DEVELOPING AN EFFICIENT IMPLEMENTATION OF A CONVEX HULL ALGORITHM

BY

Andrew Angelo T. Ang

Billy C. Chen

SCHOOL OF ARTS AND SCIENCES

ATENEO DE DAVAO UNIVERSITY

APRIL 2005

DEVELOPING AN EFFICIENT IMPLEMENTATION OF A CONVEX HULL ALGORITHM

An Independent Research

Presented to

The Faculty of the Computer Studies Division

Ateneo de Davao University

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science in Computer Science

by

Andrew Angelo T. Ang

Billy C. Chen

SCHOOL OF ARTS AND SCIENCES

ATENEO DE DAVAO UNIVERSITY

APRIL 2005

TABLE OF CONTENTS

ABSTRACT

Algorithms are methods for solving problems that are suited for computer implementation. They play a vital role in the development of a computer program. Implementing an efficient algorithm generates an efficient application. This study presents a methodology for developing an efficient implementation of a convex hull algorithm. Specifically, this study makes a comparison among implementations of existing convex hull algorithms. The results of the assessment facilitate the development of an efficient algorithm demonstrated in a convex hull rendering simulator. This study uses actual CPU time as the comparison criterion for measuring the efficiency of the algorithm.

Keywords:

Algorithm, Convex Hull, Rendering

CHAPTER I

Introduction

1.1 Background of The Study

Problem solving in Computer Science involves many phases that include gaining an understanding of the problem through designing a conceptual solution, to implementing the solution with a computer program.

One of the components of a solution, besides the data structure, is the algorithm. An algorithm is a systematic specification of a method to solve a problem within a finite amount of time. In many situations, the efficiency of the solution is the prime determinant of whether a solution is even usable. The choice of a solution's algorithm leads to significant differences in efficiency. Therefore, efficiency is one criterion used to select an algorithm and its implementation for use in an application especially in large problems.

Convex hull algorithms play a central role in the field of computational geometry. This geometric concept finds practical applications in many areas like image processing, pattern recognition, engineering and, computer graphics. As efficiency of these algorithms is crucial, it is imperative that the proponents study the different algorithms and their implementations. The study is an effort to find an efficient implementation of a convex hull algorithm applied to graphic rendering.

The motivation of this study comes from the reality that most software applications that employ the convex hull algorithms are implementing the

basic algorithms such as the Quickhull, Incremental and Gift-Wrapping algorithms. However, no study validates a comparison of the efficiency of the different implementations of these algorithms in an attempt to come up with an efficient implementation.

1.2 Statement of The Problem

The study sought to answer the general problem: How can an efficient implementation of a convex hull algorithm be developed?

Specifically, it will answer the following questions:

- What are the existing convex hull algorithms?
- How can these algorithms be implemented?
- What factors affect the efficiency of the algorithm implementation?
- How can these factors be improved to develop an efficient implementation of a convex hull algorithm?

1.3 Objective of The Study

This study aimed to develop an efficient implementation of the convex hull algorithm.

Specifically, it aimed to accomplish the following objectives:

- To identify three (3) existing convex hull algorithms
- To study how these algorithms work
- To implement the convex hull algorithms

- To make a comparison among the convex hull algorithm implementations
- To identify the factors that affect the efficiency of the algorithm implementation
- To identify the essential enhancements to improve execution time of the algorithm
- To implement the enhancements and demonstrate them in a convex hull rendering simulator

1.4 Scope and Limitation of The Study

This study focused on geometric algorithms to compute the convex hull of a given set of points. It includes the Incremental Algorithm, QuickHull Algorithm, and Gift-Wrapping Algorithm. Techniques used in 2D planes are part of the study as well.

A convex hull rendering simulator demonstrates how each implementation computes for the convex hull by stepping through each stage of each implementation.

Actual CPU time served as the only criterion for the comparison of the implementations of the algorithms.

The team used Java 5.0 of Sun Microsystems as the language for the implementation of the convex hull algorithms and the convex hull rendering simulator.

1.5 Significance of The Study

The development of an efficient implementation of the convex hull algorithm improves the execution time of generating the convex hull. Hence, it provides basis for assessing and choosing the appropriate algorithm implementation that applications use.

The enhancements of these algorithms, which facilitate efficiency in terms of execution time, provide a method for other Computer Science students to improve the implementations of other algorithms. The factors that affect algorithm efficiency serve as key attributes that students should look into when implementing algorithms.

Finally, this study will eventually help other researchers who would like to pursue further studies on improvement of execution time of other implementations of the different types of algorithms.

1.6 Definition of Terms

2D Convex Hulls - The 2D convex hull of a set of points on a single plane is the smallest convex polygon that encloses all these points.

3D Convex Hulls - The 3D convex hull of a set of points on a 3D plane is the smallest convex polygon that encloses all these points.

Triangulation – a concept used for computing 3D convex hulls. A 3-point plane created to cover a part of a hull.

Delaunay Triangulation – as a collection of edges satisfying an "empty circle" property: for each edge we can find a circle containing the edge's endpoints but not containing any other points.