
CSC317 Computer Graphics
… starts at 11:10am
Some Slides/Images adapted from Marschner and Shirley

Today: Transformations and Shaders

Transforms and Shaders

Reminder – Rasterization

Introduction to the Graphics Pipeline

Transformations

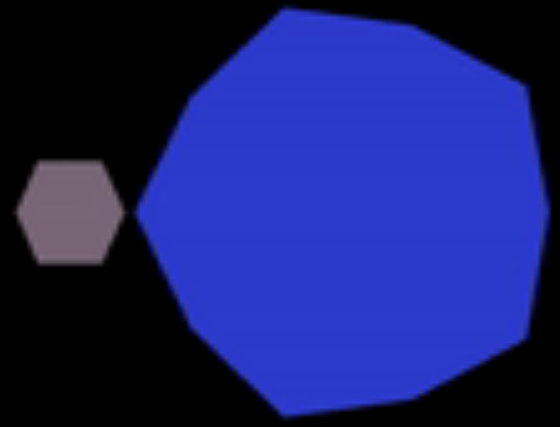
Normal and Bump Mapping

Perlin Noise

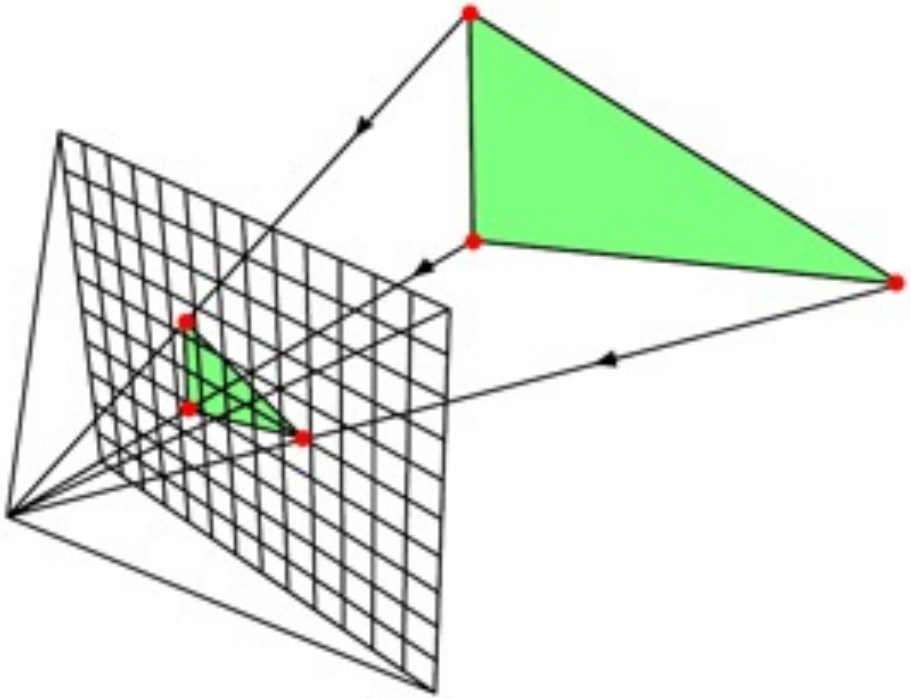
Announcements

Assignment 6 out now, due November 1st

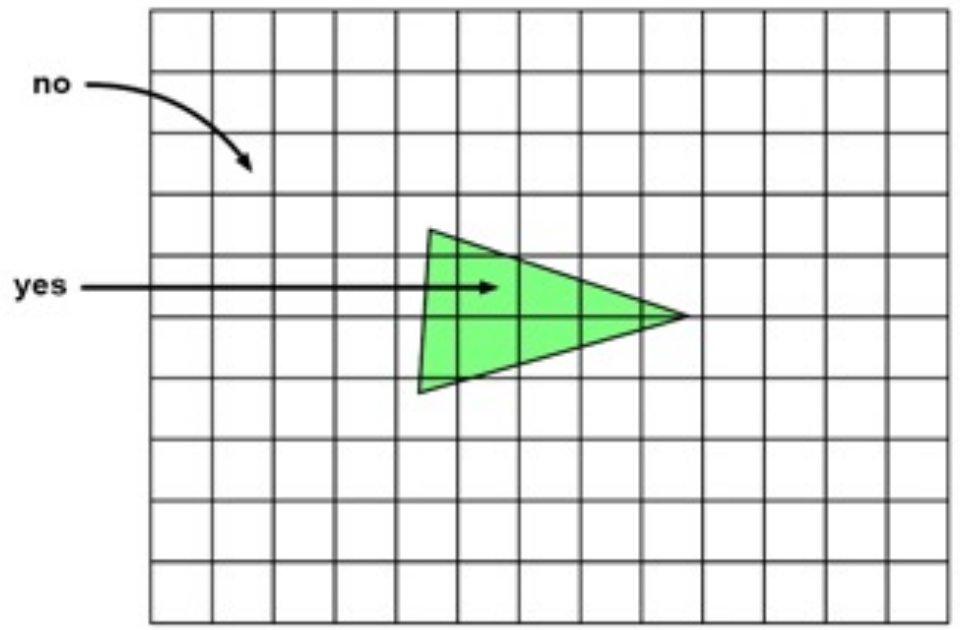
Any Questions ?



Rasterization



1. Project Vertices to Image Plane

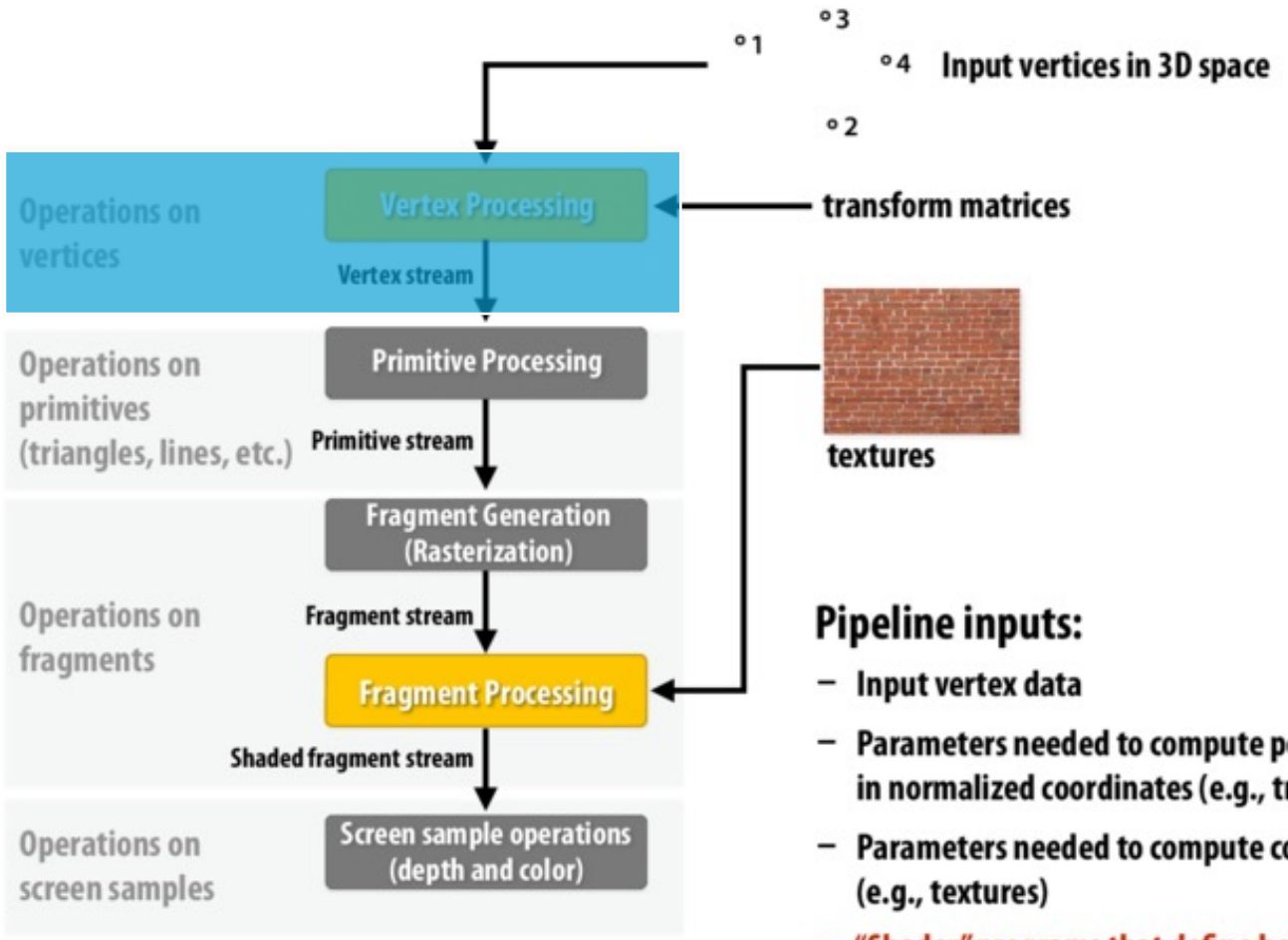


2. Turn on pixels inside triangle

https://www.scratchapixel.com/lessons/3d-basic-rendering/rasterization-practical-implementation

Modern Graphics Pipeline

OpenGL/Direct3D graphics pipeline *



Pipeline inputs:

- Input vertex data
- Parameters needed to compute position on vertices in normalized coordinates (e.g., transform matrices)
- Parameters needed to compute color of fragments (e.g., textures)
- "Shader" programs that define behavior of vertex and fragment stages

* several stages of the modern OpenGL pipeline are omitted

What is a linear transformation?

A: For vectors, a linear transformation is any operation performed by a matrix

$$Ax = b$$

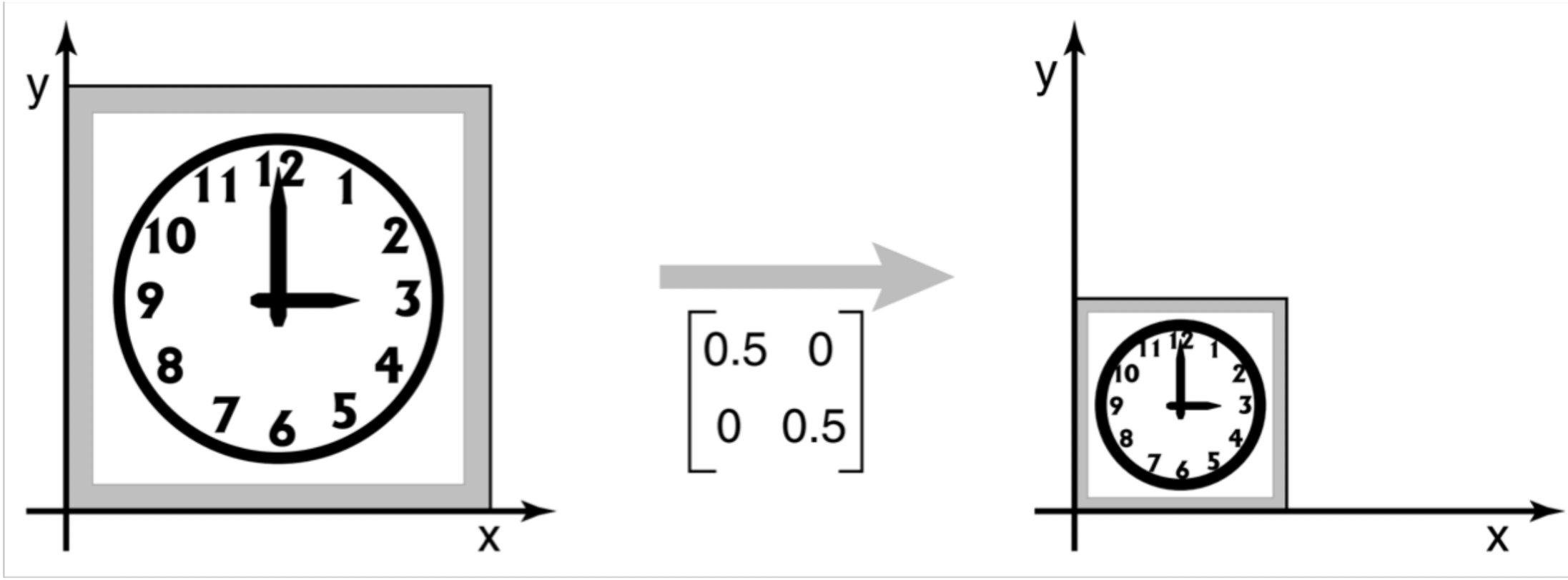
2D Linear Transformations

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_{11}x + a_{12}y \\ a_{21}x + a_{22}y \end{bmatrix}$$

2D Linear Transformations - Scale

$$\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \end{bmatrix}$$

Linear transformations in 2D: Scale

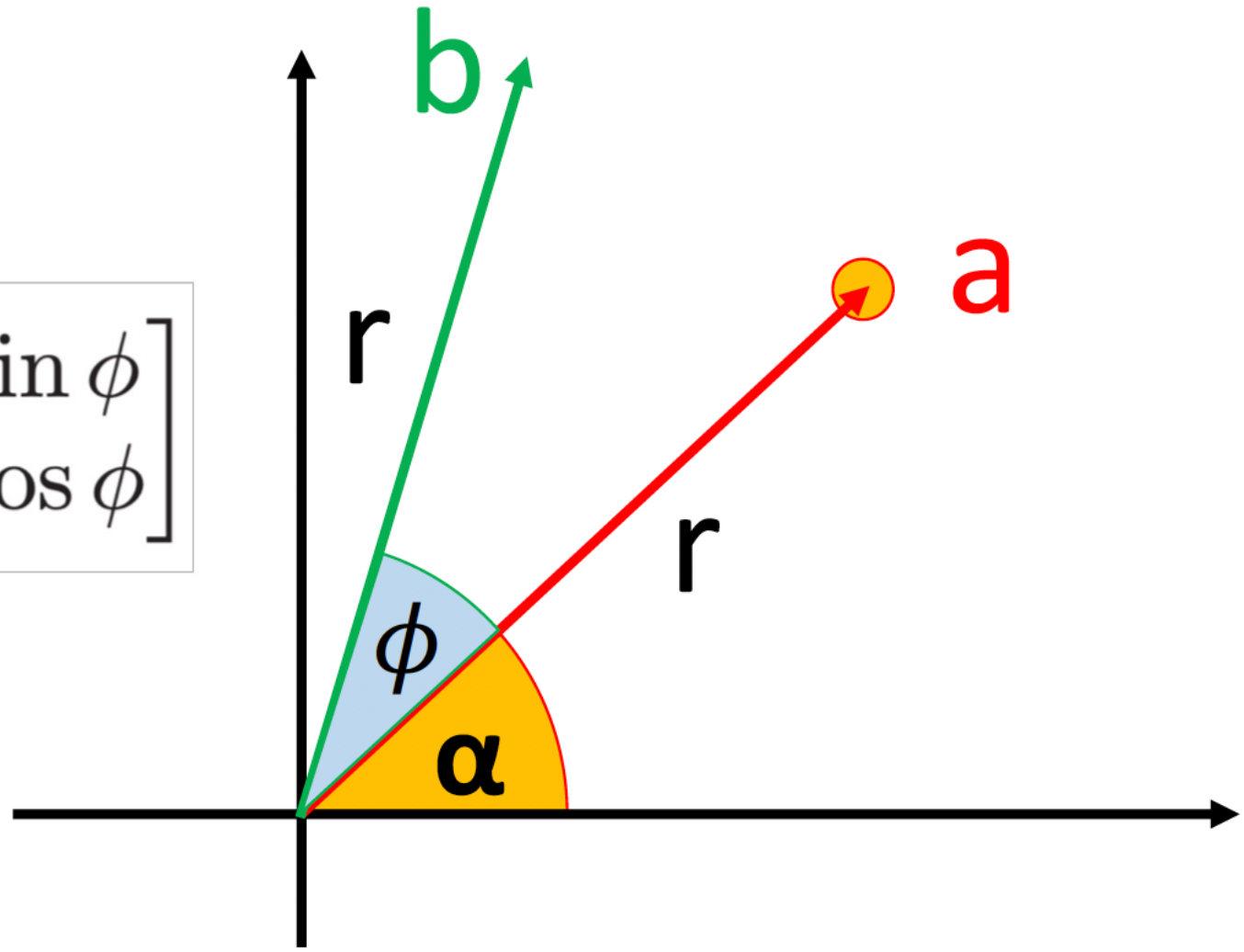


When Sx = Sy we say the scaling is uniform

2D Linear Transformations - Rotation

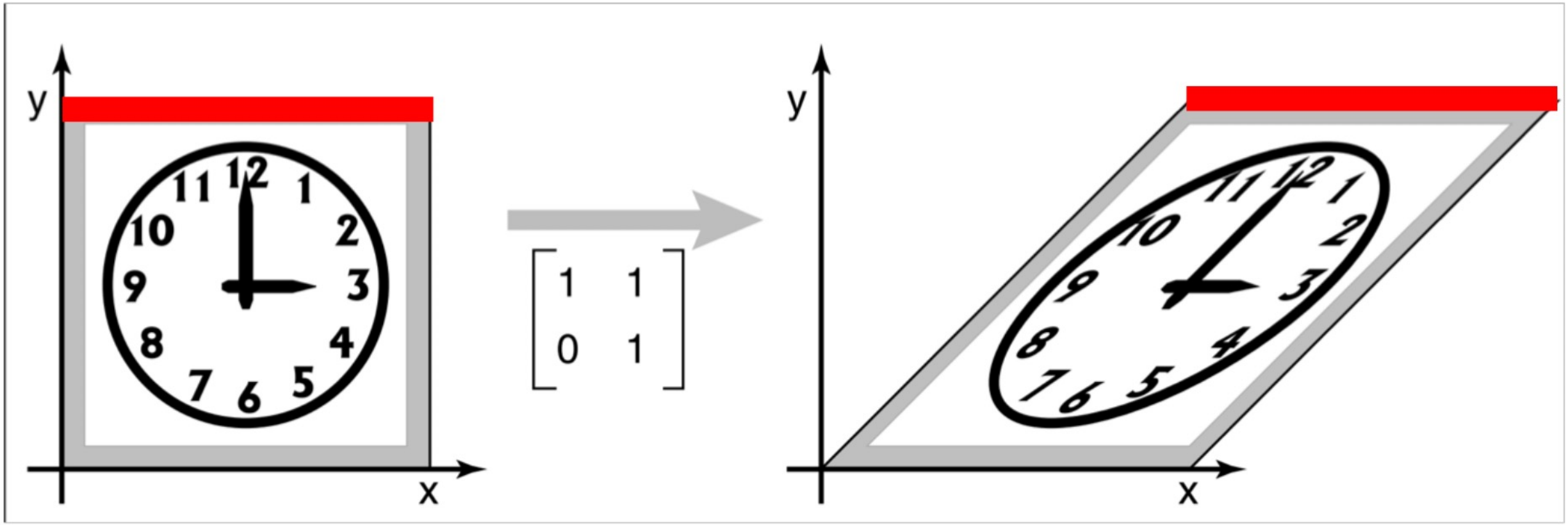
$$\text{rotate}(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}$$

$b^T b = a^T R^T R a = a^T a$



2D Linear Transformations - Shear

$$\begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x + sy \\ y \end{bmatrix}$$



These are always the same length

2D Linear Transformations - Translation

$$T \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \end{bmatrix}$$

2D Affine Transformations - Translation

$$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11}x + a_{12}y + t_x \\ a_{21}x + a_{22}y + t_y \\ 1 \end{bmatrix}$$

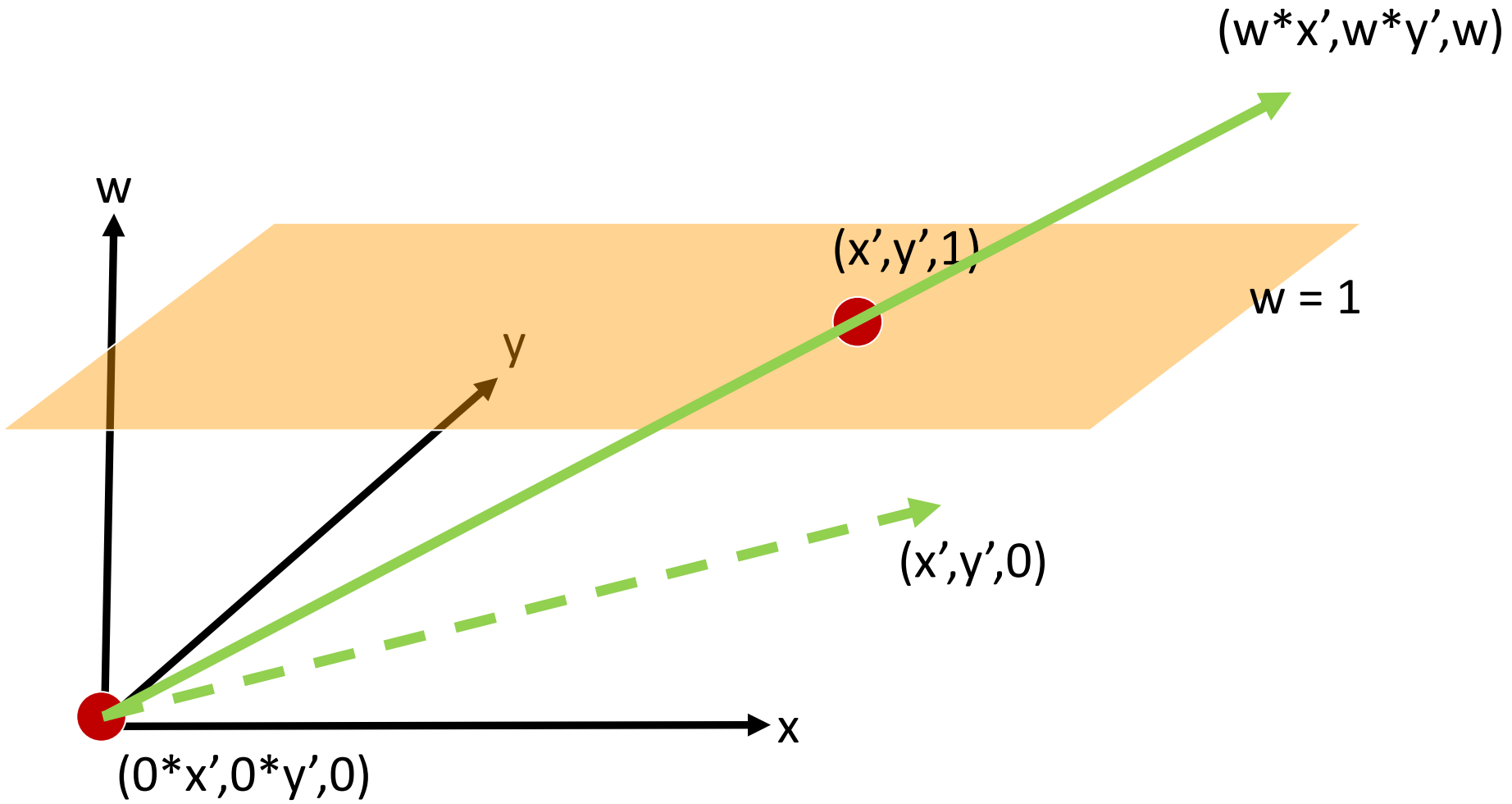
$$Ax + t = b$$

$$\begin{bmatrix} x \\ y \end{bmatrix}$$

Considered as a point in 3D homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Geometric Intuition



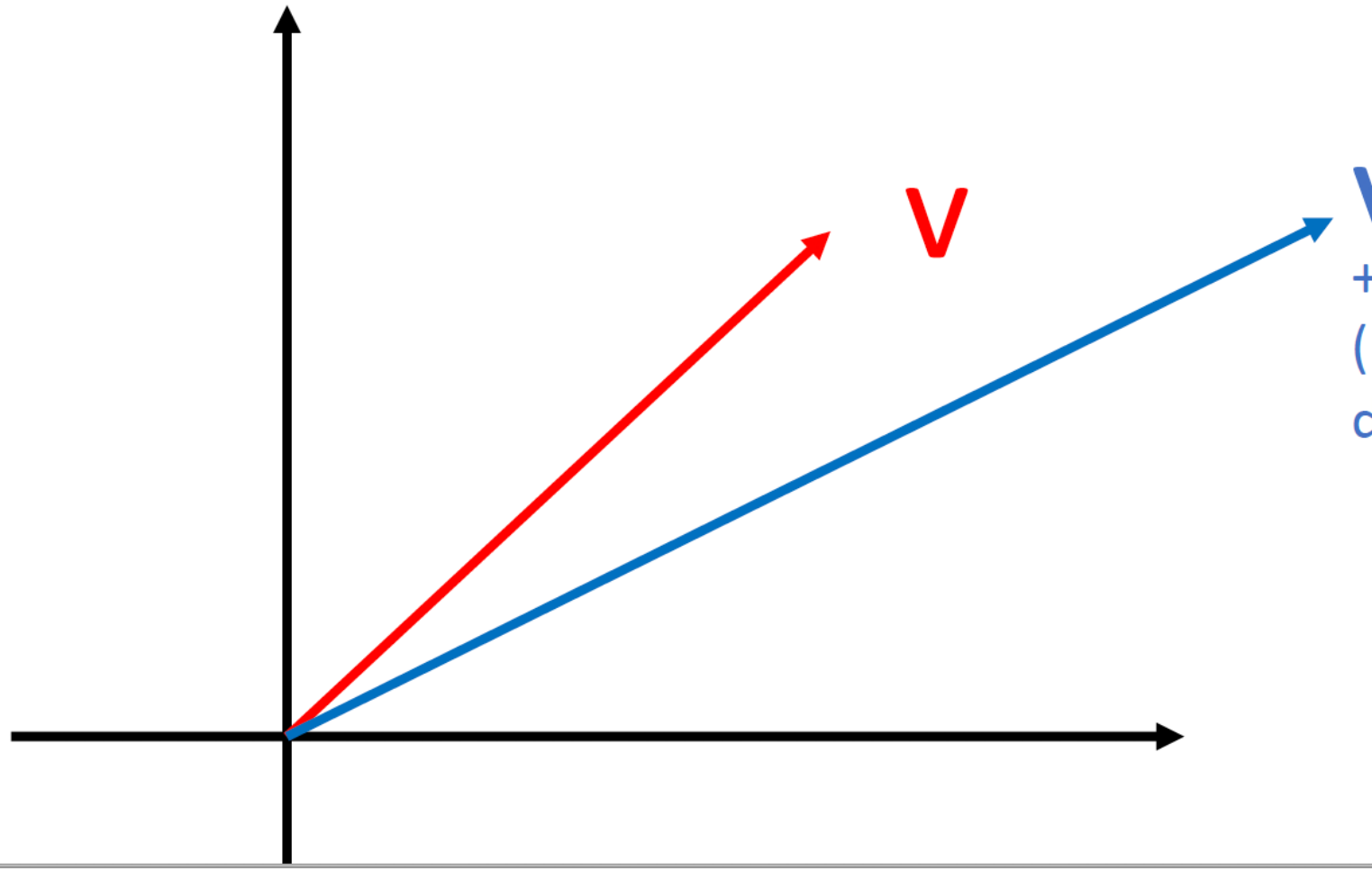
What about vectors?

$$\begin{bmatrix} x \\ y \end{bmatrix}$$

Considered as a ***vector*** in 3D homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$$

We don’t want to be able to translate a vector

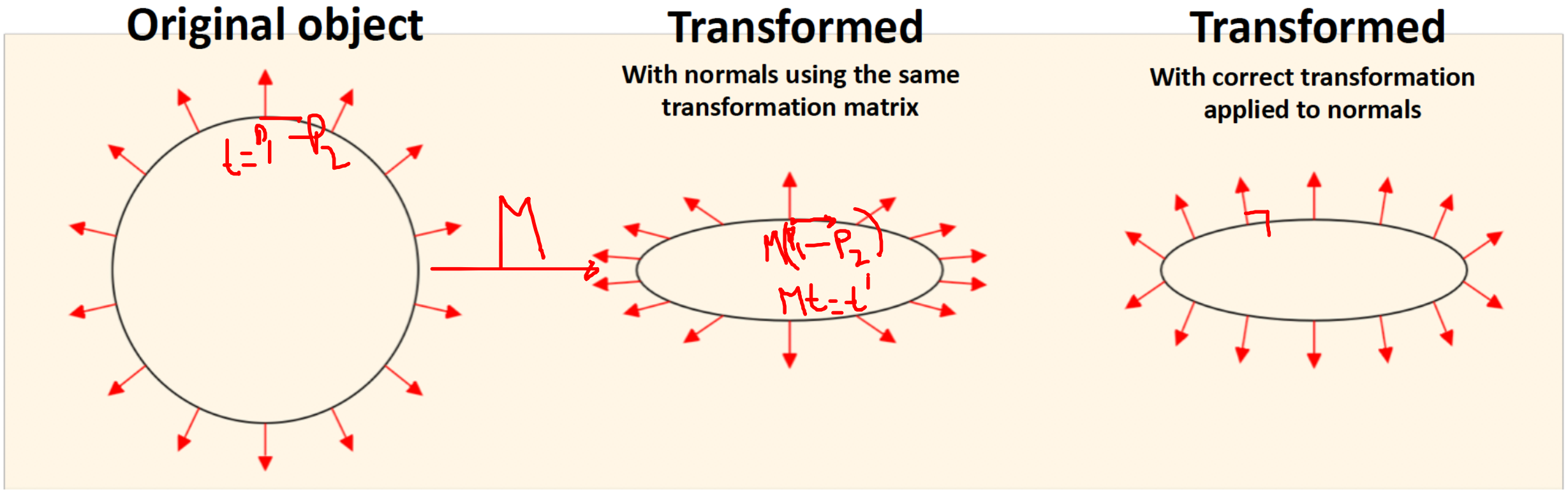


3D Linear Transformations

$$Ax = b$$

Speaking of vectors, do normals get transformed the same way an object does?

No, thank you for asking.



https://paroj.github.io/gltut/Illumination/Tut09%20Normal%20Transformation.html

What's the right way to transform a normal vector?

$$n'^T t' = 0$$

$$(Xn)^T (Mt) = 0$$

?

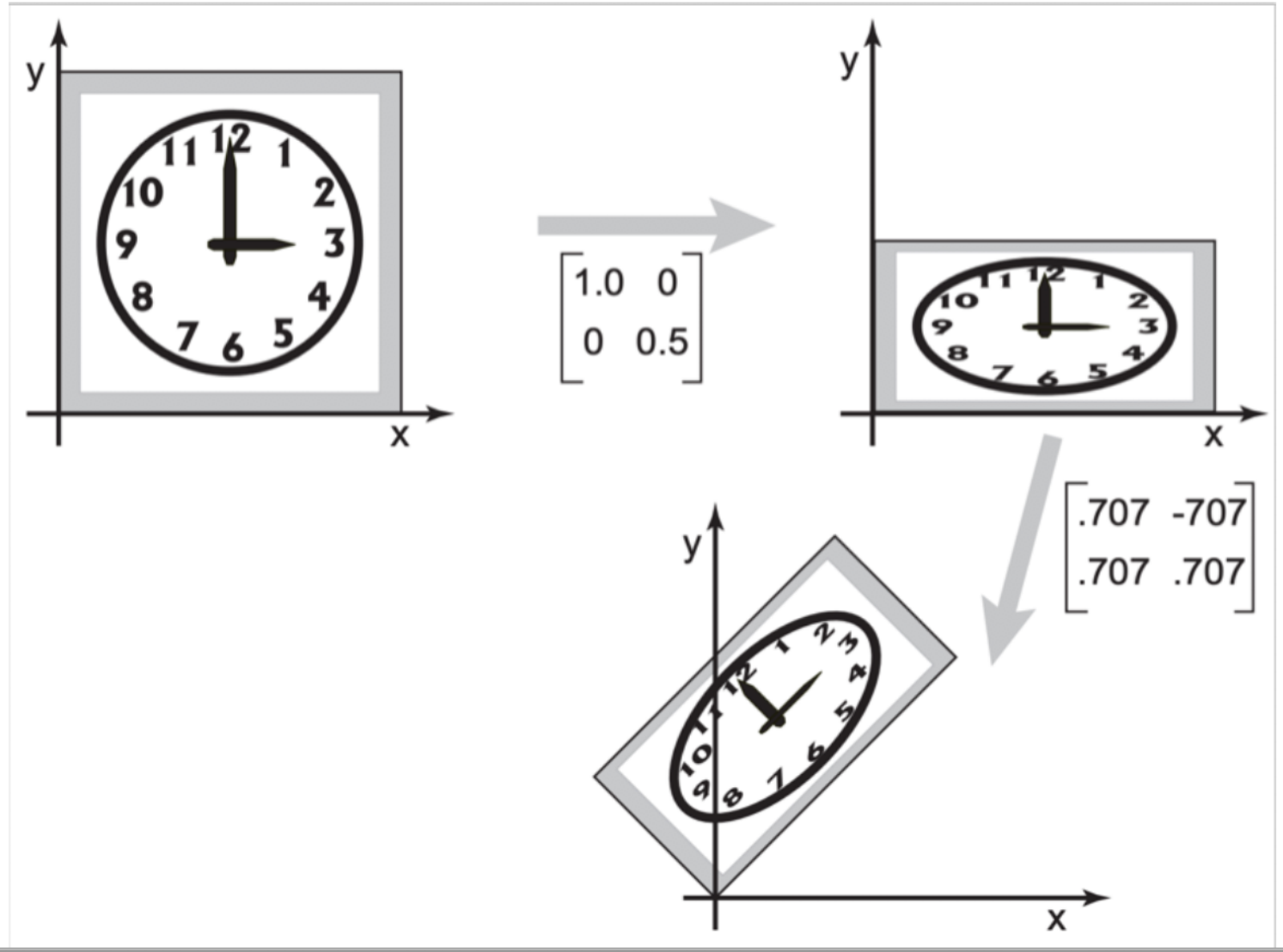
What's the right way to transform a normal vector?

We want $n^T X^T M t = 0$

So, if $X = (M^{-1})^T$ then,

$$n^T X^T M t = n^T M^{-1} M t = n^T t = 0$$

Composing transformations

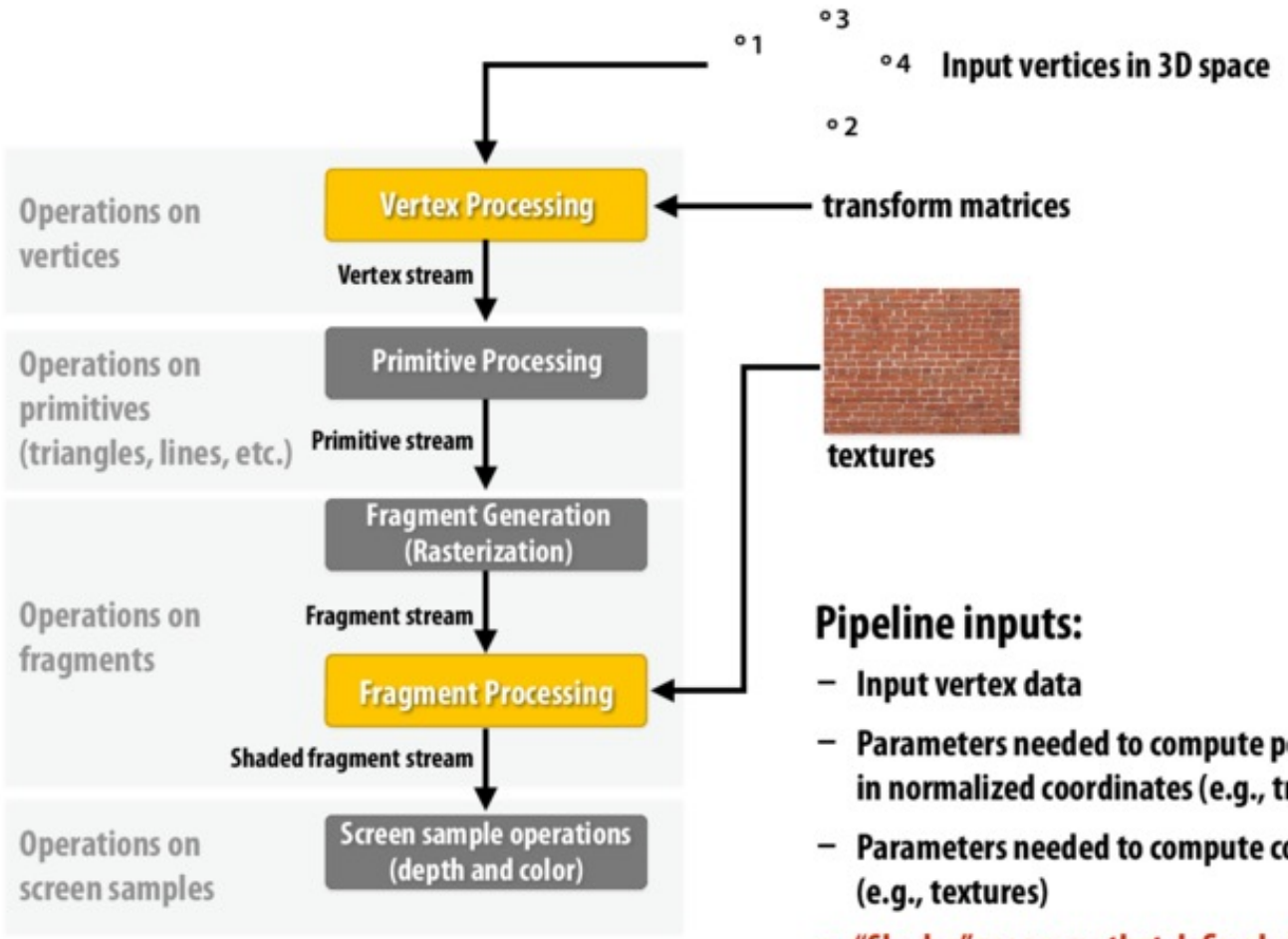


Composing transformations

$$\underbrace{\begin{bmatrix} .707 & -.707 \\ .707 & .707 \end{bmatrix}}_{\text{2nd transformation}} \underbrace{\begin{bmatrix} 1.0 & 0 \\ 0 & 0.5 \end{bmatrix}}_{\text{1st transformation}} = \begin{bmatrix} .707 & -.353 \\ .707 & .353 \end{bmatrix}$$

Modern Graphics Pipeline

OpenGL/Direct3D graphics pipeline *

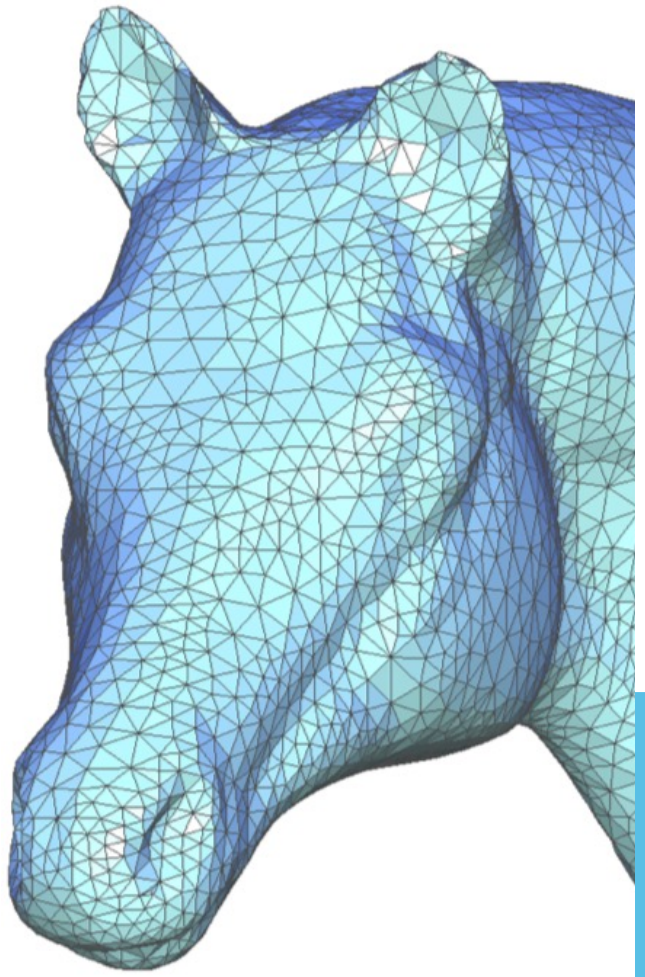


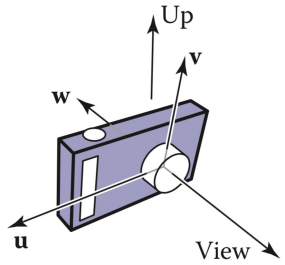
Pipeline inputs:

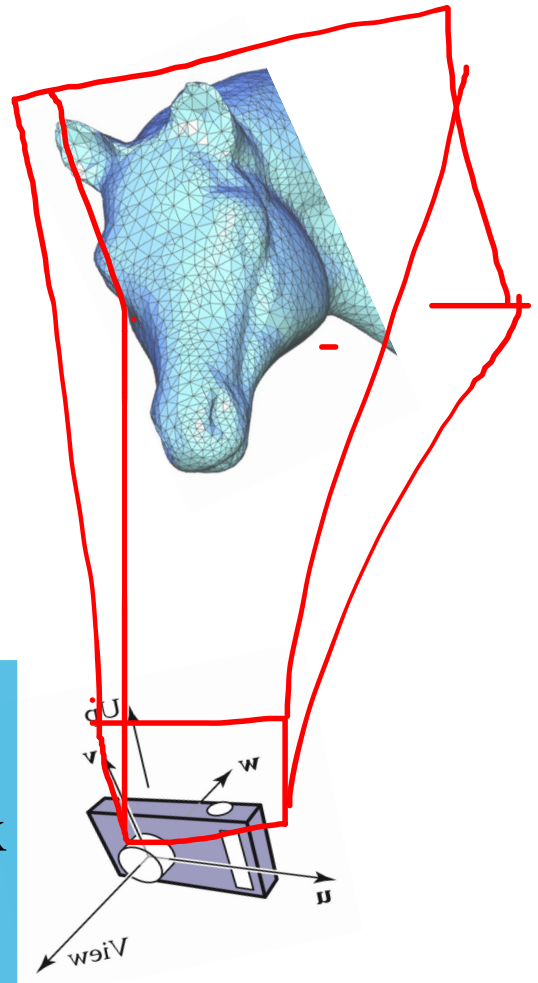
- Input vertex data
- Parameters needed to compute position on vertices in normalized coordinates (e.g., transform matrices)
- Parameters needed to compute color of fragments (e.g., textures)
- "Shader" programs that define behavior of vertex and fragment stages

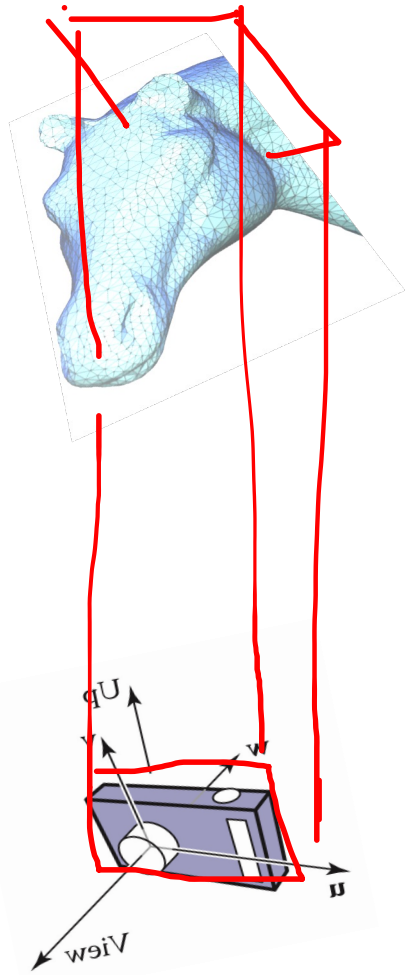
* several stages of the modern OpenGL pipeline are omitted

Getting Things Onto The Screen









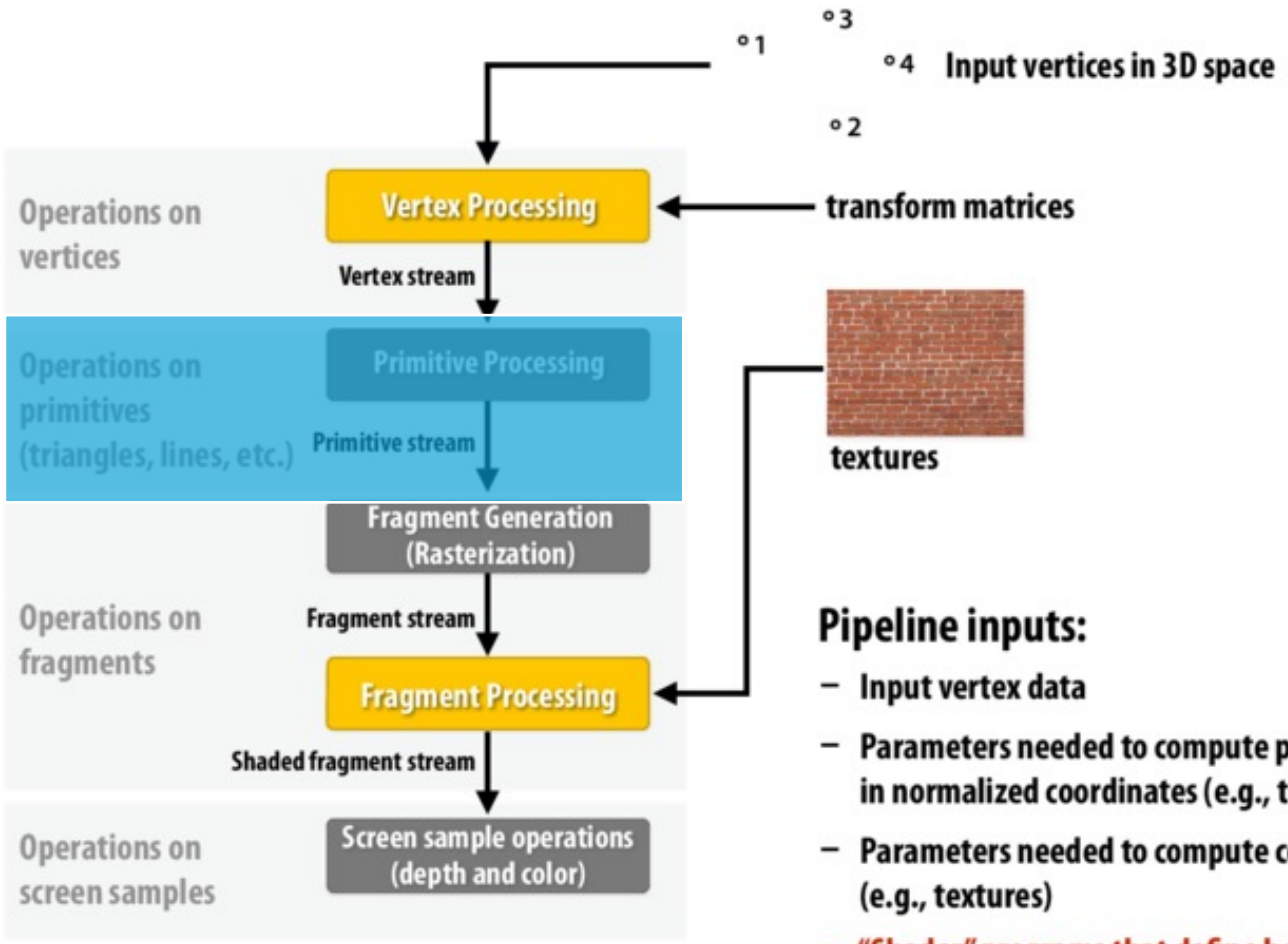
Open GL Combines these into the ModelView Matrix

M V P

Object Space | World Space | Camera Space | Canonical Space

Modern Graphics Pipeline

OpenGL/Direct3D graphics pipeline *

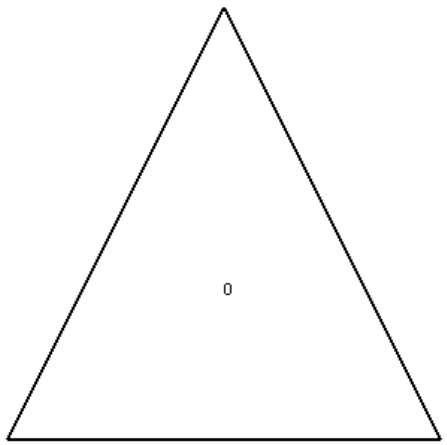


Pipeline inputs:

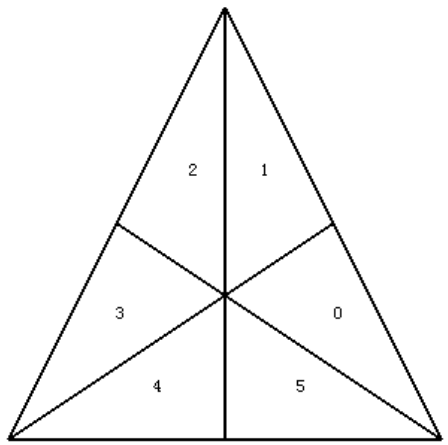
- Input vertex data
- Parameters needed to compute position on vertices in normalized coordinates (e.g., transform matrices)
- Parameters needed to compute color of fragments (e.g., textures)
- "Shader" programs that define behavior of vertex and fragment stages

* several stages of the modern OpenGL pipeline are omitted

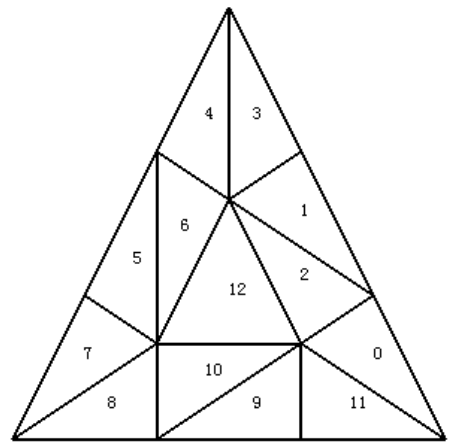
Tessellation Shader



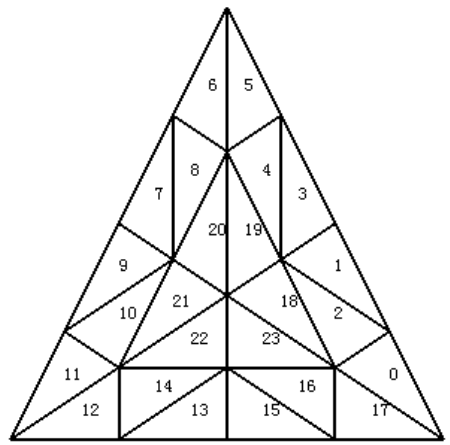
Level 1



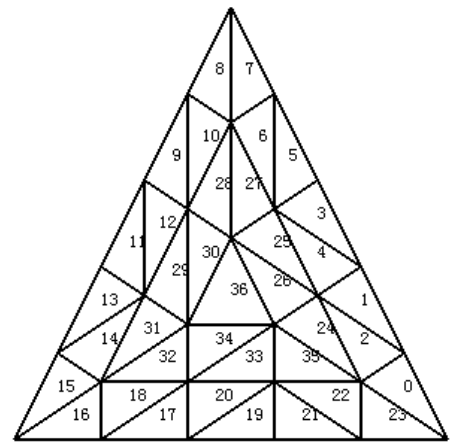
Level 2



Level 3



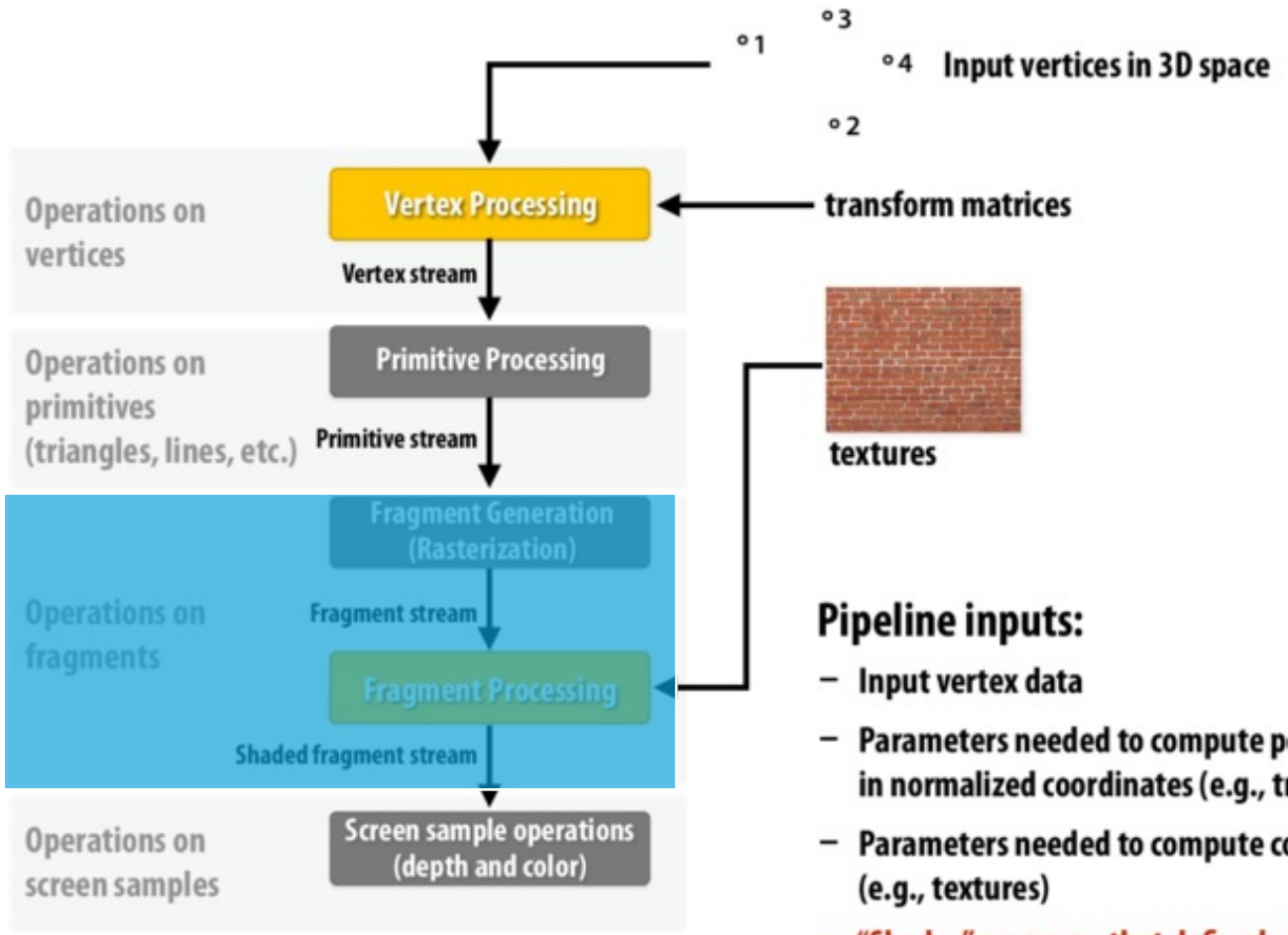
Level 4



Level 5

Modern Graphics Pipeline

OpenGL/Direct3D graphics pipeline *

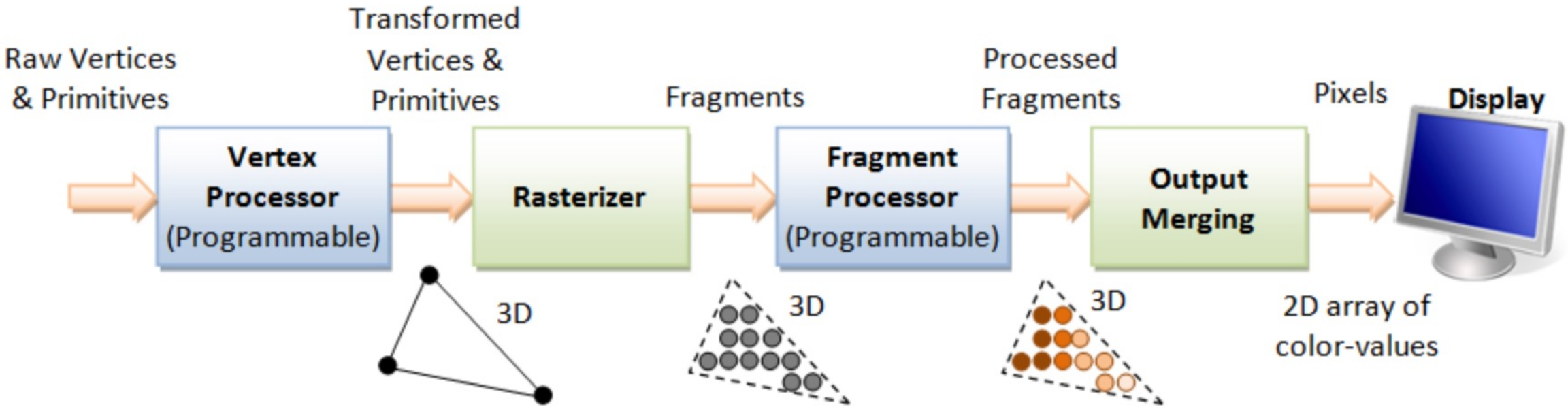


Pipeline inputs:

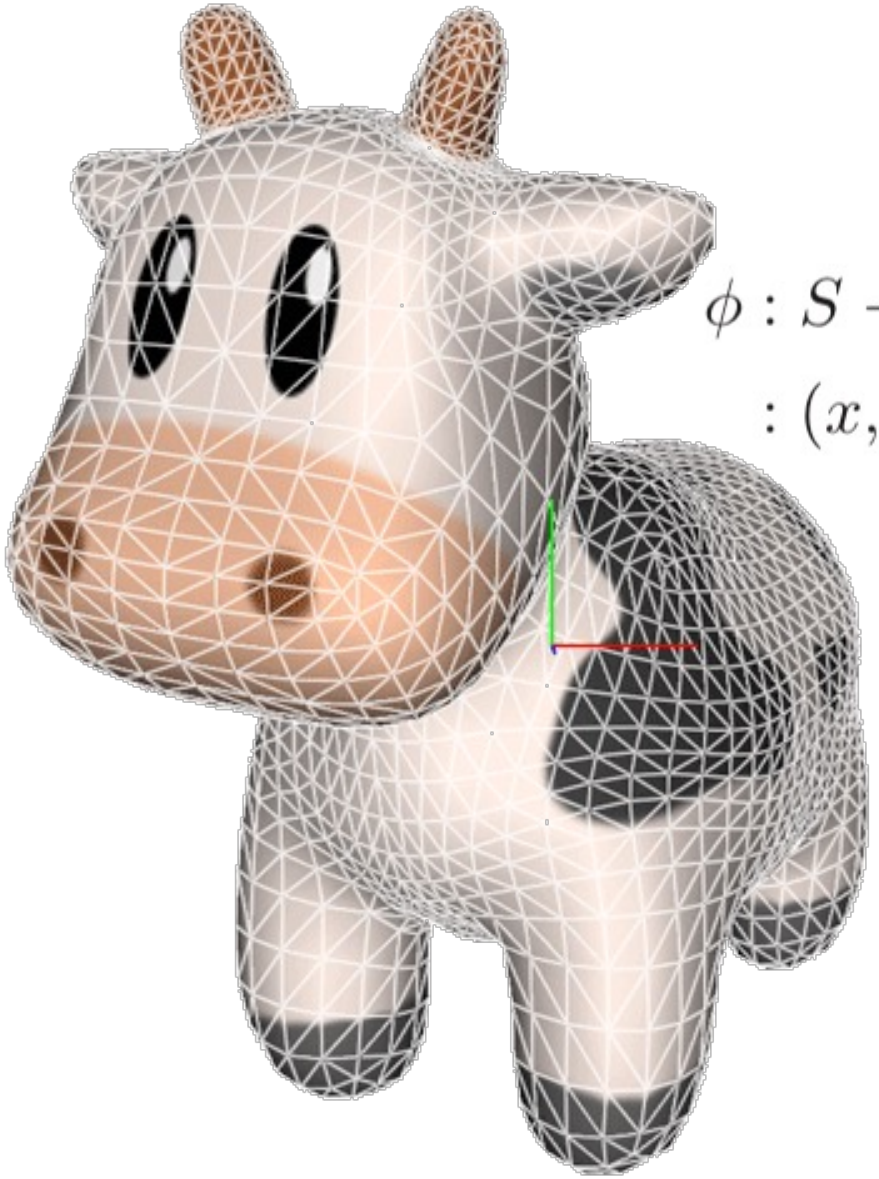
- Input vertex data
- Parameters needed to compute position on vertices in normalized coordinates (e.g., transform matrices)
- Parameters needed to compute color of fragments (e.g., textures)
- "Shader" programs that define behavior of vertex and fragment stages

* several stages of the modern OpenGL pipeline are omitted

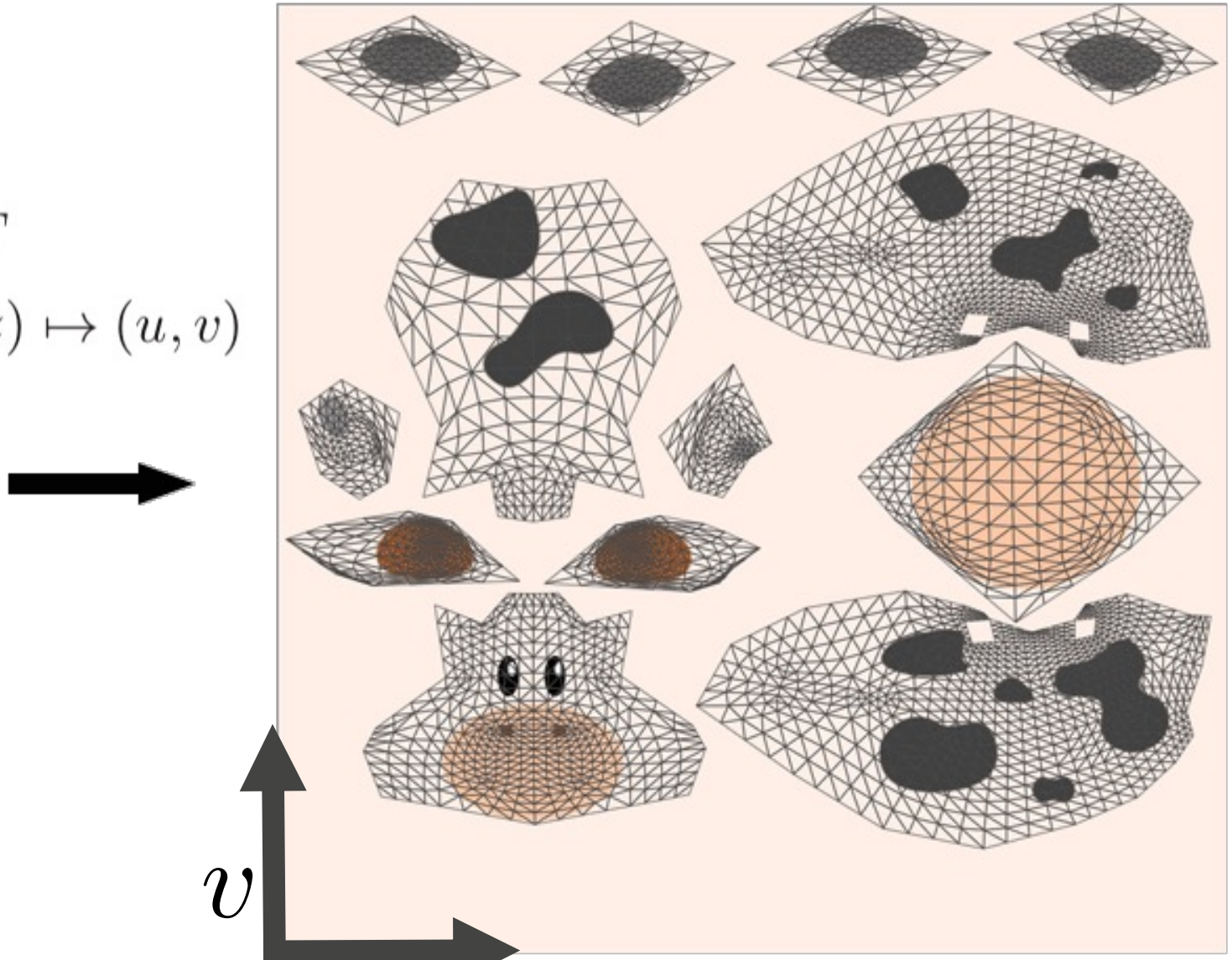
Fragment Shader



Texture coordinates



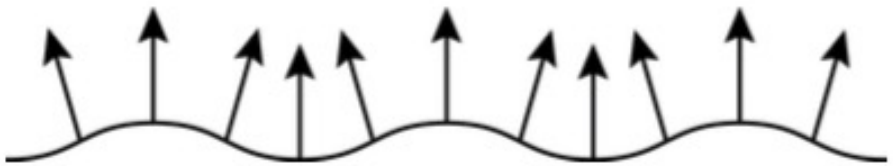
$$\phi : S \to T$$
$$: (x, y, z) \mapsto (u, v)$$

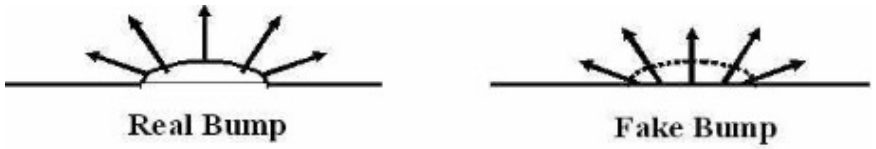


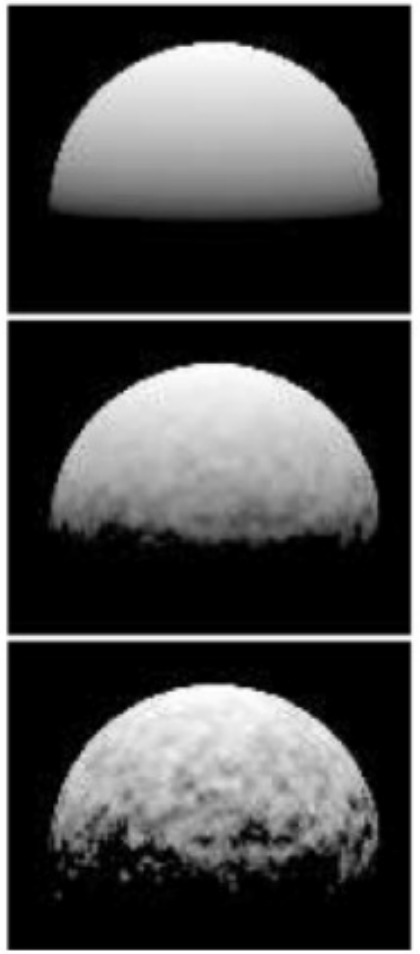
Normal Mapping

One of the reasons why we apply texture mapping:

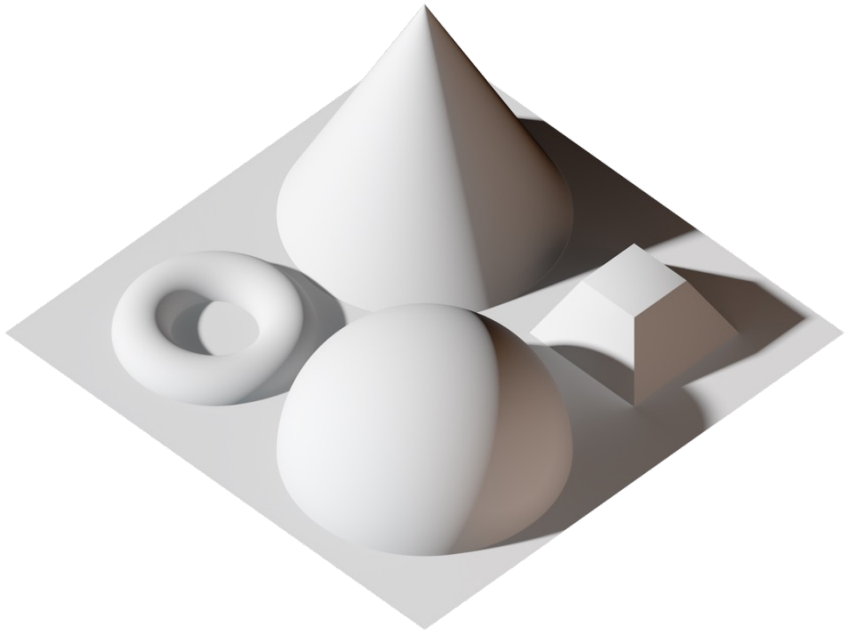
Real surfaces are hardly flat but often rough and bumpy. These bumps cause (slightly) different reflections of the light.

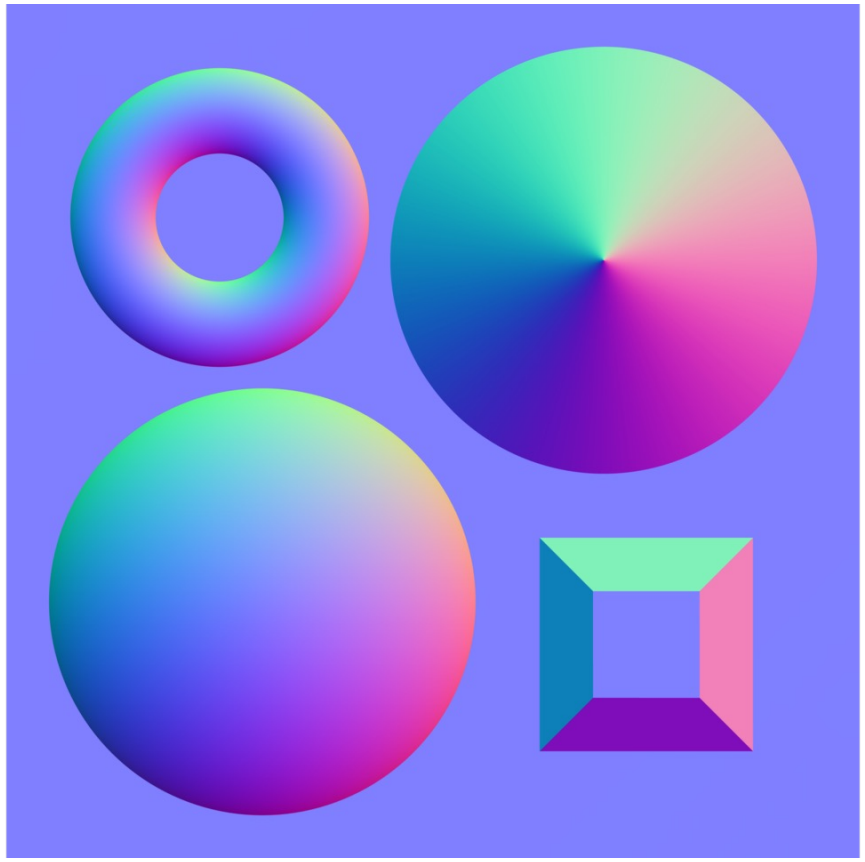


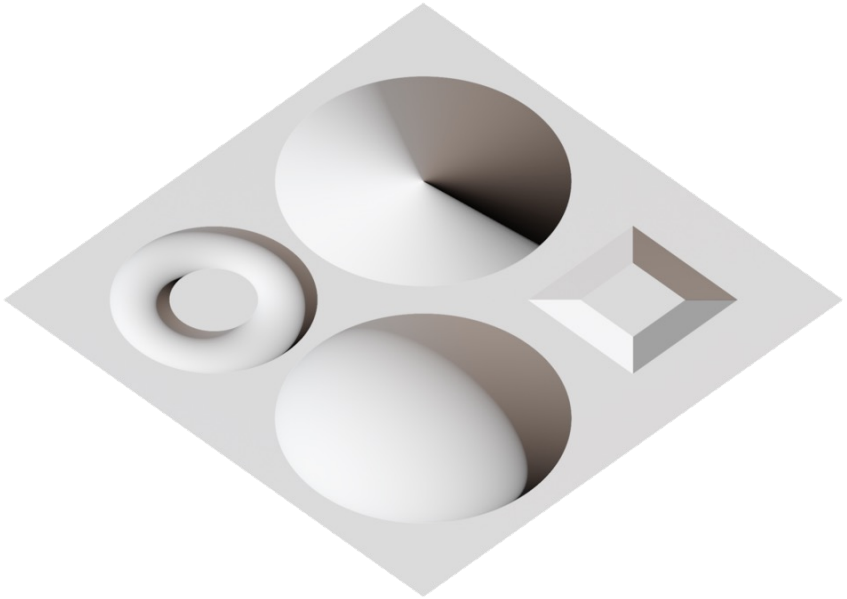




Normal Mapping

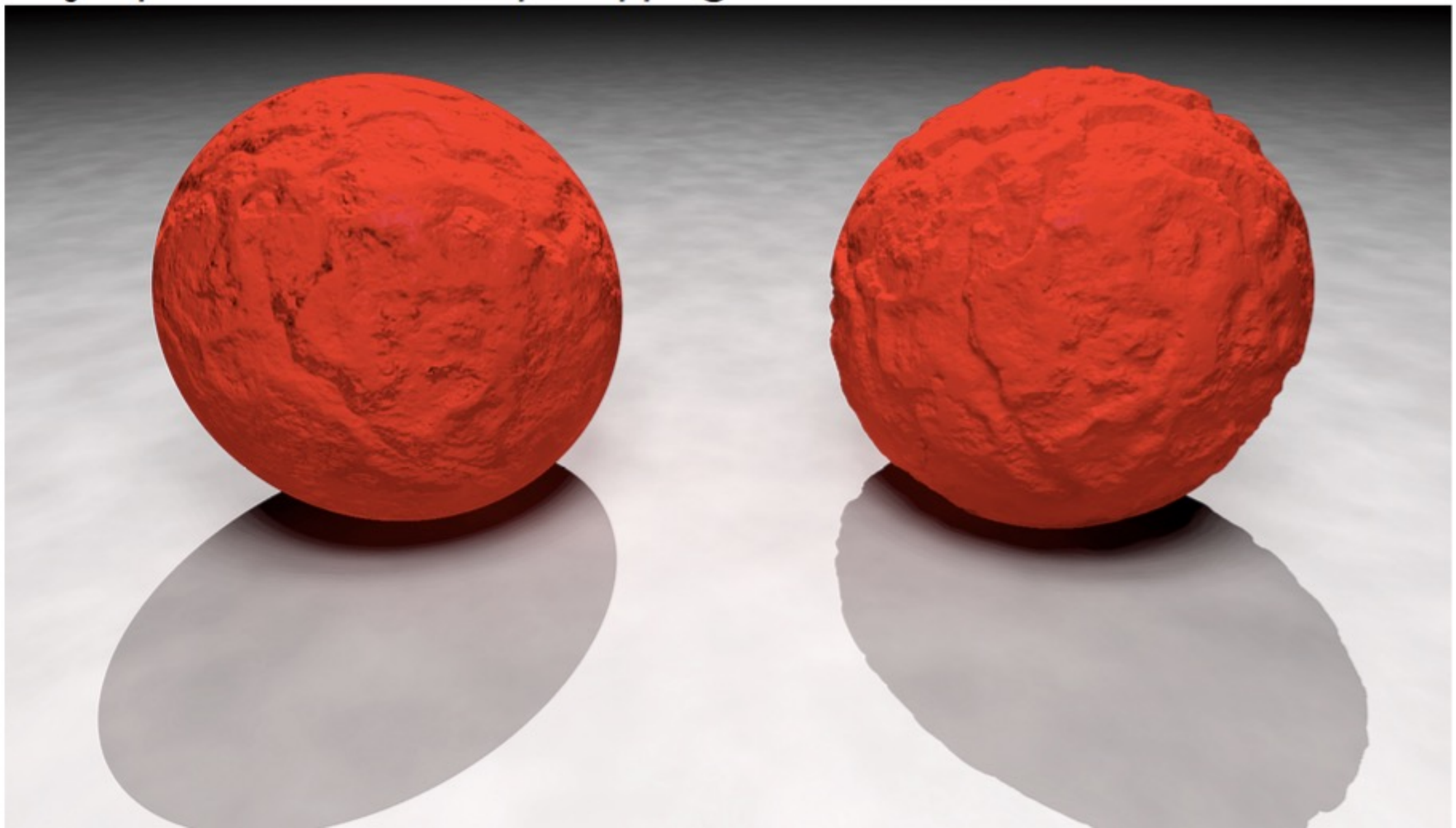




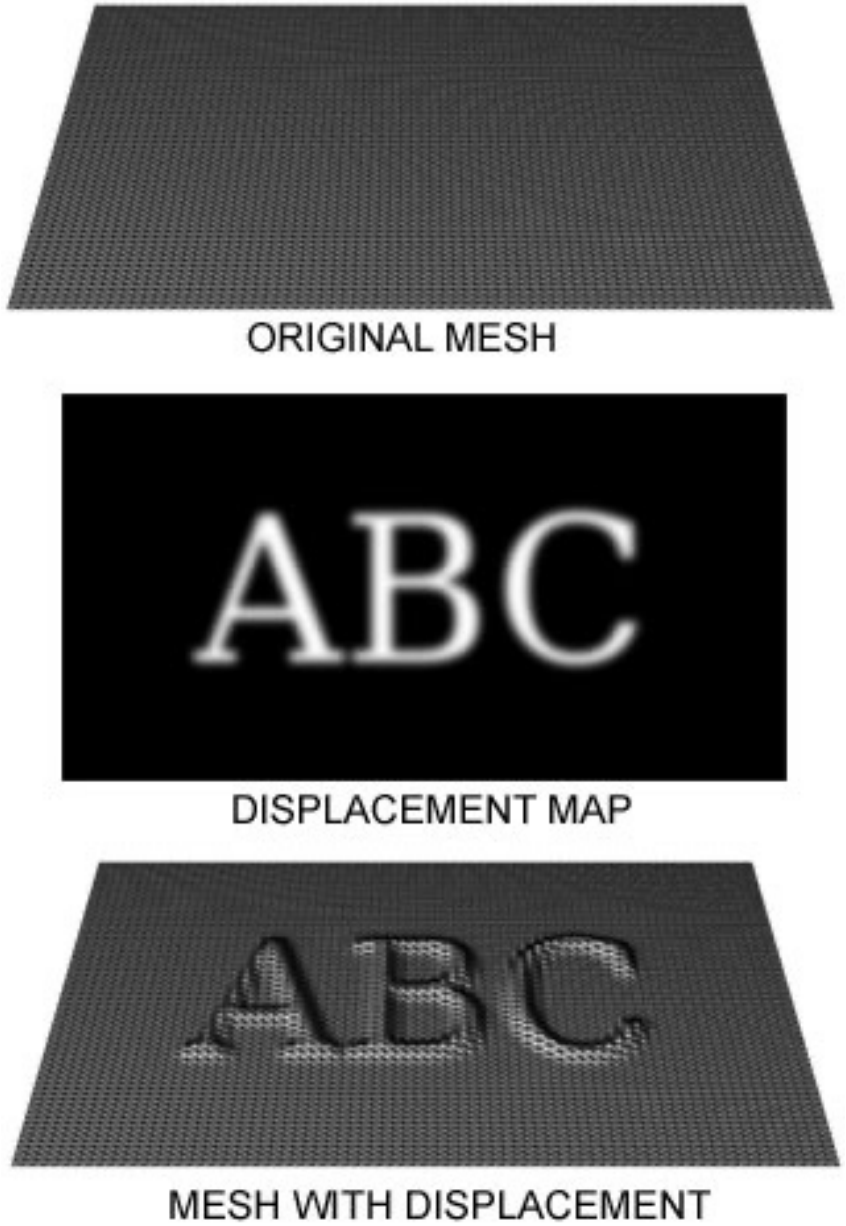


Normal Mapping vs. Geometry

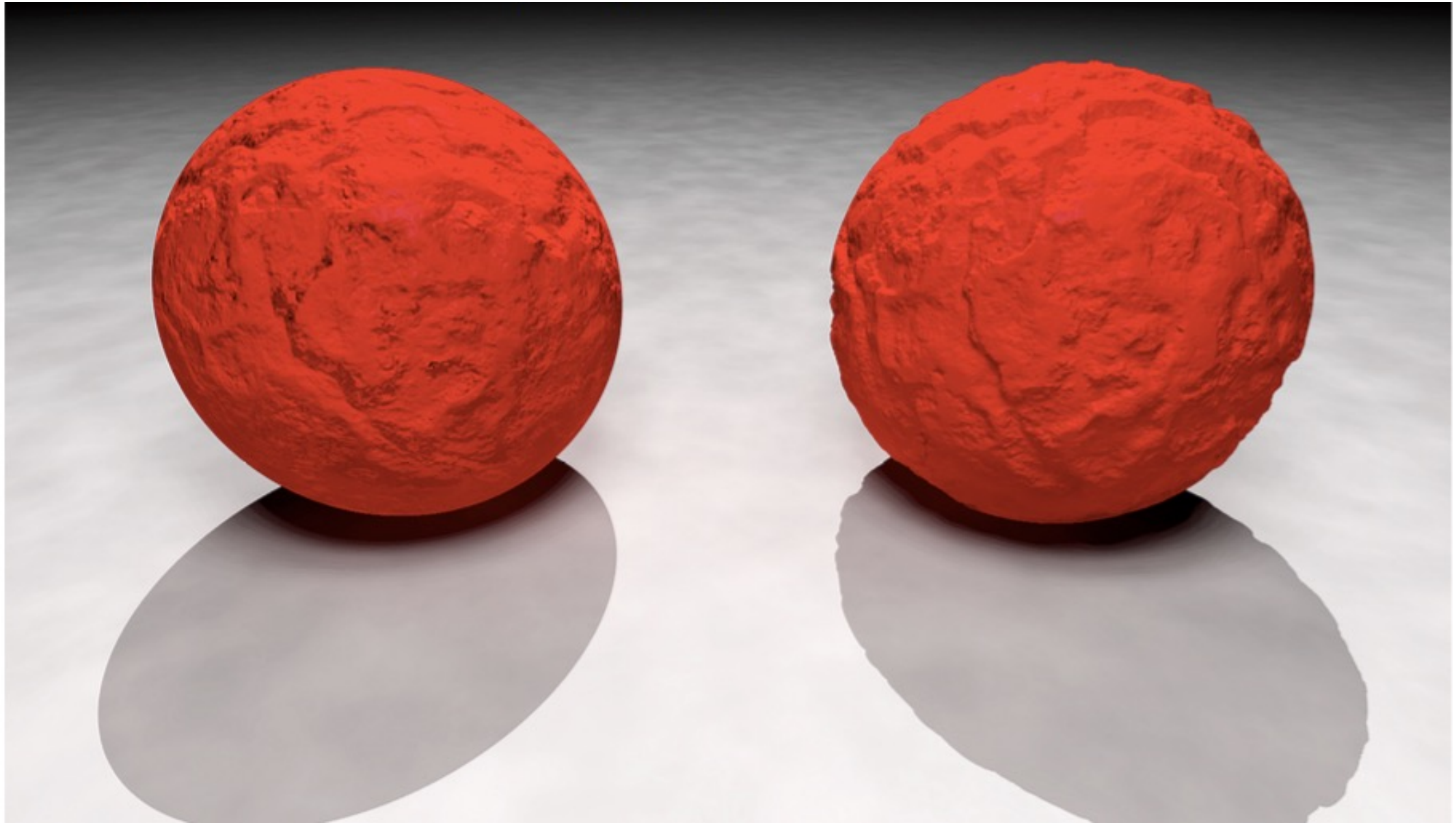
Major problems with bump mapping: silhouettes and shadows



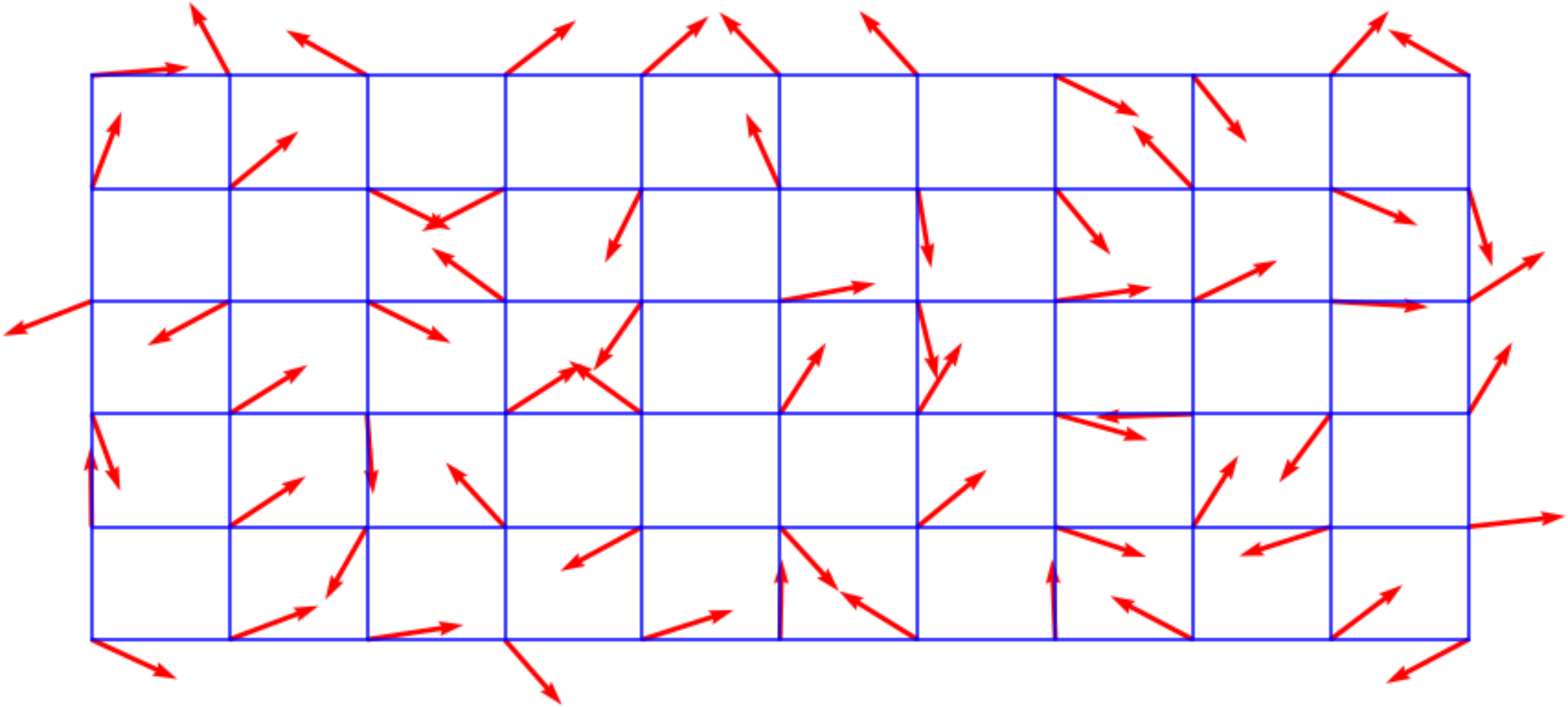
Displacement (Bump) Mapping



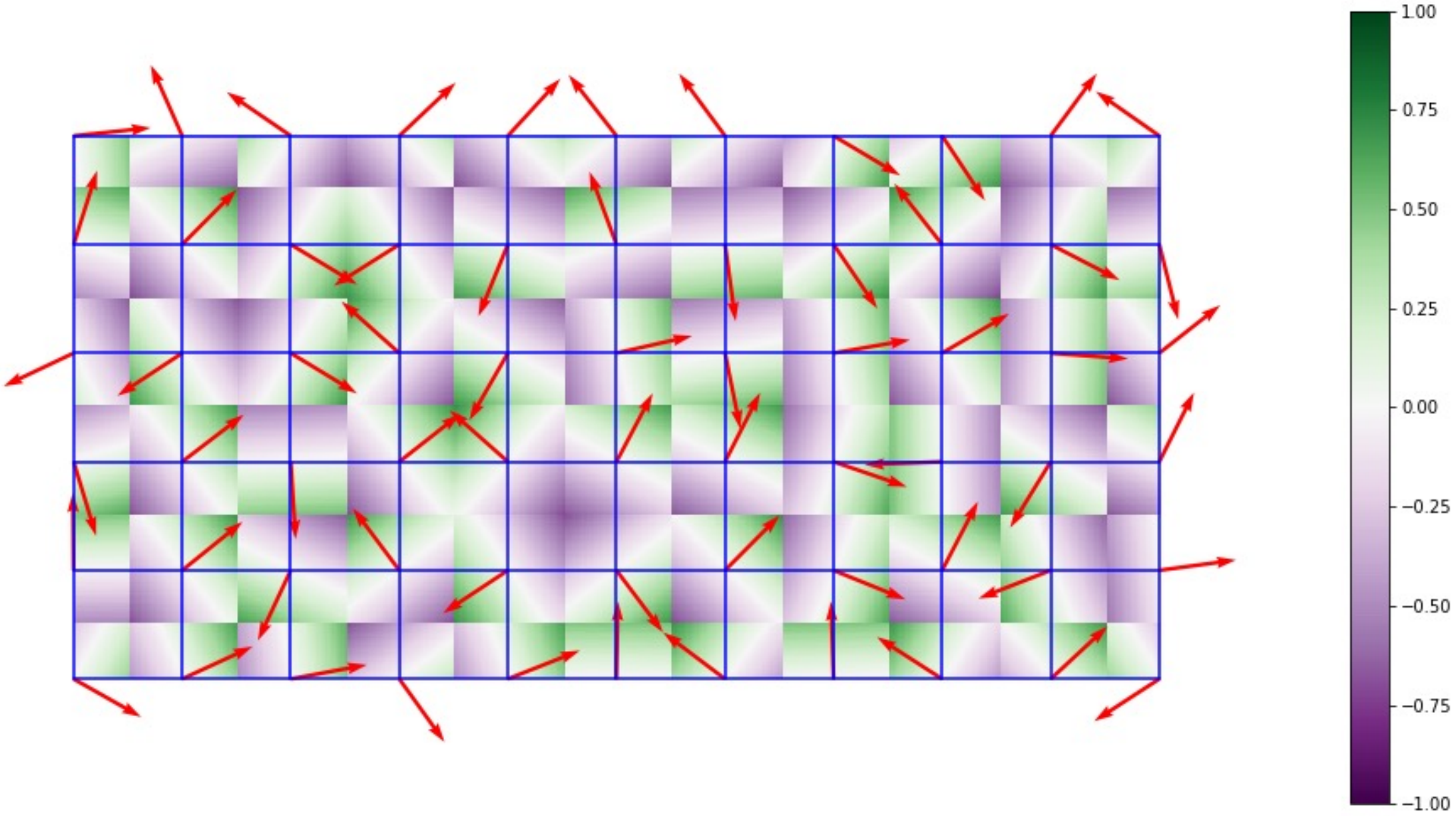
Normal Mapping vs. Displacement Mapping



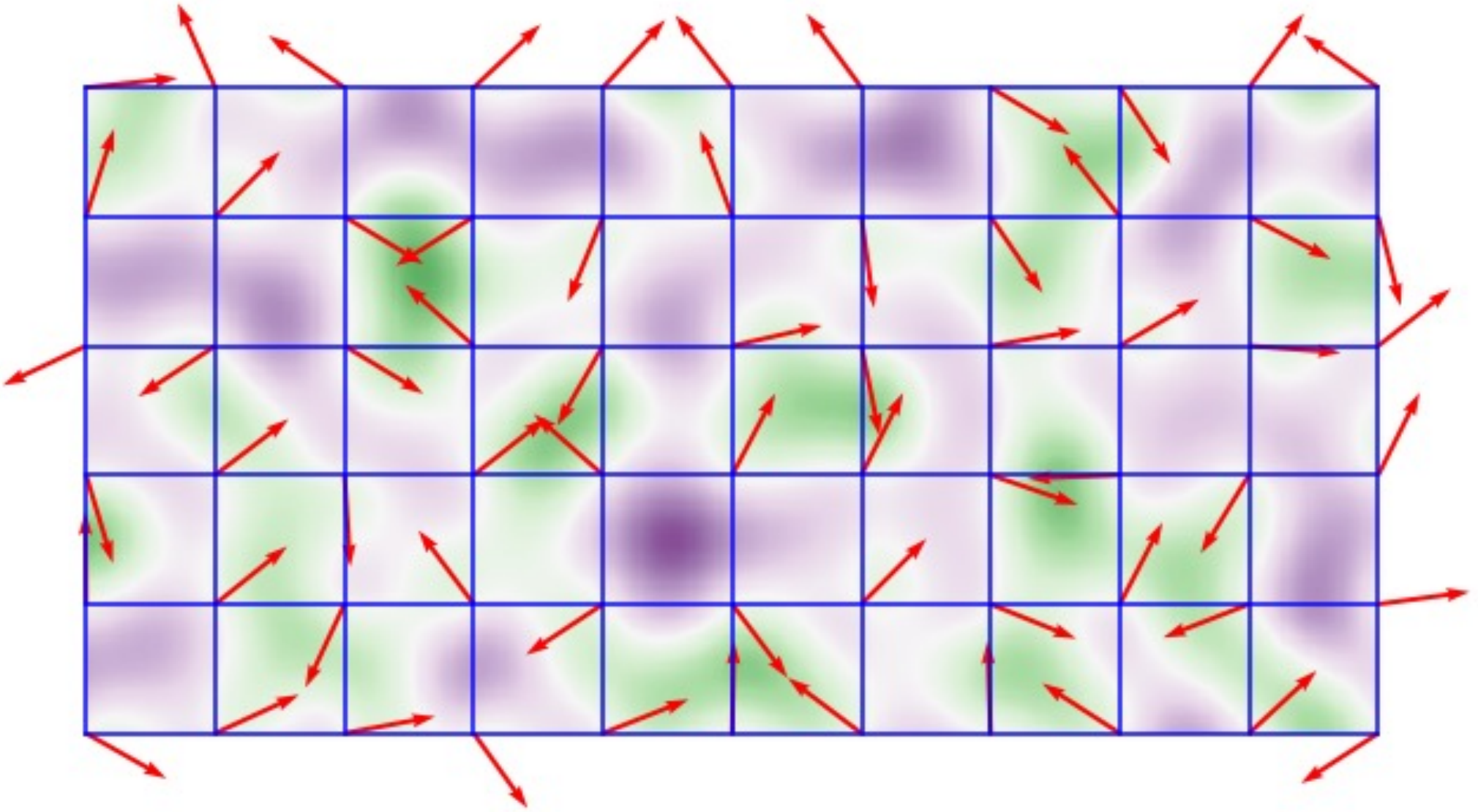
Perlin Noise



Improved Perlin Noise: https://mrl.cs.nyu.edu/~perlin/paper445.pdf



Improved Perlin Noise: https://mrl.cs.nyu.edu/~perlin/paper445.pdf



Improved Perlin Noise: https://mrl.cs.nyu.edu/~perlin/paper445.pdf

Perlin Noise

Announcements

Assignment 6 out now, due November 1st