

Torque Game Builder – File I/O Tutorial

Introduction

Torque Game Builder (TGB) gives users the ability to easily read and write their own custom files using FileObjects. In this tutorial we will dynamically write a file through code and then read the contents of that file into our console. For more information about file IO in TGB, please refer to the *TGB Reference* document that comes with the engine.

Writing a file

To begin with, we are going to create a new .cs file in our games/TGB/gameScripts folder. Name this file “*FileIO.cs*”. We are creating this file using the TGB project because it is a default project that all Torque Game Builder users should have. The first thing we are going to do in our script file is create a simple function that will write a text document for us. At the top of our *FileIO.cs* file, add our first function:

```
function writeFile()
{
    %file = new FileObject();

    if(%file.openForWrite("~/data/files/test.txt"))
    {
        %file.writeLine("Hello World!");
        echo("File Written");
    }
    else
    {
        error("File is not open for writing");
    }
    %file.close();
    %file.delete();
}
```

This function is pretty straightforward: it creates a new object (%file) of the type FileObject. Next, it checks to see if the file we are going to write to is open for writing (in this case, the file does not exist yet, so our code will create it). If the file is open for writing, it writes the line “Hello World” in the file. Next, we tell the code to give us a confirmation of the write in the console. If the openForWrite check failed, the console will receive an error message relating that fact to the user. The last two lines are cleanup: the code closes the file we just wrote to and deletes the file object we created at the beginning of the function since we no longer need it.

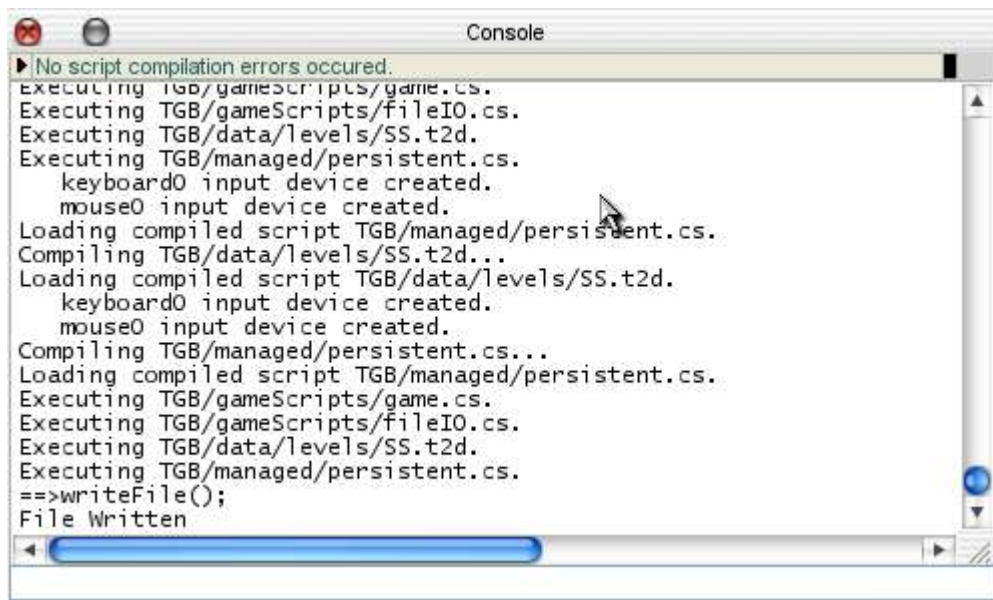
We should test this code now. Make sure that you add an exec call in the game.cs file so that your new code will be loaded. Once you have your code finished, it is time to open TGB and test it.

Torque Game Builder – File I/O Tutorial

Testing Our File

Open up your copy of TGB, either by clicking on the desktop icon or finding it in the start menu. Once you have it open, we will need to open the *TGB* project (if it isn't already open) by clicking on *Open Project* in the file menu. Also, we will need to create a new level by selecting *New Level* in the file menu. Once your new level is created, save it with any name, and choose *Run Game* in the project menu. Once you are in the level, bring up the console by hitting the tilde key (~).

We are going to quickly test the functionality of the code we just wrote. Type “*writeFile();*” into the console window and hit enter. At once, we should get an echo statement confirming that our file was written (figure 1).



```
Console
▶ No script compilation errors occurred.
Executing TGB/gameScripts/game.cs.
Executing TGB/gameScripts/fileIO.cs.
Executing TGB/data/levels/SS.t2d.
Executing TGB/managed/persistent.cs.
  keyboard0 input device created.
  mouse0 input device created.
Loading compiled script TGB/managed/persistent.cs.
Compiling TGB/data/levels/SS.t2d...
Loading compiled script TGB/data/levels/SS.t2d.
  keyboard0 input device created.
  mouse0 input device created.
Compiling TGB/managed/persistent.cs...
Loading compiled script TGB/managed/persistent.cs.
Executing TGB/game.cs.
Executing TGB/gameScripts/fileIO.cs.
Executing TGB/data/levels/SS.t2d.
Executing TGB/managed/persistent.cs.
==>writeFile();
File Written
```

Figure 1

If you did not receive the confirmation “*File Written*”, make sure to check your code and exec statement for accuracy. Now that we have our file written, we should go check to see where it is. According to our code, we created the file in `~/data/files/test.txt` (the tilde stands for the relative path of games/TGB). Navigate to that path, and you should see your newly created file sitting in your newly created files folder. Open up your test.txt document and you should see the line “*Hello World!*” all alone at the top of the file (figure 2).

Torque Game Builder – File I/O Tutorial

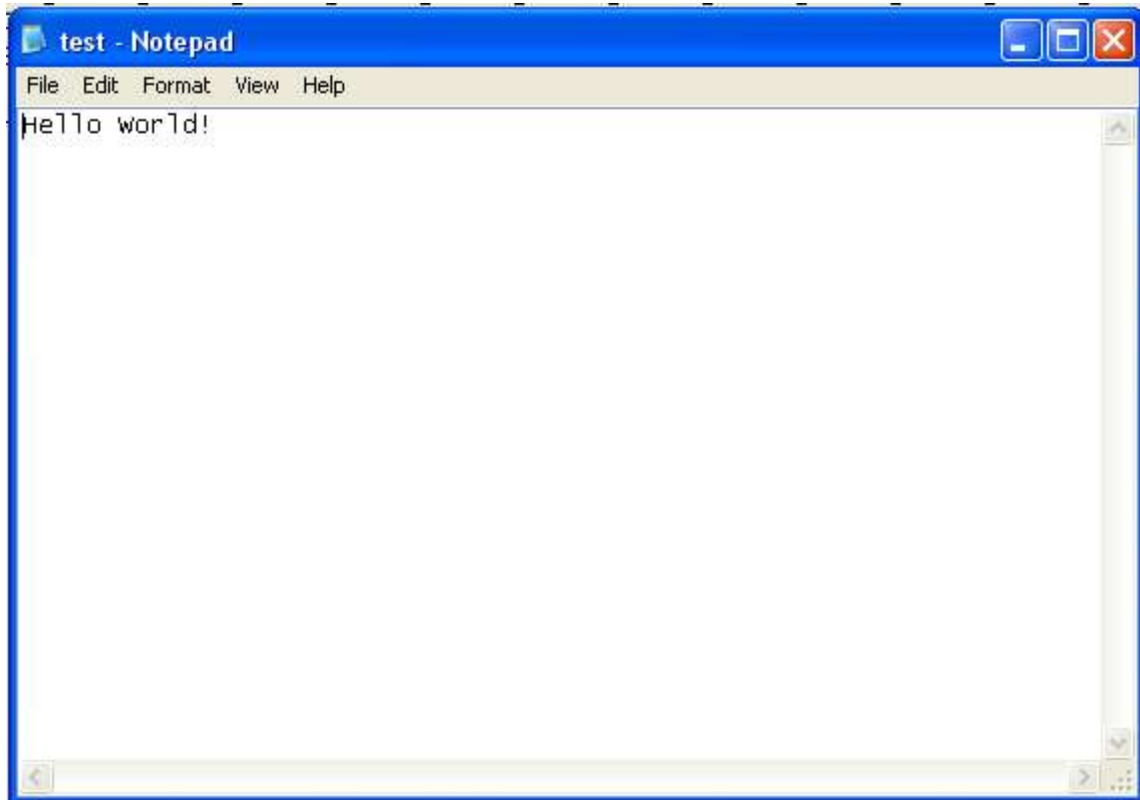


Figure 2

Now that we have seen how the write system works, let's see how our read system works.

Torque Game Builder – File I/O Tutorial

Reading in a File

While you have your “*test.txt*” file open, write a few lines into the document (figure 3).

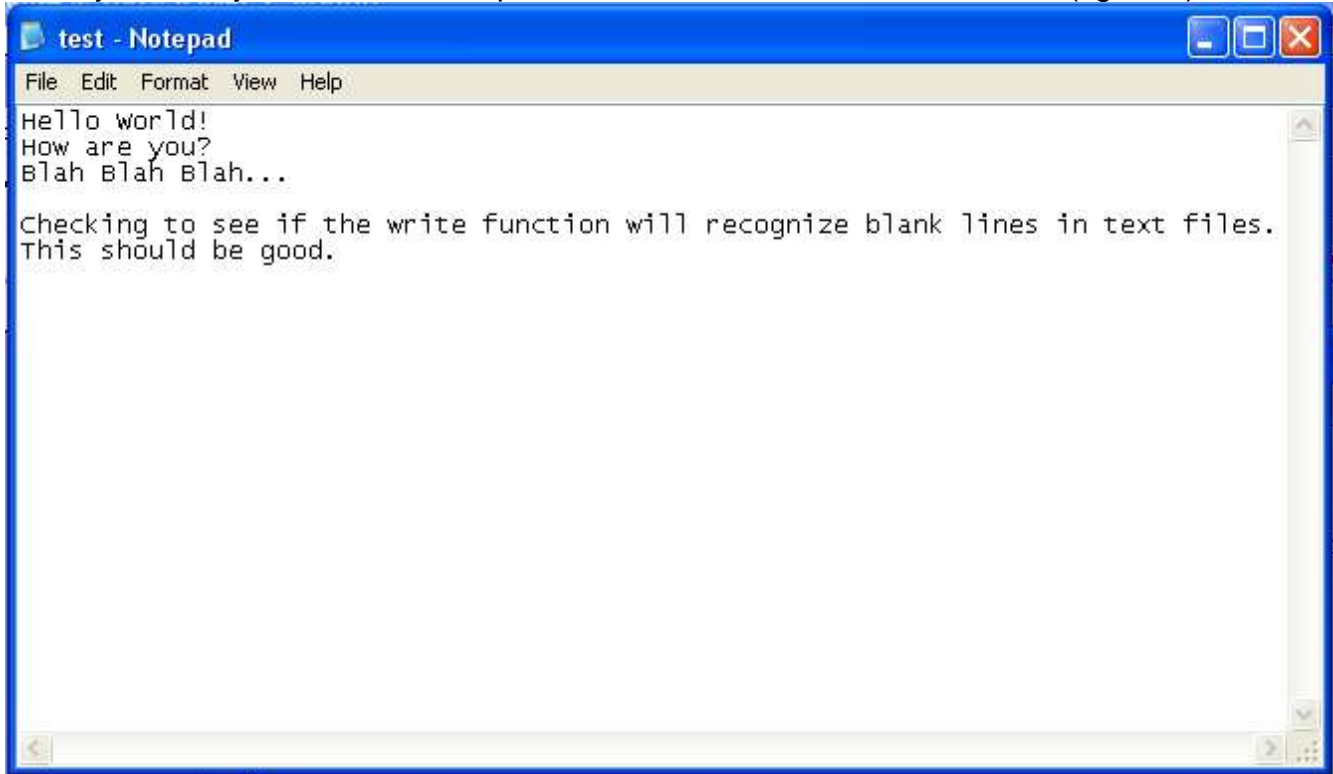


Figure 3

When you finish writing in your lines, save your file and navigate to your TGB/gameScripts folder and access your *fileIO.cs* file. Inside this file add the following function right below the one we created earlier:

Torque Game Builder – File I/O Tutorial

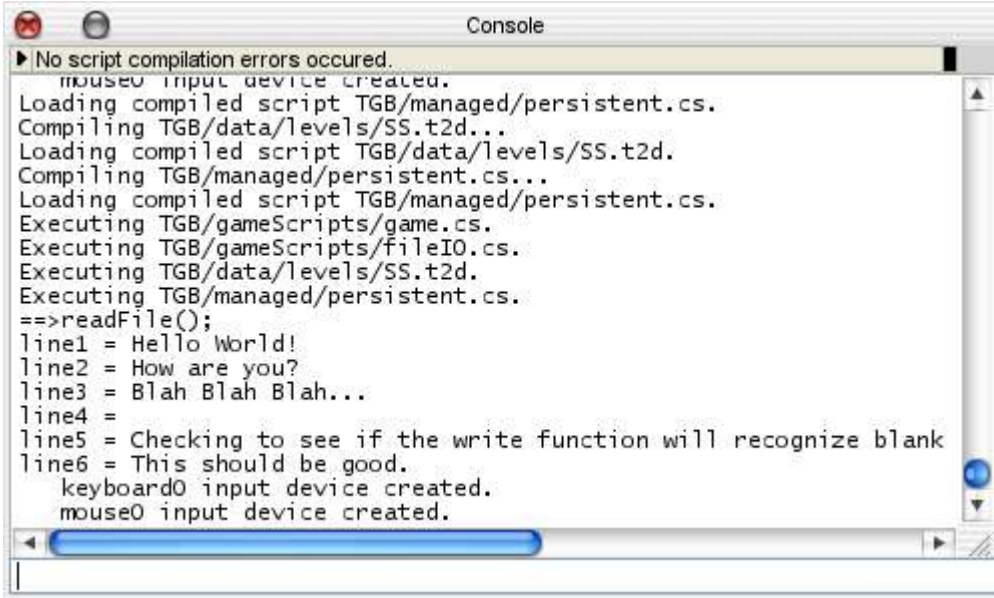
```
function readFile()
{
    %file = new FileObject();

    if(%file.openForRead("~/data/files/test.txt"))
    {
        %x=1;
        while(!%file.isEof())
        {
            %line = %file.readLine();
            echo("line" @ %x @ " = " @ %line);
            %x++;
        }
    }
    else
    {
        error("CANNOT OPEN FOR READ");
    }
    %file.close();
    %file.delete();
}
```

This code works somewhat similar to the last function. First it creates a new FileObject. Next, it checks to see if the file we are trying to read is open for reading. If it is, we start a while loop that runs as long as it is not the end of file, assigns the current line being read to a variable %line and spits it to our console. If the file is not open for read, the console will get an error message informs the user of that. Once you have this code in, save your file and go back into TGB.

When you get back into TGB, test your level again and open the console. When you console is open, type “*readFile()*,” into it and hit enter. When you do, you should see the something like figure 4 in your console:

Torque Game Builder – File I/O Tutorial



```
Console
▶ No script compilation errors occurred.
  mouse0 input device created.
Loading compiled script TGB/managed/persistent.cs.
Compiling TGB/data/levels/SS.t2d...
Loading compiled script TGB/data/levels/SS.t2d.
Compiling TGB/managed/persistent.cs...
Loading compiled script TGB/managed/persistent.cs.
Executing TGB/gameScripts/game.cs.
Executing TGB/gameScripts/fileIO.cs.
Executing TGB/data/levels/SS.t2d.
Executing TGB/managed/persistent.cs.
==>readFile();
line1 = Hello World!
line2 = How are you?
line3 = Blah Blah Blah...
line4 =
line5 = Checking to see if the write function will recognize blank
line6 = This should be good.
  keyboard0 input device created.
  mouse0 input device created.
```

Figure 4

As you can see, the contents that we typed into our file are spit back to us in our console.

Conclusion

With little effort we were able to create a file on the fly and have its contents read back to us in our console. It is not hard to see the potential a system like this would have, given careful implementation. If you require a deeper understanding of the topic discussed here, please refer to the *TGB Reference* file that comes with the engine.