

Background:

Hydrology Modelling requires either the 3D or Spatial Analyst extensions to successfully assess flood scenarios on a DEM. Licensing this with ESRI products can be expensive but other GRID based packages such as IDRISI can accomplish the following procedures more

To model floods we need a DEM and the ability to determine or predict flood levels.

Floods fall into 2 categories:

1. Storm Surge or Inundation Models with a constant elevation (flat-water “static” condition)
2. River system models with a flood gradient (tilted flood plane – dynamic condition)

Modelling the 2 requires different strategies.

Generating Sucessive Flood Stages:

The following model is based on an ArcView 3.x Avenue script authored by Bill Huber, a frequent contributor to the ESRI User Forums.

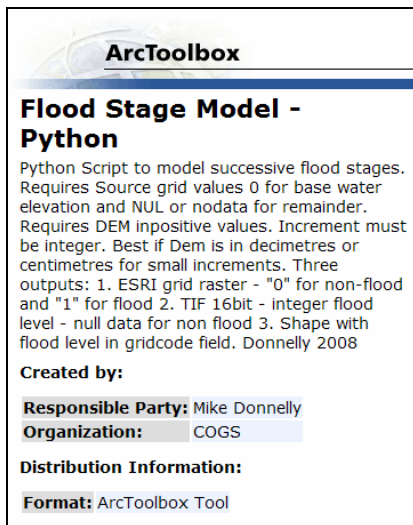
The sample script was used for the Red River Valley Flood by Cameron Bouchard, Operations Support Officer, Government Emergency Operations Coordination Centre, Ottawa.

The script has been translated to Python and considerably enhanced with calls to ArcGIS Geoprocessing functions.

The inputs have been set up as parameters in an ArcToolBox Script and can be added to arc toolbox and run with the attached Python code.

This particular version requires the Spatial Analyst Extension.

Example for the tool:



ArcToolbox

Flood Stage Model - Python

Python Script to model successive flood stages. Requires Source grid values 0 for base water elevation and NUL or nodata for remainder. Requires DEM in positive values. Increment must be integer. Best if Dem is in decimetres or centimetres for small increments. Three outputs: 1. ESRI grid raster - "0" for non-flood and "1" for flood 2. TIF 16bit - integer flood level - null data for non flood 3. Shape with flood level in gridcode field. Donnelly 2008

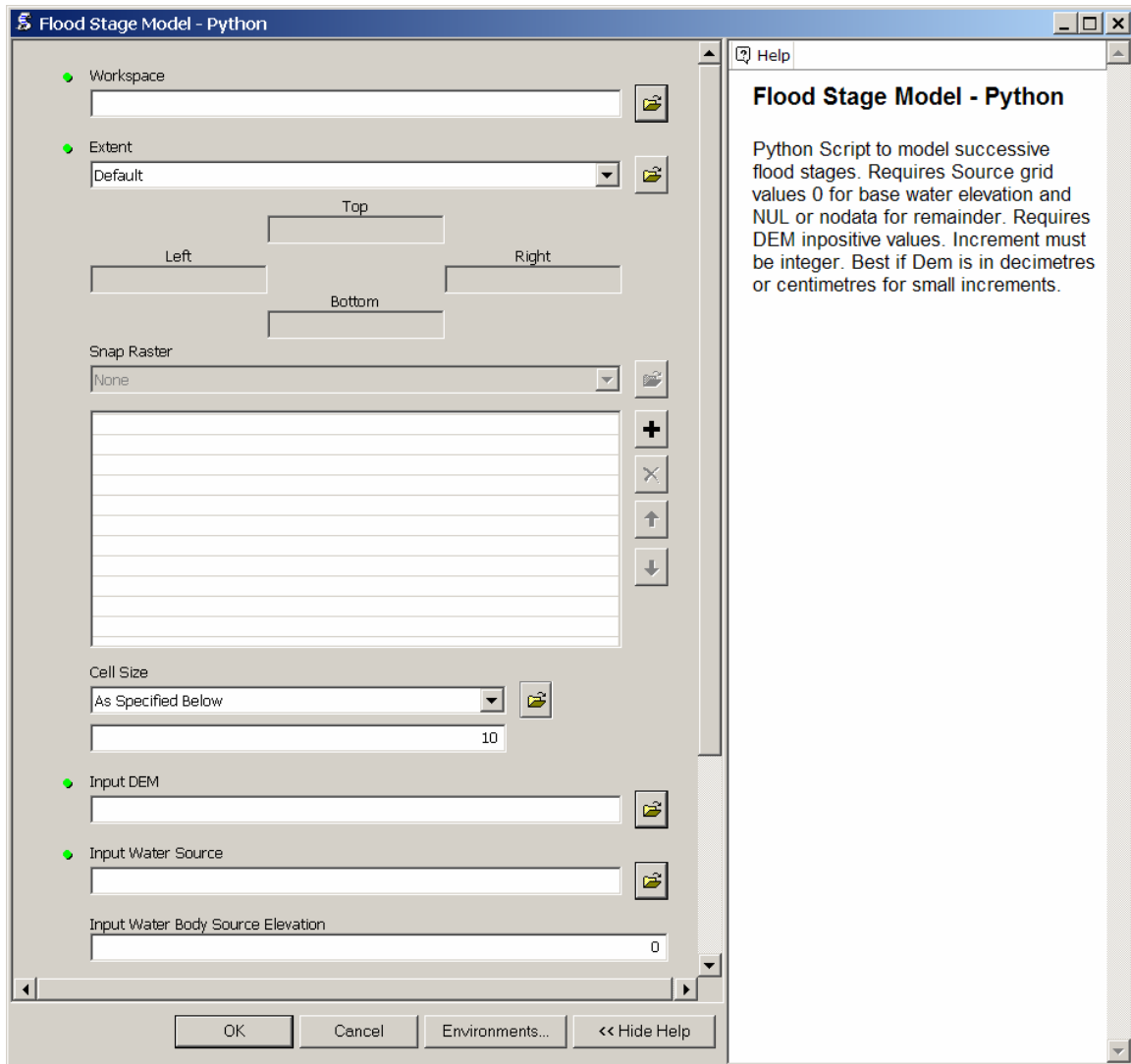
Created by:

Responsible Party: Mike Donnelly
Organization: COGS

Distribution Information:

Format: ArcToolbox Tool

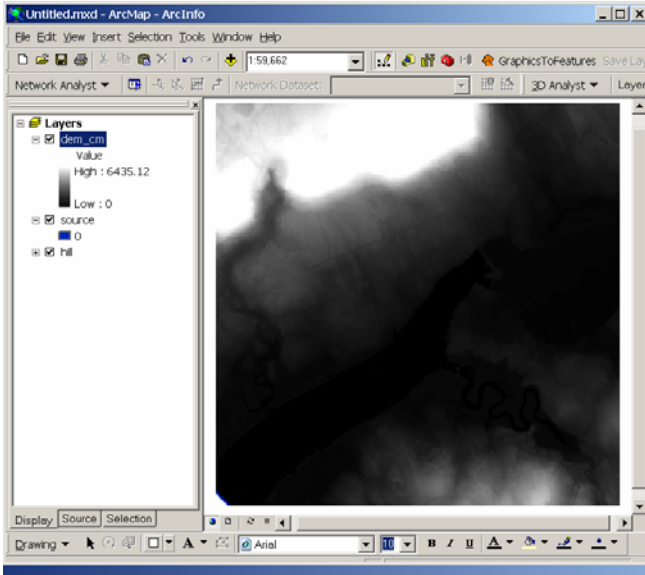
Tool Interface:



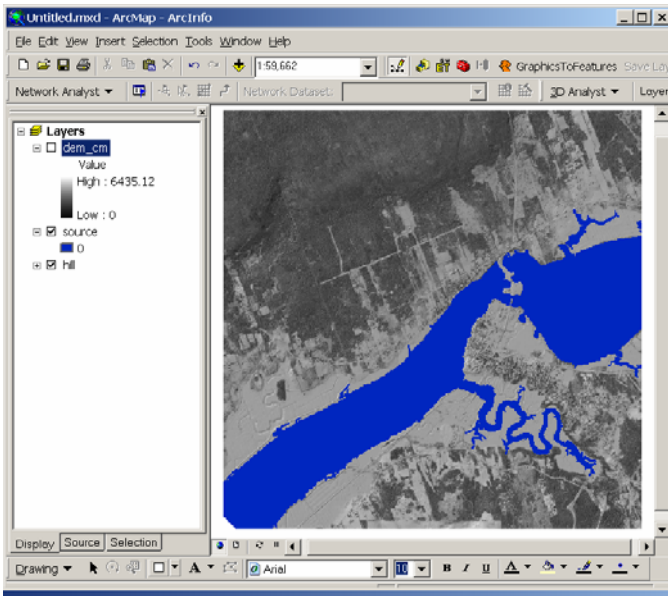
Test data for Annapolis:

The Test DEM is derived from the AGRG (Applied Geomatics Research Group) LiDAR data from 2004. The dem was converted to centimetres by multiplying elevations by 100 using the raster calculator.

Input: dem_cm



The “source” of the flood (the Annapolis Basin) was derived from NSGC water polygon data. It was converted to raster with a value of 1 for water and null or nodata for the remainder. A hillshade of the DEM is included for context.



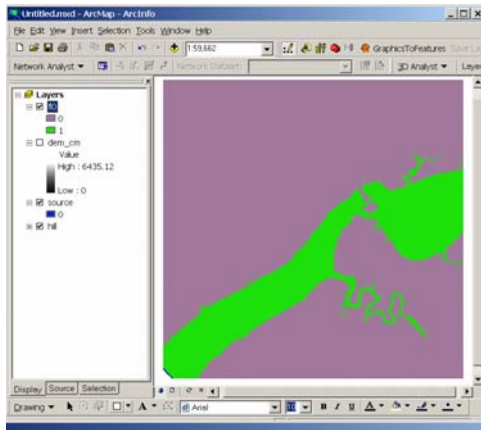
Inputs required:

1. Workspace: use a copy of session6\model on the local computer
2. Extent – pick an existing raster or specify
3. Cell Size – Pick an existing raster or specify. Coarser grids process much more quickly.
4. Input dem: use dem_cm
5. Source: use source
6. Input water elevation: use 0
7. Flood increment: Use 10 – for 10cm
8. Maximum flood level: Use 100 for 100cm
9. Extent: use 200 – this is to simulate flood inundation “creep” so for derived animations, the flood will not immediately fill an area but rather move or “creep” across the area being flooded.
- 10 The output base flood name: use fl

Outputs:

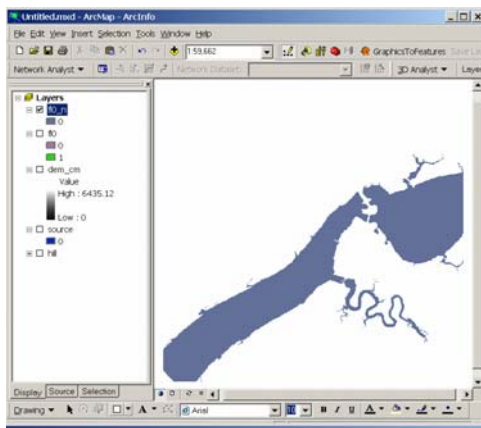
1. ESRI Integer grids:
fl0, fl10, fl20, fl30 etc

Values 1 for flood and 0 for non-flood



2. TIF 16bit integer rasters
fl0_n, f20_n, f30_n, f40_n etc

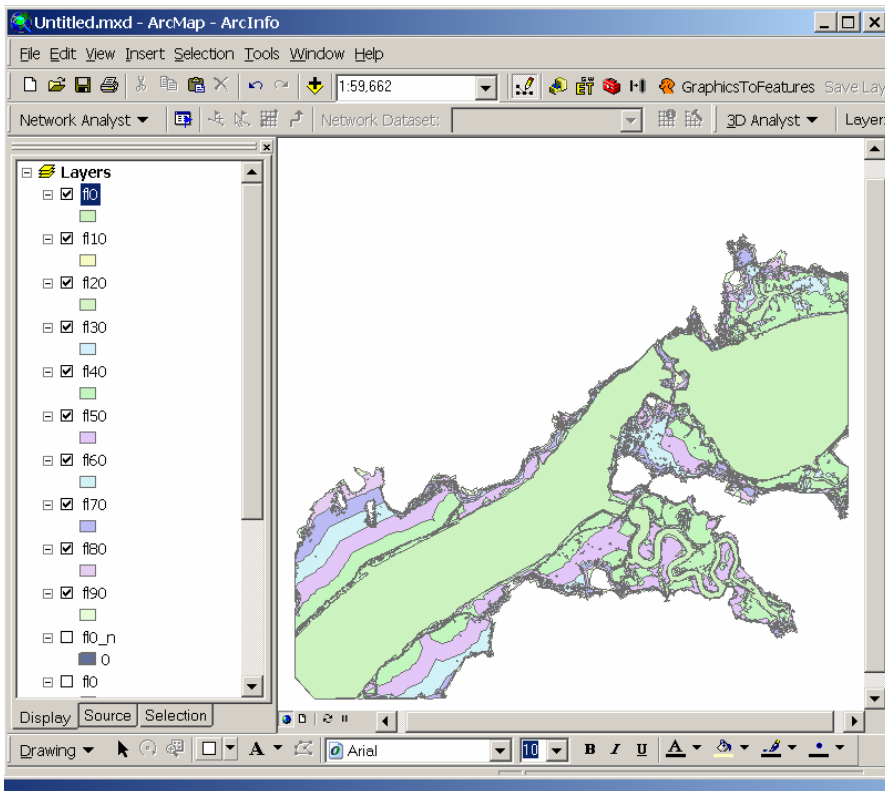
Elevation value in attribute table, remainder is classified as NULL or NODATA.



3. Shapes

f10.shp, f110.shp, f120.shp, f130.shp, f140.shp etc

Feature attribute table gridcode field contains the shape flood elevation.



Flood Scenarios can be animated in ArcMap or ArcScene

Tools such as CopyPasteRasterAllSym9.dll can be used to facilitate preparing an animation group of sequenced flood images.

Similar tools can be used to copy and paste shape symbology. Check out the ESRI User forums.

The TIFs or shapes can be used in ArcScene using actual elevation values while the ESRI grids can be used in ArcMAP.

Sample Animation Views:

1. ArcMap
2. ArcScene

DEMO AVI files are included:

Python Script:

```
# -----  
# flood_MA_test.py  
# Created by: Mike Donnelly  
# (generated by ArcGIS/ModelBuilder)  
# For AAPT 2008  
# Partially based on Huber's Avenue Model  
# Modified to create flood grids with elevation and shape files with GRIDCODE field set  
# to actual elevation.  
# My last Python Script!? 2008-05-07  
# -----  
  
# Import system modules  
import sys, string, os, arcgisscripting  
  
# Create the Geoprocessor object  
gp = arcgisscripting.create()  
  
# Check out any necessary licenses  
gp.CheckOutExtension("spatial")  
  
# Load required toolboxes...  
gp.AddToolbox("C:/Program Files/ArcGIS/ArcToolbox/Toolboxes/Spatial Analyst Tools.tbx")  
gp.AddToolbox("C:/Program Files/ArcGIS/ArcToolbox/Toolboxes/Data Management Tools.tbx")  
gp.AddToolbox("C:/Program Files/ArcGIS/ArcToolbox/Toolboxes/Conversion Tools.tbx")  
  
# Set the input workspace  
gp.workspace = sys.argv[1]  
  
#Set the Grid Extent  
gp.extent = sys.argv[2]  
  
# Set the Grid Cell Size  
gp.cellSize = sys.argv[3]  
  
# Set the Input DEM  
#theDEM = sys.argv[4]  
pDEM = gp.GetParameterAsText(3)  
  
#Set the Input Water Body Source  
#theSource = sys.argv[5]  
pSource = gp.GetParameterAsText(4)  
  
#Set the Input Source Water Elevation  
#nSourceEl = sys.argv[6]  
nSourceEl = gp.GetParameterAsText(5)  
  
#Set the Flood Increment  
#nIncrement = sys.argv[7]  
nIncrement = gp.GetParameterAsText(6)  
  
#Set the Maximum Flood Level  
#nFloodMax = sys.argv[8]  
nFloodMax = gp.GetParameterAsText(7)  
  
#Set the Maximum Flood Extent for each iteration  
#nExtent = sys.argv[9]  
nExtent = gp.GetParameterAsText(8)  
  
#Set the Output Grid base name  
#nOutGrid = sys.argv[10]  
pOutGrid = gp.GetParameterAsText(9)  
  
# Local variables...  
myFlood1 = pOutGrid  
nTiny = (0.9 / int(nExtent))  
min = int(nSourceEl)  
max = int(nFloodMax)  
inc = int(nIncrement)  
  
# Process: Single Output Map Algebra to create nOutGrid  
gp.SingleOutputMapAlgebra_sa(pSource, myFlood1)
```

```
# Loop for each map algebra costdistance command
for x in range(min,max,inc):
    myMapAlgebra1 = "CostDistance (" + myFlood1 + ", (" + pDEM + " > " + repr(x) + " ) +
" + repr(nTiny) + " , #, #, #) <= ( " + repr(nExtent) + " * " + repr(nTiny) + " ) "
    gp.SingleOutputMapAlgebra_sa(myMapAlgebra1, myFlood1 + repr(x) )
    gp.delete_management ( myFlood1 , myFlood1 )
    myMapAlgebra2 = "ln(" + myFlood1 + repr(x) + ")"
    gp.SingleOutputMapAlgebra_sa( myMapAlgebra2, myFlood1 )
    #Modifications to create NODATA for land, assign elevation and to convert to poly
    gp.Con_sa(myFlood1 + repr(x), repr(x), myFlood1 + repr(x) + "_1", "9999", "VALUE =
1")
    gp.SetNull_sa(myFlood1 + repr(x) + "_1", myFlood1 + repr(x) + "_1", myFlood1 +
repr(x) + "_n", "VALUE = 9999")
    gp.delete_management ( myFlood1 + repr(x) + "_1" , myFlood1 + repr(x) + "_1" )
    myoutshape = pOutGrid + repr(x) + ".shp"
    gp.RasterToPolygon_conversion( myFlood1 + repr(x) + "_n", myoutshape, "SIMPLIFY",
"VALUE")
```