Star generation

Density function *dens(x, y)* is initially defined in every point of the map. The values must be floating point numbers between 0 and 1 inclusive. The value close to 0 means the star is unlikely to appear at this point. The value close to 1 means this point is a good candidate for star placing. The initial values can define the shape and star density of the generated map.

An attempt to place a new star is performed by generating 3 uniform random values:

$$0 \le X \le \text{map width}$$
$$0 \le Y \le \text{map height}$$
$$0 \le Z \le 1$$

If $Z < dens(X, Y)$ then we place a new star at $(X; Y)$.

When placing a new star the function *dens(x, y)* is changed.

Updating density function after star placing

The first thing that should be defined is the distance function between two points *(x1, y1)* and *(x2, y2)*. The default is Euclidean distance:

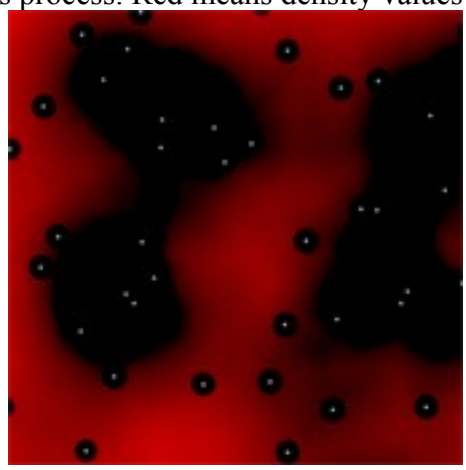
$$d(x1, y1, x2, y2) = \sqrt{(x1-x2)^2+(y1-y2)^2}$$

Then another function *f(d)* is defined which takes as an argument the distance between two points and returns the amount of the density that should be subtracted from the current point if a new star is placed at the specified distance away from this point.

When the star is placed at *(x0,y0)* the density function is updated for every point of the map:

$$dens(x, y) = dens(x, y) - f(d(x0, y0, x, y))$$

Here is the visualization of this process. Red means density values close to 1, black – close to 0.



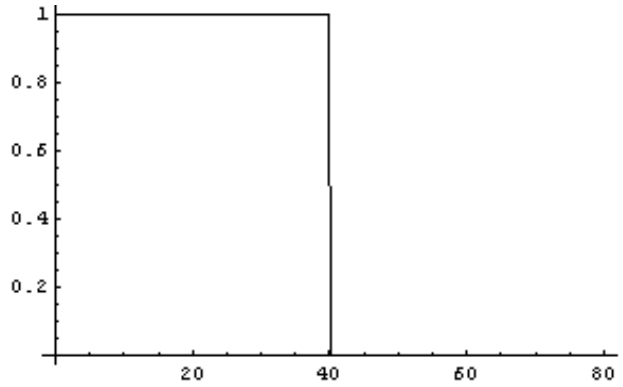
Different examples of f(d) with the resultant maps follow.

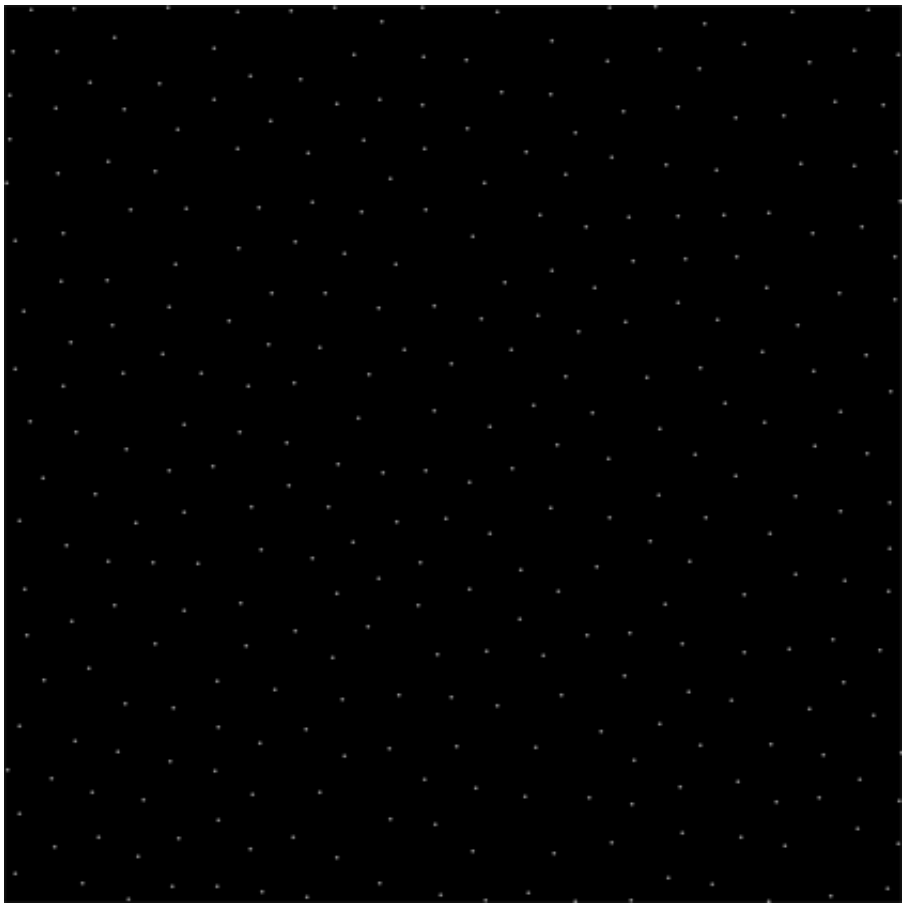
Basic uniform placement

Here is a boring uniform placement. We just avoid placing stars too close:

$$f(d)=\begin{cases}1 & \text{if } d<40\\ 0 & \text{if } 40\le d\end{cases}$$

So after placing the star the circle of radius 40 around it will become dead zone – no stars can be placed here any more.

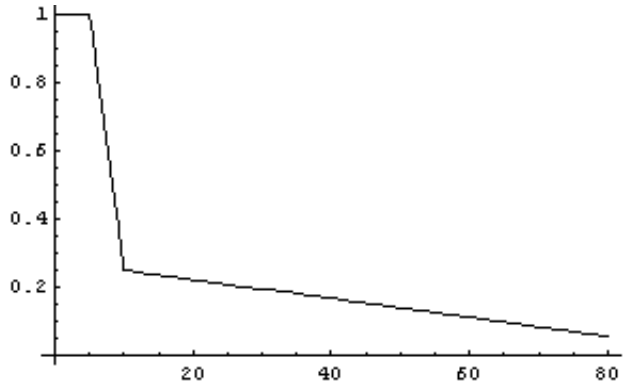


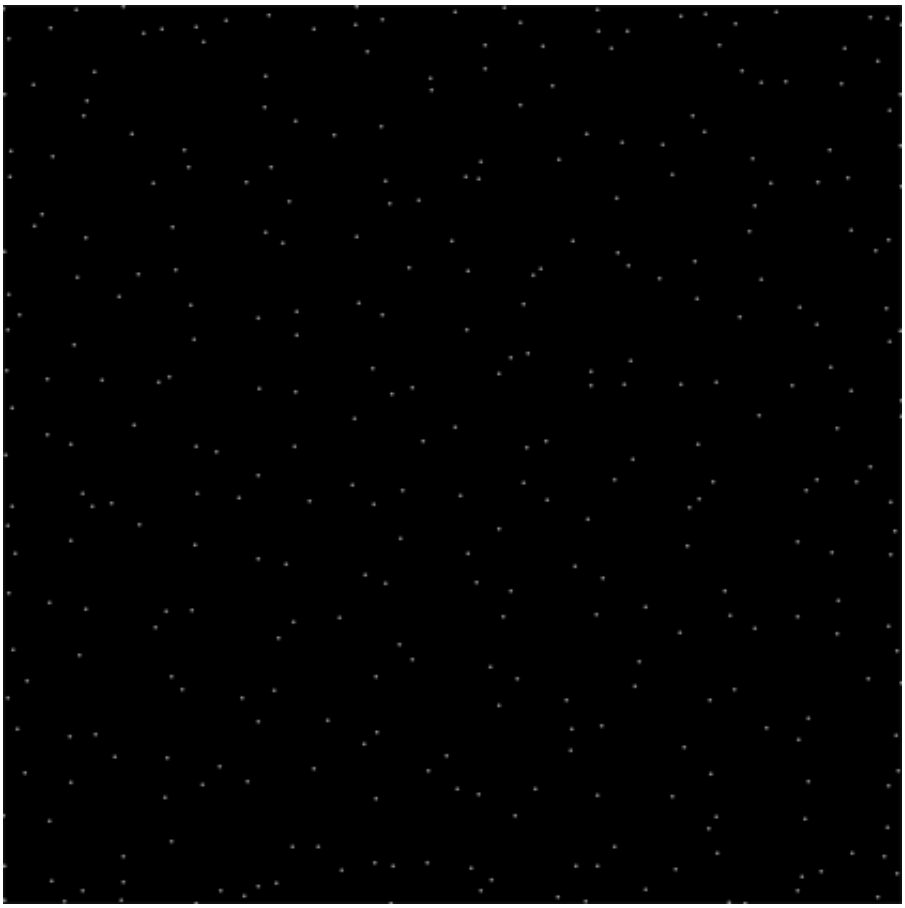


Giving more freedom

In this example stars not only define the dead zone around them but also interact with each other making some areas between them dead as well.

$$f(d)=\begin{cases} 1 & \text{if } d<5 \\ \frac{d-5}{20}+\frac{10-d}{5} & \text{if } 5\le d<10 \\ \frac{100-d}{360} & \text{if } 10\le d<100 \\ 0 & \text{if } 100\le d \end{cases}$$

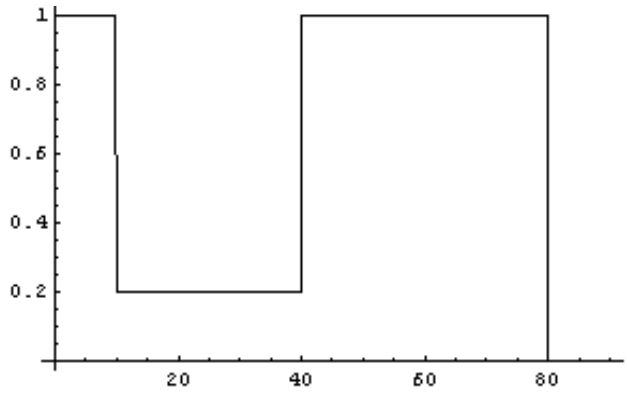


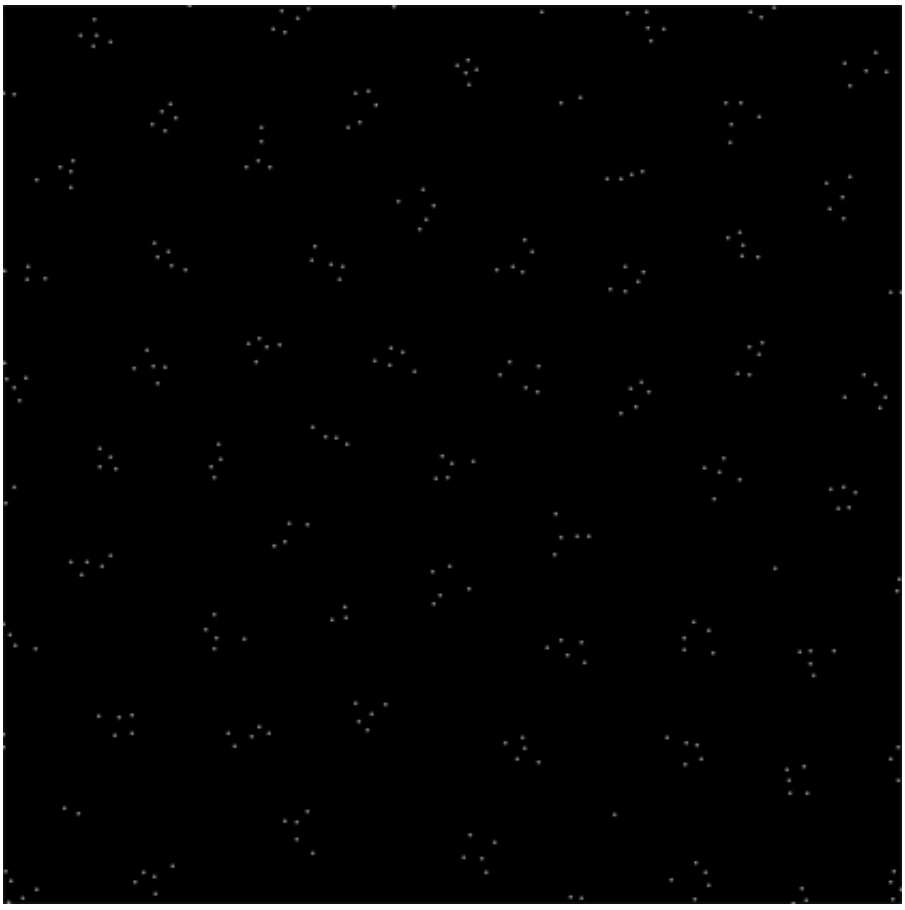


Clumping stars strictly

In this example we want any pair of stars be either close to each other or be distant. No middle distance. It is like the first example but with some “holes” for clumping stars.
The subtracted value for clumping area is 0.2. That is why the maximum number of stars in a cluster is 5.

$$f(d)=\begin{cases} 1 & \text{if } d<10 \\ 0.2 & \text{if } 10\le d<40 \\ 1 & \text{if } 40\le d<80 \\ 0 & \text{if } 80\le d \end{cases}$$

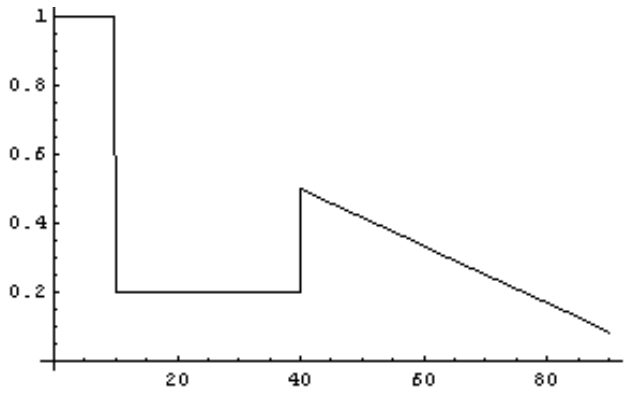


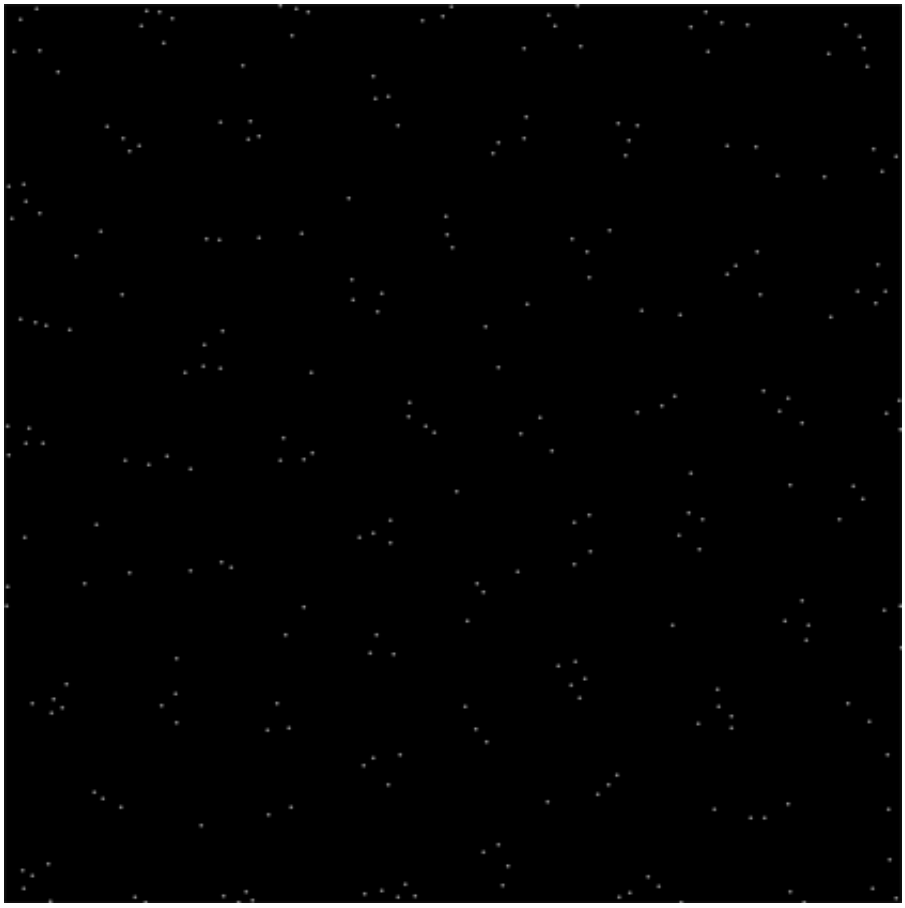


Clumping with freedom

Let's not force the stars be clumped but just advice them to do so. The clusters can influence each other, that is why their structure varies greatly.

$$f(d)=\begin{cases} 1 & \text{if } d<10 \\ 0.2 & \text{if } 10\le d<40 \\ \dfrac{100-d}{120} & \text{if } 40\le d<100 \\ 0 & \text{if } 100\le d \end{cases}$$





Additional notes

A slight drawback of this approach is that more stars are generated near the edges and corners of the map than in the center. So the initial values of density function should be adjusted to solve this problem.

This approach implies heavy processor work. Stars generation process stops when the number of failed attempts in a row exceeds the defined threshold. The more the value of threshold, the more complete will be map. But the amount of calculation performed will increase greatly. The value of 5000 seems to produce almost complete maps in a reasonable time.