Recursion: Dragon Curve

 n = 0

 n = 1

 n = 2

 n = 3

Drawing a Dragon Curve

Use simplest turtle graphics
- F: move forward one step (pen down)
- L: turn left
- R: turn right

 n = 0 F

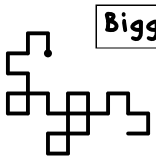
 n = 1 F L F

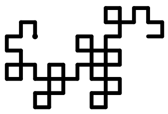
 n = 2 F L F L F R F

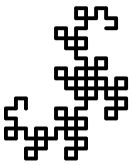
 n = 3 L L R L L R R

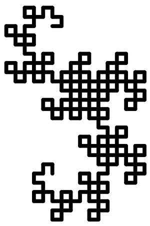
 n = 4 L L R L L R R L L L R R L R R

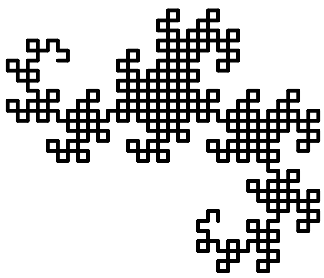
Bigger Dragons

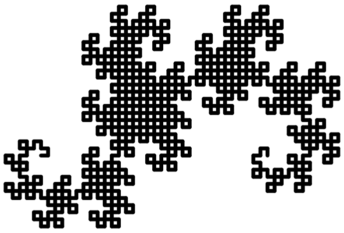


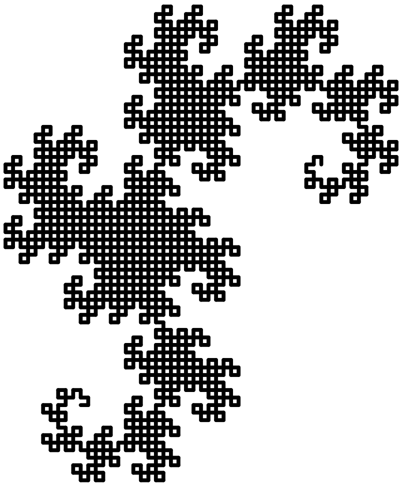












Recursive program for dragon curve

A dragon curve of order n is

* dragon curve of order n-1
* L
* dragon curve of order n-1, backwards
  read right to left, R for L

```
dragon(int n)
{
  if (n == 0)  F();
  else
    {
      dragon(n-1);
      L();
      nogard(n-1);
    }
}
```

Still need implementation of ``nogard``

Backwards dragon curve

To get ``nogard`` string from ``dragon`` string

* reverse the string
* switch L and R

dragon: F L F L F R F

nogard: F L F R F R F

D(N) = D(N-1) L revrs(D(N-1))

revrs(D(N)) = revrs(D(N-1) L revrs(D(N-1)))

revrs(D(N)) = D(N-1) R revrs(D(N-1))

```
nogard(int n)
{
  if (n == 0)  F();
  else
    {
      dragon(n-1);
      R();
      nogard(n-1);
    }
}
```

Unwinding Recursion: one step

Replace "nogard" call with its results

```
nogard(int n)
{
  if (n == 0)  F();
  else
    {
      dragon(n-1);
      R();
      nogard(n-1);
    }
}
```

```
nogard(int n)
{
  if (n == 0)  F();
  else
    {
      dragon(n-1);
      R();
      dragon(n-2);
      R();
      nogard(n-2);
    }
}
```

Still only one call on nogard, at the end

Nonrecursive "nogard"

```
nogard(int n)
{
  if (n == 0)  F();
  else
    {
      dragon(n-1);
      R();
      dragon(n-2);
      R();
      dragon(n-3);
      R();
      dragon(n-4);
      R();
      .
      .
      .
      dragon(2);
      R();
      dragon(1);
      R();
      dragon(0);
      R();
      nogard(0);
    }
}
```


Alternate "dragon"

Replace call to "nogard"
by nonrecursive version

```
dragon(int n)
{
  int k;
  if (n==0)
    { F();
      return;
    }
  dragon(n-1);
  L();
  for (k = n-2; k >= 0; k--)
    {
      dragon(k);
      R();
    }
  F();
}
```

F L F L F R F L F L F R F R F

F L F L F R F L F L F R F R F

Points out self-similarities in curve