STK Astrogator Tutorial Using the Object Model

This tutorial shows you how to use the Astrogator objects in the STK Object Model in a custom application to accomplish many of the tasks for which you might otherwise use Connect or the Ansys Systems Tool Kit® (STK®) digital mission engineering software GUI. The source code is given in C#. Familiarity with Microsoft Visual Studio 2013 or later, the STK Object Model tutorial, and STK/Astrogator® are presumed. This tutorial is the Object Model version of the "Hohmann Transfer Using the Targeter" exercise in the Astrogator Help. It may be helpful to go through the tutorial in the GUI before starting the Object Model version, to get a feel for how the scenario is constructed and executed

Source Code Location: The source code for this tutorial is in the following text file:

```
<Install Folder>\Help\STK\LinkedDocuments\AstrogatorObjectModelTutorial.txt
```

The code is broken into sections corresponding to the sections in this tutorial.

Note: If you don't see this location of the file, make sure you have the STK help installed.

Project setup

Starting Visual Studio and creating your project

You will use one STK X control for this tutorial: the 3D Globe control. To add this control and to use the STK Object Model, you must add the following references to your project:

- STK Objects – a COM Library containing types, interfaces, events, and classes representing various aspects of the STK application structure
- STK Util – a library containing objects and enumerations shared by the STK Objects and STK X type libraries
- STK X - a collection of COM components and ActiveX controls that use AGI's STK Engine embeddable technology
- STK Astrogator – the Astrogator Object Model

Here are the steps to set up your project and add the necessary references:

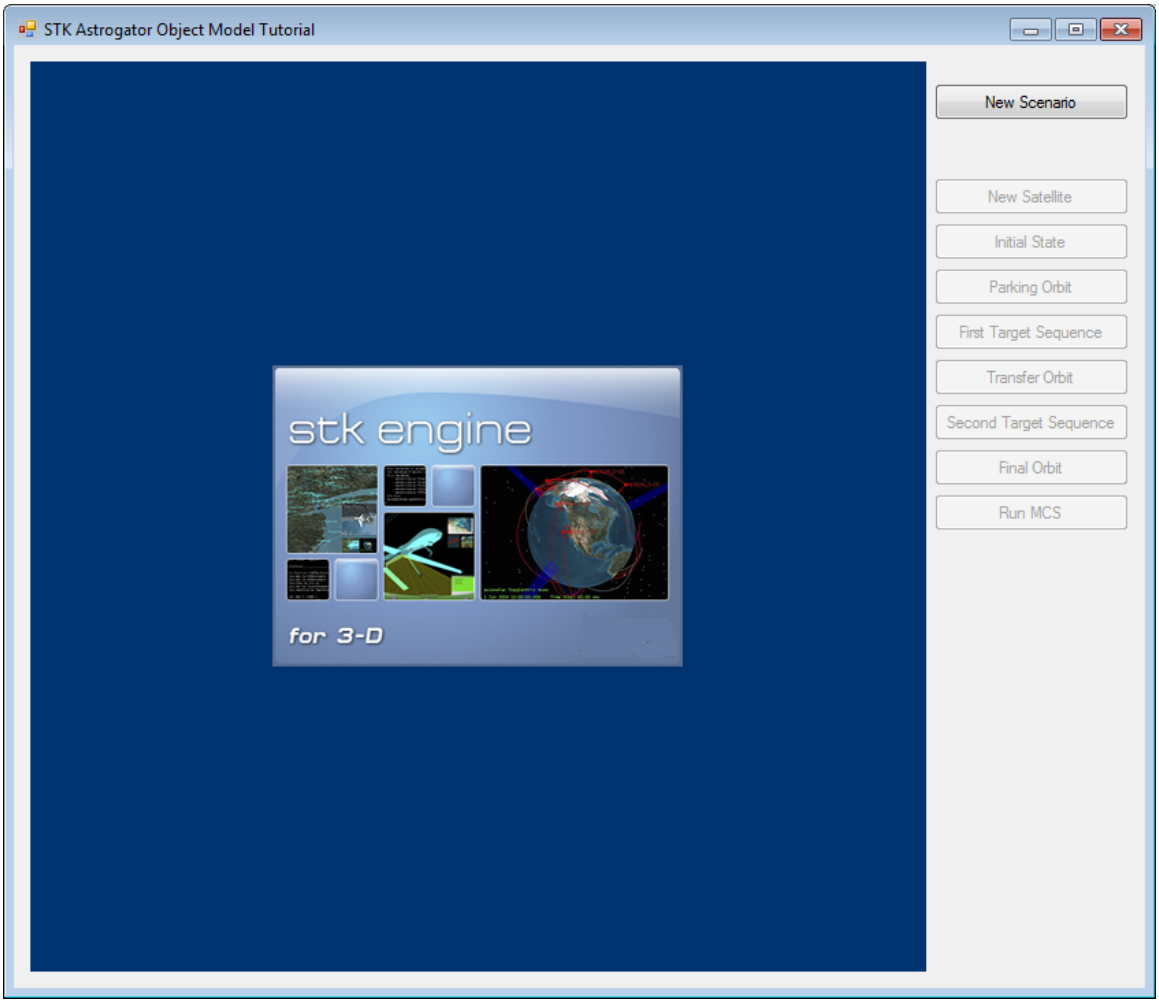
1. Start Visual Studio, and in the File menu select "New Project".
2. Under the TInstalledT templates, go to Templates, select C#, and then in the Windows page choose TWindows Forms ApplicationT. Configure the rest of your project settings as you see fit. In this tutorial, AstrogatorTutorialT is used as the project name.

3. Once your new project has loaded, go to the Project menu and select the “Add Reference…” item. Select the COM tab and add the following references to your project:

    - AGI STK Objects 13
    - AGI STK Util 13
    - AGI STK X 13
    - AGI STK Astrogator 13

Adding STK X Controls to your form

Before you begin writing Object Model code, use the design view in Visual Studio to draw the user interface for your custom application. When you are finished, the GUI you create will look something like this:



The application that you create will allow the user to:

- create a Scenario and display it in a 3D window
- configure an Astrogator MCS with targeting
- run the MCS and report values from the targeting

Here are the steps to create the GUI:

1. Open the automatically-generated “Form1” in design view and stretch the form so that it substantially fills the available workspace (with Visual Studio maximized).

2. Open its Properties window and change the “Text” property from “Form1” to “STK Astrogator Object Model Tutorial”.

3. If the Toolbox is not already in your workspace, add it by opening the View menu and clicking Toolbox. Right-click in a blank area (i.e., neither a control nor the Toolbox title bar) in the General section of the Toolbox and select “Choose Items…” from the menu.

4. On the “.NET Framework Components” tab, click the browse button and go to the following location:

   ```
   <INSTALL FOLDER>\bin\Primary Interop Assemblies
   ```

5. Select the AGI.STKX.Controls.Interop.dll, click the Open button, and then click OK. This will add the AxAgUiAx2DCntrl and AxAgUiAxVOCntrl options to your Toolbox.

6. From the Form1 design view, select the General category of the Toolbox, select AxAgUiAxVOCntrl, the STK 3D Control, and draw a rectangular shape in the form, leaving room on the far right for buttons.

7. Place a button in the upper-right-hand corner and change its Text property to "New Scenario".

8. Arrange eight more buttons vertically below the “New Scenario” button. Starting with the top button and working down, set the Text properties of the buttons to:

   - New Satellite
   - Initial State
   - Parking Orbit
   - First Target Sequence
   - Transfer Orbit
   - Second Target Sequence
   - Final Orbit

- Run MCS

9. Set the Enabled property to False for all buttons except the “New Scenario” button.

Scenario setup

Adding the Core of the STK Object Model to your project

Refer to the text file identified above under Source Code Location for the code segments to be added.

1. Open the code for the form by right-clicking in the Form Designer and selecting View Code or by right-clicking on the Form1.cs item in the Solution Explorer and selecting View Code.

2. Add statements at the beginning of the code (*using* in C#) to gain access to the types defined in the STK Objects, STK Util, and STK Astrogator libraries.

3. Define member variables for AgStkObjectRoot — the top-level object in the Object Model hierarchy — and the IAgVADriverMCS interface; IAgVADriverMCS is the interface that will enable you to interact with the Astrogator Object Model for a satellite.

4. Initialize the instance of AgStkObjectRoot in the form’s constructor. The IAgVADriverMCS interface cannot be initialized until a satellite is created, so this member variable will be initialized later.

5. Edit the Form1 Dispose() method as indicated in the source code text file. In newer versions of Visual Studio, the Dispose method may be in a separate source file, Form1.Designer.cs, so look for it there if it isn’t in the current file.

6. Referring to the source code text file, add a function to create a new scenario to the main body of the Form1 class and then set the units for the scenario and the time period for the scenario.

7. Add a Click event function for the New Scenario button, and have this call the new scenario function created in step 6.

At this point you should be able to build and run your application. Click the New Scenario button to create the scenario and display the 3D Globe.

Note: *If you receive an exception when trying to run the tutorial project, make sure the “bitness” of the STK application and your Visual Studio match. If you are using the 32-bit version of the STK software, the application needs to run as a 32-bit process. For 64-bit STK, you need to run this as a 64-bit process. Update your project Solution Platform settings from “Any CPU” to the appropriate “x86” or “x64” to avoid the error.*

Creating an Astrogator satellite, setting its Properties, and running the MCS

Mission overview

The purpose of this mission is to illustrate the classic Hohmann transfer of a satellite with an initial circular orbit with a radius of 6,700 km to a final circular orbit with a radius of 42,238 km. You will use the following MCS segments to achieve this:

- An Initial State defining a parking orbit with a radius of 6700 km
- A segment to propagate the parking orbit
- A Target Sequence containing an Impulsive Maneuver to enter the elliptical transfer orbit
- A segment to propagate the transfer orbit to apogee
- A Target Sequence containing an Impulsive Maneuver to enter the outer circular orbit
- A segment to propagate the outer orbit

Creating an Astrogator satellite

1. Create a satellite object, get an interface to the new satellite, and set the propagator to Astrogator.

2. Acquire an interface to the DriverMCS interface of Astrogator. This is the central interface from which to control Astrogator for a satellite.

3. Clear out all the segments from the MCS to start with a clean slate.

4. Add a call to this function to the Click event for the “New Satellite” button.

Configuring the Initial State

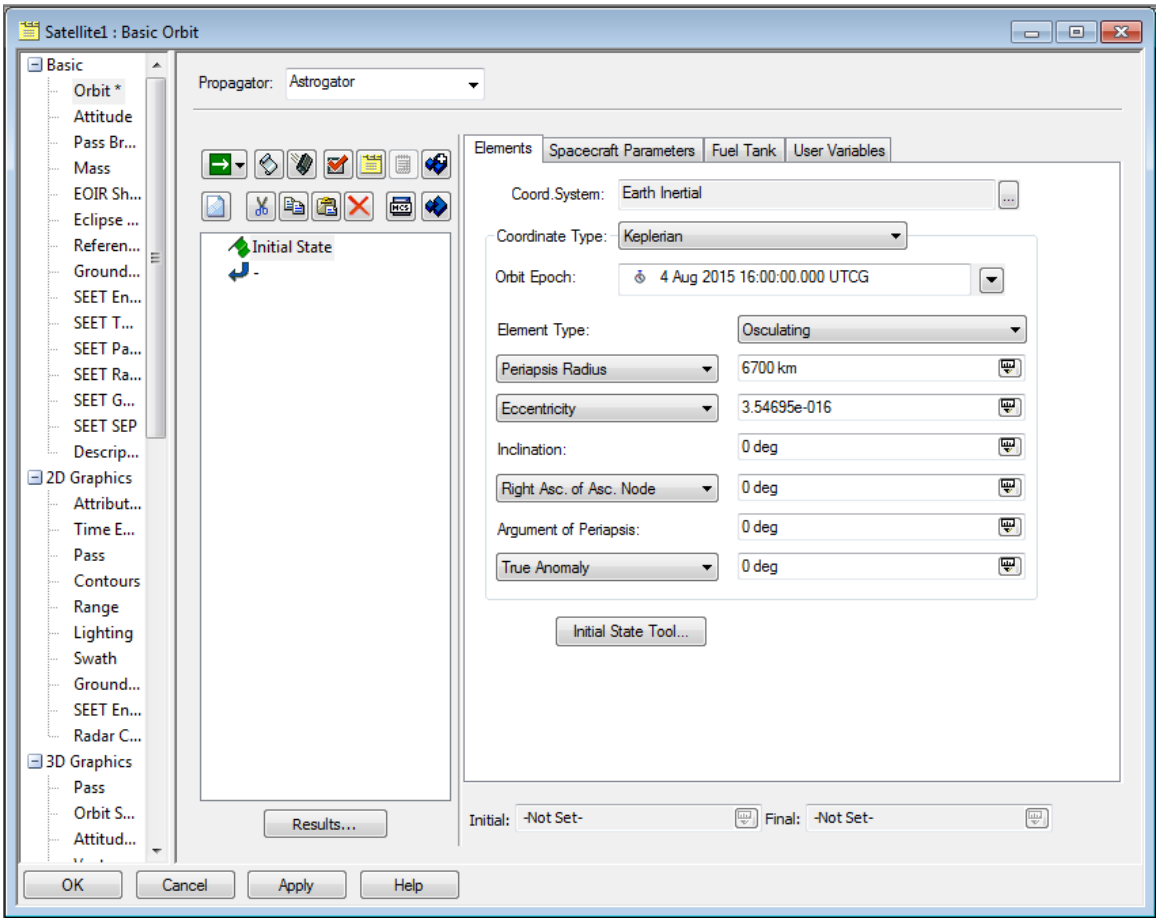
The satellite starts in a 6,700 km radius circular orbit.

1. Call the Insert method on the Main Sequence. The first argument is the segment type, the second argument is the name of the segment, and the third argument is the segment before which the new segment will be inserted. Insert an initial state segment with the name “Inner Orbit” in the Main Sequence before the end (“-“) segment.

2. Change the element type to Keplerian and acquire an interface to this element set.

3. Using the interface, set the periapsis radius size to 6700 km and set the other elements in the element set to 0.

4. Add a call to this function to the Click event for the “Initial State” button.



Configuring the parking orbit

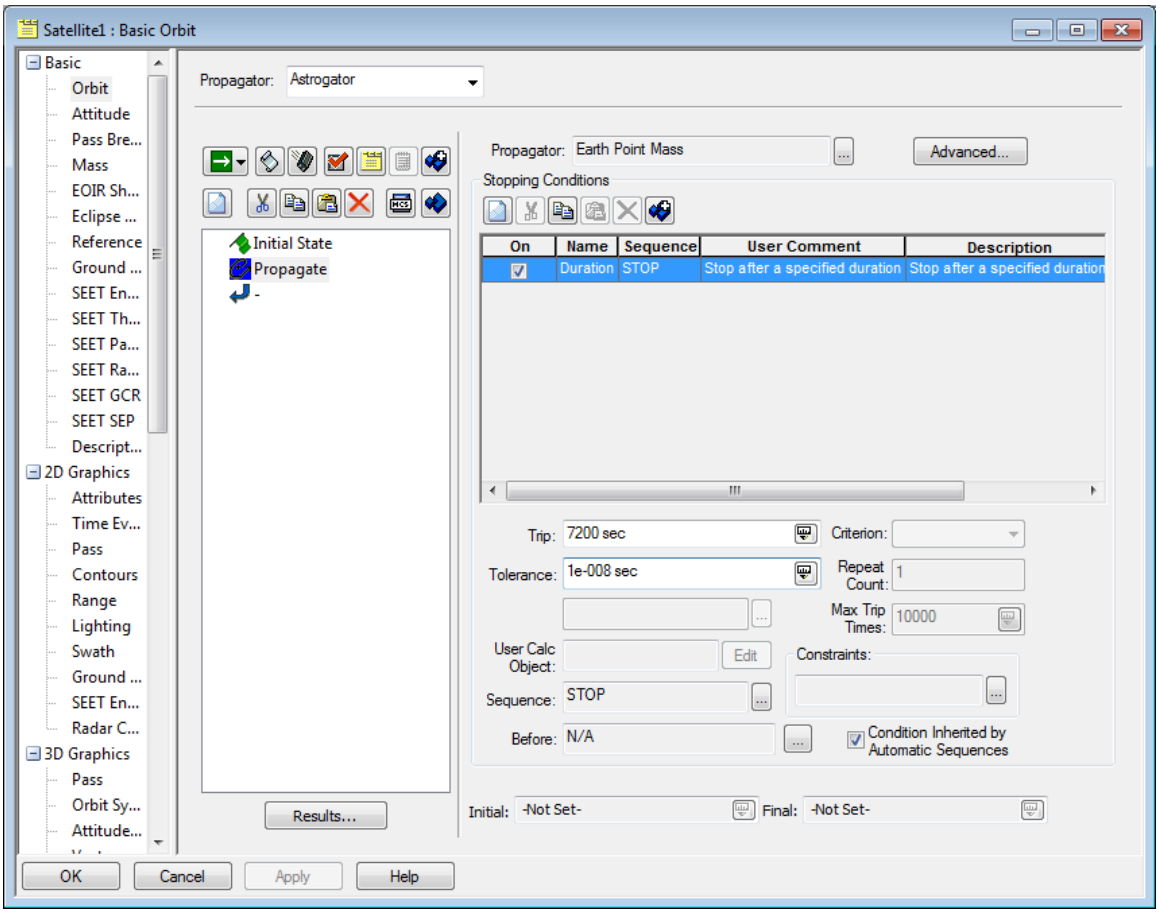
The maneuver occurs two hours after the initial state. Use a Propagate segment to propagate the satellite's orbit in time.

1. Insert a Propagate segment named "Propagate" into the main sequence before the end segment.

2. Change the propagator to "Earth Point Mass" and the segment color to blue.

3. Set the duration stopping condition trip value to end the propagation of this segment after 7200 sec.

4. Add a call to this function to the Click event for the Parking Orbit button.



Configuring the Target Sequence for the first maneuver

Now that the satellite is stopping where the maneuver should begin, you will use a target sequence that will find the correct Delta-V to place the satellite on a transfer orbit that has an apogee at the radius of the desired final orbit (42,238 km).

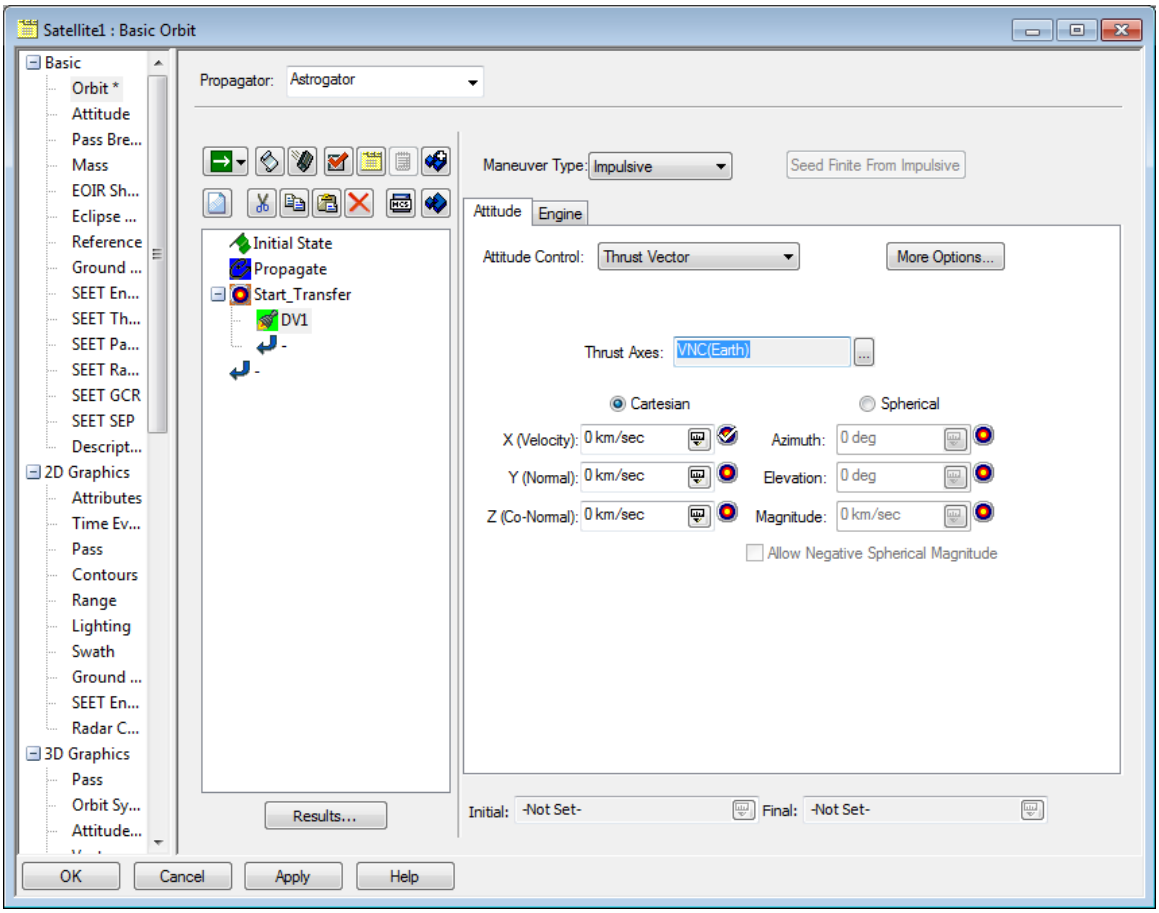
1. Insert a target sequence called “Start Transfer”.

2. Insert a maneuver segment named “DV1” inside the target sequence before the target sequence’s end segment and acquire an interface to the target sequence.

3. Using the interface, change the maneuver type to impulsive, the attitude control to thrust vector, and the thrust axes to “VNC (Earth)”.

4. A search profile can only change values that are available as controls, so set the X component of the thrust vector as an available control.

5. Search profiles use results on segments as goals. The apogee of the transfer orbit should be 42,238 km, so the segment needs to report the apogee radius as a result. Add the radius of apoapsis calc object as a result to the segment.



Next you must configure the differential corrector (DC) search profile. By default, a target sequence has one DC profile.

1. Get a handle to the default DC profile.

2. Available controls and results are held in collections on the DC called ControlParameters and Results, respectively. Get a handle to the available control, which is the X component of the thrust vector on the DV1 segment.

3. Enable the control to allow the DC to change this value during targeting.

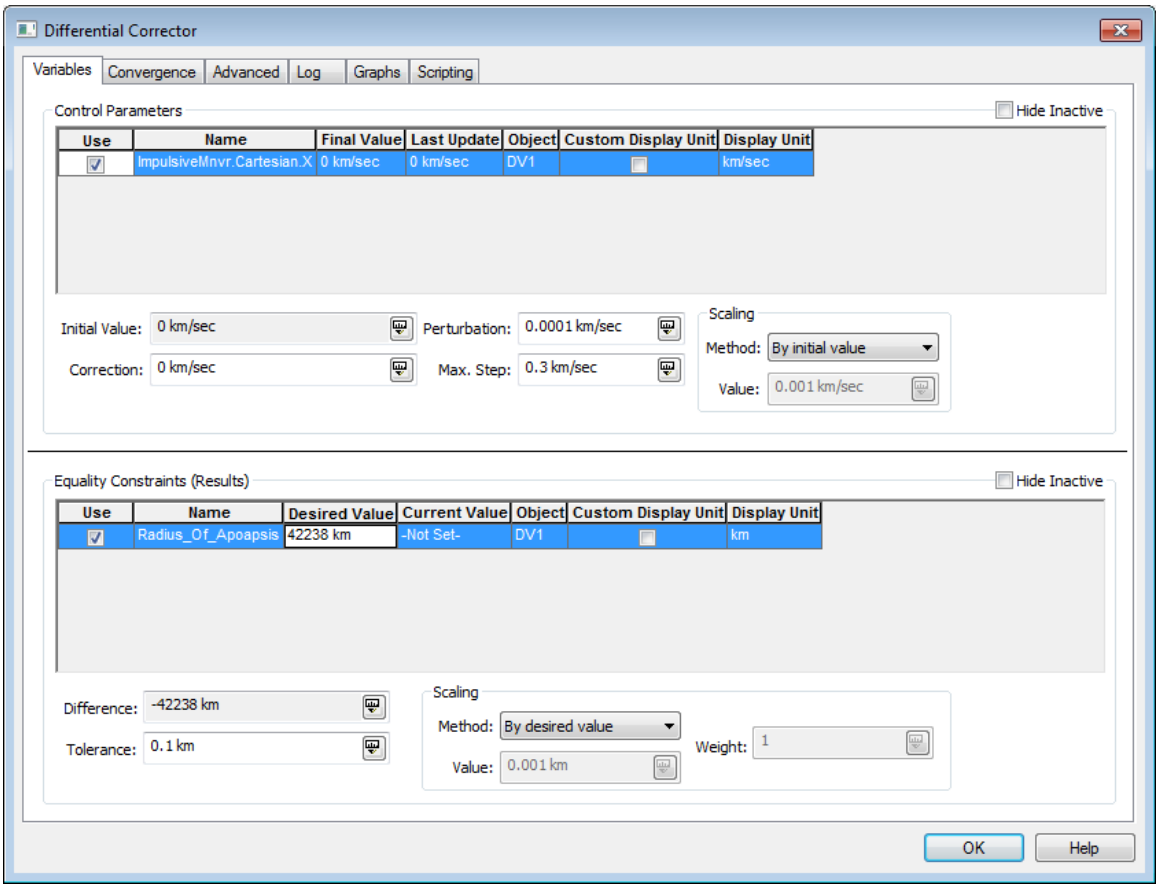
4. In order to make the DC convergence behavior better, change the max step to 0.3.

5. Get a handle to the radius of apoapsis result on the DV1 segment.

6. Enable this result as an active equality constraint on the DC; change the desired value to 42238, the tolerance to 0.1, and the maximum iterations on the DC to 50.

7. Switch the action of the target sequence to “Run Active Profiles”.

8. Add a call to this function to the Click event for the First Target Sequence button.



Configuring the transfer orbit

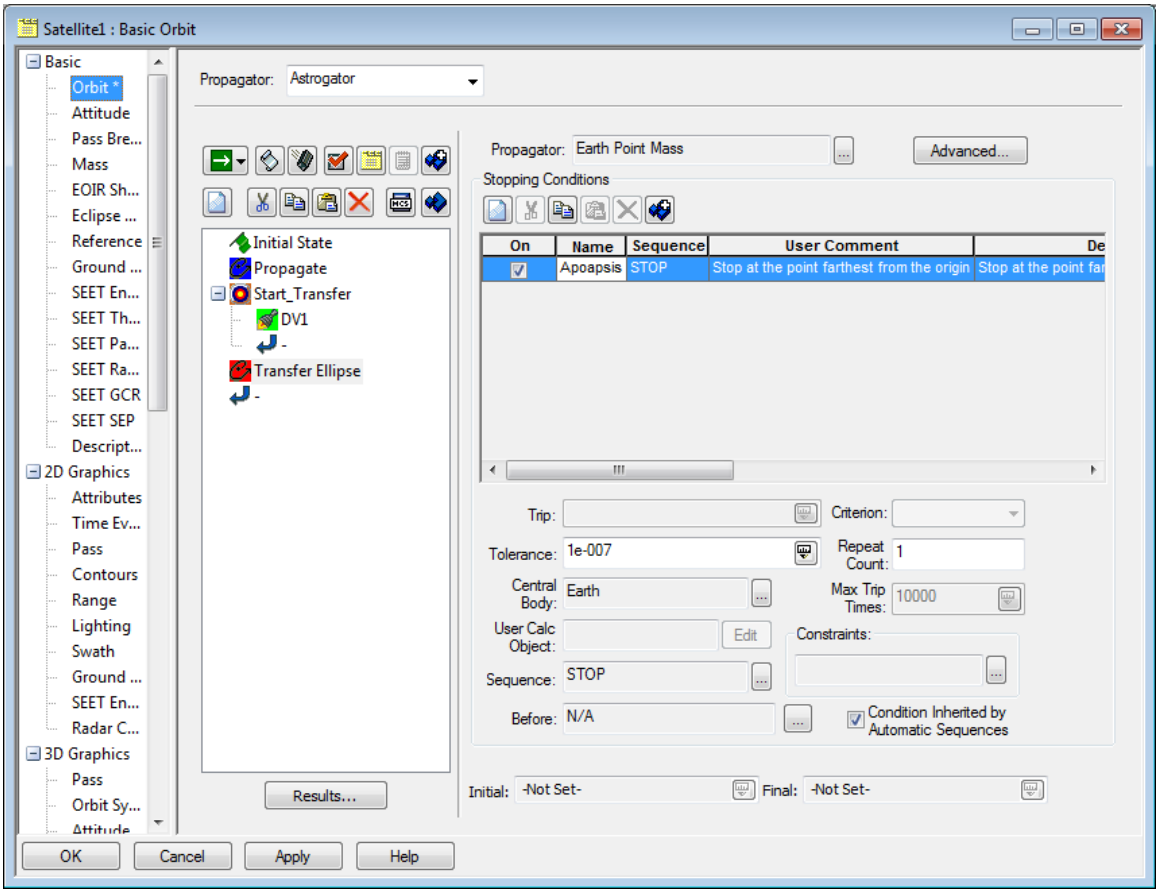
Once the target sequence finds the correct magnitude for the impulsive maneuver, the satellite will need to propagate to apogee (the location of the second maneuver).

1. Insert a propagate segment called “Transfer Ellipse” before the end segment in the Main Sequence. Change the color to red and select the “Earth Point Mass” propagator.

2. The satellite should stop at apogee, so add the “Apoapsis” stopping condition.

3. The duration stopping condition that is on all propagate segments by default is not needed, so remove it.

4. Add a call to this function to the Click event for the “Transfer Orbit” button.



Configuring the Target Sequence for the second maneuver

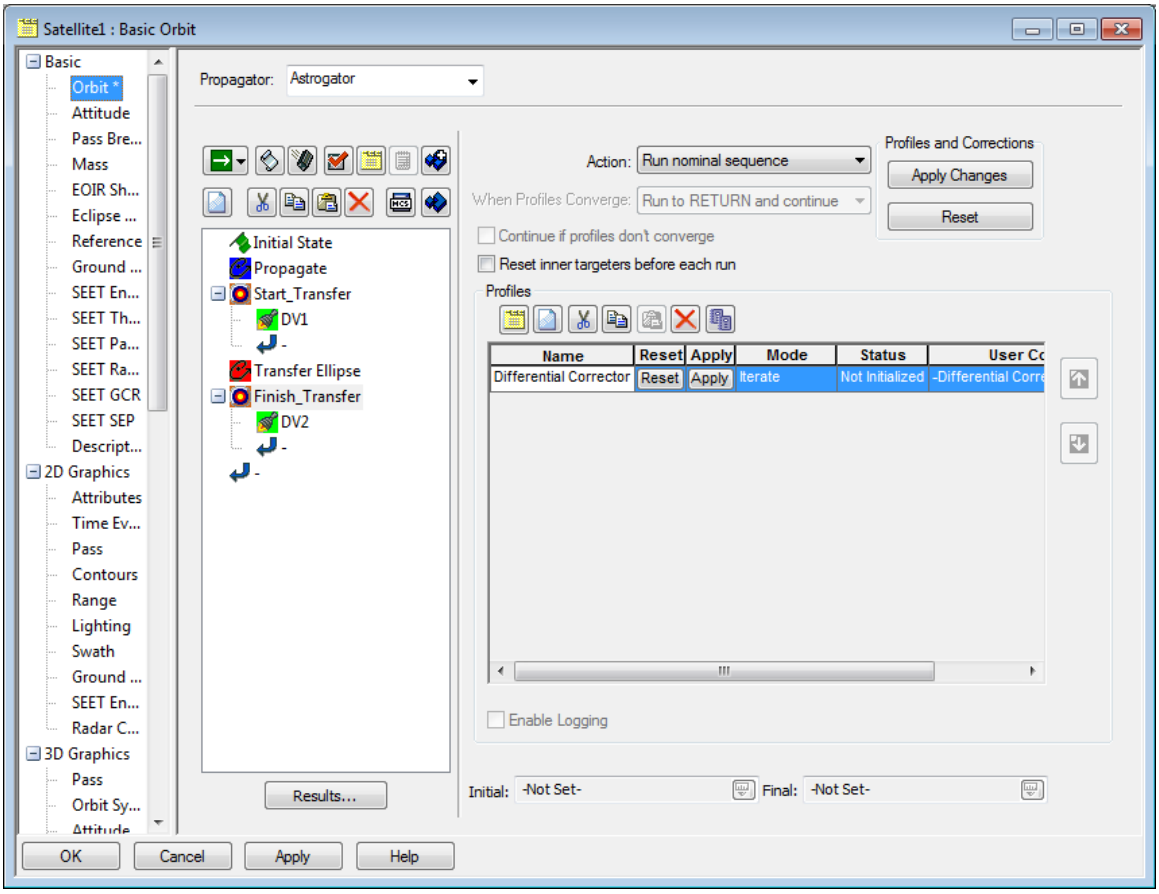
The satellite is now stopped at apogee at a radius of 42,238 km, the location for the second maneuver, which will circularize the orbit at the final radius.

1. Add a second target sequence named “Finish Transfer” to the MCS, before the end segment.

2. Insert a maneuver segment named “DV2” inside the target sequence before the target sequence’s end segment and acquire an interface to the target sequence.

3. Configure this maneuver to be identical to the DV1 maneuver, except for the result on the segment. For this maneuver, the targeter should find the value that circularizes the orbit.

4. Add the “Eccentricity” calc object to this maneuver instead of the “Radius of Apoapsis” calc object from the DV1 maneuver.



5. Get a handle to the DC profile on the second target sequence.

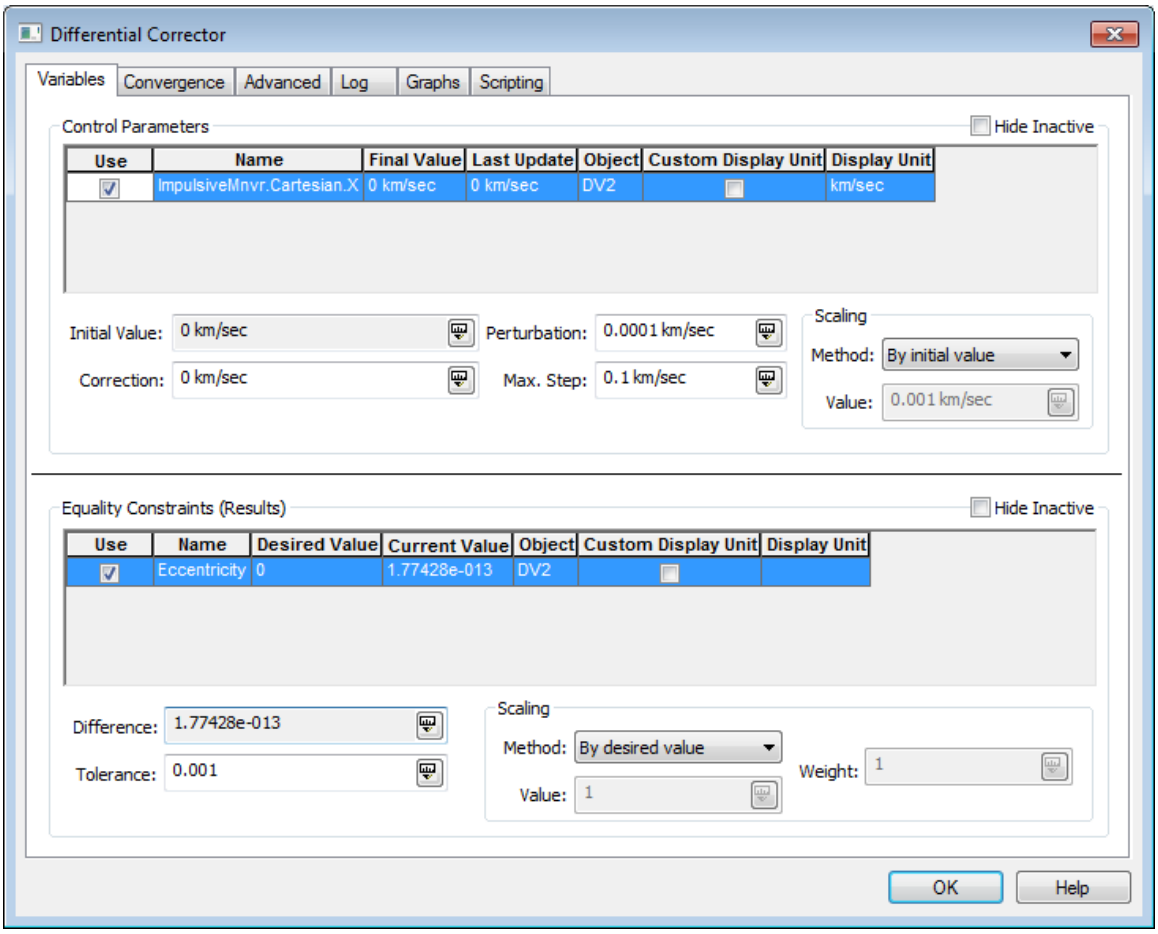
6. Configure the control — the X component of the thrust vector on DV2 — identically to DV1.

7. Enable the eccentricity result as an active result on the DC and set the desired value to 0, a circular orbit.

8. Set the tolerance on the eccentricity result to 0.001 to force the targeter to find an answer closer to 0 than the default of 0.1, which is large for eccentricity.

9. Change the action of the target sequence to “Run Active Profiles.”

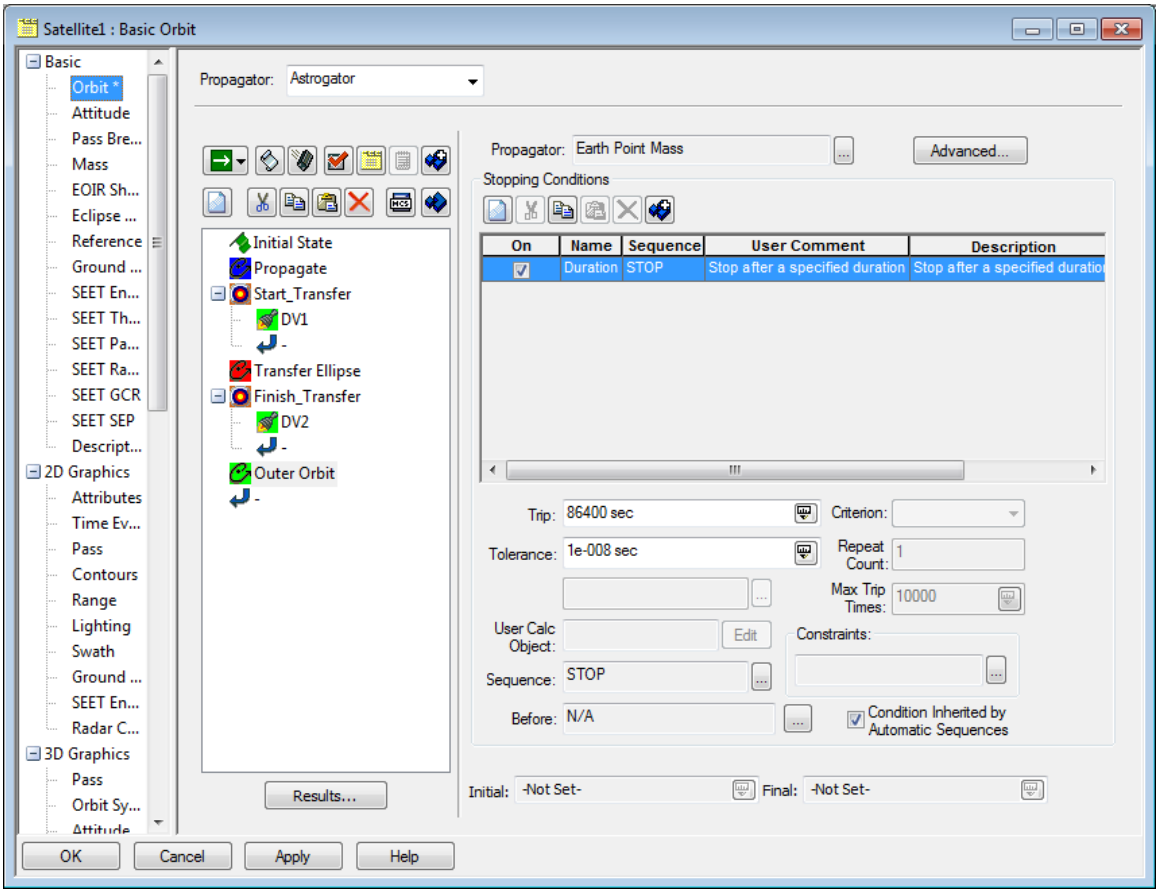
10. Add a call to this function to the Click event for the “Second Target Sequence” button.



Configuring the final orbit

Now that the satellite is in a circular orbit, propagate it for a day.

1. Add a propagate segment called “Outer Orbit” to the end of the MCS.
2. Change the segment color to green, the propagator to “Earth Point Mass”, and the trip value on the duration stopping condition to 1 day (86400 sec).
3. Add a call to this function to the Click event for the “Final Orbit” button.



Running the MCS

Now that the MCS is configured properly, run it and have the target sequences find the correct values for the maneuvers. Once the run is completed, the final values of the maneuvers' magnitudes can be reported. The values of the maneuvers are written into the Output window in the Visual Studio IDE. Just like in the Astrogator GUI, the values that the search profiles find are not immediately applied to the segments.

1. Add the RunMCS method, which will run the current MCS.

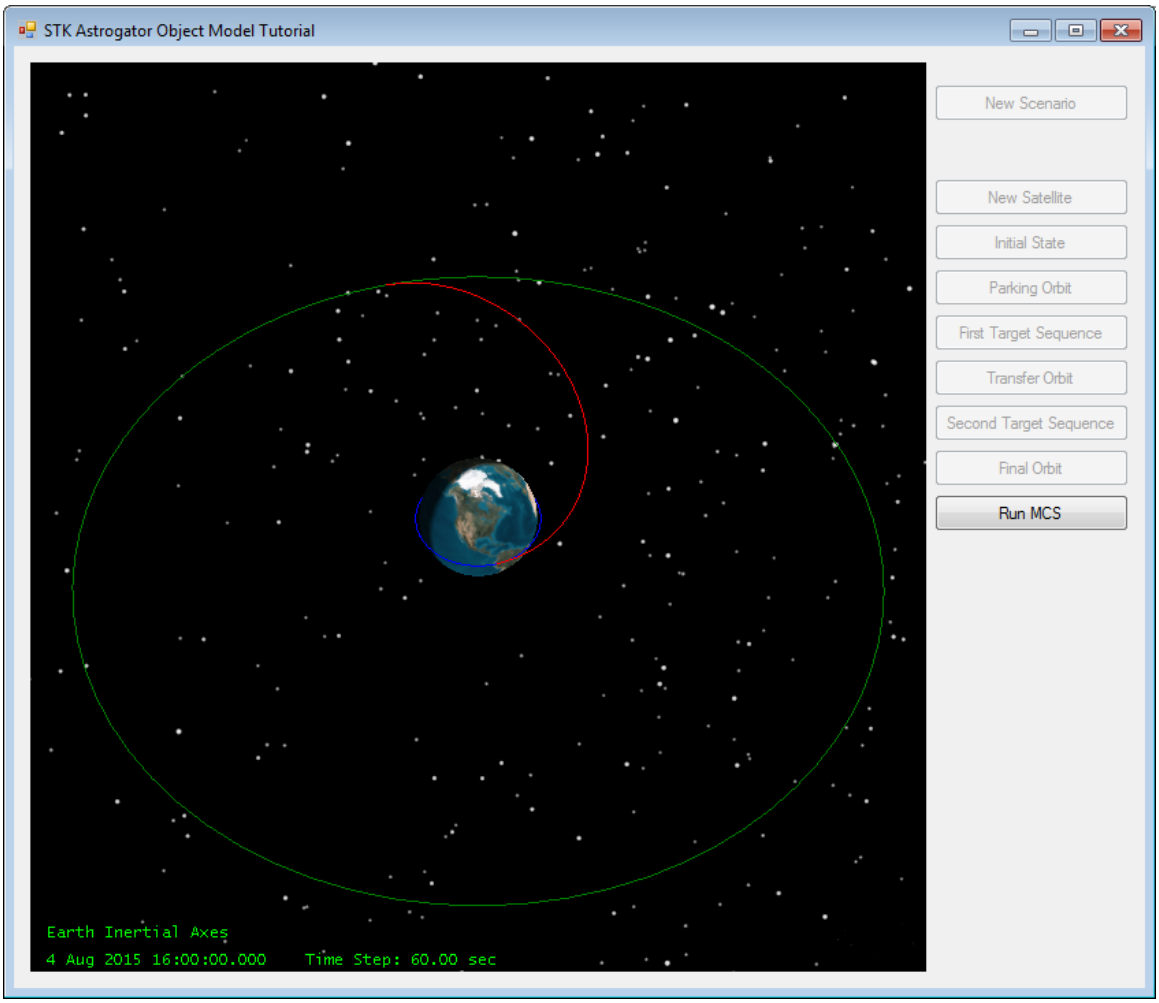
2. Get handles to the two target sequences.

3. Get handles to the differential correctors in these target sequences and write out the magnitude of the Delta-V found by the differential correctors.

4. Get handles to the two maneuvers and write out the current value for the Delta-Vs of each.

5. Apply the profiles and write out the current values for the Delta-Vs of each maneuver. They should match the solved-for values.

6. Add a call to this function to the Click event for the "Run MCS" button.



You can now run the application to see the results in the 3D area of the window.