Using Loops in Visual Basic

Basic Loop Concepts:

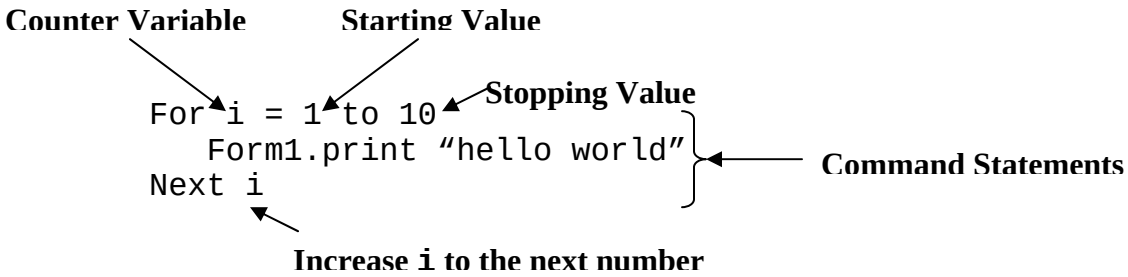
Loops are programming structures that repeat the same commands over and over until told to stop. There are 3 basic types of loops used in programming VB.

I. For-Next
II. While-Wend
III. Do-Loop Until

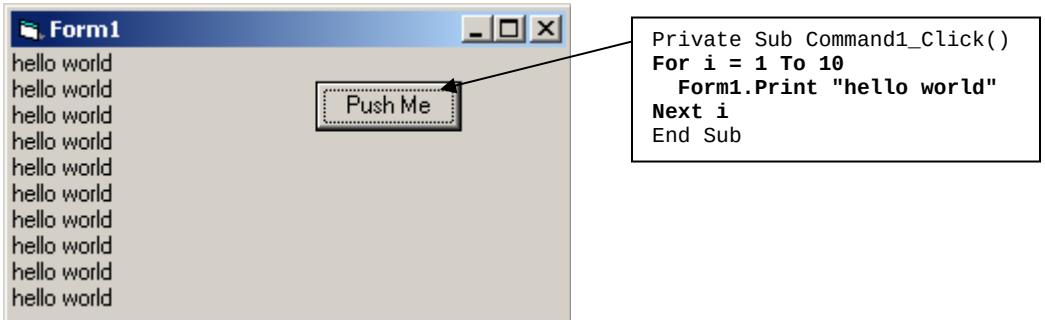
Let's examine them one at a time:

For-Next loop

These loops are designed for COUNTING. If you need the same thing done a known amount of times, like 1 to 10, -5 to 5, or 0 to 1000, then For-Next loops should be used. Now, examine an easy example of a For-Next loop.



The above code simply writes hello world to the form 10 times.



Now, that you are excited, let's examine the loop structure by looking at the diagram above.

1. **The Counter Variable:** this variable is usually an integer or long type, whose only purpose is count from a starting value to a stopping value.
2. **The Starting Value:** the value to begin counting from.
3. **The Stopping Value:** the value to end the looping process.
4. **The Command Statements:** the stuff you need to have repeated by the loop.
5. **The Next:** instructs VB to change the counter variable to the next number.

In the above example the loop variable, i, is assigned the value 1 (the first number) and then VB runs the command to print hello world to the form. After printing to the form the i is increased by 1 to 2 with next i statement. This process of printing to the form and increasing i's value continues until i reaches 10. Remember, i is a variable and variables are good for one thing, storing information! So, VB is increasing the value of i by 1 and storing it back into I every time the loop completes a cycle. Look the code below. What do you think it does?

```
For I = 1 to 10
   Form1.print I
Next i
```

Prints the numbers 1 thru 10 on the form. That's because VB is re-assigning i the next value in the count.

MORE INFO:

Can VB only count by 1's? NO, VB can count by 2's, 3's or what ever you want, just as long as your counting by integer numbers (NO COUNTING BY DECIMALS). VB can even count backwards (10,9,8,7,…) So, how is it done? Just add the Step command in and tell it how you want it to count. Look at the examples:

Count bv 2's

```
For I = 1 to 10 Step 2
   Form1.print I
Next i
```

Count backwards by 1

```
For I = 1 to 10 Step -1
   Form1.print I
Next i
```

For –Next loops are made for counting.
They are really useful while working with arrays!?!

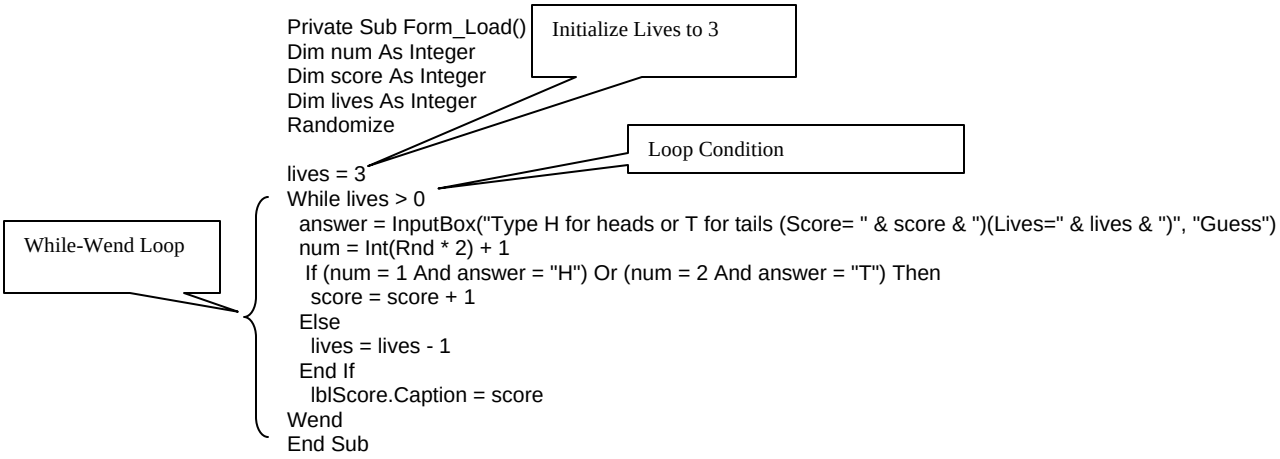
General Format for a FOR-NEXT Loops

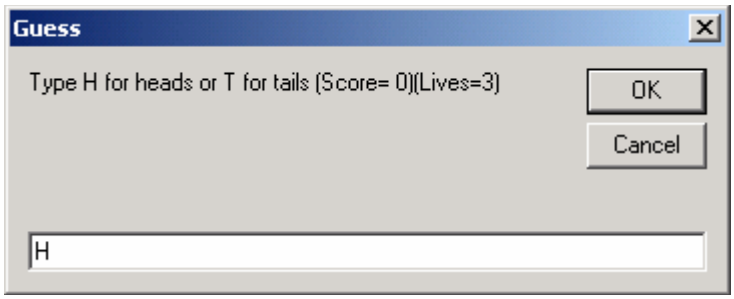
```
For <countvariable> = <startvariable> To <stopvariable>
  <command statements>

Next <countvariable>
```

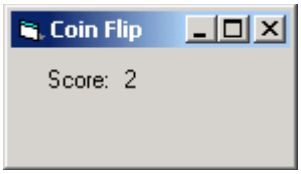
While-Wend Loops

While-Wend Loops are CONDITIONAL loops. They repeat the same commands over and over while a condition is true and stop repeating when a condition is false. Unlike, For-Next loops that stop at a number, While-Wend loops wait for a condition (Boolean

expression) to become false to stop repeating. An example of a conditional loop is a simple video game. The player is allowed to play the game while he still has lives. When the player looses all of his lives the game is over. While you still have lives – keep playing. Let's look at an example:







In the above example, the program starts and the user enters H or T. If the player guesses wrong then one life is taken. If he is correct then one is added to his score. The game stops when the While-Wend loop's condition of lives>0 is not met. In other words, keep playing while you still have lives.

MORE INFO:

While – Wend loops test their conditions at the TOP. If you don't pass the condition, then you don't get into the loop.

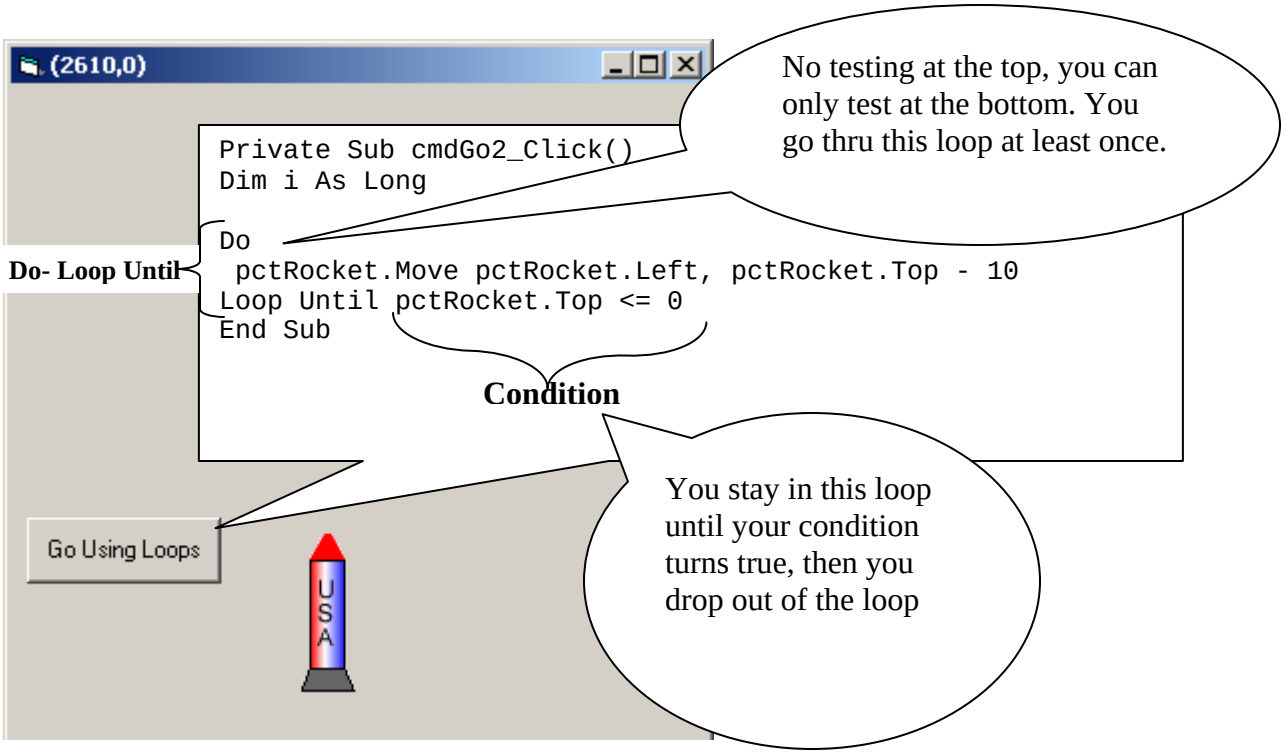
General format for a WHILE-WEND loop

```
While <condition>
  <command statements>

Wend
```

Do-Loop Until

Like the While-Wend loop, the Do-Loop Until is also a conditional loop. The difference between the two is the Do-Loop Until has its condition at the end of the loop, not at the beginning like the While-Wend. Having the condition at the BOTTOM of the loop means you are guaranteed to go thru the loop ONCE. Whereas the While – Wend you are NOT guaranteed to go thru the loop, because the condition is at the TOP of the loop. A classic example of this is rocket example. A rocket starts from the bottom of the form. The rocket moves upward until it reaches the top of the form. I know when I hit the GO button to make the rocket move. So, there is no reason to test before the rocket moves. Let the rocket move and test it's position at the bottom.

Take a look at the program:



MORE INFO:

Do-Loop Until loops test at the bottom and drop out of the loop when the condition is true. Whereas a While-Wend loop tests it's condition at the top of the loop and drops out of the loop when the condition turns false. Use the Do-Loop Until loop if you know the loop needs to be ran at least once.

General format for a Do-Loop Until

```
Do
 <command statements>
Loop Until <condition>
```