Lecture 1: Introduction and Overview
Welcome!

Logo created by Junqiu Zhu
Instructor

• Lingqi Yan
  - Assistant Professor @ UCSB
  - Web: www.cs.ucsb.edu/~lingqi/
    Email: lingqi@cs.ucsb.edu
  - Research: Rendering in Computer Graphics
  - Hobbies: research, video games, piano, NBA, traveling, etc.
Instructor’s Achievements

2019: ACM SIGGRAPH Outstanding Doctoral Dissertation Award

2019: Oscar Nominee for Best Visual Effects

2019: six APEX Champions in one evening
Course Staff

• Teaching Assistants
  - 万健洲 (wanjianzhou@qq.com)
  - 周锦超 (zhoujinchao@buaa.edu.cn)
  - 邓俊辰 (junchendeng@gmail.com)

• More will be recruited from current students (based on need)
About this Course
What is GAMES202 about?

Real-Time High Quality Rendering

Intermediate level — connecting basic knowledge and research
What is GAMES202 about?

● **Real-Time** High Quality Rendering
  - Speed: more than 30 FPS (frames per second), even more for Virtual / Augmented Reality (VR / AR): 90 FPS
  - Interactivity: Each frame generated on the fly

● **Real-Time High Quality** Rendering
  - Realism: advanced approaches to make rendering more realistic
  - Dependability: all-time correctness (exact or approximate), no tolerance to (uncontrollable) failures
What is GAMES202 about?

• Real-Time High Quality Rendering
  - What is Rendering?

3D scene (meshes, lights, etc.)

Calculating light -> eye

Image
What is GAMES202 about?

- Highest level: 4 different parts on real-time rendering

Shadows (and env)  
Global Illum. (Scene/image space, precomputed)

Physically-based Shading  
Real-time ray tracing
Course Topics

• Shadow and Environment Mapping

Real-Time, All-Frequency Shadows in Dynamic Scenes
Annen et al.
Course Topics

• Interactive Global Illumination Techniques
Course Topics

• Precomputed Radiance Transfer

Global Illumination with Radiance Regression Functions
Ren et al.
Course Topics

- Real-Time Ray Tracing
Course Topics

- Participating Media Rendering, Image Space Effects, etc.

Single scattering

Image space reflection
Course Topics

• Non-Photorealistic Rendering
  - But will not be in depth / per game

Genshin Impact

Animal Crossing: New Horizons
Course Topics

- Antialiasing and supersampling
Course Topics

- Chatting about techs!
Course Topics

- Chatting about games!

Golf II
The Last of Us Part II
(2020 Game of the Year)

Monster Hunter Rise
(2021)
What is GAMES202 NOT about?

- 3D modeling or game development using Unreal Engine (where can I learn them?)

Modeling character animation in Maya

[CgRecord](http://tutorials.cgrecord.net/2017/08/17-minute-animation-process-in-autodesk.html)

CSGO PoV Cam set up in Unreal Engine

[YouTube](https://www.youtube.com/watch?v=3TQ18SmQSw0)
What is GAMES202 NOT about?

- Expensive (but more accurate) light transport techniques in movies / animations (where can I learn this?)

Manifold Metropolis Light Transport
Jakob et al.

Gradient Domain Path Tracing
Kettunen et al.
What is GAMES202 NOT about?

• Neural Rendering

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis
[Mildenhall et al.]
What is GAMES202 NOT about?

• Using OpenGL
• Scene / shader optimization
• Reverse engineering of shaders
• High performance computing e.g. CUDA programming
Questions?
How to study GAMES202?

• Understand the difference between science and technology
  - Science != technology
  - Science == knowledge
  - Technology == engineering skills that turn science into product

• Real-time rendering = fast & approximate offline rendering + systematic engineering

• Fact: in real-time rendering technologies, the industry is way ahead of the academia

• Practice makes perfect
How to study GAMES202?

• If you are watching live streams of this course
  - Be active asking questions!

• If you are watching recordings
  - 1.25x - 1.5x playback speed is recommended!
Why study GAMES202?

Computer Graphics is AWESOME!
Course Logistics
Prerequisites

• **Strong interest** in Rendering, Graphics

• Computer Graphics experience
  - GAMES101 or equivalent
  - Basic calculus

• What else to be prepared?
  - Basic OpenGL Shader Language (GLSL)!
  - The next lecture will briefly review it
  - Assignment 0 (optional, will be released with the next lecture) will help you warm up quickly
General Information

• Modern Course
  - Comprehensive but no need to have extreme hardware!
  - Pace / contents subject to change

• Course Website
  - Has all the needed information
  - Syllabus, slides, reading materials, etc.
References

• No Required Textbooks
  - Reading materials (if any) will available online before lectures
  - Mainly SIGGRAPH courses + engine design docs
  - Lecture slides will be available after class

• Possible reference
  - Still not quite related (unlike the tiger book to GAMES101)
Q & A

• Sign up on our BBS for discussion (http://games-cn.org/forums/forum/games202/)

• And QQ group!
Assignments

• Assignments
  - 5 programming assignments (excluding assignment 0)
  - About 1.5 week for each assignment
  - Language: OpenGL Shader Language (GLSL)
  - Code skeleton will be provided (Javascript + WebGL)

• Submission
  - Submit your project by 11:59PM AoE (Anywhere on Earth) on/before the due dates
  - Feedback will be provided in a week
Assignments

- A quick look at our assignment skeleton
Assignments

• Assignment Submission Website (http://smartchair.org/GAMES202)

• No Exams

• Course Project / Final Project
  - Starting midway of this course
  - References will be provided, but you decide the topic
  - Best work will be posted online for showing off
No Need to Use An IDE!

• IDE: Integrated Development Environment

• An IDE Helps you parse a entire project
  - And gives hints on syntax / usages of member functions, etc.

• Since you’ll be focusing on writing shaders most of the times in this course
  - No need to use an IDE this time
  - A text editor is perfectly fine
    - Sublime Text, Vi / Vim, Emacs, etc.
  - Online text editors are also great
Academic integrity

- Work alone for regular assignments
  - no copy-pasting from any other sources
- Do not publish your code (on Github, etc.) for assignments using our skeleton code
- Do not post your solution online
  - Discussion / explanation is welcomed
Questions?
Today’s Lecture
Outline

• Motivation

• Evolution of real-time rendering

• Technological and algorithmic milestones
  - Programmable graphics hardware
  - Precomputation-based methods
  - Interactive Ray Tracing
Motivation

Today, Computer Graphics is able to generate photorealistic images

- Complex geometry, lighting, materials, shadows
- Computer-generated movies/special effects (difficult or impossible to tell real from rendered…)
Motivation

• But accurate algorithms (esp. ray tracing) are very slow
  - So they are called offline rendering methods
  - Remember how long it takes to render 1 frame in Zootopia?
Motivation

- With proper approximations, we can generate plausible results but runs much faster.

Toyota 2000GT, from TurboSquid (offline rendering)

Final Fantasy XV (real-time rendering)
Evolution of Real-Time Rendering

- Interactive 3D graphics pipeline as in OpenGL
  - Earliest SGI machines (Clark 82) to today
  - Most of focus on more geometry, texture mapping
  - Some tweaks for realism (shadow mapping, accum. buffer)

SGI Reality Engine 93
(Kurt Akeley)
Evolution of Real-Time Rendering

• 20 years ago
  - Interactive 3D geometry with simple texture mapping, fake shadows (OpenGL, DirectX)

Final Fantasy VII (1997)  
Counter Strike (1999)
Evolution of Real-Time Rendering

• 20 -> 10 years ago
  - A giant leap since the emergence of programmable shaders (2000)
  - Complex environment lighting, real materials (velvet, satin, paints), soft shadows

Assassin’s Creed II (2009)  DARK

Resident Evil 5 (2009)  OILY
Evolution of Real-Time Rendering

• Today
  - “Stunning graphics”

God of War (2018)
Evolution of Real-Time Rendering

• Today
  - Extended to Virtual Reality (VR) and even movies

Beat Saber, VR Game

Zafari, animation series rendered completely using Unreal game engine
Evolution of Real-Time Rendering

• Today
Evolution of Real-Time Rendering

- Today
Evolution of Real-Time Rendering

• In the future

The Matrix (1999 movie)
Evolution of Real-Time Rendering

- In the future

Ready Player One (2018 movie)
Technological and Algorithmic Milestones

- Programmable graphics hardware (shaders) (20 years ago)

A New Dawn demo, NVIDIA
https://www.geforce.com/games-applications/pc-applications/a-new-dawn/videos
Technological and Algorithmic Milestones

- Programmable graphics hardware (shaders)

![Diagram of graphics pipeline with nodes labeled for vertices, primitive operations, scan conversion, texture memory, fragment operations, and framebuffer. Each node is color-coded with notes indicating programmability in modern GPUs. Image courtesy of Prof. Ravi Ramamoorthi.]
Technological and Algorithmic Milestones

• Precomputation-based methods (15 years ago)
  - Complex visual effects are (partially) **pre-computed**
  - Minimum rendering cost **at run time**

Image courtesy of Prof. Ravi Ramamoorthi
Technological and Algorithmic Milestones

- Precomputation-based methods

All-Frequency Rendering of Dynamic, Spatially-Varying Reflectance
Wang et al.
Technological and Algorithmic Milestones

• Precomputation-based methods: Relighting
  - Fix geometry
  - Fix viewpoint
  - Dynamically change lighting

[Ng, Ramamoorthi, Hanrahan 04]
Technological and Algorithmic Milestones

- Interactive Ray Tracing (8-10 years ago: CUDA + OptiX)
  - Hardware development allows ray tracing on GPUs at low sampling rates (~1 samples per pixel (SPP))
  - Followed by post processing to denoise

Car interactively rendered using NVIDIA OptiX

Pixar’s real-time previewer
Questions?
Next Lecture

- A swift and brutal recap of some important concepts
  - Graphics Pipeline
  - Shader Language
  - Rendering Equation
  - Calculus
  - etc.
Thank you!