Final Countdown 2

import os

from qgis.core import (

QgsApplication, QgsVectorLayer, QgsField, QgsFeature,

QgsGeometry, QgsPointXY, QgsRaster, QgsRasterLayer, QgsProject, QgsCoordinateTransform, QgsCoordinateReferenceSystem,

QgsVectorFileWriter

)

from qgis.PyQt.QtCore import QVariant

from osgeo import gdal

# Initialize QGIS Application

QgsApplication.setPrefixPath("/usr", True)

app = QgsApplication([], False)

app.initQgis()

# Define the path to the vector layer and the raster layer

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

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

# Load the vector layer and the raster layer

vector\_layer = QgsVectorLayer(vector\_layer\_path, 'Boulder Polygons', 'ogr')

raster\_layer = QgsRasterLayer(raster\_layer\_path, 'Bathymetry')

# Check if layers loaded successfully

if not vector\_layer.isValid() or not raster\_layer.isValid():

raise Exception("One or more input layers failed to load!")

# Create the output vector layer with the required fields

crs = vector\_layer.crs().toWkt()

output\_layer = QgsVectorLayer('Point?crs=' + crs, 'BoulderCentroids', 'memory')

output\_dp = output\_layer.dataProvider()

output\_dp.addAttributes([

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)

])

output\_layer.updateFields()

# Define the block name

block\_name = 'B02' # Replace with the actual block name

# Function to calculate boulder height using raster data

def calculate\_boulder\_height(feature, raster\_layer):

# Get geometry of the feature and extract points

geometry = feature.geometry()

points = geometry.vertices()

# Setup transformer to convert coordinates

crsSrc = QgsCoordinateReferenceSystem(vector\_layer.crs())

crsDest = QgsCoordinateReferenceSystem(raster\_layer.crs())

transformer = QgsCoordinateTransform(crsSrc, crsDest, QgsProject.instance())

# Initialize min and max elevation

min\_elev = float('inf')

max\_elev = float('-inf')

# Iterate through all points to find min and max elevation

for point in points:

# Transform the point to the raster's CRS

point.transform(transformer)

# Get the raster value at the point

ident = raster\_layer.dataProvider().identify(point, QgsRaster.IdentifyFormatValue)

if ident.isValid():

elevation = ident.results()[1] # assuming band 1 contains elevation

min\_elev = min(min\_elev, elevation)

max\_elev = max(max\_elev, elevation)

# Calculate the height as the difference between max and min elevations

height = max\_elev - min\_elev

return height

# Iterate through each feature in the input vector layer

for idx, feature in enumerate(vector\_layer.getFeatures()):

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

water\_depth = ... # Calculate or retrieve the water depth for the centroid

length = ... # Calculate the length of the boulder polygon

width = ... # Calculate the width of the boulder polygon

height = calculate\_boulder\_height(feature, raster\_layer)

# Create a new feature for the output layer

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

new\_feature.setAttribute('Poly\_ID', idx)

new\_feature.setAttribute('Target\_ID', f'MBES\_{block\_name}\_{idx:02d}')

new\_feature.setAttribute('Block', block\_name)

new\_feature.setAttribute('Easting', centroid.x())

new\_feature.setAttribute('Northing', centroid.y())

new\_feature.setAttribute('Water\_depth', water\_depth)

new\_feature.setAttribute('Length', length)

new\_feature.setAttribute('Width', width)

new\_feature.setAttribute('Height', height)

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

# Add the feature to the output layer

output\_dp.addFeatures([new\_feature])

# Save the output layer to a shapefile

output\_layer\_path = os.path.splitext(vector\_layer\_path)[0] + '\_centroids.shp'

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

# Finalize QGIS application

app.exitQgis()

app.exit()