Slow and Steady Simulation – A sample paradigm to slowly increase the complexity and fidelity of your EOIR simulations

Keywords: EOIR, Image Simulation, Image Processing, Sensor, Modeling and Simulation,

EOIR sensor image, simulation tips and tricks, workflow, workarounds

Executive Summary

This write-up describes a step-by-step process for creating an EOIR scenario, starting with basic fast parameters and then moving into more complex modeling. It also includes many common workarounds, tips for speeding up your workflow, rendering time information, and plenty of screenshots.

Introduction

If you have used heavy duty modeling and simulation tools that take hours or even days to run, I feel your pain. STK's EOIR is one of those heavy duty modeling and simulation tools for which even a relatively small image can be computationally expensive to render. Besides throwing more hardware at the problem (faster processor and more memory), there are a few things you can do on the software side to make the process more intuitive and save you time overall to make sure you end up with the results you want.

Here are the three steps I use with STK and EOIR to get the imaging system data that I'm looking for:

- Step 1: Sensor
- Step 2: Target
- Step 3: Bells and Whistles

That's it. I'll show you an example of this in action as well as all of the workarounds and tricks that I use to save myself time throughout the process. I chose a relatively simple scenario of an aircraft imaging the ground right here in the Exton area, starting from scratch and eventually building up to the point where I'll have material maps, custom 3D models, and an atmospheric model.

The inspiration for this FAQ began when we were preparing for a virtual EOIR training. As I was getting ready for this, I happened to present EOIR to a group of visitors here in Exton. We were starting from scratch to build scenarios, and I realized that we needed to slow things down and break up the independent sets of parameters to emphasize each one along the way. I'll go through the three steps we mentioned with screenshots and using many default details so that you can easily replicate this process and apply it to your own domain-specific situations. Rendering can take quite a bit of time, and I'll include time estimates as often as possible as well.

I'm also assuming you're already pretty familiar with STK. If you're not and would like a little more background on STK, please check out AGI Technical Resources to find out more and get a better understanding of the STK fundamentals. Also, for a more general EOIR-specific background, you can check out the EOIR Overview. This is an applied process though, so don't feel bad about skimming through it as an outline and looking up specific details on an as-needed basis.

Step 1 – Setting up your imaging platform and basic sensor properties

As I mentioned earlier, we're starting from scratch. Here's the brand new scenario that we'll be filling out. We'll create a default place (our AGI headquarters in Exton, PA) and an aircraft flying a basic route around it with a sensor targeting the place.

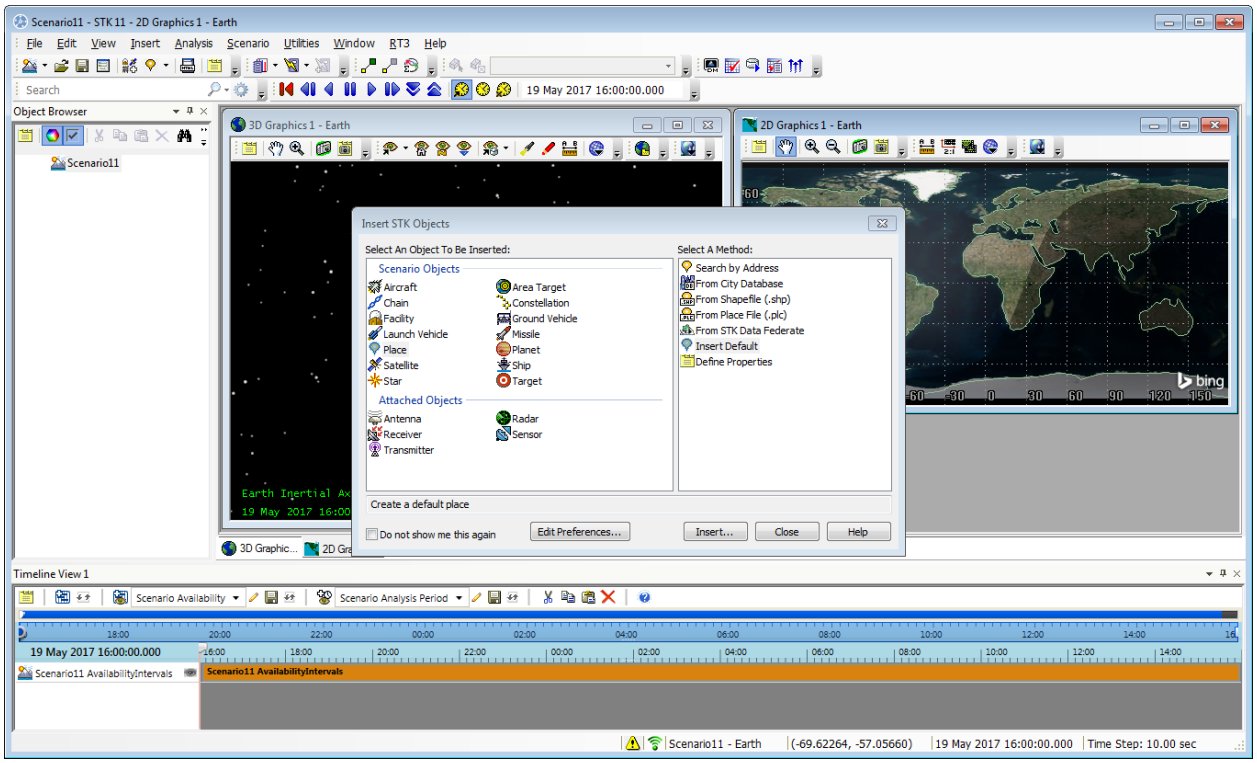

Figure 1 – Default STK window with the Insert STK Objects wizard

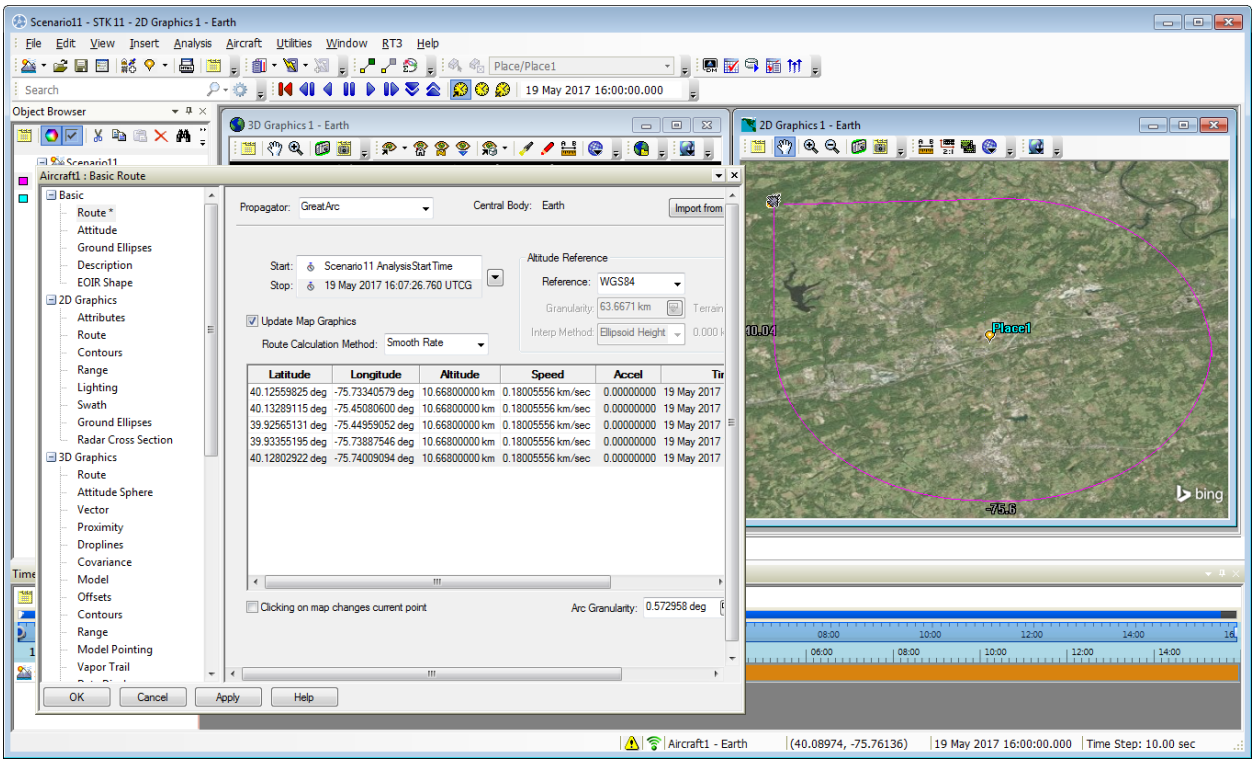

Figure 2 – Adding the default place and an aircraft circling it

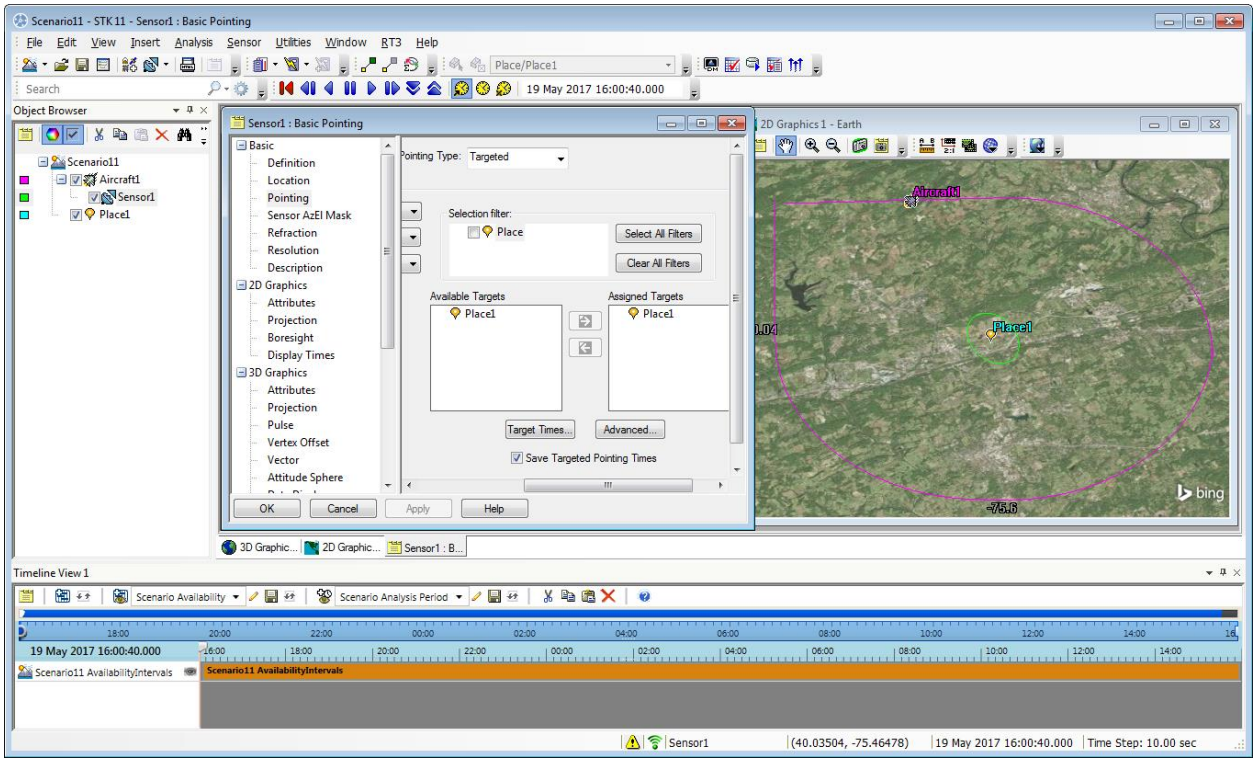

Figure 3 – Adding a 5-degree conic sensor to the aircraft pointed at the place

Now that we have the basic STK scenario set up, let's make that sensor an EOIR sensor and do a sample rendering.

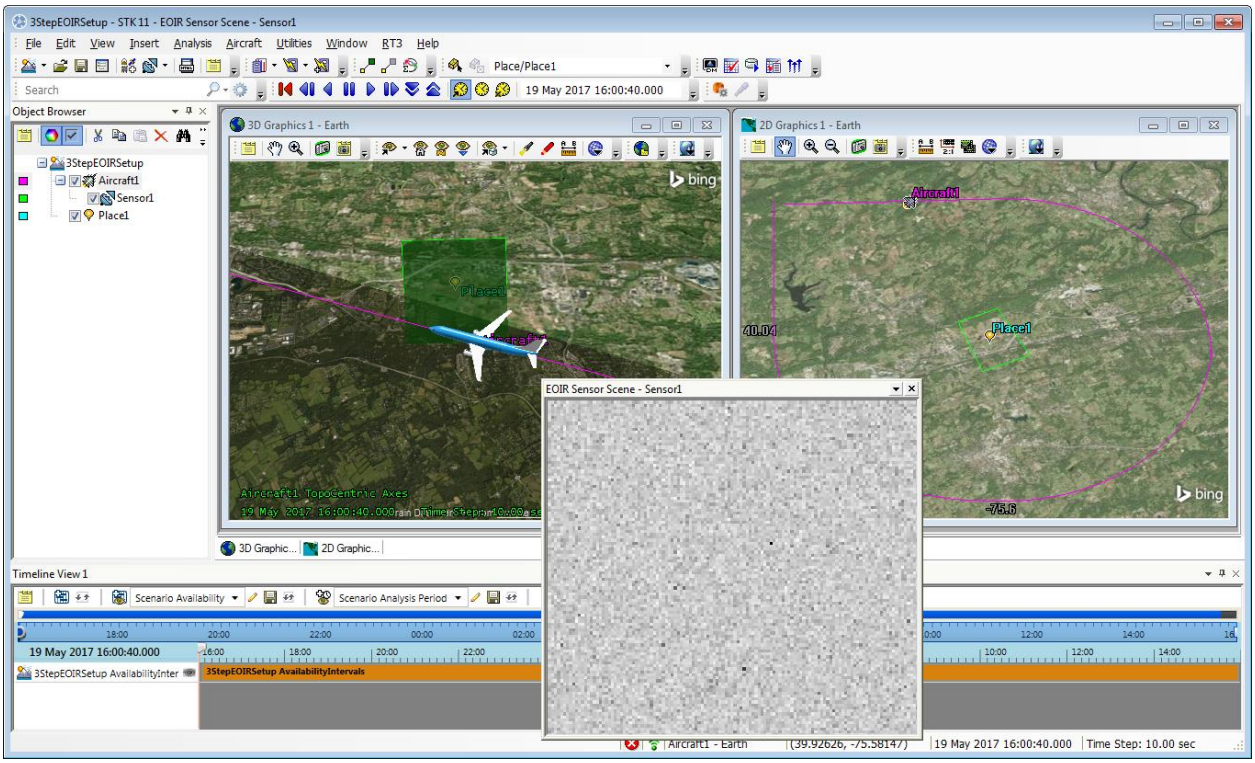

Figure 4 – First look at the low-resolution earth material map simulation

The scene looks like all static, but if we change the details to fine, we'll see the higher resolution earth material map to know that at least we're seeing the ground.

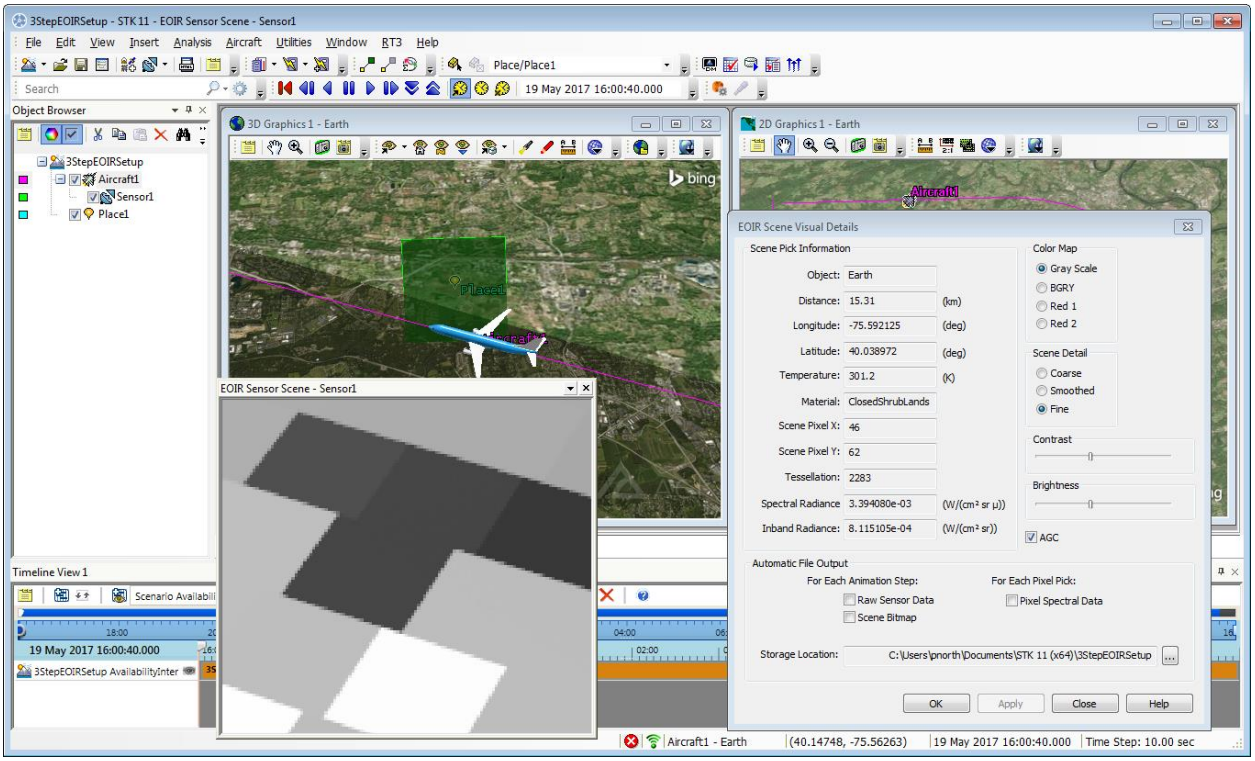

Figure 5 – EOIR simulation of earth with the default high-resolution material map

Nothing too interesting here yet, but we can see large rectangular patches in the EOIR Sensor Scene. These rectangles are the resolution of the default EOIR material map; it's not very impressive at this field-of-view. However, if we zoom out 10x, we see a much larger portion of the earth and are able to see how the default resolution of the EOIR material map is more appropriate at the global scale.

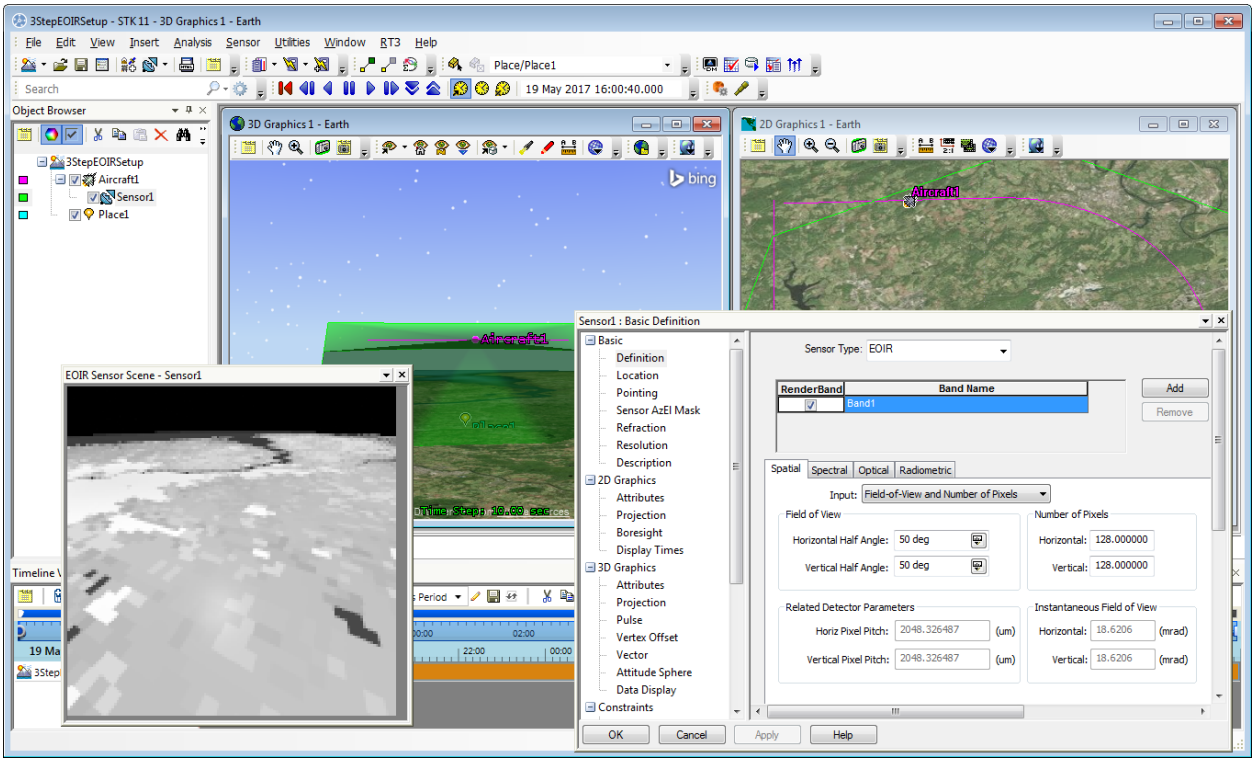

Figure 6 – Larger field-of-view simulation showing the horizon view

We'll be using a custom reflectance map and 3D target models later, but the focus at this step is to make sure our platform and sensor are set up in general. Rendering these images with all of the default values for a 128-by-128 pixel EOIR image takes less than a minute for either the 5-degree or 50-degree half angles, although the larger field-of-view does take longer to render. We can next go into all of the tabs of the senor's band being simulated, and we'll set up the correct parameters to see their effect.

This stage is about as far as I'd like to push the simulations, to keep their rendering time down to a reasonable level. Furthermore, if your sensor has many more pixels, you can also choose to render a subset of those pixels and increase the number of pixels later on in step 3. You can do this by either (a) taking a smaller central portion of the focal plane with fewer detectors and a smaller field-of-view or (b) intentionally down-sample the detectors by keeping the same field-of-view but decreasing the number of detectors. These each have different consequences and really depend on if you would rather see full-resolution details of your target (for this I would choose a) or full content of the eventual image (for this I would choose b).

You don't necessarily need all of the sensor parameters finalized, but at least have the major parameters filled in to start getting an idea of if the early simulated data that is faster to generate is along the lines of what you are looking for.

Another option to speed things up for imagery is to change the sensor's output level from Sensor Output to Radiometric Input. This will render out the entrance aperture radiance and not take the time to apply any sensor-level effects. This will also be at a higher spatial resolution as well because EOIR oversamples each detector pixel four times along each dimension, and the Radiometric Input will be at this 4x oversampled level.

Step 2 – Setting up your target parameters

Next I'd like to set up a basic target that we're looking at. For this particular scenario, I'm thinking of a car on the street that I'm interested in. Because EOIR has a separate terrain model than basic STK, it will not allow ground or water vehicles. If a surface vehicle in this scenario were following the STK terrain, it could appear that it is floating above or maybe even worse going underneath the EOIR WGS-84 surface. As a work-around, I use aircraft, explicitly set their elevation, and turn off the terrain.

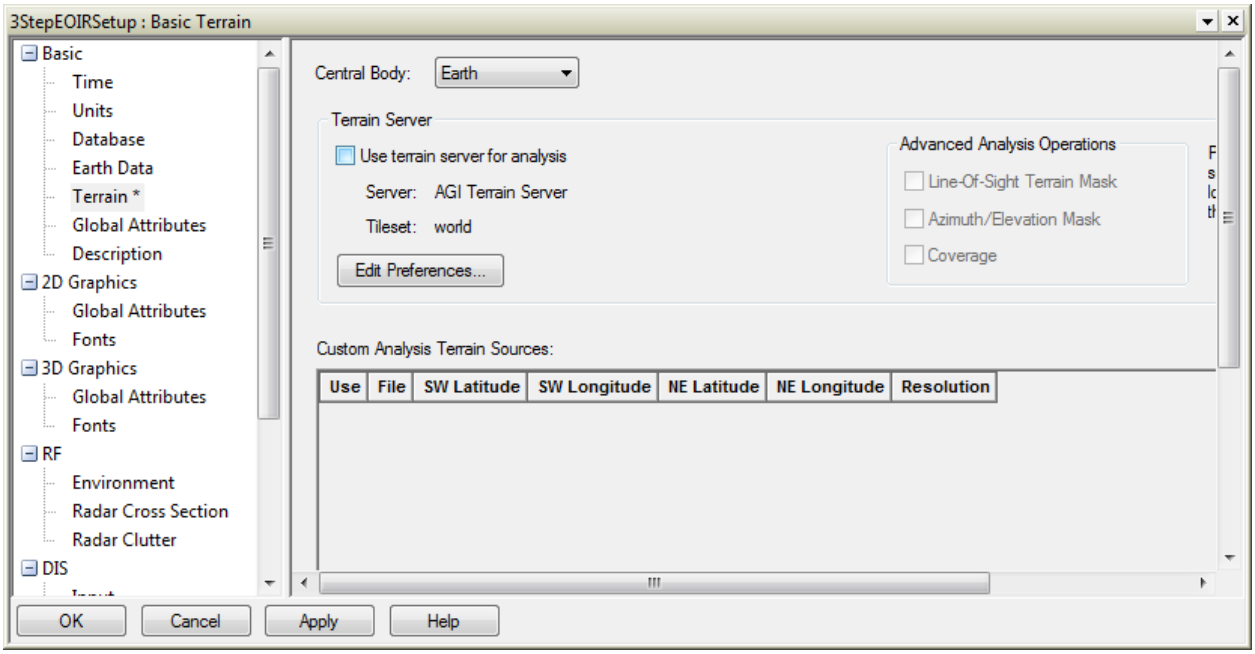

Figure 7 – Turning off the terrain for the scenario

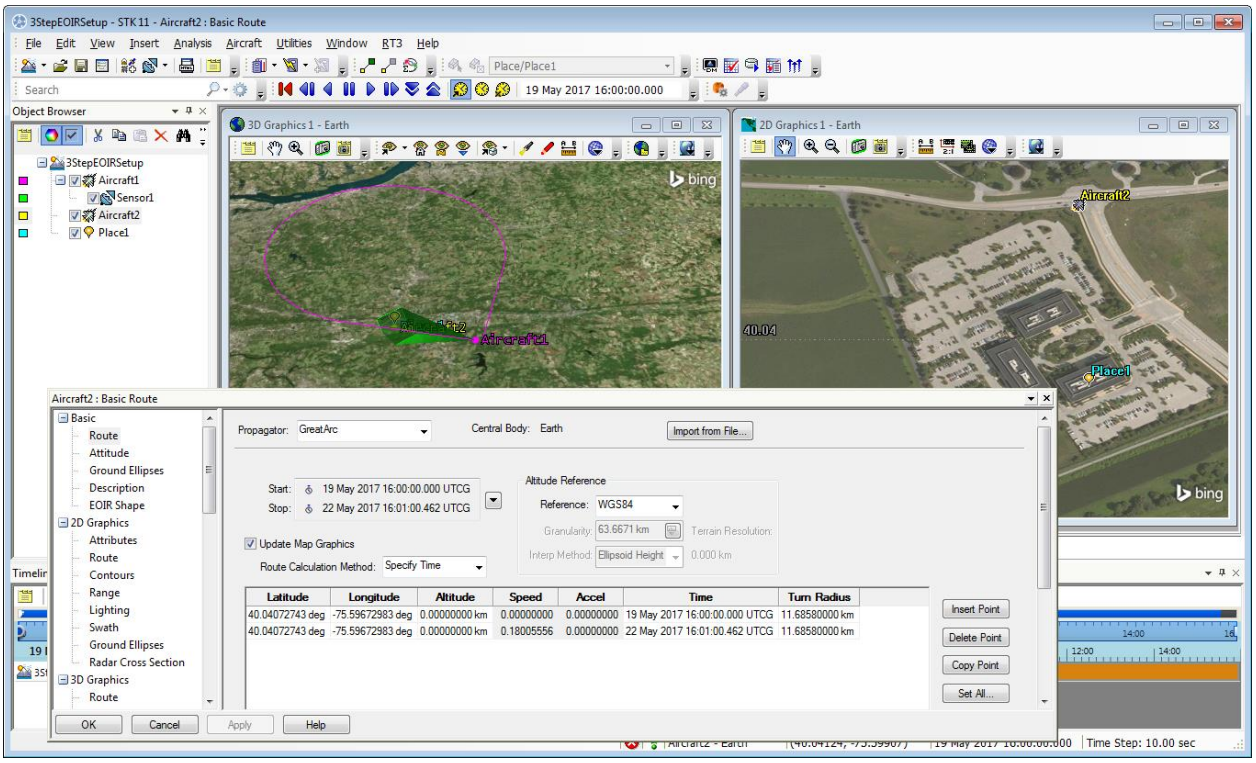

Figure 8 – Adding a stationary EOIR target on the ground as an aircraft

Now that I have added this target, I also need to add it to the EOIR configuration as well.

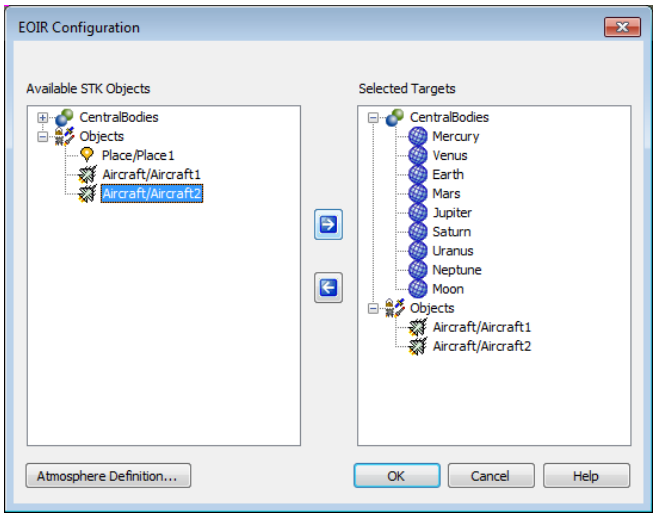

Figure 9 – EOIR configuration window

However, after adding the target to the scenario, it isn't immediately visible.

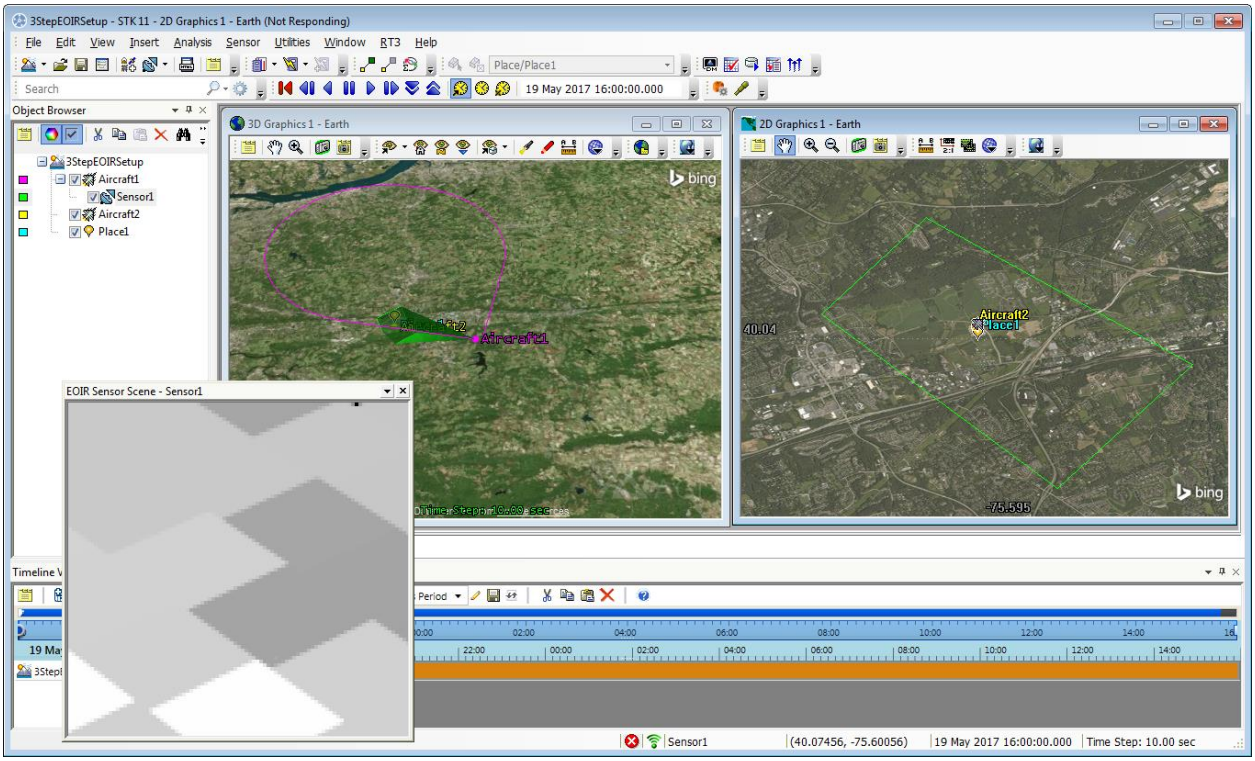

Figure 10 – EOIR simulation with the target in the image but not visible

This is because the default target parameters for EOIR are a 1-meter radius gray sphere that is 50% reflective at about room temperature. There are a few ways we can make this target more visible, either by increasing the size so it's large enough to see or increasing the temperature so it's bright enough to see.

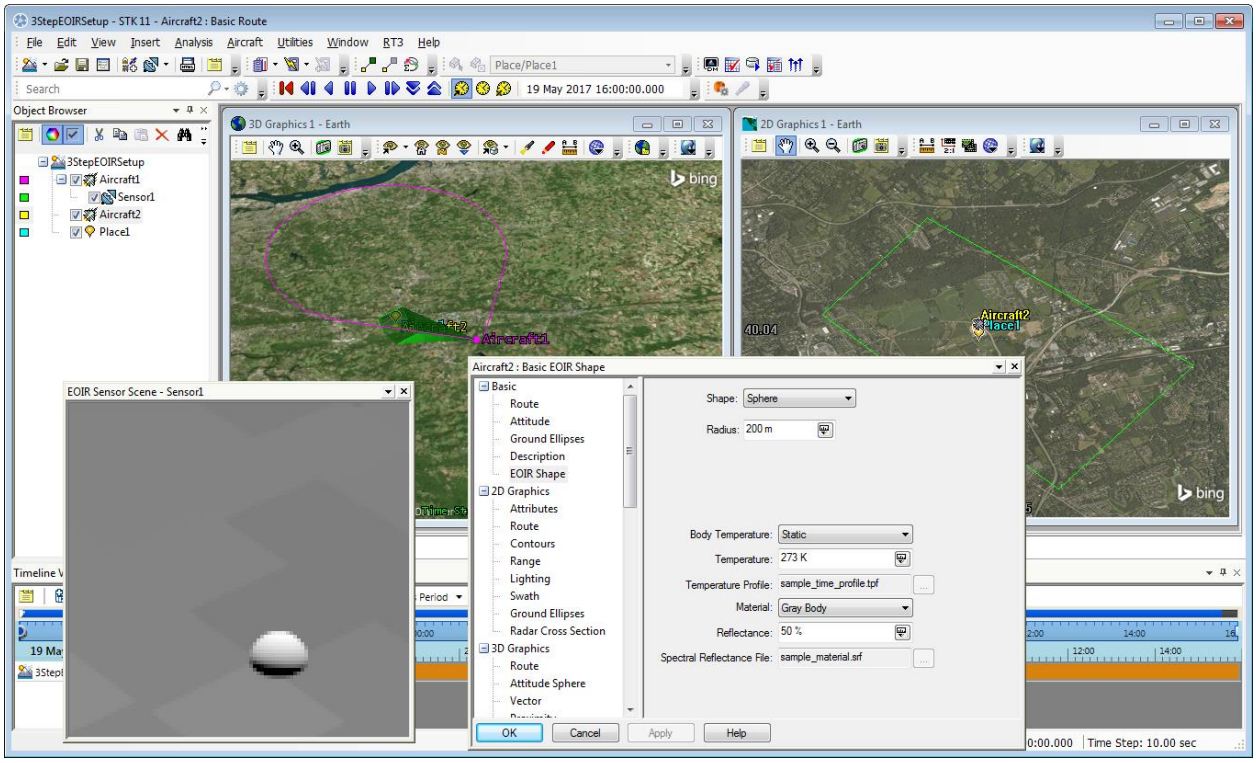

Figure 11 – Very large, visible target

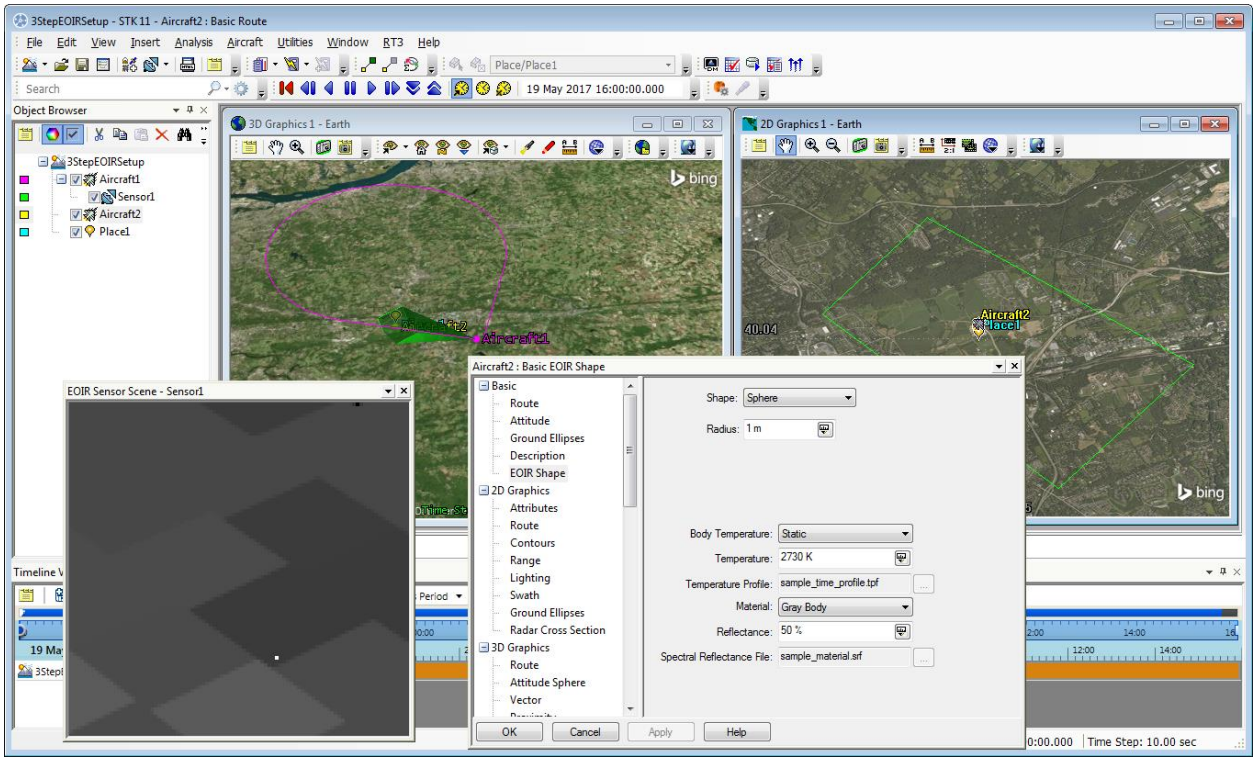

Figure 12 – Very hot, glowing visible target

Now we know that our sensor is set up in general, and we can see where our target of interest is in the scene. We can start going to the next level by setting up more appropriate size, temperature, and material properties, but I would recommend keeping the shape as a sphere initially so that the rendering time stays relatively fast. Both the larger target and hotter target images render slightly slower than without the target at all, but still under a single minute.

A few potential gotchas can sneak in at this point also, and it will be faster to fix them now and confirm that in the imagery while the scenes can render relatively quickly, to be sure that we're ready to go to the next step.

Gotcha Tip #1 – Close the EOIR Sensor Scene

When you're making changes to STK-specific parameters that don't necessarily impact the EOIR scene, it's still likely to cause the scene to be regenerated. I find that when you're done looking at a specific scene and you're ready to start changing other scenario or object properties, it's best to close the window until you'll need it again.

Gotcha Tip #2 – Sensor Orientation

Different STK objects can have different body axes, and even when you think your sensor is pointed correctly, the "up" vector may surprise you. One of our SEs, Lauren McManus, figured out a common way to avoid this and wrote it up in a nice FAQ. Her idea is to create a custom vector in Analysis Workbench that's from your sensor's center to your target's center. This way, when pointing the sensor, we can use the Along Vector option and specify a constraint vector, such as the platform's zenith angle, to make sure that up in the image is constrained to up relative to the sensor's platform.

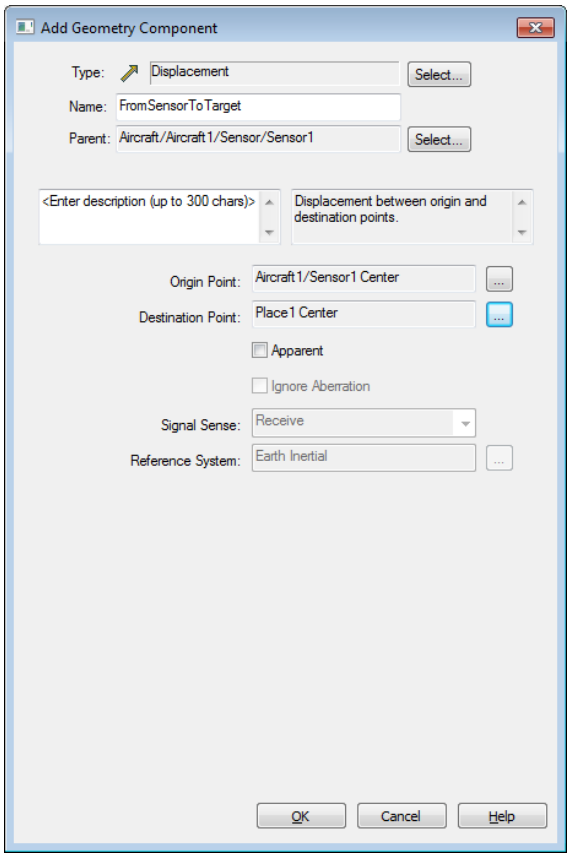

Figure 13 – Creating a custom targeting vector

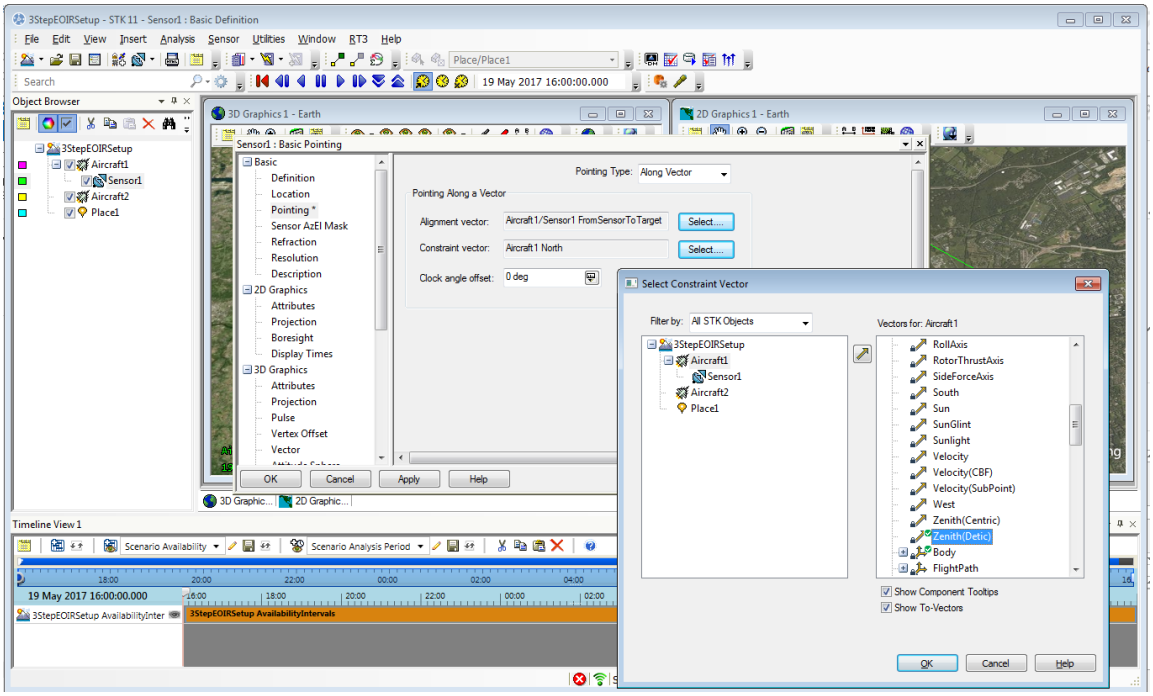

Figure 14 – Pointing with a constraint vector to orient the sensor

Gotcha Tip #3 – Target Orientation

Unfortunately, work-arounds can come with unintended consequences. One of these is that the default orientation for an aircraft or satellite is such that the positive Z vector points downward. However, all of the included custom 3D models for EOIR ground vehicles or ground sites have an orientation with the positive Z vector up.

Figure 15 – Default aircraft orientation body axes

To fix this, we can go into the target attitude and modify the properties to get our Z vector pointing upward.

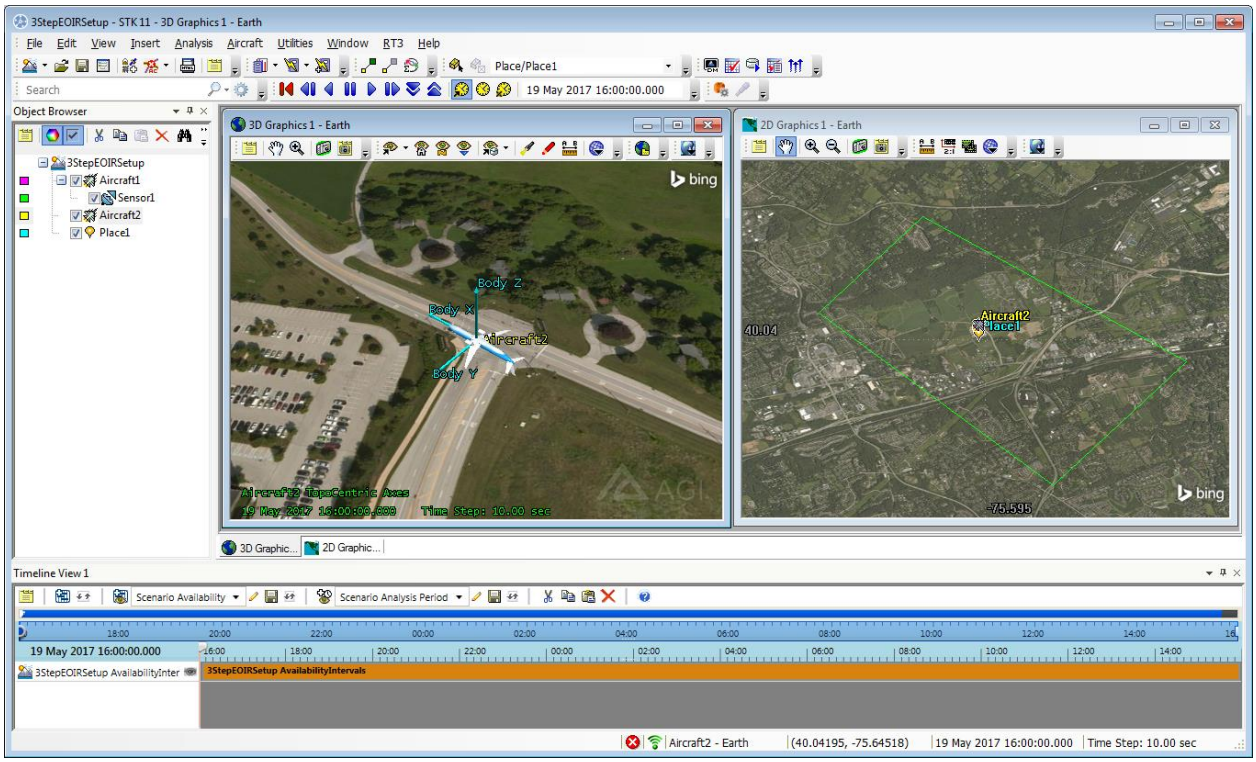

Figure 16 – Properly oriented ground target showing the desired body axes

Gotcha Tip #4 – Resolution Capability

This is an interesting one. I'll go into a little more detail for both the target specification and the sensor design in general, as well as adding in a few helpful equations that might end up being useful. For starters, as I mentioned with the target orientation, EOIR comes with a number of low-polygon sample 3D models. EOIR can take quite a bit of time to render more complex 3D geometry, so all of the models have been decimated to be very simplistic but interesting enough to use for EOIR scenarios. In this case, I'll use a well-known and characterized target, the M1 Abrams tank. Here's what that particular EOIR model looks like imported into Blender.

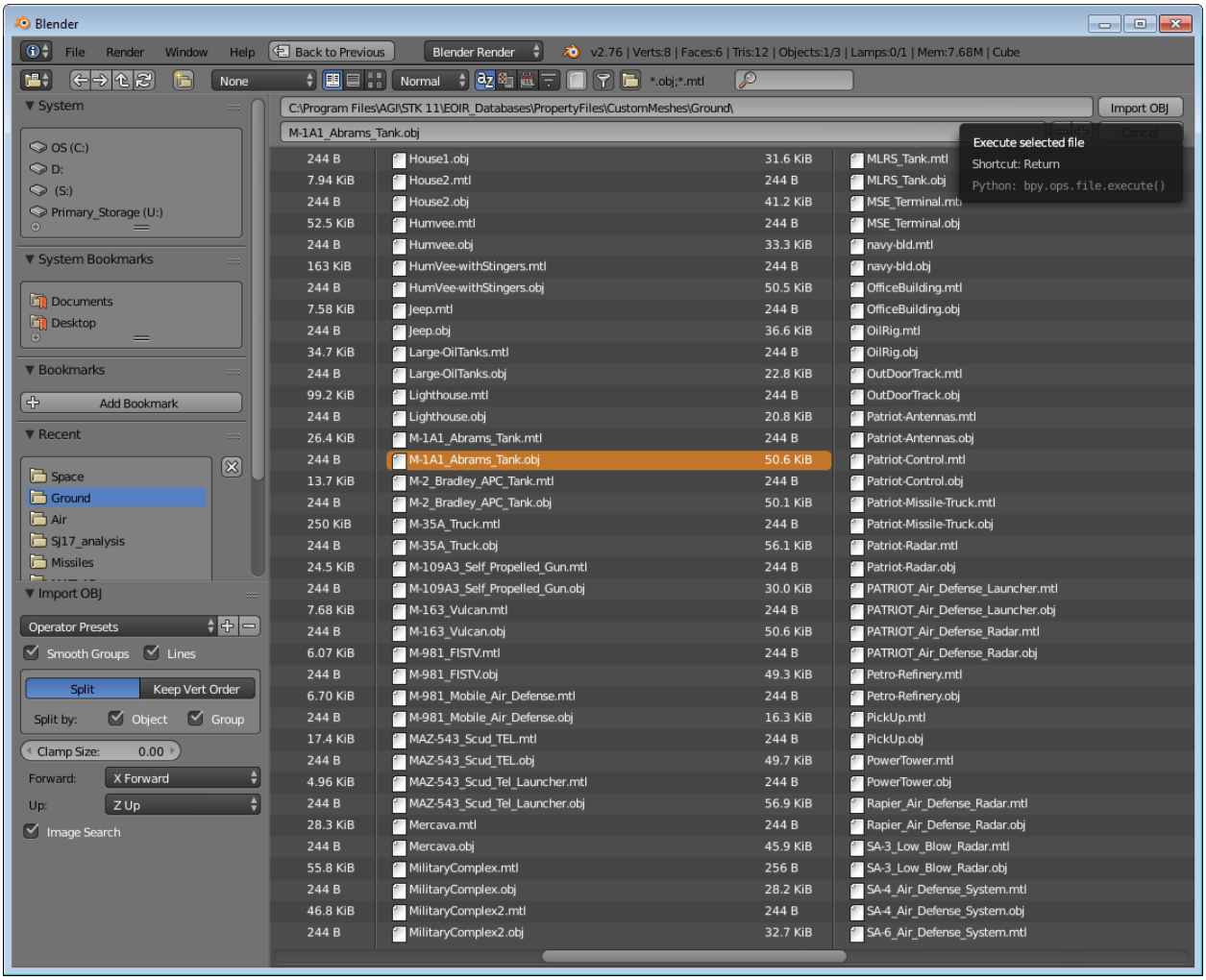

Figure 17 – Loading EOIR custom ground models into Blender

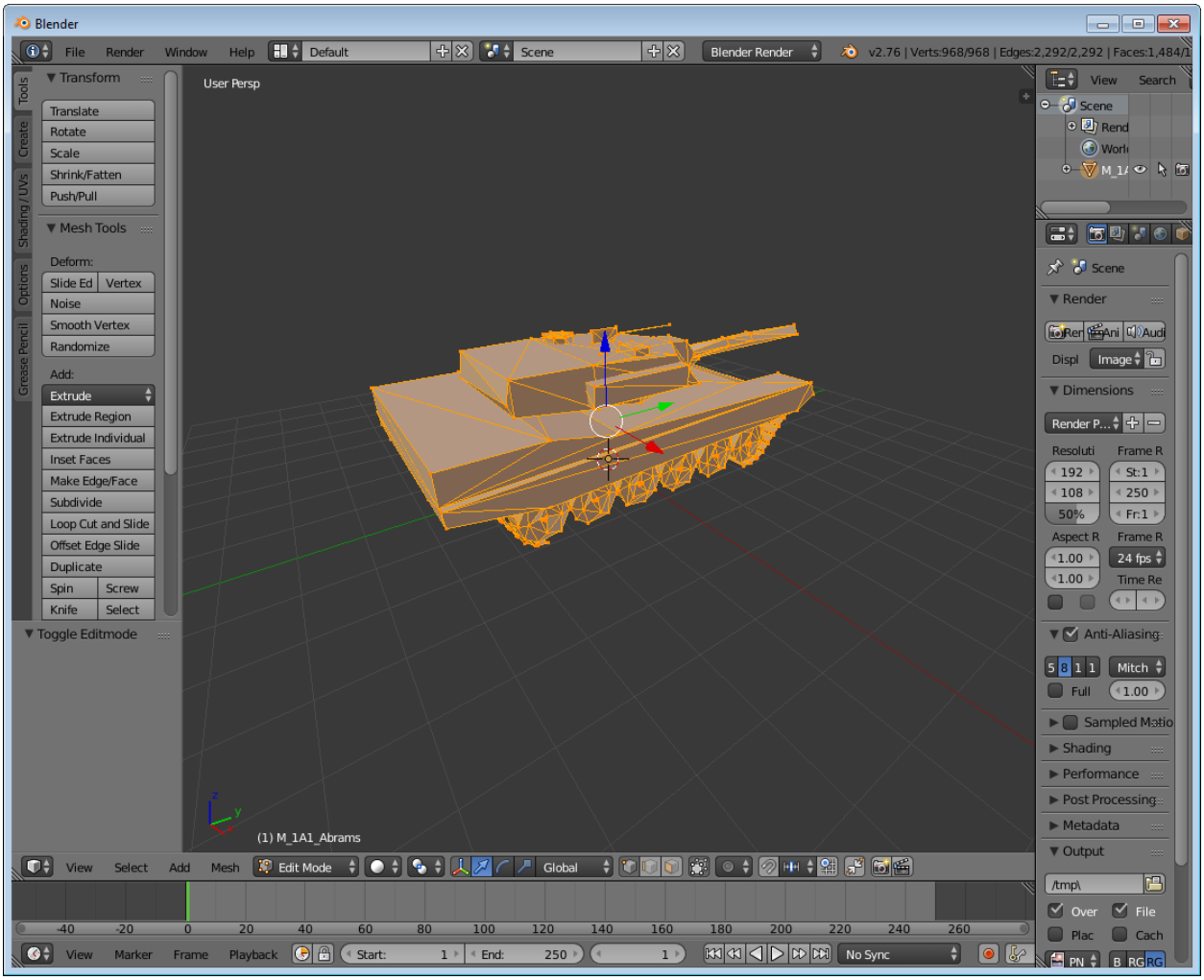

Figure 18 – Viewing the Abrams tank model in Blender

Also, from the Wikipedia page of this tank, it says the maximum dimension of this target is 9.77 meters with the gun forward. So I'll change my target EOIR shape properties to be a sphere with a radius of 4.885 meters, half the maximum dimension of our tank. I can also use STK to generate an AER (Azimuth, Elevation, Range) report that tells me my sensor-to-target range. It goes from about 18.4 km at the beginning of the scenario, down to about 14.5 km at the closest point, and then back up again near 18.9 km at the end of the loop.

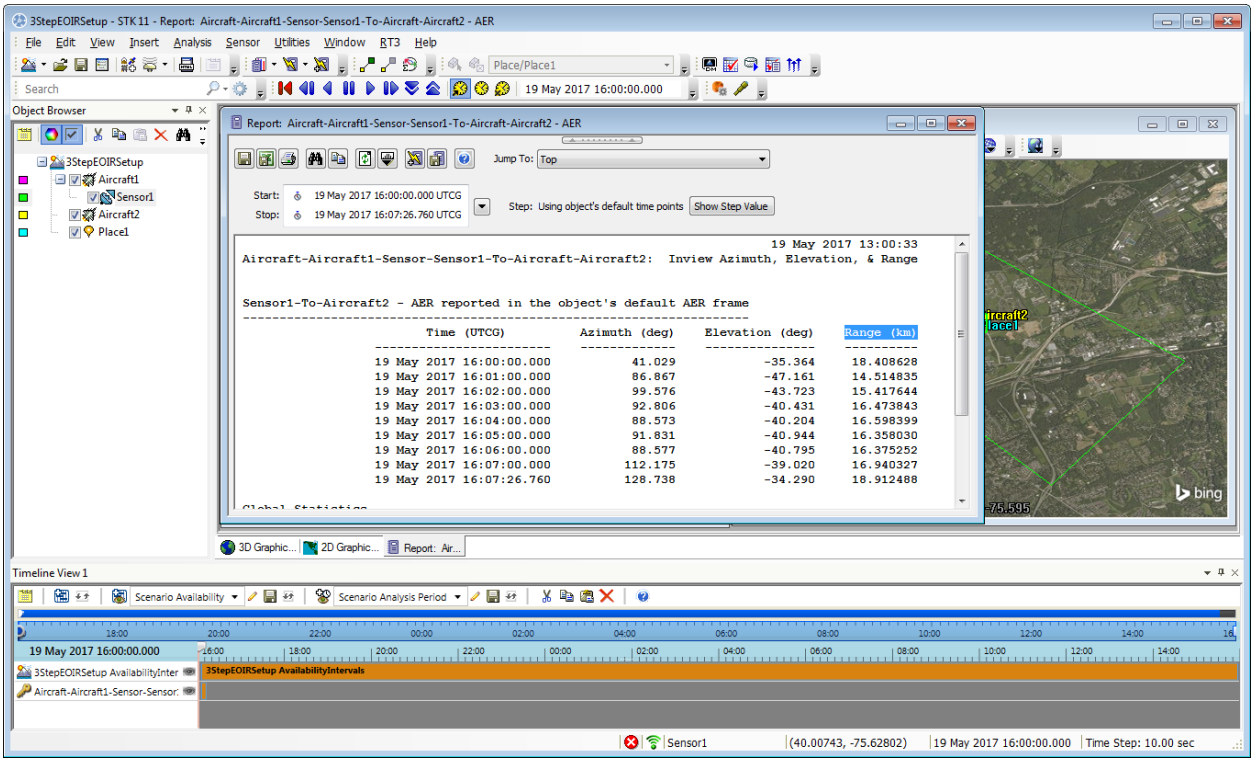

Figure 19 – Reporting the sensor-to-target range throughout the scenario

If you're trying to set up your scenario to evaluate the visibility of a target, you can use the sensor's FOV and sensor-to-target range to calculate the spatial sampling of your sensor. I'll call this the ground sample distance (GSD), although we're only going to be looking at this measurement perpendicular to the sensor's line-of-sight rather than actually projected onto the flat ground. I'll also use the variable r for range.

$$GSD = IFOV * r$$

In this case, the sensor's IFOV was 1.367 mrad. At a range of 18.4 km, that would give us a GSD of about 25 meters, not nearly high enough in resolution to make out the details of our tank, let alone even see it. Let's zoom in 10x by reducing our field-of-view half-angle from 5 degrees to 0.5 degrees and render a scene. I can play our scenario until I see that the field-of-view footprint has a clear view of our target and render it. Based on the spatial resolution, I would expect to be able to see our target now.

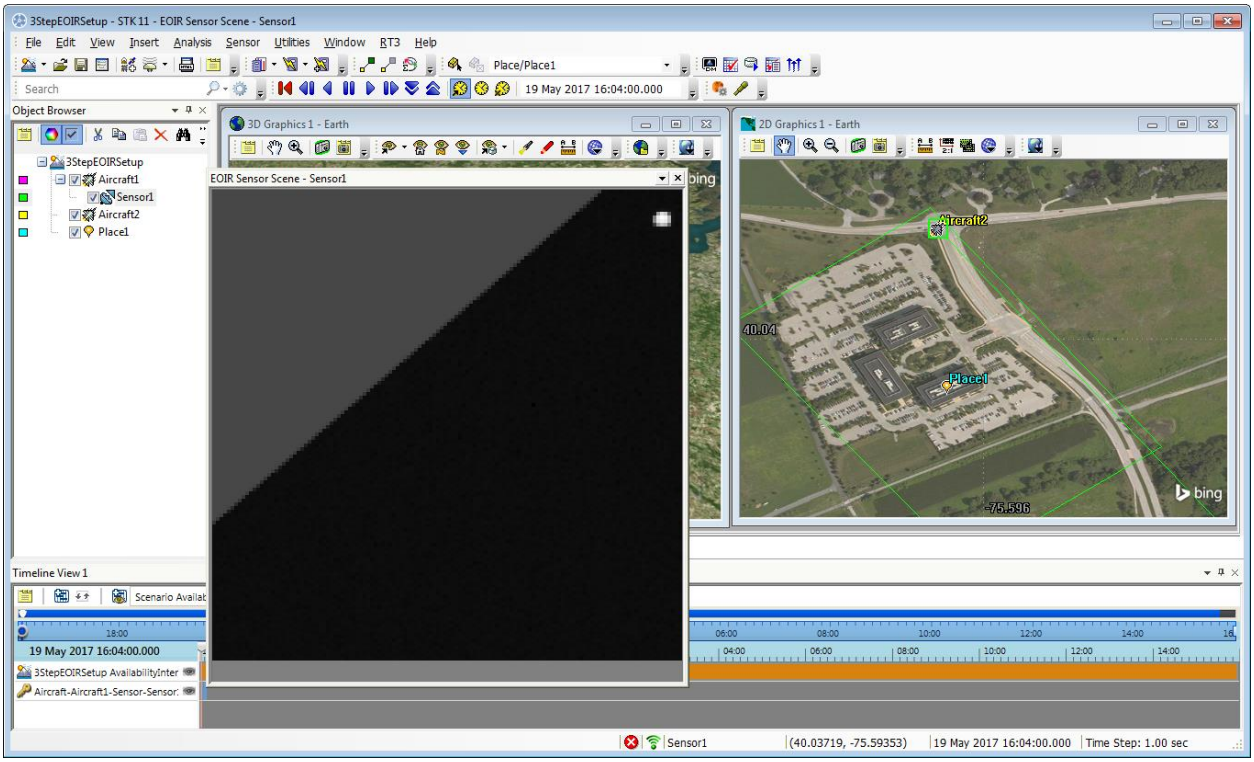

Figure 20 – Zoomed in target is visible

And there it is. At the top right corner of our image is our target sphere. However, by only adjusting the field-of-view, the overall sensor parameters may become unbalanced. For example, the detector pitch is being calculated in this configuration, but in reality the detector pitch would be fixed and when zooming in. This would normally be done by increasing the focal length. The calculated pitch is about 15 microns when zoomed in to the 0.5-degrees half-angle field-of-view with 128-by-128 pixels, which is pretty reasonable for a visible detector. But if we tried to zoom in another 10x, the calculated pitch for a field of view with 0.05-degrees half-angle would be 1.5 microns. As the pitch decreases, there are two dangers. The first is that the detectors may become too small to be physically practical. The second is, even if the detectors are still in a sensible range, the optics haven't been adjusted and the aberrations on the focal plane may not be sufficient to resolve the target. The resulting image is shown below. As you can see, even though we are able to zoom in on the target, it's starting to become blurry.

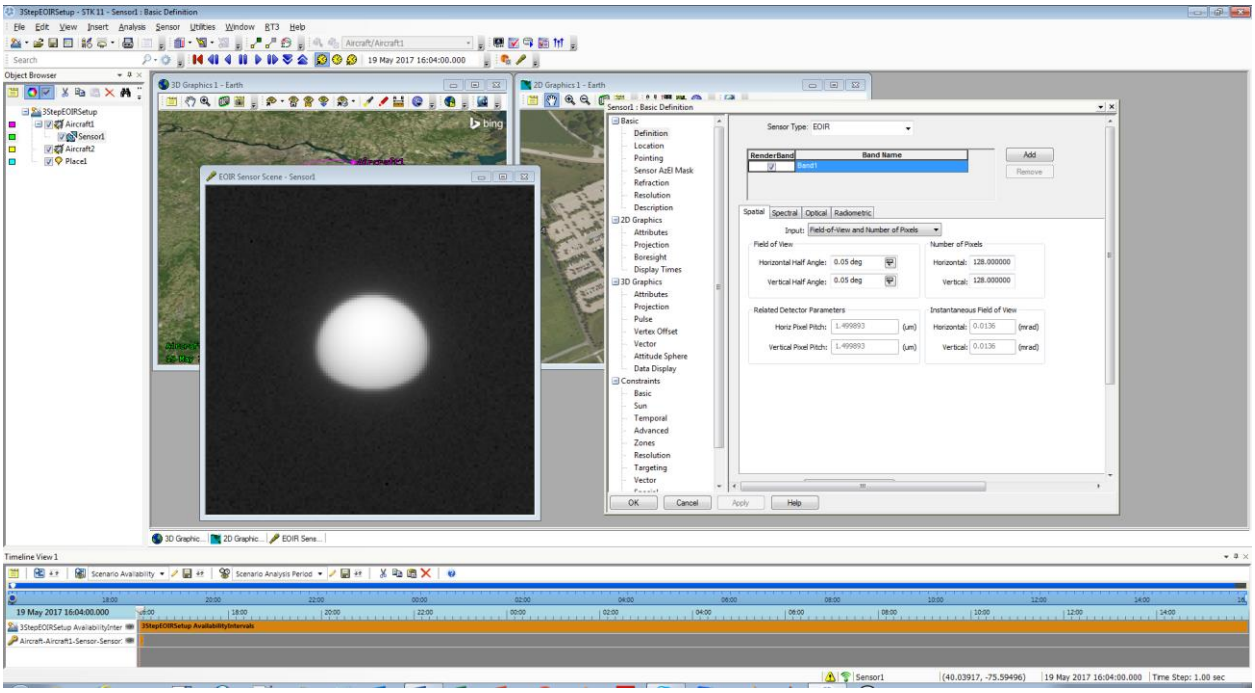

Figure 21 – Zoomed way in on our target of interest

Besides the GSD calculation, we can rewrite the definition in terms of the pitch (p) and focal length (f) as:

$$\frac{GSD}{r} = \frac{p}{f}$$

Furthermore, we can characterize how blurry our system is with a convention called System-Q (Q). This is defined below as a function of the wavelength of light (λ) and the f-number of the system (f/#). Normally you would want your system to have a Q of about 1.0. Anything lower would be a very sharp imaging system and anything higher would become more blurry due to the optical diffraction.

$$Q = \frac{\lambda \, f/\#}{p}$$

A spreadsheet can allow you to do some quick resolution calculations to determine the GSD at various ranges. This let's you see if you have sufficient spatial resolution, but also calculates the System-Q as well to make sure that, even if you have sufficient spatial resolution, you aren't stretching your optics beyond their physical limits. Here are two examples, with the first varying the range and looking at the impact on GSD, and the second varying the pitch and looking at the impact on System-Q.

Pitch [um]	15.00	15.00	15.00
Focal Length [m]	0.11	0.11	0.11
IFOV [urad]	136.36	136.36	136.36
Aperture Diameter [m]	0.055	0.055	0.055
F/# [ratio]	2.00	2.00	2.00
Range [km]	14	1.4	0.14
GSD at Range [cm]	190.91	19.09	1.91
System-Q [pixels]	0.07	0.07	0.07

GSD as a function of range

Pitch [um]	15.00	1.50	0.15
Focal Length [m]	0.11	0.11	0.11
IFOV [urad]	136.36	13.64	1.36
Aperture Diameter [m]	0.055	0.055	0.055
F/# [ratio]	2.00	2.00	2.00
Range [km]	14	14	14
GSD at Range [cm]	190.91	19.09	1.91
System-Q [pixels]	0.07	0.67	6.67

System-Q as a function of detector pitch

Step 3 – Turning on all the bells and whistles

Things that I consider bells and whistles with EOIR are:

- Full Detector Resolution
- Custom reflectance/emissivity/temperature maps
- Custom 3D target models
- Atmosphere
- Full Star Catalog

These are all computationally expensive to include but add a significant amount of fidelity to the EOIR simulations. For the scenario we've been building up, four of these parameters are really applicable (we won't see any stars looking down) and I'll include image results as well as the time to generate each of these images with the full permutation of 16 combinations for these four options.

I'll only provide a little bit of detail for each of these parameters to keep things concise:

- Full Resolution:
  - Yes – 512x512 pixel rendering
  - No – 128x128 pixel rendering
- Custom Reflectance Map:
  - Yes – 3,000x2,187 pixel high-resolution reflectance map completely covering the sensor footprint on the ground
  - No – Just using the default EOIR high-resolution material map

- Custom 3D Target:
    - Yes – Using the included 1,484 M_1A1_Abrams_Tank.obj 3D model
    - No – Using the default EOIR sphere shape
- Atmosphere:
    - Yes – Using the MODTRAN based atmospheric database for a rural aerosol 23 km visibility with 45.8% relative humidity
    - No – Using no atmosphere

The timing results I'll present here are all from a Dell Precision M4600 with a 2.3 GHz i7 processor and 8 GB of memory, which I would consider at the very low end of performance. That might be the good news, but what I can tell you is that this poor laptop has been abused for years and hasn't aged well. Regardless, I want to show performance metrics that just about any other workstation can easily outperform. I did restart the machine with the bare minimum other applications running, to give it a fighting chance. I also restarted the machine between runs to make sure that these numbers could be compared directly.

An additional note is that all of these times in the table are reported for rendering times after the initial setup. This initial setup is performed on the first rendering, which will require more time but then render faster for each subsequent rendering. When rendering a larger image size, new arrays will need to be allocated; the time for this was negligible relative to the rendering time. The custom reflectance map file will need to be read into memory (about 16 seconds), the custom 3D model file will need to be read into memory (the time for this was negligible relative to the rendering time), and the atmospheric database will need to be loaded (approximately 2 minutes).

Full Resolution	Custom Reflectance Map	Custom 3D Target	Atmosphere Turned On	Time to Render [minutes:seconds]
No	No	No	No	00:07.5
No	No	No	Yes	00:38.4
No	No	Yes	No	00:14.4
No	No	Yes	Yes	04:46.8
No	Yes	No	No	00:07.3
No	Yes	No	Yes	00:37.8
No	Yes	Yes	No	00:14.8
No	Yes	Yes	Yes	04:45.0
Yes	No	No	No	01:26.7
Yes	No	No	Yes	02:03.4
Yes	No	Yes	No	02:25.2
Yes	No	Yes	Yes	08:08.7
Yes	Yes	No	No	01:37.7
Yes	Yes	No	Yes	02:17.2
Yes	Yes	Yes	No	02:35.5
Yes	Yes	Yes	Yes	08:06.4

These results show that difference between all the bells and whistles being turned off to being turned on is about 8 minutes per image. This is a huge amount of time that you shouldn't commit to until you're fairly certain that all of your parameters are set up and it's rendering simply as you would expect. We can also see that the largest hits in this particular scenario are when both the atmosphere is turned on and we're using a custom 3D model. This requires many more complicated atmospheric paths to be considered and calculated.

Figure 22 – STK 3D scene window snapshot used to generate the reflectance map

Figure 23 – All the bells and whistles turned on, with the tank target at the top right of the image

Conclusion

I hope this discussion helps you get set up and running without all of the frustration that can come along with learning these lessons yourself the hard way. Moving forward, I'm going to be working hard to improve the performance of EOIR, so I hope to see all of these times reduced. As these performance improvements are rolled out, I'll make sure I update this table. However, if you have performance requirements, we would love to hear about them. Currently I'm reviewing EOIR fail cases where potential customers could not move forward with consideration because of specific performance limitations. Not all of these performance metrics may be realistic. However, I'd like to make sure that we're constantly grounded in reality. At the end of each development cycle, I'd like to make sure that what we're delivering directly addresses the needs of our customers today as well as keeping EOIR headed in the right direction for future generations of engineers.

Thank you, and I really hope this helps.

- Pat North, EOIR Lead, Image and Computer Scientist