Repetition: Loops in Alice

Stephen Cooper
Wanda Dann
Randy Pausch
Barb Ericson
Sept 2010

Learning Goals

- Understand at a conceptual and practical level
  - A counted for loop
  - Using a for loop in a doInOrder and in a doTogether
  - Nesting for loops
  - Using a while statement for looping
  - How to create and use a list
  - How to iterate through a list sequentially or simultaneously

Types of Loops

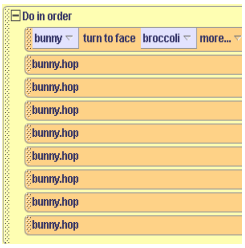
- A loop repeats a statement or block of statements
  - Rather than repeating a tile or set of tiles
- For loops can also be called counted loops
  - They repeat a known number of times
  - Like make a bunny hop 3 times
- While loops can also be called indefinite loops
  - Repeat while some condition is true
  - Like while the game hasn't been won

For Loops

- Open Chap07BunnyGarden.a2w
- We want the bunny to hop to the broccoli
  - How many hops will it take?

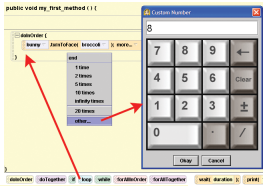
One Solution

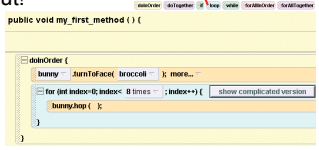
- Turn the bunny to face the broccoli
- Then drag out several of the hop tiles and test to see if the bunny is close enough
- And, if not drag out more hop tiles
- This is an example of guess and check



A better way to hop to the broccoli

- Drag up a loop tile
  - And enter 8
- Then put the hop tile inside of the loop
- Try it out!





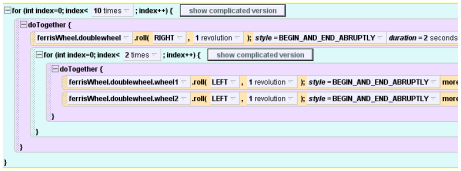
Nested Loops

- You can put one loop inside of another
  - This is called a nested loop
- Open Chap07FerrisWheel.a2w
- Add a loop that rolls the doubleWheel right 10 times

- Use BEGIN_AND_END_ABRUPTLY to make the animation look smoother
  - The default is to begin and end gently which slows down the animation at the beginning and end

Adding the inner loop

- Roll both inner wheels to the left 2 times
  - While the outer wheel is rolling right 1 time
  - Set the durations so that they take the same amount of time

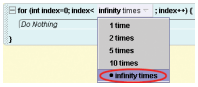


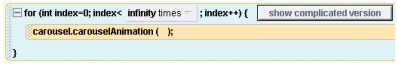
Using a function in a for loop

- Sometimes you know the number of times you want the loop to repeat
- Sometimes you don't
  - You can use a function to get the number of times

Infinite for loops

- In Alice you sometimes want the animation to continue as long as Alice is running
  - Like when animating a carousel
    - Chap07Carousel_Infinity.a2w
  - Select infinity for the number of times to run the loop

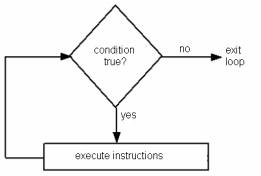




Indefinite repetition

- In some situations, we don't know exactly how many times a block of instructions should be repeated.
- All we know is that repetition is needed
  - For example, in a board game like chess or checkers, we don't know exactly how many moves it will take for a player to win or lose the game – all we know is that several moves will be needed.
- We can use a while loop in this case

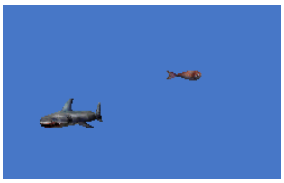
How the While statement works



- The general idea is:
  While some condition is true
  execute instruction(s)
- To write a While statement, we need to know the condition that determines whether the loop will be repeated.

Example

- A common feature in popular "action films" is an exciting chase scene.
- As an illustration of an animated chase scene, consider the hungry shark in this world. The shark is going to chase after and catch a fleeing fish.



Problem

- The problem is how do we get the shark to chase the goldfish?
  - The shark should not immediately catch the goldfish (otherwise, there would be no chase).
  - The goldfish (assuming self-preservation instincts) should appear to be fleeing.

Solution

- To create a chase scene,
  - At the same time, the shark will swim a short distance toward the fish and the fish will swim a short distance away from the shark.
  - The fish will flee to a random (but nearby) location.
  - As long as the goldfish is still 0.5 meters away from the shark, repeat the actions.

Storyboard

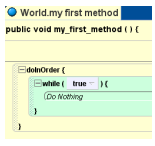
chase

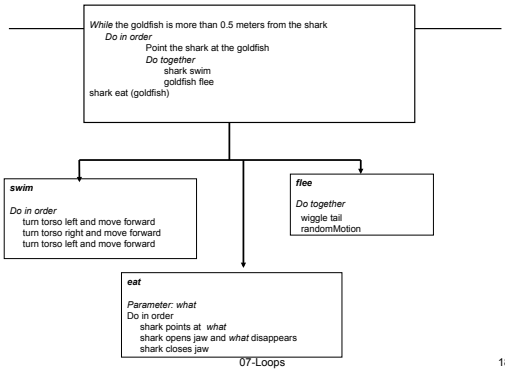
While the goldfish is more than 0.5 meters in front of the shark
  doInOrder
    shark point at the goldfish
    doTogether
      shark swim (toward the goldfish)
      goldfish flee (away from the shark)
shark eat (the goldfish)

The *shark swim*, *goldfish flee*, and *shark eat* actions are complex. Use stepwise refinement to break them down into simple steps.

Starting the chase

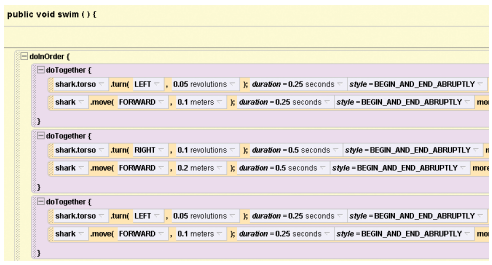
- In my_first_method
  - Drag up the while tile
    - And select true
  - Replace the true
    - With distanceInFrontOf from goldfish
      - And select shark
      - And select 0.5 for distance





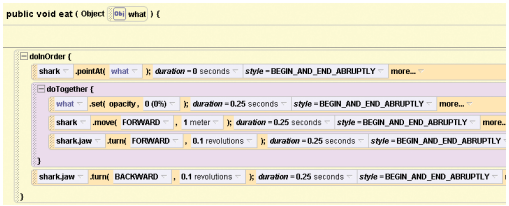
Shark swim method

- Move the shark while turning the torso



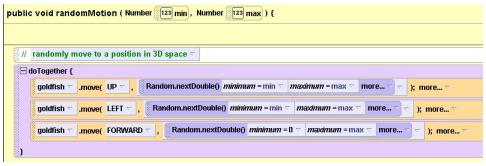
Shark eat method

- To eat the object passed as a parameter
  - Point at the object
  - Then move forward to the object as it disappears and move the jaw



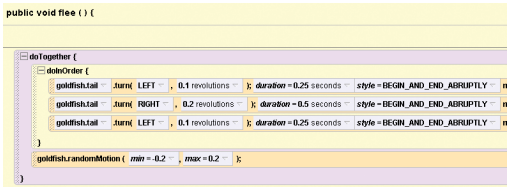
Goldfish randomMotion method

- The goldfish could move randomly in any of the six directions
  - Left, right, up, down, forward, or backward
- But, if move is given a negative number it will go the opposite direction (forward will become backward)

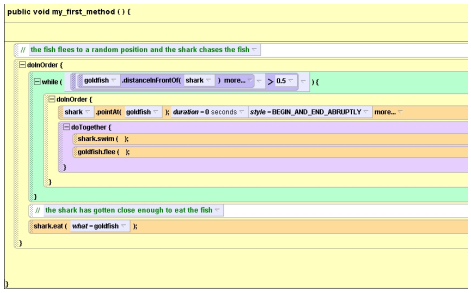


Goldfish flee method

- In a do together move the tail to look like it is swimming and move in some random direction



Final chase code

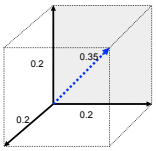


The shark will catch the goldfish

- How do we know the shark will eventually catch the goldfish?
  - The shark always moves 0.4 meters toward the goldfish
  - The goldfish's random motion is restricted by the *min* and *max* values used in the *random number* function.

The loop will end

- Geometrically, the fish can never move more than 0.35 meters away
- The shark has a distance advantage and will eventually catch up. The loop will end.



General "Rule of Thumb"

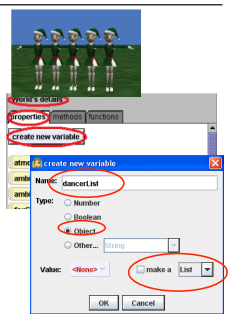
- As a general rule, a *While* loop should be written so the loop will eventually end.
  - Requires that statements within the loop change the conditions of the world such that the condition for the *While* statement will eventually become false.
- If the loop never ends, it is an **infinite** loop.

Creating and looping through lists

- A list is a popular way to organize information
  - Like a grocery list
- You can create a list in Alice
  - And put items of the same type in the list
    - Like dancers
  - Then loop through all items in the list
    - And do some action at the same time
    - Or do some action one item in the list at a time

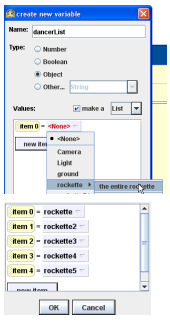
Creating a List

- Open Chap09Rockettes.a2w
- Create a new world variable
  - Click on world in the object tree
  - Click on the properties tab
  - Click on "create new variable"
  - Type in "dancerList" for the name
  - And pick Object for the type
  - Then check make a List



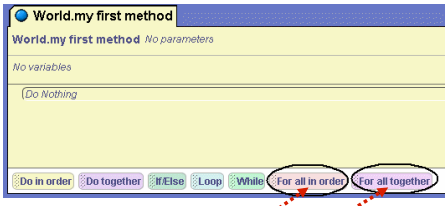
Adding to a list

- Click the "new item" button
- Then click the down arrow next to the "None"
- Select a rockette
  - The entire rockette
- Do this for the other rockettes
  - Add them in order
- Then click "OK"



Iterating through a list

- Iteration can be done in two ways:



  - in order (one at a time)
  - all together (simultaneously)

Example: Iteration in Order

- A typical chorus line dance involves each dancer (one after the other) kicking up one leg.
- Possible storyboard:

For all dancers in order
item_from_dancerList kickUpRightLeg

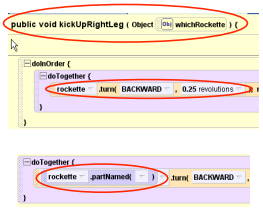
kickUpRightLeg

Parameter: *whichRockette*

doInOrder
  doTogether
    whichRockette right thigh turn back
    whichRockette right calf turn forward
  whichRockette right calf turn back

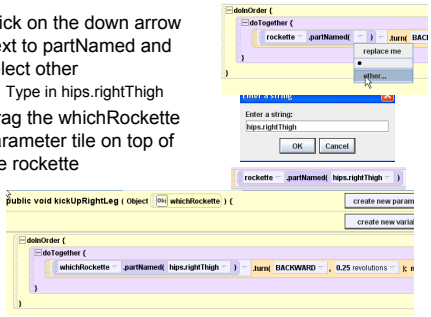
Creating a kickUpRightLeg method

- Create a kickUpRightLeg method
  - pass in which rockette as an object
  - because we are trying to access just part of the object
- Add a rockette turn backward 0.25 tile
- Replace the rockette with a partNamed function from rockette
  - You will type in the name of the part



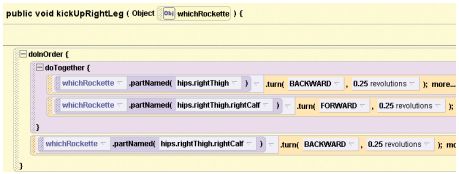
Finishing the kickUpRightLeg method

- Click on the down arrow next to partNamed and select other
  - Type in hips.rightThigh
- Drag the whichRockette parameter tile on top of the rockette



Final kickUpRightLeg Method

- Do the same thing for the hips.rightThigh.rightCalf and turn it forward 0.25 revolutions
- Then backward outside of the do together

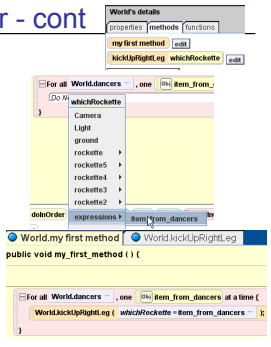


Using forAllInOrder

- Drag up a forAllInOrder tile and put it in my_first_method
- Pick expressions and the list

Using forAllInOrder - cont

- Drag out a kickUpRightLeg tile
  - Select expressions and item_from_dancers
- Try it out
  - Hit the play button
- Write a bringRightLegDown method



Using forAllTogether

- Now try the forAllTogether

Searching through a list

- Open Chap09WacAMole
  - 12 moles have been added, one in each hole
- The game will randomly select a mole to move up
  - the player needs to click on the mole with the mouse before it hides again
- 2 cylinders are used to show the players score and the score you need to win (totalScore)
  - As the playerScore increases the playerScore cylinder is moved up. The player wins when the cylinders are the same height.

Storyboard

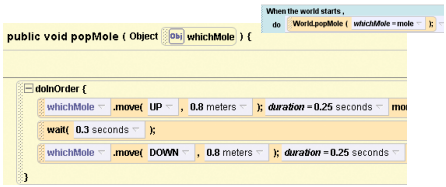
Event: *When the world starts*
Response: *World.myFirstMethod*
while the *playerScore* column is not yet completely above ground
pop (and lower) a random mole
Event: *User clicks mouse*
Response: *World.score*
If mouse was clicked on one of the moles
doTogether
move the *playerScore* column upward
play a pop sound (sound is optional)

Getting started

- When the play button is pushed Alice executes World.my_first_method
  - So we will write while loop that controls the game there. We will loop while the playerScore cylinder is not above the ground.
    - Use "! a" from World – functions for the negation

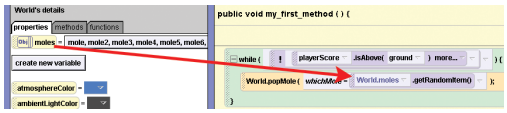
Popping up a random mole

- Write the popMole method
  - You can test just this by changing what happens when the World starts to popMole



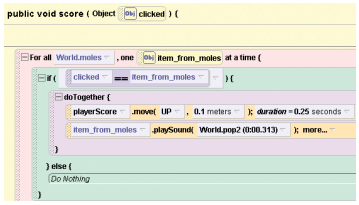
How to pop-up a random mole?

- Put all the moles in a World list called moles
  - Randomly select one to pop using getRandomItem()



Add a score method

- This method will be called when the mouse is clicked on something
  - It needs to check if the thing that was clicked on was a mole



Finishing the game

- Make sure that when the world starts World.my_first_method is called
- Add a new event
  - When the mouse is clicked on anything
  - Call World.score and pass what was clicked

Summary

- You an use a counted for loop to repeat a block of statements a set number of times
- You can nest loops inside of each other
- You can use a function to calculate the number of times a loop should repeat
- You can use a while loop to repeat a block of statements while a condition is true or false
- You can use a list to organize your objects
  - And to loop through them