Supported filetypes with ASSIMP package

Collada (*.dae;*.xml)
Blender (*.blend) 3
Biovision BVH (*.bvh)
3D Studio Max 3DS (*.3ds)
3D Studio Max ASE (*.ase)
Wavefront Object (*.obj)
Stanford Polygon Library (*.ply)
AutoCAD DXF (*.dxf)
IFC-STEP, Industry Foundation Classes (*.ifc)
Neutral File Format (*.nff)
Sense8 WorldToolkit (*.nff)
Valve Model (*.smd,*.vta) 3
Quake I (*.mdl)
Quake II (*.md2)
Quake III (*.md3)
Quake 3 BSP (*.pk3) 1
RtCW (*.mdc)
Doom 3 (*.md5mesh;*.md5anim;*.md5camera)
DirectX X (*.x).
Quick3D (*.q3o;q3s)

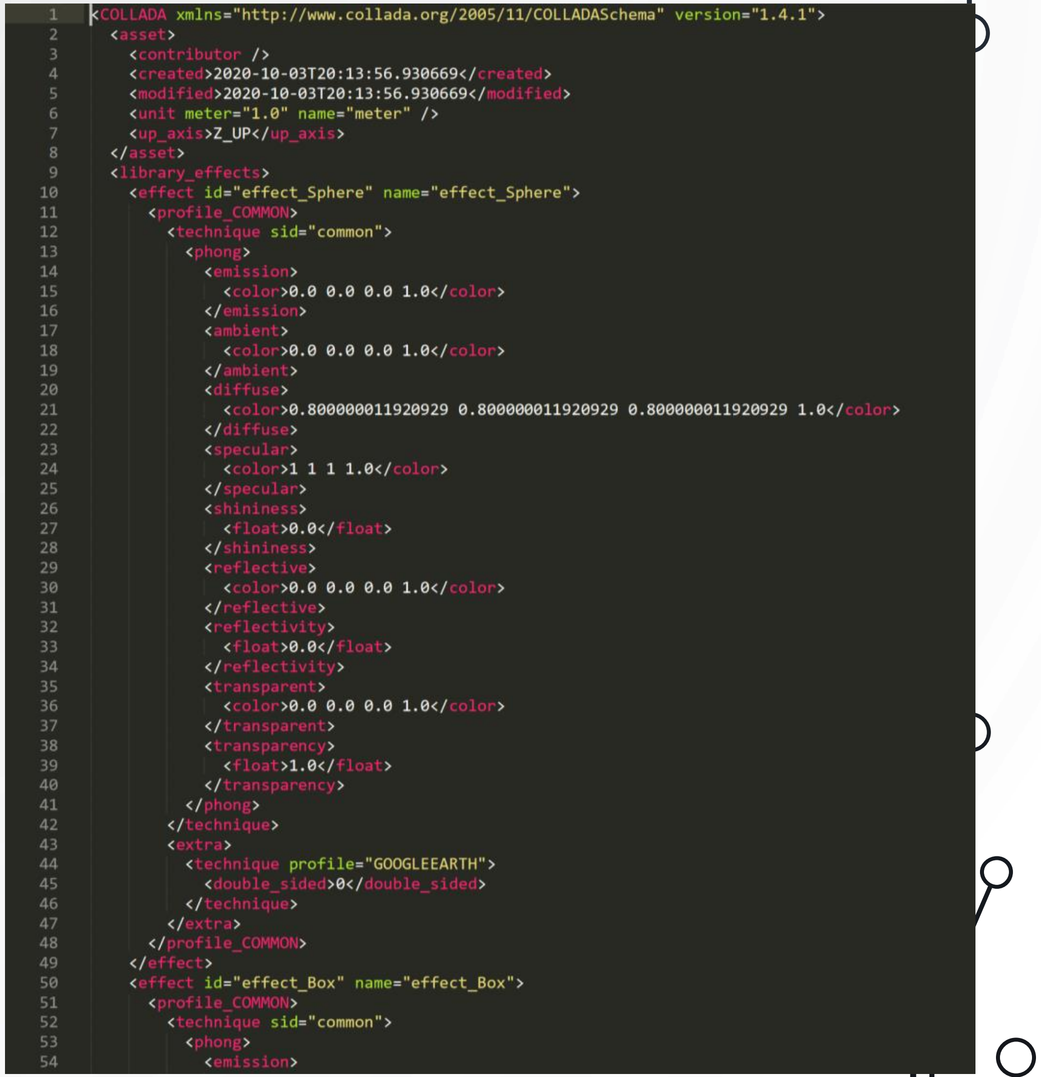
Raw Triangles (.raw)
AC3D (*.ac)
Stereolithography (*.stl)
Autodesk DXF (*.dxf)
Irrlicht Mesh (*.irrmesh;*.xml)
Irrlicht Scene (*.irr;*.xml).
Object File Format (*.off).
Terragen Terrain (*.ter)
3D GameStudio Model (*.mdl)
3D GameStudio Terrain (*.hmp)
Ogre (*.mesh.xml, *.skeleton.xml, *.material)3
Milkshape 3D (*.ms3d)
LightWave Model (*.lwo)
LightWave Scene (*.lws)
Modo Model (*.lxo)
CharacterStudio Motion (*.csm)
Stanford Ply (*.ply)
TrueSpace (*.cob, *.scn)
XGL (*.xgl, *.zgl)

Importing CAD files in DD4hep

- Start from a very simple CAD file, maybe collada export (.dae) – similar to xml
- Looking into assimp package examples
- *Would like to try out a few different filetypes:*

  STL, **STEP** → **DAE**, **3DXML**, **IFC**, **XAML**

Sample Collada (.dae) export



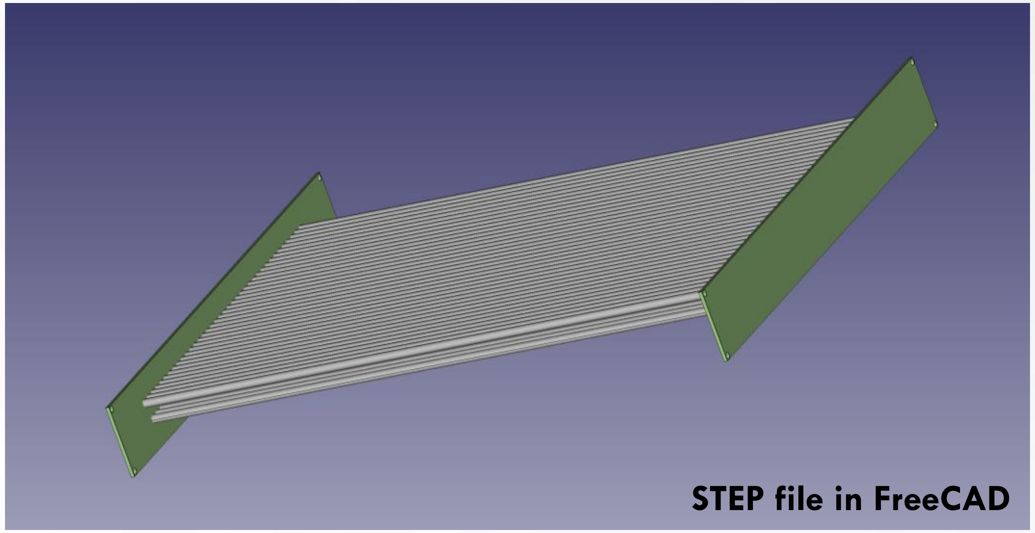
```
<COLLADA xmlns="http://www.collada.org/2005/11/COLLADASchema" version="1.4.1">
  <asset>
    <contributor />
    <created>2020-10-03T20:13:56.930669</created>
    <modified>2020-10-03T20:13:56.930669</modified>
    <unit meter="1.0" name="meter" />
    <up_axis>Z_UP</up_axis>
  </asset>
  <library_effects>
    <effect id="effect_Sphere" name="effect_Sphere">
      <profile_COMMON>
        <technique sid="common">
          <phong>
            <emission>
              <color>0.0 0.0 0.0 1.0</color>
            </emission>
            <ambient>
              <color>0.0 0.0 0.0 1.0</color>
            </ambient>
            <diffuse>
              <color>0.800000011920929 0.800000011920929 0.800000011920929 1.0</color>
            </diffuse>
            <specular>
              <color>1 1 1 1.0</color>
            </specular>
            <shininess>
              <float>0.0</float>
            </shininess>
            <reflective>
              <color>0.0 0.0 0.0 1.0</color>
            </reflective>
            <reflectivity>
              <float>0.0</float>
            </reflectivity>
            <transparent>
              <color>0.0 0.0 0.0 1.0</color>
            </transparent>
            <transparency>
              <float>1.0</float>
            </transparency>
          </phong>
        </technique>
        <extra>
          <technique profile="GOOGLEEARTH">
            <double_sided>0</double_sided>
          </technique>
        </extra>
      </profile_COMMON>
    </effect>
    <effect id="effect_Box" name="effect_Box">
      <profile_COMMON>
        <technique sid="common">
          <phong>
            <emission>
```

Importing CAD geometry in DD4hep

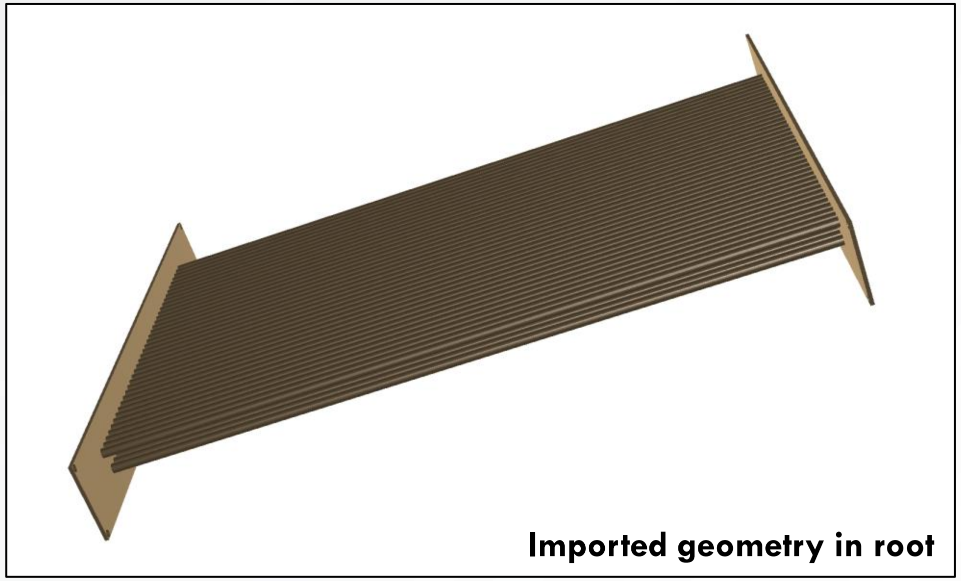
- Open .STEP file in FreeCAD software and export selected parts to a Collada .dae file.
- Import this .dae file (detector assembly) in DD4hep with DDCAD and then, ***root*** may be used for visualization.

In FreeCAD (STEP file) → Collada (.dae)

Imported in DD4hep → Visualize with ***root***



STEP file in FreeCAD



Imported geometry in root

Import & Display CAD Geometry in DD4hep/DDCAD

test.xml (simple layout):

```
<lccdd>
  <includes>
    <gdmlFile
ref="../../ClientTests/compact/CheckShape.xml"/>
  </includes>
  <detectors>
    <detector id="1" name=“HMB_STL"
type="DD4hep_TestShape_Creator">
      <check vis=“T1_Fiber">
        <shape type="CAD_Assembly"
ref="/mnt/hgfs/VMShared/HMB/allGeo.stl"/>
      </check>
    </detector>
  </detectors>
</lccdd>
```

`<lccdd>`

`<info> ... </info>` Auxiliary detector model information

`<includes> ... </includes>` Section defining GDML files to be included

`<define> ... </define>` Dictionary of constant expressions and variables

`<materials> ... </materials>` Additional material definitions

`<display> ... </display>` Definition of visualization attributes

`<detectors> ... </detectors>` Section with sub-detector definitions

`<readouts> ... </readouts>` Section with readout structure definitions

`<limits> ... </limits>` Definition of limit sets for Geant4

`<fields> ... </fields>` Field definitions

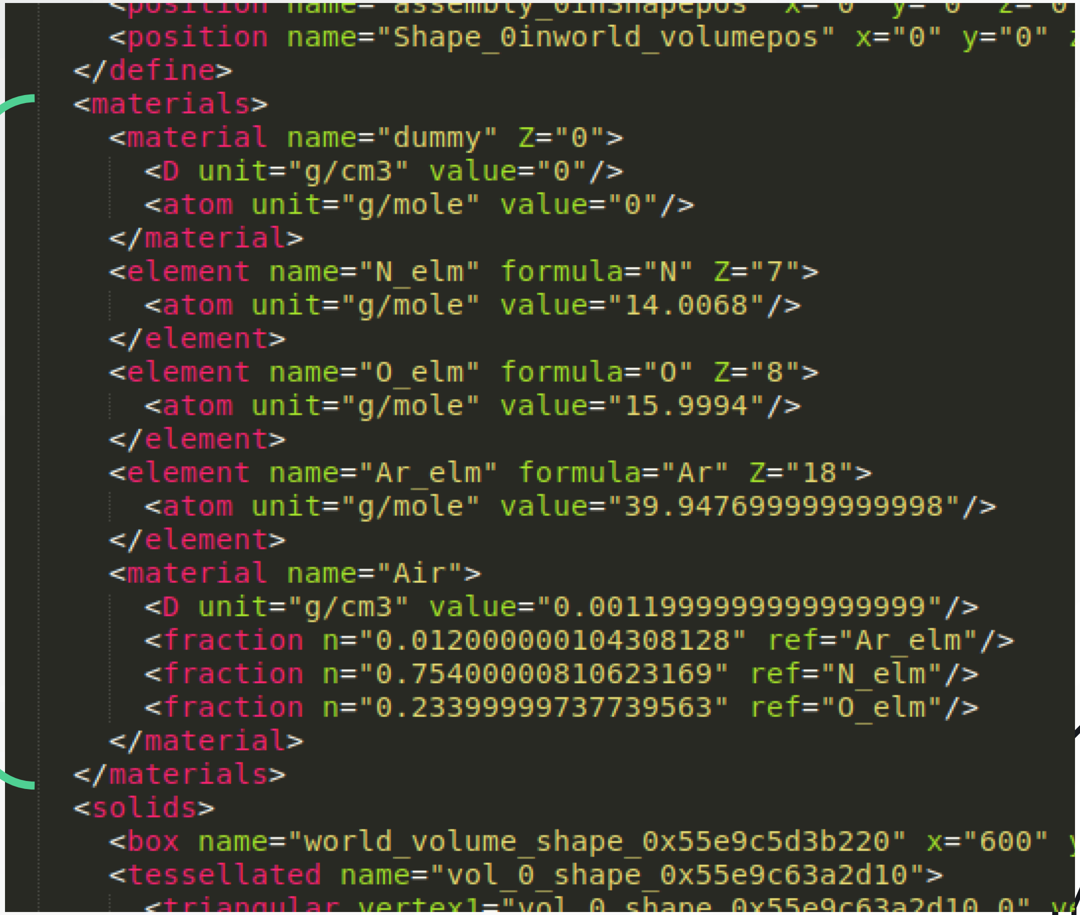
`</lccdd>`

```
geoDisplay –compact test.xml
```

CAD drawings from STL → GDML

geoConverter -compact2gdml -input test.xml -output test.gdml

- GDML: XML like syntax, compatible with Geant4
- This can be used in an independent Geant4 simulation (independent of DD4hep)
- Material info in CAD files, perhaps lost in translation! Does STL have material info?
- The resulting GDML file might not be efficiently processed with Geant4 though, especially for complex detector geometries.



```
    <position name="assembly_0inShapepos" x="0" y="0" z="0
    <position name="Shape_0inworld_volumepos" x="0" y="0" z
  </define>
  <materials>
    <material name="dummy" Z="0">
      <D unit="g/cm3" value="0"/>
      <atom unit="g/mole" value="0"/>
    </material>
    <element name="N_elm" formula="N" Z="7">
      <atom unit="g/mole" value="14.0068"/>
    </element>
    <element name="O_elm" formula="O" Z="8">
      <atom unit="g/mole" value="15.9994"/>
    </element>
    <element name="Ar_elm" formula="Ar" Z="18">
      <atom unit="g/mole" value="39.947699999999998"/>
    </element>
    <material name="Air">
      <D unit="g/cm3" value="0.0011999999999999999"/>
      <fraction n="0.012000000104308128" ref="Ar_elm"/>
      <fraction n="0.75400000810623169" ref="N_elm"/>
      <fraction n="0.23399999737739563" ref="O_elm"/>
    </material>
  </materials>
  <solids>
    <box name="world_volume_shape_0x55e9c5d3b220" x="600" y
    <tessellated name="vol_0_shape_0x55e9c63a2d10">
      <triangular vertex1="vol_0_shape_0x55e9c63a2d10_0" ve
```

WHAT'S NEXT?

- Import GDML in Geant4 – Check for efficiency, compare with previous simulation where geometry was made directly in Geant4
- Replicate CAD Tracker geometry (x4) and get started with DDG4
- Integrate CRY library with DDG4/DD4hep and Geant4