Imatge Sintètica

Ray Tracing for Realistic Image Synthesis

Ricardo Marques
(ricardo.marques@upf.edu)

Group de Tecnologies Interactives (GTI)
Departament de Tecnologies de la Informació i les Comunicacions (DTIC)
Universitat Pompeu Fabra (UPF)

Edifici Tànger - Office 55.106

Lecture 1 - Introduction and Mathematical Background

2017/2018

Class Outline

Lecture 1 - Introduction and Mathematical Background

Course Presentation

Introduction
- The Fundamentals of Ray Tracing
- Ray Tracing Applications

Background
- Frames, Points, Vectors and Normals
- Matrices and Transformations
- Images

Course Presentation

- Workshop (Taller)
- Objectives
  - Develop your own ray tracer in C^{++}
  - Groups of 2 or 3 students
- Requirements
  - Knowledge of imperative programming
  - Preferable: notions of object oriented programming
- Evaluation
  - 5 assignments (50 %)
  - Final project and presentation (50 %)
- Emails: use [IS1718] in the beginning of the title

Course Presentation

- 5 Assignments
  - Each of them corresponding to the content of a theory class
- 2 deadlines to deliver the assignments
  - First deadline: assignments 1 and 2
    - 2nd May 2018
    - Just before the third theory class
  - Second deadline: assignments 3, 4 and 5
    - 25th May 2018
    - One week after the fifth theory class
- 1 deadline to deliver the final project and its presentation
  - 12th June 2018
  - Presentations will be made in the last session(s) depending on the number of groups
- All deliverables are made per group in *Aula Global*

Section 2

Introduction

Subsection 1

The Fundamentals of Ray Tracing

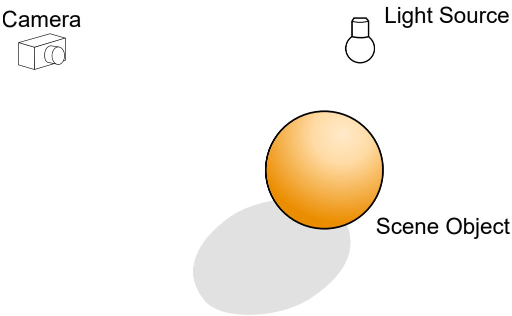
Objective

- Compute realistic images from a model of a virtual scene

- Should look like a real photo of virtual scene

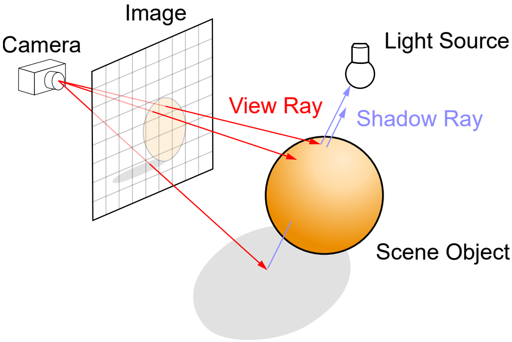
Requirements

- A full description of the virtual scene
  - Objects (geometry and materials) + light sources
- A virtual camera



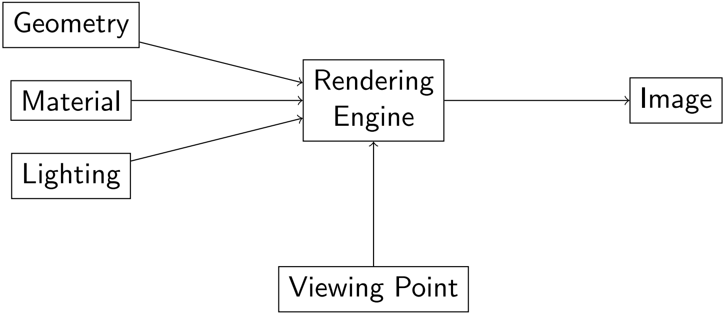
The Ray Tracing Principle

- Use rays to compute the light that enters the virtual camera
  - Simulate the light propagation from light sources
  - Model the light interaction with the scene materials



Rendering Engine Model

- Objective: representation and visualization of virtual objects



Subsection 2

Ray Tracing Applications

Ray Tracing Applications - Architecture

Ray Tracing Applications - Architecture

[Image from Andries van Dam]

Ray Tracing Applications - Architecture

[Image from CastleView3D]

Ray Tracing Applications - Films

- Real time ray tracing!
  - Mostly specular reflections and textures (fast to compute)

[Image from NVidia]

Ray Tracing Applications - Films

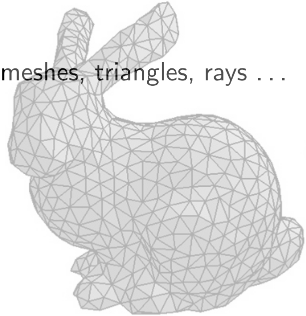
[Image from Pixar]

Section 3

Background

Motivation

- Ray Tracing requires accurate representation and manipulation of 3D entities
- Examples of 3D entities
  - Points, vectors, normals at surfaces, meshes, triangles, rays . . .
- Examples of manipulations
  - Translation, scale, rotation...



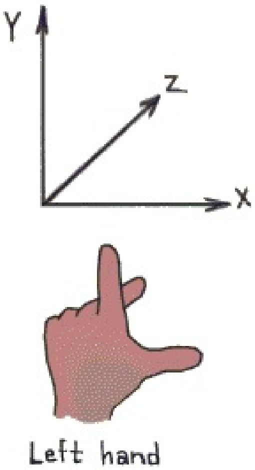
[Image from Geowave Library]

Subsection 1

Frames, Points, Vectors and Normals

3D Frame

- To represent points and vectors in a 3D space we need a 3D frame
- A 3D frame is a coordinate system represented by an origin and three base vectors
- World frame
  - Specifies the world space coordinates
  - Origin $(x, y, z) = (0, 0, 0)$
  - Base vectors $(1, 0, 0)$, $(0, 1, 0)$ and $(0, 0, 1)$
- In this course we will use a *left-handed* coordinate system



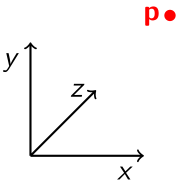
[Image from George H. Otto]

3D Point

- A 3D point **p** represents a *location* in the 3D space
- Defined by three coordinates

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

- Represented as a column vector (hence the transpose $\intercal$)



3D Vector

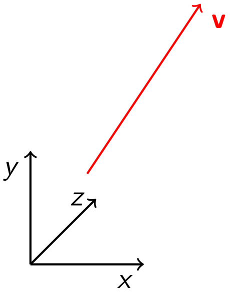
- A 3D vector $\mathbf{v}$ represents a *direction* in the 3D space
- Defined by three coordinates

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

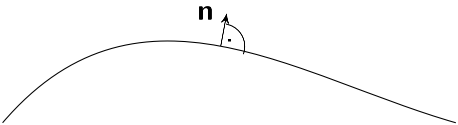
- Has a *magnitude* given by

$$\|\mathbf{v}\| = \sqrt{x^2 + y^2 + z^2}$$

- $\mathbf{v}$ is said to be normalized if $\|\mathbf{v}\| = 1$



Normal

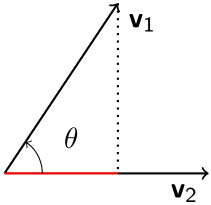


- Special case of a 3D vector
- Represents a 3D vector which is *perpendicular to a surface*
  - this property should be preserved by eventual transformations
- Attention! A normal vector might not be normalized!
  - meaning that $\|\mathbf{n}\|$ might be $\neq 1$

Dot Product

- Let $\mathbf{v}_1 = (x_1, y_1, z_1)^\intercal$ and $\mathbf{v}_2 = (x_2, y_2, z_2)^\intercal$
- The *dot product* between $\mathbf{v}_1$ and $\mathbf{v}_2$ is given by:

$$\begin{aligned}\mathbf{v}_1 \cdot \mathbf{v}_2 &= \|\mathbf{v}_1\| \, \|\mathbf{v}_2\| \cos(\theta) \\ &= x_1\, x_2 + y_1\, y_2 + z_1\, z_2 \\ &= \mathbf{v}_1^\intercal \, \mathbf{v}_2\end{aligned}$$



- If $\mathbf{v}_1$ and $\mathbf{v}_2$ are normalized ($\|\mathbf{v}_1\| = \|\mathbf{v}_2\| = 1$), then:

$$\mathbf{v}_1 \cdot \mathbf{v}_2 = \cos(\theta)$$

- Fast alternative to avoid calling the cos function (costly)

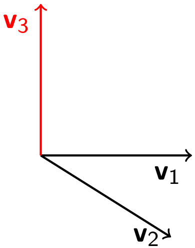
Cross Product

- Let $\mathbf{v}_1 = (x_1, y_1, z_1)^\intercal$ and $\mathbf{v}_2 = (x_2, y_2, z_2)^\intercal$
- The *cross product* between $\mathbf{v}_1$ and $\mathbf{v}_2$ yields a third vector $\mathbf{v}_3$:

$$\mathbf{v}_1 \times \mathbf{v}_2 = \mathbf{v}_3 = (x_3, y_3, z_3)^\intercal \, ,$$

where:

$$\begin{aligned} x_3 &= y_1\, z_2 - z_1\, y_2 \\ y_3 &= z_1\, x_2 - x_1\, z_2 \\ z_3 &= x_1\, y_2 - y_1\, x_2 \end{aligned}$$



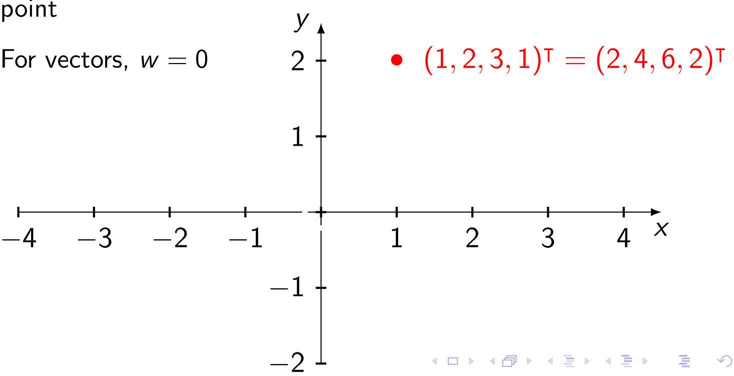
- How to arrive to the above expression?
- Some properties:
  - $\mathbf{v}_1 \times \mathbf{v}_2 = -\mathbf{v}_2 \times \mathbf{v}_1$
  - $\|\mathbf{v} \times \mathbf{v}\| = 0$ (Attention here!)

Subsection 2

Matrices and Transformations

Homogeneous Coordinates

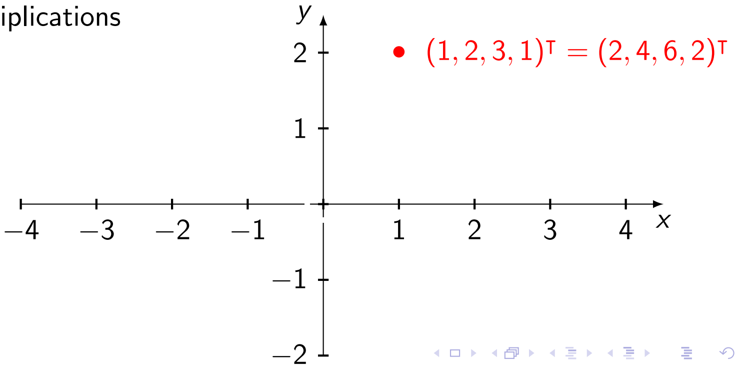
- *Homogeneous coordinates* allow distinguishing between points and vectors with the same (x, y, z) coordinates
- Consist of adding a 4^{th} coordinate w to each 3D point/vector
  - $\mathbf{p} = (x, y, z, w)^\intercal$ with $w \neq 0$ represents a point at $(\frac{x}{w}, \frac{y}{w}, \frac{z}{w})^\intercal$
  - $(1, 2, 3, 1)^\intercal$, $(2, 4, 6, 2)^\intercal$ and $(3, 6, 9, 3)^\intercal$ represent the same point
  - For vectors, $w = 0$



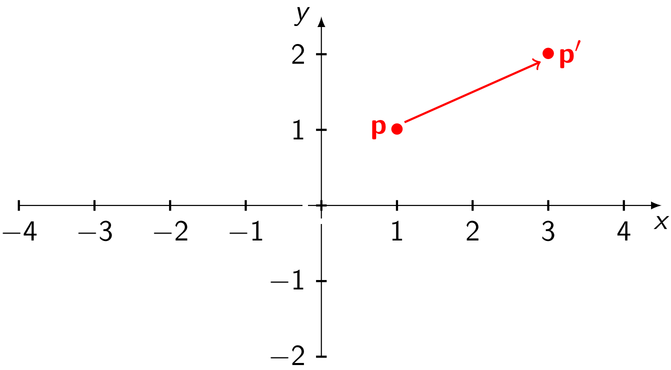
Homogeneous Coordinates

- Homogeneous coordinates allow us to represent transformations as 4x4 matrices
  - With this approach, we can represent different transforms using a single matrix
  - Example: rotation (3x3) and translation (column vector)

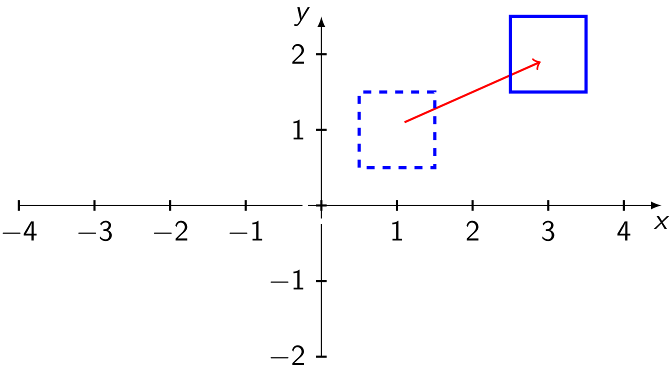
- Vectors and points are then transformed using matrix/vector multiplications



Translation



Translation



Translation

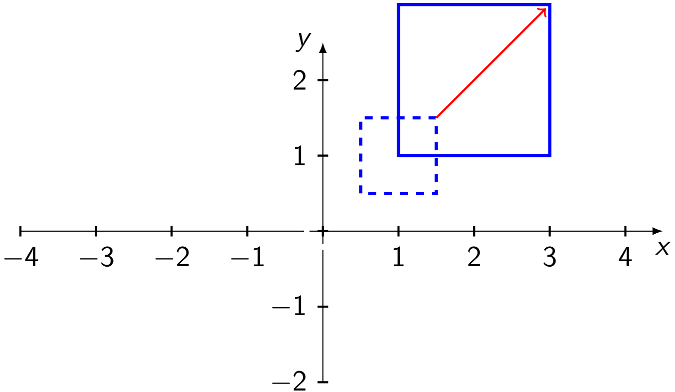
- A *translation* matrix $\mathbf{T}$ has the form $\begin{pmatrix} 1 & 0 & 0 & \boxed{\Delta_x} \\ 0 & 1 & 0 & \boxed{\Delta_y} \\ 0 & 0 & 1 & \boxed{\Delta_z} \\ 0 & 0 & 0 & 1 \end{pmatrix}$

- Multiplying $\mathbf{T}$ by a point $\mathbf{p} = (x, y, z, 1)^\intercal$ yields $\mathbf{p}'$

$$\mathbf{p}' = \begin{pmatrix} 1 & 0 & 0 & \Delta_x \\ 0 & 1 & 0 & \Delta_y \\ 0 & 0 & 1 & \Delta_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + \Delta_x \\ y + \Delta_y \\ z + \Delta_z \\ 1 \end{pmatrix}$$

- Multiplying $\mathbf{T}$ by a vector $\mathbf{v} = (x, y, z, 0)^\intercal$ leaves $\mathbf{v}$ unchanged
  - Indeed, translating a vector should not affect its value since it represents a direction

Scale



Scale

- A *scaling* matrix $\mathbf{S}$ has the form $\begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
- Multiplying $\mathbf{S}$ by a point $\mathbf{p} = (x, y, z, 1)^\mathsf{T}$ yields $\mathbf{p}'$

$$\mathbf{p}' = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} s_x\, x \\ s_y\, y \\ s_z\, z \\ 1 \end{pmatrix}$$

- Multiplying $\mathbf{S}$ by a vector $\mathbf{v} = (x, y, z, 0)^\mathsf{T}$ yields

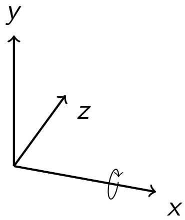
$$\mathbf{v}' = \mathbf{S}\,\mathbf{v} = (s_x\, x,\ s_y\, y,\ s_z\, z,\ 0)^\mathsf{T}$$

Rotation Around the **x**-axis

- A *rotation* matrix $\mathbf{R_x}(\theta)$ around **x** axis has the form

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

where θ gives the angle of rotation (clock-wise)



Rotation Around the **x**-axis

- Multiplying $\mathbf{R}_\mathbf{x}(\theta)$ by a point or vector given by $(x, y, z, w)^\intercal$, yields

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} x \\ y\ \cos(\theta) - z\ \sin(\theta) \\ y\ \sin(\theta) + z\ \cos(\theta) \\ w \end{pmatrix}$$

- x coordinate is unchanged

Rotation Around the **y**-axis and the **z**-axis

- Similar process to that of the rotation around the **x**-axis

$$\mathbf{R_y}(\theta) = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R_z}(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Transforming Normals

- Let $\mathbf{n}$ be a normalized vector perpendicular to $\mathbf{v}$ (such that $\mathbf{n} \cdot \mathbf{v} = 0$), and $\mathbf{S}$ be a scaling matrix, such that

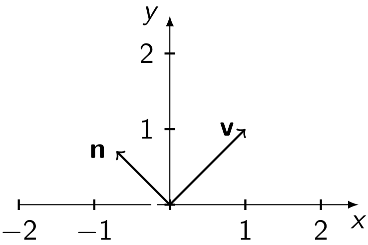
$$\mathbf{v} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{n} = \begin{pmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{S} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Suppose we transform $\mathbf{v}$ using $\mathbf{S}$ yielding:

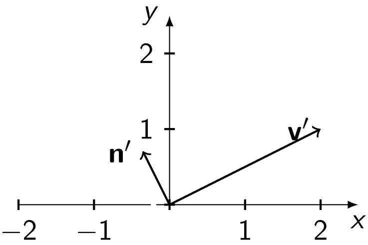
$$\mathbf{v}' = \mathbf{S}\,\mathbf{v} = (2, 1, 0, 0)^{\mathsf{T}}$$

- **Objective:** if $\mathbf{n} \perp \mathbf{v}$ then $\mathbf{n}' \perp \mathbf{v}'$

Transforming Normals



Before Transform



After Transform

$$\mathbf{v} = (1, 1, 0, 0)$$

$$\mathbf{n} = \left(-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0\right)$$

$$\mathbf{v}' = (2, 1, 0, 0)$$

$$\mathbf{n}' = (?, ?, ?, 0)$$

Transforming Normals

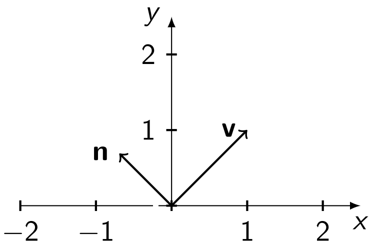
- What happens if we transform **n** directly using **S**?

$$\mathbf{n}' = \mathbf{S}\,\mathbf{n} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -2/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \\ 0 \end{pmatrix}$$

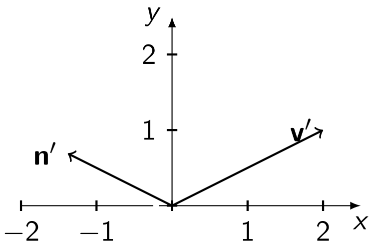
- Is $\mathbf{n}'$ perpendicular to $\mathbf{v}'$?

$$\mathbf{n}' \cdot \mathbf{v}' = (\mathbf{S}\,\mathbf{n}) \cdot (\mathbf{S}\,\mathbf{v}) = \begin{pmatrix} -2/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \\ 0 \\ 0 \end{pmatrix} = -3/\sqrt{2} \neq 0 \quad \text{NO...}$$

Transforming Normals



Before Transform



After Transform

$$\mathbf{v} = (1, 1, 0, 0)$$

$$\mathbf{n} = \left(-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0\right)$$

$$\mathbf{v}' = (2, 1, 0, 0)$$

$$\mathbf{n}' = \left(-\frac{2}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0\right)$$

Transforming Normals

- Recall the **objective**: if $\mathbf{n} \perp \mathbf{v}$ then $\mathbf{n}' \perp \mathbf{v}'$

- Expressing our objective as an equation

$$
\begin{aligned}
\mathbf{n} \cdot \mathbf{v} = \mathbf{n}' \cdot (\mathbf{S}\,\mathbf{v}) \Leftrightarrow \mathbf{n}^{\intercal}\,\mathbf{v} &= (\mathbf{n}')^{\intercal}\,\mathbf{S}\,\mathbf{v} \\
\Leftrightarrow \mathbf{n}^{\intercal} &= (\mathbf{n}')^{\intercal}\,\mathbf{S} \\
\Leftrightarrow \mathbf{n} &= \mathbf{S}^{\intercal}\,\mathbf{n}' \\
\Leftrightarrow \mathbf{n}' &= (\mathbf{S}^{\intercal})^{-1}\,\mathbf{n}
\end{aligned}
$$

- Given the above equality, if $\mathbf{n} \cdot \mathbf{v} = 0$ then $\mathbf{n}' \cdot \mathbf{v}' = 0$

Transforming Normals

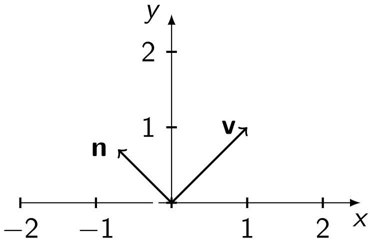
- Coming back to our example:

$$\mathbf{n}' = (\mathbf{S}^\intercal)^{-1}\,\mathbf{n} = \begin{pmatrix} 1/2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -1/(2\sqrt{2}) \\ 1/\sqrt{2} \\ 0 \\ 0 \end{pmatrix}$$

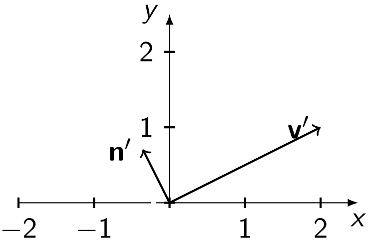
- Is $\mathbf{n}'$ perpendicular to $\mathbf{v}'$?

$$\mathbf{n}' \cdot \mathbf{v}' = \begin{pmatrix} -1/(2\sqrt{2}) \\ 1/\sqrt{2} \\ 0 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \\ 0 \\ 0 \end{pmatrix} = -\frac{2}{2\sqrt{2}} + \frac{1}{\sqrt{2}} = 0 \quad \text{YES} \smile$$

Transforming Normals



Before Transform



After Transform

$$\mathbf{v} = (1, 1, 0, 0)$$

$$\mathbf{n} = \left(-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0\right)$$

$$\mathbf{v}' = (2, 1, 0, 0)$$

$$\mathbf{n}' = \left(-\frac{1}{2\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0\right)$$

Subsection 3

Images

What is an Image

- An image is a 2D matrix with size given by its resolution ($width \times height$)
- Each element of this matrix is called pixel (picture element)
- Each pixel has an RBG value (red, green and blue)

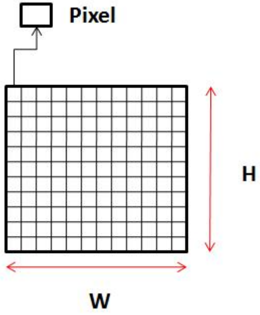
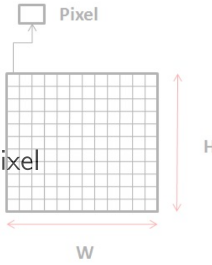
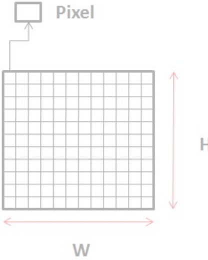


Image-Related Coordinate Spaces



- A pixel of an image can be accessed by using the pixel coordinates (x, y)
  - Such a space is usually called *image space*
  - $(x, y) \in [0, Width - 1] \times [0, Height - 1])$
- To be able to refer to different zones of a pixel (and not only to the pixel as a whole) it is useful to express the pixel location in normalized device coordinates (NDC)
  - (x, y) coordinates vary between $(0, 0)$ and $(1, 1)$
  - NDC space $\left((x, y) \in [0, 1]^2\right)$

Painting an Image in Red - Example



$$\text{for } (line = 0,\ line \leq height - 1,\ line{+}{+})$$

$$\{$$

$$\quad \text{for } (col = 0,\ col \leq width - 1,\ col{+}{+})$$

$$\quad \{$$

$$\qquad \text{image[lin][col]} = (1,\ 0,\ 0);$$

$$\quad \}$$

$$\}$$

Lecture Summary

- We have seen:
  - 3D points, 3D vectors and normals
  - Simple vector/vector operations (dot and cross products)
  - Common 3D transformations
  - How to transform points, vectors and normals
  - The notion of image
- This gives us the basic tools to learn ray tracing!

Next Classes at a Glance

- Next Seminar and Practical Class
  - Consolidate notions learned in this Lecture (Assignment 1)
- Next Lecture: learn the basics of a simple ray tracer
  - Projective camera models (orthographic and perspective)
  - Formalize the notion of ray
  - Intersect rays with spheres
  - How to generate rays from camera
  - First ray tracing image