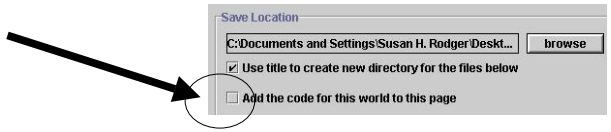
CompSci 4
Chap 4 Sec 3
Sep 22, 2005

Prof. Susan Rodger

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

Announcements

- Assignment 3 is due midnight today
  - Adding code to web page is optional



- Assignment 4 out today
- Read Chapter 6.1 for next class
- Reading Quiz

Animated Actions

- Some actions are more naturally associated with a specific class of objects rather than the overall world
- Examples
  - A person walking
  - A wheel rolling
  - A fish swimming

Class-level Methods

- Write a method to add abilities/functions to a specific class of objects
  - Class-level method
  - NOT world-level method
- Now show how to build class-level method

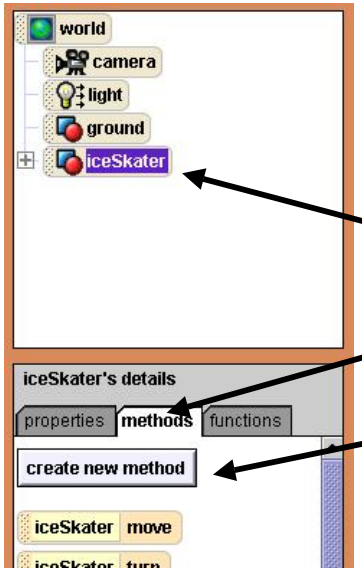
An Example

- How can we create a skate method for ice skater objects?

We need to:

1) Associate the new method with an ice skater
2) Write the new method to animate the ice skater

The solution



To associate the animation with the ice skater

- Select iceSkater tile in Object Tree
- Select methods tab in details area
- Click on "create new method" button

Storyboard for *skate*

Skate:
Do Together
  move skater forward 2 meters
  Do in order
    slide on left leg
    slide on right leg

- The **slide** actions
  - Require several motion instructions
  - We'll break these two actions into smaller pieces
  - Technique is **stepwise refinement**

Refined storyboard for skate

Skate:
Do Together
  move forward 2 meters
  Do in order
    slideLeft
    slideRight

slideLeft:
Do in order
  lift right leg and turn upper body forward
  lower right leg and return body upright

slideRight:
Do in order
  lift left leg and turn upper body forward
  lower left leg and return body upright

Writing the code

- Next step – translate design into code
- For *slideLeft*, possible translation is:

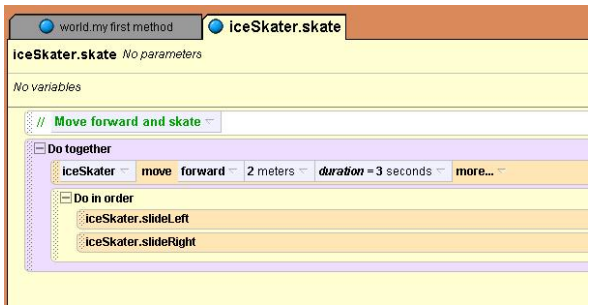
Design Step	**Instruction**
Lift the right leg	Turn the right leg forward
Turn upper body forward	Turn upper body forward
Lower the right leg	Turn the right leg backward
Return the body upright	Turn the upper body backward

SlideLeft and Demo

Where is wait?

Correspondence of design to code

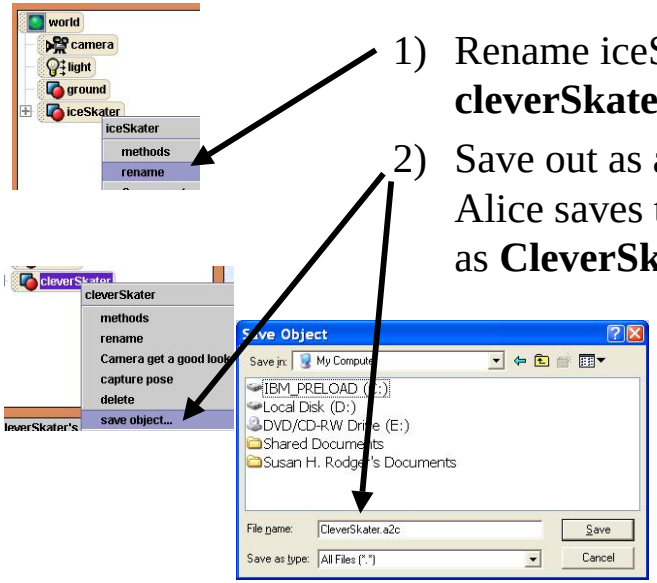
Skate:
Do Together
  move skater forward 2 meters
  Do in order
    slide on left leg
    slide on right leg



Question

- Writing methods to make ice skater perform a skating motion – intricate task
- Would like to reuse these new methods in another world
- How can you make *skate* method available for an ice skater in a different world?

Answer: Save out as a new class



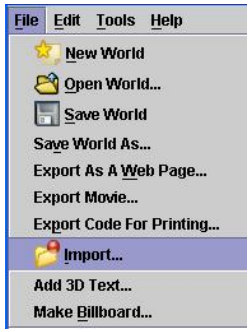
1) Rename iceSkater as **cleverSkater**
2) Save out as a new class. Alice saves the new class as **CleverSkater.a2c**

Inheritance

- The CleverSkater class
  - **inherits** all the properties and methods from the original IceSkater class
  - has newly defined methods (*skate*, *slideLeft*, *slideRight*)
- In other programming languages, the concept of creating a new class based on a previously defined class is called **inheritance**

Using CleverSkater

- An instance of the CleverSkater class can be added to a new world



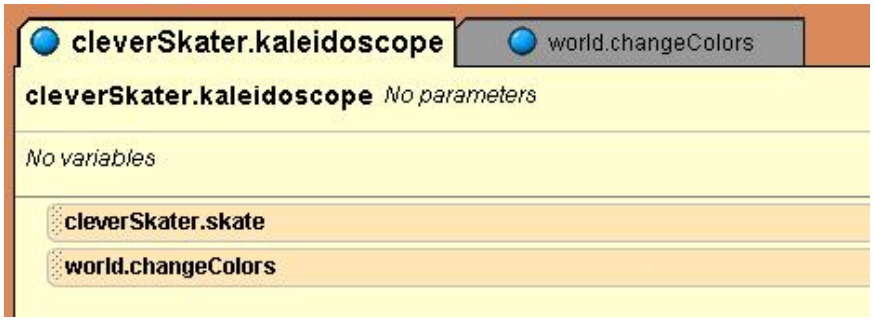
Benefits of Inheritence

- Inheritance supports
  - Reuse of program code
    - Programmer writes code once
    - Use code later in different programs
  - Sharing code with others on a team project

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

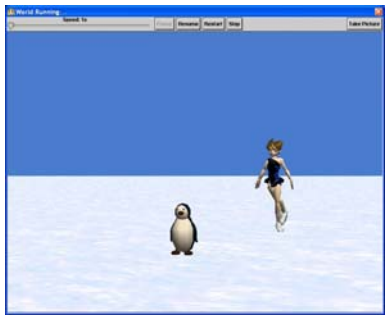
Bad Example 1



Bad Example 2

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*

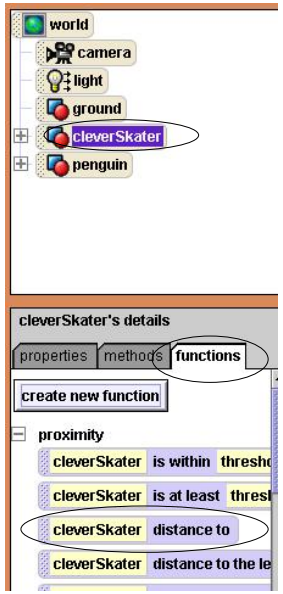
Do in order
  Do together
    cleverSkater turn to face *whichObject*
    cleverSkater lift right leg
  cleverSkater move to *whichObject*
  cleverSkater turn around *whichObject*

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

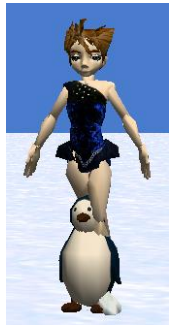


Calling the function

- Code to move skater to *whichObject*

Oops, skater will collide with penguin!

Distance between two objects is measured center-to-center



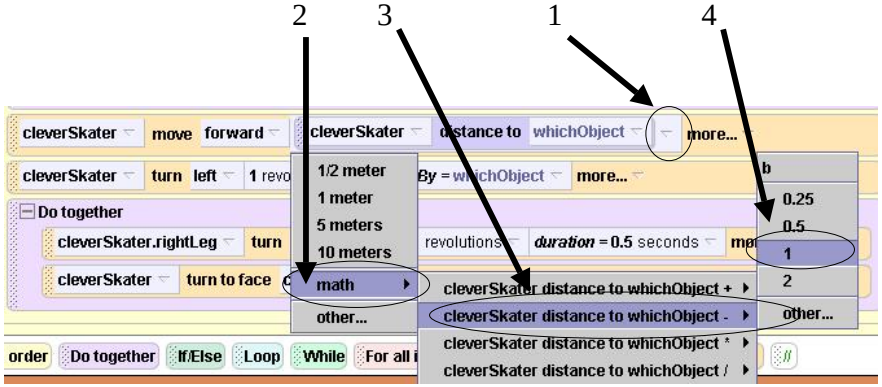
Expressions

- 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

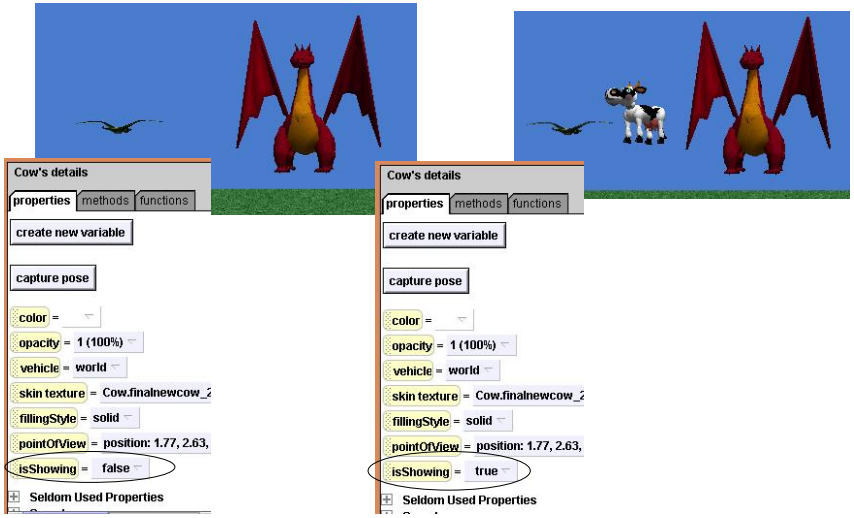
Skates **around** penguin

asSeenBy

- For skater to skate around another object
  - Pass *whichObject* as an argument to *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
create a new class example

crouching

pouncing

walking

turn and smile

Classwork today

- Create a new class
  - Inherit from another class that has 4 limbs
  - Create at least four new methods
  - One of the new methods should invoke one of the other new methods
- Create a second new class inherited from another object
  - with at least 4 methods
  - At least two of the methods must have a parameter
  - Use AsSeenBy, isShowing and math each in some method (not necessarily the same method)
- Save out new classes and read into another world
- See handout for more details