

Torque Game Builder – Config Datablocks

Introduction

When creating a large scale project, any repetitive task you can cut out of your work flow is a good thing. Thanks to config datablocks, the setting up of multiple similar or identical objects is very simple. Config datablocks are basically saved sets of parameters that you can apply to any object in the Level Builder. They are handy because when creating a great deal of identical objects, you only need to write their parameters once and then reference those parameters (via a config datablock) as many times as you need. This tutorial is going to go through the creation of a config datablock and show you how they can be used to make repetitive tasks much less time-consuming.

Creating a Config Datablock

The first thing we are going to do is create a config datablock and then later access it using the Level Builder. We are going to create this config datablock in our TGB project, since it is a default project all TGB owners will have access to. So, our first step is to navigate to our games/TGB/gameScripts folder. In this folder we are going to create a new file called configs.cs. This file will contain our config datablock. Config datablocks can be created in any part of our code, however for the sake of organization I decided to create our config datablock in a separate file. Inside this file we are going to create a very simple config datablock that is just going to show you how they work. Add this code to your configs.cs file:

```
datablock t2dSceneObjectDatablock(TestConfig)
{
    Layer = 1;
    Size = "100 100";
};
```

This code just specifies a config datablock with the name TestConfig that sets the object's layer to 1 and its size to 100 along both the X and Y. Next we need to make sure the code is ran, so navigate to your games/TGB folder and open your main.cs file. Find the following code:

```
//exec game scripts
exec("./gameScripts/game.cs");
```

And add the following code above it:

```
//exec config datablocks
exec("./gameScripts/configs.cs");
```

We put the exec statement for our config datablock first because there may be instances when we want our config datablocks to load before any of our game scripts. So, though this will not apply in this particular instance, it is a good habit to get into. This should be all we need to test out our config datablocks.

Testing Our Datablock

Now that we have our config datablock created, open up Torque Game Builder. Next, if your TGB project is not already open, navigate to File-> Open Project and open your TGB project. Once it's loaded we should begin testing of our datablock. First grab the static sprite of the GarageGames logo and drag it onto our scene. This is so when we drag in another logo with the datablock assigned to it, we will see the difference.

Torque Game Builder – Config Datablocks

Config datablocks can be set in the Datablock rollout in the Create tab of the Level Builder (as shown in Figure 1).



Figure 1

As you can see in figure 1, when you expand this dropdown, we can see the datablock we created in the last step. Select this datablock so it appears in the window of the collapsed list (as shown in Figure 2).



Figure 2

Now, every object you drag into the scene will have this config datablock assigned to it. Drag in another GarageGames logo, this time with the config datablock assigned to it (as shown in Figure 3)

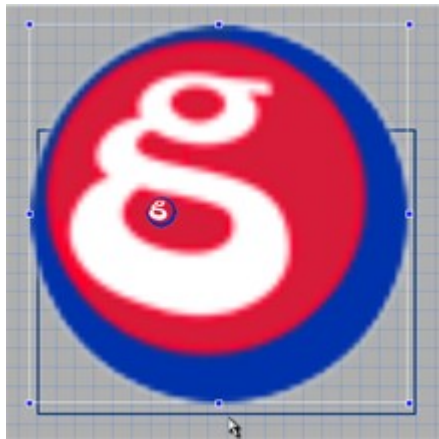


Figure 3

You should immediately notice the difference between your object that has a config datablock assigned to it, and your object that does not have one assigned to it. However, let's say that you assigned an object with a datablock, but didn't want all of the parameters assigned by the datablock to be applied to your object. Luckily, this is no problem since the parameters of a config datablock can be overridden once they are set. So if you now manually set the object's size, it will override the config datablock's size.

Conclusion

Config datablocks are a very efficient way to assign parameters to multiple objects while only having to write them in code once. They are also flexible enough that you can override them if you need. Using config datablocks well can save you a lot of time and coding by eliminating repetition; allowing programmers to give focus to other essentials.