PolyProperties

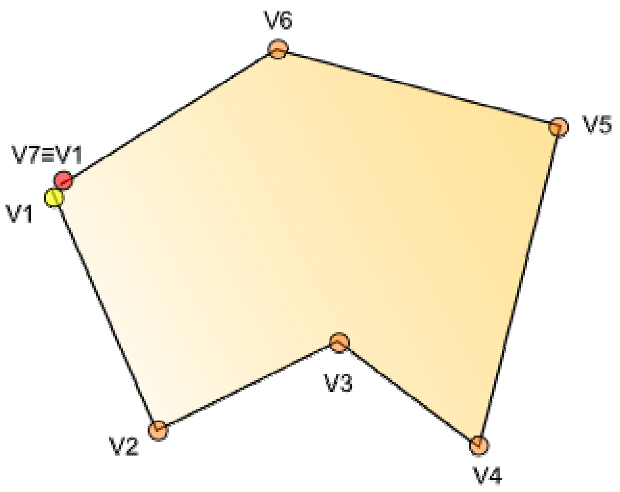
@description: calculate properties of generic polygons
@version: 2013.02.01
@author: Davide Carpi >>> davide.carpi@gmail.com
@translation: Davide Carpi
@SMath: 0.95.4594 >>> http://en.smath.info

DRAWING TIPS and RULES

A POLYGON

A polygon is a closed chain of straight lines (the polygon sides);

the points where two sides meet are the polygon's vertices.



In this SMath Studio worksheet:

- *a polygon is defined by it's vertices;*
- *definition of vertices can be either clockwise or anticlockwise;*
- *last vertex must be equal to the first;*
- *intersections points must be numered twice.*

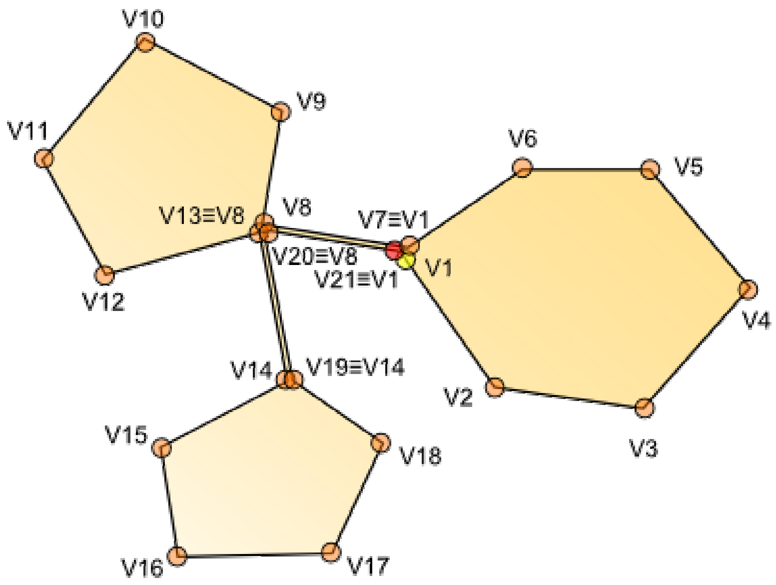
POLYGONS (discontinuous areas)

To have multiple polygons:

- draw all the vertices having same wise;

- close each polygon;

- don't forget to close the vertices chain in the first vertex.



NOTE:

- perimeter calculation cut off overlapped sides with opposite wise.

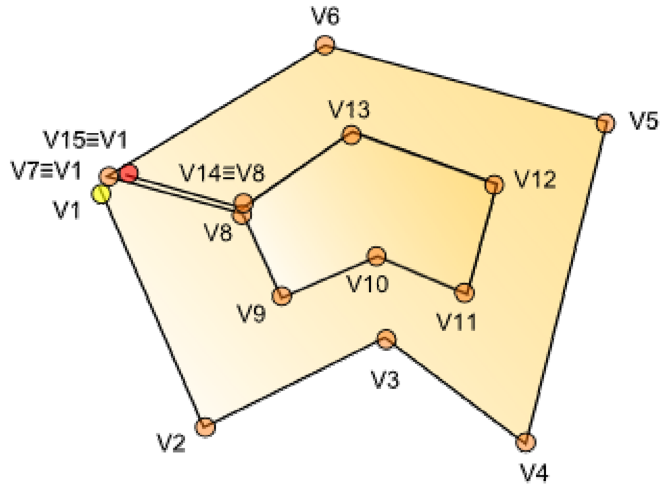
POLYGONS (overlapped areas)

With multiple polygons you can draw overlapped areas; like above:

- draw all the vertices having same wise;

- close each polygon;

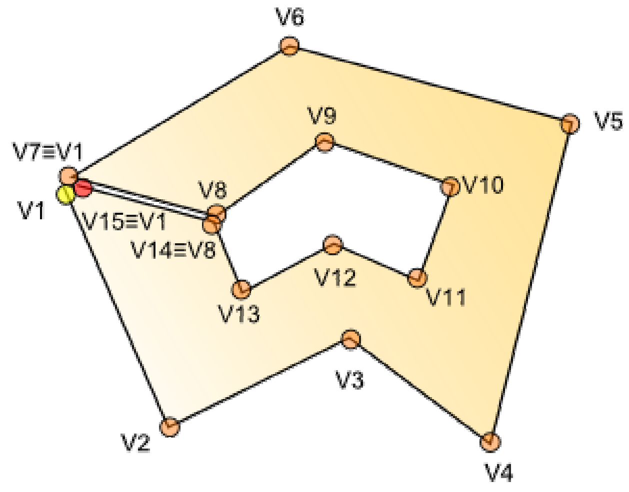
- don't forget to close the vertices chain in the first vertex.



A HOLLOW POLYGON

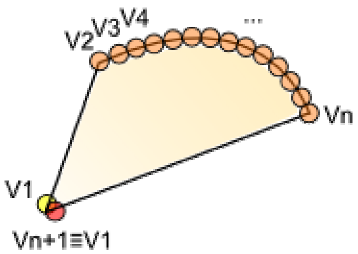
To have a hollow polygon, draw the internal and the external vertices with opposite wise;

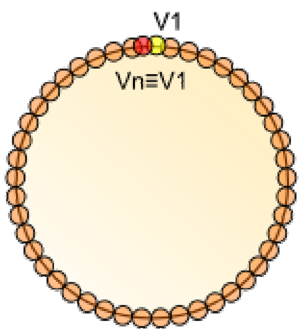
don't forget to close the vertices chain in the first vertex!



CURVED SIDES

Using an adequate approximation you can draw curved sides (and therefore generic figures).

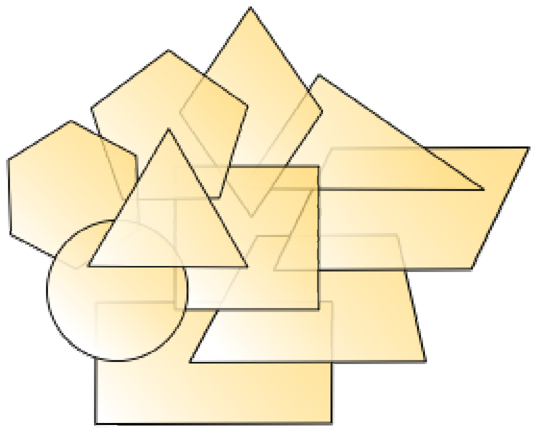


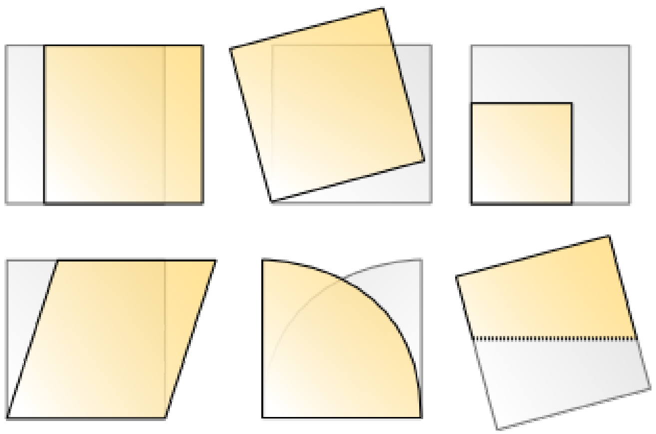


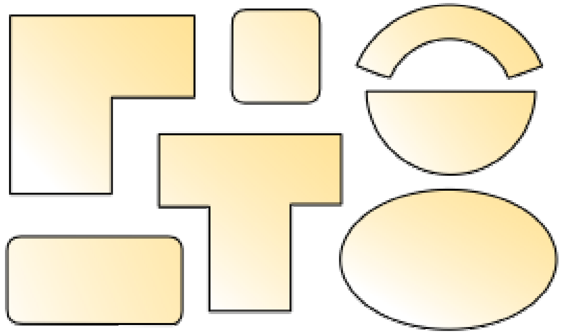
SHORTCUTS

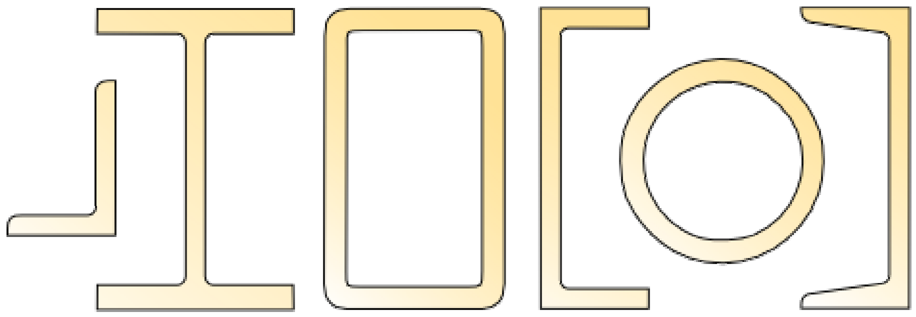
Save time: use "SHORTCUTS" to draw common shapes and to make transformations (rotate, scale, translate, etc...)

On the right of the GEOMETRY area there is a complete list of all available SHORTCUTS









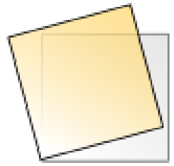
⊟— SHORTCUTS LIBRARY

⊞— Snippet: Operations with units 2012.01.00

TRANSFORMATIONS

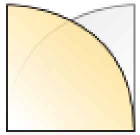
rotation matrix; rotatation about the origin of the Cartesian coordinate system.
θ: anticlockwise rotation.

$$Rot(\theta) := \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}$$



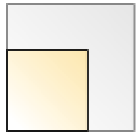
reflection matrix; reflection about a line through the origin of the Cartesian coordinate system.
θ: angle of the reflection line through the origin with the x-axis.

$$Ref(\theta) := \begin{pmatrix} \cos(2\cdot\theta) & \sin(2\cdot\theta) \\ \sin(2\cdot\theta) & -\cos(2\cdot\theta) \end{pmatrix}$$



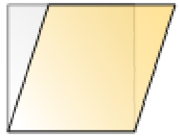
scaling matrix; scaling about the origin of the Cartesian coordinate system.
s.x: x-scaling factor;
s.y: y-scaling factor.

$$S_c(s_x, s_y) := \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix}$$



shear matrix; shearing about the origin of the Cartesian coordinate system.
λ.x: shearing factor, parallel to x;
λ.y: shearing factor, parallel to y.

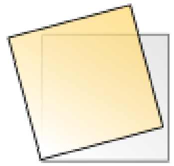
$$S_h(\lambda_x, \lambda_y) := \begin{pmatrix} 1 & \lambda_y \\ \lambda_x & 1 \end{pmatrix}$$



geometric translation of a polygon.
P: polygon matrix;
u.x: x translation;
u.y: y translation.

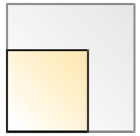
$$\mathrm{Translate}(P, u_x, u_y) := \left| \begin{array}{l} \text{"use homogeneous coordinates"} \\ \#P := \mathrm{augment}(P, (1 + \mathrm{matrix}(\mathrm{rows}(P), 1))) \\ \mathrm{submatrix}\left(\#P \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ u_x & u_y & 1 \end{pmatrix}, 1, \mathrm{rows}(P), 1, 2\right) \end{array} \right.$$

rotating a polygon about the origin of the Cartesian coordinate system.
P: polygon matrix;
θ: anticlockwise rotation.



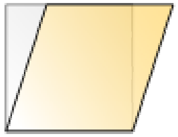
$$\mathrm{Rotate}(P, \theta) := \left| \begin{array}{l} \text{"use homogeneous coordinates"} \\ \#P := \mathrm{augment}(P, (1 + \mathrm{matrix}(\mathrm{rows}(P), 1))) \\ \mathrm{submatrix}\left(\#P \cdot \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}, 1, \mathrm{rows}(P), 1, 2\right) \end{array} \right.$$

scaling a polygon about the origin of the Cartesian coordinate system.
P: polygon matrix;
s.x: x scaling;
s.y: y scaling.



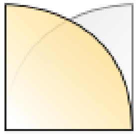
$$\mathrm{Scale}(P, s_x, s_y) := \left| \begin{array}{l} \text{"use homogeneous coordinates"} \\ \#P := \mathrm{augment}(P, (1 + \mathrm{matrix}(\mathrm{rows}(P), 1))) \\ \mathrm{submatrix}\left(\#P \cdot \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}, 1, \mathrm{rows}(P), 1, 2\right) \end{array} \right.$$

```
shearing a polygon about the origin of the Cartesian coordinate system.
P: polygon matrix;
λ.x: shearing factor, parallel to x;
λ.y: shearing factor, parallel to y.
```



```
Shear(P, λ_x, λ_y) := | "use homogeneous coordinates"
                      | #P := augment(P, (1 + matrix(rows(P), 1)))
                      |                    ( 1    λ_y  0 )
                      | submatrix( #P ·    ( λ_x  1    0 ) , 1, rows(P), 1, 2 )
                      |                    ( 0    0    1 )
```

```
mirroring a polygon.
P: polygon matrix;
θ: angle of the reflection line through the origin with the x-axis or case-insensitive axes name ["x","y","xy"].
```



```
Mirror(P, θ) := | "use homogeneous coordinates"
                | if IsString(θ)
                |   | Θ := "N.A."
                |   |        ( "x"  "X"  "y"  "Y"  "xy"  "XY" ) T
                |   | θ_a := (  0    0   π/2  π/2  π/4   π/4  )
                |   | for k ∈ 1 .. rows(θ_a)
                |   |   if θ = θ_a[k,1]
                |   |     Θ := θ_a[k,2]
                |   |   else
                |   |     0
                | else
                |   Θ := θ
                | #P := augment(P, (1 + matrix(rows(P), 1)))
                |                  ( cos(2·Θ)   sin(2·Θ)   0 )
                | submatrix( #P ·  ( sin(2·Θ)  -cos(2·Θ)   0 ) , 1, rows(P), 1, 2 )
                |                  (    0          0       1 )
```

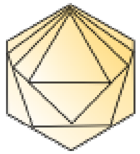
```
create a chunk of a polygon.
pgon: complete polygon;
TLV:  Threshold Limit Value;
zone: (relative) semi-plane to keep ["y+","y-","x+","x-"].
```

```
chunk(pgon, TLV, zone) :=
    Z := ( "y+"    0     1
           "y-"    π    -1
           "x+"    π/2   1
           "x-"   -π/2  -1 )
    for k ∈ 1 .. 4
      if Z[k,1] = zone
        Δθ := Z[k,2]
        #TLV := TLV · Z[k,3]
        #pgon := pgon · Rot(Δθ)
        break
      else
        0
    #chunk := (0  0)
    r := rows(pgon)
    j := 1
    for k ∈ 1 .. r - 1
      if #pgon[k,2] ≥ #TLV
        #chunk[j,1] := #pgon[k,1]
        #chunk[j,2] := #pgon[k,2]
        j := j + 1
      else
        0
      if (#pgon[k,2] - #TLV) · (#pgon[k+1,2] - #TLV) < 0
        #chunk[j,1] := eval( ((#pgon[k+1,1] - #pgon[k,1]) · (#TLV - #pgon[k,2])) / (#pgon[k+1,2] - #pgon[k,2]) + #pgon[k,1] )
        #chunk[j,2] := #TLV
        j := j + 1
      else
        0
    #chunk ← #chunk · Rot(-Δθ)
```

```
#chunk := #chunk· Rot(− Δθ)
#chunk[j,1] := eval(#chunk[1,1])
#chunk[j,2] := eval(#chunk[1,2])
#chunk
```

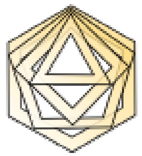
POLYGONS

```
convex regular polygons.
 s: number of sides/vertices;
 c: circumradius;
 θ: anticlockwise rotation.
```



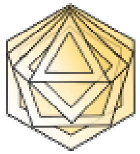
$$pgon(s, c, \theta) := \left| \begin{array}{l} \#out := (0\ \ 0) \\ s := \max\left(\begin{pmatrix} s \\ 3 \end{pmatrix}\right) \\ \text{for } k \in 0..s \\ \quad \left| \begin{array}{l} \#out_{k+1,1} := \text{eval}\left(c \cdot \sin\left(\frac{2 \cdot \pi \cdot k}{s}\right)\right) \\ \#out_{k+1,2} := \text{eval}\left(c \cdot \cos\left(\frac{2 \cdot \pi \cdot k}{s}\right)\right) \end{array} \right. \\ \left| \begin{array}{l} \#out := \#out \cdot Rot(\theta) \\ \#out_{s+1,1} := \text{eval}\left(\#out_{1,1}\right) \\ \#out_{s+1,2} := \text{eval}\left(\#out_{1,2}\right) \end{array} \right. \\ \#out \end{array} \right.$$

```
hollow convex regular polygons.
 s: number of sides/vertices;
 c: external circumradius;
 t: border thickness;
 θ: anticlockwise rotation.
```



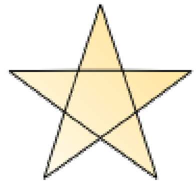
$$hPgon(s, c, t, \theta) := \left| \begin{array}{l} \#out := \text{stack}(pgon(s, c, 0), \text{reverse}(pgon(s, c - t, 0))) \\ \text{stack}(\#out, \text{eval}(\text{row}(\#out, 1))) \cdot Rot(\theta) \end{array} \right.$$

```
overlapped convex regular polygons.
 s: number of sides/vertices;
 ce: external circumradius;
 ci: internal circumradius;
 θ: anticlockwise rotation.
```



$$oPgon(s, c_e, c_i, \theta) := \left| \begin{array}{l} \#out := \text{stack}(pgon(s, c_e, 0), pgon(s, c_i, 0)) \\ \text{stack}(\#out, \text{eval}(\text{row}(\#out, 1))) \cdot Rot(\theta) \end{array} \right.$$

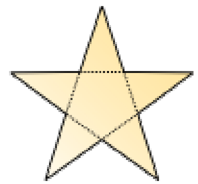
```
star regular polygons.
p: number of sides/vertices [must be ≥ 3];
q: connected vertices spacing [{p/q} is the polygon Schläfli symbol];
c: circumradius;
θ: anticlockwise rotation.
NOTE: stellations of regular or star polygons are for drawing purposes only.
```



```
sPgon(p, q, c, θ) :=
  q' := min( ( q ; round(p/2, 0) - 1 ) )
  gcd(a, b) := | if b = 0
               |   out := a
               | else
               |   out := gcd(b, mod(a, b))
               | out
  if p ≥ 3
    if (gcd(p, q') = 1) ∧ (q' > 1)
      #out := (0  0)
      for k ∈ 0 .. p
        #out_{k+1 1} := eval(c·sin(2·π·k·q'/p))
        #out_{k+1 2} := eval(c·cos(2·π·k·q'/p))
      #out := #out·Rot(θ)
      #out_{p+1 1} := eval(#out_{1 1})
      #out_{p+1 2} := eval(#out_{1 2})
      #out
    else
      "stellations"
      GCD := gcd(p, q')
      if GCD/q' = 1
        "stellation of a regular polygon"
        for k ∈ 1 .. q'
          #out_k := pgon(round(p/q', 0), c, 2·π/p·(k - 1))
        str2num(concat("sys(", substr(num2str(#out), 5)))·Rot(θ)
      else
        "stellation of a star polygon"
```

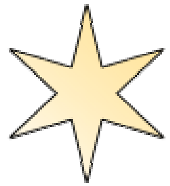
```
Stellation of a star polygon
if q'/GCD - trunc(q'/GCD) = 0
    #star := sPgon(p/GCD , GCD , c , θ)
    for k ∈ 1 .. GCD
        #out_k := #star·Rot((2·π/p)·(k - 1))
    eval(str2num(concat("sys(", substr(num2str(#out) , 5))))
else
    0
else
    0
```

edge of star polygons.
p: number of sides/vertices [must be ≥ 3];
q: connected vertices spacing [{p/q} is the polygon Schläfli symbol];
c: circumradius;
θ: anticlockwise rotation.



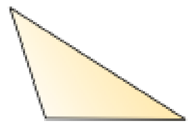
```
esPgon(p, q, c, θ) := | q' := min( ( q ; round(p/2, 0) - 1 ) )
                      | if p ≥ 3
                      |   | #out := (0  0)
                      |   |
                      |   |          c·sin( ((p/q' - 2)·π) / (2·(p/q')) )
                      |   | c_i := ------------------------------------------------
                      |   |        sin( π - ((p/q' - 2)·π) / (2·(p/q')) - (2·π)/(2·p) )
                      |   |
                      |   | for k ∈ 0 .. 2·p
                      |   |   | r := if ( k/2 - trunc(k/2) ) = 0
                      |   |   |        c
                      |   |   |      else
                      |   |   |        c_i
                      |   |   | #out_{k+1 1} := eval( r·sin( (2·π·k)/(2·p) ) )
                      |   |   | #out_{k+1 2} := eval( r·cos( (2·π·k)/(2·p) ) )
                      |   | | #out := #out·Rot(θ)
                      |   | | #out_{2·p+1 1} := eval( #out_{1 1} )
                      |   | | #out_{2·p+1 2} := eval( #out_{1 2} )
                      |   | #out
                      | else
                      |   0
```

```
star-shaped polygons.
 s: number of spikes [must be ≥ 3];
 c.e: external circumradius;
 c.i: internal circumradius;
 θ: anticlockwise rotation.
```



```
ssPgon(s, c_e, c_i, θ) := | if s ≥ 3
                          |   | #out := (0  0)
                          |   | for k ∈ 0 .. 2·s
                          |   |   | r := if (k/2 - trunc(k/2)) = 0
                          |   |   |        c_e
                          |   |   |      else
                          |   |   |        c_i
                          |   |   | #out_(k+1,1) := eval(r·sin(2·π·k/(2·s)))
                          |   |   | #out_(k+1,2) := eval(r·cos(2·π·k/(2·s)))
                          |   |  | #out := #out·Rot(θ)
                          |   |  | #out_(2·s+1,1) := eval(#out_(1,1))
                          |   |  | #out_(2·s+1,2) := eval(#out_(1,2))
                          |   | #out
                          | else
                          |   0
```

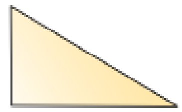
```
triangle.
 B: x-length;
 H: y-shift of top vertex;
 a: x-shift of top vertex;
 θ: anticlockwise rotation.
```



$$\text{triangle}(B, H, a, \theta) := \begin{pmatrix} 0 & 0 \\ B & 0 \\ a & H \\ 0 & 0 \end{pmatrix} \cdot \text{Rot}(\theta)$$

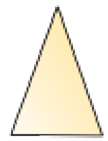
```
right triangle.
 B: x-length;
 H: y-shift of top vertex;
 θ: anticlockwise rotation.
```

$$rTriangle(B, H, \theta) := \begin{pmatrix} 0 & 0 \\ B & 0 \\ 0 & H \\ 0 & 0 \end{pmatrix} \cdot Rot(\theta)$$



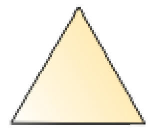
```
isosceles triangle.
 B: x-length;
 H: y-shift of top vertex;
 θ: anticlockwise rotation.
```

$$iTriangle(B, H, \theta) := \begin{pmatrix} 0 & 0 \\ B & 0 \\ \frac{B}{2} & H \\ 0 & 0 \end{pmatrix} \cdot Rot(\theta)$$

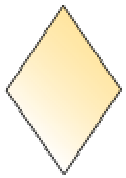


```
equilateral triangle.
 a: side length;
 θ: anticlockwise rotation.
```

$$eTriangle(a, \theta) := \begin{pmatrix} 0 & 0 \\ a & 0 \\ \frac{a}{2} & \frac{\sqrt{3}}{2} \cdot a \\ 0 & 0 \end{pmatrix} \cdot Rot(\theta)$$

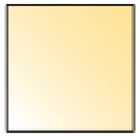


```
rhombus.
 p: x-diagonal;
 q: y-diagonal;
 θ: anticlockwise rotation.
```



$$\mathrm{rhombus}(p, q, \theta) := \begin{pmatrix} \frac{p}{2} & 0 \\ 0 & \frac{q}{2} \\ -\frac{p}{2} & 0 \\ 0 & -\frac{q}{2} \\ \frac{p}{2} & 0 \end{pmatrix} \cdot \mathrm{Rot}(\theta)$$

```
square.
 a: sides length;
 θ: anticlockwise rotation.
```

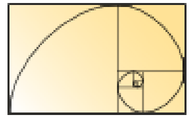


$$\mathrm{square}(a, \theta) := \begin{pmatrix} 0 & 0 \\ a & 0 \\ a & a \\ 0 & a \\ 0 & 0 \end{pmatrix} \cdot \mathrm{Rot}(\theta)$$

```
rectangle.
 B: width;
 H: height;
 θ: anticlockwise rotation.
```

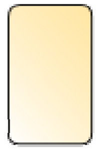
$$\mathrm{rectangle}(B, H, \theta) := \begin{pmatrix} 0 & 0 \\ B & 0 \\ B & H \\ 0 & H \\ 0 & 0 \end{pmatrix} \cdot \mathrm{Rot}(\theta)$$

```
golden rectangle.
 B: width;
 θ: anticlockwise rotation.
```



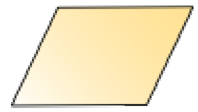
$$\text{goldenRect}(B,\theta) := \begin{array}{|l} \varphi := \dfrac{1+\sqrt{5}}{2} \\ H := \dfrac{B}{\varphi} \\ \begin{pmatrix} 0 & 0 \\ B & 0 \\ B & H \\ 0 & H \\ 0 & 0 \end{pmatrix} \cdot \text{Rot}(\theta) \end{array}$$

```
rounded rectangle.
 h: external height;
 b: external width;
 r: corner radius;
 θ: anticlockwise rotation.
```



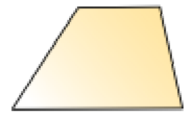
```
roundedRect(h, b, r, θ) := | #out := if r > 0
                           |           | #tmp := (0  0)
                           |           | pts := 15
                           |           | for k ∈ 0 ..pts
                           |           |    | #tmp[k+1, 1] := eval(r·cos(π/2 · k/pts) + b/2 − r)
                           |           |    | #tmp[k+1, 2] := eval(r·sin(π/2 · k/pts) + h/2 − r)
                           |           | #tmp
                           |          else
                           |           (b/2  h/2)
                           | #out := stack(#out, eval(reverse(#out·((−1, 0), (0, 1)))))
                           | #out := stack(#out, eval(reverse(#out·((1, 0), (0, −1)))), eval(row(#out, 1)))
                           | #out·Rot(θ)
```

parallelogram.
b: base;
a: x-shift of top base;
h: height;
θ: anticlockwise rotation.



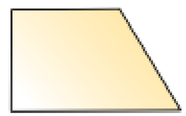
$$\mathrm{parallelogram}(b, a, h, \theta) := \begin{pmatrix} 0 & 0 \\ b & 0 \\ b+a & h \\ a & h \\ 0 & 0 \end{pmatrix} \cdot \mathrm{Rot}(\theta)$$

trapezoid.
a: top base;
b: bottom base;
c: x-shift of top base;
h: height;
θ: anticlockwise rotation.



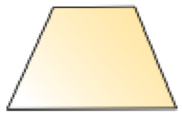
$$\mathrm{trapezoid}(a, b, c, h, \theta) := \begin{pmatrix} 0 & 0 \\ b & 0 \\ c+a & h \\ a & h \\ 0 & 0 \end{pmatrix} \cdot \mathrm{Rot}(\theta)$$

right trapezoid.
a: top base;
b: bottom base;
h: height;
θ: anticlockwise rotation.



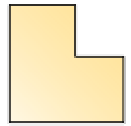
$$\mathrm{rTrapezoid}(a, b, h, \theta) := \begin{pmatrix} 0 & 0 \\ b & 0 \\ a & h \\ 0 & h \\ 0 & 0 \end{pmatrix} \cdot \mathrm{Rot}(\theta)$$

isosceles trapezoid.
a: top base;
b: bottom base;
h: height;
θ: anticlockwise rotation.



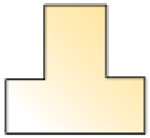
$$\text{iTrapezoid}(a, b, h, \theta) := \begin{pmatrix} 0 & 0 \\ b & 0 \\ a + \frac{(b-a)}{2} & h \\ \frac{(b-a)}{2} & h \\ 0 & 0 \end{pmatrix} \cdot \text{Rot}(\theta)$$

L shape.
a: top base;
b: bottom base;
c: top height;
h: total height;
θ: anticlockwise rotation.



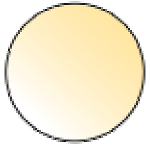
$$L(a, b, c, h, \theta) := \begin{pmatrix} 0 & 0 \\ b & 0 \\ b & h-c \\ a & h-c \\ a & h \\ 0 & h \\ 0 & 0 \end{pmatrix} \cdot \text{Rot}(\theta)$$

```
T shape.
a: top base;
b: bottom base;
c: top height;
h: total height;
θ: anticlockwise rotation.
```



$$T(a, b, c, h, \theta) := \left| \begin{array}{l} \#out := \begin{pmatrix} \frac{a}{2} & h \\ \frac{a}{2} & h-c \\ \frac{b}{2} & h-c \\ \frac{b}{2} & 0 \end{pmatrix} \\ \#out := stack\left(\#out, eval\left(reverse\left(\#out \cdot \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}\right)\right), eval(row(\#out, 1))\right) \\ \#out \cdot Rot(\theta) \end{array} \right.$$

```
circle.
D: diameter.
```



$$circle(D) := pgon\left(100, \frac{D}{2}, 0\right)$$

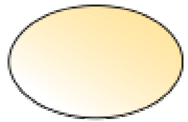
```
semicircle.
D: diameter;
θ: anticlockwise rotation.
```

$$semicircle(D, \theta) := \left| \begin{array}{l} \#out := (0\ \ 0) \\ pts := 100 \\ \text{for } k \in 0..pts \\ \quad \left| \begin{array}{l} \#out_{k+1\,1} := eval\left(\frac{D}{2} \cdot \cos\left(\frac{\pi \cdot k}{pts}\right)\right) \\ \#out_{k+1\,2} := eval\left(\frac{D}{2} \cdot \sin\left(\frac{\pi \cdot k}{pts}\right)\right) \end{array} \right. \\ \#out := stack(\#out, eval(row(\#out, 1))) \\ \#out \cdot Rot(\theta) \end{array} \right.$$

segmental arch.
r.e: external radius;
r.i: external radius;
ω: inner angle;
θ: anticlockwise rotation.

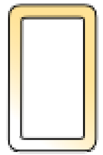
$$
arch(r_e, r_i, \omega, \theta) :=
\left|
\begin{array}{l}
\#P1 \leftarrow (0\ \ 0) \\
pts \leftarrow 50 \\
\text{for } k \in 0..pts \\
\quad \left| \begin{array}{l}
\#P1_{k+1,1} \leftarrow eval\left(r_e \cdot \cos\left(\frac{\pi-\omega}{2} + \frac{\omega \cdot k}{pts}\right)\right) \\
\#P1_{k+1,2} \leftarrow eval\left(r_e \cdot \sin\left(\frac{\pi-\omega}{2} + \frac{\omega \cdot k}{pts}\right)\right)
\end{array} \right. \\
\#P2 \leftarrow (0\ \ 0) \\
pts \leftarrow 50 \\
\text{for } k \in pts..0 \\
\quad \left| \begin{array}{l}
\#P2_{pts-k+1,1} \leftarrow eval\left(r_i \cdot \cos\left(\frac{\pi-\omega}{2} + \frac{\omega \cdot k}{pts}\right)\right) \\
\#P2_{pts-k+1,2} \leftarrow eval\left(r_i \cdot \sin\left(\frac{\pi-\omega}{2} + \frac{\omega \cdot k}{pts}\right)\right)
\end{array} \right. \\
\#P \leftarrow stack(\#P1, \#P2) \\
\#P \leftarrow stack(\#P, eval(row(\#P, 1))) \\
\#P \cdot Rot(\theta)
\end{array}
\right.
$$

```
ellipse.
 p: x-axis;
 q: y-axis;
 θ: anticlockwise rotation.
```



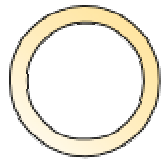
$$\text{ellipse}(p, q, \theta) := \left| \begin{array}{l} a \leftarrow \dfrac{p}{2} \\ b \leftarrow \dfrac{q}{2} \\ \#out \leftarrow \begin{pmatrix} 0 & 0 \end{pmatrix} \\ pts \leftarrow 40 \\ \text{for } k \in 0 .. pts \\ \quad \left| \begin{array}{l} \#out_{k+1\,1} \leftarrow \text{eval}\left(-a + \dfrac{p}{pts} \cdot k\right) \\ \#out_{k+1\,2} \leftarrow \text{eval}\left(b \cdot \sqrt{1 - \left(\dfrac{\#out_{k+1\,1}}{a}\right)^2}\right) \end{array} \right. \\ \#out \leftarrow \text{stack}\left(\#out, \text{eval}\left(\text{reverse}\left(\#out \cdot \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\right)\right)\right) \\ \#out \cdot \text{Rot}(\theta) \end{array} \right.$$

Hollow Structural Sections.
h: external height;
b: external width;
t: wall thickness;
r.e: external corner radius (typical=2t);
r.i: internal corner radius (typical=t);
θ: anticlockwise rotation.



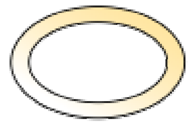
```
HSS_beam(h, b, t, r_e, r_i, θ) := | #out := stack(roundrect(h, b, r_e, 0), eval(reverse(roundrect(h - 2·t, b - 2·t, r_i, 0))))
                                   | stack(#out, eval(row(#out, 1)))·Rot(θ)
```

Circular Hollow Sections.
De: external diameter;
t: border thickness.



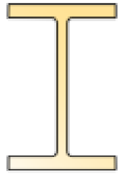
$$CHS_{beam}(D_e, t) := hPgon\left(75, \frac{D_e}{2}, t, 0\right)$$

Elliptical Hollow Sections.
p: external x-axis;
q: external y-axis;
t: wall thickness;
θ: anticlockwise rotation.



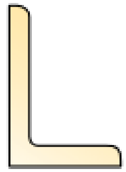
```
EHS_beam(p, q, t, θ) := | #out := stack(ellipse(p, q, θ), eval(reverse(ellipse(p - 2·t, q - 2·t, θ))))
                         | stack(#out, eval(row(#out, 1)))·Rot(θ)
```

european I-beam (UE:IPE/HE/HL/HD/HP;UK:UB/UC/UBP; US:W/HP; RU:HG; JP:H).
h: beam height;
b: flange width;
tw: web thickness;
tf: flange thickness;
r: web-flange junction radius;
θ: anticlockwise rotation.



```
I_beam(h, b, t_w, t_f, r, θ) := | #out := if r > 0
                                |            | #tmp := (0  0)
                                |            | pts := 25
                                |            | for k ∈ 0 .. pts
                                |            |    | #tmp_(k+1,1) := eval( r·cos(π/2 + π/2 · k/pts) + t_w/2 + r )
                                |            |    | #tmp_(k+1,2) := eval( r·sin(π/2 + π/2 · k/pts) + h/2 - t_f - r )
                                |            | #tmp
                                |         else
                                |            ( t_w/2   h/2 - t_f )
                                | #out := stack( [ b/2   h/2 ; b/2   h/2 - t_f ], #out )
                                | #out := stack( #out, eval( reverse( #out · [ 1  0 ; 0  -1 ] ) ) )
                                | #out := stack( #out, eval( reverse( #out · [ -1  0 ; 0  1 ] ) ), eval( row(#out, 1) ) )
                                | #out · Rot(θ)
```

```
european (un)equal legs angles.
h: height;
b: width;
t: thickness;
r1: legs junction radius;
r2: legs corner radius;
θ: anticlockwise rotation.
```



$$L_{beam}(h, b, t, r_1, r_2, \theta) :=$$

```
#P1 := | 0  h |
       | 0  0 |
       | b  0 |
#P2 := if r_2 > 0
           #tmp := (0  0)
           pts := 30
           for k ∈ 0 .. pts
               #tmp_{k+1 1} := eval(r_2 · cos(π/2 · k/pts) + b − r_2)
               #tmp_{k+1 2} := eval(r_2 · sin(π/2 · k/pts) + t − r_2)
           #tmp
       else
           (b  t)
#P3 := if r_1 > 0
           #tmp := (0  0)
           pts := 30
           for k ∈ 0 .. pts
               #tmp_{k+1 1} := eval(r_1 · cos(3/2 · π − π/2 · k/pts) + t + r_1)
               #tmp_{k+1 2} := eval(r_1 · sin(3/2 · π − π/2 · k/pts) + t + r_1)
           #tmp
       else
           (t  t)
#P4 := if r_2 > 0
           #tmp := (0  0)
           pts := 30
           for k ∈ 0 .. pts
               #tmp_{k+1 1} := eval(r_2 · cos(π/2 · k/pts) + t − r_2)
```

$$\#tmp_{k+1,2} := eval\left(r_2 \cdot \sin\left(\frac{\pi}{2} \cdot \frac{k}{pts}\right) + h - r_2\right)$$

#tmp

else

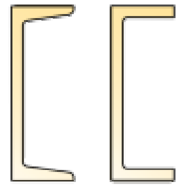
$(t \;\; h)$

$\#P5 := (0 \;\; h)$

$\#P := stack(\#P1, \#P2, \#P3, \#P4, \#P5)$

$\#P \cdot Rot(\theta)$

european U-beam [DIN 1026-1:2000] (UPN) and parallel flange channels (UPE/PFC).
h: beam height [mm];
b: flange width;
tw: web thickness;
tf: flange thickness;
r1: web-flange junction radius;
r2: flange corner radius (UPE/PFC when r2=0);
θ: anticlockwise rotation.



$U_{beam}(h, b, t_w, t_f, r_1, r_2, \theta) :=$ if $r_2 > 0$

if $h \le 300$

$u := \frac{b}{2}$

$p := \frac{8}{100}$

else

$u := \frac{b - t_w}{2}$

$p := \frac{5}{100}$

else

$u := \frac{b}{2}$

$p := 0$

$d := eval\left(h - 2 \cdot p \cdot \left(b - u - r_1 - t_w\right) - 2 \cdot \left(r_1 + t_f\right)\right)$

$$\#P1 := \begin{pmatrix} 0 & \frac{h}{2} \\ b & \frac{h}{2} \end{pmatrix}$$

$\#P2 :=$ if $r_2 > 0$

```
        #tmp := (0  0)
        pts := 25
        for k ∈ 0 .. pts
            #tmp[k+1, 1] := eval(r_2 · cos(-π/2 · k/pts) + b - r_2)
            #tmp[k+1, 2] := eval(r_2 · sin(-π/2 · k/pts) + h/2 - t_f + (u - r_2) · p + r_2)
        #tmp
    else
        (b   h/2 - t_f)
#P3 := (b - u   h/2 - t_f)
#P4 := if r_1 > 0
        #tmp := (0  0)
        pts := 25
        for k ∈ 0 .. pts
            #tmp[k+1, 1] := eval(r_1 · cos(π/2 + π/2 · k/pts) + t_w + r_1)
            #tmp[k+1, 2] := eval(r_1 · sin(π/2 + π/2 · k/pts) + d/2)
        #tmp
    else
        (t_w   h/2 - t_f)
#P5 := (0   h/2)
#P := stack(#P1 , #P2 , #P3 , #P4)
#P := stack(#P , eval(reverse(#P · [[1, 0], [0, -1]])) , #P5)
#P · Rot(θ)
```

PSEUDONYMS

$ring(D_e, t) := CHS_{beam}(D_e, t)$

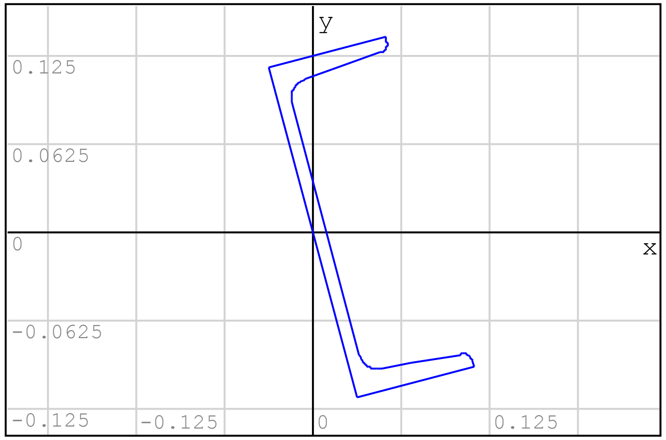
GEOMETRY

GEOMETRY

polygon coordinates - x y

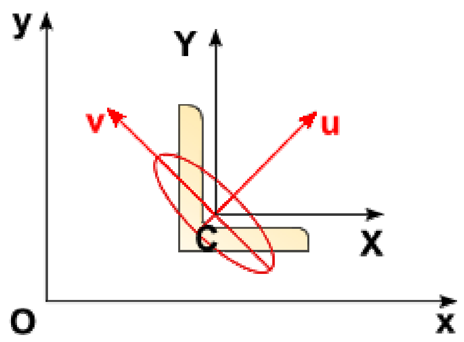
polygon := $U_{beam}(240, 85, 9.5, 13, 13, 6.5, 15°)$ mm

polygon



polygon

Axes conventions



⊞— PLOT SETTINGS

⊟— CALCULATIONS OF CROSS-SECTION PROPERTIES

coordinates

$x := eval(col(polygon, 1))$

$y := eval(col(polygon, 2))$

vertices (+1)

$n := length(x)$

perimeter

$$P := \sum_{i=1}^{n-1} \sqrt{\left(x_{i+1} - x_i\right)^2 + \left(y_{i+1} - y_i\right)^2}$$

calculate total length of overlapped segments with opposite wise

```
ΔP := eval( #links ← 0
            for j ∈ 1 .. n − 2
              sWise ← (oWise ← 0)
              for k ∈ j + 2 .. n
                if ((x_j = x_k) ∧ (y_j = y_k)) ∧ ((x_{j+1} = x_{k−1}) ∧ (y_{j+1} = y_{k−1}))
                  oWise ← oWise + 1
                else
                  if ((x_j = x_{k−1}) ∧ (y_j = y_{k−1})) ∧ ((x_{j+1} = x_k) ∧ (y_{j+1} = y_k))
                    sWise ← sWise + 1
                  else
                    0
              if oWise > 0
                #links ← eval(#links + sqrt((x_{j+1} − x_j)^2 + (y_{j+1} − y_j)^2) · (2·(oWise − sWise)))
              else
                0
            #links )
```

$P := P - \Delta P$

area

$$A := \frac{1}{2} \cdot \sum_{i=1}^{n-1} \left(x_i \cdot y_{i+1} - x_{i+1} \cdot y_i \right)$$

centroid

$$C_x := \frac{1}{6 \cdot A} \cdot \sum_{i=1}^{n-1} \left[\left(x_i + x_{i+1} \right) \cdot \left(x_i \cdot y_{i+1} - x_{i+1} \cdot y_i \right) \right]$$

$$C_y := \frac{1}{6 \cdot A} \cdot \sum_{i=1}^{n-1} \left[\left(y_i + y_{i+1} \right) \cdot \left(x_i \cdot y_{i+1} - x_{i+1} \cdot y_i \right) \right]$$

use centroid as origin

$$C_x := \mathrm{dRound}\left(C_x, 10 \right)$$

$$C_y := \mathrm{dRound}\left(C_y, 10 \right)$$

$$X := x - C_x$$

$$Y := y - C_y$$

second moments of area (any cross section allowed)

$$J_{XX} := \frac{1}{12} \cdot \sum_{i=1}^{n-1} \left[\left({Y_i}^2 + Y_i \cdot Y_{i+1} + {Y_{i+1}}^2 \right) \cdot \left(X_i \cdot Y_{i+1} - X_{i+1} \cdot Y_i \right) \right]$$

$$J_{YY} := \frac{1}{12} \cdot \sum_{i=1}^{n-1} \left[\left({X_i}^2 + X_i \cdot X_{i+1} + {X_{i+1}}^2 \right) \cdot \left(X_i \cdot Y_{i+1} - X_{i+1} \cdot Y_i \right) \right]$$

$$J_{XY} := \frac{1}{24} \cdot \sum_{i=1}^{n-1} \left(\left(X_i \cdot Y_{i+1} + 2 \cdot X_i \cdot Y_i + 2 \cdot X_{i+1} \cdot Y_{i+1} + X_{i+1} \cdot Y_i \right) \cdot \left(X_i \cdot Y_{i+1} - X_{i+1} \cdot Y_i \right) \right)$$

output adjustements due to clockwise/anticlockwise vertices definition

$$P := \mathrm{dRound}(P, 10)$$

$$A := |A|$$

$$I_X := |J_{XX}|$$

$$I_Y := |J_{YY}|$$

$$I_{XY} := \mathrm{dRound}(J_{XY}, 10)$$

radii of gyration

$$i_X := \sqrt{\frac{I_X}{A}}$$

$$i_Y := \sqrt{\frac{I_Y}{A}}$$

elastic section modulus

$$W_{el;top} := \frac{I_X}{|\max(Y)|}$$

$$W_{el;bot} := \frac{I_X}{\left|\min(Y)\right|}$$

$$W_{el;left} := \frac{I_Y}{\left|\min(X)\right|}$$

$$W_{el;right} := \frac{I_Y}{\left|\max(X)\right|}$$

orientation of principal axes of inertia (u,v)

$$\alpha := \text{if } \text{dRound}\left(I_X - I_Y, 10\right) \neq 0$$

$$-\frac{1}{2}\cdot \text{atan}\left(\frac{2\cdot I_{XY}}{I_X - I_Y}\right)\cdot \text{sign}\left(J_{XX}\right) + \left(\left(I_X - I_Y\right) < 0\right)\cdot\frac{\pi}{2}$$

$$\text{else}$$

$$\frac{\pi}{4}\cdot \text{sign}\left(I_{XY}\right)$$

principal moments of inertia

$$I_u := \frac{I_X + I_Y}{2} + \frac{1}{2}\cdot\sqrt{\left(I_X - I_Y\right)^2 + 4\cdot I_{XY}^{\;2}}$$

$$I_v := \frac{I_X + I_Y}{2} - \frac{1}{2}\cdot\sqrt{\left(I_X - I_Y\right)^2 + 4\cdot I_{XY}^{\;2}}$$

radii of gyration about principal axes of inertia

$$i_u := \sqrt{\frac{I_u}{A}}$$

$$i_v := \sqrt{\frac{I_v}{A}}$$

plot with centroid as origin

$polygon_C := Translate(polygon, -C_x, -C_y)$

ellipsoid of gyration

$ellipsoid := eval(stack(ellipse(eval(2\cdot i_v), eval(2\cdot i_u), 0), (0\ \ 0), (0\ \ i_u), (0\ \ -i_u), (0\ \ 0), (i_v\ \ 0)))$

vertices numbering

function to numbering vertices of polygons.
values for opt:
"all" to enumerate all vertices;
to enumerate vertices spaced at least the ### value;
any other text to disable numbering.

```
numbering(pgon, opt) := | #color := vertices_color
                        | #size := vertices_namesize
                        | r := rows(pgon)
                        | if ¬ IsString(opt)
                        |   | v_n := (pgon_{1,1}  pgon_{1,2}  concat("1(", num2str(r), ")")  #size  #color)
                        |   | c := 2
                        |   | for k ∈ 2 .. r-1
                        |   |   | #d_p := eval(sqrt((pgon_{k,1} - pgon_{k-1,1})^2 + (pgon_{k,2} - pgon_{k-1,2})^2))
                        |   |   | #d_f := eval(sqrt((pgon_{k,1} - pgon_{k+1,1})^2 + (pgon_{k,2} - pgon_{k+1,2})^2))
                        |   |   | if (#d_p ≥ opt) ∧ (#d_f ≥ opt)
                        |   |   |   | (v_n)_{c,1} := pgon_{k,1}
                        |   |   |   | (v_n)_{c,2} := pgon_{k,2}
                        |   |   |   | (v_n)_{c,3} := num2str(k)
                        |   |   |   | (v_n)_{c,4} := #size
                        |   |   |   | v      := #color
```

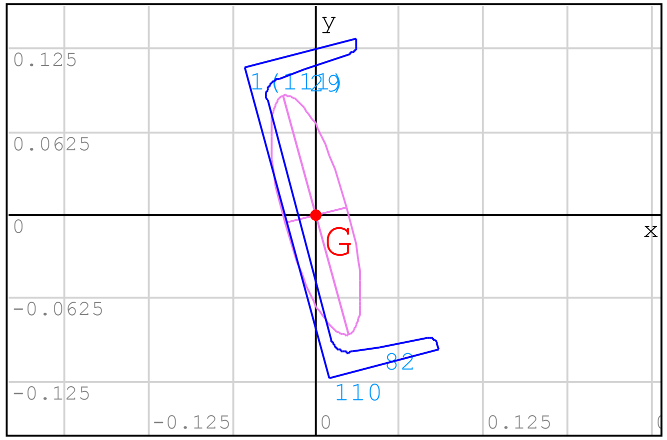
```
        v        ← "color
         n c 5
        c := c + 1
      else
        0
    v
     n
  else
    if opt = "all"
      v  := pgon
       n
      for k ∈ 1 .. r
        eval( v       := num2str(k)
               n k 3
              v       := #size
               n k 4
              v       := #color )
               n k 5
      v      := concat("1(", num2str(r), ")")
       n 1 3
      v      := ""
       n r 3
      v
       n
    else
      (0 0)
```

$vertices := numbering(polygon_C, vertices_{numbering})$

$vertices_{uv} := augment(submatrix(vertices, 1, rows(vertices), 1, 2) \cdot Rot(-\alpha), submatrix(vertices, 1, rows(vertices), 3, 5))$

CROSS-SECTION PROPERTIES

polygon refered to centroid



perimeter

$P = 775.65\ mm$

area

$A = 4230.01\ mm^2$

centroid (relative to global coordinates x,y)

$C_x = 21.6\ mm$ $C_y = 5.79\ mm$

second moments of area (relative to centroid)

$I_X = 3373.7\ cm^4$ $I_Y = 471.96\ cm^4$ $I_{XY} = 837.66\ cm^4$

radii of gyration

$i_X = 8.93\ cm$ $i_Y = 3.34\ cm$

elastic section modulus

$W_{el;top} = 255.35\ cm^3$ $W_{el;bot} = 277.22\ cm^3$ $W_{el;left} = 89.62\ cm^3$ $W_{el;right} = 51.55\ cm^3$

orientation of principal axes of inertia (u,v)

$\alpha = 15\ °$

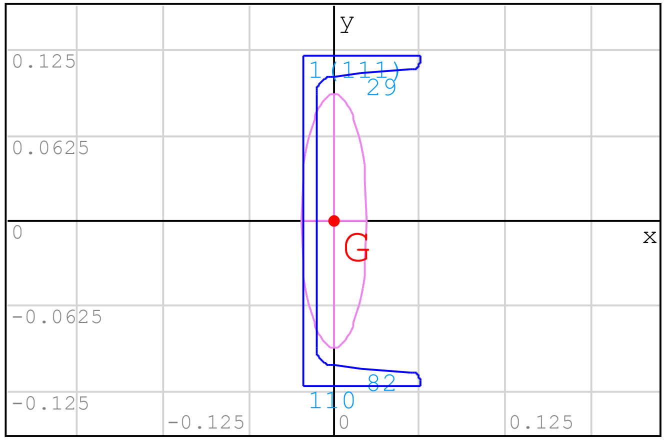
principal moments of inertia

$I_u = 3598.15\ cm^4$ $I_v = 247.51\ cm^4$

radii of gyration about principal axes of inertia

$i_u = 9.22\ cm$ $i_v = 2.42\ cm$

orientation about principal axes of inertia



CALCULATION OF PLASTIC SECTION MODULUS (CREOSS-SECTION HAVE CONSTANT YELDING STRESS)

plastic neutral axis position

polygon area

$$A_P(pgon) := \left| \begin{array}{l} r := rows(pgon) \\ \text{if } r > 3 \\ \quad \frac{1}{2} \cdot eval\left(\sum_{n=1}^{r-1} \left(pgon_{n\,1} \cdot pgon_{n+1\,2} - pgon_{n+1\,1} \cdot pgon_{n\,2} \right) \right) \\ \text{else} \\ \quad 0 \end{array} \right.$$

SMath geometry Unit of Measurement

UoM := UoM(P)

check if "polygon" could have multiple areas

$isMultiPgon := eval(\Delta P \neq 0)$

search for a Plastic Neutral Axis (PNA).
pgon: polygon matrix;
axis: axis orthogonal to the PNA orientation ["x" or "y"];
mPgon: 1 if is a multiple polygon, 0 otherwise.

```
PNA_hunter(pgon, axis, mPgon) := | column := eval(1 + (axis = "y"))
                                 | A_pgon := A
                                 | bisection(L_bound, U_bound) := | p := (L_bound + U_bound) / 2
                                 |                                | f_p := eval(round((A_pgon - 2·|A_P(chunk(pgon, p, concat(axis, "-")))|) / A_pgon, 5))
                                 |                                | if f_p = 0
                                 |                                |   p
                                 |                                | else
                                 |                                |   if f_p < 0
                                 |                                |     bisection(L_bound, p)
                                 |                                |   else
                                 |                                |     bisection(p, U_bound)
                                 | p_min := eval(min(col(pgon, column)))
                                 | p_max := eval(max(col(pgon, column)))
                                 | target := (bisection(p_min, p_max))
                                 | Δ_target := 0
                                 | if mPgon ≠ 0
                                 |   | "search PNA regions"
                                 |   | p_last := (p_tmp := (p_target := target_1))
                                 |   | P_chunk := sort(col(chunk(pgon, p_target, concat(axis, "-")), column))
                                 |   | for k ∈ length(P_chunk) - 1 .. 1
                                 |   |   | p_# := eval(P_chunk_k)
                                 |   |   | if round((p_# - p_tmp) / UoM, 5) ≠ 0
                                 |   |   |   | f_p# := (A_pgon - 2·|A_P(chunk(pgon, p_#, concat(axis, "-")))|) / A_pgon
                                 |   |   |   | if round(f_p#, 5) = 0
                                 |   |   |   |   p_last := p_#
```

```
            last    #
        else
            target_1 := p_last
            break
        p_tmp := p_#
    else
        0
  P_chunk := sort(col(chunk(pgon, p_target, concat(axis, "+")), column))
  p_last := (p_tmp := p_target)
  for k ∈ 2 .. length(P_chunk)
    p_# := eval(P_chunk_k)
    if round((p_# - p_tmp)/UoM, 5) ≠ 0
        f_p# := (A_pgon - 2·|A_P(chunk(pgon, p_#, concat(axis, "-")))|) / A_pgon
        if round(f_p#, 5) = 0
          p_last := p_#
        else
          if p_last ≠ p_target
            target_2 := p_last
            Δ_target := target_2 - target_1
          else
            0
          break
        p_tmp := p_#
    else
        0
else
  0
(target
 Δ_target)
```

PNA x-position

$PNA := eval\left(PNA_{hunter}(polygon, \text{"x"}, isMultiPgon)\right)$

$PNA_x := eval\left(PNA_1\right)$ $\qquad$ $PNA_{\Delta x} := eval\left(PNA_2\right)$

PNA y-position

$PNA := eval\left(PNA_{hunter}(polygon, \text{"y"}, isMultiPgon)\right)$

$PNA_y := eval\left(PNA_1\right)$ $\qquad$ $PNA_{\Delta y} := eval\left(PNA_2\right)$

allowable region for PNAs

$PNAs_{BOX} := Translate\left(rectangle\left(PNA_{\Delta x}, PNA_{\Delta y}, 0\right), PNA_{x_1} - C_x, PNA_{y_1} - C_y\right)$

PNA infos to plot

```
PNAs_plot := | #P := ( 0  0  centroid_symbol  centroid_symbolsize  centroid_color )
             |        ( 0  0  centroid_name    centroid_namesize    centroid_color )
             | for k ∈ 1 .. 2
             |    #P := eval( stack( #P, ( PNAs_BOX_k,1  PNAs_BOX_k,2  PNA_symbol  PNA_symbolsize  PNA_color ) ) )
             |                          ( PNAs_BOX_k,1  PNAs_BOX_k,2  PNA_name    PNA_namesize    PNA_color )
             | #P
```

$msg_1 := 0$ script has found discontinuous equal areas; green box/line show the PNAs region

$msg_2 := 0$ PNAs and plastic section modulus calculated for equal areas

$msg := if\left(\left(PNA_{\Delta x} > 0\right) \vee \left(PNA_{\Delta y} > 0\right)\right)$
$\quad description\left(msg_2\right)$

```
else
  description(msg_2)
```

plastic section modulus

$$W_{pl}(pgon, PNA_{x;y}, axis) := \left| \begin{array}{l} \#pgon \leftarrow \text{if } axis = \text{"x"} \\ \quad\quad eval\left(pgon \cdot Rot\left(\frac{\pi}{2}\right)\right) \\ \quad \text{else} \\ \quad\quad eval(pgon) \\ \left| \begin{array}{l} \#chunk \leftarrow chunk(\#pgon, PNA_{x;y}, \text{"y+"}) \\ x \leftarrow col(\#chunk, 1) \\ y \leftarrow col(\#chunk, 2) \\ r \leftarrow rows(\#chunk) \\ A_{pp} \leftarrow A_p(\#chunk) \\ C_{pp} \leftarrow \frac{1}{6 \cdot A_{pp}} \cdot eval\left[\sum_{i=1}^{r-1} \left[\left(y_i + y_{i+1}\right) \cdot \left(x_i \cdot y_{i+1} - x_{i+1} \cdot y_i\right)\right]\right] \end{array} \right. \\ \left| \begin{array}{l} \#chunk \leftarrow chunk(\#pgon, PNA_{x;y}, \text{"y-"}) \\ x \leftarrow col(\#chunk, 1) \\ y \leftarrow col(\#chunk, 2) \\ r \leftarrow rows(\#chunk) \\ A_{np} \leftarrow A_p(\#chunk) \\ C_{np} \leftarrow \frac{1}{6 \cdot A_{np}} \cdot eval\left[\sum_{i=1}^{r-1} \left[\left(y_i + y_{i+1}\right) \cdot \left(x_i \cdot y_{i+1} - x_{i+1} \cdot y_i\right)\right]\right] \end{array} \right. \\ \left|A_{pp} \cdot \left(C_{pp} - PNA_{x;y}\right)\right| + \left|A_{np} \cdot \left(C_{np} - PNA_{x;y}\right)\right| \end{array} \right.$$

plastic section modulus (x)

$$W_{pl;x} := W_{pl}\left(polygon, PNA_{y_1}, \text{"y"}\right)$$

plastic section modulus (y)

$$W_{pl;y} := W_{pl}\left(polygon, PNA_{x_1}, \text{"x"}\right)$$

plastic neutral axes (relative to global coordinates x,y)

$PNA_x := PNA_x \cdot (1)$

$PNA_y := PNA_y \cdot (1)$

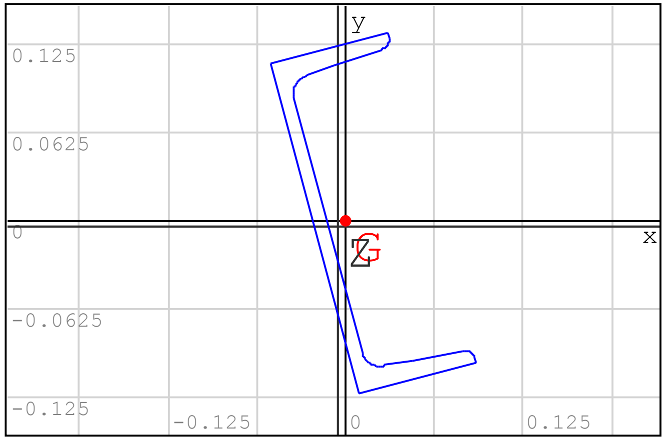
plastic neutral axes (relative to centroid)

$PNA_X := PNA_x - C_x$

$PNA_Y := PNA_y - C_y$

PLASTIC SECTION MODULUS

plastice neutral axes center/region



plastic neutral axes (relative to global coordinates x,y)

$PNA_x = 15.92\ mm$ $PNA_y = 1.23\ mm$

plastic neutral axes (relative to centroid)

$PNA_X = -5.68\ mm$ $PNA_Y = -4.56\ mm$

plastic section modulus

$W_{pl;x} = 345.42\ cm^3$ $W_{pl;y} = 111.7\ cm^3$

note: "PNAs and plastic section modulus calculated for equal areas"