Numerical Computing 2014 Project Proposal

Max Dumas

May 13, 2014

Abstract

For my project I intend to develop a two-dimensional radiosity-based illumination algorithm. The goal of this project is to be able to take a generalized two-dimensional scene, with any sort of polygonal geometry contained within, and create a fully illuminated scene in a timely fashion, with reasonably accurate simulated diffusion of light throughout the open space of the scene. The final scene will, as a result, feature realistically gradated lighting, blocked and reflected by obstacles, resulting in soft, believable shadows. A hopeful stretch goal of this project is to have a solution that works in real-time—that is, the solution will be robust enough and fast enough to be capable of recalculating lighting on-the-fly with changing geometry and light-source properties.

1 Implementation

Implementation of the project will involve the following:

1. Creating structures to represent the geometry of the level in such a way that radiosity can be computed efficiently.
2. Computing the radiosity for the geometry of the scene and illuminating the space of the scene.
3. Displaying the illuminated scene.

2 Scene Representation

The scene, which will be viewed from directly above, will be primarily represented by two data structures: The space and boundaries of the scene, which will have set, square dimensions, will be discretized into an m by m grid of spaces referred to as L, which will store information regarding their position, illumination as a scalar factor between zero and one, and their color. All grid spaces will be square and have a uniform side-length. A standard undirected graph, $G = (V, E)$, will represent the geometry contained within the scene, where V is the set of vertices and E is the set of edges which compose the graph. E will have magnitude n. Edges are right-facing, meaning

that for an edge (v, w), the normal of the edge points to the right when facing towards vertex w from vertex v.

E will also have associated vectors of magnitude n for the following: $\boldsymbol{\rho}$ represents the reflectivity of the edges, $\boldsymbol{\epsilon}$ represents the emissivity of the edges, and finally represents the radiosity of the edges. The vectors are structured so that any edge e_i in E has reflectivity ρ_i, emissivity ϵ_i, and radiosity β_i.

G, $\boldsymbol{\rho}$ and $\boldsymbol{\epsilon}$ are predefined. $\boldsymbol{\beta}$ is our unknown. The bounding dimension of L, in terms of grid-spaces, and the width of each grid-space are both also predefined and L is generated at runtime, with each grid-space being assigned a color, a position based on its index in L, and its illumination set to zero.

Finally, an n by n matrix $\boldsymbol{F}$ will represent the view-factor between any given pair of edges, with $0 \leq F_{ij} \leq 1$ being a scalar representing how visible the face of the edge i (as indicated by the direction of its normal) is to the face of an edge j. This factor will be calculated using the following equation:

$$F_{ij} = \frac{K}{r^2} \cos(\theta_i) \cos(\theta_j) Vis(i, j) \tag{1}$$

θ_i and θ_j represent the angles between the line segment drawn from the centers of the two edges and the normals of edge i and j, respectively. r is the distance between the centers of the edges, and K is an arbitrary constant. $Vis(i, j)$ represents the approximate percentage of j that is accessible by i, meaning that it is not blocked by any other geometry. If i and j have no obstacles in between them, $Vis(i, j)$ returns 1. If j is fully occluded by other geometry from i, $Vis(i, j)$ returns 0. For partial occlusion, $0 < Vis(i, j) < 1$.

3 Computing Radiosity and Illuminating the Scene

The following equation, given our previously defined data structures, represents our primary radiosity equation:

$$(\boldsymbol{I} - \boldsymbol{\rho F})\boldsymbol{\beta} = \boldsymbol{\epsilon} \tag{2}$$

Where $\boldsymbol{\rho F}$ is a row-wise multiplication, so each ρ_i is multiplied to the row F_i. We can solve for this equation iteratively using the Jacobi or Gauss-Seidel methods, giving us our radiosity with an accuracy of as many iterations as we desire. These methods are guaranteed to converge (and within fairly few iterations) as $(\boldsymbol{I} - \boldsymbol{\rho F})$ is diagonally dominant.

After solving for $\boldsymbol{\beta}$, we have radiosity for every edge. Thus, every edge of our geometry has, to some degree, become a light source. However the surfaces of the edges are not visible due to our two-dimensional perspective, so it becomes necessary to propagate the light energy that is now stored in the edges of the geometry to the grid spaces which comprise the scene. The way I am considering doing this is by using a ray-sweep algorithtm to determine the visibility of each grid space to each edge, creating a new matrix $\boldsymbol{H}$ that will be m by n, and contain a view factor for each edge to the set of all grid spaces. Light is then propagated outwards from the edges onto the grid spaces by a simple inverse function. For an edge E_i of E, the light given to any

given grid space L_j in L is represented by the following equation:

$$L_j = \frac{H_{ij}}{r^2}\beta_i \tag{3}$$

4 Displaying the Illuminated Scene

For display, I intend to use a simple low-level graphics library such as SDL for C, or a similar library in another language. Grid spaces will displayed as untextured, vertex-colored quads, and edges will simply be displayed as lines. All visible illumination effects will be achieved via the coloring of the grid spaces.