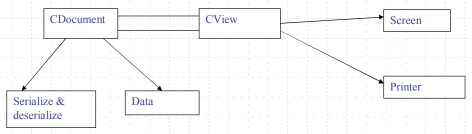
- Visual Studio MFC Programming
  - File->New-> MFC AppWizard (exe) -> SDI (MDI)
  - MFC Document/View structure



Events Driven and Messages Delivery (WM_Create, WM_Size, WM_LMouseDown etc)

Device Context and Device Independence

Same program use different hardware by letting windows take care of hardware interface

Program draws on an abstract surface called a "device context" (DC)

DC is accessed with a "handle to a DC", which can be get from windows by using GetDC( )

DC need to be released after have finished using it by ReleaseDC( )

Use OpenGL in MFC Programming

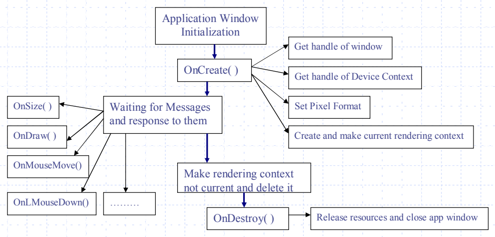
Structure of a typical OpenGL program

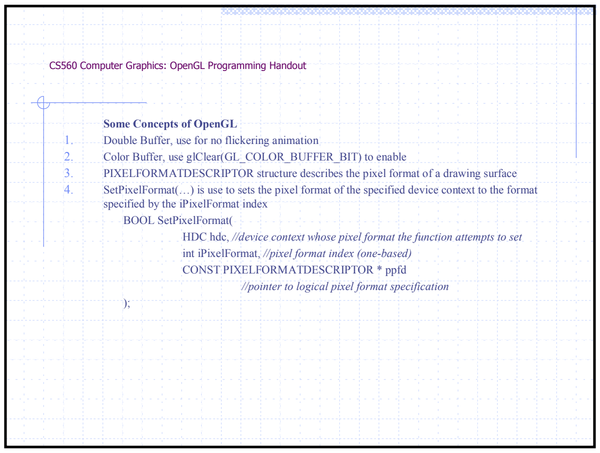
- Create application window (GetSafeHwnd())
- Get proper Device Context (m_hDC= GetDC())
- Set pixel format for windows (SetWindowPixelFormat(m_hDC))
- Create the OpenGL view context (CreateViewGLContext(m_hDC))
- Response to Messages, complete required tasks and clear resources

§ We need to create a new OpenGL rendering context that is suitable for drawing on the device referenced by device context. The rendering context has the same pixel format as the device context

Implementation, a small code example:

```
HDC hdc; HGLRC hglrc;
hglrc=wglCreateContext(hdc); //create rendering context
wglMakeCurrent(hdc, hglrc); //make it the current rendering context
//call OpenGL APIs to fulfill your task ……
//when the rendering context is no longer needed
wglMakeCurrent(NULL, NULL); //make the rendering context not current
wglDeleteContext(hglrc); //delete the rendering context
```

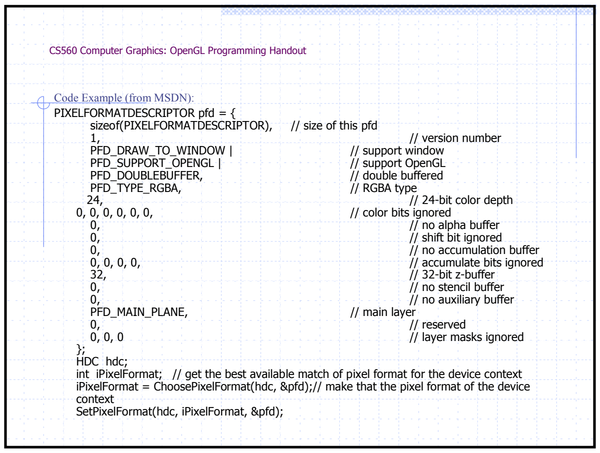

Application Window Initialization
OnCreate( )
Get handle of window
Get handle of Device Context
Set Pixel Format
Create and make current rendering context
Waiting for Messages and response to them
OnSize( )
OnDraw( )
OnMouseMove()
OnLMouseDown()
………
Make rendering context not current and delete it
OnDestroy( )
Release resources and close app window



Some Concepts of OpenGL

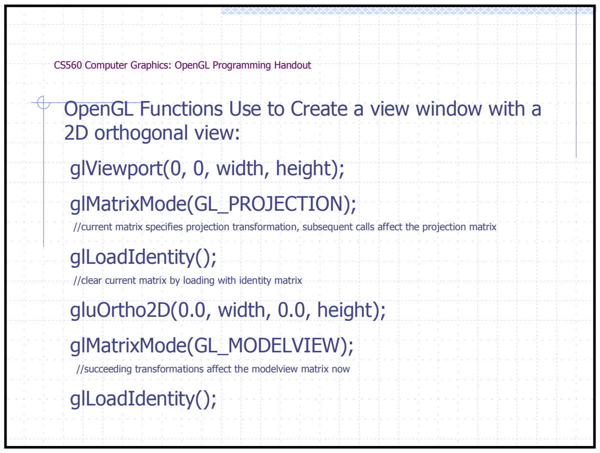
1. Double Buffer, use for no flickering animation
2. Color Buffer, use glClear(GL_COLOR_BUFFER_BIT) to enable
3. PIXELFORMATDESCRIPTOR structure describes the pixel format of a drawing surface
4. SetPixelFormat(…) is use to sets the pixel format of the specified device context to the format specified by the iPixelFormat index

```
BOOL SetPixelFormat(
        HDC hdc, //device context whose pixel format the function attempts to set
        int iPixelFormat, //pixel format index (one-based)
        CONST PIXELFORMATDESCRIPTOR * ppfd
                //pointer to logical pixel format specification
);
```



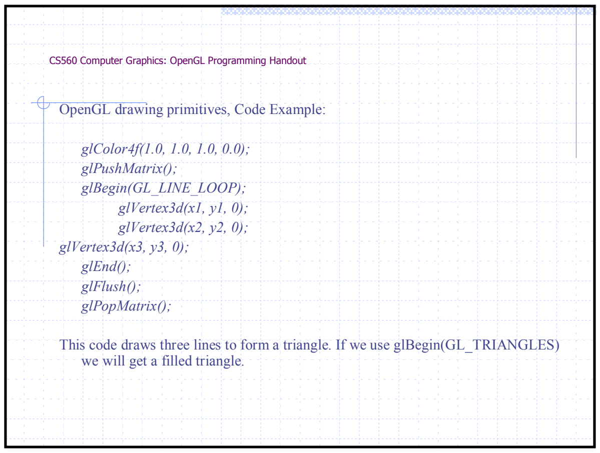
Code Example (from MSDN):

```
PIXELFORMATDESCRIPTOR pfd = {
        sizeof(PIXELFORMATDESCRIPTOR),    // size of this pfd
        1,                                              // version number
        PFD_DRAW_TO_WINDOW |                    // support window
        PFD_SUPPORT_OPENGL |                    // support OpenGL
        PFD_DOUBLEBUFFER,                       // double buffered
        PFD_TYPE_RGBA,                          // RGBA type
       24,                                              // 24-bit color depth
    0, 0, 0, 0, 0, 0,                           // color bits ignored
        0,                                              // no alpha buffer
        0,                                              // shift bit ignored
        0,                                              // no accumulation buffer
        0, 0, 0, 0,                                     // accumulate bits ignored
        32,                                             // 32-bit z-buffer
        0,                                              // no stencil buffer
        0,                                              // no auxiliary buffer
        PFD_MAIN_PLANE,                         // main layer
        0,                                              // reserved
        0, 0, 0                                         // layer masks ignored
    };
    HDC  hdc;
    int  iPixelFormat;   // get the best available match of pixel format for the device context
    iPixelFormat = ChoosePixelFormat(hdc, &pfd);// make that the pixel format of the device
    context
    SetPixelFormat(hdc, iPixelFormat, &pfd);
```



OpenGL Functions Use to Create a view window with a 2D orthogonal view:

```
glViewport(0, 0, width, height);
glMatrixMode(GL_PROJECTION);
 //current matrix specifies projection transformation, subsequent calls affect the projection matrix
glLoadIdentity();
 //clear current matrix by loading with identity matrix
gluOrtho2D(0.0, width, 0.0, height);
glMatrixMode(GL_MODELVIEW);
 //succeeding transformations affect the modelview matrix now
glLoadIdentity();
```



CS560 Computer Graphics: OpenGL Programming Handout

OpenGL drawing primitives, Code Example:

```
    glColor4f(1.0, 1.0, 1.0, 0.0);
    glPushMatrix();
    glBegin(GL_LINE_LOOP);
           glVertex3d(x1, y1, 0);
           glVertex3d(x2, y2, 0);
glVertex3d(x3, y3, 0);
    glEnd();
    glFlush();
    glPopMatrix();
```

This code draws three lines to form a triangle. If we use glBegin(GL_TRIANGLES) we will get a filled triangle.