Doomsday Engine - Bug #691

on screen text error

2009-05-20 11:25 - eunbolt

Status:	Closed	**Start date:**	2009-05-20
Priority:	Urgent	**% Done:**	100%
Assignee:	skyjake		
Category:			
Target version:	1.9.0-beta6		

Description

When entering a cheat code the text response is clipped one char short
e.g. idclip the cheat notification "no clipping on" appears as "no clipping o"
does not affect item pickup pick up notifications

Labels: System

History

#1 - 2009-05-23 00:14 - danij

Seems to work fine on my setup with both 1.9.0-beta6.2 and the 1.9.0-beta6 branch.

I haven't tested the trunk as the player message log has changed considerably since the initial beta6 release.

#2 - 2009-06-13 08:39 - skyjake

The problem seems to be that the (nonstandard) snprintf() function is defined a bit differently in Windows and Unix. The major difference is that Windows will only terminate the string with a null character if there is space in the buffer, while Unix will always terminate the string.

I think we should define our own snprintf() wrapper that hides this platform specific difference. Always terminating the string definitely is the more secure way; I think that should be the behavior on all platforms.

#3 - 2009-06-13 08:55 - danij

Agreed. That sounds like the best course of action to me.

#4 - 2009-06-13 10:18 - skyjake

I committed the functions in rev 6668. Danij, I trust you can finish the implementation on Windows?

However, if vsnprintf() already is defined by Windows, I recommend renaming the function to win32_vsnprintf() and adding a

#define vsnprintf win32_vsnprintf

to dd_share.h.

#5 - 2009-06-13 21:37 - danij

Yeah I can finish this (I expect I'll be commiting tomorrow as I don't feel much like coding right now).

#6 - 2009-06-14 10:19 - danij

Looking at the function header comments you've added for snprintf and vsnprintf in dd_share.h I would like to clarify something before I begin on these:

"@return The number of characters that would have been written to the output buffer if the size was unlimited."

This does not appear to be standard behavior.

See here: http://linux.die.net/man/3/snprintf

#7 - 2009-06-14 10:34 - skyjake

Well, from that man page I interpret that the C99 standard defines the behavior like that: the return value is the number of characters written to the output buffer (not including the null char), or if the output size was too small, the number of characters that would have been written if it was unlimited.

I'm not sure if there's a Windows function that returns that unlimited output length, though.

For our purposes we can define the return value differently, though, if it's more convenient. E.g., returning -1 if truncation occurred, and otherwise the number of characters written (not including the null).

#8 - 2009-06-14 10:42 - skyjake

Here's the definition from the C99 (draft) for vsnprintf(): [n is the size of the output buffer]

"The vsnprintf function returns the number of characters that would have been written had n been sufficiently large, not counting the terminating null character, or a negative value if an encoding error occurred. Thus, the null-terminated output has been completely written if and only if the returned value is nonnegative and less than n."

#9 - 2009-06-14 10:46 - danij

Here is the definition for vsnprintf from MSDN:

"vsnprintf, _vsnprintf, and _vsnwprintf return the number of characters written if the number of characters to write is less than or equal to count; if the number of characters to write is greater than count, these functions return -1 indicating that output has been truncated. The return value does not include the terminating null, if one is written."

It appears to me that the Windows varient is behaving the same as the C99 standard other than the fact it does not always terminate the given character string.

#10 - 2009-06-14 10:51 - skyjake

I disagree. The difference is that the Windows version will return -1 if truncation occurs. The C99 returns -1 only if an **encoding** error occurs (as in there's some weird characters in the parameters), and if truncation occurs, it returns the untruncated length of the output string (not -1).

#11 - 2009-06-14 11:05 - danij

Clearly I need another coffee. On re-reading the definitions again, yes I agree that the Windows varient is indeed behaving quite differently to the C99.

I don't think I can get the unlimited length of the encoded string (to be written) unless we implement our own version of vsnprintf (urgh).

#12 - 2009-06-14 11:13 - skyjake

On the bright side, I don't think the return value is used anywhere in the engine. It is vastly more important that the null-terminating behavior is the same. We could make it so (and document it with a @note) that on Win32 the return value is -1 on truncation, while in Unix it is the unlimited length of the output.

#13 - 2009-06-14 11:51 - danij

That sounds OK to me though we will naturally have to ensure we don't use the return value or where we do, we account for the differences.

I'm currently looking for a way to declare our vsnprintf wrapper without copying the whole attribute list from the MSVC definition for the win32, vsnprintf:

warning C4985: 'vsnprintf': attributes not present on previous declaration.

#14 - 2009-06-14 12:04 - skyjake

How about that earlier suggestion about naming it win32_vsnprintf() and using a #define from vsnprintf to that?

#15 - 2009-06-14 12:08 - danij

That is what I've been trying to do but I'm hitting a brick wall because of the way our headers are setup (there are many instances where stdio.h is included after dd_share.h and so the macro rename of vsnprintf is being applied). I have also tried #undef ing vsnprintf before then including stdio.h before the definition of win32_vsnprintf but that didn't work either (a reserved C function name maybe?).

#16 - 2009-06-14 12:12 - skyjake

Well, one option is to make it something like dd_vsnprintf() and use find/replace on all the instances of where it's called (instead of a #define). That would actually allow us to define the return value however suits us best and achieve the same behavior on all platforms.

#17 - 2009-06-14 12:14 - danij

I think that is going to be our best option. Either that or sort out our headers...

#18 - 2009-06-14 12:16 - skyjake

I think we should go the dd_vsnprintf() route because otherwise the return value thing is a bit ugly and error prone (should anyone ever use the return value).

#19 - 2009-06-14 12:21 - danij

Ok, will do.

#20 - 2009-06-14 12:51 - danij

There is one final decision to make regarding these; do we want an engine-side, exported, public implementation of dd_vsnprintf or do we put it in some public source file and let each lib compile their own? I presume the former.

#21 - 2009-06-14 13:00 - skyjake

An exported public implementation of dd_vsprintf() sounds fine.

I have one comment about your commit:

#ifdef WIN32
// Always terminate.
str[size - 1] = 0;
return result;
#else
return result > size? -1 : size;
+#endif

Since vsnprintf() returns the number of characters without the null, if result == size then it means that the last character was replaced by the null, causing truncation. So shouldn't that read:

```
return result >= size? -1 : size;
```

#22 - 2009-06-14 13:02 - skyjake

Oh, and you should be able to see the HUD log message last character truncation problem now on Windows as well.

#23 - 2009-06-14 13:04 - danij

> I have one comment about your commit... ...Since vsnprintf() returns the number of characters without the null, if result == size then it means that the last character was replaced by the null, causing truncation.

Quite right, I'll fix it.

#24 - 2009-06-14 13:43 - danij

Fixed in svn for 1.9.0-beta6.4