

CSC317 Computer Graphics

... starts at 1:15pm

Rob Kaz

Some Slides/Images adapted from Marschner and Shirley

Today: Transformations and Shaders



Transforms and Shaders

Reminder – Rasterization

Introduction to the Graphics Pipeline

Transformations

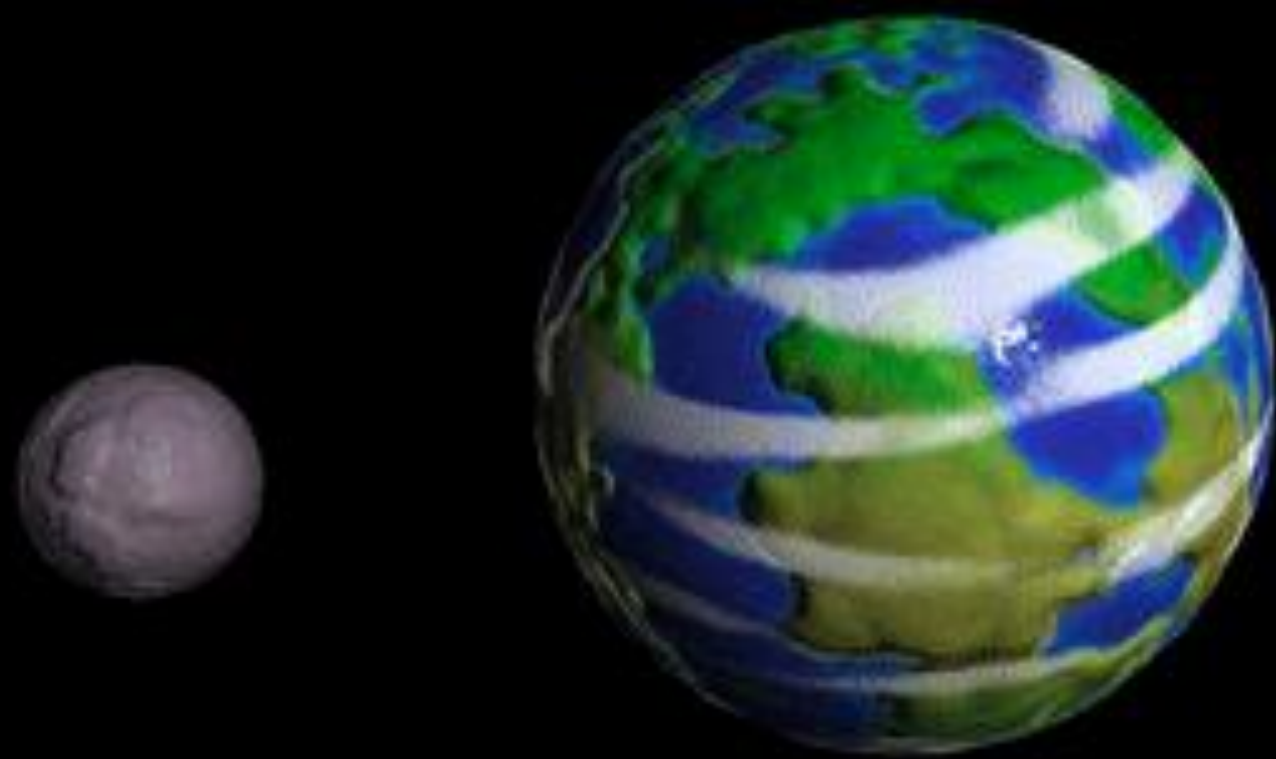
Normal and Bump Mapping

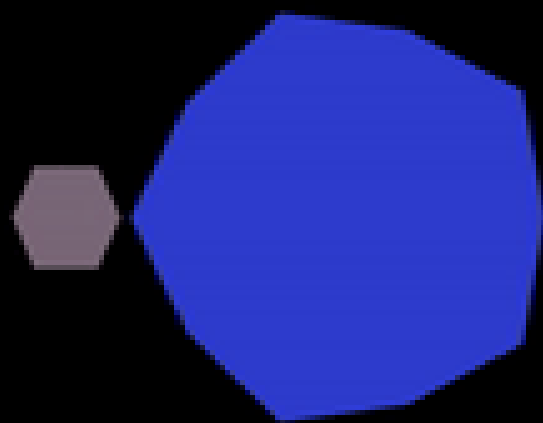
Perlin Noise

Announcements

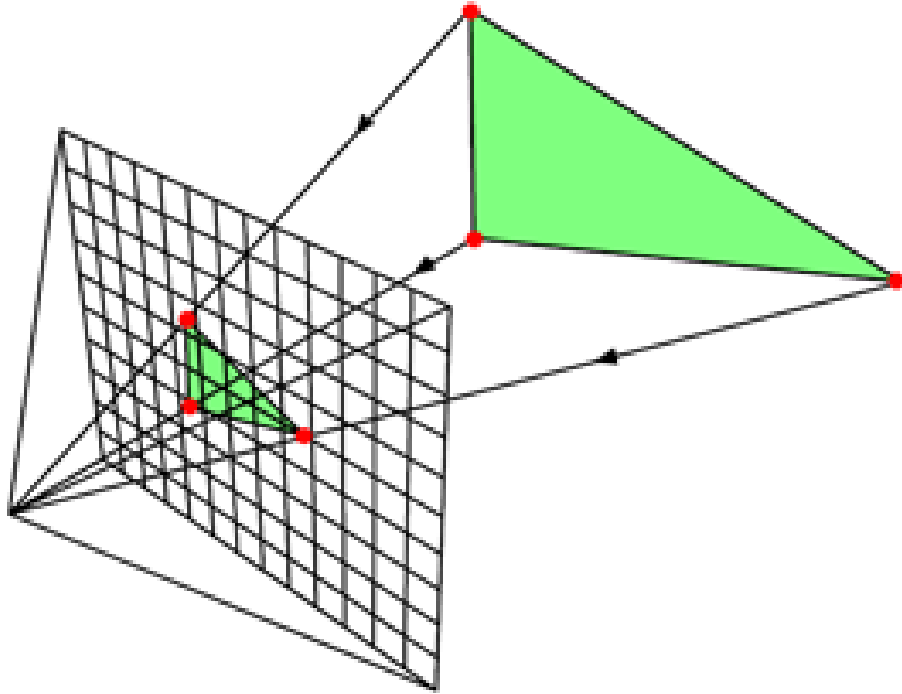
Assignment 6 out now, due November 5th

Any Questions ?



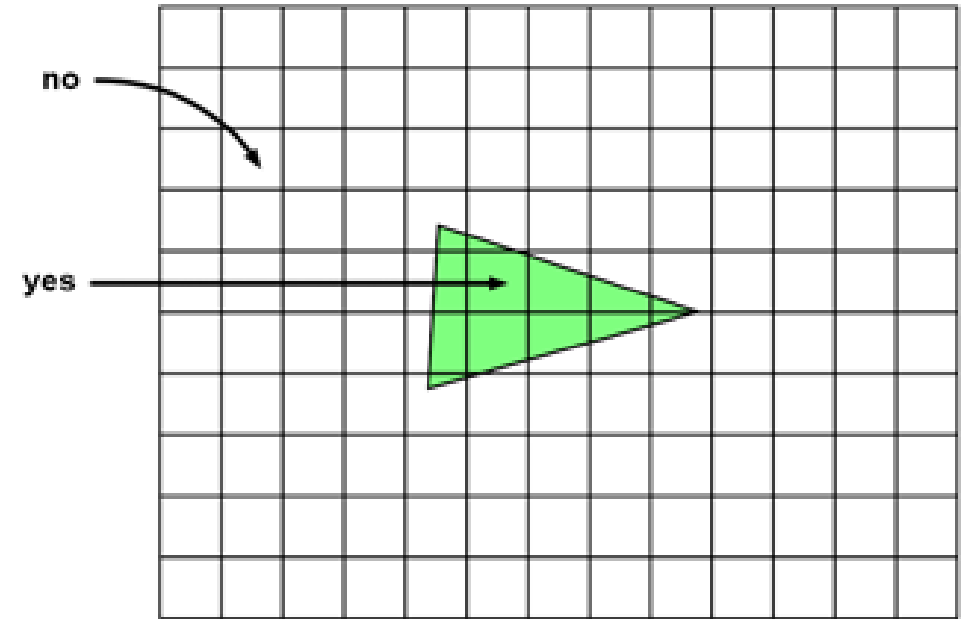


Rasterization



© www.scratchapixel.com

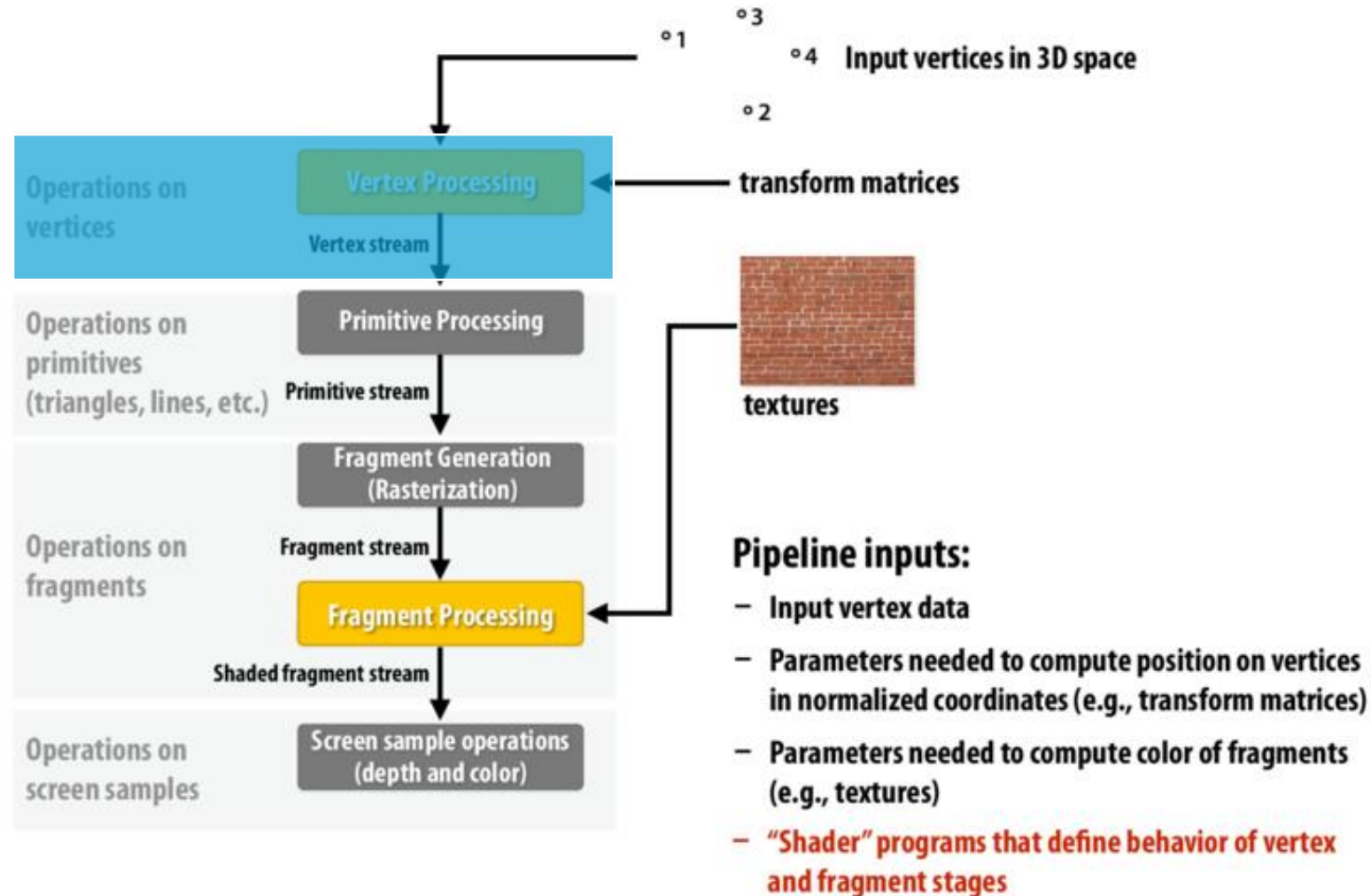
1. Project Vertices to Image Plane



2. Turn on pixels inside triangle

Modern Graphics Pipeline

OpenGL/Direct3D graphics pipeline *



* 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

$$Ax = b$$

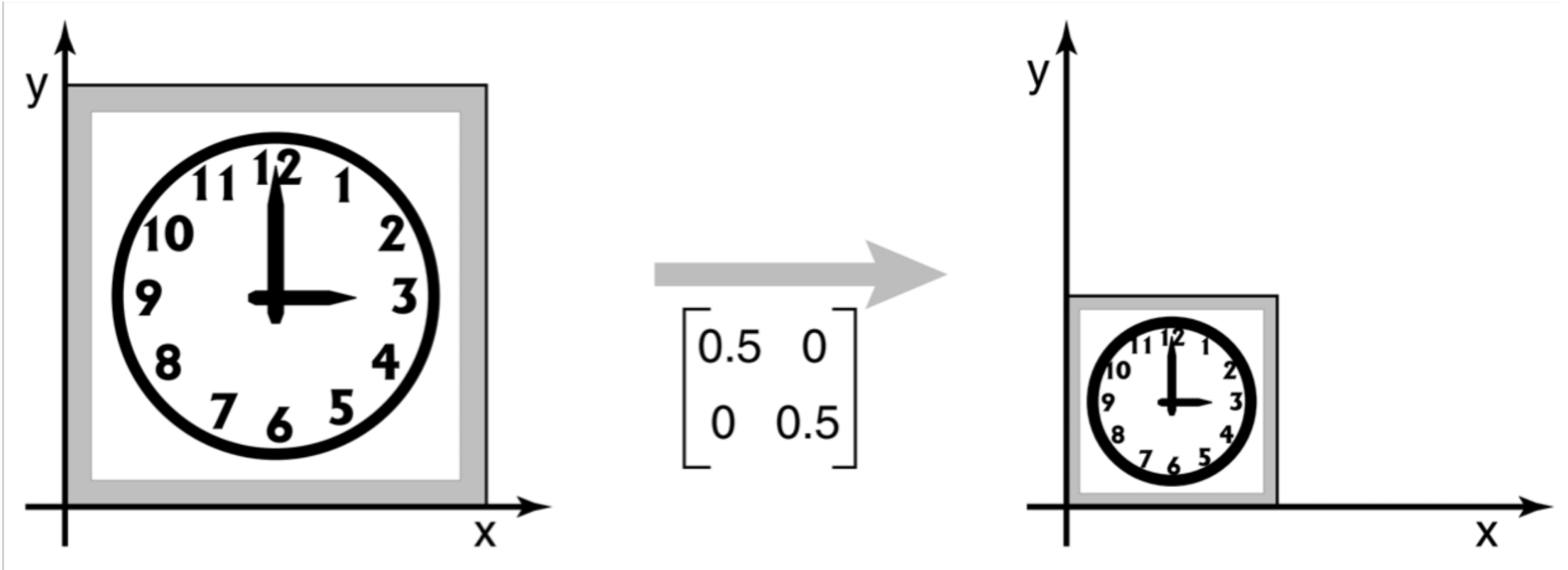
2D Linear Transformations

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

2D Linear Transformations - Scale

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

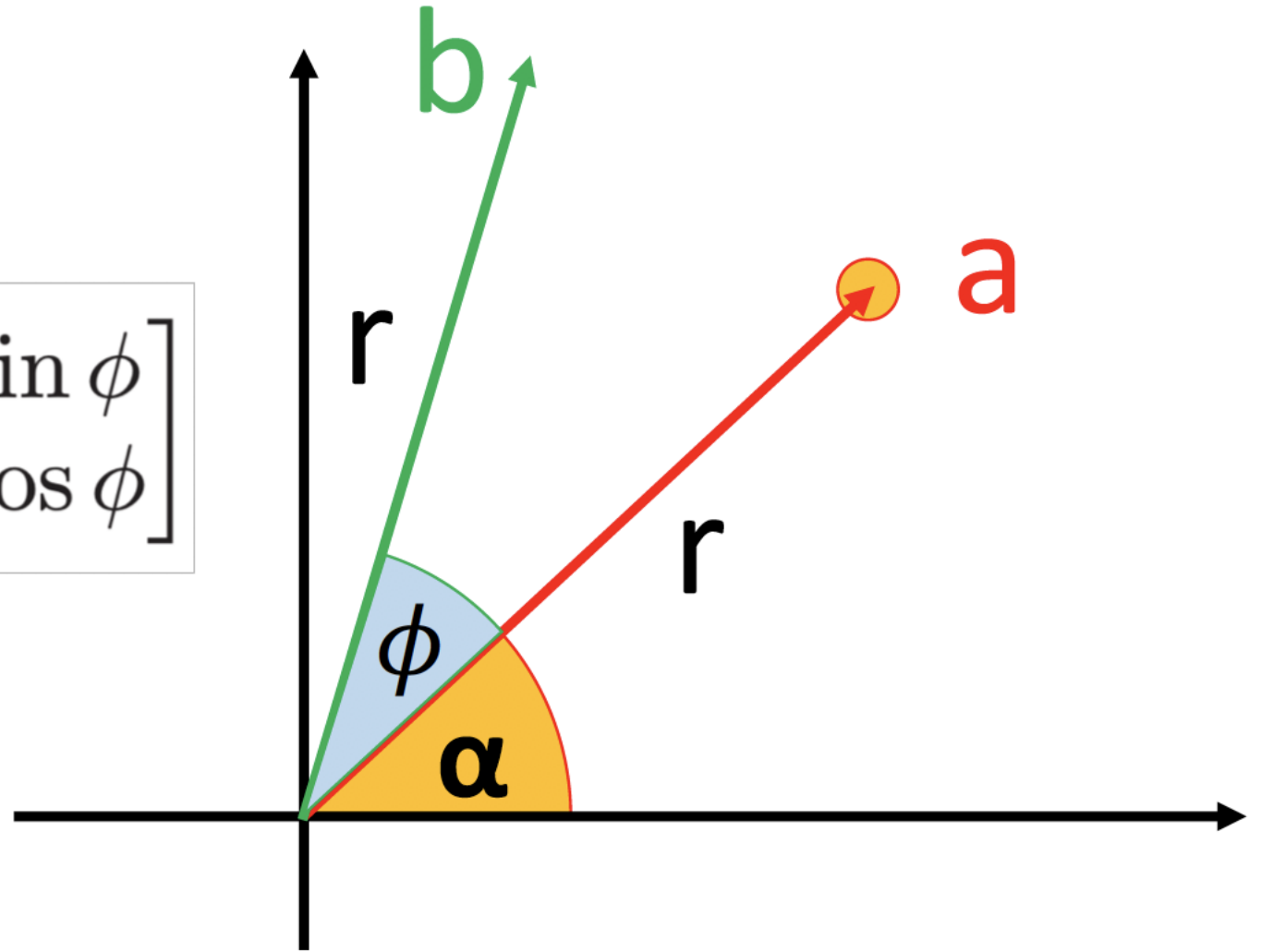
Linear transformations in 2D: Scale



When $S_x = S_y$ we say the scaling is uniform

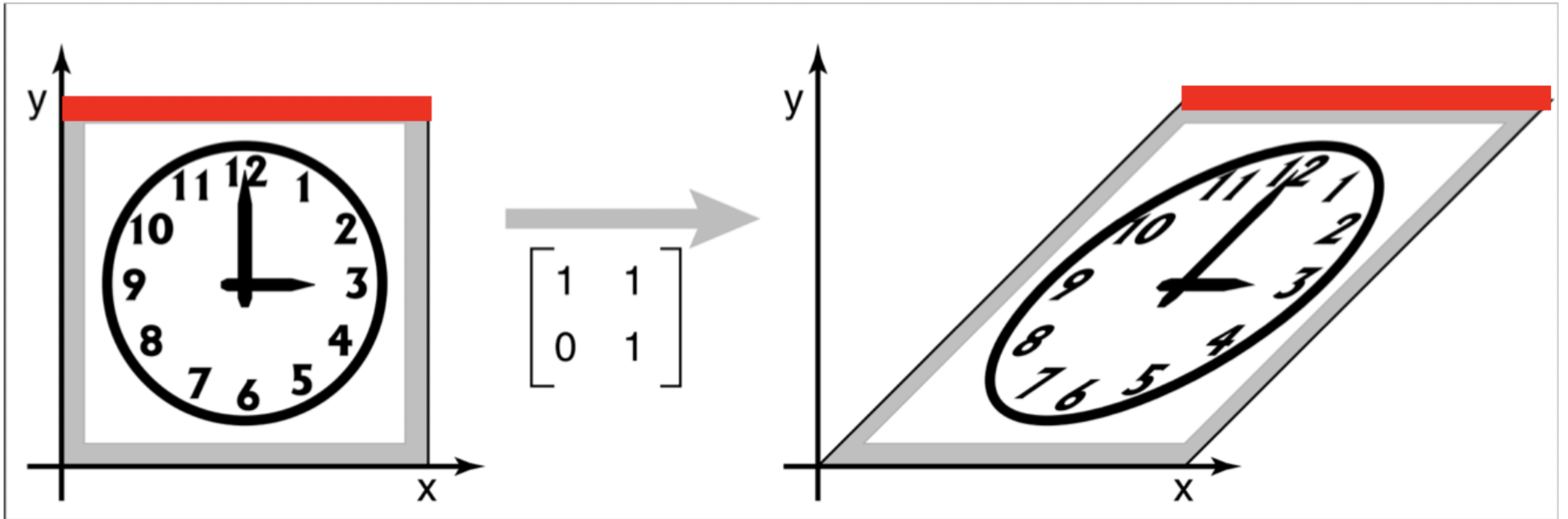
2D Linear Transformations - Rotation

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



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

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

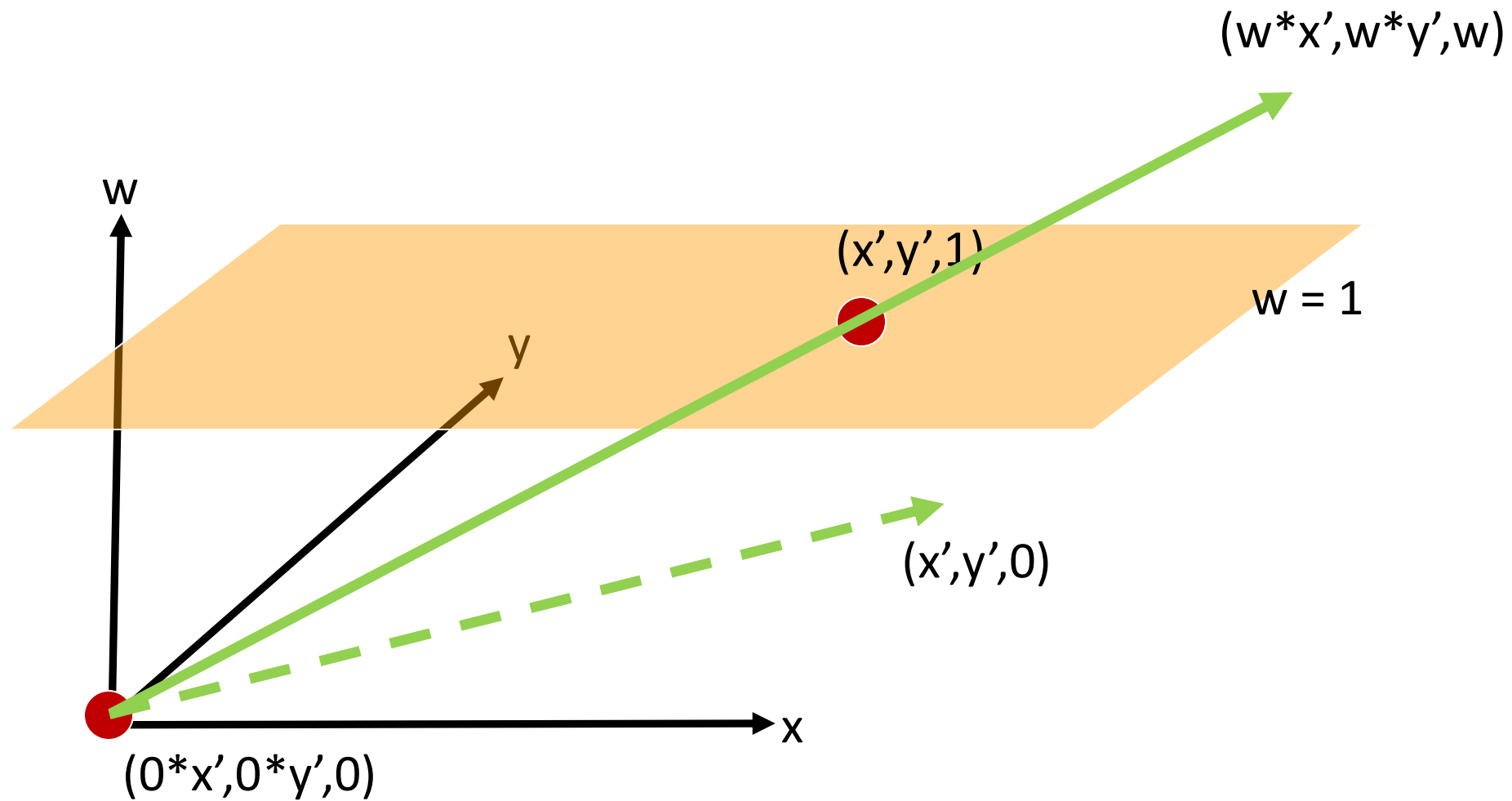
$$Ax + t = b$$

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

Considered as a point in 3D
homogeneous coordinates

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

Geometric Intuition



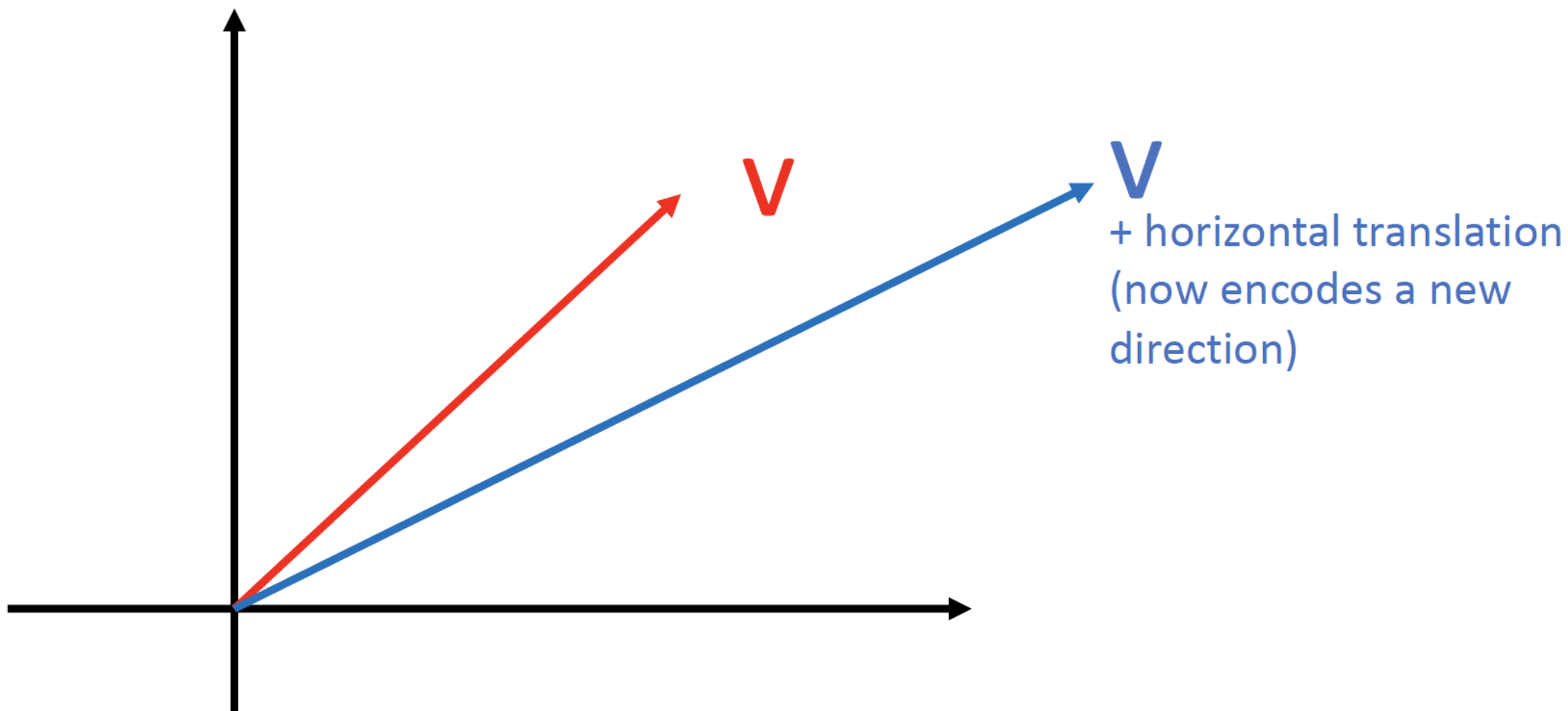
What about vectors?

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

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



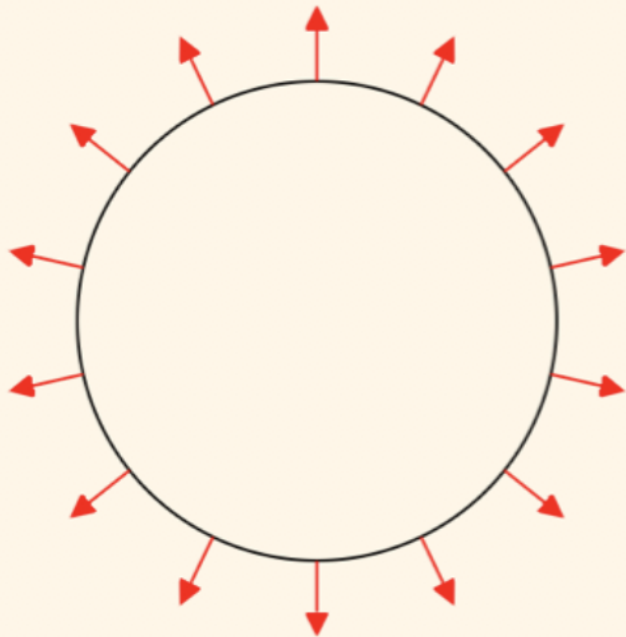
3D Linear Transformations

$$Ax = b$$

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

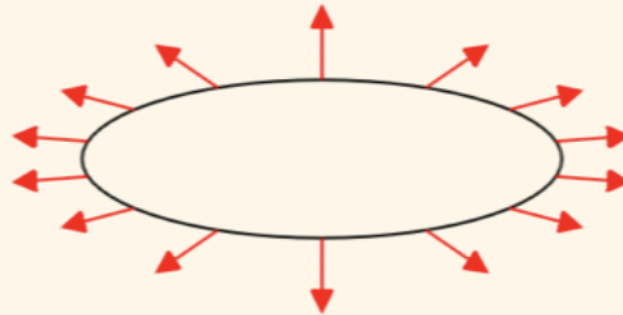
No, thank you for asking.

Original object



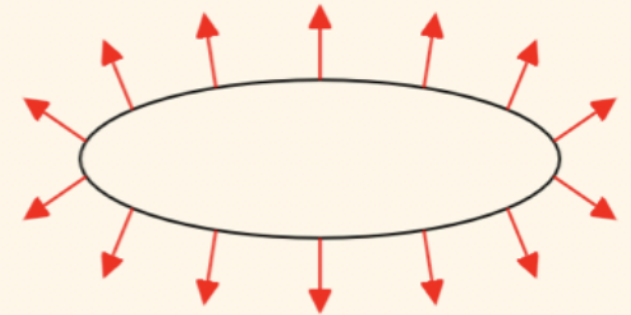
Transformed

With normals using the same transformation matrix



Transformed

With correct transformation applied to normals



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

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

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

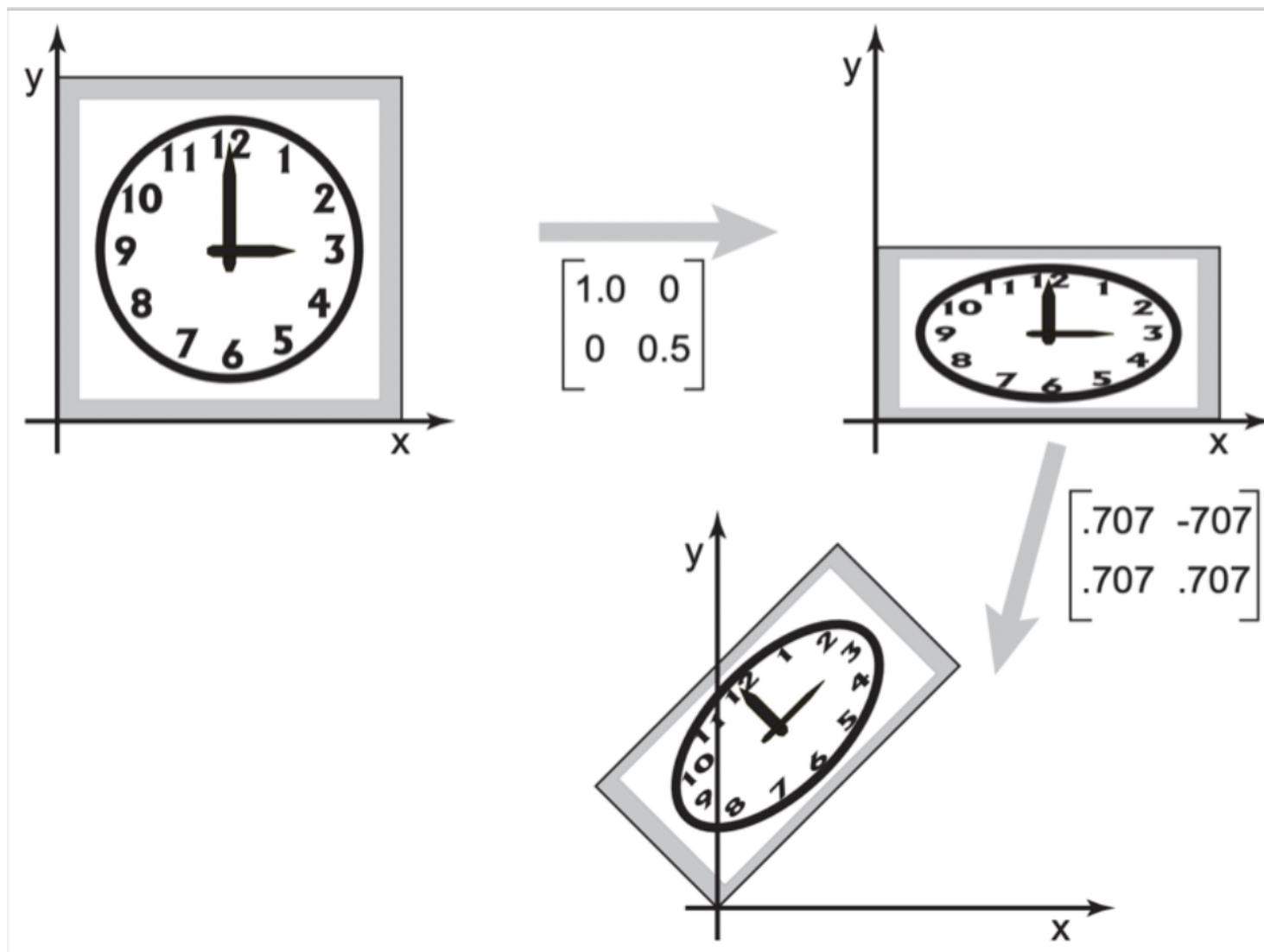
What's the right way to transform a normal vector?

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

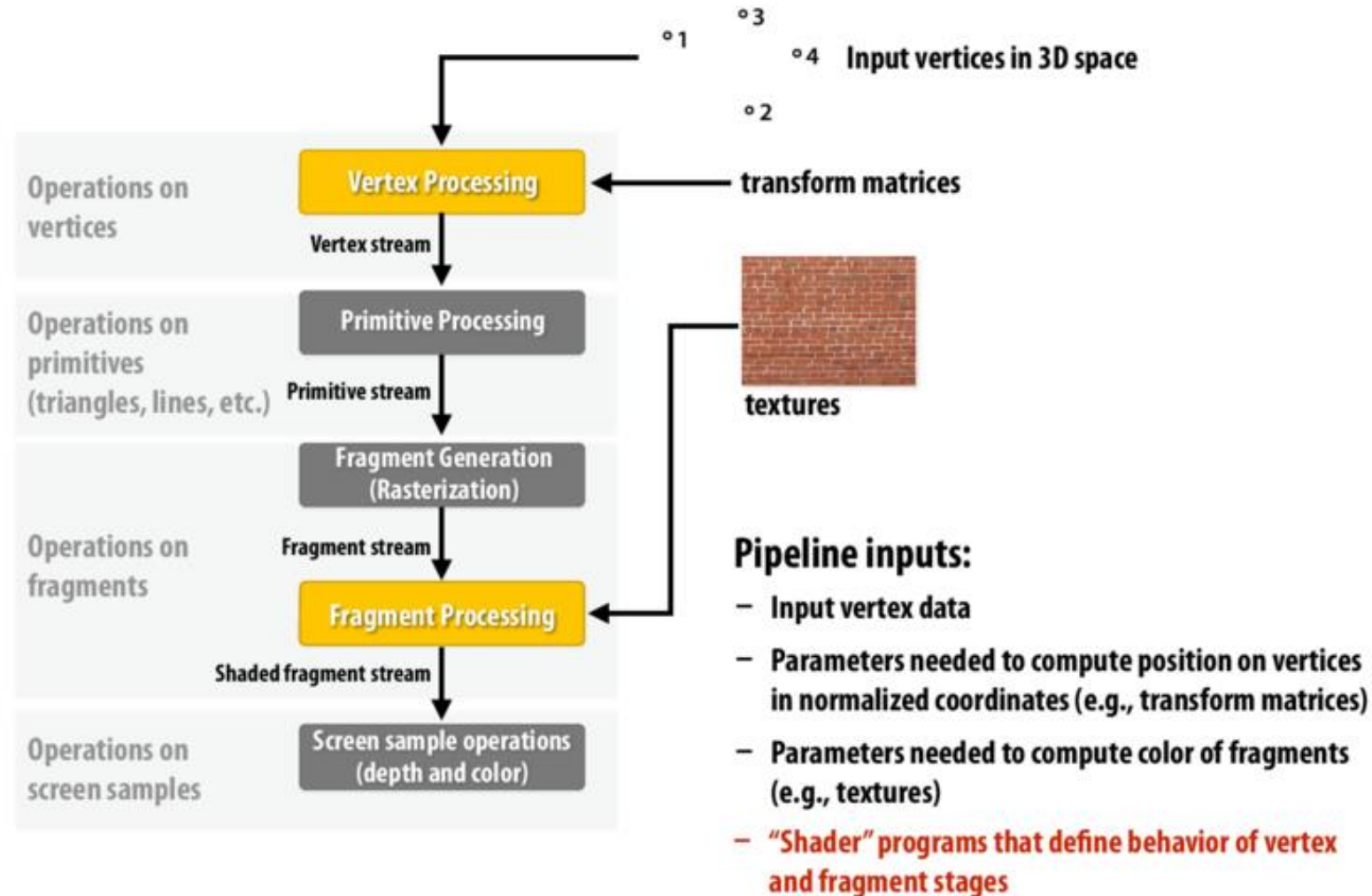


Composing transformations

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

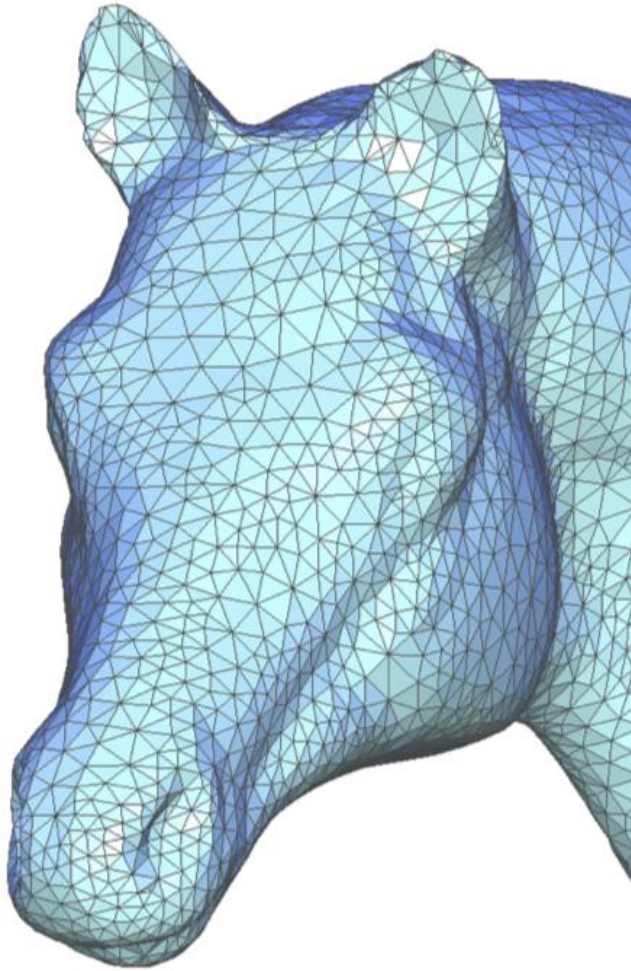
Modern Graphics Pipeline

OpenGL/Direct3D graphics pipeline *

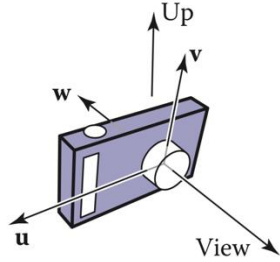


* several stages of the modern OpenGL pipeline are omitted

Getting Things Onto The Screen



Object Space

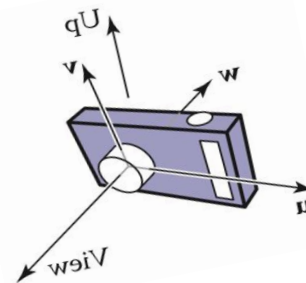
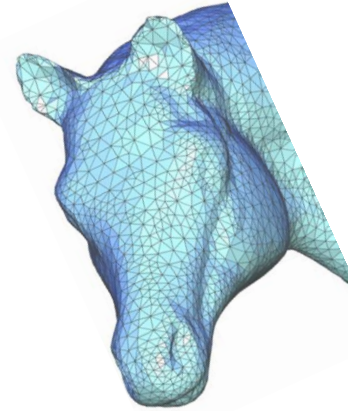


Open GL Combines these
into the ModelView Matrix

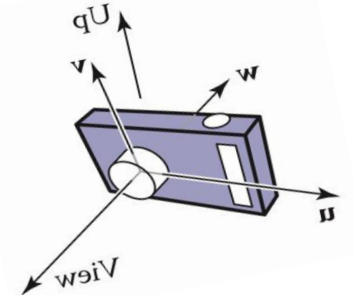
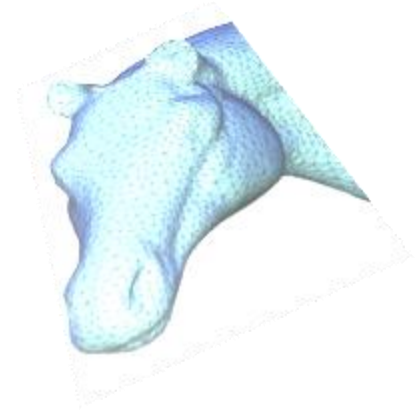
M

V

World Space



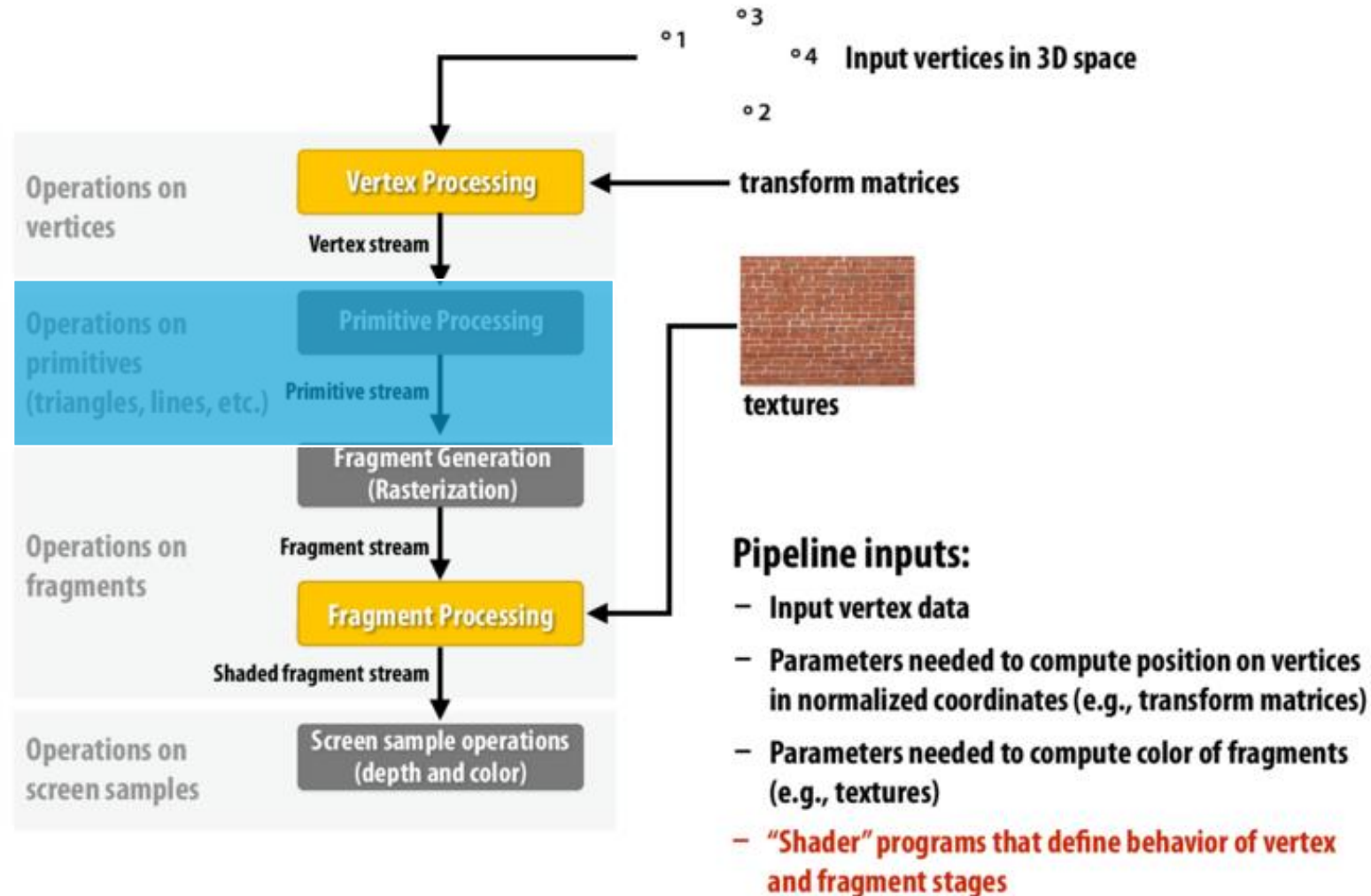
Camera Space



Canonical
Space

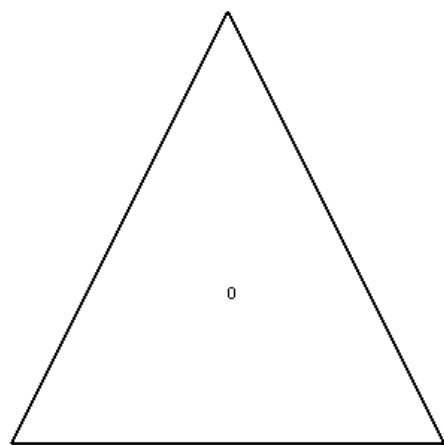
Modern Graphics Pipeline

OpenGL/Direct3D graphics pipeline *

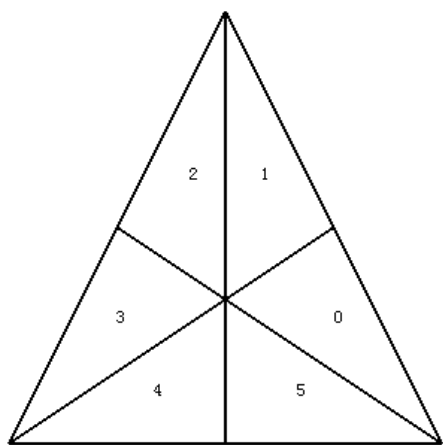


* several stages of the modern OpenGL pipeline are omitted

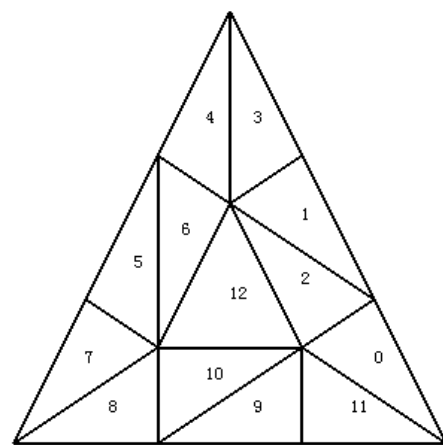
Tessellation Shader



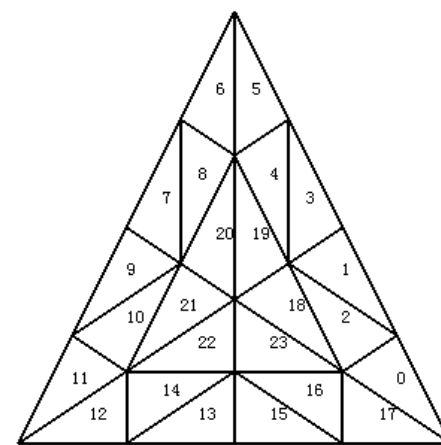
Level 1



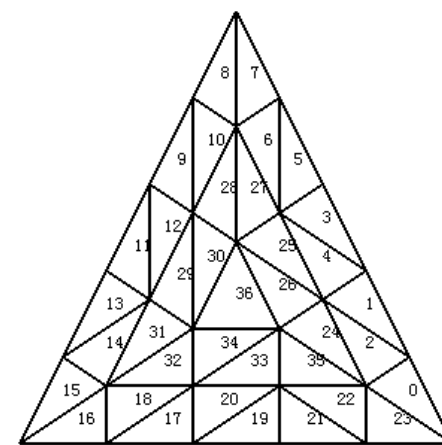
Level 2



Level 3



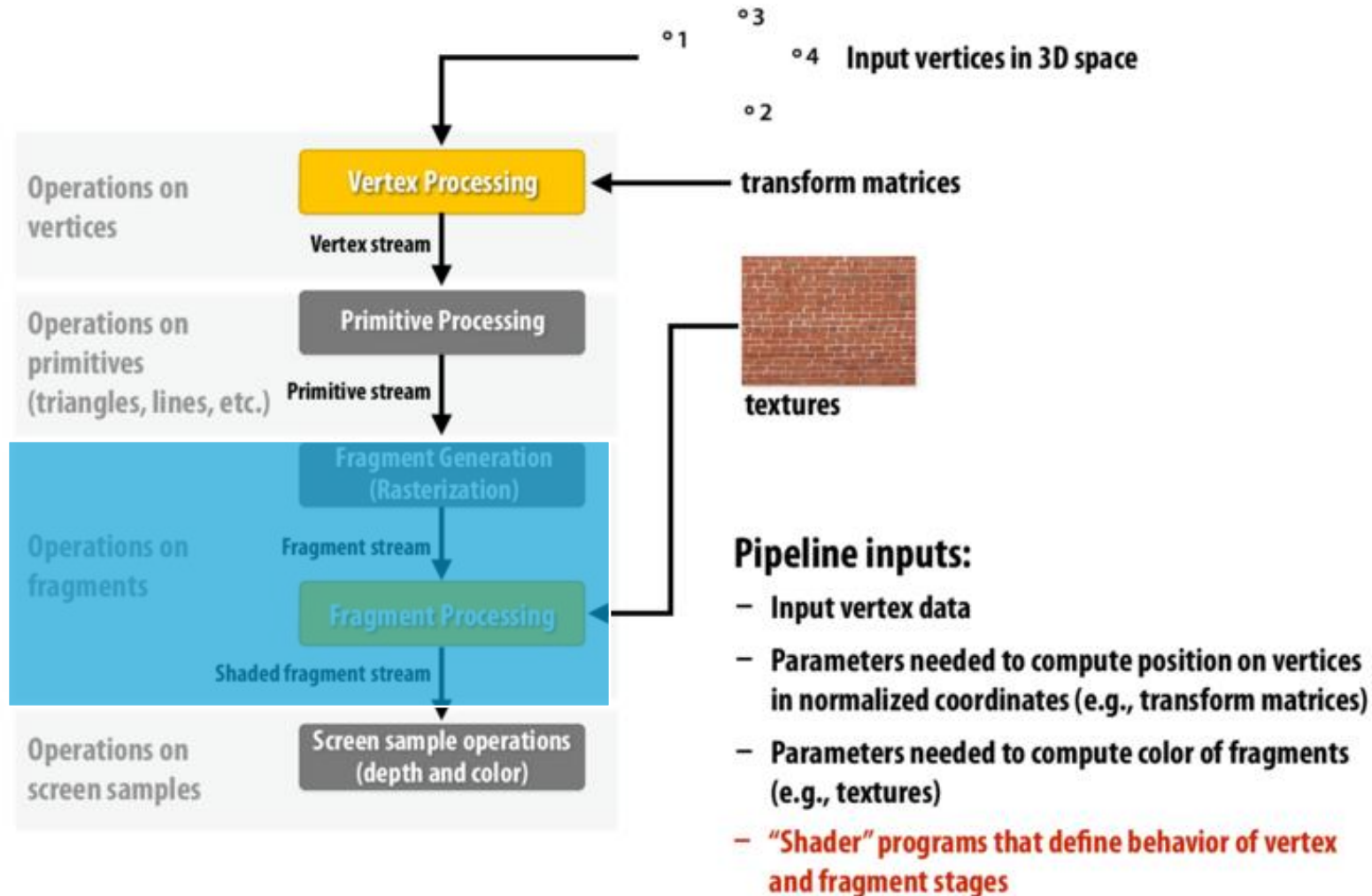
Level 4



Level 5

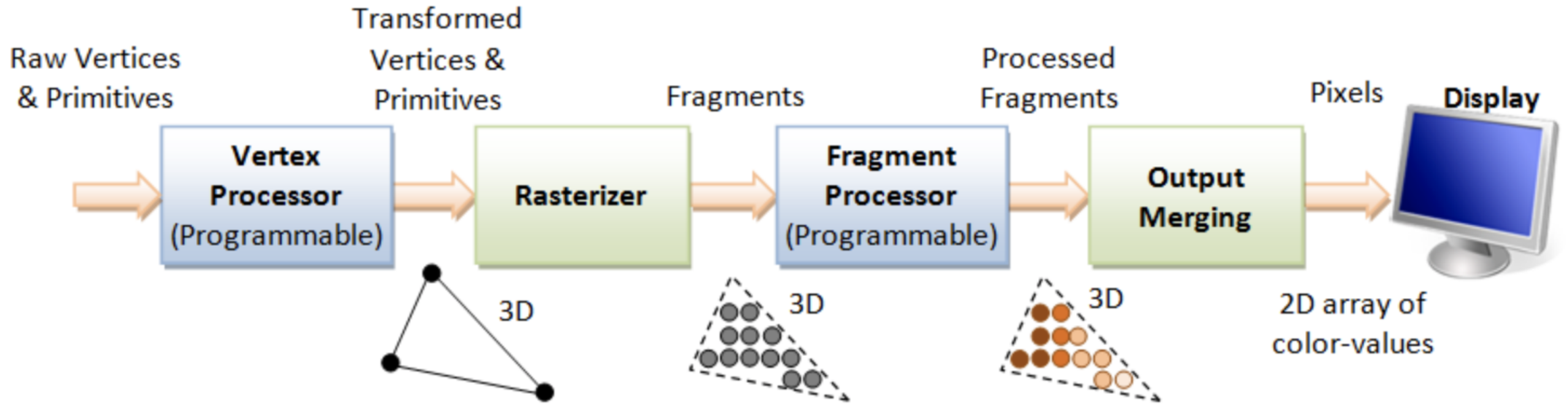
Modern Graphics Pipeline

OpenGL/Direct3D graphics pipeline *

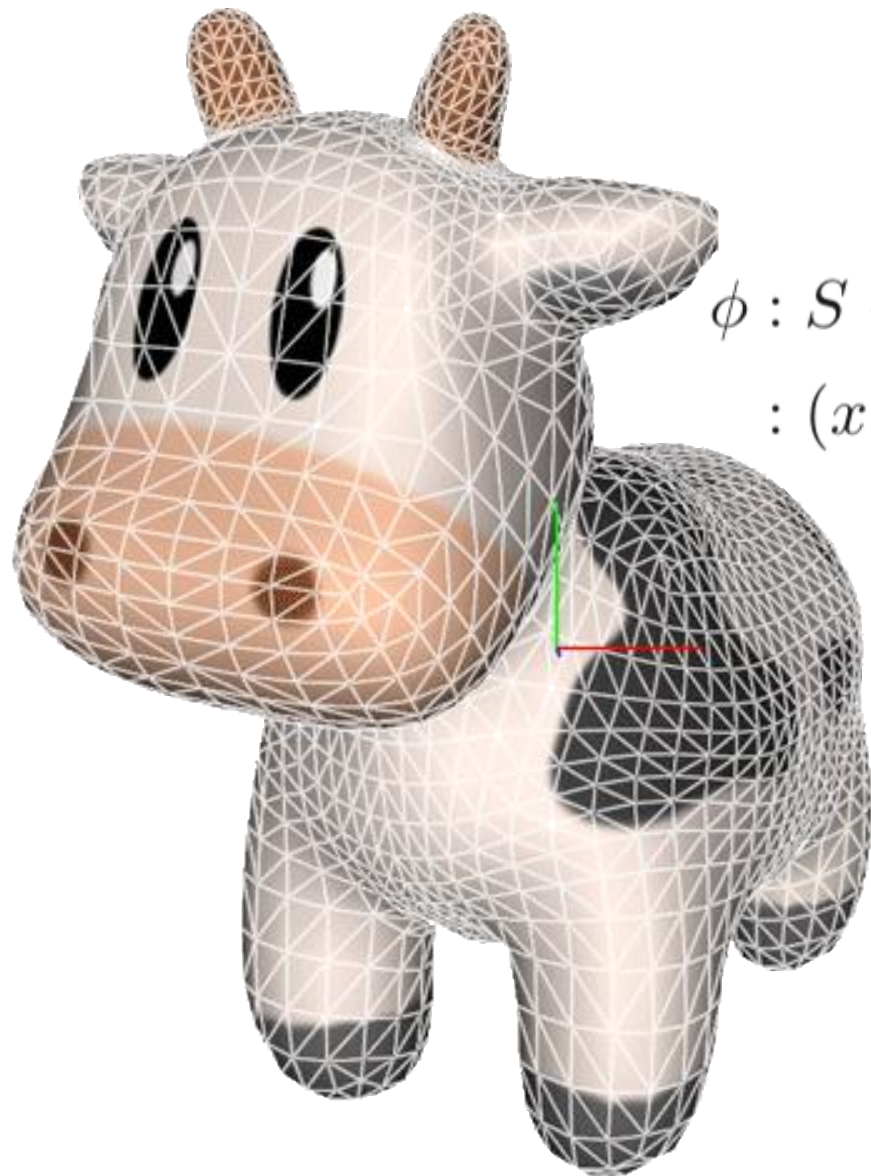


* several stages of the modern OpenGL pipeline are omitted

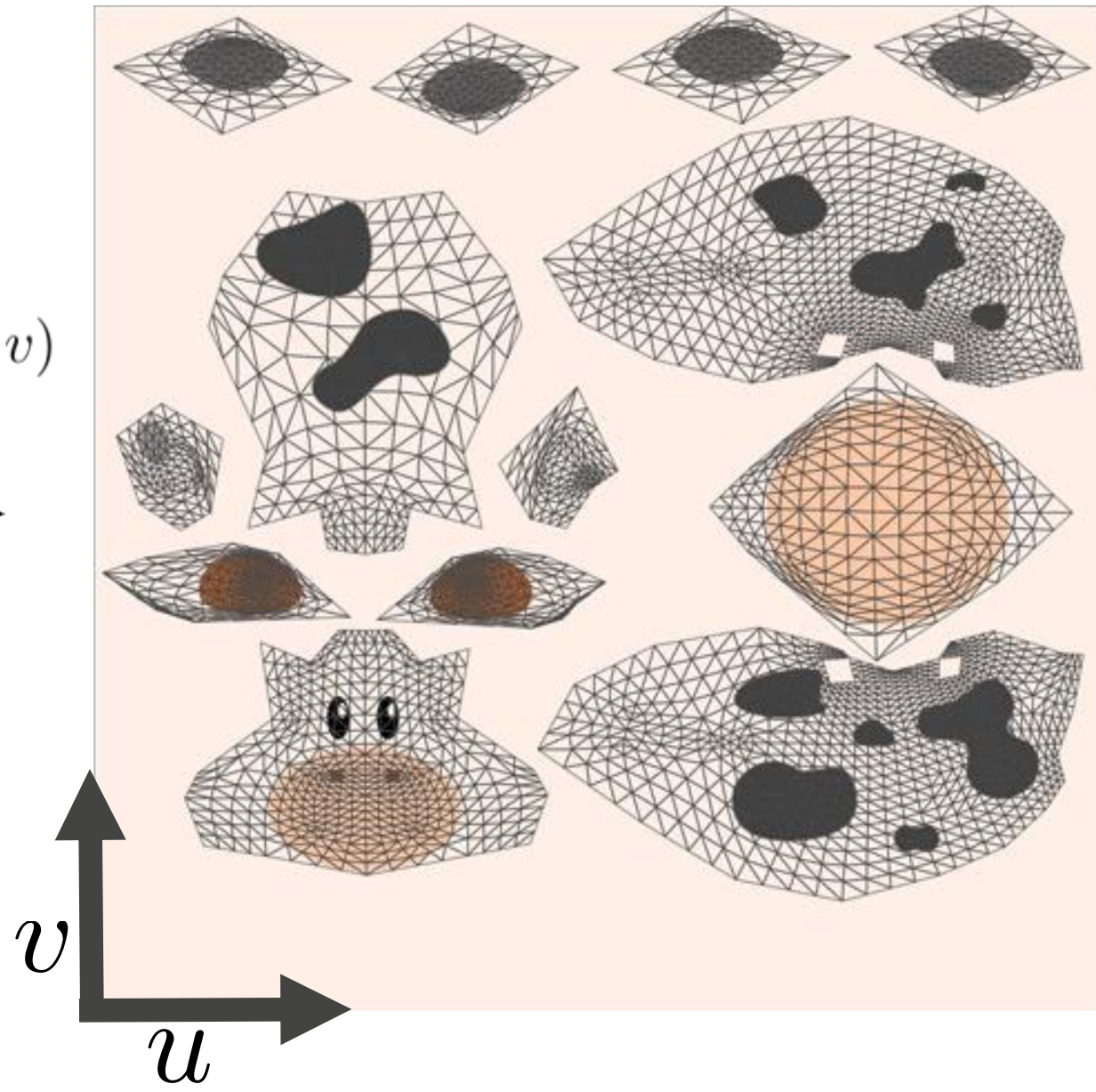
Fragment Shader



Texture coordinates



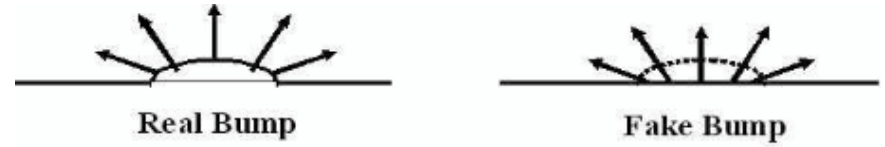
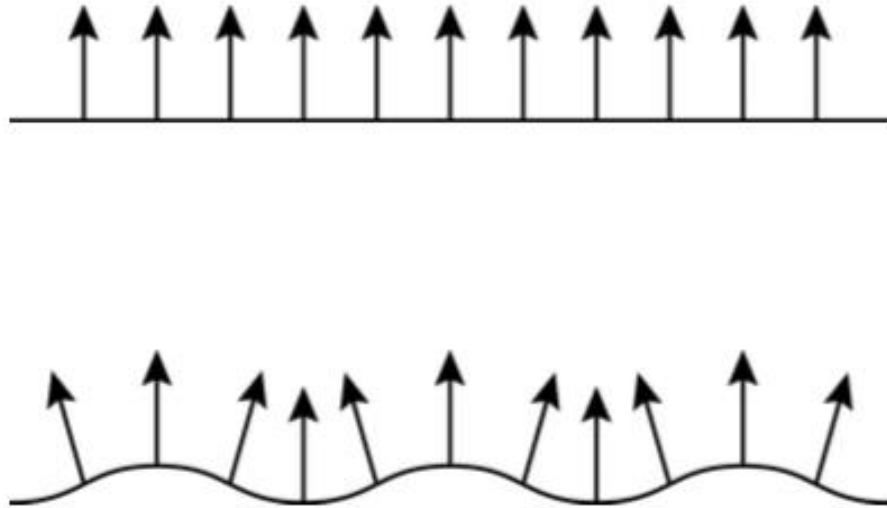
$$\begin{aligned}\phi : S &\rightarrow T \\ &: (x, y, z) \mapsto (u, v)\end{aligned}$$



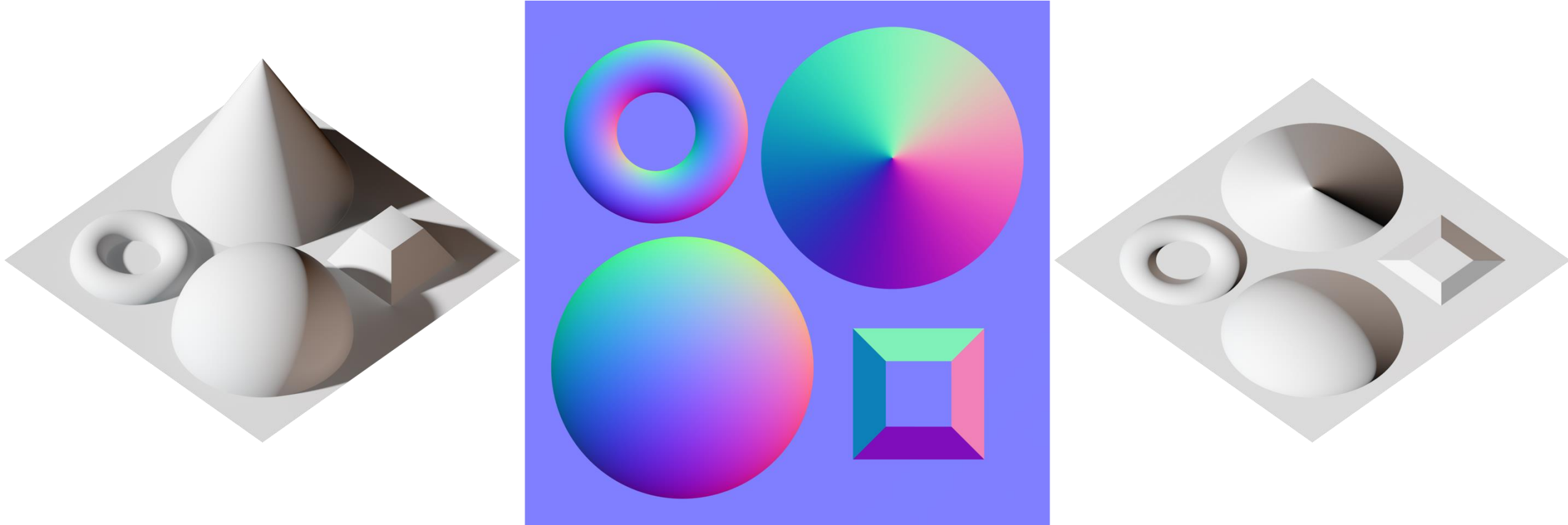
Normal Mapping

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.

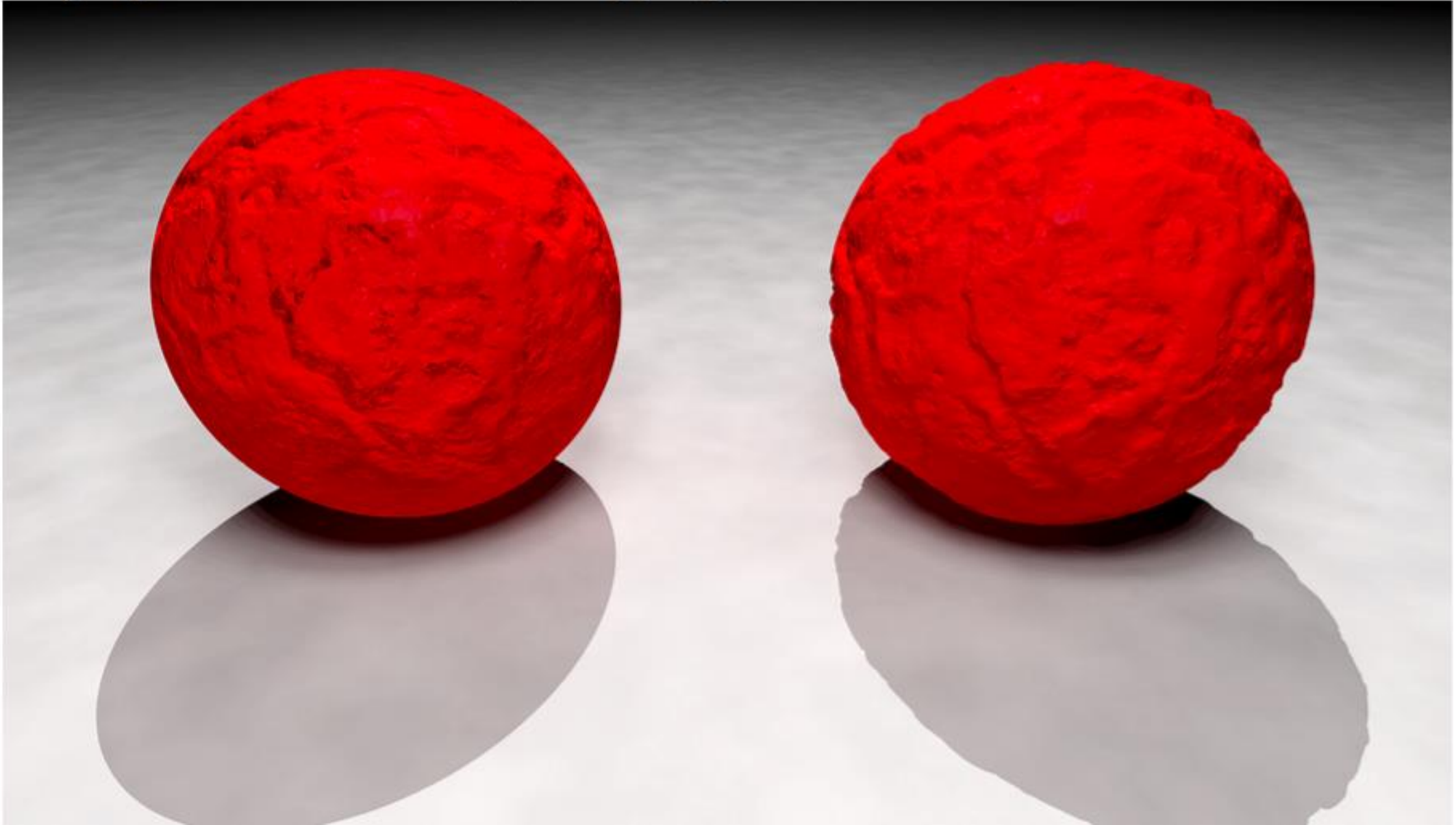


Normal Mapping

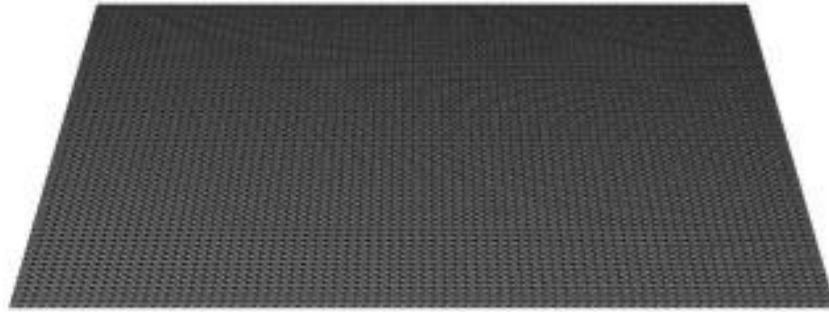


Normal Mapping vs. Geometry

Major problems with bump mapping: silhouettes and shadows



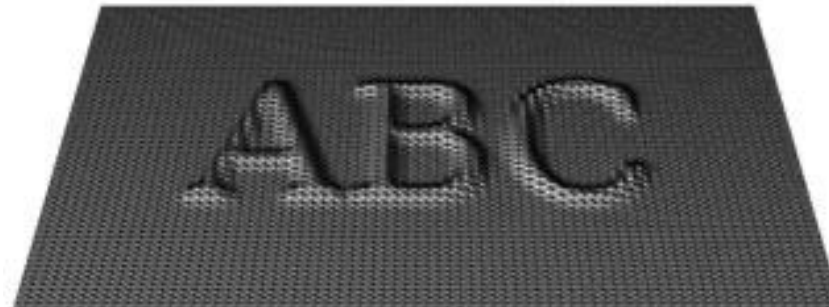
Displacement (Bump) Mapping



ORIGINAL MESH

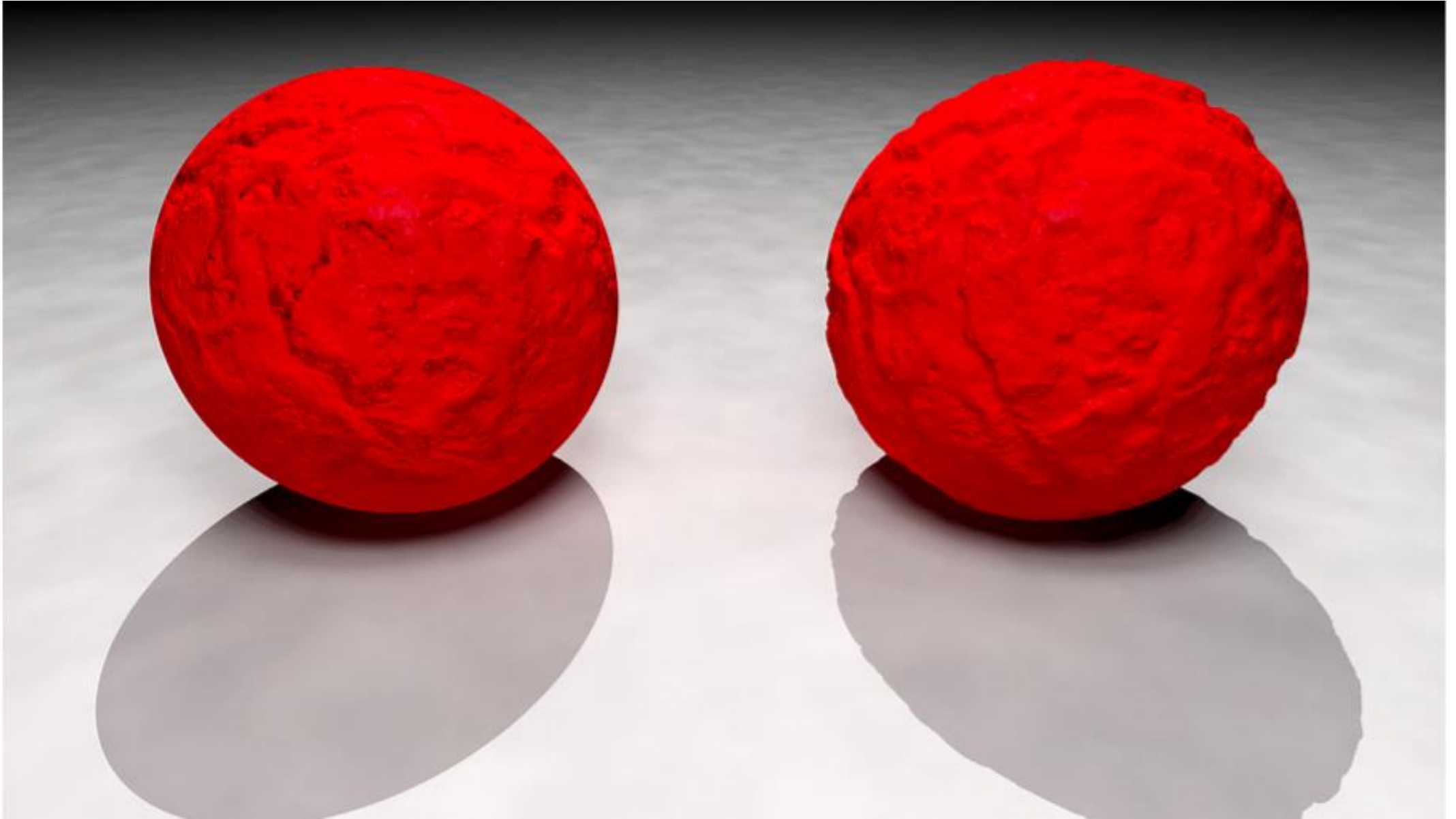


DISPLACEMENT MAP



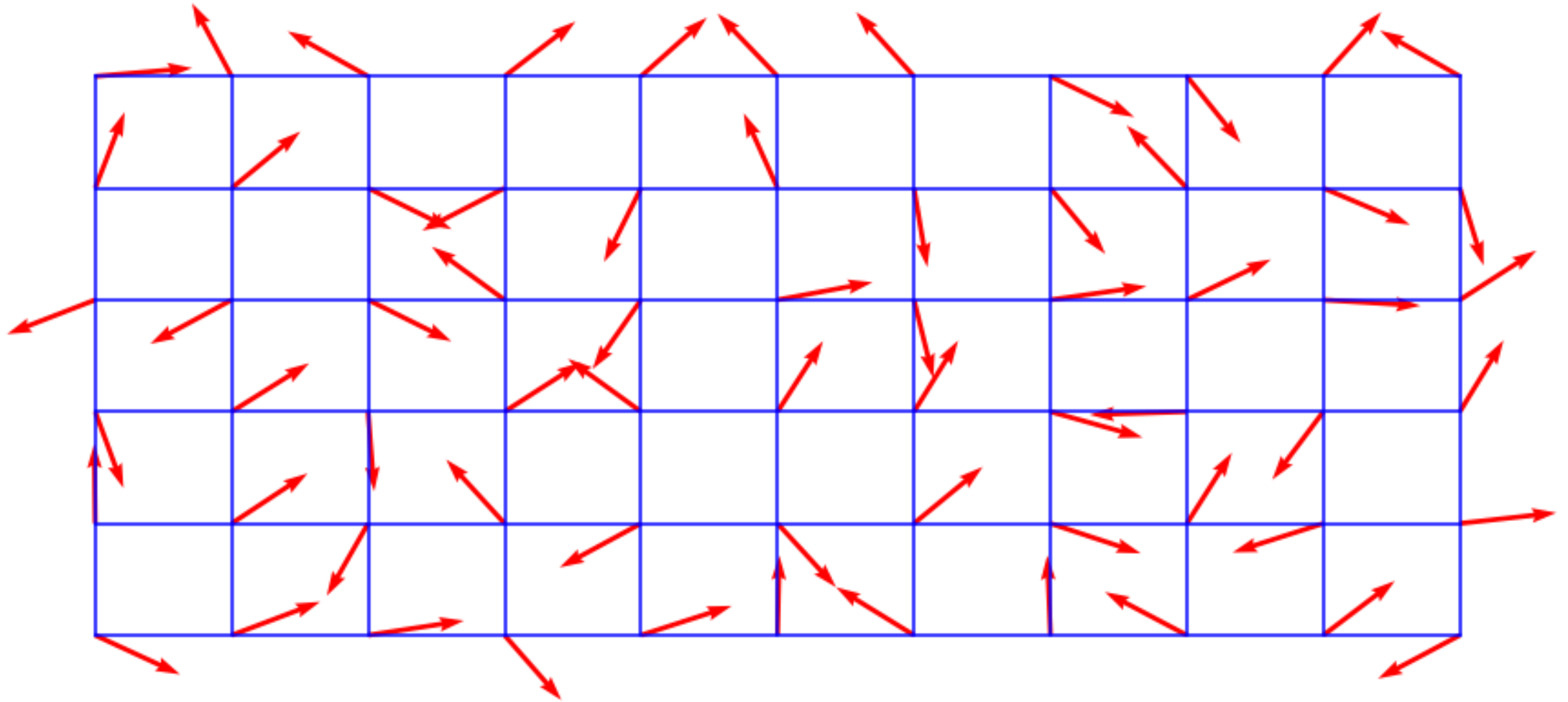
MESH WITH DISPLACEMENT

Normal Mapping vs. Displacement Mapping

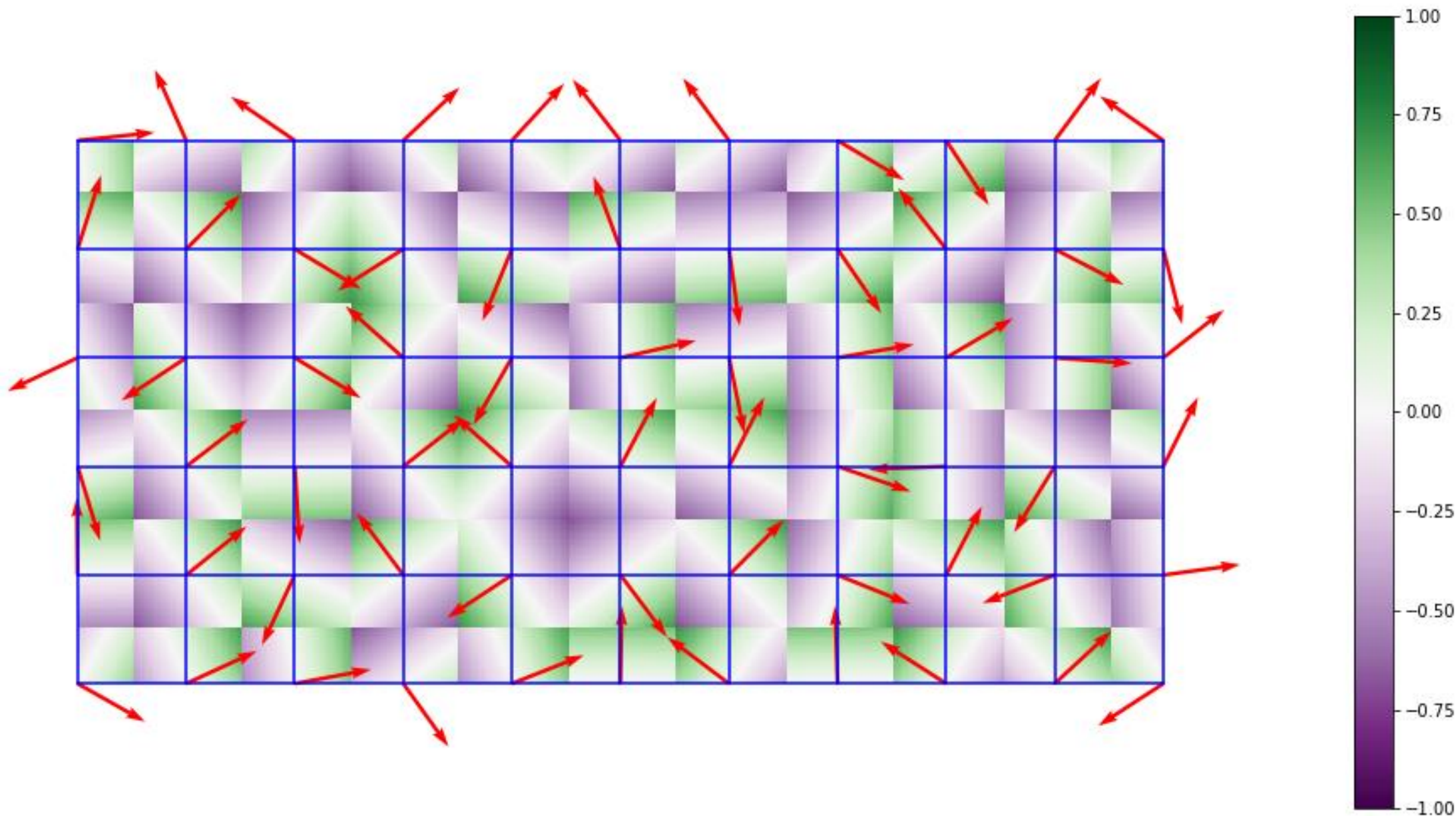


Perlin Noise

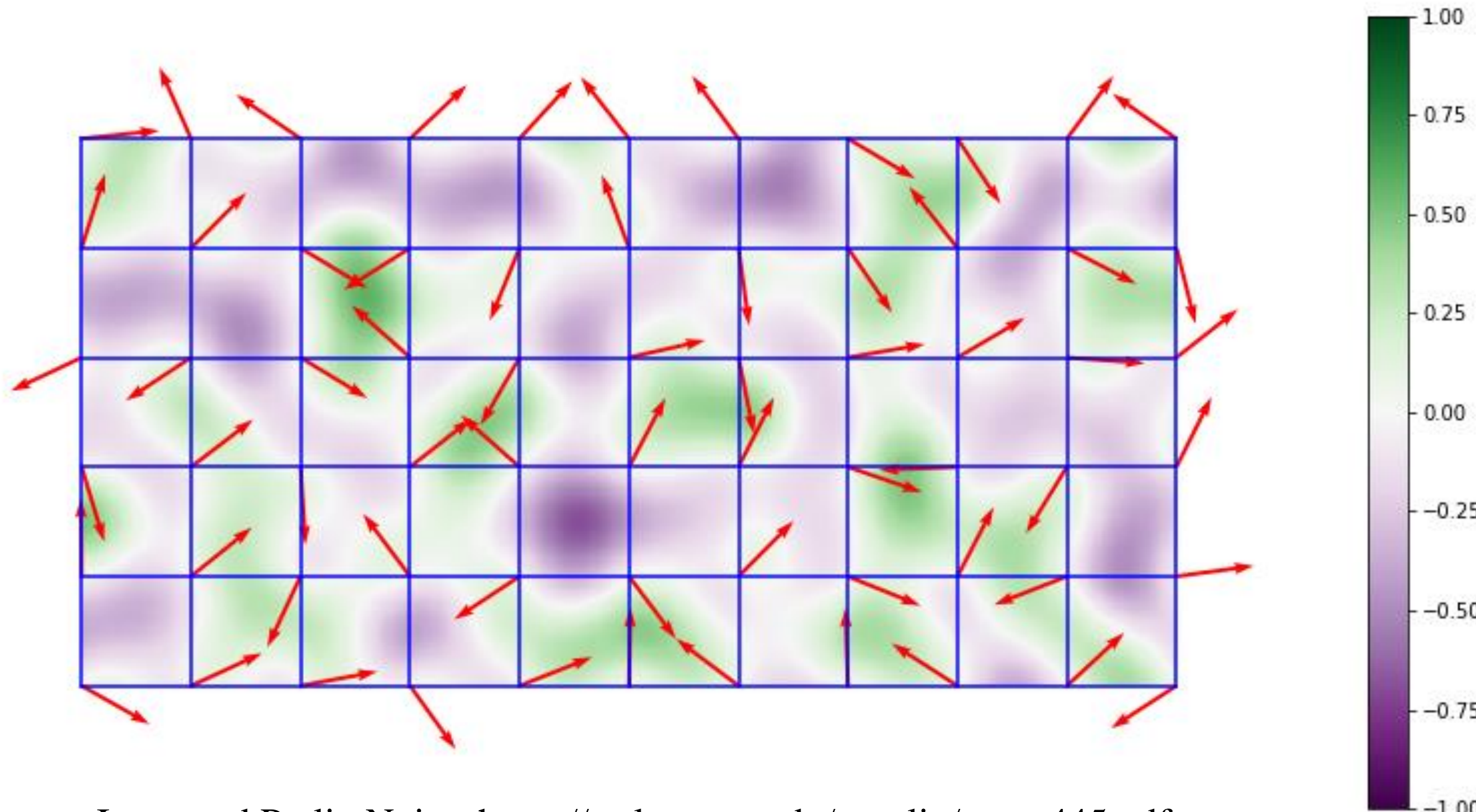




Improved Perlin Noise: <https://mrl.cs.nyu.edu/~perlin/paper445.pdf>



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 5th