

Digital Multimedia Art

CSCI-EI I Roy Pardi

Grace Murray Hopper

Computer Pioneer & Mathematician

A Foremother of the information age, Grace Hopper (1906-1992) was instrumental in creating and standardizing many of its raw materials—most notably COBOL, the business programming language. Hopper was a professor of mathematics at Vassar when the navy recruited her in 1943 to work on the first computer in the United States, the Mark I, at Harvard. There she built the first compiler to translate mathematical notation into machine code. The later development of COBOL arose from her realization that most people communicate better with prose than with mathematical symbols—and that future computer use would not be limited to science.



Week 4: Interactivity (Programming I)

92

9/9

0800 Aiken started
 1000 " stopped - aiken ✓
 13⁰⁰ (032) MP - MC ~~1.58264000~~ { 1.2700 9.037 847 025
 (033) PRO 2 2.130476415 ~~(2)~~ 4.615925059(-2)
 connect 2.130676415

Relays 6-2 in 033 failed special speed test
 in relay .. 10.000 test.

Relay
 2145
 Relay 337

1100 Started Cosine Tape (Sine check)
 1525 Started Multi-Adder Test.

1545



Relay #70 Panel F
 (moth) in relay.

First actual case of bug being found.

~~1630~~ 1630 Aiken started.
 1700 closed down.

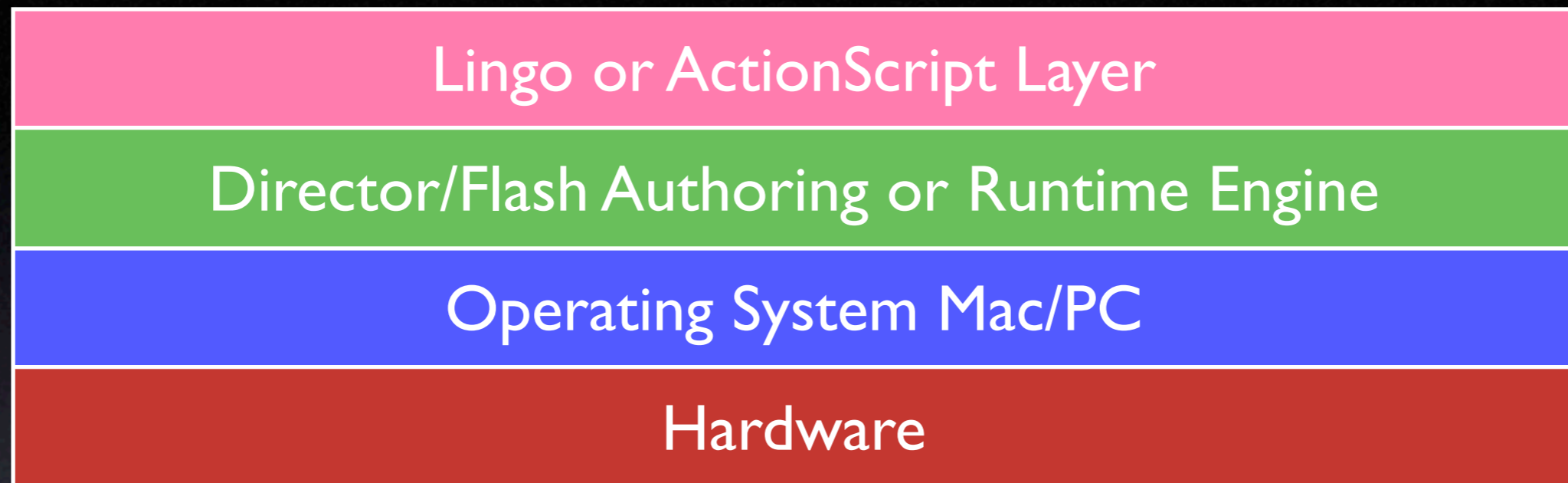
The First Computer Bug

Moth found trapped between points at Relay # 70, Panel F, of the Mark II Aiken Relay Calculator while it was being tested at Harvard University, 9 September 1945. The operators affixed the moth to the computer log, with the entry: "First actual case of bug being found". They put out the word that they had "debugged" the machine, thus introducing the term "debugging a computer program".

You already know how to program.

(yeah, rilly...)

Key Programming Concepts



- Scripting languages compared to programming languages
- Objects
- Algorithmic thinking
- Events and event handlers
- Language elements & syntax

Objects

Thinking in terms of OBJECTS:

- Everything in Director & Flash is an **OBJECT**
- Objects have **PROPERTIES** - properties are attributes that define the object
- There can be multiple **INSTANCES** of objects each with unique property values

Phases in the Programming Task

Problem Solving Phase

- Analysis and Specification
Understand (define) the problem and what the solution must do
- Algorithm Development
Develop a comprehensive unambiguous logical sequence of steps to solve the problem
- Verification of Algorithm
Follow steps closely (manually) to see if solution works

Implementation Phase

- Program Development
Translate algorithm into a program written in a programming language
- Program Testing
Test program for syntactical and logical errors. Fix the errors.

Event Model

Both Lingo and AS are event-driven - so rather than continually polling the state of something (like the mouse), messages are sent when events occur

Director	Flash
<ul style="list-style-type: none">Frame eventsSprite eventsMovie eventsIdle eventsKeyboard and mouse eventsMovie in a window (MIAW) eventsSynchronizing media eventsBrowser and Internet eventsTimeout events	<ul style="list-style-type: none">Button eventsMovie clip eventsKeyboard and mouse eventsSynchronizing media eventsBrowser and Internet eventsTimeout events

Programing Elements

These elements are common to all programming languages

- Conditional statements
- Compound statements
- Variables
- Arrays
- Operators
- Data types

Conditional Statements

Conditional statements are used to test if some condition evaluates to TRUE or FALSE

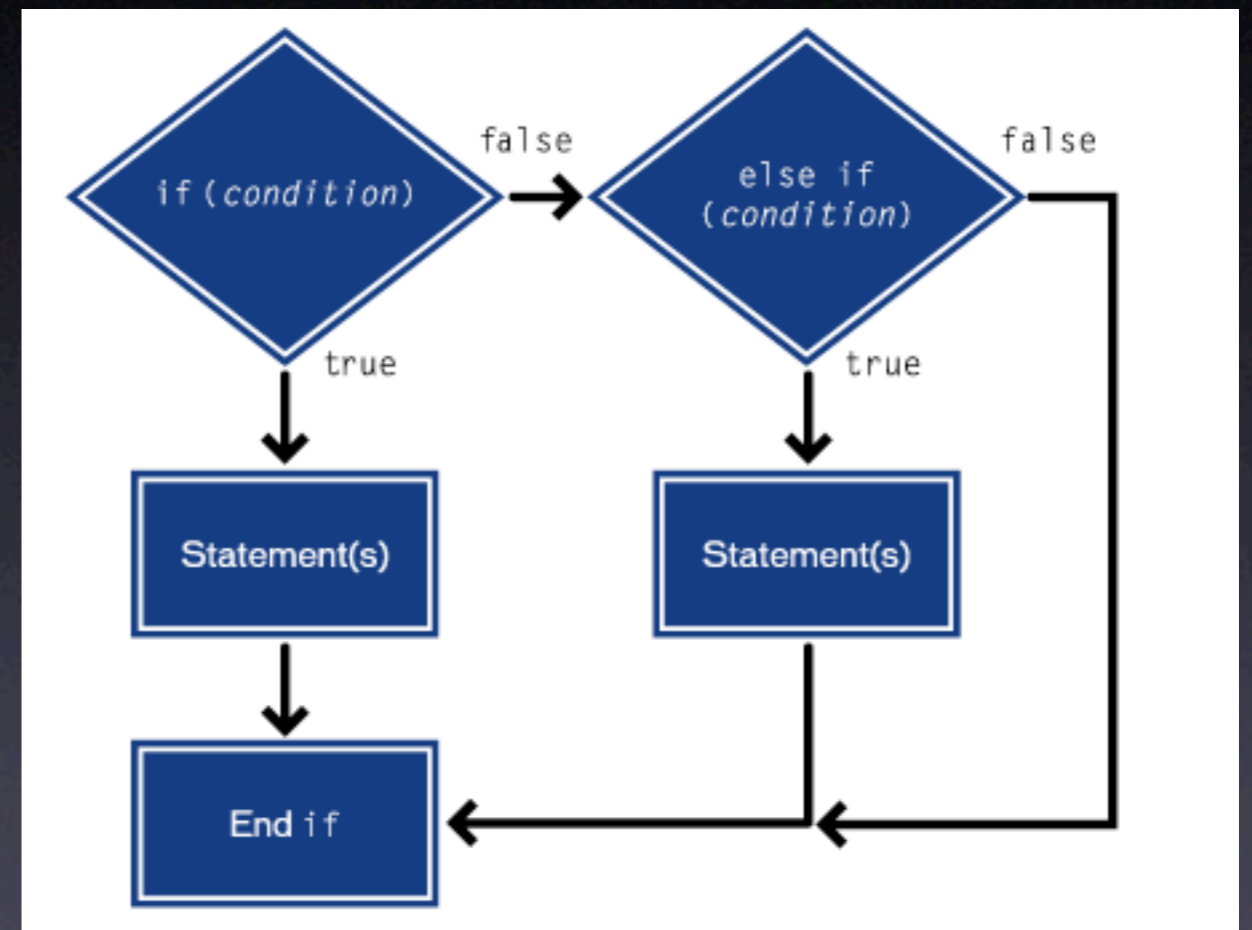
Verbose pseudo code:

```
if it is true that the forecast is for rain then  
    bring umbrella  
end if
```

Simplified pseudo code:

```
if the forecast = rain then  
    bring umbrella  
end if
```

All programming and scripting languages utilize conditionals - only the syntax varies



Conditional Statements

Default condition

```
if the forecast = rain then
  bring umbrella
else
  put umbrella away --this is the default
end if
```

Multiple tests

```
if the forecast = rain then
  bring umbrella
else if the forecast = snow then
  put on boots
end if
```

Conditional Statements

Conditional Statements can be nested

--if the first statement is FALSE then the rest of the
--statement is not evaluated

```
if the forecast = rain then
  if going to work then
    bring umbrella
  else if going hiking then
    bring raincoat
  end if
end if
```

Conditional Statements

Lingo: supports two types of syntax

Director	
<p>if - then - else</p> <pre>if x = n then --do something else --do something else end if</pre>	<p>case</p> <pre>case(n) of x: --do something otherwise --do something else end case</pre>

Conditional Statements

ActionScript: supports two types of syntax

Flash

if - else if

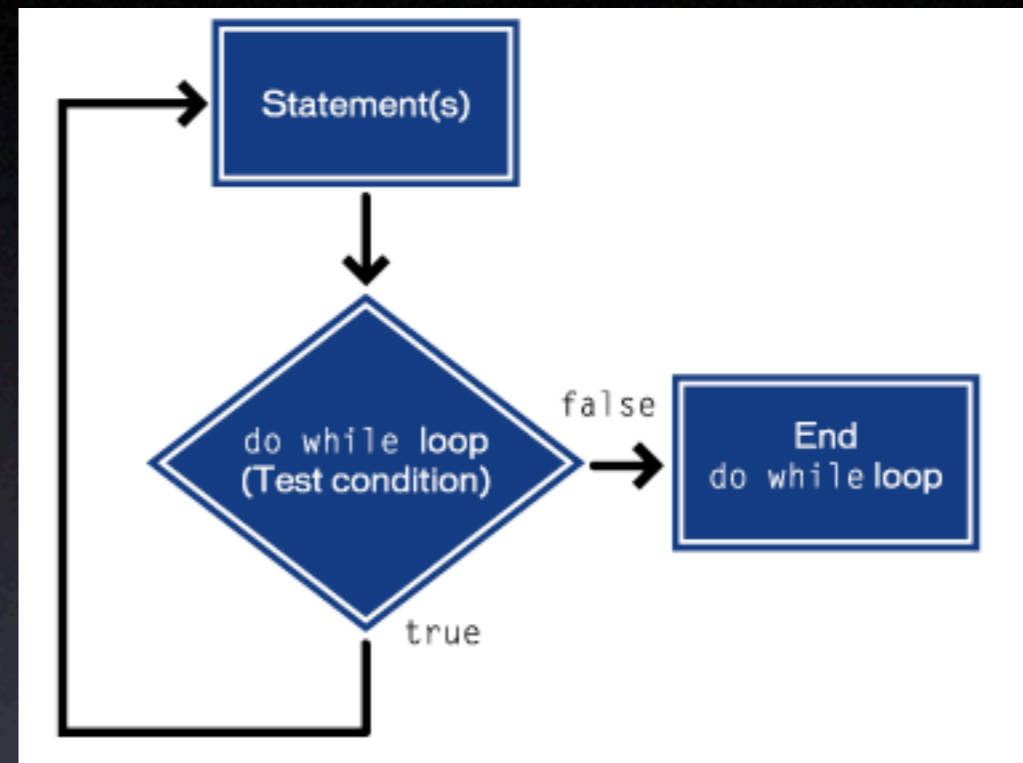
```
if (x = n) {  
    // do something  
    put "x = n"  
} else if (y = n) {  
    // do something else  
    put "x = n"  
}
```

switch

```
switch (n) {  
    case y:  
        // do something  
        break;  
    case y:  
        // do something else  
        break;  
    default:  
        // do something else  
}
```

Loops (Compound Statements)

- Loops tell the program to execute some code repeatedly until some condition is met
- Loops are very fast but lock out the application (and OS) - taking all the CPU time
- If you need to have the program do something more than once use a loop



Loops (Compound Statements)

Director	Flash
<pre>repeat with i = 1 to n --do something end repeat</pre>	<pre>do { // do something i++; } while (i < n)</pre>
<pre>repeat with i = n down to 1 --do something end repeat</pre>	<pre>i = 0; while (true) { // do something if (i >= 100) { break; } i++; }</pre>
<pre>repeat while i < n --do something i = i + 1 end repeat</pre>	
<pre>repeat with variable in list --do something end repeat</pre>	

Variables

- A variable is a container that holds information. The container itself is always the same, but the contents can change
- Variables can hold any type of data: number, string, Boolean, object, etc.
- Variables have scope; a variable's "scope" refers to the area in which the variable is known and can be referenced

Director	Flash
Local variables - available within their own block of code	
Global variables - available from anywhere in the movie (are “global” in scope)	
Property variables - only available to the object or behavior in which they have been declared	Property variable - only available to the object in which they have been declared
	Timeline variables - available to any Timeline if you use a target path

Lists - Arrays

Arrays are a type of container (code object in memory) that enable the storing of data in structured ways

Lingo and AS have specific commands and functions for working with lists/arrays

Multi-dimensional arrays

--

	Director	Flash
Types	<p>Linear List: myList = ["dog", "cat", "fish"]</p> <p>Property List: myList = [#pet:"cat", #name: "Poppy", #type: "calico"]</p>	<p>Array thisArray = new Array("dog", "cat", "fish");</p> <p>Named Array thisArray = new Array(); thisArray["pet"] = "cat"; thisArray["name"] = "Poppy"; thisArray["type"] = "calico";</p>
Usage	Lingo lists can store any Director data type	ActionScript arrays can only store strings, integers, floats and arrays (TRUE?)

Operators

Operators are elements that specify how to combine, compare, or modify the values of an expression. They include the following:

- Arithmetic operators
- Comparison operators which compare two arguments
- Logical operators which combine simple conditions into compound ones
- String operators which join strings of characters

Data Types

- A data type describes the kind of information a variable can hold
- There are two kinds of data types: primitive and reference
- Primitive data types—string, number, and boolean—have a constant value and therefore can hold the actual value of the element they represent
- Reference data types—have values that can change and therefore contain references to the actual value of the element

Authoring Conventions/Differences

Director	Flash
The Score	The Timeline
Behavior Inspector Create and edit scripts through GUI	Normal Mode Create and edit scripts through GUI
Script Window Create and write scripts directly (This is set under Preferences/Editors/ Behaviors)	Expert Mode Create and write scripts directly (This is set under pulldown menu in ActionScript window)

Authoring Conventions/Differences

	Director	Flash
Where Code Goes	Four types of script cast members Movie, Behavior, Parent, Cast Member	Always attached to timeline Does not appear in Library
Code Structure	<pre>on myMethod(parameter) --do some work end on myMethod(me, parameter) --do some work end</pre>	<pre>function myMethod(parameter) { // do some work }; myObject.prototype.myMethod = function(parm) { // do some work };</pre>

Authoring Conventions/Differences

	Director	Flash
Comments	-- each line	// each line or /* multiline block */
Compile	playback in authoring & runtime author controlled	test movie
Feedback	Message Window view results and execute code	Output Window view results only
Debugging	Built-in Debugger + Object Inspector Authoring & Runtime error alerts	Built-in Debugger + Object Inspector(expert mode only) Authoring alerts - fails silently at runtime
Code Reference	Pull-down alphabetical and categorical lingo listings in script window	Categorical Actionscript listings in Actions window

Media Types

Director		Flash
#animgif	#filmLoop	
#palette	#swa	
#bitmap	#flash	bitmaps
#picture	#text	vectors
#button	#shape	quicktime
#QuickTimeMedia	#empty	.swf
#cursor	#shockwave3D	?
#script	#field	?
#digitalVideo	#sound	?
#movie	#font	?
#vectorShape	#transition	