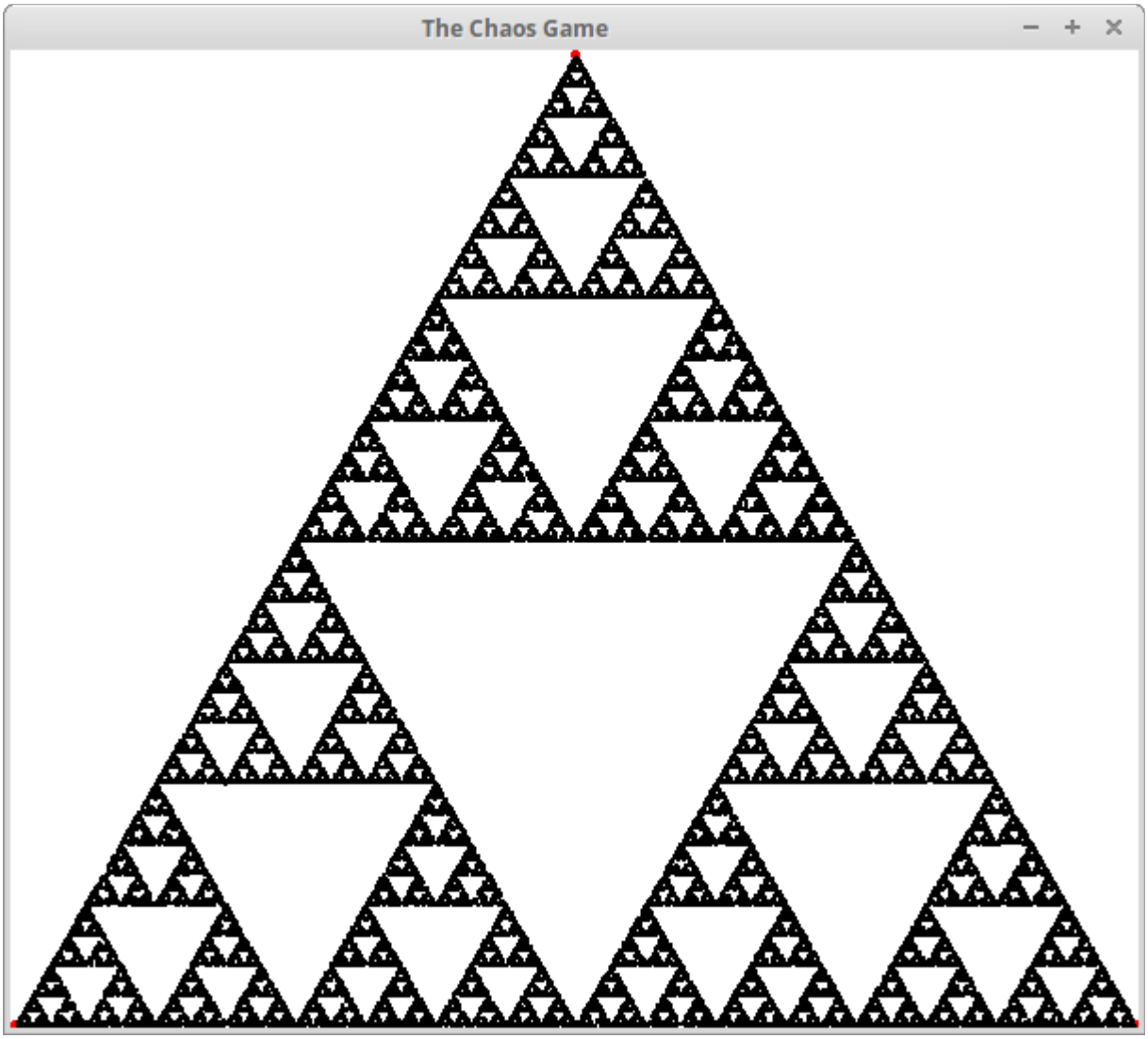
Program: The Chaos Game

Your task in this programming assignment is to implement the chaos game as introduced in a previous lesson. Specifically, you must implement the variation that results in the Sierpinski triangle. Refer to the previous lesson on Chaos if necessary.

Use your previous programming assignment to help you graphically plot points. Plot at least 50,000 points on a canvas that is 600x520 so that the fractal is uniform and clear. Color the vertices in one color (e.g., red) and the generated midpoints in another color (e.g., black). Make the vertices slightly larger than the other points. Your output should look somewhat like the following:



To plot the vertices, we simply need to determine their appropriate *x*- and *y*-coordinates on the canvas. The top vertex, for example, is located at the center horizontally. Note that (0,0) is at the top-left of the canvas. A first try may be to plot a point at (300, 0). The point may not be visible, however, because a *y*-coordinate of 0 may be slightly too high for a point to be visible on the canvas. Therefore, you may

need to experiment a bit to determine the minimum and maximum values for plotting points on the canvas. For example, try plotting a point at (0,0). If the point is not visible, try (1,1), and so on. This will help determine the minimum *x*- and *y*-coordinates. In fact, you may want to store these values as constants (e.g., `MINX` and `MINY`). The same can be done to determine the maximum *x*- and *y*-coordinates. Storing these values as constants could allow the window to be programmatically resized without having to change plotted vertices or points on the canvas! Of course, this only works if the constants are used in the calculations for all point coordinates. In fact, I suggest the following:

- Set a constant `WIDTH` to the width of the window (i.e., 600);
- Set a constant `HEIGHT` to the height of the window (i.e., 520);
- Set a constant `MIN_X` to the minimum *x*-coordinate value for a visible point (you may need to experiment with this value);
- Set a constant `MAX_X` to the maximum *x*-coordinate value for a visible point (you may need to experiment with this value);
- Set a constant `MIN_Y` to the minimum *y*-coordinate value for a visible point (you may need to experiment with this value);
- Set a constant `MAX_Y` to the maximum *y*-coordinate value for a visible point (you may need to experiment with this value);
- To make things easier, you can also set a constant that represents the center of the canvas horizontally: `MID_X` = (`MIN_X` + `MAX_X`) / 2; and
- Also set a constant that represents the center of the canvas vertically: `MID_Y` = (`MIN_Y` + `MAX_Y`) / 2.

You will need to re-implement the 2D point class. Recall the requirements:

- A 2D point is made up of an *x*-component and a *y*-component. Each component is a floating point value;
- A constructor should initialize a point either with specified values for the *x*- and *y*-components or the point (0.0,0.0) as default;
- Instance variables should be appropriately named (i.e., beginning with underscores);
- Accessors and mutators should provide read access of and write access to the instance variables;
- A function named `dist` should take **two points as input** and calculate and **return the floating point distance** between the two points;
- A function named `midpt` should take **two points as input** and calculate and **return the midpoint** of the two points; and
- A *magic* function should provide a string representation of a point in the format `(x,y)`.

You will also need to re-implement the coordinate system class. **This time, however, let's refer to it as the chaos game class.** Recall the general requirements of the coordinate system class:

- It must properly inherit from Tkinter's canvas class and fill the entire window (except for the title bar);
- Plotted points should be individual instances of your 2D point class; and
- Generally, plotted points should have a radius of 0 (i.e., a point is made up of a single pixel).

To implement the chaos game, the chaos game class has the following **additional** requirements:

- Vertices should be plotted in a color of your choice and have a radius greater than 0 (i.e., be larger than the generated midpoints that make up the fractal);

- Midpoints making up the fractal should be formed from the last plotted midpoint and a randomly chosen vertex (i.e., through the `midpt` function of your 2D point class);
- You are free to plot the first midpoint using any two vertices;
- Midpoints making up the fractal should be plotted in a color that is different than the vertices;
- You are free to set the canvas background to a color of your choice (however, ensure that the vertices and fractal are clearly visible); and
- You are free to define other functions in the chaos game class as appropriate.

To help clarify, here are some specifics and/or constraints:

(1) Make sure that your 2D point class addresses all of the requirements listed above (note that a properly implemented 2D point class in the last programming assignment ensures this);
(2) Make sure that your chaos game class addresses all of the requirements listed above;
(3) Set the window title to "The Chaos Game";
(4) Set the number of point to plots to 50,000 by default, and pass this to an appropriate function in the chaos game class from the main part of the program (through an object reference);
(5) You must include a meaningful header, use good coding style, use meaningful variable names, and comment your source code where appropriate;
(6) Your output should look somewhat like the sample run shown above; and
(7) You must submit your source code as a single .py file.