INFT3960 – Game Production

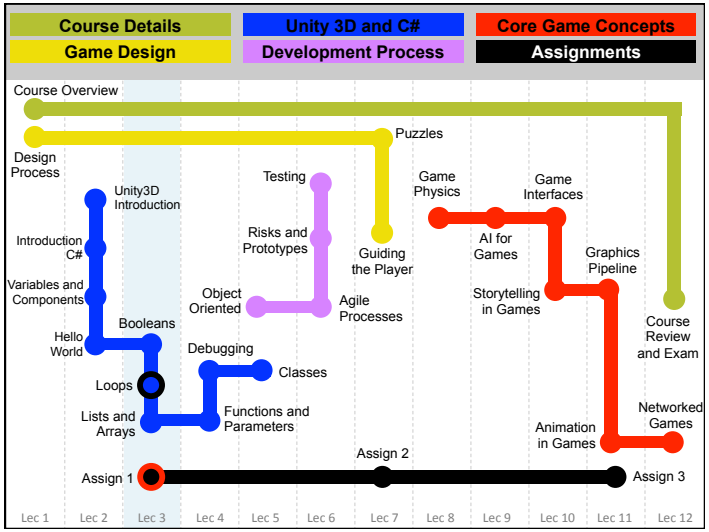
Week 03

Module 3.2

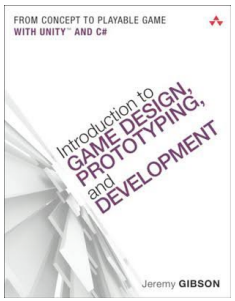
Loops

Course Overview

Lec	Date	Modules	Assignments
1	Monday 24 Jul	Mod 1.1, Mod 1.2	
2	Monday 31 Jul	Mod 2.1, Mod 2.2, Mod 2.3, Mod 2.4	
3	Monday 7 Aug	Mod 3.1, Mod 3.2, Mod 3.3	Assign 1 6 Aug, 11:00 am
4	Monday 14 Aug	Mod 4.1, Mod 4.2	
5	Monday 21 Aug	Mod 5.1, Mod 5.2	
6	Monday 28 Aug	Mod 6.1, Mod 6.2, Mod 6.3	
7	Monday 4 Sep	Mod 7.1, Mod 7.2	Assign 2 4 Sep, 11:00 am
8	Monday 11 Sep	Mod 8.1	
9	Monday 2 Oct	Mod 9.1	
10	Monday 9 Oct	Mod 10.1, Mod 10.2	
11	Monday 16 Oct	Mod 11.1, Mod 11.2	
12	Monday 23 Oct	Mod 12.1, Mod 12.2	Assign 3 23 Oct, 11:00am



Loops – (Chapter 21)



LOOPS

Loops – Topics

- Four Types of Loops
  - `while`
  - `do … while`
  - `for`
  - `foreach`
- Jump Statements in Loops
  - `break`
  - `continue`

Four Types of Loops

`while`
- The most basic loop
- Checks a condition before each loop; loops if it's true

`do … while`
- Checks a condition after each loop; loops if it's true

`for`
- Most common loop structure
- A loop structure that contains three separate statements

`foreach`
- Automatic for loop for enumerable collections

While Loop

A better while loop

while loops need an exit condition - A condition that will cause the condition to evaluate to false

Checks a condition before each loop; loops if it's true

```
int i=0;
while ( i<3 ) {
    print( "Loop: "+i );
    i++;      // Increment operator
}
```

i++ will increment i on every pass through the loop

When i reaches 3, the conditional clause will evaluate to false, and the loop will exit

do .. While Loop

Like a while loop, but checks after the loop has run
- This allows a guarantee that the loop will run at least once

Checks a condition after each loop; loops if it's true

```
int i=5;
do {
    print( "Loop: "+i );
    i++;      // Increment operator
} while (i<3);
```

When execute the loop once before checking the conditional clause and then exiting

Note the semicolon after the while clause

For Loop

A for loop contains three separate clauses

```
for (int i=0; i<3; i++) {
    print( "Loop: "+i );
}
```

Initialization clause: `int i=0;`
Condition clause: `i<3;`
Iteration clause: `i++`

The i variable only exists within the for loop - It is scoped to the for loop

The iteration clause doesn't have to be ++
`i--` is another common option for counting down instead of up

Foreach Loop

Automatically loops for each element in a collection

```
string str = "Hello";

foreach (char chr in str) {
    print( chr );
}
```

This will print each character of “Hello” individually

`foreach` will be used extensively in the following chapter

Jump Statements

Jump statements change the execution of a loop

(neither are good engineering practices – making code hard to read and maintain – for some reason used a lot in game applications – perhaps for speed of code or coding)

`break`
Breaks out of the loop entirely

`continue`
Breaks out of this iteration of the loop and moves on to the next

Break

Automatically loops for each element in a collection

```
string str = "Hello";

foreach (char chr in str) {
    print( chr );
}
```

This will print each character of “Hello” individually

`foreach` will be used extensively in the following chapter

Break

Breaks out of the loop completely

```
string str = "Hello";
foreach (char chr in str) {
    if (chr == 'l') {
        break;
    }
    print( chr );
}
```

This will print:
```
H
e
```

Once `chr` becomes '`l`', it will break out of the loop
Can be used on any kind of loop

Continue

Breaks out of the loop completely

```
string str = "Hello";
foreach (char chr in str) {
    if (chr == 'l') {
        continue;
    }
    print( chr );
}
```

This will print:
```
H
e
o
```

When chr is 'l', the loop continues without printing

Summary

Of the four types of loops:

`while` and `do…while` are somewhat dangerous
(actually this a silly thing to say - just require good coding)

`for` is by far the most common and is very flexible

`foreach` is very useful for strings, arrays, and Lists

Jump statements can be used to have more control over your loops - A break can be used to break out of an infinite loop as well
(these get used in games for speed but goto like statements are otherwise poor engineering)