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

• Read Chapter 6.1 for next time
• Midterm next Tuesday, Chap 1-2,4-5
• Assignment 5 due today
What we will do today

• Lecture on Chap 5, Sec 2
  – Event-handlers with parameters

• Classwork
Mouse Clicks

• Interactive programs – allow user to mouse click an object
  – Buttons in an interface
  – Targets in a game
  – Checklist of items on a form

• Will see how to pass information about a mouse clicked object to an event handler
Example

- People are trapped in a burning building
- Select which person will be rescued
Storyboard

- Three people are to be rescued
- Could write 3 different methods

**Event:** click on guy1

**Responding Method:**
Save guy on first floor

**Event:** click on girl2

**Responding Method:**
Save girl on second floor

**Event:** click on girl3

**Responding Method:**
Save girl on third floor
A Better Solution

- Write one event handler
- Send in information needed for action

```plaintext
firetruck.savePerson:

parameters: whichFloor, whichPerson, howFar

Do in order
- point ladder at whichFloor
- extend ladder howFar meters
- whichPerson slides down ladder to fire truck
- pull ladder back howFar meters
```

What type are the parameters?
Demo

• Demonstration of the code for `firetruck.savePerson`
Three Events

- The argument sent to parameters depends on which person is mouse clicked

• Note - we positioned fire truck so distance from floor X is X meters (to floor 3 is 3 meters)
Example 2

• Zeus was a powerful god in Greek mythology. When Zeus was angry, he would shoot a thunderbolt out of the heavens to strike anyone who got in the way.

• The user will choose the philosopher who will be the next target of Zeus’ anger.
Storyboard

- Possible design – method with Object parameter named *who*, for object clicked

<table>
<thead>
<tr>
<th>Event: an object is mouse-clicked</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event handler: <em>shootBolt</em></td>
</tr>
<tr>
<td>Parameter: who – object clicked</td>
</tr>
<tr>
<td>Do in order</td>
</tr>
<tr>
<td>prepare to strike object that was clicked</td>
</tr>
<tr>
<td>thunder plays and lightning strikes object clicked</td>
</tr>
<tr>
<td>lightning is repositioned for next strike</td>
</tr>
</tbody>
</table>

- The actions in storyboard are complex
- Break actions into simpler steps using stepwise refinement
**Event:** an object is mouse-clicked

**Event handler:** `shootBolt`

**Parameter:** `who` – object clicked

Do in order

- prepare to strike object that was clicked
- thunder plays and lightning strikes object clicked
- lightning is repositioned for next strike

**prepareToShoot**

**Parameter:** `target`

Do together

- turn Zeus to face the `target`
- make the lightning bolt visible

**lightning and Thunder:**

**Parameter:** `target`

Do together

- play sound
- call `specialEffects` method
  - send `target`
specialEffects:

**parameter**: `target`

Do in order

Do together

lightning bolt move to `target`

smoke move to `target`

Do together

set smoke to visible

set lightning to invisible

call smoke cycle – built-in method

set `target` color to black

move `target` up and down
A Driver

- `shootBolt` method - top level of our design
- It calls other methods and controls the overall action of the program – we call this a **driver**
One Link

- In the fire rescue example, we used three links – one for each person in the burning building. In this example, we use only one link by selecting “object under mouse cursor” as the argument.
prepareToShoot

- In setting up initial scene, made lightning bolt invisible by setting its opacity to 0 (0%)
- To prepare to shoot lightning bolt, make it visible, set opacity back to 1 (100%)
lightningAndThunder

- Coordinate the sound of the thunder with lightning and other special effects
specialEffects

- Smoke.cycle is a built-in instruction with a duration of about 2 ½ seconds
move to

• Several statements in `shootBolt` and `specialEffects` methods use a move to instruction.

```
// move lightning back to cloud
```

```
lightning  move to  cloud  's position  more...  more...
```

• The move to instruction moves an object to a particular position in the world. In the example above, where is the lightning bolt moved to?
**Move to with an object parameter**

- If an object parameter is used to specify a target position, creating a *move to* statement takes two steps:
  1) Drag in arbitrary object as parameter
  2) Substitute the object parameter name

This is necessary because an object parameter is only a placeholder for an object – not an actual object.
Demo

• Test run of Zeus world
• When parameters are used in interactive programming – especially important to test that all possible parameter values work as expected
  – What happens if you click on each philosopher, one at a time?
• Also try things that shouldn’t work
  – What happens if you click on a column?
  – What happens if you click on a philosopher twice?
  – What happens if you click on Zeus?
Classwork today

• Create 2 worlds (or can combine them)
  – Problem 14, page 148
  – Problem 15, page 149