COMPUTER GRAPHICS COURSE

Welcome





Andreas A. Vasilakis

Why should anyone attend the course?

Every day we interact with 3D graphics:

- 🎮 Video Games
- 🎬 Movies & Animation
- 🥽 Virtual & Augmented Reality
- 📊 Data Visualization

This course will help you understand how

- these images are created
- how modern GPUs work
- how to build your own 3D graphics applications

Why should anyone attend the course?

Course Cycles

- Software Systems

Why should anyone attend the course?

Course Cycles

- Software Systems

One step (6 ECTS) closer to your degree!

Why should anyone attend the course?

Course Cycles

- Software Systems

One step (6 ECTS) closer to your degree!

One step closer to being assigned your thesis!

One step closer to your first publication!

One step closer… to the first day of the rest of your life!

Why Computer Graphics?

Computer Graphics is everywhere!



More than pretty pictures...

combines **mathematics**, **algorithms**, **physics**, **programming**, and **creativity** to **transform data** into **visual experiences.**

What is Real?



What is Real?

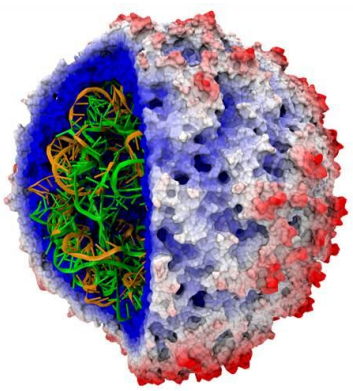
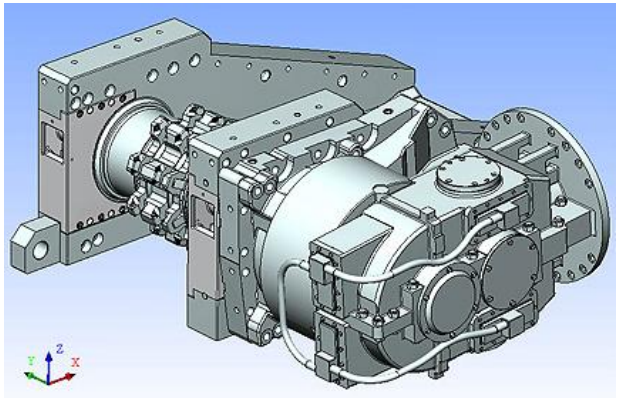


What do we use CG for?












CG Applications

- Computer games and interactive applications
  - Most high-efficiency algorithms come from and target this application domain!
  - Big industry, from indy and casual games to AAA productions
- Computer-Aided Design / Manufacturing / Engineering
  - Physical product design using surface and solid modeling and geometric tools
- Graphical User Interfaces (GUIs)
  - 2D GUI implementation and acceleration
  - VR / AR and human-friendly interactive systems

CG Applications

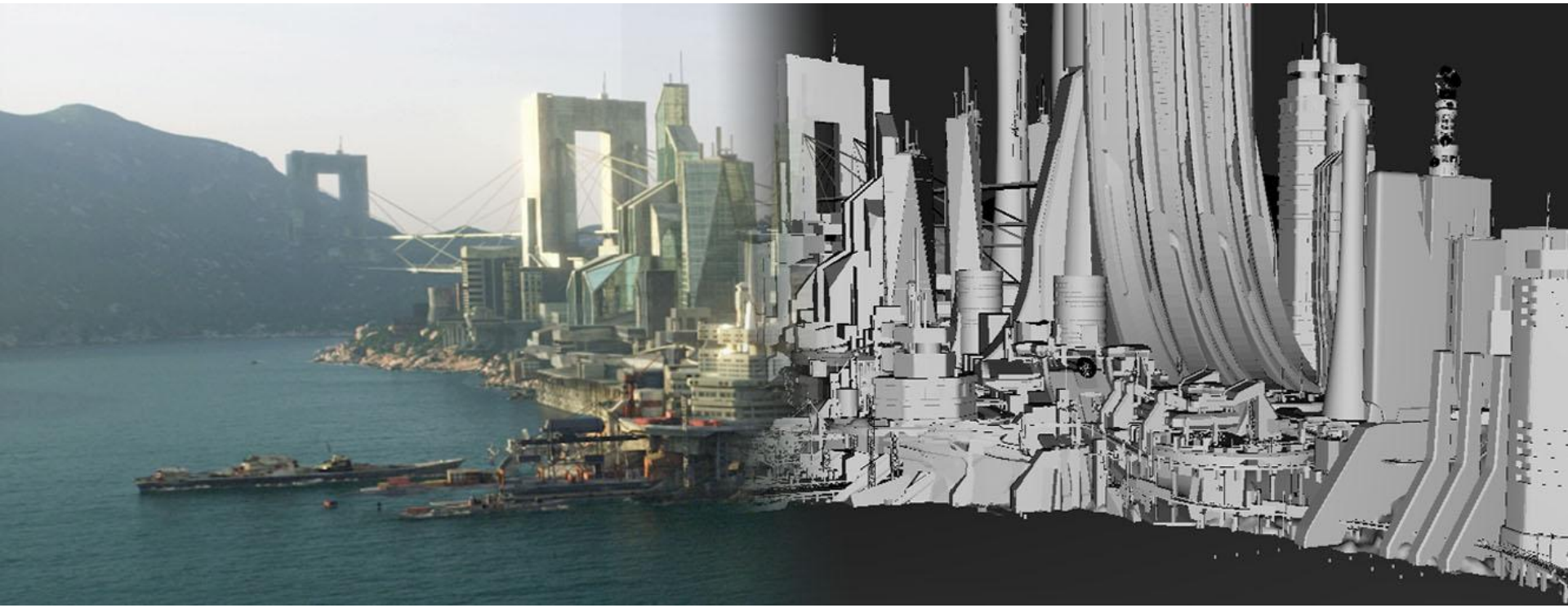
- Special effects for feature films
  - Ability to create the impossible or non-existent
- Animated films
- Scientific and medical visualization
- 2D and 3D printing technology

Image Synthesis



How can a computer turn millions of numbers into an **image** that humans can understand instantly?

Image Synthesis



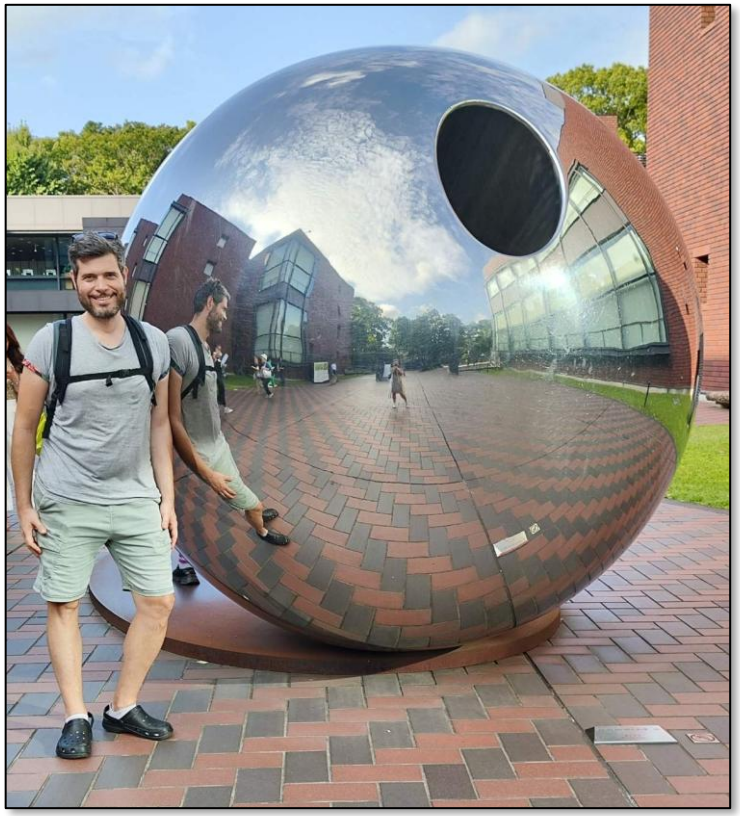
2D image generation is the core task of 3D computer graphics

Instructor Introduction

About the Instructor

Personal Information

- **1983,** *born in Corfu*
- **2001-2014,** *study in Ioannina*
- **2014-Now,** *live in Athens*



About the Instructor

Personal Information

- **1983,** *born in Corfu*
- **2001-2014,** *study in Ioannina*
- **2014-Now,** *live in Athens*





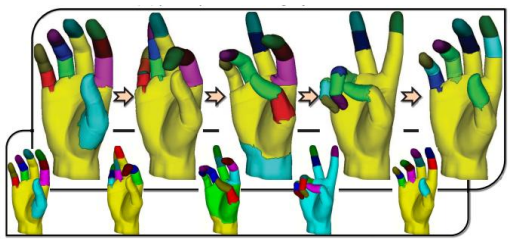


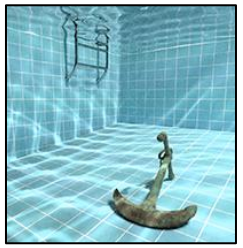
About the Instructor

Assistant Professor, AUEB (2025/12)

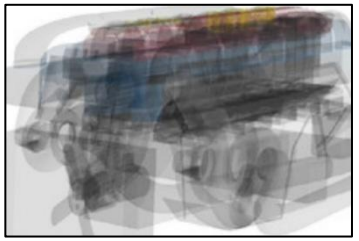
Research Interests

- Photorealistic rendering
- GPU programming
- Virtual and augmented reality
- Geometry processing and animation













Undergraduate Thesis Opportunities

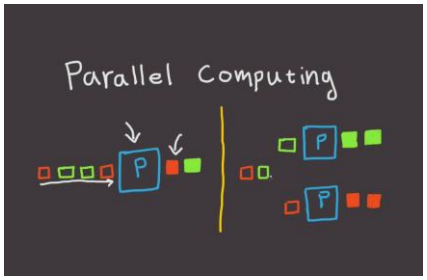


About the Instructor

Current Courses BSc, AUEB (2026-2027)



Computer Graphics
Fall 2026



Parallel Programming
Spring 2027

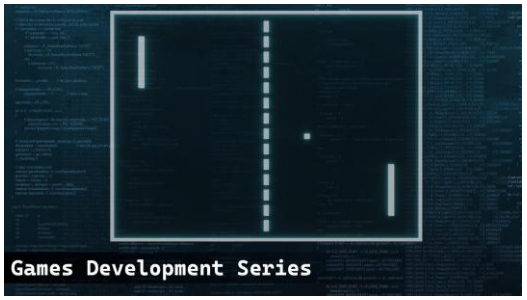


Human-Computer Interaction
Spring 2027

About the Instructor

Scientific Direction KEDIVIM, AUEB (2026-2027)

Training Program in Games Development (Greek)



Fundamentals of Programming for Games & Interactive Applications

Fall 2026



Game Design and Prototyping

Fall 2026

Computer Graphics Group



Prof. G. Papaioannou



Prof. A. A. Vasilakis



I. Evangelou



V. Kotsis-Panakakis

https://graphics.cs.aueb.gr

Join Our Research CG Group!



Prof. G. Papaioannou



Prof. A. A. Vasilakis



Your Name Here!



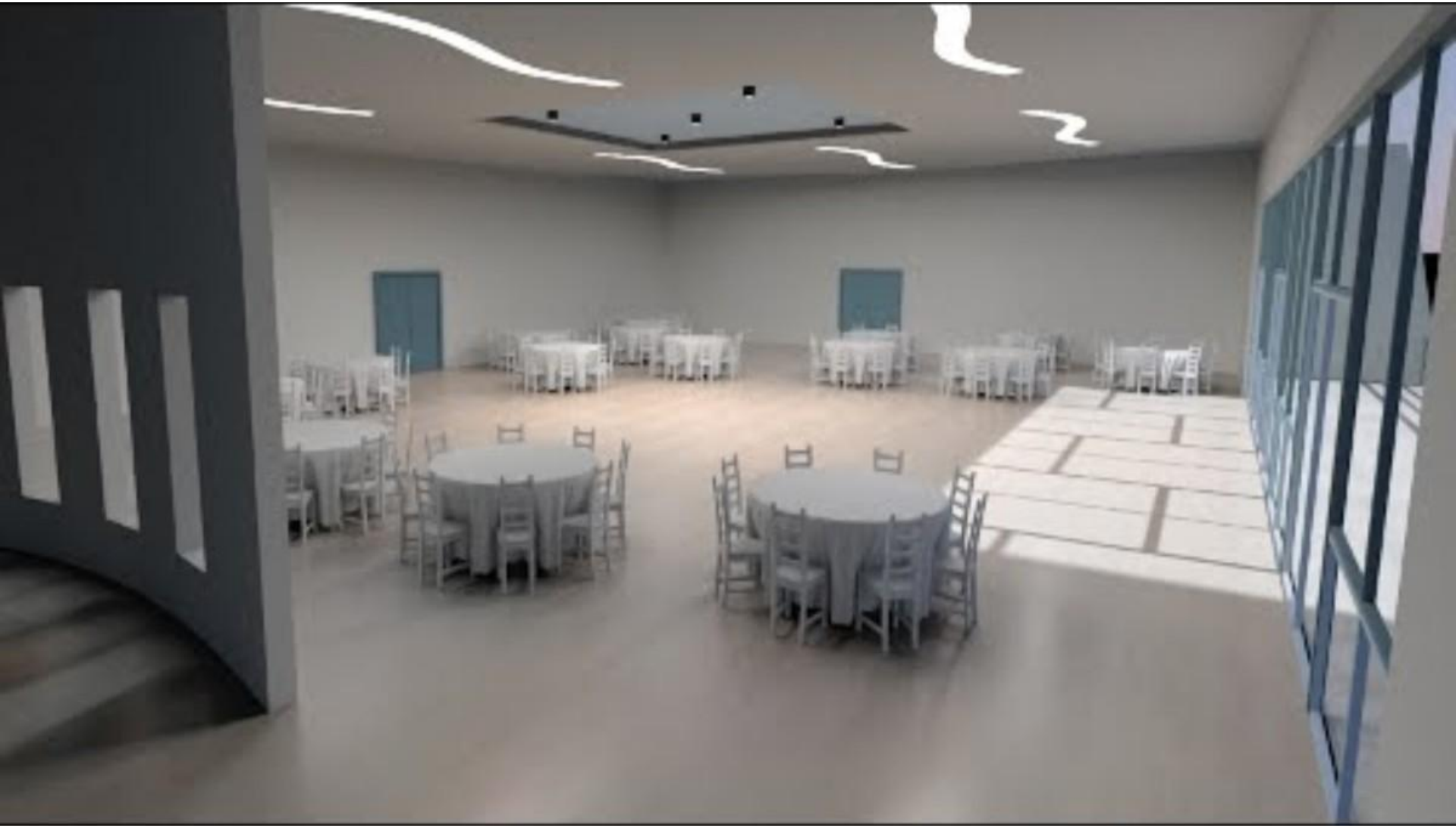
I. Evangelou



V. Kotsis-Panakakis

https://graphics.cs.aueb.gr

A Glimpse into Our Research



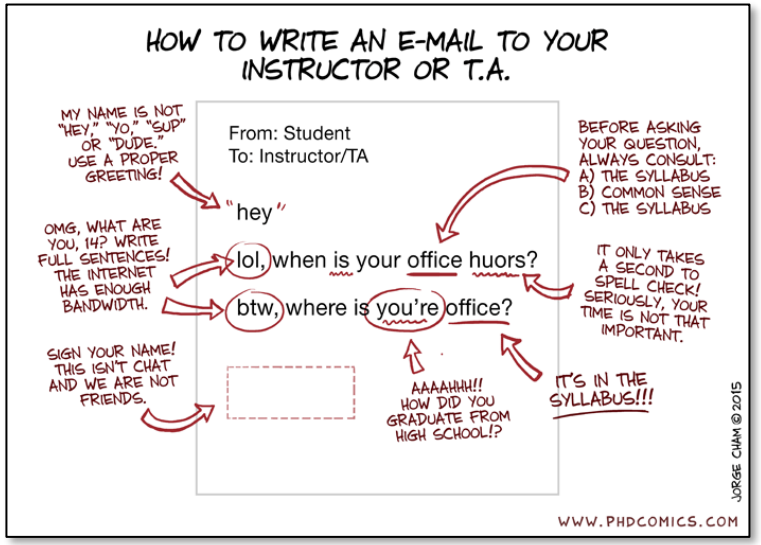
https://graphics.cs.aueb.gr

About the Instructor

Contact Information

- **Website:** *graphics.cs.aueb.gr/users/abasilak*
- **Email:** *abasilak@aueb.gr*
- **Github, Twitter, LinkedIn:** *abasilak*

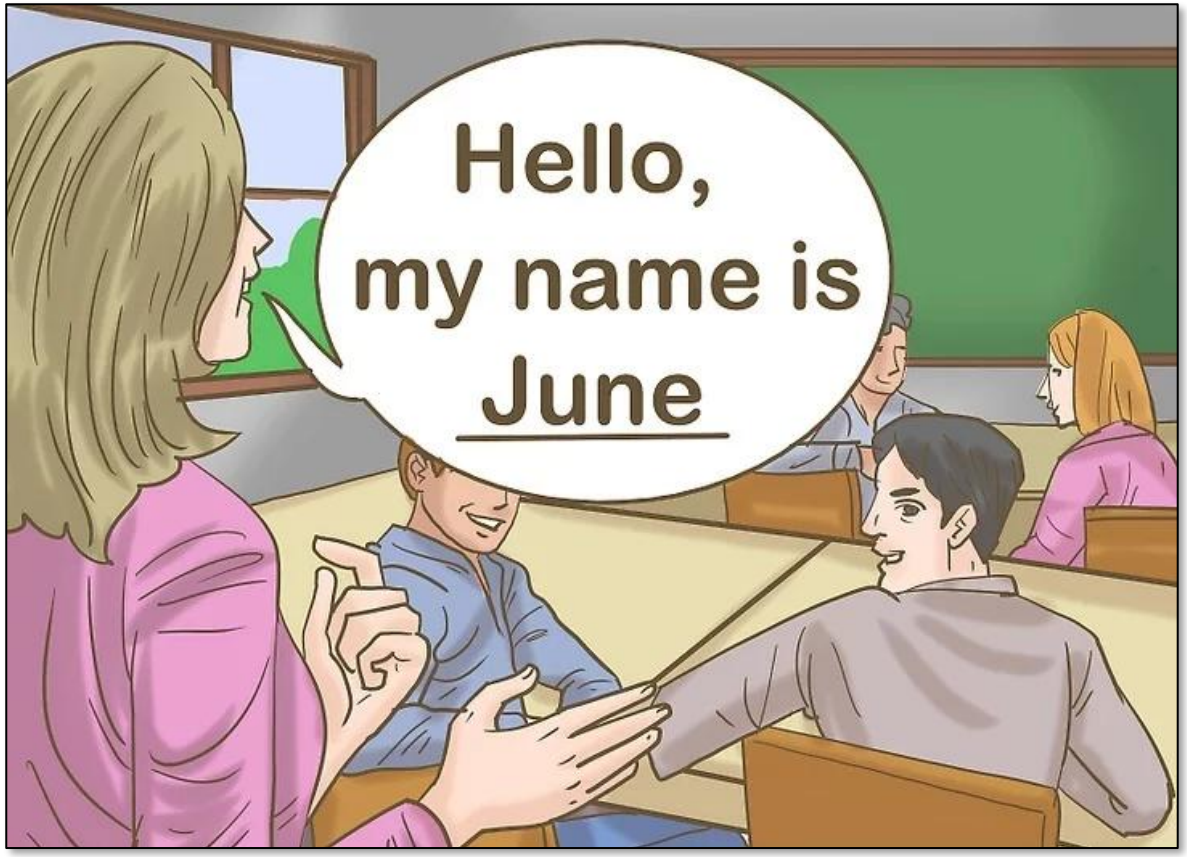
- **Office Hours:** *10:00-12:00, Monday & Friday*
- **Location:** *Office 405*, *New Building AUEB,*
  Trias 2, Athina 11362





Students Introduction

About the Students



About the Students

Department of Informatics

AUEB, Greece

Department of ???

Erasmus

Course Description

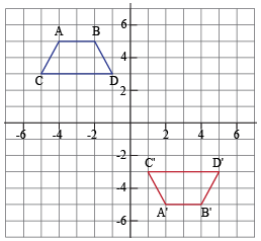
About the course



- Introduces students to the fundamentals of Computer Graphics and the principles behind the generation of digital images.
- Explores state-of-the-art computer graphics pipelines, fundamental algorithms, and examples from widely used commercial systems.
- Combines theoretical foundations with practical programming experience through the development of interactive 3D graphics applications.

Course Contents (1)

Computer Graphics Foundations



Introduction to computer graphics. Basic concepts, tools, applications, and production pipelines. Surface representations and data primitives and organization. Coordinate systems, geometric transformations, viewing and projections.



Real-time graphics – The rasterization pipeline. The hardware rasterization graphics pipeline: stages, polygon clipping and sampling, the graphics processing unit (GPU), programmable stages and image composition. Aliasing and antialiasing.



Appearance and shading. Materials and surface properties, local shading models (overview) and their properties, basic radiometric properties and the reflectance equation. Texturing and texture maps.

Course Contents (2)

Realtime and Production Rendering



Real-time graphics. Real-time shading pipelines, order-independent transparency, multi-pass algorithms: shadows, illumination caching, environment mapping, real-time global illumination. Post-processing effects.



Animation and Motion Capture. Principles and theory of basic computer animation, motion capture and tracking techniques and technologies, real-time animation concepts and implementation.



Ray tracing. The ray tracing pipeline, applications, pros and cons, ray generation, scene traversal and acceleration structures, specular reflection and transmission. Comparative study vs rasterization.

Instructional Methods



- Weekly Lectures (4 Hours)
  - **Tuesday**: 15:00 - 17:00 (Room **A44**)
  - **Thursday**: 13:00 - 15:00 (Room **A47**)
  - Theory coupled with **Unity/Blender** primers

- Laboratory Sessions (2 Hours)
  - Maximum 8 sessions [first @21 Oct]
  - **Wednesday**: 17:00-19:00 (**Remote – MS Teams**)
  - Practical experience with graphics programming

- Course Website
  - eclass.aueb.gr/courses/INF143/

Teaching Assistant



Vasileios-Ektor Kotsis-Panakakis

- vas.kotsispanakakis@aueb.gr
- Office 301

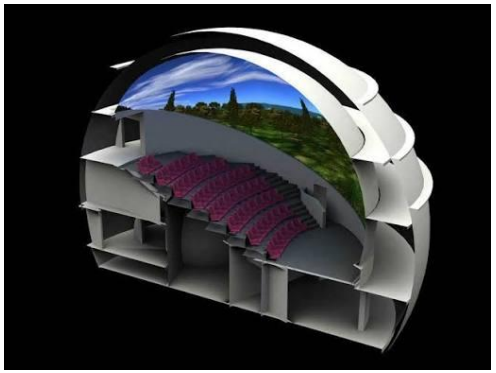
3D Graphics Programming using Three.js

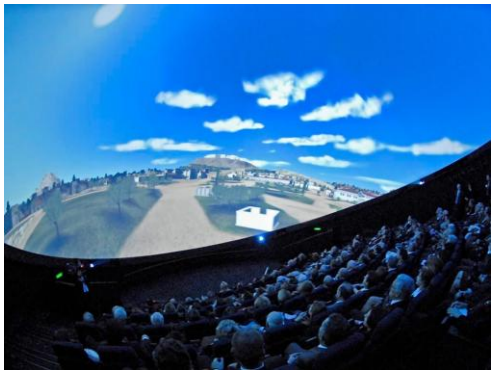
- Complement the theoretical material presented in class
- Be synchronized both chronologically & conceptually with the lecture topics
- Reinforce theoretical concepts through hands-on implementation
- Provide practical experience in developing interactive 3D graphics apps
- Introduce modern web-based graphics technologies & programming practices
- Offer guided project planning and support

Invited Speakers & Visits (TBD)



- Georgios Tsatiris D6 Learning
- Vangos Pterneas LightBuzz
- Dimitrios Christopoulos FHW
  - Academic Visit @Foundation of the Hellenic World !!!





Prerequisites

- Mandatory (one of)
  - Mathematics I
  - Computer Programming in C++
  - Computer Programming in Java
- Recommended
  - Technologies and Programming of Web Applications
  - Computer System Organization

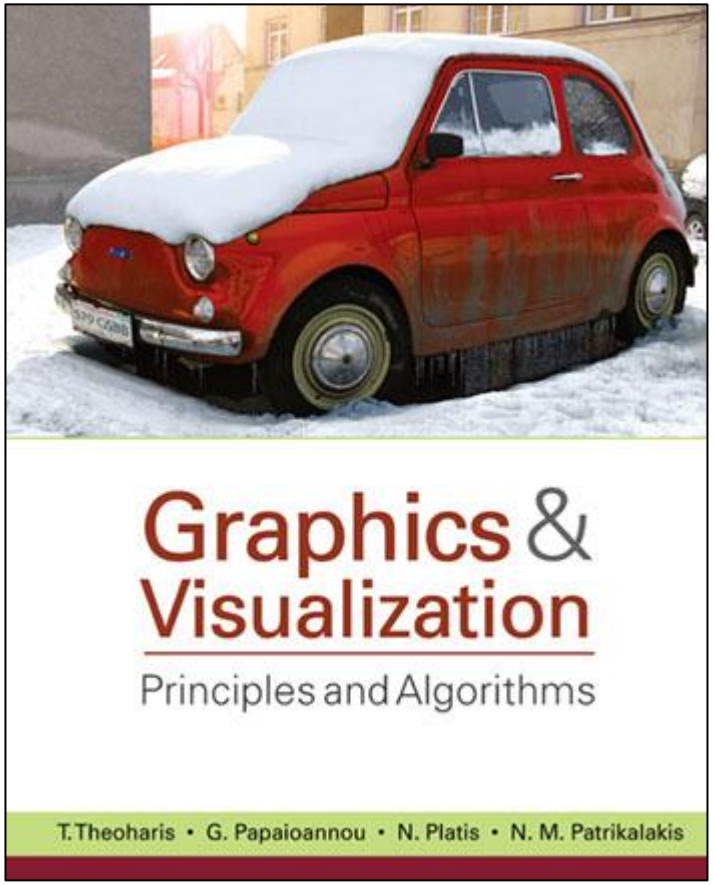


Educational Goals

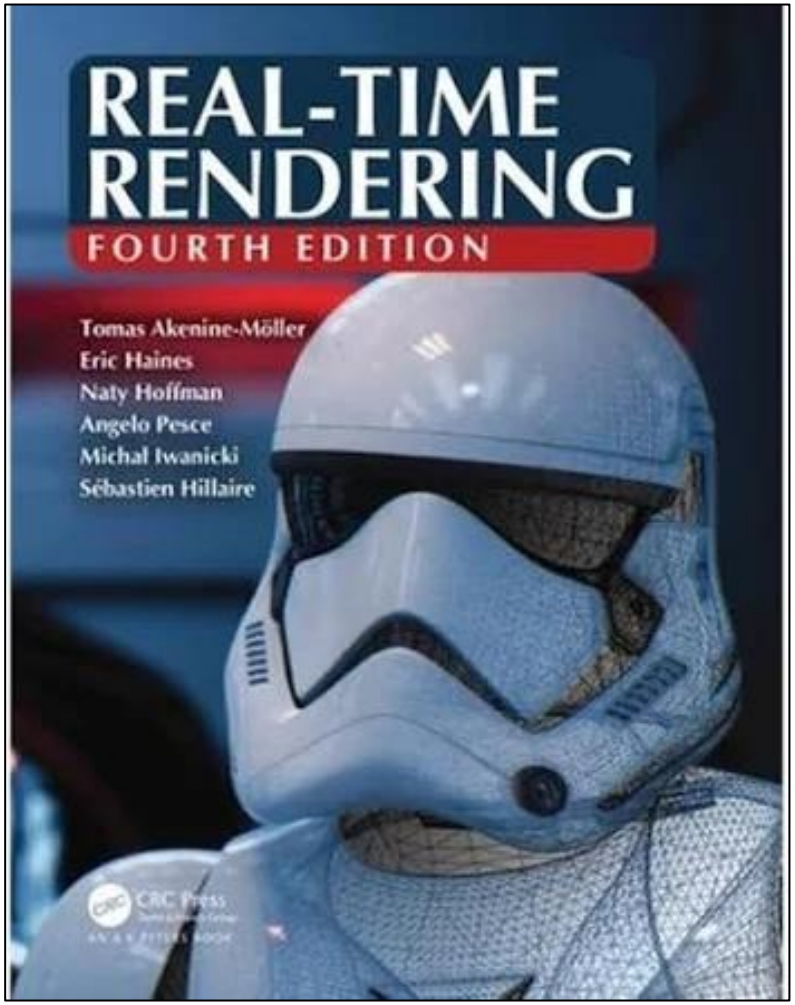
- Understand the fundamental principles of computer graphics
- Explain the graphics rendering pipeline
- Model and manipulate objects in 2D and 3D space
- Apply geometric transformations and camera projections
- Implement visibility, lighting, shading, and texturing
- Understand how modern GPUs generate images
- Develop interactive graphics applications
- Analyze and evaluate real-time rendering algorithms

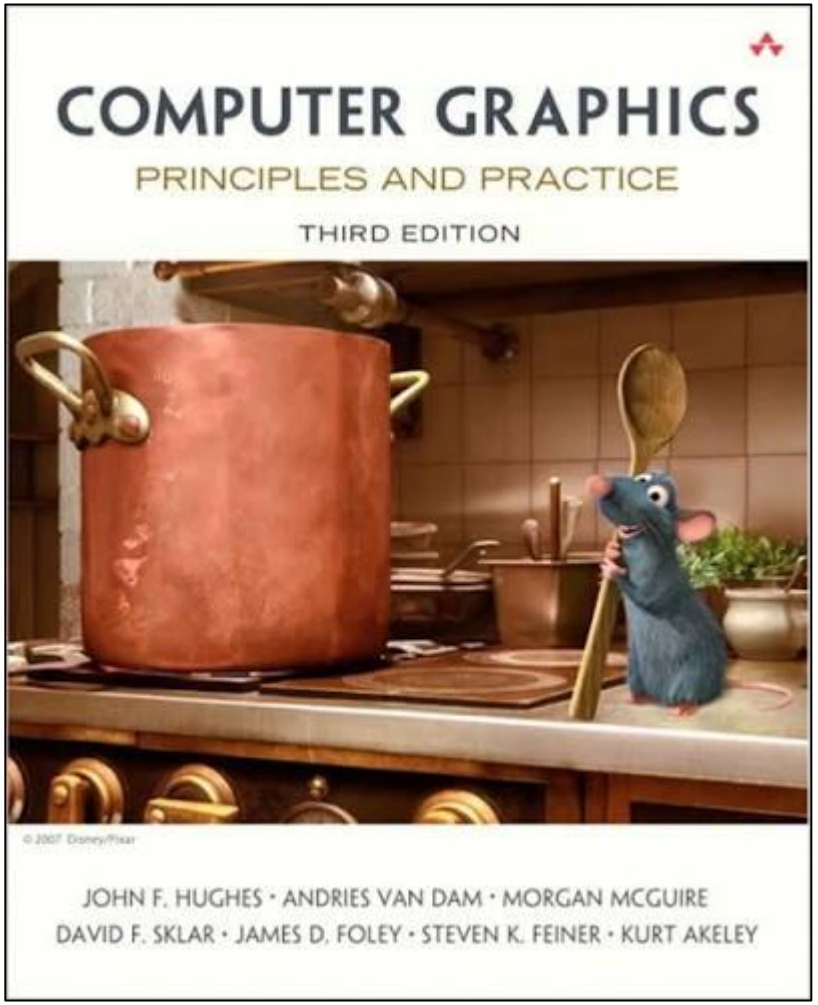
Main Textbook





Additional Reading





Additional Resources

Useful Sites

- http://graphicscodex.com/
- https://www.scratchapixel.com/
- https://www.realtimerendering.com/
- https://raytracing.github.io/
- http://immersivemath.com/



Annual international CG events

- SIGGRAPH
- Eurographics
- Complete list



SIGGRAPH 2026 Demo



Evaluation Methodology

- Written examination test (8 points)
- (Optional) Course Project (3 points)
  - Counts if the written exam grade is > 3.5/10
  - May be completed *individually* or in teams of up to *two students*.
  - Duration: 1 month, out @December
  - Oral Examination (20')
- Grading system: *max*(Exam + Project, 10)





For Erasmus Students:

The project does not affect the final grade

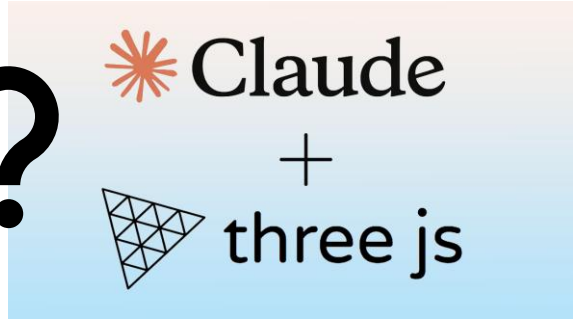
(i.e. you may omit the project)

Can I use AI (for the project)? YES, but…

If you **cannot explain** how your code works, then it is **not your code**.

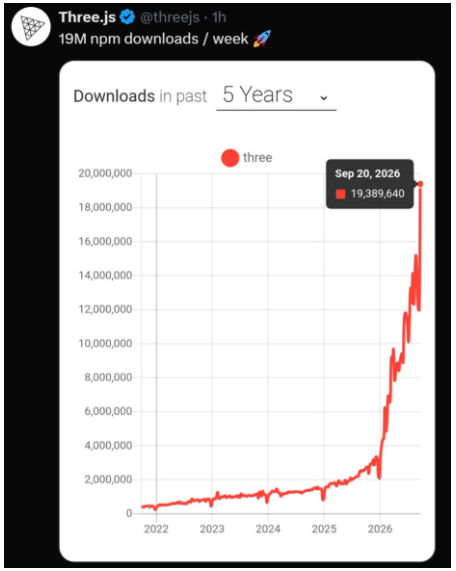
Appropriate Uses of AI

- ✅ Understanding graphics concepts
- ✅ Learning Three.js syntax and APIs
- ✅ Debugging code
- ✅ Brainstorming project ideas
- ✅ Improving documentation and reports
- ✅ Exploring alternative implementations



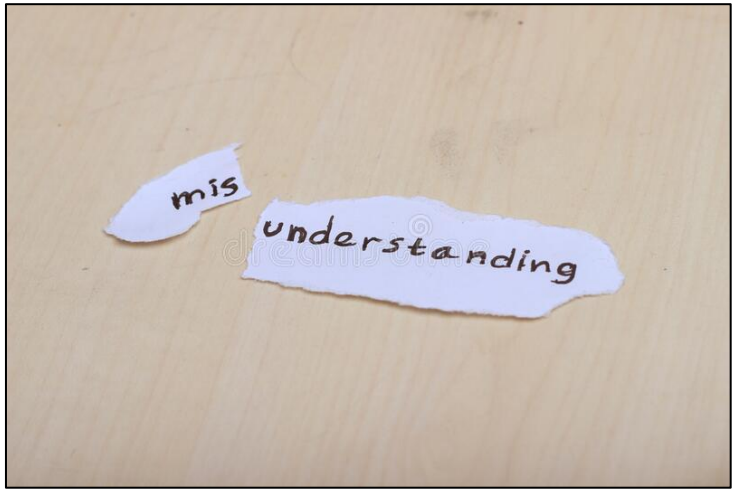
Inappropriate Uses of AI

- ❌ Submitting AI-generated code that you do not understand
- ❌ Copy-pasting entire projects without modification
- ❌ Using AI to bypass the learning process
- ❌ Claiming authorship of work you cannot explain
- ❌ Submitting code that you cannot debug or extend yourself



What You (Don’t) Know!

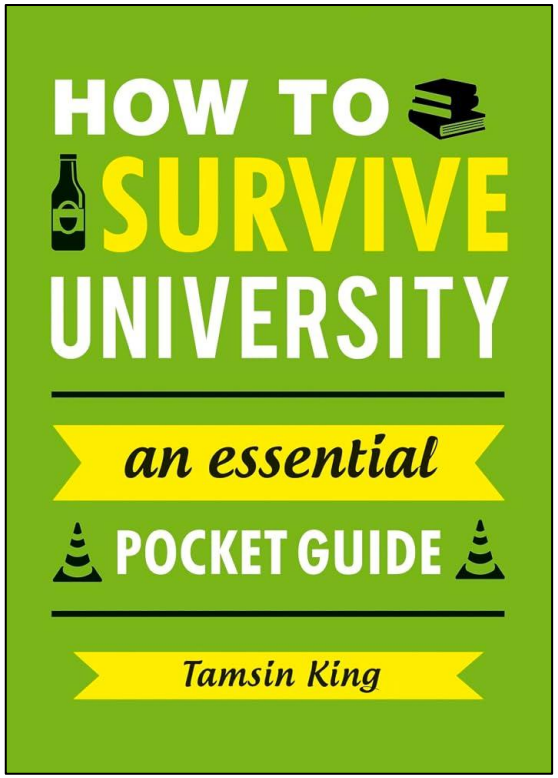
- This course was originally designed and structured by Prof. George Papaioannou



- What changed?
  - ✅ Course syllabus
  - ✅ Learning objectives
  - ✅ Assessment methodology
  - ❌ Laboratory component: **Three.js** replaces OpenGL

Survival Guide

- How Can I Do Well in This Course?
- Have You Found Your Direction?
- What Will I Do After Graduation?



Fundamentals

Rendering

- Rendering is the process of generating an image from a set of models (geometric representation)
  - It is the final product of a general image synthesis task using a rendering pipeline

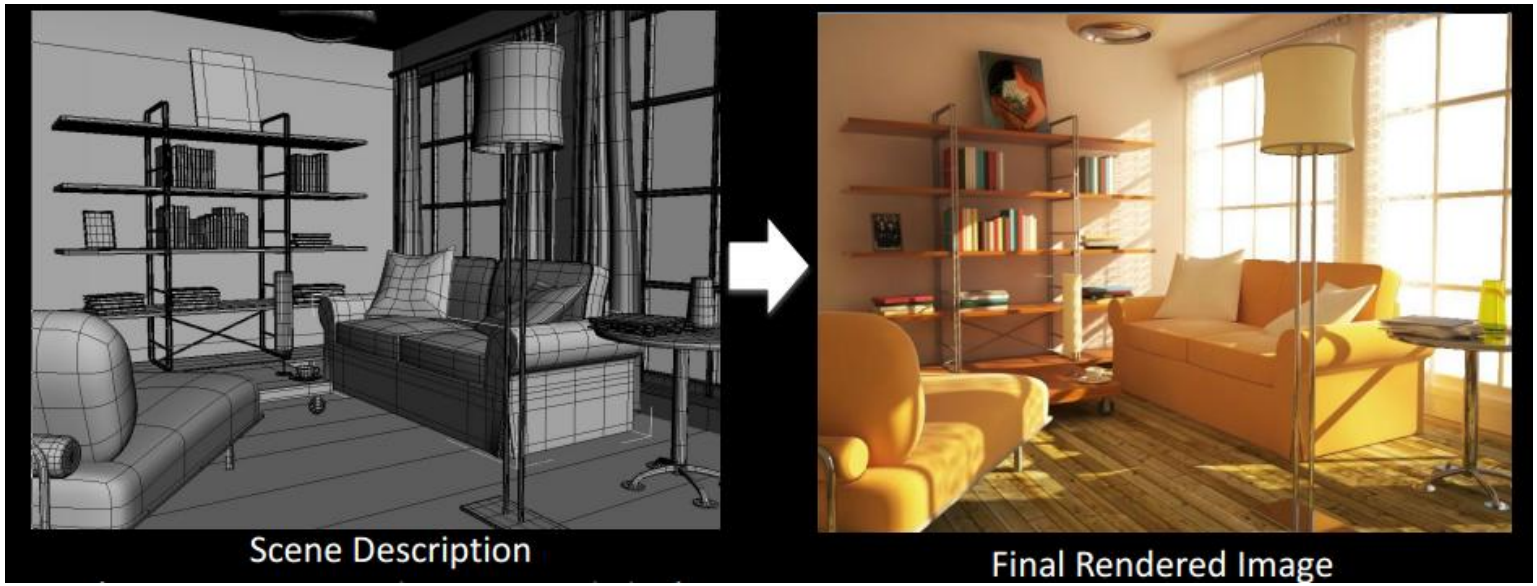
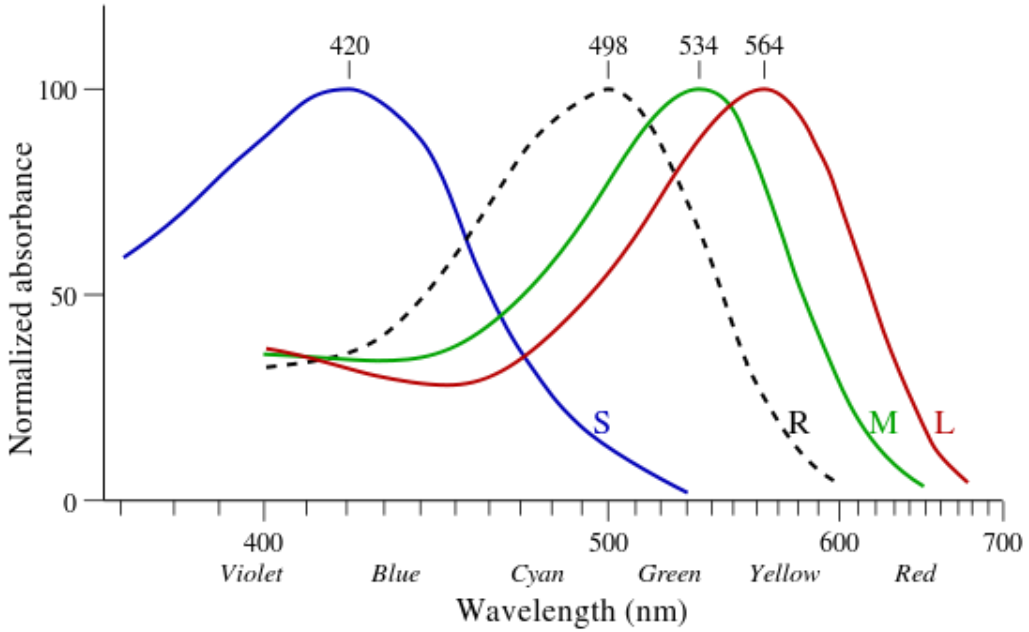


Image credit: http://alex-mccarthy.blogspot.com

The Human Visual System

- Sensitive photoreceptors: Rods and Cones
- The human visual system adapts to the level of illumination incident to the photoreceptors
  - Rods (scotoptic light): 10^{-6} – 10 cd/m^2 (nits)
  - Cones (photoptic light): 10^{-2} – 10^{8} cd/m^2 (nits)
- Total luminance range: 10^{-6} – 10^{8}



The Digital Image

A discretized configuration of intensity samples

Usually an array of pixels (a raster)

- Discrete approximation of a 2D continuous signal
- Luminance is sampled using a fixed or variable rate and attributed to the neighborhood of each pixel

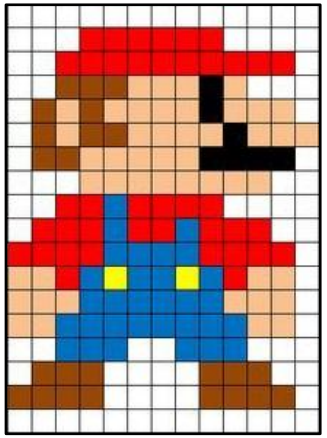
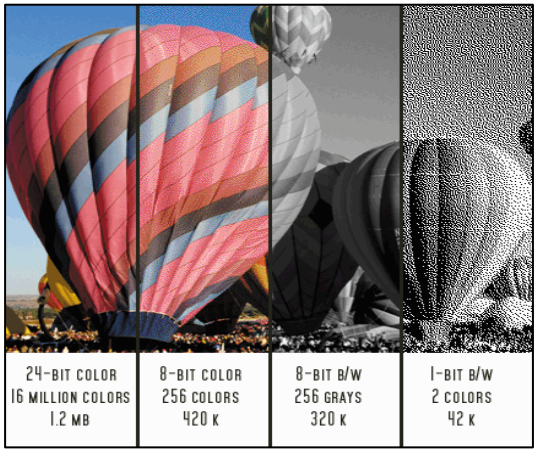


Image data are represented using a color model

- RGB (compatible with HVS),
- Other

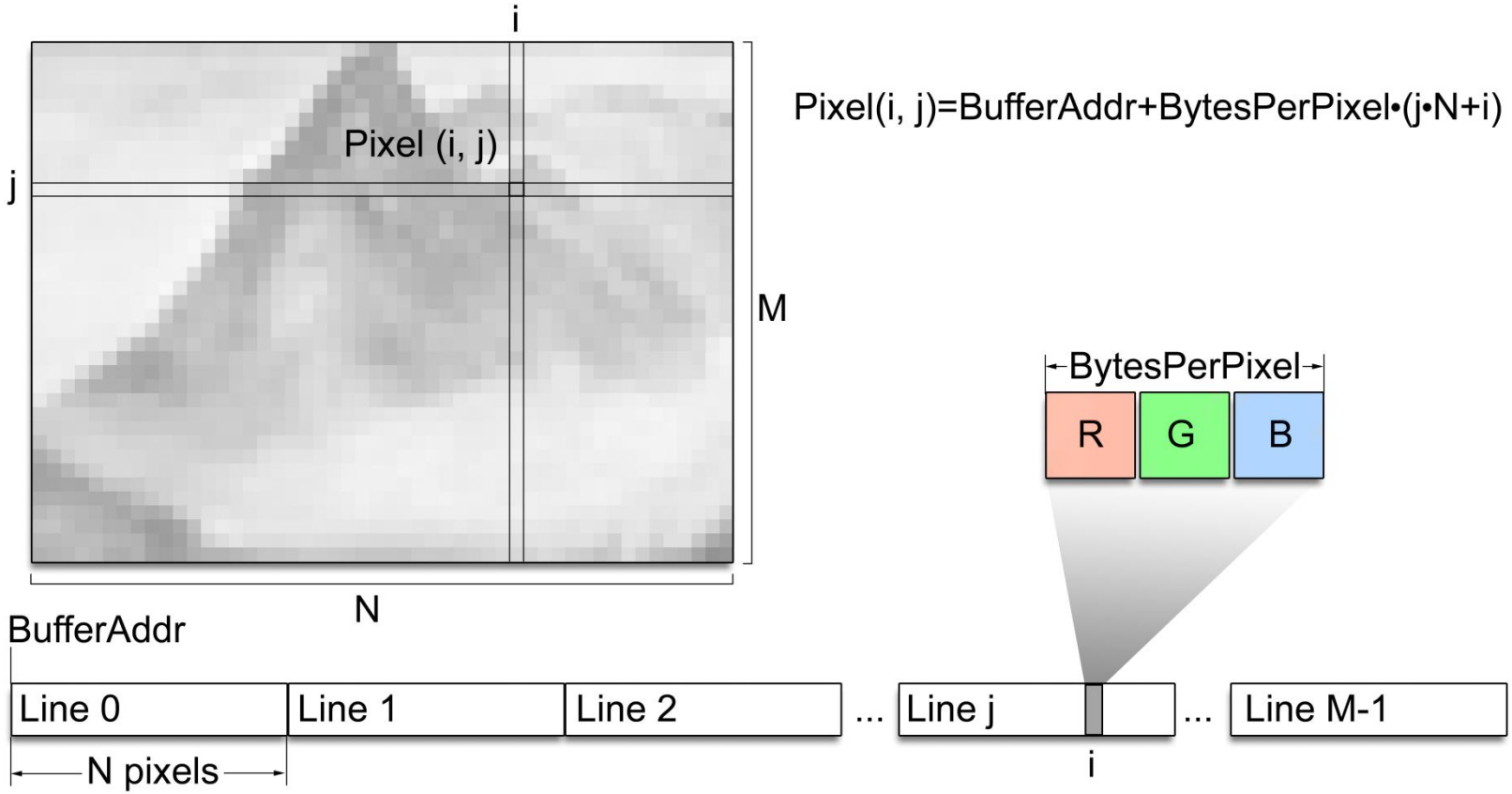
Storage

- Separate color channels per pixel
- Impact of color depth



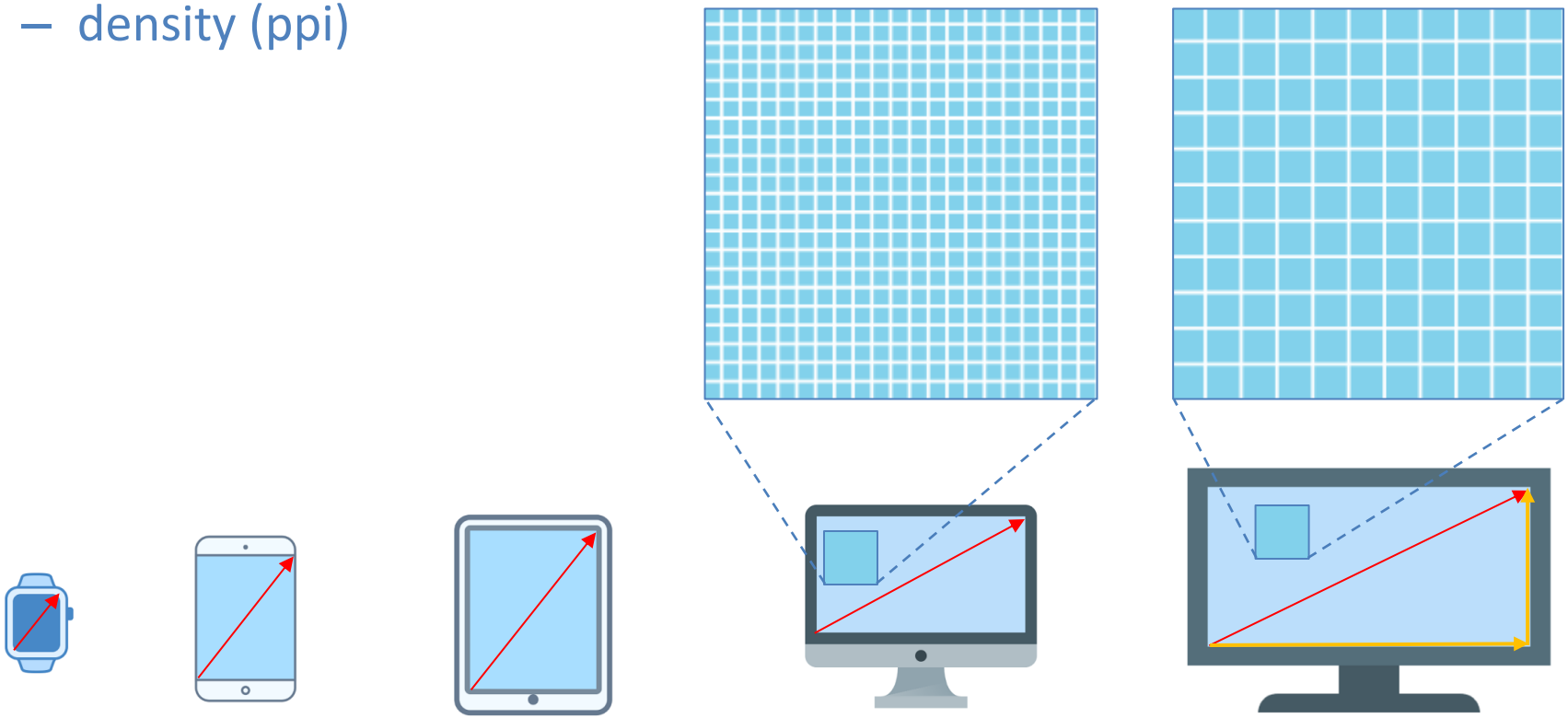
The Image Buffer

- Typically stored as an array in memory
- Interleaved color channels
- Line/column configuration, but also in blocks



The Screen Display

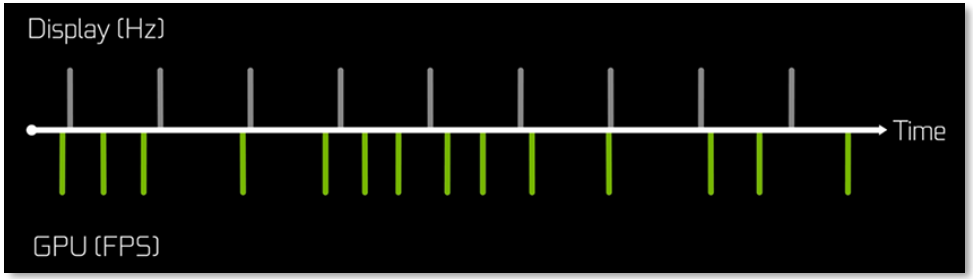
- Output device that displays information in pictorial form
  - size (inch)
  - resolution (width, height)
  - density (ppi)



Frame Rate vs Refresh Rate

Rate of image completion (FPS)

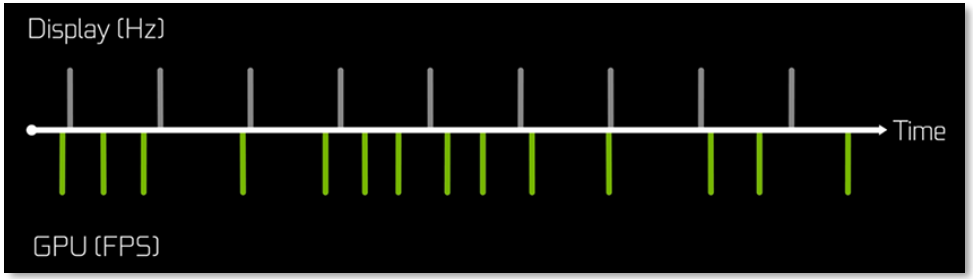
Rate of screen display (Hz)



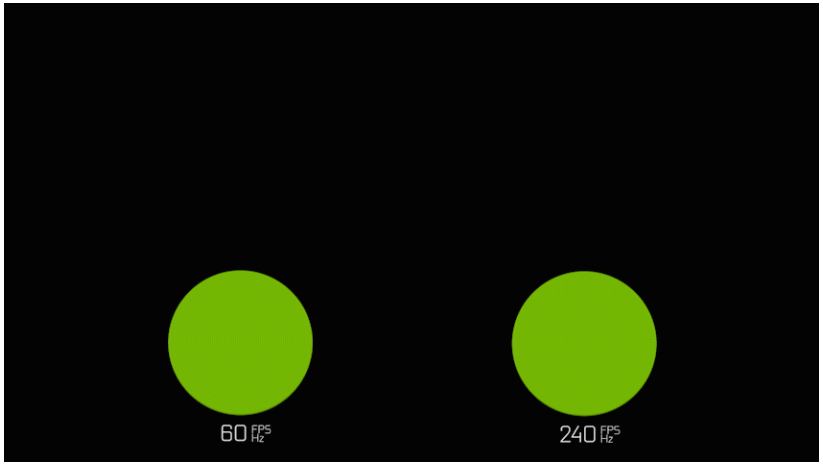
Frame Rate vs Refresh Rate

Rate of image completion (FPS)

Rate of screen display (Hz)



smoothness & ghosting



tearing



Dynamic Range

- Dynamic range: the minimum to maximum luminance level achieved by a system
- HVS range: 10^{8}:10^{-6}
- H/W cannot achieve these levels simultaneously!
  - We use *tone mapping* to adjust the "useful" range to match the output range of a device
- Physically measured or simulated radiance (therefore luminance) in a natural environment matches the HVS levels
- Typical displays can achieve a **dynamic contrast ratio** of 6000:1 and an **actual luminance level** of 1-300cd/m^2

High Dynamic Range Images

- Use floating point arithmetic representation to store a wide range of luminance values
- Used both in CG and photography
- Typical integer buffers and image formats (8 bits per color channel) are not enough
- Precision depends on storage limitations and application: unsigned bytes: 24bit color, floating point (half/full): 48/96bit images
- HDR screens use a combination of 8bpp panels and temporal dithering to increase the perceived levels.

The Frame Buffer

The area of memory that stores the resulting fragments during rendering

Can either represent:

- The final displayed frame
- Intermediate results or extra data (e.g. depth), later to be used as texture inputs



Intermediate Frame Buffer (render-to-texture)



Final Frame Buffer with Color Grading

Rendering Types

Photorealistic: Generate images indistinguishable from real photographs



*Image credit: *rjonescgi.wordpress.com*



*Image credit: *wikipedia.org*

Rendering Types

Non-Photorealistic: Generate images for visualization purposes or based on artistic freedom photographs

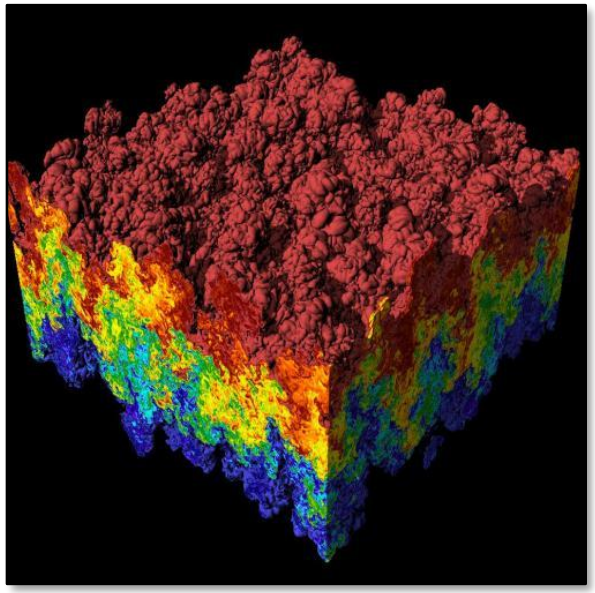


*Image credit: *wikipedia.org*



*Image credit: *polycount.com*



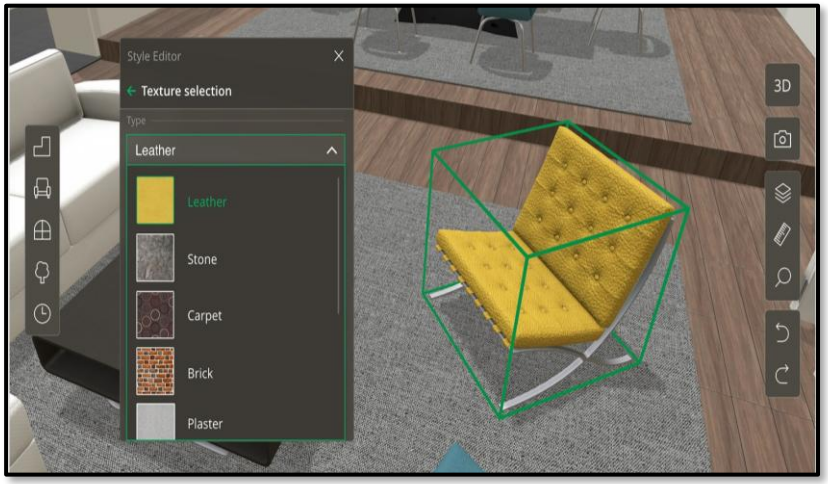
*Image credit: *community.thefoundry.co.uk*

Rendering Spectrum

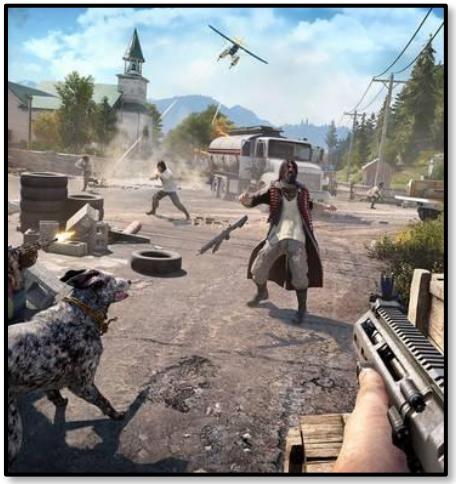
Movie



Interior Design



Game



Offline **Interactive** **Real-time**



Rendering Spectrum

Rendering Spectrum

- Zero interactivity requirements
- Focus on highest possible degree of quality
- Up to hours per frame
- Computer clusters (e.g. render farms)



*Image credit: *imdb.com*



Rendering Spectrum

Rendering Spectrum

- Focus on highest possible degree of fully interactivity, immersion & dynamic content
- Small time per frame generation (60 fps)
- Commodity hardware and consoles



*Image credit: *Ubisoft*, *Far Cry 5*



Rendering Spectrum

Rendering Spectrum

- Balance between interactivity & quality
- Up to a second per frame
- Manipulation of content is important

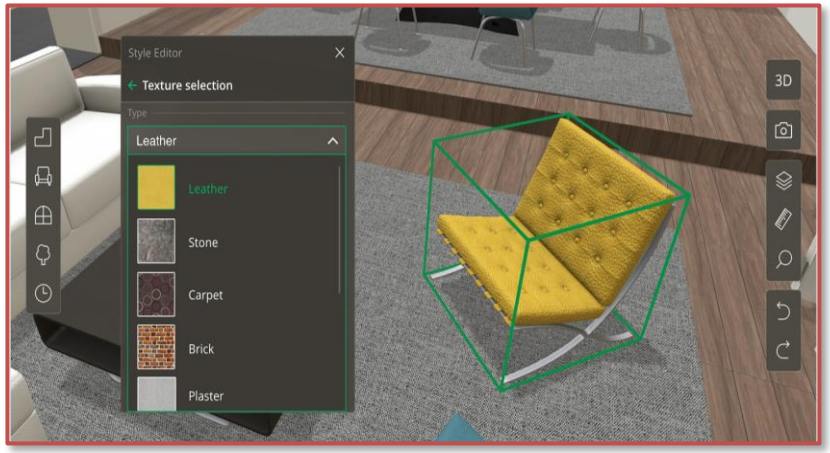


*Image credit: *planner5d.com*

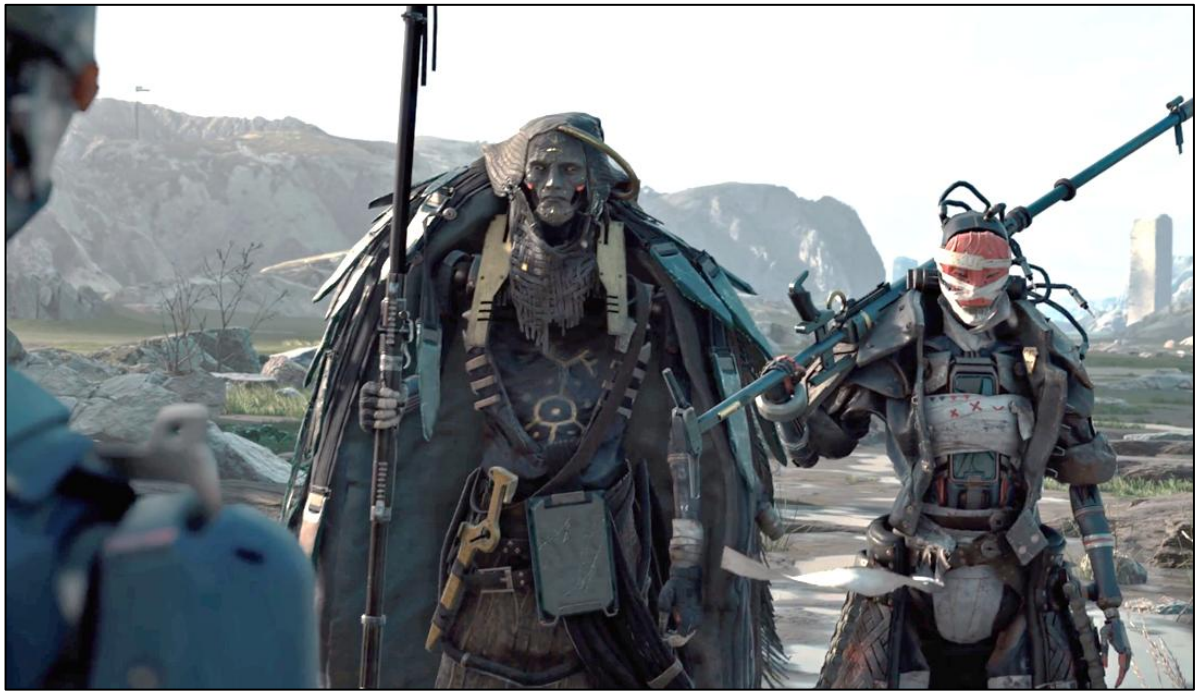
Offline **Interactive** **Real-time**



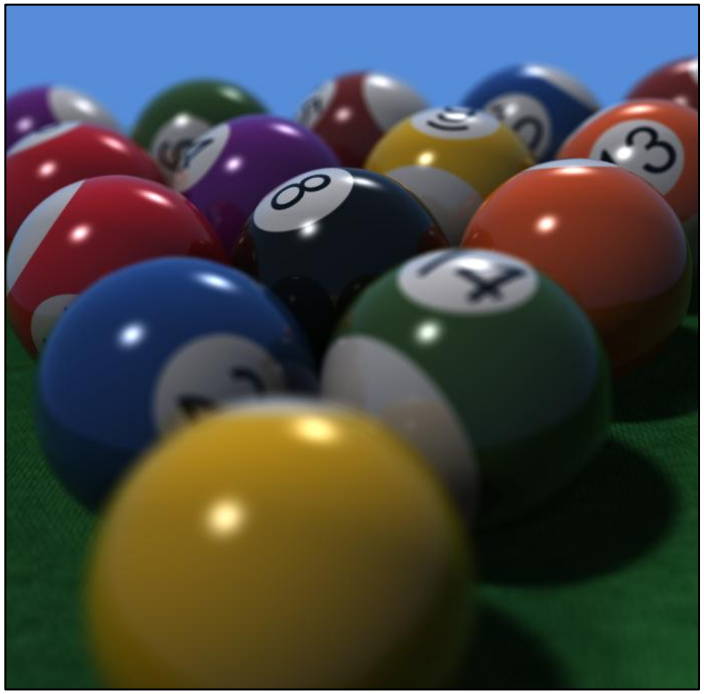
Rendering Spectrum

Where is the Borderline?

- Increasingly fast GPUs and the many-core implementation of "traditionally" offline algorithms blur the border
- Still, physically correct and high-quality rendering of complex environments are offline



Computer game (Unity 5 demo)



Real-time ray tracing of a simple scene

What About 2D Rendering?

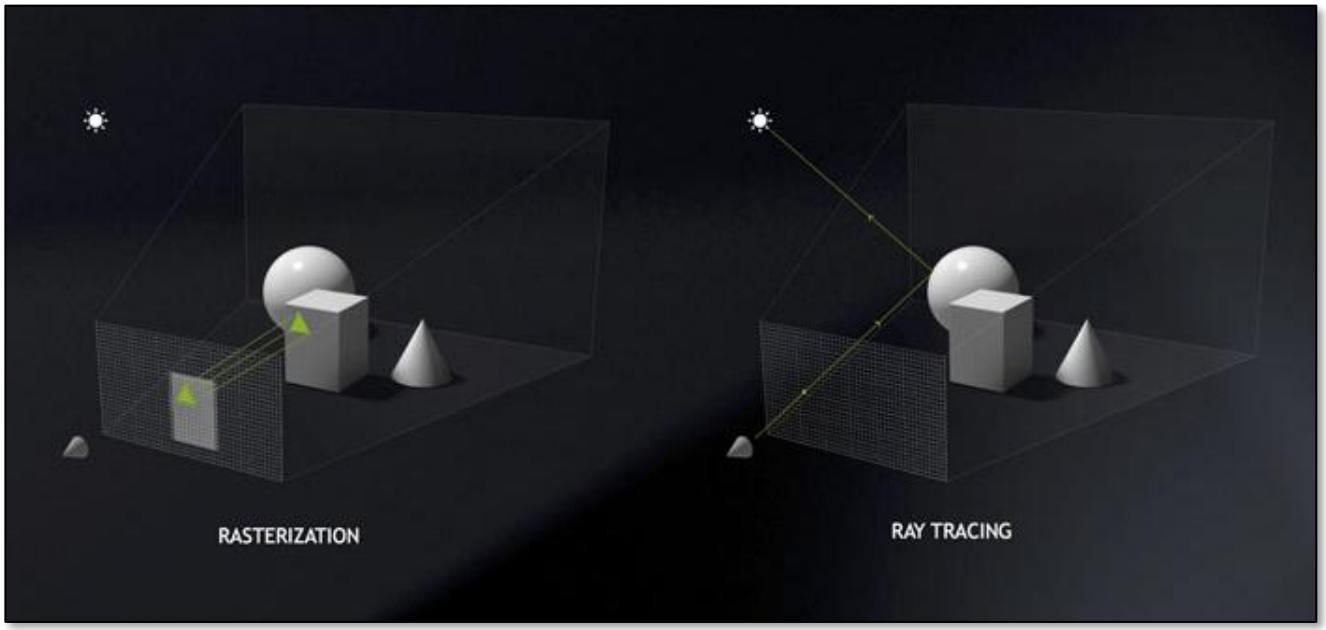
Today, most hardware-accelerated 2D drawing is handled via the 3D h/w pipeline!

- Textured polygons and framebuffers for views and bitmap storage: fast access and redraw, easy transitions and effects
- Blending and widget ordering handled by the hidden surface removal and blending of the GPU
- Shape rasterization and fills via GPU rasterization

Graphics Pipelines

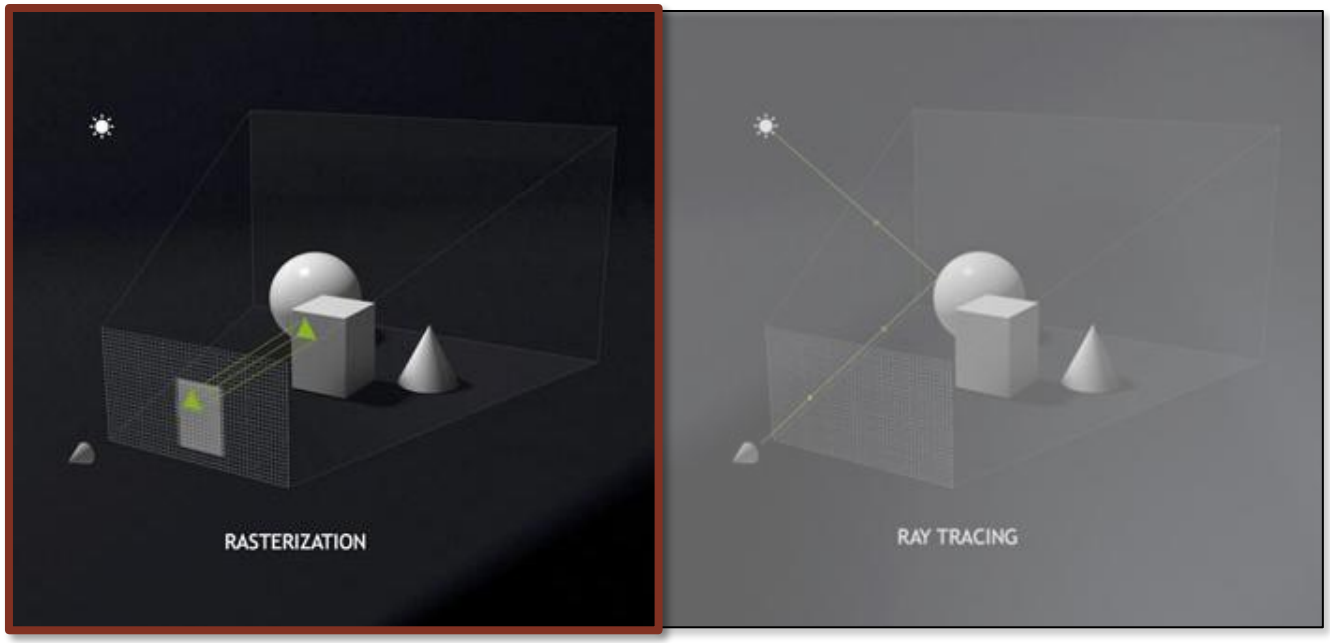
- "Graphics Pipeline" is the sequence of steps that we use to create the final image

- Many graphics/rendering pipelines have been proposed



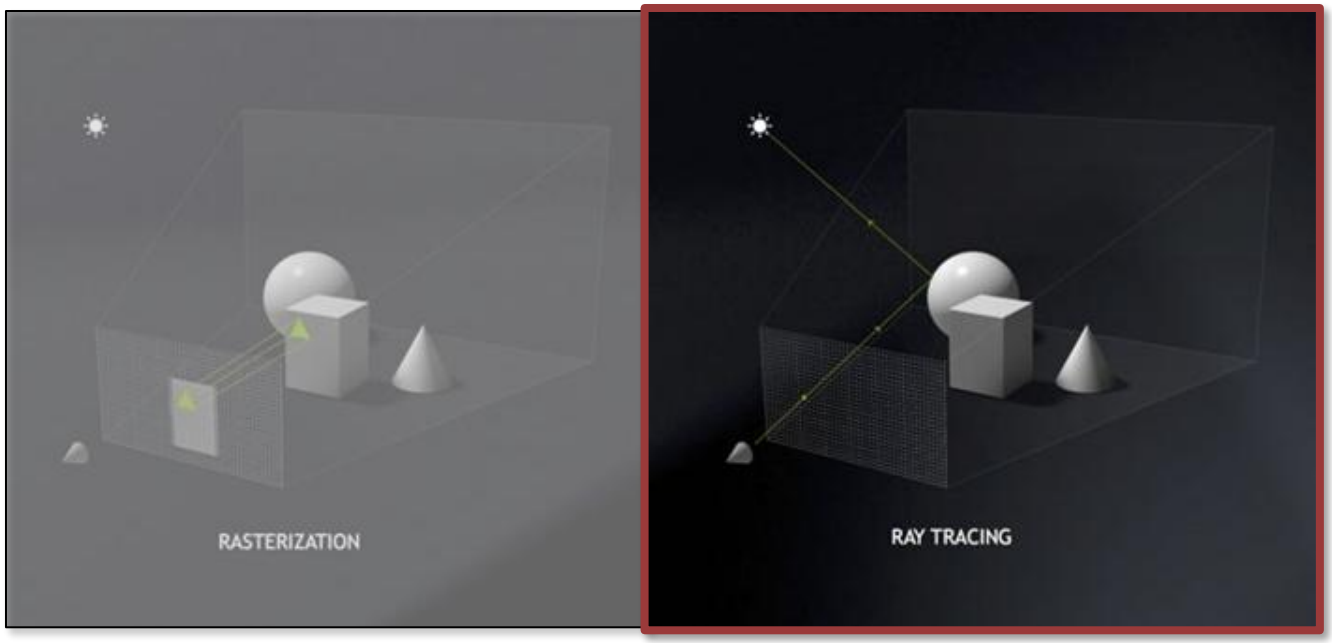
Graphics Pipelines - Rasterization

- Project all primitives to the image plane, sample them & store result.



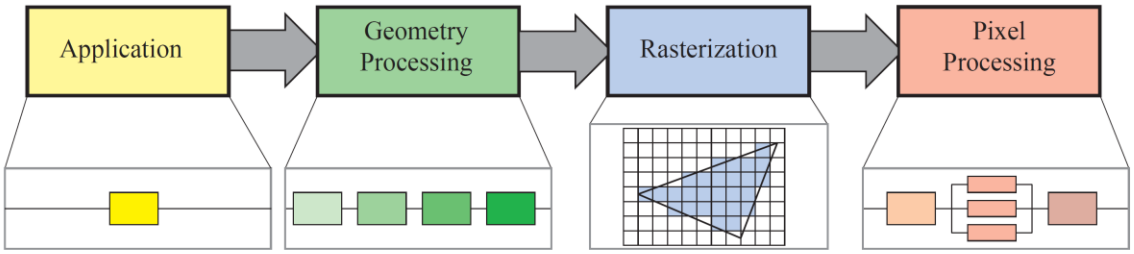
Graphics Pipelines – Ray Tracing

- For each pixel, spawn rays and check for intersections against primitives.
- Recursively follow the trajectory of rays to compute the final color.



Graphics Pipelines

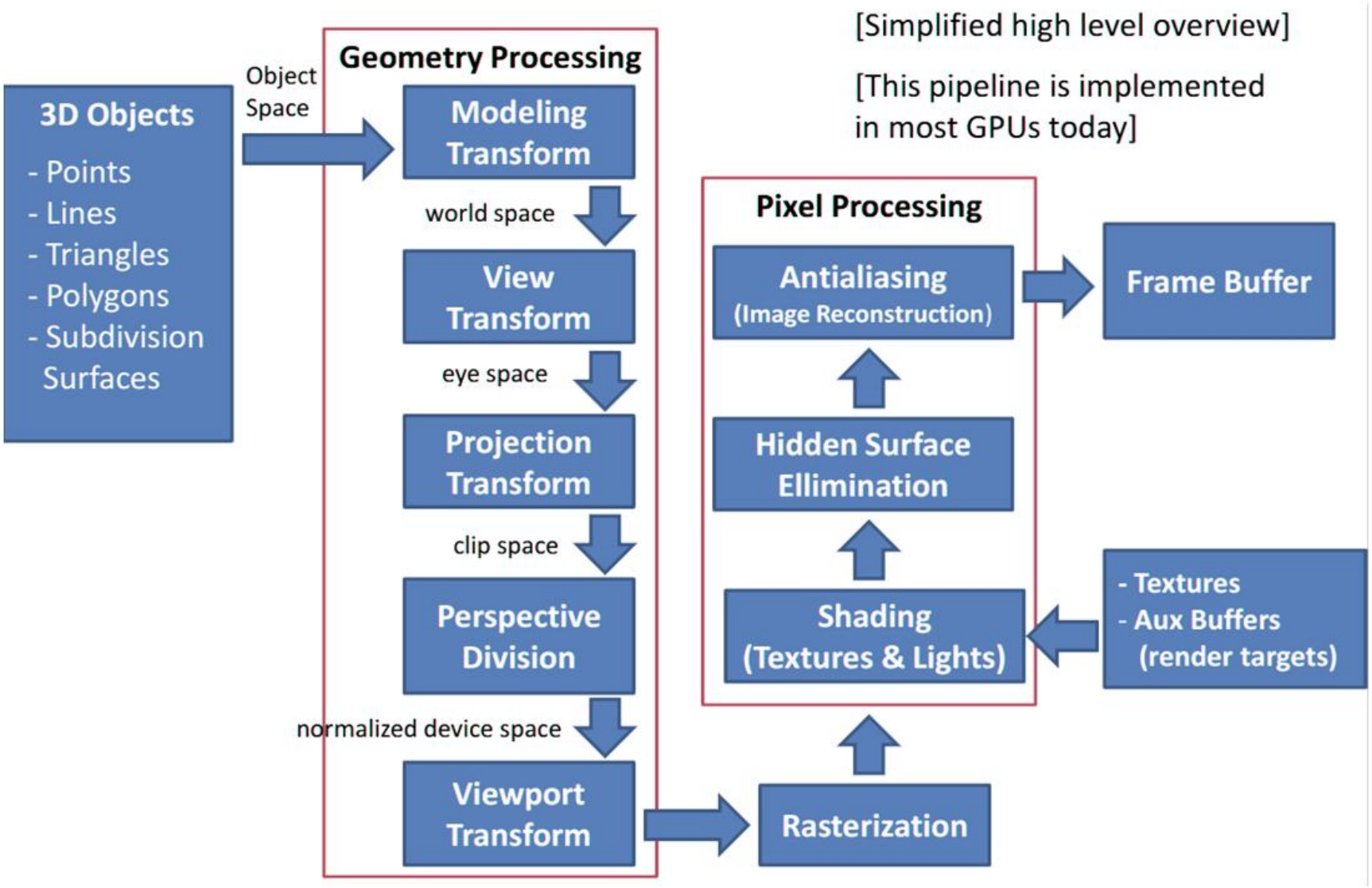
- **Software** Or **Hardware**
- Stages: Configurable, Programmable, Fixed



*Images credit: *Real-Time Rendering book*

- How to Code
  - OpenGL/ES/WebGL -> Vulkan/WebGPU
  - DirectX, Metal
  - Optix, Embree

GPU Rasterization Pipeline



In a nutshell…

- Self-Introduction
- Basic Concepts
- Course Description
- Fundamentals



Q/A

Computer Graphics





abasilak@aueb.gr

Acknowledgements

Many thanks to *Prof. G. Papaioannou* for the slides, material and guidance given for the preparation of this course.