

# Introduction and Overview

## CS425: Computer Graphics I

Fabio Miranda

<https://fmiranda.me>

Including adapted slides from:

Computer Graphics by Professor Daniele Panozzo - NYU

Interactive Computer Graphics by Professor Claudio Silva and Jonathas Costa – NYU

Interactive Computer Graphics 7<sup>th</sup> Ed by Professor Ed Angel and Dave Shreiner

# 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



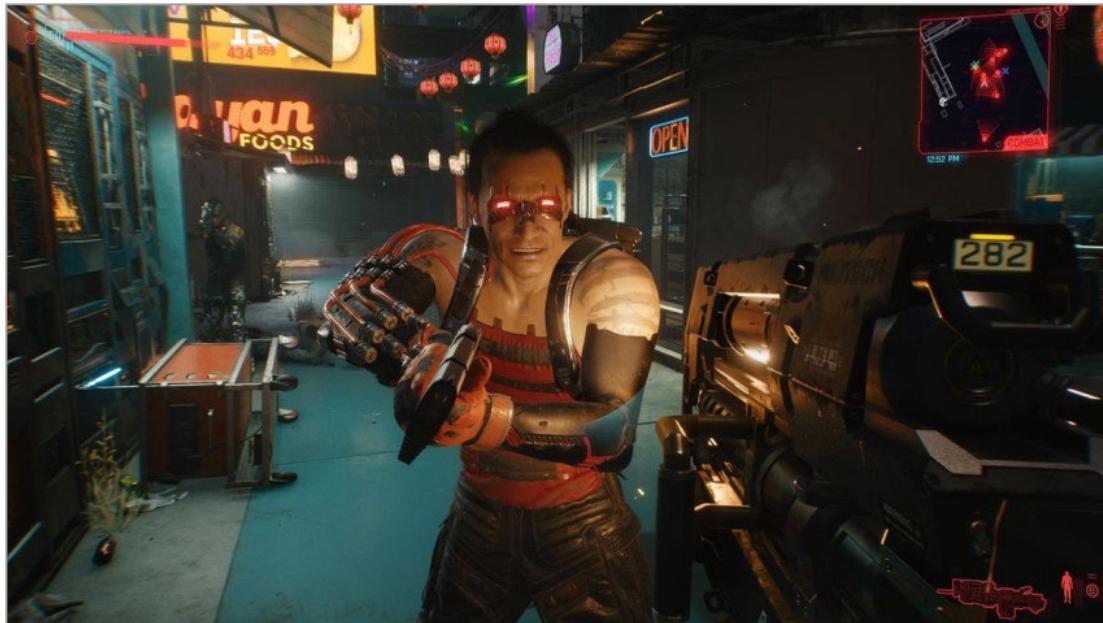
Google Earth



# Example

---

How was this image created?



Graphics system

Urbane,  
OpenSpace,  
ArcGIS,  
Google Earth, ...

Cyberpunk, Doom, FIFA, ...

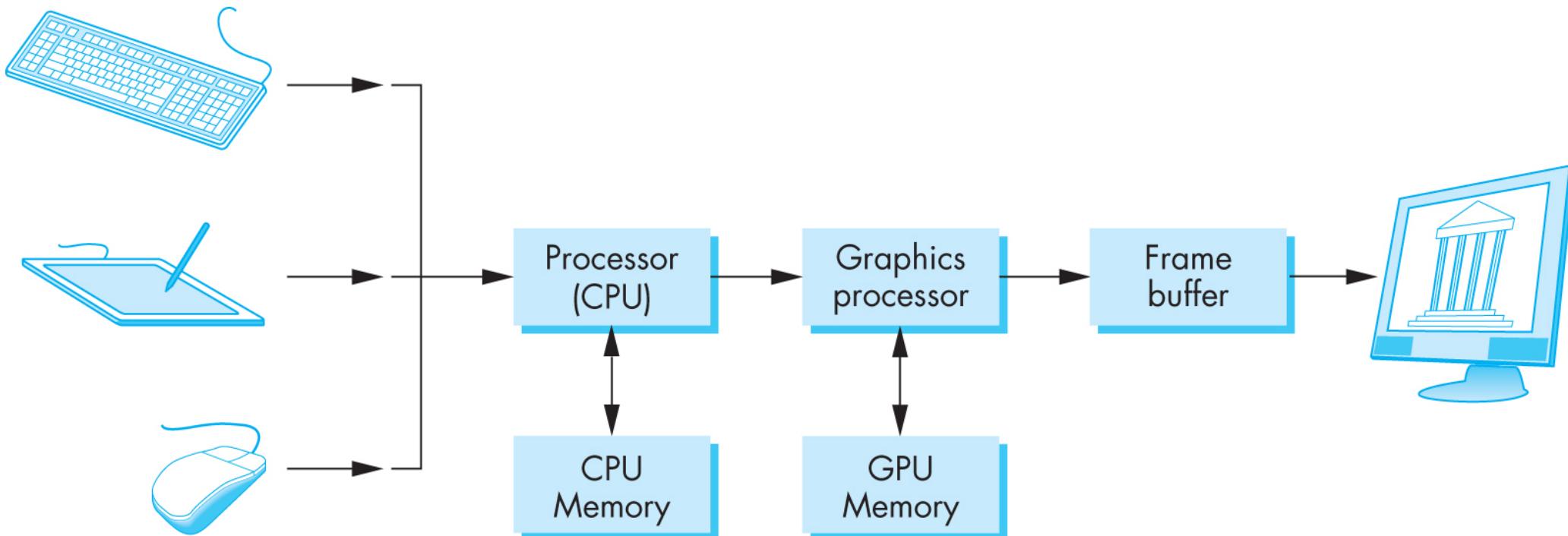
Unreal, Unity, idTech,  
Frostbite, REDengine, ...

OpenGL, DirectX, Vulkan, Metal, Glide, ...

Windows, Linux, iOS, ...

CPU, RAM, GPU, VRAM, ...

# Basic graphics system

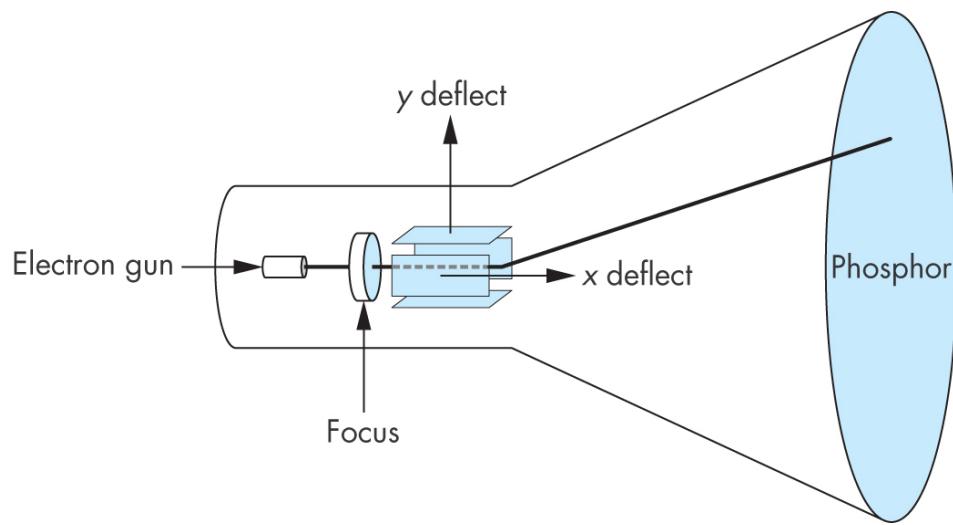


# Basic graphics system



# 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

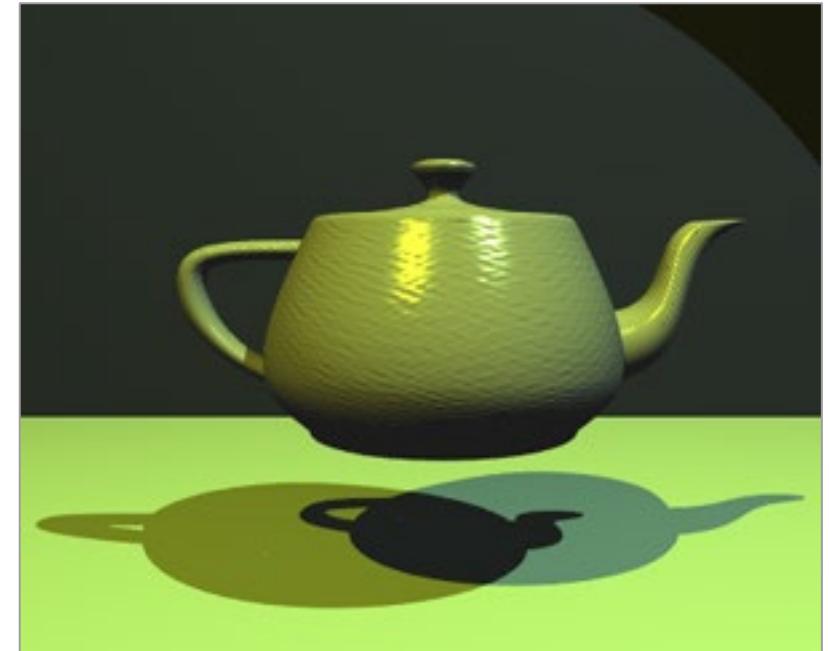
By Ivan Sutherland

1963

# Computer Graphics: 1970s

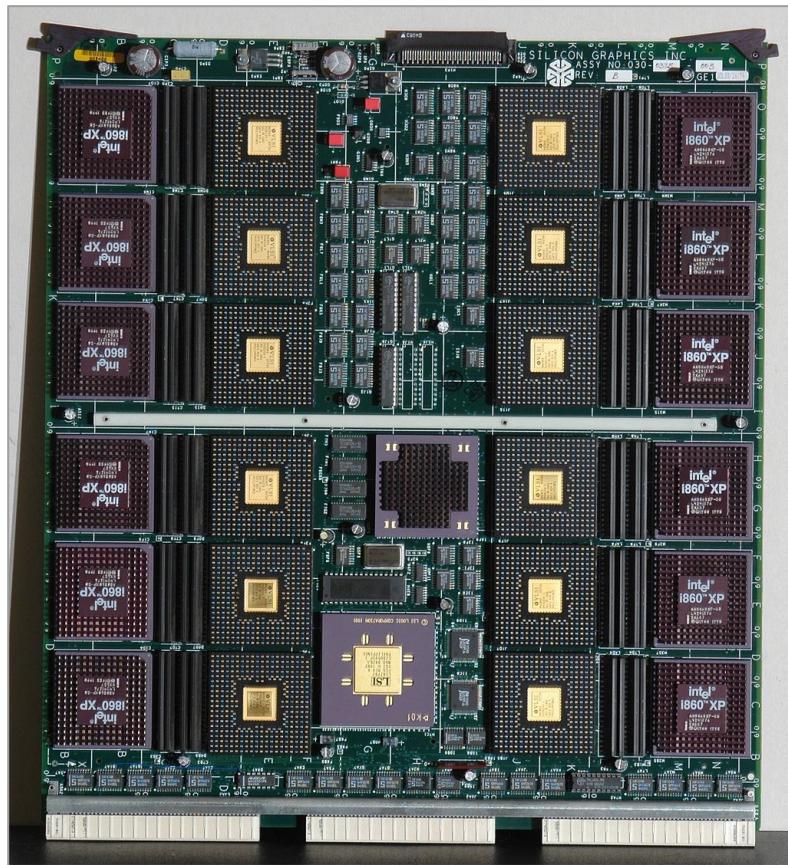
---

- 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

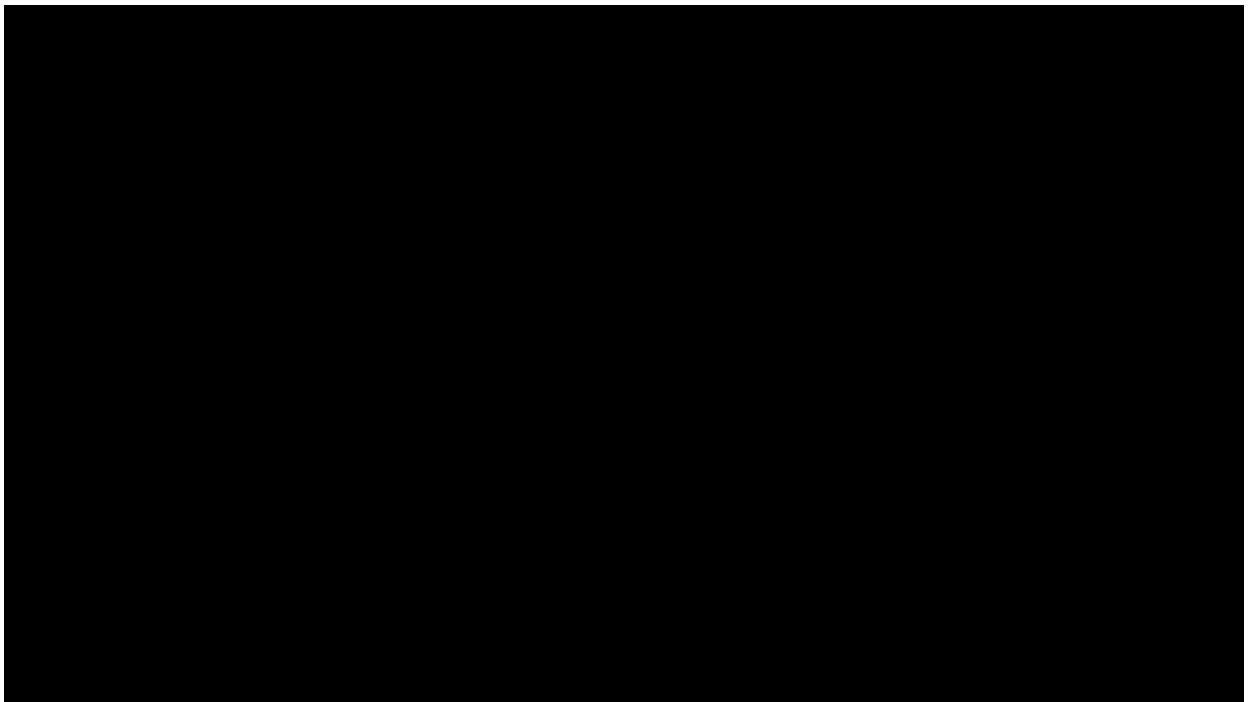
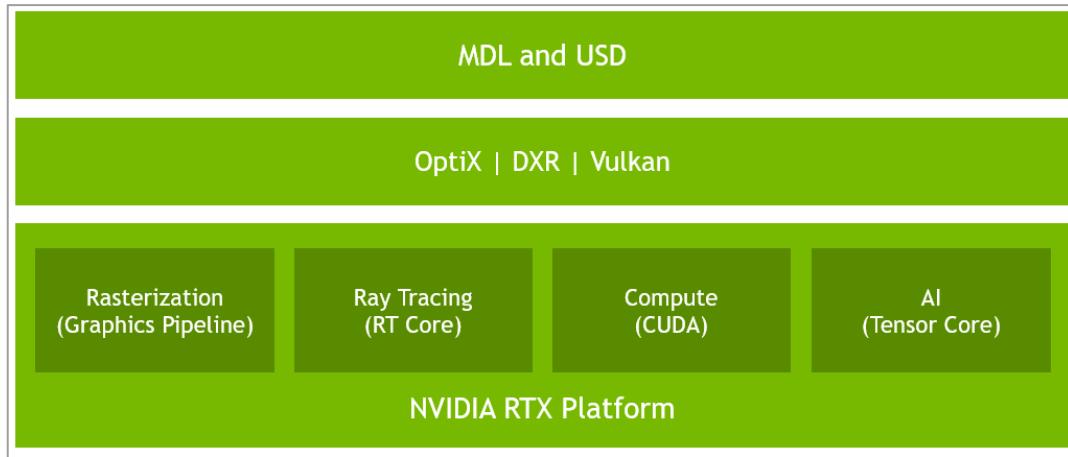
---

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



# Computer Graphics: 2020s

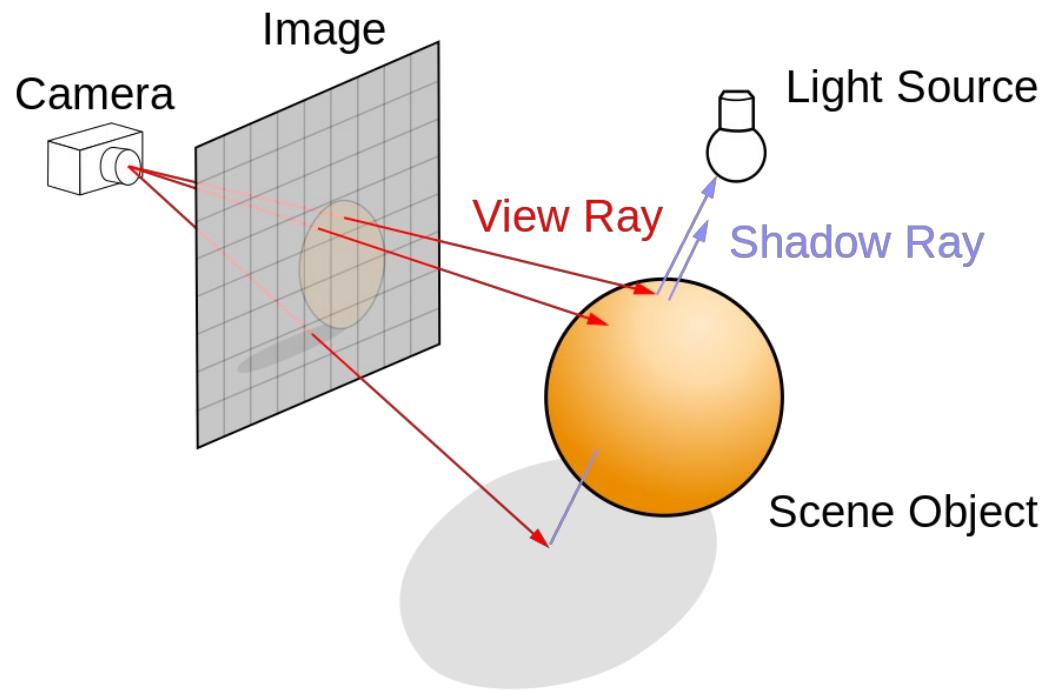
- Real-time ray tracing
- 4K resolution
- AI-based anti-aliasing



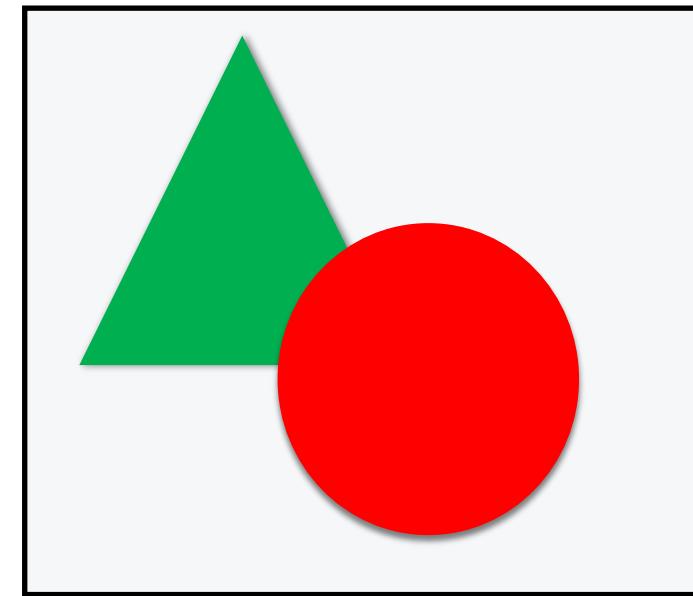
Nvidia - Reflections RTX Tech Demo

# Creating images

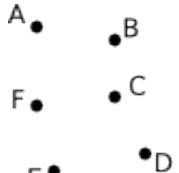
Per pixel: ray tracing



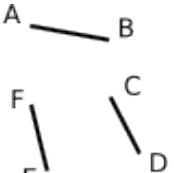
Per object: rasterization



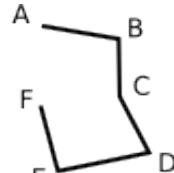
# Creating images: Objects description



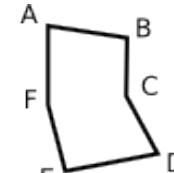
GL\_POINTS



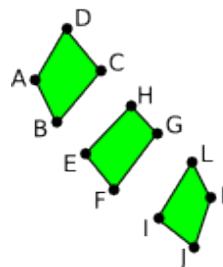
GL\_LINES



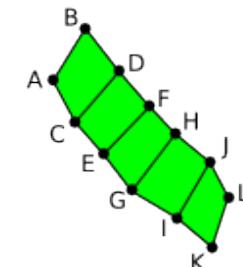
GL\_LINE\_STRIP



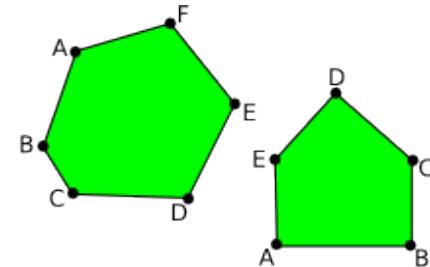
GL\_LINE\_LOOP



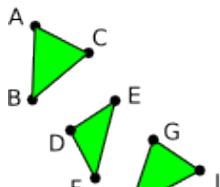
GL\_QUADS



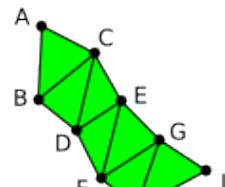
GL\_QUAD\_STRIP



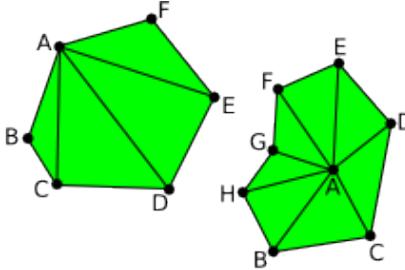
GL\_POLYGON



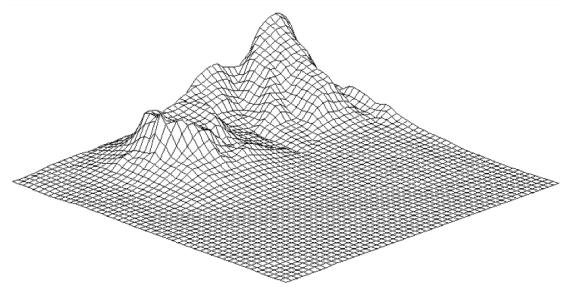
GL\_TRIANGLES



GL\_TRIANGLE\_STRIP

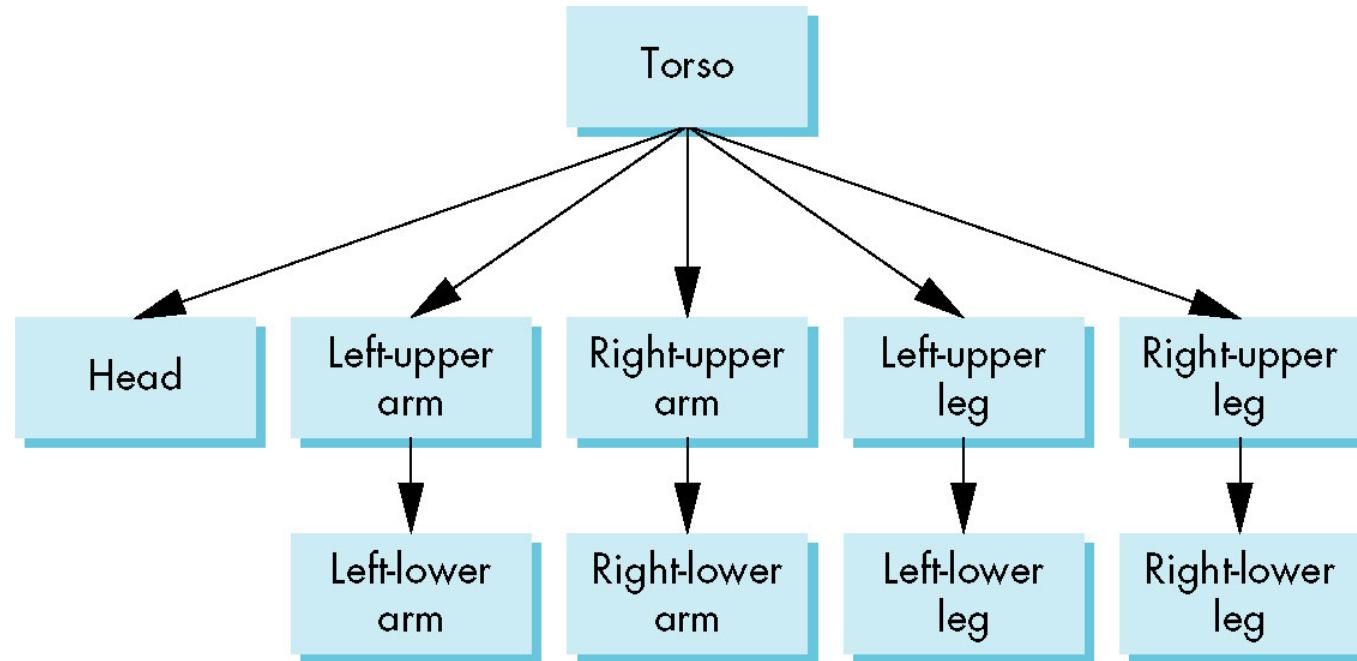
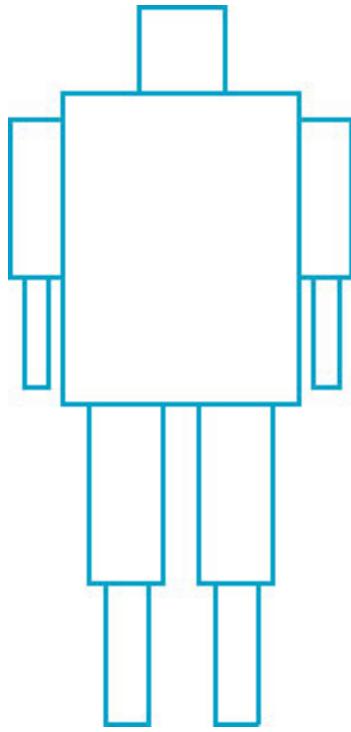


GL\_TRIANGLE\_FAN

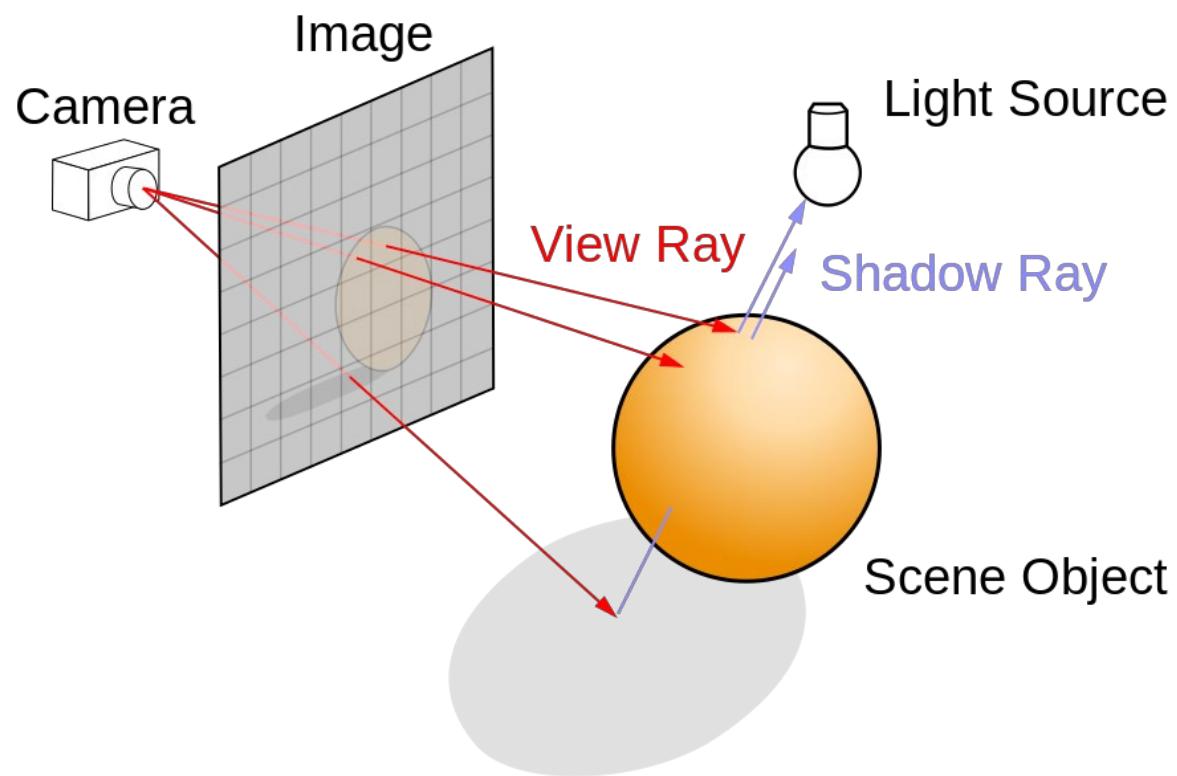


From: Introduction to Computer Graphics,  
David J. Eck

# Creating images: Scene description



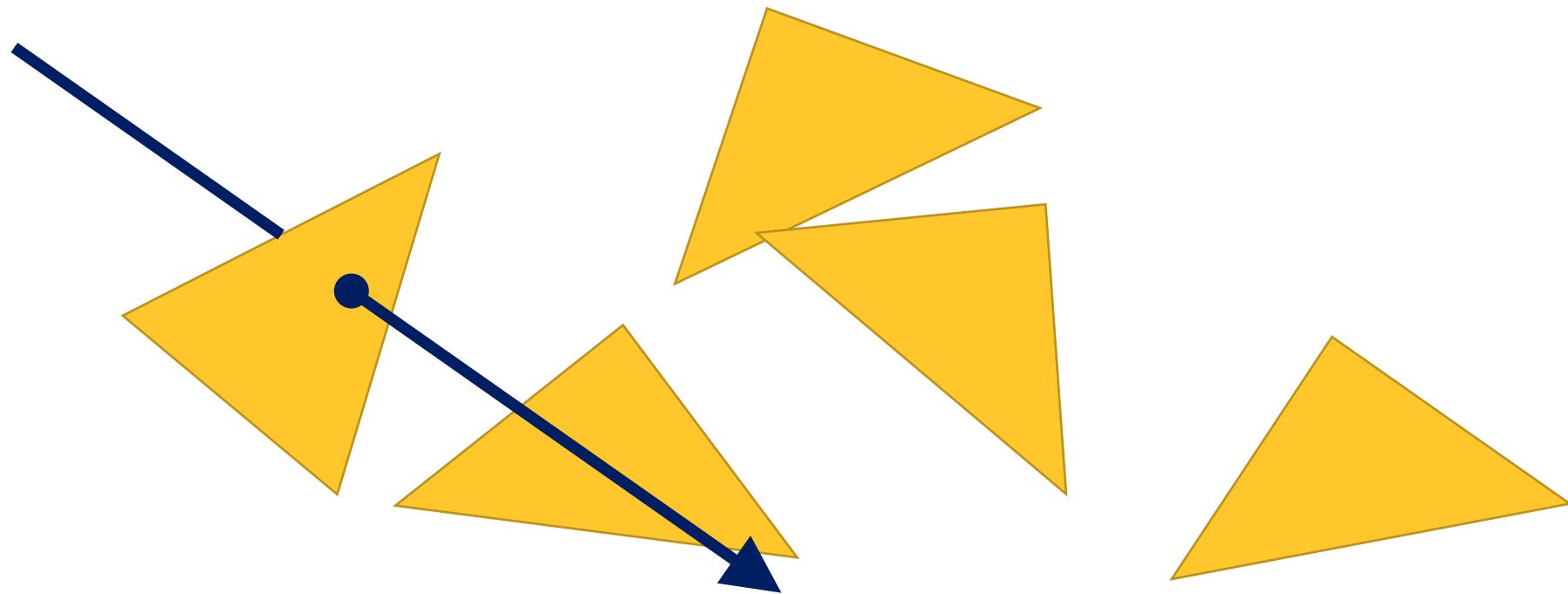
# Creating images: Ray tracing



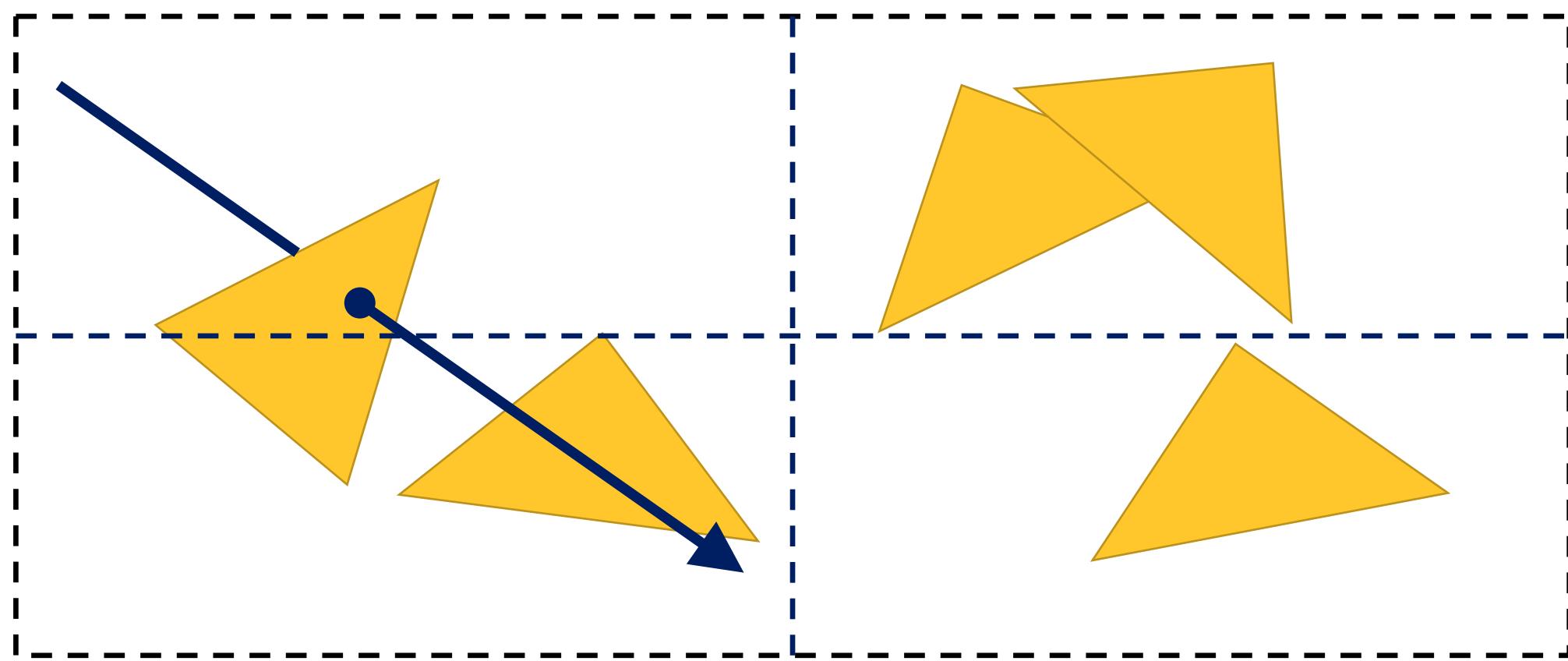
- Easy to parallelize but hard to map to hardware (up until recently)
- Expensive!
- It can be extended to model many physical phenomena (internal scattering, diffraction, reflections, etc)
- Used to obtain high-quality images

# Creating images: Ray tracing

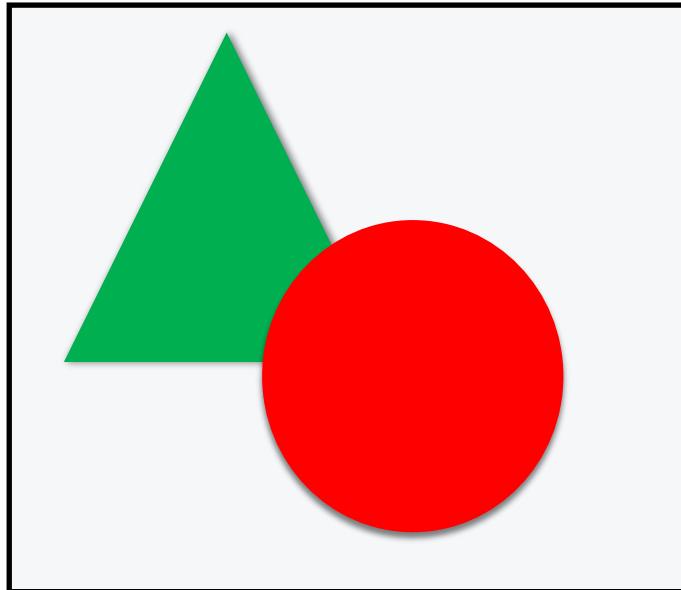
---



# Creating images: Ray tracing

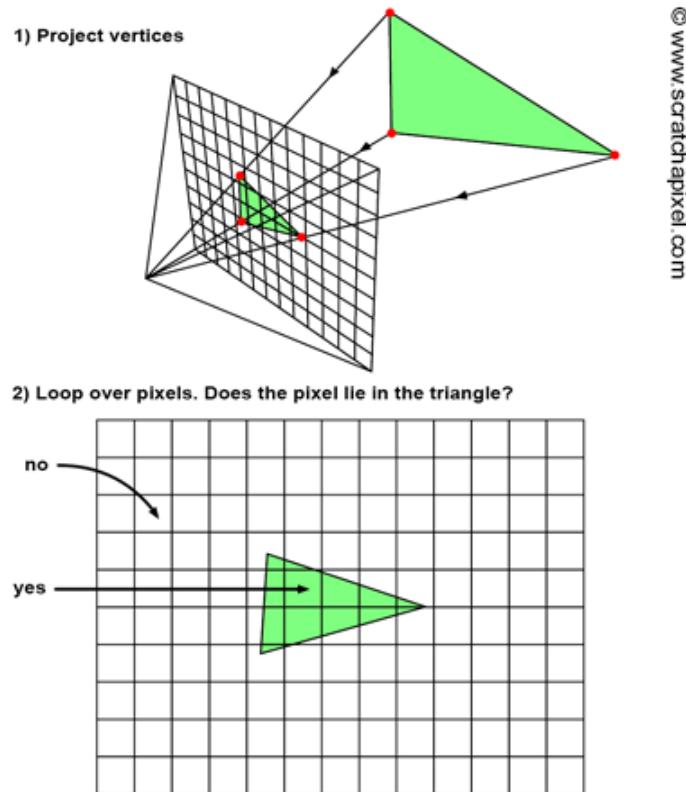


# Creating images: Rasterization



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

# Creating images: rasterization



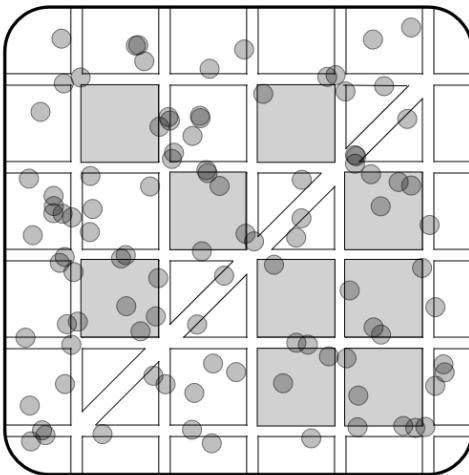
1. Project 3D vertices onto the screen.
2. Loop through pixels in the image and test if they lie within the resulting 2D triangles.

# Real-time rendering vs off-line rendering

---



# Computer Graphics & Visualization



Open Street Maps

**NYC 3-D Building Model**

Building on the rich geospatial data already provided, NYC DoITT is releasing a three-dimensional (3-D) Building Massing Model of New York City. The 3-D model includes every NYC building present in the 2014 aerial survey, the capture from which the model was developed.

Using the Open Geospatial Consortium's **CityGML** specification as the basis, the NYC 3-D Building Massing Model was developed to a hybrid specification combining elements from Level of Detail (LOD) 1 and 2. Highlights of the model include the different types of buildings, materials, including facades and ground plane. All major roof structures are modeled; however domes and rounded roofs are not. All roof appendages are excluded including but not limited to chimneys, parapets, spindles and antenna. In addition, a subset of 'iconic' NYC buildings was modeled to LOD 2. Lastly the buildings are attributed with a Building Identification Number (BIN) to provide a means to associate other NYC data for analysis and rendering. For further details, see the included metadata.

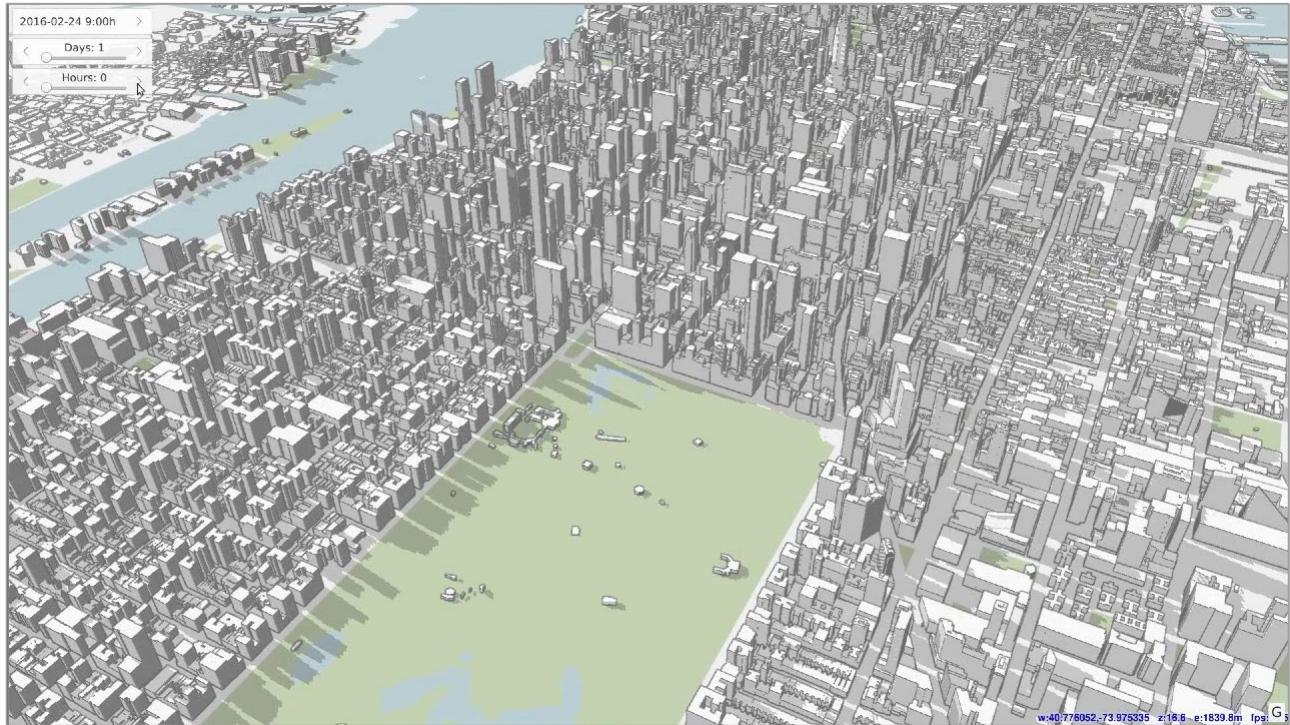
The 3-D model was a one-time capture. If the model proves useful, future updates and extension will be considered.

**Download**

Format	Size	URL
CityGML	894 MB	<a href="http://maps.nyc.gov/download/3dmodel/DA_WISE_GML.zip">http://maps.nyc.gov/download/3dmodel/DA_WISE_GML.zip</a>
Multipatch (ESRI)	251 MB	<a href="http://maps.nyc.gov/download/3dmodel/DA_WISE_Multipatch.zip">http://maps.nyc.gov/download/3dmodel/DA_WISE_Multipatch.zip</a>

DoITT

# Computer Graphics & Visualization



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



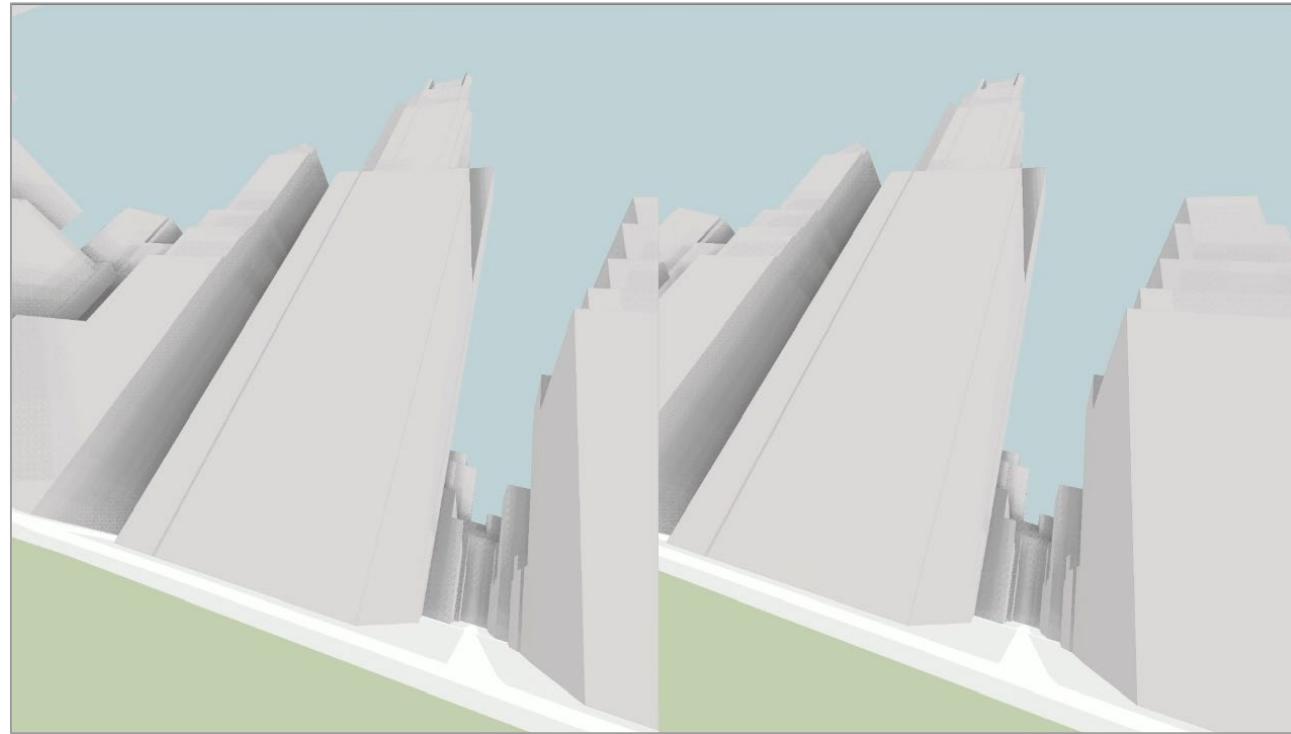
<https://nyti.ms/2k0bF0G>



COMPUTER SCIENCE

# Computer Graphics & Virtual Reality

---



Urbanrama



# Computer Graphics & Database



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

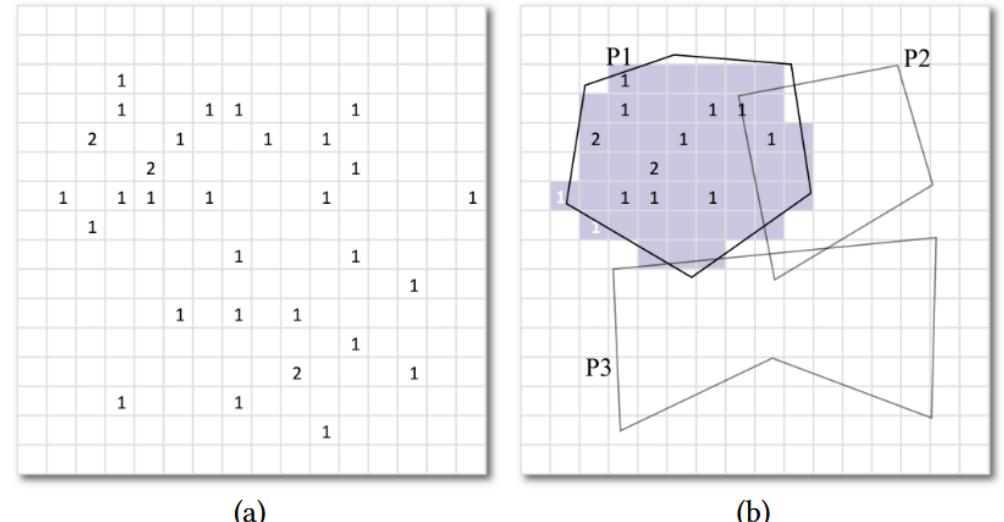
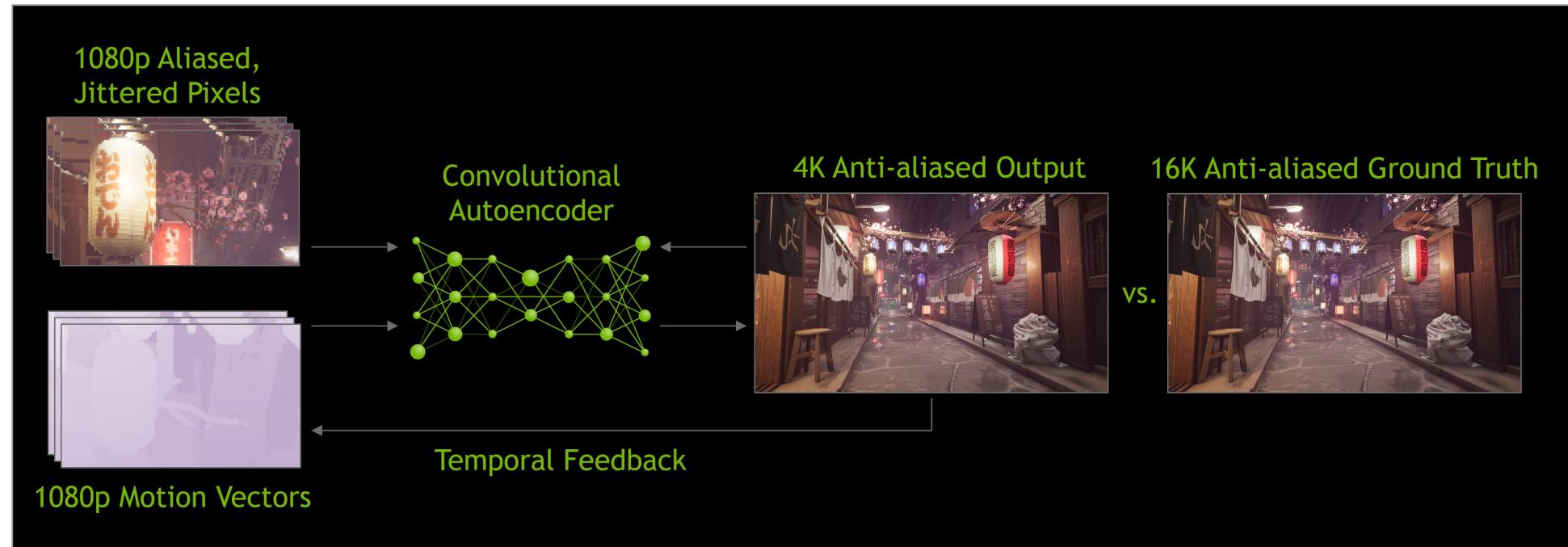


Figure 2: Raster Join first renders all points onto an FBO storing the count of points in each pixel (a). It then aggregates the pixel values corresponding to polygon fragments (b).

# Computer Graphics & Machine Learning



NVIDIA: Deep learning super sampling

# CS425: Course goals

---

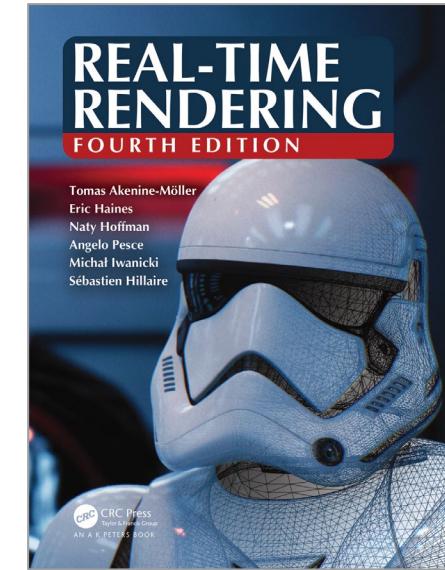
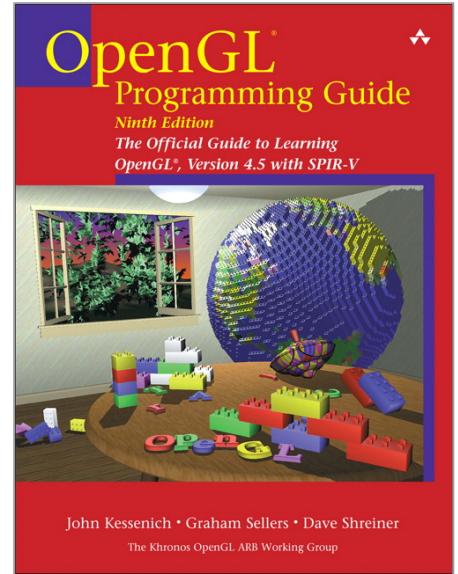
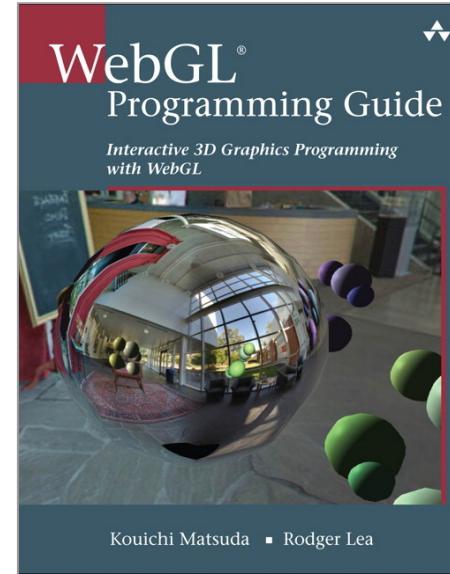
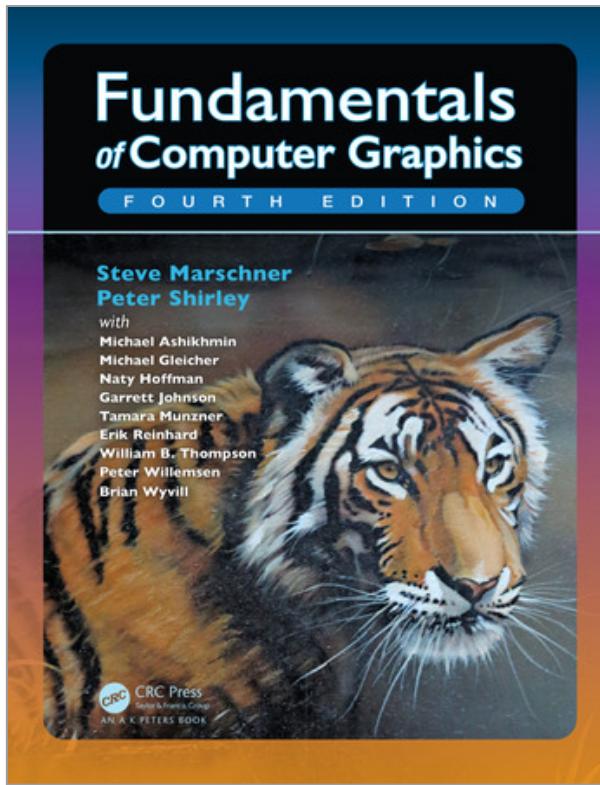
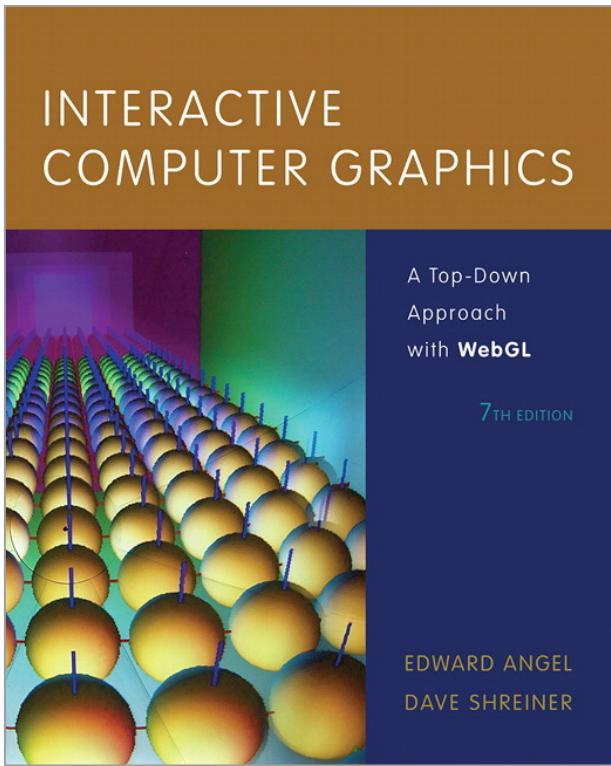
- Explain and apply the core concepts of computer graphics.
- Understand and use the different components of a programmable graphics pipeline.
- Understand and use different spatial data structures.
- Implement visual effects such as shadows.
- Implement interactive computer graphics programs using WebGL.

# CS425: Course goals

---

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

# Material



# 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

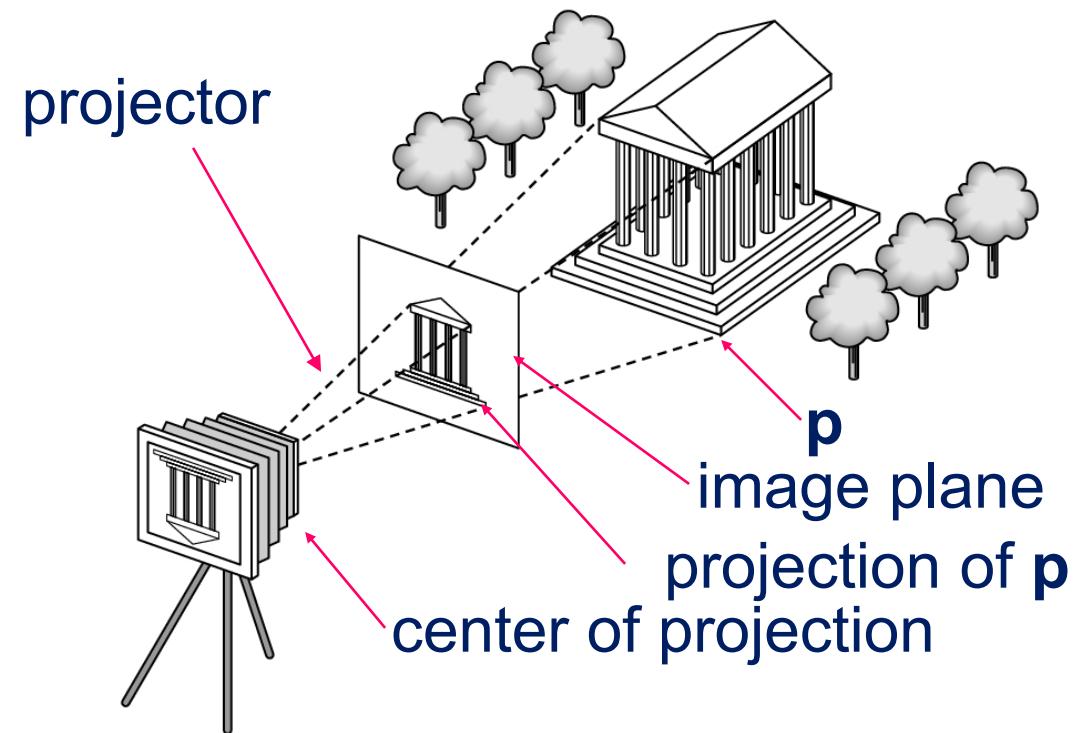
- We will meet twice a week:
  - Tuesday 11:00am – 12:15pm (Central)
  - Thursday 11:00am – 12:15pm (Central)
- Synchronous classes on Zoom, with recordings available later.
- Use Zoom chat to ask questions, or to indicate you want to say something. **Please interrupt me at any time to ask questions.**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



# WebGL & web programming



Click and drag to rotate teapot.  
[Original demo from Q3D.](#)

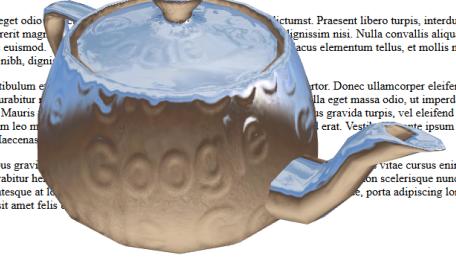
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse est ligula, egestas sed tempor vitae, pellentesque at massa. Nam a aliquet neque. Nulla a mauris eu lectus faucibus consectetur quis sed sem. Quisque posuere malesuada elementum. Etiam tortor purus, eleifend sit amet mattis vitae, ultrices 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, eger fringilla lectus consequat quis. Nunc suscipit vel arcu. In tempor 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.

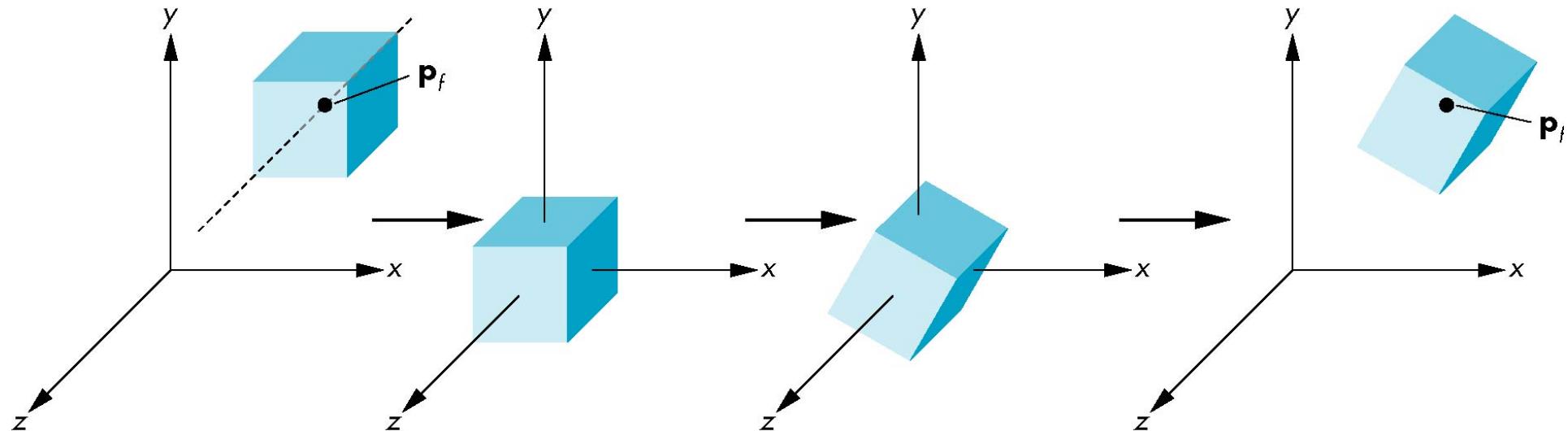
Vestibulum quam augue, vehicula ut congue nec, pulvinar in risus. Morbi eget odio, lacinia et ultricies ut, viverra non odio. Phasellus potenti. Phasellus venenatis, leo et accumsan ullamcorper, orci tortor hendrerit magna. Phasellus purus. Phasellus accumsan odio lacinia nisl auctor eget tempor nunc euismod. Phasellus euismod, ullamcorper massa ultricies eget fermentum quam consequat: Proin augue nibh, dignissim nisi. Nulla convallis aliquam est, eu interdum metus malesuada non. Sed in diam, tincidunt tellus, et mollis nibh sapien lectus dolor. Praesent egestas purus.

Vivamus at mauris velit. Pellentesque feugiat dapibus orci eu gravida. Vestibulum euismod, nisl ut fermentum, dolor sapien ultricies, turpis metus. Proin ac ante sem, quis egestas mi. Sed ultricies laoreet elit nec tempus. Curabitur laoreet, sapien et volutpat ultricies, nisl enim. Vivamus mattis diam sagittis risus volutpat at condimentum ante ultricies. Mauris non massa non massa. Quisque auctor, magna euismod. Ut sed nisi nec tortor suscipit posuere at quis sapien. Pellentesque at lectus, auctor porttitor vestibulum. Donec ullamcorper eleifend magna, sit amet ultricies quam placerat ut. Sed ultricies non magna, ut imperdiet lacus. Sed venenatis bibendum faucibus. Vivamus gravida turpis, vel eleifend risus malesuada consectetur. Ut non massa non massa non massa. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Praesent vestibulum odio sit amet nunc interdum sagittis. Maecenas.

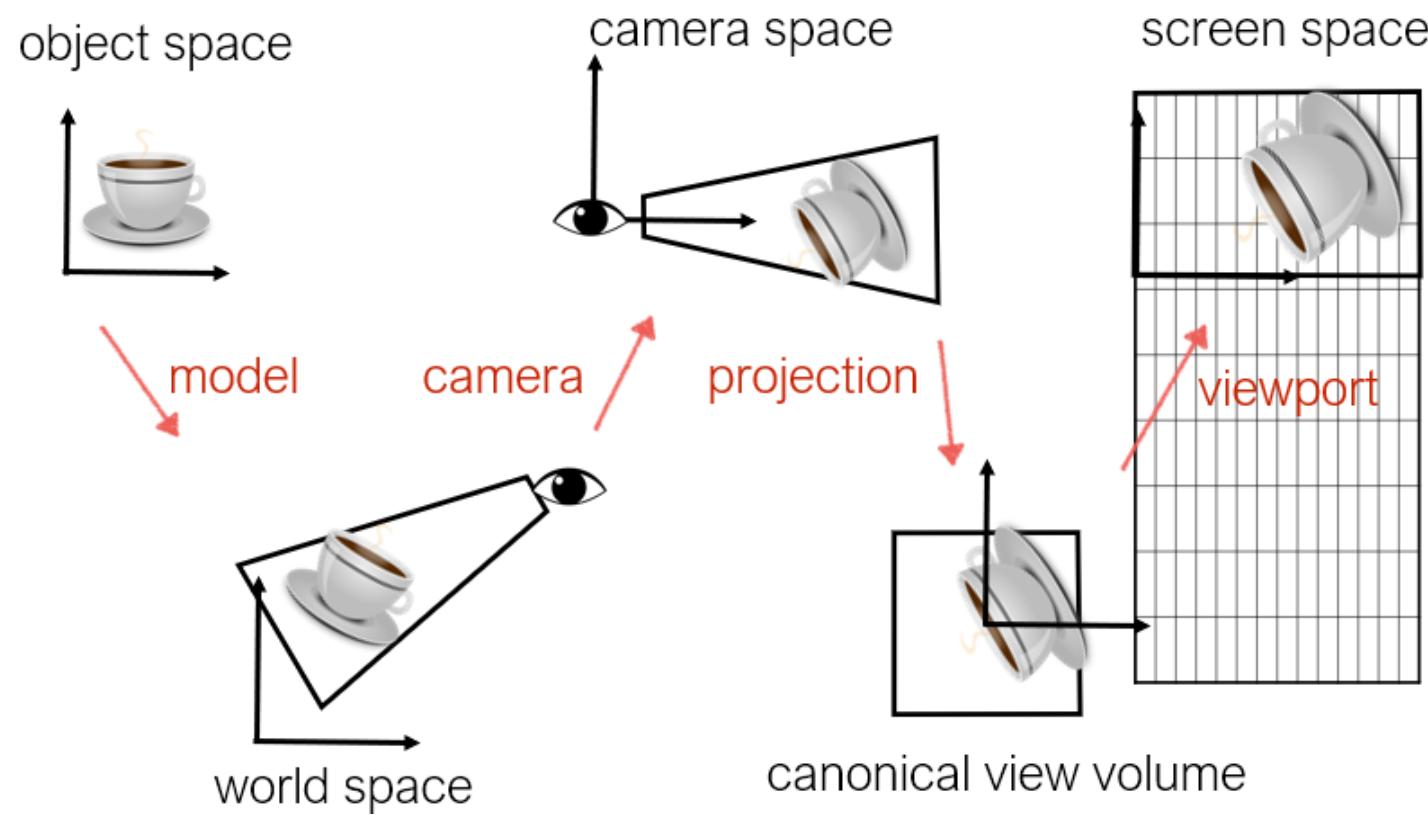
Vestibulum placerat scelerisque lectus eu facilisis. Phasellus eleifend tempus gravida, enim. Vivamus arcu justo, commodo ut luctus ac, molestie vitae enim. Curabitur laoreet, sapien in ultricies, nisl enim. Vivamus mattis diam sagittis risus volutpat at condimentum ante ultricies. Mauris non massa non massa. Quisque auctor, magna euismod. Ut sed nisi nec tortor suscipit posuere at quis sapien. Pellentesque at lectus, auctor porttitor vestibulum. Quisque sit amet felis.



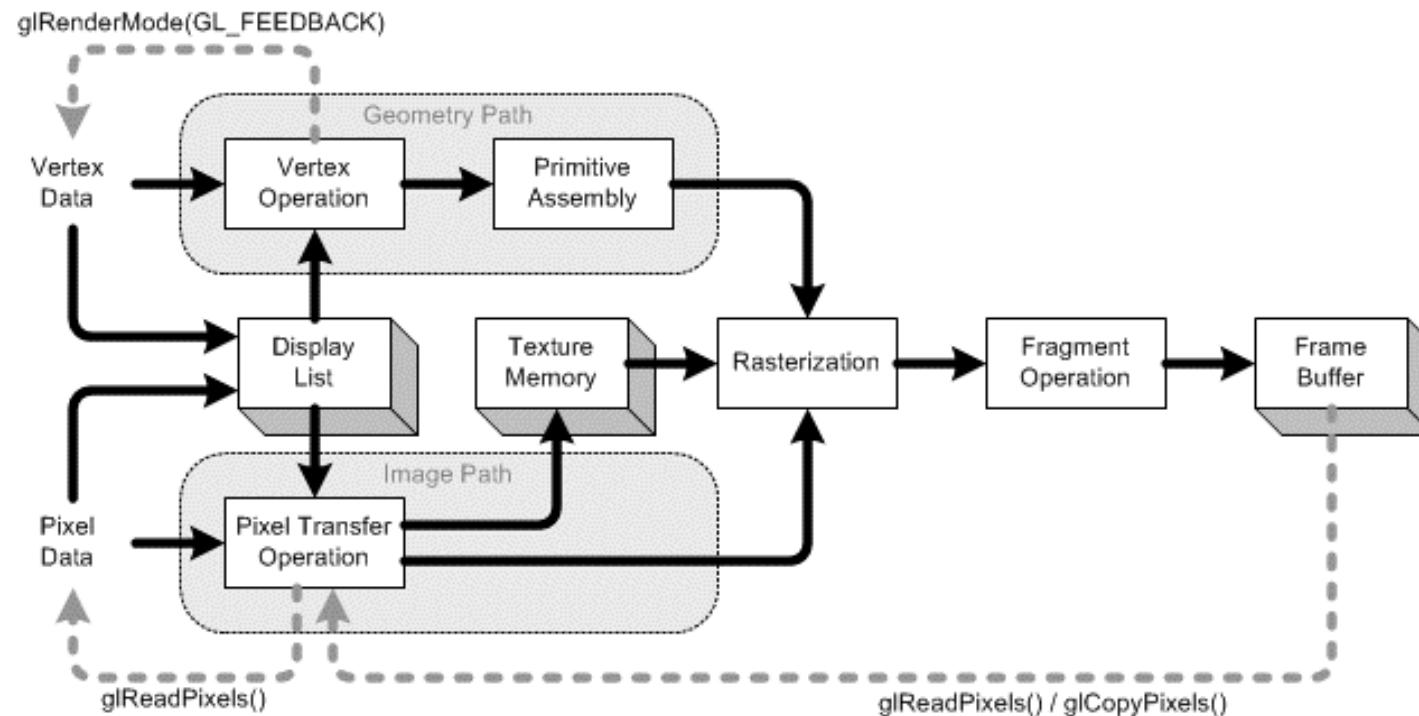
# Linear algebra and transformations



# Viewer transformations and rasterization

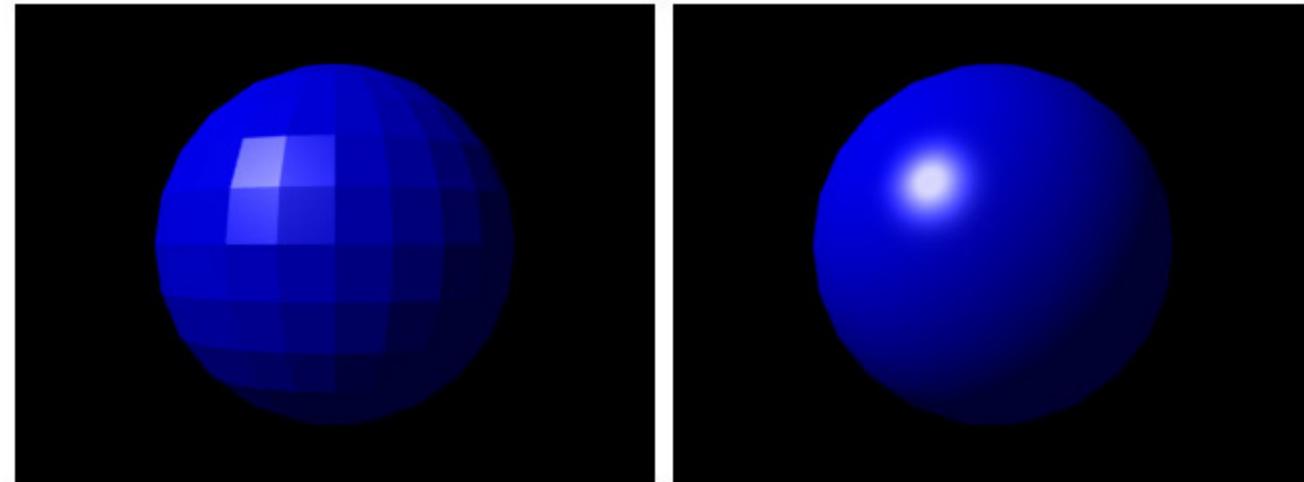


# Graphics pipeline

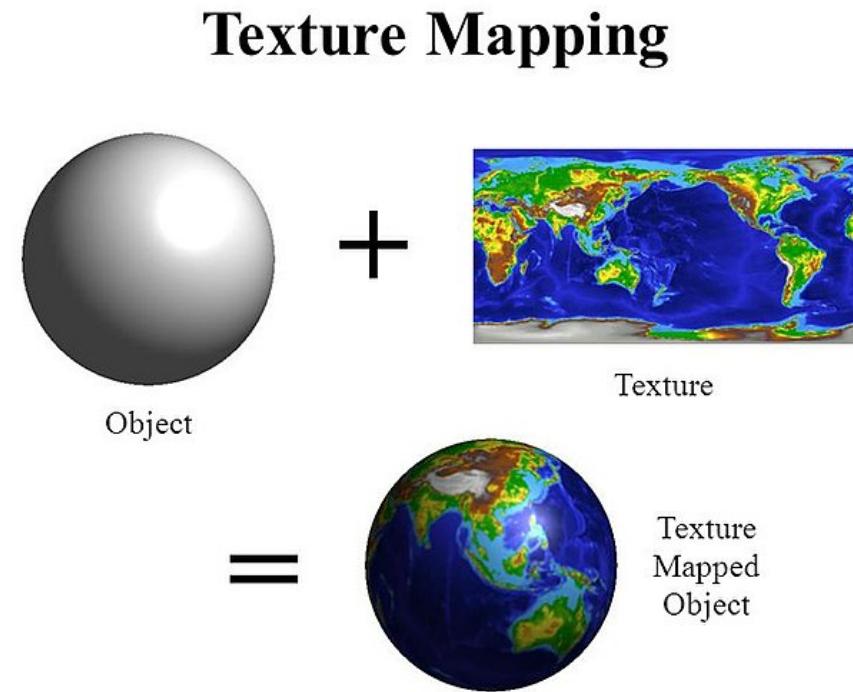


From: [songho.ca](http://songho.ca)

# Light and shading

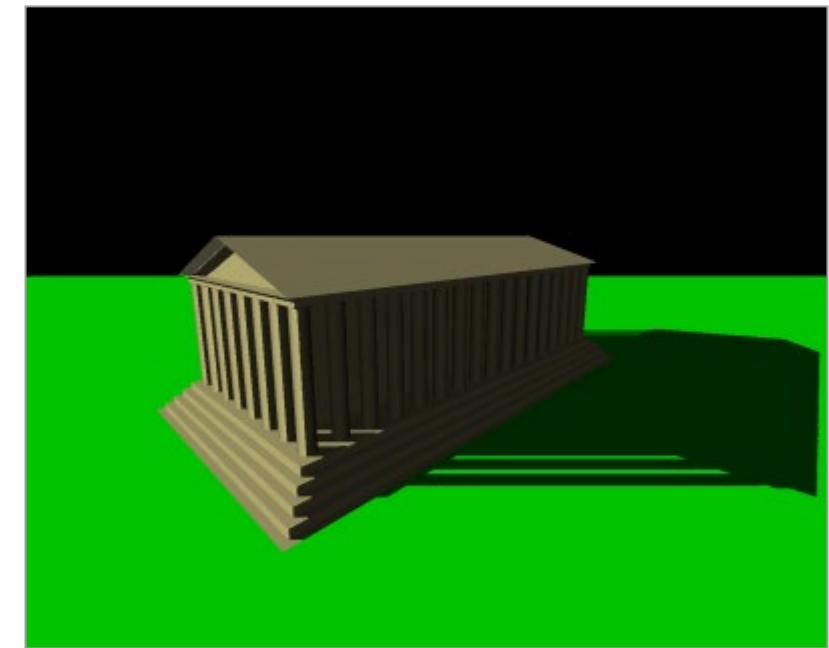
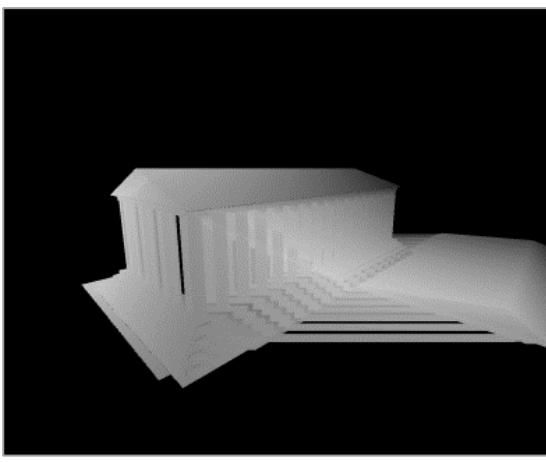
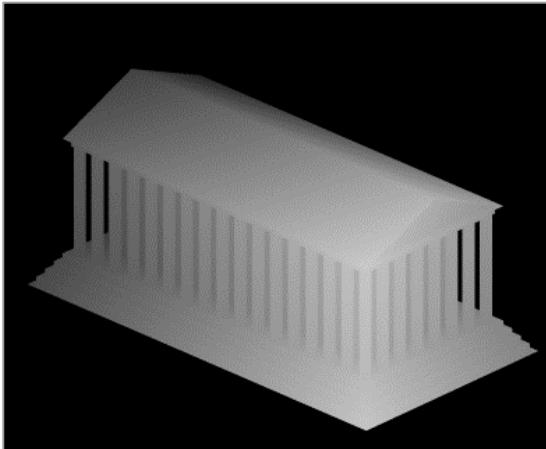
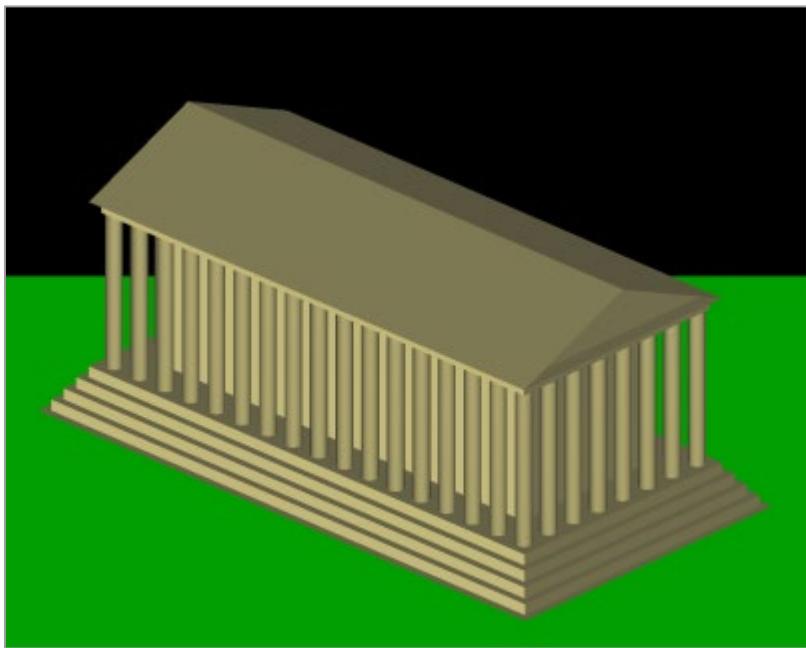


# Texture mapping

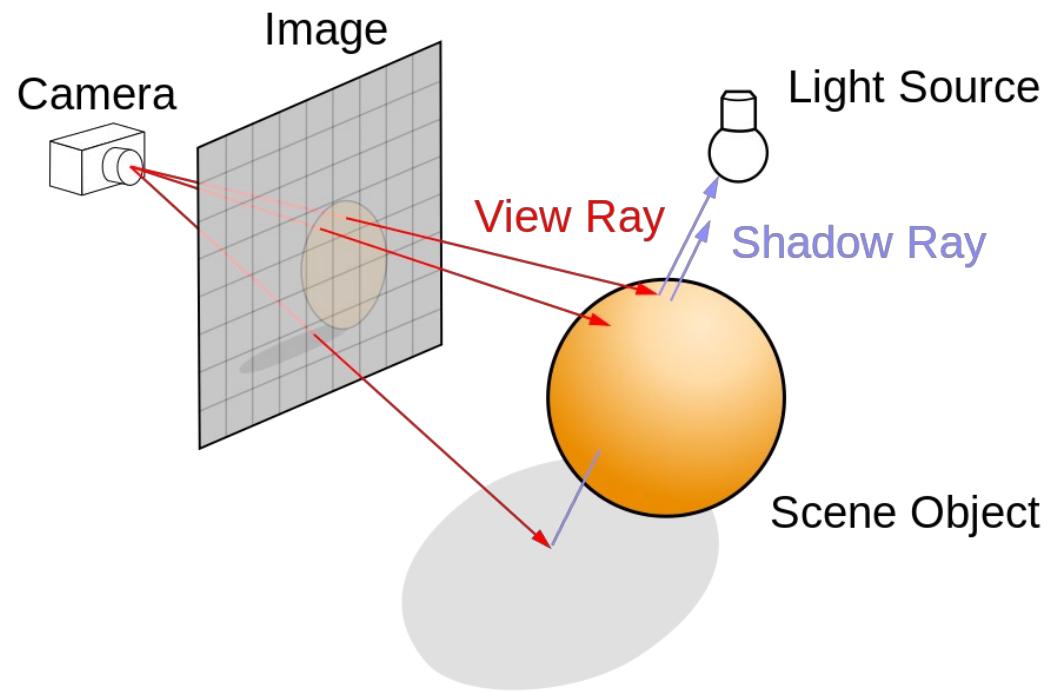


# Shadows

---

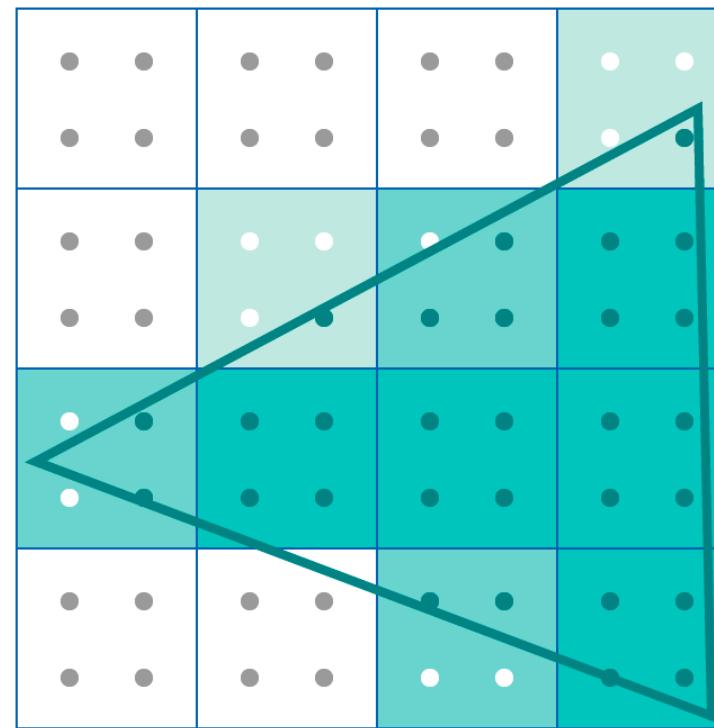


# Ray tracing



# Antialiasing

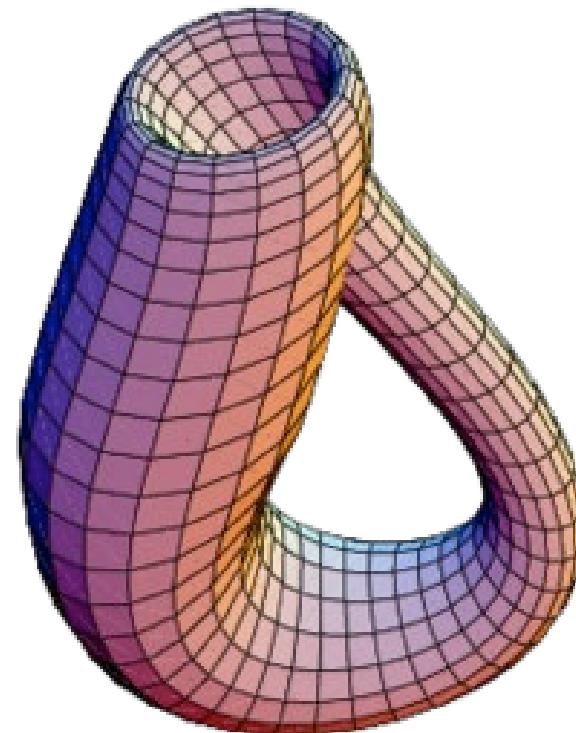
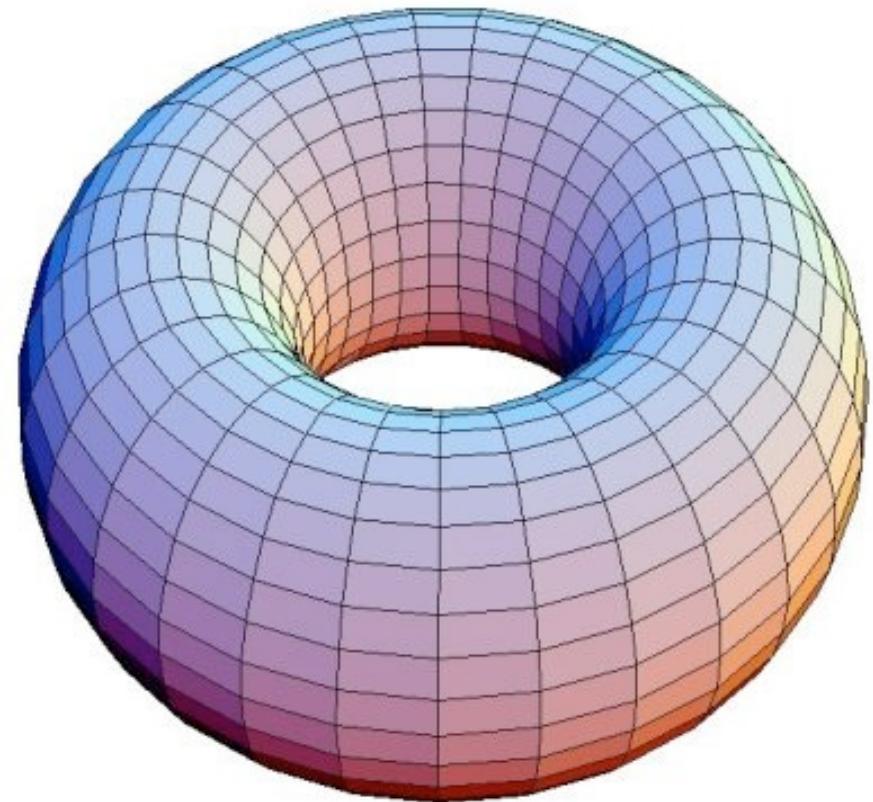
---



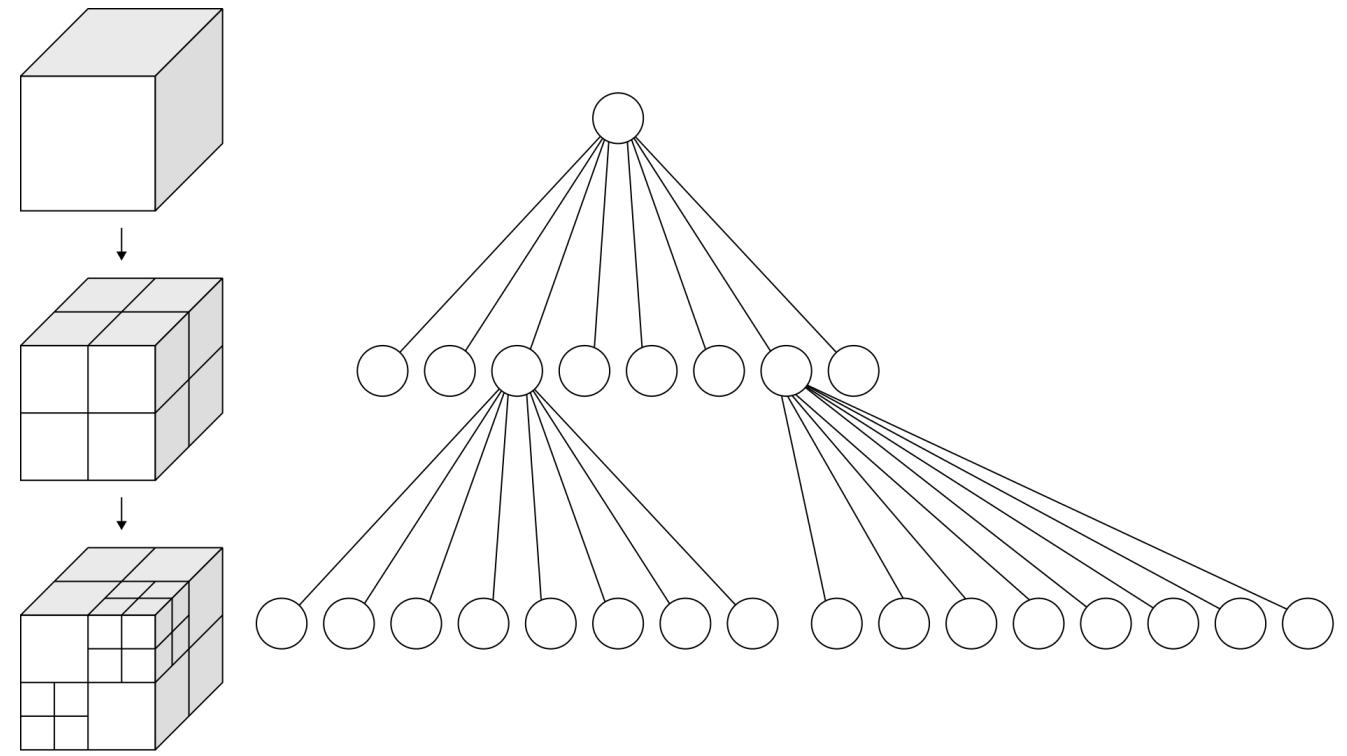
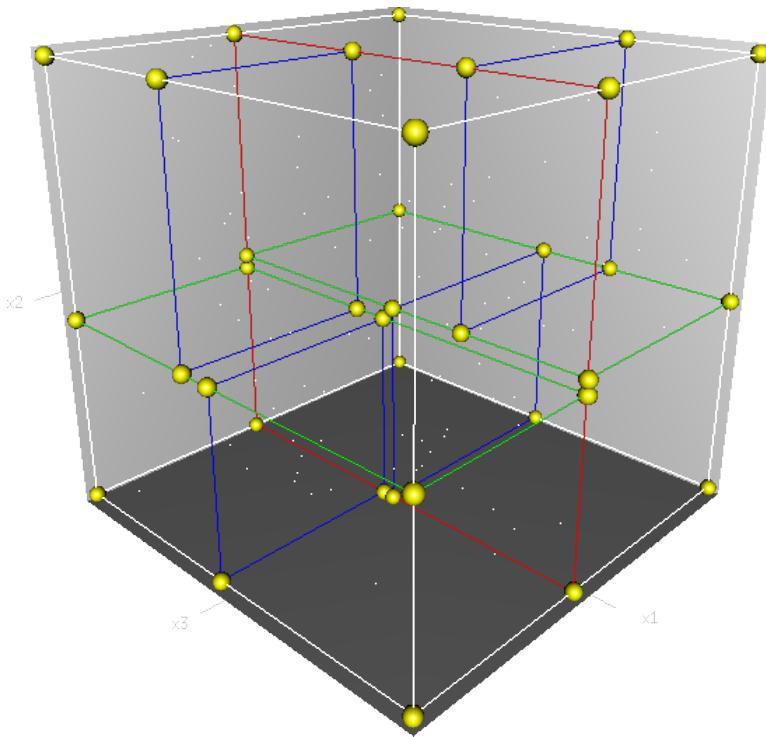
Apple MSAA

# Curves and surfaces

---



# Spatial data structures



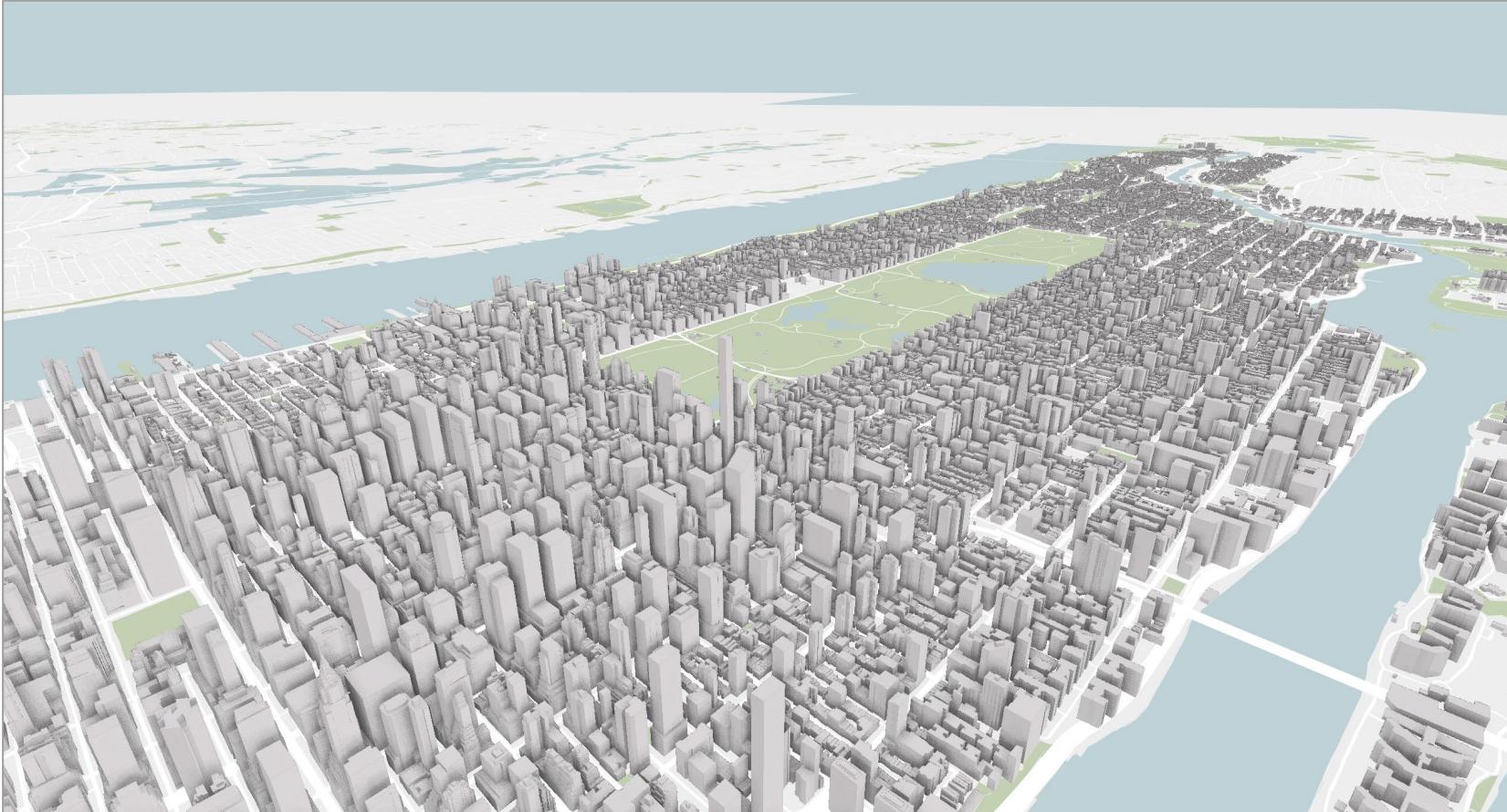
# Assignments

---

- 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.

# Resources

---

<https://open.gl>

<https://webgl2fundamentals.org/>

<https://songho.ca/opengl/>

# 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, Carlício 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, lequeiroz, carlucio}@gmail.com  
{chaimo}@dcc.ufmg.br

### 1 Introdução

O desenvolvimento de jogos é uma atividade complexa e que envolve diversas áreas: computação, arte e roteiro. Com o intuito de facilitar o desenvolvimento e 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 plataforma 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 várias áreas envolvidas na criação de jogos, como editor de fases e editor de eventos.

Alguns sistemas mais sofisticados e que provêm 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 *Game制作*. 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 os editores implementados. Na subseção 2.3 é feita uma discussão de como se dá o fluxo de desenvolvimento de um jogo com a adição da Pandorga. El finalmente na seção 3 está a conclusão e as perspectivas para trabalhos futuros.

VII SBGames - ISBN: 85-768-9216-3

### 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* [Goshin and Mine 2004], de autoria da *Disney* e que possui o seu código aberto. Apesar de possuir seu código aberto, o motor gráfico é proprietário e limitado 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. Devido a isso, optamos pelo uso de um motor de física separado: optamos pelo *ODE* (mais especificamente 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 evitaria complicações com futuras versões do *Panda3D*. A utilização dos motores possibilitou que nos concentrassemos 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*, o que determinará qual editor o jogo se encontra. Ao contrário de outras plataformas de criação de jogos, iniciadas 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>

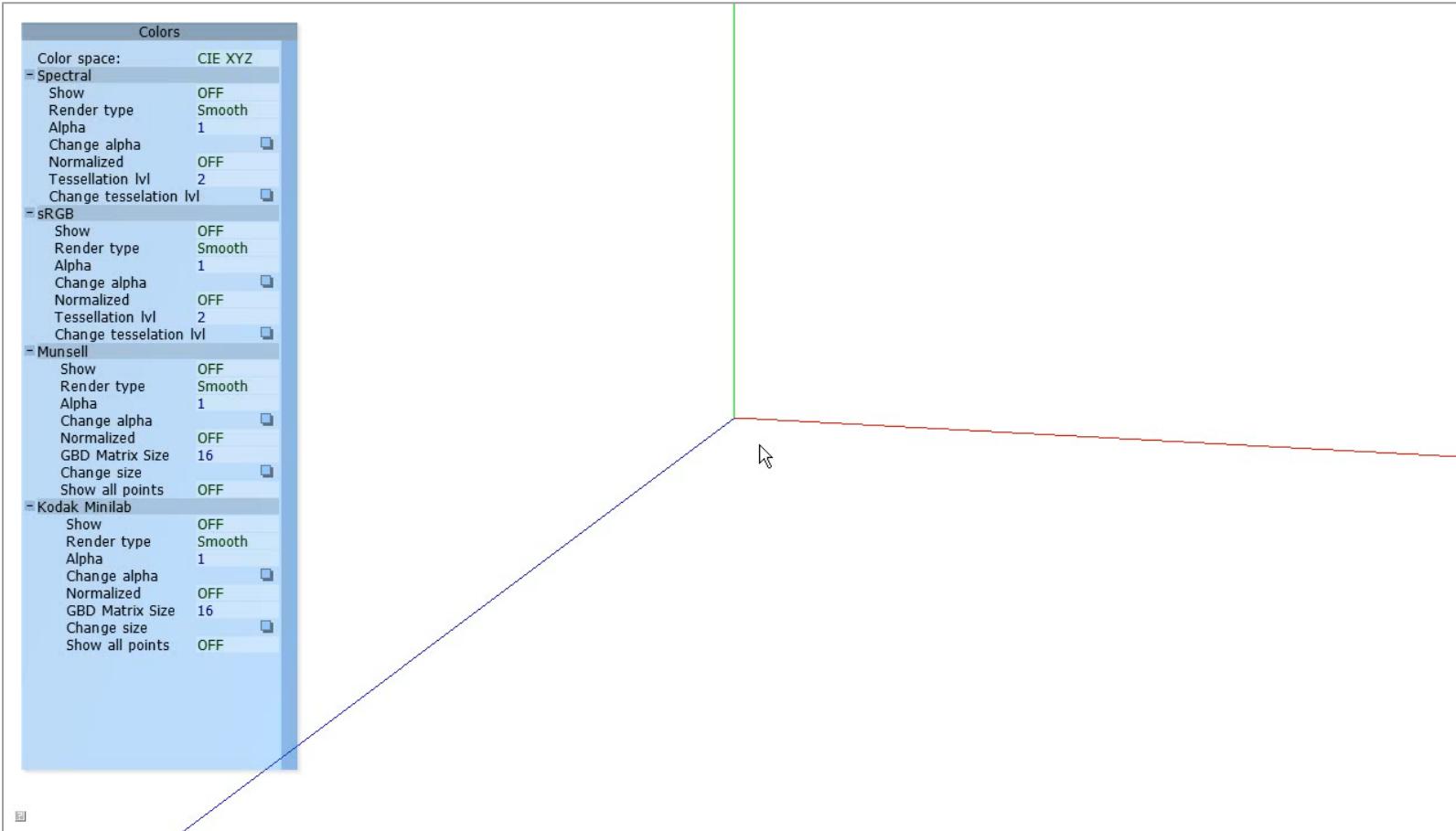
## Final note

---



[https://youtu.be/FLP4\\_CGs2Ng](https://youtu.be/FLP4_CGs2Ng)

# Final note



[https://youtu.be/cvGCO9u\\_los](https://youtu.be/cvGCO9u_los)

***“Without the fun, none of us would go on”***  
**Technology and Courage**  
**Ivan Sutherland**