While Loops

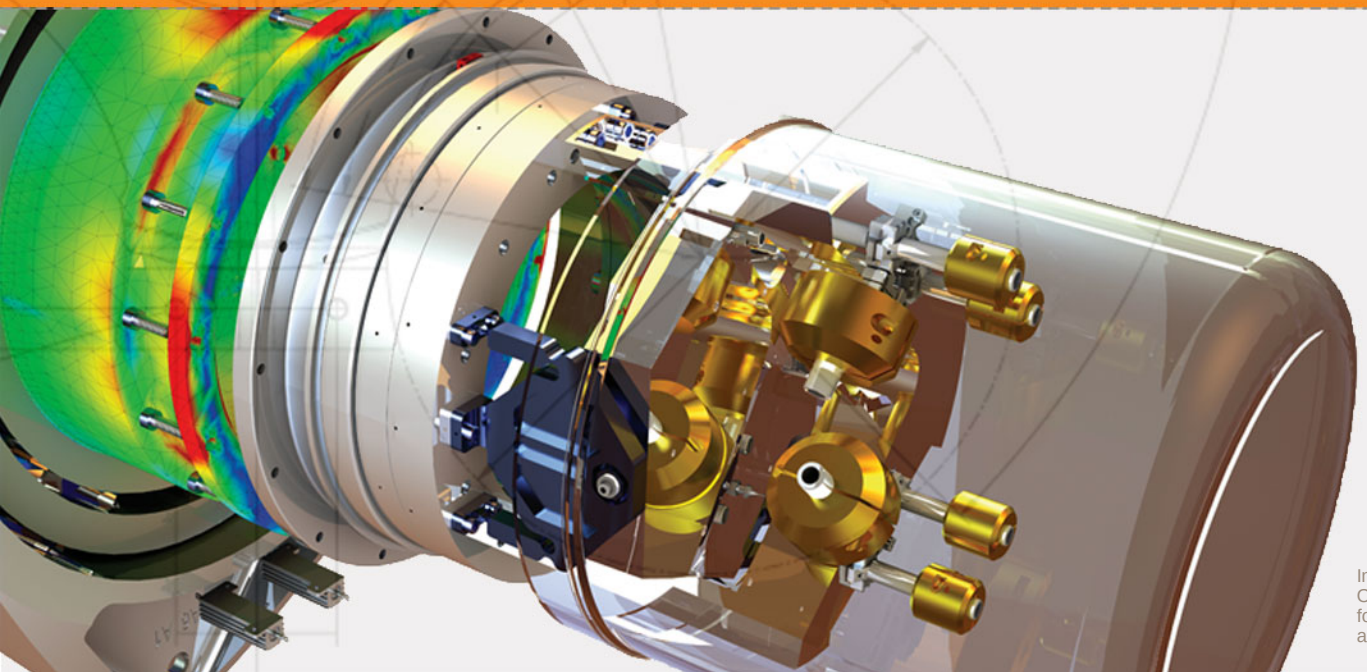

Image courtesy of National Optical Astronomy Observatory, operated by the Association of Universities for Research in Astronomy, under cooperative agreement with the National Science Foundation.

Why While Loops?

- Consider the following situation:
  - Someone has been stealing your cookies! To figure out who, you are writing a script to act as a security system for your room. You need to take a measurement from a motion sensor every 10 seconds and take a picture if the motion sensor detects someone near the cookie jar.
- Can you write this program using a For loop?
  - No, you don't know if/when someone will try to steal your cookies!

While Loops

WHILE loops repeat until some condition is no longer met

Construction:

```
while  conditional expression
       MATLAB statements
end
```

- As long as the expression is true, the MATLAB statements in the while loop will continue to execute
- If the expression is initially false, the while loop will never execute
- If the expression becomes false, the loop will terminate

While Loops

Example:

```
while  a >= 0
       MATLAB statements
end
```

- a has to be defined in the program before you hit the while loop, otherwise the program won't run
- If a is negative, the while loop will not execute at all
- If a is positive, the while loop will continue to execute as long as it stays positive

What happens if a is positive and none of the statements in the loop ever change a?

Infinite Loop!

Example 2: Simple WHILE Loop

```
sum = 50;
while sum < 100
    sum = sum + 8;
end
disp(sum)
```

```
Final Value for sum?
        106
# Times thru loop?
         7
```

sum	Loop?

Common Misconception about WHILE Loops

Many first-time programmers think that the condition for the while loop is checked continually as the loop executes and as soon as the condition is false, the loop immediately terminates – **this isn't true**

- If the while condition is true, the loop will completely execute to the **end** statement, even if the condition becomes false somewhere in the middle of the loop
- Once the loop has reached the **end** statement, the condition will be re-evaluated to determine whether or not the program should go through the loop again

Examples

- What is the output of the following code?

```
x = 1;
while x < 10
     fprintf('%d \n', x);
     x = x+1;
end
```

Output:

```
1
2
3
4
5
6
7
8
9
```

Examples

- What is the output of the following code?

```
x = 1;
exitFlag = true;
while exitFlag
    fprintf('%d \n', x);
    if x > 10
        exitFlag = false;
    end
    x = x+1;
end
```

Output:

```
1
2
3
4
5
6
7
8
9
10
11
```

Examples

- What is the output of the following code?

```
x = 1;
while x < 10
     if (x < 4 || x > 7)
          fprintf('%d', x);
     end
     fprintf('\n');
     x = x+1;
end
```

Output:

```
1
2
3


8
9
```

Examples

- What is the output of the following code?

```
x = 1;
while (x > 1)
    fprintf('%d \n', x);
    x = x-1;
end
```

Output:

```
Nothing!!!
```

Examples

- Using a while loop, you can add error checking to your program to ensure the user enters values correctly
  - Require the user to enter a number between 1 and 10 before continuing:

```
x = 0;
while (x < 1) || (x > 10)
  x = input('Please enter a number: ');
  if (x < 1) || (x > 10)
      fprintf('Number must be between 1 and 10.\n');
  end
end
```