Previous Lecture:
- Iteration using for

Today’s Lecture:
- Iteration using while
- Review loops, conditionals using graphics

Announcements:
- Discussion this week in the classrooms is listed on Student Center
- Watch MatTV: Troubleshooting Loops. Available on course website
- We do not use break in this course
- Register your iClicker! See link on course website.

Example: \( n \)-gon \( \rightarrow \) circle

- Inscribed hexagon
  \[ \frac{n}{2} \sin\left(\frac{2\pi}{n}\right) \]
- Circumscribed hexagon
  \[ n \tan\left(\frac{\pi}{n}\right) \]

As \( n \) approaches infinity, the inscribed and circumscribed areas approach the area of a circle.

When will \(|\text{OuterA} - \text{InnerA}| \leq .000001\)?

---

Find \( n \) such that \( \text{outerA} \) and \( \text{innerA} \) converge

First, itemize the tasks:
- Define how close is close enough
- Select an initial \( n \)
- Calculate \( \text{innerA}, \text{outerA} \) for current \( n \)
- \( \text{diff} = \text{outerA} - \text{innerA} \)
- Close enough?
- If not, increase \( n \), repeat above tasks

Now organize the tasks \( \rightarrow \) algorithm:

- \( n \) gets initial value
- \( \text{innerA}, \text{outerA} \) get initial values
- Repeat until difference is small:
  - Increase \( n \)
  - Calculate \( \text{innerA}, \text{outerA} \) for current \( n \)
  - \( \text{diff} = \text{outerA} - \text{innerA} \)

Guard against infinite loop

Use a loop guard that guarantees termination of the loop. Or just limit the number of iterations.

- While \( (B_n-A_n > \delta \land n < n_{\text{Max}}) \)

---

\text{areaCircle.m}
Another use of the while-loop: user interaction

- Example: Allow a user to repeatedly calculate the inscribed and circumscribed areas of n-gons on a unit circle.
- Need to define a “stopping signal”

Common loop patterns

- Do something n times
  ```matlab
  for k= 1:1:n
    % Do something
  end
  ```
- Do something an indefinite number of times
  ```matlab
  %Initialize loop variables
  while ( not stopping signal )
    % Do something
    % Update loop variables
  end
  ```

Important Features of Iteration

- A task can be accomplished if some steps are repeated; these steps form the loop body
- Need a starting point
- Need to know when to stop
- Need to keep track of (and measure) progress

In Matlab, which claim is true? (without `break`)

- A: for-loop can do anything while-loop can do
- B: while-loop can do anything for-loop can do
- C: for- and while-loops can do the same things

Pattern to do something n times

```matlab
for k= 1:1:n
  % Do something
  %Initialize loop variables
  k= 1;
  while ( k <= n )
    % Do something
    % Update loop variables
    k= k+1;
  end
end
```

for-loop or while-loop: that is the question

- **for-loop**: loop body repeats a fixed (predetermined) number of times.
- **while-loop**: loop body repeats an indefinite number of times under the control of the “loop guard.”
Review loops/conditionals using user-defined graphics function

Draw a black square; then draw a magenta disk; then draw a yellow star.

DrawRect(-1,-2,6,3,'y')

DrawDisk(1,3,4,'r')

DrawStar(1,3,4,'g')

Color Options

<table>
<thead>
<tr>
<th>Color</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>White</td>
<td>'w'</td>
</tr>
<tr>
<td>Black</td>
<td>'k'</td>
</tr>
<tr>
<td>Red</td>
<td>'r'</td>
</tr>
<tr>
<td>Blue</td>
<td>'b'</td>
</tr>
<tr>
<td>Green</td>
<td>'g'</td>
</tr>
<tr>
<td>Yellow</td>
<td>'y'</td>
</tr>
<tr>
<td>Magenta</td>
<td>'m'</td>
</tr>
<tr>
<td>Cyan</td>
<td>'c'</td>
</tr>
</tbody>
</table>

% drawDemo
close all
figure
axis equal off
hold on

DrawRect(0,0,2,2,'k')
DrawDisk(1,1,1,'m')
DrawStar(1,1,1,'y')

hold off
A general graphics framework

```matlab
% drawDemo
close all
figure
axis equal off
hold on

Code fragment to draw the objects (rectangle, disk, star)

hold off
```

Example: Nested Stars