Stateful Objects and Stable Identities

CS 5010 Program Design Paradigms
"Bootcamp"
Lesson 11.2
Key Points for Lesson 11.2

• Sometimes objects need to ask questions of each other over time.
• To accomplish this, the object being queried needs to have a stable identity that the querier can rely on.
• In this lesson, we'll show what can happen when this fails.
Sometimes making a new object doesn't do what's needed

• We now begin a sequence of programs illustrating patterns of object communication.
• These programs will involve a ball bouncing in a box.
• What’s interesting, though, is that the box has an adjustable wall, so the ball and the box need to communicate about the position of the wall.
adjustable-box.rkt

• Let’s begin with just the box.

• We will have many objects in our world, so we make our world a container that just passes the messages from big-bang down to the objects it contains.
Translating from the videos to the example files

• We’re about to view a set of videos.
• These videos were recorded a while ago, so our terminology has changed a little bit.
• Here are the most significant differences:

<table>
<thead>
<tr>
<th>Name in Videos</th>
<th>Name in Example Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container%</td>
<td>World%</td>
</tr>
<tr>
<td>Container-and-box.rkt</td>
<td>Adjustable-box.rkt</td>
</tr>
<tr>
<td>Bouncing-ball-try1.rkt</td>
<td>Bouncing-ball-functional.rkt</td>
</tr>
</tbody>
</table>

• We’ve also updated the tests to our current framework.
Code walkthrough: adjustable-box.rkt

• Watch the video at
  http://www.youtube.com/watch?v=ihE-TY17_UA (14:40)
Video demonstration: bouncing-ball-functional.rkt

- [link](http://www.youtube.com/watch?v=q1pkdjd9Djo) (14:21)
What happened here?

• Let’s draw some pictures
• We'll use a simple world with one box, called box, and one ball, called ball
• We'll call the world that big-bang sees world.
What is the invariant?

• On the next slide, we see pictures of the world before and after a tick.
• The world obeys the invariant that the world’s ball always points to the correct box: namely, the world’s box.
• This stays true even though after a tick, we have a new world.
World-after-tick in bouncing-ball-functional.rkt

(send world after-tick) = (new World%
[box (send box after-tick)]
[ball (send ball after-tick)])

box after-tick returns itself

IN Variant: the world’s ball always points to the world’s box
What happens after a drag

• After a drag, however, the world has a new box with the new dimensions.
• But it still has the old ball, pointing to the old box.
• The invariant has been violated.
• The ball still bounces at 300, where the right edge used to be.
• Here's a picture:
World after drag in bouncing-ball-functional.rkt

(send world after-mouse ...)
= (new World%
[box (send box after-mouse ...)]
[ball (send ball after-mouse ...)])

INVARIANT: the world’s ball always points to the world’s box
What we want

• What we want is for the box to have a stable identity.

• After a drag, the box’s fields will change, but it is still the same box, so the ball will still see it.

• The invariant will be preserved.

• Again, here's a picture:
What we want:

(send world after-mouse ...) = (new World%
  [box (begin (send box after-mouse ...) box)]
  [ball (send ball after-mouse ...)])

Container always has same box

ball after-mouse returns itself

INVARIANT: the world’s ball always points to the world’s box
We need to give the box STATE!

- We need to give the box a stable identity, so balls will know who to ask.
- But the information in the box must change!
- Solution: we need to make the box MUTABLE.
- In other words, it should have state.