



Computer Vision

(Course Code: 4047)

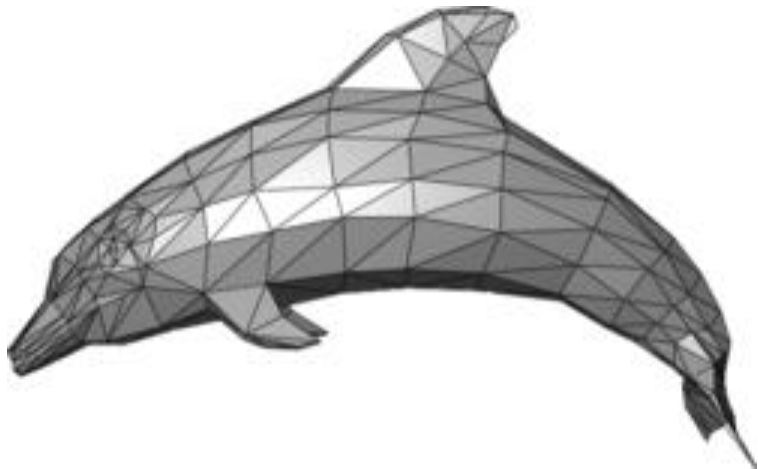
Module-6: 3D Reconstruction

Lecture-2: 3D Surface Representations

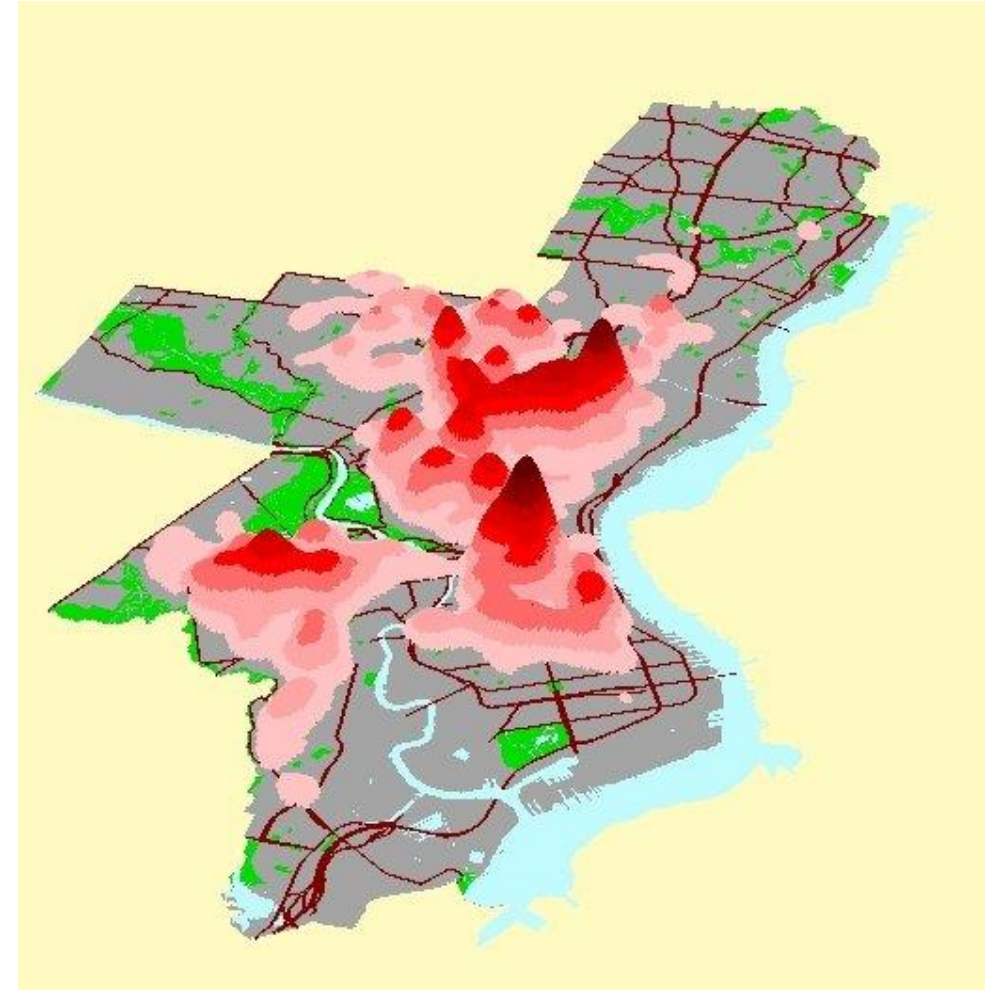
Gundimeda Venugopal, Professor of Practice, SCOPE

Surfaces

- ❖ Surfaces involve a third 'z' dimension (height/elevation/magnitude, quantity) in addition to x,y planimetric location.
- ❖ Any type of continuous data can be represented as a surface, whether it be ground elevation, barometric pressure, rainfall, crop yield, noise levels, population density, sales intensity, land value, income, crime rates, etc.



Example of a [low poly triangle mesh](#) representing a [dolphin](#)



3D Model of Crime Density in Boston, MA
http://gis.mit.edu/classes/11.521/lectures/Lecture_14/Lect14.htm

How we can describe Surface Geometry?

IMPLICIT

$$x^2 + y^2 = 1$$

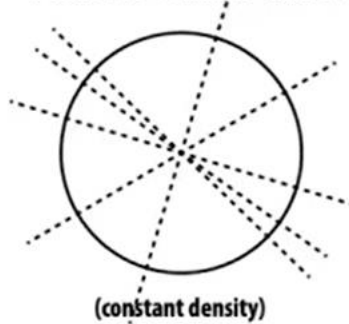
LINGUISTIC

"unit circle"

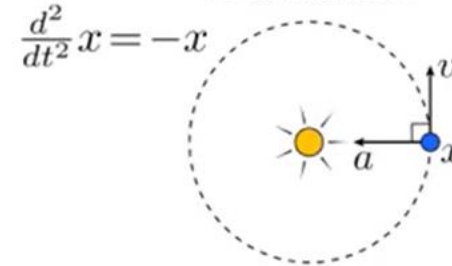
EXPLICIT

$$\underbrace{(\cos \theta)}_x, \underbrace{(\sin \theta)}_y$$

TOMOGRAPHIC



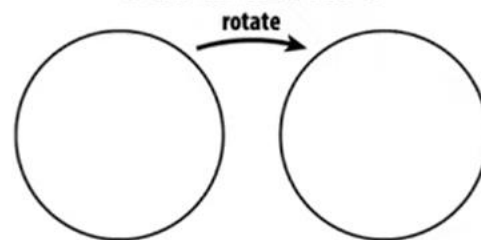
DYNAMIC

$$\frac{d^2}{dt^2} x = -x$$


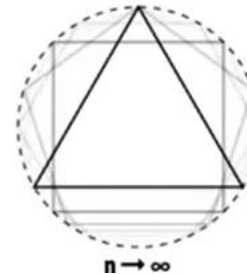
CURVATURE

$$\kappa = 1$$

SYMMETRIC



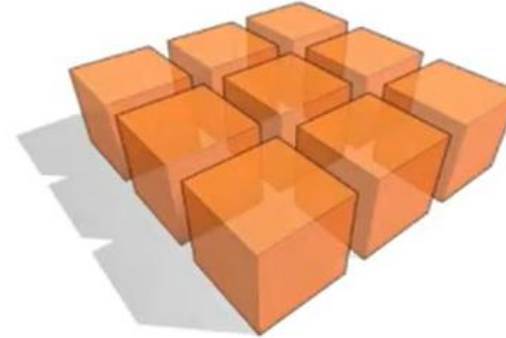
DISCRETE



Many ways to digitally encode geometry

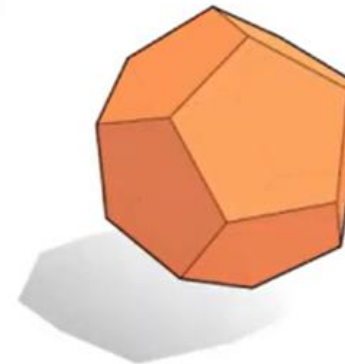
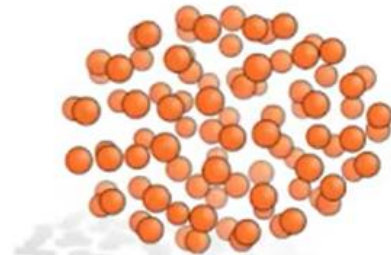
■ EXPLICIT

- point cloud
- polygon mesh
- subdivision, NURBS
- ...



■ IMPLICIT

- level set
- algebraic surface
- L-systems
- ...

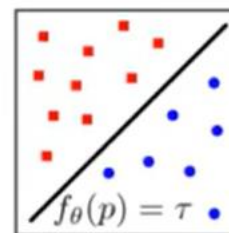
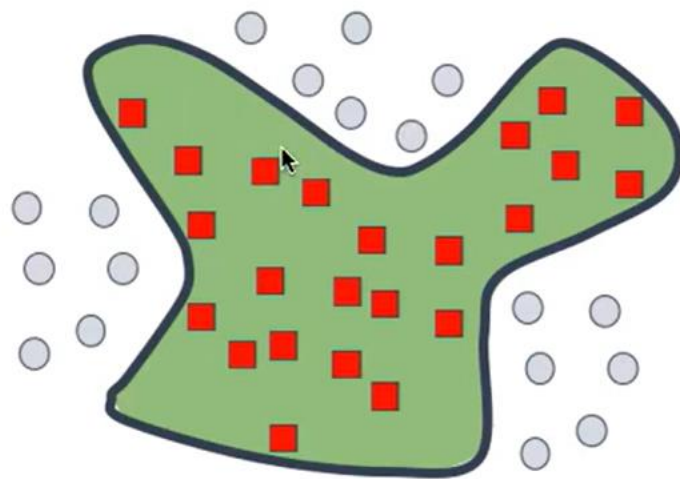


■ Each choice best suited to a different task/type of geometry

Surfaces as an Implicit Function

$$\mathbf{p} = (x, y, z) \in \mathbb{R}^3$$

$$f(\mathbf{p}) = \begin{cases} 0, & \text{if } \mathbf{p} \in \text{outside } \circ \\ 1, & \text{if } \mathbf{p} \in \text{inside } \blacksquare \end{cases}$$



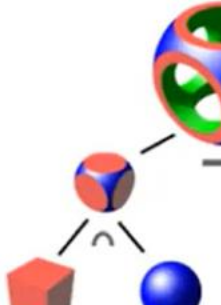
$$\mathcal{S} = \{\mathbf{p}, f(\mathbf{p}) = \tau\}$$

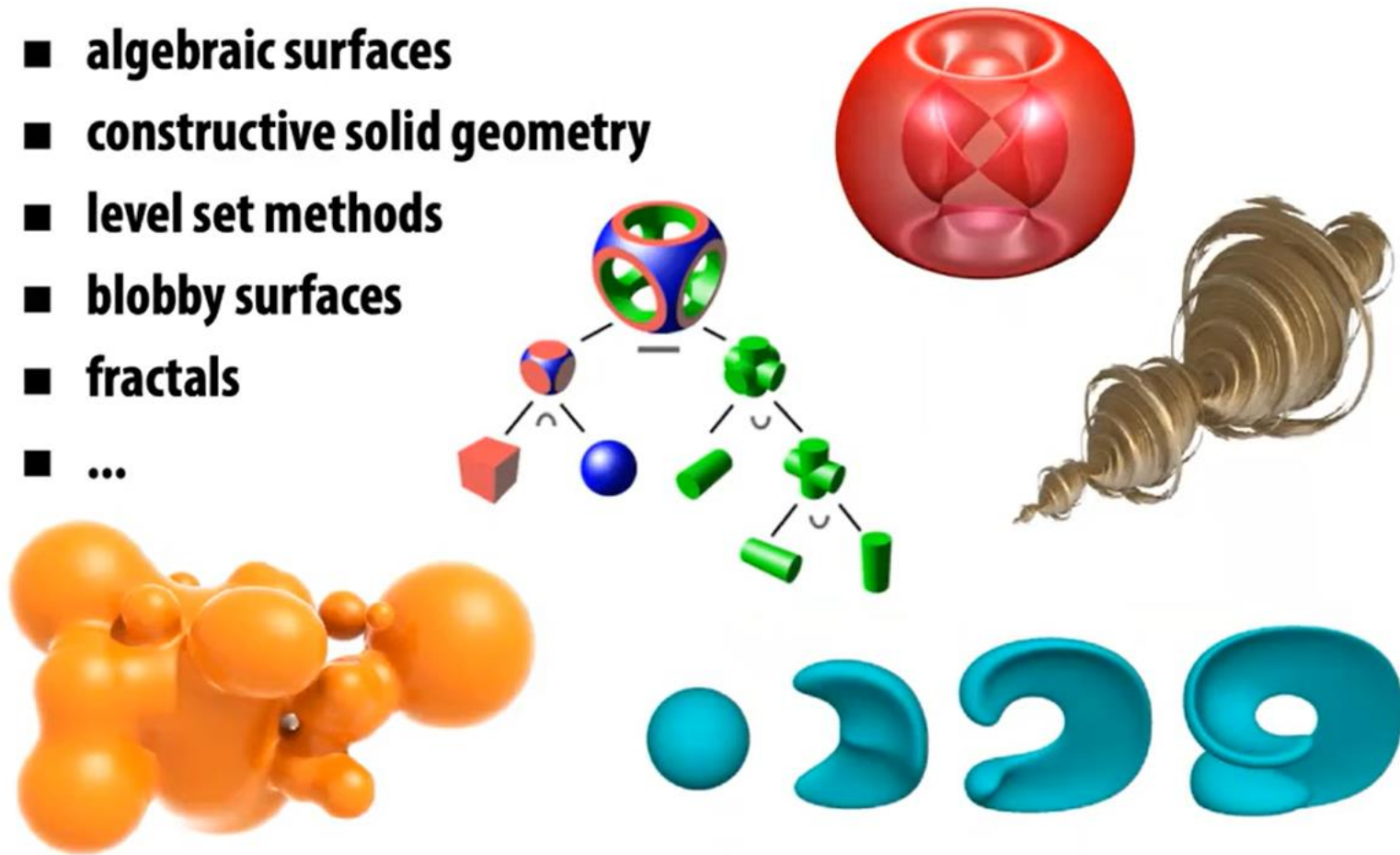
Implicit Representation of Surface Geometry

- Points aren't known directly, but satisfy some relationship
- E.g., unit sphere is all points such that $x^2+y^2+z^2=1$
- More generally, $f(x,y,z) = 0$

$f(x,y)$

Many Implicit Representation in Graphics

- algebraic surfaces
 - constructive solid geometry
 - level set methods
 - blobby surfaces
 - fractals
 - ...
- 



(Will see some of these a bit later.)

Check if the point is inside the unit sphere

I have a new surface $f(x,y,z) = x^2 + y^2 + z^2 - 1$.

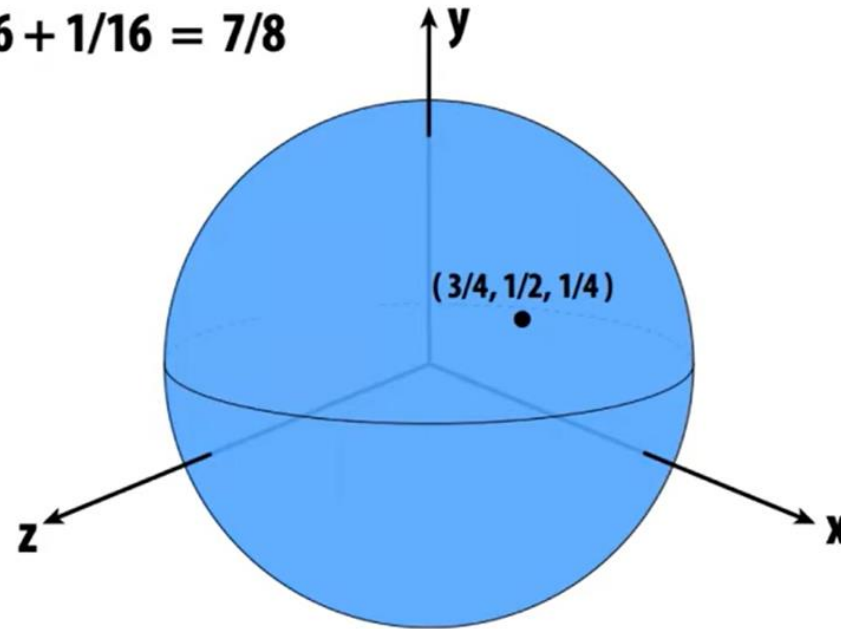
I want to see if a point is *inside* it.

How about the point $(3/4, 1/2, 1/4)$?

$$9/16 + 4/16 + 1/16 = 7/8$$

$$7/8 < 1$$

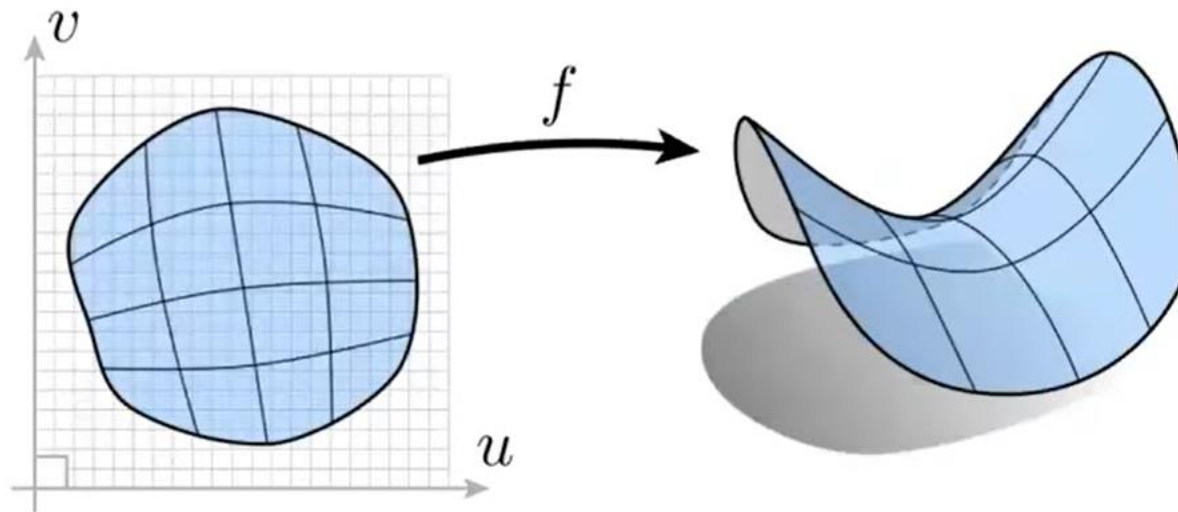
YES.



Implicit surfaces make other tasks easy (like inside/outside tests).

Explicit Representations of the Surface Geometry

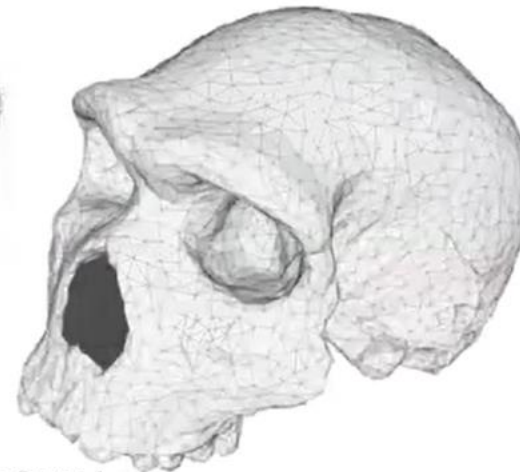
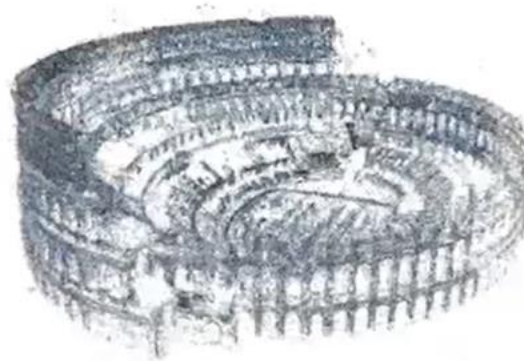
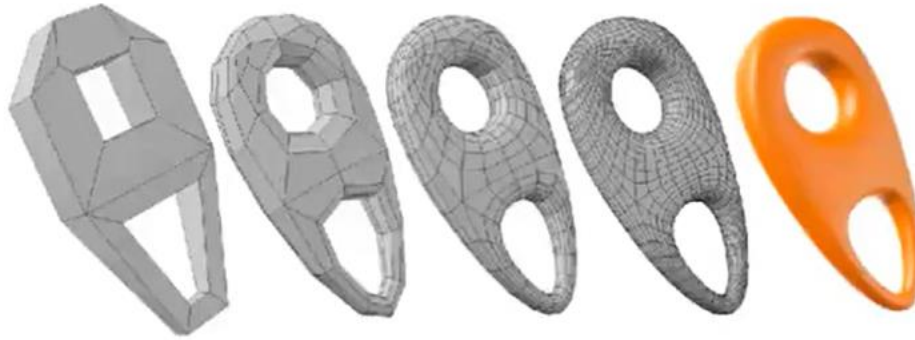
- All points are given directly
- E.g., points on sphere are $(\cos(u) \sin(v), \sin(u) \sin(v), \cos(v))$,
for $0 \leq u < 2\pi$ and $0 \leq v \leq \pi$
- More generally: $f : \mathbb{R}^2 \rightarrow \mathbb{R}^3; (u, v) \mapsto (x, y, z)$



- (Might have a bunch of these maps, e.g., one per triangle!)

Many explicit representations in Graphics

- **triangle meshes**
- **polygon meshes**
- **subdivision surfaces**
- **NURBS**
- **point clouds**
- **...**

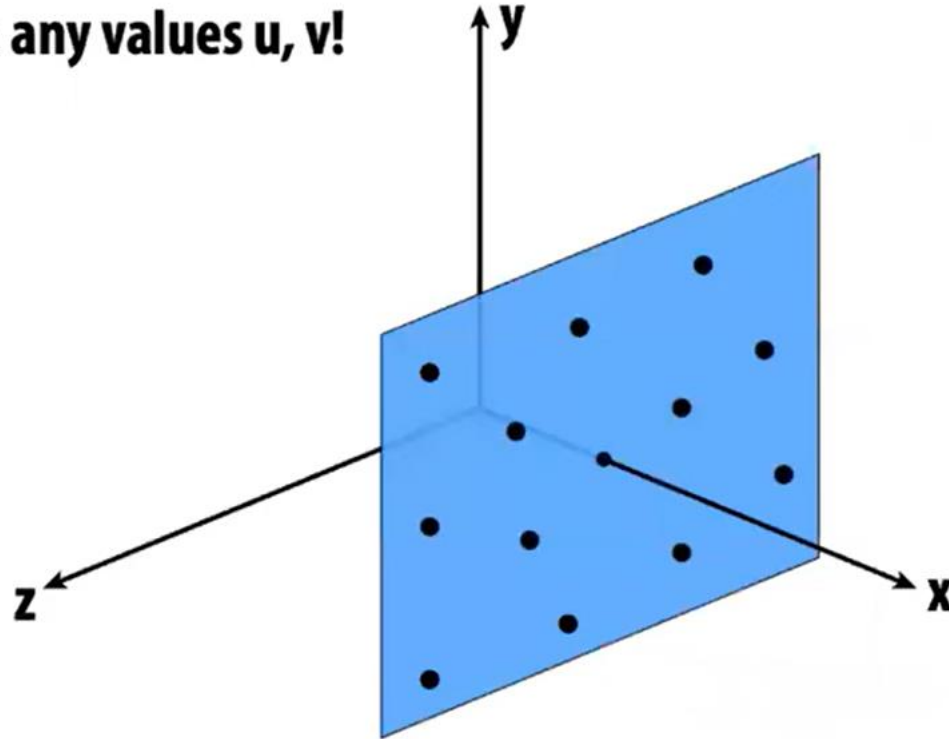


(Will see some of these a bit later.)

Sampling an explicit Surface

My surface is $f(u, v) = (1.23, u, v)$.

Just plug in any values u, v !



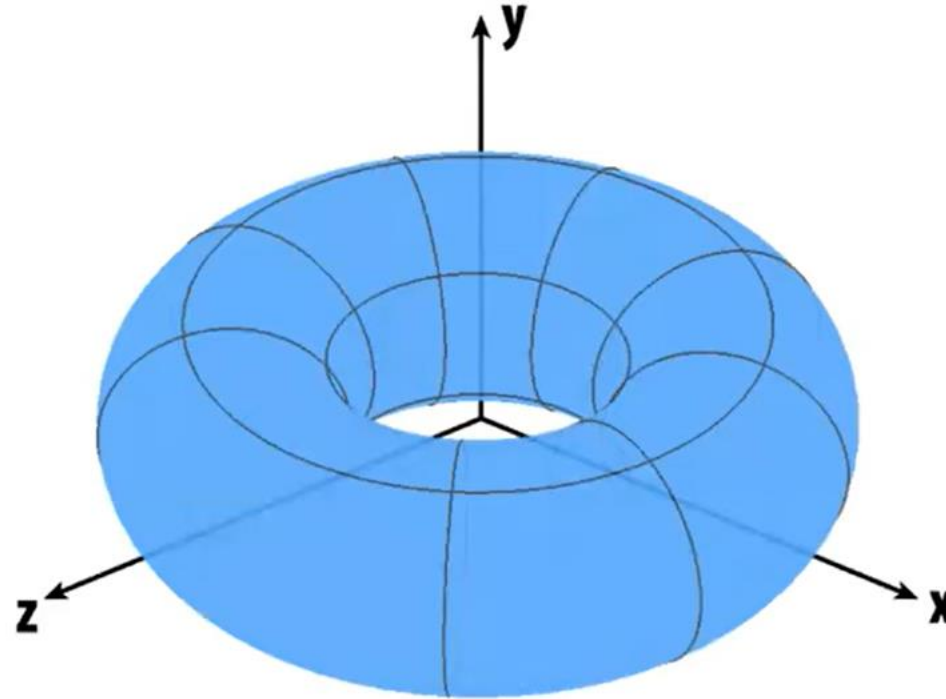
Explicit surfaces make some tasks easy (like sampling).

Explicit Functions: Check if this point inside the Torus

My surface is $f(u,v) = ((2+\cos u)\cos v, (2+\cos u)\sin v, \sin u)$

How about the point $(1.96, -0.39, 0.9)$?

...NO!



Explicit surfaces make other tasks hard (like inside/outside tests).

Algebraic Surfaces (Implicit)

- Surface is zero set of a polynomial in x, y, z

- Examples:



$$x^2 + y^2 + z^2 = 1$$



$$(R - \sqrt{x^2 + y^2})^2 + z^2 = r^2$$



$$\left(x^2 + \frac{9y^2}{4} + z^2 - 1\right)^3 = x^2 z^3 + \frac{9y^2 z^3}{80}$$

- What about more complicated shapes?

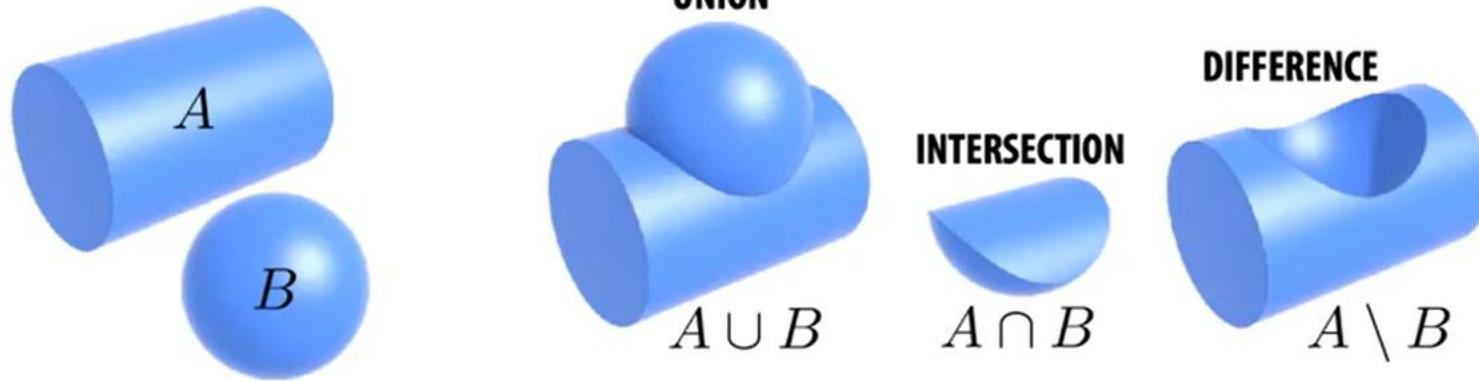


- Very hard to come up with polynomials!

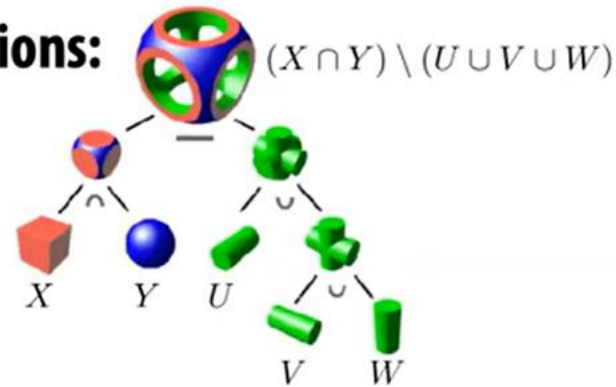
Constructive Solid Geometry (Implicit)

- Build more complicated shapes via Boolean operations

- Basic operations:



- Then chain together expressions:



Some Representations work better than others – depends on the task

Signed Distance Function

Level Sets - ISO Lines (same distance away from our shape)
Contour Lines (with same Signed Distance)

Distance:

$$\text{dist}(\mathbf{P}, \mathbf{A}) := \min\{\text{dist}(\mathbf{P}, \mathbf{Q}) \mid \mathbf{Q} \in \mathbf{A}\}$$

Signed Distance:

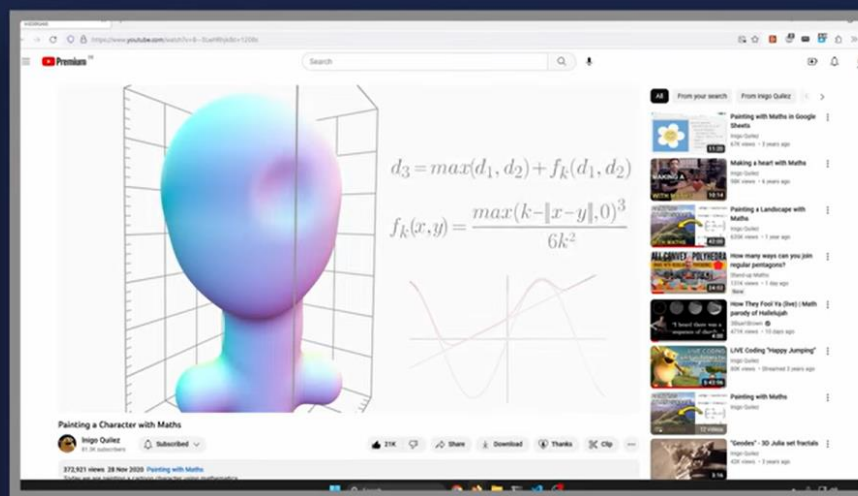
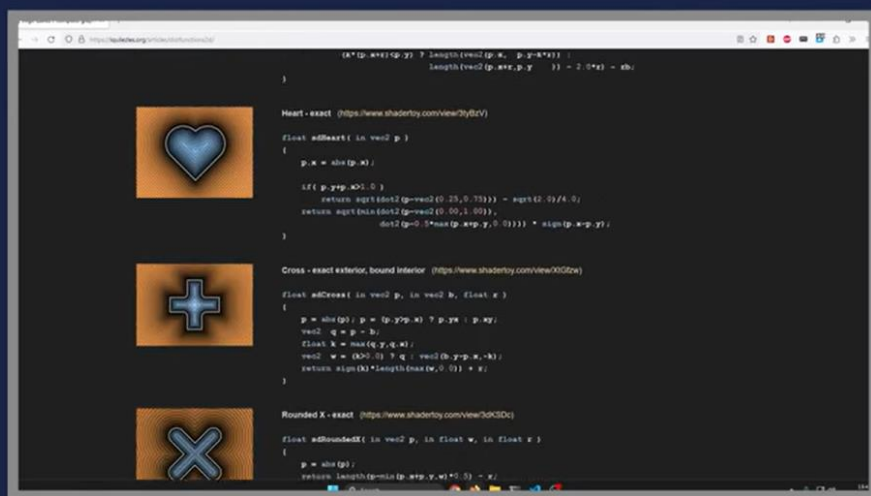
$$d_{\mathbf{A}}(\mathbf{P}) := \begin{cases} \text{dist}(\mathbf{P}, \partial\mathbf{A}), & \mathbf{P} \text{ outside} \\ -\text{dist}(\mathbf{P}, \partial\mathbf{A}), & \mathbf{P} \text{ inside} \end{cases}$$



Topographic Maps

Signed Distance Function Resources

Inigo Quilez



SDF for a Circle

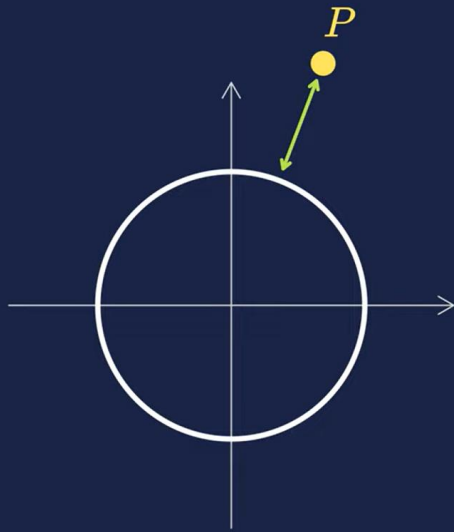
Circle

Given radius r :

$$d_{\text{circle}}(P) = \sqrt{x^2 + y^2} - r$$

Common circle equation:

$$x^2 + y^2 - r^2 = 0$$



Circle

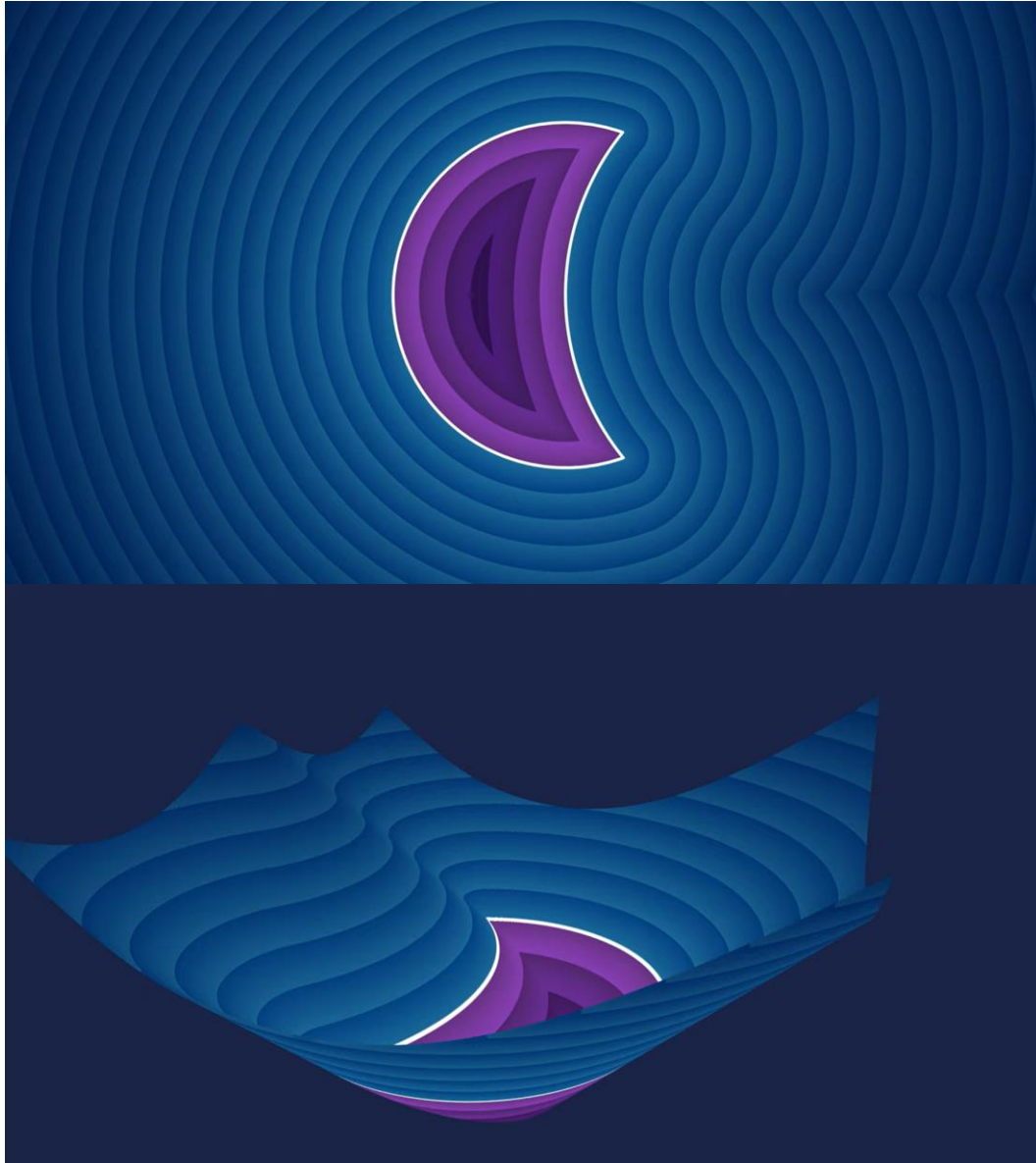
Given radius r :

$$d_{\text{circle}}(P) = \sqrt{x^2 + y^2} - r$$



Level Sets

A set of points that all map to the same value of SDF is called a Level Set



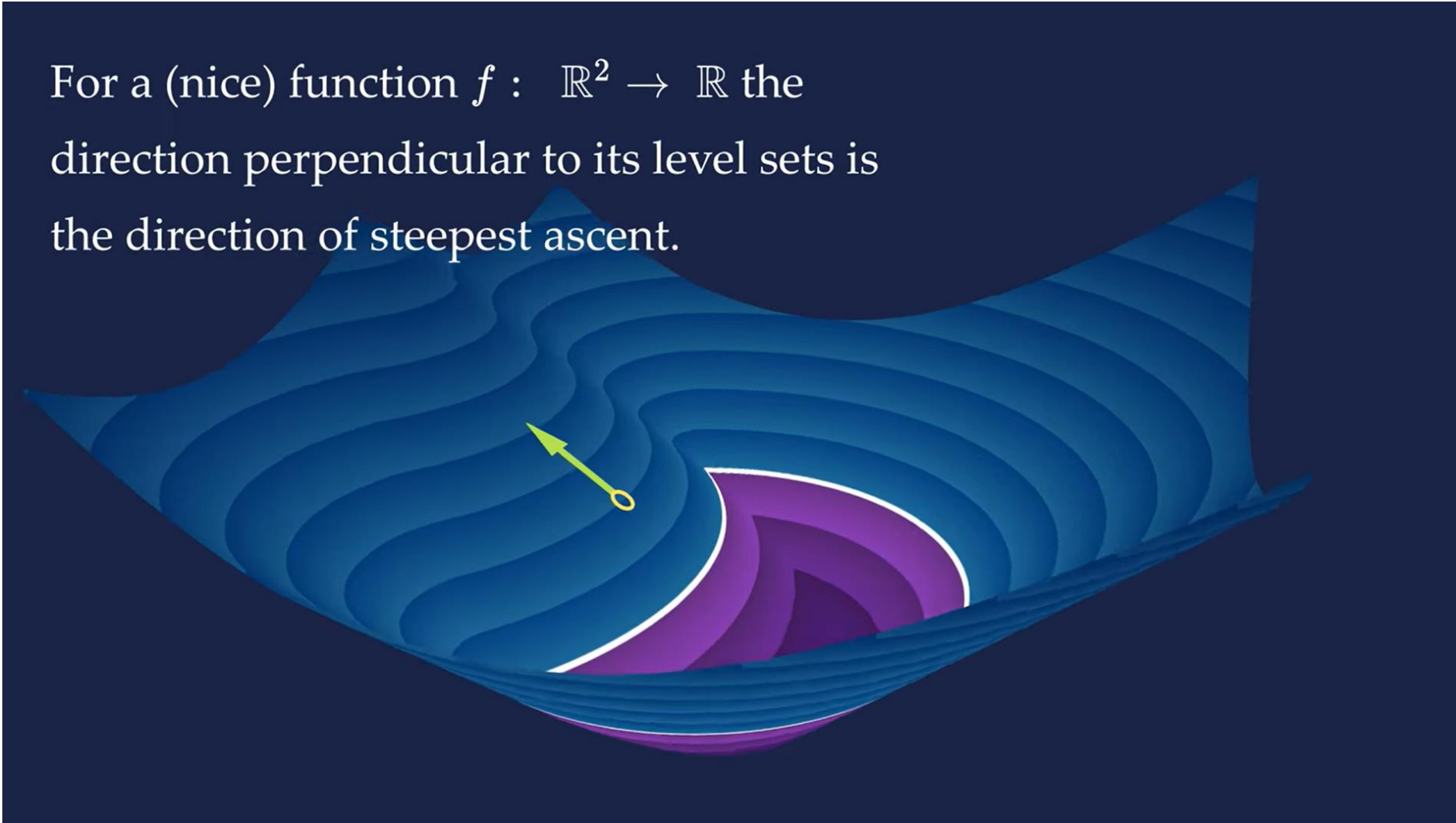
Input: X, Y coordinates

Output $Z = \text{SDF}(X,Y)$

2D plane is converted to a 3D Surface

Gradient of Level Sets

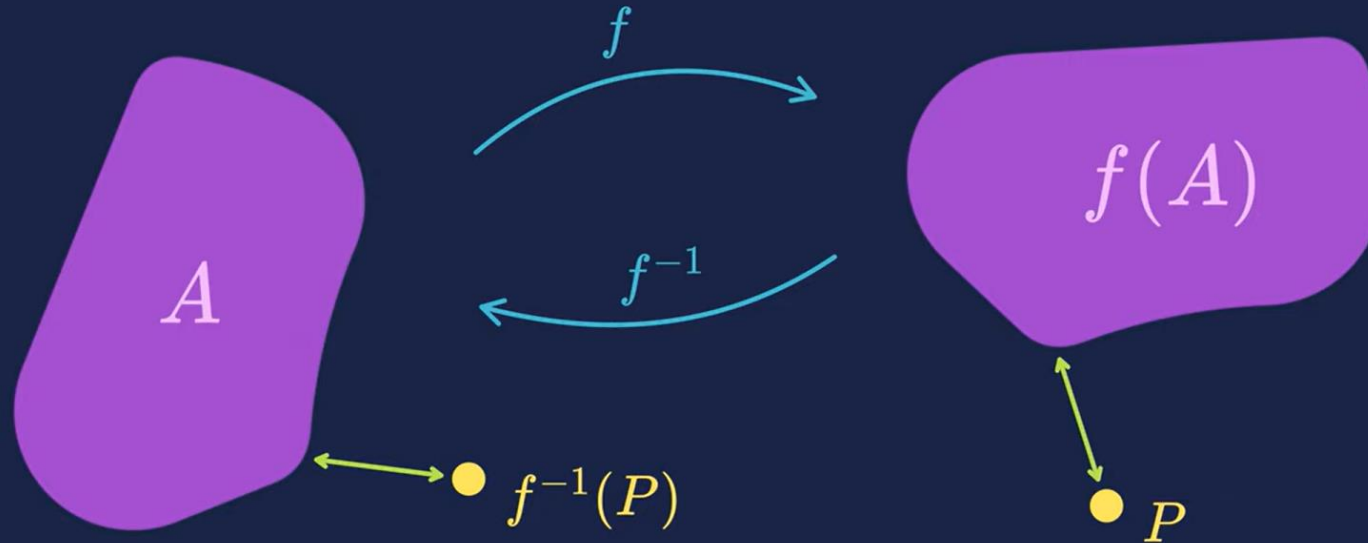
For a (nice) function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ the direction perpendicular to its level sets is the direction of steepest ascent.



SDF and Rigid Transformation

Given signed distance function d_A
and f , which is rotation + translation

$$d_{f(A)}(P) = d_A(f^{-1}(P))$$



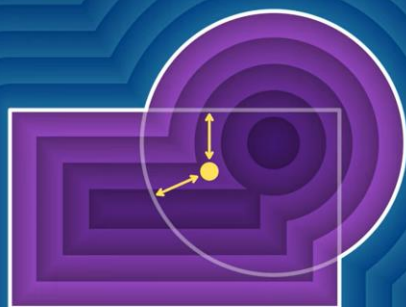
Combining SDFs

Combining SDFs

Given SDFs d_A and d_B :

$$d_{A \cup B} = \min(d_A, d_B)$$

(Kinda...)



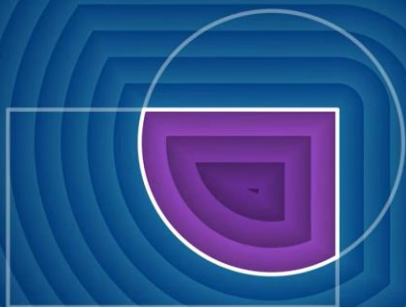
Combining SDFs

Given SDFs d_A and d_B :

$$d_{A \cup B} = \min(d_A, d_B)$$

$$d_{A \cap B} = \max(d_A, d_B)$$

(Kinda...)



Combining SDFs

Given SDFs d_A and d_B :

$$d_{A \cup B} = \min(d_A, d_B)$$

$$d_{A \cap B} = \max(d_A, d_B)$$

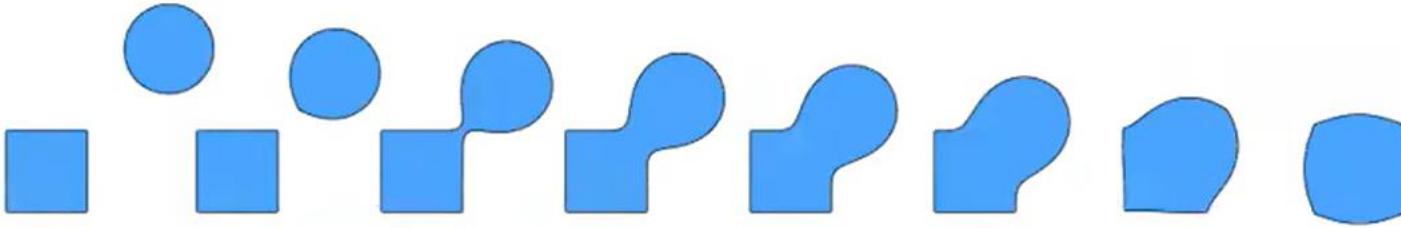
$$d_{A \setminus B} = \max(d_A, -d_B)$$

(Kinda...)



Blending Distance functions (Implicit)

- A *distance function* gives distance to closest point on object
- Can blend any two distance functions d_1, d_2 :



- Similar strategy to points, though many possibilities. E.g.,

$$f(x) := e^{d_1(x)^2} + e^{d_2(x)^2} - \frac{1}{2}$$

- Appearance depends on how we combine functions

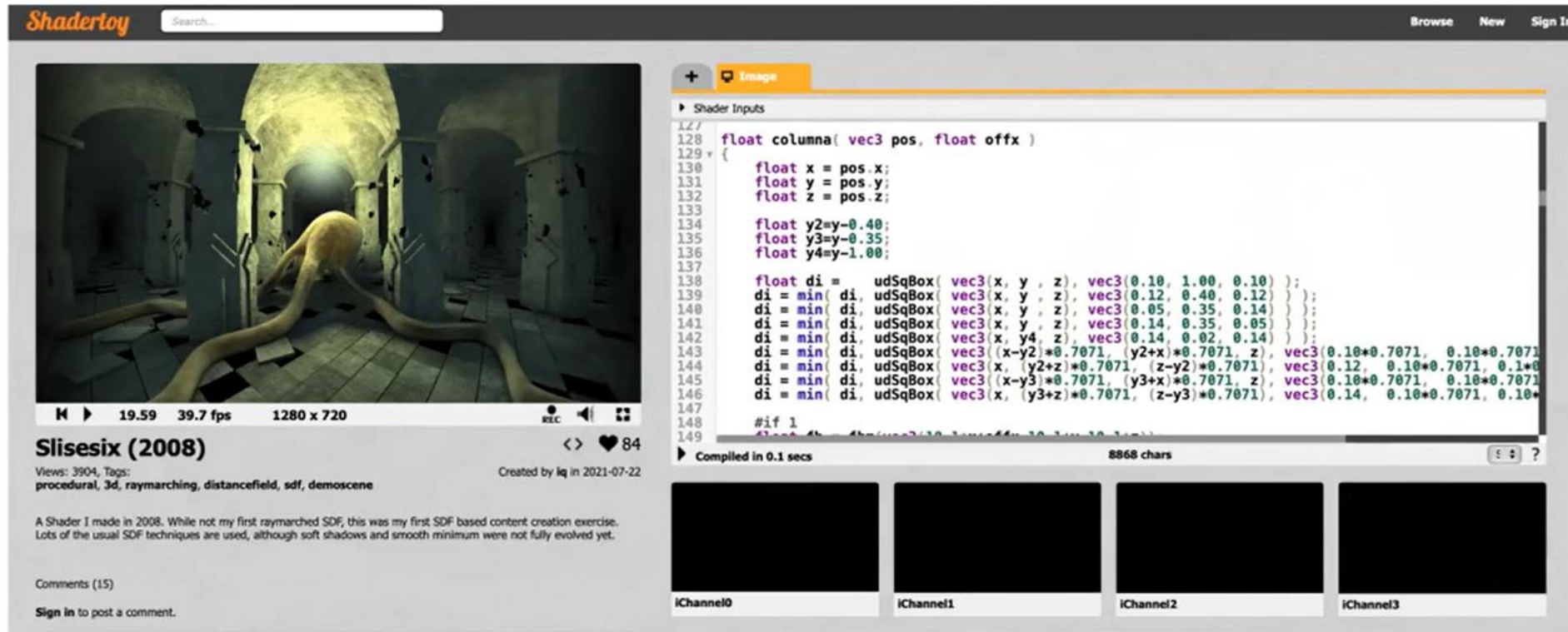
Q: How do we implement a Boolean union $d_1(x), d_2(x)$?

A: Just take the minimum: $f(x) = \min(d_1(x), d_2(x))$

Scene with purely Algebraic functions

Scene made of pure signed distance functions

Art with math -- really hard!



The screenshot displays the Shadertoy interface. On the left, a 3D rendered scene titled "Slisesix (2008)" is shown. The scene depicts a dark, atmospheric interior with a checkered floor, stone walls, and a large, glowing, organic structure in the center. Below the image, the title "Slisesix (2008)" is followed by a description: "A Shader I made in 2008. While not my first raymarched SDF, this was my first SDF based content creation exercise. Lots of the usual SDF techniques are used, although soft shadows and smooth minimum were not fully evolved yet." The scene's performance metrics are listed as 19.59, 39.7 fps, and 1280 x 720 resolution. The right side of the interface shows the GLSL shader code for the scene. The code defines a function `float columna(vec3 pos, float offx)` and uses `udSqBox` to calculate signed distances to various geometric primitives. The code is compiled in 0.1 seconds and is 8868 characters long. Below the code, four preview windows labeled `iChannel0`, `iChannel1`, `iChannel2`, and `iChannel3` are visible, all showing black.

Shadertoy

Search...

Browse New Sign In

19.59 39.7 fps 1280 x 720

Slisesix (2008)

Views: 3904, Tags: procedural, 3d, raymarching, distancefield, sdf, demoscene

Created by iq in 2021-07-22

A Shader I made in 2008. While not my first raymarched SDF, this was my first SDF based content creation exercise. Lots of the usual SDF techniques are used, although soft shadows and smooth minimum were not fully evolved yet.

Comments (15)

Sign in to post a comment.

```
127
128
129 {
130     float x = pos.x;
131     float y = pos.y;
132     float z = pos.z;
133
134     float y2=y-0.40;
135     float y3=y-0.35;
136     float y4=y-1.00;
137
138     float di = udSqBox( vec3( x, y, z ), vec3( 0.10, 1.00, 0.10 ) );
139     di = min( di, udSqBox( vec3( x, y, z ), vec3( 0.12, 0.40, 0.12 ) );
140     di = min( di, udSqBox( vec3( x, y, z ), vec3( 0.05, 0.35, 0.14 ) );
141     di = min( di, udSqBox( vec3( x, y, z ), vec3( 0.14, 0.35, 0.05 ) );
142     di = min( di, udSqBox( vec3( x, y4, z ), vec3( 0.14, 0.02, 0.14 ) );
143     di = min( di, udSqBox( vec3( (x-y2)*0.7071, (y2+x)*0.7071, z ), vec3( 0.10*0.7071, 0.10*0.7071, 0.10*0.7071 ) );
144     di = min( di, udSqBox( vec3( x, (y2+z)*0.7071, (z-y2)*0.7071 ), vec3( 0.12, 0.10*0.7071, 0.10*0.7071 ) );
145     di = min( di, udSqBox( vec3( (x-y3)*0.7071, (y3+x)*0.7071, z ), vec3( 0.10*0.7071, 0.10*0.7071, 0.10*0.7071 ) );
146     di = min( di, udSqBox( vec3( x, (y3+z)*0.7071, (z-y3)*0.7071 ), vec3( 0.14, 0.10*0.7071, 0.10*0.7071 ) );
147
148     #if 1
149     float f1 = 0.5*(1+cos(2*3.14159*(x-0.5)));
150     float f2 = 0.5*(1+cos(2*3.14159*(y-0.5)));
151     float f3 = 0.5*(1+cos(2*3.14159*(z-0.5)));
152     float f4 = 0.5*(1+cos(2*3.14159*(x+y+z-1.5)));
153     float f5 = 0.5*(1+cos(2*3.14159*(x-y-z-0.5)));
154     float f6 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
155     float f7 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
156     float f8 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
157     float f9 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
158     float f10 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
159     float f11 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
160     float f12 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
161     float f13 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
162     float f14 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
163     float f15 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
164     float f16 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
165     float f17 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
166     float f18 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
167     float f19 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
168     float f20 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
169     float f21 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
170     float f22 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
171     float f23 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
172     float f24 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
173     float f25 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
174     float f26 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
175     float f27 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
176     float f28 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
177     float f29 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
178     float f30 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
179     float f31 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
180     float f32 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
181     float f33 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
182     float f34 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
183     float f35 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
184     float f36 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
185     float f37 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
186     float f38 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
187     float f39 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
188     float f40 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
189     float f41 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
190     float f42 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
191     float f43 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
192     float f44 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
193     float f45 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
194     float f46 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
195     float f47 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
196     float f48 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
197     float f49 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
198     float f50 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
199     float f51 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
200     float f52 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
201     float f53 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
202     float f54 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
203     float f55 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
204     float f56 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
205     float f57 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
206     float f58 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
207     float f59 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
208     float f60 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
209     float f61 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
210     float f62 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
211     float f63 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
212     float f64 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
213     float f65 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
214     float f66 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
215     float f67 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
216     float f68 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
217     float f69 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
218     float f70 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
219     float f71 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
220     float f72 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
221     float f73 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
222     float f74 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
223     float f75 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
224     float f76 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
225     float f77 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
226     float f78 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
227     float f79 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
228     float f80 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
229     float f81 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
230     float f82 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
231     float f83 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
232     float f84 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
233     float f85 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
234     float f86 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
235     float f87 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
236     float f88 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
237     float f89 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
238     float f90 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
239     float f91 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
240     float f92 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
241     float f93 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
242     float f94 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
243     float f95 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
244     float f96 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
245     float f97 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
246     float f98 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
247     float f99 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
250     float f100 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
251     float f101 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
252     float f102 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
253     float f103 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
254     float f104 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
255     float f105 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
256     float f106 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
257     float f107 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
258     float f108 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
259     float f109 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
260     float f110 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
261     float f111 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
262     float f112 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
263     float f113 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
264     float f114 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
265     float f115 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
266     float f116 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
267     float f117 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
268     float f118 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
269     float f119 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
270     float f120 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
271     float f121 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
272     float f122 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
273     float f123 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
274     float f124 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
275     float f125 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
276     float f126 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
277     float f127 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
278     float f128 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
279     float f129 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
280     float f130 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
281     float f131 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
282     float f132 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
283     float f133 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
284     float f134 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
285     float f135 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
286     float f136 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
287     float f137 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
288     float f138 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
289     float f139 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
290     float f140 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
291     float f141 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
292     float f142 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
293     float f143 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
294     float f144 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
295     float f145 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
296     float f146 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
297     float f147 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
298     float f148 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
299     float f149 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
300     float f150 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
301     float f151 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
302     float f152 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
303     float f153 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
304     float f154 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
305     float f155 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
306     float f156 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
307     float f157 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
308     float f158 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
309     float f159 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
310     float f160 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
311     float f161 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
312     float f162 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
313     float f163 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
314     float f164 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
315     float f165 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
316     float f166 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
317     float f167 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
318     float f168 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
319     float f169 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
320     float f170 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
321     float f171 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
322     float f172 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
323     float f173 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
324     float f174 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
325     float f175 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
326     float f176 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
327     float f177 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
328     float f178 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
329     float f179 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
330     float f180 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
331     float f181 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
332     float f182 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
333     float f183 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
334     float f184 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
335     float f185 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
336     float f186 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
337     float f187 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
338     float f188 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
339     float f189 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
340     float f190 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
341     float f191 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
342     float f192 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
343     float f193 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
344     float f194 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
345     float f195 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
346     float f196 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
347     float f197 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
348     float f198 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
349     float f199 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
350     float f200 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
351     float f201 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
352     float f202 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
353     float f203 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
354     float f204 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
355     float f205 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
356     float f206 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
357     float f207 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
358     float f208 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
359     float f209 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
360     float f210 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
361     float f211 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
362     float f212 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
363     float f213 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
364     float f214 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
365     float f215 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
366     float f216 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
367     float f217 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
368     float f218 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
369     float f219 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
370     float f220 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
371     float f221 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
372     float f222 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
373     float f223 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
374     float f224 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
375     float f225 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
376     float f226 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
377     float f227 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
378     float f228 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
379     float f229 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
380     float f230 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
381     float f231 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
382     float f232 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
383     float f233 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
384     float f234 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
385     float f235 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
386     float f236 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
387     float f237 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
388     float f238 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
389     float f239 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
390     float f240 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
391     float f241 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
392     float f242 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
393     float f243 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
394     float f244 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
395     float f245 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
396     float f246 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
397     float f247 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
398     float f248 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
399     float f249 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
400     float f250 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
401     float f251 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
402     float f252 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
403     float f253 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
404     float f254 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
405     float f255 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
406     float f256 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
407     float f257 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
408     float f258 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
409     float f259 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
410     float f260 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
411     float f261 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
412     float f262 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
413     float f263 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
414     float f264 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
415     float f265 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
416     float f266 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
417     float f267 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
418     float f268 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
419     float f269 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
420     float f270 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
421     float f271 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
422     float f272 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
423     float f273 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
424     float f274 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
425     float f275 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
426     float f276 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
427     float f277 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
428     float f278 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
429     float f279 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
430     float f280 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
431     float f281 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
432     float f282 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
433     float f283 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
434     float f284 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
435     float f285 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
436     float f286 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
437     float f287 = 0.5*(1+cos(2*3.14159*(z+y-x-0.5)));
438     float f288 = 0.5*(1+cos(2*3.14159*(x+y-z-0.5)));
439     float f289 = 0.5*(1+cos(2*3.14159*(x-y+z-0.5)));
440     float f290 = 0.5*(1+cos(2*3.14159*(y-z-x-0.5)));
441     float f291 = 0.5*(1+cos(2*3.14159*(y+z-x-0.5)));
442     float f292 = 0.5*(1+cos(2*3.14159*(z-x-y-0.5)));
443     float f293 = 0.5*(1+cos(2*3.
```

3 basic methods for representing a surface:

- ❖ **DEM (digital elevation model):** A Digital Elevation Model, or DEM, is a 3D representation of a topographic surface of the Earth, excluding trees, buildings, and any other surface objects. It provides accurate and detailed information about the terrain, which can be used to make better-informed decisions.
- ❖ DEM is a set of regularly spaced sampled ground points in the x and y dimensions (although spacing not necessarily the same in each) accompanied by an elevation measure (z dimension).
- ❖ The DEM terminology was introduced by USGS (United States Geological Survey). Two concepts used for determining elevation at points within the grid cells:
 - Lattice: each point represents a value on the surface only at the center of the grid cell
 - Surface grid considers each sample as a square/rectangular cell with a constant surface value.
- ❖ **TIN (Triangulated Irregular Network)** a set of adjacent, non-overlapping triangles with x, y coordinates and z vertical elevations for their vertices, along with topological relationship between the triangles and their adjacent neighbors.
- ❖ **Contour lines:** lines of equal elevation, drawn at a given interval (e.g. every 6 or 25 feet)

The general term **digital terrain model** (DTM) may be used to refer to any of the above surface representations when in digital form.

DEM sometimes used synonymously with DTM—don't.

Storing & Converting Surface Data

❖ **3-D surfaces** are normally stored in one of two forms within ArcGIS

- as a **GRID**, which is ArcInfo's general raster format
- as a **TIN** which is a vector format for surfaces

❖ **However, when you download data from the Internet, surface data may be in other formats, such as**

- DEM format, as originally developed by USGS (United States Geological Survey)
- SDTS (Spatial Data Transfer Standard) format, which is an FGDC (Federal Geographic Data Committee) standard
- E00 which is ESRI's text formatted for distributing coverages and GRIDS
- Points and breaklines

❖ **Conversion to GRID or TIN is generally required for display or analysis within the ArcGIS system**

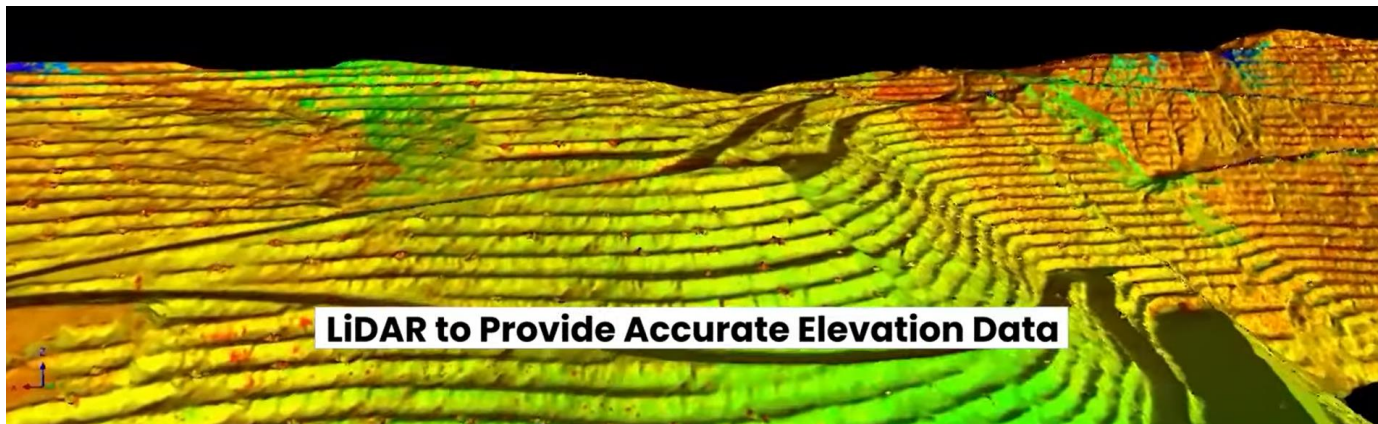
- Generally, ArcToolbox has capabilities for converting these formats to GRIDs or TINs

❖ **Contour lines can be stored as vector lines in a coverage, shapefile, or geodatabase,**

- can only be used for map display but not analysis, so this is not a recommended format for surface storage.

Digital Elevation Model

- ❖ A Digital Elevation Model, or DEM, is a 3D representation of a topographic surface of the Earth, excluding trees, buildings, and any other surface objects. It provides accurate and detailed information about the terrain, which can be used to make better-informed decisions.
- ❖ DEM data is created using LIDAR or Stereo Photogrammetry
- ❖ a sampled array of elevations (z) that are at regularly spaced intervals in the x and y directions.
- ❖ two approaches for determining the surface z value of a location between sample points.
 - In a **lattice**, each mesh point represents a value on the surface only at the center of the grid cell. The z-value is approximated by interpolation between adjacent sample points; it does not imply an area of constant value.
 - A **surface grid** considers each sample as a square cell with a constant surface value.



Advantages

- Simple conceptual model
- Data cheap to obtain
- Easy to relate to other raster data
- Irregularly spaced set of points can be converted to regular spacing by interpolation

Disadvantages

- Does not conform to variability of the terrain
- Linear features not well represented

GRID as a Storage Method

❖ GRIDs are Environmental Systems Research Institute (ESRI)'s raster data format

➤ Use for storing DEMS or other data in raster format

❖ GRID stores data as either:

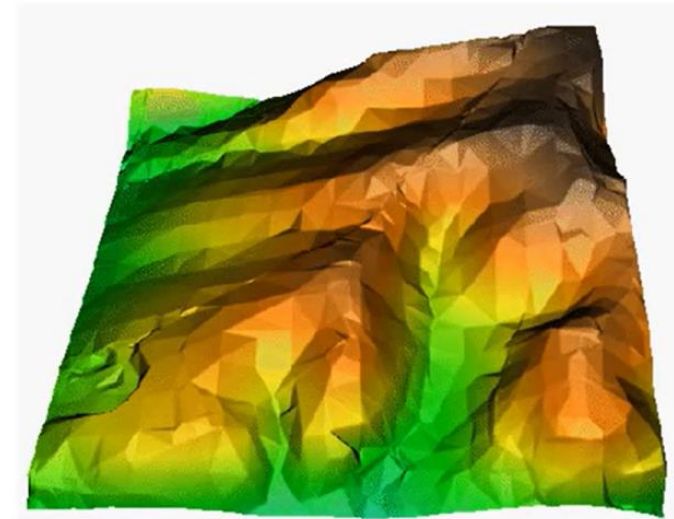
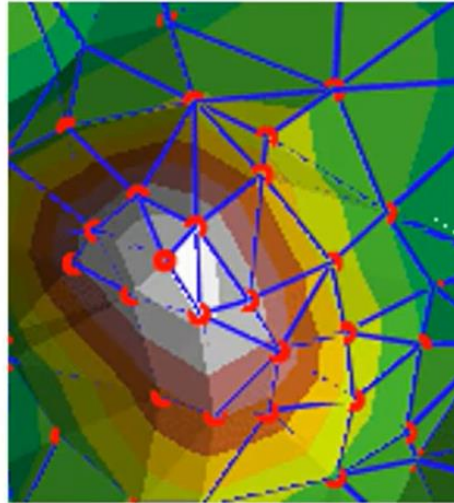
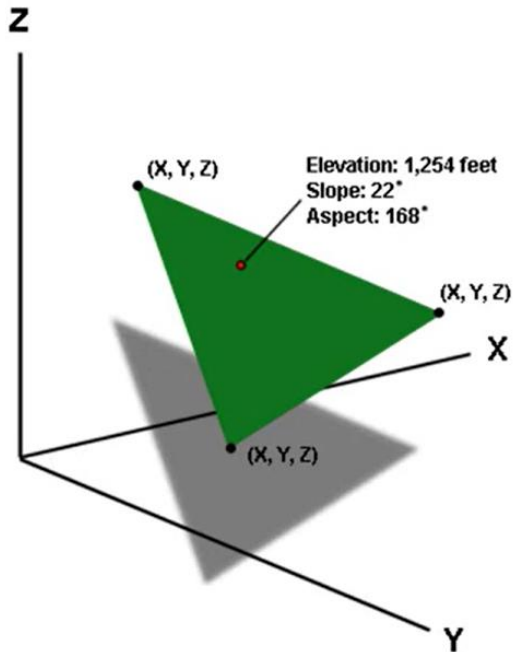
- **Integer:** in which case there is an associated *value attribute table* (VAT) which contains one record for each *different* value in the raster (thus there are normally substantially fewer records in the VAT table than there are cells in the raster);
- this record stores the value itself, a count of the number of cells with that value, and any additional attributes the user wishes to attach.
- Thus, the values could be codes for soil type and the VAT could contain fertility measures, soil name, construction suitability codes, etc.
- If you select a record in the VAT, all cells with that value will highlight in the View or Scene.
- **Floating point:** (number with a decimal point) in which case there is no VAT table, and simply one decimal value per cell

❖ Integer GRIDS are generally substantially faster to process.

What is a Triangulated Irregular Network (TIN)?

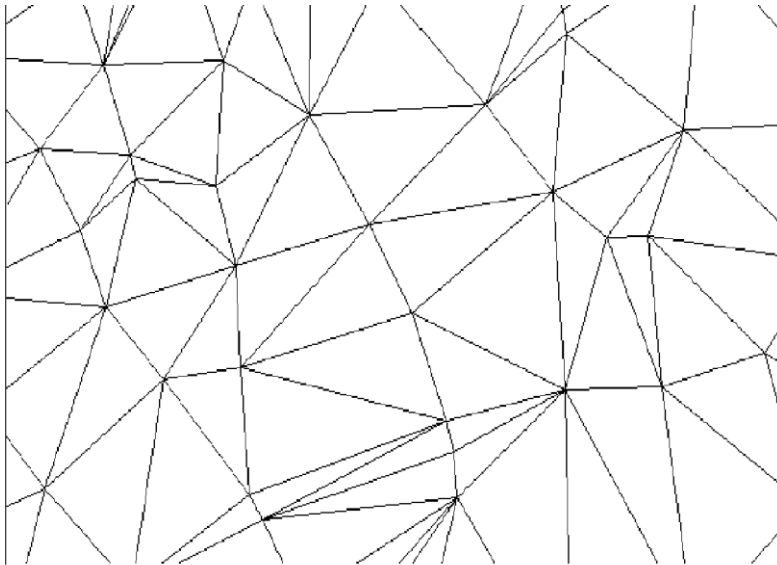
TIN is a vector representation of continuous spatial data

- A list of X,Y,Z nodes connected by edges*
- Ideal for representing 3D surfaces*



Triangulated Irregular Network

- ❖ TIN is a set of adjacent, non-overlapping triangles computed from irregularly spaced points, with x, y horizontal coordinates and z vertical elevations.
- ❖ Quick and easy way to represent irregularly spaced datasets
- ❖ Useful to represent Topography, Elevation, Contours



- ❖ Advantages

- Can capture significant slope features (ridges, etc)
- Efficient since require few triangles in flat areas
- Easy for certain analyses: slope, aspect, volume

- ❖ Disadvantages

- Analysis involving comparison with other layers difficult

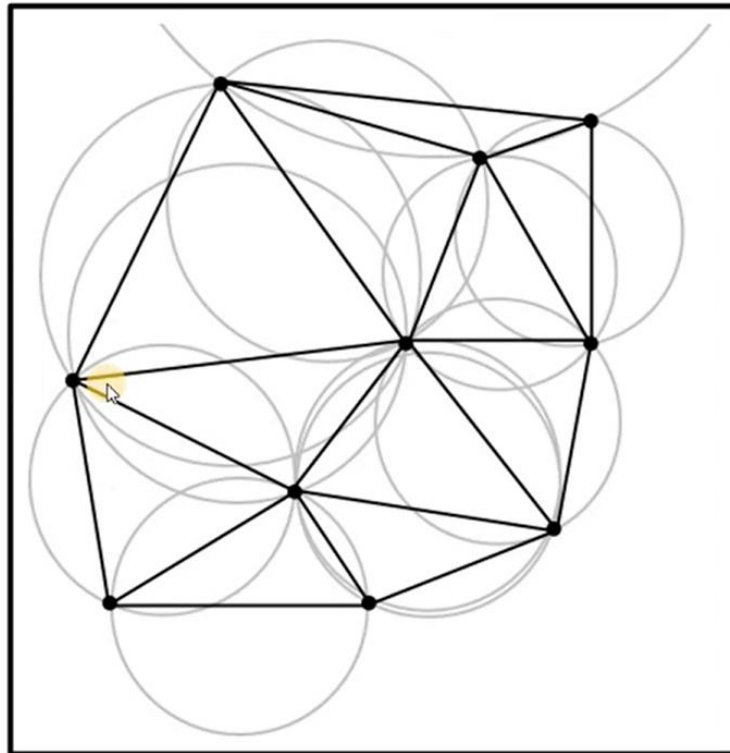
-Ideal for irregularly spaced datasets such as LiDAR or drone-based photogrammetry

-Fewer nodes can be used in areas that don't change much

TIN: Delauney Triangulation

TINs are based on Delauney Triangulation

-No node can fall within the circle circumscribed by any triangle



TIN as a Storage Method

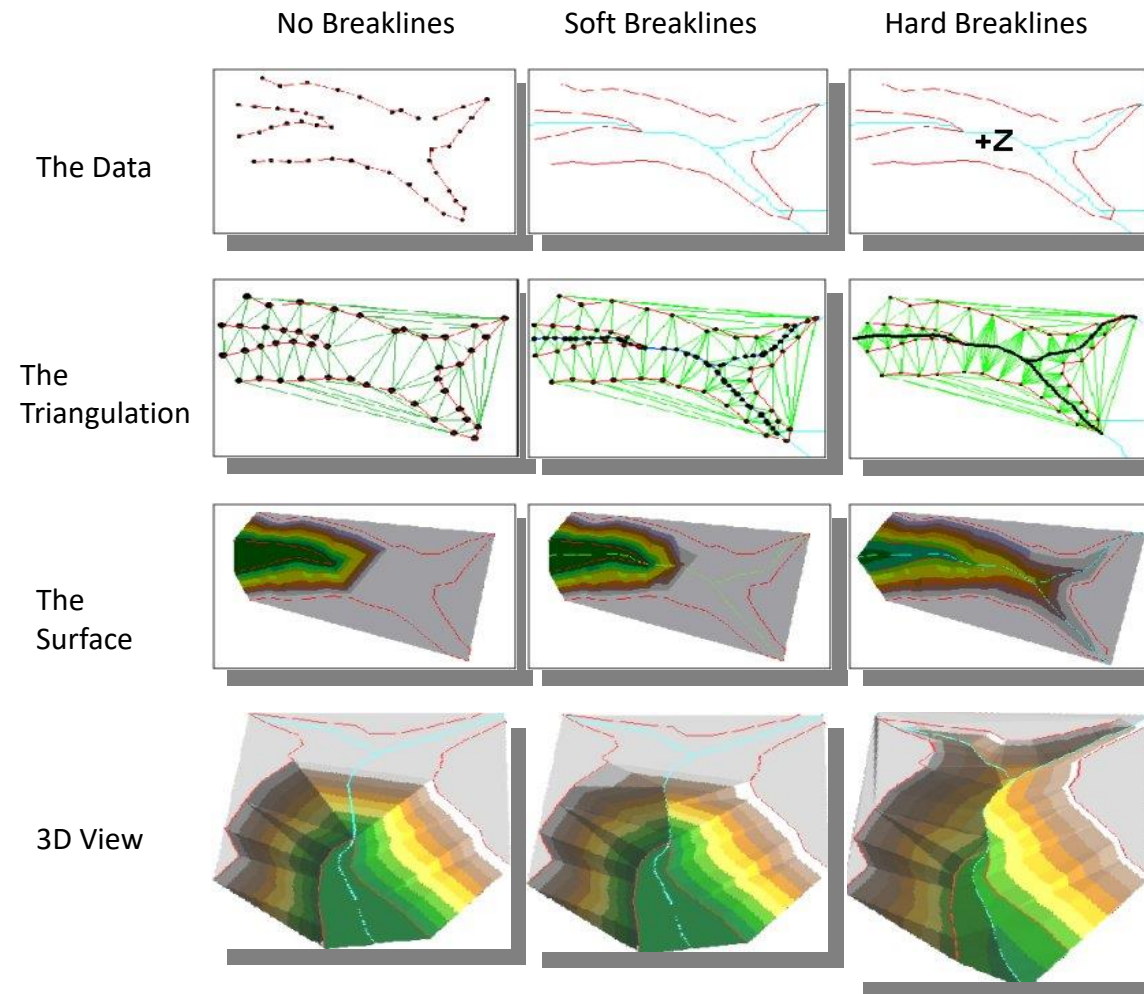
❖ TINs

- are the most useful method for representing a continuous surface in a vector GIS system.
- data sets comprising any combination of contours, breaklines and point elevations (either DEM or massed points) can be combined as input to create a TIN

❖ TINs are especially useful for analytical purposes

- Good model for representing surfaces
- slope and aspect easily derived
- simplify the calculation of surface area and volume

Creating a TIN

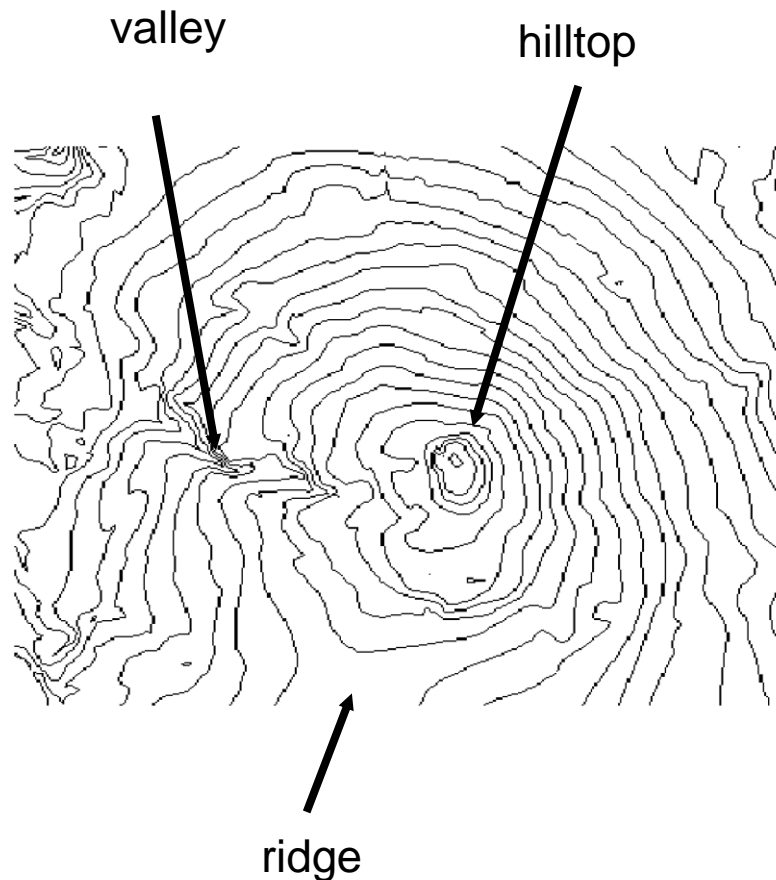


Break lines

Linear features which define and control surface behavior in terms of smoothness and continuity.

Contour (isolines) Lines

Contour lines, or isolines, of constant elevation at a specified interval,



Advantages

- ❖ Familiar to many people
- ❖ Easy to obtain mental picture of surface
 - Close lines = steep slope
 - Uphill V = stream
 - Downhill V or bulge = ridge
 - Circle = hill top or basin

Disadvantages

- ❖ Poor for computer representation: no formal digital model
- ❖ Must convert to raster or TIN for analysis
- ❖ Contour generation from point data requires sophisticated interpolation routines, often with specialized software such as *Surfer* from Golden Software, Inc., or ArcView Spatial Analyst extension

References

- ❖ Virtual Humans -- Lecture 03.1 Surface Representations: <https://www.youtube.com/watch?v=5uE7Pc5mr1I>
- ❖ The SDF of a box: <https://www.youtube.com/watch?v=62-pRVZuS5c&list=PL0EpikNmjs2CYUMePMGh3IjjP4tQlYqji>
- ❖ <https://iquilezles.org/articles/distfunctions/> (Multiple Example 3D distance functions)
- ❖ <https://iquilezles.org/articles/distfunctions2d/>
- ❖ [What is Digital Elevation Model or DEM?](#)
- ❖ [Signed Distance Functions & Ray-Marching](#) (very good)
- ❖ [What is a Triangular Irregular Network \(TIN\)?](#)
- ❖ <https://medium.com/@sim30217/truncated-signed-distance-function-f765a0f1d432>