STL to DXF, OBJ to DXF

Tools

- https://sourceforge.net/projects/ivcon-tl
- Blender (can natively export STL to 3D DXF - this is compatible to use with DXF 2 Papercraft)

Blender AutoCAD DXF Export Bugfix (`AttributeError: module 'time' has no attribute 'clock'` in Python 3.8):

```
/usr/share/blender/scripts/addons/io_export_dxf# cat export_dxf.py
# ##### BEGIN GPL LICENSE BLOCK #####
#
#  This program is free software; you can redistribute it and/or
#  modify it under the terms of the GNU General Public License
#  as published by the Free Software Foundation; either version 2
#  of the License, or (at your option) any later version.
#
#  This program is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#  GNU General Public License for more details.
#
#  You should have received a copy of the GNU General Public License
#  along with this program; if not, write to the Free Software Foundation,
#  Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.
#
# ##### END GPL LICENSE BLOCK #####

import os
import mathutils

DEBUG = os.environ.get('BLENDER_DEBUG', False) #activates debug mode
if DEBUG:
    import sys
    sys.path.append(os.environ['PYDEV_DEBUG_PATH'])
```

```
    import pydevd

from .model.migiusModel import MigiusDXFLibDrawing

SUPPORTED_TYPES = ('MESH')#,'CURVE','EMPTY','TEXT','CAMERA','LIGHT')

def exportDXF(context, filePath, settings):
    """
    Main entry point into export facility.
    """
    print("----------\nExporting to {}".format(filePath))
    from time import perf_counter as timestamp
    time1 = timestamp()

    if settings['verbose']:
        print("Generating Object list for export... (Root parents only)")

    scene = context.scene

    if settings['onlySelected'] is True:
        objects = (ob for ob in scene.objects if not ob.hide_viewport and ob.select_get() and
ob.type in SUPPORTED_TYPES)
    else:
        objects = (ob for ob in scene.objects if not ob.hide_viewport and ob.type in
SUPPORTED_TYPES)

    if DEBUG: pydevd.settrace()
    mw = get_view_projection_matrix(context, settings)

    try:
        # add Entities --------------------
        #todo: fixme: seems to be the reason for missing BLOCK-export
        #if APPLY_MODIFIERS: tmp_me = Mesh.New('tmp')
        #else: tmp_me = None

        drawing = MigiusDXFLibDrawing()
        exported = 0
        for o in objects:
            if _exportItem(context, o, mw, drawing, settings):
                exported +=1
```

```
        if not drawing.isEmpty():
            # NOTE: Only orthographic projection used now.
    #       if PERSPECTIVE: # generate view border - passepartout
    #           from .primitive_exporters.viewborder_exporter import ViewBorderDXFExporter
    #           e = ViewBorderDXFExporter(settings)
    #           e.export(drawing, ob, mx, mw)

            drawing.convert(filePath)

        duration = timestamp() - time1
        print('%s objects exported in %.2f seconds. -----DONE-----' %\
            (exported, duration))
    except IOError:
        print('DXF Exporter: Write Error: ', filePath)
    except Exception as e:
        print('Nothing exported. Error: %s' % str(e))

    print("Finished")

#------------------------------------------------
def getCommons(ob, settings):
    """set up common attributes for output style:
     color=None
     extrusion=None
     layer='0',
     lineType=None
     lineTypeScale=None
     lineWeight=None
     thickness=None
     parent=None
    """

    BYBLOCK=0 #DXF-attribute: assign property to BLOCK defaults
    BYLAYER=None #256 #DXF-attribute: assign property to LAYER defaults
    LAYERNAME_DEF='' #default layer name
    LAYERCOLOR_DEF=7 #default layer color index
    LAYERLTYPE_DEF=0 #'CONTINUOUS' - default layer lineType
    ENTITYLAYER_DEF=LAYERNAME_DEF #default entity color index
    ENTITYCOLOR_DEF=BYLAYER #default entity color index
    ENTITYLTYPE_DEF=BYLAYER #default entity lineType
```

```
    #layers = ob.layers #gives a list e.g.[1,5,19]
    layers = ob.users_collection
    if layers: ob_layer_nr = layers[0]
    if DEBUG: print('ob_layer_nr=', ob_layer_nr) #--------------

    materials = ob.material_slots
    if materials:
        ob_material = materials[0]
        ob_mat_color = ob_material.material.diffuse_color
    else: ob_mat_color, ob_material = None, None
    if DEBUG:
        print('ob_mat_color, ob_material=', ob_mat_color, ob_material) #--------------

    data_materials = ob.material_slots
    if data_materials:
        data_material = data_materials[0]
        data_mat_color = data_material.material.diffuse_color
    else: data_mat_color, data_material = None, None
    if DEBUG:
        print('data_mat_color, data_material=', data_mat_color, data_material) #--------------

    entitylayer = ENTITYLAYER_DEF
    c = settings['entitylayer_from']

#["default_LAYER","obj.name","obj.layer","obj.material","obj.data.name","obj.data.material",".
.vertexgroup","..group","..map_table"]
    if c=="default_LAYER":
        entitylayer = LAYERNAME_DEF
    elif c=="obj.layer" and ob_layer_nr:
        entitylayer = 'LAYER'+ str(ob_layer_nr)
    elif c=="obj.material" and ob_material:
        entitylayer = ob_material.name
    elif c=="obj.name":
        entitylayer = ob.name
    elif c=="obj.data.material" and ob_material:
        entitylayer = data_material.name
    elif c=="obj.data.name":
        entitylayer = ob.data.name

    entitycolor = ENTITYCOLOR_DEF
    cfrom = settings['entitycolor_from']
```

```
    if cfrom=="default_COLOR":
        entitycolor = LAYERCOLOR_DEF
    elif cfrom=="BYLAYER":
        entitycolor = BYLAYER
    elif cfrom=="BYBLOCK":
        entitycolor = BYBLOCK
    elif cfrom=="obj.layer" and ob_layer_nr:
        entitycolor = ob_layer_nr
    elif cfrom=="obj.color" and ob.color:
        entitycolor = ob.color
    elif cfrom=="obj.material" and ob_mat_color:
        entitycolor = ob_mat_color
    elif cfrom=="obj.data.material" and data_mat_color:
        entitycolor = data_mat_color

    entityltype = ENTITYLTYPE_DEF
    etype = settings['entityltype_from']
    if etype=="default_LTYPE":
        entityltype = LAYERLTYPE_DEF
    elif etype=="BYLAYER":
        entityltype = BYLAYER
    elif etype=="BYBLOCK":
        entityltype = BYBLOCK
    elif etype:
        entityltype = etype

    return entitylayer, entitycolor, entityltype

def getCameraMatrix(cam):
    raise NotImplementedError()
#   camProps = cam.data
#       mc0 = act_camera.matrix.copy()
#               #print 'deb: camera.Matrix=\n', mc0 #------------------
#               camera = Camera.Get(act_camera.getData(name_only=True))
#               #print 'deb: camera=', dir(camera) #------------------
#               if camera.type=='persp': PERSPECTIVE = 1
#               elif camera.type=='ortho': PERSPECTIVE = 0
#               # mcp is matrix.camera.perspective
#               clip_box, mcp = getClipBox(camera)
##              if PERSPECTIVE:
##                  # get border
```

```
##                  # lens = camera.lens
##                  min_X1, max_X1, min_Y1, max_Y1,\
##                  min_X2, max_X2, min_Y2, max_Y2,\
##                      min_Z, max_Z = clip_box
##                  verts = []
##                  verts.append([min_X1, min_Y1, min_Z])
##                  verts.append([max_X1, min_Y1, min_Z])
##                  verts.append([max_X1, max_Y1, min_Z])
##                  verts.append([min_X1, max_Y1, min_Z])
##                  border=verts
#               mw = mc0.copy().invert()
#           #ViewVector = mathutils.Vector(Window.GetViewVector())
#           #print 'deb: ViewVector=\n', ViewVector #------------------
#           #TODO: what is Window.GetViewOffset() for?
#           #print 'deb: Window.GetViewOffset():', Window.GetViewOffset() #---------
#           #Window.SetViewOffset([0,0,0])
#           mw0 = Window.GetViewMatrix()
#           #print 'deb: mwOrtho    =\n', mw0    #---------
#           mwp = Window.GetPerspMatrix() #TODO: how to get it working?
#           #print 'deb: mwPersp    =\n', mwp    #---------
#           mw = mw0.copy()

projectionMapping = {
  'TOP' : mathutils.Vector((0, 0, -1)),
  'BOTTOM' : mathutils.Vector((0, 0, 1)),
  'LEFT' : mathutils.Vector((0, 1, 0)),
  'RIGHT' : mathutils.Vector((0, -1, 0)),
  'FRONT' : mathutils.Vector((-1, 0, 0)),
  'REAR' : mathutils.Vector((1, 0, 0))
}

#-----------------------------------------------------
def get_view_projection_matrix(context, settings):
    """
    Returns view projection matrix.
    Projection matrix is either identity if 3d export is selected or
    camera projection if a camera or view is selected.
    Currently only orthographic projection is used. (Subject to discussion).
    """
    cam = settings['projectionThrough']
    if cam is None:
```

```
        mw = mathutils.Matrix()
        mw.identity()
    elif cam in projectionMapping.keys():
        projection = mathutils.Matrix.OrthoProjection(projectionMapping[cam], 4)
        mw = projection
    else: # get camera with given name
        c = context.scene.collection.objects[cam]
        mw = getCameraMatrix(c)
    return mw

def _exportItem(ctx, o, mw, drawing, settings):
    """
    Export one item from export list.
    mw - modelview
    """
    if settings['verbose']: print('Exporting %s' % o)
    #mx = ob.matrix.copy()
    #print 'deb: ob =', ob   #---------
    #print 'deb: ob.type    =', ob.type  #---------
    #print 'deb: mx =\n', mx     #---------
    #print 'deb: mw0    =\n', mw0    #---------
    #mx_n is trans-matrix for normal_vectors for front-side faces
    mx = o.matrix_world
    viewRotation = mw.to_euler().to_matrix()
    mx_n = o.rotation_euler.to_matrix() @ viewRotation
    mx @= mw

    #mx_inv = mx.copy().invert()
    elayer, ecolor, eltype = getCommons(o, settings)
    if settings['verbose']:
        print('elayer=%s, ecolor=%s, eltype=%s' % (elayer, ecolor, eltype))
    #TODO: use o.boundBox for drawing extends ??

    if elayer is not None and not drawing.containsLayer(elayer):
        if ecolor is not None: tempcolor = ecolor
        else: tempcolor = settings['layercolor_def']
        drawing.addLayer(elayer, tempcolor)

    if DEBUG: pydevd.settrace()
    if (o.type == 'MESH') and settings['mesh_as']:
        from .primitive_exporters.mesh_exporter import MeshDXFExporter
```

```
        e = MeshDXFExporter(settings)
    elif (o.type == 'CURVE') and settings['curve_as']:
        from .primitive_exporters.curve_exporter import CurveDXFExporter
        e = CurveDXFExporter(settings)
    elif (o.type == 'EMPTY') and settings['empty_as']:
        from .primitive_exporters.empty_exporter import EmptyDXFExporter
        e = EmptyDXFExporter(settings)
    elif (o.type == 'TEXT') and settings['text_as']:
        from .primitive_exporters.text_exporter import TextDXFExporter
        e = TextDXFExporter(settings)
    elif (o.type == 'CAMERA') and settings['camera_as']:
        from .primitive_exporters.camera_exporter import CameraDXFExporter
        e = CameraDXFExporter(settings)
    elif (o.type == 'LIGHT') and settings['light_as']:
        from .primitive_exporters.light_exporter import LampDXFExporter
        e = LampDXFExporter(settings)

    return e.export(ctx, drawing, o, mx, mx_n, color=ecolor, layer=elayer, lineType=eltype)
```

Version #1
Erstellt: 2025-10-14 23:08:38 CEST von Mario Voigt
Zuletzt aktualisiert: 2025-10-14 23:10:01 CEST von Mario Voigt