Fundamentals of Computer Graphics

COMP 557

6 September 2016

Paul Kry

"Computers are useless. They can only give you answers"

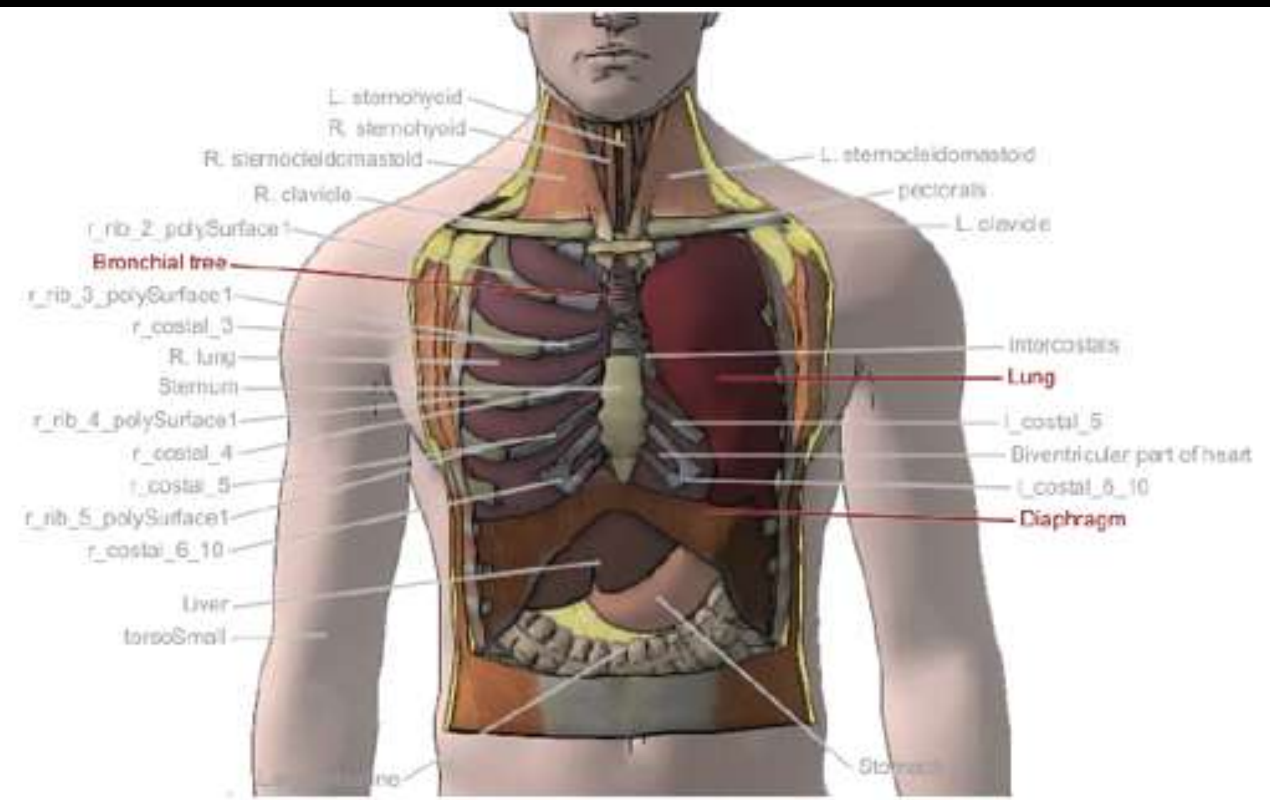
Pablo Picasso

Art

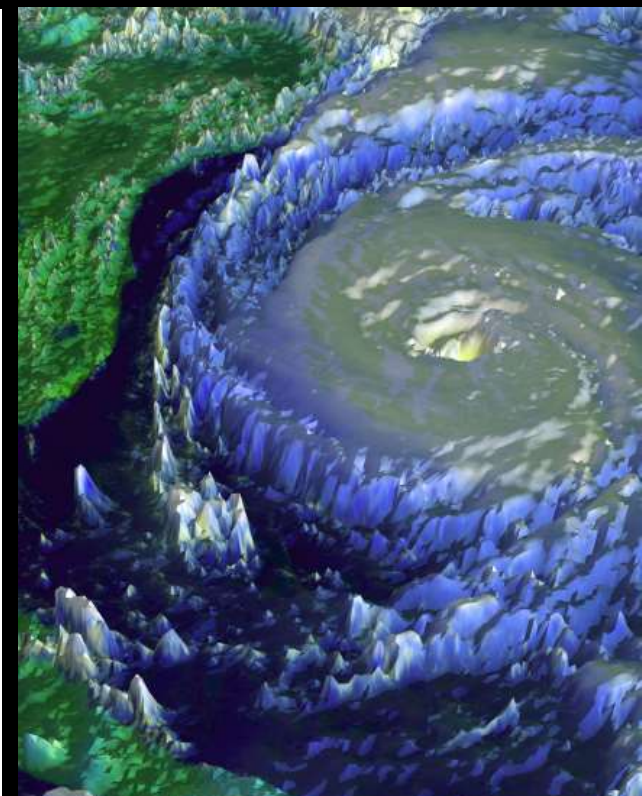
Kenneth A. Huff

Meats Meier

Visualization



[Li et al., 2007]

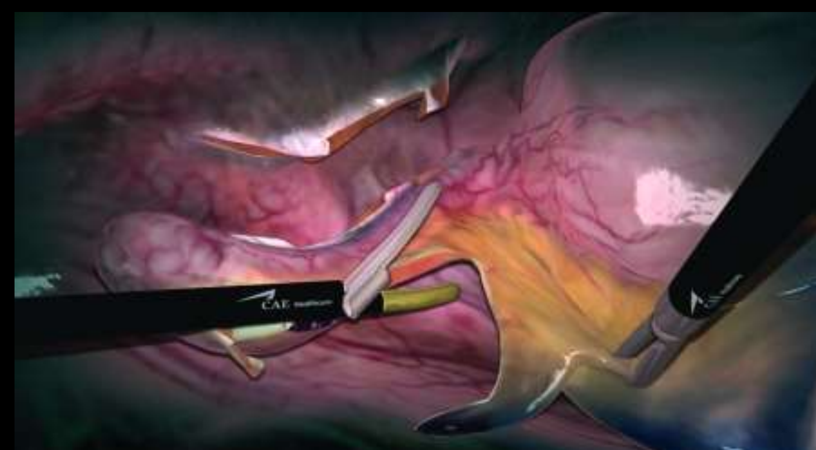


NASA

Training

CM Labs, crane simulator

Flight simulation



Appendectomy simulation (CAE LapVR)

Games

Movies

2007 FEATURE FILM WORK

[R&H Demo Reel]

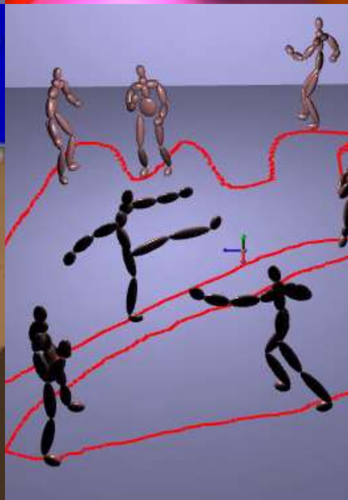
Other R&H work

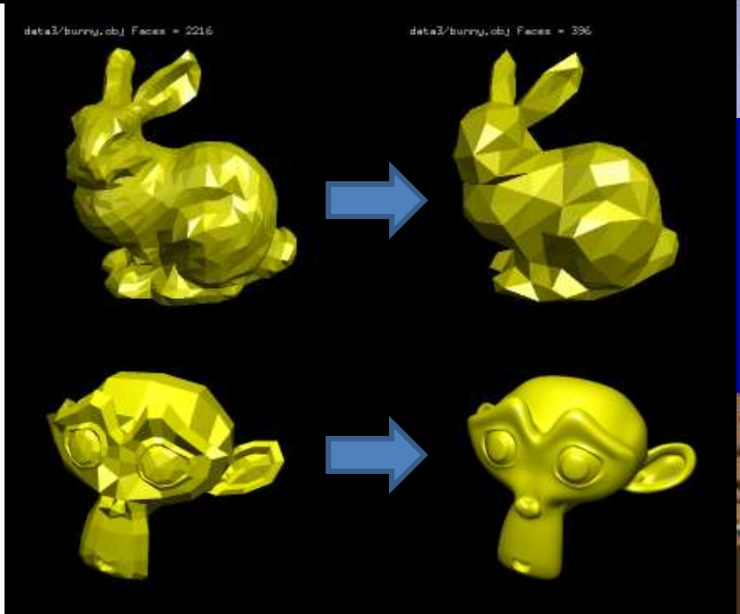
- Life of Pi VFX Breakdown
  - https://www.youtube.com/watch?v=PRE1Ot1sLTc
- Life after Pi
  - https://www.youtube.com/watch?v=TgSPys9PatU

What is Computer Graphics?

- A vast field that encompasses pretty much anything related to computer generated images
  - Modeling, rendering, animation, image processing, visualization, interactive techniques, etc.

[example assignments and projects from 557 and 559]



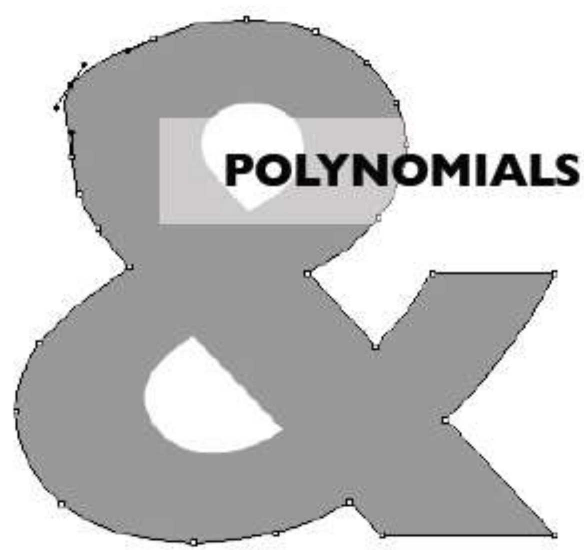


Computer Graphics

Mathematics made visible

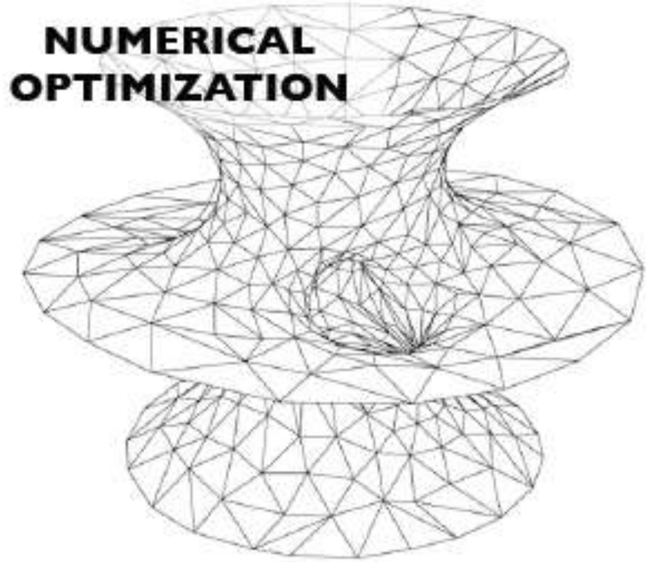
Problems in graphics

- 2D imaging
  - compositing and layering
  - digital filtering
  - color transformations
- 2D drawing
  - illustration, drafting
  - text, GUIs

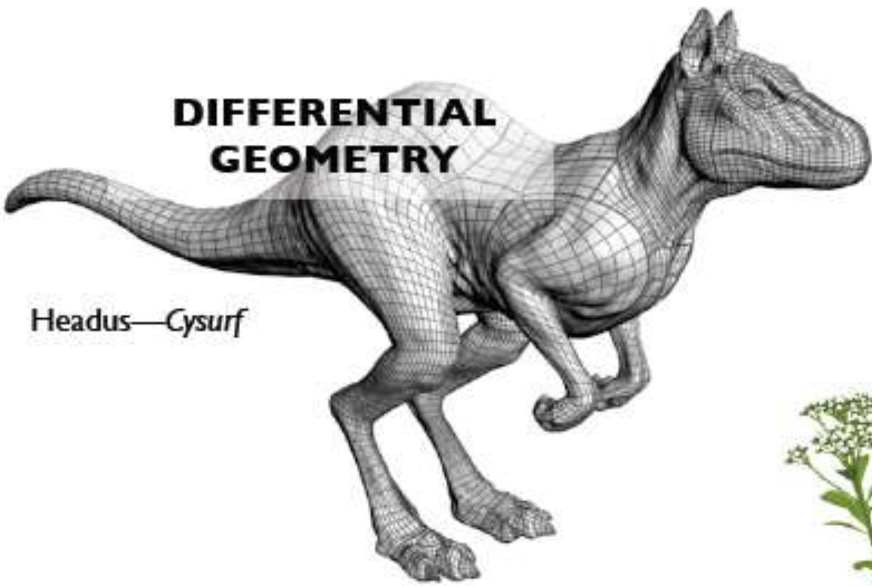


Problems in graphics CONT'D

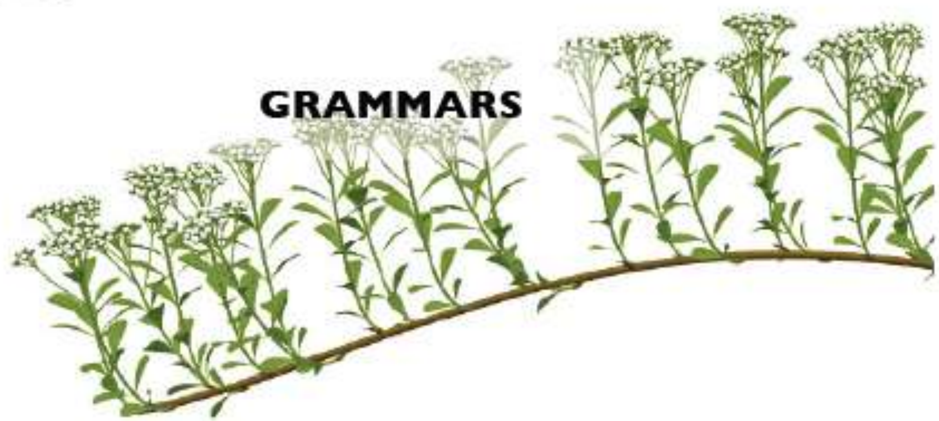
- 3D modeling
  - representing 3D shapes
  - polygons, curved surfaces, …
  - procedural modeling



[Hoppe et al. 1993]



Headus—*Cysurf*

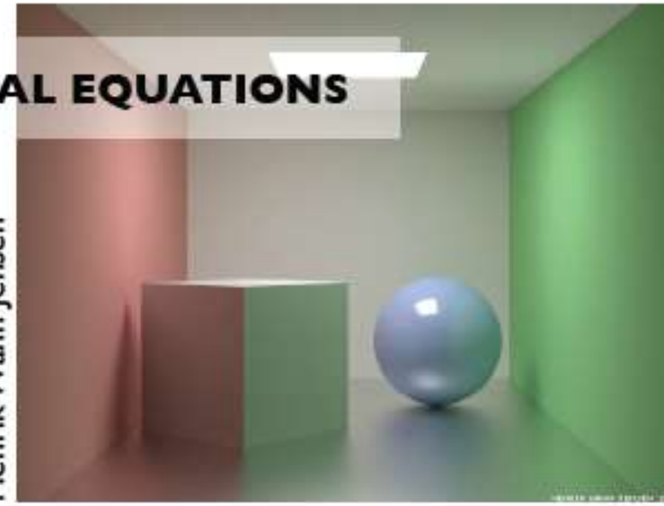


[Prusinkeiwicz et al. 2001]

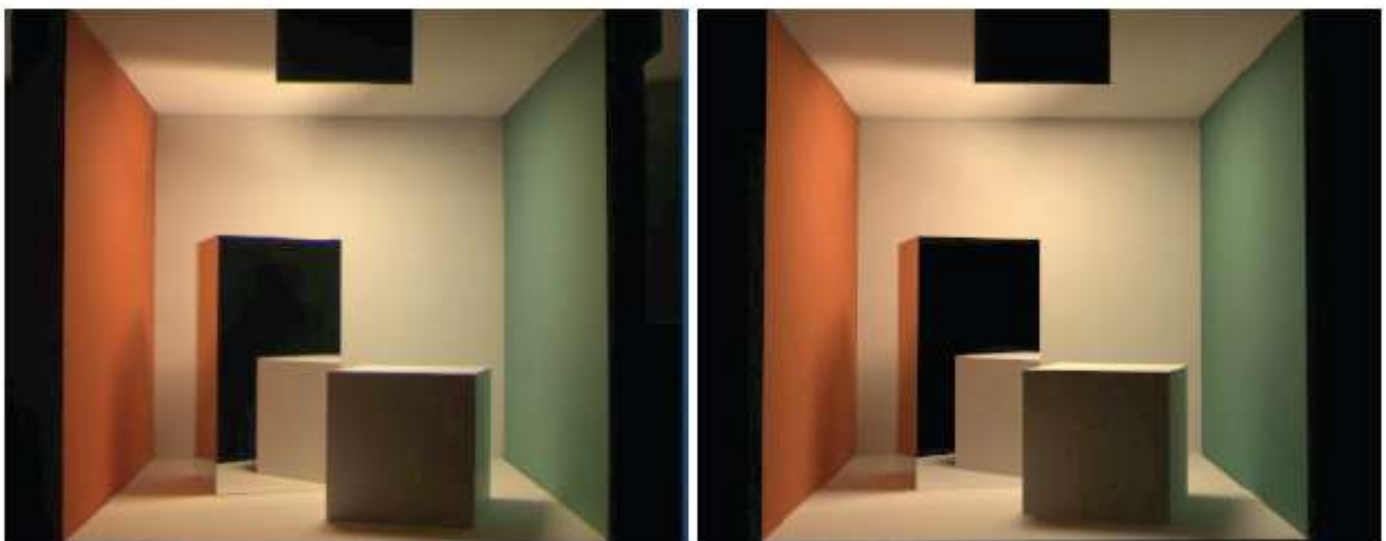
Problems in graphics CONT'D

- 3D rendering
  - 2D views of 3D geometry
  - projection and perspective
  - removing hidden surfaces
  - lighting simulation

INTEGRAL EQUATIONS



Henrik Wann Jensen



Cornell PCG

McGill

ECSE 689 | Some Motivation | Course Topics | Evaluation | Tasks | Contact Me

Welcome to ECSE 689

This class explores the technical details behind image synthesis techniques used for realistic visual effects in modern feature films, games, and architectural & product visualizations. Starting with the fundamentals of radiometry, the study and measurement of electromagnetic radiation (like visible light), we present mathematical models of how light propagates in an environment to eventually form an image on a sensor (e.g., a camera's film or the human eye).

Portal 2: a Valve Software production with realistic lighting.

Students will code the numerical methods used in industry for visual effects: each student will be guided through the implementation of a renderer capable of generating realistic images based on the physics of light. You can refer to the topics we'll cover in this class, and the programming assignments, below. You'll also find some motivational material & demos to whet your appetite.

Radiometry & Physics-based Shading

What is light? What are the physical units and intuitive interpretations behind the quantities most commonly used to measure light? How does light interact with a surface? What causes the differences in the appearance of e.g., a metal and a plastic?

Light Transport & Shading Algorithms

Once we understand how light behaves, we'll be able to derive the rendering equation, which governs the energy equilibrium of visible light in an environment. By solving the rendering equation, we can generate realistic images of virtual worlds.

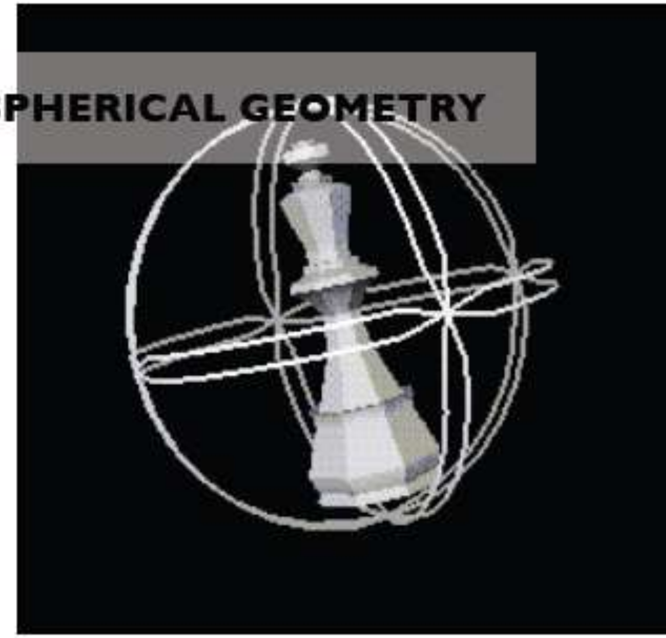
Getting your Hands Dirty

After establishing a solid (but not overly complex) theoretical foundation based on intuition and simple numerical principles, the focus will shift to developing practical hands-on experience.

While the emphasis here will be on

Problems in graphics CONT'D

- User Interaction
  - 2D graphical user interfaces
  - 3D modeling interfaces
  - virtual reality



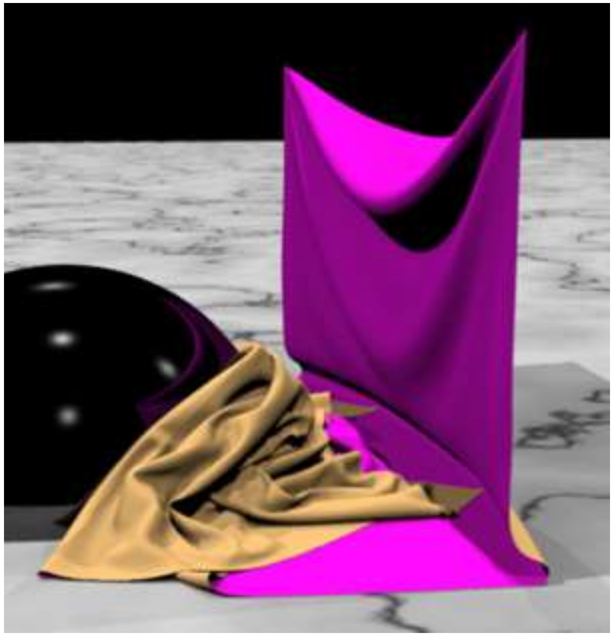
SGI—OpenInventor

TU Berlin

Problems in graphics CONT'D

- Animation
  - keyframe animation
  - physical simulation

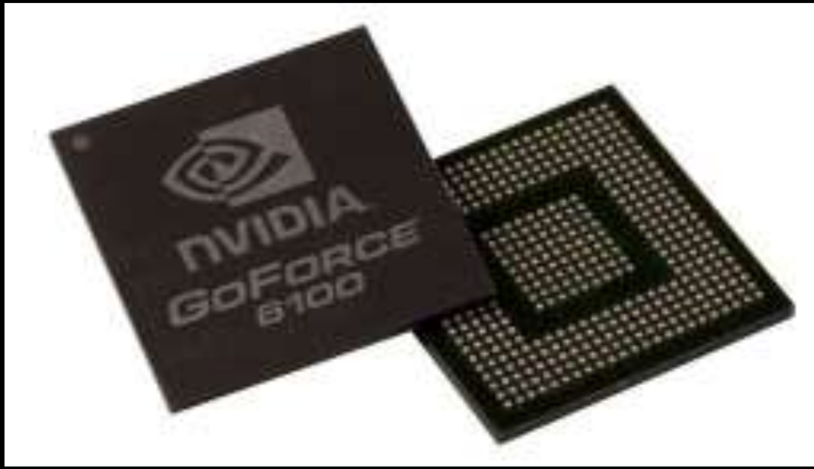
[Thürey et al. 2010]



[Bridson et al. 2002]

Evolution of computing environments

- Graphics has been a key to technology growth
  - Graphical user interfaces
  - Desktop publishing
  - Visualization
  - Gaming consoles
- Hardware revolution drives everything
  - price/performance improving exponentially (Moore's Law)
  - Graphics processors on even faster exponentials



[van Dam]

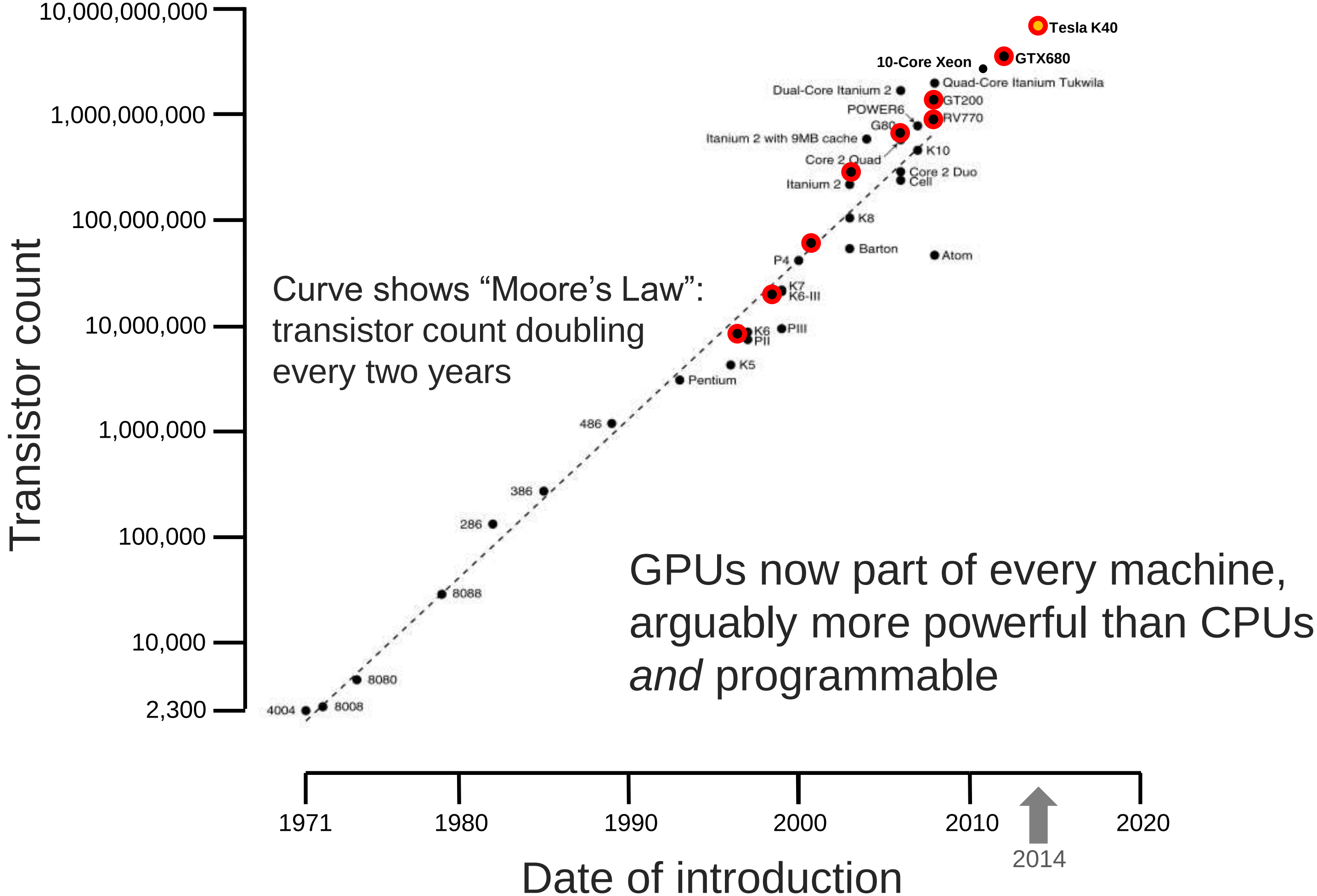
	Original Macintosh	New iMac 27”	
Date	1984	2014	+30
Price	$2500	$2000	x .8
CPU	8 MHz	3.4 GHz (Quad)	x 425 (x 1700)
Memory	128KB RAM	8.0GB DDR3 SDRAM (2 GB on GPU)	x 65536
Storage	400KB Floppy	1TB Hard Disk	x 2500000
Monitor	9” Black&White 512 x 342 68 dpi	27” Color 2560 x 1440 108 dpi	x 3 x 21 x 1.6
Devices	Mouse Keyboard	Mouse Keyboard	same same
GUI	Desktop WIMP	Desktop WIMP	same

[van Dam]

Graphics Processing Unit (GPU)

Not part of the previous comparison

CPU Transistor Counts from 1971 and Moore's Law



Today

- Research overview
- Introduction
- Course details
- Announcements
- Transforms

Who are you?

- 17 M Science
- 32 B Science Physics Earth Math
- 4 B Software Engineering
- 2 B Engineering
- 3 B Arts
- 3 Exchange / Special
- 1 Unknown / Other

Comp 557

- You will:
  - explore fundamental ideas
  - learn math essential to graphics
  - implement key algorithms
  - write cool programs
- You will not:
  - Become experts at OpenGL or DirectX
    (but you will become comfortable with OpenGL)
  - write huge programs

Prerequisites
(COMP206, COMP251, MATH223)

- Programming
  - ability to read (understand), write, and debug small Java programs (10s of classes)
  - understanding of basic data structures
  - no serious software design required
- Mathematics
  - vector geometry (dot/cross products, etc.)
  - linear algebra (matrices in 2D to 4D, linear systems)
  - basic calculus (derivatives and integrals)

Topics

- Coordinates, Transformations, Projections
- Meshes, Curves, Smooth Surfaces, Subdivision
- Lighting, Texturing, Rendering
- Images, Sampling
- Color science

Help getting started?

- Get started with Eclipse, JOGL
  - http://cs.mcgill.ca/~kry/comp557F16/asetup/
  - We will use vecmath, and some other jars too
  - In class, will provide some introduction to OpenGL, java bindings, and general graphics debugging strategies
- Need more help?
  - TA email on course web page, office hours on My Courses
  - Prof office hours 3pm Tuesdays

Evaluation

- Assignments 50%
- Midterm 20%
  - In October, during class time
- Final 30%

In case you didn't already know...

McGill University values academic integrity. Therefore, all students must understand the meaning and consequences of cheating, plagiarism and other academic offences under the Code of Student Conduct and Disciplinary Procedures. See www.mcgill.ca/integrity for more information, as well as www.mcgill.ca/integrity/studentguide, the Student Guide to Avoid Plagiarism.

It should be noted that, in accordance with article 15 of the Charter of Students' Rights, students may submit examination answers in either French or English.

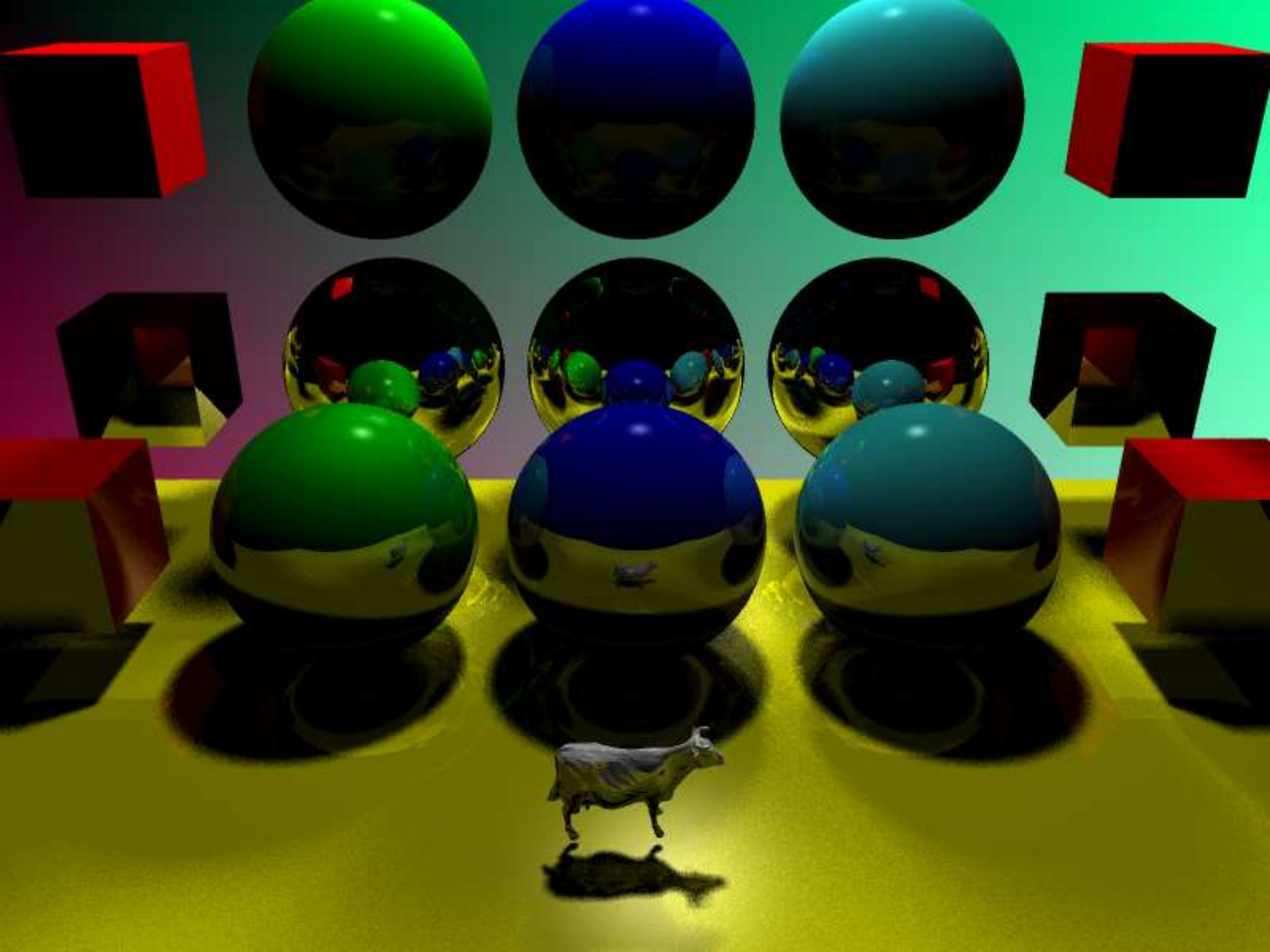
According to Senate regulations, instructors are not permitted to make special arrangements for final exams. Please consult the Calendar, section 4.7.2.1, General University Information and Regulations at www.mcgill.ca. Special arrangements in emergencies may be requested at your Student Affairs Office. If you have a disability, please advise the Office for Students with Disabilities (398-6009) as early in the term as possible so that we can provide appropriate accommodation to support your success.

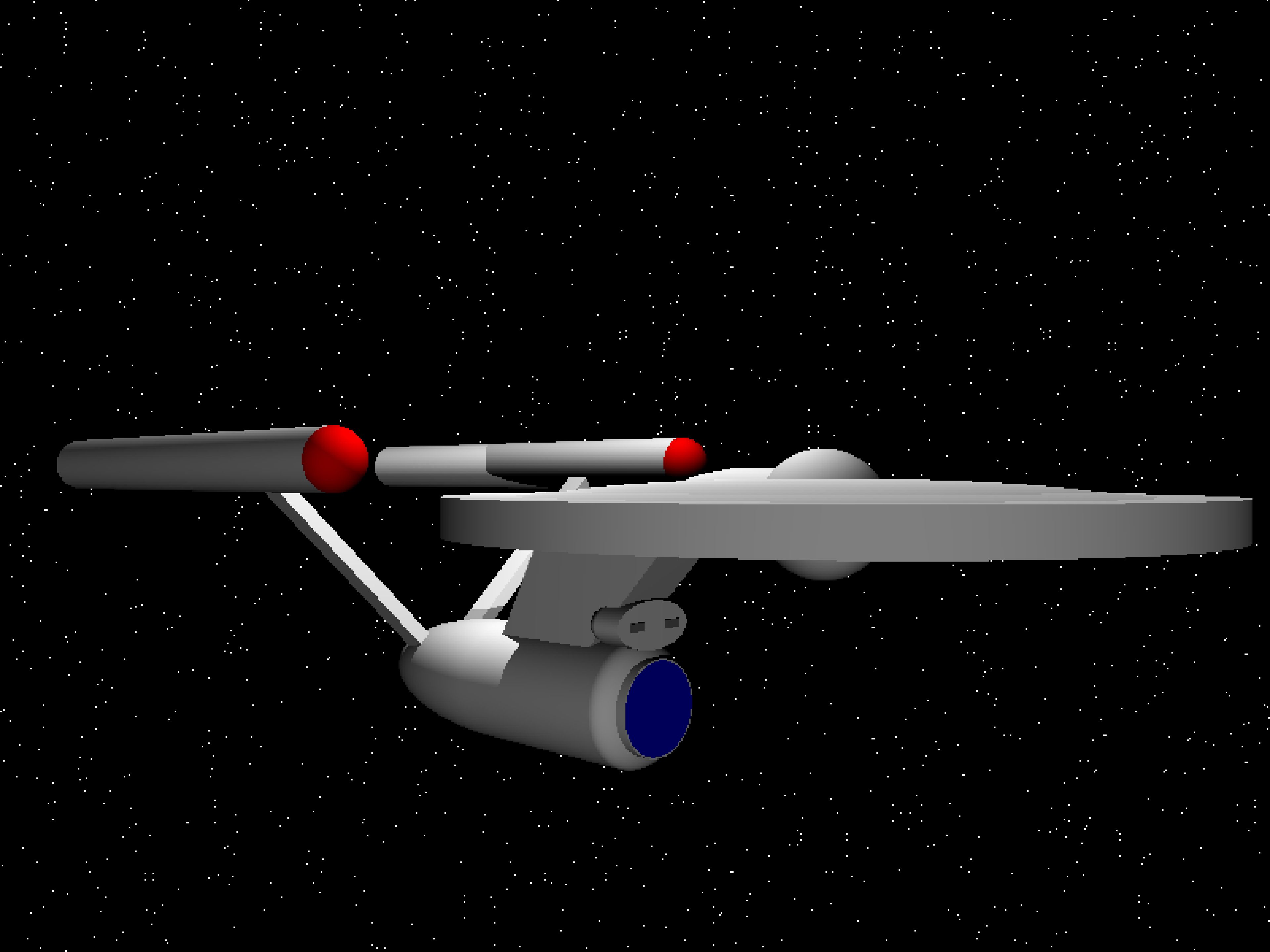
In the event of circumstances beyond the instructor's control, the evaluation scheme as set out in this document might require change. In such a case, every effort will be made to obtain consensus agreement from the class.

Additional policies governing academic issues which affect students can be found in the Handbook on Student Rights and Responsibilities, Charter of Students' Rights.

Assignments (four)

- Late Policy: 10% penalty, two days max
- Try “getting started” sample code now!
- Assignments
  - Rotations and transform hierarchies
  - Anaglyphs Projections
  - Mesh subdivision or simplification
  - Ray tracing (with competition)





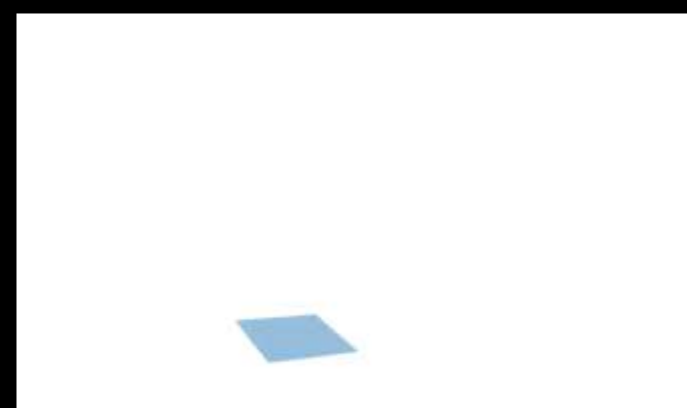
Today

- Research overview
- Introduction
- Course details
- Announcements
- Transforms

COMP 559 Fundamentals of Computer Animation

- Computational techniques for generating animation

- Physically based simulation of Rigid bodies, cloth, deformation, contact…
- Motion capture, reuse, retargeting, motion graphs, control…
- Learn by doing! Four assignments including a mini project.



Today

- Research overview
- Introduction
- Course details
- Announcements
- Transforms

Textbook

Fundamentals of Computer Graphics, 3rd ed., Peter Shirley and Steve Marschner.

- Chapter 1 – Read it for fun (some interesting bits)
- Chapter 2 – Misc Math (should all be review)
- Chapter 3 – We'll cover raster images ***later***
- Chapter 4 – We'll cover ray tracing ***later***
- Chapter 5 – Linear Algebra (should all be review)
- Chapter 6 – Transformation Matrices

A little quick math background

- Notation for sets, functions, mappings
- Linear transformations
- Matrices
  - Matrix-vector multiplication
  - Matrix-matrix multiplication
- Geometry of curves in 2D
  - Implicit representation
  - Explicit representation

Implicit representations

- Equation to tell whether we are on the curve
  $$\{\mathbf{v} \mid f(\mathbf{v}) = 0\}$$
- Example: line (orthogonal to **u**, distance *k* from **0**)
  $$\{\mathbf{v} \mid \mathbf{v} \cdot \mathbf{u} + k = 0\}$$
- Example: circle (center **p**, radius *r*)
  $$\{\mathbf{v} \mid (\mathbf{v} - \mathbf{p}) \cdot (\mathbf{v} - \mathbf{p}) - r^2 = 0\}$$
- Always define boundary of region
  - (if *f* is continuous)

Explicit representations

- Also called parametric
- Equation to map domain into plane
  $$\{f(t) \,|\, t \in D\}$$
- Example: line (containing **p**, parallel to **u**)
  $$\{\mathbf{p} + t\mathbf{u} \,|\, t \in \mathbb{R}\}$$
- Example: circle (center **b**, radius *r*)
  $$\{\mathbf{p} + r[\cos t \;\; \sin t]^T \,|\, t \in [0, 2\pi)\}$$
- Like tracing out the path of a particle over time
- Variable *t* is the "parameter"

Transforming geometry

- Move a subset of the plane using a mapping from the plane to itself

  $$S \to \{T(\mathbf{v}) \mid \mathbf{v} \in S\}$$

- Parametric representation:

  $$\{f(t) \mid t \in D\} \to \{T(f(t)) \mid t \in D\}$$

- Implicit representation:

  $$\{\mathbf{v} \mid f(\mathbf{v}) = 0\} \to \{T(\mathbf{v}) \mid f(\mathbf{v}) = 0\}$$

  $$= \{\mathbf{v} \mid f(T^{-1}(\mathbf{v})) = 0\}$$

Translation

- Simplest transformation: $T(\mathbf{v}) = \mathbf{v} + \mathbf{u}$
- Inverse: $T^{-1}(\mathbf{v}) = \mathbf{v} - \mathbf{u}$
- Example of transforming circle

Linear transformations

- One way to define a transformation is by matrix multiplication:

$$T(\mathbf{v}) = M\mathbf{v}$$

- Such transformations are *linear*, which is to say:

$$T(a\mathbf{u} + \mathbf{v}) = aT(\mathbf{u}) + T(\mathbf{v})$$

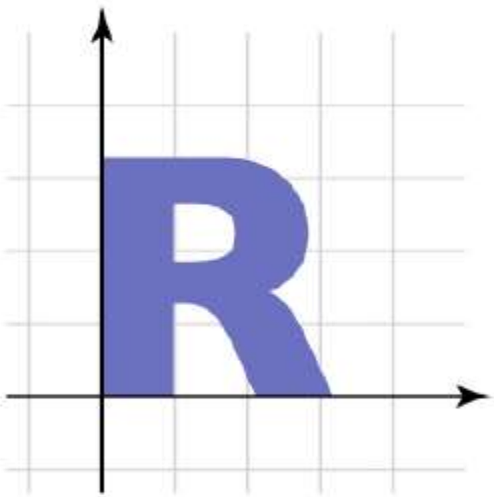
(and in fact all linear transformations can be written this way)

Geometry of 2D linear trans.

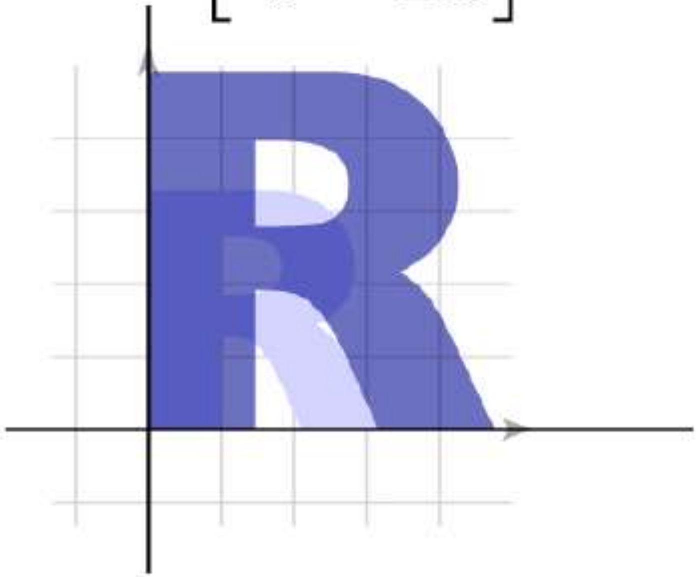
- 2x2 matrices have simple geometric interpretations
  - uniform scale
  - non-uniform scale
  - rotation
  - shear
  - reflection
- Reading off the matrix

Linear transformation gallery

- Uniform scale $\begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} sx \\ sy \end{bmatrix}$



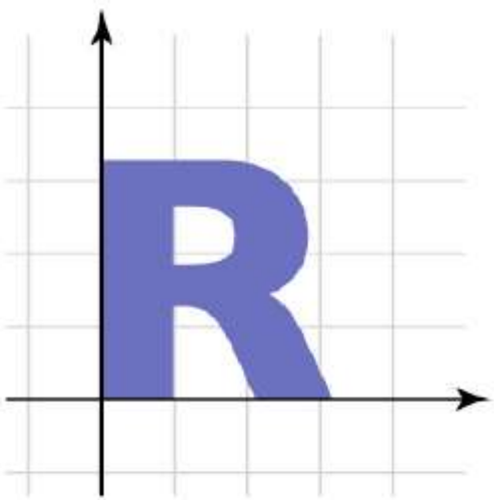
$$\begin{bmatrix} 1.5 & 0 \\ 0 & 1.5 \end{bmatrix}$$

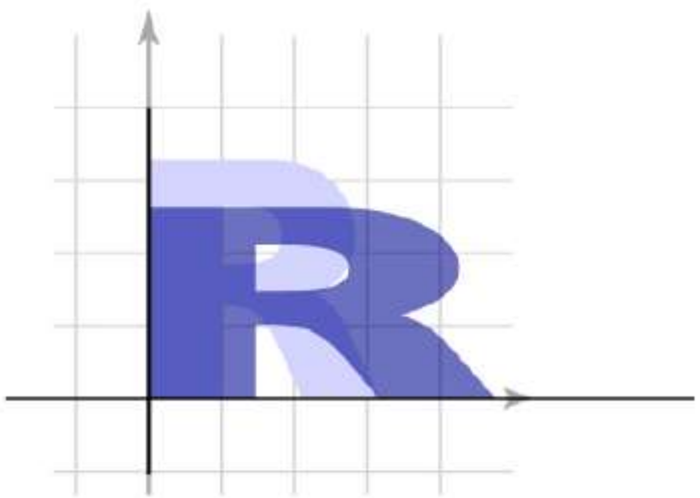


Linear transformation gallery

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

$$\begin{bmatrix} 1.5 & 0 \\ 0 & 0.8 \end{bmatrix}$$

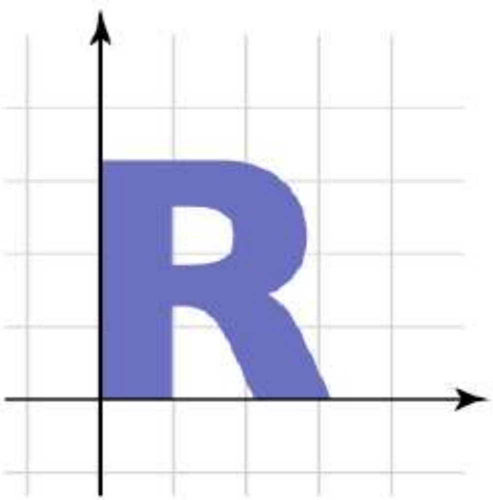


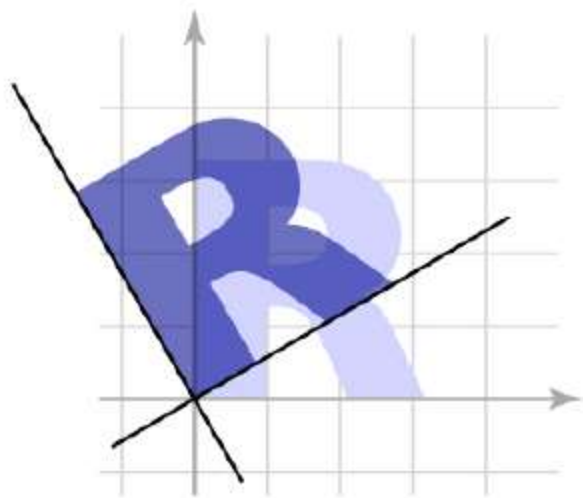


Linear transformation gallery

- Rotation $\begin{bmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}x \\ y\end{bmatrix} = \begin{bmatrix}x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta\end{bmatrix}$

$$\begin{bmatrix}0.866 & -0.5 \\ 0.5 & 0.866\end{bmatrix}$$

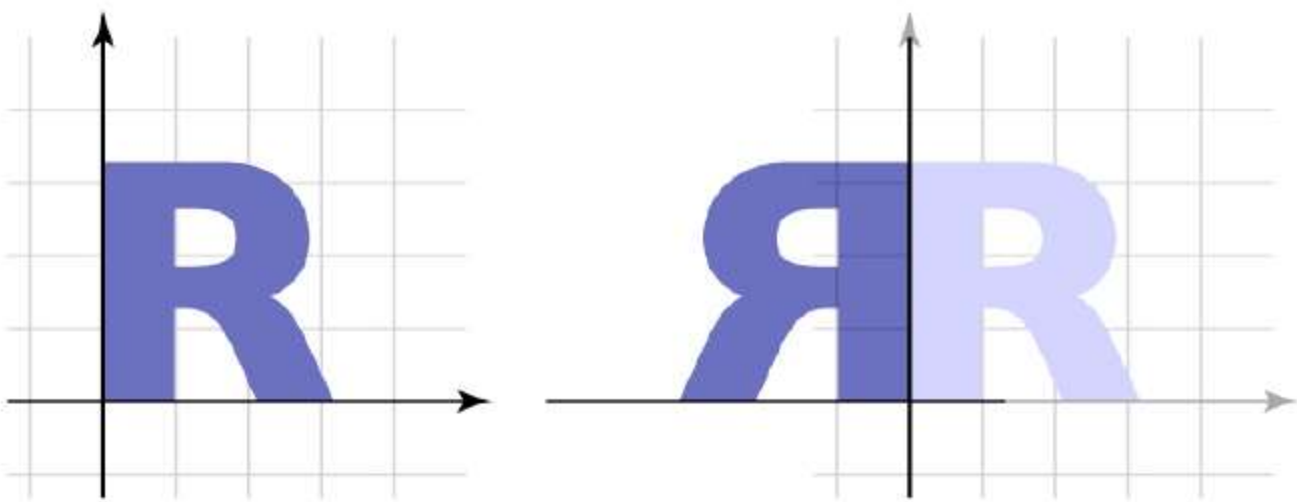




Linear transformation gallery

- Reflection
  - can consider it a special case of nonuniform scale

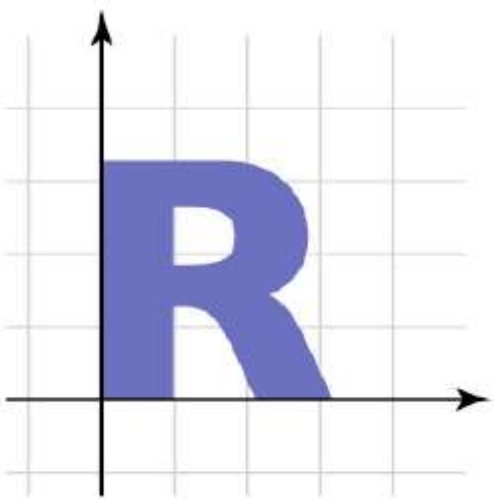
$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

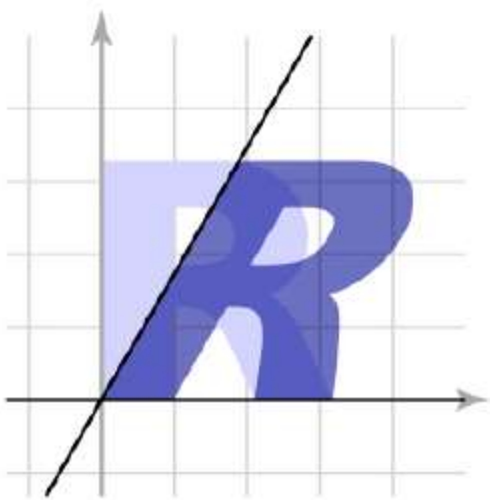


Linear transformation gallery

- Shear $\begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x + ay \\ y \end{bmatrix}$

$$\begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix}$$





Composing transformations

- Want to move an object, then move it some more
  - $\mathbf{p} \to T(\mathbf{p}) \to S(T(\mathbf{p})) = (S \circ T)(\mathbf{p})$
- We need to represent $S \circ T$ (“S compose T”)
  - and would like to use the same representation as for S and T
- Translation easy
  - $T(\mathbf{p}) = \mathbf{p} + \mathbf{u}_T; S(\mathbf{p}) = \mathbf{p} + \mathbf{u}_S$

$$(S \circ T)(\mathbf{p}) = \mathbf{p} + (\mathbf{u}_T + \mathbf{u}_S)$$

- Translation by $\mathbf{u}_T$ then by $\mathbf{u}_S$ is translation by $\mathbf{u}_T + \mathbf{u}_S$
  - commutative!

Composing transformations

- Linear transformations also straightforward
  - $$T(\mathbf{p}) = M_T\mathbf{p}; S(\mathbf{p}) = M_S\mathbf{p}$$
    $$(S \circ T)(\mathbf{p}) = M_S M_T \mathbf{p}$$
- Transforming first by M_T then by M_S is the same as transforming by $M_S M_T$
  - only sometimes commutative
    - e.g. rotations & uniform scales
    - e.g. non-uniform scales w/o rotation
  - Note $M_S M_T$, or $S \circ T$, is T first, then S

Combining linear with translation

- Need to use both in single framework
- Can represent arbitrary seq. as $T(\mathbf{p}) = M\mathbf{p} + \mathbf{u}$
  - $T(\mathbf{p}) = M_T\mathbf{p} + \mathbf{u}_T$
  - $S(\mathbf{p}) = M_S\mathbf{p} + \mathbf{u}_S$
  - $(S \circ T)(\mathbf{p}) = M_S(M_T\mathbf{p} + \mathbf{u}_T) + \mathbf{u}_S$

    $$= (M_S M_T)\mathbf{p} + (M_S\mathbf{u}_T + \mathbf{u}_S)$$
  - e.g. $S(T(0)) = S(\mathbf{u}_T)$
- Transforming by M_T and $\mathbf{u}_T$, then by M_S and $\mathbf{u}_S$, is the same as transforming by $M_S M_T$ and $\mathbf{u}_S + M_S\mathbf{u}_T$
  - This will work but is a little awkward

Homogeneous coordinates

- A trick for representing the foregoing more elegantly
- Extra component *w* for vectors, extra row/column for matrices
  - for affine, can always keep *w* = 1
- Represent linear transformations with dummy extra row and column

$$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \\ 1 \end{bmatrix}$$

Homogeneous coordinates

- Represent translation using the extra column

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

Homogeneous coordinates

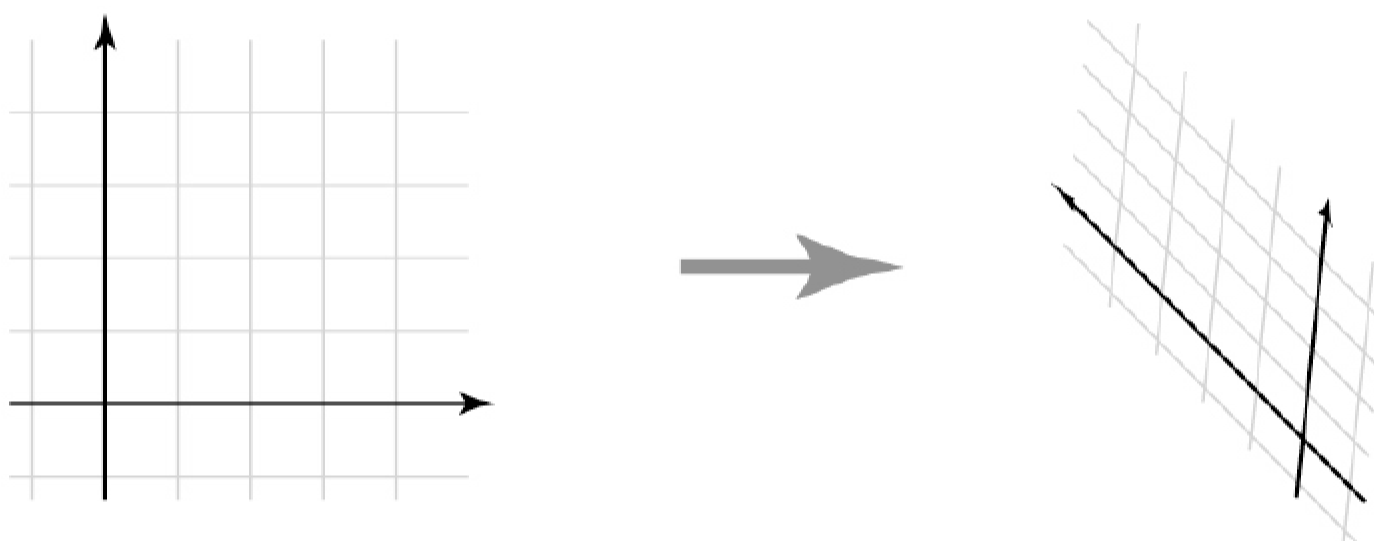
- Composition just works, by 3x3 matrix multiplication

$$\begin{bmatrix} M_S & \mathbf{u}_S \\ 0 & 1 \end{bmatrix} \begin{bmatrix} M_T & \mathbf{u}_T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} = \begin{bmatrix} (M_S M_T)\mathbf{p} + (M_S \mathbf{u}_T + \mathbf{u}_S) \\ 1 \end{bmatrix}$$

- This is exactly the same as carrying around *M* and **u**
  - but cleaner
  - and generalizes in useful ways as we'll see later

Affine transformations

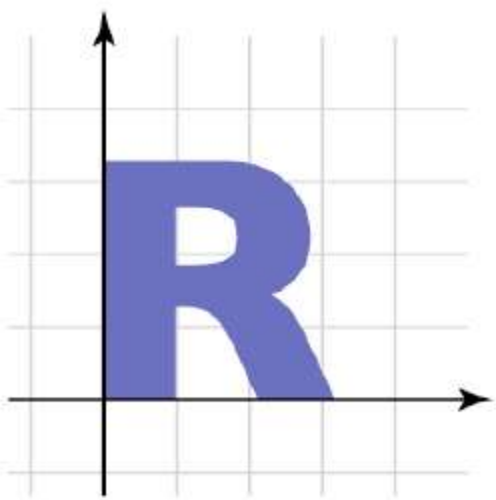
- The set of transformations we have been looking at is known as the "affine" transformations
  - straight lines preserved; parallel lines preserved
  - ratios of lengths along lines preserved (midpoints preserved)

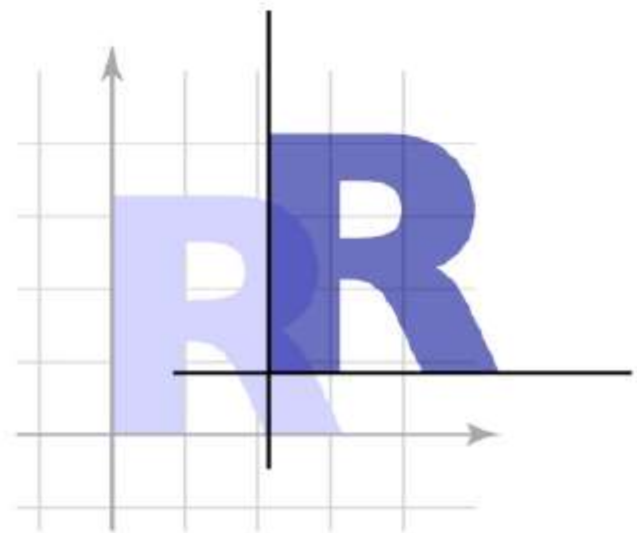


Affine transformation gallery

- Translation

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 2.15 \\ 0 & 1 & 0.85 \\ 0 & 0 & 1 \end{bmatrix}$$

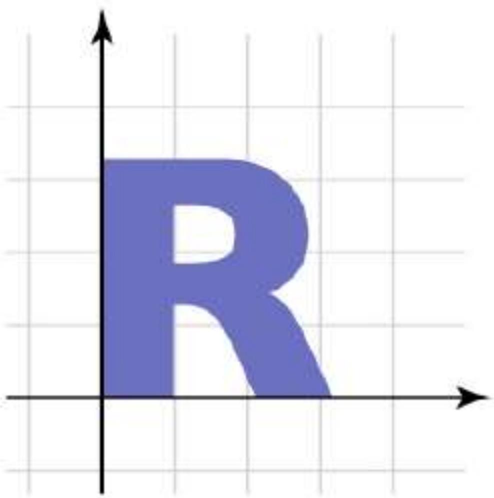


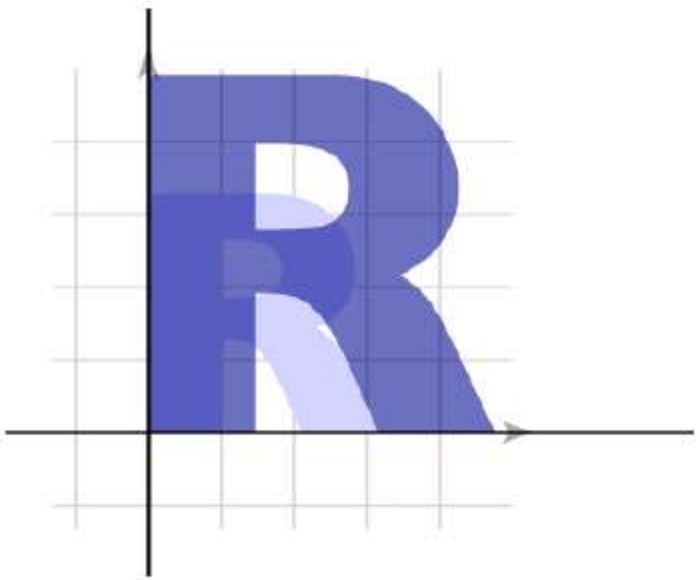


Affine transformation gallery

- Uniform scale

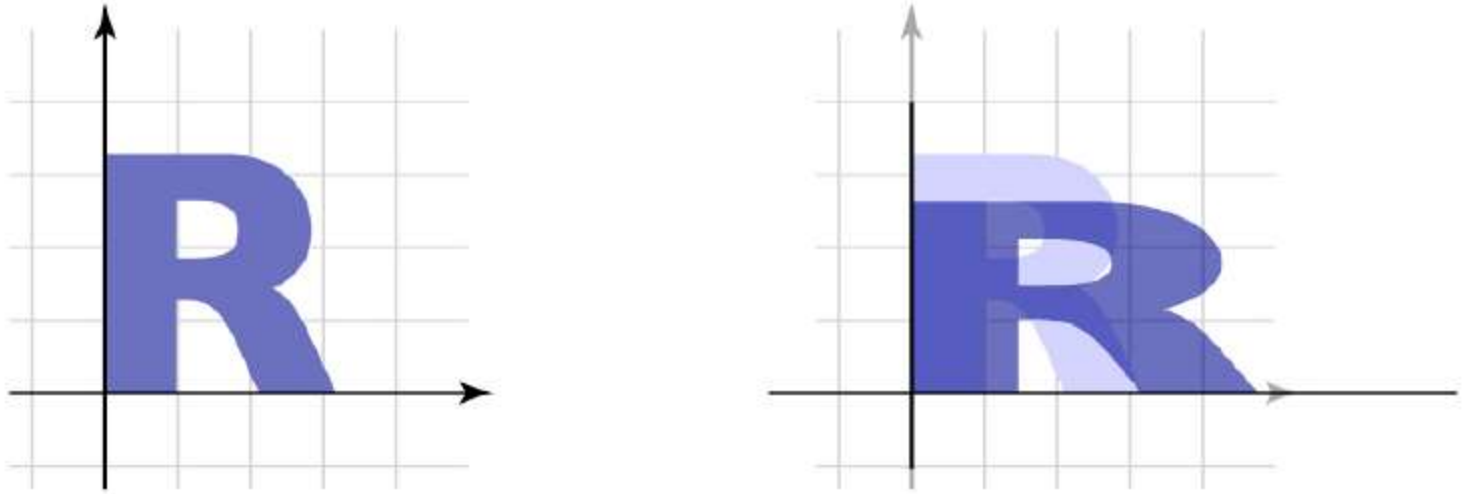
$$\begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1.5 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$





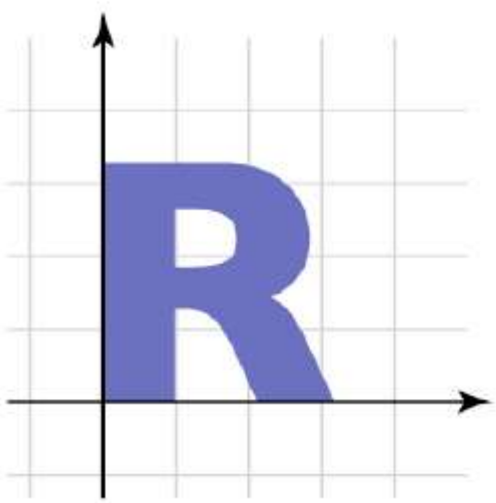
Affine transformation gallery

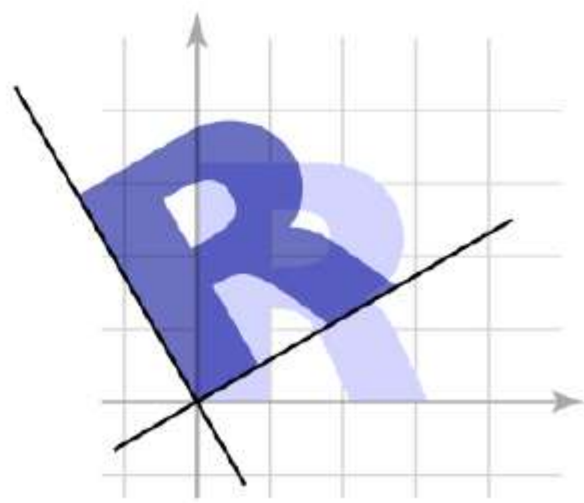
- Nonuniform scale $\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$ $\begin{bmatrix} 1.5 & 0 & 0 \\ 0 & 0.8 & 0 \\ 0 & 0 & 1 \end{bmatrix}$



Affine transformation gallery

- Rotation $\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$ $\begin{bmatrix} 0.866 & -0.5 & 0 \\ 0.5 & 0.866 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

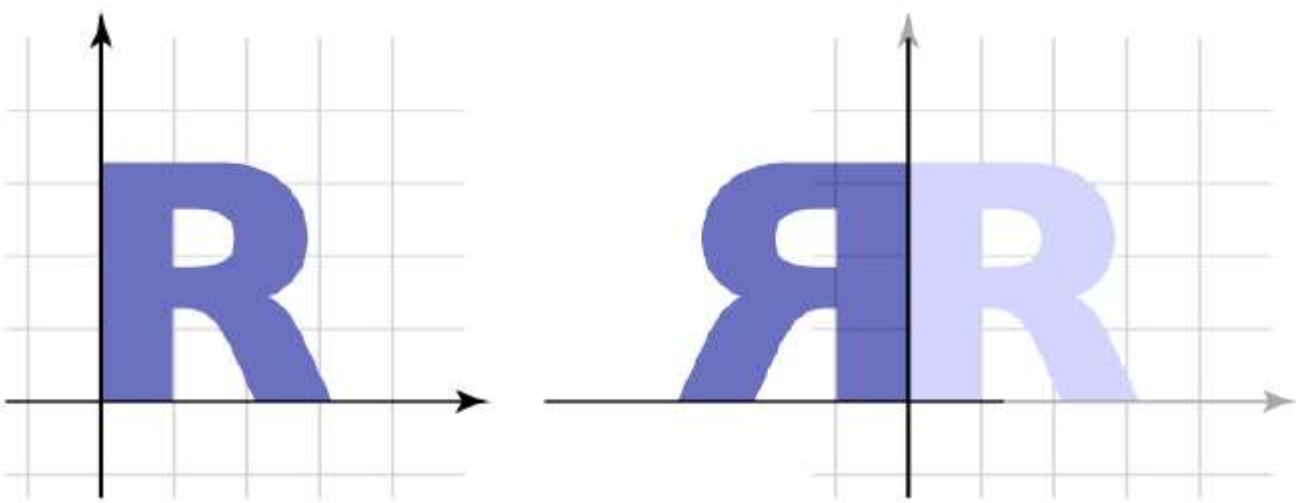




Affine transformation gallery

- Reflection
  – can consider it a special case of nonuniform scale

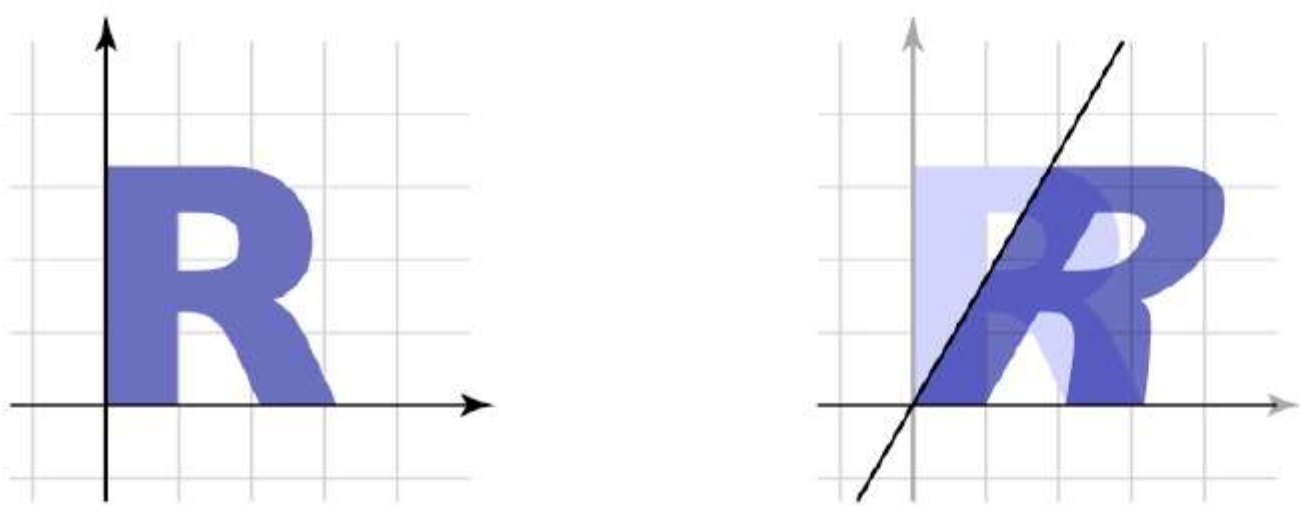
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Affine transformation gallery

- Shear

$$\begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

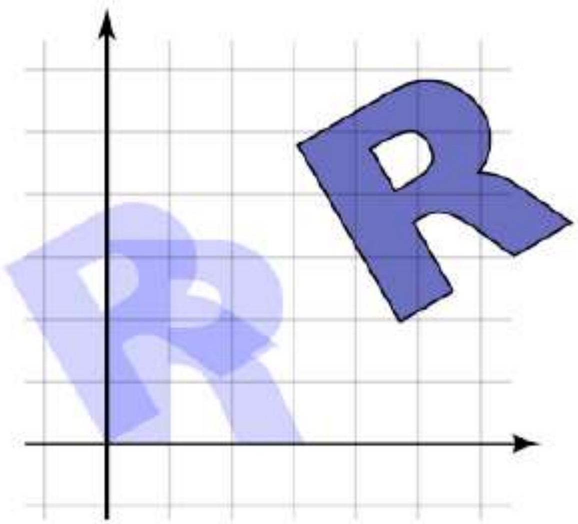


General affine transformations

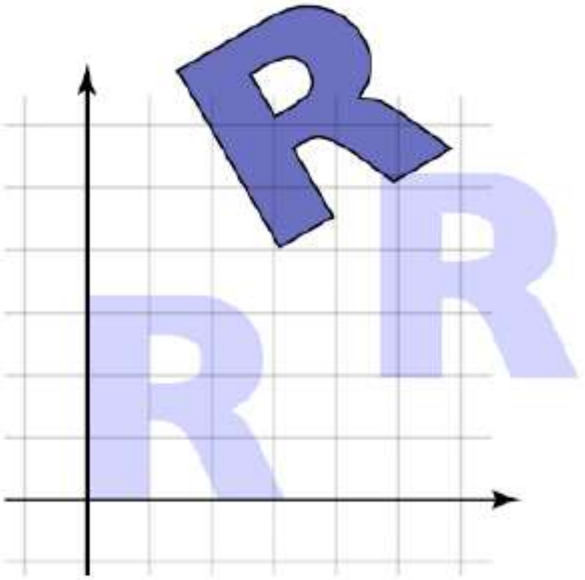
- The previous slides showed "canonical" examples of the types of affine transformations
- Generally, transformations contain elements of multiple types
  - often define them as products of canonical transforms
  - sometimes work with their properties more directly

Composite affine transformations

- In general **not** commutative: order matters!



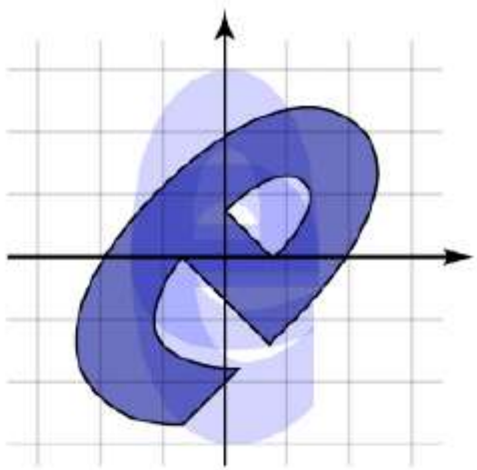
rotate, then translate



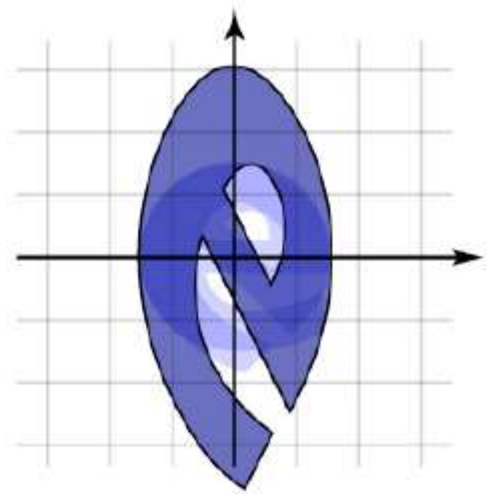
translate, then rotate

Composite affine transformations

- Another example



scale, then rotate



rotate, then scale

Rigid motions

- A transform made up of only translation and rotation is a *rigid motion* or a *rigid body transformation*
- The linear part is an orthonormal matrix

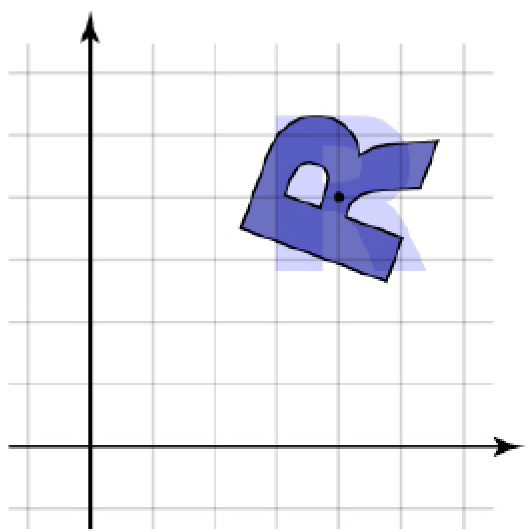
$$R = \begin{bmatrix} Q & \mathbf{u} \\ 0 & 1 \end{bmatrix}$$

- Inverse of orthonormal matrix is transpose
  - so inverse of rigid motion is easy:

$$R^{-1}R = \begin{bmatrix} Q^T & -Q^T\mathbf{u} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} Q & \mathbf{u} \\ 0 & 1 \end{bmatrix}$$

Composing to change axes

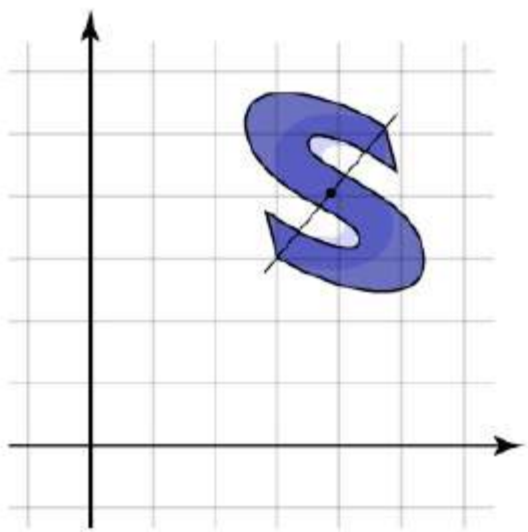
- Want to rotate about a particular point
  - could work out formulas directly…
- Know how to rotate about the origin
  - so translate that point to the origin



$$M = T^{-1}RT$$

Composing to change axes

- Want to scale along a particular axis and point
- Know how to scale along the y axis at the origin
  - so translate to the origin and rotate to align axes



$$M = T^{-1} R^{-1} S R T$$

Transforming points and vectors

- Note distinction between points and vectors
  - Vectors are offsets (differences between points)
  - Points have a location
    - Represented by vector offset from a fixed origin
- Points and vectors transform differently
  - Points respond to translation; vectors do not
  - Consider

$$v = p - q$$
$$T(x) = Mx + t$$

Transforming points and vectors

$$v = p - q$$
$$T(x) = Mx + t$$

- T is an affine transformation
- T is not a linear transformation

$$T(x + y) \neq T(x) + T(y)$$

$$\begin{aligned} T(p) - T(q) &= Mp + t - (Mq + t) \\ &= M(p - q) + (t - t) \\ &= Mv \end{aligned}$$

Transforming points and vectors

- Homogeneous coords. let us exclude translation
  - just put 0 rather than 1 in the last place

$$\begin{bmatrix} M & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} = \begin{bmatrix} M\mathbf{p} + \mathbf{t} \\ 1 \end{bmatrix} \quad \begin{bmatrix} M & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ 0 \end{bmatrix} = \begin{bmatrix} M\mathbf{v} \\ 0 \end{bmatrix}$$

  - and note that subtracting two points cancels the extra coordinate, resulting in a vector!
- Preview: projective transformations
  - what's really going on with this last coordinate?
  - think of R^2 embedded in R^3: all affine xfs. preserve z=1 plane
  - could have other transforms; project back to z=1

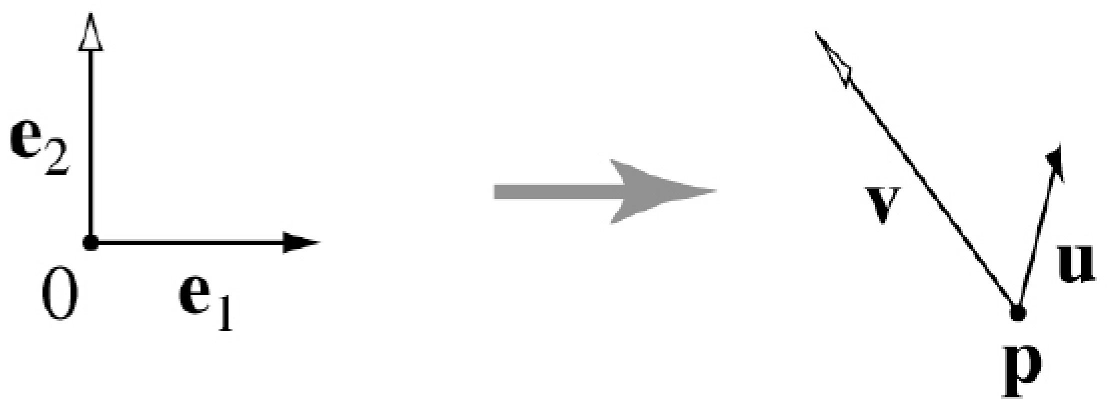
More math background

- Coordinate systems
  - Expressing vectors with respect to bases
  - Linear transformations as changes of basis

Affine change of coordinates

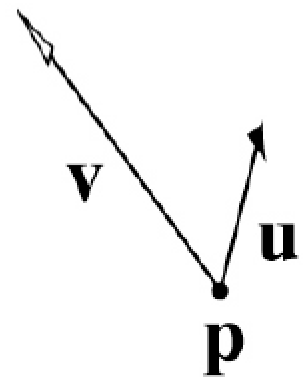
- Six degrees of freedom

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{p} \\ 0 & 0 & 1 \end{bmatrix}$$



Affine change of coordinates

- Coordinate frame: point plus basis
- Interpretation: transformation changes representation of point from one basis to another
- "Frame to canonical" matrix has frame in columns
  - takes points represented in frame
  - represents them in canonical basis
  - e.g. [0 0], [1 0], [0 1]

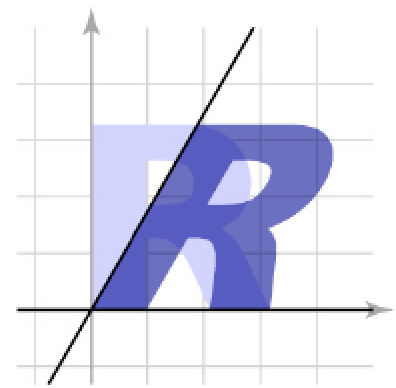


$$\begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{p} \\ 0 & 0 & 1 \end{bmatrix}$$

Affine change of coordinates

- A new way to "read off" the matrix
  - e.g. shear from earlier
  - can look at picture, see effect on basis vectors, write down matrix

$$\begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



- Also an easy way to construct transform:
  - e. g. scale by 2 across direction (1,2)

Affine change of coordinates

- When we move an object to the origin to apply a transformation, we are really changing coordinates
  - the transformation is easy to express in object's frame
  - so define it there and transform it

$$T_e = F T_F F^{-1}$$

  - T_e is the transformation expressed wrt. $\{e_1, e_2\}$
  - T_F is the transformation expressed in natural frame
  - F is the frame-to-canonical matrix $[u\ v\ p]$
- This is a *similarity transformation*

Coordinate frame summary

- Frame = point plus basis
- Frame matrix (frame-to-canonical) is

$$F = \begin{bmatrix} \mathbf{u} & \mathbf{v} & \mathbf{p} \\ 0 & 0 & 1 \end{bmatrix}$$

- Move points to and from frame by multiplying with *F*

$$p_e = F p_F \quad p_F = F^{-1} p_e$$

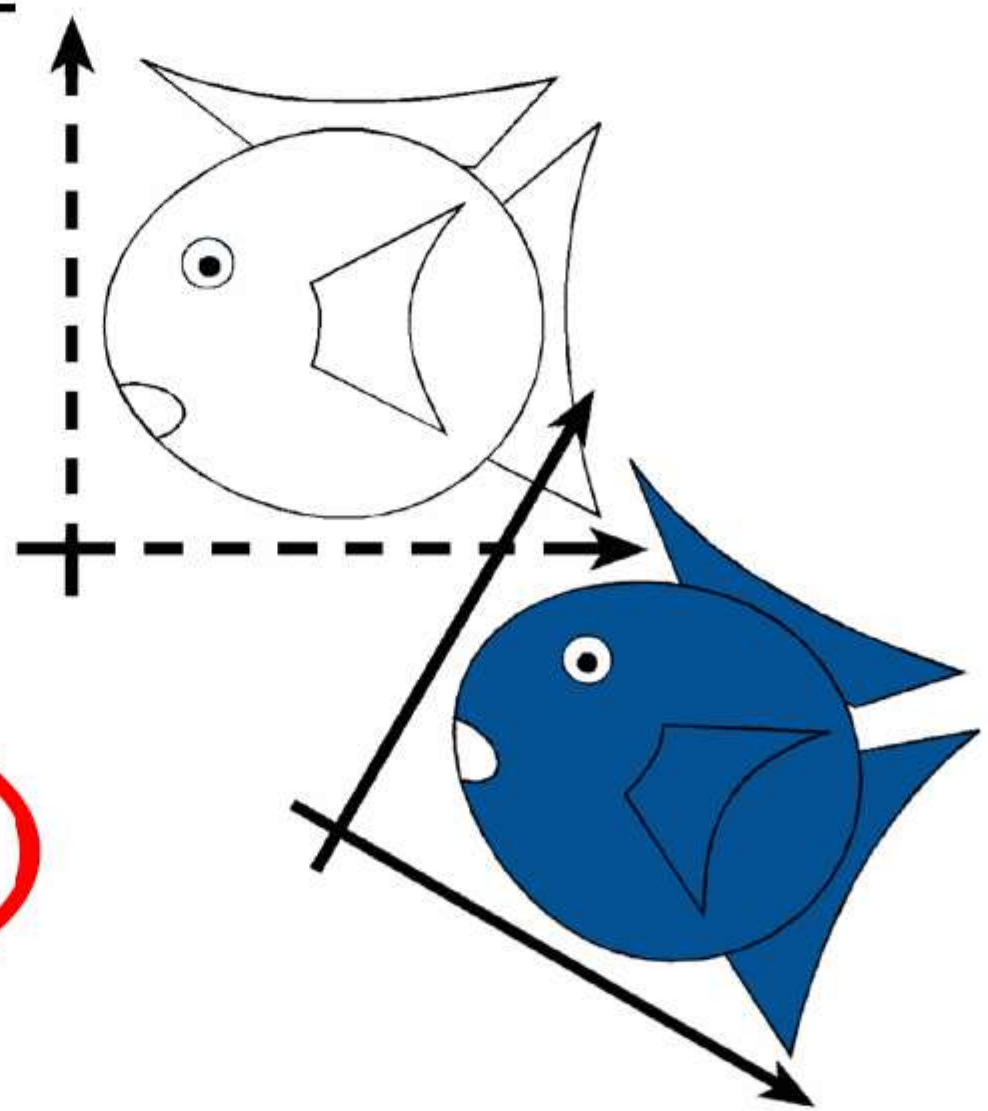
- Move transformations using similarity transforms

$$T_e = F T_F F^{-1} \quad T_F = F^{-1} T_e F$$

- Classes of Transformations
  - Rigid Body / Euclidean Transforms
  - Similitudes / Similarity Transforms
  - Linear
  - Affine
  - Projective

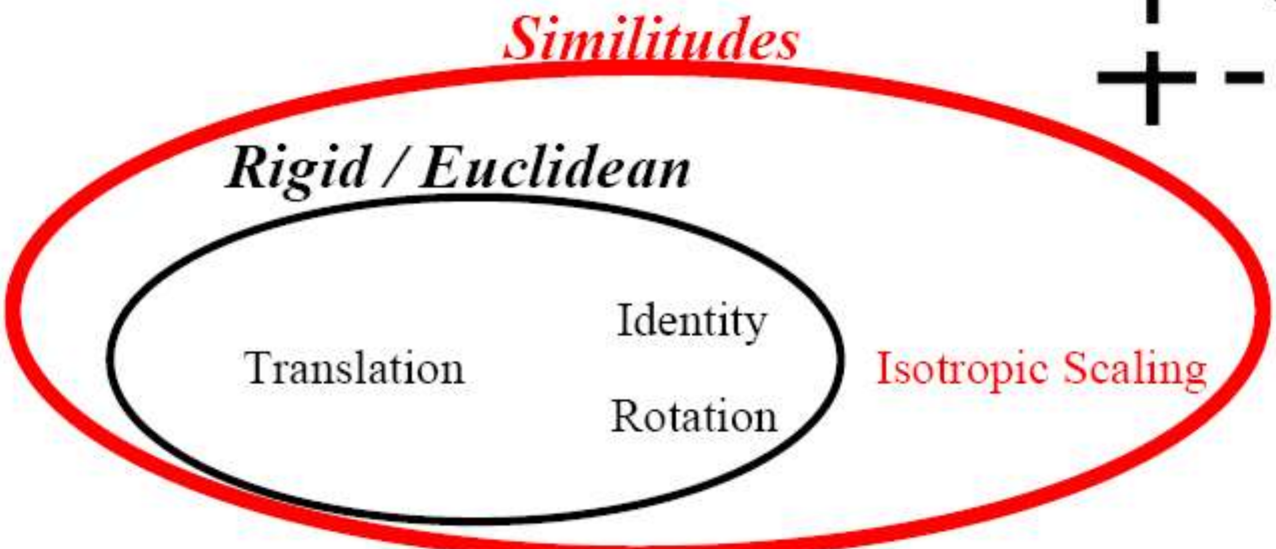
Rigid-Body / Euclidean Transforms

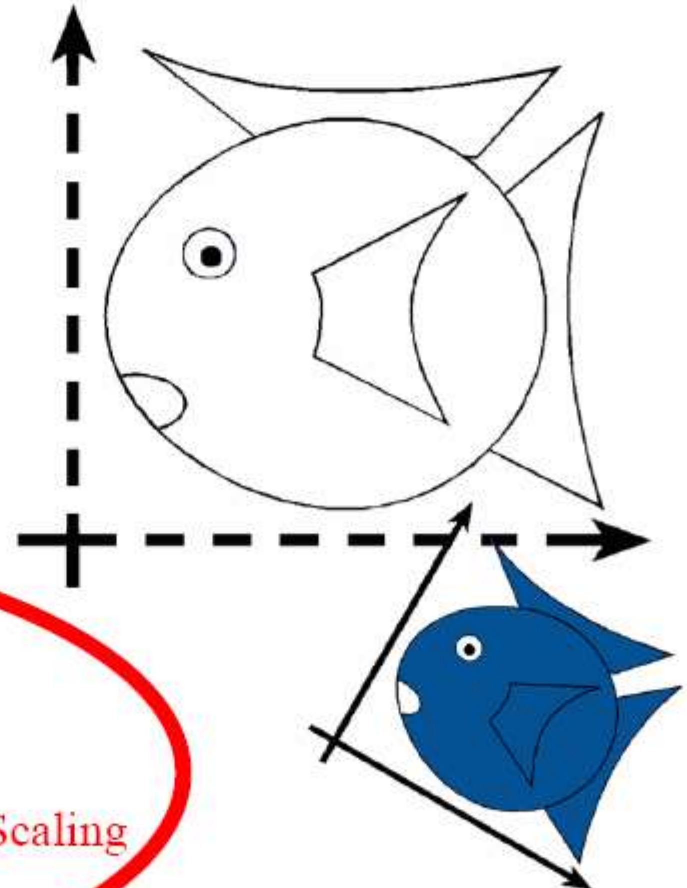
- Preserves distances
- Preserves angles



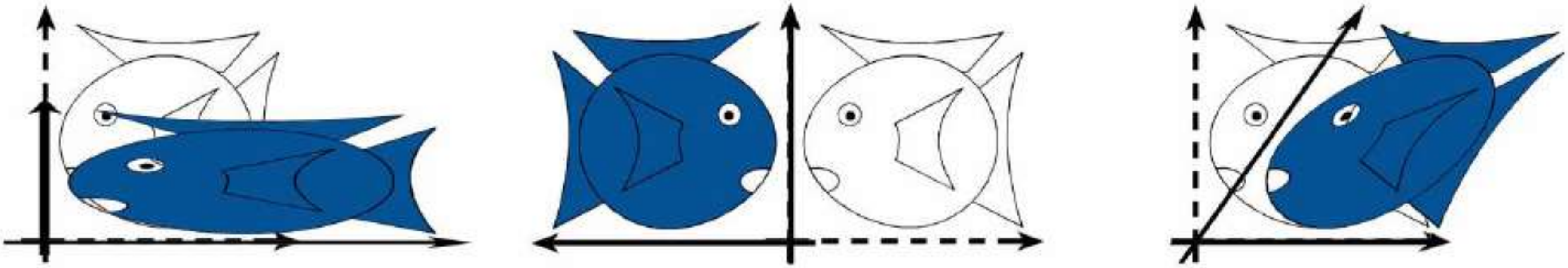
Similitudes / Similarity Transforms

- Preserves angles

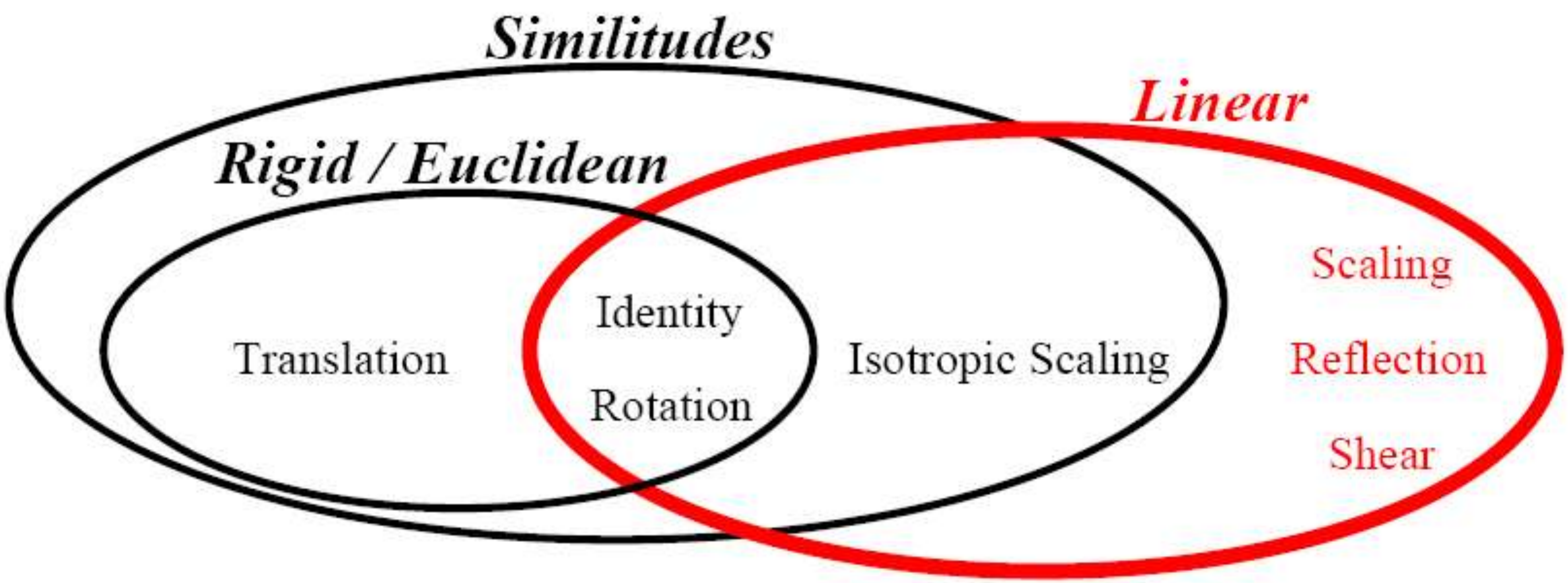




Linear Transformations

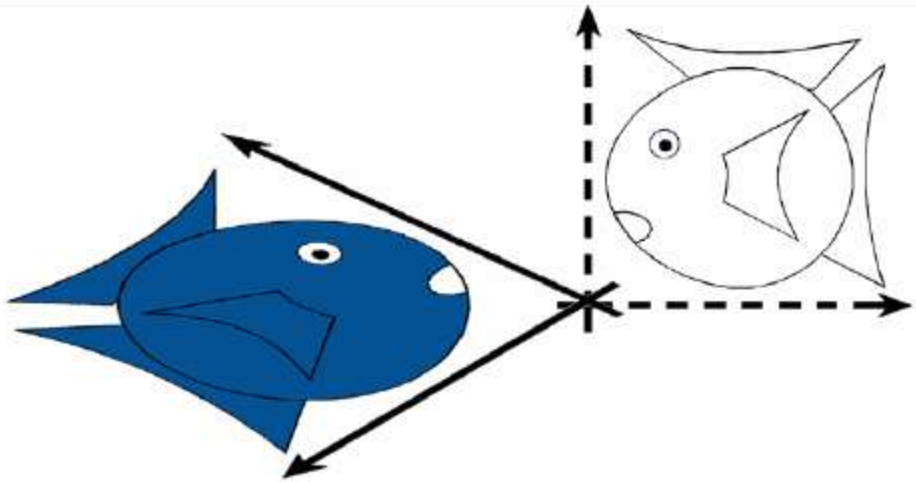


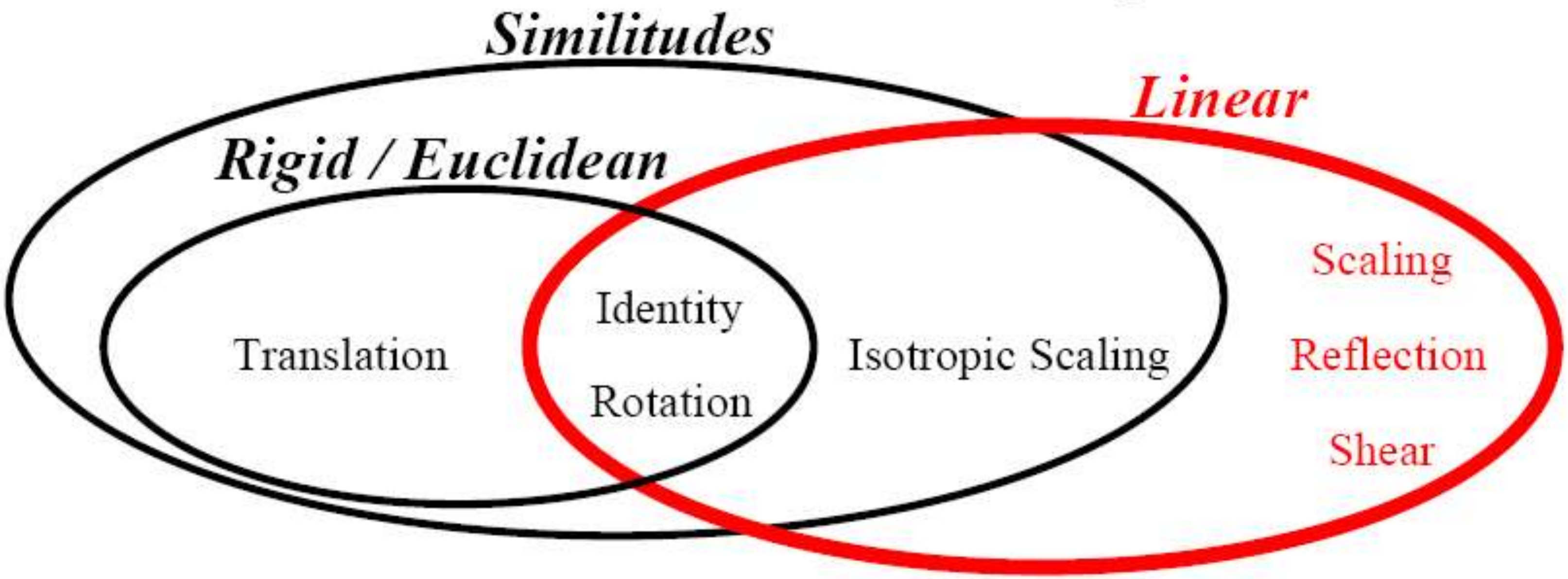
Scaling Reflection Shear



Linear Transformations

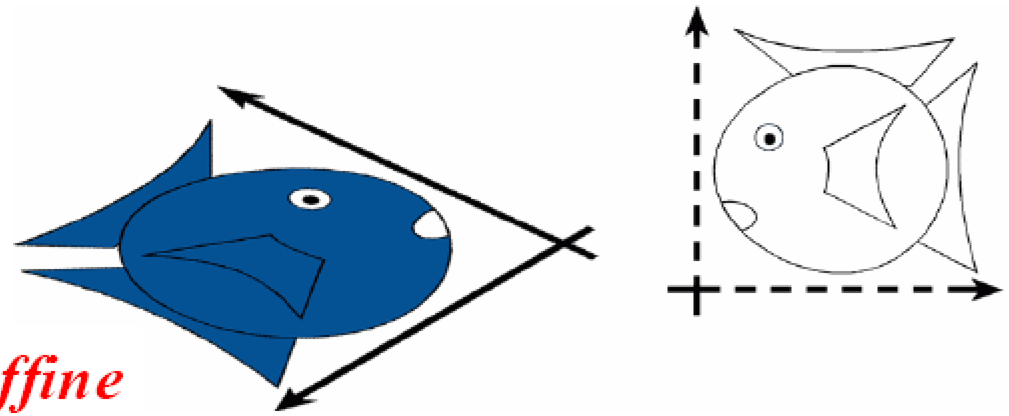
- $L(p + q) = L(p) + L(q)$
- $L(ap) = a\,L(p)$

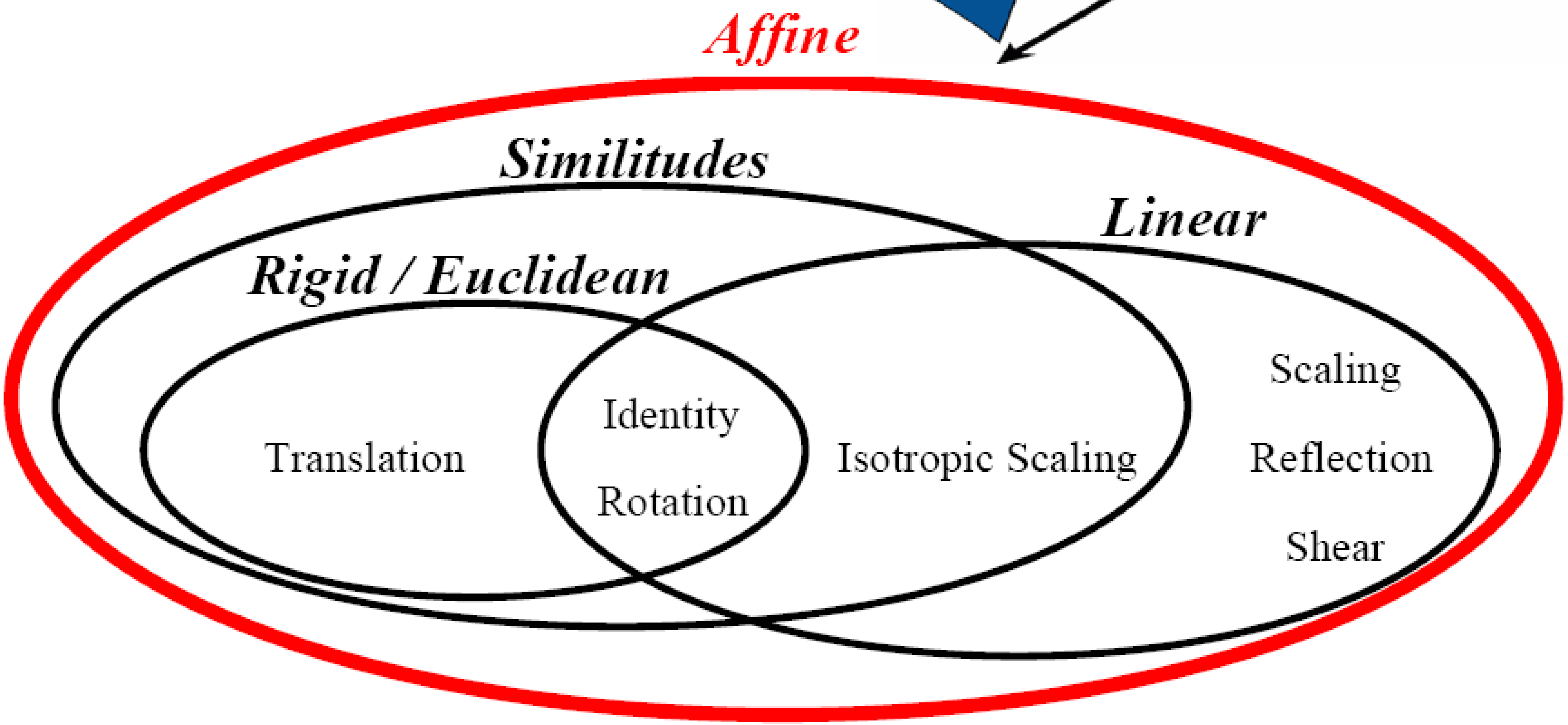




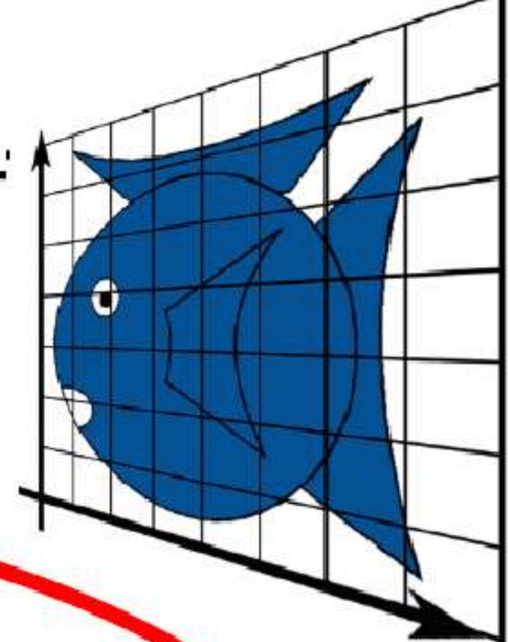
Affine Transformations

- preserves parallel lines

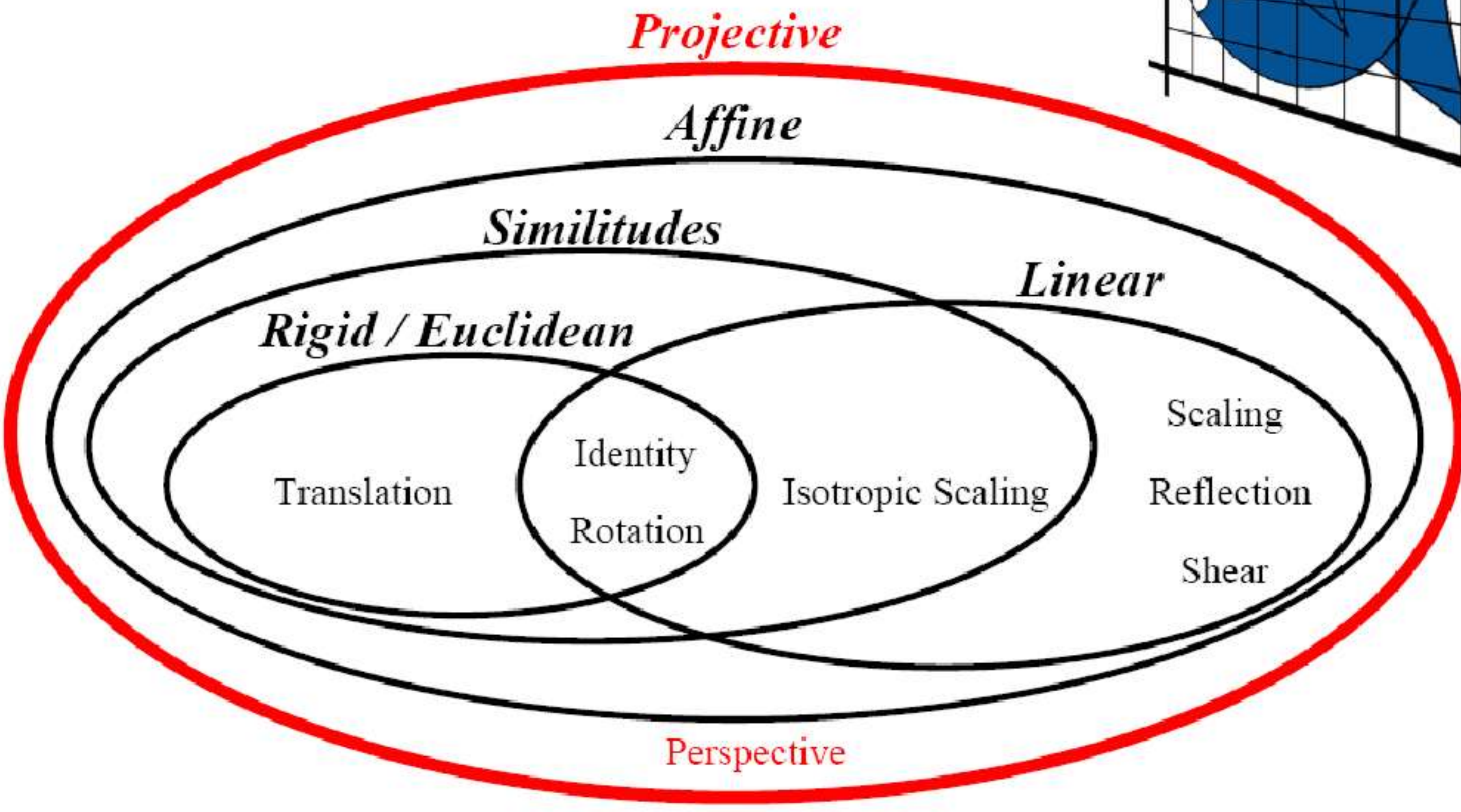




Projective Transformations



- preserves lines



Review and more information

- Review
  - Textbook Chapter 2 and Chapter 5
    - Miscellaneous math
    - Linear Algebra
- Textbook Chapter 5 - Transformation Matrices
  - 2D and 3D linear transformations
  - Translation and affine transformations
  - Inverses of transformation matrices
  - Coordinate Transformations