

Rotate an Object about an Axis

An object is defined by points \mathbf{x} . The rotation axis is defined by a reference point \mathbf{p} and a direction vector \mathbf{n} .

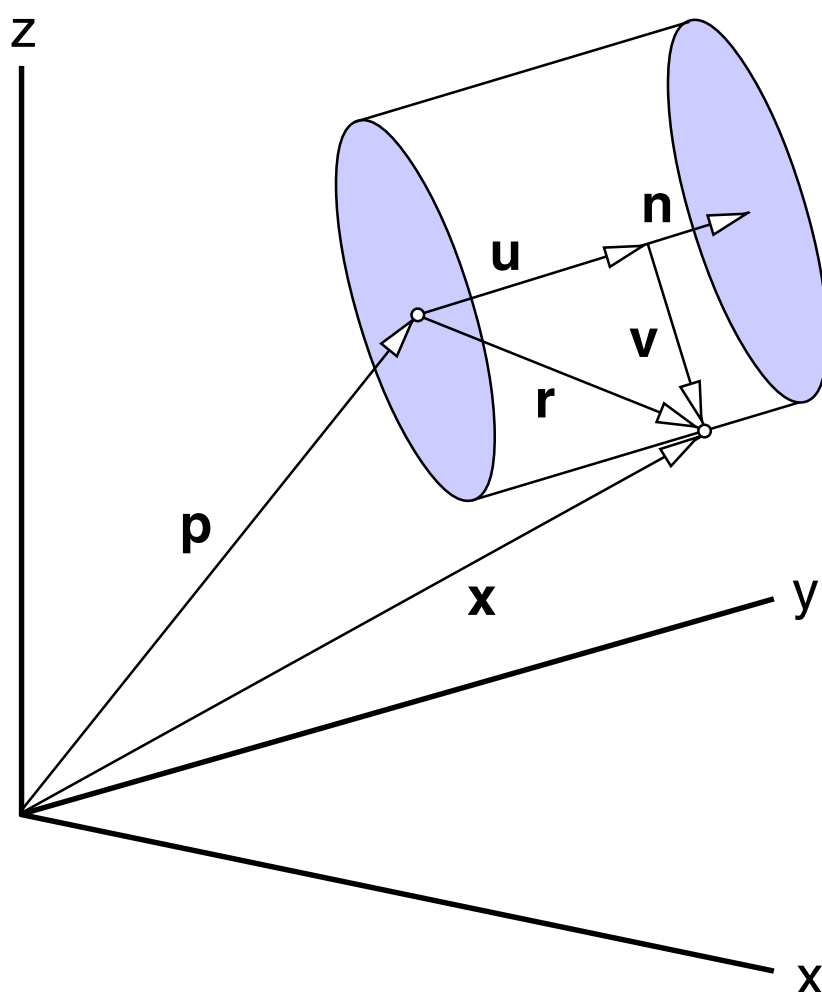
The Geometrical Construction is shown below. This algorithm can be programmed straightforward, as on page 2.

Page 3 and page 4 show two Pascal procedures, which use Rotation Matrices. These algorithms can be easily adapted to homogeneous coordinates.

\mathbf{x}_1 Object point before rotation
 \mathbf{x}_2 Object point after rotation

 \mathbf{p} Reference point of axis
 \mathbf{n} Direction vector, normalized
 η Rotation angle (right screw positive)
 \mathbf{u} Projection of \mathbf{r} on \mathbf{n}
 \mathbf{v} Orthogonal component of \mathbf{r}
 \mathbf{a}, \mathbf{b} Orthogonal vector base, each length $|\mathbf{v}|$

$\mathbf{r} = \mathbf{x}_1 - \mathbf{p}$
 $\mathbf{u} = (\mathbf{r}^T \mathbf{n}) \mathbf{n}$
 $\mathbf{v} = \mathbf{r} - \mathbf{u}$
 $\mathbf{a} = \mathbf{v}$
 $\mathbf{b} = \mathbf{n} \times \mathbf{a}$
 $\mathbf{x}_2 = \mathbf{p} + \mathbf{u} + \mathbf{a} \cos(\eta) + \mathbf{b} \sin(\eta)$



Rotation by Geometrical Construction

```
Procedure RotateA (p,n: XYZ; eta: Single; x1: XYZ; Var x2: XYZ);
{
  Gernot Hoffmann
  July 9, 2002
  Rotate object, using Geometrical Construction
  Tutorial example, not optimized
  p Reference point of axis
  n Direction of axis
  eta Rotation angle in degrees
  x1 Object point before rotation
  x2 Object point after rotation
  XYZ Record type, e.g. p.x, p.y, p.z }
Var r,u,v,a,b: XYZ;
    s,si,co : Single;
Const wrad=pi/180;
Begin
{ Normalization of normal vector }
With n Do
Begin
s:=1/Sqrt (Sqr (x)+Sqr (y)+Sqr (z));
x:=x*s; y:=y*s; z:=z*s;
End;
With x1 Do
Begin
r.x:=x-p.x; r.y:=y-p.y; r.z:=z-p.z;
End;
With n Do
Begin
s:=r.x*x+r.y*y+r.z*z;
u.x:=s*x; u.y:=s*y; u.z:=s*z;
End;
With r Do
Begin
v.x:=x-u.x; v.y:=y-u.y; v.z:=z-u.z;
End;
With v Do
Begin
a.x:=x; a.y:=y; a.z:=z;
End;
With n Do
Begin
b.x:=-a.y*z+a.z*y; b.y:=-a.z*x+a.x*z; b.z:=-a.x*y+a.y*x;
End;
SicCoc (wrad*eta,si,co);
With p Do
Begin
x2.x:=x + u.x + a.x*co+b.x*si;
x2.y:=y + u.y + a.y*co+b.y*si;
x2.z:=z + u.z + a.z*co+b.z*si;
End;
End;
```

Rotation Matrix by Quaternions

```
Procedure RotateB (p,n: XYZ; eta: Single; x1: XYZ; Var x2: XYZ);
{
  Gernot Hoffmann
  July 9, 2002
  Rotate object, using quaternions
  http://www.fho-emden.de/~hoffmann/quarter12012002.pdf
  Tutorial example, not optimized
  u = x1 - p
  x2 = C*u + p
  p Reference point of axis
  n Direction of axis
  eta Rotation angle in degrees
  x1 Object point before rotation
  x2 Object point after rotation
  XYZ Record type, e.g. p.x, p.y, p.z }
Var u : XYZ;
    s1,s2,s3,s4,si,co : Single;
    q1,q2,q3,q4 : Single;
    c11,c12,c13,c21,c22,c23,c31,c32,c33 : Single;
Const wrad=pi/180;
Begin
{ Normalize normal vector, same for all points }
With n Do
Begin
s1:=1/Sqrt (Sqr (x)+Sqr (y)+Sqr (z));
x:=x*s1; y:=y*s1; z:=z*s1;
End;
{ Half angle sine and cosine for quaternion }
SicCoc(0.5*wrad*eta,si,co);
{ Build Quaternion, same for all points }
With n Do
Begin
q1:=x*si; q2:=y*si; q3:=z*si; q4:=co;
End;
{ Build Caley Matrix, same for all points }
s1:=Sqr(q1); s2:=Sqr(q2); s3:=Sqr(q3); s4:=Sqr(q4);
c11:= s1-s2-s3+s4; c12:= 2*( q1*q2-q3*q4); c13:= 2*( q1*q3+q2*q4);
c21:= 2*( q1*q2+q3*q4); c22:= -s1+s2-s3+s4; c23:= 2*(-q1*q4+q2*q3);
c31:= 2*( q1*q3-q2*q4); c32:= 2*( q1*q4+q2*q3); c33:= -s1-s2+s3+s4;
{ Reference point translation, for each single point }
With x1 Do
Begin
u.x:=x-p.x; u.y:=y-p.y; u.z:=z-p.z;
End;
{ Rotation and re-translation, for each single point }
With u Do
Begin
x2.x:=c11*x+c12*y+c13*z + p.x;
x2.y:=c21*x+c22*y+c23*z + p.y;
x2.z:=c31*x+c32*y+c33*z + p.z;
End;
End;
```

Rotation Matrix without Quaternions

```
Procedure RotateC (p,n: XYZ; eta: Single; x1: XYZ; Var x2: XYZ);
{
  Gernot Hoffmann
  July 9, 2002
  Rotate object, straightforward rotation matrix
  Tutorial example, not optimized
  u = x1 - p
  x2 = C*u + p
  p Reference point of axis
  n Direction of axis
  eta Rotation angle in degrees
  x1 Object point before rotation
  x2 Object point after rotation
  XYZ Record type, e.g. p.x, p.y, p.z }
Var u : XYZ;
    c1,si,co : Single;
    c11,c12,c13,c21,c22,c23,c31,c32,c33 : Single;
Const wrad=pi/180;
Begin
{ Normalize normal vector, same for all points }
With n Do
Begin
c1:=1/Sqrt (Sqr (x)+Sqr (y)+Sqr (z));
x:=x*c1; y:=y*c1; z:=z*c1;
End;
{ Sine and cosine }
SicCoc(wrad*eta,si,co);
{ Build Rotation Matrix, same for all points
  J.Hoscheck,D.Lasser: Grundlagen der geometrischen Datenverarbeitung,
  B.G.Teubner Stuttgart 1992 }
With n Do
Begin
c1 := 1-co;
c11:=c1*x*x + co; c12:=c1*x*y - z*si; c13:=c1*x*z + y*si;
c21:=c1*y*x + z*si; c22:=c1*y*y + co; c23:=c1*y*z - x*si;
c31:=c1*z*x - y*si; c32:=c1*z*y + x*si; c33:=c1*z*z + co;
End;
{ Reference point translation, for each single point }
With x1 Do
Begin
u.x:=x-p.x; u.y:=y-p.y; u.z:=z-p.z;
End;
{ Rotation and re-translation }
With u Do
Begin
x2.x:=c11*x+c12*y+c13*z + p.x;
x2.y:=c21*x+c22*y+c23*z + p.y;
x2.z:=c31*x+c32*y+c33*z + p.z;
End;
End;
```