class LineNode():

def \_\_init\_\_(self, id: int, lineset: o3d.geometry.LineSet, neighbors: set[int]):

self.id = id

self.lineset = lineset

self.neighbors = neighbors

class LineGraph():

def \_\_init\_\_(self, linenodes):

self.nodes = linenodes

def insert\_linenode(self, linenode: LineNode):

self.nodes[linenode.id] = linenode

def get\_node(self, id: int):

return self.nodes[id]

def read\_linesets():

global selected\_road\_file # Import the global variable

global selected\_road\_string\_file # Declare the global variable

if selected\_road\_string\_file:

# Read the data from a string file

with open(selected\_road\_string\_file, "r") as file:

data\_str = file.read()

else:

# Handle the case where no string file is selected

data\_str = "" # Set data\_str to an empty string or any default value

# Split the data by lines

lines = data\_str.strip().split('\n')

# Initialize lists to store data

group\_id = []

x = []

y = []

z = []

#current\_group = None # Initialize the current group ID

current\_group\_id = 0 # Initialize the first group ID

for line in lines:

parts = line.split(',')

# Check if the line has enough parts to be considered a data line

if len(parts) >= 4:

try:

current\_id = int(parts[0])

if current\_id == 0:

# Group change, increase the group ID

current\_group\_id += 1

else:

# Append data to lists

group\_id.append(current\_group\_id)

x.append(float(parts[1]))

y.append(float(parts[2]))

z.append(float(parts[3]))

except (ValueError, IndexError):

# Handle errors, such as non-integer values or missing parts

pass

# Create a DataFrame

data = {

'Group ID': group\_id,

'X': y,

'Y': x,

'Z': z,

}

df = pd.DataFrame(data)

# Save to an Excel file

df.to\_excel('output\_data.xlsx', index=False)

# Determine which file to use based on the global variables

if selected\_road\_file:

road\_data\_file = selected\_road\_file

elif selected\_road\_string\_file:

road\_data\_file = 'output\_data.xlsx'

else:

# Handle the case where neither variable is set

raise Exception("No road data file selected")

# Extract the road points

road\_points\_df = pd.read\_excel(road\_data\_file, header=None)

road\_points\_df = road\_points\_df.apply(pd.to\_numeric, errors='coerce')

linesets = {}

for group\_id, group in road\_points\_df.groupby(0):

points = group.iloc[:, 1:].to\_numpy()

lines = [[i, i+1] for i in range(len(points)-1)]

lineset = o3d.geometry.LineSet()

lineset.points = o3d.utility.Vector3dVector(points)

lineset.lines = o3d.utility.Vector2iVector(lines)

lineset.paint\_uniform\_color([0, 0, 1])

linesets[group\_id] = lineset

nodes = {}

for x in range(1, len(linesets) + 1):

nodes[x] = LineNode(x, linesets[x], set())

for linenode in nodes.values():

id = linenode.id

lineset = linenode.lineset

for other\_id in range(1, len(linesets)+1):

if other\_id == id: continue

if other\_id in nodes[id].neighbors: continue

points\_one = np.asarray(lineset.points)

points\_two = np.asarray(linesets[other\_id].points)

for point\_1\_id in range(points\_one.shape[0]):

for point\_2\_id in range(points\_two.shape[0]):

if np.all(np.equal(points\_one[point\_1\_id, :], points\_two[point\_2\_id, :])):

nodes[id].neighbors.add(other\_id)

nodes[other\_id].neighbors.add(id)

global graph

graph = LineGraph(nodes)

return linesets