CH12_004

EXERCISE

THE TORQUE MEMORY MANAGER

Exercise Files

Starter – "Kit/gpgt/server/scripts/gpgt/chapter12/exercise004.cs"
Answers – "Kit/gpgt/server/scripts/gpgt/chapter12/answers/exercise004_f.cs"

Exercise Mission

Chapter 12: "004_DebuggingTorque: The TMM"

Special Setup

If you choose, you can use the debug version of the executable that is included with the kit to run this exercise. It already has the TMM enabled. However, if you want to compile it in your yourself, please open the file "engine/core/torqueConfig.h" and locate the line that specifies "TORQUE_DEBUG_GUARD" and be sure that it is uncommented, then re-compile the engine.

Synopsis

In this exercise, we will run some experiments with the Torque Memory Manager (TMM) to improve your understanding of this debug feature.

Prerequisites

1. *ch1_001.pdf "Using The Kit"*

Exercises

THE TORQUE MEMORY MANAGER

1 Setting Up

Goal: Demonstrate that you understand the basics of setting up for a TMM dump and then running the TMM dump itself.

Starter Code: In this part of the exercise, you need to focus on three functions, startExercise004(), doPrepWork(), and doTMMDump().

```
function startexercise004()
{
   $prepFile = expandFilename( "~/baseLineTMMDump" );
   $TMMFile = expandFilename( "~/debugTMMDump" );
}

function doPrepWork()
{
   // ?????
   // ?????
}

function doTMMDump()
{
   // ?????
}
```

You will notice that the exercise starter (startexercise004()) prepares two globals, $prefFile, and $TMMFile, each containing a file name. The first is the file we will store our pre-work dump in. The second, is the file we will store the post-work dump in.

Steps:

1. Please update the doPrepWork() function so that it flags all current allocations and then prints any unflagged allocations to the $prepFile.
2. Please update the doTMMDump() function so it dumps any unflagged allocations to the $TMMFile.

Once you have completed this work, please move on to the next part of the exercise.

2 Evaluating Dumps

Goal: Learn to diagnose TMM problems.

Starter Code: Normally, we would make TMM dumps to check for leaks, but since we are doing this as an exercise we will have to fake leaks.

In this part of the exercise, we will be running the following functions:

- doPrepWork() - This function, which you completed above, flags all current allocations and then creates a baseline dump.
- doTMMDump() - This function, which you also completed above, will do another TMM dump, but to a second file, allowing us to compare that file against the baseline.
- printDumps() - This function, provided by me, will print out the contents of the pre-work file and the post-work file. This way we can evaluate the results of our tests in the console.
- dummyWork() - This function will allocate some memory and create a SimObject.
- dummyCleanup() - This function deletes the SimObject created in dummyWork(), thus freeing the memory we allocated.
- tmmTest0() - This function will call the functions in this order:
  - doPrepWork()
  - dummyWork()
  - dummyCleanup()
  - doTMMDump()
  - printDumps()
- tmmTest1() - This function will call the functions in this order:
  - doPrepWork()
  - dummyWork()
  - doTMMDump()
  - printDumps()

Please notice, that tmmTest1(), does not delete the SimObject. This is how we will simulate a memory leak in this exercise.

THE TORQUE MEMORY MANAGER

Steps:

At this time, please run tmmTest0() and examine the output.

```
==>tmmTest0();
****** PRE WORK ************


****** POST WORK *********
h:\00_currentwip\bk002_gpgt2\tge_1_5_2\engine\console\consoleinternal.cc^483^24^74565
h:\00_currentwip\bk002_gpgt2\tge_1_5_2\engine\console\consoleinternal.cc^297^16^74566
h:\00_currentwip\bk002_gpgt2\tge_1_5_2\engine\console\consoleinternal.cc^301^60^74567
```

You may be surprised to find that there are some lines in the post-work file. However, let me assure you that this is normal. Because we are using the console, we can expect to see some allocations occurring in the consoleinternal.cc file. These allocations will eventually get cleaned up, but not right away.

Now, try running the second test, tmmTest1().

```
==>tmmTest1();
****** PRE WORK ************


****** POST WORK *********
h:\00_currentwip\bk002_gpgt2\tge_1_5_2\engine\console\consoleinternal.cc^297^16^64749
h:\00_currentwip\bk002_gpgt2\tge_1_5_2\engine\console\consoleinternal.cc^301^60^64750
h:\00_currentwip\bk002_gpgt2\tge_1_5_2\engine\console\consoleinternal.cc^483^24^64748
h:\00_currentwip\bk002_gpgt2\tge_1_5_2\engine\console\consoleobject.h^366^56^64747
```

Questions:

1. Does it look like there might be a leak?
2. If so, please identify the source of this leak, including file, line, and allocation size.

