

FreeCAD World

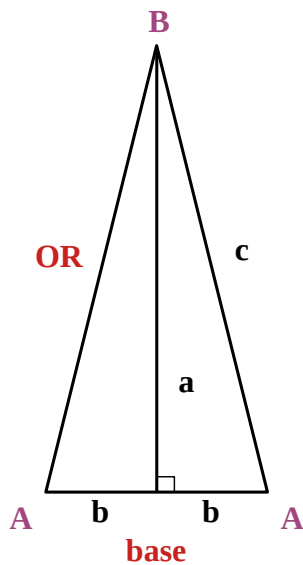
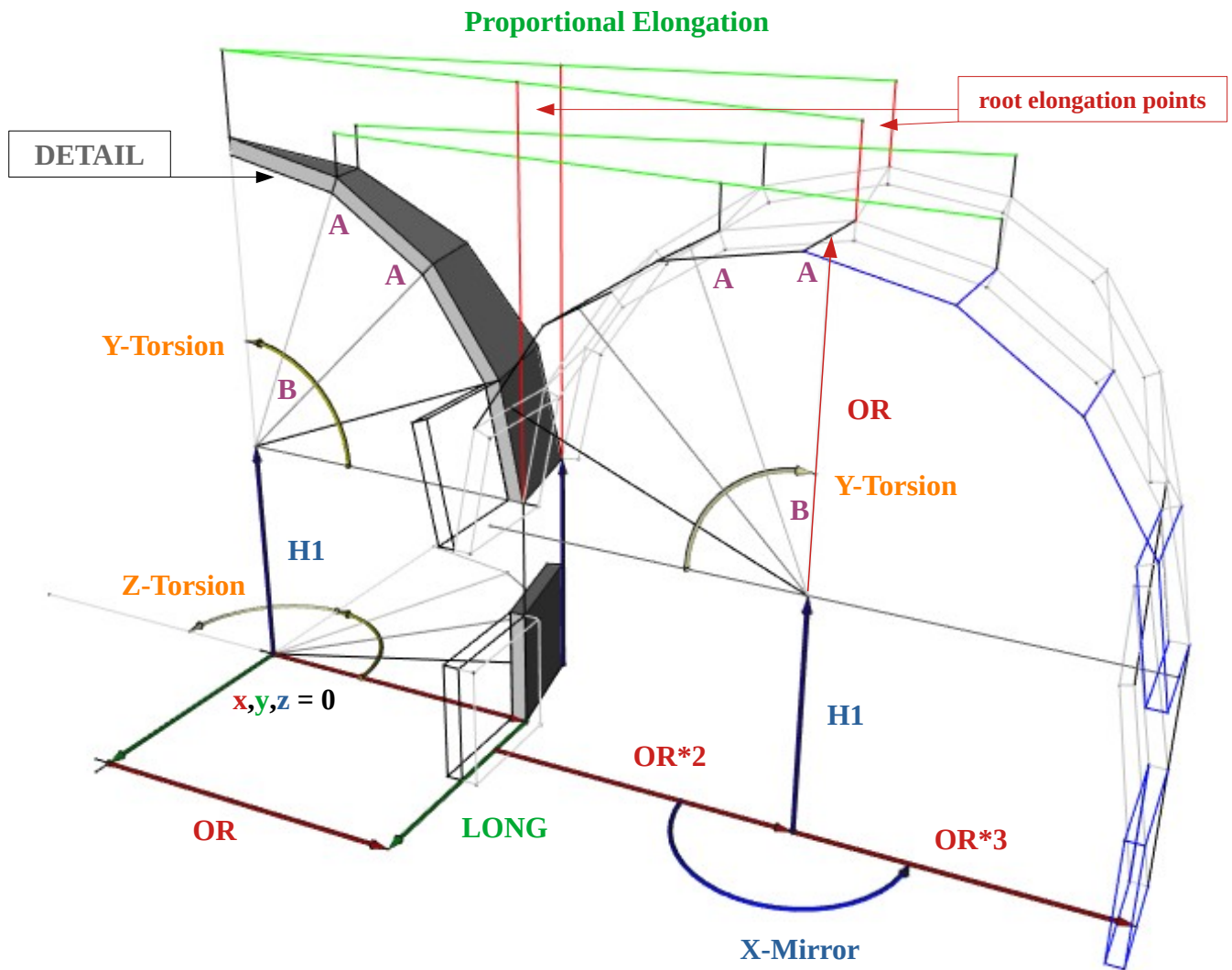
[notes]

Config types

[all types and values (except **str("eg. Value or valUe")**) are **case sensitive**]

- **int():** integers: (1, 2, 15, 144, 2048, etc);
- **float():** floating point numbers: (1.2, 2.3, 15.123, 144.9, 2048.1024, $\frac{3}{4}$, $\frac{1}{2}$, etc);
- **bool():** boolean **True** or **False**, 1 or 0. (In real life is **On/Off**);
- **tuple():** in config used as **comma separated** tuple of integers or floats;
- **str():** string as any human readable words, eg **str("CORNER")**;
- **dict():** **comma separated** dictionary of options: str(key) = value, where value may be **one** of described above types;

The Principle



$$B = 360^\circ / \text{DETAILS}$$

$$A + B/2 = 90^\circ$$

$$A = (180^\circ - B)/2$$

$$a = OR * \sin(A) = \text{cathetus}$$

$$b = OR * \cos(A) = \text{side } b$$

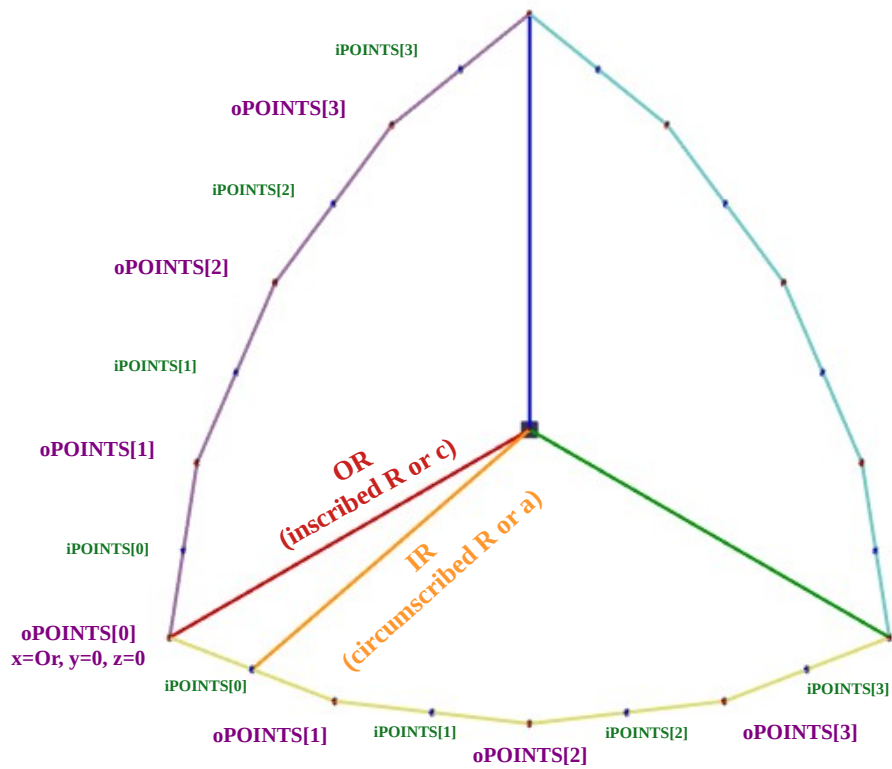
$$OR = a / \cos(B/2) = c = \text{hypotenuse}$$

$$\text{base} = b * 2 = OR * 2 * \cos(A)$$

**Isosceles Triangle
as base of
construction**

Pointing

- **oPOINTS;**
- **iPOINTS;**



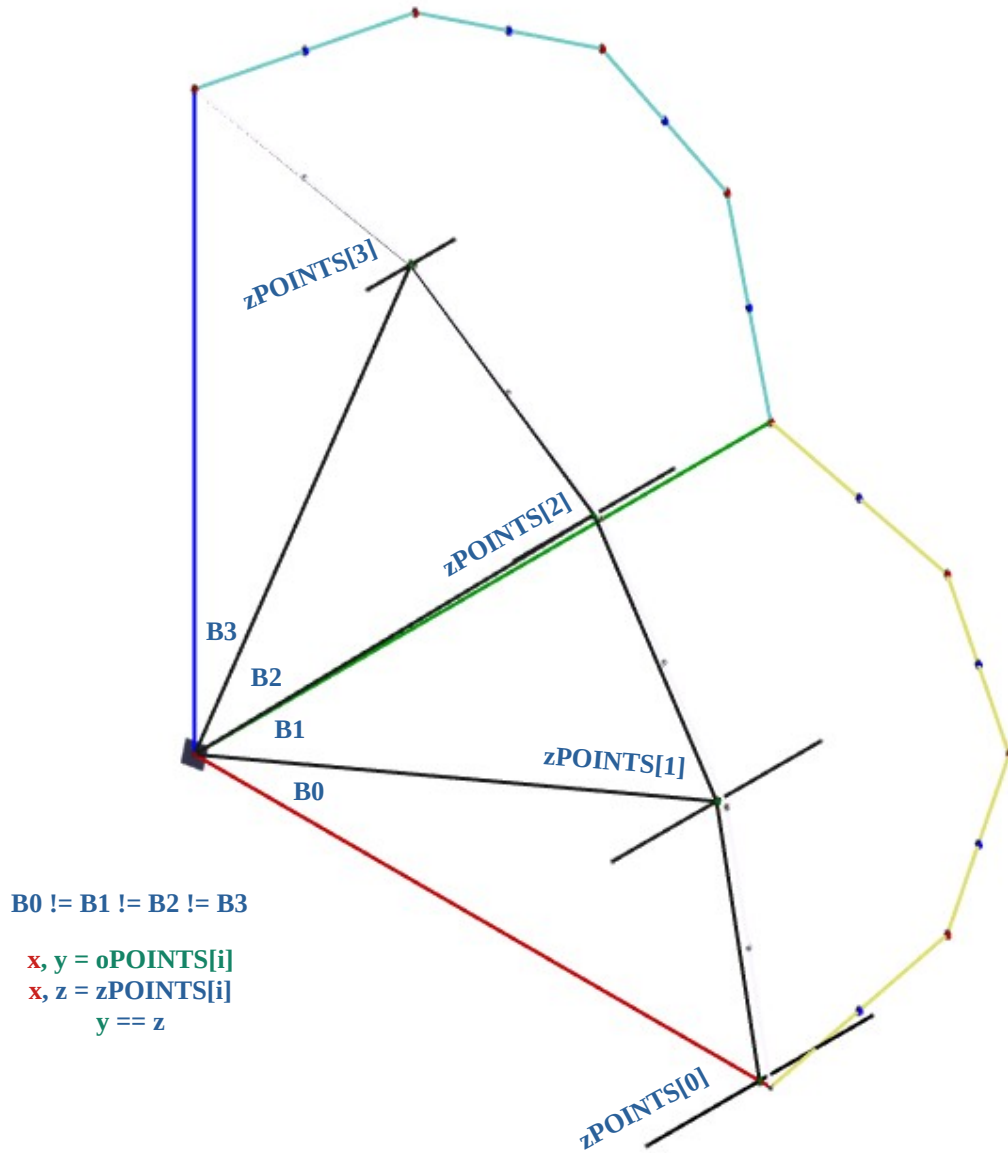
See:

- **clockWiseArray(x, y);**
- **oi_pointing.FCStd**

```
class PairOfCompasses(Trigon):
    _deviation_ = float(0.01)

    def clockwiseArray(self, x, y):
        """
            Clockwise bottom view
        """
        A = self.DETAILED_A
        B = self.DETAILED_B
        """
            Extra solution for thorax to find proportions
            on y axis of higher 'horizon poly wireframe'.
            System-wide single deviation is:
                x == 0.01 mm if 0 on x & y axis
                in case of self.DETAILED % 4 == 0
        """
        if x < self.DEVIATION: x = self.DEVIATION
        base = self.isoscelesBase(x)
        points = list([ [ x, y ] ])
        for i in range(self.POLY_QUARTER):
            x -= self.rightSideB_ByA(base, A)
            y += self.rightCathetusA_ByA(base, A)
            points.append([ x, y ])
            A -= B
        return points
```

- **zPOINTS;**



B0 != B1 != B2 != B3

```
x, y = oPOINTS[i]
x, z = zPOINTS[i]
y == z
```

See:

- **zPolyPoints(Or);**
- **z_pointing.FCStd;**

```
def zPolyPoints(self, Or):
    """
    Receives: OR of inscribed into circle polygon;
    Returns: sequence of mid x, z points of base
    of isosceles triangle on y=0 and ZERO_Z=0;
    """
    oPOINTS = self.oPolyPoints(Or)
    points = list()
    for x, z in oPOINTS:
        x, y = self.firstSectionToCircumscribed(x, z)
        x, y = self.circumscribedCounterClockWiseOnce(x, y)
        points.append([ x, z ])
    return points
```