In this project, you will create a self-inverse fractal using an iterated function system (IFS).

1 Background: Circle Inversion

Circle inversion fractals are based on the geometric operation of inversion of a point with respect to a circle, shown schematically in Fig. 1.

![Figure 1: Schematic illustration of inversion of points with respect to a circle. Points $p$ and $p'$ map to each other when inverted with respect to the circle centered at $c$.](image)

1.1 Geometric construction

To geometrically find the inverse of a point with respect to a circle centered at a point $c$, one performs one of the following constructions, depending on whether the point is inside or outside the given circle:

- If the point to invert $p$ is outside the circle:
  1. Construct the line segment from $c$ to $p$.
  2. Construct the two tangent lines from $p$ to the circle, designating these points $t_1$ and $t_2$.
  3. Construct the line segment from $t_1$ to $t_2$.
  4. The intersection of the line from $c$ to $p$ with the line from $t_1$ to $t_2$ (the midpoint of the latter) is $p'$, the inverse of point $p$ with respect to the circle.

- If the point to invert $p'$ is inside the circle:
1. Extend a line from the $c$ through and beyond $p'$.
2. Construct a perpendicular line through $p'$ until it intersects the circle at points $t_1$ and $t_2$.
3. Construct lines tangent to the circle at points $t_1$ and $t_2$.
4. The intersection of these tangent lines is $p$, the inverse of point $p'$ with respect to the circle.

1.2 Properties of inversion with respect to a circle

Inversion with respect to a circle has several interesting properties:

1. The product of the distance from $p$ to $c$ with the distance from $p'$ to $c$ is the square of the circle radius: $cp \times cp' = r^2$.
2. Points on the inversion circle map to themselves under inversion. (from property 1.)
3. Points infinitely far from the circle map to the center of the inversion circle after inversion. (from property 1.)
4. Circles invert to circles.
5. A circle that is orthogonal to the inversion circle is its own inverse (it maps to itself under inversion).
6. Lines invert to circles, where the resulting circle passes through the center of the inversion circle. A line can be thought of as a circle of infinite radius, so this is a special case of circles inverting to circles.
7. Inversion with respect to a line (a circle of infinite radius) is equivalent to a mirror operation with respect to that line.

1.3 Numerical calculation of the inverse of a point

To calculate the inverse of a point numerically, one can make use of property 1 above. Specifically, the inverse $p'$ of the point $p$ falls on the vector from $c$ to $p$, and the above formula can be used to calculate the distance $cp'$ of the inverse point from the circle: $cp' = r^2/cp$.

1.4 Inversion Fractals

When a shape is unchanged after inversion with respect to a circle, it is said to be self-inverse with respect to that circle. It is also said to possess inversion symmetry with respect to that circle. Given a set of inversion circles, such as the blue circles and vertical line shown in Fig. 2, it is interesting to ask about the simplest shape that possesses inversion symmetry with respect to all of the given circles simultaneously. Provided that the inversion circles are chosen carefully, the answer is a fractal, such as that shown in gray in Fig. 2.
1.4.1 Constructing Inversion Fractals using Iterated Function Systems

Given a set of inversion circles, it is straightforward to construct the inversion fractal that is self-inverse with respect to each of the circles. The following describes an algorithm for constructing an approximate rendering of the fractal:

1. Designate the set of inversion circles as \( C_i \), where \( i \) is an index that ranges from 1 to \( n \), where \( n \) is the number of inversion circles.

2. Choose an initial point \( p \) that is known to be on the fractal. For the fractal shown in Fig. 2, for example, one could choose \( p = (0, \phi) \).

3. For some large number of iterations, repeat the following steps:

   4. Mark the location of \( p \) on the plane.

   5. Randomly choose one of the inversion circles \( C_i \) and invert the point \( p \) with respect to that circle. Designate the new point as \( p' \), replacing the previous definition of \( p \).

The resulting sequence of points will fill in the detail of the inversion fractal. To mark the points so that they have some non-zero area when we follow step 4 in the algorithm, we use an image and mark the nearest pixel as black every time we find a new point location. (The image is stored as a two-dimensional array.)

2 Problem Statement

For this project, you will create a clear, well-commented Python program to generate an inversion fractal like that shown in Fig. 2. A complete submission will include both the code as well as the image produced by the code, as described below.

1. Using nonlinear root-finding, solve for the positions and radii of the inversion circles shown in Fig. 2, using the following facts:

   (a) The upper circle passes through the point \((0, \phi)\), where \( \phi \equiv \frac{\sqrt{5} + 1}{2} \) is the golden ratio.

   (b) The lower circle passes through the point \((0, -\phi)\).

   (c) The radius of the upper circle is twice that of the lower circle.

   (d) When any circle crosses another circle or the vertical line in the diagram, they are orthogonal to each other, i.e., they cross each other at right angles.

   (e) Whenever non-orthogonal circles or the vertical line touch, they are tangent to each other.

   (f) The three largest circles are internally tangent to the outer dotted circle, which encompasses the entire fractal and is centered at \((0, 0)\) with a radius of \( \phi \).
Store the resulting solution in a $6 \times 3$ array of values, which each row corresponding to one of the circles, and the first, second, and third columns corresponding to $cx$, $cy$, and $r$ for the circles.

2. Create a function, called $\text{inversion}(c, p)$, which will take an inversion circle $c$ and a point $p$ as inputs and return the inverted point $p_{\text{prime}}$. Utilize property 1 above, as described in Sec. 1.3. Represent the circle as a one-dimensional array containing the $x$- and $y$-coordinates of the inversion circle and its radius, i.e., $c = \text{array}(\[cx, cy, r\])$. Represent the input point as a one-dimensional array of the point’s $x$- and $y$-coordinates, i.e., $p = \text{array}(\[px, py\])$. Represent the output from the function similarly. As a test of your function, note that the inversion of the point $(5, 6)$ with respect to a circle centered at the point $(3, 4)$ with radius 5 is $(9.25, 10.25)$.

3. Create a two-dimensional array $\text{im}$ with size $1000 \times 1000$, initially filled with ones. This array will represent a grayscale image of the fractal, with the value of the pixel indicating its brightness on a scale of 0 to 1.

4. As shown schematically in Fig. 3, define a linear mapping $\text{rowcol}(p)$ from $(x, y)$ coordinates to [row, col] indices. Define the function so that it takes as input a point represented as a one-dimensional array using the same convention as your inversion function, and returns a (row, col) tuple. Take note of the fact that when indexing a two-dimensional array, we have $\text{im}[\text{row}, \text{col}]$, so that the horizontal location appears second, and that the convention is for row zero to appear at the top of the array, with increasing row number moving downward while increasing $y$ moves upward.

5. Implement the algorithm described above in Sec.1.4.1, using the 6 circles whose centers and radii you solved for in step 1. Consider the vertical line to be a 7th special circle, but note that it is inappropriate to store this as a circle of infinite radius, since the inversion formula would break down then. Rather, for inversion about the given vertical line, simply negate the $x$-coordinate of the point. At each iteration, use $\text{scipy.random.random_integers}$ or $\text{pylab.randint}$ to choose which of the 7 transformations to perform next. After each transformation, determine the coordinates of the nearest pixel using your $\text{rowcol}$ function, and set the corresponding element of the $\text{im}$ array to 0 to mark it as black. For your final submission, use a large number of iterations to produce a high-quality image, but using a much smaller number may be recommended while debugging your code.

6. Use $\text{scipy.misc.imsave}$ to save the $\text{im}$ array to a file called ‘$\text{inversion} \_\text{fractal.png}$’. Be sure to submit this image along with your code in your final submission. Note that while you are debugging, you can also use $\text{pylab.imshow(}\text{im}\text{)}$ to display your image, and $\text{pylab.gray()}$ to set the color palette to grayscale.
Figure 2: An inversion fractal and the circles that generate it. The fractal is self-inverse with respect to each of the blue circles and the blue vertical line, i.e., its shape does not change if every point in it is inverted with respect to any of those circles or the line.
Figure 3: Schematic illustration of the linear mapping between from \((x, y)\) coordinates to [row, col] array indices for a two-dimensional array of size 1000 × 1000 pixels.