Subject: Re: Headshot message for Server.
Posted by reborn on Tue, 23 Nov 2010 21:41:12 GMT
View Forum Message <> Reply to Message

I wrote this, and it's compiled, but I never actually bothered to test it, mainly because I do not have the custom .wav files you're referencing, and secondly because I couldn't be bothered really.

Here is the first version, written more in the style that jnz posted, with global variables and a code that is not exactly conforming to the standard renegade API useage. I will if I have time write a script that's all tidy and familiar to you and the renegade API.

It's important to remeber though, that while in this instance it's probably better to use the API classes and conform to that standard, it isn't always... There are thing that are deemed "impossible" by some, that really are quite simple useing "hacks" like this.
Plus, if you only learn to code using the renegade API you'll end up very confused later on when you want to write something else.

I wrote this as a SSGM plugin, and have attached the full source code.

```
/* Renegade Scripts.dll
 Example Plugin Code
 Copyright 2007 Whitedragon(MDB), Jonathan Wilson, spencer "reborn" elliott

 This file is part of the Renegade scripts.dll
 The Renegade scripts.dll is free software; you can redistribute it and/or modify it under
 the terms of the GNU General Public License as published by the Free
 Software Foundation; either version 2, or (at your option) any later
 version. See the file COPYING for more details.
 In addition, an exemption is given to allow Run Time Dynamic Linking of this code with any closed source module that does not contain code covered by this licence.
 Only the source code to the module(s) containing the licenced code has to be released.
*/

/*  This is an example of how to implement a server hack to make it register, record and display player kill sprees.
 This has been done WITHOUT the client scripts.dll in mind and has NOT respected the accepted use of the renegade API.
 It uses global variables and custom C++ functions. Ironically it is more readable to other programmers outside the renegade community.
 In this instance however, it is probably easier and more efficient to use the classes that the API lends itself to, this is just an example...
*/

#include "scripts.h"
#include <stdarg.h>
#ifdef WIN32
#define WIN32_LEAN_AND_MEAN
```

```
#include <windows.h>
#endif
#include "engine.h"
#include "gmmain.h"
#include "KillMessages.h"


int PlayerKills[127];

int ObjectHookID = 0;
ObjectCreateHookStruct *ObjectHookStruct = 0;

// This hook is caled when an object is created
void ObjectHookCall(void *data,GameObject *obj)
{
 //This is kinda like a little filter...
 //I'm basically saying here that if the object that was created is a player, then do the following...
 if(Commands->Is_A_Star(obj))
 {
  //This ensures that all players get this script attached to them
  Commands->Attach_Script(obj, "KillMessages", "");
 }
}


void Plugin_Load()
{
 //When the plugin loads this code is called and sets up the object creation hook.
 ObjectHookStruct = new ObjectCreateHookStruct;
 ObjectHookStruct->hook = ObjectHookCall;
 ObjectHookStruct->data = 0;
 ObjectHookID = AddObjectCreateHook(ObjectHookStruct);
}

//This is a C++ function.
//I created the function because I call it several times in different places,
//so it cuts down on the amount of lines I need to write, and makes things neater, it also cuts down
on chanecs for error.
//and if I ned to make a change to the function, it's only here that I need to change it.
//It's use is limited to setting the global variable of PlayerKills for a specific player to Zero.
void ZeroOutKills(int PlayerID)
{
 PlayerKills[PlayerID] = 0;
}

//Another function.
//int Amount is actually over kill TBH, I always know I will be adding only 1 kill at a time, but this is
just to demonstrate really.
```

```
void AddKill(int PlayerID, int Amount)
{
 PlayerKills[PlayerID] += PlayerKills[PlayerID] + Amount;
}

//A funtion to get the running killing spree count of a player
int GetKillingSpreeCount(int PlayerID)
{
 int kills = PlayerKills[PlayerID];
 return kills;
}


//This is the actual "Guts" of the pluging. It is the script that is attached to the players and updates kills etc.
void KillMessages::Created(GameObject *obj)
{
 //When player spawns set their running kills to zero.
 //The function ZeroOutKills takes the player ID, so the below does this: Get_Player_ID(obj)
 //That retrieves the player ID for me from the GameObject, which is the piece of information I know about the player.
 ZeroOutKills(Get_Player_ID(obj));
}

void KillMessages::Killed(GameObject *obj, GameObject *shooter)
{
 //Just a little safety check to make sure that the killer is actually a player and not a patch of tiberium or something stupid like fall damage...
 if(Commands->Is_A_Star(shooter))
 {
  //Update the player's running kill amount that shot the player.
  //So basically what you're doing here is waiting for the player to die
  //then when they do die, update the amount of kills for the person that killed them.
  AddKill(Get_Player_ID(obj), 1);
  if (GetKillingSpreeCount(Get_Player_ID(shooter)) == 2)
  {
   Commands->Create_2D_WAV_Sound("double_kill.wav");
   Console_Input(StrFormat("msg %s made a double kill (2)", Get_Player_Name(shooter)).c_str());
  }

  else if (GetKillingSpreeCount(Get_Player_ID(shooter)) == 3)
  {
   Commands->Create_2D_WAV_Sound("triplekill.wav");
   Console_Input(StrFormat("msg %s made a Tripple. (3)", Get_Player_Name(shooter)).c_str());
  }
  else if (GetKillingSpreeCount(Get_Player_ID(shooter)) == 4)
  {
   Commands->Create_2D_WAV_Sound("multikill.wav");
```

```
   Console_Input(StrFormat("msg %s made a Multikill (4)", Get_Player_Name(shooter)).c_str());
  }
  else if (GetKillingSpreeCount(Get_Player_ID(shooter)) == 6)
  {
   Commands->Create_2D_WAV_Sound("Monsterkill_F.wav");
   Console_Input(StrFormat("msg %s made a M-M-M-M-Monster Kill!!! (6)",
Get_Player_Name(shooter)).c_str());
  }
  else if (GetKillingSpreeCount(Get_Player_ID(shooter)) == 8)
  {
   Commands->Create_2D_WAV_Sound("rampage.wav");
   Console_Input(StrFormat("msg Oh my GOD! %s is on a rampage (7)",
Get_Player_Name(shooter)).c_str());
  }
  else if (GetKillingSpreeCount(Get_Player_ID(shooter)) == 9)
  {
   Commands->Create_2D_WAV_Sound("unstoppable.wav");
   Console_Input(StrFormat("msg %s is unstoppable!!", Get_Player_Name(shooter)).c_str());
  }
  else if (GetKillingSpreeCount(Get_Player_ID(shooter)) == 13)
  {
   Console_Input(StrFormat("msg %s is on a killingspree", Get_Player_Name(shooter)).c_str());
   Commands->Create_2D_WAV_Sound("monsterkill.wav");
  }
  else if (GetKillingSpreeCount(Get_Player_ID(shooter)) == 15)
  {
   Commands->Create_2D_WAV_Sound("godlike.mp3");
   Console_Input(StrFormat("msg %s is godlike...", Get_Player_Name(shooter)).c_str());
  }
 }
}


void Plugin_Unload()
{
 //This is just the unloading of the plugin, the standard example doesn't remove the hook and can
cause a crash.
 //remember to do this in your own plugins.
 RemoveObjectCreateHook(ObjectHookID);
 delete ObjectHookStruct;
}



extern "C"
{
 //This is the player join hook. This code here is called when a player joins the server.
 DLLEXPORT void SSGM_Player_Join_Hook(int ID, const char *Nick)
```

```
 {
  //When the player joins, set their kills to 0.
  //This is just good practice, but probably redundant, as it is also set to 0 when they spawn.
  ZeroOutKills(ID);
 }

 //This is the map load hook. This code is called when the map loads.
 DLLEXPORT void SSGM_Level_Loaded_Hook()
 {
  //This is a simple "for" loop. This loop basically set every single players kills to 0 when the map
loads.
  //It also is probably redundant as when the map loads and the player spawns their kills would be
set to 0.
  for(unsigned int i = 1; i <= 127; i++)
  {
   ZeroOutKills(i);
  }
 }

}




/* Renegade Scripts.dll
 Example Plugin Code
 Copyright 2007 Whitedragon(MDB), Jonathan Wilson

 This file is part of the Renegade scripts.dll
 The Renegade scripts.dll is free software; you can redistribute it and/or modify it under
 the terms of the GNU General Public License as published by the Free
 Software Foundation; either version 2, or (at your option) any later
 version. See the file COPYING for more details.
 In addition, an exemption is given to allow Run Time Dynamic Linking of this code with any
closed source module that does not contain code covered by this licence.
 Only the source code to the module(s) containing the licenced code has to be released.
*/


#define PluginName "reborn's killing spree plugin"
#define PluginVersion "1.0"


void Plugin_Load();

void ZeroOutKills(int PlayerID);
void AddKill(int PlayerID, int Amount);
int GetKillingSpreeCount(int PlayerID);
```

```
class KillMessages : public ScriptImpClass
{
 void Created(GameObject *obj);
 void Killed(GameObject *obj, GameObject *Killer);
};

void Plugin_Unload();
```

Download the full source code here.
