Lighting Prediction and Simulation in Large Nighttime Urban Scenes

by

Chia-Yuan Chuang

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science in Technology

Approved November 2011 by the
Graduate Supervisory Committee:

John Femiani, Chair
Anshuman Razdan
Ashish Amresh

ARIZONA STATE UNIVERSITY

December 2011

ABSTRACT

Night vision goggles (NVGs) are widely used by helicopter pilots for flight missions at night, but the equipment can present visually confusing images especially in urban areas. A simulation tool with realistic nighttime urban images would help pilots practice and train for flight with NVGs. However, there is a lack of tools for visualizing urban areas at night. This is mainly due to difficulties in gathering the light system data, placing the light systems at suitable locations, and rendering millions of lights with complex light intensity distributions (LID). Unlike daytime images, a city can have millions of light sources at night, including street lights, illuminated signs, and light shed from building interiors through windows. In this paper, a Procedural Lighting tool (PL), which predicts the positions and properties of street lights, is presented. The PL tool is used to accomplish three aims: (1) to generate vector data layers for geographic information systems (GIS) with statistically estimated information on lighting designs for streets, as well as the locations, orientations, and models for millions of streetlights; (2) to generate geo-referenced raster data to suitable for use as light maps that cover a large scale urban area so that the effect of millions of street light can be accurately rendered at real time, and (3) to extend existing 3D models by generating detailed light-maps that can be used as UV-mapped textures to render the model. An interactive graphical user interface (GUI) for configuring and previewing lights from a Light System Database (LDB) is also presented. The GUI includes physically accurate information about LID and also the lights' spectral power distributions (SPDs) so that a light-map can be generated for use with any sensor if the sensors luminosity function is known. Finally, for areas where more detail is required, a tool has been developed for editing and visualizing light effects over a 3D building from many light sources including area lights and windows. The above components are integrated in the PL tool to produce a night time urban view for not only a large-scale area but also a detail of a city building.

TABLE OF CONTENTS

LIST OF FIGURES

1 INTRODUCTION

Night vision goggles (NVGs), in particular the advanced helmet-mounted aviators night-vision-imaging systems, allow helicopter pilots to perform low-level flight mission at night. These missions might include night time search, rescue and military applications. However, NVGs do not directly turn night into day, and while they may often provide significant advantages for night flights they may also result in visual fatigue, high workload, and safety hazards [1]. Consequently, it is important to train end-users in a realistic environment. Figure 1 illustrates the type of nighttime imagery viewed through NVGs which we aim to simulate[1].

There are many existing applications for daytime flight simulation but few of them are built for night vision goggle simulations. This is because accurate nighttime lighting in an urban area can be difficult to achieve. Daytime imagery is, for the most part, illuminated by a single source of light (the sun), but night time images are illuminated by millions of light sources scattered throughout a large area, and with varied illumination properties. Determining locations and types of nighttime light sources presents a huge challenge for urban NVG simulation.

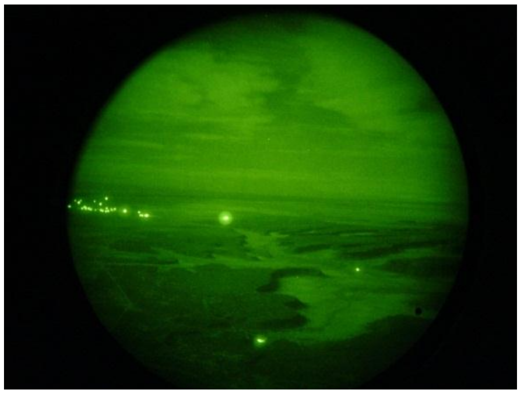


Figure 1. A scenery through a night vision goggle[2]

[1] In this thesis, I present results using CIEXYZ (visible) light. The method has also been applied to actual NVGs sensor profile but I am not permitted to show the results.

[2] http://www.classiclifeguard.com/NVG_photographs.htm

The primary objective of this research is to provide simulation designers the opportunity to rapidly design and simulate the lighting scenery which can be used in real time rendering. In particular, we have developed techniques for extending GIS data using lights whose properties are drawn from a Light System Database (LDB), which we have helped to develop along with an interactive graphical tool for configuring and previewing lights. The tool allows designers to view the light intensity distribution (LID) and the lobe, or pattern of light, projected on the ground (see Figure 23 at section 6.1). The tool also allows editing the spectral power distribution (SPD) of the light, which allows our rendering system to simulate the response of variety of sensors. For smaller areas, we have developed another tool that extends existing 3D building models such as those used for daytime flight simulations with the data required to produce accurate lighting effects from many light sources including windows (see Figure 26 in section 6.3).

An ideal PL tool can produce a plausible configuration of light systems, which means that an expert observer who is unfamiliar with an area would have difficulty distinguishing between images rendered[3] using light positions entered manually and an image produced automatically by the system. Figure 2 is an example of a nighttime image taken over Phoenix, Arizona, where the luminance is averaged over a number of cloud-free observations. This figure illustrates the amount of light in an urban area, however, it includes a blurring effect due to averaging and it is low resolution. It also captures light sources not covered in our work, such as cars and region specific lights (for example a used car lot shows as bright point). Our system produces images such as Figure 3, which includes static light sources without the blurring caused by averaging.

[3] The real-time rendering problem is beyond the scope of this document; I am concerned only with the location and properties of each luminaire.

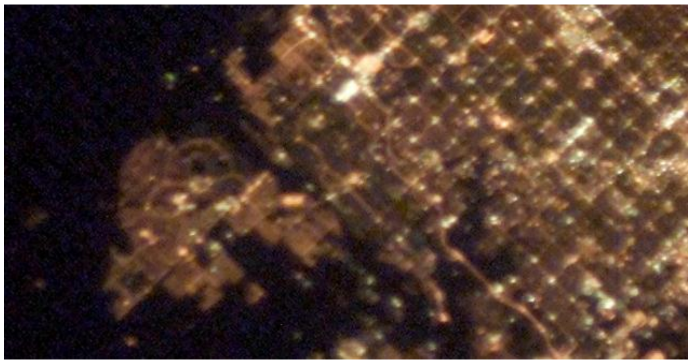


Figure 2. Nighttime image of Phoenix, AZ. This image is acquired from National Oceanic and Atmospheric Administration (NOAA).

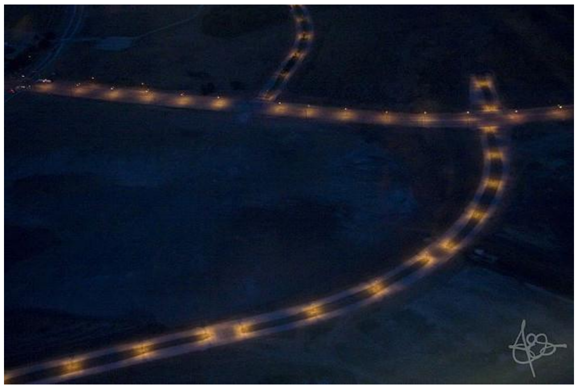


Figure 3. A real world nighttime image found on the internet[4]

When calculating the lighting from streetlights, we make the assumption that light energy is emitted from points, which are called luminaires. The luminaires are a part of “light systems”, each of which may contain several luminaires whose positions are fixed with respect to the location and orientation of light system itself. We do not handle dynamic lights such as cars, rotating or blinking signs, or search lights.

1.1 Specific Contributions of the Author

This thesis is the result of collaborative work in which the author participated significantly. The PL tool itself was developed by the author and one collaborator. The graphical user interface portions discussed here were predominantly developed by the author as part of larger tool that was developed on a capstone software development

[4] http://www.flickr.com/photos/sidstar/276225944/sizes/z/in/photostream.

team. The author obtained the manufacturers information on light intensity distributions and developed C-Sharp code to parse the data, validate the data, and use it in order to calculate the irradiance that reaches any point in space from a light source. This code, translated into C, is the backbone of the PL tool. The PL tool has three main functions; light system placement, large scale irradiance maps, and detailed light-map generation. The initial code to generate lighting properties for GIS vector data layers and placing lights was developed using pair-programming with one collaborator and the author. The second function (large scale irradiance maps) of the PL tool was prototyped by the author, including linking irradiance values to GIS raster and vector data. The prototype was extended and integrated into the main PL tool by a collaborator. The detailed building light-maps involved automatic generation of texture coordinates and shadow calculations which were not done by the author, however the author contributed to detailed shadow maps by providing code that generated 'radiant areas' and linked them to images that provide the shape and colors of the areas. Maintenance, testing, and bug fixes to this and all parts of the PL tool were done by the author, in close communication with a commercial partner.

The remainder of this manuscript is formatted as follows: Section 2 reviews the relevant prior art. Section 3 discusses the implementation of Procedural Lighting (PL) tool. Section 4 presents physically and geographically accurate irradiance maps for simulating the effects of millions of lights over large areas. Section 5 focuses on a more detailed approach that can be applied to smaller regions of interest, and includes details like pre-computing the effect of radiant areas such as windows and shadows. Section 6 presents the results of the PL tool applied to several areas, and finally section 7 concludes the thesis.

2 BACKGROUND AND PRIOR ART

Computer graphics practitioners have long been interested in building digital models of cities [2, 3] but there is a lack of tools for connecting light systems with existing city models. A light map [4], which is a raster texture map containing incident light (irradiance) of surfaces is an existing solution for storing complex lighting effects caused by global illumination and other computationally expensive procedures in order to speed up rendering[5]. We use these to capture the effect of many light sources in nighttime urban scenes. However, the positions and properties of light sources are hard to determine since there are millions of light sources, such as street lights, traffic lights and window lights. There are some tools which can help city planners and art designers develop and preview light effects over a 3D building models [5, 6] . Unfortunately, there is no tool which can automatically simulate nighttime scenery over a large-scale urban area. The following sections discuss geographic information systems, the background methods of large-scale city modeling, light map, and tools helping design lighting effect over a 3D building in a city.

2.1 Geographic Information Systems

A geographic information system (GIS) is a database of spatial features used in mapping. GIS data is arranged into multiple *layers*, where each layer can be a raster layer or a vector layer that has a single type of feature. Raster layers can hold *raster* data, which is a two dimensional array of tuples of data, such as aerial photographs, satellite imagery, digital elevation maps, or hyperspectral imagery. Vector layers hold geometric features, where features are geometric shapes such as points, closed polygons, or open polygons (polylines). A key feature of a GIS is that it maintains information on a *projection* that maps some two or three dimensional coordinate system to points on the surface of the earth. For example the most common coordinate

[5] We use the term light-map loosely; our light-maps contain physically accurate irradiance values, whereas they are sometimes used in graphics as 0-1 values indicating the percentage of light that reaches a surface.

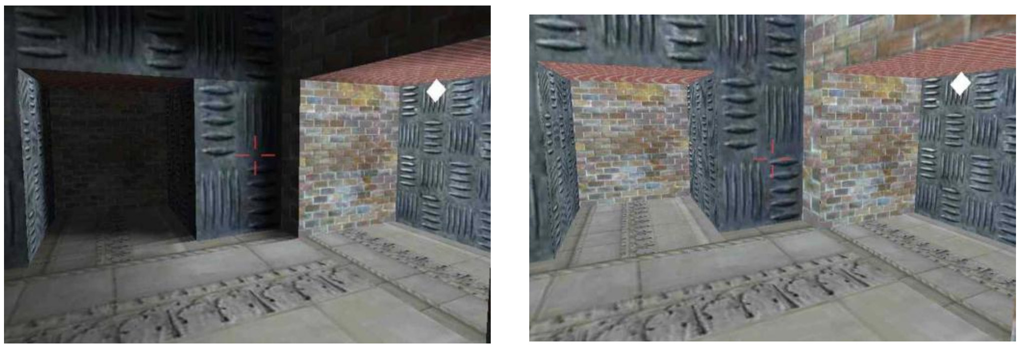
reference system (CRS) is known as WGS84 – the coordinates of geometric features are interpreted as longitude (x-axis) and latitude (y-axis) and the WGS84 projection relates the coordinates to an actual location on an oblique spheroid approximation to the shape of the Earth. Each layer (raster or vector) has an associated CRS. Raster data also maintains an additional geo-transform that indicates where the rectangular block of raster information is located within the CRS. We call a raster layer with an associated CRS and geo-transform a geo-referenced raster layer. In addition to a CRS, each vector layer contains a set of field definitions that describe the names and types of a list data values that can be associated with each feature in the layer. Field definitions are like columns in a database table, each feature is a record, and the layer itself is like the table with an extra hidden geometric column which holds the feature shape. Each layer can use different fields, and they often do. For example a key issue we address is the lack of a consistent set of field definitions describing important aspects of roads, such as the road width, whether the road has a median, sidewalks, and also any information on how the road should be lit. Our PL tool takes GIS layers as input, and we also produce GIS layers as output. We generate a copy of an input street vector layer that has all of the fields we need in order to place lights. We generate a vector layer with point features for each light source, and add fields for the lights heading and an light-system ID for the actual light system to at that point. We also produce many geo-referenced raster tiles which hold pre-computed irradiance values that cover large urban areas.

2.2 Large-scale Urban Modeling

In the past, urban planners created city models by wood, cardboard, paper and glue from manual measurement. Nowadays, new technologies in computer graphics and computer-aided design (CAD) offer powerful tools for creating, visualizing and interacting with digital urban models. For example, CityEngine [3] enables the efficient creation of large-scale 3D city models with the procedural modeling approach. These applications acquire data such as spatial database, geographic information systems (GIS), and shape grammar rules as input information and create real-world building structures automatically. The created large-scale digital models can be used not only in urban planning but also be used as film and game development. However, among these generated large-scale urban models, the information of cultural light resources, such as street light, window lights, and lights from traffic signs are limited. Therefore, it is a huge challenge to simulate vivid nighttime city models even if the city models have been generated. Furthermore, these tools generate new cities – our aim is to add lights to existing city models. Manual measurement and entry of light sources for an entire urban area is expensive and time consuming; so automatic extraction or synthesis of light properties and positions from a large-scale urban area, which the PL tool does, will have a great impact on large-urban modeling, especially for nighttime urban modeling.

2.3 Light Mapping

3D applications have been applied into various areas, such as game design, filmmaking, CAD tool for architecture or automobiles, and military simulations. In order to achieve more beautiful and vivid images, physical phenomena, for instance, lighting and shadowing, play an important role in 3D computer graphics. However, these physical effects are very complex and take much effort for simulation. Gouraud shading [7] was one of the most used algorithms especially on fixed-function graphics hardware without fragment/pixel shader support. Gouraud shading is a fast approximation to Phong shading that works by calculating the lit color of a 3D object at the vertices (or coners) and interpolating the colors of the faces of the model. One obvious problem with Gouraud shading is the lighting model can introduce difficulties with shadows. Therefore, light mapping is introduced for rendering lighting and shadowing without sacrificing high performance [8]. Quake was the first computer game to use light mapping to augment rendering [9]. Since many objects and light sources are mostly fixed, the lighting and shadowing calculation can be saved by pre-computing them and mapping them on to the surface at runtime. The lighting information is stored as textures because texture



(a) (b)

Figure 4. Examples of light mapping; (a) a 3D world with light and shadow; (b) a 3D world without light and shadow.

mapping can be done in hardware. Figure 4 (a) and (b)[6] show the effect of light maps on a course polygonal model lit by a single point light source.

In order to simulate a large-scale nighttime urban area, it is possible to implement lighting and shadowing effect with the light mapping approach because of its high performance. However, there is still in lack of light positions and properties information as described in section 3. Although previous research on mapping city lights with nighttime data from Defense Meteorological Satellite Program (DMSP) has been implemented by Elvidge and his colleagues [10], the resolution is not sufficient to create city models. The DMSP Operational Linescan System (OLS) has a unique capability to detect low levels of visible and near-infrared (VNIR) radiance at night and city lights could be mapped by nighttime data from the DMSP Operational Linescan System. However, the resolution of DMSP data is just 1-km per pixel which is too low to match light systems with a street vector layer in GIS data. There remains a need for another method which can extract properties and positions of lights with higher precision and match the light information with street vector layer. The PL tool aims to fill that gap by generating plausible lights and irradiance maps at a high enough resolution to match the sharpest daytime imagery.

[6] http://www.flipcode.com/archives/Light_Mapping_Theory_and_Implementation.shtml

2.4 Tools for Simulating Lighting Effects

Previous research on a night sky model has been implemented by Henrik Wann Jensen and his colleagues [11], which could be used in simulating a nighttime city based on a physical model. However, this model excluded cultural light resources in an urban area, such as street lights and building window lights. Figure 5 and Figure 6 illustrate that the light effects should be taken into consideration in order to simulate an accurate night urban scene; a significant amount of the light visible in Figure 6 is absent in Figure 5 which neglects lights from the building windows streets, and other urban light sources. The PL tool presented here would allow designers to place radiant areas over the windows, to generate streetlights according to existing GIS street vector data, and to preprocess the model to include the irradiance and radiance caused by these lights as a lightmap that can be rendered over the models in real time.

Work in light designing and placing light over building models has included several projects that allow users to explore a plausible nighttime scene. Dorsey and his colleagues [5] have implemented a tool for simulating opera lighting and projection. They presented an application which simulates the optical effects of scenic projectors and addresses a solution to the distoriton problem produced by angular projection. Fukuda and his colleague [6] have implemented a support system for city plans which enable the realistic view of night scenes based on virtual reality (VR) technology. Unfortunately, these techniques either only used indoor light sources or simulated a part of building model, which cannot be used in a large scale of urban area with outdoor light sources. We aim to simulate a large-scale urban area with light mapping textures which can be rendered in real-tme.

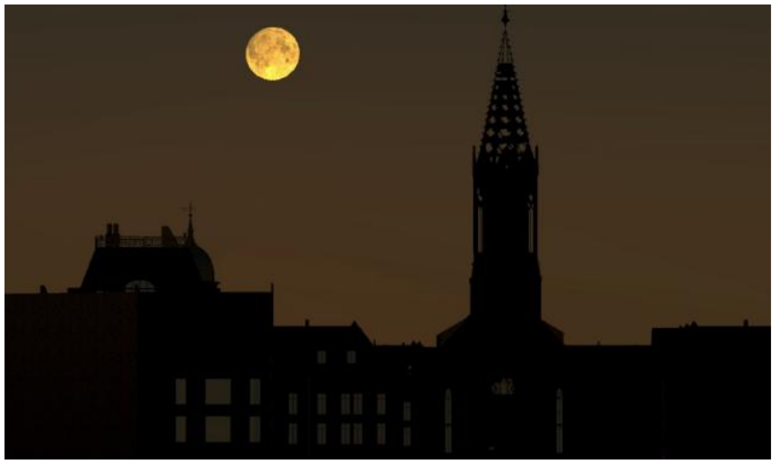


Figure 5. Simulation result of the low moon setting over a city by Jensen [11].

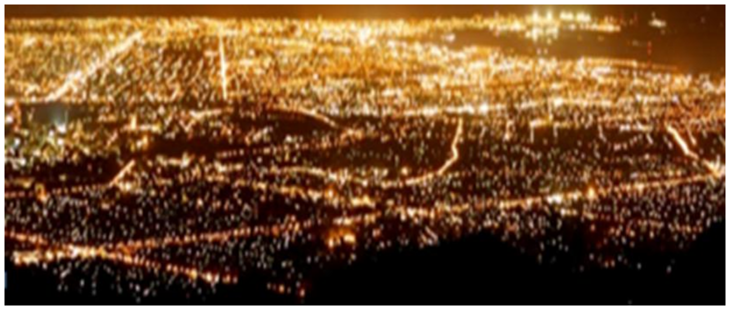
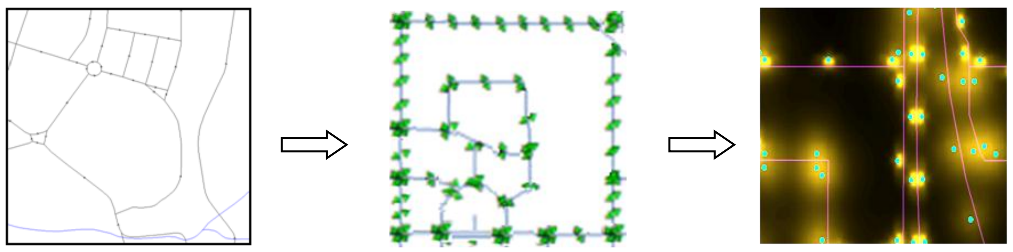


Figure 6. A real night urban scene in Phoenix, AZ.

3 GENERATING STREETLIGHT CONFIGURATIONS

We present a two-part process of generating light systems that are “geo-typical,” which means their locations and properties follow the a distribution that is expected for an area. First, given a geographic information systems (GIS) vector layer containing street map data, lighting schemes for each street are determined, and then properties and positions of candidate light points are suggested to match the street vectors. Second, given the positions of candidate lights, light system instances are fetched from a Light System Database (LDB), so that the properties match the lighting schemes determined for the streets. In the following section, we will discuss how we predict the positions and properties of lights. Figure 7 demonstrate how light systems are generated and rendered by a given GIS street vector layer and the LDB.

The overall process of generating streetlights is presented in Figure 7. First, a table is created (absent from the figure) that maps field names in a GIS street vector layer to fields names that are understood by the system. These tables are combined with a Bayesian network (section 3.3) which can be used to fill-in any required data that was missing from the street vector layer with plausible values. Second, every street vector is processed and a new street vector layer is produced, this time with all of the required information on how to light the streets. Third, we generate a new GIS vector layer of point features for lights along each street, setting the heading and light system ID fields for



(a) (b) (c)

Figure 7. Generating light systems by the Procedural Lighting (PL) Tool; (a) input GIS street vector data; (b) candidate light points match the GIS street vectors; (c) candidate lights rendered by fetching light properties from a Light System Database (LDB).

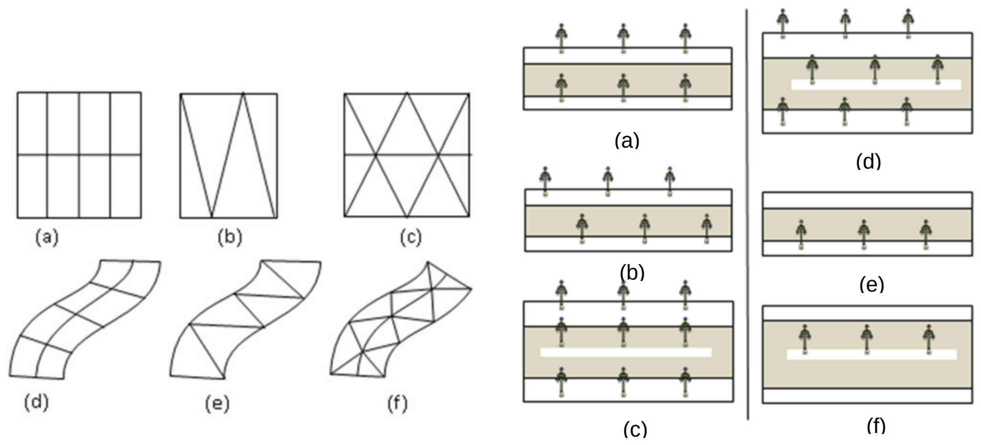


Figure 8. Light points grids; Left (a) a 3x5 rectangular grid; (b) a 2x3 triangular grid; (c) a 3x3 triangular grid; (d) a curved 3x5 rectangular grid; (e) a curved 4x1 triangular grid; (f) a curved 4x3 triangular grid; Right (a) double sided rectangular grid; (b) double sided triangle grid; (c) double sided rectangular grid with three columns; (d) double sided triangle grid with three columns; (e) Single sided rectangular grid; (f) single sided rectangular grid with three columns.

each point. The result is a map of streetlight locations that can be used to generate irradiance layers as described in section 4. The modified street map and light points can also be edited by any GIS tool to accommodate geo-specific information (such as the location of an airport).

3.1 Light Placement Suggestions

An assumption of our approach is that light systems are placed by city planners or architects in an attempt to space the systems in a regular, periodic manner. However, in order to accommodate changes in terrain or other constraints, some light systems will deviate from a perfectly periodic arrangement. Therefore, we generate suggested point locations and types for light systems by jittering one of the following configurations: (1) points on a rectangular grid, (2) points on a triangular grid, (3) points on a curved rectangular grid, or (4) on curved triangular grid. The various grids are illustrated in Figure 8. Figure 8 left shows a palette of light patterns. Figure 8 right shows different type of grid used in our Procedural Lighting (PL) tool.

3.2 Street Properties Suggestions

The input to our system is a GIS vector layer with linear features representing streets. We call this the street vector layer. In GIS tools, a "linear" feature is a sequence of straight line segments connect to form a piece wise linear open curve. Each feature is associated with a set of fields, the fields associated with street vectors vary from one dataset to the other, but very often there is some field which indicates the intended use of street (e.g. major arterial, minor arterial, collector streets, freeway, etc.). Some but not most datasets may also contain information which may include the number of lanes, the width of streets, or whether a median is present. These fields can exist, however these fields are rarely all defined. Even when the information is present, the field may have different name in GIS datasets and the value may be interpreted differently (e.g. an enumeration versus an integer to indicate the street use). Before we can determine the positions of lights points, we need to normalize the input so that it always includes GIS fields that the PL tool understands for describing how to layout lights along streets. A rule based system such as prolog, could be used to infer missing properties using a set of logical clauses. However, we also require the missing properties follow a prescribed random distribution allowing results being plausible. For instance, if every arterial of streets look the same way, it would look too artificial. Therefore, we seek a way to generate missing street properties by sampling a probability distribution which is created by a domain expert in order to capture the geo-typical lighting trends in a given area. Our solution is to sample a Bayesian network (described below), which generalizes a rule based system by allowing multiple possible outputs of the light properties for the same input.

3.3 Sampling Bayesian Network

A Bayesian network, also called belief network[7], is a directed acyclic graph (a DAG) where each node is a random variable, and directed edges connect independent variables to the other variables they dependent on. In our application, there will be a node for each GIS field we desire in a street vector layer, there will also be nodes created by a user to map the input street layer fields to other variables that the PL tool uses to place lights. In addition to node for input GIS fields and output GIS fields, there may be additional nodes created in order to capture hidden variables, such as the number of lanes in a road, which are neither input nor required as output but may still help with estimating the required fields, such as the road width. A Bayesian network can be used to determine the joint probability or likelihood of any set of fields (a sample), or to generate samples according to the same distribution of our observations. The purpose of a Bayesian network is to reduce the amount of data that needs to be used to describe the geo-typical properties of streets by keeping track of which nodes are independent of each other. The alternative would be to create an exponentially large table that assigns a probability to every combination of field values, called a joint probability table. Unlike a joint probability table, a Bayesian network exploits conditional independence and often requires far fewer values to be specified in order to represent a given joint probability distribution. In a Bayesian network, one needs only to define a conditional probability table at each node of the graph, which assigns a set of probabilities for the nodes value conditioned on each combination of values that its parent nodes could hold. Often the conditional probability tables can be kept small, and this makes them better suited for specifying the distribution of geo-typical properties because fewer parameters need to be set. In addition, the graphical structure of the Bayesian network has convenient interpretation: node A is connected to node B if B is influenced by A. This kind of cause and effect reasoning is typically not hard to guess when considering a particular geo-

[7] Note that they are called networks even though weights are not associated with the edges.

graphic region. The PL tool is designed so that Bayesian networks can be selected from a list of prebuilt networks at run-time, or mixed with tables created by a user. This way the networks themselves could be developed and distributed and updated, and sold separately by a commercial vendor.

It is well known that generating samples from a Bayesian network that exactly follows the joint distribution is an NP-complete problem. A proof is provided by Chickering [12] based on a polynomial time reduction to the well-known Boolean satisfiability (SAT) problem. For a detailed proof the reader is referred to Chickering's paper, but the idea is that a Bayesian network can be constructed in polynomial time to match a logical expression by choosing only 0 or 1 probabilities in the conditional probability tables associated with each node in the graph. If a sample from the table could be generated in polynomial time then the values would satisfy the original logical expression. Therefore a Bayesian network sampler could solve the SAT problem (which could be used to solve any other NP problem), and therefore sampling a Bayesian network is at least as hard as the NP-complete SAT problem.

In spite of the fact that sampling a Bayesian network is a NP-complete problem, it is often the case that samples can be generated quickly and there are polynomial time

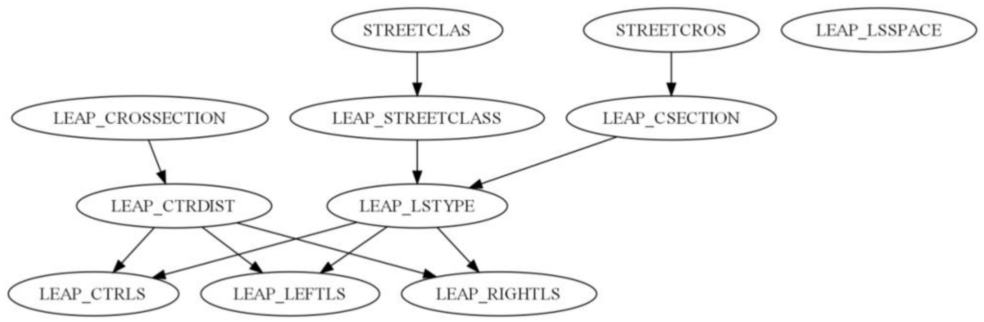


Figure 9. The Bayesian network of light system properties. This network introduces intermediate nodes not directly required by the tool, including CROSSECTION, STREETCLASS, and CSECTION nodes. The nodes are designed by a user of the tool. SREETCLAS and SREETCROS are actually GIS fields stored in GIS database.

approximations to the solution. Since there is a large amount of uncertainty involved with manually specifying the distribution of geo-typical properties, an approximate solution is well suited to our application. There are various commercial high-quality Bayesian network samplers that have been carefully tuned to give exact results when possible or else to give good approximations, however for our application these are probably overkill. We merely need a plausible result and it is believed a simple approximate sampling algorithm will work nicely. We use a likelihood weighted sampling method [13] which first generates a set of N possible samples, along with their estimated likelihood according to the Bayesian network. Samples are generated so that any field value already stored in the GIS dataset is forced to have the same value in each sample. The likelihood of each sample is calculated exactly at the same time that the sample is generated using a depth-first approach that runs in linear time on the number of samples. Once N samples are generated, we select one randomly according to their likelihood. More details of Bayesian network and likelihood weighted sampling including pseudocode are presented in [14] and [13]. Figure 9 shows a Bayesian network for estimating light system properties. Nodes STREETCLAS and STREETCROS are GIS fields in an input GIS database, with small tables created by a user in order to map the values to the LEAP_STREATCLASS and LEAP_CSECTION nodes. The rest of the Bayesian network is generic and can be reused for any input. Nodes LEAP_CROSSECTION, LEAP_STREETCLASS, LEAP_CSECTION, LEAP_CTRDIST, LEAP_LSTYPE, LEAP_CTRLS, LEAP_LEFTLS, and LEAP_RIGHTLS have joint probability tables dependent on their upper level nodes. In our system, the network is represented by a set of CSV tables under a folder, where each table lists the field it estimates as the first column (the dependent variable), followed by a list of independent variables. Figure 10 shows an example table of _LSTYPE variable. Each row of the table ends with a number that is proportional to the likelihood of that set of street properties occurring in a given area. The numbers in Figure 10 were generated by counting the number of observations in a number of GIS datasets. At

_LSTYPE	HAS_MEDIAN	WIDE_ROAD	174
C	F	F	0
DS	F	F	15
DSS	F	F	15
DSCS	F	F	0
DSC	F	F	0
SS	F	F	20
C	F	T	0
DS	F	T	10
DSS	F	T	10
DSCS	F	T	0
DSC	F	T	0
SS	F	T	3
C	T	F	10
DS	T	F	15
DSS	T	F	15
DSCS	T	F	5
DSC	T	F	5
SS	T	F	5
C	T	T	5
DS	T	T	10
DSS	T	T	10
DSCS	T	T	10
DSC	T	T	10
SS	T	T	1

Figure 10. A table of joint probability for _LYSTYPE light system variable.

runtime, rows of the table that do not match known properties of streets are removed and the remaining likelihoods are normalized sum to one. This method of providing input to the system is aimed at making it easy to set up a table by using estimated frequencies of different lighting types or by the more laborious process of actually counting occurrences. Users can also tune the joint probability tables of intermediate nodes, such as LEAP_CROSSECTION, LEAP_STREETCLASS, LEAP_CSECTION, LEAP_CTRDIST, LEAP_LSTYPE, LEAP_CTRLS, LEAP_LEFTLS, and LEAP_RIGHTLS, and generate the desired distribution of positions and properties of lights which match the GIS street vector layer. However, we envision this process being done only a few times to create a catalog of 'geo-typical' distributions that can be selected by users at run-time.

4 LIGHTS FOR TERRAIN

An effective method of creating a nighttime urban scene is rendering street lights generated from a GIS vector layer describing the locations, headings, and unique identifiers of light systems as described in section 3. In order to render a large urban nighttime scene in real-time, the naïve approach of placing a light-source at each point has issues because the most common rendering hardware in real time only supports limit number of sources, and the standard rendering pipeline does not lend itself to lights other than directional, point, or spot lights. For example, OpenGL only requires eight light sources [4]. Therefore, we generate a light map over a terrain and store incident light energy (irradiance) so that lighting can be done at interactive rates using texture compositing. The key point of implementing lights for terrain is determining the irradiance at any point in space using the light intensity distribution for each light source. The following section, we discuss what light intensity distribution is and how we use it.

4.1 Light Intensity Distribution

A light intensity distribution (LID) captures the portion of energy emitted from a light source in any given direction. LID's are often visualized using goniometric diagrams

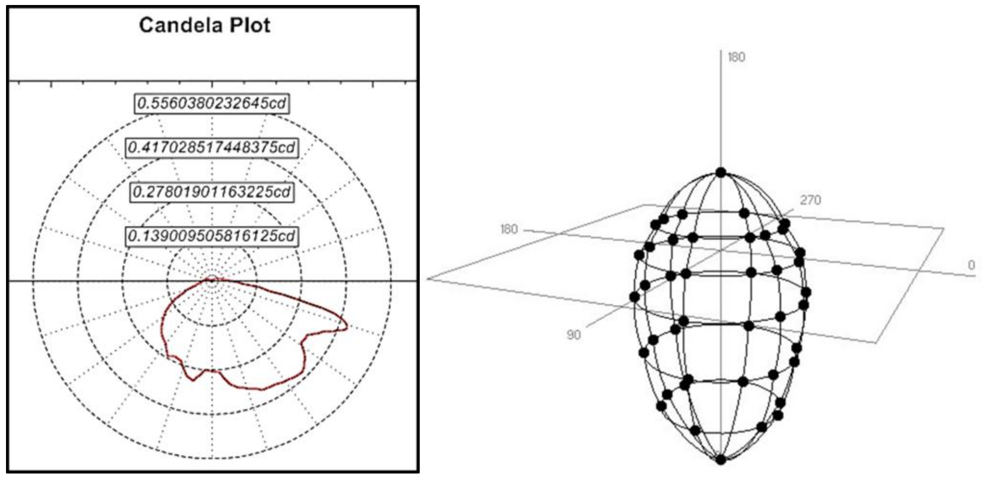
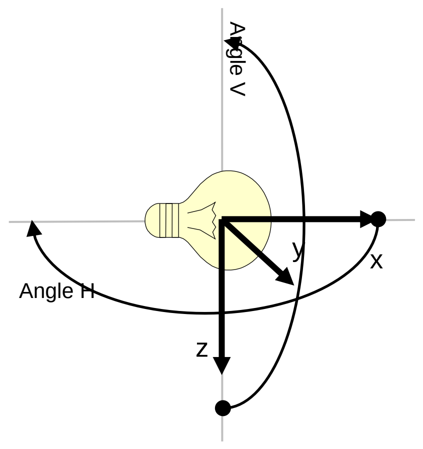


Figure 11. Visualizations of light intensity distributions; (left) A goniometric diagram of candela plot for a IES Photometric file, (right) a photometric web.

(Figure 11) which is used widely in the light industry because of easy interpretation of a non-uniform intensity distribution of light [15]. The goniometric diagram shows the light intensity (in candelas) emitted from the light source at any angle. Figure 11(left) is a single slice through the 3D LID, unless the light system is radially symmetric then each slice would look a bit different. In Figure 11(right) another LID is shown using a photometric web, which is also a polar plot of the candelas but it captures the entire 3D distribution. LIDs have found use in architectural planning and landscaping in order to place lights for the best effect, and it has also found use by city planners in choosing street light locations in order to avoid dark patches between streetlights. Some CAD tools, such as AutoCAD and 3DS Max, provide methods to import LID information and render complex lights, however these tools do not create the physically accurate irradiance maps needed for NVG simulation. The LID determines the pattern of light projected onto the ground or buildings (called ‘lobes’), and when rendering streetlights for NVGs one can easily see the difference between accurate LIDs verses simple point or spot lights. The illuminating Engineering Society (IES) provides a standard file format for photometric files



(a)

Candela Values		Horizontal Angles (degree)			
		0	60	120	180
Vertical Angles (degree)	0	20686	20686	20686	20686
	30	22065	29355	19898	10580
	45	20883	30655	16057	11880
	60	12609	28271	10658	8491
	75	7723	35460	3408	3231
	90	2561	4137	1911	1970
	125	0	0	0	0
	180	0	0	0	0

(b)

Figure 12. Coordinate System and candela values in a IES file; (a) the right-handed coordinate system in IES photometric data; (b) candela values for given vertical and horizontal angle in a IES file.

[16], and they have also measured the LIDs for many manufacturers' lights. IES files provide the amount of light energy (candelas) emitted in a given direction from a light source. The information is stored in an irregularly spaced grid with grid-points corresponding to directions emanating from the light source as Figure 12.

4.2 Finding the Energy in a Given Direction

An IES file contains metadata about a light source, and then a table of luminance values measured from a light source. Our aim is to determine the amount of energy reaching any point in space from a light source. Figure 12 (a) shows the coordinate system used when measuring LID's in IES files. Angle V and Angle H present the direction of vertical angles and horizontal angeles. The symbols, x, y, and z are the Cartesian coordinate system in IES Photometric data. Note that the positive z-axis points downward. The IES files contain a table that represents an irregularly spaced grid of energy emitted at a given H and V angle.

IES Lobe files list only half of the sphere; the other half is obtained by symmetry about the $y = 0$ plane. The most challenging part of understanding the specifications of an IES lobe is translating form horizontal and vertical angles to unit vectors in 3D. This is especially confusing since the local coordinate frame for a luminaire seems to be oriented with the z-axis pointing down toward the ground (Figure 13). We use formula (1) to convert vertical angles (V) and horizontal angles (H) into 3D unit vectors and visualize

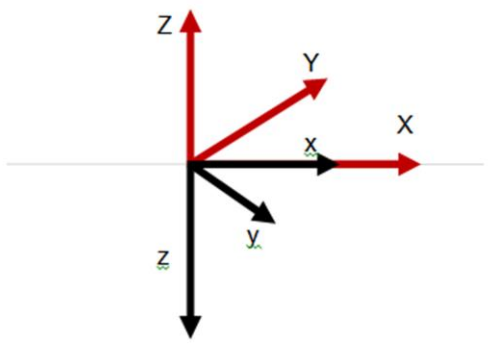


Figure 13. The relationship of up toward and down toward coordinate system.

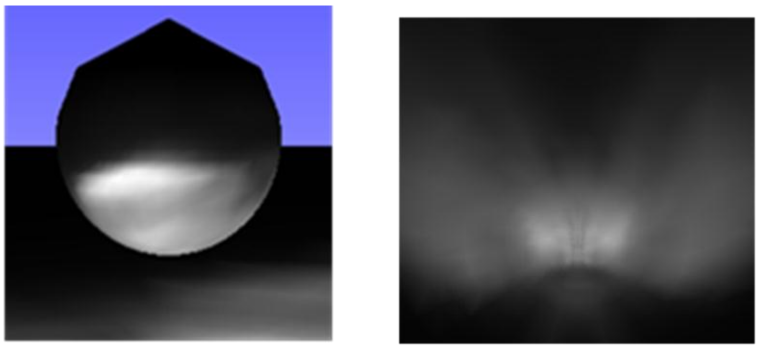

(a) (b)

Figure 14. Visualizing lobe pattern in IES file; (a) the amount of energy which passes through a point on the surface of a sphere centered at the light origin; (b) visualization of the lobe as appeared on the ground.

the lobe (see Figure 14 (a)) as projected on the ground (see Figure 14 (b)), where d is the distance between lobe center and ground plane:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = d \times \begin{bmatrix} \tan V \times \cos H \\ \tan\ V \times \sin H \\ 1 \end{bmatrix}, \quad \cos V > 0 \tag{1}$$

Since the IES lobe data includes only one half of the sphere, we use symmetry to render the other half by replacing horizontal angle H by $-H$ in the formulas above. These expression can also be used to solve for V and H, when x, y, and z, are components of a unit vector in the lights local coordinate system.

The formulas are:

$$\begin{gathered} V = \mathrm{acos}(z) \\ \mathrm{H} = \mathrm{atan}\left(\frac{\mathrm{x}}{\mathrm{y}}\right) = \ \mathrm{atan2}(\mathrm{x}, \mathrm{y}) \end{gathered} \tag{2}$$

After the 3D unit vectors have been computed, I can use formula (3) and translate coordinate system, x, y, and z, to coordinate system where z points upwards, X, Y, *and* Z, (Figure 13).

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} x \\ -y \\ -z \end{bmatrix} = d \times \begin{bmatrix} \tan V \times \cos H \\ -\tan\ V \times \sin H \\ -1 \end{bmatrix} \tag{3}$$

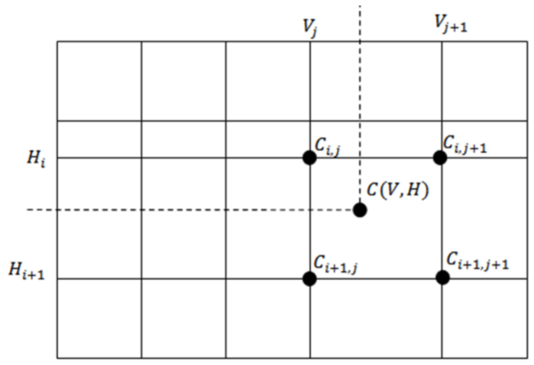


Figure 15. Bilinear interpolation of candela value $C(V,H)$ given V and H in the rectangle of V_j, V_{j+1}, H_i, and H_{i+1}.

Finally, given values V and H we aim to determine the candela value using bilinear interpolation of the candela's listed in the IES lobe data. Let $C_{i,j}$ be the entry in the grid (see Figure 15) with horizontal angle H_i and vertical angle V_j where i is a row index in the table and j is a column index. Then to determine the candela value $C(V,H)$ for arbitrary V,H we first find the interval H_i,H_{i+1} that contains H and likewise find the interval V_j,V_{j+1} that contains V. Then let $s=\frac{V-V_j}{V_{j+1}-V_j}$, $t=\frac{H-H_j}{H_{j+1}-H_j}$, and

$$C(V,H)=(1-t)\times\left[(1-s)C_{i,j}+sC_{i+1,j}\right]\ +t\times\left[(1-s)C_{i,j+1}+sC_{i+1,j+1}\right]. \qquad (4)$$

This is bilinear interpolation on the candela-grid provided as part of the IES file. The process is illustrated in Figure 15.

4.3 Light System Coordinate Frames

In order to determine the irradiance at a pixel in a geo-referenced image, we need to do a number of coordinate transformations. We will assume that the 'World' coordinates are determined using a GIS projection, and that the mapping from pixels to world coordinates is available through a library such as GDAL/OGR [17]. In order to determine the irradiance at a pixel we need to work our way from the raster layer to the emitter's coordinate systems. We introduce the transform matrices:

$$T_{world}=\ \text{Transform from pixel to LightPoint layer}$$

$$T_{ls}=\ \text{From Layer to Light-System coordinates}$$

T_{pole} = From light system to top of pole

T_{mast} = From top of pole to end of mast

T_{lum} = From end of mast to center of lobe

T_{lobe} = Make the z-axis point down as per IES specs

The transformations will represented as 16 element (4×4) homegenous matrices. We ultimately need the final matrix which takes us all the way from the pixel in a raster layer to a point in the emitters reference frame:

$$\begin{aligned} \mathbf{p}' &= \left(T_{world} \times T_{ls} \times T_{pole} \times T_{mast} \times T_{lum} \times T_{lobe}\right)^{-\mathbf{1}} \times \mathbf{p} \\ &= T_{lobe} \times T_{lum}^{-1} \times T_{mast}^{-1} \times T_{pole}^{-1} \times T_{ls}^{-1} \times T_{world}^{-1} \times \mathbf{P} \end{aligned} \qquad (5)$$

where we use the fact $T_{lobe} = T_{lobe}^{-1}$. The LPRD data uses Euler angles (heading, pitch, roll) to specify changes in orientation. We adopt the following notation for Euler angles: Let θ_z indicate the heading, or the angle about the z-axis. Let θ_x the pitch, or elevation; the angle above the x-axis. Let θ_y denote the roll, or the angle around the y-axis. We apply these rotations in ZXY order; in OpenGL we would issue the commands:

```
void rotateEuler(θz,θx,θz){
        glRotated(Θz,0,0,1);
        glRotated(Θx,1,0,0);
        glRotated(Θy,0,1,0);
        }
```

The resulting rotation matrix R is

$$R = \begin{bmatrix} c_1c_3 - s_1s_2s_3 & -s_1c_2 & c_1s_3 + s_1s_2c_3 & 0 \\ c_1s_2s_3 + s_1c_3 & c_1c_2 & s_1s_3 - c_1s_2c_3 & 0 \\ -c_2s_3 & s_2 & c_2c_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (6)$$

$$, \quad \begin{aligned} c_1 = \cos(\theta_y), c_2 = \cos(\theta_x), c_3 = \cos(\theta_z) \\ s_1 = \sin(\theta_y), s_2 = \sin(\theta_x), s_3 = \sin(\theta_z) \end{aligned}$$

To undo the rotation we would use OpenGL commands:

```
void unRotateEuler($\theta_z,\theta_x,\theta_z$){
        glRotated($-\Theta_y$,0,1,0);
        glRotated($-\Theta_x$,1,0,0);
        glRotated($-\Theta_z$,0,0,1);}
```

The result inverse rotation matrix is

$$R^{-1} = \begin{bmatrix} c_1c_3 - s_1s_2s_3 & c_3s_1s_2 + c_1s_3 & -c_2s_1 & 0 \\ -c_2s_3 & c_2c_3 & s_2 & 0 \\ c_1s_2s_3 + c_3s_1 & s_1s_3 - c_1c_3s_2 & c_1c_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

After rotation the LPRD database often includes some translations; for instance a translation to the end of a pole or to the end of a mast. The translation happens in the rotated coordinate system, so in OpenGL one would write

```
void applyLprdTransform($\theta_z,\theta_x,\theta_z,t_x,t_y,t_z$){
        rotateEuler($\theta_z,\theta_x,\theta_z$)
        glTranslate($t_x,t_y,t_z$)
}
```

The resulting transformation matrix is

$$T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1c_3 - s_1s_2s_3 & -s_1c_2 & c_1s_3 + s_1s_2c_3 & 0 \\ c_1s_2s_3 + s_1c_3 & c_1c_2 & s_1s_3 - c_1s_2c_3 & 0 \\ -c_2s_3 & s_2 & c_2c_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

To undo the transform in OpenGL we write

```
void unapplyLprdTransform($\theta_z,\theta_x,\theta_z,t_x,t_y,t_z$){
        glTranslate($-t_x,-t_y,-t_z$)
        unrotateEuler($\theta_z,\theta_x,\theta_z$)}
```

and the resulting matrix is

$$T^{-1} = \begin{bmatrix} c_1c_3 - s_1s_2s_3 & c_3s_1s_2 + c_1s_3 & -c_2s_1 & -t_x \\ -c_2s_3 & c_2c_3 & s_2 & -t_y \\ c_1s_2s_3 + c_3s_1 & s_1s_3 - c_1c_3s_2 & c_1c_2 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

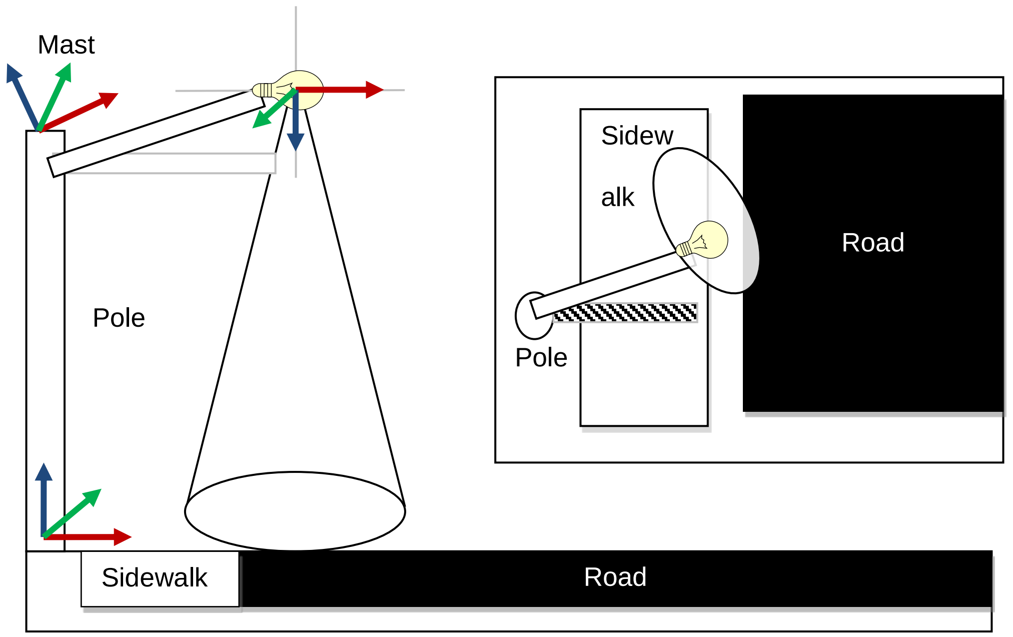


Figure 16. Components of a light system in relation to a street. Within the light system's coordinate system, $+X$ always points in towards the street, $+Z$ points up, and $+Y$ points opposite the direction of traffic.

To our knowledge the LDB database is limited to rigid body transforms (no scaling). The objects have the Euler angles specified as properties, but the translational parts must be inferred based on the type of object (e.g. mast, pole, etc.). Usually $t_x = t_y = t_z = 0$, except for a *pole* we have $t_z = height$, and for a *mast* we have $t_x = length$.

4.4 Region of Influence for Lobes

The lobe of a light source is most often directly or nearly directly below the lights, however do the rotation of the light source or irregularities in the IES values the lobe may appear offset way from the light source, or larger. In order to produce an irradiance map, there are two options; a first option is that we could project quadrilaterals out from the light source onto the raster image, then rasterize and render the quads (this is done in the GUI editor). The problem is that at the edges the quads would become quite dark, and they would project far from the light source. We would need to prune out our threshold the quads before rendering them if we are to have any hope with this method.

This is the method used to render the lobe in Figure 14 (b), and it can be done entirely in OpenGL with the fixed function pipeline (one does not even need shaders). Another problem with this approach is that it does not extend well to cases when the surface (e.g. the ground) has height values associated with it and it is not just a plane.

A second option (the one used in the PL tool) is to determine a bounding box for the lobe in the raster plane, and then visit each pixel in the box and work back to the light to determine the irradiance. One could use calculate a tight bounding box for the lobe by analyzing the candela values and applying a threshold for the smallest visible amount of energy, however we adopt a simpler approach that finds a less-tight bounding box; we calculate the lighting for all pixels within a pre-specified (e.g. 200m radius) square centered at the light source. Figure 17 shows irradiance map of a small portion of Phoenix area illuminated by street lights. Streets are shown as red lines and street light points are presented as blue dots. Figure 18 shows the close view of lights for terrain.

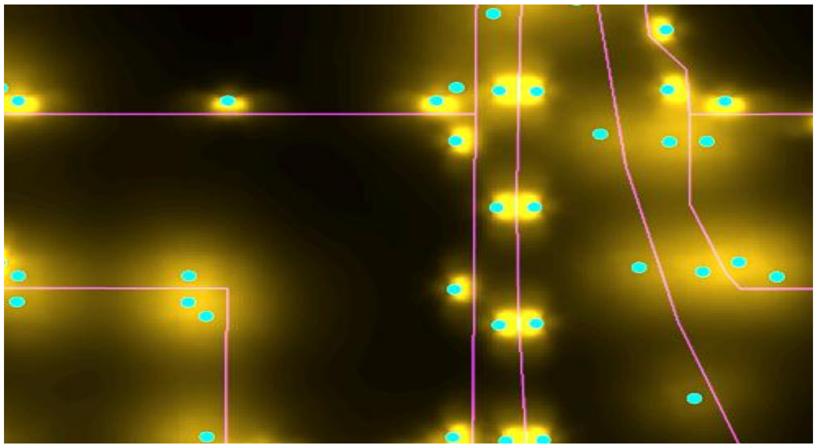


Figure 17. Irradiance values generated from a small portion of a Pheonix area dataset. The contrast has been exaggerated for viewing.

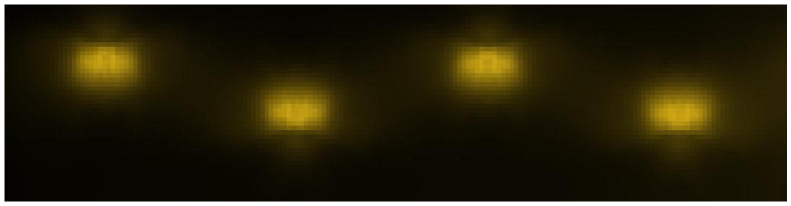
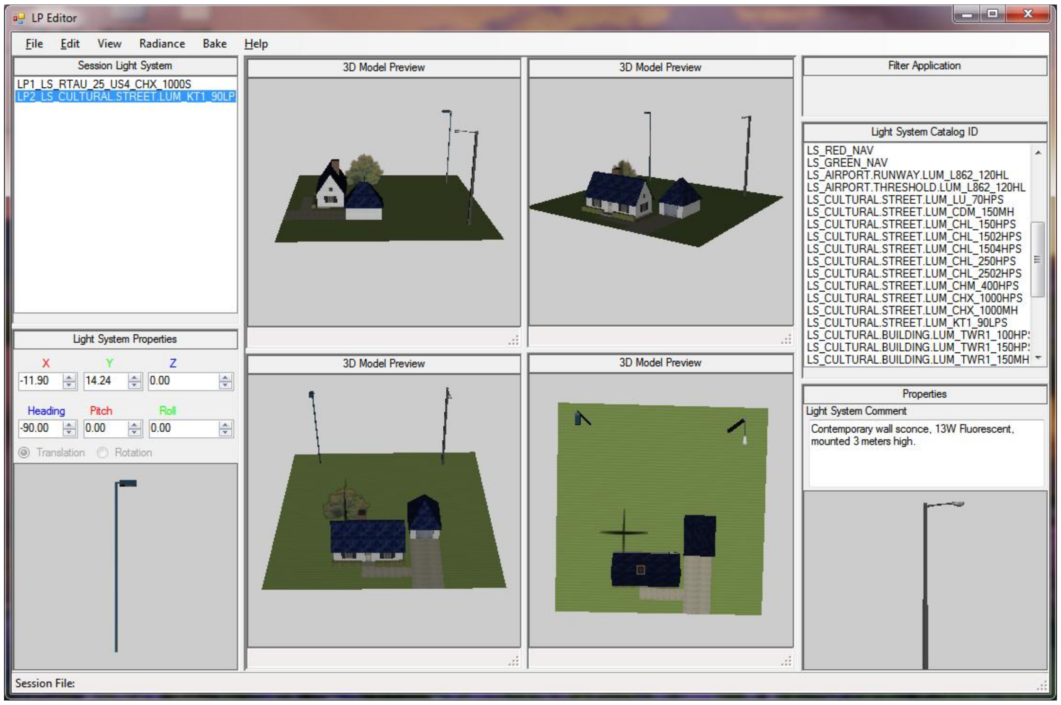


Figure 18. A close-up view of staggered lights.

5 DETAILS FOR BUILDING LIGHTS

It is important for users to have full three-dimensional simulations of the illumination effects in a nighttime urban area. We have implemented a building editor which can generate a light radiance and irradiance maps for details of lights, including street lights, lights attached to buildings, and light coming from area sources like windows. Users can have a 3D experience by exploring our simulation results. There are two types of lights that can be manipulated using our building editor: street light (which includes lights attached to buildings) and window lights. Street lights are selected by the Light Properties Editor (see Figure 19), and window lights are designed by the Radiance Editor (see Figure 20). After street lights and window lights have been placed, users can bake these two types of lights into a light map including reflectivity, radiance and irradiance textures and the generated light map can be used in real time rendering later.



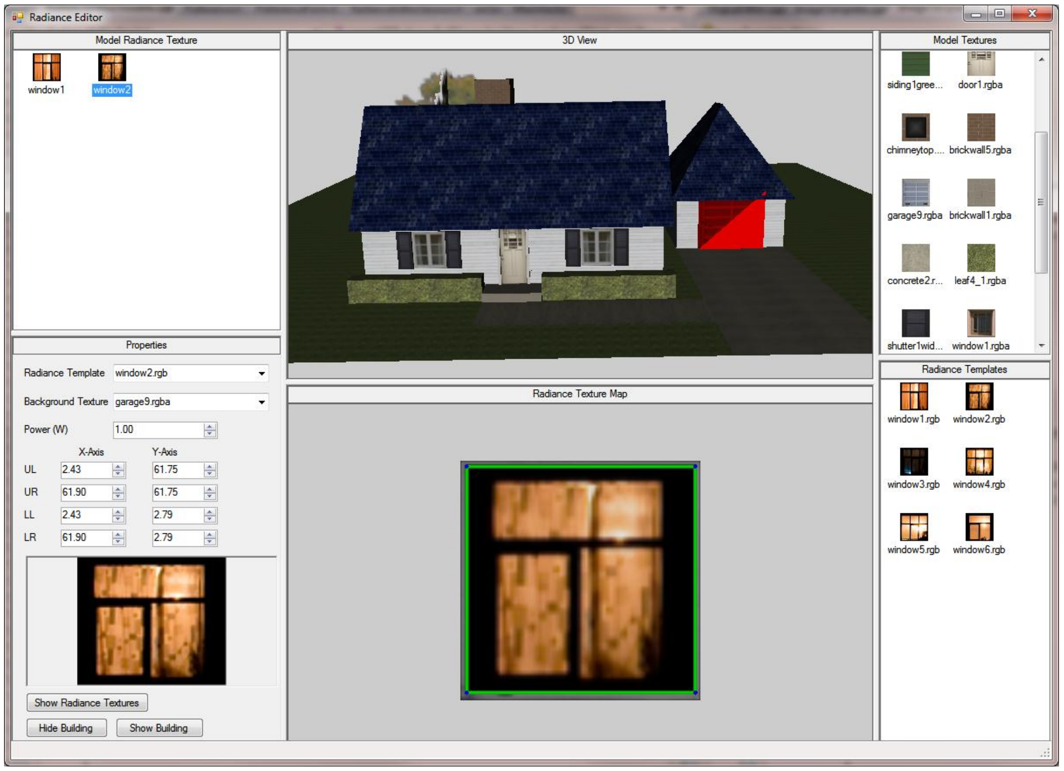
LP1_LS_RTAU_25_US4_CHX_1000S
LP2_LS_CULTURAL.STREET.LUM_KT1_90LP

X -11.90 Y 14.24 Z 0.00
Heading -90.00 Pitch 0.00 Roll 0.00

LS_RED_NAV
LS_GREEN_NAV
LS_AIRPORT.RUNWAY.LUM_L862_120HL
LS_AIRPORT.THRESHOLD.LUM_L862_120HL
LS_CULTURAL.STREET.LUM_LU_70HPS
LS_CULTURAL.STREET.LUM_CDM_150MH
LS_CULTURAL.STREET.LUM_CHL_150HPS
LS_CULTURAL.STREET.LUM_CHL_1502HPS
LS_CULTURAL.STREET.LUM_CHL_1504HPS
LS_CULTURAL.STREET.LUM_CHL_250HPS
LS_CULTURAL.STREET.LUM_CHL_2502HPS
LS_CULTURAL.STREET.LUM_CHM_400HPS
LS_CULTURAL.STREET.LUM_CHX_1000HPS
LS_CULTURAL.STREET.LUM_CHX_1000MH
LS_CULTURAL.STREET.LUM_KT1_90LPS
LS_CULTURAL.BUILDING.LUM_TWR1_100HP
LS_CULTURAL.BUILDING.LUM_TWR1_150HP
LS_CULTURAL.BUILDING.LUM_TWR1_150MH

Contemporary wall sconce, 13W Fluorescent, mounted 3 meters high.

Figure 19. Design street lights by Light Property Editor. Users can pick a street light property from the Light System Database (LDB) at the right dock panel and adjust the position and rotation of street lights through the Light Property Editor.



Radiance Template window2.rgb

Background Texture garage9.rgba

Power (W) 1.00

	X-Axis	Y-Axis
UL	2.43	61.75
UR	61.90	61.75
LL	2.43	2.79
LR	61.90	2.79

Figure 20. Design window lights by Radiance Editor. Users can choose a window or radiant area from or template database, place it on a model, adjust the size and positions of radiant area, and assign the energy emitted from that area through the Radiance Editor

In the following subsections, we will discuss the details of irradiance and radiance and the methods to compute them.

5.1 Irradiance Map

Irradiance is the power of electromagnetic radiation per unit (radiative flux) incident on a surface in W/m^2 [18]. In our building editor, we suppose the irradiance comes from the street light, which project light on an area and the area emitted energy as irradiance (Figure 21). Each street light (or light mounted on a building) is assigned a luminaire from the Light System Database (LDB), containing lobe and spectral power distribution (SPD) of light. The light energy could be computed based on the SPD and projected on the surface. Then the irradiance texture of light map was calculated by multiplying reflectivity

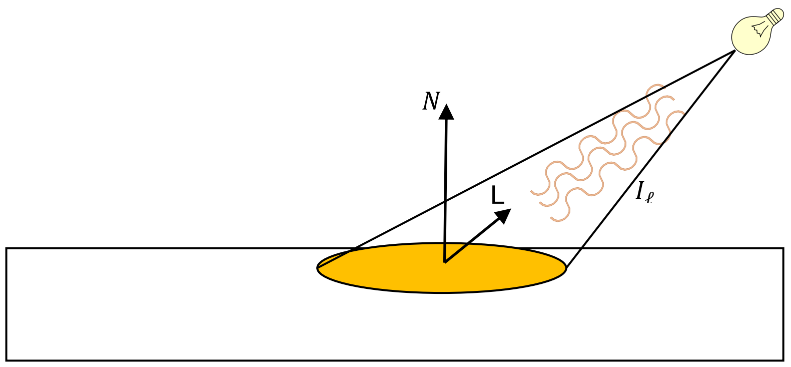


Figure 21. Illustration of irradiance Map. N is the normal vector perpendicular to the ground, L is the unit vector pointing towards to the light , and I_l is the total energy (watts) emitted by the light source.

layer by irradiance energy. The light lobe and shadow are included in the irradiance texture by projecting light intensity distribution on the ground and shadow casting method.

We maintain a full spectral power distribution for all light sources so that the spectral sensitivity of any sensor (including NVG sensors) can be used. Let $I_{e\lambda}$ denote the energy (in watts) emitted by a light at wavelength λ (in nanometers). Then the energy that can be detected by a sensor, I_e, is the integral

$$I_e = \int I_{e\lambda} \times V(\lambda)\, d\lambda, \tag{10}$$

where the integral is taken over all frequencies and $V(\lambda)$ is the sensors luminous efficiency at wavelength λ. In the LP tool, the integral in equation (10) is approximated using a summation for all wavelengths in the range of 300-1000 nm.

To calculate the irradiance, we use the LID to determine the fraction of a light's energy that reaches the surface. Let I_e denote the total energy of the light source in watts that are capable of being sensed by our sensor which is calculated in equation (10). Let L be a unit vector pointing towards the light source, and $\mathrm{LID}(L)$ is the fraction of the lights energy that is emitted in the direction of L, calculated using the method of section 4.2. The vector N is a normal vector perpendicular to the surface being lit, and d is the distance between the light source and the point receiving the light (in meters). Then irradiance E_e (in W/m^2) follows the formula

$$E_e = I_e \times \mathrm{LID}(L) \times \max(0, L^T N) \times d^{-2} \tag{11}$$

where the term $\max(0, L^T N)$ is the Lambert cosine law [19], and d^{-2} is a quadratic falloff in intensity.

5.2 Radiance Map

The window and area light effects are simulated using texture image so that the intensity of the image at each texel is treated as proportional to the radiant exitance. The radiant exitance, M_e, describes the amount of radiation emitted from particular area in W/m^2. We use a GUI (see Figure 20) to allow users to choose a window or radiant area template, place it on a model, and assign a bias and gain to the image so that the texel values (which are between zero and one) can be scaled to match the desired range of window radiant exitance interactively.

There are several approached for achieving the lighting effect from an area light source instead of a point light source; we cite several [20, 21]. One challenge is that the light source can be only partially visible from some points, leading to soft shadows. Another challenge is that the total amount of energy from an area light source would involve a double integral of equation (10) using every point in the radiant area as another light source. Instead, we generate a small fixed number of pseudo-light sources within the radiant area, and we sum the radiant emittance of the area over each point in the pseudo-lights' Voronoi cells in order to determine the pseudo-sources energy (see Figure 22). The Voronoi cell of a psuedo-light point is the set of all points on the radiant area that are closer to that point than any other pseudo-light location. In Figure 22 (a), window area is decomposed as several Voronoi cells (separated by red lines) and yellow dots are the pseudo-light source locations within each Voronoi cell. Figure 22 (b) shows the energy of area light computed by double integral of each point in the window area. In Figure 22 (c), we treat all center points of Voronoi cells as point light sources, calculate their energy by equation (10) and project the sum of all energy coming from the pseudo-light sources in Voronoi cells onto the ground.

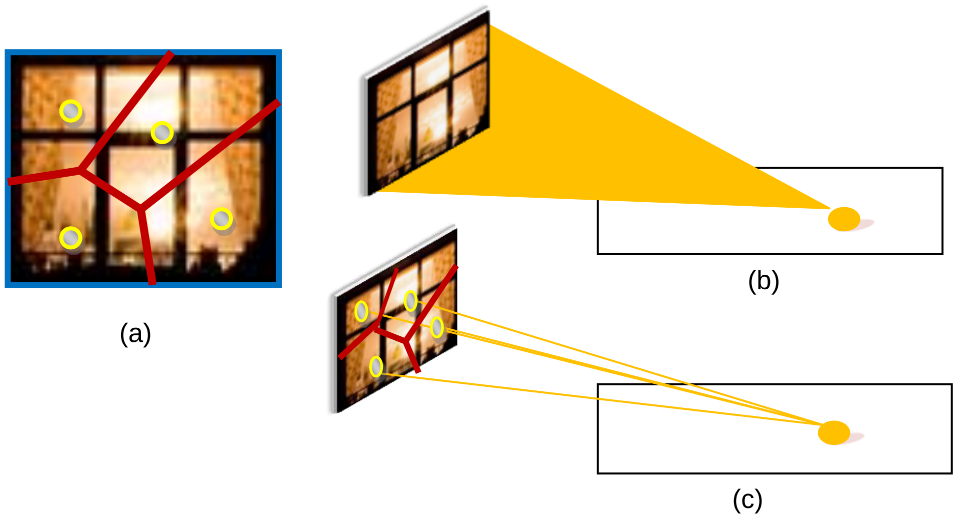


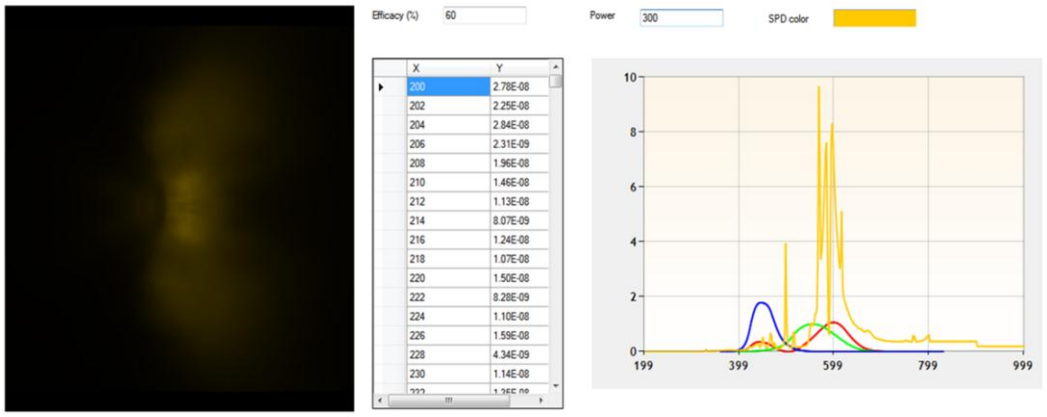
Figure 22. Compute area lights by Voronoi diagram; (a) window area decomposed as Voronoi diagram ; (b) the emitted energy of window light are computed by double integral of every point in the window area; (c) the emitted energy of window light are computed by summing all energy emitted from points in the Voronoi cells.

6 RESULTS AND SIMULATION

This theses presented three different applications, a Light System Database (LDB) editor, a Procedural Lighting (PL) tool, and a building editor. LDB editor is a tool for managing light properties dataset, a PL tool is for predicting light properties and positions, and a building editor is for generating a 3D nighttime building model. In the following subsections, I will give a summary of these tools and present the simulation results of each one.

6.1 LDB Editor

In order to generate a nighttime urban scene, a light properties datasets should be established first and then a PL tool can automatically generate lights positions and decide light properties from the established datasets. In the building editor, street lights could be also assigned by selecting from LDB. However, it will be a challenge for users to manage and establish these data because light datasets include several structures that are difficult for a human to interpret without suitable visualizations; they are intensity distributions of light for lobe pattern, spectral power distribution for color, and structures of light poles in 3D dimensions. Therefore, I have implemented a user interface, that allows people change parameters and understand how it is going to affect the scene. Figure 23 (a) and Figure 23 (b) show two different light sources in the LDB. Figure 23 (a) is a GE car lamp lobe and its spectral power distribution. Figure 23(b) is a GE building mounted lamp lobe combined with its spectral Power distribution color. SPD color can be changed interactively by tuning the efficacy and power of light. Users can see how different lamp lobe, SPD power, and SPD efficacy will affect nighttime scenes and select the desired lamp lobe and color type.



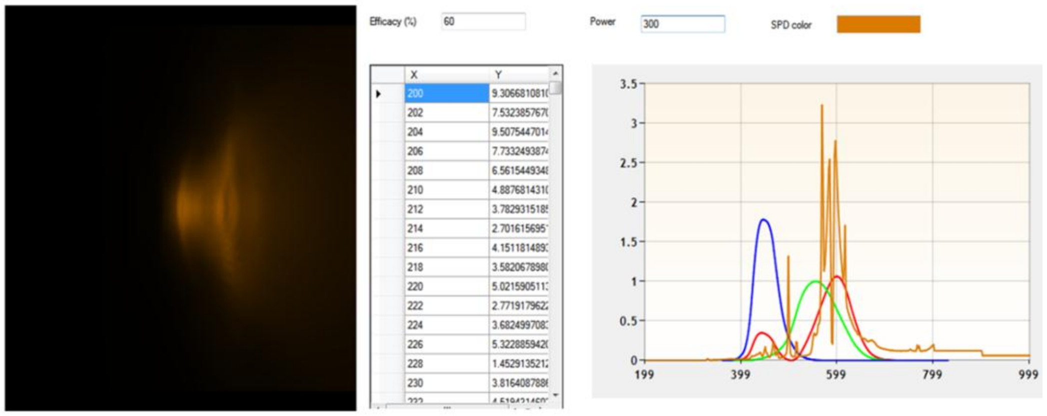
Efficacy (%) 60

Power 300

SPD color 🟨

X	Y
200	2.78E-08
202	2.25E-08
204	2.84E-08
206	2.31E-09
208	1.96E-08
210	1.46E-08
212	1.13E-08
214	8.07E-09
216	1.24E-08
218	1.07E-08
220	1.50E-08
222	8.28E-09
224	1.10E-08
226	1.59E-08
228	4.34E-09
230	1.14E-08

(a)



Efficacy (%) 60

Power 300

SPD color 🟧

X	Y
200	9.306681081(
202	7.532385767(
204	9.507544701
206	7.733249387
208	6.561544934
210	4.887681431(
212	3.782931518
214	2.701615695
216	4.151181489
218	3.582067898(
220	5.021590511
222	2.771917962
224	3.682499708
226	5.322885942(
228	1.452913521
230	3.816408788

(b)

Figure 23. (a) GE car head lamp lobe and SPD color (b) GE building mounted lamp lobe and SPD color. The CIE XYZ luminosity functions are shown superimposed on the light's SPD as blue, green, and red curves for reference.

6.2 Simulation Results of PL Tool

Figure 24 shows the predicted light positions on a portion of Phoenix GIS data by our PL tool. The purple dots and red dots are the street light poles constructed by APS and SRP separately. The green triangles are the light positions generated by our leap tool. It is apparently that the positions of street light constructed by APS and SRP (purple dots and red dots) match our predicted light positions (green triangles). There are a few discrepencies between the generated lights and the SRP lights in the upper-right quadrant of the image that are the result of streets that are not present in the input street vector layer.

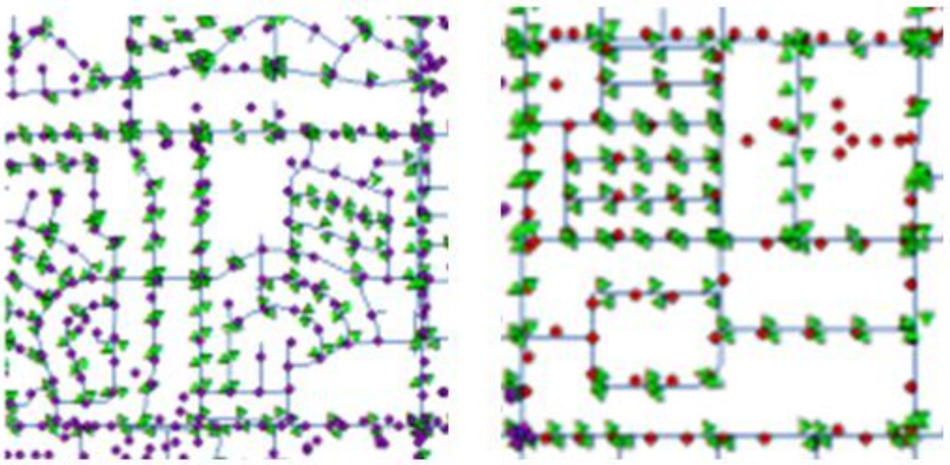
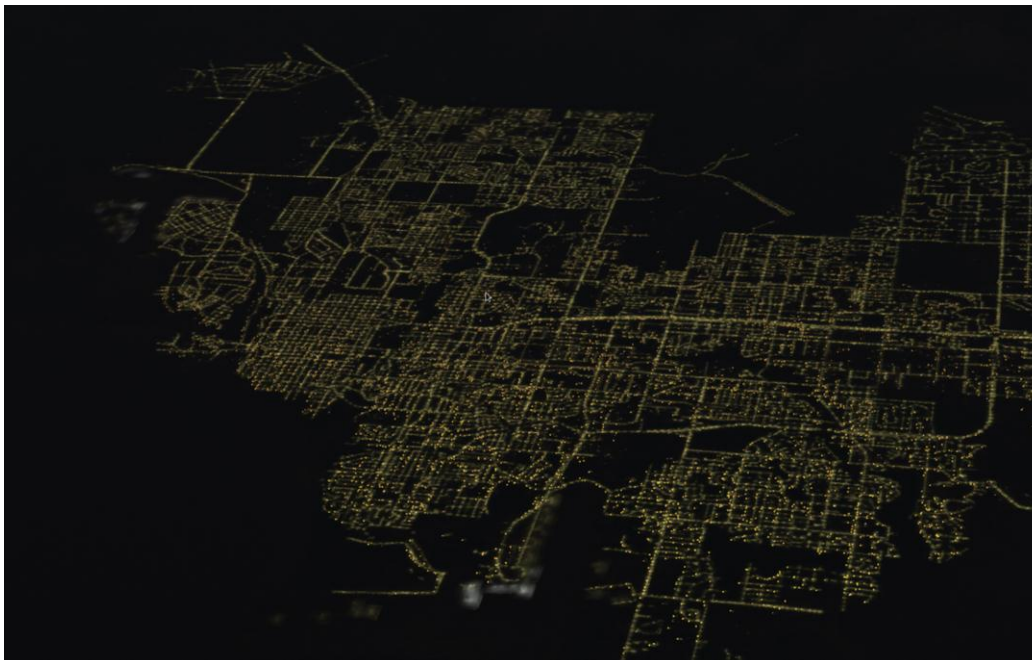
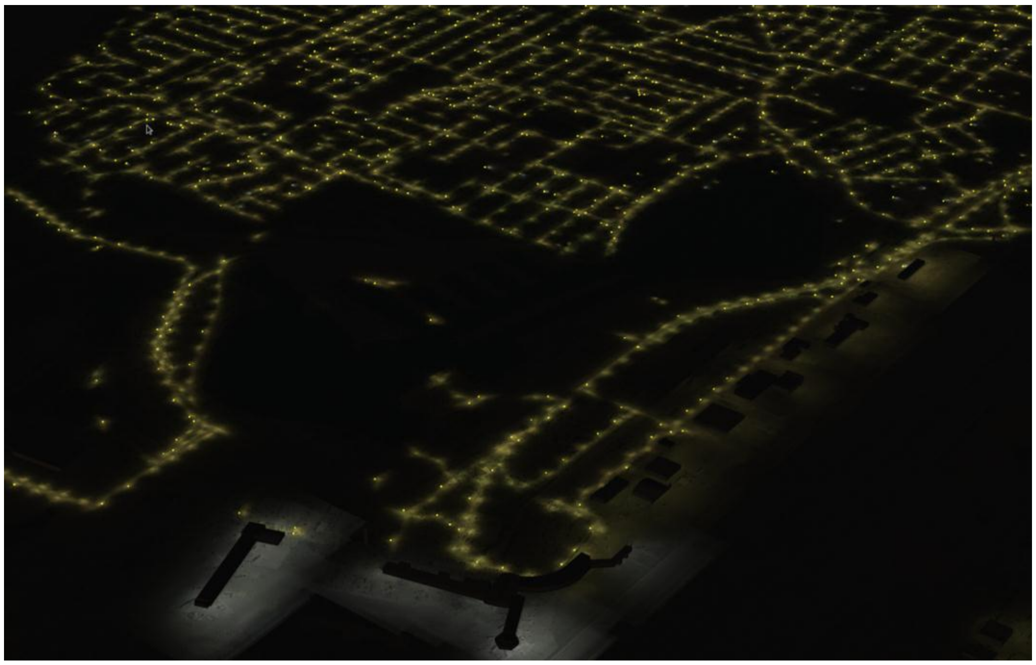


Figure 24. Street light positions predicted by PL tool; (left) procedural results are shown in green and light data from APS is shown in purple; (right) light positions from SRP are shown in red. The density and positions of lights are roughly the same wherever the street layer is accurate.

Figure 25 (a) and Figure 25 (b) show nighttime scenes of Anchorage, Alaska. The positions and properties of street lights in these two figures are generated by our PL tool and the lobes of street lights are rendered by projecting light intensity distribution of each street light. Figure 25 (a) presents a plausible nighttime urban scene which has not been modeled in such a large-scale area. Figure 25 (b) shows a closer view of nighttime scene of Anchorage, AK.



(a)



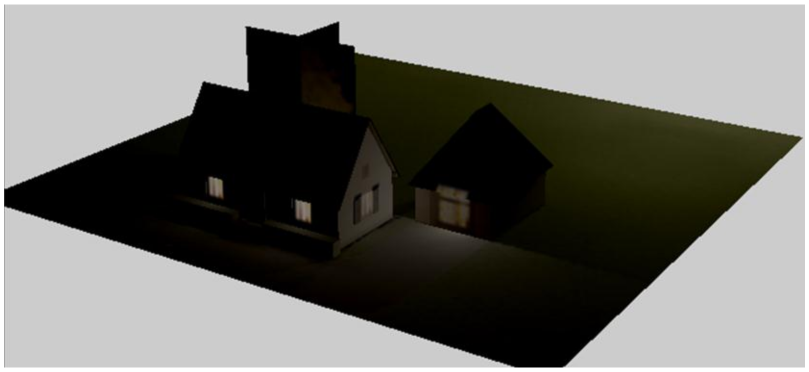
(b)

Figure 25. Nighttime scenery of Anchorage, AK; (a) an overview of nighttime urban scene of Anchorage, AK (b) a closer view of nighttime scenery of Anchorage, AK.

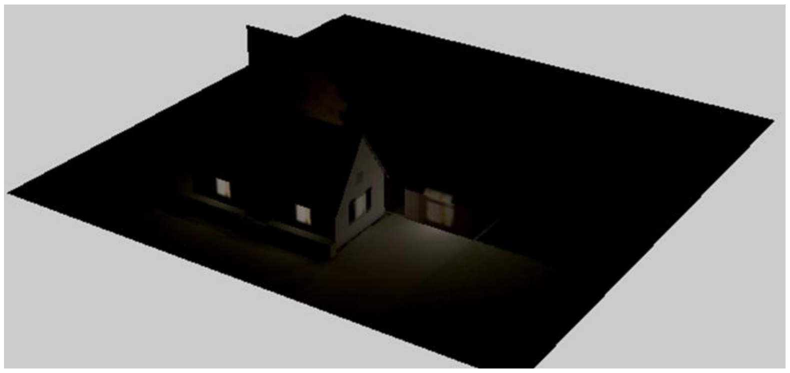
6.3 Simulation Results of Building Editor

Given a small 3D building and LDB, an art designer could create a 3D nighttime scene by building editor. In our building editor, two types of light editors, Light Properties Editor (LP Editor) and Radiance Editor, have been implemented. LP Editor can help users assign properties of street light from LDB and adjust positions and orientations of street light around a 3D building model (Figure 19).The Radiance Editor can select a back ground texture on a window form a 3D model and assign a desired radiance on that back ground window from radiance templates (Figure 20). The shape and energy of widow radiance can be also adjusted by Radiance Editor.

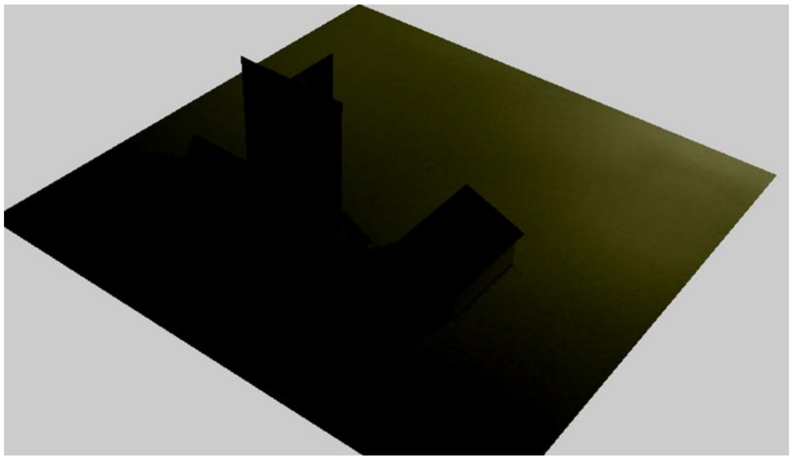
Figure 26 shows the simulation results of radiance map generated by building editor. There are four types of lights in Figure 26, including two types of street lights and two types of window radiances. In Figure 26 (a), it shows the radiance map generated by two types of street lights and two types of window lights. In Figure 26 (b), the radiance map is generated by window lights emitted with energy one watt per square meter. In Figure 26 (c), it shows the radiance map generated by street lights with irradiance and shadow.



(a)



(b)



(c)

Figure 26. The radiance map generated by building editor; (a) radiance map includes street lights and window lights (b) radiance map includes window lights only (c) radiance map includes street lights only

7 CONCLUSION AND FUTURE WORK

In this research, a set of computer graphics techniques for design and simulation of nighttime urban scenes has been presented. We made contributions to the PL tool and building editor which give the simulation games and city designer a unique opportunity to design, preview, and assess nighttime urban scenes with accurate light lobes. In addition, to our knowledge we are the first one to simulate a nighttime urban area in such a large-scale area. We also implement the Lighting System Database (LDB) editor which presents the ability to manage and preview a light datasets, which are hard to understand without a proper user interface. The results of this research clearly demonstrate that use of computer graphics and stochastic method in urban design holds great promise, particularly since these techniques afford the opportunity for simulating a plausible nighttime urban area and consequently provide great contributions in training of night vision goggles for flight purpose or in city planning for public safety and energy efficiency.

Future directions include algorithm enhancement, verification and validation of PL tool, LDB editor and building editor. For the PL tool, the next goal is automatically create parking lot lights and window lights, e.g. by processing textures to look for windows. The algorithms of predicting light sources and rendering light sources can be implemented in higher efficiency. As for the verification and validation, the PL tool can be tested in a larger area and building editor can be tested for processing different parts of models at different resolutions.

ACKNOWLEDGEMENTS

The authors are extremely grateful to the Kris (Ari) Pribadi, Ping Pribadi, Jordan Stoffer, and Jeff Clark for guidance, support, and for testing the project of simulation of nighttime urban scenes. Thanks to Dr. Femiani and Dr. Razdan for their advice in this research.

REFERENCES

[1] J. L. Dyer and K. M. Young, "Night Vision Goggle Research and Training Issues for Ground Forces: A Literature Review," DTIC Document, May 1998.

[2] J. Hu, S. You and U. Neumann, "Approaches to large-scale urban modeling," *Computer Graphics and Applications, IEEE,* pp. 62-69, 2003.

[3] B. Watson, P. Muller, P. Wonka, C. Sexton, O. Veryovka and A. Fuller, "Procedural urban modeling in practice," *Computer Graphics and Applications, IEEE,* pp. 18-26, 2008.

[4] D. Shreiner, OpenGL programming guide: the official guide to learning OpenGL, versions 3.0 and 3.1, Addison-Wesley Professional, 2009.

[5] J. O. Dorsey, "Design and simulation of opera lighting and projection effects," *Computers & graphics,* pp. 41-50, 1991.

[6] N. Fukuda, K. A. Oh and Sasada, "Collaboration Support System for Nightscape Design Based on VR Technology," 2000.

[7] H. Gouraud, "Continuous shading of curved surfaces," *IEEE Transactions on Computers,* vol. 100, pp. 623-628, 1971.

[8] A. Watt and F. Policarpo, 3D Games: Real-time Rendering and Software Technology, Addison-Wesley, 2001.

[9] M. Abrash, Graphics Programming Black Book, The Coriolis Group, 1997.

[10] C. Elvidge, K. Baugh, E. Kihn, H. Kroehl and E. Davis, "Mapping city lights with nighttime data from the DMSP Operational Linescan System," *Photogrammetric Engineering and Remote Sensing,* pp. 727-734, 1997.

[11] H. Wann Jensen, F. Durand, J. Dorsey, M. M. Stark, P. Shirley and S. Premože, "A physically-based night sky model," 2001.

[12] D. Chickering, "Learning Bayesian networks is NP-complete," *Learning from data: Artificial intelligence and statistics v,* vol. 112, pp. 121-130, 1996.

[13] S. Russell, P. Norvig, J. Canny, J. Malik and D. Edwards, Artificial Intelligence: a modern approach, 1995.

[14] D. Heckerman, "A Tutorial on Learning with Bayesian Networks," *NATO ASI SERIES D BEHAVIOURAL AND SOCIAL SCIENC,* pp. 301-354, 1998.

[15] M. Cayless and A. Marsden, Lamps and lighting: a manual of lamps and lighting, Hodder Arnold, 1983.

[16] M. S. Rea, The IESNA lighting handbook : reference & application, New York, NY: Illuminating Engineering Society of North America, 2000.

[17] T. Mitchell, Web Mapping Illustrated: Using Open Source GIS Toolkits, O'Reilly Media, 2005.

[18] I. Ashdown, Radiosity: a programmer's perspective, New York, NY: John Wiley & Sons, Inc., 1994.

[19] F. Luna, Introduction to 3D game programming with DirectX 10, Jones & Bartlett Learning, 2006.

[20] C. Goral, K. Torrance, D. Greenberg and B. Battaile, "Modeling the interaction of light between diffuse surfaces," in *SIGGRAPH '84*, New York, NY, 1984.

[21] S. J. Teller, "Computing the antipenumbra of an area light source," in *SIGGRAPH '92*, New York, NY, 1992.