Class Today

- Last time we used functions, this time we will write functions
- Continue to make decisions with IF statement
- Use logic instead of nested IFs
- Inheritance – how it relates to Alice
- making a character more functional, then saving it to put in other Alice worlds

Review 1

- What is the difference between a function and a procedure?
  - Procedure is something to do – turn, move, dance
  - Function is a calculated value – a number, an object, a direction
  - A function by itself is not very useful, a function has to be used in some way based on the type of value it calculates
Review 2

• What happens if the diningTable has width 1.7 and depth 1.0?

  – Say hello and the table is resized larger by 1.2

Logic Operators

• A and B – true if both are true
• A or B – true if one or both are true
• Not A – True if A was false

Logic Truth Table

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>not A</th>
<th>A and B</th>
<th>A or B</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>
Consider this code – Nested IFs

Rewrite Code - one IF with AND
What we want to do today

1. Teach a dog to jump over two creatures at once (write a function to determine how high the dog must jump?)
2. Have the taller of two creatures spin around and say I’m taller (write a function to determine the taller object)
3. Learn about inheritance
4. Use the jumping dog in another Alice world

Characters in Lecture World today

• Note Different types:
  – Bipeds: Alice, ChesireCat, Alien
  – Quadruped: Dalmatian
  – Flyer: Flamingo

Last time, Built-in functions

• You can use a built-in function anywhere that matches the type of value it calculates
• Move up 2.0
  ![Diagram of moving up 2.0]
• Drag over getHeight (value is a decimal number), and now moves up the dalmatian’s height

Can write your own functions

Function for Scene    OR   Function for character

Use scene function if it involves multiple objects

![Diagram of class definitions]
1) Write Scene function tallerHeight

• Inputs: two objects
• Output: the height (decimal number) of the taller object

Add two parameters – what type should we choose?

Start Scene function tallerHeight

• The start of the function

Best choice, why?

• First level that includes all our objects
Function tallerHeight – add parameters

- Added parameters of type SjointedModel: object1 and object2.
- Object1 has its own procedures and functions

How to start function tallerHeight

- We need to calculate which object is taller and then return the height of the taller object
- How do we calculate the taller value?

Problem Solve

Write out what you want to do!

```
if object1 is taller than object 2
    return object1’s height
else
    return object2’s height
```

Then convert to code!
Drag parameters to code carefully!
• Make sure you choose the correct object too!
• Object1 versus Object2

To use tallerHeight, first we wrote a Dalmatian jump Proc

Using tallerHeight
• Want the dalmatian to jump over Alice and the CheshireCat. Needs to know the height of the taller one to know how high to jump.

Using tallerHeight
• Replace the number 2.0 with the tallerHeight function, it’s a number!
• Run the world, what happens?
• Make Alice or CheshireCat taller and run again? What happens?
2) Write Scene function tallerObject

- Inputs: two objects
- Output: the taller object

\[ \text{tallerObject}(\text{object1}, \text{object2}) \]

Resulting code for tallerObject

- This returns the object who is taller

\[ \text{if } \text{object1}.\text{get-Height} > \text{object2}.\text{get-Height} \text{ then return object1; else return object2; } \]

How do you use tallerObject?

- Want the taller of Alice or Cheshire cat to say they are taller, then spin around once, and then shrink
- Must first put each command for an object, say Alice, then replace it with the function
What does this code do?

3) Inheritance
• There is a hierarchtical structure
• Alice and Alien are bipeds
• Flamingo and Alien are SjointedModels and everything above that: Smodel, Sturnable, etc

4) Inheritance - Save jumping Dalmation and put in another Alice world
• Our Dalmation knows how to jump. Since we wrote the jump method as a Dalmatian method, we can save the Dalmatian out as a DalmatianJumper, a smarter dog who inherits all the functionality of a Dalmatian but also knows how to jump
To save the Dalmation as a class, an .a3c file

- Click on Save to Class File
- I saved him in the folder *My Classes*
- I named him DalmatianJumper

Now you can put the Dalmation in another world with the jumping procedure

- Start a new Alice world on the moon
- Add in a regular Dalmatian
- Note the dog doesn’t know how to jump

Now add in a jumping Dalmatian

- Look in my Classes
- Find the DalmatianJumper and add it

You will be asked if you want the extra procedures, yes you do!
Now both all Dalmatians know how to jump

- Including the one you already added

This lecture covered

- Using Logic in place of nested Ifs
- Writing your own function. A function needs input (parameters) and has a single output of a specified type
- Inheritance – the hierarchical structure of the Alice objects
- Saving an object with functionality and using it in another Alice world