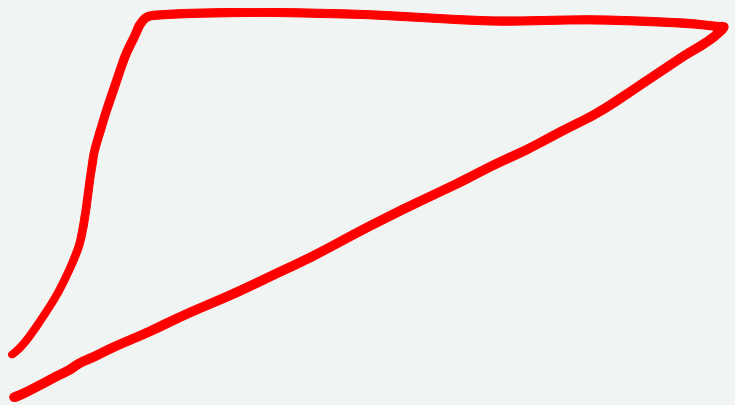
CS559 Lecture 19-20: More Texture

Part 2 - Fake Normals

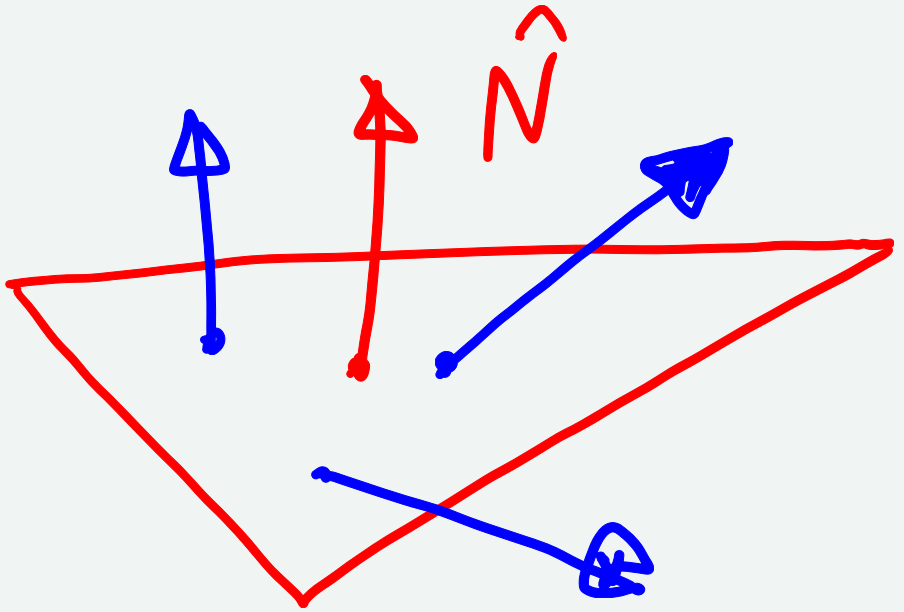
Bump Mapping

How do we get things to not be flat?

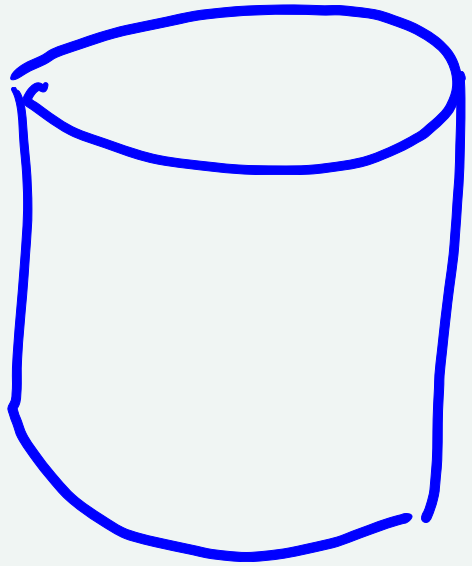
1. Make lots of triangles
2. Fake it with texture

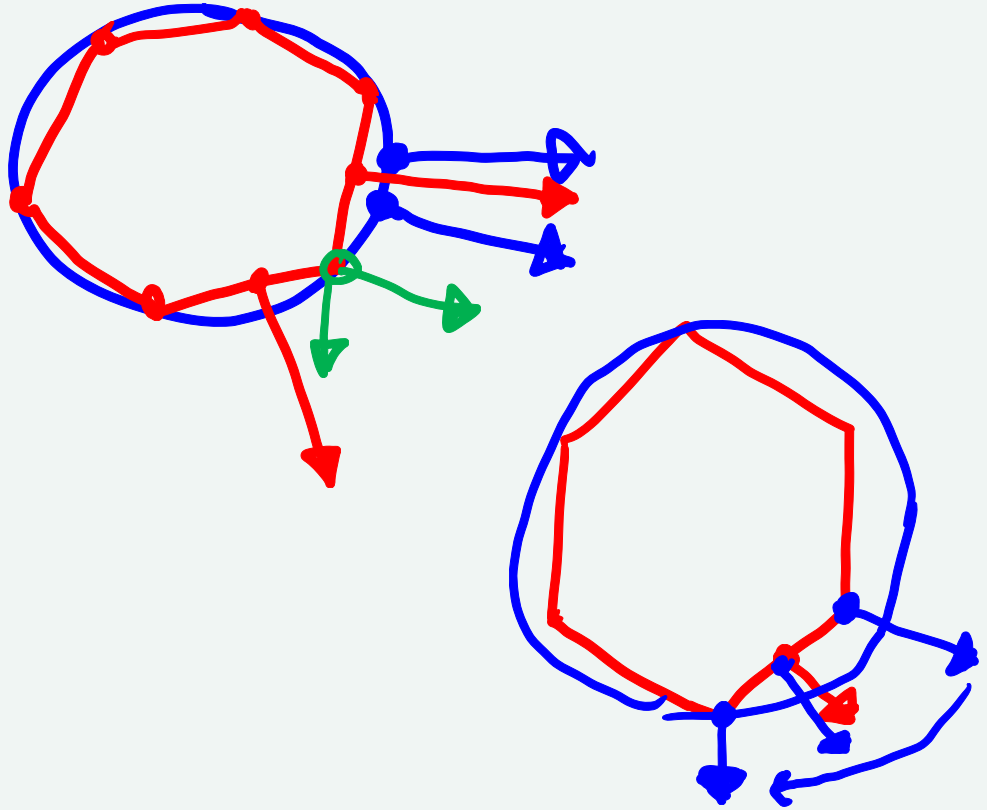


The Real Normal to a Triangle
The Fake Normal to a Triangle

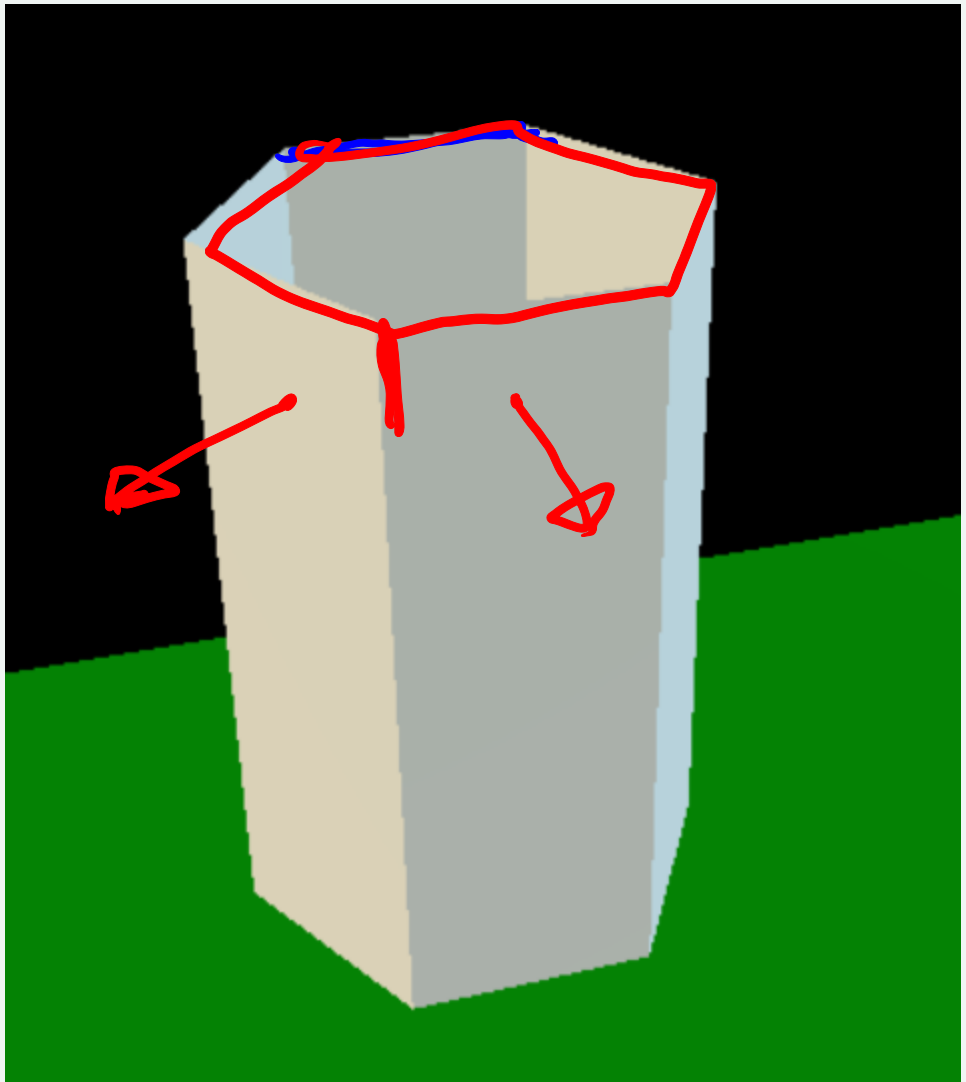


Why Fake Normals 1: Faking a Smooth Surface

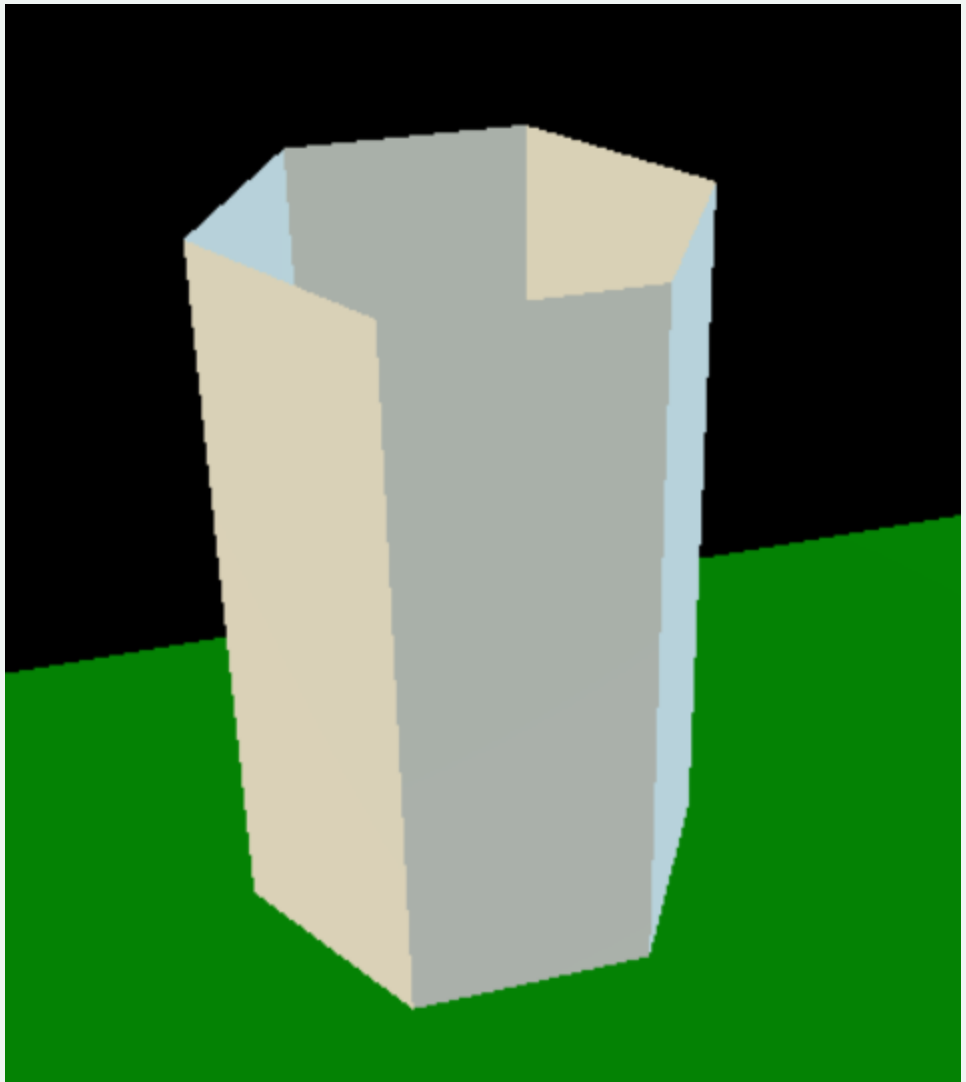


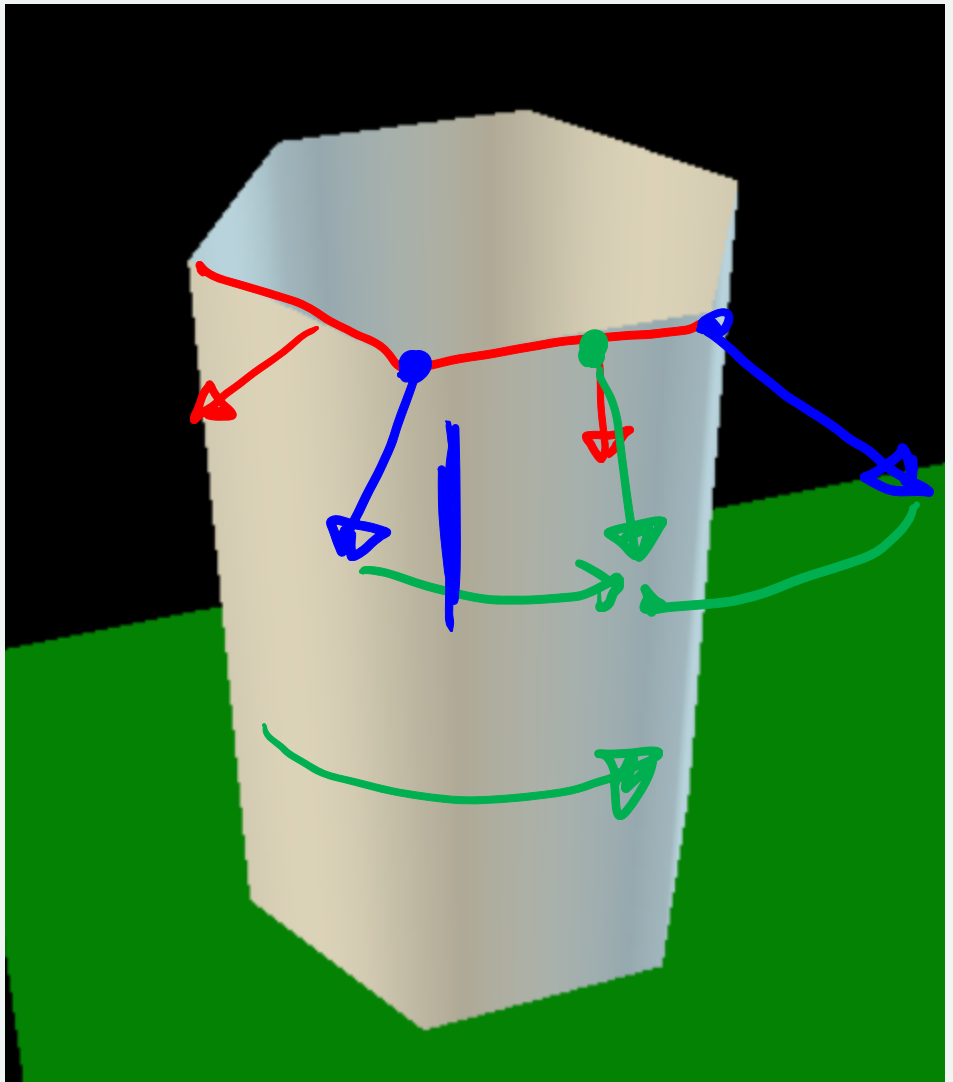


A "Cylinder" (6 sides)



A "Cylinder" (6 sides)



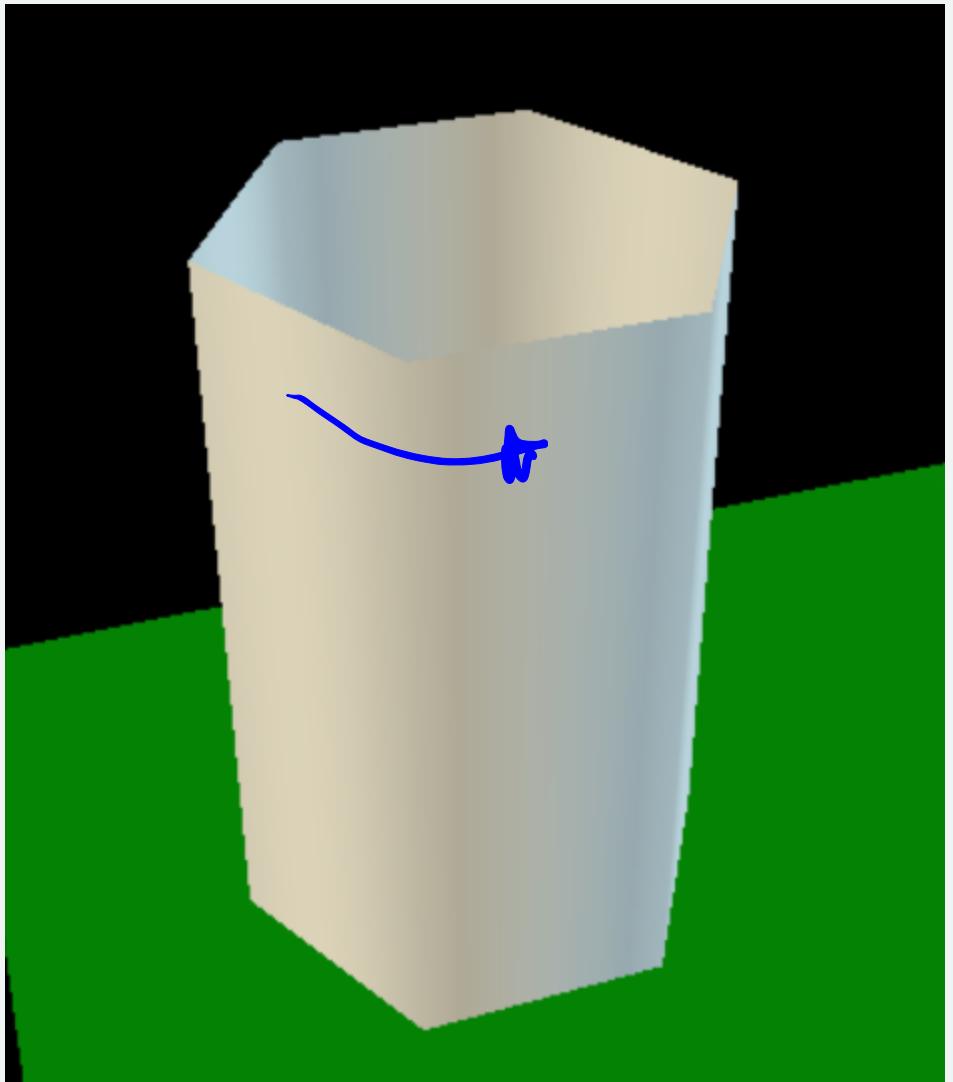


Smooth Shading

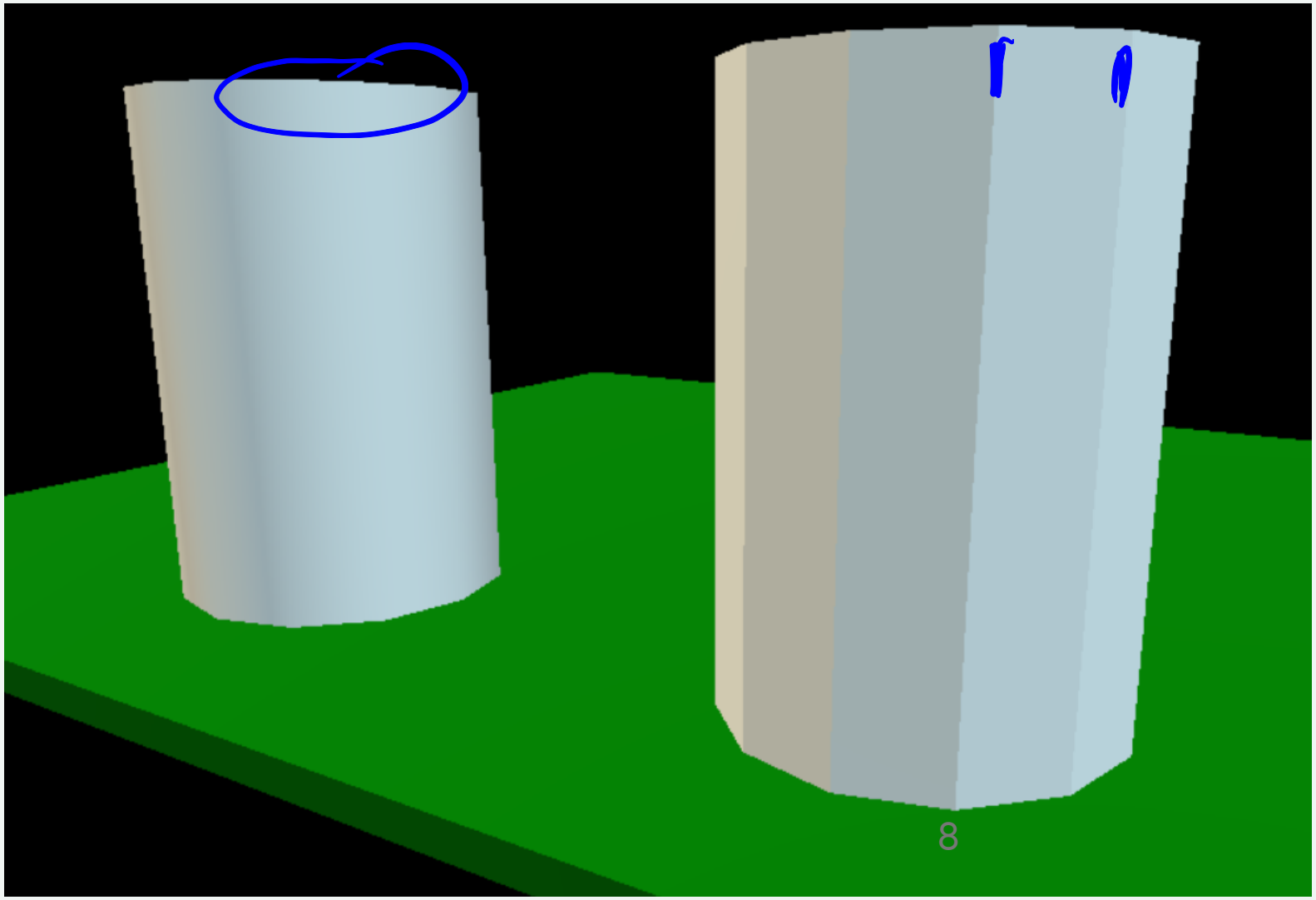
Still a 6 sided cylinder

Only changing the **lighting**

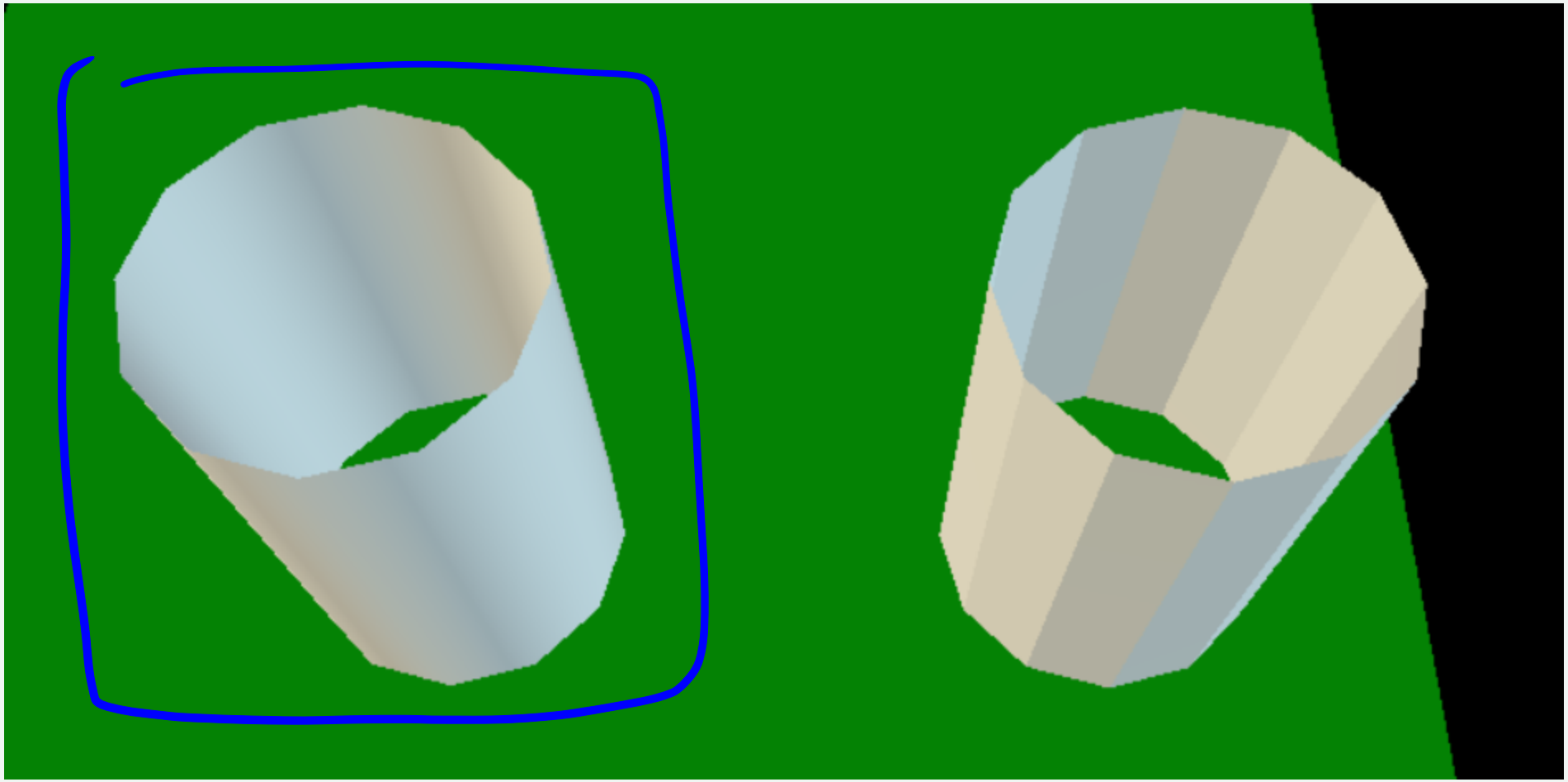
Change lighting by changing **normals**



Even better (12 sides)



Really 12 sides



Faking Shapes with Normals

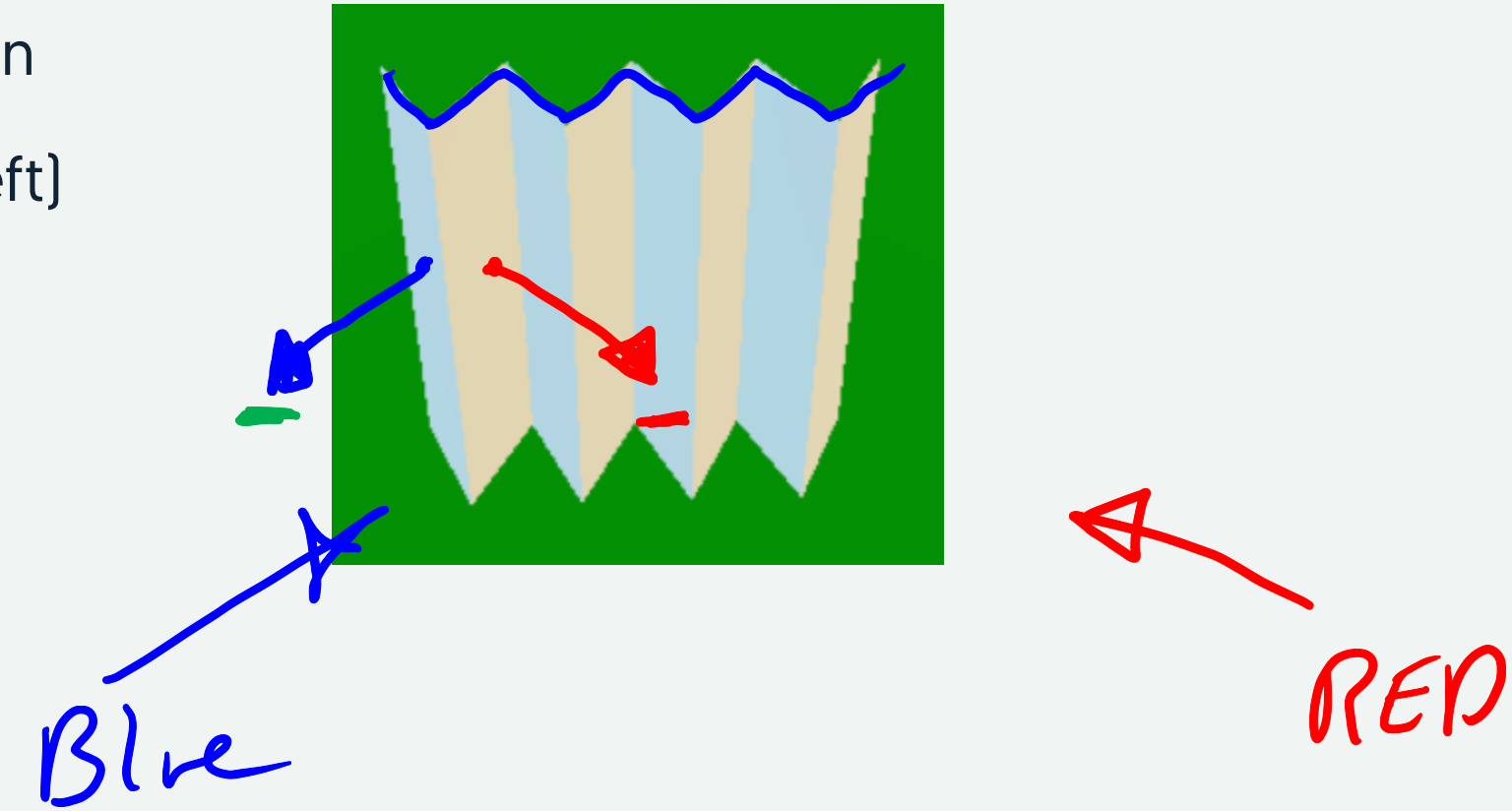
We don't have to make the right shape...

We just have to make it look right (with lighting)

Toy example: Wavy surface

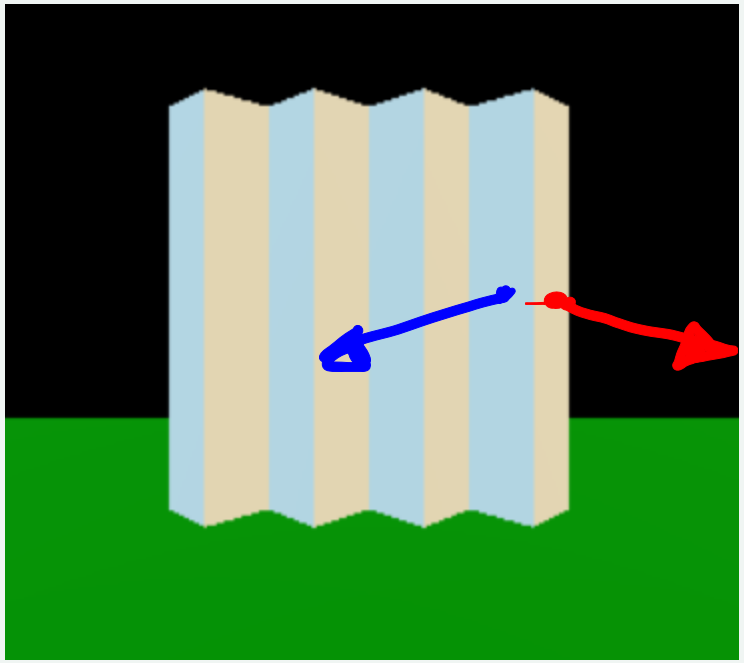
16 triangles - accordion pattern

extreme lighting (blue from left)



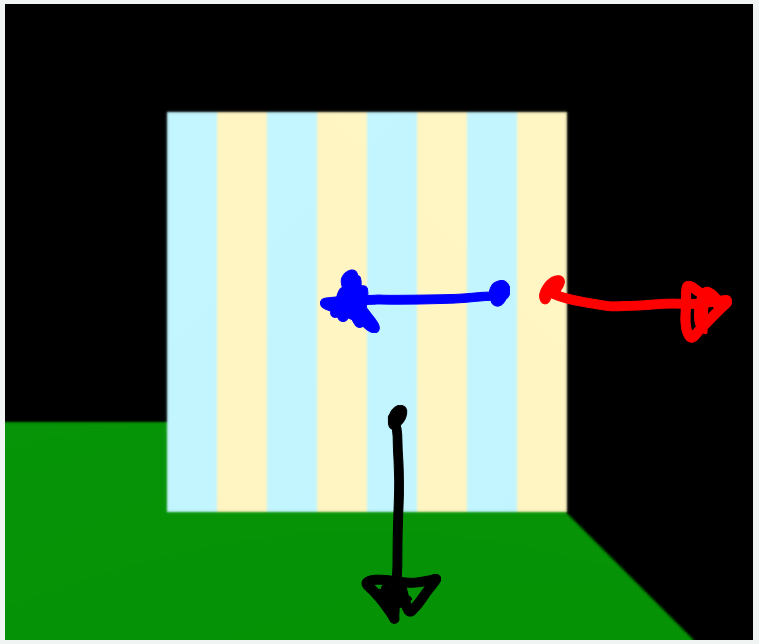
Toy example: Wavy surface

head on



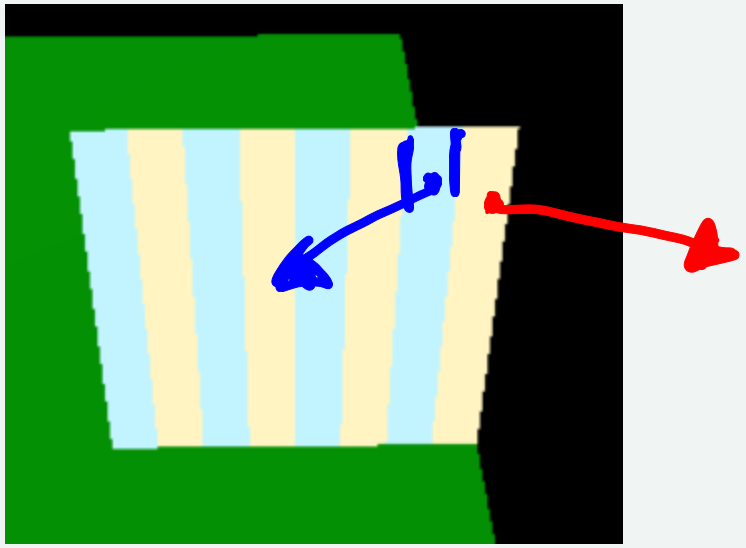
Fake Wavy surface

head on

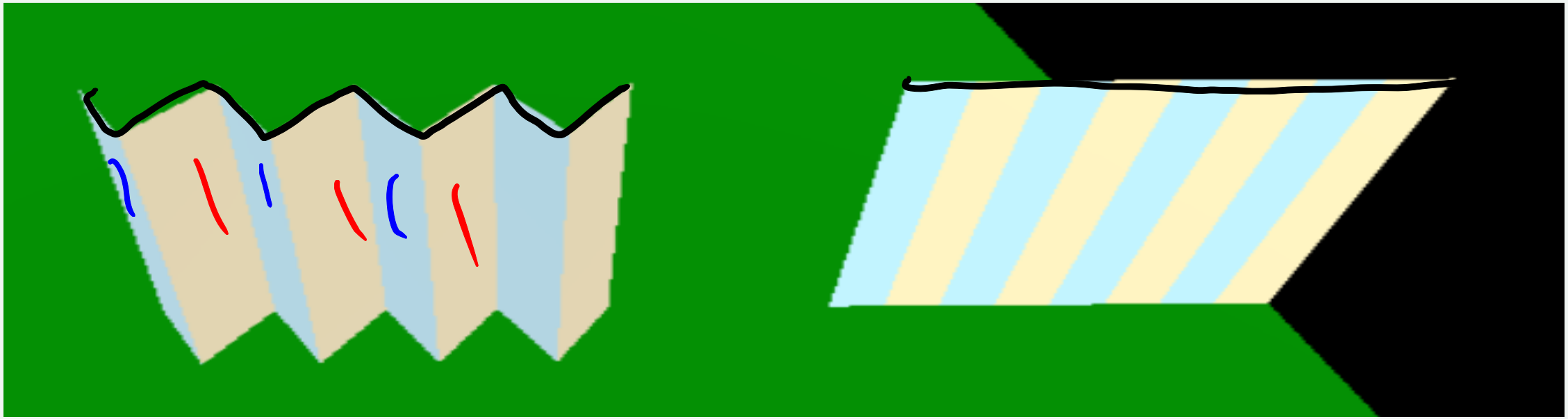


Fake?

It's flat - we just changed the normals



Real vs. Fake



Not so bad if you don't look at the edge...

Normal Maps

Idea: texture lookup of the normal vector!

R = x direction

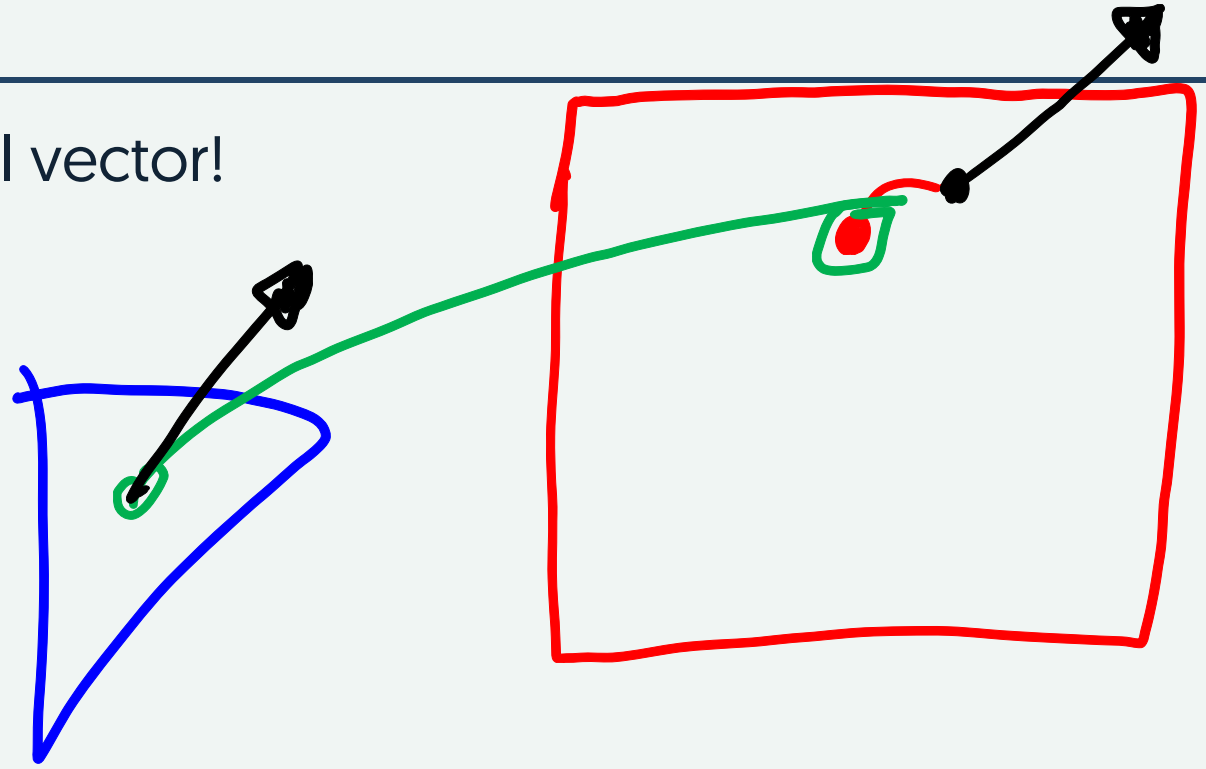
G = y direction

B = z direction

middle value (128/256) = 0

need to renormalize

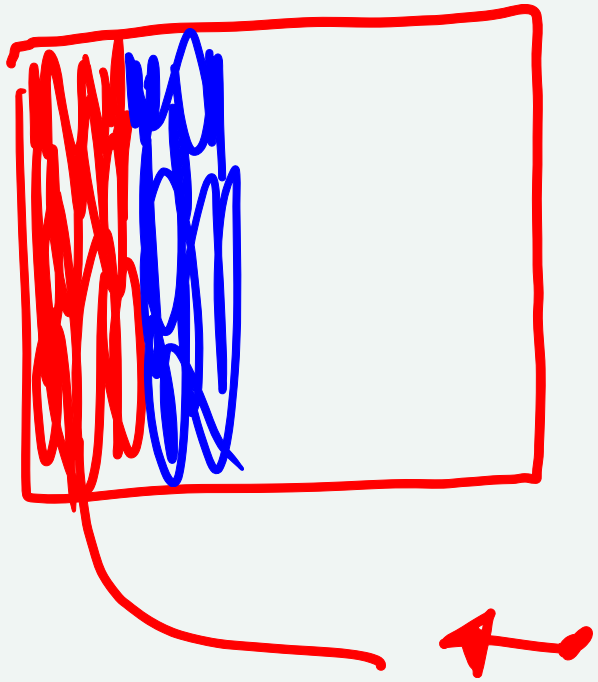
vector relative to real normal

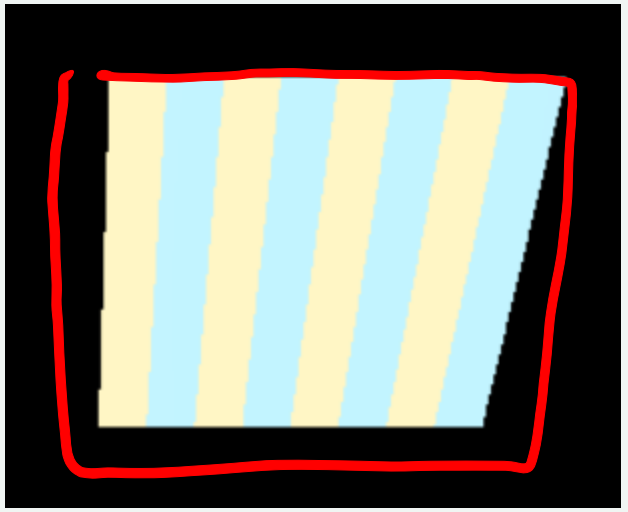


Wavy with Normal Map

One square (2 triangles)

Simple to paint pattern



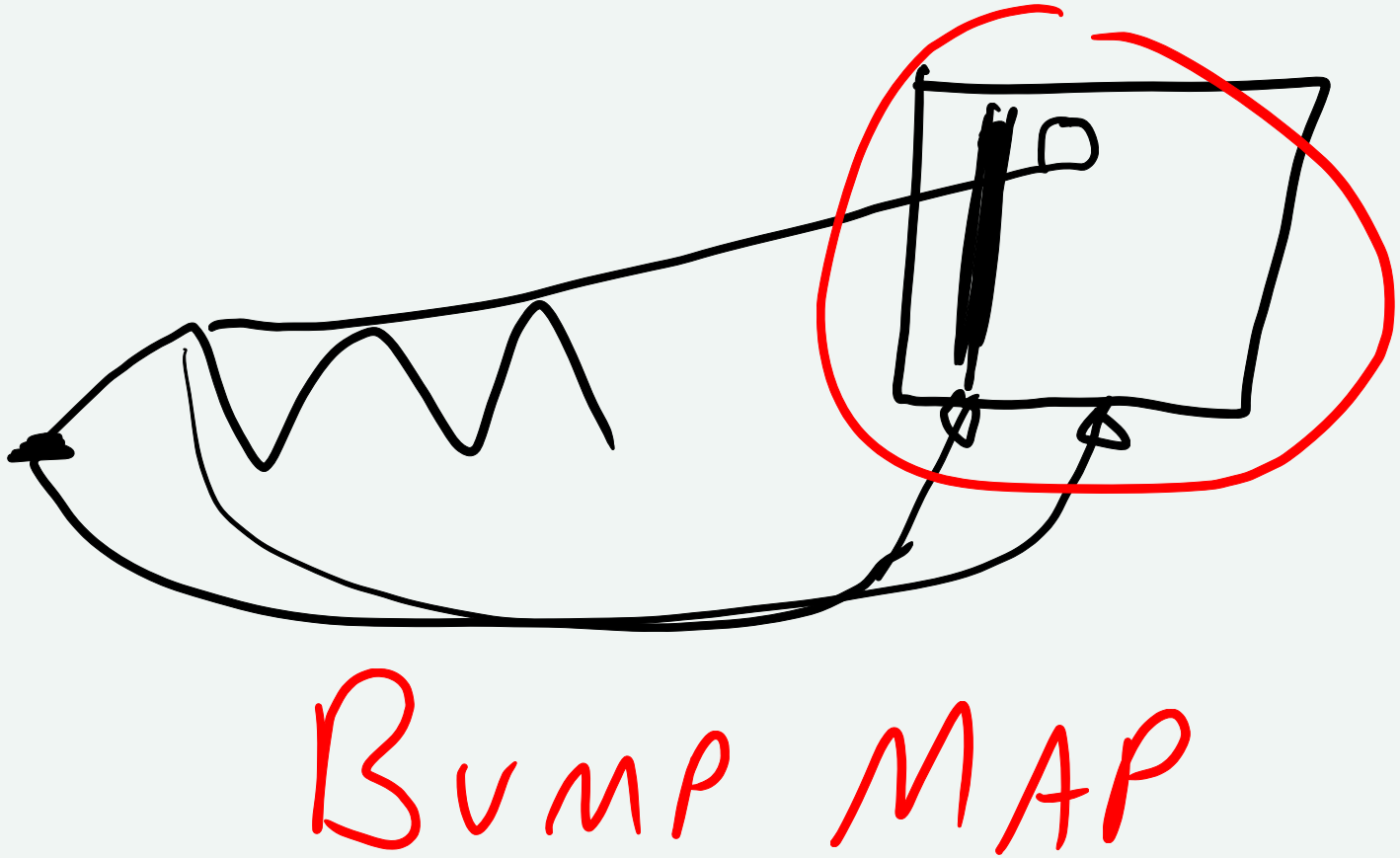


It's a pain to do XYZ

hard to think about

paint 1 dimension instead?

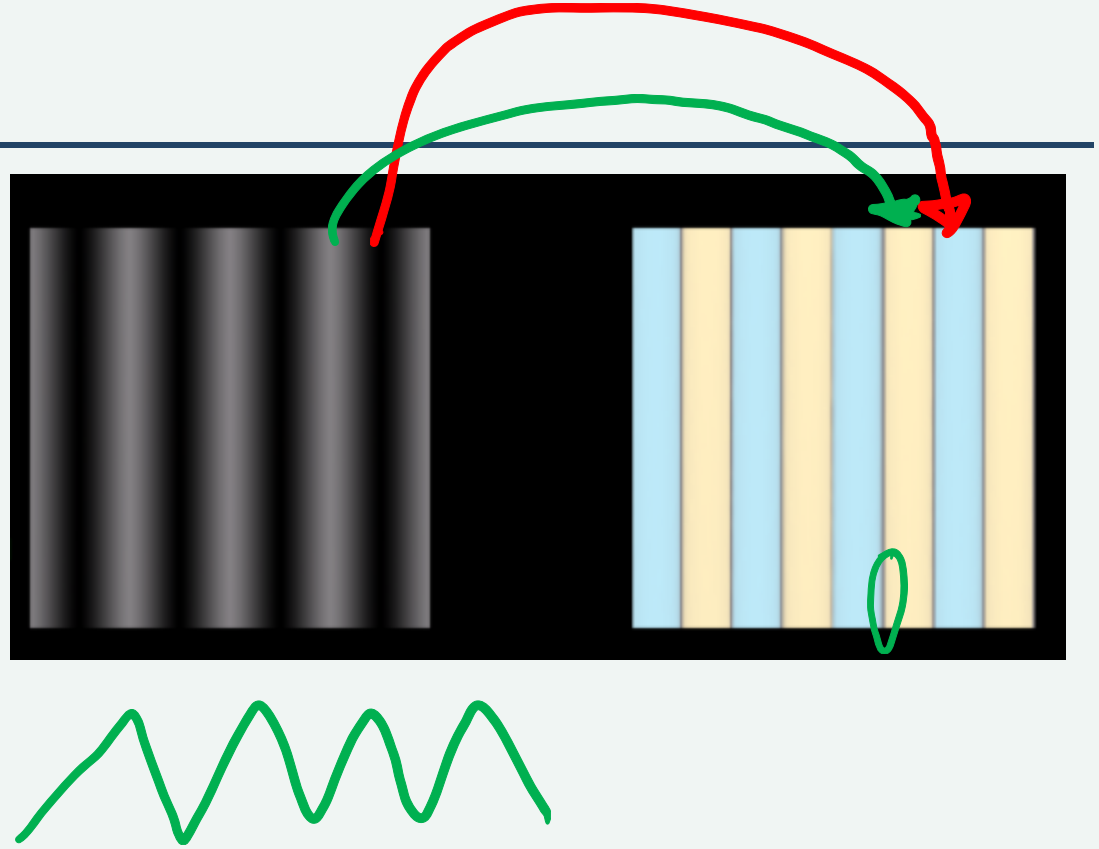
height field



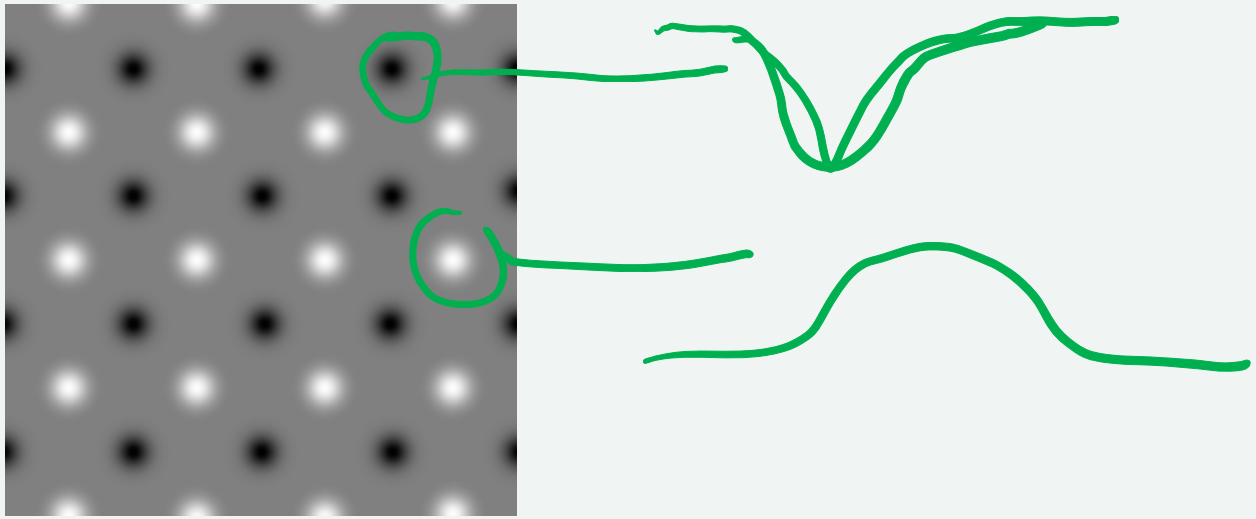
Bump Map

One square (2 triangles)

Simple to paint pattern



A more realistic example

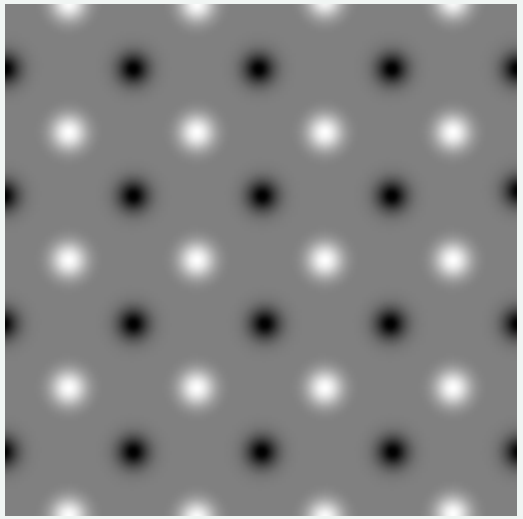


Gray = middle, black=down, white=up

(it's all relative)

A more realistic example

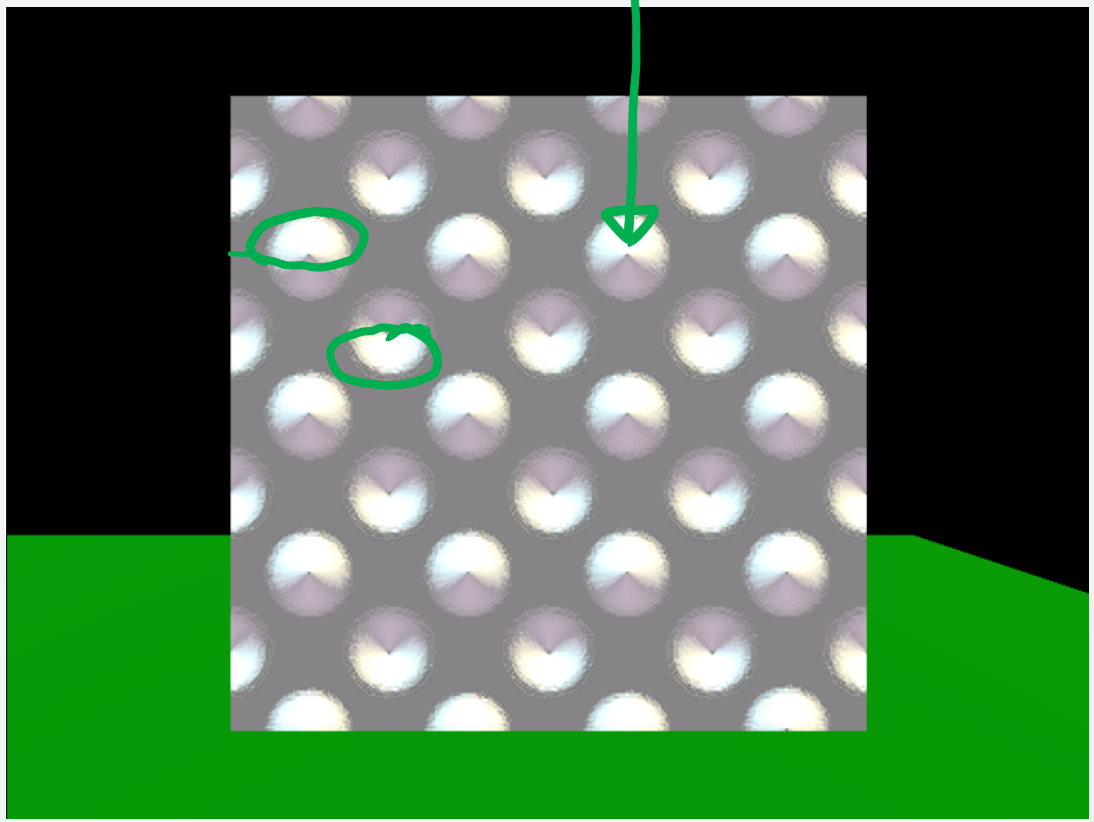
white



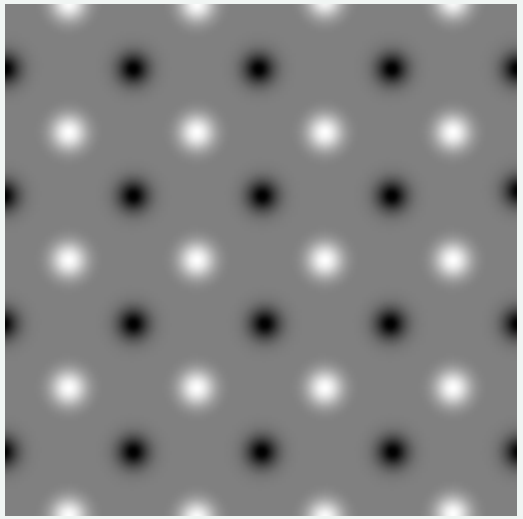
Bright from above

Dark from Below

Lighting not reflection!



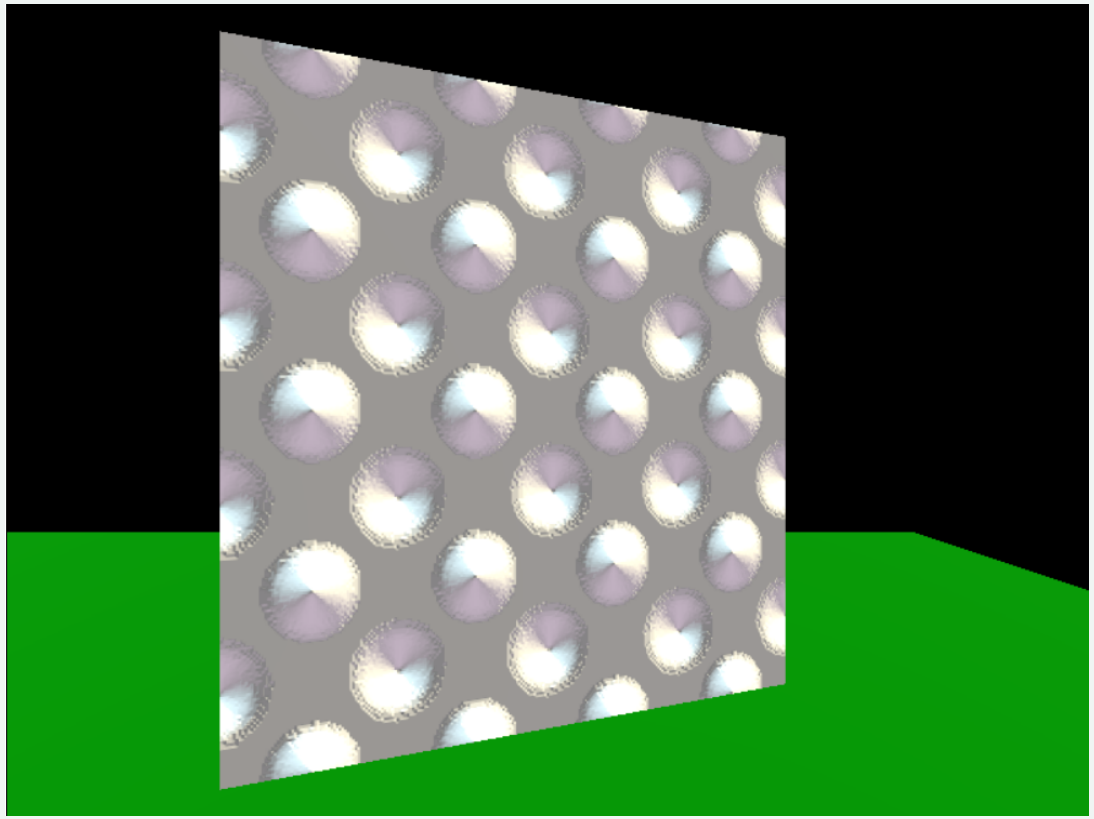
A more realistic example



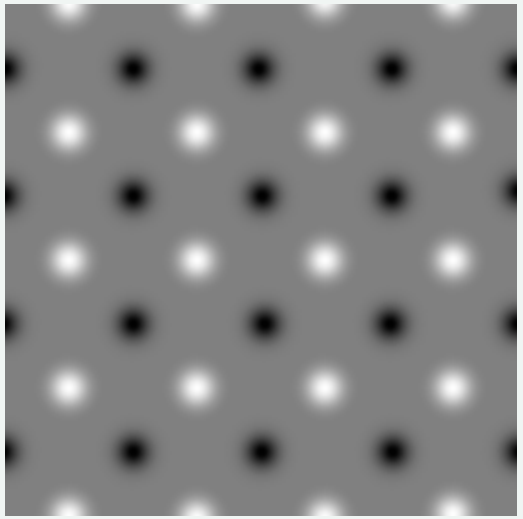
Bright from above

Dark from Below

Lighting not reflection!

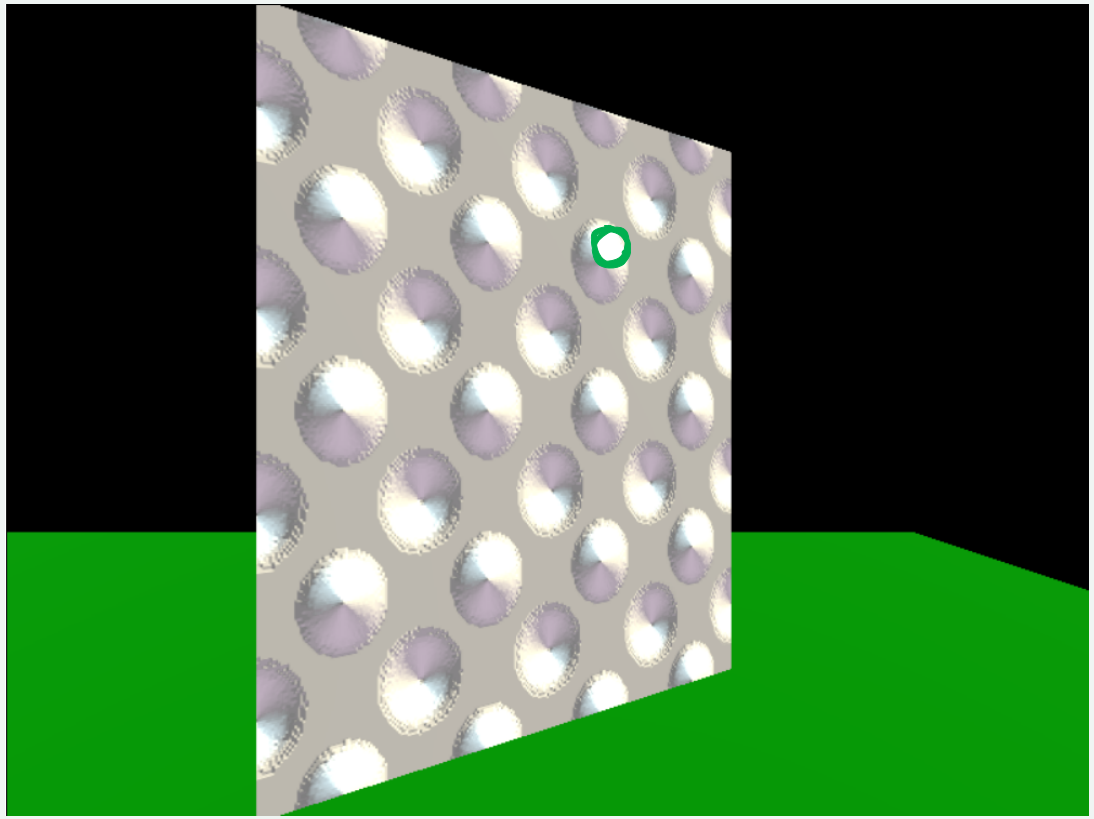


A more realistic example

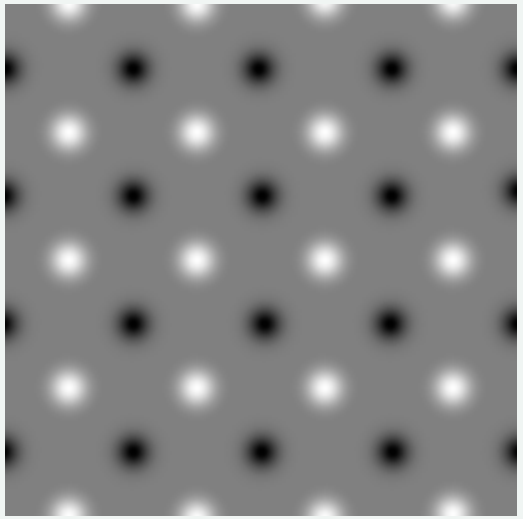


In motion, it can be convincing

(if you don't look too close)



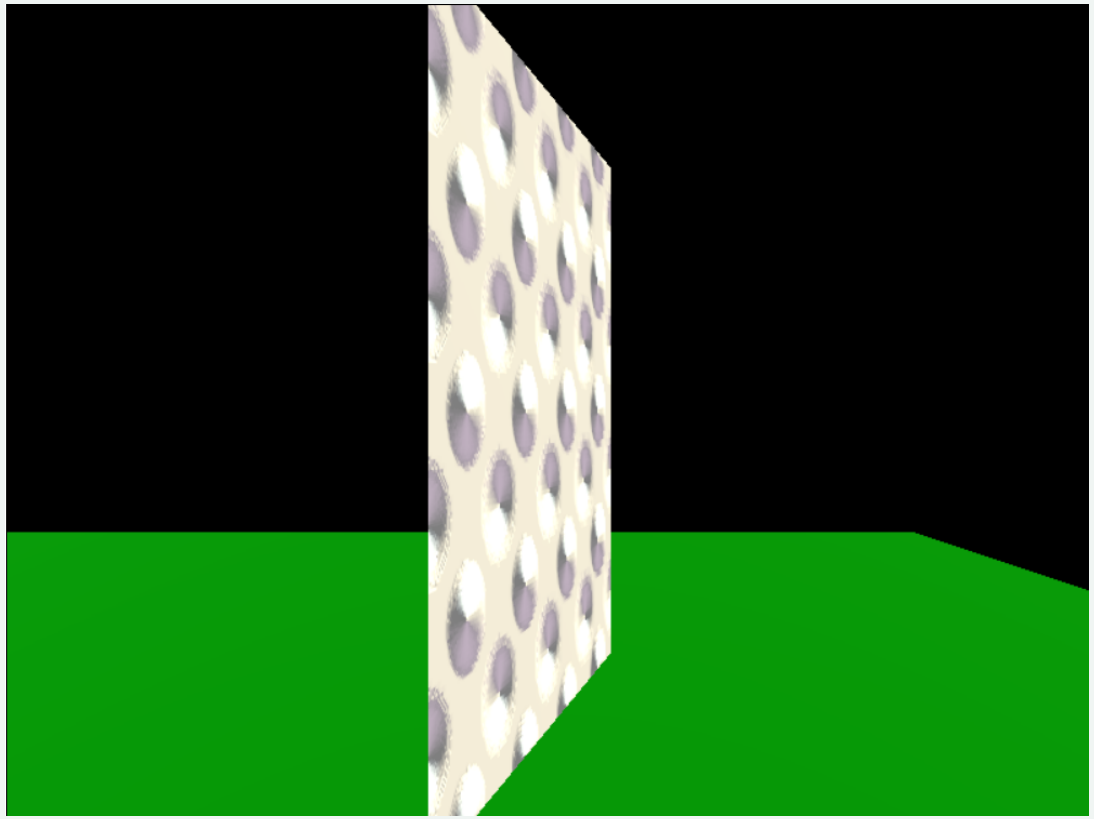
Where it breaks



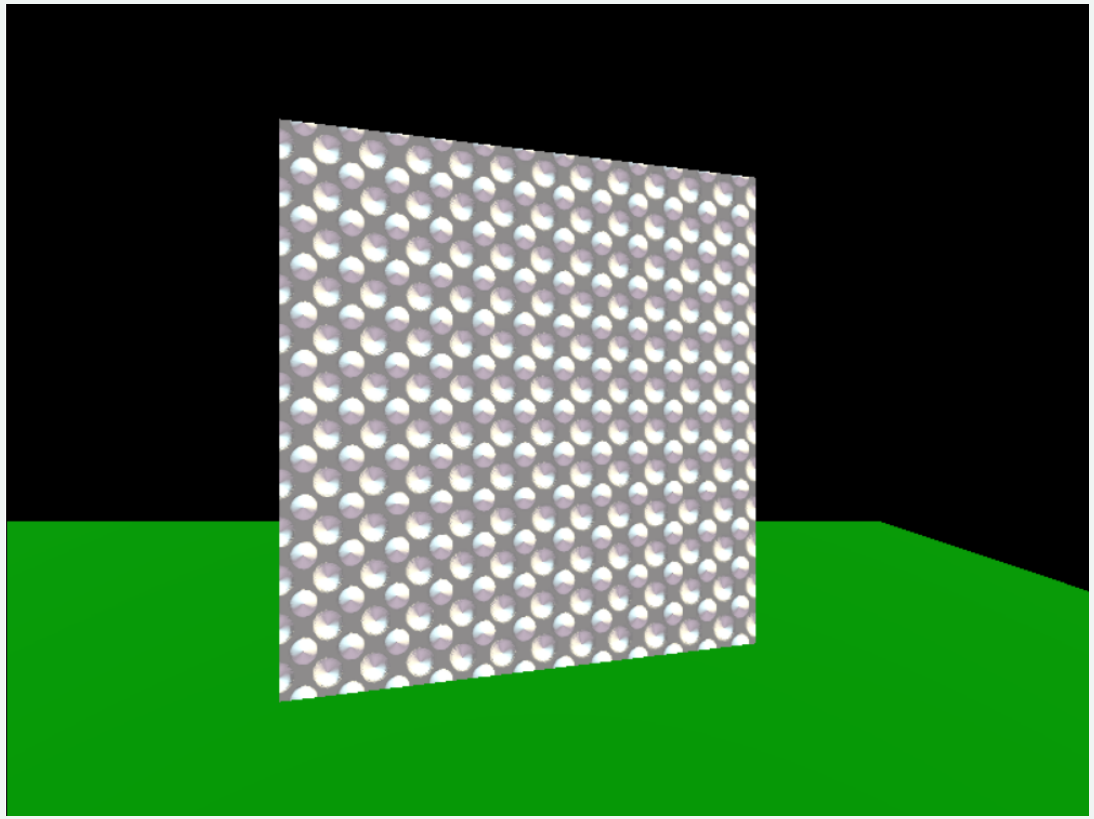
Doesn't change the shape!

It's still flat

Doesn't change the silhouette



Works better if subtle/small/moving



Easy in THREE.js!

```
let bumps = new T.TextureLoader().load("dots-bump.png");
let mat = new T.MeshStandardMaterial({bumpMap:bumps});
```

Normal Maps and Bump Maps

Good

Easy to specify surface details

Doesn't actually change shape

Gets basic lighting effects

Works with lighting

Easy in THREE

Bad

Doesn't change side view

Doesn't cause occlusions

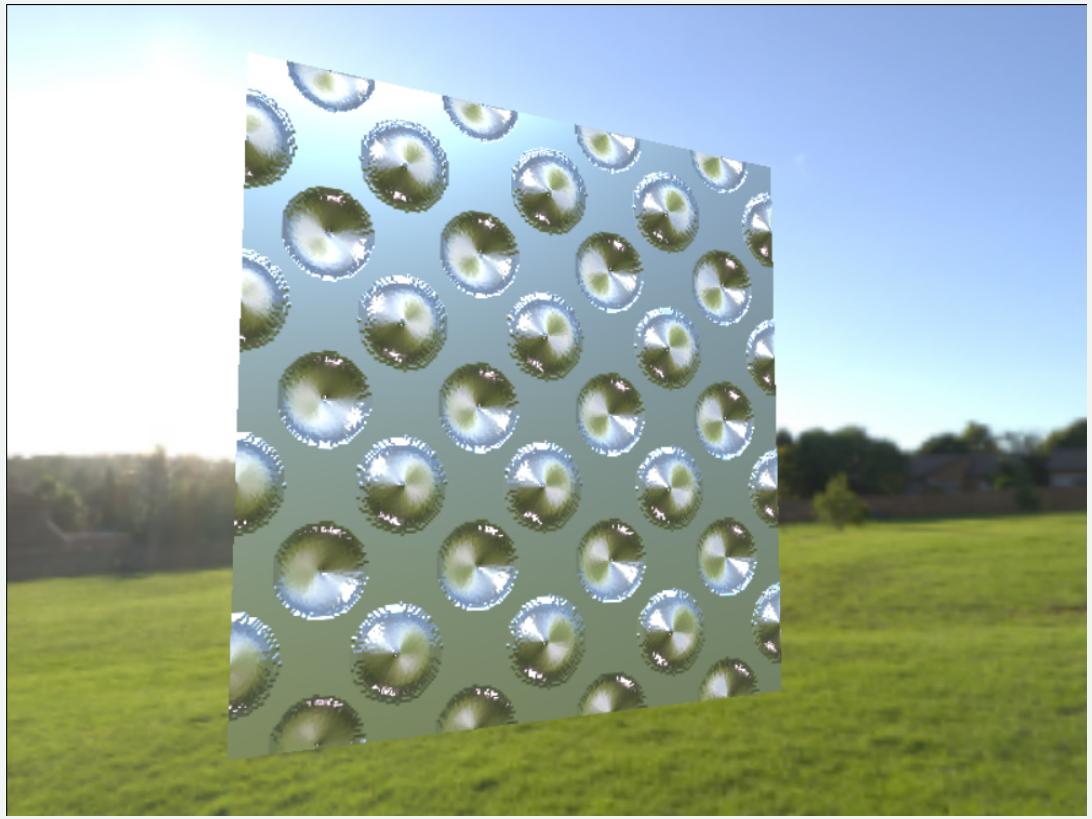
Doesn't work for big effects

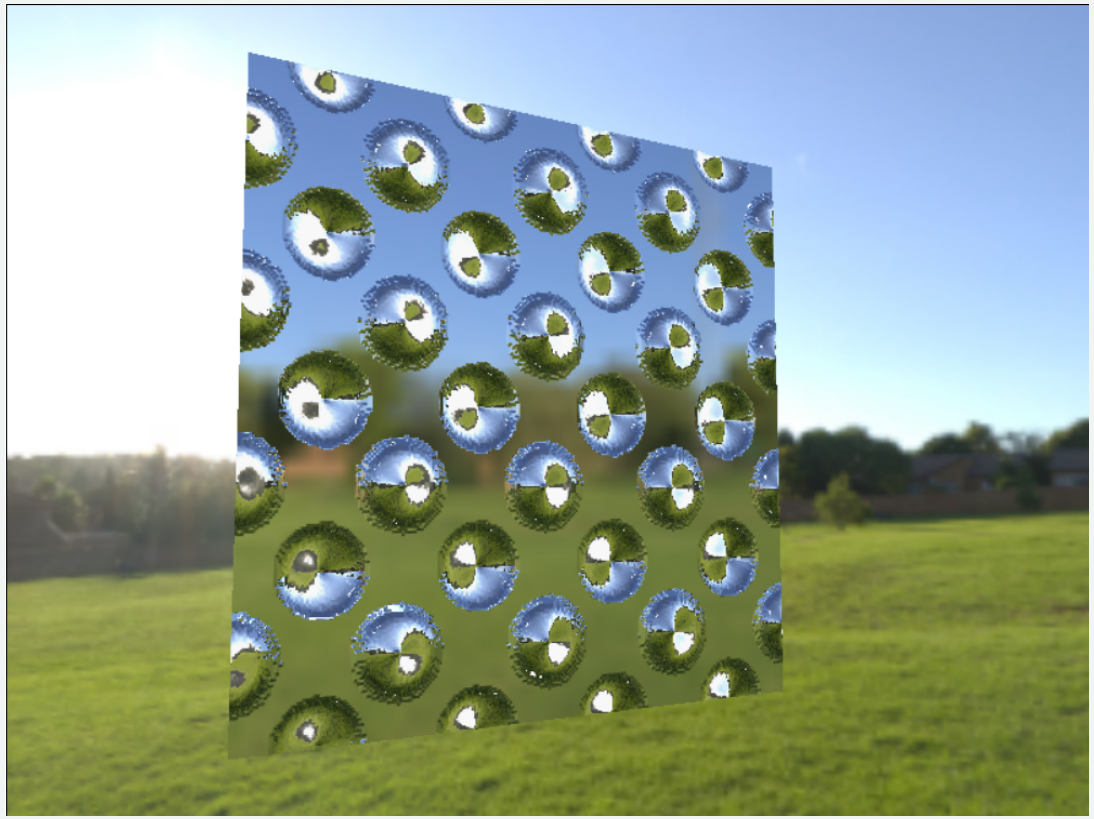
Doesn't cause shadows

Need something else for reflection

(wait a bit)

Coming Attraction...

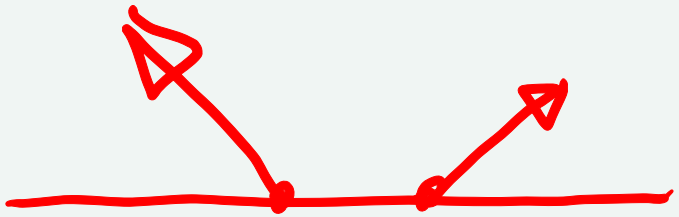


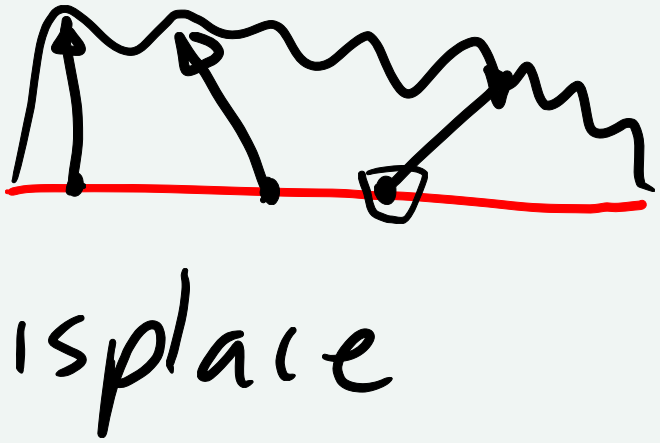


Change geometry?

RGB -> XYZ - could be a displacement

Displacement Maps!

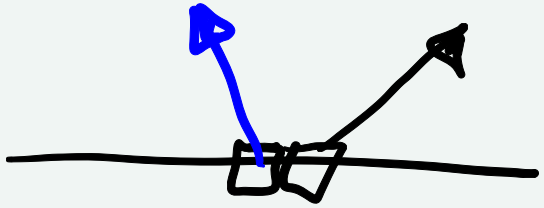




Why not Displacement Maps?

Much harder to implement

- actually moves pixels (can't do per-pixel)
- may cause gaps
- doesn't fit hardware model



Summary

- Fake Normals to get Smooth Surfaces
- Fake Normals to get Bumpy Surfaces
  - Bump Maps
  - Normal Maps
- Normal/Bump Maps are not ~~Displacement Maps~~