Diagrama de Voronoi em Imagens

Gabriel Eugenio Brito
Renan de Souza Antunes

Diagrama de Voronoi

- Também chamado de Tesselação de Voronoi ou Decomposição de Voronoi

- É um particionamento de um plano em regiões baseado na distância até um conjunto de pontos escolhidos previamente (também chamados de *seeds*, *generators* ou *sites*).

- Cada semente possui uma região correspondente (célula) composta de pontos que estão mais próximos dela do que de qualquer outra semente.

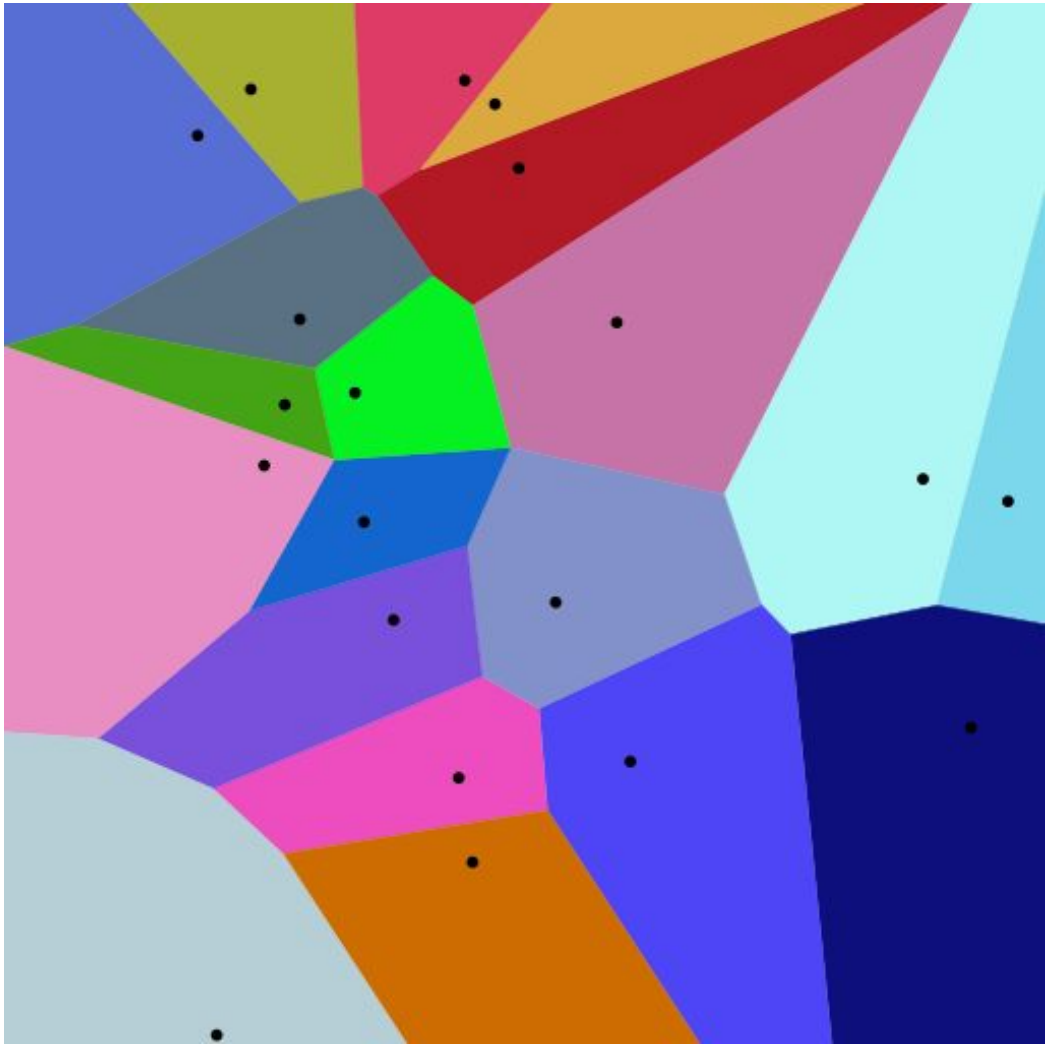
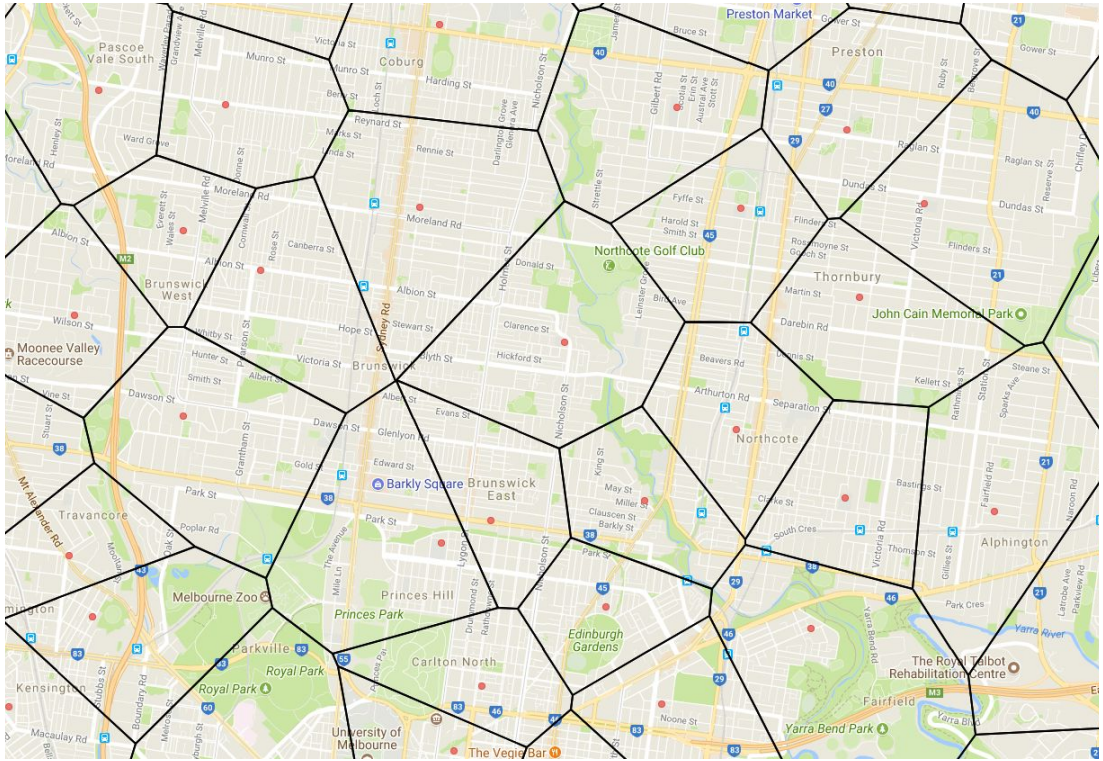


Diagrama de Voronoi

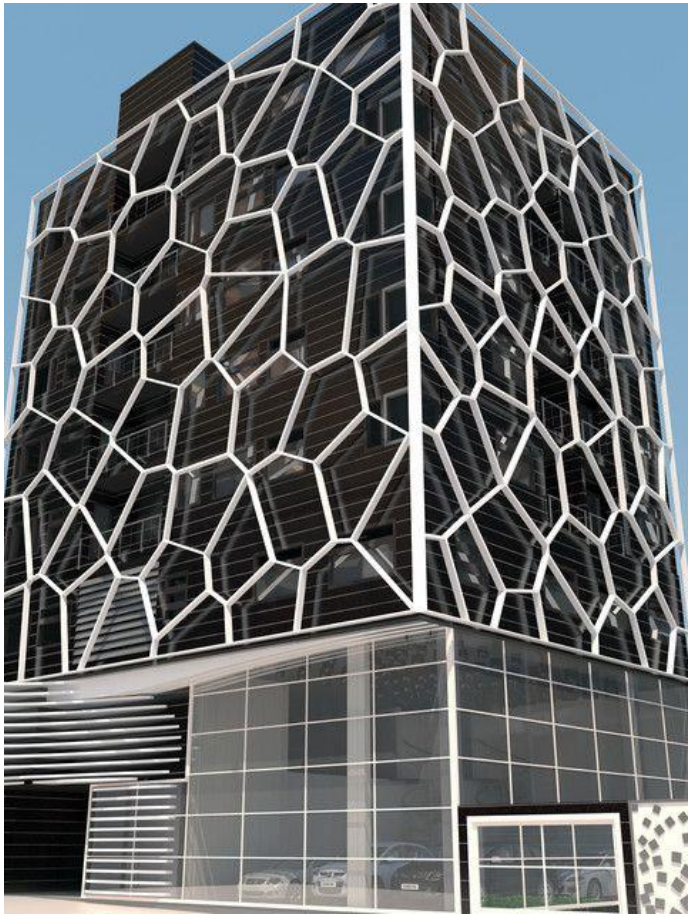
- Algumas aplicações...

- Planejamento urbano (onde construir uma nova escola, hospital, delegacia, loja, etc)

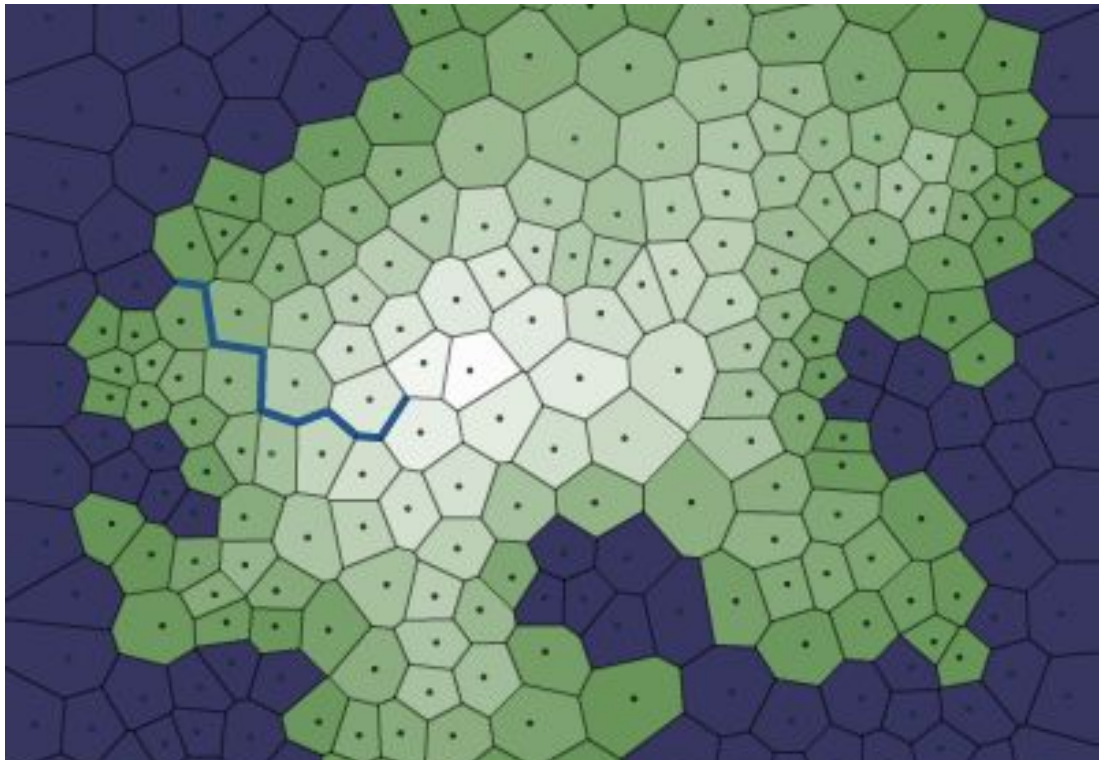


http://melbourneschoolzones.com/

- Arquitetura



- Geração procedural



Outras aplicações

- Biologia
  - Competição no crescimento de raízes de plantas
  - Território de animais
  - Representação de células
- Geografia
  - Área de influência de organizações
- Astronomia
  - Identificar clusters de estrelas e galáxias
- Robótica
  - Planejamento de caminhos na presença de obstáculos

Diagrama de Voronoi em imagens

Diagrama de Voronoi em imagens

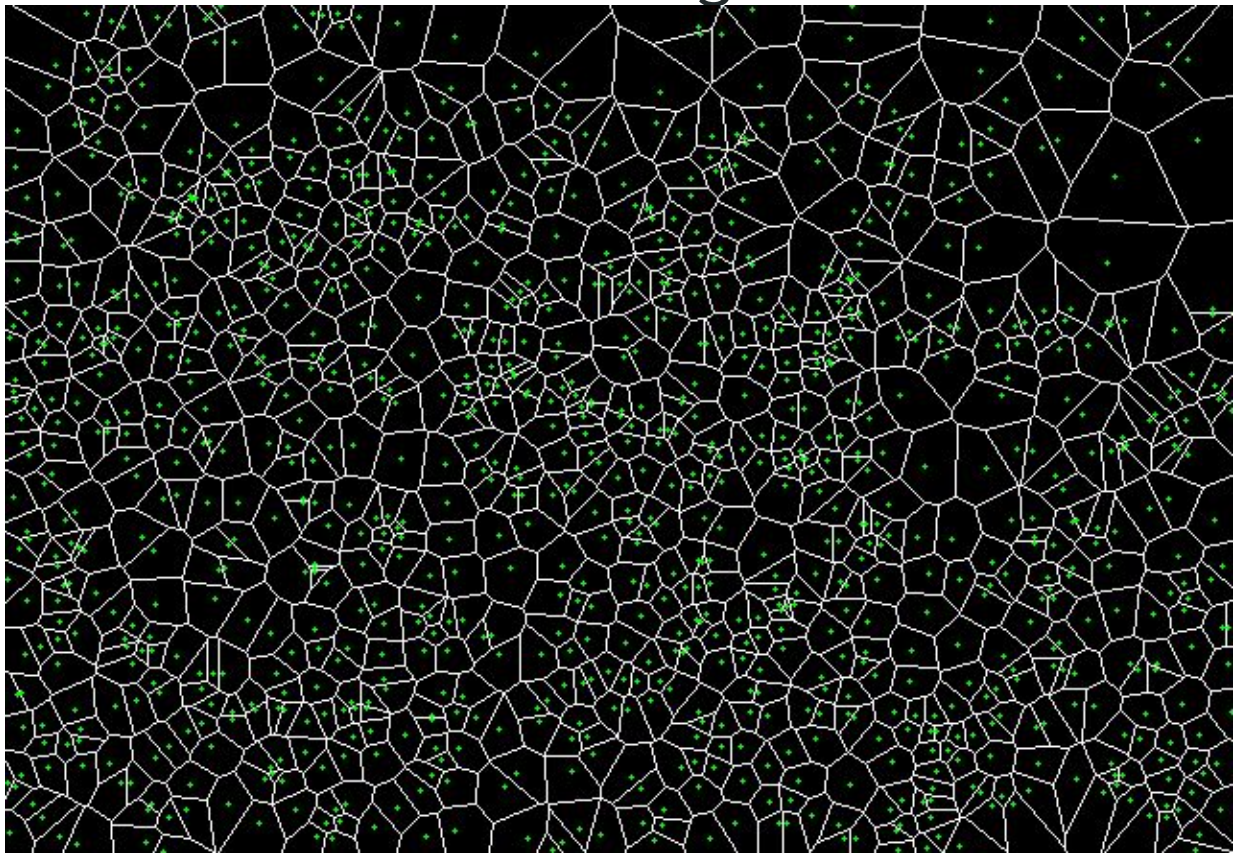


Diagrama de Voronoi em imagens

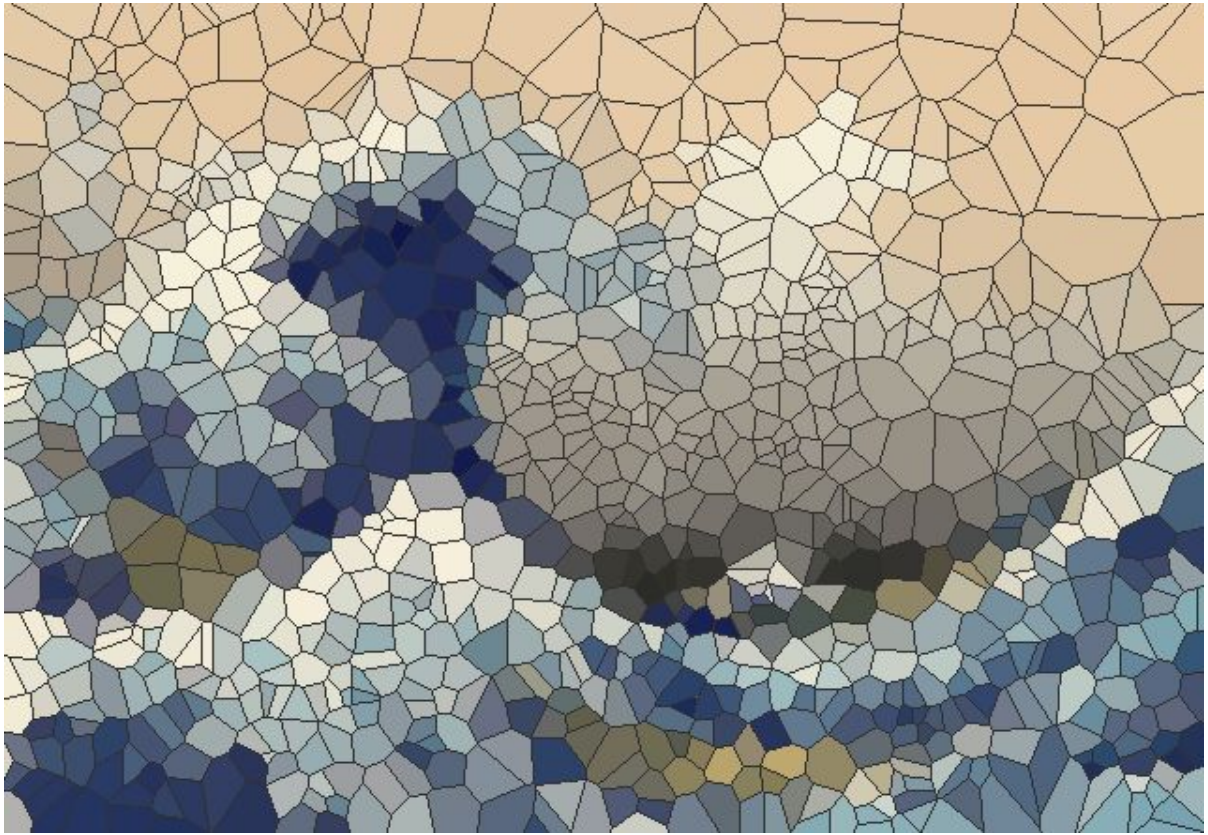


Diagrama de Voronoi em imagens

- Como fazer?

Força Bruta

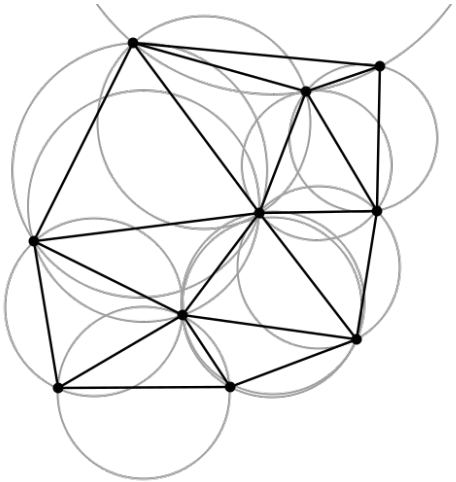
```
para y de 0 até altura:
    para x de 0 até largura:
        menor_distancia = ∞
        para cada ponto:
            calcula a distância de (x,y) até o ponto
            se distancia calculada < menor_distancia:
                menor_distancia = distancia_calculada
                mais_perto = ponto
        f(x,y) = f(mais_perto.x, mais_perto.y)
```

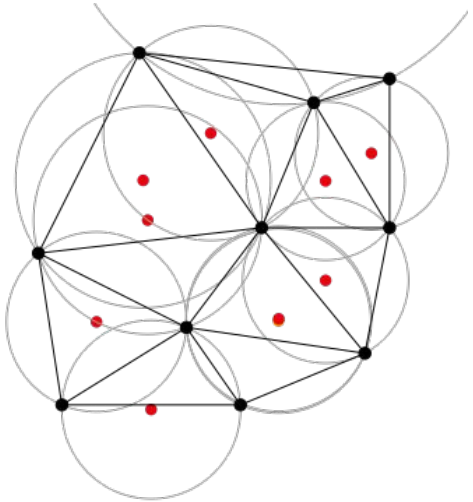
O(ruim)

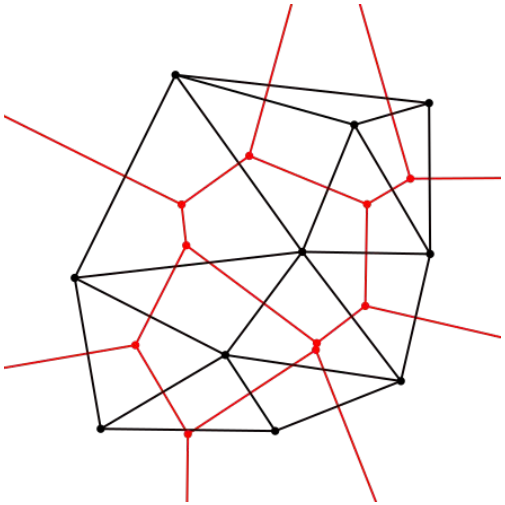
Triangulação de Delaunay

- Nenhum ponto está dentro do circuncírculo de qualquer triângulo da triangulação
- Maximiza o menor ângulo
- Dualidade entre o Diagrama de Voronoi e Triangulação de Delaunay
  - Sabendo um é fácil descobrir o outro
  - Grafo dual

Triangulação de Delaunay



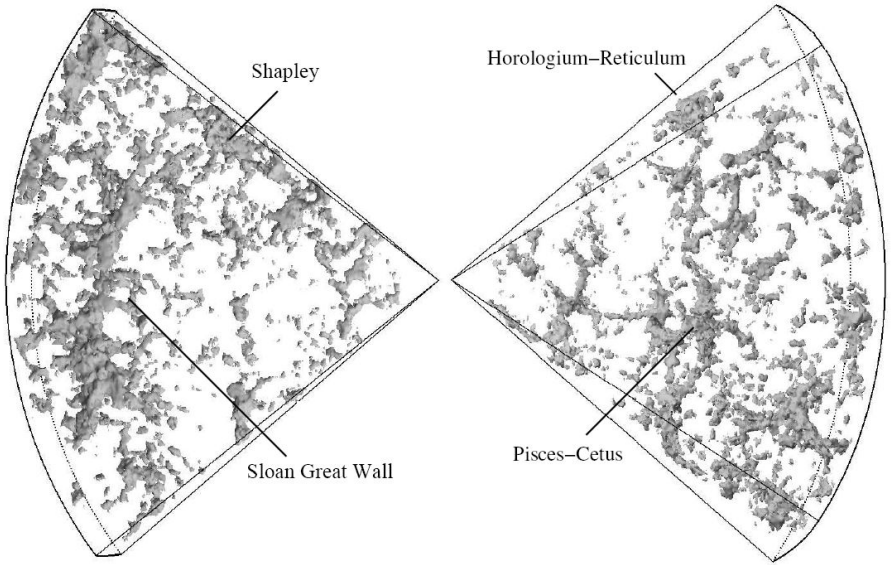




Triangulação de Delaunay

Aplicações

- Delaunay Tessellation Field Estimator - Estimador de campo de tesselação Delaunay (renderização de objetos do espaço)
- Modelagem 3D



Algoritmo

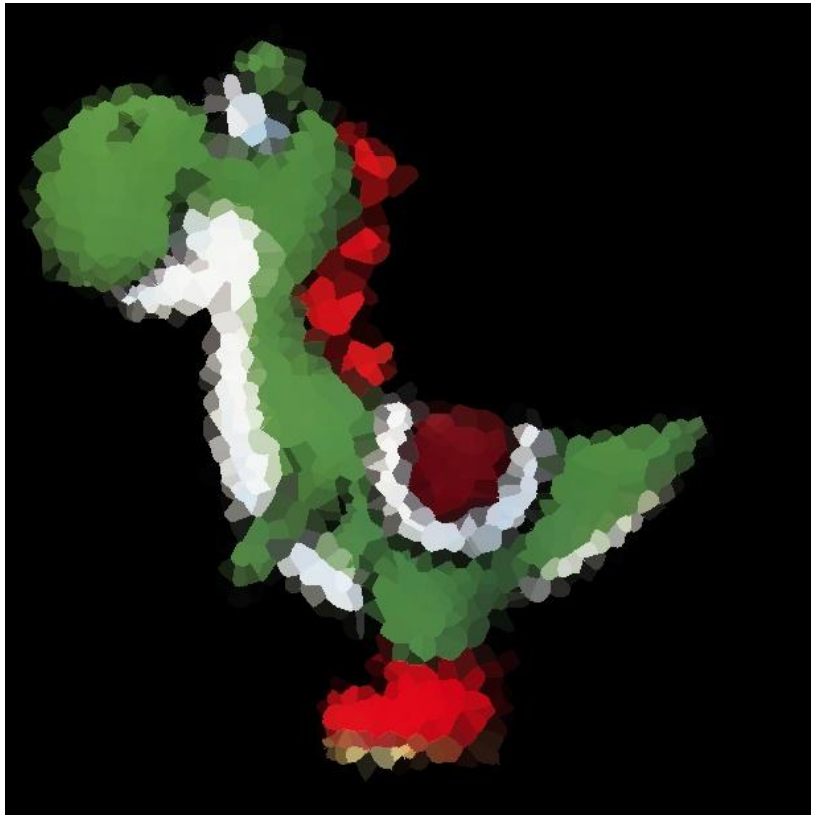
Ideia geral

1. Encontrar os pontos relevantes da imagem
2. Gerar a triangulação de Delaunay (Algoritmo de Bowyer-Watson)
3. Gerar o diagrama de Voronoi a partir da triangulação
4. Colorir as células do diagrama

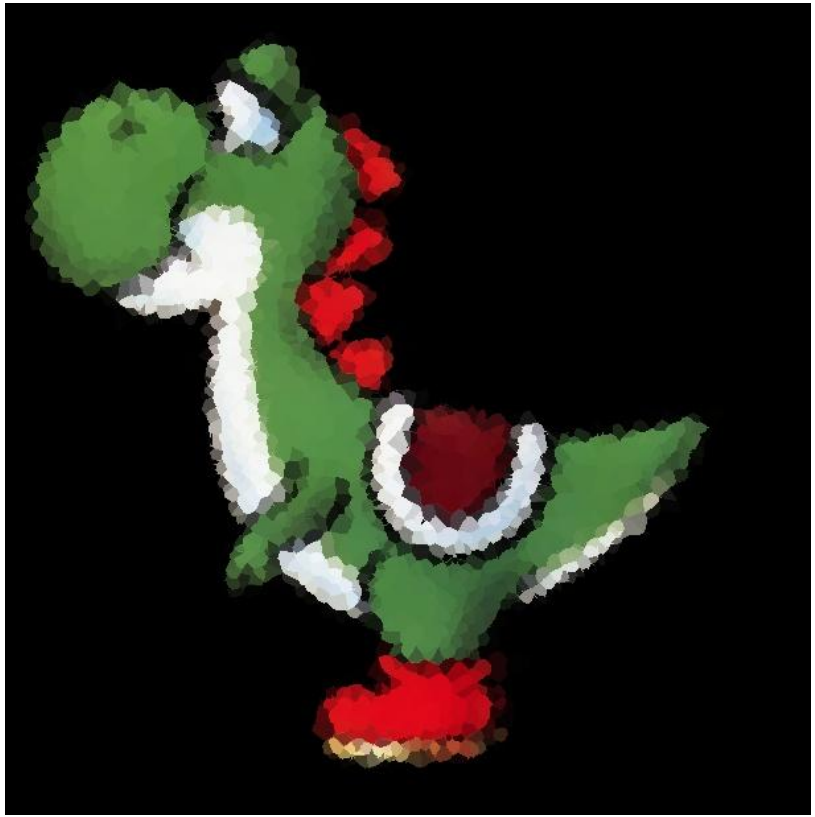
Encontrando pontos relevantes

- Totalmente aleatórios
  - Muitas células desperdiçadas em áreas com pouca variação de cor ou sem detalhes
- Aleatórios com pesos
  - Regiões perto de bordas tem mais chances de serem escolhidas
    - Regiões com detalhe = mais pontos
    - Regiões sem detalhe = poucos pontos

Pontos aleatórios



Mais pontos perto das bordas



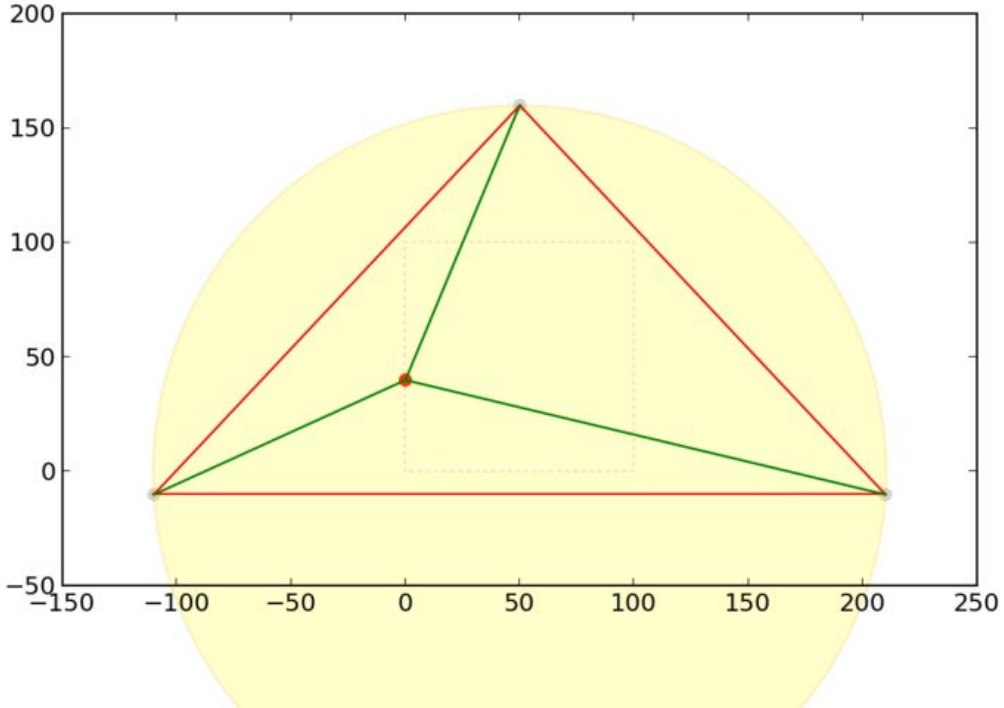
Gerando a triangulação de Delaunay

Algoritmo de Bowyer-Watson

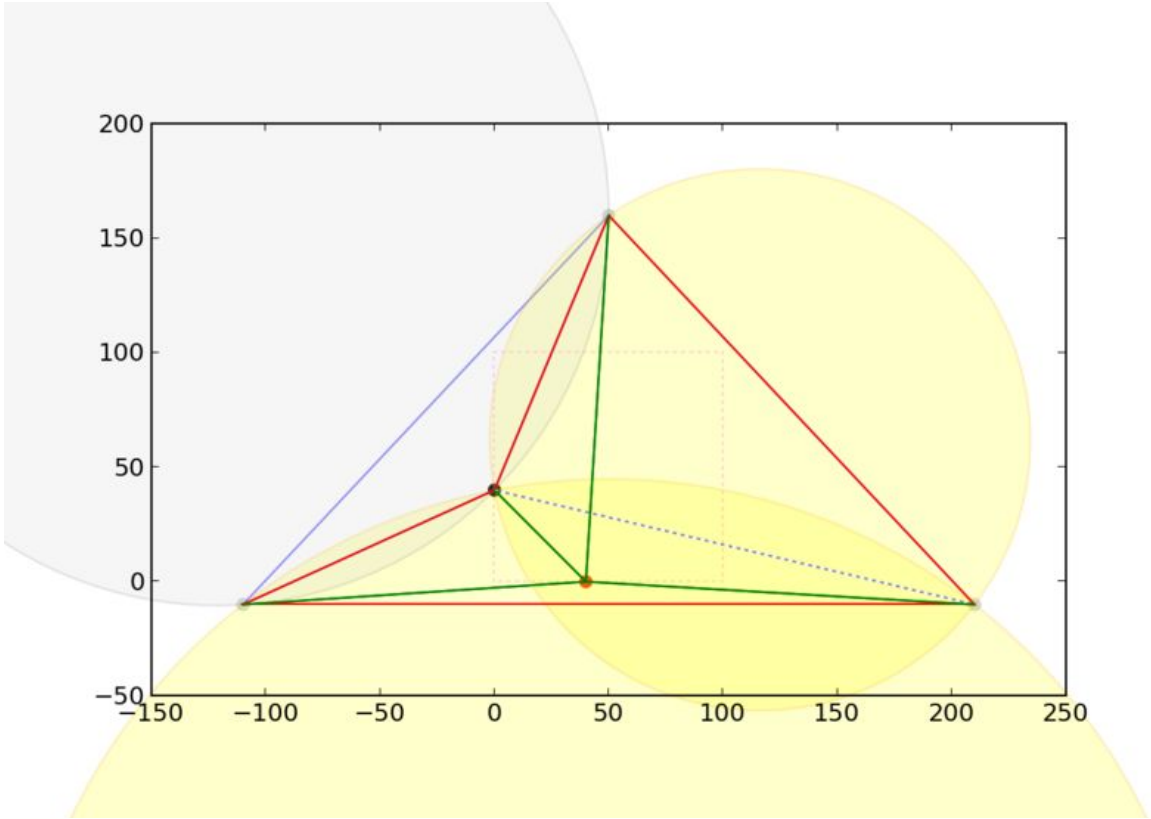
Dado um conjunto de pontos:

- Cria um super-triângulo que englobe todos os pontos (polígono inicial)
- Insere os pontos um a um (abordagem incremental) na triangulação
- Para inserir um ponto:
  - Remove da triangulação os triângulos inválidos (em que o novo ponto está dentro do circuncírculo do triângulo)
  - Define o polígono a ser usado na próxima iteração
  - Insere os novos triângulos na triangulação
    - Formados pelas extremidades de cada aresta do polígono ligadas ao ponto

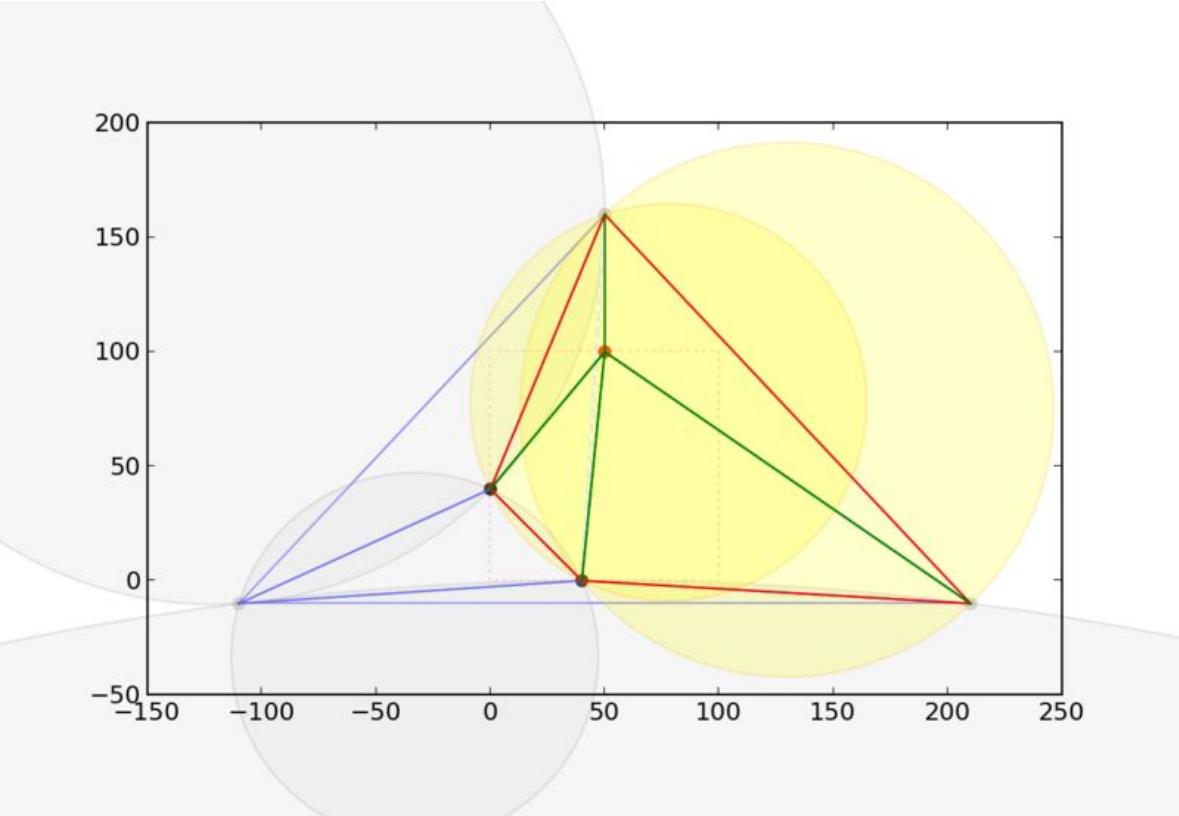
Gerando a triangulação de Delaunay



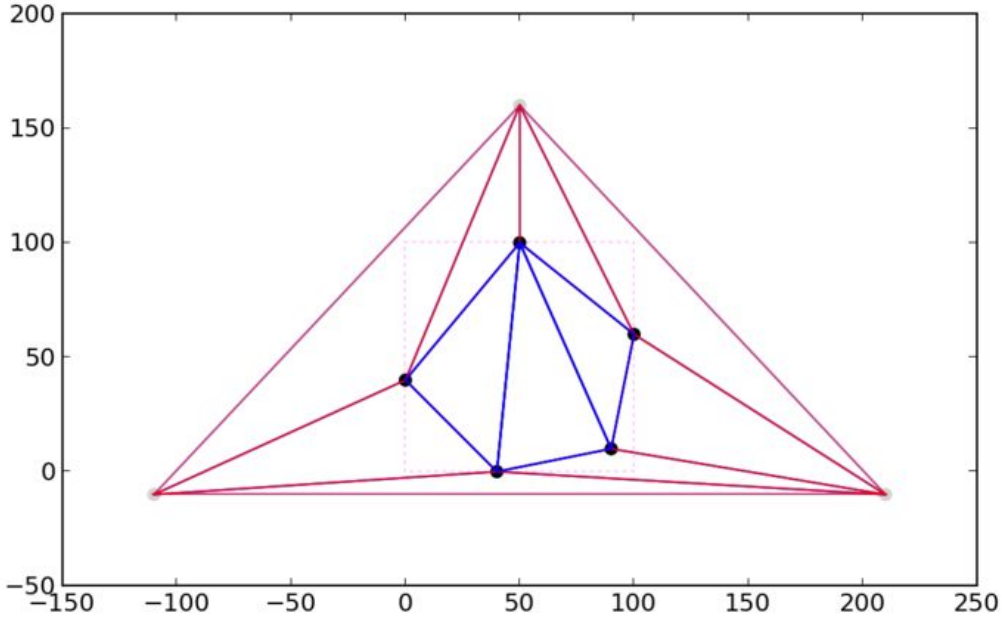
Gerando a triangulação de Delaunay



Gerando a triangulação de Delaunay



Gerando a triangulação de Delaunay



Gerando o diagrama de Voronoi

- Basta conectar o circuncentro de cada triângulo vizinho

Colorindo as células

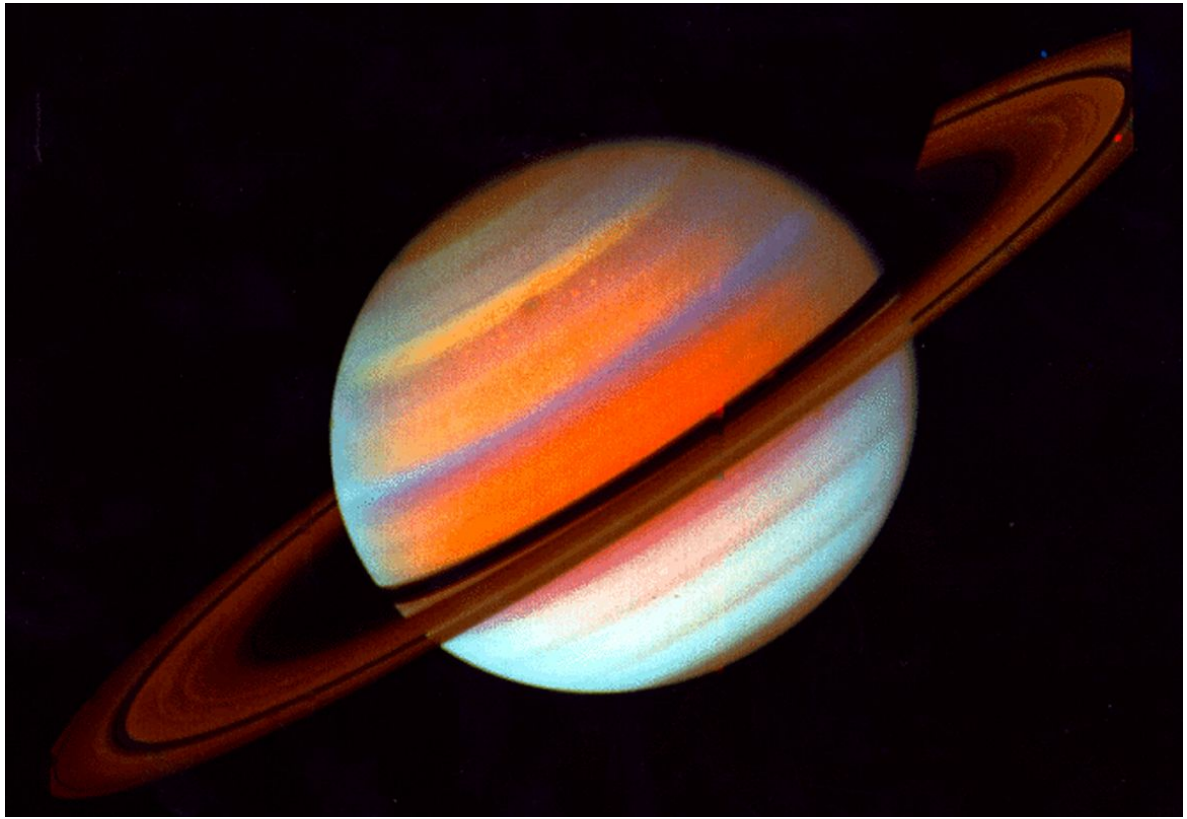
- Rotulagem de componentes conexos (connectedComponentsWithStats)
- Para cada célula:
  - Cria uma máscara para cada label
  - Calcula a cor média de cada célula usando a máscara
  - Pega o centro
  - Floodfill
- Tratando as arestas das células
  - FIltro da mediana para tirar as arestas
  - Mudar a cor para combinar mais com a imagem

Sem arestas

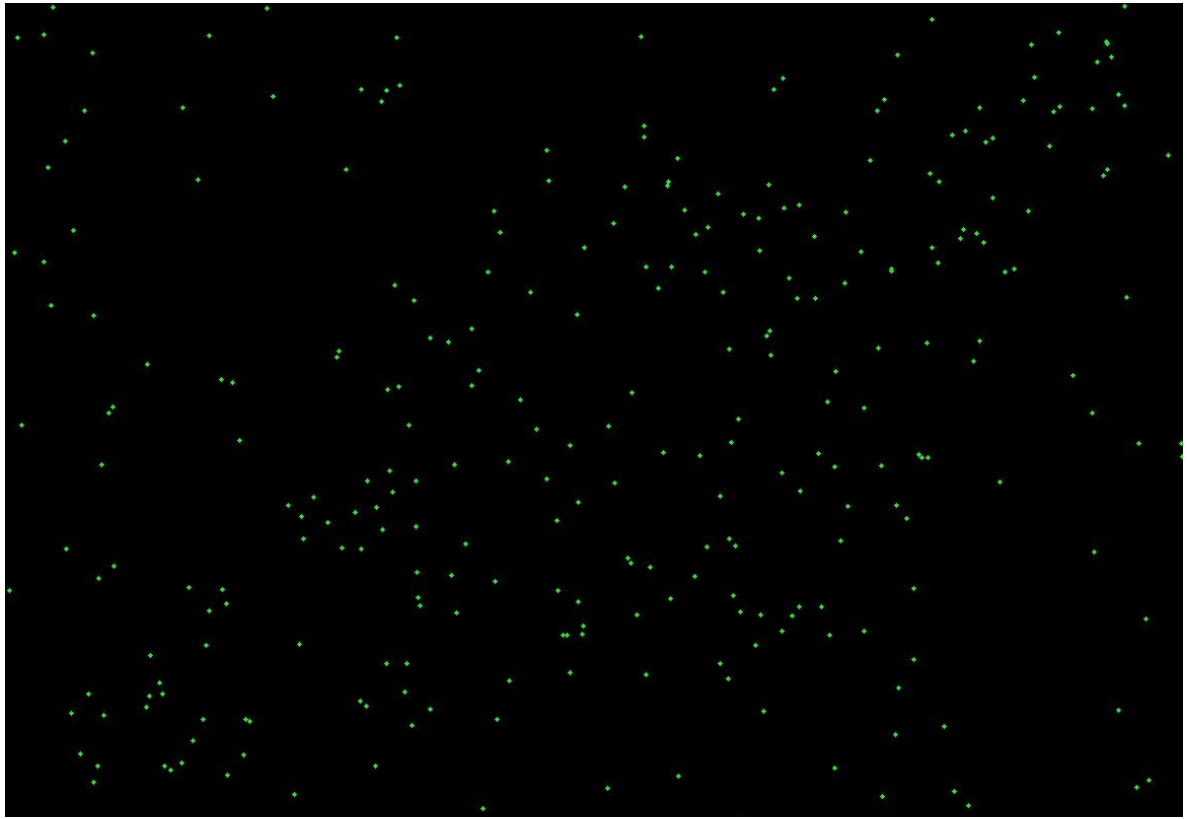
Mudando a cor das arestas

- Alguns exemplos...

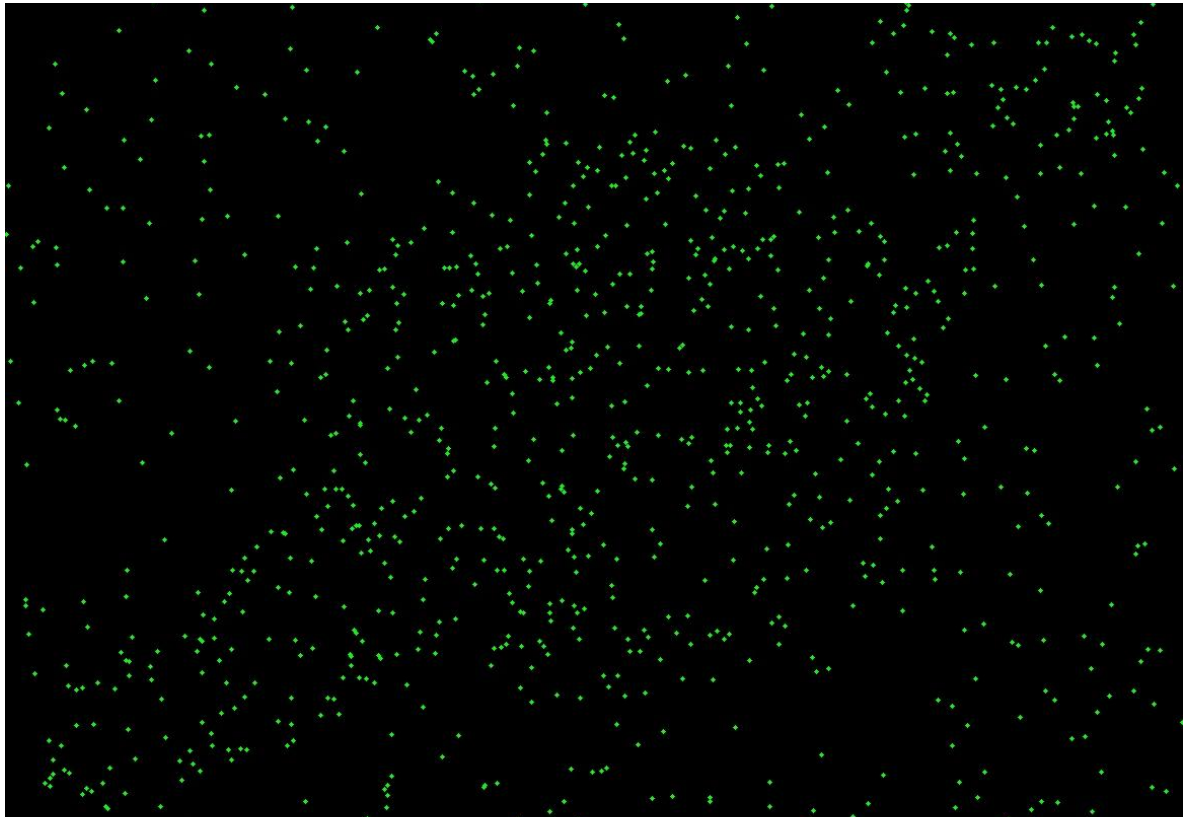
Exemplo 1



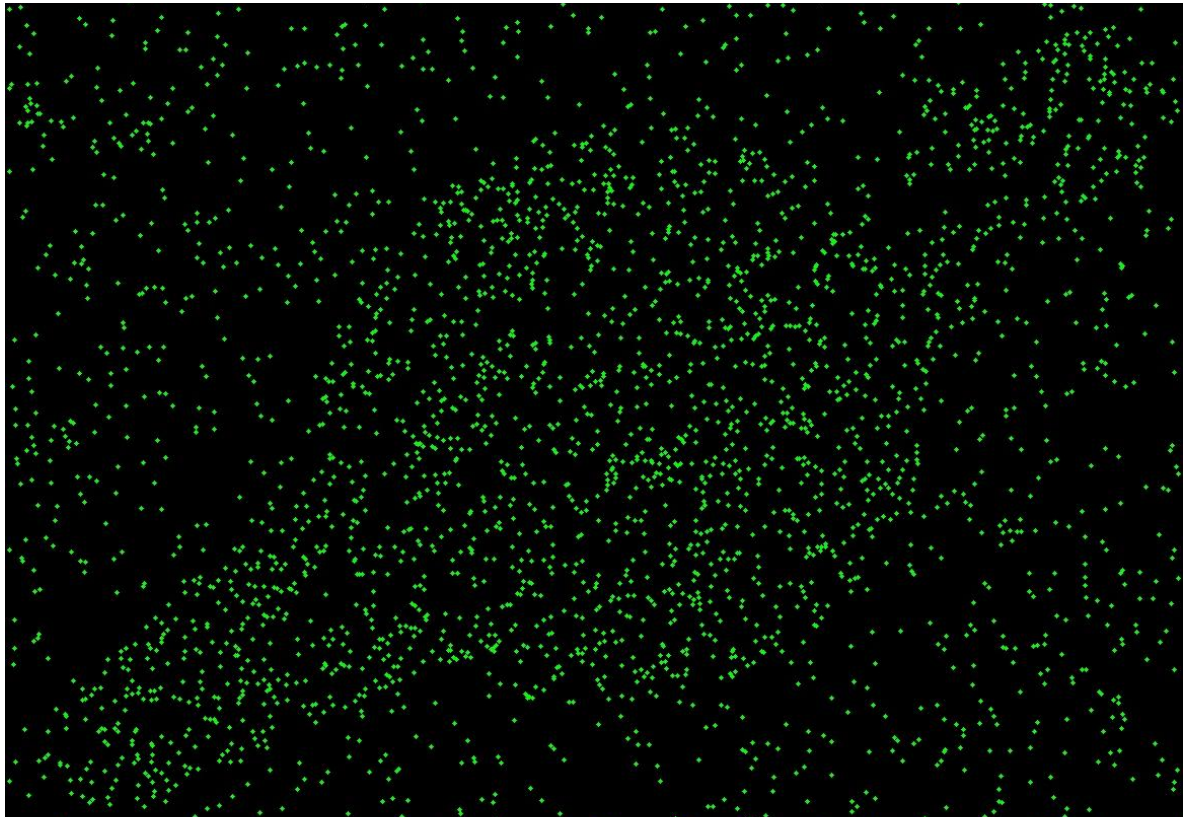
300 pontos



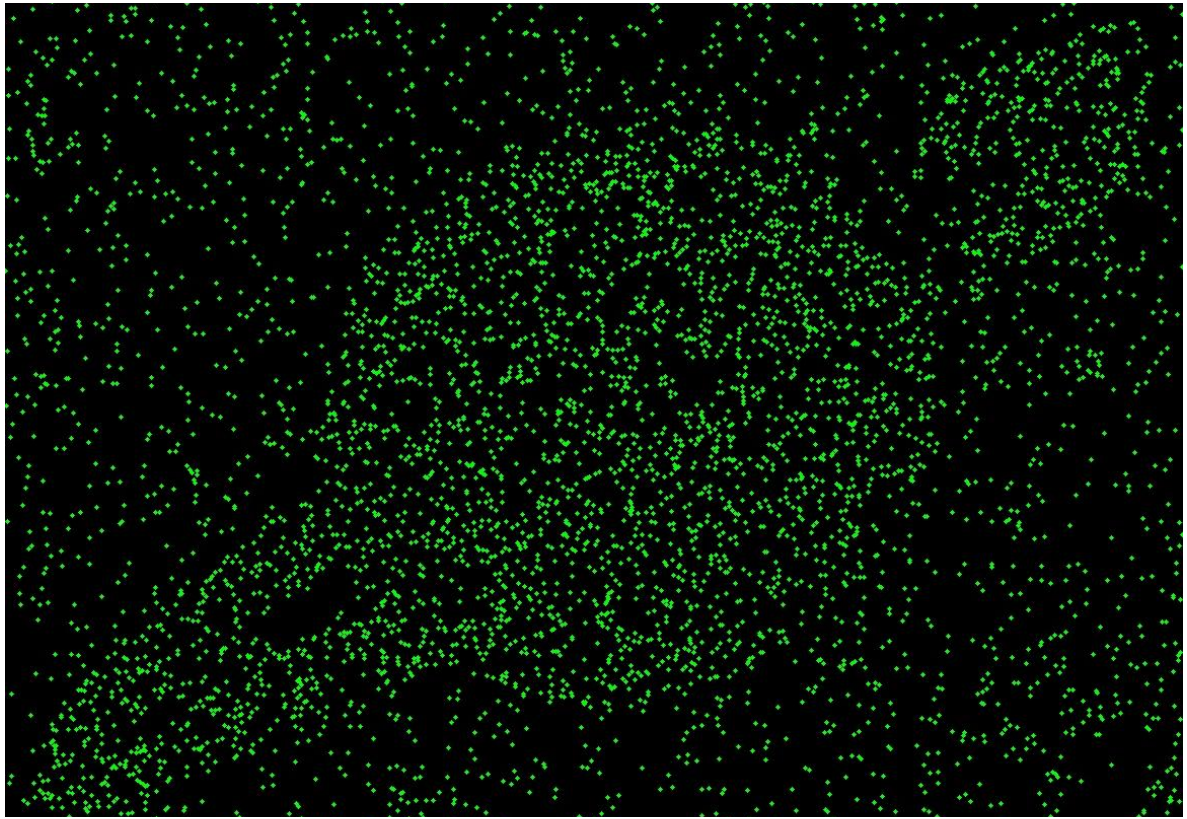
1000 pontos



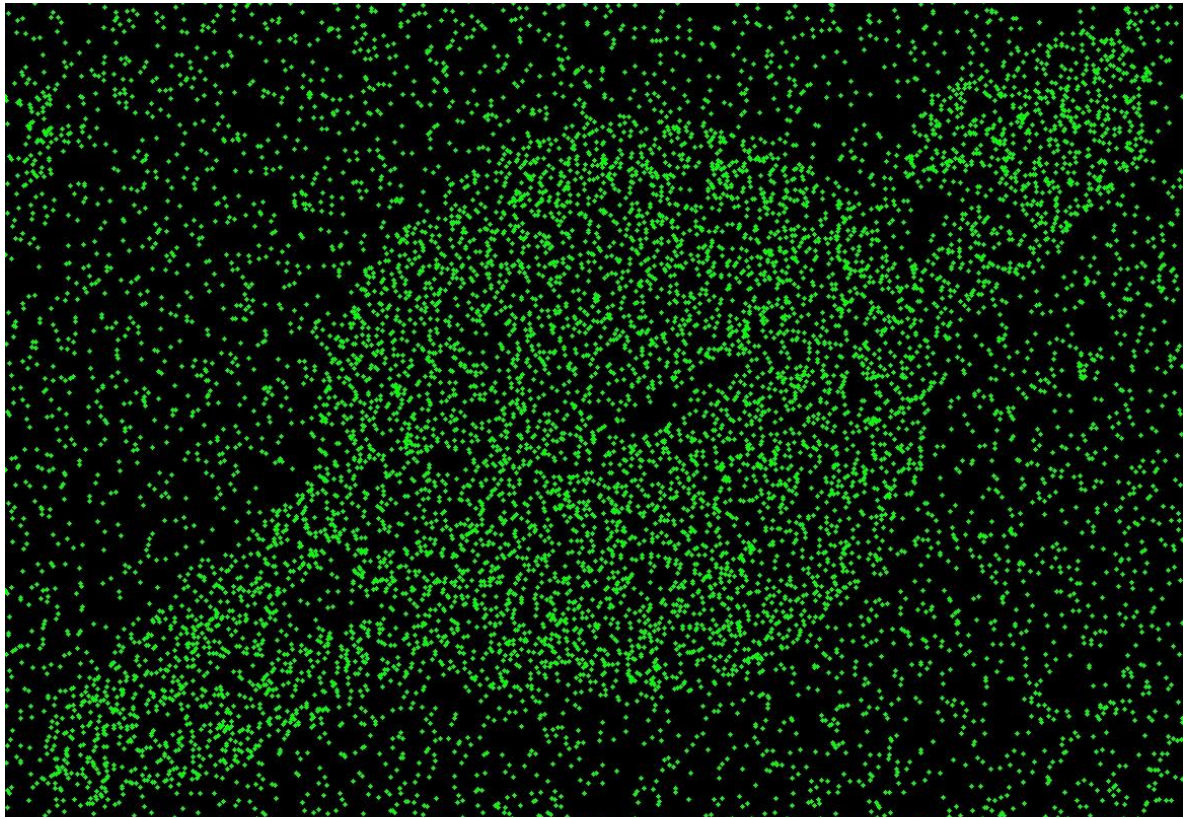
3000 pontos



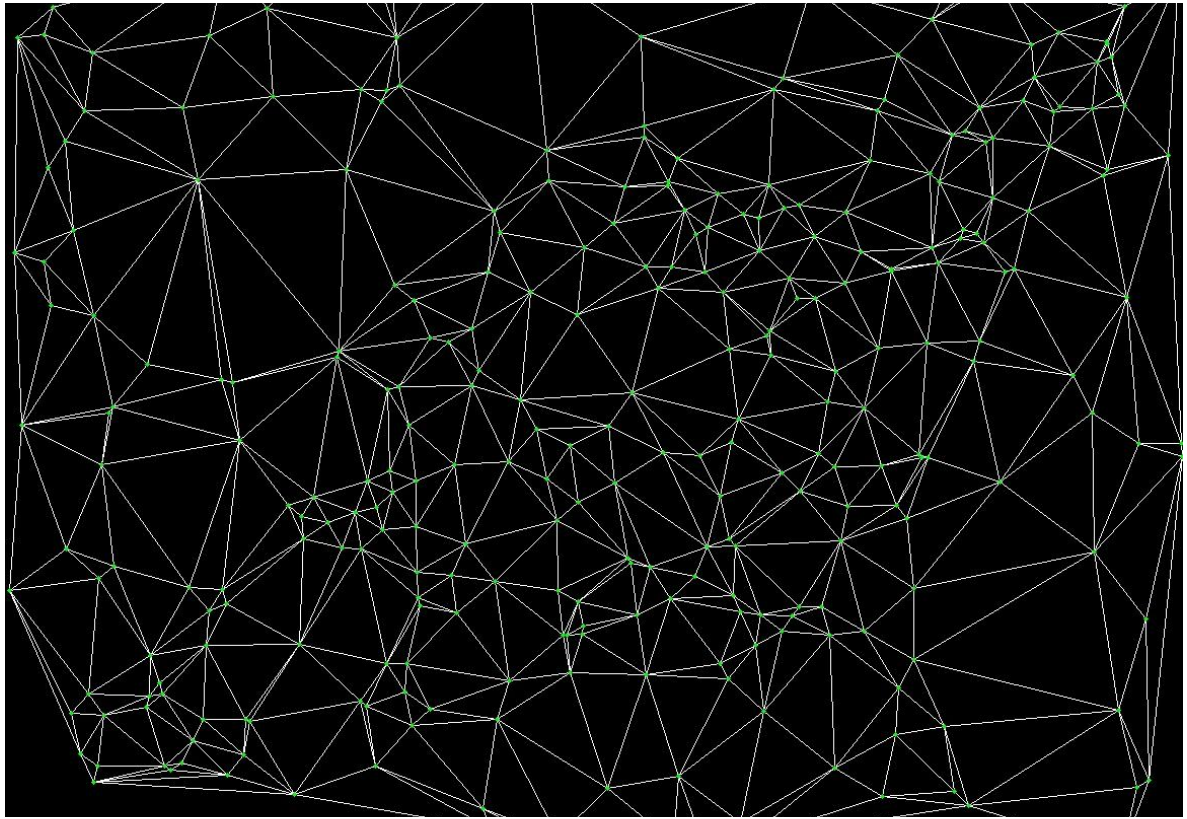
5000 pontos



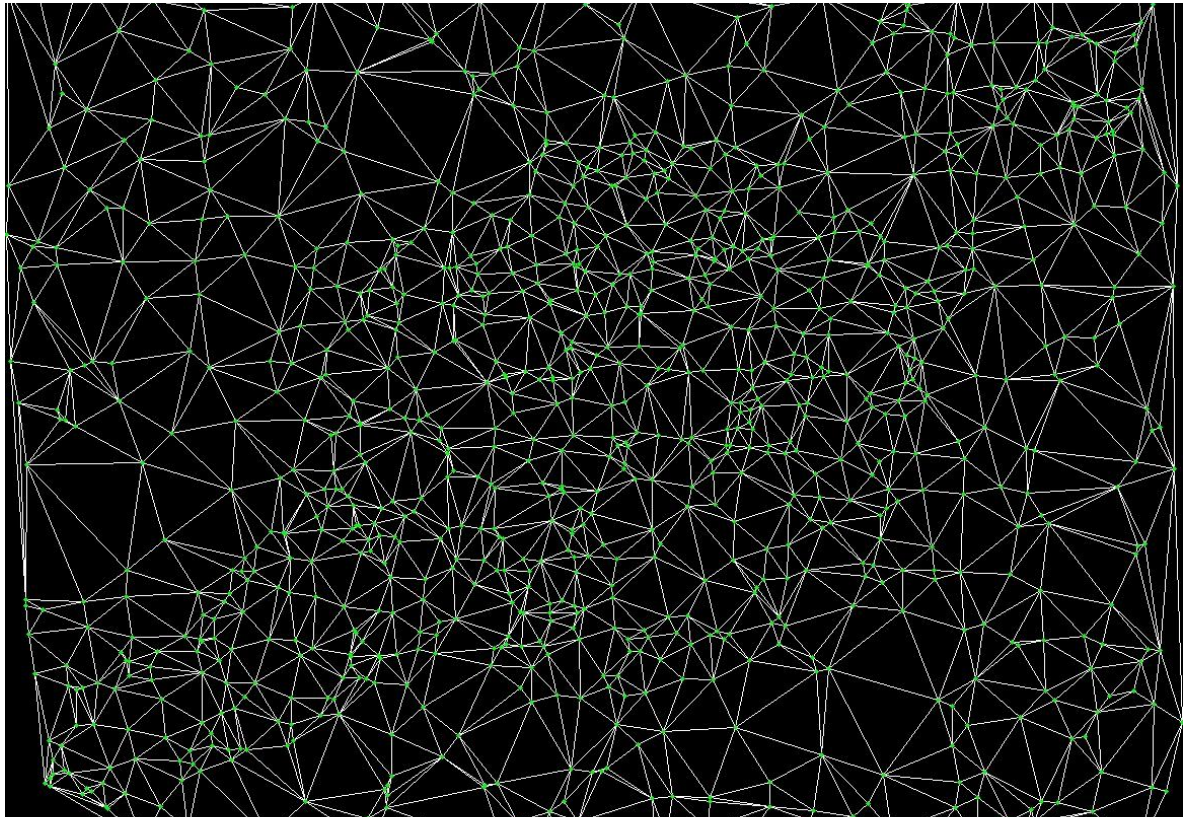
10000 pontos



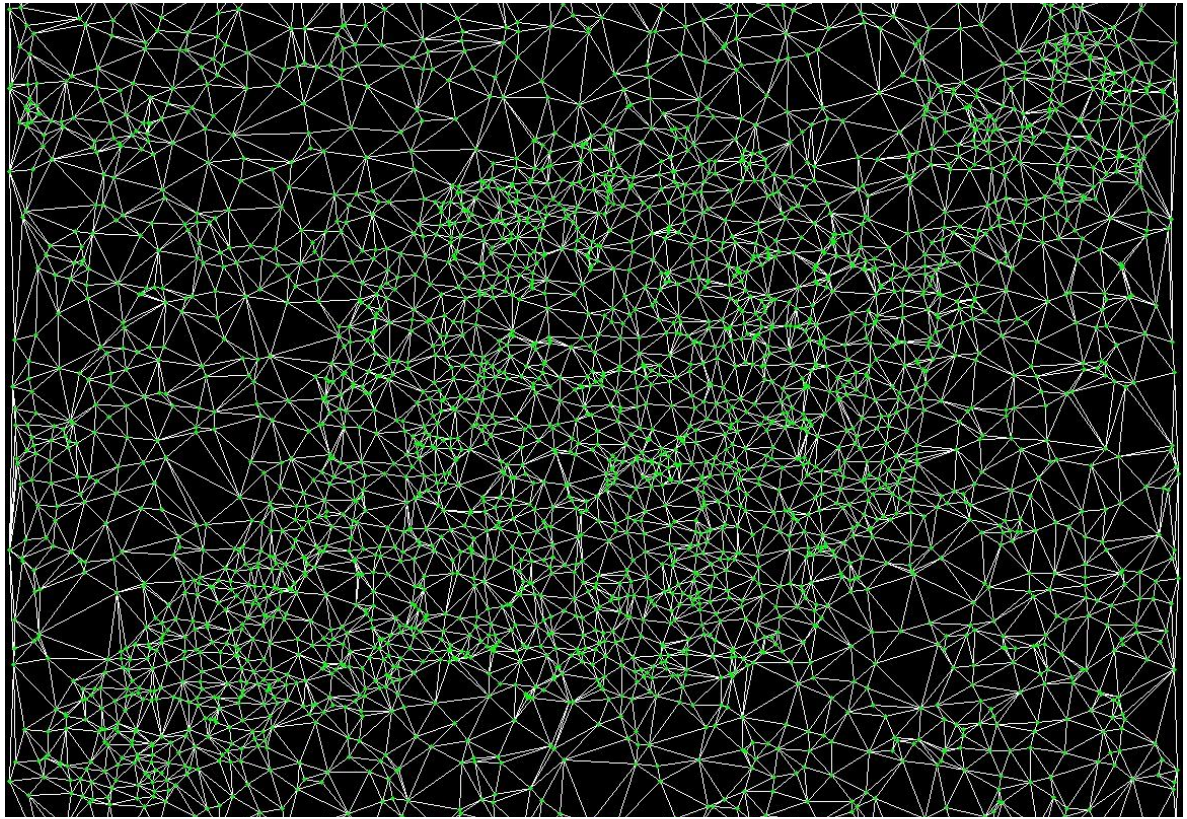
Delaunay - 300 pontos



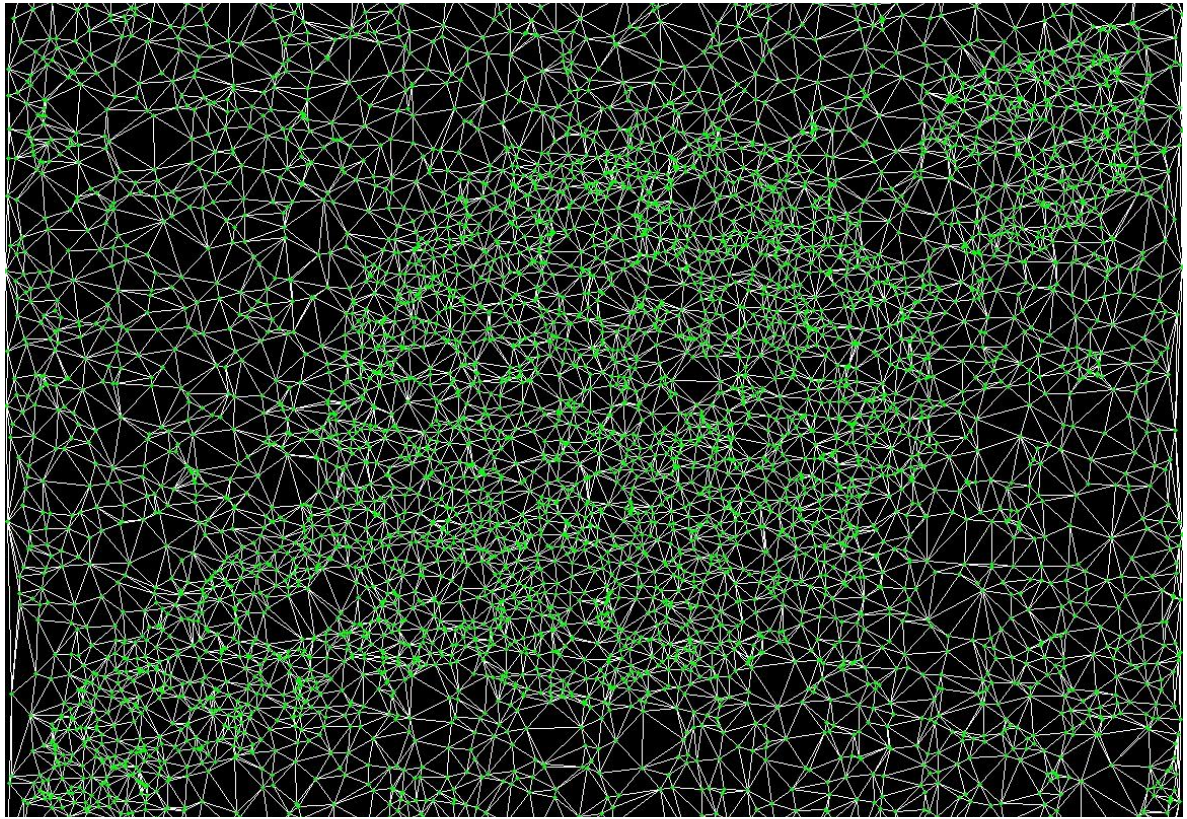
Delaunay - 1000 pontos



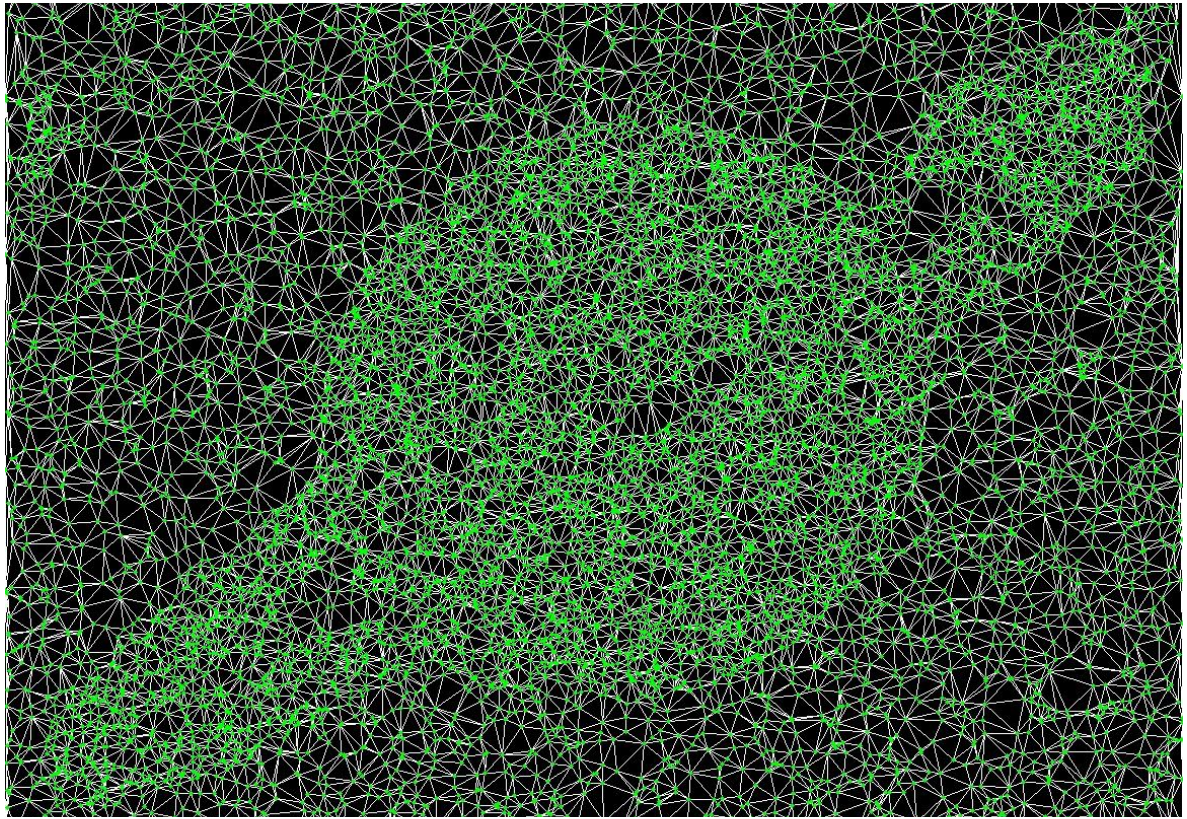
Delaunay - 3000 pontos



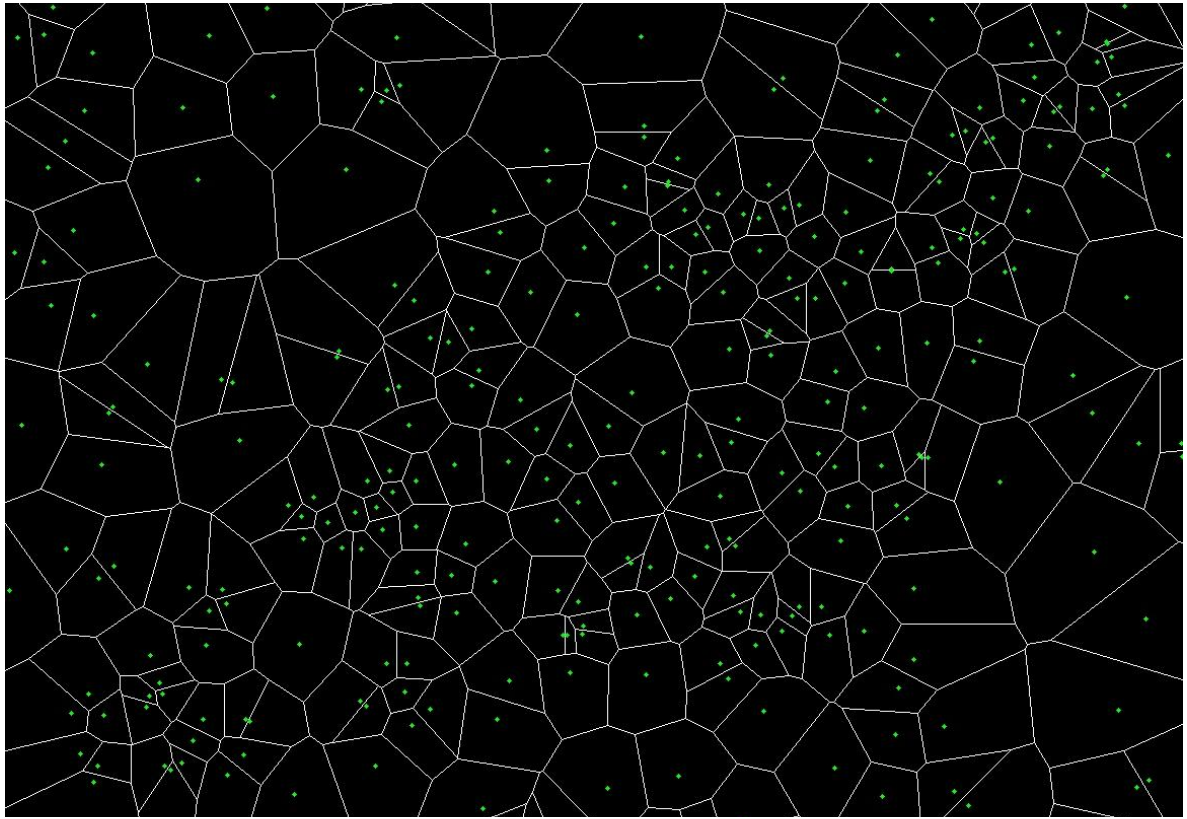
Delaunay - 5000 pontos



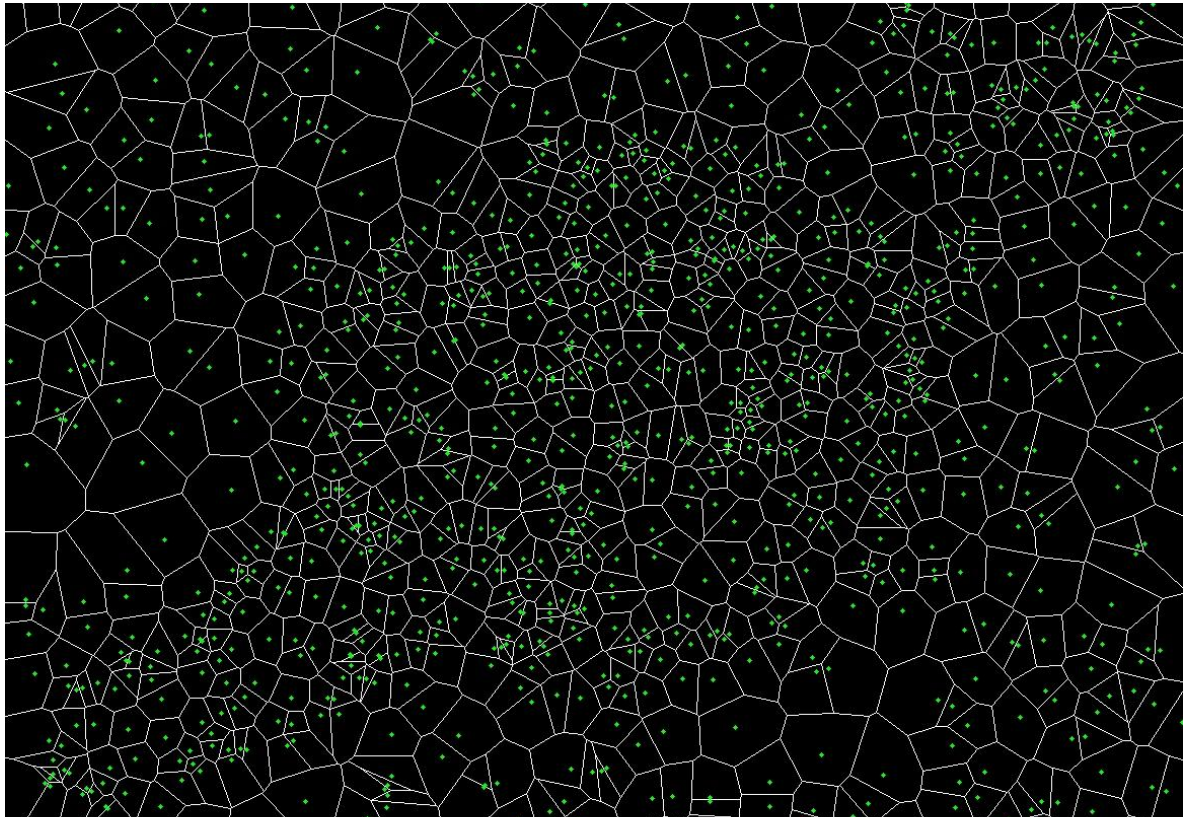
Delaunay - 10000 pontos



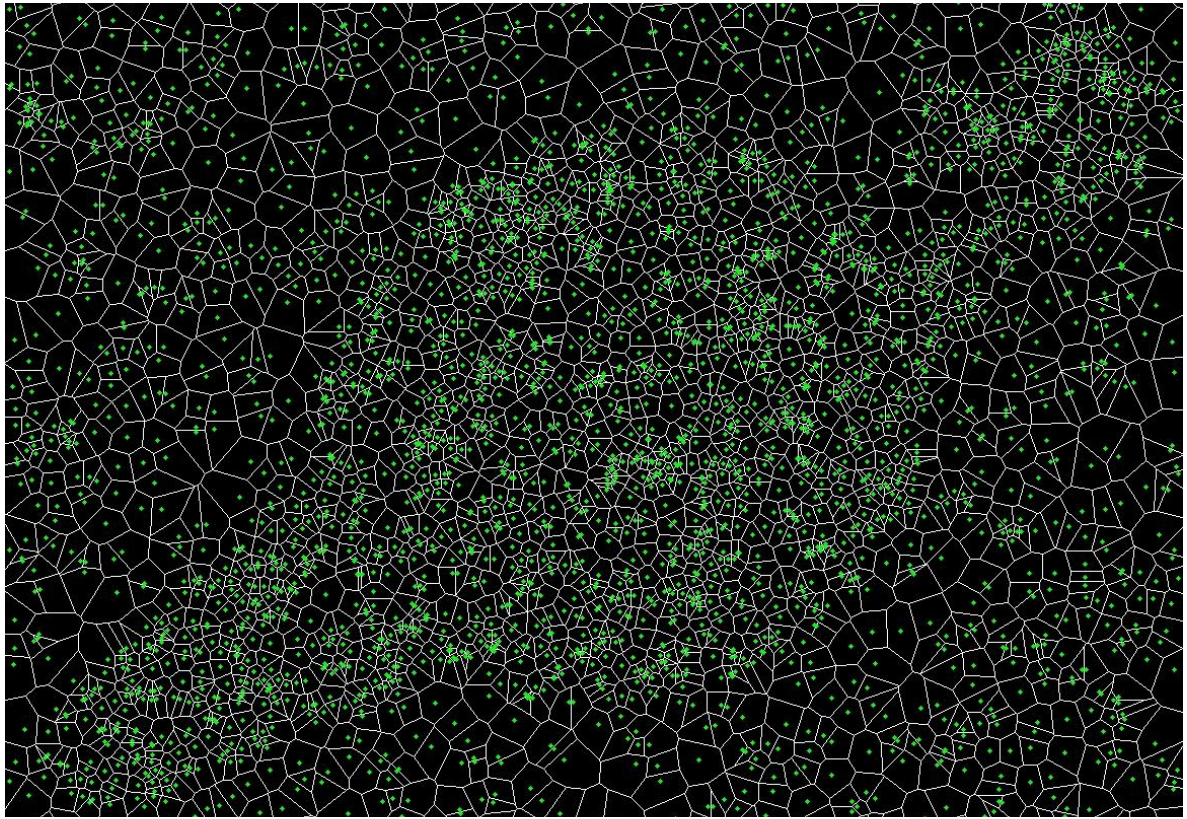
Voronoi - 300 pontos



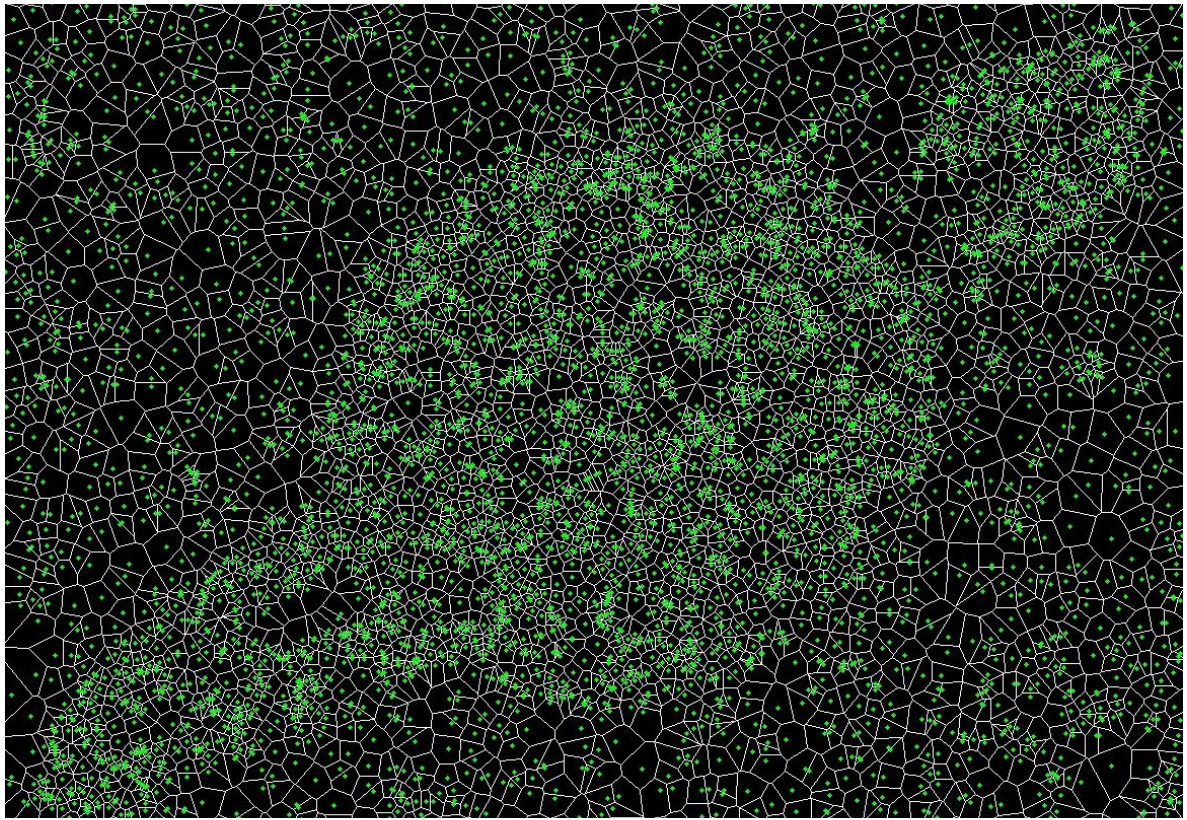
Voronoi - 1000 pontos



Voronoi - 3000 pontos



Voronoi - 5000 pontos



Voronoi - 10000 pontos

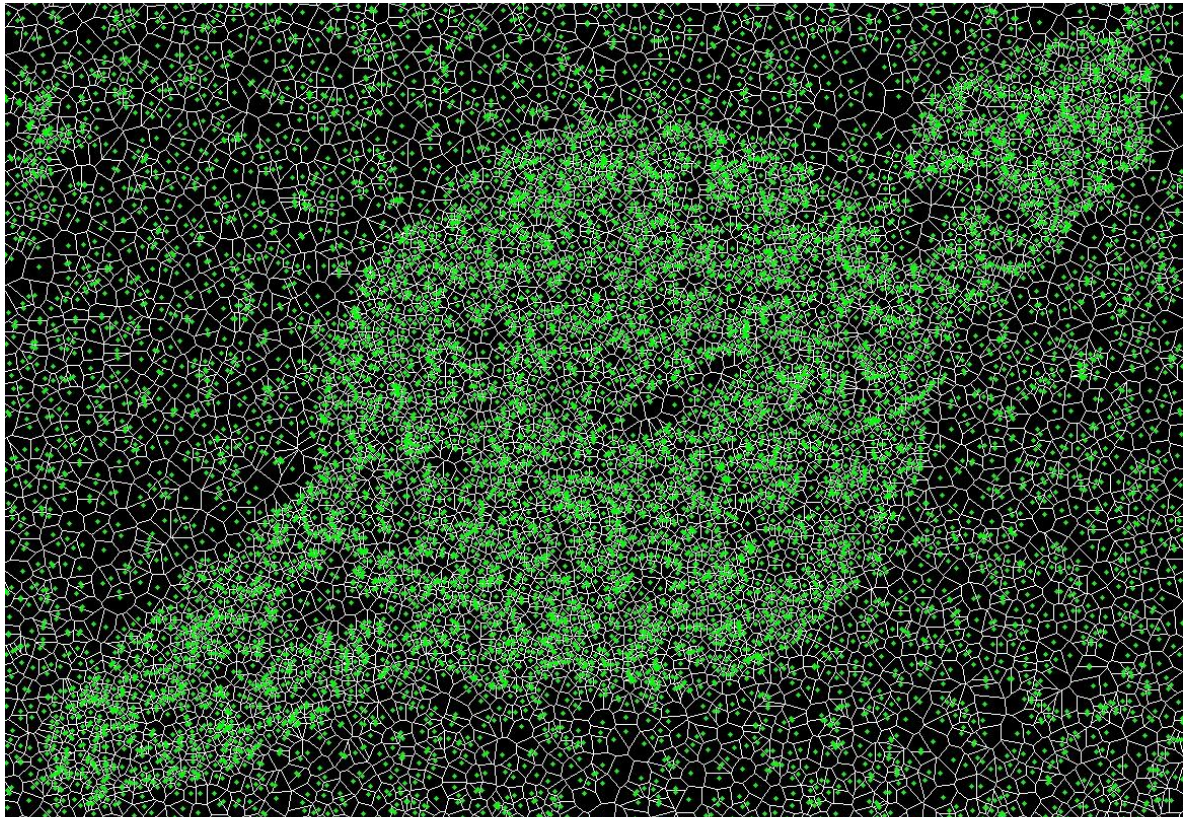


Imagem final - 300 pontos

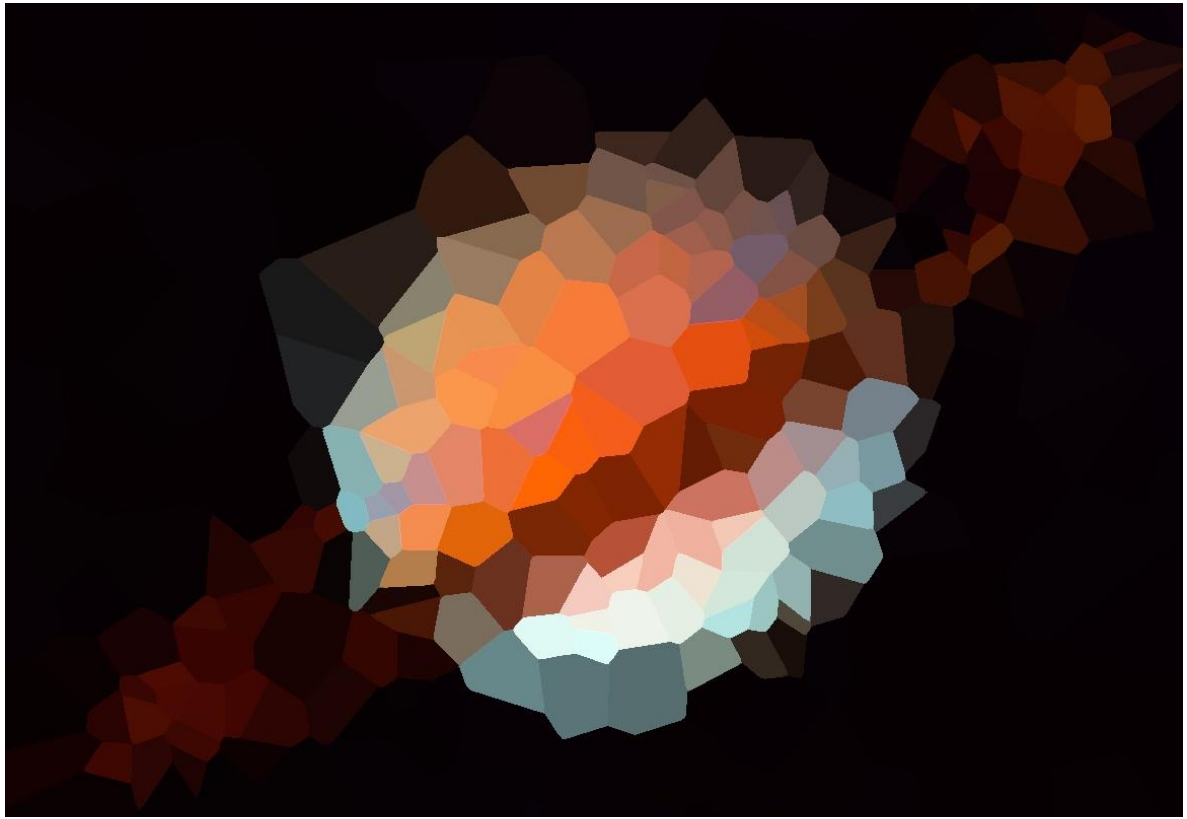


Imagem final - 1000 pontos

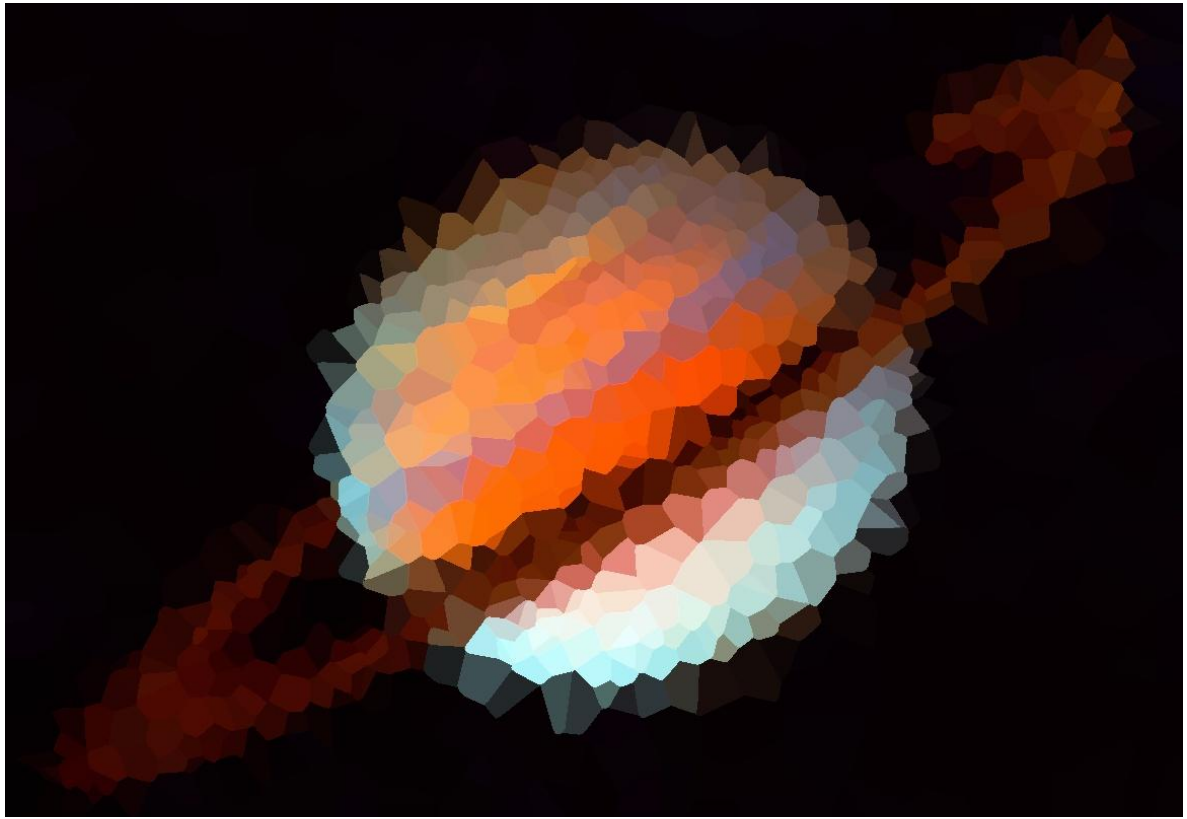


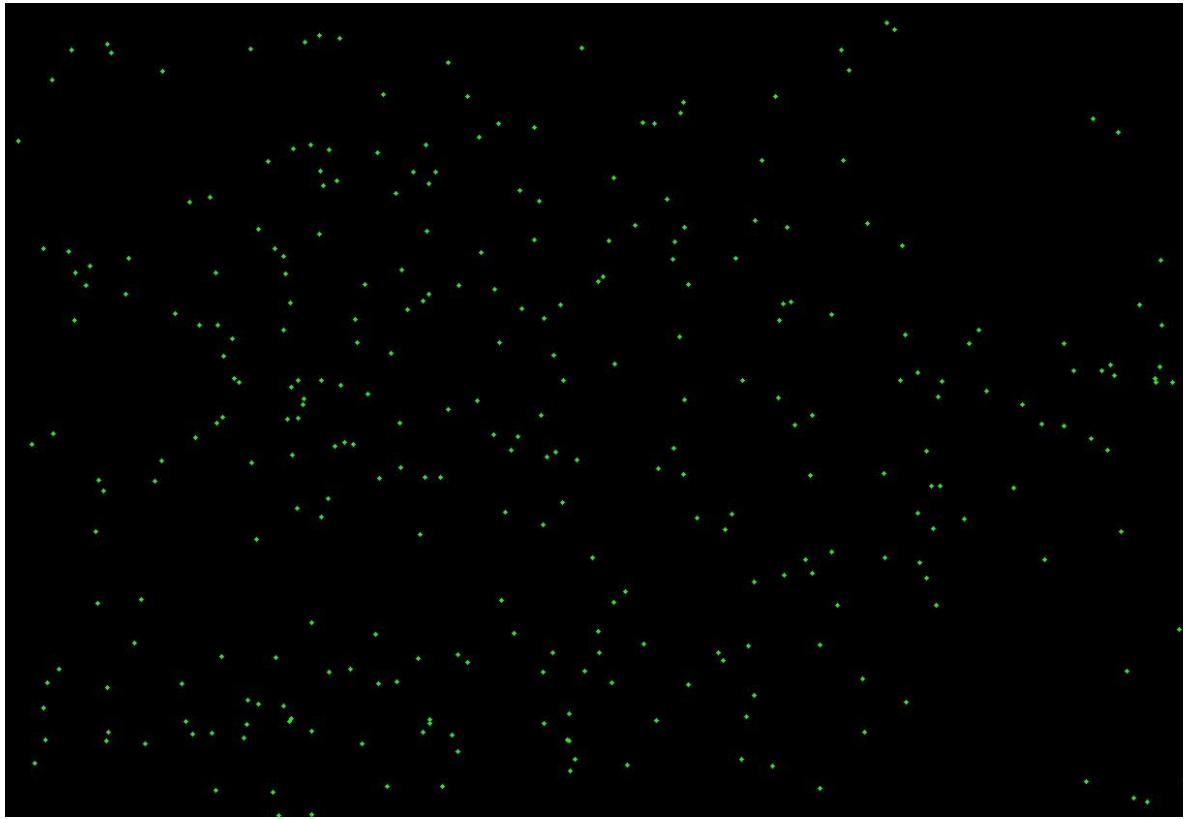
Imagem final - 3000 pontos

Imagem final - 5000 pontos

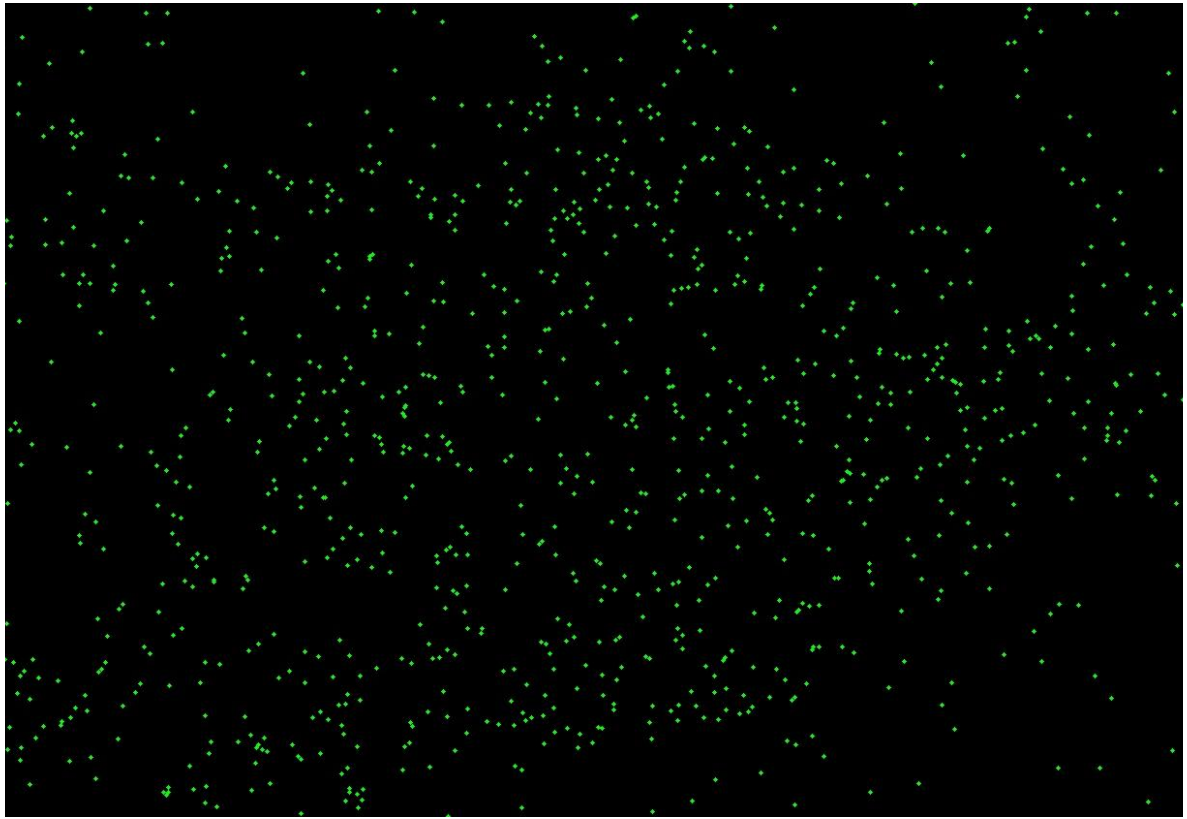
Imagem final - 10000 pontos

Exemplo 2

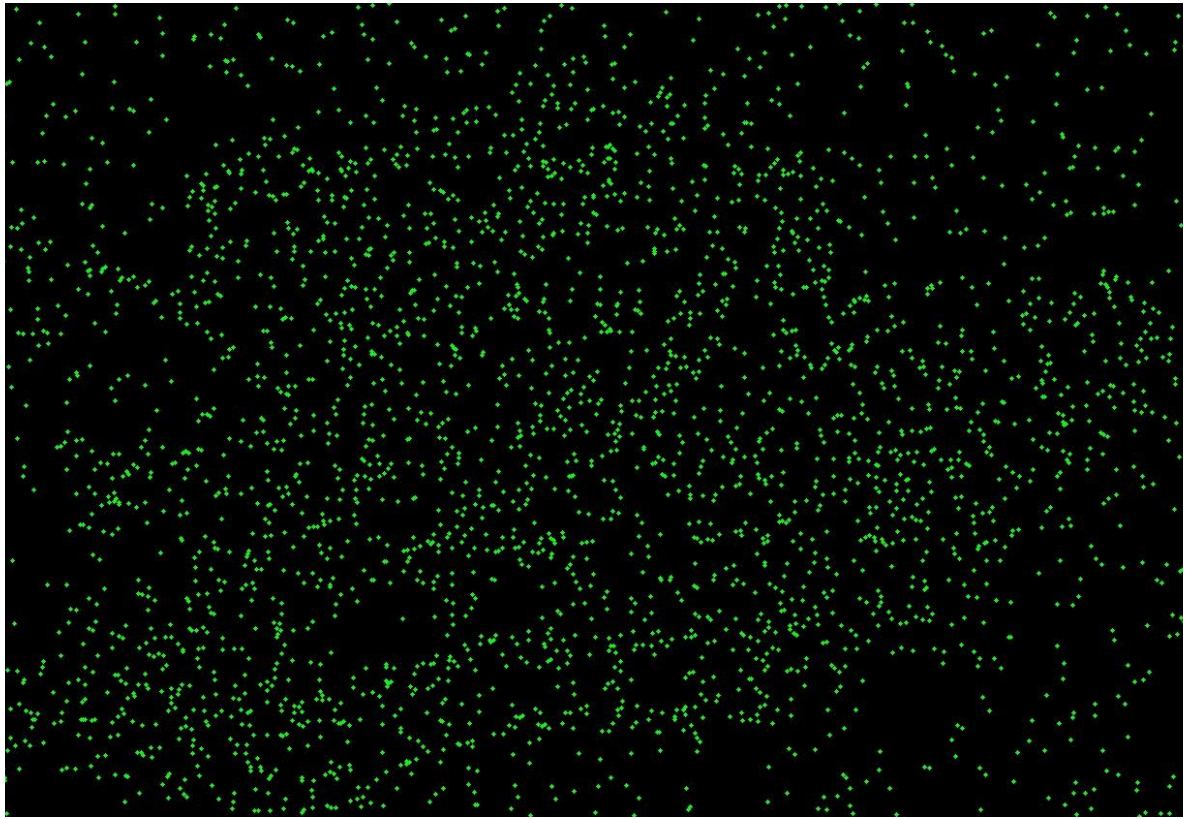
300 pontos



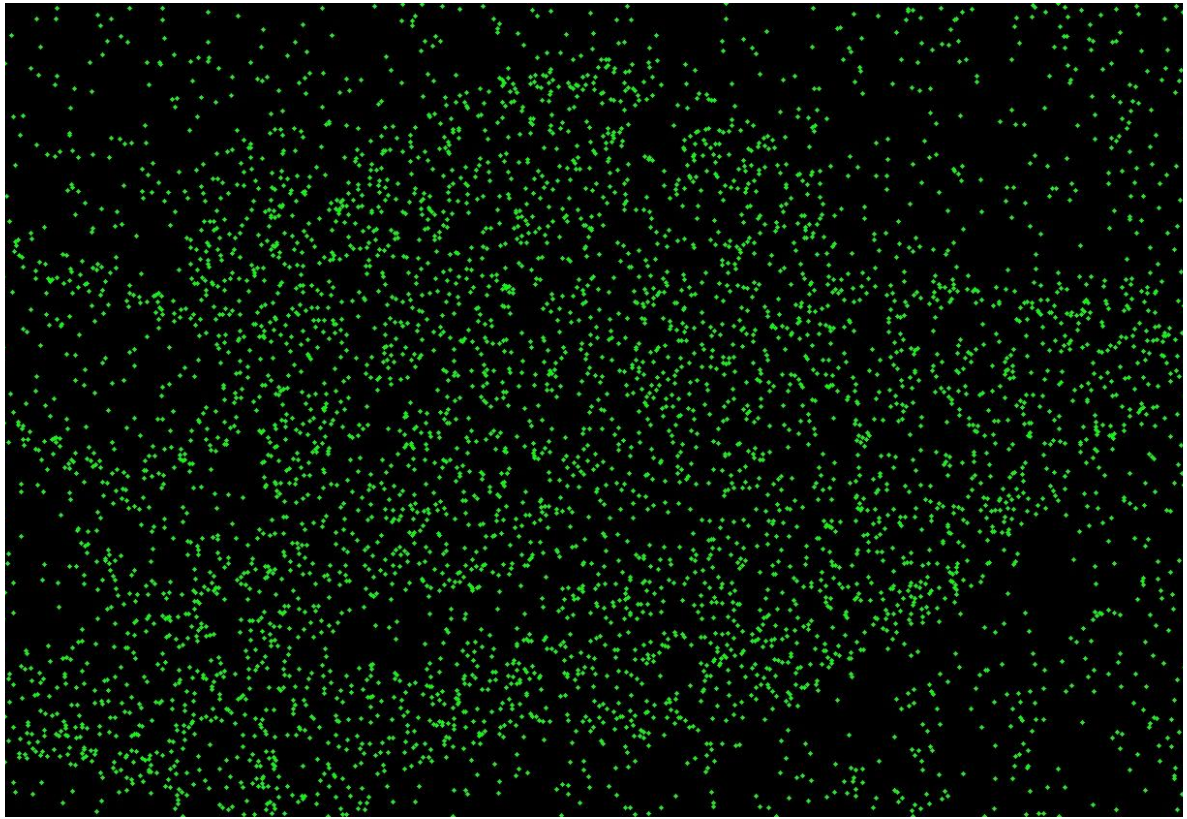
1000 pontos



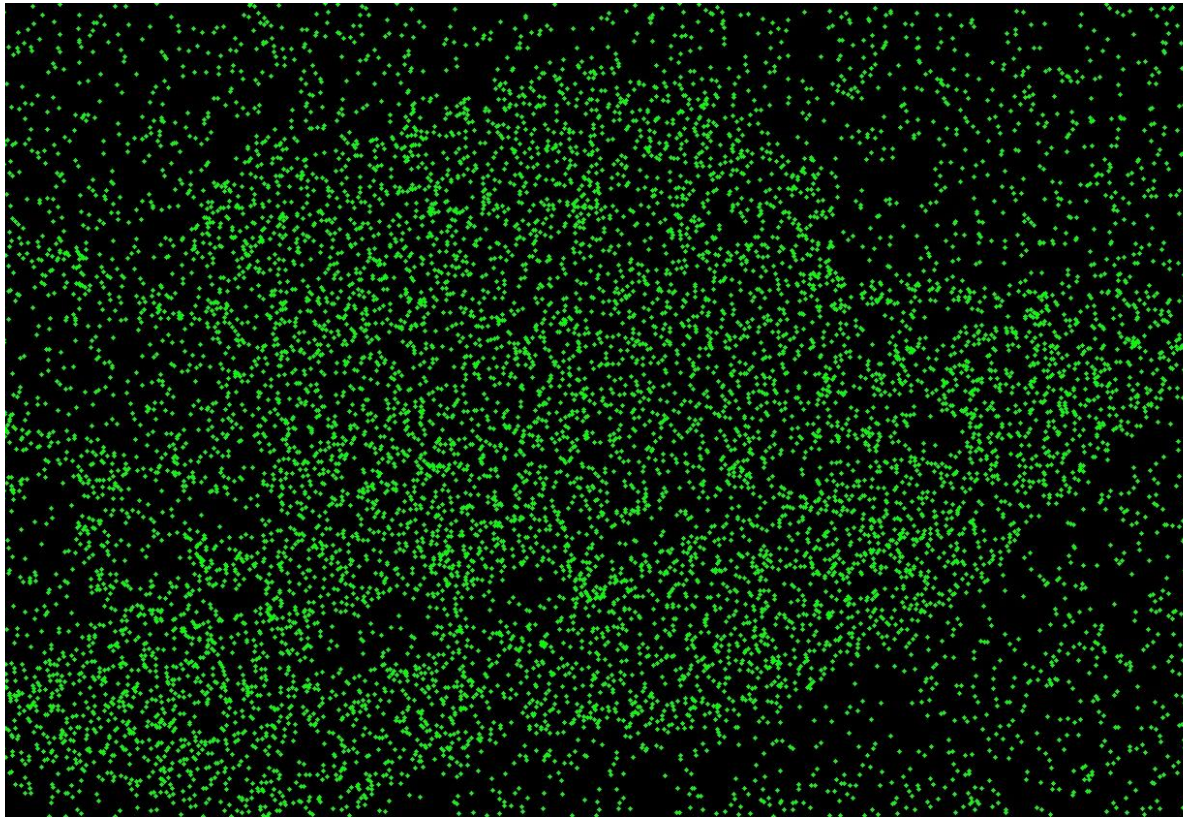
3000 pontos



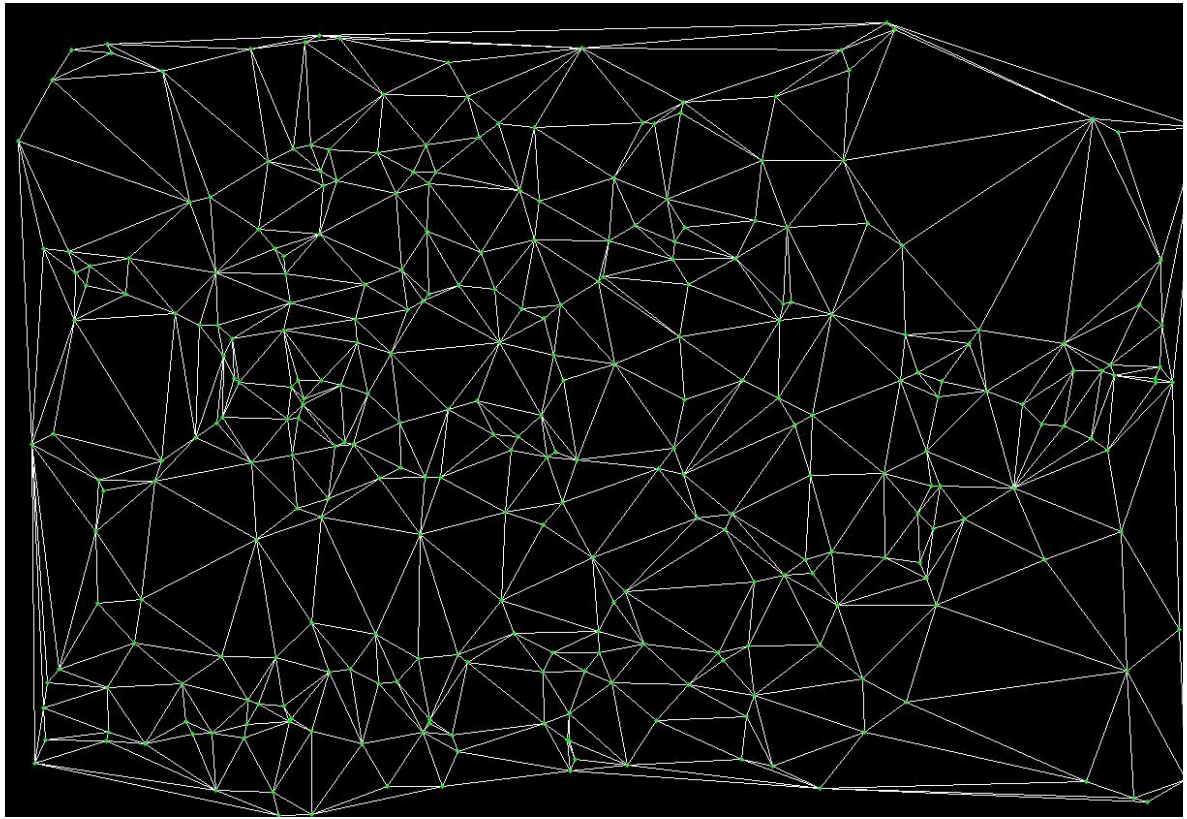
5000 pontos



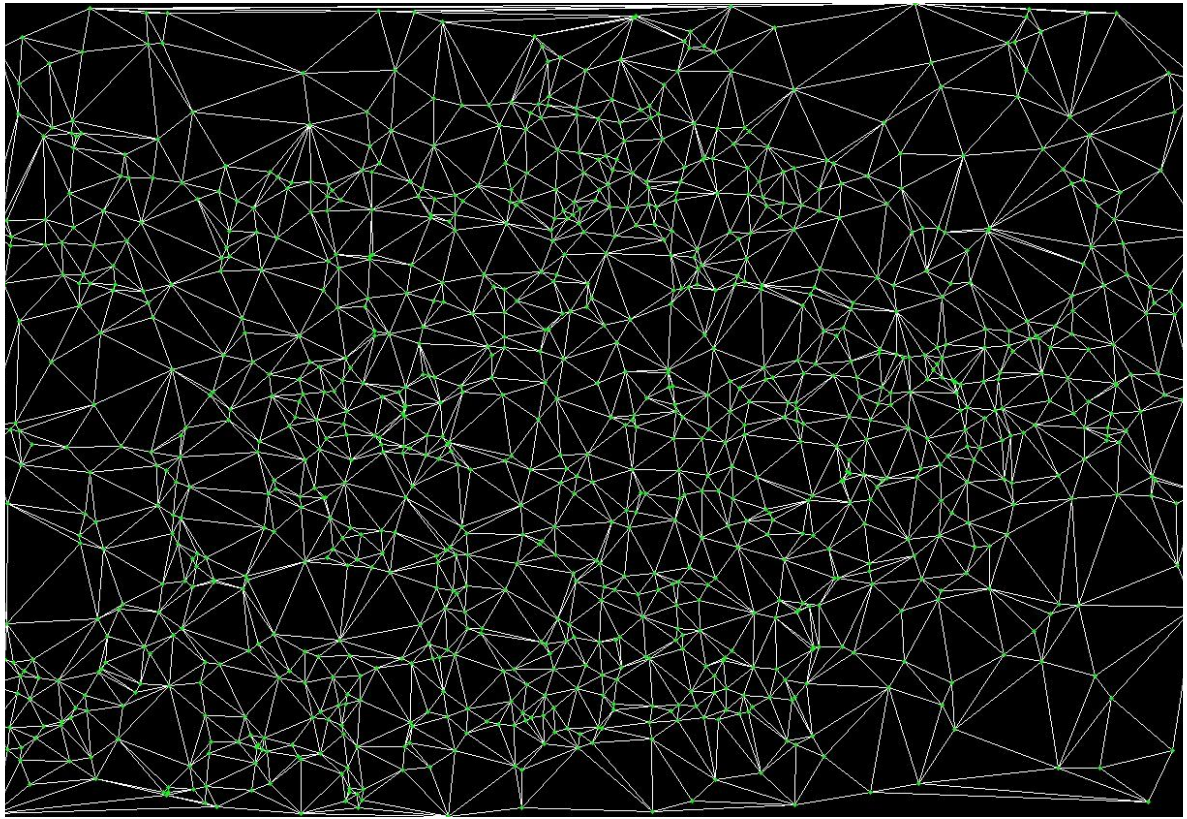
10000 pontos



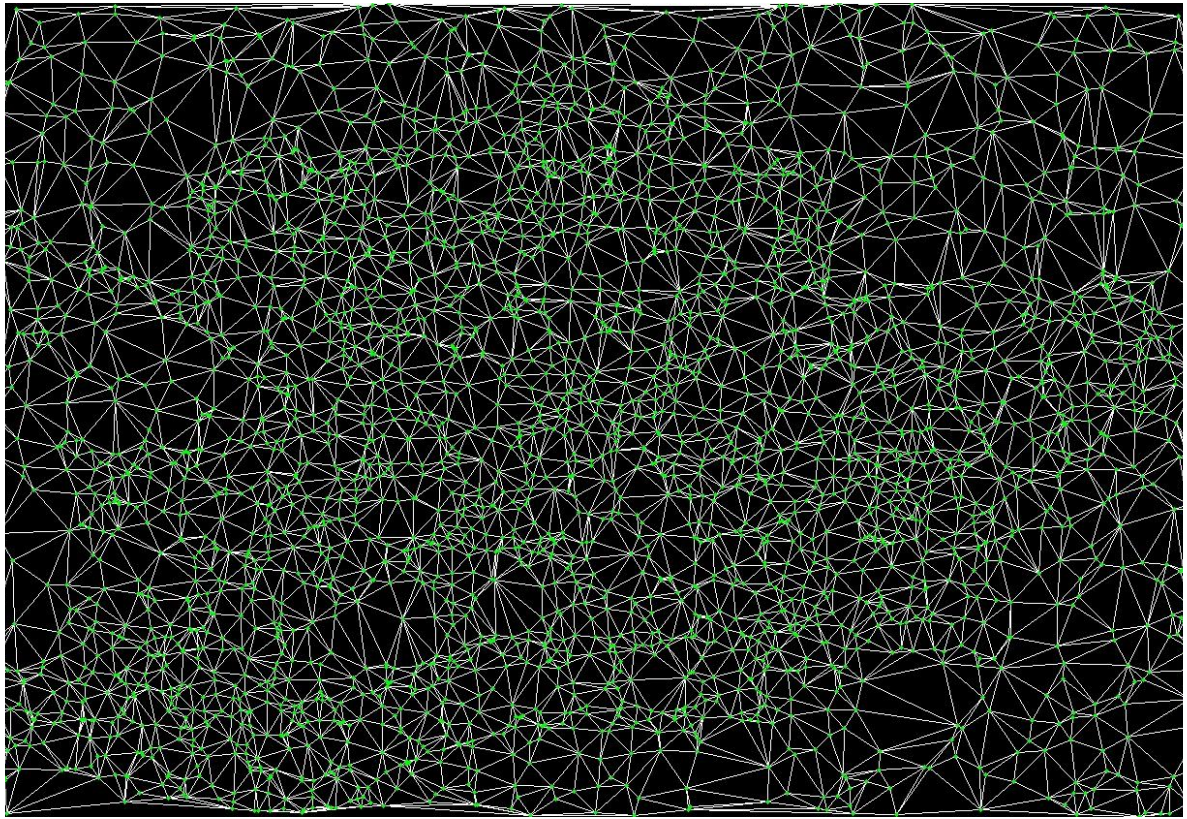
Delaunay - 300 pontos



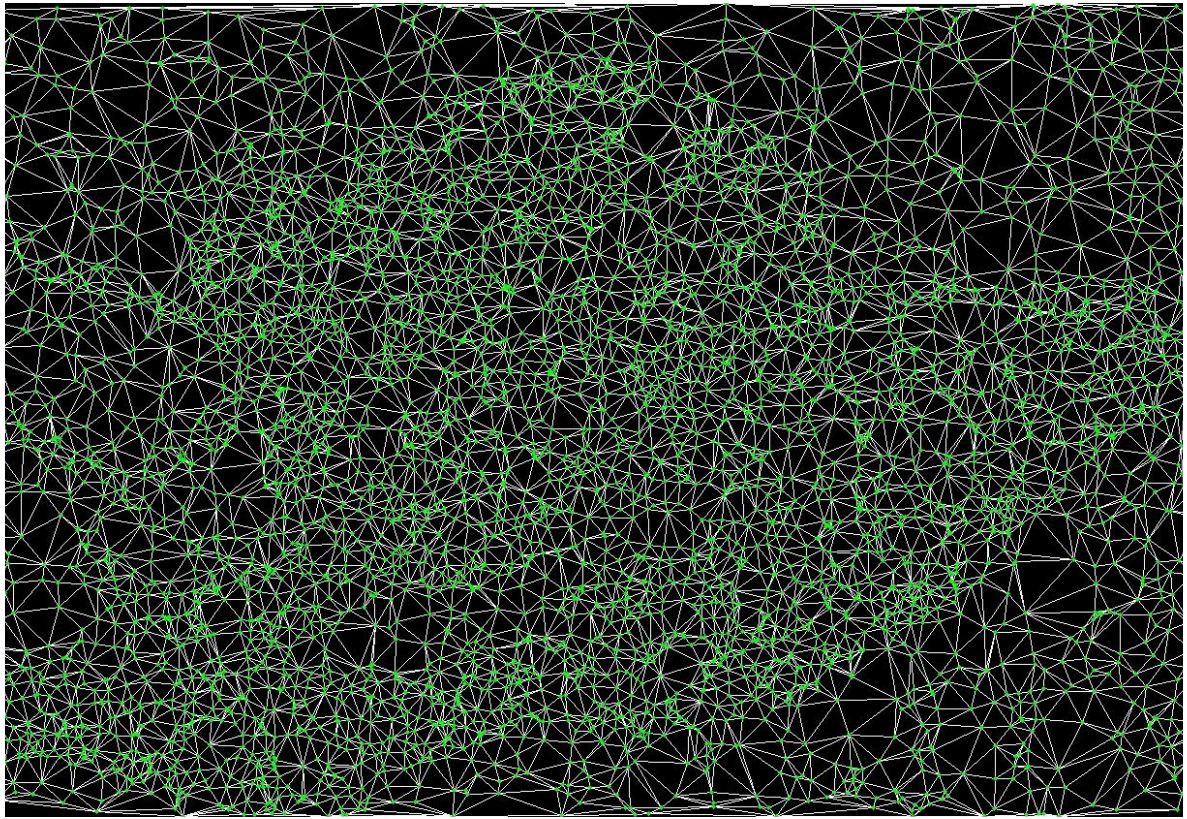
Delaunay - 1000 pontos



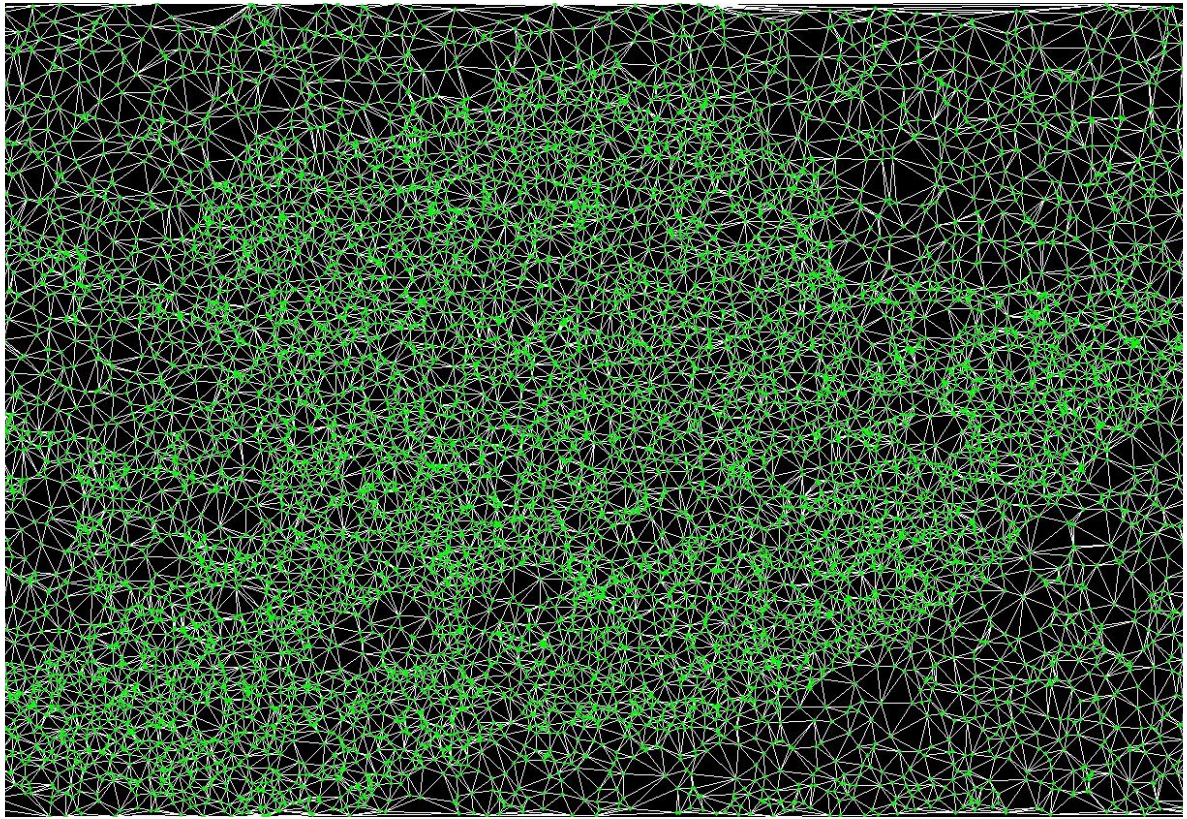
Delaunay - 3000 pontos



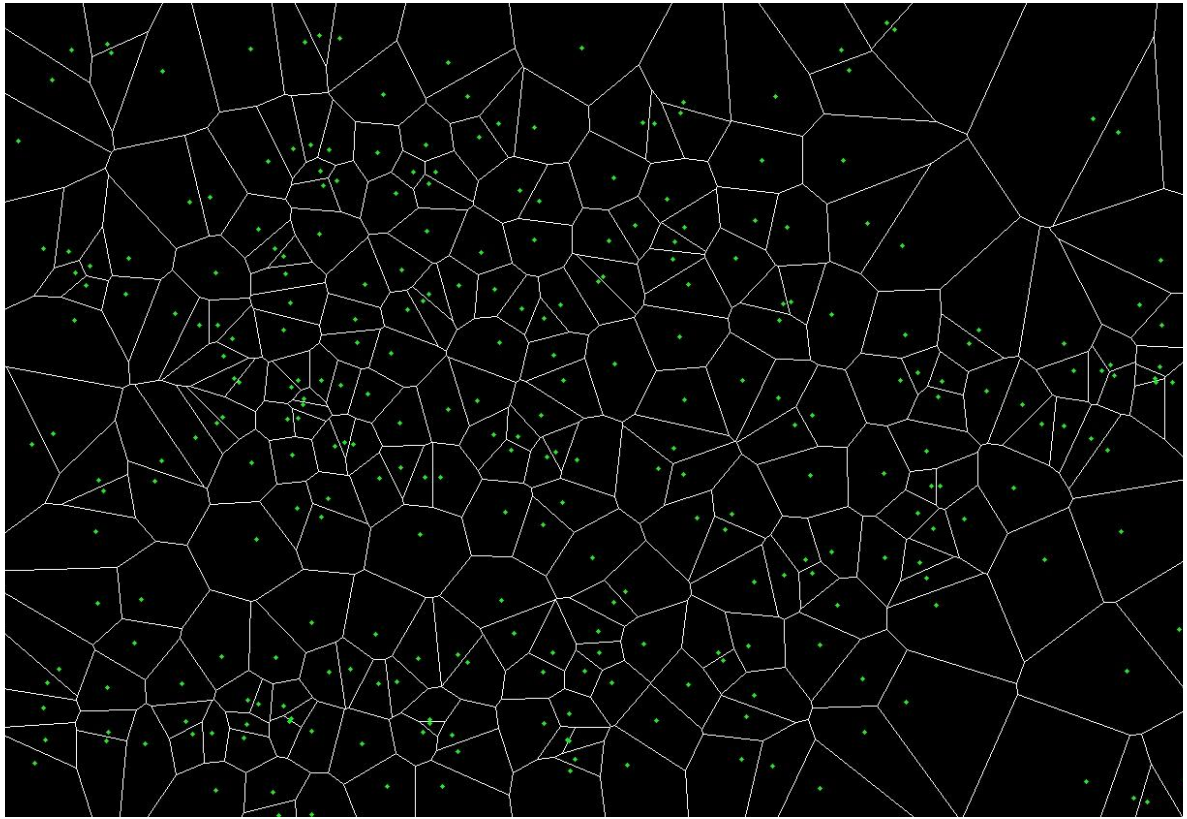
Delaunay - 5000 pontos



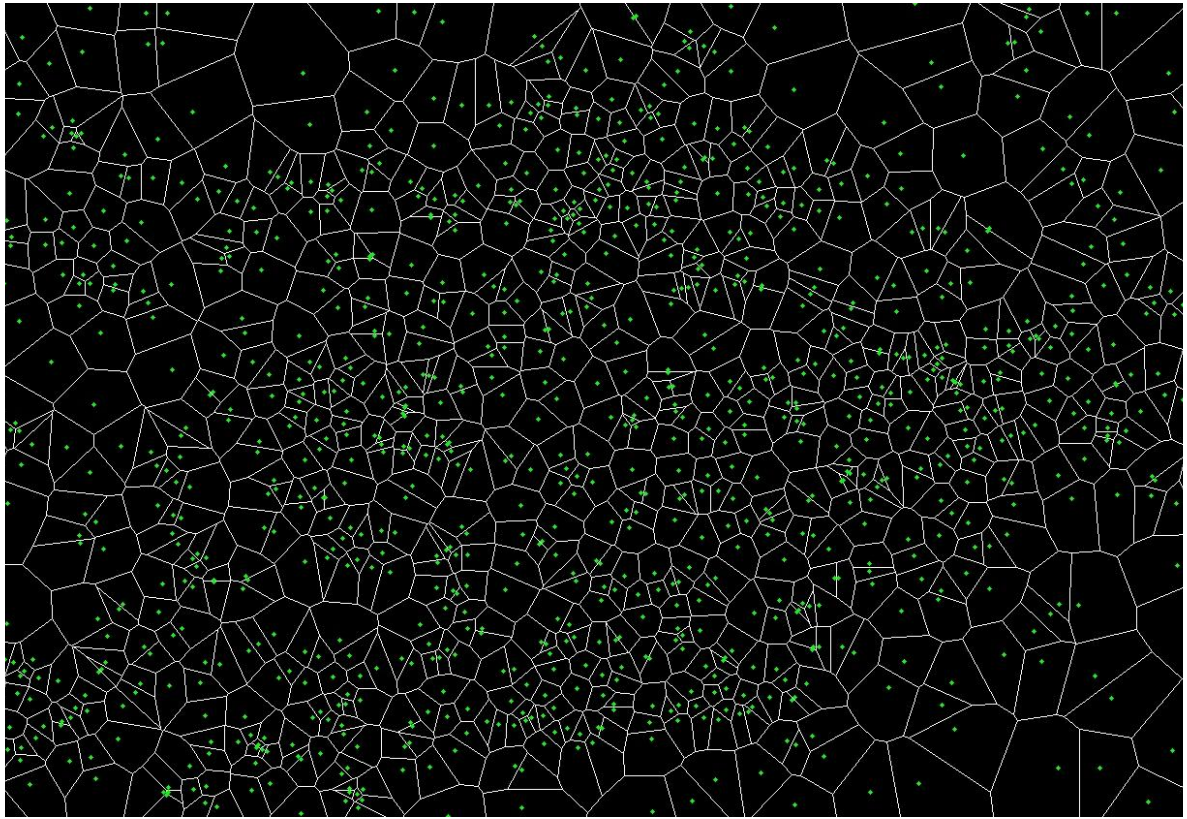
Delaunay - 10000 pontos



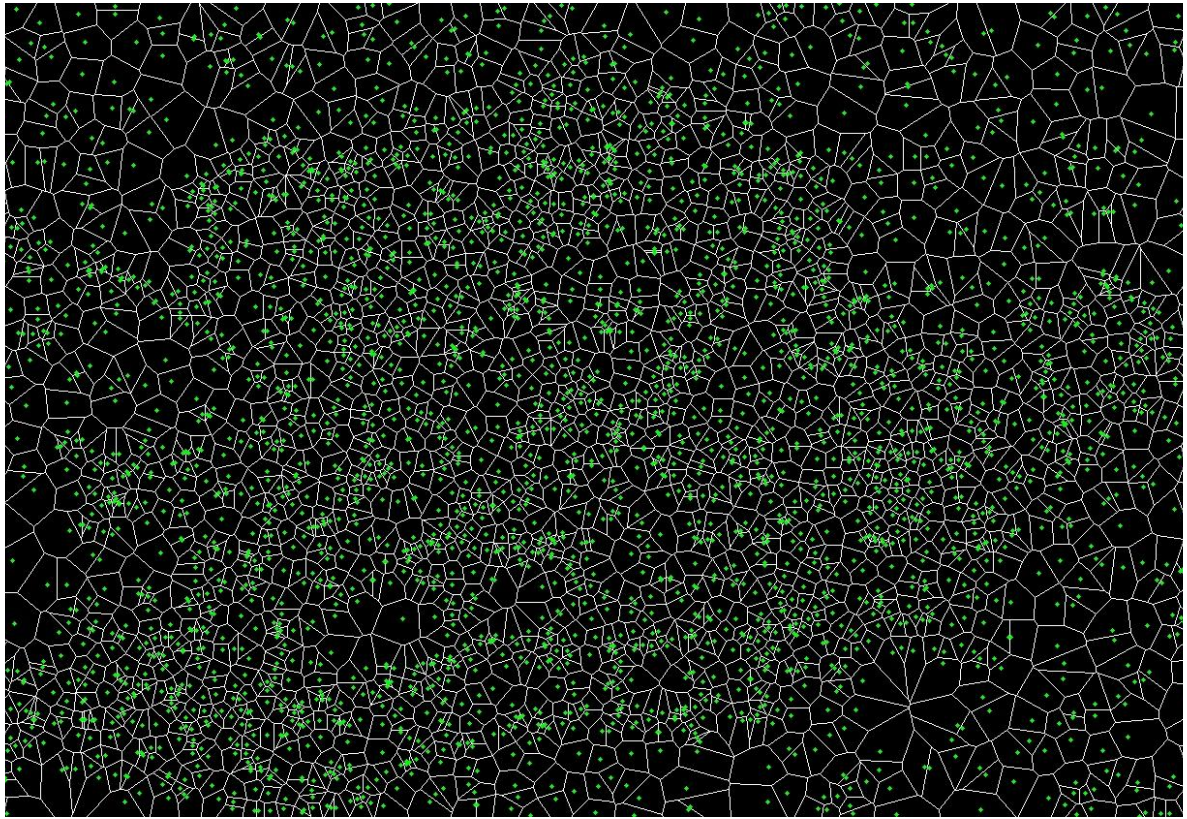
Voronoi - 300 pontos



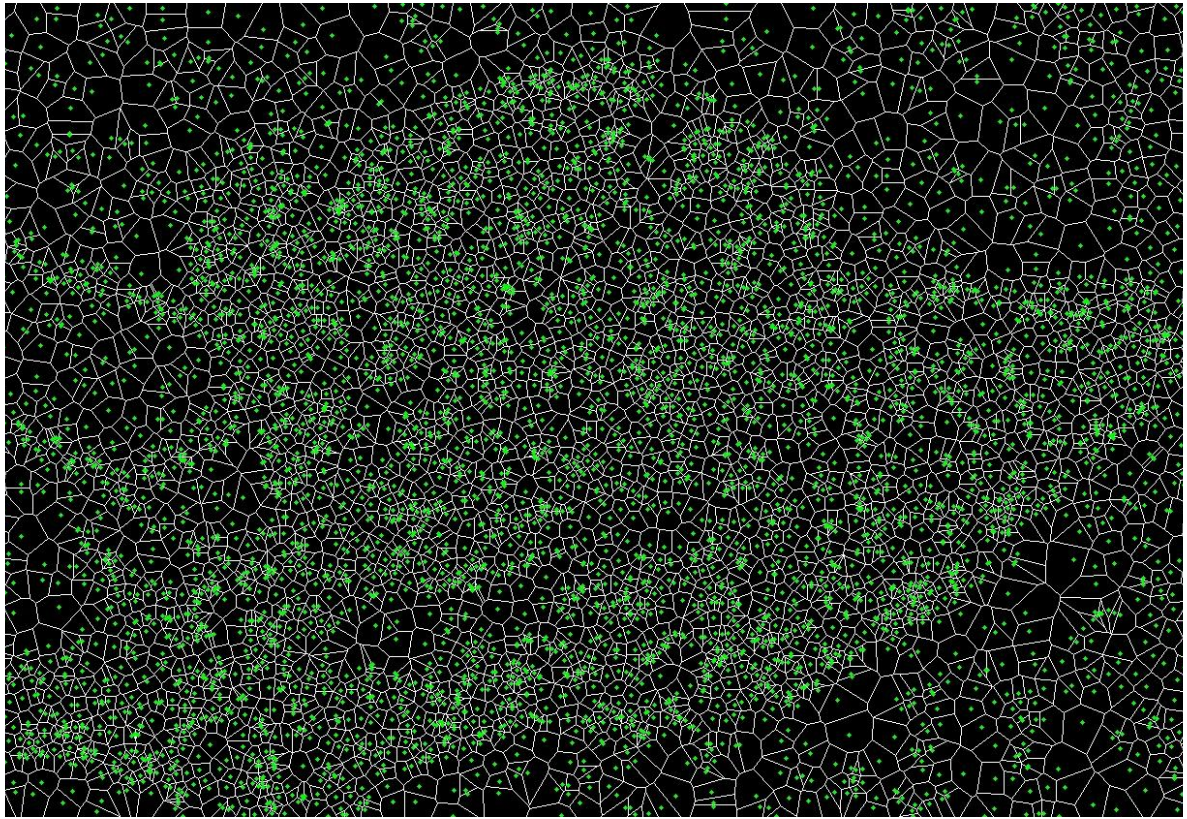
Voronoi - 1000 pontos



Voronoi - 3000 pontos



Voronoi - 5000 pontos



Voronoi - 10000 pontos

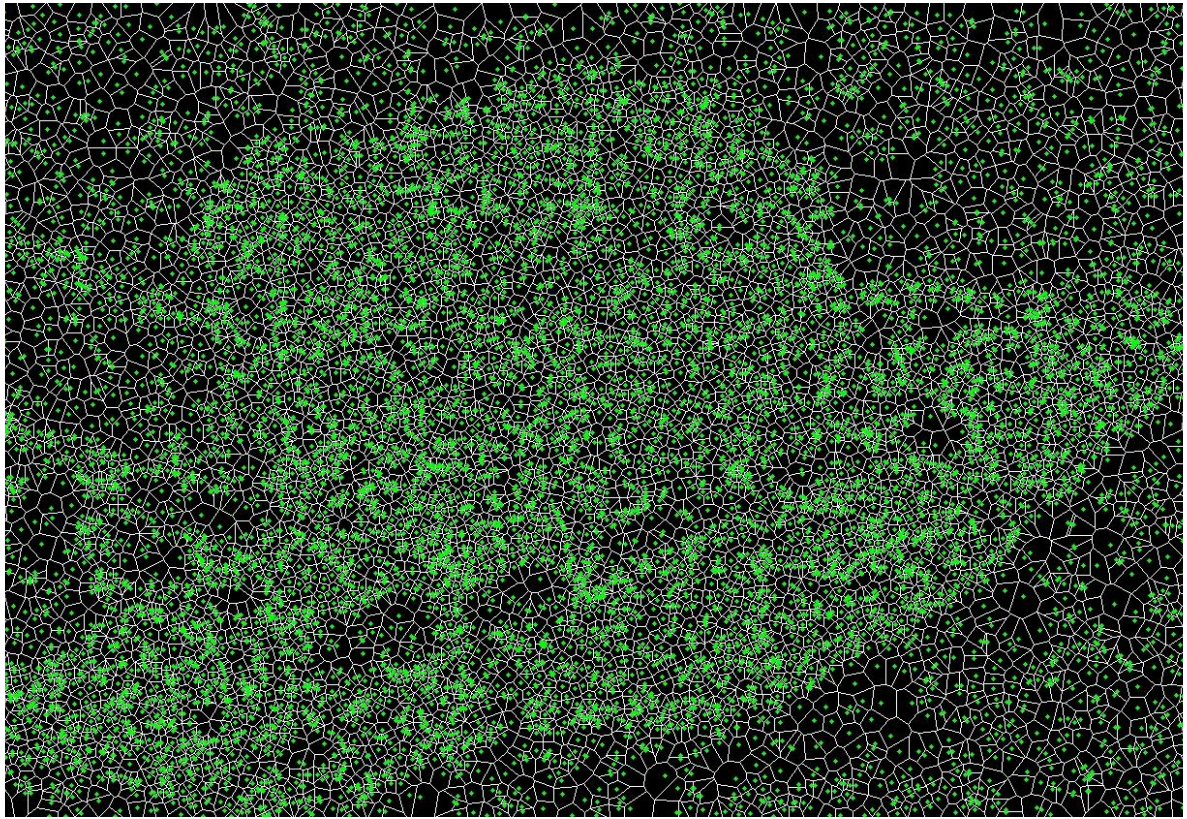


Imagem final - 300 pontos

Imagem final - 1000 pontos

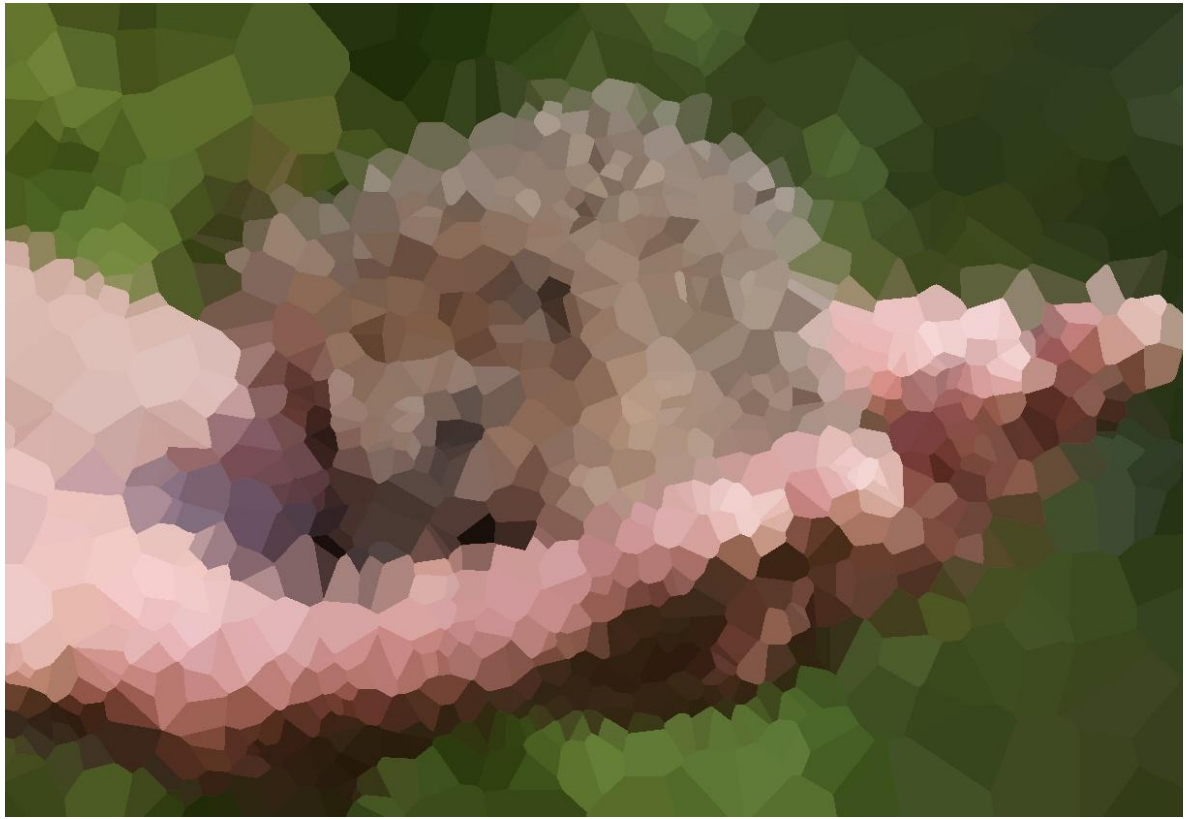


Imagem final - 3000 pontos

Imagem final - 5000 pontos

Imagem final - 10000 pontos

Problemas

- Muitos pontos juntos geram células muito pequenas em que o floodfill não funciona corretamente gerando áreas pretas na imagem final

- Como os pontos são aleatórios (mesmo que parcialmente) podem existir 3 pontos colineares que resultam em divisão por zero no cálculo do circuncentro

Perguntas ?