Fundamentals of Computer Graphics

COMP 557

3 September 2020

Paul Kry

About Me

Paul G. Kry
Associate Professor

Contact

School of Computer Science, McGill University
3480 University Street
McConnell Building, Room 318
Montréal, QC, H3A 0E9
Canada

office: MC423
phone: +1 514 398 2577
email: kry@cs.mcgill.ca

Teaching

Courses for Fall 2020 and Winter 2021

- COMP 559 Winter 2021, Fundamentals of Computer Animation
- COMP 273 Fall 2020, Introduction to Computer Systems
- COMP 557 Fall 2020, Fundamentals of Computer Graphics

+ Previous Courses

Research Interests and Information

My research interests include computer graphics, physically based animation, skin deformations of articulated characters, motion capture, interaction, and physically based modeling of humans and animals. I am specifically interested in human and animal motor control (e.g., locomotion, grasping, manipulation) in combination with natural phenomena such as the physics of rigid objects, deformation, and contact. Example application areas include computer animation for video games and movies, training simulations, ergonomics, and biologically inspired robotics and programming by demonstration. An important aspect of my work is the combination of real world measurements, approximate models, and physically based simulation. I am also interested in machine learning, numerical methods, and audio.

+ Prospective Students

+ Current and Previous Students and Lab Members

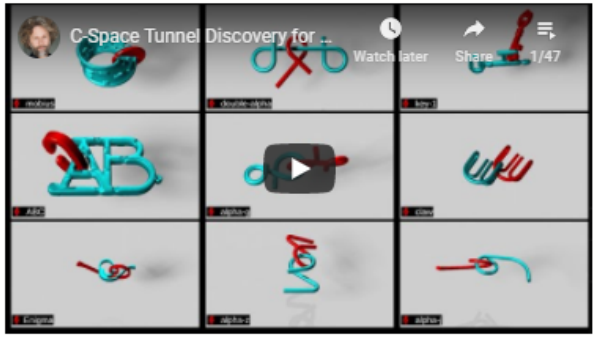
+ Computer Animation and Interaction Capture Lab

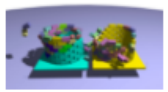
+ Workshops

+ Program Committees and Service

+ Biographical Information

McGill University Computer Animation Research YouTube Channel



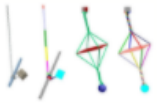


Adaptive Merging for Rigid Body Simulation
E Coevoet, O. Benchekroun, PG Kry *SIGGRAPH, 2020*

We reduce computation time in rigid body simulations by merging collections of bodies when they share a common spatial velocity. Merging relies on monitoring the state of contacts, and a metric that compares the relative linear and angular motion of bodies based on their sizes. Unmerging relies on an inexpensive single iteration projected Gauss-Seidel sweep over contacts between merged bodies, which lets us update internal contact forces over time, and use the same metrics as merging to identify when bodies should unmerge. Furthermore we use a contact ordering for graph traversal refinement of the internal contact forces in collections, which helps to correctly identify all the bodies that must unmerge when there are impacts. The general concept of merging is similar to the common technique of sleeping and waking rigid bodies in the inertial frame, and we exploit this too, but our merging \fix{is in moving frames}, and unmerging takes place at contacts between bodies rather than at the level of bodies themselves. We discuss the previous relative motion metrics in comparison to ours, and evaluate our method on a variety of soenarios.

C-Space Tunnel Discovery for Puzzle Path Planning
X Zhang, R. Belfer, PG Kry, E. Vouga *SIGGRAPH, 2020*

Rigid body disentanglement puzzles are challenging for both humans and motion planning algorithms because their solutions involve tricky twisting and sliding moves that correspond to navigating through narrow tunnels in the puzzle's configuration space (C-space). We propose a tunnel-discovery and planning strategy for solving these puzzles. First, we locate important features on the pieces using geometric heuristics and machine learning, and then match pairs of these features to discover collision free states in the puzzle's C-space that lie within the narrow tunnels. Second, we propose a Rapidly-exploring Dense Tree (RDT) motion planner variant that builds tunnel escape roadmaps and then connects these roadmaps into a solution path connecting start and goal states. We evaluate our approach on a variety of challenging disentanglement puzzles and provide extensive baseline comparisons with other motion planning techniques.



Schur complement-based substructuring of stiff multibody systems with contact
A Peiret, S Andrews, J Kövecses, PG Kry, M Teichmann *SIGGRAPH Asia, 2019*

We investigate a procedural shape modeling approach based on reaction-diffusion equations and physically based growth of thin shells. This inspiration of this work comes from the morphological development of living tissues, such as plants leaves. There are numerous choices that can be made in assembling a computer simulation of these growth system. We explore two main approaches, one where a reaction-diffusion simulation is first run with the results used to identify regions of growth, and the other where we simulate shell growth concurrently with a reaction-diffusion simulation in the manifold. We demonstrate that a variety of interesting shapes can be grown in this manner, and provide some intuition to the challenging problem of associating changes in parameter settings with the final shape.

PAPER
VIDEO
DOI

The Matchstick Model for Anisotropic Friction Cones
K. Erleben, M. Macklin, S. Andrews, P. G. Kry *Computer Graphics Forum, 2019*

Inspired by frictional behaviour that is observed when sliding matchsticks against one another at different angles, we propose a phenomenological anisotropic friction model for structured surfaces. Our model interpolates isotropic and anisotropic elliptical Coulomb friction parameters for a pair of surfaces with perpendicular and parallel structure directions (e.g., the wood grain direction). We view our model as a special case of an abstract friction model that produces a cone based on state information, specifically the relationship between structure directions. We show how our model can be integrated into LCP and NCP based simulators using different solvers with both explicit and fully implicit time-integration. The focus of our work is on symmetric friction cones, and we therefore demonstrate a

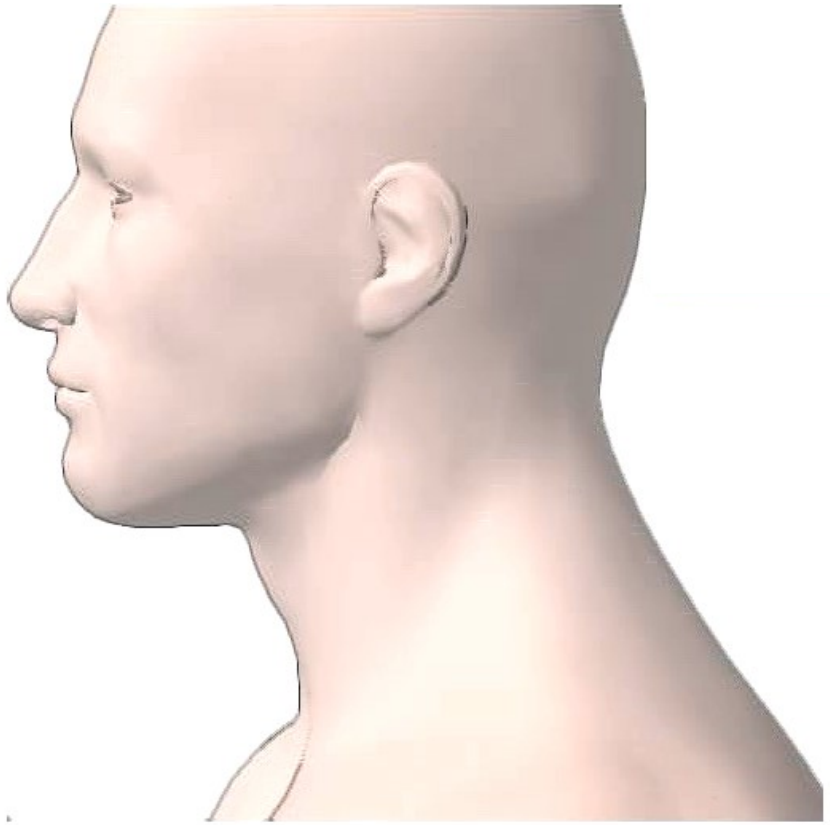
“Computers are useless. They can only give you answers”

Pablo Picasso

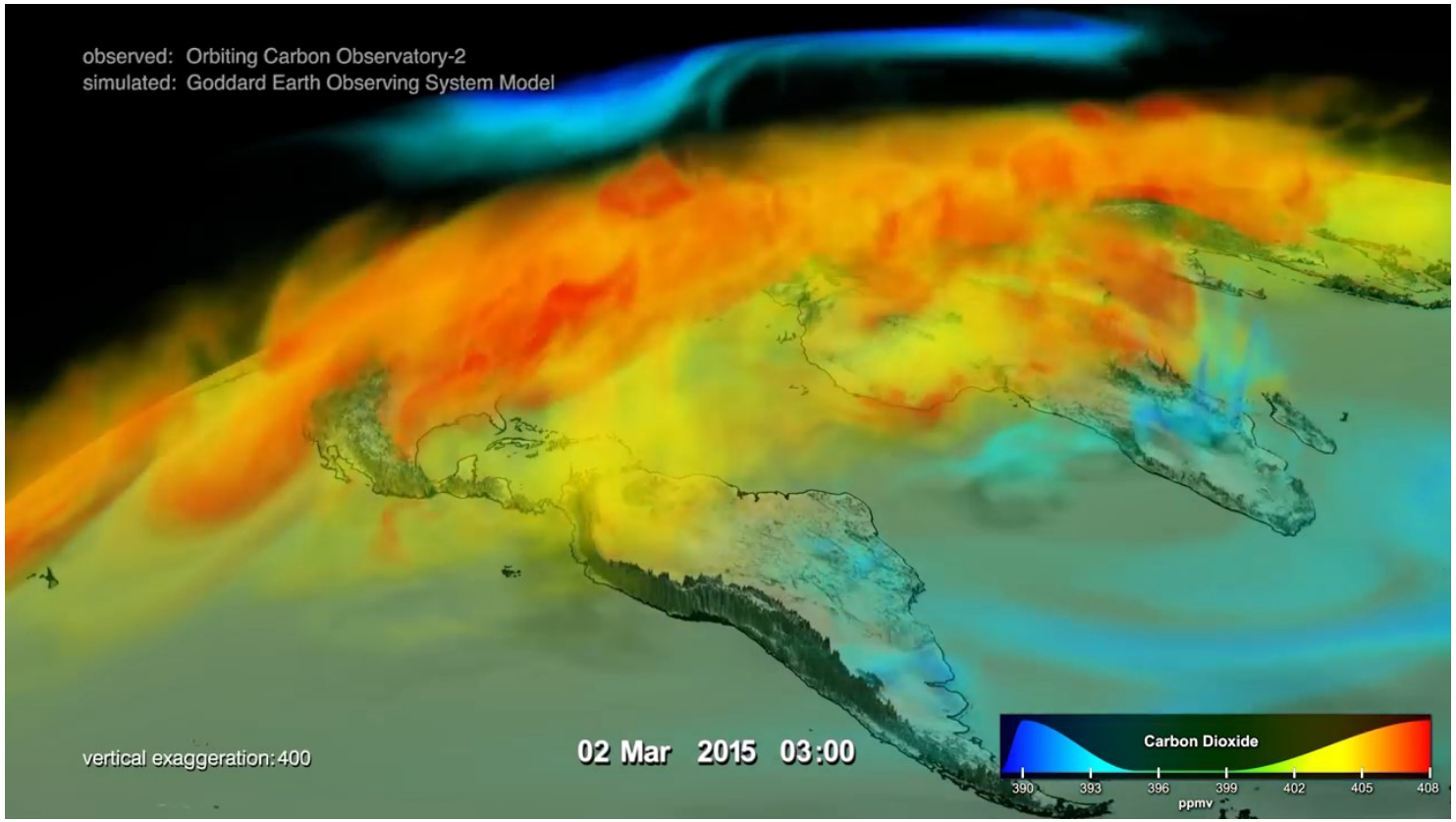
Art

Meats Meier

Visualization



Interactive Cutaway Illustrations of Complex 3D Models [Li et al., SIGGRAPH 2007]

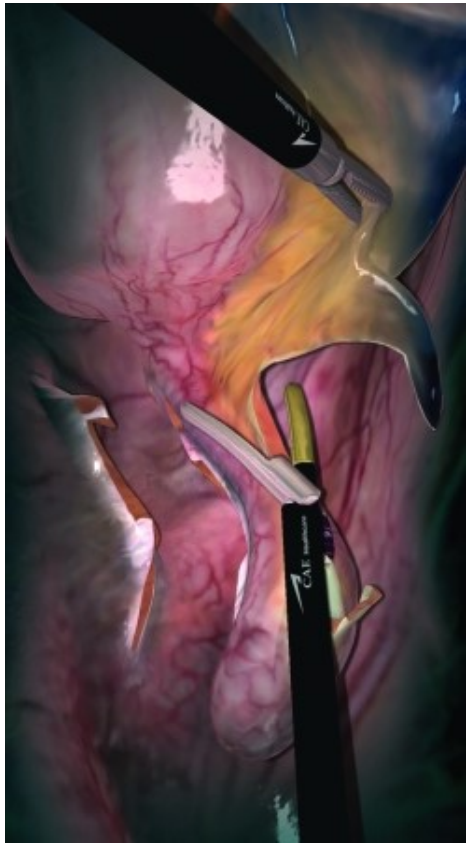


From NASA Goddard via the OCO-2 Satellite

Training

Flight simulation

CM Labs, crane simulator



Appendectomy simulation
(CAE LapVR)


Games

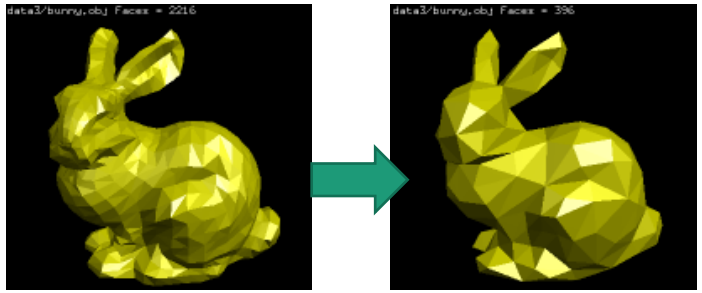
Movies

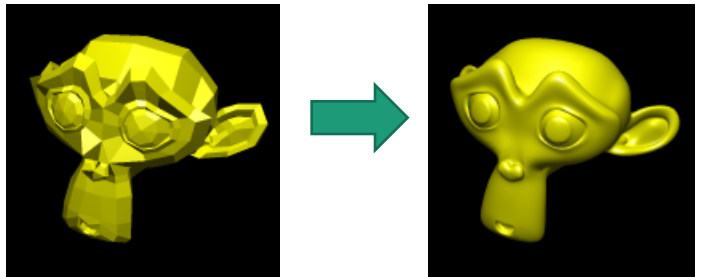
[R&H Demo Reel]

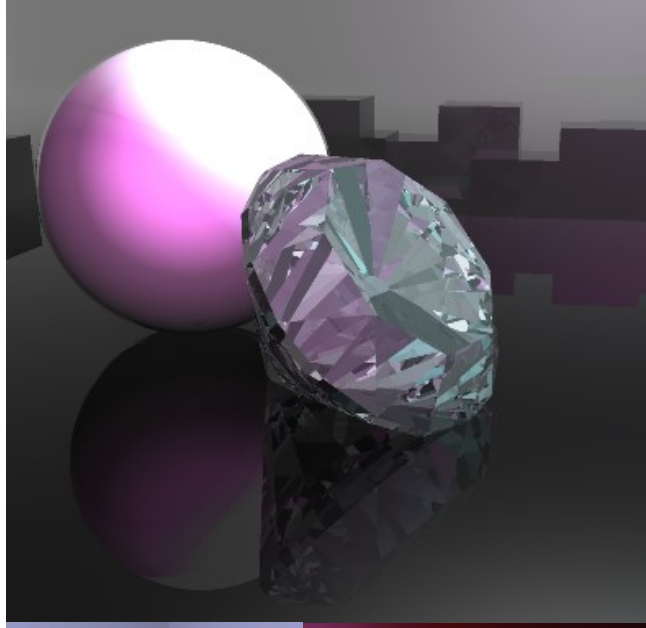
What is Computer Graphics?

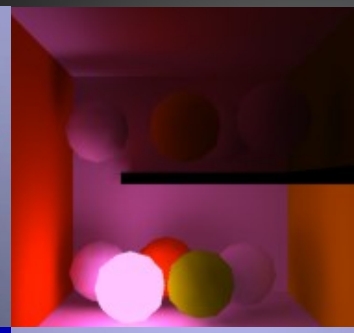
- A vast field that encompasses pretty much anything related to computer generated images
  - Modeling, rendering, animation, image processing, visualization, interactive techniques, etc.

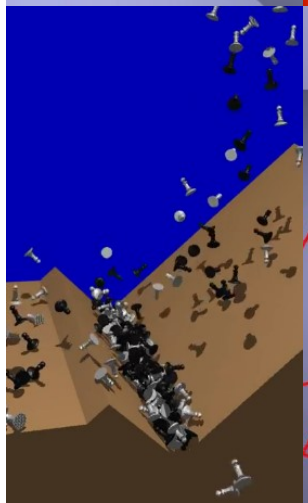
[example assignments and projects from 557 and 559]

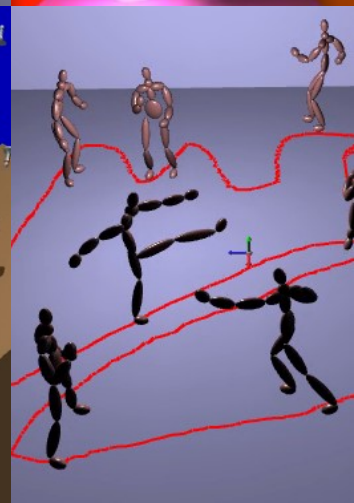












Computers
Accept, process, transform, and present information

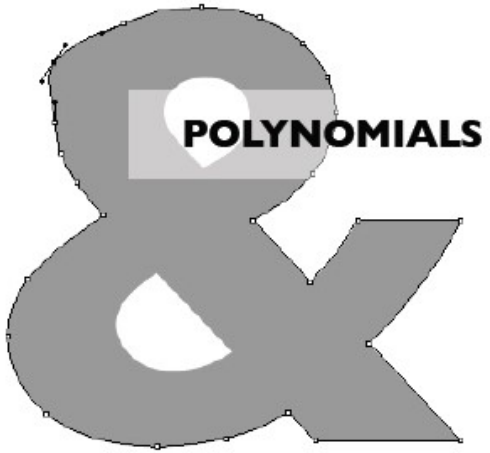
Computer graphics
Accept, process, transform, and present information, in a visual form

Computer Graphics

Mathematics made visible

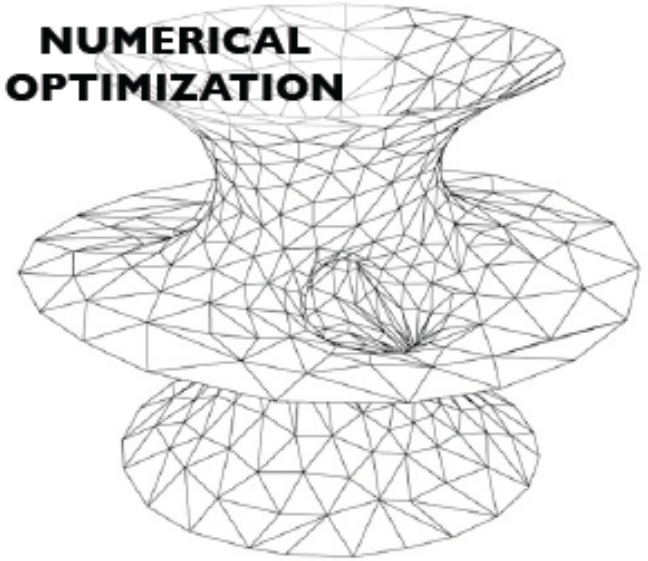
Problems in Computer Graphics

- 2D imaging
  - Compositing and layering
  - Digital filtering
  - Colour transformations
- 2D drawing
  - Illustration, drafting
  - Text
  - GUIs

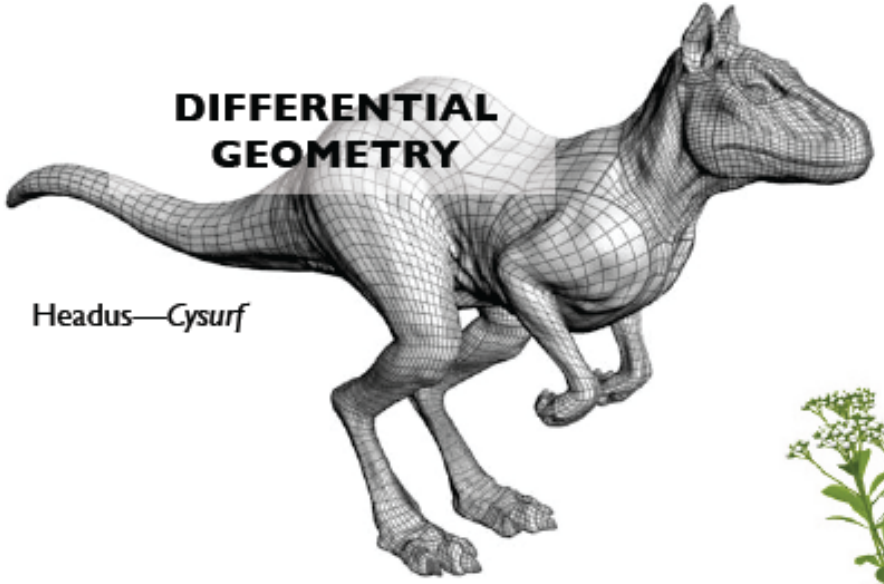


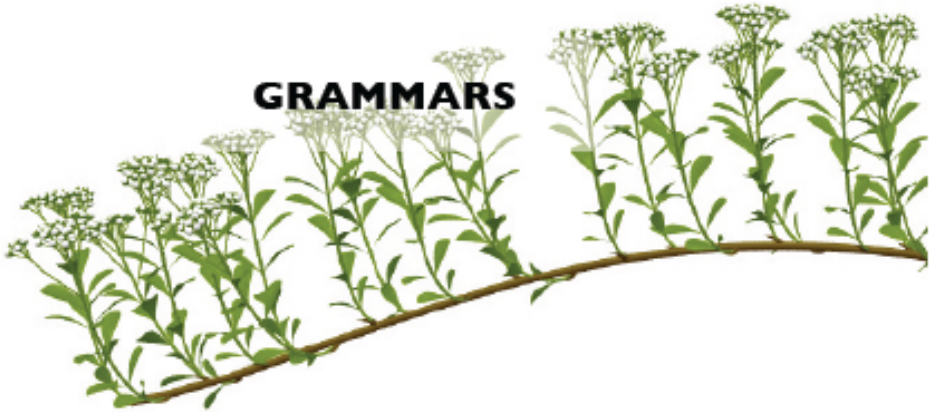
Problems in Computer Graphics

- 3D Modeling
  - Representing 3D shapes
  - Polygons, curved surfaces, …
  - Procedural modeling



[Hoppe et al. 1993]

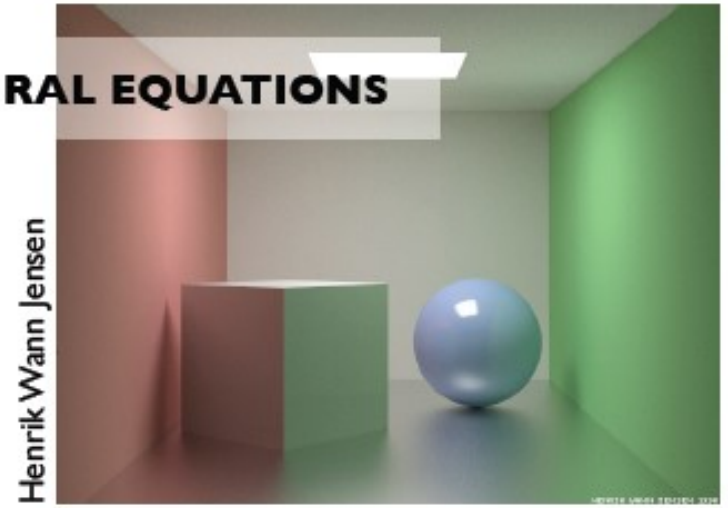




[Prusinkeiwicz et al. 2001]

Problems in Computer Graphics

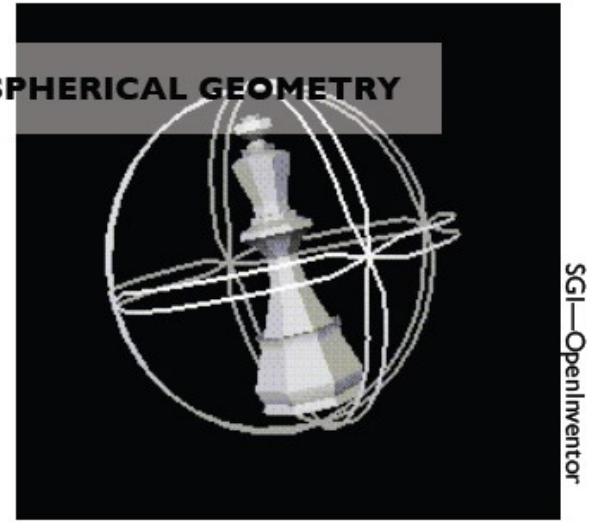
- 3D Rendering
  - 2D views of 3D geometry
  - Projection and perspective
  - Removing hidden surfaces
  - Lighting simulation



Problems in Computer Graphics

- User Interaction
  - 2D graphical user interfaces
  - 3D modeling interfaces
  - New interfaces in VR and AR

TU Berlin

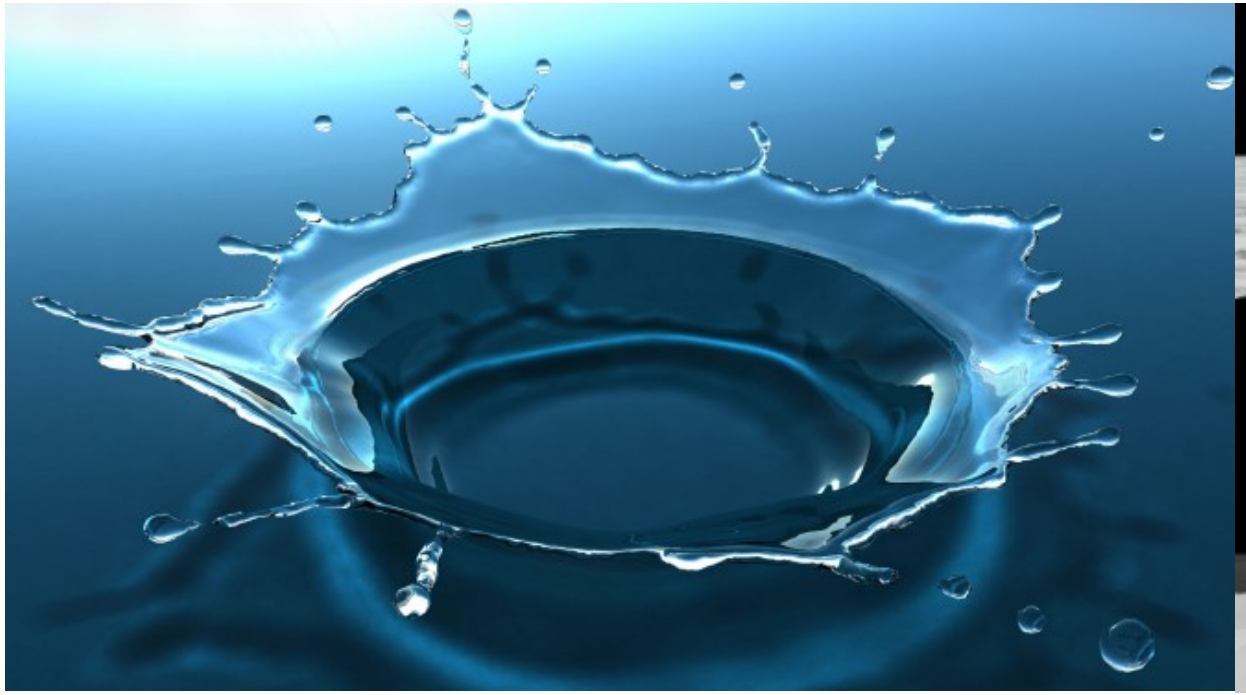


SGI—OpenInventor

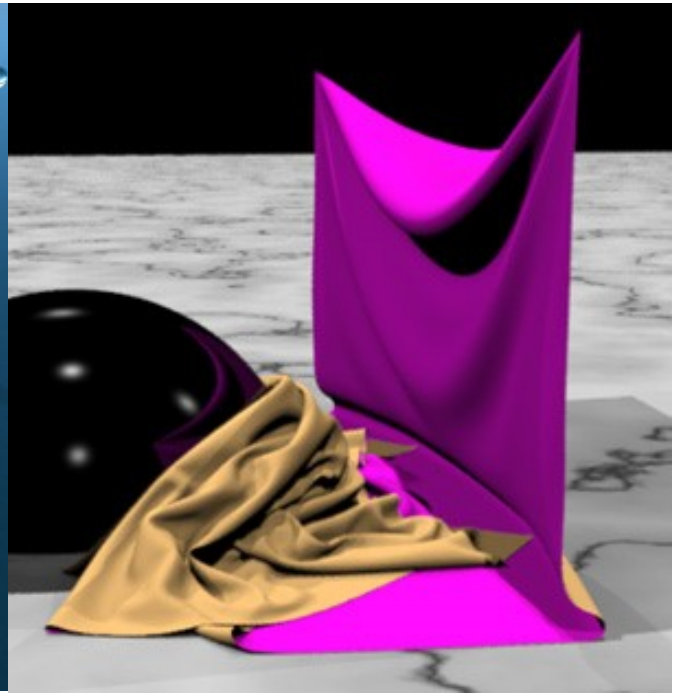
Oculus Insight

Problems in Computer Graphics

- Animation
  - Keyframe animation
  - Physical simulation



[Thürey et al. 2010]



[Bridson et al. 2002]

Evolution of computing environments

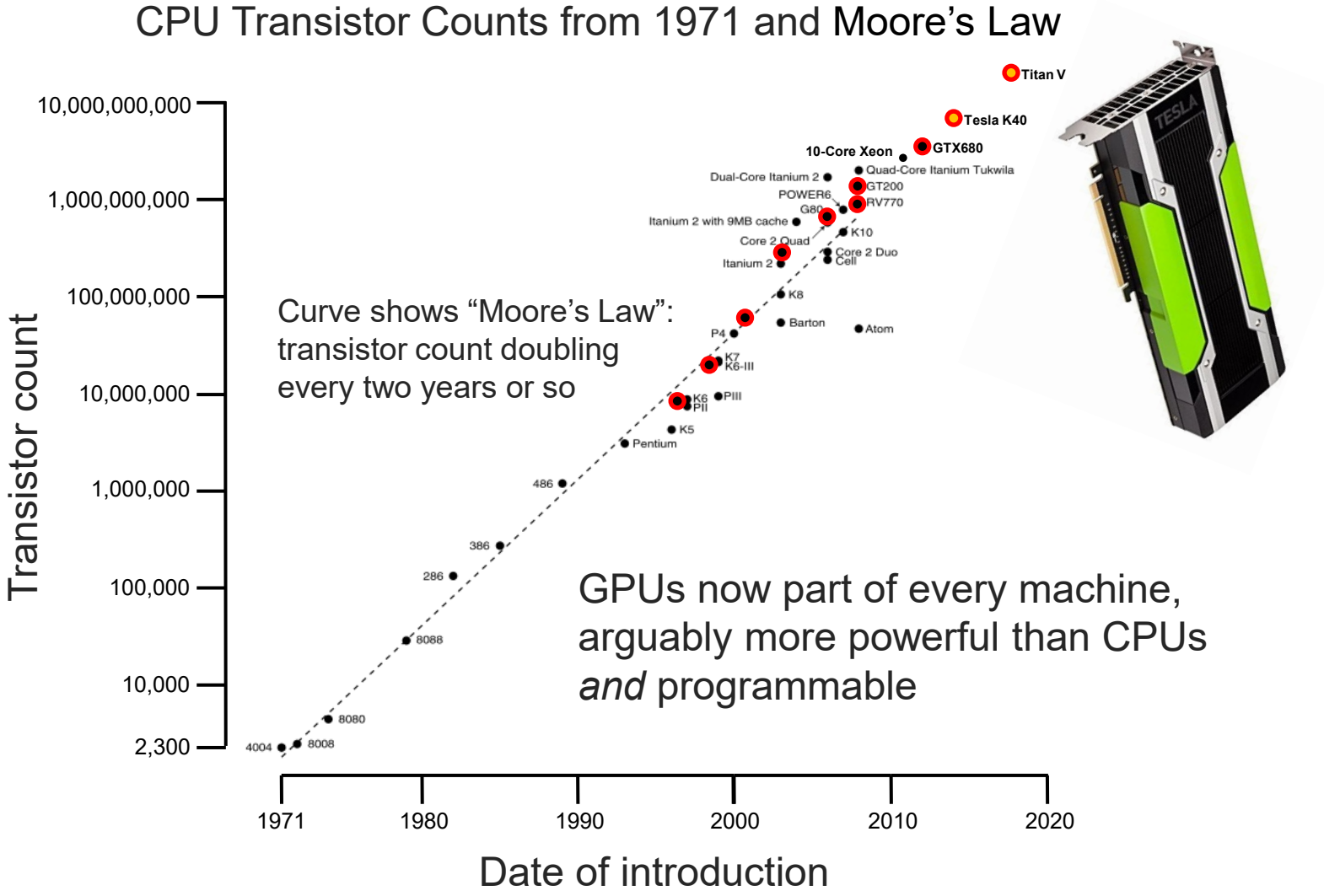
- Graphics has been a key to technology growth
  - Graphical user interfaces
  - Desktop publishing
  - Visualization
  - Gaming consoles

- Hardware revolution drives everything
  - price/performance improving exponentially (Moore's Law)
  - Graphics processors on even faster exponentials

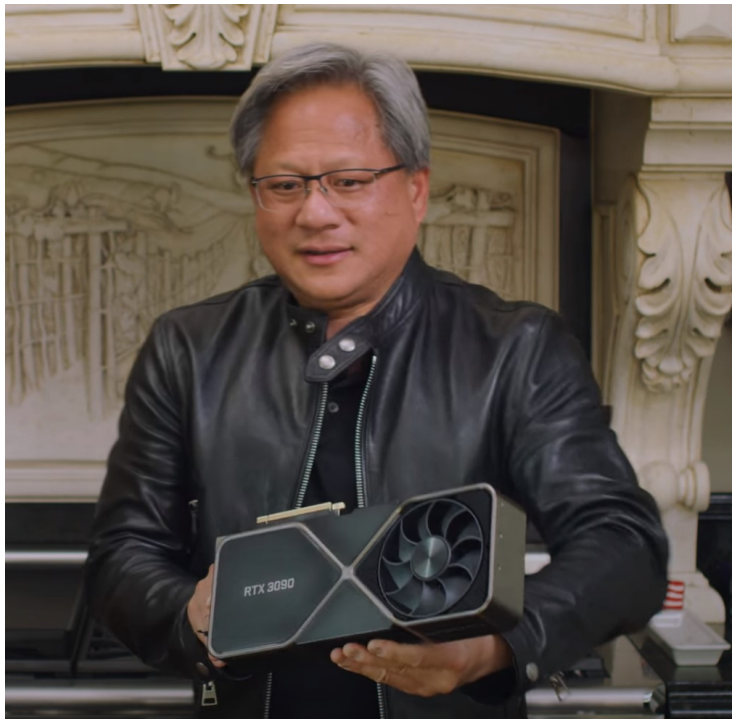
	Original Macintosh	New iMac 27”	
Date	1984	2020	+34
Price	$2500	$2400	x 1.0
CPU	8 MHz	3.1 GHz (6-core)	x 387 (x 2325)
Memory	128KB RAM	8.0GB DDR3 SDRAM (+4 GB on GPU)	x 65536
Storage	400KB Floppy	1TB Hard Disk	x 2500000
Monitor	9” Black&White 512 x 342 68 dpi	27” Color 5120 x 2880 218 dpi	x 3 x 84 x 3.2
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



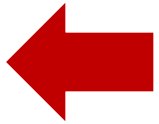
GPUs now part of every machine, arguably more powerful than CPUs *and* programmable



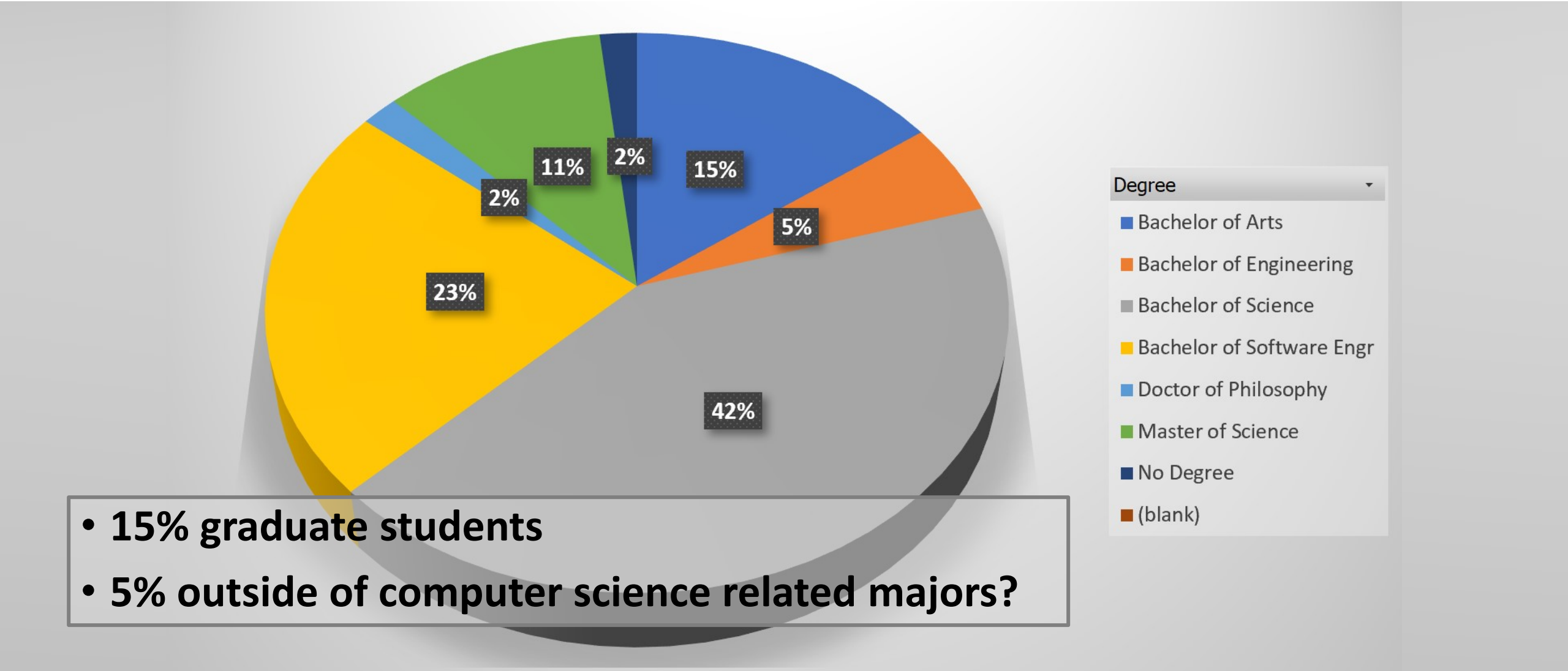
RTX 3090, Fresh out of Jensen's oven
28G Transistors
10496 Cuda Cores!
https://youtu.be/E98hC9e__Xs?t=2088

Today

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



About you…



- **15% graduate students**
- **5% outside of computer science related majors?**

Comp 557

- You will:
  - Explore fundamental ideas
  - Learn math essential to graphics
  - Implement key algorithms
  - Write cool programs
- You will not:
  - Learn Photoshop, Blender, AutoCad, Maya, Renderman, etc.
  - Become experts at OpenGL or DirectX
    (but you will become comfortable with OpenGL)
  - Write huge programs

Prerequisites
(COMP206, COMP250, MATH 222, MATH 223)

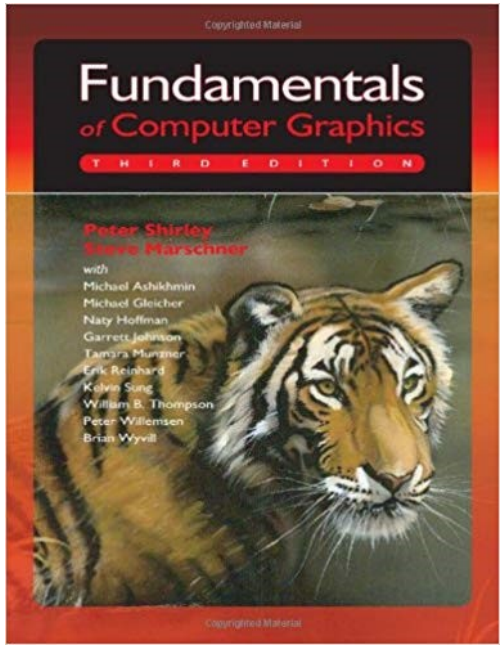
- Programming
  - ability to read (understand), write, and debug small Java programs (10s of classes)
  - understanding of basic data structures
  - no serious software design required
- Mathematics
  - vector geometry (dot/cross products, etc.)
  - linear algebra (matrices in 2D to 4D, linear systems)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

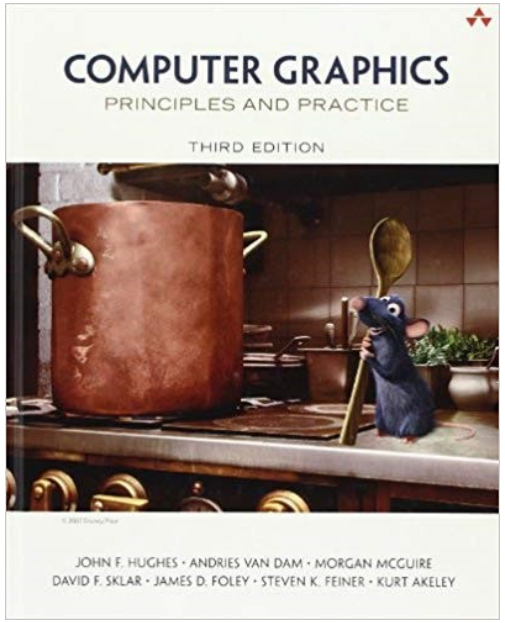
Topics

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

Textbook and Resources

- Recommended books
  - Fundamentals of Computer Graphics, 3rd ed., Peter Shirley and Steve Marschner
  - Computer Graphics Principles and Practice, 3rd ed., Hughes et al.
- Other good resources
  - OpenGL Programming Guide: The official guide to learning OpenGL (4.0 and up)
  - The Graphics Codex (mostly rendering focused)
  - Additional material will be posted to MyCourses throughout the term





Textbook and Resources

- FCG Chapter 2 and 5 Math Review (if you feel rusty)
- FCG Chapter 6 Transforms in 2D and 3D
- FCG Chapter 7 Viewing and Projection

- CGPP Chapter 10 Transforms in 2D
- CGPP Chapter 11 Transforms in 3D
- CGPP Chapter 13 Viewing and Projection

- OpenGL Programming Guide
  - Chapter 1, Introduction to OpenGL, useful to read to get up to speed
    Watch out! Older editions show “easy” immediate mode calls, while and more recent versions use *buffers* and *drawArray* and *drawElements* calls

Evaluation

- Assignments 50%
  - There will be 4, not all equal weight (10,10,15,15), and different amounts of time for completion
  - ***Do your own work***
  - ***Must be correctly submitted to MyCourses***
- Midterm 20%
  - In October, date to be set
- 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.

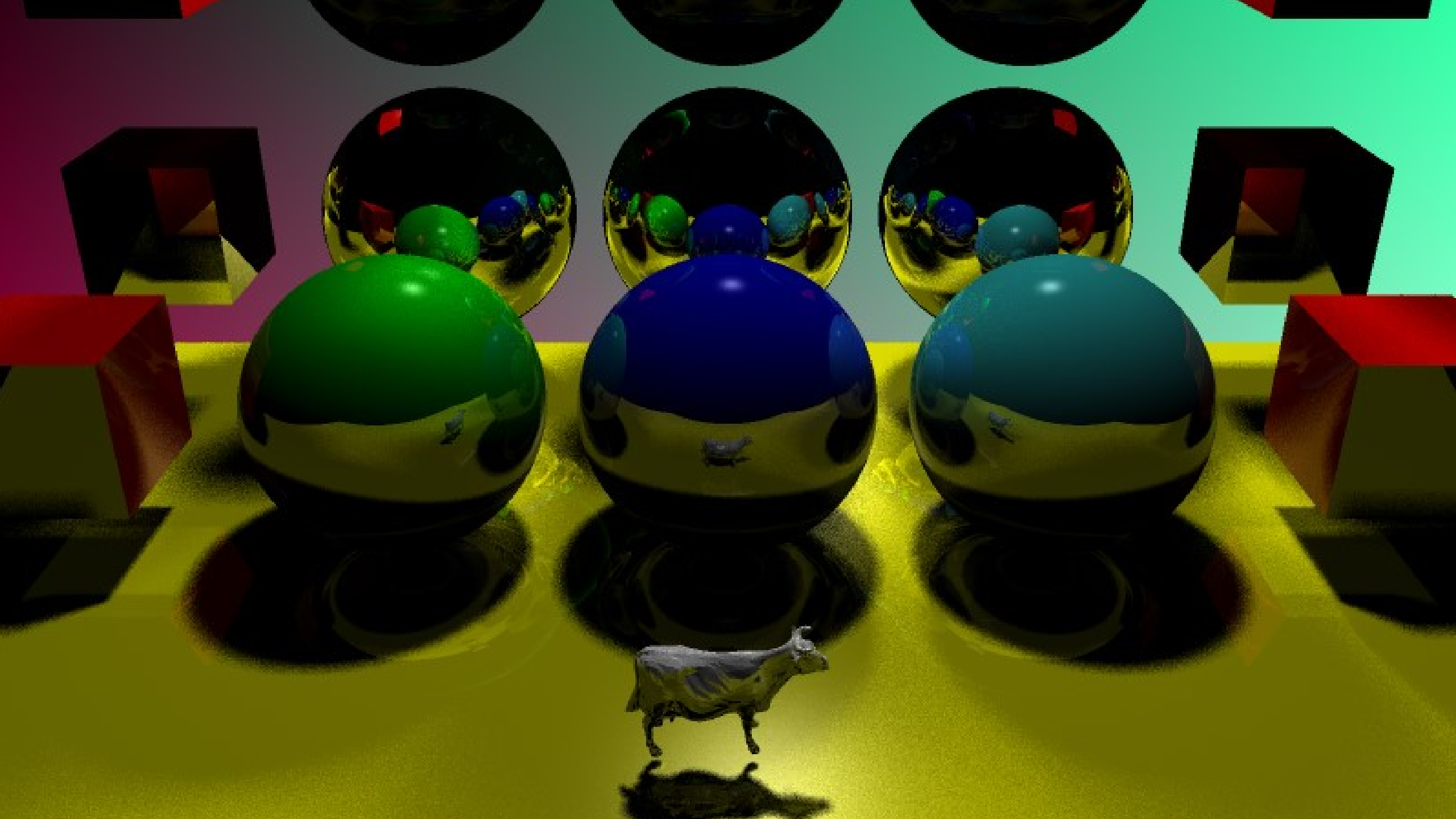
Be sure your name and student number is in the window title, and in the top comments section of each of your source files. Submit your source code as a zip file via MyCourses. If you have comments about your solution, include a readme.txt or readme.pdf file with your comments. Be sure to check that your submission is correct by downloading your submission from MyCourses. You can not receive any marks for assignments with missing or corrupt files! Note that you are encouraged to discuss assignments with your classmates, but not to the point of sharing code and answers. All code and written answers must be your own.

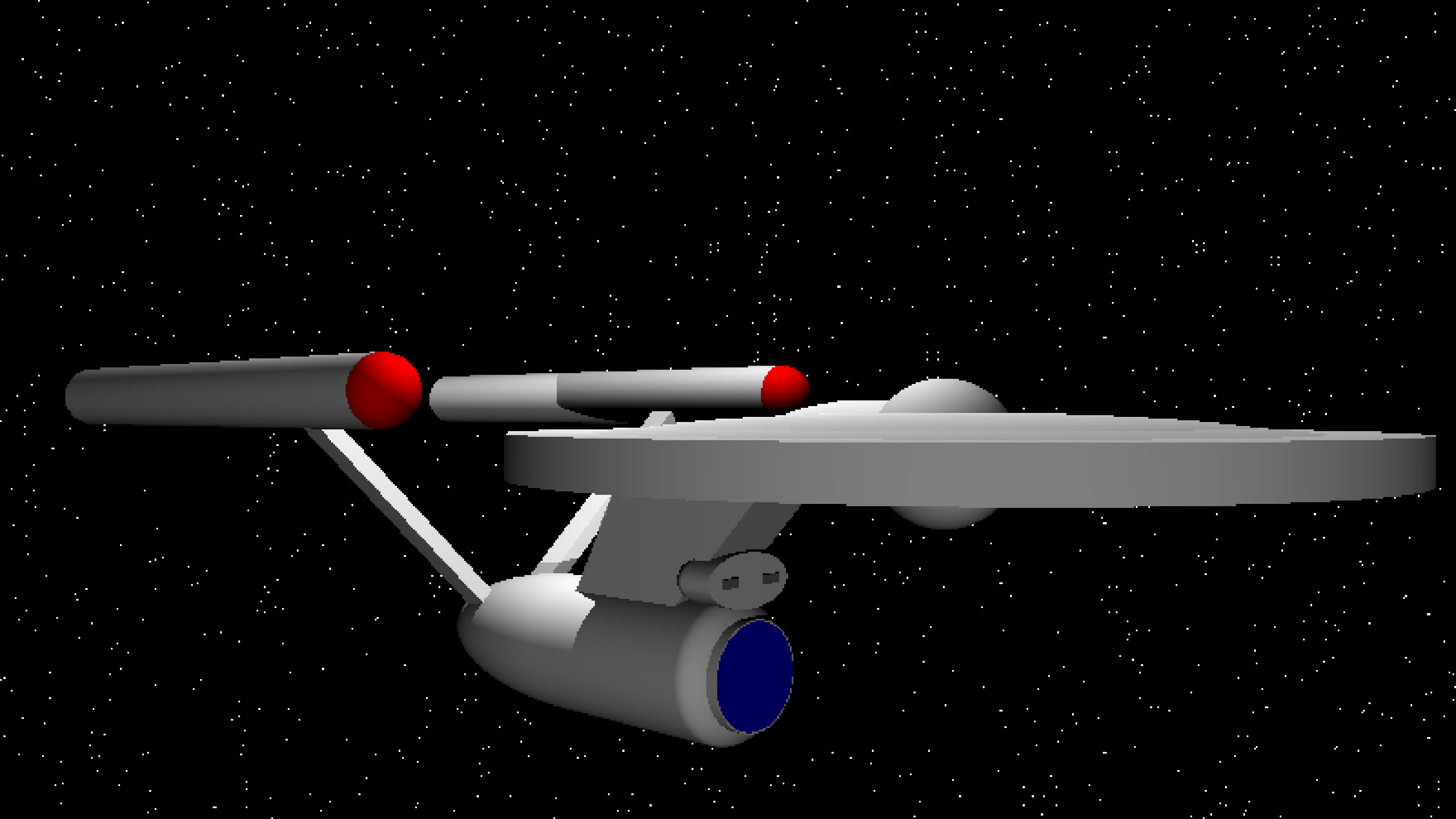
Do not share your code. Using version control is a good idea but using a publicly accessible repository is not acceptable.

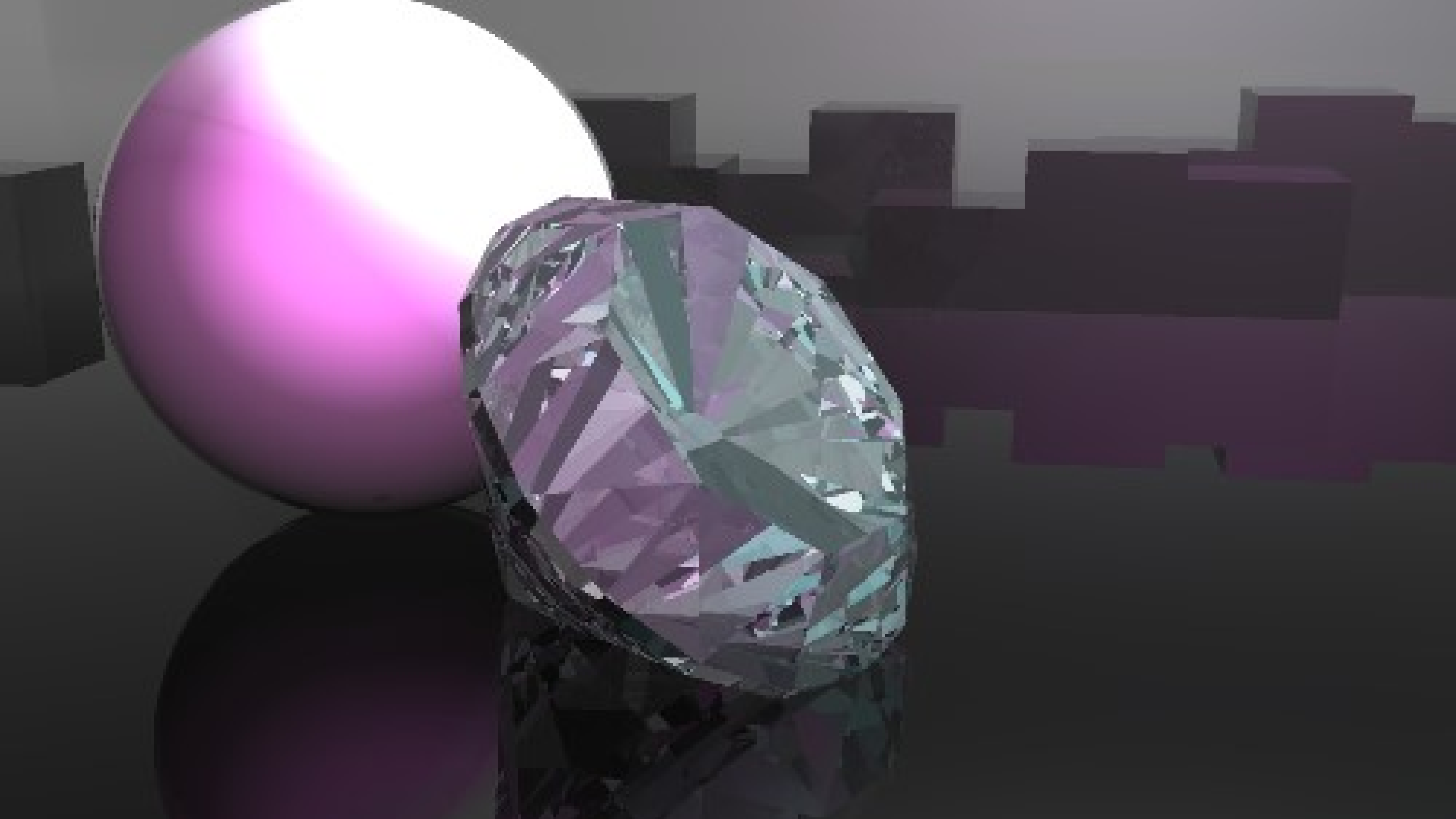
ALL WORK MUST BE SUBMITTED ON MY COURSES.
DO NOT EMAIL YOUR WORK TO THE PROF OR TAs. ALL DEADLINES ARE HARD!!!
DO NOT WAIT TO THE LAST MINUTE TO SUBMIT YOUR WORK!

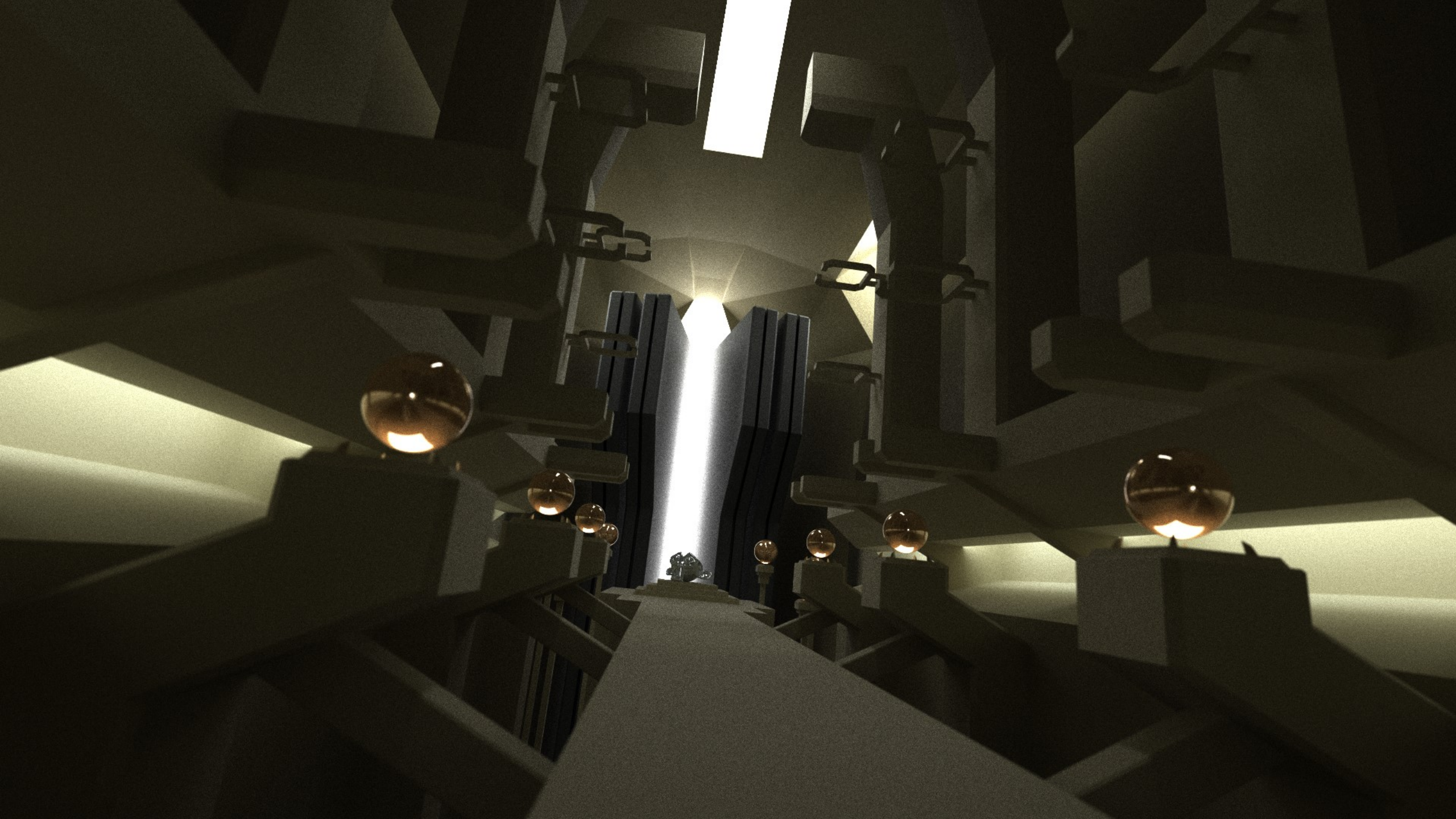
Assignments (four)

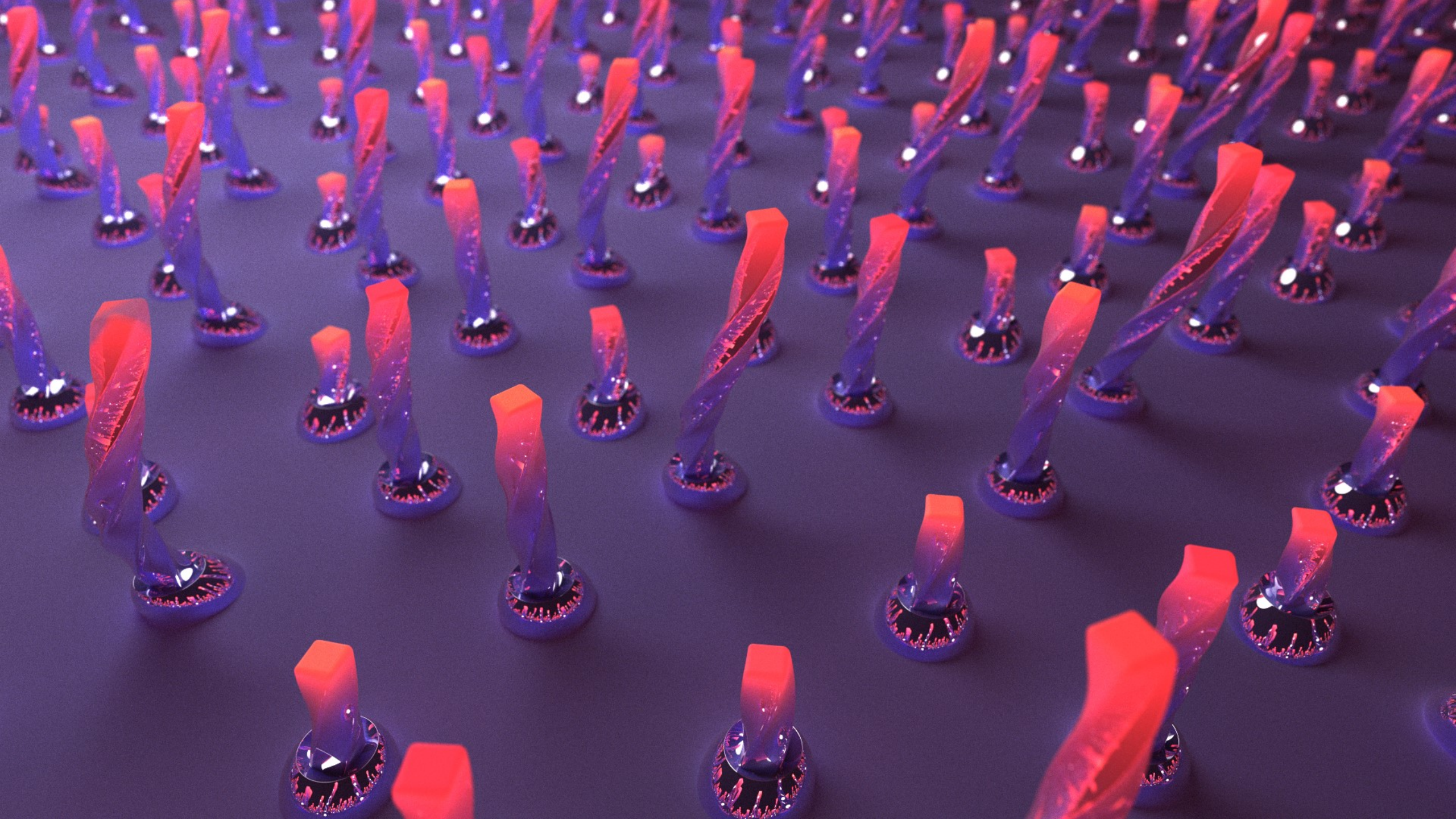
- Late Policy: 10% penalty, ***three*** days max
- Try “getting started” sample code now!
- Assignments, probably covering:
  - Rotations and transform hierarchies
  - Projections, lighting, and GLSL
  - Mesh data structures, editing, subdivision, simplification,
  - Ray tracing (with competition)











Getting started!

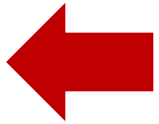
- Get started with Eclipse and JOGL
  - See MyCourses discussion board
  - We will use some other jars too
  - In class, will provide some introduction to OpenGL, java bindings, and general graphics debugging strategies
- Need more help?
  - TA contact info and office hours on My Courses
  - Prof office hours ***10-11 am every weekday but thursday***, **meet.google.com/whs-tskd-uii** and ping by email (see mycourses) or ***any other time*** by email appointment

Question

- What can you do if you are having trouble?

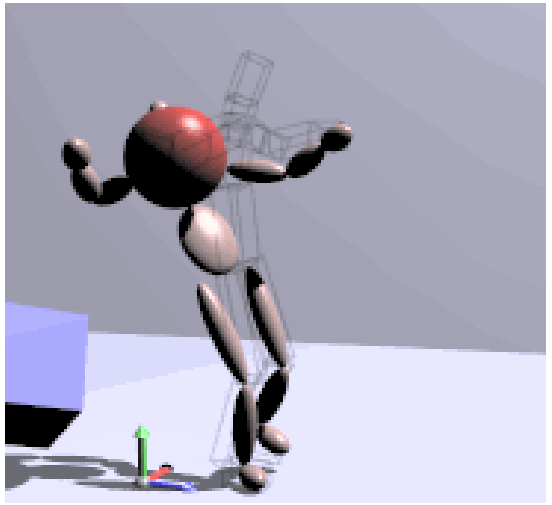
Today

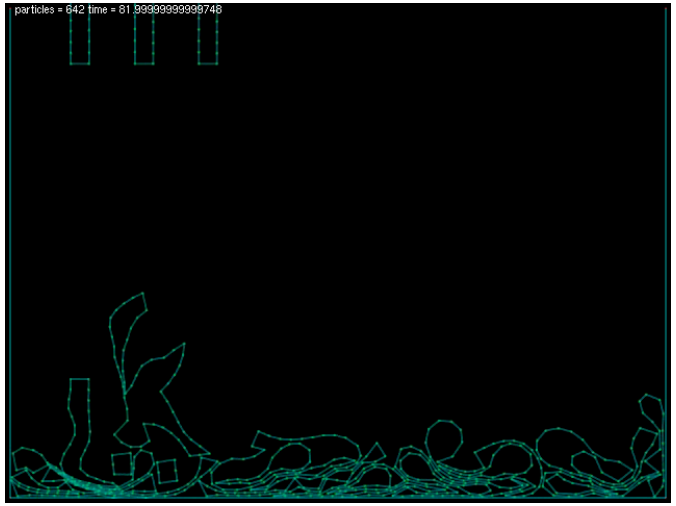
- Research overview
- Introduction
- Course details
- Announcements
- Transforms

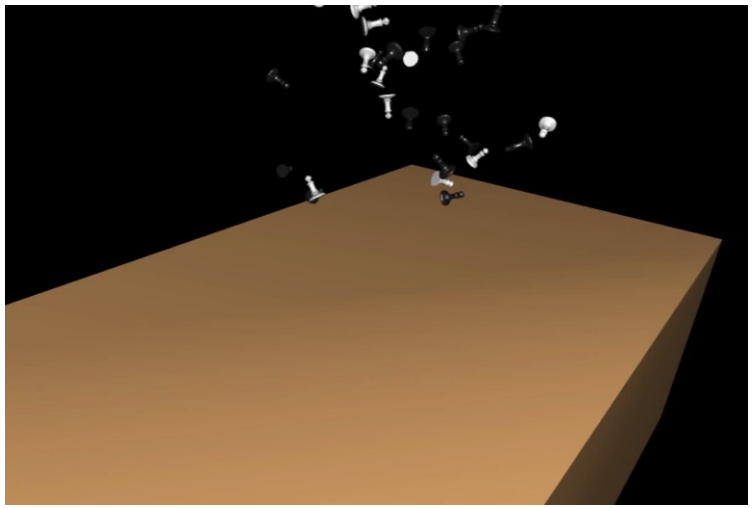


COMP 559 Fundamentals of Computer Animation

- 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 and possibly a mini project!





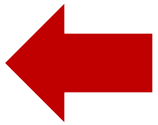


Other Graphics Related Courses

- Derek Nowrouzezahrai, Realistic & Advanced Image Synthesis (ECSE)
  - http://www.cim.mcgill.ca/~derek/ecse446.html but also listed as COMP 598/499
- Mikhail Bessmeltsev, Geometric Modeling and Shape Analysis (UdM)
  - http://www-labs.iro.umontreal.ca/~bmpix/teaching/6112/2018/
- Tiberiu Popa, Digital Geometry Modeling (Concordia)
  - https://users.encs.concordia.ca/~stpopa/teaching.html
- Sheldon Andrews, Physique des jeux (ETS)
  - http://profs.etsmtl.ca/sandrews/#teaching

Today

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



Some 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 if we are on the curve

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

  - Assume 2D, i.e., $\mathbf{v} \in \mathbb{R}^2$
  - Set always defines the boundary of a region, assuming f is continuous
- Example: line (orthogonal to $\mathbf{u}$, distance k from 0)

$$\{\mathbf{v} \mid \mathbf{v} \cdot \mathbf{u} + k = 0\}$$

- Example: circle (center $\mathbf{p}$, radius r)

$$\{\mathbf{v} \mid (\mathbf{v} - \mathbf{p}) \cdot (\mathbf{v} - \mathbf{p}) - \mathbf{r}^2 = 0\}$$

Explicit Representations

- Also called parametric
- Equation maps a domain into the plane

$$\{\mathbf{f}(t) \mid t \in D\}$$

  - Variable t is the "parameter", a scalar, and function $\mathbf{f}$ range is 2D
  - Like tracing out the path of a particle over time
- Example: line (containing $\mathbf{p}$, parallel to $\mathbf{u}$)

$$\{\mathbf{p} + t\mathbf{u} \mid t \in \mathbb{R}\}$$

- Example: circle (center $\mathbf{p}$, radius r)

$$\left\{\mathbf{p} + \mathrm{r}\begin{pmatrix}\cos t \\ \sin t\end{pmatrix} \mid t \in [0,2\pi)\right\}$$

Transforming Geometry

- Move a set of points using a mapping T from the plane to itself

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

- Parametric representation:

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

- 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: $T^{-1}(\mathbf{v}) = \mathbf{v} - \mathbf{u}$
- Consider transforming a circle
  - Explicit form
  - Implicit form

Linear Transformations

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

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

- Such transformations are *linear*, which means

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

- All linear transformations can be written by matrix

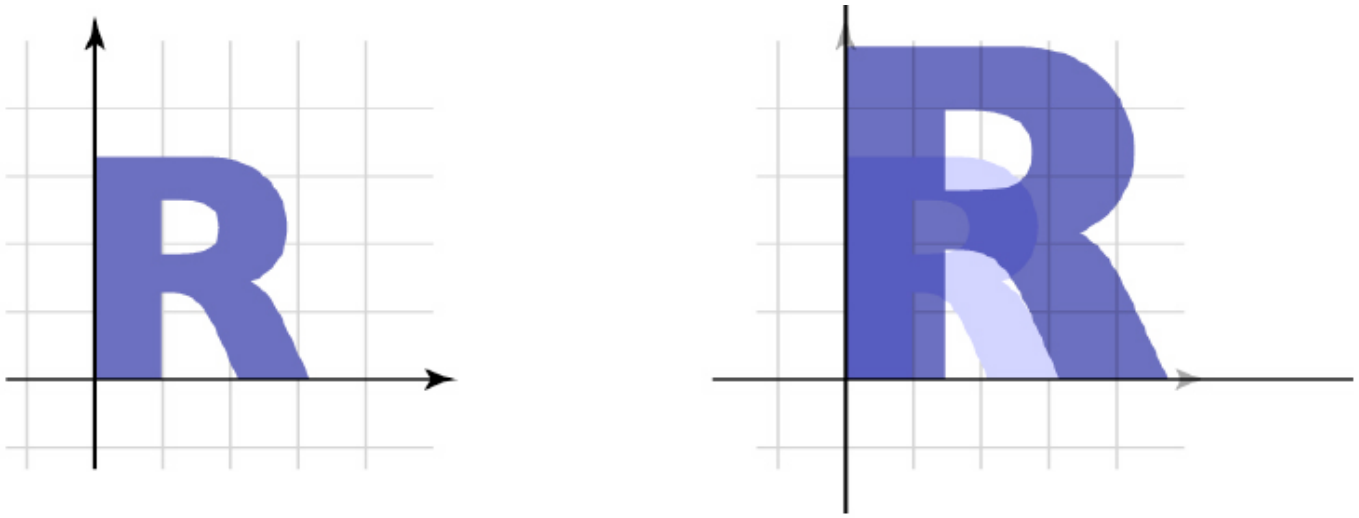
Geometry of 2D linear Transformations

- 2x2 matrices have simple geometric interpretations
  - Uniform scale
  - Non-uniform scale
  - Rotation
  - Shear
  - Reflection
- Let us look at examples of each of these

Linear transformation gallery

- Uniform scale

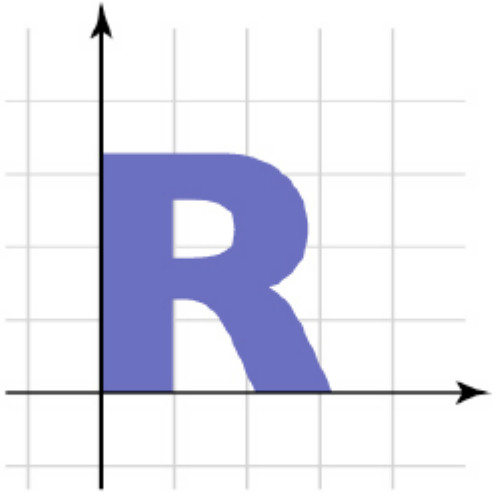
$\begin{pmatrix} s & 0 \\ 0 & s \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} sx \\ sy \end{pmatrix}$, for example, $\begin{pmatrix} 1.5 & 0 \\ 0 & 1.5 \end{pmatrix}$

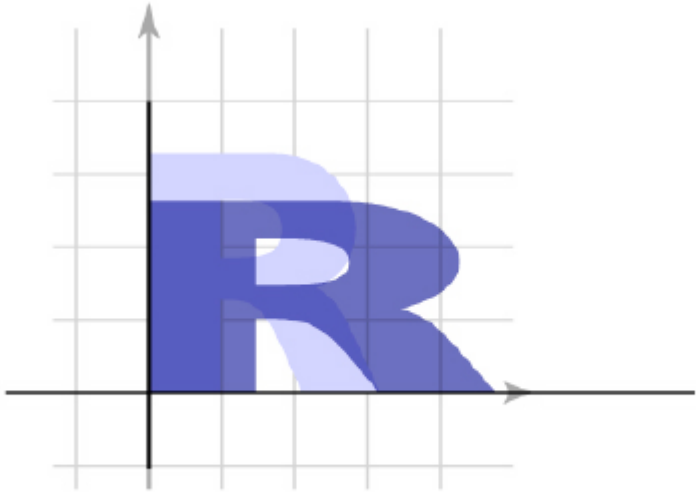


Linear transformation gallery

- Nonuniform scale

$$\begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} s_x x \\ s_y y \end{pmatrix}, \text{ for example, } \begin{pmatrix} 1.5 & 0 \\ 0 & 0.8 \end{pmatrix}$$



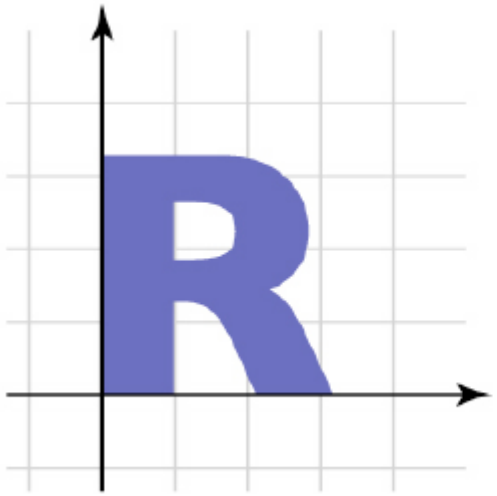


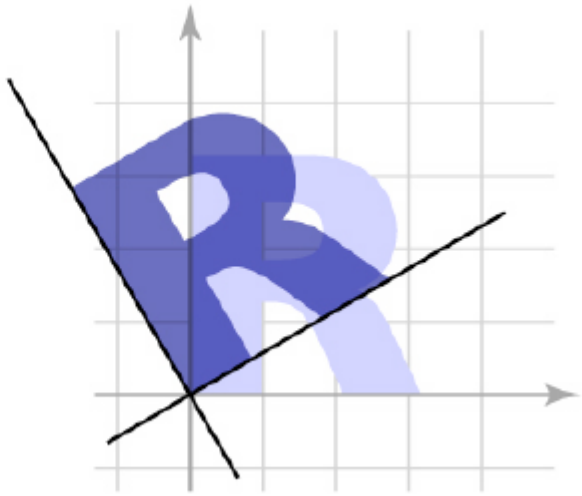
Linear transformation gallery

- Rotation

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{pmatrix},$$

for example, $\begin{pmatrix} 0.866 & -0.5 \\ 0.5 & 0.866 \end{pmatrix}$

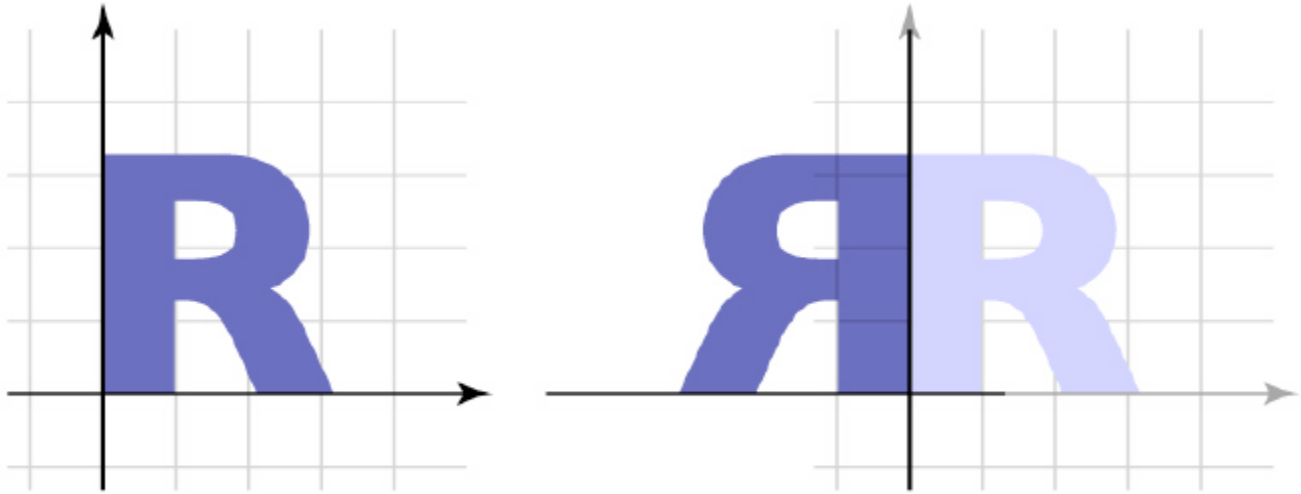




Linear transformation gallery

- Reflection, can consider it a special case of nonuniform scale, for example,

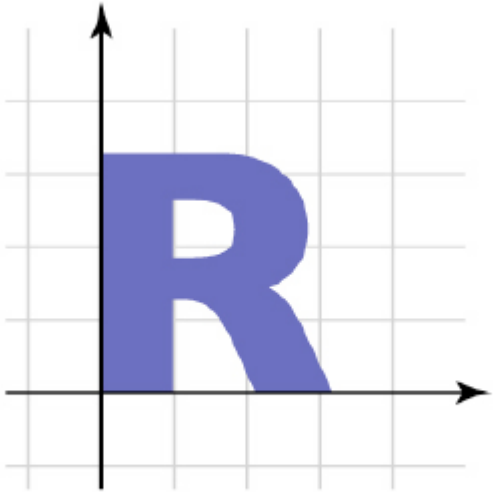
$$\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

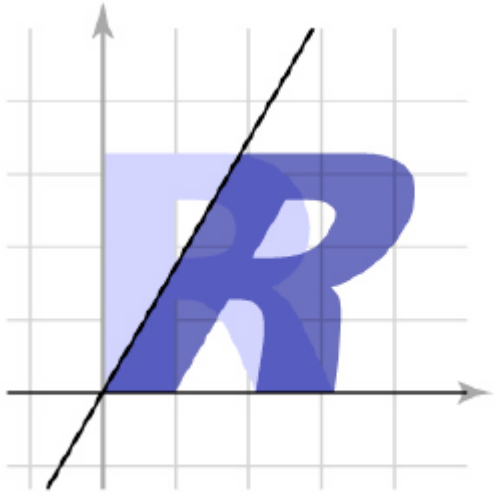


Linear transformation gallery

- Shear

$\begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x + ay \\ y \end{pmatrix}$, for example, $\begin{pmatrix} 1 & 0.5 \\ 0 & 1 \end{pmatrix}$





Composing Transformations

- Want to move an object, and move it some more?

$$\mathbf{p} \to T(\mathbf{p}) \to S\big(T(\mathbf{p})\big) = (S \circ T)(\mathbf{p})$$

- We need to represent $S \circ T$ (that is, S compose T)
  - Would like to use the same representation as for S and T
- Translation is 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 $\mathbf{u}_T + \mathbf{u}_S$

Composing Transformations

- Linear transformations are 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, same as transforming by $M_S M_T$
  - Note $M_S M_T$ or $S \circ T$ is always T first, then S
  - Only sometimes commutative!
  - What commutes in 2D?
    - Rotation with uniform scale
    - Non-uniform scale and non-uniform scale
    - Translation and translation

What doesn't commute in 2D?

- Rotations and translations
- Rotations and non-uniform scale
- Translation and scale

Combining Linear with Translation

- Need to use both in a single framework
- Can represent sequence as $T(\mathbf{p}) = M\mathbf{p} + \mathbf{u}$
- Now consider composition

$$\begin{aligned} T(\mathbf{p}) &= M_T\mathbf{p} + \mathbf{u}_T \\ S(\mathbf{p}) &= M_S\mathbf{p} + \mathbf{u}_S \\ (S \circ T)(\mathbf{p}) &= M_S(M_T\mathbf{p} + \mathbf{u}_T) + \mathbf{u}_S \\ &= (M_S M_T)\mathbf{p} + (M_S\mathbf{u}_T + \mathbf{u}_S) \end{aligned}$$

- This will work, but it is a bit awkward

Homogeneous Coordinates

- Can representing transformations more elegantly
- Extra component w for vectors, and an extra row and column for matrices
  - For points in affine space, we can keep $w = 1$
- Linear transformations do not make use of the extra row and column

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

Homogeneous Coordinates

- We represent translation using the extra column

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

Homogeneous Coordinates

- Composition works by 3-by-3 matrix multiplication (note we use block matrix notation)

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

- This is exactly the same as carrying around M and $\mathbf{u}$
  - But cleaner
  - And generalizes in useful ways as we will see later!

Review and more information

- Review
  - CGPP Chapter 7 up to and not including 7.7
    - Essential Math and Geometry of 2D and 3D
- CGPP Chapter 10 up to and including 10.10
  - 2D transformations
  - Points and vectors
  - Skip the SVD (10.3.8 and 10.3.9)
  - Inverses of transformation matrices
  - Coordinate transformations