In [1]:

```
%reload_ext autoreload
%autoreload 2
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:90% !important; }</style>"))

import matplotlib.pyplot as plt
%matplotlib widget

import sys
sys.path.insert(0, '/home/ruizhu/Documents/Projects/indoorInverse/train

from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt
import pickle
from utils_OR.utils_OR_imageops import loadHdr, loadImage, scaleHdr
```

In [2]:

```
scene_name = 'scene0270_02'
frame_idx = 5
meta_split = 'mainDiffLight_xml'

rendering_path = '/newfoundland2/ruizhu/siggraphasia20dataset/code/Rout
xml_path = '/newfoundland2/ruizhu/siggraphasia20dataset/code/Routine/sc
layout_path = '/newfoundland2/ruizhu/siggraphasia20dataset/layoutMesh'

frame_hdr_path = Path(rendering_path) / scene_name / ('im_%d.hdr'%frame
im_hdr = loadHdr(str(frame_hdr_path))
seg_file = str(frame_hdr_path).replace('im_', 'immask_').replace('hdr',
seg = 0.5 * (loadImage(seg_file) + 1)[0, :, :]
im, scale = scaleHdr(im_hdr, seg[:, :, np.newaxis])
# im = im_hdr
im_not_hdr = np.clip(im**(1.0/2.2), 0., 1.)
im_uint8 = (255. * im_not_hdr).astype(np.uint8)

fig = plt.figure()
ax = fig.gca()
ax.set_aspect("equal")
plt.imshow(im_uint8)
plt.show()

layout_file = Path(layout_path) / scene_name / ('%s_corners.npy'%scene_
corners = np.load(layout_file, allow_pickle=True)
print(corners.item())

layout_obj_file = Path(layout_path) / scene_name / 'uv_mapped.obj'
# layout_obj_file = Path(layout_path) / scene_name / ('%s_contour.obj'%
```


```
{'light_ctr': array([[3.07634198, 3.2431079 , 3.        , 0.4       ]]),
'type': ['d', 'w'], 'coords': [array([[3.36254814, 1.39612194, 0.
],
       [2.37256427, 1.24860778, 0.        ],
       [2.37256427, 1.24860778, 2.18917697],
       [3.36254814, 1.39612194, 2.18917697]]), array([[6.43443785, 5.5737
862 , 0.96130801],
       [6.8908329 , 3.55025155, 0.96130801],
       [6.8908329 , 3.55025155, 2.18917697],
       [6.43443785, 5.5737862 , 2.18917697]])]}
```

visualize room layout in birds-eye-view

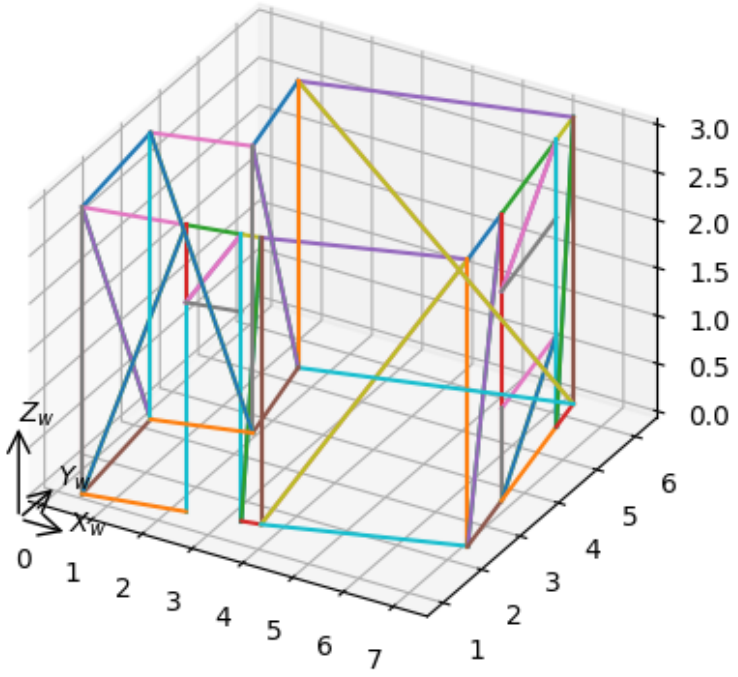
and find minimum cuboid bounding box in 2D/3D

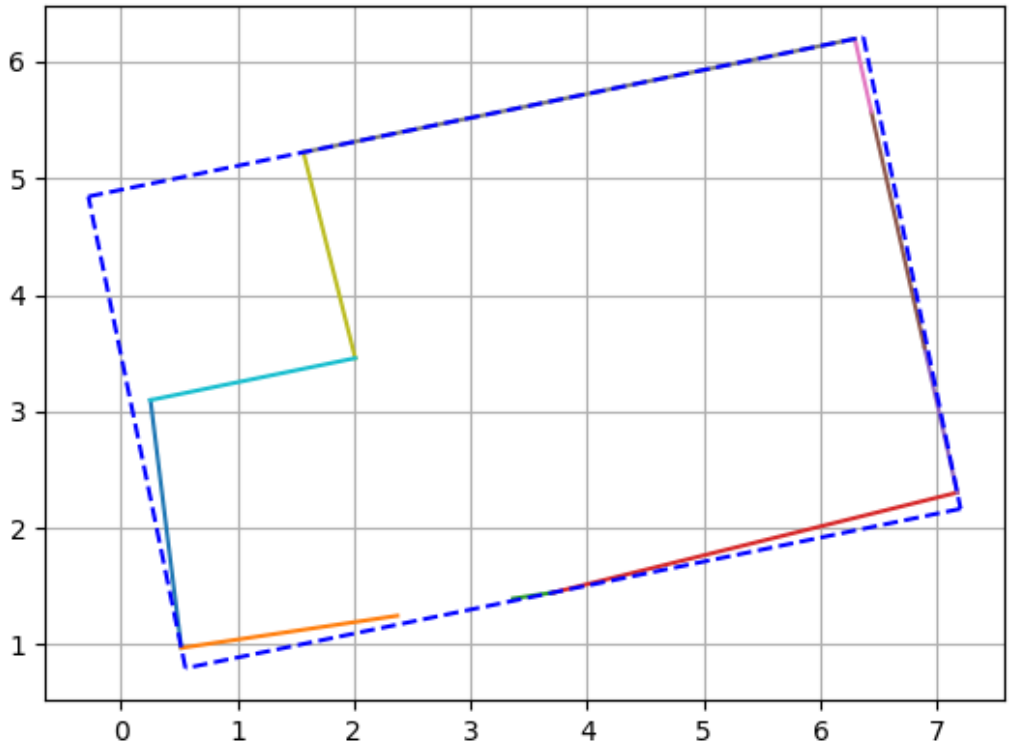
In [3]:

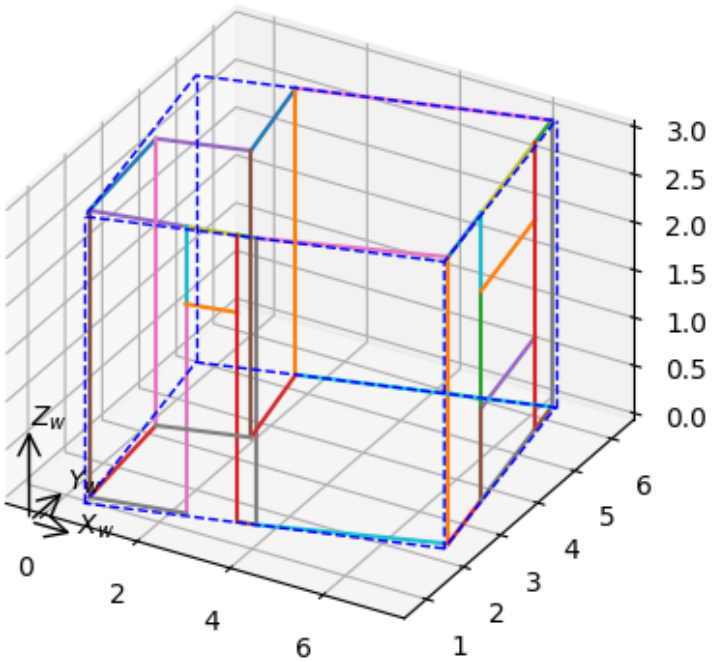
```
from utils_OR.utils_rui import vis_cube_plt, vis_axis
from utils_OR.utils_OR_mesh import minimum_bounding_rectangle, mesh_to_

mesh = load_OR_mesh(layout_obj_file)
# mesh = mesh.dump()[0]
mesh = remove_top_down_faces(mesh)
v = np.array(mesh.vertices)
e = mesh.edges

%matplotlib widget
fig = plt.figure()
ax = fig.gca(projection='3d')
ax.set_proj_type('ortho')
ax.set_aspect("auto")
vis_axis(ax)
v_pairs = v_pairs_from_v3d_e(v, e)
for v_pair in v_pairs:
    ax.plot3D(v_pair[0], v_pair[1], v_pair[2])

# find 2d floor contour
v_2d, e_2d = mesh_to_contour(mesh)
fig = plt.figure()
ax = fig.gca()
ax.set_aspect("equal")
v_pairs = v_pairs_from_v2d_e(v_2d, e_2d)
for v_pair in v_pairs:
    ax.plot(v_pair[0], v_pair[1])

# finding minimum 2d cuboid from contour
layout_hull_2d = minimum_bounding_rectangle(v_2d)
hull_pair_idxes = [[0, 1], [1, 2], [2, 3], [3, 0]]
hull_v_pairs = [([layout_hull_2d[idx[0]][0], layout_hull_2d[idx[1]][0]]
for v_pair in hull_v_pairs:
    ax.plot(v_pair[0], v_pair[1], 'b--')
plt.grid()

# simply mesh -> skeleton
v_skeleton, e_skeleton = mesh_to_skeleton(mesh)

fig = plt.figure()
ax = fig.gca(projection='3d')
ax.set_proj_type('ortho')
ax.set_aspect("auto")
vis_axis(ax)
v_pairs = v_pairs_from_v3d_e(v_skeleton, e_skeleton)
for v_pair in v_pairs:
    ax.plot3D(v_pair[0], v_pair[1], v_pair[2])

# 2d cuvboid hull -> 3d cuboid
room_height = 3.
layout_box_3d = np.hstack((np.vstack((layout_hull_2d, layout_hull_2d)),
vis_cube_plt(layout_box_3d, ax, 'b', linestyle='--')


# transfer layout to world coordinates
transformFile = Path(xml_path) / scene_name / 'transform.dat'
```

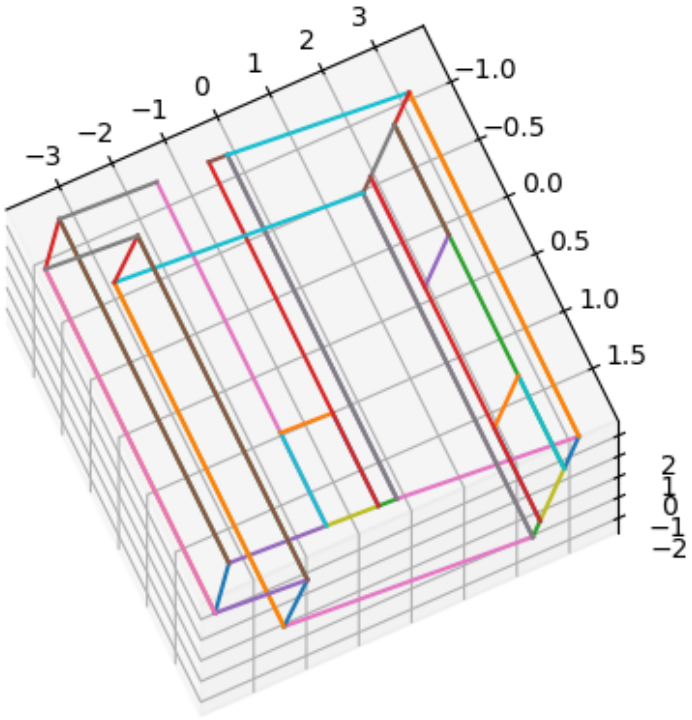
```
# load transformations # writeShapeToXML.py L588
with open(str(transformFile), 'rb') as fIn:
    transforms = pickle.load(fIn )

transforms_layout = transforms[0]

v_skeleton_transform = transform_v(v_skeleton, transforms_layout)
fig = plt.figure()
ax = fig.gca(projection='3d')
ax.set_proj_type('ortho')
ax.set_aspect("auto")
v_pairs = v_pairs_from_v3d_e(v_skeleton_transform, e_skeleton)
for v_pair in v_pairs:
    ax.plot3D(v_pair[0], v_pair[1], v_pair[2])
ax.view_init(elev=-71, azim=-65)
```

```
/tmp/ipykernel_12907/3390101525.py:12: MatplotlibDeprecationWarning: Call
ing gca() with keyword arguments was deprecated in Matplotlib 3.4. Starti
ng two minor releases later, gca() will take no keyword arguments. The gc
a() function should only be used to get the current axes, or if no axes e
xist, create new axes with default keyword arguments. To create a new axe
s with non-default arguments, use plt.axes() or plt.subplot().
  ax = fig.gca(projection='3d')
/tmp/ipykernel_12907/3390101525.py:41: MatplotlibDeprecationWarning: Call
ing gca() with keyword arguments was deprecated in Matplotlib 3.4. Starti
ng two minor releases later, gca() will take no keyword arguments. The gc
a() function should only be used to get the current axes, or if no axes e
xist, create new axes with default keyword arguments. To create a new axe
s with non-default arguments, use plt.axes() or plt.subplot().
  ax = fig.gca(projection='3d')
/tmp/ipykernel_12907/3390101525.py:65: MatplotlibDeprecationWarning: Call
ing gca() with keyword arguments was deprecated in Matplotlib 3.4. Starti
ng two minor releases later, gca() will take no keyword arguments. The gc
a() function should only be used to get the current axes, or if no axes e
xist, create new axes with default keyword arguments. To create a new axe
s with non-default arguments, use plt.axes() or plt.subplot().
  ax = fig.gca(projection='3d')
/home/ruizhu/Documents/Projects/indoorInverse/train/utils/utils_OR/utils_
rui.py:81: MatplotlibDeprecationWarning:
The M attribute was deprecated in Matplotlib 3.4 and will be removed two
minor releases later. Use self.axes.M instead.
  xs, ys, zs = proj3d.proj_transform(xs3d, ys3d, zs3d, renderer.M)
```


3.0
2.5
2.0
1.5
1.0
0.5
0.0
Z_w
Y_w
X_w
0 1 2 3 4 5 6 7
1 2 3 4 5 6


6
5
4
3
2
1
0
1
2
3
4
5
6
7


3.0
2.5
2.0
1.5
1.0
0.5
0.0
Z_w
Y_w
X_w
0
2
4
6
1
2
3
4
5
6



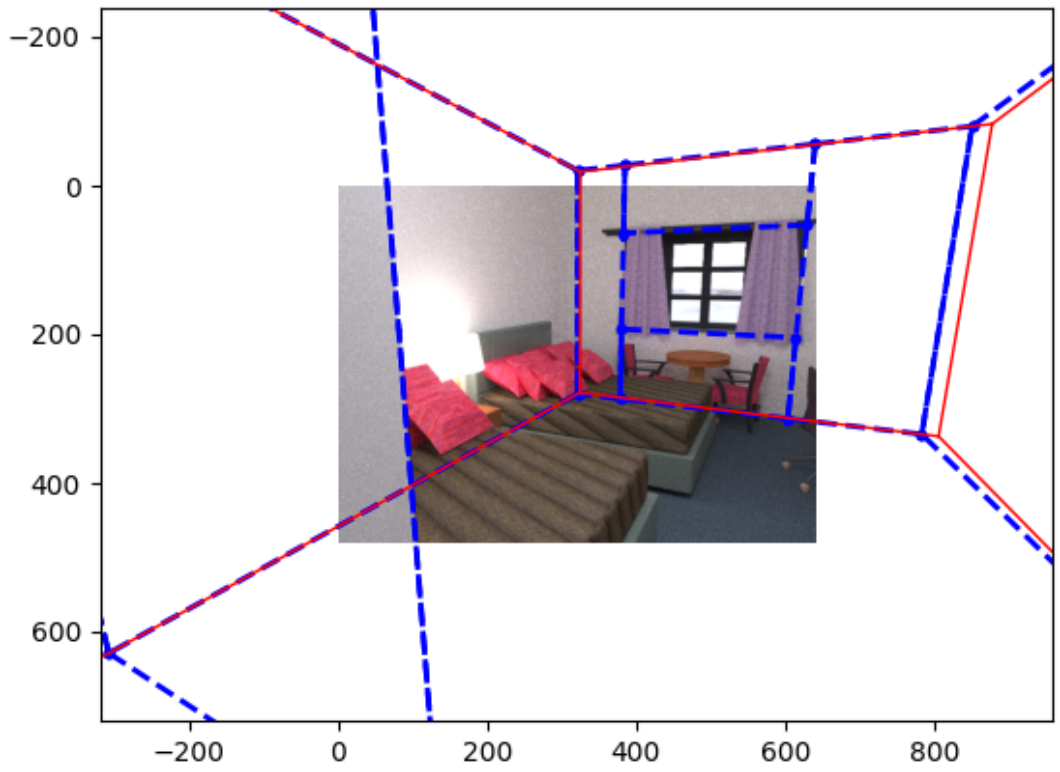
reading camera intrinsics & projection

projecting original layout and layout bbox

In [4]:

```
# == Read cam intrinsics
from utils_OR.utils_OR_xml import get_XML_root, parse_XML_for_intrinsics
from utils_OR.utils_OR_cam import read_cam_params, normalize, project_v
from PIL import Image, ImageDraw, ImageFont
from utils_OR.utils_OR_imageops import in_frame, draw_projected_bdb3d
from utils_OR.utils_OR_mesh import v_xytuple_from_v2d_e

cam_file = Path(xml_path) / scene_name / 'cam.txt'
main_xml_file = Path(xml_path) / scene_name / 'main.xml'
root = get_XML_root(main_xml_file)

cam_K, intrinsics = parse_XML_for_intrinsics(root)


# == Read cam extrinsics

cam_file = Path(xml_path) / scene_name / 'cam.txt'
cam_params = read_cam_params(cam_file)

np.set_printoptions(precision=4)
np.set_printoptions(suppress=True)

cam_param = cam_params[frame_idx-1]
origin, lookat, up = np.split(cam_param.T, 3, axis=1)
at_vector = lookat - origin
assert np.amax(np.abs(np.dot(at_vector.flatten(), up.flatten()))) < 2e-3

zaxis = normalize(lookat - origin)
xaxis = normalize(np.cross(up.T, zaxis.T).T)
yaxis = np.cross(zaxis.T, xaxis.T).T

R_c = np.hstack([xaxis, yaxis, zaxis])
R_c = np.linalg.inv(R_c)
t_c = - R_c @ origin
v_proj = project_v(v_skeleton_transform, R_c, t_c, cam_K)


layout_box_3d_transform = transform_v(layout_box_3d, transforms_layout)

plt.figure()
plt.imshow(im_uint8)

%reload_ext autoreload
%autoreload 2
from utils_OR.utils_OR_geo import isect_line_plane_v3
from utils_OR.utils_OR_cam import project_3d_line

for idx1, idx2 in e_skeleton:
    x1x2 = np.vstack((v_skeleton_transform[idx1], v_skeleton_transform[id
    x1x2_proj = project_3d_line(x1x2, R_c, t_c, cam_K, origin, zaxis)
    if x1x2_proj is not None:
        plt.plot([x1x2_proj[0][0], x1x2_proj[1][0]], [x1x2_proj[0][1], x1

for idx_list in [[0, 1, 2, 3, 0], [4, 5, 6, 7, 4], [0, 4], [1, 5], [2, 6]
# for idx_list in [[0, 1]]:
    v3d_array = layout_box_3d_transform
```

```
for i in range(len(idx_list)-1):
    x1x2 = np.vstack((v3d_array[idx_list[i]], v3d_array[idx_list[i+1]
    x1x2_proj = project_3d_line(x1x2, R_c, t_c, cam_K, origin, zaxis)
    if x1x2_proj is not None:
        plt.plot([x1x2_proj[0][0], x1x2_proj[1][0]], [x1x2_proj[0][1]

height, width = im_uint8.shape[:2]
plt.xlim([-width*0.5, width*1.5])
plt.ylim([height*1.5, -height*.5])
plt.show()
```



loading & visualizing object meshes & bboxes in 3D

In [6]:
```
# load object bboxes
root_uv_mapped = Path('/newfoundland2/ruizhu/siggraphasia20dataset/uv_map
root_layoutMesh = Path('/newfoundland2/ruizhu/siggraphasia20dataset/layou

from utils_OR.utils_OR_xml import parse_XML_for_shapes
shape_list = parse_XML_for_shapes(root, root_uv_mapped)

from utils_OR.utils_OR_mesh import loadMesh, computeBox, computeTransform
from utils_OR.utils_OR_transform import *

from tqdm import tqdm

# draw layout, cam and world coordinates in 3D
from utils_OR.utils_rui import Arrow3D

%matplotlib widget

fig = plt.figure()
ax = fig.gca(projection='3d')
ax.set_proj_type('ortho')
ax.set_aspect("auto")
v_pairs = v_pairs_from_v3d_e(v_skeleton_transform, e_skeleton)
for v_pair in v_pairs:
    ax.plot3D(v_pair[0], v_pair[1], v_pair[2])
ax.view_init(elev=-36, azim=89)
vis_axis(ax)


vertices_list = []
bverts_list = []
faces_list = []
num_vertices = 0
obj_path_list = []

# shape_list = shape_list[:4]

IF_SKIP_LAYOUT = True

for shape_idx, shape in tqdm(enumerate(shape_list)):

    if 'aligned_shape.obj' in shape['filename']:
        continue

    if 'container' in shape['filename']:
        continue

    if 'uv_mapped' in shape['filename']:
        obj_path = root_uv_mapped / shape['filename'].replace('../../../.

    if 'layoutMesh' in shape['filename']:
        if IF_SKIP_LAYOUT:
            continue
        obj_path = root_layoutMesh / shape['filename'].replace('../../../

    print(Path(obj_path).absolute())
    vertices, faces = loadMesh(obj_path) # based on L430 of adjustObjectB
```

```
    bverts, bfaces = computeBox(vertices )

    vertices_transformed, _ = transform_with_transforms_xml_list(shape['t
    bverts_transformed, transforms_converted_list = transform_with_transf
    if np.amax(bverts_transformed[:, 1]) <= np.amin(bverts_transformed[:,
        obj_color = 'k'
        print(bverts_transformed, bverts, shape['transforms_list'])
    else:
        obj_color = 'r'

    y_max = bverts_transformed[:, 1].max()
    points_2d = bverts_transformed[abs(bverts_transformed[:, 1] - y_max)
    if points_2d.shape[0] != 4:
        bverts_transformed, bfaces = computeBox(vertices_transformed) # c


    if not(any(ext in shape['filename'] for ext in ['window', 'door', 'la
        if 'scene' not in shape['filename']:
            vis_cube_plt(bverts_transformed, ax, obj_color)

    vertices_list.append(vertices_transformed)

    faces_list.append(faces+num_vertices)
    num_vertices += vertices.shape[0]

    bverts_list.append(bverts_transformed)
    obj_path_list.append(obj_path)

vis_cube_plt(layout_box_3d_transform, ax, 'b', '--')

from utils_OR.utils_rui import vis_axis_xyz
vis_axis_xyz(ax, xaxis.flatten(), yaxis.flatten(), zaxis.flatten(), origi

a = Arrow3D([origin[0][0], lookat[0][0]*2-origin[0][0]], [origin[1][0], l
            lw=1, arrowstyle="->", color="k")
ax.add_artist(a)
a_up = Arrow3D([origin[0][0], origin[0][0]+up[0][0]], [origin[1][0], orig
            lw=1, arrowstyle="->", color="r")
ax.add_artist(a_up)
plt.show()

# write to obj
vertices_combine = np.vstack(vertices_list)
faces_combine = np.vstack(faces_list)
from utils_OR.utils_OR_mesh import writeMesh

writeMesh('test_mesh.obj', vertices_combine, faces_combine)
```

```
6it [00:01,  4.08it/s]

/newfoundland2/ruizhu/siggraphasia20dataset/uv_mapped/03636649/3834d7f376
879c03eca29403b7226aa1/aligned_light.obj
/newfoundland2/ruizhu/siggraphasia20dataset/uv_mapped/03211117/e9466e8728
48075d3aeab48ed64c0fa4/alignedNew.obj

9it [00:01,  6.71it/s]

/newfoundland2/ruizhu/siggraphasia20dataset/uv_mapped/04379243/a9d890e4b6
b426dd358ffaf8d4d252a/alignedNew.obj
```

```
10it [00:02,  5.16it/s]

/newfoundland2/ruizhu/siggraphasia20dataset/uv_mapped/02818832/22b8e18050
41fe56010a6840f668b41/alignedNew.obj
/newfoundland2/ruizhu/siggraphasia20dataset/uv_mapped/04379243/dcf2462803
61e20d1bf2b66b52bf6885/alignedNew.obj

13it [00:02,  5.65it/s]
```

In [11]:

```
from pygel3d import hmesh, gl_display as gl

bunny = hmesh.load("test_mesh.obj")
# viewer = gl.Viewer()
# viewer.display(m)

print("vertices before simplification :", bunny.no_allocated_vertices())
hmesh.close_holes(bunny)
hmesh.triangulate(bunny)
hmesh.quadric_simplify(bunny, 0.05)
bunny.cleanup()
print("vertices after simplification :", bunny.no_allocated_vertices())
```

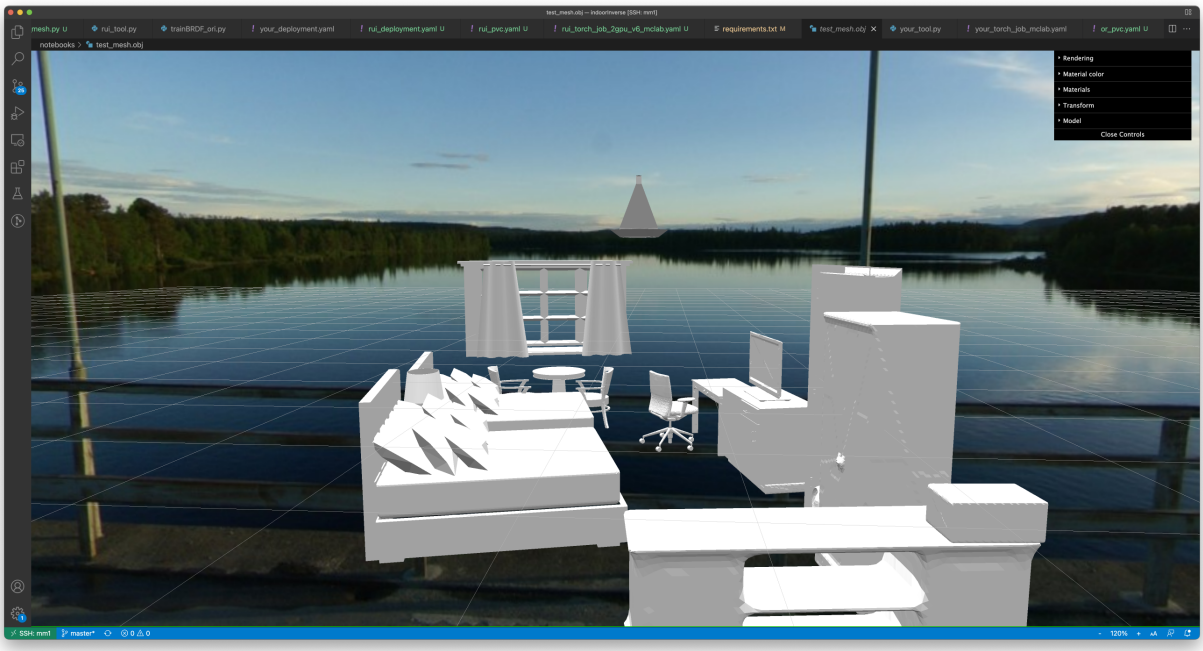
```
vertices before simplification : 284238
vertices after simplification : 19228
```

In [14]:

```
from pygel3d import jupyter_display as jd
jd.set_export_mode(True)
jd.display(bunny)
```

```
---------------------------------------------------------------------------
--
IndexError                                Traceback (most recent call las
t)
/tmp/ipykernel_12907/1649222755.py in <module>
      1 from pygel3d import jupyter_display as jd
      2 jd.set_export_mode(True)
----> 3 jd.display(bunny)

~/anaconda3/envs/py38/lib/python3.8/site-packages/pygel3d/jupyter_displa
y.py in display(m, wireframe, smooth, data)
     39         ijk = array([[ idx for idx in m_tri.circulate_face(f,'v')
] for f in m_tri.faces()])
     40         mesh = go.Mesh3d(x=xyz[:,0],y=xyz[:,1],z=xyz[:,2],
---> 41                  i=ijk[:,0],j=ijk[:,1],k=ijk[:,2],color='#dddddd',
flatshading=not smooth)
     42         if data is not None:
     43             mesh['intensity'] = data

IndexError: too many indices for array: array is 1-dimensional, but 2 wer
e indexed
```

the dumped mesh file should look like this (opened using [3D Viewer for VSCode (https://marketplace.visualstudio.com/items?itemName=slevesque.vscode-3dviewer)](https://marketplace.visualstudio.com/items?itemName=slevesque.vscode-3dviewer)



(opened using Meshlab)