Subject: Re: weird reaction with const w_chart
Posted by iRANian on Sun, 11 Mar 2012 00:05:06 GMT
View Forum Message <> Reply to Message

robbyke wrote on Fri, 09 March 2012 10:51thanks this helped alot

made me understand stringclass better to
Cheers no problem. I suggest you take a look at HashTemplateClass and HashTemplateIterator too, they provide hash maps/tables, also known as dictionaries in other languages. If you're gonna be building a bot that responds to commands, it's a nice idea to store them in a hash table, with as key the text that triggers the command and the value being a pointer to a class that inherits a generic chat command class, so you can call a function from a class when it is found in the hash table, which is done almost instantaneously, instead of having to go over a list of all triggers to trigger a specific function, which is very slow and makes the code to check what command to trigger very long-winded code.

I've written an example for this that I haven't tested, I hope this helps you:

derp.cpp:
```
#include "General.h"
#include "derp.h"
#include "HashTemplateClass.h"
#include "HashTemplateIterator.h"

// This macro creates a new Class on the heap and cals Startup for it
#define REGISTER_COMMAND(Class, Trigger, registrant) Class *registrant = new Class; registrant->Startup(Trigger)

HashTemplateClass<StringClass, ChatCommand *> ChatCommand::CommandsMap;

void ChatCommand::Load()
{
 REGISTER_COMMAND(Help, "!help", Help_Registrant); // Registers a new Help ChatCommand objectcalled with !help
 REGISTER_COMMAND(Help, "!h", HelpAlias_Registrant); // Registers a new Help ChatCommand object called with !h
}

void ChatCommand::Unload()
{
}

// This simply adds the command to our CommandsMap with the trigger, it can later be expanded to do more
void ChatCommand::Startup(const char* Trigger)
{
 ChatCommand::CommandsMap.Insert(Trigger, this); // 'this' is a pointer to the current class
}
```

```
// This is a virtual method, this is used for the default for classes that don't have the Activate()
method
void ChatCommand::Activate(int ID, int Type, StringClass Msg)
{
 Console("MSG this command hasn't been implemented yet!");
}

// This is the only thing we nee to implement for a new class inheriting from ChatCommand
void Help::Activate(int ID, int Type, StringClass Msg)
{
 Console("MSG This is a fancy help command!");
}

// Example chat hook code to get the above to work
bool KamBot::OnChat(int PlayerID, TextMessageEnum Type, const wchar_t *Message, int
recieverID)
{
 StringClass Msg = Message;

 if (ChatCommand::CommandsMap.Exists(Msg)) // Does our hash map contain a trigger that is
the same as the input message?
 { // If so
  ChatCommand* c = ChatCommand::CommandsMap.Get(Msg, 0); // Get the ChatCommand
pointer that's indexed for the chat message
  c->Activate(PlayerID, Type, Msg); // With our ChatCommand pointer called 'c', call the Activate()
function
 }

 return true;
}

derp.h:

class ChatCommand
{
public:

 // Static functions and data, dont need to be called from an object
 static HashTemplateClass<StringClass, ChatCommand *> CommandsMap; // This is our hash
map, we put ChatCommand object pointers here
                // That are triggered by a StringClass trigger text

 static void Load(); // This is our loading code, call this from somewhere
 static void Unload(); // Unloads all chat commands related stuff

 void Startup(const char* Trigger); // Loads a ChatCommand object and adds it to the hash table
```

```
 virtual void Activate(int ID, int Type, StringClass Msg); // The default Activate() function called if not defined by an inheriting class
};

// a class that inherits the ChatCommand class, this one implements the !help command
class Help : public ChatCommand
{
public:
 void Activate(int ID, int Type, StringClass Msg); // The code to execute when this function is called in the chat hook by its trigger
};
```

This code should respond to ingame chat that's "!h" or "!help", but I haven't tested it.
