paul.houx

Smooth thick lines using geometry shader

in Using Cinder • 8 years ago

Hi,

does anyone of you know a good solution to drawing thick line segments with proper end points? Right now, I simply create a GL_LINES mesh and draw it using the fixed function pipeline, but when I specify glLineWidth(8.0f), gaps appear at sharp corners. This is a common problem.

I was hoping one of you would know how to fill the gaps between two segments, either by pre-calculation during mesh generation, or by using a geometry shader. Any help is appreciated.

-Paul

Reply 1

Replies(38)

okinp

Re: Smooth thick lines using geometry shader
8 years ago

Couldn't a line segment be represented by a triangle strip? As I imagine it, as you move across the line, you find the normal to it and calculate a "left" and "right" point to it, based on the the thickness that you want.

For straight parts you would just need 4 points and for curved / "corner" parts you would need a higher "sampling".

-Nik.

Edit: I might be wrong, but If you want to use a shader, the tessellation shader is probably more appropriate than the geometry shader.

I33N

Re: Smooth thick lines using geometry shader
8 years ago

I remember using a geometry shader for that purpose.
The idea was to pass line_strip to the shader. For each original line you need to pass the previous and the next line as well. Then you output triangle strips by computing the normal and cutting the corners...
I'll try to find this shader later, I think it's on my external hard drive.

Ben

paul.houx

Reageren: Smooth thick lines using geometry shader
8 years ago

Thank you both for your replies.

@I33N: I would very much appreciate getting a look at your shader.

My current line of thought is to convert the line segments to a GL_TRIANGLES mesh on the CPU, using my own algorithm. I had hoped to find a ready-made piece of code somewhere, but wasn't successful. On paper, the algorithm is finished, I hope to implement it in the coming days. If it works, I will share it with you.

-Paul

safetydank

Re: Smooth thick lines using geometry shader

8 years ago

Something like this?

I33N

Re: Smooth thick lines using geometry shader

8 years ago

I just found the geometry shader used to draw prismoid using this technique but it's pretty much the same, you just have to remove the other faces... Got this code from the grasshopper.

Copy code

```
uniform mat4 ModelviewProjection;
varying in vec3 vPosition[4]; // Four inputs since we're using GL_LINE_STRIP_ADJACENCY
varying in vec3 vNormal[4];   // Orientation vectors are necessary for consistent alignment
vec4 prismoid[8]; // Scratch space for the eight corners of the prismoid

void emit(int a, int b, int c, int d)
{
    gl_Position = prismoid[a]; EmitVertex();
    gl_Position = prismoid[b]; EmitVertex();
    gl_Position = prismoid[c]; EmitVertex();
    gl_Position = prismoid[d]; EmitVertex();
    EndPrimitive();
}

void main()
{
    // Compute orientation vectors for the two connecting faces:
    vec3 p0, p1, p2, p3;
    p0 = vPosition[0]; p1 = vPosition[1];
    p2 = vPosition[2]; p3 = vPosition[3];
    vec3 n0 = normalize(p1-p0);
    vec3 n1 = normalize(p2-p1);
    vec3 n2 = normalize(p3-p2);
    vec3 u = normalize(n0+n1);
    vec3 v = normalize(n1+n2);

    // Declare scratch variables for basis vectors:
    vec3 i,j,k; float r = Radius;

    // Compute face 1 of 2:
```

```
    j = u; i = vNormal[1]; k = cross(i, j); i *= r; k *= r;
    prismoid[0] = ModelviewProjection * vec4(p1 + i + k, 1);
    prismoid[1] = ModelviewProjection * vec4(p1 + i - k, 1);
    prismoid[2] = ModelviewProjection * vec4(p1 - i - k, 1);
    prismoid[3] = ModelviewProjection * vec4(p1 - i + k, 1);

    // Compute face 2 of 2:
    j = v; i = vNormal[2]; k = cross(i, j); i *= r; k *= r;
    prismoid[4] = ModelviewProjection * vec4(p2 + i + k, 1);
    prismoid[5] = ModelviewProjection * vec4(p2 + i - k, 1);
    prismoid[6] = ModelviewProjection * vec4(p2 - i - k, 1);
    prismoid[7] = ModelviewProjection * vec4(p2 - i + k, 1);

    // Emit the six faces of the prismoid:
    emit(0,1,3,2); emit(5,4,6,7);
    emit(4,5,0,1); emit(3,2,7,6);
    emit(0,3,4,7); emit(2,1,6,5);
}
```

I was mistaking you have to pass line_strip_adjacency and not just line_strip.

Hope it helps!

Ben

paul.houx

Reageren: Smooth thick lines using geometry shader
8 years ago

@Safetydank: your link didn't work (the book isn't visible), but searching for "polyline quadstrips" gave me a bunch of new ideas on how to approach this problem. Thanks!

@Ben: also very helpful, I came across that shader before. It even produces 3D lines made of boxes, cool. However, I think with the knowledge I have now I can come up with a 2D version based on triangles myself. I prefer 2D lines over 3D ones for this project, and I also want to run this application on slightly older hardware that doesn't have geometry shader support (read: my computer at home).

I will keep you guys posted. Thanks for helping me out.

-Paul

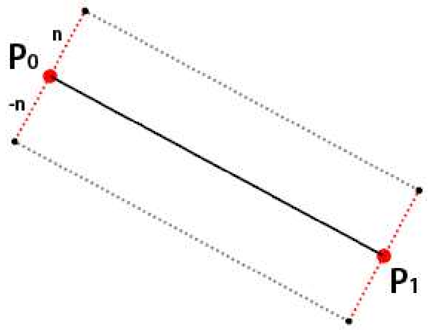
paul.houx

Reageren: Smooth thick lines using geometry shader
8 years ago

Hi,

I managed to come up with a solution of my own. It's basically a 2D version of the shader found by Ben, but with different characteristics for the miter (explained below).

Construction of the line starts with a segment:

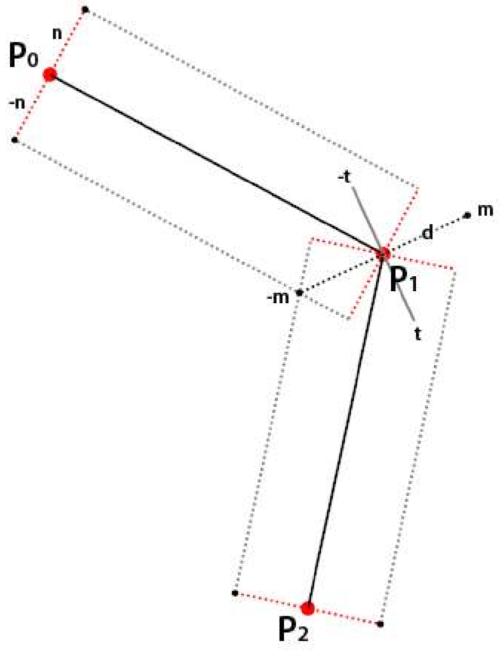


P0 and P1 are Vec2f's. The line is defined by `Vec2f line = p1 - p0`
and the normal (n) is defined by `Vec2f normal = Vec2f( -line.y, line.x).normalized().`

You could then draw the segment by multiplying the normal by the thickness and add or subtract it from the end points:

```
Vec2f a = p0 - thickness * normal;
Vec2f b = p0 + thickness * normal;
Vec2f c = p1 - thickness * normal;
Vec2f d = p1 + thickness * normal;
```

But when two segments are joined, we need to calculate the cross-section of the joint to form the miter:



First, I calculate the tangent (t) as follows:

```
Vec2f tangent = ( (p2-p1).normalized() + (p1-p0).normalized() ).normalized()
```

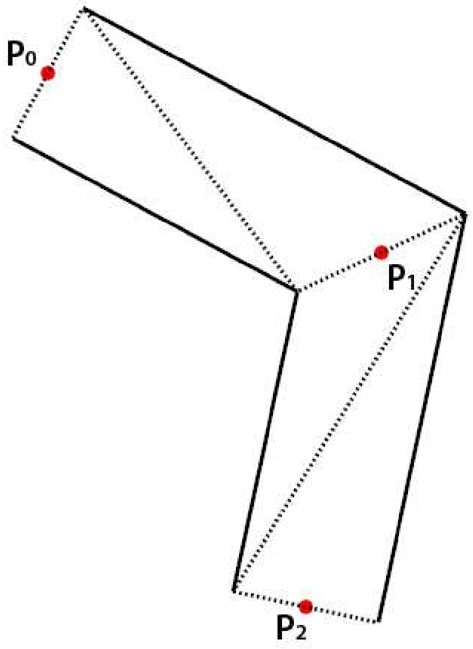
Next, I find the miter line (m), which is the normal of the tangent:

```
Vec2f miter = Vec2f( -tangent.y, tangent.x )
```

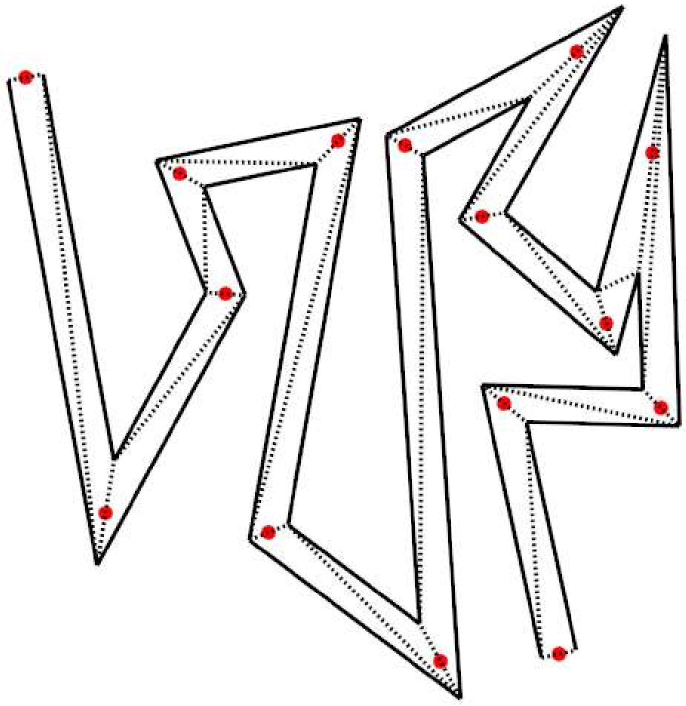
The correct length (d) of the miter can then be found by projecting it on one of the normals (shown in red) using the dotproduct. This gives us a value that is the inverse of the desired length, so:

```
float length = thickness / miter.dot( normal )
```

That's all the information we need. Now we can construct the triangle mesh:



The result is quite nice, even for sharp angles:



Note that this algorithm does not always give the desired result at very sharp angles or if the line segment is too short. This can be detected and taken care of, for example by converting the sharp angle to a blunt one by adding an extra point.

You can play with the algorithm in this sample:
http://www.cowlumbus.nl/forum/Polyline2TriMeshTest.zip

It can also be found on GitHub:
https://github.com/paulhoux/Cinder/commit/2d467c4ae9145c03bbaf014c0bceb7f806754cc8

I haven't written the TriMesh code yet, and expect some difficulties with texturing (all triangles have different sizes, requiring perspective correction). But I am hopeful I can soon draw awesome 2D textured lines :) Thanks everyone for helping me out.

-Paul

paul.houx

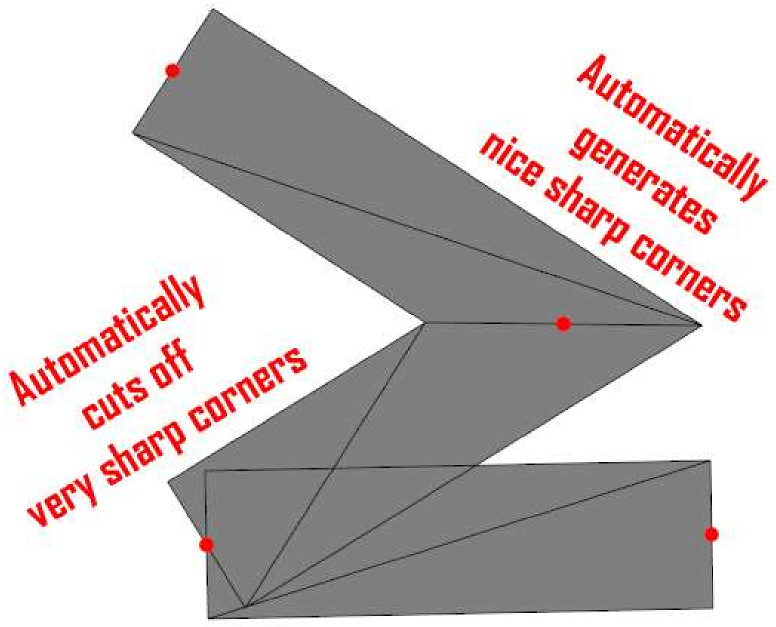
Reageren: Smooth thick lines using geometry shader
8 years ago

Hi,

I managed to write a geometry shader based on the discussion above. It accepts a GL_LINES_ADJACENCY_EXT mesh as input and outputs a smooth line without gaps.

It is slightly different from the explanation given earlier. Instead of calculating the tangent, I simply calculate the 3 normals (previous segment, current segment and next segment) and average 2 of them to find the miter.

Additional code checks for very sharp angles using a dotproduct and performs some naive culling to prevent visual glitches. Shader code can be found in the ZIP file (see end of this post).


Automatically generates nice sharp corners
Automatically cuts off very sharp corners

At the corners, it extends the line segments to create nice sharp corners that maintain the thickness and appearance of the line.

At very sharp corners, however, it will automatically fall back to normal OpenGL behavior to prevent visual glitches.

Using the shader, you can now (relatively) easily draw proper lines of any thickness. Texturing is not supported at the moment, but it should not be too hard to add it to the shader.

A geometry shader is the only way to go, because I wanted a fixed thickness in pixels. Pre-calculating the mesh in 3D would result in perspective distortion when using a 3D camera. If you would not use a shader, you'd have to recreate the mesh every time the camera moves or zooms and that would put quite a burden on the CPU.

It proved to be a bit tricky doing the conversion from 3D to 2D screen space in the shader. After many unfruitful attempts, I found out it didn't work because I was passing the window size as a `Vec2i` instead of a `Vec2f`. This caused the shader to simply not draw anything. So be careful when adopting this piece of code: make sure to properly cast `getWindowSize()` to a `Vec2f` when supplying it to the shader!

The sample and shader can be downloaded here:
http://www.cowlumbus.nl/forum/GeometryShader.zip

Edit: this sample has been updated. See below for more information.

Have fun with it,

-Paul

bantherew..

Re: Reageren: Smooth thick lines using geometry shader
8 years ago

Very cool exploration. Geometry shaders are nice to work with because you can easily write expressive code while getting much-better-than-CPU performance. Not sure about using them for a more fundamental task like this, though.

I've found that creating lines the same way you create ribbons with GL_TRIANGLES (recommended) or GL_QUADS accomplishes this task. Because the points in each segment are shared with the next, there are no open corners.

http://www.BanTheRewind.com

paul.houx

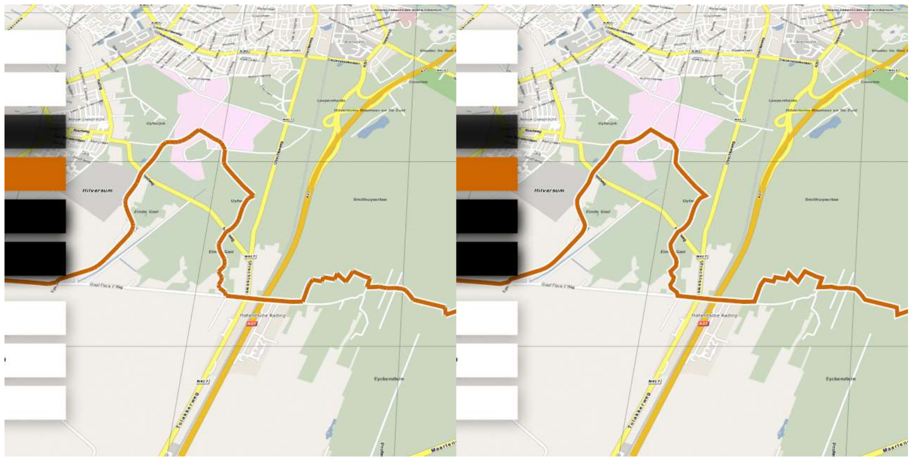
Reageren: Smooth thick lines using geometry shader
8 years ago

Hi,

the GL_TRIANGLES option would not work for me in this case. I had to draw 2D lines in a 3D world, so had to emulate OpenGL's own fixed-function line drawing as much as possible. Had I used a 3D triangle mesh (where all z-coordinates were 0), the line would become thinner at greater distances and thicker when near the camera. That was not what I wanted.

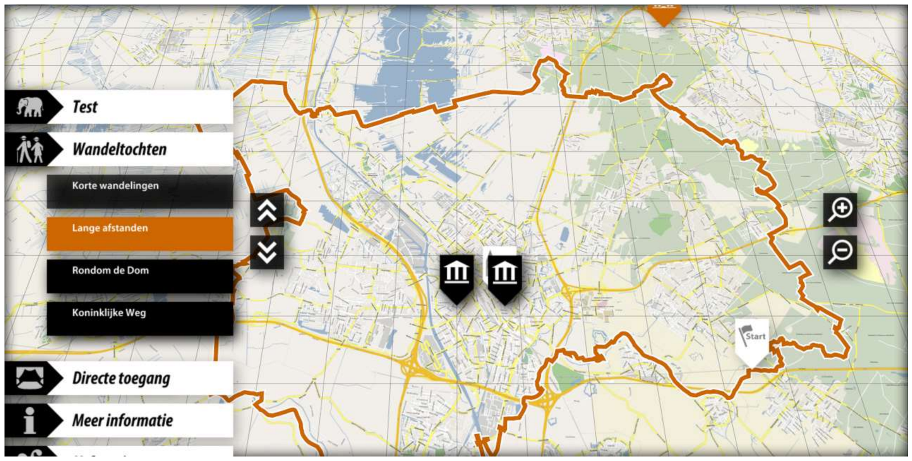
Since I had to draw in 2D screen space and had to avoid recreating the mesh every time the camera moves, I was forced to use a geometry shader. Apart from that, it was a nice opportunity to learn about geometry shaders in the first place.

Here's a screenshot to compare line drawing without and with the shader (click to enlarge):



As you can see, the map is set in perspective, but the line had to be of the same thickness wherever and however it is drawn. The reason for that is that the map can be zoomed quite a lot, from street-level to viewing the whole country. The line has to be drawn with the same thickness all the time, so it remains clearly visible at every size and does not disappear when zooming out.

Here's a zoomed out version of that same track:



-Paul

paul.houx

Reageren: Smooth thick lines using geometry shader

8 years ago

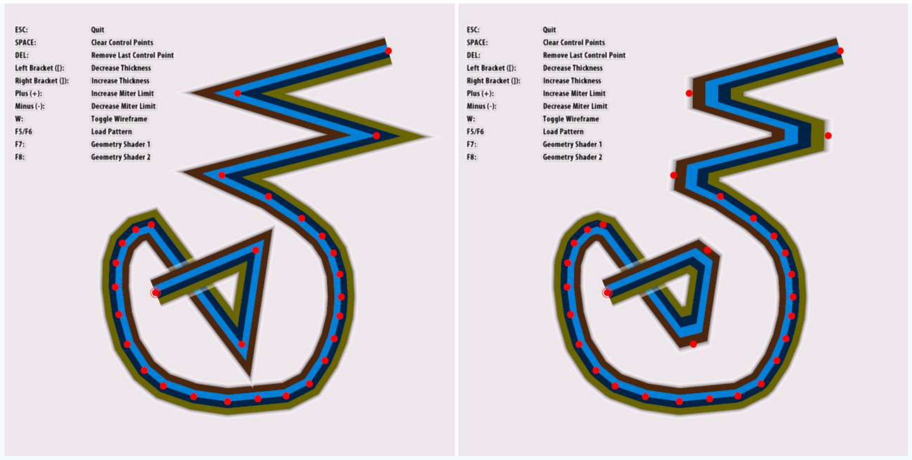
Hi,

I have updated the sample to explore a few different line rendering techniques, as well as texturing. You can still download it from: http://www.cowlumbus.nl/forum/GeometryShader.zip

When playing with it, try this:

- Draw a zig-zagging line by placing points with your mouse. Make a few very sharp corners.
- Change the miter limit using the Plus (+) and Minus (-) keys and note how some of the corners will get cut off, or not.
- Reduce the miter limit to the minimum by repeatedly pressing Minus (-), then switch to the second Geometry shader by pressing F8. The outline will remain the same, but tessellation will be different which in some cases may look better (I personally still prefer the looks of shader 1 (F7)).
- Shader 1 will be better for drawing lines with corners, but can't handle very sharp corners well.
- Shader 2 will produce smoother curves, but the line will not always be centered on the control points and won't appear to have the same thickness everywhere.

Both shaders are ready for use in your own projects. Let me know what you think.


ESC: Quit
SPACE: Clear Control Points
DEL: Remove Last Control Point
Left Bracket ([): Decrease Thickness
Right Bracket (]): Increase Thickness
Plus (+): Increase Miter Limit
Minus (-): Decrease Miter Limit
W: Toggle Wireframe
F5/F6 Load Pattern
F7: Geometry Shader 1
F8: Geometry Shader 2

-Paul

gabor_papp

Re: Smooth thick lines using geometry shader

8 years ago

Hi Paul,

Really great. Thanks for sharing.

I got some shader errors in OS X 10.6:

Copy code

```
Could not compile shader:FRAGMENT: ERROR: 0:7: 'in' :  supported in geometry shaders only

ERROR: 0:7: '' : 'varying in's need to be of array type
ERROR: 0:11: '.' : cannot apply dot operator to an array
```

modifying line 7 of lines_frag.glsl fixed it.

Copy code

```
varying vec2 gsTexCoord;
```

And it seems to work very nicely. Thank you.

paul.houx

Reageren: Re: Smooth thick lines using geometry shader

8 years ago

Thanks for that, Gabor. Yup, it is inevitable that the shader doesn't work the same way on the Mac as it does on Windows. It usually takes a little bit of tinkering.

-Paul

gabor_papp

Re: Smooth thick lines using geometry shader

8 years ago

hi Paul,

could you give me some advices how to make it work with line loops, please? i briefly added the first and last 2 vertex indices to the indices vector, but it does not seem right. do i have to create some more adjacency vertices?

paul.houx

Reageren: Re: Smooth thick lines using geometry shader
8 years ago

Hi,

the shader in this sample works with GL_LINES_ADJACENCY_EXT meshes. This means that for ***each*** segment (every 2 vertices) of the "line loop", you will have to supply 4 vertices in the index buffer.

Alternatively, you can use GL_LINE_STRIP_ADJACENCY_EXT, in which case your code should work. See also: http://web.engr.oregonstate.edu/~mjb/glman/ClassNotes/geometry_shader.pdf (page 3)

-Paul

gabor_papp

Re: Smooth thick lines using geometry shader
8 years ago

hi Paul,

thanks for the explanation and the link. it works fine now.

Gabor

mirzavio

Re: Smooth thick lines using geometry shader
7 years ago

I tried to use some of the code you posted but I have problem with WIN_SCALE attribute. I am working with QGLfunctions API which doesn't provide function getWindowSize(). My program is about to draw lines eed smooth th n andick lines.

paul.houx

Reageren: Re: Smooth thick lines using geometry shader
7 years ago

Hi Mirzavio,

the `getWindowSize()` function simply returns the resolution of the window in pixels. You should be able to get that information from somewhere, for example from a window resize event. Note that you need to supply them as floats, not integers, to make the shader work.

-Paul

h.ajla

Re: Smooth thick lines using geometry shader
6 years ago

Hi Paul,

this is very nice post. Thank you for sharing it with us. I am working on similar project and I would like to test CPU performance when calculating these points for thick line. Since you omitted part of calculations, do you have the code in C++ perhaps?

paul.houx

Reageren: Re: Smooth thick lines using geometry shader
6 years ago

Hi,

thank you for your reply. I am quite sure all code can be found in the posts. The basic algorithm is outlined above, and the C++ code can be found in the accompanying ZIP:
http://www.cowlumbus.nl/forum/Polyline2TriMeshTest.zip

The shader code can be found in the sample on GitHub:
https://github.com/paulhoux/Cinder-Samples/tree/master/GeometryShader

-Paul

h.ajla

Re: Smooth thick lines using geometry shader
6 years ago

Thanks for quick reply.

So you recommend me rewriting your shaders since I want to calculate points on CPU instead on GPU. There are two geom shaders, which one should I use.

I am sorry for asking too much, I would be very happy if you answer me.

Ajla

paul.houx

Reageren: Re: Smooth thick lines using geometry shader
6 years ago

Hi Ajla,

nope, I recommend you to take a look at Polyline2TriMeshTest.zip :) It's CPU code.

-Paul

dazzid

Re: Smooth thick lines using geometry shader
5 years ago

Hi Paul,
GeometryShader runs on cinder 0.9.0? Or should I compile using 0.8.6?
Greetings
David

paul.houx

Re: Re: Smooth thick lines using geometry shader
5 years ago

Hi David,

the samples have all been updated to Cinder 0.9.0, including the thick lines sample. No need to go back to ancient history :)

-Paul

dazzid

Re: Smooth thick lines using geometry shader
5 years ago

Awesome!
Do you think they would be a good way to make 3d paths? maybe billboarding the triangles?
It is great how much we learn from cinder
Thanks

paul.houx

Re: Re: Smooth thick lines using geometry shader
5 years ago

Hi,

actually, the technique was designed specifically for 2D (screenspace) lines, not for 3D paths. That is, the line always has the same thickness, regardless the distance from the camera. You'd have to tweak the shader to use world space coordinates and then it would also work in 3D.

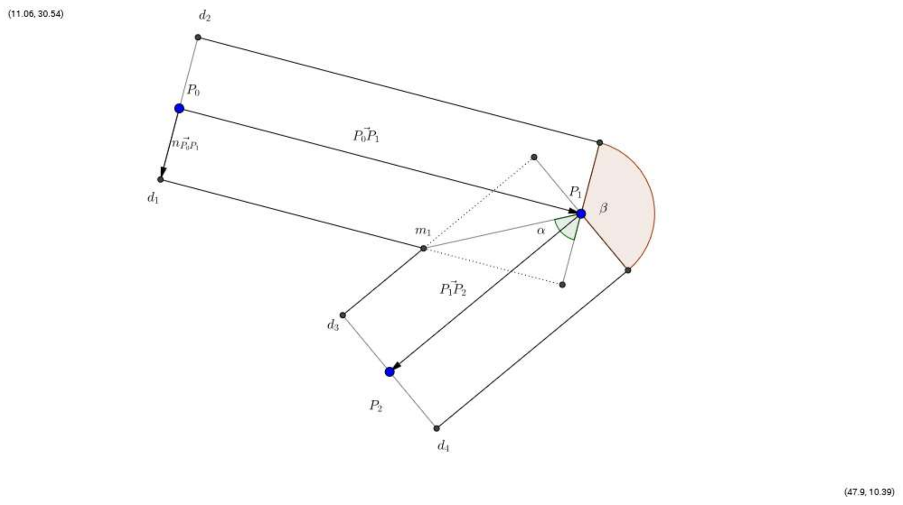
-Paul

Wojciech ..

Odp.: Smooth thick lines using geometry shader
4 years ago

Thank You for your great algorithm.

What about changing sharp corners to circles, like this?

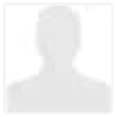


paul.houx

Re: Odp.: Smooth thick lines using geometry shader
4 years ago

Yeah, that would be awesome. Unfortunately this requires more than a few vertices and this quickly degrades performance. Maybe it's something you can try for yourself?

-Paul


Wojciech ..

Odp.: Smooth thick lines using geometry shader
4 years ago

I've already did it. Unfortunately not yet in form of shader :(

Inspired by your algorithm and mathematics I prepared my own version of line algorithm, which calculates a part of circle instead of miters. I needed it to implement it in free, open-sourced therapeutic game made in unity3d (https://github.com/Motoryka/FriendlyLines), if you don't mind. I haven't used Yours code

paul.houx

Re: Odp.: Smooth thick lines using geometry shader
4 years ago

Looks cool :)

I think you can also achieve a circular miter by using the existing, sharp miter in combination with a distance check in the fragment shader. I might give that a try later this week.

-Paul

sharkbox

Re: Smooth thick lines using geometry shader
4 years ago

If you want to do screenspace lines in 3D, I found this to be a helpful guide http://codeflow.org/entries/2012/aug/05/webgl-rendering-of-solid-trails/ It's in webgl but isn't too hard to port, and it does a good job of explaining the technique.

Caleb

Re: Smooth thick lines using geometry shader
3 years ago

@paul.houx were you ever able to explore using a distance checker in the fragment shader? I am not sure exactly how to implement this since we dont just need a circle to be drawn. We also need to have a segment attached to it.

paul.houx

Re: Re: Smooth thick lines using geometry shader
3 years ago

Hi Caleb,

Reminder: we have a new forum, in the future you may want to post new questions here: http://discourse.libcinder.org/

Therefor, I have chosen to answer your question there. Here's the link: http://discourse.libcinder.org/t/smooth-efficient-perfect-curves/925

-Paul

Re: Smooth thick lines using geometry shader
3 years ago

audioboy ..

Hi

I'm trying to port this to my own 2D graphics framework.

The coordinates seem to be 'off'. Are the input coordinates supposed to be in window pixel units, or normalised (0 to 1) or (-0.5 to 0.5)? My system has coordinates in pixels with the 0,0 origin at the top left of the screen. Its somehow working put the points appearing in the wrong place and the lines are not parallel...

EDIT: was a mistake in my ported code, sorry. working great now.

thanks

---

audioboy ..

Re: Smooth thick lines using geometry shader
3 years ago

After several hours of testing and playing around with the possiblities this offers, I just want to say this is really fantastic.   No need to extend it with rounded joints (as requested above), because you can just plot these lines through complex curved paths anyway, it behaves very well even when theres lots of dense points, and when combined with antialiasing, it can be used to draw beautifully smooth rounded rectangles that would make even Steve Jobs proud. I'm also using it to generate really nice vectorised icons for my app. Looks beautiful when animated too, line thickness, rotation etc. A really robust and elegant algorithm, thanks again.

---

paul.houx

Reageren: Re: Smooth thick lines using geometry shader
3 years ago

Thanks, I appreciate your feedback!

Reminder: we have a new forum, in the future you may want to post new questions and remarks here: http://discourse.libcinder.org/

---

audioboy ..

Re: Smooth thick lines using geometry shader
2 years ago

Hi Paul

One issue i've run into in practical usage, is that when using the algorithm to draw lines in non-square transforms, the lines are (understandably) distorted, depending on the angle.

For example I have a graph display say about 320 x 160 pixels on the screen, but the units used there are 1-1000 on the x axis and -1 to +1 on the y axis. when trying to use the algorithm there the results are not good.

Is there a way to use the algorithm in such cases? i guess i somehow need to set the line width for both axis independently....?

Noted about the other forum, is there a thread on your algorithm i should use for any future posts?

thanks

Reageren: Re: Smooth thick lines using geometry shader
2 years ago

paul.houx

Just create a new post and link to this old one.

Top Reply