VIESORE

Visual Impact Evaluation System for Offshore Renewable Energy

Chad Cooper, Snow Winters, Malcolm Williamson, Jackson Cothren

Center for Advanced Spatial Technologies

University of Arkansas, Fayetteville

Robert Sullivan

Argonne National Laboratory

Argonne, Illinois

Credits/disclaimer

- funded by Bureau of Ocean Energy Management (BOEM)
- on-going study
- not meant to represent BOEM policy

Offshore wind 101

- First offshore wind farm – Denmark 1991
- 2010 – 39 offshore farms off coasts of UK, Europe, and Scandinavia
- Largest turbines now over *500 feet* tall
- Power carried to land via transmission cable
- Wind energy contractors submit site proposals to governing bodies
  - Contractors provide photomontages

Offshore windfarms coming to the U.S.

Cape Wind

Offshore windfarms coming to the U.S.

Cape Wind

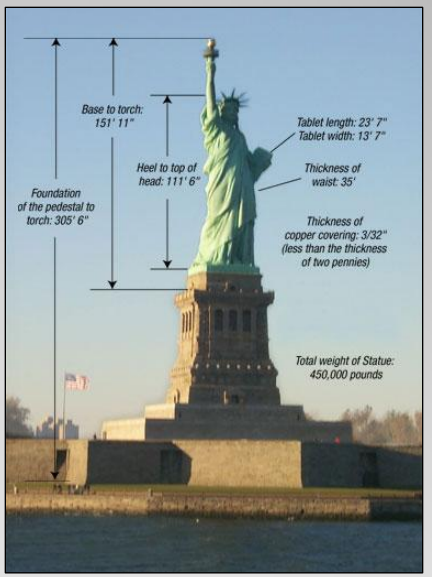
...and in come the proposals

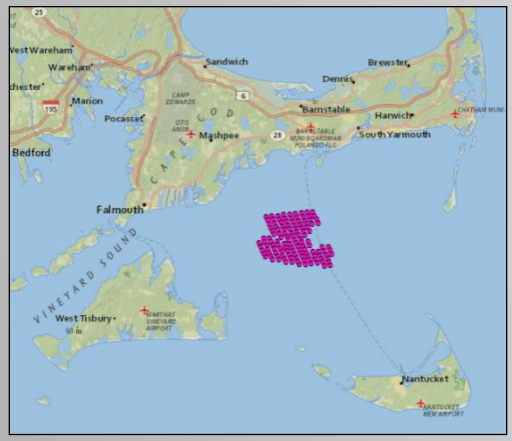
Bureau of Ocean Energy Management (BOEM) reviews all project proposals within federal waters

- offshore oil and gas exploration
- offshore wind
- wave
- tidal flow
- ocean current
- environmental laws and regulations

Offshore wind 101 – Cape Wind

- Siemens SWT 3.6-107
  - rotor diameter – 107m (351ft)
  - total turbine height – 132m (433ft)
- 130 turbines arranged in a grid
- 12.5 miles of transmission cable
- partial generation planned for 2015






VISUAL IMPACT
HAS MOVED
→ to:
MTN. VIEW Call: 964-4333

BOEM system requirements

The system must:

- enable spatial design of offshore facility
- import geospatial data
- allow user control of atmospheric, lighting, wave conditions
- generate spatially accurate and realistic visualizations
- output reports and images
- provide a user-friendly interface

It shall enable users to:

- evaluate photomontages in environmental impact statements (EISs)
- independently assess proposed facilities

Approach

Photomontages

Pros:

- realistic
- accepted
- accurate*

*dependent upon lens used

Cons:

- need photos from every potential point of interest
  - oops, we didn't take a photo from *there*!
- difficult to show different lighting/weather conditions
- human error

Approach

Photomontages

Pros:

- realistic
- accepted
- accurate*

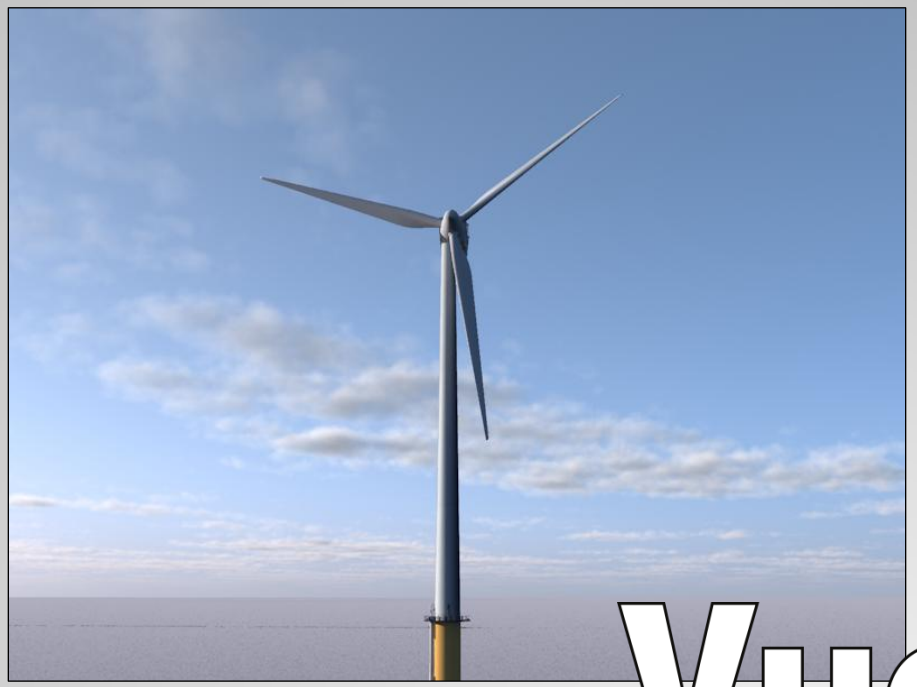
*dependent upon lens used

Cons:

- need photos from every potential point of interest
  - oops, we didn't take a photo from *there*!
- difficult to show different lighting/weather conditions
- human error

Approach

3D visualizations

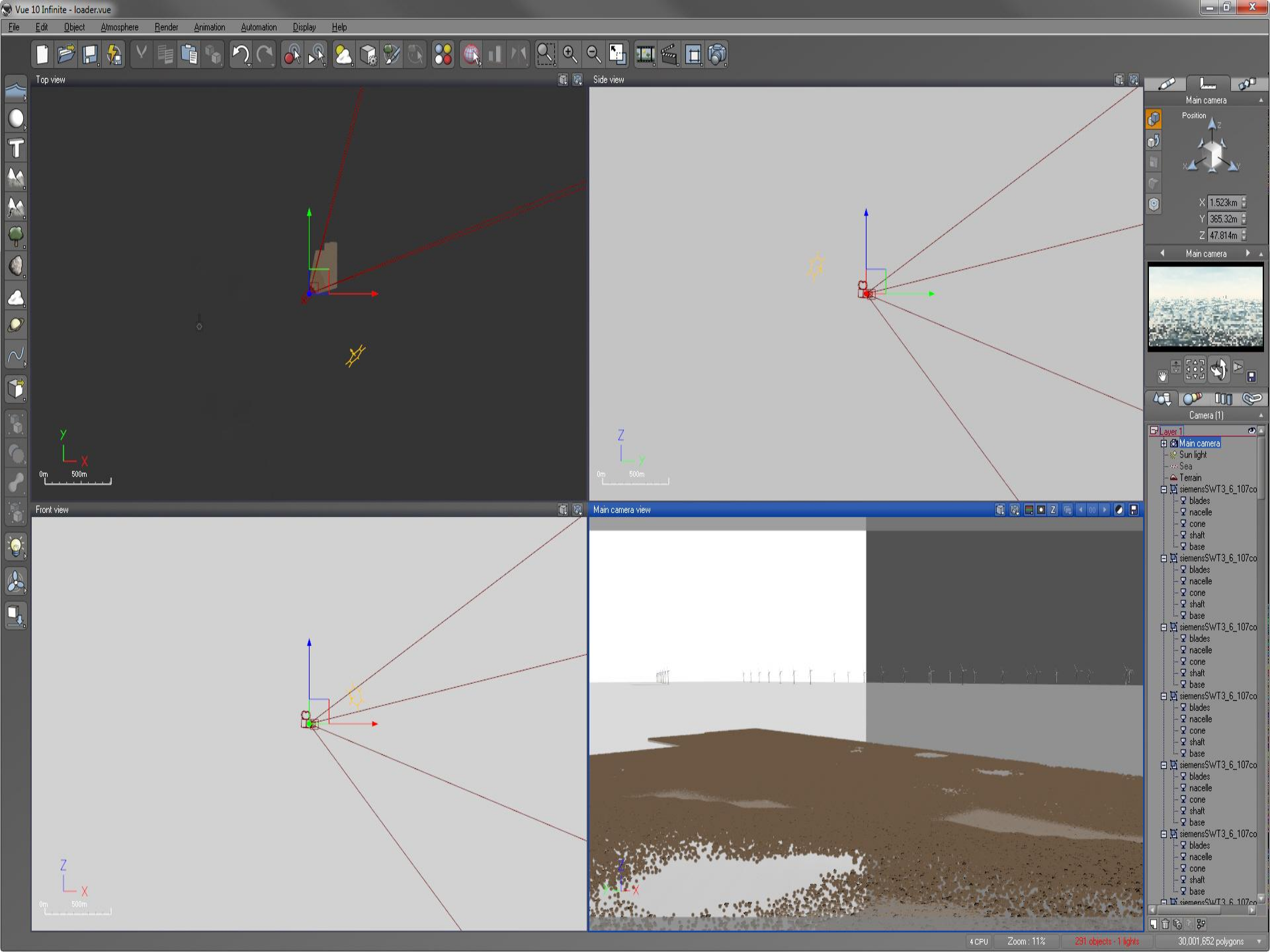


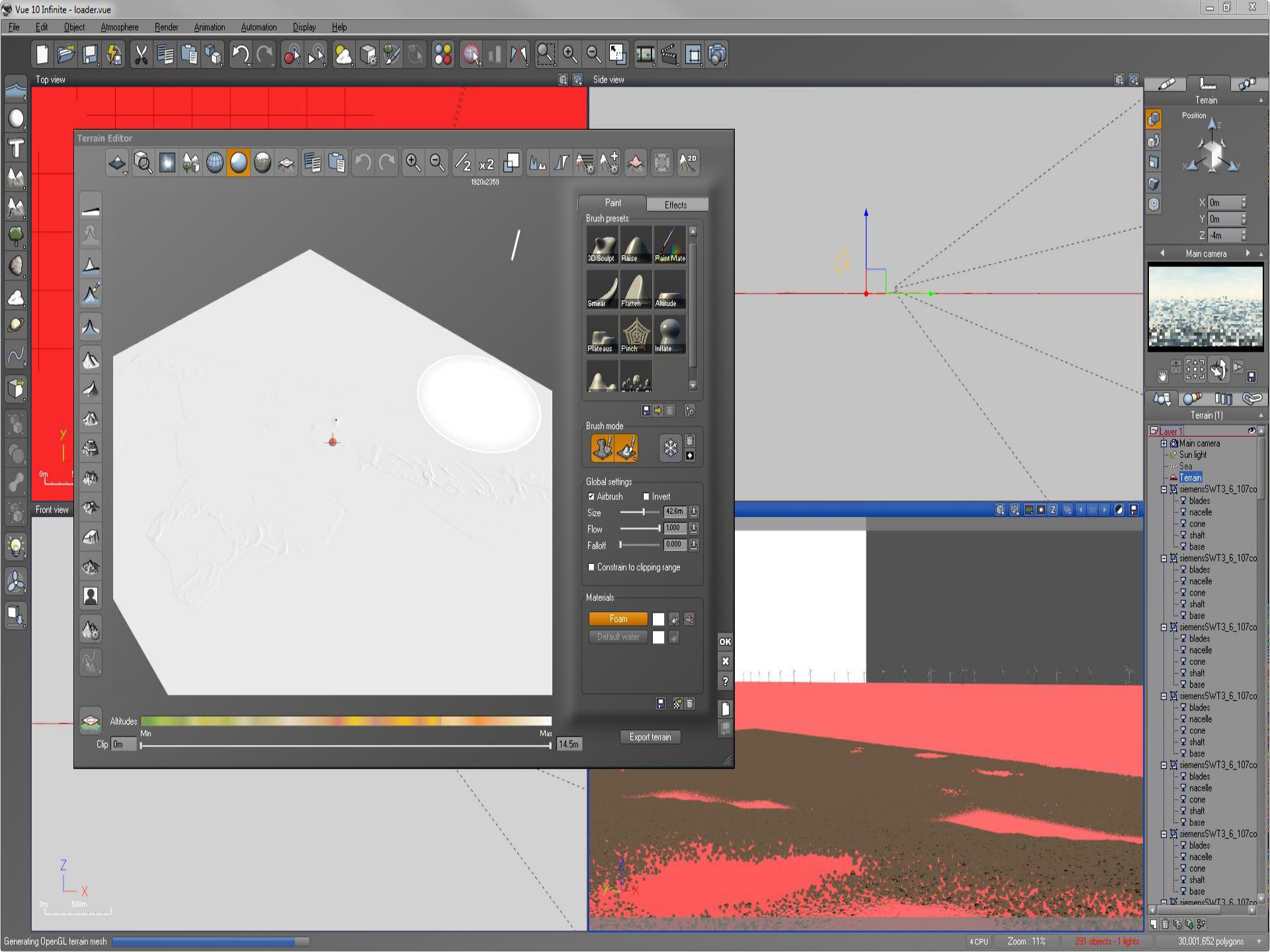
Pros:

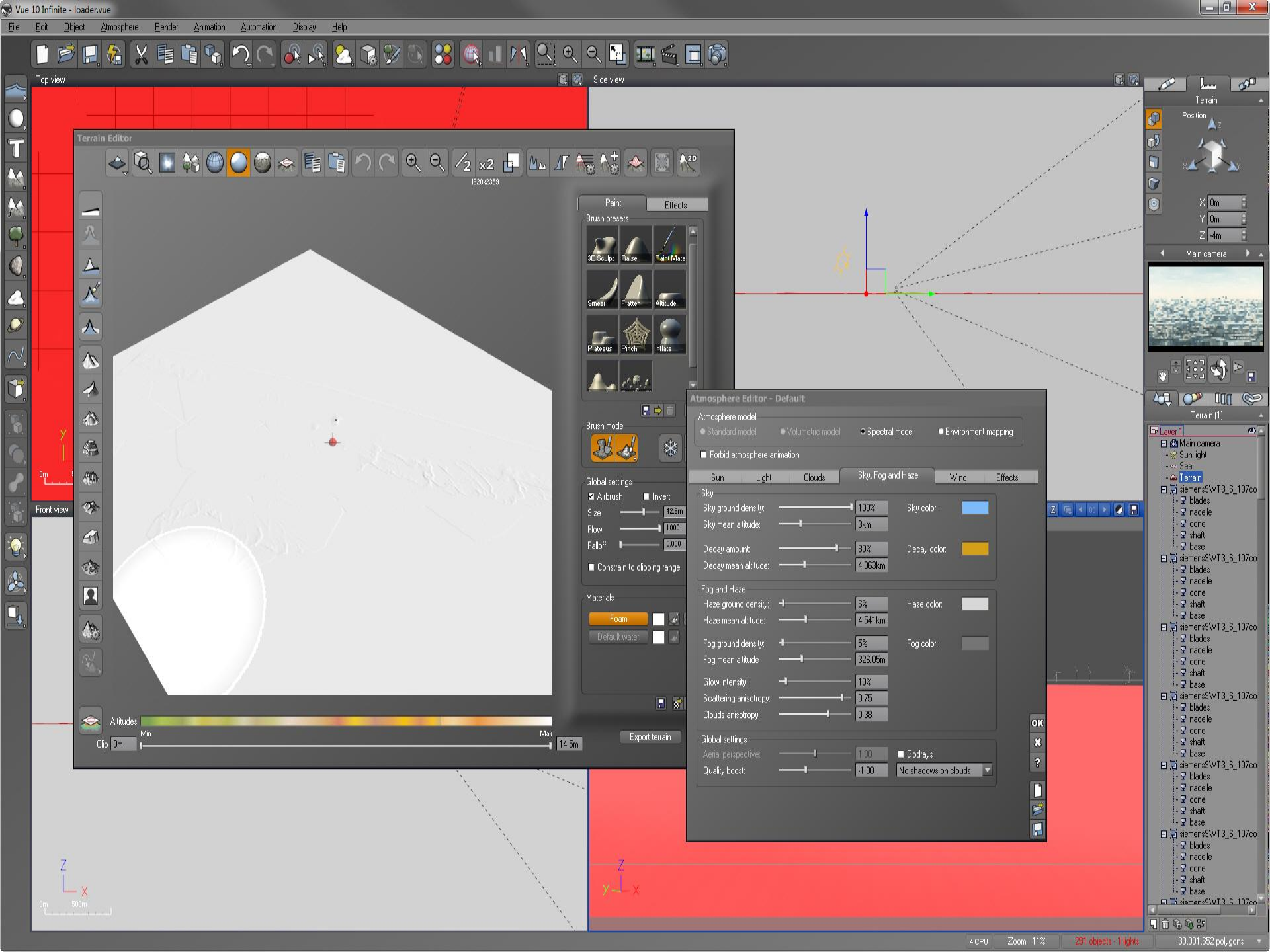
- CAST does 3D
- photorealistic
- Vue Python API

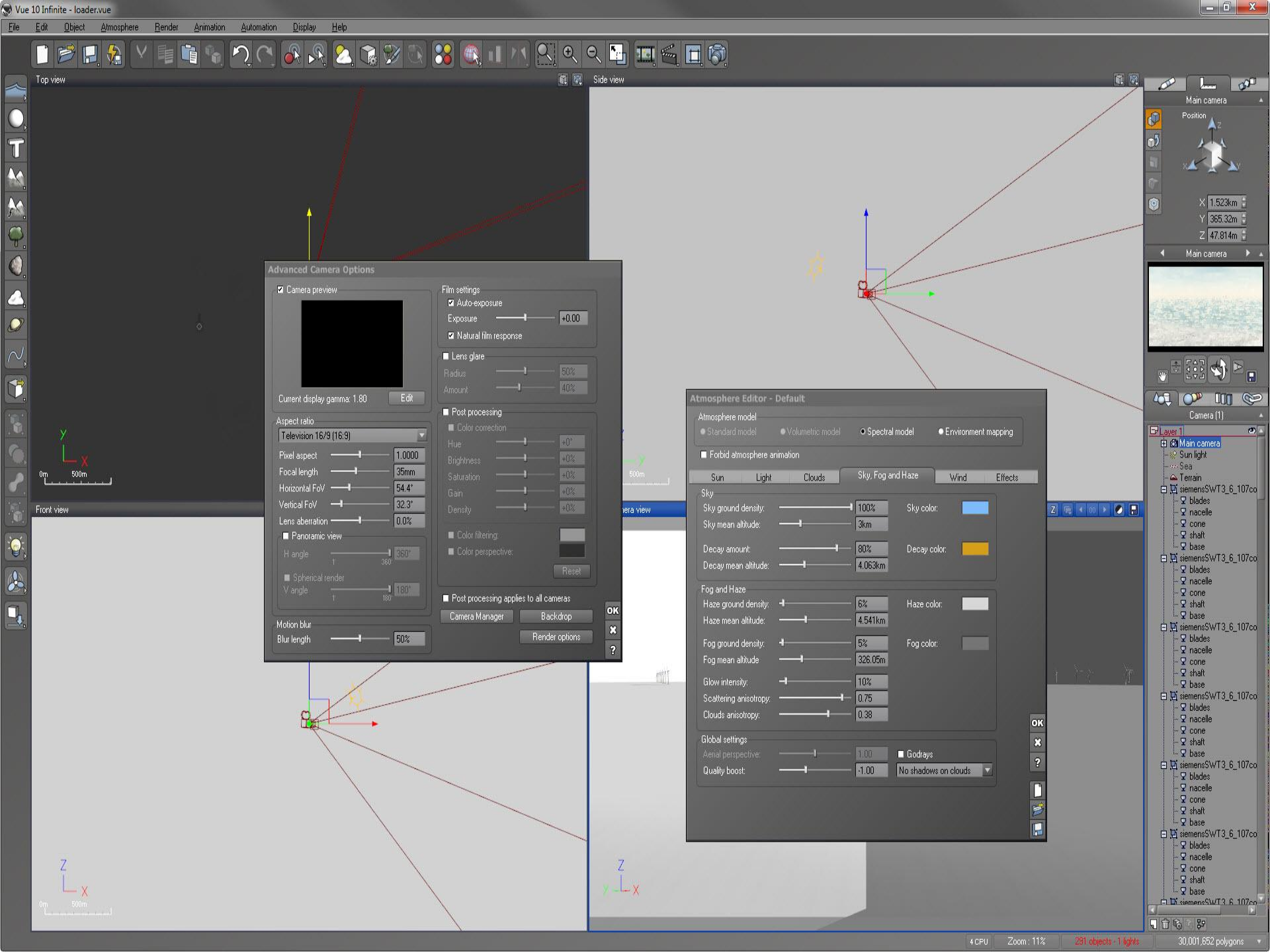
Cons:

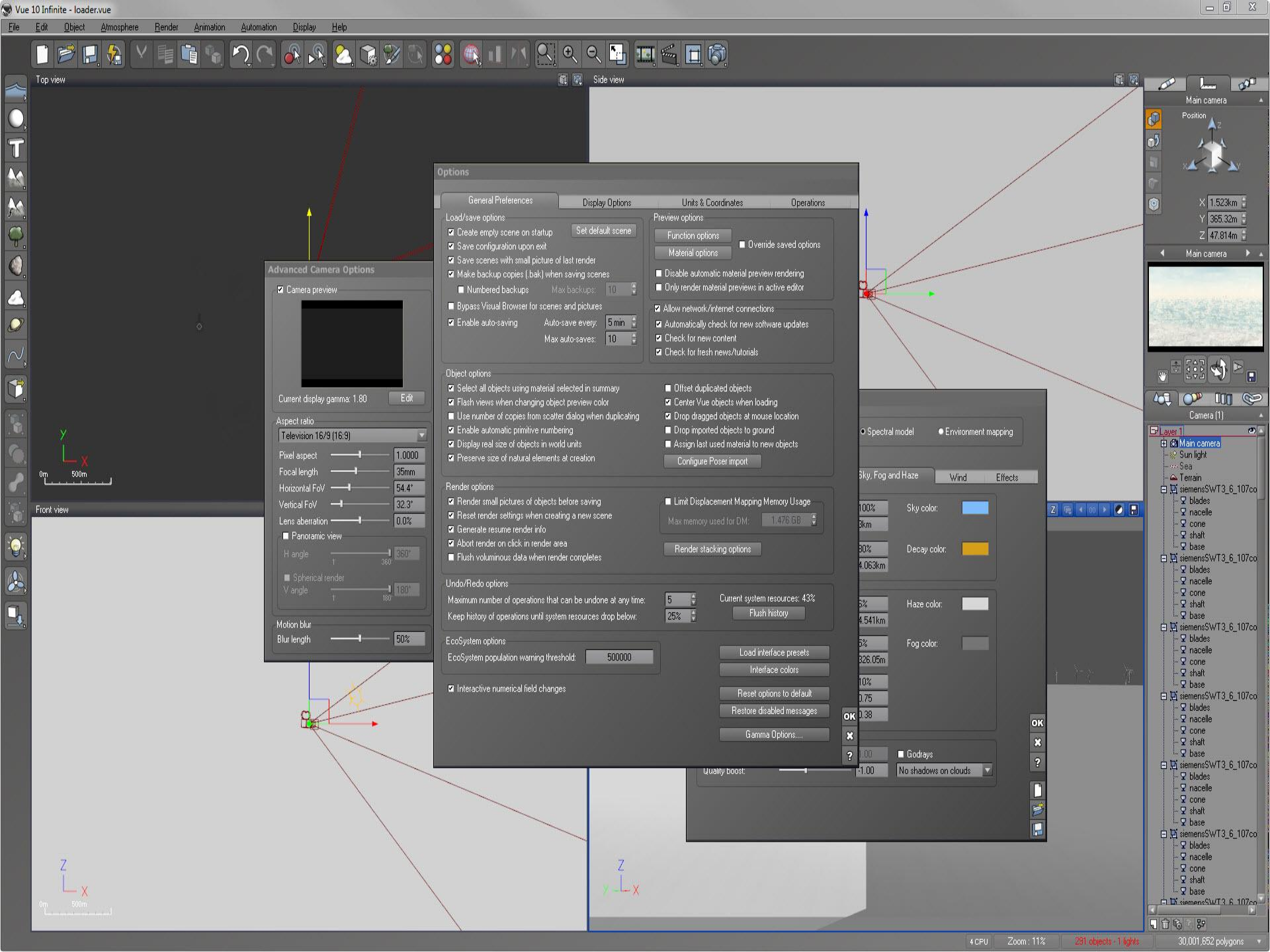
- learning curve
- you need horsepower
- user interface is scary – no likey GIS data


Vue 10 Infinite - loader.vue
File Edit Object Atmosphere Render Animation Automation Display Help
Top view
Side view
Front view
Main camera view
Main camera
Position
X 1.523km
Y 365.32m
Z 47.814m
Camera (1)
Layer 1
Main camera
Sun light
Sea
Terrain
siemensSWT3_6_107co
blades
nacelle
cone
shaft
base
4 CPU
Zoom : 11%
291 objects - 1 lights
30,001,652 polygons


Vue 10 Infinite - loader.vue
File
Edit
Object
Atmosphere
Render
Animation
Automation
Display
Help
Top view
Side view
Front view
Terrain Editor
1920x2359
Paint
Effects
Brush presets
3D Sculpt
Raise
Paint Mate
Smear
Flatten
Altitude
Plateaus
Pinch
Inflate
Brush mode
Global settings
Airbrush
Invert
Size
42.6m
Flow
1.000
Falloff
0.000
Constrain to clipping range
Materials
Foam
Default water
OK
Altitudes
Min
Max
Clip
0m
14.5m
Export terrain
Terrain
Position
X 0m
Y 0m
Z -4m
Main camera
Terrain (1)
Layer 1
Main camera
Sun light
Sea
Terrain
siemensSWT3_6_107co
blades
nacelle
cone
shaft
base
Generating OpenGL terrain mesh
4 CPU
Zoom : 11%
291 objects - 1 lights
30,001,652 polygons


Vue 10 Infinite - loader.vue
File Edit Object Atmosphere Render Animation Automation Display Help
Top view
Side view
Front view
Terrain Editor
Paint
Effects
Brush presets
3D Sculpt
Raise
Paint Mate
Smear
Flatten
Altitude
Plateaus
Pinch
Inflate
Brush mode
Global settings
Airbrush
Invert
Size
Flow
Falloff
Constrain to clipping range
Materials
Foam
Default water
Altitudes
Min
Max
Clip
Export terrain
Atmosphere Editor - Default
Atmosphere model
Standard model
Volumetric model
Spectral model
Environment mapping
Forbid atmosphere animation
Sun
Light
Clouds
Sky, Fog and Haze
Wind
Effects
Sky
Sky ground density:
Sky mean altitude:
Sky color:
Decay amount:
Decay mean altitude:
Decay color:
Fog and Haze
Haze ground density:
Haze mean altitude:
Haze color:
Fog ground density:
Fog mean altitude
Fog color:
Glow intensity:
Scattering anisotropy:
Clouds anisotropy:
Global settings
Aerial perspective:
Godrays
Quality boost:
No shadows on clouds
OK
Terrain
Position
Main camera
Terrain (1)
Layer 1
Main camera
Sun light
Sea
Terrain
siemensSWT3_6_107co
blades
nacelle
cone
shaft
base
4 CPU
Zoom : 11%
291 objects - 1 lights
30,001,652 polygons


Vue 10 Infinite - loader.vue
File Edit Object Atmosphere Render Animation Automation Display Help
Top view
Side view
Front view
Main camera
Position
X 1.523km
Y 365.32m
Z 47.814m
Camera (1)
Layer 1
Main camera
Sun light
Sea
Terrain
siemensSWT3_6_107co
blades
nacelle
cone
shaft
base
Advanced Camera Options
Camera preview
Current display gamma: 1.80
Edit
Aspect ratio
Television 16/9 (16:9)
Pixel aspect 1.0000
Focal length 35mm
Horizontal FoV 54.4°
Vertical FoV 32.3°
Lens aberration 0.0%
Panoramic view
H angle 360°
Spherical render
V angle 180°
Motion blur
Blur length 50%
Film settings
Auto-exposure
Exposure +0.00
Natural film response
Lens glare
Radius 50%
Amount 40%
Post processing
Color correction
Hue +0°
Brightness +0%
Saturation +0%
Gain +0%
Density +0%
Color filtering:
Color perspective:
Reset
Post processing applies to all cameras
Camera Manager
Backdrop
Render options
OK
Atmosphere Editor - Default
Atmosphere model
Standard model
Volumetric model
Spectral model
Environment mapping
Forbid atmosphere animation
Sun Light Clouds Sky, Fog and Haze Wind Effects
Sky
Sky ground density: 100%
Sky mean altitude: 3km
Sky color:
Decay amount: 80%
Decay mean altitude: 4.063km
Decay color:
Fog and Haze
Haze ground density: 6%
Haze mean altitude: 4.541km
Haze color:
Fog ground density: 5%
Fog mean altitude 326.05m
Fog color:
Glow intensity: 10%
Scattering anisotropy: 0.75
Clouds anisotropy: 0.38
Global settings
Aerial perspective: 1.00
Godrays
Quality boost: -1.00
No shadows on clouds
OK
4 CPU
Zoom : 11%
291 objects - 1 lights
30,001,652 polygons


Vue 10 Infinite - loader.vue
File Edit Object Atmosphere Render Animation Automation Display Help
Top view
Side view
Front view
Main camera
Position
Camera (1)
Advanced Camera Options
Camera preview
Current display gamma: 1.80
Edit
Aspect ratio
Television 16/9 (16:9)
Pixel aspect 1.0000
Focal length 35mm
Horizontal FoV 54.4°
Vertical FoV 32.3°
Lens aberration 0.0%
Panoramic view
H angle 360°
Spherical render
V angle 180°
Motion blur
Blur length 50%
Options
General Preferences
Display Options
Units & Coordinates
Operations
Load/save options
Create empty scene on startup
Set default scene
Save configuration upon exit
Save scenes with small picture of last render
Make backup copies (.bak) when saving scenes
Numbered backups
Max backups: 10
Bypass Visual Browser for scenes and pictures
Enable auto-saving
Auto-save every: 5 min
Max auto-saves: 10
Preview options
Function options
Material options
Override saved options
Disable automatic material preview rendering
Only render material previews in active editor
Allow network/internet connections
Automatically check for new software updates
Check for new content
Check for fresh news/tutorials
Object options
Select all objects using material selected in summary
Flash views when changing object preview color
Use number of copies from scatter dialog when duplicating
Enable automatic primitive numbering
Display real size of objects in world units
Preserve size of natural elements at creation
Offset duplicated objects
Center Vue objects when loading
Drop dragged objects at mouse location
Drop imported objects to ground
Assign last used material to new objects
Configure Poser import
Render options
Render small pictures of objects before saving
Reset render settings when creating a new scene
Generate resume render info
Abort render on click in render area
Flush voluminous data when render completes
Limit Displacement Mapping Memory Usage
Max memory used for DM: 1.476 GB
Render stacking options
Undo/Redo options
Maximum number of operations that can be undone at any time: 5
Keep history of operations until system resources drop below: 25%
Current system resources: 43%
Flush history
EcoSystem options
EcoSystem population warning threshold: 500000
Interactive numerical field changes
Load interface presets
Interface colors
Reset options to default
Restore disabled messages
Gamma Options....
OK
Spectral model
Environment mapping
Sky, Fog and Haze
Wind
Effects
Sky color:
Decay color:
Haze color:
Fog color:
Godrays
Quality boost:
No shadows on clouds
Layer 1
Main camera
Sun light
Sea
Terrain
siemensSWT3_6_107co
blades
nacelle
cone
shaft
base
4 CPU
Zoom : 11%
291 objects - 1 lights
30,001,652 polygons

Approach

3D visualizations *via GIS interface*

Pros:

- CAST does 3D
- photorealistic
- Vue Python API

Cons:

- user interface is scary
- learning curve
- you need horsepower, but affordable


esri®


Vue™
Infinite R10


python™


2012

System design

ArcGIS 10.0
friendly interface

ArcGIS 10.0
script tools

Renders

Reports

python

Videos

batch

Vue
Infinite R10

Input data

System design

ArcGIS 10.0
friendly interface

ArcGIS 10.0
script tools

Renders

Reports

python

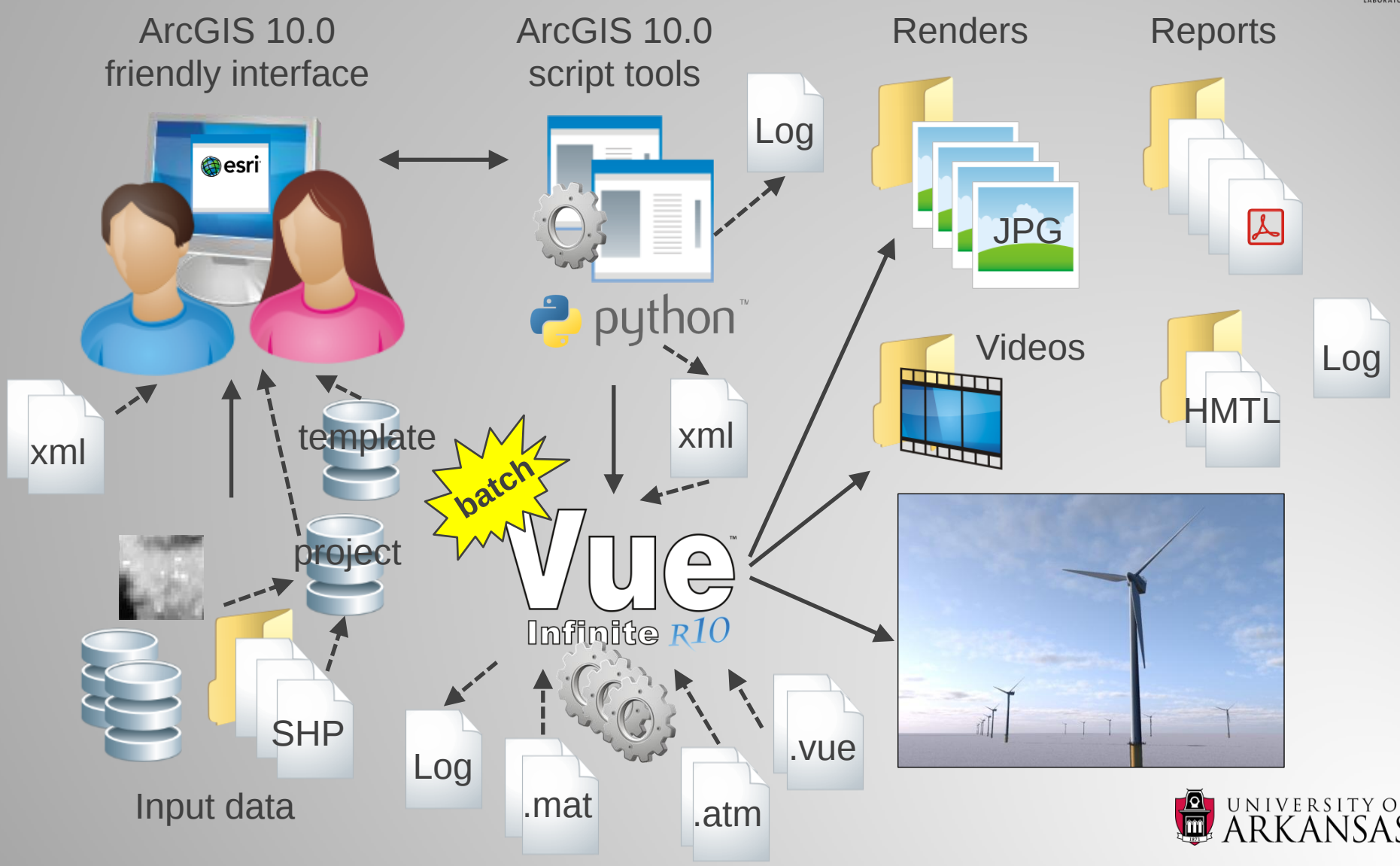
Videos

batch

Vue
Infinite R10

Input data

System design



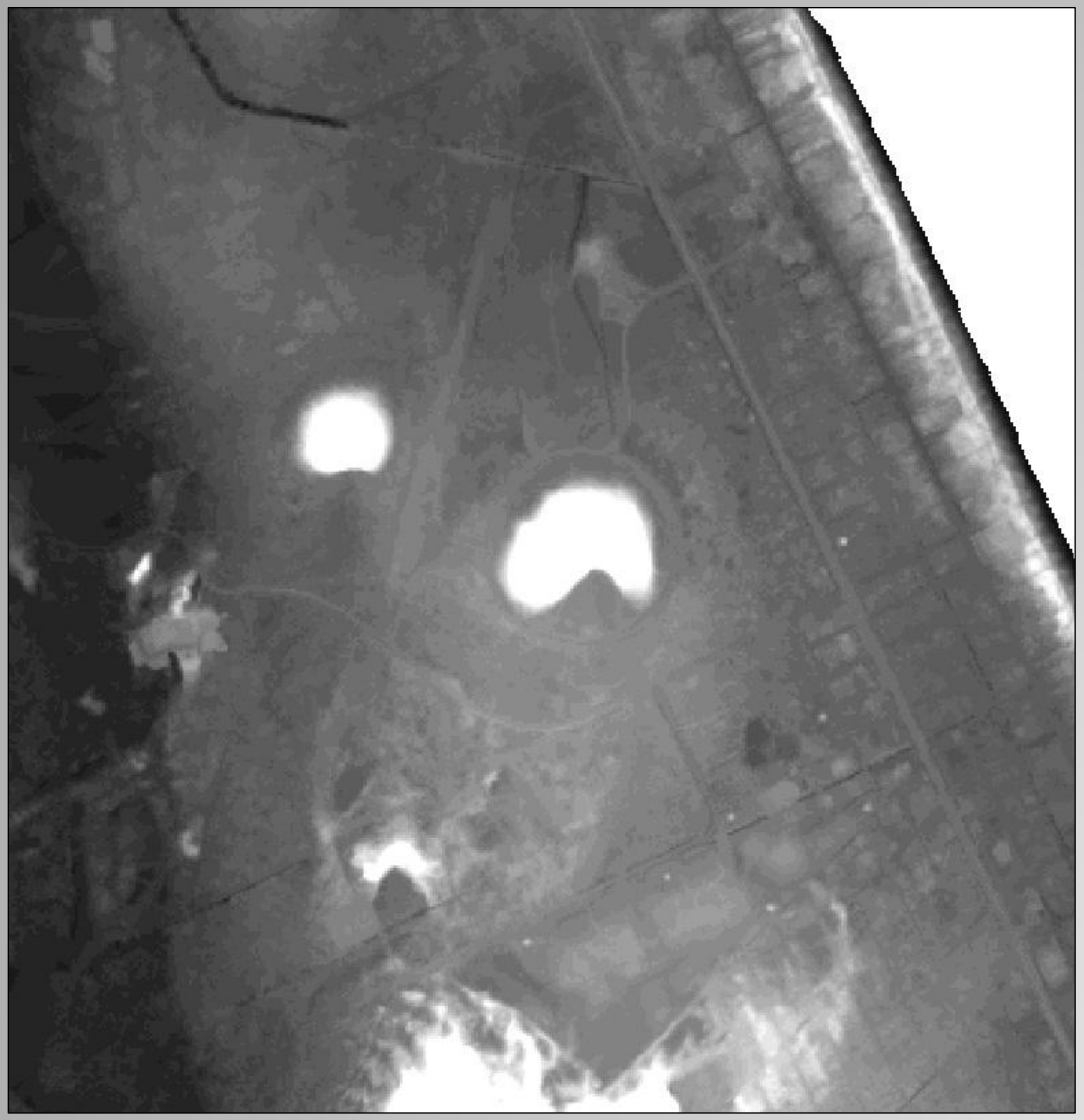
Development to date

- translate data for import into Vue
  - digital elevation
  - wind turbine generator (WTG) point locations
  - key observation points (KOPs)
- import above data into Vue
- move objects around as needed, adjust size
- import 3D WTG model
- set camera and sun position
- render to jpeg

Development to date

- translate data for import into Vue
  - digital elevation
  - wind turbine (WTG) point locations
  - key observation points (KOPs)
- import above data into Vue
- place objects around KOPs as needed, adjust size
- import 3D WTG model
- set camera and sun position
- render to jpeg

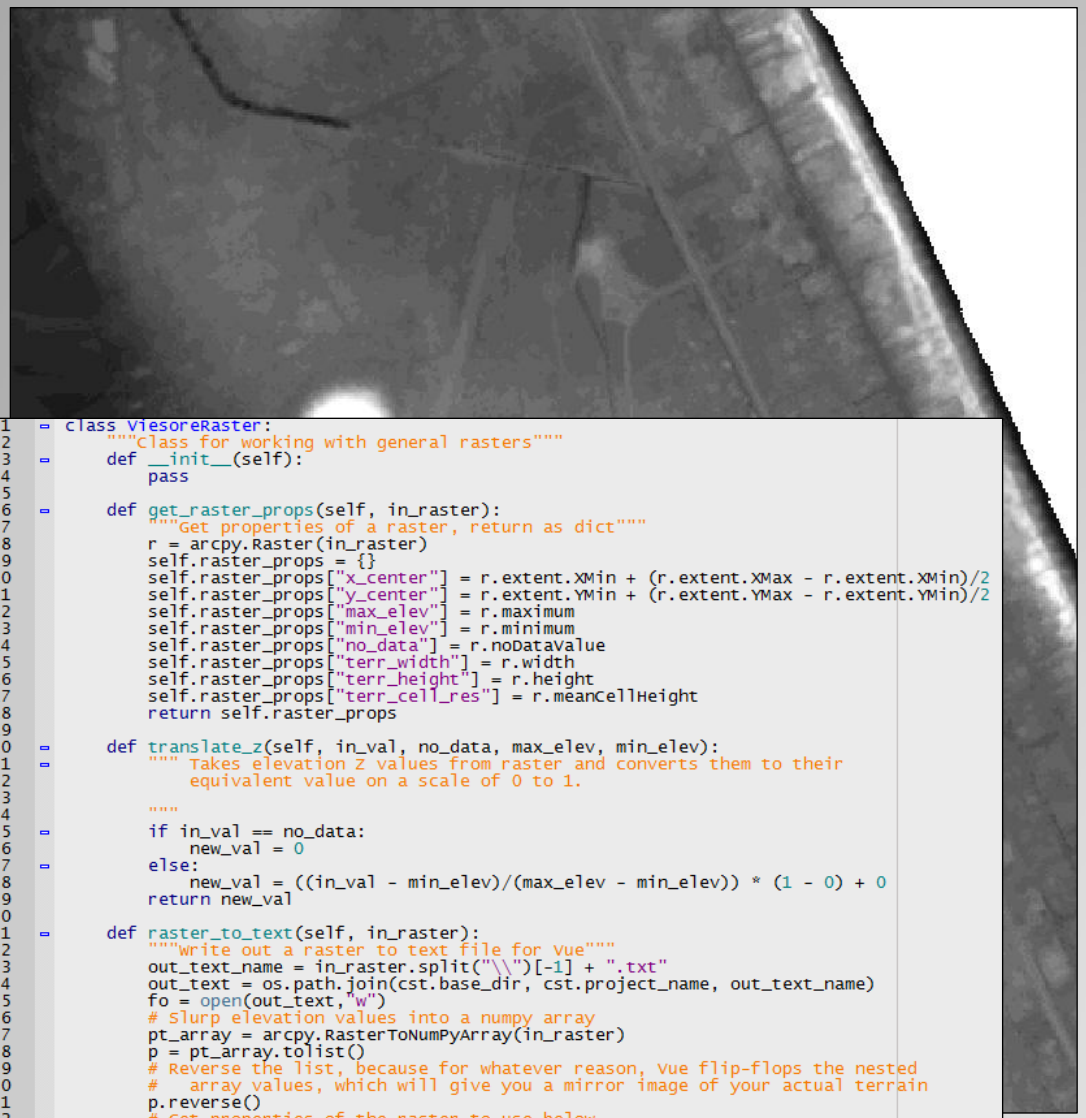
Data translations



The Center for Advanced Spatial Technologies
CAST
University of Arkansas, Fayetteville
U of A

Argonne
NATIONAL
LABORATORY

UNIVERSITY OF
ARKANSAS

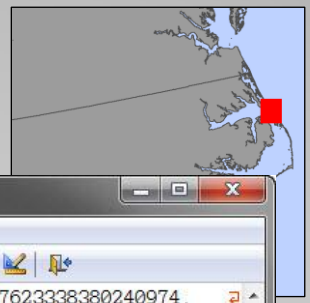


```python
class ViesoreRaster:
    """class for working with general rasters"""
    def __init__(self):
        pass

    def get_raster_props(self, in_raster):
        """Get properties of a raster, return as dict"""
        r = arcpy.Raster(in_raster)
        self.raster_props = {}
        self.raster_props["x_center"] = r.extent.XMin + (r.extent.XMax - r.extent.XMin)/2
        self.raster_props["y_center"] = r.extent.YMin + (r.extent.YMax - r.extent.YMin)/2
        self.raster_props["max_elev"] = r.maximum
        self.raster_props["min_elev"] = r.minimum
        self.raster_props["no_data"] = r.noDataValue
        self.raster_props["terr_width"] = r.width
        self.raster_props["terr_height"] = r.height
        self.raster_props["terr_cell_res"] = r.meanCellHeight
        return self.raster_props

    def translate_z(self, in_val, no_data, max_elev, min_elev):
        """ Takes elevation Z values from raster and converts them to their
            equivalent value on a scale of 0 to 1.

        """
        if in_val == no_data:
            new_val = 0
        else:
            new_val = ((in_val - min_elev)/(max_elev - min_elev)) * (1 - 0) + 0
        return new_val

    def raster_to_text(self, in_raster):
        """Write out a raster to text file for Vue"""
        out_text_name = in_raster.split("\\")[-1] + ".txt"
        out_text = os.path.join(cst.base_dir, cst.project_name, out_text_name)
        fo = open(out_text,"w")
        # Slurp elevation values into a numpy array
        pt_array = arcpy.RasterToNumPyArray(in_raster)
        p = pt_array.tolist()
        # Reverse the list, because for whatever reason, Vue flip-flops the nested
        #   array values, which will give you a mirror image of your actual terrain
        p.reverse()
        # Get properties of the raster to use below
        rp = self.get_raster_props(in_raster)
        new_list = []
        for each in p:
            new_inner_list = []
            for every in each:
                new_inner_list.append(self.translate_z(every, rp["no_data"],
                                                       rp["max_elev"],
                                                       rp["min_elev"]))
            fo.write(str(new_inner_list).strip('[]') + "\n")
        fo.close()
        return out_text_name
```

The Center for Advanced Spatial Technologies
CAST
University of Arkansas, Fayetteville
U of A

Argonne NATIONAL LABORATORY

UNIVERSITY OF ARKANSAS



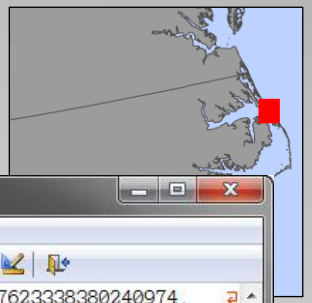
```
class ViesoreRaster:
    """class for working with general rasters"""
    def __init__(self):
        pass

    def get_raster_props(self, in_raster):
        """Get properties of a raster, return as dict"""
        r = arcpy.Raster(in_raster)
        self.raster_props = {}
        self.raster_props["x_center"] = r.extent.XMin + (r.extent.X
        self.raster_props["y_center"] = r.extent.YMin + (r.extent.Y
        self.raster_props["max_elev"] = r.maximum
        self.raster_props["min_elev"] = r.minimum
        self.raster_props["no_data"] = r.noDataValue
        self.raster_props["terr_width"] = r.width
        self.raster_props["terr_height"] = r.height
        self.raster_props["terr_cell_res"] = r.meanCellHeight
        return self.raster_props

    def translate_z(self, in_val, no_data, max_elev, min_elev):
        """ Takes elevation Z values from raster and converts them
            equivalent value on a scale of 0 to 1.

        """
        if in_val == no_data:
            new_val = 0
        else:
            new_val = ((in_val - min_elev)/(max_elev - min_elev)) *
        return new_val

    def raster_to_text(self, in_raster):
        """Write out a raster to text file for Vue"""
        out_text_name = in_raster.split("\\")[-1] + ".txt"
        out_text = os.path.join(cst.base_dir, cst.project_name, out
        fo = open(out_text,"w")
        # Slurp elevation values into a numpy array
        pt_array = arcpy.RasterToNumPyArray(in_raster)
        p = pt_array.tolist()
        # Reverse the list, because for whatever reason, Vue flip-f
        #   array values, which will give you a mirror image of you
        p.reverse()
        # Get properties of the raster to use below
        rp = self.get_raster_props(in_raster)
        new_list = []
        for each in p:
            new_inner_list = []
            for every in each:
                new_inner_list.append(self.translate_z(every, rp["no_data"],
                                                       rp["max_elev"],
                                                       rp["min_elev"]))
            fo.write(str(new_inner_list).strip('[]') + "\n")
        fo.close()
        return out_text_name
```

NcLidarGCS_HavCurvedPCS.txt - Notepad2 (Administrator)

File Edit View Settings ?

```
0.17626473697193856, 0.17626941489362349, 0.17623338380240974,
0.17409059583006709, 0.16863049866327687, 0.16750434164258635,
0.16739748566334889, 0.16740111929181922, 0.16987398649502636,
0.17393349595966948, 0.17544583675752892, 0.17615749376759246,
0.17383971373914939, 0.17292635896557054, 0.17672483642767997,
0.17684790602842734, 0.17700167480100931, 0.17685808528557118,
0.17641055095990291, 0.17847097212205809, 0.17805913111514368,
0.17126984239383883, 0.16745095104134383, 0.170553453281551,
0.17332714779457481, 0.17033727542923977, 0.17335404699227103,
0.17632976494982569, 0.17632972405628922, 0.17573986430096661,
0.17219406392506664, 0.1683560984142195, 0.1689862447155269,
0.1740431532412578, 0.16992748665290616, 0.16810377121908601,
0.17363451785285575, 0.17632155358204191, 0.17627325062583099,
0.17615064017390727, 0.17621746432905885, 0.17578359830793652,
0.17022927823503048, 0.1737628338905465, 0.17473996366948877,
0.17480815669841315, 0.16974494920947339, 0.16878328452370667,
0.17288344493514379, 0.17500462044249085, 0.17603605402561326,
0.1762807454464714, 0.17627462886028375, 0.17617701155967905,
0.17615177304714111, 0.17417241268332528, 0.16918491393905305,
0.16743795205937889, 0.16912366960506148, 0.17256262158265198,
0.16757859669675401, 0.16733795380393793, 0.16756332137132524,
0.1686154398426313, 0.16873843499268767, 0.17111797385287211,
0.17270915967151595, 0.16910795127084935, 0.16738366413710709,
0.16728530575323708, 0.16727834368978603, 0.16728499708175898,
0.16779305642275333, 0.16818754726673302, 0.16748652564656885,
0.17029673295221165, 0.17120174729202772, 0.16834899983972038,
0.16852115418327795, 0.17031392160605607, 0.17608734327648057,
0.17607805450666209, 0.17327440214282208, 0.17320088137897346,
0.17568097214689093, 0.17603917196756819, 0.17602902053942696,
0.17326620669770432, 0.17199917918776142, 0.17422089160401616,
0.17063286358415145, 0.16748869765037 81, 0.16709466155339292,
0.16706031902558893, 0.16704876949532538, 0.16849119739895432,
```

Ln 1 : 2,360 Col 1 Sel 0 | 36.9 MB | ANSI | CR+LF | INS | Default Text



```python
class ViesoreRaster:
    """class for working with general rasters"""
    def __init__(self):
        pass

    def get_raster_props(self, in_raster):
        """Get properties of a raster, return as dict"""
        r = arcpy.Raster(in_raster)
        self.raster_props = {}
        self.raster_props["x_center"] = r.extent.XMin + (r.extent.X
        self.raster_props["y_center"] = r.extent.YMin + (r.extent.Y
        self.raster_props["max_elev"] = r.maximum
        self.raster_props["min_elev"] = r.minimum
        self.raster_props["no_data"] = r.noDataValue
        self.raster_props["terr_width"] = r.width
        self.raster_props["terr_height"] = r.height
        self.raster_props["terr_cell_res"] = r.meanCellHeight
        return self.raster_props

    def translate_z(self, in_val, no_data, max_elev, min_elev):
        """ Takes elevation Z values from raster and converts them
            equivalent value on a scale of 0 to 1.

        """
        if in_val == no_data:
            new_val = 0
        else:
            new_val = ((in_val - min_elev)/(max_elev - min_e
        return new_val

    def raster_to_text(self, in_raster):
        """Write out a raster to text file for Vue"""
        out_text_name = in_raster.split("\\")[-1] + ".txt"
        out_text = os.path.join(cst.base_dir, cst.project_na
        fo = open(out_text,"w")
        # Slurp elevation values into a numpy array
        pt_array = arcpy.RasterToNumPyArray(in_raster)
        p = pt_array.tolist()
        # Reverse the list, because for whatever reason, Vue
        #   array values, which will give you a mirror image
        p.reverse()
        # Get properties of the raster to use below
        rp = self.get_raster_props(in_raster)
        new_list = []
        for each in p:
            new_inner_list = []
            for every in each:
                new_inner_list.append(self.translate_z(every, rp["no_data"],
                                                       rp["max_elev"],
                                                       rp["min_elev"]))
            fo.write(str(new_inner_list).strip('[]') + "\n")
        fo.close()
        return out_text_name
```

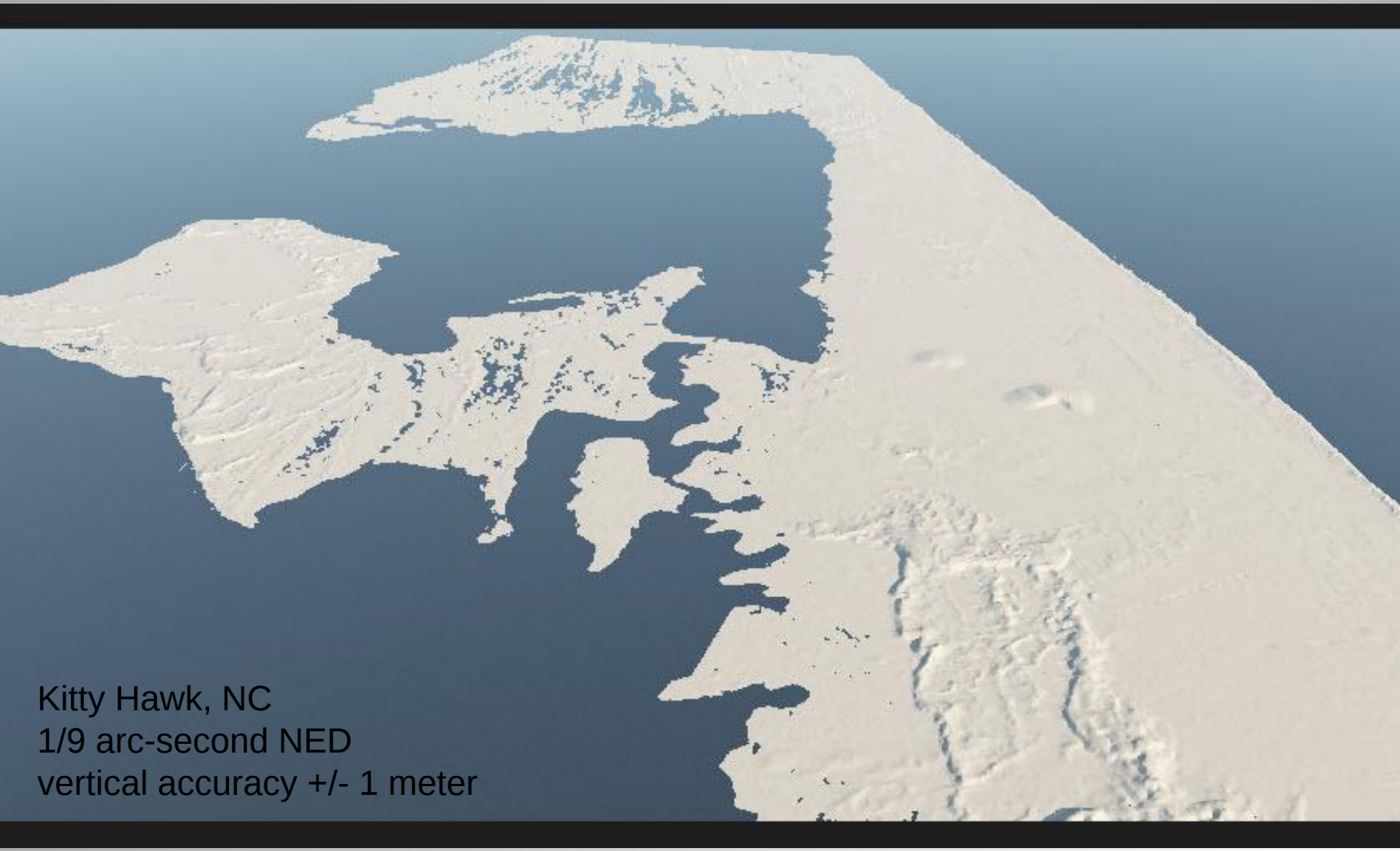
NcLidarGCS_HavCurvedPCS.txt - Notepad2 (Administrator)

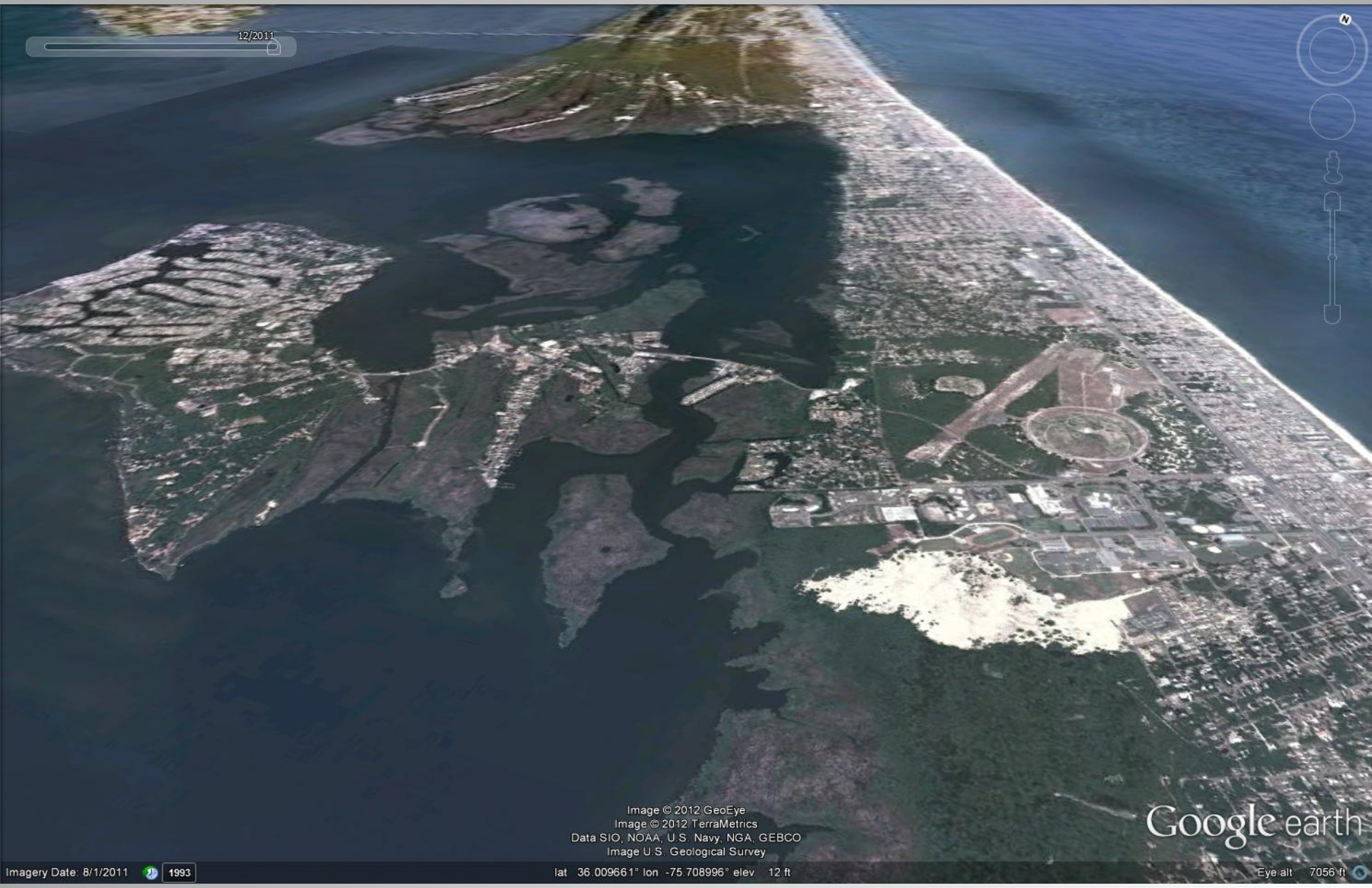
File Edit View Settings ?

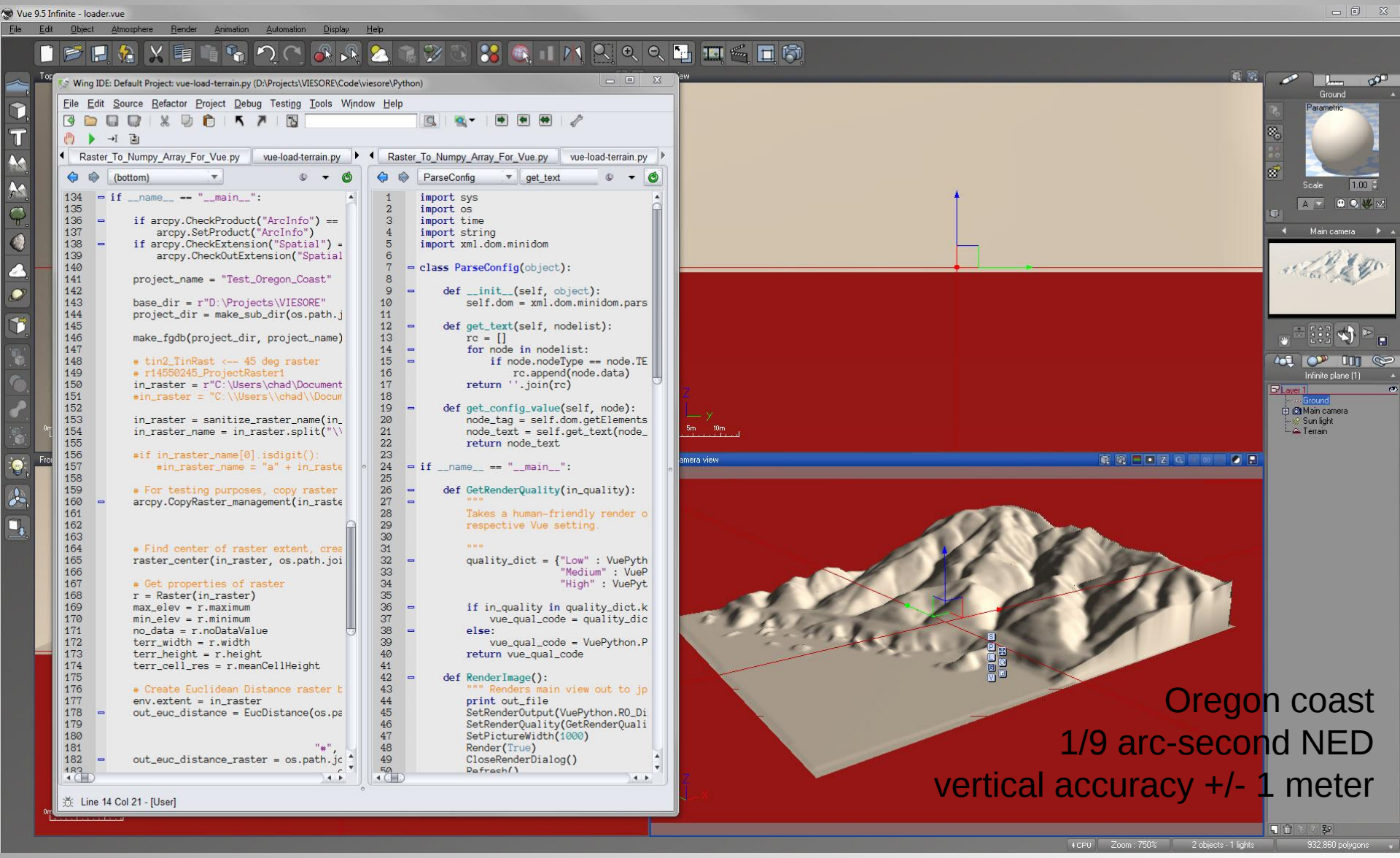
```
0.17626473697193856, 0.17626941489362349, 0.17623338380240974,
0.17409059583006709, 0.16863049866327687, 0.16750434164258635,
0.16739748566334889, 0.16740111929181922, 0.16987398649502636,
0.17393349595966948, 0.17544583675752892, 0.17615749376759246,
0.17383971373914939, 0.17292635896557054, 0.17672483642767997,
0.17684790602842734, 0.17700167480100931, 0.17685808528557118,
0.17641055095990291, 0.17847097212205809, 0.17805913111514368,
0.17126984239383883, 0.16745095104134383, 0.170553453281551,
0.17332714779457481, 0.17033727542923977, 0.17335404699227103,
0.17632976494982569, 0.17632972405628922, 0.17573986430096661,
0.17219406392506664, 0.1683560984142195, 0.1689862447155269,
0.17404315324125 78, 0.16992748665290616, 0.16810377121908601,
0.17363451785285575, 0.17632155358204191, 0.17627325062583099,
0.17615064017390727, 0.17621746432905885, 0.17578359830793652,
0.17022927823503048, 0.1737628338905465, 0.17473996366948877,
0.17480815669841315, 0.16974494920947339, 0.16878328452370667,
0.17288344493514379, 0.17500462044249085, 0.17603605402561326,
0.1762807454464714, 0.17627462886028375, 0.17617701155967905,
0.17615177204714111, 0.17417241268222528, 0.16918491202995205,
0.17063286358415145, 0.1674886976503781, 0.16709466155339292,
0.16706031902558893, 0.16704876949532538, 0.16849119739895432,
```

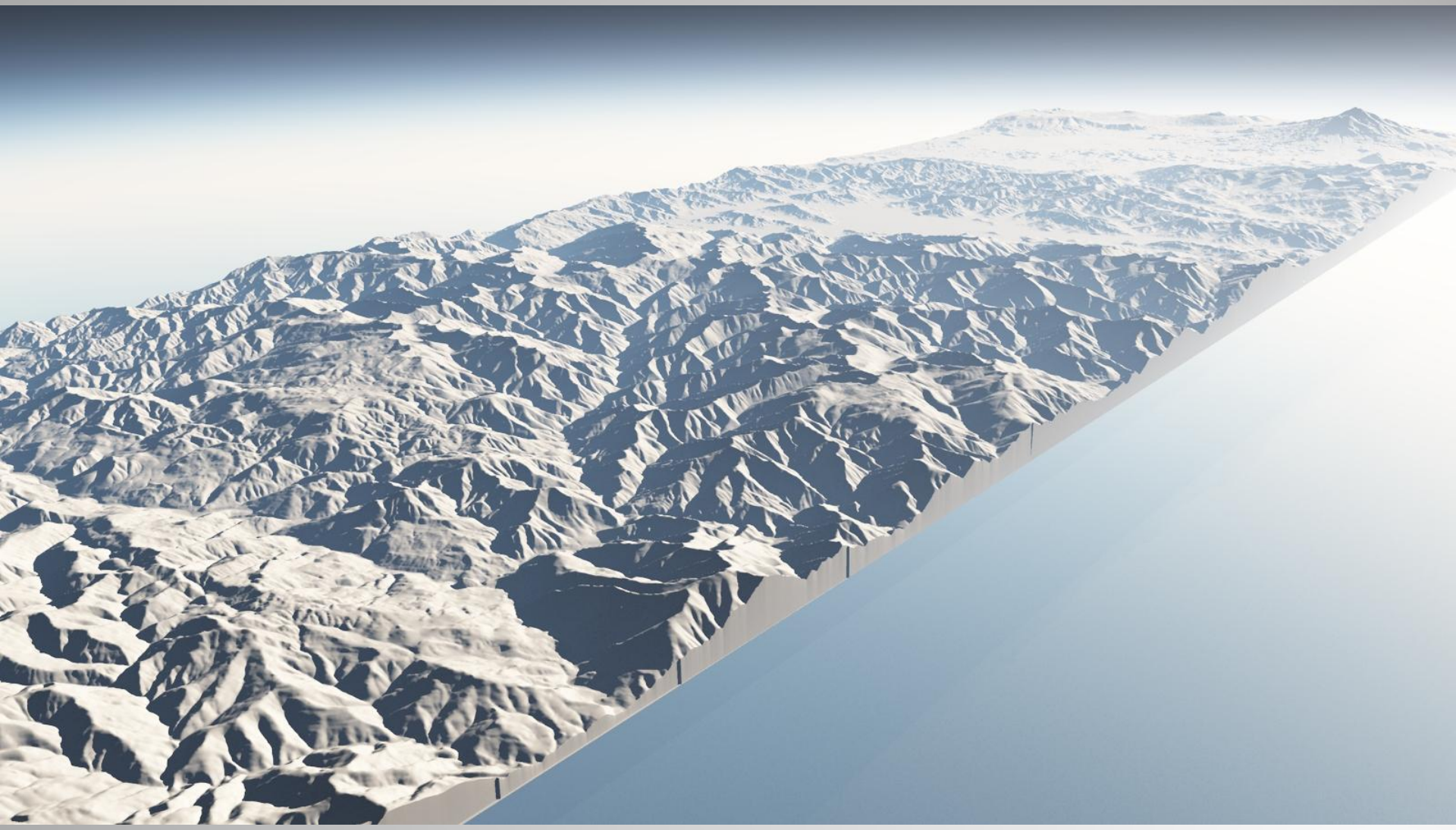
Ln 1 : 2,360 Col 1 Sel 0 36.9 MB ANSI CR+LF INS Default Text

```
<Terrain>
  <TerrainFile>NcLidarGCSShoreClipMinus200_HavCurvedPCS.txt</TerrainFile>
  <TerrainWidth>1920</TerrainWidth>
  <TerrainHeight>2359</TerrainHeight>
  <TerrainXCenter>439242.825415</TerrainXCenter>
  <TerrainYCenter>3985323.55767</TerrainYCenter>
  <TerrainMinElevation>-206.856430054</TerrainMinElevation>
  <TerrainMaxElevation>28.2428207397</TerrainMaxElevation>
  <TerrainCellResolution>6.18222055595</TerrainCellResolution>
  <TerrainNoDataValue>-3.40282346639e+38</TerrainNoDataValue>
</Terrain>
```

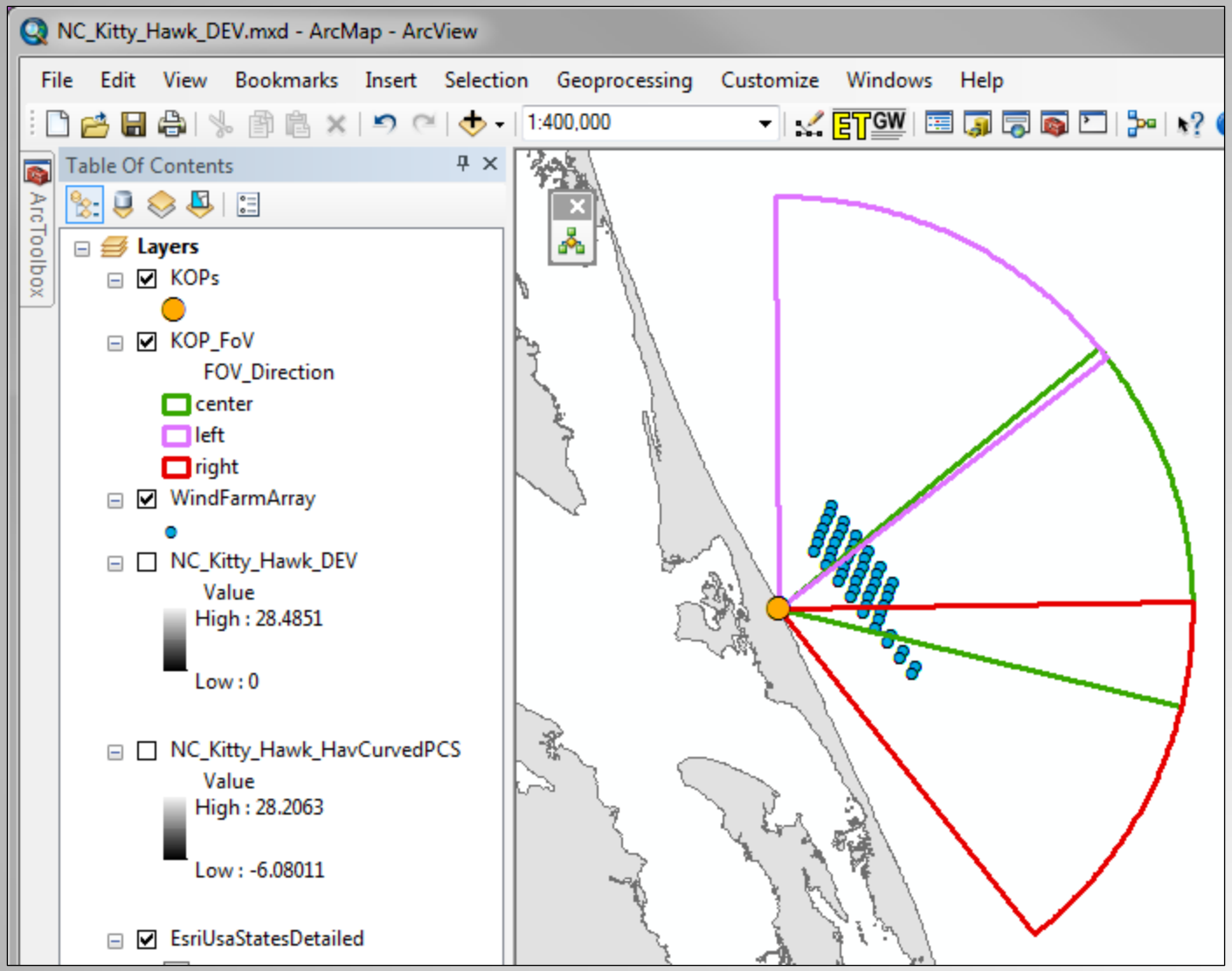

Kitty Hawk, NC
1/9 arc-second NED
vertical accuracy +/- 1 meter

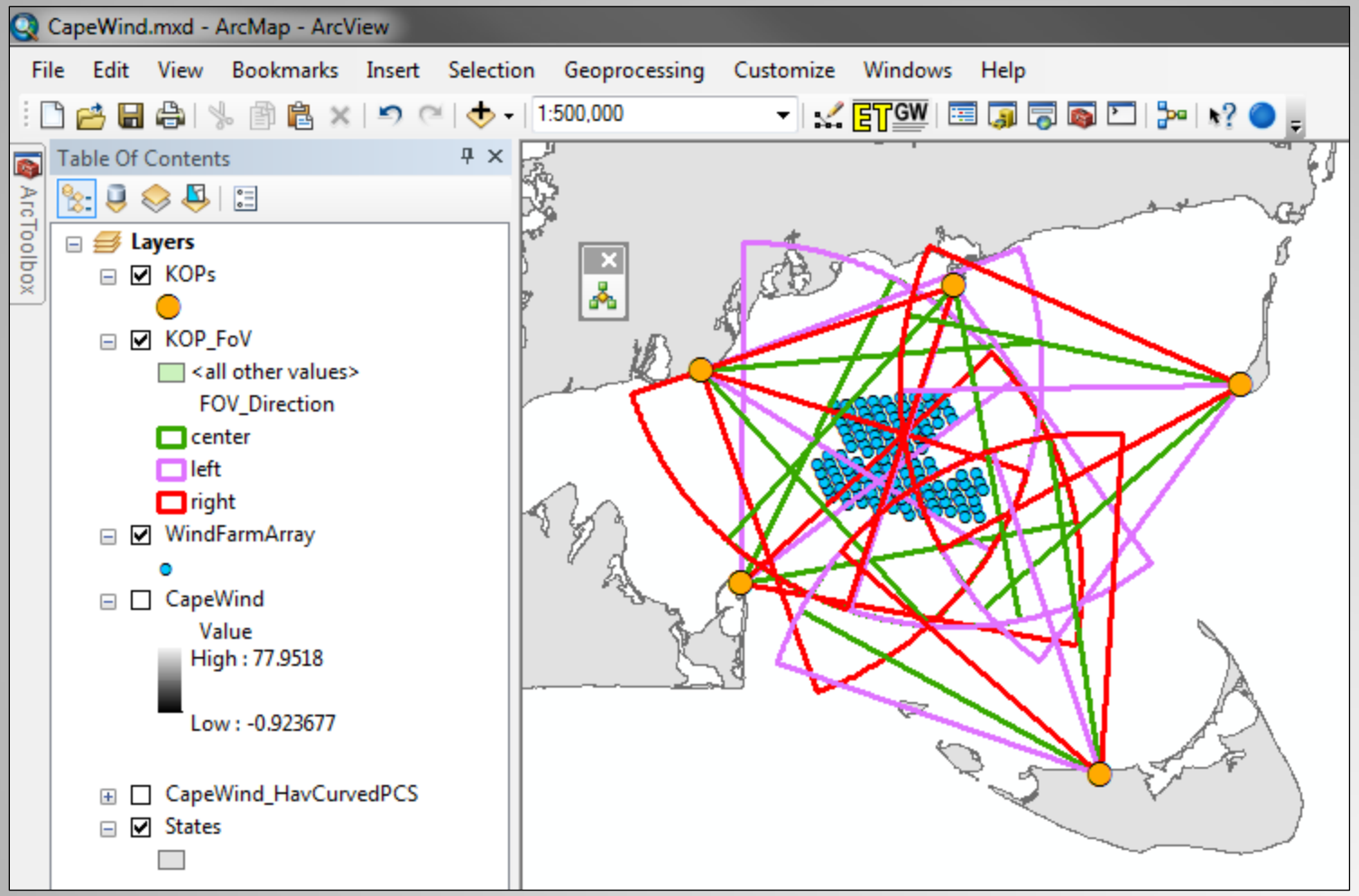

12/2011
Image © 2012 GeoEye
Image © 2012 TerraMetrics
Data SIO, NOAA, U.S. Navy, NGA, GEBCO
Image U.S. Geological Survey
Google earth
Imagery Date: 8/1/2011
1993
lat 36.009661° lon -75.708996° elev 12 ft
Eye alt 7056 ft

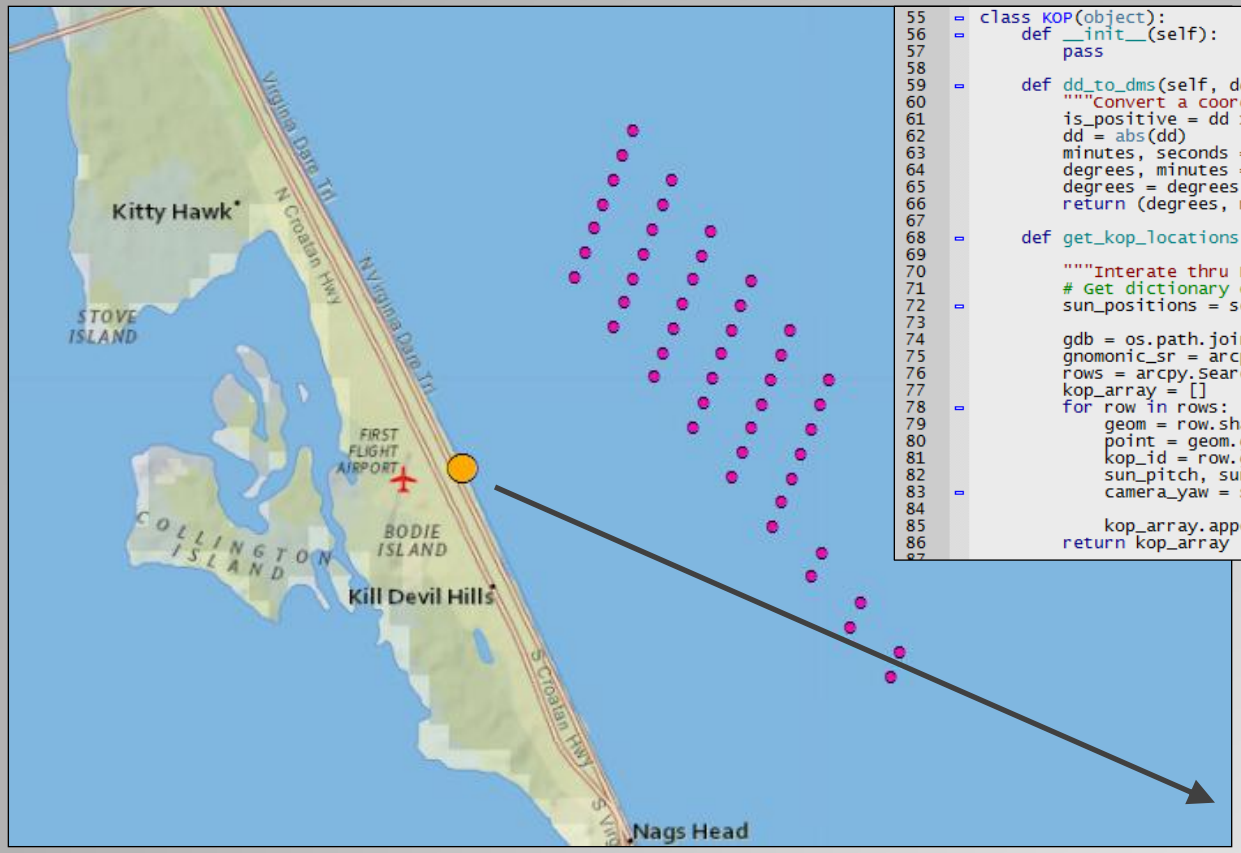

Vue 9.5 Infinite - loader.vue
Wing IDE: Default Project: vue-load-terrain.py (D:\Projects\VIESORE\Code\viesore\Python)
Raster_To_Numpy_Array_For_Vue.py
vue-load-terrain.py
Line 14 Col 21 - [User]
Oregon coast
1/9 arc-second NED
vertical accuracy +/- 1 meter



Key observation points A.K.A., the camera

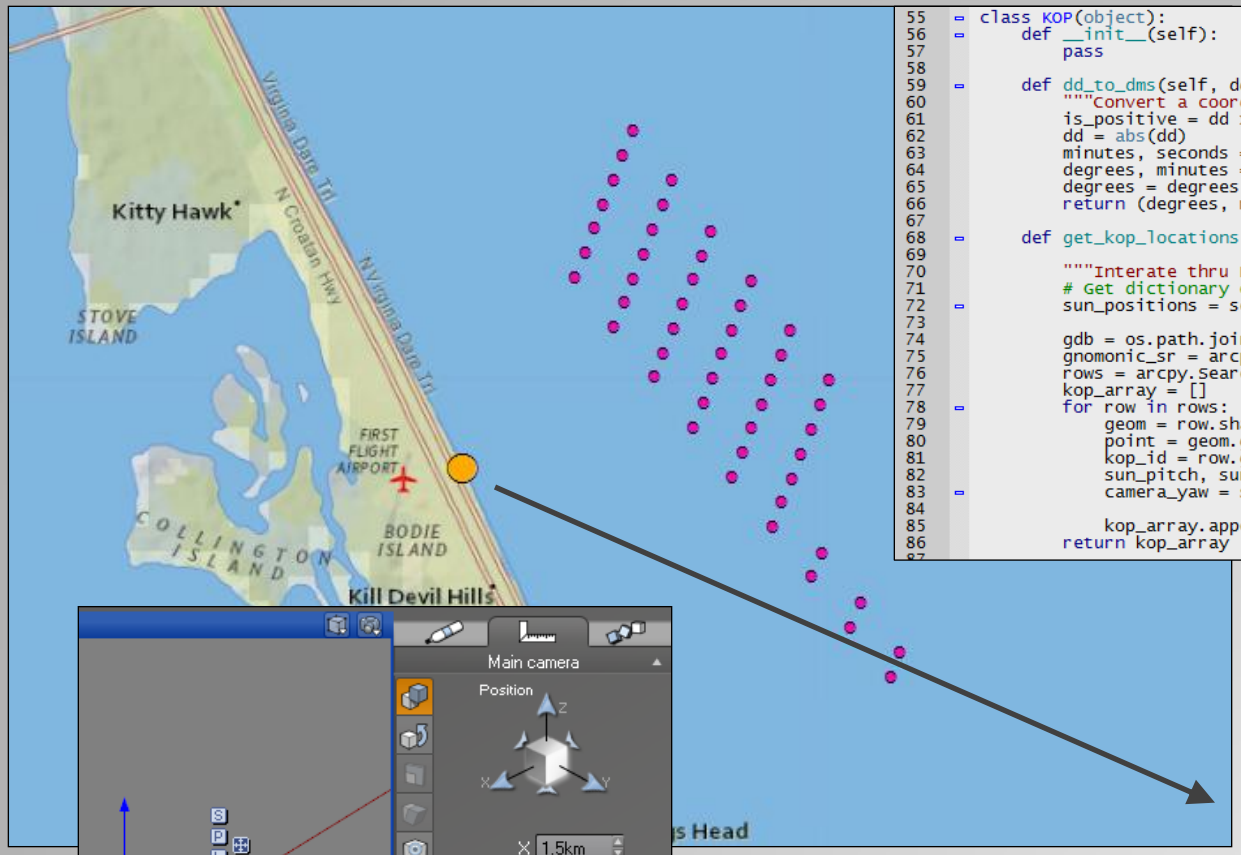

NC_Kitty_Hawk_DEV.mxd - ArcMap - ArcView
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
1:400,000
Table Of Contents
ArcToolbox
Layers
KOPs
KOP_FoV
FOV_Direction
center
left
right
WindFarmArray
NC_Kitty_Hawk_DEV
Value
High : 28.4851
Low : 0
NC_Kitty_Hawk_HavCurvedPCS
Value
High : 28.2063
Low : -6.08011
EsriUsaStatesDetailed


CapeWind.mxd - ArcMap - ArcView
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
1:500,000
ETGW
ArcToolbox
Table Of Contents
Layers
KOPs
KOP_FoV
<all other values>
FOV_Direction
center
left
right
WindFarmArray
CapeWind
Value
High : 77.9518
Low : -0.923677
CapeWind_HavCurvedPCS
States

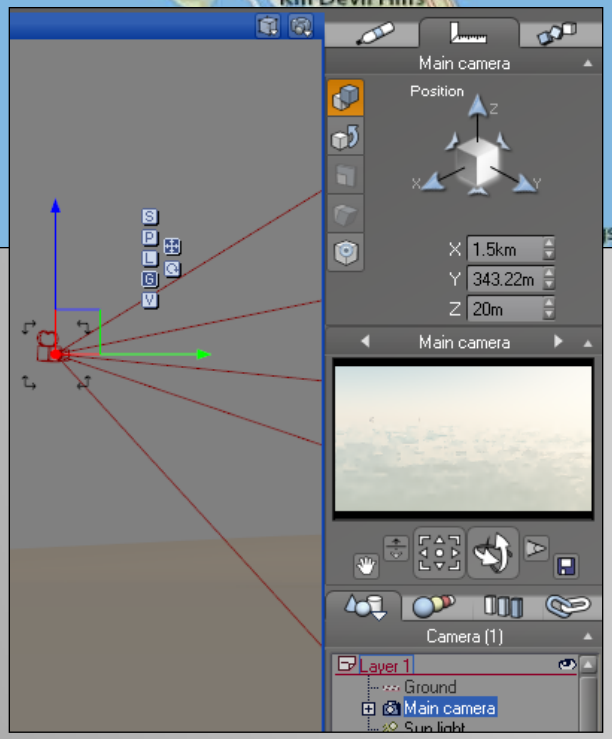



```python
class KOP(object):
    def __init__(self):
        pass

    def dd_to_dms(self, dd):
        """Convert a coordinate in decimal degrees to a tuple of (
        is_positive = dd >= 0
        dd = abs(dd)
        minutes, seconds = divmod(dd * 3600, 60)
        degrees, minutes = divmod(minutes, 60)
        degrees = degrees if is_positive else -degrees
        return (degrees, minutes, seconds)

    def get_kop_locations(self, project_path, project_name, render_
                          render_time, camera_position):
        """Interate thru KOP featureclass and gather KOP array inf
        # Get dictionary of sun positions for KOPs
        sun_positions = self.get_sun_position(project_path, project
                                              render_date, render_
        gdb = os.path.join(project_path, project_name + ".gdb")
        gnomonic_sr = arcpy.Describe(os.path.join(gdb, project_nam
        rows = arcpy.SearchCursor(os.path.join(gdb, "KOPs"), "", g
        kop_array = []
        for row in rows:
            geom = row.shape
            point = geom.getPart()
            kop_id = row.getValue("OBJECTID")
            sun_pitch, sun_yaw, utc_time = sun_positions.get(kop_i
            camera_yaw = self.get_camera_yaw_angle(project_path, p
                                                   kop_id, camera_
            kop_array.append([kop_id, point.Y, point.X, sun_pitch,
        return kop_array
```

```xml
<KOPArray>
  <KOP>
    <KOPID>1</KOPID>
    <VueNorthing>-253.927738287</VueNorthing>
    <VueEasting>2437.09785053</VueEasting>
    <CameraZPosition>3.7</CameraZPosition>
    <FOVPosition>Center</FOVPosition>
    <CameraYaw>102.334980719</CameraYaw>
    <CameraPitch>93</CameraPitch>
    <SunPitch>129.848225203</SunPitch>
    <SunYaw>116.553015413</SunYaw>
  </KOP>
</KOPArray>
```



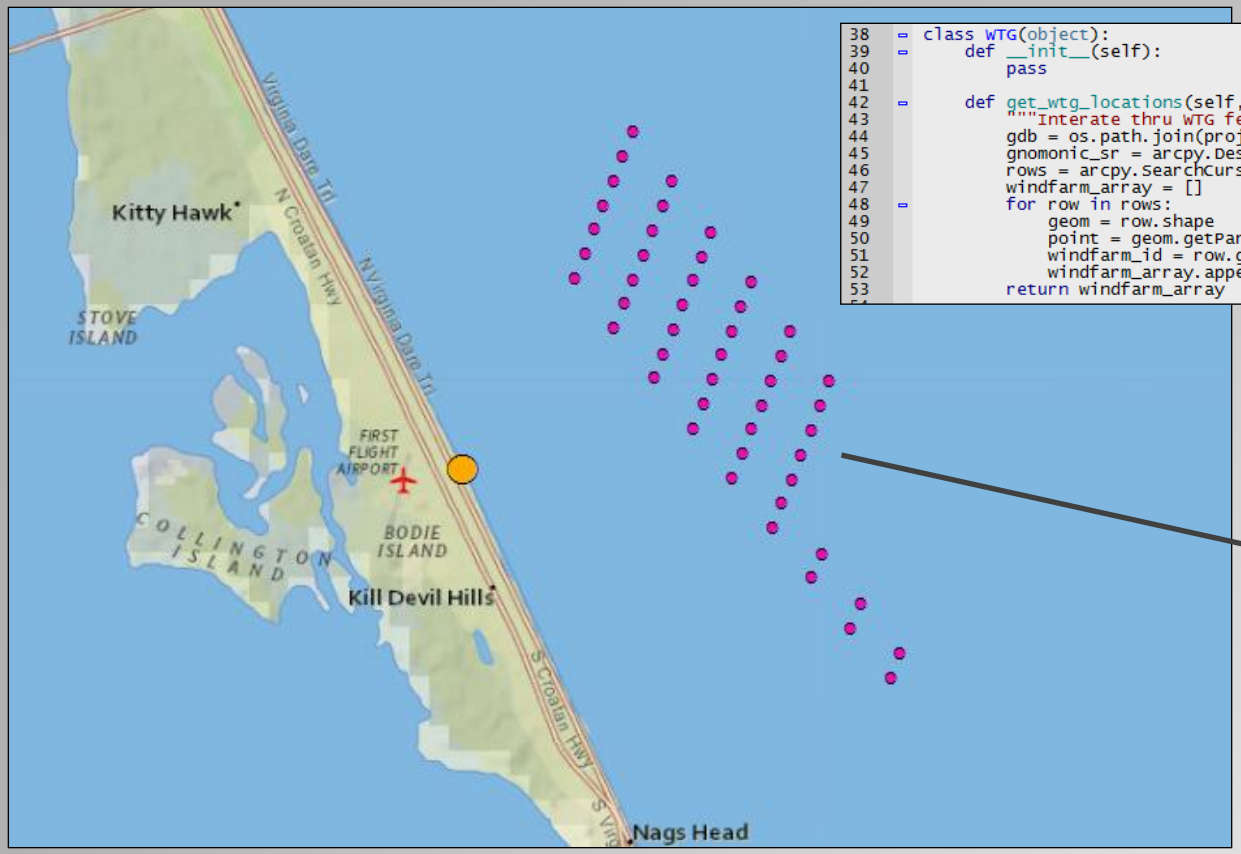

```python
class KOP(object):
    def __init__(self):
        pass

    def dd_to_dms(self, dd):
        """Convert a coordinate in decimal degrees to a tuple of (
        is_positive = dd >= 0
        dd = abs(dd)
        minutes, seconds = divmod(dd * 3600, 60)
        degrees, minutes = divmod(minutes, 60)
        degrees = degrees if is_positive else -degrees
        return (degrees, minutes, seconds)

    def get_kop_locations(self, project_path, project_name, render_
                          render_time, camera_position):
        """Interate thru KOP featureclass and gather KOP array inf
        # Get dictionary of sun positions for KOPs
        sun_positions = self.get_sun_position(project_path, project
                                              render_date, render_
        gdb = os.path.join(project_path, project_name + ".gdb")
        gnomonic_sr = arcpy.Describe(os.path.join(gdb, project_nam
        rows = arcpy.SearchCursor(os.path.join(gdb, "KOPs"), "", g
        kop_array = []
        for row in rows:
            geom = row.shape
            point = geom.getPart()
            kop_id = row.getValue("OBJECTID")
            sun_pitch, sun_yaw, utc_time = sun_positions.get(kop_i
            camera_yaw = self.get_camera_yaw_angle(project_path, p
                                                   kop_id, camera_
            kop_array.append([kop_id, point.Y, point.X, sun_pitch,
        return kop_array
```




```xml
<KOPArray>
  <KOP>
    <KOPID>1</KOPID>
    <VueNorthing>-253.927738287</VueNorthing>
    <VueEasting>2437.09785053</VueEasting>
    <CameraZPosition>3.7</CameraZPosition>
    <FOVPosition>Center</FOVPosition>
    <CameraYaw>102.334980719</CameraYaw>
    <CameraPitch>93</CameraPitch>
    <SunPitch>129.848225203</SunPitch>
    <SunYaw>116.553015413</SunYaw>
  </KOP>
</KOPArray>
```

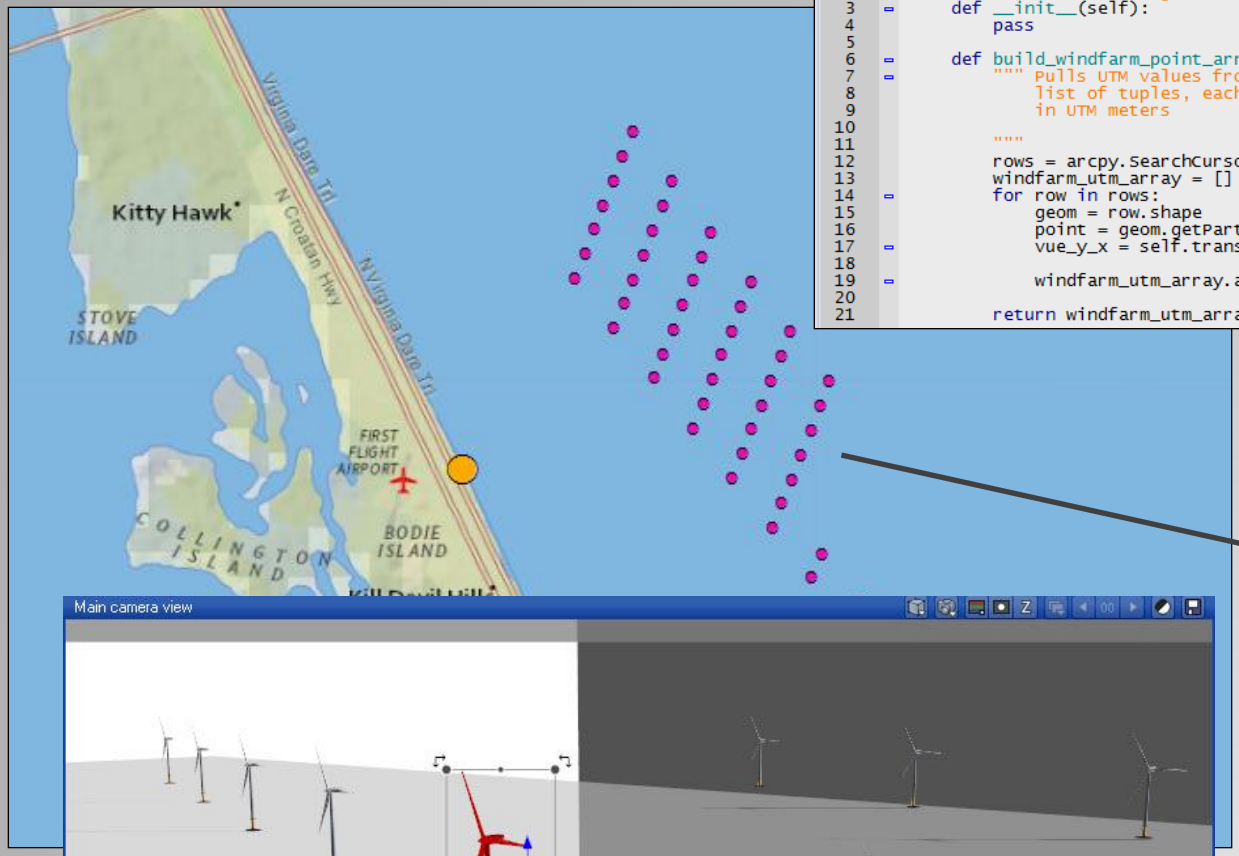
```python
select_camera = SelectByName("Main camera")
camera = GetSelectedObjectByIndex(0)
for k, v in kop_dict.iteritems():
    camera.SetPosition(float(v[1]), float(v[0]), float(v[2]))
    Refresh()
    camera.SetRotationAngles(float(v[5]), 0, float(v[4]), true)
    Refresh()
    DeselectAll()
```

WTG placement



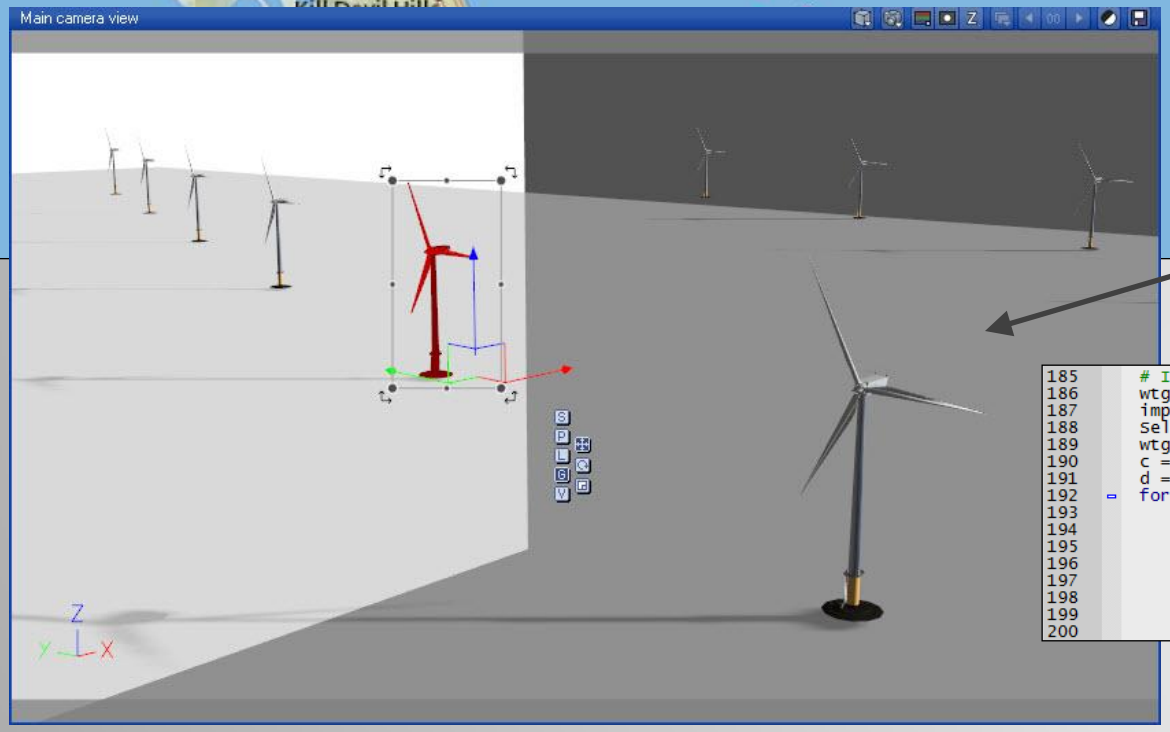

```python
class WTG(object):
    def __init__(self):
        pass

    def get_wtg_locations(self, project_path, project_name):
        """Interate thru WTG featureclass and gather WTG array info"""
        gdb = os.path.join(project_path, project_name + ".gdb")
        gnomonic_sr = arcpy.Describe(os.path.join(gdb, project_name + "
        rows = arcpy.SearchCursor(os.path.join(gdb, "WindFarmArray"), "
        windfarm_array = []
        for row in rows:
            geom = row.shape
            point = geom.getPart()
            windfarm_id = row.getValue("OBJECTID")
            windfarm_array.append([windfarm_id, point.Y, point.X])
        return windfarm_array
```

```xml
<WTGArray>
  <WTG>
    <WTGID>1</WTGID>
    <VueNorthing>6068.67431977</VueNorthing>
    <VueEasting>5592.75417541</VueEasting>
  </WTG>
  <WTG>
    <WTGID>2</WTGID>
    <VueNorthing>5134.24113799</VueNorthing>
    <VueEasting>6330.87555269</VueEasting>
  </WTG>
  <WTG>
    <WTGID>3</WTGID>
    <VueNorthing>4199.69976142</VueNorthing>
    <VueEasting>7069.29084193</VueEasting>
  </WTG>
  <WTG>
```




```python
class WTG:
    """Class for creating and working with WTG and WTG array data"""
    def __init__(self):
        pass

    def build_windfarm_point_array(self, gdb, x_center, y_center):
        """ Pulls UTM values from WindfarmArray FC and returns a
            list of tuples, each tuple representing a WTG location
            in UTM meters

        """
        rows = arcpy.SearchCursor(r"D:\NC_GCS.gdb\WindFarmArrayNC")
        windfarm_utm_array = []
        for row in rows:
            geom = row.shape
            point = geom.getPart()
            vue_y_x = self.transform_utm_to_vue_coords(point.Y, point.X,
                                                       x_center, y_center)
            windfarm_utm_array.append((point.Y, point.X,
                                       vue_y_x[0], vue_y_x[1]))
        return windfarm_utm_array
```

```
<WTGArray>
  <WTG>
    <WTGID>1</WTGID>
    <VueNorthing>6068.67431977</VueNorthing>
    <VueEasting>5592.75417541</VueEasting>
  </WTG>
  <WTG>
    <WTGID>2</WTGID>
    <VueNorthing>5134.24113799</VueNorthing>
    <VueEasting>6330.87555269</VueEasting>
  </WTG>
  <WTG>
    <WTGID>3</WTGID>
    <VueNorthing>4199.69976142</VueNorthing>
    <VueEasting>7069.29084193</VueEasting>
  </WTG>
  <WTG>
```



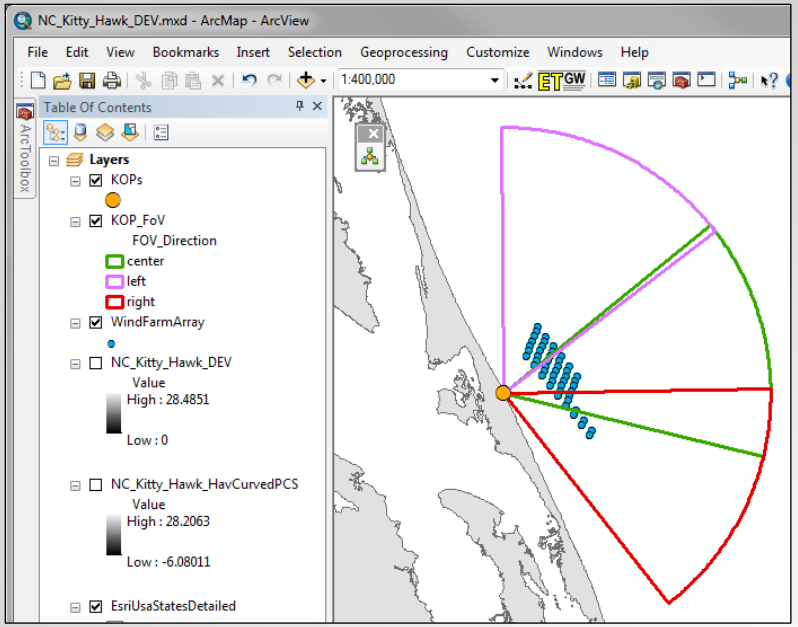

```
# Import the WTG model
wtg = os.path.join(viesore_root_dir, "Templates\WTGs\siemensSW
import_wtg = ImportObject(wtg, 0, true, -1)
Select(import_wtg)
wtg_height = get_object_size(import_wtg)[2]
c = Copy()
d = Delete()
for k, v in wtg_dict.iteritems():
    # TODO: Set name using SetName() method, use FID of the WTG
    # Paste the WTG
    p = Paste()
    # Move the WTG
    curr_wtg = GetSelectedObjectByIndex(0)
    curr_wtg.Move(float(v[1]), float(v[0]), int(wtg_height)/2)
    Refresh()
    DeselectAll()
```

Current results

Real-world GIS data brought into Vue

Photorealistic renderings

User-friendly interface



Current results

Real-world GIS data brought into Vue

Photorealistic renderings

User-friendly interface

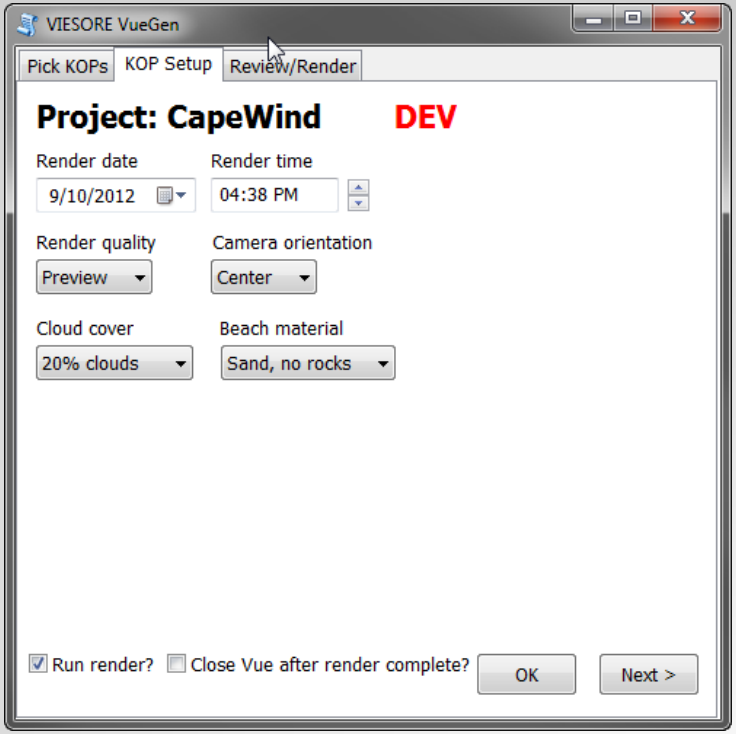
Current results

Real-world GIS data brought into Vue

Photorealistic renderings

User-friendly interface

- still working on that...



Renders


The Center for Advanced Spatial Technologies
CAST
University of Arkansas, Fayetteville
U of A


Argonne
NATIONAL
LABORATORY

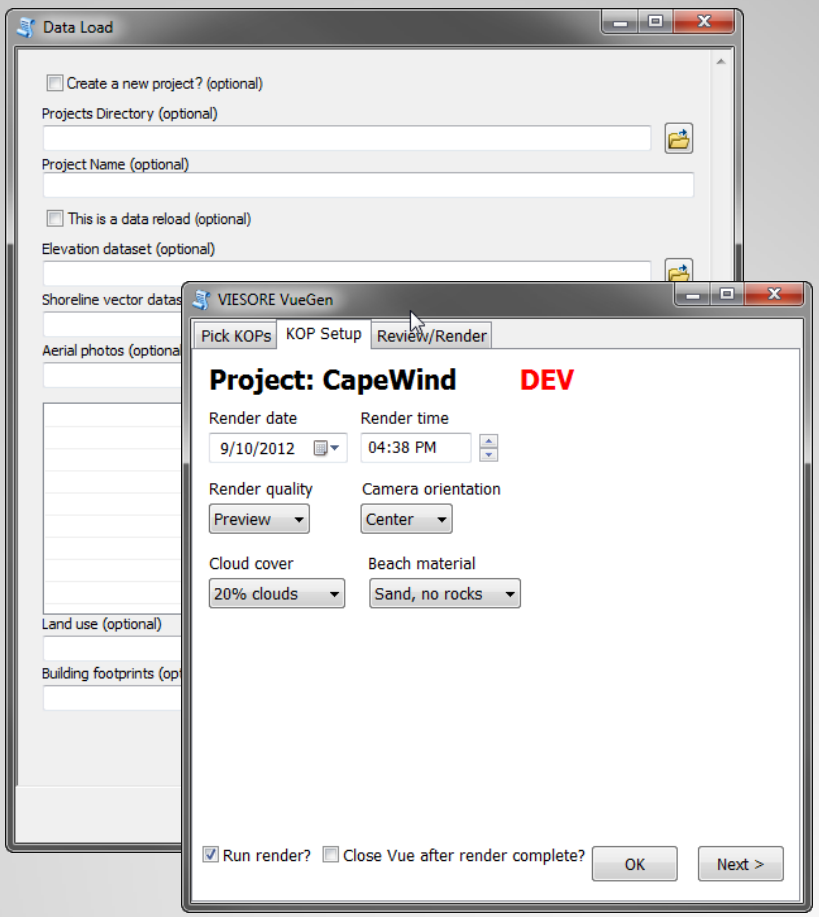

UNIVERSITY OF
ARKANSAS

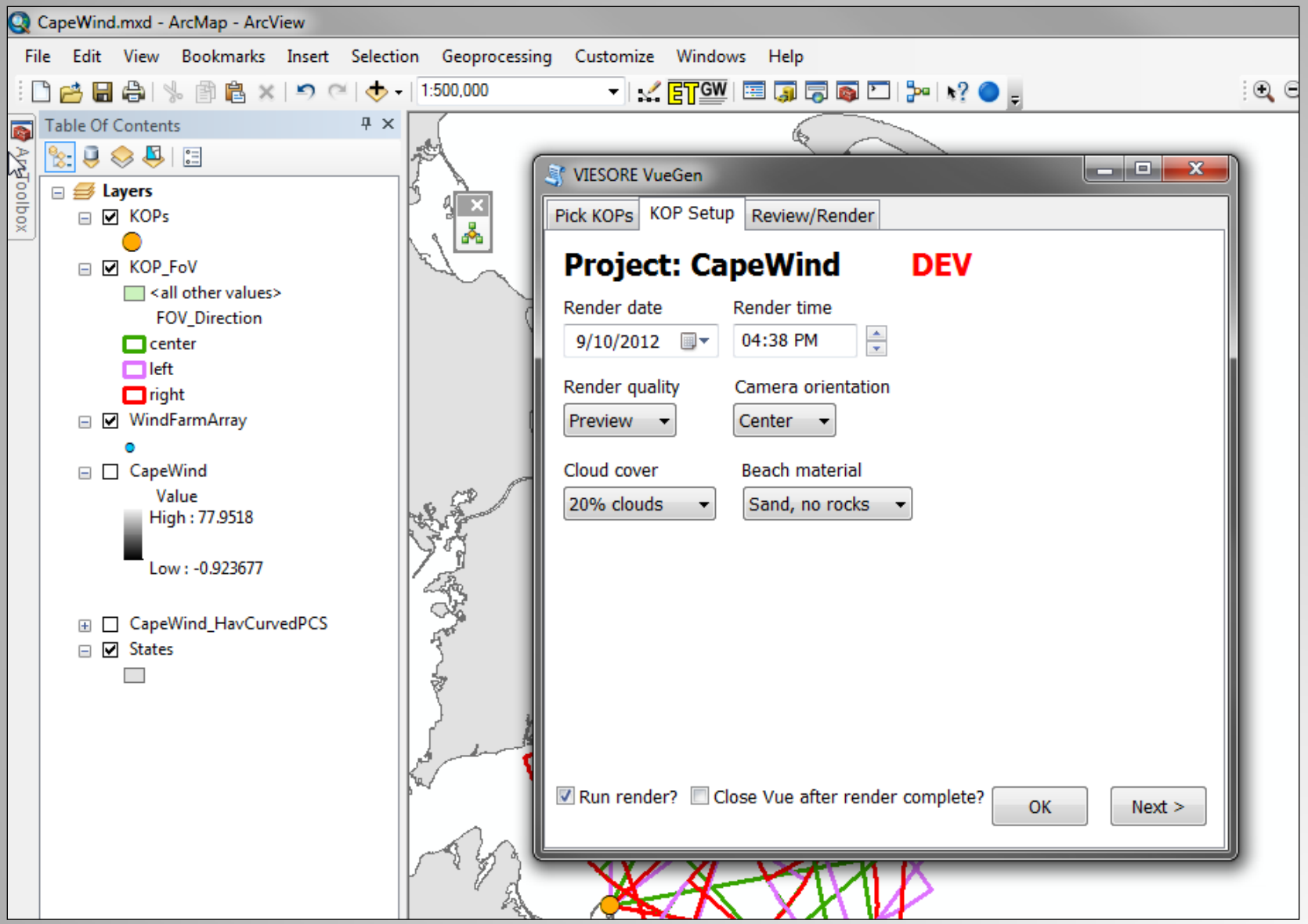
Issues tackled

- Vue data formats
- Data translations galore
- ***Using real-world data in software that wasn't really made for it***
- Coarse-grained Vue Python API makes you come up with creative workarounds

Still to do

- User interface to Vue
  - Toolbox designs and linkage to Python scripts
    - ArcToolbox tools
    - wxPython
- Report generation
  - Python
- Help/documentation




CapeWind.mxd - ArcMap - ArcView
File Edit View Bookmarks Insert Selection Geoprocessing Customize Windows Help
1:500,000
Table Of Contents
ArcToolbox
Layers
KOPs
KOP_FoV
<all other values>
FOV_Direction
center
left
right
WindFarmArray
CapeWind
Value
High : 77.9518
Low : -0.923677
CapeWind_HavCurvedPCS
States
VIESORE VueGen
Pick KOPs
KOP Setup
Review/Render
Project: CapeWind
DEV
Render date
9/10/2012
Render time
04:38 PM
Render quality
Preview
Camera orientation
Center
Cloud cover
20% clouds
Beach material
Sand, no rocks
Run render?
Close Vue after render complete?
OK
Next >

Questions?

chad@cast.uark.edu

http://cast.uark.edu