CSE 167 (FA21)
Computer Graphics
Albert Chern
Course Logistics
Lectures

• Lectures (Albert Chern)
  ▶ In-person or remotely

• Discussion (Joseph Warmus & Dylan Rowe)
  ▶ Fri 12:00–12:50 in-person or remotely
    (starting from Week1: 10/1)

• TAs & Tutors
  ▶ Joseph Warmus
  ▶ Baichuan Wu
  ▶ Dylan Rowe
  ▶ Edward Xie
CSE 167 (FA 2021) Computer Graphics


- **Lecture:** Mon Wed Fri 15:00 - 15:50
- **Discussion Session:** Fri 12:00-12:50
- **Classroom:** Center Hall 214. You can also participate the class via Zoom.

- **Instructor:** Albert Chern (office hour (OH) Tue 15:30–16:30 CSE B270A)
  - **TA:** Joseph Warmus (OH Tue 14:30–15:30 CSE B270A)
  - **TA:** Dylan Rowe (OH Wed 12:00–13:00 CSE B270A)
  - **Tutor:** Baichuan Wu (OH Wed 10:00–11:00 CSE B240A)
  - **Tutor:** Edward Xie (OH hour Mon 13:00–14:00 CSE B270A)

- **Sites:**
  - **This page:** Slides, lecture notes, HW
  - **Canvas:** Link to Zoom (and Zoom recordings), link to Gradescope, and link back to this page.
  - **Piazza:** piazza.com/ucsd/fall2021/cse167 Q&A forum for the class.
  - **Gradescope:** HW submission.

- **Lecture note:** Introduction to Computer Graphics

Visit Canvas (for all info including Zoom links), Gradescope (HW submission) and Piazza (Q&A).
CSE 167 (FA 2021) Computer Graphics


- **Lecture**: Mon Wed Fri 15:00 - 15:50
- **Discussion Session**: Fri 12:00-12:50
- **Classroom**: Center Hall 214. You can also participate the class via Zoom.
- **Instructor**: Albert Chern (office hour (OH) Tue 15:30–16:30 CSE B270A)
  - **TA**: Joseph Warmus (OH Tue 14:30–15:30 CSE B270A)
  - **TA**: Dylan Rowe (OH Wed 12:00–13:00 CSE B270A)
  - **Tutor**: Baichuan Wu (OH Wed 10:00–11:00 CSE B240A)
  - **Tutor**: Edward Xie (OH hour Mon 13:00–14:00 CSE B270A)
- **Sites**:
  - **This page**: Slides, lecture notes, HW
  - **Canvas**: Link to Zoom (and Zoom recordings), link to Gradescope, and link back to this page.
  - **Piazza**: piazza.com/ucsd/fall2021/cse167 Q&A forum for the class.
  - **Gradescope**: HW submission.
- **Lecture note**: Introduction to Computer Graphics

Visit Canvas (for all info including Zoom links), Gradescope (HW submission) and Piazza (Q&A).
Gradescope

• From Canvas

CSE 167 - Computer Graphics - Chern [FA21]

CSE 167 Computer Graphics.
This course provides an introduction to computer graphics – the use of comp
synthesize visual informations, or more specifically, the process of converting
representations of the geometric data to sets of pixel colors that best visualiz
virtual world.

Main Course Webpage
The main course webpage is: https://cseweb.ucsd.edu/~alchem/teaching/cse
which contains the class logistics, homework, syllabus, lecture note, etc.

We will also be using Piazza (sign-up & access link https://e
piazza.com/ucsd/fall2021/cse167 ) as the Q&A forum.

The assignments, weekly one-minute reports, final project etc are submitted t
Gradescope with link on the side menu of this Canvas site.

You can join the lecture remotely with Zoom link on the side menu of this Car

• From gradescope.com

login with School credential

CSE167_FA21_A00
CSE 167 - Computer Graphics - Chern [FA21]

Dashboard

Entry Code: RW3XEG

info available also on Piazza
References
https://learnopengl.com

about transforming all 3D coordinates to 2D pixels that fit on your screen. The process of transforming 3D coordinates to 2D pixels is managed by the graphics pipeline of OpenGL. The graphics pipeline can be divided into two large parts: the first transforms your 3D coordinates into 2D coordinates and the second part transforms the 2D coordinates into actual colored pixels. In this chapter we'll briefly discuss the graphics pipeline and how we can use it to our advantage to create fancy pixels.

The graphics pipeline takes as input a set of 3D coordinates and transforms these to colored 2D pixels on your screen. The graphics pipeline can be divided into several steps where each step requires the output of the previous step as its input. All of these steps are highly specialized (they have one specific function) and can easily be executed in parallel. Because of their parallel nature, graphics cards of today have thousands of small processing cores to quickly process your data within the graphics pipeline. The processing cores run small programs on the GPU for each step of the pipeline. These small programs are called shaders.

Some of these shaders are configurable by the developer which allows us to write our own shaders to replace the existing default shaders. This gives us much more fine-grained control over specific parts of the pipeline and because they run on the GPU, they can also save us valuable CPU time. Shaders are written in the OpenGL Shading Language (GLSL) and we'll delve more into that in the next chapter.

Below you'll find an abstract representation of all the stages of the graphics pipeline. Note that the blue sections represent sections where we can inject our own shaders.
Grades

- No quiz or exam
- Weekly one-minute report 10% (due every Friday starting 10/1)

Q1

1 Point

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

Was there anything confusing?
Grades

- No quiz or exam
- Weekly one-minute report 10% (due every Friday starting 10/1)
- HW 0, 1, 2, 3, 4 (65%)
  - Individual work (don’t share the code online)
- Final project: 25%
  - In a group of 2 people or 1 person (finalize grouping in Week 6)
- Passing grade: 70%
- Curve letter grade condition: 3/4 of the class complete the CAPE course evaluation in Week 10.
Prerequisite and some expectation

• Experience with
  ▶ basic linear algebra (matrix & vector)
  ▶ C++

• You can expect that you will
  ▶ Deal with lots of floating point numbers (continuous math)
  ▶ Think of problems geometrically
  ▶ View math operators “structurally”
  ▶ See some physics (optics, mechanics)
Computer Graphics
• What is computer graphics (brief history)?
• Topics of this course
• Two 3D computer graphics paradigms: Rasterization v.s. Ray tracing
What is Computer Graphics?
What is Computer Graphics?

Jurassic Park 1993

Ratatouille 2007
Why Computer Graphics?

punched card 1890’s

early computer (ENIAC) 1945

ENIAC I/O 1945
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.
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

3DCG: Ed Catmull & others in Utah
- 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

V2 Saturn flyby 1981

V1 Jupiter flyby 1979

Cosmos “DNA” 1980

Cosmos “evolution” 1980

James Blinn
History of Graphics: 1980’s

Nintendo 1981

LINKS-1 CG System 1983

Macintosh 1984

PC graphics

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

Graphics API’s 1990s
- Simple to tell GPU to draw
- Hardcoded fixed function for drawing (fast but not flexible).

Graphics API’s after mid 2000–10s
- Drawing stages are programmable.
- Flexible shading.
- General purpose parallel computing.

New era of graphics 2020
- Realtime photorealism
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

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 BST
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
SIGGRAPH: Annual conference since 1974 with ~20k attendees per year.

Youtube search “SIGGRAPH technical paper trailer”
My research

Geometry processing & physical simulation

- Find underlying geometric structures to make simulations easier
Topics of this course
This course

Modern OpenGL
• Command the graphics card

Foundation of 3D Computer Graphics
• Convert geometries in a 3D scene into pixel colors in a 2D screen.

Foundation of Vector Graphics
• Build smooth geometries from only a few control points
Additional topics

Perception of color
• Physical color, displayed color, perceived color

Physics-based animation
• Behind the scenes of special effects.

Optics
• Light transport equation

Geometry processing
• Differential geometry of discrete meshes
This course

Modern OpenGL
• Command the graphics card

Foundation of 3D Computer Graphics
• Convert geometries in a 3D scene into pixel colors in a 2D screen.

Foundation of Vector Graphics
• Build smooth geometries from only a few control points
This course

Modern OpenGL
• Command the graphics card

Foundation of 3D Computer Graphics
• Convert geometries in a 3D scene into pixel colors in a 2D screen.

Foundation of Vector Graphics
• Build smooth geometries from only a few control points
Foundation of 3D Computer Graphics

- Convert geometries in a 3D scene into pixel colors in a 2D screen.
  
  ▶ Rasterization
  
  ▶ Ray tracing
Rasterization v.s. Ray tracing
Main task in 3D computer graphics

**Task** Convert geometric elements in a scene into pixel colors in a 2D screen.

**Step 1** Determine which geometric elements correspond to which pixels.  
**Step 2** Color the pixel according to the scene information (lighting, material, orientation).
Scene–Screen Incidence Relationship

Task
Determine which triangles of the scene are incident to which pixels of the screen.

Method 1
for each triangle in scene
  for each pixel in screen
    output the pair if the triangle occupies that pixel.
  end for
end for
Scene–Screen Incidence Relationship

**Task**
Determine which triangles of the scene are incident to which pixels of the screen.

**Method 1**

```plaintext
for each triangle in scene
for each pixel in screen
    output the pair if the triangle occupies that pixel.
end for
end for
```

**Method 2**

```plaintext
for each pixel in screen
for each triangle in scene
    output the pair if the triangle occupies that pixel.
end for
end for
```
Task
Determine which triangles of the scene are incident to which pixels of the screen.

**Rasterization**

```plaintext
for each triangle in scene
  for each pixel in screen
    output the pair if the triangle occupies that pixel.
  end for
end for
```

**Ray tracing (a.k.a. ray casting)**

```plaintext
for each pixel in screen
  for each triangle in scene
    output the pair if the triangle occupies that pixel.
  end for
end for
```

*fragment*
We are constructing a mapping between scene and screen.
Scene–Screen Incidence Relationship

**Rasterization**

```
for each triangle in scene
    for each pixel in screen
        output the pair if the triangle occupies that pixel.
    end for
end for
```

- Hardcoded in every GPU
- Realtime
- Approximately physical

**Ray tracing (a.k.a. ray casting)**

```
for each pixel in screen
    for each triangle in scene
        output the pair if the triangle occupies that pixel.
    end for
end for
```

- Implement by ourselves in CPU
- Offline
- Physics-based
Scene–Screen Incidence Relationship

Rasterization

- Hardcoded in every GPU
- Realtime
- Approximately physical

Ray tracing (a.k.a. ray casting)

- Implement by ourselves in CPU
- Offline
- Physics-based
Next week

- Next week, we will start working with OpenGL.
- OpenGL is a rasterizer.
- HW0 is due 10/1
  - Just compile, run, and upload the result (should look like this Fig).
  - Compilation problem: office hour (see TA/tutor’s platform) or piazza.
  - Next week, we explain what happens in the code.