Commonality Analysis for Family of Lighting Models

Sasha Soraine

December 4, 2019

1 Revision History

Date	Version	Notes
October 1, 2019	1.0	Original draft
October 17, 2019	1.1	Updates in Response to GitHub Issues
November 3, 2019	2.0	Major Revisions Based on Dr. Smith's Feedback

Contents

2 Reference Material

This section records information for easy reference.

2.1 Table of Units

Throughout this document SI (Système International d'Unités) is employed as the unit system. In addition to the basic units, several derived units are used as described below. For each unit, the symbol is given followed by a description of the unit and the SI name.

symbol	unit	SI
rad	angle	radian
cd	luminous intensity	candela

2.2 Table of Symbols

The table that follows summarizes the symbols used in this document along with their units. The choice of symbols was made to be consistent with the physics and calculus notation. The symbols are listed in alphabetical order.

symbol	unit	description
$\mathbb{R}$	—	Set of Real numbers.
$\mathbb{Z}$	—	Set of Integers.
$\mathbb{Z}_+$	—	Set of Positive Integers.
ψ	rad	Angle between the normal and halfway vector.
θ	rad	Angle between the viewer and the reflected ray.
θ_i	rad	Angle of incidence between incident ray and surface normal.
θ_r	rad	Angle of reflection between reflected ray and surface normal.
θ_1	rad	Angle of incidence between incident ray and surface normal in material 1.
θ_2	rad	Angle of refraction between refracted ray and surface normal in material 2.
n_1	—	Refractive index of first material.
n_2	—	Refractive index of second material.
n_i	—	Refractive index of i-th material.
p	—	Point in space or on a surface.
p_0	—	Point of origin.
v	—	Position of viewer represented as a 3D point in space.

P	—	n-dimensional Vector.
P_x	—	x dimension of Vector P.
P_y	—	y dimension of Vector P.
P_z	—	z dimension of Vector P.
P_i	—	ith dimension of Vector P.
Q	—	n-dimensional Vector.
Q_x	—	x dimension of Vector Q.
Q_y	—	y dimension of Vector Q.
Q_z	—	z dimension of Vector Q.
Q_i	—	ith dimension of Vector Q.
L_i	—	Vector form of incident ray.
L_r	—	Vector form of reflected ray.
V	—	Vector from point to the viewer.
n	—	Vector form of surface normal.
N	—	Unit vector of surface normal, n.
H	—	Unit vector halfway between the incident ray, L_i, and the viewer vector, V.
I_a	cd	Ambient luminous intensity.
I_{ar}	cd	Red Colour ambient luminous intensity.
I_{ag}	cd	Green Colour ambient luminous intensity.
I_{ab}	cd	Blue Colour ambient luminous intensity.
k_a	–	Coefficient of Ambient Reflection.
I_d	cd	Diffuse luminous intensity.
I_{dr}	cd	Red Colour Diffuse luminous intensity.
I_{dg}	cd	Green Colour Diffuse luminous intensity.
I_{db}	cd	Blue Colour Diffuse luminous intensity.
k_d	–	Coefficient of Diffuse Reflection.
I_s	cd	Specular luminous intensity.
I_{sr}	cd	Red Colour Specular luminous intensity.
I_{sg}	cd	Green Colour Specular luminous intensity.
I_{sb}	cd	Blue Colour Specular luminous intensity.
k_s	–	Coefficient of Specular Reflection.
α	–	Shininess Coefficient.
I_T	cd	Total luminous intensity.
$i(p)$	cd	Intensity of light at a point p.

| $i(p, p_0)$ | cd | Intensity of light from point p_0 at a point p. |

2.3   Abbreviations and Acronyms

symbol	description
A	Assumption
AS	Scope Time Assumption
AB	Build Time Assumption
AR	Run Time Assumption
DD	Data Definition
GD	General Definition
GS	Goal Statement
IM	Instance Model
LC	Likely Change
PS	Physical System Description
R	Requirement
SRS	Software Requirements Specification
CA	Commonality Analysis
LM	Lighting Model
Family of Lighting Models	Family of Lighting Models
T	Theoretical Model
3D	Three Dimensional
API	Application Programming Interface

2.4   Notational Convention

This section outlines common notation convention used throughout this document.

Notation	Meaning
P	Capital letters denote a vector.
P_x	The subscript symbol with a capital letter denotes the subscript-component of the vector. In this case, we see the x-component of vector P.
$\langle P_x, P_y, P_z \rangle$	Angled brackets denote the components of a vector.
(x, y, z)	Rounded brackets represent a tuple. These are used to denote coordinates, and multi-variable information like colour.

Table 1: Notation used in this document.

2.5 System Variables

This section outlines system variables used throughout this document. The purpose is to increase the readability and maintainability of this document by abstracting threshold values to variables that are defined here.

Variable Name	Value	Meaning
MIN_NUM_LIGHTS	1	The minimum number of light sources that must be in a scene.
MAX_NUM_LIGHTS	1	The maximum number of light sources that can be in a scene.
LIGHT_POSITION		The position of the light source as a point, (x,y,z). All types of light sources treat their origin as a point source.
LIGHT_COLOUR		The light colour represented as an (r,g,b)-tuple.
MIN_NUM_OBJECTS	1	The minimum number of objects that must be in a scene.
MAX_NUM_OBJECTS	1	The maximum number of objects that can be in a scene.
OBJECT_POSITION		The position of the object's centrepoint represented as 3D coordinates (x,y,z).
OBJECT_COLOUR		The object colour represented as an (r,g,b)-tuple.
OBSERVER_POSITION		The position of the observer represented as a point, (x,y,z)
MAX_HEIGHT	100.0f	The maximum height that a scene can be.
MAX_WIDTH	100.0f	The maximum width that a scene can be.
MAX_DEPTH	100.0f	The maximum depth that a scene can be.
DEF_HEIGHT	10.0f	The default height of the scene.
DEF_WIDTH	10.0f	The default width of the scene.
DEF_DEPTH	10.0f	The default depth of the scene.
MODIFICATION_TIME_THRESHOLD	2 DAYS	The maximum amount of time it should take to modify (add, delete, or change) an element of the system.
USABILITY_THRESHOLD	90%	The threshold of users that determines usability success.
DEF_SHADE	Gouraud	The default shading model for the scene.
DEF_LIGHT	Lambertian	The default lighting model for the scene.

Table 2: System variable definitions of this document.

3   Introduction

An important part of computer graphics is modeling the lighting of objects in a virtual environment. Understanding and modeling how light interacts with objects of different materials is necessary to provide realistic lighting to virtual environments. Having an open source lighting model library that is reliable and robust would allow for new computer graphics programmers to more efficiently create scenes.

The following section provides an overview of the Commonality Analysis for a family of lighting models. This section explains the purpose of the document, scope of the family, characteristics of the intended reader, and organization of the document.

3.1   Purpose of Document

This document describes a family of lighting models to be used for computer graphics. This document is intended to be used as a reference to be provide the necessary information to verify the family of models, and implement the different family members.

This document captures the problem domain, theoretical models used to address the problem, the commonalities and variabilities between members of the family, and the requirements common to those members. It serves as a starting point to the design and implementation of a library of lighting models, and will be referenced in the creation of a verification and validation plan.

3.2   Scope of the Family

In the physical world, light is described through the physics of optics which describes the interaction of light with objects. The simplest approximation of light's behaviour is geometrical optics, where light is treated as a continuous ray that interacts with object surfaces based on a set of well defined principles. Even though these behaviour principles are well defined, lighting 3D graphical environments has been computationally difficult. This is because individual light sources emit a near infinite number of rays whose behaviour needs to be individually calculated and tracked as it reflects and refracts off of object surfaces.

Lighting in 3D graphics happens at two levels: global and local illumination. Global illumination is an indirect lighting method that describes how light interacts with all objects in a scene; this includes light bouncing off of objects onto other objects. This is performed through ray-tracing and radiosity. Local illumination is a direct lighting method that describes how light from a specific light source interacts with the surface of individual objects. This is done through calculating how light rays reflect and refract in relation to surface normals of an object. Global and local illumination are both necessary to fully light a scene.

Local illumination is broken up into shading models, and lighting models. **Lighting models** define the colour of individual parts of objects. This is done through combining the specular, ambient, and diffuse components of lighting for each part of the object. The simplification here is through the approximations made in calculating those three components. **Shading models** approximate how light interacts with individual objects; the underlying

simplification being that we can't continuously calculate light across the whole surface of an object, so the shading model chooses representative points to calculate the light interaction. The shading model also decides whether that information how that information will be interpolated between calculated points. For example, a shading model may decide that all lighting calculations will happen at the vertices of an object's mesh and that the values will be held constant between calculated points. Essentially, shading models cap the number of computations to make and simplifies the problem of interaction.

We can organize the relationship between lighting methods in 3D graphics and its physics basis in graphical form (see 1). The most abstract separation is the type of optics that the techniques are based on (wave or geometrical optics). In the domain of geometrical optics based techniques we can separate methods based on whether they address global or local illumination. Finally at the level of local illumination we can see the techniques of shading and lighting models.
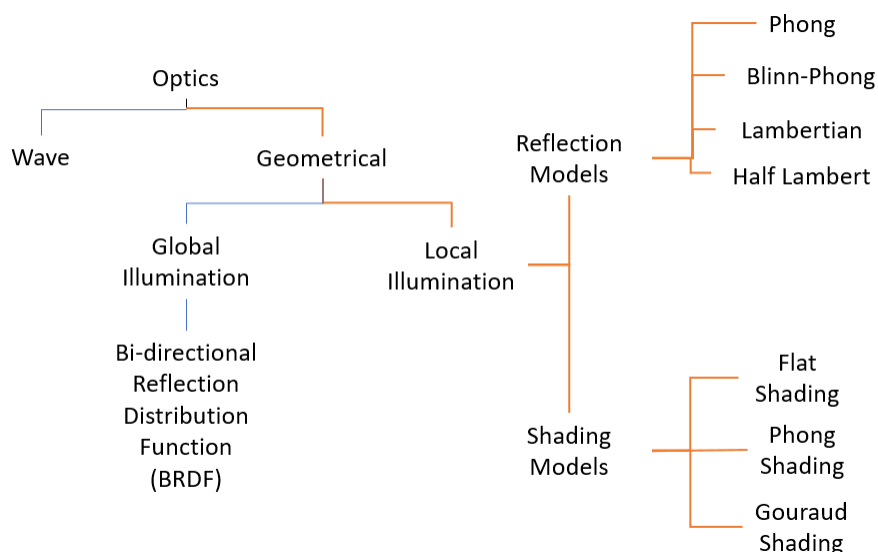

Figure 1: Problem Domain for Lighting Models of Computer Graphics.

The problem domain for lighting models in computer graphics is obviously broad. To simplify this problem we invoke assumptions AS1, AS2. This allows us to focus on understanding the techniques of local illumination (all aspects connected by the orange line in 1).

The scope of the family includes geometrical optics simulation of light reflection for 3D material objects in a local illumination context.

3.3   Characteristics of Intended Reader

The intended readers of this document should have understanding of Grade 12 Physics (particularly Optics) and a undergraduate Level 2 understanding of Linear Algebra and Matrix operations.

3.4 Organization of Document

This document is organized in accordance with the CA template for scientific computing software provided by Dr. Smith for CAS 741. These templates are based on work by Smith (2006).

4 General System Description

This section identifies the interfaces between the system and its environment, describes the potential user characteristics and lists the potential system constraints.

4.1 Potential System Contexts

Figure 2 shows the high level system context. The circle represents the system user. The box is the library of lighting models system. Arrows are used to show the flow of data between the system and its environment.
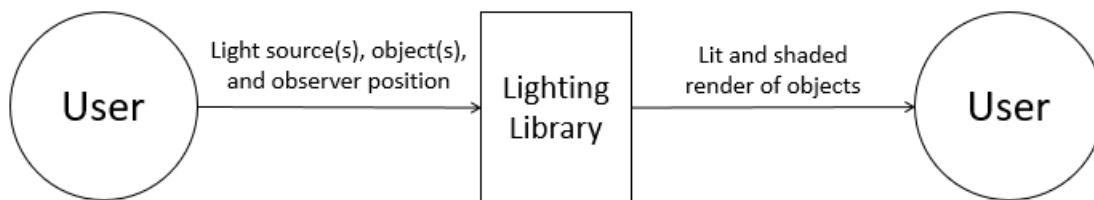

Figure 2: High level system context for Family of Lighting Models

- User Responsibilities:

  - Ensure that the problem they are looking to solve matches the assumptions made for this family.
  - Provide information on the light source(s), object(s) in scene, and observer position.
  - Declare shading model to use from preset options.
  - Declare lighting model to use from present options.
  - Ensure application programming interface use complies with the user guide.

- Family of Lighting Models Responsibilities:

  - Calculate the reflections of all light rays coming from the light source(s).
  - Determine which light rays (reflected or from source) reach the observer.
  - Render a lit environment based on selected shading and lighting model.

– Update the calculations and render in response to changes in the input data.

– Detect data type mismatch, such as a string of characters instead of a floating point number.

4.2 Potential User Characteristics

The end user of Family of Lighting Models should have an Computer Science/Software Engineering Undergraduate Level 3 understanding of Computer Graphics (such as through completing the SFWR ENG/CS 3GC3) and moderate experience with programming.

4.3 Potential System Constraints

There are no system constraints.

4

5 Commonalities

This section presents a high-level view of the problem. It captures terminology and definitions relevant to the problem, theoretical models that are common to all members of the family, goal statements of the family, data definitions that will be used to solve these problems, as well as commonalities in inputs, calculations, and outputs.

This section also provides a background overview which explains the motivation for this work.

5.1 Background Overview

Computer graphics and the techniques developed therein have become extremely useful in a variety of fields. Animation, games and media all rely on computer graphics to render semi-realistic images for their applications. As we move into an era of virtual and augmented reality we see that computer graphics has a place beyond the scope of entertainment. Training and rehabilitation simulations are extremely popular uses for computer graphics.

Modeling the interaction of light in realistic ways is difficult for computers. As such it becomes essential to have mostly accurate approximations of lights behaviours that can be used to render scenes in near real-time. On top of the need for efficient approximations, it's imperative to make the libraries and programs that handle lighting easy to use. It would create an incredibly high learning curve if users needed to understand all of the linear algebra and physics behind the approximations to mock up a scene. Existing paradigms for graphics coding offer APIs to try and make it more user friendly, but there is still a disproportionate focucs in graphics to manually fixing the scene and matrix manipulations.

The broad purpose of this work is to abstract the details of lighting in computer graphics. We aim for end-users to specify their objects and light sources, and have the system render a fully lit and shaded scene. This type of work would be useful in teaching lighting model

concepts, as it would easily allow for switching between different models. More concretely the exercise of understanding the family of lighting models would allow for future students to review this creation process and have a deeper understanding of lighting in computer graphics. Outside of academia, it would similarly be useful for individuals working on small scale graphics problems, like prototyping a simulation environment. Work like this would allow them to not worry about the math involved in getting the environment lit adequately, and instead to focus on their more abstract problem.

5.2 Terminology and Definitions

This subsection provides a list of terms that are used in the subsequent sections and their meaning, with the purpose of reducing ambiguity and making it easier to correctly understand the requirements. Definitions are borrowed or synthesized from Lengyel (2003); Comninos (2005); Shreiner and Angel (2012).

Geometrical Optics: The study of lights as rays.

Ray: A view of light as a continuous beam, represented as a vector.

Reflection: The redirection caused by collision of light rays with objects.

Specular reflection: Perfect reflection of a light ray on a smooth/glossy surface. This is used to create highlights on objects, or create highly reflective objects like mirrors.

Diffuse (Lambertian) reflection: Reflected light ray is scattered in random directions due to microfacets in the surface. This type of reflection does not depend of the position of an observer.

Refraction: The redirection of light rays when passing through different materials.

Illumination: Process by which amount of light reaching a surface is determined.

Global Illumination: Interactions of light through an entire scene as it reflects off of objects onto other objects. Often done through techniques like ray tracing, and radiosity.

Ray tracing: A rendering technique that traces the path of light as pixels through a scene and models its interaction with the objects it collides with.

Radiosity:

Local Illumination: Interactions of light from light sources with individual objects as defined by the shading and lighting models.

Shading Model: methods used to determine the colour and intensity of light reflected towards viewer.

Lighting model: Method used to calculate the colour of a point on a surface. Incorporates a reflection model and a shading model.

Reflection model: How light reflects off a surface.

Shading model: How the normal vector is calculated for polygons in the scene.

Light source: The origin of light rays.

Ambient light: Light with no identifiable source or direction. It has equal intensity in every direction, and illuminates every part of an object uniformly.

Directional (or infinite) light: Light defined only by direction; light travels infinitely in that single direction with constant intensity.

Point light: Light defined only by a point; light travels uniformly in every direction from that point, with intensity decreasing with inverse square of distance from source.

Spotlight: Light defined by a point and direction; light travels infinitely in a single direction from the defined source point with intensity decreasing with inverse square of distance from source and increased radius of coverage.

Render: The output of a graphics program.

Environment: The background objects in a scene along with information about light sources.

Scene: A collection of objects with specific material properties.

Object: Set of polygons, edges, and vertices.

Polygon: Closed shape (with well defined interior and exterior).

Normal: A vector perpendicular to a surface, point, or pair of vectors.
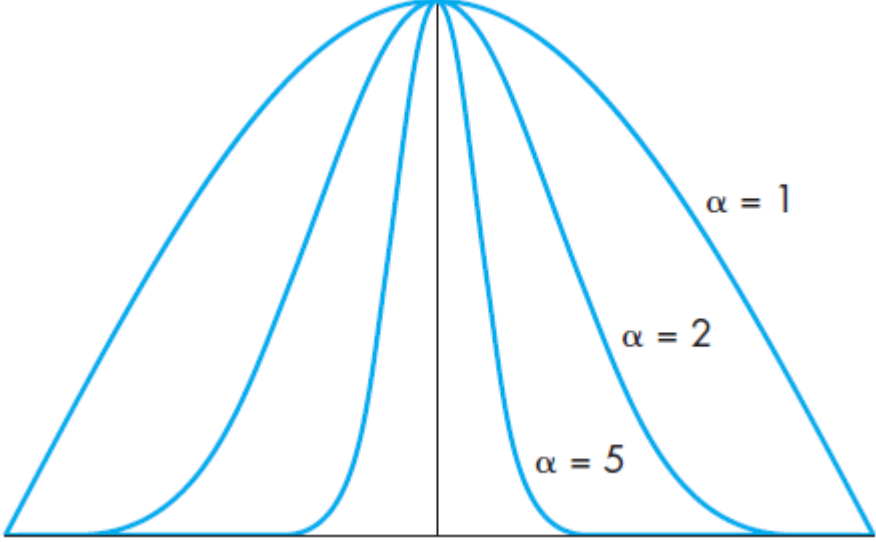
5.3   Data Definitions

This section collects and defines all the data needed to build the instance models. The dimension of each quantity is also given.

Number	DD1
Label	**Dot Product of two n-dimensional vectors**
Symbol	$P \bullet Q$
SI Units	–
Equation	$P \bullet Q = \sum_1^n P_i Q_i$
Description	P and Q are two n-dimensional vectors. The dot product ($P \bullet Q$) is the scalar sum of the products of each component. The dot product acts a measure of the difference between the directions in which P and Q point. P and Q are perpendicular when $P \bullet Q = 0$..
Sources	Lengyel (2003)
Ref. By	AB1, AS5, IM1, IM2, IM3, IM4, T1 and GD1

Number	DD2
Label	**Cross Product of two 3-dimensional vectors**
Symbol	$P \times Q$
SI Units	–
Equation	$P \times Q = \langle P_y Q_z - P_z Q_y, P_z Q_x - P_x Q_z, P_x Q_y - P_y Q_x \rangle$
Description	P and Q are two 3-dimensional vectors. The cross product ($P \times Q$) is the unit vector normal to both P and Q. The calculation of the cross product's direction follows the *right hand rule* paradigm. This is used in T1 and GD1.
Sources	Lengyel (2003)
Ref. By	AB1

Number	DD3
Label	**Ambient Intensity for a Light Source**
Symbol	I_a
SI Units	cd
Equation	$I_a = (I_{ar}, I_{ag}, I_{ab}) = i(p, p_0) \cdot k_a$
Description	$i(p, p_0)$ is the intensity of the light at point p from light source position p_0. Depending on the type of light source the intensity will change with distance from the source. k_a is the coefficient of ambient reflection.
Sources	Shreiner and Angel (2012)
Ref. By	IM3, IM4
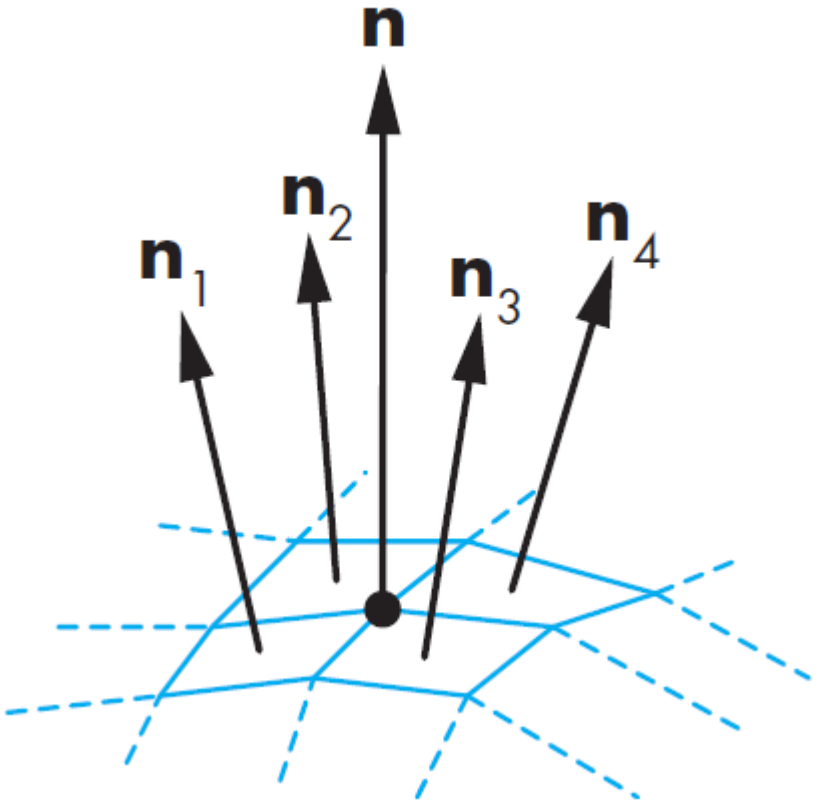
Number	DD4
Label	**Diffuse Luminance for a Light Source**
Symbol	I_d
SI Units	cd
Equation	$I_d = (I_{dr}, I_{dg}, I_{db}) = k_d \cdot i(p, p_0) \cdot \max(0, (L_i \bullet N))$
Description	p is the point where L_i hits the surface of an object. $i(p, p_0)$ is the intensity of L_i at point p from light source position p_0. k_d is the coefficient of diffuse reflection. L_i is the incidence ray. N is the unit vector surface normal.
Sources	Shreiner and Angel (2012)
Ref. By	IM1, IM2, IM3, IM4
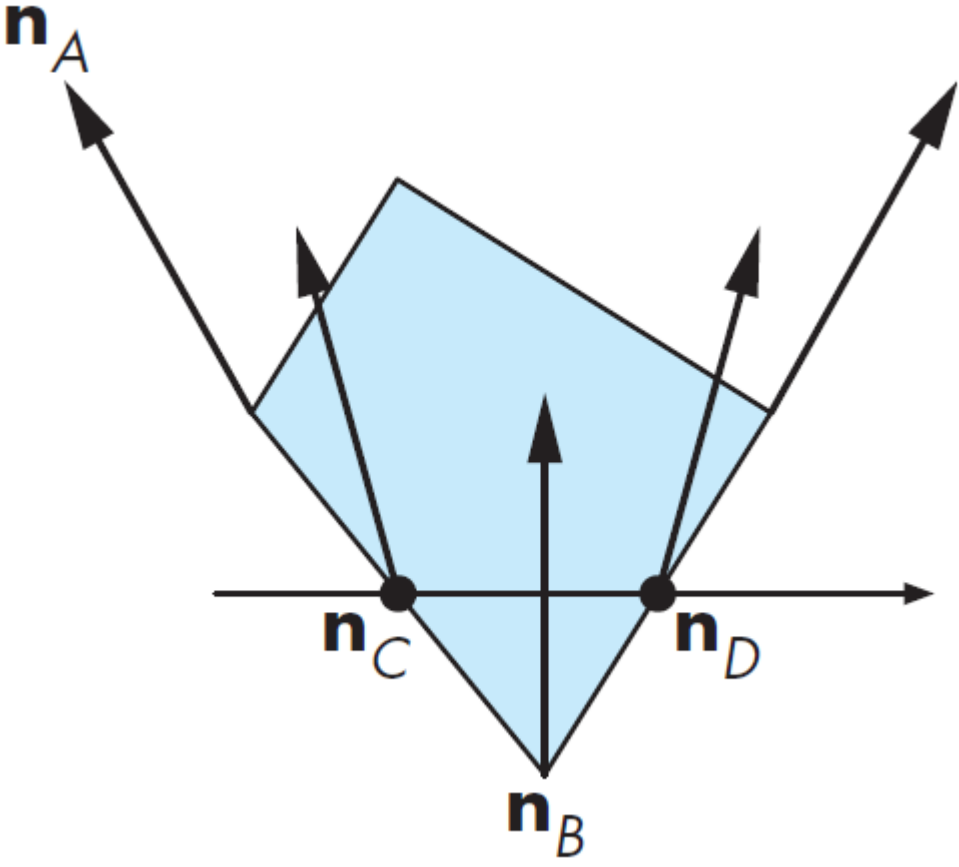
Number	DD5
Label	**Specular Luminance for a Light Source**
Symbol	I_s
SI Units	cd
Equation	$I_s = (I_{sr}, I_{sg}, I_{sb}) = k_s \cdot i(p, p_0) \cdot \max(0, (L_r \bullet V))^\alpha$
Description	p is the point where L_i hits the surface of an object.
	$i(p, p_0)$ is the intensity of L_i at point p from light source position p_0.
	L_r is the reflected ray. It is the reflection of the incidence ray, L_i, about the normal, N.
	V is the unit vector between the point where the incidence ray hit the object, and the observer.
	k_s is the specular coefficient of the material.
	α is the shininess coefficient to determine the size of the specular highlight. The following image describes the behaviour of the specular highlight size based on α
Sources	Shreiner and Angel (2012)
Ref. By	IM3, IM4

Number	DD6
Label	**Total Luminance for a Light Source**
Symbol	I_T
SI Units	cd
Equation	$I_T = I_a + I_d + I_s$
Description	Total luminance is the combination of ambient, diffuse, and specular luminance.
Sources	Shreiner and Angel (2012)
Ref. By	IM3

Number	DD7
Label	**Luminous intensity at point p from a point source, p_0**
Symbol	$i(p, p_0)$
SI Units	cd
Equation	$i(p, p_0) = \frac{1}{|p-p_0|^2} I(p_0)$
Description	Calculates the intensity of light at point p from a light source at position p_0 in Cartesian Coordinates.
	$I(p_0)$ is the intensity of the light source at it's source. Therefore it is just the straight intensity of the light.
Sources	Lengyel (2003)
Ref. By	AB1, AB2

Number	DD8
Label	**Calculation of Normals for Flat Shading**
Symbol	$\vec{n}$
SI Units	–
Equation	$\vec{n} = \frac{\vec{P} \times \vec{Q}}{|\vec{P} \times \vec{Q}|}$
Description	Flat shading calculates the surface normal of a polygon in a mesh. Given two non-parallel vectors tangent to the surface of the polygon, $\vec{P}$ and $\vec{Q}$, the normal ($\vec{n}$) is the normalized cross-product of $\vec{P}$ and $\vec{Q}$
Sources	Shreiner and Angel (2012)
Ref. By	AB1, AB2

Number	DD9
Label	**Calculation of Normals for Gouraud Shading**
Symbol	$\vec{n_v}$
SI Units	–
Equation	$\vec{n_v} = \frac{\sum_{i=1}^{t_v} n_i}{|\sum_{i=1}^{t_v} n_i|}$
Description	Gouraud shading calculates the normal at the vertices of a polygon in a mesh. Given a vertex, v, the normal ($\vec{n_v}$) is the normalized average of the surface normals ($\vec{n_i}$) for all polygons that share that vertex. t_v is the number of polygons that share the vertex v.
Sources	Shreiner and Angel (2012)
Ref. By	AB1, AB2

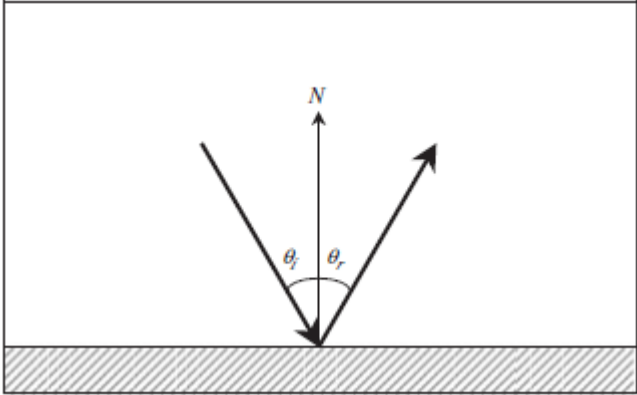| Number	DD10
Label	**Calculation of Normals for Phong Shading**
Symbol	$\vec{n_v}$
SI Units	–
Equation	
Description	Phong shading linearly interpolates normals across each the surface of the polygon from the vertex normals. Surface normals are then normalized per pixel.
Sources	Shreiner and Angel (2012)
Ref. By	AB1, AB2

5.4 Goal Statements

Given some light source(s), some object(s) and their respective material properties, and an observer the goal statements are:

GS1: Render a fully lit and shaded scene of the objects based on the observer location.

[I like this goal statement. However, I feel like the rest of the document doesn't completely refine it. I feel like there should be an instance model that takes the refined version of these inputs and outputs a refined version of the output. —SS]
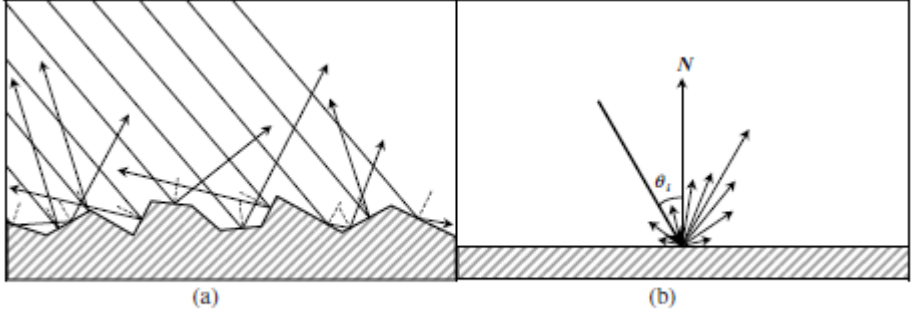
5.5 Theoretical Models

This section focuses on the general equations and laws that Family of Lighting Models is based on.

Number	T1
Label	**Law of Reflection**
Equation	$\theta_i = \theta_r$ or $L_i \bullet N = L_r \bullet N$
Description	When a ray of light reflects off a surface, the angle of incident is equal to the angle of reflection. This can also be written as the dot product of the incident ray and the normal is equal to the dot product of the reflected ray and the normal.
Source	Comninos (2005)
Ref. By	GD1

5.5.1 General Definitions

This section collects the laws and equations that will be used in building the instance models.

Number	GD1
Label	**Diffuse Reflection**
SI Units	-
Equation	$\theta_i = \theta_r$ or $L_i \bullet N = L_r \bullet N$
Description	Law of Reflection applied to rough surfaces. (a) The surface micro-facets. (b) Diffuse reflection.
Source	Comninos (2005)
Ref. By	DD1

6 Variabilities

The follow section outlines variabilities between family members. Section 6.1 covers the assumptions made to simplify/realise the problem. Section 6.2 captures variabilities in the implementation.

Before tackling those sections, we summarize here variabilities in the problem.

6.1 Assumptions

This section outlines the various assumptions made in defining this problem. The nature of these assumptions is to make decisions in regards to the variabilities of the system. These are invoked to limit the scope of the problem being modeled, and often indicate areas where likely changes may occur.

Assumptions are divided based on their binding time. All assumptions are used to refine the GS1 and define the potential members of the family that will be built.

6.1.1 Scope Time Bindings

These assumptions are applied in the abstract design of the program. They dictate the family members that we cover, and the potential design choices available to us. Changing these assumptions changes the underlying problem; as such this would change the family we are discussing. Therefore these assumptions will not change, and apply to all members of this family.

Physics Assumptions

AS1: Light interactions will be defined by Geometrical Optics principles (ray-based).

Rendering Limitations

AS2: Lighting will be handled at the local illumination level.

6.1.2 Build Time Bindings

These assumptions are applied in the concrete design of any individual family member. They dictate the specifics of that family member, and concretize our design choices. Changing these assumptions would change the particular family member that is being built. As such, these assumptions are likely to change for different contexts.

Scene Assumptions

AB1: The virtual environment will be described by a 3D Caretsian Coordinate System using right-hand rules.

> *Alternate Assumptions*
> * The virtual environment will be described by a polar coordinate system.

AB2: Only one scene can be loaded into the system at a time.

AB3: The scene size is defined as a rectangular room with dimensions SCENE_HEIGHT, SCENE_WIDTH, and SCENE_DEPTH.

Object Assumptions

AB4: All objects are opaque; light will not experience refraction with any object.

> *Alternate Assumptions*
> * Objects can be transparent or opaque; light will experience refraction with transparent objects.
> * Objects can be transparent, translucent, or opaque; light will experience refraction with transparent and translucent objects.

AB5: Objects will not cast shadows on each other. Related to AS2.

AB6: Objects will not reflect light at each other. Related to AS2.

AB7: Objects will not emit light. Related to AS2.

AB8: Positions of objects are defined on a 3D Cartesian Coordinate System. Related to AB1.

AB9: Objects have a fixed position and will not be moved during the running of the program.

AB10: Object shapes will be approximated by a polygon mesh of triangles.

> *Alternate Assumptions*
>   * Objects will be approximated by a polygon mesh of quadrilaterals.
>   * Objects will be approximated by a polygon mesh of convex polygons.

AB11: Object polygon mesh is stored as a list of vertices and a set of polygon surfaces that are associated with vertices. Related to AB**??**.

> *Alternate Assumptions*
>   * Polygon mesh is stored as list of all vertices, and edges connecting them.

AB12: Object meshes will be closed and convex (there is a well defined interior and exterior). Related to AB**??**

Lighting Assumptions

AB13: Positions of light sources are defined on a 3D Cartesian Coordinate System. Related to AB1.

AB14: Light sources have a fixed position and will not be moved during the running of the program. Related to AB**??**.

Observer Assumptions

AB15: There will only be a single observer in a scene.

AB16: Observer position is defined on a 3D Cartesian Coordinate System. Related to AB1.

AB17: The observer position will be fixed for the run-time of the program.

Input Assumptions

AB18: Input will be accepted as a formatted JSON file.

6.1.3 Run Time Bindings

These assumptions are used to detail the specifics of the individual program that will run. Most of these are captured as inputs to the system. These can change as the program is developed.

Lighting Assumptions

AR1: There will be only one light source in the scene.

>> *Alternate Assumptions*
>>> * There will be between MIN_NUM_LIGHTS and MAX_NUM_LIGHTS in the scene.

AR2: The light source will one of the following types: point, spot light, directional, or ambient.

AR3: The light source will be positioned at LIGHT_POSITION.

AR4: The LIGHT_POSITION will not overlap with the OBSERVER_POSITION or any OBJECT_POSITION.

AR5: The light source will have a colour defined by an (r,g,b)-tuple called LIGHT_COLOUR.

AR6: The light source colour will default to white (255,255,255) if not specified.

Objects Assumptions

AR7: There will be only one object in the scene.

>> *Alternate Assumptions*
>>> * There will be between MIN_NUM_OBJECTS and MAX_NUM_OBJECTS in the scene.

AR8: The object will one of the following types: sphere, cube, torus, or teapot.

AR9: The object will be positioned at OBJECT_POSITION.

AR10: The OBJECT_POSITION will not overlap with the OBSERVER_POSITION or LIGHT_POSITION.

AR11: The object will have values for the reflection, specular, diffuse, and shininess coefficients of its material.

AR12: The object will have a colour defined by an (r,g,b)-tuple called OBJECT_COLOUR.

AR13: The object colour will default to white (255,255,255) if not specified.

Observer Assumptions

AR14: The observer will be positioned at OBSERVER_POSITION.

AR15: The OBSERVER_POSITION will not overlap with any OBJECT_POSITION or LIGHT_POSITION.

6.2 Calculation

This section outlines variabilities in design details, capturing variabilities in: input, calculations, algorithms, and data structures.

Input variabilities:

Variability	Parameter of Variation	Constraints	Related
Scene Size	$SIZE\_HEIGHT : \mathbb{R}$	$0 \ll SIZE\_HEIGHT \leq MAX\_HEIGHT$	
	$SIZE\_WIDTH : \mathbb{R}$	$0 \ll SIZE\_WIDTH \leq MAX\_WIDTH$	AB1
	$SIZE\_DEPTH : \mathbb{R}$	$0 \ll SIZE\_DEPTH \leq MAX\_DEPTH$	AB??
Position of light source (x, y, z)	$x : \mathbb{R}$	$0 \ll x \leq SIZE\_HEIGHT$	AR??
	$y : \mathbb{R}$	$0 \ll y \leq SIZE\_WIDTH$	AR??
	$z : \mathbb{R}$	$0 \ll z \leq SIZE\_DEPTH$	AB??
		$(x, y, z) \neq OBSERVER\_POSITION$	AB??
		$(x, y, z) \neq OBJECT\_POSITION$	AR??

Table 3: Parameters of Variation for Input (continued).

Variability	Parameter of Variation	Constraints	Related
Position of object(s) (x, y, z)	$x : \mathbb{R}$	$0 \ll x \leq SIZE\_HEIGHT$	
	$y : \mathbb{R}$	$0 \ll y \leq SIZE\_WIDTH$	AB1
	$z : \mathbb{R}$	$0 \ll z \leq SIZE\_DEPTH$	AR??
		$(x, y, z) \neq OBSERVER\_POSITION$	
		$(x, y, z) \neq LIGHT\_POSITION$	
Position of Observer (x, y, z)	$x : \mathbb{R}$	$0 \ll x \leq SIZE\_HEIGHT$	
	$y : \mathbb{R}$	$0 \ll y \leq SIZE\_WIDTH$	AB1
	$z : \mathbb{R}$	$0 \ll z \leq SIZE\_DEPTH$	AR??
		$(x, y, z) \neq OBJECT\_POSITION$	
		$(x, y, z) \neq LIGHT\_POSITION$	
Colour of light (r, g, b)	$r : \mathbb{Z}$	$0 \leq r \leq 255$	–
	$g : \mathbb{Z}$	$0 \leq g \leq 255$	IM1, IM2, IM3, IM4
	$b : \mathbb{Z}$	$0 \leq b \leq 255$	IM1, IM2, IM3, IM4
Colour of object (r, g, b)	$r : \mathbb{Z}$	$0 \leq r \leq 255$	IM1, IM2, IM3, IM4
	$g : \mathbb{Z}$	$0 \leq g \leq 255$	IM1, IM2, IM3, IM4
	$b : \mathbb{Z}$	$0 \leq b \leq 255$	IM1, IM2, IM3, IM4

Table 4: Parameters of Variation for Input (continued).

Variability	Parameter of Variation	Constraints	Related
Ambient coefficient of material	$k_a : \mathbb{R}$	$0 \leq k_a \leq 1$	DD3, IM1, IM2, IM3, IM4
Specular coefficient of material	$k_s : \mathbb{R}$	$0 \leq k_s \leq 1$	DD5, IM3, IM4
Diffuse coefficient of material	$k_d : \mathbb{R}$	$0.5 \leq k_d \leq 1$	DD4, IM1, IM2, IM3, IM4
Shininess coefficient of material	$\alpha : \mathbb{Z}_+$	–	DD5, IM3, IM4
Type of light source	One of $\{Point, Spot, Directional, Ambient\}$	–	–
Type of object(s)	One of $\{Sphere, Cube, Torus, Teapot\}$	–	–
Type of shading	One of $\{Flat, Gouraud, Phong\}$	–	–
Number of object(s)	$\mathbb{R}$	$MIN\_NUM\_LIGHTS \leq Number of object(s) \leq MAX\_NUM\_OBJECTS$	
Number of light(s)	$\mathbb{R}$	$MIN\_NUM\_LIGHTS \leq Number of light(s) \leq MAX\_NUM\_LIGHTS$	
Number of observer(s)	$\mathbb{R}$	Number of observer(s) = 1	–
Input method	Single file with light, object and other data; multiple files (one for lights and one for objects)	–	–

Table 5: Parameters of Variation for Input (continued).

Calculation variabilities:

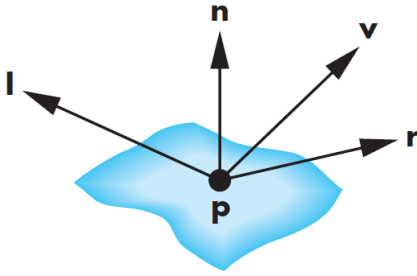| Variability	Parameters of Variation	Constraints	Ref.
Interpolation of normals for shading	One of {DD8, DD9, DD10}	–	AS1

Table 6: Parameters of Variation in Calculations.
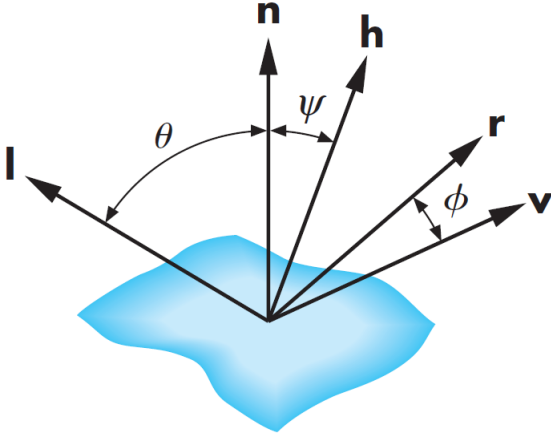
6.2.1 Instance Models

This section transforms the problem defined in Section 3.2 into one which is expressed in mathematical terms. It uses concrete symbols defined in Section 5.3 to replace the abstract symbols in the models identified in Sections 5.5 and 5.5.1.

Number	IM1
Label	**Lambertian (Diffuse) Reflection Model**
Input	L_i : Vector $\langle x, y, z \rangle$ p_0 : Point (x, y, z) LIGHT_COLOUR : (r,g,b) p : Point (x,y,z) OBJECT_BASE_COLOUR : (r,g,b) N : Vector $\langle x, y, z \rangle$
~~Output~~ Equation	OBJECT_RENDERING_COLOUR : (r,g,b) $OBJECT\_RENDERING\_COLOUR = ((L_i \bullet N)(I_{L_ip}) \cdot (LIGHT\_COLOUR) \cdot (OBJECT\_BASE\_COLOUR)$
Description	The Lambertian Reflection Model considers only the diffuse elements of light when calculating the lighting model. The approximation is assuming there's no specular component. This is a refinement on GD1. L_i is the vector representation of the incident ray. p is the point the incident ray intercepts the surface of an object. p_0 is position of the light source. N is the surface normal unit vector of the object at the position p. I_{L_ip} is the intensity of the light at point p, calculated by invoking DD7, $i(p, p_0)$
Sources	Comninos (2005); Lengyel (2003); Shreiner and Angel (2012)
Ref. By	IM2
Uses	DD7, DD1

Number	IM2
Label	**Mod Lambert (Diffuse Wrap) Reflection Model**
Input	L_i : Vector $\langle x, y, z \rangle$ p : Point (x,y,z) OBJECT_BASE_COLOUR : (r,g,b) k_d : $\mathbb{R}$ LIGHT_COLOUR : (r,g,b) p_0 : Point (x,y,z) N : Vector $\langle x, y, z \rangle$
Output	OBJECT_RENDER_COLOUR : (r,g,b)
Equation	$OBJECT\_RENDER\_COLOUR = (LambertTerm) \cdot$ $(OBJECT\_BASE\_COLOUR) \cdot (LIGHT\_COLOUR) \cdot (I_{L_ip})$
Description	This is a refinement of IM1. We incorporate the idea of a coefficient of diffuse reflection to modify the sharpness of the shadows. When $k_d = 1$ this model acts like IM1. We limit modifications be constraining $k_d \geq 0.5$; this keeps the shadows visible and conceptually matches the idea that all objects have some element of diffuse reflection. When $k_d = 0.5$ we call this special case the Half-Lambert Model. The $LambertTerm$ is the modified diffuse reflection. It is defined by: $LambertTerm = [(N \bullet L_i) \cdot k_d + (1 - k_d)]^2$ L_i is the vector representation of the incident ray. N is the surface normal unit vector of the object at position p. k_d is the coefficient of diffuse reflection. p is the point the incident ray intercepts the surface of an object. p_0 is position of the light source. I_{L_ip} is the intensity of the light at point p, calculated by invoking DD7, $i(p, p_0)$
Sources	Comninos (2005); Lengyel (2003); Shreiner and Angel (2012)
Ref. By	–
Uses	DD7, DD1

Number	IM3
Label	**Phong Reflection Model**
Input	L_i : Vector $\langle x, y, z \rangle$ p : Point (x,y,z) OBSERVER_POSITION : Point (x,y,z) OBJECT_POSITION : Point (x,y,z) OBJECT_BASE_COLOUR : (r,g,b) k_d : $\mathbb{R}$ OBJECT_SPECULAR_COLOUR: (r,g,b) k_s : $\mathbb{R}$ α : $\mathbb{R}$ N : Vector $\langle x, y, z \rangle$ p_0 : Point (x,y,z) LIGHT_COLOUR : (r,g,b)
Output	I_T : (r,g,b) OBJECT_RENDERING_COLOUR: (r,g,b)
Equation	$I_T = I_a + I_d + I_s$ $OBJECT\_RENDERING\_COLOUR = (I_T) \cdot (LIGHT\_COLOUR)$
Description	This model approximates lighting as a linear combination of ambient, diffuse, and specular components of light. All of these aspects are calculated as noted in their respective Data Definitions. The following graphic shows the vectors needed to complete all Phong Model calculations. p is the point where the incident ray L_i intersects the object's surface. p_0 is position of the light source. L_i is the normalized incident ray hitting point p.

Number	IM3 (continued)
Label	**Phong Reflection Model**
Description	N is the surface normal at point p.
	k_d is the diffuse coefficient.
	k_s is the specular coefficient
	α : is the shininess coefficient
	L_r is the normalized reflection of L_i about N.
	V is the normalized vector between the observer and p. $V = \|OBSERVER\_POSITION - p\|$
	I_a is the ambient intensity of the light described by L_i at point p. It is described by DD3 as:
	I_d is the diffuse intensity of the light described by L_i at point p. It is described by DD4 as :
	$I_d = k_d \cdot \max(0, (N \bullet L_i)) \cdot i(p, p_0) \cdot OBJECT\_BASE\_COLOUR$
	I_s is the specular intensity of the light described by L_i at point p. It is described by DD5 as:
	$I_s = k_s \cdot \max(0, (L_r \bullet V))^{\alpha} \cdot i(p, p_0) \cdot OBJECT\_SPECULAR\_COLOUR$
Sources	Comninos (2005); Lengyel (2003); Shreiner and Angel (2012)
Ref. By	IM4
Uses	DD5,DD4, DD3, DD6, DD7, DD1

Number	IM4
Label	**Blinn-Phong Reflection Model**
Input	L_i : Vector $\langle x, y, z \rangle$ p : Point (x,y,z) OBSERVER_POSITION : Point (x,y,z) OBJECT_POSITION : Point (x,y,z) OBJECT_BASE_COLOUR : (r,g,b) k_d : $\mathbb{R}$ OBJECT_SPECULAR_COLOUR: (r,g,b) k_s : $\mathbb{R}$ α : $\mathbb{R}$ N : Vector $\langle x, y, z \rangle$ p_0 : Point (x,y,z) LIGHT_COLOUR : (r,g,b)
Output	I_T : (r,g,b) OBJECT_RENDERING_COLOUR: (r,g,b)
Equation	$I_T = I_a + I_d + I_s$ $OBJECT\_RENDERING\_COLOUR = (I_T) \cdot (LIGHT\_COLOUR)$
Description	The Blinn-Phong model is a modification of the Phong model where calculations use the half-direction vector, H. This changes the calculation of I_s but nothing else.

Number	IM4 (continued)
Label	**Blinn-Phong Reflection Model**
Description	p is the point where the incident ray L_i intersects the object's surface.
	p_0 is position of the light source.
	L_i is the normalized incident ray hitting point p.
	N is the surface normal at point p.
	k_d is the diffuse coefficient.
	k_s is the specular coefficient
	α : is the shininess coefficient
	L_r is the normalized reflection of L_i about N.
	V is the normalized vector between the observer and p. $V = \|\|OBSERVER\_POSITION - p\|\|$
	H is the half-direction vector between V and L_i. $H = \|\|V + L_i\|\|$
	I_a is the ambient intensity of the light described by L_i at point p. It is described by DD3 as:
	I_d is the diffuse intensity of the light described by L_i at point p. It is described by DD4 as :
	$I_d = k_d \cdot max(0, (N \bullet L_i)) \cdot i(p, p_0) \cdot OBJECT\_BASE\_COLOUR$
	I_s is the specular intensity of the light described by L_i at point p. It is described by:
	$I_s = k_s \cdot \max(0, (N \bullet H))^{\alpha} \cdot i(p, p_0) \cdot OBJECT\_SPECULAR\_COLOUR$
Sources	Comninos (2005); Lengyel (2003); Shreiner and Angel (2012)
Ref. By	IM4
Uses	DD4, DD3, DD6, DD7, DD1

6.3 Output

Variability	Parameter of Variation
Output file	Generated code file that when compiled renders a scene, executable file that shows the rendered scene

Table 7: Parameters of Variation for Output.

7 Requirements

This section provides the functional requirements, the business tasks that the software is expected to complete, and the nonfunctional requirements, the qualities that the software is expected to exhibit.

7.1 Family of Functional Requirements

The following are a list of the services that the system shall provide:

R1: When presented with a scene in a file, the system shall correctly read from file the input data for light source(s) and object(s).

R2: System responds with specific error message when system cannot read input files.

R3: The system asks user if they would like to use the default settings when scene size, shading model, and/or reflection model information is missing from input, and applies (DEF_HEIGHT, DEF_WIDTH, DEF_DEPTH), DEF_SHADE and/or DEF_LIGHT if the user answers yes.

R4: When no input file is given, the system provides a default scene of dimension (DEF_HEIGHT, DEF_WIDTH, DEF_DEPTH) with one point light source with the default light colour, one sphere with the default material properties, rendered using default shading (DEF_SHADE) and lighting (DEF_LIGHT).

R5: The system shall verify that all input data meets constraints laid out in Table 1.

R6: The system responds with specific error message when user inputs contain errors (type mismatch, data outside of constraints).

R7: The library shall correctly calculate the surface normals for object(s) based on shading model.

R8: The library shall calculate the incidence and reflection vectors off of object(s) surface(s) based on light position(s), object(s) properties, shading model and observer position.

R9: The library shall calculate the light intensity based on light position(s), object(s) material properties, and shading model.

R10: The library shall calculate the final colour object(s) faces based on the intensities calculated in R5.

R11: The library will output code for a lit and shaded scene.

R12: Users can render a default scene (define in R??) faster than in OpenGL.

7.2 Nonfunctional Requirements

R13: The addition of new input methods should not affect the usability of the system.

R14: The addition of new lighting models, shading models, types of light sources and/or types of objects should be completable in MODIFICATION_TIME_THRESHOLD.

R15: USABILITY_THRESHOLD % of users can install the system without requiring assistance.

R16: USABILITY_THRESHOLD % of users can load an existing scene with no assistance.

R17: USABILITY_THRESHOLD % of users can change the parameters of the lighting models and re-render an existing scene with no assistance.

R18: USABILITY_THRESHOLD % of users can initialise a new scene with the default parameters (default object, light source, lighting model, and shader) with no assistance.

R19: USABILITY_THRESHOLD % of users perceive Family of Lighting Modelsto be easier to use than OpenGL.

R20: USABILITY_THRESHOLD % of users perceive Family of Lighting Modelsto allow them more control than the built in Unity shader options.

8 Likely Changes

LC1: Refractive materials incorporated into family.

LC2: More reflection models incorporated into family.

LC3: Variations on how normals are computed (e.g. partial derivatives) may be added to the family.

	AS1	AS2	AS3	AS4	AS5	AB1	AB2	AB3	AB4	AB5	AB6
T1	X				X	X					X
GD1	X		X		X	X					
IM1	X	X	X	X	X						
IM2	X	X	X	X	X						
IM3	X	X	X	X	X		X	X			
IM4	X	X	X	X	X		X	X			
DD1						X	X				
DD2						X	X				
DD3	X	X	X	X							
DD4		X			X						X
DD5		X			X						X
DD6		X	X	X	X						X
DD7	X	X				X	X				
DD8	X				X	X	X		X		
DD9	X	X	X		X	X	X		X	X	X
DD10	X	X	X		X	X	X		X	X	X
LC3	X					X	X		X	X	
LC2		X	X	X	X		X				X
LC1		X	X	X	X						X

Table 8: Traceability Matrix matching Scope and Build Time Assumptions to Theoretical Models, General Definitions, Instance Models, Data Definitions and Likely Changes.

9 Traceability Matrices and Graphs

10 Appendix

10.1 First Stage of Implementation

In this project I aim to implement the local illumination shading and lighting models outlined in this document. This means I've invoked A1, A2 and will only be working with members of the family outlined in this document. The overall goal is to implement these shaders as plug-ins for Unity (a game development engine), as such I have made certain build-time decisions to fit this context.

The following table illustrates the build-time decisions that I've made to scope the project further.

Variability	Decisions	Related Assumptions
Input	Input file will be formatted JSON, using standard conventions	
	Files will be stored in SCENE_DIR	
	File contents can be modified by the user from the system GUI, the changes will be stored as proper JSON information	
	Files will list information in the following order: • scene size; • lighting model; • shading model; • light source information; • object information; • observer information;	

Variability	Decisions	Related Assumptions
Coordinate System	3D Cartesian Coordinates with right-hand rules. System coordinates will be handled with floating point numbers.	A1
Scene	Scene size will be ordered as (height, width, depth) Scene size can be passed in at run-time through file Default scene size will be (DEF_HEIGHT, DEF_WIDTH, DEF_DEPTH) Only one scene is loaded at a time.	A?? A??
Lighting Model	Lighting model can be one of: IM1, IM2, IM3, IM4	
Shading Model	Shading model can be one of: DD8, DD9, DD10	
Lights	Lights can be point lights, spotlights, directional lights, or ambient lights Light can be passed in at run-time through file. Light structures will contain the type of light, the colour of the light, the position of the light,	
Objects	Objects can be spheres, cubes, toruses, or teapot Objects can be passed in at run-time through file. Object structures will contain a mesh containing a list of vertices and surfaces associated with vertices, the centrepoint of the object, and the material properties of the object (base colour, specular colour, coefficient of diffuse, specular, and ambient reflections, coefficient of shininess) Objects will all have a white specular colour (255,255,255)	

Variability	Decisions	Related Assumptions
Observer	There can only be 1 observer in a scene. Observer information can be passed in at run-time through a file. Observer structure contains a position (x,y,z) that it is located at, and a direction that it is facing.	
Output	Output will be a rendered scene using the Unity Engine. Rendered output image will be saved as an Unity executable in the director RENDERS_DIR	

We prioritize the implementation of the default scene's shading (DEF_SHADE) and lighting model (DEF_LIGHTS).

References

Peter Comninos. *Mathematical and Computer Programming Techniques for Computer Graphics.* Springer-Verlag, Berlin, Heidelberg, 2005. ISBN 1852339020.

Eric Lengyel. *Mathematics for 3D Game Programming and Computer Graphics, Second Edition.* Charles River Media, Inc., Rockland, MA, USA, 2003. ISBN 1584502770.

Dave Shreiner and Edward Angel. Interactive computer graphics: A top-down approach with shader-based opengl, 2012.

W. Spencer Smith. Systematic development of requirements documentation for general purpose scientific computing software. In *Proceedings of the 14th IEEE International Requirements Engineering Conference, RE 2006*, pages 209–218, Minneapolis / St. Paul, Minnesota, 2006. URL http://www.ifi.unizh.ch/req/events/RE06/.