Windows/OpenGL Graphics

CS134

Chris Pollett

Nov. 15, 2004.

Outline

- Doing Windows Graphics
- CDC
- Persistent Display
- Converting to Pixel Positions
- Memory-based Device Contexts
- Linking to OpenGL
- The OpenGL State Machine
- OpenGL Code in Windows
- OpenGL in Pop

Doing Windows Graphics

- Typically:
  - do some preparation work
  - draw some graphics
  - do some clean-up
- Example:

```
COLORREF bubblecolor;
int intcenterx, intcentery, intradius;
CBrush cbrush, *pbrush_old;
cbrush.CreateSolidBrush(bubblecolor); //prepare
pbrush_old = pDC->SelectObject(&cbrush);
pDC->Ellipse(intcenterx - intradius, intcentery - intradius, intcenterx +
    intradius, intcentery + intradius); //draw
pDC->SelectObject(pbrush_old); //clean-up
cbrush.DeleteObject();
```

CDC

- Handle to a device context
- Has one primary member an HDC
- Each CDC has six tools which inherit from GDIObject: CPen, CBrush, CBitmap, CPallette, CFont, CRegion.
- When a CDC is created, it comes with six default objects
- Need to exchange these for ones one wants to use.
- Don’t delete tools on a CDC
- Delete any tools one creates after one is done with them

More CDC

- Do graphics calls within CView::OnDraw(CDC *pDC);
- One exception is if use Memory Device contexts.
- If somewhere else in CView you need to get a device context use GetDC() and when done ReleaseDC(CDC *pDC);
- Once have context follow same kinds of steps in first example

Persistent Display

- Want displays that stay the same under window resizing.
- Want displays that stay the same when covered and then uncovered.

To do this need to understand how the OnDraw method is invoked.

The OnDraw method

- When CView is created by a File| New or Window|New call, the constructor followed by OnCreate and then OnDraw are called.
- When CView is resized then OnDraw is called
- Similarly , whenever the window is covered, OnDraw is called
- Lastly, if CView::Invalidate is called the OnDraw is called when no other messages on queue. Can use UpdateView after Invalidate to make things happen faster

More on Persistent displays

- Two common approaches.
  - Have a bitmap of your scene. Draw bitmap with BitBlt or StretchBlt each time OnDraw called.
  - Have a display list consisting of the locations and what should be displayed at them. Cycle through this list drawing object each time OnDraw called.
- Can also mix approaches. Pop has a class cMemoryDC which holds a bitmap and can be added to the display lists of things to draw

Converting to Pixel Positions

- Need to be able to translate the floating point CPoint’s and cVector’s we are using to actual points on the screen as int’s
- Closest thing available in GDI are the functions SetMapMode and SetViewport (OpenGL has a glViewport function).
- Pop has its own class for handling this called cRealPixelConverter.

cRealPixelConverter Examples

```
cRealConverter _rpconverter; /*should have for each
   view */
_rpconverter.setRealWindow(lox,loy,hix,hiy);
// set size of world
//Override CView’s on size method
void MyCView::Onsize(UINT nType, int cx, int cy)
{
    CView::OnSize(nType, cx, cy);
    _rpconverter.setPixelWindow(cx,cy); //sets view size
}
```

More Pixel Conversion

- CPopView’s OnSize calls _pgraphics->setViewport(cx,cy); which calls cRealPixelConverter::setPixelConverter in the Windows case. Also, CPopView::OnSize calls pviewpointcritter()->setAspect((Real)cx,(Real)cy);
- Once the converter has been set up the methods:

realToPixel and pixelToReal can be called to do conversion.

Memory-based Device Contexts

- Pop uses the class cMemoryDC, a subclass of CDC, to hold bitmaps and as offscreen buffers.
- CPopView has a _cMemDC field which is a cMemoryDC used for a backbuffer.
- A scene is drawn here and then copyTo is invoked to write to actual window CDC.
- This avoids flicker.

OpenGL

- OpenGL is a graphics library
- Pop uses it to do 3D graphics
- To set up a Windows program so that it can use this library:
  - Add opengl32.lib and glu32.lib to the project
  - Use the following #includes:

    ```
    #include “gl/gl.h”
    #include “gl/glu.h”
    ```
- OpenGL functions typically begin with gl, glu, wgl. gl -- core library, glu -- OpenGL Utilities, wgl -- Windows extensions

The OpenGL State Machine

- Can think of OpenGL as a finite state machine.
- OpenGL keeps track of what is the current color it is drawing with, what is the current matrix it is using, etc.
- After setting up a scene for drawing we use glFinish() or glFlush() to get it to draw.

Sample OpenGL Fragment

```
//Init Window
glClearColor(0.0, 0.0, 0.0, 0.0);
glClear(GL_COLOR_BUFFER_BIT);
glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0); //set clipping region.
glBegin(GL_POLYGON);
    glVertexf(.25, .25, 0.0);
    glVertexf(.75, .25, 0.0);
   …
glEnd();
glFinish();
```

OpenGL Code in Windows

- OpenGL works under X-windows, Windows, and Mac.
- The Win32 extensions include a few built-in data types and functions which can be used with OpenGL. For example: ChoosePixelFormat, SelectPixelFormat, wglCreateContext and SwapBuffers.

Sample Windows Fragment

```
PIXELFORMATDESCRIPTOR pixelformat;
int pixelformat_index;
HGLRC openglrenderingcontext;
pixelformatindex = ChoosePixelFormat(hdc, &pixelformat);
SelectPixelFormat(hdc, pixelformatindex, &pixelformat);
openglrenderingcontext = wglCreateContext(hdc_view);
wglMakeCurrent(hdc_view, openglrenderingcontext); /*done also in
    cGraphicsOpenGL::activate()*/
//draw something
glFinish();
SwapBuffers(hdc_view);
```

OpenGL in Pop

- When OpenGL is being used CPopView::OnDraw(CDC *pDC) makes a bunch of calls to _pgraphics which in turn do OpenGL calls.
- For example, _pgraphics->activate() calls wglMakeCurrent(_pdc->getSafeHdc(), _hRC);
- The graphics background is cleared using:

  _pgraphics->clear(targetrect);

  which calls glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
- Setting up the view and projection matrices are then done with:

  _pviewpointcritter->loadViewMatrix();

  _pviewpointcritter->loadProjectionMatrix();

  //…

More OpenGL in Pop

These in turn call:

glMatrixMode(GL_MODELVIEW);

glLoadMatrix(_pviewpointcritter->attitude().inverse());

glMatrixMode(GL_PROJECTION)

gluPerspective(fieldofviewangledegrees, xtoyaspectratio, nearzclip, farzclip);

Drawing the game world calls:

pgame()->drawCritters(_pgraphics, _drawflags);

This generates a bunch of gl and glu calls. For instance in the case of Polygons:

glEnableClientState(GL_VERTEX_ARRAY);

glVertexPointer(…);

glDrawArrays(…);

Yet more OpenGL in Pop

Finally, graphics are drawn with:

```
_pgraphics->display(this, pDC);
```

which calls:

```
glFinish();
SwapBuffers(_pDC->getSafeHDC());
```