Subject: [CODE] cScTextObj class definition
Posted by iRANian on Fri, 06 Apr 2012 13:14:47 GMT
View Forum Message <> Reply to Message

This is what the engine uses to send out most text "stuff" to players. It's used by the PPAGE, MSG, AMSG, TMSG console commands and it's called whenever a player sends team, public and private chat to the server (to echo the text to the target(s)).

ID is the player ID of the guy to send the text as, this is set to -1 if you want to send as the host (for PPAGE, MSG etc)

Target is a specific player ID to send the message to, leave this -1 if you don't have a specific target (e.g. want to send a team chat or a HOST message).

Type can be public, private or team chat. Team chat is done with an 'ID' that is a valid player ID, 'Target' set to -1 and type set to TEXT_MESSAGE_TEAM (value is 1).

Message is the message to send..

IsPopup is used by the AMSG console command and variants of it, it will show the text as a message window.

.h

```
class cScTextObj : public NetworkObjectClass
{
public:

 int ID;
 int Target;
 TextMessageEnum Type;
 WideStringClass Message;
 bool IsPopup;

 void Init(const WideStringClass& Message, TextMessageEnum Type, bool IsPopup, int ID, int Target);
 cScTextObj* Constructor();

 void Act();
}
```

.cpp (without the header #includes)

```
void* HookupAT3x(void* a, void* b, void* c, void* patch_start, void* patch_end, int (*version_selector)())
{
 return HookupAT3(a,b,c,patch_start,patch_end,version_selector);
}
```

```
RENEGADE_FUNCTION
uint Send_Object_Update(NetworkObjectClass* object, int remoteHostId)
AT2(0x00461820,0x004612F0);

RENEGADE_FUNCTION
void cScTextObj::Act()
AT2(0x004B9720, 0x004B9720);

RENEGADE_FUNCTION
cScTextObj* __thiscall cScTextObj::Constructor()
AT2(0x004B9140, 0x004B9140);
//AT2(0x004B5AA0, 0x004B5AA0);

RENEGADE_FUNCTION
void cScTextObj::Init(const WideStringClass& Message, TextMessageEnum Type, bool IsPopup,
int ID, int Target)
AT2(0x004B91F0, 0x004B91F0);
```

Examples:

Have PlayerID show him sending a team chat mesage with the text "DERP", only on his client (the other players won't see this team chat message)

```
// Create a new cScTextObj
cScTextObj* TextObj = (cScTextObj*)operator new(sizeof(cScTextObj));
TextObj = TextObj->Constructor();


TextObj->ID = PlayerID; // Sender is PlayerID
TextObj->Type = TEXT_MESSAGE_TEAM; // Team chat message
TextObj->Message = L"DERP"; // Message to send

// Show the message only to PlayerID
TextObj->Set_Object_Dirty_Bits(PlayerID, NetworkObjectClass::BIT_CREATION);
Send_Object_Update(TextObj, PlayerID);

// Delete stuff
TextObj->Set_Delete_Pending();
delete TextObj;
```

Show a message window for every player (same thing as AMSG does):

```
cScTextObj* TextObj = (cScTextObj*)operator new(sizeof(cScTextObj));
TextObj = TextObj->Constructor();
TextObj->Init(L"DERP", TEXT_MESSAGE_PUBLIC, true, -1, -1);
```

Send a host message to all players not running scripts 2.9 or higher and send a white coloured message for those who do (can be expanded on to change the name of a player and send the text as that player, then change his name back):

```
void Send_Special_Host_Message(const char *Format, ...)
{
 char buffer[256];

 va_list va;
 _crt_va_start(va, Format);
 vsnprintf(buffer, 256, Format, va);
 va_end(va);

 cScTextObj* TextObj = (cScTextObj*)operator new(sizeof(cScTextObj));
 TextObj = TextObj->Constructor();

 TextObj->Message = buffer;

 for (SLNode<cPlayer>* PlayerIter = Get_Player_List()->Head(); (PlayerIter != NULL); PlayerIter =
PlayerIter->Next())
 {
  cPlayer *p = PlayerIter->Data();

  if (p->IsActive && Get_GameObj(p->Get_Id()))
  {
   float Version = Get_Client_Version(p->Get_Id());

   if (Version < 2.9)
   {
    TextObj->Set_Object_Dirty_Bits(p->Get_Id(), NetworkObjectClass::BIT_CREATION);
    Send_Object_Update(TextObj, p->Get_Id());
   }
   else
   {
     Send_Message_Player(Get_GameObj(p->Get_Id()), 250, 255, 255, buffer);
   }
  }
 }
 TextObj->Set_Delete_Pending();
 delete TextObj;
}
```

Thanks to StealthEye with helping me out with getting this to work and showing that intermixing CMSGP and MSG can be done. I got lots of useful information for reversing this from the OnOeS source code (the cCsTextObj hooking stuff).

Subject: Re: [CODE] cScTextObj class definition
Posted by halo2pac on Fri, 06 Apr 2012 19:00:52 GMT
View Forum Message <> Reply to Message

Im curious, what is the benefit of this versus what we have been using for years?

Subject: Re: [CODE] cScTextObj class definition
Posted by iRANian on Fri, 06 Apr 2012 20:18:13 GMT
View Forum Message <> Reply to Message

There are various uses, see the last example in my OP, another example would be sending a host message to one player only and you can "create" messages that are sent as if a player typed them in, e.g. you can create a public chat message for a player with chat message being "your maps is bad" or whatever you want (same thing as what TMSG does but then for public chat).

Subject: Re: [CODE] cScTextObj class definition
Posted by ThisLittleGirl on Sat, 07 Apr 2012 18:57:59 GMT
View Forum Message <> Reply to Message

Just wondering, there's probably good reasons you do it this way and I may be missing something badly but...

iRANian wrote on Fri, 06 April 2012 06:14
RENEGADE_FUNCTION
cScTextObj* __thiscall cScTextObj::Constructor()
AT2(0x004B9140, 0x004B9140);
//AT2(0x004B5AA0, 0x004B5AA0);

Why not just declare the constructor as such instead of as a member function, like so:

RENEGADE_FUNCTION
cScTextObj::cScTextObj()
AT2(0x004B9140, 0x004B9140);
Wouldn't that work? The __thiscall should be redundant anyway because it's the default for member functions on MSVC++. And what exactly does the RENEGADE_FUNCTION macro do?

iRANian wrote on Fri, 06 April 2012 06:14

```
// Create a new cScTextObj
cScTextObj* TextObj = (cScTextObj*)operator new(sizeof(cScTextObj));
TextObj = TextObj->Constructor();
// ...
delete TextObj;
```

Also you could just put the thing on the stack instead of that tedious allocation. And isn't what the Constructor function would return the this pointer that you passed in (implicitly via ecx (thiscall))

anyways? In that case the assignment is redundant. If it wasn't the case you'd certainly fuck up yourself later by deleting a bogus pointer.

So, to sum it up. If you declare the constructor as such, you could just do:

cScTextObj text_obj;

Just curious, haha. ;p

Subject: Re: [CODE] cScTextObj class definition
Posted by iRANian on Sat, 07 Apr 2012 20:13:27 GMT
View Forum Message <> Reply to Message

The __thiscall should be unnecessary yeah, it's there just to be clear that it's a member function and not a static member function. The RENEGADE_FUNCTION macro is a define for __declspec naked, if you try to define the Constructor() function as a normal constructor (i.e. cScTextObj:cScTextObj()) the linker gives annoying errors. Creating the object on the stack causes the FDS to freeze or crash on scope exit (not sure why this is), that's why there's the awkward new operator call line, deleting pointers shouldn't be an issue with the simplistic use of the class (you shouldn't need to do anything more exotic than the examples).

Subject: Re: [CODE] cScTextObj class definition
Posted by ThisLittleGirl on Sat, 07 Apr 2012 20:58:10 GMT
View Forum Message <> Reply to Message

iRANian wrote on Sat, 07 April 2012 13:13The The RENEGADE_FUNCTION macro is a define for __declspec naked,
Ah, thought so.

iRANian wrote on Sat, 07 April 2012 13:13if you try to define the Constructor() function as a normal constructor (i.e. cScTextObj:cScTextObj()) the linker gives annoying errors.
Perhaps the NetworkObjectClass you're deriving from (or a base class of it) has its constructor declared private and/or left undefined so you can't actually instantiate it from your code. Because __declspec (naked) works for me on constructor definitions as well. It would in some way make sense do disallow construction within your code since all the code is within renegade. And you're not actually constructing the object yourself, renegade's Constructor function does it. Construction on pre-allocated memory is done with placement-new, something like:

new (TextObj) cScTextObj;

The many faces of new, haha. I hate it, hardly ever even use explicit new/delete since C++11.

iRANian wrote on Sat, 07 April 2012 13:13
Creating the object on the stack causes the FDS to freeze or crash on scope exit (not sure why

this is),

Hmm, weird, that means you can construct it though. But you'll still have to call your Constructor function because your actual constructor will do/does nothing.

Have you thought about the possibility that renegades code may actually write beyond the class' size (as in, you didn't declare all members)? Does it even write into the object? If so writing past it on the stack almost guarantees a crash, haha.

BTW, where can I get to know more about this FDS? Ty.

Subject: Re: [CODE] cScTextObj class definition
Posted by iRANian on Sat, 07 Apr 2012 21:57:37 GMT
View Forum Message <> Reply to Message

Don't think that's the issue, the constructor function calls the NetworkObjectClass' constructor. The cScTextObj class has the NetworkObjectClass' member variables (which has a size of 0x6B4) and those member variables for the class itself, I haven't seen any other variables used and the constructor only calls the NetworkObjectClass constructor and only specifically initializes those 5 member variables (which obviously makes sense as else they would have garbage values) it doesn't use any other variables.

Subject: Re: [CODE] cScTextObj class definition
Posted by jonwil on Sun, 08 Apr 2012 01:24:33 GMT
View Forum Message <> Reply to Message

Because of the way the constructor is being called (which is really a big ugly hack but I know of no better way to do it that doesn't involve sharing large chunks of netcode that I would rather not share), you cant create this cScTextObj object on the stack, you have to create it with operator new.

Subject: Re: [CODE] cScTextObj class definition
Posted by jonwil on Mon, 09 Apr 2012 17:08:59 GMT
View Forum Message <> Reply to Message

I have added a new engine call to 4.0 beta 5 as follows:
cScTextObj *Send_Client_Text(const WideStringClass& message, TextMessageEnum type, bool popup, int senderId, int receiverId, bool dodirtybit, bool doact)

dodirtybit determines whether cScTextObj::Init should make its normal calls to the various Set_Object_Dirty_Bit functions or not

doact determines whether cScTextObj::Init should make its normal calls to cScTextObj::Act

The function will create a new cScTextObj, call its constructor, call cScTextObj::Init and return the new object.
Note that you should not call Set_Delete_Pending, nor should you delete the returned object as Set_Delete_Pending is already handled by cScTextObj::Init and the object will be automatically deleted by the netcode.

You are free to call any of the Set_Object_Dirty_Bit functions and to call Send_Object_Update. Note however that you can only call Send_Object_Update ONCE for each cScTextObj object as cScTextObj was not designed to be sent multiple times. You can also (at any point) call cScTextObj::Act (a virtual function) to cause the message to be processed by the server. (i.e. on the FDS it would do its normal thing and display the output on the console)

The purpose of the engine call is to make this stuff easier to use and to make sure it wont break in the future if cScTextObj has to change for some reason.
