Diss. ETH Nr. 7538

AN IMPLEMENTATION OF A PLANE-SWEEP ALGORITHM ON A PERSONAL COMPUTER

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH

for the degree of
Doctor of Technical Sciences

presented by
Giordano Bruno Beretta
Dipl. Math. ETH
born April 14, 1951
citizen of Lugano, Ticino

accepted on the recommendation of
Prof. Jürg Nievergelt, examiner
Prof. Walter Nef, co-examiner

1984

Abstract

Plane-sweep algorithms were introduced by Shamos to solve the intersection problem of n straightline segments in time $O(n \log n)$. Nievergelt and Preparata have shown that regions of arbitrary shape and topology can be processed with the same asymptotic effort required for line segments.

In this paper we consider a set of polygons in the plane. The polygons are given by their edges, and each edge is given by its two endpoints. The x-axis is marked as the direction of the process which leads to a *unidirectional sweep*. The segment endpoints along with all the information required to process them represent a queue of tasks to be accomplished. The queue is described by the abstract data type x-queue. A parallel to the y-axis, called a cross section, is decomposed into intervals by the regions which partition the plane. The abstract data type describing the state of the current cross section is called the y-table.

A key feature of the algorithm is that the intersection points of the originally given segments need not be known in advance, but are found during the sweep and considered as forthcoming tasks. No backtracking is needed and no intersection points are left undiscovered.

This paper focuses on implementation aspects. It is shown that an *application independent skeleton* can be extracted from the original plane-sweep algorithm. This skeleton performs most of the work; the applications must only supply the code to process a single region. The x-queue has been implemented using two different data structures: a partially ordered tree stored in a heap, and a bucket organization. The y-table has been implemented using a height-balanced binary tree. Only 4681 words of memory are required on the *Lilith* personal computer for the storage of the code of the plane-sweep skeleton.

Sweep algorithms are said to lack robustness. This is due to *numerical problems* caused by the fact that the mapping from real numbers to machine representable numbers is not bijective. For each floating-point number represented in a given machine there is a corresponding interval on the number line such that all numbers in the number line have the same machine representation. If the segment endpoints are assumed to have integer coordinate values in the range $[0, M]$, then at least $5 \log M + 4$ bits are required for the storage of the fraction part of floating-point numbers in order for the algorithm to run correctly.

A performance analysis has been carried out based on the described implementation. *Statistical experiments* confirm the hypothesized result that the running time of programs using the plane-sweep skeleton is $T(n, s) = c\ (n + s) \log n$, where n is the number of originally given segment endpoints and s is the number of segment intersection points. The constant factor has been estimated to be $c = 0.0027 \pm 0.0003$ seconds on the *Lilith* personal computer. Thus the algorithm is fast even for small values of n and s.

This paper also considers a number of applications which have been implemented using the plane-sweep skeleton, e. g. raster-scan conversion.

Zusammenfassung

Ebenen-Durchlaufs-Algorithmen wurden von Shamos eingeführt, um das Schnittpunkt-Problem für n Strecken in der Zeit $O(n \log n)$ zu lösen. Nievergelt und Preparata haben gezeigt, dass man mit dem gleichen asymptotischen Aufwand Gebiete beliebiger Topologie verarbeiten kann.

Wir betrachten eine Menge von Polygonen in der Ebene, wobei die Polygone durch ihre Kanten und die Kanten durch ihre beiden Endpunkte gegeben sind. Die x-Achse ist als Richtung des Verfahrens ausgezeichnet, was zu einem *gerichteten Durchlauf* der Ebene führt. Die Endpunkte der Strecken mit der für ihre Verarbeitung notwendigen Information, stellen eine Warteschlange zu verarbeitender Aufgaben dar. Sie ist durch den abstrakten Datentyp X-Queue beschrieben. Die Ebene wird in Gebiete unterteilt, welche jede Parallele zur y-Achse (genannt Querschnitt) in entsprechende Intervalle zerlegen. Der den Zustand des gegenwärtigen Querschnittes beschreibende abstrakte Datentyp wird Y-Table genannt.

Eine wesentliche Eigenschaft des Algorithmus ist, dass die Schnittpunkte nicht von vornherein bekannt sein müssen, sondern während dem Durchlauf gefunden werden. Dazu ist kein Backtracking erforderlich.

Implementations-Aspekte bilden den Schwerpunkt dieser Arbeit. Es wird gezeigt, dass aus dem ursprünglichen Algorithmus ein *anwendungsunabhängiges Gerüst* herausgeschält werden kann. Diese Gerüst erledigt den Grossteil der Arbeit, während für Anwendungen lediglich die für die Verarbeitung eines einzelnen Gebietes spezifischen Anweisungen beigesteuert werden müssen. Die X-Queue ist durch zwei verschiedene Datenstrukturen realisiert worden: nämlich durch einen partiell geordneten Baum (Haufen) beziehungsweise durch eine Fächerung, die Y-Table als höhenbalancierter binärer Baum. Der Programmcode des Ebenen-Durchlaufs-Gerüstes benötigt auf dem *Lilith* Arbeitsplatzrechner nur 4681 Worte Speicherplatz.

Ebenen-Durchlaufs-Algorithmen werden geringe Robustheit vorgeworfen. Dies verursacht *numerische Probleme*, da die Abbildung von reellen Zahlen auf Maschinenzahlen nicht ein-eindeutig ist. In der Tat entspricht jeder Gleitkomma-Zahl ein Intervall auf der Zahlengerade, so dass alle Zahlen im Intervall durch die gleiche Maschingrösse dargestellt werden. Unter der Annahme, dass die Strecken-Endpunkte ganzahlige Koordinaten im Bereich $[0, M]$ aufweisen, werden mindestens $5 \log M + 4$ Bits für die Speicherung des Bruchteils der Gleitkomma-Zahlen gebraucht.

Eine Effizienzanalyse basierend auf *statistischen Versuchen* hat die theoretisch vorausgesagte Ausführungszeit $T(n, s) = c\,(n + s) \log n$ bestätigt, wobei n die Anzahl der im voraus gegebenen Strecken-Endpunkte ist und s die Anzahl der Strecken-Schnittpunkte. Nach einer Schätzung auf dem *Lilith* Arbeitsplatzrechner, ist der konstante Faktor $c = 0.0027 \pm 0.0003$ Sekunden, demzufolge ist der Algorithmus auch für kleine n und s schnell.

Zusätzlich wird eine Anzahl von Anwendungen diskutiert, die das Ebenen-Durchlaufs-Gerüst verwenden, speziell auch Konvertierung von Rasterbildern.