avl Library – Development Journal

# 10-31-11

I finally have a break in schoolwork – it’s been just over a month since I’ve had time to work on the library. Not surprisingly, I don’t remember where I was when I last worked on it. As such, I’m beginning this development journal to keep track of what I’ve been doing and what I need to do. School isn’t going to let up for the next two years, so I need a system to make these cold starts quicker and easier.

The last thing that I did was add processing for newline characters in the utility::TextBox component – it now properly formats for newline characters. I don’t believe that I tested this functionality very thoroughly at all, but I’ve decided to move on to something else instead. There are some major bugs which were identified in the Arkanoid Clone prototype:

* Exception thrown in debugger when program exits after focus has switched to a different window and then back to the application.
* Pressing ctrl+alt+delete makes the window client area go black, and it doesn’t recover.
* In fullscreen mode, the application is just a black screen. Input still works.

These are serious bugs – it’s lazy to just keep adding features instead of dealing with them. As such, that’s the current task. Once these bugs are dealt with, then I need to go back and clean up the code a bit. I know that the view::BasicRenderer::RenderSprites() code is jumbled and confusing; it definitely needs to be refactored into multiple functions.

* Identify the major areas where refactoring is necessary. First is view::BasicRenderer::RenderSprites().

## Step 1:

I need easier control over the creation of a window. I’m adding a LeastSquaredProfile() method to view::Direct3DRendererBase which will find the best-fitting display profile for the requested parameters.

# 11-3-11

So…I finished the LeastSquaredProfile() method mentioned in the previous entry, except that I haven’t decided what exception to throw when it fails. So I started designing a better, more comprehensive hierarchy of exceptions to use in the library. Partway through, I realized that I need a better documentation solution, and set up Doxygen. So now I’m:

1. Switching over to Doxygen style comments.
2. Finishing an exception hierarchy.
3. Finishing the LeastSquaredProfile() method.
4. Fixing bugs!!!

# 12-7-11

Fall quarter is over! I’ve continued converting all header-file comments over to Doxygen style comments, and I’ve been making use of dOxygen’s built-in To-Do list functionality to document areas that need refactoring and/or bug fixes. I’ve been refraining from making any significant changes to the code because I don’t want to introduce any new bugs.

# 12-27-11

I’ve finished several tasks since the last entry:

1. All of the headers have been converted over to dOxygen style comments. Source files haven’t been, as I don’t want to have to maintain consistency between headers and sources (copy/paste, anyone?).
2. The Component Builder has been completely rewritten to be far more flexible and extensible, and now outputs dOxygen-style source files. I’ve also added the Component Builder as an external tool within the Visual Studio IDE, so it is much more easy and efficient to add new components. Simply go to Tools->Component Builder, answer the two prompts appropriately, and it will create a new directory for the component files in the appropriate location, and then output the component files. Then I simply add the new files to the solution and begin working on them.
3. I’ve rewritten the main exceptions hierarchy to be more sensible, flexible, and to provide more useful debugging information. I’m in the process of converting all of the old throw statements to use the new hierarchy.
4. I rewrote most of the utility::SettingsFile component, as it previously had methods that ran on for hundreds of lines. It’s now much more manageable, understandable, and safer. I also got rid of the use of void pointers in the utility::SettingsFile::SettingValue class and instead opted for a union (so there’s no more casting).
5. The view::LeastSquaredProfile method is finished, but hasn’t yet been converted over to the new exceptions hierarchy.

I haven’t yet been able to do much bug fixing, as things turned out to be much more time consuming than I expected. I’ve also been playing a fair amount of games (but, to be fair, I truly was burned out after Fall quarter). The quality of this latest round of updates has been, in my opinion, much improved.

# 12-31-11

I again modified the utility::SettingsFile component to simply use two maps (one for integer values, and one for strings…much simpler). I’ve also touched up the exception hierarchy in the view component and then modified all of the projects to compile with the new exceptions. I also dramatically cleaned up the Git index by not tracking dOxygen documentation (which can be generated from the source code) and getting rid of other crap.

At this point I’m ready to continue major refactoring and bug fixing. I now plan on going through the To-Do list generated by dOxygen and fixing those items. This will mainly involve refactoring messy code and fixing major bugs.

# 1-4-12

I’ve refactored all of the overly complex methods in the view::D3DRendererBase class into smaller, less complex methods, but I haven’t yet done any testing of the changes. Additionally, there are a couple of quick bug fixes that I need to do: *constructors and destructors must never call virtual functions.* This needs to be changed in the D3DRendererBase class. Additionally, *all assignment operators should return a reference to \*this, including compound assignments.* I need to check to see if I need to make that change in either the Vertex2D or Sprite classes.

After the above are completed, the next step is to continue refactoring (BasicRenderer, and then the windowing code, and then the input component). After that, bug fixing and running down the To-Do list.