From: tinycad@yahoogroups.co.uk on behalf of Kaz Kylheku
To: tinycad@yahoogroups.co.uk
Subject: RE: [tinycad] Build errors trying to build TinyCad: calling nonexistent MFC methods.
Date: Wednesday, December 05, 2012 2:19:45 PM

On Wed, 5 Dec 2012 01:27:57 -0500, "Don Lucas"
<Don.Lucas@OrikonSolutions.com> wrote:
> Hi Kaz –
>
> You have made it further than I ever did with MFC 4.2 and the Express
> versions of studio. If you can figure out how to work around the
> remaining issues, I will make an attempt to include your changes in
> the TinyCad source. If you would like to become a member of the
> development team, that can be accommodated, also ;-)

Hi Don,

I've pared down the compile errors quite a bit, down to four errors,
all related to the HTML help stuff.

I made a CTokenizingString class
derived from CString, which has Tokenize. If your MFC version is above
0x0600 then it just typedefs to CString.

```
#if _MFC_VER <= 0x0600

CString CTokenizingString::Tokenize(const TCHAR *set, int &pos)
{
const TCHAR *str = operator const TCHAR *();
size_t skip = _tcsspn(str + pos, set);
size_t take = _tcscspn(str + pos + skip, set);

CString ret(str + pos + skip, take);

pos += skip + take;

if (str[pos] == 0)
pos = -1;

return ret;
}

#endif
```

I suppose I could have just done (TCHAR *) *this, instead of the fancy
operator calls. :)

The DrawTextEx issue is just a matter of lacking a trivial wrapper for
a new Win32 function, and can be handled like this:

```
#if _MFC_VER > 0x0600
m_pDC->DrawTextEx(t, q,
DT_EDITCONTROL|DT_TOP|DT_LEFT|DT_EXPANDTABS|DT_TABSTOP|DT_WORDBREAK,
lpDTParams);
#else
::DrawTextEx(m_pDC->m_hDC, const_cast<TCHAR *>(t), _tcslen(t), q,
DT_EDITCONTROL|DT_TOP|DT_LEFT|DT_EXPANDTABS|DT_TABSTOP|DT_WORDBREAK,
lpDTParams);
#endif
```

The fallback case in the command line parsing virtual function can just
do this, I think. None of this is tested yet:

```
else
{
ATLTRACE2("CTinyCadCommandLineInfo::ParseParam(): Found non-TinyCAD
command line option \"%S\"\n", pszParam);
m_eLastFlag = TCFlag_Unknown;
CCommandLineInfo::ParseParam(pszParam, bFlag, bLast); //Not one of
TinyCad's - let the regular CCommandLineInfo object parse it.
}
```

You see? Instead of trying to pick one of the protected helpers to call, let's just call the superclass implementation of ParseParam, handing down the same parameters. So MFC's ParseParam can then do whatever it has to using whatever calls to its helpers.

I suspect that those protected functions in CCommandLineInfo, namely ParseParamFlag, ParseParamNotFlag and ParseLast, are part of its implementation, not intended to be used. There is even a comment which says "Implementation". And they are not documented on MSDN:

```
// Implementation
protected:
void ParseParamFlag(const char* pszParam);
void ParseParamNotFlag(const TCHAR* pszParam);
#ifdef _UNICODE
void ParseParamNotFlag(const char* pszParam);
#endif
void ParseLast(BOOL bLast);
```

This is probably why the TCHAR version of ParseParamFlag is omitted. The class doesn't need it internally for its implementation so they didn't bother.

Cheers ...

Reply to **sender** | Reply to **group** | Reply **via web post** | **Start a new topic**
Messages in this topic (**5**)

RECENT ACTIVITY: **New Members 5** |
Visit Your Group

Switch to: Text-Only, Daily Digest • Unsubscribe • Terms of Use