Tetraeder

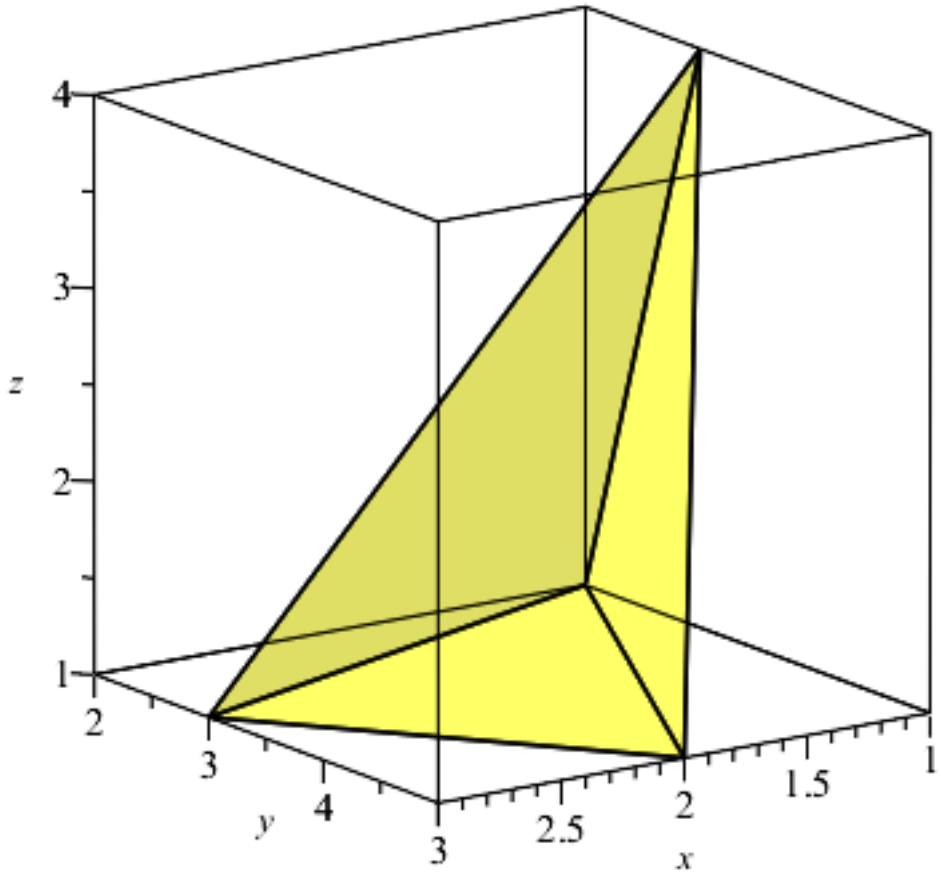
```
restart
with(plots) :
with(plottools) :
```

Tegning af tetraederet

De 4 hjørner i tetraederet defineres (koordinaterne som lister af hensyn til tegningen):

```
P := [1, 2, 1] : Q := [3, 3, 1] : R := [2, 5, 1] : S := [1, 3, 4] :
```

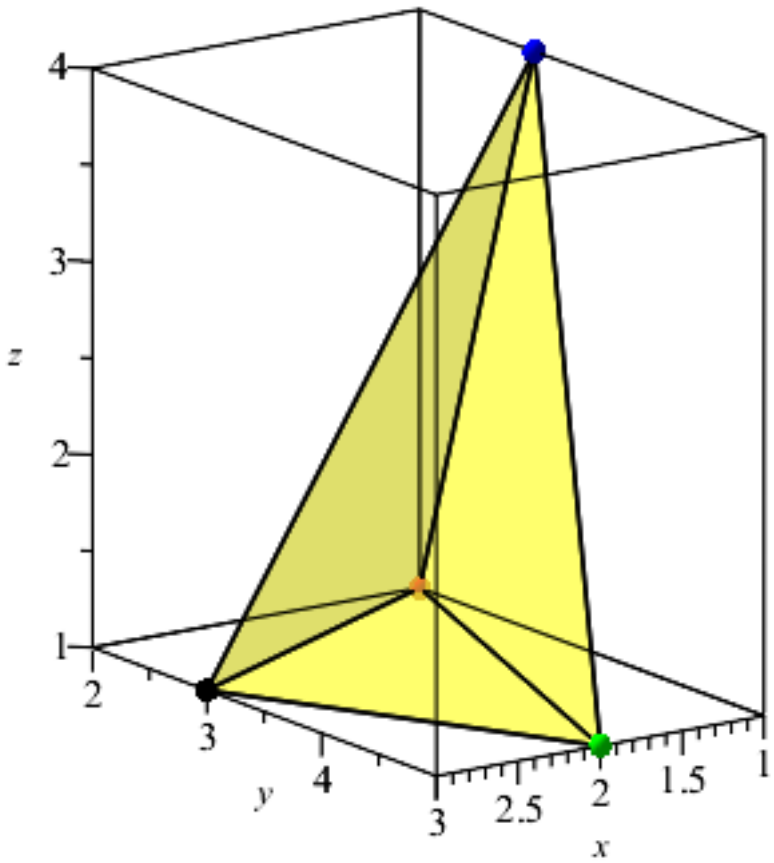
```
Tetra := display(tetrahedron([P, Q, R, S]), color = yellow, transparency = 0.5, labels = [x, y, z])
```




```
P1 := pointplot3d([P], color = red, symbol = solidsphere, symbolsize = 20) :
Q1 := pointplot3d([Q], color = black, symbol = solidsphere, symbolsize = 20) :
R1 := pointplot3d([R], color = green, symbol = solidsphere, symbolsize = 20) :
S1 := pointplot3d([S], color = blue, symbol = solidsphere, symbolsize = 20) :
```

Illustration med farvede hjørner, så man kan holde styr på trekanterne:

```
F := display(Tetra, P1, Q1, R1, S1, scaling = constrained)
```



NB: P er **rød**, Q er **sort**, R er **grøn**, S er **blå**.

Generering af en binær STL-fil ("tera1.stl")

Export("tetra1.stl", *Tetra*, *base* = *homedir*) = 284

Dvs. STL-filen fylder blot 284 bytes.

Tjek af trekanterne i STL-filen

For at kunne vise en større matrix skrives koden:
interface(*rtablesize* = 11) :

Indlæsning af trekanterne i STL-filen:

Import("tetra1.stl", *base* = *homedir*, *output* = *triangles*) =

	x1	*y1*	*z1*	*x2*	*y2*	*z2*	*x3*	*y3*	*z3*
1	1.	2.	1.	3.	3.	1.	2.	5.	1.
2	1.	2.	1.	3.	3.	1.	1.	3.	4.
3	1.	2.	1.	2.	5.	1.	1.	3.	4.
4	3.	3.	1.	2.	5.	1.	1.	3.	4.