Subject: Groovey
Posted by dudymas on Sat, 11 Nov 2006 00:48:26 GMT
View Forum Message <> Reply to Message

So, as a sort of proof of concept, I'll be trying to code this guy:
http://code.google.com/p/gparticles/

And just so you know, I also posted this topic at Dream In Code forums too... but mostly just to see if they are interested at all. I will prolly use this topic as my 'usual' place of updates. Just so that if this turns out to be a success, I can look back and see a sort of journal and anyone else can do the same.

I think I might post the UML here if it seems like anyone is interested in watching the work happen and give me pointers or just laugh and discuss.

This is GOING to be hard. I know it. Crossplatform threading WITH openGL as well is going to have to be confusing... or at least it has to be because I'm not using java.

I have a macbook pro that's currently sick, though, so my main platform will not be tested first. I'll get it working on windows first.

Here's the particle models I want to get working in three months... they are given in the order I'll work on getting them done at:

Basic Gravity (A bunch of particles in a domain that bounce around happily. This will test two things: Threads don't starve or race. Particle model is at least working decently.)

Zombies (A bunch of people sit on the screen. Randomly spawn a single zombie. People have three behaviors: become a zombie if one touches you, run and panic if you see a zombie or another person panicking, and if everything is fine the people just walk around leisurely. Designed to test particle behavior system to make sure they have an efficient collision detection model and all that).

Whizz Bangers (Create a screen of self-attracting particles. They are given an overall whirlpool force that will continue to slosh them. As the particles begin to clump, set a critical mass size that causes them to behave like an atom bomb... eventually a bunch of clumps form and when they hit critical mass they explode violently. The cycle continues nonstop, because they just blow apart into the smaller beginning particles again. This will test both behaviors and models of force with both the exploding and the whirlpool. Finally, it will add a couple new features to test: can we get the particle driver to parse a config file with definitions of particles and can we make the particles change color and make smoke.)

And the rest of the particle sets I want to finish I'll keep secret for now... but I eventually want it so that anyone can make whatever group of particles they want and make them interact in complex or simple ways. This means I have to abstract each model (particles, behaviors, and models of forces) three ways:

What does the user see when they configure a particle system.

What does a computer see when it manipulates the system.
What does the graphics card see when it renders the system.

I'll be using differential equation solvers eventually... before you ask. And also some really funky random number generator foo and all.

So... the class structure is already looking rather funky. I mean... each time I look at it, it's like as if I suddenly hear "Get Funkayeeee, boyeeeeee!" in my head. Or maybe that's just me needing more caffeine.

Yay! Doesn't this sound like fun?

And after all this project is finished... I have some crazy graphics tests for the whole thing (transforming bodies for the particles so they can look gooey, throbbing, smoldering, gaseous, probabilistic, magical, etc.)

On a final note, here's what I have to say about UPP so far, since this is the first time I've ever touched it:
I like it better than devcpp so far. As well, I love how you guys do all these packaging tricks. My gripes so far:

I wanna use boost libraries (for threading, primarily)... I also want to get UPP on mac if I can. As much as I love xcode, it's huge, and I'd like something freely available on windows as well as mac so that anyone can pick up my code and use it as thoroughly as possible with as little intervention from me or anyone else as possible.

You have a wonderful thing going here. It's simple, light, and potent. And so help me, I love the ALT-C thingy

---

Subject: Re: Groovey
Posted by dudymas on Sat, 11 Nov 2006 22:39:48 GMT
View Forum Message <> Reply to Message

Oh, and also, I finally got over my laziness and updated the gparticles trunk. Currently, it'll compile to nothing. Have fun with that. I guess up next I'll consider putting the upp file on there, but for now... nah. It's nothing special and might be more of a problem to use than to create one from scratch.

Subject: Re: Groovey

Posted by dudymas on Sun, 12 Nov 2006 06:57:54 GMT
View Forum Message <> Reply to Message

Ah yes... got most of the code done for the gravity case. The subversion trunk should reflect that.... hopefully.

I'll go ahead and admit that I'm confused as to where to go from here. A lot of threading, basically. I need to make a render thread and a transform thread at the minimum. And I need to mutex the stuff so that I don't close the program at the wrong state and all.

That's the next fun stage.

Then I can compile it and let all you eager fans (hahahaah... I do mean that sarcastically... with a pinch of egotism) know how it goes.

Woo!

Subject: Re: Groovey
Posted by mirek on Sun, 12 Nov 2006 07:50:51 GMT
View Forum Message <> Reply to Message

Quote:
I'll go ahead and admit that I'm confused as to where to go from here. A lot of threading, basically. I need to make a render thread and a transform thread at the minimum. And I need to mutex the stuff so that I don't close the program at the wrong state and all.

Not sure whether you are actually using U++ as library, but mutexes, threads etc... and in Core/Mt.h and Core/Thread.h.

U++ GUI should run in single thread. See reference/GuiMT as an example of multithreaded GUI application.

Mirek

Subject: Re: Groovey
Posted by dudymas on Sun, 12 Nov 2006 22:09:48 GMT
View Forum Message <> Reply to Message

thanks! I didn't realize there was an example already in there on threads.

I want the framework to be multi-platform, however, so unless I'm reassured that the threads in u++ work on mac, linux, and windows reasonably, I'm going to have to opt out of using it

particularly. But, seeing as how I have no experience with threads yet, I could still definitely find a lot of help out of looking at the u++ code you mentioned there. I'll read through the example and see what insight it can give me.

I'll attach a look at the system I want to use... where one thread does either rendering (grabbing particles and drawing them to a buffer) or drawing (putting the buffer to the screen) and the other thread either manipulates particles (basically begins changing the particles) and updating the particles (committing the changes to the particles finally). Each thread can only switch between these two states, and yet I don't want to allow the system to update particles while they are being rendered, because that means that some of the particles will be changed before they are rendered, causing a loss in data or tearing on the window. And yet, the particles can be manipulated while there is rendering because none of the changes are committed to the particles yet... only logged.

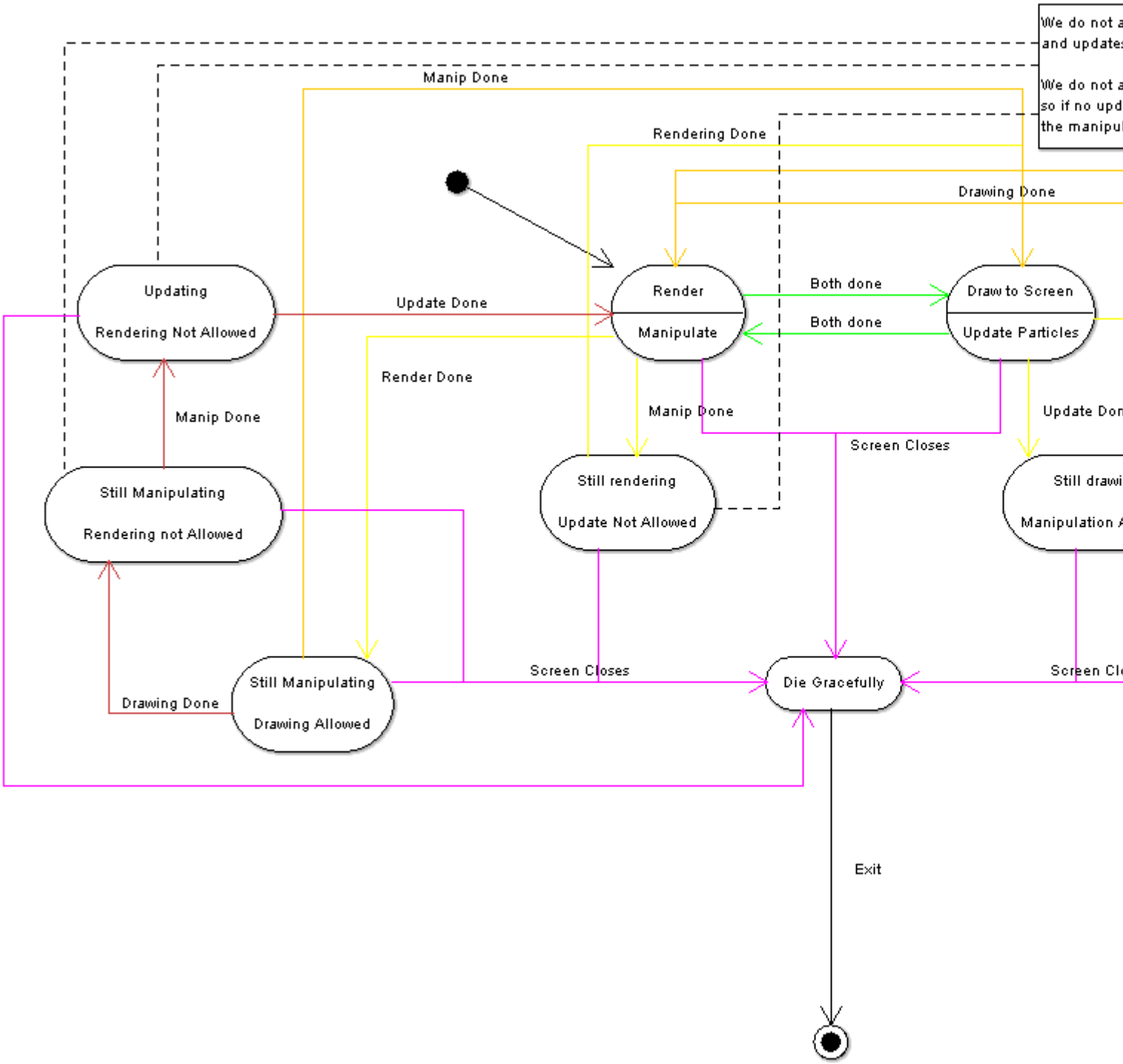
So this state diagram (I admit, it's horrid to look at) is supposed to help show this will all the possible states the two threads can be in. The green lines show best case, yellow is nominal, and orange is unwanted and unexpected but still needs to be considered. Reddish means it's a rare scenario, but definitely still possibility. Finally, the fuchsia/violet lines mean the program is closed at any of those states.

Man, this is getting more fun each second.

So here's my plan:
I'll give the renderer states that rule over how the manipulation/update thread governs its transitions. Hopefully the renderer is going to be slow enough that it won't bully everything. We'll see.

File Attachments

1) Grooveyactivity.gif, downloaded 2524 times



Subject: Ready to patch up
Posted by dudymas on Mon, 13 Nov 2006 02:24:42 GMT
View Forum Message <> Reply to Message

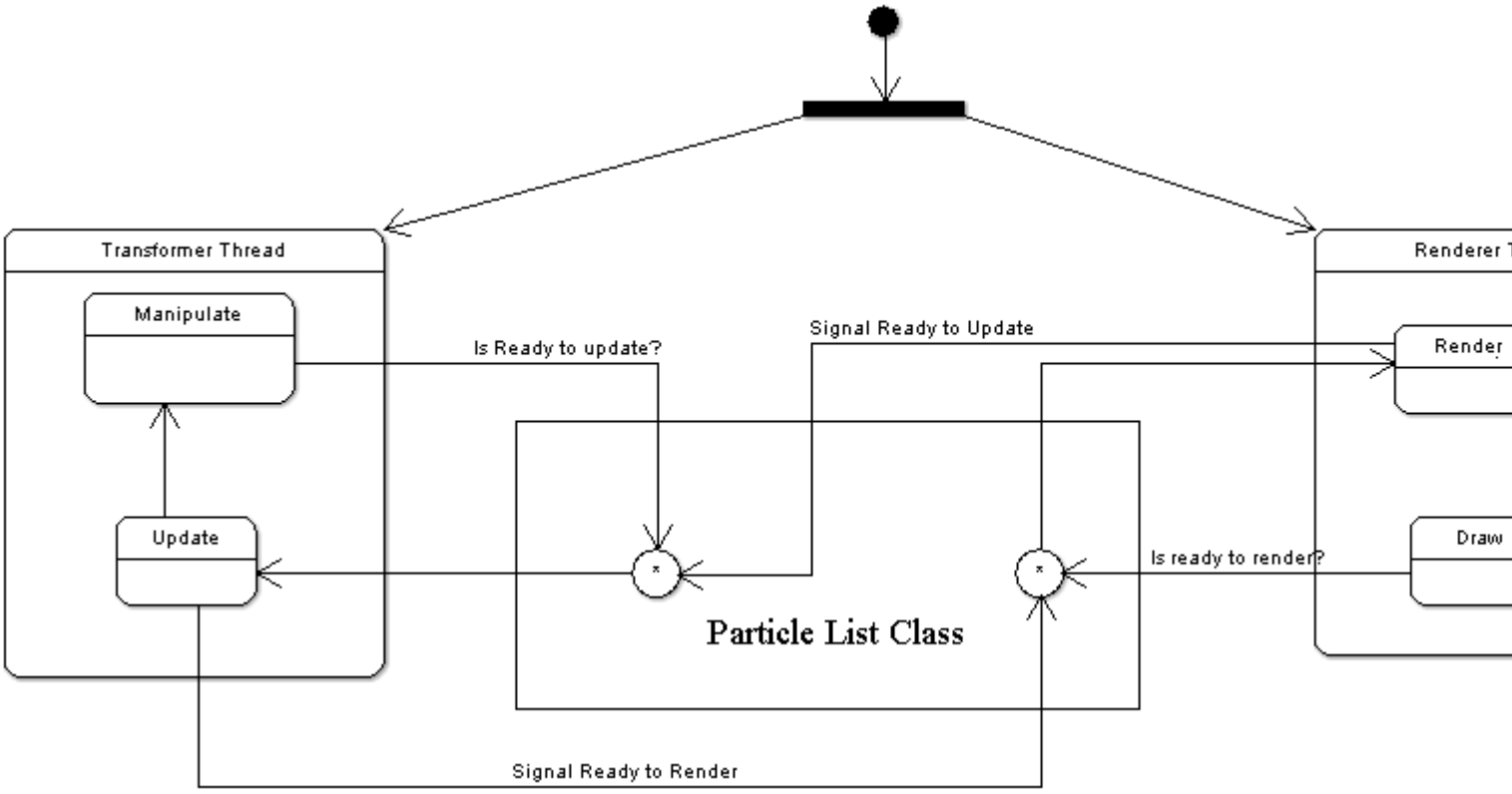
Okay, glut is HARD to wield. Seriously. So, I made some really messy code (I have to use a global declaration and all... and a few other public functions I didn't want to throw out there).

Here is the way I want the two threads to run for now (each circle with an asterisk is a different mutex, and they are controlled by conditionals. A particle list must not be updated before it is updated, and a particle list must not be rendered before it is rendered. Updating causes a particle list to not be rendered. Rendering causes a particle list to not be updated. Seems like that logic won't crack on me for now.

The attached diagram is a graphical way of viewing this (sorry for butchering UML so painfully... excuse me).

File Attachments

1) GrooveyStateMachine Control.gif, downloaded 2452 times



Subject: Re: Groovey
Posted by dudymas on Tue, 14 Nov 2006 02:19:42 GMT
View Forum Message <> Reply to Message

Oy!

I'm getting too many linker errors to continue using upp for the moment. Need to find a way to resolve those guys first... then I'll come back and give feedback.

More specifically, I thought it'd be easy to find a way to include the boost.threads libraries. It turns out quite the opposite. For some reason, the mingw that comes with upp doesn't want anything to do with compiling boost libraries. I'm sure there's a way around it, and I'm not at all trying to hate on upp. I'll find a resolution for this in my spare time.

For now, unless anyone else knows a quick solution, I'll get this running in devC++, and then sometime late get those libraries into upp, as I still prefer upp's charm.

More still to come, I'm know for sure.
