Computer Graphics

Guide to AI and Data Courses · Part III: Signals, Images and Audio

Transformations | Graphics pipeline | Lighting | Ray tracing | Animation | Neighboring courses

Sercan KÜLCÜ · Giresun University · Computer Engineering

© Sercan Külcü. All rights reserved. · Interactive HTML version of these slides: https://sercankulcu.github.io/files/ai_courses/slides/Chapter_22_Computer_Graphics.html

INTRODUCTION

What Question Does It Ask?

> **How is a realistic image produced from the description of a scene?**

Computer graphics studies how to **render an image** from a scene described by geometry, materials, lights and a camera. It is the inverse of computer vision: vision recovers the scene from an image, graphics produces the image from the scene.

At a glance

APPROACH	INPUT	OUTPUT
geometry and light simulation	scene: models, lights, camera	image or animation

SUCCESS MEASURE	MATH LOAD	DATA NEEDS
realism, frame rate	high: linear algebra, geometry	none

A Short History

1963
Sutherland's Sketchpad shows drawing on screen with a light pen; interactive graphics begins.

1975
Phong publishes his reflection model and shading method; shiny surfaces look real.

1980
Whitted's ray tracing unifies reflection, refraction and shadows in one method.

1986
Kajiya publishes the rendering equation: the basis of physically based rendering.

1995
Toy Story becomes the first fully computer-generated feature film.

2018
Hardware ray tracing reaches consumer GPUs; real-time ray tracing begins.

THE COURSE

Core Concepts

Homogeneous coordinates
Writing translation, rotation and scaling as 4×4 matrices.

Model-view-projection
The chain of transforms from object space to screen space.

Triangle mesh
Representing surfaces with vertices and triangles.

Rasterization
Finding the pixels a triangle covers.

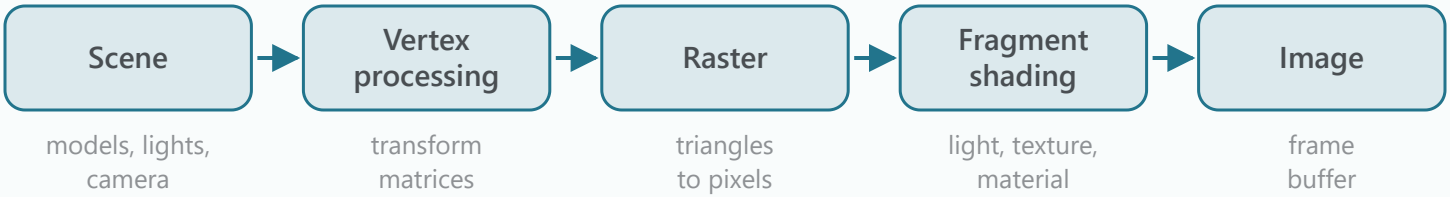
Z-buffer
Visibility by keeping the nearest surface per pixel.

Lighting model
Ambient, diffuse and specular: Phong, Blinn-Phong.

Texture mapping
Wrapping an image or pattern on a surface; UV coordinates.

Ray tracing
Shooting rays from the camera and tracing light paths backward.

A Typical Workflow



Example: In a game, a character's triangle mesh is projected to the screen by the world and camera matrices, rasterization finds the pixels each triangle covers, and a shader computes each pixel's color from light and texture. This repeats 60 times a second.

THE COURSE

What Does the Course Cover?

1. Raster images, colour and gamma
2. Vectors, matrices and 2D/3D transformations
3. Viewing: camera, projection, view volume
4. Triangle meshes and surface representation
5. Rasterization and visibility with the z-buffer
6. Shading: Phong and Blinn-Phong
7. Texture mapping, UV coordinates and mipmaps
8. The graphics pipeline and GPU shaders
9. Ray tracing and acceleration structures (BVH)
10. Curves and surfaces: Bezier, B-splines
11. The rendering equation and Monte Carlo path tracing
12. Animation: keyframes, skeletons, physics simulation

Skills you will gain

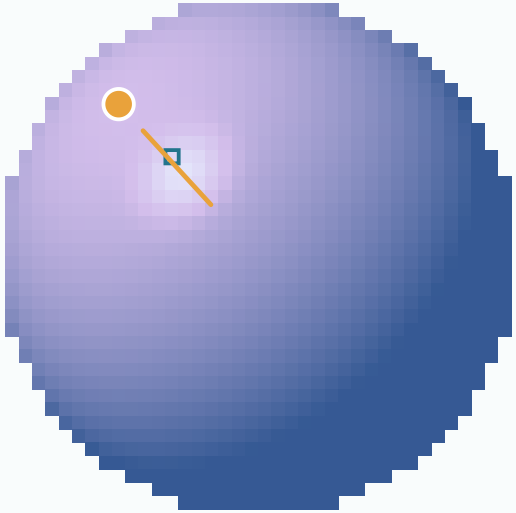
drawing with OpenGL or WebGL | writing GLSL shaders | building 3D scenes with three.js | writing a simple ray tracer | preparing models and scenes in Blender

A typical semester; the order and coverage vary between universities.

TRY IT

Shading a Sphere with Light

Every frame, a game engine computes how bright each pixel is from the surface normal and the light direction. Move the light with the sliders, switch the shading method, and click the sphere to see how one pixel is computed.



Light azimuth: **-40°**

Light elevation: **35°**

Flat (per face) | **Smooth (per pixel)** | **Phong highlight**

Selected pixel: N = (-0.34, 0.39, 0.86)
N·L = 0.94
I = 0.08 + 0.92·0.94 + 0.55·0.67 = **1.31**

959 of 1176 pixels face the light (82%). The normal is recomputed per pixel: the surface looks smooth.

A graphics course teaches this computation: **Lambert's** law (I ∝ N·L), the Phong highlight and where to compute the normal. The GPU runs the same formula for millions of pixels every frame, in parallel.

CONTEXT

Prerequisites

Courses worth knowing before this one, and why they are needed.

Course	Why it is needed
Linear Algebra	transformation matrices, projection
Programming with C	APIs such as OpenGL and Vulkan are used from C/C++
Analytic geometry	vectors, planes, line-plane intersection: the language of ray tracing

CONTEXT

Are You Ready for This Course?

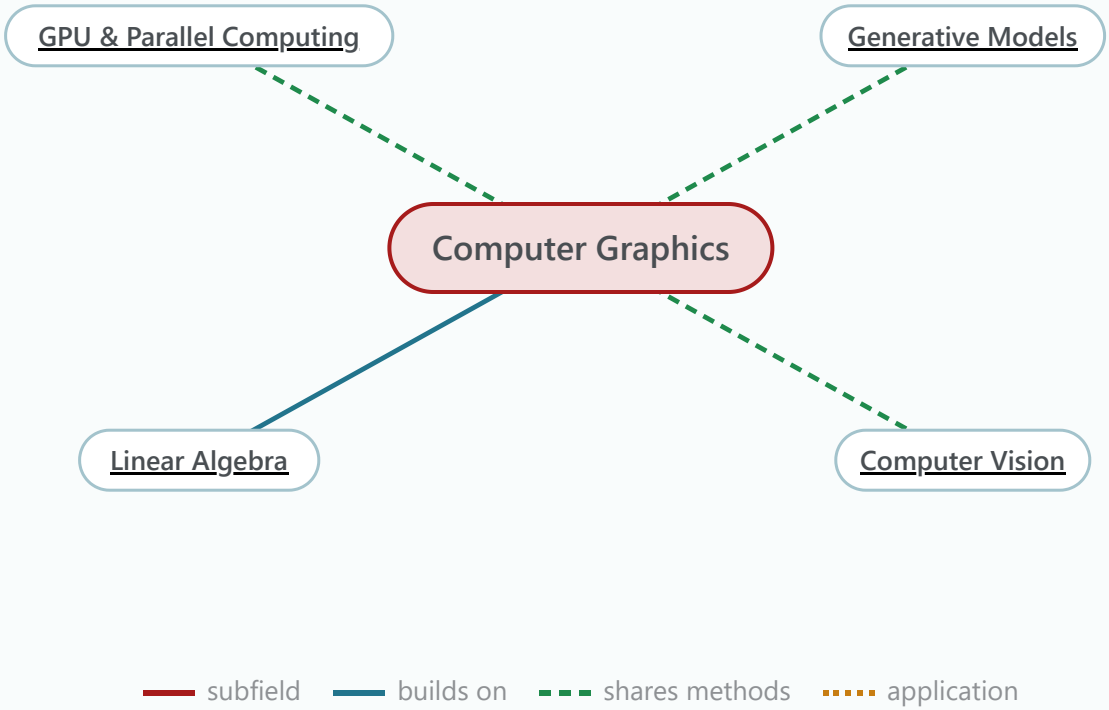
Ask yourself; for every "no", the right-hand column shows where to start.

Ask yourself	If not, start with
Can you rotate and translate a point with a 4x4 matrix?	Linear Algebra
Can you find the point where a ray hits a plane?	Line and plane equations from analytic geometry
Can you write a C or C++ program that uses arrays and structs?	An intro C course, then a short introduction to C++
Can you explain why a GPU runs thousands of threads in parallel?	GPU Programming and Parallel Computing

If you can say "yes" to most of these, you are ready; one or two gaps can be closed during the semester.

CONTEXT

Neighbouring Courses



OVERLAP

Overlapping Courses

Relation	Course	Shared ground
builds on	This course builds on Linear Algebra	transformation matrices, projection
shares methods	Shares methods with Generative Models	image and scene synthesis
shares methods	Shares methods with Computer Vision	camera model; graphics renders scenes, vision recovers them
shares methods	Shares methods with GPU Programming and Parallel Computing	graphics pipeline, shaders

Where It Differs (1/2)

Criterion	Computer Graphics	Computer Vision	Image Processing
Core question	How is a realistic image produced from the description of a scene?	How does a computer understand what is in an image or a video?	How is an image enhanced, transformed and divided into regions?
Approach	geometry and light simulation	extracting meaning from images	pixel-level transformations
Input	scene: models, lights, camera	image or video	image
Output	image or animation	label, box, depth, 3D structure	processed image or regions
Success measure	realism, frame rate	accuracy, mAP, IoU	visual quality, PSNR
Math load	high: linear algebra, geometry	high: geometry, linear algebra	medium: convolution, Fourier
Data needs	none	many labeled images	little; learning not required

- **Computer Vision:** Computer vision tries to recover the scene from an image; computer graphics produces the image from the scene.
- **Image Processing:** Image processing transforms an existing image; computer graphics builds the image from geometry and light.

Where It Differs (2/2)

Criterion	Computer Graphics	Generative Models	GPU & Parallel Computing
Core question	How is a realistic image produced from the description of a scene?	How does a model generate new samples that look like real data?	How do thousands of cores share the same computation?
Approach	geometry and light simulation	learning the data distribution	data-parallel computation
Input	scene: models, lights, camera	huge, mostly unlabeled data	parallelizable algorithm
Output	image or animation	new samples: images, text, audio	accelerated program
Success measure	realism, frame rate	realism and diversity (FID etc.)	speedup, memory bandwidth
Math load	high: linear algebra, geometry	high	low-medium
Data needs	none	very large	none

- **Generative Models:** Generative models synthesize images learned from data; computer graphics computes them from physics and geometry.
- **GPU & Parallel Computing:** GPU programming focuses on general parallel computing; computer graphics uses the GPU as a tool for rendering.

PRACTICE

Applications

Games
Real-time graphics at tens of frames per second.

Film and animation
Physically based rendering taking minutes per frame.

CAD and architecture
Seeing a design before it is built.

Simulation
Flight and driving simulators.

Virtual reality
Low-latency images for two eyes.

Scientific visualization
Medical volume data, flow and weather models.

PRACTICE

Risks and Ethics

Photorealistic fakes

Realistic rendering makes it easy to produce a convincing photo or video of an event that never happened.

Photosensitive epilepsy

Fast flashing patterns can trigger seizures; in 1997 a Pokemon episode sent hundreds of children in Japan to hospital.

VR sickness

High latency or a low frame rate causes dizziness and nausea in VR; timing is as much a safety issue as the image.

Convincing but wrong

Architectural and scientific visualization approximate light and materials; a beautiful image can justify a wrong decision.

In depth: Explainable AI and AI Ethics.

PRACTICE

Tools and Datasets

Tools and libraries

OpenGL · Vulkan · WebGL and three.js · Blender · Unity · Unreal Engine

Classic test scenes

Utah teapot · Stanford bunny · Cornell box · Sponza atrium · Suzanne (Blender)

A graphics course gives a **scene** instead of a dataset: once geometry, materials, lights and camera are known, the image is computed.

PRACTICE

Common Misconceptions

Common belief	In fact
Computer graphics is just making games.	Film, architecture, medicine, simulation and scientific visualization are graphics too.
Ray tracing is a new invention.	The method dates from 1980; what is new is hardware that runs it in real time.
Graphics and computer vision are the same.	Each is the inverse of the other; they share the camera model but go opposite ways.
More triangles are enough for realism.	Light, shadow and material often matter more than geometric detail.

NEXT STEPS

After This Course

Courses that build on or extend this one

This course is mostly an end point: it builds on the others.

Related careers

Graphics programmer | Game engine developer | Technical artist | XR developer | Visualization engineer

Graphics uses linear algebra and GPU programming more concretely than any other course. From here you can move to real-time rendering, physically based rendering, or in the other direction to computer vision; generative models and neural rendering are steadily merging the two fields.

NEXT STEPS

Further Reading

- Marschner and Shirley, *Fundamentals of Computer Graphics*, 4th ed., CRC Press, 2016.
- Hughes, van Dam, McGuire et al., *Computer Graphics: Principles and Practice*, 3rd ed., Addison-Wesley, 2013.
- Akenine-Möller, Haines, Hoffman et al., *Real-Time Rendering*, 4th ed., CRC Press, 2018.
- Pharr, Jakob and Humphreys, *Physically Based Rendering*, 4th ed., MIT Press, 2023.

The End

Chapter 22: Computer Graphics

← Previous chapter · Map of the field · Next chapter →