CustomScript

Erstellt von King Pin, zuletzt geändert von Rochet2 am Feb 24, 2016

Adding a script to the server

This guide works for Linux and Windows.

Lets assume we you have created a script and saved it in a file called **My_script.cpp** in the source directory **/src/server/scripts/Custom**
For this guide lets use this script as an example of what My_script.cpp could look like:

My_script.cpp

```
#include "ScriptMgr.h"
// .. more includes

class my_script_class: CreatureScript
{
public:
    my_script_class() : CreatureScript("my_script") {}

    // more code ..
};

void AddSC_my_script()
{
    new my_script_class();
}
```

The example script uses the function called **AddSC_my_script** to link it to the core code. You need to use an *unique name* for your function - otherwise the compiler will error as there are two identical functions.

First you need to add the declaration of the function that is used to link your script and then you add the call to the function - soon we show where to place them.
For our example code the function was called **AddSC_my_script**. Here is an example of how the declaration and the call look like:

```
void AddSC_my_script(); // this is a declaration
AddSC_my_script();      // this is a call
```

Open the file: /src/server/scripts/Custom/custom_script_loader.cpp and scroll to the bottom of it.
The declaration needs to be added below the line that says: **This is where scripts' loading functions should be declared**
Here is an example of how the code in the file can look after adding the function declaration to it:

```
// This is where scripts' loading functions should be declared:
```

Next you add the function call inside the function called **AddCustomScripts**
Again here is an example of how the code could look afterwards:

```
// The name of this function should match:
// void Add${NameOfDirectory}Scripts()
void AddCustomScripts()
{
    AddSC_my_script();
}
```

Here is a final example of how the file could look after both edits:

```
// This is where scripts' loading functions should be declared:
void AddSC_my_script();


// The name of this function should match:
// void Add${NameOfDirectory}Scripts()
void AddCustomScripts()
{
    AddSC_my_script();
}
```

Next you need to use cmake and compile.

Linking the script in the database

You usually need to link your script to the appropriate areatrigger/creature/gameobject/instance/item that the script is for.
Some script types, such as CommandScripts, PlayerScripts and WorldScripts, do not need to be linked to anything.

You should be able to find a **scriptname** column in the database for the entity you need to link the script to. Very often it is in the same database table the entity itself is in.
To this column you should place the script name - *which is supposed to be unique*. The script name in our example script is **my_script** and can be found in the code in this part: **CreatureScript("my_script")**

Examples - note that only the creature_template SQL is correct for our example script

```
UPDATE `areatrigger_scripts` SET ScriptName='my_script' WHERE `entry`=XXXX;
UPDATE `creature_template` SET ScriptName='my_script' WHERE `entry`=XXXXX;
UPDATE `gameobject_template` SET ScriptName='my_script' WHERE `entry`=XXXXXX;
UPDATE `instance_template` SET ScriptName='my_script' WHERE `map`=XXX;
UPDATE `item_template` SET ScriptName='my_script' WHERE `entry`=XXXXX;
```

Note: You might need to edit the creature/gameobject/etc to make sure the script has effect. For example, if you have created a gossip script for a creature, you will need to set creature_template.npcflag=1 to enable gossip flag on the creature or the script will not work.

TK gefällt das.

guides_for_trinity