Volume3D programming examples

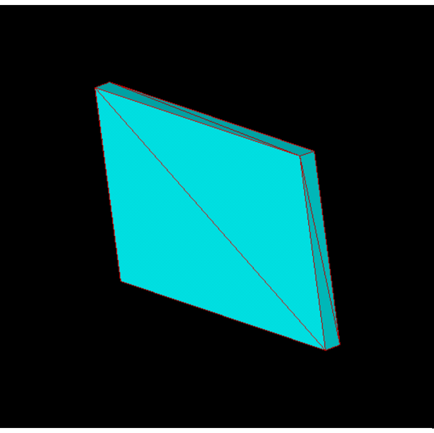
Principal sequence of Volume3D programs

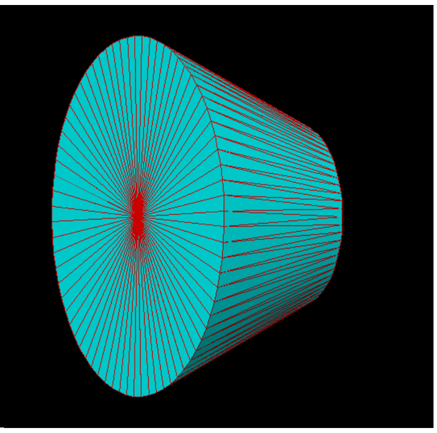
1. generate or import 3D models
2. manipulate 3D models
3. export 3D models

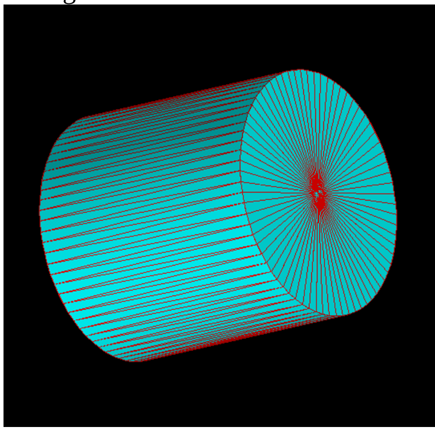
Model generation:

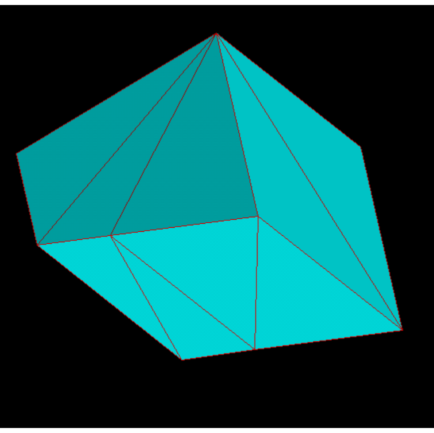
Basic shapes:

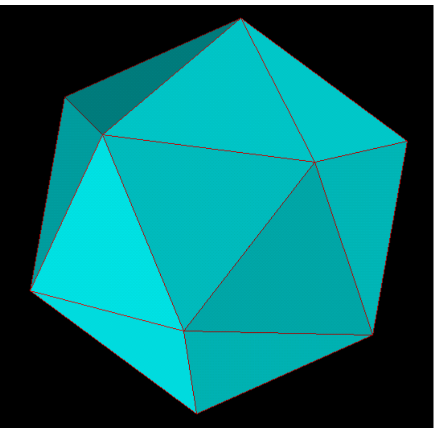
Triangle

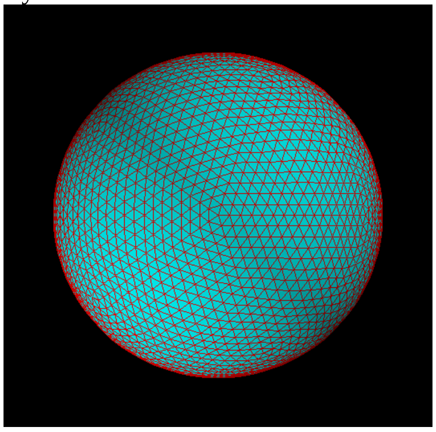

Box


Cone


Cylinder


Prism


Icosaeder


Sphere

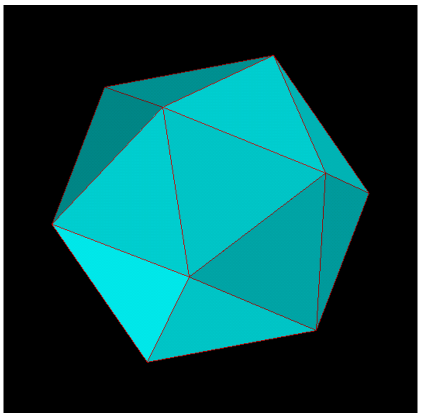
Generation is easy:

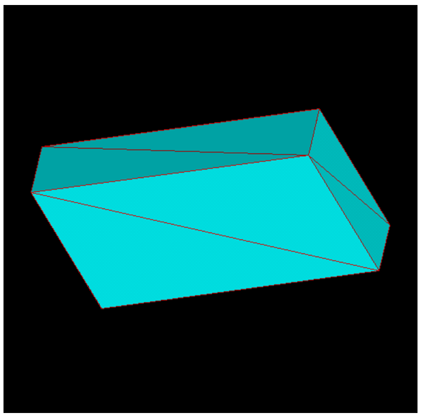
```
var
  Traingle,Box,Cone,Cylinder,Prism,Icosaeder,Sphere:pshape_3d;
  Polygon:ppoly;
begin
  Triangle:=triangle(0,0,0,10,0,0,10,10,0,’Dreieck’,100);

  Box:=box(0,0,0,10,2,6,’Quader’,1000);

  Cone:=cone(5,10,10,’Kegel’,2000);

  Cylinder:=cylinder(5,10,’Zylinder’,2000);

  Polygon:=create_poly(5);
  Polygon^[0]:=point(0,0,0);
  Polygon^[1]:=point(10,0,0);
  Polygon^[2]:=point(5,10,0);
  Polygon^[3]:=point(0,10,0);
  Polygon^[4]:=point(0,0,0);
  Prism:=prism(0,10,5,poly,’Prisma’,100);
  Polygon:=free_poly(5);

  Icosaeder:=icosaeder(5,’Ikosaeder’,1000);

  Sphere:=sphere(5,’Kugel’,10000);

{ manipulation functions }

  dispose(Sphere,done);
  dispose(Icosaeder,done);
  dispose(Prism,done);
  dispose(Cylinder,done);
  dispose(Cone,done);
  dispose(Box,done);
  dispose(Triangle,done);
end.
```

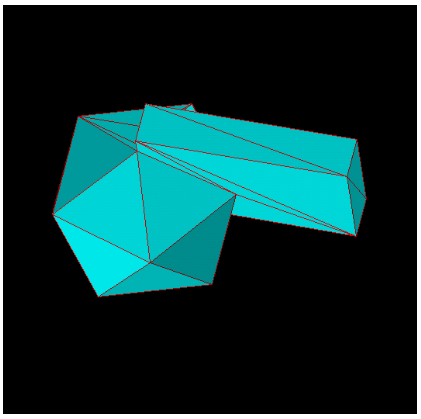
Model import

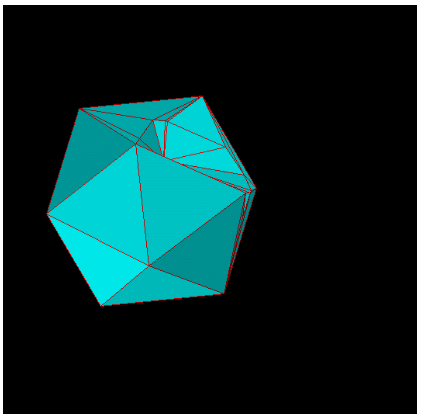
```
var
  Box,Icosaeder:pshape_3d;
begin
  Box:=import_gts(’Box.gts’);

{  be careful: gts has no proper facet orientation}

  Icosaeder:=import_stl(’Ikosaeder.stl’,1000);

{ manipulation functions }

  dispose(Icosaeder,done);
  dispose(Box,done);
end.
```

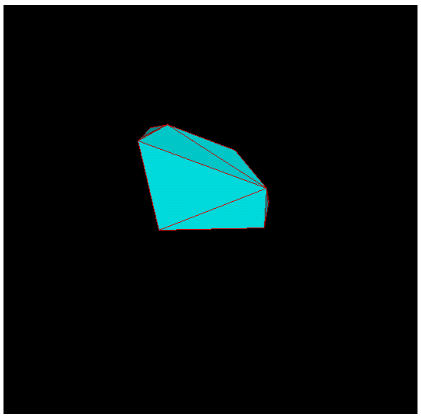
Manipulation (boolean operations)

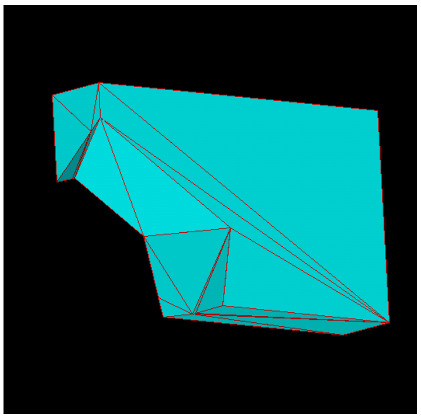

Icosaeder


Box

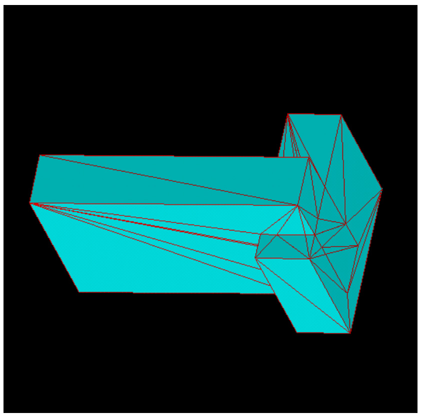
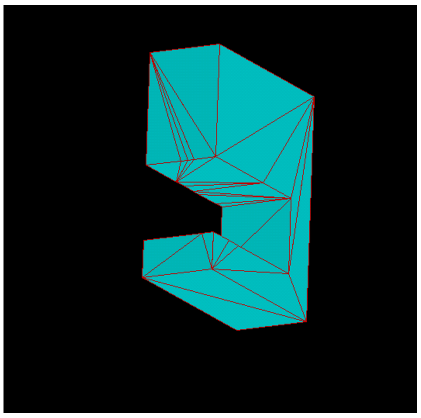
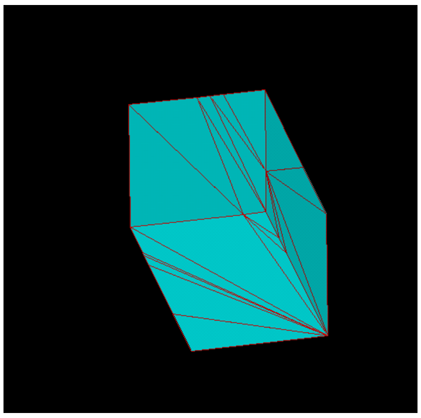
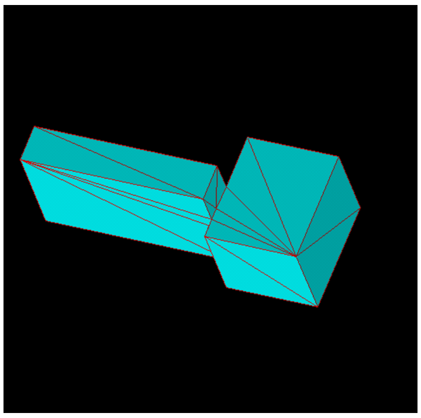

Add:=add_shape(icosaeder,box);


Sub:=sub_shape(icosaeder,box);

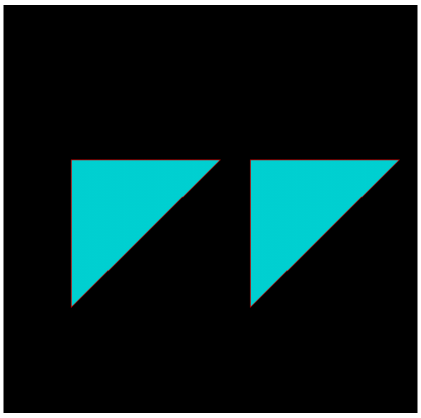

Inter:=inter_shape(icosaeder,box);


Sub_:=sub_shape(box,icosaeder);

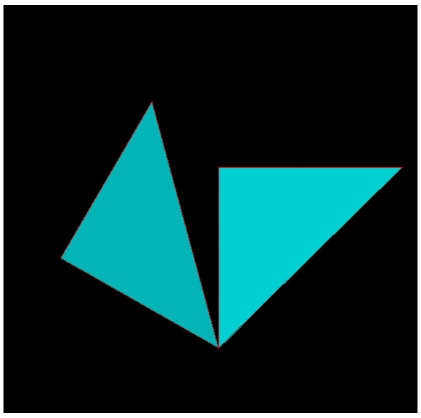
Special cases with identic surface areas:

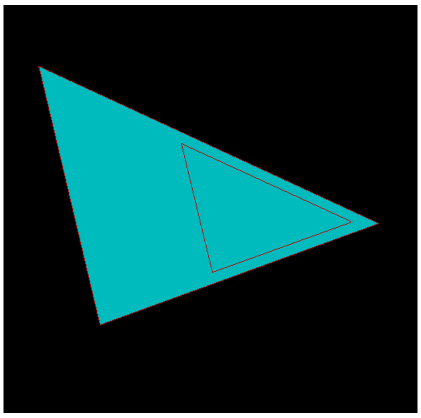
Transformations


Translation

```
tri[1]:=triangle(0,0,0,0,10,5,10,10,5,’Dreieck_1’,100);
tri[2]:=triangle(0,0,0,0,10,5,10,10,5,’Dreieck_2’,100);
translate(12,0,0,tri[2]);
tri[1]^.include_triangles(tri[2]);
export_gts(tri[1],’Translat.gts’);
```


Rotation

```
tri[1]:=triangle(0,0,0,0,10,5,10,10,5,’Dreieck_1’,100);
tri[2]:=triangle(0,0,0,0,10,5,10,10,5,’Dreieck_2’,100);
rotate(0,0,0,0,0,1,PI/3,tri[2]);
tri[1]^.include_triangles(tri[2]);
export_gts(tri[1],’Rotat.gts’);
```


Scaling

```
tri[1]:=triangle(0,0,0,0,10,5,10,10,5,’Dreieck_1’,100);
tri[2]:=triangle(0,0,0,0,10,5,10,10,5,’Dreieck_2’,100);
rotate(0,0,2.5,2,tri[2]);
tri[1]^.include_triangles(tri[2]);
export_gts(tri[1],’Scale.gts’);
```

Model export

```
var
  MyModel:pshape_3d;
begin

{  generation of terrific 3D model}

  export_gts(MyModel’Mymodel.gts’);
  export_stl(MyModel,’Mymodel.stl’);

  dispose(MyModel,done);
end.
```