Section 22: Python Scripting with Spatial Modeler

Section Objective

The ERDAS IMAGINE® Spatial Modeler (2013 release and later) contains capabilities for creating and executing spatial models in the Python® scripting language, and for embedding a Python script in a spatial model. The 2015 release supports 32-bit installations of Python 2.7 only. Almost any spatial model that can be constructed in the Spatial Model Editor can also be created from a Python script, or a previously created model can be loaded from a .gmdx file. Model inputs can be set from Python, the model can be executed in the Python process, and output values can be read.

The Spatial Model Editor also provides a Python operator that can execute any Python script whose inputs and output are the simplest data types: text string, integer, or floating-point number. This document assumes that the reader is familiar with the basic concepts and operation of the Spatial Modeler Editor.

Tools Used

Python IDLE	3rd Party Scripting Tool to incorporate into IMAGINE Spatial Modeler
Python Operator	Used to input and output a python script into and out of Spatial Modeler

Class Notes

Class Notes

Python Scripting with Spatial Modeler

Task 1: Configuring Python for Use with Spatial Modeler

For reference, the ERDAS IMAGINE Python libraries are installed in:

$IMAGINE_HOME\usr\lib\Win32Release\python

1. To test if the ERDAS IMAGINE Python libraries have been configured correctly, open Python 2.7 > IDLE (Python GUI) from the start menu.
2. Go to File > New Window.
3. Save the script as **test.py.**
4. To load the Spatial Modeler Libraries type the following:

```
# Load the Spatial Modeler Libraries
from imagine import modeler

print "Done"
```

5. Save the script by going to File > Save (Ctrl + s)
6. Run the script by going to Run > Run Model (F5)
7. Your script will take several seconds to process while it loads the ERDAS IMAGINE Python libraries.
8. If your Script says `done` then your libraries have been loaded and you can move on to Task 2.
9. If you received an error message, please try the following steps.
10. Open ERDAS IMAGINE.
11. From the Help tab search for 'python'.
12. Click the button **Reconfigure PYTHON.**

13. Run **test.py** again.
14. If you are still having issues loading the libraries you will need to check your environment variables to ensure PYTHONPATH is set to the location of your python. Eg. **$IMAGINE_HOME\usr\lib\Win32Release\python**

Operator Limitations

The following Python syntax lists all the operators directly available to Python:

```
from imagine import modeler
modeler.list()
```

The syntax for a given operator is documented in the Spatial Model Editor online-help. Operator spelling and syntax is also available via the default Python IDLE extensions: AutoComplete and CallTips.

Task 2: Summing Image Layers

This exercise will focus on building a script to sum the bands of a Landsat dataset

The Spatial Modeler libraries are accessed in Python by loading the IMAGINE.Modeler library:

```
from imagine import modeler
```

We need to set the input and output data path directories. Your directories may be different to those below. You may need to create an 'Outputs' folder.

```
dataPath = "C:\\python-examples\\data\\"
outputPath = "C:\\python-examples\\data\\Outputs\\"
```

A spatial model object must be created to hold operators:

```
m = modeler.Model()
```

Operators are created from the model itself. Each parameter of the constructor makes a connection to an input port on the operator or provides a constant value to the port. The following statement creates a Raster Input operator and sets its first input to the desired filename:

```
ri = m.RasterInput(dataPath + "lanier.img")
```

Rather than constant values, as above, a previously-created operator can be passed as an input to the next operator. If the operator passed in has more than one output, its first output is connected.

```
bandSelect = m.BandSelection(ri, "1:4")
```

```
We need to set a Raster Output location which writes
sumBands to the outputPath and
Ro = m.RasterOutput(sumBands, outputPath + "summing-image-
layers.img")
```

Your completed script should look similar to the one below;

```
# Load the Spatial Modeler libraries
from imagine import modeler

# Get directories for input and output data -
dataPath = "C:\\python-examples\\data\\"
outputPath = "C:\\\\python-examples\\data\\Outputs\\"

# Create a spatial model
m = modeler.Model()

# Add operators to the model
ri = m.RasterInput(dataPath + "lanier.img") # open image
bandSelect = m.BandSelection(ri, "1:4")     # select bands 1-4
sumBands = m.StackTotal(bandSelect)         # sum all the selected bands
ro = m.RasterOutput(sumBands, outputPath + "summing-image-layers.img")

# Run the model
m.Execute()
print "Done!"
```

Optional -

It is not necessary to keep a variable reference to the created operators; the Model object keeps track of them.

If we wanted to we could also shorten the script by changing the operator creation:

```
# Load the Spatial Modeler libraries
from imagine import modeler

# Get directories for input and output data -
dataPath = "C:\\python-examples\\data\\"
outputPath = "C:\\\\python-examples\\data\\Outputs\\"

# Create a spatial model
m = modeler.Model()
m.RasterOutput(m.StackTotal(m.BandSelection(m.RasterInput(dataPath + "lanier.img"), \
                                            "1:4")), \
               outputPath + "summing-image-layers.img")

# Run the model
m.Execute()
print 'Done!'
```

Task 3: Preview an Output in ERDAS IMAGINE

One unique capability of the Spatial Modeler is the ability to display a dynamic preview In this exercise we will modify the previous script to produce a dynamic preview in ERDAS IMAGINE.

1. In order to use the preview, ERDAS IMAGINE must be running. Open ERDAS IMAGINE from the start menu.
2. Leave the remaining script as is but remove or comment out the m.Execute statement.
3. Insert the following syntax

```
m.Preview(ro)
```

4. The “ro” in the parentheses is the output file we wish to preview. We could also change this to “ri” (rasterInput) if we wanted to preview the input dataset.
5. Run the script and view the Preview in IMAGINE.

Task 4: Saving a python script as a Spatial Model (*.gmdx)

Spatial models can be saved to disk, or loaded from a saved file. These include models ccreated in the Spatial Model Editor. In some cases it may be more convenient to create a complex model in the editor and execute it later from Python.

```
# Load the Spatial Modeler libraries
from imagine import modeler

# Get directories for input and output data -
dataPath = "C:\\python-examples\\data\\"
outputPath = "C:\\\\python-examples\\data\\Outputs\\"

# Create a spatial model
m = modeler.Model()

# Add operators to the model
ri = m.RasterInput(dataPath + "lanier.img") # open image
bandSelect = m.BandSelection(ri, "1:4")     # select bands
1-4
sumBands = m.StackTotal(bandSelect)         # sum all the
selected bands
ro = m.RasterOutput(sumBands, outputPath + "summing-image-
layers.img")
```

```
# Save the model
outputFile = outputPath + "sum-layers.gmdx"
modeler.Solution.Save(m, outputFile)
print "Saved spatial model to: " + outputFile
```

Create a variable called outputFile assign the name of the sum-layers Spatial Model (*.gmdx)

```
outputFile = outputPath + "sum-layers.gmdx"
```

Add the following to save the model ‘m’ to the OutputFile

```
modeler.Solution.Save(m, outputFile)
```

Add a print command to verify the script has completed.

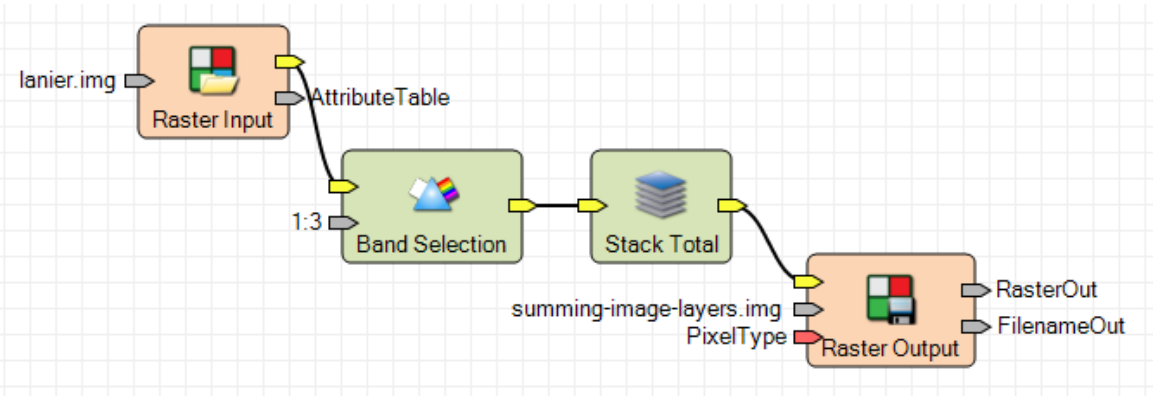
```
print "Saved spatial model to: " + outputFile
```

Run the script.

We will now verify if the model has been generated correctly.

1. Open ERDAS IMAGINE from Start > All Programs > ERDAS IMAGINE 20XX > ERDAS IMAGINE 20XX.
2. Create a new spatial model by going to File > New > Spatial Model Editor.
3. Open the Spatial Model by going to File > Open Spatial Model.
4. Navigate to your Outputs directory and select **sum-layers.gmdx**.

   Your model should look similar the one below.



```
# Load the Spatial Modeler libraries
from imagine import modeler

# Get directories for input and output data -
dataPath = "C:\\python-examples\\data\\"
outputPath = "C:\\\\python-examples\\data\\Outputs\\"

# Create a spatial model
m = modeler.Model()

# Add operators to the model
ri = m.RasterInput(dataPath + "lanier.img") # open image
bandSelect = m.BandSelection(ri, "1:4")     # select bands 1-4
sumBands = m.StackTotal(bandSelect)         # sum all the selected bands
ro = m.RasterOutput(sumBands, outputPath + "summing-image-layers.img")

# Save the model
outputFile = outputPath + "sum-layers.gmdx"
modeler.Solution.Save(m, outputFile)

print "Done!"
```

Task 5: Loading and Executing a Saved Model

Spatial models can be saved to disk, or loaded from a saved file. These include models created in the Spatial Model Editor; in some cases it may be more convenient to create a complex model in the editor and execute it later from Python We can also directly load a Spatial Model (*.gmdx) into a Python script.

This script will load the sum-layers.gmdx and write the summing-image-layers.img dataset to the output directory again.

```
from imagine import modeler

# Get directories for output data –
outputPath = "C:\\\\python-examples\\data\\Outputs\\"

# Load a saved model
modelFile = outputPath + "sum-layers.gmdx"
m = modeler.Solution.Load(modelFile)

# Run the model
m.Execute()
print "Finished running " + modelFile
```

Set the output directory. In this case we will use the Output location as the Input location, as this is where our sum-layers.gmdx model resides.

```
outputPath = "C:\\\\python-examples\\data\\Outputs\\"
```

We will Create a variable called modelFile and set the outputPath and the sum-layers spatial model.

```
modelFile = outputPath + "sum-layers.gmdx"
```

This time we will use the Solution.Load command to load the spatial model.

```
m = modeler.Solution.Load(modelFile)
```

Add a print statement to confirm he model has completed.

```
m.Execute()
print "Finished running " + modelFile
```

Run the script

Check your Outputs director to see if summing-image-layers.img has been re-written.

If you receive an error, you may need to delete the pre-existing image-layers.img file.

```
from imagine import modeler

# Get directories for output data -
outputPath = "C:\\python-examples\\data\\Outputs\\"

# Load a saved model
modelFile = outputPath + "sum-layers.gmdx"
m = modeler.Solution.Load(modelFile)

# Run the model
m.Execute()
print "Finished running " + modelFile
```

Task 6: Setting a No Data region on an image

We will now look at an example which combines both raster and vector data. This example will use a vector file to apply a NoData region to a raster. This exercise uses the lanier.img raster and the lanier_no_data.shp vector.

1. Import the modeler, create the model and set the dataPath as we did in previous exercises.

```
# Import ERDAS IMAGINE's spatial modeling suite
from imagine import modeler

# Create the owner model
m = modeler.Model()

# Set the location of the input files
dataPath = "C:\\python-examples\\data\\"
```

2. Create a variable for an the lanier.img image and the lanier_no_data.shp vector.

```
# Set the input datasets
ri = m.RasterInput(dataPath + "lanier.img")
vi = m.VectorInput(dataPath + "lanier_no_data.shp")
```

3. Start off by using the Rasterize operator to rasterize the vector dataset. Place this within a variable called rasterizedVector.

```
rasterizedVector = m.Rasterize(vi)
```

4. Use a conditional statement to select only the pixels from lanier.img that fall within rasterizedVector

```
selectPixelsWithin = m.Conditional( rasterizedVector,
ri )
```

5. Use the SetNoData operator to set the pixels to “0”.

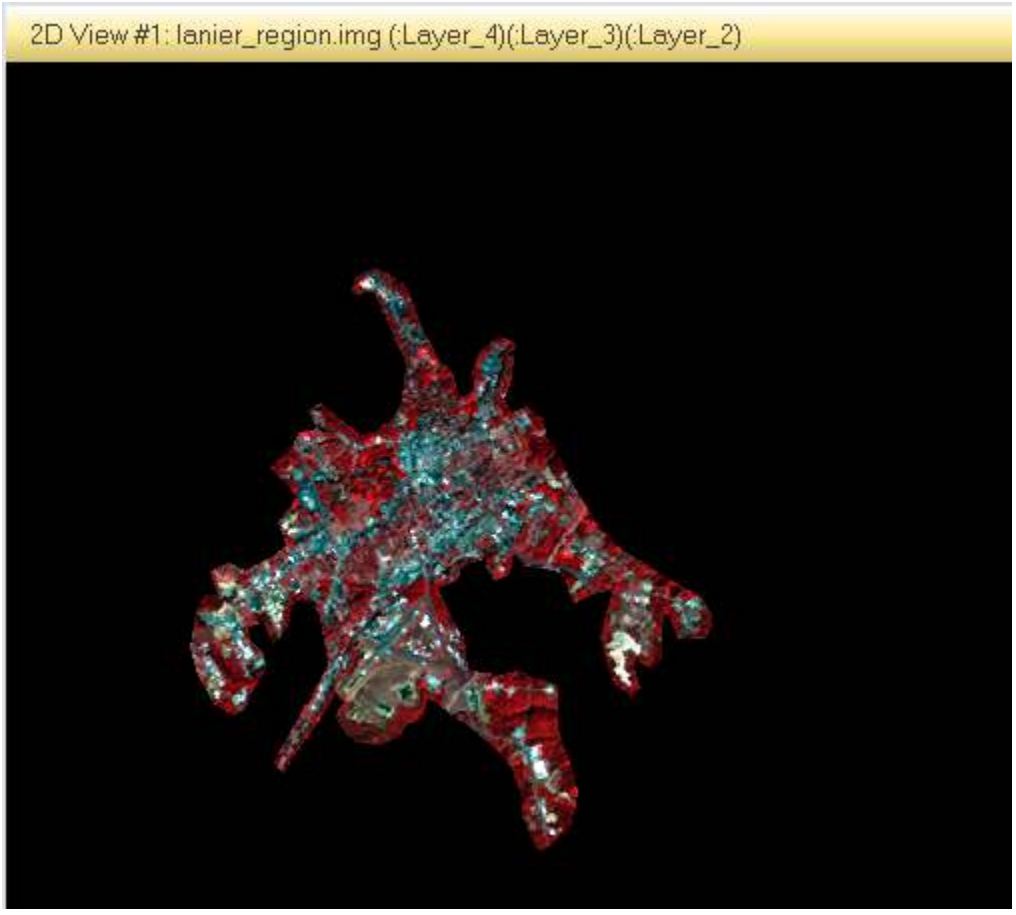
```
noDataRaster = m.SetToNoData( selectPixelsWithin, "0" )
```

6. Create the raster output and execute.

```
ro = m.RasterOutput (noDataRaster, dataPath +
"lanier_region.img")

m.Execute()
```

7. Run the script and assess the result in ERDAS IMAGINE



```
# Import ERDAS IMAGINE's spatial modeling suite
from imagine import modeler

# Create the owner model
m = modeler.Model()

# Set the location of the input files
dataPath = "C:\\python-examples\\data\\"

# Set the input datasets
ri = m.RasterInput(dataPath + "lanier.img")
vi = m.VectorInput(dataPath + "lanier_no_data.shp")

# Rasterize the vector
rasterizedVector = m.Rasterize(vi)

# Use a conditional statement to select only the pixels with rasterized Vector
selectPixelsWithin = m.Conditional( rasterizedVector, ri )

# Set the area the pixels didnt meet the conditional to 0
noDataRaster = m.SetToNoData( selectPixelsWithin, "0" )

ro = m.RasterOutput (noDataRaster, dataPath + "lanier_region.img", 'u8')

m.Execute()

print "Done"
```

Task 7: Modifying a Spatial Model

A Model object loaded in Python can be modified; most commonly, to change parameters of operators. The following example shows how to examine the contents of a model, add operators, and read and modify values.

```
from imagine import modeler
from exampleshelper import dataPath, outputPath

# Load a saved model
modelFile = outputPath + "sum-layers.gmdx"
m = modeler.Solution.Load(modelFile)

# Display everything
print m
for op in m.GetOperators():
     print op
     for port in op.GetPorts():
     print port

# Find a specific operator
rasterInput = next(op for op in m.GetOperators() \
                   if op.name == u"RasterInput")

# Change an input to that operator
rasterInput["Filename"] = dataPath + "/wasia1-
65535.img"

# Add a new operator to the loaded model
stats = m.Statistics(rasterInput)

# Add a TableSubset operator to the loaded model
tableSub = m.TableSubset(stats["Max"], "2:2")

# Read an output value of the added operator,
implicitly causing it

# to be executed.
band3Max = tableSub["TableOut"].data
print 'Maximum value of band 3 is: ' + str(band3Max)
```

Task 8: Viewing a Spatial Model in the Editor

The following kinds of objects can be exchanged between Python and Spatial Modeler:

- Text strings (including file and directory names)
- Integers and unsigned numbers
- Floating point numbers
- Boolean flags
- Number ranges (Spatial Modeler's Range data type)
- Lists of ranges (Spatial Modeler's RangeList data type)

Interaction with ERDAS IMAGINE

Spatial Models can be run from Python without opening an interactive window; however, if an ERDAS IMAGINE workspace is running, a model can be viewed in the Spatial Modeler Editor or previewed in the 2D View.

```
from imagine import modeler
from exampleshelper import dataPath, outputPath

# Load a saved model
modelFile = outputPath + "sum-layers.gmdx"
m = modeler.Solution.Load(modelFile)

# Display everything
print m
for op in m.GetOperators():
     print op
     for port in op.GetPorts():
          print port

# Add a new operator to the loaded model
rasterInput = next(op for op in m.GetOperators() \
 if op.name == u"RasterInput")
stats = m.Statistics(rasterInput)

# Open the modified model in the editor. Note that
changes made in the

# editor are not reflected in "m".
# ERDAS IMAGINE must be running!
m.ShowInEditor()
```

Class Notes