Beginners guide to T3D part three

Lukas Peter Joergensen

September 26, 2012

Revision 1

Introduction

This is part 3 of a beginners introduction to T3D tutorial series I am writing.
In this tutorial we will create some GUI, more specifically we will create our own scoreboard.
We will handle several topics in this part of the series. If statements in TorqueScript, msgCallBacks, dynamic interface and more.

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. Objects and child objects
3. Setting up
4. The scoreboard
   - The structure
   - The style
   - The functionality
     - Keybindings
   - Callbacks
5. Quick guides

Objects and child objects

Remember the thing I called *the constructor* when instancing new objects?
This constructor has an interesting feature. If you were to instance and object like a SimGroup, which contains more objects you could do this:

```
%obj = new SimGroup(){};
%obj.addObject(new SimObject(){});
%obj.addObject(new SimObject(){});
etc..
```

But the constructor allows us to instantiate the child objects inside the constructor:

```
%obj = new SimGroup(theGroup){
   new SimObject(){};
   new SimObject(){};
};
```

We will use this feature when writing new .gui files:

```
%guiContent = new GuiControl(ScoreBoardGui){
   new GuiBitmapBorderControl(){
      new GuiBitmapControl(){
      };
   };
};
```

This code will create a new control with a bitmap border inside, and a bitmap inside the border. Get the idea? This is how we parent the Gui elements.

Setting up

I prefer writing my GUI in script, because it gives a better and cleaner output, and the editor feels a bit clumsy to me.

Setting up the files

First add a new file in **art/gui** called *scoreBoard.gui*.
Then add a new file in **scripts/gui** called *scoreboardGui.cs*
In **scripts/client/scriptExec.cs** add these two lines:

```
exec("./../gui/scoreboardGui.cs");
exec("art/gui/scoreBoard.gui");
```

The server shouldn't handle GUI scripts. The GUI is for the clients which is why we exec them in the client's scriptExec.cs file.
Can't find scriptExec.cs? then you should go through part 1 again!

The scoreBoard.gui

scoreBoard.gui is where we will specify the structure of the scoreboard.
First we will create the base container
Write this inside of scoreBoard.gui

```
//--- OBJECT WRITE BEGIN ---
%guiContent = new GuiControl(ScoreBoardGUI) {
    position = "0 0";
    extent = "1024 768";
    profile = "GuiModelessDialogProfile";
    tooltipProfile = "GuiToolTipProfile";
    isContainer = "1";
    canSaveDynamicFields = "1";
        enabled = "1";
        noCursor = "1";
};
//--- OBJECT WRITE END ---
```

This is a simple control that basically fills the whole screen (starts at top left corner, scales down to the top right corner and it is anchored to the right and bottom sides)
Now we need some content inside of it!

A centered container

Now we will need to center the content inside the ScoreBoardGUI so it is centered on the screen. Add this to the GuiControl ScoreBoardGUI (Remember how we did it? Read frame 5)

```
new GuiPanel() {
      docking = "None";
      position = "370 271";
      extent = "283 226";
      horizSizing = "center";
      vertSizing = "height";
      profile = "ScoreBoardProfile";
      tooltipProfile = "GuiToolTipProfile";
   };
```

This creates a new panel in the center of the screen. Extent means the size of the panel. This panel is 283px wide and 226px tall.
We make sure it is centered by letting position be (screenwidth/2)-(extent/2). In this case it is (1024/2)-(283/2). (Likewise for height)
We set vertSizing to height, which means that it will follow the height of the screen and thus scale vertically.
We set horizSizing to center, so that it wont scale horizontally. This is because a horizontal scaling would corrupt the design of the scoreboard.

Headers for the scores

We will be using a GuiTextListCtrl to list the players. We can consider it like a table. Since every new Text in the list is a row, and we can specify columns by tabs.
Therefore we need some headers for the values "Coins, Kills, Deaths" (yes we will add in some 'kill each other' feature in a later tutorial)
So put this inside the GUIPanel you just created:

```
new GuiTextCtrl() {
   %text = "Coins";
   maxLength = "255";
   position = "104 2";
   extent = "33 18";
   profile = "HudTextBoldProfile";
   tooltipProfile = "GuiToolTipProfile";
};
new GuiTextCtrl() {
   %text = "Kills";
   maxLength = "255";
   position = "158 2";
   extent = "30 18";
   profile = "HudTextBoldProfile";
   tooltipProfile = "GuiToolTipProfile";
};
```

... Continued

```
new GuiTextCtrl() {
    %text = "Deaths";
    maxLength = "255";
    position = "206 2";
    extent = "37 18";
    profile = "HudTextBoldProfile";
    tooltipProfile = "GuiToolTipProfile";
};
```

As you can see the position value is not very high on these elements, thats because their parent is the GuiPanel so their position is relative to the panel!.

The scrollbars

If alot of players is joining, we will need some scrollbars. For this we need a GuiScrollCtrl placed inside of the GuiPanel

```
new GuiScrollCtrl() {
    willFirstRespond = "1";
    hScrollBar = "alwaysOff";
    vScrollBar = "dynamic";
    lockHorizScroll = "1";
    lockVertScroll = "0";
    constantThumbHeight = "0";
    childMargin = "0 0";
    mouseWheelScrollSpeed = "-1";
    position = "0 24";
    extent = "228 202";
    horizSizing = "width";
    vertSizing = "height";
    profile = "ChatHudScrollProfile";
    tooltipProfile = "GuiToolTipProfile";
    isContainer = "1";
}
```

As you might be able to see, we don't want a horizontal scrollbar so it is always off (hScrollBar = "alwaysOff") and the vertical scrollbar only shows if it is necessary (vScrollBar = "dynamic") beside from that, all these settings should be fairly straight forward.

Now last but not least we need the aforementioned GuiTextListCtrl placed inside the GuiScrollCtrl.

```
new GuiTextListCtrl(ScoreBoardGUIList) {
    columns = "0 98 153 200";
    fitParentWidth = "1";
    clipColumnText = "0";
    position = "0 0";
    extent = "228 8";
    horizSizing = "width";
    vertSizing = "height";
    profile = "HudTextNormalProfile";
    tooltipProfile = "GuiToolTipProfile";
    isContainer = "1";
};
```

The important things to note here is the **columns** attribute, the **fitParentWidth** attribute and the **clipColumnText** attribute.
columns this specifies where the columns will be (on the x-axis relative to the GuiTextListCtrl). As you might have noticed this matches the positions of the GuiText controls!
fitParentWidth this makes the TextListCtrl fill the Scroll horizontally.
clipColumnText this makes sure that the columns don't break or overwrite each other. If the string in this column is longer than the column it will be cut out.

The final structure

When you are done your structure for the GUI should look something like this:

- **GuiControl** *"ScoreboardGUI"*
  - **GuiPanel**
    - **GuiTextCtrl** *"Coins"*
    - **GuiTextCtrl** *"Kills"*
    - **GuiTextCtrl** *"Deaths"*
    - **GuiScrollCtrl**
      - **GuiTextListCtrl** *"ScoreBoardGUIList"*

Styling the GUI

There is one thing that you may have noticed and that I haven't explained!
The profiles! The profile is to GUI objects what CSS is to HTML. They define the style of the GUI's
I wont spend a lot of time with the profiles even tho they are quite important. Therefore we will use alot of the stock profiles and only create a single custom one.
In **art/gui/customProfiles.cs** (or create your own profile file) add the following snippet:

```
singleton GuiControlProfile (ScoreBoardProfile)
{
   opaque = "1";
   fillColor = "0 0 0 200";
   fillColorHL = "0 0 0 200";
   borderColor = "0 0 0 255";
   borderThickness = "5";
   border = "1";
};
```

This should be pretty easy to understand. All controls using this profile has a black background which is transparent and the border is 5 px wide, black and not transparent.
There is alot more settings that can be played with, for setting text color bevel etc etc.. This is not within the scope of this tutorial unfortunately!

The functionality

Now we need to add some functionality to the scoreboard!
First we should be able to see it shouldn't we?
At the bottom of **scripts/gui/scoreboardGUI.cs** add the following code:

```
//-----------------------------------------------------------------------------
// ScoreBoardGUI utility methods
//-----------------------------------------------------------------------------
function ScoreBoardGUI::toggle(%this)
{
   if (%this.isAwake())
      Canvas.popDialog(%this);
   else
      Canvas.pushDialog(%this);
}

function ScoreBoardGUI::clear(%%this)
{
   // Override to clear the list.
   ScoreBoardGUIList.clear();
}
```

This toggles the ScoreBoardGUI so a call to ScoreBoardGUI.toggle() will make it pop up on the screen. If we then call toggle() again it will hide itself.
Now, this is just a function we need a way to call it. And calling it through the console is quite.. Clumsy.. So lets add a keybinding!

Keybindings

Keybindings should always go in the **scripts/client/default.bind.cs** file.

```
function showScoreBoard(%val)
{
    if (%val)
        ScoreBoardGUI.toggle();
}

moveMap.bind( keyboard, F, showScoreBoard );
```

ActionMaps is kindda like, a set of keybindings. You can pop and push action maps and as such change how our input affects the game. Read old docs about Torques input model
So what happens here is that we bind the key F to the function showScoreBoard on the action map *moveMap*. moveMap is the actionmap that is on by default when you enter the game.
If you open up the game you might experience that pressing F doesn't work.. This is because T3D generates a **scripts/client/config.cs** that is used to save the users custom keybindings. Simply delete config.cs to revert all keybindings to default and to invoke your new keybinding, or add the binding there aswell.

Add and remove a player

We want to be able to add and remove players. So we need a function for this in **scripts/gui/scoreboardGUI.cs**. Add this before the Utility header:

```
//-----------------------------------------------------------------------------
// ScoreBoardGUI data handler methods
//-----------------------------------------------------------------------------

function ScoreBoardGUI::addPlayer(%this, clientID, %name, %score,
                                  %kills, %deaths)
{
   %text = StripMLControlChars(%name) TAB %score TAB %kills
                                      TAB %deaths;

   // Update or add the player to the control
   if (ScoreBoardGUIList.getRowNumById(%clientId) == -1)
      ScoreBoardGUIList.addRow(%clientId, %text);
   else
      ScoreBoardGUIList.setRowById(%clientId, %text);

   // Sorts by score
   ScoreBoardGUIList.sortNumerical(1, false);
   ScoreBoardGUIList.clearSelection();
}
```

Add and remove a player (continued)

So lets see what this does. We create a new variable '%text' which stores name, score, kills and deaths with columns seperated by tabs.
Then we check if there is already stored a row with that id. If not, then we add a new row else we update the row with that id.
Finally we sort the list by column 1 (the second column because it starts at 0).
Now we need functionality to remove a player again! Add the following below addPlayer:

```
function ScoreBoardGUI::removePlayer(%this, %clientId)
{
    PlayerListGuiList.removeRowById(%clientId);
}
```

Very simple, remove the row with the given id.

Updating a players status

Add this below removePlayer

```
function ScoreBoardGUI::updatePlayer(%this, %clientId, %score,
                                     %kills, %deaths)
{
   %text = ScoreBoardGUIList.getRowTextById(%clientId);
   if(%score !$= "null")
      %text = setField(%text, 1, %score);
   %text = setField(%text, 2, %kills);
   %text = setField(%text, 3, %deaths);
   ScoreBoardGUIList.setRowById(%clientId, %text);
   ScoreBoardGUIList.sortNumerical(1, false);
   ScoreBoardGUIList.clearSelection();
}
```

I made 2 quick guides about the setField method and the if statement. So i wont explain them here! Rest of it is pretty simple, so I wont spend time a lot of time explaining these things. Of course setRow sets the row with the new data etc.
However why is only %score checked for being null? Because, later we will want to call this function without setting the score, and then we pass null as a string. It has nothing to do with validating the data or checking if it is an empty string!

Resetting scores

Put this below UpdatePlayer

```
function ScoreBoardGUI::resetScores(%this)
{
   for (%idx = 0; %idx < ScoreBoardGUIList.rowCount(); %idx++)
   {
      %text = ScoreBoardGUIList.getRowText(%i);
      %text = setField(%text, 1, "0");
      %text = setField(%text, 2, "0");
      %text = setField(%text, 3, "0");
      ScoreBoardGUIList.setRowById(ScoreBoardGUIList.getRowId(%idx),
   }
   ScoreBoardGUIList.clearSelection();
}
```

This iterates through all the rows in ScoreBoardGUIList and sets score, kills and deaths to 0. This is very similar to the UpdatePlayer function.

Make the scoreboard update

So what we have been doing so far, is to add alot of functionality. However if we don't put it to use what is it worth?
So we will add something called messagecallbacks. Which is very similar to a commandToClient function but has more functionality, and can call multiple functions, not just a single one.
First lets add the callbacks put this in top of scoreboardGUI.cs

```
//------------------------------------------------------------------------------
//  Callbacks
//------------------------------------------------------------------------------
addMessageCallback( 'MsgClientJoin', SBGUIPlayerJoined );
addMessageCallback( 'MsgClientDrop', SBGUIPlayerLeft );
addMessageCallback( 'MsgClientScoreChanged', SBGUIScoreChanged );
addMessageCallback( 'MsgCoinPickedUp', SBGUICoinPickup );
```

'MsgClientJoin', 'MsgClientDrop' and 'MsgClientScoreChanged' is some messages that is already implemented. The last one *'MsgCoinPickedUp'* is one that we will specify later so you can see how they work!.

Add these right below the 'addMessageCallback' calls

```
function SBGUIPlayerJoined(%msgType, %msgString, %clientName,
                           %clientId, guid, %score, %kills,
                           %deaths, isAI, isAdmin, isSuperAdmin)
{
   ScoreBoardGUI.addPlayer(%clientId, %clientName, 0, %kills,
                                                      %deaths);
}

function SBGUIPlayerLeft(%msgType, %msgString, %clientName,
                                                      %clientId)
{
   ScoreBoardGUI.removePlayer(%clientId);
}

function SBGUIScoreChanged(%msgType, %msgString, %score,
                                          %kills, %deaths, %clientId)
{
   ScoreBoardGUI.updatePlayer(%clientId, "null", %kills, %deaths);
}

function SBGUICoinPickup(%msgType, %msgString, %clientName,
                                       %clientId, %score, %kills, %deaths
{
   ScoreBoardGUI.updatePlayer(%clientId, %score, %kills, %deaths);
}
```

Custom callback

If you open up the game now and press 'F' you should see that a player has been added with 0 coins, kills and deaths. Great! But if you pick a coin up nothing happens.
This is because we need to add our own message. So go into **scripts/server/Coin.cs** and add the following in **Coin::onCollision after %col.client.coinsfound++;**:

```
messageAll('MsgCoinPickedUp', -1, %col.client.playername,
        %col.client, %col.client.coinsfound,
        %col.client.kills, %col.client.deaths);
```

This sends a message to all clients, telling them that the client has picked up a new coin. This gets picked up by our callback in the previous slides and makes the scoreboard add one to the "Coin" score of the client.

Finishing touch

We want to make a couple of calls from different places in the scripts so that the table is empty when a new game is started
In **Scripts/client/game.cs** in **clientCmdGameStart** Add *ScoreBoardGUI.resetScores();* In **Scripts/-client/serverConnection.cs disconnectedCleanup** add *ScoreBoardGUI.clear();* And thats it! Enjoy your new scoreboard!

If statements in TorqueScript

Remember that i said everything in TorqueScript is strings?
Well this is important to remember with if statements! Because in if statements you either compare by int or by string.
To compare by string you should write $= instead of ==
Example:

```
if("asd" == "asd") // This is true
if("asd" == "def") // This is true aswell
if("asd" $= "asd") // This is true
if("asd" $= "def") // This is false
```

Fields and words

There is some pretty cool functions called getWord, setWord, getField, setField.
These functions lets you edit a string at an index based on the number of either spaces or tabs.
getField, is based on tabs. getWord is based on spaces.
Examples:

```
%val = "1 2 3 4";
echo(getWord(%val, 2)); //Echoes 3
echo(getWordCount(%val)); //Echoes 4
echo(setWord(%val, 2, "10")); //Echoes "1 10 3 4"
echo(%val); //Still echoes "1 2 3 4"
%val = setWord(%val,2,"10");
echo(%val); //Echoes "1 10 3 4"
%val = 1 TAB 2 TAB 3 TAB 4;
echo(getField(%val, 2)); //Echoes 3
```