Passing Parameters to Event-Handler Methods

Alice
Overview

- Interactive programming with mouse-click events
- Several mouse-click targets, one event-handler method
  - Passing a parameter to identify the target
- “move to” instruction
- Testing
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.
Mouse-click events

We want to create an interactive animation that simulates an ancient Greek tragedy.

The user will choose the philosopher that will be the next victim of Zeus’s anger. When the user mouse-clicks on one of the philosophers, Zeus will point at that philosopher and a thunderbolt will strike the object that was clicked.
Problem

In the initial scene, four philosophers (Euripides, Plato, Socrates, and Homer) are possible targets for a mouse-click.

The problem is: how do we write our program to aim the thunderbolt at the selected philosopher?
One possible solution to this problem is to write four event-handlers, one for each target object.

Event: Click on **Euripides**
Responding Method:
- Zeus turns to point at Euripides
- Shoot thunderbolt at Euripides
- Euripides meets a tragic ending

Event: Click on **Socrates**
Responding Method:
- Zeus turns to point at Socrates
- Shoot thunderbolt at Socrates
- Socrates meets a tragic ending

Event: Click on **Plato**
Responding Method:
- Zeus turns to point at Plato
- Shoot thunderbolt at Plato
- Plato meets a tragic ending

Event: Click on **Homer**
Responding Method:
- Zeus turns to point at Homer
- Shoot thunderbolt at Homer
- Homer meets a tragic ending
A Better Solution

A better solution is to write one event-handler method and pass in a parameter that identifies the target object.

Event: Click on **target object**
Parameter: **target object**
Responding Method:
- Zeus turns to point at **target object**
- Shoot thunderbolt at **target object**
  - **target object** meets a tragic ending
Demo: Zeus world

// make Zeus point at object that was mouse clicked and lightening appear

// shoot lightening bolt at object clicked

// do together

Lightning -> set opacity to 1 (100%) -> duration = 1 second -> more...

// do in order

// Zeus point at who onlyAffectYaw = true duration = 1 second more...

// shoot lightening bolt at object clicked

// do together

Lightning -> move to who's position duration = 1 second more...

World -> play sound World.thunder (0:03.345) duration = 1 second more...

smoke -> move to who's position duration = 1 second more...
// hide lightning and make smoke appear to turn clicked object to a crisp

**Do together**

- **Lightning**: set opacity to 0 (0%) \( \text{duration} = 0.5 \text{ seconds} \) more...
- **smoke**: set opacity to 1 (100%) \( \text{duration} = 0.5 \text{ seconds} \) more...
- **smoke.cycleSmoke**
- **who**: set color to \( \text{duration} = 0.5 \text{ seconds} \) more...

**Do in order**

- **who**: move up .05 meters \( \text{duration} = 0.25 \text{ seconds} \) more...
- **who**: move down 0.05 meters \( \text{duration} = 0.25 \text{ seconds} \) more...

// move lightning back to cloud

- **Lightning**: move to cloud's position \( \text{duration} = 0.5 \text{ seconds} \) more...
When linking the mouse-click event to the shootBolt event-handler method, select “object under mouse cursor” as the parameter.
move to

Several statements in the shootBolt method use a move to instruction.

The move to instruction moves an object to a particular position in the world. In the example above, the lightening bolt is moved to the position of the cloud. (The move to instruction is described in detail in Tips & Techniques 3.)
move to with an object parameter

If an object parameter is used to specify a location, the process of creating the move to statement involves three steps:

1) Use default parameter

```
Lightning \move to \Vector3(0, 0, 0) \more...
```

2) Drag in an arbitrary object as the parameter

```
Lightning \move to Lightning \'s position \more... \more...
```

3) Substitute the object parameter name

```
Lightning \move to \who \'s position \more... \more...
```
Testing

When parameters are used in interactive programming, it is especially important to test run the animation several times, each time creating different events to be sure all possible parameter values work as expected.

It is also important to try things that shouldn’t work.
Demo

Test run of Zeus world.
What happens if you click on each philosopher, one at a time?
What happens if you click on a column in the scene, instead of a philosopher?
Assignment

Read Chapter 5 Section 2
- Mouse-clicks
- Passing a parameter to an event-handler
- move to
- Testing

Read Tips & Techniques 5