Note: thanks to Wanda Dann and Steve Cooper for slide ideas
Announcements

• Read Chapter 6 Tips and Techniques for next time
• Assignment 7 out
Last time – if/else statements

```
World.avoidCollision
```

```
World.avoidCollision  Obj aircraftOne, Obj aircraftTwo
```

```
No variables
```

```
If
   aircraftOne is above aircraftTwo more...
```

```
Do together
   aircraftOne move up 5 meters more...
   aircraftTwo move down 5 meters more...
```

```
Else
   Do together
   aircraftOne move down 5 meters more...
   aircraftTwo move up 5 meters more...
```
Last time – Boolean functions

Use Function in a method:
What we will do today

• Lecture on Chap 6, Part 2
  – Execution control
  – Nested ifs
  – Logical Operators

• Classwork
Revisit the Zeus world

- Recall the Zeus world

Testing this world – two problems
- Anything clicked gets zapped
- Philosophers can get zapped more than once
Problem

• We need conditional execution
  – Check conditions
    • Is selected object a philosopher?
    • If so, has philosopher already been zapped?
  – Conditional execution
    • Lightning bolt will be shot or not
Multiple Conditions

• First problem to solve - restrict Zeus to shoot only philosophers
• Four possible conditions must be checked
  – User could click on one of four philosophers
One Condition

- Begin with just one philosopher - homer

Start with blank if statement

Drag in the who tile, then select == and homer

== is a relational operator meaning “is equal to”
Add in *Do in order* block

The *Do in order* block is **nested** inside the *if*

Now, lightning will strike homer, only if user clicks on homer. But, what about the other three philosophers?
Logical Operators

Use Boolean logic operators to check more than one condition

homer or plato or socrates or euripides
Demo

- Demo Zeus world with *if* statement and logic operators
Abstraction

- Multiple conditions, as in this example,
  - become complex
  - are difficult to read
  - are difficult to debug
- Better solution – write a Boolean function that checks the conditions
- This is an example of abstraction – allowing you to think on a higher plane
isPhilosopher function

- Used nested if statements

- If one of the conditions is true, function returns true, and nothing else in the function is checked!

- If none of the conditions are true, the function returns false
Demo of isPhilosopher function
Completing the Zeus world

- Second problem – how to prevent lightning striking the same philosopher more than once
- How do we know if a philosopher has already been struck by lightning?
  - When zapped, *color* property is set to black
  - Checking color is convenient way to check for a previous lightning strike
Preventing a duplicate strike

• Only need to check for a possible duplicate-strike if we already know a philosopher is clicked
  – Nest a second if statement inside the first

!= is a relational operator meaning “is not equal to”
Complete shootBolt

World.shootBolt

No variables

- If
  World.isPhilosopher possiblePhilosopher = who

- If
  who.color !=

- Do in order
  World.prepareToShoot target = who
  World.lightningAndThunder target = who

// move lightning back to cloud

lightning move to <None> offset by = cloud 's position more...

Else
  zeus say That philosopher is already zapped!!! more...

Else
  zeus say I only shoot at philosophers! more...
Demo

• Demo of completed Zeus world
Classwork today

• New partners
• Redo the ball and goals animation
  – Start over
• Demo