UNIVERSITATEA DIN BUCURESTI

Generarea unor puncte uniform distribuite intr-un poligon neregulat

de

Stefan Niculae

Referat realizat pentru
Didactica Specialitatii

in
Facultatea de Matematica si Informatica
Departamentul de Informatica

Mai 2015

“Nimic nu este intamplator. Chiar daca pare asa, este doar pentru ca nu ii cunosti cauzele.”

Johnny Rich, specialist in educaie

UNIVERSITATEA DIN BUCURESTI

Facultatea de Matematica si Informatica
Departamentul de Informatica

Absolvent Didactica Specialitatii

de Stefan Niculae

Abstract

In aceasta lucrare aratam un algoritm eficient de generare a unui numar de puncte in interiorul unui poligon neregulat. Constructia algoritmului este una incrementala, pornind de la cel mai simplu caz, unidimensional, trecand prin cazul dreptunghiului, poligonului convex si in ultimul capitol ajungand la cel concav....

Cuprins

Capitolul 1

Introducere

In cadrul acestui capitol vom vorbi despre motivatia alegeri temei si vom detalia procesul de construire al algoritmului.

1.1 Motivatie

In cadrul cursului de Metode de Dezvoltare Software am ales ca proiect de semestru dezvoltarea unui joc video. Este vorba de un joc cu tancuri cu grafica bidimensionala. In cadrul acestuia exista elemente ale mediului distructibile (butoaie explozive) cat si indistructibile, de decor (copaci).

Elementele mediului pot fi asezate in orice pozitie pe harta, insa plasarea manuala a multor astfel de elemente este obositoare. Astfel am ajuns la aceasta problema: cum pot genera in mod automat un numar de copaci intr-o maniera cat mai uniforma intr-un poligon definit.

1.2 Constructia algoritmului

Vom incepe detalierea algoritmului plecand de la cazuri simple, crescand pe parcursul capitolelor in dificultate. Rezolvand mai intai cazuri mai putin generale insa mai simple si ajungand in final la situatia cea mai generala.

In primul capitol vom trata un caz foarte simlplu, cel unidimensional.

Al doilea capitol va fi dedicat unuei probleme banale - distributia uniforma a punctelor intr-un dreptunghi.

In cel de-al treilea capitol ne vom ocupa de generarea in interiorul unui poligon regulat.

In cel de-al patrulea capitol vom atinge esenta lucrarii - generarea intr-un poligon neregulat.

1.3 Despre Random

Pentru a putea simula o distributie uniforma ne vom folosi de functiile oferite de limbaj pentru generarea de numere aleatoare. Cu toate ca aceste functii nu sunt cu adevarat aleatoare, aceasta restrictie nu poate fi depasita din cauza insusi definitiei algoritmului, o succesiune finita de instructiuni ce produce un output care va fi mereu acelasi dat acelasi input.

Generarea numerelor aleatoare aparent nu primeste niciun argument in momentul apelarii, insa argumentul este ascuns in modul de constructie al rezultatului. Numarul aleator returnat este defapt rezultatul aplicarii unui numar de operatii asupra unui seed si a unor constante alese special datorita bunelor proprietati combinatorice. Acest seed este un numar foarte mare, implicit numarul de secunde trecute de la 1 ianuarie 1970. De aceea numerele generate par aleatoare: intre orice doua apeluri ale functiei cu siguranta trece o anumita perioada de timp deci seedul folosit in generare va fi diferit. [Com15]

In continuare vom considera pseudo-random number generator-ul ca un generator de numere aleatoare.

Capitolul 2

Cazul unidimensional

Cea mai simpla instanta a problemei este cea unidimensionala. Aceasta se reduce pur si simplu la generarea unui numar in intervalul dat.

2.1 Functia Random

In contiunare, functia Random folosita in pseudocod este functia care accepta doua argumente si genereaza un numar aleator in intervalul inchis definit de argumente.

Algorithm 1 Generare punct aleator pe dreapta cu capetele [a, b]

function GenereazaPeDreapta(a, b)
  return $Random(a, b)$

Capitolul 3

Cazul dreptunghiului

Acest caz extinde in spatiu bidimensional algoritmul din capitolul anterior, generand doua numere, pentru coordonatele x si y.

In continuare, coordonatele sunt considerate a fi intr-un sistem de coordonate xOy cu x crescand de la stanga la dreapta si y crescand de jos in sus.

Consideram Punctul A coltul din stanga jos al dreptunghiului si Punctul B coltul din dreapta sus.

Algorithm 2 Generare punct aleator in dreptunghiul cu colturile A, B

---

function GENEREAZAINDREPTUNGHI(A, B)
  $x = Random(A.x, B.x)$
  $y = Random(A.y, B.y)$
  return $Point(x, y)$

Capitolul 4

Cazul poligonului convex

In acest caz vom proceda foarte similar cu cel anterior insa cu o modificare esentiala: vom returna punctul generat numai daca acesta se afla in interiorul poligonului. In caz contrar, generam un alt astfel de punct.

Consideram ca n-gonul este retinut in memorie sub forma unei liste cu n elemente. Generarea bounding box-ului presupune in aflarea minimului si maximului atat coordonatei x cat si y prin parcurgerea fiecarui colt al poligonului.

Algorithm 3 Generare punct aleator in interiorul poligonul P

function GenereazaInPoligonRegulat(P)
  $BB = BoundingBox(P)$
  $X = GenereazaInDreptunghi(BB)$
  while $P \quad nu \quad contine \quad X$ **do**
    $X = GenereazaInDreptunghi(BB)$

  return X

4.1 Apartenenta unui punct la un poligon regulat

Un punct apartine unui poligon regulat daca acesta se afla de aceeasi parte a fiecarei dintre muchiile acestuia. Orientarea unui punct r fata de o muchie pq este data de semnul determinantului ariei triunghiului pqr. [MdBO08]

Formula ariei:

$$A(\triangle pqr) = \frac{1}{2} \begin{vmatrix} p_x & q_x & r_x \\ p_y & q_y & r_y \\ 1 & 1 & 1 \end{vmatrix}$$

Interpretarea rezultatului:

$$r \quad fata \quad de \quad pq \quad este \quad \begin{cases} la \quad stanga, & A(\triangle pqr) > 0 \\ coliniar, & A(\triangle pqr) = 0 \\ la \quad dreapta, & A(\triangle pqr) < 0 \end{cases} \tag{4.1}$$

Implementarea:

Algorithm 4 Testarea pozitiei unui punct X fata de un poligon convex P

function ORIENTARE(P, Q, R)
  $val = (Q.y - P.y) * (R.x - Q.x) - (Q.x - P.x) * (R.y - Q.y)$
  return $signum(val)$

function CONTINE(P, X)
  $parte = Orientare(P_0, P_1, P_2)$
  for $i \leftarrow 2, P.n$ **do**
    if $Orientare(P_{i-1}, P_i, punct) \neq parte$ **then**
      return $false$

  return $true$

4.2 Terminarea algoritmului

In teorie algoritmul pare sa admita si apeluri in care sa ramane blocate in bucla de re-generare a numerelor si astfel sa nu se termine.

In practica insa, aceasta problema nu este una reala. Generatorul de numere aleatoare alegand coordonate destul de departate unul de celalalt astfel incat problema neterminarii algoritmului sa devina neexistenta.

Capitolul 5

Cazul poligonului concav

Cazul poligonului concav este asemanator cu cel al poligonului convex, singura schimbare fiind modul in care se testeaza pozitia unui punct fata de poligon.

5.1 Numarul de bobinari

Un algoritm foarte intuitiv pentru testarea pozitiei unui punct fata de un poligon concav este cel al numarului de bobinari.

Definim numarul de bobinari al unui punct fata de un poligon ca fiind numarul de rotiri in jurul axei pe care le face punctul daca ne imaginam ca acesta se 'uita' pe rand la fiecare colt al poligonului. Pentru rotirea in sens trigonometric vom adauga 1 la numarul de bobinari iar pentru rotirea in sens invers trigonometric, vom scadea 1. [Sun15]

Algorithm 5 Testarea pozitiei unui punct X fata de un poligon concav P

function CONTINE(P, X)
    $wn = 0$
    for $i \leftarrow 0, P.n$ **do**
        if $P_i.y \leq X.y$ **then**
            if $P_{i+1}.y > X.y$ **then**
                if $Orientare(P_i, P_{i+1}, X) > 0$ **then**
                    $wn = wn + 1$

        else
            if $P_{i+1}.y \leq X.y$ **then**
                if $Orientare(P_i, P_{i+1}, X) < 0$ **then**
                    $wn = wn - 1$

    return $wn \neq 0$

Bibliografie

[Com15] Community. *Random Number Generation.* May 2015. http://en.wikipedia.org/wiki/Random-number-generation.

[MdBO08] Marc van Kreveld Mark de Berg, Otfried Cheong and Mark Overmars. *Computational Geometry Algorithms and Applications.* Springer, 3 edition, 2008.

[Sun15] Dan Sunday. *Inclusion of a Point in a Polygon.* May 2015. http://geomalgorithms.com/a03-inclusion.html.