

ArcGIS Pro and 3D Printing

Objective

This workflow is used to produce 3D models from a DEM and ArcGIS Feature Class. It was developed for a project to create a 3D printed puzzle where each piece represents a “mountain shed” (opposed to a traditional watershed). This document can also serve as a guide for creating 3D models of any ArcGIS Polygon Feature Class. For our puzzle map the pieces will be printed in 4-colors to give scale to the mountains, breaking at 2000’ intervals (green to 8k, brown to 10k, gray to 12k, and white above).

The intention was to produce a large puzzle, on the order of a few square feet. The largest puzzle piece has to fit on the print bed (12”x12” for our 3D printer), then we scaled the smaller models accordingly. Other constraints included file size limits at each step in the process, which was managed by adjusting the cell size on the output TIFF file.

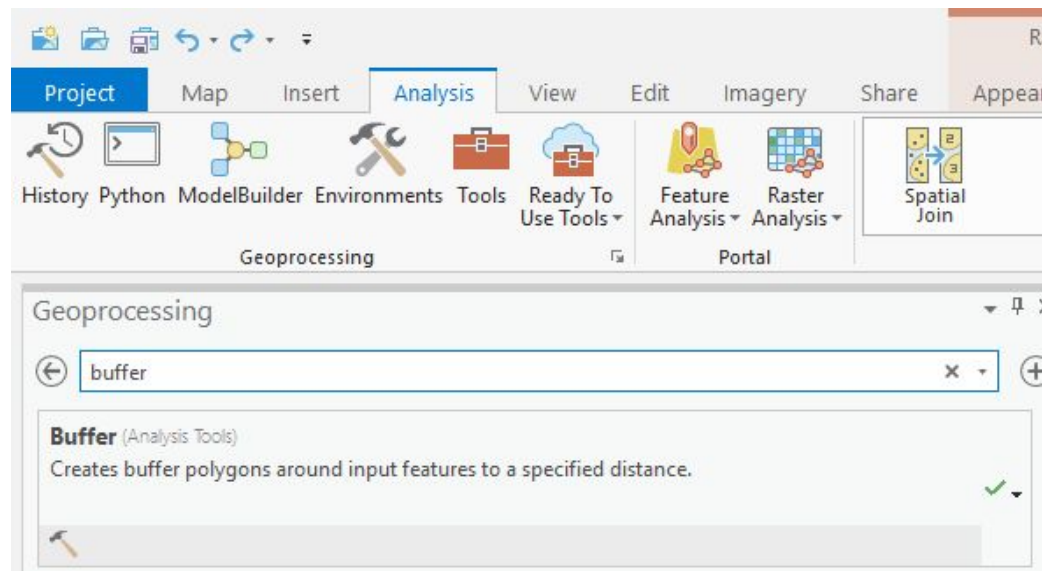
A basemap should be created to accompany the 3D parts, and magnets can be attached to the bottom for mounting on a vertical surface. The level of detail in the basemap will drive the difficulty of the puzzle. Once assembled, the puzzle provides a new perspective to look at terrain that you might have otherwise felt very familiar with.

Resources

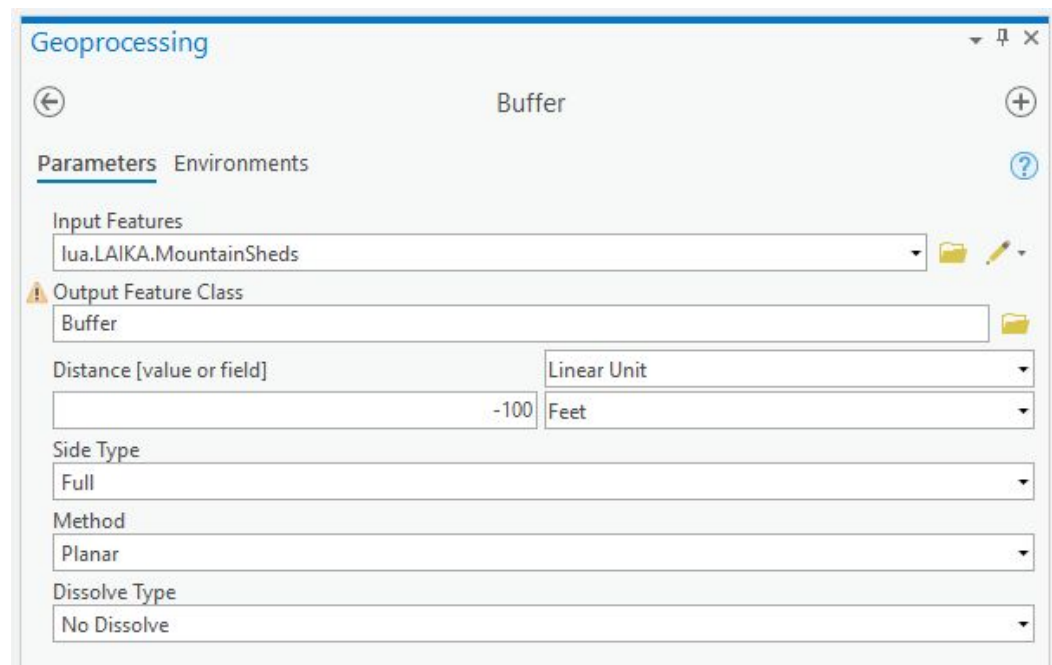
- ArcGIS Pro 2.3.3
- Blender 2.79
- Python 2.7
 - Phstl.py (Author: Jim DeVona, MIT License)
 - Blender.py (Developed by Pitkin County)

GIS Workflow

1. Import all required data to a New Map in ArcGIS Pro
 - a. **MountainShed** Feature Class (lua.LAIKA.MountainSheds)
 - b. **BaseMap** Terrain Raster (NED_mosaic)
2. Buffer (Analysis Tools) Mountaished
 - a. This tool produces a feature class **buffer** with each item being buffered
 - -100ft is appropriate for this scale (negative to shrink the polygon)

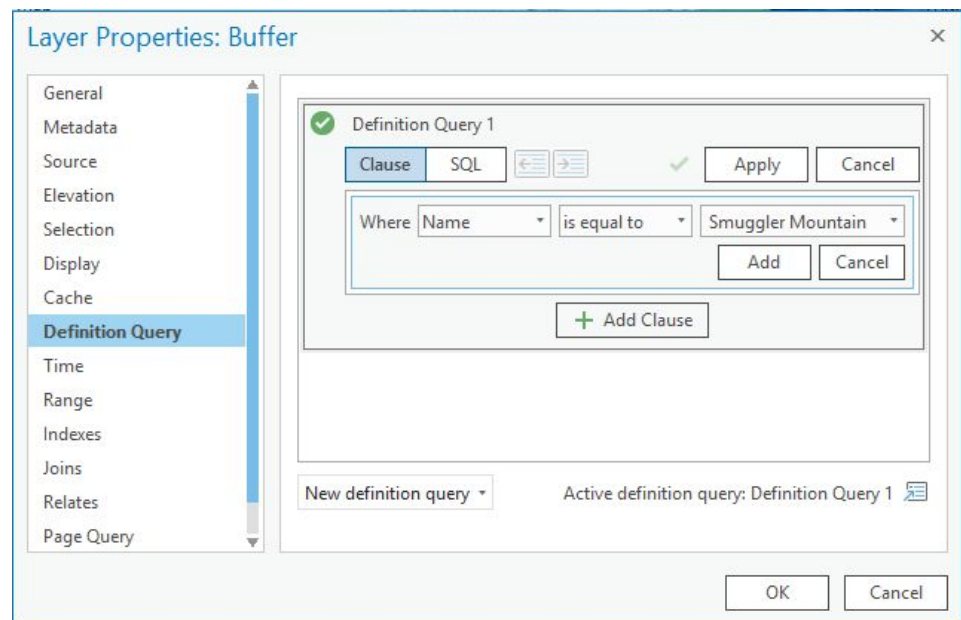


b.

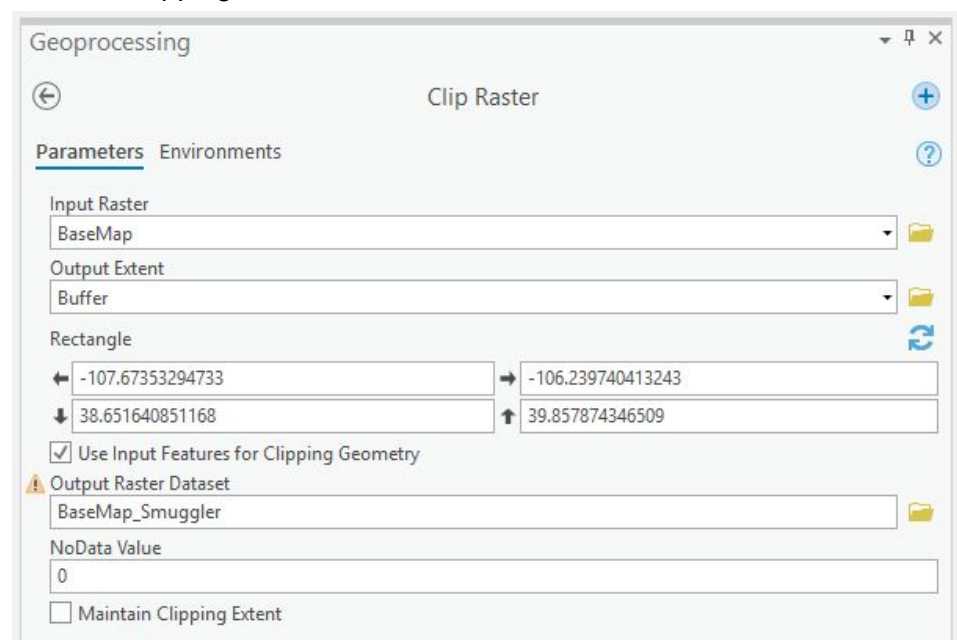


c.

3. Foreach Mountain Shed polygon...
 - a. Set the definition query on **Buffer** to isolate specific piece
 - i.e. {Name}='Smuggler'



- b. Clip Raster (Data Management Tools) **BaseMap** to **Buffer**
 - Input Raster = **BaseMap**
 - Output Extent = **Buffer**
 - Use Input Features for Clipping Geometry - YES
 - Output Raster = **BaseMap_{NAME}**
 - NoData Value = 0
 - Maintain Clipping Extent = FALSE



c. Export Raster as TIFF

- Input Raster = **BaseMap_{NAME}**
- Output Raster = **{NAME}.tif**
- Coordinate System = MountainSheds
- Geometric Transformations = None
- Clipping Geometry = Default
- Resolution (Cell Size): X = 100, Y = 100
- Pixel Type = 32 Bit float
- NoData Value = 0
- Output Format = TIFF
- Compression Type = None

Export Raster

BaseMap_Smuggler

General Settings

Output Raster Dataset

C:\Users\andrew.shewmaker\Documents\ArcGIS\Projects\MountainShed-3\Smuggler.tif

Coordinate System

NAD_1983_StatePlane_Colorado_Central_FIPS_0502_Feet

Geographic Transformations

Clipping Geometry

Default

Cell Size

X 100 Y 100

Raster Size

Columns 496 Rows 366

Pixel Type

32 Bit float

NoData value

0

▼ Renderer Settings

☐ Force RGB

☐ Use Colormap

☐ Use Renderer

Output Format

TIFF

Compression Type

None


Compression Quality

Export

Script Workflow

1. Run TIFF through python scripts to produce 3D model as STL file
 - a. Python scripts are posted to GitHub
 - `phstl.py`
 - `printv4.py`
 - b. **`python phstl.py -m 1000 {name}.tif {name}.stl`**
 - Ignore values below 1000ft
 - Do not scale the output in X or Y

```
run o-help for a list of available commands
C:\OSGeo4W64>python phstl.py -m 1000 Smuggler.tif Smuggler.stl
0...10...20...30...40...50...60...70...80...90...100 - done.
C:\OSGeo4W64>
```

- c. 
- d. `blender untitled.blend --background --python printv4.1.py`
`C:\OSGeo4W64\{name}.stl`
 - This script must be run in the Blender directory
 1. C:\Program Files\Blender Foundation\Blender

[illegible]

- f. Result is a 3D model of the piece, named **{NAME}-3D.stl**

3D Printing Workflow

1. Setup Slice Profile
 - a. To change color at the correct elevation, our slice profile is programmed to pause at a specific height/layer (GCode command: M2000), at which point we manually swap the filament to a new color and resume the print.
2. Post-Process 3D Model
 - a. Import 3D.stl into IdeaMaker, scale down, and align with other pieces
 - Scale : X = 0.20%, Y = 0.20%, Z = 1.00% (for our example)
 - Move into X-Y location, then adjust Z down for vertical alignment
 - b. FreeCut: After aligning the top of the model with the rest of the pieces, use the FreeCut tool at Z=0.00, and delete excess material (below Z=0)
 - c. Export the scaled model from IdeaMaker
 - d. Import the scaled model into Blender, and use the MakeFace tool to cleanup the base geometry
 - Check for internal anomalies
 - e. Export the cleaned model from Blender
 - f. Import the cleaned model into XYZMaker, along with Magnet-v#.stl
 - g. Create copies and hole out the magnet(s)
 - h. Export final model from XYZMaker as {Name}-{v#}.stl
 - i. Import the final model into the overall assembly, and organize files in ArcGIS Pro
 - j. Result is an STL file ready for printing, named {NAME}-v#.stl
3. Print with IdeaMaker
 - a. Import file(s) and arrange on the platform
 - b. Pre-Print Check
 - Confirm same model aligns with the overall assembly
 - Check for magnet holes in the base
 - c. Slice using the "Terrain-Huge" profile (including layer pause points)
 - d. Cloud Print
 - Export gcode on computer, and upload to RaiseCloud
 - Clean print bed, load green filament, and adjust nozzle height
 - Start print remotely, planning to be in the office for color switches

Additional Development/Testing

- TIF Export
 - Cell Size
 - Smaller cell size equals more cells
 - More cells equals larger file size, and increased detail in the model
 - Q: How does cell size translate to model size?
 - NoData Value = 4000'
- Control relative sizing: scale should be obvious and derived from the GIS data workflow
 - Phstl.py
 - -m 5000 (ignore points below 5000' elevation)
 - -x 250 (set output width to 250mm)
 - Blender.py
 - BaseHeight: currently it's calculated, but we could try to hard-code the value for consistency within the same project.
 - Terrain model should be same scale as Perimeter models
 - If we could output all TIF files clipped to the same extent, we should be able to control the model sizes
 - Perimeter models should then include a flat base
 - Base of all models (including terrain) should line up in X,Y and Z
- Create 3D Models for each (selected?) Polygon in a FeatureClass
 - Extend ArcGIS Pro with a Custom Tool
- Develop a batch file that iterates through a folder of TIF files
 - Requirements
 - All tif files must be the same extent?
 - Foreach TIF in DIR: call phstl.py
 - Foreach STL in DIR: call blender.py