FOREACH*() FUNCTIONALITY

Exercise Files
Starter – n/a

Answers
"gpgt/engine/answers/chapter11/forEach.cc"

Exercise Mission
Chapter 11: "007_GameClasses: ForEach"

Special Setup
If you have not already done so, please install the the engine SDK (with source code) and make a directory under the "SDK/engine" directory. Name it anything you like. I added a directory and a subdirectory named "engine/EngineCodingExercises/chapter11/". Now, as you add files to this directory, please add them to your build files too.

Synopsis

This is a difficult exercise, designed to test your ability to put together several topics that have been discussed in this chapter to add a useful feature to the Torque engine.

The useful feature you will be adding is the ability to execute the same (named) function on every object in a SimSet by making a single console method call on the SimSet. This is much like the forEach() feature found in PERL, except that it has been adapted to operate on the contents of SimSets in TorqueScript.

Prerequisites
1. *ch1_001.pdf "Using The Kit"*
2. *ch10_001.pdf "Compiling Torque in Windows" and/or ch10_002.pdf "Compiling Torque in OSX"*

Exercises

FOREACH*() FUNCTIONALITY

1 forEachSimple()

Goal: In C++, add a new console method to the SimSet class named "forEachSimple()". This method is specified as follows.

- Takes a maximum of two extra arguments, and a minimum of one:
  - method – The unadorned name of the method to be executed on every object in the SimSet.
  - debug – An optional boolean value that specifies whether the method should print out debug messages. By default, debug is false.
- Returns void.

Starter Code:

To help you along, here is the code for iterating over a SimSet in C++.

```
SimSet *mySet;

// at some time mySet is populated with a valid pointer to
// a SimSet.


for (SimSetIterator obj( mySet ); *obj; ++obj)
{
   SimObject* nobj = dynamic_cast<SimObject*>(*obj);

   if (nobj)
   {
      // Do something with nobj, which is an object in the SimSet
   }
}
```

Additional Notes:

In this exercise, when writing this new method please use the Con::evaluate() to call method on every object in the SimSet. How you do this is up to you.

FOREACH*() FUNCTIONALITY

Hints:

1. Con::evaluate() provides a debug output feature.
2. If you want, you can simply create a single .cc file and implement a new console method in it. You don't need to modify the SimSet files themselves.
   - For example, you can create a file named "forEach.cc" containing the code below and then add that file to your project. Then, you can use this new file as a starting point for adding new console functionality to the SimSet class. (This isolates your code from the original files in the SDK, making it easy to migrate them later.)

```
#include "console/consoleTypes.h"
#include "console/console.h"
#include "core/bitStream.h"

#include "console/ConsoleObject.h"
#include "console/simBase.h"

// Add your new console method declaration here.
```

Testing Your Code

For testing purposes, I have included a script named "testforEachSimple".

```
function testforEachSimple()
{
   new SimSet( testSet );

   testSet.add( new SimObject() );
   testSet.add( new SimObject() );
   testSet.add( new SimObject() );

   testSet.forEachSimple( doIt );

   testSet.deleteSet( true );
}
```

This script is loaded when you load the mission associated with this exercise. Then, assuming you have properly added the console method, compiled it, and are now running the newly compiled executable, you will see the following when you run this script.

```
==>testforEachSimple();
Called doIt() on simObject 1982
Called doIt() on simObject 1983
Called doIt() on simObject 1984
```

This testing code has created three objects, placed them in a SimSet, and then called the method doIt() on them.

2 forEachAdvanced()

Goal: Improve on the implementation of forEachSimple() and add the ability to pass in any number of arguments that should be passed as part of the method call. i.e. Call a method on every object in a SimSet and pass them each the same set of arguments.

This is a pretty advanced exercise and you shouldn't feel bad if you need to look at the answer for help as you work on it.

Testing Your Code

For testing purposes, I have included the following scripts in the kit.

```
function testforEachAdvanced()
{
   new SimSet( testSet );

   testSet.add( new SimObject() );
   testSet.add( new SimObject() );
   testSet.add( new SimObject() );

   testSet.forEachAdvanced( doIt2, true );

   testSet.deleteSet( true );
}

function testforEachAdvanced2()
{
   new SimSet( testSet );

   testSet.add( new SimObject() );
   testSet.add( new SimObject() );
   testSet.add( new SimObject() );

   testSet.forEachAdvanced( doIt2, true , 1 , 2 , 3 );

   testSet.deleteSet( true );
}

function simObject::doIt( %obj )
{
   echo("Called doIt() on simObject" SPC %obj );
}

function simObject::doIt2( %obj, %arg0, %arg1, %arg2 )
{
   echo("Called doIt2() on simObject" SPC %obj SPC "with args:" SPC
        %arg0 SPC %arg1 SPC %arg2 );
}
```


FOREACH*() FUNCTIONALITY

These scripts are loaded when you load the mission associated with this exercise. Then, assuming you have properly added the console method, compiled it, and are now running the newly compiled executable, you will see the following when you run this script.

```
==>testforEachAdvanced();
% 1645.doIt2();
Called doIt2() on simObject 1645 with args:
% 1646.doIt2();
Called doIt2() on simObject 1646 with args:
% 1647.doIt2();
Called doIt2() on simObject 1647 with args:
```

This test calls a method named doit2() and enables debug but does not pass extra arguments.

To see the effect of passing extra arguments try running "testforEachAdvanced2()". You should output like the following.

```
==>testforEachAdvanced2();
% 1649.doIt2(1,2,3);
Called doIt2() on simObject 1649 with args: 1 2 3
% 1650.doIt2(1,2,3);
Called doIt2() on simObject 1650 with args: 1 2 3
% 1651.doIt2(1,2,3);
Called doIt2() on simObject 1651 with args: 1 2 3
```

This test calls a method named doit2(), enables debug, and passes three extra arguments.

