Graph rendering using 3D acceleration

Asked 16 years, 2 months ago Modified 16 years ago Viewed 3k times

8

We generate graphs for huge datasets. We are talking 4096 samples per second, and 10 minutes per graph. A simple calculation makes for 4096 * 60 * 10 = 2457600 samples per linegraph. Each sample is a double (8 bytes) precision FP. Furthermore, we render multiple linegraphs on one screen, up to about a hundred. This makes we render about 25M samples in a single screen. Using common sense and simple tricks, we can get this code performant using the CPU drawing this on a 2D canvas. Performant, that is the render times fall below one minute. ***As this is scientific data, we cannot omit any samples. Seriously, this is not an option. Do not even start thinking about it.***

Naturally, we want to improve render times using all techniques available. Multicore, pre-rendering, caching are all quite interesting but do not cut it. We want 30FPS rendering with these datasets at minimum, 60FPS preferred. We now this is an ambitious goal.

A natural way to offload graphics rendering is using the GPU of the system. GPU's are made to work with huge datasets and process them parrallel. Some simple

HelloWorld tests showed us a difference of day and night in rendering speed, using the GPU.

Now the problem is: GPU API's such as OpenGL, DirectX and XNA are made for 3D scenes in mind. Thus, using them to render 2D linegraphs is possible, but not ideal. In the proof of concepts we developed, we encountered that we need to transform the 2D world into a 3D world. Suddnely we have to work with and XYZ coordinate system with polygons, vertices and more of the goodness. That is far from ideal from a development perspective. Code gets unreadable, maintenance is a nightmare, and more issues boil up.

What would your suggestion or idea be to to this in 3D? Is the only way to do this to actually convert the two systems (2D coordinates versus 3D coordinates & entities)? Or is there a sleeker way to achieve this?

-Why is it usefull to render multiple samples on one pixel? Since it represents the dataset better. Say on one pixel, you have the values 2, 5 and 8. Due to some sample omitting algorithm, only the 5 is drawn. The line would only go to 5, and not to 8, hence the data is distorted. You could argue for the opposite too, but fact of the matter is that the first argument counts for the datasets we work with. This is exactly the reason why we cannot omit samples.

.net **3d** **gpu** **linegraph**

Share

Improve this question

Follow

edited Oct 21, 2008 at 2:14

asked Oct 20, 2008 at 20:52

user29688

328 ● 3 ● 11

12 Answers

Sorted by: Highest score (default)

6

I'd like to comment on your assertion that you cannot omit samples, on the back of tgamblin's answer.

You should think of the data that you're drawing to the screen as a sampling problem. You're talking about 2.4M points of data, and you're trying to draw that to a screen that is only a few thousand points across (at least I assuming that it is, since you're worried about 30fps refresh rates)

So that means that for every pixel in the x axis you're rendering in the order of 1000 points that you don't need to. Even if you do go down the path of utilising your gpu (eg. through the use of opengl) that is still a great deal of work that the gpu needs to do for lines that aren't going to be visible.

A technique that I have used for presenting sample data is to generate a set of data that is a subset of the whole set, just for rendering. For a given pixel in the x axis (ie. a given x axis screen coordinate) you need to render an

absolute maximum of 4 points - that is the minimum y, maximum y, leftmost y and rightmost y. That will render all of the information that can be usefully rendered. You can still see the minima and maxima, and you retain the relationship to the neighbouring pixels.

With this in mind, you can work out the number of samples that will fall into the same pixel in the x axis (think of them as data "bins"). Within a given bin, you can then determine the particular samples for maxima, minima etc.

To reiterate, this is only a subset that is used for display - and is only appropriate until the display parameters change. eg. if the user scrolls the graph or zooms, you need to recalculate the render subset.

You can do this if you are using opengl, but since opengl uses a normalised coordinate system (and you're interested in real world screen coordinates) you will have to work a little harder to accurately determine your data bins. This will be easier without using opengl, but then you don't get the full benefit of your graphics hardware.

Share Improve this answer Follow

answered Oct 21, 2008 at 22:36
Andrew Edgecombe
40.3k ● 3 ● 38 ● 63

> It is possible to derive a dataset from the original dataset for displaying purposes, so that the visual representation is the same as the original one. However, this process is calculation intensive too. Net gain of this process is only limited, and

does not add significant render time advantage.
– user29688 Oct 24, 2008 at 21:01

5

A really popular toolkit for scientific visualization is VTK, and I think it suits your needs:

1. It's a high-level API, so you won't have to use OpenGL (VTK is built on top of OpenGL). There are interfaces for C++, Python, Java, and Tcl. I think this would keep your codebase pretty clean.
2. You can import all kinds of datasets into VTK (there are tons of examples from medical imaging to financial data).
3. VTK is pretty fast, and you can distribute VTK graphics pipelines across multiple machines if you want to do very large visualizations.
4. Regarding:

   > This makes we render about 25M samples in a single screen.
   >
   > [...]
   >
   > As this is scientific data, we cannot omit any samples. Seriously, this is not an option. Do not even start thinking about it.

You can render large datasets in VTK by sampling and by using LOD models. That is, you'd have a model where you see a lower-resolution version from far out, but if you zoom in you would see a higher-resolution version. This is how a lot of large dataset rendering is done.

You don't need to eliminate points from your actual dataset, but you can surely incrementally refine it when the user zooms in. It does you no good to render 25 million points to a single screen when the user can't possibly process all that data. I would recommend that you take a look at both the VTK library and the VTK user guide, as there's some invaluable information in there on ways to visualize large datasets.

Share Improve this answer Follow

edited Oct 20, 2008 at 21:16

answered Oct 20, 2008 at 21:09

Todd Gamblin
59.7k ● 15 ● 91 ● 96

5

You really don't have to worry about the Z-axis if you don't want to. In OpenGL (for example), you can specify XY vertices (with implicit Z=0), turn of the zbuffer, use a non-projective projection-matrix, and hey presto you're in 2D.

Share Improve this answer Follow

answered Oct 20, 2008 at 20:58

Menkboy
1,635 ● 11 ● 12

I believe t "non-projective matrix" you are talking about is called an orthographic projection. – Matt Jun 25, 2010 at

18:43

3

Mark Bessey mentioned it, that you might lack the pixels to display the graph. But given your explanations, I assume you know what you are doing.

OpenGL has an orthogonal mode that has a z-coordinate inside (0;1). There is no perspective projection, the polygons you draw will be planar to the screen clipping area.

DirectX will have similar. On OpenGL, it's called gluOrtho2d().

Share Improve this answer Follow

answered Oct 20, 2008 at 21:04

mstrobl
2,391 ● 15 ● 16

2

OpenGL is happy to render 2D if you setup the projection to be Ortho (no z). Also you should decimate your data. Rendering the same pixel 1000 times is a waste of GPU. Spend your time upfront with a performat multi-thread decimator. The be sure to blast large arrays at the GPU using vertex arrays or vertex buffer objects (clearly I'm an OpenGL kinda of guy)

Share Improve this answer Follow

answered Oct 20, 2008 at 21:02

basszero
30k ● 9 ● 58 ● 80

1

> This makes we render about 25M samples in a single screen.

No you don't, not unless you've got a **really really large** screen. Given that the screen resolution is probably more like 1,000 - 2,000 pixels across, you really ought to consider decimating the data before you graph it. Graphing a hundred lines at 1,000 points per line probably won't be much of a problem, performance wise.

Share Improve this answer Follow

answered Oct 20, 2008 at 20:57

Mark Bessey
19.8k ● 4 ● 49 ● 69

1

If your code gets unreadable because you're dealing with the 3D stuff directly, you need to write a thin adaptor layer that encapsulates all the 3D OpenGL stuff, and takes 2D data in a form convenient to your application.

Forgive me if I've missed something, and am preaching basic object oriented design to the choir. Just sayin'...

Share Improve this answer Follow

answered Oct 20, 2008 at 21:15

comingstorm
26.1k ● 3 ● 46 ● 68

> You don't need to eliminate points from your actual dataset, but you can surely incrementally

1

> refine it when the user zooms in. It does you no good to render 25 million points to a single screen when the user can't possibly process all that data. I would recommend that you take a look at both the VTK library and the VTK user guide, as there's some invaluable information in there on ways to visualize large datasets.

Thank you very much. This is exactly what I was looking for. It seems VTK uses hardware to offload these kind of rendering, too. Btw, i guess you mean **valuable** ;). Second, the user does get information of the example i gave. However not really concise, the overview of the data can really be pure gold for the scientist. It is not about processing all the data for the user, it is about getting valuable information out of the rendering. Users seem to do this, even in the very 'zoomed out' representation of the dataset.

Any more suggestions?

Share Improve this answer Follow

answered Oct 20, 2008 at 21:17

user29688
328 ● 3 ● 11

No problem. No more suggestions off the top of my head. But I did mean invaluable (noun: valuable beyond estimation : priceless <providing invaluable assistance>). See merriam-webster.com/dictionary/invaluable – Todd Gamblin Oct 20, 2008 at 21:21

1

I wanted to point out that in addition to using VTK directly there are two other products built on VTK that may be of interest to you.

1) ParaView (paraview.org) is a user interface built on top of VTK that makes scientific visualization products much easier. You can render all the data you want provided you have the hardware to handle it, and it supports MPI for multiple processors / cores / clusters. It is extensible via user created plugins and uses automated tools for project building and compiling.

2) ParaViewGeo (paraviewgeo.mirarco.org) is a geology and mining exploration derivative of ParaView produced by the company I work for. It has built-in support for reading file formats that ParaView does not, such as Gocad, Datamine, Geosoft, SGems, and others. More importantly, we often do work with other groups who have an interest in scientific viz with a loosely-ties-to-mining deliverable, such as our recent work with a group doing Finite/Discrete element modelling. It might be worth checking out.

In both cases (PV and PVG) your data is considered separate from your view of that data, and as such, you will never "render" all of your data (since you wouldn't likely have a monitor large enough to do so) but rest assured it will all "be there" processed from your data set as you expected. If you run additional filters on your data, only what can be seen will be "rendered" but the filters will compute on ALL of your data, which although may not all be visible at once, will all exist in memory.

If you're looking for numbers, today I computed three regular grids of 8 million cells in PVG. One contained a 7-tuple vector property (7x 8 million double values), the other two each contained a scalar property (1x 8 million double values each) for a total of 72 million double values in memory. I believe the memory footprint was close to 500MB but I also had a 400,000 point set where each point had a 7-tuple vector property and some other miscellaneous data available as well.

Share Improve this answer Follow

answered Dec 9, 2008 at 3:17

user44484

0

Not sure if this is helpful, but could you use time as a dimenion? i.e. one frame is one z? That might make things clearer, perhaps? Then perhaps you could effectively be applying deltas to build up (i.e on z axis) the image?

Share Improve this answer Follow

answered Oct 20, 2008 at 20:57

Egwor
1,322 ● 9 ● 16

> No you don't, not unless you've got a really really large screen. Given that the screen resolution is

0

> probably more like 1,000 - 2,000 pixels across, you really ought to consider decimating the data before you graph it. Graphing a hundred lines at 1,000 points per line probably won't be much of a problem, performance wise.

First of all, we cannot omit any samples when rendering. This is impossible. This would mean the rendering is not accurate to the data the graph is based on. This really is a no-go area. Period.

Secondly, we **are** rendering all the samples. It might be that multiple samples end up on the same pixel. But still, we are rendering it. The sample data is converted on the screen. Thus, it is rendered. One can doubt the usefullness of this visualized data, byt scientists (our customers) are actually demanding it we do it this way. And they have a good point, IMHO.

Share Improve this answer Follow

answered Oct 20, 2008 at 21:03
user29688
328 ● 3 ● 11

You can use LOD techniques to get around this. Rendering multiple samples to the same pixel is useless unless you can zoom in and incrementally refine the data. – Todd Gamblin Oct 20, 2008 at 21:05

Well, I'm glad that you've accepted the suggestion to try out VTK, since it'll take care of the scaling/decimation for you. I do find it bewildering that you say it's important to render each data point, even when they end up on the same pixel. I feel there must be a communication breakdown here. – Mark Bessey Oct 21, 2008 at 1:06

I tried to explain this in my question (I just edited it). Really, my first repsonse working on this project was the same as yours. Working along, I understood that we cannot omit samples, and require to draw them all. – user29688 Oct 21, 2008 at 2:17

0

Wrap the library in a gentler, kinder 2D library with the Z and rotations all set to 0.

-Adam

Share Improve this answer Follow

answered Oct 20, 2008 at 21:04

Adam Davis
93.5k ● 60 ● 271 ● 333