Liang–Barsky algorithm

In computer graphics, the **Liang–Barsky algorithm** (named after You-Dong Liang and Brian A. Barsky) is a line clipping algorithm. The Liang–Barsky algorithm uses the parametric equation of a line and inequalities describing the range of the clipping window to determine the intersections between the line and the clip window. With these intersections it knows which portion of the line should be drawn. This algorithm is significantly more efficient than Cohen–Sutherland. The idea of the Liang–Barsky clipping algorithm is to do as much testing as possible before computing line intersections.

Consider first the usual parametric form of a straight line:

$$\begin{aligned} x &= x_0 + t(x_1 - x_0) = x_0 + t\Delta x, \\ y &= y_0 + t(y_1 - y_0) = y_0 + t\Delta y. \end{aligned}$$

A point is in the clip window, if

$$x_{\min} \le x_0 + t\Delta x \le x_{\max}$$

and

$$y_{\min} \le y_0 + t\Delta y \le y_{\max},$$

which can be expressed as the 4 inequalities

$$tp_i \le q_i, \quad i = 1, 2, 3, 4,$$

where

$$\begin{aligned} p_1 &= -\Delta x, & q_1 &= x_0 - x_{\min}, & &\text{(left)} \\ p_2 &= \Delta x, & q_2 &= x_{\max} - x_0, & &\text{(right)} \\ p_3 &= -\Delta y, & q_3 &= y_0 - y_{\min}, & &\text{(bottom)} \\ p_4 &= \Delta y, & q_4 &= y_{\max} - y_0. & &\text{(top)} \end{aligned}$$

To compute the final line segment:

1. A line parallel to a clipping window edge has $p_i = 0$ for that boundary.
2. If for that i, $q_i < 0$, then the line is completely outside and can be eliminated.
3. When $p_i < 0$, the line proceeds outside to inside the clip window, and when $p_i > 0$, the line proceeds inside to outside.
4. For nonzero p_i, $u = q_i/p_i$ gives t for the intersection point of the line and the window edge (possibly projected).
5. The two actual intersections of the line with the window edges, if they exist, are described by u_1 and u_2, calculated as follows. For u_1, look at boundaries for which $p_i < 0$ (i.e. outside to inside). Take u_1 to be the largest among $\{0, q_i/p_i\}$. For u_2, look at boundaries for which $p_i > 0$ (i.e. inside to outside). Take u_2 to be the minimum of $\{1, q_i/p_i\}$.
6. If $u_1 > u_2$, the line is entirely outside the clip window. If $u_1 < 0 < 1 < u_2$ it is entirely inside it.

```
// Liang--Barsky line-clipping algorithm
#include<iostream>
#include<graphics.h>
#include<math.h>

using namespace std;

// this function gives the maximum
float maxi(float arr[],int n) {
  float m = 0;
  for (int i = 0; i < n; ++i)
    if (m < arr[i])
      m = arr[i];
  return m;
}

// this function gives the minimum
float mini(float arr[], int n) {
  float m = 1;
  for (int i = 0; i < n; ++i)
    if (m > arr[i])
      m = arr[i];
  return m;
}

void liang_barsky_clipper(float xmin, float ymin, float xmax, float ymax,
                          float x1, float y1, float x2, float y2) {
  // defining variables
  float p1 = -(x2 - x1);
  float p2 = -p1;
  float p3 = -(y2 - y1);
  float p4 = -p3;

  float q1 = x1 - xmin;
  float q2 = xmax - x1;
  float q3 = y1 - ymin;
  float q4 = ymax - y1;

  float posarr[5], negarr[5];
  int posind = 1, negind = 1;
  posarr[0] = 1;
  negarr[0] = 0;

  rectangle(xmin, ymin, xmax, ymax); // drawing the clipping window

  if ((p1 == 0 && q1 < 0) || (p2 == 0 && q2 < 0) || (p3 == 0 && q3 < 0) || (p4 == 0 && q4 <
0)) {
      outtextxy(80, 80, "Line is parallel to clipping window!");
      return;
  }
  if (p1 != 0) {
    float r1 = q1 / p1;
    float r2 = q2 / p2;
    if (p1 < 0) {
      negarr[negind++] = r1; // for negative p1, add it to negative array
      posarr[posind++] = r2; // and add p2 to positive array
    } else {
      negarr[negind++] = r2;
      posarr[posind++] = r1;
    }
  }
  if (p3 != 0) {
    float r3 = q3 / p3;
    float r4 = q4 / p4;
    if (p3 < 0) {
      negarr[negind++] = r3;
      posarr[posind++] = r4;
    } else {
      negarr[negind++] = r4;
      posarr[posind++] = r3;
    }
  }

  float xn1, yn1, xn2, yn2;
  float rn1, rn2;
  rn1 = maxi(negarr, negind); // maximum of negative array
```

```
  rn2 = mini(posarr, posind); // minimum of positive array

  if (rn1 > rn2)  { // reject
    outtextxy(80, 80, "Line is outside the clipping window!");
    return;
  }

  xn1 = x1 + p2 * rn1;
  yn1 = y1 + p4 * rn1; // computing new points

  xn2 = x1 + p2 * rn2;
  yn2 = y1 + p4 * rn2;

  setcolor(CYAN);

  line(xn1, yn1, xn2, yn2); // the drawing the new line

  setlinestyle(1, 1, 0);

  line(x1, y1, xn1, yn1);
  line(x2, y2, xn2, yn2);
}

int main() {
  cout << "\nLiang-barsky line clipping";
  cout << "\nThe system window outlay is: (0,0) at bottom left and (631, 467) at top right";
  cout << "\nEnter the co-ordinates of the window(wxmin, wxmax, wymin, wymax):";
  float xmin, xmax, ymin, ymax;
  cin >> xmin >> ymin >> xmax >> ymax;
  cout << "\nEnter the end points of the line (x1, y1) and (x2, y2):";
  float x1, y1, x2, y2;
  cin >> x1 >> y1 >> x2 >> y2;

  int gd = DETECT, gm;

  // using the winbgim library for C++, initializing the graphics mode
  initgraph(&gd, &gm, "");
  liang_barsky_clipper(xmin, ymin, xmax, ymax, x1, y1, x2, y2);
  getch();
  closegraph();
}
```

See also

Algorithms used for the same purpose:

- Cyrus–Beck algorithm
- Nicholl–Lee–Nicholl algorithm
- Fast clipping

References

- Liang, Y. D., and Barsky, B., "A New Concept and Method for Line Clipping (http://cumincad.scix.net/cgi-bin/works/Show?05b4)", *ACM Transactions on Graphics*, 3(1):1–22, January 1984.
- Liang, Y. D., B. A., Barsky, and M. Slater, *Some Improvements to a Parametric Line Clipping Algorithm (http://www.academia.edu/download/46463850/CSD-92-688.pdf)*, CSD-92-688, Computer Science Division, University of California, Berkeley, 1992.
- James D. Foley. *Computer graphics: principles and practice (https://books.google.com/books/about/Computer_graphics.html?id=-4ngT05gmAQC)*. Addison-Wesley Professional, 1996. p. 117.

External links

- http://hinjang.com/articles/04.html#eight
- Skytopia: The Liang-Barsky line clipping algorithm in a nutshell! (http://www.skytopia.com/project/articles/compsci/clipping.html)