

# Random points inside a parallelogram

Asked 16 years, 1 month ago    Modified 7 years, 3 months ago

Viewed 29k times



I have a 4 side convex Polygon defined by 4 points in 2D, and I want to be able to generate random points inside it.

49



If it really simplifies the problem, I can limit the polygon to a parallelogram, but a more general answer is preferred.



Generating random points until one is inside the polygon wouldn't work because it's really unpredictable the time it takes.



algorithm

matlab

random

2d

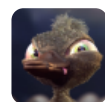
polygon

Share

Improve this question

Follow

edited Sep 5, 2017 at 3:07



Tipx

7,505 ● 4 ● 39 ● 61

asked Oct 27, 2008 at 17:40



Andres

1,895 ● 1 ● 19 ● 28

what do you mean by random? you can choose random points which are laying on the diagonals. Or do you want to

complete fill the entire polygon, if you produce enough random points? – [Peter Parker](#) Oct 27, 2008 at 17:49

If I produce enough I want to fill the entire polygon – [Andres](#) Oct 27, 2008 at 17:51

- 2 This couldn't be simpler: draw a plain rectangle that's just big enough to enclose your poly. (Or indeed, any "shape or thing" whatsoever.) Now create points that are randomly distributed in this enclosing plain square. For each one, test if it is within your shape. Discard those that are outside the shape. It's just that simple. Hope it helps! – [Fattie](#) Dec 25, 2011 at 22:03

## 11 Answers

Sorted by:

Highest score (default)



45

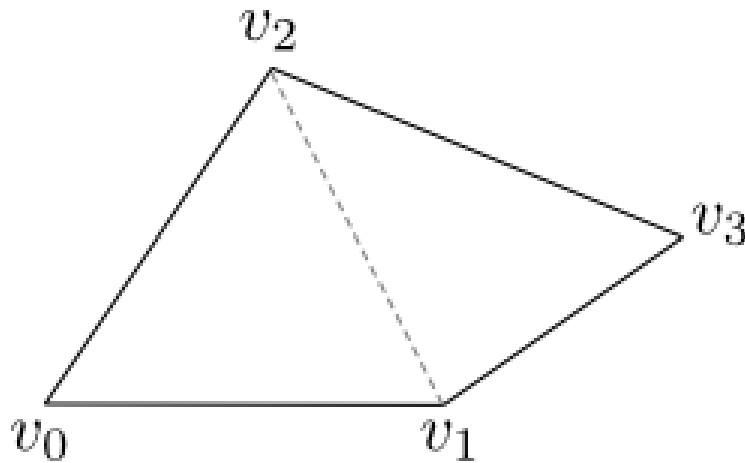


The question by the OP is a bit ambiguous so the question I will answer is: **How to generate a point from a uniform distribution within an arbitrary quadrilateral**, which is actually a generalization of **How to generate a point from a uniform distribution within an arbitrary (convex) polygon**. The answer is based on the case of generating a sample from a uniform distribution in a triangle (see <http://mathworld.wolfram.com/TrianglePointPicking.html>, which has a very nice explanation).

In order to accomplish this we:

1. Triangulate the polygon (i.e. generate a collection of non-overlapping triangular regions which cover the polygon). For the case of a quadrilateral, create an edge across any two non-adjacent vertices. For other polygons, see

[http://en.wikipedia.org/wiki/Polygon\\_triangulation](http://en.wikipedia.org/wiki/Polygon_triangulation) for a starting point, or <http://www.cgal.org/> if you just need a library.



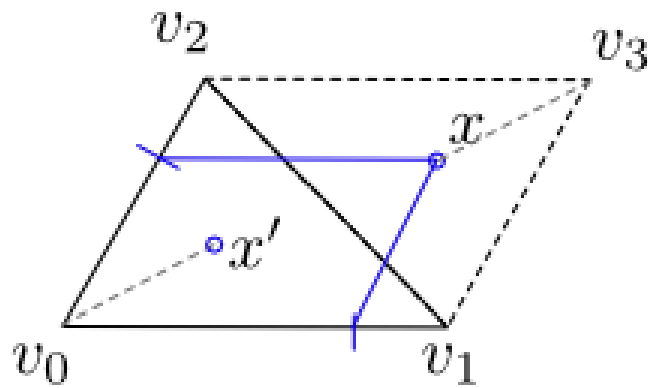
2. To pick one of the triangles randomly, let us assign an index to each triangle (i.e. 0,1,2,...). For the quadrilateral, they will be 0,1. For each triangle we assign a weight equal as follows:

$$weight_i = \frac{area_i}{\sum_j^n area_j}$$

3. Then generate a random index  $i$  from the finite distribution over indexes given their weights. For the quadrilateral, this is a Bernoulli distribution:

$$p(i) = \begin{cases} \frac{area_0}{area_0+area_1}, & i = 0 \\ \frac{area_1}{area_0+area_1}, & i = 1 \end{cases}$$

4. Let  $v_0, v_1, v_2$  be vertices of the triangle (represented by their point locations, so that  $v_0 = (x_0, y_0)$ , etc. Then we generate two random numbers  $a_0$  and  $a_1$ , both drawn uniformly from the interval  $[0,1]$ . Then we calculate the random point  $x$  by  $x = a_0 (v_1 - v_0) + a_1 (v_2 - v_0)$ .



$$v_3 = (v_1 - v_0) + (v_2 - v_0)$$

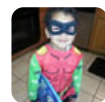
$$x' = v_0 + (x - v_3)$$

5. Note that with probability 0.5,  $x$  lies outside the triangle, however if it does, it lies inside the parallelogram composed of the union of the triangle with its image after a rotation of  $\pi$  around the midpoint of  $(v_1, v_2)$  (dashed lines in the image). In that case, we can generate a new point  $x' = v_0 + R(\pi)(x - v_3)$ , where  $R(\pi)$  is a rotation by  $\pi$  (180 deg). The point  $x'$  will be inside the triangle.
6. Further note that, if the quadrilateral was already a parallelogram, then we do not have to pick a triangle at random, we can pick either one deterministically, and then choose the point  $x$  without testing that it is inside its source triangle.

Share Improve this answer

Follow

edited Jun 1, 2016 at 15:32



gariepy

3,664 ● 6 ● 23 ● 34

answered Dec 7, 2011 at 18:18




cheshirekow

4,897 ● 6 ● 45 ● 48

---

1 Great answer. Lovely pictures. – [Thomas Ahle](#) Dec 7, 2011 at 23:15

---

1 I'm trying to implement this and I think it should be  $x' = v_0 + (v_3 - x)$  Am I totally off base? Looking at it some more I'm not sure I'm right but my test case of  $v_0 = [0,0]$  puts  $x'$  outside of the triangle. – [Gabriel Littman](#) Jun 23, 2014 at 22:37 

---

@gabriel\_littman. I believe you are correct. In the graphic for the equation there is a missing  $R(\pi)$ , which is present in the text... i.e. rotation by 180 degrees. I think that rotation matrix is  $[-1, 0; 0, -1]$  which is to say that we take the negative of its operand. – [cheshirekow](#) Jun 24, 2014 at 14:17

---

This is the actual answer to the question! – [Gustavo Maciel](#) Jul 23, 2014 at 1:45

---

I've tried implementing this in python but I think something is broken. See [gist.github.com/astromme/599de466236adc534bc6e33cf2af8e7b](https://gist.github.com/astromme/599de466236adc534bc6e33cf2af8e7b). For a triangle with points  $[0, 1]$ ,  $[1, 0]$ ,  $[1,0]$   $v_3$  is  $[2, -1]$  which doesn't seem to make sense. Furthermore, I get points that are outside of the quad. Any ideas? – [Andrew Stromme](#) Jun 16, 2017 at 1:06

---



A. If you can restrict your input to parallelogram, this is really simple:

30



1. Take two random numbers between 0 and 1. We'll call them  $u$  and  $v$ .

2. If your parallelogram is defined by the points ABCD such that AB, BC, CD and DA are the sides, then take your point as being:





$$p = A + (u * AB) + (v * AD)$$



Where  $AB$  is the vector from A to B and  $AD$  the vector from A to D.

B. Now, if you cannot, you can still use the barycentric coordinates. The barycentric coordinates correspond, for a quad, to 4 coordinates  $(a, b, c, d)$  such that  $a+b+c+d=1$ . Then, any point  $P$  within the quad can be described by a 4-uple such that:

$$P = a A + b B + c C + d D$$

In your case, you can draw 4 random numbers and normalize them so that they add up to 1. That will give you a point. Note that the distribution of points will NOT be uniform in that case.

C. You can also, as proposed elsewhere, decompose the quad into two triangles and use the half-parallelogram method (i.e., as the parallelogram but you add the condition  $u+v=1$ ) or the barycentric coordinates for triangles. However, if you want uniform distribution, the probability of having a point in one of the triangle must be equal to the area of the triangle divided by the area of the quad.

Share Improve this answer

edited Aug 27, 2017 at 14:44

Follow



Niyaz

54.8k ● 56 ● 152 ● 183

answered Oct 27, 2008 at 17:43



PierreBdR

43.2k ● 10 ● 48 ● 66

---

Whether barycenter approach will work for the case of polygons with holes? – [Pranav](#) Jul 14, 2016 at 9:16

---

@Pranav No it won't ... barycentric coordinate requires continuous domain, and I would guess probably convex (to be checked). – [PierreBdR](#) Jul 15, 2016 at 0:34

---



18

Assuming you want a uniform distribution: Form two triangles from your polygon. Pick which triangle to generate the point in according to their area ratio.



Call the corners of the triangle A, B, C, the side vectors AB, BC, AC and generate two random numbers in  $[0,1]$  called u and v. Let  $p = u * AB + v * AC$ .



If  $A+p$  is inside the triangle, return  $A+p$

If  $A+p$  is outside the triangle, return  $A + AB + AC - p$

(This is basically PierreBdR's formula except for the preprocessing and the last step that folds the point back into a triangle, so it can handle other shapes than parallelograms).

Share Improve this answer

Follow

edited Aug 27, 2017 at 16:32



Niyaz

54.8k ● 56 ● 152 ● 183

answered Oct 27, 2008 at 18:09



jakber

3,559 ● 22 ● 20

---

For anyone else looking, here is how to find if a point is inside a triangle: [stackoverflow.com/questions/2049582/...](https://stackoverflow.com/questions/2049582/) – Niyaz  
Aug 27, 2017 at 16:55

---



Your polygon is two triangles, so why not randomly select one of those, then find a random point in the triangle.

4

Probably not the best solution, but it'd work.



Share Improve this answer

answered Oct 27, 2008 at 17:46



Follow



jonni

28.3k ● 8 ● 82 ● 110



---

3 If you need a uniform distribution for the random points, make sure you take into account the area of each of the two triangles and weight appropriately. – Robert Gamble Oct 27, 2008 at 18:07

---



A somewhat less "[naïve](#)" approach would be to use a [polygon fill algorithm](#), and then select points from the fill lines randomly.

3



## C Code Sample



```
// public-domain code by Darel Rex Finley, 2007
```



```

int  nodes, nodeX[MAX_POLY_CORNERS], pixelX, pixelY, i

// Loop through the rows of the image.
for (pixelY=IMAGE_TOP; pixelY<IMAGE_BOT; pixelY++) {

    // Build a list of nodes.
    nodes=0; j=polyCorners-1;
    for (i=0; i<polyCorners; i++) {
        if (polyY[i]<(double) pixelY && polyY[j]>=(double)
            || polyY[j]<(double) pixelY && polyY[i]>=(double)
            nodeX[nodes++]=(int) (polyX[i]+(pixelY-polyY[i])
                *(polyX[j]-polyX[i])); }
        j=i; }

    // Sort the nodes, via a simple "Bubble" sort.
    i=0;
    while (i<nodes-1) {
        if (nodeX[i]>nodeX[i+1]) {
            swap=nodeX[i]; nodeX[i]=nodeX[i+1]; nodeX[i+1]=s
        else {
            i++; }}

    // Fill the pixels between node pairs.
    // Code modified by SoloBold 27 Oct 2008
    // The flagPixel method below will flag a pixel as
    for (i=0; i<nodes; i+=2) {
        if (nodeX[i ]>=IMAGE_RIGHT) break;
        if (nodeX[i+1]> IMAGE_LEFT ) {
            if (nodeX[i ]< IMAGE_LEFT ) nodeX[i ]=IMAGE_LE
            if (nodeX[i+1]> IMAGE_RIGHT) nodeX[i+1]=IMAGE_RI
            for (j=nodeX[i]; j<nodeX[i+1]; j++) flagPixel(j,

        // TODO pick a flagged pixel randomly and fill it,
        list.
        // Repeat until no flagged pixels remain.

```

Share Improve this answer

edited Oct 27, 2008 at 17:57

Follow

answered Oct 27, 2008 at 17:44

wpri

25.4k ● 11 ● 57 ● 70

- 
- 1 I suspect this is not what Turambar needs, but it will work. Some lines are longer than others, so to get a uniform distribution, don't pick a line, then pick a pixel. Count the pixels, then choose one randomly, and find its location from the list... – [dmckee](#) --- [ex-moderator kitten](#) Oct 27, 2008 at 18:00
- 



2



By "general" do you mean all non-parallelogram 4-side polygons in general or all possible polygons?

How about drawing a random line connecting the 4 sides e.g. If you have this:

```
.BBBB.  
A      C  
A      C  
.DDDD.
```

Then generate a random point on a unit square, then mark the point on the line B and D at the percentage of distance on the X axis. Do the same on line A and C using value from the Y axis.

Then connect the point on line A to line C and line B to line D, the intersection point is then used as the random point.

It's not uniform because rounding errors will aid certain points but it should be close if you are working with

floating points values.

Implementation should be rather easy, too, since you are already working with polygons. You should already have code that does those simple tasks.

Here's a quick pseudocode:

```
void GetRandomPoint(Polygon p, ref float x, ref float  
  
    float xrand = random();  
    float yrand = random();  
  
    float h0 = p.Vertices[0] + xrand * p.Vertices[1];  
    float h1 = p.Vertices[2] + yrand * p.Vertices[3];  
  
    float v0 = p.Vertices[0] + xrand * p.Vertices[2];  
    float v1 = p.Vertices[1] + yrand * p.Vertices[3];  
  
    GetLineIntersection(h0, h1, v0, v1, x, y);  
  
}
```

Share Improve this answer

edited Oct 27, 2008 at 18:22

Follow

answered Oct 27, 2008 at 18:09



chakrit

61.5k ● 25 ● 136 ● 163



2

This works for general, convex quadrilaterals:

You can borrow some concepts from the Finite Element Method, specifically for quadrilateral (4-sided) elements



([refer to section 16.5 here](#)). Basically, there is a bilinear parameterization that maps a square in  $u$ - $v$  space (for  $u, v \in [-1, 1]$  in this case) to your quadrilateral that consists of points  $p_i$  (for  $i = 1, 2, 3, 4$ ). Note that In the provided reference, the parameters are called  $\eta$  and  $\xi$ .

Basic recipe:

1. Choose a suitable random number generator to generate well-distributed points in a square 2D domain
2. Generate random  $u$ - $v$  pairs in the range  $[-1, 1]$
3. For each  $u$ - $v$  pair, the corresponding random point in your quad =  $\frac{1}{4} * ((1-u)(1-v) * p_1 + (1+u)(1-v) * p_2 + (1+u)(1+v) * p_3 + (1-u)(1+v) * p_4)$

The only problem is that uniformly distributed points in the  $u$ - $v$  space won't produce uniformly distributed points in your quad (in the Euclidean sense). If that is important, you can work directly in 2D within the bounding box of the quad and write a point-in-quad (maybe by splitting the problem into two point in tris) test to cull random points that are outside.

Share Improve this answer

answered Jan 14, 2011 at 3:20

Follow



Not Sure

131 ● 1



Do the points need to be uniformly distributed, or is any distribution ok?

1

Can the polygon be concave, or is it guaranteed to be convex?



If the answer to both the above is no, then pick any two of the vertices and pick a random point on the line segment between them. This is limited to the line segments connecting the vertices (ie, VERY non-uniform); you can do a bit better by picking a third vertex and then picking a point between that and the first point -- still non-uniform, but at least any point in the polygon is possible

Picking a random point on a line between two points is easy, just  $A + p(B-A)$ , where  $A$  and  $B$  are the points and  $p$  is a random number between 0.0 and 1.0

Share Improve this answer

answered Oct 27, 2008 at 17:54

Follow



Chris Dodd

2,950 ● 15 ● 10



1



What kind of distribution do you want the points to have? If you don't care, the above methods will work fine. If you want a uniform distribution, the following procedure will work: Divide the polygon into two triangles,  $a$  and  $b$ . Let  $A(a)$  and  $A(b)$  be their areas. Sample a point  $p$  from the uniform distribution on the interval between 0 and  $A(a)+A(b)$ . If  $p < A(a)$ , choose triangle  $a$ . Otherwise, choose triangle  $b$ . Choose a vertex  $v$  of the chosen triangle, and let  $c$  and  $d$  be the vectors corresponding to the sides of the triangle. Sample two numbers  $x$  and  $y$  from the exponential distribution with unit average. Then

the point  $(xc+yd)/(x+y)$  is a sample from the uniform distribution on the polygon.

Share Improve this answer

answered Oct 27, 2008 at 18:08

Follow



Alex Coventry

70.7k ● 5 ● 39 ● 40



1



The MATLAB function [cprnd](#) generates points from the uniform distribution on a general convex polytope. For your question a more specialized algorithm based on decomposing the quadrilateral into triangles is more efficient.



Share Improve this answer

edited Oct 27, 2012 at 6:07



Follow



Peter O.

32.8k ● 14 ● 84 ● 97

answered Jan 7, 2012 at 15:50



Tim Benham

11 ● 1



0



For PostGIS, this is what I am using (you might want a ward for possible infinite loops). You might export the algorithm to your programming language:



```
CREATE or replace FUNCTION random_point(geometry)
RETURNS geometry
AS $$
DECLARE
    env geometry;
    corner1 geometry;
    corner2 geometry;
```

```

minx real;
miny real;
maxx real;
maxy real;
x real;
y real;
ret geometry;
begin

select ST_Envelope($1) into env;
select ST_PointN(ST_ExteriorRing(env),1) into corner1;
select ST_PointN(ST_ExteriorRing(env),3) into corner2;
select st_x(corner1) into minx;
select st_x(corner2) into maxx;
select st_y(corner1) into miny;
select st_y(corner2) into maxy;
loop
    select minx+random()*(maxx-minx) into x;
    select miny+random()*(maxy-miny) into y;
    select ST_SetSRID(st_point(x,y), st_srid($1)) into
    if ST_Contains($1,ret) then
        return ret ;
    end if;
end loop;
end;
$$
LANGUAGE plpgsql
volatile
RETURNS NULL ON NULL INPUT;

```

Share Improve this answer

edited Apr 13, 2011 at 12:25

Follow

community wiki

2 revs

rpto



**Highly active question.** Earn 10 reputation (not counting the [association bonus](#)) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.