Note: thanks to Wanda Dann and Steve Cooper for slide ideas
Announcements

- Read Chapter 5.1 for next time
- Assignment 5 out today
- Prof Rodger – extra office hours friday, no office hours next week!
What we will do today

• Lecture on Chap 4 Sec 3 (continued)
• Classwork
Guidelines

• To avoid misuse of class level methods
  – Avoid references to other objects
  – Avoid calls to world-level methods
  – Play a sound only if the sound has been imported and saved out as part of the new class

• If these guidelines are not followed and an instance of the new class is added to another world
  – Alice will open an Error dialog box to tell you something is wrong
Bad Example 1
Bad Example 2

```plaintext

*cleverSkater.skateAroundBad*  No parameters

No variables

- **Do in order**
  - **Do together**
    - `cleverSkater` turn to face *penguin* more...
    - `cleverSkater.rightLeg` turn forward 0.1 revolutions duration = 0.5 second
    - `cleverSkater` move forward 2 meters more...
    - `cleverSkater` turn right 1 revolution `asSeenBy` *penguin* more...
  - **Do together**
    - `cleverSkater` turn to face *camera* more...
    - `cleverSkater.rightLeg` turn backward 0.1 revolutions more...
```
Problem

• What if you were convinced you needed to write a class-level method where another object is involved?
• For example, a method for ice skater to skate around another object – here a penguin
Solution

• Class-level method with object parameter

cleverSkater.skateAround

Parameter: whichObject

Do in order
  Do together
    cleverSkater turn to face whichObject
    cleverSkater lift right leg
  cleverSkater move to whichObject
  cleverSkater turn around whichObject
Translating Design into Code

• Most of skateAround storyboard easy to code

• Last two steps, require more thought
  – cleverSkater move to whichObject
    • What distance should cleverSkater move?
  – cleverSkater turn around whichObject
    • How do we tell cleverSkater to turn (in a circle) around another object?
Built-in Functions (or questions)

- The built-in function *distance to*
  - used to determine the distance the skater must move
Calling the function

- Code to move skater to \textit{whichObject}

Oops, skater will collide with penguin!

Distance between two objects is measured center-to-center
Expressions

• To avoid collision
  – Use math operator to create an expression that adjusts the distance

• Math operators in Alice
  addition +   subtraction -  
multiplication *   division /  

• Example:
How to put in an Expression

Result:
Result

Stops **before** penguin  
Skates **around** penguin
asSeenBy

- For skater to skate around another object
  - Pass \textit{whichObject} as an argument to \textit{asSeenBy} parameter in turn instruction
More on AsSeenBy

- Use invisible object (isShowing set to false) to have objects fly around in a circle
Testing

• Each time you create a new class, test it!
  – Add an instance of new class to new world
  – Write a short test program
    • Test each new method

• Testing increases your confidence in the ability to reuse your code in other worlds
Classwork today

• Modify your two classes from last time
  – Add methods with parameters
  – Use asSeenBy
  – Use isShowing
  – Use math
• Create a new world with 3 objects and demo
• See handout for more requirements