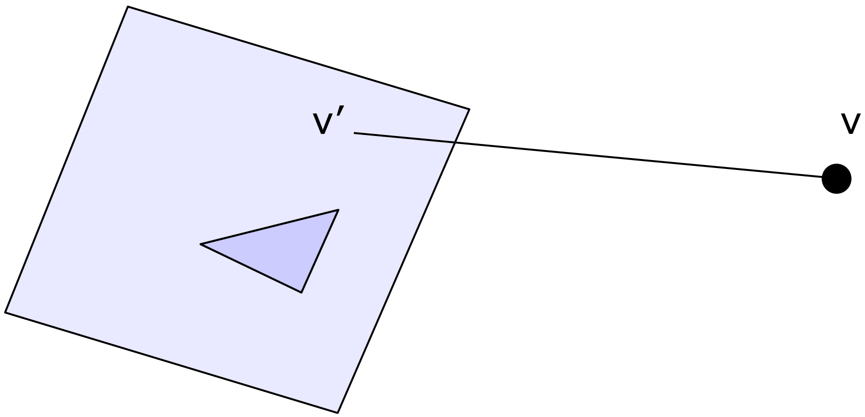
$$= \frac{1}{2} \sqrt{| \hat{f} |^2 | \hat{s} |^2 - \hat{f} \bullet \hat{s}}$$

4. Distance to Triangle Polygon

- It is similar to the Distance to Plane but It is NOT Easy

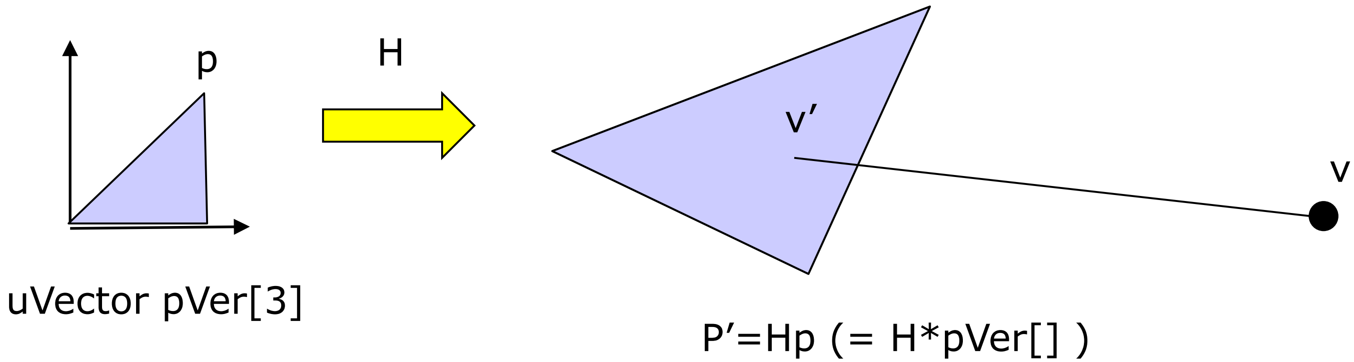


Projection Position, v′ is
IN the triangle



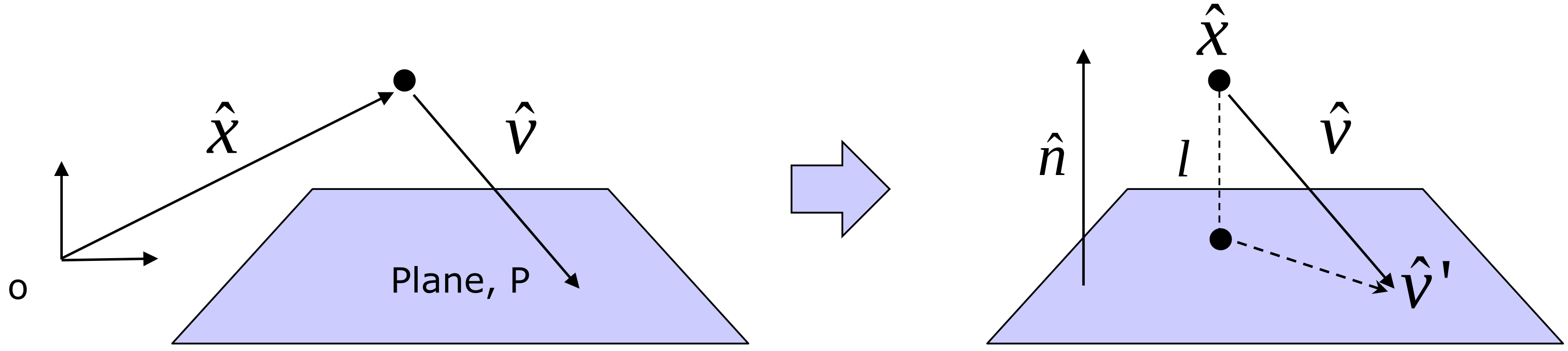
Projection Position, v′ is
Out of the triangle

Transform must be Applied



- Remind Vertices should be Transformed, H
- Get Distance from a Polygon
  - **uses Three vertices, which is transformed by H.**

Distance to Plane



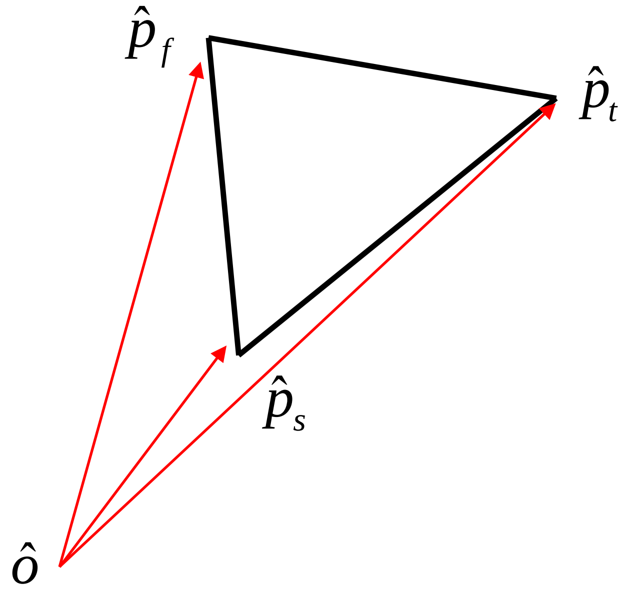
- Vector $\hat{v}$ from vector $\hat{x}$ is passing through the plane, p.
- Distance to Plane, p is simply defined,

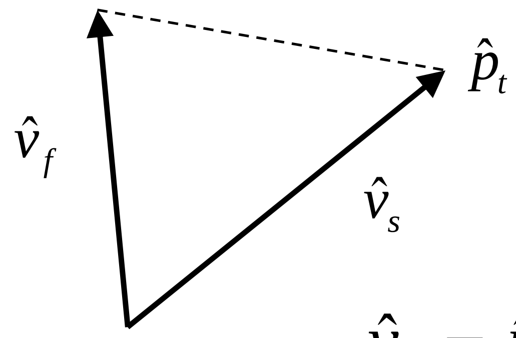
$$ax + by + cz + d = 0 \quad \left(\hat{n} \bullet \hat{X} + d = 0\right)$$

$$\hat{x} = (x_x, x_y, x_z)$$

$$\therefore l = \frac{|ax_x + bx_y + cx_z + d|}{\sqrt{a^2 + b^2 + c^2}}$$

Get Normal of Polygon



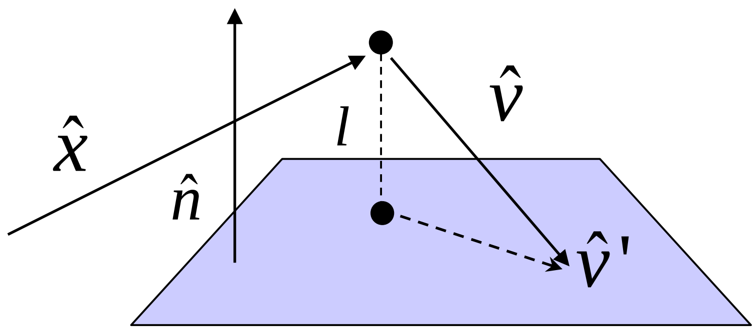


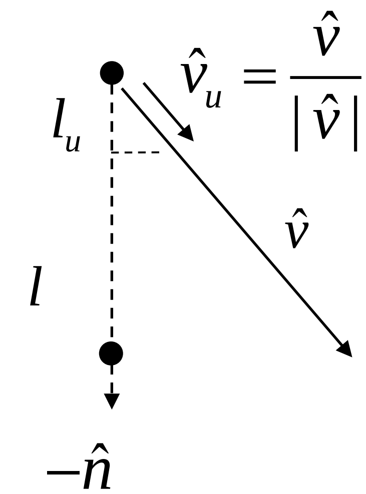
$$\hat{v}_f = \hat{p}_f - \hat{p}_s$$
$$\hat{v}_s = \hat{p}_t - \hat{p}_s$$
$$\therefore \hat{n} = \hat{v}_s \times \hat{v}_f$$

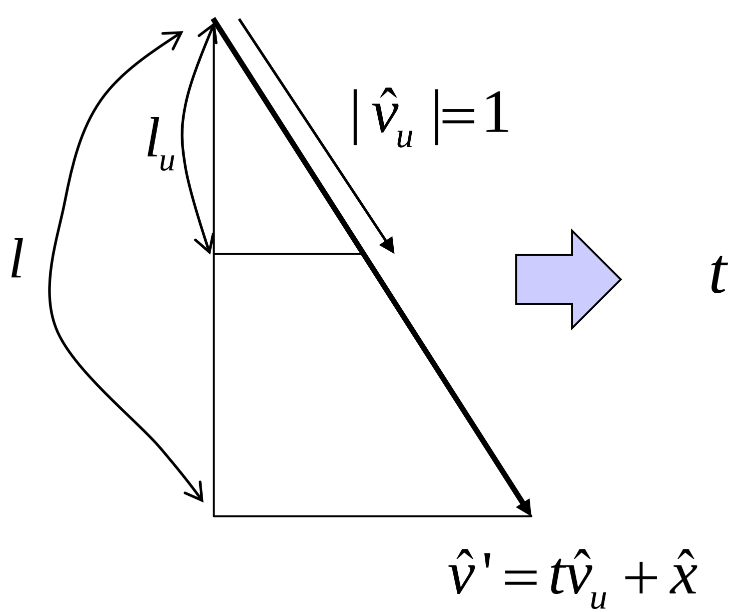
- Cross product is Helpful to get Normal vector, $\hat{n}$
- Then, Plane, P is derived as,

$$Plane, P: \quad \hat{n} \circ \hat{X} + d = 0$$
$$\therefore d = -\hat{n} \circ \hat{p}_s$$

Case 1: Get $\hat{v}'$ by Dot Product



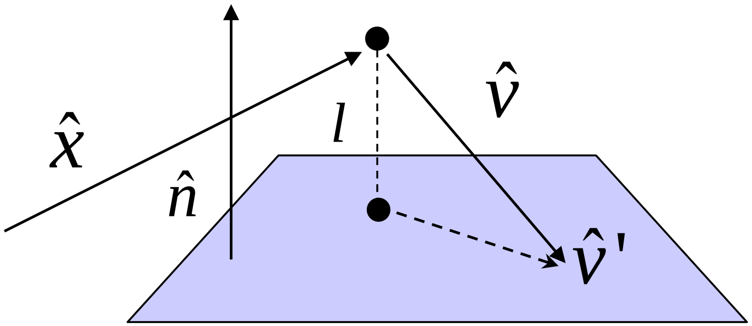




$$t = \frac{l}{l_u} = \frac{l}{-\hat{n} \bullet \hat{v}_u}$$

$$\therefore \hat{v}' = t\hat{v}_u + \hat{x} = \frac{l\hat{v}_u}{-\hat{n} \bullet \hat{v}_u} + \hat{x}$$

Case 2: Get $\hat{v}'$ by Solving Eq.



$$Plane, P: \quad \hat{n} \circ \hat{X} + d = 0$$

$$\hat{n} \circ \hat{v}' + d = 0$$

$$\hat{n} \circ (t\hat{v}_u + \hat{x}) + d = 0$$

$$t\hat{n} \circ \hat{v}_u = -d - \hat{n} \circ \hat{x}$$

$$\therefore t = \frac{-d - \hat{n} \circ \hat{x}}{\hat{n} \circ \hat{v}_u}$$

$$\therefore \hat{v}' = t\hat{v}_u + \hat{x}$$

3. Check If 3Dim. Point is Inside a Polygon or Not uPolygon:IsIn(vertices, vector)

```
BOOL uPolygon::IsIn(uVector *p, uVector o)
{
    uVector n;
    n    = (p[2]-p[1])*(p[0]-p[1]);
    n    = n.Unit();

    uVector vf,vs,vt,t;
    vf   = p[0]-o;
    vs   = p[1]-o;
    vt   = p[2]-o;

    float sgn,sgn2,sgn3;

    // Check if sign chagnes or Not
    // first
    t    = vf*vs;
    if (n.Dot(t)>=0)      sgn=1;
    else                  sgn=-1;

    // second
    t    = vs*vt;
    if (n.Dot(t)>=0)      sgn2=1;
    else                  sgn2=-1;
    if (sgn*sgn2<0) return FALSE;

    // third
    t    = vt*vf;
    if (n.Dot(t)>=0)      sgn3=1;
    else                  sgn3=-1;
    if (sgn*sgn3<0) return FALSE;

    return TRUE;
}
```

- It is very similar to PP. 7
- But it is 3Dim. Point and vertices.

- Normal vector check is added.

$$sign1 = \hat{n} \bullet (\hat{f} \times \hat{s})$$

$$sign2 = \hat{n} \bullet (\hat{s} \times \hat{t})$$

$$sign3 = \hat{n} \bullet (\hat{t} \times \hat{f})$$

$$\therefore sign1 * \mathrm{sign}\,2 > 0$$

$$\therefore sign1 * \mathrm{sign}\,3 > 0$$

Distance to a Triangular Polygon

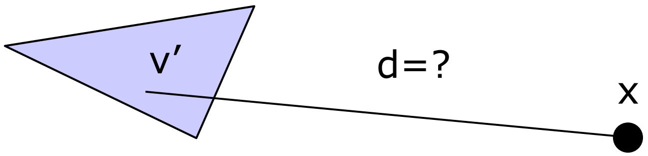
- 1. Get normal vector and d for Plane, P

$$\hat{n} \circ \hat{X} + d = 0$$

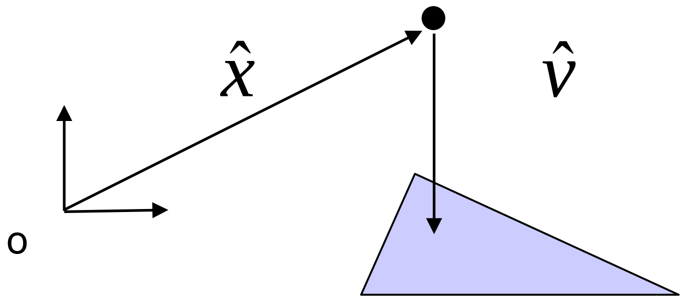
- 2. Get $\hat{v}'$ that is a vector for passing plane P,

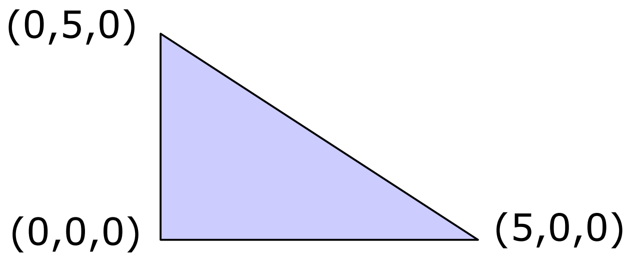
$$\therefore \hat{v}' = t\hat{v}_u + \hat{x}$$

- 3. Check if $\hat{v}'$ is inside of Polygon, then $d = \| \hat{v}' - \hat{x} \|^2$
  - Otherwise d = infinite.



Example of Polygon Distance





$$\hat{x} = (2,2,2)$$
$$\hat{v} = (0,0,-2)$$

1. Normal

$$\hat{f} = (0,5,0) - (0,0,0) = (0,5,0)$$
$$\hat{s} = (5,0,0) - (0,0,0) = (5,0,0)$$
$$\therefore \hat{n} = \frac{\hat{s} \times \hat{f}}{|\hat{s} \times \hat{f}|} = (0,0,1)$$

Get Plane Equation

$$\hat{n} \circ \hat{X} + d = 0$$
$$(0,0,1)\hat{O} + d = 0$$
$$\therefore d = 0$$
$$\therefore Plane\ P: \quad z = 0$$

2. *case* 1

$l: Distance\ from\ x\ to\ plane,\ ax+by+cz+d=0\ \ \left(\hat{n}\bullet\hat{X}+d=0\right)$

$\hat{x}=(2,2,2)$

$$\therefore l=\frac{|a\cdot 2+b\cdot 2+c\cdot 2+d|}{\sqrt{a^2+b^2+c^2}}=\frac{|0+0+2+0|}{\sqrt{1}}=2$$

$$t=\frac{l}{l_u}=\frac{l}{-\hat{n}\bullet\hat{v}_u}$$

$$t=\frac{l}{l_u}=\frac{2}{-\hat{n}\circ\hat{v}_u}=\frac{2}{(0,0,-1)\circ(0,0,-1)}=2$$

$$\therefore \hat{v}'=t\hat{v}_u+\hat{x}$$

$$\hat{v}'=t\hat{v}_u+\hat{x}=2(0,0,-1)+(2,2,2)=(2,2,0);$$

$$\hat{n}\circ\hat{v}'+d=0$$

$$\hat{n}\circ(t\hat{v}_u+\hat{x})+d=0$$

$$t\hat{n}\circ\hat{v}_u=-d-\hat{n}\circ\hat{x}$$

$$\therefore t=\frac{-d-\hat{n}\circ\hat{x}}{\hat{n}\circ\hat{v}_u}$$

$$\therefore \hat{v}'=t\hat{v}_u+\hat{x}$$

2. *case* 2

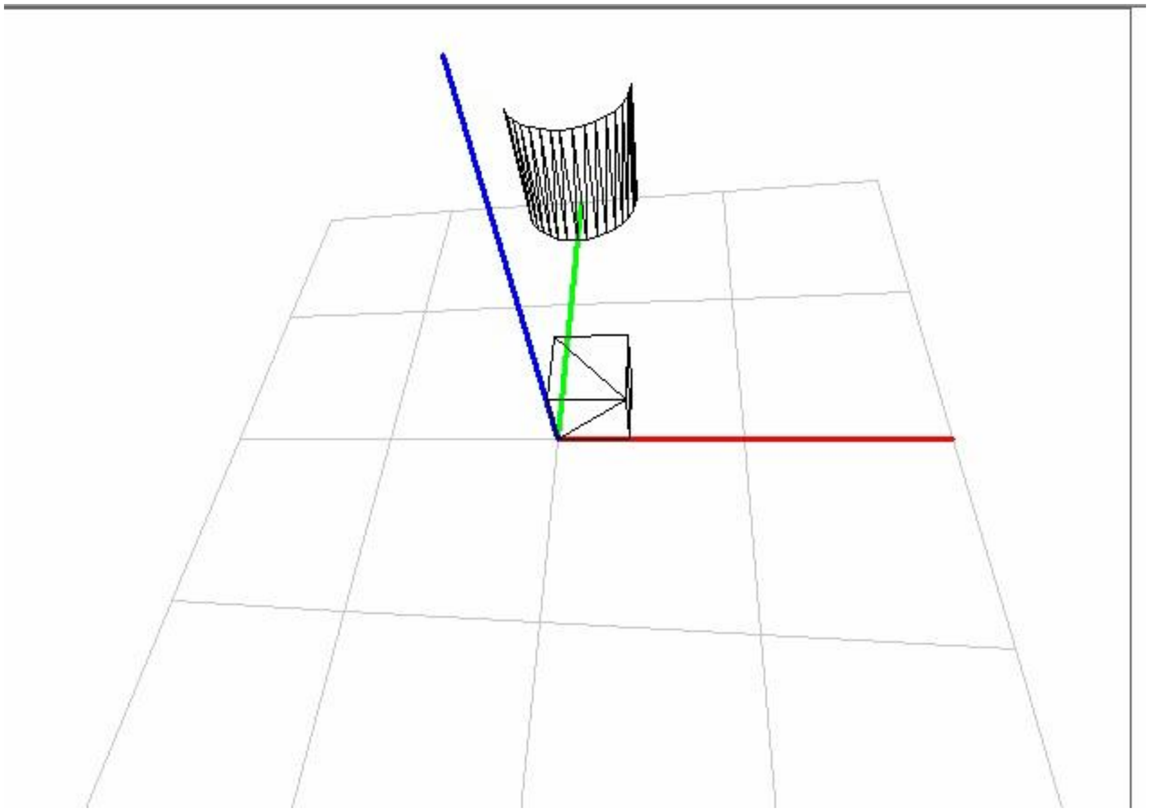
$$t=\frac{-d-\hat{n}\circ\hat{x}}{\hat{n}\circ\hat{v}_u}=\frac{-0-(0,0,1)\circ(2,2,2)}{(0,0,1)\circ(0,0,-1)}=\frac{-2}{-1}=2$$

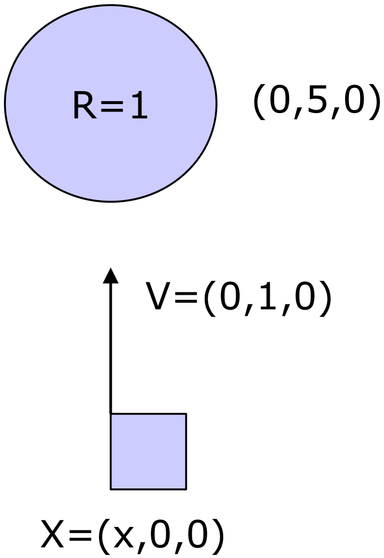
$$\hat{v}'=t\hat{v}_u+\hat{x}=2(0,0,-1)+(2,2,2)=(2,2,0)$$

3. Point Inside Check by PP.7

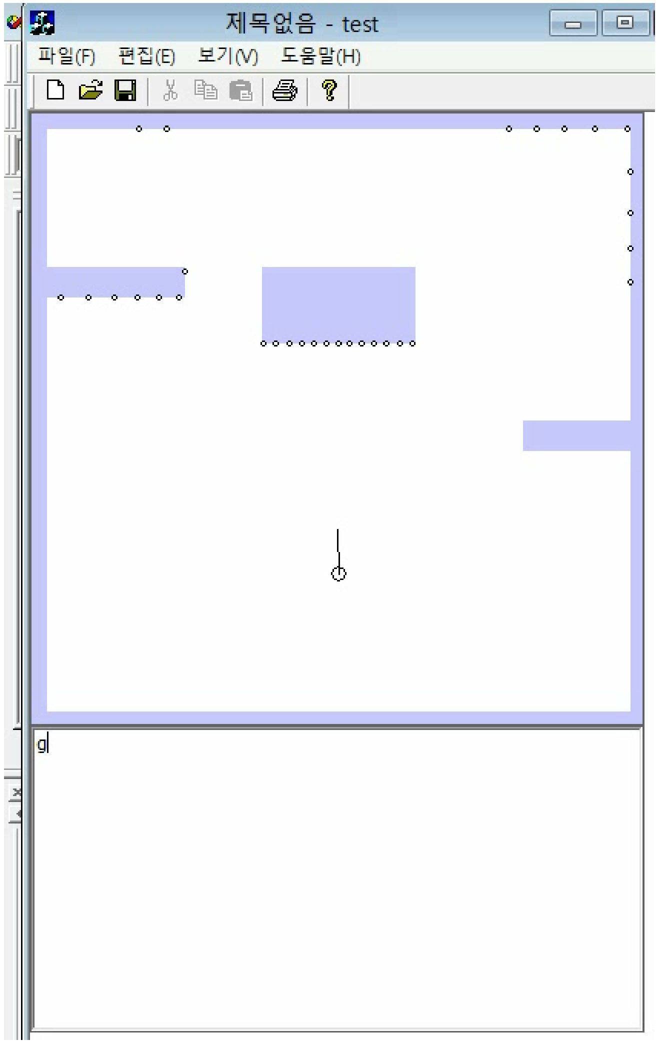
Ex. uWnd-61-OP-Distance

Find Distance for Cylinder Object





Why this Distance in 3D is so Important



- In Graphics,
  - Ray tracing uses distance.
- In Game field,
  - Collision Detection

- In Robotics,
  - Virtual sensor for simulation
  - Useful in Calibration, and so on.