Elimination of intermountain lakes on fractal landscapes by erosion

Douglas F. Marsh

The University of Montana

Follow this and additional works at: https://scholarworks.umt.edu/etd
Let us know how access to this document benefits you.

Recommended Citation
https://scholarworks.umt.edu/etd/8072

This Thesis is brought to you for free and open access by the Graduate School at ScholarWorks at University of Montana. It has been accepted for inclusion in Graduate Student Theses, Dissertations, & Professional Papers by an authorized administrator of ScholarWorks at University of Montana. For more information, please contact scholarworks@mso.umt.edu.
COPYRIGHT ACT OF 1976

This is an unpublished manuscript in which copyright subsists. Any further reprinting of its contents must be approved by the author.

MANSFIELD LIBRARY
UNIVERSITY OF MONTANA
DATE: 1986
ELIMINATION OF INTERMOUNTAIN LAKES
ON FRACTAL LANDSCAPES
BY EROSION

By
Douglas F. Marsh
B. A., University of Montana, 1983

Presented in partial fulfillment of the requirements
for the degree of
Master of Science
University of Montana
1986

Approved by
Chairman, Board of Examiners

Dean, Graduate School

Sept 9, 1986
Fractal geometry is a field of mathematics developed by Benoît Mandelbrot. The irregular objects generated by using fractal based techniques are defined as fractals and have been found to be excellent models of natural objects. An algorithm developed by Loren Carpenter creates realistic images of natural landscapes that have fractal characteristics. However, these landscapes have the unnatural characteristics of too many intermountain lakes and a lack of a realistic drainage pattern. This thesis contains an algorithm that eliminates the intermountain lakes of a fractal landscape and establishes a dendritic drainage pattern on the surface.

The algorithm is modeled after the geologic process of water erosion. The process is taken in its simplest form by eroding the outlet of each lake and the outlet's drainage. This process is continued until all lakes are eliminated.

The result is an algorithm which, when applied to the landscape produced by Carpenter's algorithm, produces an eroded landscape with a natural drainage network. The algorithm can be customized to control the erosion or can be used as a basis for geologic modeling and other applications.
# Table of Contents

Abstract

Table of Contents

List of Figures

Acknowledgements

1. Project Formulation
   1.1. Background
   1.2. Introduction to Problem
   1.3. Statement of Problem
   1.4. Proposed Work Plan
   1.5. Suggested Reading Approach

2. Computation and Algorithm Overview
   2.1. Development Approach
   2.2. Algorithm Background and Existing Data Structures
   2.3. Abstract Data Structure and Operations
   2.4. Restricted Data Structures and Operations

3. Algorithm and Description
   3.1. High Level
      3.1.1. High Level Background
      3.1.2. High Level Informal Description
      3.1.3. High Level Formal Description
   3.2. Description of Step A
      3.2.1. Informal Description of Step A
      3.2.2. Formal Description of Step A
   3.3. Description of Step B
      3.3.1. Informal Description of Step B
      3.3.2. Formal Description of Step B
   3.4. Description of Step C
      3.4.1. Informal Description of Step C
      3.4.2. Formal Description of Step C
   3.5. Description of Step C.1.6
      3.5.1. Informal Description of Step C.1.6
      3.5.2. Formal Description of Step C.1.2
   3.6. Description of Step C.1.7
      3.6.1. Informal Description of C.1.7
      3.6.2. Formal Description of C.1.7
   3.7. Algorithm Example

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>5</td>
</tr>
<tr>
<td>2-1</td>
<td>12</td>
</tr>
<tr>
<td>3-1</td>
<td>25</td>
</tr>
<tr>
<td>3-2</td>
<td>28</td>
</tr>
<tr>
<td>3-3</td>
<td>33</td>
</tr>
<tr>
<td>3-4</td>
<td>37</td>
</tr>
<tr>
<td>3-5</td>
<td>39</td>
</tr>
<tr>
<td>3-6</td>
<td>45</td>
</tr>
<tr>
<td>3-7</td>
<td>52</td>
</tr>
<tr>
<td>3-8</td>
<td>57</td>
</tr>
<tr>
<td>3-9</td>
<td>58</td>
</tr>
<tr>
<td>3-10</td>
<td>61</td>
</tr>
<tr>
<td>4-1</td>
<td>64</td>
</tr>
<tr>
<td>4-2</td>
<td>65</td>
</tr>
<tr>
<td>4-3</td>
<td>67</td>
</tr>
<tr>
<td>4-4</td>
<td>69</td>
</tr>
<tr>
<td>4-5</td>
<td>70</td>
</tr>
<tr>
<td>4-6</td>
<td>71</td>
</tr>
<tr>
<td>4-7</td>
<td>71</td>
</tr>
</tbody>
</table>
Acknowledgements

I would like to thank my mother and father for their continued emotional and financial support. Without this support I could not have accomplished the level of education that I have. They have given me the opportunity to discover how enjoyable, exciting and rewarding education can be.

I would particularly like to thank Dr. Alden Wright for his efforts and support as both a professor and a personal friend. I can not thank him enough for his dedication to the computer science department and its student body. It is this dedication that has created a pleasant learning environment for everybody.

Special thanks to Charles Zacky. Without his help I would have had a hard time completing this thesis. It was those late night, thought-provoking conversations that were most educational, as well as beneficial.
Chapter 1

Project Formulation

1.1. Background

In the past ten years the field of fractal geometry has become an increasingly important tool for the representation of natural phenomena. Computer graphics has become the primary media for display of fractal based images, because of the need to perform numerous, repetitive calculations to display the calculated information.

Prior to the use of fractal geometry there had been limited success by using standard euclidean geometry for creating realistic looking displays of objects found in nature. By using fractal based techniques natural objects, landscapes and motion can be simulated [Mandelbrot, 1978].

Fractals are defined as objects that obtain a fractional dimension. That is, they are objects which do not have standard euclidean dimensions such as one, two, three, etc.; rather they have dimensions between one and two or between two or three, etc. This is the sole definition of a fractal as defined by Benoit Mandelbrot [Mandelbrot, 1978/p. 14]. However, all fractals do have a common characteristic. Every fractal that has ever been of use has also had self-similarity in one way or another – the small parts look like the big parts. Random fractals are statistically self-similar, so even though they do not repeat their pattern...
exactly, they clearly have the same look to them, no matter at which level of detail they are observed [Sorenson, 1984/p. 157].

Of particular interest are fractal generated landscapes. These landscapes are not models of existing landscapes; rather they are randomly generated, fictitious landscapes. The primary use today of fractal landscapes is in the film industry to generate background images for motion pictures. Another common use has been in flight simulators for testing pilots over various terrain. Fractals are useful in the film industry and in flight simulators because of the self-similar characteristic. This creates the ability to zoom into a landscape with increasing detail without sacrificing resolution.

1.2. Introduction to Problem

Techniques created to build random landscapes generate very realistic looking computer graphic images. Although these landscapes can look very realistic, there exists a geologically uncommon characteristic. All of the images observed have an abundance of intermountain depressions that would naturally fill with water runoff, creating many intermountain lakes and ponds. Geologically this would be uncommon.

Geologic landscapes do exist which have an abundance of intermountain lakes or ponds, such as karst topography or intermountain glaciated topography. However, karst topography is uncommon and has a drainage pattern that is primarily beneath the surface, and intermountain glaciated topography has localized or clustered lakes in specific regions. The images created by fractal
techniques show an even distribution of lakes and ponds, which is not representative of either karst or intermountain glaciated topography.

These depressions would naturally become ponds or lakes which would overflow their banks and drain into other depressions. As a result the landscape would show evidence of erosion by water. If the above course of events were to take place, erosion of the outlet rim would occur. This would continue until the depression is eliminated. This erosional process would eventually produce a more complete drainage pattern with no lakes, swamps, or depressions.

All of the images that I have observed which used a fractal based methodology for generation of mountain landscapes show no evidence of erosion produced drainage patterns. Drainage on such a landscape would be short, terminating in one of the many depressions that exist. These images usually depict a lunar like landscape where topography has had no water erosion.

The process of water erosion applied to a fractal landscape surface would create a realistic dendritic drainage pattern. The removal of the lunar appearance by the addition of a dendritic drainage pattern may begin to create images similar to earth landscapes where water erosion occurs and is a factor in shaping the topography. This creates a potential for additional applications.

1.3. Statement of Problem

The problem of this thesis is to develop a concise algorithm to eliminate the existence of intermountain lakes or depressions on fractal generated landscapes of any degree or complexity. The desired result is a landscape that demonstrates a
further degree of realism from the initial landscape. The resulting landscape should allow for magnification of the surface in either a positive or negative direction demonstrating a dendritic drainage pattern at any level of detail.

If possible, the algorithm should be flexible enough to allow for modification, if partial elimination of lakes, elevation control of lake positions or control of lake size is desired for additional applications.

There are a couple of fractal landscape algorithms that are all based on the same method: they vary only in the geometric shape that is used. Of the two geometric shapes, a triangle or square, the triangle appears to more commonly used. The triangle develops a more realistic landscape, because it generalizes to a mountain's shape more closely [Cunningham, 1986].

The triangle as the basic unit for the fractal landscapes will be the method used for this project. The algorithm used to develop the initial fractal landscape was developed by Loren Carpenter and is outlined as follows [Smith, 1984/p. 4]:

First a sequence of subdivisions of a triangle is developed. Starting with a single triangle find the midpoint of each side. Create level one by connecting the three new points to one another to form four new triangles. The above steps are repeated for each of the new triangles to create level two of sixteen new triangles, and so on to the desired level (See Figure 1-1).

Now, elevations are assigned to the vertices of these subdivisions to generate the landscape. The three vertices of the original triangle must be assigned a value that represents each vertex's elevation. The elevation of a vertex is the vertex's vertical displacement from a flat lying triangle. Each new vertex of
Figure 1-1:

level one is now assigned a value. The value is determined by generating a
random number that is proportional to the length of the corresponding side. The
values for the vertices of level two are then determined in a similar manner. This
process is continued until all the vertices are assigned a value up to the desired
level. Carpenter's algorithm is discussed in greater detail in section 2.2.

Although the proposed algorithm to eliminate lakes on a fractal surface is
based on a landscape that uses a triangle as the basic unit, it is believed that with
minor modifications the technique could be used for landscapes based on a square
as well. However, this will not be discussed, leaving the scope of the problem to
triangular based landscapes only.

1.4. Proposed Work Plan

The development of the algorithm was approached by first creating an initial
high level algorithm based on water erosion. Further refinement of the algorithm
lead to the development of a software prototype written in the C programming
language to test and further refine the algorithm.

1.5. Suggested Reading Approach

Chapter two describes the computational background and the definitions of
the abstract data structures and operations used to define the developed
algorithm. Chapter three is the formal and informal description of the algorithm.
Each step of the algorithm constitutes a section of the chapter. Each section has
two subsections, the informal description of the algorithm and the formal
description of the algorithm.

Chapters two and three directly focus on the developed algorithm and can
be difficult to follow for the first time reader. It is suggested that the first time reader first read section 2.1 of chapter two and subsections 3.1.1, 3.2.1, 3.3.1, 3.4.1, 3.5.1, 3.6.1. of chapter three before reading the formal descriptions of the algorithm and its data structures.
2.1. Development Approach

The development of the algorithm is based around the geologic process of water erosion. The erosional process of water is the basis of the algorithm because the goal is to eliminate the abundance of depressions found on the surface. If the depressions are thought of as lakes, then the natural course of events that would follow would involve water erosion of the lake’s outlet.

The point at which a lake overflows its bank is considered its outlet. The water flowing over the outlet would continue down its drainage as a stream or river until reaching another lake or the edge of the defined region. Over geologic time the outlet and the drainage will erode downward, lowering the level of the lake until the lake is completely drained or another outlet develops. At this point the lake or depression is eliminated or the process continues on the new outlet. If this process is performed on all lakes within a defined region then at some point all lakes or depressions are eliminated. What remains is a drainage pattern with no lakes.

This is a very simple model of a natural geologic process; however, the desired result of draining all lakes and achieving a drainage pattern is achieved.
2.2. Algorithm Background and Existing Data Structures

The outcome of Carpenter's algorithm is a lower triangular array of values that represent elevations on a landscape surface. The algorithm begins with a single triangle that represents level zero of the algorithm. Each vertex of the triangle corresponds to a value in an array of three elements.

Example 2.1

100

200 300

The example is the top view of a triangle whose lower left vertex has elevation 200 units, whose lower right vertex has elevation 300 units, and whose top vertex has elevation 100 units. The triangle is tilted in three dimensional space toward the upper left.

Level one is achieved by taking the midpoint of each side of the triangle and displacing it vertically by a random amount. The values at the vertices of the triangle are used as constraints on the random value for generating the new midpoint value. How the values are used as constraints for determining the new midpoint is flexible and may vary to change surface texture. The new points are

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
then linked to form four new triangles. The triangle at the completion of level one might look as follows:

```
Example 2.2

100

188  176

200  255  300
```

The same operation can be performed on each of the four level one triangles to generate a level two with 16 triangles. The process is continued to the desired level, increasing the detail at each iteration.

A lower triangular array is the data structure used for storing the values of the vertices. A lower triangular array is usually thought of as a square matrix where only the elements in the main diagonal and to the lower left of the diagonal are used. The above triangle can be represented in a triangular array as
Example 2.3

100
188 176
200 255 300

Therefore, the level to which Carpenter's algorithm is carried out determines the size of the lower triangular array.

The rows and columns of a triangular array are referenced in the same manner as a square array. However, the value of the column index can not be larger than the value of the row index. In this paper lower triangular arrays will be pictured as in Figure 2-1.

If an element [r,c] has row index r and column index c, then it is adjacent to elements [r,c-1], [r-1,c-1], [r-1,c], [r,c+1], [r+1,c+1], and [r+1,c] (for those elements that exist in the triangular array). For example, in Figure 2-1, only the elements represented by the values 10, 45, 16, 27, 14 and 15 are adjacent to the element with value 11 at row 6, column 4. Each internal element is adjacent to six other elements.
2.3. Abstract Data Structure and Operations

The algorithm is based on the concept of a LIST as the basic data structure. The central data base from which information is obtained for the LISTs is the TRIANGULAR ARRAY created by the initial fractal landscape algorithm.

A LIST is defined as an ordered sequence of zero or more elements and/or sub-LISTs. In this paper LISTs will not have repetitions. LISTs will be enclosed in parenthesis. An example of a LIST is as follows:
Example 2.4

\[(101 \ 88 \ 76 \ (75 \ 33) \ (73 \ 45 \ 30))\]

The operations that are performed on lists are represented by the following functions.

CREATE() Creates a new list.

REMOVE(ele, lst) Removes the element, “ele” from the list, “lst”. This operation is not performed on an empty list.

REMOVE_FRONT(lst) Removes the front element from the list, “lst”. This operation is not performed on an empty list.

INSERT(ele, lst) Inserts the element, “ele” onto the front of the list, “lst”.

CHECK(ele, lst) Checks for the existence of element, “ele” in the list, “lst”.

APPEND(lst1, lst2) Appends list, “lst2” to list, “lst1”.
ATTACH(lst1, lst2) Attaches list, "lst2" to list, "lst1" to form a new list.

Example 2.5

\[ \text{lst1} = (11 \ 10) \]
\[ \text{lst2} = (13 \ 12 \ 11) \]
\[ \text{lst3} = \text{ATTACH}(\text{lst1}, \text{lst2}) \]

\[ \text{lst3 becomes:} \]
\[ ((13 \ 12 \ 11)(11 \ 10)) \]

2.4. Restricted Data Structures and Operations

There are certain data structures that are commonly used throughout the algorithm that are logical groupings that represent geologic features. Some of these data structures have various restrictions placed upon them. The LIST operations as described above can be performed upon these data structures so long as the restrictions are maintained. The data structures of this type are logically named and described as follows (refer to Figure (trilakes) for all examples).
**POINT** is an element that describes a location within the TRIANGULAR ARRAY by row and column indices. The value of a POINT is found by retrieving the value stored at the location within the TRIANGULAR ARRAY. Square brackets will be used to encase a row-column pair. This will represent a POINT.

Example 2.6

\[ [4,2] \]

In Figure 2-1 this POINT has the value 44.

**LAKE LIST** is a LIST whose elements are POINTs and/or other LAKE LISTs. A LAKE LIST represents a body of water. There are no repetitions of POINTs within a LAKE LIST or the LAKE LIST's sub-LAKE LISTs.

Example 2.7

(This is a LAKE LIST from Figure 2-1)

\[
( [6,3] [5,2] [7,4] \\
( [6,2] [7,2] [7,3] ) \\
( [6,4] [5,3] ) )
\]
OUTLET is a POINT that represents an outlet for a LAKE LIST. An OUTLET POINT is defined as a lowest POINT of the set of POINTs adjacent to the POINTs of a LAKE LIST. (The POINTs of a LAKE LIST include all POINTs of the LAKE LIST, all POINT elements of the sub-LISTs, etc.) Note that an OUTLET for a LAKE LIST is never a POINT of the LAKE LIST. If there is more than one element of equal value that are determined to be the lowest, then there are multiple OUTLETS that create an OUTLET LIST.

OUTLET LIST is a LIST of OUTLETS. The OUTLET LIST represents all OUTLETS for a LAKE LIST. All locations represented by OUTLETS in an OUTLET LIST have equal values stored in the TRIANGULAR ARRAY. An OUTLET LIST can not contain sub-LISTs.

Example 2.8

(This is the OUTLET LIST for the LAKE LIST of Example 2.7)

([8,5] [8,2] [6,5])
PAIR contains one LAKE LIST and one OUTLET LIST. A PAIR represents a body of water and its corresponding outlets. A PAIR is represented by enclosing the PAIR in curly brackets and separating the OUTLET LIST and LAKE LIST with an asterisk "*".

Example 2.9

(This is a PAIR made up of the LAKE LIST in Example 2.7 and the OUTLET LIST from Example 2.8)

{([8,5] [8,2] [6,5]) *
  ([6,3] [5,2] [7,4]
   ([6,2] [7,2] [7,3])
   ([6,4] [5,3]))}

([8,5] [8,2] [6,5])

is an OUTLET LIST for LAKE LIST,

([6,3] [5,2] [7,4]
  ([6,2] [7,2] [7,3])
  ([6,4] [5,3]))

together they form a PAIR.

PAIR LIST is an ordered LIST of PAIRs. PAIRs are the only elements that can be contained in a PAIR LIST. The PAIRs are always ordered so that the values of the OUTLETS are in ascending order.

WORK PAIR LIST is a PAIR LIST. There exists only one WORK PAIR LIST for manipulating PAIRs within the algorithm. The WORK PAIR LIST
represents all bodies of water whose shape and size is in the process of being defined.

**DEFINED PAIR LIST** is a **PAIR LIST**. There exists only one **DEFINED PAIR LIST** for manipulating **PAIRs** within the algorithm. The **DEFINED PAIR LIST** represents all bodies of water whose shape and size has been defined and is awaiting erosion.

It has been found that the [row,col] notation for a **POINT** is cumbersome to follow and that the value of a **POINT** is quite often the information needed to base a decision. Therefore in an effort to establish readability a **POINT** will be represented by its value and not its row and column. For example the value 10 will be used instead of [5,3]. The above examples, 2.7, 2.8 and 2.9 are rewritten in the value notation below.

---

**Example 2.10**

LAKE LIST

\(([6,3] [5,2] [7,4] \)
\n\(\quad ([6,2] [7,2] [7,3])\)
\n\(\quad ([6,4] [5,3]))\)

is written as (15 14 14 (13 12 11) (11 10))
Example 2.11

OUTLET LIST \{[8,5] [8,2] [6,5]\}

is written as (16 16 16)

Example 2.12

PAIR \{([8,5] [8,2] [6,5]) *
    ([6,3] [5,2] [7,4]
     ([6,2] [7,2] [7,3])
     ([6,4] [5,3]))\}

is written as \{(16 16 16) *
    (15 14 14
     (13 12 11)
     (11 10))\}

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Note that values repeat, but the values represent different locations within the TRIANGULAR ARRAY.

In addition to the basic LIST operations, a series of high level operations that are specific to the restricted data structures need to be defined. Since the following operations are commonly performed, it is necessary to create the following definitions for easier readability of the algorithm.

**INSERTPAIR**(*pair*, /*pair_lst*/)

Inserts the PAIR, "pair" into the PAIR LIST, "pair_lst". The order of a PAIR LIST is maintained, where PAIRs are always ordered so that the values of the OUTLETS are in ascending order.

**ADJACENT**(*pt*, /*lake*/)

returns true if the POINT, "pt" is adjacent to any POINT in the LAKE LIST, "lake". For example, POINT 16 at row 8, column 2 is adjacent to LAKE LIST, (15 14 14 (13 12 11) (11 10)).

**DRAIN**(*outlet*, /*lake*/)

returns true if the lowest POINT ADJACENT to the OUTLET "outlet" that is not contained within the LAKE LIST "lake" has a value less than "outlet". This will indicate that "outlet" drains the "lake". For example, POINT 5, at row 9, column 6, is the lowest POINT ADJACENT to the OUTLET 16 at row 14, column 5, and it is not contained within the LAKE LIST, (15 14 14 (13 12 11) (11 10)), that has a value less than the OUTLET'S. In this case DRAIN would return true. DRAIN will also return true for POINT 16 at row 6, column 5 for the same LAKE LIST.

**DEAD-END**(*outlet*, /*lake*/)

returns true if all POINTs ADJACENT to the OUTLET, "outlet", that are not contained within the LAKE LIST, "lake", have a value greater than or equal to the value of "outlet". This indicates that "outlet" does not drain the "lake". For example, OUTLET 16 at row 8, column 2 returns true for LAKE LIST, (15 14 14 (13 12 11) (11 10)). DEAD-END is the negation of DRAIN.

**SET_STATUS**(*outlet*, /*lake*/)

marks the OUTLET, "outlet" with the status of either ADJACENT, DRAIN, or DEAD-END. "Outlet" is marked ADJACENT if only
operation ADJACENT returns true for any LAKE LIST of the PAIRs in the WORK PAIR LIST except for "lake". An OUTLET is marked DRAIN, if only operation DRAIN returns true for any LAKE LIST of the PAIRs in the WORK PAIR LIST except for "lake". "outlet" is marked DEAD-END if only operation DEAD-END returns true for any LAKE LIST of the PAIRs in the WORK PAIR LIST. For example, SET STATUS will set the status of OUTLET 16 at row 8, column 2 to DEAD-END when given the LAKE LIST, (15 14 14 (13 12 11)(11 10)) as an argument.

It is possible for "outlet" to be both ADJACENT and DRAIN, or ADJACENT and DEAD-END, but not DRAIN and DEAD-END since DRAIN and DEAD-END are the negation of each other. If both ADJACENT and DRAIN return true for "outlet", then "outlet" is marked DRAIN. If both ADJACENT and DEAD-END return true for "outlet", then "outlet" is marked ADJACENT.

FIND_OUTLETS(lake)

creates and returns an OUTLET LIST that is the LIST of OUTLETS for the LAKE LIST, "lake". For example, FIND_OUTLETS will return (16 16 16) when given the LAKE LIST, (15 14 14 (13 12 11) (11 10)) as an argument. FIND_OUTLETS is an operation that could be implemented in a number of different ways and therefore will be left to the reader to develop a method that is satisfactory.

DRAINAGE(outlet,lake)

creates and returns one LIST of POINTs, and only POINTs, that represent a drainage path for the LAKE LIST "lake" from its OUTLET "outlet". The drainage is conceptually the path that water would flow from a "lake", at its "outlet", to the edge of the TRIANGULAR ARRAY. For example, DRAINAGE will return the LIST (7 10 15) when given the POINT 16 at row 6, column 5 as the OUTLET for the LAKE LIST (15 14 14 (13 12 11) (11 10)). Every POINT in the LIST that is returned by DRAINAGE must either be on the edge of the TRIANGULAR ARRAY or must have an ADJACENT POINT that has a lower value. The function DRAINAGE can not be used with an OUTLET and LAKE LIST combination that does not have a drainage to the edge of the TRIANGULAR ARRAY.

COMBINE_PAIRS(pair1, pair2)

ATTACHes the LAKE LIST of “pair2” to the LAKE LIST of “pair1”
and APPENDs the OUTLET LIST of "pair2" to the OUTLET LIST of "pair1". When the OUTLET LIST of "pair2" is APPENDED to the OUTLET LIST of "pair1" all duplicate POINTs are eliminated.
Chapter 3

Algorithm and Description

The contents of this chapter is the algorithm expressed in terms of the abstract data structures and operations described in chapter 2. The algorithm breaks down into three main steps that are outlined in the section 3.1. Every step and sub-step that is preceded by an asterisk "*" is broken down further in another section. Those steps that are not preceded by an asterisk "*" are complete and terminate with that section.

Every section is broken down into two subsections. The first is an informal description of a step or sub-step of the algorithm. The second subsection is the formal description of the step or sub-step expressed in terms of the abstract data structures and operations discussed in Chapter two.

3.1. High Level

3.1.1. High Level Background

The positions on a landscape surface where water begins to collect must be located. Each location is the beginning of a lake. As each location fills with water simultaneously, a number of different situations may occur at any given moment. Water at a location may pour over an outlet and down a stream into another lake or it may pour over an outlet and off the edge of the defined surface. Another
possibility is that two locations filling with water simultaneously may join to create a single lake.

At some point in time all depressions on the surface will be completely filled, and water will pour from each lake’s outlet or outlets. The water will either pour down a drainage into another lake or off the edge of the surface. At this point the size and shape of each lake is complete.

Computationally only one location can be filling with water at any given moment. The order in which each lake is filled is therefore important. If a location is filling without regard to order and an outlet point is reached, then it can not be said that the lake’s size and shape is complete. It is possible that a lake at a lower elevation will fill with water and merge with the previously filled lake to create a larger single lake. Consider the topography cross sectioned in Figure 3-1. If lake X is filled first then its outlet is determined to be point G. Lake Y is then filled and merges with Lake X to create the larger lake Z whose outlet is Point B.

The solution is to determine all the low points on the surface and each low point’s possible outlet point. The possible outlet is defined as the lowest adjacent point. It should be kept in mind that Figure 3-1 is only a two dimensional representation of a three dimensional surface, however the same process holds true for three dimensions. In Figure 3-1 points E and H are the low points. E’s possible outlet is point D, and H’s is point I. After all low points have been located, they are now filled with water beginning with the low point that has the lowest elevation outlet.
In Figure 3-1 low point E is processed first because point D is lower in elevation than point I. As lake E is filled, point D is no longer an outlet, but rather part of lake E (low points can also be referred to as lakes). Lake E's possible outlet has become part of the lake and point F is found to be the new possible outlet. Point F is also found to be part of lake E, and point C becomes the new possible outlet. However, lake E is no longer the lake with the lowest elevation outlet, since point I is lower in elevation than point C. Filling now begins with lake H. Point I is determined to be part of lake H and point G becomes the new possible outlet. Filling now goes back to lake E because point C is lower in elevation than G. However, C becomes part of lake E and point G is determined to
be the new possible outlet for lake E. Since lake E and lake H have the same possible outlet, they merge to become one lake.

Filling continues on the larger single lake until point B is found to be the final outlet. The larger single lake referred to as lake Z is said to be defined because point B has an adjacent point A that is lower and no other lakes have lower elevation outlets waiting to be filled.

By processing lakes in order of lowest elevation outlet a lake can be determined complete when its possible outlet is adjacent to a point of lower elevation and no other lakes have lower elevation outlets waiting to be processed.

An important characteristic in the above example is that lake Z is made up of two sub-lakes, X and Y. When two lakes are merged and develop into one larger lake the two lakes should not lose their definition. For example, lake X is made up of points I and H with point G as the outlet. The notation \{(G) * (I H)\} could be used to define lake X. Similarly sub-lake Y's definition would be \{(G) * (C F D E)\}. When lake X and Y are merged to form lake Z, the final definition of lake Z becomes \{(B) * (J G (C F D E) (I H))\}. Notice that definitions of sub-lake X and Y are not lost, but rather nested in the definition of lake Z.

When all lakes have been filled and their outlets determined, the next step is to erode each outlet point and all points in each outlet's drainage. The drainage is defined as that path taken by the water as it flows down slope from an outlet, terminating at the edge of the landscape or into another lake.

If lakes and their outlets are again ordered by lowest elevation outlet, the complexity of the erosion process is minimized. By eroding outlets and their
drainage in ascending order of their outlets, the erosion of each drainage point working down from the outlet will always terminate by reaching the edge of the defined surface. If lakes and their outlets were ordered in any other way, then erosion of each drainage point working down from the outlet would terminate by reaching the edge of the defined surface or by reaching another lake. This adds another condition. Consider the cross section in Figure 3-2.

The importance of the order in which lakes and their outlets are eroded is fully appreciated when taking into account that all points in an outlet's drainage are eroded. In the algorithm, the amount each point is eroded by is the same amount eroded from the outlet point. The amount to erode from each outlet will be discussed in subsection 3.4.2. In Figure 3-2 the outlet for lake X is point B and the outlet for lake Y is point E. If point B erodes to B', then point A is eroded to A'. The edge of the defined surface is reached and erosion is terminated.

Again, if the order in which the lakes are processed is neglected then lake Y could be processed before lake X. If this were to occur then E is eroded to E' and D to D'. Erosion is terminated because lake X is reached. Now D' should now become part of lake X, because D' is lower then lake X's outlet point.

However, if order is preserved by first processing the lake with the lowest elevation outlet, then an outlet's drainage will never terminate into another lake. The drainage can only terminate at the edge of the defined landscape. This simplifies the algorithm substantially.

For example, if lake X is processed first then outlet B is eroded to B' and point A to A'. Lake X is now completely drained and processing now starts with
lake Y. Lake Y will now drain to the edge of the defined surface. Outlet E erodes to E', D to D', C to C', B' to B" and A' to A".

Notice that a new outlet point develops for lake Y at point F. It is important that the method used for calculating the erosion amount created a new point that
is less than the lowest adjacent point in the lake. In Figure 3-2, the erosion of outlet B created a new point that is less than the adjacent lake point C, and the erosion of outlet E created a new point less than the adjacent lake point F. If, for example B' was greater than C, then B' would become the new outlet and little progress of eliminating the lake would have been accomplished.

3.1.2. High Level Informal Description

The algorithm breaks down into three very basic steps. First the low points on the landscape surface where water begins to collect must be located. Low points are points on the surface where all surrounding adjacent points have greater elevation. For each of the low points, find the low point’s possible outlet. The possible outlet is the lowest elevation point of the surrounding adjacent points. The low points are ordered by lowest elevation outlet and are referred to as lakes.

The second step is to continually fill the lakes until all lakes are defined. The lakes are filled in order by lowest elevation outlet. A lake is considered defined when its outlet is reached and can no longer increase in size. Water flowing out of an outlet or outlets from defined lakes may flow into other lakes or off the defined region.

The third step is to erode the lakes outlets and their outlet’s drainage points. Erosion is also performed in order by lowest elevation outlet. Water flowing over an outlet and down a drainage will erode both the outlet and all points in the path that the water flows. The outlet and the outlet’s drainage is therefore eroded to an elevation that allows the water level in the lake to lower, decreasing its size.
until either the lake is completely drained or a lower elevation outlet develops at a new location. If the lake completely drains then the process is complete for that particular lake or depression, however, if a new outlet develops at a different location then the lakes shape and size must be redefined and the erosion process is repeated with respect to the lake's new outlet.
3.1.3. High Level Formal Description

* A. Find all low POINTs in the TRIANGULAR ARRAY to create the initial WORK PAIR LIST.

* B. For each PAIR in the WORK PAIR LIST, determine the shape and size of the lake and its OUTLETs to create the DEFINED PAIR LIST.

* C. For each PAIR in the DEFINED PAIR LIST, erode an OUTLET in the OUTLET LIST and the OUTLET's drainage POINTs. DRAIN each PAIR's LAKE LIST until the DEFINED PAIR LIST is empty.
3.2. Description of Step A

3.2.1. Informal Description of Step A

The object of step A is to find all depressions or low points on the surface of the landscape, determine the outlet points for each depression, and order all depressions by lowest elevation outlet. These depressions are the start of individual lakes and must have at least one outlet, therefore outlets must be found for each individual depression. The lakes' final shape, size and outlets will be determined in Step B, however Step B requires that the lakes be ordered by lowest elevation outlet. The ordered list will be initially created in Step A.

To determine all depressions on the landscape the elevation of each point on the surface is compared with each of its surrounding adjacent points. Those points surrounded by adjacent points of greater or equal elevation are considered low points or depressions (a point on the edge of the defined surface cannot be a low point). The low points or depressions on the surface can also be referred to as lakes.

For every low point found, an outlet or outlets must be determined. The outlets may not be the final outlets for a lake, but are better thought of as possible outlets. An outlet is defined as the lowest elevation point of the surrounding adjacent points. It is possible for more than one point to become an outlet of a lake. If there is more than one point of equal elevation that has the lowest elevation of the surrounding adjacent points, then there is more than one outlet. The lakes and their corresponding outlets are inserted into a list that is

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
ordered by lowest elevation outlet, referred to as the work list.

When a point has been determined that it is lower in elevation than its adjacent points it must be checked with previously determined lakes stored in the work list. If the low point already exists as an outlet for another lake in the work list, then the lake is removed from the work list and the outlets of the removed lake are added to the original lake as lake elements. New outlets are then determined for the lake. The lake and its outlets are placed back in the ordered work list and the original point is no longer a low point. Consider the cross section in Figure 3–3.

![Figure 3-3:](image)

Every point on the surface is checked for being a depression on the surface. In Figure 3–3 the points are examined left to right, but the direction is irrelevant. Points C and E are determined to be depressions with points D and F respectively determined as their outlets. However G is initially a depression, because its surrounding points, F and H, are of greater or equal elevation, but point G's outlet
is determined to be point H. When H is considered as being a depression, it is found that H is an outlet for G. At this point H becomes part of lake G and a new outlet is found for lake G–H. The outlet is determined to be point I.
3.2.2. Formal Description of Step A

A.1 CREATE the WORK PAIR LIST as an empty list.

A.2 For each POINT "pt" in the TRIANGULAR ARRAY that has 6
ADJACENT POINTs

A.2.1 If ("pt" <= all 6 ADJACENT POINTs) then

A.2.1.1 CREATE a new LAKE LIST "lake".

A.2.1.2 INSERT "pt" into "lake".

A.2.1.3 CHECK all PAIRs in the WORK PAIR LIST for "pt"
contained in a PAIR's OUTLET LIST.

A.2.1.3.1 If "pt" is in a PAIR's OUTLET LIST

A.2.1.3.1.1 REMOVE the "pair" from WORK PAIR LIST.

A.2.1.3.1.2 For each OUTLET in "pair"'s OUTLET
LIST, INSERT OUTLET into "lake".

A.2.1.4 FIND_OUTLETS for "lake" to create a new OUTLET
LIST.

A.2.1.5 CREATE a new PAIR "new pair" consisting of LAKE
LIST "lake" and the new OUTLET LIST.

A.2.1.6 INSERT_PAIR the "new pair" into WORK PAIR LIST.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
3.3. Description of Step B

3.3.1. Informal Description of Step B

The objective of Step B is to fill the individual lakes defined in Step A with water until outlets for the lakes are located. This will result in an ordered list of lakes whose size and shape is complete.

The result of step A is a list of lakes in ascending order by lowest elevation outlet referred to as the work list. Step B will work from the work list, pulling from the top the lake with the lowest elevation outlet. The lake removed from the top of the work list will be filled with water until its outlets are higher in elevation then the current top lake in the ordered list or until one of the lake's outlets begins to drain from the lake.

An outlet is said to drain its lake when it is adjacent to a point that is lower in elevation and the adjacent point is not within the outlet's lake. At this point the lake is defined and is placed in another ordered list by lowest elevation outlet. This list is referred to as the defined list and will contain only lakes that are defined and complete.

When a lake's outlets become higher in elevation then the current top lake in the ordered list then the lake is placed back in the work list. The top lake in the work list is removed and the process is repeated until all lakes are defined and reside in the defined list.

During the process of defining a lake's outlet, shape and size, a number of different conditions can occur. After a lake is removed from the top of the work
list every outlet associated with the lake is examined and given a status. There are three conditions that an outlet may have: DEAD-END, DRAIN, or ADJACENT.

The simplest condition is DEAD-END. If an outlet is surrounded by adjacent points of higher elevation, not including those points contained in the outlet’s lake, then the outlet is given a status of DEAD-END. In Figure 3-3 the outlet I, for lake G-H is considered a DEAD-END outlet. Water at a DEAD-END outlet can not flow down or into another lake since all surrounding points are higher.

An outlet with the status of DRAIN has the characteristics normally associated with an outlet. The outlet has at least one surrounding adjacent point that is not in the outlet’s lake, that is lower in elevation. In other words, water from the lake will flow over the outlet and down its drainage. Any lake that has an outlet with a status of DRAIN is a defined lake.

An outlet that is adjacent to another lake besides its own has the status of ADJACENT. Consider Figure 3-4.

![Diagram](image.png)

Figure 3-4:
At the start of Step B point D will be listed as the outlet for point C as well as point E. Either lake C or lake E will be at the top of the work list. Since they both have the same outlet the order does not matter. If for example, point C with outlet D is removed from the top of the work list first, point D will be given a status of ADJACENT, since it is adjacent to another lake other than lake C. The ADJACENT lake is lake E.

It is possible for one outlet to be both ADJACENT and DEAD-END or ADJACENT and DRAIN. In the case of ADJACENT and DEAD-END the outlet is given the status of ADJACENT, since the two adjacent lakes will merge to form a larger lake. However, in the case of ADJACENT and DRAIN the outlet is given the status of DRAIN, since anytime a lake has an outlet that drains, the lake is complete and placed in the defined list. It is not possible for an outlet to be both DRAIN and DEAD-END since they are complete opposites of each other.

Consider the three dimensional representation of a landscape in Figure 3-5, where each point is represented by a value and the value is a point's elevation.

The lake points, 8 and 10 are circled and the outlet point 15 is boxed. If lake point 10 is the current lake being worked on, then outlet point 15 is considered both ADJACENT and DRAIN. It is both adjacent to another lake, lake 8, and adjacent to a point of lower elevation, point 12.

After all outlets for a lake have been assigned a status of either DEAD-END, DRAIN or ADJACENT the outlet statuses are compared. The comparison of the statuses determines how to process the lake.

If the comparison indicates that the one or more outlets have a status of
ADJACENT and one or more have a status of DRAIN, then all outlets of the lake that have the status of DEAD-END are removed. Now for each lake that is adjacent to an outlet of ADJACENT status combine its lake points with the outlet's lake. After all the lakes have been combined to the original lake the lake is placed in the defined list with the remaining outlets as the outlets for the lake.

If the comparison indicates that one or more outlets have a status of DRAIN and no outlets have the status of ADJACENT, then remove all the outlets that do not have the status of DRAIN. The lake is considered defined at this point and is placed in the defined list.

If the comparison indicates that one or more outlets have the status of ADJACENT and no outlets have the status of DRAIN, then for each lake that is adjacent to an outlet of ADJACENT status combine its lake points with the outlet's
lake points. Next combine all the lake's outlets to the lake itself and find new outlets for the lake. Insert the lake back into the work list to be processed based on its new outlets. In Figure 3-4 lake C has an outlet D that is adjacent to lake E. Point E is added to lake C along with the outlet D. Point B then becomes the new outlet point. The lake D-E-C with the outlet B is then placed back into the work list.

If the comparison indicates that all outlets have the status of DEAD-END, then make all outlets part of their lake and find new outlets for the lake. After the new outlets are determined place the lake back into the work list. In Figure 3-4 outlet B for the lake of points D-E-C has a status of DEAD-END. Point B then is made part of lake D-E-C and a new outlet is found. Lake D-E-C becomes B-D-E-C with outlet F and is placed back in the work list.
3.3.2. Formal Description of Step B

B.1 CREATE the DEFINED PAIR LIST as an empty LIST.

B.2 While WORK PAIR LIST is not empty

B.2.1 REMOVE_FRONT PAIR, "front pair" from the WORK PAIR LIST.

B.2.2 SET_STATUS for each OUTLET in "front pair"'s OUTLET LIST, using "front pair"'s LAKE LIST.

B.2.3 Compare status of all OUTLETS in "front pair"'s OUTLET LIST.

CASE 1: (comparison indicates one or more OUTLETS are ADJACENT and one or more OUTLETS DRAIN)

B.2.3.1 REMOVE all OUTLETS from "front pair"'s OUTLET LIST that do not have the status of DRAIN or ADJACENT.

B.2.3.2 For each "pair" from the WORK PAIR LIST whose LAKE LIST contains a POINT that is ADJACENT to any OUTLET in "front pair"'s OUTLET LIST, COMBINE_PAIRS "pair" to "front pair".

B.2.3.3 INSERT_PAIR "front pair" into DEFINED PAIR LIST.

CASE 2: (comparison indicates one or more OUTLETS
DRAIN and no OUTLETS are ADJACENT)

B.2.3.4 REMOVE all OUTLETS from "front pair"'s OUTLET LIST that do not have the status of DRAIN.

B.2.3.5 INSERT_PAIR "front pair" into DEFINED PAIR LIST.

CASE 3: (comparison indicates one or more OUTLETS are ADJACENT and no OUTLETS DRAIN)

B.2.3.6 For each "pair" from the WORK PAIR LIST whose LAKE LIST contains a POINT that is ADJACENT to any OUTLET in "front pair"'s OUTLET LIST, COMBINE_PAIRES "pair" to "front pair".

B.2.3.7 INSERT all OUTLETS in "front pair"'s OUTLET LIST into "front pair"'s LAKE LIST.

B.2.3.8 FIND_OUTLETS for "front pair"'s LAKE LIST and place new OUTLET LIST into "front pair".

B.2.3.9 INSERT_PAIR "front pair" into WORK PAIR LIST.

CASE 4: (comparison indicates all OUTLETS DEAD-END)

B.2.3.10 INSERT all OUTLETS in "front pair"'s
OUTLET LIST into "front pair"'s LAKE LIST.

B.2.3.11 FIND_OUTLETS for "front pair"'s LAKE LIST and place new OUTLET LIST into "front pair".

B.2.3.12 INSERTPAIR "front pair" into WORK PAIR LIST.
3.4. Description of Step C

3.4.1. Informal Description of Step C

Now that all lakes are defined and ordered by lowest elevation outlet in the defined list the outlets and the outlets' drainages can be eroded to eliminate all lakes. This is the objective of Step C. Until the defined list is empty the following steps are performed.

The first step is to remove the front lake from the defined list. Since this lake is the front lake in the ordered list it has the lowest elevation outlets. Therefore water flowing over the outlets and down their respective drainages can only flow off the defined landscape. It can not flow into another lake because all other lakes are at higher elevations.

It is possible for a lake to have more than one outlet, as in Figure 3-6. If a lake has more than one outlet, then all outlets of the lake have the same elevation.

In the case where a lake has more than one outlet one of the outlets of the lake and its drainage must be eroded. Since only those outlets that are adjacent to the lake can be selected for erosion the outlets that are not adjacent to the lake are removed. From the remaining outlets an outlet is randomly selected.

For the outlet chosen, the amount of erosion must be determined, and the outlet and each point in the outlet’s drainage must be eroded. The amount that is eroded from an outlet is the same amount eroded from each of its drainage points.

In order to determine the amount that must be eroded, the elevation of the lowest point adjacent to the outlet that is in the outlet's lake must be found. By
Figure 3-6:

producing a new outlet that has an elevation lower than the lowest adjacent lake point, the maximum amount of drainage of the lake is assured. In Figure 3-6 the lowest adjacent point in lake X to the outlet C is point D, and for outlet F the lowest adjacent point is E. On a three-dimensional landscape it is possible for more than one point in the lake to be adjacent to an outlet.

At this point the necessary amount to erode from the selected outlet, and each one of the outlet's drainage points must be determined and subtracted from each point's elevation. This step is further explained and broken down in section 3.5.

The final step of Step C is to drain the lake based on its eroded outlet. Since the outlet for the lake has been eroded in the previous step, the lake can no longer hold as much water, if any at all. In Figure 3-2 the erosion of lake X's
outlet point B to B' and drainage point A to A' drains all the water contained in lake X. However the erosion of lake Y's outlet E to E' and drainage points D to D', C to C', B' to B'', and A' to A'' drains all water contained by lake X above the elevation of point F. This step is also broken down into greater detail in section 3.6. However, the result of this step on lake Y is a smaller lake. The erosion and drainage of outlet E resulted in the development of a new outlet F. The water restricted by outlet F is essentially a new lake. This smaller lake is placed back in the defined list for further erosion. This guarantees the eventual elimination of all lakes. They are continually placed back in the defined list until all water is drained.
3.4.2. Formal Description of Step C

C.1 While the DEFINED PAIR LIST is not empty

C.1.1 REMOVE_FRONT PAIR "front pair" from DEFINED PAIR LIST.

C.1.2 REMOVE all POINTs from "front pair"'s OUTLET LIST that are not ADJACENT to "front pair"'s LAKE LIST.

C.1.3 Randomly select an OUTLET "current outlet" from "front pair"'s OUTLET LIST.

C.1.4 REMOVE all OUTLETS from "front pair"'s OUTLET LIST except "current outlet".

C.1.5 Find "adjacent point" by finding the smallest value of the POINTs in "front pair"'s LAKE LIST that is ADJACENT to "current outlet".

* C.1.6 Erode one of "front pair"'s OUTLETS and each point in the OUTLET's DRAINAGE.

* C.1.7 Drain the "front pair"'s LAKE LIST based on the OUTLET's new value.
3.5. Description of Step C.1.6

3.5.1. Informal Description of Step C.1.6

This step is a sub-step of Step C. There is a fair amount involved in eroding a lakes outlet point and the outlet’s respective drainage points, therefore to maintain readability the process is explained in greater detail in this section.

The result of the previous steps in Step C is the removal of the top lake from the defined list and the selection of the outlet to be eroded.

To calculate the amount to erode from the outlet and its drainage points, take the outlet’s elevation and subtract the elevation of the lowest adjacent lake value found in the previous step. Add one to the difference to create a base value. Add the base value to the difference previously calculated multiplied by a random number between zero and one.

For example, when calculating the amount to erode from point C in Figure 3-6, calculate the base by subtracting the elevation of D (300 units) from the elevation of outlet C (550 units) and add 1. The base (151 units) is added to the difference (150 units) multiplied by a random number between zero and one. The calculation is summarized as:

\[
((550 - 300) + 1) + ((550 - 300) \times \text{RANDOM()})
\]

The difference between the outlet’s elevation and the elevation of the lowest adjacent lake point is calculated in order to generate an erosion amount that will erode the outlet below the elevation of the lowest adjacent lake point. When the difference is multiplied by a random number between zero and one a product of...
something less than the difference is produced, therefore when the product is added to the difference, an erosion value between the difference and twice the difference is generated. However, if the random number is zero then the erosion value will become equal to the difference. If an erosion value equal to the difference is subtracted from the outlet point the new elevation of the outlet will be the same elevation as the lowest adjacent lake point. Using the previous example, point C' would have the same elevation as point D. If one is added to the difference in creating the base then the erosion value is assured to be greater, and never equal to the lowest adjacent lake point.

Before the calculated erosion value can be subtracted from the outlet and its drainage points, the points that make up the drainage must be determined. The drainage points should only be those points that make up a single path to the edge of the defined landscape. It is possible for a drainage to fork into more than one path, for example at an aluvial fan or the delta at the mouth of a river. Any single path to the edge is sufficient.

The erosion value can now be subtracted from the outlet's elevation and from the elevations of each point in the selected drainage path.

The above steps are performed on all outlets of a single lake.
3.5.2. Formal Description of Step C.1.2

C.1.6.1 Find the LIST of DRAINAGE POINTs "drainage" for "current outlet".

C.1.6.2 Calculate the amount to be subtracted from "current outlet"'s value and the values of its drainage POINTs.

C.1.6.2.1 Calculate the "difference" between "current outlet" and "adjacent point" by

"difference" = VALUE("current outlet") - VALUE("adjacent point")

C.1.6.2.2 Calculate the "subtraction value" by

"subtraction value" = INTEGER("difference" + 1) + ("difference" * RANDOM())

(The RANDOM number function returns a value between zero and one)

C.1.6.3 Subtract "subtraction value" from "current outlet"'s value and replace "current outlet"'s value in the TRIANGULAR ARRAY with the difference.

C.1.6.4 For each POINT "pt" in "drainage", subtract the "subtraction value" from "pt"'s value and replace "pt"'s value in the TRIANGULAR ARRAY with the difference.
3.6. Description of Step C.1.7

3.6.1. Informal Description of C.1.7

At this point in the algorithm the selected outlet and its drainage points for a lake have been eroded. The next step is to remove the water from each lake based on its outlet point's new elevation. This may result in completely draining the lake, simply reducing the lake's size or separating the lake into smaller lakes.

In Figure 3-2 the erosion of outlet B and drainage point A resulted in the complete draining of lake X, however in the case of lake Y the erosion of outlet E and drainage points D, C, B' and A' resulted in reducing the size of lake Y.

When a lake is filling with water it is possible for the lake to merge with an adjacent lake to form a larger single lake. It is noted that the larger lake was made up of two smaller sub-lakes. When draining the larger lake it is common for the smaller sub-lakes to reappear as the individual lakes. This is the separation of a lake into smaller sub-lakes. The separation of a lake into smaller lakes is demonstrated in Figure 3-3 with the erosion of outlet J to J'. The single lake that used to drain at point J has now separated into two lakes, one draining at point I the other at point F.

When a lake is drained based on its eroded outlet, either the lake's size is decreased or it separates into smaller sub-lakes. Each of these newly developed lakes are placed back into the defined list for erosion like the original lake. New outlets must be determined for each of these lakes before it is placed back in the defined list. In fact, after the outlet has been eroded and before the lake is
drained the new outlets must be found to determine at what elevation the water will stop draining. The water will stop draining at the new outlet's location.

To determine the new outlets, it is noted that a new outlet will always either be the lowest adjacent lake point to the old outlet, or a dividing point between two sub-lakes. Notice in Figure 3-7 when outlet point B is eroded that sub-lake X is completely drained, but primary lake Z's new outlet becomes point D. Point D is a dividing point between two sub-lakes, sub-lake X and sub-lake W.

![Figure 3-7:](image)

Also in Figure 3-7, an example exists of a lowest adjacent lake point becoming a new outlet. If outlet point K is chosen to be eroded, it would erode to K'. The new outlet for sub-lake Y becomes point J. J is the lowest adjacent lake point to the original outlet K.

The only time that a lake will completely drain is when at least one of its
outlets is adjacent to the lowest point in the lake. In Figure 3-2 lake X completely drained when outlet point B was eroded because point B was adjacent to the lowest point in the lake, point C. The lowest point in a lake is one of the low points on a landscape surface, originally determined in step A.
3.6.2. Formal Description of C.1.7

C.1.7.1 CREATE a new OUTLET LIST "new outlet list".

C.1.7.2 INSERT into "new outlet list" the "adjacent point" and those POINTs of equal value immediately in front and following "adjacent point" in "current outlet"'s LAKE LIST.

C.1.7.3 For each "list" in "current pair"'s LAKE LIST that is not ADJACENT to "current outlet"

C.1.7.3.1 REMOVE "list" from "current pair"'s LAKE LIST.

C.1.7.3.2 CREATE an OUTLET LIST "outlet list" with the first POINT in front of "list" in "current pair"'s LAKE LIST and every POINT of equal value immediately in front of the first POINT.

C.1.7.3.3 CREATE a new PAIR using "list" as the LAKE LIST and "outlet list" as the OUTLET LIST.

C.1.7.4 REMOVE FRONT all POINTs in front of the "new outlet list".

C.1.7.4.1 If the "new outlet list" is the only POINTs remaining in "current pair"'s LAKE LIST then REMOVE "new outlet list".

C.1.7.5 CREATE a new PAIR using all POINTs and LISTs following "new outlet list" as the LAKE LIST and the "new outlet list" as the OUTLET LIST.
C.1.7.6 INSERT all new PAIRs created in steps C.1.7.3.3 and C.1.7.5 into DEFINED PAIR LIST.
3.7. Algorithm Example

The following is a walk through of the algorithm using an example. The following example will demonstrate the development of lakes on a landscape surface and follow through the lakes erosion until eventual elimination in Step C. The landscape surface is represented by the TRIANGULAR ARRAY data structure as in Figure 2-1.

At the termination of Step A the TRIANGULAR ARRAY that is produced by the implementation of Carpenter's algorithm is shown in Figure 3-8 with the low points enclosed in circles and their respective outlets enclosed in squares.

The WORK PAIR LIST that is produced by Step A is as follows:

WORK PAIR LIST

{(11) * (10)}
{(12) * ( 9)}
{(44) * (42)}

The OUTLET LIST for LAKE LIST (10) is (11), and the OUTLET LIST for LAKE LIST (9) is (12), etc. Notice that WORK PAIR LIST is in order by lowest elevation OUTLET.

At the end of Step B all lakes have been defined and are shown in the TRIANGULAR ARRAY of Figure 3-9.

The WORK PAIR LIST is empty at the end of Step B, because all lakes are
completely defined and reside in the DEFINED PAIR LIST. The DEFINED PAIR LIST looks like the following.

**DEFINED PAIR LIST**

\{(16, 16) \ast (15, 14, 14, (13, 12, 9), (11, 10))\}
\{\{44\} \ast \{42\}\}

All OUTLETS in the DEFINED PAIR LIST at the completion of Step B have been set to a status of DRAIN. There is another POINT ADJACENT to the LAKE LIST (15
14 14 (13 12 9) (11 10)) with the value of 16 at row 8, column 2, but it had the status of DEAD-END and was removed at Step B.2.3.4.

After the completion of 3 iteration through the while loop of Step B.2 the WORK PAIR LIST looked as follows:

WORK PAIR LIST

{((14 14) * (13 12 9))
{((14 14) * (11 10))
{(44) * (42)}

Figure 3-9:
The status of both OUTLETS with the value of 14 is ADJACENT. When the front PAIR was removed from the WORK PAIR LIST, the comparison of the OUTLETS statuses indicates that one or more OUTLETS are ADJACENT and no OUTLETS DRAIN. Therefore CASE 3 of the algorithm is performed and PAIR \{(14 14) * (11 10)\} is combined with PAIR \{(14 14) * (13 12 9)\} to create PAIR \{(14 14) * ((13 12 9) (11 10))\}. The OUTLET LIST was inserted into the LAKE LIST and a new outlet list was created to eventually form the PAIR \{(15) * (14 14 (13 12 9) (11 10))\}.

During Step C, after the first front PAIR \{(16 16) * (15 14 14 (13 12 9) (11 10))\} is removed from the DEFINED PAIR LIST, the OUTLET at row 6, column 5 is randomly selected as the OUTLET to be eroded. The amount eroded is calculated at 8 after using a random number of .3 and the result rounded up from 7.5. The drainage points (15 10 7) are eroded to (7 2 -1).

After finishing Step C.1.3 the DEFINED PAIR LIST appears as follows:

\[
\text{DEFINED PAIR LIST}
\]

\[
\{(11) * (10)\} \\
\{(14 14) * (13 12 9)\} \\
\{(44) * (42)\}
\]

The front PAIR \{(11) * (10)\} is removed next and OUTLET (11) is eroded by 7 using .8 as the random number. Therefore OUTLET (11) becomes (4) and the drainage list \{(8 7 2 -1)\} becomes \{(1 0 -5 -6)\}. After Step C.1.3 is completed the DEFINED PAIR LIST becomes:
DEFINED PAIR LIST

{(14 14) * (13 12 9)}
{(44) * (42)}

The front PAIR \{(14 14) * (13 12 9)\} is removed and OUTLET (14) at row 5, column 2 is selected at random to be eroded. It is eroded by 2 using .1 as the random number. The OUTLET becomes (12) and the drainage LIST (10 4 1 0 -5 -6) becomes (8 2 -1 -2 -7 -8). After Step C.1.3 the DEFINED PAIR LIST becomes:

DEFINED PAIR LIST

{(13) * (12 9)}
{(44) * (42)}

The OUTLET (13) is eroded by 5 using .2 as the random number. The OUTLET then becomes (9) and the drainage (12 8 2 -1 -2 -7 -8) becomes (8 4 -2 -5 -6 -11 -12). The PAIR \{(44) * (42)\} is the last PAIR in the DEFINED PAIR LIST and is removed to have its OUTLET (44) eroded by 3 using a random number 0.0. The OUTLET becomes (41) with the drainage of (4 -2 -5 -6 -11 -12) becoming (1 -5 -8 -9 -14 -15). The DEFINED PAIR LIST is empty and the final TRIANGULAR ARRAY is shown in Figure 3-10.
<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>98</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>88</td>
<td>78</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>42</td>
<td>84</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>41</td>
<td>55</td>
<td>53</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>10</td>
<td>3</td>
<td>45</td>
<td>67</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>11</td>
<td>15</td>
<td>-3</td>
<td>-6</td>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>12</td>
<td>9</td>
<td>14</td>
<td>27</td>
<td>-7</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>16</td>
<td>28</td>
<td>26</td>
<td>16</td>
<td>37</td>
<td>-12</td>
<td>72</td>
</tr>
<tr>
<td>18</td>
<td>19</td>
<td>87</td>
<td>46</td>
<td>9</td>
<td>5</td>
<td>8</td>
<td>-13</td>
</tr>
</tbody>
</table>

Figure 3-10:
Chapter 4

Results

An example of an image generated from the implementation of Carpenter's algorithm can be seen in Figure 4-2. All Figures in this chapter are drawn with the base triangle rotated 180 degrees from the triangles and the triangular arrays of previous chapters. The reason for this rotation is purely for clarity. When an image is displayed without rotating the triangle, it becomes difficult to visualize the landscape. The image often looks like a single mountain, when in fact there are many different high points and valleys within the image. Displaying a fractal landscape in this manner is misleading. The high points and valleys stand out by simply rotating the triangle 180 degrees.

The images have also been tilted away from the reader by 30 degrees. If the landscape had not been tilted the reader would be looking straight down on the landscape, losing the three dimensional affect. Topography appears flat, with very little relief when viewed from directly above the surface. Any ridges and ravines, peaks or valleys could not be distinguished from each other. In viewing a surface at an angle the relief begins to standout, giving the viewer a better perspective.
4.1. Generation of Carpenter Landscapes

The Carpenter algorithm discussed in sections 1.3 and 2.1 produces the landscapes that are the basis for the developed algorithm. It is the landscapes produced by Carpenter's algorithm that have an abundance of intermountain depressions or lakes that are eroded by the erosional algorithm developed in this thesis.

The progression of a landscape produced by Carpenter's algorithm is shown in Figure 4-1. Level zero is a flat lying triangle with each vertex having an elevation of zero. The vertices could arbitrarily be assigned different values or elevations, but for visual conception the initial triangle will have all vertices set at zero. At level one the midpoints of each side have been offset vertically by a random amount proportional to the length of each side. The vertical displacement is the elevation of that point. As Carpenter's algorithm continually operates at greater levels a landscape begins to take form. It should be noticeable how the flat lying, two dimensional triangle is given relief by offsetting each vertex. Figure 4-2 is a level 5 plot of the triangles shown in Figure 4-1.

4.2. Eroded Landscapes

The intermountain depressions or lakes found on a Carpenter generated landscape are not easily seen in the images depicted in this thesis. The reason for this is perhaps the lack of additional high level operations that are necessary to bring out the detail of an image. For example, hidden line removal, color, shading and ray tracing all enhance the detail of an image. The plots generated to create
Figure 4-1:

Level 0

Level 1

Level 2

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 4-2:
the Figures in this chapter are created without the use of any high level operations. It is believed that the effect of erosion on the surface is evident and can be shown through a series of different kinds of plots.

The fractal landscape images in Mandelbrot's book, "The Fractal Geometry of Nature" incorporate the use of high level operations such as the ones mentioned. The images of Plate C11 and C12 in [Mandelbrot, 1978] are good examples of the existence of many intermountain depressions and lakes on a fractal landscape. It is evident that a natural drainage pattern does not exist on these surfaces.

4.3. Drainage Pattern

The drainage pattern of a fractal landscape can be seen graphically, rather than by visual observation of the image. The path in which water would flow from any given point on the surface can be shown by drawing a line from each internal point on the surface to its lowest adjacent point that is lower in elevation than itself. This is analogous to the path that water would follow from the given point. If there is not an adjacent point lower in elevation then no line is drawn. Those points that are low points found in step A of the algorithm would not have a line drawn from itself, but may in fact have a line drawn to itself from an adjacent point. At the completion of this process on a triangular array, the regions in which lakes exist will appear as a group of lines that are connected but a path can not be followed to the edge.

For example, the triangular array represented in Figure 3-9 has two lakes on its surface. The lake regions on the surface are circled. The drainage pattern for
Figure 3-9 is depicted in Figure 4-3 using the method described above. This is before the developed erosional algorithm is applied.

Figure 4-3:

After the erosional algorithm is applied to fractal landscapes created by Carpenter's algorithm the eroded drainages become visually evident in the images produced. The uneroded fractal landscape of Figure 4-2 is shown in Figure 4-4 after the erosional algorithm is applied to its triangular array data structure. A few of the major drainages of the landscape show up as large "V"s or ravines at the
Visually it is not possible to determine whether or not the landscape is completely free of any lakes or depressions on its surface. If the same process used to generate Figure 4-3 is applied to the same triangular array after erosion, then no evidence of lakes should appear. If from every internal point in the triangular array a path can be followed to the edge of the array, then no lakes exist on the landscape surface. For example, consider the triangular array produced by the walk through the example of section 3.3. The initial triangular array, before erosion, is shown in Figure 4-3 to demonstrate the existence of lakes. After the walk through of the erosional algorithm in section 3.3 the triangular array produced is that of Figure 3-10. Figure 4-5 is the drainage pattern of the surface after erosion of the triangular array.

Figures 4-6 and 4-7 are plots of the drainage patterns produced before and after erosion of the landscape of Figure 4-4. Figure 4-6 corresponds to the uneroded landscape of Figure 4-2, while Figure 4-7 is the drainage representation of the eroded landscape of Figure 4-4. Notice that all drainage paths of Figure 4-7 find their way to the edge, while it is not true for Figure 4-6.
Figure 4-5:
Chapter 5

Conclusion

The goal of this thesis has been to develop a concise algorithm to eliminate the existence of intermountain lakes or depressions on fractal generated landscapes for any degree of complexity. The desired result is a landscape that demonstrates a further degree of realism from the initial landscape.

It is believed that the algorithm presented in this paper accomplishes these goals. However, the images produced by the application of the algorithm are somewhat surprising. During the development process, hidden applications of the algorithm became apparent along with other methods that perhaps accomplish the same goals. This chapter discusses the images produced by the algorithm, and points out some future applications, enhancements, and other possible approaches to the same problem.

5.1. Discussion of the Algorithm

It can be seen by comparing Figures 4-2 and 4-4 that the eroded fractal landscape of Figure 4-4 is a rougher, more rugged, landscape than the uneroded landscape of Figure 4-2. The eroded landscape might be considered more a ridge and ravine type topography which is common to the Appalachian mountains of eastern North America.

It appears as though the ravines are cut deeper into the landscape than
would naturally occur. These ravines are the main arteries for water runoff as seen along the right edge of Figure 4-4. This is perhaps a result of how the amount eroded from each point within the algorithm is calculated or perhaps the fact that each point in the drainage is eroded by the same amount eroded from the outlet. This is one of the unexpected results of the algorithm. The extreme depth of the ravines might be alleviated some by developing a different method for determining the erosion amount at Steps C.1.6.2, C.1.6.3 and C.1.6.4. However, the depth to which a drainage is eroded is dependent on the maximum depth of the lake being eliminated. Complete solutions to this problem are perhaps beyond the scope of this algorithm. The appropriate method may be dependent on the desired result.

Other steps that with minor adjustments may change the image are the steps that determine the drainage points for an outlet and the random selection of the outlets for a given lake. In Step C.1.6.1 the drainage for a given outlet of a lake is determined. It is possible for a drainage to fork into two or more paths distributing the drainage's volume over the various arteries. At Step C.1.6.1, if a drainage has more than one path to the edge of the surface then one path is selected randomly for erosion. This step could be altered to take into account all channels in a drainage's distributary.

In Step C.1.3 and C.1.4 the outlet for a given lake is randomly selected from the list of outlets for the lake. In this step the outlet could perhaps be selected in a more logical manner. For a lake occurring in nature with multiple outlets, the outlet that eventually dominates over geologic time is dependent on a variety of
geologic factors. Such determining factors might be the type of bedrock being eroded, the pitch that water flows over an outlet, and the width of the outlet.

These are perhaps a few of many places in Step C that the developed algorithm may be customized to produce varying results. However, Steps A and B are less flexible. The determination of the lakes, their size, shape and outlet locations is the basis of the entire algorithm. Once the attributes of the landscape's lakes are determined then the alteration of the landscape based on the water runoff from the lakes can be adjusted to one's satisfaction. Any erosion of the landscape is performed in Step C. Steps A and B can be thought of as a continuation of Carpenter's algorithm in that it is further definition of the landscape, not alteration.

The order in which the lakes are eroded in Step C is important and without much room for change. As pointed out in section 3.1.1 the level of difficulty is minimized by eroding lakes in order of lowest elevation outlet. The order in which the lakes are processed is not directly affecting the outcome, such as the calculation of the erosion amount or paths are being eroded. This leaves all changes that may vary the results, to the steps nested within C.1.
5.2. Future Enhancements and Applications

5.2.1. Geologic Modeling Possibilities

The algorithm was developed with the geologic process of water erosion in mind. The goal in developing the algorithm was to eliminate the lakes known to exist on fractal landscapes and to create a realistic drainage pattern on the surface. To achieve this, modeling water erosion caused by lakes spilling over their outlets and down their drainages was a logical approach.

Having modeled this process in its very primitive form creates possibilities for future enhancements to the algorithm to approach a more complete geologic model. It must be kept in mind that the goal in developing this algorithm was not to model the geologic process; however, modeling the geologic process is a plausible application for the algorithm.

The first two steps of the algorithm simply define the lakes and their outlets as they exist on the fractal surface. There is no erosion performed in these two steps, therefore any enhancements made to the algorithm to model a geologic process should be made after Steps A and B. All erosion of the surface is performed in Step C.

To model water erosion a number of factors would be taken into account that were not in the developed algorithm, such as, the volume of water flowing from each lake, the bedrock type being eroded, the velocity of the draining water, the deposition of eroded material, the amount of rainfall, etc. A number of these factors could be incorporated incrementally to approach a more complete model.
Those factors that have the greatest effect should be incorporated first.

5.2.2. Control of Landscape Features

The applications of the algorithm presented in this paper are as broad as the applications of fractal landscapes themselves. But it is conceptualized that control over the erosion process may be desirable. One might want control over the number of lakes on a landscape, or perhaps the shape and size of the existing lakes. This allows the user greater control over the style of the landscape generated. The algorithm not only performs erosion, but it also defines the lakes' shape and size. It may be desirable to simply display the existing lakes and not perform any erosion. For example, the user may require a landscape that has lakes only at high elevations in an attempt to create intermountain glaciated topography, or to eliminate only the small lakes and keep large lakes for a water dominated surface.

Another result of the algorithm is a dendritic drainage pattern on the surface of the landscape. This is another attribute that could be fully utilized to perhaps display rivers and streams on the surface or a combination of rivers, streams, and lakes.

5.3. Lessons Learned

The overall result of the thesis, for me, has been more than just the final product. Educationally, the development of this thesis has been a great challenge. The knowledge that I have gained developing the algorithm and in studying the field of fractals as it relates to computer science has been most rewarding.
While developing the algorithm I have come to appreciate the difficulty in formally wording an algorithm without contradiction and confusion. I have also come to realize that with continual massaging of an algorithm through rework, implementation, rewording, and continual thought, that an algorithm can be improved substantially.

I have come to appreciate the development technique of rapid prototyping. I more or less used rapid prototyping when I implemented the algorithm. The implementation was used as a tool for checking the algorithm to discover any unforeseen conditions. The implementation pointed out a number of problems along with a number of redundancies.

One aspect of the development process that I would change is the implementation of the algorithm in the C programming language. Even though the algorithm is language independent the formal definition of the algorithm lends itself more to a Lisp programming style. I feel implementing the algorithm in Lisp would have greatly helped in the formal wording of the algorithm. This is largely because both the algorithm and Lisp are based on a list style data structure.

The study of Fractals, as it relates to computer graphics, appears to be a very open field, with plenty of room for the development of new techniques and applications. Probably the greatest benefit of this thesis is the interest it has sparked in me concerning fractals and its relation to computer graphics. It has created a desire to pursue further implementation and development techniques of fractals, particularly with landscapes and geologic modeling.
5.4. Summary

Even though the algorithm is defined in this thesis formally and informally, additional work could be done for a more complete description. Analysis of the algorithm's complexity and proof of its correctness has not been done. Both of which appear to be fairly extensive tasks, but none the less important.

Whether or not the images produced after erosion, appear more realistic than before is a matter of opinion, and judgment might be withheld until the images are further enhanced with color and shading. However, it does appear that with the erosion algorithm developed in this thesis applications for fractal landscapes are further increased, allowing greater flexibility and the option of working with more landscape features such as lakes, ponds, streams and rivers.

The developed algorithm is not a new fractal technique, but rather a tool for manipulation of fractal landscapes. Its use is up to the imagination of the user, whether it be the display of existing lakes, a base for geologic modeling, or a tool for developing rivers and streams. It does appear, however, to be a useful tool for future development of fractal landscapes.
BIBLIOGRAPHY


