how to use <insert here> in CSQC

Post Reply

Search this topic...

13 posts • Page **1** of **1**

how to use <insert here> in CSQC

by **gnounc** » Fri Jan 23, 2015 12:22 pm

gnounc

Posts: 424
Joined: Mon Apr 06, 2009 6:26 am

Theres been a fair amount of "how do I USE this function" going on lately, and I thought this thread would be a great way to clear some of that up!

So if you use CSQC and find yourself having to ask how to use a function because you cant find any doxx, run back here and drop off
what you learn!

To start things off shpuld and I will cover the most used basic functions, from there I hope this thread becomes the place to go for a real quick
"how do I?"

NOTE: for anyone posting, I'd love it if you guys threw the functions prototype up in a codeblock up top so that the readers can instantly see what arguments it takes.
thx 

Last edited by gnounc on Sun Jan 25, 2015 11:21 am, edited 2 times in total.

---

my git repository
gnounc's project graveyard

Re: how to use <insert here> in CSQC

by **gnounc** » Fri Jan 23, 2015 12:23 pm

gnounc

Posts: 424
Joined: Mon Apr 06, 2009 6:26 am

First things first.

CODE: SELECT ALL

```
        noref void(float vwidth, float vheight, float notmenu) CSQC_UpdateView; /* Calle
```

If you're going to be using csqc, you're probably going to be doing a bit of 2d drawing.
For that you will want to set up your screen size, and probably screen center globals.

In fte and dp both, you'll want to do this in CSQC_UpdateView();

In fte its as simple as declaring a global screen_size in your defs,
and feeding it the parameters of CSQC_UpdateView vwidth and vheight

screen_center then of course, would be half of that.
or

CODE: SELECT ALL

```
        screen_center = [screen_size_x/2, screen_size_y/2];
```

Darkplaces is slightly different. The parameters to CSQC_UpdateView in Darkplaces
give you virtual pixels. If you try to position things the way you would in fte using the method above, things
will look wonky.

Instead we use cvars.

CODE: SELECT ALL

```
        screen_size_x = cvar("vid_conwidth");
        screen_size_y = cvar("vid_conheight");
```

now you can

CODE: SELECT ALL

```
        drawstring(vector position, string text, vector size, vector rgb, float alpha, f
        drawpic(vector position, string pic, vector size, vector rgb, float alpha, optio
```

and

CODE: SELECT ALL

```
        drawfill(vector position, vector size, vector rgb, float alpha, optional float d
```

to the topleft corner with position vector '0 0'
or to the bottomleft corner with screen_size. (remembering of course to subtract the size of the image from its
position,
or it will be drawn offscreen)

Last edited by gnounc on Sat Jan 24, 2015 7:18 am, edited 1 time in total.

my git repository
gnounc's project graveyard

Re: how to use <insert here> in CSQC

by **Shpuld** » Fri Jan 23, 2015 1:14 pm

Shpuld

Posts: 106
Joined: Sat Feb 13, 2010 1:48 pm

First I'd like to point out that the stuff I'm gonna write is FTEQW ONLY. Darkplaces doesn't have multicasts or sendevents, in DP you can hack temp entities and sv_parseclientcommand.

Your game/mod is most likely going to need the client and server parts to work together, this requires information exchange between those 2, which in our case are of course CSQC and SSQC.
A lot of this stuff happens automatically in the engine, for example calling 'AddEntities' will make the client know about the entities that are sent by the server in the existing protocol already.

Now let's say you want to do some client<->server communication manually. CSQC provides good ways to do this.
For server -> client I generally like to use a multicast in SSQC and then handle it client-side in CSQC_Parse_Event. You get to decide exactly what you send and how it's interpreted. Very nice.

Example: We want to know client-side when we touch a thing serverside and we also want to get the things classname and origin for some reason.

In SSQC we would have a .touch function for the entity that looks like this:

CODE: SELECT ALL

```
const float EVENT_THINGTOUCH = 123; //you probably wanna keep lines like this in defs or

void() thing_touch =
{
        if(other.classname != "player")
                return;

        msg_entity = other; //make the client who touches this thing to be the receiver
        WriteByte(MSG_MULTICAST, SVC_CGAMEPACKET); //when sending a SVC_CGAMEPACKET we c
        WriteByte(MSG_MULTICAST, EVENT_THINGTOUCH); //it's smart to use the first byte o
        WriteString(MSG_MULTICAST, self.classname);
        WriteCoord(MSG_MULTICAST, self.origin_x); //sending vectors unfortunately requir
```

I'd like to elaborate a bit on the last line before moving on to the client-side part. The first argument is a position where the message comes from, which is not useful when doing a direct MULTICAST_ONE to a single client, but it's crucial when you wanna do stuff like sending the message to all clients in the same Potentially Visible Set (PVS) as our self.origin for example.
Here are all the MULTICAST_* options and short documentation by Spike himself:

CODE: SELECT ALL

```
const float MULTICAST_ALL = 0;  /* The multicast message is unreliably sent to all playe
const float MULTICAST_PHS = 1;  /* The multicast message is unreliably sent to only play
const float MULTICAST_PVS = 2;  /* The multicast message is unreliably sent to only play
const float MULTICAST_ONE = 6;  /* The multicast message is unreliably sent to the playe
const float MULTICAST_ALL_R = 3;        /* The multicast message is reliably sent to all
const float MULTICAST_PHS_R = 4;        /* The multicast message is reliably sent to onl
const float MULTICAST_PVS_R = 5;        /* The multicast message is reliably sent to onl
const float MULTICAST_ONE_R = 7;        /* The multicast message is reliably sent to the
```

Now let's get to the client-side handling for the event, it happens in CSQC_Parse_Event, a built-in entry point.

CODE: SELECT ALL

```
const float EVENT_THINGTOUCH = 123; //make sure this matches between client and server

//function is called whenever the client receives a SVC_CGAMEPACKET
noref void() CSQC_Parse_Event =
{
        local float first = readbyte(); //we read the first byte after the initial SVC_C
        if(first == EVENT_THINGTOUCH)
        {
                local vector pos;
                local string thing_msg;
                //we proceed to read all the information sent server-side, make sure you
                //IMPORTANT: make sure you read EVERYTHING you sent in the packet
```

Now this example isn't very useful but I hope you get the idea.

In the next post I'll tell how to tell the server to do things from the client using sendevent.

Last edited by Shpuld on Sat Jan 24, 2015 11:23 am, edited 1 time in total.

Re: how to use <insert here> in CSQC

by **Shpuld** » Fri Jan 23, 2015 1:43 pm

Shpuld

Posts: 106
Joined: Sat Feb 13, 2010 1:48 pm

Let's go through one way of doing client -> server communication.
Like mentioned in the last post I'll go through sendevent:

CODE: SELECT ALL

```
void(string evname, string evargs, ...) sendevent = #359; /*
                Invoke Cmd_evname_evargs in ssqc. evargs must be a string of initials re
```

So basically with this we can call our custom functions in SSQC from CSQC.

Example time: let's say we have Deus Ex -style numpads in our game, touching/using them opens a nice CSQC based numpad GUI on the screen, we input the code and want to let the server know what kind of bollocks we typed in.

CSQC:

CODE: SELECT ALL

```
//let's imagine this function gets called when clicks on the "Enter" button on the UI
void() numpad_send =
{
        local string code; //we somehow set this to what the user has inputted, not the
...
        sendevent("checkcode", "s", code);
        //we're sending only one arg here, if you want to send more and different types
        //sendevent("checkcode", "svf", code, somevector, somefloat);
        //of course this will call a different function in server-side because of the di
}
```

Now let's get into the SSQC:

CODE: SELECT ALL

```
//the name of this function is important, it has to match with the stuff we've done in C
//void(string code, vector somevector, float somefloat) Cmd_checkcode_svf =
//and this of course is a different function already

//self will also be the client that sent the event, so we can easily give him feedback h
void(string code) Cmd_checkcode_s =
```

```
{
        if(code != "1337")
                sprint(self, PRINT_HIGH, "Wrong code you dumbo!\n");
        else
        {
```

Quite simple, isn't it? Now you should have a good knowledge of how to do some basic information exchange between client and server with the UNLIMITED POWER of CSQC.

Re: how to use <insert here> in CSQC

by **gnounc** » Sat Jan 24, 2015 11:23 am

gnounc

Posts: 424
Joined: Mon Apr 06, 2009 6:26 am

Now on to some basic string handling.
strings are important for a variety of reasons, one of the most obvious, is constructing and deconstructing filepaths

most recently i've been using a custom save stuffcmd, adding things like the players name, mapname, kills and secrets right into the save filename
so i can parse it out later.

to concatenate strings together, we use strcat(); strcat takes up to 8 arguments, but all it does is smoosh the strings together.
on the ssqc side, ive been doing something like

CODE: SELECT ALL

```
        path = strcat(name, "\", mapname, "\", kills, "-", secrets);
```

to construct the save paths in the folders gnounc\e1m1 for example
with a file name of 5-4.sav

then I use tokenizebystring(); to get the information back out in csqc like so

CODE: SELECT ALL

```
        idx = tokenizebystring(path, "\", "-");
```

I believe tokenize does the same job, with all the obvious delimiters assumed, so use tokenize by string if you DONT want to split by \ or - or .
tokenize and tokenizebystring both return the number of items the string was split into

and populate argv();

CODE: SELECT ALL

```
name = argv(0);
mapname = argv(1);
etc...
```

or of course, you could just use a for loop, and the idx that was returned by tokenize

CODE: SELECT ALL

```
for(x = 0; x < idx - 1; x++)
{
        print(argv(idx);
}
```

really quickly, a few other useful string functions are

CODE: SELECT ALL

```
strlen(string s); which returns the number of characters in the string,
```

really useful for positioning string elements on screen.

CODE: SELECT ALL

```
float stringwidth(string text, float allowColorCodes, vector size) = #327; // ge
```

returns the size the string will appear on screen..horizontally obv

and lastly, if your temporary strings are getting garbled, being blank, or anything like that try strzone()
it returns a string that is protected from being freed.
which of course means you'll need to free it yourself when you're finished with ti

CODE: SELECT ALL

```
stringVar = strzone("i'd like to preserve this string so i can use it in five mi
```

and

CODE: SELECT ALL

```
strunzone(stringVar);
```

addendum: apparently tokenizebystring is called

CODE: SELECT ALL

```
        float(string s, string separator1, ...) tokenizebyseparator = #479;
```

in darkplaces.

..more to come

Last edited by gnounc on Sun Jan 25, 2015 9:46 am, edited 1 time in total.

my git repository
gnounc's project graveyard

Re: how to use <insert here> in CSQC

by **Shpuld** » Sat Jan 24, 2015 12:15 pm

Shpuld

Posts: 106
Joined: Sat Feb 13, 2010 1:48 pm

If you want to do manipulation in CSQC for certain entities that share a model, for example players, you can use deltalisten().

CODE: SELECT ALL

```
float(string modelname, float(float isnew) updatecallback, float flags) deltalisten = #3
                Specifies a per-modelindex callback to listen for engine-networking enti
                The various standard entity fields will be overwritten each frame before
```

I'll be using an example where we scale players up and down in CSQC.
EDIT: originally I said call deltalisten every frame, which is not required.
We call the deltalisten() in CSQC_Init, and for every entity with the model "progs/player.mdl" the specified listen function will be called every frame.

CODE: SELECT ALL

```
noref void(float apiver, string enginename, float enginever) CSQC_Init =
{
        ...
        deltalisten("progs/player.mdl", our_custom_listen, 0);
        ...
```

```
}
```

Now we need to define our_custom_listen. I just put it above the CSQC_UpdateView function.

CODE: SELECT ALL

```
float(float isnew) our_custom_listen =
{
        //this prints when you "see" an ent for the first time, or when it leaves PVS an
        if(isnew)
                print("I am new\n");

        self.scale = 1 + (0.5 * sin(time * 4));
        return 1;
};
```

Note, you want to make sure you have CSQC_Ent_Remove defined, you're gonna get null function calls otherwise whenever the object is removed from client, like when leaving PVS.
You probably want it to look like this for most cases:

CODE: SELECT ALL

```
void() CSQC_Ent_Remove =
{
        remove(self);
};
```

Re: how to use <insert here> in CSQC

by **gnounc** » Sun Jan 25, 2015 9:37 am

gnounc

Posts: 424
Joined: Mon Apr 06, 2009 6:26 am

Now it's time to talk about one of the two functions that inspired this thread.
I found

CODE: SELECT ALL

```
        void() calltimeofday = #231; /* Part of FTE_CALLTIMEOFDAY*/
```

after speaking with spike and finding a page on the linux system call, I had it all sorted out.

calltimeofday fills a global (I believe) and calling

CODE: SELECT ALL

```
        string(float uselocaltime, string format, ...) strftime = #478; /* Part of DP_QC
```

checks it, and returns the string in the format given as an argument

the applicable string formats can be found in the linux manpages, as strftime is a unix system call
http://linux.die.net/man/3/strftime

below is the call I've been using, remember you may need to strzone it, depending on how you use the string.

CODE: SELECT ALL

```
        calltimeofday();
        tod = strftime(1, "%Y-%m-%d %H-%M-%S");
```

---

my git repository
gnounc's project graveyard

Re: how to use <insert here> in CSQC

by **gnounc** » Sun Jan 25, 2015 9:45 am

gnounc

Posts: 424
Joined: Mon Apr 06, 2009 6:26 am

quick followup, as I believe calltimeofday is in fte and not darkplaces, you may desire to use

CODE: SELECT ALL

```
        float(float tmr) gettime = #519;
```

instead, which will work in either engine.

the list of available arguments to gettime are
(taken straight from darkplaces copy of csprogsdef.qc

CODE: SELECT ALL

```
        float GETTIME_FRAMESTART = 0; // time of start of frame
        float GETTIME_REALTIME = 1; // current time (may be OS specific)
        float GETTIME_HIRES = 2; // like REALTIME, but may reset between QC invocations
        float GETTIME_UPTIME = 3; // time since start of the engine
```

gettime fills the same constants as calltimeofday, so simply stftime it the same as before

CODE: SELECT ALL

```
        gettime(GETTIME_REALTIME);
        s_timeofday = strftime(1, "%Y-%m-%d_%H-%M-%S");
```

and done. remember to check your linux manpage for strftime use!

---

my git repository
gnounc's project graveyard

Re: how to use <insert here> in CSQC

by **Spike** » Sun Jan 25, 2015 10:12 am

Spike

Posts: 2914
Joined: Fri Nov 05, 2004 3:12 am
Location: UK
Contact:

CODE: SELECT ALL

```
void(float sec, float min, float hour, float day, float mon, float year, string datestri
{
//you're expected to write the stuff to some global and then just return so the caller c
print(sprintf("sec=%g, min=%g, hour=%g\nday=%g,month=%g,year=%g\n\"%s\"\n", sec, min, ho
//note that the date string will be something like "Sun Jan 26, 10:07:21 2015". Why it u
};
void() somefunction = //called from qc from elsewhere or whatever
{
calltimeofday(); //get the engine to invoke the timeofday function with the current date
}
```

obviously its kinda clumsy and not supported in dp, so just use strftime multiple times and hope the clock doesn't tick weirdly in the race condition between calls, or use substring to make it consistant or something.

FTE SVN Builds.

Re: how to use <insert here> in CSQC

by **gnounc** » Mon Jan 26, 2015 8:26 am

gnounc

Posts: 424
Joined: Mon Apr 06, 2009 6:26 am

Ok, now for some basic file handling.
Today we'll be working with

CODE: SELECT ALL

```
	float(string pattern, float caseinsensitive, float quiet) search_begin = #444; /
	float(float handle) search_getsize = #446; /* Part of DP_QC_FS_SEARCH*/
	string(float handle, float num) search_getfilename = #447; /* Part of DP_QC_FS_S
	void(float handle) search_end = #445; /* Part of DP_QC_FS_SEARCH*/
```

This is the last group of functions I have in my back pocket to do a showcase on. Its the other one I had to look up recently to figure out how to use.
I googled xonotic and nexuiz and search_begin to find a reference use.

search_begin's argument list is pretty self explainatory, with the exception that its first argument "pattern" is just filepath, except that it accepts wildcards */* and the like
search_begin returns a handle to the file list

search_getsize returns how many files were found and added to that list

search_getfilename takes the handle, and the index number of the list as arguments, and tells you the filename of that item

and lastly search_end takes the handle, and closes the file, freeing the memory.

put together it looks something like this. (note i'm using a nonstandard filepath.)

CODE: SELECT ALL

```
void printSaves()
{
	string path;
	path = "saves/*/*.fsv";

	float globhandle, idx, fileCount;
	globhandle = search_begin(path, TRUE, FALSE);		//get the file handle

	//return so we dont try to access garbage data
	if (globhandle <= 0)
		return;
```

Re: how to use <insert here> in CSQC

by **gnounc** » Mon Jan 26, 2015 9:17 am

gnounc

Posts: 424
Joined: Mon Apr 06, 2009 6:26 am

It dawned on me, that I cant very well show how to get a list of files
and not show how to use the most common reason for wanting a list of files.
Namely reading and writing to the files.
for this we will be using

CODE: SELECT ALL

```
float(string filename, float mode, optional float mmapminsize) fopen = #110; /* 
string(float fhandle) fgets = #112; /* Part of FRIK_FILE*/
void(float fhandle, string s, optional string s2, optional string s3, optional s
void(float fhandle) fclose = #111; /* Part of FRIK_FILE*/
```

first up, fopen();
The filename argument is obvious. so we'll move on to the mode.
Mode is read, write, append, that sort of thing, for the opened file.
fopen() returns a filehandle, as you might've guessed

CODE: SELECT ALL

```
const float FILE_READ = 0;
const float FILE_APPEND = 1;
const float FILE_WRITE = 2;
const float FILE_READNL = 4;
const float FILE_MMAP_READ = 5;
const float FILE_MMAP_RW = 6;
```

Write overwrites the file, append tacks your additions to the end of it.
not sure what memory map minimum size argument is for, but its optional, so I elect to ignore it.

fgets(), accepts the filehandle, and returns the first line of the file the first time you use it,
and the second line of the file the next, returning an empty string for a newline, and null for end of file.

fputs(), accepts the filehandle, and strings,
then shoves the strings specified in your argument into the file, remember that write overwrites, and append puts it at the bottom.

and fclose(), closes the file freeing the memory.

used altogether it would look something like this

CODE: SELECT ALL

```
	float idx;
	filehandle = fopen(myFile, FILE_READ);

	string line;
	while(line = fgets(filehandle))
	{
		print(line);
	}

	fclose(filehandle);
```

or..

CODE: SELECT ALL

```
	filehandle = fopen("playerList", FILE_WRITE);

	entity e;
	string line;
	for(e = find(world, classname, "player"); e ;e = find(e, classname, "player"))
	{
		fputs(filehandle, line);
	}

	fclose(filehandle);
```

---

my git repository
gnounc's project graveyard

Re: how to use <insert here> in CSQC

by **gnounc** » Sun Feb 08, 2015 12:55 am

gnounc

Posts: 424
Joined: Mon Apr 06, 2009 6:26 am

Today we'll be learning to use draw rotated pic.

CODE: SELECT ALL

```
	void(vector pivot, vector mins, vector maxs, string pic, vector rgb, float alpha
```

```
//Draws an image rotating at the pivot. To rotate in the center, use min
```

drawrotpic is a brand new function in fteqw, as such, you'll need the latest version (NOT STABLE VERSION..thats insanely outdated)
the latest versions of fte eschew traditional builtin numbers, any builtin you see registered as #0 requires the latest fte, and wont
work in darkplaces.

drawrotpic does not use position and size arguments like drawpic does,
instead the mins is the origin position, and the maxs is the bottom right corner
a good way to do this is to use a variable named img_pos or something along those lines for the mins,
and for maxs, use img_pos + img_size.

the pivot vector is also in screenspace, and not relative to the image.
As noted in the builtin description, find the center vector for the image,
and use that as the pivot if you want the image to rotate around its middle like a turntable

if you'd rather it rotate out like a pocketknife, use the origin (or any corner really) of the image as the pivot instead.

thus we end up with some code like this

CODE: SELECT ALL

```
vector img_pos, img_size, img_color;
string img_path;
float img_alpha, img_angles;

img_pos = '-32 -32';
img_size = '64 64';
img_color = '1 1 1';
img_path = "gfx/icon";
img_alpha = 1;
img_angles = time*100;
```

and the effect, is an image mostly offscreen on the topleft (only the bottom right quadrant is visible),
that rotates like a turntable.

my git repository
gnounc's project graveyard

Re: how to use <insert here> in CSQC

by **PrimalLove** » Sat Feb 28, 2015 2:23 am

PrimalLove

Posts: 14
Joined: Sun Oct 19, 2014 2:37 am

First want to start out by saying this is similar to Shplud's post about CSQC_Parse_Event. As far as I know this is what you would use for FTE instead of CSQC_Parse_TempEntity(). But with DP it seems we are stuck with this.. So I suppose think of this as a continuation of his post but more DP specific.

I recently had some misunderstandings about CSQC_Parse_TempEntity that was cleared up by Spike (Thanks again!) and I noticed little to no reference to this on the forums. Gnounc has a small explanation of it in his very helpful clean csqc but otherwise I rarely see it. Spike mentioned how evil it is but in any event I wanted to give hopefully a more thorough explanation of it its use with some examples. This will apply to both NQ or QW with slight differences in usage that will be noted. This will also assume you have all the appropriate and relevant builtin extensions defined.

CODE: SELECT ALL

```
// CSQC_Parse_TempEntity : Handles all temporary entity network data in the CSQC layer.
// You must ALWAYS first acquire the temporary ID, which is sent as a byte.
// Return value should be 1 if CSQC handled the temporary entity, otherwise return 0 to l
```

This does exactly what it sounds like. It takes write events from the server and handles them in CSQC.
So on the SSQC side we will look at one example in W_FireAxe. At the bottom of the function you will see something like this:

CODE: SELECT ALL

```
NetQuake
        {       // hit wall
                sound (self, CHAN_WEAPON, "player/axhit2.wav", 1, ATTN_NORM);
                WriteByte (MSG_BROADCAST, SVC_TEMPENTITY);
                WriteByte (MSG_BROADCAST, TE_GUNSHOT);
                WriteCoord (MSG_BROADCAST, org_x);
                WriteCoord (MSG_BROADCAST, org_y);
                WriteCoord (MSG_BROADCAST, org_z);
        }

QuakeWorld
```

You'll notice that QW has some differences. NQ does not support MSG_MULTICAST (Unless you use it in FTE : Credit Spike). MSG_BROADCAST is unreliable. In QW you can use MSG_MULTICAST which is better but still not perfect and its still an unreliable data stream(because it has to be). I should also note that you can still use MSG_BROADCAST in qw its just avoided. (Corrections made here thanks to Spike's input!)

The other thing to note is that the latter arugments in QW have two extra calls. The first is a count argument. In some cases (TE_BLOOD) the count argument will only accept 1 as a value. The last call is to let the engine know that this should only be sent to those that meet the requirements. In this case, only to players that can actually see the origin. This will additionally lower bandwidth usage and network traffic. Other than these differences, the code is essentially used the same way on both QW and NQ. (Corrections made here thanks to Spike's input! )
In QW particle() is not used so blood is also handled as a temp entity writebyte event. It looks like this:

CODE: SELECT ALL

```
/*
================
SpawnBlood
================
*/
void(vector org, float damage) SpawnBlood =
{
        WriteByte (MSG_MULTICAST, SVC_TEMPENTITY);
        WriteByte (MSG_MULTICAST, TE_BLOOD);
        WriteByte (MSG_MULTICAST, 1);
        WriteCoord (MSG_MULTICAST, org_x);
        WriteCoord (MSG_MULTICAST, org_y);
```

You can do this same thing in NQ source so that all of the blood calls will also be sent as temp entities. Here is an example:

CODE: SELECT ALL

```
void(vector org, vector vel, float damage) SpawnBlood =
{
        damage = damage*2;
        vel = vel*0.1;
        WriteByte (MSG_BROADCAST, SVC_TEMPENTITY);
        WriteByte (MSG_BROADCAST, TE_BLOOD);
        WriteCoord (MSG_BROADCAST, org_x);
        WriteCoord (MSG_BROADCAST, org_y);
        WriteCoord (MSG_BROADCAST, org_z);
        WriteCoord (MSG_BROADCAST, vel_x);
        WriteCoord (MSG_BROADCAST, vel_y);
        WriteCoord (MSG_BROADCAST, vel_z);
```

This is how many if not most of the particle effects are handled in standard NQ and QW game source. For the most part nothing to change on the SSQC side of things. Here are the constant float values you will need in both SSQC and

CSQC to use this builtin CSQC Temp Entity Parse function. (Note: this could be different depending on the source the important thing is they match on both sides)

CODE: SELECT ALL

```
NQ:
float   TE_SPIKE                = 0;
float   TE_SUPERSPIKE   = 1;
float   TE_GUNSHOT              = 2;
float   TE_EXPLOSION    = 3;
float   TE_TAREXPLOSION = 4;
float   TE_LIGHTNING1   = 5;
float   TE_LIGHTNING2   = 6;
float   TE_WIZSPIKE             = 7;
float   TE_KNIGHTSPIKE  = 8;
float   TE_LIGHTNING3   = 9;
float   TE_LAVASPLASH   = 10;
```

It may be helpful to place these in a shared.qc file so you can use one file for both ssqc and csqc defs. That is up to you as long as they are defined in both. Now in your main.qc or a new qc file you will want to place this to start: (NOTE: Please note ReadCoord is the same as readcoord in FTE. The builtin numbers are what is important so look to them for references if needed the rest that follows assumes DP. The method used by Shplud above is as far as I know standard practice for FTE using Use CSQC_Parse_Event and Multicasts.)

CODE: SELECT ALL

```
float CSQC_Parse_TempEntity()
{
        float Handled, damage;
        vector pos, vel;
        Handled  = TRUE;
        float TEID = ReadByte();
        switch(TEID)
        {
                case TE_SPIKE:
                        pos_x = ReadCoord();
                        pos_y = ReadCoord();
                        pos_z = ReadCoord();
```

The important thing to remember here is, if this function returns TRUE you must read all values sent by the writebyte event otherwise a crash will ensue. You also need to read them in the order they are sent. Lets take TE_GUNSHOT for an example:

CODE: SELECT ALL

```
                WriteByte (MSG_BROADCAST, SVC_TEMPENTITY); // This tells the server to se
                 WriteByte (MSG_BROADCAST, TE_GUNSHOT); // This tells the server the name
                 WriteCoord (MSG_BROADCAST, org_x); // The coords need to be read in this
                 WriteCoord (MSG_BROADCAST, org_y);
                 WriteCoord (MSG_BROADCAST, org_z);
```

You'll notice this only takes care of the particle event itself. If you also want it to replace the sound event as well you can also do this using pointsound(). So:

CODE: SELECT ALL

```
case TE_GUNSHOT:
                        pos_x = ReadCoord();
                        pos_y = ReadCoord();
                        pos_z = ReadCoord();
                        pointparticles(particleeffectnum("TE_GUNSHOT"), pos, '0 0 1', 1)
            pointsound(pos, "player/axhit2.wav", 1.0, ATTN_NORM);
                        Handled = TRUE;
                        break;
```

And on the ssqc side just comment out the sound() call. Pointsound does not rely on Channels but its attenuates differently from what I can tell from other types of sounds. You can adjust this in DP with snd_soundradius to get better results. You can use any custom effect you have made for any of these particles and put them as your particleeffectnum, etc. Anyway, hope this helps explain its usage a bit. If any errors please PM me or find me on qc# so I can correct any mistakes. Will credit you in EDIT. Thanks. 

EDIT: Spike also wanted me to mention to use sv_csqcdebug cvar in FTE.
Here is a brief explanation of the cvar:

> “
> This cvar changes fte's protocol in a subtle way. It adds size counts on a per-entity basis. This makes it easier to find which entity sizes are read/written incorrectly. This is highly FTE specific, and this cvar should only ever be set by csqc modders (or their testers). This will allow the csqc to remain running and also print which entities were not read correctly.

Post Reply

13 posts • Page **1** of **1**

Return to “CSQC Programming”

Jump to

Board index

Delete cookies All times are UTC