Introduction and Overview

CS425: Computer Graphics I

Fabio Miranda

https://fmiranda.me

Overview

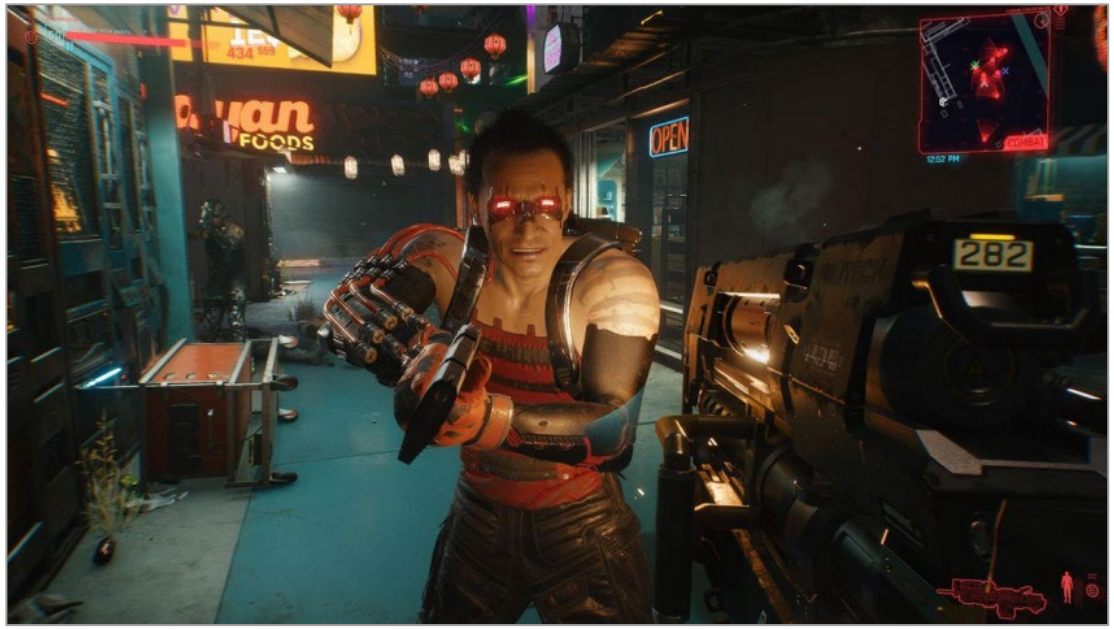
- Intro to computer graphics
- Course goals
- Logistics
- Requirements
- Content overview
- Grading
- Assignments
- Policy
- Workflow pattern
- Resources
- Communication

What is Computer Graphics?

- Broad sense: use of computer to create and manipulate images
- Combination of hardware (input, processing, output) and software
- 2D or 3D

Example

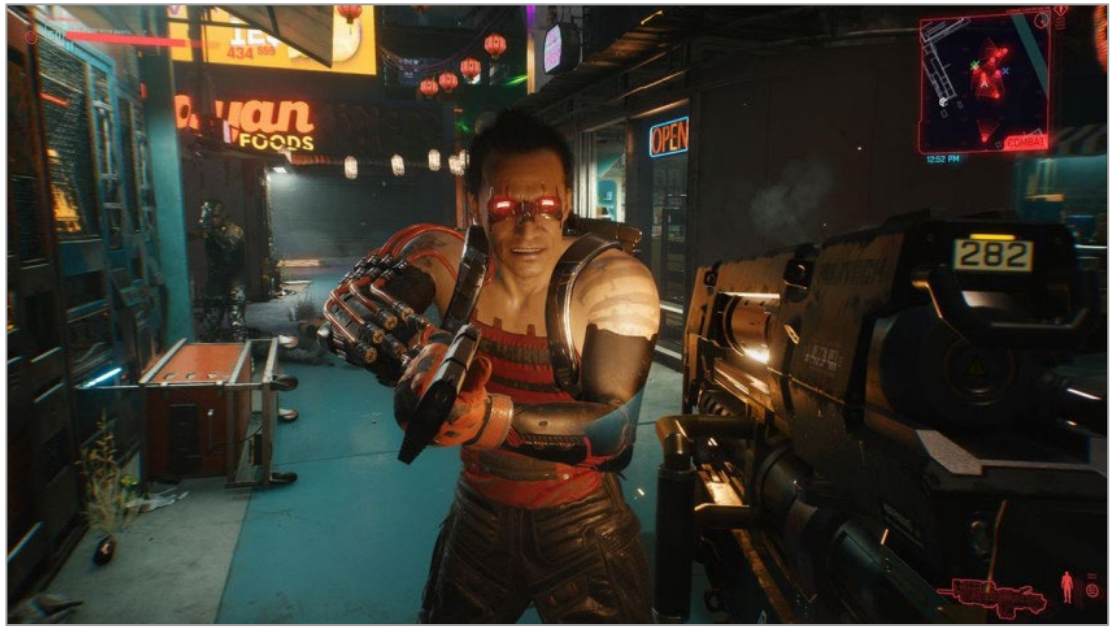
Cyberpunk 2077



Google Earth

Example

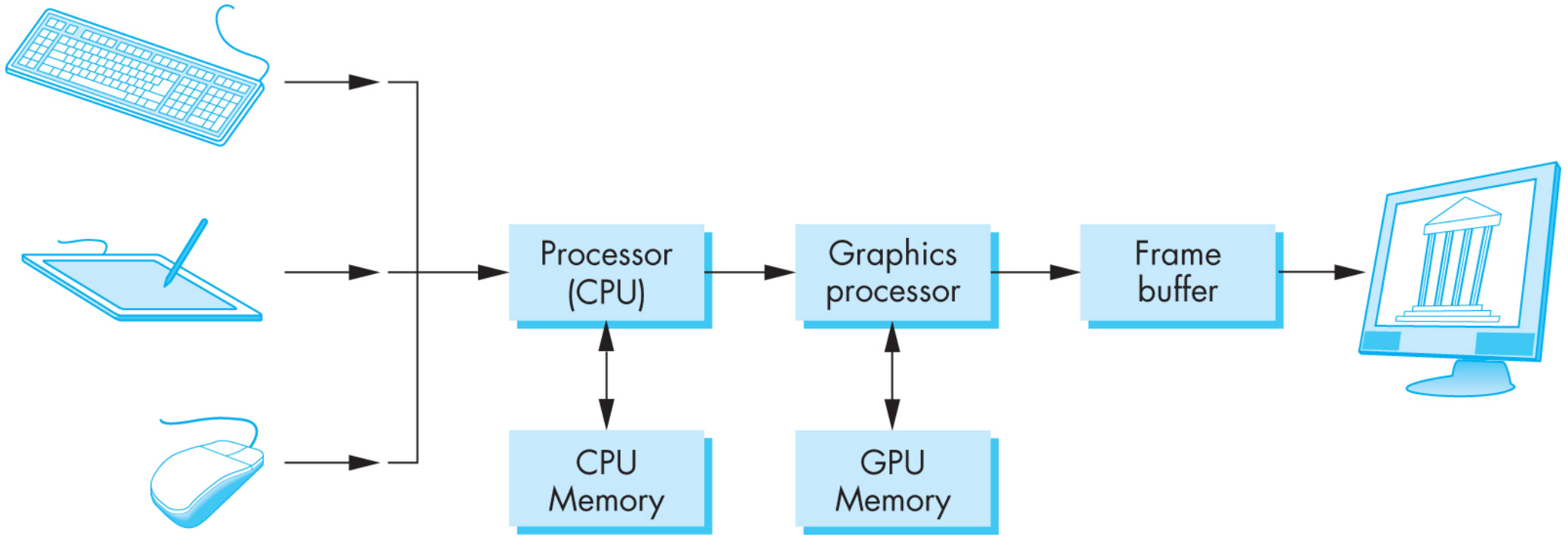
How was this image created?



Graphics system

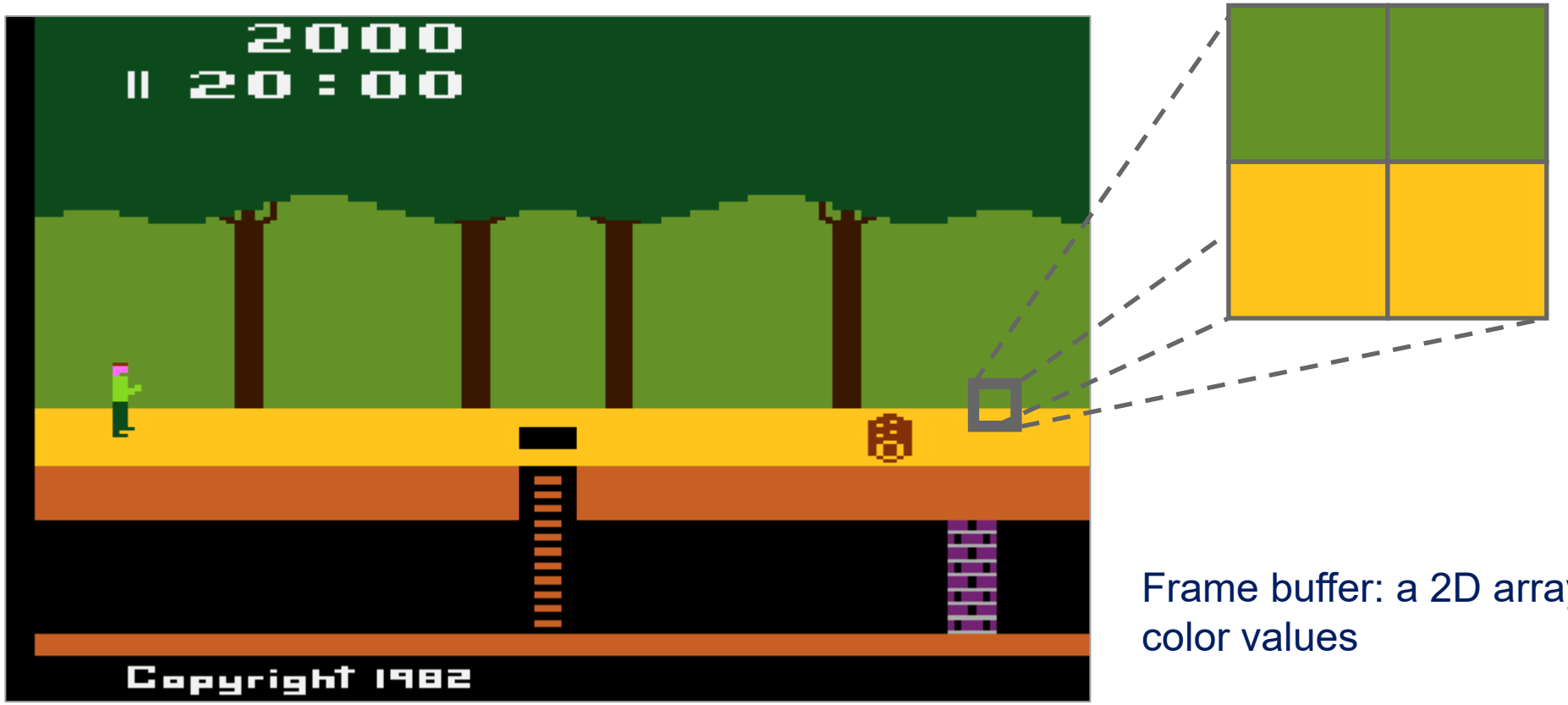
Urbane, OpenSpace, ArcGIS, Google Earth, …	Cyberpunk, Doom, FIFA, …
	Unreal, Unity, idTech, Frostbite, REDengine, …
OpenGL, DirectX, Vulkan, Metal, Glide, …	
Windows, Linux, iOS, …	
CPU, RAM, GPU, VRAM, …	

Basic graphics system



From: Interactive Computer Graphics 7th Ed by Professor Ed Angel and Dave Shreiner

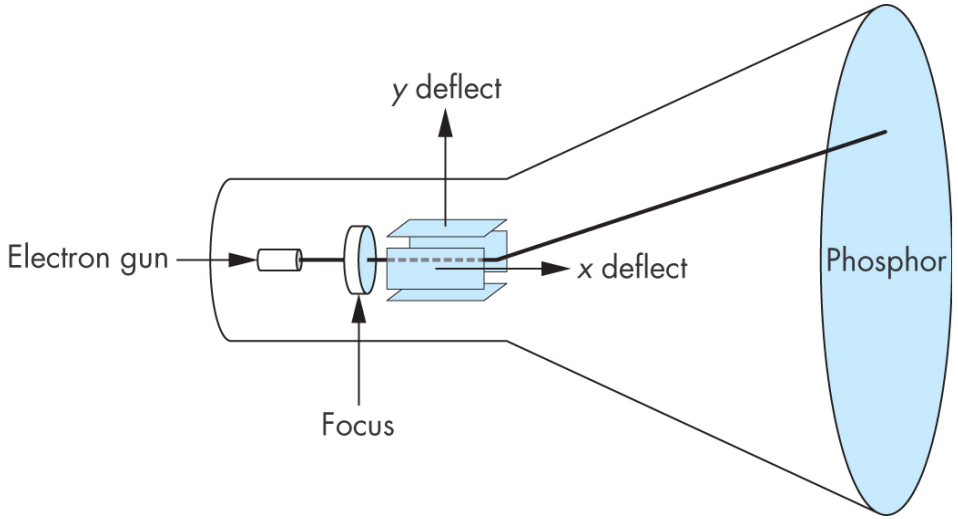
Basic graphics system



Frame buffer: a 2D array with color values

Computer Graphics: 1950s

- Introduction of cathode-ray tube (CRT) as a viable display
- Light pen as input



Computer Graphics: 1960s

- The phrase “computer graphics” is coined by Verne L. Hudson and William Fetter from Boeing.
- Wireframe graphics.
- Sutherland introduces Sketchpad, the first computer graphical user interface:
  - Precise drawing
  - Erase, move
  - Zoom in and out
- First ray casting algorithm by Arthur Appel

Computer Graphics: 1960s

"The Sketchpad system uses drawing as a novel communication medium for a computer. The system contains input, output, and computation programs which enable it to interpret information drawn directly on a computer display. It has been used to draw electrical, mechanical, scientific, mathematical, and animated drawings; it is a general purpose system. Sketchpad has shown the most usefulness as an aid to the understanding of processes, such as the notion of linkages, which can be described with pictures. Sketchpad also makes it easy to draw highly repetitive or highly accurate drawings and to change drawings previously drawn with it."

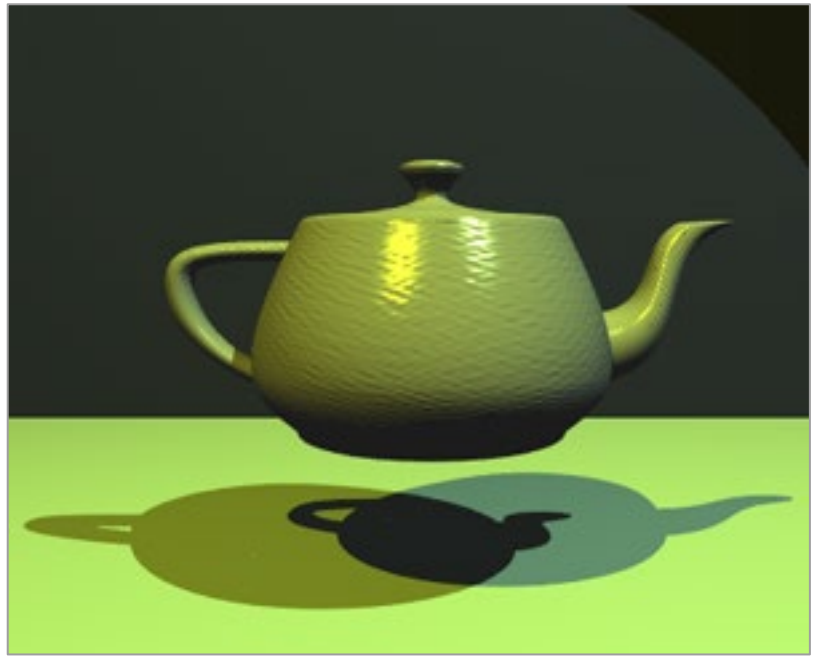
Sketchpad: A man-machine graphical communication system

By Ivan Sutherland

1963

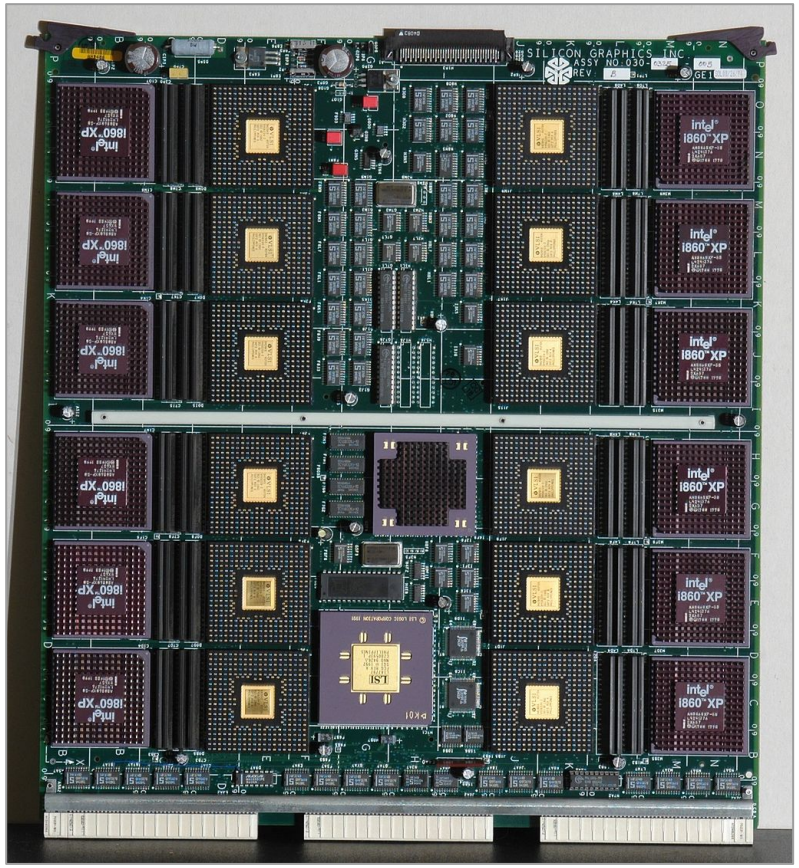
Computer Graphics: 1970s

- Beginning of graphics standards
  - GKS: ISO 2D standard
  - Core: 3D, not ISO standard
- Pong, the first commercially successful video game
- First commercial framebuffer
- Utah teapot
- Gouraud and Blinn-Phong shading models
- Workstations and PCs



Computer Graphics: 1980s

- Special purpose hardware
  - Silicon Graphics geometry engine
  - SGI: 3D raster graphics hardware
- Shaded solids, no textures



Computer Graphics: 1990s

- OpenGL API
- Toy Story: First computer-generated feature-length movie
- GeForce 256: first consumer-level with hardware-accelerated transforming and lighting
- Texture mapping
- Doom, Quake

Computer Graphics: 2000s

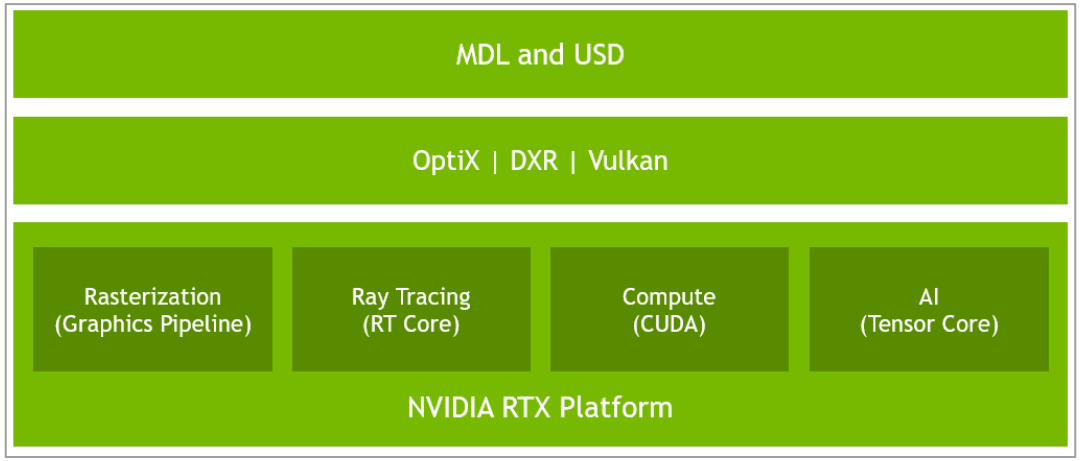
- GeForce 3: first to support programmable shaders
- Proliferation of CG movies
- GPU being used in other area: computer vision, machine learning, bioinformatics, etc.
- CUDA, OpenCL

Computer Graphics: 2010s

- Graphics is now ubiquitous
  - Cell phones
  - Embedded
- OpenGL ES and WebGL
- Virtual Reality
- 3D movies and TV

Computer Graphics: 2020s

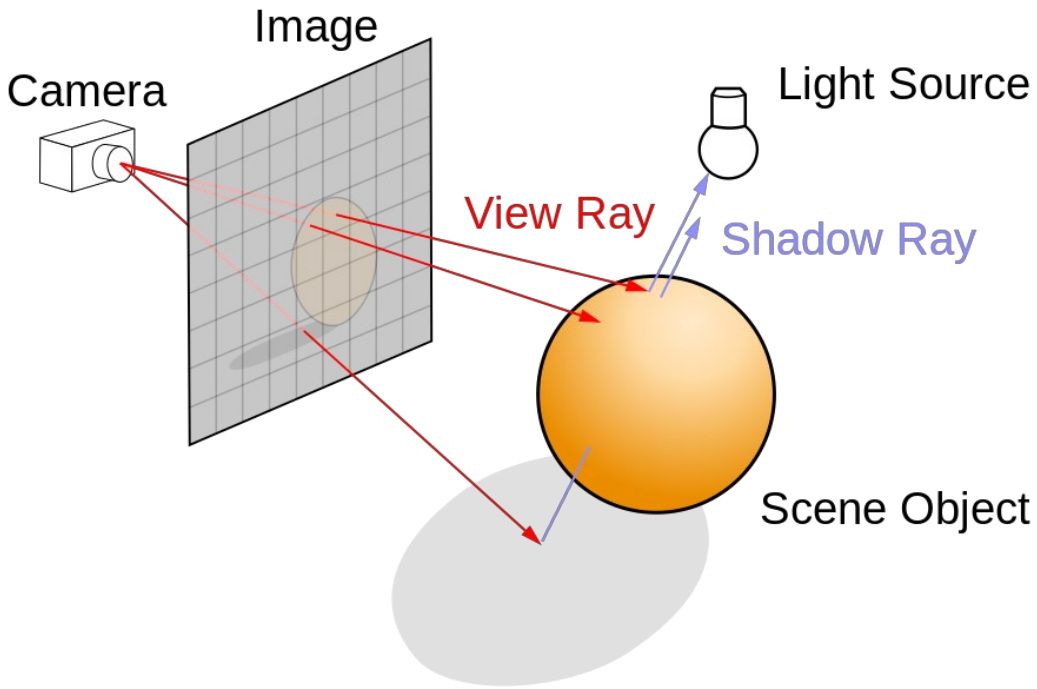
- Real-time ray tracing
- 4K resolution
- AI-based anti-aliasing



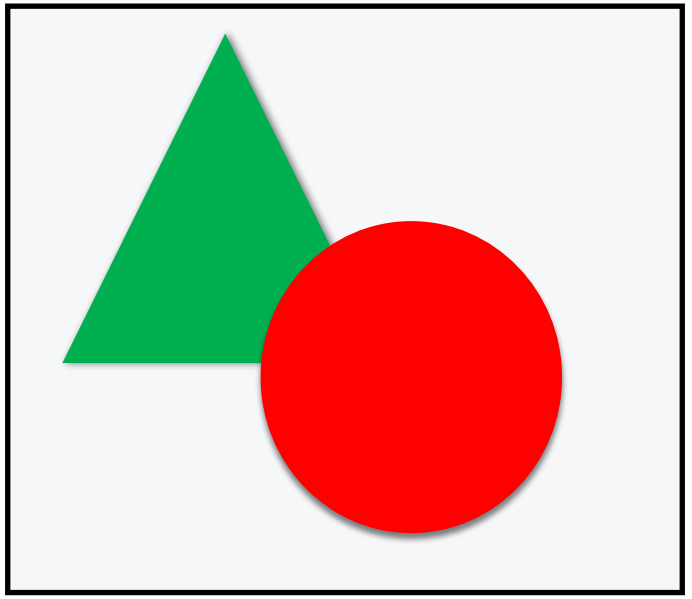
Nvidia - Reflections RTX Tech Demo

Creating images

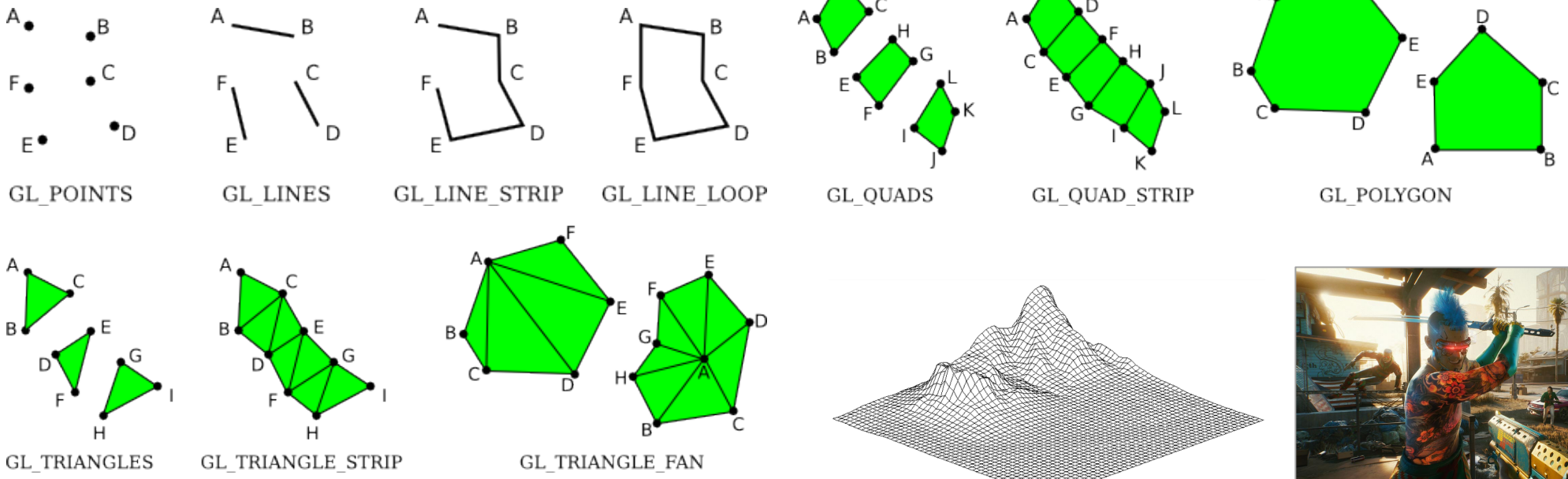
Per pixel: ray tracing



Per object: rasterization

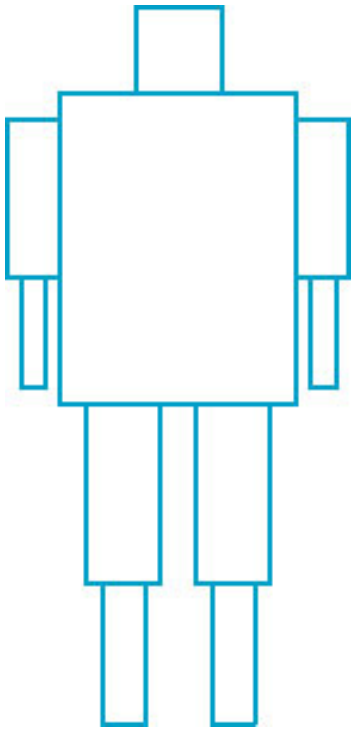


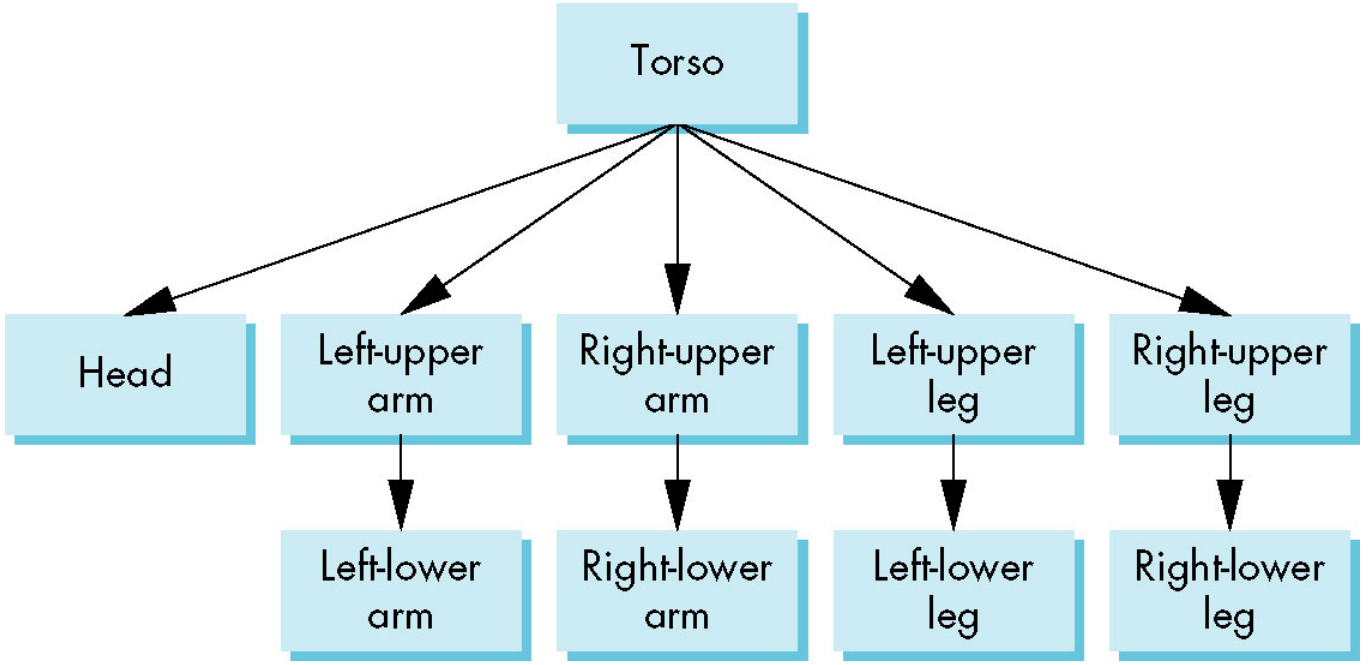
Creating images: Objects description



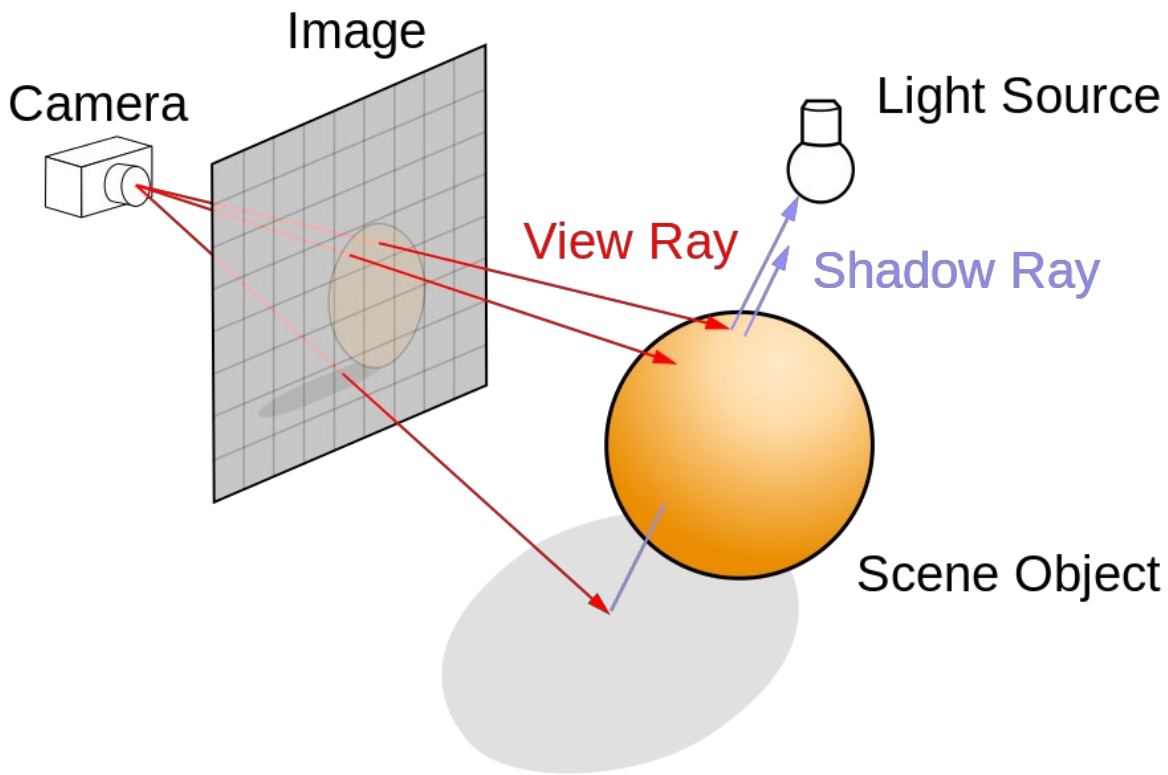
From: Introduction to Computer Graphics, David J. Eck

Creating images: Scene description



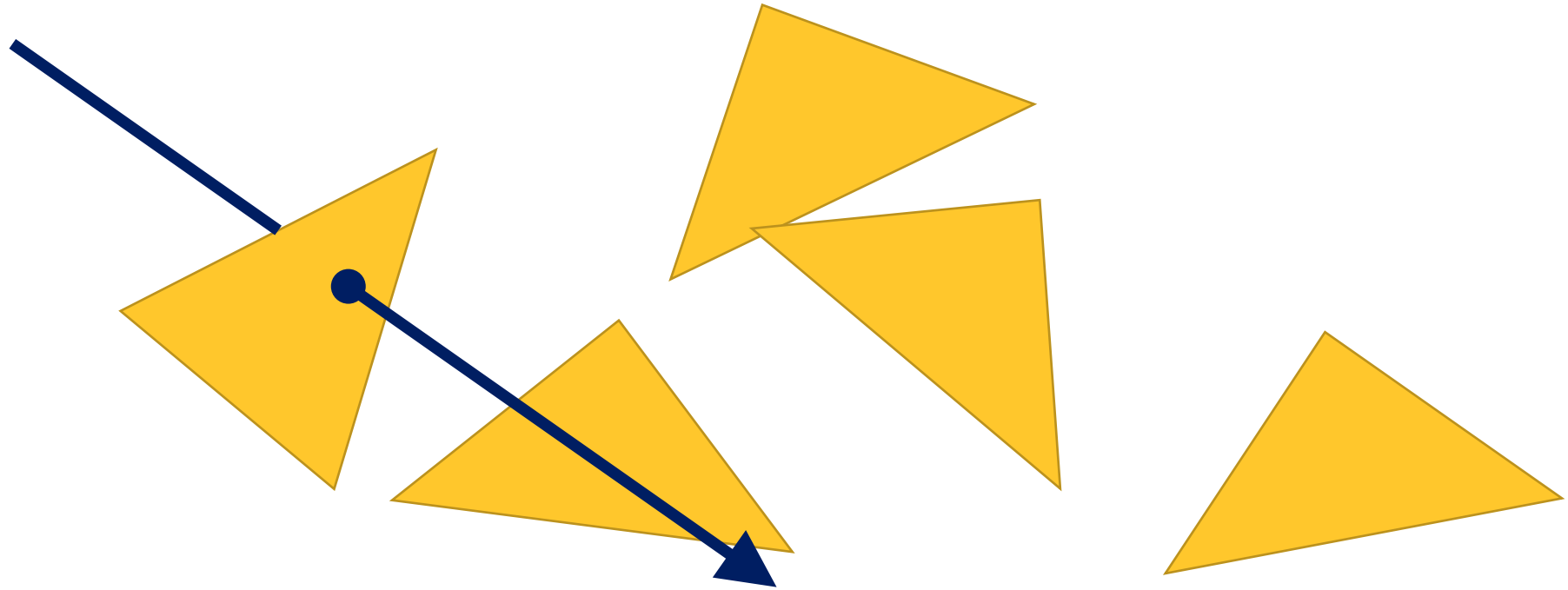


Creating images: Ray tracing

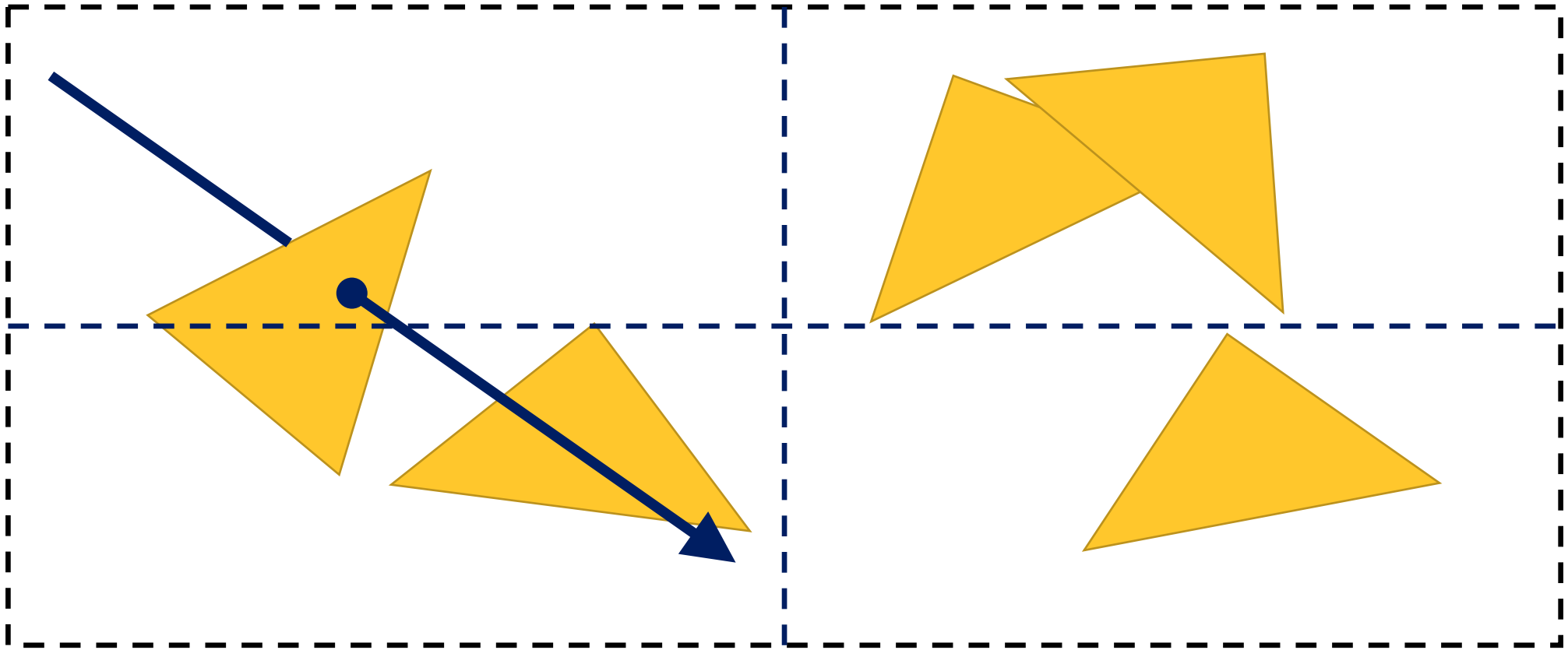


- Easy to parallelize but hard to map to hardware (up until recently)
- Expensive!
- It can be extended to model many physical phenomena (internal scattering, diffraction, reflections, etc)
- Used to obtain high-quality images

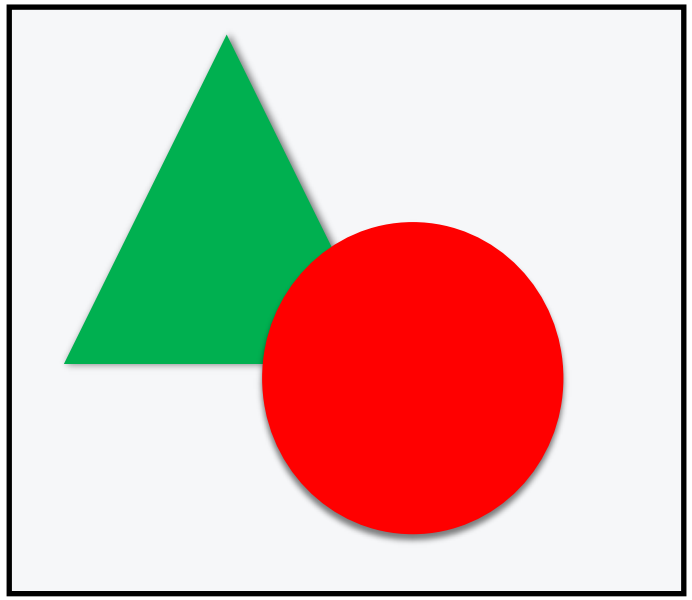
Creating images: Ray tracing



Creating images: Ray tracing

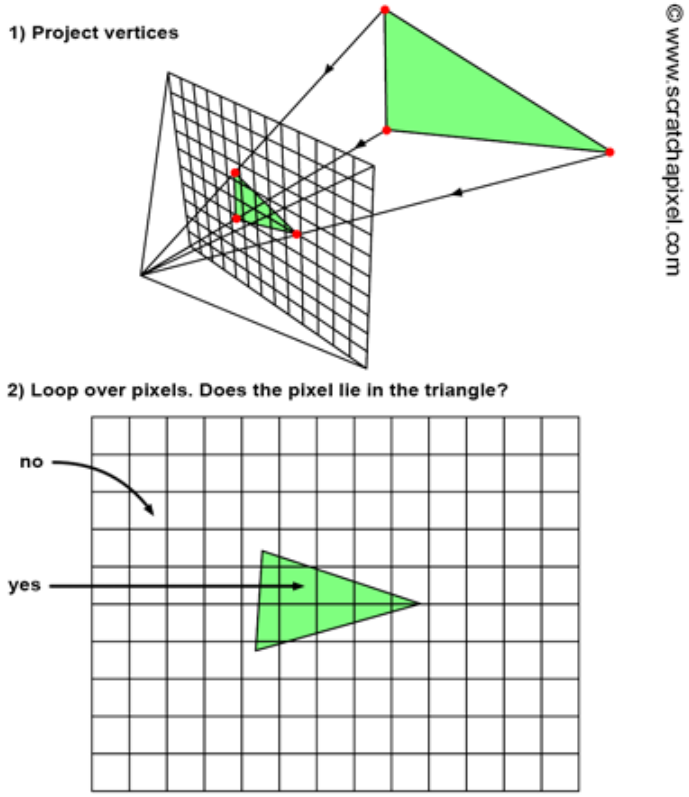


Creating images: Rasterization



- Easy to map to hardware
- While it cannot model directly complex effects, we can approximate them
- Used in interactive applications (mostly)

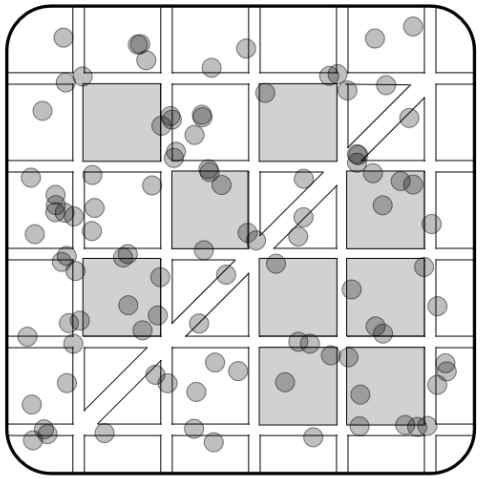
Creating images: rasterization

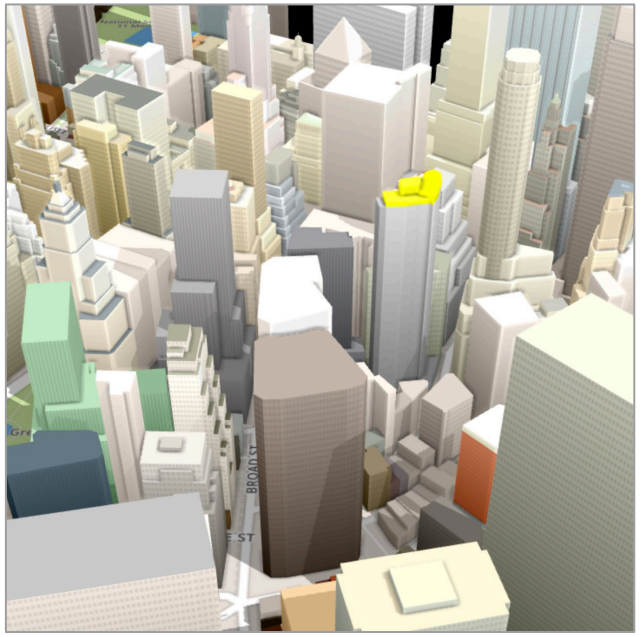


1. Project 3D vertices onto the screen.
2. Loop through pixels in the image and test if they lie within the resulting 2D triangles.

Real-time rendering vs off-line rendering

Computer Graphics & Visualization

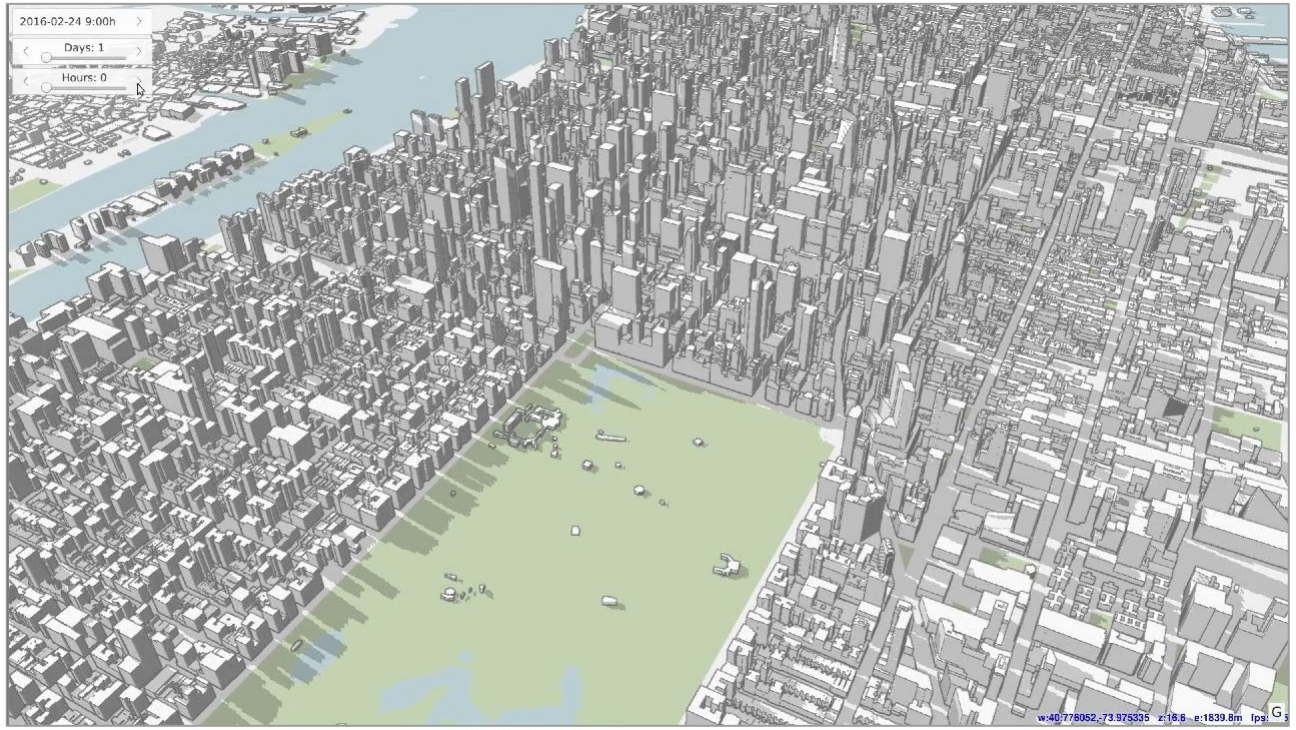




Open Street Maps

DoITT

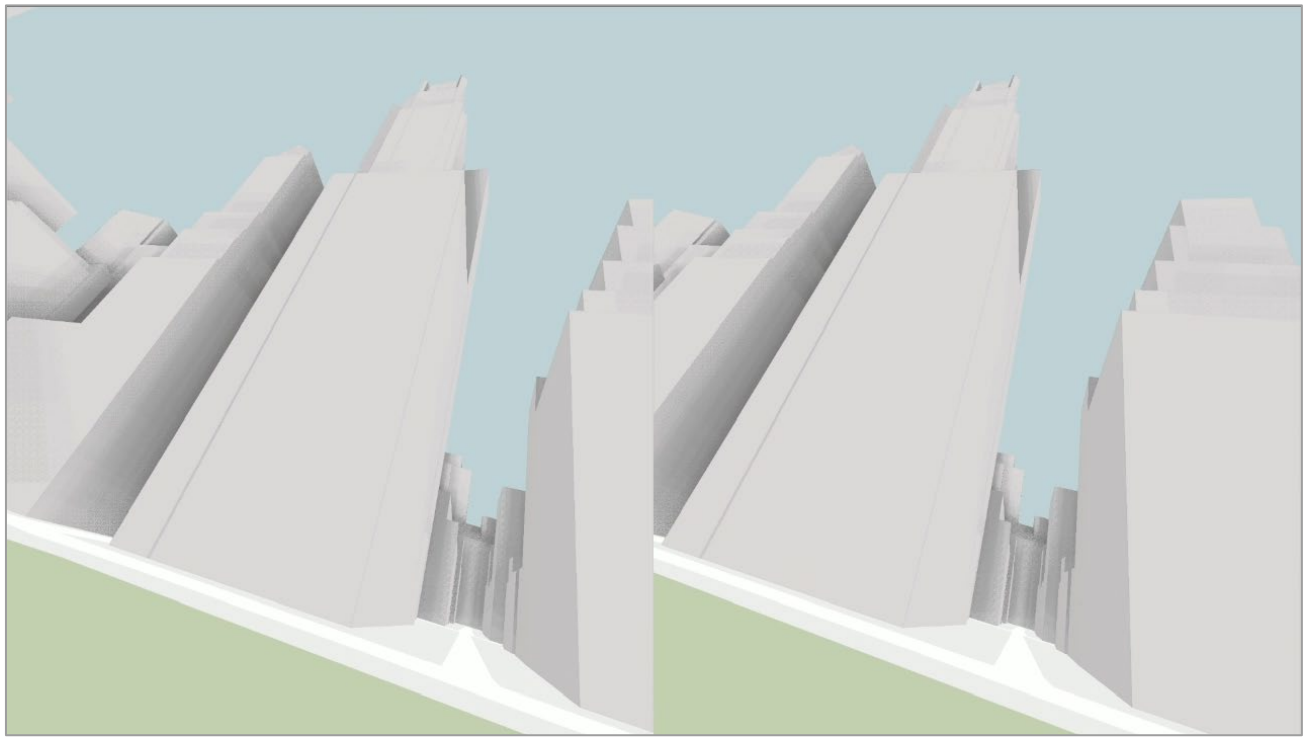
Computer Graphics & Visualization



Shadow Accrual Maps: Efficient Accumulation of City-Scale Shadows Over Time

https://nyti.ms/2k0bF0G

Computer Graphics & Virtual Reality



Urbanrama

Computer Graphics & Database

Interactive Visual Exploration of Spatio-Temporal Urban Data Sets using Urbane

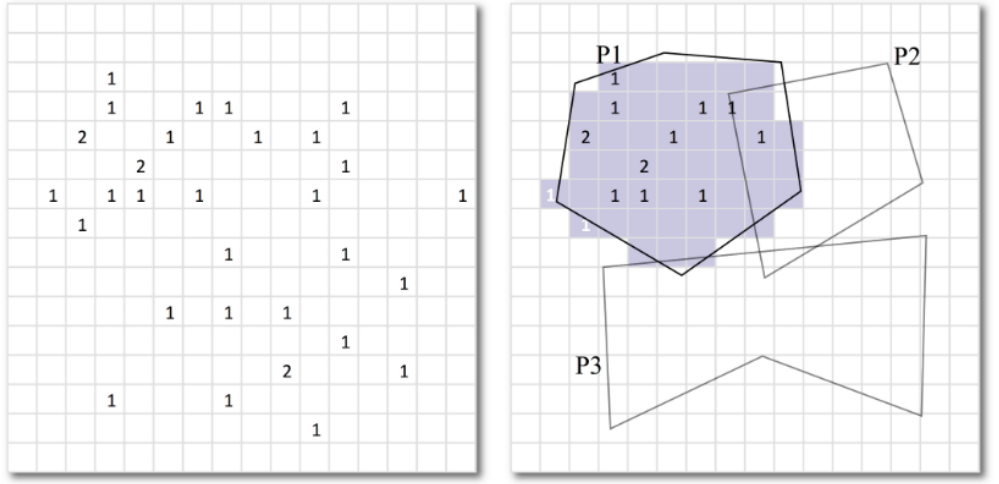
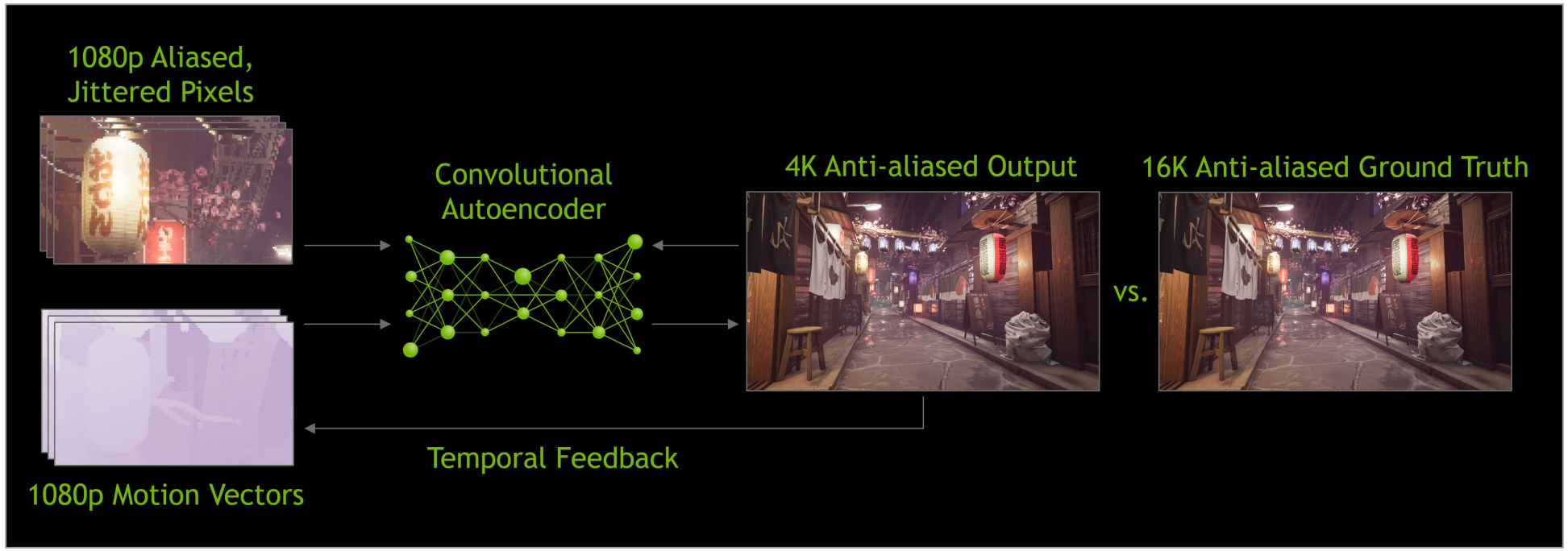


Figure 2: Raster Join first renders all points onto an FBO storing the count of points in each pixel (a). It then aggregates the pixel values corresponding to polygon fragments (b).

Computer Graphics & Machine Learning



NVIDIA: Deep learning super sampling

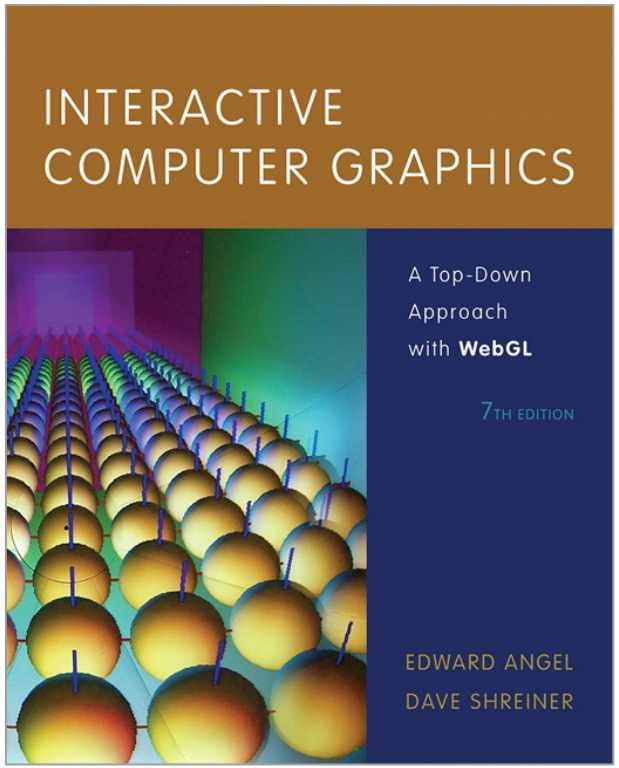
CS425: Course goals

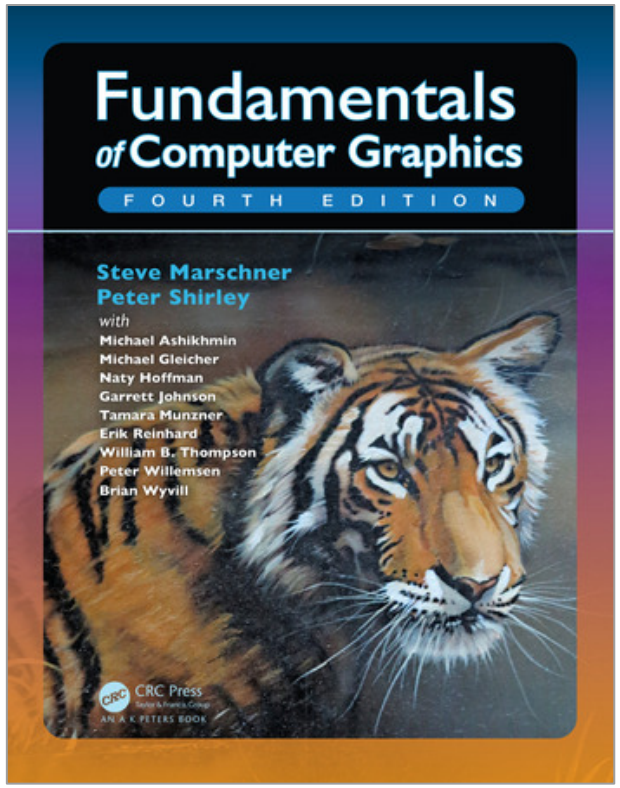
- Explain and apply the core concepts of computer graphics.
- Understand and use the different components of a programmable graphics pipeline.
- Understand and use different spatial data structures.
- Implement visual effects such as shadows.
- Implement interactive computer graphics programs using WebGL.

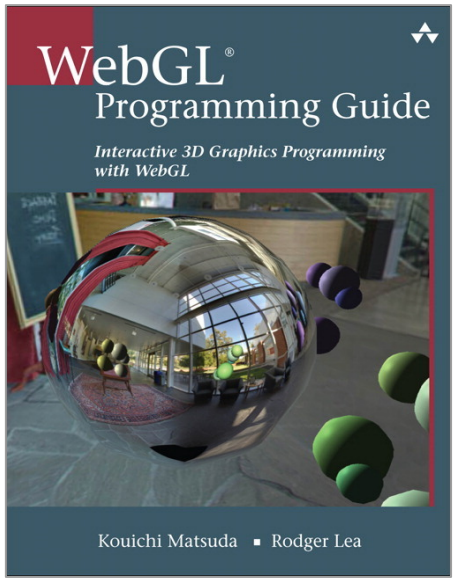
CS425: Course goals

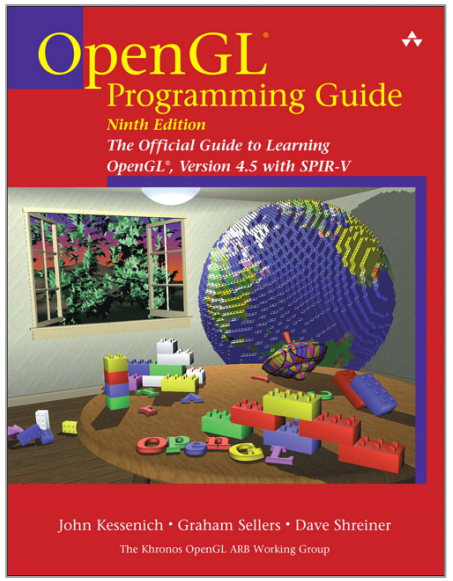
- Theory and systems behind applications
- Mathematics:
  - Physics of light, color, …
  - Geometry, perspective, …
- Systems:
  - Graphics APIs
  - Interaction devices

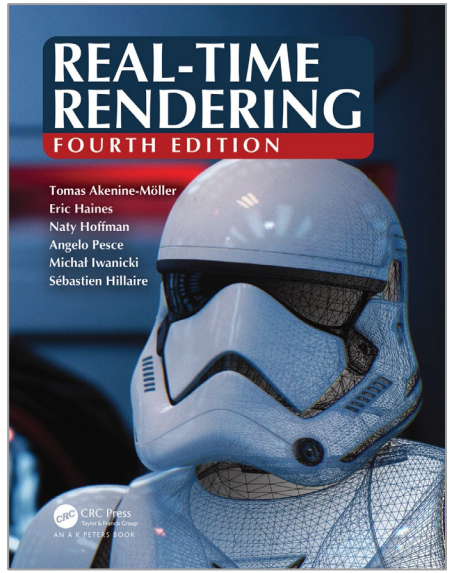
Material


INTERACTIVE
COMPUTER GRAPHICS
A Top-Down
Approach
with WebGL
7TH EDITION
EDWARD ANGEL
DAVE SHREINER


Fundamentals
of Computer Graphics
FOURTH EDITION
Steve Marschner
Peter Shirley
with
Michael Ashikhmin
Michael Gleicher
Naty Hoffman
Garrett Johnson
Tamara Munzner
Erik Reinhard
William B. Thompson
Peter Willemsen
Brian Wyvill
CRC Press


WebGL
Programming Guide
Interactive 3D Graphics Programming
with WebGL
Kouichi Matsuda ▪ Rodger Lea


OpenGL
Programming Guide
Ninth Edition
The Official Guide to Learning
OpenGL, Version 4.5 with SPIR-V
John Kessenich • Graham Sellers • Dave Shreiner
The Khronos OpenGL ARB Working Group


REAL-TIME
RENDERING
FOURTH EDITION
Tomas Akenine-Möller
Eric Haines
Naty Hoffman
Angelo Pesce
Michał Iwanicki
Sébastien Hillaire
CRC Press

Requirements

- CS251 Data structures.
- We will use WebGL and JavaScript.
  - Portability
  - No need to install dependencies or plugins to run code: just open a browser.
  - Polyglot programming: JavaScript, GLSL, HTML, CSS, …
  - Easy to produce interactive examples.

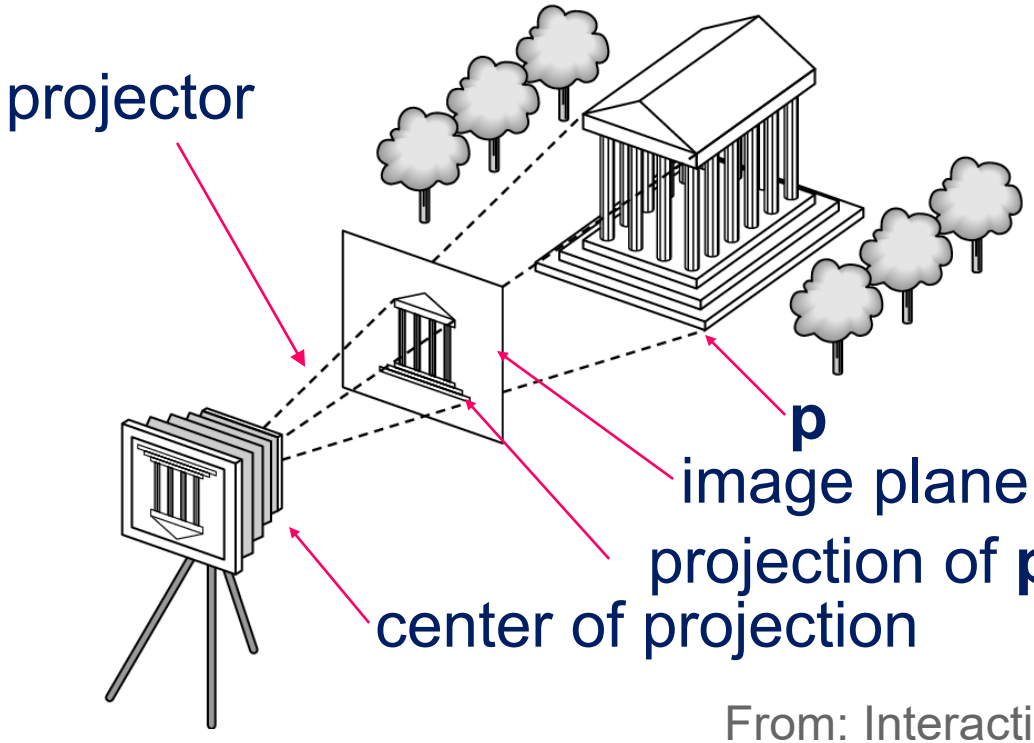
Logistics

- We will meet twice a week:
  - Tuesday 11:00am – 12:15pm (Central)
  - Thursday 11:00am – 12:15pm (Central)
- Synchronous classes on Zoom, with recordings available later.
- Use Zoom chat to ask questions, or to indicate you want to say something. **Please interrupt me at any time to ask questions.**
- Camera on or off, up to you.
- Discord or blackboard?

Content overview

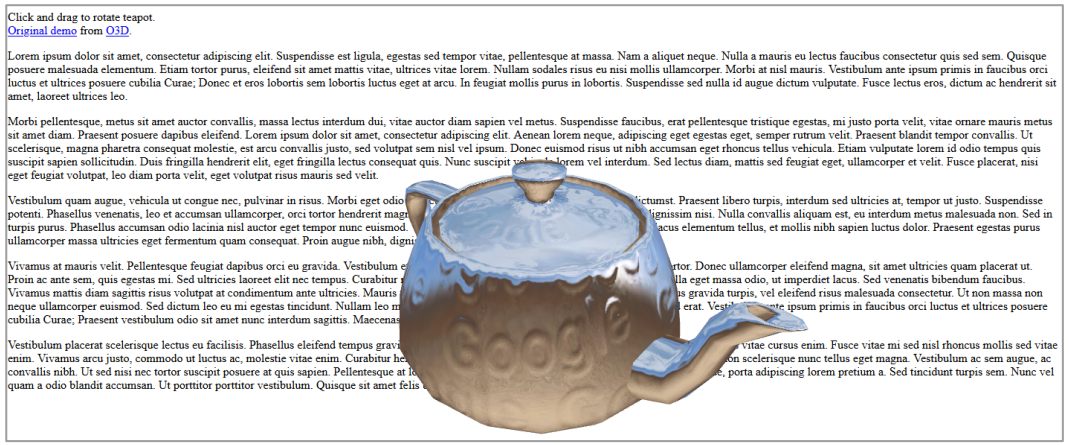
- Images and color
- WebGL
- Linear algebra and transformation
- Viewer transformations and rasterization
- Graphics pipeline
- Light and shading
- Texture mapping
- Shadows
- Ray tracing
- Antialiasing
- Curves and surfaces
- Spatial data structures

Images and color



From: Interactive Computer Graphics 7th Ed by Professor Ed Angel and Dave Shreiner

WebGL & web programming


Click and drag to rotate teapot.
Original demo from O3D.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse est ligula, egestas sed tempor vitae, pellentesque at massa. Nam a aliquet neque. Nulla a mauris eu lectus faucibus consectetur quis sed sem. Quisque posuere malesuada elementum. Etiam tortor purus, eleifend sit amet mattis vitae, ultrices vitae lorem. Nullam sodales risus eu nisi mollis ullamcorper. Morbi at nisl mauris. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Donec et eros lobortis sem lobortis luctus eget at arcu. In feugiat mollis purus in lobortis. Suspendisse sed nulla id augue dictum vulputate. Fusce lectus eros, dictum ac hendrerit sit amet, laoreet ultrices leo.

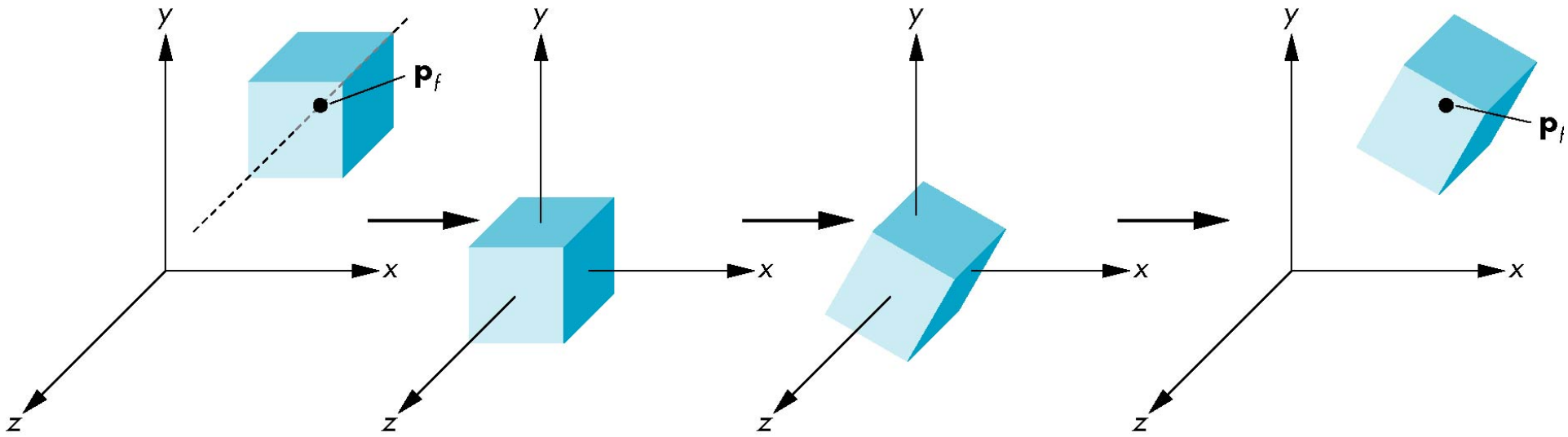
Morbi pellentesque, metus sit amet auctor convallis, massa lectus interdum dui, vitae auctor diam sapien vel metus. Suspendisse faucibus, erat pellentesque tristique egestas, mi justo porta velit, vitae ornare mauris metus sit amet diam. Praesent posuere dapibus eleifend. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean lorem neque, adipiscing eget egestas eget, semper rutrum velit. Praesent blandit tempor convallis. Ut scelerisque, magna pharetra consequat molestie, est arcu convallis justo, sed volutpat sem nisl vel ipsum. Donec euismod risus ut nibh accumsan eget rhoncus tellus vehicula. Etiam vulputate lorem id odio tempus quis suscipit sapien sollicitudin. Duis fringilla hendrerit elit, eget fringilla lectus consequat quis. Nunc suscipit
lorem vel interdum. Sed lectus diam, mattis sed feugiat eget, ullamcorper et velit. Fusce placerat, nisi eget feugiat volutpat, leo diam porta velit, eget volutpat risus mauris sed velit.

Vestibulum quam augue, vehicula ut congue nec, pulvinar in risus. Morbi eget odio
ictumst. Praesent libero turpis, interdum sed ultricies at, tempor ut justo. Suspendisse potenti. Phasellus venenatis, leo et accumsan ullamcorper, orci tortor hendrerit magn
lignissim nisi. Nulla convallis aliquam est, eu interdum metus malesuada non. Sed in turpis purus. Phasellus accumsan odio lacinia nisl auctor eget tempor nunc euismod.
acus elementum tellus, et mollis nibh sapien luctus dolor. Praesent egestas purus ullamcorper massa ultricies eget fermentum quam consequat. Proin augue nibh, digni

Vivamus at mauris velit. Pellentesque feugiat dapibus orci eu gravida. Vestibulum e
rtor. Donec ullamcorper eleifend magna, sit amet ultricies quam placerat ut. Proin ac ante sem, quis egestas mi. Sed ultricies laoreet elit nec tempus. Curabitur
lla eget massa odio, ut imperdiet lacus. Sed venenatis bibendum faucibus. Vivamus mattis diam sagittis risus volutpat at condimentum ante ultricies. Mauris
s gravida turpis, vel eleifend risus malesuada consectetur. Ut non massa non neque ullamcorper euismod. Sed dictum leo eu mi egestas tincidunt. Nullam leo m
erat. Vest
ipsum primis in faucibus orci luctus et ultricies posuere cubilia Curae; Praesent vestibulum odio sit amet nunc interdum sagittis. Maecenas

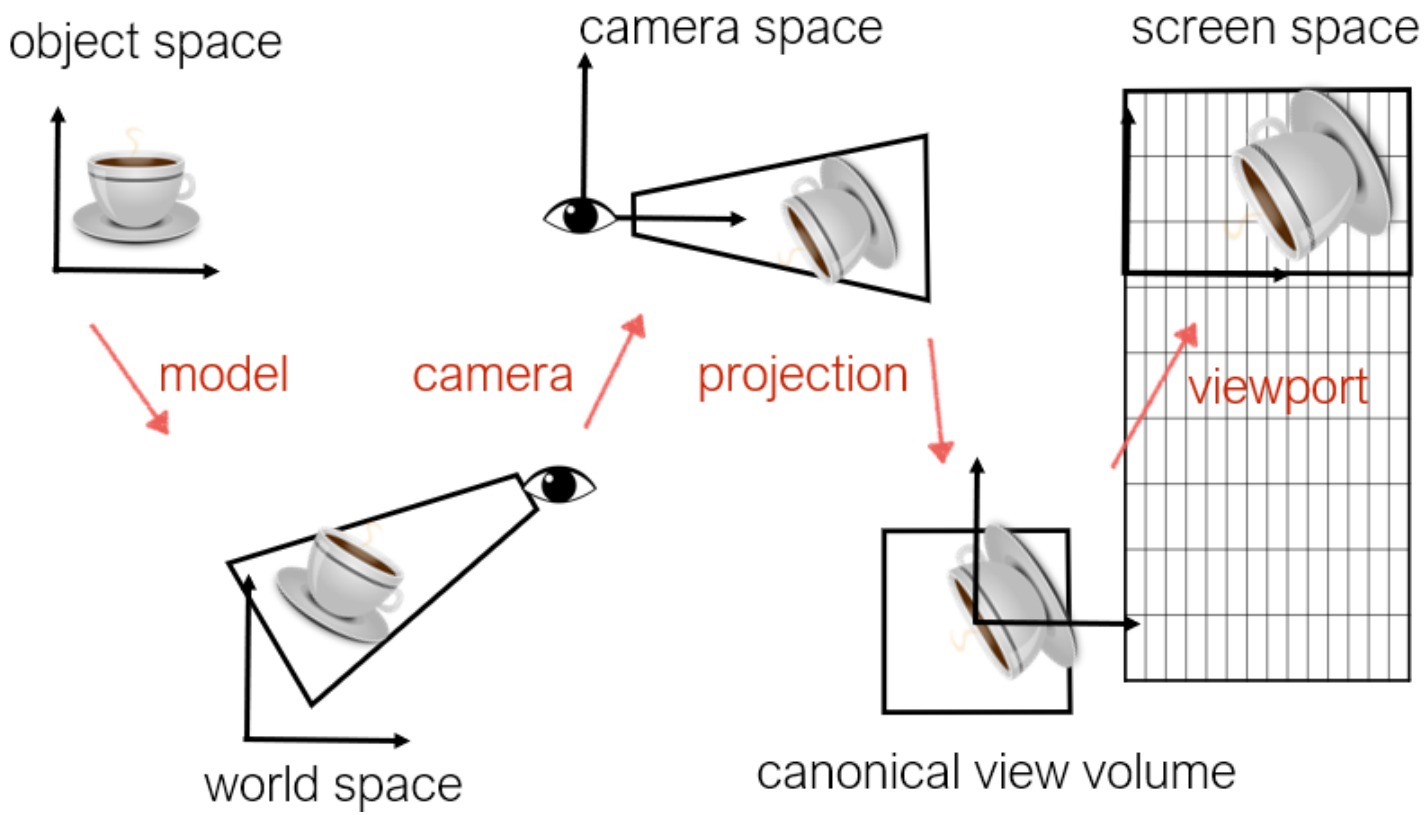
Vestibulum placerat scelerisque lectus eu facilisis. Phasellus eleifend tempus gravi
vitae cursus enim. Fusce vitae mi sed nisl rhoncus mollis sed vitae enim. Vivamus arcu justo, commodo ut luctus ac, molestie vitae enim. Curabitur he
on scelerisque nunc tellus eget magna. Vestibulum ac sem augue, ac convallis nibh. Ut sed nisi nec tortor suscipit posuere at quis sapien. Pellentesque at l
, porta adipiscing lorem pretium a. Sed tincidunt turpis sem. Nunc vel quam a odio blandit accumsan. Ut porttitor porttitor vestibulum. Quisque sit amet felis

Linear algebra and transformations



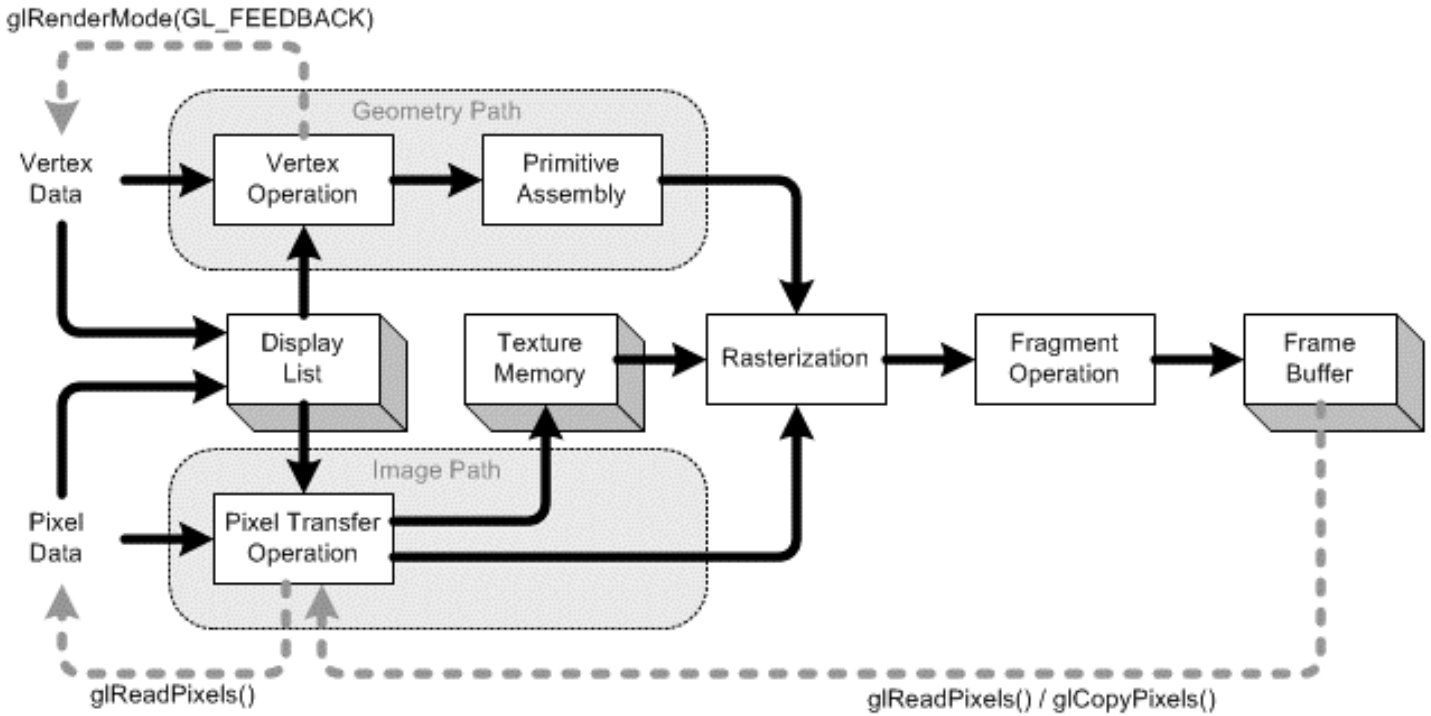
From: Computer Graphics by Professor Daniele Panozzo - NYU

Viewer transformations and rasterization



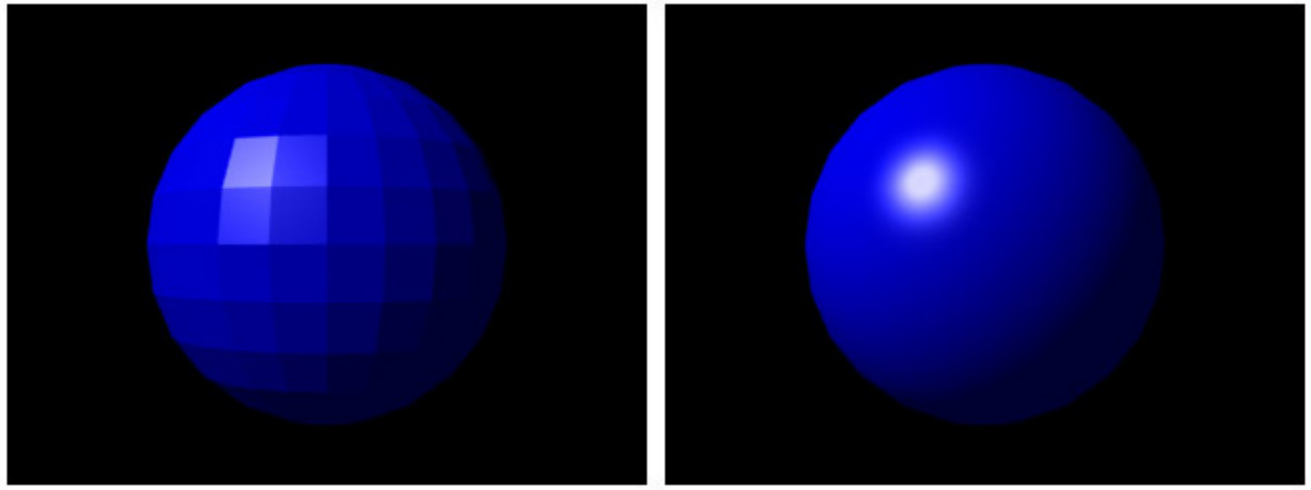
From: Computer Graphics by Professor Daniele Panozzo - NYU

Graphics pipeline

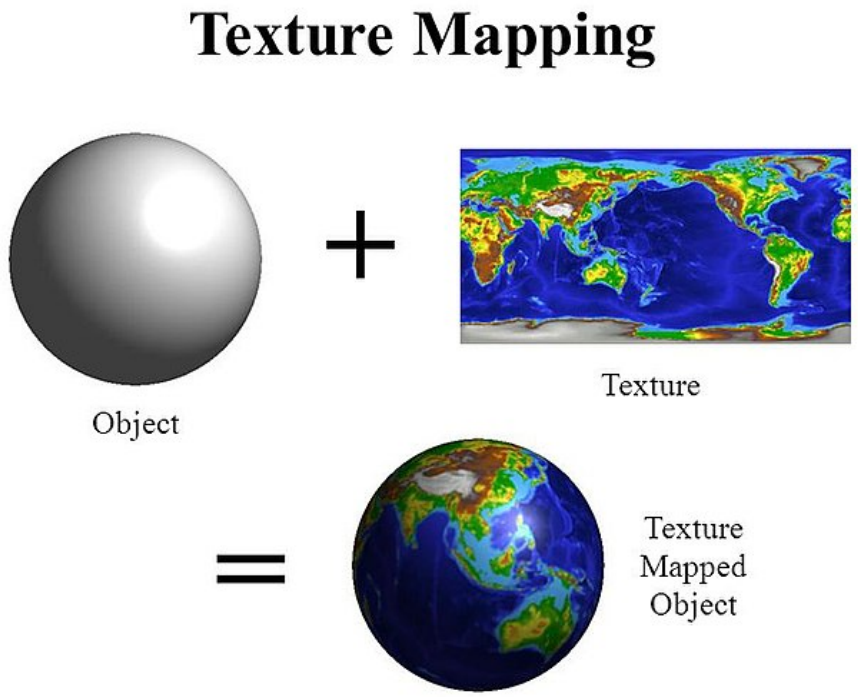


From: songho.ca

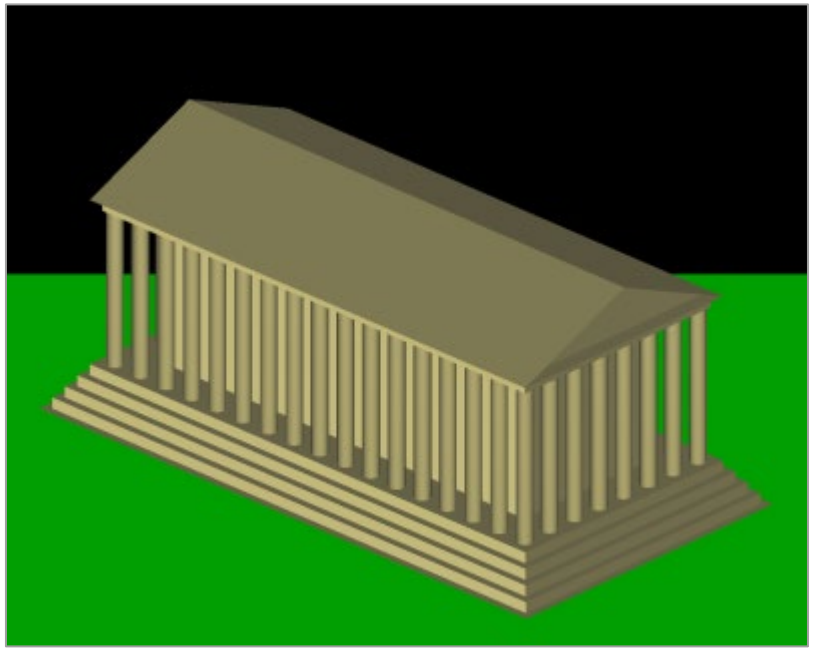
Light and shading

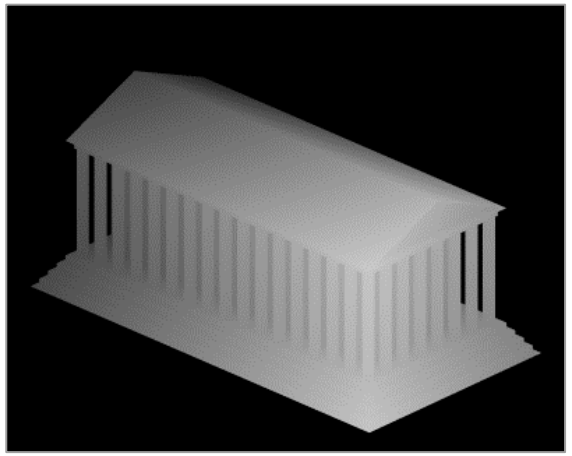


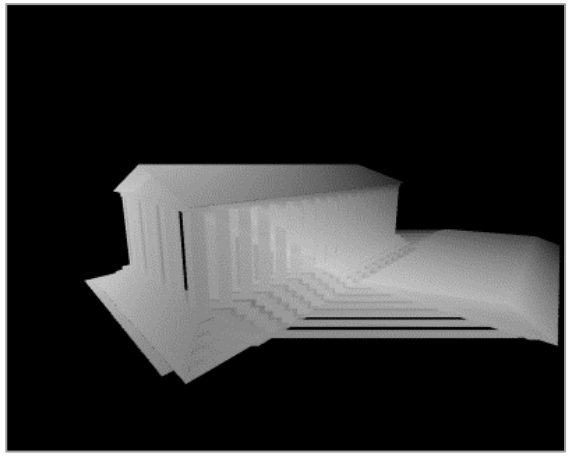
Texture mapping

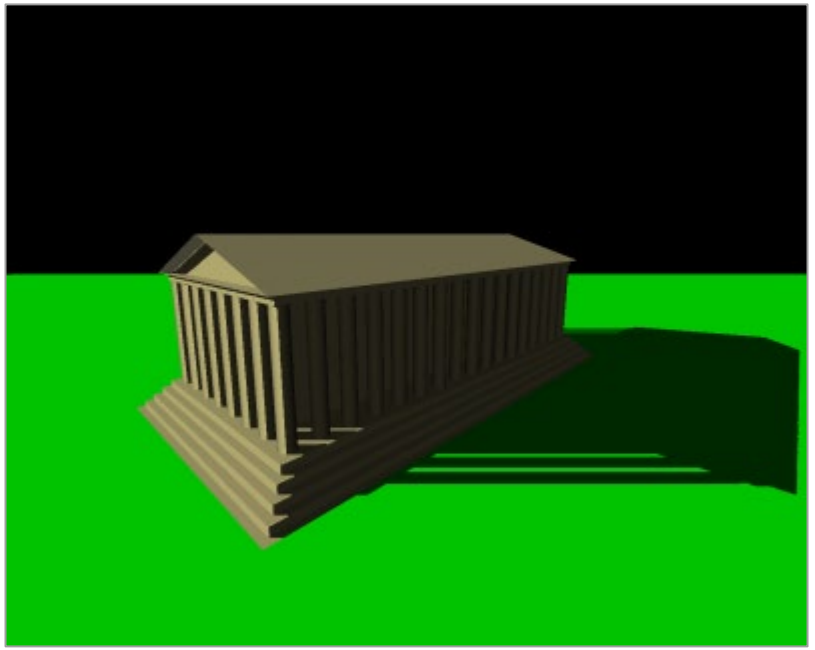


Shadows

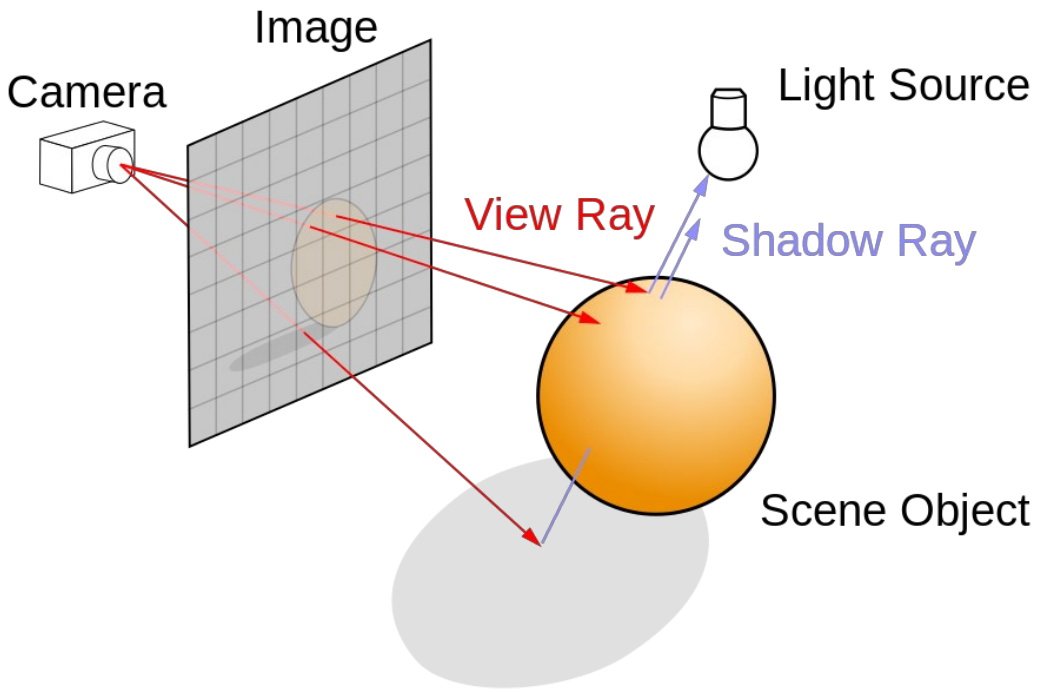






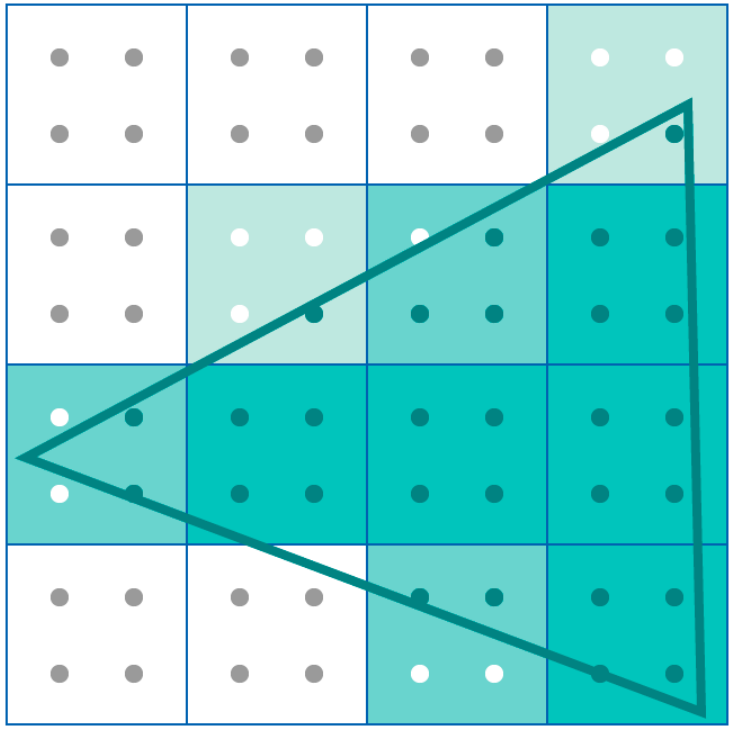


Ray tracing



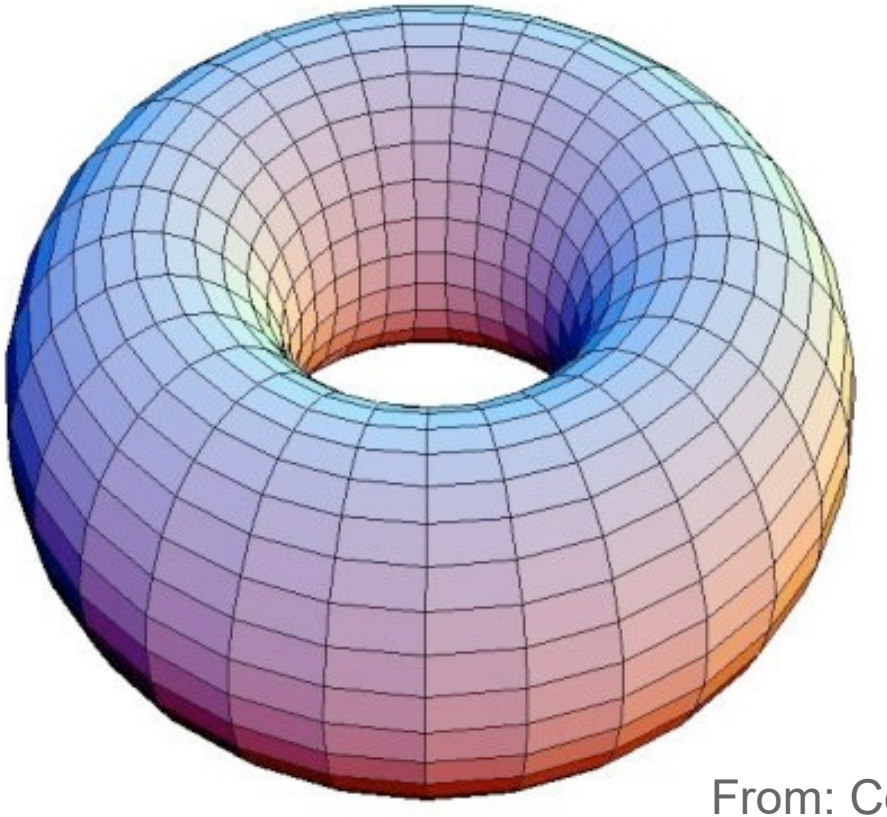
From: Computer Graphics by Professor Daniele Panozzo - NYU

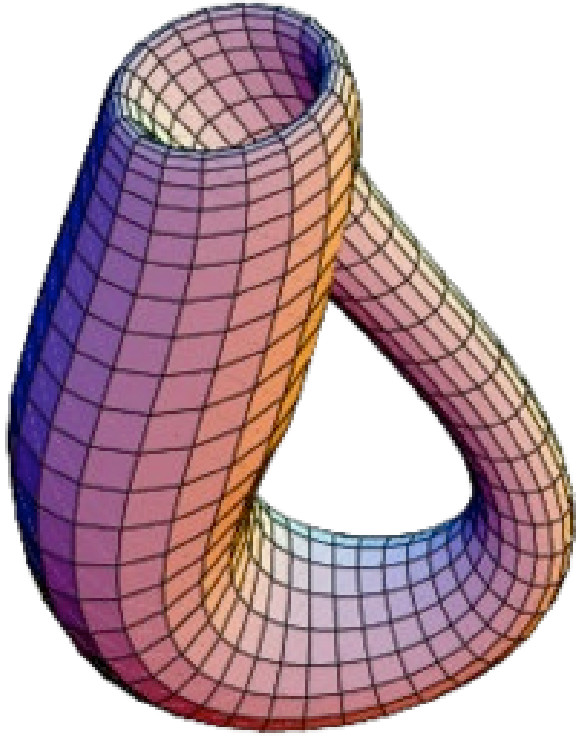
Antialiasing



Apple MSAA

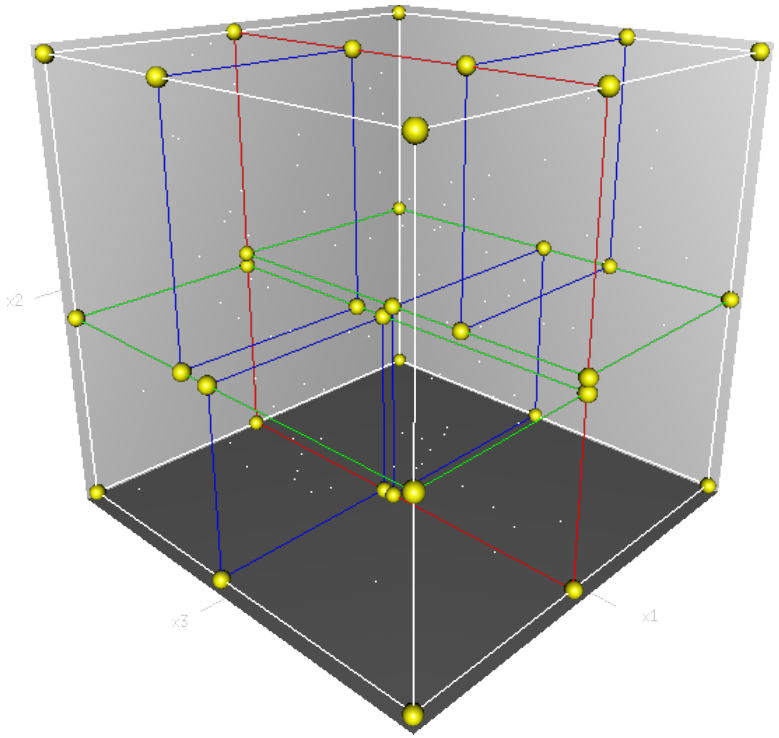
Curves and surfaces

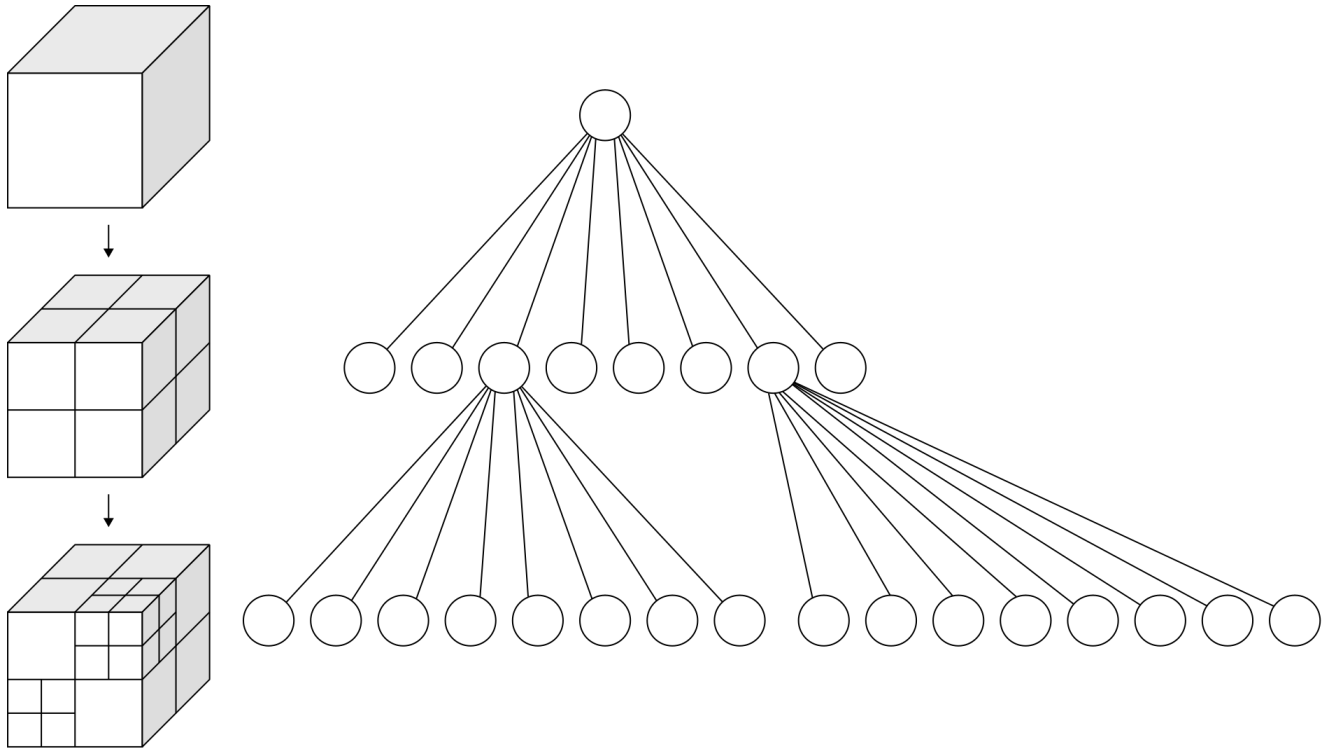




From: Computer Graphics by Professor Daniele Panozzo - NYU

Spatial data structures

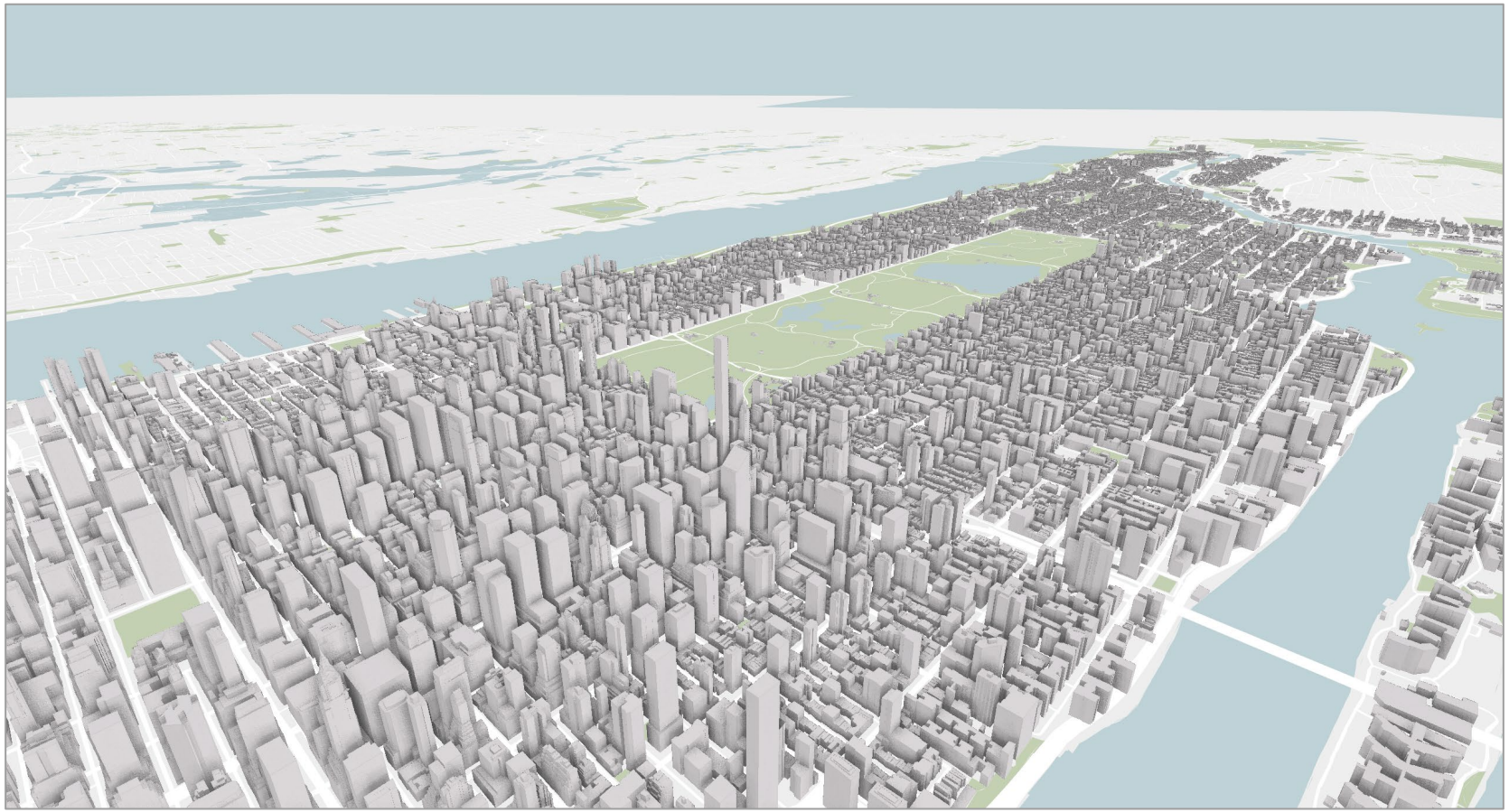




Assignments

- Assignment 0: WebGL + Web environment
- Assignment 1: Triangle meshes rendering
- Assignment 2: Shading, texture and shadows
- Assignment 3: Ray tracing
- Assignment 4: Procedural meshes

Assignment 4 goal



Assignment considerations

- Create a GitHub project named **cs425-spring-2021**
- Inside this project create a folder named **assignment[X]** for each assignment.
- For each assignment you will hand in the following files:
  - index.html: main html file.
  - assignment[x].js: assignment main source code.
  - readme.md: markdown readme file with a description of the program.

Grading

- Assignment 0: 10%
- Assignments 1-4: 70% (17.5% each)
- Take-home exam: 20%

Recommended workflow

Day 0: Read the assignment

Day 1-2: Familiarize yourself with the main topics of the assignment

Day 3-10: Code the different components of the assignment

Day 8-10: Start writing README.md file, make sure code runs on different computers

Day 11-12: Fix problems, double check README.md

Day 13: Submit

Policies

- You are encouraged to discuss the assignments with your classmates, but collaboration in the assignment is not allowed.
- You are not allowed to copy code from online sources or use external libraries for any of the assignments (unless clearly specified in the assignment). Do not share implementation details or anything solution-oriented.

Useful links

https://open.gl

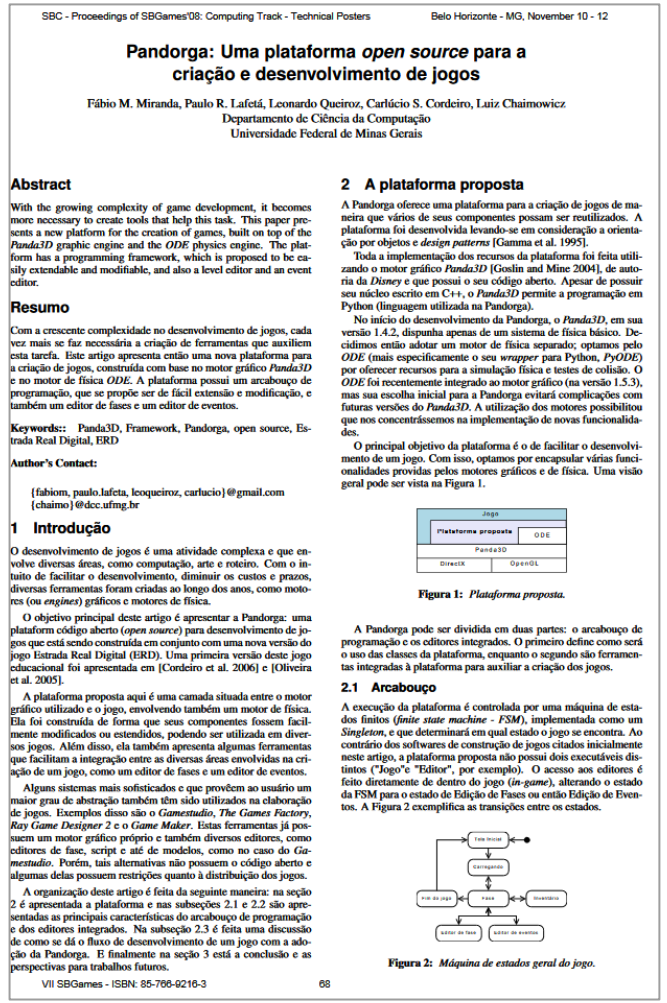
https://webgl2fundamentals.org/

https://songho.ca/opengl/

https://learnopengl.com/

Final note

https://youtu.be/wjnELNMC5kA



SBC - Proceedings of SBGames'08: Computing Track - Technical Posters — Belo Horizonte - MG, November 10 - 12

Pandorga: Uma plataforma *open source* para a criação e desenvolvimento de jogos

Fábio M. Miranda, Paulo R. Lafetá, Leonardo Queiroz, Carlúcio S. Cordeiro, Luiz Chaimowicz
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais

Abstract

With the growing complexity of game development, it becomes more necessary to create tools that help this task. This paper presents a new platform for the creation of games, built on top of the *Panda3D* graphic engine and the *ODE* physics engine. The platform has a programming framework, which is proposed to be easily extendable and modifiable, and also a level editor and an event editor.

Resumo

Com a crescente complexidade no desenvolvimento de jogos, cada vez mais se faz necessária a criação de ferramentas que auxiliem esta tarefa. Este artigo apresenta então uma nova plataforma para a criação de jogos, construída com base no motor gráfico *Panda3D* e no motor de física *ODE*. A plataforma possui um arcabouço de programação, que se propõe ser de fácil extensão e modificação, e também um editor de fases e um editor de eventos.

Keywords:: Panda3D, Framework, Pandorga, open source, Estrada Real Digital, ERD

Author's Contact:

{fabiom, paulo.lafeta, leoqueiroz, carlucio}@gmail.com
{chaimo}@dcc.ufmg.br

1 Introdução

O desenvolvimento de jogos é uma atividade complexa e que envolve diversas áreas, como computação, arte e roteiro. Com o intuito de facilitar o desenvolvimento, diminuir os custos e prazos, diversas ferramentas foram criadas ao longo dos anos, como motores (ou *engines*) gráficos e motores de física.

O objetivo principal deste artigo é apresentar a Pandorga: uma plataform código aberto (*open source*) para desenvolvimento de jogos que está sendo construída em conjunto com uma nova versão do jogo Estrada Real Digital (ERD). Uma primeira versão deste jogo educacional foi apresentada em [Cordeiro et al. 2006] e [Oliveira et al. 2005].

A plataforma proposta aqui é uma camada situada entre o motor gráfico utilizado e o jogo, envolvendo também um motor de física. Ela foi construída de forma que seus componentes fossem facilmente modificados ou estendidos, podendo ser utilizada em diversos jogos. Além disso, ela também apresenta algumas ferramentas que facilitam a integração entre as diversas áreas envolvidas na criação de um jogo, como um editor de fases e um editor de eventos.

Alguns sistemas mais sofisticados e que provêem ao usuário um maior grau de abstração também têm sido utilizados na elaboração de jogos. Exemplos disso são o *Gamestudio*, *The Games Factory*, *Ray Game Designer 2* e o *Game Maker*. Estas ferramentas já possuem um motor gráfico próprio e também diversos editores, como editores de fase, script e até de modelos, como no caso do *Gamestudio*. Porém, tais alternativas não possuem o código aberto e algumas delas possuem restrições quanto à distribuição dos jogos.

A organização deste artigo é feita da seguinte maneira: na seção 2 é apresentada a plataforma e nas subseções 2.1 e 2.2 são apresentadas as principais características do arcabouço de programação e dos editores integrados. Na subseção 2.3 é feita uma discussão de como se dá o fluxo de desenvolvimento de um jogo com a adoção da Pandorga. E finalmente na seção 3 está a conclusão e as perspectivas para trabalhos futuros.

2 A plataforma proposta

A Pandorga oferece uma plataforma para a criação de jogos de maneira que vários de seus componentes possam ser reutilizados. A plataforma foi desenvolvida levando-se em consideração a orientação por objetos e *design patterns* [Gamma et al. 1995].

Toda a implementação dos recursos da plataforma foi feita utilizando o motor gráfico *Panda3D* [Goslin and Mine 2004], de autoria da *Disney* e que possui o seu código aberto. Apesar de possuir seu núcleo escrito em C++, o *Panda3D* permite a programação em Python (linguagem utilizada na Pandorga).

No início do desenvolvimento da Pandorga, o *Panda3D*, em sua versão 1.4.2, dispunha apenas de um sistema de física básico. Decidimos então adotar um motor de física separado; optamos pelo *ODE* (mais especificamente o seu *wrapper* para Python, *PyODE*) por oferecer recursos para a simulação física e testes de colisão. O *ODE* foi recentemente integrado ao motor gráfico (na versão 1.5.3), mas sua escolha inicial para a Pandorga evitará complicações com futuras versões do *Panda3D*. A utilização dos motores possibilitou que nos concentrássemos na implementação de novas funcionalidades.

O principal objetivo da plataforma é o de facilitar o desenvolvimento de um jogo. Com isso, optamos por encapsular várias funcionalidades providas pelos motores gráficos e de física. Uma visão geral pode ser vista na Figura 1.

Figura 1: *Plataforma proposta.*

A Pandorga pode ser dividida em duas partes: o arcabouço de programação e os editores integrados. O primeiro define como será o uso das classes da plataforma, enquanto o segundo são ferramentas integradas à plataforma para auxiliar a criação dos jogos.

2.1 Arcabouço

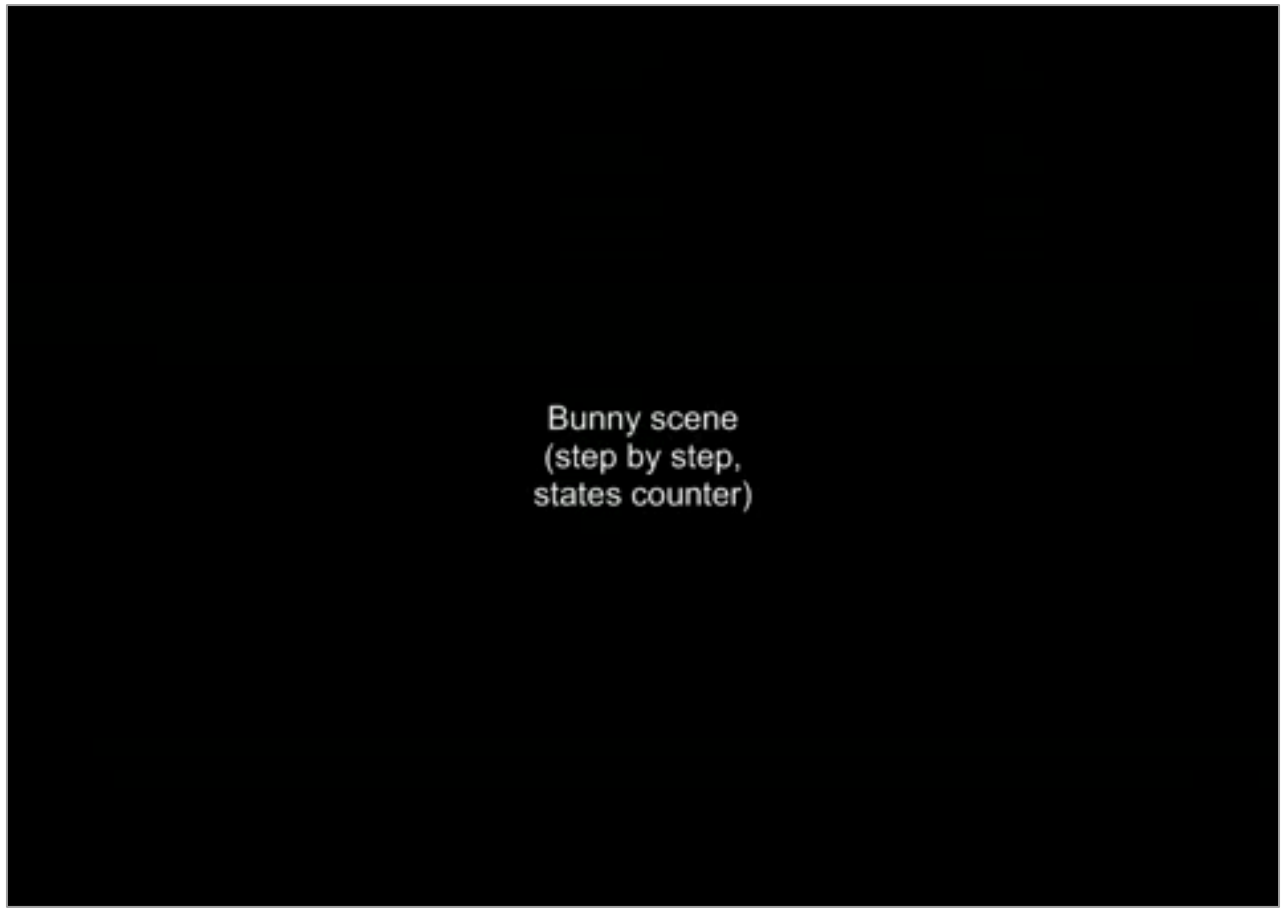
A execução da plataforma é controlada por uma máquina de estados finitos (*finite state machine - FSM*), implementada como um *Singleton*, e que determinará em qual estado o jogo se encontra. Ao contrário dos softwares de construção de jogos citados inicialmente neste artigo, a plataforma proposta não possui dois executáveis distintos ("Jogo"e "Editor", por exemplo). O acesso aos editores é feito diretamente de dentro do jogo (*in-game*), alterando o estado da FSM para o estado de Edição de Fases ou então Edição de Eventos. A Figura 2 exemplifica as transições entre os estados.

Figura 2: *Máquina de estados geral do jogo.*

Final note

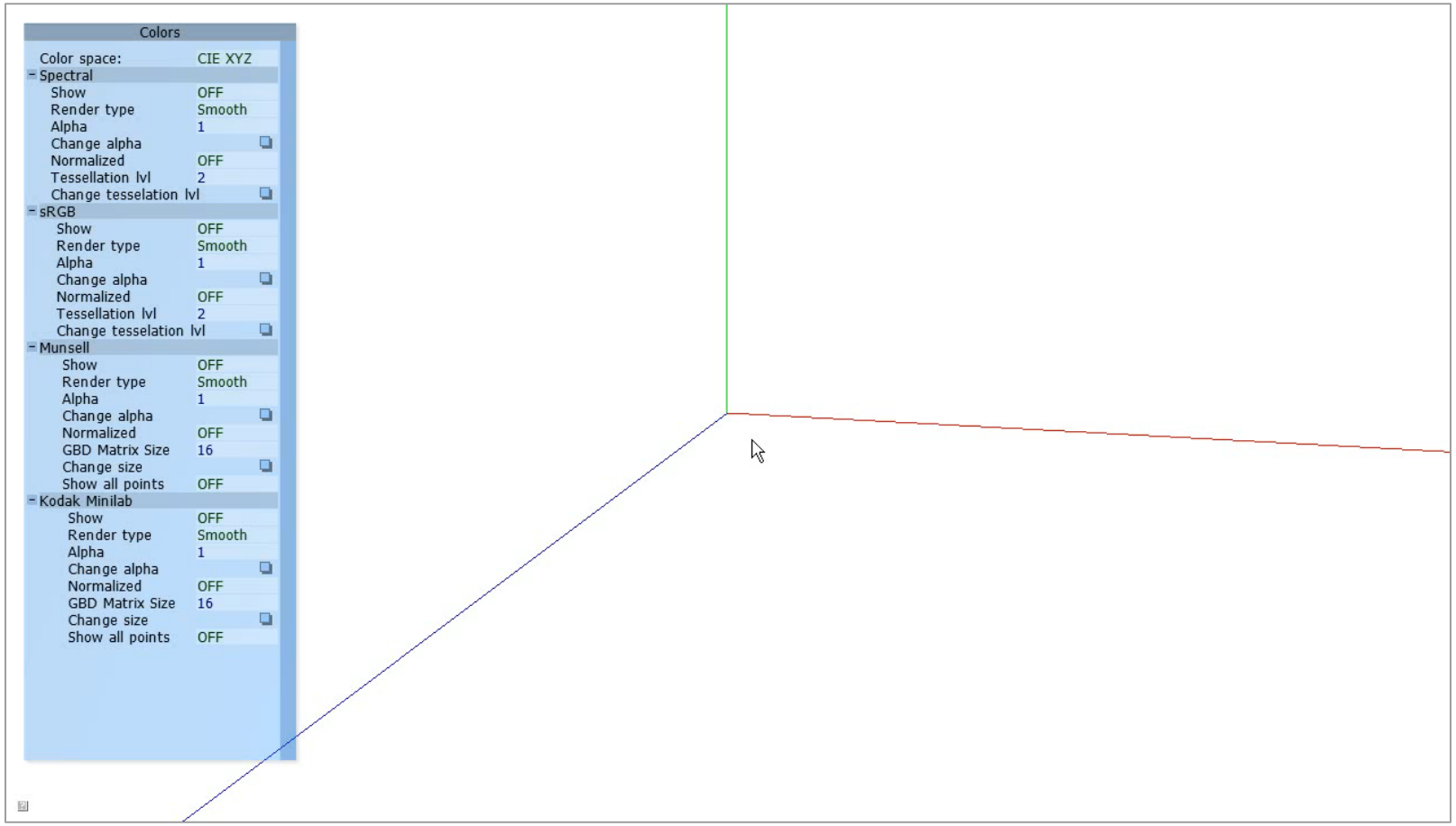
https://youtu.be/fRQ-NIQqOJg

Final note



https://youtu.be/FLP4_CGs2Ng

Final note



https://youtu.be/cvGCO9u_Ios

“Without the fun, none of us would go on”

Technology and Courage

Ivan Sutherland