Fractals

Consider a complex number $z = a + bi$ as a point (a, b) or vector in the Real Euclidean plane [1, i] with modulus $|z|$ the length of the vector and equal to $\sqrt{a^2 + b^2}$.

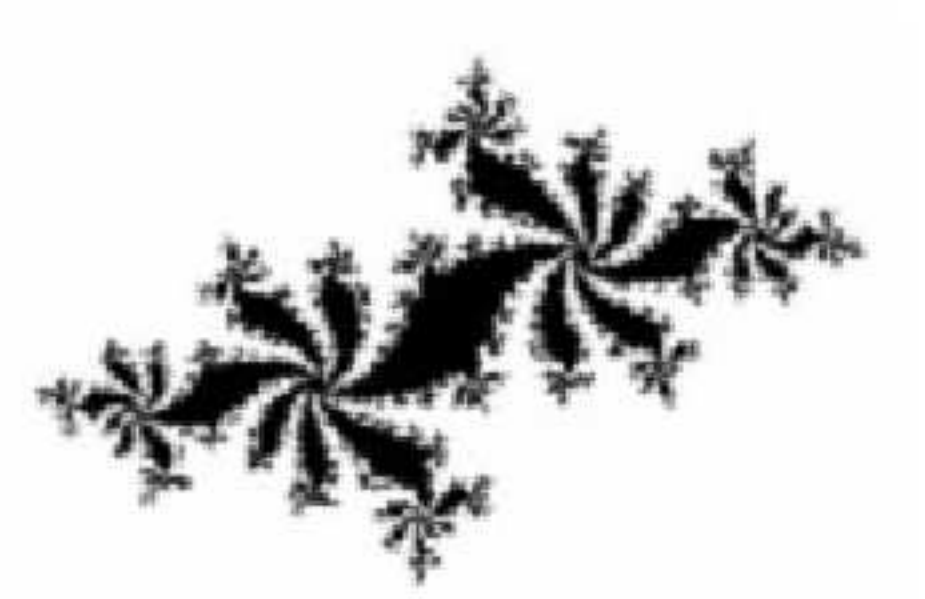
Complex arithmetic rules:

$$(a + bi) + (c + di) = (a + c) + (b + d)i$$
$$(a + bi)(c + di) = (ac - bd) + (ad + bc)i$$

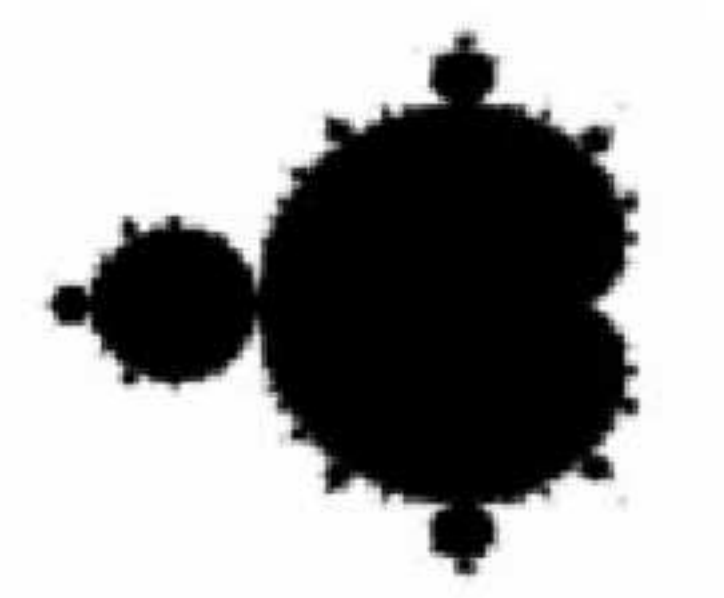
$z \rightarrow z^2$

All numbers with modulus 1 will stay at modulus 1 and is the *attractor set* or *fixed-point* of this **iterated function system**.

Julia Set for the point c: The attractor set of the iterated function system $z \rightarrow z^2 + c$ with c a complex constant



Julia Set for $c = -0.62 - 0.44i$



Mandelbrot Set: Color the point c black if **Julia** (c) is connected, and *white* otherwise.

Fractal Dimension:

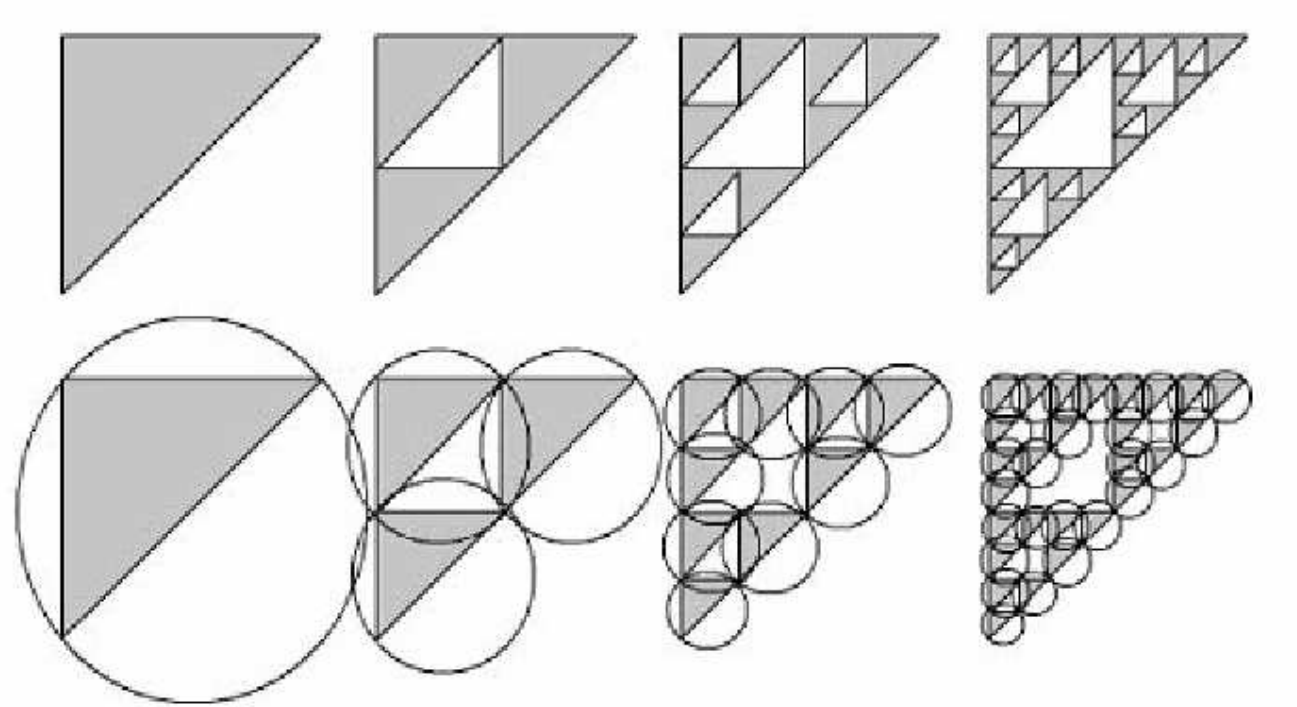
$$N(A, \epsilon) = \text{smallest number of } \epsilon\text{-balls needed to cover } A.$$

Object A has dimension d if $N(A, \epsilon)$ grows as $C(1/\epsilon)^d$ for constant C

$$\text{Fractal dimension } d = \lim_{\epsilon \to 0} \frac{\ln N(A, \epsilon)}{\ln(1/\epsilon)}$$

A **fractal** is an object which is *self-similar at different scales* and has a *non-integer fractal dimension*

$$\begin{aligned} d &= \lim_{\epsilon \to 0} \frac{\ln N(A, \epsilon)}{\ln(1/\epsilon)} \\ &= \lim_{k \to \infty} \frac{\ln N(A, (1/2^k))}{\ln(1/(1/2^k))} \\ &= \lim_{k \to \infty} \frac{\ln 3^k}{\ln 2^k} = \lim_{k \to \infty} \frac{k \ln 3}{k \ln 2} \\ &= \lim_{k \to \infty} \frac{\ln 3}{\ln 2} = \frac{\ln 3}{\ln 2} \approx 1.58496. \end{aligned}$$



The Sierpinski triangle covered by 3^k $(1/2^k)$-balls

Repeated Subdivision rule:

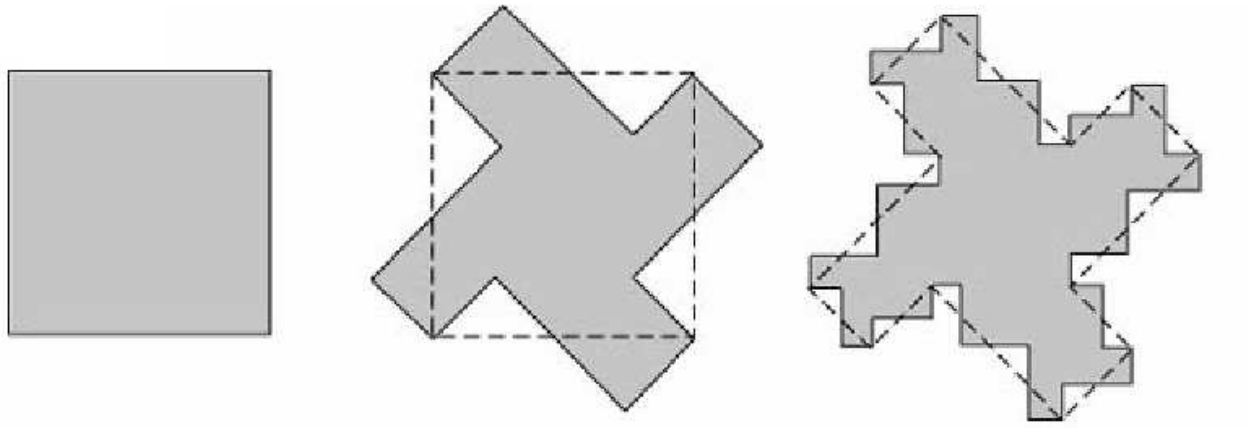
Replace each piece of length x by b nonoverlapping piece of length x/a.

Fractal dimension is

$$d = \frac{\ln b}{\ln a}$$

For object below the area doesn't change but boundary length does. The fractal dimension is

$$\frac{\ln 4}{\ln(2\sqrt{2})} = 1.3333.$$

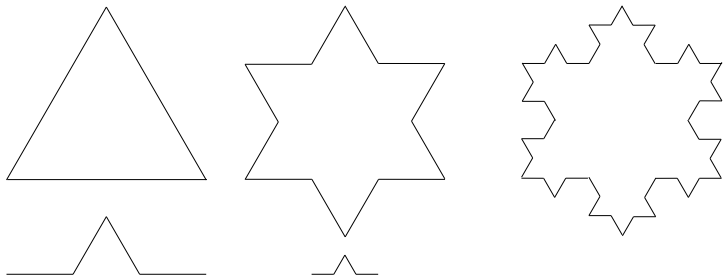


An object with a fractal boundary via repeated subdivision.

L-Systems (Lindenmayer-Systems)

- Aristid Lindenmayer, a botanist, initially developed this as a mathematical theory for modeling plants
- Przemyslaw Prusinkiewicz (Dr. P.) fleshed this out for Graphics Modeling applications
- Central concept is of string rewriting, using productions or rewriting rules (e.g. F $\rightarrow$ F + F - - F + F with all symbols +, - as characters not operators)
- Longer strings can be generated by repeated applications of the productions, starting from an axiom (e.g. F $\rightarrow$ F + F - - F + F $\rightarrow$ F + F - - F + F + F + F - - F + F - - F + F - - F + F + F + F - - F + F)
- See (http://mathforum.org/advanced/robertd/lsys2d.html) for other examples.

String Re-Writing and Turtle Graphics



- Turtle is a hypothetical drawing cursor on the screen or object coordinate system. Initially assume Turtle at origin (0,0) and facing UP.
- Interpret F as "Move turtle forward one unit and draw a line segment"
- Interpret - by "Turn counter-clockwise (ccw) by $\frac{\pi}{3}$"
- Interpret + "Turn clockwise (cw) by $\frac{\pi}{3}$"
- So then the string - F - - F - - F intepreted in Turtle graphics shall draw a triangle.
- Applying the production (or rule) F $\rightarrow$ F + F - - F + F once to the axiom (- F - - F - - F) yields a Star.
- Iterated applications of this rule, yields the Koch snowflake fractal.

Using Recursion

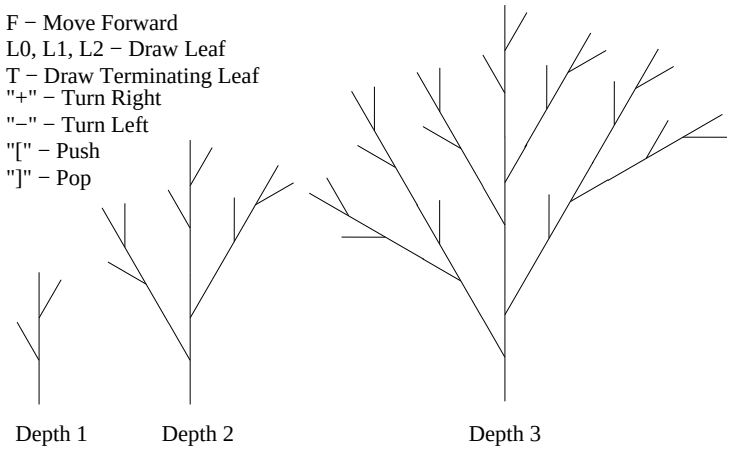
A rewriting rule can be captured by a recursive function. Implement the turtle as the matrix which describes the current object coordinate system, and "turn left" and "turn right" functions turn the turtle by an angle $\frac{\pi}{3}$.

```
drawbump(i){
if(i==0){ draw line() }
else {
drawbump(i-1);  turn left ();  drawbump(i-1);  turn right ();  turn right ();
drawbump(i-1);  turn left (); drawbump(i-1) }
 }
```

and the initial triangle (drawflake) is the function that starts the recursion

```
drawflake(i){
initialize(); turn left ();  drawbump(i);   turn right ();  turn right ();
drawbump(i);  turn right ();  turn right ();   drawbump(i) }
 }
```

Constructing Trees




```
Axiom: L0
1. L0 -> F [ - F L1 ] F [ + F L2 ] F L0   (center branch)
2. L1 -> F [ - F L1 ] F [ + F T  ] F L1   (left half of tree)
3. L2 -> F [ - F T  ] F [ + F L2 ] F L2   (right half of tree)
```

- The Turtle can make wiggly paths, but not branching.
- For branching we use a Stack, and the L-system symbols [for Push, and] as Pop
- A stack can be implemented using OpenGl operators PushMatrix() and PopMatrix () or the Program Stack implicit in Recursion).

Constructing Trees II

Recursive psuedo code an example L-system for constructing trees.

```
drawleaf(i){
 if(i==0){ actual-draw-leaf() }
else {
drawbranch(i-1); pushState() ; turn left ();  drawleaf (i-1);  popState();
pushState();  turn right ();  drawleaf(i-1); popState() }
 }
drawbranch(i){
 if(i==0){ actual-draw-branch() }
else { grow(); drawbranch(i-1)}
 }
drawtree(i){
initialize();  drawleaf(i) }
 }
```

Note "actual-draw-branch" changes turtle position, while "actual-draw-leaf" does not.

Model Transformations

- Model Turtle (position, direction, size) by a Matrix C
- We use OpenGL by loading C into MODELVIEW matrix.
- Assume Turtle initially at origin (0,0) and facing UP :(0,1). This initial position is captured in C by the identity matrix
- Now we wish to find the model transformation that moves Turtle to (50,100) and facing an angle $\frac{5\pi}{6}$ measured ccw from the x-axis.
- One easy way is by the sequence of Modelling transformations T(50,100) R($\frac{\pi}{3}$) applied to C. Remember, the transformations need to be applied in the correct right2left order.
- Next if we wish to move the turtle foward by 10 units in the direction its facing, we mutiply the sequence of transformations by another translation T(8.66,5).
- Note carefully, how we derived these transformations !

Using Program Stack for Recursion

- Use Recursion to replace PushMatrix (), PopMatrix() pairs with save and restore of the current matrix in the resolution of recursive calls by the Program Stack. For example replace

  PushMatrix()

  TurnRight() DrawLeaf(i-1)

  PopMatrix()
- with a call to a new function, say DrawRightleaf(i) which would be something like:

  DrawRightLeaf(i) Double[9]SavedMatrix;

  Copy(C,SavedMatrix); TurnRight(); DrawLeaf(i-1); copy(SavedMatrix,C)
- Here the DrawRightLeaf() function does not change C. Note that some L-system functions do change C; for example "turn left", "turn right", "grow", "shrink", and you need to keep track. The best way to remember of course while programming is via your code comments.

Reading Assignment and News

Before the next class please review Chapter 10 and its practice exercises, of the recommended text.

(Recommended Text: Interactive Computer Graphics, by Edward Angel, Dave Shreiner, 6th edition, Addison-Wesley)

Please track Blackboard for the most recent Announcements and Project postings related to this course.

(http://www.cs.utexas.edu/users/bajaj/graphics2012/cs354/)