Final Countdown

from qgis.core import (

QgsApplication,

QgsVectorLayer,

QgsField,

QgsFeature,

QgsGeometry,

QgsPointXY,

QgsRasterLayer,

QgsVectorFileWriter,

QgsProject

)

from qgis.PyQt.QtCore import QVariant

from osgeo import gdal

# Initialize QGIS Application

QgsApplication.setPrefixPath("/usr", True)

app = QgsApplication([], False)

app.initQgis()

# Input paths - replace these with your actual paths

input\_polygon\_layer\_path = 'C:\\Users\\korisnik\\Desktop\\Hidrocibalae\\01\_Task\_BoulderHeights\_toSend\\Boulder polygons\\Test\_Manually\_Picked\_Boulders.shp'

input\_raster\_path = 'C:\\Users\\korisnik\\Desktop\\Hidrocibalae\\01\_Task\_BoulderHeights\_toSend\\MBES grid\\Test\_Encoded\_Depths\_File.tif'

block\_name = 'B02' # Replace with your block name

# Load the boulder polygon vector layer

boulder\_layer = QgsVectorLayer(input\_polygon\_layer\_path, 'BoulderPolygons', 'ogr')

if not boulder\_layer.isValid():

print('Boulder layer failed to load!')

# Load the bathymetric raster layer

bathymetry\_raster = QgsRasterLayer(input\_raster\_path, 'Bathymetry')

if not bathymetry\_raster.isValid():

print('Bathymetric layer failed to load!')

# Create the output point layer with the required fields

output\_fields = [

QgsField('Poly\_ID', QVariant.Int),

QgsField('Target\_ID', QVariant.String),

QgsField('Block', QVariant.String),

QgsField('Easting', QVariant.Double),

QgsField('Northing', QVariant.Double),

QgsField('Water\_depth', QVariant.Double),

QgsField('Length', QVariant.Double),

QgsField('Width', QVariant.Double),

QgsField('Height', QVariant.Double)

]

crs = boulder\_layer.crs().authid()

output\_layer = QgsVectorLayer(f'Point?crs={crs}', 'BoulderCentroids', 'memory')

dp = output\_layer.dataProvider()

dp.addAttributes(output\_fields)

output\_layer.updateFields()

# Function to calculate boulder dimensions

def calculate\_dimensions(feature):

# Replace this with the actual code to calculate dimensions from the polygon geometry

# For example, using the bounding box or minimum area rectangle to determine length and width

# Height could be derived from the bathymetric data if available

return length, width, height

# Function to get water depth from the raster layer

def get\_water\_depth(point, raster\_layer):

# Get the value of the raster at the given point

# Replace this with the code to extract the depth value from your bathymetric data

return depth

# Process each boulder polygon feature

for count, feature in enumerate(boulder\_layer.getFeatures()):

# Calculate centroid

centroid = feature.geometry().centroid().asPoint()

# Get water depth from bathymetric raster

water\_depth = get\_water\_depth(centroid, bathymetry\_raster)

# Calculate boulder dimensions

length, width, height = calculate\_dimensions(feature)

# Create a new feature for the output layer

new\_feature = QgsFeature(output\_layer.fields())

new\_feature['Poly\_ID'] = count

new\_feature['Target\_ID'] = f'MBES\_{block\_name}\_{count:02d}'

new\_feature['Block'] = block\_name

new\_feature['Easting'] = centroid.x()

new\_feature['Northing'] = centroid.y()

new\_feature['Water\_depth'] = water\_depth

new\_feature['Length'] = length

new\_feature['Width'] = width

new\_feature['Height'] = height

new\_feature.setGeometry(QgsGeometry.fromPointXY(centroid))

# Add the new feature to the output layer

dp.addFeatures([new\_feature])

# Save the output point layer to a Shapefile in the same folder as the input vector layer

output\_path = input\_polygon\_layer\_path.rsplit('.', 1)[0] + '\_centroids.shp'

QgsVectorFileWriter.writeAsVectorFormat(output\_layer, output\_path, 'UTF-8', boulder\_layer.crs(), 'ESRI Shapefile')

# Add the output layer to the map (optional, only if you are running this script in the QGIS interface)

QgsProject.instance().addMapLayer(output\_layer)

# Finalize the QGIS application

app.exitQgis()

app.exit()