Game Graphics & Real-time Rendering

CMPM 163, W2018

Prof. Angus Forbes (instructor)
angus@ucsc.edu

Lucas Ferreira (TA)
lferreira@ucsc.edu

creativecoding.soe.ucsc.edu/courses/cmpm163
github.com/CreativeCodingLab

Last week

- Looked at how to track mouse points in a fragment shader, and how to find the distance between a fragment and that point
- Showed how to use a texture as a height map in a vertex displacement shader
- Introduced Perlin noise to generate naturalistic dynamic textures and meshes
- Looked at how to use point sprites in Three.js and to texture them using a fragment shader
- Introduced particle systems & the dat.gui library
- https://creativecoding.soe.ucsc.edu/courses/cmpm163/code/week3_codeExamples.zip
- https://creativecoding.soe.ucsc.edu/courses/cmpm163/code/week4_codeExamples.zip

This week

- Homework #2 introduced – Due Feb. 18th at 12noon

- Voronoi cells, redux
- Drawing 2D shapes using signed distance functions (SDFs)
- Drawing nice looking text using SDFs
- Raymarching 3D objects
- Shading 3D SDFs
- Morphing between SDFs
- https://creativecoding.soe.ucsc.edu/courses/cmpm163/code/week5_codeExamples.zip

Homework 2

- Homework #2 introduced – Due Feb. 18th at 12noon

- Design two scenes:

  1) Outdoor scene using cube map, height map, textures
  2) Abstract scene using noise functions and particle systems

- Some points given for creativity, beauty, composition.
- Extra credit given for additional functionality (textured point sprites, reflection+refractive effects)

This week

- Homework #2 introduced – Due Feb. 18th at 12noon

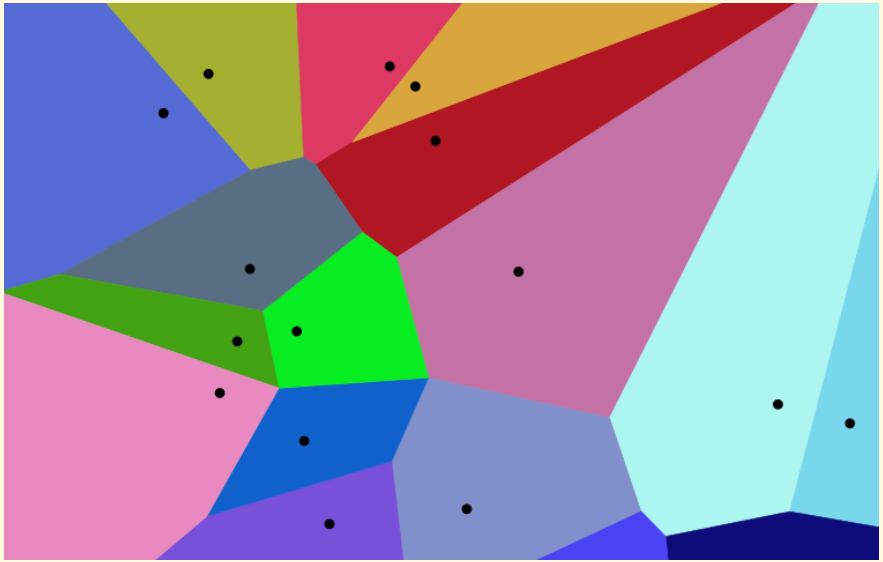
- Voronoi cells, redux
- Drawing 2D shapes using signed distance functions (SDFs)
- Drawing nice looking text using SDFs

- Midterm review

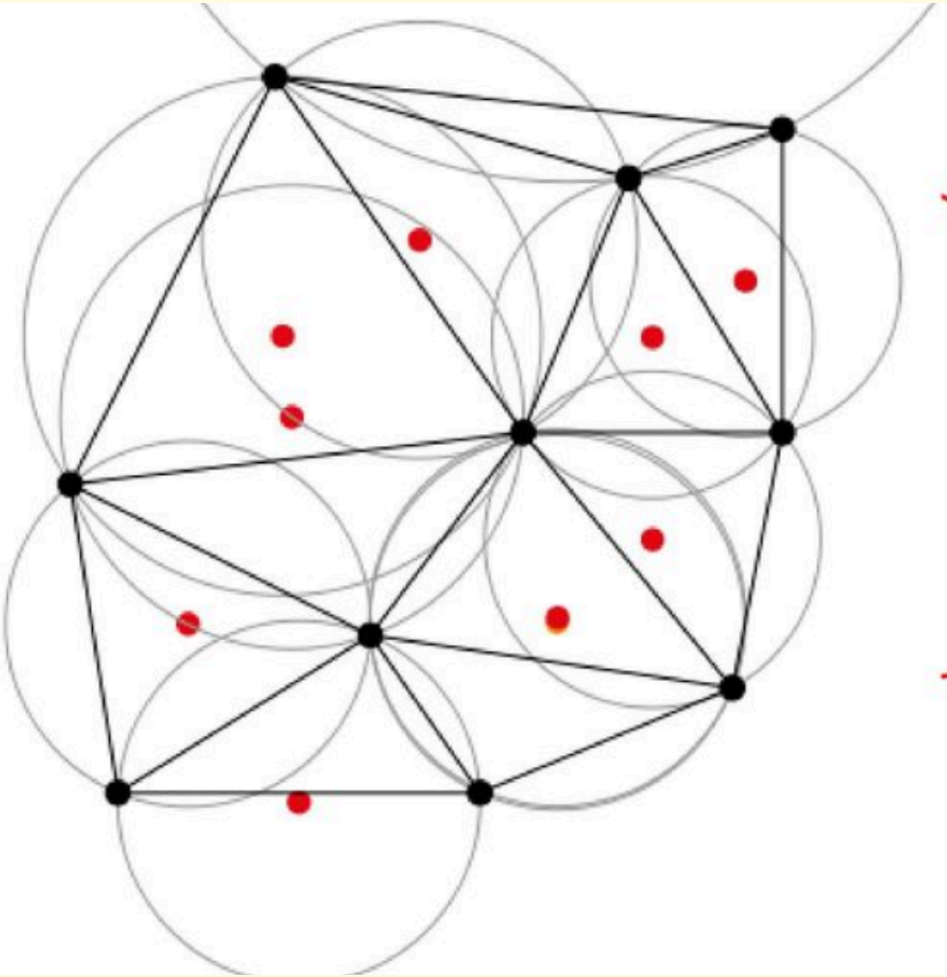
- https://creativecoding.soe.ucsc.edu/courses/cmpm163/code/week5_codeExamples.zip

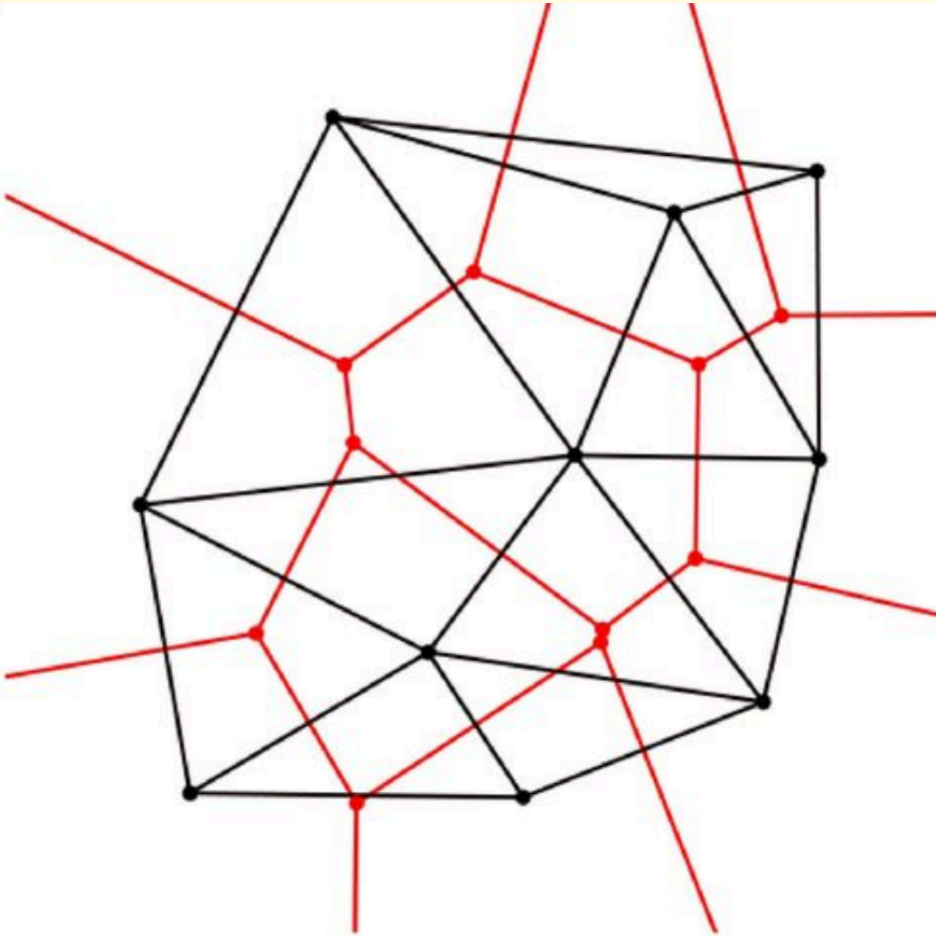
Voronoi tesselation

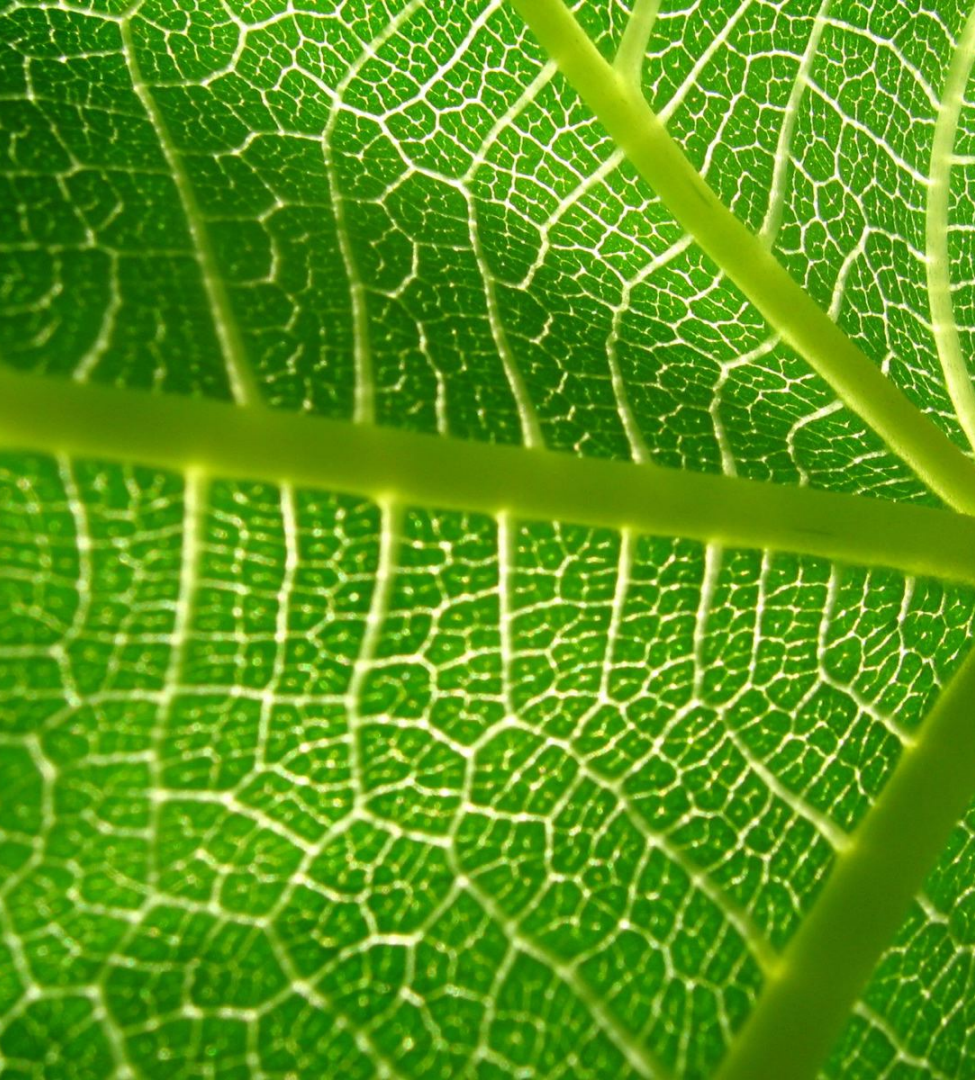
Given a set of points $\{p_1, \ldots, p_n\}$ on a 2D plane, a Voronoi cell is defined for each point p_k where the distance to p_k is less than the distance to any other cell p_j.

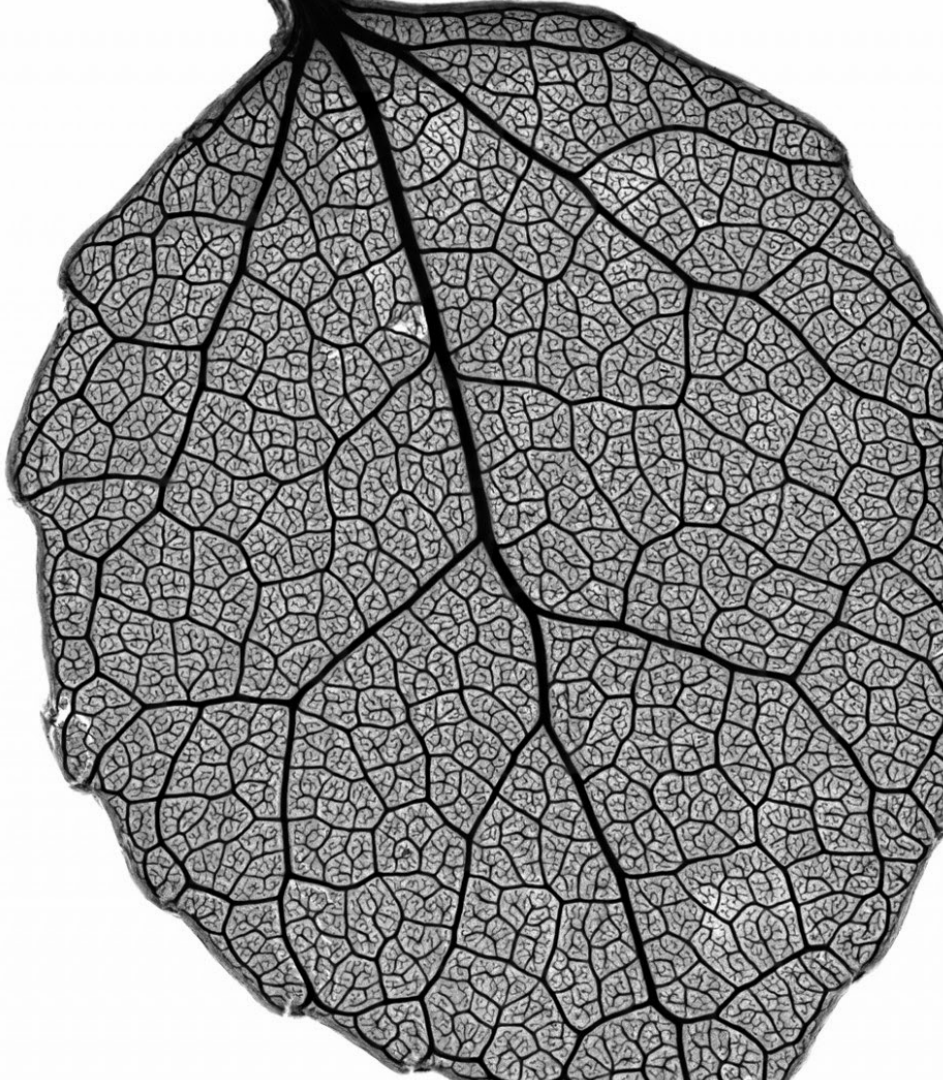


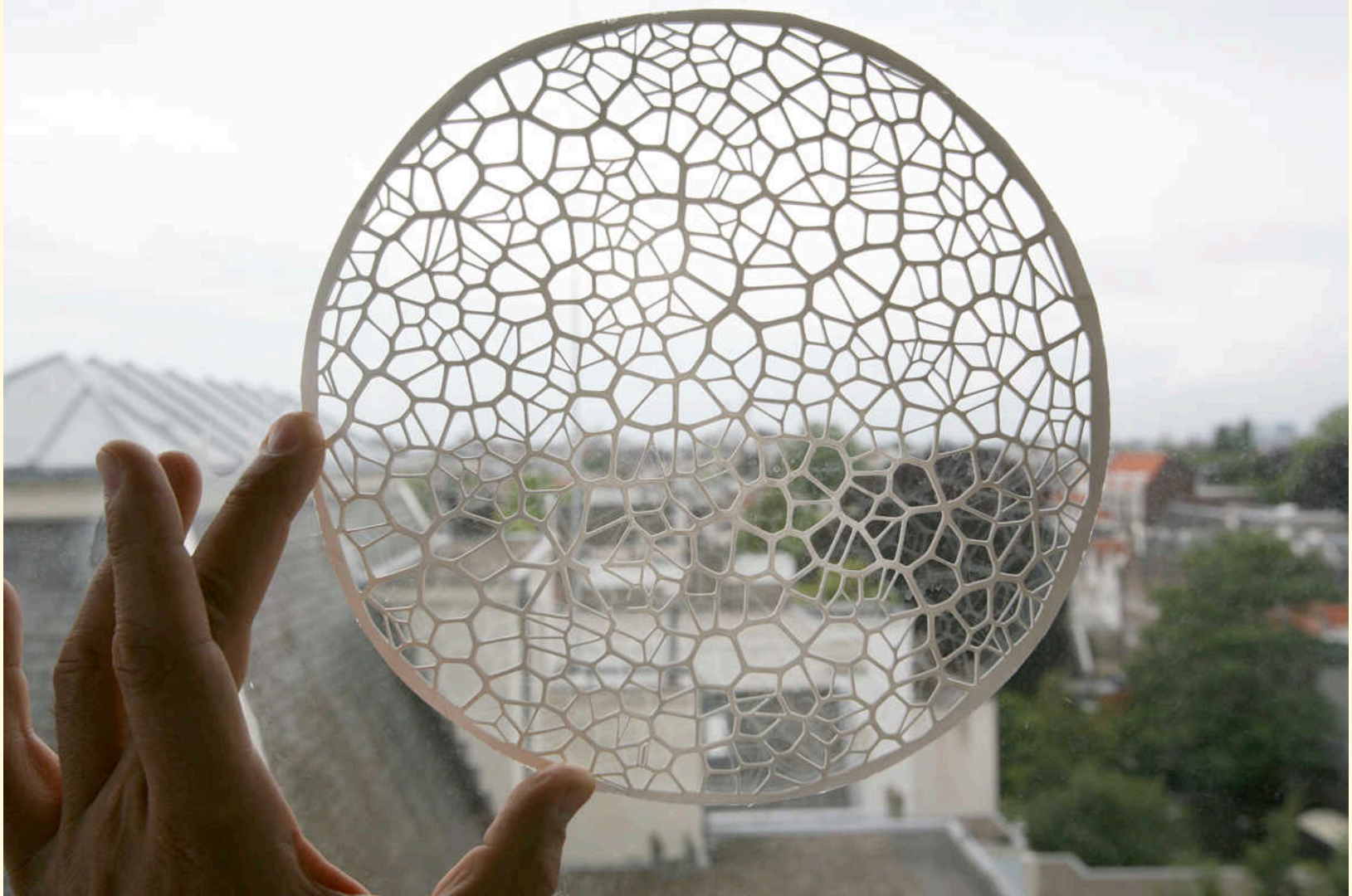
$$R_k = \{x \in X \mid d(x, P_k) \leq d(x, P_j) \text{ for all } j \neq k\}$$

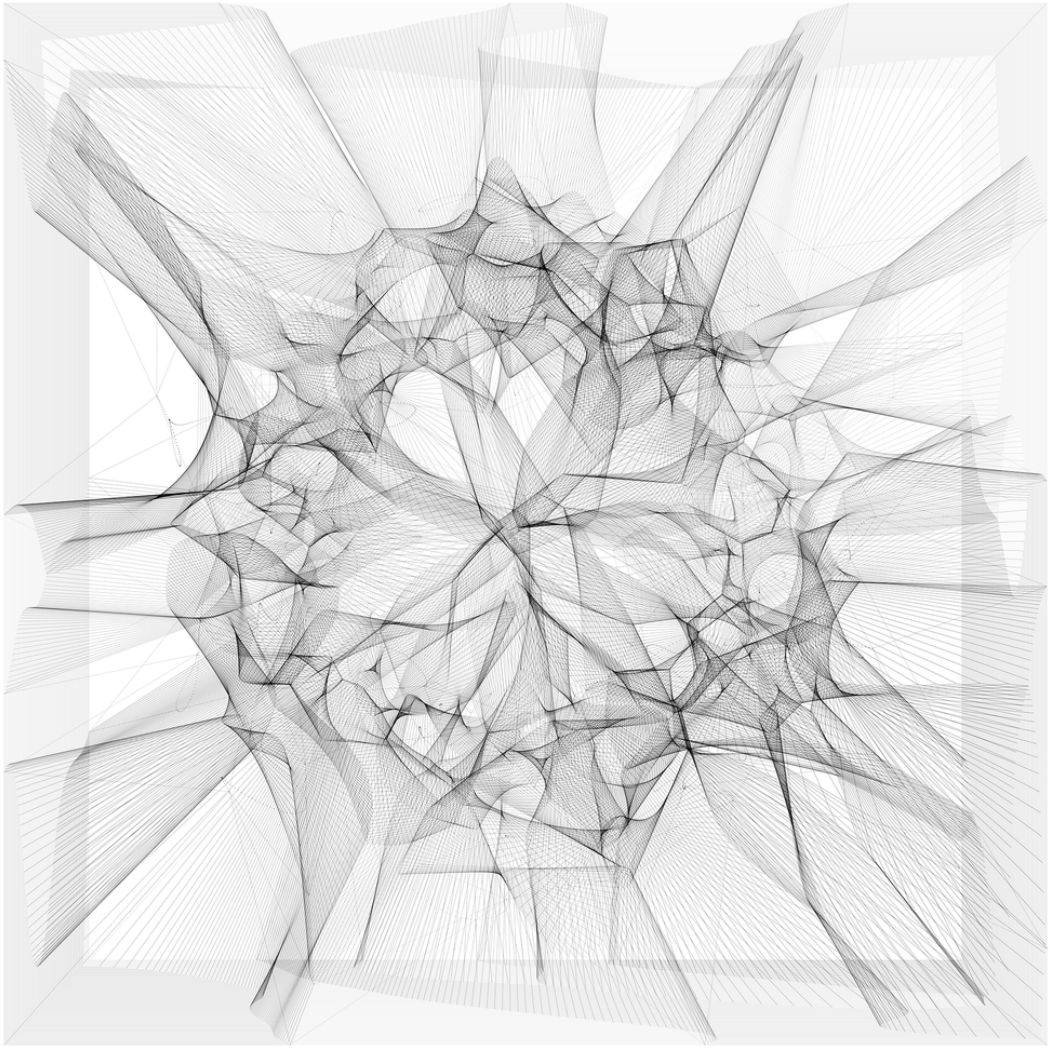


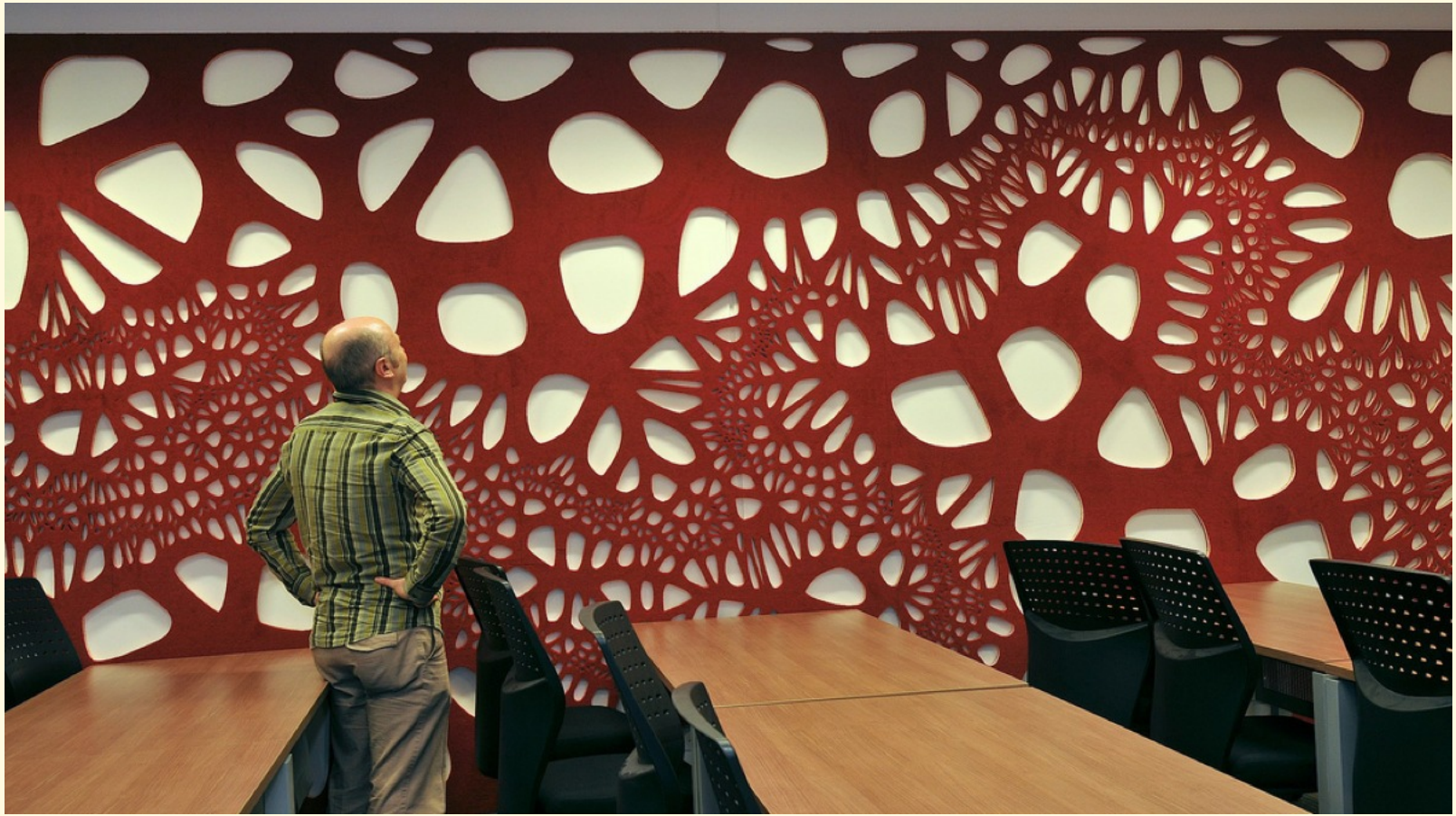












Voronoi tesselation

- In fragment shader, we are able to determine the geometry *without* passing in any vertices to the vertex shader (other than those that define the fullscreen quad).

- That is, we let each pixel in the fragment shader determine whether or not it is part of a shape defined by some function. (see iquilezles.org/www/articles/voronoilines/voronoilines.htm for a discussion on how to calculate cell borders accurately.)

- We can extend this idea to draw arbitrary shapes: https://www.shadertoy.com/view/4dfXDn

Signed distance functions (SDF)

```
float circleDist(vec2 p, float radius) {
    return length(p) - radius;
}
```

Ex:

```
p = vec2(200.0,0.0);
radius = 100.0;

return value > 0.0; //is outside the circle
```

Signed distance functions (SDF)

```
float circleDist(vec2 p, float radius) {
    return length(p) - radius;
}
```

Ex:

```
p = vec2(25.0,25.0);
radius = 40.0;

return value < 0.0; //is inside the circle
```

Signed distance functions (SDF)

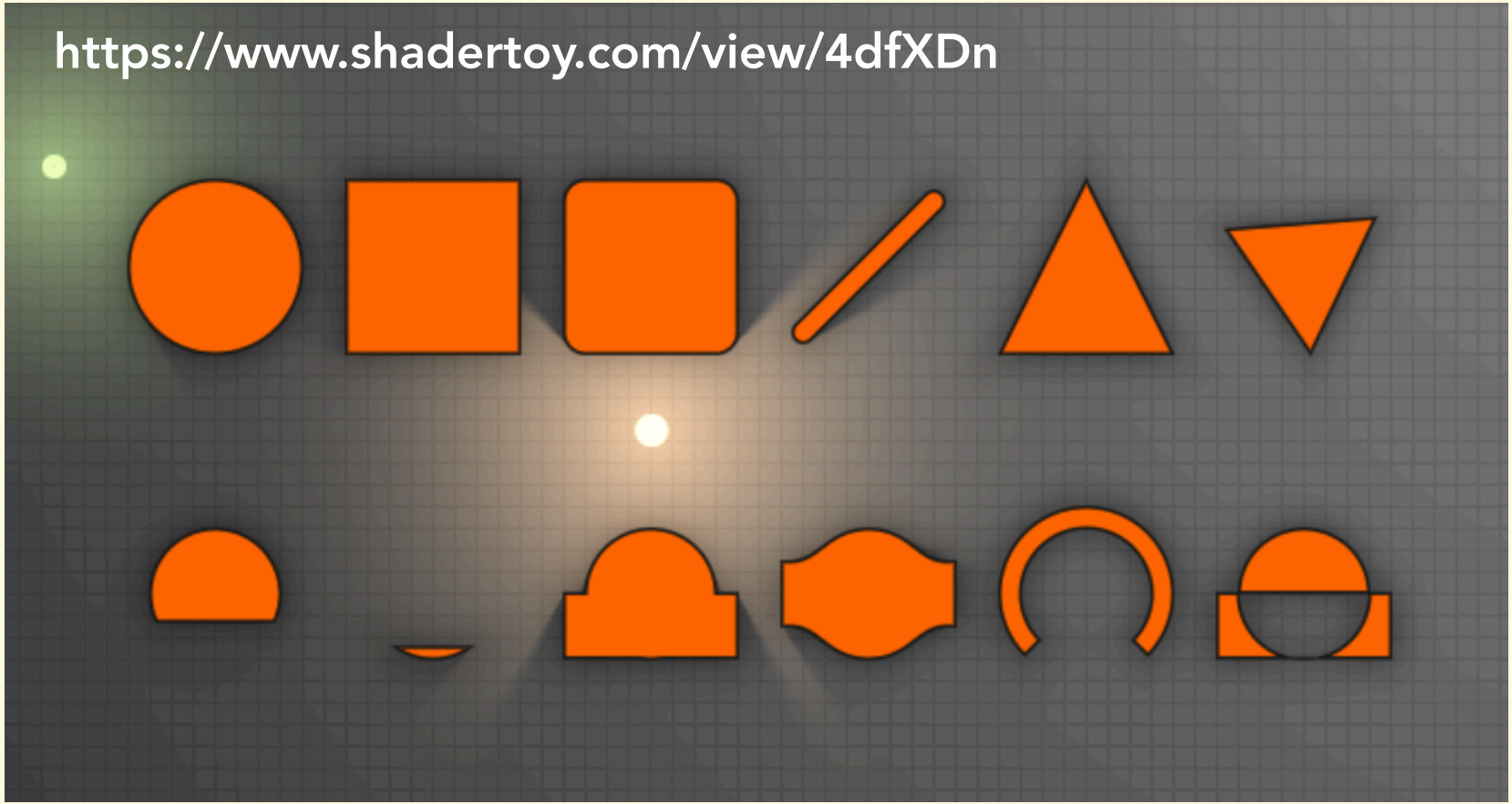
```
float circleDist(vec2 p, float radius) {
    return length(p) - radius;
}
```

Ex:

```
p = vec2(0.0,-100.0);
radius = 100.0;

return value = 0.0; //is on the circle's edge
```

2D Signed distance functions (SDF)



Different ways of rendering text

HTML: overlay a div on top of the webGL canvas and draw with normal html

- Can't do anything super fancy with the text, not part of OpenGL pipeline

Font atlas: store each letter onto an image, keep track of the texture coords for each letter, use those texture coords to decal a rectangle

- Good for overlaying on top of scene, can use color info to find edges or update colors inside or outside of text
- Looks blurry if zooming in and out

Different ways of rendering text

SDF atlas: Using the vector information for the characters in the font, generate a SDF for each letter, such that any pixel inside the font outline is positive and distance falls off for pixels outside the font outline.

- Looks good at a much wider range of sizes
- Can use distance information to create outlines of various widths
- Can use distance information to define more sophisticated texturing strategies
- Not 100% perfect - Can still create some artifacts if too large/small

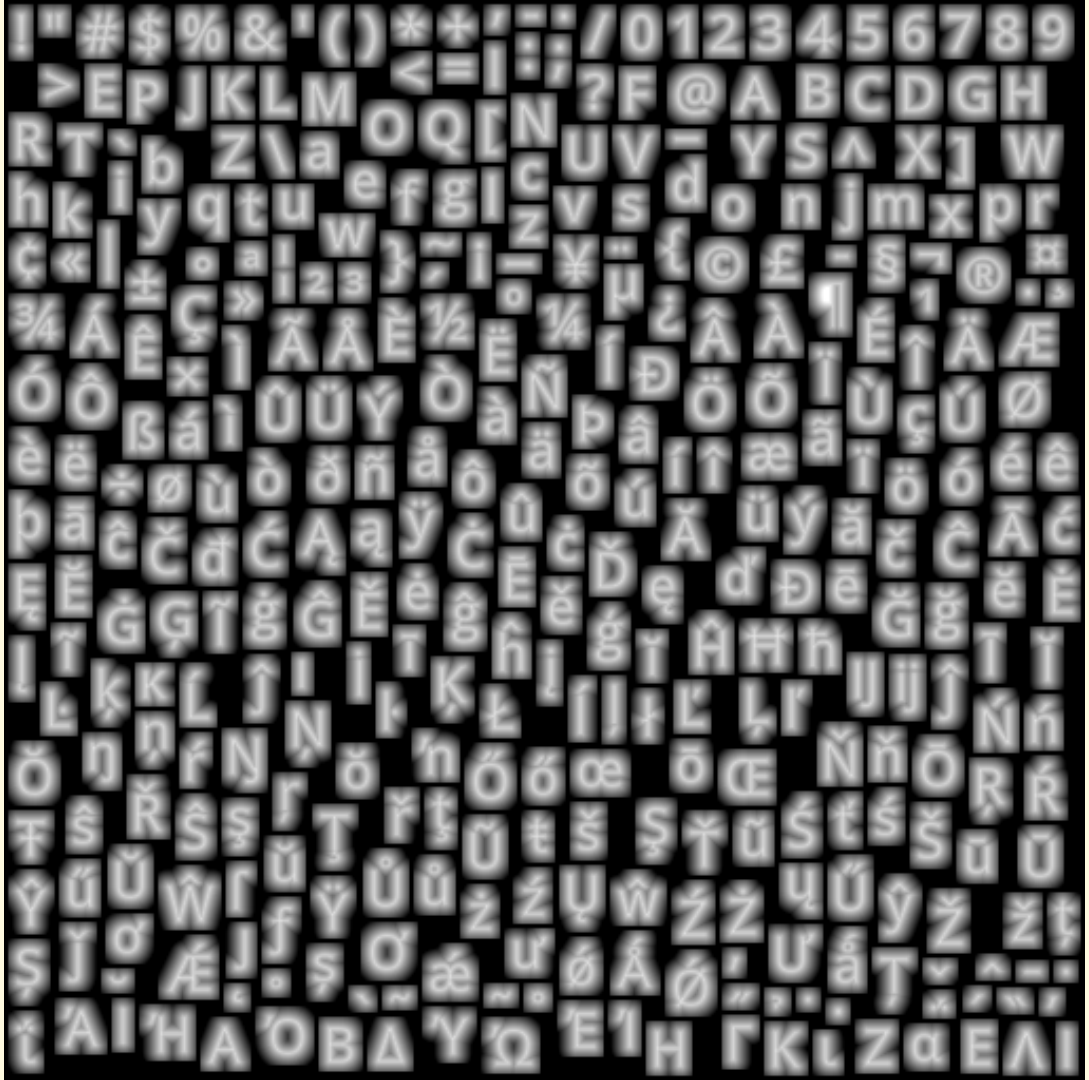
Different ways of rendering text

Vector atlas: Using the mathematical functions (bezier curves + lines) that define each character in the font, generate a texture that stores all of this data, as well as a complex indexing scheme to access these functions.

- Looks PDF perfect
- Handles all vector graphics, emojis, SVGs, etc.
- Complex to get right
- Only solution is commercial (http://sluglibrary.com), doesn't have a Javascript/WebGL version yet (although there is a published paper about the technique which could probably be re-implemented for the web)

Font atlas

SDF atlas



SDF for rendering text

Overview:

https://blog.mapbox.com/drawing-text-with-signed-distance-fields-in-mapbox-gl-b0933af6f817

Demo:

https://mapbox.s3.amazonaws.com/kkaefer/sdf/index.html

Source:

view-source:https://mapbox.s3.amazonaws.com/kkaefer/sdf/index.html

3D Signed distance functions (SDF)

3D Signed distance functions (SDF)

Midterm review

Concepts:

- Rendering pipeline
- Lighting
- Texturing
- Render-to-texture

Midterm review

Code:

- Vertex & fragment shaders
- GLSL data types (attribute, uniform, varying)
- Environment mapping (reflection)
- Offscreen buffers
- Vertex displacement
- Point sprites
- Height maps

Next week

- Midterm on Tuesday!

- Raymarching 3D objects
- Shading 3D SDFs
- Morphing between SDFs

Game Graphics & Real-time Rendering

CMPM 163, W2018

Prof. Angus Forbes (instructor)
angus@ucsc.edu

Lucas Ferreira (TA)
lferreira@ucsc.edu

creativecoding.soe.ucsc.edu/courses/cmpm163
github.com/CreativeCodingLab