JSC-61777-models/environment/earth_lighting

JSC Engineering Orbital Dynamics Low Earth Orbit Lighting Model

Simulation and Graphics Branch (ER7)
Software, Robotics, and Simulation Division
Engineering Directorate

Package Release JEOD v5.0

Document Revision 1.1
July 2022

National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas

JSC-61777-models/environment/earth_lighting

JSC Engineering Orbital Dynamics
Low Earth Orbit Lighting Model

Document Revision 1.1
July 2022

Andrew Spencer

Simulation and Graphics Branch (ER7)
Software, Robotics, and Simulation Division
Engineering Directorate

National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas

Abstract

In the analysis of complex dynamic systems, such as the rendezvous of two spacecraft orbiting the Earth, visual simulation can be an extremelly useful tool. To give the imagery a sense of realism, the ambient lighting experienced by the spacecraft must be estimated in some form. The Low Earth Orbit Lighting Model computes the lighting conditions experienced by a vehicle, for a specific time, traveling in low earth orbit. The Low Earth Orbit Lighting Model calculates these effects using the current ephemeris information for the Sun, Earth and Moon and determines if the lighting body is partially or completely occluded. It then calculates a lighting fraction parameter using the occlusion fraction and the current phase of the body. It also performs a very crude approximation of Earth-Albedo effects; however, this calculation should not be used where accurate knowledge of the Earth-Albedo effects are desired.

Contents

1 Introduction **1**

1.1 Purpose and Objectives of The Low Earth Orbit Lighting Model . . . 1

1.2 Document History . . . 2

1.3 Documentation Organization . . . 2

2 Product Requirements **4**

2.1 Data Requirements . . . 4

2.2 Functional Requirements . . . 5

3 Product Specification **7**

3.1 Conceptual Design . . . 7

3.2 Mathematical Formulations . . . 7

3.2.1 Body Visibility . . . 7

3.2.2 Lighting Effects Calculation . . . 9

3.3 Detailed Design . . . 9

3.3.1 Class Design . . . 9

3.3.2 EarthLighting . . . 10

3.4 Inventory . . . 11

4 User Guide **15**

4.1 Analysis . . . 15

4.2 Integration . . . 16

4.3 Extension . . . 17

5 Verification and Validation **18**

5.1 Verification . . . 18

5.2 Validation 18
5.3 Metrics 24
    5.3.1 Code Metrics 24
5.4 Requirements Traceability 25

Chapter 1

Introduction

1.1 Purpose and Objectives of The Low Earth Orbit Lighting Model

Simulation is an important tool in the analysis of complex dynamical systems such as the rendezvous of two spacecraft orbiting the Earth. Computer animations of these simulations provide an aid for visualizing and demonstrating the system behaviors. The imagery portrayed by the computer animated graphics are generated by simulating cameras aimed at the simulated vehicles. Light from various sources illuminates the vehicles, making the scene visible to the cameras.

A vehicle in low Earth orbit is subject to three light sources: the sun, sunlight reflected off of the Earth, and sunlight reflected off of the moon. The amount of light impinging on a spacecraft from these light sources varies over time, due to the following factors:

- The intensity of the sunlight reflected by the Earth or moon varies with the phase of the Earth or moon.

- The Earth may obscure either of the two more distant light sources from view.

The purpose of the Low Earth Orbit Lighting Model is to model phase variations and obscuration effects for these three bodies, thereby providing a measure of realism for the graphical simulations. The Low Earth Orbit Lighting Model accomplishes this by interacting with a JEOD Dynamics Manager object [2] , leveraging information from the manager to track the current relative positions of the sun, Earth and moon. A chart of the information flow for the Low Earth Orbit Lighting Model is shown in Figure 1.1.

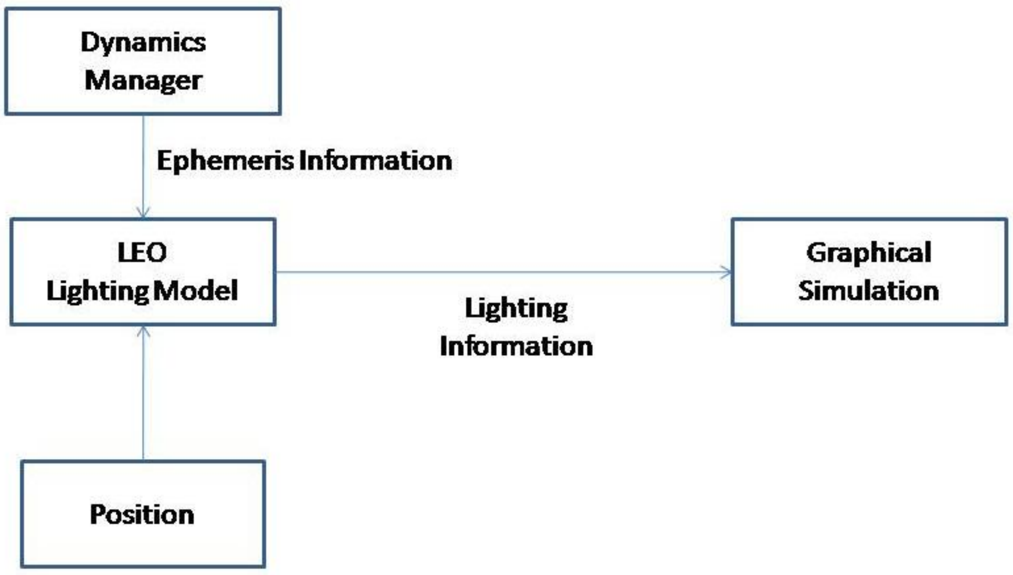

Figure 1.1: A Flowchart of Data for the Low Earth Orbit Lighting Model

1.2 Document History

Author	Date	Revision	Description
Andrew Spencer	November, 2009	1.0	Initial Version
Andrew Spencer	October, 2010	1.1	Added metrics

This document derives heavily from it's predecessor, JSC Engineering Orbital Dynamics Low Earth Orbit Lighting Model, released with JEOD v1.5.2.

The following document is parent to this document:

- *JSC Engineering Orbital Dynamics* [4]

1.3 Documentation Organization

This document is formatted in accordance with the NASA Software Engineering Requirements Standard [5] and is organized into the following chapters:

Chapter 1: Introduction - This introduction contains four sections; objective and purpose, contex within JEOD, status, and organization. The section titled objective and purpose provides the true introduction to the Low Earth Orbit Lighting Model and its reason for existance. The next section identifies the context within JEOD of this document and the Low Earth Orbit Lighting Model. This is achieved with a directory diagram, a brief description of the interconnections with other models, and references to any supporting documents. This is followed by the status of the document which includes author, date, and reason for each revision. Finally, there is a description of the how the document is organized.

Chapter 2: Product Requirements - Describes requirements for the Low Earth Orbit Lighting Model.

Chapter 3: Product Specification - Describes the underlying theory, architecture, and design of the Low Earth Orbit Lighting Model in detail. It will be organized in three sections; Conceptual Design, Mathematical Formulations, and Detailed Design.

Chapter 4: User Guide - Describes how to use the Low Earth Orbit Lighting Model in a Trick simulation. It is broken into three sections to represent the JEOD defined user types; Analysts or users of simulations (Analysis), Integrators or developers of simulations (Integration), and Model Extenders (Extension).

Chapter 5: Verification and Validation - Contains Low Earth Orbit Lighting Model verification and validation procedures and results.

Chapter 2

Product Requirements

This chapter identifies the requirements for the Low Earth Orbit Lighting Model.

Requirement earthlighting_1: Top-level requirement

Requirement:
This model shall meet the JEOD project requirements specified in the JEOD v5.0 top-level document.

Rationale:
This model shall, at a minimum, meet all external and internal requirements applied to the JEOD v5.0 release.

Verification:
Inspection

2.1 Data Requirements

This section identifies requirements on the data represented by the Low Earth Orbit Lighting Model. These as-built requirements are based on the Low Earth Orbit Lighting Model data definition header files.

Requirement earthlighting_2: Vehicle State Encapsulation

Requirement:
The Low Earth Orbit Lighting Model shall encapsulate the descriptions of the sun, Earth, and moon as seen by a spacecraft and the extent to which these bodies illuminate the spacecraft in a single data object.

Rationale:
The overarching purpose of the Low Earth Orbit Lighting Model is to model the lighting pro-

vided by the sun, Earth, and moon for use by simulation graphics models. This requirement constrains the design of the module data.

Verification:
Inspection

2.2 Functional Requirements

This section identifies requirements on the functional capabilities provided by the Low Earth Orbit Lighting Model. These as-built requirements are based on the Low Earth Orbit Lighting Model source files.

Requirement earthlighting_3: Summarize Lighting Bodies

Requirement:
For each lighting body (the sun, Earth, and moon) that illuminates a spacecraft in low Earth orbit, the Low Earth Orbit Lighting Model shall compute:

3.1 Relative Position
The position of the lighting body relative to the spacecraft.

3.2 Apparent Size
The angular size of the lighting body as seen from the spacecraft.

Rationale:
These calculations are needed by the animated graphics models.

Verification:
Inspection, Test

Requirement earthlighting_4: Calculate Illumination

Requirement:
For each lighting body that illuminates a spacecraft in LEO, the Low Earth Orbit Lighting Model shall compute:

4.1 Occlusion The fraction of the hemisphere of each remote lighting body (the Sun and Moon) facing a spacecraft in low Earth orbit that is occluded by the Earth as observed by the spacecraft.

4.2 Lighting The amount of illumination provided by each lighting body (the Sun, Earth, and Moon) illuminating the spacecraft as a fraction of the maximum light provided by that body.

Rationale:
These calculations are needed by the animated graphics models.

Verification:

Inspection, Test

Chapter 3

Product Specification

This chapter defines the conceptual design, the mathematical formulations, and the detailed design for the Low Earth Orbit Lighting Model.

3.1 Conceptual Design

Purpose and Scope

The Low Earth Orbit Lighting Model is designed to compute the lighting environment that a spacecraft in Earth orbit is experiencing. It takes in ephemeris information on the system that the spacecraft is operating in (planetary locations, light intensity, etc.), as well as information about the current state of the spacecraft, and uses this information to calculate the lighting environment.

Goals and Objectives

The goal of the Low Earth Orbit Lighting Model is to compute the lighting effects that the spacecraft sees.

3.2 Mathematical Formulations

The Low Earth Orbit Lighting Model relies on classical geometric formulations in order to compute the lighting effects on a body. It assumes that the only bodies that are effecting the lighting environment are the Earth, sun, and moon. In order to determine what effect these bodies have on the lighting environment, the spacecraft must first determine if it can "see" the body. In other words, is the body being occluded by the Earth.

3.2.1 Body Visibility

For each of the lighting bodies in question (sun, moon and Earth), the software has to determine if the body is completely visible, partially occluded, or completely blocked by the primary body

(Earth). An example of these conditions is shown below in Figure 3.1.

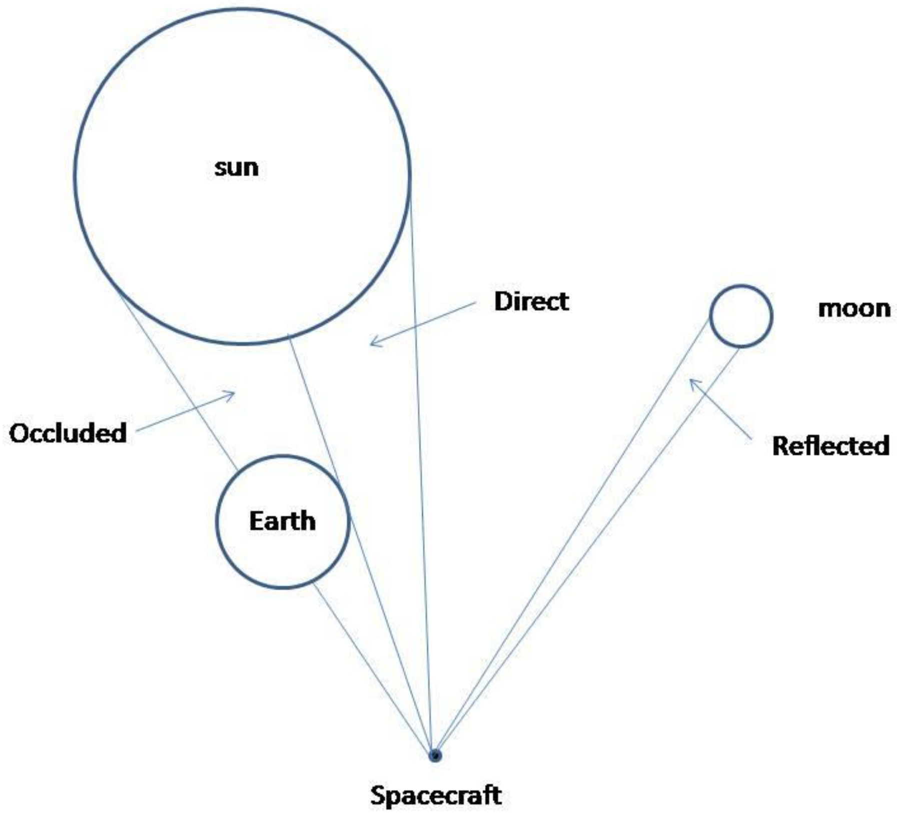

Figure 3.1: Examples of visible, partially occluded, and reflected light. Not to scale.

The software first calculates the angular size of the lighting body disc in the sky. In order to do this, it uses the arcsine of the equatorial radius over the distance from the spacecraft to the lighting body. This is then stored as the half-angle for the lighting body.

The software then calculates the body-referenced inertial position vectors from the spacecraft to the Earth, moon, and sun. It then uses this information to calculate the angles between the spacecraft-Earth vector and the spacecraft sun and moon vectors.

Using all of this information, the software then calculates whether the body is completely visible, partially occluded, or completely blocked from view.

Circle Intersection

The software essentially treats the spacecrafts's field of view like a unit sphere and calculates arc-lengths using this unit sphere. The three arc-lengths of interest are sent into the circle-intersection function (arc-length of the Earth-disc half-angle, arc-length of the lighting body half-angle, and arc- length of the angle between the spacecraft-Earth and spacecraft-lighting body vectors). Using these arc-lengths, the software determines what, if any, the intersections between the lighting body disc and the Earth-disc there are. The software then sends this information out to the lighting calculation software.

3.2.2 Lighting Effects Calculation

With the information on the percentage of lighting body occlusion, the software is then capable of calculating what the lighting effects of that particular lighting body are.

The software takes the shared area of the two discs and divides this value by the total area of the lighting body disc to calculate the percentage of the disc that is occluded by the Earth. The software then subtracts this from 1 to determine the fraction of the lighting body that is visible. The final lighting condition is then a phase fraction multiplied by the visible fraction to simulate the phases of the moon.

3.3 Detailed Design

The Low Earth Orbit Lighting Model uses three classes to accomplish its objectives. This section will describe these classes in detail.

3.3.1 Class Design

The *Earth_lighting.h* file contains all of the classes that are used by the Low Earth Orbit Lighting Model source code. These classes are described in the following sections.

LightingBody

The class LightingBody contains the information on the parameters of a lighting body. These parameters are:

- radius: Celestial body mean equitorial radius, in meters,
- position: A three vector of the inertial position of the lighting body, relative to the observer, in meters,
- position: The distance from the observer to the lighting body, in meters,
- half_angle: The Apparent half angle of the body disk, in radians.

The LightingBody class only contains a default constructor and a destructor, and has no other member functions.

LightingParams

The class LightingParams contains the information on how the lighting body is affecting the spacecraft. This information is represented by the following parameters:

- obs_angle: The apparent observation angle from the light source, in radians,
- phase: The apparent lighting phase of the planet,
- occlusion: Fraction of planetary surface occlusion,
- visible: Fraction of planetary surface visible,
- lighting Fraction of lighting (phase * visible).

The LightingParams class only contains a default constructor and a destructor, and has no other member functions.

3.3.2 EarthLighting

The EarthLighting class fully defines and calculates the lighting parameters seen by the vehicle of interest. This class contains the following parameters:

- active: A booleon flag indicating if the model is active or not,
- Earth: A pointer to a Planet object representing the Earth, obtained from the DynManager the class is initialized from,
- moon: A pointer to a Planet object representing the moon, obtained from the DynManager the class is initialized from,
- sun: A pointer to a Planet object representing the sun, obtained from the DynManager the class is initialized from,
- Earth_frame: A pointer to the Earth inertial frame, obtained from the DynManager the class is initialized from,
- moon_frame: A pointer to the moon inertial frame, obtained from the DynManager the class is initialized from,
- sun_frame: A pointer to the sun inertial frame, obtained from the DynManager the class is initialized from,
- sun_body: A LightingBody representing the sun stellar parameters.
- Earth_body: A LightingBody representing the Earth stellar parameters,
- moon_body: A LightingBody representing the moon stellar parameters,
- sun_Earth: A LightingParams representing the lighting of the sun with respect to the vehicle orbiting the Earth,

- moon_Earth: A LightingParams representing the lighting of the moon with respect to the vehicle orbiting the Earth,
- Earth_albedo: A LightingParams representing the effects of the Earth albedo on the vehicle.

In addition to the default constructor and the destructor, the EarthLighting class also contains the following member functions.

initialize This member function initializes the internal member variables from the supplied DynManager object.

- return: void - no return
- IN: DynManager& manager: The DynManager object that contains the ephemeris for the sun, Earth and moon.

circle_intersect This member function calculates the area of an intersection between two circles, based on the supplied input parameters.

- return: int - Returns 0 for no intersection, 1 for intersection
- IN: double r_bottom - Radius of the obscured circle.
- IN: double r_top - Radius of the obscuring circle.
- IN: double d_centers - Distance between the centers of the circles.
- OUT: double * area - Covered area (the intersection between the circles).

calc_lighting This member function calculates the lighting parameters on the vehicle at the supplied vehicle position and stores them in the internal EarthLighting member variables.

- return: void - none
- IN: double pos_veh[3] - The position of the vehicle of interest, with respect to the Earth Centered Inertial (ECI) reference frame

Further information about the design of this model can be found in the *Reference Manual* [1].

3.4 Inventory

All Low Earth Orbit Lighting Model files are located in the directory `${JEOD_HOME}/models/environment/earth_lighting`. Relative to this directory,

- Header and source files are located in the model `include` and `src` subdirectories. Table 3.1 lists the configuration-managed files in these directories.

- Documentation files are located in the model `docs` subdirectory. See table 3.2 for a listing of the configuration-managed files in this directory.
- Verification files are located in the model `verif` subdirectory. See table 3.3 for a listing of the configuration-managed files in this directory.

Table 3.1: Source Files

File Name
include/class_declarations.hh
include/earth_lighting.hh
include/earth_lighting_messages.hh
src/earth_lighting.cc
src/earth_lighting_messages.cc

Table 3.2: Documentation Files

File Name
docs/earth_lighting.pdf
docs/refman.pdf
docs/tex/earth_lighting.bib
docs/tex/earth_lighting.sty
docs/tex/earth_lighting.tex
docs/tex/earth_lightingAbstract.tex
docs/tex/earth_lightingChapters.tex
docs/tex/makefile
docs/tex/model_name.mk
docs/tex/pics/circle_area1.jpg
docs/tex/pics/circle_area2.jpg
docs/tex/pics/flowchart.jpg
docs/tex/pics/lighting_pic.jpg

Table 3.3: Verification Files

File Name
verif/SIM_LIGHT_CIR/S_define
verif/SIM_LIGHT_CIR/S_overrides.mk
verif/SIM_LIGHT_CIR/DP_Product/DP_validation.xml
verif/SIM_LIGHT_CIR/Log_data/LIGHT_verif_rec.py

Continued on next page

Table 3.3: Verification Files (continued from previous page)

File Name
verif/SIM_LIGHT_CIR/Modified_data/default_mods.py
verif/SIM_LIGHT_CIR/SET_test/RUN_T01_LIGHT_VER/input.py
verif/SIM_LIGHT_CIR/SET_test/RUN_T02_LIGHT_VER/input.py
verif/SIM_LIGHT_CIR/SET_test/RUN_T03_LIGHT_VER/input.py
verif/SIM_LIGHT_CIR/SET_test/RUN_T04_LIGHT_VER/input.py
verif/SIM_LIGHT_CIR/SET_test/RUN_T05_LIGHT_VER/input.py
verif/SIM_LIGHT_CIR/SET_test/RUN_T06_LIGHT_VER/input.py
verif/SIM_LIGHT_CIR/SET_test/RUN_T07_LIGHT_VER/input.py
verif/SIM_LIGHT_CIR/SET_test/RUN_T08_LIGHT_VER/input.py
verif/SIM_LIGHT_CIR/SET_test/RUN_T09_LIGHT_VER/input.py
verif/SIM_LIGHT_CIR/SET_test/RUN_T10_LIGHT_VER/input.py
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T01_LIGHT_VER/input.py
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T01_LIGHT_VER/log_light_verif.header
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T01_LIGHT_VER/log_light_verif.trk
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T02_LIGHT_VER/input.py
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T02_LIGHT_VER/log_light_verif.header
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T02_LIGHT_VER/log_light_verif.trk
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T03_LIGHT_VER/input.py
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T03_LIGHT_VER/log_light_verif.header
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T03_LIGHT_VER/log_light_verif.trk
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T04_LIGHT_VER/input.py
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T04_LIGHT_VER/log_light_verif.header
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T04_LIGHT_VER/log_light_verif.trk
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T05_LIGHT_VER/input.py
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T05_LIGHT_VER/log_light_verif.header
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T05_LIGHT_VER/log_light_verif.trk
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T06_LIGHT_VER/input.py

Continued on next page

Table 3.3: Verification Files (continued from previous page)

File Name
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T06_LIGHT_VER/log_light_verif.header
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T06_LIGHT_VER/log_light_verif.trk
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T07_LIGHT_VER/input.py
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T07_LIGHT_VER/log_light_verif.header
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T07_LIGHT_VER/log_light_verif.trk
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T08_LIGHT_VER/input.py
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T08_LIGHT_VER/log_light_verif.header
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T08_LIGHT_VER/log_light_verif.trk
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T09_LIGHT_VER/input.py
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T09_LIGHT_VER/log_light_verif.header
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T09_LIGHT_VER/log_light_verif.trk
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T10_LIGHT_VER/input.py
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T10_LIGHT_VER/log_light_verif.header
verif/SIM_LIGHT_CIR/SET_test_val/RUN_T10_LIGHT_VER/log_light_verif.trk
verif/include/lighting_ephem.hh
verif/include/lighting_sim_verif.hh
verif/src/lighting_ephem.cc
verif/src/lighting_sim_verif.cc

Chapter 4

User Guide

The Analysis section of the user guide is intended primarily for users of pre-existing simulations. It contains:

- A description of how to modify Low Earth Orbit Lighting Model variables after the simulation has compiled, including an in-depth discussion of the input file,
- An overview of how to interpret (but not edit) the S_define file,
- A sample of some of the typical variables that may be logged.

The Integration section of the user guide is intended for simulation developers. It describes the necessary configuration of the Low Earth Orbit Lighting Model within an S_define file, and the creation of standard run directories. The Integration section assumes a thorough understanding of the preceding Analysis section of the user guide. Where applicable, the user may be directed to selected portions of Product Specification (Chapter 3).

The Extension section of the user guide is intended primarily for developers needing to extend the capability of the Low Earth Orbit Lighting Model. Such users should have a thorough understanding of how the model is used in the preceding Integration section, and of the model specification (described in Chapter 3).

Note that the Low Earth Orbit Lighting Model depends heavily on the Dynamics Manager (DynManager) class, and any simulation involving a Low Earth Orbit Lighting Model object will require the successful setup of a DynManager object [2].

4.1 Analysis

The Analysis and the Integration sections will assume, for the purposes of illustration, an S_define object of the following form:

```
sim_object {
   environment/Earth_lighting: EarthLighting lighting;

   double position[3]; /*spacecraft position*/


   P_ENV (initialization) environment/Earth_lighting:
      LIGHT.lighting.initialize(
      In DynManager& manager = mngr.dyn_manager);


   /* Call the lighting source code */
   (DYNAMICS, scheduled) environment/Earth_lighting:
      LIGHT.lighting.calc_lighting(
      In    double    pos_veh[3]   = &LIGHT.position[0]);

} LIGHT;
```

The S_define will also contain an accompanying sim object titled 'mngr' containing an appropriately named DynManager instantiation.

The input files for the Low Earth Orbit Lighting Model are straight-forward. The only information not directly pulled from the supplied DynManager object is the current phase of the moon and the sun, as seen from the Earth. In the above example code, these parameters can be set with the input file commands:

```
LIGHT.lighting.sun_Earth.phase = 1;
LIGHT.lighting.moon_Earth.phase = 1;
```

Note that the phase stated here simulates the fraction of the lighting body that is actually radiating light. In the moon's case this value can range from 0-1. In the case of the sun, this value should always be 1.

4.2 Integration

Integrating the Low Earth Orbit Lighting Model into an S_define is a simple process. An Earth-Lighting object must first be instantiated, as shown in the first lines of the example code in the Analysis section above.

Before the Low Earth Orbit Lighting Model can be used to calculate lighting characteristics, it must be initialized using a correctly instantiated and initialized DynManager object. The following code gives an example of this initialization:

```
P_ENV (initialization) environment/Earth_lighting:
   LIGHT.lighting.initialize(
   In DynManager& manager = mngr.dyn_manager);
```

Note that the DynManager object must be initialized before this function is called. Also, the Low Earth Orbit Lighting Model requires that the DynManager object used for initialization contains three Planet objects, with the following specific names: “Earth”, “Moon”, and “Sun”. If this requirement is not met, then the Low Earth Orbit Lighting Model initialization function will fail and produce an error message. Information on correctly initializing can be found in the DynManager documentation [2].

Finally, the specific lighting information can be calculated with the following call.

```
(DYNAMICS, scheduled) environment/Earth_lighting:
   LIGHT.lighting.calc_lighting(
   In    double    pos_veh[3]   = &LIGHT.position[0]);
```

The input is the current position of the vehicle in the ECI frame. For most simulations where the use of the Low Earth Orbit Lighting Model is appropriate, this will be the default integration frame, and the position can be obtained from the appropriate DynBody object [3]. However, if the integration frame is not the ECI frame, the correct position of the vehicle can be obtained using the JEOD reference frame utilities combined with the DynManager [2] object. This position can also be determined using a user define DerivedState object [6].

4.3 Extension

The current implemention of the Low Earth Orbit Lighting Model depends heavily on the assumption of low Earth orbit, and is not intended to be extendable to other space environments.

Chapter 5

Verification and Validation

5.1 Verification

Inspection earthlighting_1: Top-level inspection

This document structure, the code, and associated files have been inspected, and together satisfy requirement earthlighting_1.

Inspection earthlighting_2: Data conversion

By inspection, the data structures of the Low Earth Orbit Lighting Model completely satisfy requirement earthlighting_2.

5.2 Validation

In each test case, simulations were run with different input files. Each individual input file has its own associated run directory. These run directories are named with the convention RUN_T##_LIGHT_VER where the ## characters are replaced by a 2-digit number. These run directories are contained in the SET_test sub-directory of SIM_LIGHT_CIR.

Test earthlighting_1: Identical Circle Intersection

Purpose:

This test case is designed to examine the performance of the circle intersection calculation with two circles that have the same radius and are located adjacent to each other.
Run directory: RUN_T01_LIGHT_VER (Test 1)
RUN_T02_LIGHT_VER (Test 2)
RUN_T03_LIGHT_VER (Test 3)
RUN_T04_LIGHT_VER (Test 4)

RUN_T05_LIGHT_VER (Test 5)

Requirements: By passing this test, this model partially satisfies requirements earthlighting_3, and earthlighting_4.

Procedure:

To test the capabilities of the circle_intersect function included in the lighting model, a MATLAB script was written that will test the area calculation function by completely separate means. The x and y coordinates for two circles were generated using an array of angles and the sin and cos functions. The points of intersection for these two circles were then calculated analytically. Using this intersection information, a polygon was generated from the points forming the common area. The area of this polygon was calculated, using MATLAB's polyarea function and compared with the analytical output from the function. A plot of the circle intersection for the first set of parameters from Table 5.1 can be seen in Figure 5.1.

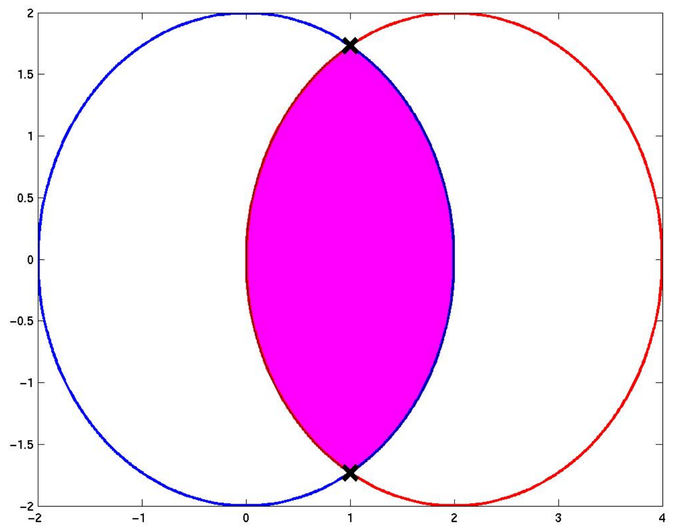

Figure 5.1: Circle Intersection Illustration

Results:

Table 5.1 shows the results of the various tests run for various circle parameters.

As this table shows, the model performed correctly in every case. The level of error seen in each sub-test is due to the discretization of the circles used in the MATLAB function. In the final two tests, the MATLAB function was not able to calculate the common area because there were no intersections between the circles. However, since in test 4 there was no common area, and in test 5, the circles were directly on top of each other, the correct value was easy to determine.

Test	r_{top} (m)	r_{bot} (m)	d_{cent} (m)	Model Output (m^2)	MATLAB (m^2)	Difference (m^2)
1	2.0	2.0	2.0	4.913	4.913	1.68E-7
2	3.0	3.0	1.0	22.302	22.302	4.77E-7
3	1.5	1.5	1.25	0.563	0.563	6.96E-8
4	0.25	0.25	1.0	0	0	0
5	4.0	4.0	0.0	50.266	50.266	0

Table 5.1: Table of Circle Intersection Parameters for First Test

Test earthlighting_2: Different Circle Intersection

Purpose:

This test case is designed to examine the performance of the circle intersection calculation with two circles that have the same radius and are located adjacent to each other.
Run directory: RUN_T06_LIGHT_VER (Test 6)
RUN_T07_LIGHT_VER (Test 7)
RUN_T08_LIGHT_VER (Test 8)
RUN_T09_LIGHT_VER (Test 9)
RUN_T10_LIGHT_VER (Test 10)

Requirements: By passing this test, this model partially satisfies requirements earthlighting_3, and earthlighting_4.

Procedure:

To test, the capabilities of the circle_intersect function included in the lighting model, a MATLAB script was written that tested the area calculation function by completely separate means. The x and y coordinates for two circles were generated using an array of angles and the sin and cos functions. The points of intersection for these two circles were then calculated analytically. Using this intersection information, a polygon was generated from the points forming the common area. The area of this polygon was calculated using MATLAB's polyarea function and compared with the analytical output from the function. A plot of the circle intersection for the first set of parameters from Table 5.2 can be seen in Figure 5.2.

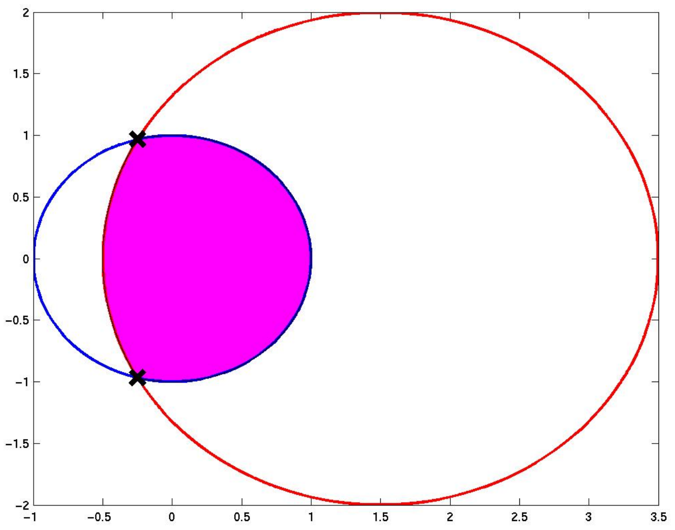

Figure 5.2: Circle Intersection Illustration for Different Radius

Results:

Table 5.2 shows the results of the various tests run for various circle parameters.

Test	r_{top} (m)	r_{bot} (m)	d_{cent} (m)	Output (m^2)	MATLAB (m^2)	Difference (m^2)
6	1.0	2.0	1.5	2.393	2.393	8.96E-8
7	3.0	1.5	2.0	6.084	6.084	2.76E-7
8	3.0	1.5	4.25	0.233	0.233	1.69E-7
9	3.0	1.5	1.75	6.693	6.693	1.93E-7
10	3.0	2.0	0.5	12.566	12.566	2.07E-7

Table 5.2: Table of Circle Intersection Parameters for Second Test

As this table shows, the model once again performed correctly even with circles of different radius as inputs. Note that in the final test, the bottom circle was completely occluded by the top circle. In this case, the MATLAB function did calculate the area correctly and so the discretization error is seen here again.

Test earthlighting_3: Planetary Disc Intersection

Purpose:

This test case is designed to examine the performance of the lighting function when the sun is partially occluded by the Earth.
Run directory: RUN_T01_LIGHT_VER

Requirements: By passing this test, this model partially satisfies requirements earthlighting_3, and earthlighting_4.

Procedure:

This test is designed to ensure that the code correctly computes the fraction of the sun's area that is visible when the sun is partially occluded by the Earth. All planetary radii were set to 1 m, the spacecraft was placed at (-2, 0, 0) m, the sun at (1, .75, 0) m, the moon at (1, -.75, 0) m, and all planetary phases were set to 0.

Results:

The major outputs of the lighting model will be detailed here. It is expected that from this test, the half-angles of both the sun and the moon will be calculated as 18.87 degrees. The model computed these angles correctly out to machine precision.

The half-angle of the Earth in this case should be 30 degrees and this was also computed correctly out to machine precision.

The observation angle between the Earth-vector and the sun-vector should then be 14.036 degrees. This value was also calculated to machine precision. The moon observation vector should also equal this value, and this was calculated correctly as well.

With this data, the function should correctly calculate the occluded area for each of the planetary bodies. In this case, the values will be the same for both the sun and the moon, and should be equal to .3226. This value is not recorded in the simulation, but it was checked with a debugger and agreed with the values out to the expected precision.

The fraction of the occluded area should then be this value divided by the total area of the body on the unit sphere which is equal to .947 and the visible area should be equal to 1 minus this value. In both cases, the results output from the simulation were accurate to the expected precision. The final lighting fraction should be zero based on the phase value of zero used in simulation and this was correct as well.

Test earthlighting_4: Complete moon Occlusion

Purpose:

This test case is designed to examine the performance of the lighting function when the sun is partially occluded by the Earth.
Run directory: RUN_T02_LIGHT_VER

Requirements: By passing this test, this model partially satisfies requirements earthlighting_3, and earthlighting_4.

Procedure:

This test is designed to ensure that the code correctly computes the fraction of the sun's area that is visible when the sun is partially occluded by the Earth. All planetary radii were set to 1 m, the spacecraft was placed at (-2, 0, 0) m, the sun at (1, 1.5, 0) m, the moon at (1, 0, 0) m, and all planetary phases were set to 1.

Results:

In this test, the moon should be completely occluded. The sun should have a new observation angle calculated as 26.57 degrees. This value was calculated out to machine precision by the model.

The moon should be completely occluded by the Earth and it was according to the code.

The sun should be partially occluded albeit with a different value due to its new position in the sky. The sun was in a different position and the calculated occluded fraction should be equal to .56137, and it agreed with the MATLAB calculation out to the expected 8 digits of precision.

Once again, the model preformed exactly as expected.

Test earthlighting_5: Full Sun Exposure

Purpose:

This test case is designed to examine the performance of the lighting function when the sun is not occluded by the Earth.
Run directory: RUN_T03_LIGHT_VER

Requirements: By passing this test, this model partially satisfies requirements earthlighting_3, and earthlighting_4.

Procedure:

This test is designed to ensure that the code correctly computes the fraction of the sun's area that is visible when the sun is not occluded. All planetary radii were set to 1 m except for the moon's which was set to 2 m, the spacecraft was placed at (-2, 0, 0) m, the sun at (-3, 0.0, 0) m, the moon at (1, 1.5, 0) m, and all planetary phases were set to 0.5.

Results:

In this test, the sun should be completely visible to the spacecraft. The moon should be partially occluded by the Earth although it will have a larger visible area than in the first test because of its increased radius. The Earth-Albedo effect in this case should be equal to the value of the Solar fraction.

The new lunar half-angle should be equal to 36.604 degrees, and this value matched the expected value to machine precision. The larger visible area should be equal to .5294, and this agreed out to the expected precision. The occluded and visible fraction agreed with the correct values (0.4128 and 0.5871) out to the expected precision. And the lighting fraction was then half of this value, as expected.

The Earth-Albedo effects were then calculated to the best of the code's abilities. The calculated fraction was 0.5. It is important to note this behavior here. When calculating the Earth-Albedo effects, the code basically takes the fraction of the Solar radiation normal to the surface of the Earth and uses this value for the Earth-Albedo fraction. There is no correction factor applied, so the Earth is essentially treated as a perfect reflector. This is a bad approximation and will make results inaccurate if used blindly. However, for the intended purposes of this model, to give approximate lighting effects only for visualization and not for analytical engineering, this approximation can be used successfully. Otherwise the user must be aware of the method's limitations, and not use this approximation for any application where precision engineering is required.

5.3 Metrics

5.3.1 Code Metrics

Table 5.3 presents coarse metrics on the source files that comprise the model.

Table 5.3: Coarse Metrics

	Number of Lines			
File Name	**Blank**	**Comment**	**Code**	**Total**
include/class_declarations.hh	8	21	9	38
include/earth_lighting.hh	56	127	71	254
include/earth_lighting_messages.hh	23	41	15	79
src/earth_lighting.cc	114	165	206	485
src/earth_lighting_messages.cc	16	30	6	52
Total	217	384	307	908

Table 5.4 presents the extended cyclomatic complexity (ECC) of the methods defined in the model.

Table 5.4: Cyclomatic Complexity

Method	**File**	**Line**	**ECC**
jeod::LightingBody::Lighting Body (void)	src/earth_lighting.cc	68	1
jeod::LightingBody::~Lighting Body (void)	src/earth_lighting.cc	87	1
jeod::LightingParams:: LightingParams (void)	src/earth_lighting.cc	106	1
jeod::LightingParams::~ LightingParams (void)	src/earth_lighting.cc	129	1
jeod::EarthLighting::Earth Lighting (void)	src/earth_lighting.cc	148	1
jeod::EarthLighting::~Earth Lighting (void)	src/earth_lighting.cc	172	1
jeod::EarthLighting::initialize (DynManager& manager)	src/earth_lighting.cc	197	4
jeod::EarthLighting::circle_intersect (double r_bottom, double r_top, double d_centers, double* area)	src/earth_lighting.cc	264	5

Continued on next page

Table 5.4: Cyclomatic Complexity (continued)

Method	**File**	**Line**	**ECC**
jeod::EarthLighting::calc_lighting (double pos_veh[3])	src/earth_lighting.cc	374	4

5.4 Requirements Traceability

Table 5.5: Requirements Traceability

Requirement	**Inspection and Testing**
earthlighting_1 - Top-level Requirements	Insp. earthlighting_1
earthlighting_2 - Vehicle State Encapsulation	Insp. earthlighting_2
earthlighting_3 - Summarize Lighting Bodies	Test earthlighting_1 Test earthlighting_2 Test earthlighting_3 Test earthlighting_4 Test earthlighting_5
earthlighting_4 - Calculate Illumination	Test earthlighting_1 Test earthlighting_2 Test earthlighting_3 Test earthlighting_4 Test earthlighting_5

Bibliography

[1] Generated by doxygen. *Low Earth Orbit Lighting Reference Manual*. National Aeronautics and Space Administration, Johnson Space Center, Software, Robotics & Simulation Division, Simulation and Graphics Branch, 2101 NASA Parkway, Houston, Texas, 77058, July 2022.

[2] Hammen, D. Dynamics Manager Model. Technical Report JSC-61777-dynamics/dyn_manager, NASA, Johnson Space Center, Houston, Texas, July 2022.

[3] Hammen, D. Dynamic Body Model. Technical Report JSC-61777-dynamics/dyn_body, NASA, Johnson Space Center, Houston, Texas, July 2022.

[4] Jackson, A., Thebeau, C. JSC Engineering Orbital Dynamics. Technical Report JSC-61777-docs, NASA, Johnson Space Center, Houston, Texas, July 2022.

[5] NASA. NASA Software Engineering Requirements. Technical Report NPR-7150.2, NASA, NASA Headquarters, Washington, D.C., September 2004.

[6] Turner, G. Derived State Model. Technical Report JSC-61777-dynamics/derived_state, NASA, Johnson Space Center, Houston, Texas, July 2022.