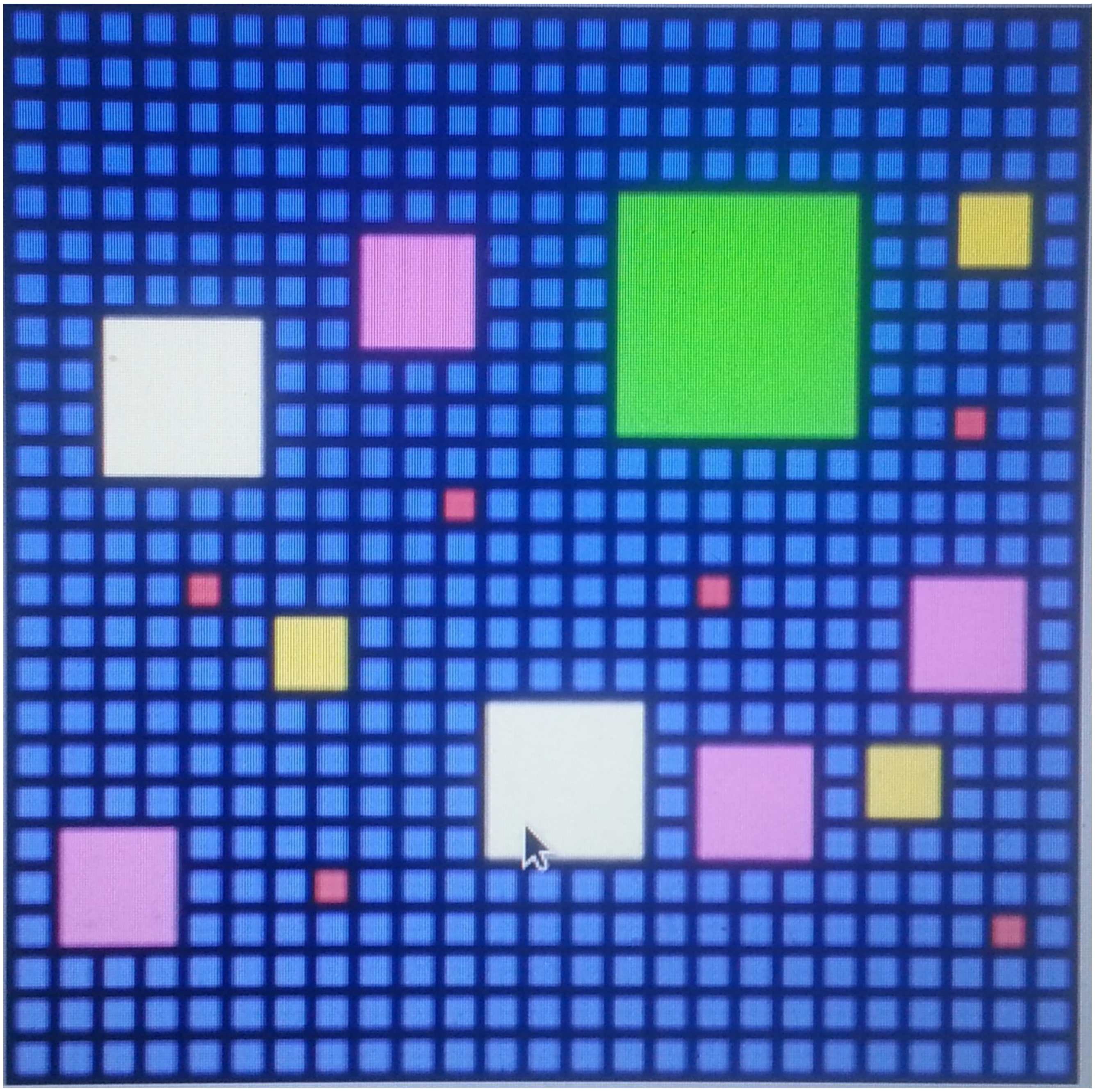
```matlab
%% advanced matlab final project , main

% simon teshuva, 302207220
% last updated 17/6/2014

% this project attempts to model the effects of natural disasters hitting a
% city. it models tornados, tsunamis and earthquakes.

% this is the main m file. from it the type of disaster is chosen. after
% that the relevatnt function is called to determine additional information
% about the properties of the disaster such as its path, starting location
% and strength. a rgb image of the city before and after the disaster is
% created and are then montaged together. the total financial damgade done
% by the disaster is also calculated.

% each building has a color code. as the building gets more damaged it gets
% progressiely darker.
%                    fine            damaged                 destroyed
% road              0 0 0              0 0 0                    0 0 0
% house            100 100 255       75 75 190              40 40 100
% petrol station  255 100 100       190 75 75              100 40 40
% shopping centre 255 255 255      200 200 200            100 100 100
% university       100 255 100     75 190 75                  40 100 40

% park            255 100 255       190 75 190                100 40 100
% supermarket     255 255 100        190 190 75               100 100 40

clear all;
clc;

%% initialising all variables


map = load('mapGrid.txt');
buildingIDs = load('mapIDsModded.txt');

% create an image to display to user
% dimensions of image = sizeShownImage * sizeShownImage
sizeShownImage = 101;
% call the method to create the image
shownImage = createStartImage(map);
% resize the image by a factor of 3
shownImage= imresize(shownImage, [sizeShownImage*3 sizeShownImage*3]);
% display to user
imshow(shownImage);


% information for user
display('color code for buildings;');
display('black : road');
display('blue : house');
display('orange : petrol station');
display('white : shopping center');
display('green : university');
display('pink : park');
display('yellow : supermarket');
```

```
%% re-assign IDs to sensible values

% many of the buildings have IDs that are outside of the range of 1 - 505;
% this set of loops and ifs bring all builings to inside that range to
% allow for easier computation
for xAxis = 1:length(buildingIDs)
    for yAxis = 1:length(buildingIDs)
        % roads = 999 -- > 1
        if buildingIDs(yAxis, xAxis) == 999
            buildingIDs(yAxis, xAxis) = 1;
            % IDs ranging from 602 - 698 go to 2 - 98
        else if buildingIDs(yAxis, xAxis) > 600
                buildingIDs(yAxis, xAxis) = buildingIDs(yAxis, xAxis) - 600;
                % miscelaneous buildings to be adjested for the next 4
            else if buildingIDs(yAxis, xAxis) == 509
                    buildingIDs(yAxis, xAxis) = 200;
                else if buildingIDs(yAxis, xAxis) == 508
                        buildingIDs(yAxis, xAxis) = 328;
                    else if buildingIDs(yAxis, xAxis) == 507
                            buildingIDs(yAxis, xAxis) = 329;
                        else if buildingIDs(yAxis, xAxis) == 506
                                buildingIDs(yAxis, xAxis) = 330;
                            end
                        end
                    end
                end
            end

        end
    end
end

% a menu system for choosing what type of disaster
display('welcome to the main');
display('what type of event to you want to model?');
display('0: quit');
display('1: tornado');
display('2: tsunami');
display('3: earthquake');
choice = input('');
choice = round(choice);

while choice <0 || choice > 3
    display('invalid choice, try again');
    display('0: quit');
    display('1: tornado');
    display('2: tsunami');
    display('3: earthquake');
    choice = input('');
    choice = round(choice);
end

% if not quitting, do the main part of the algorithm
if choice ~= 0

    % call the appropriate setEvenData and simulateEvent functions
if choice == 1
    [entryPointY entryPointX strength yDir xDir] = setEventData1();
    [startRGBMap finishRGBMap damageArray] = simulateEvent1(entryPointY,
entryPointX, strength, yDir, xDir, map);
else if choice == 2
        [entryPointY entryPointX strength] = setEventData2();
        [startRGBMap finishRGBMap damageArray] = simulateEvent2(entryPointY,
entryPointX, strength, map);
    else if choice == 3
            [entryPointY entryPointX strength] = setEventData3();
            [startRGBMap finishRGBMap damageArray] = simulateEvent3(entryPointY,
entryPointX, strength, map);
        end
    end
end

% calculate the total finacial cost from the disaster
totalCost = calculateDamages(map, buildingIDs, damageArray);


% use the image processing toolbox to save the initial and final maps to
% computer, then scale them up by a factor of 5 and save to computer again.
% once this is done, montage the two images together to show the difference

size = 101;

startImage = startRGBMap;
finishImage = finishRGBMap;

imwrite(startImage, 'start.jpg');
imwrite(finishImage, 'finish.jpg');

start = imread('start.jpg');
finish = imread('finish.jpg');

% may need a 3rd dimension
scaledStart = imresize(start, [5*size 5*size]);
scaledFinish = imresize(finish, [5*size 5*size]);

imwrite(scaledStart, 'scaledStart.jpg');
imwrite(scaledFinish, 'scaledFinish.jpg');

finalStart = imread('scaledStart.jpg');
finalFinish = imread('scaledFinish.jpg');

% may be wrong
montage([finalStart finalFinish]);
display('the total damage caused in dollars was ');
display(totalCost);

% create a graph and then a tree out of our data
[damagedGraph damagedVector destroyedGraph destroyedVector] = createGraph(map,
damageArray, buildingIDs, entryPointX, entryPointY);
[examinedGraph names] = graphs(damagedGraph, damagedVector, destroyedGraph,
destroyedVector);
display('the IDs of the nodes in the graph are');
for i = 1:length(names)
    node = int2str(i);
    ID = int2str(names(i));
    display(strcat(node, ' = ', ID));
end
```

```
end
```

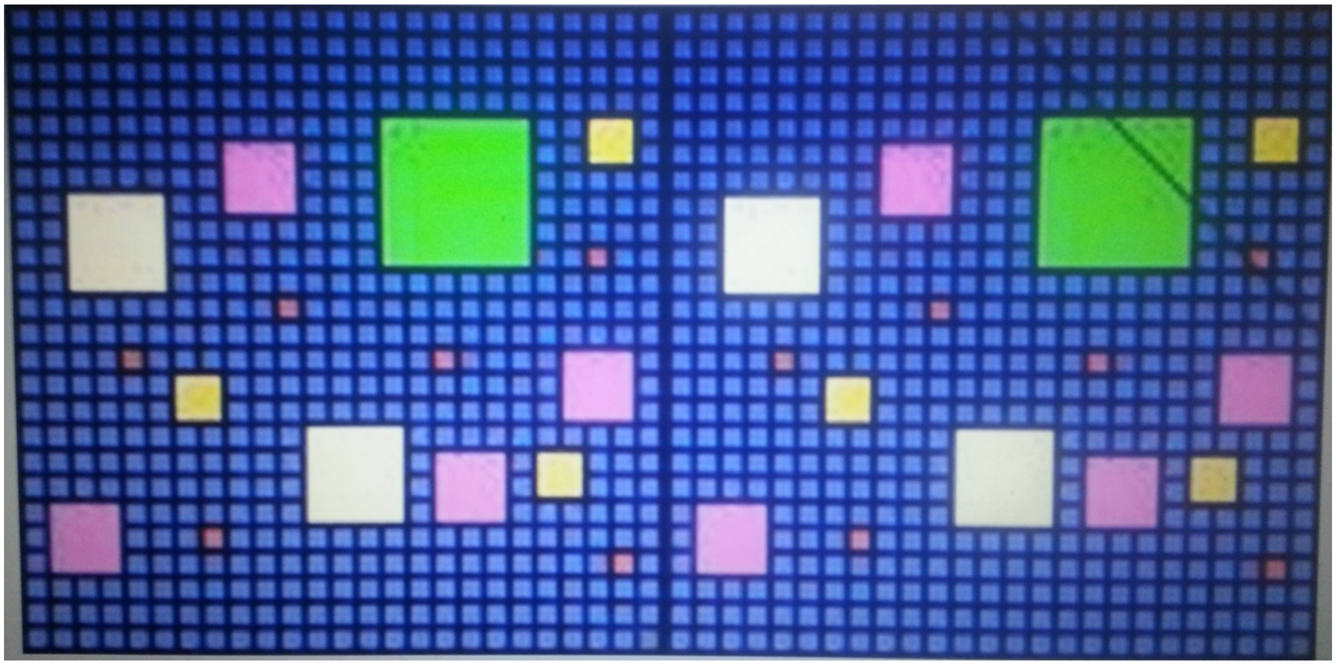


```
%% advanced matlab final project, setEventData1

% simon teshuva, 302207220
% last updated 17/6/2014

% this function sets the relevant data for a tornado

function [entryPointY entryPointX strength yDir xDir] = setEventData1()
%% set angle
% in this model, a tornado comes in from one of 8 directins, inputed from
% user
```

```matlab
display('from which direction is the tornado coming from?');
display('1: north');
display('2: north-east');
display('3: east');
display('4: south-east');
display('5: south');
display('6: south-west');
display('7: west');
display('8: north-west');
choice = input('');
choice = round(choice);

while choice < 1 || choice > 8
    display('invalid choice, try again');
    display('from which direction is the tornado coming from?');
    display('1: north');
    display('2: north-east');
    display('3: east');
    display('4: south-east');
    display('5: south');
    display('6: south-west');
    display('7: west');
    display('8: north-west');
    choice = input('');
    choice = round(choice);
end


%% set entry points

% using the user's choice, assing the starting location for the tornado

if choice == 1
    entryPointX = 51;
    entryPointY = 1;
else if choice == 2
        entryPointX = 101;
        entryPointY = 1;
    else if choice == 3
            entryPointX = 101;
            entryPointY = 51;
        else if choice == 4
                entryPointX = 101;
                entryPointY = 101;
            else if choice == 5
                    entryPointX = 51;
                    entryPointY = 101;
                else if choice == 6
                        entryPointX = 1;
                        entryPointY = 101;
                    else if choice == 7
                        entryPointX = 1;
                        entryPointY = 51;
                        else if choice == 8
                                entryPointX = 1;
                                entryPointY = 1;
                            end
                        end
                    end
                end
            end
        end
    end
end




%% set strength
    display('how strong is the tornado?');
    display('pick a value from 1 to 100');
    strength = input('');

    while strength < 0 || strength > 100
    display('how strong is the tornado?');
    display('pick a value from 1 to 100');
    strength= input('');
    end

%% set path
% calculate which direction the tornado is moving in. pick a point for the tornado
to move towards
% create a line from start in the direction that the tornado moves. re-set
% the point that the tornado moves towards as the last point in the map
% that the tornado passes through

    display('in what direction will the tornado move');
    display('set this by choosing a point in the map for the tornado to move
towards');
    display('choose a number from 1 to 100 for both the x and y co-ordinates');
    xPoint = input('choose x value');
    xPoint = round(xPoint);
    while xPoint < 1 || xPoint > 100
        display('invalid choice for x, try again');
        xPoint = input('pick a value from 1 to 100');
    end
    xPoint = round(xPoint);

    yPoint = input('choose y value');

    while yPoint < 1 || yPoint > 100
        display('invalid choice for y, try again');
        yPoint = input('pick a value from 1 to 100');
    end
    yPoint = round(yPoint);

    % order the points in acending order
    if entryPointX > xPoint
        x = xPoint:entryPointX;
    else
        x = entryPointX:xPoint;
    end

    % calculate the path as a linear line
    grad = (entryPointY - yPoint)/(entryPointX - xPoint);
    c = entryPointY - grad*entryPointX;
    y = grad * x + c;
    xDir = x(end-1);
    yDir = round(y(end-1));


end



%% advanced matlab final project, setEventData2

% simon teshuva, 302207220
% last updated 17/6/2014

% this function sets the relevant data for a tsunami

function [entryPointY entryPointX strength] = setEventData2()
%% set entry points
% these are determined by calculating the direction that the tornado enters
% the city from and then finding the point which approximates this best.
display('from which direction is the tsunami coming from?');
display('1: north');
display('2: north-east');
display('3: east');
display('4: south-east');
display('5: south');
display('6: south-west');
display('7: west');
display('8: north-west');
choice = input('');
choice = round(choice);

while choice < 1 || choice > 8
    display('invalid choice, try again');
    display('from which direction is the tsunami coming from?');
    display('1: north');
    display('2: north-east');
    display('3: east');
    display('4: south-east');
    display('5: south');
    display('6: south-west');
    display('7: west');
    display('8: north-west');
    choice = input('');
    choice = round(choice);
end

%% entry point

% using the user's choice, assing the starting location for the tornado

if choice == 1
    entryPointX = 51;
    entryPointY = 1;
else if choice == 2
        entryPointX = 101;
        entryPointY = 1;
    else if choice == 3
            entryPointX = 101;
            entryPointY = 51;
        else if choice == 4
```

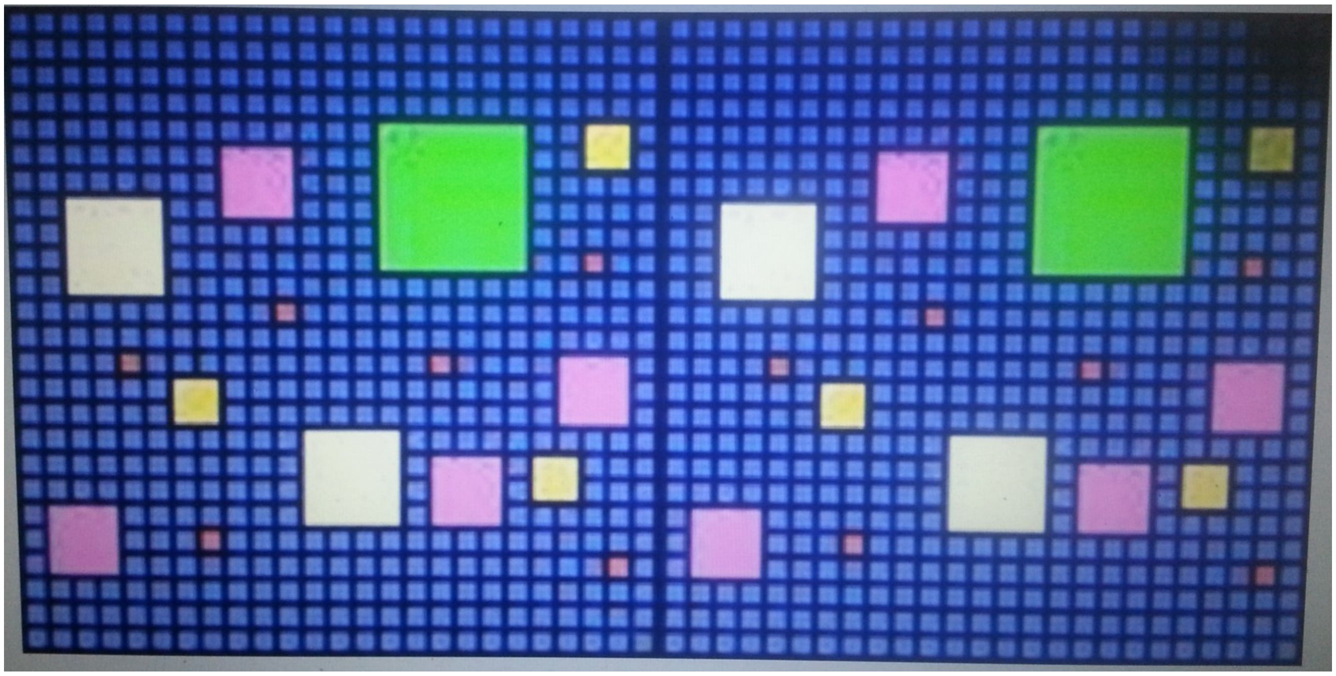
```
                entryPointX = 101;
                entryPointY = 101;
            else if choice == 5
                    entryPointX = 51;
                    entryPointY = 101;
                else if choice == 6
                        entryPointX = 1;
                        entryPointY = 101;
                    else if choice == 7
                        entryPointX = 1;
                        entryPointY = 51;
                        else if choice == 8
                                entryPointX = 1;
                                entryPointY = 1;
                            end
                        end
                    end
                end
            end
        end
    end
end

%% set strength
    display('how strong is the tsunami?');
    display('pick a value from 1 to 100');
    strength= input('');

    while strength < 0 || strength > 100
    display('ivalid choice, try again. how strong is the tsunami?');
    display('pick a value from 1 to 100');
    strength= input('');
    end



end



%% advanced matlab final project, setEventData3

% simon teshuva, 302207220
% last updated 17/6/2014

% this function sets the relevant data for a earthquake

function [entryPointY entryPointX strength] = setEventData3()
%% set origin points
% these are determined by calculating the direction that the tornado enters
% the city from and then finding the point which approximates this best
    display('where does the earthquake start from?');
    display('set this by choosing a point in the map to be the origin');
    display('choose a number from 1 to 101 for both the x and y co-ordinates');

    entryPointX = input('choose x value');
    while entryPointX < 1 || entryPointX > 101
        display('invalid choice for x, try again');
        entryPointX = input('pick a value from 1 to 101');
    end

    entryPointY = input('choose y value');
    while entryPointY < 1 || entryPointY > 101
        display('invalid choice for y, try again');
        entryPointY = input('pick a value from 1 to 101');
    end

    entryPointX = round(entryPointX);
    entryPointY = round(entryPointY);

%% set strength
    display('how strong is the earthquake?');
    display('pick a value from 1 to 100');
    strength= input('');

    while strength < 0 || strength > 100
    display('how strong is the earthquake?');
    display('pick a value from 1 to 100');
    strength= input('');
    end
end



%% advanced matlab final project, simulateEvent1

% simon teshuva, 302207220
% last updated 17/6/2014

% this function simulates the results of a tornado

function [startRGBMap finishRGBMap endDamageArray] = simulateEvent1(entryPointY,
entryPointX, strength, yDir, xDir, map)

% need to deal with roads (remove them!!)

% extraData contains health info, color code, id etc...
damageArray = zeros(101,101);


%% path

% calculate the final point for the domain. it will either be the first
% column or the last column depending on which direction the tornado is
% moving towards
xFin = (entryPointX - xDir) * strength  + entryPointX;
if xFin < 1
    xFin = 1;
else if xFin > 101
        xFin = 101;
    end
end

xVals = entryPointX:xFin;
xVals = round(xVals);

% calculate the path as a linear line based on the input
grad = (entryPointY - yDir)/(entryPointX - xDir);
```

```
if grad < 0
    grad = grad*-1;
end

c = entryPointY - grad*entryPointX;

yVals = grad*xVals + c;
yVals = round(yVals);
%% damage at each point (need to rework)

tempStrength = strength;
for i = 1:length(xVals)
    damageArray(yVals(i), xVals(i)) = tempStrength;
    tempStrength = tempStrength -1;
end

endDamageArray = damageArray;

%% call simulateEvenrt
[startRGBMap finishRGBMap] = simulateEvent(map, damageArray);
end
```

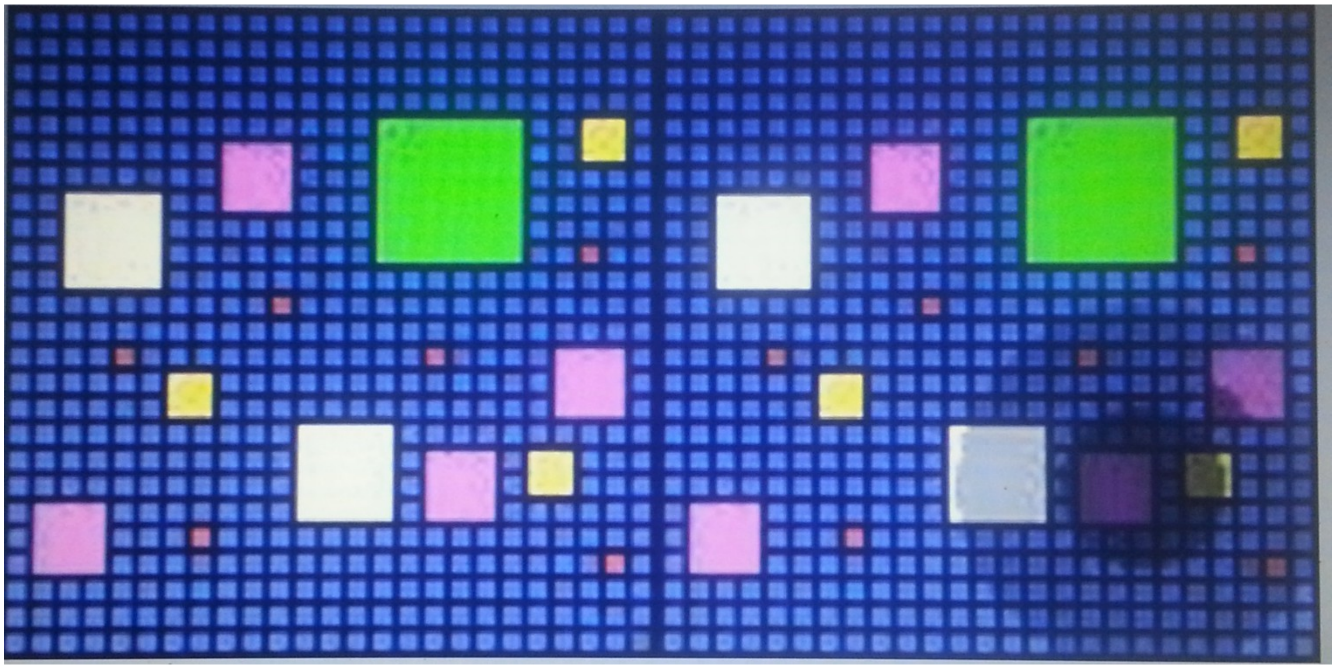


```
%% advanced matlab final project, simulateEvent2

% simon teshuva, 302207220
% last updated 17/6/2014

% this function simulates the results of a tsunami

function [startRGBMap finishRGBMap endDamageArray] = simulateEvent2(entryPointY,
entryPointX, strength, map)
% use modified earthquake
% extraData contains health info, color code, id etc...
```

```
damageArray = zeros(101,101);
strength = strength * 1.75;


%% damage done to each square;

% the damage to each square = strength - distance from start

for axisY = 1:101
    for axisX = 1:101
        xDis = (axisX - entryPointX)^2;
        yDis = (axisY - entryPointY)^2;
        dist = sqrt(xDis + yDis);
        if dist == 0
            damage = strength;
        else
            damage = strength - dist;
        end

        % if the distance is larger than the strength, no damage was done,
        % instead of letting it go into negatives set it to 0
        if damage < 0
            damage = 0;
        end

        damageArray(axisY,axisX) = damage;

    end
end

%% call simulateEvent

endDamageArray = damageArray;

[startRGBMap finishRGBMap] = simulateEvent(map, damageArray);
end
```



```matlab
%% advanced matlab final project, simulateEvent3

% simon teshuva, 302207220
% last updated 17/6/2014

% this function simulates the results of a earthquake

function [startRGBMap finishRGBMap endDamageArray] = simulateEvent3(originY,
originX, strength, map)

% extraData contains health info, color code, id etc...
damageArray = zeros(101,101);
strength = strength * 1.75;

%% damage done at each square

% the damage to each square = strength - distance from start

for axisY = 1:101
    for axisX = 1:101
        xDis = (axisX - originX)^2;
        yDis = (axisY - originY)^2;
        dist = sqrt(xDis + yDis);
        if dist == 0
            damage = strength;
        else
            damage = strength - dist;
        end

        % if the distance is larger than the strength, no damage was done,
        % instead of letting it go into negatives set it to 0

        if damage < 0
            damage = 0;
        end
        damageArray(axisY,axisX) = damage;

    end
end

%% call simulateEvent

endDamageArray = damageArray;

[startRGBMap finishRGBMap] = simulateEvent(map, damageArray);
end
```



```
%% advanced matlab final project, simulateEvent

% simon teshuva, 302207220
% last updated 17/6/2014

% this function is a generic simulation of any disaster. it is called by
% all other simulateEventX functions to do the main computations

function [finishStartRGBMap, finishFinalRGBMap] = simulateEvent(map, damageArray)

totalHealthArray = [0 45 30 45 60 30 45];

% update colors
% an array of rgb maps which can be used for the montage
% an array; 101 by 101 by 3 by length(path)
startRGBMap = map;
startRGBMap(:,:,2) = map;
startRGBMap(:,:,3) = map;




%% set initial values for rgb

% color scheme
% road = 0 0 0 - 0 0 0 - 0 0 0
% house = 100 100 255 - 75 75 190 - 40 40 100
% petrol station = 255 100 100 - 190 75 75 - 100 40 40
% shopping centre = 255 255 255 - 200 200 200 - 100 100 100
% university - 100 255 100 - 75 190 75 - 40 100 40
% park - 255 100 255 - 190 75 190 - 100 40 100
% supermarket - 255 255 100 - 190 190 75 - 100 100 40

% for each co-ordinate in the map, based off of its ID, color it in for its
% initial colors colors as seen in the table above
for xAxis = 1:length(map(:, 1))
    for yAxis = 1:length(map(1, :))
        if map(yAxis, xAxis) == 1
```

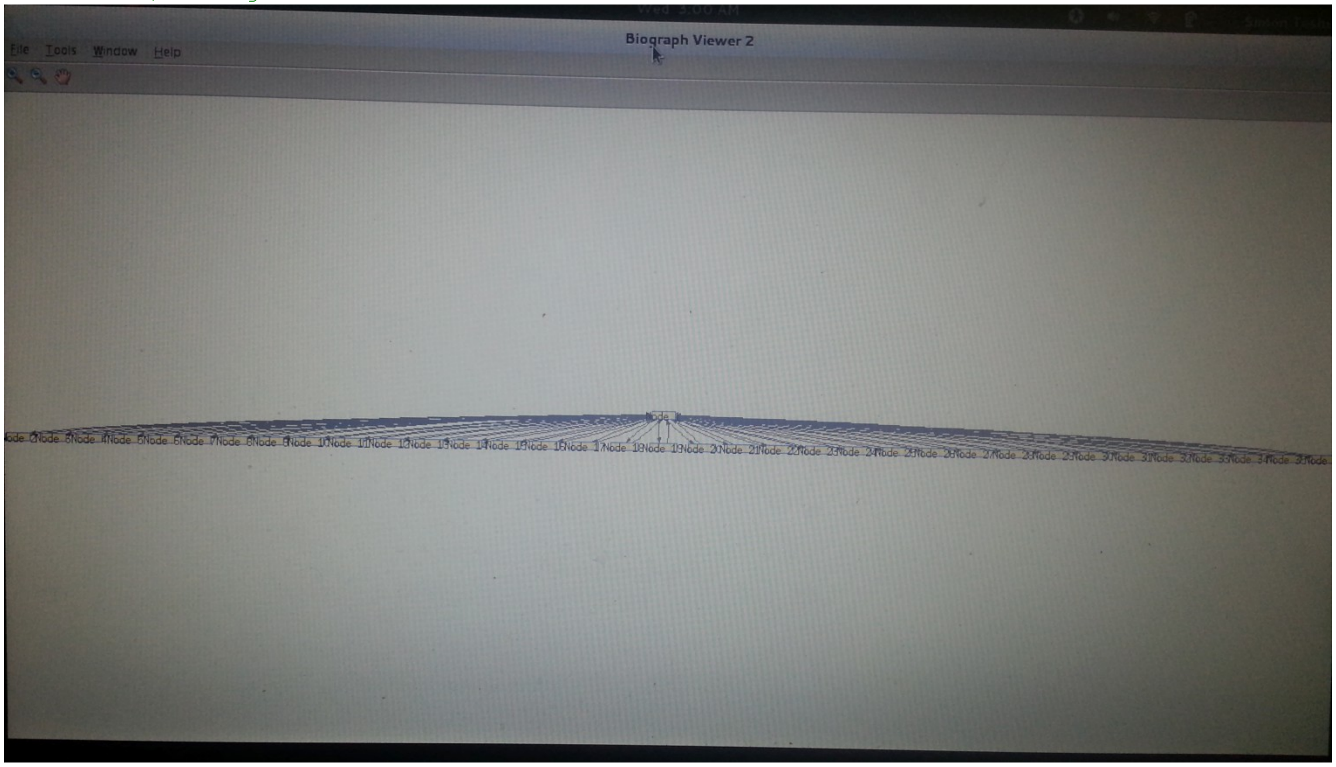
```
            startRGBMap(yAxis, xAxis, 1) = 0;
            startRGBMap(yAxis, xAxis, 2) = 0;
            startRGBMap(yAxis, xAxis, 3) = 0;
        else if map(yAxis, xAxis) == 2
                startRGBMap(yAxis, xAxis, 1) = 100;
                startRGBMap(yAxis, xAxis, 2) = 100;
                startRGBMap(yAxis, xAxis, 3) = 255;
            else if map(yAxis, xAxis) == 3
                    startRGBMap(yAxis, xAxis, 1) = 255;
                    startRGBMap(yAxis, xAxis, 2) = 100;
                    startRGBMap(yAxis, xAxis, 3) = 100;
                else if map(yAxis, xAxis) == 4
                        startRGBMap(yAxis, xAxis, 1) = 255;
                        startRGBMap(yAxis, xAxis, 2) = 255;
                        startRGBMap(yAxis, xAxis, 3) = 255;
                    else if map(yAxis, xAxis) == 5
                            startRGBMap(yAxis, xAxis, 1) = 100;
                            startRGBMap(yAxis, xAxis, 2) = 255;
                            startRGBMap(yAxis, xAxis, 3) = 100;
                        else if map(yAxis, xAxis) == 6
                                startRGBMap(yAxis, xAxis, 1) = 255;
                                startRGBMap(yAxis, xAxis, 2) = 100;
                                startRGBMap(yAxis, xAxis, 3) = 255;
                            else if map(yAxis, xAxis) == 7
                                    startRGBMap(yAxis, xAxis, 1) = 255;
                                    startRGBMap(yAxis, xAxis, 2) = 255;
                                    startRGBMap(yAxis, xAxis, 3) = 100;
                                end
                            end
                        end
                    end
                end
            end
        end
    end
end

%% updating color


% the updated map
finalRGBMap = startRGBMap;

% for each co-ordinate in the map
for xAxis = 1:length(map(:, 1))
    for yAxis = 1:length(map(1, :))

        % based of the coordinates ID, damage taken and total health per
        % square
        status = damageArray(yAxis, xAxis);
        ID = map(yAxis,xAxis);
        totalHealth = totalHealthArray(ID);

        % if less than 15% of the building has been damaged, dont update
        % anything
        if status < 0.15*totalHealth

            % else if between 15 and 50 % of the building is damaged,
            % update the colors to "damaged" status
```

```
else if status >= 0.15*totalHealth && status < 0.5*totalHealth
        if map(yAxis, xAxis) == 1
            finalRGBMap(yAxis, xAxis, 1) = 0;
            finalRGBMap(yAxis, xAxis, 2) = 0;
            finalRGBMap(yAxis, xAxis, 3) = 0;
        else if map(yAxis, xAxis) == 2
                finalRGBMap(yAxis, xAxis, 1) = 75;
                finalRGBMap(yAxis, xAxis, 2) = 75;
                finalRGBMap(yAxis, xAxis, 3) = 190;
            else if map(yAxis, xAxis) == 3
                    finalRGBMap(yAxis, xAxis, 1) = 190;
                    finalRGBMap(yAxis, xAxis, 2) = 75;
                    finalRGBMap(yAxis, xAxis, 3) = 75;
                else if map(yAxis, xAxis) == 4
                        finalRGBMap(yAxis, xAxis, 1) = 190;
                        finalRGBMap(yAxis, xAxis, 2) = 190;
                        finalRGBMap(yAxis, xAxis, 3) = 190;
                    else if map(yAxis, xAxis) == 5
                            finalRGBMap(yAxis, xAxis, 1) = 75;
                            finalRGBMap(yAxis, xAxis, 2) = 190;
                            finalRGBMap(yAxis, xAxis, 3) = 75;
                        else if map(yAxis, xAxis) == 6
                                finalRGBMap(yAxis, xAxis, 1) = 190;
                                finalRGBMap(yAxis, xAxis, 2) = 75;
                                finalRGBMap(yAxis, xAxis, 3) = 190;
                            else if map(yAxis, xAxis) == 7
                                    finalRGBMap(yAxis, xAxis, 1) = 190;
                                    finalRGBMap(yAxis, xAxis, 2) = 190;
                                    finalRGBMap(yAxis, xAxis, 3) = 75;
                                end
                            end
                        end
                    end
                end
            end
        end
    % else if over 50 % of the building is damaged,
    % update the colors to "destroyed" status
    else if status > totalHealth * 0.5
            if map(yAxis, xAxis) == 1
                finalRGBMap(yAxis, xAxis, 1) = 0;
                finalRGBMap(yAxis, xAxis, 2) = 0;
                finalRGBMap(yAxis, xAxis, 3) = 0;
            else if map(yAxis, xAxis) == 2
                    finalRGBMap(yAxis, xAxis, 1) = 40;
                    finalRGBMap(yAxis, xAxis, 2) = 40;
                    finalRGBMap(yAxis, xAxis, 3) = 100;
                else if map(yAxis, xAxis) == 3
                        finalRGBMap(yAxis, xAxis, 1) = 100;
                        finalRGBMap(yAxis, xAxis, 2) = 40;
                        finalRGBMap(yAxis, xAxis, 3) = 40;
                    else if map(yAxis, xAxis) == 4
                            finalRGBMap(yAxis, xAxis, 1) = 100;
                            finalRGBMap(yAxis, xAxis, 2) = 100;
                            finalRGBMap(yAxis, xAxis, 3) = 100;
                        else if map(yAxis, xAxis) == 5
                                finalRGBMap(yAxis, xAxis, 1) = 40;
                                finalRGBMap(yAxis, xAxis, 2) = 100;
                                finalRGBMap(yAxis, xAxis, 3) = 40;
```

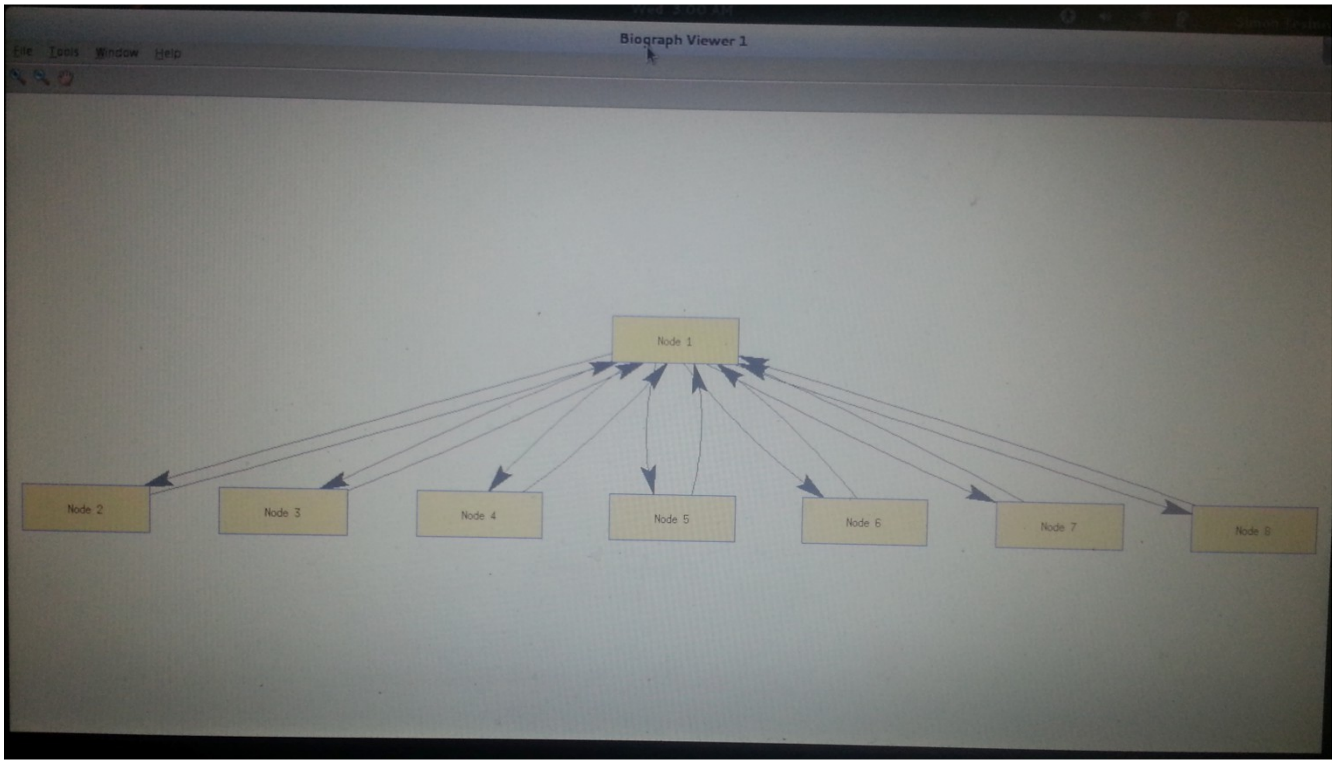
```
                                        else if map(yAxis, xAxis) == 6
                                                finalRGBMap(yAxis, xAxis, 1) = 100;
                                                finalRGBMap(yAxis, xAxis, 2) = 40;
                                                finalRGBMap(yAxis, xAxis, 3) = 100;
                                            else if map(yAxis, xAxis) == 7
                                                    finalRGBMap(yAxis, xAxis, 1) = 100;
                                                    finalRGBMap(yAxis, xAxis, 2) = 100;
                                                    finalRGBMap(yAxis, xAxis, 3) = 40;
                                                end
                                            end
                                        end
                                    end
                                end
                            end
                        end
                    end
                end
            end
        end
    end
end
%% normalise values

% normalise the values of the start and finish map to be in the range from
% 0 to 1
finishStartRGBMap = startRGBMap/255;
finishFinalRGBMap = finalRGBMap/255;

end


%% advanced matlab final project, createGraph

% simon teshuva, 302207220
% last updated 17/6/2014

% this is a function that creates a graph based on where the
% disaster damages

function [damagedGraph damagedVector destroyedGraph destroyedVector]=
createGraph(map, damageArray, buildingIDs, xStart, yStart)
%% intiialise values
totalHealthArray = [0 45 30 45 60 30 45];
dimensionArray = [0 3 3 15 23 11 7];

numberOfBuildings = 505;
damagePerBuilding = zeros(1, numberOfBuildings);
buildingTypeArray = zeros(1, numberOfBuildings);

startBuilding = buildingIDs(yStart, xStart);

% for each point in the map (but effectivley for each one of the 505
% buildings
for yAxis = 1:length(map(:,1))
    for xAxis = 1:length(map(1,:))
        % the building's type is determined from the value of 'map' at that
        % poit
        ID = buildingIDs(yAxis, xAxis);
        type = map(yAxis, xAxis);
        buildingTypeArray(ID) = type;
    end
end

%% calculate damage per bilding

% using the damageArray and buildingIDs, for each point, find which
% building we are looking at, how much damage has been done at that point
% and update the ammount of damage dealt to that building overall
for xAxis = 1:length(map(1,:))
    for yAxis = 1:length(map(:, 1))
        type = map(yAxis, xAxis);
        ID = buildingIDs(yAxis, xAxis);
        damage = damageArray(yAxis, xAxis);
        if type ~= 1
            damagePerBuilding(ID) = damagePerBuilding(ID) + damage;
        end
    end
end



%% add in nodes to graph

% two graphs will be created, one for buildings significantly damaged and
% one for those destroyed
buildingsInDamagedGraph = zeros(1, numberOfBuildings);
buildingsInDestroyedGraph = zeros(1, numberOfBuildings);

% for each building
for i = 1:numberOfBuildings
    type = buildingTypeArray(i);
    health = totalHealthArray(type)*dimensionArray(type)^2;
    DPB = damagePerBuilding(i);
    % if more than 15% of the building has been damaged it goes in to the
    % list of damaged buildings
    if DPB > health * 0.15
        buildingsInDamagedGraph(i) = 1;
        % if more than 50% of the building has been damaged it also goes in
        % to the list of destroyed buildings
        if DPB > health * 0.5
            buildingsInDestroyedGraph(i) = 1;
        end
    end
end

% creating vectors of all buildings that have been damaged or destroyed
% first need to know how large the vectors will be,
% trace trace through the arrays buildingsInDestroyedGraph and
% buildingsInDestroyedGraph and keep a counter of the number of 1s
numberDamaged = 0;
numberDestroyed = 0;
for i = 1:numberOfBuildings
    if buildingsInDestroyedGraph(i) == 1
        numberDamaged = numberDamaged + 1;
    end

    if buildingsInDestroyedGraph(i) == 1
        numberDestroyed = numberDestroyed + 1;
    end
```

```
end

% create the two vectors based on the counters calculated above
damagedVector = zeros(1, numberDamaged);
destroyedVector = zeros(1, numberDestroyed);
damagedCounter = 1;
destroyedCounter = 1;

% for each building, check if it has been damaged or destroyed, and if it
% has add it in to the appropriate vector. then update the counter for that
% vector, which indicates where to place the next piece of data
for i = 1:numberOfBuildings
    if buildingsInDamagedGraph(i) == 1 && startBuilding ~= i
        damagedVector(damagedCounter) = i;
        damagedCounter = damagedCounter + 1;
    end

    if buildingsInDestroyedGraph(i) == 1 && startBuilding ~= i
        destroyedVector(destroyedCounter) = i;
        destroyedCounter = destroyedCounter + 1;
    end
end

% the graphs will be represented as arrays of size
% #buildings damaged x #buildings damaged
% #buildings destroyed x #buildings destroyed
damaged = length(damagedVector);
destroyed = length(destroyedVector);

myDamagedGraph = zeros(damaged , damaged);
myDestroyedGraph = zeros(destroyed, destroyed);

% for each building other the first

% create an undirected
% edge joining the first building to the one being looked at

% do this for both damaged and destroyed buildings
for i = 2:damaged
    myDamagedGraph(1, i) = 1;
    myDamagedGraph(i, 1) = 1;
end

for i = 2:destroyed
    myDestroyedGraph(1, i) = 1;
    myDestroyedGraph(i, 1) = 1;
end

%% return graph

damagedGraph = myDamagedGraph;
destroyedGraph = myDestroyedGraph;
end
```

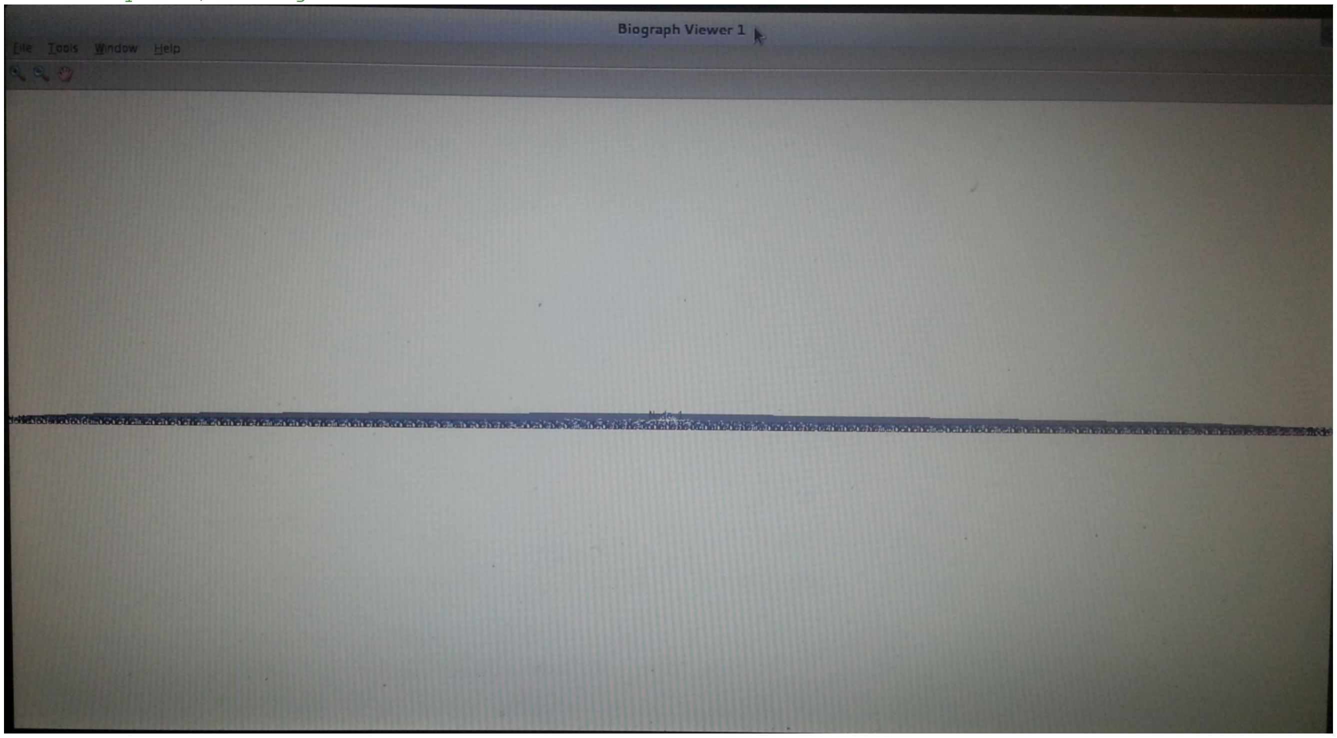
```
% tsunami, damaged
```



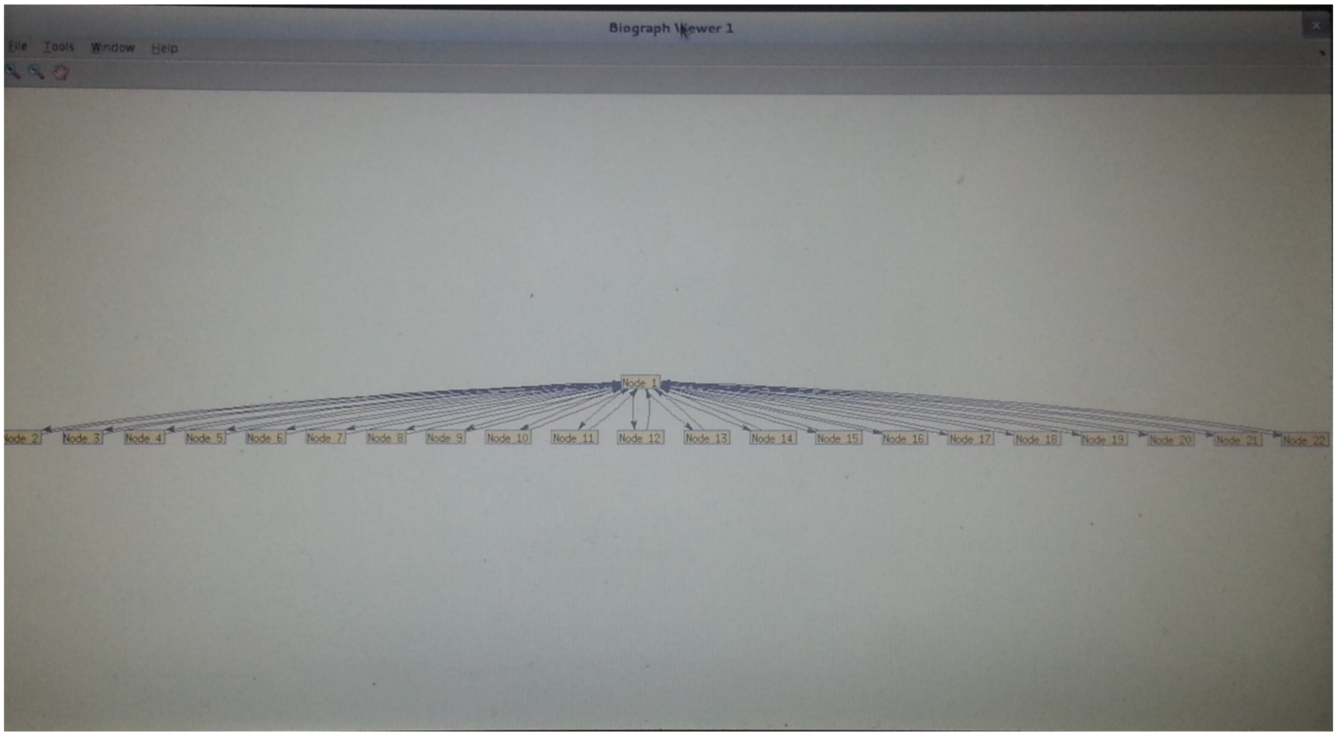
```
% tsunami, destroyed
```



```
% tsunami, destroyed
% earthquake, damaged
```



```
% earthquake, destroyed
```



```
%% advanced matlab final project, graphs

% simon teshuva, 302207220
% last updated 17/6/2014

% this is a function that takes as input an array of buildings and creates
% and returns a graph and list of names using simbiology.

function [outputGraph outputNames] = graphs(damagedGraph, damagedVector,
destroyedGraph, destroyedVector)


display('which graph would you like to see?');
display('1: buildingds that were damaged significantly');
display('2: buildingds that were completly destroyed');
display('0: none');
choice = input('');
choice = round(choice);

while choice <0 || choice > 2
    display('invalid choice, try again');
    display('which graph would you like to see?');
    display('1: buildingds that were damaged significantly');
    display('2: buildingds that were completly destroyed');
    display('0: none');
    choice = input('');
    choice = round(choice);
end



if choice == 1
    finalGraph = biograph(damagedGraph);
    view(finalGraph);
    outputNames = damagedVector;
```

```matlab
else if choice == 2
        finalGraph = biograph(destroyedGraph);
        view(finalGraph);
        outputNames = destroyedVector;
    end
end



outputGraph = finalGraph;
end



%% advanced matlab final project, calculateDamages

% simon teshuva, 302207220
% last updated 17/6/2014

% this is a function that calculates the total damage done by a disaster

function totalCost = calculateDamages(map, buildingIDs, damageArray)
%% intialise variables
totalHealthArray = [0 45 30 45 60 30 45];
totalCostArray = [0 300000 400000 25000000 300000000 750000 2000000];
dimensionArray = [0 3 3 15 23 11 7];

numberOfBuildings = 505;

%% iterativly calculate cost of damage to each building add sum it

tempCost = 0;
% for each building
for counter = 1:numberOfBuildings
    % re-initialise variables
    thisCost = 0;
    found = false;
    y = 1;
    x = 1;

    % for find top - left most point of the building
    for counterY = 1:length(buildingIDs(1,:))
        for counterX = 1:length(buildingIDs(:,1))
            if buildingIDs(counterY, counterX) == counter && found == false
                found = true;
                x = counterX;
                y = counterY;
            end
        end
    end

    % find data about the building
    type = map(y,x);
    size = dimensionArray(type);
    buildingCost = totalCostArray(type);

    % the total damgage a building can take
    buildingHealth = size*size*totalHealthArray(type);
    % calculate the damade done to that building
    thisRealDamage = calculateCost(type, y, x, damageArray);


    % if less than 15% of the building has been damaged (mostly fine)
    if thisRealDamage < 0.15*buildingHealth
        thisCost = thisRealDamage * buildingCost * 0.33;
        % if between 15 and 50 % of th ebuilding has been damaged (damged)
    else if thisRealDamage >= 0.15*buildingHealth && thisRealDamage <
0.5*buildingHealth
            thisCost = thisRealDamage * buildingCost * 0.66;
            % if more than 50% of the building has been damaged (destroyed)
        else if thisRealDamage >= 0.5*buildingHealth
                thisCost = thisRealDamage * buildingCost;
            end
        end
    end

    % bring the cost in to sensible regions
    thisCost = thisCost / 10;

    % if the total cost of the
    % damage done to a building is greater than the value of
    % the building, make it equal to the actual cost of the building
    if thisCost > buildingCost
        thisCost = buildingCost;
    end

    % update the total cost of the damage
    tempCost = tempCost + thisCost;

end

% return the total cost
totalCost = tempCost;
end



%% advanced matlab final project, createGraph

% simon teshuva, 302207220
% last updated 17/6/2014

% this is a function that calculates the total damage done by a disaster to
% one particular building

function thisRealDamage = calculateCost(type, y, x, damageArray)
%% initialise variables

% the dimensions for each building type
dimensionArray = [0 3 3 15 23 11 7];
% the ammound of damage each square of a given building can take
totalHealthArray = [0 45 30 45 60 30 45];
% the layout for the above two vectors is
% array = [road house petrolStation shoppingCenter university park
% supermarket]

size = dimensionArray(type);
```

```matlab
healthPerSquare = totalHealthArray(type);

%% calculate damage to the building

% the building extends from the top left most corner an equal ammount in
% either direction, the ammount is based off of the building type

totalDamage = 0;
% for each point in the building keep a running sum of the damage to that
% building
for xAxis = 0:(size-1)
    for yAxis = 0:(size-1)
            tempDamage = damageArray(y + yAxis, x + xAxis);
            if tempDamage > healthPerSquare
                tempDamage = healthPerSquare;
            end
            totalDamage = totalDamage + tempDamage;
    end
end

%% bound it by max building health
% if the total damage done to the building is greater than the total damage
% that it can sustain, set total damage done to total max damage possible
buildingHealth = size*size*healthPerSquare;
if totalDamage > buildingHealth
    thisRealDamage = buildingHealth;
else
    thisRealDamage = totalDamage;
end

end



%% advanced matlab final project, findNearestDamagedBuilding

% simon teshuva, 302207220
% last updated 17/6/2014

% this is a function that takes one damaged building and finds the nearest
% damaged building

function nearest = findNearestDamagedBuilding(building, map, damageArray,
buildingIDs, damageType)
%% initialise variables

totalHealthArray = [0 45 30 45 60 30 45];
dimensionArray = [0 3 3 15 23 11 7];

counterX = 1;
counterY = 1;
found = false;
yPoint = 1;
xPoint = 1;

%% find our building and relevant data about it

% trace trough the map until the top - left most point of our building has
% been found. save its co-ordinates
```

```
while counterX < length(buildingIDs(1,:)) && found == false
    while counterY < length(buildingIDs(:,1)) && found == false
        if buildingIDs(counterY,counterX) == building
            found = true;
            yPoint = counterY;
            xPoint = counterX;
        end
        counterX = counterX+1;
        counterY = counterY+1;
    end
end

% find the middle of the building

buildingType = map(yPoint, xPoint);

% based on the building's type, toAdd = (building size + 1)/2
toAdd = 0;
if buildingType == 2
    toAdd = 1;
else if buildingType == 3
        toAdd = 1;
    else if buildingType == 4
            toAdd = 7;
        else if buildingType == 5
                toAdd = 11;
            else if buildingType == 6
                    toAdd = 5;
                else if buildingType == 7
                        toAdd = 3;
                    end
                end
            end
        end
    end
end

% by adding toAdd to the xPoint and yPoint we find the middle of the
% building
xPoint = xPoint + toAdd;
yPoint = yPoint + toAdd;


%% find nearest building and return
dist = 10000000;
ID = 10000;

% for each point in the map
for counterX = 1:length(buildingIDs(1,:))
    for counterY = 1:length(buildingIDs(:,1))
        % find relevant data about that point
        currentDamage = damageArray(counterY, counterX);
        currentType = map(counterY, counterX);

        % calculate the distance from our initial building and find the ID
        % of the building we are currently looking at
        currentDist = sqrt((counterX - xPoint)^2 + (counterY - yPoint)^2);
        currentID = buildingIDs(counterY, counterX);
        % if the building being looked at is not a road, and has been
        % damaged, and is not our initial building, examine it further
        if currentType ~= 1 && currentDamage > 0 && currentID ~= building
            % if it is closer to our building than the previous closest
            % building
            if currentDist < dist
                % if we are looking for damaged or destroyed buildings and
                % if more than 15% of the building has been damaged
                if damageType == 1 && currentDamage > 0.15 *
totalHealthArray(currentType)
                % store the distance from this building and the building's
                % ID
                dist = currentDist;
                ID = currentID;
                % else if we are only looking for buildings that have beeen
                % destroyed, and more than 50% of the building has been
                % damaged
                else if damageType == 2 && currentDamage > 0.5 *
totalHealthArray(currentType)
                % store the distance from this building the the building's
                % ID
                dist = currentDist;
                ID = currentID;
                    end
                end

            end
        end
    end
end

% return the closest building's ID
nearest = ID;

end



%% advanced matlab final project, shownImage

% simon teshuva, 302207220
% last updated 17/6/2014

% creates a map of the city to show to the user. it is called by the main
% file at the very beginning and is in the form of a function in order to
% save space

function shownImage = createStartImage(map)

% create an array of size 3x101x101 for the rgb componenets of the map
myMap = map;
myMap(:,:,2) = map;
myMap(:,:,3) = map;



%% set initial values for rgb

% color scheme
```

```matlab
% road = 0 0 0 - 0 0 0 - 0 0 0
% house = 100 100 255 - 75 75 190 - 40 40 100
% petrol station = 255 100 100 - 190 75 75 - 100 40 40
% shopping centre = 255 255 255 - 200 200 200 - 100 100 100
% university - 100 255 100 - 75 190 75 - 40 100 40
% park - 255 100 255 - 190 75 190 - 100 40 100
% supermarket - 255 255 100 - 190 190 75 - 100 100 40

% for each point in the map, based on the building type, color that point
% based on the color scheme above
for xAxis = 1:length(map(:, 1))
    for yAxis = 1:length(map(1, :))
        if map(yAxis, xAxis) == 1
            myMap(yAxis, xAxis, 1) = 0;
            myMap(yAxis, xAxis, 2) = 0;
            myMap(yAxis, xAxis, 3) = 0;
        else if map(yAxis, xAxis) == 2
                myMap(yAxis, xAxis, 1) = 100;
                myMap(yAxis, xAxis, 2) = 100;
                myMap(yAxis, xAxis, 3) = 255;
            else if map(yAxis, xAxis) == 3
                    myMap(yAxis, xAxis, 1) = 255;
                    myMap(yAxis, xAxis, 2) = 100;
                    myMap(yAxis, xAxis, 3) = 100;
                else if map(yAxis, xAxis) == 4
                        myMap(yAxis, xAxis, 1) = 255;
                        myMap(yAxis, xAxis, 2) = 255;
                        myMap(yAxis, xAxis, 3) = 255;
                    else if map(yAxis, xAxis) == 5
                            myMap(yAxis, xAxis, 1) = 100;
                            myMap(yAxis, xAxis, 2) = 255;
                            myMap(yAxis, xAxis, 3) = 100;
                        else if map(yAxis, xAxis) == 6
                                myMap(yAxis, xAxis, 1) = 255;
                                myMap(yAxis, xAxis, 2) = 100;
                                myMap(yAxis, xAxis, 3) = 255;
                            else if map(yAxis, xAxis) == 7
                                    myMap(yAxis, xAxis, 1) = 255;
                                    myMap(yAxis, xAxis, 2) = 255;
                                    myMap(yAxis, xAxis, 3) = 100;
                                end
                            end
                        end
                    end
                end
            end
        end
    end
end
% normalise and returnt the map
shownImage = myMap/255;
end
```