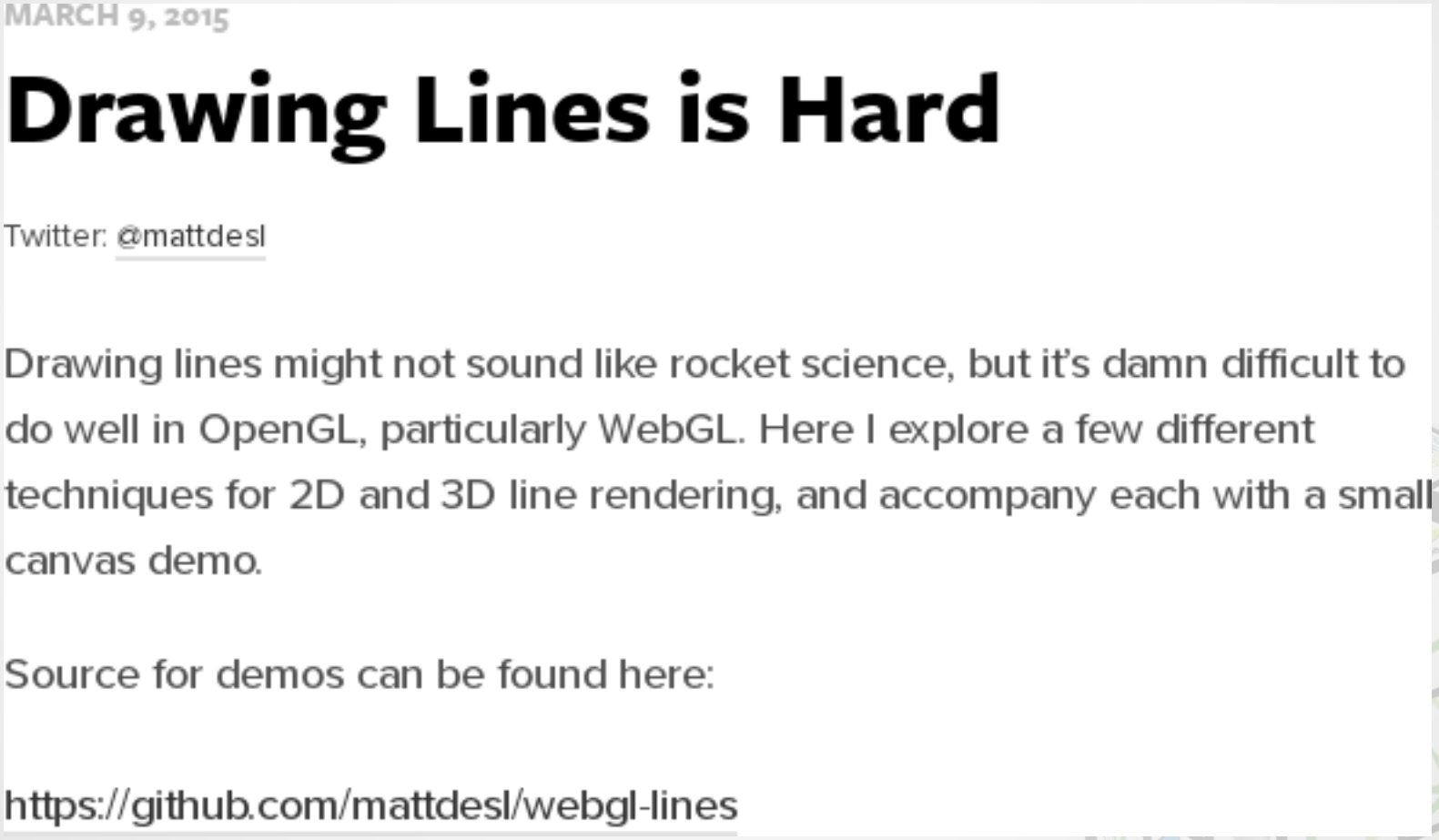
Vonalat húzni nehéz



MARCH 9, 2015

Drawing Lines is Hard

Twitter: @mattdesl

Drawing lines might not sound like rocket science, but it's damn difficult to do well in OpenGL, particularly WebGL. Here I explore a few different techniques for 2D and 3D line rendering, and accompany each with a small canvas demo.

Source for demos can be found here:

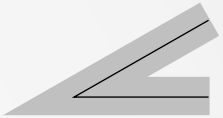
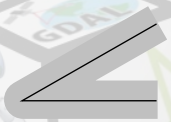
https://github.com/mattdesl/webgl-lines

Farkas Gábor
doktorandusz
PTE TTK FDI

Vonalat húzni nehéz

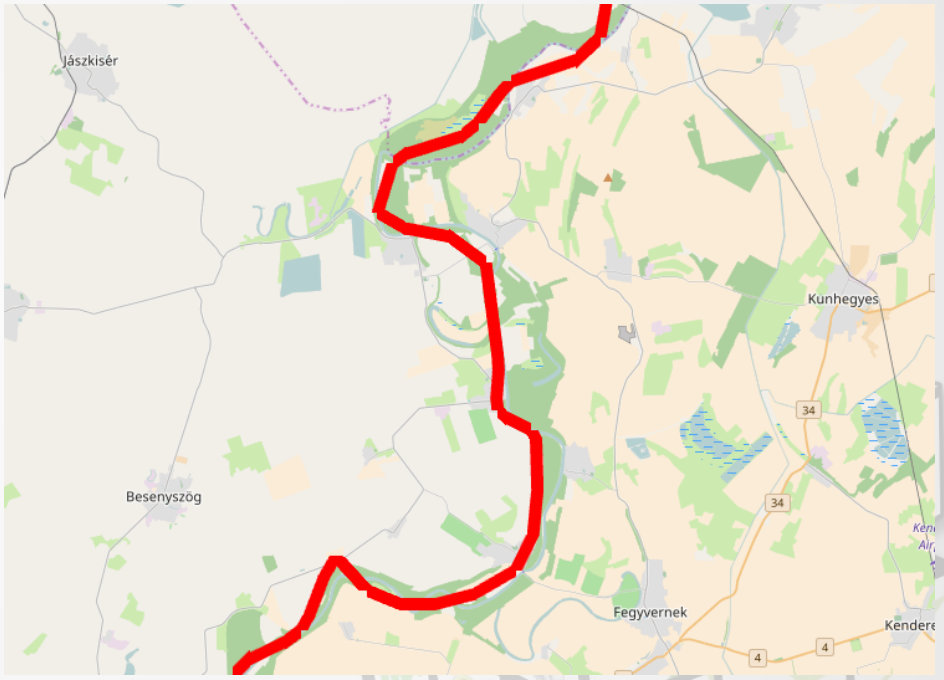
- A beépített vonal primitív (gl.LINES) korlátolt
- A vonal háromszögelése komplex és macerás
- A vonalak csatlakozási pontjára külön figyelmet kell fordítani
- 3D-ben nehéz megtartani a vonalvastagságot
- Geometry shader nem támogatott WebGL-ben

Vonalak csatlakozási pontja?

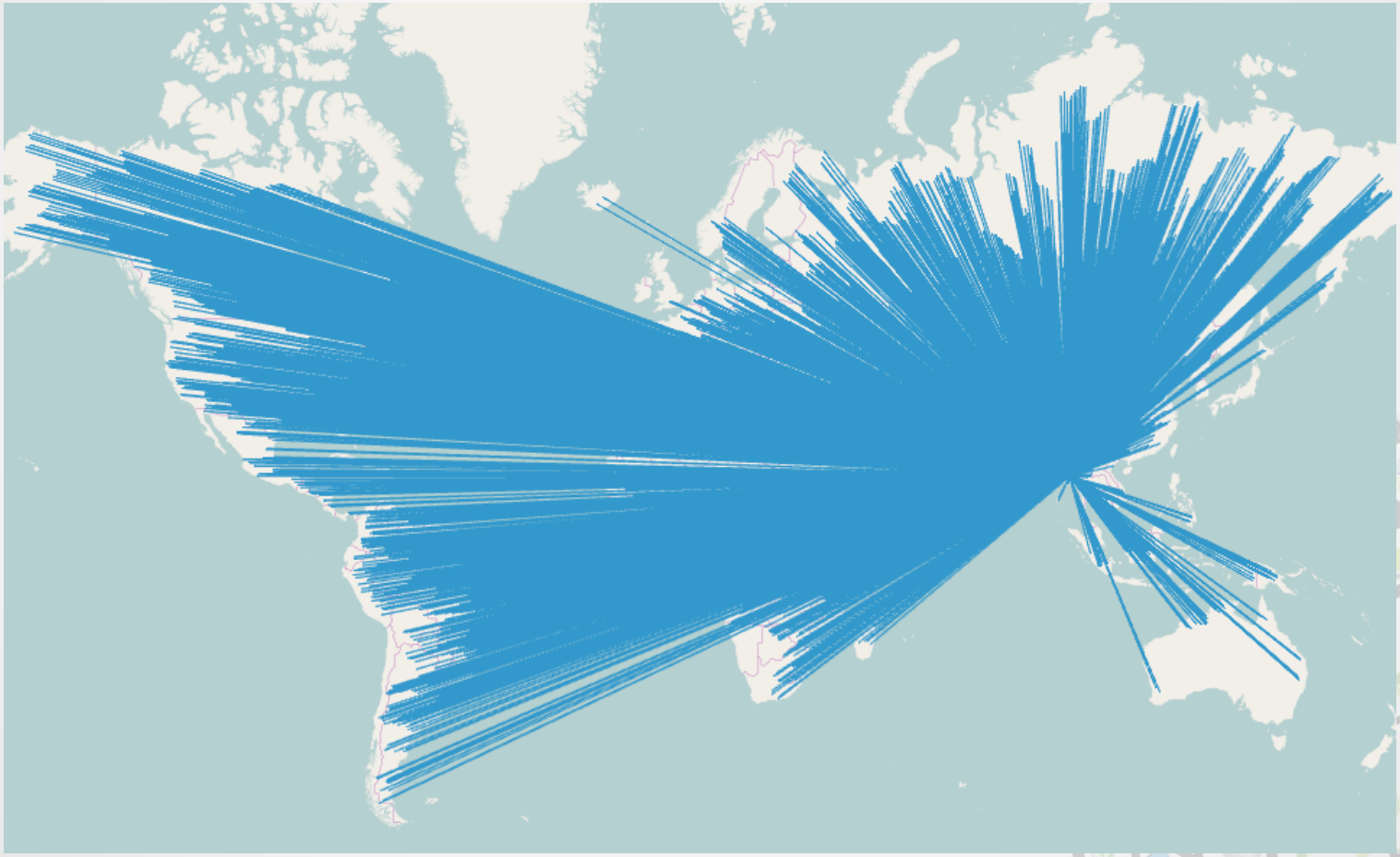
- Vonalvég lehet:
  - Square 
  - Butt 
  - Round 
- Vonalcsatlakozás lehet:
  - Miter 
  - Bevel 
  - Round 

gl.LINES

- Vonalvastagság 1-10 px és hardverfüggő
- Vagy vonalvastagság vagy csatlakozási pont biztosított
- Ha a csatlakozási pontot választjuk, a vonalvasagság 1 pixeles

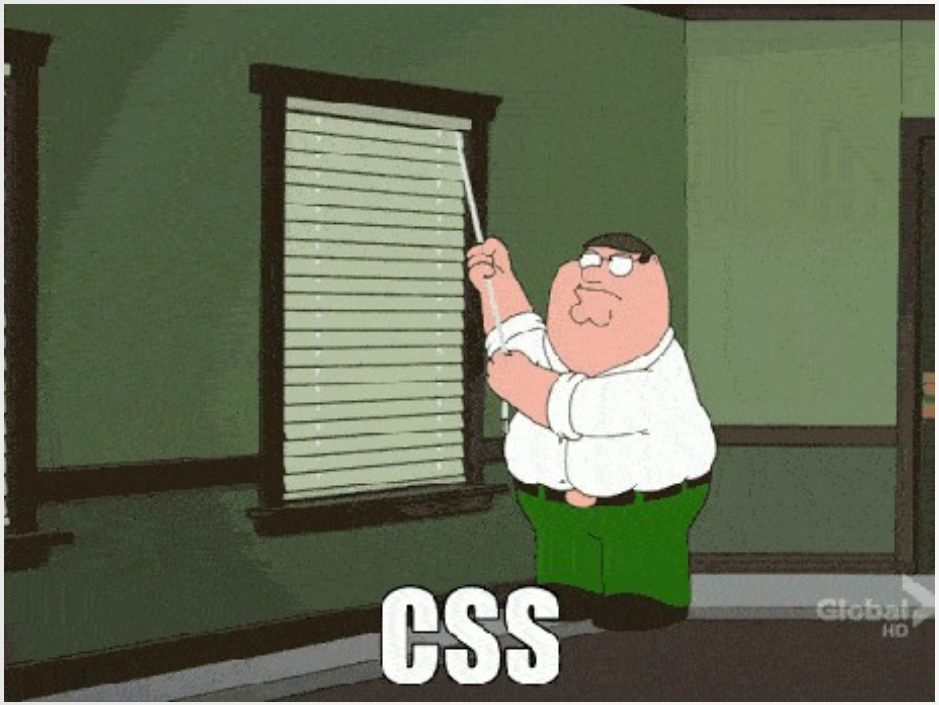


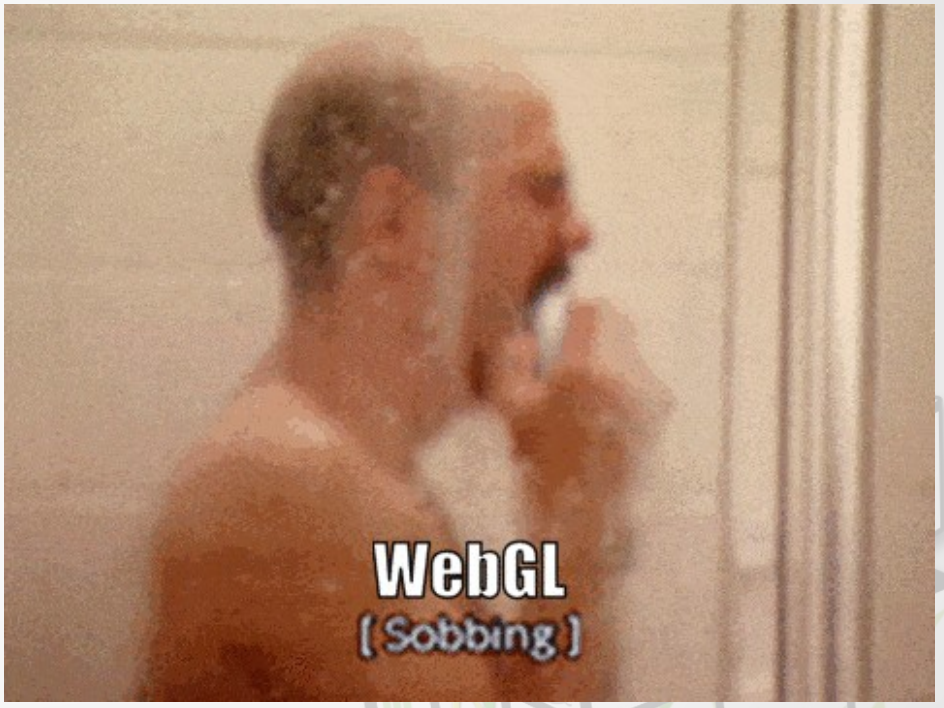
Majd megtanulom menet közben



Majd megtanulom menet közben

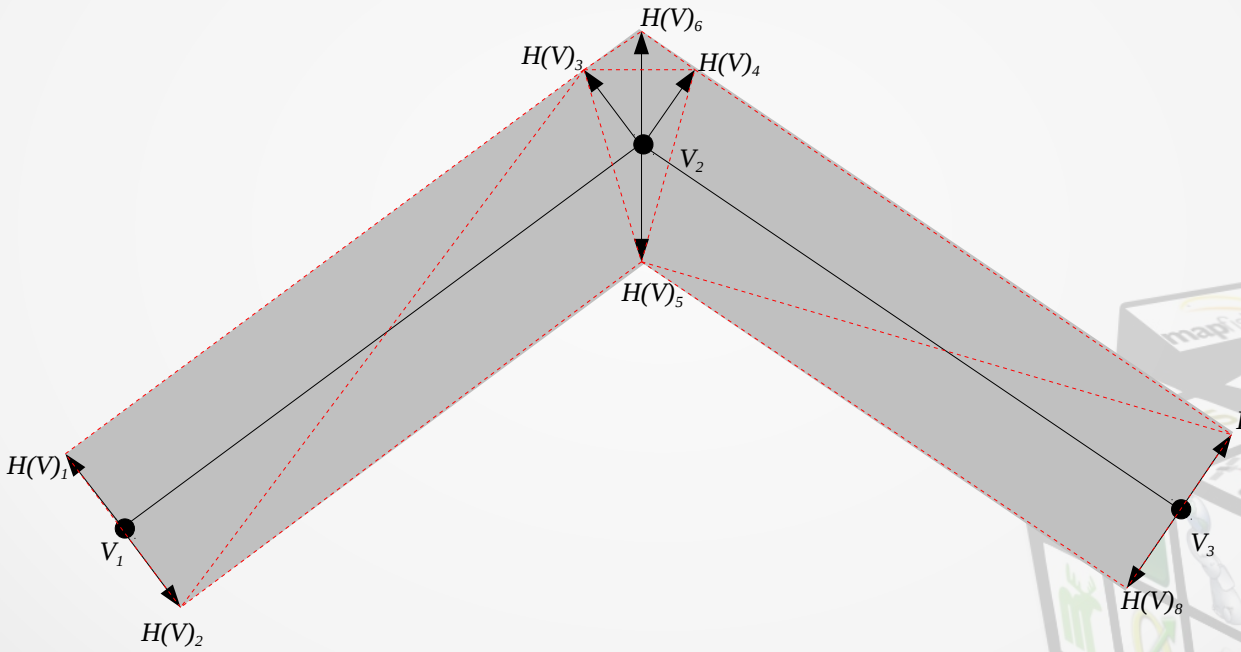
Vissza az iskolapadba





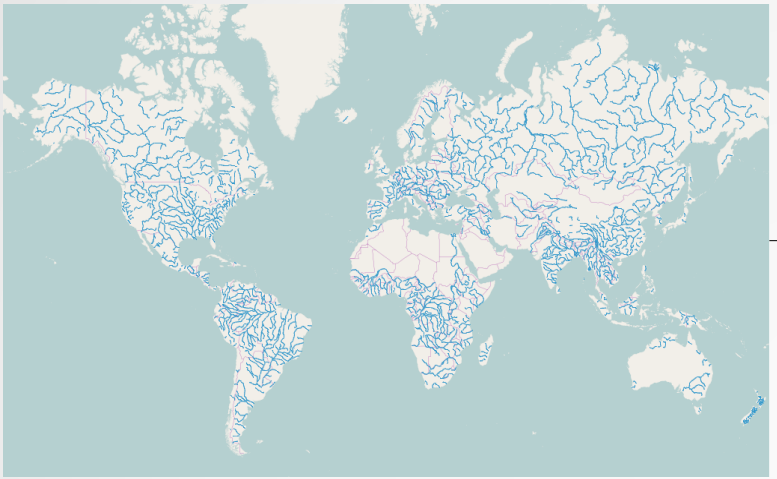
Vonallánc háromszögelése

- Pont úgy, ahogy pufferzónát alakítunk ki GIS-ben

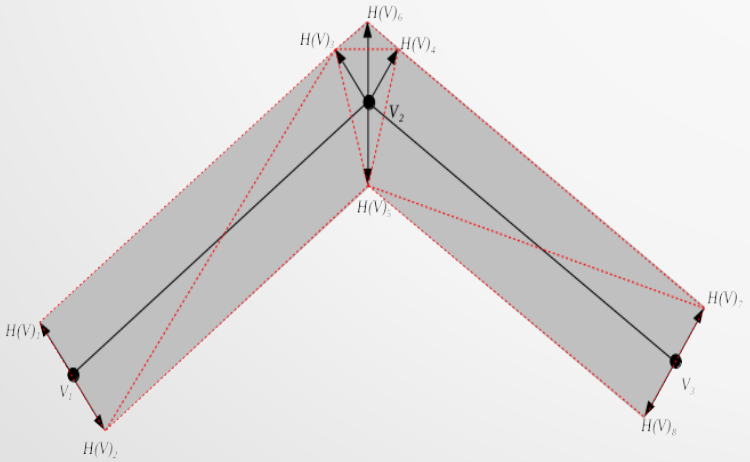


Vetületi rendszerek

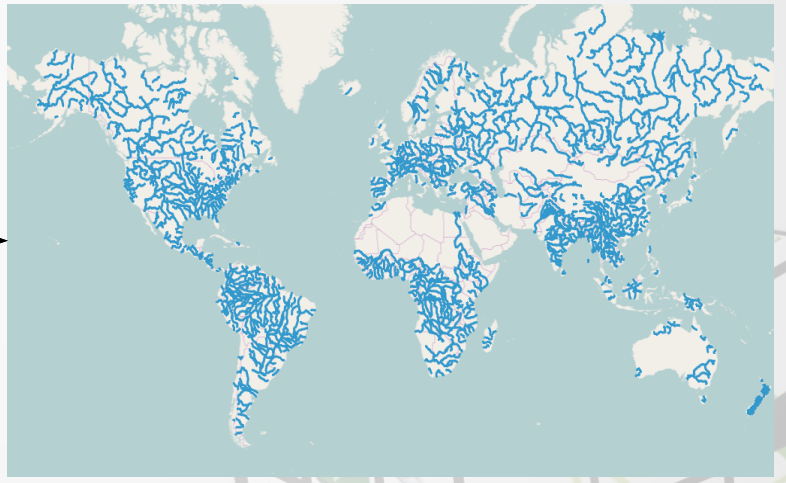
m



px



NDC



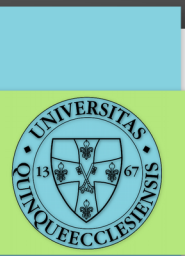
Mi kell ahhoz, hogy videókártyán fusson?

- Ami van:
  - Térképi vetület → NDC mátrix
  - Képernyő vetület → NDC mátrix
  - Koordináták, irányok, sorrend
- Ami nincs:
  - Geometry Shader
- Ami kell még:
  - Párhuzamosítható (állapot nélküli) algoritmus

Párhuzamosítható algoritmus

- Meghatározzuk a háromszögelés pontjait, anélkül, hogy kiszámolnánk azokat (sorrend)
- Ehhez a videókártyának meg kell kapnia minden pontra:
  - Az előző pont koordinátáit (ha van): vec2f
  - A jelenlegi pont koordinátáit: vec2f
  - A következő pont koordinátáit (ha van): vec2f
  - Az utasítást: float
  - Az irányt: float
  - A kerekítés szükségességét: float

Egy primitív tömörítés



- Érték = utasítás × irány × kerekítés
  - Utasítás csak prímszám lehet (3, 5, 7, 11, 13, 17, 19, 23)
  - Irány lehet: 1 vagy -1
  - Kerekítés szükségessége lehet: 1 vagy 2
- Így a videókártyán:
  - 0 maradék megadja az utasítást
  - Előjel az irányt
  - Paritás a kerekítés szükségességét

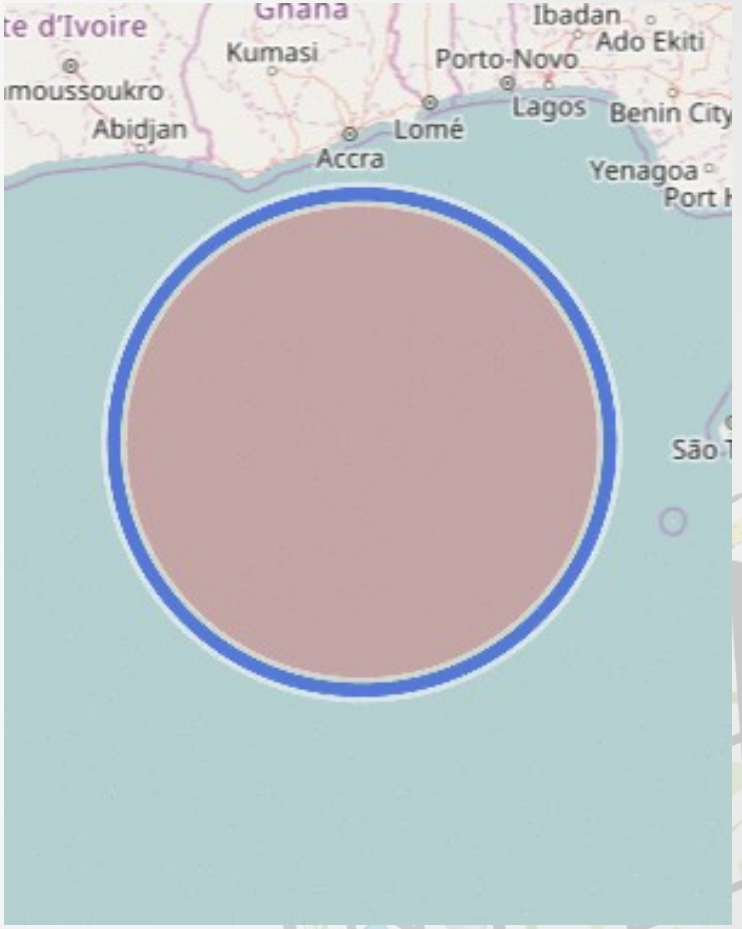
Amire érdemes figyelni: kerekítés

- EPSG:3857 - m-ben a világ
- Ha túl nagy a pontosság, a videókártya kerekít
- Megoldás: RTE!
- Cesium: Relative To Eye megjelenítés (Chris Thorne: Floating Origin alapján)

Amire érdemes figyelni: miter

- Felső ék könnyen limitálható maximum érték megadásával
- Nagyon hegyes szögnél azonban egy alsó ék is kialakul
- Megoldás: metszéspont ellenőrzés! Ha van, oda kell rakni az alsó éket.

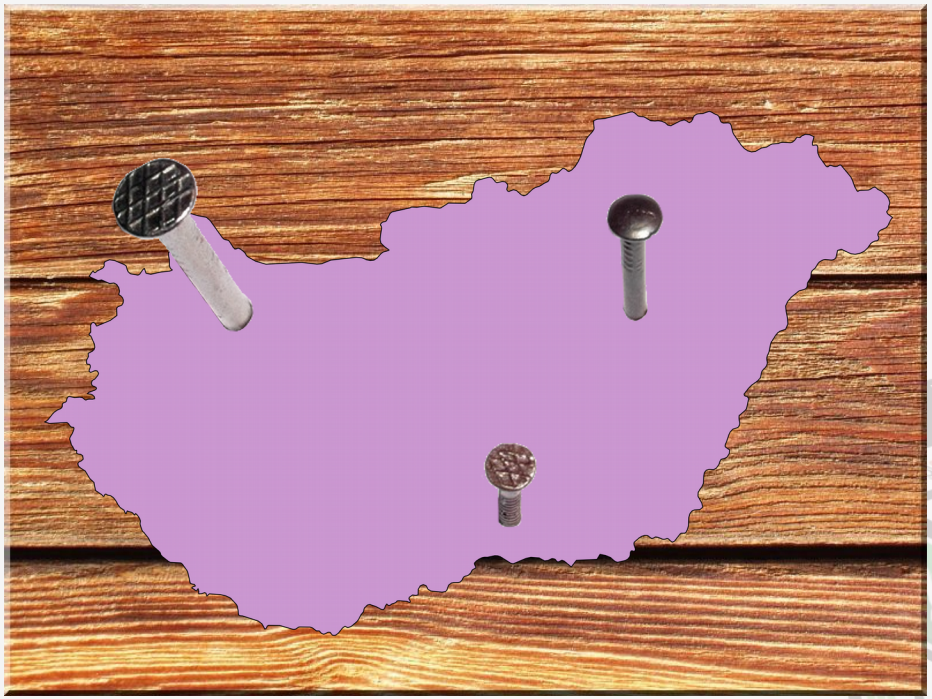
Amire érdemes figyelni: DPR



- Device Pixel Ratio, azaz mennyi fizikai pixel^2 alkot egy logikai pixelt
- HDPI készülékek, pl. Apple “retina” kijelzők, azóta egyre több fajta
- Megoldás: vegyük számításba a pixelarányt, amikor pixelben számolunk! (window.devicePixelRatio)

Vonalat húzni könnyű

Poligont háromszögelni nehéz!



Példa: https://gaborfarkas.github.io/FOSS4GHU_2016

Köszönöm a figyelmet!