Computer Graphics (cse167)
Introduction
Albert Chern
• Lecture (Albert Chern)
  ▶ Tue, Thu 11:00–12:20

• Discussion (Joseph Warmus, Guowei Yang)
  ▶ Wed 11:00–11:50 (starting from week2, 1/13)

• TAs
  ▶ Joseph Warmus        ▶ Guowei Yang

• Tutors
  ▶ Cynthia Butarbutar   ▶ Yihan Wang   ▶ Baichuan Wu

• Office hours: We will pin it on Piazza.
• Zoom lectures
  ▶ Ask questions via chat
  ▶ Keep muted unless asking/answering questions

• Zoom lectures are recorded
  ▶ If you need to rewatch, do it as soon as possible.
  ▶ If the recordings got automatically deleted (after 30 days),
     tell us on Piazza.
• 4 webpages
  ▶ Canvas (logistics, infos, links to slides & assignments)
  ▶ Gradescope (homework submissions)
  ▶ Piazza (Q&A)
  ▶ public site (slides, assignments)

http://cseweb.ucsd.edu/~alchern/teaching/cse167_2021/
• Rule
  ▶ Individual work
  ▶ Do not share your source code online (such as on your github).
  ▶ (see more on Canvas)
Overview of Computer Graphics

Computer Graphics Pipeline

What to expect in this course (see also Canvas)
What is Computer Graphics?

Jurassic Park 1993

Ratatouille 2007
Why Computer Graphics?

early computer (ENIAC) 1945

ENIAC I/O 1945

punched card 1890’s
Why Computer Graphics?

30% of the brain is devoted to visual processing
Most efficient way to receive information is in the form of visual data
Computer graphics
The use of computer to synthesize visual informations.
History of Graphics: Visual output

CRT monitors 1950’s–1960’s
“Sketchpad” – Ivan Sutherland 1963
Discovery of “solitons” in the Korteweg–de Vries (KdV) equation by Zabusky & Kruskal in 1965 while making the above film
History of Graphics: Math animation

Fully computer animated film
“Regular Homotopies in the Plane” 1974
History of Graphics: 1970’s

Arcade games 1970’s
- Raster (pixel) graphics
- Graphic processing unit (GPU)
- Realtime

Ed Catmull
- Z-buffering
- Texture mapping
- Subdivision

Lucasfilm Star Wars IV 1977
- Computer graphics in blockbusters
History of Graphics: Voyagers & Cosmos

Bump map for Venus 1978

Texture reconstruction for moons of Jupiter

V1 Jupiter flyby 1979

V2 Saturn flyby 1981

Cosmos “DNA” 1980

Cosmos “evolution” 1980

James Blinn
History of Graphics: 1980’s

Nintendo 1981

Macintosh 1984

PC graphics

LINKS-1 CG System 1983

Pixar *Luxo Jr.* 1986
History of Graphics: interface with GPU

Graphics API’s 1990s

- Simple to tell GPU to draw
- Hardcoded fixed function for transformation and interaction with lights.

Graphics API’s after mid 2000–10s

- Drawing stages are programmable.
- Flexible shading.
- General purpose parallel computing.
- We will learn this modern OpenGL programming.
Beyond controlling pixels

Physical simulations in entertainment
Beyond controlling pixels

Mathematical and Scientific Visualizations
Beyond controlling pixels

Data visualizations

New confirmed cases of Covid-19 in United States, New Zealand, Brazil, India, Pakistan and Mauritius

Seven-day rolling average of new cases, by number of days since 10 average cases first recorded

Brand Value (in billions of $)

Source: FT analysis of data from the European Centre for Disease Prevention and Control and the Covid Tracking Project. Data updated June 18, 2020 2:44pm EST
Beyond controlling pixels

Synthesize sound (not only visual information)

and touch senses
Beyond controlling pixels

3D printing

Constrained deformable geometry

architecture
Beyond controlling pixels

Industrial computer aided design

Fonts on your screen and on newspapers
Frontier of Computer Graphics

SIGGRAPH: Annual conference since 1974 with ~20k attendees per year.
Youtube search “SIGGRAPH technical paper trailer”
My research

Geometry processing & physical simulation

- Classical physics & geometry → Modern differential geometry & field theory

- Vortex represented as topological defects
- Deforming stiff material → Field optimization for local rotations
- Restricting dynamics to lower dimensions
This course

Foundation of Computer Graphics

Theory
- Geometry of our 3D space
- Representation of geometry
- Shading
- Radiometry, light transport
- Perception

System
- Rendering pipeline
- Parallel processing
- Modern OpenGL
buffer = data storage
shader = program running on GPU

Rendering pipeline

1. **Vertex array buffer**
   - Some description of 3D geometries
   - Ready to be projected onto an XY-screen

2. **Vertex shader**
   - Rasterize
     - Fragments (pixels from geometry)

3. **Texture buffer**
   - Texture

4. **Fragment shader**
   - Pixel colors

5. **Frame buffer**
Rendering pipeline

Some description of 3D geometries

Vertex shader

Ready to be projected onto an XY-screen

Vertex array buffer

Some description of 3D geometries

Rasterize

Fragments (pixels from geometry)

Frame buffer

Pixel colors

Fragment shader

Texture buffer

Texture

geometry ready-to-be projected

vertex shader
Ames Room

A seemingly normal room (right) can be the projection of a distorted room (left).
Rendering pipeline

Some description of 3D geometries

Vertex shader

Ready to be projected onto an XY-screen

Vertex array buffer

Some description of 3D geometries

Rasterize

Fragments (pixels from geometry)

Fragment shader

Pixel colors

Frame buffer

Texture buffer

Texture

“geometry in an Ames room”
Rendering pipeline

- Geometry ready-to-be projected
- Rasterize
- Texture buffer
- Fragment shader
- Frame buffer
- Vertex array buffer
- Vertex shader
- Rasterize
- Fragment shader
- Frame buffer

Some description of 3D geometries
Ready to be projected onto an XY-screen
Fragments (pixels from geometry)
Pixel colors
Rendering pipeline

- **Vertex array buffer**: Some description of 3D geometries
- **Vertex shader**: Ready to be projected onto an XY-screen
- **Rasterize**: Fragments (pixels from geometry)
- **Fragment shader**: Texture (per fragment, finalizing color)
- **Frame buffer**: Pixel colors

The process flow involves rendering, starting with the vertex array buffer, passing through the vertex shader, then rasterizing to create fragments, using the fragment shader to determine pixel colors, and finally resolving depth to create the frame buffer.
Rendering pipeline

**Vertex array buffer**
- Some description of 3D geometries
- Ready to be projected onto an XY-screen

**Vertex shader**

**Fragment shader**
- Texture buffer
- Texture
- Programmable
- Programmable
- Fragments (pixels from geometry)
- Pixel colors
- Frame buffer

**Rasterize**
We will explore

• Geometry of our 3D space
  ▶ Transformations
  ▶ Projective geometry

• Bring a virtual world to life
  ▶ OpenGL & GLSL
  ▶ Lights and shadows

• Vector graphics design
  ▶ Bézier curve, B-spline
  ▶ Vector graphic art

• Photorealistic art
  ▶ Ray tracing

• Additional topics
  ▶ Physically-based animations
  ▶ Radiometry
  ▶ Geometry processing
• Grade
  ▶ Weekly one-minute report (due every Friday): 10%
  ▶ HW0 (due 1/12): 5%
  ▶ HW1: 20%
  ▶ HW2: 25%
  ▶ HW3: 20% (+10%)
  ▶ HW4: 20% (+10%)
One-minute report

• Due every Friday (including this week)

Q1
1 Point

What are the few most exciting things you have learned this week?

Was there anything confusing?
• HW0 (5% due 1/12)
  ▶ HW0 is just make sure you can compile the skeleton code.
  ▶ In principle, compilation works out of the box.
  ▶ If it doesn’t compile, go to Piazza.