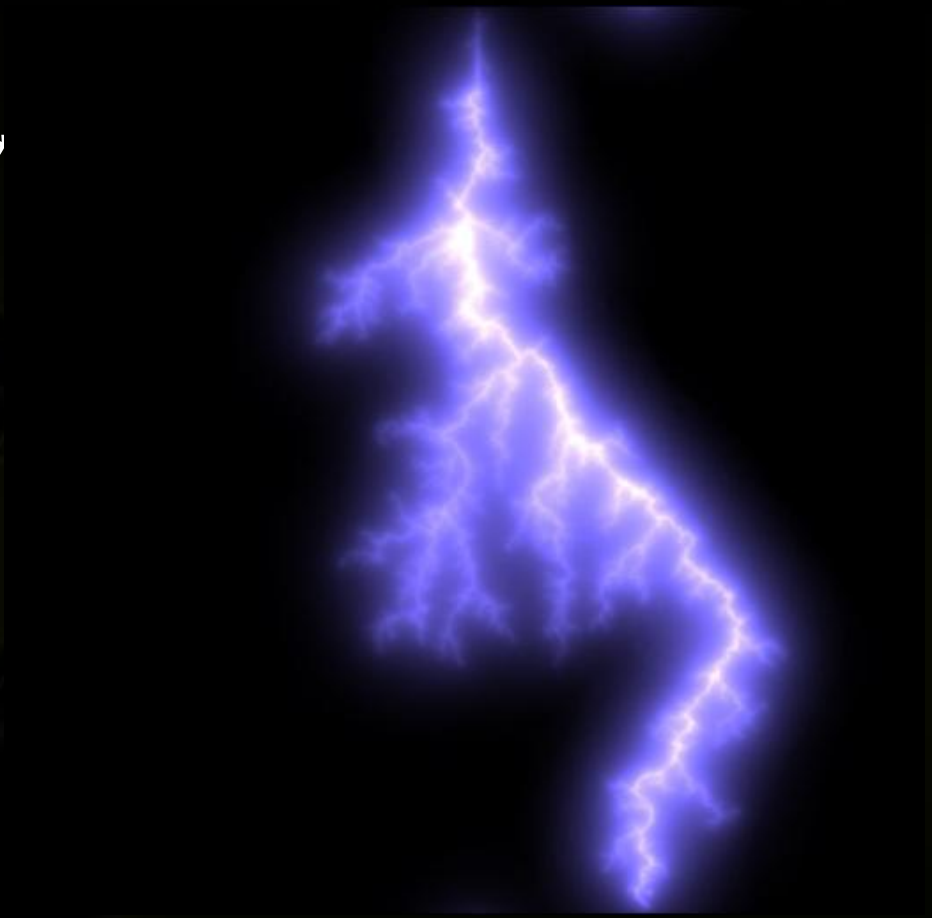
Lightning

SDK demo explained

Previous Work

- Physical simulation using Dielectric Breakdown Model*
  - Slow
  - Convolution with wide filter
- Structure from statistics
  - Raytracing / volume traversal
- Pregenerated animation
  - Not flexible
- CPU based generation

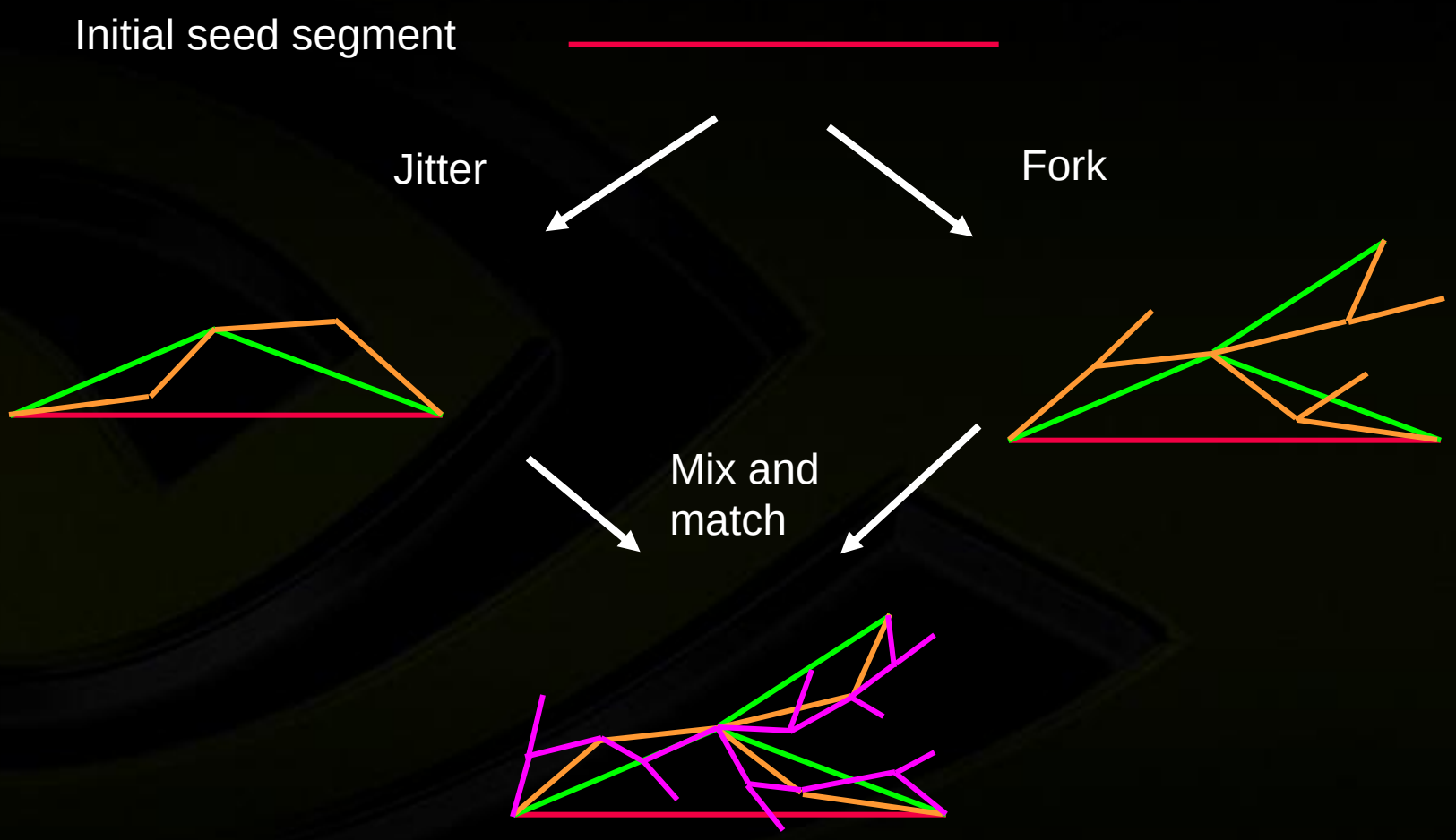


*http://gamma.cs.unc.edu/LIGHTNING/

Algorithm

- Generate lightning structure
  - Random fractal L – system in the GS
  - Multiple iterations of subdivisions to generate more segments
  - Animation
- Rendering
  - Constrained billboards
- Post processing
  - Blurring for glow & atmospheric scattering

Generation of Lightning Structure (1)



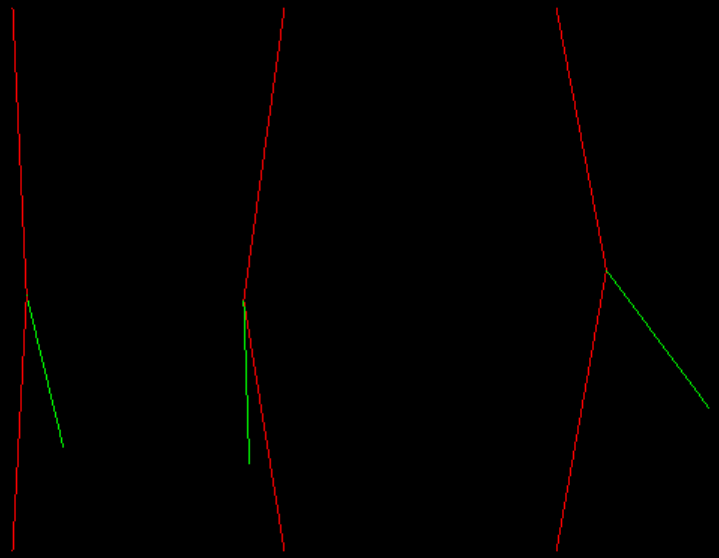
Generation of Lightning Structure (2)

- Store line segment as single vertex and render points
  - contains also “up” vector for orientation of deviation
- GS emits 2 or 3 vertices, depending whether to jitter or fork
  - controlled by current (global) subdivision level
- Loop with StreamOut & BufferPingPong
- 5 to 6 subdivisions give good results
  - Between 64 and 729 segments per seed segment
  - 2 * fork + 3 * jitter looks good

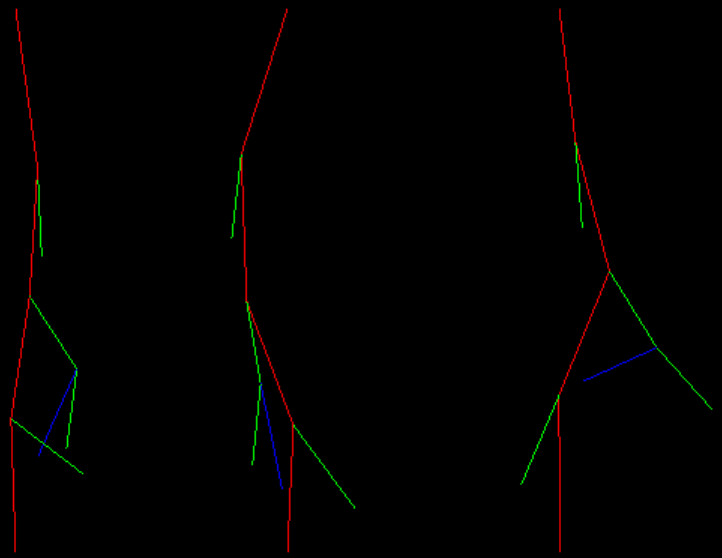
Generation of Lightning Structure (3)

- Pseudo random numbers
  - DX9 style: store as textures
  - DX10 style: use integer / bit operations
- Code from Numerical Recipes
  (http://www.library.cornell.edu/nr/cbookcpdf.html)
- Ideally would have persistent seed value
- Instead use primitive ID as seed value
  - Variation across primitives
  - Animation is easy, just add time to seed value
    - Jumpy appearance, no change in topology
- Use base_value * $e^{-decay * subdivision}$ for control

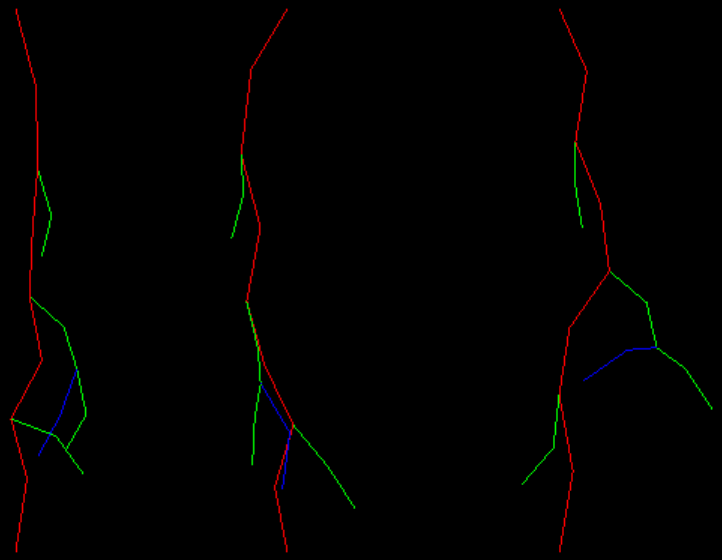
Generation of Lightning Structure (4)



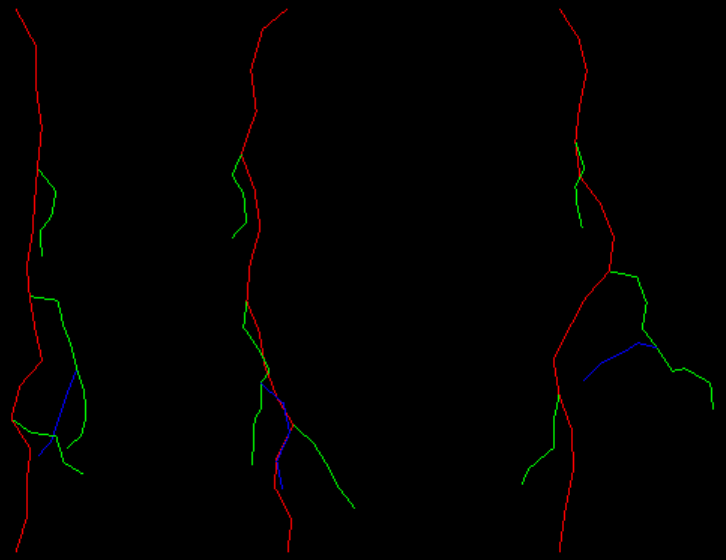
Generation of Lightning Structure (4)



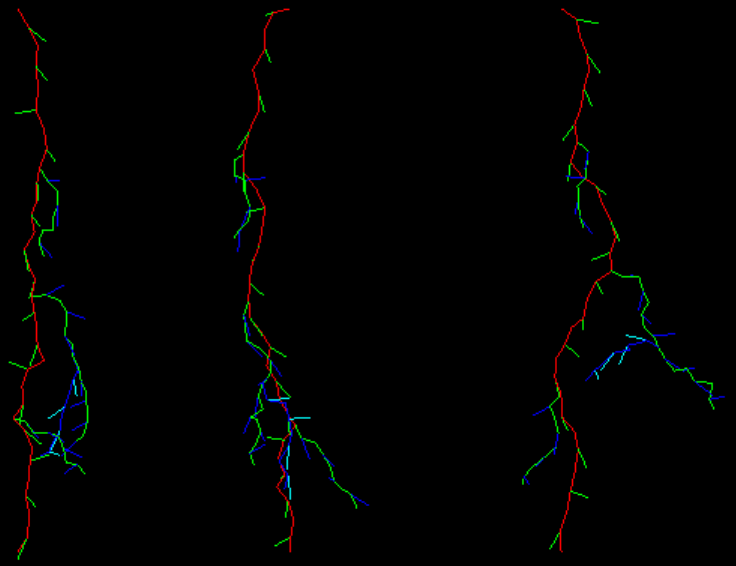
Generation of Lightning Structure (4)



Generation of Lightning Structure (4)



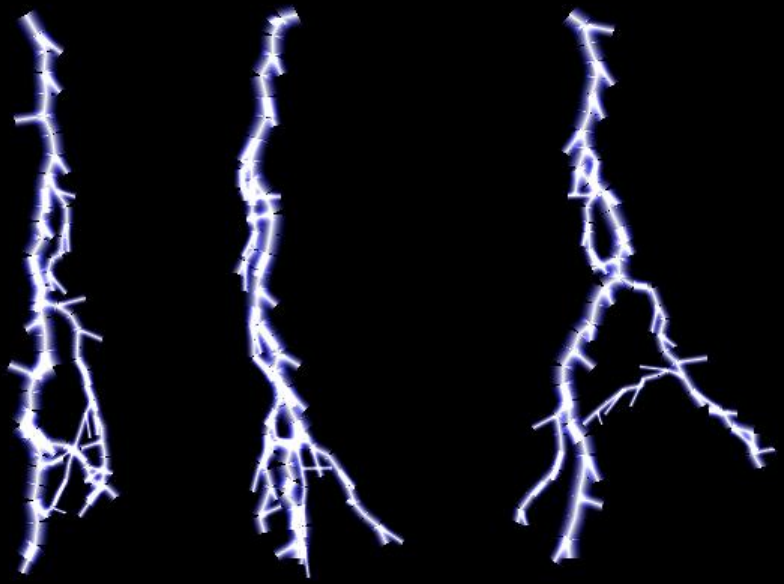
Generation of Lightning Structure (4)



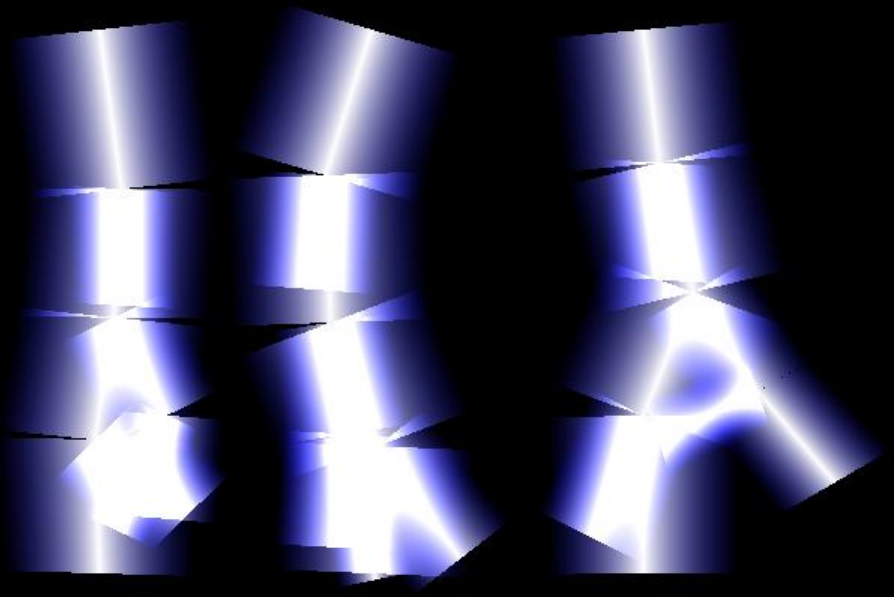
Rendering

- Rendering into separate off-screen RT, but using scene depth buffer, (with matching MSAA settings)
- Generate segment aligned and camera aligned quad with gradient between 2 colors
  - Vary width based on segment “level”
  - Gaps between segments
  - What about segments nearly orthogonal to view direction

Closing Gaps (2) – Starting point



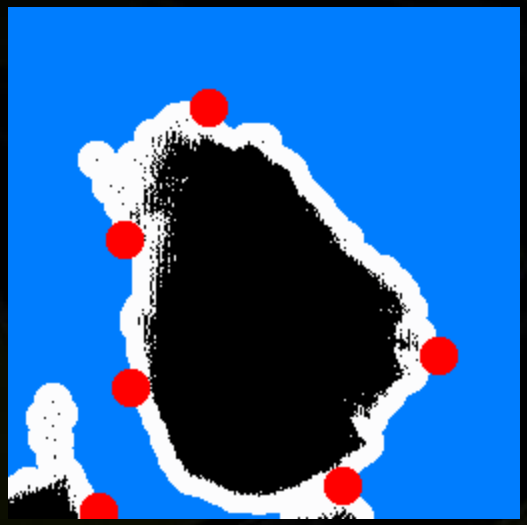
Closing Gaps (2) – Starting point



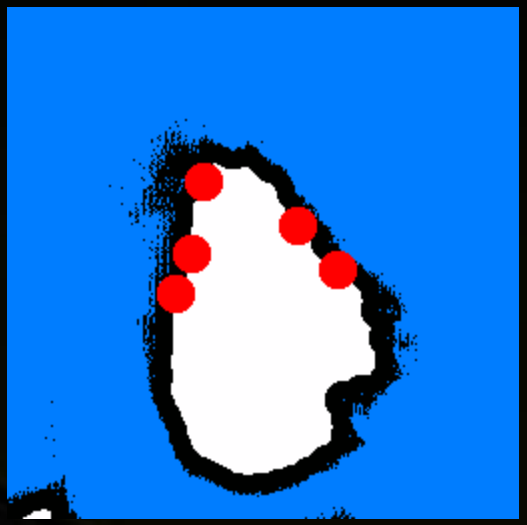
Closing Gaps (1)

- Adjacent vertices unknown during subdivision
  - Cannot use them to adjust / connect quads ☹
- Tried image space growing and shrinking using dilation and erosion
  - Works for small resolutions / gaps
  - Leads to ugly artifacts ☹

Erosion / Dilation



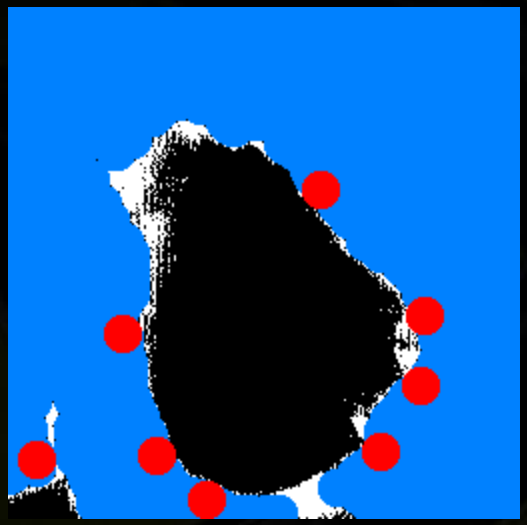
Dilation



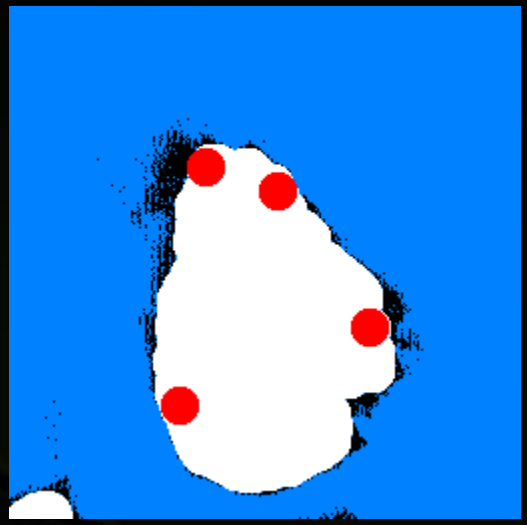
Erosion

http://www.dca.fee.unicamp.br/dipcourse/html-dip/c9/s4/front-page.html

Opening / Closing



Closing = dilation followed by erosion

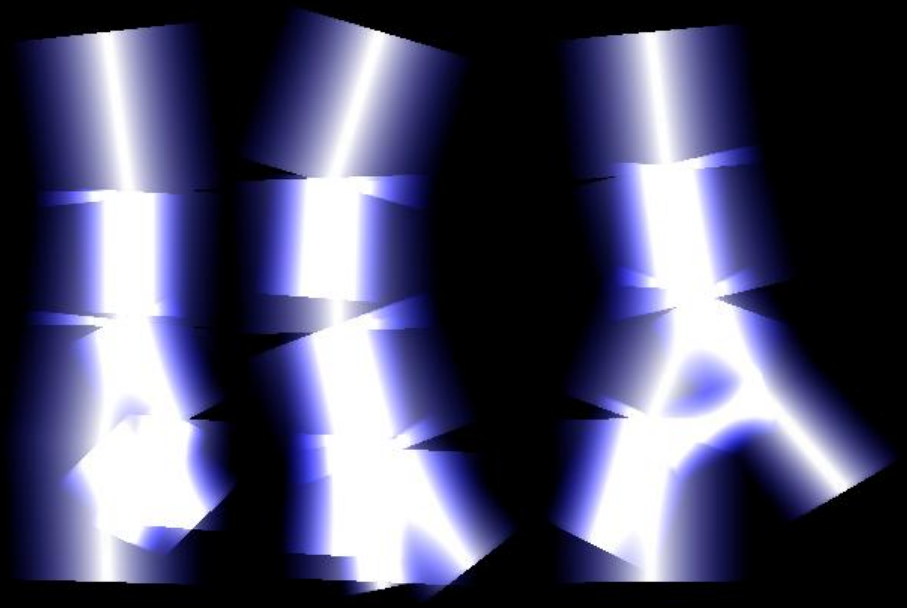


Opening = erosion followed by dilation

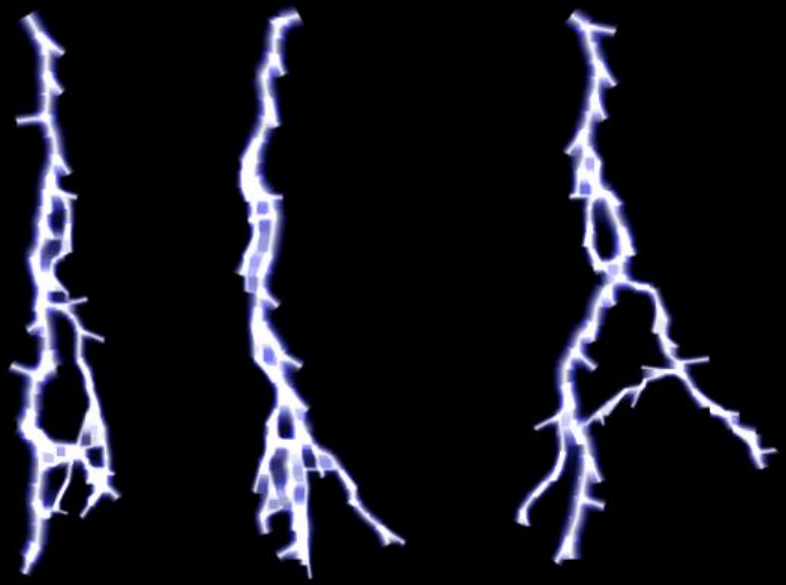
http://www.dca.fee.unicamp.br/dipcourse/html-dip/c9/s4/front-page.html

Closing Gaps (3) — Dilation

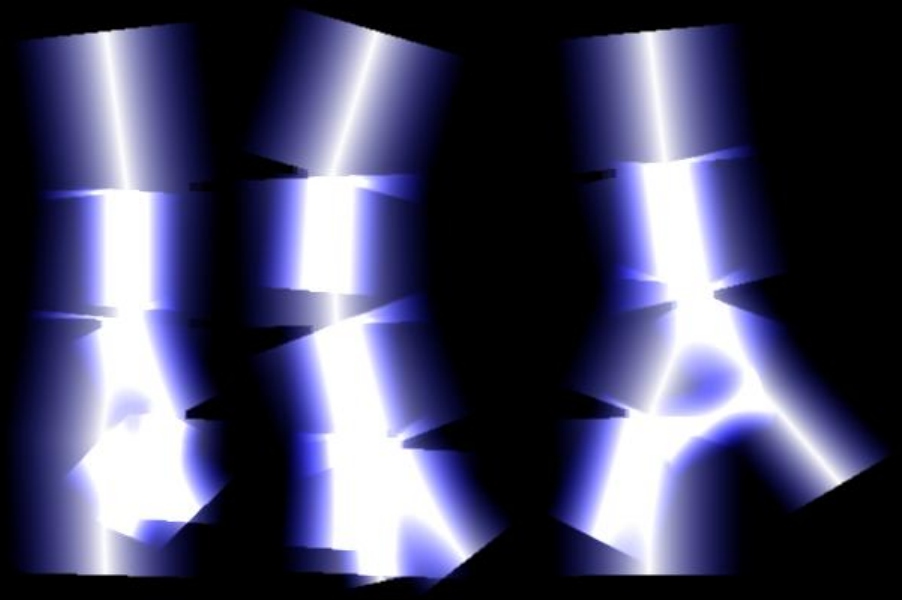
Closing Gaps (3) — Dilation



Closing Gaps (4) — Dilation + Erosion



Closing Gaps (4) — Dilation + Erosion

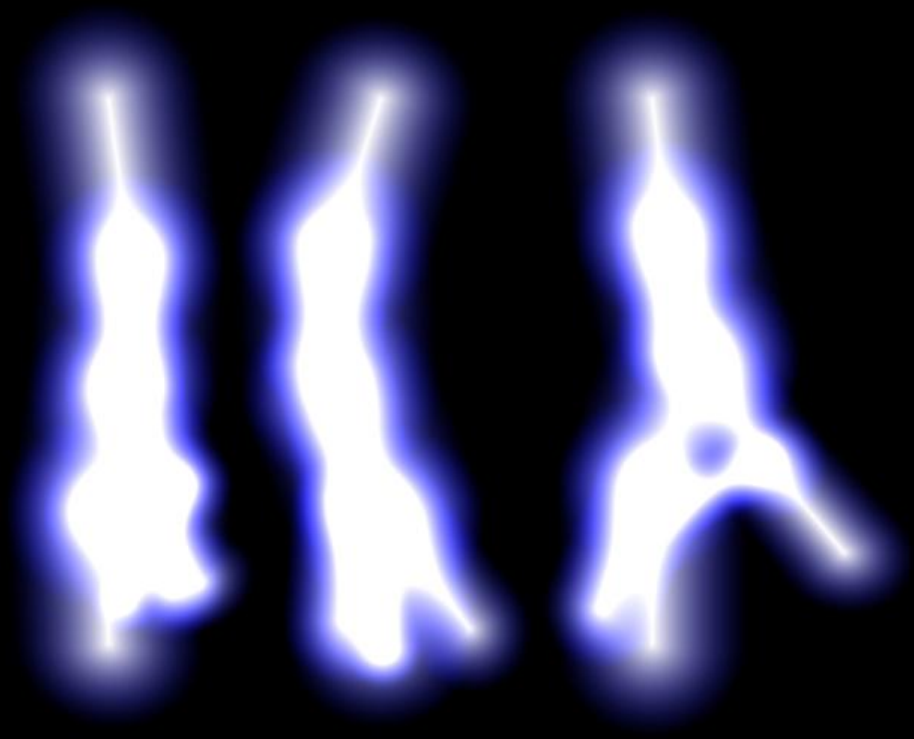


Closing Gaps (5) - Solution

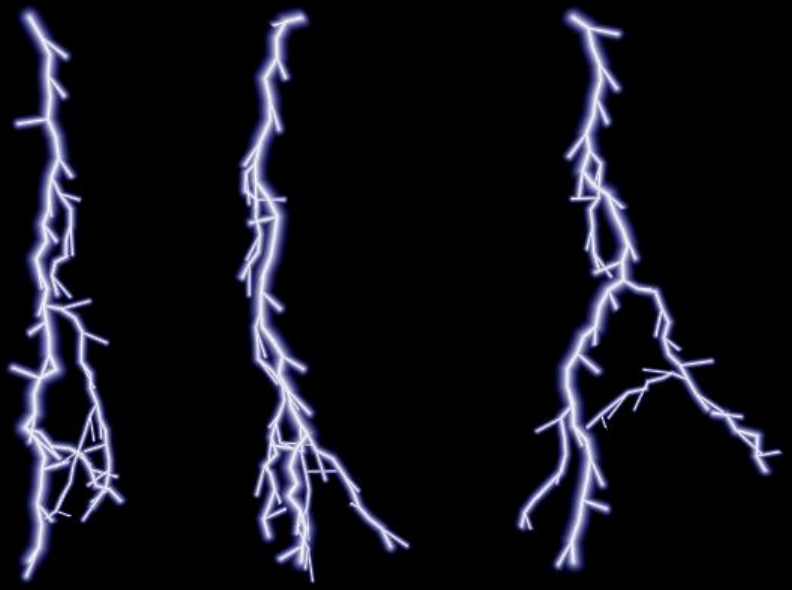
- Terminate each quad with a small square with a semicircular gradient
- Quads of neighbors overlap
  - Leads to overbright spots with additive blending
- Max blending deals with that:
  - fragment_color = max(source, destination)
  - If glow is used as a postprocessing step, additive blending works fine, i.e. it looks better

Closing Gaps (6) — Terminating Quads

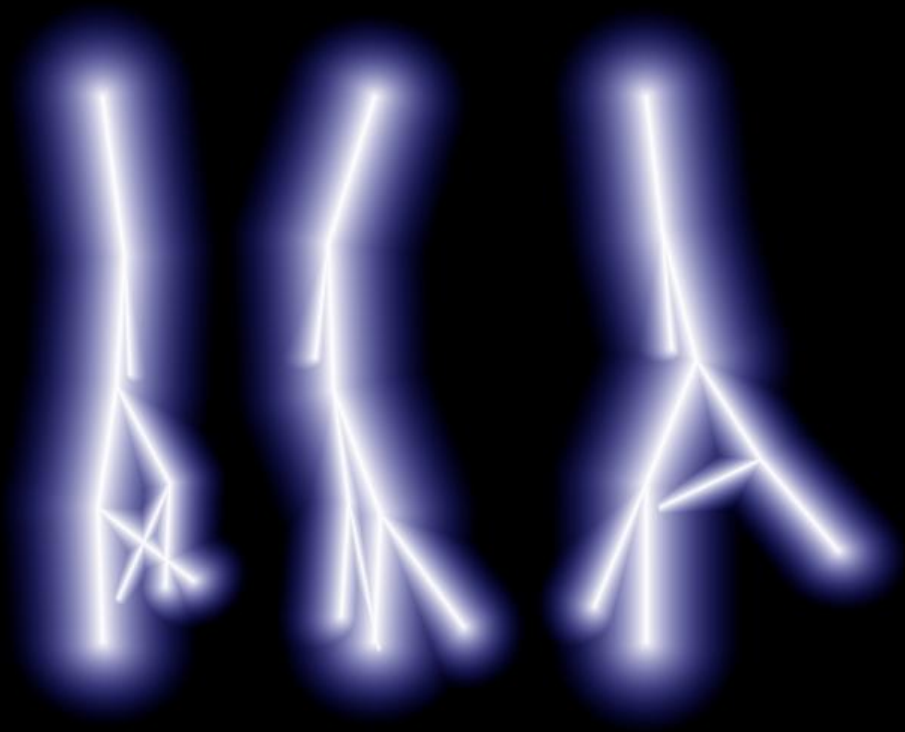
Closing Gaps (6) – Terminating Quads



Closing Gaps (7) — Max Blending



Closing Gaps (7) — Max Blending



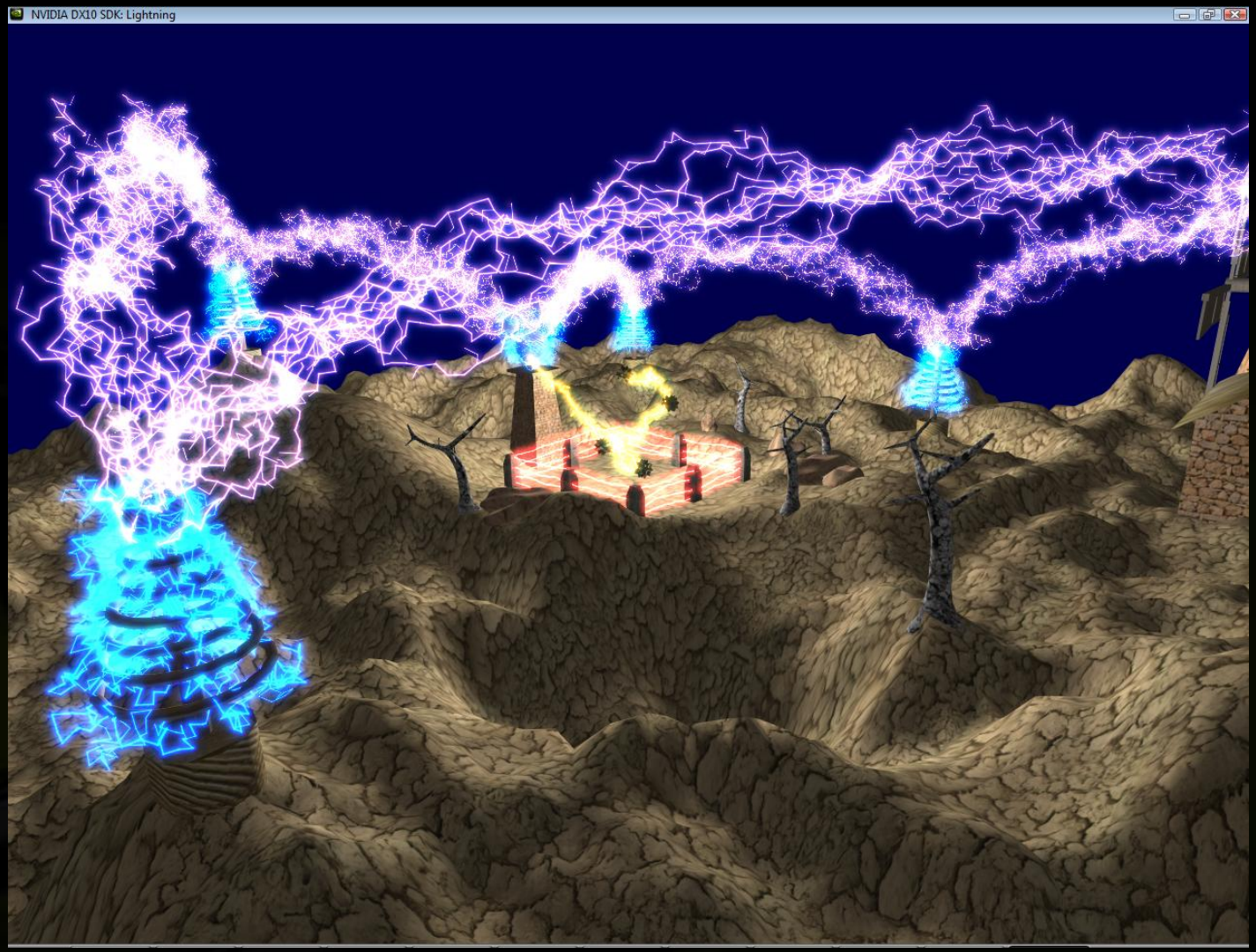
Post processing (1)

- Down sample to ¼ of width and height
  - Blurring less dependent of screen resolution
    - Could have minimal size of downsampled buffer
  - Faster ☺
- Separable Gaussian blur, e.g. 9 pixels support
  - Falloff σ separate for RGB to fake atmospheric scattering
- Scale blurred version up and add to unblurred
  - Small glow

Post Processing (2)

Post Processing (2)

Results



Use cases (1)

- Weather effects
- Electric discharges
  - Beams between electrodes
  - Broken panels, computers
  - Combination with sparkles
- 42 kV fences
- Nebula / clouds in space games

Use cases (2)

- Force lightning
  - Radial lightning burst, restrict deviation to one plane
- Chain lightning
  - Targets can be tracked by spell caster
- Lighting missiles
  - Use 3D cross as seeding lines
- Lightning elementals
  - Use GS to extract edges of low resolution mesh as seed lines

Extensions

- Apply HDR
  - Render bright single pixel lines and let HDR resolve deal with glow
- Wide glow
  - Render dim and very wide bolts following coarse structure (e.g. 2 subdivision levels)
- Add lighting to lightning
  - Use segment centers of coarse subdivision as point light sources