**Second Tutorial**

This tutorial assumes Tutorial 1 has been reviewed by the reader. In this tutorial we will make use of some of the more interesting features included in the Next language. The program that follows will utilize a probability statement to non-deterministically execute segments of code and we will use a choose statement to take simple keyboard input from the user and execute commands accordingly. The commands will make an item show or hide(disappear) from a location and make a character drop or grab an item from a location if the conditions are right or output the automatic error message if it is not possible to perform the action.

We will begin by creating an integer, a location and a character. Because no value will be given to the integer it will take the Next default value of 0. The location and character declarations will be created with empty lists which means that they will not have attributes, items or in the case of the location, characters. Those declarations look as follows:

int num;

character person {(),()}

location here {(),(),()}

We will then declare an item. Items can only have attributes and in this case we will also leave this item’s attribute list empty. The declaration looks as follows:

item object {()}

Next we will declare a start statement, very much like we did in the last example:

start here end (num == 1) {

Inside this start declaration we will begin by putting a probability statement. Probability statements begin with “[?” which in Next is known as an opening probability bracket. Probability statements end with a closing probability bracket which looks like this “?]”. Inside the probability statement code that is executed with a given probability is defined with the notation *prob number statement* where *number* is an integer that specifies the probability that the *statement* code will be executed. The probability statement can have many such *prob number statement* clusters. One of them is chosen from inside the probability statement according to the *number* part which specifies the probability of that cluster. It is important to remember that the sum of all the integers defined by the *number* part of the cluster must equal 100. The probability statement used for this program looks as follows:

[?

prob 50 output "This is line 1 of a possible 2";

prob 50 output "This is line 2 of a possible 2";

?]

This probability statement means that with a 50 percent probability the line "This is line 1 of a possible 2" will be output. If that line is not output then "This is line 2 of a possible 2" will be output (which, of course, means the second line also has a 50 percent probability of occurring as specified in the probability statement).

Following the probability statement we use two if statements. The first one checks if the item **object** exists in the **person** character. It outputs a message that notifies if it does or not. The second if statement checks if the item **object** exists in location **here** and again outputs a message that notifies the result. The code looks like this:

if (exists person.object) then

output "The person has the object";

else

output "The person does not have the object";

if(exists here.object) then

output "The object is in the location";

else

output "The object is not in the location";

These if statements become useful to easily see the results of the choose statement that follows. A choose statement contains lists of three elements that specify a variable name, a string that will assist in output and a letter that becomes mapped to the keyboard. The first list of the choose statement in this program looks as follows:

(grabItem, "character grab item", "g")

This means that when the choose statement is executed the output “Press g for character grab item” will be presented to the user. The g keyboard key will be mapped and if the user presses the g key followed by the enter key the code associated with the **grabItem** variable will be executed. These associations occur through the use of “when” statements which will soon be explained. An arbitrary amount of lists for action/keyboard mappings may be used in a choose statement. The choose statement and its lists used in this example are as follows:

choose (grabItem, "character grab item", "g")

(dropItem, "character drop item", "d")

(showItem, "show item in location", "s")

(hideItem, "hide item from location", "h")

(exit, "exit", "e")

The lists for a choose statement for action/keyboard bindings are followed by a series of “when” statements enclosed by braces. These “when” statements specify code to be executed for each action. At the end of the code specified for each “when” statement a “next” statement is used that specifies a location to be executed next. The first when statement in this example is:

when grabItem

grab person.object;

next here

This means that when the **grabItem** action is specified (in this example by the user choosing the g key on the keyboard) the character **person** will grab the item **object** if **object** is present in the current location. This means that if the item **object** is present in the location “grab” will cause **object** to be removed from the location and added to the list of the character’s **items.** If the item is not present in the location when grab is used an error message is output and execution continues with no changes to the character or the location. Once “grab” is executed in the above “when” statement the next statement sends execution to the beginning of the **here** location.

The when statement that follows in our example is:

when dropItem

drop person.object;

next here

In this case the “drop” action is used to have the **person** character drop the **object** item in the current location. “drop” removes an item from a character’s item list if the character has that item and adds it to a location’s item list. If the character does not have that item an error message is output and execution continues.

After that the following two when statements follow:

when showItem

show here.object;

next here

when hideItem

hide here.object;

next here

“show” and “hide” statements are used to add or remove an item or character from a location. In this case the item **object** is added to the **here** location by using the “show” keyword or removed by using the “hide” keyword. “show”, “hide”, “drop” and “grab” are used in conjunction with “exists” to create connections between items, locations and characters in Next programs to simulate real world relationships when creating games.

Finally the last “when” statement is used to end execution of location **here** by making its end condition true. The entire program can be seen bellow.

int num;

character person {(),()}

location here {(),(),()}

item object {()}

start here end (num == 1) {

[?

prob 50 output "This is line 1 of a possible 2";

prob 50 output "This is line 2 of a possible 2";

?]

if (exists person.object) then

output "The person has the object";

else

output "The person does not have the object";

if(exists here.object) then

output "The object is in the location";

else

output "The object is not in the location";

choose (grabItem, "character grab item", "g")

(dropItem, "character drop item", "d")

(showItem, "show item in location", "s")

(hideItem, "hide item from location", "h")

(exit, "exit", "e")

{

when grabItem

grab person.object;

next here

when dropItem

drop person.object;

next here

when showItem

show here.object;

next here

when hideItem

hide here.object;

next here

when exit

num = 1;

next here

}

}