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

DINO C-REx Technical Memorandum

Document ID: DINO_C-REx-Image Generation

CAMERA OBJECT UNIT TESTS

Prepared by	Matt Muszynski

Status: Initial Version
Scope/Contents
Description of unit tests completed for the DINO C-REx camera object. This includes tests 4.18 through 4.25.

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

Contents

1 Overview

This document describes all unit tests written for the light sim functions library by the DINO C-REx camera model team. Each function in the library has an associated test, and additional integrated tests are performed where deemed useful by the team. Each test has an entry in this document for status (including date of status change), responsible DINO C-REx team member, a qualitative description of the test performed and a brief technical discussion of the method employed in code to achieve the test.

2 Tests

2.1 Test 4.20: lightSimFunctions.checkFOV()

Status: Passes. 11.30.17
Responsible Team Member: Matt Muszynski
Description: This test creates an image with several bodies in and out of the FOV and checks that they are handled appropriately. Some objects should be completely in view, some completely out of view, and still some should have their centers out of view but other portions in view.
Method: Four images are made to test lightSimFunctions.checkFOV(). In each, the Earth is placed at [1 au, 0, 0] in inertial space and the spacecraft points in the [1,0,0] direction. The location of the spacecraft is varied in order to place the image of the earth at the desired places on the focal plane. Plots illustrating each can be seen in fig. **??** Details for each follow:

- Spacecraft is placed at [1 au - 1e5 km, 0, 0]. This puts the center of the earth in the [1,0,0] direction. Because this is the same direction that the spacecraft is pointing in, the Earth appears in the center of the frame. Pytest asserts that pixel and line positions for earth facets are indeed being created.

- Spacecraft is placed at [1 au , - 1e5 km, 0]. This puts the center of the earth in the [0,1,0] direction, or 90° degrees away from the camera boresight. Because the camera's FOV is only 22°, this is far ouf of the FOV, and the Earth should not appear in the image at all. Pytest asserts that no pixel and line positions for earth facets are not being created.

- Spacecraft is placed at [1 au - 1e5 km, - 2.3e4 km, 0], placing the center of the earth in the direction [0.97455519, 0.22414769, 0.], about 13.4° from the camera boresight. Because the camera half field of view is only 11°, this places the center of the Earth outside the vield of view. However, because half of the earth's angular size at this distance is 3.5°, part of the body still appears on the focal plane. Pytest asserts that pixel coordinates for some beacon facets are being made as they should be.

- Spacecraft is placed at [1 au - 1e5 km, 0, - 2.3e4 km], placing the center of the earth in the direction [0.97455519, 0, 0.22414769], about 13.4° from the camera boresight. Because the camera half field of view is only 11°, this places the center of the Earth outside the vield of view. However, because half of the earth's angular size at this distance is 3.5°, part of the body still appears on the focal plane. Pytest asserts that pixel coordinates for some beacon facets are being made as they should be.

2.2 Test 4.21: lightSimFunctions.mapSphere()

Status: Passes. 12.2.17
Responsible Team Member: Matt Muszynski
Description: This test checks that the sum of surface area facets computed is equal to half the surface area of a sphere of the same size.

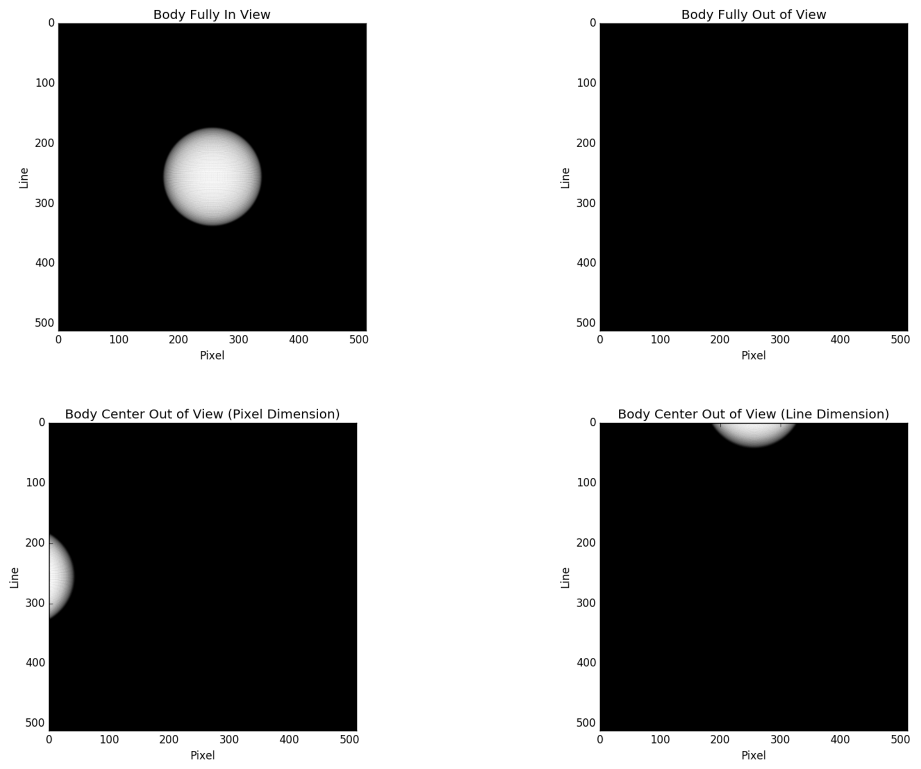

Fig. 1: Clockwise from the top left: image fully in FOV, image fully out of FOV, image partially out of FOV in line direction, image partially out of FOV in pixel direction.

Method: The test itself calculates 3 different resolutions of body facets for the Earth at, 100x100, 250x250, and 500x500. It asserts that the sum of the facetArea array that is returned by lightSimFunctions.mapSphere() is given by the following within machine precision:

$$A_{Earth} = \frac{4\pi r_{earth}^2}{2} \tag{1}$$

2.3 Test 4.22: lightSimFunctions.mapSphere()

Status: Passes. 12.2.17
Responsible Team Member: Matt Muszynski
Description: This test checks that the longitude and latitude coordinates computed are symmetric about the body's equator.
Method: There are two sets of assertions for the test. Each checks the same three mappings that are created for test 4.21. The first takes each set of latitiudes as a list, flips it, and subtracts it from the original list. The assertion checks that the sum of the result array is equal to zero within machine precision. The second set of asserts does a similar procedure of flipping arrays, but instead adds them. The longitude entries are symmetric if all entries in the sum are 180. The assert shows that they are within machine precision.

2.4 Test 4.23: lightSimFunctions.lumos()

Status: Passes. 12.3.17
Responsible Team Member: Matt Muszynski
Description: This test ensures that the sum of the facetArea array is exactly equal to the the projected

surface area of the body.
Method: This tests whether the projected area of body facets is being computed correctly by comparing the total surface area (the returned value facetArea times the number of facets returned in ptsHelio and fluxDecayNet) with the analytic solution (πr_{earth}^2). The analytic computation for projected surface area follows:

$$\int_{surface} \cos\psi dA \tag{2}$$

Where dA is an infinitesimal area element and ψ is the angle between vector normal to dA ($\hat{n}$) and the vector pointing from the dA to the sun ($\hat{s}$). $\cos\psi$ is then simply the dot product of the two:

$$\int_{surface} \hat{n} \cdot \hat{s} dA \tag{3}$$

Because we place the earth at 1AU and the observer at [0,0,0], $\hat{s} = \hat{x}$, making the integral:

$$\int_{surface} n_x dA \tag{4}$$

And because we are modeling bodies as spheres, n_x is simply the x coordinate of the unit vector from the center of the earth to dA

$$\int_{surface} \sin\phi \cos\theta dA \tag{5}$$

Changing variables to get dA in terms of ϕ and θ:

$$\int_{surface} \sin\phi \cos\theta dA \tag{6}$$

$$\int_{surface} \sin\phi \cos\theta r_{earth}^2 \sin\theta d\theta d\phi \tag{7}$$

And integrating over only the illuminated portion of the sphere:

$$\int_0^{\pi} \int_{-\pi/2}^{\pi/2} \sin\phi \cos\theta r_{earth}^2 \sin\phi d\theta d\phi \tag{8}$$

$$r_{earth}^2 \int_0^{\pi} \int_{-\pi/2}^{\pi/2} \sin^2\phi \cos\theta d\theta d\phi \tag{9}$$

$$r_{earth}^2 \int_0^{\pi} \sin^2\phi \int_{-\pi/2}^{\pi/2} \cos\theta d\theta \tag{10}$$

$$r_{earth}^2 (\pi/2)(2) \tag{11}$$

So the projected area of the sphere is simply the area of a circle.

$$\pi r_{earth}^2 \tag{12}$$

2.5 Test 4.24: lightSimFunctions.lumos()

Status: Incomplete. 10.31.17
Responsible Team Member: Matt Muszynski
Description: This test compares the fluxDecayNet array to a verified analytic solution.
Method:

2.6 Test 4.25: lightSimFunctions.project2CamView()

Status: Passes. 10.31.17
Responsible Team Member: Matt Muszynski
Description: Compare the output variable facetAreaCamview with the area of a circle with the same radius as the spherical body.
Method: This test follows the same derivation as test 4.23. Because writing the test was easier that way, it checks the output of lightSimFunctions.lightSim() directly. This is equivalent to checking lightSimFunctions.project2CamView() as the relevant output from lightSimFunctions.project2CamView() is passed directly as an output to lightSimFunctions.lightSim(). Once again, the sum of facetAreaCamview is checked against πr^2. Asserts that the values match to within 0.2%.