OpenGL Build Touch Source Code

Disclaimer:

Okay then, this is the deal. I don't give a flying fuck if you like how I program or not. If you can't understand what the hell is going on, then don't complain. If you do complain, you can go and screw yourself. If you screw something up, or someone else does, it ain't my fault, and I don't give a rats.

I have selected A4 for all of the pages, for some of them landscape would have been better, so you could actually see the lines as they are, instead of them wrapping back to the start again, but it doesn't matter anyway, and I really don't care.

This was all done under Borland C++ v5.02, using the Win32 compiler. OpenGL is a registered trademark of the Sun Corporation. Borland is a registered trademark of the Inprise Corporation. If it doesn't compile under your compiler, then I suggest looking at the error messages to find what the problem is. It usually states what you stuffed up in.

Some of the info used to create this program came from other resources (ie; things like the map format, art format, and the sector effector values cam from either buildhlp.exe or artform.txt). These were then ported or administered by me.

You can use whatever you find in this file to your own advantage, or disadvantage if you stuff up somewhere along the way, or already have. I wrote it, the copyright is mine, but if you learn something from it, then good for you. Just keep out of my way, and I will be happy.

There is no structure to this program. I wrote it over a period of time, a block at a time (updates) and have lost track of what the hell most of the original stuff did. It just sits there and works, so why change it? If it is happy, leave it alone, or it will bite you in the arse.

The reason why I distributed this is for two reasons:

1) you lot asked (no, not all of you, only a few)
2) I am being a prick, and have chosen to use the Adobe PDF with a few of the security features enabled. There are work arounds of course, but this is tedious, so if you do bypass it, you deserve all the info in this document for your efforts. Of course you could just retype it.

If you ask really nicely, I may disable to the security features and send you a copy, or maybe even upload it, but don't bet your buttons on it.

If you hate the fact I barely comment my code, you can go and shove it somewhere. If you hate my style of programming, you can go and stick that thought up your arse too. All I have to say is that if you have any problems with this document, then bugger off.

So what all this boils down to is what it started as.

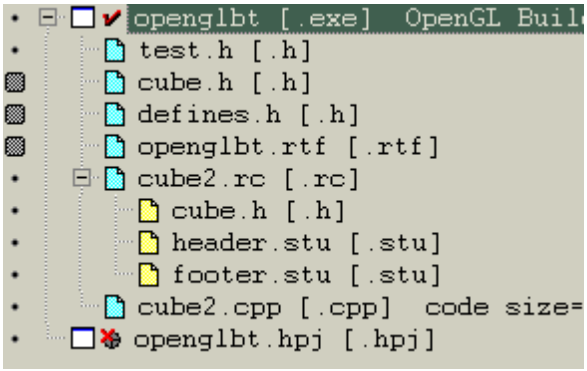
If anything has gone wrong, is going wrong, will go wrong or may go wrong with any thing associated with or without me or anything I do, I take no responsibility.
It's all your fault, just as your girlfriend told you.

OpenGL Build Touch Source Code

Thanks to:

Adobe - for making the PDF
Borland - for the greatest compiler (in my opinion) around. (well under DOS and Windows anyways)
Sun - for OpenGL
3DRealms - for making Duke3D
Dukeworld - for hosting the OpenGLBT site
Lotus - great apps, what I am typing this message in at the moment.

Info about the source:



This is the basic structure of which file goes where
All files named *.stu (or *.exe) are not included as they are binaries.
Other files will be included in this file, organised in the order of the filenames in the image.

OpenGL Build Touch Source Code

test.h

```
//    sprintf(feedback,"Including %s",include_name);SetDlgItemText(hwndDlg,IDC_STATUS,feedback);
char driv1[3];
char dir1[260];
char name1[256];
char ext1[256];
char feedback[260];
FILE *in,*out;
char bigarray[1048576];
char temparray[1048576];
long array_size;
char include_name[2048];
char include_num[128];
//void remove_comments(void);
//char insert_includes(void);
//char remove_deadspace(void);
//char remove_deadspace2(void);
long lastinclude=0;

void remove_comments(HWND hwndDlg)
{
 static long a;
 static long array_temp=0;
 array_temp=lastinclude;
 sprintf(feedback,"Removing comments");SetDlgItemText(hwndDlg,IDC_STATUS,feedback);
 for(a=lastinclude;a<array_size;a++)
 {
  if(bigarray[a]=='/'&&bigarray[a+1]=='/')
  {
   while(!(bigarray[a]==13&&bigarray[a+1]==10))
    a++;
  }
  else if(bigarray[a]=='/'&&bigarray[a+1]=='*')
  {
   while(!(bigarray[a]=='*'&&bigarray[a+1]=='/'))
   a++;
   a++;
  }
  else
   temparray[array_temp++]=bigarray[a];
 }
 array_size=array_temp;
 #ifdef USE_MEM
  memcpy(bigarray,temparray,array_size);
 #else
  for(a=0;a<array_size;a++)
  bigarray[a]=temparray[a];
 #endif
}


char insert_includes(HWND hwndDlg)
{
 FILE *in2;
 static long a,b,c;
 static long array_temp;
 char trail[260];
 array_temp=lastinclude;
 sprintf(feedback,"Searching for files to include");SetDlgItemText(hwndDlg,IDC_STATUS,feedback);
 for(a=lastinclude;a<array_size;a++)
 {
   if(
   (a==0||bigarray[a-1]==32||bigarray[a-1]==10)
   &&bigarray[a]=='i'
   &&bigarray[a+1]=='n'
   &&bigarray[a+2]=='c'
   &&bigarray[a+3]=='l'
   &&bigarray[a+4]=='u'
   &&bigarray[a+5]=='d'
   &&bigarray[a+6]=='e'
   &&(bigarray[a+7]==32||bigarray[a+7]==13)
   )
   {
    lastinclude=a;
    a+=8;
    while(bigarray[a]==32||bigarray[a]==10||bigarray[a]==13)
```

```
     a++;
    b=0;
    while(bigarray[a]!=32&&bigarray[a]!=13)
    {
     include_name[b]=bigarray[a];
     a++;b++;
    }
    include_name[b]=0;
    sprintf(feedback,"Including %s",include_name);SetDlgItemText(hwndDlg,IDC_STATUS,feedback);
//    in2=fopen(include_name,"rb");
sprintf(trail,"%s%s%s",driv1,dir1,include_name);

//if((in2=fopen(include_name,"rb"))==NULL)
if((in2=fopen(trail,"rb"))==NULL)
{
   MessageBox(NULL, "Can't open include file", "Feedback", MB_OK);
   return 3;
}

    fseek(in2,0,2);
    c=ftell(in2);
    fseek(in2,0,0);
    for(b=0;b<c;b++)
    {
     temparray[array_temp]=fgetc(in2);
     array_temp++;
    }
    while(a<array_size)
    {
     temparray[array_temp]=bigarray[a];
     array_temp++;a++;
    }
    fclose(in2);
    sprintf(feedback,"Including %s",include_name);SetDlgItemText(hwndDlg,IDC_STATUS,feedback);
    array_size=array_temp;
 #ifdef USE_MEM
  memcpy(bigarray,temparray,array_size);
 #else
  for(a=0;a<array_size;a++)
  bigarray[a]=temparray[a];
 #endif
    return 1;
   }
   else
   {
    temparray[array_temp]=bigarray[a];
    array_temp++;
   }
 }
 return 0;
}

char change_defs(HWND hwndDlg)
{
 static char r,f;
 static long lastloc=0;
 static long a,b,c,d;
 static long array_temp=lastloc;
 f=0;
 array_temp=lastloc;
 sprintf(feedback,"Searching for definitions");SetDlgItemText(hwndDlg,IDC_STATUS,feedback);
 for(a=lastloc;a<array_size;a++)
 {
   if((a==0||bigarray[a-1]==32||bigarray[a-1]==10)&&bigarray[a]=='d'&&bigarray[a+1]=='e'&&bigarray[
a+2]=='f'
   &&bigarray[a+3]=='i'&&bigarray[a+4]=='n'&&bigarray[a+5]=='e'&&(bigarray[a+6]==32||bigarray[a+6]=
=13))
   {
    lastloc=a;
    a+=7;
    while(bigarray[a]==32||bigarray[a]==10||bigarray[a]==13)
     a++;
    b=0;
    while(bigarray[a]!=32&&bigarray[a]!=13&&bigarray[a]!=10)
     include_name[b++]=bigarray[a++];
    include_name[b]=0;
```

```
    while(bigarray[a]==32||bigarray[a]==10||bigarray[a]==13)
     a++;
    d=0;
    while(bigarray[a]!=32&&bigarray[a]!=13&&bigarray[a]!=10)
    {
     include_num[d++]=bigarray[a++];
    }
    include_num[d]=0;
    sprintf(feedback,"Replacing %s with %s",include_name,include_num);SetDlgItemText(hwndDlg,IDC_ST
ATUS,feedback);



    while(a<array_size)
    {
  if((bigarray[a-1]==32||bigarray[a-1]==10)&&bigarray[a]=='d'

  )
  if(
   (
      bigarray[a+1]=='e'
   && bigarray[a+2]=='f'
   && bigarray[a+3]=='i'
   && bigarray[a+4]=='n'
   && bigarray[a+5]=='e'
   )
   &&
   (
    (
       bigarray[a+6]=='q'
    && bigarray[a+7]=='u'
    && bigarray[a+8]=='o'
    && bigarray[a+9]=='t'
    &&bigarray[a+10]=='e'
    &&(bigarray[a+11]==32||bigarray[a+11]==13))
    ||
    (
       bigarray[a+6]=='l'
    && bigarray[a+7]=='e'
    && bigarray[a+8]=='v'
    && bigarray[a+9]=='e'
    &&bigarray[a+10]=='l'
    &&bigarray[a+11]=='n'
    &&bigarray[a+12]=='a'
    &&bigarray[a+13]=='m'
    &&bigarray[a+14]=='e'
    &&(bigarray[a+15]==32||bigarray[a+15]==13))
   )
  )
 {
  while(bigarray[a]!=10)
   temparray[array_temp++]=bigarray[a++];
  temparray[array_temp++]=10;
 }

    if(bigarray[a]==include_name[0])
    {
     if((a==0||bigarray[a-1]==32||bigarray[a-1]==10)&&bigarray[a]==include_name[0]&&(bigarray[a+b]=
=32||bigarray[a+b]==13||bigarray[a+b]==10))
     {
      c=0;
      r=1;
      while(c<b&&r==1)
      {
       temparray[array_temp+c]=bigarray[a+c];
       if(bigarray[a+c]!=include_name[c])
       {
        array_temp+=c;
        a+=c;
        r=0;
       }
       c++;
      }
      if(r)
       if(bigarray[a+b]!=32&&bigarray[a+b]!=13&&bigarray[a+b]!=10)
        r=0;
```

```
      if(r)
      {
       f++;
       for(c=0;c<d;c++)
       {
        temparray[array_temp++]=include_num[c];
        a++;
       }
       a--;
       while(bigarray[a]!=32&&bigarray[a]!=10)a++;
      }
     }
    }
     temparray[array_temp++]=bigarray[a++];
    }
    array_size=array_temp;
    for(a=lastloc;a<array_size;a++)
     bigarray[a]=temparray[a];
    return 1;
   }
   else
   {
    temparray[array_temp]=bigarray[a];
    array_temp++;
   }
 }
 return 0;
}

char remove_deadspace2(HWND hwndDlg)
{
 static long a;
 static long array_temp=0;
 array_temp=0;
 sprintf(feedback,"Cleaning (phase 2)");SetDlgItemText(hwndDlg,IDC_STATUS,feedback);

 for(a=0;a<array_size;a++)
  if(bigarray[a]==13)bigarray[a]=10;


 for(a=0;a<array_size;a++)
 {
  if((array_temp!=0&&temparray[array_temp-1]==10)||array_temp==0)
   while(bigarray[a]==32||bigarray[a]==10)
    a++;
 temparray[array_temp++]=bigarray[a];
 }


 array_size=array_temp;
 #ifdef USE_MEM
  memcpy(bigarray,temparray,array_size);
 #else
  for(a=0;a<array_size;a++)
  bigarray[a]=temparray[a];
 #endif
 array_temp=0;
 for(a=0;a<array_size;a++)
  if(bigarray[a]==10)
  {
   temparray[array_temp++]=13;
   temparray[array_temp++]=10;
  }
  else
   temparray[array_temp++]=bigarray[a];

 array_size=array_temp;
 #ifdef USE_MEM
  memcpy(bigarray,temparray,array_size);
 #else
  for(a=0;a<array_size;a++)
  bigarray[a]=temparray[a];
 #endif
 return 1;
}
```

```
char remove_deadspace(HWND hwndDlg)
{
 char v=0;
 static long a;
 static long array_temp=0;
 array_temp=0;
    sprintf(feedback,"Cleaning (phase 1)");SetDlgItemText(hwndDlg,IDC_STATUS,feedback);

 while(bigarray[a]==32||bigarray[a]==10||bigarray[a]==13)a++;
 while(a<array_size)
 {
  if((a==0||bigarray[a-1]==32||bigarray[a-1]==10||bigarray[a-1]==13)
  &&   bigarray[a]=='d'
  && bigarray[a+1]=='e'
  && bigarray[a+2]=='f'
  && bigarray[a+3]=='i'
  && bigarray[a+4]=='n'
  && bigarray[a+5]=='e'
  &&
  (
   (  bigarray[a+6]=='q'
   && bigarray[a+7]=='u'
   && bigarray[a+8]=='o'
   && bigarray[a+9]=='t'
   &&bigarray[a+10]=='e'
   &&(bigarray[a+11]==32||bigarray[a+11]==13||bigarray[a+11]==10))
   ||
   (  bigarray[a+6]=='s'
   && bigarray[a+7]=='o'
   && bigarray[a+8]=='u'
   && bigarray[a+9]=='n'
   &&bigarray[a+10]=='d'
   &&(bigarray[a+11]==32||bigarray[a+11]==13||bigarray[a+11]==10))
   ||
   (  bigarray[a+6]=='l'
   && bigarray[a+7]=='e'
   && bigarray[a+8]=='v'
   && bigarray[a+9]=='e'
   &&bigarray[a+10]=='l'
   &&bigarray[a+11]=='n'
   &&bigarray[a+12]=='a'
   &&bigarray[a+13]=='m'
   &&bigarray[a+14]=='e'
   &&(bigarray[a+15]==32||bigarray[a+15]==13||bigarray[a+15]==10))
  )
 )
 {
  if(a!=0)
  {
   bigarray[a-1]==10;
   temparray[a-1]==10;
  }
  if(bigarray[a+6]=='s')v='s';
  else if(bigarray[a+6]=='q')v='q';
  else v='l';

  if(v=='q')
  {
   while(bigarray[a]!=32)
   {
    temparray[array_temp]=bigarray[a];
    a++;
    array_temp++;
   }
   temparray[array_temp++]=bigarray[a++];
   while(bigarray[a]==32)a++;
   while(bigarray[a]!=32)
   {
    temparray[array_temp]=bigarray[a];
    a++;
    array_temp++;
   }
   temparray[array_temp++]=bigarray[a++];
   while(bigarray[a]==32)a++;
  }
```

```
 if(v=='l')
 {
  while(bigarray[a]!=32)
  {
   temparray[array_temp]=bigarray[a];
   a++;
   array_temp++;
  }
  temparray[array_temp++]=bigarray[a++];
  while(bigarray[a]==32)a++;
  while(bigarray[a]!=32)
  {
   temparray[array_temp]=bigarray[a];
   a++;
   array_temp++;
  }
  temparray[array_temp++]=bigarray[a++];
  while(bigarray[a]==32)a++;
  while(bigarray[a]!=32)
  {
   temparray[array_temp]=bigarray[a];
   a++;
   array_temp++;
  }
  temparray[array_temp++]=bigarray[a++];
  while(bigarray[a]==32)a++;
  while(bigarray[a]!=32)
  {
   temparray[array_temp]=bigarray[a];
   a++;
   array_temp++;
  }
  temparray[array_temp++]=bigarray[a++];
  while(bigarray[a]==32)a++;
  while(bigarray[a]!=32)
  {
   temparray[array_temp]=bigarray[a];
   a++;
   array_temp++;
  }
  temparray[array_temp++]=bigarray[a++];
  while(bigarray[a]==32)a++;
  while(bigarray[a]!=32)
  {
   temparray[array_temp]=bigarray[a];
   a++;
   array_temp++;
  }
  temparray[array_temp++]=bigarray[a++];
  while(bigarray[a]==32)a++;
 }

/*
 if(v=='s')
 {
  while(bigarray[a]!=32)
  {
   temparray[array_temp]=bigarray[a];
   a++;
   array_temp++;
  }
  temparray[array_temp++]=bigarray[a++];
  while(bigarray[a]==32)a++;
  while(bigarray[a]!=32)
  {
   temparray[array_temp]=bigarray[a];
   a++;
   array_temp++;
  }

  temparray[array_temp++]=bigarray[a++];
  while(bigarray[a]==32)a++;
  while(bigarray[a]!=32)
  {
   temparray[array_temp]=bigarray[a];
   a++;
```

```
    array_temp++;
   }
   temparray[array_temp++]=bigarray[a++];
   while(bigarray[a]==32)a++;
   while(bigarray[a]!=32)
   {
    temparray[array_temp]=bigarray[a];
    a++;
    array_temp++;
   }
   temparray[array_temp++]=bigarray[a++];
   while(bigarray[a]==32)a++;
   while(bigarray[a]!=32)
   {
    temparray[array_temp]=bigarray[a];
    a++;
    array_temp++;
   }
   temparray[array_temp++]=bigarray[a++];
   while(bigarray[a]==32)a++;
   while(bigarray[a]!=32)
   {
    temparray[array_temp]=bigarray[a];
    a++;
    array_temp++;
   }
   temparray[array_temp++]=bigarray[a++];
   while(bigarray[a]==32)a++;
   while(bigarray[a]!=32)
   {
    temparray[array_temp]=bigarray[a];
    a++;
    array_temp++;
   }
   temparray[array_temp++]=bigarray[a++];
   while(bigarray[a]==32)a++;
  }
*/
  while(bigarray[a]!=10&&bigarray[a]!=13)
  {

   if(v=='s')
   {
    temparray[array_temp]=bigarray[a];
    if(bigarray[a]==32)while(bigarray[a]==32)a++;
    else
    a++;
    array_temp++;
   }
   else

   {
    temparray[array_temp]=bigarray[a];
    a++;
    array_temp++;
   }
  }
  temparray[array_temp]=10;
  array_temp++;
  a++;
 }
 else
 {

  if(array_temp!=0&&(temparray[array_temp-1]==32))
  while(bigarray[a]==32||bigarray[a]==10||bigarray[a]==13)a++;
  if(bigarray[a]==13||bigarray[a]==10)
  {
   while(bigarray[a]==10||bigarray[a]==13||bigarray[a]==32)a++;
    if(temparray[array_temp-1]!=32)
     temparray[array_temp]=32;
    else
     array_temp--;
  }
  else
  {
```

```
//*/
   temparray[array_temp]=bigarray[a];
   a++;
  }
  if(
  (temparray[a-5]==32||temparray[a-5]==13||temparray[a-5]==10)&&
  temparray[a-4]=='e'&&
  temparray[a-3]=='n'&&
  temparray[a-2]=='d'&&
  (
  temparray[a-1]=='a'||temparray[a-1]=='s'
  )&&
  (temparray[a]==32||temparray[a]==13||temparray[a]==10)
  )
  {
  temparray[++array_temp]=13;
  temparray[++array_temp]=10;
  }
  array_temp++;
 }
}
 array_size=array_temp;
 #ifdef USE_MEM
  memcpy(bigarray,temparray,array_size);
 #else
  for(a=0;a<array_size;a++)
  bigarray[a]=temparray[a];
 #endif
 return 1;
}
```

OpenGL Build Touch Source Code

cube.h

```
#define ABOUT  8
#define IDC_STATICTEXT57   104
#define IDC_STATICBITMAP1  102
#define IDC_STATICBITMAP2  103
#define TEST   7
#define RCDATA_2  2
#define RCDATA_1  1
#define STUB   6
#define IDC_STATICTEXT44   101
#define DLG_GROUP 5
#define IDC_LISTBOX3 101
#define IDC_GROUP 101
#define IDC_LISTBOX4 102
#define IDC_GROUPSIZE   102
#define IDC_STATICTEXT40   102
#define IDD_DIALOG2  1002
#define IDD_DIALOG1  1001
#define IDC_TREEVIEW1   1101
#define IDB_BITMAP1  1001
#define szAppName "SimplePaint"
#define LINE 1
#define ELLIPSE 2
#define RECTANGLE 3

#define MID_QUIT 100
#define MID_LINE 201
#define MID_ELLIPSE 202
#define MID_RECTANGLE 203
#define MID_THIN 301
#define MID_REGULAR 302
#define MID_THICK 303
#define MID_RED 304
#define MID_GREEN 305
#define MID_BLACK 306


#define OpenGLBT  1
#define CM_OPTIMISE  11854
#define CM_FILEITEM5 18873
#define CM_Patch  11855
#define CM_PATCH  11854
#define CM_FILEITEM4 18873
#define IDM_HELP  11847
#define CM_POPUPITEM8   18873
#define CM_POPUPITEM7   18873
#define CM_Group  11846
#define CM_GROUP  11854
#define CM_External  11853
#define CM_Test   11852
#define CM_POPUPITEM6   18874
#define CM_POPUPITEM5   18873
#define CM_ABOUT  11851
#define CM_EXIT   11850
#define CM_Sprite 11849
#define CM_POPUPITEM4   18873
#define CM_POPUPITEM3   18873
#define CM_WallSect  11848
#define CM_POPUPITEM2   18873
#define IDM_TILE  11847
#define CM_FILEITEM3 18873
#define IDM_GROUP 11846
#define CM_FILEITEM2 18873
#define CM_GROUPITEM1   18873
#define IDM_GOPEN 11846
#define CM_POPUPITEM1   18873
#define IDM_SAVEAS   11845
#define IDM_SAVE  11844
#define CM_FILEITEM1 18873
#define IDM_NEW   11843
#define IDM_OPEN  11842
#define IDI_ICON2 2
#define DLG_EXTERNAL 4
#define DLG_TEST  3
#define IDC_BUTTON15 101
#define IDC_R_CANCEL 304
#define IDC_RADIOBUTTON1   103
```

```
#define IDC_RADIOBUTTON2   104
#define IDC_RADIOBUTTON3   105
#define IDC_R_3   105
#define IDC_RADIOBUTTON4   106
#define IDC_STATICTEXT33   109
#define IDC_STATICTEXT34   110
#define IDC_R_CLOSW  304
#define IDC_R_CLOSE  304
#define IDC_R_NEW 101
#define IDC_R_ORIGINAL  102
#define IDC_R_CEILINGS  107
#define IDC_R_FLOORS 103
#define IDC_R_WALLS  104
#define IDC_R_SPRITES   105
#define IDO_REPLACE  218
#define IDC_EDIT49   102
#define IDC_STATICTEXT38   103
#define IDC_EDIT56   104
#define IDC_BUTTON24 102
#define IDC_BUTTON23 105
#define IDC_R_REPLACE   306
#define IDC_STATICFRAME17  219
#define IDC_EDIT55   220
#define IDC_STATICTEXT36   241
#define IDC_SINFO 220
#define IDC_STATICFRAME18  161
#define IDC_ABOUT 152
#define IDC_ADDRESS  151
#define IDC_SINFOBOX 246
#define IDC_BUTTON16 103
#define IDC_BOPTIM   103
#define IDC_GROUPEXTRACT   103
#define IDC_STATICTEXT41   105
#define IDC_STATICTEXT42   106
#define IDC_STATICTEXT59   107
#define IDC_STATICTEXT43   116
#define IDC_STATICFRAME25  117
#define IDC_BLOG  108
#define IDC_BORIGI   102
#define IDC_BORIG 102
#define IDC_STATUS   116
#define IDC_VERBOSE  115
#define IDC_START 107
#define IDC_ORIG  104
#define IDC_OPTIM 109
#define IDC_CHECKLOG 114
#define IDC_LOG   110
#define IDC_GROUPTEXT   107
#define IDC_GROUPPATH   104
#define IDC_BUTTON12 218
#define IDC_TEST_TEST   306
#define IDC_TEST_OK  305
#define IDI_ICON1 1
#define DLG_SPRITE   2
#define IDC_CHECKBOX1   101
#define IDC_CHECKBOX2   102
#define IDC_CHECKBOX3   103
#define IDC_CHECKBOX4   104
#define IDC_CHECKBOX5   105
#define IDC_CHECKBOX6   106
#define IDC_CHECKBOX7   107
#define IDC_CHECKBOX8   108
#define STUB_MED  108
#define STUB_HOLO 105
#define STUB_JP   106
#define STUB_NV   107
#define STUB_INV  102
#define STUB_INVL 103
#define STUB_INVR 104
#define IDC_CHECKBOX9   109
#define IDC_CHECKBOX10  110
#define STER_KICK 110
#define STUB_STER 109
#define IDC_LISTBOX2 112
#define IDS_INVISIBLE   110
#define IDC_EDIT9 111
```

```
#define IDC_EDIT10   112
#define IDC_N_SETUPFILE 112
#define IDC_EDIT11   113
#define STUB_SHRAP   113
#define STUB_LAMEL   111
#define STUB_LUNAR   112
#define IDC_EDIT12   114
#define IDC_CHECKBOX11  115
#define STUB_1 115
#define IDC_CHECKBOX12  116
#define STUB_2 116
#define IDC_CHECKBOX13  114
#define IDC_STATICTEXT45   117
#define IDC_STATICTEXT46   118
#define IDC_STATICFRAME28  119
#define IDC_STATICTEXT47   119
#define IDC_EDIT13   115
#define IDS_SHADE 115
#define IDC_EDIT14   116
#define IDC_EDIT15   117
#define IDC_EDIT16   118
#define IDC_EDIT17   119
#define IDS_YREP  119
#define IDC_EDIT18   120
#define IDC_EDIT19   121
#define IDS_YOFF  121
#define IDC_EDIT20   122
#define IDS_SECTNUM  122
#define IDC_EDIT21   123
#define IDS_STATNUM  123
#define IDC_EDIT22   124
#define IDS_ANG   124
#define IDC_EDIT23   125
#define IDS_XVEL  125
#define IDC_EDIT24   126
#define IDS_YVEL  126
#define IDC_EDIT25   127
#define IDS_ZVEL  127
#define IDC_EDIT39   128
#define IDS_LOTAG 128
#define IDC_EDIT40   129
#define IDS_HITAG 129
#define IDC_EDIT41   130
#define IDS_EXTRA 130
#define IDC_STATICTEXT4 131
#define IDC_STATICTEXT10   132
#define IDC_STATICTEXT11   133
#define IDC_STATICTEXT13   134
#define IDC_STATICTEXT14   135
#define IDS_XOFF  120
#define IDS_XREP  118
#define IDS_CLIPDIST 117
#define IDS_PAL   116
#define IDS_PICNUM   114
#define IDS_Z  113
#define IDS_Y  112
#define IDS_X  111
#define IDS_TRANSREV 109
#define IDS_HITSCAN  108
#define IDS_CENTER   107
#define IDS_ONESIDE  106
#define IDS_FACEWALLFLOOR  105
#define IDS_XFLIP 103
#define IDS_TRANS 102
#define IDS_YFLIP 104
#define IDS_BLOCK 101
#define IDWNEXT   242
#define IDWLAST   243
#define IDWFLAST  244
#define IDWFNEXT  245
#define IDC_STATICFRAME10  216
#define IDC_EDIT8 151
#define IDC_BUTTON1  153
#define IDNEWSECTOR  153
#define IDC_BUTTON2  154
#define IDNEWWALL 154
```

```
#define IDC_BUTTON3  155
#define IDNEWMAP  155
#define IDC_EDIT43   156
#define STUB_PATCHED 156
#define IDC_STATICTEXT55   157
#define IDC_STATICFRAME22  158
#define IDC_STATICTEXT56   159
#define STUB_PATCH   156
#define IDC_EDIT44   184
#define IDC_EDIT45   188
#define IDC_EDIT46   189
#define IDC_STATICTEXT16   190
#define IDC_STATICTEXT25   191
#define IDC_STATICTEXT26   192
#define IDC_STATICTEXT27   193
#define IDC_EDIT47   194
#define IDC_STATICTEXT28   195
#define IDC_STATICTEXT29   196
#define IDC_STATICFRAME15  197
#define IDC_STATICFRAME16  217
#define IDC_EDIT50   105
#define IDC_STATICFRAME5   215
#define IDC_EDIT51   101
#define IDC_TEST_CANCEL 304
#define IDC_STATICTEXT30   158
#define IDC_BUTTON13 106
#define IDC_BUTTON20 107
#define IDC_BUTTON21 108
#define IDC_EDIT68   109
#define IDC_BUTTON22 109
#define IDC_BUTTON25 110
#define IDC_BUTTON26 111
#define IDC_BUTTON27 112
#define IDC_EDIT57   110
#define IDC_STATICFRAME24  112
#define IDC_STATICFRAME23  111
#define IDC_N_PLAYERS   110
#define IDC_STATICTEXT39   111
#define IDC_RUN_EDITART 109
#define IDC_RUN_BUILD   108
#define IDC_RUN_SETUP   107
#define IDC_RUN_DUKE 106
#define IDC_N_EDITART   105
#define IDC_N_BUILD  104
#define IDC_N_SETUP  102
#define IDC_N_DUKE   101
#define IDC_EDIT48   150
#define IDC_BUTTON11 194
#define IDSAVEAS  194
#define IDOPEN 105
#define IDP_ANG   198
#define IDC_BUTTON10 150
#define IDP_SECTOR   188
#define IDP_Z  184
#define IDP_Y  156
#define IDP_X  189
#define IDC_STATICTEXT1 150
#define IDC_STATICFRAME19  151
#define IDC_STATICFRAME20  152
#define IDC_STATICFRAME21  153
#define IDC_STATICTEXT20   151
#define IDC_STATICTEXT12   152
#define IDC_STATICTEXT21   153
#define IDC_STATICTEXT22   154
#define IDC_STATICTEXT23   155
#define STUB_KICK 110
#define IDC_STATICTEXT24   156
#define IDC_BUTTON5  157
#define IDC_BUTTON6  140
#define IDC_BUTTON7  158
#define IDC_BUTTON8  159
#define IDC_BUTTON9  160
#define IDC_BUTTON14 199
#define IDO_CRACKREMOVE 199
#define IDC_STATICTEXT31   200
#define IDO_CRACKAPPLY  160
```

```
#define IDO_SEENINEREMOVE  104
#define IDO_SEENINEAPPLY   106
#define IDS_FNEXT 160
#define IDC_STATICTEXT32   161
#define IDC_STATICFRAME1   162
#define IDC_BUTTON17 164
#define IDC_ENABLE   164
#define IDC_EDIT53   161
#define IDC_INFOBOX  161
#define IDC_STATICTEXT35   163
#define IDC_EDIT54   165
#define IDS_OWNER 165
#define IDC_BUTTON18 166
#define IDC_BUTTON19 167
#define IDS_AUTOUPDATE  167
#define IDC_STATICTEXT37   168
#define IDC_INFOTITLE   168
#define IDS_UPDATE   166
#define IDC_EDIT52   163
#define IDC_INFOBOX1 1610
#define IDS_CSECTNUM 157
#define IDC_CSECTOR  157
#define IDS_FLAST 140
#define IDS_LAST  158
#define IDS_NEXT  159
#define IDS_CENTRE   107
#define DLG_DELETEITEM  1
#define IDC_STATICFRAME2   113
#define IDC_STATICTEXT58   114
#define IDC_STATICTEXT60   112
#define IDC_STATICFRAME26  121
#define IDC_STATICFRAME29  122
#define IDC_GROUPNAME   120
#define IDC_OPEN  110
#define IDC_NEW   111
#define IDC_DELETEFILE  109
#define IDC_ADDFILE  108
#define IDC_METHOD2  117
#define IDC_METHOD1  116
#define IDC_METHODTEXT  118
#define IDC_STATICFRAME27  115
#define IDC_BUTTON28 116
#define IDC_BUTTON29 117
#define IDCURSECT 166
#define IDC_STATICFRAME6   210
#define IDC_STATICFRAME7   211
#define IDC_STATICFRAME8   107
#define IDC_STATICFRAME9   212
#define ID_WONEWAY   213
#define IDJOININGSECTOR 187
#define IDJOININGWALL   186
#define IDADJACENTWALL  185
#define IDCURRENTWALL   183
#define IDFIRSTWALL  214
#define ID_WREV   208
#define ID_WXFLIP 204
#define ID_WYFLIP 209
#define ID_WBLOCKED  201
#define ID_WSWAPPED  202
#define ID_WALIGNED  203
#define ID_WMASKED   205
#define ID_WHITSCAN  206
#define ID_WTRANS 207
#define ID_WNEXTS 170
#define ID_WNEXTW 169
#define ID_WPOINT2   168
#define ID_WNUM   182
#define ID_WEXTRA 181
#define ID_WHITAG 180
#define ID_WLOTAG 179
#define ID_WYPAN  178
#define ID_WXPAN  177
#define ID_WYREP  176
#define ID_WXREP  174
#define ID_WPAL   175
#define ID_WSHADE 173
```

```
#define ID_WOVERPIC  172
#define ID_WPIC   171
#define ID_WY  167
#define ID_WX  114
#define IDC_STATICTEXT2 120
#define IDC_STATICTEXT48   121
#define IDC_STATICTEXT49   122
#define IDC_STATICTEXT50   123
#define IDC_STATICTEXT51   124
#define IDC_STATICTEXT52   125
#define IDC_STATICTEXT53   126
#define IDC_CHECKBOX14  127
#define IDC_CHECKBOX15  128
#define IDC_CHECKBOX16  129
#define IDC_CHECKBOX17  130
#define IDC_CHECKBOX18  131
#define IDC_CHECKBOX19  132
#define IDC_CHECKBOX20  133
#define IDC_STATICFRAME3   130
#define ID_CPAL   131
#define ID_VIS 140
#define ID_FPAL   124
#define ID_FYFLIP 111
#define IDLOAD 105
#define ID_CYFLIP 128
#define ID_CXFLIP 127
#define ID_CSMOOTH   125
#define ID_CSWAP  126
#define ID_FPARALAX  103
#define ID_FSWAP  109
#define ID_FSMOOTH   108
#define ID_FXFLIP 110
#define ID_FALIGN 112
#define ID_FY  119
#define ID_FX  117
#define ID_FSHADE 121
#define ID_FPIC   118
#define ID_FSLOPE 122
#define ID_FZ  116
#define ID_CALIGN 129
#define IDC_LISTBOX1 220
#define IDC_EDIT1 221
#define IDC_EDIT2 222
#define IDC_EDIT3 223
#define IDC_EDIT4 224
#define IDC_EDIT5 225
#define IDC_EDIT6 226
#define IDC_EDIT7 227
#define IDC_EDIT26 228
#define IDC_EDIT27 229
#define IDC_EDIT28 230
#define IDC_EDIT29 231
#define IDC_EDIT30 232
#define IDC_EDIT31 233
#define IDC_EDIT32 234
#define IDC_EDIT33 235
#define IDC_EDIT34 236
#define IDC_EDIT35 237
#define IDC_EDIT36 238
#define IDC_EDIT37 239
#define IDC_EDIT38 240
#define ID_CY  143
#define ID_CX  142
#define ID_CSHADE 137
#define ID_CZ  144
#define ID_CSLOPE 141
#define ID_CPIC   132
#define ID_CPARALAX  123
#define ID_SECTORNUM 101
#define ID_NUMWALL   115
#define ID_EXTRA  139
#define ID_HITAG  138
#define ID_LOTAG  133
#define ID_FHEINUM   134
#define ID_CHEINUM   135
#define IDC_STATICTEXT5 145
```

```
#define IDC_STATICTEXT6 146
#define IDC_STATICTEXT7 147
#define IDC_STATICTEXT8 148
#define IDC_STATICTEXT9 149
#define IDC_STATICFRAME4   157
#define IDSAVE 159
#define IDNEXT 162
#define IDLAST 163
#define IDFNEXT   165
#define IDFLAST   164
#define IDC_STATICTEXT3 136
#define IDC_STATICTEXT54   137
#define IDC_EDIT58   138
#define STUB_NAME 138
#define IDC_STATICFRAME11  137
#define IDC_STATICFRAME12  138
#define IDC_STATICTEXT15   139
#define IDC_EDIT59   140
#define IDC_EDIT60   141
#define IDC_EDIT61   142
#define IDC_EDIT62   143
#define IDC_EDIT63   144
#define IDC_EDIT64   145
#define IDC_STATICTEXT17   141
#define IDC_STATICTEXT18   142
#define IDC_STATICTEXT19   143
#define IDC_STATICFRAME13  144
#define IDC_STATICFRAME14  145
#define IDC_EDIT42   146
#define IDC_EDIT65   147
#define IDC_EDIT66   148
#define IDC_EDIT67   149
#define STUB_W0   149
#define STUB_W10  149
#define STUB_10   132
#define STUB_W9   148
#define STUB_W8   147
#define STUB_W7   146
#define STUB_W6   145
#define STUB_W5   144
#define STUB_W4   143
#define STUB_W3   142
#define STUB_W2   141
#define STUB_W1   140
#define STUB_9 132
#define STUB_8 131
#define STUB_7 130
#define STUB_6 129
#define STUB_5 128
#define STUB_4 127
#define STUB_3 114
#define IDS_SPRITE   146
#define IDC_BUTTON4  147
#define IDS_CSPRITE  147
#define ID_FIRSTWALL 102
```

OpenGL Build Touch Source Code

defines.h

```
#define IDH_S_X                 100
#define IDH_S_Y                 101
#define IDH_S_Z                 102
#define IDH_S_BLOCKING          103
#define IDH_S_TRANSLUCENT       104
#define IDH_S_REVERSING         105
#define IDH_S_HITSCAN           106
#define IDH_S_X_FLIP            107
#define IDH_S_Y_FLIP            108
#define IDH_S_FACE              109
#define IDH_S_SINGLE            110
#define IDH_S_CENTRE            111
#define IDH_S_INVISIBLE         112
#define IDH_S_PICNUM            113
#define IDH_S_SHADE             114
#define IDH_S_PAL               115
#define IDH_S_CLIPDIST          116
#define IDH_S_XREPEAT           117
#define IDH_S_YREPEAT           118
#define IDH_S_XOFFSET           119
#define IDH_S_YOFFSET           120
#define IDH_S_SECTNUM           121
#define IDH_S_AUTO              122
#define IDH_S_UPDATE            123
#define IDH_S_LAST              124
#define IDH_S_NEXT              125
#define IDH_S__LAST             126
#define IDH_S__NEXT             127
#define IDH_S_SPRITENUM_SEL     128
#define IDH_S_SPRITENUM         129
#define IDH_S_ENABLE            130
#define IDH_S_INFO              131
#define IDH_S_INFOBOX           132
#define IDH_S_STATNUM           133
#define IDH_S_ANG               134
#define IDH_S_XVEL              135
#define IDH_S_YVEL              136
#define IDH_S_ZVEL              137
#define IDH_S_OWNER             138
#define IDH_S_EXTRA             139
#define IDH_S_LOTAG             140
#define IDH_S_HITAG             141
#define IDH_S_SECTNUMSEL        142

#define IDH_E_N_DUKE            200
#define IDH_E_N_SETUP           201
#define IDH_E_N_BUILD           202
#define IDH_E_N_EDITART         203
#define IDH_E_N_CONFIG          204
#define IDH_E_N_PLAYERS         205
#define IDH_E_S_DUKE            206
#define IDH_E_S_SETUP           207
#define IDH_E_S_BUILD           208
#define IDH_E_S_EDITART         209




#define IDH_O_PARALLAX          300
#define IDH_O_SLOPED            301
#define IDH_O_SWAP              302
#define IDH_O_SMOOTH            303
#define IDH_O_XFLIP             304
#define IDH_O_YFLIP             305
#define IDH_O_ALIGN             306
#define IDH_O_Z                 307
#define IDH_O_SLOPE             308
#define IDH_O_TEXTURE           309
#define IDH_O_SHADE             310
#define IDH_O_PALETTE           311
#define IDH_O_XOFF              312
#define IDH_O_YOFF              313
#define IDH_O_CSECT             314
#define IDH_O_CSECTSEL          315
#define IDH_O_FWALL             316
#define IDH_O_FWALLSEL          317
```

```
#define IDH_O_NWALLS            318
#define IDH_O_VISIBILITY        319
#define IDH_O_LOTAG             320
#define IDH_O_HITAG             321
#define IDH_O_EXTRA             322
#define IDH_O_LAST              323
#define IDH_O_NEXT              324
#define IDH_O__LAST             325
#define IDH_O__NEXT             326
#define IDH_O_NEWWALL           327
#define IDH_O_NEWSECT           328
#define IDH_O_NEWMAP            329

#define IDH_W_X                 400
#define IDH_W_Y                 401
#define IDH_W_TEXTURE           402
#define IDH_W_MASK              403
#define IDH_W_SHADE             404
#define IDH_W_PALETTE           405
#define IDH_W_XREP              406
#define IDH_W_YREP              407
#define IDH_W_XPAN              408
#define IDH_W_YPAN              409
#define IDH_W_LOTAG             410
#define IDH_W_HITAG             411
#define IDH_W_EXTRA             412
#define IDH_W_BLOCK             413
#define IDH_W_SWAP              414
#define IDH_W_ALIGN             415
#define IDH_W_MASKED            416
#define IDH_W_HITSCAN           417
#define IDH_W_ONEWAY            418
#define IDH_W_CWALL             419
#define IDH_W_CWALLSEL          420
#define IDH_W_AWALL             421
#define IDH_W_AWALLSEL          422
#define IDH_W_JWALL             423
#define IDH_W_JWALLSEL          424
#define IDH_W_JSECT             425
#define IDH_W_JSECTSEL          426
#define IDH_W_XFLIP             427
#define IDH_W_YFLIP             428
#define IDH_W_TRANSLUCENCE      429
#define IDH_W_REVERSING         430
#define IDH_W_LAST              431
#define IDH_W_NEXT              432
#define IDH_W__LAST             433
#define IDH_W__NEXT             434

#define IDH_M_OPEN              500
#define IDH_M_SAVE              501
#define IDH_M_SAVEAS            502
#define IDH_M_X                 503
#define IDH_M_Y                 504
#define IDH_M_Z                 505
#define IDH_M_ANG               506
#define IDH_M_SECTOR            507
#define IDH_M_SAPPLY            508
#define IDH_M_SREMOVE           509
#define IDH_M_CAPPLY            510
#define IDH_M_CREMOVE           511
#define IDH_M_REPLACE           512
#define IDH_M_INFOBOX           513
#define IDH_M_INFO              514
#define IDH_M_ADDRESS           515

#define IDH_R_WALL              600
#define IDH_R_FLOOR             601
#define IDH_R_CEILING           602
#define IDH_R_SPRITE            603
#define IDH_R_REPLACE           604
#define IDH_R_NEW               605
#define IDH_R_ORIGINAL          606
#define IDH_R_CLOSE             607
```

OpenGL Build Touch Source Code

openglbt.rtf

{\rtf1\ansi\deff4\deflang1024
{\fonttbl
{\f0\froman Times New Roman;}
{\f1\froman Symbol;}
{\f2\fswiss Arial;}
{\f3\froman MS Serif;}
{\f4\fswiss MS Sans Serif;}
{\f5\fmodern Courier;}
{\f6\fdecor Zapf Dingbats;}
{\f7\fswiss Helvetica Condensed;}
{\f8\fswiss Helvetica;}
}
\f0
\fs22\b
OpenGL Build Touch v1.9.1\b0 - © James Ferry 1999.\par\par
\fs19
#{\footnote MAIN}
Please note:\par
 I have increased the memory again, it now uses approximatly 12mb of memory.\par
 Please use a resolution of 1024x800 or greater as some of the dialogue boxes are rather large.\par
\par

Controls (these buttons are for the main screen only, which is the one with the map)\par
  escape\tab quit\par
  F1\tab this information screen\par
  1\tab toggle one sided wall verteces\par
  2\tab 2d mode\par
  3\tab 3d mode\par
  4\tab toggle wall verteces\par
  5\tab opens sprite editing window\par
  6\tab previous sprite\par
  7\tab next sprite\par
  8\tab previous sector\par
  9\tab next sector\par
  0/o\tab opens main editing window (walls, sectors and file management)\par
  -=_\tab previous sprite\par
  =/+\tab next sprite\par
  e\tab opens the external program manager\par
  c\tab opens the patch file creation function\par
  t\tab opens the con optimisation screen\par
  up\tab scrolls map up\par
  down\tab scrolls map down\par
  left\tab scrolls map left\par
  right\tab scrolls map right\par
\par
A menu has been added where most of the functions can be accessed from.\par
Group file extraction has been added to the program.\par
Group file creation has been added to the program.\par
Group file addition has been added to the program. (two methods, one uses temporary file, the other
 modifies directly\par
Group file deletion has been added to the program. (uses a temporary file)\par
Patch file creation has been added to the program.\par
Con file optimisation has been added to the program.\par
The about screen has been changed.\par
\par
Current email address can be found at\par
\f4
\fs19
http://members.xoom.com/HCAD\par\par
\f0
\fs19
OpenGL Build Touch v1.9.1 - © James Ferry 1999.
\page
\f4
\fs16
#{\footnote IDH_S_X}x axis co-ordinate.\page
#{\footnote IDH_S_Y}y axis co-ordinate.\page
#{\footnote IDH_S_Z}z axis co-ordinate.\page
#{\footnote IDH_S_BLOCKING}Check to make a blocking sprite (range and distances).\page
#{\footnote IDH_S_TRANSLUCENT}Check to make the sprite slightly translucent.\page
#{\footnote IDH_S_REVERSING}Check to make the sprite more translucent.\page
#{\footnote IDH_S_HITSCAN}Check to make a blocking sprite (weapons).\page
#{\footnote IDH_S_X_FLIP}Check to flip the sprite along the x axis.\page
#{\footnote IDH_S_Y_FLIP}Check to flip the sprite along the y axis.\page
#{\footnote IDH_S_FACE}No check mark makes the object "real", a partial check mark for a wall (xz p
lane) sprite, and a check mark for a flat sprite in the floor (xy) plane.\page

#{\footnote IDH_S_SINGLE}Check to make the visible from one side only.\page
#{\footnote IDH_S_CENTRE}Check to make the sprite's co-ordinate origin at either its foot or centre
.\page
#{\footnote IDH_S_INVISIBLE}Check to make the invisible.\page
#{\footnote IDH_S_PICNUM}Image number.\page
#{\footnote IDH_S_SHADE}Shade of the sprite.\page
#{\footnote IDH_S_PAL}Palette lookup table number\page
#{\footnote IDH_S_CLIPDIST}The size of the movement clipping square (face/"real" sprites only).\pag
e
#{\footnote IDH_S_XREPEAT}Used to change the size of sprite along the x axis.\page
#{\footnote IDH_S_YREPEAT}Used to change the size of sprite along the y axis.\page
#{\footnote IDH_S_XOFFSET}Used to center the sprite along the x axis.\page
#{\footnote IDH_S_YOFFSET}Used to center the sprite along the y axis.\page
#{\footnote IDH_S_SECTNUMSEL}Changes current sector to that of the sprite.\page
#{\footnote IDH_S_SECTNUM}Current sector of sprite.\page
#{\footnote IDH_S_AUTO}Automatically updates the map.\page
#{\footnote IDH_S_UPDATE}Updates the map.\page
#{\footnote IDH_S_LAST}Decrement current sprite by one.\page
#{\footnote IDH_S_NEXT}Increment current sprite by one.\page
#{\footnote IDH_S__LAST}Decrement current sprite by ten.\page
#{\footnote IDH_S__NEXT}Increment current sprite by ten.\page
#{\footnote IDH_S_SPRITENUM_SEL}Makes the current sprite the one with the below number.\page
#{\footnote IDH_S_SPRITENUM}This is the current sprites number (location of structure).\page
#{\footnote IDH_S_ENABLE}This will enable extra information regarding New World III TC.\page
#{\footnote IDH_S_INFO}Information about this sprite.\page
#{\footnote IDH_S_INFOBOX}Information about this sprite.\page
#{\footnote IDH_S_STATNUM}Current status of sprite (inactive/monster/bullet, etc.).\page
#{\footnote IDH_S_ANG}The angle that the sprite is facing.\page
#{\footnote IDH_S_XVEL}The x axis velocity of the sprite.\page
#{\footnote IDH_S_YVEL}The y axis velocity of the sprite.\page
#{\footnote IDH_S_ZVEL}The z axis velocity of the sprite.\page
#{\footnote IDH_S_OWNER}Unknown.\page
#{\footnote IDH_S_EXTRA}Unknown.\page
#{\footnote IDH_S_LOTAG}The sprites current low tag.\page
#{\footnote IDH_S_HITAG}The sprites current high tag.\page

#{\footnote IDH_E_N_DUKE}Enter the name of the d3d executionable here.\page
#{\footnote IDH_E_N_SETUP}Enter the name of the d3d setup executionable here.\page
#{\footnote IDH_E_N_BUILD}Enter the name of the build executionable here.\page
#{\footnote IDH_E_N_EDITART}Enter the name of the editart executionable here.\page
#{\footnote IDH_E_N_CONFIG}Enter the name of your configuration file here here.\page
#{\footnote IDH_E_N_PLAYERS}Enter the number of players that you would like to use for testing mult
iplayer.\page
#{\footnote IDH_E_S_DUKE}Start d3d.\page
#{\footnote IDH_E_S_SETUP}Start setup.\page
#{\footnote IDH_E_S_BUILD}Start build.\page
#{\footnote IDH_E_S_EDITART}Start editart.\page

#{\footnote IDH_O_PARALLAX}Makes the surface look like an arching sky or horizon.\page
#{\footnote IDH_O_SLOPED}Enables sloping - used to enable surfaces with a gradient.\page
#{\footnote IDH_O_SWAP}Swaps the texture along the x and y axis.\page
#{\footnote IDH_O_SMOOTH}Expands and contracts the texture.\page
#{\footnote IDH_O_XFLIP}Flips the texture along the x axis.\page
#{\footnote IDH_O_YFLIP}Flips the texture along the y axis.\page
#{\footnote IDH_O_ALIGN}Enables relative alignment to the first wall.\page
#{\footnote IDH_O_Z}Height of the surface.\page
#{\footnote IDH_O_SLOPE}Gradient of the surface - sloping must be enabled.\page
#{\footnote IDH_O_TEXTURE}Texture/image of the surface.\page
#{\footnote IDH_O_SHADE}Shade of the surface.\page
#{\footnote IDH_O_PALETTE}Palette lookup table to be used for the surface.\page
#{\footnote IDH_O_XOFF}Offset of the texture along the x axis.\page
#{\footnote IDH_O_YOFF}Offset of the texture along the y axis.\page
#{\footnote IDH_O_CSECT}Current sector structure number.\page
#{\footnote IDH_O_CSECTSEL}Set the current sector to the sector with the given number.\page
#{\footnote IDH_O_FWALL}first wall structure number.\page
#{\footnote IDH_O_FWALLSEL}Set the current wall to the first wall of the sector.\page
#{\footnote IDH_O_NWALLS}Number of walls in the sector.\page
#{\footnote IDH_O_VISIBILITY}Visibility of the sector.\page
#{\footnote IDH_O_LOTAG}Low tag of the sector.\page
#{\footnote IDH_O_HITAG}High tag of the sector.\page
#{\footnote IDH_O_EXTRA}Unknown.\page
#{\footnote IDH_O_LAST}Decrement current sector by one.\page
#{\footnote IDH_O_NEXT}Increment current sector by one.\page
#{\footnote IDH_O__LAST}Decrement current sector by ten.\page
#{\footnote IDH_O__NEXT}Increment current sector by ten.\page
#{\footnote IDH_O_NEWWALL}Creates a new, empty wall structure.\page

```
#{\footnote IDH_O_NEWSECT}Creates a new empty sector structure.\page
#{\footnote IDH_O_NEWMAP}Resets current map.\page

#{\footnote IDH_W_X}The x co-ordinate of the first vertex of the wall.  The second vertex is given
by the first vertex of the adjacent wall.\page
#{\footnote IDH_W_Y}The y co-ordinate of the first vertex of the wall.  The second vertex is given
by the first vertex of the adjacent wall.\page
#{\footnote IDH_W_TEXTURE}Texture/image of the wall.\page
#{\footnote IDH_W_MASK}Texture/image of the masked wall.  Only visible when masking is enabled.\pag
e
#{\footnote IDH_W_SHADE}Shade of the wall.\page
#{\footnote IDH_W_PALETTE}Palette lookup table to use for the wall.\page
#{\footnote IDH_W_XREP}Sets the actual size/dimensions of the texture along the x axis.\page
#{\footnote IDH_W_YREP}Sets the actual size/dimensions of the texture along the y axis.\page
#{\footnote IDH_W_XPAN}Panning along the x axis.\page
#{\footnote IDH_W_YPAN}Panning along the y axis.\page
#{\footnote IDH_W_LOTAG}Low tag of the wall.\page
#{\footnote IDH_W_HITAG}High tag of the wall.\page
#{\footnote IDH_W_EXTRA}Unknown.\page
#{\footnote IDH_W_BLOCK}Makes the wall blockable for sprites that require a clipping distance. This
 will stop actors from moving over it.\page
#{\footnote IDH_W_SWAP}When checked, it uses the current texture for the lower part of the wall, an
d the upper texture is that of the wall on the other side, ie: the texture of the joining wall.\pag
e
#{\footnote IDH_W_ALIGN}Align the image at the top (unchecked) or bottom (checked) of its sector.\p
age
#{\footnote IDH_W_MASKED}Enables masking.\page
#{\footnote IDH_W_HITSCAN}Enables hit scanning. This will stop actors from shooting through the wal
l.\page
#{\footnote IDH_W_ONEWAY}Only lets actors and objects pass through one direction.\page
#{\footnote IDH_W_CWALL}Current wall structure number.\page
#{\footnote IDH_W_CWALLSEL}Set the current wall to the wall with the given number.\page
#{\footnote IDH_W_AWALL}Adjacent wall structure number.\page
#{\footnote IDH_W_AWALLSEL}Set the current wall to the adjacent wall.\page
#{\footnote IDH_W_JWALL}Joining wall structure number.\page
#{\footnote IDH_W_JWALLSEL}Set the current wall to the joining wall.\page
#{\footnote IDH_W_JSECT}Joining sector structure number.\page
#{\footnote IDH_W_JSECTSEL}Set the current sector to the joining sector.\page
#{\footnote IDH_W_XFLIP}Texture/image is flipped along the x axis.\page
#{\footnote IDH_W_YFLIP}Texture/image is flipped along the y axis.\page
#{\footnote IDH_W_TRANSLUCENCE}Enables translucence of masked walls.\page
#{\footnote IDH_W_REVERSING}Enables more translucence of masked walls.\page
#{\footnote IDH_W_LAST}Decrement current wall by one.\page
#{\footnote IDH_W_NEXT}Increment current wall by one.\page
#{\footnote IDH_W__LAST}Decrement current wall by ten.\page
#{\footnote IDH_W__NEXT}Increment current wall by ten.\page

#{\footnote IDH_M_OPEN}Open a file.\page
#{\footnote IDH_M_SAVE}Save the current file.\page
#{\footnote IDH_M_SAVEAS}Save the current file under another name.\page
#{\footnote IDH_M_X}The first players starting x co-ordinate.\page
#{\footnote IDH_M_Y}The first players starting y co-ordinate.\page
#{\footnote IDH_M_Z}The first players starting z co-ordinate.\page
#{\footnote IDH_M_ANG}The first players starting anglex.\page
#{\footnote IDH_M_SECTOR}The first players starting sector.\page
#{\footnote IDH_M_SAPPLY}Applies seenine optimization.\page
#{\footnote IDH_M_SREMOVE}Removes seenine optimization.\page
#{\footnote IDH_M_CAPPLY}Applies crack optimization.\page
#{\footnote IDH_M_CREMOVE}Removes crack optimization.\page
#{\footnote IDH_M_REPLACE}Opens replacment window.\page
#{\footnote IDH_M_INFOBOX}Information.\page
#{\footnote IDH_M_INFO}Information.\page
#{\footnote IDH_M_ADDRESS}http://members.xoom.com/HCAD\page

#{\footnote IDH_R_WALL}Check to replace wall images.\page
#{\footnote IDH_R_FLOOR}Check to replace floor images..\page
#{\footnote IDH_R_CEILING}Check to replace ceiling images.\page
#{\footnote IDH_R_SPRITE}Check to replace sprite images.\page
#{\footnote IDH_R_REPLACE}Replace all the original images with the new images\page
#{\footnote IDH_R_NEW}The new images to use.\page
#{\footnote IDH_R_ORIGINAL}Original images to replace.\page
#{\footnote IDH_R_CLOSE}Closes the window.\page

}
```

OpenGL Build Touch Source Code

cube2.rc

```
/************************************************************************

cube.rc

produced by Borland Resource Workshop

************************************************************************/



#include "cube.h"

DLG_DELETEITEM DIALOG 512, 0, 593, 344
STYLE DS_MODALFRAME | DS_3DLOOK | DS_CENTER | DS_CONTEXTHELP | WS_POPUP | WS_VISIBLE | WS_CAPTION |
 WS_SYSMENU
CLASS ""
CAPTION "Wall and sector editor"
FONT 8, "MS Sans Serif"
{
 CONTROL "OK", IDOK, "BUTTON", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 128,
 8, 48, 16
 CONTROL "Cancel", IDCANCEL, "BUTTON", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABST
OP, 128, 32, 48, 16
 CONTROL "<<", IDFLAST, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 1
15, 108, 12, 16
 CONTROL "<", IDLAST, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 128
, 108, 23, 16
 CONTROL ">", IDNEXT, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 153
, 108, 23, 16
 CONTROL ">>", IDFNEXT, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 1
77, 108, 12, 16
 CONTROL "parallax", ID_FPARALAX, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 3
2, 24, 40, 12
 CONTROL "sloped", ID_FSLOPE, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 32, 4
0, 40, 12
 CONTROL "swap", ID_FSWAP, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 32, 56,
40, 12
 CONTROL "smooth", ID_FSMOOTH, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 32,
72, 40, 12
 CONTROL "xflip", ID_FXFLIP, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 32, 88
, 40, 12
 CONTROL "yflip", ID_FYFLIP, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 32, 10
4, 40, 12
 CONTROL "align", ID_FALIGN, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 32, 12
0, 40, 12
 CONTROL "floorz", ID_FZ, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABS
TOP, 12, 136, 80, 13
 CONTROL "floorangle", ID_FHEINUM, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER
| WS_TABSTOP, 12, 152, 80, 12
 CONTROL "floorpic", ID_FPIC, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_
TABSTOP, 12, 168, 80, 12
 CONTROL "floorshade", ID_FSHADE, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER |
 WS_TABSTOP, 12, 184, 40, 12
 CONTROL "floorpal", ID_FPAL, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_
TABSTOP, 52, 184, 40, 12
 CONTROL "floorx", ID_FX, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABS
TOP, 12, 200, 40, 12
 CONTROL "floory", ID_FY, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABS
TOP, 44, 200, 40, 12
 CONTROL "parallax", ID_CPARALAX, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 2
32, 23, 40, 12
 CONTROL "sloped", ID_CSLOPE, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 232,
39, 40, 12
 CONTROL "swap", ID_CSWAP, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 232, 55,
 40, 12
 CONTROL "smooth", ID_CSMOOTH, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 232,
 71, 40, 12
 CONTROL "xflip", ID_CXFLIP, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 232, 8
7, 40, 12
 CONTROL "yflip", ID_CYFLIP, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 232, 1
03, 40, 12
 CONTROL "align", ID_CALIGN, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 232, 1
19, 40, 12
```

```
 CONTROL "ceilingz", ID_CZ, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TA
BSTOP, 212, 136, 80, 12
 CONTROL "ceilingangle", ID_CHEINUM, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDE
R | WS_TABSTOP, 212, 152, 80, 12
 CONTROL "ceilingpic", ID_CPIC, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | W
S_TABSTOP, 212, 168, 80, 13
 CONTROL "ceilingshade", ID_CSHADE, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER
 | WS_TABSTOP, 212, 184, 40, 12
 CONTROL "ceilingx", ID_CX, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TA
BSTOP, 212, 200, 40, 13
 CONTROL "ceilingpal", ID_CPAL, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | W
S_TABSTOP, 252, 184, 40, 12
 CONTROL "ceilingy", ID_CY, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TA
BSTOP, 252, 200, 40, 13
 CONTROL "Current Sector", IDCURSECT, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE |
 WS_TABSTOP, 308, 12, 60, 12
 CONTROL "sectornum", ID_SECTORNUM, "edit", ES_CENTER | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORD
ER | WS_TABSTOP, 308, 24, 60, 12
 CONTROL "First Wall", IDFIRSTWALL, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | W
S_TABSTOP, 308, 44, 60, 12
 CONTROL "firstwall", ID_FIRSTWALL, "edit", ES_LEFT | ES_LOWERCASE | ES_NUMBER | WS_CHILD | WS_VISI
BLE | WS_BORDER | WS_TABSTOP, 308, 56, 60, 12
 CONTROL "numwall", ID_NUMWALL, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | W
S_TABSTOP, 308, 84, 60, 12
 CONTROL "visibility", ID_VIS, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS
_TABSTOP, 308, 112, 60, 12
 CONTROL "lotag", ID_LOTAG, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TA
BSTOP, 308, 140, 60, 12
 CONTROL "hitag", ID_HITAG, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TA
BSTOP, 308, 168, 60, 12
 CONTROL "extra", ID_EXTRA, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TA
BSTOP, 308, 196, 60, 12
 CONTROL "X", ID_WX, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP,
8, 228, 52, 12
 CONTROL "Y", ID_WY, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP,
60, 228, 52, 12
 CONTROL "pic", ID_WPIC, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABST
OP, 8, 244, 52, 12
 CONTROL "overpic", ID_WOVERPIC, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP, 60, 244, 52, 12
 CONTROL "shade", ID_WSHADE, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_T
ABSTOP, 8, 260, 52, 12
 CONTROL "pal", ID_WPAL, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABST
OP, 60, 260, 52, 12
 CONTROL "xrep", ID_WXREP, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TAB
STOP, 8, 276, 52, 12
 CONTROL "yrep", ID_WYREP, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TAB
STOP, 60, 276, 52, 12
 CONTROL "xpan", ID_WXPAN, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 8, 292
, 52, 12
 CONTROL "ypan", ID_WYPAN, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TAB
STOP, 60, 292, 52, 12
 CONTROL "Lotag", ID_WLOTAG, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_T
ABSTOP, 8, 308, 52, 12
 CONTROL "Hitag", ID_WHITAG, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_T
ABSTOP, 60, 308, 52, 12
 CONTROL "Extra", ID_WEXTRA, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_T
ABSTOP, 8, 324, 52, 12
 CONTROL "Clip/block", ID_WBLOCKED, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP,
 236, 232, 60, 12
 CONTROL "Swapped", ID_WSWAPPED, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 23
6, 248, 60, 12
 CONTROL "Aligned", ID_WALIGNED, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 23
6, 264, 60, 12
 CONTROL "Masked", ID_WMASKED, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 236,
 280, 60, 12
 CONTROL "Hitscan", ID_WHITSCAN, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 23
6, 296, 60, 12
 CONTROL "One Way", ID_WONEWAY, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 236
, 312, 60, 11
 CONTROL "Current Wall", IDCURRENTWALL, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE
 | WS_TABSTOP, 308, 228, 60, 12
 CONTROL "currentwall", ID_WNUM, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP, 308, 240, 60, 12
 CONTROL "Adjacent Wall", IDADJACENTWALL, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIB
LE | WS_TABSTOP, 308, 256, 60, 12
```

```
 CONTROL "right", ID_WPOINT2, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_
TABSTOP, 308, 268, 60, 12
 CONTROL "Joining Wall", IDJOININGWALL, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE
 | WS_TABSTOP, 308, 284, 60, 12
 CONTROL "other wall", ID_WNEXTW, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER |
 WS_TABSTOP, 308, 296, 60, 12
 CONTROL "Joining Sector", IDJOININGSECTOR, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VIS
IBLE | WS_TABSTOP, 308, 312, 60, 12
 CONTROL "other sector", ID_WNEXTS, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER
 | WS_TABSTOP, 308, 324, 60, 12
 CONTROL "X Flipped", ID_WXFLIP, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 38
4, 232, 60, 12
 CONTROL "Y Flipped", ID_WYFLIP, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 38
4, 248, 60, 12
 CONTROL "Translucence", ID_WTRANS, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP,
 384, 264, 60, 12
 CONTROL "(Reversing)", ID_WREV, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 38
4, 280, 60, 12
 CONTROL "<", IDWLAST, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 38
4, 296, 27, 16
 CONTROL ">", IDWNEXT, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 41
3, 296, 27, 16
 CONTROL "<<", IDWFLAST, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP,
384, 313, 27, 12
 CONTROL ">>", IDWFNEXT, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP,
413, 313, 27, 12
 CONTROL "player x", IDP_X, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 412,
24, 60, 12
 CONTROL "player y", IDP_Y, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 412,
40, 60, 12
 CONTROL "player z", IDP_Z, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 412,
56, 60, 12
 CONTROL "player angle", IDP_ANG, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP,
 412, 72, 60, 12
 CONTROL "player sector", IDP_SECTOR, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABS
TOP, 412, 88, 60, 12
 CONTROL "New Wall", IDNEWWALL, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TA
BSTOP, 128, 56, 48, 12
 CONTROL "New Sector", IDNEWSECTOR, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | W
S_TABSTOP, 128, 72, 48, 12
 CONTROL "New Map", IDNEWMAP, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABS
TOP, 128, 88, 48, 12
 CONTROL "Open", IDOPEN, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP,
388, 148, 60, 12
 CONTROL "Save", IDSAVE, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP,
388, 164, 60, 12
 CONTROL "Save As", IDSAVEAS, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABS
TOP, 388, 180, 60, 12
 CONTROL "Frame2", IDC_STATICFRAME2, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 4, 4, 96, 21
6
 CONTROL "Floor", IDC_STATICTEXT2, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 16, 12, 72, 8
 CONTROL "Frame2", IDC_STATICFRAME3, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 204, 4, 96,
216
 CONTROL "Ceiling", IDC_STATICTEXT3, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 216, 12, 72, 8
 CONTROL "Number Of Walls", IDC_STATICTEXT5, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 308, 76,
60, 8
 CONTROL "Visibility", IDC_STATICTEXT6, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 308, 104, 60,
8
 CONTROL "Lotag", IDC_STATICTEXT7, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 308, 132, 60, 8
 CONTROL "Extra", IDC_STATICTEXT8, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 308, 188, 60, 8
 CONTROL "Hitag", IDC_STATICTEXT9, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 308, 160, 60, 8
 CONTROL "Z (height)", IDC_EDIT1, "edit", ES_LEFT | ES_READONLY | WS_CHILD | WS_VISIBLE | WS_BORDER
, 112, 136, 80, 12
 CONTROL "Texture", IDC_EDIT2, "edit", ES_LEFT | ES_READONLY | WS_CHILD | WS_VISIBLE | WS_BORDER, 1
12, 168, 80, 13
 CONTROL "Shade", IDC_EDIT3, "edit", ES_LEFT | ES_READONLY | WS_CHILD | WS_VISIBLE | WS_BORDER, 112
, 184, 40, 12
 CONTROL "X (offset)", IDC_EDIT4, "edit", ES_LEFT | ES_READONLY | WS_CHILD | WS_VISIBLE | WS_BORDER
, 112, 200, 40, 13
 CONTROL "Y (offset)", IDC_EDIT5, "edit", ES_LEFT | ES_READONLY | WS_CHILD | WS_VISIBLE | WS_BORDER
, 152, 200, 40, 13
 CONTROL "Palette", IDC_EDIT6, "edit", ES_LEFT | ES_READONLY | WS_CHILD | WS_VISIBLE | WS_BORDER, 1
52, 184, 40, 12
 CONTROL "Slope (4096=46*)", IDC_EDIT7, "edit", ES_LEFT | ES_READONLY | WS_CHILD | WS_VISIBLE | WS_
BORDER, 112, 152, 80, 12
 CONTROL "Frame5", IDC_STATICFRAME4, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 100, 128, 10
```

```
4, 92
 CONTROL "X", IDC_EDIT26, "edit", ES_LEFT | ES_READONLY | WS_CHILD | WS_VISIBLE | WS_BORDER, 120, 2
28, 52, 12
 CONTROL "Y", IDC_EDIT27, "edit", ES_LEFT | ES_READONLY | WS_CHILD | WS_VISIBLE | WS_BORDER, 172, 2
28, 52, 12
 CONTROL "Texture", IDC_EDIT28, "edit", ES_LEFT | ES_READONLY | WS_CHILD | WS_VISIBLE | WS_BORDER,
120, 244, 52, 12
 CONTROL "Mask Texture", IDC_EDIT29, "edit", ES_LEFT | ES_READONLY | WS_CHILD | WS_VISIBLE | WS_BOR
DER, 172, 244, 52, 12
 CONTROL "Shade", IDC_EDIT30, "edit", ES_LEFT | ES_READONLY | WS_CHILD | WS_VISIBLE | WS_BORDER, 12
0, 260, 52, 12
 CONTROL "X Repeat", IDC_EDIT31, "edit", ES_LEFT | ES_READONLY | WS_CHILD | WS_VISIBLE | WS_BORDER,
 120, 276, 52, 12
 CONTROL "Palette", IDC_EDIT32, "edit", ES_LEFT | ES_READONLY | WS_CHILD | WS_VISIBLE | WS_BORDER,
172, 260, 52, 12
 CONTROL "Y Repeat", IDC_EDIT33, "edit", ES_LEFT | ES_READONLY | WS_CHILD | WS_VISIBLE | WS_BORDER,
 172, 276, 52, 12
 CONTROL "X Panning", IDC_EDIT34, "edit", ES_LEFT | ES_READONLY | WS_CHILD | WS_VISIBLE | WS_BORDER
, 120, 292, 52, 12
 CONTROL "Y Panning", IDC_EDIT35, "edit", ES_LEFT | ES_READONLY | WS_CHILD | WS_VISIBLE | WS_BORDER
, 172, 292, 52, 12
 CONTROL "Lotag", IDC_EDIT36, "edit", ES_LEFT | ES_READONLY | WS_CHILD | WS_VISIBLE | WS_BORDER, 12
0, 308, 52, 12
 CONTROL "Hitag", IDC_EDIT37, "edit", ES_LEFT | ES_READONLY | WS_CHILD | WS_VISIBLE | WS_BORDER, 17
2, 308, 52, 12
 CONTROL "Extra", IDC_EDIT38, "edit", ES_LEFT | ES_READONLY | WS_CHILD | WS_VISIBLE | WS_BORDER, 12
0, 324, 52, 12
 CONTROL "Frame6", IDC_STATICFRAME6, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 116, 224, 11
2, 116
 CONTROL "Frame7", IDC_STATICFRAME7, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 4, 224, 112,
 116
 CONTROL "Frame8", IDC_STATICFRAME8, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 376, 224, 72
, 116
 CONTROL "Frame8", IDC_STATICFRAME9, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 228, 224, 72
, 116
 CONTROL "Frame10", IDC_STATICFRAME10, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 452, 224,
140, 116
 CONTROL "OpenGL Build Touch v1.9.1\n\n© James Ferry 1999", IDC_ABOUT, "static", SS_CENTER | WS_CHI
LD | WS_VISIBLE, 460, 260, 124, 28
 CONTROL "http://members.xoom.com/HCAD", IDC_ADDRESS, "edit", ES_CENTER | ES_READONLY | WS_CHILD |
WS_VISIBLE, 468, 296, 112, 12
 CONTROL "Player", IDC_STATICTEXT16, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 388, 12, 84, 9
 CONTROL "Y", IDC_STATICTEXT25, "static", SS_RIGHT | WS_CHILD | WS_VISIBLE, 388, 40, 20, 12
 CONTROL "Angle", IDC_STATICTEXT26, "static", SS_RIGHT | WS_CHILD | WS_VISIBLE, 388, 72, 20, 12
 CONTROL "X", IDC_STATICTEXT27, "static", SS_RIGHT | WS_CHILD | WS_VISIBLE, 388, 24, 20, 12
 CONTROL "Sector", IDC_STATICTEXT28, "static", SS_RIGHT | WS_CHILD | WS_VISIBLE, 388, 88, 20, 12
 CONTROL "Z", IDC_STATICTEXT29, "static", SS_RIGHT | WS_CHILD | WS_VISIBLE, 388, 56, 20, 12
 CONTROL "Frame11", IDC_STATICFRAME15, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 380, 4, 96
, 104
 CONTROL "Apply", IDO_SEENINEAPPLY, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | W
S_TABSTOP, 516, 28, 32, 13
 CONTROL "Seenine", IDC_STATICTEXT1, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 486, 28, 26, 12
 CONTROL "Remove", IDO_SEENINEREMOVE, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE |
 WS_TABSTOP, 552, 28, 32, 12
 CONTROL "Cracks", IDC_STATICTEXT30, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 486, 52, 26, 12
 CONTROL "Apply", IDO_CRACKAPPLY, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_
TABSTOP, 516, 52, 32, 12
 CONTROL "Remove", IDO_CRACKREMOVE, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | W
S_TABSTOP, 552, 52, 32, 12
 CONTROL "Optimisation", IDC_STATICTEXT31, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 484, 12, 10
0, 8
 CONTROL "Frame10", IDC_STATICFRAME5, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 3, 223, 446
, 118
 CONTROL "Frame11", IDC_STATICFRAME16, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 3, 3, 372,
 218
 CONTROL "Replacement", IDO_REPLACE, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE |
WS_TABSTOP, 508, 76, 52, 13
 CONTROL "Frame12", IDC_STATICFRAME17, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 480, 4, 10
8, 104
 CONTROL "Sectorinfo", IDC_SINFOBOX, "edit", ES_LEFT | ES_MULTILINE | ES_READONLY | ES_WANTRETURN |
 WS_CHILD | WS_VISIBLE | WS_VSCROLL, 464, 128, 120, 88
 CONTROL "Sector type", IDC_SINFO, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 464, 116, 120, 8
 CONTROL "Frame13", IDC_STATICFRAME18, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 460, 112,
128, 108
}
```

```
DLG_SPRITE DIALOG 9, 7, 511, 231
STYLE DS_MODALFRAME | DS_3DLOOK | DS_CONTEXTHELP | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Sprte editor"
FONT 8, "MS Sans Serif"
{
 CONTROL "OK", IDOK, "BUTTON", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 216,
 22, 50, 16
 CONTROL "Cancel", IDCANCEL, "BUTTON", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABST
OP, 216, 44, 50, 16
 CONTROL "Blocking Sprite", IDS_BLOCK, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABST
OP, 8, 12, 64, 8
 CONTROL "Translucent", IDS_TRANS, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP,
76, 44, 60, 9
 CONTROL "X flipped", IDS_XFLIP, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 8,
 44, 64, 8
 CONTROL "Y flipped", IDS_YFLIP, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 8,
 60, 64, 8
 CONTROL "Face/Wall/Floor", IDS_FACEWALLFLOOR, "button", BS_AUTO3STATE | WS_CHILD | WS_VISIBLE | WS
_TABSTOP, 8, 76, 64, 8
 CONTROL "1 sided", IDS_ONESIDE, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 76
, 12, 60, 8
 CONTROL "Real Centred", IDS_CENTRE, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP
, 76, 28, 60, 8
 CONTROL "Hitscan", IDS_HITSCAN, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 8,
 28, 64, 9
 CONTROL "Reversing", IDS_TRANSREV, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP,
 76, 60, 60, 8
 CONTROL "Invisible", IDS_INVISIBLE, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP
, 76, 76, 60, 8
 CONTROL "X", IDS_X, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 12, 108, 52,
 12
 CONTROL "Y", IDS_Y, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 12, 132, 52,
 12
 CONTROL "Z", IDS_Z, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 12, 156, 52,
 12
 CONTROL "Picnum", IDS_PICNUM, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 80
, 108, 52, 12
 CONTROL "Shade", IDS_SHADE, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 80,
132, 52, 12
 CONTROL "Pal", IDS_PAL, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 80, 156,
 52, 12
 CONTROL "Clipdist", IDS_CLIPDIST, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP
, 80, 180, 52, 12
 CONTROL "X repeat", IDS_XREP, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 14
8, 108, 52, 12
 CONTROL "Y repeat", IDS_YREP, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 14
8, 132, 52, 12
 CONTROL "X offset", IDS_XOFF, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 14
8, 156, 52, 12
 CONTROL "Y offset", IDS_YOFF, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 14
8, 180, 52, 12
 CONTROL "Sectnum", IDS_SECTNUM, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP,
12, 208, 52, 12
 CONTROL "Statnum", IDS_STATNUM, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP,
80, 204, 52, 12
 CONTROL "ang", IDS_ANG, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 12, 180,
 52, 12
 CONTROL "Xvel", IDS_XVEL, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 216, 8
8, 52, 12
 CONTROL "Yvel", IDS_YVEL, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 216, 1
12, 52, 12
 CONTROL "Zvel", IDS_ZVEL, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 216, 1
36, 52, 12
 CONTROL "lotag", IDS_LOTAG, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 216,
 160, 52, 12
 CONTROL "hitag", IDS_HITAG, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 216,
 184, 52, 12
 CONTROL "extra", IDS_EXTRA, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 216,
 208, 52, 12
 CONTROL "X Position", IDC_STATICTEXT4, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 12, 100, 52, 8
 CONTROL "Y Position", IDC_STATICTEXT10, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 12, 124, 52,
8
 CONTROL "Z Position", IDC_STATICTEXT11, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 12, 148, 52,
8
 CONTROL "X Velocity", IDC_STATICTEXT13, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 216, 80, 52,
```

```
8
 CONTROL "Y Velocity", IDC_STATICTEXT14, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 216, 104, 52,
 8
 CONTROL "Z Velocity", IDC_STATICTEXT3, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 216, 128, 52,
8
 CONTROL "Frame1", IDC_STATICFRAME11, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 4, 96, 68,
132
 CONTROL "Frame2", IDC_STATICFRAME12, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 208, 76, 68
, 152
 CONTROL "Angle", IDC_STATICTEXT15, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 12, 172, 52, 8
 CONTROL "Lotag", IDC_STATICTEXT17, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 216, 152, 52, 8
 CONTROL "Hitag", IDC_STATICTEXT18, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 216, 176, 52, 8
 CONTROL "Extra", IDC_STATICTEXT19, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 216, 200, 52, 8
 CONTROL "Frame3", IDC_STATICFRAME13, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 72, 96, 68,
 132
 CONTROL "Frame4", IDC_STATICFRAME14, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 140, 96, 68
, 132
 CONTROL "Sprite", IDS_SPRITE, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 14
4, 20, 56, 12
 CONTROL "Current Sprite", IDS_CSPRITE, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE
 | WS_TABSTOP, 144, 8, 56, 12
 CONTROL "Image", IDC_STATICTEXT8, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 80, 100, 52, 8
 CONTROL "Shade", IDC_STATICTEXT9, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 80, 124, 52, 8
 CONTROL "Palette", IDC_STATICTEXT1, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 80, 148, 52, 8
 CONTROL "Clipping", IDC_STATICTEXT20, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 80, 172, 52, 8
 CONTROL "Status", IDC_STATICTEXT12, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 80, 196, 52, 8
 CONTROL "X Repeat", IDC_STATICTEXT21, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 148, 100, 52, 8
 CONTROL "Y Repeat", IDC_STATICTEXT22, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 148, 124, 52, 8
 CONTROL "X Offset", IDC_STATICTEXT23, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 148, 148, 52, 8
 CONTROL "Y Offset", IDC_STATICTEXT24, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 148, 172, 52, 8
 CONTROL "Sector Num", IDS_CSECTNUM, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE |
WS_TABSTOP, 12, 196, 52, 12
 CONTROL "<<", IDS_FLAST, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP,
 136, 40, 12, 14
 CONTROL "Last", IDS_LAST, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP
, 148, 40, 24, 14
 CONTROL "Next", IDS_NEXT, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP
, 172, 40, 24, 14
 CONTROL ">>", IDS_FNEXT, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP,
 196, 40, 12, 14
 CONTROL "Frame5", IDC_STATICFRAME1, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 280, 8, 228,
 220
 CONTROL "Enable NW III TC sprite information", IDC_ENABLE, "button", BS_PUSHBUTTON | BS_CENTER | W
S_CHILD | WS_VISIBLE | WS_TABSTOP, 284, 12, 220, 13
 CONTROL "Edit22", IDC_INFOBOX, "edit", ES_LEFT | ES_MULTILINE | ES_READONLY | ES_WANTRETURN | WS_C
HILD | WS_VISIBLE | WS_VSCROLL | WS_TABSTOP, 284, 48, 220, 176
 CONTROL "Owner", IDC_STATICTEXT35, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 148, 196, 52, 7
 CONTROL "Owner", IDS_OWNER, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP, 148,
 204, 52, 12
 CONTROL "Update", IDS_UPDATE, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TAB
STOP, 138, 60, 68, 14
 CONTROL "Disable auto update", IDS_AUTOUPDATE, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS
_VISIBLE | WS_TABSTOP, 138, 78, 68, 14
 CONTROL "Text21\nTest21", IDC_INFOTITLE, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 284, 30, 220,
18
}




IDI_ICON1 ICON
{
 '00 00 01 00 01 00 20 20 00 01 00 00 00 00 A8 08'
 '00 00 16 00 00 00 28 00 00 00 20 00 00 00 40 00'
 '00 00 01 00 08 00 00 00 00 00 80 04 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 40 00 00 00 80 00 00 00 FF 00 00 00 00 20'
 '00 00 40 20 00 00 80 20 00 00 FF 20 00 00 00 40'
 '00 00 40 40 00 00 80 40 00 00 FF 40 00 00 00 60'
 '00 00 40 60 00 00 80 60 00 00 FF 60 00 00 00 80'
 '00 00 40 80 00 00 80 80 00 00 FF 80 00 00 00 A0'
 '00 00 40 A0 00 00 80 A0 00 00 FF A0 00 00 00 C0'
 '00 00 40 C0 00 00 80 C0 00 00 FF C0 00 00 00 FF'
 '00 00 40 FF 00 00 80 FF 00 00 FF FF 00 00 00 00'
```

```
'20 00 40 00 20 00 80 00 20 00 FF 00 20 00 00 20'
'20 00 40 20 20 00 80 20 20 00 FF 20 20 00 00 40'
'20 00 40 40 20 00 80 40 20 00 FF 40 20 00 00 60'
'20 00 40 60 20 00 80 60 20 00 FF 60 20 00 00 80'
'20 00 40 80 20 00 80 80 20 00 FF 80 20 00 00 A0'
'20 00 40 A0 20 00 80 A0 20 00 FF A0 20 00 00 C0'
'20 00 40 C0 20 00 80 C0 20 00 FF C0 20 00 00 FF'
'20 00 40 FF 20 00 80 FF 20 00 FF FF 20 00 00 00'
'40 00 40 00 40 00 80 00 40 00 FF 00 40 00 00 20'
'40 00 40 20 40 00 80 20 40 00 FF 20 40 00 00 40'
'40 00 40 40 40 00 80 40 40 00 FF 40 40 00 00 60'
'40 00 40 60 40 00 80 60 40 00 FF 60 40 00 00 80'
'40 00 40 80 40 00 80 80 40 00 FF 80 40 00 00 A0'
'40 00 40 A0 40 00 80 A0 40 00 FF A0 40 00 00 C0'
'40 00 40 C0 40 00 80 C0 40 00 FF C0 40 00 00 FF'
'40 00 40 FF 40 00 80 FF 40 00 FF FF 40 00 00 00'
'60 00 40 00 60 00 80 00 60 00 FF 00 60 00 00 20'
'60 00 40 20 60 00 80 20 60 00 FF 20 60 00 00 40'
'60 00 40 40 60 00 80 40 60 00 FF 40 60 00 00 60'
'60 00 40 60 60 00 80 60 60 00 FF 60 60 00 00 80'
'60 00 40 80 60 00 80 80 60 00 FF 80 60 00 00 A0'
'60 00 40 A0 60 00 80 A0 60 00 FF A0 60 00 00 C0'
'60 00 40 C0 60 00 80 C0 60 00 FF C0 60 00 00 FF'
'60 00 40 FF 60 00 80 FF 60 00 FF FF 60 00 00 00'
'80 00 40 00 80 00 80 00 80 00 FF 00 80 00 00 20'
'80 00 40 20 80 00 80 20 80 00 FF 20 80 00 00 40'
'80 00 40 40 80 00 80 40 80 00 FF 40 80 00 00 60'
'80 00 40 60 80 00 80 60 80 00 FF 60 80 00 00 80'
'80 00 40 80 80 00 80 80 80 00 FF 80 80 00 00 A0'
'80 00 40 A0 80 00 80 A0 80 00 FF A0 80 00 00 C0'
'80 00 40 C0 80 00 80 C0 80 00 FF C0 80 00 00 FF'
'80 00 40 FF 80 00 80 FF 80 00 FF FF 80 00 00 00'
'A0 00 40 00 A0 00 80 00 A0 00 FF 00 A0 00 00 20'
'A0 00 40 20 A0 00 80 20 A0 00 FF 20 A0 00 00 40'
'A0 00 40 40 A0 00 80 40 A0 00 FF 40 A0 00 00 60'
'A0 00 40 60 A0 00 80 60 A0 00 FF 60 A0 00 00 80'
'A0 00 40 80 A0 00 80 80 A0 00 FF 80 A0 00 00 A0'
'A0 00 40 A0 A0 00 80 A0 A0 00 FF A0 A0 00 00 C0'
'A0 00 40 C0 A0 00 80 C0 A0 00 FF C0 A0 00 00 FF'
'A0 00 40 FF A0 00 80 FF A0 00 FF FF A0 00 00 00'
'C0 00 40 00 C0 00 80 00 C0 00 FF 00 C0 00 00 20'
'C0 00 40 20 C0 00 80 20 C0 00 FF 20 C0 00 00 40'
'C0 00 40 40 C0 00 80 40 C0 00 FF 40 C0 00 00 60'
'C0 00 40 60 C0 00 80 60 C0 00 FF 60 C0 00 00 80'
'C0 00 40 80 C0 00 80 80 C0 00 FF 80 C0 00 00 A0'
'C0 00 40 A0 C0 00 80 A0 C0 00 FF A0 C0 00 00 C0'
'C0 00 40 C0 C0 00 80 C0 C0 00 FF C0 C0 00 00 FF'
'C0 00 40 FF C0 00 80 FF C0 00 FF FF C0 00 00 00'
'FF 00 40 00 FF 00 80 00 FF 00 FF 00 FF 00 00 20'
'FF 00 40 20 FF 00 80 20 FF 00 FF 20 FF 00 00 40'
'FF 00 40 40 FF 00 80 40 FF 00 FF 40 FF 00 00 60'
'FF 00 40 60 FF 00 80 60 FF 00 FF 60 FF 00 00 80'
'FF 00 40 80 FF 00 80 80 FF 00 FF 80 FF 00 00 A0'
'FF 00 40 A0 FF 00 80 A0 FF 00 FF A0 FF 00 00 C0'
'FF 00 40 C0 FF 00 80 C0 FF 00 FF C0 FF 00 00 FF'
'FF 00 40 FF FF 00 80 FF FF 00 FF FF FF 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 A0 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 A0 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 A0 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 A0 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 A0 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 A0 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 A0 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 A0 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 A0 A0 A0'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 A0 A0 A0'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
```

```
 '00 00 00 00 00 00 00 00 00 00 00 00 00 A0 A0 A0'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 A0 A0 A0'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 A0 A0 A0'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 A0 A0 A0 A0'
 'A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 A0 A0 A0 A0'
 'A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 A0 A0 A0 A0'
 'A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 A0 A0 A0 A0'
 'A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 A0 A0 A0'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 A0 A0 A0'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 A0 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 A0 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 A0 A0 00 00 00 00 A0 00'
 '00 00 00 A0 A0 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 A0 A0 A0 A0 A0 00 00 A0 00'
 '00 A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 A0 A0 A0 A0 A0 A0 A0 A0 A0'
 'A0 A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 A0 A0 A0 A0 A0 00 00 A0 00'
 '00 A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 A0 A0 00 00 00 00 A0 00'
 '00 00 00 A0 A0 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 A0 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 A0 A0 A0'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 A0 A0 A0'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 A0 A0 A0 A0'
 'A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 A0 A0 A0 A0'
 'A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 A0 A0 A0'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF'
 '7F FF FF FF 7F FF FF FF 7F FF FF FF 7F FF FF FF'
 '7F FF FF FF 7F FF FF FF 7F FF FF FF 7F FF FF FE'
 '3F FF FF FE 3F FF FF FE 3F FF FF FE 3F FF FF FE'
 '3F FF FF FC 1F FF FF FC 1F FF FF FC 1F FF FF FC'
 '1F FF FF FE 3F FF FF FE 3F FF FF FF 7F FF FF FF'
 '7F FF FF CF 79 FF FF 83 60 FF FF 80 00 FF FF 83'
 '60 FF FF CF 79 FF FF FF 7F FF FF FE 3F FF FF FE'
 '3F FF FF FC 1F FF FF FC 1F FF FF FE 3F FF'
}


DLG_TEST DIALOG 0, 0, 167, 72
STYLE DS_MODALFRAME | DS_3DLOOK | DS_CONTEXTHELP | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Optimisation processor"
FONT 8, "MS Sans Serif"
{
 CONTROL "Replace", IDC_R_REPLACE, "BUTTON", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS
_TABSTOP, 52, 52, 50, 14
 CONTROL "Close", IDC_R_CLOSE, "BUTTON", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TAB
STOP, 108, 52, 50, 14
 CONTROL "new image", IDC_R_NEW, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP,
104, 32, 56, 12
 CONTROL "original image", IDC_R_ORIGINAL, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS
_TABSTOP, 104, 12, 56, 13
 CONTROL "Original image", IDC_STATICTEXT33, "static", SS_RIGHT | WS_CHILD | WS_VISIBLE, 56, 12, 44
, 12
 CONTROL "New image", IDC_STATICTEXT34, "static", SS_RIGHT | WS_CHILD | WS_VISIBLE, 56, 32, 44, 12
 CONTROL "Ceilings", IDC_R_CEILINGS, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP
, 8, 4, 40, 12
 CONTROL "Floors", IDC_R_FLOORS, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 8,
20, 40, 12
 CONTROL "Walls", IDC_R_WALLS, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 8, 3
```

```
6, 40, 16
 CONTROL "Sprites", IDC_R_SPRITES, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP,
8, 56, 40, 12
}

DLG_EXTERNAL DIALOG 0, 0, 240, 134
STYLE DS_MODALFRAME | DS_3DLOOK | DS_CONTEXTHELP | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "External file manager"
FONT 8, "MS Sans Serif"
{
 CONTROL "Close", IDOK, "BUTTON", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 1
84, 20, 50, 14
 CONTROL "duke3d.exe", IDC_N_DUKE, "edit", ES_LEFT | ES_OEMCONVERT | WS_CHILD | WS_VISIBLE | WS_BOR
DER | WS_TABSTOP, 12, 52, 104, 13
 CONTROL "setup.exe", IDC_N_SETUP, "edit", ES_LEFT | ES_OEMCONVERT | WS_CHILD | WS_VISIBLE | WS_BOR
DER | WS_TABSTOP, 12, 68, 104, 12
 CONTROL "Please note that to use these functions, this program should be run from the same directo
ry as the duke3d.exe and utilities.\nDuke3D may not work with faked multipayer if you have demos in
 your current duke directory", IDC_STATICTEXT38, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 12, 4,
168, 40
 CONTROL "build.exe", IDC_N_BUILD, "edit", ES_LEFT | ES_OEMCONVERT | WS_CHILD | WS_VISIBLE | WS_BOR
DER | WS_TABSTOP, 12, 84, 104, 12
 CONTROL "editart.exe", IDC_N_EDITART, "edit", ES_LEFT | ES_OEMCONVERT | WS_CHILD | WS_VISIBLE | WS
_BORDER | WS_TABSTOP, 12, 100, 104, 12
 CONTROL "Duke 3D", IDC_RUN_DUKE, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_
TABSTOP, 124, 52, 44, 12
 CONTROL "Setup", IDC_RUN_SETUP, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_T
ABSTOP, 124, 68, 44, 12
 CONTROL "Build", IDC_RUN_BUILD, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_T
ABSTOP, 124, 84, 44, 12
 CONTROL "Editart", IDC_RUN_EDITART, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE |
WS_TABSTOP, 124, 100, 44, 11
 CONTROL "Players", IDC_N_PLAYERS, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER
| WS_TABSTOP, 188, 84, 40, 13
 CONTROL "Number of players", IDC_STATICTEXT39, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 184, 6
0, 48, 16
 CONTROL "duke3d.cfg", IDC_N_SETUPFILE, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TA
BSTOP, 12, 116, 104, 12
}


IDI_ICON2 ICON
{
'00 00 01 00 01 00 10 10 00 01 00 00 00 00 68 05'
'00 00 16 00 00 00 28 00 00 00 10 00 00 00 20 00'
'00 00 01 00 08 00 00 00 00 00 40 01 00 00 00 00'
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
'00 00 40 00 00 00 80 00 00 00 FF 00 00 00 00 20'
'00 00 40 20 00 00 80 20 00 00 FF 20 00 00 00 40'
'00 00 40 40 00 00 80 40 00 00 FF 40 00 00 00 60'
'00 00 40 60 00 00 80 60 00 00 FF 60 00 00 00 80'
'00 00 40 80 00 00 80 80 00 00 FF 80 00 00 00 A0'
'00 00 40 A0 00 00 80 A0 00 00 FF A0 00 00 00 C0'
'00 00 40 C0 00 00 80 C0 00 00 FF C0 00 00 00 FF'
'00 00 40 FF 00 00 80 FF 00 00 FF FF 00 00 00 00'
'20 00 40 00 20 00 80 00 20 00 FF 00 20 00 00 20'
'20 00 40 20 20 00 80 20 20 00 FF 20 20 00 00 40'
'20 00 40 40 20 00 80 40 20 00 FF 40 20 00 00 60'
'20 00 40 60 20 00 80 60 20 00 FF 60 20 00 00 80'
'20 00 40 80 20 00 80 80 20 00 FF 80 20 00 00 A0'
'20 00 40 A0 20 00 80 A0 20 00 FF A0 20 00 00 C0'
'20 00 40 C0 20 00 80 C0 20 00 FF C0 20 00 00 FF'
'20 00 40 FF 20 00 80 FF 20 00 FF FF 20 00 00 00'
'40 00 40 00 40 00 80 00 40 00 FF 00 40 00 00 20'
'40 00 40 20 40 00 80 20 40 00 FF 20 40 00 00 40'
'40 00 40 40 40 00 80 40 40 00 FF 40 40 00 00 60'
'40 00 40 60 40 00 80 60 40 00 FF 60 40 00 00 80'
'40 00 40 80 40 00 80 80 40 00 FF 80 40 00 00 A0'
'40 00 40 A0 40 00 80 A0 40 00 FF A0 40 00 00 C0'
'40 00 40 C0 40 00 80 C0 40 00 FF C0 40 00 00 FF'
'40 00 40 FF 40 00 80 FF 40 00 FF FF 40 00 00 00'
'60 00 40 00 60 00 80 00 60 00 FF 00 60 00 00 20'
'60 00 40 20 60 00 80 20 60 00 FF 20 60 00 00 40'
'60 00 40 40 60 00 80 40 60 00 FF 40 60 00 00 60'
'60 00 40 60 60 00 80 60 60 00 FF 60 60 00 00 80'
'60 00 40 80 60 00 80 80 60 00 FF 80 60 00 00 A0'
```

```
'60 00 40 A0 60 00 80 A0 60 00 FF A0 60 00 00 C0'
'60 00 40 C0 60 00 80 C0 60 00 FF C0 60 00 00 FF'
'60 00 40 FF 60 00 80 FF 60 00 FF FF 60 00 00 00'
'80 00 40 00 80 00 80 00 80 00 FF 00 80 00 00 20'
'80 00 40 20 80 00 80 20 80 00 FF 20 80 00 00 40'
'80 00 40 40 80 00 80 40 80 00 FF 40 80 00 00 60'
'80 00 40 60 80 00 80 60 80 00 FF 60 80 00 00 80'
'80 00 40 80 80 00 80 80 80 00 FF 80 80 00 00 A0'
'80 00 40 A0 80 00 80 A0 80 00 FF A0 80 00 00 C0'
'80 00 40 C0 80 00 80 C0 80 00 FF C0 80 00 00 FF'
'80 00 40 FF 80 00 80 FF 80 00 FF FF 80 00 00 00'
'A0 00 40 00 A0 00 80 00 A0 00 FF 00 A0 00 00 20'
'A0 00 40 20 A0 00 80 20 A0 00 FF 20 A0 00 00 40'
'A0 00 40 40 A0 00 80 40 A0 00 FF 40 A0 00 00 60'
'A0 00 40 60 A0 00 80 60 A0 00 FF 60 A0 00 00 80'
'A0 00 40 80 A0 00 80 80 A0 00 FF 80 A0 00 00 A0'
'A0 00 40 A0 A0 00 80 A0 A0 00 FF A0 A0 00 00 C0'
'A0 00 40 C0 A0 00 80 C0 A0 00 FF C0 A0 00 00 FF'
'A0 00 40 FF A0 00 80 FF A0 00 FF FF A0 00 00 00'
'C0 00 40 00 C0 00 80 00 C0 00 FF 00 C0 00 00 20'
'C0 00 40 20 C0 00 80 20 C0 00 FF 20 C0 00 00 40'
'C0 00 40 40 C0 00 80 40 C0 00 FF 40 C0 00 00 60'
'C0 00 40 60 C0 00 80 60 C0 00 FF 60 C0 00 00 80'
'C0 00 40 80 C0 00 80 80 C0 00 FF 80 C0 00 00 A0'
'C0 00 40 A0 C0 00 80 A0 C0 00 FF A0 C0 00 00 C0'
'C0 00 40 C0 C0 00 80 C0 C0 00 FF C0 C0 00 00 FF'
'C0 00 40 FF C0 00 80 FF C0 00 FF FF C0 00 00 00'
'FF 00 40 00 FF 00 80 00 FF 00 FF 00 FF 00 00 20'
'FF 00 40 20 FF 00 80 20 FF 00 FF 20 FF 00 00 40'
'FF 00 40 40 FF 00 80 40 FF 00 FF 40 FF 00 00 60'
'FF 00 40 60 FF 00 80 60 FF 00 FF 60 FF 00 00 80'
'FF 00 40 80 FF 00 80 80 FF 00 FF 80 FF 00 00 A0'
'FF 00 40 A0 FF 00 80 A0 FF 00 FF A0 FF 00 00 C0'
'FF 00 40 C0 FF 00 80 C0 FF 00 FF C0 FF 00 00 FF'
'FF 00 40 FF FF 00 80 FF FF 00 FF FF FF 00 00 00'
'00 00 00 00 00 00 A0 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 A0 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 A0 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 A0 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 A0 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 A0 A0 A0 00 00 00 00 00 00 00 00'
'00 00 00 00 00 A0 A0 A0 00 00 00 00 00 00 00 00'
'00 00 00 00 00 A0 A0 A0 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 A0 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 00 A0 00 00 00 00 00 00 00 00 00'
'00 00 A0 A0 00 00 A0 00 00 A0 A0 00 00 00 00 00'
'00 00 A0 A0 A0 A0 A0 A0 A0 A0 A0 00 00 00 00 00'
'00 00 A0 A0 00 00 A0 00 00 A0 A0 00 00 00 00 00'
'00 00 00 00 00 00 A0 00 00 00 00 00 00 00 00 00'
'00 00 00 00 00 A0 A0 A0 00 00 00 00 00 00 00 00'
'00 00 00 00 00 A0 A0 A0 00 00 00 00 00 00 FF 7F'
'00 00 FF 7F 00 00 FF 7F 00 00 FF 7F 00 00 FF 7F'
'00 00 FE 3F 00 00 FE 3F 00 00 FE 3F 00 00 FF 7F'
'00 00 FF 7F 00 00 F3 67 00 00 F0 07 00 00 F3 67'
'00 00 FF 7F 00 00 FE 3F 00 00 FE 3F 00 00'
}


OpenGLBT MENU
{
 POPUP "&File"
 {
  MENUITEM "&New", IDM_NEW
  MENUITEM "&Open", IDM_OPEN
  MENUITEM "&Save", IDM_SAVE
  MENUITEM "Save &As", IDM_SAVEAS
  MENUITEM SEPARATOR
  MENUITEM "&Group file manager", CM_Group
  MENUITEM "&Create patch file", CM_Patch
  MENUITEM "Optimise &con file", CM_OPTIMISE
  MENUITEM SEPARATOR
  MENUITEM "E&xit", CM_EXIT
 }

 POPUP "&Edit"
 {
```

```
  MENUITEM "&Wall and sector editor", CM_WallSect
  MENUITEM "&Sprite editor", CM_Sprite
  MENUITEM "&Optimisation processor", CM_Test
  MENUITEM "&External file manager", CM_External
 }

 POPUP "&Help"
 {
  MENUITEM "&Help Topics", IDM_HELP
  MENUITEM SEPARATOR
  MENUITEM "&About", CM_ABOUT
 }

}

OpenGLBT ACCELERATORS
{
 "O", IDM_OPEN, ASCII, NOINVERT
}

DLG_GROUP DIALOG 0, 0, 304, 224
STYLE DS_MODALFRAME | DS_3DLOOK | DS_CONTEXTHELP | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Group file manager"
FONT 8, "MS Sans Serif"
{
 CONTROL "Close", IDOK, "BUTTON", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 1
52, 56, 48, 16
 CONTROL "ListBox1", IDC_GROUP, "listbox", LBS_NOTIFY | LBS_HASSTRINGS | LBS_NOINTEGRALHEIGHT | LBS
_EXTENDEDSEL | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_VSCROLL | WS_TABSTOP, 4, 12, 88, 196
 CONTROL "ListBox2", IDC_GROUPSIZE, "listbox", LBS_NOINTEGRALHEIGHT | LBS_NOSEL | WS_CHILD | WS_VIS
IBLE | WS_BORDER | WS_TABSTOP, 92, 12, 44, 196
 CONTROL "", IDC_GROUPPATH, "edit", ES_LEFT | ES_MULTILINE | ES_AUTOVSCROLL | WS_CHILD | WS_VISIBLE
 | WS_BORDER | WS_TABSTOP, 144, 104, 152, 28
 CONTROL "Extract file", IDC_GROUPEXTRACT, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISI
BLE | WS_TABSTOP, 152, 84, 48, 16
 CONTROL "Name", IDC_STATICTEXT41, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 8, 4, 80, 8
 CONTROL "Size", IDC_STATICTEXT42, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 96, 4, 36, 8
 CONTROL "Add file", IDC_ADDFILE, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_
TABSTOP, 152, 144, 48, 16
 CONTROL "Delete file", IDC_DELETEFILE, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE
 | WS_TABSTOP, 232, 56, 48, 16
 CONTROL "Open", IDC_OPEN, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP
, 152, 12, 48, 16
 CONTROL "New", IDC_NEW, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP,
152, 32, 48, 16
 CONTROL "Frame1", IDC_STATICFRAME2, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 140, 4, 160,
 48
 CONTROL "Frame3", IDC_STATICFRAME27, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 140, 76, 16
0, 60
 CONTROL "Method 1", IDC_METHOD1, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_
TABSTOP, 144, 168, 36, 12
 CONTROL "Method 2", IDC_METHOD2, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_
TABSTOP, 144, 188, 36, 12
 CONTROL "Please select a method", IDC_METHODTEXT, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 184,
168, 112, 31
 CONTROL "Frame4", IDC_STATICFRAME28, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 140, 137, 1
60, 71
 CONTROL "Idle", IDC_GROUPTEXT, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 0, 212, 300, 8
 CONTROL "File name:\nFile size: 0\nNumber of entries: 0\nTotal data size: 0", IDC_GROUPNAME, "stat
ic", SS_CENTER | WS_CHILD | WS_VISIBLE, 204, 12, 92, 32
 CONTROL "File Addition Method", IDC_STATICTEXT58, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 208
, 144, 80, 16
 CONTROL "Extraction Destination Path", IDC_STATICTEXT60, "static", SS_CENTER | WS_CHILD | WS_VISIB
LE, 208, 84, 80, 16
 CONTROL "Frame4", IDC_STATICFRAME26, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 140, 53, 72
, 22
 CONTROL "Frame5", IDC_STATICFRAME29, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 213, 53, 87
, 22
}

RCDATA_1 RCDATA "header.stu"
RCDATA_2 RCDATA "footer.stu"

STUB DIALOG 0, 0, 305, 272
STYLE DS_MODALFRAME | DS_3DLOOK | DS_CONTEXTHELP | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Create patch file"
```

```
FONT 8, "MS Sans Serif"
{
 CONTROL "Create Patch", IDOK, "BUTTON", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TAB
STOP, 250, 24, 50, 14
 CONTROL "Cancel", IDCANCEL, "BUTTON", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABST
OP, 250, 44, 50, 14
 CONTROL "FILENAME", STUB_NAME, "edit", ES_LEFT | ES_UPPERCASE | WS_CHILD | WS_VISIBLE | WS_BORDER
| WS_TABSTOP, 164, 28, 68, 13
 CONTROL "FILENAME", STUB_PATCH, "edit", ES_LEFT | ES_UPPERCASE | WS_CHILD | WS_VISIBLE | WS_BORDER
 | WS_TABSTOP, 164, 74, 68, 13
 CONTROL "L.A. MELTDOWN", STUB_LAMEL, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABS
TOP, 54, 32, 88, 12
 CONTROL "SHRAPNEL CITY", STUB_SHRAP, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABS
TOP, 54, 72, 88, 12
 CONTROL "", STUB_1, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 58, 124, 8, 8
 CONTROL "1", STUB_W1, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP
, 92, 122, 16, 12
 CONTROL "", STUB_2, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 58, 138, 8, 8
 CONTROL "2", STUB_W2, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP
, 92, 136, 16, 12
 CONTROL "", STUB_3, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 58, 152, 8, 8
 CONTROL "3", STUB_W3, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP
, 92, 150, 16, 12
 CONTROL "", STUB_4, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 58, 166, 8, 8
 CONTROL "4", STUB_W4, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP
, 92, 164, 16, 12
 CONTROL "", STUB_5, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 58, 180, 8, 8
 CONTROL "5", STUB_W5, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP
, 92, 178, 16, 12
 CONTROL "", STUB_6, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 58, 194, 8, 8
 CONTROL "6", STUB_W6, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP
, 92, 192, 16, 12
 CONTROL "", STUB_7, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 58, 208, 8, 8
 CONTROL "7", STUB_W7, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP
, 92, 206, 16, 12
 CONTROL "", STUB_8, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 58, 222, 8, 8
 CONTROL "8", STUB_W8, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP
, 92, 220, 16, 12
 CONTROL "", STUB_9, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 58, 236, 8, 8
 CONTROL "9", STUB_W9, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP
, 92, 234, 16, 12
 CONTROL "", STUB_10, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 58, 250, 8, 8
 CONTROL "10", STUB_W10, "edit", ES_LEFT | ES_NUMBER | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABST
OP, 92, 248, 16, 12
 CONTROL "Inventory", STUB_INV, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 224
, 126, 64, 8
 CONTROL "Invenroty Left", STUB_INVL, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTO
P, 224, 140, 64, 8
 CONTROL "Inventory Right", STUB_INVR, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABST
OP, 224, 154, 64, 8
 CONTROL "Holoduke", STUB_HOLO, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 224
, 168, 64, 8
 CONTROL "Jetpack", STUB_JP, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 224, 1
82, 64, 8
 CONTROL "Night Vision", STUB_NV, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 2
24, 196, 64, 8
 CONTROL "Medikit", STUB_MED, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 224,
210, 64, 8
 CONTROL "Steroids", STUB_STER, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE, 224, 224, 64, 8
 CONTROL "Quick Kick", STUB_KICK, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 2
24, 238, 64, 8
 CONTROL "Disable", IDC_STATICTEXT44, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 50, 114, 26, 8
 CONTROL "LUNAR APOCALYPSE", STUB_LUNAR, "edit", ES_LEFT | WS_CHILD | WS_VISIBLE | WS_BORDER | WS_T
ABSTOP, 54, 52, 88, 12
 CONTROL "Weapon 1", IDC_STATICTEXT45, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 16, 124, 40, 8
 CONTROL "Weapon 2", IDC_STATICTEXT46, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 16, 138, 40, 8
 CONTROL "Weapon 3", IDC_STATICTEXT47, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 16, 152, 40, 8
 CONTROL "Weapon 4", IDC_STATICTEXT2, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 16, 166, 40, 8
 CONTROL "Weapon 5", IDC_STATICTEXT48, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 16, 180, 40, 8
 CONTROL "Weapon 6", IDC_STATICTEXT49, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 16, 194, 40, 8
 CONTROL "Weapon 7", IDC_STATICTEXT50, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 16, 208, 40, 8
 CONTROL "Weapon 8", IDC_STATICTEXT51, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 16, 222, 40, 8
 CONTROL "Weapon 10", IDC_STATICTEXT52, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 16, 250, 40, 8
 CONTROL "Weapon 9", IDC_STATICTEXT53, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 16, 236, 40, 8
 CONTROL "Capitals will use the default red font, lower case will use a metallic font instead.", ID
C_STATICTEXT13, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 12, 12, 132, 16
```

```
 CONTROL "Episode 1", IDC_STATICTEXT14, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 14, 32, 36, 12
 CONTROL "Episode 2", IDC_STATICTEXT3, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 14, 52, 36, 12
 CONTROL "Episode 3", IDC_STATICTEXT54, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 14, 72, 36, 12
 CONTROL "Please enter the desired file name for the patched executionable.", IDC_STATICTEXT15, "st
atic", SS_CENTER | WS_CHILD | WS_VISIBLE, 160, 46, 76, 24
 CONTROL "New Weapon Number", IDC_STATICTEXT11, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 84, 96
, 32, 26
 CONTROL "Disable", IDC_STATICTEXT1, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 216, 116, 26, 8
 CONTROL "Frame1", IDC_STATICFRAME19, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 4, 91, 124,
 176
 CONTROL "Frame2", IDC_STATICFRAME20, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 196, 91, 10
6, 176
 CONTROL "Frame3", IDC_STATICFRAME21, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 4, 4, 150,
88
 CONTROL "Weapons", IDC_STATICTEXT22, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 8, 95, 34, 9
 CONTROL "Inventory", IDC_STATICTEXT23, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 204, 95, 32, 9
 CONTROL "Please enter the desired file name for the patch.", IDC_STATICTEXT55, "static", SS_CENTER
 | WS_CHILD | WS_VISIBLE, 160, 10, 76, 16
 CONTROL "Frame4", IDC_STATICFRAME22, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 153, 4, 91,
 88
 CONTROL "This function is for D3D v1.3d only.\n\nNo responsibility is taken for the use of the pro
gram produced by this one.", IDC_STATICTEXT56, "static", SS_CENTER | WS_CHILD | WS_VISIBLE, 132, 15
0, 60, 48
}

TEST DIALOG 0, 0, 240, 100
STYLE DS_MODALFRAME | DS_3DLOOK | DS_CONTEXTHELP | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Optimise con file"
FONT 8, "MS Sans Serif"
{
 CONTROL "Close", IDOK, "BUTTON", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 1
84, 6, 50, 14
 CONTROL "Status:", IDC_STATICTEXT44, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 8, 83, 24, 8
 CONTROL "Browse", IDC_BORIG, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABS
TOP, 136, 10, 32, 12
 CONTROL "GAME.CON", IDC_ORIG, "edit", ES_LEFT | ES_UPPERCASE | WS_CHILD | WS_VISIBLE | WS_BORDER |
 WS_TABSTOP, 60, 10, 72, 12
 CONTROL "Original file", IDC_STATICTEXT41, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 12, 12, 48,
8
 CONTROL "Optimised file", IDC_STATICTEXT42, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 12, 34, 48,
 8
 CONTROL "Browse", IDC_BLOG, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABST
OP, 136, 56, 32, 12
 CONTROL "TEST.CON", IDC_OPTIM, "edit", ES_LEFT | ES_UPPERCASE | WS_CHILD | WS_VISIBLE | WS_BORDER
| WS_TABSTOP, 60, 32, 72, 12
 CONTROL "TEST.LOG", IDC_LOG, "edit", ES_LEFT | ES_UPPERCASE | WS_CHILD | WS_VISIBLE | WS_BORDER |
WS_TABSTOP, 60, 56, 72, 12
 CONTROL "Frame2", IDC_STATICFRAME24, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 4, 27, 172,
 24
 CONTROL "Frame3", IDC_STATICFRAME23, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 4, 3, 172,
25
 CONTROL "Frame4", IDC_STATICFRAME2, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 4, 50, 172,
23
 CONTROL "Log file", IDC_CHECKLOG, "button", BS_AUTOCHECKBOX | BS_LEFTTEXT | WS_CHILD | WS_VISIBLE
| WS_TABSTOP, 12, 56, 40, 12
 CONTROL "Optimise", IDC_START, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TA
BSTOP, 184, 32, 50, 14
 CONTROL "Verbose", IDC_VERBOSE, "button", BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 18
4, 54, 44, 14
 CONTROL "Idle", IDC_STATUS, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 32, 83, 200, 8
 CONTROL "Frame5", IDC_STATICFRAME25, "static", SS_ETCHEDFRAME | WS_CHILD | WS_VISIBLE, 4, 77, 232,
 20
 CONTROL "Browse", IDC_BOPTIM, "button", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TAB
STOP, 136, 32, 32, 12
}

ABOUT DIALOG 0, 0, 337, 335
STYLE DS_MODALFRAME | DS_3DLOOK | DS_CONTEXTHELP | WS_POPUP | WS_VISIBLE | WS_CAPTION
CAPTION "About"
FONT 8, "MS Sans Serif"
{
 CONTROL "OK", IDOK, "BUTTON", BS_PUSHBUTTON | BS_CENTER | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 144,
 276, 50, 14
 CONTROL "Please note:\n  Please use a resolution of 1024x800 or greater as some of the dialogue bo
xes are rather large.\n\n", IDC_STATICTEXT40, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 16, 28, 30
4, 16
```

```
 CONTROL "Controls (these buttons are for the main screen only, which is the one with the map)\n\tE
SC\tquit\n\tF1\tthis information screen\n\t1\ttoggle one sided wall verteces\n\t2\t2d mode\n\t3\t3d
 mode\n\t4\ttoggle wall verteces\n\t5\topens sprite editing window\n\t6\tprevious sprite\n\t7\tnext
 sprite\n\t8\tprevious sector\n\t9\tnext sector\n\t0/o\topens main editing window (walls, sectors a
nd file management)\n\t-=_\tprevious sprite\n\t=/+\tnext sprite\n\te\topens the external program ma
nager\n\tup\tscrolls map up\n\tdown\tscrolls map down\n\tleft\tscrolls map left\n\tright\tscrolls m
ap right\n\nA menu has been added, as well as group file extraction, patch file creation and con fi
le optimisation.  This about screen has also been changed to a dialogue box.", IDC_STATICTEXT38, "s
tatic", SS_LEFT | WS_CHILD | WS_VISIBLE, 16, 48, 304, 188
 CONTROL "This is the site address at the time of release of this version.", IDC_STATICTEXT57, "sta
tic", SS_CENTER | WS_CHILD | WS_VISIBLE, 72, 300, 192, 9
 CONTROL "http://members.xoom.com/HCAD", IDC_EDIT50, "edit", ES_LEFT | ES_READONLY | WS_CHILD | WS_
VISIBLE | WS_TABSTOP, 112, 316, 112, 12
 CONTROL "The speed of the group file extraction has been increased again, and addiction and delete
ion of files from the group file has been added.\nThis could be the last version of OpenGL Build To
uch.  I may however add faster deletion and addition code to the group file manager.", IDC_STATICTE
XT42, "static", SS_LEFT | WS_CHILD | WS_VISIBLE, 16, 234, 304, 34
 CONTROL "OpenGL Build Touch v1.9.1\n© James Ferry 1999", IDC_STATICTEXT59, "static", SS_CENTER | W
S_CHILD | WS_VISIBLE, 16, 8, 304, 16
}
```

OpenGL Build Touch Source Code

cube2.cpp

```
/*
#define IDH_extra 2
#define IDH_newmap 3
#define IDH_newsector 4
#define IDH_newwall 7
#define IDH_owner 1
#define IDH_paralax 5
#define IDH_sloped 6
//*/
///*
/*
#include <algorithm>
#include <vector>
#include <iterator>
using namespace std;
template<class Arg>
struct all_true : public unary_function<Arg, bool>
{
   bool operator()(const Arg& x){ return 1; }
};
*/
#include "defines.h"
#include "cube.h"
#include <mem.h>
#include <stdio.h>
#include <dir.h>
#include <process.h>
#include <windows.h>
#include <commctrl.h>
#include <float.h>
#include <GL/gl.h>
#include "test.h"

#define DLG_DELETEITEM 1
#define DLG_TEST 3
#define GETDIRLISTING DlgDirList(hwndDlg,"*.map",ID_LOADNAME,0,DDL_ARCHIVE|DDL_DIRECTORY);
#define OPTIMIZED_WINDOW IS_ON
char info_enable=0;
char auto_update=0;
char first_load=1;
char szFile[260];       // buffer for filename
#if (processor_type == 586)
#pragma message ("OpenGL Build Touch Version 1.9.1 - © James Ferry 1999")
char *windowName = "OpenGL Build Touch Version 1.9.1 - © James Ferry 1999";
#elif (processor_type == 486)
#pragma message ("OpenGL Build Touch Version 1.9.1 - © James Ferry 1999 - i486 version")
char *windowName = "OpenGL Build Touch Version 1.9.1 - © James Ferry 1999 - i486 version";
#else
char *windowName = "OpenGL Build Touch Version 1.9.1 - © James Ferry 1999";
#endif

#ifdef OPTIMIZED_WINDOW
   #define ViewWallBox glViewport(500, 500, 200, 200);
   #define RightView glViewport(500, 0, 200, 500);
   #define FrontView glViewport(0, 0, 500, 500);
   #define TopView glViewport(0, 500, 500, 200);
   #define GLVIEWPORT3D glViewport(0, 0, 700, 700);
#else
   #define ViewWallBox glViewport(700, 500, 200, 200);
   #define RightView glViewport(700, 0, 200, 500);
   #define FrontView glViewport(200, 0, 500, 500);
   #define TopView glViewport(200, 500, 500, 200);
   #define GLVIEWPORT3D glViewport(200, 0, 700, 700);
#endif
//char *old_dir;
int CURSECT=0;
int CURSPRITE=0;
int CURWALL=0;
    HWND hWnd;
RECT rect;
char Redraw_all=0;
char show_one_side3d=0;
char show_wall3d=0;
#ifdef FLOATING_POINT
   #define VERTEXxy glVertex2f(x,y);
```

```
#else
   #define VERTEXxy glVertex2i(x,y);
#endif


#define BLACK_INDEX     0
#define RED_INDEX       13
#define GREEN_INDEX     14
#define YELLOW_INDEX     15
#define WHITE_INDEX     9
#define BLUE_INDEX      16
float  latinc=0.0F,longinc=0.0F;
long mapversion,posx,posy,posz;
int ang, cursectnum;
int numsectors,numwalls,numsprites;
//long ceiling, floor;
typedef  struct
{
   short wallptr, wallnum;
   long ceilingz, floorz;
   short ceilingstat, floorstat;
   short ceilingpicnum, ceilingheinum;
   signed  char ceilingshade;
   char ceilingpal, ceilingxpanning, ceilingypanning;
   short floorpicnum, floorheinum;
   signed  char floorshade;
   char floorpal, floorxpanning, floorypanning;
   char visibility, filler;
   short lotag, hitag, extra;
} sectortype;
sectortype sector[1024];

typedef  struct
{
   long x, y;
   short point2, nextwall, nextsector, cstat;
   short picnum, overpicnum;
   signed  char shade;
   char pal, xrepeat, yrepeat, xpanning, ypanning;
   short lotag, hitag, extra;
} walltype;
walltype wall[8192];


typedef  struct
{
   long x, y, z;
//   short cstat, picnum;
   unsigned  short cstat, picnum;
   signed  char shade;
   char pal, clipdist, filler;
   unsigned  char xrepeat, yrepeat;
   signed  char xoffset, yoffset;
   short sectnum, statnum;
   short ang, owner, xvel, yvel, zvel;
   short lotag, hitag, extra;
} spritetype;
spritetype sprite[4096];




int a;
HINSTANCE hinst;
PAINTSTRUCT ps;


char  *className = "OpenGL Build Touch by James Ferry";
char  *classMenu = "OpenGLBT";
int winX = 0, winY = 0;


int minX = 0, minY = 0;
#ifdef OPTIMIZED_WINDOW
   int maxX = 700, maxY = 700;
```

```
#else
   int maxX = 900, maxY = 700;
#endif



HDC hDC;
HGLRC hGLRC;
HPALETTE hPalette;

void
init(void)
{
    /* set viewing projection */
    glMatrixMode(GL_PROJECTION);
    glFrustum(-100.0F, 100.0F, -100.0F, 100.0F, 100.0F, 800.0F);

    /* position viewer */
   glMatrixMode(GL_MODELVIEW);
   glTranslatef(0.0F, 0.0F, -500.0F);

}




void
redraw3d(void)
{
register int a,b,c,e;
register int x,y,z;
   glMatrixMode(GL_MODELVIEW);
    glRotatef(-45.0F,1.0F,0.0F,0.0F);

   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
//   glIndexi(RED_INDEX);
// #define   BLUE_COLOUR glColor3b(0,0,0);
//   BLUE_COLOUR
glIndexi(BLUE_INDEX);
glColor3b(0,0,0);

GLVIEWPORT3D

for(a=0;a<numsectors;a++){
      c=sector[a].wallnum;
      e=sector[a].wallptr;
    glBegin(GL_LINE_LOOP);
      for(b=0;b<c;b++)
      {
      x=wall[e].x/250;
      y=-wall[e].y/250;
      z=-sector[a].ceilingz/500;
      e=wall[e].point2;
      glVertex3i(x,y,z);
      }
    glEnd();
    glBegin(GL_LINE_LOOP);
      for(b=0;b<c;b++)
      {
      x=wall[e].x/250;
      y=-wall[e].y/250;
      z=-sector[a].floorz/500;
      e=wall[e].point2;
      glVertex3i(x,y,z);
      }
    glEnd();
   }



if(show_wall3d)
{
   for(a=0;a<numsectors;a++){
      c=sector[a].wallnum;
```

```
      e=sector[a].wallptr;
      for(b=0;b<c;b++)
      {
         glBegin(GL_LINES);
if(wall[e].nextsector==-1&&wall[wall[e].point2].nextsector==-1)
{
if(show_one_side3d)
{
         x=wall[wall[e].point2].x/250;
         y=-wall[wall[e].point2].y/250;
         z=-sector[a].ceilingz/500;
         glVertex3i(x,y,z);
         x=wall[wall[e].point2].x/250;
         y=-wall[wall[e].point2].y/250;
         z=-sector[a].floorz/500;
         glVertex3i(x,y,z);
}
}
else
{
         x=wall[e].x/250;
         y=-wall[e].y/250;
         z=-sector[a].floorz/500;
         glVertex3i(x,y,z);
         x=wall[e].x/250;
         y=-wall[e].y/250;
         z=-sector[wall[e].nextsector].floorz/500;
         glVertex3i(x,y,z);
         x=wall[e].x/250;
         y=-wall[e].y/250;
         z=-sector[a].ceilingz/500;
         glVertex3i(x,y,z);
         x=wall[e].x/250;
         y=-wall[e].y/250;
         z=-sector[wall[e].nextsector].ceilingz/500;
         glVertex3i(x,y,z);
}
         glEnd();
         e=wall[e].point2;
         }
      }
   }
    SwapBuffers(hDC);
   glMatrixMode(GL_MODELVIEW);
    glRotatef(45.0F,1.0F,0.0F,0.0F);
}

void Right(void)
{
register int a,b,c,e;
#ifdef FLOATING_POINT
register float x,y;
#else
register int x,y;
#endif
    RightView
   glMatrixMode(GL_MODELVIEW);
   glTranslatef(0.0F, 0.0F, 400.0F);
   glIndexi(BLACK_INDEX);
   glColor3b(0,0,0);
   glBegin(GL_QUADS);
   glVertex2i(-100,-100);
   glVertex2i(100,-100);
   glVertex2i(100,100);
   glVertex2i(-100,100);
   glEnd();
   glMatrixMode(GL_MODELVIEW);
   glTranslatef(0.0F, 0.0F, -400.0F);

   glIndexi(BLUE_INDEX);
   glColor3b(0,0,255);
   for(a=0;a<numsectors;a++){
      c=sector[a].wallnum;
      e=sector[a].wallptr;
      for(b=0;b<c;b++)
      {
```

```
         glBegin(GL_LINE_STRIP);
         y=-wall[wall[e].point2].y/100+latinc;
         x=-sector[a].ceilingz/500;
         VERTEXxy
         y=-wall[e].y/100+latinc;
         x=-sector[a].ceilingz/500;
         VERTEXxy
         y=-wall[e].y/100+latinc;
         x=-sector[a].floorz/500;
         VERTEXxy
         y=-wall[wall[e].point2].y/100+latinc;
         x=-sector[a].floorz/500;
         VERTEXxy
         e=wall[e].point2;
         glEnd();
      }
   }
   glIndexi(WHITE_INDEX);
   glColor3b(255,255,255);
   glBegin(GL_LINES);
         y=-sprite[CURSPRITE].y/100+latinc-2000;
         x=-sprite[CURSPRITE].z/500;
         VERTEXxy
         y=-sprite[CURSPRITE].y/100+latinc+2000;
         x=-sprite[CURSPRITE].z/500;
         VERTEXxy
         y=-sprite[CURSPRITE].y/100+latinc;
         x=-sprite[CURSPRITE].z/500-2000;
         VERTEXxy
         y=-sprite[CURSPRITE].y/100+latinc;
         x=-sprite[CURSPRITE].z/500+2000;
         VERTEXxy
   glEnd();
}

void Top(void)
{
register int a,b,c,e;
#ifdef FLOATING_POINT
register float x,y;
#else
register int x,y;
#endif
    TopView
   glMatrixMode(GL_MODELVIEW);
   glTranslatef(0.0F, 0.0F, 400.0F);
   glIndexi(BLACK_INDEX);
   glColor3b(0,0,0);
   glBegin(GL_QUADS);
   glVertex2i(-100,-100);
   glVertex2i(100,-100);
   glVertex2i(100,100);
   glVertex2i(-100,100);
   glEnd();
   glMatrixMode(GL_MODELVIEW);
   glTranslatef(0.0F, 0.0F, -400.0F);
glIndexi(BLUE_INDEX);
glColor3b(0,0,255);

   for(a=0;a<numsectors;a++){
      c=sector[a].wallnum;
      e=sector[a].wallptr;
      for(b=0;b<c;b++)
      {
         glBegin(GL_LINE_STRIP);
         x=wall[wall[e].point2].x/100-longinc;
         y=-sector[a].ceilingz/500;
         VERTEXxy
         x=wall[e].x/100-longinc;
         y=-sector[a].ceilingz/500;
         VERTEXxy
         x=wall[e].x/100-longinc;
         y=-sector[a].floorz/500;
         VERTEXxy
         x=wall[wall[e].point2].x/100-longinc;
         y=-sector[a].floorz/500;
```

```
        VERTEXxy
        glEnd();
        e=wall[e].point2;
      }
   }
   glIndexi(WHITE_INDEX);
   glColor3b(255,255,255);
   glBegin(GL_LINES);
        x=sprite[CURSPRITE].x/100-longinc+2000;
        y=-sprite[CURSPRITE].z/500;
        VERTEXxy
        x=sprite[CURSPRITE].x/100-longinc-2000;
        y=-sprite[CURSPRITE].z/500;
        VERTEXxy
        x=sprite[CURSPRITE].x/100-longinc;
        y=-sprite[CURSPRITE].z/500+2000;
        VERTEXxy
        x=sprite[CURSPRITE].x/100-longinc;
        y=-sprite[CURSPRITE].z/500-2000;
        VERTEXxy
   glEnd();
}

void Front(void)
{
register int a,b,c,e;
#ifdef FLOATING_POINT
register float x,y;
#else
register int x,y;
#endif
   FrontView
   glMatrixMode(GL_MODELVIEW);
   glTranslatef(0.0F, 0.0F, 400.0F);
   glIndexi(BLACK_INDEX);
   glColor3b(0,0,0);
   glBegin(GL_QUADS);
   glVertex2i(-100,-100);
   glVertex2i(100,-100);
   glVertex2i(100,100);
   glVertex2i(-100,100);
   glEnd();
   glMatrixMode(GL_MODELVIEW);
   glTranslatef(0.0F, 0.0F, -400.0F);

   glIndexi(BLUE_INDEX);
   glColor3b(0,0,255);
for(a=0;a<numsectors;a++){
      c=sector[a].wallnum;
      e=sector[a].wallptr;
    glBegin(GL_LINE_LOOP);
      for(b=0;b<c;b++)
      {
      x=wall[e].x/100-longinc;
      y=-wall[e].y/100+latinc;
      e=wall[e].point2;
      VERTEXxy
/*#ifdef FLOATING_POINT
#define VERTEX glVertex2f(x,y);
#else
#define VERTEX glVertex2i(x,y);
#endif
*/
      }
    glEnd();
   }
   glIndexi(WHITE_INDEX);
   glColor3b(255,255,255);
   glBegin(GL_LINES);
        x=sprite[CURSPRITE].x/100-longinc+2000;
        y=-sprite[CURSPRITE].y/100+latinc;
        VERTEXxy
        x=sprite[CURSPRITE].x/100-longinc-2000;
        y=-sprite[CURSPRITE].y/100+latinc;
        VERTEXxy
        x=sprite[CURSPRITE].x/100-longinc;
```

```
         y=-sprite[CURSPRITE].y/100+2000+latinc;
         VERTEXxy
         x=sprite[CURSPRITE].x/100-longinc;
         y=-sprite[CURSPRITE].y/100-2000+latinc;
         VERTEXxy
   glEnd();
}

void FrontSingle(int index)
{
register int b,c,e;
#ifdef FLOATING_POINT
register float x,y;
#else
register int x,y;
#endif
   FrontView
   glIndexi(index);
      c=sector[CURSECT].wallnum;
      e=sector[CURSECT].wallptr;
    glBegin(GL_LINE_LOOP);
      for(b=0;b<c;b++)
      {
      x=wall[e].x/100-longinc;
      y=-wall[e].y/100+latinc;
      e=wall[e].point2;
      VERTEXxy
      }
    glEnd();
}

void FrontWall(int index)
{
#ifdef FLOATING_POINT
register float x,y;
#else
register int x,y;
#endif
   FrontView
   glIndexi(index);
    glBegin(GL_LINES);
      x=wall[CURWALL].x/100-longinc;
      y=-wall[CURWALL].y/100+latinc;
      VERTEXxy
      x=wall[wall[CURWALL].point2].x/100-longinc;
      y=-wall[wall[CURWALL].point2].y/100+latinc;
      VERTEXxy
    glEnd();
}

void resize(void)
{
    /* set viewport to cover the window */
//    glViewport(0, 0, 700, 700);
//    glViewport(0, 0, winWidth, winHeight);
     GetClientRect(    hWnd,    &rect    );
//    glViewport(0, 0, rect.right, rect.bottom);
}




BOOL bSetupPixelFormat(HDC hdc)
{
    PIXELFORMATDESCRIPTOR pfd, *ppfd;
    int pixelformat;

    ppfd = &pfd;

    ppfd->nSize = sizeof(PIXELFORMATDESCRIPTOR);
    ppfd->nVersion = 1;
    ppfd->dwFlags = PFD_DRAW_TO_WINDOW | PFD_SUPPORT_OPENGL |
                        PFD_DOUBLEBUFFER;
    ppfd->dwLayerMask = PFD_MAIN_PLANE;
    ppfd->iPixelType = PFD_TYPE_COLORINDEX;
    ppfd->cColorBits = 8;
```

```
    ppfd->cDepthBits = 16;
    ppfd->cAccumBits = 0;
    ppfd->cStencilBits = 0;

    if ( (pixelformat = ChoosePixelFormat(hdc, ppfd)) == 0 )
    {MessageBox(NULL, "ChoosePixelFormat failed", "Error", MB_OK);return FALSE;}

    if (SetPixelFormat(hdc, pixelformat, ppfd) == 0)
    {MessageBox(NULL, "SetPixelFormat failed", "Error", MB_OK);return FALSE;}

    return 1;

}

void
setupPalette(HDC hDC)
{
bSetupPixelFormat(hDC);
}

char szItemName[260];
char file[260];
int load(void)
{
   FILE *fil;
   if ((fil=fopen(szFile,"rb")) == NULL)
   {
if(first_load)
{
first_load=0;
*strcat(szFile,".map");
   if ((fil=fopen(szFile,"rb")) == NULL)
   {
      return 1;
   }
}
else
{
MessageBox(WindowFromDC(hDC), "Cannot open input file.", "File Error While Loading",MB_OK|MB_ICONEX
CLAMATION);
return 1;
}
   }
   {
      //Load map version number (current version is 7L)
   fread(&mapversion,4,1,fil);
      //Load starting position
   fread(&posx,4,1,fil);
   fread(&posy,4,1,fil);
   fread(&posz,4,1,fil);          //Note: Z coordinates are all shifted up 4
   fread(&ang,2,1,fil);           //All angles are from 0-2047, clockwise
   fread(&cursectnum,2,1,fil);    //Sector of starting point
      //Load all sectors (see sector structure described below)
   fread(&numsectors,2,1,fil);
   fread(sector,sizeof(sectortype),numsectors,fil);
      //Load all walls (see wall structure described below)
   fread(&numwalls,2,1,fil);
   fread(wall,sizeof(walltype),numwalls,fil);
      //Load all sprites (see sprite structure described below)
   fread(&numsprites,2,1,fil);
   fread(sprite,sizeof(spritetype),numsprites,fil);
   fclose(fil);
   }
   return 0;
}
void save(void)
{
   FILE *fil;
//   if ((fil=fopen(file,"wb")) == NULL)
   if ((fil=fopen(szFile,"wb")) == NULL)
   {
   MessageBox(WindowFromDC(hDC), "Sorry, I cannot open that file.", "File Error While Saving",MB_OK
);
   }
   else
   {
```

```
   fwrite(&mapversion,4,1,fil);

      //Load starting position
   fwrite(&posx,4,1,fil);
   fwrite(&posy,4,1,fil);
   fwrite(&posz,4,1,fil);          //Note: Z coordinates are all shifted up 4
   fwrite(&ang,2,1,fil);           //All angles are from 0-2047, clockwise
   fwrite(&cursectnum,2,1,fil);    //Sector of starting point
      //Load all sectors (see sector structure described below)
   fwrite(&numsectors,2,1,fil);
   fwrite(sector,sizeof(sectortype),numsectors,fil);
      //Load all walls (see wall structure described below)
   fwrite(&numwalls,2,1,fil);
   fwrite(wall,sizeof(walltype),numwalls,fil);
      //Load all sprites (see sprite structure described below)
   fwrite(&numsprites,2,1,fil);
   fwrite(sprite,sizeof(spritetype),numsprites,fil);
   fclose(fil);
   }
}



char callme=BST_CHECKED;
int szItemNum;

void SetupSectorProc(HWND hwndDlg)
{
/*if(filestatus==0)
{
sprintf(szItemName,"%c%s%c is currently loaded",34,file,34);
SetDlgItemText(hwndDlg,ID_MAPNAME,szItemName);
filestatus=1;
}
*/
char repstring[80];
char repstring2[80];
if(sector[CURSECT].lotag>=10000)
sprintf(repstring2,"Plays sound %i once",sector[CURSECT].lotag-10000);

sprintf(repstring,"Sector type - %s",
sector[CURSECT].lotag==0?"Normal":
sector[CURSECT].lotag==1?"Water - above surface":
sector[CURSECT].lotag==2?"Water - below surface":
sector[CURSECT].lotag==9?"Sliding Star Trek door":
sector[CURSECT].lotag==15?"Elevator transport":
sector[CURSECT].lotag==16?"Elevator platform down":
sector[CURSECT].lotag==17?"Elevator platform up":
sector[CURSECT].lotag==18?"Elevator down":
sector[CURSECT].lotag==19?"Elevator up":
sector[CURSECT].lotag==20?"Ceiling door":
sector[CURSECT].lotag==21?"Floor door":
sector[CURSECT].lotag==22?"Split door":
sector[CURSECT].lotag==23?"Swinging door":
sector[CURSECT].lotag==25?"Sliding door":
sector[CURSECT].lotag==26?"Split Star Trek door":
sector[CURSECT].lotag==27?"Bridge":
sector[CURSECT].lotag==28?"Drop floor":
sector[CURSECT].lotag==29?"Teeth floor":
sector[CURSECT].lotag==30?"Rotating and rising bridge":
sector[CURSECT].lotag==31?"Two way train":
sector[CURSECT].lotag==32767?"Secret room":
sector[CURSECT].lotag==-1?"End level":
sector[CURSECT].lotag>=10000?repstring2:
"Unknown"
);
SetDlgItemText(hwndDlg,IDC_SINFO,repstring);
SetDlgItemText(hwndDlg,IDC_SINFOBOX,
"\
 1   - Water - above surface\r\n\
 2   - Water - below surface\r\n\
 9   - Sliding Star Trek door\r\n\
 15  - Elevator transport (SE 17)\r\n\
```

```
 16  - Elevator platform down\r\n\
 17  - Elevator platform up\r\n\
 18  - Elevator down\r\n\
 19  - Elevator up\r\n\
 20  - Ceiling door\r\n\
 21  - Floor door\r\n\
 22  - Split door\r\n\
 23  - Swinging door\r\n\
 25  - Sliding door\r\n\
 26  - Split Star Trek door\r\n\
 27  - Bridge (SE 21)\r\n\
 28  - Drop floor (SE 21)\r\n\
 29  - Teeth door (SE 22)\r\n\
 30  - Rotating and rising bridge\r\n\
 31  - Two way train\r\n\
 10000+S - Plays sound S once\r\n\
 32767     - Secret room\r\n\
 65535     - End level");

sprintf(szItemName,"%li",posx);
SetDlgItemText(hwndDlg,IDP_X,szItemName);
sprintf(szItemName,"%li",posy);
SetDlgItemText(hwndDlg,IDP_Y,szItemName);
sprintf(szItemName,"%li",posz);
SetDlgItemText(hwndDlg,IDP_Z,szItemName);
SetDlgItemInt(hwndDlg,IDP_ANG,ang,TRUE);
SetDlgItemInt(hwndDlg,IDP_SECTOR,cursectnum,TRUE);

SetDlgItemInt(hwndDlg,ID_SECTORNUM,CURSECT,TRUE);
SetDlgItemInt(hwndDlg,ID_FIRSTWALL,sector[CURSECT].wallptr,TRUE);
SetDlgItemInt(hwndDlg,ID_NUMWALL,sector[CURSECT].wallnum,TRUE);
SetDlgItemInt(hwndDlg,ID_VIS,sector[CURSECT].visibility,TRUE);
SetDlgItemInt(hwndDlg,ID_LOTAG,sector[CURSECT].lotag,TRUE);
SetDlgItemInt(hwndDlg,ID_HITAG,sector[CURSECT].hitag,TRUE);
SetDlgItemInt(hwndDlg,ID_EXTRA,sector[CURSECT].extra,TRUE);

if(sector[CURSECT].floorstat&1)
   CheckDlgButton(hwndDlg,ID_FPARALAX,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,ID_FPARALAX,BST_UNCHECKED);
if(sector[CURSECT].floorstat&2)
   CheckDlgButton(hwndDlg,ID_FSLOPE,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,ID_FSLOPE,BST_UNCHECKED);
if(sector[CURSECT].floorstat&4)
   CheckDlgButton(hwndDlg,ID_FSWAP,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,ID_FSWAP,BST_UNCHECKED);
if(sector[CURSECT].floorstat&8)
   CheckDlgButton(hwndDlg,ID_FSMOOTH,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,ID_FSMOOTH,BST_UNCHECKED);
if(sector[CURSECT].floorstat&16)
   CheckDlgButton(hwndDlg,ID_FXFLIP,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,ID_FXFLIP,BST_UNCHECKED);
if(sector[CURSECT].floorstat&32)
   CheckDlgButton(hwndDlg,ID_FYFLIP,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,ID_FYFLIP,BST_UNCHECKED);
if(sector[CURSECT].floorstat&64)
   CheckDlgButton(hwndDlg,ID_FALIGN,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,ID_FALIGN,BST_UNCHECKED);

sprintf(szItemName,"%li",sector[CURSECT].floorz);
SetDlgItemText(hwndDlg,ID_FZ,szItemName);
SetDlgItemInt(hwndDlg,ID_FHEINUM,sector[CURSECT].floorheinum,TRUE);
SetDlgItemInt(hwndDlg,ID_FPIC,sector[CURSECT].floorpicnum,TRUE);
SetDlgItemInt(hwndDlg,ID_FSHADE,sector[CURSECT].floorshade,TRUE);
SetDlgItemInt(hwndDlg,ID_FPAL,sector[CURSECT].floorpal,TRUE);
SetDlgItemInt(hwndDlg,ID_FX,sector[CURSECT].floorxpanning,TRUE);
SetDlgItemInt(hwndDlg,ID_FY,sector[CURSECT].floorypanning,TRUE);

if(sector[CURSECT].ceilingstat&1)
   CheckDlgButton(hwndDlg,ID_CPARALAX,BST_CHECKED);
```

```
else
   CheckDlgButton(hwndDlg,ID_CPARALAX,BST_UNCHECKED);
if(sector[CURSECT].ceilingstat&2)
   CheckDlgButton(hwndDlg,ID_CSLOPE,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,ID_CSLOPE,BST_UNCHECKED);
if(sector[CURSECT].ceilingstat&4)
   CheckDlgButton(hwndDlg,ID_CSWAP,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,ID_CSWAP,BST_UNCHECKED);
if(sector[CURSECT].ceilingstat&8)
   CheckDlgButton(hwndDlg,ID_CSMOOTH,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,ID_CSMOOTH,BST_UNCHECKED);
if(sector[CURSECT].ceilingstat&16)
   CheckDlgButton(hwndDlg,ID_CXFLIP,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,ID_CXFLIP,BST_UNCHECKED);
if(sector[CURSECT].ceilingstat&32)
   CheckDlgButton(hwndDlg,ID_CYFLIP,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,ID_CYFLIP,BST_UNCHECKED);
if(sector[CURSECT].ceilingstat&64)
   CheckDlgButton(hwndDlg,ID_CALIGN,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,ID_CALIGN,BST_UNCHECKED);
sprintf(szItemName,"%li",sector[CURSECT].ceilingz);
SetDlgItemText(hwndDlg,ID_CZ,szItemName);
SetDlgItemInt(hwndDlg,ID_CHEINUM,sector[CURSECT].ceilingheinum,TRUE);
SetDlgItemInt(hwndDlg,ID_CPIC,sector[CURSECT].ceilingpicnum,TRUE);
SetDlgItemInt(hwndDlg,ID_CSHADE,sector[CURSECT].ceilingshade,TRUE);
SetDlgItemInt(hwndDlg,ID_CPAL,sector[CURSECT].ceilingpal,TRUE);
SetDlgItemInt(hwndDlg,ID_CX,sector[CURSECT].ceilingxpanning,TRUE);
SetDlgItemInt(hwndDlg,ID_CY,sector[CURSECT].ceilingypanning,TRUE);
SetDlgItemInt(hwndDlg,ID_WNUM,CURWALL,TRUE);
SetDlgItemInt(hwndDlg,ID_WLOTAG,wall[CURWALL].lotag,TRUE);
SetDlgItemInt(hwndDlg,ID_WHITAG,wall[CURWALL].hitag,TRUE);
SetDlgItemInt(hwndDlg,ID_WEXTRA,wall[CURWALL].extra,TRUE);

if(wall[CURWALL].cstat&1)
   CheckDlgButton(hwndDlg,ID_WBLOCKED,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,ID_WBLOCKED,BST_UNCHECKED);
if(wall[CURWALL].cstat&2)
   CheckDlgButton(hwndDlg,ID_WSWAPPED,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,ID_WSWAPPED,BST_UNCHECKED);
if(wall[CURWALL].cstat&4)
   CheckDlgButton(hwndDlg,ID_WALIGNED,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,ID_WALIGNED,BST_UNCHECKED);
if(wall[CURWALL].cstat&8)
   CheckDlgButton(hwndDlg,ID_WXFLIP,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,ID_WXFLIP,BST_UNCHECKED);
if(wall[CURWALL].cstat&16)
   CheckDlgButton(hwndDlg,ID_WMASKED,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,ID_WMASKED,BST_UNCHECKED);
if(wall[CURWALL].cstat&32)
   CheckDlgButton(hwndDlg,ID_WONEWAY,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,ID_WONEWAY,BST_UNCHECKED);
if(wall[CURWALL].cstat&64)
   CheckDlgButton(hwndDlg,ID_WHITSCAN,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,ID_WHITSCAN,BST_UNCHECKED);
if(wall[CURWALL].cstat&128)
   CheckDlgButton(hwndDlg,ID_WTRANS,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,ID_WTRANS,BST_UNCHECKED);
if(wall[CURWALL].cstat&256)
   CheckDlgButton(hwndDlg,ID_WYFLIP,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,ID_WYFLIP,BST_UNCHECKED);
if(wall[CURWALL].cstat&512)
```

```
   CheckDlgButton(hwndDlg,ID_WREV,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,ID_WREV,BST_UNCHECKED);
sprintf(szItemName,"%li",wall[CURWALL].x);
SetDlgItemText(hwndDlg,ID_WX,szItemName);
sprintf(szItemName,"%li",wall[CURWALL].y);
SetDlgItemText(hwndDlg,ID_WY,szItemName);
SetDlgItemInt(hwndDlg,ID_WPOINT2,wall[CURWALL].point2,TRUE);
SetDlgItemInt(hwndDlg,ID_WNEXTW,wall[CURWALL].nextwall,TRUE);
SetDlgItemInt(hwndDlg,ID_WNEXTS,wall[CURWALL].nextsector,TRUE);
SetDlgItemInt(hwndDlg,ID_WPIC,wall[CURWALL].picnum,TRUE);
SetDlgItemInt(hwndDlg,ID_WOVERPIC,wall[CURWALL].overpicnum,TRUE);
SetDlgItemInt(hwndDlg,ID_WSHADE,wall[CURWALL].shade,TRUE);
SetDlgItemInt(hwndDlg,ID_WPAL,wall[CURWALL].pal,TRUE);
SetDlgItemInt(hwndDlg,ID_WXREP,wall[CURWALL].xrepeat,TRUE);
SetDlgItemInt(hwndDlg,ID_WYREP,wall[CURWALL].yrepeat,TRUE);
SetDlgItemInt(hwndDlg,ID_WXPAN,wall[CURWALL].xpanning,TRUE);
SetDlgItemInt(hwndDlg,ID_WYPAN,wall[CURWALL].ypanning,TRUE);
//DlgDirList(hwndDlg,"*.map",IDC_LISTBOX1,0,DDL_ARCHIVE);
}



void ReplaceProc(HWND hwndDlg)
{
int original_tile;
int new_tile;
int a;
int count[5]={0,0,0,0,0};
char repstring[80];
original_tile=GetDlgItemInt(hwndDlg,IDC_R_ORIGINAL,NULL,TRUE);
new_tile=GetDlgItemInt(hwndDlg,IDC_R_NEW,NULL,TRUE);
// HERE
if(IsDlgButtonChecked(hwndDlg,IDC_R_CEILINGS)==BST_CHECKED)
   for(a=0;a<numsectors;a++)
      if(sector[a].ceilingpicnum==original_tile)
      {count[0]++;sector[a].ceilingpicnum=new_tile;}
if(IsDlgButtonChecked(hwndDlg,IDC_R_FLOORS)==BST_CHECKED)
   for(a=0;a<numsectors;a++)
      if(sector[a].floorpicnum==original_tile)
      {count[1]++;sector[a].floorpicnum=new_tile;}
if(IsDlgButtonChecked(hwndDlg,IDC_R_WALLS)==BST_CHECKED)
   for(a=0;a<numwalls;a++)
   {
      if(wall[a].picnum==original_tile)
      {count[2]++;wall[a].picnum=new_tile;}
      if(wall[a].overpicnum==original_tile)
      {count[3]++;wall[a].overpicnum=new_tile;}
   }
if(IsDlgButtonChecked(hwndDlg,IDC_R_SPRITES)==BST_CHECKED)
   for(a=0;a<numsprites;a++)
      if(sprite[a].picnum==original_tile)
      {count[4]++;sprite[a].picnum=new_tile;}

sprintf(repstring,
"Converted \
%i ceilings, \
%i floors, \
%i walls, \
%i overwals,\nand \
%i sprites from image \
%i to image \
%i.",count[0],count[1],count[2],count[3],count[4],original_tile,new_tile);
MessageBox(NULL, repstring, "Completed", MB_OK);

}
BOOL CALLBACK ReplaceItemProc(HWND hwndDlg, UINT message, WPARAM wParam, LPARAM lParam)
/*
BOOL CALLBACK ReplaceItemProc(hwndDlg, message, wParam, lParam)
HWND hwndDlg;
UINT message;
WPARAM wParam;
*/
{
```

```
static DWORD aIds[] = {
IDC_R_WALLS,IDH_R_WALL,
IDC_R_FLOORS,IDH_R_FLOOR,
IDC_R_CEILINGS,IDH_R_CEILING,
IDC_R_SPRITES,IDH_R_SPRITE,
IDC_R_REPLACE,IDH_R_REPLACE,
IDC_R_NEW,IDH_R_NEW,
IDC_R_ORIGINAL,IDH_R_ORIGINAL,
IDC_R_CLOSE,IDH_R_CLOSE,
 0,0
 };









   switch (message)
   {
// HERE
    case WM_HELP:
        WinHelp(((LPHELPINFO) lParam)->hItemHandle, "openglbt.hlp",
            HELP_WM_HELP, (DWORD) (LPSTR) aIds);

        break;

    case WM_CONTEXTMENU:
        WinHelp((HWND) wParam, "openglbt.hlp", HELP_CONTEXTMENU,
            (DWORD) (LPVOID) aIds);
        break;

      case WM_INITDIALOG:
SetDlgItemText(hwndDlg,IDC_R_ORIGINAL,"0");
SetDlgItemText(hwndDlg,IDC_R_NEW,"0");
//       SetupSpriteProc(hwndDlg);
         break;
    case WM_CLOSE:
               EndDialog(hwndDlg, wParam);
               return FALSE;
      case WM_COMMAND:
         switch (LOWORD(wParam))
         {
            case IDOK:
               EndDialog(hwndDlg, wParam);
               return FALSE;
            case IDCANCEL:
               EndDialog(hwndDlg, wParam);
               return FALSE;
            case IDC_R_REPLACE:
               ReplaceProc(hwndDlg);
               return FALSE;
            case IDC_R_CLOSE:
               EndDialog(hwndDlg, wParam);
               return FALSE;
         }
         break;
   }
   return FALSE;
}









void ProcessStubProc(HWND hwndDlg)
{
int original_tile;
int new_tile;
int a;
int count[5]={0,0,0,0,0};
```

```
char repstring[80];
original_tile=GetDlgItemInt(hwndDlg,IDC_R_ORIGINAL,NULL,TRUE);
new_tile=GetDlgItemInt(hwndDlg,IDC_R_NEW,NULL,TRUE);
// HERE
if(IsDlgButtonChecked(hwndDlg,IDC_R_CEILINGS)==BST_CHECKED)
   for(a=0;a<numsectors;a++)
      if(sector[a].ceilingpicnum==original_tile)
      {count[0]++;sector[a].ceilingpicnum=new_tile;}
if(IsDlgButtonChecked(hwndDlg,IDC_R_FLOORS)==BST_CHECKED)
   for(a=0;a<numsectors;a++)
      if(sector[a].floorpicnum==original_tile)
      {count[1]++;sector[a].floorpicnum=new_tile;}
if(IsDlgButtonChecked(hwndDlg,IDC_R_WALLS)==BST_CHECKED)
   for(a=0;a<numwalls;a++)
   {
      if(wall[a].picnum==original_tile)
      {count[2]++;wall[a].picnum=new_tile;}
      if(wall[a].overpicnum==original_tile)
      {count[3]++;wall[a].overpicnum=new_tile;}
   }
if(IsDlgButtonChecked(hwndDlg,IDC_R_SPRITES)==BST_CHECKED)
   for(a=0;a<numsprites;a++)
      if(sprite[a].picnum==original_tile)
      {count[4]++;sprite[a].picnum=new_tile;}

sprintf(repstring,
"Converted \
%i ceilings, \
%i floors, \
%i walls, \
%i overwals,\nand \
%i sprites from image \
%i to image \
%i.",count[0],count[1],count[2],count[3],count[4],original_tile,new_tile);
MessageBox(NULL, repstring, "Completed", MB_OK);

}
BOOL CALLBACK StubProc(HWND hwndDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
HRSRC stubdata;
LPVOID stubhandle;
HGLOBAL stubblock;
HRSRC stubdata2;
LPVOID stubhandle2;
HGLOBAL stubblock2;
FILE *stubfile;

char a,b;
char inv;
char invr;
char invl;
char holo;
char jp;
char nv;
char med;
char ster;
char kick;
long weapon[10];
long weapon2[10];
char nweap;
char Weapon_[10][10];
char ep1[17];
char ep2[17];
char ep3[17];
char stub_name[13];
char stub_patch[13];

   switch (message)
   {
/*
      case WM_INITDIALOG:
//SetDlgItemText(hwndDlg,IDC_R_ORIGINAL,"0");
//SetDlgItemText(hwndDlg,IDC_R_NEW,"0");
SetDlgItemInt(hwndDlg,STUB_W1,3,TRUE);
SetDlgItemInt(hwndDlg,STUB_W2,2,TRUE);
SetDlgItemInt(hwndDlg,STUB_W3,1,TRUE);
```

```
SetDlgItemInt(hwndDlg,STUB_W4,4,TRUE);
SetDlgItemInt(hwndDlg,STUB_W5,5,TRUE);
SetDlgItemInt(hwndDlg,STUB_W6,6,TRUE);
SetDlgItemInt(hwndDlg,STUB_W7,7,TRUE);
SetDlgItemInt(hwndDlg,STUB_W8,8,TRUE);
SetDlgItemInt(hwndDlg,STUB_W9,9,TRUE);
SetDlgItemInt(hwndDlg,STUB_W10,10,TRUE);
        break;
//*/
    case WM_CLOSE:
                EndDialog(hwndDlg, wParam);
                return FALSE;
      case WM_COMMAND:
         switch (LOWORD(wParam))
         {
            case IDOK:
nweap=0;
if(IsDlgButtonChecked(hwndDlg,STUB_1)!=BST_CHECKED)
{weapon[nweap]=GetDlgItemInt(hwndDlg,STUB_W1,NULL,TRUE);nweap++;}
if(IsDlgButtonChecked(hwndDlg,STUB_2)!=BST_CHECKED)
{weapon[nweap]=GetDlgItemInt(hwndDlg,STUB_W2,NULL,TRUE);nweap++;}
if(IsDlgButtonChecked(hwndDlg,STUB_3)!=BST_CHECKED)
{weapon[nweap]=GetDlgItemInt(hwndDlg,STUB_W3,NULL,TRUE);nweap++;}
if(IsDlgButtonChecked(hwndDlg,STUB_4)!=BST_CHECKED)
{weapon[nweap]=GetDlgItemInt(hwndDlg,STUB_W4,NULL,TRUE);nweap++;}
if(IsDlgButtonChecked(hwndDlg,STUB_5)!=BST_CHECKED)
{weapon[nweap]=GetDlgItemInt(hwndDlg,STUB_W5,NULL,TRUE);nweap++;}
if(IsDlgButtonChecked(hwndDlg,STUB_6)!=BST_CHECKED)
{weapon[nweap]=GetDlgItemInt(hwndDlg,STUB_W6,NULL,TRUE);nweap++;}
if(IsDlgButtonChecked(hwndDlg,STUB_7)!=BST_CHECKED)
{weapon[nweap]=GetDlgItemInt(hwndDlg,STUB_W7,NULL,TRUE);nweap++;}
if(IsDlgButtonChecked(hwndDlg,STUB_8)!=BST_CHECKED)
{weapon[nweap]=GetDlgItemInt(hwndDlg,STUB_W8,NULL,TRUE);nweap++;}
if(IsDlgButtonChecked(hwndDlg,STUB_9)!=BST_CHECKED)
{weapon[nweap]=GetDlgItemInt(hwndDlg,STUB_W9,NULL,TRUE);nweap++;}
if(IsDlgButtonChecked(hwndDlg,STUB_10)!=BST_CHECKED)
{weapon[nweap]=GetDlgItemInt(hwndDlg,STUB_W10,NULL,TRUE);nweap++;}
for(a=0;a<nweap;a++)
 for(b=0;b<nweap;b++)
  if(weapon[a]==weapon[b]&&a!=b)
  {
   MessageBox(NULL, "Weapons may nut be allocated the same number as other weapons.", "Error", MB_O
K);
   return 1;
  }
for(a=0;a<nweap;a++)
 if(weapon[a]<1||weapon[a]>10)
 {
  MessageBox(NULL, "Weapon numbers must be between 1 and 10 inclusive.", "Error", MB_OK);
  return 1;
 }

  weapon[0]=STUB_1;
  weapon[1]=STUB_2;
  weapon[2]=STUB_3;
  weapon[3]=STUB_4;
  weapon[4]=STUB_5;
  weapon[5]=STUB_6;
  weapon[6]=STUB_7;
  weapon[7]=STUB_8;
  weapon[8]=STUB_9;
  weapon[9]=STUB_10;
  weapon2[0]=STUB_W1;
  weapon2[1]=STUB_W2;
  weapon2[2]=STUB_W3;
  weapon2[3]=STUB_W4;
  weapon2[4]=STUB_W5;
  weapon2[5]=STUB_W6;
  weapon2[6]=STUB_W7;
  weapon2[7]=STUB_W8;
  weapon2[8]=STUB_W9;
  weapon2[9]=STUB_W10;

for(a=0;a<10;a++)
if(IsDlgButtonChecked(hwndDlg,weapon[a])!=BST_CHECKED)
{   if(GetDlgItemInt(hwndDlg,weapon2[a],NULL,TRUE)<10)
```

```
{
      sprintf(Weapon_[a],"Weapon_%i",GetDlgItemInt(hwndDlg,weapon2[a],NULL,TRUE));
      Weapon_[a][8]=0;
}
 else sprintf(Weapon_[a],"Weapon_%i",GetDlgItemInt(hwndDlg,weapon2[a],NULL,TRUE));
}else sprintf(Weapon_[a],"?eapon_??");

if(IsDlgButtonChecked(hwndDlg,STUB_INV)==BST_CHECKED)inv='?';else inv='I';
if(IsDlgButtonChecked(hwndDlg,STUB_INVL)==BST_CHECKED)invl='?';else invl='I';
if(IsDlgButtonChecked(hwndDlg,STUB_INVR)==BST_CHECKED)invr='?';else invr='I';
if(IsDlgButtonChecked(hwndDlg,STUB_HOLO)==BST_CHECKED)holo='?';else holo='H';
if(IsDlgButtonChecked(hwndDlg,STUB_JP)==BST_CHECKED)jp='?';else jp='J';
if(IsDlgButtonChecked(hwndDlg,STUB_NV)==BST_CHECKED)nv='?';else nv='N';
if(IsDlgButtonChecked(hwndDlg,STUB_MED)==BST_CHECKED)med='?';else med='M';
if(IsDlgButtonChecked(hwndDlg,STUB_STER)==BST_CHECKED)ster='?';else ster='S';
if(IsDlgButtonChecked(hwndDlg,STUB_KICK)==BST_CHECKED)kick='?';else kick='Q';

GetDlgItemText(hwndDlg,STUB_LAMEL,ep1,17);
GetDlgItemText(hwndDlg,STUB_LUNAR,ep2,17);
GetDlgItemText(hwndDlg,STUB_SHRAP,ep3,17);
ep1[16]=0;
b=0;
for(a=0;a<16;a++)
{
 if(ep1[a]==0)b=1;
 if(b==1)
  ep1[a]=32;
}
ep2[16]=0;
b=0;
for(a=0;a<16;a++)
{
 if(ep2[a]==0)b=1;
 if(b==1)
  ep2[a]=32;
}
ep3[16]=0;
b=0;
for(a=0;a<16;a++)
{
 if(ep3[a]==0)b=1;
 if(b==1)
  ep3[a]=32;
}

GetDlgItemText(hwndDlg,STUB_NAME,stub_name,13);
for(a=0;a<13;a++)
{
 if(
 ((stub_name[a]<'A'||stub_name[a]>'Z')&&(stub_name[a]<'0'||stub_name[a]>'9'))
 ||a>=8)
 {
  stub_name[a]='.';
  stub_name[a+1]='C';
  stub_name[a+2]='O';
  stub_name[a+3]='M';
  a+=4;
  while(a<13)
  {stub_name[a]=0;a++;}
 }
}
if(stub_name[0]==0||stub_name[0]=='.')
{
 MessageBox(NULL, "The patch file needs a name.", "Error", MB_OK);
 return 1;
}

GetDlgItemText(hwndDlg,STUB_PATCH,stub_patch,13);
for(a=0;a<13;a++)
{
 if(((stub_patch[a]<'A'||stub_patch[a]>'Z')&&(stub_patch[a]<'0'||stub_patch[a]>'9'))||a>=8)
 {
  stub_patch[a]='.';
  stub_patch[a+1]='E';
  stub_patch[a+2]='X';
  stub_patch[a+3]='E';
```

```
  a+=4;
  while(a<13)
  {stub_patch[a]=0;a++;}
 }
}
if(stub_patch[0]==0||stub_patch[0]=='.')
{
 MessageBox(NULL, "The patched executionable file needs a name.", "Error", MB_OK);
 return 1;
}



stubdata=FindResource(NULL,"#1",RT_RCDATA);
if(stubdata==NULL)
 {MessageBox(NULL, "stubdata", "Error", MB_OK);return 1;}
stubblock=LoadResource(NULL,stubdata);
if(stubblock==NULL)
 {MessageBox(NULL, "stubblock", "Error", MB_OK);return 1;}
stubhandle=LockResource(stubblock);
if(stubhandle==NULL)
 {MessageBox(NULL, "stubhandle", "Error", MB_OK);return 1;}

stubdata2=FindResource(NULL,"#2",RT_RCDATA);
if(stubdata2==NULL)
 {MessageBox(NULL, "stubdata2", "Error", MB_OK);return 1;}
stubblock2=LoadResource(NULL,stubdata2);
if(stubblock2==NULL)
 {MessageBox(NULL, "stubblock2", "Error", MB_OK);return 1;}
stubhandle2=LockResource(stubblock2);
if(stubhandle2==NULL)
 {MessageBox(NULL, "stubhandle2", "Error", MB_OK);return 1;}

if((stubfile=fopen(stub_name,"wb"))==NULL)
 {MessageBox(NULL, "Could not open patch file for output.", "Error", MB_OK);return 1;}
//fwrite(stubhandle,11016,1,stubfile);
fwrite(stubhandle,11224,1,stubfile);
fputc(inv,stubfile);
fputc(invr,stubfile);
fputc(invl,stubfile);
fputc(holo,stubfile);
fputc(jp,stubfile);
fputc(nv,stubfile);
fputc(med,stubfile);
fputc(ster,stubfile);
fputc(kick,stubfile);
fputc(0,stubfile);fputc(123,stubfile);fputc(45,stubfile);
for(a=0;a<17;a++)
fputc(ep1[a],stubfile);
for(a=0;a<17;a++)
fputc(ep2[a],stubfile);
for(a=0;a<17;a++)
fputc(ep3[a],stubfile);

for(a=0;a<10;a++)
{
fputs(Weapon_[a],stubfile);fputc(0,stubfile);
if(GetDlgItemInt(hwndDlg,weapon2[a],NULL,TRUE)<10)
fputc(0,stubfile);
}
for(a=0;a<13;a++)
fputc(stub_patch[a],stubfile);
fwrite(stubhandle2,2218,1,stubfile);
fclose(stubfile);

MessageBox(NULL, "Patch file created.", "Success", MB_OK);
               EndDialog(hwndDlg, wParam);
               return FALSE;
            case IDCANCEL:
               EndDialog(hwndDlg, wParam);
               return FALSE;
         }
         break;
   }
   return FALSE;
}
```

```
char filename[256];
char smallfilename[256];
char count=0;
long int fileloc[1024];
char filenames[1024][13];
long EntrySize[1024],EntryOffset[1024],remaining;
char EntryName[1024][12];
long IdxCnt;
char DN3GrpSig[12];
char grouppath[256];
long int nSelItems;
long int nSelItemsInBuffer;
long int top_index=0;
char outputstring[256];
//#define variable 2097152
#define variable 7340032
// char bigblock[524288];
//char bigblock[1048576];
char bigblock[variable];
char GROUPFILEISOPEN=0;
char method=0;

UINT APIENTRY OldStyle(HWND hdlg,UINT uiMsg,WPARAM wParam,LPARAM lParam)
{
return false;
}

BOOL CALLBACK GroupProc(HWND hwndDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
   char addressstring[25],tempstr[256],tempstr2[256],tempfilename[256];
   long int i,i2,i3;
   HWND hwndList;
   HWND hwndList2;
   char temp_name[MAXFILE];
   char temp_ext[MAXEXT];
   long temp_size=0;
   long temp_datasize=0;
   char smalltemp_name[MAXFILE];
   char smalltemp_ext[MAXEXT];
   char smalltemp[256];
   long smallfirstoffset;
   long smallsecondoffset;

   FILE *dukef,*outf,*tempf,*smalltempf;
   fpos_t filepos;

   DWORD cchCurDir;
   LPTSTR lpszCurDir;
   LPTSTR lpszFileToDelete;
   OPENFILENAME ofn;
   ZeroMemory(&ofn, sizeof(OPENFILENAME));
   ofn.lStructSize = sizeof(OPENFILENAME);
   ofn.hwndOwner=hwndDlg;
   ofn.lpstrFileTitle = NULL;
   ofn.nMaxFileTitle = 0;
   ofn.lpstrInitialDir = NULL;
   ofn.lpfnHook=OldStyle;
   if(message!=WM_INITDIALOG)
   {
      if(count>2)
      {
         if(SendDlgItemMessage(hwndDlg,IDC_GROUP,LB_GETTOPINDEX,0,0)
         !=SendDlgItemMessage(hwndDlg,IDC_GROUPSIZE,LB_GETTOPINDEX,0,0))
```

```
         {
            SendDlgItemMessage(hwndDlg,IDC_GROUPSIZE,LB_SETTOPINDEX,
            (WPARAM)SendDlgItemMessage(hwndDlg,IDC_GROUP,LB_GETTOPINDEX,0,0),0);
            count=0;
         }
      }
      else
         count++;
   }
   switch (message)
   {
      case WM_INITDIALOG:
         GROUPFILEISOPEN=0;
         hwndList = GetDlgItem(hwndDlg, IDC_GROUP);
         hwndList2 = GetDlgItem(hwndDlg, IDC_GROUPSIZE);
         SendMessage(hwndList,LB_RESETCONTENT,0,0);
         SendMessage(hwndList2,LB_RESETCONTENT,0,0);
         method=0;
         SetDlgItemText(hwndDlg,IDC_METHODTEXT,"Please select a method to add/delete files.");
         break;
      case WM_CLOSE:
         EndDialog(hwndDlg, wParam);
         return FALSE;
      case WM_COMMAND:
         switch (LOWORD(wParam))
         {
            case IDC_METHOD1:
               SetDlgItemText(hwndDlg,IDC_METHODTEXT,"Method 1 - faster, less permenant space neede
d.  Volotile if your platform crashes.");
               method=1;
            break;
            case IDC_METHOD2:
               SetDlgItemText(hwndDlg,IDC_METHODTEXT,"Method 2 - slower, more permenant space neede
d.  Safer in the event of your platform crashing.");
               method=2;
            break;
            case IDC_ADDFILE:
               if(method==0){MessageBox(NULL,"You must select a method to process the file with.",
"Error", MB_OK);return 1;}
               if(!GROUPFILEISOPEN){MessageBox(NULL,"You must either open or create a new group fil
e first.", "Error", MB_OK);return 1;}
               ofn.nMaxFile = sizeof(smallfilename);
               ofn.lpstrFilter =
"Animation files\0*.ANM\0\
Art files\0*.ART\0\
Configuration files\0*.CFG\0\
Control files\0*.CON\0\
Data files\0*.BIN;*.DAT;*.TMB\0\
Demonstration\0*.DMO\0\
Documentation files\0*.DOC;*.HLP;*.TXT;*.RTF\0\
Executionable files\0*.BAT;*.COM;*.EXE\0\
Group files\0*.GRP\0\
Initialisation files\0*.INI\0\
Map files\0*.MAP\0\
MIDI files\0*.MID\0\
RTS files\0*.RTS\0\
Save game files\0*.SAV\0\
Temporary files\0*.$$$\0\
Voice files\0*.VOC\0\
Wave files\0*.WAV\0\
et al.\0*.$$$;*.ANM;*.ART;*.BAT;*.BIN;*.CFG;*.COM;*.CON;*.DAT;*.DMO;*.DOC;*.EXE;*.GRP;*.HLP;*.INI;*
.MAP;*.MID;*.RTF;*.RTS;*.SAV;*.TMB;*.TXT;*.VOC;*.WAV\0\
All\0*.*\0\0";
               ofn.nFilterIndex = 19;
               ofn.lpstrDefExt= NULL;
               hwndList = GetDlgItem(hwndDlg, IDC_GROUP);
               hwndList2 = GetDlgItem(hwndDlg, IDC_GROUPSIZE);
               ofn.lpstrFile = smallfilename;
               ofn.Flags = OFN_ENABLEHOOK|OFN_NOLONGNAMES|OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST;
               if (GetOpenFileName(&ofn)!=TRUE)return 1;
               fnsplit(smallfilename,NULL,NULL,smalltemp_name,smalltemp_ext);
               sprintf(smalltemp,"%s%s",smalltemp_name,smalltemp_ext);
               i=SendMessage(hwndList,LB_FINDSTRINGEXACT,0,(LPARAM)smalltemp);
               if(LB_ERR!=i)
               {MessageBox(NULL,"File already exists.", "Error", MB_OK);return false;}
               if(method==1)
```

```
{
   if((dukef=fopen(filename,"r+b"))==NULL)
   {MessageBox(NULL,"Can not open the file.", "Error", MB_OK);return false;}
   if((smalltempf=fopen(smallfilename,"rb"))==NULL)
   {MessageBox(NULL,"Can not open the file.", "Error", MB_OK);return false;}
   fseek(dukef,0,2);
   temp_size=ftell(dukef);
   fseek(dukef,0,0);
   fseek(smalltempf,0,2);
   i=ftell(smalltempf);
   fseek(smalltempf,0,0);
   smallsecondoffset=temp_size-IdxCnt*16-16;
   smallfirstoffset=IdxCnt*16+16;
   while(smallsecondoffset>variable)
   {
      fseek(dukef,smallfirstoffset,0);
      fseek(dukef,smallsecondoffset-variable,1);
      fread(&bigblock,variable,1,dukef);
      fseek(dukef,16,1);
      fwrite(&bigblock,variable,1,dukef);
      sprintf(tempstr,"Writing data  -  %li",smallsecondoffset);
      SetDlgItemText(hwndDlg,IDC_GROUPTEXT,tempstr);
      smallsecondoffset-=variable;
   }
   if(smallsecondoffset>0)
   {
      fseek(dukef,smallfirstoffset,0);
      fread(&bigblock,smallsecondoffset,1,dukef);
      fseek(dukef,16,1);
      fwrite(&bigblock,smallsecondoffset,1,dukef);
      sprintf(tempstr,"Writing data  -  0");
      SetDlgItemText(hwndDlg,IDC_GROUPTEXT,tempstr);
   }
   filenames[IdxCnt][12]=0;
   EntrySize[IdxCnt]=i;
   sprintf(filenames[IdxCnt],smalltemp);
   sprintf(tempstr,"%li",EntrySize[IdxCnt]);
   SendMessage(hwndList,LB_ADDSTRING,IdxCnt,(LPARAM)filenames[IdxCnt]);
   SendMessage(hwndList,LB_SETITEMDATA,IdxCnt,(LPARAM)IdxCnt);
   SendMessage(hwndList2,LB_ADDSTRING,IdxCnt,(LPARAM)tempstr);
   SendMessage(hwndList2,LB_SETITEMDATA,IdxCnt,(LPARAM)IdxCnt);
   fseek(dukef,0,2);
   while(i>variable)
   {
      fread(&bigblock,variable,1,smalltempf);
      fwrite(&bigblock,variable,1,dukef);
      SetDlgItemText(hwndDlg,IDC_GROUPTEXT,tempstr);
      sprintf(tempstr,"Writing new data  -  %li",i);
      SetDlgItemText(hwndDlg,IDC_GROUPTEXT,tempstr);
      i-=variable;
   }
   if(i>0)
   {
      fread(&bigblock,i,1,smalltempf);
      fwrite(&bigblock,i,1,dukef);
      SetDlgItemText(hwndDlg,IDC_GROUPTEXT,"Writing new data  -  0");
   }
   sprintf(tempstr,"Updating header",i);
   SetDlgItemText(hwndDlg,IDC_GROUPTEXT,tempstr);
   fseek(dukef,IdxCnt*16+16,0);
   sprintf(smalltemp,"%s%s",smalltemp_name,smalltemp_ext);
   fwrite(smalltemp,12,1,dukef);
   fwrite(&EntrySize[IdxCnt],4,1,dukef);
   fseek(dukef,12,0);
   IdxCnt++;
   fwrite(&IdxCnt,4,1,dukef);
   fclose(dukef);
   fclose(smalltempf);
   SetDlgItemText(hwndDlg,IDC_GROUPTEXT,"Idle");
}
else
{
   if((dukef=fopen(filename,"rb"))==NULL)
   {MessageBox(NULL,"Can not open the file.", "Error", MB_OK);return false;}
   if((smalltempf=fopen(smallfilename,"rb"))==NULL)
   {MessageBox(NULL,"Can not open the file.", "Error", MB_OK);return false;}
```

```
                tmpnam(tempfilename);
                if((tempf=fopen(tempfilename,"wb"))==NULL)
                {MessageBox(NULL,"Can not open the file.", "Error", MB_OK);return false;}
                fseek(dukef,0,2);
                temp_size=ftell(dukef);
                fseek(dukef,0,0);
                fseek(smalltempf,0,2);
                i=ftell(smalltempf);
                fseek(smalltempf,0,0);
                smallfirstoffset=IdxCnt*16+16;
                while(smallfirstoffset>variable)
                {
                   fread(&bigblock,variable,1,dukef);
                   fwrite(&bigblock,variable,1,tempf);
                   smallfirstoffset-=variable;
                   SetDlgItemText(hwndDlg,IDC_GROUPTEXT,"Writing header");
                }
                if(smallfirstoffset>0)
                {
                   fread(&bigblock,smallfirstoffset,1,dukef);
                   fwrite(&bigblock,smallfirstoffset,1,tempf);
                }
                sprintf(smalltemp,"%s%s",smalltemp_name,smalltemp_ext);
                fwrite(smalltemp,12,1,tempf);
                fwrite(&i,4,1,tempf);
                smallsecondoffset=temp_size-IdxCnt*16-16;
                while(smallsecondoffset>variable)
                {
                   fread(&bigblock,variable,1,dukef);
                   fwrite(&bigblock,variable,1,tempf);
                   sprintf(tempstr,"Writing data  -  %li/%li",smallsecondoffset,temp_size);
                   SetDlgItemText(hwndDlg,IDC_GROUPTEXT,tempstr);
                   smallsecondoffset-=variable;
                }
                if(smallsecondoffset>0)
                {
                   fread(&bigblock,smallsecondoffset,1,dukef);
                   fwrite(&bigblock,smallsecondoffset,1,tempf);
                   sprintf(tempstr,"Writing data  -  0/%li",temp_size);
                   SetDlgItemText(hwndDlg,IDC_GROUPTEXT,tempstr);
                }
                filenames[IdxCnt][12]=0;
                EntrySize[IdxCnt]=i;
                sprintf(filenames[IdxCnt],smalltemp);
                sprintf(tempstr,"%li",EntrySize[IdxCnt]);
                SendMessage(hwndList,LB_ADDSTRING,IdxCnt,(LPARAM)filenames[IdxCnt]);
                SendMessage(hwndList,LB_SETITEMDATA,IdxCnt,(LPARAM)IdxCnt);
                SendMessage(hwndList2,LB_ADDSTRING,IdxCnt,(LPARAM)tempstr);
                SendMessage(hwndList2,LB_SETITEMDATA,IdxCnt,(LPARAM)IdxCnt);
                while(i>variable)
                {
                   fread(&bigblock,variable,1,smalltempf);
                   fwrite(&bigblock,variable,1,tempf);
                   sprintf(tempstr,"Writing new data  -  %li",i);
                   SetDlgItemText(hwndDlg,IDC_GROUPTEXT,tempstr);
                   i-=variable;
                }
                if(i>0)
                {
                   fread(&bigblock,i,1,smalltempf);
                   fwrite(&bigblock,i,1,tempf);
                   SetDlgItemText(hwndDlg,IDC_GROUPTEXT,"Writing new data  -  0");
                }
                fseek(tempf,12,0);
                IdxCnt++;
                fwrite(&IdxCnt,4,1,tempf);
                fclose(dukef);
                fclose(tempf);
                fclose(smalltempf);
                SetDlgItemText(hwndDlg,IDC_GROUPTEXT,"Replacing");
                if(remove(filename)!=0)
                {
                   sprintf(tempstr,"Could not remove %s!\nTemporary file saved as %s.",filename,t
empfilename);
                   MessageBox(NULL,tempstr, "Error", MB_OK);
                   SetDlgItemText(hwndDlg,IDC_GROUPTEXT,"Idle");
```

```
                    return false;
                }
                if(rename(tempfilename,filename)!=0)
                {
                    sprintf(tempstr,"Could not rename %s to %s!\nTemporary file saved as %s.",temp
filename,filename,tempfilename);
                    MessageBox(NULL,tempstr, "Error", MB_OK);
                    SetDlgItemText(hwndDlg,IDC_GROUPTEXT,"Idle");
                    return false;
                }
                SetDlgItemText(hwndDlg,IDC_GROUPTEXT,"Idle");
             }
          break;
          case IDC_DELETEFILE:
             hwndList = GetDlgItem(hwndDlg, IDC_GROUP);
             hwndList2 = GetDlgItem(hwndDlg, IDC_GROUPSIZE);
   if(!GROUPFILEISOPEN){MessageBox(NULL,"You must either open or create a new group file first.", "
Error", MB_OK);return 1;}
   nSelItems=SendDlgItemMessage(hwndDlg,IDC_GROUP,LB_GETSELITEMS,(WPARAM)1024,(LPARAM)(LPINT)filelo
c);
   if((dukef=fopen(filename,"rb"))==NULL)
   {MessageBox(NULL,"Can not open the file.", "Error", MB_OK);return false;}
   tmpnam(tempfilename);
   if((tempf=fopen(tempfilename,"wb"))==NULL)
   {MessageBox(NULL,"Can not open the file.", "Error", MB_OK);return false;}
   SetDlgItemText(hwndDlg,IDC_GROUPTEXT,"Writing header info");
   fwrite("KenSilverman",12,1,tempf);
   i=IdxCnt-nSelItems;
   fwrite(&i,4,1,tempf);
   smallfirstoffset=0;
   fseek(dukef,16,0);
   for(i=0;i<nSelItems;i++)
   {
      sprintf(tempstr,"Writing header block %li",i+1);
      SetDlgItemText(hwndDlg,IDC_GROUPTEXT,tempstr);
      fread(bigblock,fileloc[i]*16-smallfirstoffset,1,dukef);
      fwrite(bigblock,fileloc[i]*16-smallfirstoffset,1,tempf);
      smallfirstoffset=fileloc[i]*16+16;
      fseek(dukef,16,1);
   }
   SetDlgItemText(hwndDlg,IDC_GROUPTEXT,"Writing last header block");
   fread(bigblock,IdxCnt*16-smallfirstoffset,1,dukef);
   fwrite(bigblock,IdxCnt*16-smallfirstoffset,1,tempf);
   fseek(dukef,16,1);
   smallfirstoffset=IdxCnt*16+16;
   for(i=0;i<nSelItems;i++)
   {
      sprintf(tempstr,"Writing data block %li",i+1);
      SetDlgItemText(hwndDlg,IDC_GROUPTEXT,tempstr);
      fseek(dukef,smallfirstoffset,0);
      smallsecondoffset=EntryOffset[fileloc[i]];
      while(smallsecondoffset-smallfirstoffset>variable)
      {
         fread(bigblock,variable,1,dukef);
         fwrite(bigblock,variable,1,tempf);
      }
      if(smallsecondoffset-smallfirstoffset>0)
      {
         fread(bigblock,smallsecondoffset-smallfirstoffset,1,dukef);
         fwrite(bigblock,smallsecondoffset-smallfirstoffset,1,tempf);
      }
      smallfirstoffset=smallsecondoffset+EntrySize[fileloc[i]];
   }
   fseek(dukef,smallfirstoffset,0);
   smallsecondoffset=EntryOffset[IdxCnt];
   SetDlgItemText(hwndDlg,IDC_GROUPTEXT,"Writing last data block");
   while(smallsecondoffset-smallfirstoffset>variable)
   {
      fread(bigblock,variable,1,dukef);
      fwrite(bigblock,variable,1,tempf);
   }
   if(smallsecondoffset-smallfirstoffset>0)
   {
      fread(bigblock,smallsecondoffset-smallfirstoffset,1,dukef);
      fwrite(bigblock,smallsecondoffset-smallfirstoffset,1,tempf);
   }
```

```
   fclose(tempf);
   fclose(dukef);
                  SetDlgItemText(hwndDlg,IDC_GROUPTEXT,"Replacing");
                  if(remove(filename)!=0)
                  {
                     sprintf(tempstr,"Could not remove %s!\nTemporary file saved as %s.",filename,t
empfilename);
                     MessageBox(NULL,tempstr, "Error", MB_OK);
                     SetDlgItemText(hwndDlg,IDC_GROUPTEXT,"Idle");
                     return false;
                  }
                  if(rename(tempfilename,filename)!=0)
                  {
                     sprintf(tempstr,"Could not rename %s to %s!\nTemporary file saved as %s.",temp
filename,filename,tempfilename);
                     MessageBox(NULL,tempstr, "Error", MB_OK);
                     SetDlgItemText(hwndDlg,IDC_GROUPTEXT,"Idle");
                     return false;
                  }
/*
      vector<int> vsize(EntrySize, EntrySize+1024);
      vector<int> voffset(EntryOffset, EntryOffset+1024);
      vector<int> vname(EntryName, EntryName+1024);
      vector<int> vfilenames(filenames, filenames+1024);

      remove_if(vsize.begin()+i,vsize.begin()+i+1,all_true<int>());
      remove_if(voffsett.begin()+i,voffsett.begin()+i+1,all_true<int>());
      remove_if(vname.begin()+i,vname.begin()+i+1,all_true<int>());
      remove_if(vfilename.begin()+i,vfilename.begin()+i+1,all_true<int>());
*/

                  SetDlgItemText(hwndDlg,IDC_GROUPTEXT,"Reopening");
                  SendMessage(hwndList,LB_RESETCONTENT,0,0);
                  SendMessage(hwndList2,LB_RESETCONTENT,0,0);
                  GROUPFILEISOPEN=0;
                  SetDlgItemText(hwndDlg,IDC_GROUPNAME,"File name:\nFile size: 0\nNumber of entries: 0
\nTotal data size: 0");
                  if((dukef=fopen(filename,"rb"))==NULL)
                  {MessageBox(NULL,"Can not open the file.", "Error", MB_OK);
                  SetDlgItemText(hwndDlg,IDC_GROUPTEXT,"Idle");return false;}
                  fread(DN3GrpSig,12,1,dukef);
                  fread(&IdxCnt,4,1,dukef);
                  EntryOffset[0]=(IdxCnt+1)*16;//<<4;
                  for(i=0;i<IdxCnt;i++)
                  {
                     fread(EntryName[i],12,1,dukef);
                     fread(&EntrySize[i],4,1,dukef);
                     sprintf(filenames[i],"%s",EntryName[i]);
                     filenames[i][12]=0;
                     SendMessage(hwndList,LB_ADDSTRING,i,(LPARAM)filenames[i]);
                     SendMessage(hwndList,LB_SETITEMDATA,i,(LPARAM)i);
                     sprintf(tempstr,"%li",EntrySize[i]);
                     SendMessage(hwndList2,LB_ADDSTRING,i,(LPARAM)tempstr);
                     SendMessage(hwndList2,LB_SETITEMDATA,i,(LPARAM)i);
                     temp_datasize+=EntrySize[i];
                     if(i<1023)
                     EntryOffset[i+1]=EntrySize[i]+EntryOffset[i];
                  }
                  fseek(dukef,0,2);
                  temp_size=ftell(dukef);
                  fclose(dukef);
                  GetCurrentDirectory(256,grouppath);
                  SetDlgItemText(hwndDlg,IDC_GROUPPATH,grouppath);
                  fnsplit(filename,NULL,NULL,temp_name,temp_ext);
                  sprintf(outputstring,"File name: %s%s\nFile size: %li\nNumber of entries: %li\nTotal
 data size: %li",temp_name,temp_ext,temp_size,IdxCnt,temp_datasize);
                  SetDlgItemText(hwndDlg,IDC_GROUPNAME,outputstring);
                  GROUPFILEISOPEN=1;
                  SetDlgItemText(hwndDlg,IDC_GROUPTEXT,"Idle");
               break;
               case IDC_OPEN:
                  ofn.nMaxFile = sizeof(filename);
                  ofn.lpstrFilter = "Group files\0*.GRP\0Temporary files\0*.$$$\0All\0*.*\0";
                  ofn.nFilterIndex = 1;
                  ofn.lpstrDefExt= "GRP\0";
                  ofn.lpstrFile = filename;
```

```
            ofn.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST |OFN_NOREADONLYRETURN|OFN_HIDEREAD
ONLY;
            hwndList = GetDlgItem(hwndDlg, IDC_GROUP);
            hwndList2 = GetDlgItem(hwndDlg, IDC_GROUPSIZE);
            SendMessage(hwndList,LB_RESETCONTENT,0,0);
            SendMessage(hwndList2,LB_RESETCONTENT,0,0);
            GROUPFILEISOPEN=0;
            SetDlgItemText(hwndDlg,IDC_GROUPNAME,"File name:\nFile size: 0\nNumber of entries: 0
\nTotal data size: 0");
            if (GetOpenFileName(&ofn)!=TRUE)return 1;
            dukef=fopen(filename,"rb");
            if(dukef==NULL)
            {MessageBox(NULL,"Can not open the file.", "Error", MB_OK);return false;}
            if(fread(DN3GrpSig,12,1,dukef)!=1)
            {MessageBox(NULL,"Error while reading from file.", "Error", MB_OK);return false;}
            else
            if(DN3GrpSig[0]!='K'||DN3GrpSig[1]!='e'||DN3GrpSig[2]!='n'||DN3GrpSig[3]!='S'||DN3Gr
pSig[4]!='i'||DN3GrpSig[5]!='l'||DN3GrpSig[6]!='v'||DN3GrpSig[7]!='e'||DN3GrpSig[8]!='r'||DN3GrpSig
[9]!='m'||DN3GrpSig[10]!='a'||DN3GrpSig[11]!='n')
            {MessageBox(NULL,"File seems to be of a different format.", "Error", MB_OK);return f
alse;}
            fread(&IdxCnt,4,1,dukef);
            EntryOffset[0]=(IdxCnt+1)*16;//<<4;
            for(i=0;i<IdxCnt;i++)
            {
               fread(EntryName[i],12,1,dukef);
               fread(&EntrySize[i],4,1,dukef);
               sprintf(filenames[i],"%s",EntryName[i]);
               filenames[i][12]=0;
               SendMessage(hwndList,LB_ADDSTRING,i,(LPARAM)filenames[i]);
               SendMessage(hwndList,LB_SETITEMDATA,i,(LPARAM)i);
               sprintf(tempstr,"%li",EntrySize[i]);
               SendMessage(hwndList2,LB_ADDSTRING,i,(LPARAM)tempstr);
               SendMessage(hwndList2,LB_SETITEMDATA,i,(LPARAM)i);
               temp_datasize+=EntrySize[i];
               if(i<1023)
               EntryOffset[i+1]=EntrySize[i]+EntryOffset[i];
            }
            fseek(dukef,0,2);
            temp_size=ftell(dukef);
            fclose(dukef);
            GetCurrentDirectory(256,grouppath);
            SetDlgItemText(hwndDlg,IDC_GROUPPATH,grouppath);
            fnsplit(filename,NULL,NULL,temp_name,temp_ext);
            sprintf(outputstring,
            "File name: %s%s\nFile size: %li\nNumber of entries: %li\nTotal data size: %li",temp
_name,temp_ext,temp_size,IdxCnt,temp_datasize);
            SetDlgItemText(hwndDlg,IDC_GROUPNAME,outputstring);
            GROUPFILEISOPEN=1;
         break;
         case IDC_NEW:
            ofn.nMaxFile = sizeof(filename);
            ofn.lpstrFilter = "Group files\0*.GRP\0Temporary files\0*.$$$\0All\0*.*\0";
            ofn.nFilterIndex = 1;
            ofn.lpstrDefExt= "GRP\0";
            ofn.lpstrFile = filename;
            ofn.Flags = OFN_PATHMUSTEXIST|OFN_NOREADONLYRETURN|OFN_HIDEREADONLY;
            hwndList = GetDlgItem(hwndDlg, IDC_GROUP);
            hwndList2 = GetDlgItem(hwndDlg, IDC_GROUPSIZE);
            SendMessage(hwndList,LB_RESETCONTENT,0,0);
            SendMessage(hwndList2,LB_RESETCONTENT,0,0);
            GROUPFILEISOPEN=0;
            SetDlgItemText(hwndDlg,IDC_GROUPNAME,"File name:\nFile size: 0\nNumber of entries: 0
\nTotal data size: 0");
            if (GetSaveFileName(&ofn)!=TRUE)return 1;
            dukef=fopen(filename,"wb");
            if(dukef==NULL)
            {MessageBox(NULL,"Can not open the file.", "Error", MB_OK);return false;}
            IdxCnt=0;
            fwrite("KenSilverman",12,1,dukef);
            fwrite(&IdxCnt,4,1,dukef);
            fclose(dukef);
            GetCurrentDirectory(256,grouppath);
            SetDlgItemText(hwndDlg,IDC_GROUPPATH,grouppath);
            fnsplit(filename,NULL,NULL,temp_name,temp_ext);
            sprintf(outputstring,"File name: %s%s\nFile size: 12\nNumber of entries: 0\nTotal da
```

```
ta size: 0",temp_name,temp_ext);
                SetDlgItemText(hwndDlg,IDC_GROUPNAME,outputstring);
                GROUPFILEISOPEN=1;
            break;
            case IDOK:
                EndDialog(hwndDlg, wParam);
                return FALSE;
            case IDC_GROUPEXTRACT:
                if(!GROUPFILEISOPEN){MessageBox(NULL,"You must either open or create a new group fil
e first.", "Error", MB_OK);return 1;}
                {
                    nSelItems=SendDlgItemMessage(hwndDlg,IDC_GROUP,LB_GETSELITEMS,(WPARAM)1024,(LPARA
M)(LPINT)fileloc);
                    GetDlgItemText(hwndDlg,IDC_GROUPPATH,grouppath,256);
                    i=strlen(grouppath);
                    if(i>0)
                    {
                        if(grouppath[i-1]!='\\')
                        {
                            grouppath[i+1]=grouppath[i];
                            grouppath[i]='\\';
                        }
                    }
                    SetCurrentDirectory(grouppath);
                    if(nSelItems==0)
                    {
                        if(MessageBox(NULL,"Do you want to extract all of the files?", "No files have
been selected", MB_YESNO)==IDYES)
                        {
                            SendDlgItemMessage(hwndDlg,IDC_GROUP,LB_SELITEMRANGEEX,0,IdxCnt-1);
                            nSelItems=SendDlgItemMessage(hwndDlg,IDC_GROUP,LB_GETSELITEMS,(WPARAM)1024,
(LPARAM)(LPINT)fileloc);
                        }
                        else return false;
                    }
                    if((dukef=fopen(filename,"rb"))==NULL)
                    {
                        sprintf(tempstr,"Could not open the input file.");
                        MessageBox(NULL,tempstr, "Error", MB_OK);
                        return false;
                    }
                else
                for(i=0;i<nSelItems;i++)
                {
                    sprintf(tempstr,"%s%s",grouppath,filenames[fileloc[i]]);
                    if((outf=fopen(tempstr,"wb"))==NULL)
                    {
                        sprintf(tempstr,"Could not open the file %s%s for output.",grouppath,filenames
[fileloc[i]]);
                        MessageBox(NULL,tempstr, "Error", MB_OK);
                    }
                    else
                    {
                        remaining=EntrySize[fileloc[i]];
                        fsetpos(dukef,&EntryOffset[fileloc[i]]);
                        sprintf(tempstr,"%s  -  %li/%li",filenames[fileloc[i]],remaining,EntrySize[fil
eloc[i]]);
                        SetDlgItemText(hwndDlg,IDC_GROUPTEXT,tempstr);
                        while(remaining>variable)
                        {
                            fread(&bigblock,variable,1,dukef);
                            if(fwrite(&bigblock,variable,1,outf)!=1)
                            {
                                sprintf(tempstr,"Not enough space in target directory/drive.\nStopped at
 %s.",filenames[fileloc[i]]);
                                MessageBox(NULL,tempstr, "Insufficiant Space", MB_OK);
                                return false;
                            }
                            remaining-=variable;
                            sprintf(tempstr,"%s  -  %li/%li",filenames[fileloc[i]],remaining,EntrySize[
fileloc[i]]);
                            SetDlgItemText(hwndDlg,IDC_GROUPTEXT,tempstr);
                        }
                        if(remaining>0&&remaining<variable)
                        {
                            if(fread(&bigblock,remaining,1,dukef)!=1)
```

```
                        {
                           MessageBox(NULL,"Error while reading from input file.", "Error", MB_OK);
                           return false;
                        }
                        if(fwrite(&bigblock,remaining,1,outf)!=1)
                        {
                           sprintf(tempstr,"Not enough space in target directory/drive.\nStopped at
 %s.",filenames[fileloc[i]]);
                           MessageBox(NULL,tempstr, "Insufficiant Space", MB_OK);
                           return false;
                        }
                        sprintf(tempstr,"%s\n 0/%i",filenames[fileloc[i]],EntrySize[fileloc[i]]);
                        SetDlgItemText(hwndDlg,IDC_GROUPTEXT,tempstr);
                     }
                  }
                  fclose(outf);
               }
               SetDlgItemText(hwndDlg,IDC_GROUPTEXT,"Idle");
               fclose(dukef);
            }
            return FALSE;
         }
         break;
   }
   return FALSE;
}
```

```
void SaveSectorProc(HWND hwndDlg)
{
char *endptr;
//CURSECT=GetDlgItemInt(hwndDlg,ID_SECTORNUM,NULL, TRUE);
sector[CURSECT].wallptr=GetDlgItemInt(hwndDlg,ID_FIRSTWALL,NULL,TRUE);
sector[CURSECT].wallnum=GetDlgItemInt(hwndDlg,ID_NUMWALL,NULL,TRUE);
sector[CURSECT].visibility=GetDlgItemInt(hwndDlg,ID_VIS,NULL,TRUE);
sector[CURSECT].lotag=GetDlgItemInt(hwndDlg,ID_LOTAG,NULL,TRUE);
sector[CURSECT].hitag=GetDlgItemInt(hwndDlg,ID_HITAG,NULL,TRUE);
sector[CURSECT].extra=GetDlgItemInt(hwndDlg,ID_EXTRA,NULL,TRUE);

sector[CURSECT].floorheinum=GetDlgItemInt(hwndDlg,ID_FHEINUM,NULL,TRUE);
sector[CURSECT].floorpicnum=GetDlgItemInt(hwndDlg,ID_FPIC,NULL,TRUE);
sector[CURSECT].floorshade=GetDlgItemInt(hwndDlg,ID_FSHADE,NULL,TRUE);
sector[CURSECT].floorpal=GetDlgItemInt(hwndDlg,ID_FPAL,NULL,TRUE);
sector[CURSECT].floorxpanning=GetDlgItemInt(hwndDlg,ID_FX,NULL,TRUE);
sector[CURSECT].floorypanning=GetDlgItemInt(hwndDlg,ID_FY,NULL,TRUE);

sector[CURSECT].ceilingheinum=GetDlgItemInt(hwndDlg,ID_CHEINUM,NULL,TRUE);
sector[CURSECT].ceilingpicnum=GetDlgItemInt(hwndDlg,ID_CPIC,NULL,TRUE);
sector[CURSECT].ceilingshade=GetDlgItemInt(hwndDlg,ID_CSHADE,NULL,TRUE);
sector[CURSECT].ceilingpal=GetDlgItemInt(hwndDlg,ID_CPAL,NULL,TRUE);
sector[CURSECT].ceilingxpanning=GetDlgItemInt(hwndDlg,ID_CX,NULL,TRUE);
sector[CURSECT].ceilingypanning=GetDlgItemInt(hwndDlg,ID_CY,NULL,TRUE);


GetDlgItemText(hwndDlg,ID_FZ,szItemName,260);
sector[CURSECT].floorz = strtol(szItemName, &endptr, 10);

GetDlgItemText(hwndDlg,ID_CZ,szItemName,260);
sector[CURSECT].ceilingz = strtol(szItemName, &endptr, 10);


GetDlgItemText(hwndDlg,IDP_X,szItemName,260);
posx = strtol(szItemName, &endptr, 10);
GetDlgItemText(hwndDlg,IDP_Y,szItemName,260);
posy = strtol(szItemName, &endptr, 10);
GetDlgItemText(hwndDlg,IDP_Z,szItemName,260);
posz = strtol(szItemName, &endptr, 10);
ang = GetDlgItemInt(hwndDlg,IDP_ANG,NULL,TRUE);
cursectnum = GetDlgItemInt(hwndDlg,IDP_SECTOR,NULL,TRUE);
```

```
sector[CURSECT].floorstat=0;

if(IsDlgButtonChecked(hwndDlg,ID_FPARALAX)==BST_CHECKED)
   sector[CURSECT].floorstat+=1;
if(IsDlgButtonChecked(hwndDlg,ID_FSLOPE)==BST_CHECKED)
   sector[CURSECT].floorstat+=2;
if(IsDlgButtonChecked(hwndDlg,ID_FSWAP)==BST_CHECKED)
   sector[CURSECT].floorstat+=4;
if(IsDlgButtonChecked(hwndDlg,ID_FSMOOTH)==BST_CHECKED)
   sector[CURSECT].floorstat+=8;
if(IsDlgButtonChecked(hwndDlg,ID_FXFLIP)==BST_CHECKED)
   sector[CURSECT].floorstat+=16;
if(IsDlgButtonChecked(hwndDlg,ID_FYFLIP)==BST_CHECKED)
   sector[CURSECT].floorstat+=32;
if(IsDlgButtonChecked(hwndDlg,ID_FALIGN)==BST_CHECKED)
   sector[CURSECT].floorstat+=64;


sector[CURSECT].ceilingstat=0;
if(IsDlgButtonChecked(hwndDlg,ID_CPARALAX)==BST_CHECKED)
   sector[CURSECT].ceilingstat+=1;
if(IsDlgButtonChecked(hwndDlg,ID_CSLOPE)==BST_CHECKED)
   sector[CURSECT].ceilingstat+=2;
if(IsDlgButtonChecked(hwndDlg,ID_CSWAP)==BST_CHECKED)
   sector[CURSECT].ceilingstat+=4;
if(IsDlgButtonChecked(hwndDlg,ID_CSMOOTH)==BST_CHECKED)
   sector[CURSECT].ceilingstat+=8;
if(IsDlgButtonChecked(hwndDlg,ID_CXFLIP)==BST_CHECKED)
   sector[CURSECT].ceilingstat+=16;
if(IsDlgButtonChecked(hwndDlg,ID_CYFLIP)==BST_CHECKED)
   sector[CURSECT].ceilingstat+=32;
if(IsDlgButtonChecked(hwndDlg,ID_CALIGN)==BST_CHECKED)
   sector[CURSECT].ceilingstat+=64;

}



void SaveWallProc(HWND hwndDlg)
{
char *endptr;
//CURWALL=GetDlgItemInt(hwndDlg,ID_WNUM,NULL, TRUE);

wall[CURWALL].lotag=GetDlgItemInt(hwndDlg,ID_WLOTAG,NULL,TRUE);
wall[CURWALL].hitag=GetDlgItemInt(hwndDlg,ID_WHITAG,NULL,TRUE);
wall[CURWALL].extra=GetDlgItemInt(hwndDlg,ID_WEXTRA,NULL,TRUE);
wall[CURWALL].point2=GetDlgItemInt(hwndDlg,ID_WPOINT2,NULL,TRUE);
wall[CURWALL].nextwall=GetDlgItemInt(hwndDlg,ID_WNEXTW,NULL,TRUE);
wall[CURWALL].nextsector=GetDlgItemInt(hwndDlg,ID_WNEXTS,NULL,TRUE);
wall[CURWALL].picnum=GetDlgItemInt(hwndDlg,ID_WPIC,NULL,TRUE);
wall[CURWALL].overpicnum=GetDlgItemInt(hwndDlg,ID_WOVERPIC,NULL,TRUE);
wall[CURWALL].shade=GetDlgItemInt(hwndDlg,ID_WSHADE,NULL,TRUE);
wall[CURWALL].pal=GetDlgItemInt(hwndDlg,ID_WPAL,NULL,TRUE);
wall[CURWALL].xrepeat=GetDlgItemInt(hwndDlg,ID_WXREP,NULL,TRUE);
wall[CURWALL].yrepeat=GetDlgItemInt(hwndDlg,ID_WYREP,NULL,TRUE);
wall[CURWALL].xpanning=GetDlgItemInt(hwndDlg,ID_WXPAN,NULL,TRUE);
wall[CURWALL].ypanning=GetDlgItemInt(hwndDlg,ID_WYPAN,NULL,TRUE);

GetDlgItemText(hwndDlg,ID_WX,szItemName,260);
wall[CURWALL].x = strtol(szItemName, &endptr, 10);
GetDlgItemText(hwndDlg,ID_WY,szItemName,260);
wall[CURWALL].y = strtol(szItemName, &endptr, 10);

wall[CURWALL].cstat=0;
if(IsDlgButtonChecked(hwndDlg,ID_WBLOCKED)==BST_CHECKED)
   wall[CURWALL].cstat+=1;
if(IsDlgButtonChecked(hwndDlg,ID_WSWAPPED)==BST_CHECKED)
   wall[CURWALL].cstat+=2;
if(IsDlgButtonChecked(hwndDlg,ID_WALIGNED)==BST_CHECKED)
   wall[CURWALL].cstat+=4;
if(IsDlgButtonChecked(hwndDlg,ID_WXFLIP)==BST_CHECKED)
   wall[CURWALL].cstat+=8;
if(IsDlgButtonChecked(hwndDlg,ID_WMASKED)==BST_CHECKED)
   wall[CURWALL].cstat+=16;
if(IsDlgButtonChecked(hwndDlg,ID_WONEWAY)==BST_CHECKED)
```

```
   wall[CURWALL].cstat+=32;
if(IsDlgButtonChecked(hwndDlg,ID_WHITSCAN)==BST_CHECKED)
   wall[CURWALL].cstat+=64;
if(IsDlgButtonChecked(hwndDlg,ID_WTRANS)==BST_CHECKED)
   wall[CURWALL].cstat+=128;
if(IsDlgButtonChecked(hwndDlg,ID_WYFLIP)==BST_CHECKED)
   wall[CURWALL].cstat+=256;
if(IsDlgButtonChecked(hwndDlg,ID_WREV)==BST_CHECKED)
   wall[CURWALL].cstat+=512;

//DlgDirList(hwndDlg,"*.map",IDC_LISTBOX1,0,DDL_ARCHIVE|DDL_READWRITE|DDL_READONLY|DDL_EXCLUSIVE);
}









//#define IDC_CHECKBOX1 103

BOOL CALLBACK DeleteItemProc(HWND hwndDlg, UINT message, WPARAM wParam, LPARAM lParam)
/*
BOOL CALLBACK DeleteItemProc(hwndDlg, message, wParam, lParam)
HWND hwndDlg;
UINT message;
WPARAM wParam;
//LPARAM lParam;
*/
{
//HANDLE hf;              // file handle
register  long a;

OPENFILENAME ofn;       // common dialog box structure
ZeroMemory(&ofn, sizeof(OPENFILENAME));
ofn.lStructSize = sizeof(OPENFILENAME);
ofn.hwndOwner = hwndDlg;
ofn.lpstrFile = szFile;
ofn.nMaxFile = sizeof(szFile);
ofn.lpstrFilter = "Maps\0*.MAP\0All\0*.*\0";
ofn.nFilterIndex = 1;
ofn.lpstrFileTitle = NULL;
ofn.nMaxFileTitle = 0;
ofn.lpstrInitialDir = NULL;
ofn.lpstrDefExt= "MAP\0";

//#define ID_PARRALAXED 103
//#define ID_ITEMNAME 101
//#define ID_ITEMNAME2 102
//#define IDC_LISTBOX1 104


static DWORD aIds[] = {
 ID_FPARALAX,  IDH_O_PARALLAX,
 ID_FSLOPE,  IDH_O_SLOPED,
 ID_FSWAP,  IDH_O_SWAP,
 ID_FSMOOTH,  IDH_O_SMOOTH,
 ID_FXFLIP,  IDH_O_XFLIP,
 ID_FYFLIP,  IDH_O_YFLIP,
 ID_FALIGN,  IDH_O_ALIGN,
 ID_FZ,  IDH_O_Z,
 ID_FHEINUM,  IDH_O_SLOPE,
 ID_FPIC,  IDH_O_TEXTURE,
 ID_FSHADE,  IDH_O_SHADE,
 ID_FPAL,  IDH_O_PALETTE,
 ID_FX,  IDH_O_XOFF,
 ID_FY,  IDH_O_YOFF,
 ID_CPARALAX,  IDH_O_PARALLAX,
 ID_CSLOPE,  IDH_O_SLOPED,
 ID_CSWAP,  IDH_O_SWAP,
 ID_CSMOOTH,  IDH_O_SMOOTH,
 ID_CXFLIP,  IDH_O_XFLIP,
 ID_CYFLIP,  IDH_O_YFLIP,
```

```
ID_CALIGN,  IDH_O_ALIGN,
ID_CZ,  IDH_O_Z,
ID_CHEINUM,  IDH_O_SLOPE,
ID_CPIC,  IDH_O_TEXTURE,
ID_CSHADE,  IDH_O_SHADE,
ID_CPAL,  IDH_O_PALETTE,
ID_CX,  IDH_O_XOFF,
ID_CY,  IDH_O_YOFF,
ID_SECTORNUM,  IDH_O_CSECT,
IDCURSECT,  IDH_O_CSECTSEL,
ID_FIRSTWALL,  IDH_O_FWALL,
IDFIRSTWALL,  IDH_O_FWALLSEL,
ID_NUMWALL,  IDH_O_NWALLS,
ID_VIS,  IDH_O_VISIBILITY,
ID_LOTAG,  IDH_O_LOTAG,
ID_HITAG,  IDH_O_HITAG,
ID_EXTRA,  IDH_O_EXTRA,
IDLAST,  IDH_O_LAST,
IDNEXT,  IDH_O_NEXT,
IDFLAST,  IDH_O__LAST,
IDFNEXT,  IDH_O__NEXT,
IDNEWWALL,  IDH_O_NEWWALL,
IDNEWSECTOR,  IDH_O_NEWSECT,
IDNEWMAP,  IDH_O_NEWMAP,

ID_WX,  IDH_W_X,
ID_WY,  IDH_W_Y,
ID_WPIC,  IDH_W_TEXTURE,
ID_WOVERPIC,  IDH_W_MASK,
ID_WSHADE,  IDH_W_SHADE,
ID_WPAL,  IDH_W_PALETTE,
ID_WXREP,  IDH_W_XREP,
ID_WYREP,  IDH_W_YREP,
ID_WXPAN,  IDH_W_XPAN,
ID_WYPAN,  IDH_W_YPAN,
ID_WLOTAG,  IDH_W_LOTAG,
ID_WHITAG,  IDH_W_HITAG,
ID_WEXTRA,  IDH_W_EXTRA,
ID_WBLOCKED,  IDH_W_BLOCK,
ID_WSWAPPED,  IDH_W_SWAP,
ID_WALIGNED,  IDH_W_ALIGN,
ID_WMASKED,  IDH_W_MASKED,
ID_WHITSCAN,  IDH_W_HITSCAN,
ID_WONEWAY,  IDH_W_ONEWAY,
ID_WNUM,  IDH_W_CWALL,
IDCURRENTWALL,  IDH_W_CWALLSEL,
ID_WPOINT2,  IDH_W_AWALL,
IDADJACENTWALL,  IDH_W_AWALLSEL,
ID_WNEXTW,  IDH_W_JWALL,
IDJOININGWALL,  IDH_W_JWALLSEL,
ID_WNEXTS,  IDH_W_JSECT,
IDJOININGSECTOR,  IDH_W_JSECTSEL,
ID_WXFLIP,  IDH_W_XFLIP,
ID_WYFLIP,  IDH_W_YFLIP,
ID_WTRANS,  IDH_W_TRANSLUCENCE,
ID_WREV,  IDH_W_REVERSING,
IDWLAST,  IDH_W_LAST,
IDWNEXT,  IDH_W_NEXT,
IDWFLAST,  IDH_W__LAST,
IDWFNEXT,  IDH_W__NEXT,

IDOPEN,  IDH_M_OPEN,
IDSAVE,  IDH_M_SAVE,
IDSAVEAS,  IDH_M_SAVEAS,
IDP_X,  IDH_M_X,
IDP_Y,  IDH_M_Y,
IDP_Z,  IDH_M_Z,
IDP_ANG,  IDH_M_ANG,
IDP_SECTOR,  IDH_M_SECTOR,
IDO_SEENINEAPPLY,  IDH_M_SAPPLY,
IDO_SEENINEREMOVE,  IDH_M_SREMOVE,
IDO_CRACKAPPLY,  IDH_M_CAPPLY,
IDO_CRACKREMOVE,  IDH_M_CREMOVE,
IDO_REPLACE,  IDH_M_REPLACE,
IDC_SINFOBOX,  IDH_M_INFOBOX,
IDC_SINFO,  IDH_M_INFO,
```

```
 IDC_ADDRESS,  IDH_M_ADDRESS,
    0,0
};


   switch (message)
   {
      case WM_INITDIALOG:
//SetDlgItemText(hwndDlg,IDC_ABOUT,"OpenGL Build Touch v1.5\n\nWritten by James Ferry\n\nCopyright
©\n Monday, 8 February 1999");
         SetupSectorProc(hwndDlg);
//       GETDIRLISTING
         break;



    case WM_HELP:
        WinHelp(((LPHELPINFO) lParam)->hItemHandle, "openglbt.hlp",
            HELP_WM_HELP, (DWORD) (LPSTR) aIds);

        break;

    case WM_CONTEXTMENU:
        WinHelp((HWND) wParam, "openglbt.hlp", HELP_CONTEXTMENU,
            (DWORD) (LPVOID) aIds);
        break;









      case WM_COMMAND:
         switch (LOWORD(wParam))
         {
            case IDOK:
               SaveSectorProc(hwndDlg);SaveWallProc(hwndDlg);
               EndDialog(hwndDlg, wParam);
               return FALSE;
            case IDCURSECT:
               FrontSingle(BLUE_INDEX);
               SaveSectorProc(hwndDlg);SaveWallProc(hwndDlg);
               CURSECT=GetDlgItemInt(hwndDlg,ID_SECTORNUM,NULL, TRUE);
               if(CURSECT<0)CURSECT=numsectors-1;
               else if(CURSECT>=numsectors)CURSECT=0;
               SetupSectorProc(hwndDlg);
               FrontSingle(RED_INDEX);
               FrontWall(YELLOW_INDEX);
               SwapBuffers(hDC);
               break;
            case IDADJACENTWALL:
               SaveSectorProc(hwndDlg);SaveWallProc(hwndDlg);
               FrontWall(BLUE_INDEX);
               CURWALL=GetDlgItemInt(hwndDlg,ID_WPOINT2,NULL, TRUE);
               if(CURWALL<0)CURWALL=numwalls-1;
               else if(CURWALL>=numwalls)CURWALL=0;
               SetupSectorProc(hwndDlg);
               FrontSingle(RED_INDEX);
               FrontWall(YELLOW_INDEX);
               SwapBuffers(hDC);
               break;
            case IDJOININGWALL:
               SaveSectorProc(hwndDlg);SaveWallProc(hwndDlg);
               FrontWall(BLUE_INDEX);
               CURWALL=GetDlgItemInt(hwndDlg,ID_WNEXTW,NULL, TRUE);
               if(CURWALL<0)CURWALL=numwalls-1;
               else if(CURWALL>=numwalls)CURWALL=0;
               SetupSectorProc(hwndDlg);
               FrontSingle(RED_INDEX);
               FrontWall(YELLOW_INDEX);
               SwapBuffers(hDC);
               break;
            case IDJOININGSECTOR:
```

```
      SaveSectorProc(hwndDlg);SaveWallProc(hwndDlg);
      FrontSingle(BLUE_INDEX);
      CURSECT=GetDlgItemInt(hwndDlg,ID_WNEXTS,NULL, TRUE);
      if(CURSECT<0)CURSECT=numsectors-1;
      else if(CURSECT>=numsectors)CURSECT=0;
      SetupSectorProc(hwndDlg);
      FrontSingle(RED_INDEX);
      FrontWall(YELLOW_INDEX);
      SwapBuffers(hDC);
      break;
   case IDFIRSTWALL:
      SaveSectorProc(hwndDlg);SaveWallProc(hwndDlg);
      FrontWall(BLUE_INDEX);
      CURWALL=GetDlgItemInt(hwndDlg,ID_FIRSTWALL,NULL, TRUE);
      if(CURWALL<0)CURWALL=numwalls-1;
      else if(CURWALL>=numwalls)CURWALL=0;
      SetupSectorProc(hwndDlg);
      FrontSingle(RED_INDEX);
      FrontWall(YELLOW_INDEX);
      SwapBuffers(hDC);
      break;
   case IDCURRENTWALL:
      FrontWall(BLUE_INDEX);
      SaveSectorProc(hwndDlg);SaveWallProc(hwndDlg);
      CURWALL=GetDlgItemInt(hwndDlg,ID_WNUM,NULL, TRUE);
      if(CURWALL<0)CURWALL=numwalls-1;
      else if(CURWALL>=numwalls)CURWALL=0;
      SetupSectorProc(hwndDlg);
      FrontSingle(RED_INDEX);
      FrontWall(YELLOW_INDEX);
      SwapBuffers(hDC);
      break;
   case IDCANCEL:
      EndDialog(hwndDlg, wParam);
      return FALSE;
   case IDNEXT:
      SaveSectorProc(hwndDlg);SaveWallProc(hwndDlg);
      FrontSingle(BLUE_INDEX);
      CURSECT++;
      if(CURSECT<0)CURSECT=numsectors-1;
      else if(CURSECT>=numsectors)CURSECT=0;
      SetupSectorProc(hwndDlg);
      FrontSingle(RED_INDEX);
      FrontWall(YELLOW_INDEX);
      SwapBuffers(hDC);
      break;
   case IDLAST:
      SaveSectorProc(hwndDlg);SaveWallProc(hwndDlg);
      FrontSingle(BLUE_INDEX);
      CURSECT--;
      if(CURSECT<0)CURSECT=numsectors-1;
      else if(CURSECT>=numsectors)CURSECT=0;
      SetupSectorProc(hwndDlg);
      FrontSingle(RED_INDEX);
      FrontWall(YELLOW_INDEX);
      SwapBuffers(hDC);
      break;
   case IDFNEXT:
      SaveSectorProc(hwndDlg);SaveWallProc(hwndDlg);
      FrontSingle(BLUE_INDEX);
      CURSECT+=10;
      if(CURSECT<0)CURSECT=numsectors-1;
      else if(CURSECT>=numsectors)CURSECT=0;
      SetupSectorProc(hwndDlg);
      FrontSingle(RED_INDEX);
      FrontWall(YELLOW_INDEX);
      SwapBuffers(hDC);
      break;
   case IDFLAST:
      SaveSectorProc(hwndDlg);SaveWallProc(hwndDlg);
      FrontSingle(BLUE_INDEX);
      CURSECT-=10;
      if(CURSECT<0)CURSECT=numsectors-1;
      else if(CURSECT>=numsectors)CURSECT=0;
      SetupSectorProc(hwndDlg);
      FrontSingle(RED_INDEX);
```

```
               FrontWall(YELLOW_INDEX);
               SwapBuffers(hDC);
               break;
            case IDWNEXT:
               SaveSectorProc(hwndDlg);SaveWallProc(hwndDlg);
               FrontWall(BLUE_INDEX);
               CURWALL++;
               if(CURWALL<0)CURWALL=numwalls-1;
               else if(CURWALL>=numwalls)CURWALL=0;
               SetupSectorProc(hwndDlg);
               FrontSingle(RED_INDEX);
               FrontWall(YELLOW_INDEX);
               SwapBuffers(hDC);
               break;
            case IDWLAST:
               SaveSectorProc(hwndDlg);SaveWallProc(hwndDlg);
               FrontWall(BLUE_INDEX);
               CURWALL--;
               if(CURWALL<0)CURWALL=numwalls-1;
               else if(CURWALL>=numwalls)CURWALL=0;
               SetupSectorProc(hwndDlg);
               FrontSingle(RED_INDEX);
               FrontWall(YELLOW_INDEX);
               SwapBuffers(hDC);
               break;
            case IDWFNEXT:
               SaveSectorProc(hwndDlg);SaveWallProc(hwndDlg);
               FrontWall(BLUE_INDEX);
               CURWALL+=10;
               if(CURWALL<0)CURWALL=numwalls-1;
               else if(CURWALL>=numwalls)CURWALL=0;
               SetupSectorProc(hwndDlg);
               FrontSingle(RED_INDEX);
               FrontWall(YELLOW_INDEX);
               SwapBuffers(hDC);
               break;
            case IDWFLAST:
               SaveSectorProc(hwndDlg);SaveWallProc(hwndDlg);
               FrontWall(BLUE_INDEX);
               CURWALL-=10;
               if(CURWALL<0)CURWALL=numwalls-1;
               else if(CURWALL>=numwalls)CURWALL=0;
               SetupSectorProc(hwndDlg);
               FrontSingle(RED_INDEX);
               FrontWall(YELLOW_INDEX);
               SwapBuffers(hDC);
               break;
//          case IDREFRESH:
//             GETDIRLISTING
//             break;
            case IDNEWMAP:
if(MessageBox(NULL, "Are you sure that you wish to create a new map?",
   "New Map", MB_YESNO)==IDYES)
   {
               numsprites=0;
               CURSECT=numsectors=0;
               CURWALL=numwalls=0;
               sprintf(szFile,"untitled.map");
               SetupSectorProc(hwndDlg);
   }
               break;
            case IDNEWSECTOR:
//              int numsectors,numwalls,numsprites;
               CURSECT=numsectors;
               numsectors++;
               SetupSectorProc(hwndDlg);
               break;
            case IDNEWWALL:
//              int numsectors,numwalls,numsprites;
               CURWALL=numwalls;
               numwalls++;
               SetupSectorProc(hwndDlg);
               break;
               // HERE
            case IDO_REPLACE:
               DialogBox(hinst,
```

```
               MAKEINTRESOURCE(DLG_TEST),
               hwndDlg, (DLGPROC)ReplaceItemProc);
               SetupSectorProc(hwndDlg);
               //hinst hWnd
            break;
            case IDO_SEENINEAPPLY:
               for(a=0;a<numsprites;a++)
                  if(sprite[a].xrepeat<=4 && sprite[a].picnum==1247)
                     if(!(sprite[a].cstat&32768))sprite[a].cstat+=32768;
               MessageBox(NULL, "Non-visible C-9 sprites have been made invisible.", "Finished", MB
_OK);
            break;
            case IDO_SEENINEREMOVE:
               for(a=0;a<numsprites;a++)
                  if(sprite[a].xrepeat<=4 && sprite[a].picnum==1247)
                     if(sprite[a].cstat&32768)sprite[a].cstat-=32768;
               MessageBox(NULL, "Invisible C-9 sprites have been made visible again.", "Finished",
MB_OK);
            break;
            case IDO_CRACKAPPLY:
               if(MessageBox(NULL,
               "Do you wish to make all cracks invisible?\nOtherwise I will change only paletted (2
 or greater) sprites.\nPalettes of 2 will create multiplayer variants.", "Hide cracks", MB_YESNO|MB
_ICONQUESTION)==IDYES)
               {
                  for(a=0;a<numsprites;a++)
                     if(sprite[a].picnum>=546 && sprite[a].picnum<=549)
                        if(!(sprite[a].cstat&32768))
                        {
                           sprite[a].cstat+=32768;
                           if(sprite[a].pal==2)sprite[a].pal=1;
                        }
                  MessageBox(NULL, "Cracks are now invisible.", "Finished", MB_OK);
               }
               else
               {
                  for(a=0;a<numsprites;a++)
                     if(sprite[a].picnum>=546 && sprite[a].picnum<=549)
                        if(!(sprite[a].cstat&32768))
                           if(sprite[a].pal>1)
                           {
                              sprite[a].cstat+=32768;
                              if(sprite[a].pal==2)sprite[a].pal=1;
                           }
                  MessageBox(NULL, "Paletted cracks are now invisible.", "Finished", MB_OK);
               }
            break;
            case IDO_CRACKREMOVE:
               if(MessageBox(NULL,
               "Do you wish to make invisible multiplayer cracks to palette 2?\nOtherwise they will
 remain as the standard multiplayer palette.", "Hide cracks", MB_YESNO|MB_ICONQUESTION)==IDYES)
               {
                  for(a=0;a<numsprites;a++)
                     if(sprite[a].picnum>=546 && sprite[a].picnum<=549)
                        if(sprite[a].cstat&32768)
                        {
                           sprite[a].cstat-=32768;
                           if(sprite[a].pal==1)sprite[a].pal=2;
                        }
                  MessageBox(NULL, "Cracks are now visible and multiplayer cracks have palette 2.",
 "Finished", MB_OK);
               }
               else
               {
                  for(a=0;a<numsprites;a++)
                     if(sprite[a].picnum>=546 && sprite[a].picnum<=549)
                        if(sprite[a].cstat&32768)
                           sprite[a].cstat-=32768;
                  MessageBox(NULL, "Cracks are now visible again.", "Finished", MB_OK);
               }
            break;
            case IDLOAD:
ofn.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST |OFN_NOREADONLYRETURN|OFN_HIDEREADONLY;
if (GetOpenFileName(&ofn)==TRUE)
{
//    MessageBox(NULL, szFile,"Filename", MB_OK);
```

```
      if(load()==0)
      {
         Right();Top();Front();
         FrontSingle(RED_INDEX);
         FrontWall(YELLOW_INDEX);
         SwapBuffers(hDC);
         CURSECT=0;
         CURWALL=sector[CURSECT].wallptr;
         SetupSectorProc(hwndDlg);
      }
else MessageBox(WindowFromDC(hDC), "Cannot open map!","Error While Loading",MB_OK);
}
               break;
            case IDSAVE:
               SaveSectorProc(hwndDlg);SaveWallProc(hwndDlg);
//if(MessageBox(WindowFromDC(hDC), "Save?","Save Map",MB_YESNO)==IDYES)
               save();
break;
            case IDSAVEAS:
ofn.Flags = OFN_PATHMUSTEXIST|OFN_NOREADONLYRETURN|OFN_HIDEREADONLY;
if (GetSaveFileName(&ofn)==TRUE)
{
      SaveSectorProc(hwndDlg);SaveWallProc(hwndDlg);
      save();
}
else MessageBox(WindowFromDC(hDC), "Cannot save map!","Error While Saving",MB_OK);
break;
/*
               SaveSectorProc(hwndDlg);SaveWallProc(hwndDlg);
               GetDlgItemText(hwndDlg,IDSAVENAME,szItemName,260);
               sprintf(file,"%s.map",szItemName);
               sprintf(szItemName,"Save as %s?",file);
if(MessageBox(WindowFromDC(hDC), szItemName,"Save Map",MB_YESNO)==IDYES)
               save();
               GETDIRLISTING
               break;
*/
/*            case IDPLAY:
///              playCDTrack(NULL, 1);
               playCDTrack(1);
               EndDialog(hwndDlg, wParam);
               return TRUE;
               */
         }
   }
   return FALSE;
}




void SetupSpriteProc(HWND hwndDlg)
{
if(info_enable)
   SetDlgItemText(hwndDlg,IDC_ENABLE,"Disable NW III TC sprite information");
else
   SetDlgItemText(hwndDlg,IDC_ENABLE,"Enable NW III TC sprite information");
SetDlgItemInt(hwndDlg,IDS_SPRITE,CURSPRITE,TRUE);
SetDlgItemInt(hwndDlg,IDS_X,sprite[CURSPRITE].x,TRUE);
SetDlgItemInt(hwndDlg,IDS_Y,sprite[CURSPRITE].y,TRUE);
SetDlgItemInt(hwndDlg,IDS_Z,sprite[CURSPRITE].z,TRUE);
SetDlgItemInt(hwndDlg,IDS_STATNUM,sprite[CURSPRITE].statnum,TRUE);
SetDlgItemInt(hwndDlg,IDS_PAL,sprite[CURSPRITE].pal,TRUE);
SetDlgItemInt(hwndDlg,IDS_SHADE,sprite[CURSPRITE].shade,TRUE);
SetDlgItemInt(hwndDlg,IDS_ANG,sprite[CURSPRITE].ang,TRUE);
SetDlgItemInt(hwndDlg,IDS_OWNER,sprite[CURSPRITE].owner,TRUE);
SetDlgItemInt(hwndDlg,IDS_SECTNUM,sprite[CURSPRITE].sectnum,TRUE);
SetDlgItemInt(hwndDlg,IDS_PICNUM,sprite[CURSPRITE].picnum,TRUE);
SetDlgItemInt(hwndDlg,IDS_CLIPDIST,sprite[CURSPRITE].clipdist,TRUE);
SetDlgItemInt(hwndDlg,IDS_EXTRA,sprite[CURSPRITE].extra,TRUE);
SetDlgItemInt(hwndDlg,IDS_LOTAG,sprite[CURSPRITE].lotag,TRUE);
SetDlgItemInt(hwndDlg,IDS_HITAG,sprite[CURSPRITE].hitag,TRUE);
SetDlgItemInt(hwndDlg,IDS_XVEL,sprite[CURSPRITE].xvel,TRUE);
SetDlgItemInt(hwndDlg,IDS_YVEL,sprite[CURSPRITE].yvel,TRUE);
```

```
SetDlgItemInt(hwndDlg,IDS_ZVEL,sprite[CURSPRITE].zvel,TRUE);
SetDlgItemInt(hwndDlg,IDS_XREP,sprite[CURSPRITE].xrepeat,TRUE);
SetDlgItemInt(hwndDlg,IDS_YREP,sprite[CURSPRITE].yrepeat,TRUE);
SetDlgItemInt(hwndDlg,IDS_XOFF,sprite[CURSPRITE].xoffset,TRUE);
SetDlgItemInt(hwndDlg,IDS_YOFF,sprite[CURSPRITE].yoffset,TRUE);


if(sprite[CURSPRITE].cstat&1)
   CheckDlgButton(hwndDlg,IDS_BLOCK,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,IDS_BLOCK,BST_UNCHECKED);
if(sprite[CURSPRITE].cstat&2)
   CheckDlgButton(hwndDlg,IDS_TRANS,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,IDS_TRANS,BST_UNCHECKED);
if(sprite[CURSPRITE].cstat&4)
   CheckDlgButton(hwndDlg,IDS_XFLIP,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,IDS_XFLIP,BST_UNCHECKED);
if(sprite[CURSPRITE].cstat&8)
   CheckDlgButton(hwndDlg,IDS_YFLIP,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,IDS_YFLIP,BST_UNCHECKED);

if(sprite[CURSPRITE].cstat&16 && !(sprite[CURSPRITE].cstat&32))
   CheckDlgButton(hwndDlg,IDS_FACEWALLFLOOR,BST_CHECKED);
else if(!(sprite[CURSPRITE].cstat&16) && sprite[CURSPRITE].cstat&32)
   CheckDlgButton(hwndDlg,IDS_FACEWALLFLOOR,BST_INDETERMINATE);
else if(sprite[CURSPRITE].cstat&16 && sprite[CURSPRITE].cstat&32)
   CheckDlgButton(hwndDlg,IDS_FACEWALLFLOOR,BST_UNCHECKED);
else if(!(sprite[CURSPRITE].cstat&16) && !(sprite[CURSPRITE].cstat&32))
   CheckDlgButton(hwndDlg,IDS_FACEWALLFLOOR,BST_UNCHECKED);

if(sprite[CURSPRITE].cstat&64)
   CheckDlgButton(hwndDlg,IDS_ONESIDE,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,IDS_ONESIDE,BST_UNCHECKED);
if(sprite[CURSPRITE].cstat&128)
   CheckDlgButton(hwndDlg,IDS_CENTRE,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,IDS_CENTRE,BST_UNCHECKED);
if(sprite[CURSPRITE].cstat&256)
   CheckDlgButton(hwndDlg,IDS_HITSCAN,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,IDS_HITSCAN,BST_UNCHECKED);
if(sprite[CURSPRITE].cstat&512)
   CheckDlgButton(hwndDlg,IDS_TRANSREV,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,IDS_TRANSREV,BST_UNCHECKED);
/*
if(sprite[CURSPRITE].cstat&1024)
if(sprite[CURSPRITE].cstat&2048)
if(sprite[CURSPRITE].cstat&4096)
if(sprite[CURSPRITE].cstat&8192)
if(sprite[CURSPRITE].cstat&16384)
*/
if(sprite[CURSPRITE].cstat&32768)
   CheckDlgButton(hwndDlg,IDS_INVISIBLE,BST_CHECKED);
else
   CheckDlgButton(hwndDlg,IDS_INVISIBLE,BST_UNCHECKED);
}




void SaveSpriteProc(HWND hwndDlg)
{
char *endptr;
sprite[CURSPRITE].statnum=GetDlgItemInt(hwndDlg,IDS_STATNUM,NULL,TRUE);
sprite[CURSPRITE].pal=GetDlgItemInt(hwndDlg,IDS_PAL,NULL,TRUE);
sprite[CURSPRITE].shade=GetDlgItemInt(hwndDlg,IDS_SHADE,NULL,TRUE);
sprite[CURSPRITE].ang=GetDlgItemInt(hwndDlg,IDS_ANG,NULL,TRUE);
sprite[CURSPRITE].owner=GetDlgItemInt(hwndDlg,IDS_OWNER,NULL,TRUE);
```

```
sprite[CURSPRITE].sectnum=GetDlgItemInt(hwndDlg,IDS_SECTNUM,NULL,TRUE);
sprite[CURSPRITE].picnum=GetDlgItemInt(hwndDlg,IDS_PICNUM,NULL,TRUE);
sprite[CURSPRITE].clipdist=GetDlgItemInt(hwndDlg,IDS_CLIPDIST,NULL,TRUE);
sprite[CURSPRITE].extra=GetDlgItemInt(hwndDlg,IDS_EXTRA,NULL,TRUE);
sprite[CURSPRITE].lotag=GetDlgItemInt(hwndDlg,IDS_LOTAG,NULL,TRUE);
sprite[CURSPRITE].hitag=GetDlgItemInt(hwndDlg,IDS_HITAG,NULL,TRUE);
sprite[CURSPRITE].xvel=GetDlgItemInt(hwndDlg,IDS_XVEL,NULL,TRUE);
sprite[CURSPRITE].yvel=GetDlgItemInt(hwndDlg,IDS_YVEL,NULL,TRUE);
sprite[CURSPRITE].zvel=GetDlgItemInt(hwndDlg,IDS_ZVEL,NULL,TRUE);
sprite[CURSPRITE].xrepeat=GetDlgItemInt(hwndDlg,IDS_XREP,NULL,TRUE);
sprite[CURSPRITE].yrepeat=GetDlgItemInt(hwndDlg,IDS_YREP,NULL,TRUE);
sprite[CURSPRITE].xoffset=GetDlgItemInt(hwndDlg,IDS_XOFF,NULL,TRUE);
sprite[CURSPRITE].yoffset=GetDlgItemInt(hwndDlg,IDS_YOFF,NULL,TRUE);

GetDlgItemText(hwndDlg,IDS_X,szItemName,260);
sprite[CURSPRITE].x = strtol(szItemName, &endptr, 10);

GetDlgItemText(hwndDlg,IDS_Y,szItemName,260);
sprite[CURSPRITE].y = strtol(szItemName, &endptr, 10);

GetDlgItemText(hwndDlg,IDS_Z,szItemName,260);
sprite[CURSPRITE].z = strtol(szItemName, &endptr, 10);


sprite[CURSPRITE].cstat=0;

if(IsDlgButtonChecked(hwndDlg,IDS_BLOCK)==BST_CHECKED)
   sprite[CURSPRITE].cstat+=1;
if(IsDlgButtonChecked(hwndDlg,IDS_TRANS)==BST_CHECKED)
   sprite[CURSPRITE].cstat+=2;
if(IsDlgButtonChecked(hwndDlg,IDS_XFLIP)==BST_CHECKED)
   sprite[CURSPRITE].cstat+=4;
if(IsDlgButtonChecked(hwndDlg,IDS_YFLIP)==BST_CHECKED)
   sprite[CURSPRITE].cstat+=8;

if(IsDlgButtonChecked(hwndDlg,IDS_ONESIDE)==BST_CHECKED)
   sprite[CURSPRITE].cstat+=64;
if(IsDlgButtonChecked(hwndDlg,IDS_CENTRE)==BST_CHECKED)
   sprite[CURSPRITE].cstat+=128;
if(IsDlgButtonChecked(hwndDlg,IDS_HITSCAN)==BST_CHECKED)
   sprite[CURSPRITE].cstat+=256;
if(IsDlgButtonChecked(hwndDlg,IDS_TRANSREV)==BST_CHECKED)
   sprite[CURSPRITE].cstat+=512;
if(IsDlgButtonChecked(hwndDlg,IDS_INVISIBLE)==BST_CHECKED)
   sprite[CURSPRITE].cstat+=32768;

if(IsDlgButtonChecked(hwndDlg,IDS_FACEWALLFLOOR)==BST_CHECKED)
   sprite[CURSPRITE].cstat+=16;
else if(IsDlgButtonChecked(hwndDlg,IDS_FACEWALLFLOOR)==BST_INDETERMINATE)
   sprite[CURSPRITE].cstat+=32;

}


char path1[260];
char path2[260];
char path3[260];

//BOOL CALLBACK TestProc(hwndDlg, message, wParam, lParam)
//HWND hwndDlg;
//UINT message;
//WPARAM wParam;
BOOL CALLBACK TestProc(HWND hwndDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
char teststring[128];
char c;
OPENFILENAME ofn;
ZeroMemory(&ofn, sizeof(OPENFILENAME));
ofn.lStructSize = sizeof(OPENFILENAME);
ofn.hwndOwner = hwndDlg;
ofn.nFilterIndex = 1;
ofn.lpstrFileTitle = NULL;
ofn.nMaxFileTitle = 0;
ofn.lpstrInitialDir = NULL;
DWORD value;
if(message!=WM_INITDIALOG)
```

```
{
 if(IsDlgButtonChecked(hwndDlg,IDC_VERBOSE)!=BST_CHECKED)
 {
  CheckDlgButton(hwndDlg,IDC_VERBOSE,BST_CHECKED);
  SetDlgItemText(hwndDlg,IDC_STATUS,"Optimising without verbose checked is pretty much suicidal.");
 }
 if(IsDlgButtonChecked(hwndDlg,IDC_CHECKLOG)!=BST_UNCHECKED)
 {
  CheckDlgButton(hwndDlg,IDC_CHECKLOG,BST_UNCHECKED);
  SetDlgItemText(hwndDlg,IDC_STATUS,"Logging has not been enabled as yet.");
 }
}
   switch (message)
   {
      case WM_INITDIALOG:
         CheckDlgButton(hwndDlg,IDC_VERBOSE,BST_CHECKED);
         break;
      case WM_COMMAND:
         switch (LOWORD(wParam))
         {
            case IDC_BORIG:
               ofn.lpstrFile = path1;
               ofn.nMaxFile = sizeof(path1);
               ofn.lpstrFilter = "Con files\0*.CON\0All\0*.*\0";
               ofn.lpstrDefExt= "CON\0";
               ofn.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST;
               if (GetOpenFileName(&ofn)==TRUE)
                SetDlgItemText(hwndDlg,IDC_ORIG,path1);
               return FALSE;
            case IDC_BOPTIM:
               ofn.lpstrFile = path2;
               ofn.nMaxFile = sizeof(path2);
               ofn.lpstrFilter = "Con files\0*.CON\0All\0*.*\0";
               ofn.lpstrDefExt= "CON\0";
               ofn.Flags = OFN_PATHMUSTEXIST | OFN_NOREADONLYRETURN|OFN_HIDEREADONLY;
               if (GetSaveFileName(&ofn)==TRUE)
                SetDlgItemText(hwndDlg,IDC_OPTIM,path2);
               return FALSE;
            case IDC_BLOG:
               ofn.lpstrFile = path3;
               ofn.nMaxFile = sizeof(path3);
               ofn.lpstrFilter = "Log files\0*.LOG\0Text files\0*.TXT\0All\0*.*\0";
               ofn.lpstrDefExt= "LOG\0";
               ofn.Flags = OFN_PATHMUSTEXIST | OFN_NOREADONLYRETURN|OFN_HIDEREADONLY;
               if (GetSaveFileName(&ofn)==TRUE)
                SetDlgItemText(hwndDlg,IDC_LOG,path3);
               return FALSE;
            case IDOK:
               EndDialog(hwndDlg, wParam);
               return FALSE;
            case IDC_START:
               if(MessageBox(NULL,
               "Just to let you know:\nThis program will not update the screen, except for the stat
us window.\nThere is also no way to cancel (except, of coarse, Ctrl-alt-del)"
               , "Inportant message", MB_OKCANCEL)==IDCANCEL)return FALSE;
               GetDlgItemText(hwndDlg,IDC_ORIG,path1,256);
               GetDlgItemText(hwndDlg,IDC_OPTIM,path2,256);
               GetDlgItemText(hwndDlg,IDC_LOG,path3,256);
/*
              sprintf(teststring,"%s\n%s\n%s\nVerbose: %i\nLoged: %i",
              path1,path2,path3,
              IsDlgButtonChecked(hwndDlg,IDC_VERBOSE)==BST_CHECKED?1:0,
              IsDlgButtonChecked(hwndDlg,IDC_CHECKLOG)==BST_CHECKED?1:0);
              MessageBox(NULL, teststring, "Feedback", MB_OK);
*/
               SetDlgItemText(hwndDlg,IDC_STATUS,"Opening main file");


lastinclude=0;
array_size=0;

fnsplit(path1,driv1,dir1,name1,ext1);


if((in=fopen(path1,"rb"))==NULL)
{
```

```
   MessageBox(NULL, "Can't open input file", "Feedback", MB_OK);
   return FALSE;
}
 fseek(in,0,2);
 array_size=ftell(in);
 fseek(in,0,0);
 fread(bigarray,array_size,1,in);
 fclose(in);
if((out=fopen(path2,"wb"))==NULL)
{
   MessageBox(NULL, "Can't open output file", "Feedback", MB_OK);
   return FALSE;
}
 remove_comments(hwndDlg);
 c=1;
 while(c==1)
 {
  c=insert_includes(hwndDlg);
  if(c==1)remove_comments(hwndDlg);
  else if(c==3)return FALSE;
 }
 while(change_defs(hwndDlg));
 remove_deadspace(hwndDlg);
 remove_deadspace2(hwndDlg);
// remove_deadspace(hwndDlg);
 fwrite(bigarray,array_size,1,out);
 fclose(out);

                SetDlgItemText(hwndDlg,IDC_STATUS,"Writing output to file");
                SetDlgItemText(hwndDlg,IDC_STATUS,"Idle");
               return FALSE;
         }
         break;
   }
   return FALSE;
}









void NewWorldProc(HWND hwndDlg)
{
//here2
char repstring[2048];
char repstring2[80];
char reptitle[80];
               sprintf(reptitle,"No information");
               sprintf(repstring,"");
if(sprite[CURSPRITE].lotag==10)
sprintf(repstring2,"Door auto closes in %i seconds (hitag %i)",div(sprite[CURSPRITE].hitag,32).quot
,sprite[CURSPRITE].hitag);

               if(sprite[CURSPRITE].picnum==2200 && info_enable)
               {
               sprintf(repstring,"\
 pal 10 - Assault rifle\r\n\
 pal 11 - Shells\r\n\
 pal 12 - Grenades\r\n\
 pal 13 - Lascannon\r\n\
 pal 14 - Battery\r\n\
 pal 15 - Clip\r\n\
 pal 16 - Flamethrower\r\n\
 pal 17 - Pipebombs\r\n\
 pal 18 - Armour");
sprintf(reptitle,"Purchase (2200) - %s",
sprite[CURSPRITE].pal==10?"Assault rifle":
sprite[CURSPRITE].pal==11?"Shells":
sprite[CURSPRITE].pal==12?"Grenades":
sprite[CURSPRITE].pal==13?"Lascannon":
sprite[CURSPRITE].pal==14?"Battery":
```

```
sprite[CURSPRITE].pal==15?"Clip":
sprite[CURSPRITE].pal==16?"Flamethrower":
sprite[CURSPRITE].pal==17?"Pipebombs":
sprite[CURSPRITE].pal==18?"Armour":
"Invalid palette"
);}
                else
                if(sprite[CURSPRITE].picnum==23 && info_enable)
                sprintf(reptitle,"Credit Clip (23)");
                else
                if(sprite[CURSPRITE].picnum==489 && info_enable)
{
                sprintf(reptitle,"People (489)%s",
sprite[CURSPRITE].pal==9?" - Operative (flame thrower)":
sprite[CURSPRITE].pal==10?" - Personnel acquistion target":
sprite[CURSPRITE].pal==13?" - Operative (assault rifle)":
sprite[CURSPRITE].pal==14?" - Operative (pistol)":
" - Civilian"
);
                sprintf(repstring,"\
 pal 9  - Operative (flame thrower)\r\n\
 pal 10 - Personnel acquisition target\r\n\
 pal 13 - Operative (assault rifle)\r\n\
 pal 14 - Operative (pistol)\r\n\
 other  - Civilian");}
                else
                if(sprite[CURSPRITE].picnum==1680 && info_enable)
                {
                sprintf(reptitle,"Delayed person (1680) - %s",
sprite[CURSPRITE].pal==14?"Operative (pistol)":
"Non-uniformed (pistol)"
);
                sprintf(repstring,"\
 pal 14 - Operative (pistol)\r\n\
 other  - Non-uniformed operative\r\n\nnote: un-uniformed operatives carry pistols and do not drop
ammunition and weaponry");}
                else
                if(sprite[CURSPRITE].picnum==1715 && info_enable)
                {
                sprintf(reptitle,"Delayed person (1715) - %s",
sprite[CURSPRITE].pal==13?"Operative (assault rifle)":
sprite[CURSPRITE].pal==9?"Operative (flame thrower)":
"Non-uniformed (assault rifle)"
);
                sprintf(repstring,"\
 pal 13 - Operative (assault rifle)\r\n\
 pal 9 - Operative (flame thrower)\r\n\
 other  - Non-uniformed operative\r\n\nnote: un-uniformed operatives carry assault rifles and do no
t drop ammunition, weaponry and access cards.\n\rUniformed operatives carry access cards.");}
                else
                if(sprite[CURSPRITE].picnum==487 && info_enable)
                {
                sprintf(reptitle,"Panel (487) - %s",
sprite[CURSPRITE].pal==2?
   sector[sprite[CURSPRITE].sectnum].ceilingstat&2&&
   sector[sprite[CURSPRITE].sectnum].floorstat&2
   ?"\nSurface damaging - warning: do not use on gradient":
sprite[CURSPRITE].z==sector[sprite[CURSPRITE].sectnum].ceilingz&&
   !sector[sprite[CURSPRITE].sectnum].ceilingstat&2
   ?"\nCeiling surface damaging":
sprite[CURSPRITE].z!=sector[sprite[CURSPRITE].sectnum].floorz&&
   !sector[sprite[CURSPRITE].sectnum].floorstat&2
   ?"\nFloor surface damaging":
   !sector[sprite[CURSPRITE].sectnum].ceilingstat&2&&
   !sector[sprite[CURSPRITE].sectnum].floorstat&2
   ?"\nSurface damaging - warning: appears to far from surface":
"\nSurface damaging - warning: on a gradient or to far from surface"

:
sprite[CURSPRITE].pal==0?"\nWall damaging":
sprite[CURSPRITE].pal==10?"Ladder":
sprite[CURSPRITE].pal==11?"Blue flag*":
sprite[CURSPRITE].pal==12?"Damagable wall panel":
sprite[CURSPRITE].pal==13?"Red flag*":
sprite[CURSPRITE].pal==14?"Blue team*":
sprite[CURSPRITE].pal==15?"Red team*":
```

```
"Invalid palette");
                sprintf(repstring,"\
 pal 0  - Wall damaging - damaged by area of effect weapons\r\n\
 pal 2  - Surface damaging (floor and ceiling) - damaged by area of effect weapons\r\n\
 pal 10 - Ladder\r\n\
 pal 11 - Blue flag*\r\n\
 pal 12 - Damagable wall panel\r\n\
 pal 13 - Red flag*\r\n\
 pal 14 - Blue team*\r\n\
 pal 15 - Red team*\r\n\r\n\
(*) These are used for capture the flag maps." );
 }
                else
                if(sprite[CURSPRITE].picnum==3400 && info_enable)
                {
                sprintf(reptitle,"Glass (3400) - %s",
sprite[CURSPRITE].pal==2?"Sugar glass":
sprite[CURSPRITE].pal==10?"Glass (normal reflection)":
sprite[CURSPRITE].pal==11?"Glass (strong reflection)":
"Invalid palette");
                sprintf(repstring,"\
 pal 2  - Sugar glass\r\n\
 pal 10 - Glass (normal reflection)\r\n\
 pal 11 - Glass (strong reflection)");
 }
                else if(sprite[CURSPRITE].picnum==51 && info_enable)
                sprintf(reptitle,"Equipment acquisition target (51)");
                else if(sprite[CURSPRITE].picnum==52 && info_enable)
                sprintf(reptitle,"Sick bay (52)");

                else if(sprite[CURSPRITE].picnum==130)
                sprintf(reptitle,"Access lock (130) - locked");
                else if(sprite[CURSPRITE].picnum==131)
                sprintf(reptitle,"Access lock (131) - access");

                else if(sprite[CURSPRITE].picnum==132)
                sprintf(reptitle,"Switch (132) - off (red)");
                else if(sprite[CURSPRITE].picnum==133)
                sprintf(reptitle,"Switch (133) - on (green)");

                else if(sprite[CURSPRITE].picnum==134)
                sprintf(reptitle,"Switch (134) - off (red)");
                else if(sprite[CURSPRITE].picnum==135)
                sprintf(reptitle,"Switch (135) - on (green)");

                else if(sprite[CURSPRITE].picnum==136)
                sprintf(reptitle,"Switch (136) - off (red)");
                else if(sprite[CURSPRITE].picnum==137)
                sprintf(reptitle,"Switch (137) - on (green)");

                else if(sprite[CURSPRITE].picnum==138)
                sprintf(reptitle,"Switch (138) - off (red)");
                else if(sprite[CURSPRITE].picnum==139)
                sprintf(reptitle,"Switch (139) - on (green)");

                else if(sprite[CURSPRITE].picnum==140)
                sprintf(reptitle,"Switch (140) - off (red)");
                else if(sprite[CURSPRITE].picnum==141)
                sprintf(reptitle,"Switch (141) - on (green)");

                else if(sprite[CURSPRITE].picnum==142)
                sprintf(reptitle,"End level (142)");

                else if(sprite[CURSPRITE].picnum==146)
                sprintf(reptitle,"Multiple position switch (146) - position 4");
                else if(sprite[CURSPRITE].picnum==147)
                sprintf(reptitle,"Multiple position switch (147) - position 1");
                else if(sprite[CURSPRITE].picnum==148)
                sprintf(reptitle,"Multiple position switch (148) - position 2");
                else if(sprite[CURSPRITE].picnum==149)
                sprintf(reptitle,"Multiple position switch (149) - position 3");

                else if(sprite[CURSPRITE].picnum==162)
                sprintf(reptitle,"Switch (162) - off (red)");
                else if(sprite[CURSPRITE].picnum==163)
                sprintf(reptitle,"Switch (163) - on (green)");
```

```
else if(sprite[CURSPRITE].picnum==164)
sprintf(reptitle,"Switch (164) - off (red)");
else if(sprite[CURSPRITE].picnum==165)
sprintf(reptitle,"Switch (165) - on (green)");

else if(sprite[CURSPRITE].picnum==166)
sprintf(reptitle,"Switch (166) - off (red)");
else if(sprite[CURSPRITE].picnum==167)
sprintf(reptitle,"Switch (167) - on (green)");

else if(sprite[CURSPRITE].picnum==168)
sprintf(reptitle,"Switch (168) - off (red)");
else if(sprite[CURSPRITE].picnum==169)
sprintf(reptitle,"Switch (169) - on (green)");

else if(sprite[CURSPRITE].picnum==170)
sprintf(reptitle,"Access lock (170) - locked");
else if(sprite[CURSPRITE].picnum==171)
sprintf(reptitle,"Access lock (171) - access");

else if(sprite[CURSPRITE].picnum==407)
sprintf(reptitle,"Fan (407) - undamaged");
else if(sprite[CURSPRITE].picnum==411)
sprintf(reptitle,"Fan (411) - broken");

else if(sprite[CURSPRITE].picnum==712)
sprintf(reptitle,"Light switch (712) - switch is down");
else if(sprite[CURSPRITE].picnum==713)
sprintf(reptitle,"Light switch (713) - switch is up");

else if(sprite[CURSPRITE].picnum==860)
sprintf(reptitle,"Switch (860) - off (red)");
else if(sprite[CURSPRITE].picnum==861)
sprintf(reptitle,"Switch (861) - on (green)");

else if(sprite[CURSPRITE].picnum==862)
sprintf(reptitle,"Switch (862) - locked (red)");
else if(sprite[CURSPRITE].picnum==863)
sprintf(reptitle,"Switch (863) - open (green)");

else if(sprite[CURSPRITE].picnum==864)
sprintf(reptitle,"Switch (864) - off (red)");
else if(sprite[CURSPRITE].picnum==865)
sprintf(reptitle,"Switch (865) - on (green)");

else if(sprite[CURSPRITE].picnum==901)
sprintf(reptitle,"Ball (901)");
else if(sprite[CURSPRITE].picnum==902)
sprintf(reptitle,"Que ball (902)");
else if(sprite[CURSPRITE].picnum==903)
sprintf(reptitle,"Pocket (903)");

else if(sprite[CURSPRITE].picnum==908)
sprintf(reptitle,"Tree (908)");
else if(sprite[CURSPRITE].picnum==910)
sprintf(reptitle,"Tree (910)");
else if(sprite[CURSPRITE].picnum==911)
sprintf(reptitle,"Cactai (911)");
else if(sprite[CURSPRITE].picnum==916)
sprintf(reptitle,"Fire extinguisher (916)");
else if(sprite[CURSPRITE].picnum==939)
sprintf(reptitle,"Damage cactai (939)");
else if(sprite[CURSPRITE].picnum==940)
sprintf(reptitle,"Mine (940)");
else if(sprite[CURSPRITE].picnum==950)
sprintf(reptitle,"Broken fire hydrant (950)");
else if(sprite[CURSPRITE].picnum==952)
sprintf(reptitle,"Bullet hole (952)");
else if(sprite[CURSPRITE].picnum==981)
sprintf(reptitle,"Fire hydrant (981)");

else if(sprite[CURSPRITE].picnum==994)
sprintf(reptitle,"Pipe (994)");
else if(sprite[CURSPRITE].picnum==995)
sprintf(reptitle,"Pipe (995)");
```

```
else if(sprite[CURSPRITE].picnum==996)
sprintf(reptitle,"Pipe (996)");
else if(sprite[CURSPRITE].picnum==997)
sprintf(reptitle,"Pipe (997) - broken version of 996");
else if(sprite[CURSPRITE].picnum==1005)
sprintf(reptitle,"Pipe (1005) - broken version of 994");
else if(sprite[CURSPRITE].picnum==1260)
sprintf(reptitle,"Pipe (1260) - broken version of 995");

else if(sprite[CURSPRITE].picnum==1111)
sprintf(reptitle,"Switch (1111) - off/closed (normal)");
else if(sprite[CURSPRITE].picnum==1112)
sprintf(reptitle,"Switch (1112) - on/open (depressed)");

else if(sprite[CURSPRITE].picnum==1122)
sprintf(reptitle,"Switch (1122) - off (unlit)");
else if(sprite[CURSPRITE].picnum==1123)
sprintf(reptitle,"Switch (1123) - on (lit)");

else if(sprite[CURSPRITE].picnum==1142)
sprintf(reptitle,"Switch (1142) - off (unlit)");
else if(sprite[CURSPRITE].picnum==1143)
sprintf(reptitle,"Switch (1143) - on (lit/iris open)");

else if(sprite[CURSPRITE].picnum==1113)
sprintf(reptitle,"Vent (1113) - undamaged");
else if(sprite[CURSPRITE].picnum==1114)
sprintf(reptitle,"Vent (1114) - damaged");

else if(sprite[CURSPRITE].picnum==595)
sprintf(reptitle,"Vent (595) - undamaged");
else if(sprite[CURSPRITE].picnum==596)
sprintf(reptitle,"Vent (596) - damaged");

else if(sprite[CURSPRITE].picnum==341)
sprintf(reptitle,"Vent panel (341) - damaged");
else if(sprite[CURSPRITE].picnum==342)
sprintf(reptitle,"Vent panel (342) - un damaged");
else if(sprite[CURSPRITE].picnum==343)
sprintf(reptitle,"Vent panel (343) - slightly damaged");

else if(sprite[CURSPRITE].picnum==1222)
sprintf(reptitle,"Mechanical claw (1222)");
else if(sprite[CURSPRITE].picnum==1232)
sprintf(reptitle,"Garbage can (1232)");
else if(sprite[CURSPRITE].picnum==2566)
sprintf(reptitle,"Tripbomb (2566) - already set");

else if(sprite[CURSPRITE].picnum==621)
sprintf(reptitle,"Security camera (621)");
else if(sprite[CURSPRITE].picnum==499)
sprintf(reptitle,"Security screen (499)");
else if(sprite[CURSPRITE].picnum==502)
sprintf(reptitle,"Security screen (502)");

else if(sprite[CURSPRITE].picnum==71 && info_enable)
sprintf(reptitle,"Sign (71) - \"Flamethrower\"");
else if(sprite[CURSPRITE].picnum==72 && info_enable)
sprintf(reptitle,"Sign (72) - \"M16\"");
else if(sprite[CURSPRITE].picnum==73 && info_enable)
sprintf(reptitle,"Sign (73) - \"Grenades\"");
else if(sprite[CURSPRITE].picnum==74 && info_enable)
sprintf(reptitle,"Sign (74) - \"Shells\"");
else if(sprite[CURSPRITE].picnum==75 && info_enable)
sprintf(reptitle,"Sign (75) - \"Lascannon\"");
else if(sprite[CURSPRITE].picnum==76 && info_enable)
sprintf(reptitle,"Sign (76) - \"Battery\"");
else if(sprite[CURSPRITE].picnum==77 && info_enable)
sprintf(reptitle,"Sign (77) - \"Clip\"");
else if(sprite[CURSPRITE].picnum==78 && info_enable)
sprintf(reptitle,"Sign (78) - \"Armour\"");
else if(sprite[CURSPRITE].picnum==79 && info_enable)
sprintf(reptitle,"Sign (79) - \"Pipebombs\"");
else if(sprite[CURSPRITE].picnum==1247)
sprintf(reptitle,"C-9 canister (1247)");
```

```
                else if(sprite[CURSPRITE].picnum==1312 && info_enable)
                sprintf(reptitle,"Screen target (1312) - end of game");

                else if(sprite[CURSPRITE].picnum==546)
                sprintf(reptitle,"Crack 1 (546)");
                else if(sprite[CURSPRITE].picnum==547)
                sprintf(reptitle,"Crack 2 (547)");
                else if(sprite[CURSPRITE].picnum==548)
                sprintf(reptitle,"Crack 3 (548)");
                else if(sprite[CURSPRITE].picnum==549)
                sprintf(reptitle,"Crack 4 (549)");

                else if(sprite[CURSPRITE].picnum==2)
                sprintf(reptitle,"Activator (2)");
                else if(sprite[CURSPRITE].picnum==3)
                sprintf(reptitle,"Touchplate (3)");
                else if(sprite[CURSPRITE].picnum==4)
                sprintf(reptitle,"Activator locked (4)");
                else if(sprite[CURSPRITE].picnum==5)
                sprintf(reptitle,"Music and SFX (5)");
                else if(sprite[CURSPRITE].picnum==6)
                sprintf(reptitle,"Locator (6)");
                else if(sprite[CURSPRITE].picnum==7)
                sprintf(reptitle,"Cycler (7)");
                else if(sprite[CURSPRITE].picnum==8)
                sprintf(reptitle,"Master switch (8)");
                else if(sprite[CURSPRITE].picnum==9 && info_enable)
                sprintf(reptitle,"Respawn (9) - Respawn doesn't seem to work with NW III TC");
                else if(sprite[CURSPRITE].picnum==9)
                sprintf(reptitle,"Respawn (9)");
                else if(sprite[CURSPRITE].picnum==10)
                sprintf(reptitle,"GP Speed (10)");

                else if(sprite[CURSPRITE].picnum==1405)
                {
                sprintf(reptitle,"A player start (1405) - %s",sprite[CURSPRITE].lotag?"Co-operative"
:"Dukematch");
                sprintf(repstring,"lotag 1 - co-operative\r\nother   - dukematch");
                }

                else if(sprite[CURSPRITE].picnum==55 && info_enable)
                {
                sprintf(reptitle,"Persuadatron (55)");
                sprintf(repstring,"Used for personnel acquisition missions with sprite 489 (human) o
f pal 10 and sprite 60 (drop zone).");
                }
                else if(sprite[CURSPRITE].picnum==60 && info_enable)
                {
                sprintf(reptitle,"Personnel acquisition drop zone (60)");
                sprintf(repstring,"Used with blue lock, with sprite 55 (drop zone) of pal 10 and spr
ite 60 (drop zone).");
                }
                else
                if(sprite[CURSPRITE].picnum==1)
{
sprintf(repstring,"\
Lotags\r\n\
 0  - Rotating sector\r\n\
 1  - Pivot point for SE 0\r\n\
 2  - Earthquake\r\n\
 3  - Random lights after shot out\r\n\
 4  - Random lights\r\n\
 6  - Train\r\n\
 7  - Transport (water with ST 1 and 2)\r\n\
 8  - Up open door lights\r\n\
 9  - Down open door lights\r\n\
 10 - Door auto close (hitag=delay)\r\n\
 11 - Rotate sector door (ST 23)\r\n\
 12 - Light switch\r\n\
 13 - C-9 explosion\r\n\
 14 - Train car\r\n\
 15 - Sliding door (ST 25)\r\n\
 16 - Rotate reactor sector\r\n\
 17 - Elevator transport (ST 15, SHT 1/0)\r\n\
 19 - Shot touchplate ceiling down\r\n\
 20 - Bridge (ST 27)\r\n\
```

```
 21 - Drop floor (ST 28)\r\n\
 22 - Prong (ST 29)\r\n\
 24 - Convairbelt\r\n\
 25 - Engine\r\n\
 27 - Camera for playback\r\n\
 29 - Float sector\r\n\
 30 - Two way train (ST 31)\r\n\
 31 - Floor rise\r\n\
 32 - Ceiling lower\r\n\
 33 - Earthquakes debris\r\n\
 34 - Shooting sector");
sprintf(reptitle,"Sector effector (1) - %s",
sprite[CURSPRITE].lotag==0?"Rotating sector":
sprite[CURSPRITE].lotag==1?"Pivot point for SE 0":
sprite[CURSPRITE].lotag==2?"Earthquake":
sprite[CURSPRITE].lotag==3?"Random lights after shot out":
sprite[CURSPRITE].lotag==4?"Random lights":
sprite[CURSPRITE].lotag==6?"Train":
sprite[CURSPRITE].lotag==7&&sector[sprite[CURSPRITE].sectnum].lotag==1?"Water transport - above sur
face":
sprite[CURSPRITE].lotag==7&&sector[sprite[CURSPRITE].sectnum].lotag==2?"Water transport - below sur
face":
sprite[CURSPRITE].lotag==7?"Transport":
sprite[CURSPRITE].lotag==8?"Up open door lights":
sprite[CURSPRITE].lotag==9?"Down open door lights":
sprite[CURSPRITE].lotag==10?repstring2:
sprite[CURSPRITE].lotag==11&&sector[sprite[CURSPRITE].sectnum].lotag==23?"Rotate sector door":
sprite[CURSPRITE].lotag==11?"Rotate sector door - needs ST 23":
sprite[CURSPRITE].lotag==12?"Light switch":
sprite[CURSPRITE].lotag==13?"C-9 explosion":
sprite[CURSPRITE].lotag==14?"Train car":
sprite[CURSPRITE].lotag==15&&sector[sprite[CURSPRITE].sectnum].lotag==25?"Sliding door":
sprite[CURSPRITE].lotag==15?"Sliding door - needs ST 25":
sprite[CURSPRITE].lotag==16?"Rotate reactor sector":
sprite[CURSPRITE].lotag==17&&sector[sprite[CURSPRITE].sectnum].lotag==15&&sector[CURSECT].hitag==0?
"Elevator transport - bottom floor":
sprite[CURSPRITE].lotag==17&&sector[sprite[CURSPRITE].sectnum].lotag==15&&sector[CURSECT].hitag==1?
"Elevator transport - top floor":
sprite[CURSPRITE].lotag==17&&sector[sprite[CURSPRITE].sectnum].lotag==15?"Elevator transport - unkn
own hitag":
sprite[CURSPRITE].lotag==17?"Elevator transport - needs ST 15":
sprite[CURSPRITE].lotag==19?"Shot touchplate ceiling down":
sprite[CURSPRITE].lotag==20&&sector[sprite[CURSPRITE].sectnum].lotag==27?"Bridge":
sprite[CURSPRITE].lotag==20?"Bridge - needs ST 27":
sprite[CURSPRITE].lotag==21&&sector[sprite[CURSPRITE].sectnum].lotag==28?"Drop floor":
sprite[CURSPRITE].lotag==21?"Drop floor - needs ST 28":
sprite[CURSPRITE].lotag==22&&sector[sprite[CURSPRITE].sectnum].lotag==29?"Prong":
sprite[CURSPRITE].lotag==22?"Prong - needs ST 29":
sprite[CURSPRITE].lotag==24?"Convairbelt":
sprite[CURSPRITE].lotag==25?"Engine":
sprite[CURSPRITE].lotag==27?"Camera for playback":
sprite[CURSPRITE].lotag==29?"Float sector":
sprite[CURSPRITE].lotag==30&&sector[sprite[CURSPRITE].sectnum].lotag==31?"Two way train":
sprite[CURSPRITE].lotag==30?"Two way train - needs ST 31":
sprite[CURSPRITE].lotag==31?"Floor rise":
sprite[CURSPRITE].lotag==32?"Ceiling lower":
sprite[CURSPRITE].lotag==33?"Earthquakes debris":
sprite[CURSPRITE].lotag==34?"Shooting sector":
"Unknown"
 );}
//              else
//              {
//              sprintf(reptitle,"No information");
//              sprintf(repstring,"");
//              }
              SetDlgItemText(hwndDlg,IDC_INFOTITLE,reptitle);
              SetDlgItemText(hwndDlg,IDC_INFOBOX,repstring);
              if(auto_update){
              Right();Top();Front();
FrontSingle(RED_INDEX);
FrontWall(YELLOW_INDEX);
              SwapBuffers(hDC);
              }
}
```

```
            char EXTERNAL_DUKE[80]="duke3d.exe";
            char EXTERNAL_SETUP[80]="setup.exe";
            char EXTERNAL_BUILD[80]="build.exe";
            char EXTERNAL_EDITART[80]="editart.exe";
            char EXTERNAL_SETUPFILE[80]="duke3d.cfg";
            long int EXTERNAL_PLAYERS=1;

BOOL CALLBACK ExternalProc(HWND hwndDlg, UINT message, WPARAM wParam, LPARAM lParam)
/*
BOOL CALLBACK ExternalProc(hwndDlg, message, wParam, lParam)
HWND hwndDlg;
UINT message;
WPARAM wParam;
//LPARAM lParam;
*/
{

static DWORD aIds[] = {
    IDC_N_DUKE,       IDH_E_N_DUKE,
    IDC_N_SETUP,      IDH_E_N_SETUP,
    IDC_N_BUILD,      IDH_E_N_BUILD,
    IDC_N_EDITART,    IDH_E_N_EDITART,
    IDC_N_SETUPFILE, IDH_E_N_CONFIG,
    IDC_N_PLAYERS,    IDH_E_N_PLAYERS,
    IDC_RUN_DUKE,     IDH_E_S_DUKE,
    IDC_RUN_SETUP,    IDH_E_S_SETUP,
    IDC_RUN_BUILD,    IDH_E_S_BUILD,
    IDC_RUN_EDITART, IDH_E_S_EDITART,
    0,0
};
char repstring[80]="";

   switch (message)
   {
    case WM_HELP:
        WinHelp(((LPHELPINFO) lParam)->hItemHandle, "openglbt.hlp",
            HELP_WM_HELP, (DWORD) (LPSTR) aIds);

        break;

    case WM_CONTEXTMENU:
        WinHelp((HWND) wParam, "openglbt.hlp", HELP_CONTEXTMENU,
            (DWORD) (LPVOID) aIds);
        break;

      case WM_INITDIALOG:
            SetDlgItemText(hwndDlg,IDC_N_DUKE,EXTERNAL_DUKE);
            SetDlgItemText(hwndDlg,IDC_N_SETUP,EXTERNAL_SETUP);
            SetDlgItemText(hwndDlg,IDC_N_BUILD,EXTERNAL_BUILD);
            SetDlgItemText(hwndDlg,IDC_N_EDITART,EXTERNAL_EDITART);
            SetDlgItemText(hwndDlg,IDC_N_SETUPFILE,EXTERNAL_SETUPFILE);
            SetDlgItemInt(hwndDlg,IDC_N_PLAYERS,EXTERNAL_PLAYERS,TRUE);
         break;
            case WM_CLOSE:
               GetDlgItemText(hwndDlg,IDC_N_DUKE,EXTERNAL_DUKE,80);
               GetDlgItemText(hwndDlg,IDC_N_SETUPFILE,EXTERNAL_SETUPFILE,80);
               GetDlgItemText(hwndDlg,IDC_N_BUILD,EXTERNAL_BUILD,80);
               GetDlgItemText(hwndDlg,IDC_N_EDITART,EXTERNAL_EDITART,80);
               EXTERNAL_PLAYERS=GetDlgItemInt(hwndDlg,IDC_N_PLAYERS,NULL,TRUE);
               if(EXTERNAL_PLAYERS<1)
                  EXTERNAL_PLAYERS=1;
               else if(EXTERNAL_PLAYERS>8)
                  EXTERNAL_PLAYERS=8;
               EndDialog(hwndDlg, wParam);
               return FALSE;
      case WM_COMMAND:
         switch (LOWORD(wParam))
         {


            case IDC_RUN_DUKE:
               GetDlgItemText(hwndDlg,IDC_N_DUKE,EXTERNAL_DUKE,80);
               GetDlgItemText(hwndDlg,IDC_N_SETUPFILE,EXTERNAL_SETUPFILE,80);
               EXTERNAL_PLAYERS=GetDlgItemInt(hwndDlg,IDC_N_PLAYERS,NULL,TRUE);
               if(EXTERNAL_PLAYERS<1)
```

```
                {
                   EXTERNAL_PLAYERS=1;
                   SetDlgItemInt(hwndDlg,IDC_N_PLAYERS,EXTERNAL_PLAYERS,TRUE);
                }
                else if(EXTERNAL_PLAYERS>8)
                {
                   EXTERNAL_PLAYERS=8;
                   SetDlgItemInt(hwndDlg,IDC_N_PLAYERS,EXTERNAL_PLAYERS,TRUE);
                }

                if(EXTERNAL_PLAYERS==1)
{
                sprintf(repstring,"%s -map %s SETUPFILE %s",EXTERNAL_DUKE,szFile,EXTERNAL_SETUPFILE)
;
                system(repstring);
}               else
                {
                sprintf(repstring,"%s -map %s /q%li SETUPFILE %s",EXTERNAL_DUKE,szFile,EXTERNAL_PLAY
ERS,EXTERNAL_SETUPFILE);
                system(repstring);
                }
            break;
            case IDC_RUN_SETUP:
               GetDlgItemText(hwndDlg,IDC_N_SETUPFILE,EXTERNAL_SETUPFILE,80);
               GetDlgItemText(hwndDlg,IDC_N_BUILD,EXTERNAL_BUILD,80);
               sprintf(repstring,"%s SETUPFILE %s",EXTERNAL_SETUP,EXTERNAL_SETUPFILE);
               system(repstring);
            break;
            case IDC_RUN_BUILD:
               GetDlgItemText(hwndDlg,IDC_N_BUILD,EXTERNAL_BUILD,80);
               sprintf(repstring,"%s %s",EXTERNAL_BUILD,szFile);
               system(repstring);
            break;
            case IDC_RUN_EDITART:
               GetDlgItemText(hwndDlg,IDC_N_EDITART,EXTERNAL_EDITART,80);
               sprintf(repstring,"%s",EXTERNAL_EDITART);
               system(repstring);
            break;
            case IDOK:
               GetDlgItemText(hwndDlg,IDC_N_DUKE,EXTERNAL_DUKE,80);
               GetDlgItemText(hwndDlg,IDC_N_SETUPFILE,EXTERNAL_SETUPFILE,80);
               GetDlgItemText(hwndDlg,IDC_N_BUILD,EXTERNAL_BUILD,80);
               GetDlgItemText(hwndDlg,IDC_N_EDITART,EXTERNAL_EDITART,80);
               EXTERNAL_PLAYERS=GetDlgItemInt(hwndDlg,IDC_N_PLAYERS,NULL,TRUE);
               if(EXTERNAL_PLAYERS<1)
                  EXTERNAL_PLAYERS=1;
               else if(EXTERNAL_PLAYERS>8)
                  EXTERNAL_PLAYERS=8;
               EndDialog(hwndDlg, wParam);
               return FALSE;
         }
         break;
   }
   return FALSE;
}









BOOL CALLBACK SpriteItemProc(HWND hwndDlg, UINT message, WPARAM wParam, LPARAM lParam)
/*
BOOL CALLBACK SpriteItemProc(hwndDlg, message, wParam, lParam)
HWND hwndDlg;
UINT message;
WPARAM wParam;
//LPARAM lParam;
*/
{
```

```
static DWORD aIds[] = {
    IDS_X,    IDH_S_X,
    IDS_Y,    IDH_S_Y,
    IDS_Z,    IDH_S_Z,
    IDS_BLOCK,    IDH_S_BLOCKING,
    IDS_TRANS,    IDH_S_TRANSLUCENT,
    IDS_TRANSREV,    IDH_S_REVERSING,
    IDS_HITSCAN,    IDH_S_HITSCAN,
    IDS_XFLIP,    IDH_S_X_FLIP,
    IDS_YFLIP,    IDH_S_Y_FLIP,
    IDS_FACEWALLFLOOR,    IDH_S_FACE,
    IDS_ONESIDE,    IDH_S_SINGLE,
    IDS_CENTRE,    IDH_S_CENTRE,
    IDS_INVISIBLE,    IDH_S_INVISIBLE,
    IDS_PICNUM,    IDH_S_PICNUM,
    IDS_SHADE,    IDH_S_SHADE,
    IDS_PAL,    IDH_S_PAL,
    IDS_CLIPDIST,    IDH_S_CLIPDIST,
    IDS_XREP,    IDH_S_XREPEAT,
    IDS_YREP,    IDH_S_YREPEAT,
    IDS_XOFF,    IDH_S_XOFFSET,
    IDS_YOFF,    IDH_S_YOFFSET,
    IDS_SECTNUM,    IDH_S_SECTNUM,
    IDS_AUTOUPDATE,    IDH_S_AUTO,
    IDS_UPDATE,    IDH_S_UPDATE,
    IDS_LAST,    IDH_S_LAST,
    IDS_NEXT,    IDH_S_NEXT,
    IDS_FLAST,    IDH_S__LAST,
    IDS_FNEXT,    IDH_S__NEXT,
    IDS_CSPRITE,    IDH_S_SPRITENUM_SEL,
    IDS_SPRITE,    IDH_S_SPRITENUM,
    IDC_ENABLE,    IDH_S_ENABLE,
    IDC_INFOTITLE,    IDH_S_INFO,
    IDC_INFOBOX,    IDH_S_INFOBOX,
    IDS_STATNUM,    IDH_S_STATNUM,
    IDS_ANG,    IDH_S_ANG,
    IDS_XVEL,    IDH_S_XVEL,
    IDS_YVEL,    IDH_S_YVEL,
    IDS_ZVEL,    IDH_S_ZVEL,
    IDS_OWNER,    IDH_S_OWNER,
    IDS_EXTRA,    IDH_S_EXTRA,
    IDS_LOTAG,    IDH_S_LOTAG,
    IDS_HITAG,    IDH_S_HITAG,
    IDS_CSECTNUM,    IDH_S_SECTNUMSEL,
    0,0
};

   switch (message)
   {
    case WM_HELP:
        WinHelp(((LPHELPINFO) lParam)->hItemHandle, "openglbt.hlp",
            HELP_WM_HELP, (DWORD) (LPSTR) aIds);

        break;

    case WM_CONTEXTMENU:
        WinHelp((HWND) wParam, "openglbt.hlp", HELP_CONTEXTMENU,
            (DWORD) (LPVOID) aIds);
        break;

      case WM_INITDIALOG:
         SetupSpriteProc(hwndDlg);
               NewWorldProc(hwndDlg);
            if(auto_update)
            SetDlgItemText(hwndDlg,IDS_AUTOUPDATE,"Disable auto update");
            else
            SetDlgItemText(hwndDlg,IDS_AUTOUPDATE,"Enable auto update");
//       GETDIRLISTING
         break;
      case WM_COMMAND:
         switch (LOWORD(wParam))
         {
```

```
            case IDC_ENABLE:
               if(info_enable)
               {SetDlgItemText(hwndDlg,IDC_ENABLE,"Enable NW III TC sprite information");info_enabl
e=0;}
               else
               {SetDlgItemText(hwndDlg,IDC_ENABLE,"Disable NW III TC sprite information");info_enab
le=1;}
               NewWorldProc(hwndDlg);
            break;
            case IDOK:
               SaveSpriteProc(hwndDlg);
               EndDialog(hwndDlg, wParam);
               return FALSE;
            case IDCANCEL:
               EndDialog(hwndDlg, wParam);
               return FALSE;
            case IDS_CSPRITE:
               CURSPRITE=GetDlgItemInt(hwndDlg,IDS_SPRITE,NULL, TRUE);
               if(CURSPRITE<0)CURSPRITE=numsprites-1;
               else if(CURSPRITE>=numsprites)CURSPRITE=0;
               SetupSpriteProc(hwndDlg);
               NewWorldProc(hwndDlg);
               break;
            case IDS_UPDATE:
         Right();Top();Front();
         SwapBuffers(hDC);
               break;
            case IDS_AUTOUPDATE:
            if(auto_update)
            {auto_update=0;SetDlgItemText(hwndDlg,IDS_AUTOUPDATE,"Enable auto update");}
            else
            {auto_update=1;SetDlgItemText(hwndDlg,IDS_AUTOUPDATE,"Disable auto update");}
            break;
            case IDS_NEXT:
               SaveSpriteProc(hwndDlg);
               CURSPRITE++;
               if(CURSPRITE>=numsprites)CURSPRITE=0;
               SetupSpriteProc(hwndDlg);
               NewWorldProc(hwndDlg);
               break;
            case IDS_LAST:
               SaveSpriteProc(hwndDlg);
               CURSPRITE--;
               if(CURSPRITE<0)CURSPRITE=numsprites-1;
               SetupSpriteProc(hwndDlg);
               NewWorldProc(hwndDlg);
               break;
            case IDS_FNEXT:
               SaveSpriteProc(hwndDlg);
               CURSPRITE+=10;
               if(CURSPRITE>=numsprites)CURSPRITE=0;
               SetupSpriteProc(hwndDlg);
               NewWorldProc(hwndDlg);
               break;
            case IDS_FLAST:
               SaveSpriteProc(hwndDlg);
               CURSPRITE-=10;
               if(CURSPRITE<0)CURSPRITE=numsprites-1;
               SetupSpriteProc(hwndDlg);
               NewWorldProc(hwndDlg);
               break;
//                SaveSectorProc(hwndDlg);SaveWallProc(hwndDlg);
//           IDC_RADIOBUTTON1
            case IDS_CSECTNUM:
               FrontSingle(BLUE_INDEX);
               SaveSpriteProc(hwndDlg);
               CURSECT=GetDlgItemInt(hwndDlg,IDS_SECTNUM,NULL, TRUE);
               if(CURSECT<0)CURSECT=numsectors-1;
               else if(CURSECT>=numsectors)CURSECT=0;
//            SetupSectorProc(hwndDlg);
               FrontSingle(RED_INDEX);
               FrontWall(YELLOW_INDEX);
               SwapBuffers(hDC);
               break;

         }
```

```
        break;
    }
    return FALSE;
}
```

```
BOOL CALLBACK AboutProc(HWND hwndDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    switch (message)
    {

       case WM_COMMAND:
          switch (LOWORD(wParam))
          {
             case IDOK:
                SaveSpriteProc(hwndDlg);
                EndDialog(hwndDlg, wParam);
                return FALSE;
             case IDCANCEL:
                EndDialog(hwndDlg, wParam);
                return FALSE;
          }
          break;
    }
    return FALSE;
}
```

```
void help(void)
{
MessageBox(NULL, "\
This version of OpenGL Build Touch only has a brief help file so hence this prologue.\n\n\
Please note:\n\
  Please use a resolution of 1024x800 or greater as some of the dialogue boxes are rather large.\n\
n\
Controls (these buttons are for the main screen only, which is the one with the map)\n\
  escape\tquit\n\
  F1\tthis information screen\n\
  1\ttoggle one sided wall verteces\n\
  2\t2d mode\n\
  3\t3d mode\n\
  4\ttoggle wall verteces\n\
  5\topens sprite editing window\n\
  6\tprevious sprite\n\
  7\tnext sprite\n\
  8\tprevious sector\n\
  9\tnext sector\n\
  0/o\topens main editing window (walls, sectors and file management)\n\
  -=_\tprevious sprite\n\
  =/+\tnext sprite\n\
  e\topens the external program manager\n\
  up\tscrolls map up\n\
  down\tscrolls map down\n\
  left\tscrolls map left\n\
  right\tscrolls map right\n\
\nA menu has been added, as well as group file extraction. This can be accessed via the menu. There
 is context help in the editing screens.", "Information", MB_ICONINFORMATION|MB_OK);
}
```

```
void redraw(void)
{
         Right();
         Front();
FrontSingle(RED_INDEX);
FrontWall(YELLOW_INDEX);
         Top();
    SwapBuffers(hDC);
}

//LRESULT APIENTRY
//WndProc(
LONG WINAPI WndProc (
    HWND hWnd,
    UINT message,
    WPARAM wParam,
    LPARAM lParam)
{
///*
OPENFILENAME ofn;       // common dialog box structure
ZeroMemory(&ofn, sizeof(OPENFILENAME));
ofn.lStructSize = sizeof(OPENFILENAME);
ofn.hwndOwner = hWnd;
ofn.nFilterIndex = 1;
ofn.lpstrFileTitle = NULL;
ofn.nMaxFileTitle = 0;
ofn.lpstrInitialDir = NULL;
ofn.lpstrFile = szFile;
ofn.nMaxFile = sizeof(szFile);
ofn.lpstrFilter = "Maps\0*.MAP\0All\0*.*\0";
ofn.lpstrDefExt= "MAP\0";
    switch (message) {
    case WM_COMMAND:
   switch(LOWORD(wParam))
   {
      case CM_WallSect:
         DialogBox(hinst,
         MAKEINTRESOURCE(DLG_DELETEITEM),
         hWnd, (DLGPROC)DeleteItemProc);
      break;
      case CM_Sprite:
         DialogBox(hinst,
         MAKEINTRESOURCE(DLG_SPRITE),
         hWnd, (DLGPROC)SpriteItemProc);
      break;
      case CM_Test:
                DialogBox(hinst,
                MAKEINTRESOURCE(DLG_TEST),
                hWnd, (DLGPROC)ReplaceItemProc);
      break;
      case CM_OPTIMISE:
                DialogBox(hinst,
                MAKEINTRESOURCE(TEST),
                hWnd, (DLGPROC)TestProc);
      break;
      case CM_External:
         DialogBox(hinst,
         MAKEINTRESOURCE(DLG_EXTERNAL),
         hWnd, (DLGPROC)ExternalProc);
      break;
      case CM_Patch:
         DialogBox(hinst,
         MAKEINTRESOURCE(STUB),
         hWnd, (DLGPROC)StubProc);
      break;
      case CM_Group:
/*
ofn.lpstrFile = filename;
ofn.nMaxFile = sizeof(filename);
      ofn.lpstrFilter = "Group files\0*.GRP\0All\0*.*\0";
      ofn.lpstrDefExt= "GRP\0";
```

```
      ofn.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST |OFN_NOREADONLYRETURN|OFN_HIDEREADONLY;
      if (GetOpenFileName(&ofn)==TRUE)
      {
         DialogBox(hinst,
         MAKEINTRESOURCE(DLG_GROUP),
         hWnd, (DLGPROC)GroupProc);
      }
*/

         DialogBox(hinst,
         MAKEINTRESOURCE(DLG_GROUP),
         hWnd, (DLGPROC)GroupProc);

      break;
      case CM_EXIT:
         DestroyWindow(hWnd);
      break;
      case CM_ABOUT:
         DialogBox(hinst,
         MAKEINTRESOURCE(ABOUT),
         hWnd, (DLGPROC)AboutProc);
//       help();
      break;
      case IDM_NEW:
if(MessageBox(NULL, "Are you sure that you wish to create a new map?","New Map", MB_YESNO)==IDYES)
   {
               numsprites=0;
               CURSECT=numsectors=0;
               CURWALL=numwalls=0;
               sprintf(szFile,"untitled.map");
               redraw();
   }
      break;
      case IDM_HELP:
        WinHelp(hWnd, "openglbt.hlp",
            HELP_CONTENTS,0);
        break;
      case IDM_SAVE:
         save();
      break;
      case IDM_SAVEAS:
ofn.lpstrFile = szFile;
ofn.nMaxFile = sizeof(szFile);
ofn.lpstrFilter = "Maps\0*.MAP\0All\0*.*\0";
ofn.lpstrDefExt= "MAP\0";
      ofn.Flags = OFN_PATHMUSTEXIST|OFN_NOREADONLYRETURN|OFN_HIDEREADONLY;
      if (GetSaveFileName(&ofn)==TRUE)
         save();
      else MessageBox(WindowFromDC(hDC), "Cannot save map!","Error While Saving",MB_OK);
      break;
      case IDM_OPEN:
ofn.lpstrFile = szFile;
ofn.nMaxFile = sizeof(szFile);
ofn.lpstrFilter = "Maps\0*.MAP\0All\0*.*\0";
ofn.lpstrDefExt= "MAP\0";
      ofn.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST |OFN_NOREADONLYRETURN|OFN_HIDEREADONLY;
      if (GetOpenFileName(&ofn)==TRUE)
      {
         if(load()==0)
         {
            Right();Top();Front();
            FrontSingle(RED_INDEX);
            FrontWall(YELLOW_INDEX);
            SwapBuffers(hDC);
            CURSECT=0;
            CURWALL=sector[CURSECT].wallptr;
         }
         else MessageBox(WindowFromDC(hDC), "Cannot open map!","Error While Loading",MB_OK);
      }
      break;
      }
    break;
    case WM_CREATE:
   /* initialize OpenGL rendering */
   hDC = GetDC(hWnd);
// bSetupPixelFormat(hDC);
```

```
   setupPalette(hDC);
   hGLRC = wglCreateContext(hDC);
   wglMakeCurrent(hDC, hGLRC);
   init();
   return 0;
    case WM_DESTROY:
   /* finish OpenGL rendering */
   if (hGLRC) {
       wglMakeCurrent(NULL, NULL);
       wglDeleteContext(hGLRC);
   }
   if (hPalette) {
       DeleteObject(hPalette);
   }
   ReleaseDC(hWnd, hDC);
   PostQuitMessage(0);
   return 0;
    case WM_SIZE:
   /* track window size changes */
// if (hGLRC) {
//     winWidth = (int) LOWORD(lParam);
//     winHeight = (int) HIWORD(lParam);
//     resize();
//     return 0;
// }
    case WM_PALETTECHANGED:
   /* realize palette if this is *not* the current window */
   if (hGLRC && hPalette && (HWND) wParam != hWnd) {
       UnrealizeObject(hPalette);
       SelectPalette(hDC, hPalette, FALSE);
       RealizePalette(hDC);
       redraw();
       break;
   }
   break;
    case WM_QUERYNEWPALETTE:
   /* realize palette if this is the current window */
   if (hGLRC && hPalette) {
       UnrealizeObject(hPalette);
       SelectPalette(hDC, hPalette, FALSE);
       RealizePalette(hDC);
       redraw();
       return TRUE;
   }
   break;
   case WM_CHAR:
   {
   switch (wParam) {
      case 'E':
      case 'e':
         DialogBox(hinst,
         MAKEINTRESOURCE(DLG_EXTERNAL),
         hWnd, (DLGPROC)ExternalProc);
         break;
      case '_':
      case '-':
if(CURSPRITE==0)CURSPRITE=numsprites-1;
else CURSPRITE--;
Right();Top();Front();
FrontSingle(RED_INDEX);
FrontWall(YELLOW_INDEX);
    SwapBuffers(hDC);
      break;
      case '+':
      case '=':
if(CURSPRITE>=numsprites-1)CURSPRITE=0;
         CURSPRITE++;
         Right();Top();Front();
FrontSingle(RED_INDEX);
FrontWall(YELLOW_INDEX);
    SwapBuffers(hDC);
      break;
   }
   }
   break;
    case WM_KEYDOWN:
```

```
    {
        switch (wParam) {
      case '1':
         if(show_one_side3d)show_one_side3d=0;
         else show_one_side3d=1;
         redraw3d();
      break;
      case '2':
         Front();
FrontSingle(RED_INDEX);
FrontWall(YELLOW_INDEX);
         Top();
         Right();
         SwapBuffers(hDC);
      break;
      case '3':
         redraw3d();
      break;
      case '4':
         if(show_wall3d)show_wall3d=0;
         else show_wall3d=1;
         redraw3d();
      break;
      case VK_LEFT:
         longinc += 10.0F;
         Front();
FrontSingle(RED_INDEX);
FrontWall(YELLOW_INDEX);
         Top();
//       redraw();
    SwapBuffers(hDC);
         break;
      case '8':
FrontSingle(BLUE_INDEX);
//FrontWall(BLUE_INDEX);
if(CURSECT==0)CURSECT=numsectors-1;
else CURSECT--;
FrontSingle(RED_INDEX);
FrontWall(YELLOW_INDEX);
    SwapBuffers(hDC);
      break;
      case '9':
FrontSingle(BLUE_INDEX);
//FrontWall(BLUE_INDEX);
if(CURSECT>=numsectors-1)CURSECT=0;
         CURSECT++;
FrontSingle(RED_INDEX);
FrontWall(YELLOW_INDEX);
    SwapBuffers(hDC);
      break;
      case '6':
//FrontSingle(BLUE_INDEX);
FrontWall(BLUE_INDEX);
if(CURWALL==0)CURWALL=numwalls-1;
else CURWALL--;
FrontSingle(RED_INDEX);
FrontWall(YELLOW_INDEX);
    SwapBuffers(hDC);
      break;
      case '7':
//FrontSingle(BLUE_INDEX);
FrontWall(BLUE_INDEX);
if(CURWALL>=numwalls-1)CURWALL=0;
         CURWALL++;
FrontSingle(RED_INDEX);
FrontWall(YELLOW_INDEX);
    SwapBuffers(hDC);
      break;
      case VK_RIGHT:
         longinc -= 10.0F;
         Front();
FrontSingle(RED_INDEX);
FrontWall(YELLOW_INDEX);
         Top();
//       redraw();
    SwapBuffers(hDC);
```

```
         break;
      case VK_UP:
         latinc += 10.0F;
         Front();
FrontSingle(RED_INDEX);
FrontWall(YELLOW_INDEX);
         Right();
//       redraw();
    SwapBuffers(hDC);
         break;
      case VK_DOWN:
         latinc -= 10.0F;
         Front();
FrontSingle(RED_INDEX);
FrontWall(YELLOW_INDEX);
         Right();
//       redraw();
    SwapBuffers(hDC);
         break;
case VK_F1:
help();
break;
      case VK_ESCAPE:
         DestroyWindow(hWnd);
         break;
      case 'C':
         DialogBox(hinst,
         MAKEINTRESOURCE(STUB),
         hWnd, (DLGPROC)StubProc);
         break;
      case 'O':
         DialogBox(hinst,
         MAKEINTRESOURCE(DLG_DELETEITEM),
         hWnd, (DLGPROC)DeleteItemProc);
         break;
      case '0':
         DialogBox(hinst,
         MAKEINTRESOURCE(DLG_DELETEITEM),
         hWnd, (DLGPROC)DeleteItemProc);
         break;

      case 'T':
//      case 't':
         DialogBox(hinst,
         MAKEINTRESOURCE(TEST),
         hWnd, (DLGPROC)TestProc);
         break;
      case '5':
         DialogBox(hinst,
         MAKEINTRESOURCE(DLG_SPRITE),
         hWnd, (DLGPROC)SpriteItemProc);
         break;
//      default:
//         break;
       }
     }
   break;
/*   case WM_MENUSELECT:
//   switch(LOWORD(wParam))
// {
//       case IDM_OPEN:
     ofn.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST |OFN_NOREADONLYRETURN|OFN_HIDEREADONLY;
     if (GetOpenFileName(&ofn)==TRUE)
     {
        if(load()==0)
        {
           Right();Top();Front();
           FrontSingle(RED_INDEX);
           FrontWall(YELLOW_INDEX);
           SwapBuffers(hDC);
           CURSECT=0;
           CURWALL=sector[CURSECT].wallptr;
        }
        else MessageBox(WindowFromDC(hDC), "Cannot open map!","Error While Loading",MB_OK);
     }
//    break;
```

```
// }
   break;
   //*/
    case WM_PAINT:
//    {
//     PAINTSTRUCT ps;
       BeginPaint(hWnd, &ps);
       if (hGLRC) {
      redraw();
       }
       EndPaint(hWnd, &ps);
       return 0;
//      }
// break;
//  case WM_COMMAND:
//                break;
//            return 0L;
// default:
// break;
    }
    return DefWindowProc(hWnd, message, wParam, lParam);
}
```

```
int  DoWMCommand(WPARAM wParam, HWND hWnd);
void DoLButtonDown(HWND hWnd, LONG lParam);
void DoLButtonUp(HWND hWnd, LONG lParam);
void DoMouseMove(HWND hWnd, LONG lParam);
void DoPaint(HWND hWnd);
void DrawShape(HDC hdc, int x, int y, int x2, int y2, int Shape,
               int PenWidth, COLORREF PenColor, int Slope);
LRESULT CALLBACK WndProc2 (HWND, UINT, WPARAM, LPARAM);

//   GLOBAL VARIABLES

#define ShapeI 100
#define STRICT

int ShapeNumber = -1;           // Indicates the number of shapes drawn.

int CurrentShape = LINE;        // The shape the user is drawing.

int PenWidth = 3;               // The current pen width.
                                // Default width is medium.

COLORREF PenColor = RGB(255, 128, 128);    // The current pen color.
                                        // Default is red.

typedef struct SSHAPE           // Struct which tracks the drawn shapes.
{
        RECT       Points;      // Location of shape.
        int        PenWidth;    // Pen width for the shape.
        int        Shape;       // Type of shape.
        COLORREF   PenColor;    // Color of shape.
        int        Slope;       // Used to draw lines correctly.
} SHAPE;
```

```
SHAPE Shapes[ShapeI];           // Array that stores the shapes.

#pragma argsused

int APIENTRY
WinMain(
    HINSTANCE hCurrentInst,
    HINSTANCE hPreviousInst,
    LPSTR lpszCmdLine,
    int nCmdShow)
{
    WNDCLASS wndClass;
    WNDCLASS wndClass2;
//    HWND hWnd;
    MSG msg;
    HINSTANCE hCurrentInst2=hCurrentInst;

_control87(MCW_EM, MCW_EM);
sprintf(szFile,"e1l1.map");

if(lpszCmdLine[0]!=0)

{

 if(lpszCmdLine[0]=='-'||lpszCmdLine[0]=='/'||lpszCmdLine[0]=='?')
 {
        MessageBox(NULL,"openglbt.exe [filename]\nopens the map file called filename.","OpenGL Buil
d Touch v1.9.1 - Command line help",MB_ICONINFORMATION|MB_OK);

   exit(0);
 }
 else
 sprintf(szFile,"%s",lpszCmdLine);
}
load();


classMenu=MAKEINTRESOURCE(OpenGLBT);
    wndClass.style = CS_SAVEBITS|CS_BYTEALIGNCLIENT|CS_BYTEALIGNWINDOW;
    wndClass.lpfnWndProc = (WNDPROC)WndProc;
    wndClass.cbClsExtra = 0;
    wndClass.cbWndExtra = 0;
    wndClass.hInstance = hCurrentInst;
    wndClass.hIcon = LoadIcon(hCurrentInst, MAKEINTRESOURCE(IDI_ICON1));
    wndClass.hCursor = LoadCursor(NULL, IDC_ARROW);
    wndClass.hbrBackground = GetStockObject(BLACK_BRUSH);
    wndClass.lpszMenuName = classMenu;
    wndClass.lpszClassName = className;
    RegisterClass(&wndClass);


    /* create window */
    hWnd = CreateWindow(
   className, windowName,
WS_SIZEBOX  |  WS_OVERLAPPEDWINDOW| WS_CLIPCHILDREN  ,//| WS_CLIPCHILDREN | WS_CLIPSIBLINGS,
// 0, 0, maxX, maxY,
   0, 0, 700, 700,
   NULL, NULL, hCurrentInst, NULL);
//CreateMenu();
//hWnd2=CreateStatusWindow(WS_CLIPCHILDREN|WS_VISIBLE|WS_CHILD,"Text",hWnd,1000);
    /* display window */
    ShowWindow(hWnd, nCmdShow);
     GetClientRect(    hWnd,    &rect    );
//    UpdateWindow(hWnd);



/*    while (GetMessage(&msg, NULL, 0, 0) == TRUE) {
   TranslateMessage(&msg);
   DispatchMessage(&msg);
    }
*/

    while (1) {
        while (PeekMessage(&msg, NULL, 0, 0, PM_NOREMOVE) == TRUE)
```

```
        {
            if (GetMessage(&msg, NULL, 0, 0) )
            {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
            } else {
                return TRUE;
            }
        }
//        drawScene();
//        redraw();
    }
//    return msg.wParam;
}
```

OpenGL Build Touch Source Code

openglbt.hpj

```
; This file is maintained by HCW. Do not modify this file directly.


[OPTIONS]
HCW=0
COMPRESS=12 Hall Zeck
LCID=0xc09 0x0 0x0 ;English (Australian)
REPORT=Yes
TITLE=OpenGL Build Touch v1.9.1 Help
COPYRIGHT=© James Ferry 1999.
HLP=.\openglbt.hlp

[FILES]
.\openglbt.rtf

[MAP]
#include .\defines.h
```