Course 3DC7, C1: OpenGL, Bauanleitung mit C++/MFC7.0

Projekt ogl1 mit leerem Fenster

`Microsoft Visual Studio.NET starten`
`File – New – Project – Project Types: Visual C++ Projects, Templates: MFC Application`
Name: `ogl1`
Location: `C:\temp`
Button `OK` unten Mitte klicken
Es meldet sich der `MFC Application Wizard - ogl1` mit der Seite `Overview`, die uninteressant ist.
Links unter `Overview` auf `Application Type` klicken.
Schritt1: **`Single document`** einschalten
Schritt 2: **`Document/View architecture support`** Checkbox ausschalten
Schritt 3: `Finish` VS.NET legt nun unter `C:/temp` ein Directory `ogl1` an und schreibt dorthin jede Menge Files, die insgesamt das Projekt `ogl1` bilden.
Schritt 4: Klicken Sie im Hauptmenu von VS.NET auf den Menupunkt `View` und dann auf den Unterpunkt `Class View Ctrl+Shift+C`.
Sie sehen im rechten Bereich des VS.NET-Hauptfensters das Unterfenster `Class View - draw1`.
An dessen unteren Rand die Registerkarte Klassen klicken.
Jetzt sehen Sie oben die Zeile `+ ogl1` , klicken Sie mit der rechten Maustaste auf `ogl1`.
Es öffnet sich ein Kontextmenü. Klicken Sie ganz unten auf `Properties`. Es öffnet sich ein Fenster `ogl1 Property Pages`.
Auf der linken Seite der `ogl1 Property Pages` öfnnen Sie den Eintrag `Linker`. Klicken Sie auf den Untereintrag `Input`.
Auf der rechten Seite der `ogl1 Property Pages` erscheint eine Tabelle. In deren erster Zeile steht `Additional Dependencies`.
Dahinter in das freie Feld in der zweiten Spalte schreiben Sie `opengl32.lib` und verlassen die `ogl1 Property Pages` mit `OK`.

Klicken Sie `Debug` in der Menü-Zeile oberhalb des Hauptfensters
Es öffnet sich ein Untermenü. Klicken Sie auf `Start Without Debugging Ctrl F5`
Es erscheint eine Message Box: `These project configuration(s) are out of date: ogl1 - Debug Win 32. Would you like to build them ?`
Sie antworten mit `Yes`

Falls alles ok, dann `ogl1.exe` erst beenden, bevor der erste Code eingegeben wird !

Präprozessorbefehle und Deklarationen in ChildView.h

Klicken Sie im Hauptmenu von VS.NET auf den Menupunkt `View` und dann auf den Unterpunkt `Class View Ctrl+Shift+C`.
Sie sehen im rechten Bereich des VS.NET-Hauptfensters das Unterfenster `Class View - draw1`.
An dessen unteren Rand die Registerkarte Klassen klicken.
Darin sehen Sie die Zeile `+ ogl1` , klicken Sie das Pluszeichen.
Sie sehen u.a. `+ CChildView`, doppelklicken Sie diesen Klassennamen.
Sie editieren jetzt das File `ChildView.h`, Schreiben Sie dort vor die Klasse `class CChildView : public CWnd` am besten unter die Zeile `#pragma once` die Präprozessoranweisungen:

```
#define nMax 100
#include < gl/gl.h >
```

Schreiben Sie in die Klasse direkt unter die sich öffnende geschweifte Klammer noch vor `//Construction` die Deklarationen:

```
private:
   CPoint old_vertex;
   CPoint p[nMax];
   typedef struct { float x; float y; float z; } FPoint;
   FPoint f[nMax];
   int    n;
   HGLRC  hglrc;
   BOOL   done;
```

Sie sehen nun diese Member-Variablen im Klassenbaum von `CChildView` im rechten `ClassView - ogl1` - Fenster, wenn Sie das Pluszeichen vor `CChildView` klicken.
Klicken Sie dort auf das Plus-Zeichen vor `CMainFrame`.
Doppelklicken Sie auf `CMainFrame::PreCreateWindow(...)`.
Ersetzen Sie in der Funktion `CMainFrame::PreCreateWindow(...)` die 2 TODO-Kommentarzeilen durch folgende zwei Befehle:

```
   cs.cx = 500;
   cs.cy = 500;
```

Diese Befehle initialisieren das MainFrame-Fenster quadratisch auf 500x500 Pixel.

Behandlungsroutinen für Windows-Nachrichten

Klicken Sie im Klassenbaum von `CChildView` mit der rechten Maustaste auf `CChildView` (nicht verwechseln mit dem gleichnamigen Konstruktor `CChildView(void)`).
Es öffnet sich ein Kontextmenü. Klicken Sie ganz unten im Kontextmenü auf `Properties`.
Es öffnet sich unter dem `Class View - ogl1` - Fenster ein `Properties` - Fenster mit der Überschrift `CChildview VCCodesClass`.
Im Toolbar dieses Fensters klicken Sie rechts neben dem gelben Blitz auf das `Messages` - Symbol, das aussieht wie ein weißer Topf mit blauem Deckel.
Sie sehen nun die Liste der Windows-Messages von `CChildView`.
Klicken Sie auf `WM_CREATE` und dann auf das schwarze Abwärtsdreieck rechts in dieser Zeile und dann auf `Add OnCreate`.
Klicken Sie auf `WM_LBUTTONDOWN` und dann auf das schwarze Abwärtsdreieck rechts in dieser Zeile und dann auf `Add OnLButtonDown`.
Klicken Sie auf `WM_MOUSEMOVE` und dann auf das schwarze Abwärtsdreieck rechts in dieser Zeile und dann auf `Add OnMouseMove`.
Klicken Sie auf `WM_LBUTTONUP` und dann auf das schwarze Abwärtsdreieck rechts in dieser Zeile und dann auf `Add OnLButtonUp`.
Klicken Sie auf `WM_TIMER` und dann auf das schwarze Abwärtsdreieck rechts in dieser Zeile und dann auf `Add OnTimer`.

Sie sehen nun u.a. die Methoden `OnCreate(...), OnLButtonDown(..), OnMouseMove(..), OnLButtonUp(..)` und `OnTimer(...)` im Klassenbaum von `CChildView`, und Sie sehen die fünf fast leeren Funktionsrümpfe in `CChildView.cpp`, die Visual Studio automatisch generiert hat.
Die ziemlich leeren Funktionshülsen stehen in `ChildView.cpp` unterhalb der fertigen Funktion `void CChildView::OnPaint()`. Verschieben Sie die Funktionshülsen samt Inhalt und geschweiften Klammern per Drag und Drop so, dass diese eine logisch vernünftige Reihenfolge bilden (das ist wichtig für den Überblick).

```
BOOL CChildView::PreCreateWindow(...)
int CChildView::OnCreate(...)
void CChildView::OnPaint()
void CChildView::OnLButtonDown(...)
void CChildView::OnMouseMove(...)
void CChildView::OnLButtonUp(...)
void CChildView::OnTimer(...)
```

Einfügen von Code

Doppelklicken Sie im Klassenbaum auf `OnCreate(...)`. (oder gehen Sie direkt in den Code in die Funktion `void CChildView::OnCreate(...)`). Ändern Sie die Funktion bis sie so aussieht:

```
int CChildView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{  if (CWnd ::OnCreate(lpCreateStruct) == -1) return -1;
   CClientDC dc( this );
   PIXELFORMATDESCRIPTOR pfd = {
      sizeof( PIXELFORMATDESCRIPTOR ),
      1,
      PFD_DRAW_TO_WINDOW | PFD_SUPPORT_OPENGL | PFD_DOUBLEBUFFER,
      PFD_TYPE_RGBA,
      24,0,0,0,0,0,0,0,0,0,0,0,0,32,0,0,
      PFD_MAIN_PLANE,0,0,0 };
   int i = ChoosePixelFormat( dc.m_hDC, &pfd );
   SetPixelFormat( dc.m_hDC, i, &pfd );
   hglrc = wglCreateContext( dc.m_hDC );
   done = FALSE;
   SetTimer(1, 1, NULL);
   return 0;
}
```

Füllen Sie die anderen Funktionen nach diesem Muster, bis sie so aussehen:

```
void CChildView::OnPaint()
{  CPaintDC dc(this);
   dc.TextOut( 0, 0, "Press the left mouse button and move!" );
}
void CChildView::OnLButtonDown(UINT nFlags, CPoint point)
{  done = FALSE;
   old_vertex = p[0] = point;
   n = 1;
   Invalidate();
}
void CChildView::OnMouseMove(UINT nFlags, CPoint point)
{  if (!nFlags) return;
   int dx = point.x - old_vertex.x;
   int dy = point.y - old_vertex.y;
   if ( dx*dx + dy*dy < 100 ) return;
   if ( n > nMax - 2 ) return;
   CClientDC dc(this);
   dc.MoveTo( old_vertex ); dc.LineTo( point );
   old_vertex = p[n++] = point;
}
```

```
void CChildView::OnLButtonUp(UINT nFlags, CPoint point)
{  CRect r; GetClientRect( r );
   int i, xnew, ynew;
   int halfwidth  = r.Width () / 2;
   int halfheight = r.Height() / 2;
   for ( i = 0; i < n; i++ )
   {   xnew = p[i].x - halfwidth;
       ynew = p[i].y - halfheight;
       ynew *= -1;
       f[i].x = float(xnew) / float(halfwidth);
       f[i].y = float(ynew) / float(halfheight);
       f[i].z = 0.f;
   }
   done = TRUE;
}
void CChildView::OnTimer(UINT nIDEvent)
{  if ( !done ) return;
   int i;
   CClientDC dc( this );
   wglMakeCurrent( dc.m_hDC, hglrc );
   glDrawBuffer( GL_BACK );
   glMatrixMode( GL_MODELVIEW );
   glRotatef( 2.0f, 1.f, 1.f, 1.f );
   glLineWidth( 3 );
   glPointSize( 9 );
   glColor3f( 0.f, 1.f, 0.f );
   glClearColor( 1.f, 1.f, 1.f, 0.f);
   glClear(GL_COLOR_BUFFER_BIT);
   glBegin( GL_LINE_LOOP );
      for ( i = 0; i < n; i++ )
         glVertex3f( f[i].x, f[i].y, f[i].z );
   glEnd();
   glBegin( GL_POINTS );
      for ( i = 0; i < n; i++ )
         glVertex3f ( f[i].x, f[i].y, f[i].z );
   glEnd();
   SwapBuffers( dc.m_hDC );
   dc.Polyline( p, n );
}
```

`ogl1` ist damit fertig, ausführen, erproben, beenden.

Experimente

Beenden Sie `ogl1`.
1. Untersuchen Sie Methoden zur Verlangsamung der Drehung
2. Untersuchen Sie die Drehung um einzelne Achsen
3. Experimentieren Sie mit verschiedenen Drehrichtungen.
4. Erproben Sie andere Zahlen in `glLineWidth` und `glPointSize`.
5. Erproben Sie andere Farben für Vorder- und Hintergrund
6. Versuchen Sie kleine Scrolls und Zooms

Vereinfachungen

Vereinfachen Sie das Programm, indem Sie das Malen mit der Maus ganz weglassen und ein festes Dreick, Viereck etc. vorprogrammieren.

Weitere Aufgaben

Klicken Sie auf Help in der Menüleiste von VS.NET. Klicken Sie auf das Untermenü Index.
Suchen Sie die Schlüsselwörter der Ihnen unbekannten Sprachelemente. Lesen Sie die Hilfetexte.