
Subject: [GICtrl] Integration and use of Glad loader instead of Glew

Posted by [Xemuth](#) on Wed, 08 Jul 2020 12:37:03 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hello,

Glew have not been updated since 2017 and people on reddit and stackoverflow seems more rely on Glad (wich is another openGL loader/provider).

From here I propose to set up a Glad package in order to replace Glew.

(With glad we wont need to include glew anymore)

What you guys think about it ?

Best regard, Xemuth

Subject: Re: [GICtrl] Integration and use of Glad loader instead of Glew

Posted by [Xemuth](#) on Wed, 08 Jul 2020 14:07:44 GMT

[View Forum Message](#) <> [Reply to Message](#)

If you want to have a look at how the package looks : <https://github.com/Xemuth/UPP-glad>

I have try it with OpenGL example (from reference) by replacing every include glew had by a single `#include <glad/glad.h>` and it worked well (on windows)

I also changed the `MakeWGLContext` function :

```
void MakeWGLContext(int depthBits, int stencilBits, int samples)
{
    /*...*/
    if(pass == 0) {
        HGLRC hRC = wglCreateContext(hDC);
        wglMakeCurrent(hDC, s_openGLContext);
        if(!gladLoadGL()){
            RLOG("Failed to load all OpenGL functions");
            exit(-1);
        }
        if(!gladLoadWGL(hDC)){
            RLOG("Failed to load Wiggle API");
            exit(-1);
        }
        if ("GLAD_GL_VERSION_2_1" enhanced_mode=true;
        wglMakeCurrent(NULL, NULL);
    }
    /*...*/
}
```

instead of :

```

void MakeWGLContext(int depthBits, int stencilBits, int samples)
{
    /*...*/
    if(pass == 0) {
        HGLRC hRC = wglCreateContext(hDC);
        wglMakeCurrent(hDC, s_openGLContext);
        glewInit();
        if (glewIsSupported("GL_VERSION_2_1")) enhanced_mode=true;

        wglMakeCurrent(NULL, NULL);
    }
    /*...*/
}

```

I don't have made any change for X11 and GTK yet.

If you are interesse in change I can provide a patch file

Subject: Re: [GICtrl] Integration and use of Glad loader instead of Glew

Posted by [Klugier](#) on Sat, 29 Aug 2020 11:40:10 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hello Xemuth,

It seems that GLEW will be soon updated to version 2.2.0 (Documentation updates for GLEW 2.2.0 release). So, the problem with outdated package will not be true. Bazaar should be as much compatible with uppsrc as possible. Introducing new library that do exactly the same is not very responsible decision. It is always add the need of maintenance, which is costly. You need to spend time to update the library and test if new version doesn't break compatibility. We should focus on one selected.

If we think that GLEW is bad then we should start discussion about moving towards GLAD for whole Upp. What things should be done in order to migrate to new library? Do I need to rewrite my app that loads OpenGL extensions?

I would like to also ask you about Vulkan. Do you plan any work in this area? It seems that this API is "OpenGL Next"? On the other hand do you consider to creating tutorial for learning 3D graphics with U++. I think something like "Learn 3D Graphics with C++ and Ultimate++". I think it would perfect opportunity to increase our ranges.

Sincerely,
Klugier

Subject: Re: [GLCtrl] Integration and use of Glad loader instead of Glew
Posted by [Xemuth](#) on Sat, 29 Aug 2020 15:47:56 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hello Klugier,

Klugier wrote on Sat, 29 August 2020 13:40

If we think that GLEW is bad then we should start discussion about moving towards GLAD for whole Upp.

GLEW is not bad, purpose of both lib is to provide wrapper around all new OpenGL Function / Features. So in practice Glad/Glew do the same work, the problem originally was the glew version in Upp was sync with OpenGL2 2.0(I'm not sure)(today we are at OpenGL 4.6) so many function and facilities introduced with new version are not visible. If GLew get an update then we could just keep it.

Klugier wrote on Sat, 29 August 2020 13:40

What things should be done in order to migrate to new library? Do I need to rewrite my app that loads OpenGL extensions?

From my point of view, switching from Glew to Glad mean modification in GLCtrl, the way OpenGL is loaded (via Glew or Glad). All the code around OpenGL function is unchanged. A proof of this come from my SurfaceCtrl package(in Bazaar) which is just an inheritance of GLCtrl but instead of loading OpenGL with Glew it load it with Glad (That's why my GLCtrl is called "GLCtrl_glad"), today if Glew was up to date, I could easily swap my inheritance from GLCtrl_glad to GLCtrl (which use glew) and all the package would work the same.

Klugier wrote on Sat, 29 August 2020 13:40

I would like to also ask you about Vulkan. Do you plan any work in this area? It seems that this API is "OpenGL Next"?

Indeed Vulkan should replace OpenGL in future. It is provided by Khronos Groupe (the same company which own OpenGL). It can be way more efficient in terme of performance however it's harded to utilize. One of my objective is to master it. However before learning it I first must know OpenGL deeply (both api have a similare way of working in many area)

Klugier wrote on Sat, 29 August 2020 13:40

On the other hand do you consider to creating tutorial for learning 3D graphics with U++. I think something like "Learn 3D Graphics with C++ and Ultimate++". I think it would perfect opportunity to increase our ranges.

I would be really happy to provide some tutorial on How to do 3D graphics / basic games?(why not) with Ultimate++ (using U++ framework + package like SurfaceCtrl). And I think like you, this kind of initiative could bring some new members which want to learn C++ and OpenGL. However, before starting to do some article on the subject I first need a solide package in terme of OpenGL technique and possibility. (SurfaceCtrl package is on a good way and is introduction should come soon on the forum !)

Subject: Re: [GICtrl] Integration and use of Glad loader instead of Glew

Posted by [koldo](#) on Mon, 31 Aug 2020 05:48:35 GMT

[View Forum Message](#) <> [Reply to Message](#)

I completely agree with Xemuth, and support him in his effort to create an useful control to show 3D geometries.
