Graphical Recursion II
Announcements & Logistics

- **No in person lecture today**: watch recording on GLOW
- **Lab 7** is due Wed/Thursday at 10 pm
- Come to scheduled lab today/tomorrow as usual!
- **HW 6** due tonight, 10 pm
- Midterm graded feedback was returned on Friday Oct 29
  - If you did not pick it up, stop by office hours
  - Shikha's office hours today **Mon 3-5 pm**
- **Scheduled final**: December 18, 9.30 am

Do You Have Any Questions?
Single Recursive Call: **Recursive Spirals**

\[
\text{spiral}(200, 90, 0.9, 10) \quad \text{spiral}(200, 72, 0.97, 10) \quad \text{spiral}(200, 80, 0.95, 10) \\
\text{spiral}(200, 121, 0.95, 15) \quad \text{spiral}(200, 95, 0.93, 10)
\]
Recursive Spirals

sideLen * shrinkFactor * shrinkFactor

sideLen

sideLen * shrinkFactor
Function Frame Model to Understand spiral
def spiral(sideLen, angle, scaleFactor, minLength):
    """Draw a spiral recursively."""
    if sideLen >= minLength:
        fd(sideLen)
        lt(angle)
        spiral(sideLen*scaleFactor, angle, scaleFactor, minLength)
spiral(625, 90, 0.8, 250)

sideLen = 625

if sideLen > 250:
    fd(sideLen)
    lt(90)
    spiral(500, ...)

625
spiral(625, 90, 0.8, 250)

sideLen 625

if sideLen > 250:
    fd(sideLen)
    lt(90)
spiral(500, ...)

625
spiral(625, 90, 0.8, 250)

sideLen  625  .......

if sideLen > 250:
    fd(sideLen)
    lt(90)
    spiral(500, ...)

spiral(500, 90, 0.8, 250)

sideLen  500

if sideLen > 250:
    fd(sideLen)
    lt(90)
    spiral(400, ...)

625
spiral(625, 90, 0.8, 250)

sideLen  625  .......

if sideLen > 250:
    fd(sideLen)
    lt(90)
    spiral(500, ...)

spiral(500, 90, 0.8, 250)

sideLen  500

if sideLen > 250:
    fd(sideLen)
    lt(90)
    spiral(400, ...)

spiral(625, 90, 0.8, 250)

sideLen 625

if sideLen > 250:
    fd(sideLen)
    lt(90)
    spiral(500, ...)

spiral(500, 90, 0.8, 250)

sideLen 500

if sideLen > 250:
    fd(sideLen)
    lt(90)
    spiral(400, ...)
spiral(625, 90, 0.8, 250)

sideLen 625

if sideLen > 250:
    fd(sideLen)
    lt(90)
    spiral(500, ...)

spiral(500, 90, 0.8, 250)

sideLen 500

if sideLen > 250:
    fd(sideLen)
    lt(90)
    spiral(400, ...)

spiral(400, 90, 0.8, 250)

sideLen 400

if sideLen > 250:
    fd(sideLen)
    lt(90)
    spiral(320, ...)
spiral(625, 90, 0.8, 250)

if sideLen > 250:
    fd(sideLen)
    lt(90)
    spiral(500, ...)

spiral(500, 90, 0.8, 250)

if sideLen > 250:
    fd(sideLen)
    lt(90)
    spiral(400, ...)

spiral(400, 90, 0.8, 250)

if sideLen > 250:
    fd(sideLen)
    lt(90)
    spiral(320, ...)

spiral(320, 90, 0.8, 250)
Invariant Spiralling
Invariance

• A function is invariant relative to an object's state if the state of the object is the same before and after a function is invoked.

```
Do state change 1
Do state change 2
...
Do state change n-1
Do state change n
```

Perform changes to state

Recursive call to function

```
Undo state change n
Undo state change n-1
...
Undo state change 2
Undo state change 1
```

Undo state changes in opposite order

Source: [http://cs111.wellesley.edu/spring19](http://cs111.wellesley.edu/spring19)
Recursive Trees

tree(trunkLen, levels, angle, shrinkFactor)

- **trunkLen**: is the length of the base trunk of the tree
- **levels**: is the number of branches on any path from root to leaf
- **angle**: is the angle from the trunk of the right and left branches
- **shrinkFactor**: is the factor by which the trunkLen of the branches goes down by
How to make a 4-level tree?

\[ \text{tree(100, 4, 45, 0.6)} \]

Step 1
Make a trunk of size 100

Step 2
and two 3-level trees with 60% trunks set at 45° angles
How to make a 3-level tree?

Make a trunk of size 60

\[ \text{tree}(60, 3, 45, 0.6) \]

and two 2-level trees with 60% trunks set at 45° angles

Make a trunk of size 36

\[ \text{tree}(36, 2, 45, 0.6) \]

and two 1-level trees with 60% trunks set at 45° angles

Make a trunk of size 21.6

\[ \text{tree}(21.6, 1, 45, 0.6) \]

and two 0-level trees set at 45° angles

How to make a 2-level tree?

How to make a 1-level tree?

Do nothing!

\[ \text{tree}(12.96, 0, 45, 0.6) \]

How to make a 0-level tree?
Trees

tree(200, 3, 45, 0.6)

tree(200, 7, 15, 0.8)

tree(200, 7, 30, 0.8)

tree(200, 10, 45, 0.7)
Function Frame Model to Understand \texttt{tree(60, 3, 45, 0.6)}
Draw trunk and turn to draw level 2 tree

tree(3, 60, 45, 0.6)
fd(60)
rt(45)
Begin recursive invocation to draw level 2 tree

```
tree(2,36,45,0.6)
```

```
tree(3,60,45,0.6)
```
```
fd(60)
rt(45)
tree(2,36,45,0.6)
```
Draw trunk and turn to draw level 1 tree

```
tree(2, 36, 45, 0.6)
fd(36)
rt(45)
```

```
tree(3, 60, 45, 0.6)
fd(60)
rt(45)
tree(2, 36, 45, 0.6)
```
Begin recursive invocation to draw level 1 tree
Draw trunk and turn to draw level 0 tree

```
tree(3,60,45,0.6)
  fd(60)
  rt(45)
  tree(2,36,45,0.6)
```

```
tree(2,36,45,0.6)
  fd(36)
  rt(45)
  tree(1,21.6,45,0.6)
```

```
tree(1,21.6,45,0.6)
  fd(21.6)
  rt(45)
```
Begin recursive invocation to draw level 0 tree

```
fd(60)
rt(45)
tree(2,36,45,0.6)
```

```
fd(36)
rt(45)
tree(1,21.6,45,0.6)
```

```
tree(2,36,45,0.6)
```

```
fd(21.6)
rt(45)
tree(0,12.96,45,0.6)
```

```
tree(1,21.6,45,0.6)
```

```
tree(3,60,45,0.6)
```

Source: http://cs111.wellesley.edu/spring19
Complete level 0 tree and turn to draw another level 0 tree
Begin recursive invocation to draw level 0 tree

```
tree(60, 3, 45, 0.6)
```

Source: http://cs111.wellesley.edu/spring19
Complete level 0 tree and return to starting position of level 1 tree

```
fd(36)
rt(45)
tree(1,21.6,45,0.6)
```

```
fd(21.6)
rt(45)
tree(0,12.96,45,0.6)
lrt(90)
tree(0,12.96,45,0.6)
rt(45)
bk(21.6)
```

```
fd(60)
rt(45)
tree(2,36,45,0.6)
```

```
tree(3,60,45,0.6)
```

Source: http://cs111.wellesley.edu/spring19
Complete level 1 tree and turn to draw another level 1 tree
Begin recursive invocation to draw level 1 tree
Draw trunk and turn to draw level 0 tree

```
fd(60)
rt(45)
tree(2,36,45,0.6)
```

```
fd(36)
rt(45)
tree(1,21.6,45,0.6)
lt(90)
tree(1,21.6,45,0.6)
```

```
fd(21.6)
rt(45)
tree(0,12.96,45,0.6)
lt(90)
tree(0,12.96,45,0.6)
```

```
tree(1,21.6,45,0.6)
```

```
fd(21.6)
rt(45)
```

```
tree(1,21.6,45,0.6)
```

```
tree(3,60,45,0.6)
```

```
fd(60)
rt(45)
tree(2,36,45,0.6)
```
Complete two level 0 trees and return to starting position of level 1 tree.
Complete level 1 tree and return to starting position of level 2 tree:

```
fd(60)
rt(45)
tree(2,36,45,0.6)
```

```
fd(36)
rt(45)
tree(1,21.6,45,0.6)
lt(90)
tree(1,21.6,45,0.6)
rt(45)
bk(36)
```

```
tree(2,36,45,0.6)
```

```
tree(1,21.6,45,0.6)
fd(21.6)
rt(45)
tree(0,12.96,45,0.6)
lt(90)
tree(0,12.96,45,0.6)
rt(45)
bk(21.6)
```

```
tree(1,21.6,45,0.6)
fd(21.6)
rt(45)
tree(0,12.96,45,0.6)
lt(90)
tree(0,12.96,45,0.6)
rt(45)
bk(21.6)
```
Complete level 2 tree and turn to draw another level 2 tree.
Draw trunk and turn to draw level 1 tree
Draw trunk and turn to draw level 0 tree

```plaintext
tree(1, 21.6, 45, 0.6)
  fd(21.6)
  rt(45)
  tree(0, 12.96, 45, 0.6)
    lt(90)
    tree(0, 12.96, 45, 0.6)
      rt(45)
      bk(21.6)
    tree(1, 21.6, 45, 0.6)
      fd(21.6)
      rt(45)
      tree(0, 12.96, 45, 0.6)
        lt(90)
        tree(0, 12.96, 45, 0.6)
          rt(45)
          bk(21.6)
        tree(1, 21.6, 45, 0.6)
          fd(21.6)
          rt(45)
          tree(0, 12.96, 45, 0.6)
            lt(90)
            tree(0, 12.96, 45, 0.6)
              rt(45)
              bk(21.6)
            tree(1, 21.6, 45, 0.6)
              fd(21.6)
```

```
tree(2, 36, 45, 0.6)
  fd(36)
  rt(45)
  tree(1, 21.6, 45, 0.6)
    lt(90)
    tree(1, 21.6, 45, 0.6)
      rt(45)
      bk(36)
    tree(2, 36, 45, 0.6)
      fd(36)
      rt(45)
      tree(2, 36, 45, 0.6)
        lt(90)
        tree(2, 36, 45, 0.6)
          rt(45)
          bk(36)
```
Complete two level 0 trees and return to starting position of level 1 tree
Complete level 1 tree and turn to draw another level 1 tree

```
    tree(1,21.6,45,0.6)
    fd(21.6)
    rt(45)
    tree(0,12.96,45,0.6)
    lt(90)
    tree(0,12.96,45,0.6)
    rt(45)
    bk(21.6)
```

```
  tree(1,21.6,45,0.6)
  fd(21.6)
  rt(45)
  tree(0,12.96,45,0.6)
  lt(90)
  tree(0,12.96,45,0.6)
  rt(45)
  bk(21.6)
```

```
    tree(2,36,45,0.6)
    fd(36)
    rt(45)
    tree(1,21.6,45,0.6)
    lt(90)
    tree(1,21.6,45,0.6)
    rt(45)
    bk(36)
```

```
  tree(3,60,45,0.6)
  fd(60)
  rt(45)
  tree(2,36,45,0.6)
  lt(90)
  tree(2,36,45,0.6)
```

```
    tree(2,36,45,0.6)
    fd(36)
    rt(45)
    tree(1,21.6,45,0.6)
    lt(90)
```

Source: http://cs111.wellesley.edu/spring19
Draw trunk and turn to draw level 0 tree

```
tree(3,60,45,0.6)
fd(60)
rt(45)
tree(2,36,45,0.6)
lrt(90)
tree(2,36,45,0.6)
```

```
tree(2,36,45,0.6)
fd(36)
rt(45)
tree(1,21.6,45,0.6)
lrt(90)
tree(1,21.6,45,0.6)
```

```
tree(1,21.6,45,0.6)
fd(21.6)
rt(45)
tree(0,12.96,45,0.6)
lrt(90)
tree(0,12.96,45,0.6)
```

```
tree(1,21.6,45,0.6)
fd(21.6)
rt(45)
tree(0,12.96,45,0.6)
lrt(90)
tree(0,12.96,45,0.6)
```

```
tree(1,21.6,45,0.6)
fd(21.6)
rt(45)
tree(0,12.96,45,0.6)
lrt(90)
tree(0,12.96,45,0.6)
```

```
```
Complete two level 0 trees and return to starting position of level 1 tree

Source: http://cs111.wellesley.edu/spring19
Complete level 1 tree and return to starting position of level 2 tree

```
fd(60)
rt(45)
tree(2,36,45,0.6)
lt(90)
tree(2,36,45,0.6)
```

```
fd(36)
rt(45)
tree(1,21.6,45,0.6)
lt(90)
tree(1,21.6,45,0.6)
```

```
fd(21.6)
rt(45)
tree(0,12.96,45,0.6)
lt(90)
tree(0,12.96,45,0.6)
```

```
tree(1,21.6,45,0.6)
fd(21.6)
rt(45)
tree(0,12.96,45,0.6)
lt(90)
tree(0,12.96,45,0.6)
```

```
tree(1,21.6,45,0.6)
fd(21.6)
rt(45)
tree(0,12.96,45,0.6)
lt(90)
tree(0,12.96,45,0.6)
```

Source: http://cs111.wellesley.edu/spring19
Complete level 2 tree and return to starting position of level 3 tree

```
tree(3, 60, 45, 0.6)
  fd(60)
  rt(45)
  tree(2, 36, 45, 0.6)
    lt(90)
    tree(2, 36, 45, 0.6)
    rt(45)
    bk(36)
    tree(2, 36, 45, 0.6)
      fd(36)
      rt(45)
      tree(1, 21.6, 45, 0.6)
      lt(90)
      tree(1, 21.6, 45, 0.6)
      rt(45)
      bk(36)
      tree(1, 21.6, 45, 0.6)
```
Trace the invocation of \( \text{tree}(3, 60, 45, 0.6) \)

1. \( \text{tree}(3, 60, 45, 0.6) \)
   - \( \text{fd}(60) \)
   - \( \text{rt}(45) \)
   - \( \text{tree}(2, 36, 45, 0.6) \)
     - \( \text{fd}(36) \)
     - \( \text{rt}(45) \)
     - \( \text{tree}(1, 21.6, 45, 0.6) \)
       - \( \text{lt}(90) \)
       - \( \text{tree}(0, 12.96, 45, 0.6) \)
     - \( \text{rt}(45) \)
     - \( \text{bk}(36) \)

2. \( \text{tree}(2, 36, 45, 0.6) \)
   - \( \text{fd}(36) \)
   - \( \text{rt}(45) \)
   - \( \text{tree}(1, 21.6, 45, 0.6) \)
     - \( \text{lt}(90) \)
     - \( \text{tree}(0, 12.96, 45, 0.6) \)
   - \( \text{rt}(45) \)
   - \( \text{bk}(36) \)

3. \( \text{tree}(1, 21.6, 45, 0.6) \)
   - \( \text{fd}(21.6) \)
   - \( \text{rt}(45) \)
   - \( \text{tree}(0, 12.96, 45, 0.6) \)
     - \( \text{lt}(90) \)
     - \( \text{tree}(0, 12.96, 45, 0.6) \)
   - \( \text{rt}(45) \)
   - \( \text{bk}(21.6) \)

4. \( \text{tree}(1, 21.6, 45, 0.6) \)
   - \( \text{fd}(21.6) \)
   - \( \text{rt}(45) \)
   - \( \text{tree}(0, 12.96, 45, 0.6) \)
     - \( \text{lt}(90) \)
     - \( \text{tree}(0, 12.96, 45, 0.6) \)
   - \( \text{rt}(45) \)
   - \( \text{bk}(21.6) \)

5. \( \text{tree}(2, 36, 45, 0.6) \)
   - \( \text{fd}(36) \)
   - \( \text{rt}(45) \)
   - \( \text{tree}(1, 21.6, 45, 0.6) \)
     - \( \text{lt}(90) \)
     - \( \text{tree}(0, 12.96, 45, 0.6) \)
   - \( \text{rt}(45) \)
   - \( \text{bk}(36) \)

6. \( \text{tree}(1, 21.6, 45, 0.6) \)
   - \( \text{fd}(21.6) \)
   - \( \text{rt}(45) \)
   - \( \text{tree}(0, 12.96, 45, 0.6) \)
     - \( \text{lt}(90) \)
     - \( \text{tree}(0, 12.96, 45, 0.6) \)
   - \( \text{rt}(45) \)
   - \( \text{bk}(21.6) \)

7. \( \text{tree}(1, 21.6, 45, 0.6) \)
   - \( \text{fd}(21.6) \)
   - \( \text{rt}(45) \)
   - \( \text{tree}(0, 12.96, 45, 0.6) \)
     - \( \text{lt}(90) \)
     - \( \text{tree}(0, 12.96, 45, 0.6) \)
   - \( \text{rt}(45) \)
   - \( \text{bk}(21.6) \)
Fruitful Recursion:
Branch Count
Random Trees
Sierpinski Triangle
Sierpinski Triangle

sierpinski(sideLen, level)

- **sideLen**: length of the outermost triangle
- **level**: determines # of subpatterns:
  - level = 0 nothing is drawn
  - level = 1 only a single triangle (no sub patterns)
  - level = $\ell$ has level $\ell - 1$ sierpinski triangles as its subpatterns
sierpinski(sideLen, level)

sierpinski(600, 1)  sierpinski(600, 2)  sierpinski(600, 3)

sierpinski(600, 4)  sierpinski(600, 5)  sierpinski(600, 6)
def drawTriangle(sideLen):
    """Draws triangle with sides of length sideLen starting from one end point""
    pd()
    fd(sideLen)
    lt(120)
    fd(sideLen)
    lt(120)
    fd(sideLen)
    lt(120)
    pu()
sierpinski(sideLen, level)

- First draw outer big triangle
- Then recursively
  - Draw upper triangle
  - Draw upper left
  - Draw lower right
Acknowledgments

These slides have been adapted from:

• http://cs111.wellesley.edu/spring19 and