

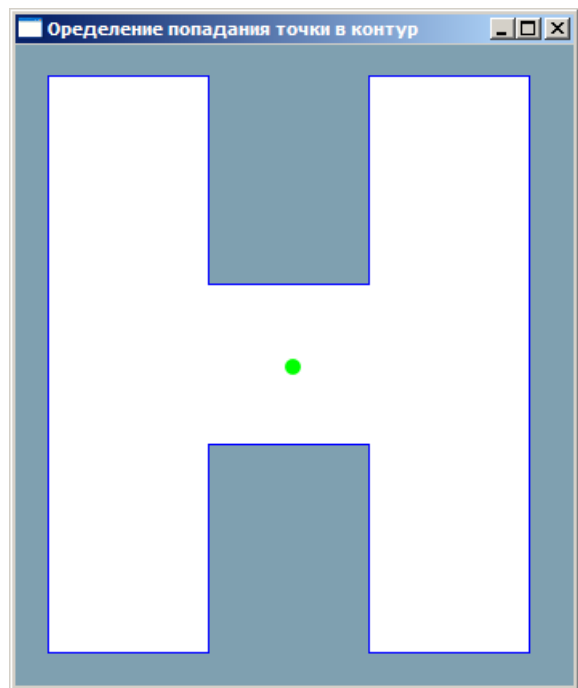
Алгоритм определения попадания точки в контур на основе комплексного анализа

🕒 4 мин 👁 128К

Алгоритмы*Математика*

Из песочницы

Привет всем Хабра людям. Хочу представить уважаемым читателям пример, когда сухая и далекая от жизни в нашем понимании высшая математика дала не плохой практический результат.



Сначала немного воспоминаний

Было это в бытность мою студентом одного из технических Вузов в 90-е, курсе наверно втором. Попал я как-то на олимпиаду по программированию. И вот на этой самой олимпиаде и было задача: задать координаты треугольника, тестовой точки на плоскости, и определить принадлежит ли эта точка области треугольника. В общем, левая задачка, но тогда я ее так и не решил. Но после задумался – над более общей задачей – принадлежность полигону. Повторюсь – была середина 90 –х, интернета не было, книжек по компьютерной геометрии не было, а были лекции по вышке и лаборатория 286 –х с турбо паскалем. И вот так совпали звезды, что как раз в то время когда я размышлял над проблемой, на вышке нам читали теорию комплексного переменного. И одна формула (о ней ниже) упала на благодатную почву. Алгоритм был придуман и реализован на паскале (к сожалению мой полтора гигабайтный винт погиб и унес в небытие этот код и кучу других моих юношеских наработок). После института я попал работать в один НИИ. Там мне пришлось заниматься разработкой ГИС для нужд работников института и собственной одной из задач было определение попадания объектов в контур. Алгоритм был переписан на C++ и отлично зарекомендовал себя в работе.

Задача для алгоритма

Дано:

Г- замкнутая ломаная (далее полигон) на плоскости, заданная координатами своих вершин (x_i, y_i) , и координата тестовой точки (x_0, y_0)

Определить:

принадлежит ли точка области D, ограниченной полигоном.

Вывод формул для последующего написания алгоритма ни в коем случае не претендует на математическую полноту и точность, а лишь демонстрирует инженерный (потребительский подход) к Царице ~~полей~~ наук.



Пояснение с рабочей-крестьянской-инженерной точки зрения:

- граница Γ наш заданный контур,
- z_0 -тестируемая точка
- $f(z)$ — комплексная функция от комплексного аргумента нигде в контуре не обращается в бесконечность.

То есть, чтобы установить принадлежность точки контуру, нам необходимо вычислить интеграл и сравнить его со значением функции в данной точке. Если они совпадают, то точка лежит в контуре. Замечание: интегральная теорема Коши гласит, что если точка не лежит в контуре, то подынтегральное выражение нигде не обращается в бесконечность, то интеграл равен нулю. Это упрощает дело — нужно лишь вычислить интеграл и проверить его на равенство нулю: равен нулю точка не контура, отличен — лежит в контуре.

Займемся вычислением интеграла. За $f(z)$ примем простую функцию 1. Не нарушая общности можно за z_0 принять точку 0 (всегда можно сдвинуть координаты).



Избавляемся от мнимой единицы в знаменателе подынтегральной части и расщепим интеграл на действительную и мнимую части:

$$\int_{\Gamma} \frac{z}{dz} = \int_{\Gamma} \frac{x dx + y dy}{x^2 + y^2} + i \int_{\Gamma} \frac{x dy - y dx}{x^2 + y^2}$$

Получилось два криволинейных интеграла II рода.

Вычислим первый

$$\int_{\Gamma} \frac{x dx + y dy}{x^2 + y^2} = \int_{\Gamma} \frac{x dx}{x^2 + y^2} + \int_{\Gamma} \frac{y dy}{x^2 + y^2} = \int_{\Gamma} P(x, y) dx + Q(x, y) dy$$

$$\frac{\partial P}{\partial y} = \frac{-2xy}{(x^2 + y^2)^2}$$

$$\frac{\partial Q}{\partial x} = \frac{-2xy}{(x^2 + y^2)^2}$$

$$\frac{\partial Q}{\partial x} \equiv \frac{\partial P}{\partial y}$$

Выполняется условие не зависимости интеграла от пути, следовательно, первый интеграл равен нулю и его вычислять не нужно.

С мнимой частью такой фокус не проходит. Вспоминаем, что наша граница состоит из отрезков прямых, получаем:



Где Γ_i - это отрезок $(x_i, y_i) - (x_{i+1}, y_{i+1})$

Вычислим i -ый интеграл. Для этого запишем уравнение i -го отрезка в параметрическом виде

$$\begin{cases} x(t) = (x_{i+1} - x_i)t + x_i \\ y(t) = (y_{i+1} - y_i)t + y_i \end{cases}; 0 \leq t \leq 1$$

Подставим в интеграл

$$\int_{\Gamma_i} = \int_0^1 \frac{((x_{i+1} - x_i)(y_{i+1} - y_i)t + x_i) dy - ((y_{i+1} - y_i)t + y_i)(x_{i+1} - x_i) dt}{((x_{i+1} - x_i)t + x_i)^2 + ((y_{i+1} - y_i)t + y_i)^2}$$

и после громоздких и нудных преобразований получим следующую прельстивую формулу:

$$\int_{\Gamma_i} = \arg \operatorname{tg} \left(\frac{x_i^2 + y_i^2 - x_{i+1}x_i - y_{i+1}y_i}{x_i y_{i+1} - x_{i+1}y_i} \right) + \arg \operatorname{tg} \left(\frac{x_{i+1}^2 + y_{i+1}^2 - x_{i+1}x_i - y_{i+1}y_i}{x_i y_{i+1} - x_{i+1}y_i} \right)$$

Окончательно получаем

$$\int_{\Gamma} \frac{z}{dz} = \sum_i \arg tg \left(\frac{x_i^2 + y_i^2 - x_{i+1}x_i - y_{i+1}y_i}{x_i y_{i+1} - x_{i+1} y_i} \right) + \arg tg \left(\frac{x_{i+1}^2 + y_{i+1}^2 - x_{i+1}x_i - y_{i+1}y_i}{x_i y_{i+1} - x_{i+1} y_i} \right)$$

Алгоритм на C++:

```

    template <class T>
bool pt_in_polygon(const T &test,const std::vector &polygon)
{
    if (polygon.size()<3) return false;

    std::vector::const_iterator end=polygon.end();

    T last_pt=polygon.back();

    last_pt.x-=test.x;
    last_pt.y-=test.y;

    double sum=0.0;

    for(
        std::vector::const_iterator iter=polygon.begin();
        iter!=end;
        ++iter
    )
    {
        T cur_pt=*iter;
        cur_pt.x-=test.x;
        cur_pt.y-=test.y;

        double del= last_pt.x*cur_pt.y-cur_pt.x*last_pt.y;
        double xy= cur_pt.x*last_pt.x+cur_pt.y*last_pt.y;

        sum+=
        (
            atan((last_pt.x*last_pt.x+last_pt.y*last_pt.y - xy)/del)+
            atan((cur_pt.x*cur_pt.x+cur_pt.y*cur_pt.y- xy )/del)
        );

        last_pt=cur_pt;

    }

    return fabs(sum)>eps;

}

```

T – тип точки, например:

```

struct PointD
{
    double x,y;
};

```

Пример

Пример работы алгоритма написан с применением самой на мой взгляд великой библиотеки 2D графики: Anti-Grain Geometry (AGG) .

Управление:

клик левой кнопкой – добавление новой точки контура

правой кнопкой – замыкание контура

левой с зажатым Shift-ом – перенос тестовой точки

PS

Господа, кому интересно, привожу более быстрый алгоритм. Уже не мой.

Отдельное и огромное спасибо forgotten за статейку.

```
template bool pt_in_polygon2(const T &test,const std::vector &polygon)
{

static const int q_patt[2][2]= { {0,1}, {3,2} };

if (polygon.size()<3) return false;

std::vector::const_iterator end=polygon.end();
T pred_pt=polygon.back();
pred_pt.x-=test.x;
pred_pt.y-=test.y;

int pred_q=q_patt[pred_pt.y<0][pred_pt.x<0];

int w=0;

for(std::vector::const_iterator iter=polygon.begin();iter!=end;++iter)
{
T cur_pt = *iter;

cur_pt.x-=test.x;
cur_pt.y-=test.y;

int q=q_patt[cur_pt.y<0][cur_pt.x<0];

switch (q-pred_q)
{
case -3:++w;break;
case 3:--w;break;
case -2:if(pred_pt.x*cur_pt.y>=pred_pt.y*cur_pt.x) ++w;break;
case 2:if(!(pred_pt.x*cur_pt.y>=pred_pt.y*cur_pt.x)) --w;break;
}

pred_pt = cur_pt;
pred_q = q;

}

return w!=0;
```

}

Теги: математикас++алгоритм2d графика

Хабы: АлгоритмыМатематика

Редакторский дайджест

Присылаем лучшие статьи раз в месяц



Электронпочта

→



0

0

Карма Рейтинг

Колганов Константин @Coolgun

Пользователь

Реклама

Комментарии 100



• Laplace 1 авг 2011 в 11:22

Выглядит круто, а теперь вопрос, разрешима ли задача определения, пересекаются ли (ну или касаются) две фигуры друг друга? Пусть даже вторая для простоты будет отрезком.

♦ +4 Ответить 📖 ...

• aml 1 авг 2011 в 11:55 ^

Понадобится примитив проверки, пересекаются ли два отрезка. Соответственно смотрим, пересекается ли второй отрезок хоть с одной стороной многоугольника. Если да, то задача решена — ответ ДА. Если нет, то либо отрезок находится целиком вне фигуры, или целиком внутри. Это проверяется тестом на принадлежность фигуре одного из концов отрезка. Если принадлежит, то ответ — ДА, иначе — НЕТ.

♦ +5 Ответить 📖 ...

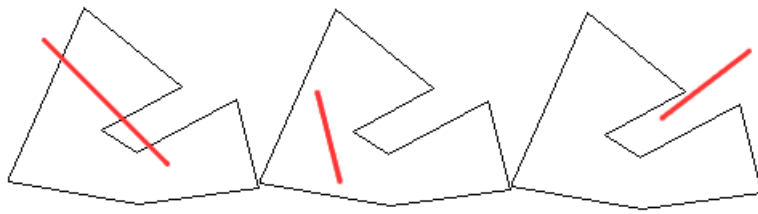
• Laplace 1 авг 2011 в 16:00 ^

Я может быть понял неправильно, но Вы хотите сказать, что пересечение отрезка и фигуры можно определить проверив принадлежность фигуре его концов?

♦ 0 Ответить 📖 ...

• aml 1 авг 2011 в 16:17 ^

Нет, смотрите. Есть три базовых случая — отрезок пересекает границу, отрезок не пересекает границу и целиком лежит в фигуре, и отрезок не пересекает границу и лежит вне фигуры:



Первый случай проверяется наличием пересечения отрезка с границами многоугольника. Второй и третий различаются по наличию одного из концов отрезка внутри многоугольника.

◆ +3 Ответить

○ **Laplace** 1 авг 2011 в 16:24 ^

Понял. Да для многоугольников должно работать.

◆ 0 Ответить

○ **ykrop** 1 авг 2011 в 18:32 ^

дополню: насколько я помню из курса — для выпуклой фигуры проверка значительно упрощается.

◆ 0 Ответить

○ **Coolgun** 1 авг 2011 в 11:44

Хм...

Спасибо за вопрос, первое, что приходи на ум — разбить отрезок на точки с каким-то шагом и проверять их вхождение. Но чувствую, что это некрасиво и очень не оптимально. Подумаю над аналитическим решением.

◆ +2 Ответить

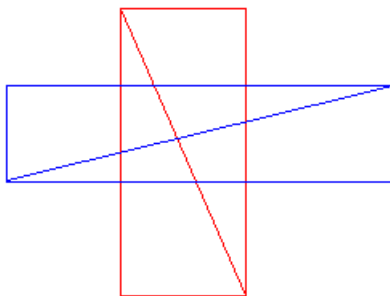
○ **Levsha100** 1 авг 2011 в 12:11 ^

Првое что мне пришло на ум: разбить фигуру на выпуклые многоугольники(либо треугольники) а далее вхождение точки в выпуклую фигуру(а это проще простого).

◆ 0 Ответить

○ **Coolgun** 1 авг 2011 в 12:24 ^

Как разрешится такой пример?




◆ +2 Ответить

○ **dom1n1k** 1 авг 2011 в 13:10 ^

Вхождение точки в выпуклую фигуру — да, проверяется очень просто.


А вот триангуляция произвольного полигона в общем случае — не такая уж тривиальная задача, как может показаться на первый взгляд.

♦ +2 Ответить

o  **ykrop** 1 авг 2011 в 18:35 ^

надеюсь, не ошибусь, если скажу, что в современной компьютерной графике графическим примитивом как раз является треугольник. из его преимуществ, помимо прочего, то что сумма углов его всегда 180, это выпуклая фигура и он всегда однозначно определяет плоскость, что удобно в трехмерных преобразованиях.


♦ +3 Ответить

o  **ханер** 1 авг 2011 в 13:06 ^

Как по мне, то намного проще проверить пересечение отрезка с каждой линией контура. Пересекает хоть одну линию контура — вуаля. Ну еще нужна проверка того, что весь отрезок целиком не лежит внутри полигона (тогда нет пересечения с линией контура, но отрезок внутри) — любой из концов отрезка проверить лежит внутри полигона или нет.

3.bl. Надеюсь не надо рассказывать как проверить пересекаются ли 2 отрезка.

♦ 0 Ответить

o  **Polsky** 1 авг 2011 в 11:51


С познавательной точки зрения подход, определённо, интересный, но на практике есть более быстрый и надёжный алгоритм. Считается количество пересечений горизонтального луча из точки с рёбрами полигона. Чётное число пересечений — точка снаружи, нечётное — внутри. Этот алгоритм более быстрый, т.к. для определения факта пересечения луча и отрезка не требуется считать тангенсы, и даже можно обойтись без деления. А более надёжный, т.к. факт пересечения можно определить с абсолютной точностью (т.е. нет погрешности, этого >eps).

♦ +32 Ответить

o  **8bitjoey** 1 авг 2011 в 11:53 ^

насколько я помню, это только для выпуклых многоугольников

♦ -3 Ответить

o  **red1ynx** 1 авг 2011 в 12:07 ^

Не обязательно.

♦ +2 Ответить

o // **anarleen** 1 авг 2011 в 12:08 ^


Нарисуйте на листике просто пару многоугольников и проверьте.

♦ +1 Ответить

o  **snizovtsev** 1 авг 2011 в 12:08 ^

С выпуклым это вообще не задача.

♦ 0 Ответить

o  **SHVV** 1 авг 2011 в 12:22 ^

С выпуклыми многоугольниками всё еще проще.

Достаточно проверить с какой стороны относительно каждого ребра лежит точка. Как только обнаруживается ребро, относительно которого точка лежит с противоположной стороны, можно перебор прекращать, так как точка за пределами многогранника. Соответственно, если точка лежит по одну сторону от всех рёбер, то она лежит внутри.

Сторона определяется знаком векторного произведения $(P_2 - P_1) \wedge (P - P_1)$, где P_i — точки многогранника, а P — тестируемая точка.

♦ +3 Ответить

alexkolzov 1 авг 2011 в 16:33 ^

Кстати, при нуле пересечений возникает неопределенность и требуется выбрать другое направление луча

♦ 0 Ответить

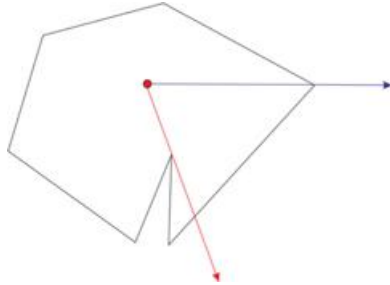
Eol 2 авг 2011 в 19:24 ^

Да нет, если ноль пересечений, то точка лежит вне полигона.

♦ 0 Ответить

alexkolzov 2 авг 2011 в 21:16 ^

Неопределенность возникает, если луч пересекает вершину невыпуклого полигона. Проще всего такие пересечения не считать, но тогда может возникнуть ноль пересечений даже если точка внутри. Вот пример:



Красный луч — два пересечения, но точка внутри полигона. Для устранения ошибки достаточно не считать пересечение с вершиной. Но тогда получаем следующее. Синий луч — одно пересечение с вершиной. Вершину не считаем, итого ноль пересечений.

Вот вам и неопределенность

♦ 0 Ответить

Eol 2 авг 2011 в 22:08 ^

Значит не считать вершину — плохая идея, и не считать пересечения с вершиной всё-таки недостаточно :)


Более того, есть еще случай, когда все ребро лежит на луче, его тоже неплохо было бы рассмотреть отдельно.

♦ 0 Ответить

alexkolzov 2 авг 2011 в 22:59 ^

Не считать вершину — простейшее решение, причем полное. Единственный недостаток — необходимость пуска другого луча в случае неопределенности. Ребро лежит на луче — это интересный случай, его разрешение зависит от того, считать точку, лежащую на ограничивающей кривой, принадлежащей фигуре. Если да — то случай сводится к лучу, проходящему через вершину (для его разрешения возможны варианты), если нет — то вершину не считаем. По другому варианту, не требующему повторного пуска луча, в точке пересечения нужно считать производную функции полигона (для полигона весьма тривиальная задача), если она равна нулю — то вершина есть экстремум и ее считать не надо.


♦ 0 Ответить

- 


○



alexkolzov 1 авг 2011 в 16:34 ^

Простите, не Вам

 0

[Ответить](#)





- 


○



alexkolzov 1 авг 2011 в 16:30 ^

Описанный в комментарии алгоритм действительно только для выпуклых. Для любых многоугольников нужно отсеивать пересечение луча с вершинами полинома

 0

[Ответить](#)





- 


○



alexkolzov 1 авг 2011 в 16:36 ^

не полинома, а полигона, конечно

 0

[Ответить](#)





- 


○



mokus 1 авг 2011 в 12:15 ^

Луч может быть и не горизонтальный, а вообще в любую сторону — правильно я понял?

 0

[Ответить](#)





- 


○



Unlogic 1 авг 2011 в 12:24 ^

Да, но важно не попасть лучом в вершину многоугольника. Я лично пускал 3 (три) случайных луча, 3 одинаковых (с точки зрения четности\нечетности пересечений) результата даёт судить о их верности с достаточной вероятностью.

 0

[Ответить](#)





- 


○



SergeyBankevich 1 авг 2011 в 13:20 ^

Ну можно и с попаданием. Только тогда надо считать отрезки полуоткрытыми и аккуратно разбираться с точностью, если координаты нецелые.

 0

[Ответить](#)






- 

○


alexkolzov 1 авг 2011 в 17:37 ^



Вопрос на засыпку. Какова эта вероятность? Вот пример, где Ваша программа ошибается



 0


[Ответить](#)





- 


○



alexkolzov 1 авг 2011 в 17:39 ^

 Полигон

 0

[Ответить](#)





- 


○



Zabatov 1 авг 2011 в 14:50 ^

правильно. Ведь вы можете повернуть не ось, а фигуру, результат не изменится


 0

[Ответить](#)




- 

○


НЛО прилетело и опубликовало эту надпись здесь
- 

○

Polsky 1 авг 2011 в 12:32 ^

Проблем не будет. Представьте, что горизонтальный луч приподнят на бесконечно малую величину. Тогда если обе вершины ребра лежали на прямой луча, ребро станет параллельно лучу (нет пересечения). Если ребро касалось луча верхней вершиной — касание исчезнет (опять нет пересечения). Наконец, если ребро касалось луча нижней вершиной касание превратится в пересечение. Алгоритмически это выглядит просто как засчитывание пересечений только когда ребро либо явно пересекает луч, либо касается его нижней вершиной.

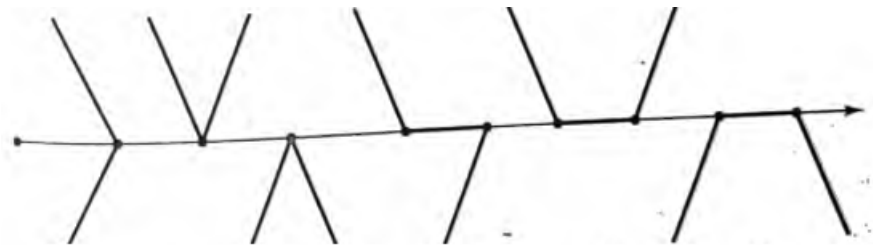
♦ +2 Ответить

○  **Vidocq** 3 авг 2011 в 16:06 ^

хотелось только добавить элюстрации

♦ 0 Ответить

○  **Vidocq** 3 авг 2011 в 16:39 ^



♦ 0 Ответить

○  **Aspos** 1 авг 2011 в 12:44 ^


Если координаты вершин целочисленные, луч может идти с наклоном $2^{0.5}$, такой луч не пересекает целочисленных точек

♦ 0 Ответить

○  **dime** 1 авг 2011 в 13:57 ^

Проще луч на пол-пикселя сместить относительно точки.
Но это всё только для случая, когда рёбра фигуры пересекаются в целочисленных точках.
Что в общем случае совершенно не обязательно.

♦ +1 Ответить

○  **acerv** 1 авг 2011 в 12:24 ^

поясните пожалуйста, про лучи. как их правильно рисовать?


есть более подробное описание метода?

♦ +1 Ответить

○  **zerkms** 1 авг 2011 в 12:36 ^


Просто луч в любую сторону, в произвольную

♦ 0 Ответить

○  **Polsky** 1 авг 2011 в 12:42 ^

Более подробно — в литературе, например: Шикин, Боресков «Компьютерная графика полигональные модели»

♦ 0 Ответить

o  **Coolgun** 1 авг 2011 в 12:42 ^


Книга Майкла Ласло — Вычислительная Геометрия и компьютерная Графика на C++

◆ 0 Ответить  ...

o  **icc** 1 авг 2011 в 12:32 ^

А математическое обоснование есть или выведено опытным путем?

◆ 0 Ответить  ...

o  **Polsky** 1 авг 2011 в 13:08 ^

Мне достаточно интуитивного. Замкнутая кривая пересекается, например (без потери общности), горизонтальной прямой чётное число раз, касания не рассматриваем, применяя всё то же поднятие прямой на бесконечно малую величину (см. выше). Для кривой без самопересечений (а именно такие ограничивают полигон) пары точек вдоль горизонтальной оси ограничивают отрезки прямой внутри области. Некоторая точка разбивает прямую на два луча. И либо на обоих лучах будет чётное число точек пересечения, что означает что точка не попала между пары точек, задающих отрезок внутри области. Либо на обоих лучах будет нечётное число точек пересечения, и значит точка попала на отрезок внутри области, а значит и в саму область.

◆ 0 Ответить  ...

o  **ckald** 1 авг 2011 в 23:02 ^

Математическое обоснование ровно такое же, как и у теоремы интегрирования Гаусса (хотя она может носить и любое другое название). Возьмем, например, заряд и сферу. Если заряд находится внутри, поток поля будет ненулевым. Если снаружи — понятно, общий поток сведется к нулю.

А в этом случае у нас тоже самое на плоскости — теорема Стокса (хотя, вообще говоря, обе теоремы — частные случаи обобщенной формулы Стокса для дифференциальных форм).

Возможно, связь основных векторных теорем и лучей в случайные стороны неочевидна на первый взгляд, но все так :) А насчет крайнего случая с попаданием луча в вершину — так в математике такого не бывает))


◆ +1 Ответить  ...

o  **alexkolzov** 2 авг 2011 в 21:19 ^

> А насчет крайнего случая с попаданием луча в вершину — так в математике такого не бывает))

Это почему, простите?

◆ 0 Ответить  ...

o  **ckald** 2 авг 2011 в 21:26 ^

Неосторожно ляпнул. Имел в виду тот случай с теоремой Гаусса, когда мы рассматриваем гладкую замкнутую поверхность и некий луч, проходящий касательно снаружи нее. Эта точка ведь просто не попадает в поток.

Так будет лучше?

◆ 0 Ответить  ...

o  **alexkolzov** 2 авг 2011 в 21:58 ^

Намного. Хотя здесь не совсем корректно сравнение с теоремой Гаусса, поскольку в ее формулировке используется понятие заряда внутри поверхности, т.е. определено замкнутое множество точек, или другими словами, заряд на самой поверхности не

рассматривается. Применительно к рассматриваемой задаче мы не можем пренебрегать точками, лежащими на ограничивающей кривой. Хотя, в общем, это вопрос терминологии — считать ли точку на ограничивающей кривой лежащей внутри фигуры. В теории обычно — нет, на практике чаще всего — да.

◆ 0 Ответить

ckald 2 авг 2011 в 22:15 ^

Моя аналогия пришла еще и от детского объяснения Гаусса. Когда вводят линии поля, каждая из которых пересекает поверхность четное или нечетное количество раз, а плотность их падает с расстоянием — потому они компенсируются, если точка снаружи.

Знаете, это все довольно забавно. Изначально задача относится к прикладным задачам численных методов (назовем это так — хотя бы потому что речь идет о решении ее на компьютере), автор статьи решил очень успешно приткнуть теорию функций комплексной переменной, а мне все электродинамика мерещится :)

◆ 0 Ответить

alexkolzov 2 авг 2011 в 22:46 ^

По мне решение автора весьма остроумное и вполне имеет право на жизнь. Так что может векторный анализ вкупе с дифференциальной геометрией подкинет методу. Интересно, кстати, что решение автора распространяется на любые плоские фигуры, не только полигоны

◆ 0 Ответить

ckald 2 авг 2011 в 22:54 ^

Меня не оставляет ощущение, что я о чем-то таком тоже думал — поэтому я в восторге от решения. Хоть оно и медленнее других, но оно использует свойства нашего пространства — самые основные законы природы.

Точно также меня беспокоит идея интегрирования по произвольным n -мерным многообразиям с помощью дифференциальных форм и обобщенной формулы Стокса — уж больно элегантно и соблазнительно выглядит оператор границы. Но если мы возьмем хотя бы n -мерную сферу — хороший метод через квадрат интеграла Пуассона или плохой через n -мерный якобиан сферических координат окажутся во многие-много раз быстрее. Потому что внешние формы сильно связаны с определителями, а определитель — уж очень дорогая операция.

◆ 0 Ответить

alexkolzov 2 авг 2011 в 23:04 ^

Мне кажется, что овчинка стоит выделки. В конце концов дороговизна отдельной операции — ничто по сравнению с многообразием применений. В моей практике попадались задачи, где как раз требовалось определить факт попадания точки внутрь области, ограниченной кривой в N -мерном пространстве. Я их решал аппроксимацией полигоном и подсчетом углов, но не оставляло ощущение «костыля». Так что наработки в области интегрирования по многообразиям могут оказаться очень востребованными

◆ 0 Ответить

ckald 2 авг 2011 в 23:10 ^

Вы первый, кто не назвал меня психом :) Аспиранты, кандидаты и доценты скептически отнеслись к этому

0 Ответить

alexkolzov 2 авг 2011 в 23:12

Знакомая история :)

0 Ответить

wildMan 1 авг 2011 в 12:44

Ага. Простейшая задача «Обрезка карты по границе области».

Оперирует цифрами порядка:

- тысяча точек в полигоне границы
- 5-6 миллионов точек всего
- 3-4 миллиона точек попадает в границы

Сами посчитаете количество операций на решение?

-1 Ответить

Polsky 1 авг 2011 в 13:23

Не понял, о чём речь.

1. Обрезка, оно же отсечение — это совсем другая задача, решается другими алгоритмами.
2. Судя по тому, что 3-4 млн. точек из 5-6 млн попадает на границы — имеются ввиду не точки, а пиксели. А в статье решается непрерывная задача. Для дискретных задач, опять же есть другие алгоритмы.

+1 Ответить

wildMan 1 авг 2011 в 13:31

Географическая карта. Объекты — точки заданные координатами. Соответственно задача именно такая — определить попадание в полигон. Дальше — или забыть, или перенести в результат.

И попадают не «на границы», а в «границы» — т.е. должны оказаться в результирующем наборе.

Соответственно по алгоритму «сканирующей прямой» для того чтобы определить что точка нам нужна/не нужна необходимо сделать $\text{количество_проверок} = \text{количество_точек} * \text{количество_рёбер_границы}$

И это количество одинаково как для случая попадания, так и для случая непопадания.

В реальных условиях это непозволительные затраты.

0 Ответить

Polsky 1 авг 2011 в 13:45

Ясно. Согласен, что прямое решение для такой задачи работает медленно. Обычно, когда данных много применяют какую-то пространственную иерархию (дерево, сетку и т.п.). И сначала определяют как узел иерархии соотносится с областью — если он полностью внутри или снаружи, то его внутренние элементы можно дальше не проверять. А вот если узел лежит в области частично, то в любом случае придётся использовать точную проверку для всех элементов этого узла.

+2 Ответить

Bashuk 1 авг 2011 в 13:47

На самом деле, случай с картами можно (с натяжкой) считать дискретным, т.к. используется определенная точность (скажем, до $1e-5$, хотя в NASA все вычисления

проводятся до $1e-16$).

Количество_проверок == количество_ребер, и вот почему. Например, луч ориентирован вправо. Для каждого отрезка границы (ребра) нужно найти точку пересечения прямой-продолжения луча и прямой-продолжения отрезка, и проверить принадлежность этой точки лучу и отрезку, что делается довольно просто за $O(1)$. Таким образом, алгоритм не зависит от количества_точек как параметра.

0 Ответить

Ogoun 1 авг 2011 в 12:53 ^

Тангенсы можно высчитывать по таблице, в математических библиотеках так и сделано, кажется, но можно и самому сделать, т.е. на них скорость уже не потеряем. Ну и как сказано уже, ваш подход только для выпуклых многоугольников.

-3 Ответить

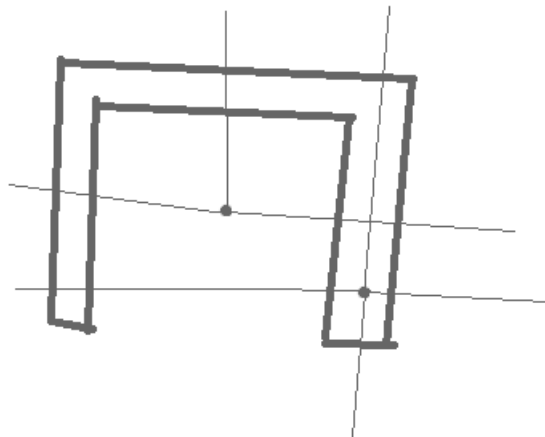
Rustam 1 авг 2011 в 13:35 ^

работает для любого многоугольника.

0 Ответить

Ogoun 1 авг 2011 в 16:44 ^

Хорошо, тогда почему у меня и у точки снаружи и у точки внутри по 4 пересечения с ребрами у горизонтального луча, и по 6 если и горизонтальный и вертикальный провести?



Где ошибка?

>>Считается количество пересечений горизонтального луча из точки с рёбрами полигона. Чётное число пересечений — точка снаружи, нечётное — внутри.

-3 Ответить

Polsky 1 авг 2011 в 17:05 ^

Ошибка в том, что слишком много лучей. Должен быть один луч для точки. А на рисунке три луча из левой точки, и четыре из правой. Понятно, что четыре нечётных числа в сумме дадут чётное :).

+1 Ответить

bioroot 1 авг 2011 в 17:06 ^

Луч — это когда в одну сторону от точки, а не в обе.

+4 Ответить

o  **skor** 1 авг 2011 в 13:25 ^

Абсолютно верно, более того, подсчётом пересечений и касаний можно определить находится ли точка внутри вообще любой фигуры, необязательно многогранника.

◆ +1 Ответить  ...

o  **Bashuk** 1 авг 2011 в 13:41 ^

А по-моему еще проще — через площадь. Если сумма площадей треугольников, которые построены на ребрах полигона и заданной точке, равна общей площади полигона — следовательно, точка лежит внутри. Алгоритм также работает и для не выпуклых многоугольников, сложность — линейная, равно как и в Вашем подходе. Да, кстати, насчет «более быстрый» — это будет мало ощутимо. Сложность будет тех же $O(N)$, просто константа побольше. Впрочем, конечно же, если можно сократить константу — почему бы этого не сделать? :) Насчет абсолютной точности — я с Вами не соглашусь. С чем я могу согласиться, так это с тем, что координаты пересечения луча и отрезка вычисляются достаточно точно (гораздо точнее, чем использование тангенсов), но я пока не представляю себе, как вы решили избавиться от ерс в вычислительной геометрии — буду рад услышать ответ.

◆ +3 Ответить  ...

o НЛО прилетело и опубликовало эту надпись здесь

o  **Bashuk** 1 авг 2011 в 22:03 ^

Протест и баста! :)


Окей, вот вам тест для формы Н: нашу точку выносим «далеко-далеко». Тогда сумма площадей треугольников будет, очевидно, «очень-очень большой», а площадь самой буквы Н — сравнительно маленькой.

Возможно, вы имели ввиду ориентированную площадь? В таком случае да, он не всегда сработает вообще. Но для абсолютных (неотрицательных) площадей такой алгоритм, кажется, должен работать.

И еще: плюс такого подхода не только в скорости работы, а и в скорости реализации (пригодится на олимпиадах) :)


◆ 0 Ответить  ...

o НЛО прилетело и опубликовало эту надпись здесь

o  **Bashuk** 1 авг 2011 в 22:30 ^

Вы правы. Но вот Вам еще одна идея: а что будет, если действительно таки использовать ориентированную площадь в вычислениях? Если я не ошибаюсь, то в Н-подобном случае и точки внутри сумма ориентированных площадей треугольников как раз будет равна ориентированной площади всего полигона. Но что и как будет происходить для точки за пределами, пока остается вопросом. Впрочем, это пока только догадки, и будет здорово, если у Вас есть ответ.

◆ 0 Ответить  ...

o  **EndUser** 2 авг 2011 в 02:12 ^

Ориентированная площадь из любой точки плоскости даст $\pm S$ многоугольника (в зависимости от направления обхода рёбер).

Более того, как считать площадь многоугольника, кроме как способом модуля от ориентированной площади? :-D

Ориентированная площадь для меня наиболее простой и очевидный способ при наличии координат вершин многоугольника.

EndUser 1 авг 2011 в 13:46

Да, надо сделать бесконечный луч в любую сторону из точки. Если количество пересечений нечётное, то точка внутри.

Если луч совпал с отрезком фигуры, тогда алгоритму задница.

+1 Ответить

pisarevsky 1 авг 2011 в 15:21

Я, может быть, немного троллю, но этот метод не работает, например, для множества бесконечно вложенных друг в друга плоских торов с радиусом от 0 до 1 :)

0 Ответить

EndUser 2 авг 2011 в 02:18

Это «многоугольник с дырками». в случае с
(o)
считается, что
o
содержит внешнее пространство, а пространство между
((
считается внутренним.

И к чему относится содержимое самой маленькой o вложенности этой маленькой o, если чётность-нечётность вы не знаете?

0 Ответить

monitorika 1 авг 2011 в 12:45

Принадлежность точки к выпуклой фигуре можно определить проще — точка должна находиться по одну и ту же сторону от каждого отрезка этой фигуры. В Вашем примере точка находится справа от каждого отрезка фигуры если смотреть обход фигуры по часовой стрелке.

Принадлежность к одной и той же стороне от отрезка легко посчитать через векторные произведения. Если все векторные произведения отрезков и вектора на исследуемую точку имеют один и тот же знак (положительны или отрицательны) то точка принадлежит фигуре, если они разные, то не принадлежит.

Тут всё элементарно просто, но если кому то нужно могу описать подробнее.

0 Ответить

hedgehog 1 авг 2011 в 17:11

xpro.ws/h.png

Тут лежит картинка, опровергающая ваш метод. Красным помечены отрезки, для которых точка лежит по другую сторону, в отличие от зеленых.

0 Ответить

monitorika 1 авг 2011 в 17:13

Я же в самом начале поста написал — Принадлежность точки к **выпуклой фигуре**

0 Ответить

hedgehog 1 авг 2011 в 17:17

Тогда прошу прощения.

0 Ответить

halyavin 1 авг 2011 в 12:50

Этот алгоритм не правильный — если исходная точка лежит на продолжении какой-нибудь стороны многоугольника, то произойдет деление на ноль с непредсказуемым результатом.

Правильно брать $\text{atan2}(y_{i+1}, x_{i+1}) - \text{atan2}(y_i, x_i)$ и нормировать до $(-\pi; \pi]$ (если результат близок к π , то точка лежит на стороне многоугольника). Еще нужно учесть совпадение какой-либо вершины многоугольника с нулем, если это допускается.

+5 Ответить

Aspos 1 авг 2011 в 15:04

Не. А также линия.

Задача сводится к определению пересечения двух отрезков. Ситуация, когда исходная точка лежит на продолжении стороны обрабатывается алгоритмом пересечения отрезков.

0 Ответить

Veterinar 1 авг 2011 в 12:55

Мне тоже попалась эта задача на олимпиаде. Тоже имел дело с турбо паскалем. Решил так:

В режиме graph рисуем зеленый треугольник, ставим красную точку. Далее заливаем треугольник белым. Проверяем цвета соседних с точкой пикселей. Если черные, значит точка не входит. Да, есть нюансы и вообще это мега читьство, но на олимпиаде главное чтобы программа правильно обрабатывала, а как никого не волнует.

+5 Ответить

Coolgun 1 авг 2011 в 12:58

Была такая мысль, но на олимпиадных машинах не bgi драйверов :)

0 Ответить

Veterinar 1 авг 2011 в 13:07

Безобразие)

+2 Ответить

Laplace 1 авг 2011 в 16:04

Мои одноклассники тоже так делали) И у них получалось гораздо быстрее, чем у меня реализовать аналитическое решение. Там правда задача была гораздо проще, с вычислением площади наложения прямоугольников.

0 Ответить

kashey 1 авг 2011 в 13:20

берем полигон из 1000 точек

для каждой грани находим ее MBR и записываем в некий двумерный массив $N \times M$ (100x100?) флаги присутствия граней в какой либо ячейке.

Далее имеем точку в X, Y или в nX, mY в пространстве наложенной на полигон сетке.

достаточно пустить луч в минимальную сторону(влево\право\верх\низ) и проверить попадание

через обычный алгоритм пересечения отрезка чтобы получить профит.

Время создания изначальной структуры данных по присутствию граней ака двумерный массив ~ в 10 раз дольше чем одиночная проверка.

Итоговый тест вхождения, при наличии структуры, работает примерно в $\sim N \times M$ раз быстрее чем одиночная проверка

0 Ответить

 **forgotten** 1 авг 2011 в 13:24

Вообще, для принадлежности точки полигону используют обычно метод подсчета числа пересечений. На каждой итерации данного метода выполняется: 2 сложения, 4 вычитания, 1 умножение, 1 деление и одна операция взятия знака числа.

В вашем алгоритме на каждую итерацию выполняется: 4 сложения, 2 вычитания, 8 умножений, 2 деления и 2 арктангенса. Где профит?

0 Ответить

 **Coolgun** 1 авг 2011 в 14:18 ^

Профита нет. Я не в коем разе не претендовал на написание самого крутого и быстрого алгоритма, упаси господи. В статье хотел лишь показать ход своих мыслей (уточню — был 96 год и до создания Википедии было так лет 5 :)). Спасибо за статью на Вики — там есть интересная ссылка на pdf-ку. И вот в этой самой статье, начало всех мат. выкладок очень похоже на мои потуги, что не может меня, конечно же, не радовать. Ну на следующие шаги моего серого вещества уже не хватило.

+1 Ответить

 **EndUser** 1 авг 2011 в 13:41

Я делал так: считал сумму углов из точки до концов отрезка, и так для каждого отрезка ломаной. Сам, блин, кустарно изобретал.

Результат был простой:

- если сумма равна $\pm 2\pi$, то точка внутри.
- если сумма равна 0, то точка снаружи.
- побочно: если сумма равна $+\pi$, то точка на границе (с оговорками на точность арифметики).


Дырка была в том случае, если концы незамкнутой ломаной лежат на луче, исходящем из точки. Но по условиям на входе был таки замкнутый многоугольник. ;-)

Позже проконсультировался с математиком, он предложил в общем случае интеграл по контуру (забыл какой функции), а в частном случае с ломаной таки сумму видимым углов обозрения, так как «изобрёл» я.

0 Ответить

НЛО прилетело и опубликовало эту надпись здесь

НЛО прилетело и опубликовало эту надпись здесь

 **EndUser** 1 авг 2011 в 23:07 ^

он работает для любых многоугольников, как выпуклых, так и невыпуклых.

исключения, я назвал, незамкнутые ломаные и точка на ребре (приходится доверять значению $\pi + \epsilon$).

угол вспомню чуть позже — с компа вместо мобилки.

но, вероятно вы правы: скорее через арккосинус скалярного.
тогда вычисления самозащищены от деления на ноль.

0 Ответить

 **EndUser** 2 авг 2011 в 02:02 ^

Так-с... Я прикинул три фрагмента (на псевдокоде)

Фрагмент 1, через арксинус векторного произведения:

```
//=====
//sin(a, b) = len(vector_prod(a, b)) / (len(a) * len(b))
float P0x, P0y; // origin of the angle
float P1x, P1y; // start of an angle segment
float P2x, P2y; // finish of an angle segment

float Ax = P1x - P0x, Ay = P1y - P0y; // convert segment (P0,P1) to (0,A)
float Bx = P2x - P0x, By = P2y - P0y; // convert segment (P0,P2) to (0,B)

if ( Ax == 0.0 && Ay == 0.0 ) // len_a == 0.0
throw Exception__Point_is_on_Vertex;

if ( Bx == 0.0 && By == 0.0 ) // len_b == 0.0
throw Exception__Point_is_on_Vertex;

float len_a = sqrt(Ax * Ax + Ay * Ay); // length of vector a
float len_b = sqrt(Bx * Bx + By * By); // length of vector b

float vector_prod_x = Ay * 0.0 - 0.0 * By; // Bz and Az are zero
float vector_prod_y = 0.0 * Bx - Ax * 0.0; // Az and Bz are zero
float vector_prod_z = Ax * By - Ay * Bx;
float vector_prod_len = vector_prod_z; // singular case, no sqrt() needed

{//bug here!!!
if ( vector_prod_len == 0.0 )
throw Exception__Point_is_on_Segment;
float sin_gamma = vector_prod_len / (len_a * len_b);
};//bugged block finished

float gamma = asin(sin_gamma);

return gamma;
// total:
// arcsine: 1
// square root: 2
// multiplication: 11
// division: 1
// addition: 2
// subtraction: 7
// test condition: 2
// conjunction: 2
// equality: 4
```

Фрагмент 2, через арккосинус скалярного произведения:

```
//cos(a, b) = scalar_prod(a, b) / (len(a) * len(b))
float P0x, P0y; // origin of the angle
float P1x, P1y; // start of an angle segment
float P2x, P2y; // finish of an angle segment

float Ax = P1x - P0x, Ay = P1y - P0y; // convert segment (P0,P1) to (0,A)
float Bx = P2x - P0x, By = P2y - P0y; // convert segment (P0,P2) to (0,B)
```

```

if ( Ax == 0.0 && Ay == 0.0 ) // len_a == 0.0
throw Exception__Point_is_on_Vortex;

if ( Bx == 0.0 && By == 0.0 ) // len_b == 0.0
throw Exception__Point_is_on_Vortex;

float len_a = sqrt(Ax * Ax + Ay * Ay); // length of vector a
float len_b = sqrt(Bx * Bx + By * By); // length of vector b

float scalar_prod = Ax * Bx + Ay * By;

float cos_gamma = cos_gamma = scalar_prod / (len_a * len_b);

float gamma = acos(cos_gamma);

return gamma;
// total:
// arccosine: 1
// square root: 2
// multiplication: 7
// division: 1
// addition: 3
// subtraction: 4
// test condition: 2
// conjunction: 2
// equality: 4

```

Фрагмент 3, через площадь треугольника детерминантом и площадь синусом:

```

//=====
// surface of a triangulum is equal to a half
// of the coordinated matrix determinant:
// |Ox, Oy, 1|
// S = (1/2) * |Ax, Ay, 1|
// |Bx, By, 1|
// S = (1/2) * (Ox * (Ay — By) + Ax * (By — Oy) + Bx * (Oy — Ay) =
// (considering Ox, Oy are zero)
// = Ax * By — Bx * Ay
//
// also
// surface of triangulum is
// S = (1/2) * |OA| * |OB| * sin_gamma
//
// therefore
// sin_gamma = 2 * S / (|OA| * |OB|) = determinant / (|OA| * |OB|)

float x0, y0; // point to test, point_0
float x1, y1; // segment start, point_1
float x2, y2; // segment finish, point_2

float Ax = P1x — P0x, Ay = P1y — P0y; // convert segment (P0,P1) to (0,A)
float Bx = P2x — P0x, By = P2y — P0y; // convert segment (P0,P2) to (0,B)

if ( Ax == 0.0 && Ay == 0.0 ) // len_a == 0.0
throw Exception__Point_is_on_Vortex;

if ( Bx == 0.0 && By == 0.0 ) // len_b == 0.0
throw Exception__Point_is_on_Vortex;

float len_a = sqrt(Ax * Ax + Ay * Ay); // length of vector a
float len_b = sqrt(Bx * Bx + By * By); // length of vector b

float determinant = Ax * By — Bx * Ay;

```

```

float sin_gamma;
// if the triangle is degenerate, determinant is zero, angle is zero,
// and point is on the segment

{//bug here!!!
if ( determinant == 0.0 )
throw Exception__Point_is_on_Segment;
float sin_gamma = determinant / (len_a * len_b);
};{//bugged block finished

float gamma = asin(sin_gamma);

return gamma;
// total:
// arcsine: 1
// square root: 2
// multiplication: 6
// division: 1
// addition: 2
// subtraction: 5
// test condition: 3
// conjunction: 2
// equality: 5

```

Первый фрагмент глюкавый (помечено).

Второй фрагмент на одно умножение сложнее третьего и

Третий фрагмент получается самым оптимальным, но опять глюк на синусе...


Ну, так и получается, что через арккосинус скалярного проще и вернее.

Кстати, если точка внутри фигуры, то я насчитал два случая $+2\pi$ и -2π .

И если точка на ребре, то тоже вижу два случая π и 3π .

А так вообще везде надо перепроверить операцию « $== 0.0$ » на корректность. Мало ли...

◆ 0 Ответить 📖 ...

○  **SBJoker** 1 авг 2011 в 14:42

Классический способ — создание горизонтального отрезка от точки до точки гарантированно лежащей вне контура, обычно очень далеко влево. После чего считаем число пересечений отрезка с гранями полигона.

Если число чётное — мы вне полигона, иначе — внутри.

◆ 0 Ответить 📖 ...

○  **RazB0YniK** 1 авг 2011 в 15:56 ^

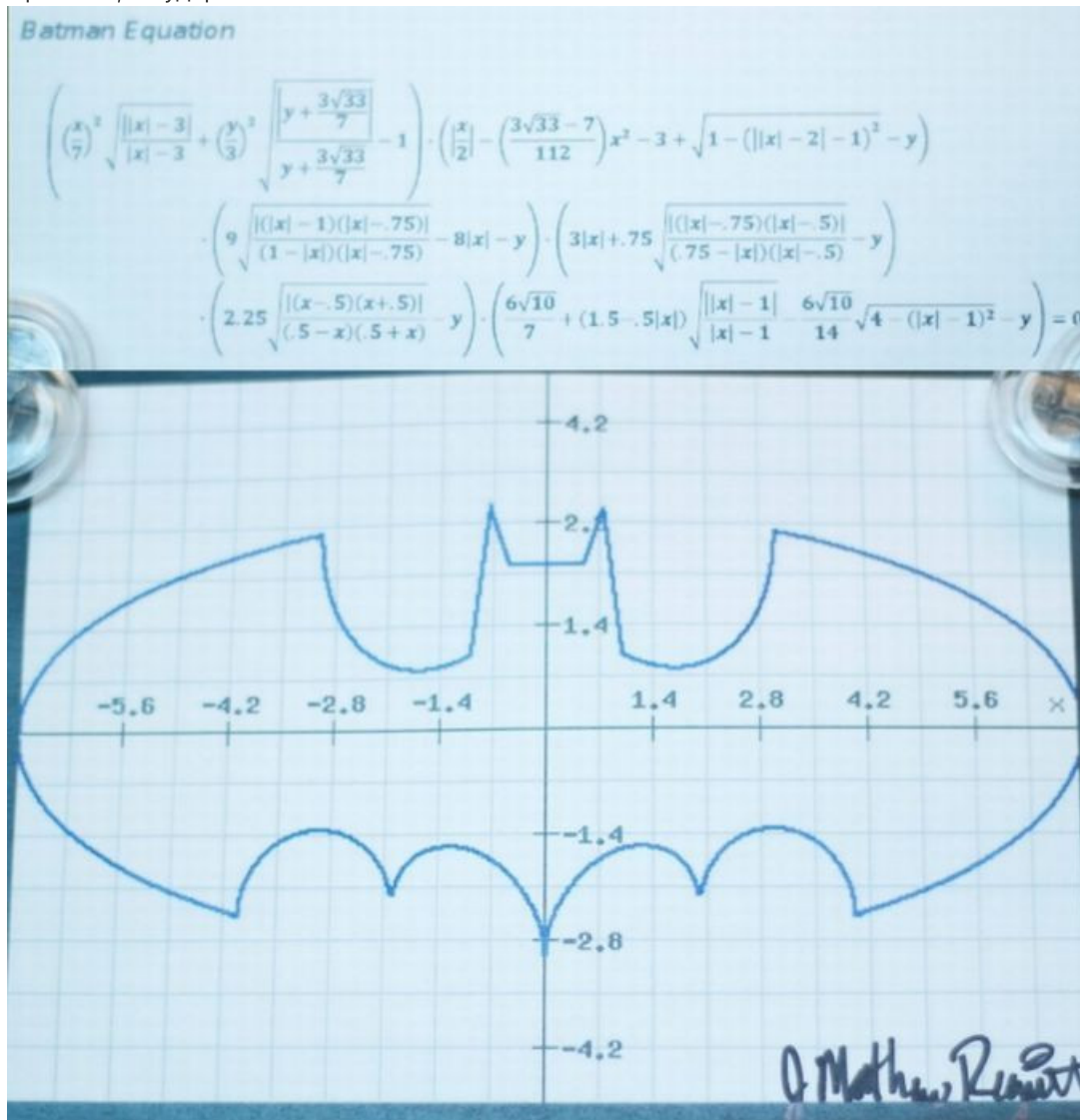
Если конечно среди «пересечений» нет касания.

◆ 0 Ответить 📖 ...


○ НЛО прилетело и опубликовало эту надпись здесь

○  **RazB0YniK** 1 авг 2011 в 16:29

Простите, не удержался.



◆ +12 Ответить

○  **ckald** 1 авг 2011 в 23:06 ^

Знаете, я в математике умею рисовать ромашку с абсолютно ассиметричными лепестками и ножкой :)

◆ 0 Ответить

○  **S_talker** 2 авг 2011 в 10:22 ^

Я тоже... Только в Paint'e :)

◆ +1 Ответить

○ НЛО прилетело и опубликовало эту надпись здесь

○  **freopen** 26 мая 2012 в 13:02 ^

Что Вы называете математическими задачами? Встречаются задачи, в которых требуется вывести одну формулу математическими методами и тупо считать для нее аргументы, но таких задач — одна из ста. Если Вас беспокоит, что многие алгоритмы и идеи имеют под собой немаленькую математическую базу, то тут тоже нечему удивляться, ведь теория алгоритмов — раздел математики. А на смекалку задачи бывают и на интуицию бывают и реализационные, в которых надо написать 800 строк кода просто сделав, что требуется.

◆ 0 Ответить

