

INTERACTIVE COMPUTER GRAPHICS

Module 1

Introduction to Computer Graphics

BITM3213 • Interactive Computer Graphics

Faculty of Information & Communication Technology (FTMK)

Universiti Teknikal Malaysia Melaka • Session 2025/2026



Learning Outcomes — Module 1



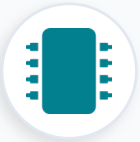
Define computer graphics

Explain what CG is and how it differs from image processing and computer vision.



Identify applications

Recognise where CG is used across industry, science and entertainment.



Describe a graphics system

Outline the hardware and software that turn a model into pixels.



Connect theory to practice

See how these ideas map directly onto OpenGL and Lab 1.

WHY SHOULD YOU CARE?

Almost every image on your screen was computed, not captured.



Games

Real-time worlds rendered 60+ times a second.



Film & VFX

Photoreal creatures and entire digital sets.



Medicine

CT and MRI scans turned into 3D anatomy.



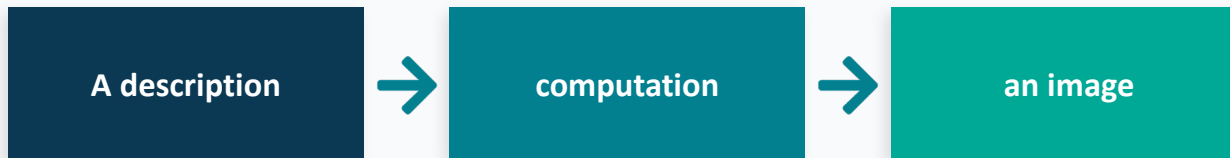
Research

Your own field — simulation & visualization.

What is Computer Graphics?

Computer Graphics is the field concerned with **creating, storing and manipulating images using a computer.**

In one line:



Think of it as the reverse of seeing

Your eyes take light and form a picture in your brain. Computer graphics does the opposite direction: it starts with a description of a scene and computes what the picture should look like.

PHYSICAL IMAGE

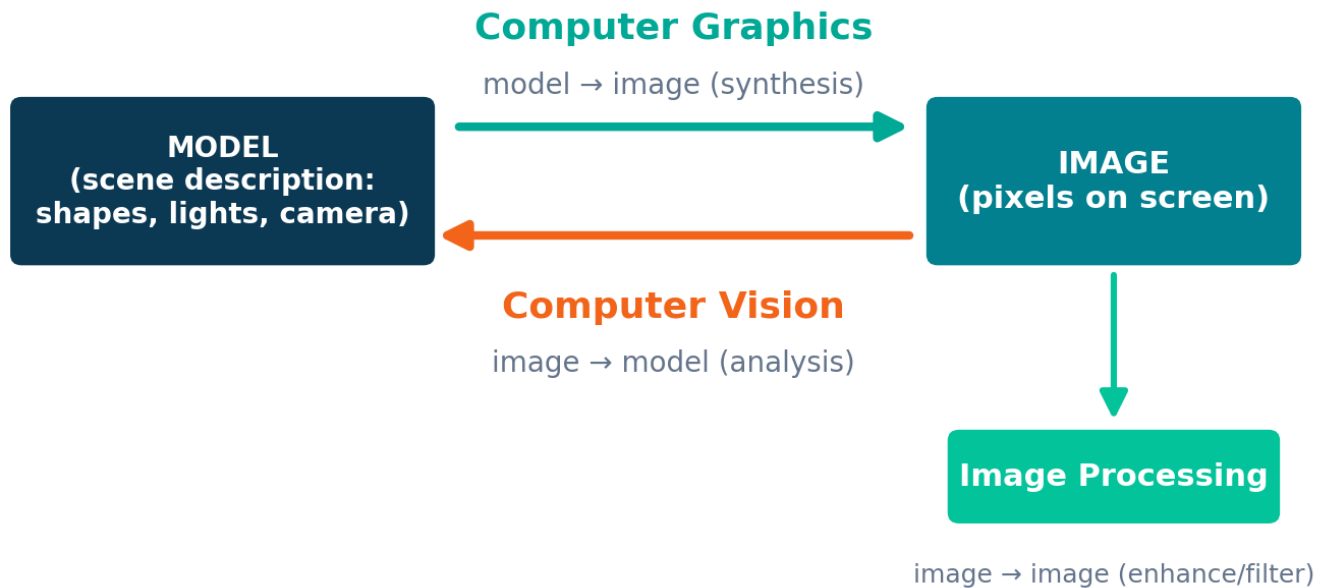
Captured by a camera
Real light, real lens
Photograph / scan
Limited by physics

SYNTHETIC IMAGE

Computed by a program
Simulated light & camera
Rendered by GPU
Full creative control

Physical images are captured; synthetic images are computed.

Graphics vs Image Processing vs Vision



Computer Graphics

$\text{model} \rightarrow \text{image}$

Computer Vision

$\text{image} \rightarrow \text{model}$

Image Processing

$\text{image} \rightarrow \text{image}$

Where Computer Graphics Is Used

Where Computer Graphics Shows Up

**Games &
Entertainment**

**Film & VFX /
Animation**

**CAD / CAM
& Design**

**Medical
Imaging**

**Scientific
Visualization**

**VR / AR &
Simulation**



Common thread

Every one of these needs to turn data or a 3D description into a picture a human can understand — that is exactly what computer graphics provides. As your lecturer's research shows, simulation and visualization are among the fastest-growing uses.

Physical vs Synthetic Images



Two ways an image can come to exist:

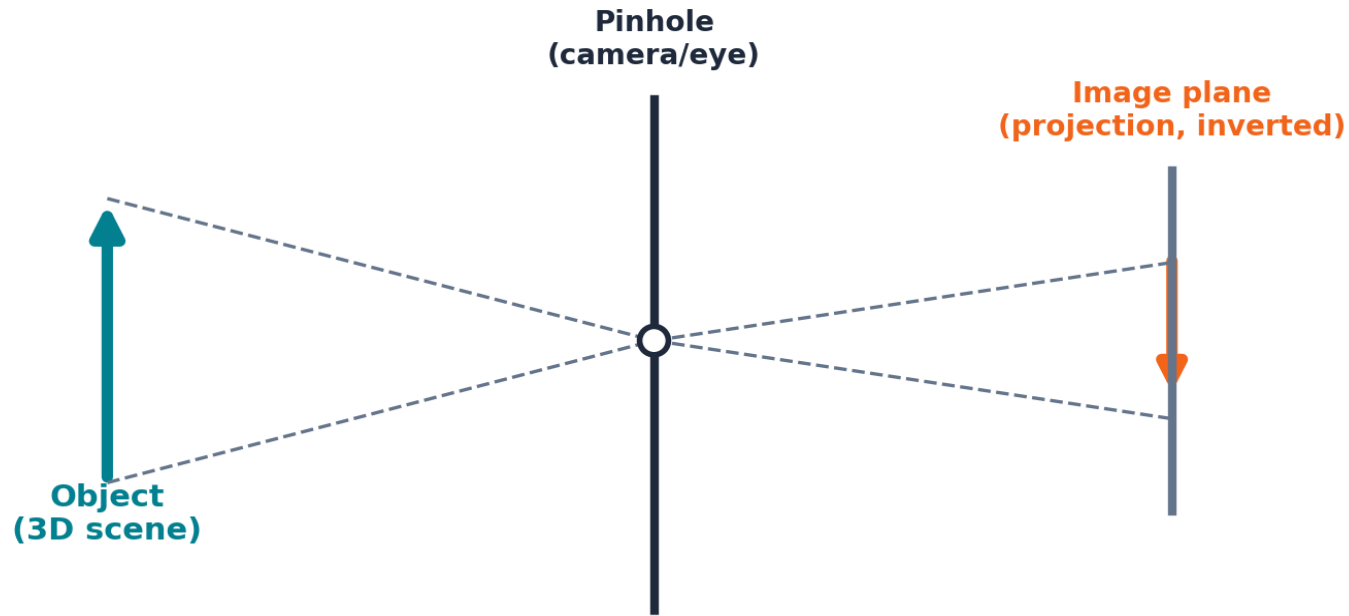
Physical Image

- Formed by real light hitting a sensor or film
- Example: a photo from your phone camera
- Bound by optics and physics
- Captured — it records what already exists

Synthetic Image

- Computed entirely by a program
- Example: a frame in a video game
- Light and camera are simulated in maths
- Generated — it builds something new

The Pinhole Camera Model



Synthetic image = simulating how light/rays form an image

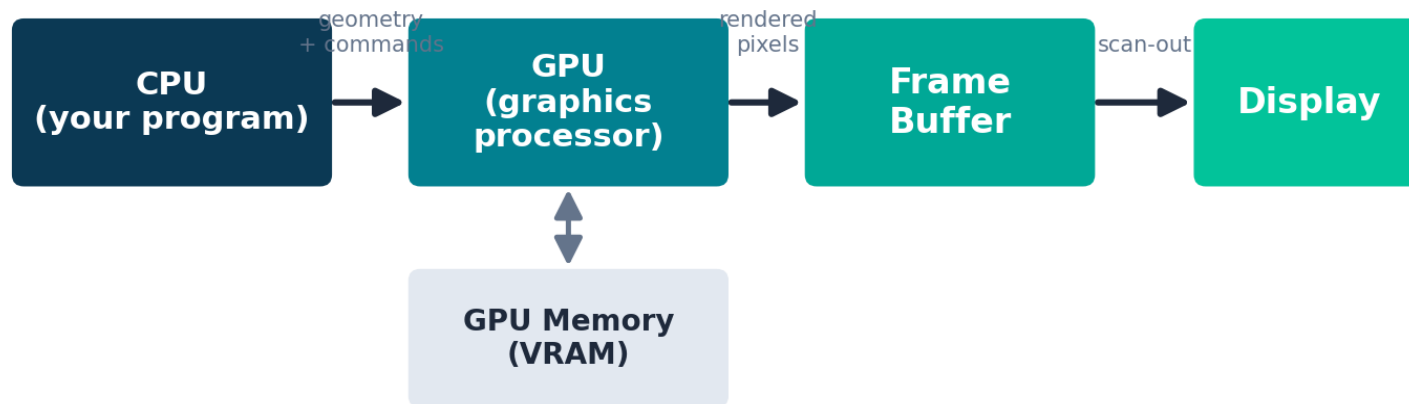


Why we start here

To compute an image, the program must imagine a camera. The simplest model is a pinhole: light from the scene passes through a single point and lands on an image plane. OpenGL's projection maths is a more flexible version of exactly this idea — so understanding the pinhole makes 'projection' later feel natural.

A Graphics System

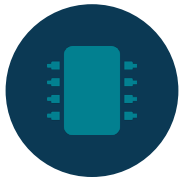
A Graphics System: from program to pixels



The hand-off

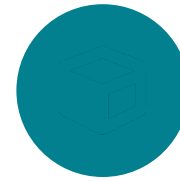
Your CPU code prepares geometry and commands, then hands them to the GPU. The GPU does the heavy parallel work and writes pixels into the frame buffer — a block of memory that the display reads to show the picture.

CPU vs GPU



CPU

- A few very fast, flexible cores
- Runs your main program logic
- Great at sequential, branching tasks
- Sends work + data to the GPU



GPU

- Thousands of small parallel cores
- Runs the same operation on many pixels/vertices
- Great at massively parallel maths
- Where shaders actually run

Where Does OpenGL Fit In?

OpenGL is a cross-platform **API** — a standard set of commands your program uses to talk to the GPU.

Your C++ program

OpenGL API (the common language)

GPU driver

GPU hardware



Why an API matters

You never write code for a specific graphics card. You write OpenGL commands, and each vendor's driver translates them for their hardware. That is why the same program runs on NVIDIA, AMD or Intel GPUs — write once, run on any compliant GPU.

Modern OpenGL, Not Legacy

Legacy (old)

avoid

- Uses glBegin() / glEnd()
- Fixed-function: the GPU decides how
- Simple to start, but a dead end
- Not how real software is written today

Modern (this course)

use this

- Core profile 3.3+, shader-based
- You write the pipeline: VAO, VBO, GLSL
- Matches industry and game engines
- More setup, but the right mental model

The Modern OpenGL Pipeline

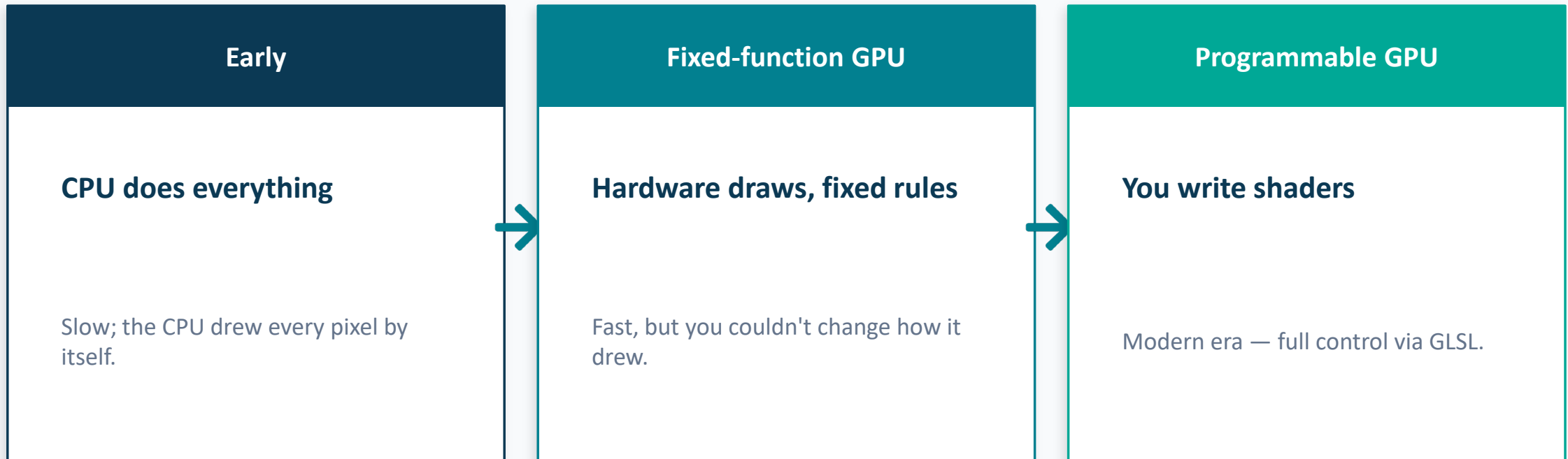
The Modern OpenGL Pipeline (you'll build this in Lab 1)



Read it left to right

You hand the GPU your vertex data, the vertex shader positions each point, the rasterizer fills in pixels between them, the fragment shader colours each pixel, and the result lands in the frame buffer for display. In Lab 1 you will write the two shaders yourself.

Graphics Architectures — A Short Evolution



Computer Imaging: Vision & Image Processing



Computer Vision

- Goal: understand an image
- Input is pixels, output is meaning
- Example: face detection, object tracking
- The 'inverse' of graphics



Image Processing

- Goal: improve or transform an image
- Input and output are both images
- Example: sharpening, blurring, filters
- No 3D model is involved

What you'll do in Lab 1

1

Set up the tools

Install GLFW, GLAD and GLM in Visual Studio.

2

Open a window

Create an OpenGL 3.3 core-profile window.

3

Send a triangle

Upload vertices to the GPU via a VAO and VBO.

4

Write shaders

A vertex shader and a fragment shader in GLSL.

5

Render

See a colour-interpolated RGB triangle on screen.

Key Terms to Remember

Rendering

Computing an image from a scene description.

GPU

Massively parallel processor that does the drawing.

Shader

A small GPU program you write in GLSL.

API

A standard command set to talk to the GPU (e.g. OpenGL).

Frame buffer

Memory holding the pixels shown on screen.

Synthetic image

An image computed by a program, not captured.

Quick Self-Check

Q1

In one sentence, what is computer graphics?

Q2

Which direction does the arrow go for graphics: model→image or image→model?

Q3

Name three real-world applications of CG.

Q4

What is the role of the GPU in a graphics system?

Q5

Why do we use modern OpenGL instead of glBegin/glEnd?

Module 1 — What We Covered

- ✓ Computer graphics = computing images from a description (model → image).
- ✓ It differs from computer vision (image → model) and image processing (image → image).
- ✓ CG powers games, film, medicine, CAD, science and VR/AR.
- ✓ A graphics system flows: CPU → GPU → frame buffer → display.
- ✓ We use modern, shader-based OpenGL as our API.
- ✓ Every concept today maps directly onto Lab 1.



What's Next

Lab 1 — Setting Up Modern OpenGL

GLFW + GLAD + GLM in Visual Studio, then render your first triangle.

References

Gordon & Clevenger (2024). Computer Graphics Programming in OpenGL with C++, 3rd Ed. Mercury Learning.
Angel & Shreiner (2014). Interactive Computer Graphics: A Top-Down Approach with WebGL, 8th Ed. Pearson.
de Vries (2020). Learn OpenGL. • Online: learnopengl.com

Any questions? • See you in Lab 1