Beginners guide to T3D

Lukas Peter Joergensen

September 25, 2012

Revision 2

Introduction

Welcome to this basic introductional tutorial for Torque3D OS.
This tutorial is a very introductional tutorial for people who know how to program, but are new to Torque. This guide aids to help learning some of the peculiar things in TorqueScript.
It is not a very comprehensive guide, just a guide with the complete basics, and a set of useful links.
If you have any request or ideas for improvements, write to me at LukasPJ@FuzzyVoidStudio.com.

The Author

Who am i?

My name is Lukas Joergensen, I am 19 years old and live in Denmark, currently studying Computer Science at Aarhus University.

What is your experience with T3D?

Currently, my most actual project is the IPS Lite and Pro for T3D. You can read about the IPS Lite on the GitHub page
Or visit my personal website at FuzzyVoidStudio.com

Outline

1. Introduction
2. An introduction to T3D
3. TorqueScript, the concepts
4. Project 1 writing a simple coin collection game
5. Quick guides
6. Troubleshooting
7. Useful links

Introduction to T3D

What is Torque 3D?

Torque3D is an engine, it is not based on graphic elements like Unity thus it is not as easy to get into and understand.
Unity has a lot of drag and drop style world building and object creation, in Torque3D you need to write how the objects is composed.
There is no 'make game button' in Torque3D you need to have an understanding of how to code, if you are an absolute beginner at programming and have never written even a simple program then this might not be the right guide to start with.

Introduction to T3D

What can Torque 3D do out of the box?

Apart from the advanced deferred rendering model and the physics and all the other great stuff Torque can 'do'. What you are thinking of here is more likely what can you do if you load up the game and start walking around?
Actually, T3D has alot of gameplay features out of the box. You can take the full template and you will be able to run around, shoot, mount vehicles, throw mines etc.
There is a lot of FPS features already implemented, including multiplayer support!
So if you want to make a FPS you can probably (at first) treat the development as modding the engine. A good bet for you would be to start at the FPS tutorial (see link at the end of the tutorial)

Setting up your environment

What do i need to get started with developing Torque 3D?

(The following steps is only necessary if you are not satisfied with just scripting and at some point want more than that. If you are only here for the scripting the binary files in the repo will be enough.)

Before you download the repo, you should be sure to setup your environment correctly!

First do you have an IDE? (Integrated Development Environment)

No? If you are running windows you should go ahead and download Visual Studio 2010 (the express version is free).

Visual studio 2012 is okay aswell, but there is some small fixes to be done in that version.

Now, you need to download the DirectX SDK or else you can't compile the engine.

If you plan to be using the physics you will need the PhysX SDK

Setting up your environment (continued) - Scripting

If you wanna throw some money at a scripting IDE you can use Torsion it is a very good IDE for scripting TorqueScript.
If you would rather go the free route however you can either download this extension for notepad++
(Installation instructions by Seizure 22) Quote: *Place it in the notepad++ root folder or its application data folder, depending on the option you set while installing*
For me it was *"C:/Users/***UserName***/AppData/Roaming/Notepad++"*
You can also go for the TIDE by Paul Dana, a free cross platform IDE for TorqueScript.

What is TorqueScript?

TorqueScript(TS) is a C-like language, it is very basic. There is no 'types' in TorqueScript, everything is handled as strings or ints.
TorqueScript is not object oriented, you can treat it as such but it wasn't made with object oriented programming in mind.
One of the most interesting things about scripting in TS is that it is event driven. The engine runs the game and sends the necessary callbacks to the Script interface.
TS is a (at the time of writing this) a pretty slow scripting language so you should avoid having all your core features in script.
Personally one of thing i use TS for is triggering spell casts and spawn emitters, these are commands that aren't triggered alot of times and extremely handy to have in script.
I wouldn't use scripts for minds for the AI players, for prototyping it is fine but not for the release version of your game.

Learning more about TorqueScript

I can recommend checking the TorqueScript guide

Introduction

To me there is no better way of learning something than to throw yourself into it and get dirty.
So we will start right away with the first project.
I will not go in depth with how the editors works in this tutorial but I will link you to places where you can get help to a specific task.
So for a start I will ask you to create an empty project (choose the full template)
If you don't know how to, take a look at this guide. Now enter the game, press F11 to enter the editor and play around with the terrain editing tools.
An official guide to creating your first terrain

Place some objects

This is a good place to start

But we don't need decoration right now. We need something we can interact with. So we need to create a new object type!

First go into **Scripts/Server/ScriptExec.cs**, at the end of the file add a new line *exec("Coin.cs");* **This is absolutely vital!** If you don't execute your scripts with a exec(""); call. They wont be loaded, you have to execute the scripts to use them.

Now create a new file in **Scripts/Server** called Coin.cs see next page for what to put into the file. **Notice! Here I am not following the conventions on where to place datablocks. Please read the Folder guider**.

The reason I do this is because 1. I wanted to keep it as simple as possible in the first tutorial, and 2. I like to go against the convention, in some cases I prefer having the datablock with the scripts. Of cause it is up to you how you do this, but for a start I recommend following the conventions. Which is what I will do for the rest of the tutorials aswell unless I have a good reason not to.

Coin.cs

```
datablock StaticShapeData( Coin )
{
    category = "TutorialObjects";
    shapeFile = "art/shapes/items/kit/healthkit.dts";
};
```

Granted, it doesn't really look like a coin. But use your imagination!
Alright, to spawn your new object load up the **world editor** *(F11 in game)* make sure you are in the **object editor** *(F1)* and in the **scene tree menu** *(to the right)* press the *'Library'* tab.
Here you will notice that under the *'Scripted'* tab a new folder has appeared, the *'TutorialObjects'* folder. Inside this folder an object called **Coin** has appeared.
Double click it and your first coin will get spawned.

Coin.cs Explained

- **datablock** this is very similar to a struct, basically in a datablock you define a set of default values which will get copied to the spawned objects upon creation.
- **StaticShapeData(Coin)** here we state that we want to create a datablock of the type StaticShapeData, for creating new StaticShapes. We call this new datablock Coin.
- **category** the category attribute is (AFAIK) only for telling the world editor where the new object can be found. (Which is why the Coin object showed up in the 'TutorialObjects' folder)
- **shapeFile** this string defines where the 3D model which the static shape should use is located.

Read about relative path in TS here

The OnCollision Callback

Add this function to the Coin.cs file

```
function Coin::onCollision(%this, %obj, %col, %vec, %len)
{
    %obj.delete();
}
```

Now what can we make of this?
For all static shapes that are using the datablock *Coin*, hence all our Coin objects, we define a function for the onCollision callback.
The engine calls this callback when two objects collide. The parameters it gets is:

- **%this** refers to the datablock.
- **%obj** refers to the coin object.
- **%col** refers to the object colliding with the coin.
- **%vec** is the impact vector which indicates the direction of the impact.
- **%len** is the length of the impact vector, hence the force of the impact.

When an object collides with the coin, it deletes itself. That was pretty easy huh?
This is one of the benefits of an event driven language, it can be incredibly easy to create something cool!
Read more about variables in TorqueScript

SimGroups

Remember how to spawn those coins? Well lets get back to that. We want a way to manage how many coins are left in the game, so when there is no more coins, the game is done.
We will do this by using a **SimGroup**. A **SimGroup** is a collection of objects. If you delete a **Simgroup** all objects inside it will be deleted aswell.
Again head to the *'Library'* tab in the *scene tree menu*. Now i want you to find the *'Level'* tab. Inside the *'Level'* tab go to the *'System'* folder. Inside of this folder there will be a new object called SimGroup, which has an icon similar to a folder. Double click on it to spawn a new SimGroup.

SimGroups (continued)

Make sure you have created atleast, 5 *'Coin'* objects and 1 SimGroup. You can check this by clicking on the *Scene* tab under the *Scene Tree menu* and check the list of objects there.
Now you noticed how the SimGroup looked like a folder? This is because it is used to store other objects!
You can drag and drop the *'Coin'* objects onto the newly created SimGroup and they will be stored there. **Do so now.**
Make sure that you have only stored *'Coin'* objects in the SimGroup.
In the *'Inspector'* panel under the *Scene Tree menu* you will a find a list of attributes you can set for the selected object.
Select the SimGroup and rename it to "Coins" do the same for all of the coin object, rename them to: "Coin1", "Coin2" "Coin3"... Etc.

Adding a victory condition

Did you read the guide to variables? if not then you should go ahead and do it now.
Now we want to add some sort of victory condition, lets say that when all the coins are gone, then we will stop the game.

```
function Coin::onCollision(%this, %obj,%col,%vec,%len)
{
    %obj.delete();
    if(Coins.getCount() <= 0)
    {
        commandToClient(%col.client,'ShowVictory');
    }
}
```

- **if(Coins.getCount() <= 0)** - Very simple. If there is less or equal to 0 coins left in the game, then we should shut the game down.
- **commandToClient(%col.client,'ShowVictory');** this function sends a command to the client, telling it to show the victory message.

You should read the networking commands guide.
Now as you have read in the networking commands guide, we will need a clientCmd function. We will define this in a commands.cs file, read next frame.

Commands.cs

First, we need to add an *exec* call so our new script will get loaded. We will do this in **"Scripts/client/init.cs"**.
Around line 80 after the *exec("./serverConnection.cs");* add: *exec("./commands.cs");*
Now create a new file called commands.cs in **"Scripts/client"** and write the following function in it:

```
function clientCmdShowVictory(%score)
{
    MessageBoxOK("You Win!",
        "Congratulation you found" SPC %score SPC "coins!",
        "disconnect();");
}
```

You should read the guide about string concatenation in Torque
This function simply shows a message box telling the client that he has won, and the amount of coins he collected. When he presses ok, the game ends and he returns to the mainmenu.
You will notice that we never passed the score to this command from the server. That is what we are going to fix now.

Point system

Now we are going to make use of some global variables to keep track of the amount of coins we've collected.
Back in *Coins.cs* add *$CoinsFound = 0;* to the top of the file.
And now change the onCollision callback to this:

```
function Coin::onCollision(%this, %obj, %col, %vec, %len)
{
    %obj.delete();
    $CoinsFound++;
    if(Coins.getCount() <= 0)
    {
        commandToClient( %col.client,'ShowVictory',$CoinsFound);
    }
}
```

Now, everytime we pickup a coin the CoinFound global will raise by one. When there there is no more coins in the world the client will win the game when a pop-up shows him how many coins he gathered.

Relative paths in TorqueScript

- "./" refers to the location of the file calling the function.
- " /" refers to the root-child the file lives in ("Game/Scripts/Clients/somefile.cs" becomes "Game/Scripts")
- "/" or nothing refers to the location of the executeable.

Variables in TorqueScript

Variables should always start with either a % or a $.
% means the variable is a local variable and cannot be referenced outside of the function which defined it.
$ means the variable is a global variable and can be accessed from anywhere.
Globals defined on the client side is not defined on the server side. They are only accessible from the current instance!
Arrays in TorqueScript can be faked, meaning that $array[0] is the same as $array0.
If you want a true array you should use the ArrayObject class.

Commands between server and client

The functions *commandToClient and commandToServer* sends a command to the client or the server.
commandToClient takes 2+ parameters. The first one being the id of the client, the second is the tag of the command the client should execute. All following parameters is the arguments passed to the command.
commandToServer is similar, except it doesn't need an ID so it takes 1+ parameters.
To define a function which can be called by a network command, it has to follow a special naming convention.
clientCmd*theCommandName***(args);**

String concatenation

- @ (Concatenates two strings)
- TAB (Concatenation with a tab)
- SPC (Concatenation with a space)
- NL (Newline)

Folders in T3D

Art
Contains all your art files (obviously), datablocks, GUI, fonts. Everything non-game logic related should be located here

Core
Contains alot of core functionality and functions. You can look at this as a *"framework"* for your game logic.

Scripts
Contains all of your script files in 3 specific folders. The client scripts in the client folder, the server scripts in the server folder, the GUI scripts in the GUI folder

Levels
Contains your level files (your terrains go here when you create them)

Tools
The world tools, this only contains the tools for editing the world and can be deleted upon release

Shaders
Contains your shaders.

Troubleshooting

I just downloaded the T3D OS repo and I can't move stuff around in the world editor! This is probably due to the grid settings. See the magnets at the top of the window?
Try turning them on and off. This should help. Make sure they are off. Especially the magnet with a grid.

Useful links

- FPS Tutorial
- Official documentation
- Guides
- GarageGames on Vimeo
- T3D docs
- Plastic Gems
- T3D noob help hotline: The forums or my email: LukasPJ@FuzzyVoidStudio.com