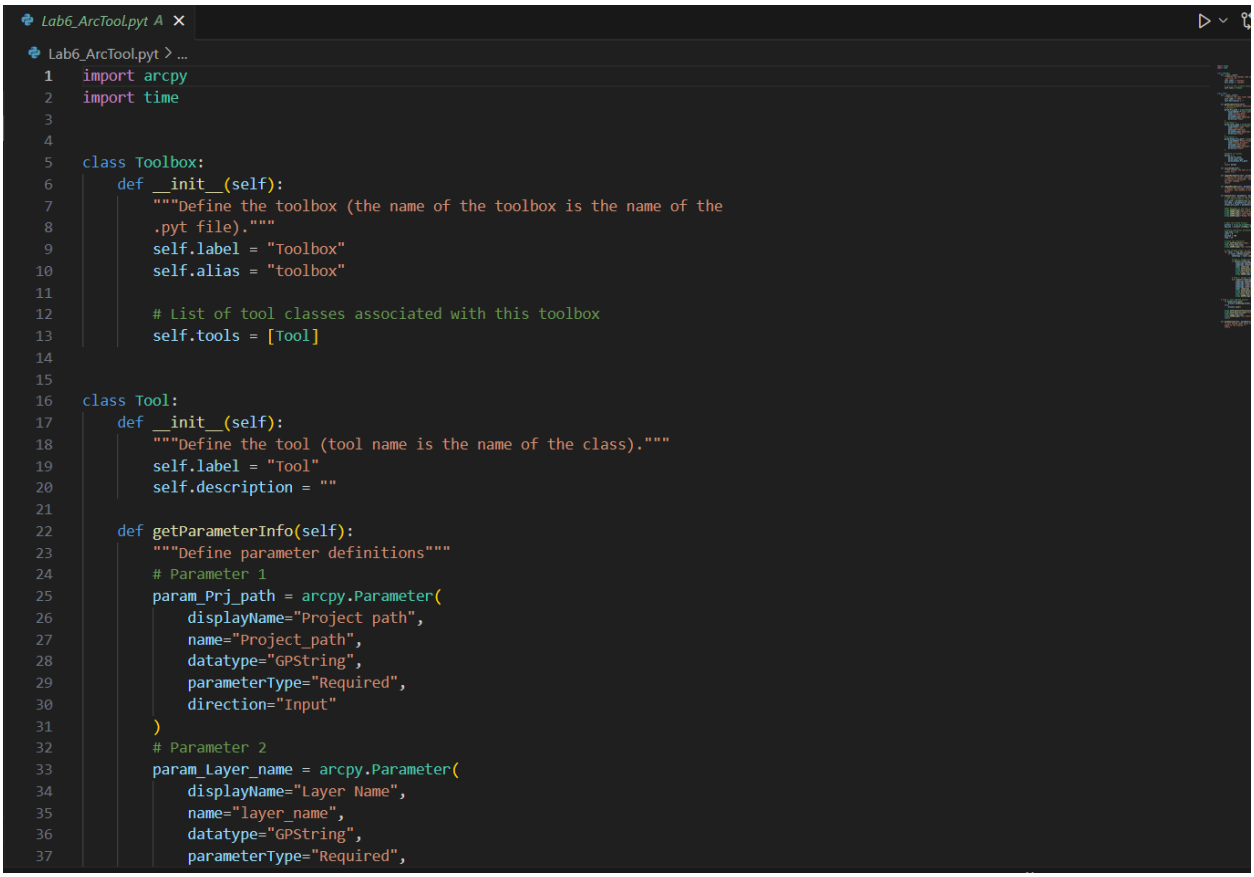
Github Link

https://github.com/bhavikgupta/Lab6

1) Screenshots of implementation of toolbox (.pyt file)

Working project path, Layer name



```python
import arcpy
import time


class Toolbox:
    def __init__(self):
        """Define the toolbox (the name of the toolbox is the name of the
        .pyt file)."""
        self.label = "Toolbox"
        self.alias = "toolbox"

        # List of tool classes associated with this toolbox
        self.tools = [Tool]


class Tool:
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Tool"
        self.description = ""

    def getParameterInfo(self):
        """Define parameter definitions"""
        # Parameter 1
        param_Prj_path = arcpy.Parameter(
            displayName="Project path",
            name="Project_path",
            datatype="GPString",
            parameterType="Required",
            direction="Input"
        )
        # Parameter 2
        param_Layer_name = arcpy.Parameter(
            displayName="Layer Name",
            name="layer_name",
            datatype="GPString",
            parameterType="Required",
```

Output project path

```python
        # Parameter 3
        param_Output_Prj_path = arcpy.Parameter(
            displayName="Output Project Path",
            name="output_prj_path",
            datatype="GPString",
            parameterType="Optional",
            direction="Input"
        )

        #Combine all params
        params = [
            param_Prj_path,
            param_Layer_name,
            param_Output_Prj_path
        ]
        return params

    def isLicensed(self):
        """Set whether the tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):
        """Modify the values and properties of parameters before internal
        validation is performed.  This method is called whenever a parameter
        has been changed."""
        return

    def updateMessages(self, parameters):
        """Modify the messages created by internal validation for each tool
        parameter. This method is called after internal validation."""
        return
```

```python
    def execute(self, parameters, messages):
        """The source code of the tool."""
        # Set input parameters from toolbox
        proj_path = parameters[0].valueAsText   # Input project path (.aprx)
        lyr_name = parameters[1].valueAsText    # Layer Name for rendering
        output_proj_path = parameters[2].valueAsText  # Optional: Output project path for saving results

        #adds Messages to the tool window
        arcpy.AddMessage("User Input:")
        arcpy.AddMessage(f"Project Path: {proj_path}")
        arcpy.AddMessage(f"Layer Name: {lyr_name}")
        arcpy.AddMessage(f"Output Project Path(Optional): {output_proj_path}")



        # Open the ArcGIS Project
        project = arcpy.mp.ArcGISProject(proj_path)
        map_obj = project.listMaps('Map')[0]  # Assumes only one map named 'Map'

        # Define progressor parameters
        read_time = 1.5
        start = 0
        maximum = 100
        step = 25

        # Setup the progressor
        arcpy.SetProgressor("step", "Initializing...", start, maximum, step)
        time.sleep(read_time)
        arcpy.AddMessage("Tool initialized...")
```

Renderer change from Structures to Unique Value Renderer, Renderer change of Trees to Graduated Color Renderer.

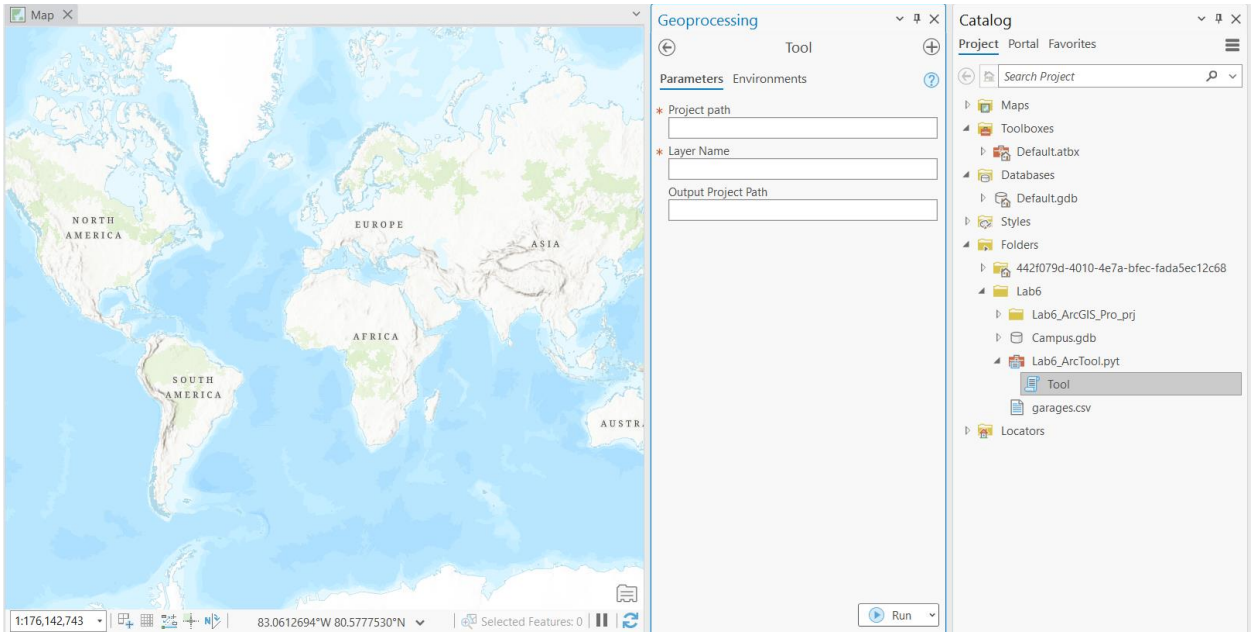
```
        # Loop through layers to set renderers
        for layer in map_obj.listLayers():
            if layer.isFeatureLayer:
                symbology = layer.symbology

                # Step 1: Change Structures Layer to UniqueValueRenderer
                if hasattr(symbology, 'renderer') and layer.name == "Structures":
                    symbology.updateRenderer('UniqueValueRenderer')
                    symbology.renderer.fields = ["Type"]  # Field for Unique Value rendering
                    layer.symbology = symbology
                    arcpy.SetProgressorPosition(start + step)
                    arcpy.SetProgressorLabel("Rendering Structures layer...")
                    time.sleep(read_time)
                    arcpy.AddMessage("Structures layer renderer updated.")

                # Step 2: Change Trees Layer to GraduatedColorsRenderer
                elif hasattr(symbology, 'renderer') and layer.name == "Trees":
                    symbology.updateRenderer('GraduatedColorsRenderer')
                    symbology.renderer.classificationField = "Shape_Area" # Field for Graduated Colors rendering
                    symbology.renderer.breakCount = 5  # Number of classes
                    symbology.renderer.colorRamp = project.listColorRamps('Oranges (5 Classes)')[0]  # Color ramp
                    layer.symbology = symbology
                    arcpy.SetProgressorPosition(start + 2 * step)
                    arcpy.SetProgressorLabel("Rendering Trees layer...")
                    time.sleep(read_time)
                    arcpy.AddMessage("Trees layer renderer updated.")

```

Progressor bar

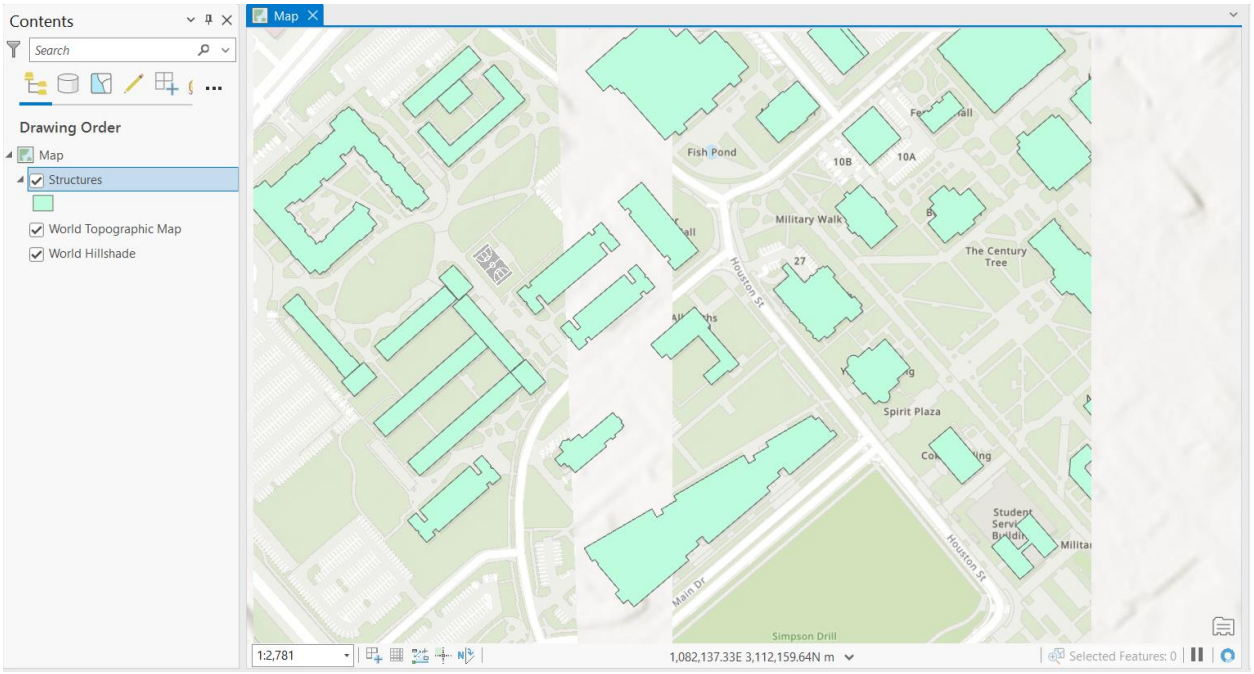
```
    # Step 3: Save updated project
        if output_proj_path:
            project.saveACopy(output_proj_path)
        else:
            project.save()

        arcpy.SetProgressorPosition(maximum)
        arcpy.SetProgressorLabel("Finalizing...")
        time.sleep(read_time)
        arcpy.AddMessage("Tool execution completed.")
        return

    def postExecute(self, parameters):
        """This method takes place after outputs are processed and
        added to the display."""
        return


```

2) Results Screenshots

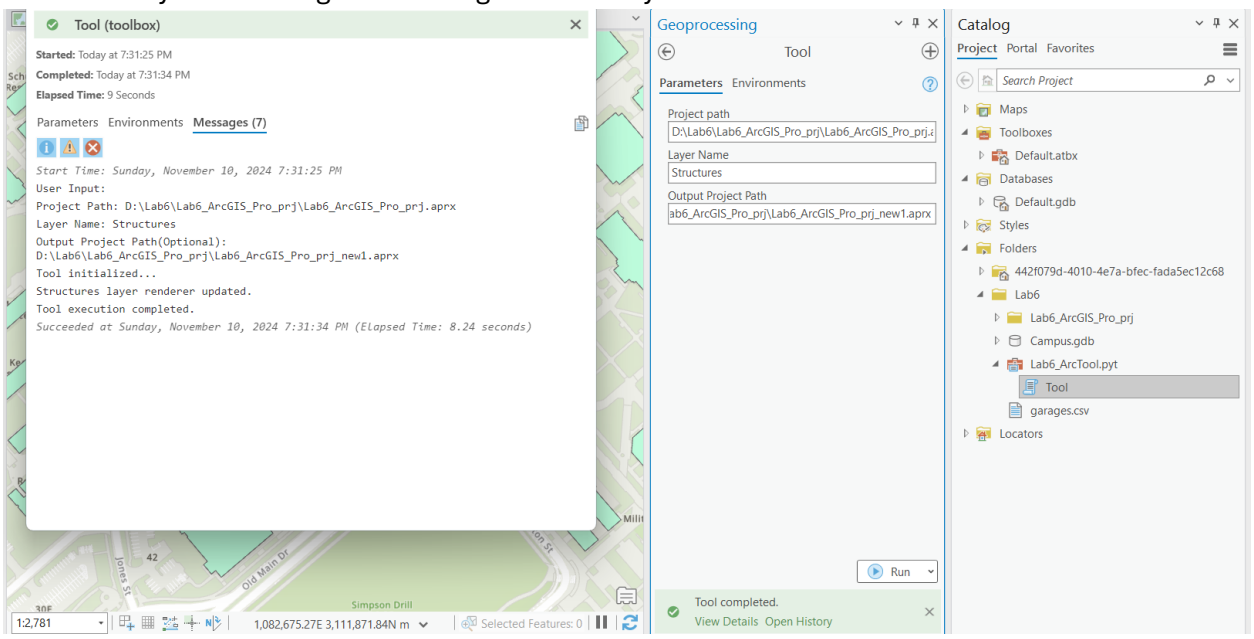
Tool Interface



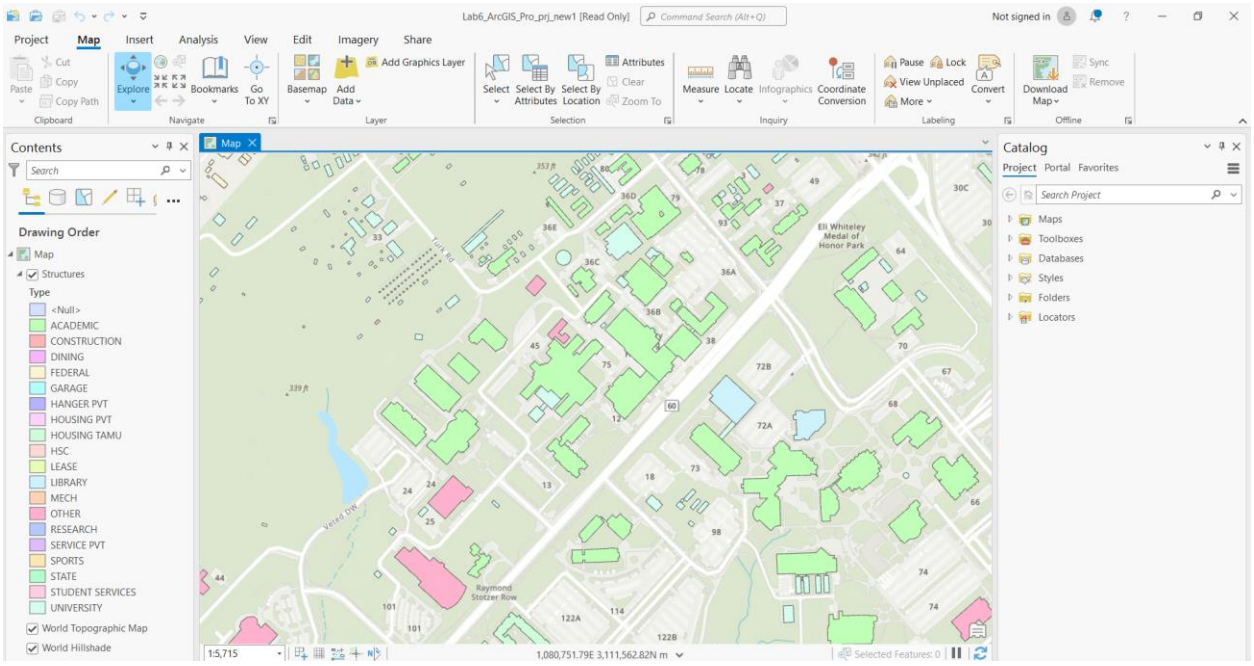
Original Map



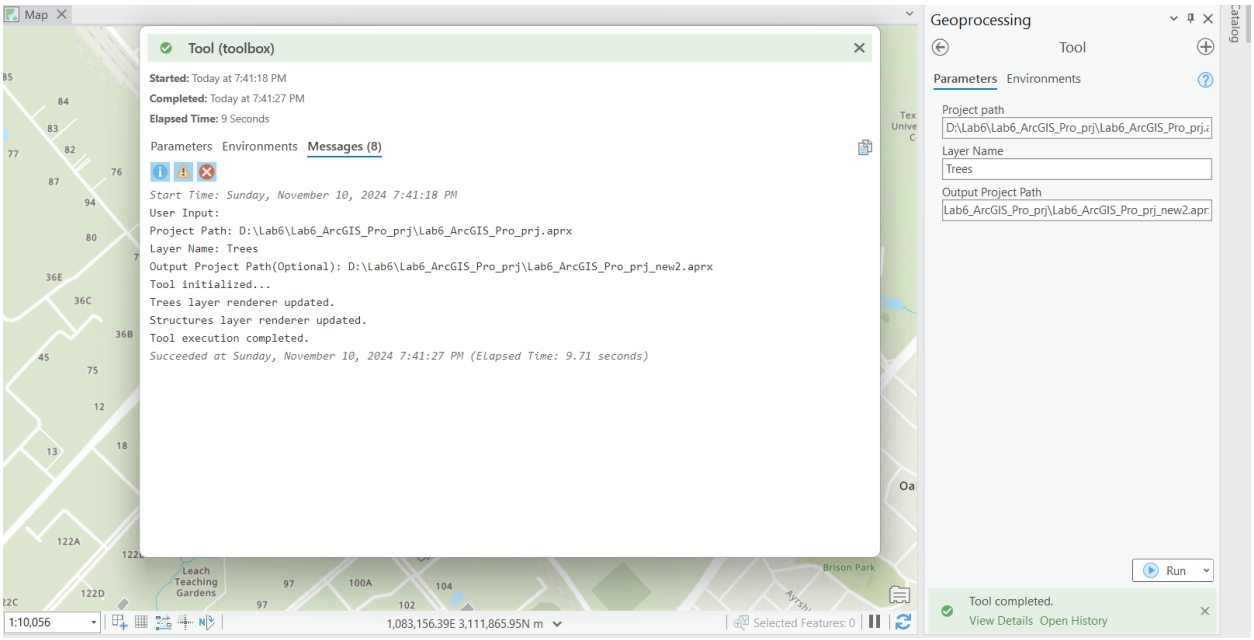
Structure Layer Rendering Tool running Successfully



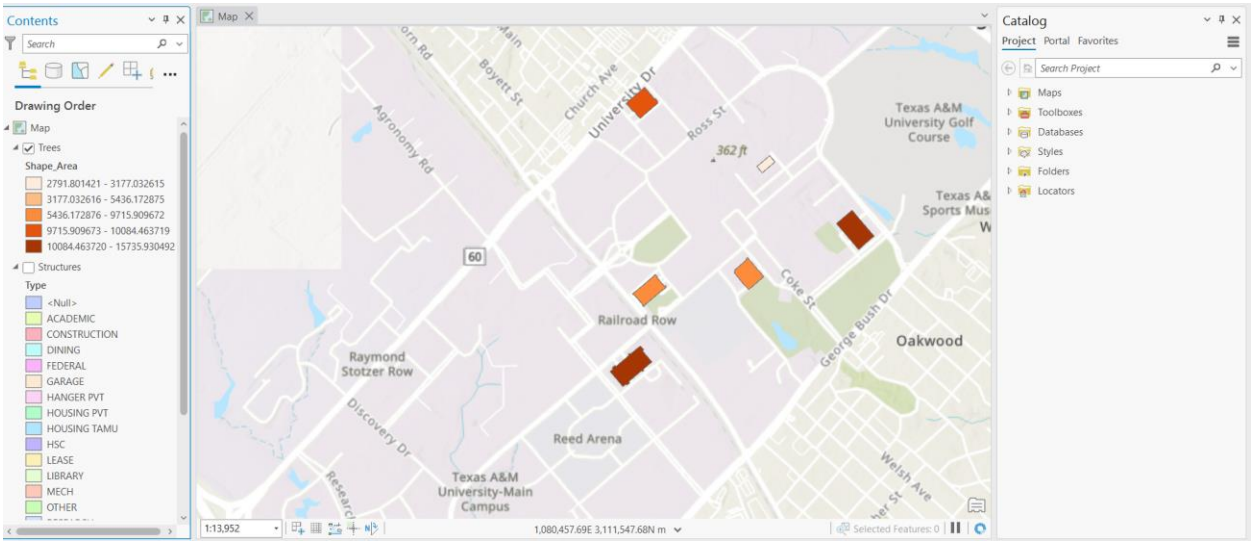
Result (Structure Layer Rendering Tool)



Tree Layer Rendering Tool running Successfully



Result (Tree Layer Rendering Tool)



Progress Bar