LOOPS

Topics

Topics

- Four Types of Loops

Topics

- Four Types of Loops
  - `while`

Topics

- **Four Types of Loops**
  - `while`
  - `do…while`

Topics

- **Four Types of Loops**
  - `while`
  - `do…while`
  - `for`

Topics

- **Four Types of Loops**
  - `while`
  - `do…while`
  - `for`
  - `foreach`

Topics

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

Topics

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

Topics

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

Four Types of Loops

Four Types of Loops

- `while`

Four Types of Loops

- `while`
  - The most basic loop

Four Types of Loops

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

Four Types of Loops

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

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

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`

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

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

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`

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

Four Types of Loops

Four Types of Loops

- `while`

Four Types of Loops

- `while`
  - The most basic loop

Four Types of Loops

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

Four Types of Loops

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

```
while (true) {
    print( "Loop" );
}
```

Four Types of Loops

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

```
while (true) {
    print( "Loop" );
}
```

  - This will cause an *infinite loop*!!!

Four Types of Loops

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

```
while (true) {
    print( "Loop" );
}
```

  - This will cause an *infinite loop*!!!
  - "Loop" will never appear in the Console pane because the entire Unity process will be frozen

Four Types of Loops

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

```
while (true) {
    print( "Loop" );
}
```

  - This will cause an *infinite loop*!!!
  - "Loop" will never appear in the Console pane because the entire Unity process will be frozen
  - This would necessitate *force quitting* Unity

Four Types of Loops

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

```
while (true) {
    print( "Loop" );
}
```

  - This will cause an *infinite loop*!!!
  - "Loop" will never appear in the Console pane because the entire Unity process will be frozen
  - This would necessitate *force quitting* Unity
  - On old, single-threaded computers, this would require turning the computer off!

Four Types of Loops

Four Types of Loops

- `while` – A better `while` loop

Four Types of Loops

- `while` – A better `while` loop
  - `while` loops need an exit condition

Four Types of Loops

- **`while` – A better `while` loop**
  - **`while` loops need an exit condition**
    - A condition that will cause the condition to evaluate to false

Four Types of Loops

- **`while` – 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**

Four Types of Loops

- **`while` – 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
}
```

Four Types of Loops

- **`while` – 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**

Four Types of Loops

- **`while` – 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**

Four Types of Loops

Four Types of Loops

- `do…while`

Four Types of Loops

- `do…while`
  - Like a `while` loop, but checks *after* the loop has run

Four Types of Loops

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

Four Types of Loops

- **`do…while`**
  - **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**

Four Types of Loops

- **`do…while`**
  - **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);
```

Four Types of Loops

- **`do…while`**
  - **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**

Four Types of Loops

- **`do…while`**
  - **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**

Four Types of Loops

Four Types of Loops

- `for`

Four Types of Loops

- `for`
  - A `for` loop contains three separate clauses

Four Types of Loops

- `for`
  - A `for` loop contains three separate clauses

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

Four Types of Loops

- `for`
  - A `for` loop contains three separate clauses

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

  - Initialization clause: `int i=0;`

Four Types of Loops

- `for`
  - 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;`

Four Types of Loops

- `for`
  - 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++`

Four Types of Loops

- `for`
  - 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

Four Types of Loops

- `for`
  - 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

Four Types of Loops

- `for`
  - 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 ++

Four Types of Loops

- `for`
  - 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

Four Types of Loops

Four Types of Loops

- `foreach`

Four Types of Loops

- `foreach`
  - Automatically loops for each element in a collection

Four Types of Loops

- **foreach**
  - Automatically loops for each element in a collection

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

Four Types of Loops

- `foreach`
  - 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

Four Types of Loops

- `foreach`
  - 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 Within Loops

Jump Statements Within Loops

- Jump statements change the execution of a loop

Jump Statements Within Loops

- Jump statements change the execution of a loop
  - `break`

Jump Statements Within Loops

- **Jump statements change the execution of a loop**
  - **`break`**
    - Breaks out of the loop entirely

Jump Statements Within Loops

- **Jump statements change the execution of a loop**
  - **`break`**
    - Breaks out of the loop entirely
  - **`continue`**

Jump Statements Within Loops

- **Jump statements change the execution of a loop**
  - **`break`**
    - Breaks out of the loop entirely
  - **`continue`**
    - Breaks out of this iteration of the loop and moves on to the next

Jump Statements Within Loops

Jump Statements Within Loops

- `break`

Jump Statements Within Loops

- `break`
  - Breaks out of the loop completely

Jump Statements Within Loops

- **`break`**
  - Breaks out of the loop completely

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

Jump Statements Within Loops

- **`break`**
  - Breaks out of the loop completely

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

Jump Statements Within Loops

- **break**
  - Breaks out of the loop completely

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

Jump Statements Within Loops

- **break**
  - Breaks out of the loop completely

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

  - This will print:

Jump Statements Within Loops

- **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
```

Jump Statements Within Loops

- `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

Jump Statements Within Loops

- `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

Jump Statements Within Loops

Jump Statements Within Loops

- `continue`

Jump Statements Within Loops

- `continue`
  - Breaks out of the loop completely

Jump Statements Within Loops

- `continue`
  - Breaks out of the loop completely

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

Jump Statements Within Loops

- `continue`
  - Breaks out of the loop completely

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

Jump Statements Within Loops

- `continue`
  - Breaks out of the loop completely

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

Jump Statements Within Loops

- `continue`
  - Breaks out of the loop completely

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

  - This will print:

Jump Statements Within Loops

- `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
```

Jump Statements Within Loops

- `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

Chapter 21 – Summary

Chapter 21 – Summary

- Of the four types of loops:

Chapter 21 – Summary

- **Of the four types of loops:**
  - **`while` and `do…while` are somewhat dangerous**

Chapter 21 – Summary

- Of the four types of loops:
  - `while` and `do…while` are somewhat dangerous
  - `for` is by far the most common and is very flexible

Chapter 21 – Summary

- **Of the four types of loops:**
  - **`while` and `do…while` are somewhat dangerous**
  - **`for` is by far the most common and is very flexible**
  - **`foreach` is very useful for strings, arrays, and Lists**

Chapter 21 – Summary

- **Of the four types of loops:**
  - **`while` and `do…while` are somewhat dangerous**
  - **`for` is by far the most common and is very flexible**
  - **`foreach` is very useful for strings, arrays, and Lists**
    - We'll talk about it a lot more in Chapter 22

Chapter 21 – Summary

- **Of the four types of loops:**
  - **`while` and `do…while` are somewhat dangerous**
  - **`for` is by far the most common and is very flexible**
  - **`foreach` is very useful for strings, arrays, and Lists**
    - We'll talk about it a lot more in Chapter 22
- **Jump statements can be used to have more control over your loops**

Chapter 21 – Summary

- **Of the four types of loops:**
  - **`while` and `do…while` are somewhat dangerous**
  - **`for` is by far the most common and is very flexible**
  - **`foreach` is very useful for strings, arrays, and Lists**
    - We'll talk about it a lot more in Chapter 22
- **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**

Chapter 21 – Summary

- **Of the four types of loops:**
  - **`while` and `do…while` are somewhat dangerous**
  - **`for` is by far the most common and is very flexible**
  - **`foreach` is very useful for strings, arrays, and Lists**
    - We'll talk about it a lot more in Chapter 22
- **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**
- **Chapter 22 will cover arrays and Lists, two kinds of collections in C#**