Project 2

|  |  |
| --- | --- |
| Due Date: | 7/5/2021 (We do not have class on 7/5) |
| Assignment Type: | Individual (no external help) |
| Assignment Title: | Project 2– JavaScript and Canvas |
| Style: | Submit all before due date to GL |
| External Sources: | You are allowed notes, books, and searches |
| Description: | This project uses JavaScript and Canvas to make fractals. |
| Points | 80 |
| Starting Files | One for part 2 on Blackboard. |
| Turn in | Turn in all files zipped on Blackboard. |

# Part 1. FRACTAL IMAGES (40 pts)

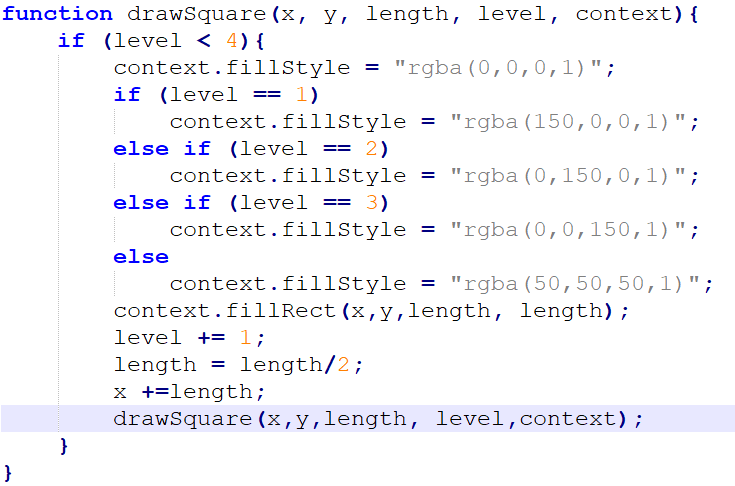
# Overview

Make sure you understand how to draw on the canvas using a variety of tools before tackling this one. In this assignment you will implement a recursive algorithm to create a fractal image on the canvas. The most points go to those with the best implementations.

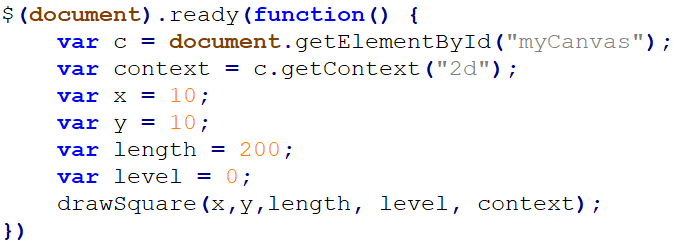
# Step 1 – Introduction (with introductory code)

For your part1, please name your files proj2\_1.html and proj2\_1.js.

Here is a starting point for a sample recursive run.

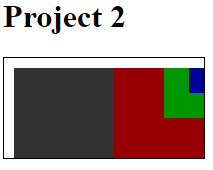


This would be the call to drawSquare:



Additionally, the $(document).ready is from the jQuery library. Here is the sample html for that.

When you implement this, you will get something like this:



Now you are going to modify this code to create a fractal.

# Section 2 – Algorithm for Fractal

Here is the algorithm for a function that will draw branches on a tree. The function is recursive and will not run more than maxLevels times. All it needs to know are the following items of information:

• x – the x coordinate of the line starting point

• y – the y-coordinate of the line starting point

• angle – the angle of the line (in degrees)

• len – the length of the line

• level – the current level of recursion

Given this information, a recursive function (which we shall call drawBranch()) might look like this: Algorithm for drawBranch(int x, int y, in tang, int len, int level)

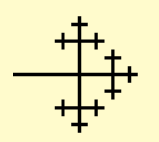
1. if the level < maxLevels then
   1. convert the angle from degrees to radians by dividing ang by 180 and multiplying by PI
   2. calculate the newx coordinate of the endpoint of the line (multiply len by the cosine of the angle and add the result to x)
   3. calculate the newy coordinate of the endpoint of the line (multiply len by the sine of the angle and add the result to y)
   4. draw the line from (x,y) to (newx, newy)
   5. Make three recursive to drawBranch calls substituting newx for x, newy for y, with len half as short and 1 added to the level but change the angle each time
      1. The original angle
      2. angle + 90
      3. angle - 90

In this project, your job is to create the code for the drawBranch() function. Test it by drawing one branch of the tree, setting maxLevels to 4. If that works you can add more branches. Set your canvas dimensions to width=”500” height=”500” and begin with the angle set to 0 and the starting positions for x and y as

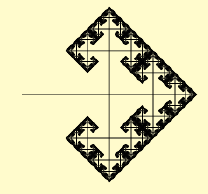
x = canvas.width / 2;

y = canvas.height/2;

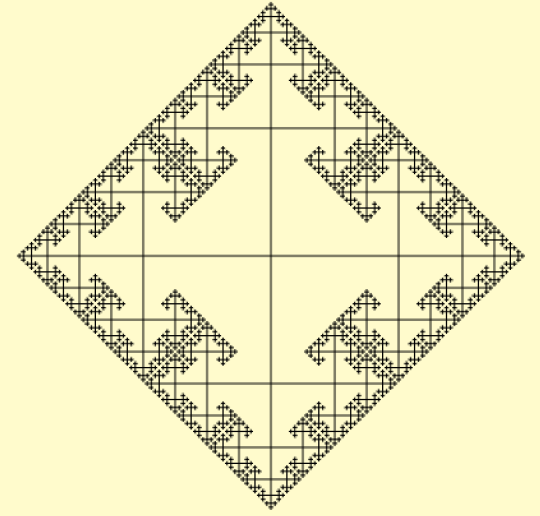
With one branch, it should look like this:



With eleven levels of recursion, it should look like this:



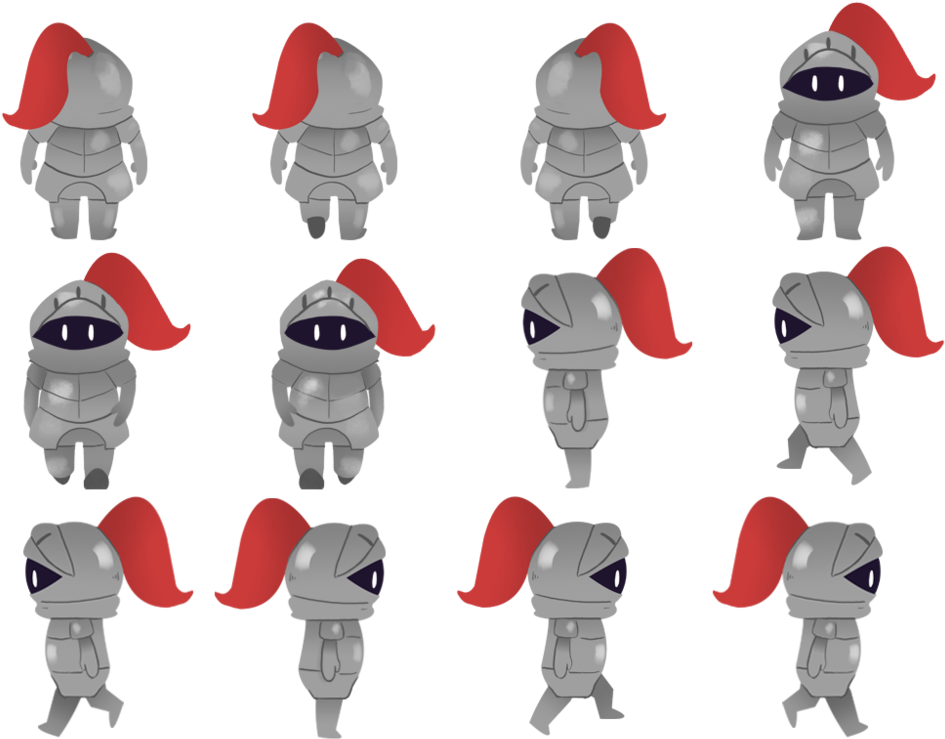
With four branches and 7 levels - The example below is four branches, drawn with canvas.lineWidth set to its default and 7 levels of recursion. There are four calls to drawBranch() from within the (document).ready function, each with a different angle (0, 90, -90, 180).



Try a variety of setups!

Part 2. Animations (40 pts)

Based on what we did in lab 7, you are going to be animating a knight to move around an HTML5 canvas. You can download the full sprite from Blackboard (do not copy and paste this one!).



If you are not pressing a button, the knight should not move and should just be facing you.

If you press the right arrow key or “d” then it should move to the right.

If you press the left arrow key or “a” then it should move to the left.

If you press the up arrow key or “w” then it should move up.

If you press the down arrow key or “s” then it should move down.

If it gets to the edge of the canvas, it should keep walking but not move.

When you have this working, save it as proj2\_2.html and proj2\_2.js.