Deepspace Interplanetary Navigation Operations Colorado Research EXplorer (DINO C-REx)

DINO C-REx Technical Memorandum

Document ID: DINO_C-REx-Image Generation

SYSTEMS ENGINEERING REPORT 4.10: CAMERA MODULE USER GUIDE

Prepared by	Matt Muszynski

Status: Initial Version
Scope/Contents
User documentation for the DINO C-REx Camera Module.

Rev:	**Change Description**	**By**
1.0	Initial Release	Matt Muszynski

Contents

1 Overview

This SER is intended as the user manual for the DINO C-REx camera module. Reading this document should serve as the minimal preparation a user needs before being able to effectively use the model. This document focuses on using the model in Python without a particular propagator.

2 Directory Structure

Within the camera module, there are several subdirectories containing the code, data files, and documentation needed to run the module.

2.1 root

Along with several subdirectories, the root of the camera module (as opposed to the DINO C-REx or Basilisk root) contains demo.py.. demo.py is an out of the box running demo of the camera module that creates a handful of images demonstrating the capability of the module. It should also serve as an effective tutorial for how to set use the camera module. As it demonstrates all functionality of the module, it can be used as a template for any work the user wishes to do with it.

2.2 db

The db directory contains the stellar database for DINO C-REx (tycho.db)[1] as well as two files that should be helpful should the database need to be changed in the future. addComputedTemperature.py and addSolidAngleSubtended.py are utilities for organizing editing stellar data. Both may be re-run at any time to update the computed_temperature and reduction_term columns in tycho.db. These are values are computed at runtime, and the scripts are included to allow developers to implement different stellar models in the future if desired. Both scripts serve as an example of how to add the necessary data to the database.

2.3 _UnitTest

Contains camera_test.py and supporting materials for running tests of camera module. Run pytest in this directory to run all DINO C-REx camera module unit tests.

2.4 doc

Contains all LaTex documents (SERs).

2.5 dependencies

Dependencies contains all .py files needed to run demo.py (or any camera module script). It contains the following files and directories:

2.5.1 camera.py

camera.py contains all main functionality of the camera module, including the object model. See SER 4.8 for more detail.

2.5.2 lightSimFunctions.py

Contains all functions related to DINO C-REx beacon lighting model simulations. See SER 4.9.

2.5.3 adcs.py

Contains utility functions for computing attitudes. Most used from this library is Euler321_2DCM.

[1] Described in detail in SER 4.12

2.5.4 orbits.py

Contains utility functions for computing orbits. This file was borrowed from another project, and as such has many functions that are not used by DINO C-REx.

2.5.5 em.py

Contains functions pertaining to electromagnetic phenomena (Planck, Stephan Boltzmann, etc)

2.5.6 constants.py

Contains standard constants used by module.

2.5.7 bodies

Contains definition of body and spacecraft object, both of which are required to run the full suite of camera module functionality. Also contains stock information about Earth, Moon, and several Planets. Other beacons must be added by the user.

2.5.8 util.py

Contains several helper functions, some of which are not used by DINO C-REx.

2.5.9 latex

Directory containing all LaTeX documentation generated By Doxygen.

2.5.10 html

Directory containing all HTML documentation generated by Doxygen,

2.6 tc

Contains out-of-the box transmission curves as described in SER 4.3b.

2.7 qe

Contains out-of-the box quantum efficiency curves as described in SER 4.3b.

3 Creating an Image

Before creating an image there are several setup functions that must be accomplished. This portion of the document follows demo.py and adds extra qualitative descriptions.

3.0.1 Imports

demo.py begins by importing several necessary libraries. camera is imported so we have access to the DINO C-REx camera object model, bodies is imported because it has information on bodies that will be used to add beacons to images, Euler321_DCM is imported from ADCS to facilitate pointing a spacecraft, and au is imported from constants to facilitate pupping planets in the right places in space. Other tools from matplotlib, numpy, and pdb are also imported and should be self explanatory to experienced python users.

3.0.2 Transmission and QE Curves

demo.py loads transmission and quantum efficiency curves from the files tc/20D.npz and qe.ACS.npz. Descriptions of those files can be found in SER 4.3b. Any transmission and quantum efficiency curves can be used by the camera model, but they must be formatted as described in SER 4.3.

3.0.3 Initializing beacons

A late addition to the DINO C-REx camera model is the beacon class. This is a simple class used to hold data on planetary bodies. demo.py initializes two beacons (the earth and moon) with the minimum attributes necessary to successfully run the model, r_eq, id, albedo, and state.

3.0.4 Spacecraft Info

The Camera object also requires information about the spacecraft that it rides on. The variables scState and scDCM hold the position of the spacecraft (in meters and in HCI) and the intertial to body-frame direction cosine matrix respctively. Both must be passed to the camera at initialization in order to work. They are held in the attributes camera.scState and camera.scDCM, where they can be updated as the simulation progresses.

3.0.5 The debug message

The python dictionary msg is gives the user a knob to turn to change parameters of the scenario. Most entries are booleans that provide switches for the user to turn on and off functionality for debugging purposes. It is called when initializing a camera object as a keyword. When not present, all camera features will be turned on. All entries are described below:

- addStars: Boolean switch determining if stars will be added to the camera object and then to images that are part of it. Notice that the object cam does not have stars added, but starCam does. changing the addStars message after the camera cam is made will not effect images taken by cam as stars are added at initialization. By the time the message is changed on line 135, the fact that cam has no stars cannot be changed.
- rmOcc: Boolean switch determining if stars and beacon facets occuted by beacons will be removed. Should always be set to 1 unless debugging.
- addBod: Boolean switch determining if beacon facets should be added to images. Should always be set to 1 unless debugging.
- psf: Boolean switch determining if point spread function should be applied to images. Should always be set to 1 unless debugging.
- raster: Boolean switch determining if rasterization should be applied to scenes. Should always be set to 1 unless debugging.
- photon: Boolean switch determining if photon noise should be added to images. Should always be set to 1 unless debugging.
- dark: Boolean switch determining if dark current should be added to scenes. Should always be set to 1 unless debugging.
- read: Boolean switch determining if read noise should be added to images. Should always be set to 1 unless debugging.
- hotDark: Boolean switch determining whether to apply hot and dark pixels as described in SER 4.4.
- verbose: Boolean switch determining whether to keep debug data in image and scene objects. verbose defaults to zero, and should be set to zero if making many images. Setting verbose to 1 allows inspection of many intermediary data products, facilitating debugging

3.0.6 Camera Initialization

Camera initialization is somewhat self explanatory. demo.py has commends describing the parameters as they are entered. Parameters are further described in the comments of camera.py and in Doxygen comments. Two cameras are initialized in demo.py because some images include stars while others do not.

3.0.7 Image 1

Image 1 in demo.py is created by placing the earth and moon in space, setting the attitude of the spacecraft such that the camera can see the beacons, and taking the images. Each time cam.updateState() is run while cam.takeImage = 1 will add one scene to the image. When cam.updateState() is run with cam.takeImage = 0, the image is closed. As such, image 1 has only one frame. Because it has only one frame, spacecraft location, body location, and spacecraft attitude are all static.

3.0.8 Image 7

Since all other images made by demo.py are fairly similar to image one, this document skips next to image 9, which illustrates a simple slew during the exposure. Here, starCam.updateState() is run 10 times while cam.takeImage = 1, meaning 10 scenes will be added to the image. Between each scene, the spacecraft slews by 0.1 degrees, for a total of 1 degree through the entire image.

3.0.9 Image 8

Image 8 is similar to image 7, but includes two blurred beacons.

4 Module Output

As can be seen in the imshow that accompanies each image in demo.py, the main output of the module is a detector array, a list of the number of electrons read off of the detector. The data for the detector array is held in the image object that it refers to. To access the array, the user should use starCam.images[i].detectorArray, where i is the index of the image the user wants to see. There is also further debugging information in starCam.images[i] and starCam.images[i].scenes[j] that is documented thoroughly in the Doxygen documentation.