

GIS 590

Oceanfront Viewshed Analysis

Surf City, NJ

Rachel Albritton

Fall 2013

Table of Contents

Background	2
Problem	2
Methods	2
Advanced Geospatial Analysis & Programming	2
Data Management	7
Script Tool	7
Web Application	9
Web Application	11
Documentation	13
Results	13
Script Tool Outputs	13
Web Application Outputs	15
Discussion	16
Script Tool	16
Web Application	16
Future Research	17
Appendix I. Preparing Data for Viewshed Analysis	18
Appendix II. Calculating Viewshed Methods	20
Appendix III. Raster Fix Python Script	23
Appendix IV. Oceanfront Viewshed Python Script	25
Appendix V. Web Application User Manual	33
Navigation	33
Task Bar	33
Appendix VI. Web Application Updates	38
Appendix V. Web Application Models & Scripts	50

Figures

Figure 1. Raster Fix tool interface	3
Figure 2. Oceanfront viewshed tool interface	5
Figure 3. Original Data Files	8
Figure 4. Processed Data Files.....	8
Figure 5. Output Data Files.....	8
Figure 6. PostgreSQL Database	9
Figure 7. Postgre Database Connection in ArcCatalog.....	10
Figure 8. Enterprise Geodatabase Structure.....	10
Figure 9. Splash/Welcome screen.....	11
Figure 10. Web application interface	12
Figure 11. Web Application: Search by address tool.....	12
Figure 12. Web Application: Search by parcel ID	12
Figure 13. Web Application: Graphical Search	13
Figure 14. Oceanfront viewshed output data	13
Figure 15. File naming convention	14
Figure 16. Oceanfront viewshed output viewshed example	14
Figure 17. Viewshed output table results.....	14
Figure 18. Web Application: Viewshed results	15
Figure 19. Web Application: Search by Address Results	15
Figure 20. Web Application: Search results from geometry tool	16

Background

Long Beach Island is an 18-mile barrier island located in Ocean County, New Jersey and contains six different municipalities. In 1999, the U.S. Army Corps of Engineers received federal support to build a 22-ft. dune across the entire length of the island to assist in storm protection, however, construction could only occur with the full cooperation of oceanfront landowners. As a result, only three sections of the island were complete by 2007.

Problem

Many of the oceanfront home owners are concerned about the loss of property value, claiming the dunes will obstruct their homes ocean view, and therefore drive down their property value. The purpose of this project was to build a custom GIS tool for the client that can be used to run a viewshed analysis for parcels in both 2005 (pre-construction) and 2010 (post-construction) at several project sites. The output of the tool will provide information on each parcels viewshed, in degrees, allowing for further evaluation of the effects of the dunes construction on residents oceanfront views.

In addition to the custom tool, a web application was designed that will allow users to search for a parcel by address, and conduct a pre and post dune construction viewshed analysis for that parcel. The results of the application present the user with both a visual output of the viewshed for each year as well as the parcels viewshed, in degrees, for each year.

Methods

Advanced Geospatial Analysis & Programming

The municipality of Surf City served as the study region for this project. This area contains 445 parcels that needed a viewshed analysis completed for both 2005 and 2010. Additionally, the client needed a tool that could be applied to other municipalities along Long Beach. To accomplish this, two custom GIS tools were created using python scripting. The first script tool, "raster fix", allows the user to pre-process lidar data needed for the viewshed, analysis. This simple script takes a floating point, non-grid raster and converts it to an integer based grid, raster (Figure 1).

The raster fix tool makes two assumptions.

Assumption #1: The tool assumes that the input raster is in a non-GRID format. This is common for lidar data which tends to be in a .tif or .sid format.

Assumption #2: The tool assumes that the raster is floating point. This is also common for lidar data.

Inputs: Raster Fix

This tool should only need to be ran once on a single raster, and takes two inputs

1. Elevation Raster (Non-Grid)
2. Output location where the new raster will be saved

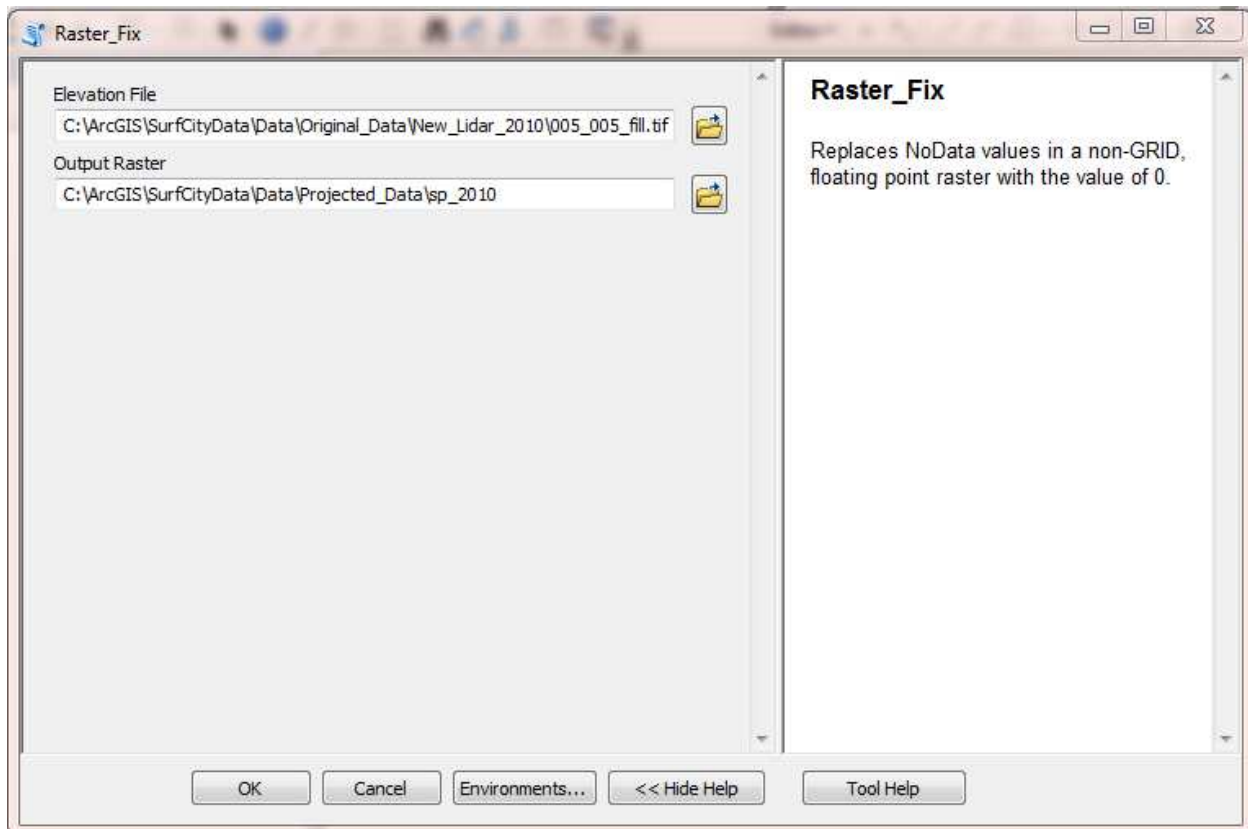
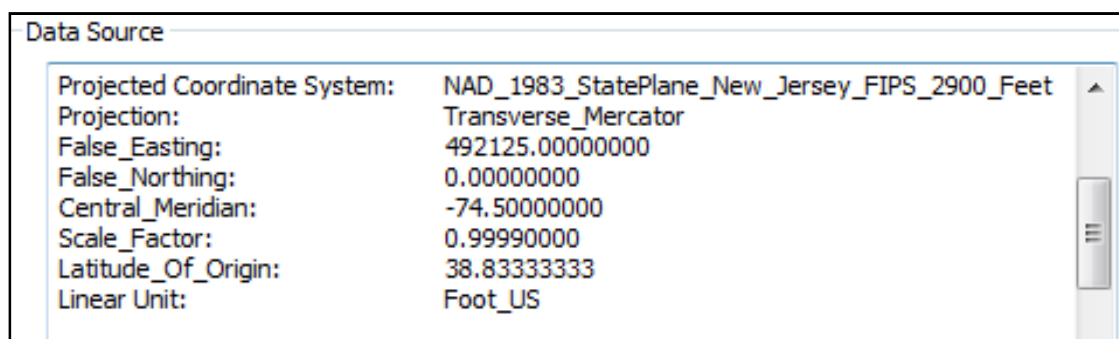


Figure 1. Raster Fix tool interface

The second tool, “oceanfront viewshed”, creates a viewshed at first floor (10ft.) and second floor heights (20 ft., where applicable) for each parcel. Any parcel with 1.5 floors or greater were given a second floor height viewshed.

The oceanfront viewshed tool has three assumptions, each are described below.

Assumption #1: This tool assumes that all data inputs have been projected into a coordinate system that has linear units.



If a shapefile or raster is projected in a geographic coordinate system that only has angular units, the results may be invalid, and the tool may fail completely.

Bottom	39.6254642037
<input type="checkbox"/> Spatial Reference	GCS_North_American_1983
Linear Unit	
Angular Unit	Degree (0.0174532925199433)
Datum	D_North_American_1983
<input type="checkbox"/> Statistics	
<input type="checkbox"/> Band_1	Statistics have not been calculated.
Build Parameters	

No linear units!

Assumption #2: This script also assumes that any null values that may be present within the elevation data have been converted to a value of 0. The Raster_Fix.py script located within this tool box will complete this task, and is discussed in more detail in the next section. This tool does assume that the original raster is in a non-GRID format (i.e .tif, .sid, .img).

Assumption #3: This tool assumes that the observation point shapefile already contains an attribute field containing the number of floors for each observation point. The user will be required to indicate this field within the tool.

Inputs

The oceanfront viewshed analysis tool takes nine inputs (Figure 2). Each input is described below

1. Workspace. This is typically where the data folder is located.
2. Output workspace, the folder where you want the results saved.
3. Observation points
4. Building footprints
5. Elevation
6. Bare Elevation
7. Ocean
8. Year the analysis is for
9. Floors. The field within the observation point feature class that contains the number of floors for each observation point.

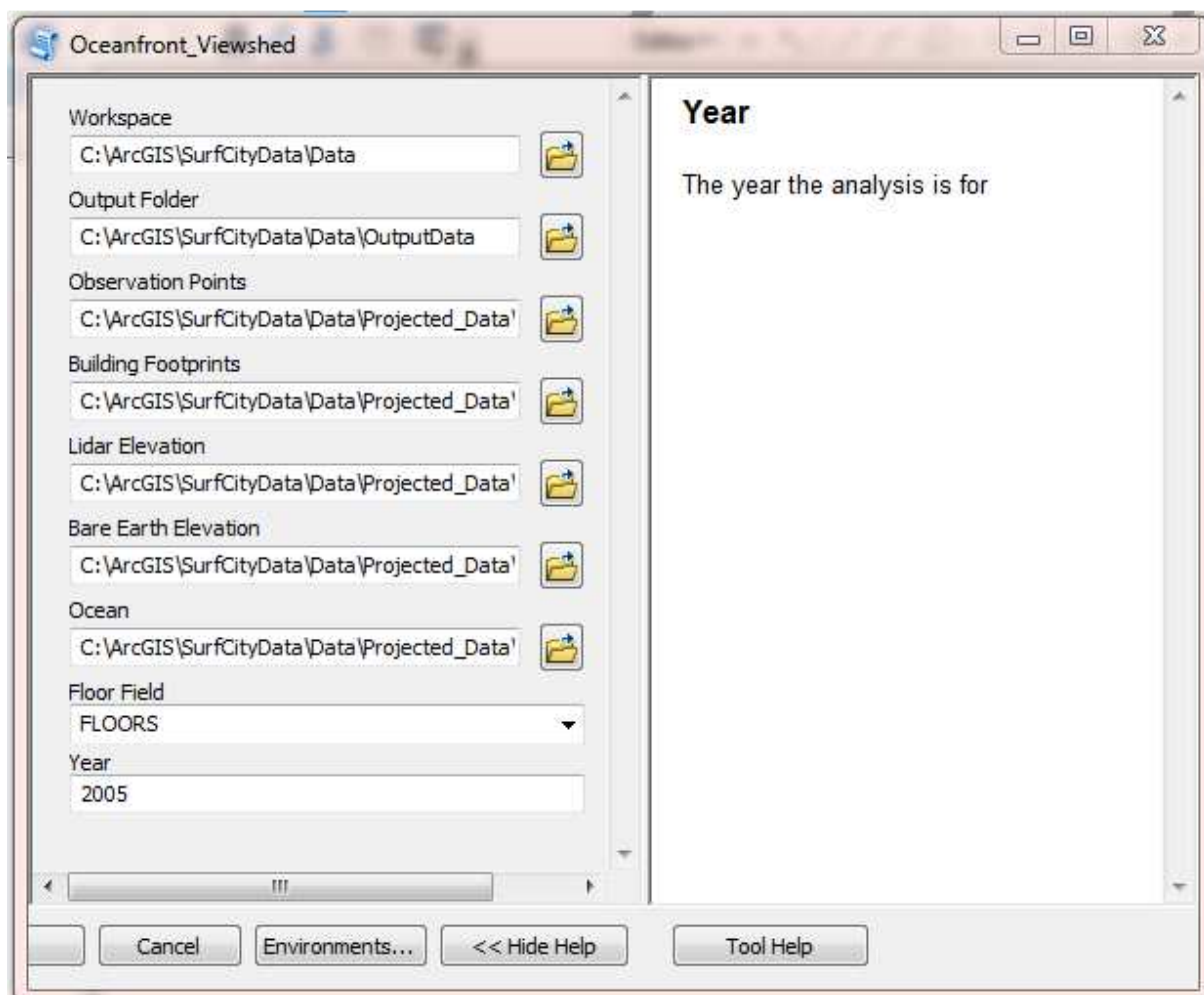


Figure 2. Oceanfront viewshed tool interface

The creation of elevation surfaces and viewsheds followed the methods proposed by Patterson and Boyle 2002; Bin et al., 2008; Morgan and Ashton, 2010; Hindsley et al. 2012; and Crawford et al., 2013.

For parcels 1 to n, complete the following for both first and second floors:



Following the guidance of previous research, a final view measure was created by extracting out the top portion of the viewshed arc for each viewshed output, and calculating the sum of viewable arcs. This arc length was divided by the total possible view radius of 3.14 miles and multiplied by the total possible view angle of 180 degrees.

$$\text{Viewshed} = (\text{Sum arc-lengths}/3.14)*180$$

Please refer to appendix II for a step by step guide with screenshots.

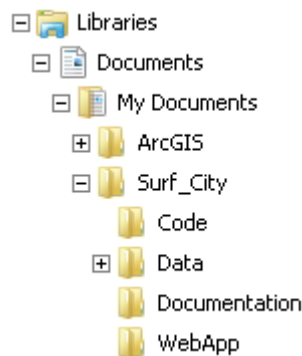
Data Management

Script Tool

Server ID: 152.46.17.82

Tool Data

Data Location: C:\Users\rrolbrit\Documents\Surf_City



The client will be running the script based on shapefiles, and did not have a need to manage the data within any type of RDBMS. The data folder contains three subfolders, original data, projected data, and output data. As the names imply, the original data was data provided at the beginning of the project along with any new data downloaded to perform the analysis for Surf City. Data in the original data folder was kept in the organization structure provided by the client. The projected data folder contains all preprocessed data needed to run the script tools. This data was organized by topic within file folders. The output data folder contains the script tool outputs for Surf City for both 2005 and 2010.

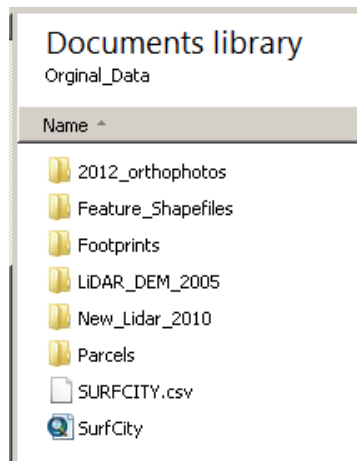


Figure 3. Original Data Files

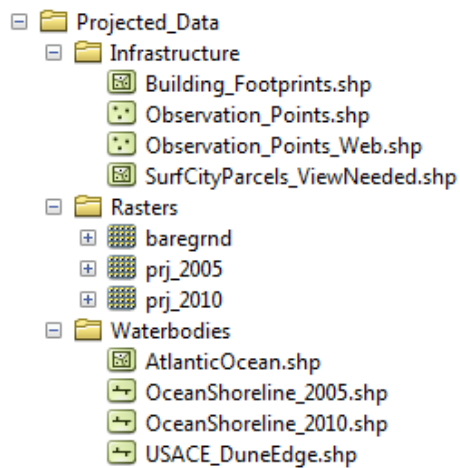


Figure 4. Processed Data Files

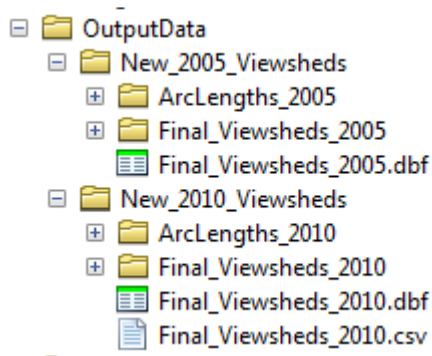


Figure 5. Output Data Files

Tool Documentation

A user guide for using the script tool is located within the documentation folder:

C:\Users\rrealbrit\Documents\Surf_City\Documentation

Tool Codes

The Code folder contains the two scripts created to run the script tools. Scripts are provided within the appendices and are located on the server here: **C:\Users\rrealbrit\Documents\Surf_City\Code**

Web Application

Data for the web application is stored in an enterprise geodatabase. There was only one table, containing one attribute field needed for the analysis so there wasn't a need to manage tabular data within postgres, however, as the project grows there may be a need to integrate tabular data in the future. To accommodate this possible future need as well as connect to an enterprise geodatabase, a postgres database was created (Figure 6).

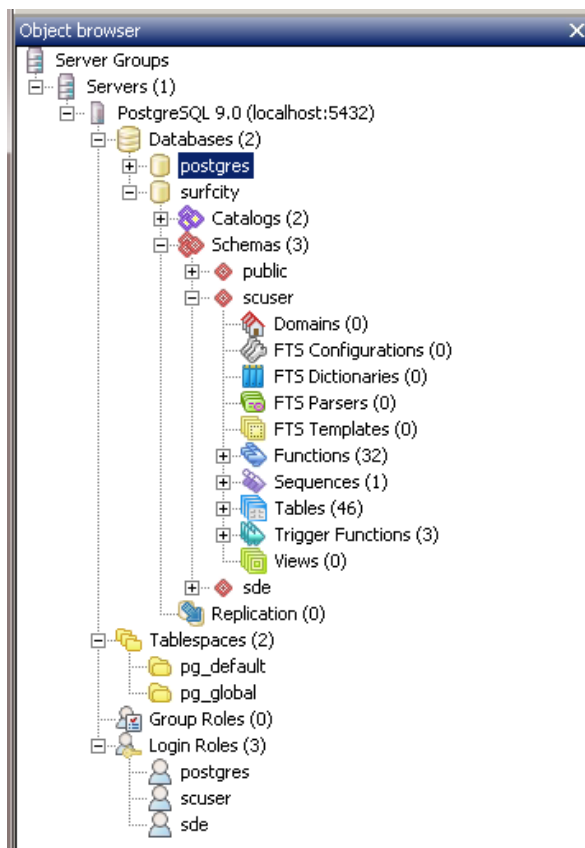


Figure 6. PostgreSQL Database

An enterprise geodatabase was created within ArcCatalog to store the spatial data needed for the web application (Figure 7, Figure 8).

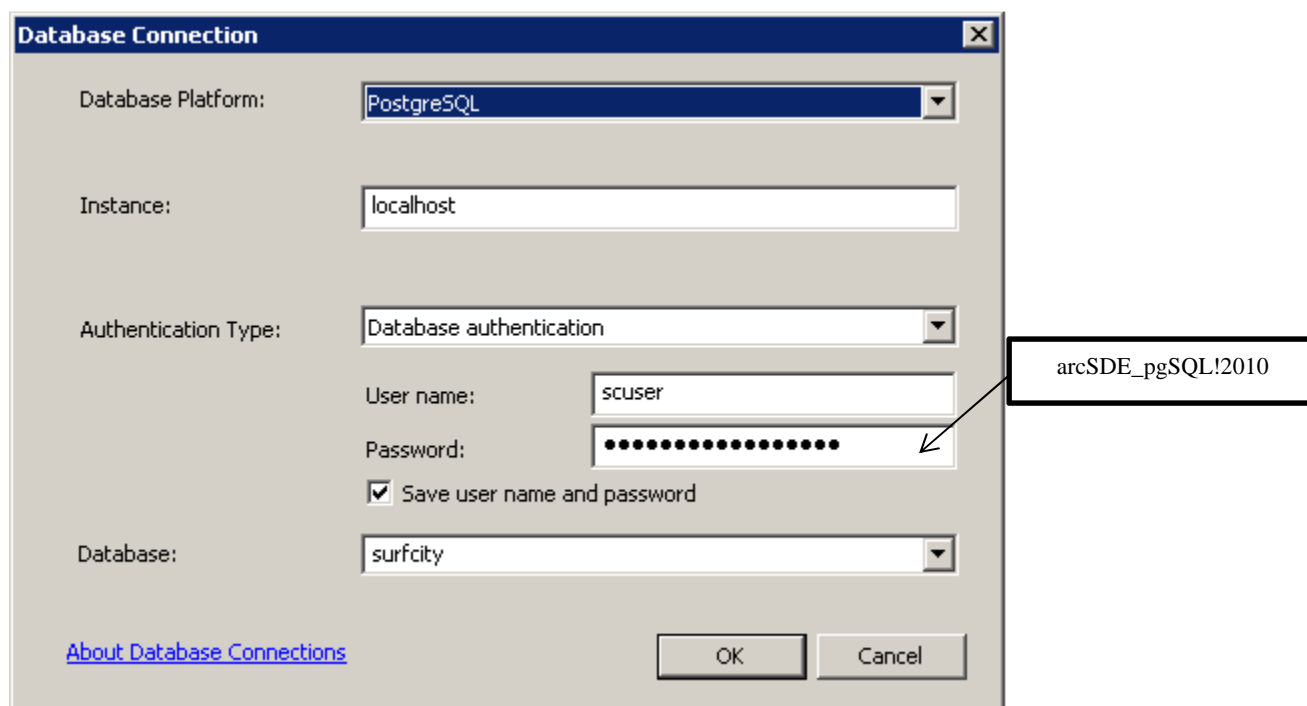


Figure 7. Postgre Database Connection in ArcCatalog

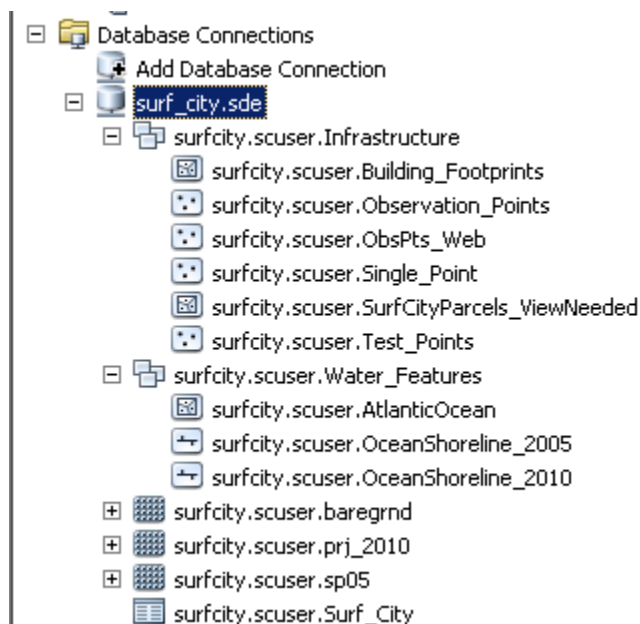


Figure 8. Enterprise Geodatabase Structure

Web Application

The purpose of the web application is to provide users with tabular results of a parcels viewshed, as well as provide a visual output of where the viewshed for that parcel exists at the first floor level.

- Web application services are hosted on ArcGIS Server Manager. Username and password information can be found on the server at C:\server class\Passwords.txt
- The web application was built using ArcGIS Viewer for Flex.
 - Server Location: C:\inetpub\wwwroot\flexviewer\Surf_City
 - URL: http://152.46.17.82/flexviewer/Surf_City/
 - Location of scripts and models for the web application:
C:\Users\rralbrit\Documents\Surf_City\WebApp
 - Documentation: C:\inetpub\wwwroot\flexviewer\Surf_City\Documentation
 - User Guide
 - Manual to update application
 - Copy of this final report

User Interaction

When the web application opens, the user is presented with a dialogue box that provides a brief summary of the web applications purpose, and a link to a user manual (Figure 9).

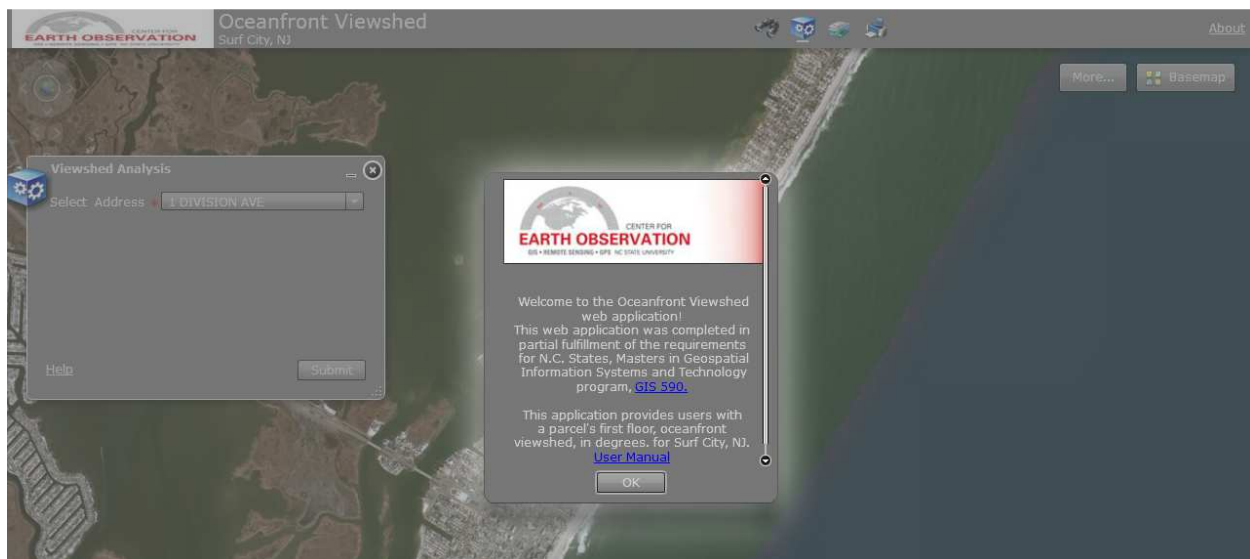


Figure 9. Splash/Welcome screen

Once the user clicks OK, there will be a dialogue box open for the user to run a viewshed analysis. The user can choose an address from the drop-down menu and then click OK. Addresses are in alpha, numerical order. The viewshed takes approximately 60-90 seconds to run (Figure 10).

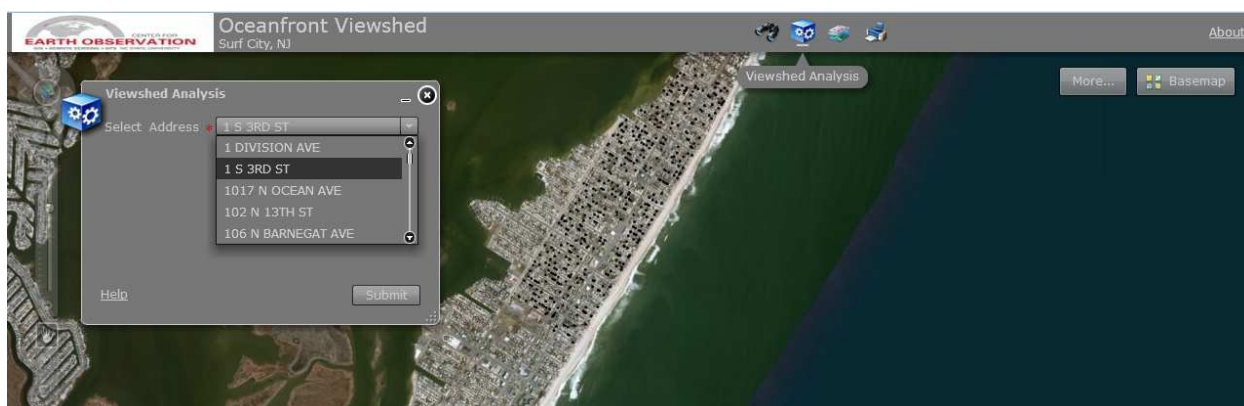


Figure 10. Web application interface

The user could also choose to search for a parcel using the search tool. This search tool allows users to search for a parcel by address (Figure 11), by parcel id (Figure 12), or by graphical search (Figure 13).

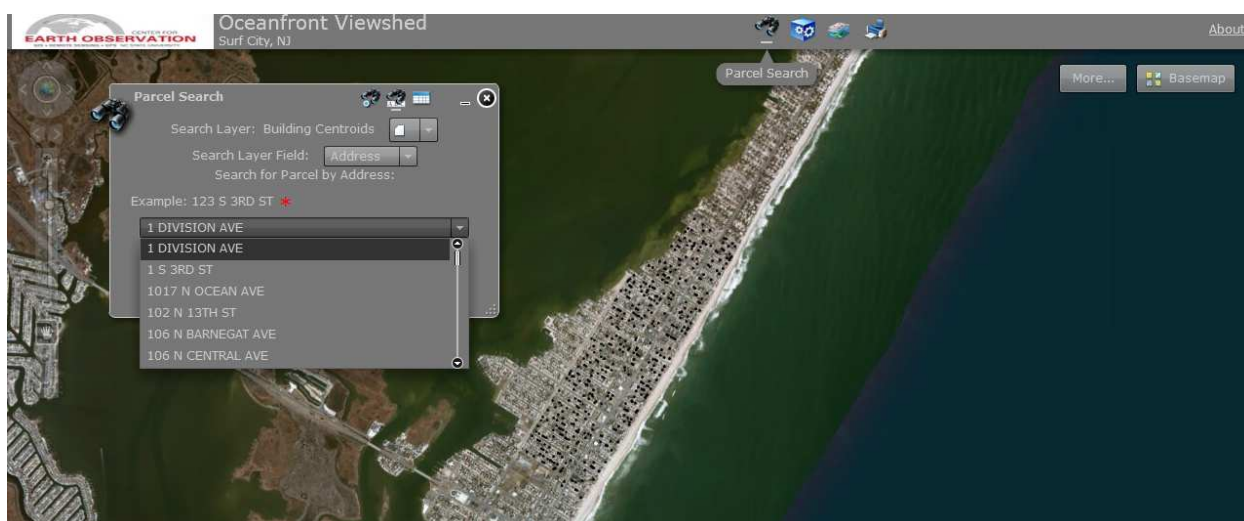


Figure 11. Web Application: Search by address tool

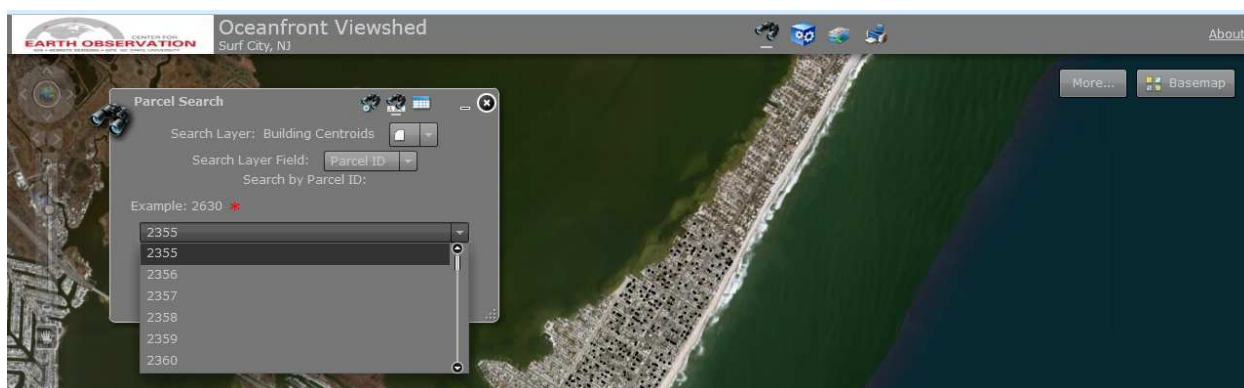


Figure 12. Web Application: Search by parcel ID

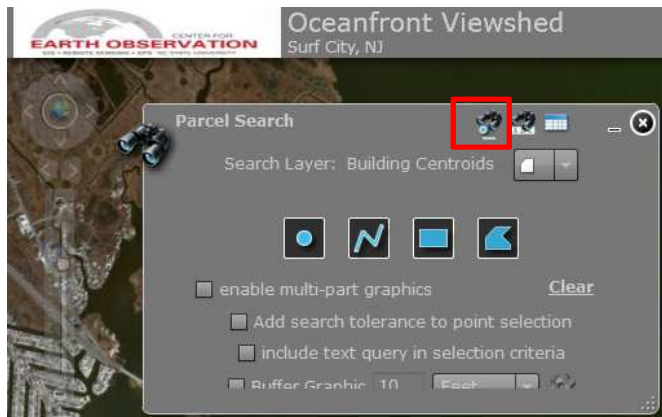


Figure 13. Web Application: Graphical Search

Documentation

A user manual as well as a web application update manual are located on the server at **C:\inetpub\wwwroot\flexviewer\Surf_City\Documentation**. Copies of these manual have also been included within this appendices of this report.

Results

This project resulted in two major outputs, a custom script tool and a web application. The following sections describes these outputs in more detail.

Script Tool Outputs

The outputs of the oceanfront viewshed script tool are saved in a user specified location. Within this location, the user will find three major outputs from each run of the tool. First, there will be a folder containing the final viewsheds for each observation point, at each floor, in vector format (Figure 14, Figure 16). Secondly, there will be a folder called “ArcLengths”. This folder contains the arc segments for each viewshed at each floor for any parcel that contained a viewshed. Filenames are structured to include the floor number and the unique ID number for each viewshed or arc length so that each is easily identifiable (Figure 15).

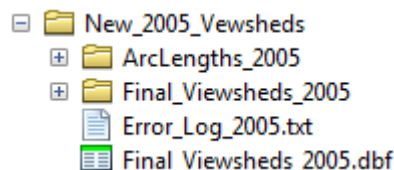


Figure 14. Oceanfront viewshed output data

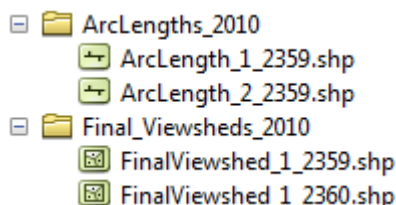


Figure 15. File naming convention

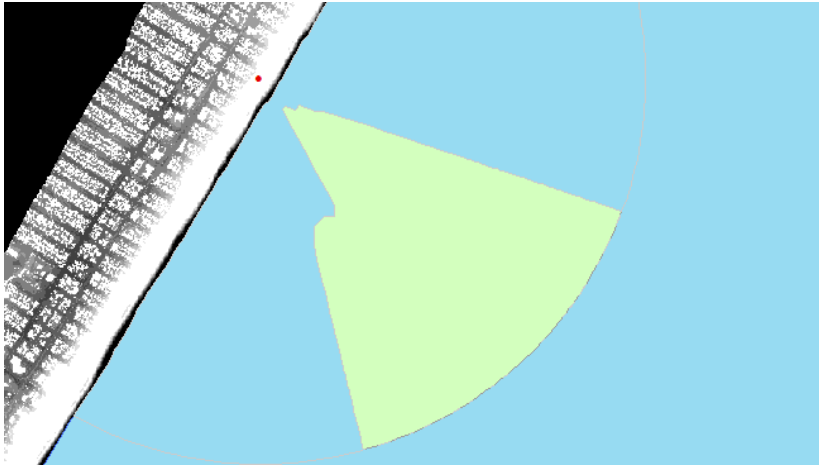


Figure 16. Oceanfront viewshed output viewshed example

There will also a .dbf table. This table contains the viewshed output in degrees, for each observation point for each floor. Below is a description of some of the fields within the output table and a description of the information the field contains (Figure 17).

GRIDCODE	=	Indicates whether there are viewable areas (1) or not (0).
Id	=	The unique ID assigned to each observation point.
SPOT	=	The average bare earth elevation of a parcel.
OFFSETA	=	The height of the observer. This value is added to the SPOT value when calculating the final viewshed.
FLOORS	=	The total number of floors in the parcel.
MAX_LENGTH	=	The viewshed arc-length used in calculating the viewshed in degrees for that parcel.
FLR_RAN	=	The floor ran for the corresponding viewshed.
VIEW_YEAR	=	The viewshed in degrees for the specified year (i.e 2005).

Final_Viewshed_2005										
	OID	GRIDCODE	Id	SPOT	OFFSETA	FLOORS	FREQUENCY	MAX_LENGTH	FLR_RAN	VIEW_2005
	1	0	2358	7.5	10	2	1	4.944024	1	0
	42	1	2358	7.5	20	2	3	1.433571	2	41.089628
	0	1	2359	14.25	10	2	6	3.092898	1	88.649942
	41	1	2359	14.25	20	2	2	3.889143	2	111.472257
	12	0	2360	8.5	10	1	1	4.944385	1	0

Figure 17. Viewshed output table results

Lastly, there is an error log in text format. If any errors occurred during the scripts processing (i.e a viewshed for a point failed, folder couldn't be deleted etc.) the errors are written to this file for reference.

Web Application Outputs

When the user performs a viewshed analysis with the web application, the results will appear showing the 2005 viewshed in red, the 2010 viewshed in blue, and unviewable areas in black (Figure 18) . The 2005 areas shown in red were lost as a result of the dune construction. A pop-up window will provide the user with information on the viewshed in degrees for both 2005 and 2010.



Figure 18. Web Application: Viewshed results

When the user performs a search by address or parcel number, the application will zoom to the parcel location and provide a gridview of the results (Figure 19). These results can be exported out to a csv file.

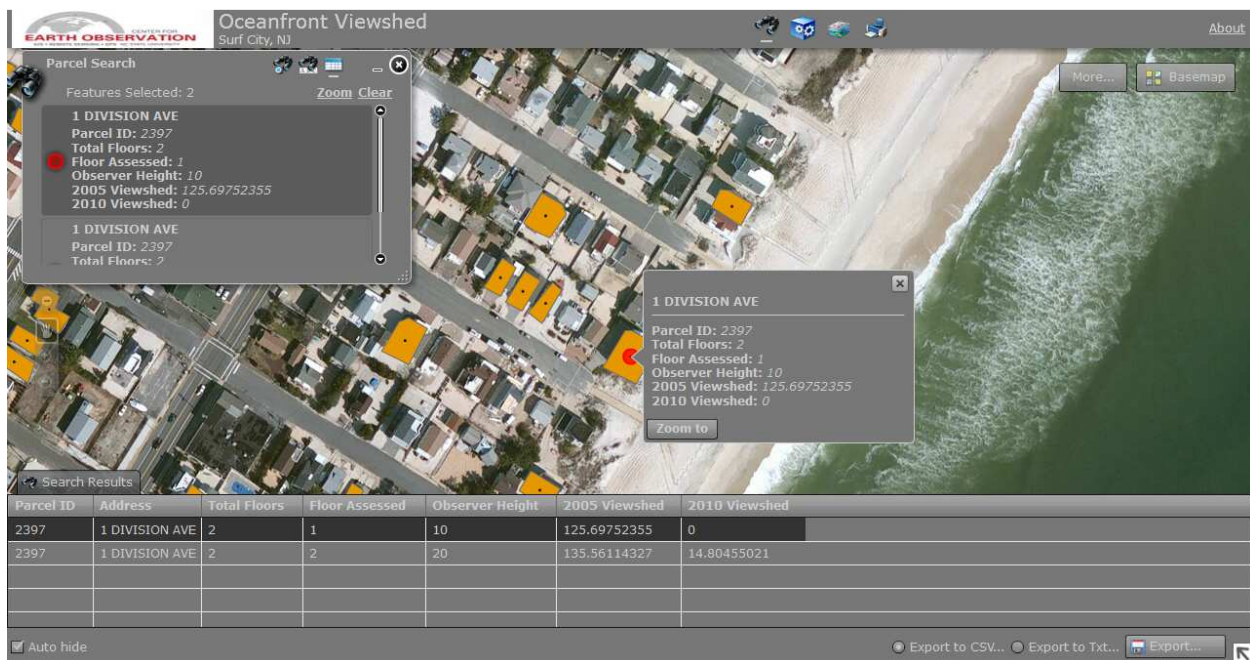


Figure 19. Web Application: Search by Address Results

Multiple parcels can be searched by using the graphical search tool (Figure 20).

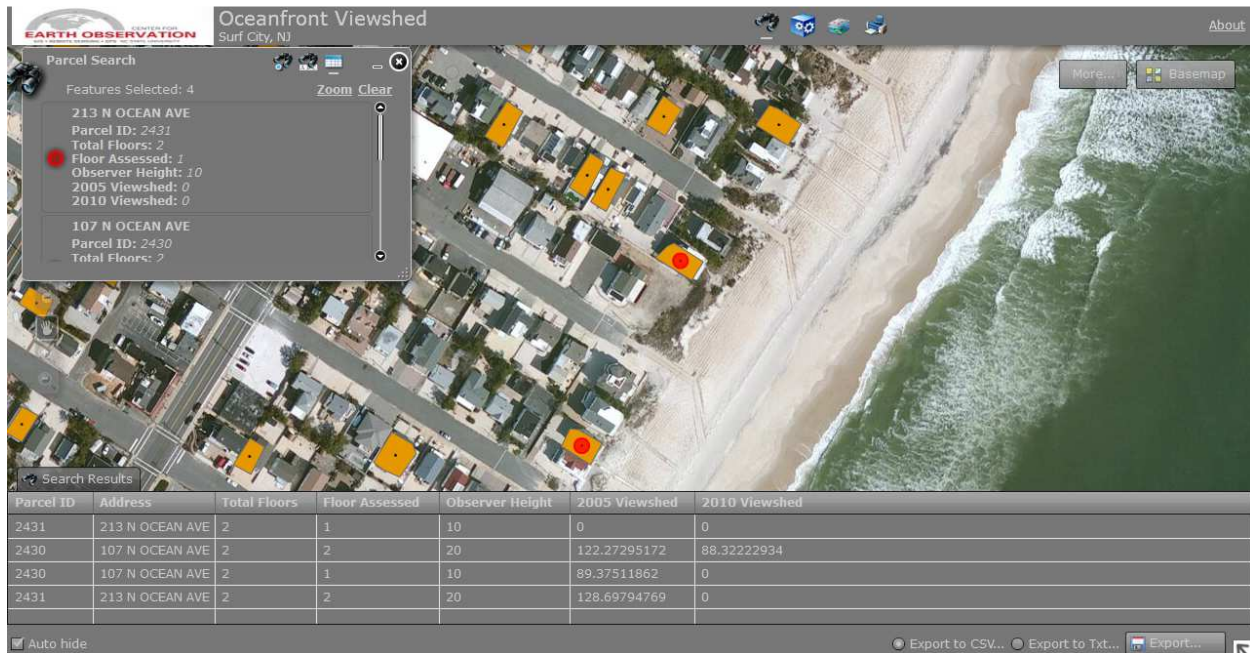


Figure 20. Web Application: Search results from geometry tool

Discussion

Script Tool

The purpose of this project was to provide the client with a custom GIS tool that would assess a parcels viewshed in 2005 and 2010, at both first and second floor height (where applicable). This was achieved by writing a custom GIS script tool using python and ArcMap 10.1. Although issues were encountered during the tool development phase, these issues were a result of the specific, yet unclearly defined needs within ArcMap. After researching forums and talking with professionals most of these issues were resolved. There are two remaining issues. First, the script is written to automatically delete all intermediate data, however, some files within the folder being deleted are locked, and deleting the intermediate files folder fails forcing the user to delete the folder manually. Secondly, it was challenging to figure out an automated method for only measuring the top length of the arc. Based on other conversations with others who had done this type of analysis in the past, the original script calculated the viewshed based on the entire perimeter of the largest segment of viewable area, however, this was creating problems in developing a correct viewshed degree measure. After many trial and error runs, a method was developed for extracting the top arc of the viewable portion of a viewshed, measuring the combined sum of these arcs, and calculating the degree measure.

Web Application

The purpose of the web application was to provide users the ability to run a comparative viewshed analysis. It was originally intended to have the application run at both first and second floor height, but I was unable to make the floor heights as a choice list. The height parameter appears to only be an open value option which means any value could be input. Since it was important to have the web application results to be consistent with the script tool results it was decided to only show a comparative viewshed at the first floor since this was most likely to be the view most affected by dune construction.

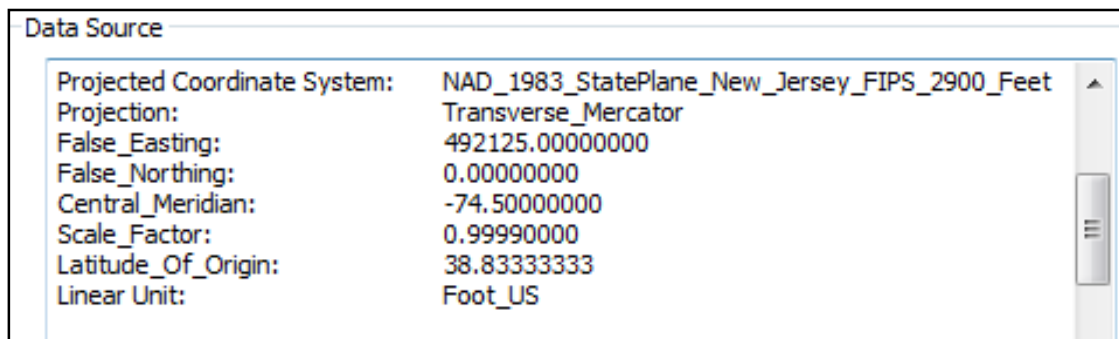
Future Research

As mentioned in the script tool discussion, some irregular viewsheds were created in the analysis processes, and developing a methodology to extract only the top of the viewshed arc was challenging. The tool also takes a very long time to run on datasets larger than 200 points (roughly 12-15 hours). Future efforts could investigate alternative methods that may reduce processing time. The visual output of this project produced a web application. Other visualization outputs such as 3D flyovers could also be produced.

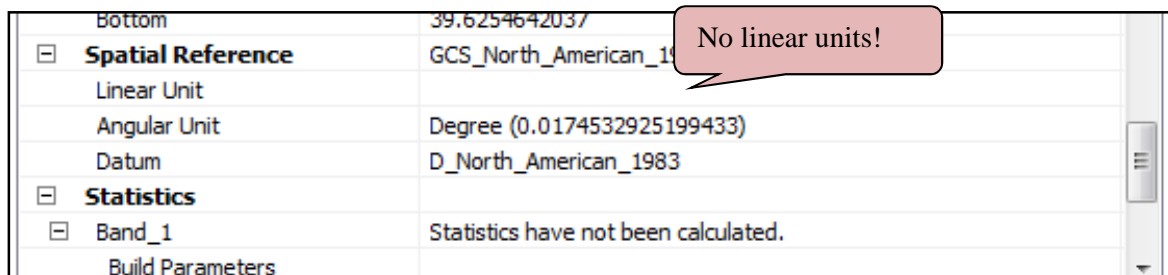
Appendix I. Preparing Data for Viewshed Analysis

The purpose of this appendix is to provide some general guidance in ensuring that the input data meets the assumptions of the Oceanfront Viewshed script tool.

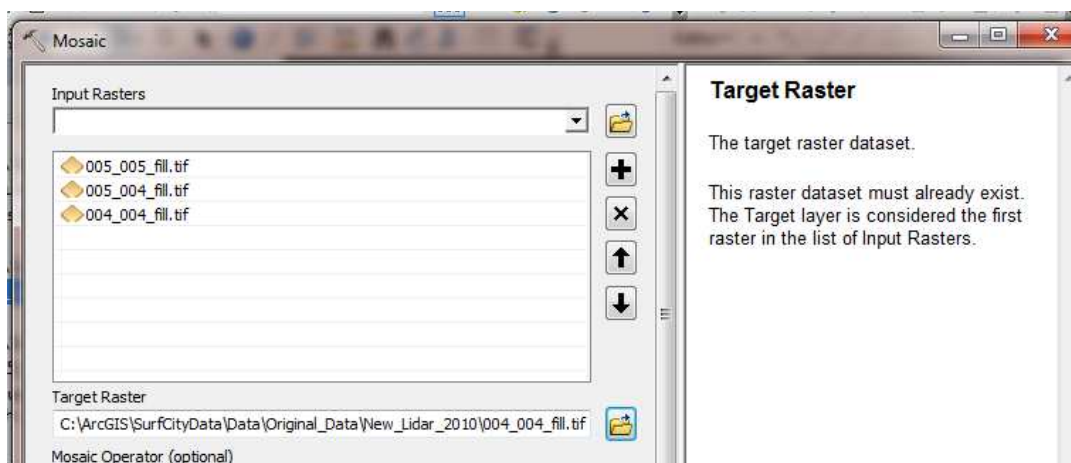
1. Make copies of all the original data. Although this is not an assumption of the tool(s), it's good data management practice.
2. The script assumes that the input observations points field names, particularly the "Id" field, is the same in all observation input files. Field names are case sensitive.
3. Make sure all data is in the same projection that contain linear units (i.e NAD 1983 StatePlane New_Jersey FIPS 2900 Feet).



If a shapefile or raster is projected in a geographic coordinate system that only has angular units, the results may be invalid, and the tool may fail completely.



4. Mosaic rasters for each analysis year into single raster dataset (Data Management>Raster>Raster Dataset). Since copies were made of all the original data, and data preparation is being based of the copied dataset, make the target raster one of the input rasters. This will add all of the rasters to that specified target raster.

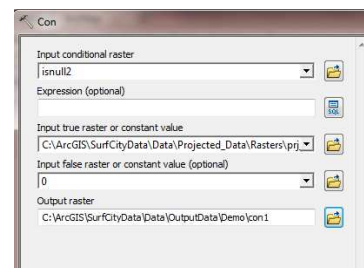
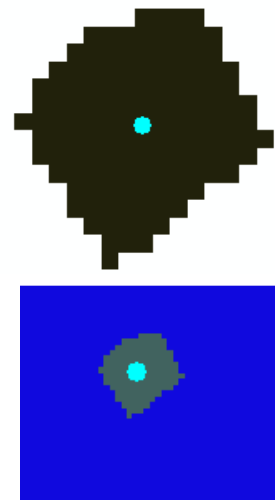


Use the Raster Fix script tool to convert the new mosaic to an integer base, grid raster. Results may not be automatically added to the display

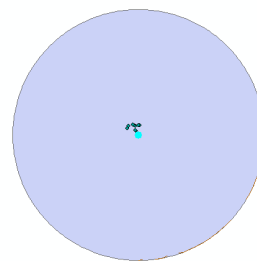
Appendix II. Calculating Viewshed Methods

For n points perform the following to get a degree view (these steps do not include raster processing):

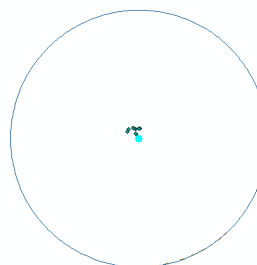
1. **Bare Earth.** Calculate the average bare earth elevation for each parcel. Select out the building footprint that intersects with the select (n) point. Use the selected footprint to select out the intersecting lidar data (Extract by Mask).
2. Use the statistics tool to get the average elevation of the selected parcel. Join the resulting statistics table to the observation point file. Use the Field calculator to transfer the average elevation to the SPOT field in the observation point file. Bare earth elevation for the parcel is set.
3. **Elevation Raster.** Create final elevation raster for the selected centroid point by changing the elevation of the temporary footprint raster to zero and merging this with the Elevation raster. This allows the selected footprint to be changed to an elevation of 0, while retaining elevation values for all other areas within the raster. This also ensures that the observer height will start at the bare earth SPOT elevation.
4. Convert the selected footprint to a raster based on Maximum Area.
5. Use the IsNull to identify Null areas.
6. Use the Con tool to replace all null areas with lidar data, and keep elevation footprint as 0. This output will be the input to the viewshed analysis. The elevation footprint of 0 is replaced with the SPOT value that was calculated in steps 1-2.



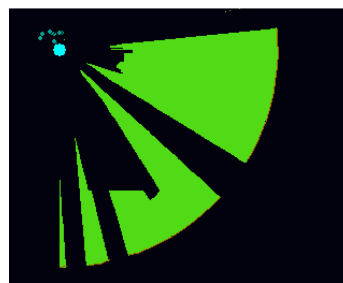
7. **1-Mile Buffer.** Buffer the selected n point with a 1 mile buffer.



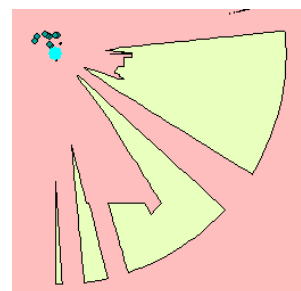
8. Convert this buffer to a polyline.



9. **Viewshed.** Perform a viewshed using the viewshed tool.



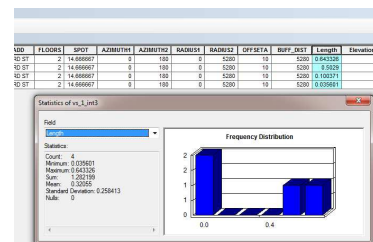
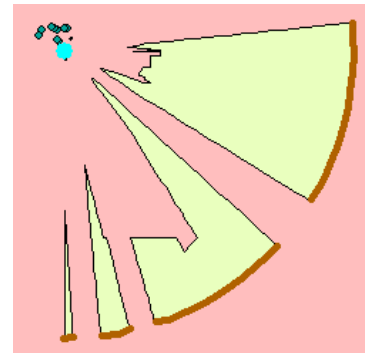
10. Convert viewshed to a polygon





(Overlaid with viewshed polygon for reference)

11. Select gridcode values of 1 in the viewshed polygon and intersect with the buffer polyline in step 2 (tolerance = 10 ft.).



12. **Degree Value.** Calculate arc length and get sum of arc lengths using the summary statistics tool. Calculate viewshed with field calculator using the formula $(\text{arclength}/3.14)*180$.
i.e. $(1.28\text{mi}/3.14\text{mi})*180 = 73.375$ degrees

Sum = 1.28 mi

Appendix III. Raster Fix Python Script

#Name: Raster_NoData.py
#Purpose: Replace "NoData" values in a raster with 0
#Assumptions: Assumes that the input raster is in a non-GRID format
#Inputs: This script takes 2 inputs
0 = Input Raster
1 = Name and location of output raster
#Outputs: The script will produce 1 output, an output raster.
#Author: Rachel Albritton
#Last Update: October 15, 2013
#-----

```
import arcpy, os
from arcpy import env
from arcpy.sa import *
```

```
# Check out the ArcGIS 3D Analyst extension license
arcpy.CheckOutExtension("3D")
arcpy.CheckOutExtension("Spatial")
arcpy.env.overwriteOutput = True
```

```
#Variables
Elevation = arcpy.GetParameterAsText(0)
OutRaster = arcpy.GetParameterAsText(1)#File name saved
```

```
#Convert .tif to GRID
GridElev = os.path.dirname(OutRaster)+"\\grid_"+Elevation.split(".")[0][:7]
arcpy.CopyRaster_management(Elevation, GridElev)
arcpy.AddMessage("\n"+arcpy.GetMessages()+"\n")
```

```
#Convert floating point to integer
outInt = Int(GridElev)
outInt.save(os.path.dirname(OutRaster)+"\\int_rast")
arcpy.AddMessage(arcpy.GetMessages()+"\n")
```

```
#Identify Null Cells
outIsNull = IsNull(outInt) #Identify NoData Areas
outIsNull.save(os.path.dirname(OutRaster)+"\\rast_null")
outCon = Con(outIsNull,0,outInt) #Use Con tool to change building footprint to elevation of 0 while
leaving all other building footprints as is
outCon.save(os.path.dirname(OutRaster)+"\\rast_con")
arcpy.AddMessage(arcpy.GetMessages()+"\n")
```

```
#It is assumed that negative elevation values correspond to the oceanfloor and not the surface of the ocean
#These values can be changed to 0 using the Con tool
outCon2 = Con(outCon,outCon,0, "VALUE > 1")
outCon2.save(OutRaster)
arcpy.AddMessage(arcpy.GetMessages()+"\n")
```

```
#Delete intermediate data
arcpy.Delete_management(GridElev)
arcpy.Delete_management(outInt)
arcpy.Delete_management(outIsNull)
arcpy.Delete_management(outCon)

arcpy.AddMessage("Raster NoData tool complete")
```

Appendix IV. Oceanfront Viewshed Python Script

```
##-----
#Name:      OceanFront_Viewshed.py
#Purpose:    Calculates viewshed, in degrees, for all points within a point file
#Author:     Rachel Albritton
# Last Updated: October 19, 2013

#Inputs:     This script takes NINE inputs. Each are described below
              #0 = Workspace. This is typically where the folder data is located
              #1 = Output workspace, the folder where you want the results saved.
              #2 = Observation points
              #3 = Building footprints
              #4 = Elevation
              #5 = BareElevation = Elevation that parcel sits at
              #6 = Ocean
              #7 = Floor Field - the feild located within the observation point file
                  #that contains the number f floors for each record
              #8 = Year the anaylsis is for

#Assumptions: This tool assumes that all data inputs have been projected into a
              #coordinate system that has linear units. Angular units will provide
              #incorrect results, and may cause the script to fail completely.

              #This script also assumes that null values that may be present
              #within the elevation data have been converted to a value of 0.
              #The Raster_Fix.py script located within this tool box will
              #complete this task.

##-----

import arcpy, os, shutil, datetime
from arcpy import env
from arcpy.sa import*

#set workspaces
arcpy.env.workspace = "C:\\ArcGIS\\SurfCityData\\Data" #arcpy.GetParameterAsText(0)
outputWorkspace = "C:\\ArcGIS\\SurfCityData\\Data\\Test" #arcpy.GetParameterAsText(1)
arcpy.env.overwriteOutput = True

#Check out the ArcGIS 3D Analyst extension license
arcpy.CheckOutExtension("3D")
arcpy.CheckOutExtension("Spatial")

arcpy.SetProgressor("default","Conducting Viewshed Analysis")

#Variables
ObsPts = "C:\\ArcGIS\\SurfCityData\\Data\\Projected_Data\\TestPts3.shp"
#arcpy.GetParameterAsText(2)
footprint = "C:\\ArcGIS\\SurfCityData\\Data\\Projected_Data\\Infrastructure\\Building_Footprints.shp"
#arcpy.GetParameterAsText(3)
```

```

Elevation = "C:\\ArcGIS\\SurfCityData\\Data\\Projected_Data\\Rasters\\prj_2010"
#arcpy.GetParameterAsText(4)
BareElevation = "C:\\ArcGIS\\SurfCityData\\Data\\Projected_Data\\Rasters\\baregrnd"
#arcpy.GetParameterAsText(5)
Ocean = "C:\\ArcGIS\\SurfCityData\\Data\\Projected_Data\\Waterbodies\\AtlanticOcean.shp"
#arcpy.GetParameterAsText(6)
FloorField = "FLOORS" #arcpy.GetParameterAsText(7)
Year = "2010" #arcpy.GetParameterAsText(8)

#Set analysis extent to elevation raster
arcpy.env.extent = Elevation
arcpy.env.cellSize = Elevation

#Create a temp folder to hold intermediate files.
#Some of these folders will be deleted after the analysis has runs successfully.
IntermediateFiles = outputWorkspace+"\\IntermediateFiles_"+Year
Final_Floor_Viewsheds = outputWorkspace+"\\Final_Viewsheds_"+Year
SummaryTables = outputWorkspace+"\\Summary_Tables_"+Year
ElevAvgTables=outputWorkspace+"\\ElevAvg_Tables_"+Year
ArcLengths = outputWorkspace+"\\ArcLengths_"+Year

if not os.path.exists(IntermediateFiles): os.makedirs(IntermediateFiles)
if not os.path.exists(Final_Floor_Viewsheds) : os.makedirs(Final_Floor_Viewsheds)
if not os.path.exists(SummaryTables) : os.makedirs(SummaryTables)
if not os.path.exists(ElevAvgTables): os.makedirs(ElevAvgTables)
if not os.path.exists(ArcLengths): os.makedirs(ArcLengths)

def outName(input,post="_Output"):
    """Returns output name."""
    outName=os.path.basename(input).split(".")[0]+post
    return outName

def DegViewshed (FLOOR, HEIGHT):
    """Calculates a parcels viewshed, in degrees"""

    #Select Record
    arcpy.SelectLayerByAttribute_management(PointsFL,"NEW_SELECTION",SQL)

    #Set Observer Height (OffSETA)
    arcpy.CalculateField_management(PointsFL,"OFFSETA",HEIGHT,"PYTHON_9.3")

    #perform viewshed analysis
    arcpy.SetProgressorLabel("Performing Viewshed Analysis for point "+str(value))
    outViewshed = IntermediateFiles+"\\vs_"+str(FLOOR)+"_"+str(value).split(".")[0]
    arcpy.Viewshed_3d(outCon,PointsFL,outViewshed)

    #convert viewshed to polygon
    arcpy.SetProgressorLabel("Converting viewshed"+str(value)+" on floor "+str(FLOOR)+" to polygon.")
    OutPoly = IntermediateFiles+"\\"+os.path.basename(outViewshed).split(".")[0]+"_poly.shp"
    arcpy.RasterToPolygon_conversion(outViewshed,OutPoly)

```

```

#Intersect viewshed polygon with buffer clip
#This will allow the viewshed poly to inherit attribute fields needed for later analysis
FinalView = Final_Floor_Viewsheds+"\\FinalViewshed_"+str(FLOOR)+"_"+str(value)+".shp"
arcpy.Intersect_analysis([BufferClip,OutPoly],FinalView)

#Select features in viewshed polygon with Gridcode = 1
#If no records with grid = 1 exist, script will skip to setting viewshed in degrees to 0

#Convert viewshed polygon to layer
ViewshedLayer = outName(FinalView,"lyr")
arcpy.MakeFeatureLayer_management(FinalView,ViewshedLayer)

#Select records with gridcode = 1
arcpy.SelectLayerByAttribute_management(ViewshedLayer,"NEW_SELECTION","GRIDCODE
="+str(1)+"")

#Get count of the # of records selected in viewshed poly layer
VsLyrCount = int(arcpy.GetCount_management(ViewshedLayer).getOutput(0))

NoView = SummaryTables+"\\summary_"+str(FLOOR)+"_"+str(value)+".dbf"
YesView = SummaryTables+"\\summary_"+str(FLOOR)+"_"+str(value)+".dbf"
StatsField0 = [{"GRIDCODE","SUM"}]
CaseField0 = ["ID","SPOT",FloorField]
StatsField1 = [{"LENGTH","SUM"}]
CaseField1 = ["GRIDCODE","ID","SPOT",FloorField]
VsArcLengths = ArcLengths+"\\ArcLength_"+str(FLOOR)+"_"+str(value)+".shp"

if VsLyrCount == 0: #no viewable areas exist
    arcpy.SelectLayerByAttribute_management(ViewshedLayer,"CLEAR_SELECTION")
    arcpy.SetProgressorLabel("Calculating viewshed statistics for parcel "+str(value))
    arcpy.Statistics_analysis(ViewshedLayer,NoView, StatsField0,CaseField0)

    #Add field to summary table to hold viewshed value of 0
    #Add field to note which floor viewshed corresponds to
    arcpy.AddField_management(NoView, "FLR_RAN","SHORT")
    arcpy.AddField_management(NoView, "VIEW_"+Year,"DOUBLE")
    arcpy.AddField_management(NoView,"OFFSETA","SHORT")
    arcpy.CalculateField_management(NoView,"FLR_RAN",1)
    arcpy.CalculateField_management(NoView,"VIEW_"+Year,0)
    arcpy.CalculateField_management(NoView,"OFFSETA",HEIGHT)

else: #Calculate viewshed, in degrees, for selected records
    arcpy.SetProgressorLabel("Getting arc length for parcel "+str(value)+" at the "+str(FLOOR)+"
floor.")
    arcpy.Intersect_analysis([BufferLine,ViewshedLayer],VsArcLengths,"",10,"LINE")
    arcpy.AddField_management(VsArcLengths, "Length","DOUBLE")

arcpy.CalculateField_management(VsArcLengths,"Length","!SHAPE.length@miles!","PYTHON_9.3")
arcpy.Statistics_analysis(VsArcLengths,YesView,StatsField1,CaseField1)

#Add fields to output summary table

```

```

arcpy.AddField_management(YesView,"FLR_RAN","SHORT")
arcpy.AddField_management(YesView,"VIEW_"+Year,"DOUBLE")
arcpy.AddField_management(YesView,"OFFSETA","SHORT")
arcpy.CalculateField_management(YesView,"FLR_RAN",FLOOR)
arcpy.CalculateField_management(YesView,"OFFSETA",HEIGHT)

arcpy.CalculateField_management(YesView,"VIEW_"+Year,"(!SUM_LENGTH!/3.14)*180)","PYTHON_9.3")
    arcpy.SelectLayerByAttribute_management(ViewshedLayer,"CLEAR_SELECTION")

#Open error log file
infile = open(outputWorkspace+"\\Error_Log_"+Year+".txt","w")

#Perform field check for viewshed parameters within the observation point attribute table.
#Script will add field to the attribute table if the field does not already exist.
#Needed fields are SPOT - used to define the surface elevations for the observation points.
#Azimuth -define the horizontal angle limits to the scan (start and end points in degrees).
#Radius - defines the search distance when identifying areas visible from each observation point.
#Cells that are beyond a certain distance can be excluded from the analysis.

VSFieldList = ["SPOT","OFFSETA","AZIMUTH1","AZIMUTH2","RADIUS1","RADIUS2"]
arcpy.SetProgressorLabel("Checking fields in observation point attribute table")

for FieldList in VSFieldList:
    ObsPtsFieldList=arcpy.ListFields(ObsPts,FieldList)
    fieldNames=[field.name for field in ObsPtsFieldList]

    if FieldList in fieldNames:
        print "Field", FieldList, "found in", ObsPts
    else:
        print"Field", FieldList, "NOT found in", ObsPts
        arcpy.AddField_management(ObsPts,FieldList, "DOUBLE")
        print FieldList, "created"

#Populate viewshed parameters with correct values for viewshed
Az1Cal = 0
Az2Cal = 180
Rad1Cal = 0
Rad2Cal = 5280

arcpy.CalculateField_management(ObsPts,"AZIMUTH1",Az1Cal)
arcpy.CalculateField_management(ObsPts,"AZIMUTH2",Az2Cal)
arcpy.CalculateField_management(ObsPts,"RADIUS1",Rad1Cal)
arcpy.CalculateField_management(ObsPts,"RADIUS2",Rad2Cal)

#Create Feature Layers
arcpy.SetProgressorLabel("Creating feature layers")
PointsFL = outName(ObsPts,"_Lyr")
footprintFL = outName(footprint,"_Lyr")
arcpy.MakeFeatureLayer_management(ObsPts, PointsFL)

```

```

arcpy.MakeFeatureLayer_management(footprint,footprintFL)

#Select observation points one by one
arcpy.SetProgressorLabel("Starting viewshed analysis...")
RangeCount = int(arcpy.GetCount_management(PointsFL).getOutput(0))

#Count number of parcels being processed
arcpy.AddMessage("Calculating viewshed for "+str(RangeCount)+" parcels")

sc = arcpy.SearchCursor(PointsFL)

for row in sc:

    #Get Parcel ID value
    value = row.ID
    count = row.FID+1
    FlrCnt = row.getValue(FloorField)

    #Get bare earth elevation of parcel
    arcpy.SetProgressorLabel("Changing elevation footprint to bare earth elevation for point "+str(value))
    SQL = "Id =" +str(value)+"
    arcpy.SelectLayerByAttribute_management(PointsFL,"NEW_SELECTION",SQL)
    arcpy.SelectLayerByLocation_management(footprintFL,"INTERSECT",PointsFL)

    arcpy.env.workspace = IntermediateFiles #need to change workspace so that the .save files get saved
correctly
    outExtractByMask = ExtractByMask(BareElevation,footprintFL)
    outExtractByMask.save(IntermediateFiles+"\\ebm_"+str(value))

    ElevAvg = ElevAvgTables+"\\avgelev_"+str(value)+".dbf"
    arcpy.Statistics_analysis(outExtractByMask,ElevAvg,[["VALUE","MEAN"]])

    arcpy.AddField_management(ElevAvg,"Pt_ID","SHORT")
    arcpy.CalculateField_management(ElevAvg,"Pt_ID",value)
    arcpy.AddJoin_management(PointsFL,"Id",ElevAvg,"Pt_ID","KEEP_COMMON")

    Field1 = os.path.basename(ObsPts).split(".")[0]+".SPOT"
    Field2 = "!" +os.path.basename(ElevAvg).split(".")[0]+".MEAN_VALUE!"
    arcpy.CalculateField_management(PointsFL,Field1,Field2,"PYTHON_9.3")
    arcpy.RemoveJoin_management(PointsFL)

    #Set parcel elevation to 0 this will be replaced by SPOT value calculated above
    RastFootprint = IntermediateFiles+"\\fp_"+str(value).split(".")[0]
    arcpy.PolygonToRaster_conversion(footprintFL,"FID",RastFootprint,"MAXIMUM_AREA","",6)
    outIsNull = IsNull(RastFootprint) #Identify NoData Areas
    outIsNull.save(IntermediateFiles+"\\null1_"+str(value))
    outCon = Con(outIsNull,Elevation,0,"") #Use Con tool to change building footprint to elevation of 0
while leaving all other building footprints as is
    outCon.save(IntermediateFiles+"\\con1_"+str(value)) #Final raster to be used in viewshed analysis

```

```

#buffer selected viewpoint
arcpy.SetProgressorLabel("Buffering point "+str(value))
outBuffer = IntermediateFiles+"\\buffer_"+str(value)+".shp"
arcpy.Buffer_analysis(PointsFL,outBuffer,"1 mile")

#Convert buffer polygon to line
BufferLine = IntermediateFiles+"\\BufferLine_"+str(value)+".shp"
arcpy.FeatureToLine_management(outBuffer,BufferLine)

#Clip buffer to Ocean
arcpy.SetProgressorLabel("Clipping point "+str(value)+" buffer to ocean")
BufferClip = IntermediateFiles+"\\buffer_clipped_"+str(value).split(".")[0]+".shp"
arcpy.Clip_analysis(outBuffer, Ocean, BufferClip)

if FlrCnt ==1: #parcel floor count =1
    arcpy.AddMessage("Parcel "+str(value)+" has 1 story to process. Calculating viewshed now...")
    print "Parcel ",str(value)," has 1 story to process. Calculating viewshed now..."

    DegViewshed(1,10) #Calculate the viewshed with an observer height of 10 feet then move to point

    arcpy.AddMessage("First floor viewshed for parcel "+str(value)+" has been completed...")
    print "First floor viewshed for parcel ",str(value)," has been completed..."
    arcpy.AddMessage(str(count)+" of "+str(RangeCount)+" parcels has been completed.\n")
    print str(count)," of "+str(RangeCount)," parcels has been processed.\n"

else: #if parcel has 1.5 floors or greater do this
    arcpy.AddMessage("Parcel "+str(value)+" has 2 stories to process. Calculating viewsheds now...")
    print "Parcel ",str(value)," has 2 stories to process. Calculating viewsheds now..."
    DegViewshed(1,10) #Calculate first floor view, then
    arcpy.AddMessage("First floor viewshed for parcel "+str(value)+" has been completed...")
    print "First floor viewshed for parcel ",str(value)," has been completed..."

    DegViewshed(2,20) #Calculate second floor view
    arcpy.AddMessage("Second floor viewshed for parcel "+str(value)+" has been completed...")
    print "Second floor viewshed for parcel ",str(value)," has been completed..."
    arcpy.AddMessage("Viewsheds for "+str(count)+" of "+str(RangeCount)+" parcels have been
processed.\n")
    print "Viewsheds for",str(count)," of ",str(RangeCount)," parcels have been processed.\n"
    del row
del sc

#Merge all summary tables into a single table
arcpy.SetProgressorLabel("Creating final viewshed table")

arcpy.env.workspace = SummaryTables
FinalTable = outputWorkspace+"\\Final_Viewsheds_"+Year+".dbf"
Tables = arcpy.ListTables()
arcpy.Merge_management(Tables,FinalTable)

#Delete unneeded fields from final table

```

```
arcpy.DeleteField_management(FinalTable,["FREQUENCY","SUM_GRIDCO"])

##import win32com.client

print "Final viewshed table for",Year,"is located in",outputWorkspace
arcpy.AddMessage("Final viewshed table for "+Year+" is located in "+outputWorkspace)

#save copy of table to CSV format

try:

    excel = win32com.client.Dispatch('Excel.Application')

    inDBF = FinalTable
    outCSV = FinalTable.split(".")[0]+".csv"

    workbook = excel.Workbooks.Open(inDBF)
    # 24 represents xlCSVMSDOS
    workbook.SaveAs(outCSV,FileFormat=24)
    workbook.Close(SaveChanges=0)
    excel.Quit()

except:
    arcpy.AddMessage("\nERROR: Could not convert final viewshed table to csv file\n")
    arcpy.AddMessage(arcpy.GetMessages())
    infile.write("Could not convert final viewshed table to csv file\n")
    infile.write(arcpy.GetMessages()+"\n\n")

#Delete individual summary tables
arcpy.SetProgressorLabel("Deleting intermediate files ")
try:
    arcpy.env.workspace = SummaryTables
    for tables in arcpy.ListTables():
        arcpy.Delete_management(tables)
    shutil.rmtree(SummaryTables)
    print SummaryTables, "folder successfully deleted from", outputWorkspace,"\n"
    arcpy.AddMessage(os.path.basename(SummaryTables) + " folder successfully deleted from "+
outputWorkspace+"\n")
except:
    print "\nERROR: Could not delete intermediate files from",outputWorkspace,"\nYou will need to
delete these manually\n"
    arcpy.AddMessage("\nERROR: Could not delete individual tables in "+outputWorkspace+"\nYou will
need to delete these manually\n")
    infile.write("Could not delete individual tables in "+outputWorkspace+"\nYou will need to delete these
manually\n\n")

#Delete Elevation Stats tables
try:
    shutil.rmtree(ElevAvgTables)
    print "Average Elevation tables successfully deleted from",outputWorkspace
    arcpy.AddMessage("Average Elevation tables successfully deleted from "+outputWorkspace)
```

```
except:
    print "\nERROR: Could not delete",ElevAvgTables,"folder.\nYou will need to delete this manually\n"
    arcpy.AddMessage("\nERROR: Could not delete "+ElevAvgTables+" folder.\nYou will need to delete
this manually\n")
    infile.write("Could not delete "+ElevAvgTables+" folder.\nYou will need to delete this manually\n\n")

#Delete intermediate files

try:
    arcpy.env.workspace = IntermediateFiles
    for files in arcpy.ListFeatureClasses():
        arcpy.Delete_management(files)
    shutil.rmtree(IntermediateFiles)
    print IntermediateFiles, "folder successfully deleted from", outputWorkspace,"\n"
    arcpy.AddMessage(IntermediateFiles+ " folder successfully deleted from "+outputWorkspace+"\n")

except:
    print "\nERROR: Could not delete all intermediate files in",IntermediateFiles,"\nYou will need to
delete these manually\n"
    arcpy.AddMessage("\nERROR: Could not delete intermediate files in "+IntermediateFiles+"\nYou will
need to delete these manually\n")

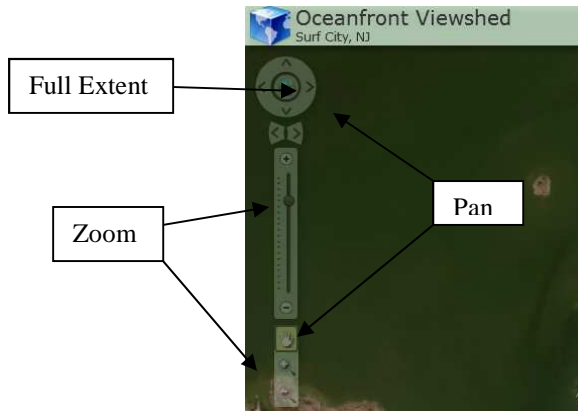
infile.close()
print "\nScript Complete\n"
arcpy.AddMessage("\nScript Complete")
```

Appendix V. Web Application User Manual

The purpose of the web application is to provide users with tabular results of a parcels viewshed, as well as provide a visual output of where the viewshed for that parcel exists at the first floor level.

Navigation

The navigation panel is located on the left had side of the screen.



Task Bar

The task bar is located along the top of the application, and contains four main tasks.



Search task



Draw Viewshed



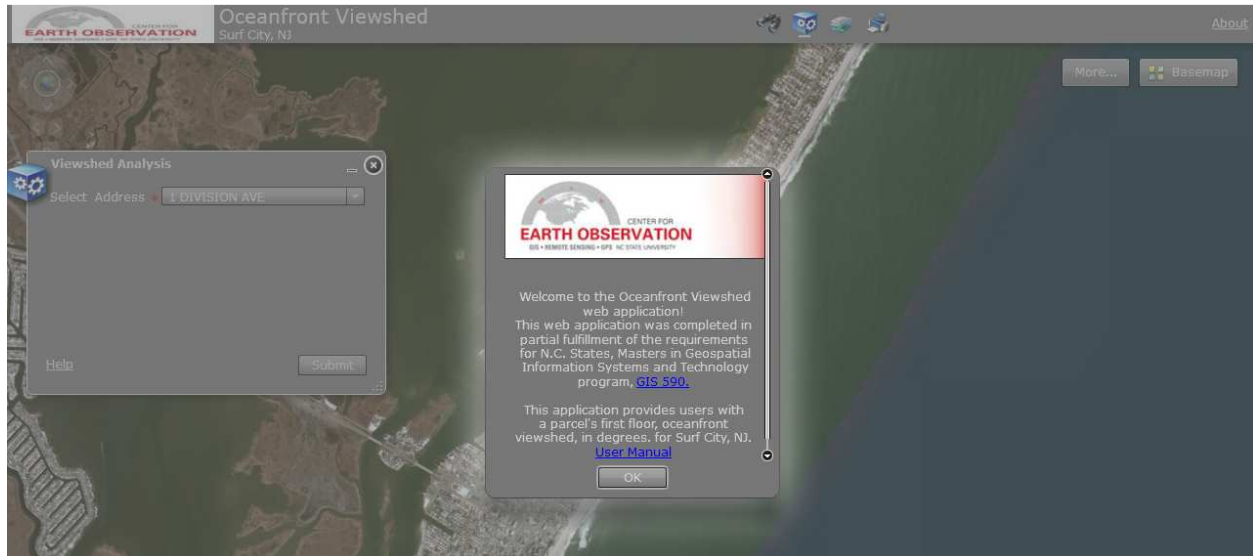
Map Legend



Print Map

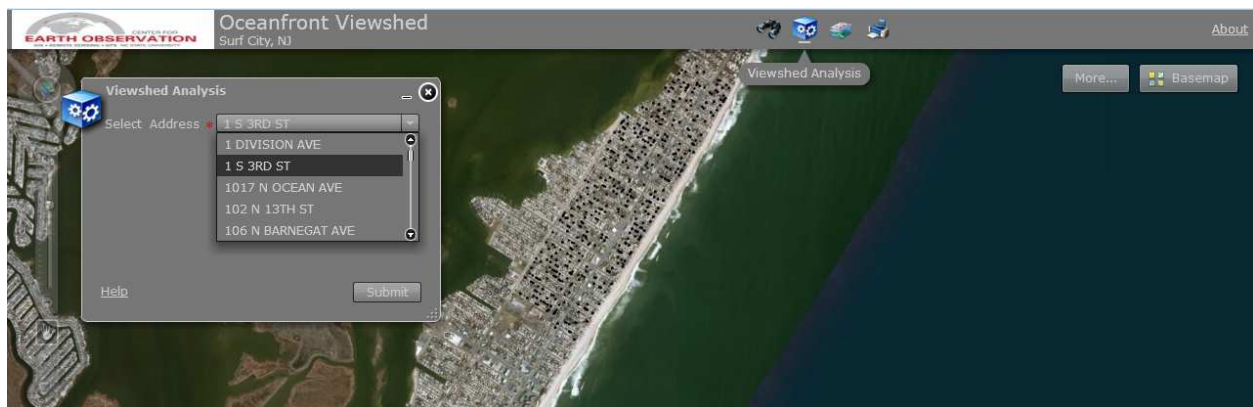
Welcome Screen

When the web application opens, the user is presented with a dialogue box that provides a brief summary of the web applications purpose, and a link to a user manual.



Draw Viewshed

Once the user clicks OK on the welcome screen, there will be a dialogue box open to run a viewshed analysis. If this dialogue box I not open, click the draw viewshed icon located on the task bar (🗺️). The user can choose an address from the drop-down menu and then click OK. Addresses are in alpha, numerical order. The viewshed takes approximately 60-90 seconds to run.



The results will appear showing the 2005 viewshed in red, the 2010 viewshed in blue, and unviewable areas in black . The 2005 areas shown in red were lost as a result of the dune construction. A pop-up window will provide the user with information on the viewshed in degrees for both 2005 and 2010.

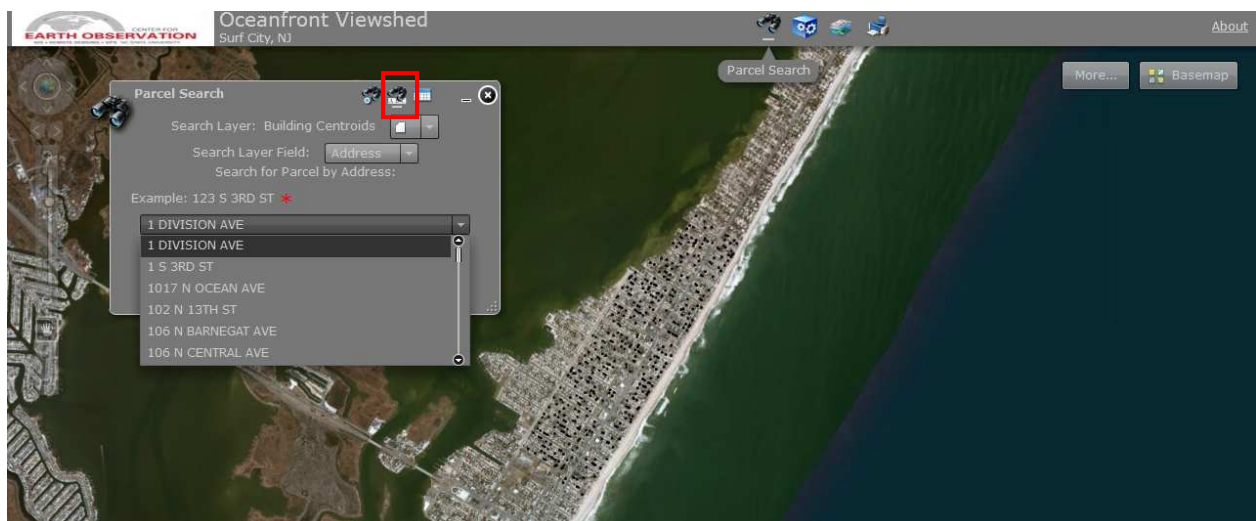


Search Task

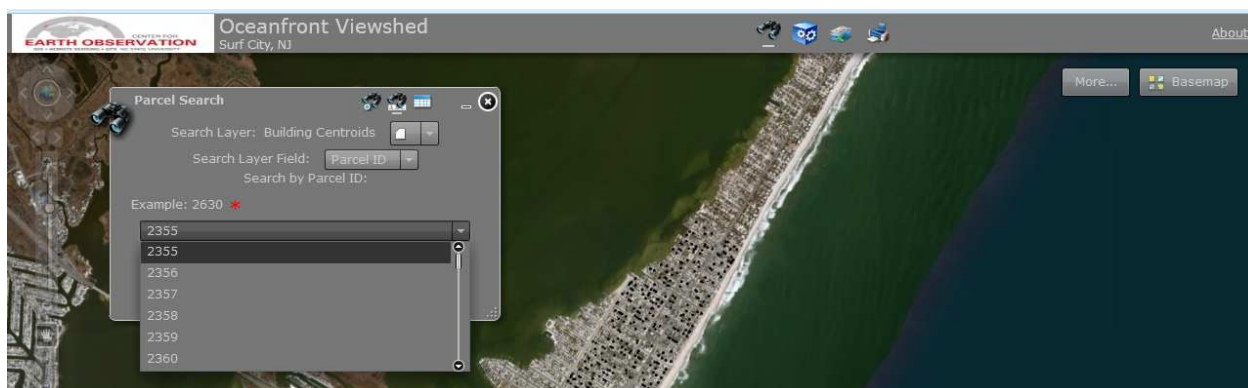
The user could also choose to search for a parcel using the search tool. This search tool allows users to search for a parcel by address, by parcel id, or by a graphical search. To search for a parcel by address, open up the search task by clicking



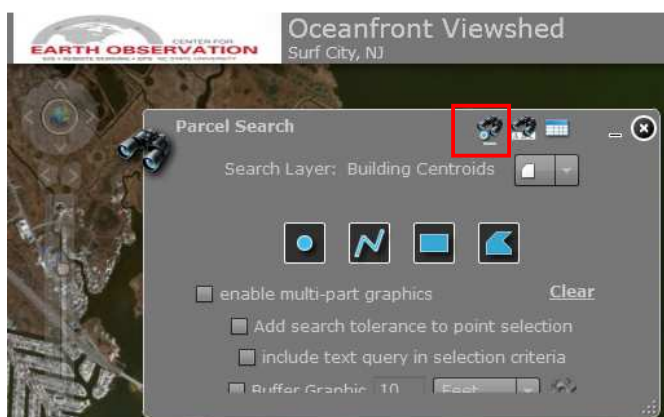
The dialogue box will open and select the search by text option at the top.



From the search by text option, a parcel can be search by address or by parcel ID. The layer which is search is controlled by the “Search Layer Field” drop down menu.

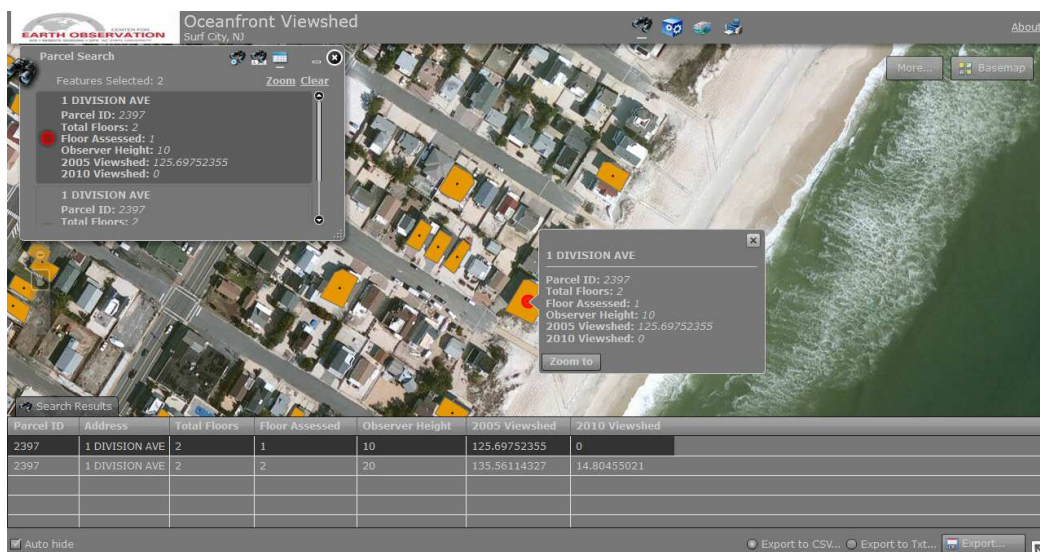


A graphical search can also be used to identify more than one parcel. In the parcel search task box, click on the graphical search icon at the top.



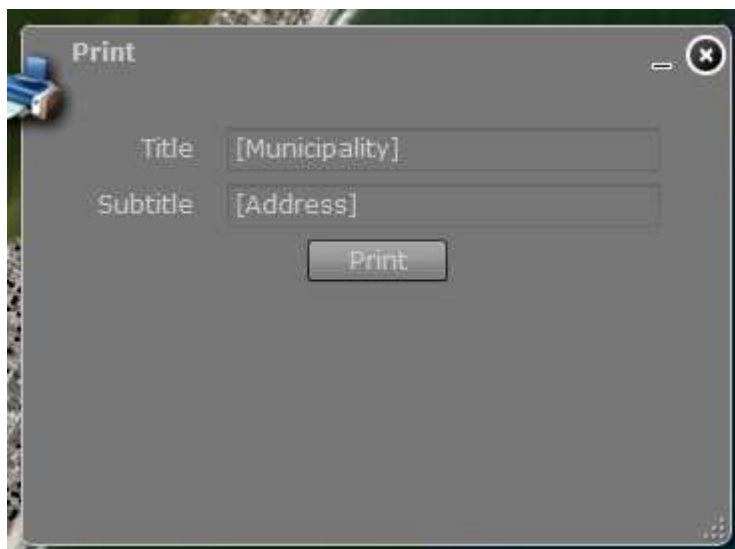
Select a geometry tool and draw an area of interest on the map that contains parcels.

For all search options, the results will appear in a table grid at the bottom of the screen. This table can be exported out to a CSV file or can be printed.



Print

The print function allows users to print the results of a viewshed.



The output of the print function will provide the user with a printed version of the current screenshot.



Appendix VI. Web Application Updates

Introduction

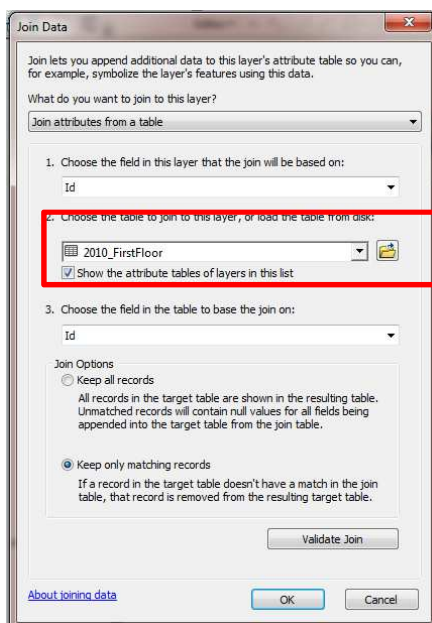
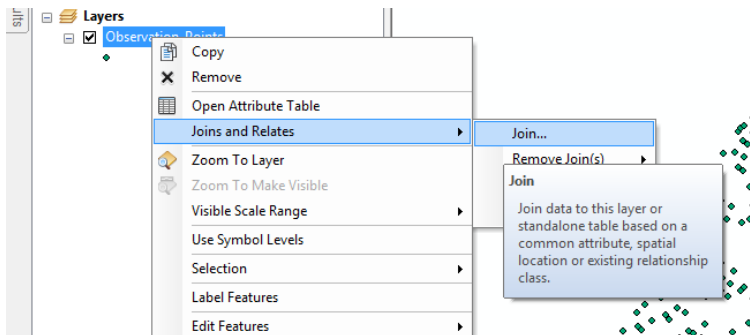
This appendix provides instructions on how to update the current application to expand the applications focus to include other municipalities within the Long Beach Island region.

Step 1: Update Model Input Layers

The observation point feature class, 2005 raster, and 2010 raster files will need to be updated. The following section outlines steps in updating this feature class to hold viewshed degree measure for each parcels first floor height in both 2005 and 2010.

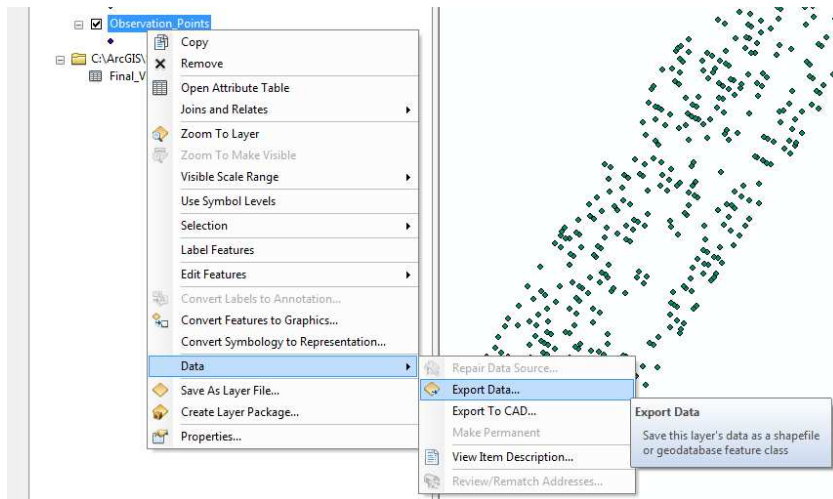
The first step in updating the web application will be to prepare the new observation point feature class to hold the viewshed data.

1. Add the observation point file and the output table for 2005 to ArcMap. Right click on the observation point file and join the output table to the observation point feature class by the “Id” field. Choose to “keep common” files. Click Ok. If a popup window comes up asking if you want field to be indexed, click yes.



Make sure the new exported table is selected

2. Once the .dbf table is joined to the observation point feature class, export the observation point file to a new feature class. This file will be temporary and will hold information for first floor viewsheds.



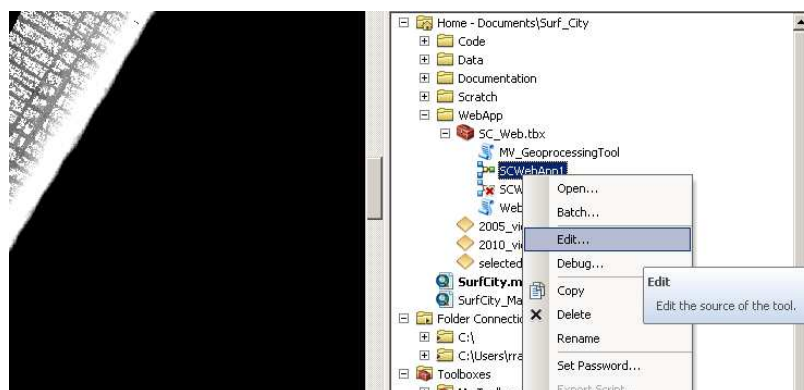
3. Cleanup the attribute field of the new observation point feature class by using the delete field tool (Data Management>Field>Delete Field). Delete all duplicated fields.
4. Repeat steps 1-3 for 2010. This should result in a new point file containing viewshed measures at each parcels first floor level in both 2005 and 2010.
5. Merge this new observation point file with the existing Observation Point file (ObsPts_Web_GP)
Location: Database Connections\surf_city.sde\surfcity.scuser.Infrastructure\surfcity.scuser.ObsPts_Web_GP

Update Raster Layers

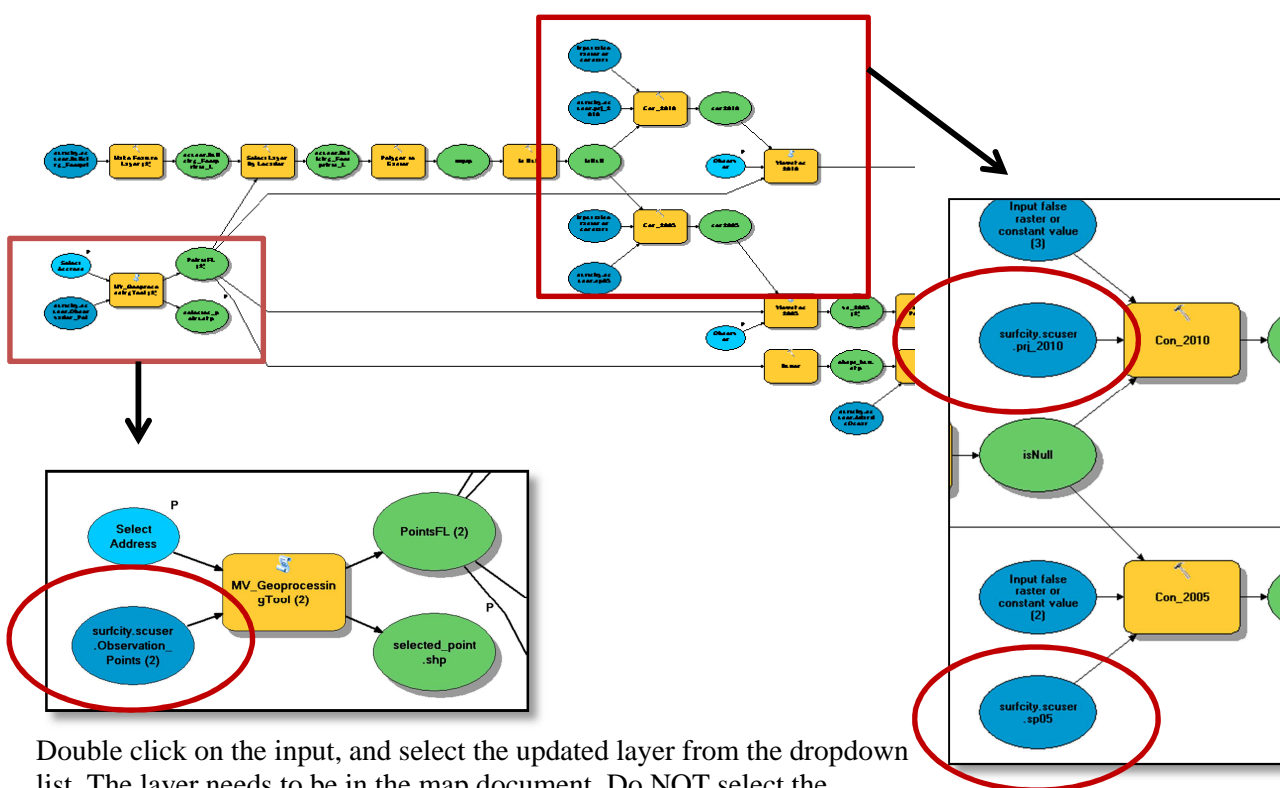
The 2005 and 2010 raster layers will also need to be updated to include the expanded study area. This can be completed by following the steps outlined in Appendix III for preparing raster layers. Import the updated raster's to the geodatabase.

Step 2: Update Web Application Model

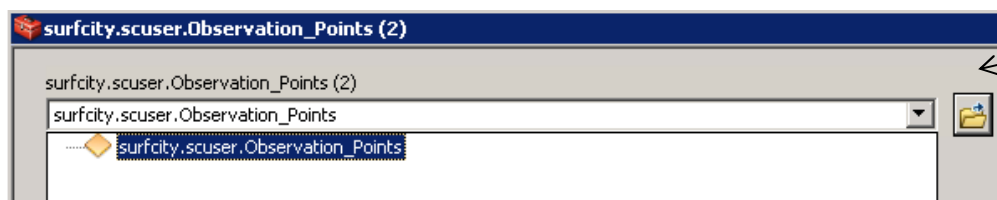
1. Add the updated raster layers and observation point layer to the map.
2. From the ArcCatalog window within ArcMap, navigate to the Surf City web application project folder (C:\Users\rralbrit\Documents\Surf_City\WebApp1).
3. Right click on the SCWebApp model and choose edit.

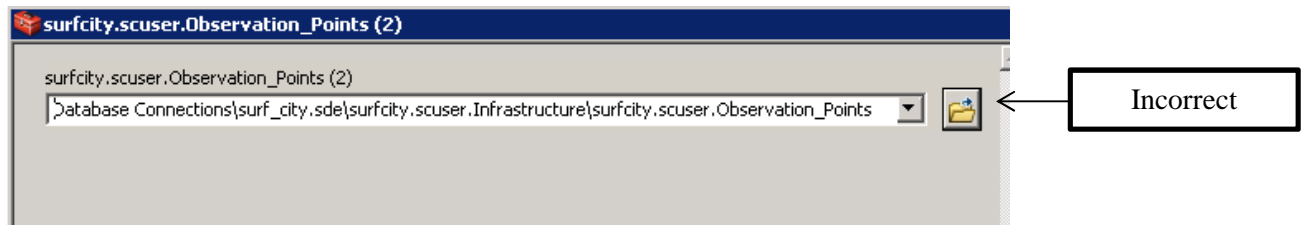


4. Update the Observation Point, raster 2005, and raster 2010 with the updated layers.



Double click on the input, and select the updated layer from the dropdown list. The layer needs to be in the map document. Do NOT select the updated layer by navigating to the database directly. Doing this will create a space in the layer name and cause the web application to fail.





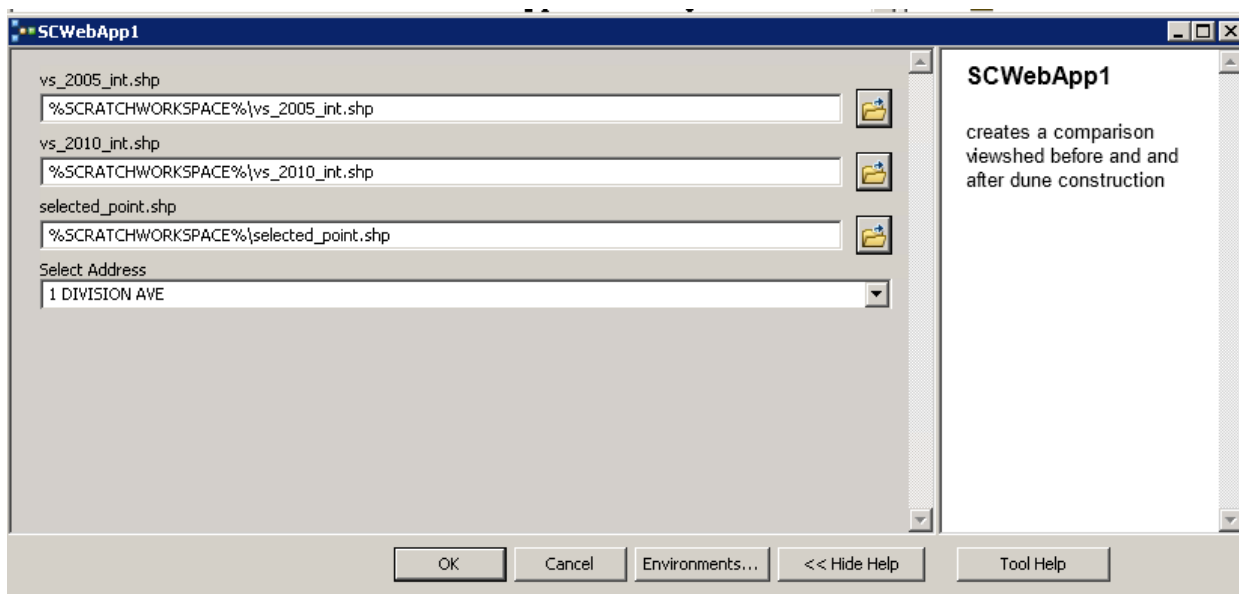
Once the observation point layer and the raster's have been correctly updated within the model, save the model and close it.

Step 3: Update Geoprocessing Service

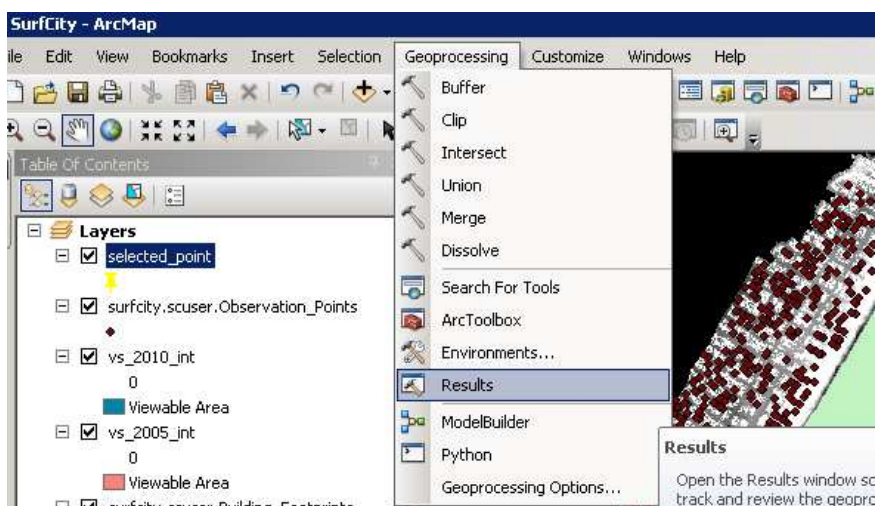
The following section provides a brief overview of updating a geoprocessing service. More information can be found at ArcGIS Resources at

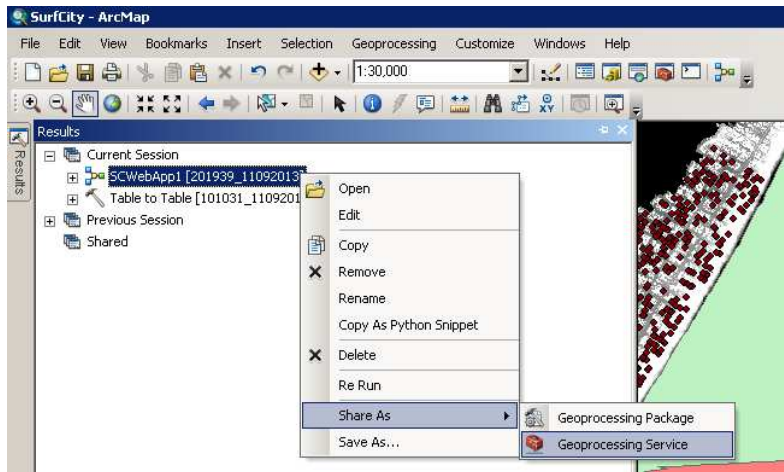
http://resources.arcgis.com/en/help/main/10.1/index.html#/A_quick_tour_of_authoring_and_sharing_geoprocessing_services/00570000007q000000/.

1. Pause the currently running geoprocessing service.
 - a. Login to ArcGIS Server Manager (<http://152.46.17.82/arcgis/manager/login.html?url=>).
 - i. User name: gis_admin
 - ii. Password: gis_2013
 - b. Navigate to the geoprocessing service (Site>surf_city)
 - c. Stop the "SurfCity_Viewshed" geoprocessing service by clicking on the stop button.
2. In ArcMap, add all inputs to the map if they are not already there. There are five inputs
 - a. Observation Points
 - b. 2005 Raster
 - c. 2010 Raster
 - d. Building Footprints
 - e. Atlantic Ocean
3. From the ArcCatalog window within ArcMap, navigate to the Surf City web application project folder (C:\Users\rralbrit\Documents\Surf_City\WebApp1).
4. Double click on the WebApp1 Model. Keep all defaults, and click OK.



5. Once the model runs successfully, navigate to the results window (Geoprocessing>Results). Right click on the successfully run task, and share the results as a geoprocessing service.






6. Set Geoprocessing service parameters.
 - a. Choose to “Overwrite an Existing Service” > SurfCity_Viewshed.
 - b. Under “Pooling” change “The maximum time a client can use the service” to 1800, and change “The maximum time a client will wait to get a service to 120”
 - c. Click on SCWebApp1. Change the name to “Viewshed Analysis”.
 - d. Click on Item Description. Give geoprocessing tool a short summary and tags.
7. Analyze geoprocessing results and address any errors.
8. Publish results. (*Note: If publishing the service fails try publishing the results as a new service. Give the service a different name then the one specified in 6a. Follow all other steps.*)
9. Once the service publishes successfully, visit the service REST endpoint and test it.
 - a. Visit http://152.46.17.82/arcgis/rest/services/surf_city and navigate to the geoprocessing service REST service endpoint. For the current service that link is http://152.46.17.82/arcgis/rest/services/surf_city/SurfCity_Viewshed/GPServer/Surf%20City%20Viewshed
 - b. Scroll to the bottom of the page for the geoprocessing service endpoint and click the “submit job” link.
 - c. Keep all defaults and click “Get Job (GET)”. Give the job a minute to run then click “Check Job Details Again”. If the new window says “*esriJobMessageTypeInformative: Executing...*” give the job a few more seconds to complete and click “Check Job Details Again.”
 - i. If the job runs successfully, the Job status will read “esriJobSucceeded” and links will be provided to the results. You are ready to update the viewer for flex file.
 - ii. If the job fails, the job status will read “esriJobFailed.”
 1. Navigate back to ArcServer Manager and visit “Logs” to find out what the error is. Address the error message within the ArcMap model and rerun steps 2-9.

Step 4: Update Viewer for Flex File

Once the geoprocessing service has been successfully publish and tested from the REST endpoint, the Flex application will need to be updated. These instructions assume this is being updated within the 152.46.17.82 server environment.

1. In your web browser navigate to the geoprocessing service REST service endpoint.
2. Open View for Flex and choose to edit () the Surf_City application.
3. Click the widgets tab and edit the Viewshed Analysis tool located under the “Widgets in this Application.”
4. Under “Task URL” copy and paste the REST service endpoint URL and click “Browse.”
5. Examine input, output, and layer order tabs.

a. Input

Widget Settings

Task URL: Browse

Input (1) | Output (3) | Layer Order | Options

Name: Select_Address Default value: 1 DIMENSION AVE ▼

Label:

Visible: ☒








Tooltip text:

b. Outputs

Widget Settings

Task URL: Browse

Input (1) | Output (3) | Layer Order | Options

Name	Label	Type	Visible	Required	Action
selected_point_shp	Selected House	featurerecordset	true	true	 
vs_2005_int_shp	2005 Viewshed	featurerecordset	true	true	  
vs_2010_int_shp	2010 Viewshed	featurerecordset	true	true	 

Selected Point Settings

Widget Settings

Task URL: [Browse](#)

Input (1) **Output (3)** Layer Order Options

Name: selected_point_shp

Label:

Visible: ☒

Renderer Pop-up

Type: Simple

Symbol Type: Simple Marker

Fill color: Outline color: Width: Alpha: Size:

[Go Back](#)

Viewshed 2005 Settings

Widget Settings

Task URL: [Browse](#)

Input (1) **Output (3)** Layer Order Options

Name: vs_2005_int_shp

Label:

Visible: ☒

Renderer Pop-up

Type: Unique Value

Field: GRIDCODE

Default symbol Fill color: Outline color: Width: Alpha:

Value: [Remove Symbol](#)

Symbol Fill color: Outline color: Width: Alpha:

[+ Add Symbol](#)

[Go Back](#)

Viewshed 2010 Settings

Widget Settings

Task URL: [Browse](#)

Input (1) **Output (3)** Layer Order Options

Name: vs_2010_int_shp

Label:

Visible: ☒

Renderer Pop-up

Type: Unique Value

Field: GRIDCODE

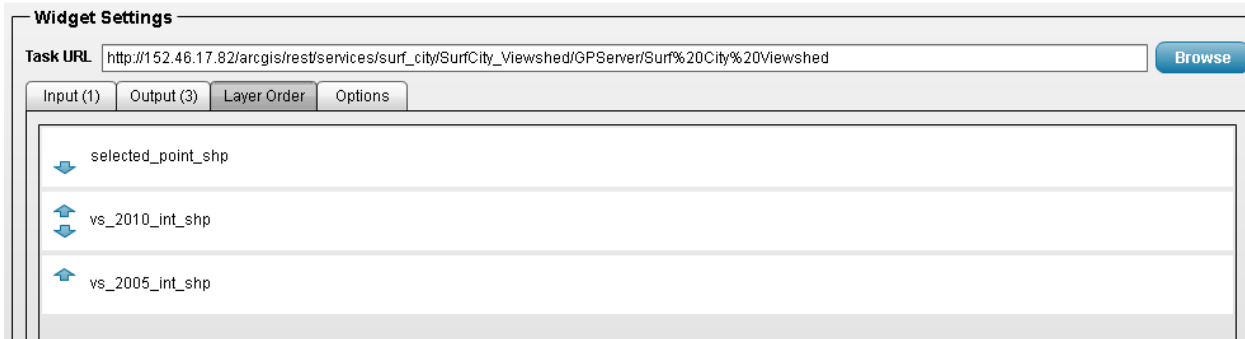
Default symbol Fill color: Outline color: Width: Alpha:

Value: [Remove Symbol](#)

Symbol Fill color: Outline color: Width: Alpha:

[+ Add Symbol](#)

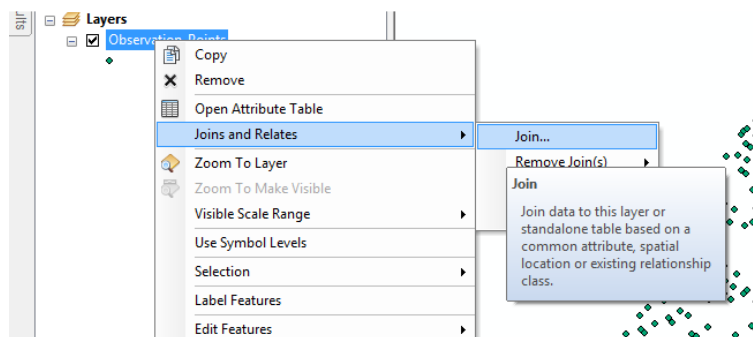
c. Layer Order

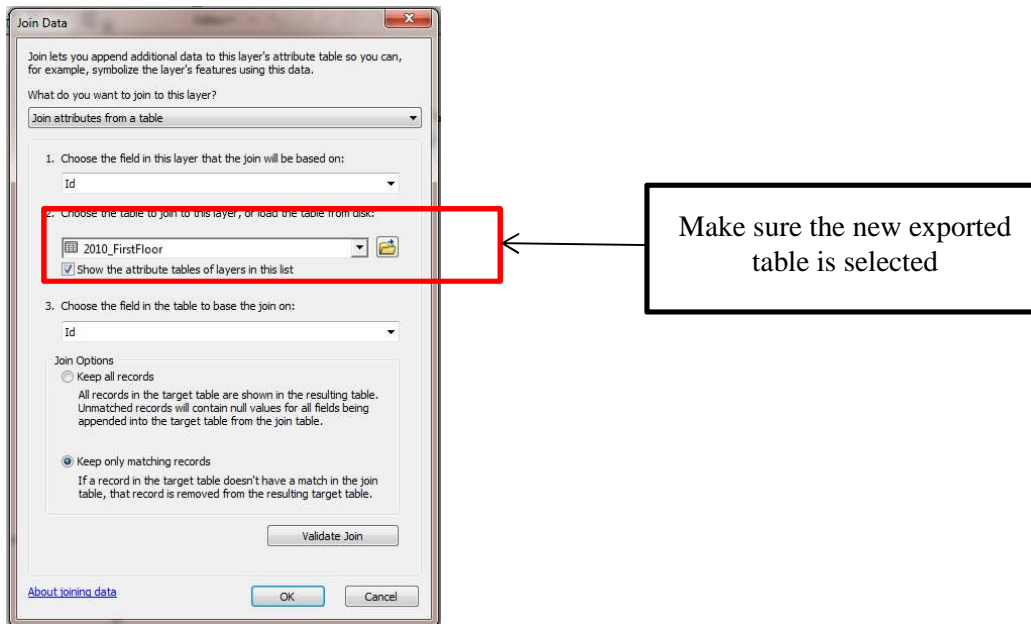


6. Click save within the Widgets Window.
7. Click on the Preview Tab. Select an address in the Viewshed Analysis tool and click submit. Give it some time to run.
8. If the job runs successfully, Click the application URL. Test the application once again from the browser. The cache may need to be cleared first.
9. If the job fails, visit ArcGIS Server Manager, navigate to Logs, and address error statements. Rerun steps outlined in “Step 3: Update Web Application Model” and “Step 4: Update Viewer for Flex File.”

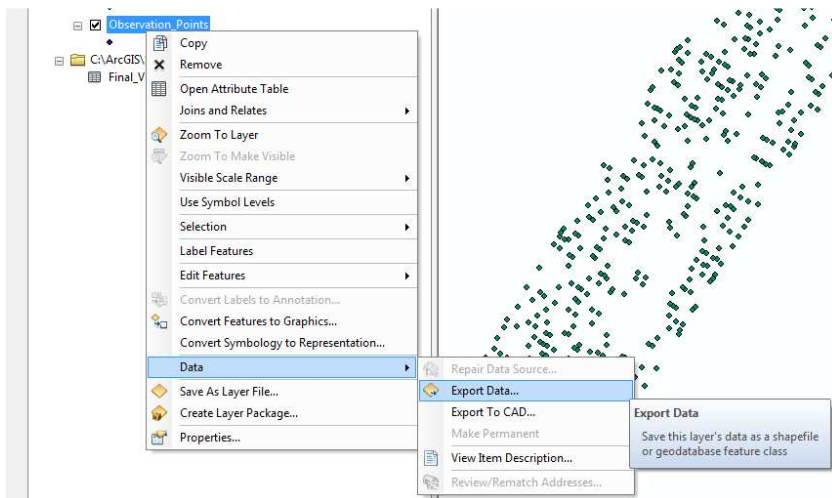
Step 5: Update Operational Layer

1. In Arc Server Manager pause the current map service (SurfCity_Map).
2. In ArcMap, open the current map document containing the observation points operational layer.
 - a. C:\Users\raralbrit\Documents\Surf_City
3. Add the additional observation point file(s) and the new output table for 2005 to ArcMap. Right click on the observation point file and join the output table to the observation point feature class by the “Id” field. Choose to “keep common” files. Click Ok. If a popup window comes up asking if you want field to be indexed, click yes.



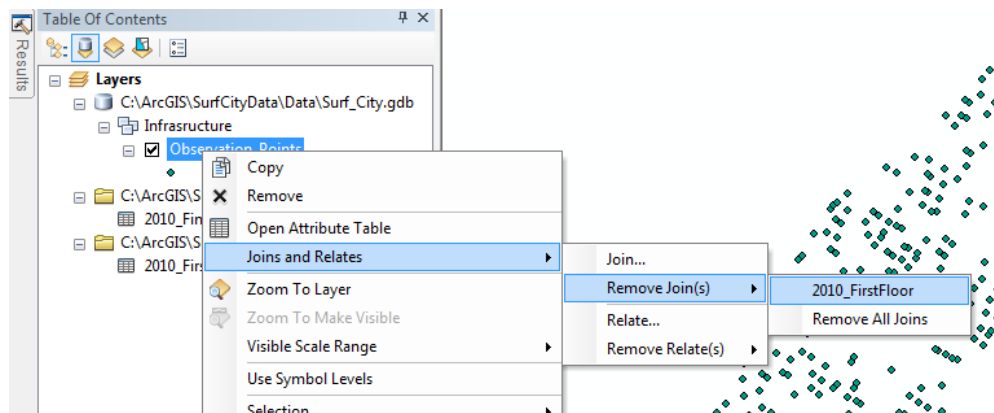


4. Once the .dbf table is joined to the observation point feature class, export the observation point file to a new feature class. This file will be temporary and will hold information for first floor viewsheds.

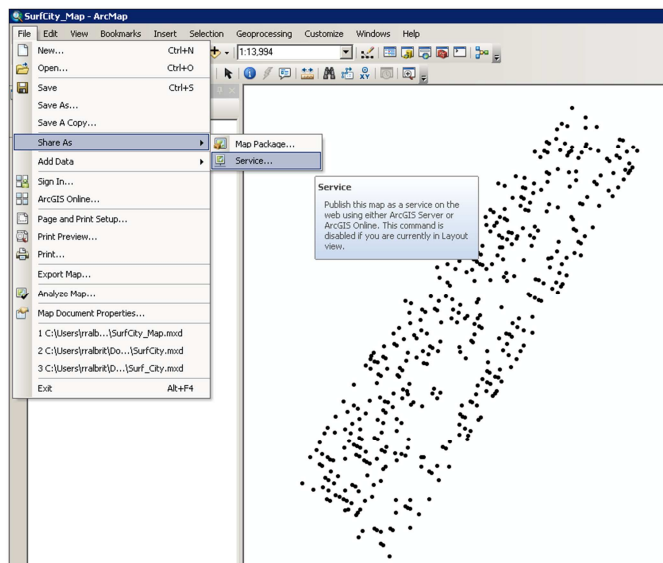


5. Cleanup the attribute field of the new observation point feature class by using the delete field tool (Data Management>Field>Delete Field). Delete all duplicated fields.
6. From the original observation point feature class, select all records that have floors greater than or equal to 1.5. Export these features to a new feature class and call it SecondFlrPoints.
7. Repeat steps 1-3 for the second floor points.
8. Open the new SecondFlrPoint_Web feature class. Use the field calculator to make sure the OFFSETA values are all set to 20, and the FLR_RAN value is equal to 2.

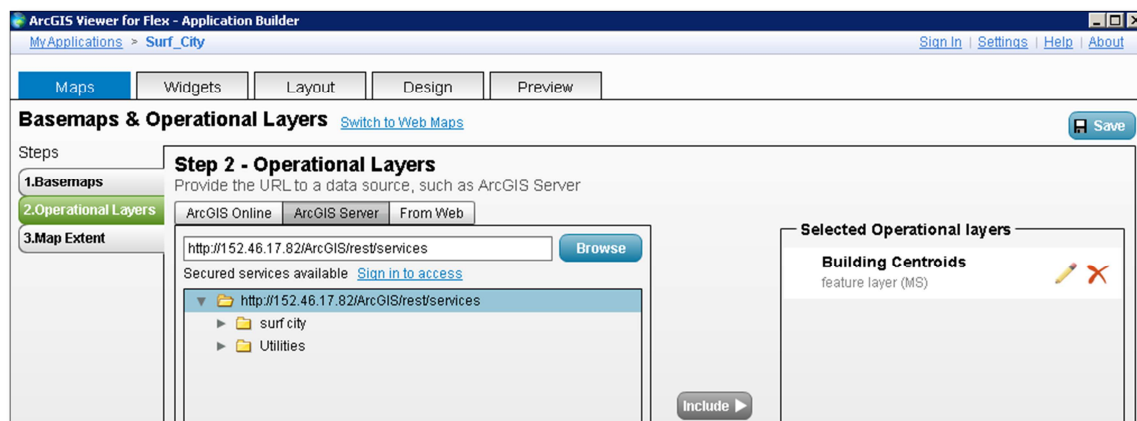
9. Merge the observation point web files for the first and second floor records into a single feature class. The final output should be a feature class contain attribute fields for the 2005 and 2010 viewshed at both the first and second floor heights. This is the layer the Search tool works off of.
10. Repeat steps 3-9 for 2010.



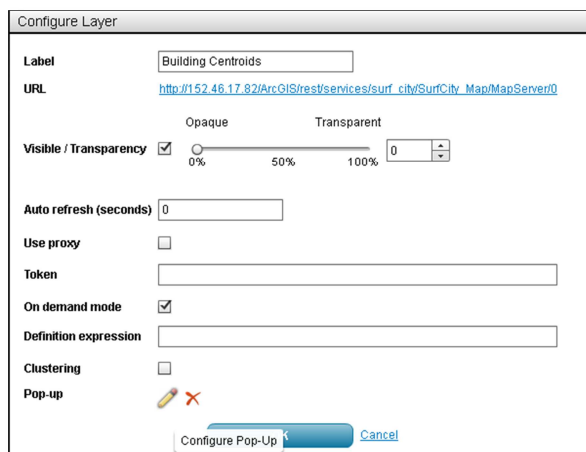
11. Share the map document as a Map Service.



12. Chose to overwrite an existing map service.
13. Analyze service and address any issues.
14. Publish the service.
15. Make sure the service restarted in Arc Server Manager.
16. Open View for Flex, and choose to edit the Surf_City Application.
17. Under the maps tab, choose operational layers from and ArcGIS server. Be sure to click browse.



18. Delete the current Building Centroids operational layer.
19. Navigate to the updated Building Centroids map service and include it.
20. Click on the edit icon next to the Building Centroids operation layer.
21. In the pop-up window, click on the edit button next to “Pop-up” at the bottom.



22. Make sure all field names are unselected.
23. Click save when done, and preview results.

The Web Application model contains a custom script to select a parcel by address. The custom script details are described below.

Script name: MV_GeoprocessingTool.py

Location: C:\Users\rralbrit\Documents\Surf_City\WebApp\SC_Web.tbx\MV_GeoprocessingTool

Script:

```
import arcpy, os
from arcpy import env
from arcpy.sa import*

arcpy.env.workspace = "C:\\DatabaseConnections\\surf_city.sde" #Connection File location
arcpy.env.overwriteOutput = True

#Check out the ArcGIS 3D Analyst extension license
arcpy.CheckOutExtension("3D")
arcpy.CheckOutExtension("Spatial")

def outName(input,post="_Output"):
    """Returns output name."""
    outName=os.path.basename(input).split(".")[0]+post
    return outName

def whereClause(table, field, values):
    """Takes a semicolon-delimited list of values and constructs a SQL WHERE
    clause to select those values within a given field and table."""

    # Add field delimiters
    fieldDelimited = arcpy.AddFieldDelimiters(arcpy.Describe(table).path, field)

    # Split multivalue at semicolons and strip quotes
    valueList = [value[1:-1]
                  if (value.startswith('"') and value.endswith('"'))
                  else value for value in values.split(';')]

    # Determine field type
    fieldType = arcpy.ListFields(table, field)[0].type

    # Add single-quotes for string field values
    if str(fieldType) == 'String':
        valueList = ["'%s'" % value for value in valueList]

    # Format WHERE clause in the form of an IN statement
    whereClause = "%s IN (%s)%"(fieldDelimited, ', '.join(valueList))
    return whereClause

#Variables
ObsPts = arcpy.GetParameterAsText(0)
PointsFL = arcpy.GetParameterAsText(1)
Field = arcpy.GetParameterAsText(2)
Address = arcpy.GetParameterAsText(3)
outPoint = arcpy.GetParameterAsText(4)

#Create Feature Layers
arcpy.MakeFeatureLayer_management(ObsPts, PointsFL)

#Select parcel by hadd attribute
SQL= whereClause(PointsFL,Field,Address)
arcpy.SelectLayerByAttribute_management(PointsFL,"NEW_SELECTION",SQL)
```

```
#Copy selected feature to temp feature class
arcpy.CopyFeatures_management(PointsFL,outPoint)
arcpy.AddMessage(arcpy.GetMessages())

arcpy.AddMessage("Script complete")
```