Decisions and Logical Questions

Alice
Overview

Thinking about more advanced worlds
- decisions
- logical expressions
- conditional execution

An Interactive Example
Thinking About More Advanced worlds

No doubt you have started to think about building animations like simulations and video games…

To build more advanced worlds, we need to make decisions
Example

In a car-race simulation, the driver steers the car around curves and past mile-markers.

- If the car stays on the road, the score increases.
- If the car goes off the road into the stands, the car crashes.
- If the driver gets the car over the finish-line, the time is posted and the driver wins!
Logical Expressions

A decision is made based on current conditions

A condition is checked in a logical expression

- car on road
- car over finish line
Boolean Value

Logical expressions evaluate to "true" or "false"

- car on road \( \rightarrow \) true
- car over finish line \( \rightarrow \) false

"True" and "False" are called **Boolean** values
So, we know:
A decision instruction checks a condition
...a condition is written as a logical expression
...a logical expression evaluates to true/false

The true/false result is used to decide whether or not a method will be invoked
- if true, do something
- if false, don't do it (or do something else)

This is Conditional Execution
The Zeus world revisited

Recall the Zeus world

Testing this world, we found two significant problems

Anything the user clicked got shot by Zeus’ bolt – even Zeus himself!

Philosophers could get “refried!” In other words, clicking on a philosopher who had already been hit with lightening led to that philosopher being hit with lightening again.
Problem

What we need in the Zeus world is conditional execution

Check conditions:
- Is the object clicked a philosopher?
- If so, has the philosopher already been hit by lightening?

Conditional execution:
- Lightening bolt will be shot or not
The If/Else statement

In Alice (and many, many other programming languages), an If/Else decision instruction is used for conditional execution.

The If/Else statement

```plaintext
If <something is true>
  <do this>
Else
  <do that>
```

Note: the "else" part (or even the "if" part) of the instruction can be

```plaintext
<do nothing>
```
Using the if statement

We’ll tackle the problem of Zeus only shooting thunderbolts at philosophers first.

Start with a blank if statement…

drag in the who tile, then select == and Homer

== is a relational operator meaning “is equal to”
The **if statement** encloses the **Do in order** block. (We say the Do in order is **nested** inside the if.)

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

We will use logical operators!

Boolean logic operators allow us to check more than one condition.

Using logical operators, we build the statement to test whether the object clicked was one of the four philosophers or not.
Demo

A demonstration of the Zeus world with if statement and boolean logic operators.
Completing the Zeus world

The second problem we want to solve is how to prevent lightening striking the same philosopher more than once.

We only need to check for a possible duplicate-strike if we already know a philosopher object is clicked.

The way to handle this is to nest a second if/else statement inside the first if/else statement.
Using a property in a condition

How do we know if a philosopher has already been struck by lightening?

- When a lightening bolt strikes, the philosopher “is fried” and the object color property is set to black.

- So, we can use the object color property to check for a previous lightening strike.
Nested if statements

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

A demonstration of completed Zeus world
Assignment

Read Chapter 6 Section 1

- If/Else statements
- Logical operators
- Relational operators