Digital differential analyzer (graphics algorithm)

From Wikipedia, the free encyclopedia

(Redirected from Digital Differential Analyzer (graphics algorithm))

This article is about a graphics algorithm. For the digital implementation of a differential analyzer, see Digital differential analyzer.

In computer graphics, a **digital differential analyzer** (**DDA**) is hardware or software used for linear interpolation of variables over an interval between start and end point. DDAs are used for rasterization of lines, triangles and polygons. In its simplest implementation, the DDA algorithm interpolates values in interval by computing for each x_i the equations $x_i = x_{i-1}+1/m$, $y_i = y_{i-1} + m$, where $\Delta x = x_{end} - x_{start}$ and $\Delta y = y_{end} - y_{start}$ and $m = \Delta y/\Delta x$

Contents [hide]

1 Performance
2 Algorithm
3 See also
4 References

Performance [edit]

The DDA method can be implemented using floating-point or integer arithmetic. The native floating-point implementation requires one addition and one rounding operation per interpolated value (e.g. coordinate x, y, depth, color component etc.) and output result. This process is only efficient when an FPU with fast add and rounding operation is available.

The fixed-point integer operation requires two additions per output cycle, and in case of fractional part overflow, one additional increment and subtraction. The probability of fractional part overflows is proportional to the ratio m of the interpolated start/end values.

DDAs are well suited for hardware implementation and can be pipelined for maximized throughput.

This slope can be expressed in DDA as

$$m = \frac{y_{end} - y_{start}}{x_{end} - x_{start}}$$

where *m* represents the slope of the line and *c* is the y intercept. In fact any two consecutive point(x,y) lying on this line segment should satisfy the equation.

Algorithm [edit]

The DDA starts by calculating the smaller of dy or dx for a unit increment of the other. A line is then sampled at unit intervals in one coordinate and corresponding integer values nearest the line path are determined for the other coordinate.

Considering a line with positive slope, if the slope is less than or equal to 1, we sample at unit x intervals (dx=1) and compute successive y values as

$$y_{k+1} = y_k + m$$

Subscript k takes integer values starting from 0, for the 1st point and increases by 1 until endpoint is reached. y value is rounded off to nearest integer to correspond to a screen pixel.

For lines with slope greater than 1, we reverse the role of x and y i.e. we sample at dy=1 and calculate consecutive x values as

$$x_{k+1} = x_k + \frac{1}{m}$$

Similar calculations are carried out to determine pixel positions along a line with negative slope. Thus, if the absolute value of the slope is less than 1, we set dx=1 if $x_{start} < x_{end}$ i.e. the starting extreme point is at the left.

Wikipedia
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Español
Français
Italiano
Русский
Українська
Edit links

See also [edit]

- Bresenham's line algorithm is an algorithm for line rendering.
- Xiaolin Wu's line algorithm is an algorithm for line anti-aliasing

References [edit]

This article includes a list of references, related reading or external links, **but its sources remain unclear because it lacks inline citations**. Please improve this article by introducing more precise citations. *(June 2011)*

- Alan Watt: *3D Computer Graphics*, 3rd edition 2000, p. 184 (Rasterizing edges). ISBN 0-201-39855-9

Categories: Computer graphics algorithms | Digital geometry