Creating A New Instruction
(Programmer-defined Method)

Alice
Overview

- Creating our own (programmer-defined) methods
- Demo
- Why we want to write our own methods
  - Abstraction
- Re-using code
- Comments
Previously…

This is the code we wrote to animate the snowman trying to catch the snowwoman’s attention

He only tries to catch her attention once!
A Problem

To make the problem more realistic, we would probably want the snowman to try to get the snowwoman’s attention several times. How can we do this?

We would need to:

- drag-and-drop each instruction multiple times into the edit box, or
- use the clipboard to copy-and-paste
A Quicker Solution

A quicker and easier solution to this problem is to:

- define our own instruction (method)
- name the new method `catchAttention`

Then, we can drag-and-drop the `catchAttention` instruction into the edit box just like the built-in instructions (move, turn, roll, ...)

A demonstration of creating the catchAttention method in the world
World.catchAttention  No parameters

No variables

- Do in order
  - Snowman.Head \point\ Camera \duration\ 0.5\ seconds\ more...

- Do together
  - Snowman \say\ Hey there!!! \more...

- Do in order
  - Snowman.Head.Eyes \move\ up\ 0.04\ meters\ duration\ 0.5\ seconds\ more...
  - Snowman.Head.Eyes \move\ down\ 0.04\ meters\ duration\ 0.5\ seconds\ more...

- Snowman.Head \point\ at\ Snowwoman.Head \more...
Using the catchAttention method

The `catchAttention` method is executed by invoking (calling) the method from `my first method`
Why? Why do we want to write our own (programmer-defined) methods?

- saves time -- we can call the method again and again without reconstructing code
- allows us to "think at a higher level"
  - can think “catchAttention" instead of “point head at camera, then say ‘hey there’ while moving eyes up and down"
- the technical term for "think at a higher level" is "abstraction"
Comments

While Alice instructions are easy to understand, it is often desirable to be able to explain (in English) what is going on in a program.

We use comments to explain to the human reader what a particular section of code does.
The snow people world with comments

Notes:
1) Comments appear in green
2) Alice ignores comments.
3) Comments make the program easier to read.
Assignment

Read Chapter 3 section 1

- Methods
  - How to create them
  - How to invoke them
  - When is it appropriate to use them

- Comments
Passing a Parameter to a Method

Alice
Overview

- The need for more flexible methods
- Passing a parameter to a method
- Demos
  - Using the Alice interface to write code for a parameter object
Our task is to create an animation for a bug band as an advertisement for their next concert. In the animation, each band member will be highlighted with a spotlight as they perform a solo.
The storyboard

*Playing a solo*
- The spotlight points at a band member
- The insect moves about, “playing” its instrument
- The spotlight points back at the crowd
This code will only work for the bee.
Since each band member performs a solo, we will need four versions of the code.
A Better Solution

Four versions of the same code seems a bit tedious. The only things that change are the insect and the music that plays.

A better solution is to write a more flexible method.

We want to have only one method and to specify which band member is to perform the solo, and which music should be played.
Parameters

Built-in Alice instructions allow you to specify parameter values such as direction and duration.

In writing your own instruction methods, you can also specify parameters.
Kinds of Parameters

Alice supports several kinds of parameters.
Demo

Demonstration of creating a parameter in the solo method to specify which band member is to play a solo and which music should be played during that solo.
solo with parameters

World.my first method

World.solo

Obj bandMember, music

No variables

Do in order

StageSpotLight point at bandMember duration = 0.25 seconds more...

Do together

World play sound music duration = 1 second more...

Do in order

bandMember move up .25 meters duration = 0.5 seconds more...

bandMember move down 0.25 meters duration = 0.5 seconds more...

StageSpotLight point at concert_stage.crowd more...
Invoking solo method

Note that both parameters must be specified for the method invocation to work correctly.
Another example

Write an animation for a magic act, where a magician points a magic wand at an object, and that object levitates!!!

What parameters will be needed?

1) Which object should be levitated (an object parameter)

2) How high it should levitate (a number parameter)
A storyboard for the levitate method

Parameters: floatingObject, height

Do in order

- Magician’s wand point at floatingObject
- floatingObject move upward height meters
- floatingObject move downward height meters
Demo: levitate

World.levitate

No variables

Do in order
- Magician’sWand point at floatingObject
- floatingObject move up height meters
- floatingObject move down height meters
Invoking the method

Note that both parameters must be specified for the method invocation to work correctly.
Assignment

Read Chapter 3 section 2

Parameters
- Object parameters
- Other types of parameters

Calling methods and passing information

Read Animation Tips 3