Subject: UppGL
Posted by unodgs on Tue, 17 May 2011 09:42:07 GMT
View Forum Message <> Reply to Message

Hi!

Yesterday I created a new branch called uppgl that is ultimate++ that uses opengl as a main painting engine. This is a part of my current project in which I needed animated interface with some cool effects. First I wanted to create completely separate ui engine but over the years I got used to u++ so much I didn't want to resign from it. Besides positioning system in upp is very powerful and easy to use at the same time. So I added another system drawer.
You can try it by creating assembly that points to "branches\uppgl;branches\uppgl\uppsrc" and running GuiTest. If everything will go fine you should see gui controls rendered by opengl. This demo can be for now run only in windows, but my plan is to make it working under linux as well, but later

Right now I'm using OpenGL in version 1.0 in immediate mode but to my surprise it is quite efficient if it comes to ui rendering. My goal is to switch to OpenGL 2.0 ES so it could work almost everywhere.
Not everything is working:
- popup windows are not automatically closed (I had problems with calling Deactivate method)
- there is only one font implemented for now - tahoma 14 (normal and bold)
- all themes work except host one. Unfortunately it is impossible to cache painting elements that are dynamically rendered by a system
- child windows are not implemented yet

Anyway this is only a start. I wanted to share in hope that someone else than me could make it better and would help to develop missing parts.

Of course there there are many places to improve:
- fonts could be dynamically generated (there is wglUseFontBitmaps but it works only under windows)
- all ui textures could be gathered into one texture. For now many small textures are generated.
- intermediate mode should be replaced with vbo's and shaders

In the project you can see plugins/glew package which is automatically initialized on application startup so you can use every extension possible.

During implementation I came to the conclusion that CtrlCore should be reorganized. It's quite hard to add another system draw kind of class. Right now depending on compiler flag SystemDraw is a win32 draw or x11 draw. It would be better if there was SystemDraw a an interface (like Draw is now) and X11SystemDraw, Win32SystemDraw so I could easily add OpenGLSystemDraw. That could also let to have Win32SystemDraw and OpenGLSystemDraw at the same time and use them interchangeably. OpenGl for gui rendering, Win32 one for pdf generating or printing or anything else that OpenGL draw is not good for. Right now I have BaseDraw that is old system draw and SystemDraw that uses OpenGL. Akward but had no time to implement it better and I wanted to hear your (especialy Mirek's) opinion on that first.

That's all for now. Please tell what you think about it.

PS: Of course it would be better to have hardware accelerated Painter and to render everything through it, but current solution seemed easier to me.

Subject: Re: UppGL
Posted by Tom1 on Tue, 17 May 2011 11:05:32 GMT
View Forum Message <> Reply to Message

Hi Uno,

This is a very interesting topic for me. I agree with you on most every item you represent. Especially my favorite subject: The hardware accelerated Painter.

Anyway, one issue comes to mind though: As far as I know, Windows supports off-the-shelf only OpenGL 1.0 and will not include anything else by default. Although other implementations are available, I'm worried that many end-user issues may arise because of lacking support for OpenGL 2.0 ES.

I hope I can get a little time to try out your efforts on this one.

Best regards,

Tom

ONE MORE DETAIL: Have you looked at OpenVG in order to get hardware accelerated 2D graphics?

Subject: Re: UppGL
Posted by mirek on Tue, 17 May 2011 16:42:48 GMT
View Forum Message <> Reply to Message

unodgs wrote on Tue, 17 May 2011 05:42
During implementation I came to the conclusion that CtrlCore should be reorganized. It's quite hard to add another system draw kind of class. Right now depending on compiler flag SystemDraw is a win32 draw or x11 draw. It would be better if there was SystemDraw a an interface (like Draw is now) and X11SystemDraw, Win32SystemDraw so I could easily add OpenGLSystemDraw. That could also let to have Win32SystemDraw and OpenGLSystemDraw at the same time and use them interchangeably. OpenGl for gui rendering, Win32 one for pdf generating or printing or anything else that OpenGL draw is not good for. Right now I have BaseDraw that is old system draw and SystemDraw that uses OpenGL. Akward but had no time to implement it better and I wanted to hear your (especialy Mirek's) opinion on that first.

Agreed. This is what "rainbow" is supposed to be, unfortunately the progress is slow and so far only "in my head". Well, at least I am now pretty confident what path to take... (it will be sort of ugly but effective approach using some macros).

Mirek

Subject: Re: UppGL
Posted by tojocky on Wed, 18 May 2011 11:15:59 GMT
View Forum Message <> Reply to Message

Seems to be very excited implementation.

Subject: Re: UppGL
Posted by cbpporter on Wed, 18 May 2011 12:45:42 GMT
View Forum Message <> Reply to Message

I actually have something similar for Irrlicht. Reimplementing the entire Draw core was not to my liking, so I actually wrote a new GUI very similar to U++. Some programing style and method names, but not really compatible. But it is very lightweight. It is designed for touchscreens and its conventions. Also, font support is very poor because it does not have TTF. But you can change the back end, using software rendering, OpenGl or DirectX. Only dependencies are Core and Irrlicht.

It is a very early version, but I may be able to compile some reference examples without modifying the logic in the near future.

On modern hardware it can do hundreds of frames/second.

But enough about me .

Any chance of your OpenGL implementation becoming a standard alternative back end?

Subject: Re: UppGL
Posted by koldo on Wed, 18 May 2011 14:00:32 GMT
View Forum Message <> Reply to Message

Nice efforts. When you have a demo please upload it .

Subject: Re: UppGL
Posted by mirek on Wed, 18 May 2011 16:46:21 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Wed, 18 May 2011 08:45I actually have something similar for Irrlicht. Reimplementing the entire Draw core was not to my liking, so I actually wrote a new GUI very similar to U++. Some programing style and method names, but not really compatible. But it is very lightweight. It is designed for touchscreens and its conventions. Also, font support is very poor because it does not have TTF. But you can change the back end, using software rendering, OpenGl or DirectX. Only dependencies are Core and Irrlicht.

It is a very early version, but I may be able to compile some reference examples without modifying the logic in the near future.

On modern hardware it can do hundreds of frames/second.

But enough about me .

Any chance of your OpenGL implementation becoming a standard alternative back end?

Yep. But we really need to make CtrlCore more friendly for it...

Subject: Re: UppGL
Posted by cbpporter on Thu, 19 May 2011 06:00:30 GMT
View Forum Message <> Reply to Message

On problem that I am having is the texture sizes: they need to be a power of two. If not you get an unsatisfying and unpredictable resize operation executed in the background in the best case scenario and horrible visual artifacts in the worst case scenario.

Uno, did you encounter this problem? Right now I am solving it by small pictures actually pointing to a large texture and containing the positional and size information and merging textures together in a texture pool when an image is loaded.

Subject: Re: UppGL
Posted by unodgs on Thu, 19 May 2011 06:23:10 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Thu, 19 May 2011 02:00On problem that I am having is the texture sizes: they need to be a power of two. If not you get an unsatisfying and unpredictable resize operation executed in the background in the best case scenario and horrible visual artifacts in the worst case scenario.

Uno, did you encounter this problem? Right now I am solving it by small pictures actually pointing to a large texture and containing the positional and size information and merging textures together in a texture pool when an image is loaded.
Like I wrote in my post, right now I have many textures of different sizes. But I want to rewrite texture manager to be more intelligent to be able to create one big texture (which is power of two)

and put as many as possible smaller textures there. Other big textures will be simply resized to the power of two with empty spaces on the right and the bottom. So I'm gonna solve it in exactly the same way you are I guess there is no better way to do it.

But that must wait a moment (it's not an issue on modern pc hardware). Right now I'm trying to use stencil buffer for nested transformed clipping.

Subject: Re: UppGL
Posted by cbpporter on Thu, 19 May 2011 06:29:12 GMT
View Forum Message <> Reply to Message

Not an issue on modern hardware? For me it is. I brought up the subject because my entire GUI is useless and horribly misshapen every time I forget the rule an create a texture of sizes not power of two and I forget to add them to the cache builder. So I was wondering if your experience is as dire as mine or only a small annoyance?

I merge textures into 512x512 textures. I am not supporting larger yet. In my experience it is slower to use non-square textures.

Subject: Re: UppGL
Posted by unodgs on Thu, 19 May 2011 08:20:27 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Thu, 19 May 2011 02:29Not an issue on modern hardware? For me it is. I brought up the subject because my entire GUI is useless and horribly misshapen every time I forget the rule an create a texture of sizes not power of two and I forget to add them to the cache builder. So I was wondering if your experience is as dire as mine or only a small annoyance?

I merge textures into 512x512 textures. I am not supporting larger yet. In my experience it is slower to use non-square textures.
According to this paper http://www.nvidia.com/dev_content/nvopenglspecs/GL_ARB_texture_non_power_of_two.txt support for textures of dimensions not being power of 2 started in 2004. Now we have 2011 so most of hardware available today should support this feature (I'm not talking about mobile gpus). Of course handling such textures might be slower but that's something I don't know. On my 8800GTX and RandeonHD 3650 everything runs fine and looks good.
Or maybe I don't get your point. Could you explain your problem with more details?

Subject: Re: UppGL
Posted by cbpporter on Thu, 19 May 2011 08:33:51 GMT
View Forum Message <> Reply to Message

I have Nvidia and you can't rely on that, or at least not while using Irrlicht, which supports multiple

back ends. Textures will get resized with the resulting relative size depending on the hardware it is run and on some hardware you will get ugly vertical artifacts in pink and orange colors.

This is why I am using my own padding and tiling system, but it is a little bit wasteful memory wise.

Anyway, I'll try to get some tutorial examples working and post some samples. But I doubt that my results will be compatible enough. I actually have all widgets in two versions, the main version using a straightforward getter/setter system without chaining and the extended class supporting only chaining. This allows you to use a Button specific method after a control specific method, like SetRect(...).SetText(), because the parent control class does not define the chained methods, only the extended class and they have as a return type the actual class.

Subject: Re: UppGL
Posted by raxvan on Thu, 19 May 2011 21:26:24 GMT
View Forum Message <> Reply to Message

I would suggest to stick to pow2 textures. You might never know when you want to use this on something that doesn't know other than 2^n textures and in the end you only end up refactoring source code. There is not a big problem if you have a 1K texture with black unused borders, the texture addressing has a relatively constant speed regardless of texture size (i'm referring to 2^n textures).
Also it is very important to use batching and no alpha blending if possible, otherwise you will kill the GPU.
In some games the gui rendering can take almost as much as the rendering of the scene.

Subject: Re: UppGL
Posted by nlneilson on Fri, 20 May 2011 05:04:11 GMT
View Forum Message <> Reply to Message

I have tiled many large geo-referenced .tif file sets.
This uses openGL, gdal, etc..

The lowest resolution level is often referred to as lztd (Level Zero Tile Degrees)
The pow2 doesn't make any difference as far as I am concerned.
That started real early as a concept that dividing by two made sense. Google Earth still uses it.

Another reason pow2 made any sense was with sets of images that covered a full globe like the Earth, Moon, etc. is so they matched at the poles and the dateline (+/- 180 degrees).
As long as the lztd divides into 360 evenly they match at the poles/dateline.

I tiled all the U.S. FAA Charts. The merged Sectional and WAC charts use 10 deg, the TAC at 1 deg, the separate Sectionals at 2 deg.
http://www.nlneilson.com/

Many of the NASA WorldWind Satellite image tiles are 36 deg.

GlobalMapper started with pow2, I suggested to Mike to change that so his app could be used with the NASA WW project which it does now.

Nawak coded dstile to merge and tile images.
http://worldwindcentral.com/wiki/Making_Layers

Just today I pulled down the source of what_nick's mod of Nawak's code for Windows.
http://whatnick.dyndns.org:8080/tisham/DsTileQtGUI.zip

Trying to compile in MSVC 10 (it was in 8) but get this error:
1>LINK : fatal error LNK1181: cannot open input file 'gdal_i.lib'
I have that linked to C:\FWTools2.4.7\lib and C:\OSGeo4W\lib

I would like to do that in Upp, any help/suggestions would be appreciated on getting the dstile code to work in theIde.

The BIGGEST difference in performance is the tiles should be in the .dds format so the GPU does not have to do it. I modified Chiss's code and it uses the DDSConverter.java from the latest NASA WWJ SDK, it's an executable .jar.
http://www.nlneilson.com/apps/jpg2dds.jar
or Chiss's
http://forum.worldwindcentral.com/attachment.php?attachmenti d=1594&d=1208570928

Any questions on this I will be glad to help if I can, UppGL is a good idea.
Maybe this can be combined with Mirek's "rainbow" concept.

Subject: Re: UppGL
Posted by raxvan on Fri, 20 May 2011 19:18:03 GMT
View Forum Message <> Reply to Message

nlneilson wrote on Fri, 20 May 2011 07:04
The pow2 doesn't make any difference as far as I am concerned.

Except when they are not loading. Some platforms will not allow you to have non power of two textures.

Subject: Re: UppGL
Posted by nlneilson on Fri, 20 May 2011 20:54:40 GMT
View Forum Message <> Reply to Message

raxvan wrote on Fri, 20 May 2011 12:181. Except when they are not loading.

2. Some platforms will not allow you to have non power of two textures.

1. Do you have a link to tests that have confirmed this??
On the same computer a file will get to the GPU in the same amount of time if the files are the same size in bytes. I thought I made it clear in my last post that it is the format (.jpg or .dds) that effects performance. .dds is much faster getting through the GPU and displayed. You can do a Google search for "dds v jpg"

2. I assume you meant application rather than platform which is usually used as OS, at least that is what I am used to.
If so you are correct. Some applications use pow2 to determine the lztd, like Google Earth and some do not. Some servers have a large image that are made up of many smaller images like the Satellite images and what is requested is tiled on the fly for whatever is requested.
In this case a request can be made for whatever lztd, pow2, 2 deg, 10 deg, 36 deg, whatever your age is divided by pi or ???, it is just numbers.

Also note that GE references from the top left and many others from the bottom left. Some have a pixel size of 256x256, some 512x256 and some 512x512.

The fastest (and with the fewest problems) is if the images are all pre-tiled so any time for the server to tile them is eliminated, that is how I have all the FAA Charts. In this case the lztd must be what they were pre-tiled with.
These are used on a true 3D app with the images draped over an ellipsoidal globe (WGS84) with elevation data (DEM).

Just to make it clear so there is less chance of a misunderstanding the lztd or lowest resolution layer can be whatever one chooses (except it should divide evenly into 360 if it crosses the date line.
For every level the size in degrees is one half of the size of the level with one step lower resolution regardless of the lztd chosen.
This is referred to as the LOD or Level Of Detail.
GE uses 180 deg as level zero.
For my merged sets I used 10 deg.
For my zero level (LOD 10) the tile size is 10 degrees.
GE's size near that would be LOD 4 at 11.25 degrees.
The tile naming is a different but basically it's the lat and lon from the reference point divided by the tile size for that level. Mentally with 10 deg the locations lat and lon can be determined. With 11.25 deg it would be time to pull out the Abacus beads.

Download gdal2tiles.py and take a look at the formats that will make.

It's good you have an interest in tiling.

Subject: Re: UppGL

Once the lowest resolution tile size (lztd) has been chosen subsequent level sizes can be found by:

n = LOD; // from 0 to 17 with lztd = 180 deg
lztd DIVIDED BY 2**n // maybe your reference to pow2 ???

0 : 180.0
1 : 90.0
2 : 45.0
3 : 22.5
4 : 11.25
5 : 5.625
6 : 2.8125
7 : 1.40625
8 : 0.703125
9 : 0.3515625
10 : 0.17578125
11 : 0.087890625
12 : 0.0439453125
13 : 0.02197265625
14 : 0.010986328125
15 : 0.0054931640625
16 : 0.00274658203125
17 : 0.001373291015625

with an lztd of 1.0 which I chose for the TAC charts
0 : 1.0
1 : 0.5
2 : 0.25
3 : 0.125
4 : 0.0625
5 : 0.03125
6 : 0.015625
7 : 0.0078125
8 : 0.00390625
9 : 0.001953125
Note that with an lztd of 180.0 and the levels shown the number of decimal places is reaching the limits for a double.
With an lztd of 1.0 for the TAC charts only 4 levels (0 to 3) were needed to show the grain of the paper the FAA (NACO) scanned the images from.

NASA's WWJ cache structure is like this:
...\Earth\NASA\Marble\3\30\30_15.dds
the actual file name, 30_15.dds, the first integer relates to latitude and the second longitude.

I wrote a python app that takes two arguments, the lztd and file name (including the path) which gives the location (lat and lon) of the bottom left of that tile (file) with the reference at -90 lat and -180 lon.

The same concept is used for high resolution images like X-Rays.
In that case the lowest level of resolution would just be the integer 1 for level 0 (zero) or a size in mm, inch or ?? if scaled.

Maybe I can help if UppGL gets into tiling images.

Subject: Re: UppGL
Posted by raxvan on Sat, 21 May 2011 13:04:29 GMT
View Forum Message <> Reply to Message

Hello,

Well , we are a bit off topic here, but my initial suggestion was that unodgs in his implementation should stick to power of two texture because if he will ever plan to port the system to another platform he might run into problems. By platform i mean the actual device you are running on, and the driver associated with that might not let you to create non power of two textures.

nlneilson:
By "Except when they are not loading" i ment that you will get no performance at all, the system will not work at all if you can't load your textures, right?
And by platform i really ment the platform you are running on not the host application (OS as you call it),your PC,IPAD,IPOD,MAC,WII console,PS3,XBOX360 whatever that might be. The driver which is associated with the platform might not have support for non power of two textures.
Also google earh is a really basic example of MipMap ussage, there is no fancy technology there, no quantum mechanics or black magic. Here we were talking about 2D GUI rendering that has to be pixel perfect and the is no reasons to use mipmaps there for multiple reasons:
1.There is no need for zooming out from the GUI(as far as i know).
2.If you are using RenderToTexture(for clipping,masking, and performance boosts) you will have to make mipmaps by yourself which is not a nice process at runtime.
3.MipMapping works only on the minification filter meaning that when the texture is getting smaller and smaller the next mipmap is sellected.
4. Any scaled texture in a GUI system creates aliasing problems which are very hard to deal with on the GPU (not impossible but still... hard).

Subject: Re: UppGL
Posted by nlneilson on Sat, 21 May 2011 14:49:08 GMT
View Forum Message <> Reply to Message

It isn't that far off topic even as you mentioned since how the UppGl should cause the least amount of problems in the long run is important.

The pow2, however you intended it, for the level of detail it should be "pow2" as I showed in the previous post.

In the early days, covered wagons and such, the "pow2" was used to determine the lztd, so that would mean 5.625 rather than 5.0 and 1.40625 or 0.703125 instead of 1.0, seems a bit silly with the advantage of hindsight.

If images are in a format an application or "platform" is not compatible with of course they will not "load". An Etch-a-Sketch Model II, I don't know which model of Mac that would compare to, would have problems also.

This should first work with a regular computer with a full OS rather than an Android, cell phone, Dick Tracy Watch or some cereal box gismo.

As someone mentioned and you may have intended with pow2 and "should stick to power of two texture" is the image should be square and also it's important the size like 256x256 or 512x512. There was an example in Mercator projection with 512x256 which was real bad as far as performance but some still use it for 2D.

As far as zooming for gaming it may not be necessary but for 3D like GE or WWJ just seeing the whole Earth without zooming in would be pretty useless.

I must admit I have no experience in programming for games, nor intend to in the near future.

For that with your experience the input can be beneficial to UppGL or any project.