Lesson 5: Loops

3D Game Programming With C++

Digital Media Academy (Summer 2011)

Written by: Andrew Uzilov (andrew.uzilov@gmail.com)
Feel free to contact me with any questions.

Sometimes you want to make the computer do something over and over. After all, computers were invented to do repetitive tasks. We can do that with loops. There are two kinds (`for` loops and `while` loops), depending on the occasion!

Every loop has a loop body and a conditional that tells us if we should keep running the loop or stop. It works like this:

Check the conditional, then:

- If it is `true`, we run the loop body exactly once, and go to the "check the conditional" step.
- If it is `false`, we quit and go to whatever code is after the loop body.

WHILE LOOPS

These kinds of loops will keep executing "while" some condition is true. The basic syntax of a "while" loop is this:

```
while (health > 0) {
    /* some code for doing game stuff must go here */
}
```

The part between the parentheses is the test (conditional expression). If the test is true, we execute the body of the loop (which can change the value of `health`), then do the test again, and if the test is true execute again, and so forth… rinse and repeat.

Here is an example that shows how we can "break out" of a loop – we can prevent any future loop iterations from happening by changing the state of a boolean variable.

```
bool gameOver = false;
while (!gameOver) {
    /* fight enemies! */
    if (health <= 0) {
        gameOver = true;  // this will break out of the loop
    }
}
cout << "Game over!" << endl;
```

FOR LOOPS

These are useful if you want to loop a specific number of times. They consist of three parts, separated by semicolons (`;`). The parts are:

1) The initialization. This part executes exactly once before the loop starts, no matter what. You can declare and/or define a variable here.

2) The test. Just like in "while" loops, we will do this test before each loop iteration, which will decide if we should do the loop body or not.

3) The step. This part executes exactly once at the end of each loop iteration.

Here is an example "for" loop that prints out six numbers (0, 10, 20, 30, 40, 50), with the three parts color-coded as above:

```
for (int i = 0; i < 6; i++) {
      cout << i * 10 << endl;
}
cout << "Done!" << endl;
```

Try putting this into a new Eclipse project and compiling/running it!

The code works like this:

- We create an `int` variable named `i` and initialize its value to 0.
- We test to see if `i` is less than 6. The first test will pass and the loop body will print "0" (because zero times 10 is zero).
- We increment `i` by 1 (`i++`), so `i` now equals 1.
- We test `i < 6` again, which passes, and the loop body will print 10; increment `i` by 1 again.
- The above steps continue until `i` reaches 6. Then, the test will fail and the loop will quit. The next statement will print "Done!".

Just for the comparison, the following "while" loop will do the same exact thing as the previous "for" loop:

```
int i = 0;
while (i < 6) {
      cout << i * 10 << endl;
      i++;
}
```

Pick whichever one is easiest for the task at hand!

BREAKING OUT OF LOOPS

Another way to break out of a loop is to use the `break` keyword. This works for both "for" and "while" loops. Here is an example of how to use it:

```
while (!gameOver) {
    /* fight enemies! */
    if (health <= 0) {
        break; // this will break out of the loop
    }
}
cout << "Game over!" << endl;
```

As soon as your health drops to 0 or below, the loop will end, and "Game over!" will be printed.

Exercise 5.1: Your first "for" loop

Write a program that uses a "for" loop to print out all even numbers from -50 to 50, one number per line, WITHOUT using any conditionals to check if a number is even or odd.

Exercise 5.2: Fiddling with the "step" part of a "for" loop

Copy your solution to Exercise 5.1 to a new Eclipse project and change it so that the numbers are printed backwards, from 50 to -50.

Exercise 5.3: Putting it all together – using "for" loops

Write a program that "thinks of" (randomly picks) a number and asks you to guess that number in three tries, then tells you if you won or not. **Hints:**

- How do we pick a random number? For this, there is a `rand()` function in the library `cstdlib`. Here is a minimal program demonstrating it:

```
#include <iostream>
using namespace std;
#include <cstdlib>

int main() {
	// Seed the random number generator with system time.
	// This must be done so that we get different random
	// numbers every time the program runs.
	srand (time (NULL));
	// We will only get random numbers up to, but NOT
	// including, "maxNum".
	int maxNum = 6;
	// The rand() function returns a number between 0 and
	// some very high value.  The % operator divides that
	// by "maxNum" and returns the remainder.  The remainder
	// can never be equal to or higher than "maxNum", so
	// "randomNum" will always be between 0 (inclusive) and
	// maxNum (not inclusive).
	int randomNum = rand() % maxNum;
	cout << randomNum << endl;
	return 0;
}
```

- Use a "for" loop to loop over the three tries.
- Use a `break` keyword to break out of the loop when you guess correctly.

Exercise 5.4: Putting it all together – using "while" loops

Copy your solution to Exercise 5.3 to a new Eclipse project, then change the program to use a "while" loop instead of a "for" loop, and also DO NOT use the `break` keyword.