CS 120 Lecture 03

Methods (part 2)
Alice In Action, Ch 2

4 September 2012

Slides Credit: Joel Adams, Alice in Action

Objectives

- Build class-level methods to elicit desirable behaviors from objects
- Reuse a class-level method in multiple worlds
- Understand how an object's *position*, *orientation*, and *point of view* are described, changed and determined
- Documenting your code with comments.
- Understand Flow of Control with methods.

2

Methods

- Methods
  - behavior-producing messages (from the sender's view)
  - behaviors/actions in response to requests, messages (from the recipient's view)
  - E.g. in world.my_first_method: whiteRabbit.pointat(camera)
- Convention for naming methods
  - Name should be a verb or verb phrase
  - Name should describe what the method does
- A method is a way to name a block of code.

Methods

- Objects have predefined methods for common tasks
- Methods may also be created by Alice developers
  - Two main reasons for building your own methods
    - To provide an object with additional behaviors (Today)
    - To organize your story and program into more manageable pieces (last Tuesday)
- Divide and conquer methodology
  - Break a big problem into smaller problems
  - Solve each of the smaller problems
  - Combine the solutions of smaller problems into a solution for the original, big problem
- Hiding complex details with abstraction.

World Methods for Scenes and Shots

- User stories can be divided into scenes and shots
  - Scene: segment of a story, usually set in one location
  - Shot: part of a scene, normally from one fixed camera view
- Use multiple scenes and shots to create a program that reflects the user story and has a modular design
-

Two shots of one scene

World and Object Methods

- World method: affects behavior of all objects in a world
- Object method: defines behavior for a single object (that may have multiple parts)
  - examples: **`flapWings()`** for dragon, hop() for a rabbit…

Program Documentation

- Standalone readme, manual…
- Comments: explanatory remark ignored by Alice
  - an integral part of code
  - Used to describe what code does at various levels
    - the overall program, individual methods, blocks of statements….
  - Useful for collaborators and developers themselves
  - Important part of programming
    - Also a component evaluated for your program grades

Object Methods for Object Behaviors

- Example 1: Hiding complex details
  - Add a **`EvilStepsister1`** object to the world
  - Select **`evilStepsister1`** from object tree and click methods tab
  - Click **`create new method`** and enter **`melt`**
    - Make her melt (she's a witch!!): Send messages to **`evilStepsister1`**
      - **`Set opacity to 0`**
      - **`Resize her`**
      - **`Say something`**
  - Invoke **`melt()`** from **`my_first_method()`**
  - Add comments to the **`melt()`** method

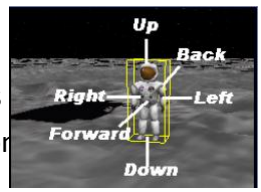
Thinking in 3D

- Learn about 3D movement to work in Alice
- Object's position
  - determines object's location in the 3D world
  - is changed an object's translational motion
- Object's orientation
  - determines the way an object is facing
  - is changed by an object's rotational motion

9

3 Dimensions, 6 Degrees of Freedom

- A 3D object has
  - 3 dimensions in the directions of
    - height, width, depth
  - 6 degrees of freedom
    - Pose: 3 positions, 3 orientations
    - Motion: 3 translations, 3 rotation



An Object's Position

- Three axes are used to define the world space
  - LEFT-RIGHT (LR): world's width dimension
  - UP-DOWN (UD): world's height dimension
  - FORWARD-BACKWARD (FB): world's depth
- Three values specify object center's position in the **`world`**: lr, up, fb

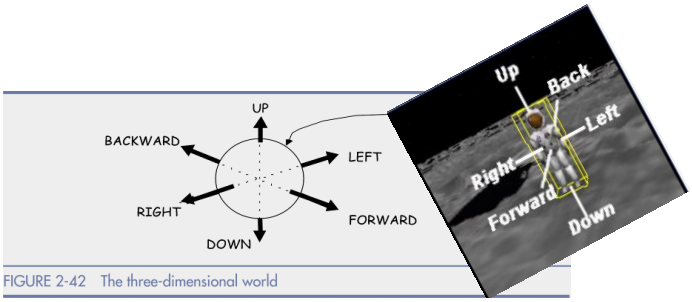


FIGURE 2-42 The three-dimensional world

11

Center of an object

– At the center of mass

– Where it stands on the ground

– Where it is held

An Object's Position

- The position of the center of an object, *i.e.* its lr, up and fb values, are with respect to the world's axes
- Change an object's position using **`move()`**
  - Directional values (up, down…) are with respect to object's axes

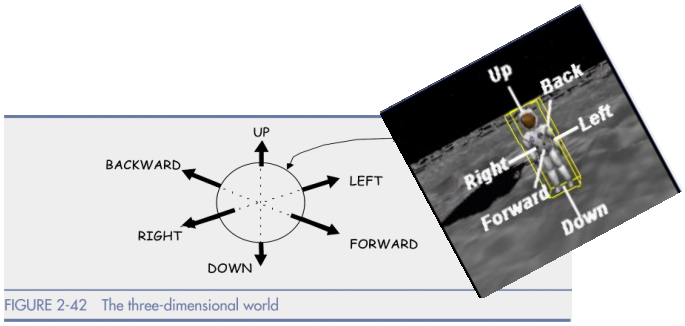


FIGURE 2-42 The three-dimensional world

An Object's Orientation

- The orientation of an object, *i.e.* its yaw, pitch and roll values, are with respect to the world's axes
- Change an object's position using **`turn(),roll()`**
  - Directional values (left, right,…) are with respect to object's axes

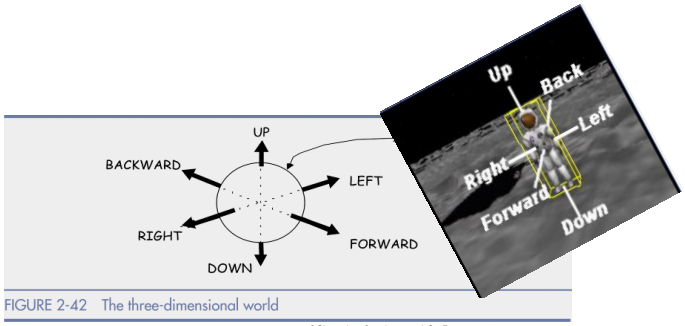
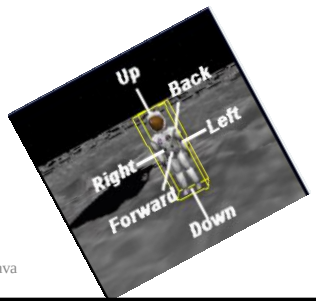


FIGURE 2-42 The three-dimensional world

An Object's Rotational Motion

- Again: rotational motion are specified w.r.t. object's own axes
- Yaw: amount of object's rotation about the UD-axis
  - Example: shaking head for no, **`turn(RIGHT,…)`**
- Pitch: amount of object's rotation about the LR-axis
  - Example: nodding head for yes, **`turn(FORWARD,…)`**
- Roll: amount of object's rotation about the FB-axis
  - Example: **`roll(LEFT, 0.25)`**



An Object's Orientation (continued)

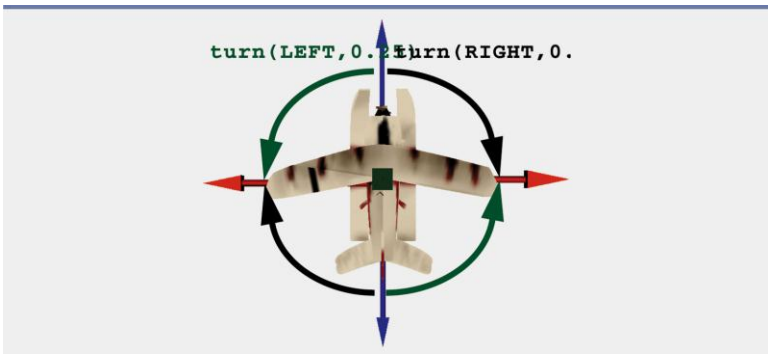


FIGURE 2-45 Changing yaw: turning left or right

An Object's Orientation (continued)

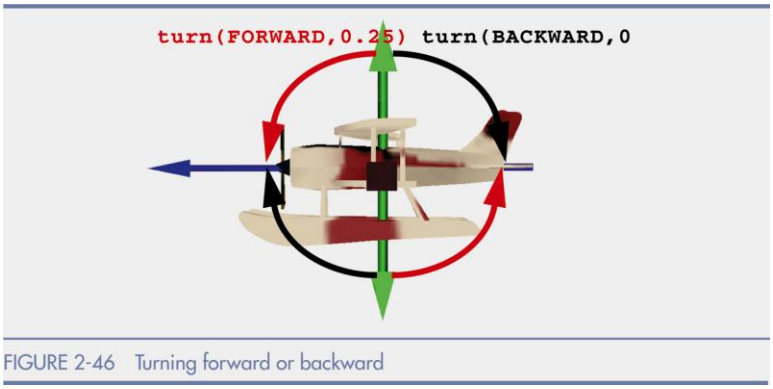


FIGURE 2-46 Turning forward or backward

e.g. from the current orientation,
turn(forward, 0.25): nose down
turn(backword,0.25): nose up

17

An Object's Orientation (continued)

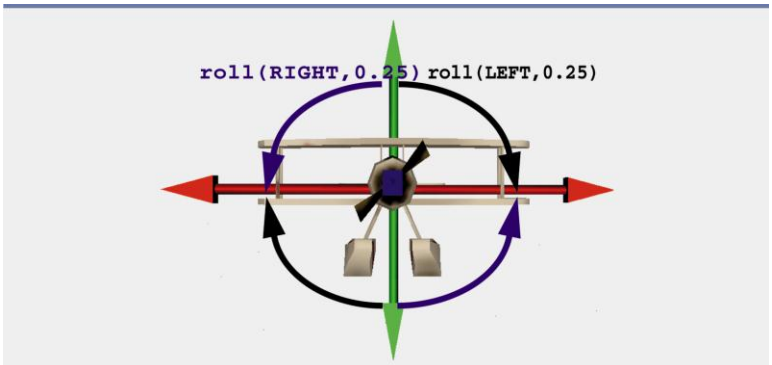


FIGURE 2-47 Rolling left or right

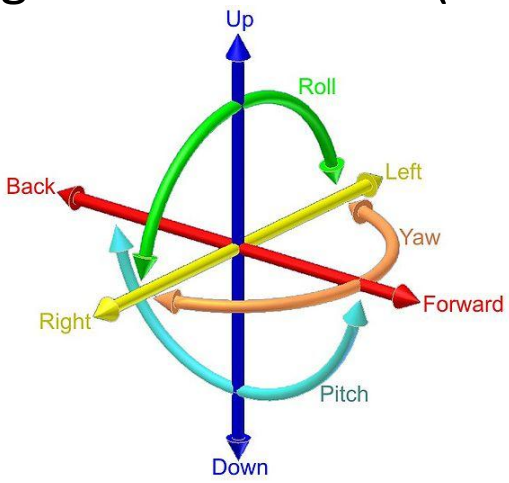
18

Point of View

- Combines object's position and orientation
- Six values in point of view: [(*lr*, *ud*, *fb*),(yaw, pitch, roll)]
  - Six values correspond to six degrees of freedom
- Methods used to change six values
  - **`move()`**, **`turn()`**, and **`roll()`**
- Method used to change point of view
  - **`setPointOfView()`**

19

Six Degrees of Freedom (6 DOF)



http://en.wikipedia.org/wiki/Six_degrees_of_freedom

20

Object Methods for Object Behaviors

- Example 2: Telling a Dragon to Flap its Wings
  - Add a **`dragon`** object to the world
  - Select **`dragon`** from object tree and click methods tab
  - Click **`create new method`** and enter **`flapWings`**
  - Send **`roll()`** messages to each of the dragon's wings
  - Invoke **`flapWings()`** from **`my_first_method()`**
  - Add comments to the **`flapWings()`** method

Object Methods for Object Behaviors (continued)

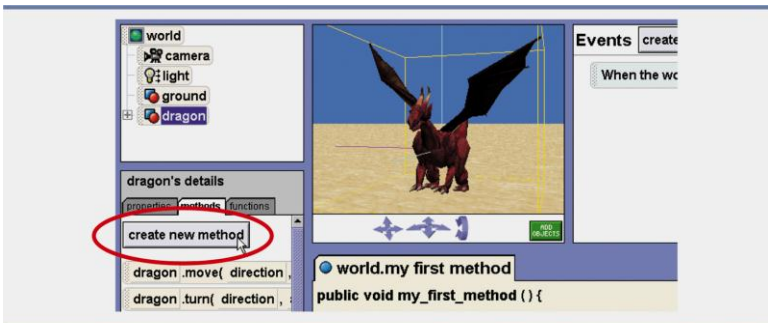


FIGURE 2-12 Creating a new class-level method

Object Methods for Object Behaviors (continued)

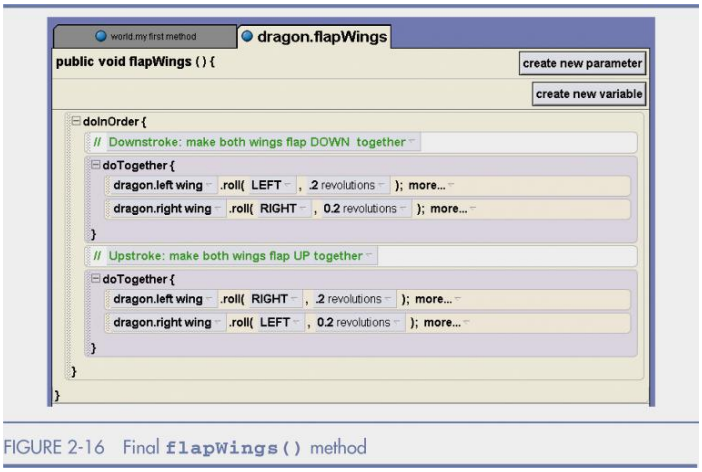


FIGURE 2-16 Final `flapWings()` method

Object Methods for Object Behaviors (continued)

- Example 3: Telling a Toy Soldier to March
  - Four actions correspond to four steps for **`march()`**
    - **`1 marchLeft;`**
    - **`2 marchRight;`**
    - **`3 marchRight;`**
    - **`4 marchLeft.`**
  - Define **`marchLeft()`** and **`marchRight()`** methods
    - These methods produce reverse behaviors
  - Incorporate new methods into **`march()`**
  - Call **`march()`** four times from **`my_first_method()`**

Object Methods for Object Behaviors (continued)

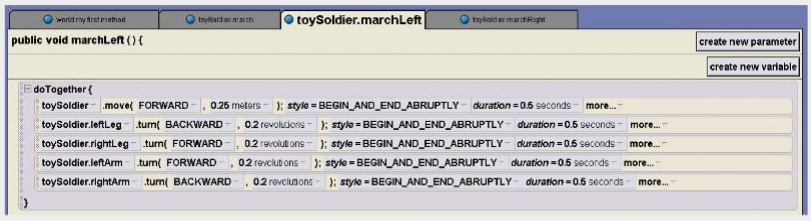


FIGURE 2-17 The `marchLeft()` method

Object Methods for Object Behaviors (continued)

FIGURE 2-19 The `march()` method

Alice Tip: Reusing Your Work

- Copy and past techniques speed up development
- How to use **`make copy`** to duplicate statements
  - Right-click bar in editing area containing method
  - Select **`make copy`**
- Example using **`make copy`**
  - Refer to **`my_first_method()`** in Toy Soldier program
  - Copy three **`march()`** statements from first **`march()`**

Alice Tip: Reusing Your Work (continued)

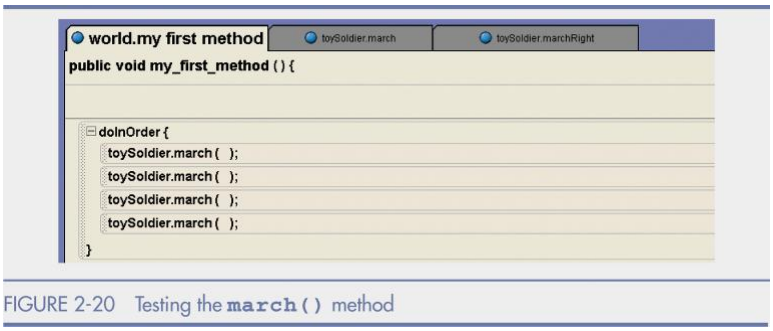


FIGURE 2-20 Testing the **`march()`** method

Using the Clipboard

- Alice clipboard
  - Used to copy and paste all statement types
  - Located in the events area
- Using Alice clipboard in Toy Soldier program
  - Drag **`doInOrder`** in **`my_first_method()`** to clipboard
  - Create **`scene1()`** method
  - Drag statement in clipboard to editing area
  - Drop statement in the **`scene1()`** method
- Only one statement may be placed in the clipboard
- You can increase the number of available clipboards

Using the Clipboard (continued)

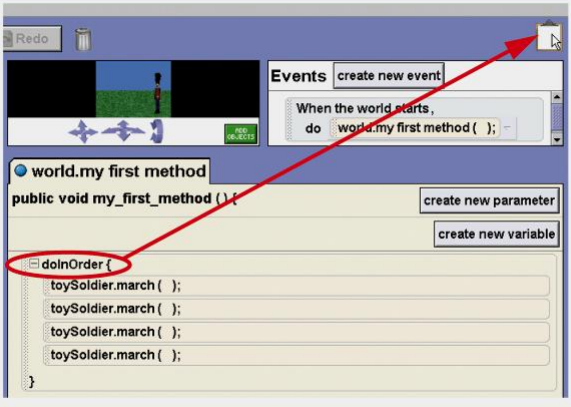


FIGURE 2-21 Dragging a statement to the clipboard

Using the Clipboard (continued)

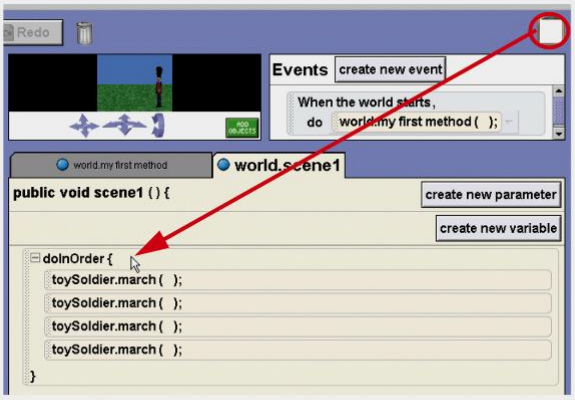


FIGURE 2-22 Dragging a statement from the clipboard

Reusing an Object in a Different World

- Alice lets you reuse objects in different worlds
- Reusing operation involves save and import tasks
- How to save the **`dragon`** object
  - Rename the **`dragon`** object **`flappingDragon`**
  - Right-click **`flappingDragon`**, select **`save object`**…
  - Navigate to appropriate storage location in the directory
  - Click the **`Save`** button
- How to import an object into a new world
  - Open new world and choose **`Import`** from **`File`** menu
  - Navigate to object location and select .a2c file

Reusing an Object in a Different World (continued)

FIGURE 2-24 Saving an object

Reusing an Object in a Different World (continued)

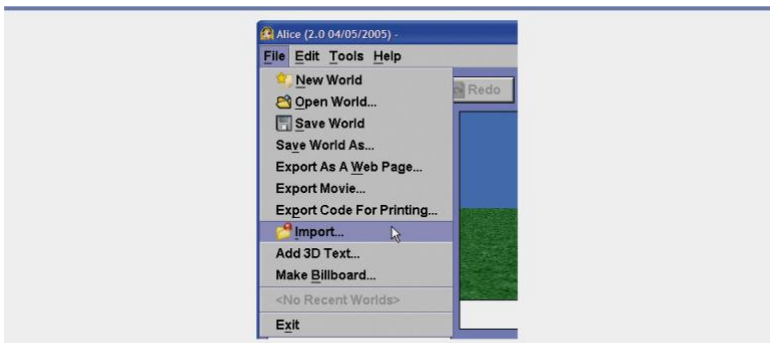


FIGURE 2-26 Importing an object

Summary

- Divide and conquer approach: decomposing a user story into scenes and shots
- Define methods to support modular design and provide advanced operations
- World methods: messages sent to the **`world`**
- Object methods: define behaviors for a single object
- Comments: remarks that explain program statements
- Flow of Control: How methods modify sequential execution.

35

Summary (continued)

- Alice clipboard: stores a copy of any statement
- An 3D object has six degrees of freedom
  - Object's position: lr, up, fb
  - Object's orientation: yaw, pitch, and roll
- Consider the **many** reasons to create a method.

36

Student To Do's

- Readings:
  - Alice in Action, Chapter 2
- Practice on your own: "Lab 1" to be posted on website.
- Homework:

  1) Recreate a movie scene or create your own short story.

  2) Create your own Scene 1 and 3 for the Wizard vs. Trolls program.
  - For both problems: Use world methods to break the stories up into shots. Move the camera around. Comment your program.
  - Demo your Alice HW in Lab on Monday (Bring it with you!)

37