CompSci 4
Chap 6 Sec 2
Sep 29, 2005

Prof. Susan Rodger
Announcements

• Review for test next time.
  – Hand out Midterm from last semester
    • Note – Midterm covered Chap 5, not Chap6!
  – Old Quizzes will be available on Blackboard
  – Study classwork and lecture notes

• Next assignment handed out after fall break
  – enjoy your break!
What we will do today

• Lecture on Chap 6, Sec 2 through page 167
  – Execution control with if/else statements and Boolean functions
  – Relational operators
  – Logical operators
• Exam 1 covers up through today's topic.
• Classwork
Thinking about More Advanced Worlds

• How do you build animations like simulations and video games?
• Need to write code that involves decisions
Examples of Decisions

• A car-race simulation – the driver steers the car around curves and past mile markers
  – If the car stays on the road the score increases
  – If the car goes off the road into the stands, the car crashes
  – If the driver gets the car over the finish line, the time is posted and the driver wins!
Logical Expressions

- Decision is made based on current conditions.
- Condition is checked in a logical expression that evaluates to *true* or *false* (Boolean) value.
  - car on road → true
  - car over finish line → false
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: Boolean Functions

• Suppose we build a simulation system used to train flight controllers

• One of the tasks of a flight controller is to be alert for possible collisions in flight space
Storyboard

- Two aircraft – biplane and helicopter
- As the biplane moves towards the helicopter we want to make sure they do not collide
- If they are too close, they need to adjust their altitude (height)
Storyboard (cont)

• Two factors in determining whether two aircraft are in danger of collision
  – distance between them
  – Vertical distance between them
• We can write functions to determine these
• Both functions return true if aircraft are too close, otherwise false
isTooCloseByDistance:

Parameters: aircraft1, aircraft2, minDistance

If distance between aircraft1 and aircraft2 is less than minDistance
  return true
Else
  return false
Using a Relational Operator

• Use the < relational operator from the World’s built-in functions to check the distance against the minimum
Implementing the Function

```
World.IsTooCloseByDistance (Obj aircraft1, Obj aircraft2, 123 minDistance)

No variables

If
    aircraft1 distance to aircraft2 < minDistance
    Return true
Else
    Return false

Return true
```
Vertical Distance Function

• To find the difference in altitude, use the built-in *distance above* function
  – Don’t know which aircraft is above the other
  – To avoid a possible negative value, use *absolute value* of the distance
istooCloseByVertical
Storyboard

forwardAndCheckCollision

Parameters: aircraft1, aircraft2, distance

aircraft1 move forward distance
If aircraft1 and aircraft2 are closer than twice distance
  avoid collision if they are too close heightwise
  move aircraft1 forward twice the distance
Implementation and Calling Function

```
World.forwardAndCheckCollision (Obj aircraft1, Obj aircraft2, Int distance) {
    No variables

    If World.IsTooCloseByDistance (Obj aircraft1 = aircraft1, Obj aircraft2 = aircraft2, Int minDistance = distance * 2) {
        World.adjustForHeightCollision (Obj aircraft1 = aircraft1, Obj aircraft2 = aircraft2, Int distance = distance)
    }
    Else {
        Move forward (distance * 2)
    }
    Do Nothing
}
```
adjustForHeightCollision

World.adjustForHeightCollision(Obj aircraft1, Obj aircraft2, 123 distance)

No variables

If World.IsTooCloseByVertical aircraftOne = aircraft1 \ aircraftTwo = aircraft2 \ minDistance = distance

World.avoidCollision aircraftOne = aircraft1 \ aircraftTwo = aircraft2

Else
(Do Nothing)
Avoid Collision

World.avoidCollision

World.avoidCollision (Obj aircraftOne, Obj aircraftTwo)

No variables

- If
  - aircraftOne is above aircraftTwo
    - Do together
      - aircraftOne move up 5 meters
      - aircraftTwo move down 5 meters
  - Else
    - Do together
      - aircraftOne move down 5 meters
      - aircraftTwo move up 5 meters
Putting it All Together - Demo

```
World.my first method  No parameters

No variables

biplane  turn to face  helicopter  more...

biplane.backup  distance = 25

// Run simulation with different heights for helicopter, - up 5, up 10, the same

helicopter  move  up  5 meters  more...

Start!  set  isShowing to  true  more...

Camera  move  backward  25 meters  more...

Start!  set  isShowing to  false  more...

// run simulation

Do in order

World.forwardAndCheckCollision  aircraft1 = biplane  aircraft2 = helicopter  distance = 10

World.forwardAndCheckCollision  aircraft1 = biplane  aircraft2 = helicopter  distance = 10

World.forwardAndCheckCollision  aircraft1 = biplane  aircraft2 = helicopter  distance = 10

World.forwardAndCheckCollision  aircraft1 = biplane  aircraft2 = helicopter  distance = 10

World.forwardAndCheckCollision  aircraft1 = biplane  aircraft2 = helicopter  distance = 10

World.forwardAndCheckCollision  aircraft1 = biplane  aircraft2 = helicopter  distance = 10
```
Demo and Testing

• Try helicopter at different heights
  – Move up 5 meters
  – Move up 10 meters
  – Stay the same
  – Down 5 meters
Problem

• The helicopter may go below the ground!

• How do we fix this?
  – Only move down if above a certain distance!
  – Use nested if’s to check more than one condition
How do we fix this? Nest ifs
Another Way - Logical Operators

- Use Boolean logic operators to check more than one condition.
Check

- Where do you get the if?
- Do you have to fill all the parts of the if?
- Where do you find the relational operators?
- Where do you find the logical operators?
Classwork today
avoidCollisionGroundCheck1

If aircraftOne is above aircraftTwo more...

If aircraftTwo distance above ground more... >= 5

Do together
- aircraftOne move up 5 meters more...
- aircraftTwo move down 5 meters more...

Else
- aircraftOne move up 10 meters more...

Else
    // aircraftTwo is equal height or above aircraftOne

If aircraftOne distance above ground more... >= 5

Do together
- aircraftOne move down 5 meters more...
- aircraftTwo move up 5 meters more...

Else
- aircraftTwo move up 10 meters more...