# **CS5625** Interactive Computer Graphics

Feel free to explore the rooms in the Ohyay space!
We'll start a few minutes late to allow for login/access issues.

Steve Marschner
Spring 2022
**01** Introduction

# Introductions

**Steve Marschner**

- instructor, Professor of CS specializing in graphics

**Joy Zhang**

- PhD TA, expert on physics simulation for graphics, game developer and C++ hacker

**Andrei Shpilenok**

- ugrad TA, 5625 veteran, Linux and post-OpenGL API enthusiast

**Ruby Min**

- ugrad TA, 5625 veteran, CS/ECE double major

# Inclusion

**I want you here in this class, with your own distinct…**

…voice

…perspective

…needs

…differences

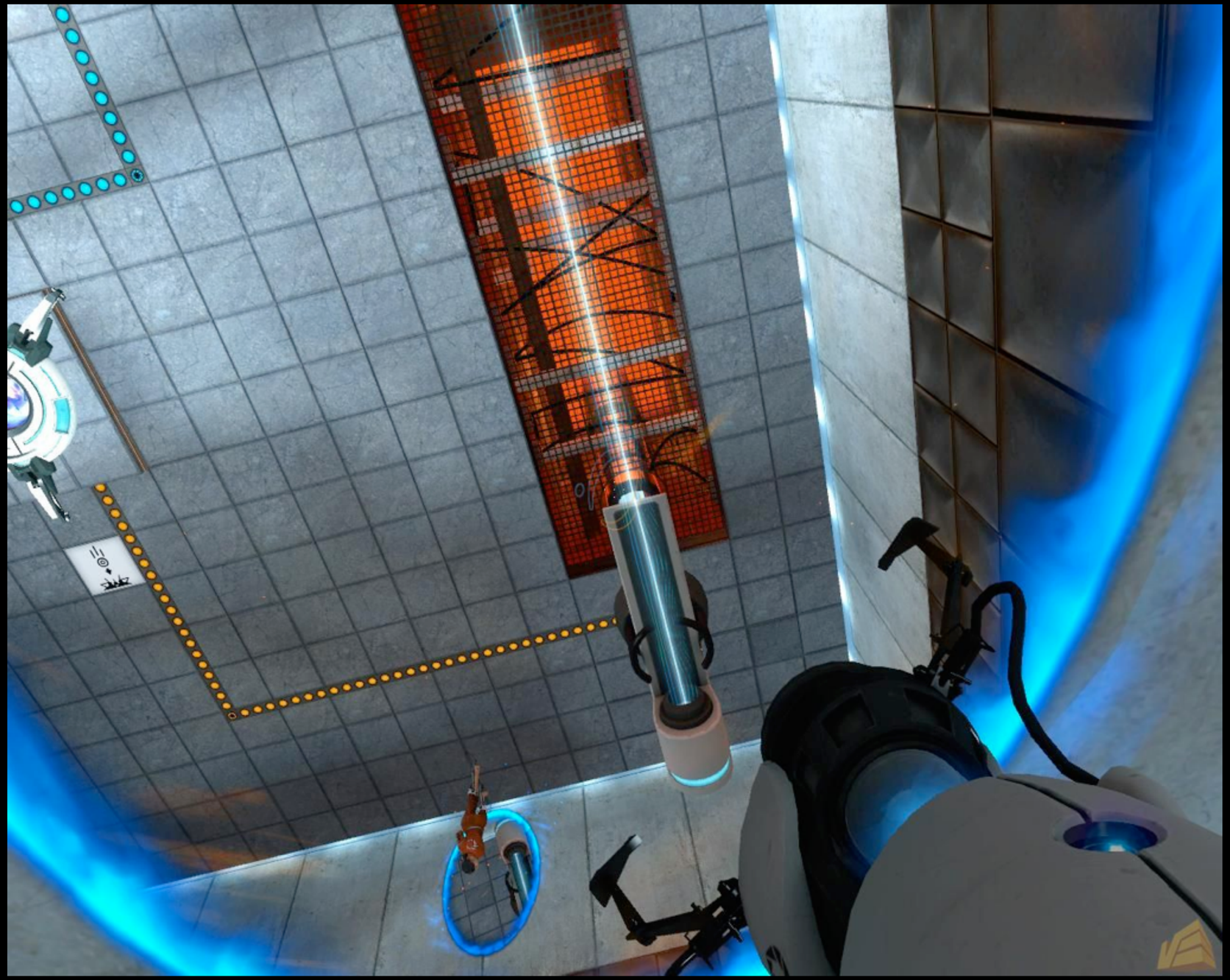
I and the course staff promise to always do our best to create an environment where you know you belong and can have fun learning about graphics—but please let us know if we are not succeeding; we want to know how to do better.

# Applications

Rockstar Games—*Red Dead Redemption 2* (2018)


CD Projekt RED—*Cyberpunk 2077* (2020)

Naughty Dog—*The Last of Us* (Remastered, 2014)



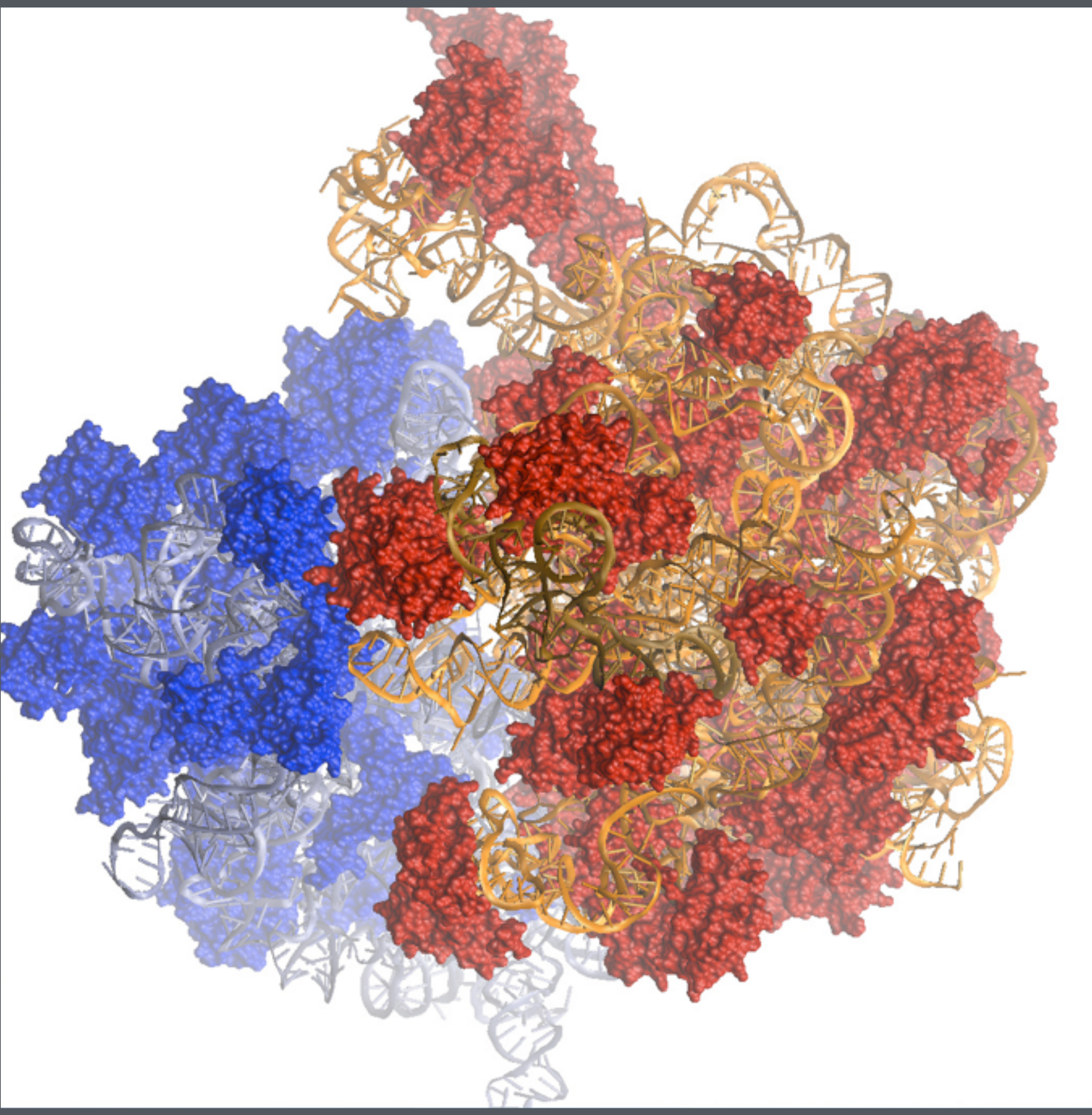
Valve—*Portal* (2007)


Ubisoft—*Child of Light* (2014)

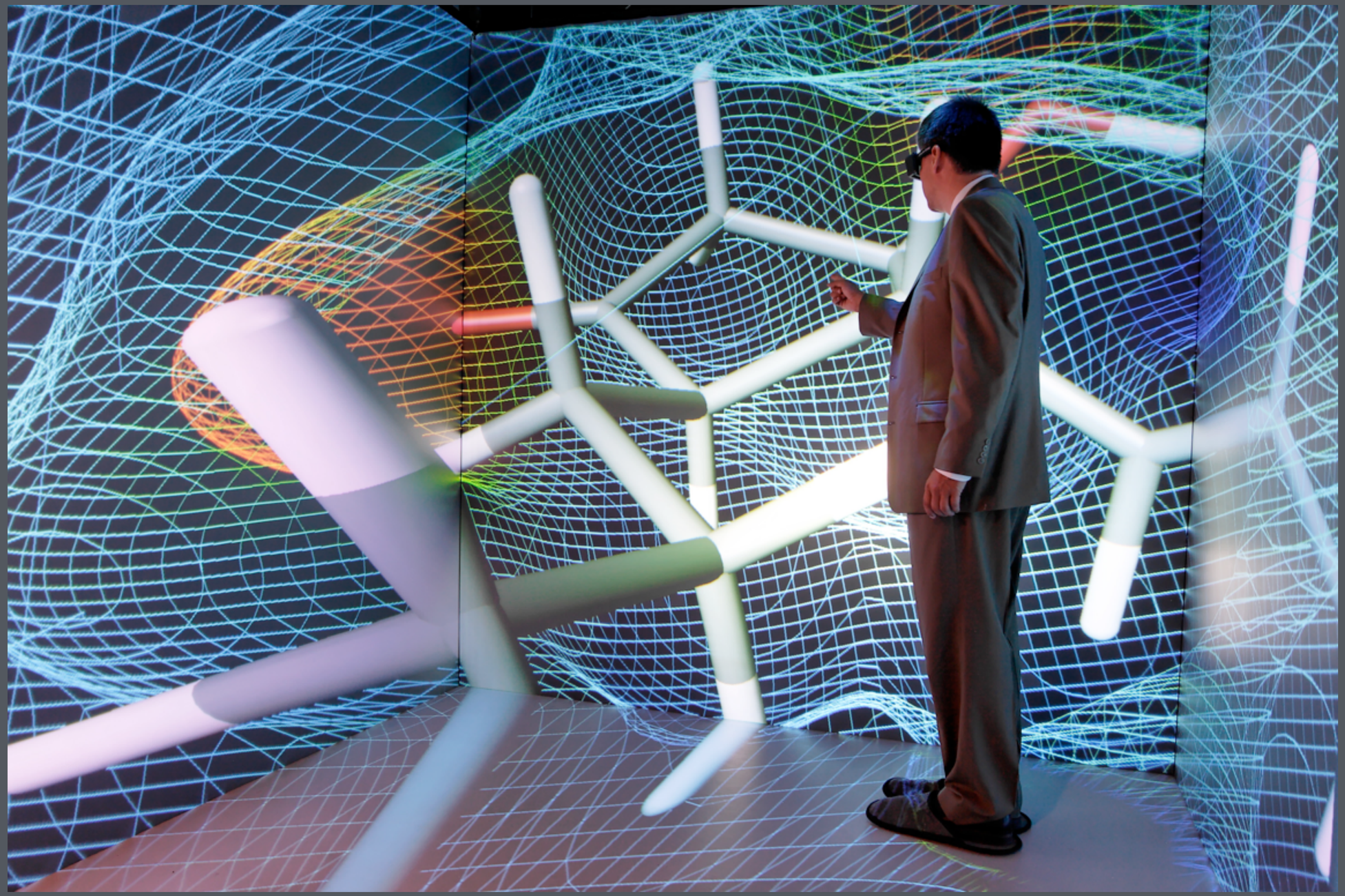
Meta Quest 2 — Oculus VR


Oculus VR

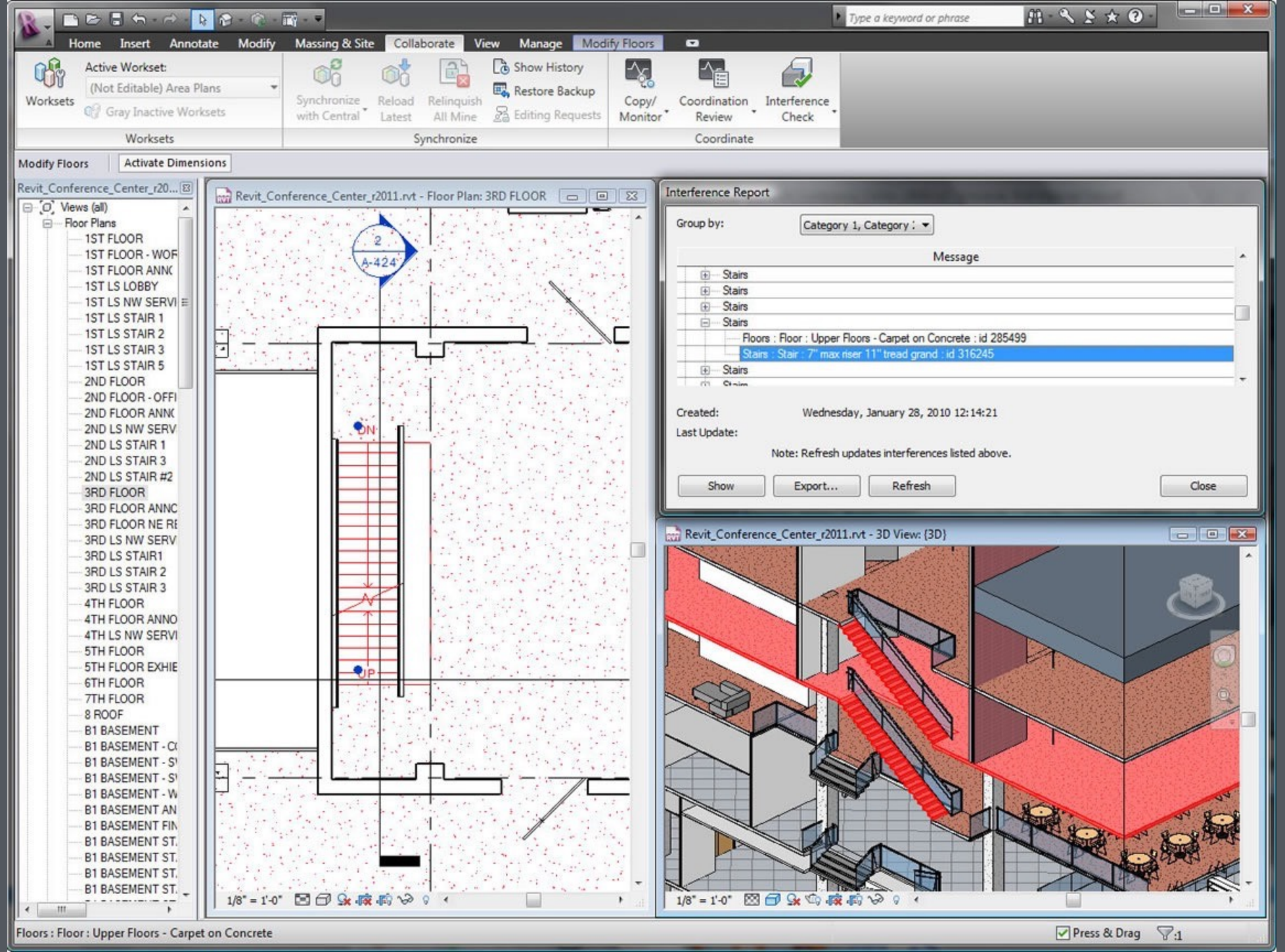
NASA



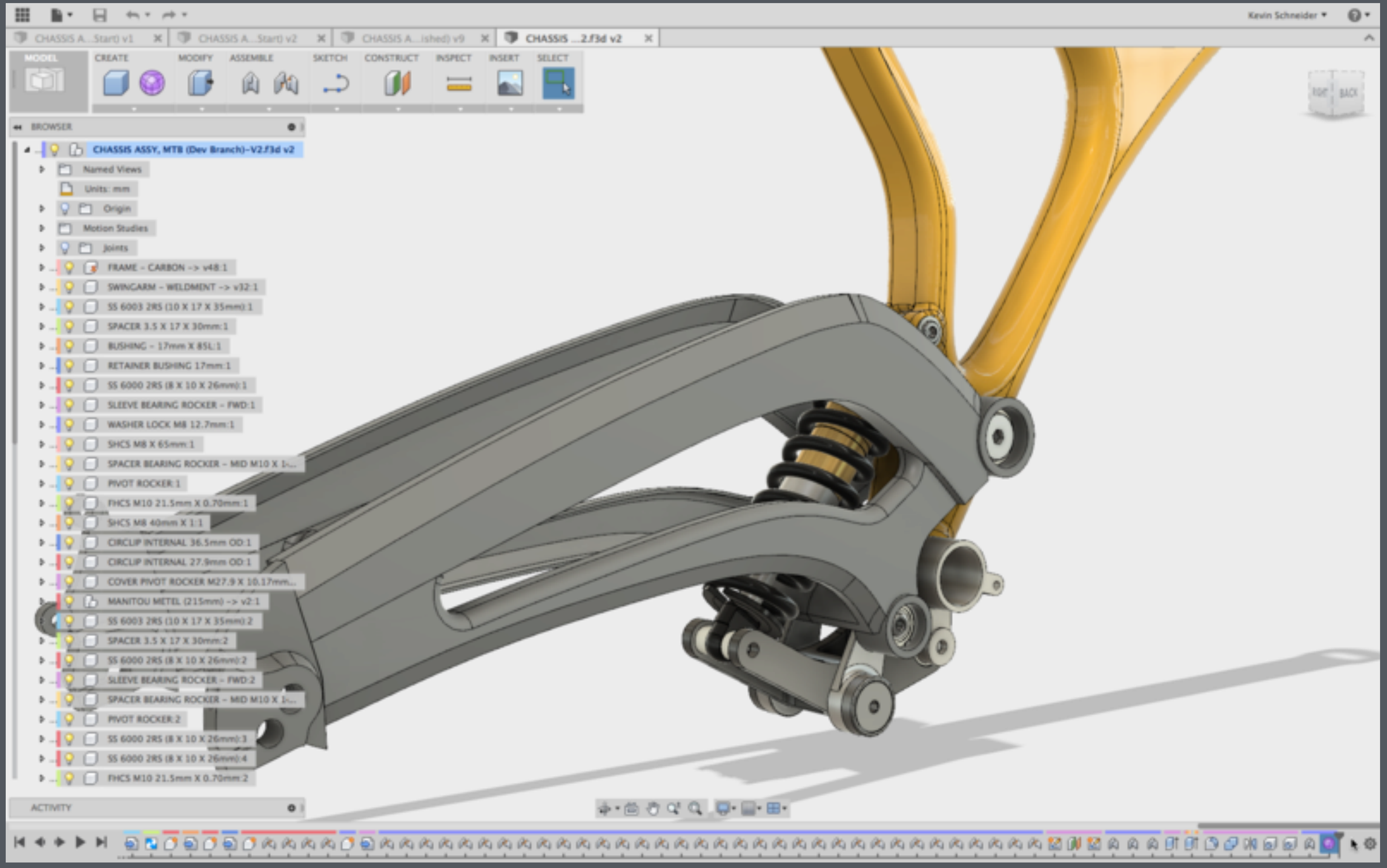
[John C. Stone, UIUC]



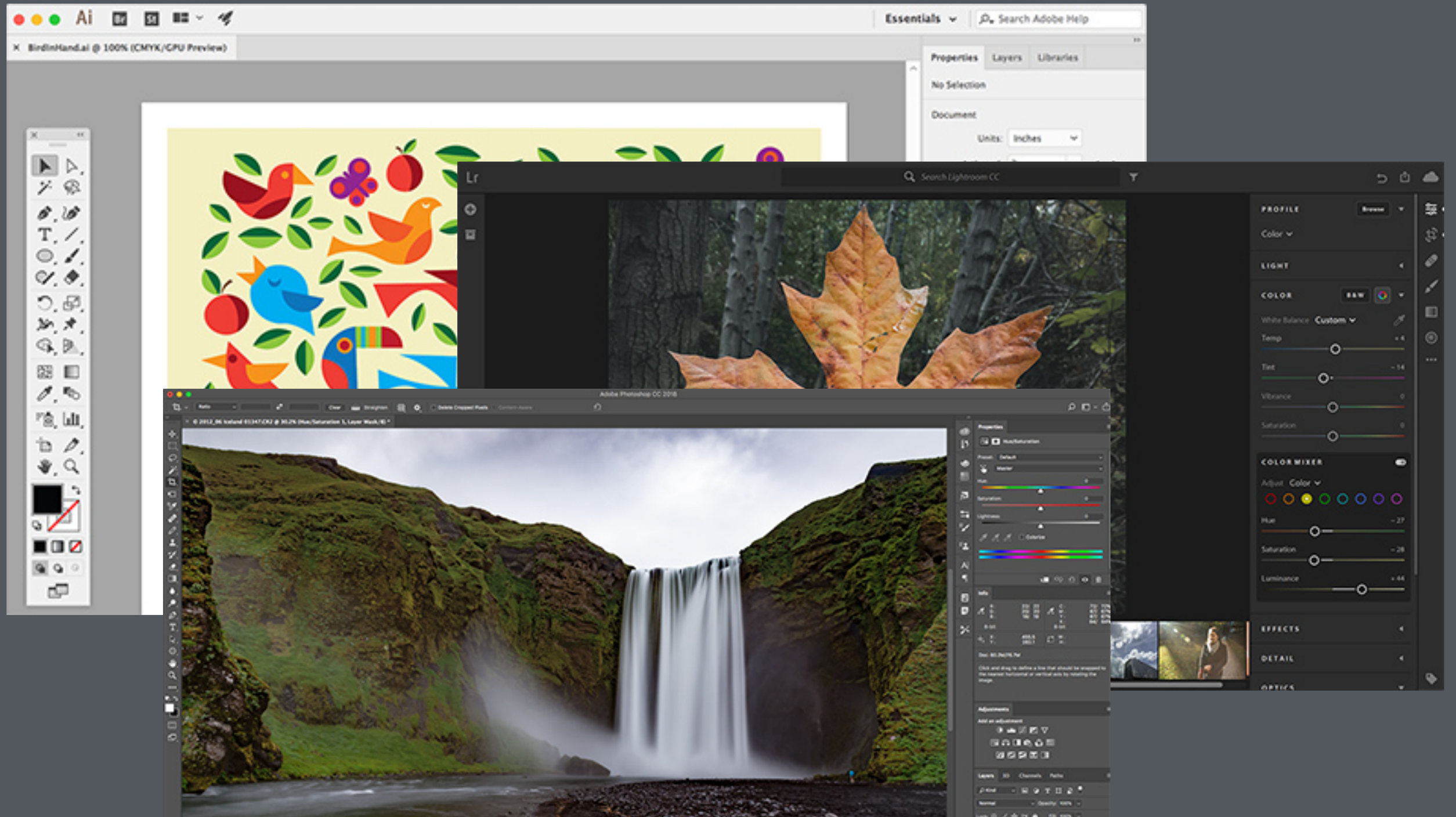
University of Calgary




Autodesk Revit Architecture 2015




Autodesk Fusion 360


Essentials
Search Adobe Help
BirdInHand.ai @ 100% (CMYK/GPU Preview)
Properties
Layers
Libraries
No Selection
Document
Units:
Inches
Search Lightroom CC
PROFILE
Color
LIGHT
COLOR
B&W
White Balance
Custom
Temp
Tint
Vibrance
Saturation
COLOR MIXER
Adjust
Color
Hue
Saturation
Luminance
EFFECTS
DETAIL
Adobe Photoshop CC 2018

# Breakout

**Practice with breakout mechanics…**

- Choose a small group room (each fits up to 6).  Some interst keywords to help you pick
  - 1. Art/Design
  - 2. Machine vision
  - 3. Math
  - 4. PhD/MS/Meng student
  - 5. Animation
  - 6. High performance code
  - 7. Realism
  - 8. Non-CS major
- Feel free to move between rooms if you feel like it!
- Introduce yourselves with name, major/field/year, and something you thought was fun about your previous graphics course (probably 4620 for most people)
- I'll call you back here after 5 or 10 minutes

# History

# How To Draw a Triangle, c. 1985

**Transform vertices to screen coordinates**

**Find all the pixels covered by the triangle**

**Fill all the pixels with the triangle’s color**

# How To Draw a Triangle, c. 1988

**Perform lighting calculations to find vertex colors**

**Transform vertices to screen coordinates**

**Find all the pixels covered by the triangle**

**Fill all unoccluded pixels with the interpolated vertex colors and depth**

# How To Draw a Triangle, c. 1992

**Perform lighting calculations to find vertex colors**

**Transform vertices to screen coordinates**

**Find all the pixels covered by the triangle**

**Look up a texture map value**

**Fill all unoccluded pixels with a function of the texture and the interpolated vertex colors, as well as the depth**

# How To Draw a Triangle, c. 1999

**Perform elaborate lighting calculations to find vertex colors**

**Transform vertices to screen coordinates**

**Find all the pixels covered by the triangle**

**Look up a value from one or more 1D, 2D, or 3D texture maps**

**Fill all unoccluded pixels with a complicated, adjustable function of the textures and the interpolated vertex colors, as well as the depth**


NVIDIA GeForce2 tech demo (2001) recoded by GameForest (YouTube)


NVIDIA GeForce2 tech demo (2001) recoded by GameForest (YouTube)


Pixar—*Monsters Inc.* (2001)

# How To Draw a Triangle in 2001

**Execute a vertex program over all the vertices**

**Find all the pixels covered by the triangle**

**Execute a fragment program over all those pixels**

**Fill all unoccluded pixels with the resulting color and depth**

# How To Draw a Triangle in 2007

**Execute a vertex program over all the vertices**

**Execute a geometry program over all primitives**

**Find all the pixels covered by the triangle**

**Execute a fragment program over all those pixels**

**Fill all unoccluded pixels with the resulting color and depth**

# How To Draw a Triangle in 2010

**Execute a vertex program over all the vertices**

**Execute tesselation programs to refine primitives**

**Execute a geometry program over refined primitives**

**Find all the pixels covered by the triangle**

**Execute a fragment program over all those pixels**

**Fill all unoccluded pixels with the resulting color and depth**


NVIDIA tech demo (c.2016) recoded by GameForest (YouTube)


NVIDIA tech demo (c.2016) recoded by GameForest (YouTube)

# How To Draw a Triangle in 2020

**Execute a vertex program over all the vertices**

**Execute tesselation programs to refine primitives**

**Execute a geometry program over refined primitives**

**Find all the pixels covered by the triangle**

**Execute a fragment program over all those pixels**

**Fill all unoccluded pixels with the resulting color and depth**

**Integrate rasterized results with GPU ray tracing as needed**


Minecraft RTX (2019) demo by PC Centric (YouTube)


Minecraft RTX (2019) demo by PC Centric (YouTube)

# How to draw a triangle in 2025?

**Hit it with a ray**

**Execute a ray-hit shader that computes a Monte Carlo estimate**

**Denoise the result with learned filters**

NVIDIA / Omniverse RTX tech demo (2020) (YouTube) (SIGGRAPH presentation)

NVIDIA / Omniverse RTX tech demo (2020) (YouTube) (SIGGRAPH presentation)

# Breakout

**Questions for discussion:**

- What are some projects you've worked on where interactive graphics was part of the tools?
- What was bad about the graphics that made your life harder?
  How could it be improved?

**After we come back I'll encourage you to let us know about some of the examples you came up with**

# Course mechanics

# About CS5625

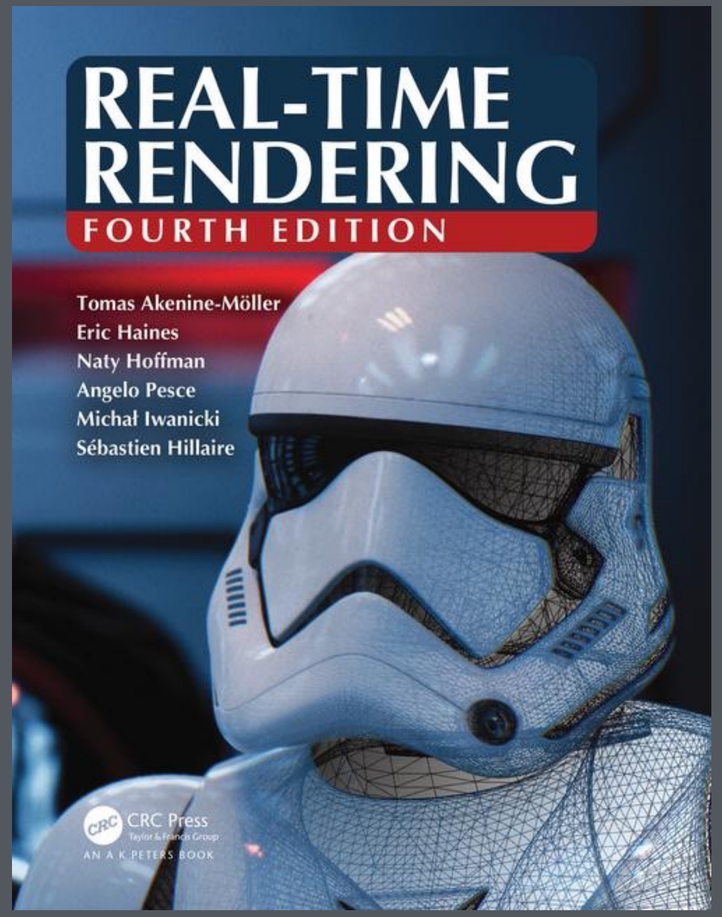
## Prereqs

- introductory graphics course (e.g. 4620) or instructor permission
- some familiarity with C/C++, or the time to learn on the fly

## Resources

- Course materials on website
  - schedule (very much subject to change!)
  - lecture slides, notes, readings
- Ohyay space for virtual interaction (experimental and very changeable — give us feedback!)
  - Lecture for first two weeks
  - Office hours, help sessions, hanging out to work on projects
- Handins and grades on CMS
- Ed Discussions for discussion and questions

# Recommended texts



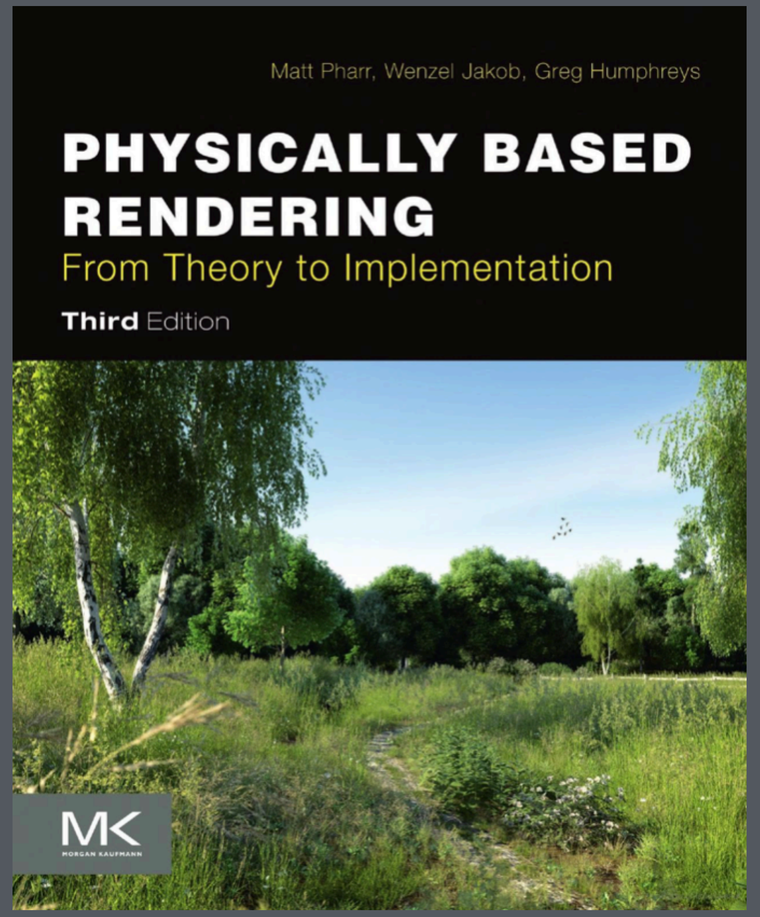

## Real-time Rendering

- Akenine-Moller, Haines, Hoffman
- encyclopedia of raster tricks
- available via library

## Physically-Based Rendering

- Pharr, Jakob, Humphries
- Oscar winning book on ray tracing
- available via pbr-book.org

## Various other readings



# Grading

## Course breakdown

- 60% from assignments
- 30% from final project
- 10% from participation

## Grading

- Holistic grading based on what you show us
- Grading principle: you prove to us what your code can do
    - written project report
    - Demo sessions or video demos

# Academic Integrity

**Don’t copy code from Web without careful attribution**

- small snippets of, e.g. OpenGL boilerplate OK **with attribution**

**Collaboration**

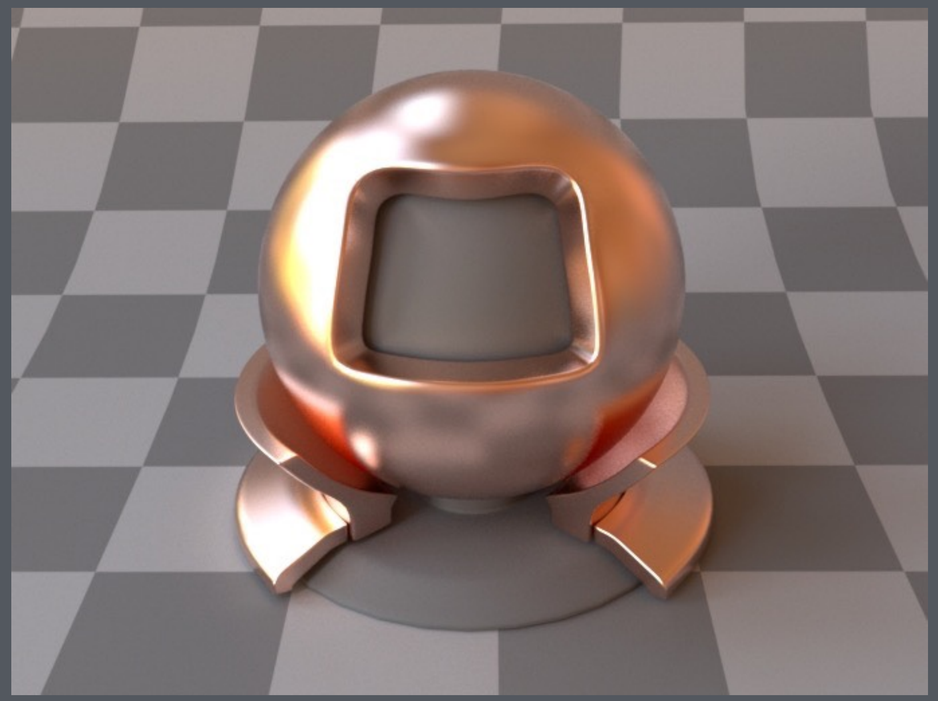
- do: help each other out, dispense advice, chat about design, help track down memory bugs
- don’t: tell other students exactly what to put in their code
- don’t: paste someone else’s code into yours

**Always cite sources of code and ideas**

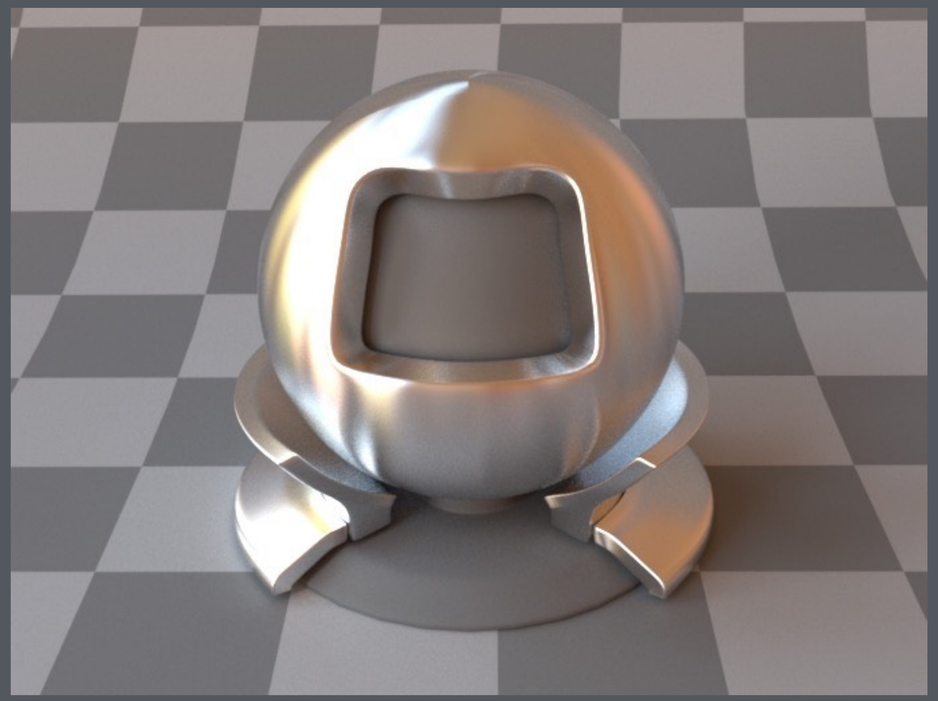
- think carefully about who and what contributed to your work
- if you tell me what is going on, there is never any AI problem

# Topics

# Shading and light reflection



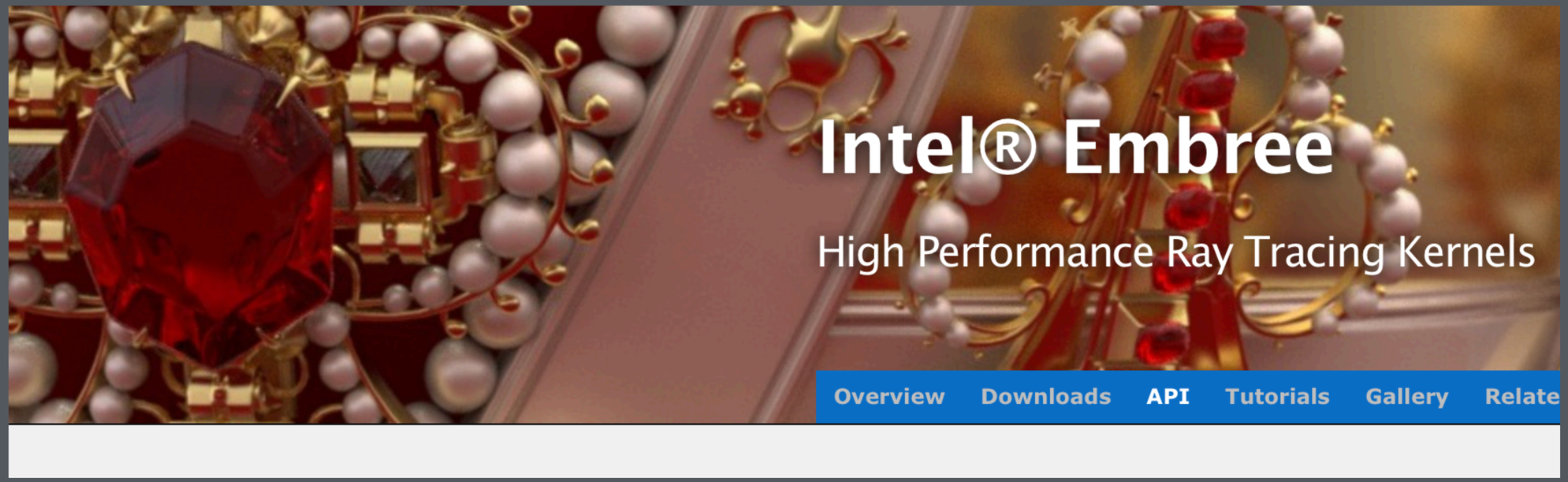
Cu ($\alpha = 0.1$)



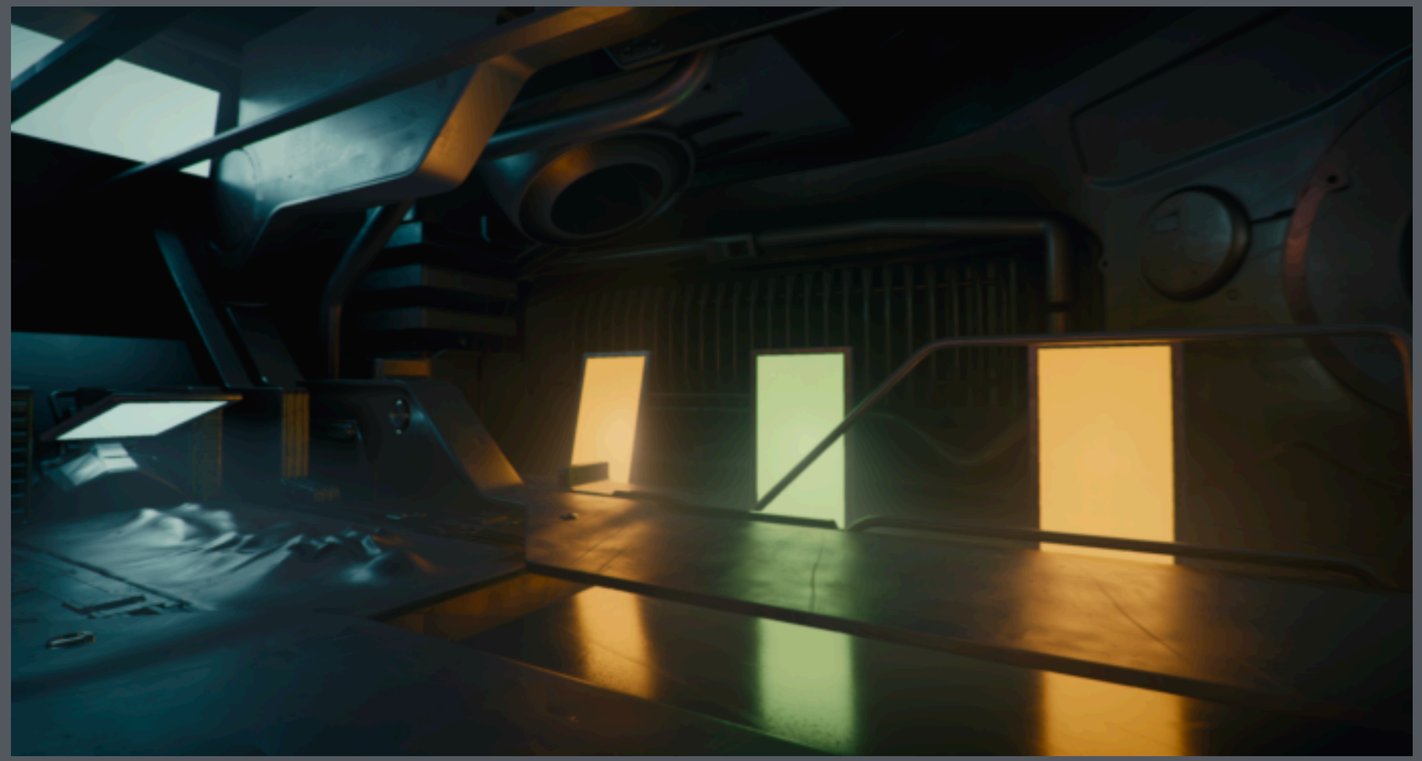
Al (anisotropic)

Wenzel Jakob / Mistuba

# Ray Tracing


Intel® Embree
High Performance Ray Tracing Kernels
Overview
Downloads
API
Tutorials
Gallery
Relate

# Illumination

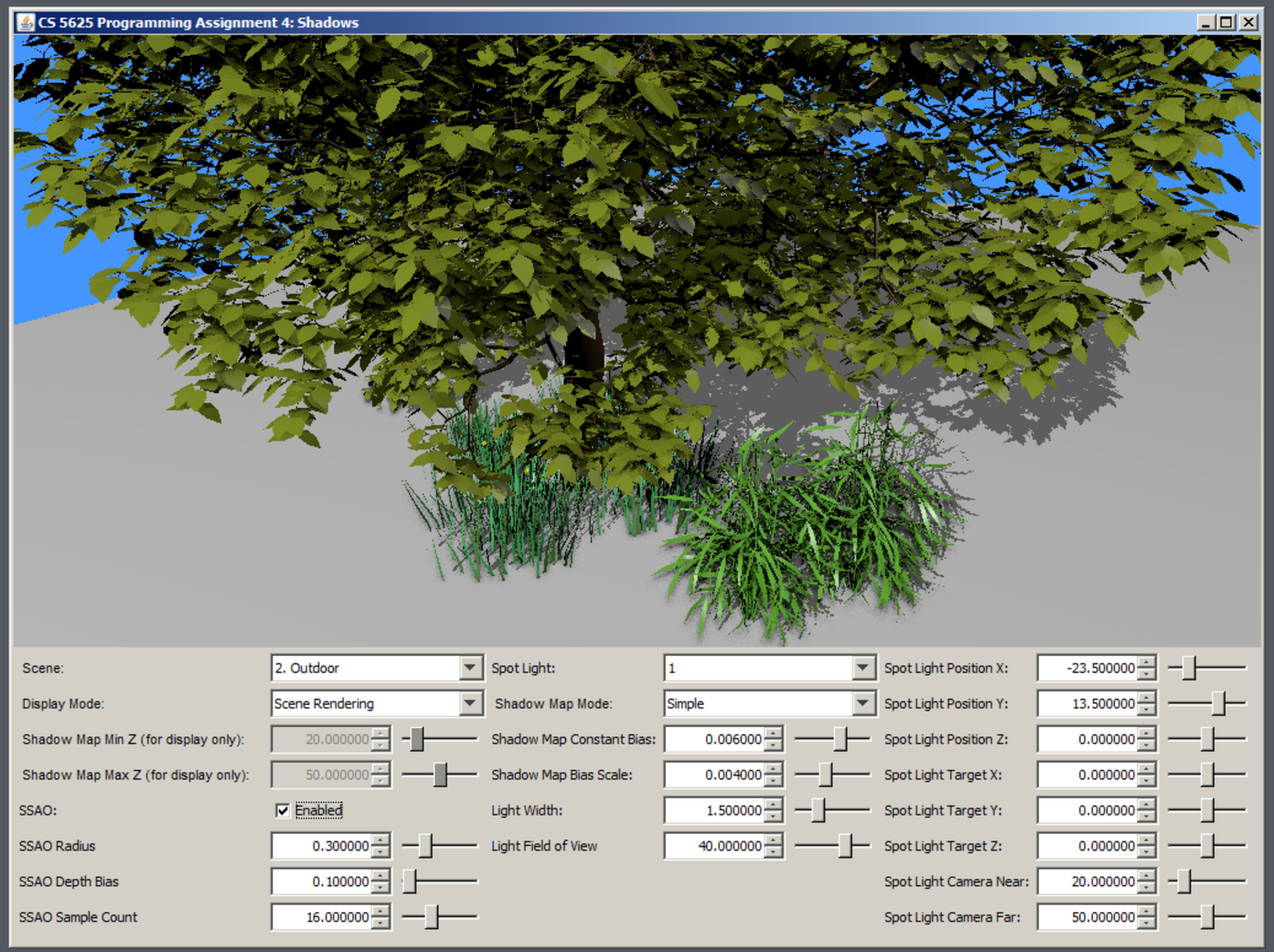


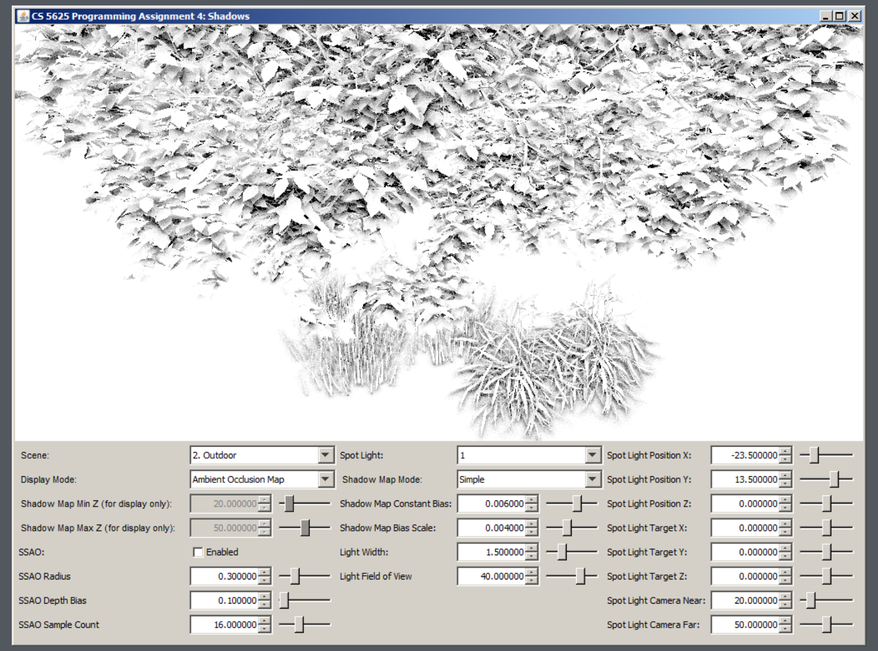
Eric Heitz

area lights

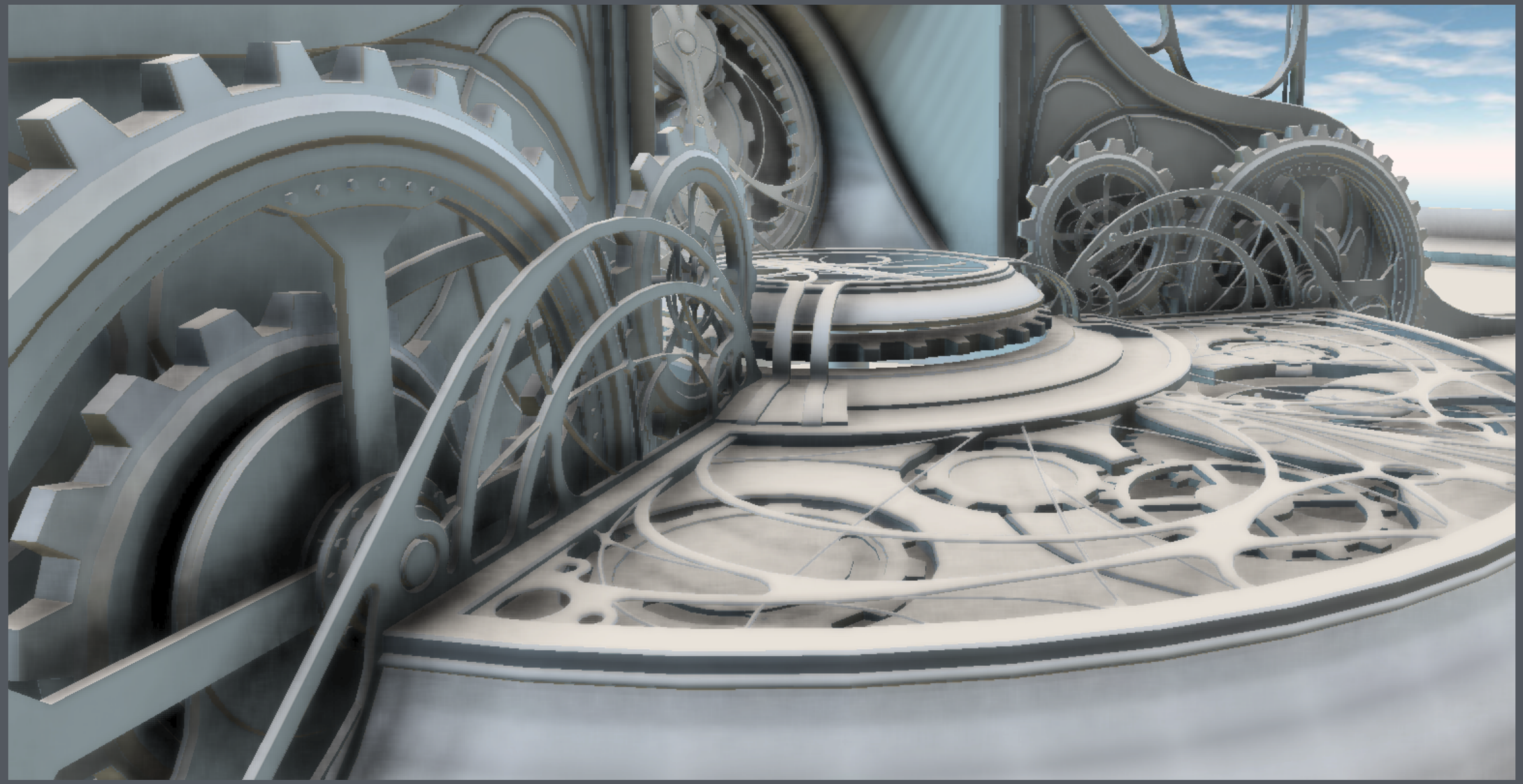

environment light

# Shadow algorithms



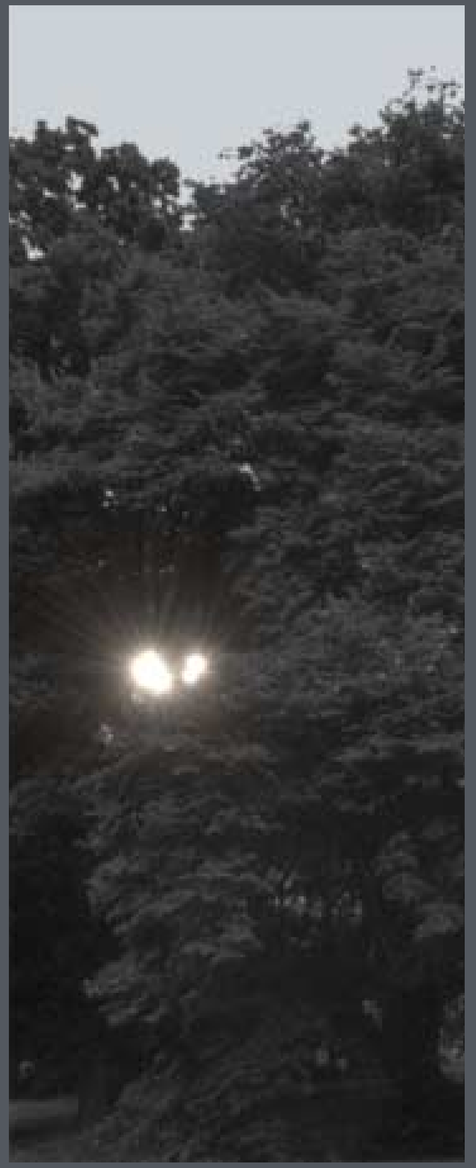



# Approximate soft illumination



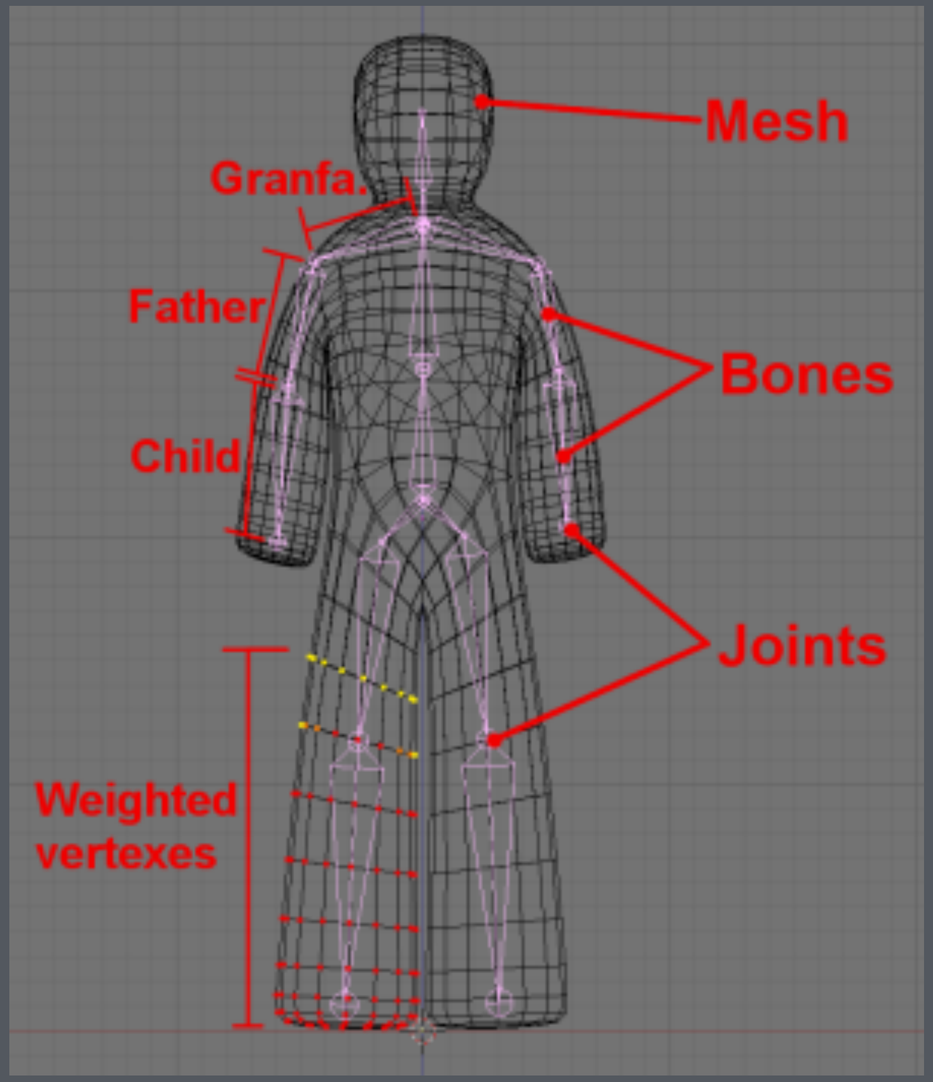
Morgan McGuire

# Post-processing effects

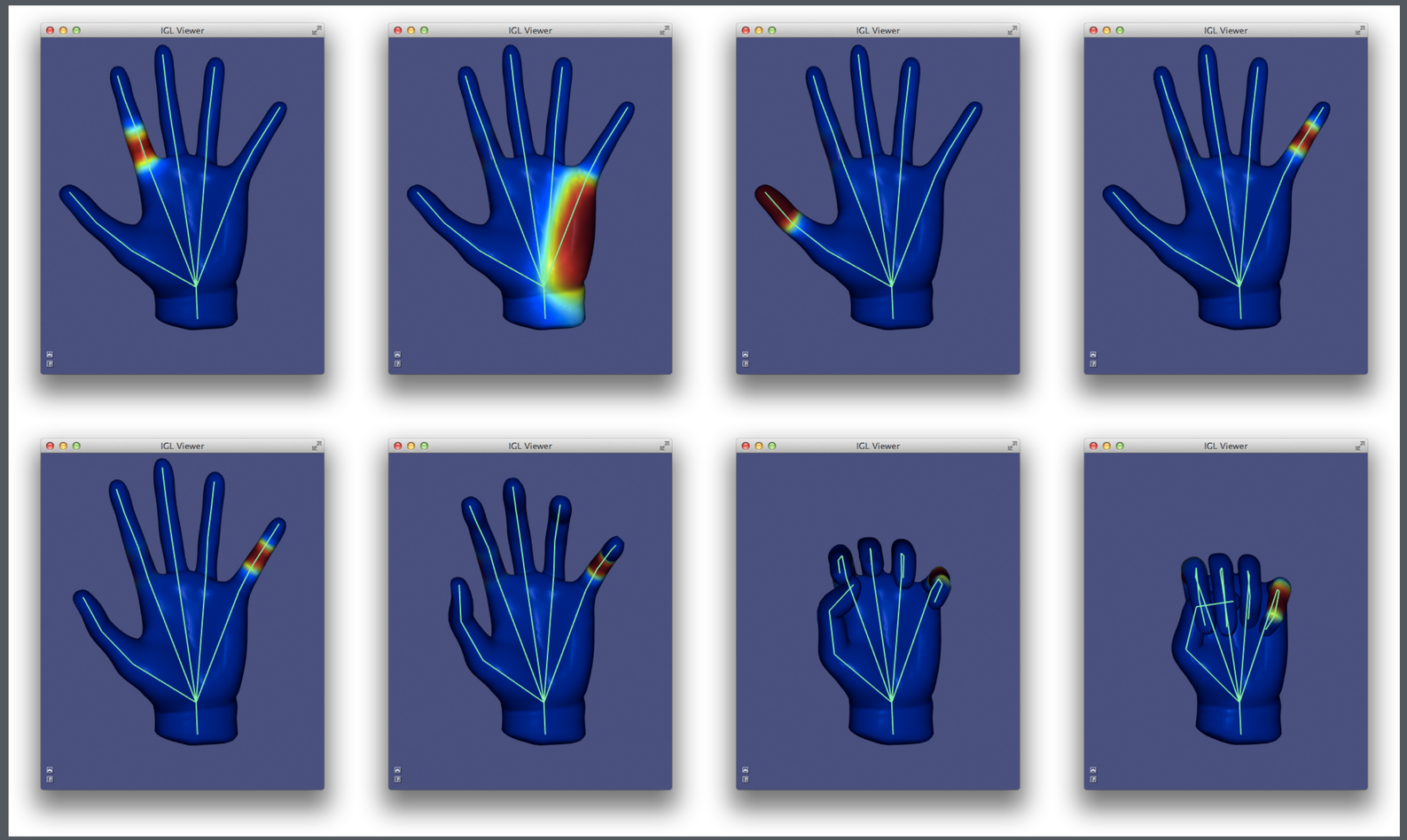


Greger et al. SIGGRAPH 2005

# Mesh animation




Game Programming Wiki (gpwiki.org)




Panozzo & Jacobson, libigl tutorial (libigl.github.io/libigl)

STAR: A Sparse Trained Articulated Human Body Regressor
Osman, Bolkart, and Black, ECCV 2020

STAR: A Sparse Trained Articulated Human Body Regressor
Osman, Bolkart, and Black, ECCV 2020

# CS 5625 Coursework

**3 projects** (working in pairs recommended)

- mostly implementation, sometimes written problems to work out math
- style is ground-up: we provide libraries but you write main()
- C++ and OpenGL
- topics: ray tracing, monte carlo, illumination, shading, texturing, shadows, mesh animation

**Final project** (groups of 3–5)

- project proposal
- milestone presentations and evaluation
- final project demos, presentations, writeup

# Final project

**An interactive 3D environment with fancy graphics**

- many groups build something game-like
- other kinds of proposals also welcomed

**Open ended, needs to have technically impressive results**

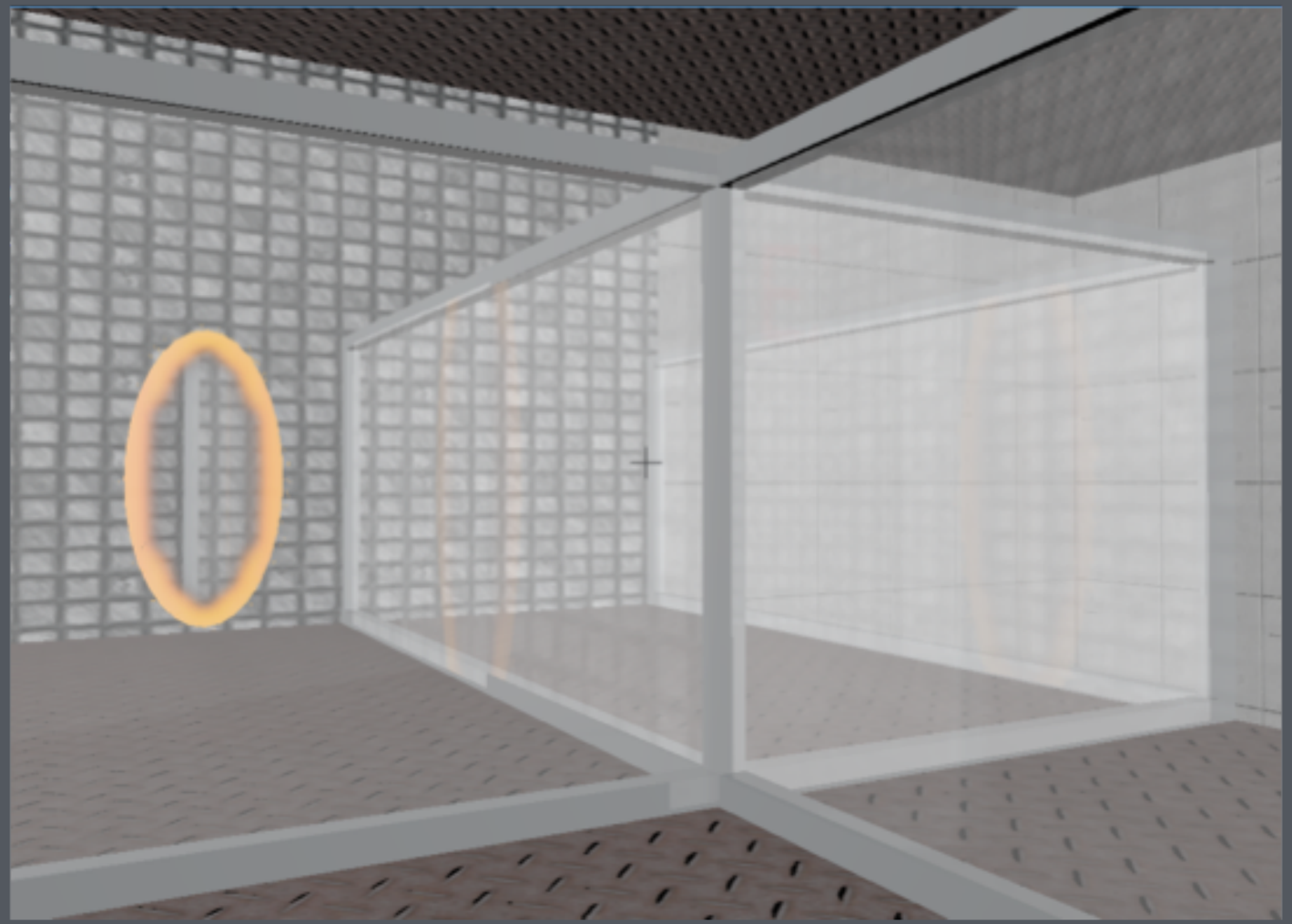
**Ways to impress**

- rendering: shading, shadows, global illumination, …
- modeling: splines, subdivision surfaces, procedural generation, …
- animation: character motion, collision detection, simulation, …
- imaging: flare, antialiasing, noise filtering, …

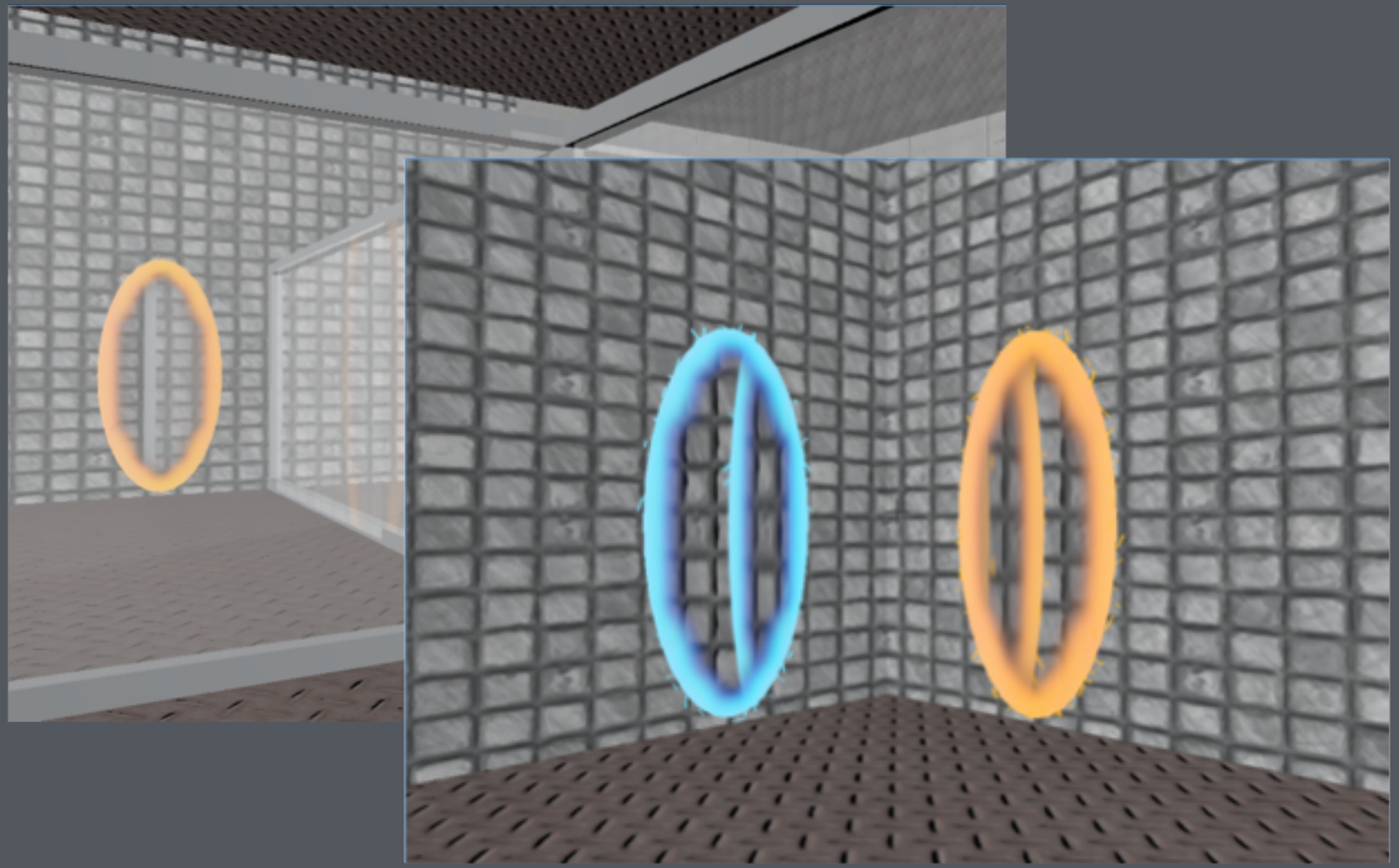
**Focus is on graphics, not gameplay**

# Final project examples

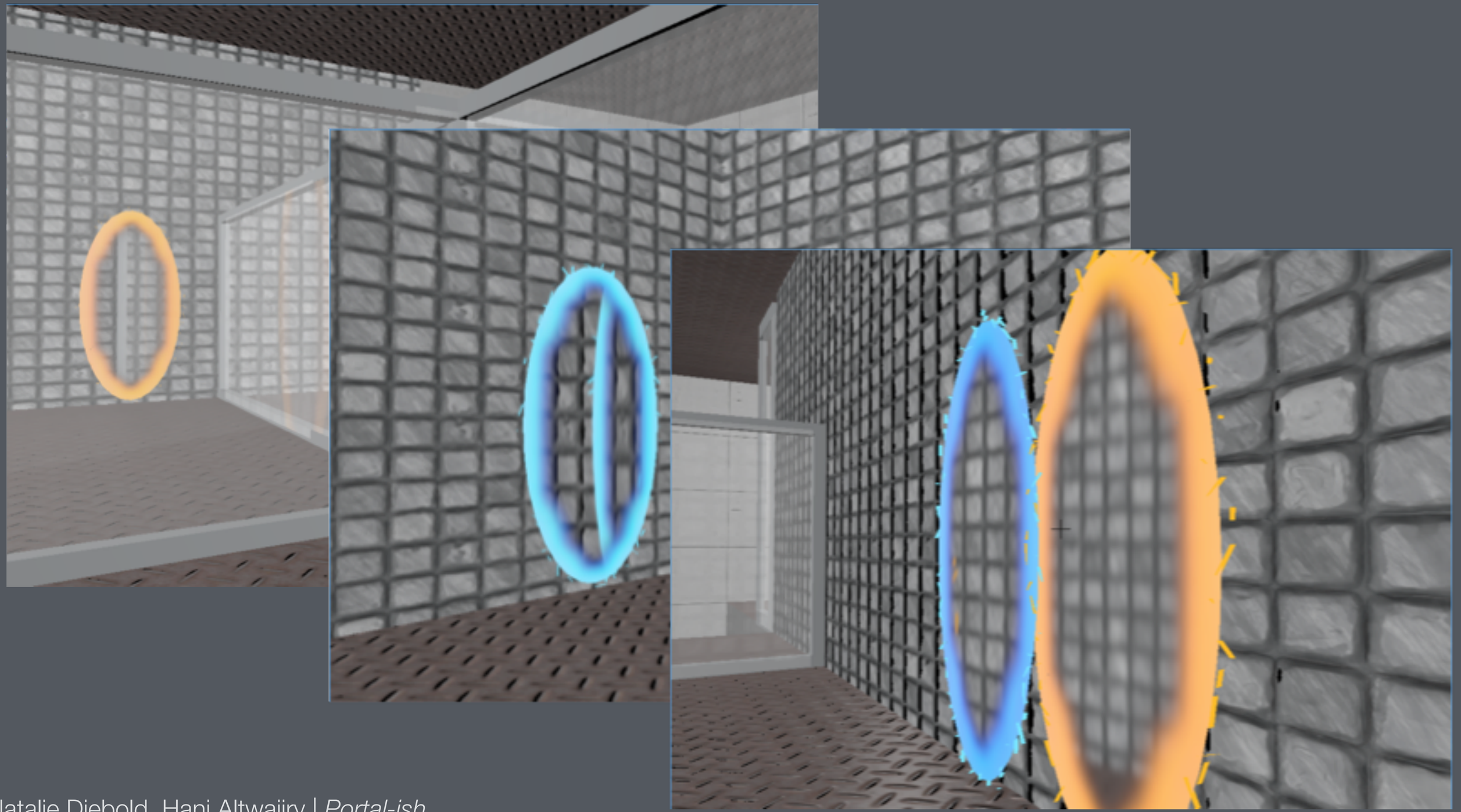
**2015 – 2019**



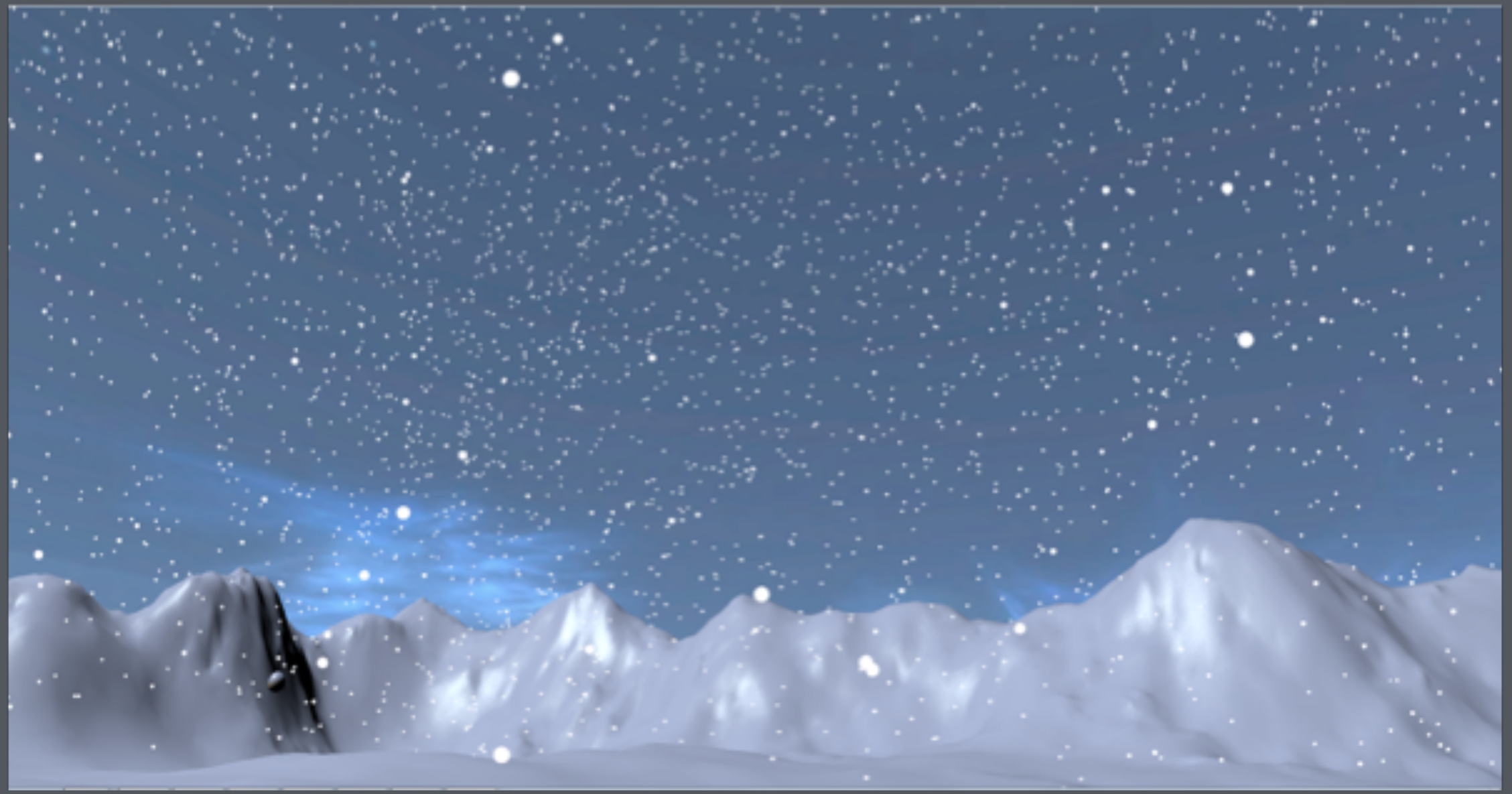
Natalie Diebold, Hani Altwaijry | *Portal-ish*



Natalie Diebold, Hani Altwaijry | *Portal-ish*



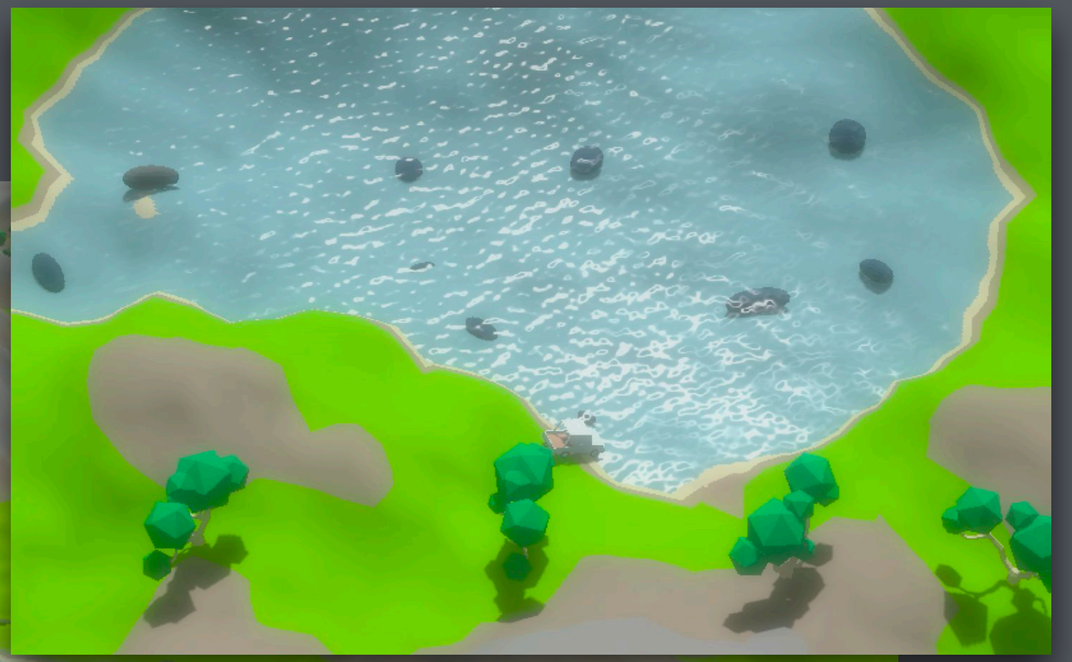
Natalie Diebold, Hani Altwaijry | *Portal-ish*



Victoria Dye, Joshua Reichler | *White-Out 2k17: The Snowening*


**Fight your way through the zombie fairies to rescue Orin!**

Ari Karo, Christopher Yu, Jonathan Behrens, Jeremy Cytryn | *Subterranean Arsonism*





Ryan Lefkowitz, Meredith Young-Ng, Erika Yu | *Open World Driver*

Daniel Konviser, Lily Lin, Sitian Chen | *Jump!*