

COMPUTER GRAPHICS FOR GAMES

P2 2025-26

Carlos Martinho



Epic Games Student Showcase 2024

**“WHAT I CANNOT CREATE,
I DO NOT UNDERSTAND.”**

Richard Feynman

OUTLINE

- Course introduction
- Importance of Computer Graphics
- Brief history of Computer Graphics
- Quarter planning

GOALS

GOALS

- Develop a real-time graphics engine ‘from scratch’ using modern 3D graphics programming with GPU.
(technical challenge)
- Understand the theory and the algorithms underlying modern 3D graphics systems.
(intellectual challenge)
- Development will be in Modern C++ / OpenGL / GLSL.
- Help creating a strong portfolio.

PREREQUISITES

PREREQUISITES

- Linear algebra (review in the context of Computer Graphics and addenda next lectures).
- Proficient object programming skills. Good knowledge of C++ and STL will help.
- Knowledge of Computer Graphics (e.g. image formats, raster algorithms, etc.) may help.

GRADING

BASE PROJECT

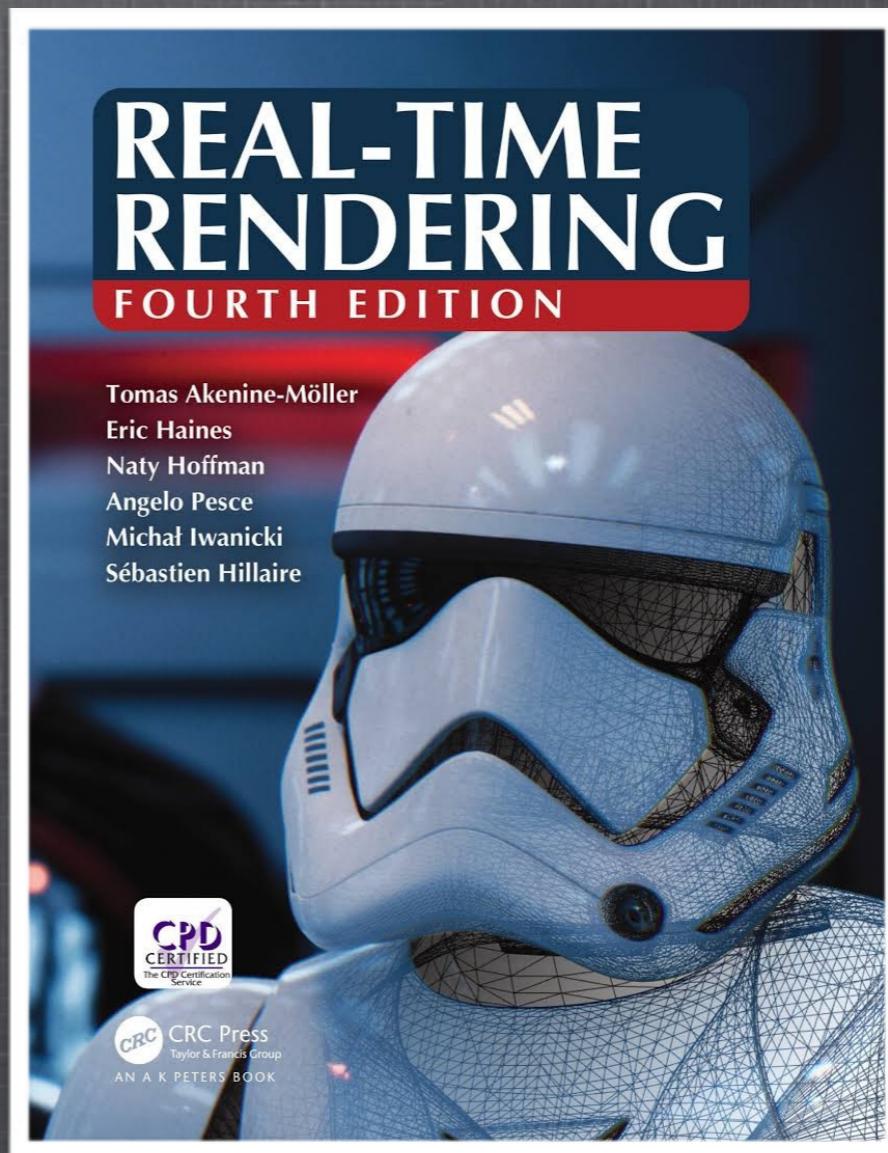
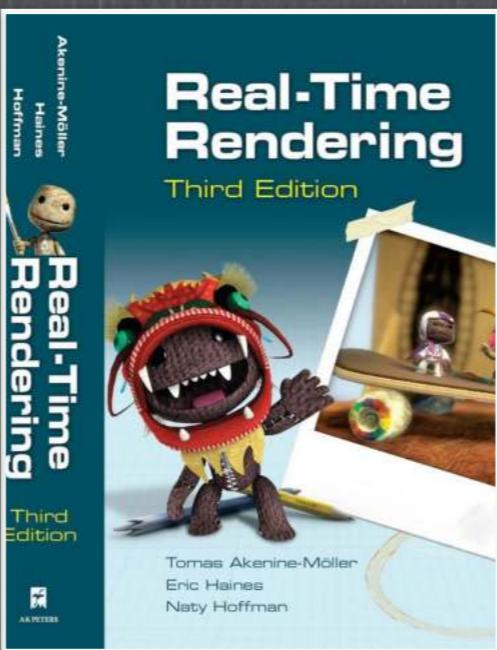
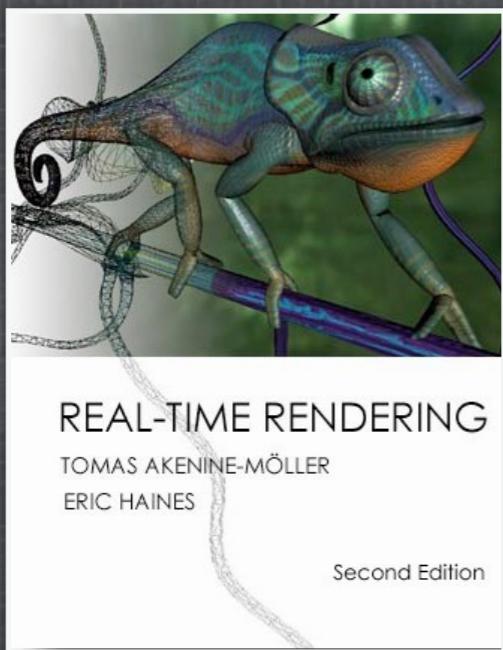
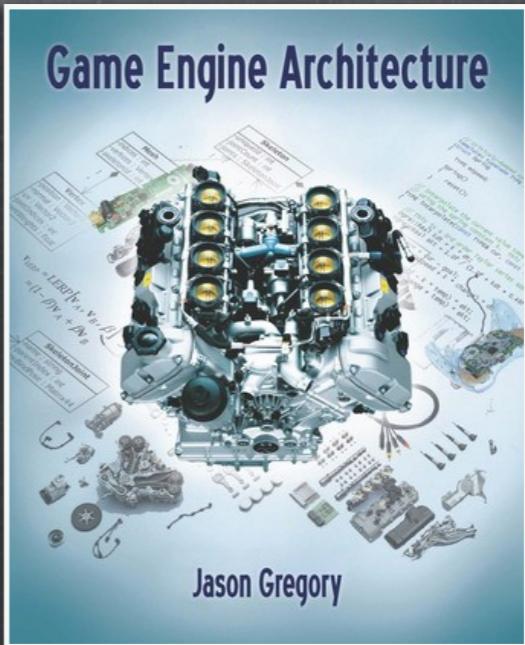
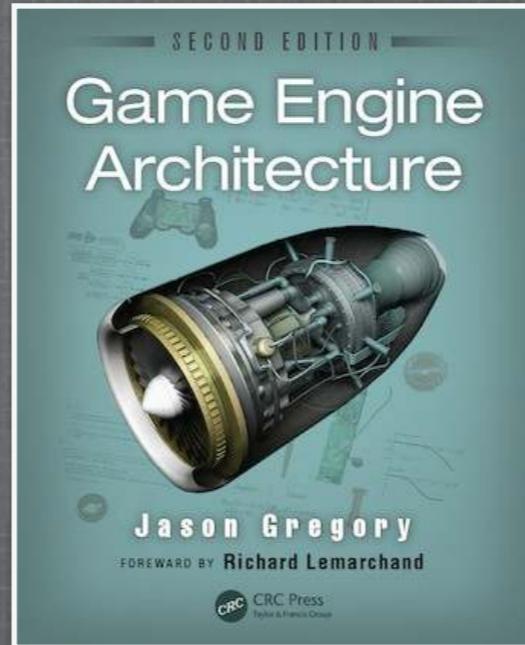
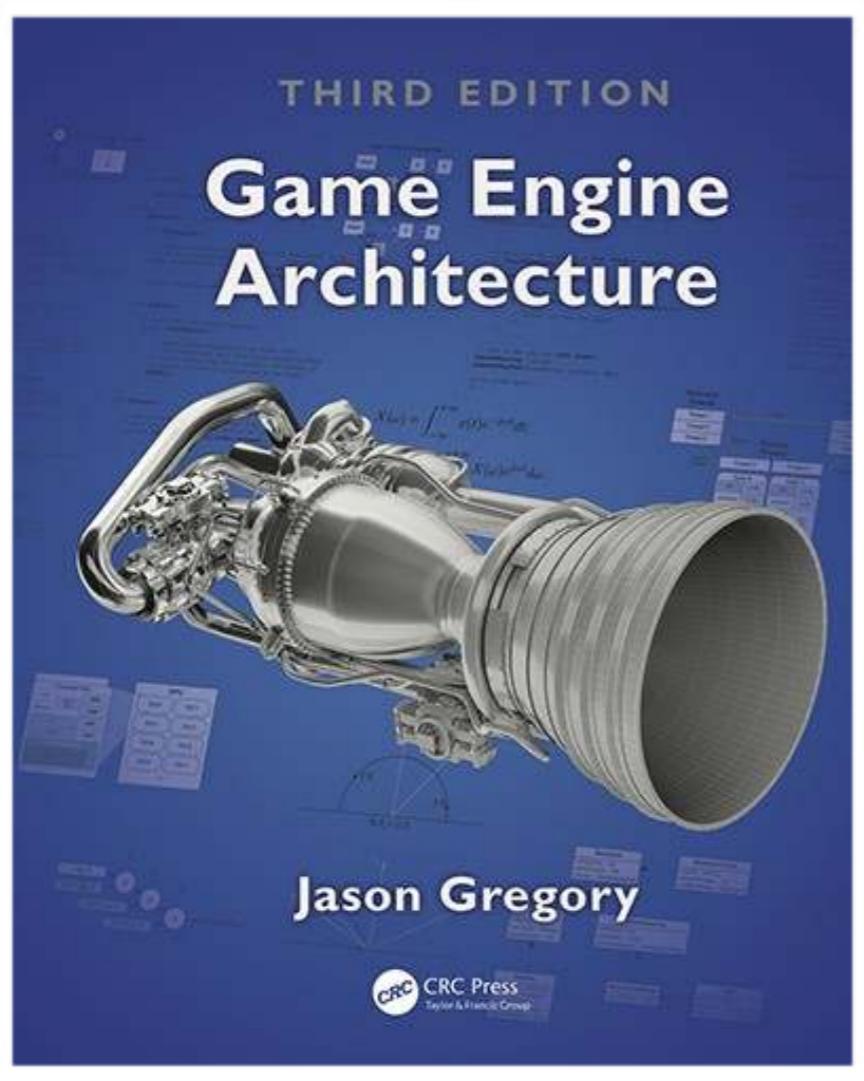
(T2, 40%, >9.5)

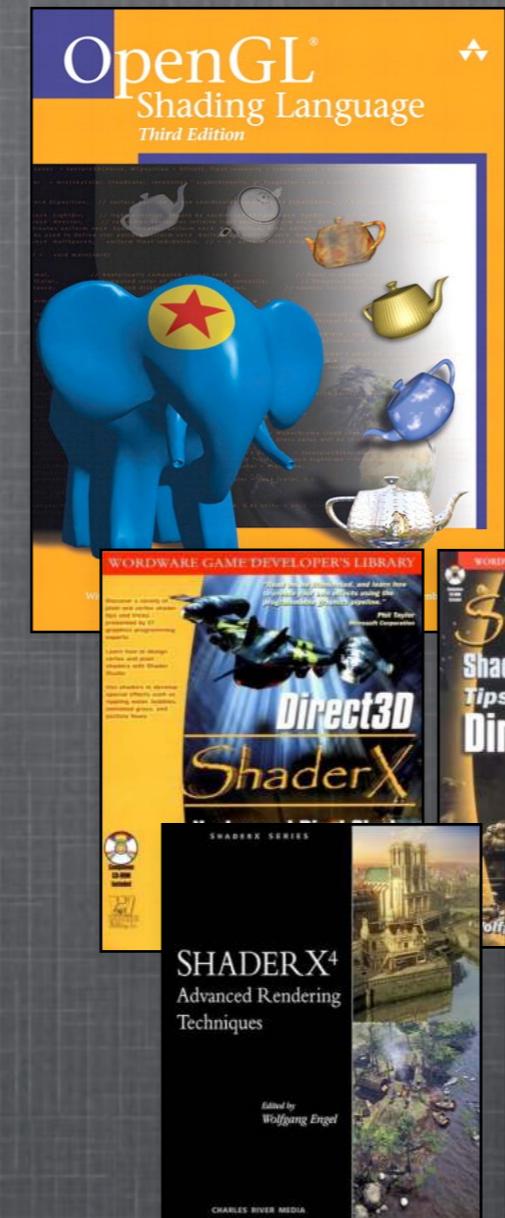
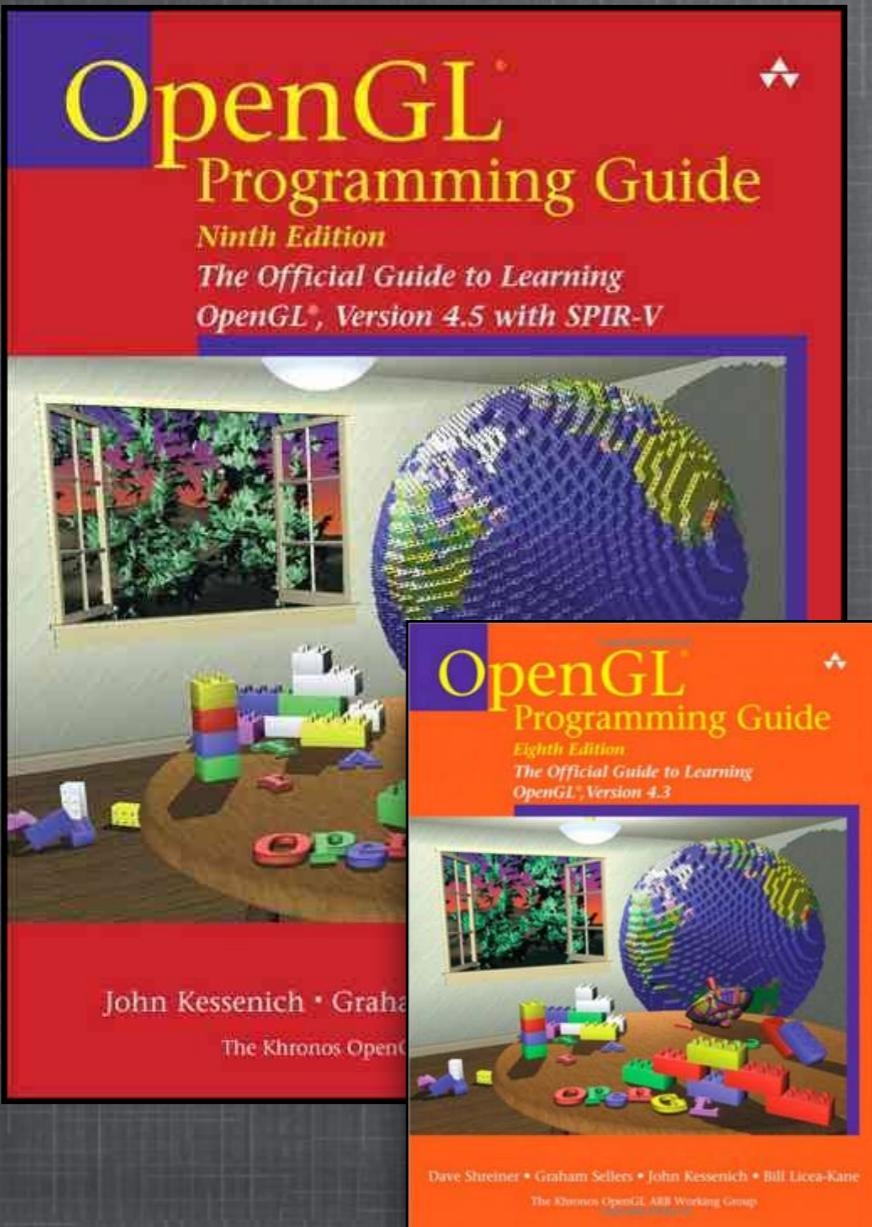
+

SHADER PROJECT

(T1, 60%, >9.5)

BIBLIOGRAPHY





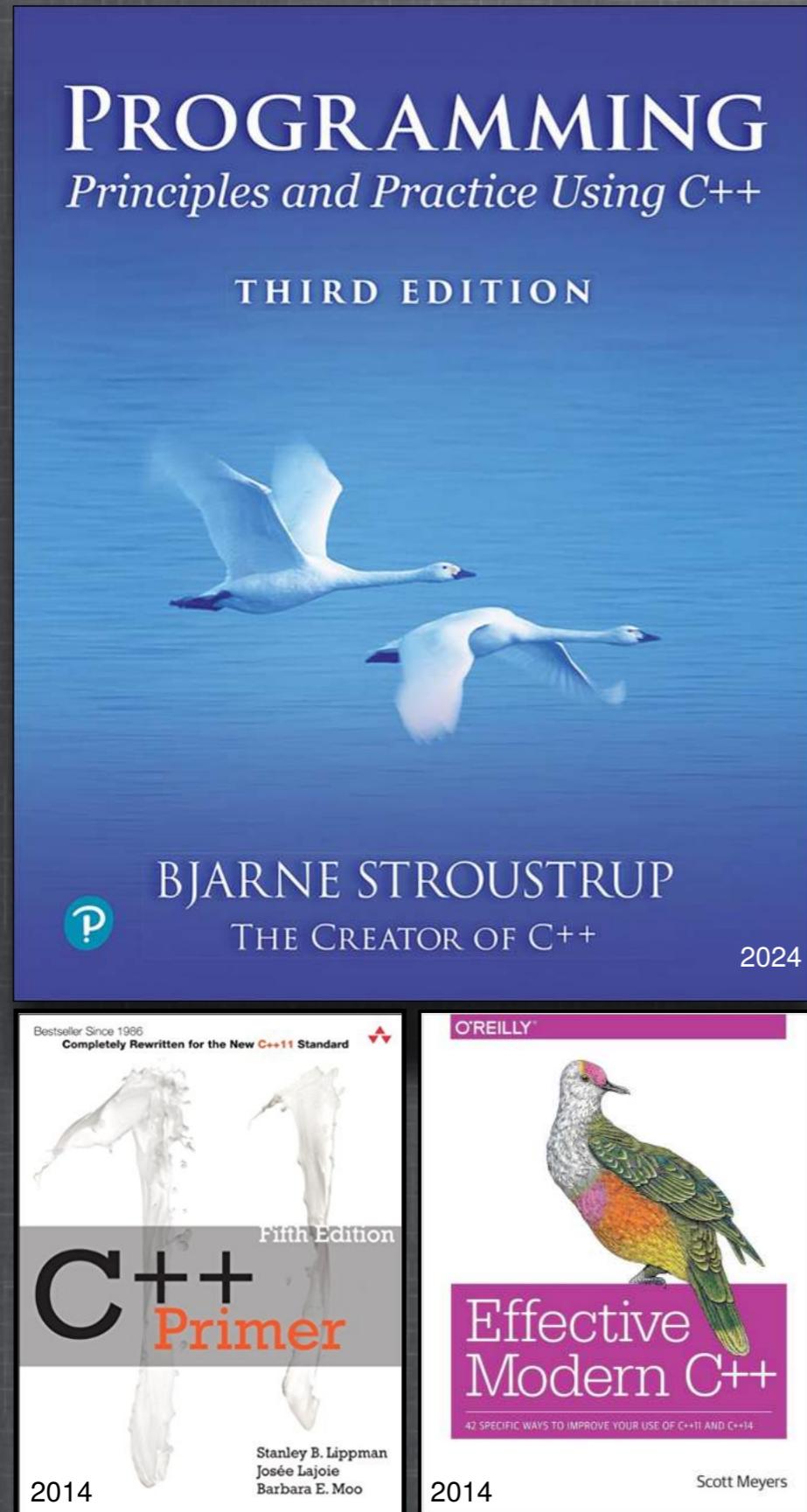
2004-2007



2002-2009



2010-2016



cppreference.com [Create account](#)

[Page](#) [Discussion](#) Standard revision: [Diff](#)

C++ reference

C++11, C++14, C++17, C++20, C++23, C++26 | Compiler support C++11, C++14, C++17, C++20,

Language	Diagnostics library	Strings
Keywords – Preprocessor ASCII chart Basic concepts Comments Names (lookup) Types (fundamental types) The main function Expressions Value categories Evaluation order Operators (precedence) Conversions – Literals Statements if – switch for – range-for (C++11) while – do-while Declarations – Initialization Functions – Overloading Classes (unions) Templates – Exceptions Freestanding implementations	Assertions – System error (C++11) Exception types – Error numbers basic_stacktrace (C++23) Debugging support (C++26)	basic basic Null-te byte
Standard library (headers)	Memory management library	Text pro
Named requirements	Allocators – Smart pointers Memory resources (C++17)	Primiti Forma locale text_e Regula bas Defa
Feature test macros (C++20)	Metaprogramming library (C++11)	Numeric
Language – Standard library	Type traits – ratio integer_sequence (C++14)	Comm Mathe Mathe Basic I Pseudo Floating comple
Language support library	General utilities library	Date an
Program utilities Signals – Non-local jumps Basic memory management Variadic functions source_location (C++20) Coroutine support (C++20) Comparison utilities (C++20) Type support – type_info numeric_limits – exception initializer_list (C++11)	Function objects – hash (C++11) Swap – Type operations (C++11) Integer comparison (C++20) pair – tuple (C++11) optional (C++17) expected (C++23) variant (C++17) – any (C++17) bitset – Bit manipulation (C++20)	Calend
Concepts library (C++20)	Containers library	Input/ou
	vector – deque – array (C++11) list – forward_list (C++11) map – multimap – set – multiset unordered_map (C++11) unordered_multimap (C++11) unordered_set (C++11) unordered_multiset (C++11) Container adaptors span (C++20) – mspan (C++23)	Print fu Strea basic Synci File sy
Technical specifications	Iterators library	Concurr
Standard library extensions (library fundamentals TS)	Range factories – Range adaptors generator (C++23)	thread atomic atomic Mutua Condit latch Safe R
resource_adaptor – invocation_type	Ranges library (C++20)	Executio
Standard library extensions v2 (library fundamentals TS v2)	Range factories – Range adaptors generator (C++23)	Parallel (parallelism simd
propagate_const – ostream_joiner – randint observer_ptr – Detection idiom	Algorithms library	Concurr (concurrent Transact Reflectio
Standard library extensions v3 (library fundamentals TS v3)	Numeric algorithms Execution policies (C++17) Constrained algorithms (C++20)	
scope_exit – scope_fail – scope_success – unique_resource		

[External Links](#) – [Non-ANSI/ISO Libraries](#) – [Index](#) – [std Symbol Index](#)

ONLINE LIBRARIES

The OpenGL Extension Wrangler Library (GLEW) is a cross-platform open-source C/C++ extension loading library. GLEW provides efficient run-time mechanisms for determining which OpenGL extensions are supported on the target platform. OpenGL core and extension functionality is exposed in a single header file. GLEW has been tested on a variety of operating systems, including Windows, Linux, Mac OS X, FreeBSD, Irix, and Solaris.

Downloads

GLEW is distributed as source and precompiled binaries. The latest release is 1.10.0[07-22-13]:

Source ZIP | TGZ
Binaries Windows 32-bit and 64-bit |

An up-to-date copy is also available using git:

```
github
git clone
https://github.com/nigels-com/glew.git glew
```

GLFW is an Open Source, multi-platform library for OpenGL, OpenGL ES and Vulkan development on the desktop. It provides a simple API for creating windows, contexts and surfaces, receiving input and events.

GLFW is written in C and supports Windows, macOS, the X Window System and the Wayland protocol.

GLFW is licensed under the zlib/libpng license.

macOS pre-compiled binaries
Posted on July 22, 2019
Pre-compiled macOS binaries for GLFW 3.3 are now available for download. macOS binaries will be included from the start in future releases.

Delphi bindings added
Posted on May 24, 2019
Erik van Bilzen has written the Delphi bindings [DelphiGlfw](#) that cover GLFW 3.2. They support Windows (32-bit and 64-bit) and macOS (32-bit) and come with HTML and HTML Help documentation.

D bindings released
Posted on April 25, 2019
Mike Parker has released the D bindings

OpenGL Mathematics (GLM) is a header only C++ mathematics library for graphics and game engines.

Downloads

Manual
API
Code samples
Snapshot
Updates
Downloads
Repository
Report a bug

GLSL Specification
GLSL man pages
Forum
Stack Overflow
OpenGL SDK page

It is a platform independent library with no dependence to external libraries. It can take advantage of C++11 when available. It supports the following:

- Clang 2.6 and higher
- CUDA 3.0 and higher
- GCC 3.4 and higher
- Intel C++ Composer XE 2013 and higher
- LLVM 2.3 through GCC 4.2 front-end and higher
- Visual C++ 2005 and higher
- Any conform C++98 or C++11 compiler

FreeImage is an Open Source library project for developers who would like to support popular graphics image formats like PNG, BMP, JPEG, TIFF and others as needed by today's multimedia applications. FreeImage is easy to use, fast, multithreading safe, compatible with all 32-bit or 64-bit versions of Windows, and cross-platform (works both with Linux and Mac OS X).

WHAT IS FREEIMAGE?
FreeImage is an Open Source library project for developers who would like to support popular graphics image formats like PNG, BMP, JPEG, TIFF and others as needed by today's multimedia applications. FreeImage is easy to use, fast, multithreading safe, compatible with all 32-bit or 64-bit versions of Windows, and cross-platform (works both with Linux and Mac OS X).

DOWNLOAD NOW
You can download FreeImage distributions (source code, DLL and wrappers) from the downloads page.

DEVELOPER'S CORNER
Find all resources you need to build your application using FreeImage: Documentation, FAQ, Mailing List, Changes Log, and Sample Source Code.

Assimp is a portable Open Source library to import various well-known 3D model formats in a uniform manner. The most recent version also knows how to export 3d files and is therefore suitable as a general-purpose 3D model converter. See the [feature list](#).

What?

About
News
Progress
Help Out
Download
Install
API
OpenGL Wikibook
Report a Bug
Request a Feature
Project Interface

OpenGL
SOURCEFORGE.NET
W3C XHTML 1.0
View PHP Source

freeglut was originally written by Paweł W. Olszta with contributions from

CREATIVE CODING

OF [about](#) [download](#) [documentation](#) [learning](#) [gallery](#) [community](#) [development](#)
» [forum](#) » [github](#) » [addons](#) » [slack](#) » [blog](#) [日本語](#) [한국어](#) [简体中文](#)

openFrameworks is an [open source C++](#) toolkit designed to assist the creative process by providing a simple and intuitive framework for experimentation.



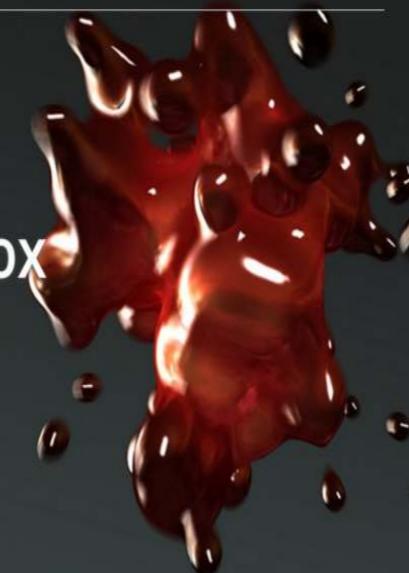
openFrameworks is designed to work as a general purpose glue, and wraps together several commonly used libraries, including:

- » [OpenGL, GLEW, GLUT, libtess2 and cairo](#) for graphics
- » [rtAudio, PortAudio, OpenAL and Kiss FFT or FMOD](#) for audio input, output and analysis
- » [FreeType](#) for fonts
- » [FreeImage](#) for image saving and loading
- » [Quicktime, GStreamer and vlcInput](#) for video playback and grabbing
- » [Poco](#) for a variety of utilities
- » [OpenCV](#) for computer vision
- » [Assimp](#) for 3D model loading

The code is written to be massively cross-compatible. Right now we support five operating systems (Windows, OSX, Linux, iOS, Android) and four IDEs (Xcode, Code::Blocks, and Visual Studio and Eclipse). The API is designed to be minimal and easy to grasp.

OPEN
FRAMEWORKS

CINDER



A Powerful, Intuitive Toolbox

Cinder is a free and open source library for professional-quality creative coding in C++.

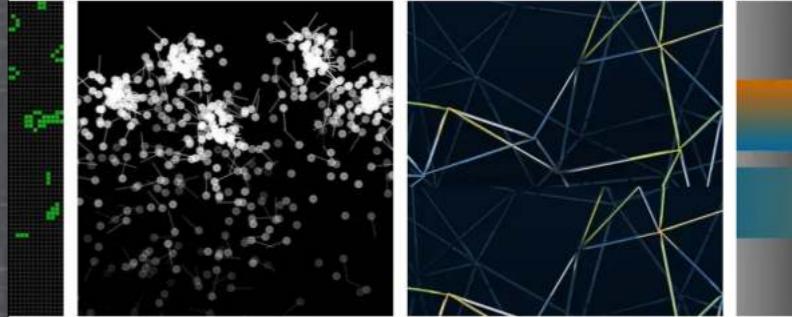
[SEE WHAT CINDER CAN DO](#) [DOWNLOAD CINDER](#)

Processing [Download](#) [Documentation](#) [Learn](#) [Teach](#) [About](#) [Donate](#)

Welcome to Processing!

Processing is a flexible software sketchbook and a language for learning how to code. Since 2001, Processing has promoted software literacy within the visual arts and visual literacy within technology. There are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning and prototyping.

[Download](#) [Reference](#) [Donate](#)



CINDER

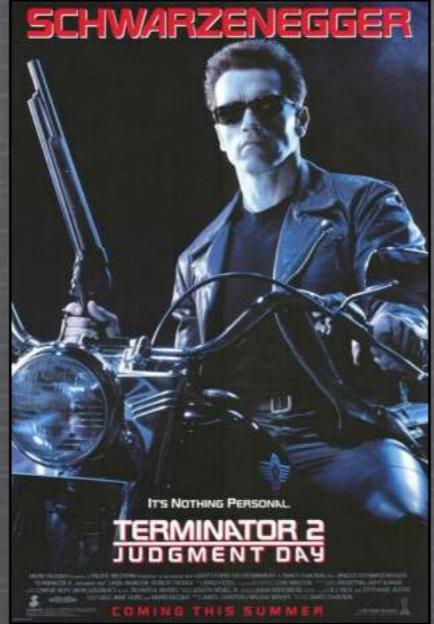
PROCESSING

IMPORTANCE OF COMPUTER GRAPHICS

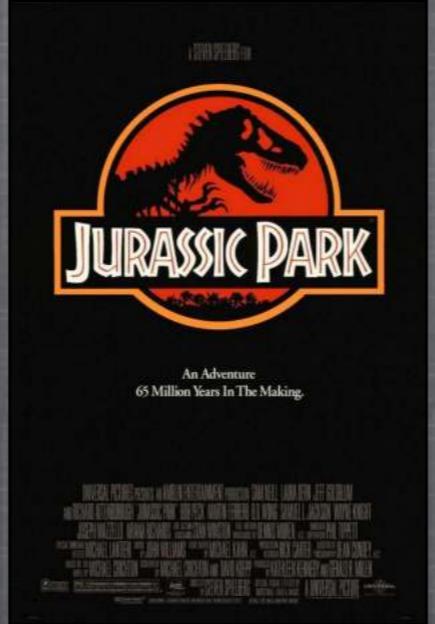
COMPUTER GRAPHICS IN MOVIES



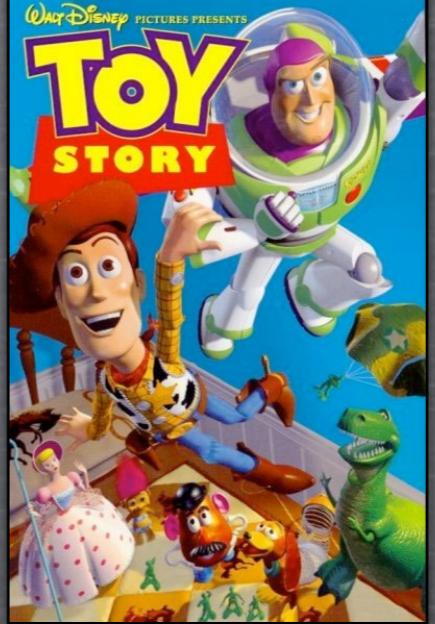
(1982)



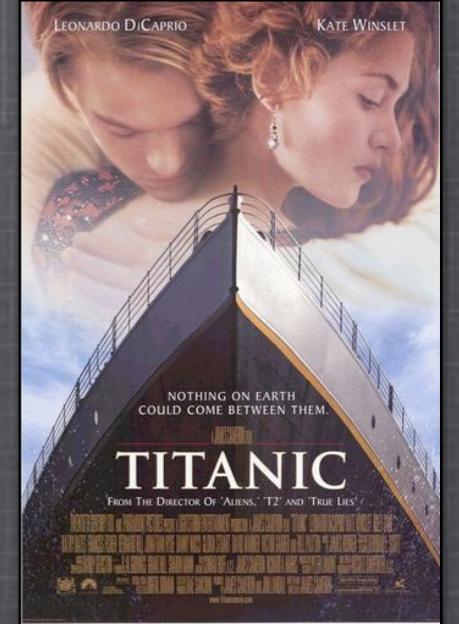
(1991)



(1993)



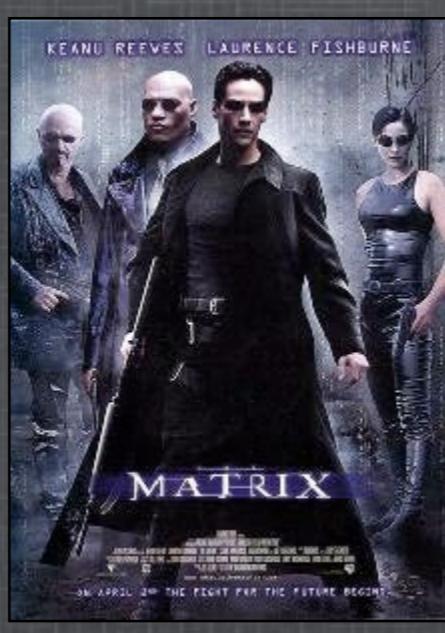
(1995)



(1997)



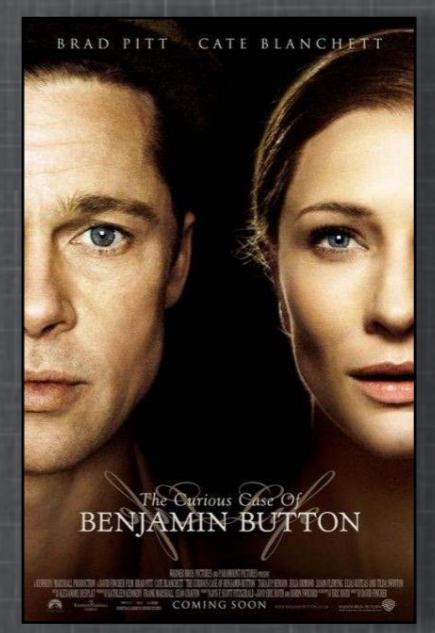
(1999,2002,2005)



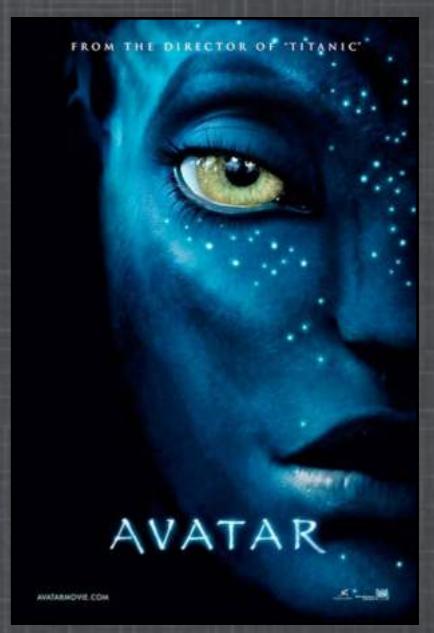
(1999)



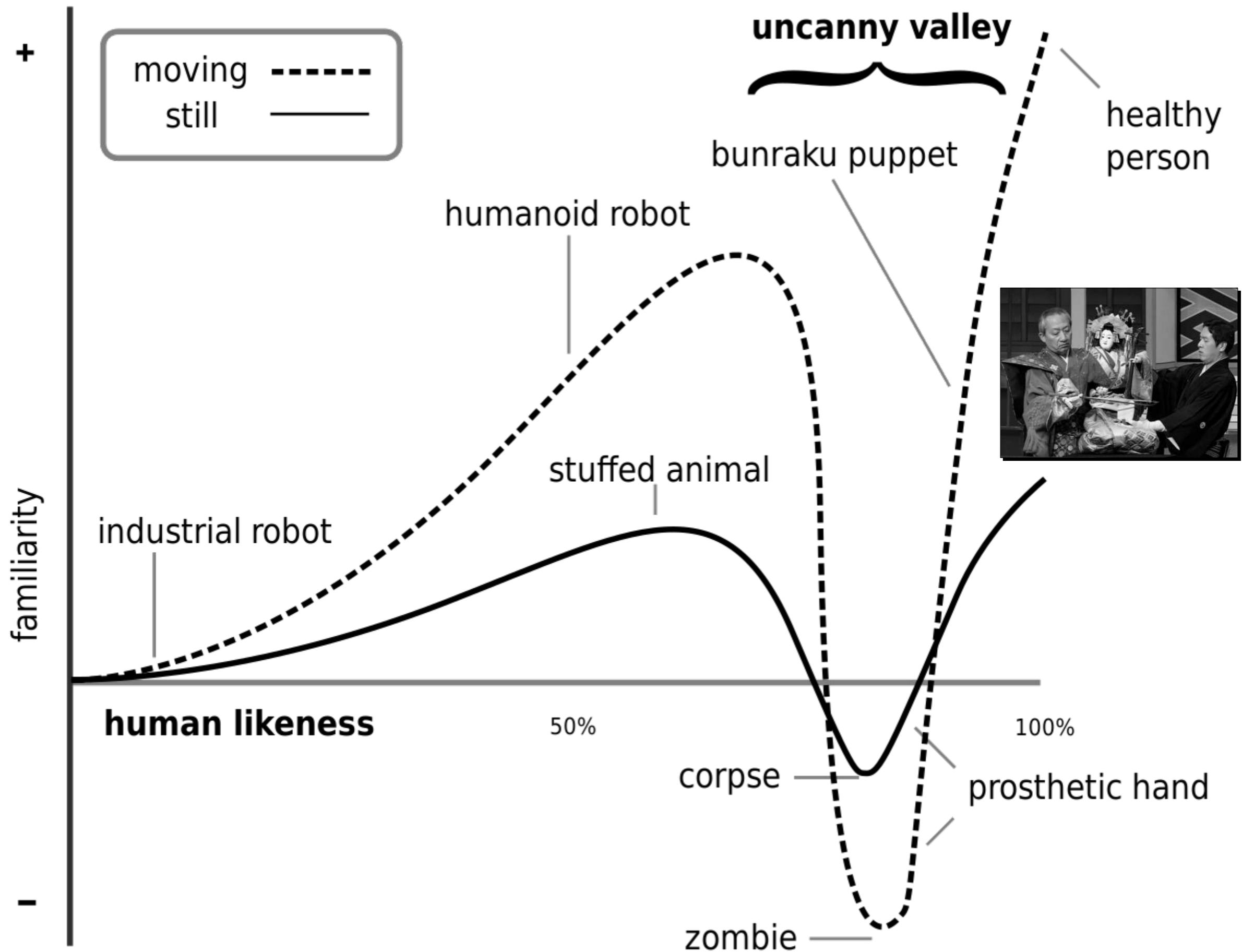
(2001, 2002, 2003)



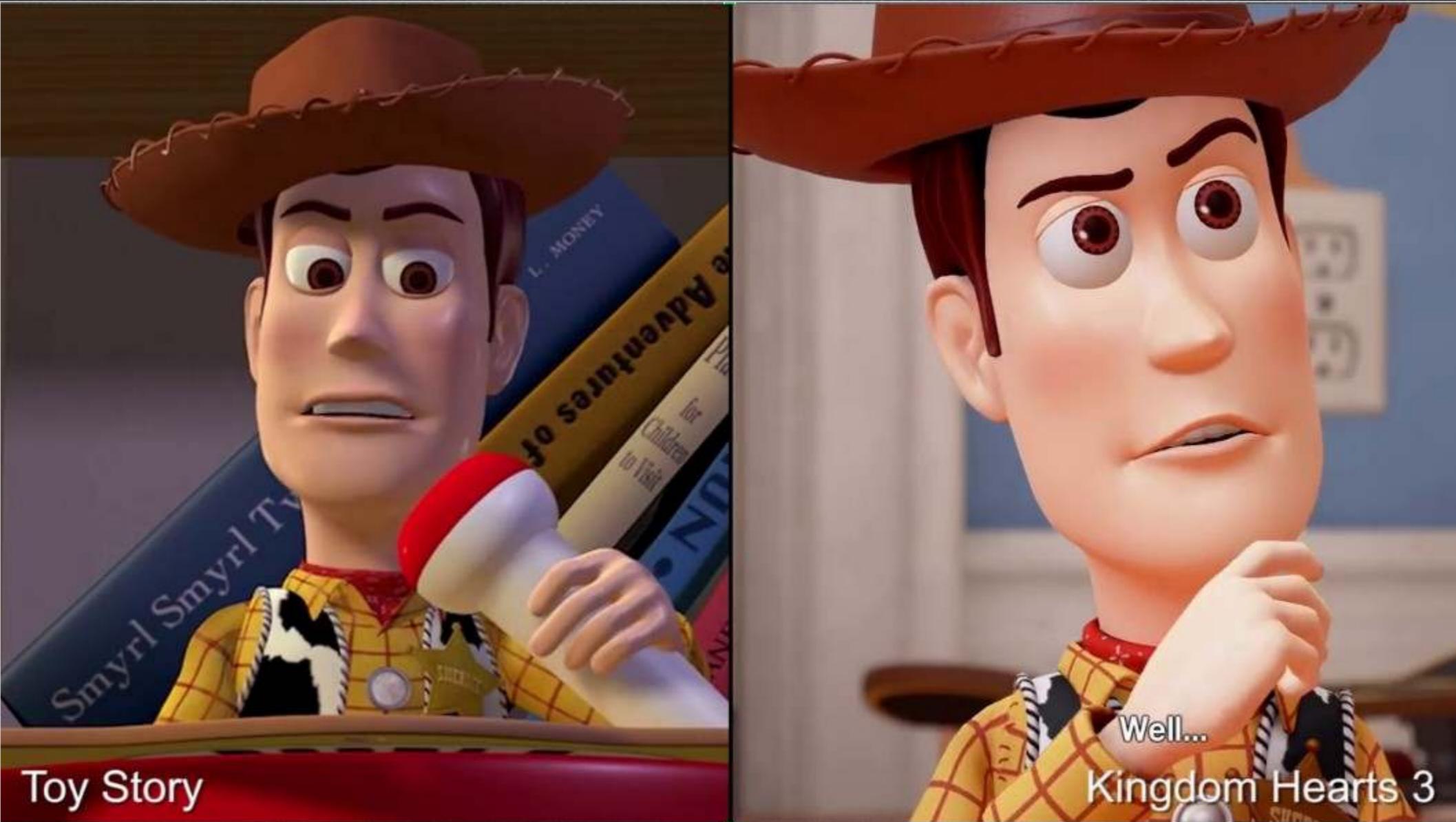
(2008)



(2009)



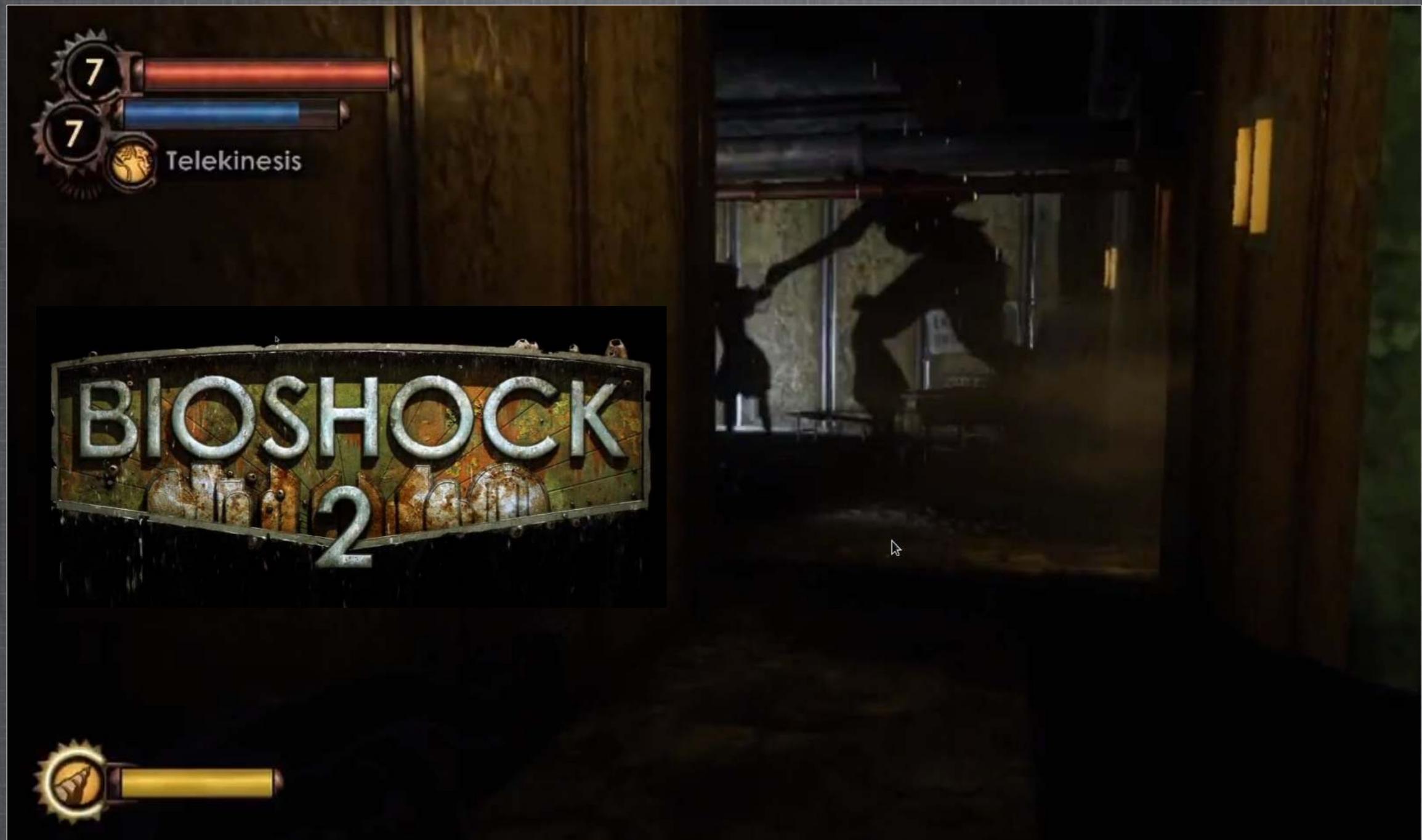
COMPUTER GRAPHICS IN GAMES



DIGITALFOUNDRY 2017
(KINGDOM HEARTS III, SQUARE ENIX, 2019)



COMPUTER GRAPHICS IN GAMES



BIOSHOCK 2 (2K BOSTON, 2010)

COMPUTER GRAPHICS IN GAMES



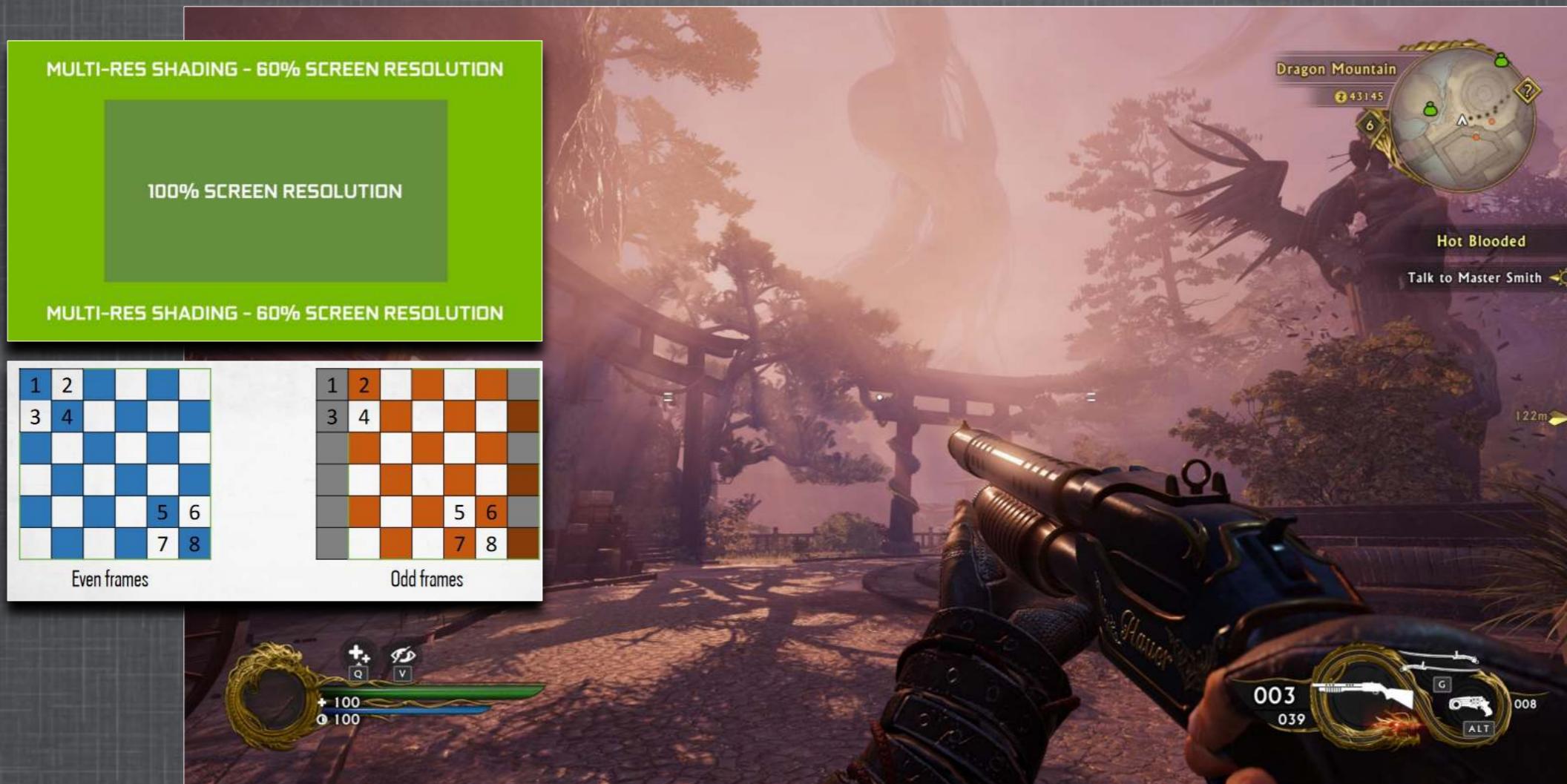
LAST REMNANT, 2018 REMASTER

COMPUTER GRAPHICS IN GAMES



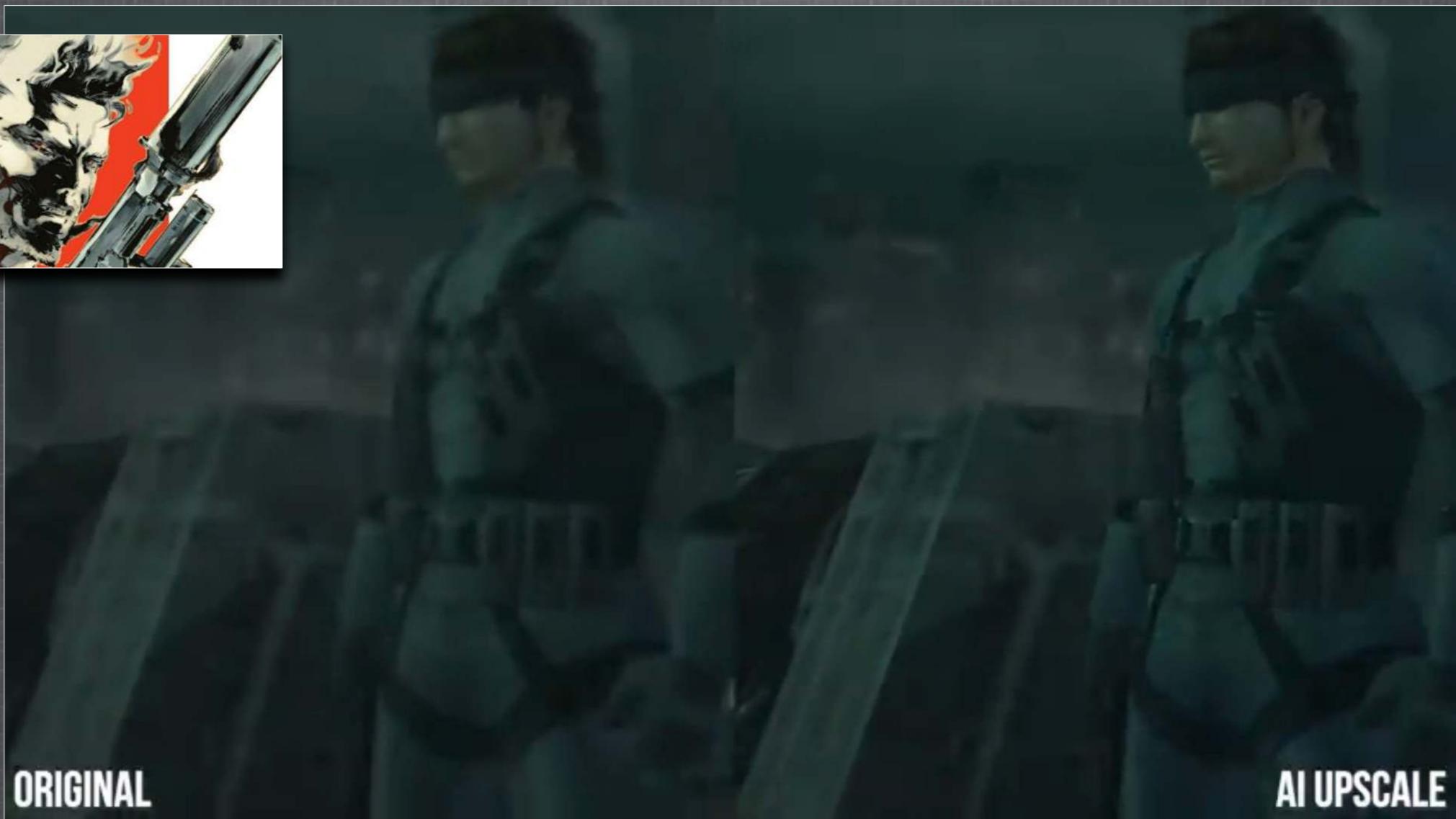
DIGITAL FOUNDRY, 2019
BORDERLANDS 2 AND 3

COMPUTER GRAPHICS IN GAMES



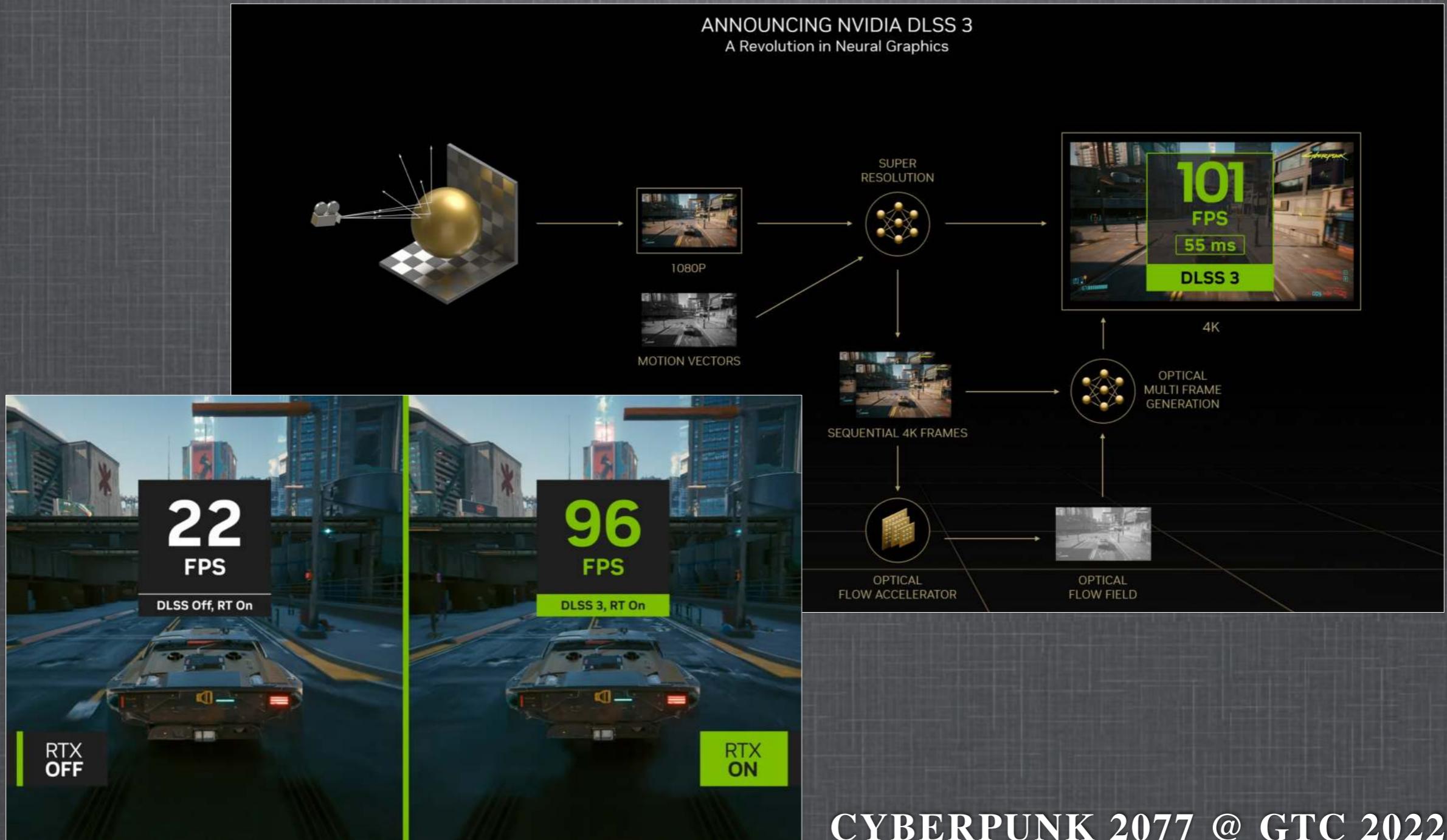
SHADOW WARRIOR 2
MULTI-RESOLUTION MULTI-FRAME SHADING

COMPUTER GRAPHICS IN GAMES



METAL GEAR SOLID 2 – 2021 UPSCALED TRAILER

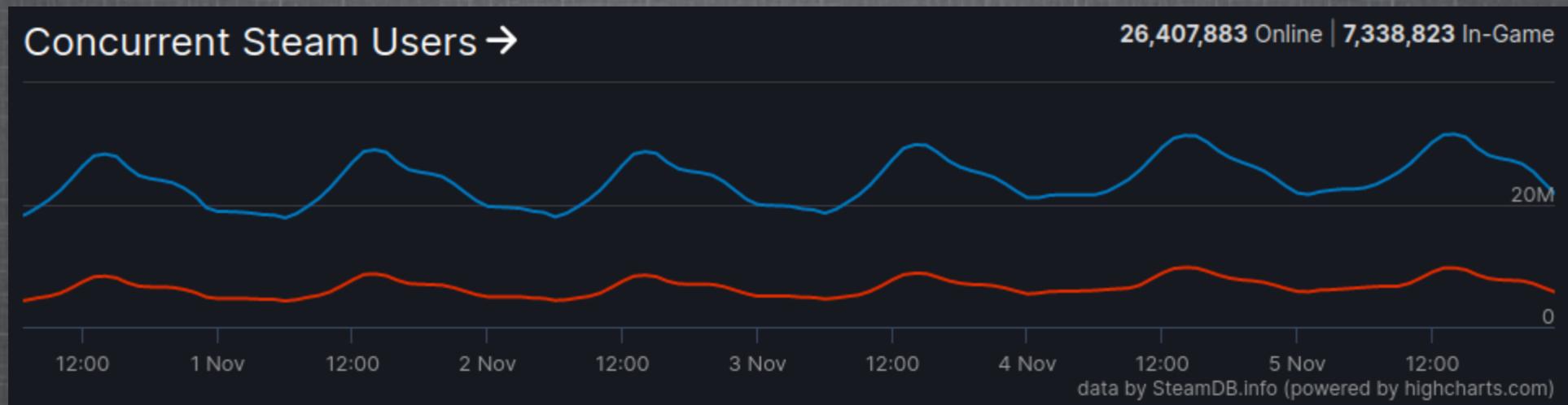
COMPUTER GRAPHICS IN GAMES



COMPUTER GRAPHICS IN GAMES

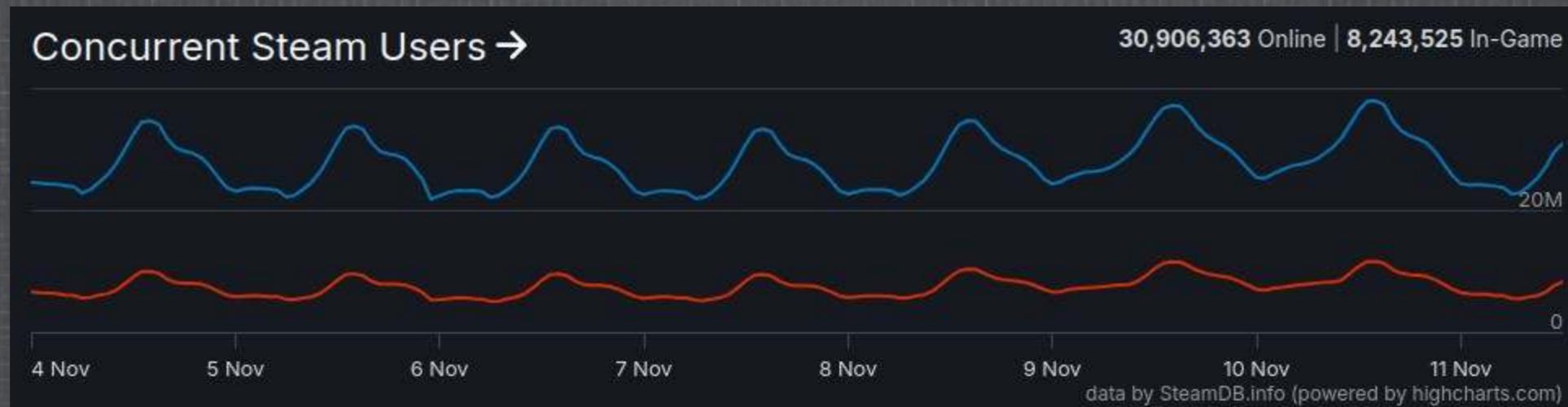
Online Gaming

6 nov 2023 (Steam, 120M+ monthly active users, 62.6M daily active users)



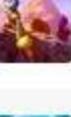
revenue of US \$8.56 billion for 2024

11 nov 2024 (Steam, 132M+ monthly active users, 69M+ daily active users)



projected revenue of US \$8.58 billion for 2024

COMPUTER GRAPHICS IN GAMES

	Game title	Change	Publisher
1.	 Counter-Strike 2 & GO	-	Valve
2.	 Minecraft	-	Mojang Studios
3.	 ROBLOX	-	Roblox
4.	 Fortnite	↑ 1	Epic Games
5.	 League of Legends	↑ 2	Riot Games
6.	 Dota 2	↑ 3	Valve
7.	 The Sims 4	↑ 1	Electronic Arts
8.	 VALORANT	↑ 2	Riot Games
9.	 PEAK (Aggro Crab)	↓ 3	Aggro Crab Games
10.	 HELLDIVERS 2	↑ 4	Sony Interactive Entertainment

Most popular PC games globally by Monthly Active Users
(Newzoo.com, September 2025)

DIGITAL MEDIA

DIGITAL MEDIA

From *text* to *hypertext*, *image* and *video*

Manchester Mark I



1949

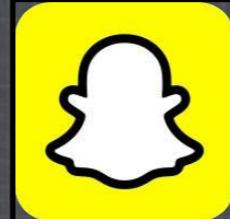
Fiction (1975-77)



Reality (1999)



2020



DIGITAL MEDIA

Digital Twins

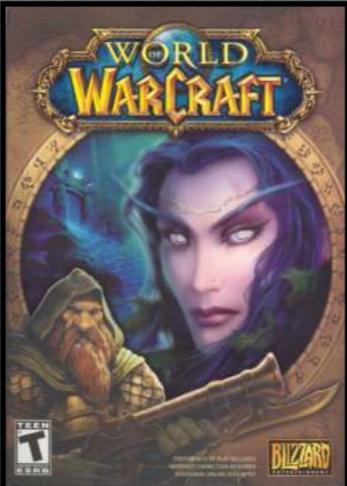
Google Earth and Google Maps



Home Simulation



Social Spaces



Second Life



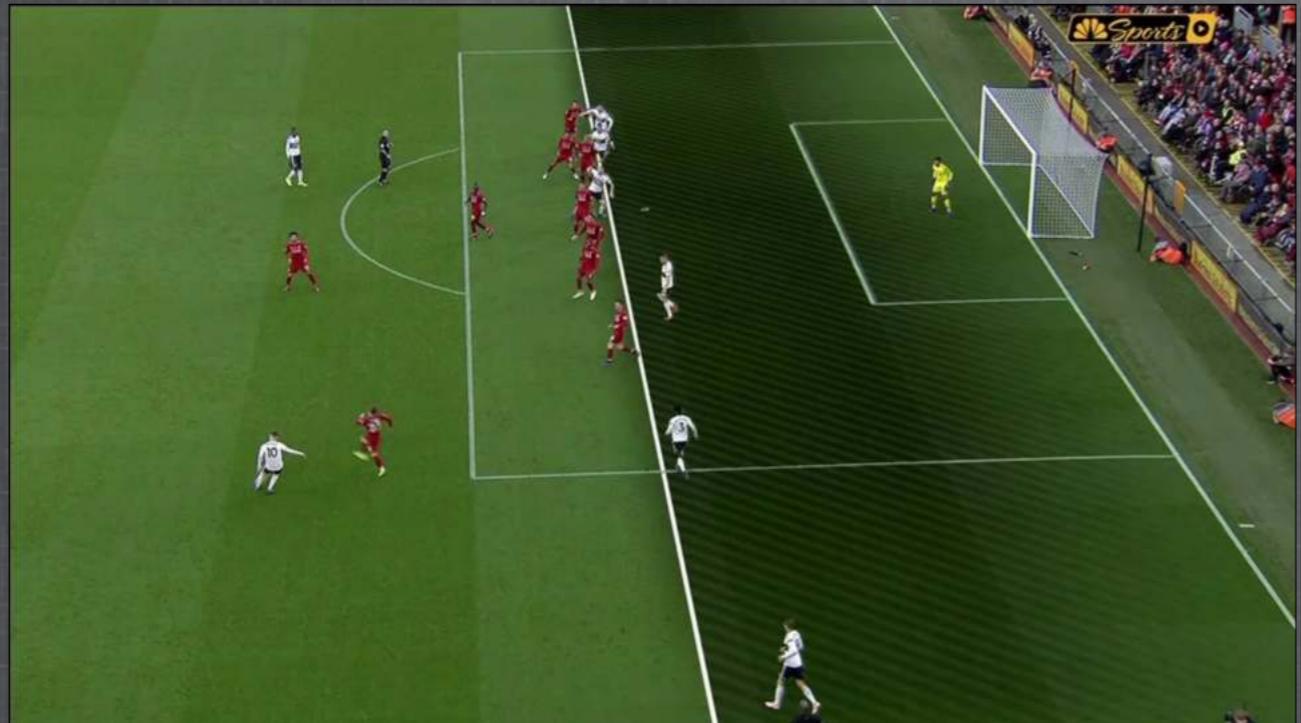
Electronic Publishing



DIGITAL MEDIA

Processing Feelings

Information overlay



Which one is the original? Are they real?



DIGITAL MEDIA

How real is our memory of a capture?



Apple X s MAX launch (sep 2018)



DIGITAL MEDIA

How real is our memory of a capture?



Lens Flare

DIGITAL MEDIA

Future of Media



COMPUTER AIDED DESIGN AND VISUALISATION

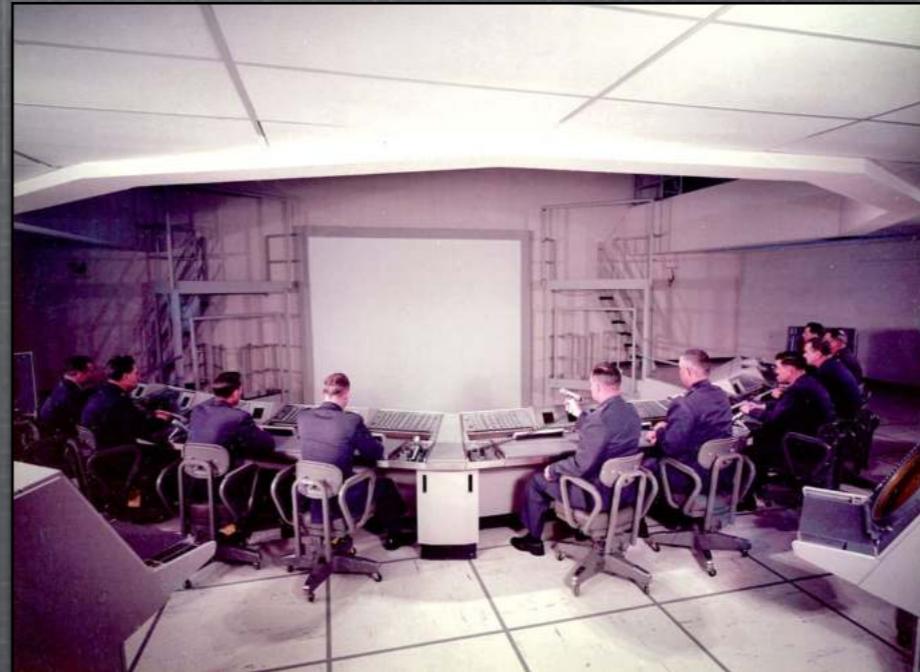


Unreal Engine 4 - Unreal Paris (January 2015)

BRIEF HISTORY OF COMPUTER GRAPHICS

COMPUTER GRAPHICS

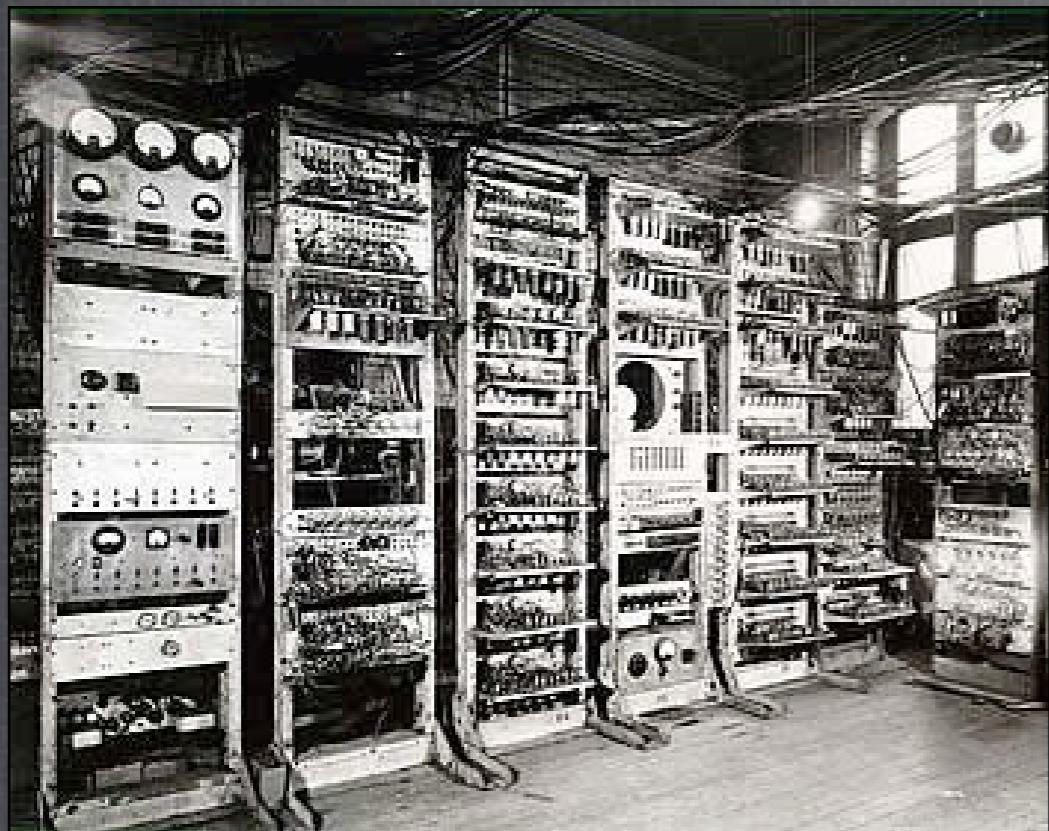
TERM COINED BY WILLIAM FETTER (BOEING) IN 1960



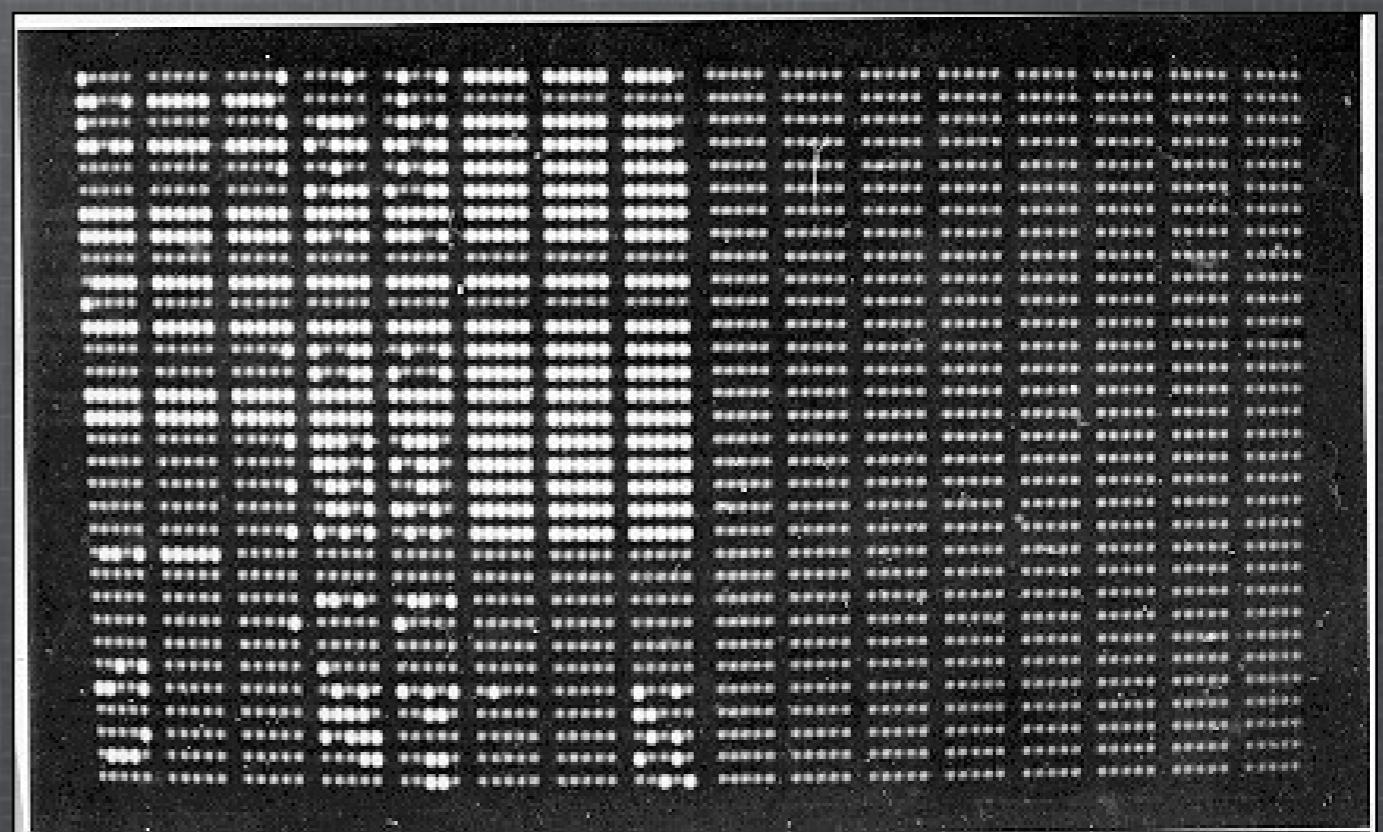
1ST GRAPHICS SYSTEM
USAF SAGE RADAR DATA (MIT) IN MID 1950'S

2D COMPUTER GRAPHICS

1ST TEXT SYSTEMS

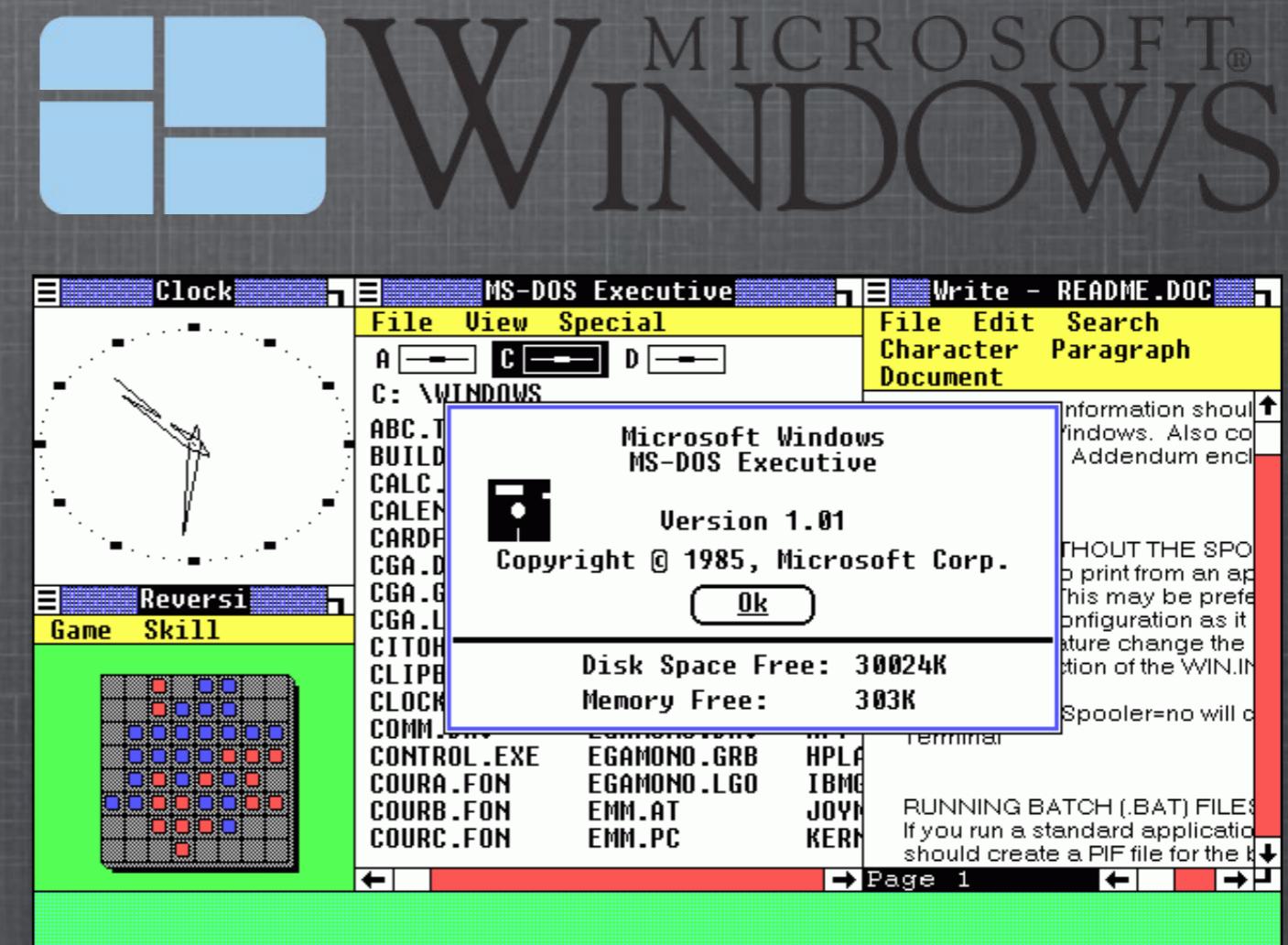
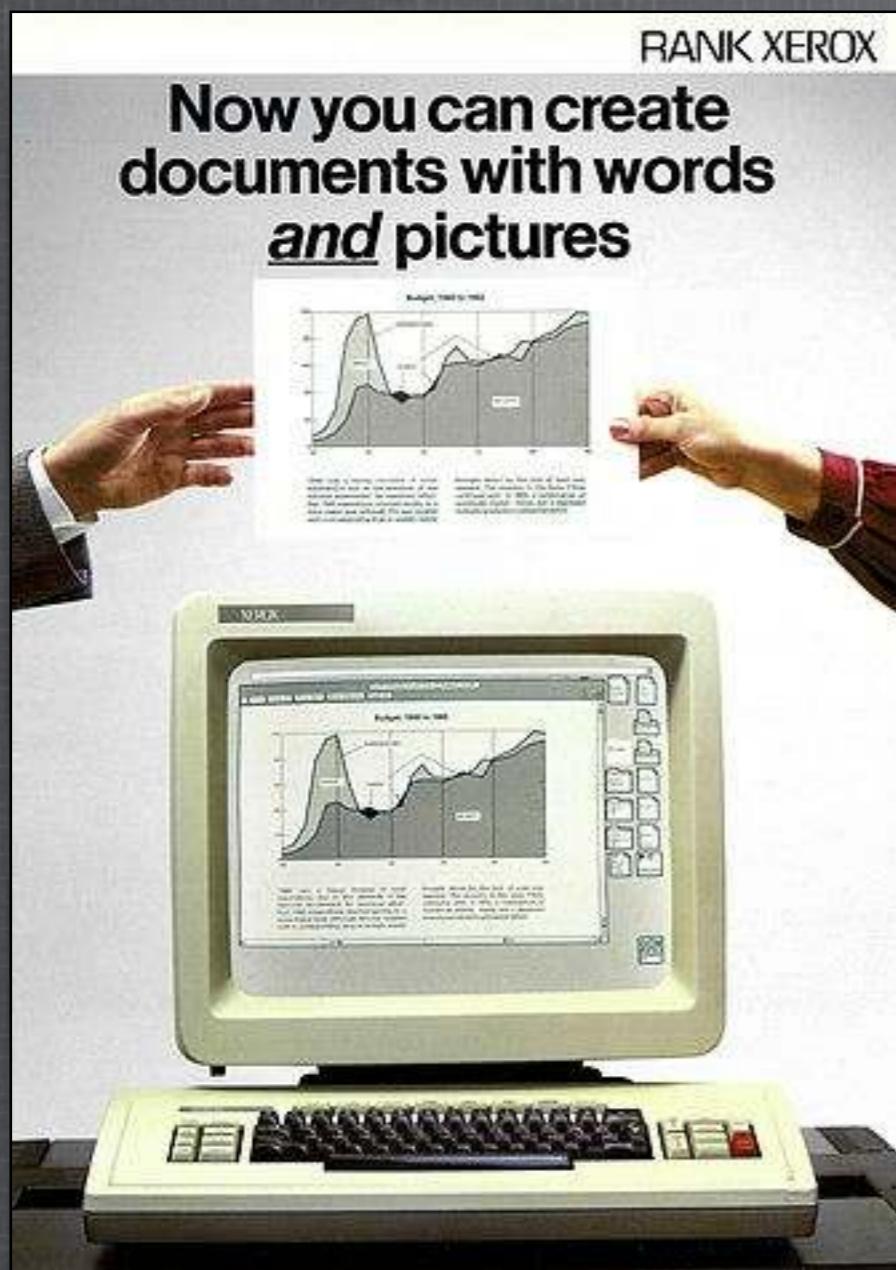


Manchester Mark I



Display

1ST GRAPHICAL USER INTERFACES



1ST DRAWING SYSTEMS

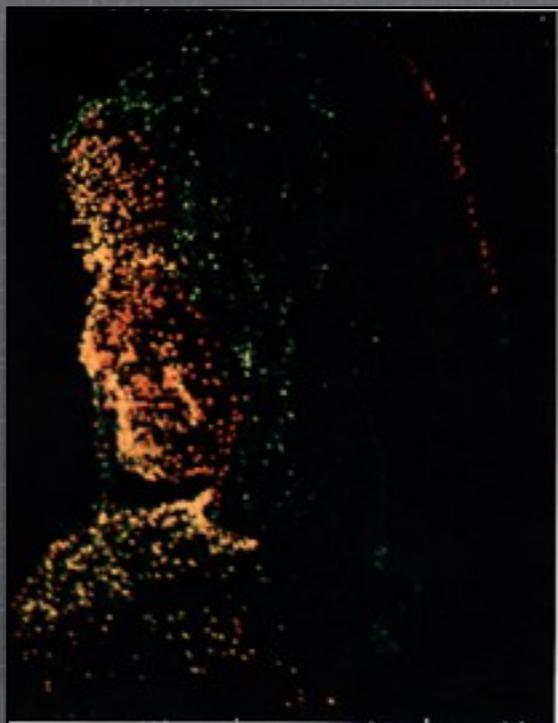
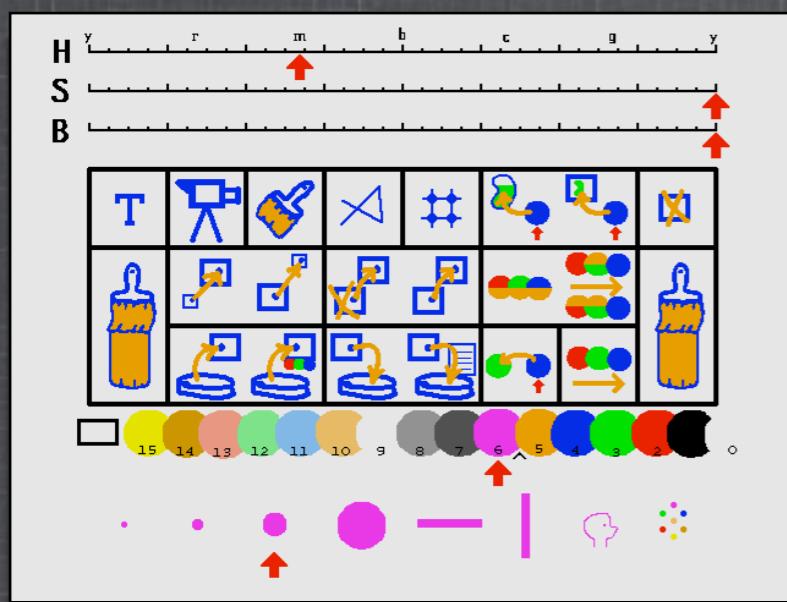


Sketchpad (Sutherland, MIT 1963)

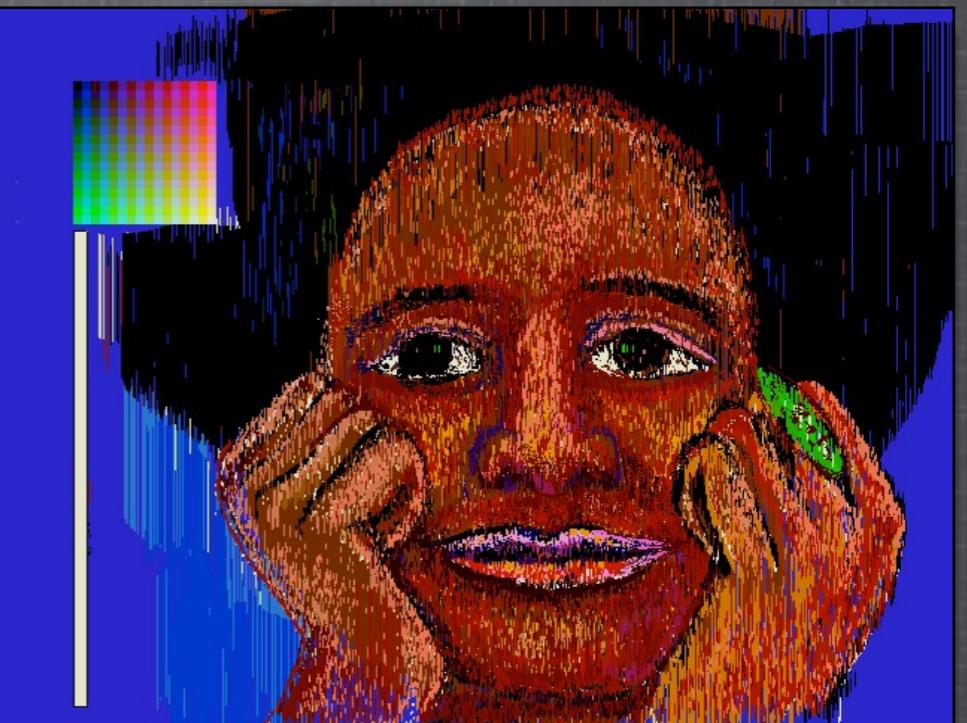
bioRxiv
science



1ST PAINT SYSTEMS

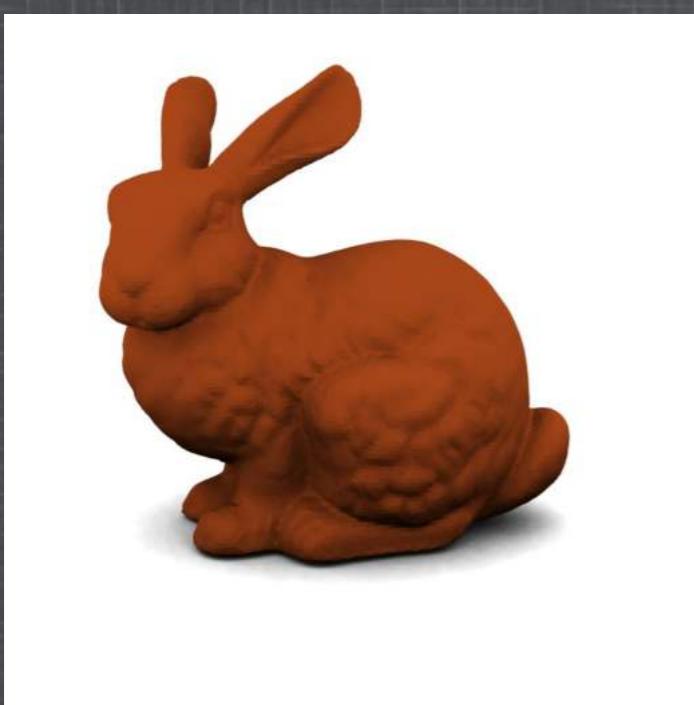
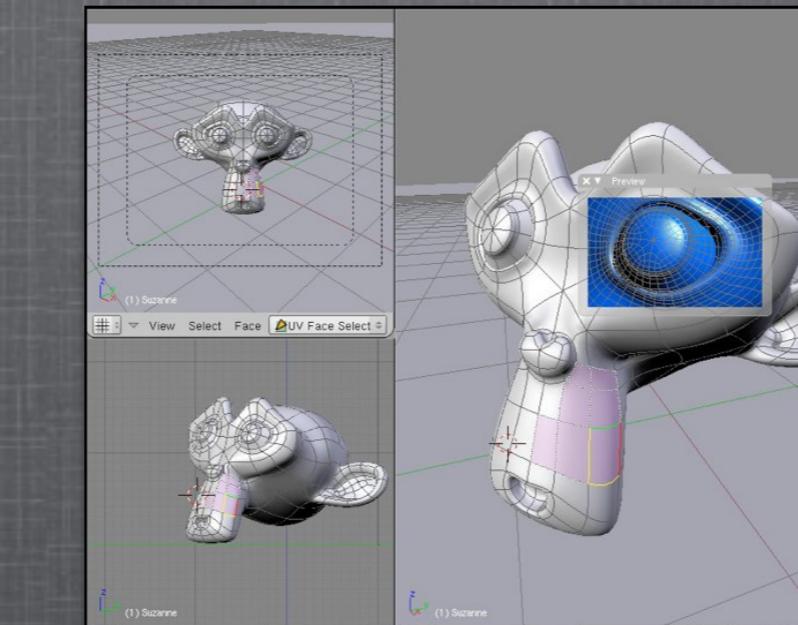


SuperPaint Framebuffer System (Shoup and Smith, 1973-1979)

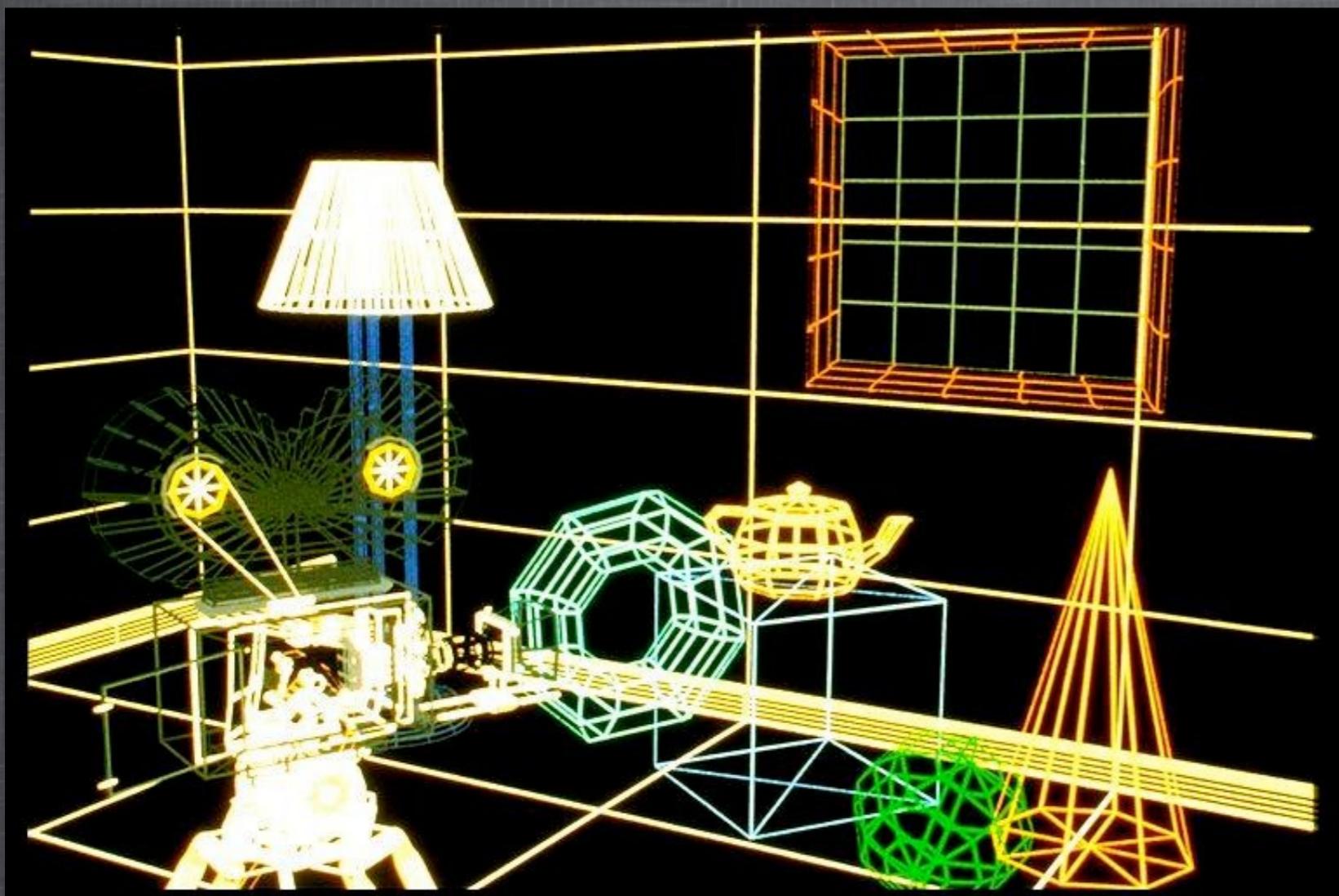


3D COMPUTER GRAPHICS

MODELING

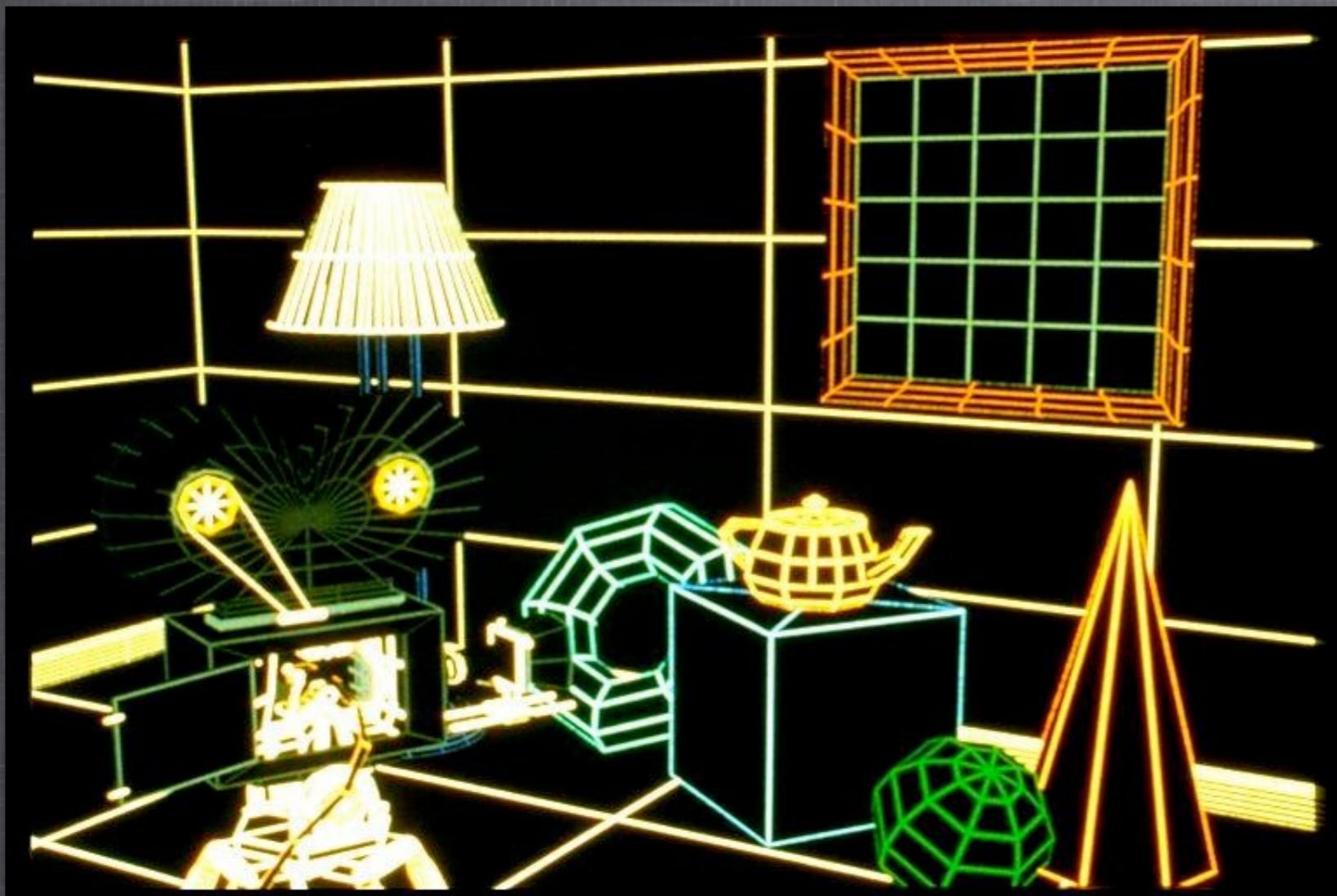


RENDERING



RENDERING

Visibility (1960's)



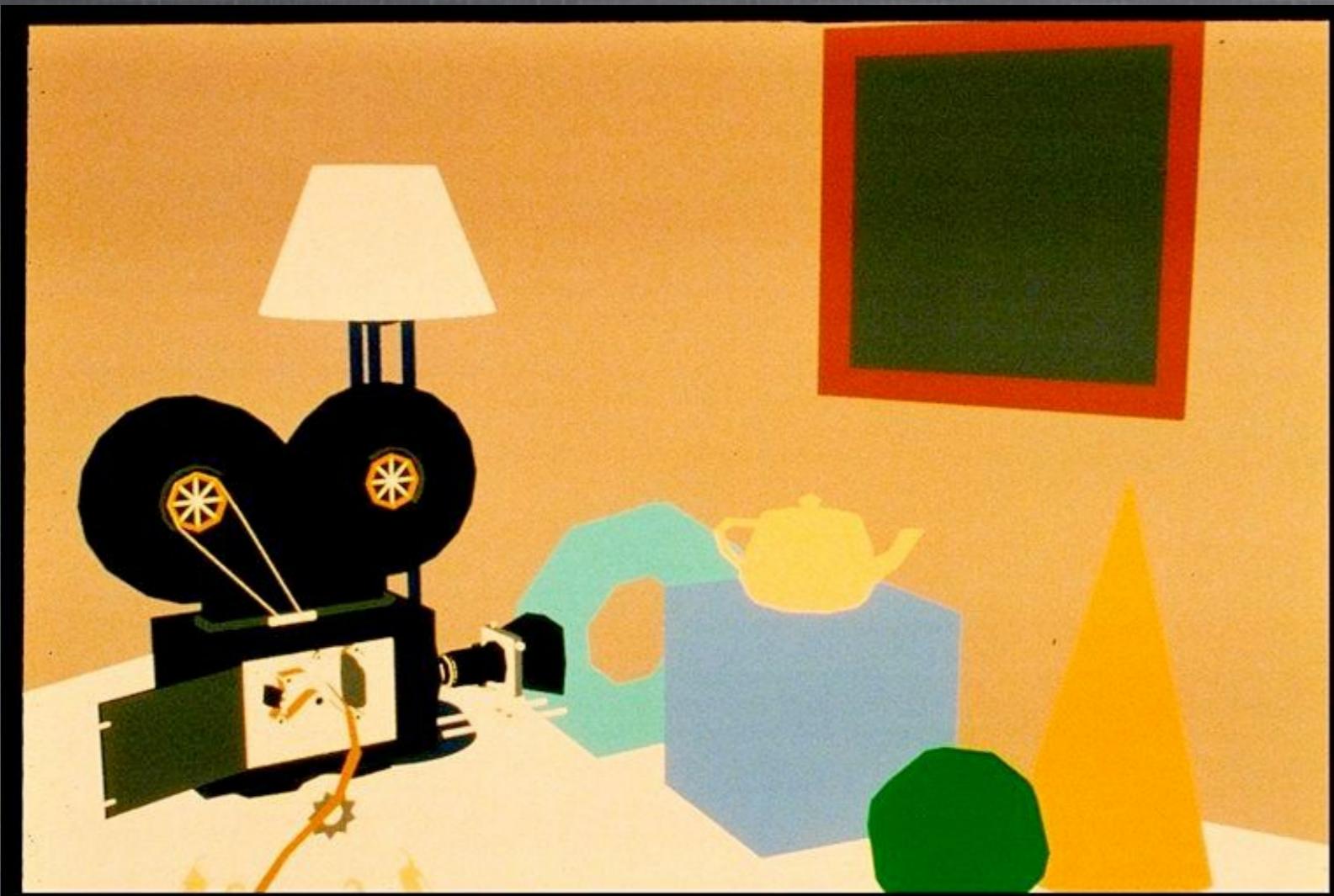
Hidden Line Algorithms:
Roberts (63), Appel (67)

Hidden Surface Algorithms:
Warnock (69), Watkins (70)

Visibility and Sorting:
Sutherland (74)

RENDERING

Visibility and Lighting (1970's)



Hidden Line Algorithms:
Roberts (63), Appel (67)

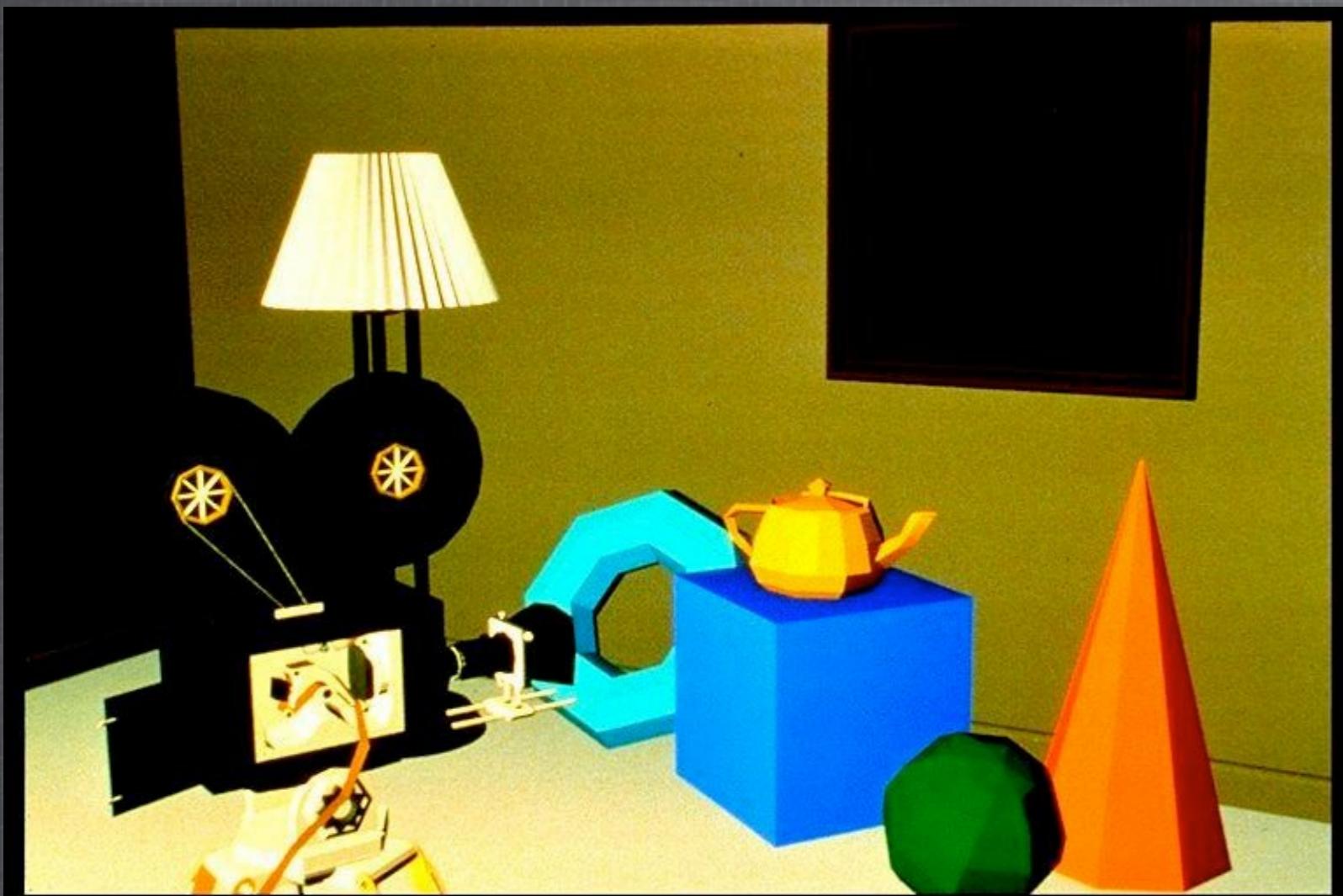
Hidden Surface Algorithms:
Warnock (69), Watkins (70)

Visibility and Sorting:
Sutherland (74)

Constant shading

RENDERING

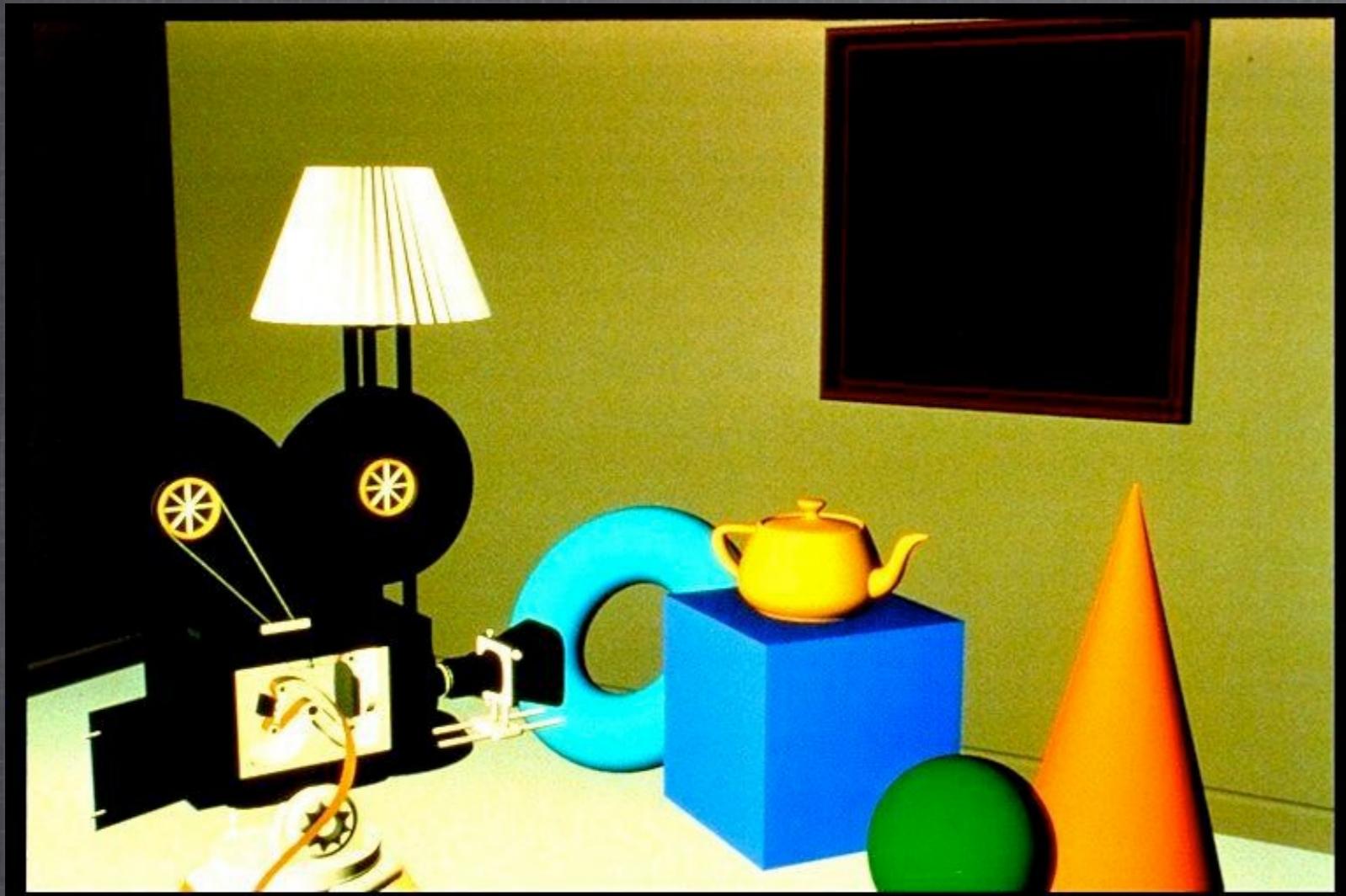
Lighting (1970's)



Flat shading

RENDERING

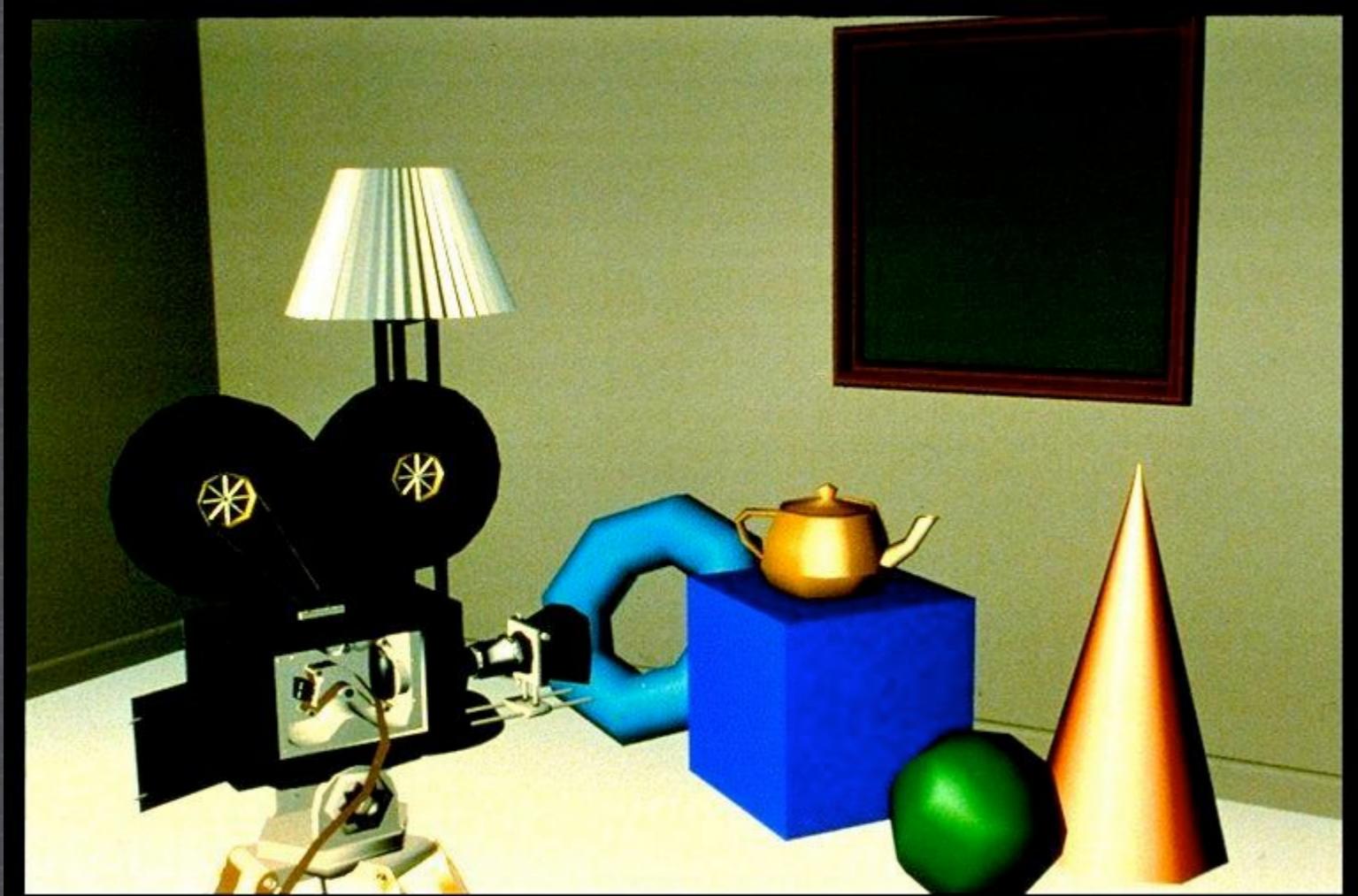
Lighting (1970's)



Diffuse lighting
(Gouraud 1971)

RENDERING

Lighting (1970's)



Diffuse lighting
(Gouraud 1971)

Specular lighting
(Phong 1974)

RENDERING

Lighting (1970's)



Diffuse lighting
(Gouraud 1971)

Specular lighting
(Phong 1974)

RENDERING

Lighting (1970's)



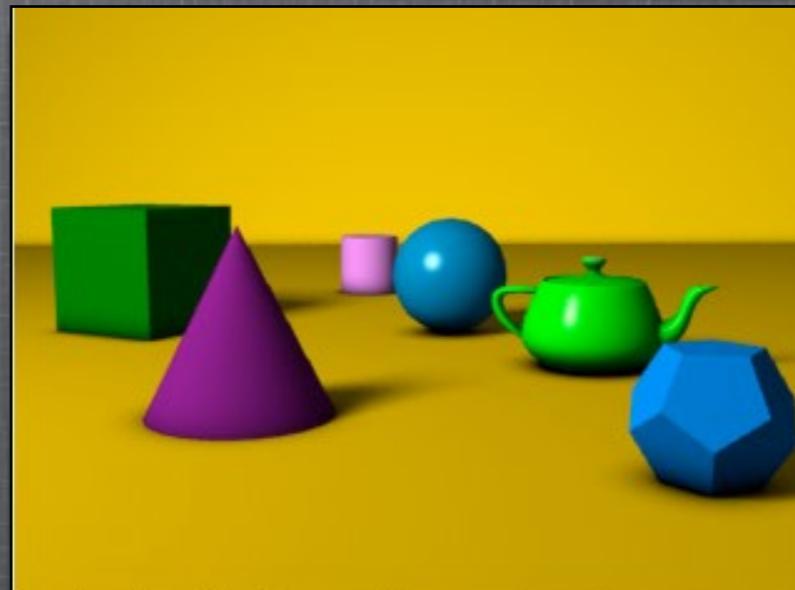
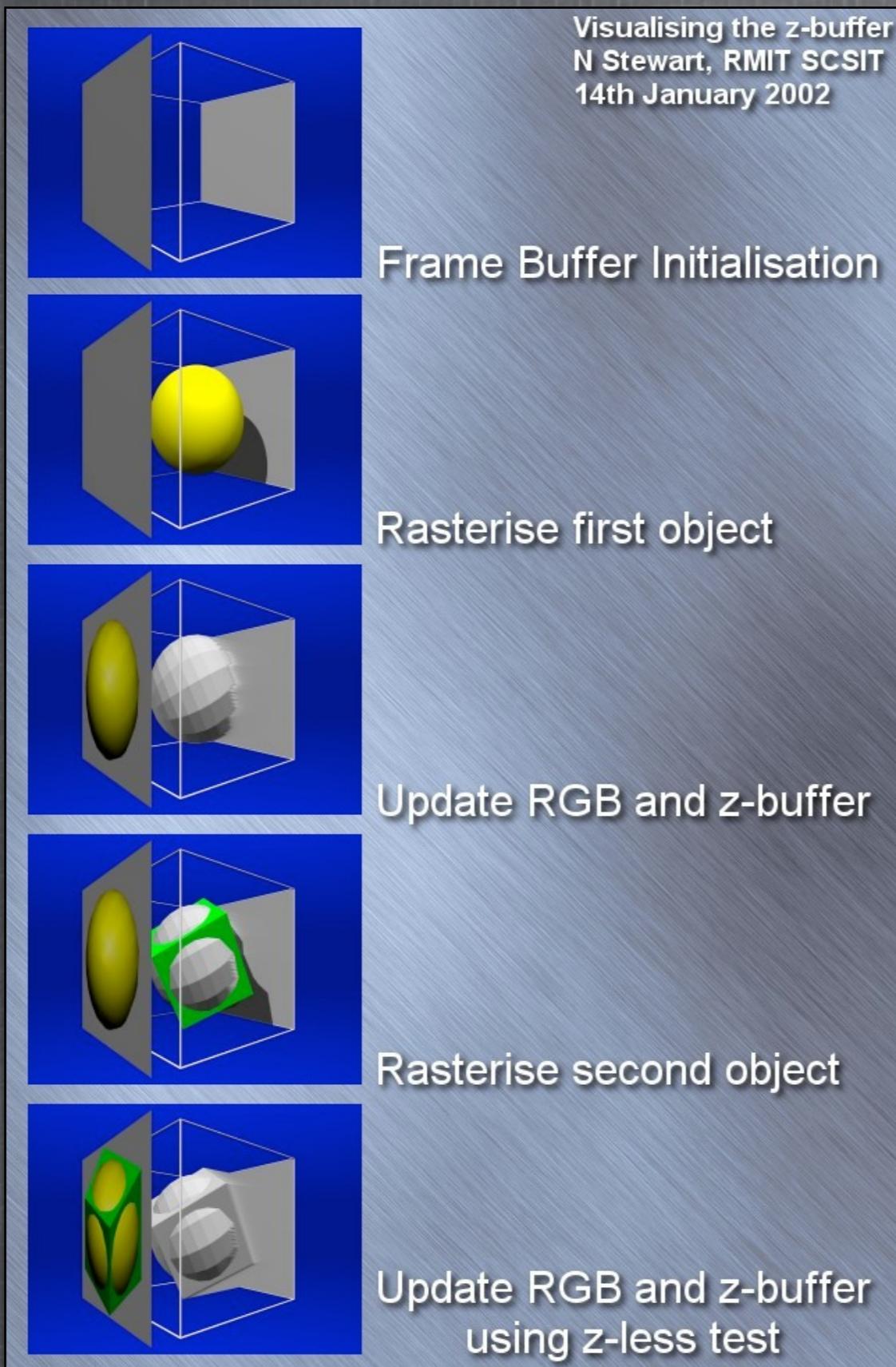
Diffuse lighting
(Gouraud 1971)

Specular lighting
(Phong 1974)

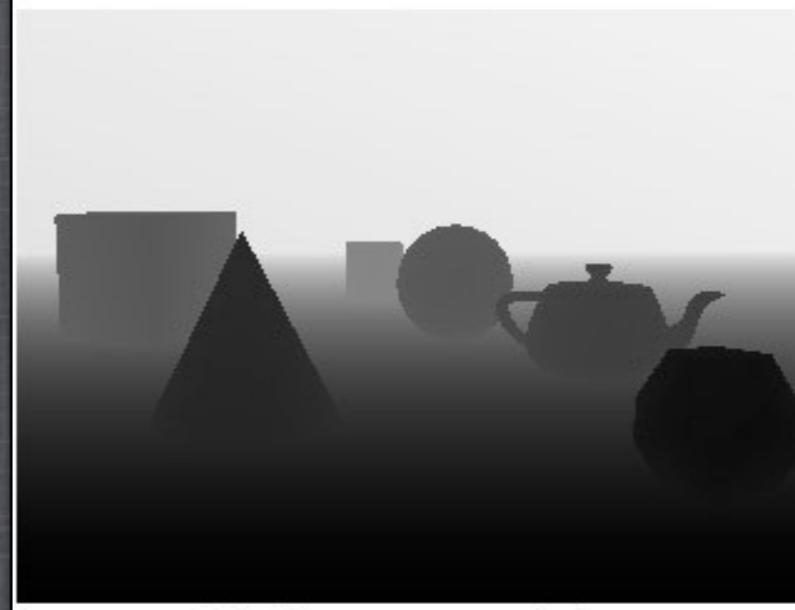
Curved Surfaces, Texture
(Blinn, 1974)

RENDERING

Z-buffer Hidden Surface
(Catmull, 1974)



A simple three dimensional scene



Z-buffer representation

RENDERING

Lighting (1970's)



Diffuse lighting
(Gouraud 1971)

Specular lighting
(Phong 1974)

Curved Surfaces, Texture
(Blinn, 1974)

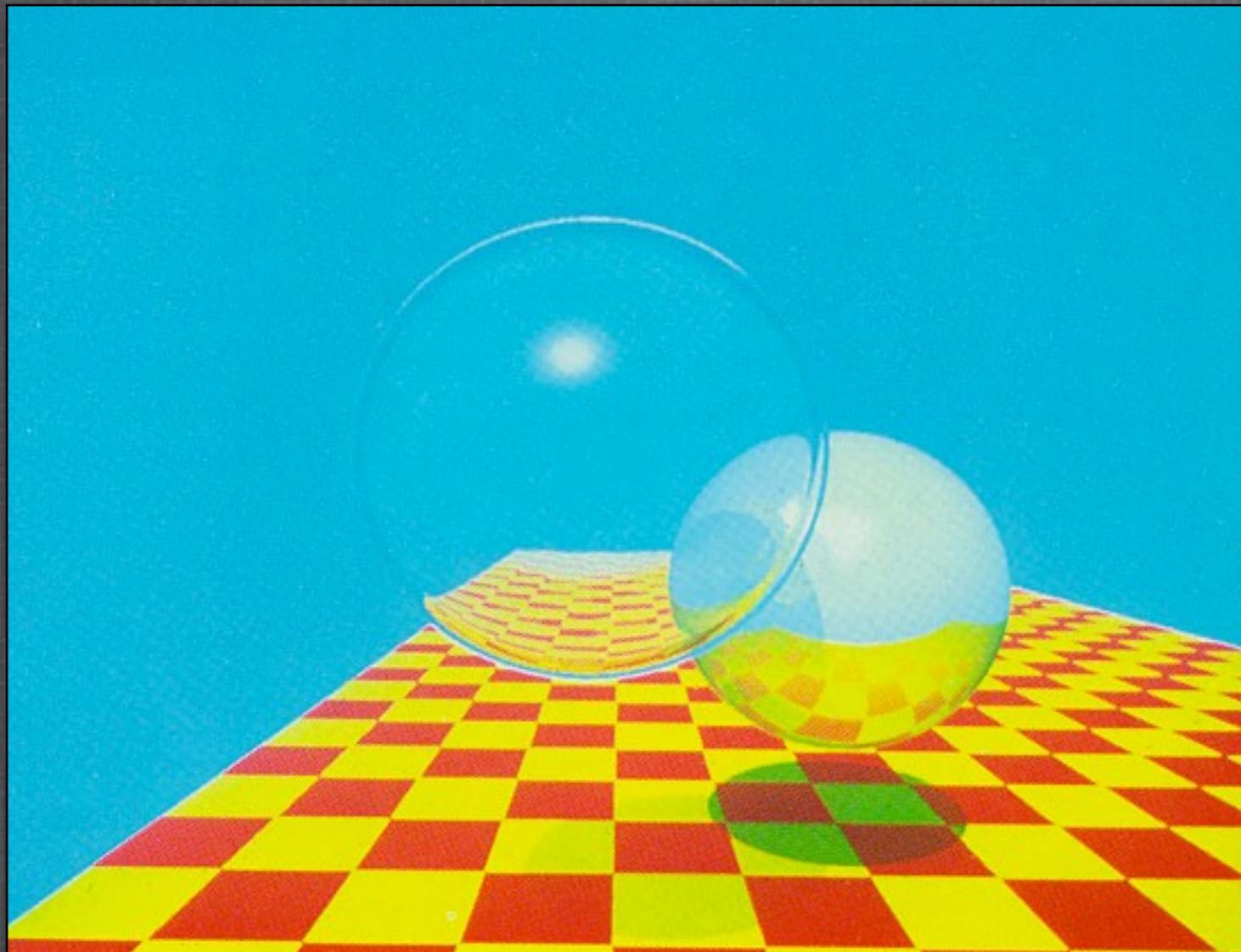
Antialias and Shadows
Crow (1977)

GLOBAL ILLUMINATION

(1980's and 1990's)

RENDERING

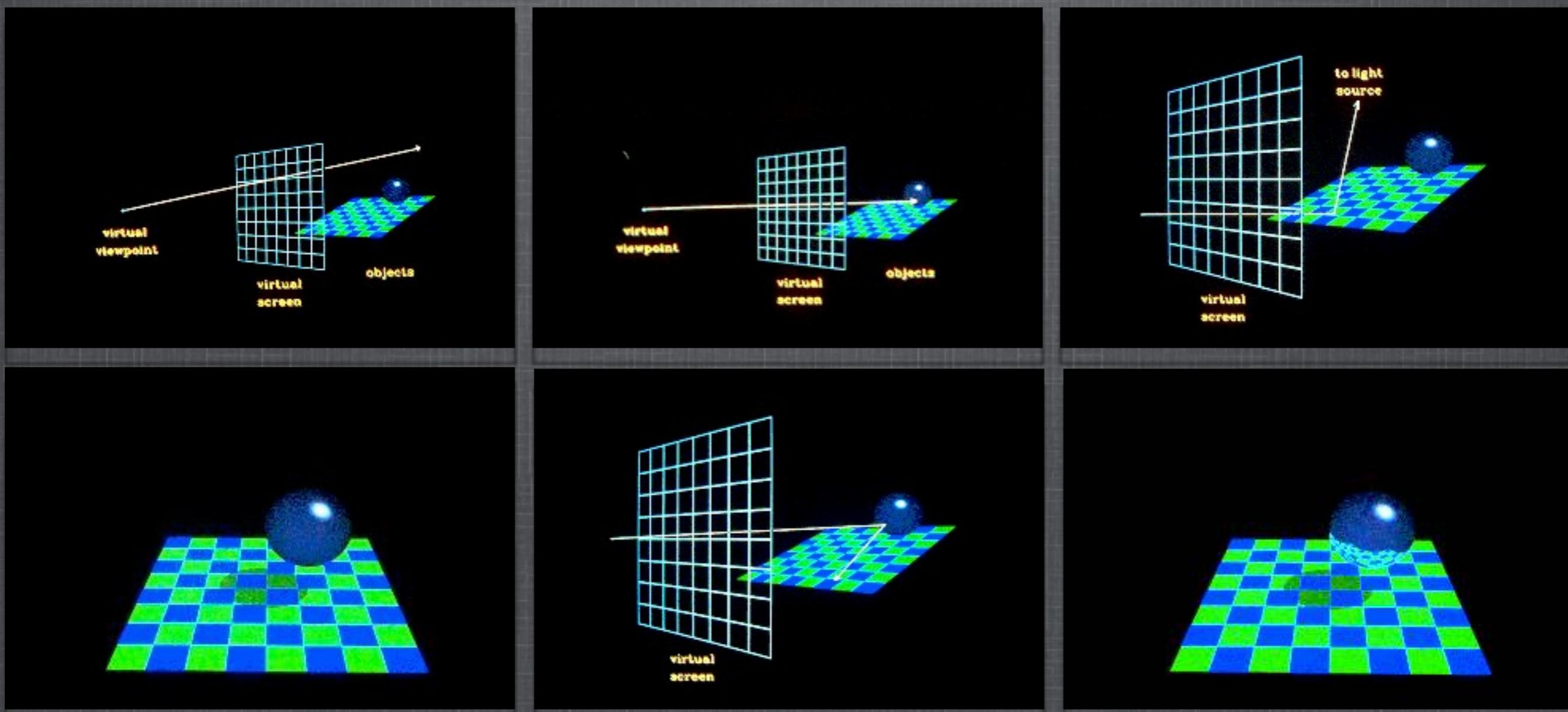
Global Illumination (1980's and 1990's)



Raytracing (recursive)
(Whitted 1980)

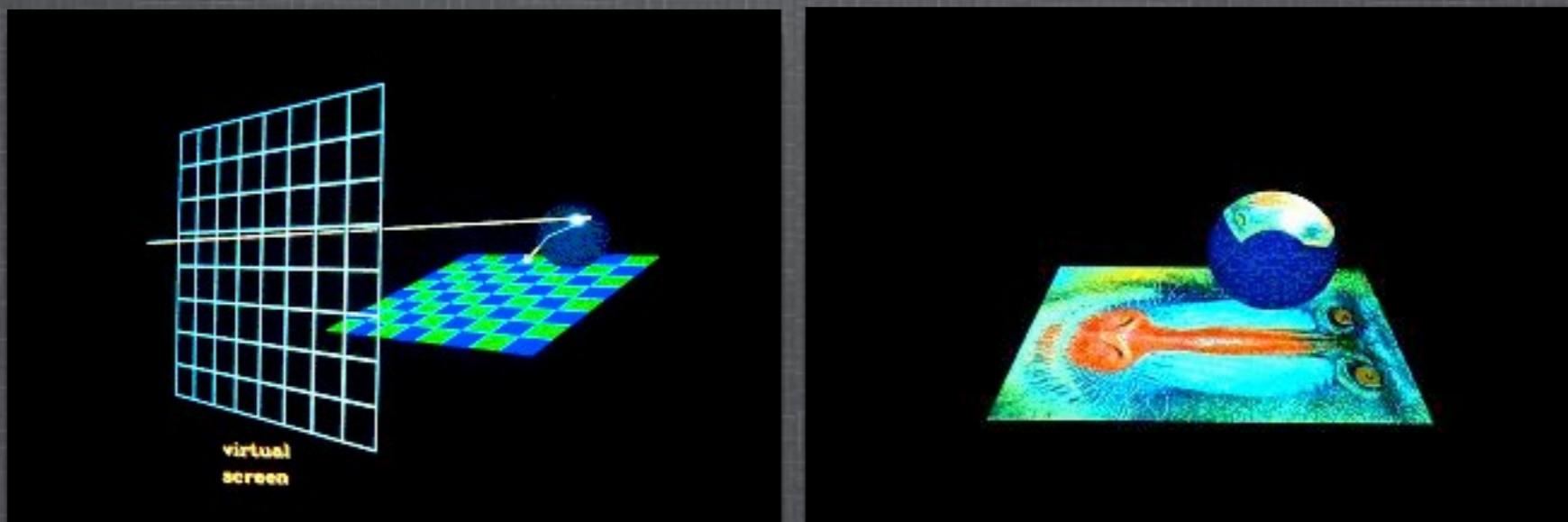
RENDERING

Raytracing (Whitted 1980)



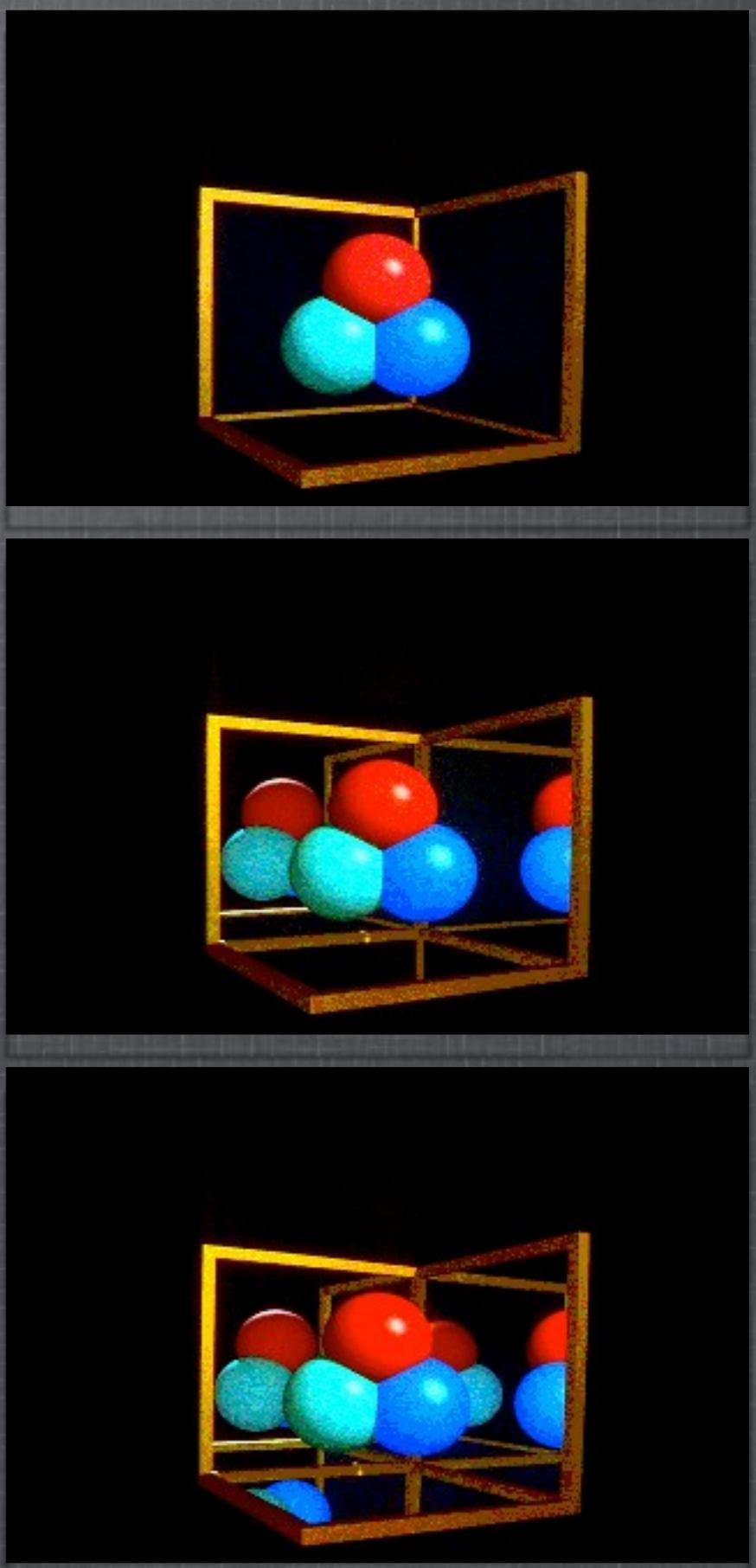
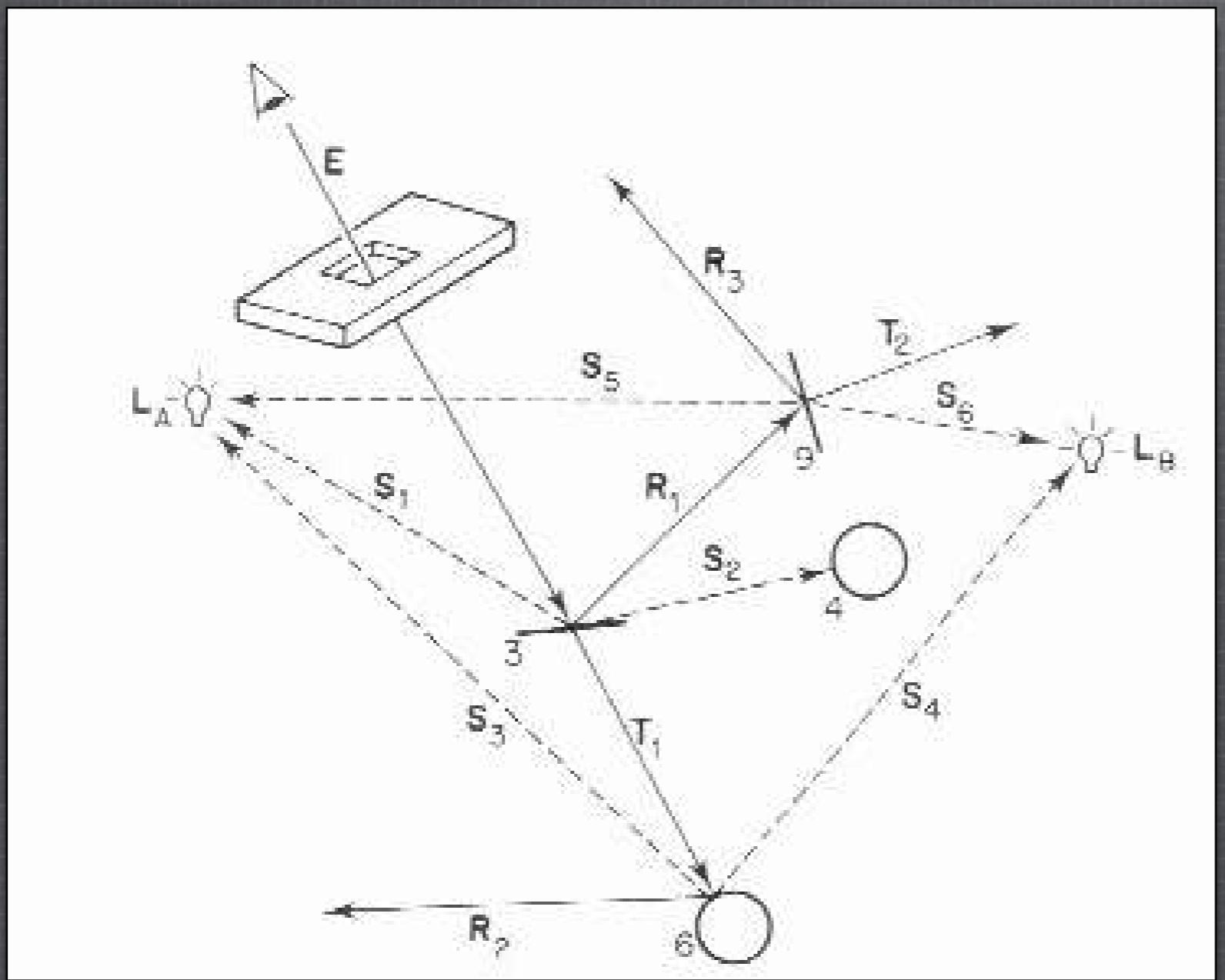
RENDERING

Raytracing (Whitted 1980)

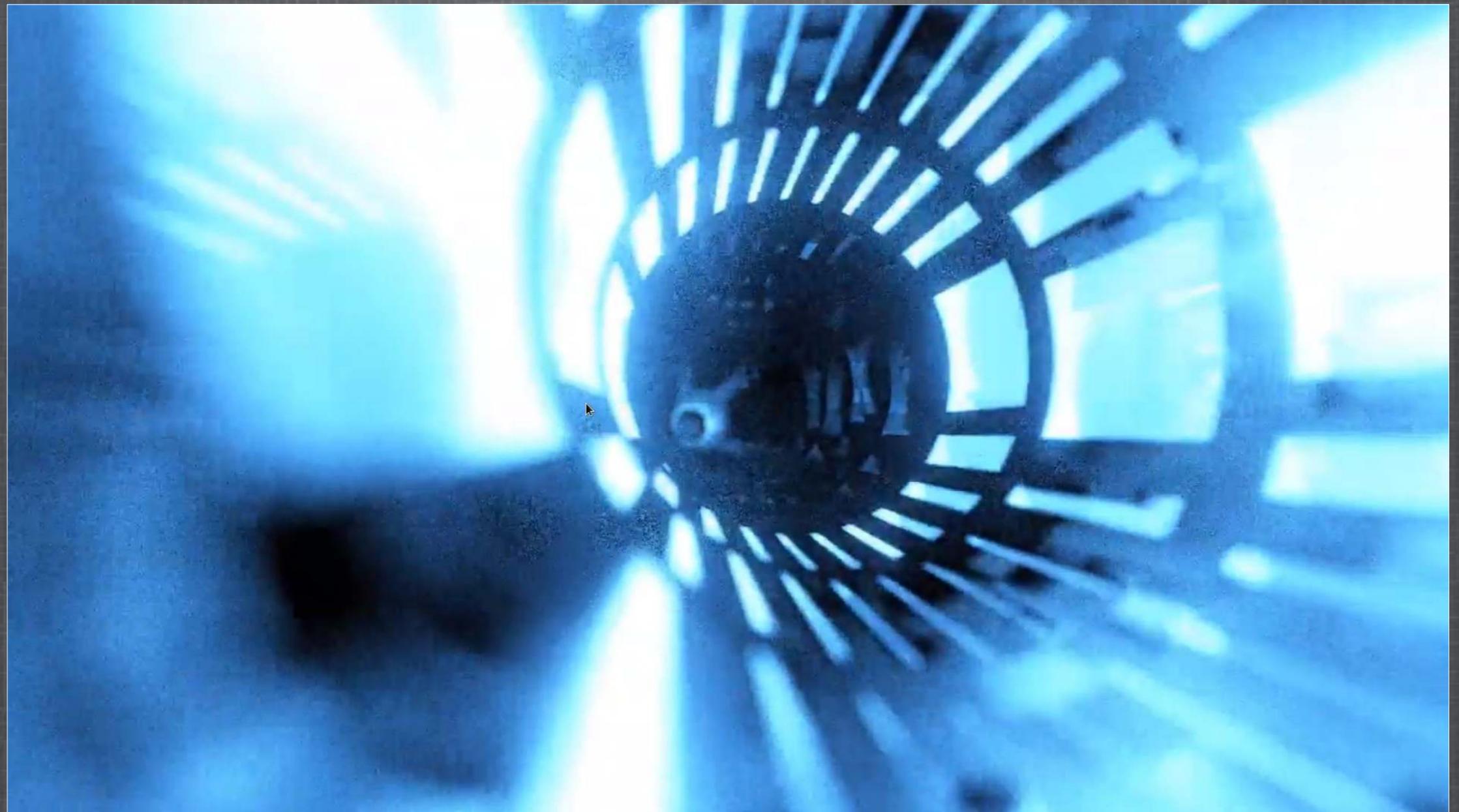


RENDERING

Raytracing (Whitted 1980)



DEMO SCENE



Sync Cord by NuSan & Valden (PC 4k intro, 2020)

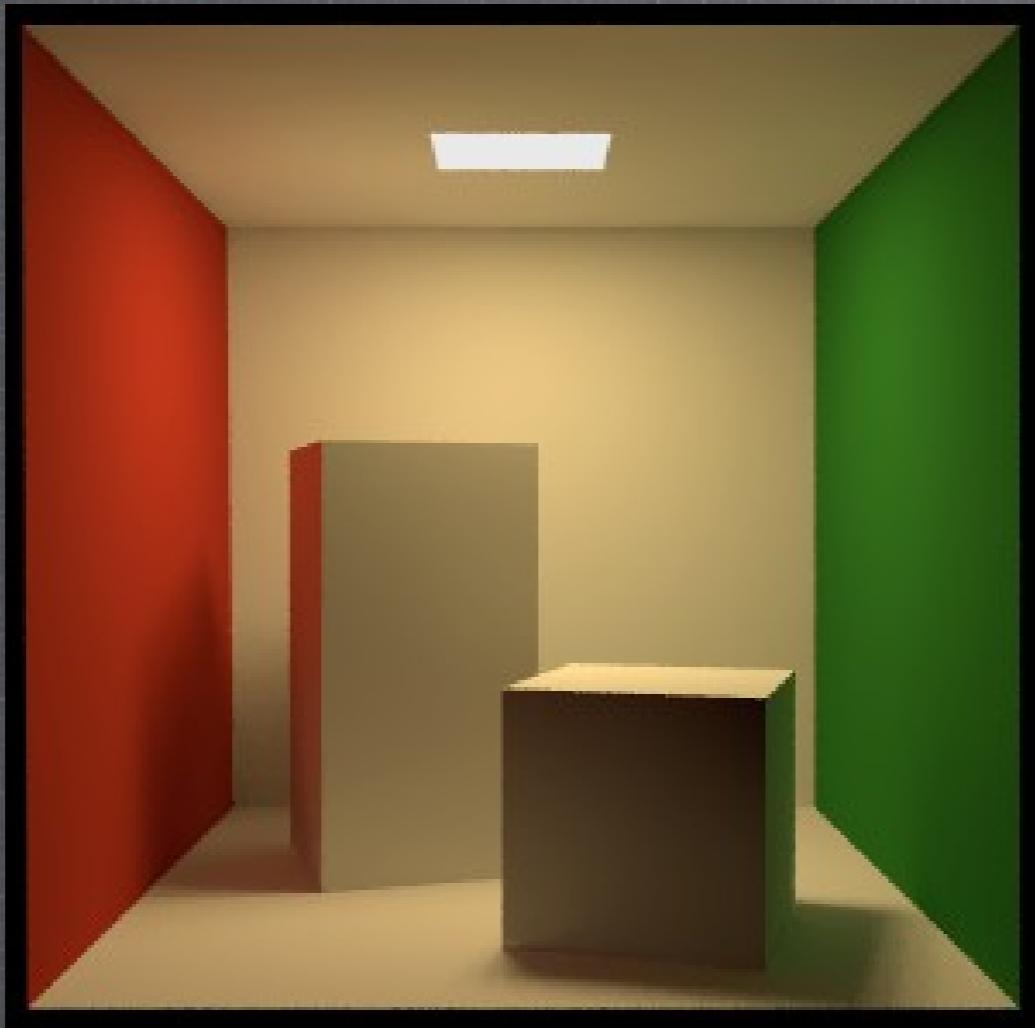
REFLECTIONS

Real-time Ray-Tracing in UE4 (2018)



RENDERING

Global Illumination (1980's and 1990's)

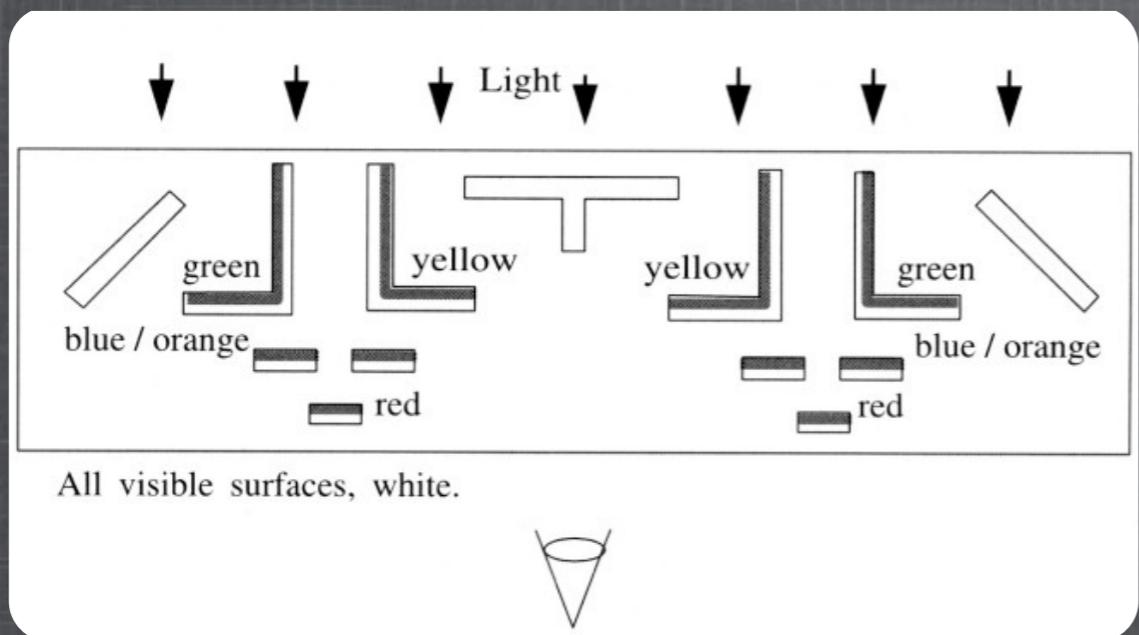


Raytracing
(Whitted 1980)

Radiosity
(Goral, Torrance et al. 1984)

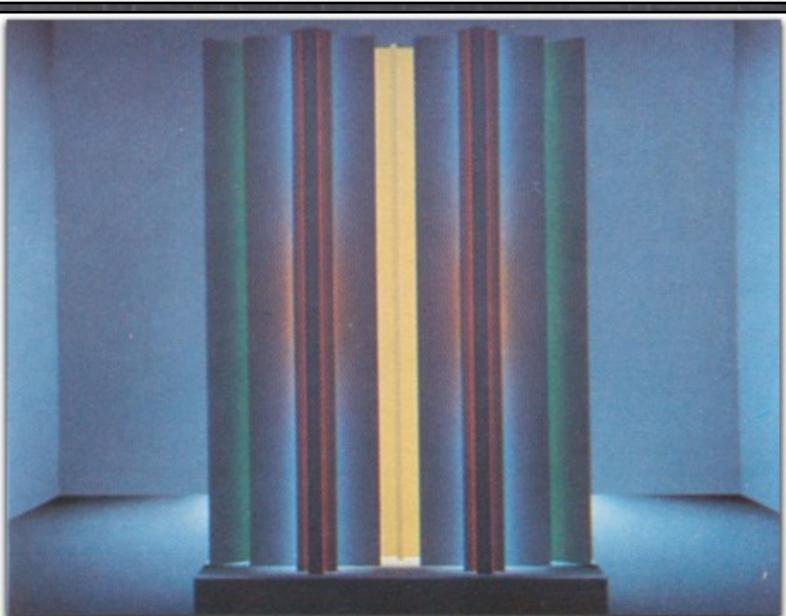
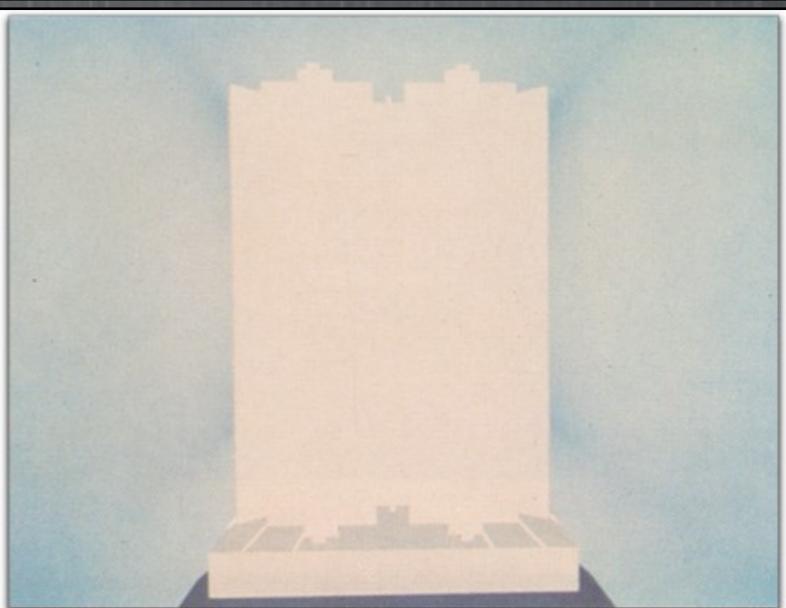
RENDERING

Radiosity (Goral, Torrance et al. 1984)



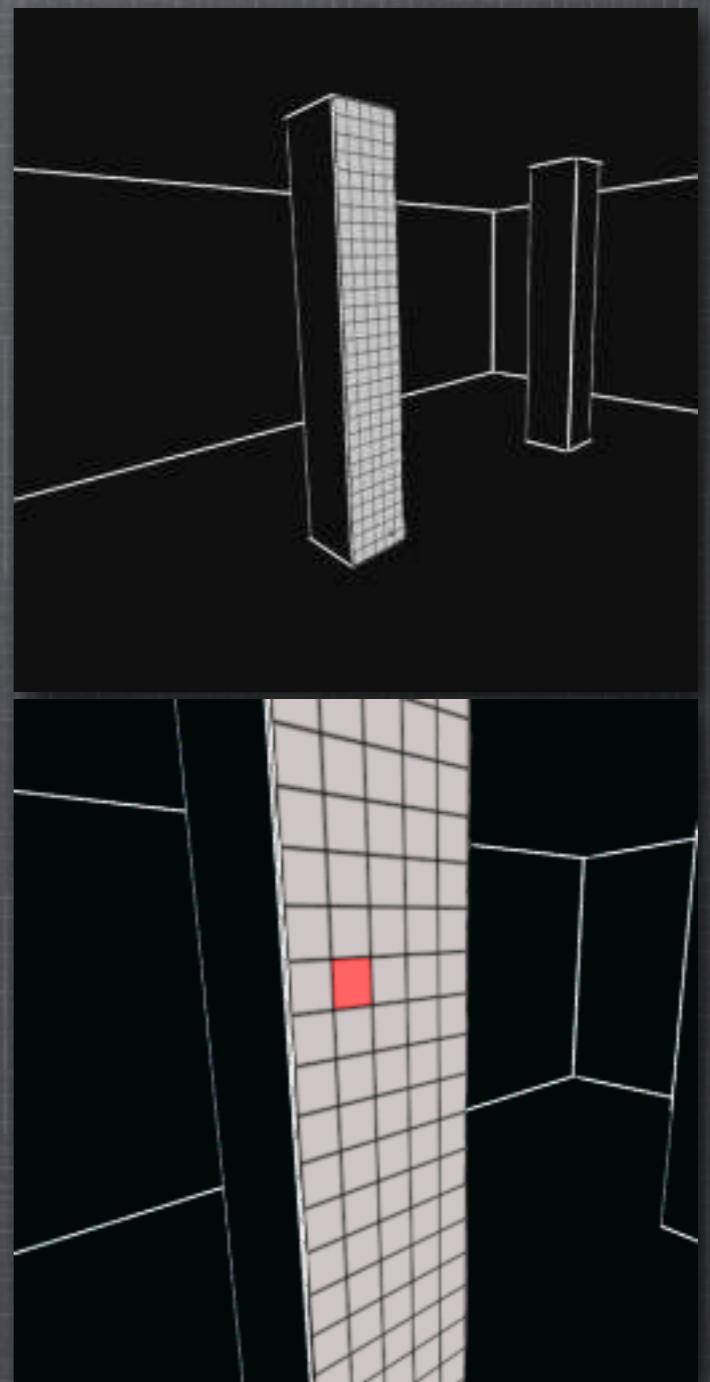
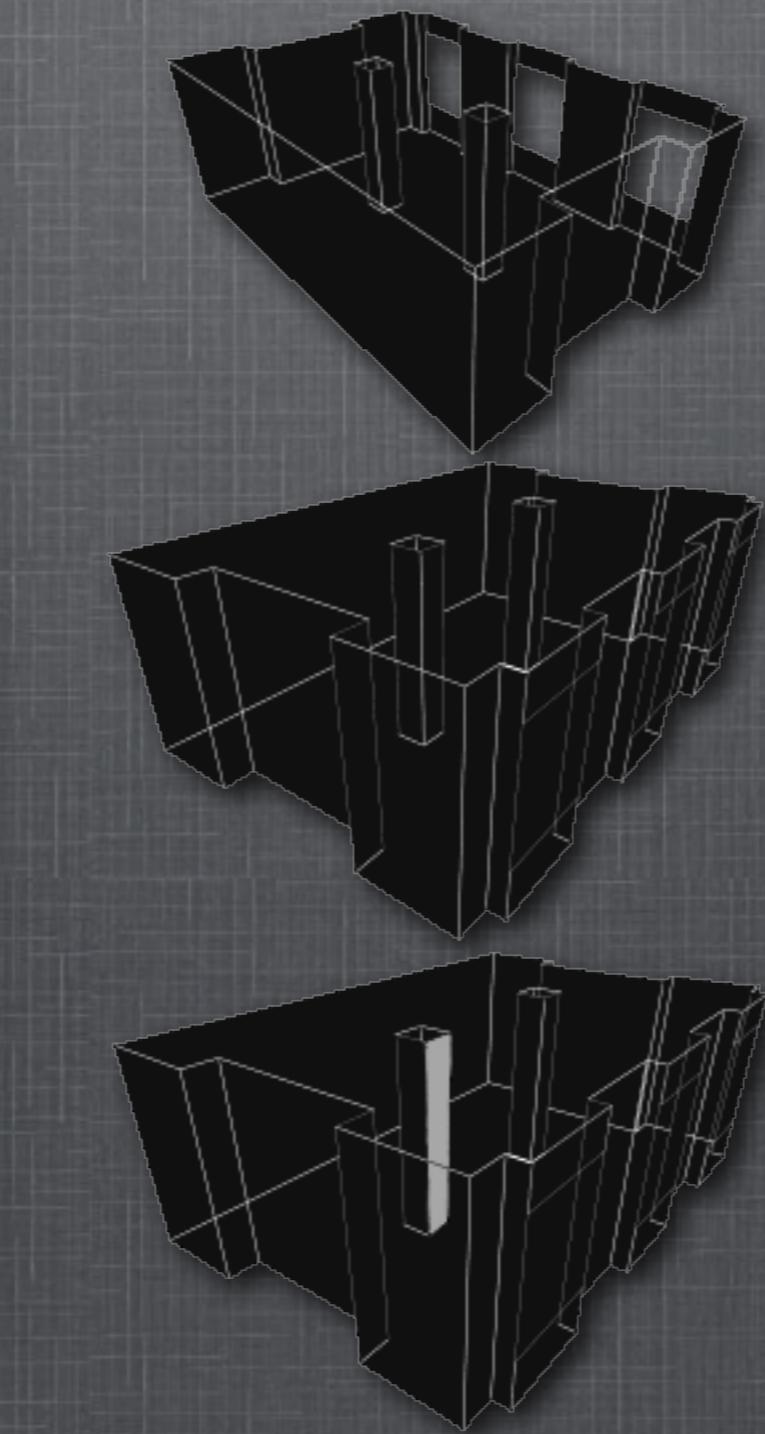
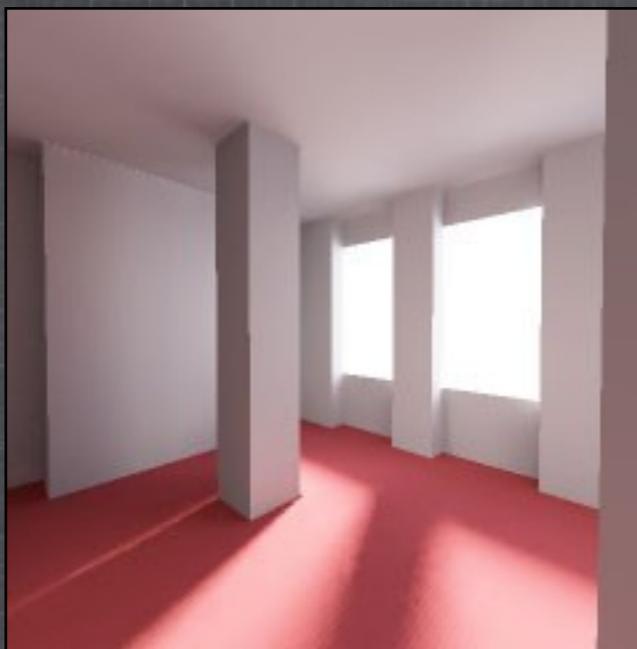
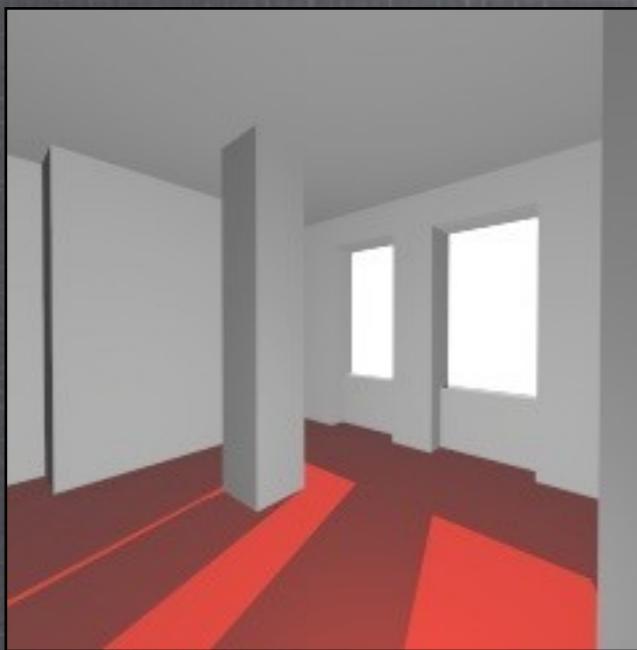
Sculpture by John Ferren

Goral et al. 1984



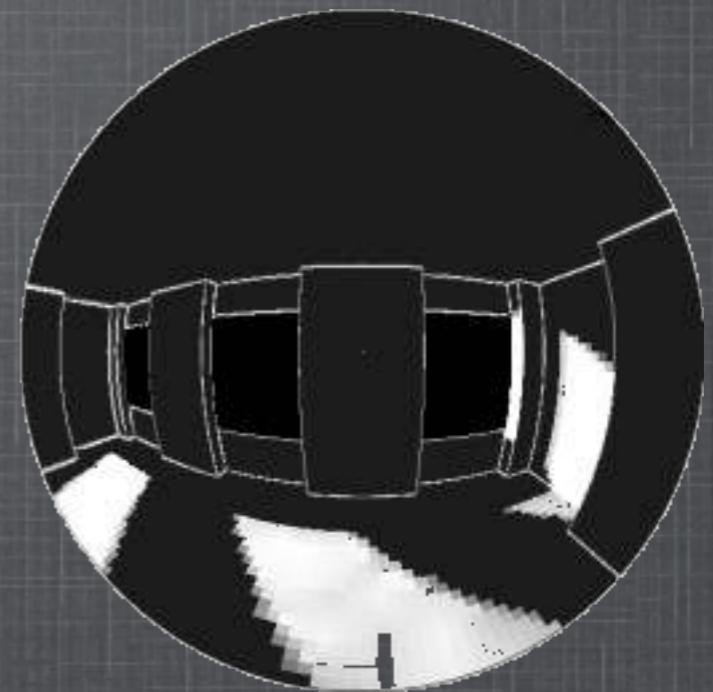
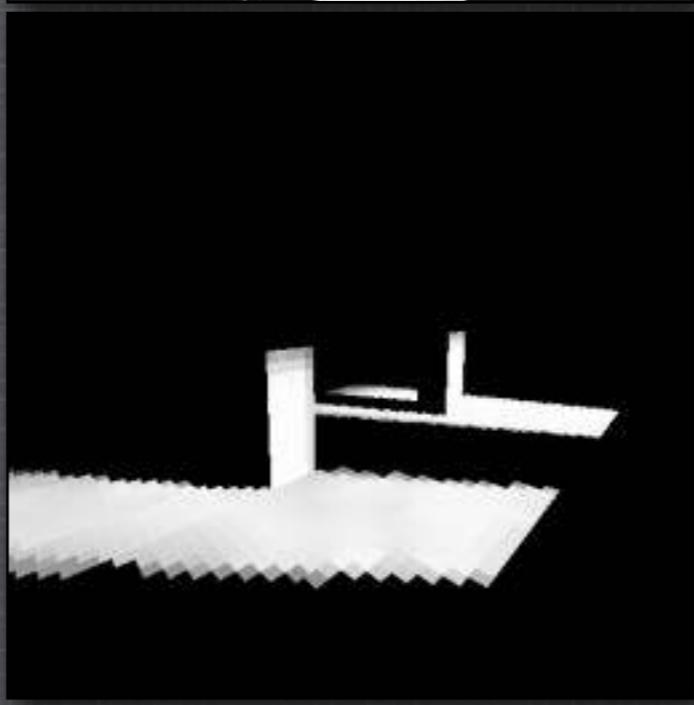
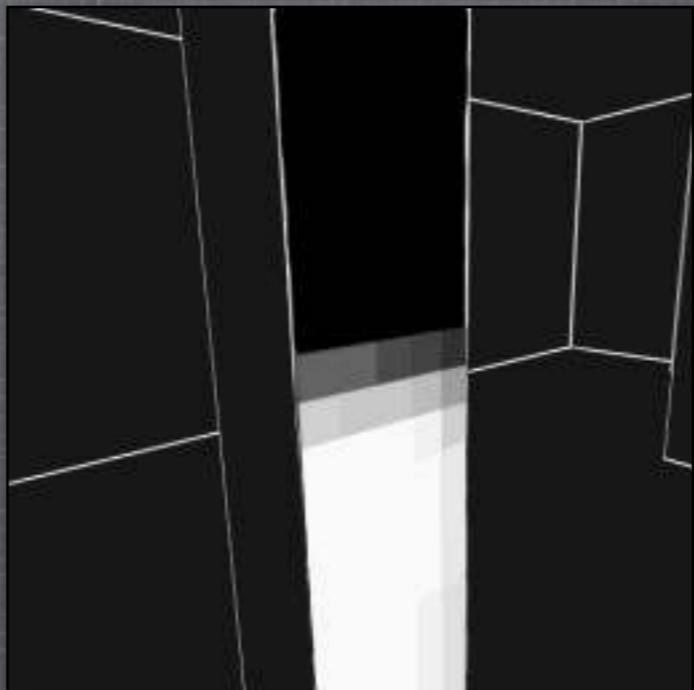
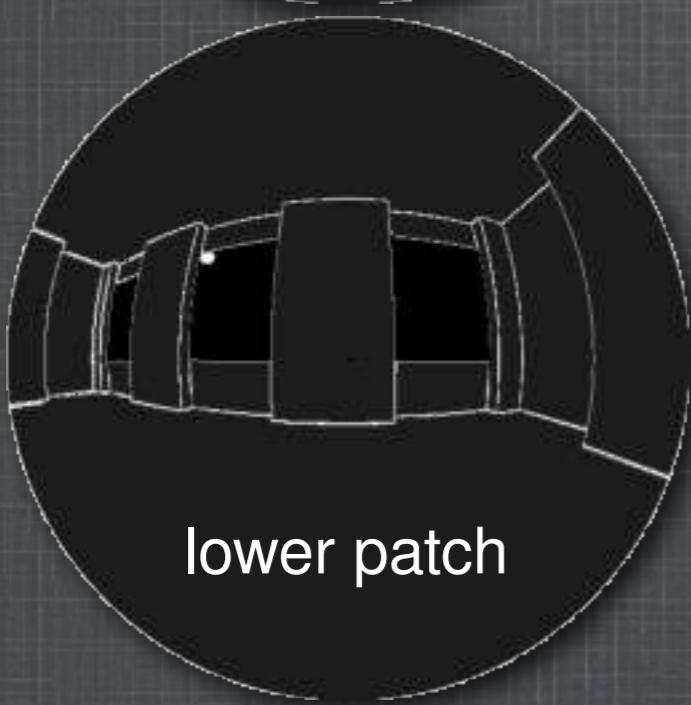
RENDERING

Radiosity (Goral, Torrance et al. 1984)



RENDERING

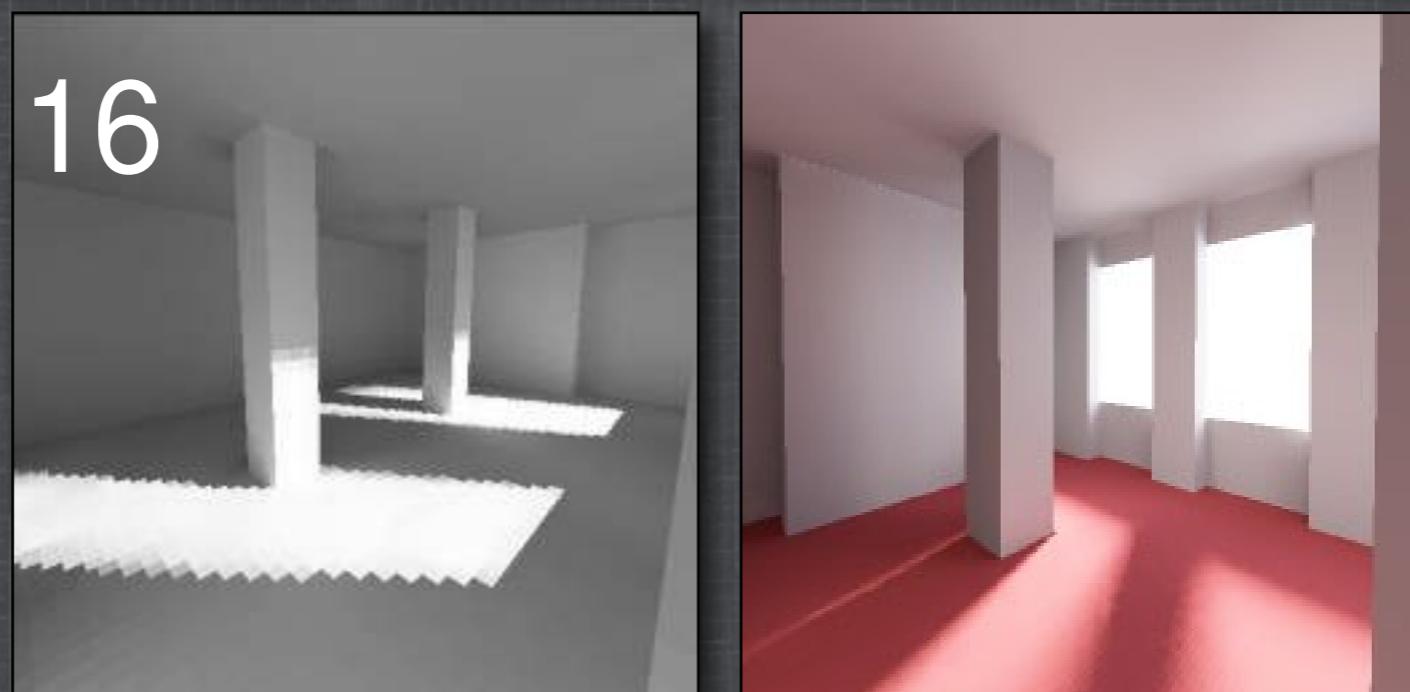
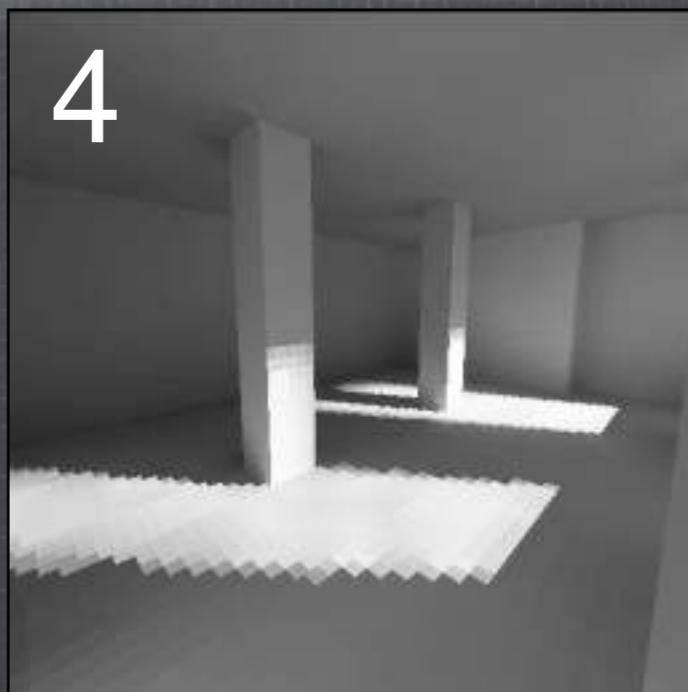
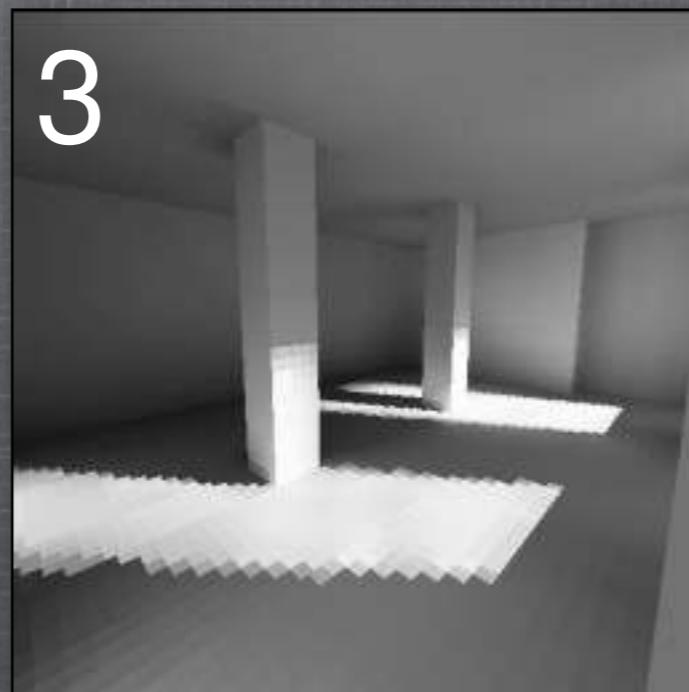
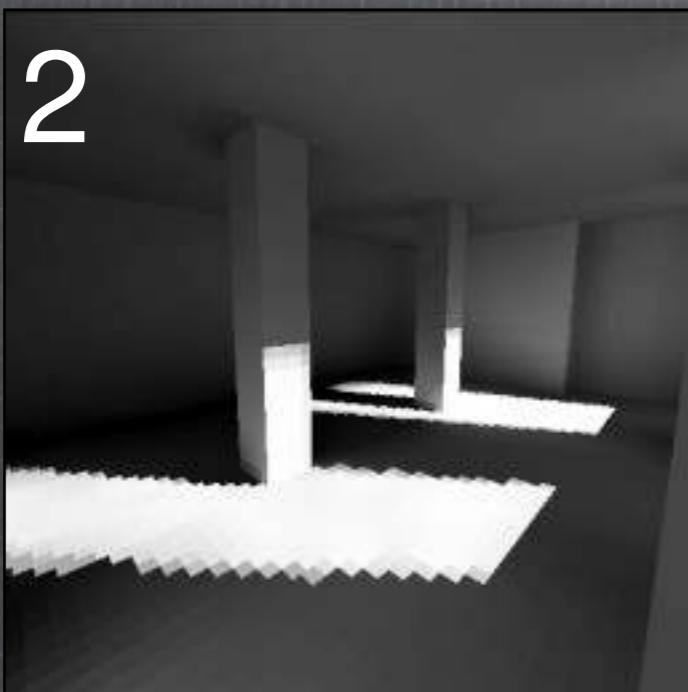
Radiosity (Goral, Torrance et al. 1984)



dark patch
not dark
anymore

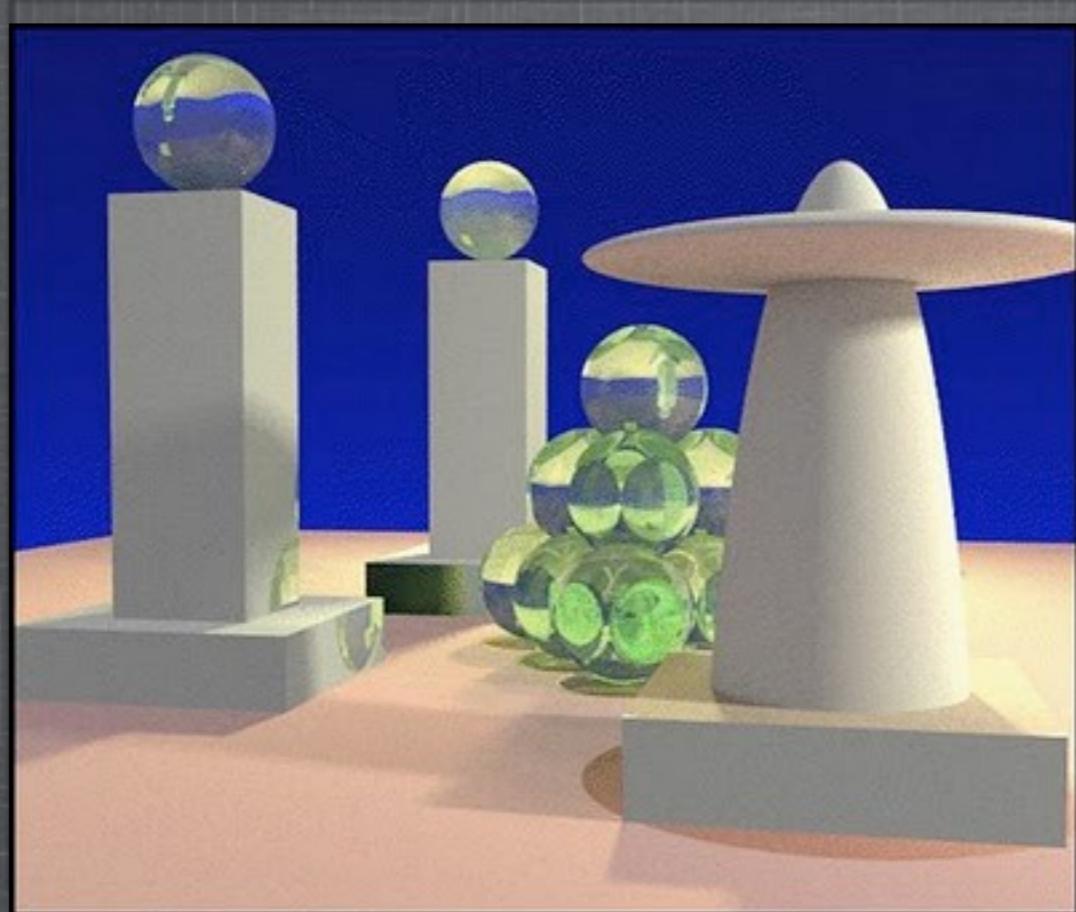
RENDERING

Radiosity (Goral, Torrance et al. 1984)



RENDERING

Global Illumination (1980's and 1990's)



Raytracing
(Whitted 1980)

Radiosity
(Goral, Torrance et al. 1984)

Rendering equation
(Kajiya, 1986)

RENDERING

Global Illumination (1980's and 1990's)

The Rendering Equation

Jim Kajiya, 1986



$$I(x, x') = g(x, x') \left[\epsilon(x, x') + \int_S \rho(x, x', x'') I(x', x'') dx'' \right]$$

- $I(x, x')$ – the total intensity from point x' to x
- $g(x, x') = 0$ when x/x' are occluded and $1/d^2$ otherwise (d = distance between x and x')
- $\epsilon(x, x')$ – the intensity emitted by x' to x
- $\rho(x, x', x'')$ – intensity of light reflected from x'' to x through x'
- S – all points on all surfaces

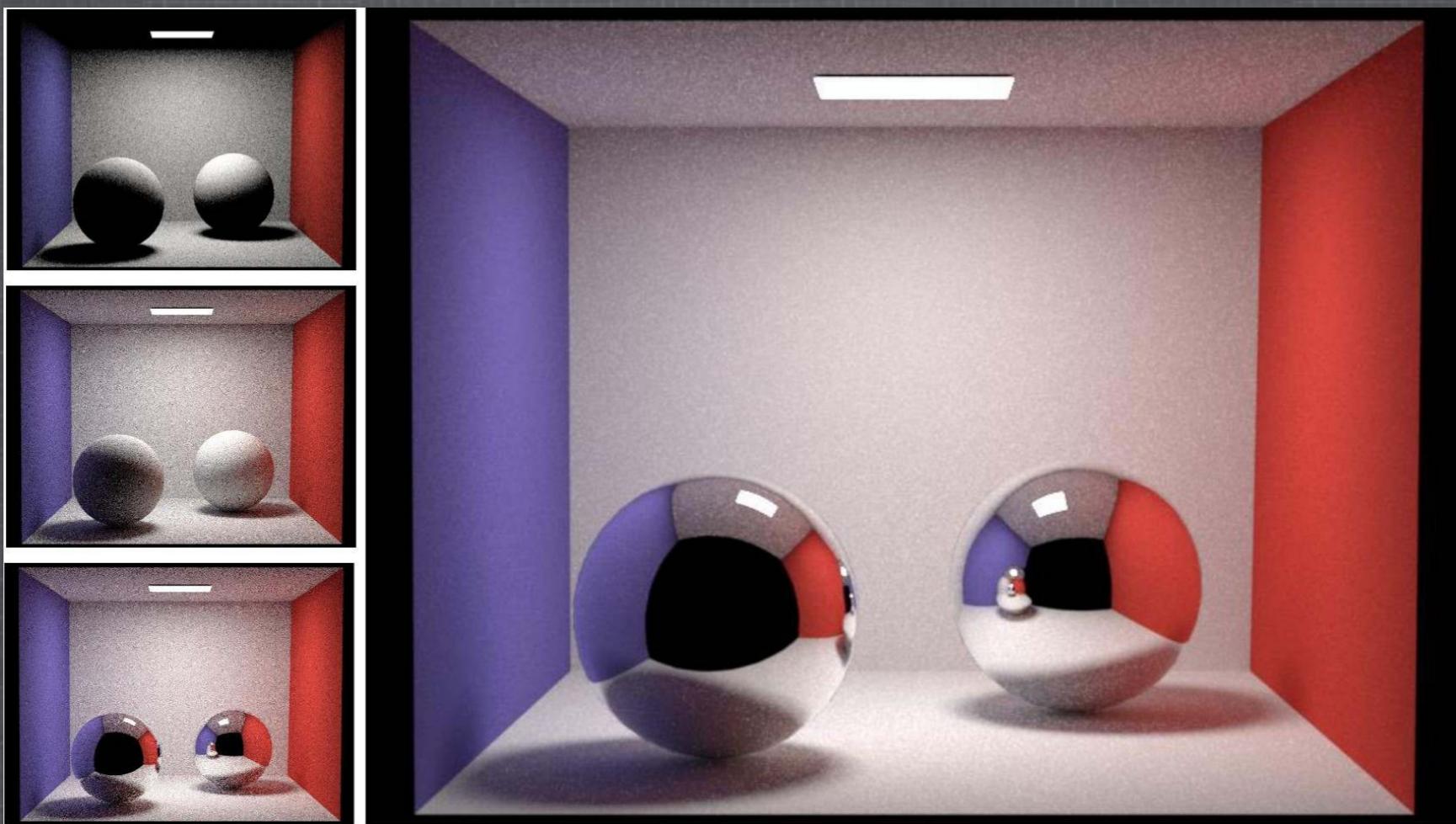
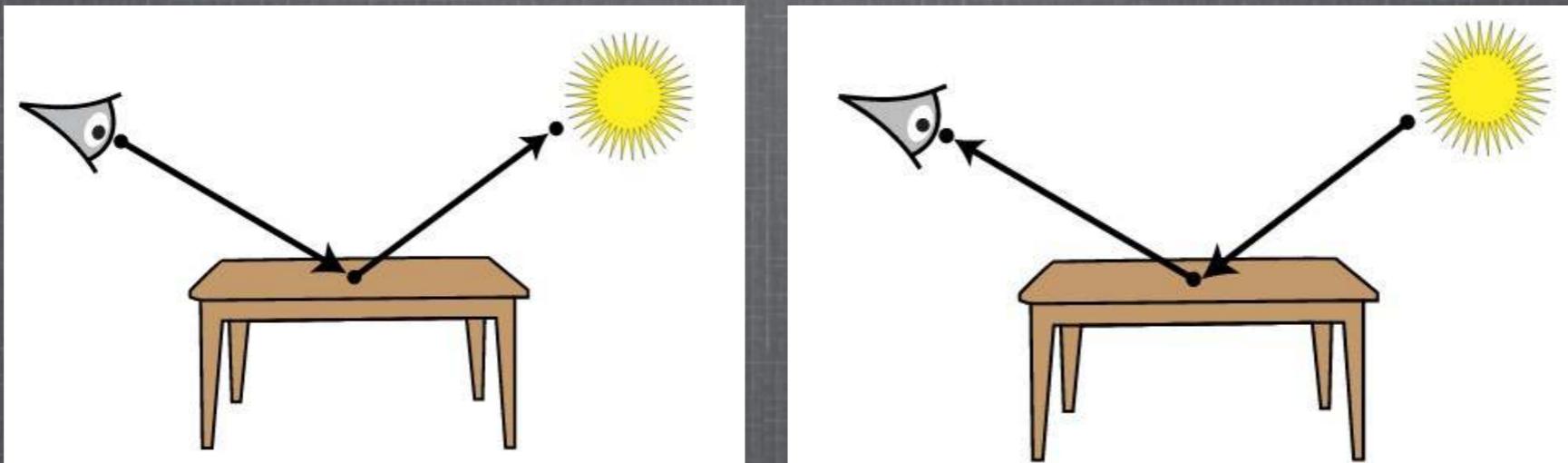
Raytracing
(Whitted 1980)

Radiosity
(Goral, Torrance et al. 1984)

Rendering equation
Path Tracing
(Kajiya, 1986)

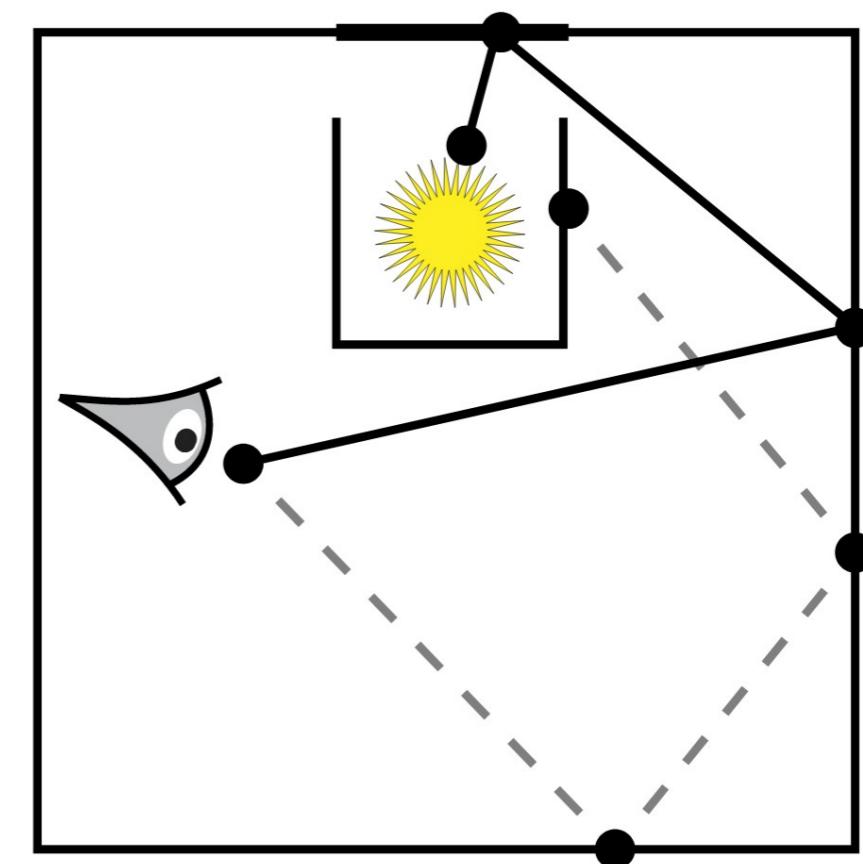
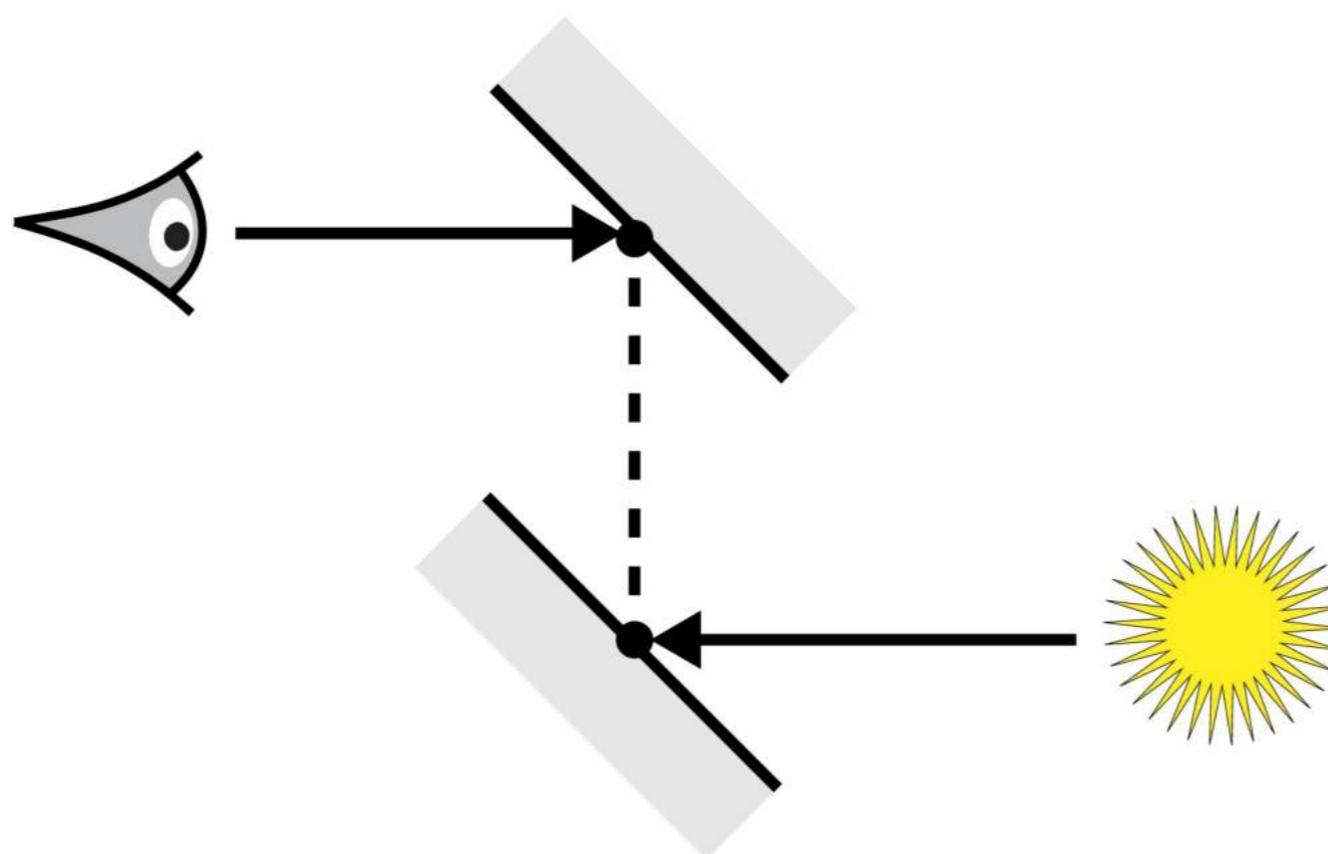
RENDERING

Path Tracing



MODERN PATH TRACING

Bidirectional Path Tracing



RENDERING

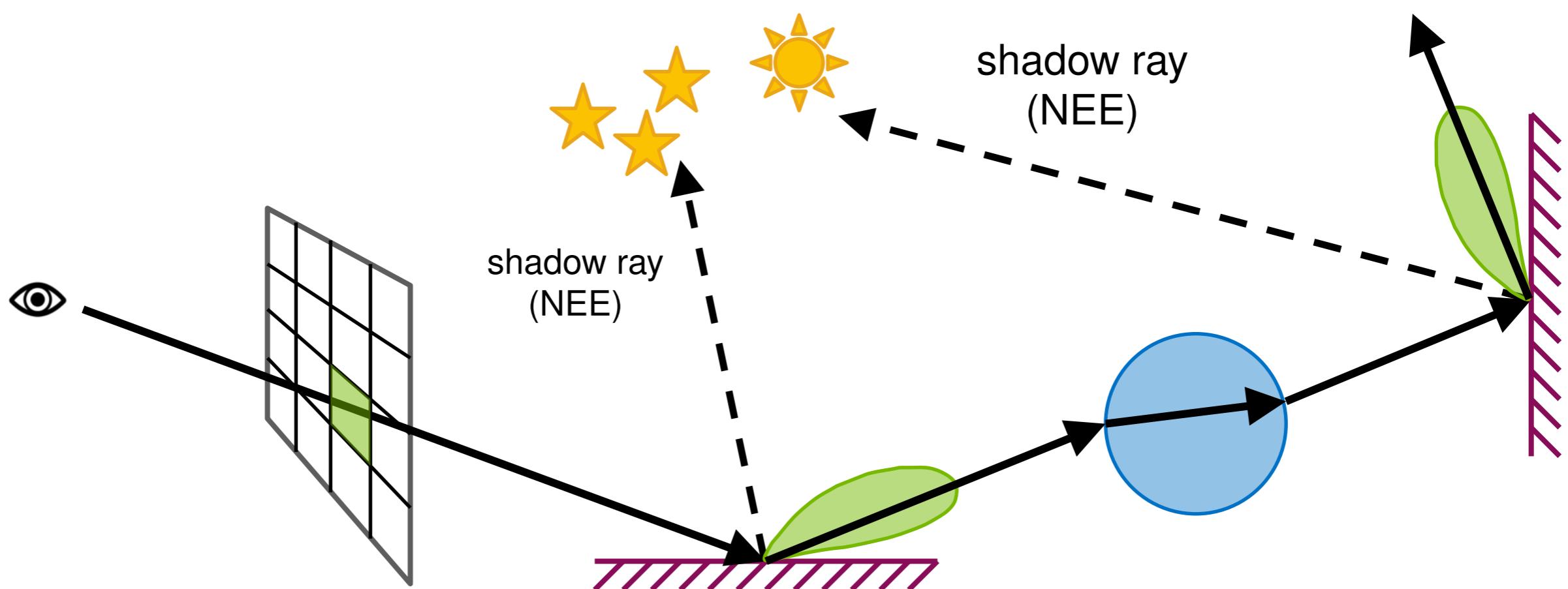
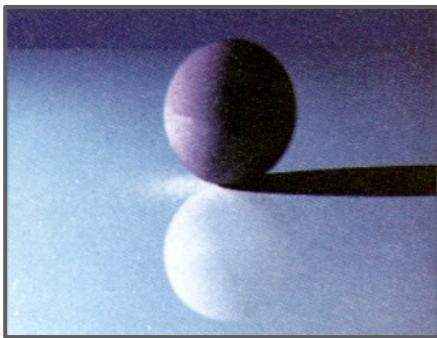
(early) Realtime Path Tracing



RENDERING EQUATION

Kajiya @ SIGGRAPH 1986

MODERN PATH TRACING



RENDERING WITH DENOISING



Realtime Path Tracing NVIDIA 2022 (30FPS, 1080p, 30 bounces, 30B triangles)



Realtime Path Tracing NVIDIA 2022 (30FPS, 1080p, 30 bounces, 30B triangles)

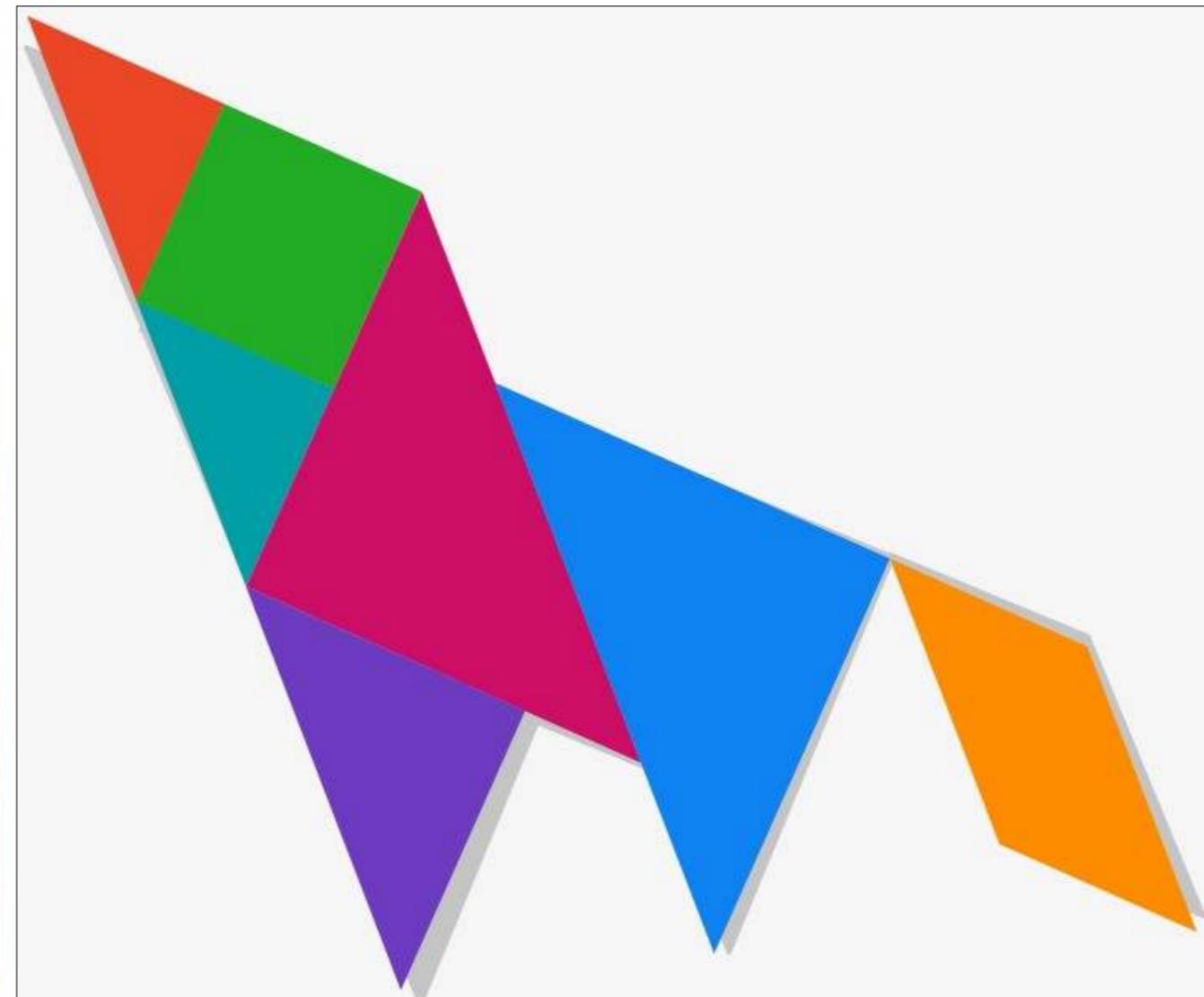
QUARTER PLANNING



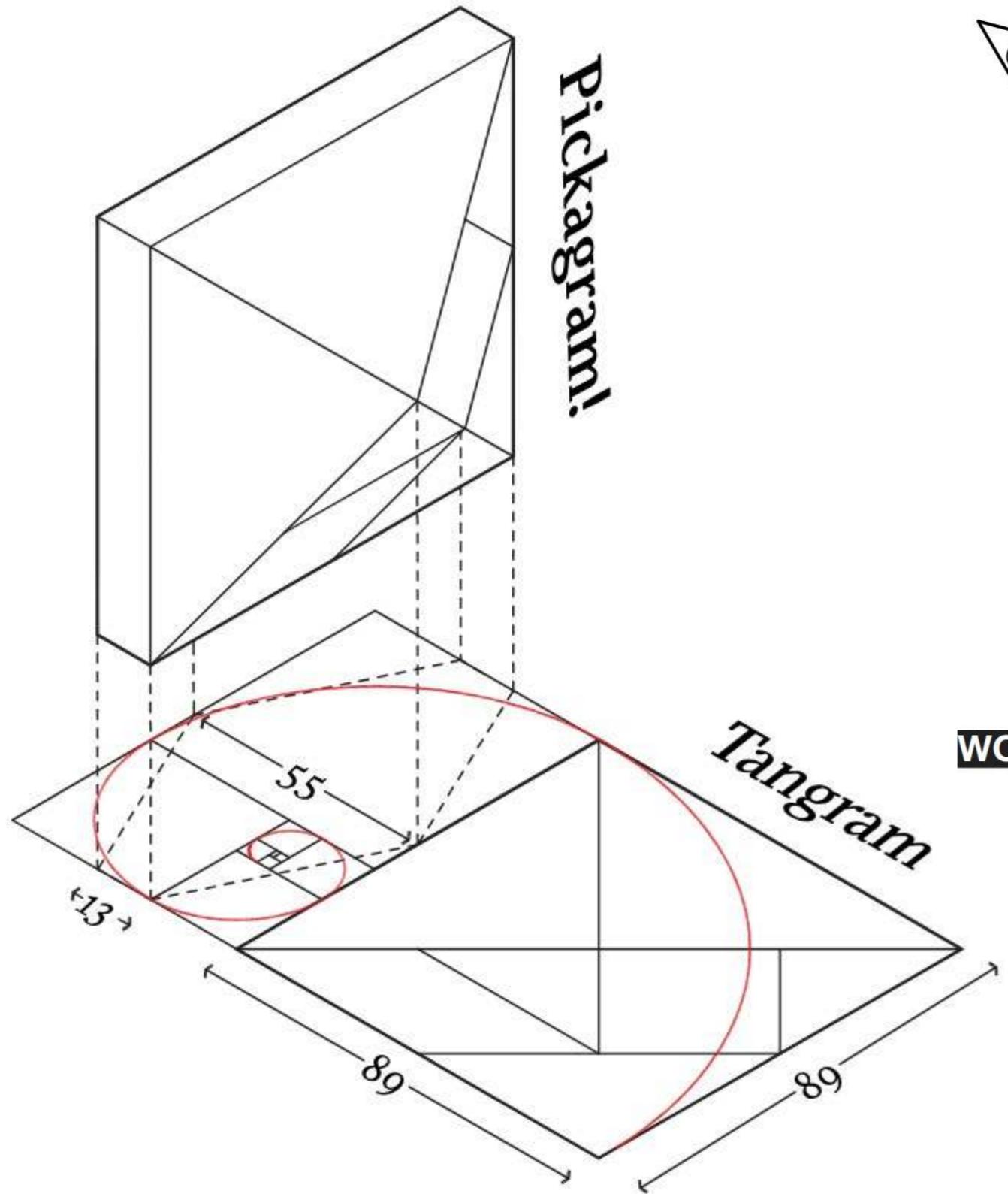
November 2025

W	Date	Lectures	Labs	Assignments due	Grade
1	Nov 11 (Tue)	Introduction, Review	Team forming C++ IDE		
1	Nov 14 (Fri)	Mathematics for CG (part 1): dot, cross, matrices	GLM / GLM setup		
2	Nov 18 (Tue)	Mathematics for CG (part 2): matrices, quaternions	GLM library	A1 : GLM (50% in class) Nov 20 (Thu)	1.0
2	Nov 21 (Fri)	Programmable Graphics Pipeline, Drawing in OpenGL	OpenGL / GLFW / GLEW Setup	Tan/Pick-a-gram Shape (in class)	
3	Nov 25 (Tue)	Viewing Pipeline Viewing in OpenGL	Drawing + Transformations	A2 : 2D Tangram Nov 27 (Thu)	3.0
3	Nov 28 (Fri)	Geometry Management Meshes and Scene Graphs	Viewing + Quaternions		

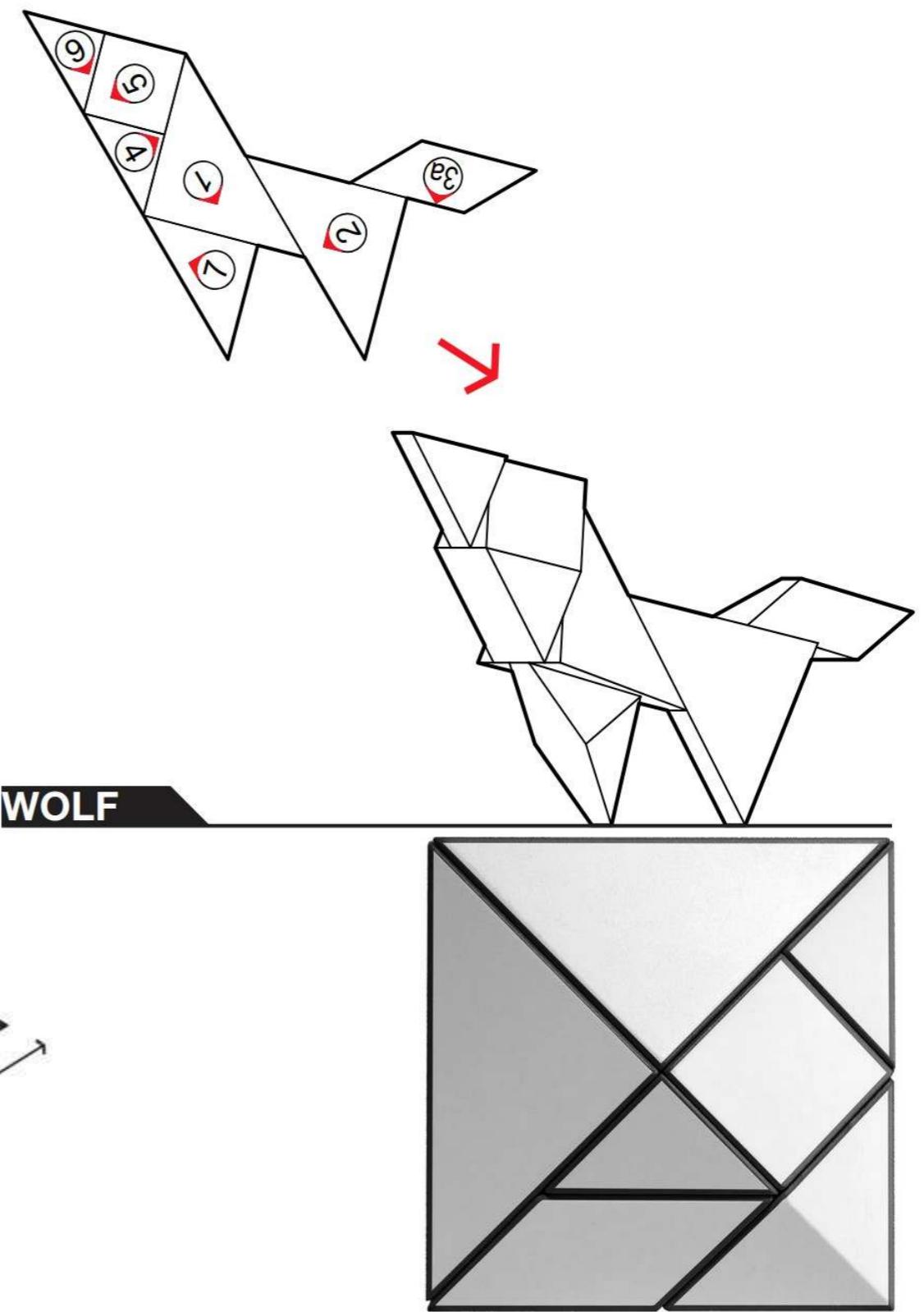
Tangram



Pickagram

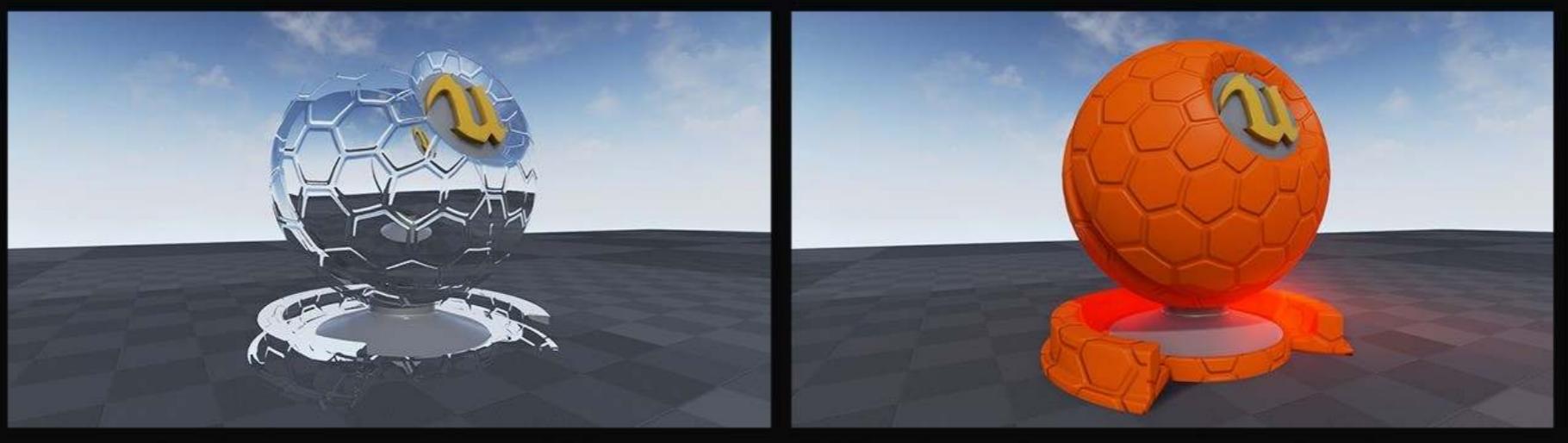


Tangram



December 2025

W	Date	Lectures	Labs	Assignment due	Grade
4	Dec 2 (Tue)	GLSL Shaders Lighting (1): Cel Shading	Meshes + ASSIMP Library		
4	Dec 5 (Fri)	Lighting (2): Gouraud and Phong Shading	Project Support	A3 : 3D Pickagram Dec 8 (Mon)	4.0
5	Dec 9 (Tue)	Mapping (1): Texturing, Sampling and Filtering	Shader Project Individual Project Definition	A4 : Project Proposal Dec 11 (Thu)	1.0
5	Dec 12 (Fri)	Mapping (2): Procedural Textures and Noise	Goal Definition and Execution Planning		
6	Dec 16 (Tue)	Mapping (3): Environment Mapping and Fresnel Reflectance	Project Support		
6	Dec 19 (Fri)	Mapping (4): Render Target Textures, Tangent Space, Normal and Displacement Mapping	Project Support		



January 2026

W	Date	Lectures	Labs	Assignments Due	Grade
7	Jan 6 (Tue)	Mapping (5): Ambient Occlusion and Physically Based Rendering	Project Support		
7	Jan 9 (Fri)	(project report Q&A)	Project Support	A5 : Shader Project Jan 9 (Fri)	8.0
8				A5: Project Report Jan 12 (Mon) Project Discussions Jan 15-16 (Thu-Fri)	3.0

IN BRIEF

- 40% - Base Project (teams of 2, 3 checkpoints)
- 60% - Shader Project (individual, 2 checkpoints)

PROGRAMMING

Principles and Practice Using C++

THIRD EDITION



BJARNE STROUSTRUP
THE CREATOR OF C++

Choose your C++/GLSL IDE



VISUAL STUDIO 2022

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

CONTACT

CARLOS.MARTINHO@TECNICO.PT

CGJ Office hours on Fénix.