Subject: What is the minimum OpenGL version required for GLCtrl?
Posted by Tom1 on Thu, 29 Aug 2019 12:21:46 GMT
View Forum Message <> Reply to Message

Hi,

Does anybody know what is the lowest OpenGL version that supports running software based on GLCtrl?

I'm currently working on a 3D viewer based on GLCtrl and decided to test it on my older laptop (running Windows 10 1903 though) to check for adequate performance with older computers too. It turned out that it did not run at all. My app simply closed without any explanations. It's not very complicated app from OpenGL point of view; just some colored or textured 3D surfaces with simple lighting.

Thanks and best regards,

Tom

EDIT (twice): I know it works on OpenGL 2.1.2, 3.1, 3.3, 4.3, 4.4 and 4.5, which I have on other computers here.

Subject: Re: What is the minimum OpenGL version required for GLCtrl?
Posted by mirek on Fri, 30 Aug 2019 06:12:29 GMT
View Forum Message <> Reply to Message

GLCtrl is not bound to minimal version.

That said, IME I have found GL code itself quite tricky - some drivers tolerate hidden violations of GL contract in your GL code, others do not.

Have you tried with reference/OpenGL ?

Mirek

Subject: Re: What is the minimum OpenGL version required for GLCtrl?
Posted by Tom1 on Fri, 30 Aug 2019 07:26:50 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Yesterday evening I switched over to testing with reference/OpenGL and I'm having the same experience: Crash right in start.

I started to look around and found that commenting out these two lines in MakeWGLContext() in

Win32GLCtrl.cpp would prevent the crash and give me in exchange an incredibly slow OpenGL context:

```
    if(!wglChoosePixelFormatARB(hDC, attr, NULL, 1, &s_pixelFormatID, &numFormats))
     return;
```

The context reports OpenGL version 1.1.0, so I guess that's the Windows software implementation of OpenGL then.

According to Intel, the "Mobile Intel(R) 4 Series Express Chipset" I have in this old laptop supports OpenGL 2.0 and then some further extensions:

https://software.intel.com/en-us/articles/opengl-extensions-supported-in-intel-4-series-express-chipsets-and-beyond

In case the hardware does not support the current wglChoosePixelFormatARB() implementation, as is the case with my old laptop, could we have a clean fallback to a more basic OpenGL context in GLCtrl? In this case e.g. MSAA and double buffering could be dropped to maximize compatibility with old hardware. Of course it would be nice to have any OpenGL platform give its best possible performance, but I guess that's too much work to check every supported feature separately.

Most importantly, we should avoid the crash when OpenGL does not fill the requirements of GLCtrl.

Best regards,

Tom

Subject: Re: What is the minimum OpenGL version required for GLCtrl?
Posted by Tom1 on Fri, 30 Aug 2019 08:06:51 GMT

View Forum Message <> Reply to Message

Hi,

More on the subject: In my case it crashes in here:

```
if(!wglChoosePixelFormatARB(hDC, attr, NULL, 1, &s_pixelFormatID, &numFormats))
```

Tracked this down with RLOGs, since when attempting to run in debugger, the entire system crashed with BSOD!

Best regards,

Tom

Subject: Re: What is the minimum OpenGL version required for GLCtrl?
Posted by Tom1 on Fri, 30 Aug 2019 10:46:57 GMT
View Forum Message <> Reply to Message

Hi Mirek,

I think I have solved the issue. Now MakeWGLContext() cleanly skips the second pass employing wglChoosePixelFormatARB (simultaneously disabling MSAA and possibly also double buffering) if an OpenGL version below 2.1 is detected. Not sure though, if this version limit is correct/optimal, but at least I have successfully used wglChoosePixelFormatARB with OpenGL version 2.1.2.

Here's the revised code:

```
void MakeWGLContext(int depthBits, int stencilBits, int samples)
{
 ONCELOCK {
  for(int pass = 0; pass < 2; pass++) {
   HWND hWND = CreateWindow("UPP-CLASS-A", "Fake Window",
                         WS_CAPTION|WS_SYSMENU|WS_CLIPSIBLINGS|WS_CLIPCHILDREN,
                         0, 0, 1, 1, NULL, NULL,
                         NULL, NULL);
   if(!hWND)
    return;
   HDC hDC = ::GetDC(hWND);
   if(!hDC)
    return;
   memset(&s_pfd, 0, sizeof(s_pfd));
   if(pass == 0) {
    s_pfd.nSize = sizeof(s_pfd);
    s_pfd.nVersion = 1;
    s_pfd.dwFlags =
PFD_DRAW_TO_WINDOW|PFD_SUPPORT_OPENGL|PFD_GENERIC_ACCELERATED|PFD_
GENERIC_FORMAT|PFD_DOUBLEBUFFER_DONTCARE;
    s_pfd.iPixelType = PFD_TYPE_RGBA;
    s_pfd.cColorBits = 32;
    s_pfd.cAlphaBits = 8;
    s_pfd.cDepthBits = 24;
    s_pfd.cStencilBits = 8;
    s_pfd.iLayerType = PFD_MAIN_PLANE;
    s_pixelFormatID = ChoosePixelFormat(hDC, &s_pfd);
   }
   else {
    Vector<int> attr;
    attr
     << WGL_DRAW_TO_WINDOW_ARB << GL_TRUE
     << WGL_SUPPORT_OPENGL_ARB << GL_TRUE
     << WGL_DOUBLE_BUFFER_ARB << GL_TRUE
     << WGL_PIXEL_TYPE_ARB << WGL_TYPE_RGBA_ARB
     << WGL_ACCELERATION_ARB << WGL_FULL_ACCELERATION_ARB
     << WGL_COLOR_BITS_ARB << 32
     << WGL_ALPHA_BITS_ARB << 8
     << WGL_DEPTH_BITS_ARB << depthBits
     << WGL_STENCIL_BITS_ARB << stencilBits
    ;
    if(samples > 1)
     attr
      << WGL_SAMPLE_BUFFERS_ARB << GL_TRUE
      << WGL_SAMPLES_ARB << samples
     ;
    attr << 0;
    UINT numFormats;
    if(!wglChoosePixelFormatARB(hDC, attr, NULL, 1, &s_pixelFormatID, &numFormats))
     return;
   }

   DescribePixelFormat(hDC, s_pixelFormatID, sizeof(PIXELFORMATDESCRIPTOR), &s_pfd);
   if(!SetPixelFormat(hDC, s_pixelFormatID, &s_pfd))
    return;

   s_openGLContext = wglCreateContext(hDC);

   bool enhanced_mode=false;

   if(pass == 0) {
    HGLRC hRC = wglCreateContext(hDC);
    wglMakeCurrent(hDC, s_openGLContext);
    glewInit();

       if (glewIsSupported("GL_VERSION_2_1")) enhanced_mode=true;

    wglMakeCurrent(NULL, NULL);
   }

      ReleaseDC(hWND, hDC);
      DestroyWindow(hWND);

   if(!enhanced_mode) break; // In basic mode, this is it.
  }
 }
}
```

Reference/OpenGL now runs OK with Windows software OpenGL version 1.1.0 and also on HW accelerated OpenGL version 4.5. On 1.1.0 MSAA and DoubleBuffering are silently ignored even when they are enabled in GLCtrl.

Can you check if this is OK from your point of view and possibly commit?

Best regards,

Tom

Subject: Re: What is the minimum OpenGL version required for GLCtrl?
Posted by mirek on Fri, 30 Aug 2019 16:26:01 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Fri, 30 August 2019 12:46Hi Mirek,

I think I have solved the issue. Now MakeWGLContext() cleanly skips the second pass employing wglChoosePixelFormatARB (simultaneously disabling MSAA and possibly also double buffering) if an OpenGL version below 2.1 is detected. Not sure though, if this version limit is correct/optimal, but at least I have successfully used wglChoosePixelFormatARB with OpenGL version 2.1.2.

Here's the revised code:

```
void MakeWGLContext(int depthBits, int stencilBits, int samples)
{
 ONCELOCK {
  for(int pass = 0; pass < 2; pass++) {
   HWND hWND = CreateWindow("UPP-CLASS-A", "Fake Window",
                            WS_CAPTION|WS_SYSMENU|WS_CLIPSIBLINGS|WS_CLIPCHILDREN,
                            0, 0, 1, 1, NULL, NULL,
                            NULL, NULL);
   if(!hWND)
    return;
   HDC hDC = ::GetDC(hWND);
   if(!hDC)
    return;
   memset(&s_pfd, 0, sizeof(s_pfd));
   if(pass == 0) {
    s_pfd.nSize = sizeof(s_pfd);
    s_pfd.nVersion = 1;
    s_pfd.dwFlags =
PFD_DRAW_TO_WINDOW|PFD_SUPPORT_OPENGL|PFD_GENERIC_ACCELERATED|PFD_GENERIC_FORMAT|PFD_DOUBLEBUFFER_DONTCARE;
    s_pfd.iPixelType = PFD_TYPE_RGBA;
    s_pfd.cColorBits = 32;
    s_pfd.cAlphaBits = 8;
    s_pfd.cDepthBits = 24;
    s_pfd.cStencilBits = 8;
    s_pfd.iLayerType = PFD_MAIN_PLANE;
    s_pixelFormatID = ChoosePixelFormat(hDC, &s_pfd);
   }
   else {
    Vector<int> attr;
    attr
     << WGL_DRAW_TO_WINDOW_ARB << GL_TRUE
     << WGL_SUPPORT_OPENGL_ARB << GL_TRUE
     << WGL_DOUBLE_BUFFER_ARB << GL_TRUE
     << WGL_PIXEL_TYPE_ARB << WGL_TYPE_RGBA_ARB
     << WGL_ACCELERATION_ARB << WGL_FULL_ACCELERATION_ARB
     << WGL_COLOR_BITS_ARB << 32
     << WGL_ALPHA_BITS_ARB << 8
     << WGL_DEPTH_BITS_ARB << depthBits
     << WGL_STENCIL_BITS_ARB << stencilBits
    ;
    if(samples > 1)
     attr
      << WGL_SAMPLE_BUFFERS_ARB << GL_TRUE
      << WGL_SAMPLES_ARB << samples
     ;
    attr << 0;
    UINT numFormats;
    if(!wglChoosePixelFormatARB(hDC, attr, NULL, 1, &s_pixelFormatID, &numFormats))
     return;
   }

   DescribePixelFormat(hDC, s_pixelFormatID, sizeof(PIXELFORMATDESCRIPTOR), &s_pfd);
   if(!SetPixelFormat(hDC, s_pixelFormatID, &s_pfd))
    return;

   s_openGLContext = wglCreateContext(hDC);

   bool enhanced_mode=false;

   if(pass == 0) {
    HGLRC hRC = wglCreateContext(hDC);
    wglMakeCurrent(hDC, s_openGLContext);
    glewInit();

       if (glewIsSupported("GL_VERSION_2_1")) enhanced_mode=true;

    wglMakeCurrent(NULL, NULL);
   }

      ReleaseDC(hWND, hDC);
      DestroyWindow(hWND);

   if(!enhanced_mode) break; // In basic mode, this is it.
  }
 }
}
```

Reference/OpenGL now runs OK with Windows software OpenGL version 1.1.0 and also on HW accelerated OpenGL version 4.5. On 1.1.0 MSAA and DoubleBuffering are silently ignored even when they are enabled in GLCtrl.

Can you check if this is OK from your point of view and possibly commit?

Best regards,

Tom

Commited, thank you.