Regosketch 1 : Simulate Regolith Agglutinate Grains

Mick Goulish
18 March, 2023

Overview

I wrote a Python program to try to simulate 2D regolith agglutinate grains. The results aren't going to fool anybody, but they're interesting. I think they show that this kind of approach could be improved to more closely simulate such grains.

Doing the Simulation

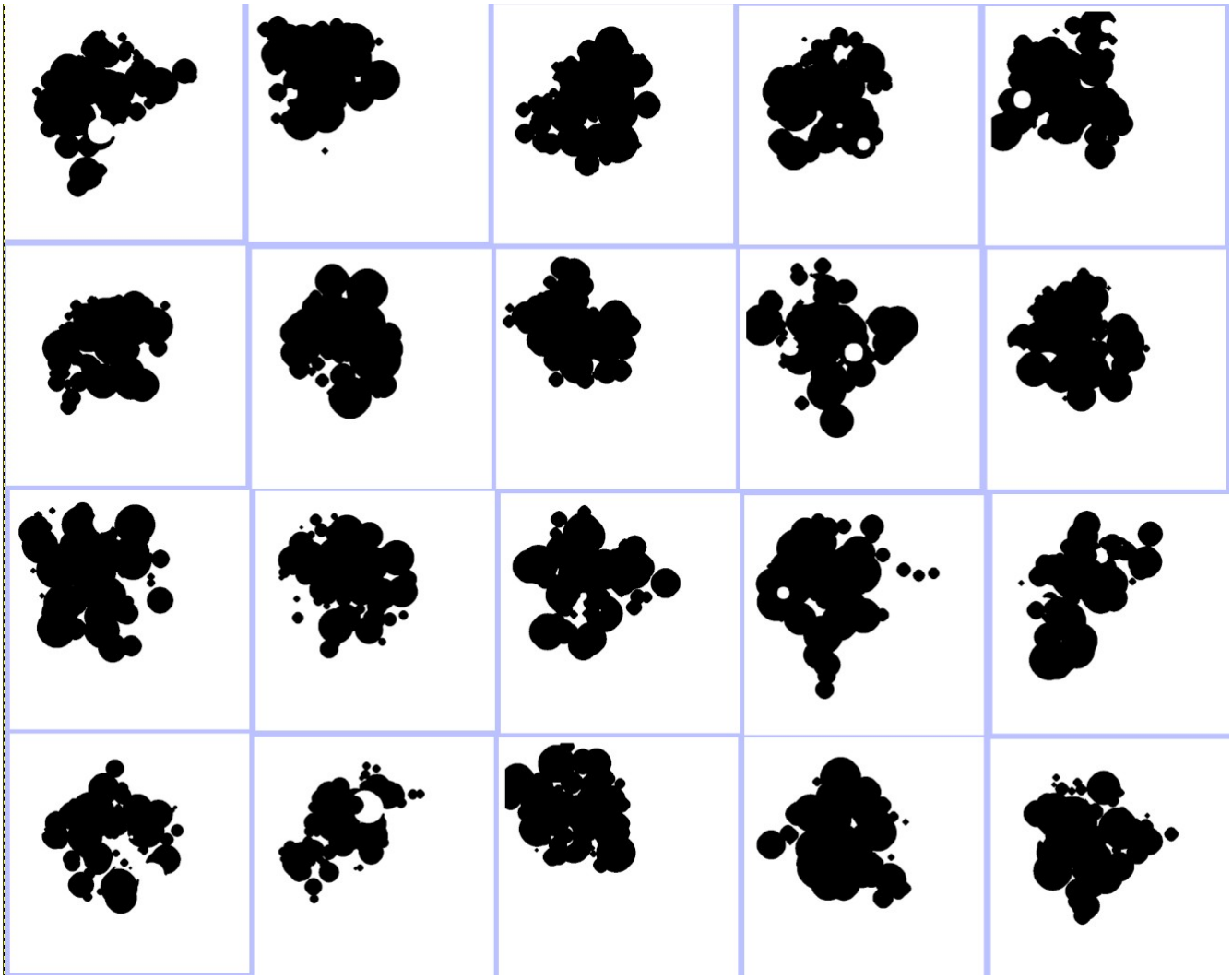
I simulated the agglutinate grains with the following steps:

- The agglutinate grain is simulated as a bunch of black, and an occasional white circle drawn on a white image. The resultant image is saved at the end. This is done in a loop, to make many 'grain' images.
- Build up the agglutinate grains with many circular 'particles'.
- The size of the circular particles are random, with an exponential distribution, with minimum and maximum cutoffs. The exponential distribution causes small grains to be more numerous than large.
- New circular particles are added to the growing agglutinate this way:
    - Find the edge image of the current agglutinate. Only pixels on the perimeter of the growing shape will be nonzero. (But the perimeter is pretty complex.)
    - Make an array of all the (x,y) coordinates of those edge pixels.
    - Choose one (x,y) pair randomly from that array. This will be the centerpoint of the next circular 'particle' to be added.
    - Choose a random radius for the new circle. The radii were created with an exponential distribution, with high and low cutoffs. Small is more common than large.
    - Add the new circle to the drawing.
    - Except – 5% of the time, make the circle white instead of black. This is like taking a bite out of the growing shape.

How Well Did It Work?

Well, I am not expecting a Nobel Prize, but I do think it shows some promise.

Here are 20 results.



How Do I Run It?

I'm including the code at the end of this paper, but it is also available at:
https://github.com/mgoulish/moondust

To get it, do this: git clone https://github.com/mgoulish/moondust

Then go into the directory it made, make sure the Python file is executable, and execute it.

The Code

Here is the actual code, in case this is the only way you can get it.

```
#! /usr/bin/env python3

# If you don't have the skimage package, do this:
#       pip install scikit_image
# For all other packages, it is just:
#       pip install PACKAGE_NAME
#
import sys
import math
import time
import numpy as np
import matplotlib.pyplot as mpl
from random import *
from skimage.draw import (disk, circle_perimeter)
from skimage import (filters, morphology)


#-------------------------------------
# Given an edge-image, return a list
# of all edge points.

#-------------------------------------
def get_edge_points ( img ) :
  edge_points = []
  for y in range(img.shape[0]) :
    for x in range(img.shape[1]) :
      if img[y][x] > 0 :
        #print ( f" edge point: {img[y][x]}" )
        edge_points.append ( (x, y) )
  return edge_points



#-------------------------------------
# The first particle that starts off
# the regolith grain has its location
# and size just set directly from the
# caller of this fn.
#-------------------------------------
def add_first_particle ( x, y, r ) :
  xs, ys = disk((x, y), r, shape=image.shape)
  image[xs, ys] = particle_color



# Add a particle (all but the first one)
```

```
def add_particle ( radii, edge_points ) :
  n_edge_points = len(edge_points)
  if n_edge_points < 1 :
    print ( f"add_particle: too few edge points: {n_edge_points}" )
    return

  # Choose from the array of radii at random.
  radius = radii [ randint(0, len(radii) - 1) ]

  # Choose the new circle's center randomly
  # from the array of the current edge points
  # of the shape.
  center = edge_points [ randint(0, n_edge_points-1) ]
  xs, ys = disk(center, radius, shape=image.shape)

  # Once in a while, make the particle ‘negative’.
  color = particle_color
  if randint(1, 100) > (100 - negative_particle_percent) :
    print ( "Negative Particle!" )
    color = background_color

  # Draw the particle.
  image[xs, ys] = color



def make_regolith_grain ( image_size,
                          image,
                          image_file_name,
                          radii,
                          n_particles ) :
  # First particle just goes at center.
  # There are as yet no edge points to use.
  add_first_particle ( image_size/3, image_size/3, 60 )
  edge_img = filters.roberts(image)
  edge_points = get_edge_points ( edge_img )
  n_edge_points = len(edge_points)

  #-------------------------------------------------------
  # Add circular particles to grow the regolith grain.
  #-------------------------------------------------------
  for p in range(n_particles - 1) :
    print ( f"Adding particle {p+1} of {n_particles}" )
    add_particle ( radii, edge_points )

    # Find all edge pixels. One of these will be
    # randomly selected to be the location for
    # the next particle.
    edge_img = filters.roberts(image)
    # If you want to see the edge image generated
    # during each particle addition, uncomment these
    # two lines.
    # You will have to hit 'q' to allow the program
    # to continue.
    # mpl.imshow ( edge_img )
    # mpl.show()
    edge_points = get_edge_points ( edge_img )
    n_edge_points = len(edge_points)
    #print ( f"particle: {p} edge points: {n_edge_points}" )
```

```
  #---------   End of particle-adding for-loop  --------

  mpl.imshow ( image )
  mpl.axis ( 'off' )
  mpl.savefig ( image_file_name, bbox_inches='tight' )
  # mpl.show()



# ================================================
# Main
# ================================================

mpl.rc('image', cmap='gray')
image_size = 600

negative_particle_percent = 5

n_regolith_grains = 20
n_particles       = 100  # How many particles per regolith grain.
particle_color    = 0    # black
background_color  = 1    # white

# Make a temporary array of radii, and then filter it
# to throw out radii that are too small to see on the
# image, and any that are overwhelmingly large.
# This is probably similar to something that happens
# in reality.
# The exponential distribution used here gets very steep
# toward x=0. So smaller particles will greatly outnumber
# larger.
temp_radii = 1000
mean_radius = 3
temp_radii = np.random.exponential ( mean_radius, temp_radii )

# Scale up the radii so they're visible in a pixelated image.
temp_radii = 10 * temp_radii
radii = []
radii = [r for r in temp_radii if 1 <= r and r <= 50]

# Uncomment these two lines if you want to see a display
# of the particle radius distribution.
# You will have to hit 'q' to quit the display and allow
# the rest of the program to continue.
#--------
# count, bins, ignored = mpl.hist(particle_radii, 14, density = True)
# mpl.show()

for n in range(n_regolith_grains) :
  image = np.ones((image_size, image_size), dtype=np.double)
  print ( f"\n\nMaking Regolith Grain {n} ======================\n" )
  make_regolith_grain ( image_size,
                        image,
                        f"result_{n}.png",
                        radii,
                        n_particles )
```