McNair Scholars Journal

Volume 7 | Issue 1 Article 10

2003

A Heuristic Algorithm: Simulating Light Propagation in Orthogonal Polygons

Omar Hwail
Grand Valley State University

Follow this and additional works at: http://scholarworks.gvsu.edu/mcnair

Recommended Citation

Hwail, Omar (2003) "A Heuristic Algorithm: Simulating Light Propagation in Orthogonal Polygons," *McNair Scholars Journal*: Vol. 7: Iss. 1, Article 10.
Available at: http://scholarworks.gvsu.edu/mcnair/vol7/iss1/10

Copyright © 2003 by the authors. McNair Scholars Journal is reproduced electronically by ScholarWorks@GVSU. http://scholarworks.gvsu.edu/mcnair?utm_source=scholarworks.gvsu.edu%2Fmcnair%2Fvol7%2Fiss1%2F10&utm_medium=PDF&utm_campaign=PDFCoverPages

A Heuristic Algorithm: Simulating Light Propagation in Orthogonal Polygons

Omar Hwail
McNair Scholar

Christian Trefftz, Ph.D.
Faculty Mentor

ABSTRACT

The purpose of this study is to determine the area of light emitted by a source in an orthogonal polygon on a two-dimensional lattice using the cellular automata construction method. By applying this method, an efficient algorithm was tested and developed to determine the area of light propagated. The algorithm, although not optimal, gives a close approximation of the number of cells on the lattice that are to be illuminated. Furthermore, the algorithm acknowledged in this research is sufficient to work with any orthogonal polygon. This research is based on a classical computational geometry problem – the art gallery problem. It is hoped that the results of this research can contribute to finding more efficient solutions to the problem as well as other computational geometry problems.

Introduction

In 1973 during a discussion with other mathematicians, Victor Klee introduced the art gallery problem: How many guards are sufficient to guard any polygon with *n* vertices? The problem was called the art gallery problem or the illumination problem because it resembled a security configuration in an art gallery as well as represented the illumination of an art gallery. For example, if an owner of an art gallery wants to place cameras (source of light) such that the whole gallery will be thief proof, before that owner can configure his/her security setup, he or she will first have to answer a few questions. Questions like “What is the minimum number of cameras required in order to protect the expensive art collection?” and “Where will the cameras be placed so that the whole gallery is guarded?”

There are many forms of the art gallery problem, dealing with many types of polygons. In this research we looked only at using orthogonal polygons to represent a gallery. Orthogonal polygons are polygons that have a set of mutually perpendicular axis, meeting at right angles (see fig. 1). An orthogonal polygon can also be dissected at its vertexes, resulting in squares or rectangles.

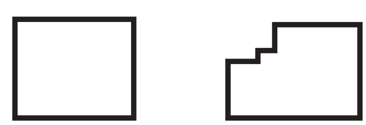

Fig. 1. *Set of orthogonal polygons*

Related Work

The art gallery problem is now a classical problem in the study of algorithms. Although it is extended into many forms, most forms deal with the classical idea of "line-of-light" illumination model; one point can illuminate another point, as long as the line segment between the two points is not intersected by any object. In our problem, we restrict the illumination problem to an orthogonal polygon.

One of the optimal solutions created to solve this problem in orthogonal polygons is triangulation. The theoretical basis behind this is simple. If a single light source is enough to light the simplest polygon, a square, then it is evident that a single light source should be enough to illuminate any convex polygon[1]. The problem becomes interesting when the convex polygons take complex shapes. Triangulation deals with dividing the polygon into non-overlapping triangles and placing a source in each of these triangles. This is done because it is known that one source is sufficient to illuminate a simple convex polygon; therefore one source should be sufficient to illuminate a triangle, which is also classified as a simple convex polygon. Because it is also proved that any simple polygon can be triangulated, then any simple polygon triangulated into *n* triangles will need *n* light sources to sufficiently illuminate the entire polygon. However, it is obvious that one light source can sometimes illuminate more than one triangle. In fact, a single source can illuminate up to three triangles: the triangle itself and its two neighboring triangles. Thus, in the worst-case scenario, a polygon would need *n* light sources and in the best-case scenario it would need *n/3* light sources.

Theoretical Basis

Before we begin to compute the minimum number as well as the final position of the sources that will be needed to maximize the illumination throughout a polygon, we must first compute the area of propagation to be illuminated by one source, regardless of the position of that source. Like many other computational geometry problems, a strategy for solving this is to decompose the problem into small sets and begin computation on each set by using a specialized algorithm. Finally, we combine the partial solutions from all sets to form a complete solution to the problem.

We looked at a method that is well known for decomposing complex problems, such as the current problem at hand, into small workable sets. The method we looked at was cellular automata construction. A cellular automata is represented by a spatial lattice consisting of an infinite number of cells aligned in rows and columns. The state of all cells is updated simultaneously according to a local rule and, thus, the state of the entire lattice advances in discrete time steps to form a final state for the whole lattice. For example, the states of some cells neighboring a certain cell are randomly changing; when all the cells neighboring this particular cell take their final state, this is called the quiescent state. Thereafter, this particular cell will change into its final state by evaluating the final state of its neighbors and applying it to itself.

Using the cellular automata construction, we were able to represent an orthogonal polygon. We implemented a 100x100 lattice that consisted of 10,000 independent cells, each of which can have one of five states: *source, open, wall, lit,* or *unlit*. Thus, the polygon, which is represented by the 10,000 individual cells, can be easily divided into small workable portions.

Decomposing a polygon into individual cells makes it easier to dissect a polygon along specific cells. In our analysis, we dissected a polygon along particular cells, which represent positions with critical angles in respect to the cell that represents the position of a source. These critical angles are 0, 45, 90, 135, 180, 225, 270, and 315 degrees. As shown in fig. 2, a polygon is dissected into eight regions along the critical angles in respect to the source location.

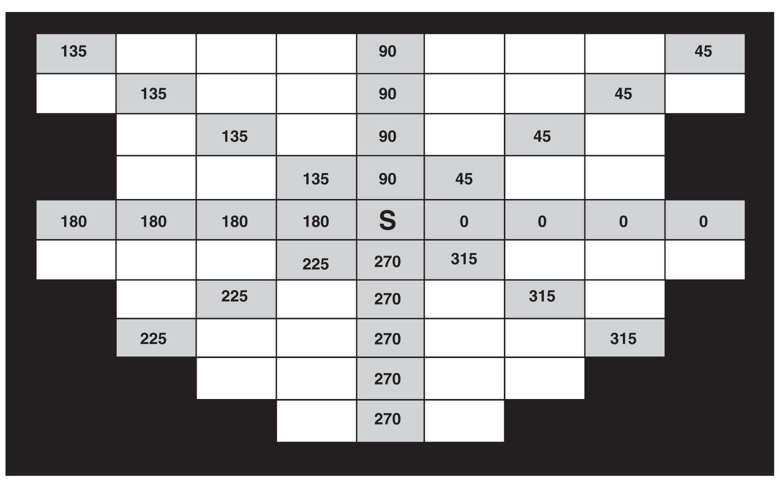

Fig. 2. *A polygon dissected into eight regions*

[1] convex polygon – A polygon such that no side extended cuts through any other side or vertex; it can be cut by a line in at most two points.

Algorithm Analysis

The algorithm we implemented comprised two phases: the orientation phase and the propagation phase. The orientation phase consisted of two steps as well (see fig. 3).The first step in the orientation phase is to determine if a cell lies on a critical angle in respect to the source. This is done by traversing each cell then evaluating the *x*- and *y*-axis positions of the current cell along with the *x*- and *y*-axis positions of the source cell. It was shown that perpendicular angles use a certain type of formula, whereas diagonal angles use another type of formula. These formulas are shown in fig. 4.

The second step of the orientation phase is to assign a region location to every cell. We begin this process by traversing each cell, inspecting neighboring cells, and determining whether one of the adjacent cells has a base angle or if it is a cell that has already been assigned a region. If one of the neighboring cells to a particular cell has a base angle, then using the location of that neighboring cell, we can determine the appropriate region of that particular cell. If, however, one of the neighboring cells has already been assigned a region, then using the location of that neighboring cell in respect to other neighboring cells, we can evaluate the region location of that particular cell.

The second phase of the algorithm is the propagation phase in which we determine if a cell is to be illuminated by a source (see fig. 5). We undergo this phase by traversing individually each cell and, depending on the region of that particular cell, inspect the two critical adjacent cells (see fig. 6). The evaluation of the state of the adjacent cells determines whether the particular cell should change its state to either UNLIT or LIT. For example, if the particular cell we are on during our traversing lies in region 4, then we would inspect the south-east and east adjacent cells. If both of the adjacent cells have the state WALL, then the particular cell will change its state from being OPEN to being UNLIT. If both adjacent cells are illuminated, then the current cell will change to be LIT. If, however, one of the adjacent cells is a WALL and the other LIT, then we would inspect the slope of the particular cell in respect to the source along the slope of the wall and thus determine whether the particular cell should be lit. Another example can be seen in fig. 7, where a DFA[2] simulates the computation of region 7 by determining whether a cell should be LIT or not. Note that the result of the slope from the particular cell in comparison with the slope from the NW cell concludes whether states M_6, M_7, and M_9 are to be accepted, therefore resulting in the particular cell being changed to LIT.

```
ORIENTATION PHASE

Traverse all cells
        If (current cell applies to any base cell
              Set current cell as a base cell

Until Done
Traverse all cells
{
        Inspect neighboring cells
        Determine current cell's region
        Set current cell's slope in respect to the
}
```

Fig. 3. *Algorithm pseudo code: Orientation phase*

```
S=Source Cell
C=Current Cell

0°   – [ (Sy – Cy==0) && (Cx > Sx) ]
90°  – [ (Sx – Cx==0) && (Sy > Cy) ]
180° – [ (Sy – Cy==0) && (Sx > Cx) ]
270° – [ (Sx – Cx==0) && (Cy > Sy) ]
45°  – [ (Cx – Sx)==(Sy– Cy) ]
135° – [ (Sx – Cx)==(Sy– Cy) ]
225° – [ (Sx – Cx)==(Cy– Sy) ]
315° – [ (Cx – Sx)==(Cy– Sy) ]
```

Fig. 4. *Formulas for determining cells with critical angles*

[2] DFA - Deterministic Finite-state Automaton is a model of computation which consists of a set of states, a start state, an input alphabet, a set of accepted states, and a transition.

```
PROPAGATION PHASE

Until Done
Traverse all cells
If (current cell is OPEN)
{
        Switch Case (Region of current cell)
        {
                If (Cell is a base cell)
                {
                        Inspect preceding adjacent base cell
                        If (Adjacent cell is LIT or SOURCE)
                                Set cell state as LIT
                        Else if (Adjacent cell is UNLIT)
                                Set cell state as UNLIT
                }
                If (Appropriate adjacent cells are LIT)
                        Set current cell's state as LIT
                Else if (Adjacent cells are not OPEN)
                {
                        Evaluate the state of adjacent cells
                        If (Current cell's slope passes condition)
                                Set current cell's state as LIT
                        Else
                                Set current cell's state as UNLIT
                }
        }
}
```

Fig. 5. *Algorith pseudo code – Propagation phase*

Cells in region 1
    W and SW cell will be inspected
Cells in region 2
    SW and S cell will be inspected
Cells in region 3
    S and SE cell will be inspected
Cells in region 4
    SE and E cell will be inspected
And so forth…

Fig. 6. *Critical adjacent cells of some regions*

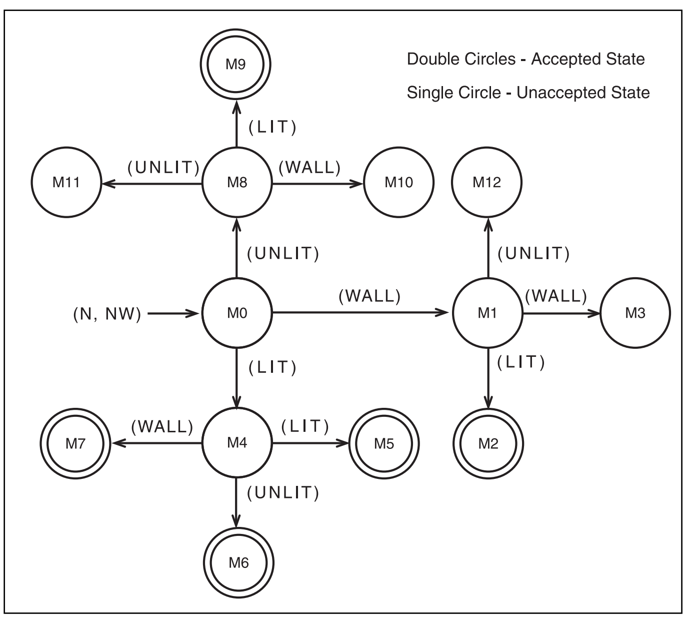

Fig. 7. *DFA² of region 7*

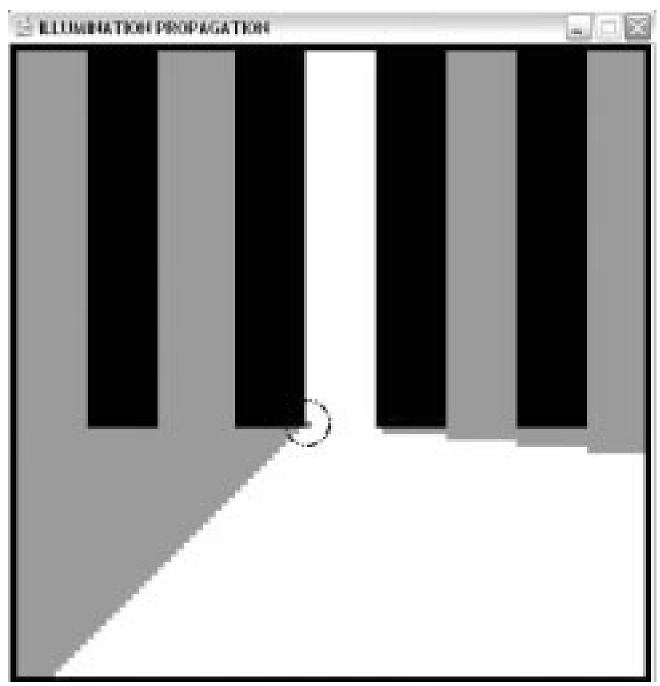

Fig. 8. *An orthogonal polygon with one source*

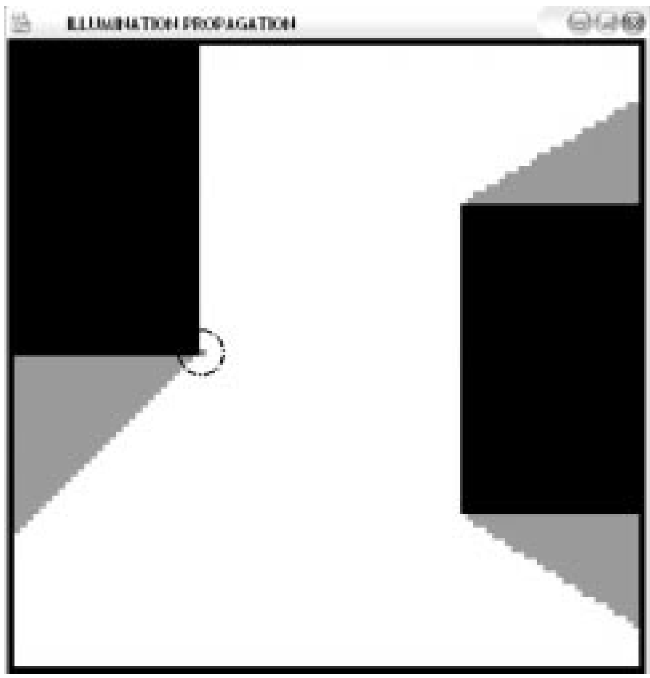

Fig. 9. *Another orthogonal polygon with one source*

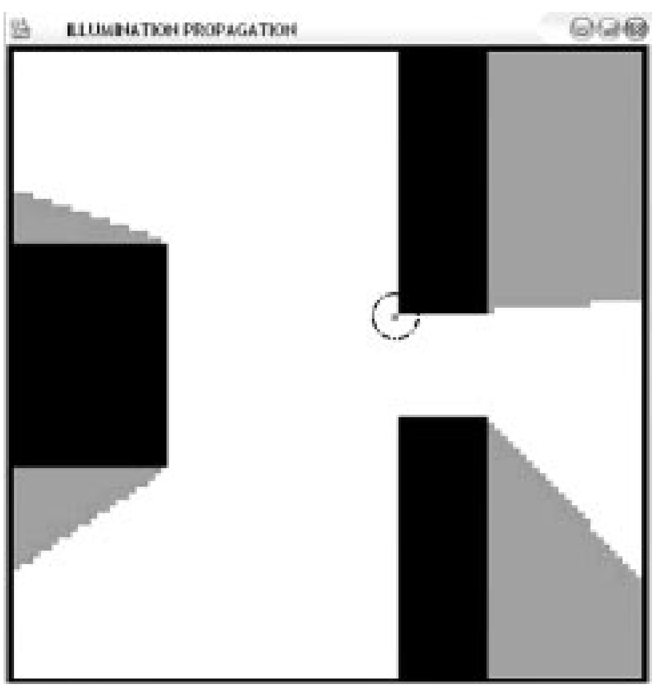

Fig. 10. *Another orthogonal polygon with one source*

Algorithm Complexity

The complexity of our algorithm has been computed to be $O(N^2)$. Although the time complexity seems sufficient to cover small-scale models, it is not an optimal solution for large scale models, which could take a considered amount of time to compute. It is also important to note that the error of approximation, when computing angles between a source and any cell, can be reduced by increasing the number of cells. For example, the fine line error of approximation between illumination and shadows can be reduced by 50% in 100x100 lattice with 10,000 cells, as opposed to a 50x50 lattice with 2,500 cells.

Results

Figures 8, 9, and 10 illustrate some random orthogonal polygons that utilized the fore mentioned algorithm in an application we implemented.

Conclusion and Future Work

The art gallery problem has been approached and studied from different perspectives. Even though the classic idea of the problem is to find the minimum number of guards to guard a polygon with *n* vertices, it gives inspiration to solve many practical problems. Different to many other approaches, the cellular automata method proved to efficiently simulate the use of one guard in the problem. We represented this by illustrating the propagation of light by one source in random polygons. Subsequent to being able to compute the propagation of one source using the algorithm we have presented, we hope to implement this phase into the main problem. Then, we will be able to compute the minimum number of sources needed and the position of those sources in order to maximize the illumination throughout a polygon. Another aspect we would like to investigate in the future is using non-orthogonal polygons, such as complex convex polygons.

Acknowledgements

The authors would like to thank the Grand Valley State University Ronald E. McNair Scholars Program for giving us the opportunity to conduct this research. We would like to also thank Dr. Hans Dulimarta for his helpful discussions.

Works Cited

Aggarwal, Alok. "*The Art Gallery Theorem: Its Variations, Applications and Algorithmic Aspects.*" Diss. The John Hopkins University, 1984.

Aronovl, Boris, Alan R. Davis, Tamal K. Dey, and P. Suclebkunar Pals. "Visibility with Reflection." *Proceedings of the Eleventh Annual Symposium on Computational Geometry*. ACM Press, 316-326, 1995.

Banos, Gonzalez Hector, and Jean-Claude Latombe. "A Randomized Art-Gallery Algorithm for Sensor Placement", *ACM Computing Surveys*, 232-240, June 2001.

Keil, J. Mark. "*Minimally Covering a Horizontal Convex.*" *Proceedings of the Second Annual Symposium on Computational Geometry*. ACM Press, 43-51, August 1986.

M.de Berg, M. van Kreveld, M. Overmars, and M. Schwarzkopf. "*Guarding and Triangulations, Computational Geometry Algorithms and Applications.*" Springer-Verlag, 46-49, 1997.

Sarkar, Palash. "A Brief History of Cellular Automata", *ACM Computing Surveys*, 32:1, 80-107, March 2000.

Trefftz, Christian, and Helmuth Trefftz. "A Heuristic for the Orthogonal Polygon Art Gallery Problem." *Applied Informatics*, 850-855, Feb 2000.