Course Logistics
• Lectures (Albert Chern)
  ▶ Center Hall 115, In-person or remotely

• Discussion (by the TAs)
  ▶ Fri 18:00–18:50, Center Hall 119
    (starting from Week1: 9/30)

• TAs
  ▶ Peter Wu
  ▶ Sina Nabizadeh
  ▶ Abhinav Gupta
  ▶ Dylan Rowe

• Tutors
  ▶ Flynn Sui
  ▶ Evan Yao

- **Lecture**: Mon Wed Fri 15:00 - 15:50. Center Hall 115.
- **Discussion Session**: Fri 18:00-18:50. Center Hall 119.
- **Classroom**: You can also participate the class via Zoom.
- **Sites**:
  - **This page**: Slides, lecture notes, HW
  - **Canvas**: Link to Zoom (and Zoom recordings), link to Gradescope, and link back to this page.
  - **Piazza**: https://piazza.com/ucsd/fall2022/cse167 Q&A forum for the class.
  - **Gradescope**: https://www.gradescope.com/courses/444876 (log in with School credential, entry code: K3N5V7) HW submission.
- **Lecture note**: Introduction to Computer Graphics
  Visit Canvas (for all info including Zoom links), Gradescope (HW submission) and Piazza (Q&A).
<table>
<thead>
<tr>
<th>Instructor</th>
<th>Office hour</th>
<th>OH location (all in CSE Building (EBU3B))</th>
<th>OS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Instructor</strong></td>
<td>Albert Chern</td>
<td>Tue 3pm–4pm</td>
<td>4112</td>
</tr>
<tr>
<td><strong>TA</strong></td>
<td>Abhinav Gupta</td>
<td>Wed &amp; Fri 8:30am–9:30am</td>
<td>B270A</td>
</tr>
<tr>
<td><strong>TA</strong></td>
<td>Mohammad &quot;Sina&quot; Nabizadeh</td>
<td>Tue 1pm–2pm</td>
<td>B215</td>
</tr>
<tr>
<td><strong>TA</strong></td>
<td>Dylan Rowe</td>
<td>Mon 10am–12pm</td>
<td>B275</td>
</tr>
<tr>
<td><strong>TA</strong></td>
<td>Baichuan &quot;Peter&quot; Wu</td>
<td>Thu 2pm–3pm &amp; Fri 10am–11am</td>
<td>B240A</td>
</tr>
<tr>
<td><strong>Tutor</strong></td>
<td>Ziyan &quot;Flynn&quot; Sui</td>
<td>Fri 2pm–3pm</td>
<td>B215</td>
</tr>
<tr>
<td><strong>Tutor</strong></td>
<td>Yunchao &quot;Evan&quot; Yao</td>
<td>Mon 5pm–6pm</td>
<td>B240A</td>
</tr>
</tbody>
</table>
References

REAL-TIME RENDERING
FOURTH EDITION

Tomas Akenine-Möller
Eric Haines
Naty Hoffman
Angelo Pesce
Michal Iwanicki
Sébastien Hillaire

(Available online from UC Library)
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 written exercise 40%  
  
  individual work
- Programming HW 60%
  - HW0–4: 40%  
    individual work
  - Final HW: 20%  
    group of 1 or 2

- Passing grade: 70%

- Curve letter grade condition: 3/4 of the class complete the CAPE course evaluation in Week 9,10.
Prerequisite and some expectation

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

• **You can expect that you will**
  ▶ Command and run programs on GPUs (using OpenGL)
  ▶ Deal with lots of floating point numbers (continuous math)
  ▶ Think of problems geometrically
  ▶ View math operators “structurally”
  ▶ See some physics (optics, mechanics)
Getting Started
Getting started

• Next week, we will start working with **OpenGL**

• HW0 & Exer1 are due 9/30
  ▶ Compile, run, and upload the result (should look like this Fig).
  ▶ Modify the code to get the cat/fox shape
  ▶ Compilation problem: office hour (see TA/tutor’s platform) or piazza.
  ▶ Next week, we explain what happens in the code.
Computer Graphics
Computer Graphics

- What is computer graphics? (brief history)
- Topics of this course
- How to draw programmatically? (computer graphics pipeline)
What is Computer Graphics?
What is Computer Graphics?

Jurassic Park 1993

Ratatouille 2007
Why Computer Graphics?

Early computer (ENIAC) 1945

ENIAC I/O 1945

Punched card 1890’s
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
History of Graphics: Visual input

“Sketchpad” – Ivan Sutherland 1963
Discovery of “solitons” in the Korteweg–de Vries (KdV) equation by Zabuksy & Kruskal in 1965 while making the above film.
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

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

LINKS-1 CG System 1983

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

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”
Our 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
3D Computer Graphics

Foundation of 3D Computer Graphics

• Convert geometries in a 3D scene into pixel colors in a 2D screen.

• How to draw pictures algorithmically?
  ▶ Rasterization v.s. Ray tracing
  ▶ Graphics pipeline
  ▶ Hardware: GPU