```
% Final Project: Stochastic Simulation of Integrin Diffusion and Activation
% in a 2D Plasma Membrane

% Clears and closes any data/plots to make it a blank slate
clear all
close all
clc

% Variable Initialization

D = 100; % diameter, units in nm
R = D/2; % radius, units in nm

N = 10; % number of integrins
T = 500; % maximum number of steps
Pa = rand(); % , activation probability, random value between 0 and 1
F = randi([1,10]); % force value, random integer between 1 and 10
Pub = 1/(1.5*F); % unbinding probability
k = 0;

% Calling initialiation function for integrin generation, position, and
% state
integrin = initIntegrin(R,N);
initialIntegrin = integrin;

% Initial x and y coordinates
intial.x = integrin(1).x;
intial.y = integrin(1).y;


% Simulation
for t = 1:T
    % Calling position and displacement function for each integrin
    integrin = disPosition(integrin,R);

    % Activation and ligand binding --> First State
    integrin = stateActandLig(integrin,Pa);

    % Ligand binding and deactivation ---> Second State
    integrin = stateLigandDeact(integrin,Pub);

    if (rem(t,20) == 0) % saves a fr
        % ame every iteration
        k = k + 1; % counter

        for n = 1:N
            if (integrin(n).activation_state == 1)
                scatter(integrin(n).x, integrin(n).y, 'r', 'filled')
                hold on
                % this will make the integrin red when its activation state
                % is 1
            else
                scatter(integrin(n).x, integrin(n).y, 'g', 'filled')
                hold on
                % this will make the integrin green when its activation
                % state is not 1 (most likely 0)
            end
        end
        circle(R) % calls the circle function (2D circular domain)

        M(k) = getframe; % uses the frames from the iterations
        hold off
        pause(1) % pauses 1 second each frame
   end
end

% makes the movie from the frames above
movie(M)

% Analysis

displace = zeros(N,T); % empty displacement vector to hold values

% for loop to account for all the integrins and iterations
for n = 1:N
    for t = 1:T
        % finds the displacement between each x and y and their intial
        % values
        dx = integrin(n).x(t) - initial(n).x;
        dy = integrin(n).y(t) - initial(n).y;
        % finds the displacement vector value
        displace(n,t) = sqrt(dx.^2 + dy.^2);

    end
end

% calculate the root mean squared displacement of each integrin
rmsd = sqrt(mean(displace.^2,2));
```