
Subject: [Proposal] Adding a GLLock struct to GLCtrl
Posted by [Xemuth](#) on Wed, 02 Sep 2020 16:55:14 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hello, with the new version of Glew I moved my application to GLCtrl. However, in many part of my application, I need to get the OpenGL context to perform many actions. The actual way of executing something in the good OpenGL Context is to call `ExecuteGL(Event<> paint, bool swap_buffers)`. Which force me to surround my code around this function. It work but not convenient and for some part it block me and force me to change my architecture.

To allow more flexibility in the way we work with GLCtrl I propose to add a structure to GLCtrl named GLLock, which can be invoqued to get the OpenGL context and release it when the object got destroyed :

```
//class GLCtrl : public Ctrl{
    public:
    struct GLLock{
    private:
        HWND hwnd;
        HDC hdc;
    public:
        GLLock(GLCtrl& ctrl);
        ~GLLock();
    };
//};
```

Win32GLCtrl.cpp :

```
GLCtrl::GLLock::GLLock(GLCtrl& ctrl){
    hwnd = ctrl.pane.GetHWND();
    GLCtrl::CreateContext();
    hdc = GetDC(hwnd);
    wglMakeCurrent(hdc, s_openGLContext);
}
```

```
GLCtrl::GLLock::~~GLLock(){
    wglMakeCurrent(NULL, NULL);
    ReleaseDC(hwnd, hdc);
}
```

XGLCtrl.cpp:

```
GLCtrl::GLLock::GLLock(GLCtrl& ctrl){
    glXMakeCurrent(s_Display, ctrl.win, s_GLXContext);
}
```

```
GLCtrl::GLLock::~~GLLock(){
    glXMakeCurrent(s_Display, None, NULL);
}
```

Here is a trivial exemple of how to use it :

```
bool LoadSTL(){ //This function is used when a button is pressed
    try {
        GLCtrl::GLLock ____(canvas); //Canvas is the GLCtrl object
        //Shaders Stuff
        //OpenGL buffer filling and loading
        //Definition of draw
        return true;
    } catch (Exc e) {
        Exclamation(DeQt(f(e));
    }
    return false;
}
```

File Attachments

- 1) [GLCtrl.h.patch](#), downloaded 230 times
- 2) [Win32GLCtrl.cpp.patch](#), downloaded 202 times
- 3) [XGLCtrl.cpp.patch](#), downloaded 202 times

Subject: Re: [Proposal] Adding a GLLock struct to GLCtrl

Posted by [Klugier](#) on Wed, 02 Sep 2020 22:37:34 GMT

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

Hello Xemuth,

From my point of view the idea of "locking" looks reasonable. However, we should make it more smarter. Please noticed that the API we commit will be with us for the very long time. So, we need to be careful here. All we need to do is make context current and release it at the end of the block. So, I would suggest adding three dedicated methods to context attaching/activating GLCtrl and buffer swapping:

- ActivateContext(); (Anyway void ActivateContext(); from Win32 GLPane seems unimplemented)
- DeactivateContext();
- SwapBuffers();

So, in your case the logic will look as follow:

```
canvas.ActivateContext();
```

```
//Shaders Stuff
//OpenGL buffer filling and loading

canvas.SwapBuffers();
canvas.DeactivateContext();
// All these method will be proxy and send work to GLPane's classes.

return true;
```

In case of simplification we should add lock mechanisms as you suggested (The three above method should still be available, so you will have a choice):

```
GLCtrl::ContextLock ____(canvas); // <- If this is nested glass GL prefix is redundant.

// Do we always want buffer swpaing if not I would see additional lock...
GLCtrl::ContextLockWithSwapBuffers ____(canvas); // <- Alternatively ContextLock can have
additional bool parameter (true by default),
// however according to the clean code bool parameter should be
avoided, so I
// use more verbose notation here.
```

We also need documentation for GLCtrl and it seems that Copying file is missing too. Mirek can we add the second file?

Klugier

Subject: Re: [Proposal] Adding a GLLock struct to GLCtrl
 Posted by [Xemuth](#) on Thu, 03 Sep 2020 00:30:57 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hello Klugier, You are right about adding method in addition to struct:

Here is the code I got:
 GLCtrl.h

```
//Class GLCtrl : public Ctrl{
void ActivateContext();
void SwapBuffer();
void DesactivateContext();

struct ContextLock {
private:
  GLCtrl& parent;
```

```

public:
    ContextLock(GLCtrl& ctrl);
    ~ContextLock();
};
struct ContextLockWithSwapBuffers{
private:
    GLCtrl& parent;
public:
    ContextLockWithSwapBuffers(GLCtrl& ctrl);
    ~ContextLockWithSwapBuffers();
};
//};

```

Win32GLCtrl.cpp :

```

void GLCtrl::ActivateContext(){
    HWND hwnd = pane.GetHWND();
    GLCtrl::CreateContext();
    HDC hDC = GetDC(hwnd);
    wglMakeCurrent(hDC, s_openGLContext);
}
void GLCtrl::SwapBuffer(){
    HWND hwnd = GetHWND();
    HDC hDC = GetDC(hwnd);

    SwapBuffers(hDC);
}
void GLCtrl::DeactivateContext(){
    HWND hwnd = GetHWND();
    HDC hDC = GetDC(hwnd);

    wglMakeCurrent(NULL, NULL);
    ReleaseDC(hwnd, hDC);
}
GLCtrl::ContextLock::ContextLock(GLCtrl& ctrl) : parent(ctrl){
    HWND hwnd = parent.pane.GetHWND();
    GLCtrl::CreateContext();
    HDC hDC = GetDC(hwnd);
    wglMakeCurrent(hDC, s_openGLContext);
}
GLCtrl::ContextLock::~ContextLock(){
    HWND hwnd = parent.pane.GetHWND();
    HDC hDC = GetDC(hwnd);
    wglMakeCurrent(NULL, NULL);
    ReleaseDC(hwnd, hDC);
}
GLCtrl::ContextLockWithSwapBuffers::ContextLockWithSwapBuffers(GLCtrl& ctrl) : parent(ctrl){

```

```

HWND hwnd = parent.pane.GetHWND();
GLCtrl::CreateContext();
HDC hDC = GetDC(hwnd);
wglMakeCurrent(hDC, s_openGLContext);
}
GLCtrl::ContextLockWithSwapBuffers::~ContextLockWithSwapBuffers(){
HWND hwnd = parent.pane.GetHWND();
HDC hDC = GetDC(hwnd);

SwapBuffers(hDC);
wglMakeCurrent(NULL, NULL);
ReleaseDC(hwnd, hDC);
}

XGLCtrl.cpp :
void GLCtrl::ActivateContext(){
glXMakeCurrent(s_Display, win, s_GLXContext);
}
void GLCtrl::SwapBuffer(){
glXSwapBuffers(s_Display, win);
}
void GLCtrl::DeactivateContext(){
glXMakeCurrent(s_Display, None, NULL);
}
GLCtrl::ContextLock::ContextLock(GLCtrl& ctrl) : parent(ctrl){
glXMakeCurrent(s_Display, parent.win, s_GLXContext);
}
GLCtrl::ContextLock::~ContextLock(){
glXMakeCurrent(s_Display, None, NULL);
}
GLCtrl::ContextLockWithSwapBuffers::ContextLockWithSwapBuffers(GLCtrl& ctrl) : parent(ctrl){
glXMakeCurrent(s_Display, parent.win, s_GLXContext);
}
GLCtrl::ContextLockWithSwapBuffers::~ContextLockWithSwapBuffers(){
glXSwapBuffers(s_Display, parent.win);
glXMakeCurrent(s_Display, None, NULL);
}

```

File Attachments

- 1) [GLCtrl.h.patch](#), downloaded 198 times
 - 2) [XGLCtrl.cpp.patch](#), downloaded 201 times
 - 3) [Win32GLCtrl.cpp.patch](#), downloaded 212 times
-
-

Subject: Re: [Proposal] Adding a GLLock struct to GLCtrl

Posted by [mirek](#) on Thu, 03 Sep 2020 07:51:18 GMT

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

Xemuth wrote on Wed, 02 September 2020 18:55Hello, with the new version of Glew I moved my application to GLCtrl.

However, in many part of my application, I need to get the OpenGL context to perform many actions. The actual way of executing something in the good OpenGL Context is to call `ExecuteGL(Event<> paint, bool swap_buffers)`. Which force me to surround my code around this function. It work but not convenient and for some part it block me and force me to change my architecture.

Here is a trivial exemple of how to use it :

```
bool LoadSTL(){ //This function is used when a button is pressed
try {
    GLCtrl::GLLock ____(canvas); //Canvas is the GLCtrl object
    //Shaders Stuff
    //OpenGL buffer filling and loading
    //Definition of draw
    return true;
} catch (Exc e) {
    Exclamation(DeQt(f(e)));
}
return false;
}
```

I am not against, but in the example you have posted this does not really feel like improvement...

```
bool LoadSTL(){ //This function is used when a button is pressed
try {
    canvas.ExecuteGL([&] {
        //OpenGL buffer filling and loading
        //Definition of draw
    });
    return true;
} catch (Exc e) {
    Exclamation(DeQt(f(e)));
}
return false;
}
```

Subject: Re: [Proposal] Adding a GLLock struct to GLCtrl

Posted by [Xemuth](#) on Thu, 03 Sep 2020 15:21:45 GMT

You are right my example do not really feel like improvement... however this struct is not a huge improvement at all, it allow me to make simpler some portion of my code without surrounding it with anonymous function passed to ExecuteGL function.

At some place in my package, I still use ExecuteGL() because only a little portion of code need OpenGL Context however, a lot of function of my package requiere the context and I feel like it is more esthetique and simpler to simply call a function or invoke/Destroy a lock to get or release the context.

Subject: Re: [Proposal] Adding a GLLock struct to GLCtrl
Posted by [Klugier](#) on Thu, 03 Sep 2020 21:46:23 GMT

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

Hello Xemuth and Mirek,

We should think about the drawback of this addition:

- maintenance (current implementation delivered by Xemuth can be simplify, however still the implementation will need to be provided when adding new platform).
- concurrent solution (one thing can be done using two different approach - my solution is better than yours (code review problem))

Drawbacks:

- no need to open new additional layer of indentation
- more...

Backing to the "new layer of indentation", you could always call new function (which will make original one smaller):

```
bool LoadSTL(){ //This function is used when a button is pressed
    bool rendered;
    canvas.ExecuteGL([&] { rendered = renderSTL(); });
    return rendered;

    // Maybe with template magic we could simplify to
    // return canvas.ExecuteGL([&] -> bool { return renderSTL(); });
}

bool rednerSTL() {
    // All methods calls from here could call to OpenGL... The same is true with lock
    approach...
    try {
        //Context is here...
        //Shaders Stuff
        //OpenGL buffer filling and loading
        //Definition of draw
```

```
} catch (Exc e) {  
    Exclamation(DeQtf(e));  
}  
return false;  
}
```

I do not have strong opinion and the solution seems to have more drawbacks than pluses. Would be nice investigating, how we can extend ExecuteGL to return custom types.

Klugier

Subject: Re: [Proposal] Adding a GLLock struct to GLCtrl
Posted by [Xemuth](#) on Fri, 04 Sep 2020 14:34:37 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hello Klugier,

Klugier wrote on Thu, 03 September 2020 23:46

We should think about the drawback of this addition:

- maintenance (current implementation delivered by Xemuth can be simplify, however still the implementation will need to be provided when adding new platform).

Since my proposal simply call used function in ExecuteGL() I don't fairly see how it could grow up the maintenance needed on the package

Klugier wrote on Thu, 03 September 2020 23:46

- concurrent solution (one thing can be done using two different approach - my solution is better than yours (code review problem))

Both solution have the same purpose : Giving the OpenGL Context to allow GL Commands, The difference come from the timelive of the context, ExecuteGL() Release it when reached the end of function when the struct / function allow you to decide when releasing it. I think (maybe it's naive) both solutions complements each other and it is the developer job to choice the well suited one for is need at the moment. I see this choice like allocating on heap or stack depending on what you need, both allow you to reserve space for a data however the difference between both are huges and it is the developer job to chose the well suited one at a time.

Maybe my point of view is biased. What you think ?

Subject: Re: [Proposal] Adding a GLLock struct to GLCtrl
Posted by [Klugier](#) on Fri, 04 Sep 2020 20:28:32 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hello Xemuth,

Quote: Since my proposal simply call used function in `ExecuteGL()` I don't fairly see how it could grow up the maintenance needed on the package

Every new line of code requires some kind of maintenance for example you need to test if it's working after modifying some code nearby. This generates costs. The question here is do we really need them? I know that risk is very small, but it could happen. In software engineering everything is possible. It could be automated for example with unit tests then you shouldn't care.

Quote:

Both solution have the same purpose : Giving the OpenGL Context to allow GL Commands, The difference come from the timelive of the context, `ExecuteGL()` Release it when reached the end of function when the struct / function allow you to decide when releasing it. I think (maybe it's naive) both solutions complements each other and it is the developer job to choice the well suited one for is need at the moment. I see this choice like allocating on heap or stack depending on what you need, both allow you to reserve space for a data however the difference between both are huges and it is the developer job to chose the well suited one at a time.

There are several schools there. We could give people more freedom or restrict them to use only one approach. Both ways have it's pluses and minuses. From my long-term experience more freedom leads to unnecessary discussions. How many times I had situation like this:

- Hay I implemented that that way?
- Hey do you know that you can implement it that way and it is much better than your current approach?
- I am not sure my approach is fine.
- Hey read this article about my approach they said that it is better
- I found this article and it says mine is better
- ...

With one approach the above discussion will never happen. Until no one decides to modify framework to gave people choice ;) So as you noticed not everything is black and white.

Quote:

I see this choice like allocating on heap or stack

Hey, let's create garbage collector and forget about the choice. Now everything is allocated on stack ;)

The final decision is up to Mirek as a owner and maintainer of OpenGL package. I am here for constructive discussion :)

Klugier

Subject: Re: [Proposal] Adding a `GLLock` struct to `GLCtrl`

Posted by [Xemuth](#) on Sat, 05 Sep 2020 13:39:52 GMT

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

I guess you are right. I will let Mirek decide.

Subject: Re: [Proposal] Adding a GLLock struct to GLCtrl

Posted by [koldo](#) on Wed, 09 Sep 2020 06:15:03 GMT

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

It is nice to do the complex things easy.

GuiLock __; does it pretty well.

GLLock __(canvas); can do it as well.

Clean and simple.

Subject: Re: [Proposal] Adding a GLLock struct to GLCtrl

Posted by [koldo](#) on Wed, 09 Sep 2020 18:41:08 GMT

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

There are other options...

but really, they are less clean.

File Attachments

1) [Sin título.png](#) , downloaded 423 times

Subject: Re: [Proposal] Adding a GLLock struct to GLCtrl

Posted by [Klugier](#) on Wed, 09 Sep 2020 19:25:25 GMT

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

Hello Koldo, Mirek and Xemuth,

Koldo, you convinced me :) If two members of the community wants that functionality and probably more. I opt to add it also - with the version that will have the lowest maintenance cost.

And I think version without swapping should be:

GLLock __(canvas, false);

Without the need to call three methods/functions instead.

I do not have strong opinion about GLCtrl::GLLock - maybe simple GLLock within Upp namespace is better. Less typing at least.

So, Mirek we are waiting for final decision. We know that you follow this topic :)

Klugier

Subject: Re: [Proposal] Adding a GLLock struct to GLCtrl

Posted by [Xemuth](#) on Thu, 10 Sep 2020 14:52:00 GMT

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

Hello Klugier !

I then propose this integration :

```
struct GLLock{
private:
    GLCtrl& ctrl;
    bool swap;
public:
    GLLock(GLCtrl& glCtrl, bool swapBuffer = false);
    ~GLLock();
};
```

```
class GLCtrl : public Ctrl {
    typedef GLCtrl CLASSNAME;
    friend class GLLock;
//*****
//}
```

with for Win32GLCtrl.cpp :

```
GLLock::GLLock(GLCtrl& glCtrl, bool swapBuffer) : ctrl(glCtrl), swap(swapBuffer)
{
    HWND hwnd = ctrl.pane.GetHWND();
    HDC hDC = GetDC(hwnd);
    wglMakeCurrent(hDC, s_openGLContext);
}
```

```
GLLock::~~GLLock()
{
    if(swap){
        HWND hwnd = ctrl.pane.GetHWND();
        HDC hDC = GetDC(hwnd);
        SwapBuffers(hDC);
    }else{
        glFlush();
    }
```

```

}
wglMakeCurrent(NULL, NULL);
}

and for XGLCtrl.cpp :
GLLock::GLLock(GLCtrl& glCtrl, bool swapBuffer) : ctrl(glCtrl), swap(swapBuffer)
{
    glXMakeCurrent(s_Display, ctrl.win, s_GLXContext);
}

GLLock::~GLLock()
{
    if(swap){
        glXSwapBuffers(s_Display, win);
    }else{
        glFlush();
    }
    glXMakeCurrent(s_Display, None, NULL);
}

```

Here is diff file :

File Attachments

- 1) [GLCtrl.h.patch](#), downloaded 189 times
 - 2) [Win32GLCtrl.cpp.patch](#), downloaded 192 times
 - 3) [XGLCtrl.cpp.patch](#), downloaded 203 times
-