Real State vs. Simulated State

CS 5010 Program Design Paradigms
"Bootcamp"
Lesson 10.3
Real State vs. Simulated State

• Before, our objects were immutable. If we needed to construct an object with a different state, we just constructed a new one.
• But we couldn't do that and have objects with stable identities.
• In this lesson, we'll consider the difference between "real state" and "simulated state".
Let's think about traffic lights

• nextstate : TLState $\rightarrow$ TLState
  – If the traffic light were in state $s$, what state should it be in next?

• This is a *specification* of how a traffic light should behave.

• This is simulated state: a mathematical function

• Real state: *make the traffic light change to the next state!*
Similarly for on-tick

• **on-tick**: WorldState $\rightarrow$ WorldState
  – If the world is in state $s$, what state should it be in after the next tick?

• This is a specification of the desired behavior of the world.

• It's just a mathematical function.

• **big-bang** takes these functions and constructs a world that really behaves that way.
Real state is about sharing

• How can you tell the difference between a traffic light and a TLState?
• Ans: everybody sees the same traffic light.
• The traffic light has a stable identity.
• If its state changes everybody sees it.
Are they seeing the traffic light or a model of the traffic light?
Are they seeing the traffic light or a model of the traffic light?
Are they seeing the traffic light or a model of the traffic light?

Now we know!
What just happened?

• The people who were looking at the real traffic light saw it change to green.
• The people who were looking at the simulated traffic light (in their newspapers) didn't see the change.
States vs. Values

• TLStates are just values. They never change. "red" is always "red".

• The traffic light was an object with stable identity, which had real state: the light could change from one TLState to another.

• If two observers are looking at the same object, and the object's state changes, then both observers see the change.
Blackboard metaphor

• State is like a blackboard: when I write on the blackboard, everybody sees it, because you are all looking at the same blackboard.
  – The blackboard has a stable identity!
• With state, I don't care who's in the room.
• I could distribute the changes by sending messages to each of you, but then I'd have to know who I needed to talk to.
  – That's what we do when we pass parameters in our programs!
• Blackboards foster collaboration
Where is the state?

• In World programs, there was just one piece of state—the world—and it was kept inside **big-bang**.

• In OO system, each object can have its own state.

• This complicates things
An object is a little blackboard

- Lots of little blackboards!
- Of course you can't just look at a blackboard, you have to ask it questions, like (send traffic-light1 are-you-red?)
- This means we have to worry about patterns of collaboration again.
What has to be shared in our example?

- When the wall changes, all the balls have to see it.
- The balls all have to look at the same wall.
- So let's make the wall stateful.
- In the next lesson, we'll see how to design code that accomplishes this.
  - Remember, "design" means getting what you know out of your head and onto the page
Next Steps

• If you have questions about this lesson, ask them on the Discussion Board
• Go on to the next lesson