

The screenshot shows a terminal window titled "modelrenderer.py (-/ros/indigo/ros\_ws/src/deeptracking/deeptracking/data) - gedit". The window contains Python code for rendering a 3D model using OpenGL and PEGI. The code includes imports for os, sys, and various OpenGL and PEGI modules. It defines a class with methods for initializing OpenGL, drawing elements, and retrieving data from the window. The code uses PEGI to handle context creation and OpenGL functions like glTexParameteri and glDrawElements. It also handles depth conversion and array reshaping. The terminal window has a dark theme and shows standard Linux desktop icons on the left.

```
modelrenderer.py (-/ros/indigo/ros_ws/src/deeptracking/deeptracking/data) - gedit
[1] generate_synthetic_data.py [2] generate_synthetic_example.pcd [3] 无标题文档1 x [4] modelrenderer.py x
gedit
打开 保存 撤消 撤消
generate_synthetic_data.py
generate_synthetic_example.pcd
无标题文档1
modelrenderer.py

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT)
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT)
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR)
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR)

glDrawElements(GL_TRIANGLES, len(self.faces) * 3, GL_UNSIGNED_INT, ctypes.c_void_p(0))

# -- retrieve data
depth_array = glReadPixels(0, 0, self.window_size[0], self.window_size[1], GL_DEPTH_COMPONENT, GL_FLOAT)
depth_array = depth_array.reshape(self.window_size[1:-1])
depth_array = self.gldepth_to_worlddepth(depth_array)
rgb_array = glReadPixels(0, 0, self.window_size[0], self.window_size[1], GL_RGB, GL_UNSIGNED_BYTE)
rgb_array = np.frombuffer(rgb_array, dtype=np.uint8).reshape((self.window_size[1], self.window_size[0], 3))
return rgb_array, depth_array

def InitOpenGL(width, height, hide_window=True):
    """
    dpy = Display()
    conf = pegl.config.get_configs(dpy, {'RENDERABLE_TYPE': ClientAPIs.OPENGL=1,
                                         'SUBPIXELS': SurfaceTypes(PBUFFER=1,
                                         'BLUE_SIZE': 8,
                                         'GREEN_SIZE': 8,
                                         'RED_SIZE': 8,
                                         'DEPTH_SIZE': 8)}
    conf = conf[0]
    surf = pegl.surface.Pbuffer(dpy, conf, {'WIDTH': width, 'HEIGHT': height})

    pengl.context.bind_api(ContextAPIs.OPENGL)
    ctx = pengl.context.Context(dpy, conf)
    ctx.make_current(draw_surface=surf)
    ...

    if not glfw.init():
        print("Failed to initialize GLFW\n", file=sys.stderr)
        sys.exit(-1)
    window = glfw.create_window(width, height, "ViewpointRender", None, None)
    if not window:
        print(
            "Failed to open GLFW window. If you have an Intel GPU, they are not 3.3 compatible. Try the 2.1 version of the tutorials.\n",
            file=sys.stderr)
        glfw.terminate()
        sys.exit(-1)
    glfw.make_context_current(window)

    glewExperimental = True
    if glewInit() != GLEW_OK:
        print("Failed to initialize GLEW\n", file=sys.stderr)
        sys.exit(-1)
    glClearColor(0, 0, 0, 0)
    window = None
    ...

    # OpenGL Flags
    glEnable(GL_DEPTH_TEST)
```