

Art 486: Advanced Interactive Media

mcdo@umbc.edu

Lecture 4, Sept 10

Schedule

- Take Roll
- practice test
- grade practice test
- functions

practice test 1

www.umbc.edu/~mcdo/486/testA.pdf

functions

- you are familiar with function calls
 - gotoAndPlay(20);
- functions have a name (“gotoAndCall”)
- functions have parameters – inputs (in this case, 20)

making new functions

```
// initializes debris to fly out
public function initExplosion():void {
    var i:int;
    sym_castle.x = -1000;
    for (i=0; i<60; ++i) {
        debris[i][0] = Math.random()*6.28; // travel direction
        debris[i][1] = Math.random()*10.0 + 5.0; // travel speed
        debris[i][2] = Math.random()*40.0 + 10.0; // length
        debris[i][3] = Math.random()*6.28; // starting angle
        debris[i][4] = Math.random()* .2 - .1; // spin rate
        debris[i][5] = Math.random()*30.0 + 60.0; // ejection time
    }
}
```

```
function initExplosion():void {
```

- the word “function” tells AS3 that you’re declaring a function
- this function will be called “initExplosion”
- it takes no parameters
- “:void” it does not return anything
- and then the “body” of the function
 - same as the body of a loop
 - containing one or more statements

a simpler example

- `function mine(i:int):int {`
- `return i*2;`
- `}`

- named “mine”
- takes one parameter, an integer
- “returns” a value– $i*2$

why make functions?

- code reuse
 - make it work once, use it forever, everywhere
- readability
 - bundling up code that does some larger task
 - hiding the complexity of that task
- attaching functions to objects

a simplifying function

- function average(float a, float b):float {
- return (a+b)/2.0;
- }

- now, you can write an expression
 - a = average(45.0, 33.5);

```
a = average(45.0, 33.5);
```

- the code itself tells you what is in a
- once you know “average” works, it’s less to debug
 - okay, “average” is simple— how about “calculateMonthlyMorgageTax(price, rate)”
 - compare to “a = (45.0*33.5)/2.0;”

return values

- functions take input, give output
- output is sent from the function to the code that called it
- “return” is the command that sends the data
- when you call “return”, the function ends

return types

- function average(float a, float b):float {
- return (a+b)/2.0;
- }

- this returns a floating

local variables

- big topic— look out
- “local” variables exist only while the function is running, disappear afterwards
- “global” variables are what we’ve been using so far

```
function mine(i:int):int { return i*2; }
```

- “i” is an input variable, a kind of local variable
- if you have a global “i”, you can’t get to it—
 - “i*2” will use the input, return double that

a function with a local variable

- fuction a(int i):int {
- var k:int;
- k = i*2;
- return k;
- }