There Can Be (at least) Two Introductory Graphics Courses: Teaching Introduction to Non-Interactive Computer Graphics

Michael D. Shah
Northeastern University
Boston, Massachusetts, USA
mi.shah@northeastern.edu

ABSTRACT

For decades many colleges and universities offering an introductory to computer graphics course have been teaching computer graphics through a single course focusing on real-time computer graphics. Many of these real-time or interactive computer graphics courses have a focus on graphics in the gaming domain, and also require students to be proficient in at least one instructor-selected language (e.g. C++, Java, JavaScript) through the schools introductory sequence. In this work, we describe motivations to a 'second' alternative and complementary introductory graphics course focused on non-interactive computer graphics (i.e. rendering images for movies) that can be taken at nearly any point in the curriculum after the first programming course. We have observed that this additional course may better foster a more diverse pool of students to garner interest in computer graphics. This non-interactive graphics course is taught with a focus on rendering (using path tracing), uses free resources, and is programming language-agnostic.

ACM Reference Format:
Michael D. Shah. 2024. There Can Be (at least) Two Introductory Graphics Courses: Teaching Introduction to Non-Interactive Computer Graphics. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Educator's Forum (SIGGRAPH Educators Forum '24), July 27 - August 01, 2024.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3641235.3664429

1 INTRODUCTION

Introductory computer graphics course in computer science programs at many colleges and universities often focus on real-time graphics using a variant of Direct3D, OpenGL, Vulkan, WebGL, or WebGPU. Pragmatically, this prepares students for a variety of careers in domains in gaming, VR, simulation, and visualization to name a few. We browsed the top 50 universities and found that while most all had real-time rendering or interactive computer graphics courses, there existed a scarce few alternative introductory points to the field of graphics [Berger 2023]. We additionally analyzed the introductory courses at SIGGRAPH and SIGGRAPH Asia (from 1996 to 2022) going back in the archives to see if there were areas of emphasis for introductory topics in programming computer graphics. In total we found 6 introductory courses on WebGL/three.js and 18 courses introducing OpenGL [Cou 2023; Angel and Shreiner 2022]. We however, did not find any introductory material on 'ray tracing' or 'path tracing' that specifically introduced students to computer graphics (without having prior knowledge of computer graphics) until 2022 (note: we did find 'Introduction to DirectX Raytracing' and 'Introduction to Real-Time Ray Tracing', though these are not targeted at newcomers to computer graphics). In 2022, there was 1 SIGGRAPH course on 'WebGPU and Raytracing' found that could form the basis of an introductory course for motivated students or otherwise inspire instructors to develop introductory content around raytracing [Web 2023]. Shene postulated work nearly 20 years earlier using a raytracer and geometry to teach graphics [Shene 2002]. Given the recent industry trends in computer graphics regarding advancements in raytracing [Shirley et al. 2008] (and inspiration with other pathways to interest students in graphics using simulation and VR [Marai 2009; Vasylevska et al. 2017].), we propose a new pathway to expose students to become interested in computer graphics. In this work you will find our philosophies, syllabus, and student feedback from four iterations of our new course focused on 'non-interactive' computer graphics focusing on rendering with raytracing.

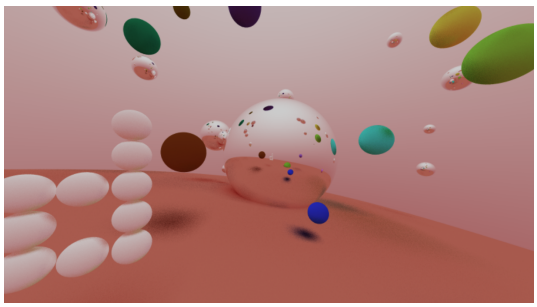

Figure 1: The course on Non-Interactive Computer Graphics focuses on implementation of raytracers (specifically path tracers) in order to allow a diverse pool of students to try computer graphics in a programming language-agnostic manner. The above image represents an image a student can generate by the third week of the course with primitive shapes, shadows, multiple materials, and reflections.

2 COURSE PHILOSOPHY

While there have been attempts in teaching computer graphics in learner-friendly environments such as GameX [Hoetzlein and Schwartz 2005], we believe there should exist other introductory courses that do not hide behind abstraction layers (e.g. a game engine or Object-Oriented codebase) to foster student interest while being introduced to the field of graphics. Our course has been

SIGGRAPH Educators Forum '24, July 27 - August 01, 2024, Denver, CO, USA

ACM ISBN 979-8-4007-0517-5/24/07.
https://doi.org/10.1145/3641235.3664429

offered four times successfully with this approach and we have a few core philosophies that inspired the course.

(1) **Philosophy 1:** Students may choose the programming language they use (i.e. language and framework agnostic).
(2) **Philosophy 2:** Students should generate something visual as soon as possible (i.e. on day 1 of the course), while avoiding complicated setups (i.e. no need for multiple frameworks such as glew, glfw, etc.)
(3) **Philosophy 3:** This programming course could be taken in parallel, in place of, or otherwise before or after a traditional interactive real-time graphics course. The course should truly be an alternative introduction to graphics, but need not be 'lesser' in rigor or status.
(4) **Philosophy 4:** Resources (e.g. expensive graphics hardware) and books should be freely accessible to students to prevent barrier to entry.
(5) **Philosophy 5:** This course should cover fundamentals of relevant mathematics, likely a similar subset of trigonometry and linear algebra in a traditional interactive computer graphics course. Students who end up taking both courses benefit by seeing math tools applied in different framings.
(6) **Philosophy 6:** The course should attempt to motivate students to pursue industry or research by introducing a breadth of applications where the graphics, mathematics, and art skills are applied (e.g. not just games, but also inclusive of art, mathematics, GPGPU, Modeling, and movie production).

3 COURSE FORMAT

Provided in Table 1 is an example of the modules that we teach. We primarily teach the first 10 modules from the freely available Peter Shirley textbooks [Shirley 2018, 2023]. We occasionally also use other curated web articles such as Scratchapixel [Scr 2023]. Later modules may utilize freely distributed books such as the Ray Tracing Gems series for students final projects [Haines and Akenine-Möller 2019; Marrs et al. 2021].

3.1 Course Programming Assignments

Provided below is a listing and brief objective of assignments in our course. Each assignment builds upon the other assignment allowing the students to build a portfolio project of significance. We have received feedback from students that each assignment adds just enough complexity to build a significant product in the end, but the assignments are not so big that a refactor would cause students immense panic. Students also reported enjoying the opportunity to think about code architecture in their chosen language as well.

(1) Basic ray tracer - Generate spheres with reflections and shadows that are output to a PPM image format.
(2) Camera and Materials - Perspective rendering from multiple viewpoints of spheres of different materials.
(3) Textures - Explore procedural texture generation and different materials.
(4) Triangles - Load and path trace triangle-based .obj meshes.
(5) Optimization - Use bounding boxes to speed up rendering.
(6) Introduction to GPU Compute with Path tracers - Explore CPU and GPU's parallel abilities for accelerating path tracers.

Module	Title
1	Course Introduction and Image Generation
2	A Ray Tracer
3	Math, Camera, Shading, and Materials
4	Code Architecture and Performance
5	Matrices and Moving Cameras
6	Textures
7	Ray-Triangle Intersection and Geometric Data
8	Motion Blur and Acceleration Data Strategy
9	Lighting and Shading
10	Parallelization
11	Hybrid Ray Tracers
12	Photon Mapping and Other Topics
13	Instructor Choice
14	Final Project Presentations

Table 1: A listing of each module in a semester long course consisting with one module per week for 3 hours and 15 minutes. Rows 10-13 could be omitted in a quarter long course.

(7) The final projects for this course are left up to students to explore a topic of their interest. Students have previously implemented rendering techniques involving: animation, volumetric lighting, emissive materials, physically-based rendering, integrating their raytracers into Blender3D, and building real-time ray tracers for those with OpenGL experience. The final product is a 3-minute YouTube video showing their code, discussing their design decisions, and ultimately some images from the ray tracer (See Figure 2).

4 RESULTS

We have found that this course has drawn from a more diverse population of undergraduate and graduate students who would not otherwise pursue an introductory to computer graphics course typically targeted at our game development majors. Some direct feedback is provided below.

- *"I thought this was a really interesting course! I had a fun time building the ray tracer. I would say that linear algebra is really not my strong suit, but I still feel that I learned a lot and had a fun and learning time seeing it applied in the form of computer graphics." - Summer '22 (Art+CS Student)*
- *"It was incredible to see how even the simplest raytracing code can produce realistic lighting and images that far exceed the rasterization techniques worked on in the previous course." - Summer '21 (CS+Math Student)*

5 CONCLUSIONS AND FUTURE WORK

In this work we presented our core philosophies and outline of a graphics course that does not focus on a traditional real-time (i.e. 'introduction to OpenGL/Vulkan/Direct3D' API) approach, but instead on path tracers for rendering. From student feedback, this approach is programming language-agnostic, less intimidating to setup, and has fostered interest from a variety of student majors who are not solely computer science majors interested in games.