Diagrama Voronoi

Galu Darius Alexandru

Anul 2, gr.2

1. O diagramă Voronoi este o diagramă ce constă dintr-un număr de arii(puncte). Fiecare arie Voronoi are de asemenea o celula Voronoi care cuprinde toate punctele cel mai apropiate de s. Sarcina este de a demonstra cum se generează și se afișează o diagramă Voroni.

În matematică, o diagramă Voronoi este o împărțire a unui plan în regiuni bazate pe distanța față de punctele dintr-un subset specific al planului.

Utilizarea informală a diagramelor Voronoi poate fi trasată înapoi în Descartes în 1644. Peter Gustav Lejeune Dirichlet a folosit diagrame Voronoi bidimensionale și tridimensionale în studiul formelor patrate în 1850. Medicul britanic John Snow a folosit o diagramă Voronoi în 1854 pentru a ilustra modul în care majoritatea persoanelor care au murit în epidemia de holeră Broad Street a trăit mai aproape de pompa infectată Broad Street decât cu orice altă pompă de apă.

Diagramele Voronoi sunt numite după matematicianul rus Georgy Fedosievych Voronoy care a definit și a studiat cazul general n-dimensional din 1908. Diagramele Voronoi utilizate în geofizică și meteorologie pentru a analiza datele distribuite spațial (cum ar fi măsurătorile precipitațiilor) se numesc poligoane Thiessen după meteorologul american Alfred H. Thiessen. În fizica materiei condensate, astfel de tessellations sunt, de asemenea, cunoscute ca celule unitate Wigner-Seitz. Teselările Voronoi ale latticei reciproce a momentei sunt numite zone Brillouin. Pentru laturile generale din grupurile Lie, celulele sunt pur și simplu numite domenii fundamentale. În cazul spațiilor metrice generale, celulele sunt adesea numite poligoane fundamentale metrice.

Cel mai optim algoritm pentru diagrama Voronoi este algoritmul lui Fortune, este un algoritm de linie de curbare pentru generarea unei diagrame Voronoi dintr-un set de puncte într-un plan folosind timpul O (n log n) și spațiul O (n). A fost inițial publicată de Steven Fortune în 1986 în lucrarea "Un algoritm de curbură pentru diagramele Voronoi.

Algoritmul menține atât o linie de curățare, cât și o linie de plajă, care se deplasează prin plan, pe măsură ce progresează algoritmul. Linia de ștergere este o linie dreaptă, pe care, prin convenție, o presupunem că este verticală și se mișcă spre stânga spre dreapta în plan. În orice moment în timpul algoritmului, punctele de intrare din stânga liniei de curățare vor fi încorporate în diagrama Voronoi, în timp ce punctele din dreapta liniei de ștergere nu vor fi fost încă luate în considerare. Linia de pe plajă nu este o linie dreaptă, ci o curbă complicată, în formă de piesă, la stânga liniei de curățare, compusă din bucăți de parabole; ea divide porțiunea planului în interiorul căruia poate fi cunoscută diagrama Voronoi, indiferent de ce alte puncte ar putea fi potrivite liniei de maturare, din restul planului. Pentru fiecare punct din stânga liniei de maturare, se poate defini o parabolă de puncte echidistantă de la acel punct și de la linia de maturare; linia de plajă este granița unirii acestor parabole. Pe măsură ce linia de măturat progresează, vârfurile liniei de pe plajă, la care se traversează două parabole, trasează marginile diagramei Voronoi. Linia de plajă progresează prin păstrarea fiecărei baze parabolice exact la jumătatea distanței dintre punctele inițial preluate cu linia de maturare și noua poziție a liniei de curățare. Din punct de vedere matematic, aceasta înseamnă că fiecare parabolă este formată folosind linia de maturare drept directrix și punctul de intrare ca focalizare.

**De asemenea este un algoritm greu de implementat și se folosește ca admitere la Oxford. ”So if you can make it, you are a beast”.**

2. Pentru design al diagramei s-a folosit librăria jFrame cu care am afișat punctele, cu buffered image s-au introdus datele aceasta funcție este o subclasă în care se introduc date, iar cu “CreateGraphics2D” s-a desenat în “BufferedImage”.vezi Fig.1

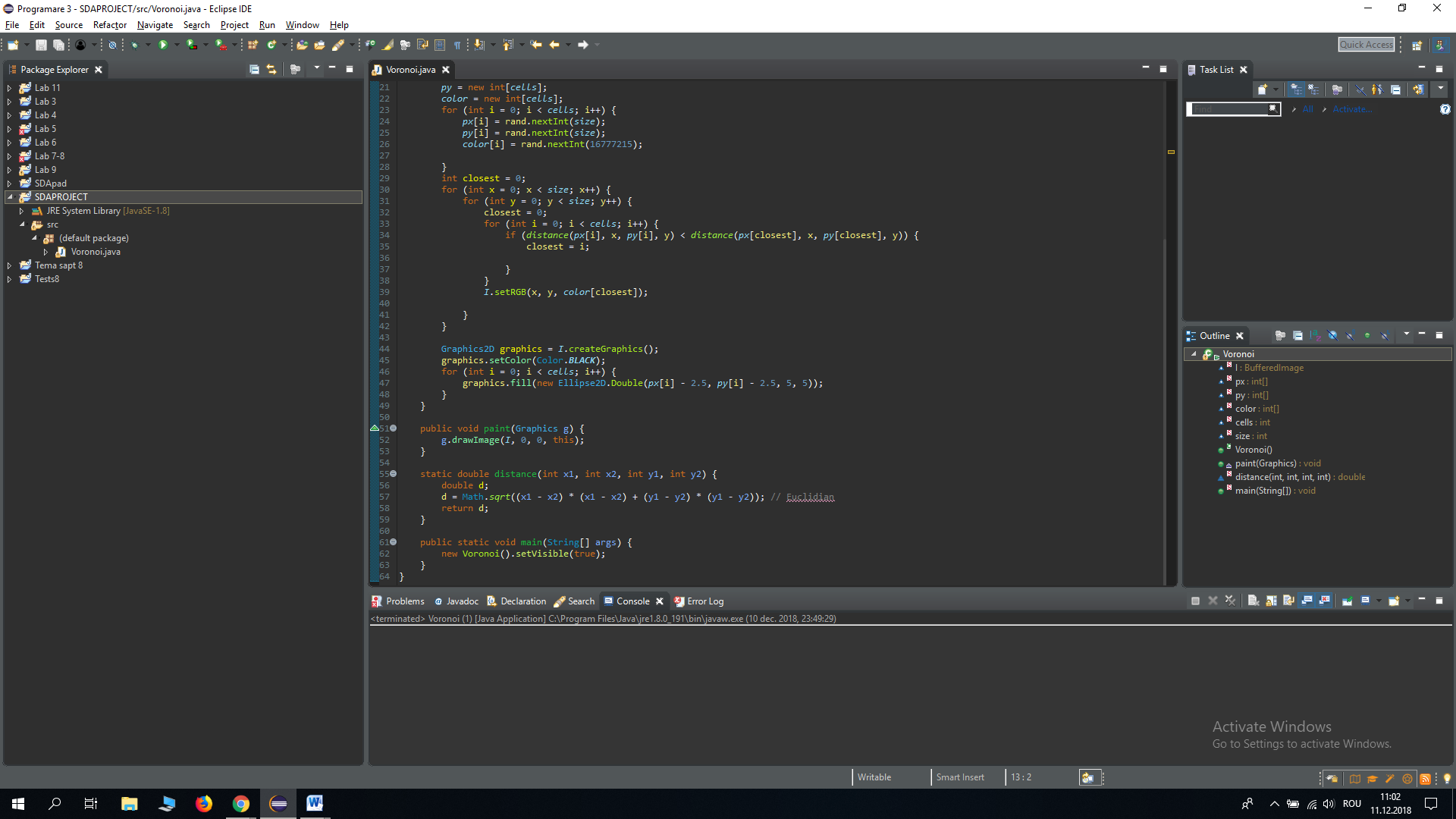


Fig.1

|  |
| --- |
| Voronoi |
| +BufferedImage(): I  +px[]:int  +py[]:int  +color[]:int  +cells:int  +size: int |
| +paint():void  +distance():double |

3. Codul este implementat în Java și conține o singură clasă ce extinde librăria JFrame. Această librările afișează punctele în plan, folosim un buffered image în care se introduc datele, un set bounds adică locul și dimensiunile ferestrei Jframe, acesta fiind realizarea prototipului.vezi Fig.2



Fig.2

4. În ceea ce privesc bug-urile am întâmpinat probleme la formula cu care se calculează distanța euclidiană deoarece am greșit modul de implementare, bineînțeles acea problemă s-a remediat.vezi Fig.3

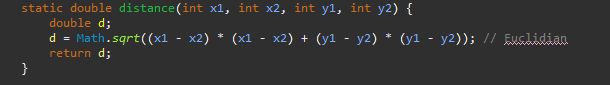


Fig.3

5 Având în vedere documentația codului, s-a început cu crearea clasei Voronoi de unde am extins o librărie Jframe care afișează puncte

* un buffered image care este o subclasă și în aceasta se introduc

datele

* am creat un px[] și py[] ce reprezintă 2 puncte în plan
* un color[] pentru colorarea imaginii în funcție de cel mai apropiat punct
* un cells ce reprezintă numărul de puncte date în plan
* size reprezintă mărimea planului mai precis fereastra
* avem constructorul Voronoi unde am moștenit jFrame pe care l-am denumit “Voronoi Diagram”
* setBounds reprezintă locul și dimensiunea unde se afișează pe ecran diagrama
* setDefaultClose... adică se închide procesul în momentul în care dai x diagramei
* apoi avem un random cu care spawn-ează random punctele si culorile în plan vezi Fig.4

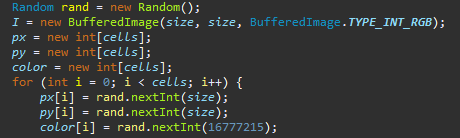


Fig.4

* closest este indicele celui mai apropiat punct și de la linia 28-41 se schimă culoarea în funcție de cel mai apropiat punct și fiecărui pixel îi este dată o culoare în funcție de cel mai apropiat punct vezi Fig.5

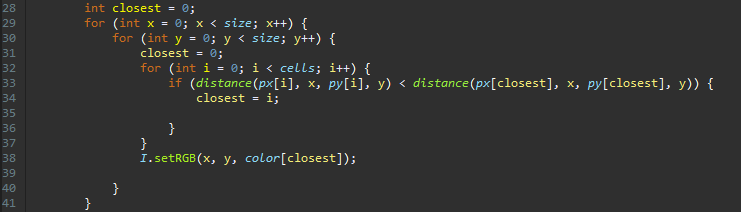


Fig.5

* Graphics2D este folosit pentru a desena în Buffered Image vezi Fig.6

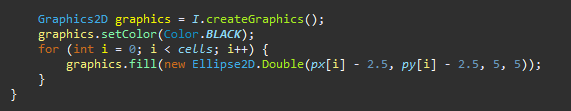


Fig.6

* funcție paint graphics ce desenează imaginea vezi Fig.7



Fig.7

* funcție distance ce calculează distanța euclidiană dintre două puncte x1 y1 x2 y2 vezi Fig.8

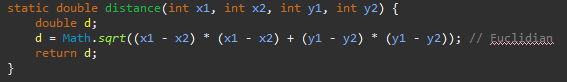


Fig.8

* În ultima parte am apelat într-un main Voronoi-ul și cu un boolean true pentru afișarea ferestrei vezi Fig.9

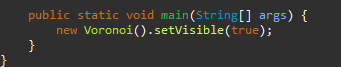


Fig.9