

# Project Group: Vector Graphics on Modern Hardware (VGMH)

Dorian Rudolph (dorian.rudolph@upb.de)

Prof. Dr. Sevag Gharibian

<https://git.cs.uni-paderborn.de/vgmh/info>

## Canonical example



"Ghostscript tiger"

# Motivation

*Why are the quantum computing people doing a project group on vector graphics?*

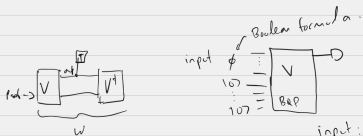
Def: TCount

Input: circuit  $U$ , integer  $K$ , real  $\epsilon = \frac{1}{\text{poly}}$

Output: Yes if  $\exists$  circuit  $V$  with  $\leq k$  T gates  
s.t.  $\exists \theta \in [0, 2\pi)$ .  $\|UV^\dagger - e^{i\theta}I\| < \epsilon$

No if  $\forall$  circuits  $V$  with  $\leq k$  T gates  
and  $\forall \theta \in [0, 2\pi)$ .  $\|UV^\dagger - e^{i\theta}I\| > 2\epsilon$

Stable is coQMA complete



$$W|1\rangle|0\rangle \approx \tau|1\rangle \otimes |\phi\rangle$$

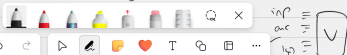
↑  
acc. prob

parameter  $k$ .

input:  $V_{in}$ , quantum verification circuit.

eq-2:  $\exists$  circuit desc  $V$  with  $\leq k$  gates.  
s.t.  $\forall \phi \in \mathcal{H}_k$   $\|V|x\rangle - V_{in}|\phi\rangle\| \leq \frac{1}{\text{poly}}$ .

Yes:  $\exists x \forall |y\rangle$   $V$  accepts  $(x, |y\rangle)$  w.p.  $\geq \frac{2}{3}$   
No:  $\forall x \exists |y\rangle$   $V$  " " " "  $\leq \frac{1}{3}$



Why are

s?

# Motivation

*Why are the quantum computing people doing a project group on vector graphics?*

- ▶ None of the existing digital whiteboard apps are *great*.

# Motivation

*Why are the quantum computing people doing a project group on vector graphics?*

- ▶ None of the existing digital whiteboard apps are *great*.
  - ▶ not supporting all platforms (Linux, Windows, Mac, Android, iOS, Web)

# Motivation

*Why are the quantum computing people doing a project group on vector graphics?*

- ▶ None of the existing digital whiteboard apps are *great*.
  - ▶ not supporting all platforms (Linux, Windows, Mac, Android, iOS, Web)
  - ▶ bad performance

# Motivation

*Why are the quantum computing people doing a project group on vector graphics?*

- ▶ None of the existing digital whiteboard apps are *great*.
  - ▶ not supporting all platforms (Linux, Windows, Mac, Android, iOS, Web)
  - ▶ bad performance
  - ▶ missing “multiplayer”



# Motivation

*Why are the quantum computing people doing a project group on vector graphics?*

- ▶ None of the existing digital whiteboard apps are *great*.
  - ▶ not supporting all platforms (Linux, Windows, Mac, Android, iOS, Web)
  - ▶ bad performance
  - ▶ missing “multiplayer”
  - ▶ bad/missing PDF integration

# Motivation

*Why are the quantum computing people doing a project group on vector graphics?*

- ▶ None of the existing digital whiteboard apps are *great*.
  - ▶ not supporting all platforms (Linux, Windows, Mac, Android, iOS, Web)
  - ▶ bad performance
  - ▶ missing “multiplayer”
  - ▶ bad/missing PDF integration

*Ok, so how can we build a better app?*

# Motivation

*Why are the quantum computing people doing a project group on vector graphics?*

- ▶ None of the existing digital whiteboard apps are *great*.
  - ▶ not supporting all platforms (Linux, Windows, Mac, Android, iOS, Web)
  - ▶ bad performance
  - ▶ missing “multiplayer”
  - ▶ bad/missing PDF integration

*Ok, so how can we build a better app?*

- ▶ Need to render vector graphics

# Motivation

*Why are the quantum computing people doing a project group on vector graphics?*

- ▶ None of the existing digital whiteboard apps are *great*.
  - ▶ not supporting all platforms (Linux, Windows, Mac, Android, iOS, Web)
  - ▶ bad performance
  - ▶ missing “multiplayer”
  - ▶ bad/missing PDF integration

*Ok, so how can we build a better app?*

- ▶ Need to render vector graphics
- ▶ Chrome uses Skia, so let's try that...

## Stroke rendering is hard

<https://issues.skia.org/issues/40043907>: Filled paths of width less than 1px defeat GPU anti-aliasing

## Stroke rendering is hard

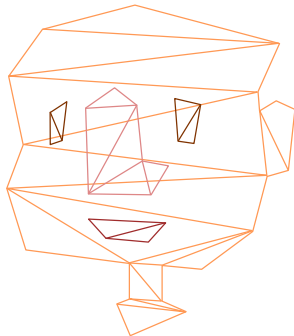
<https://issues.skia.org/issues/40043907>: Filled paths of width less than 1px defeat GPU anti-aliasing

Chrome vs. Inkscape:



## But why?

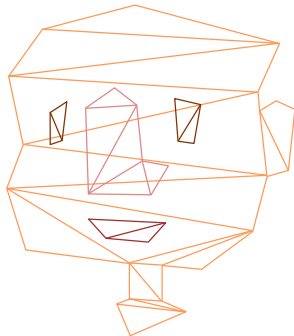
- Skia renders by tessellating 2D paths (turn into triangles)



Source: [https://docs.rs/lyon\\_tessellation](https://docs.rs/lyon_tessellation)

## But why?

- ▶ Skia renders by tessellating 2D paths (turn into triangles)
- ▶ Use 3D pipeline for rendering (rasterization of triangles)

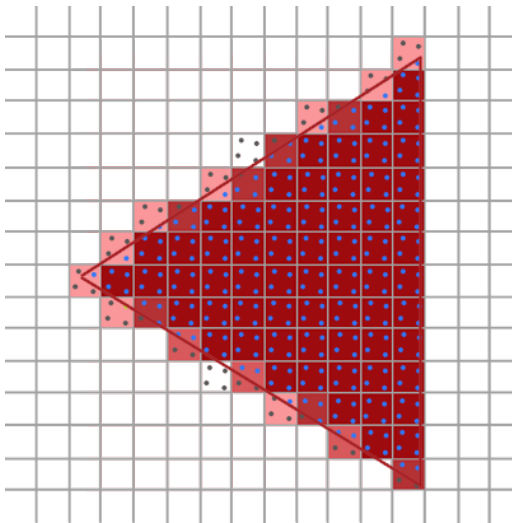


Source: [https://docs.rs/lyon\\_tessellation](https://docs.rs/lyon_tessellation)



## But why?

- ▶ Skia renders by tessellating 2D paths (turn into triangles)
- ▶ Use 3D pipeline for rendering (rasterization of triangles)
- ▶ MSAA for anti-aliasing

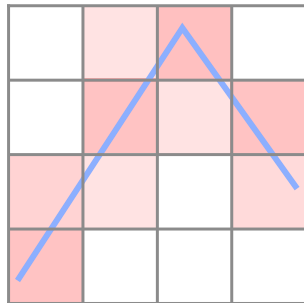
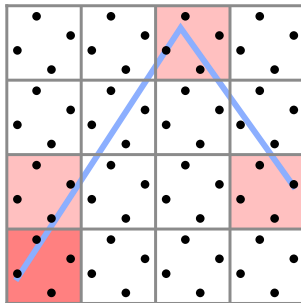


Source:

<https://learnopengl.com/Advanced-OpenGL/Anti-Aliasing>

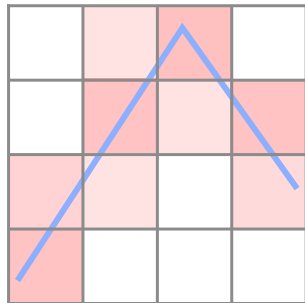
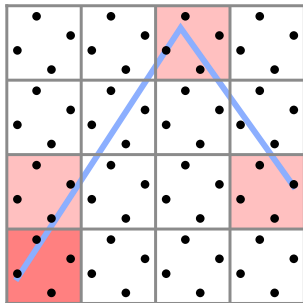
## But why?

- ▶ Skia renders by tessellating 2D paths (turn into triangles)
- ▶ Use 3D pipeline for rendering (rasterization of triangles)
- ▶ MSAA for anti-aliasing
- ▶ Thin paths can “snake” through the sample points



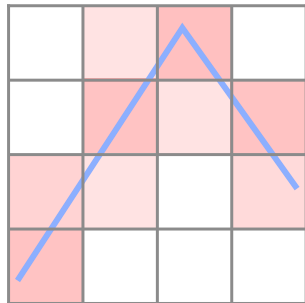
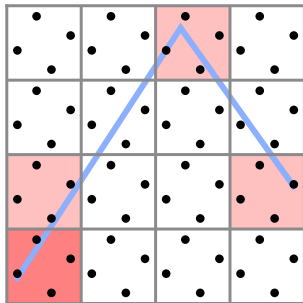
## But why?

- ▶ Skia renders by tessellating 2D paths (turn into triangles)
- ▶ Use 3D pipeline for rendering (rasterization of triangles)
- ▶ MSAA for anti-aliasing
- ▶ Thin paths can “snake” through the sample points
- ▶ Use “analytic anti-aliasing” (compute exact overlap of path with pixel)



## But why?

- ▶ Skia renders by tessellating 2D paths (turn into triangles)
- ▶ Use 3D pipeline for rendering (rasterization of triangles)
- ▶ MSAA for anti-aliasing
- ▶ Thin paths can “snake” through the sample points
- ▶ Use “analytic anti-aliasing” (compute exact overlap of path with pixel)
- ▶ Difficult on GPU



## Prior Work

- LP05 Charles Loop and Jim Blinn. 2005. Resolution independent curve rendering using programmable graphics hardware. ACM Trans. Graph. 24, 3 (July 2005), 1000–1009. <https://doi.org/10.1145/1073204.1073303>
- NH08 Diego Nehab and Hugues Hoppe. 2008. Random-access rendering of general vector graphics. ACM Trans. Graph. 27, 5, Article 135 (December 2008). <https://doi.org/10.1145/1409060.1409088>
- GLFN14 Francisco Ganacim, Rodolfo S. Lima, Luiz Henrique de Figueiredo, and Diego Nehab. 2014. Massively-parallel vector graphics. ACM Trans. Graph. 33, 6, Article 229 (November 2014). <https://w3.impa.br/~diego/projects/GanEtA114/>
- LHZ16 Rui Li, Qiming Hou, and Kun Zhou. 2016. Efficient GPU path rendering using scanline rasterization. ACM Trans. Graph. 35, 6, Article 228 (November 2016). <http://kunzhou.net/zjugaps/pathrendering/>

# Research Project

How do the different rendering approaches compare?

## Research Project

How do the different rendering approaches compare?

- ▶ CPU vs. Tessellation vs. Compute Shader (compare mobile and desktop)

## Research Project

How do the different rendering approaches compare?

- ▶ CPU vs. Tessellation vs. Compute Shader (compare mobile and desktop)
  - ▶ How to do AA in “raster pipeline”?



## Research Project

How do the different rendering approaches compare?

- ▶ CPU vs. Tessellation vs. Compute Shader (compare mobile and desktop)
  - ▶ How to do AA in “raster pipeline”?
- ▶ [github.com/linebender/vello](https://github.com/linebender/vello) has no paper, but is supposedly competitive.

# Research Project

How do the different rendering approaches compare?

- ▶ CPU vs. Tessellation vs. Compute Shader (compare mobile and desktop)
  - ▶ How to do AA in “raster pipeline”?
- ▶ `github.com/linebender/vello` has no paper, but is supposedly competitive.
- ▶ Papers only use CUDA on nVidia GPU, but how do their techniques fare on mobile/laptop/web? → *modern hardware*

# Research Project

How do the different rendering approaches compare?

- ▶ CPU vs. Tessellation vs. Compute Shader (compare mobile and desktop)
  - ▶ How to do AA in “raster pipeline”?
- ▶ `github.com/linebender/vello` has no paper, but is supposedly competitive.
- ▶ Papers only use CUDA on nVidia GPU, but how do their techniques fare on mobile/laptop/web? → *modern hardware*

Your task:

# Research Project

How do the different rendering approaches compare?

- ▶ CPU vs. Tessellation vs. Compute Shader (compare mobile and desktop)
  - ▶ How to do AA in “raster pipeline”?
- ▶ `github.com/linebender/vello` has no paper, but is supposedly competitive.
- ▶ Papers only use CUDA on nVidia GPU, but how do their techniques fare on mobile/laptop/web? → *modern hardware*

Your task:

- ▶ Understand and write up the different approaches.

# Research Project

How do the different rendering approaches compare?

- ▶ CPU vs. Tessellation vs. Compute Shader (compare mobile and desktop)
  - ▶ How to do AA in “raster pipeline”?
- ▶ [github.com/linebender/vello](https://github.com/linebender/vello) has no paper, but is supposedly competitive.
- ▶ Papers only use CUDA on nVidia GPU, but how do their techniques fare on mobile/laptop/web? → *modern hardware*

Your task:

- ▶ Understand and write up the different approaches.
- ▶ Find common abstraction and implement algorithms.

# Research Project

How do the different rendering approaches compare?

- ▶ CPU vs. Tessellation vs. Compute Shader (compare mobile and desktop)
  - ▶ How to do AA in “raster pipeline”?
- ▶ [github.com/linebender/vello](https://github.com/linebender/vello) has no paper, but is supposedly competitive.
- ▶ Papers only use CUDA on nVidia GPU, but how do their techniques fare on mobile/laptop/web? → *modern hardware*

Your task:

- ▶ Understand and write up the different approaches.
- ▶ Find common abstraction and implement algorithms.
- ▶ Benchmark on various devices (both performance and visual fidelity).

# Research Project

How do the different rendering approaches compare?

- ▶ CPU vs. Tessellation vs. Compute Shader (compare mobile and desktop)
  - ▶ How to do AA in “raster pipeline”?
- ▶ [github.com/linebender/vello](https://github.com/linebender/vello) has no paper, but is supposedly competitive.
- ▶ Papers only use CUDA on nVidia GPU, but how do their techniques fare on mobile/laptop/web? → *modern hardware*

Your task:

- ▶ Understand and write up the different approaches.
- ▶ Find common abstraction and implement algorithms.
- ▶ Benchmark on various devices (both performance and visual fidelity).
- ▶ Try to find improvements or even your own algorithms.

# Research Project

How do the different rendering approaches compare?

- ▶ CPU vs. Tessellation vs. Compute Shader (compare mobile and desktop)
  - ▶ How to do AA in “raster pipeline”?
- ▶ `github.com/linebender/vello` has no paper, but is supposedly competitive.
- ▶ Papers only use CUDA on nVidia GPU, but how do their techniques fare on mobile/laptop/web? → *modern hardware*

Your task:

- ▶ Understand and write up the different approaches.
- ▶ Find common abstraction and implement algorithms.
- ▶ Benchmark on various devices (both performance and visual fidelity).
- ▶ Try to find improvements or even your own algorithms.
- ▶ How to deal with scene updates? (Not discussed much in the literature)



# Programming Project

Build a collaborative digital whiteboard app using your own renderer.

# Programming Project

Build a collaborative digital whiteboard app using your own renderer.

- ▶ Compare different renderers in practice.

# Programming Project

Build a collaborative digital whiteboard app using your own renderer.

- ▶ Compare different renderers in practice.
- ▶ Publish as open source.

# Programming Project

Build a collaborative digital whiteboard app using your own renderer.

- ▶ Compare different renderers in practice.
- ▶ Publish as open source.
- ▶ Port to different platforms: Linux, Windows, Mac, Android, iOS, web

# Programming Project

Build a collaborative digital whiteboard app using your own renderer.

- ▶ Compare different renderers in practice.
- ▶ Publish as open source.
- ▶ Port to different platforms: Linux, Windows, Mac, Android, iOS, web
  - ▶ Don't need to port to all platforms, but make sure to use platform independent techniques.

# Programming Project

Build a collaborative digital whiteboard app using your own renderer.

- ▶ Compare different renderers in practice.
- ▶ Publish as open source.
- ▶ Port to different platforms: Linux, Windows, Mac, Android, iOS, web
  - ▶ Don't need to port to all platforms, but make sure to use platform independent techniques.

Nice-to-have features:

# Programming Project

Build a collaborative digital whiteboard app using your own renderer.

- ▶ Compare different renderers in practice.
- ▶ Publish as open source.
- ▶ Port to different platforms: Linux, Windows, Mac, Android, iOS, web
  - ▶ Don't need to port to all platforms, but make sure to use platform independent techniques.

Nice-to-have features:

- ▶ Pen input: Converting coordinates and pressures to strokes is almost its own research question...

# Programming Project

Build a collaborative digital whiteboard app using your own renderer.

- ▶ Compare different renderers in practice.
- ▶ Publish as open source.
- ▶ Port to different platforms: Linux, Windows, Mac, Android, iOS, web
  - ▶ Don't need to port to all platforms, but make sure to use platform independent techniques.

Nice-to-have features:

- ▶ Pen input: Converting coordinates and pressures to strokes is almost its own research question...
- ▶ PDF integration (can use PDFium library)



# Programming Project

Build a collaborative digital whiteboard app using your own renderer.

- ▶ Compare different renderers in practice.
- ▶ Publish as open source.
- ▶ Port to different platforms: Linux, Windows, Mac, Android, iOS, web
  - ▶ Don't need to port to all platforms, but make sure to use platform independent techniques.

Nice-to-have features:

- ▶ Pen input: Converting coordinates and pressures to strokes is almost its own research question...
- ▶ PDF integration (can use PDFium library)
- ▶ End-to-end encryption

# Programming Project

Build a collaborative digital whiteboard app using your own renderer.

- ▶ Compare different renderers in practice.
- ▶ Publish as open source.
- ▶ Port to different platforms: Linux, Windows, Mac, Android, iOS, web
  - ▶ Don't need to port to all platforms, but make sure to use platform independent techniques.

Nice-to-have features:

- ▶ Pen input: Converting coordinates and pressures to strokes is almost its own research question...
- ▶ PDF integration (can use PDFium library)
- ▶ End-to-end encryption
- ▶ Your own ideas...

# Skills

- ▶ Programming in a system programming language (probably Rust)
- ▶ GPU programming (probably wgpu/WebGPU) and rendering
- ▶ Read and understand scientific papers