Functions and Conditionals in Alice

Stephen Cooper
Wanda Dann
Barb Ericson
September 2010

Learning Goals

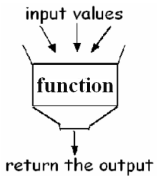
- Understand at a conceptual and practical level
  - What an Alice function is and how it differs from an Alice method
  - How to use an Alice function
  - How to create an Alice function
  - How to conditionally execute a statement or a block of statements

What is a function in Alice?

- A named group of statements that computes and returns value
- Functions are special kinds of methods
  - They don't change the state of the objects in the world, they only return a value
- Alice built-in functions let you check conditions in the world while it is running
  - Like an object's width, distance, color etc
- Functions are an abstraction

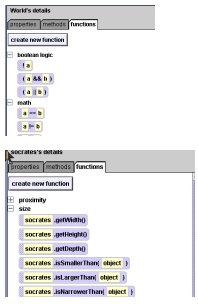
How a function works

- A **function** may receive value(s), performs some computation, and returns (sends back) a value.



Alice Functions

- The world has a set of math functions
  - and the ability to add new functions
- Each object in the world has a set of functions
  - and the ability to add new functions



Bouncing the ball over the net

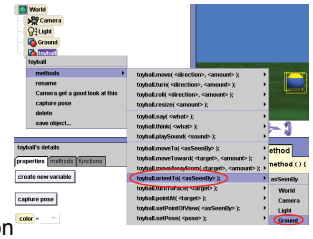
- Open Chap06BallBounceOverTennisNet.a2w
- The ball is 1 meter from the net
- We want to bounce the ball over the net

World.ballOverNet:

Do in order
  toyball turn to face the net
  Do together
    toyball move up
    toyball move forward
  Do together
    toyball move down
    toyball move forward

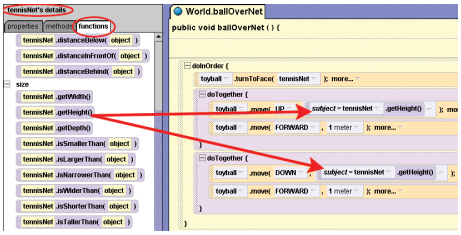
Setting the Ball's Orientation

- We don't know which way the ball thinks is up
- But we want to be able to tell the ball to move up
- So, set the ball's orientation to the ground's orientation
  - Using orientTo the Ground



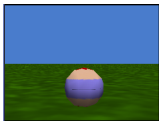
Using a built-in function in a method

- Drag out the tennisNet.getHeight() function and use it to determine how much to move the ball up and down



Creating a realistic roll

- A sports game action involves rolling a ball along the ground.
- We want a realistic motion rather than a slide.
- The ball must simultaneously move and roll.

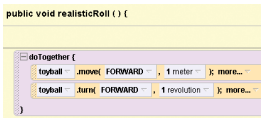


toyball.realisticRoll

Do together
move ball forward 1 meter
turn ball forward 1 revolution

Writing a realistic roll method

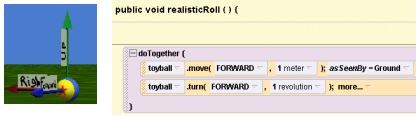
- Open Chap06BallRoll
  - Add a realisticRoll method to toyball



  - Now try it out
    - Remember to add a call to realisticRoll to my_first_method
    - Does it do what you expected?

What went wrong?

- Our design assumed that the ball's forward motion is relative to the ground. But the ball moves forward relative to it's own sense of direction (which is constantly changing as it rolls)
- *AsSeenBy ground* can be used to make the ball turn relative to the ground
  - Add it to the move forward

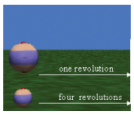


Revising the approach

- The ball is made to roll 1 revolution. Suppose we want the ball to roll a certain distance (say 3 or 4 or even 10 meters) along the ground.
- How many times should the ball turn 1 complete revolution?

Number of revolutions

- The number of revolutions depends on the size of the ball
  - The number of revolutions is distance/(Pi * diameter)



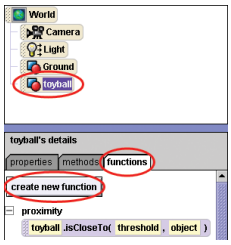
- But there is no built-in function to return the number of revolutions
- We will write our own!

Parameters

- We want to return the value computed as
  distance/(Π * diameter)
- Obviously, what is needed is
  - the ball's diameter
    - the ball object has a built-in *width* function
  - the distance the ball is to travel
    - can be a parameter of the function

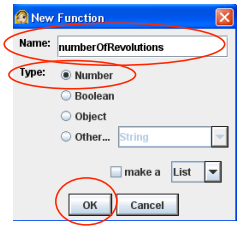
Creating a new function

- Click on toyball in the object tree
- Click on the functions tab in the details
- Click the "create new function"



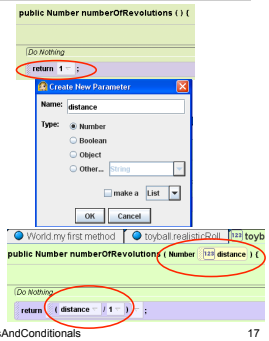
Entering function name and return type

- Type in the function name
  - Lower case first letter and upper case the first letter of each new word
- Pick the type of value returned
  - The number of revolutions will be a number
- Click "OK"



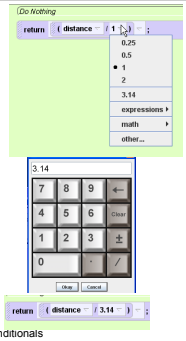
Creating the function

- Functions always return a value
  - it starts with a default value of 1
- We want to return
  distance/(Π * diameter)
  - We need a parameter that specifies the distance
  - Drag the distance tile over the 1 and use the arrow to add the / 1



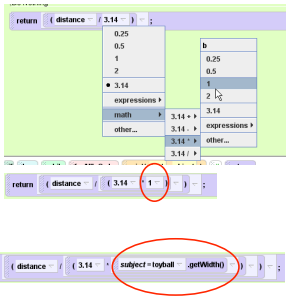
Adding Pi (3.14)

- Click on the down arrow next to the 1
  - And select other
- Use the custom number to enter
  - 3.14

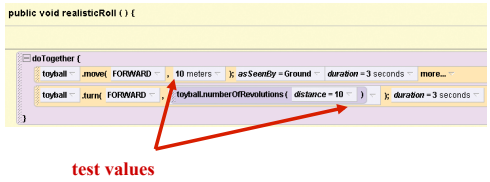


Multiplying pi by the diameter

- To finish the calculation we want to multiply 3.14 by the diameter of the ball
  - Click on the down arrow next to 3.14 and select math then 3.14 * 1
  - Replace the 1 with the built-in getWidth() function



Calling the function



test values

We should run the animation with several test values to be sure it works as expected.

What happens if you use a negative value?

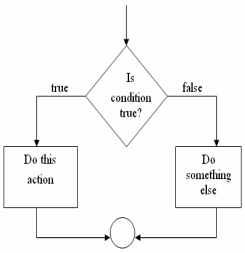
Java versus Alice

- In Java we only have methods
  - Methods can return a value or not
  - We have been only writing methods that don't return anything
    - So the return type is void to show that nothing is returned
  - To change a Java method to be like an Alice function
    - Change the return type to match the value being returned like int for an integer
    - Add a return statement to the method
    - Methods can have more than one return statement

Conditionals

- Allow you to execute some code when a Boolean expression is true
- If it is raining
  - Take an umbrella
- If you get your homework done
  - You can go out tonight
  - otherwise (else) you can't
- If the basketball goes through the hoop
  - Add to the score

If/Else



- In Alice, a logical expression is used as the condition in an ***If/ Else*** control structure.
- Decisions (using If/Else) are used in
  - functions
  - methods

Example: Using a conditional

- Suppose you are building a simulation system used to train air traffic controllers.
- One of the tasks of an traffic controller is to be alert for possible collisions in the flight space.
- Open Chap06FlySpaceCollision.a2w

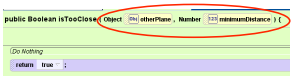
Storyboard

- One factor in determining whether two aircraft are in danger of collision is the vertical distance (difference in altitudes) between them.
- We can write a function that checks the vertical distance against a minimum difference in altitudes.
- The function returns *true* if they are too close, otherwise *false*.

biplane.isTooClose
Parameters: *otherPlane, minimumDistance*
If the vertical distance to the other plane is less than *minimumDistance*
return *true*
Else
return *false*

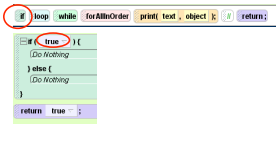
Create a function biplane.isTooClose

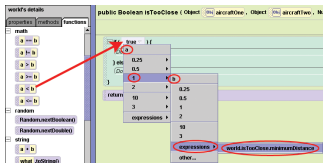
- It will return true if the biplane it too close to another plane and otherwise return false
- True and false are Boolean values
- Create a parameter of type Object for the other plane
- And a parameter of type Number for the minimum distance



Adding an if and a relational operator

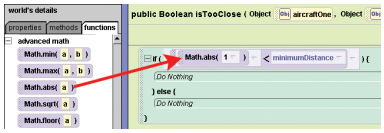
- Drag up an if before the return
- Drag out the "a < b" tile from the world functions to replace the "true"
- pick 1 for the value of a and for b pick expressions and then the minimumDistance



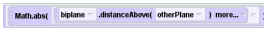


Add the absolute value

- Drag out from World functions the Math.abs(a) and put in on the 1
  - Be sure to drop when the 1 is boxed in green

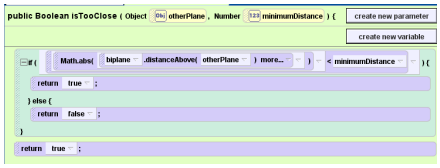


  - Replace the 1 with biplane.distanceAbove and select expressions – otherPlane



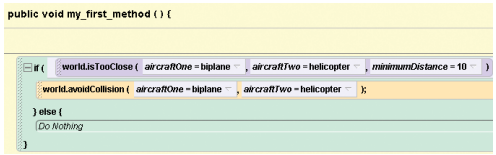
Final istooClose function

- When the absolute value of the distance that the biplane is above the other plane is less than the passed minimum distance return true, else return false



Using the isTooClose function

- If the biplane is too close to the helicopter then avoid the collision
  - Else do nothing

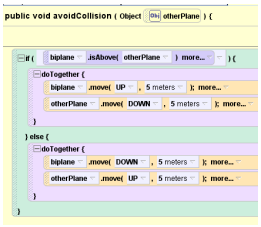


The avoidCollision Method

avoidCollision

Parameters: *otherPlane*

If biplane is above otherPlane
  Do together
    biplane move up
    otherPlane move down
Else
  Do together
    biplane move down
    otherPlane move up



Summary

- Use a function in Alice to return a value
- Use the if-else statement for conditional execution based on a Boolean test
  - A Boolean test is true or false

```
if (boolean test)
{
   // statements to execute if the test is true
}
else
{
  // statements to execute if the test is false
}
```