# Computer Graphics & OpenGL

Sam Sartor

February 1, 2017

Mines Linux Users Group

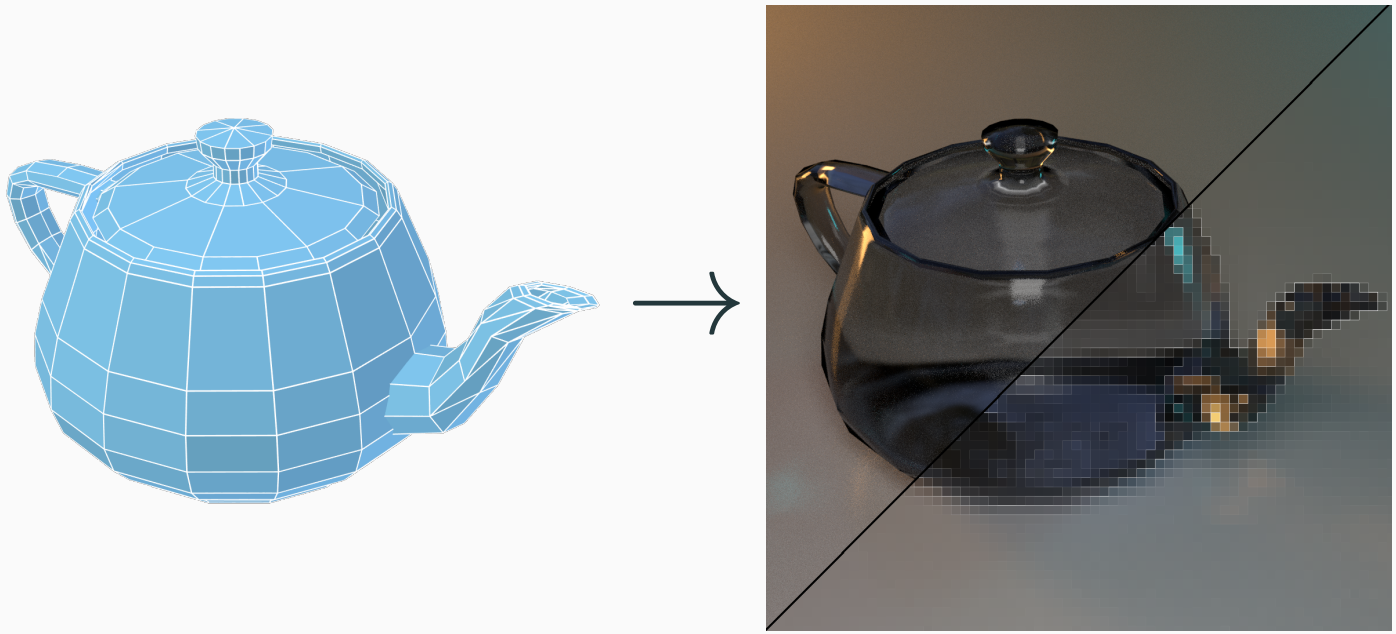
# Introduction

# Uses

Computer graphics is everywhere!

- Your terminal
- Web browsers
- Video games
- CAD software
- Movies, TV Shows
- Virtual reality
- Your bootloader
- QT, GTK+, wxWidgets
- Vim, Emacs, Notepad
- Embedded devices

# Definition



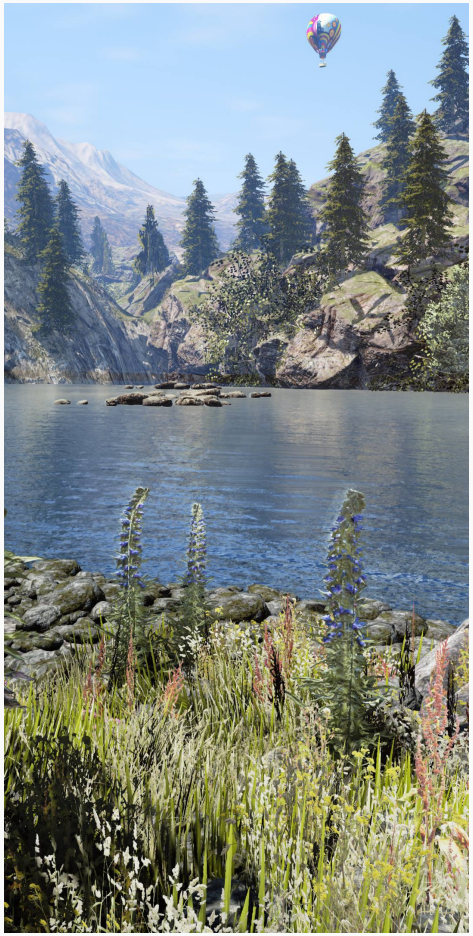
Computer graphics is the science of turning *shapes* into *pixels*. [1]

[1] Kindof, it can get more interesting than that

# Behind the Scenes

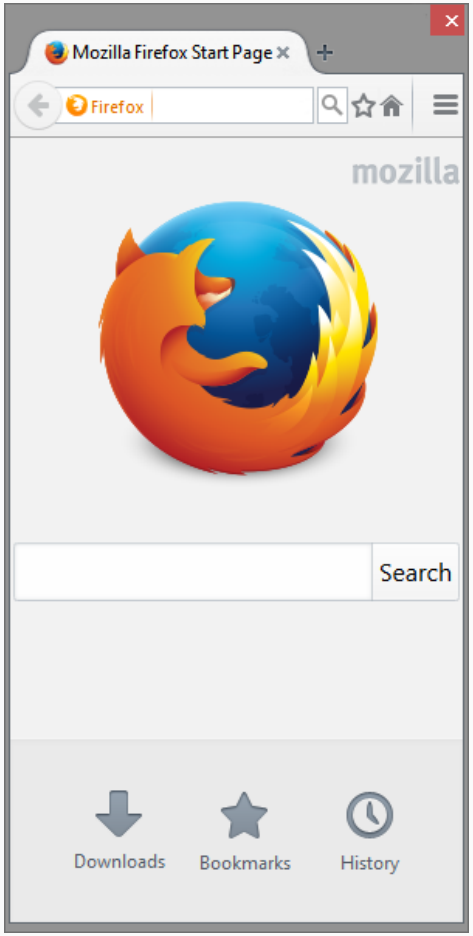
# Realtime

Realtime graphics use OpenGL or Direct3D to rasterize and shade triangular geometry on a graphics card/chip. Performance is very important due to the high framerate that is required for smooth gameplay/interactivity/animation. Lighting and materials focus on being "good enough" rather than on being truly accurate.



# UIs

While they look different, UIs generally use OpenGL or Direct3D as well. Everything is still made of textured & shaded triangles. Anti-aliasing, text fidelity, etc. are all more important while lighting effects are generally absent. Responsiveness is key, but the frame can be updated as needed, not every 30th of a second.



# Offline

Offline graphics are used when the medium is non-interactive (movies, advertisements, etc). Because the available resources are limited only by budget and patience, offline graphics have unmatched fidelity. CPUs are often used instead of GPUs because this allows for more advanced calculations. However, this comes at a cost. Individual frames may take days to render.

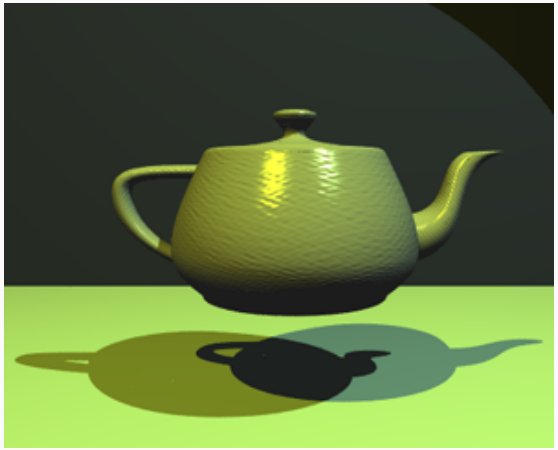
# History

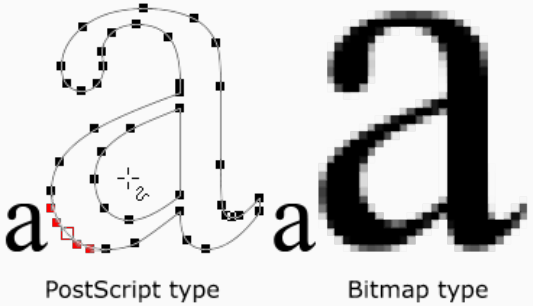
# 1950s & 1960s

- Military used computer controlled oscilloscopes to display strategic information
- Very simple graphical CAD programs and visualizers created
- Very first computer games
- Research into elementary 3D wireframe graphics
- Very early raster displays

# 1970s & 1980s

- Basic lighting models such as Phong developed
- Low-res, 2D games become commercially available
- CGI starts to be used in Movies such as 1982's *Wrath of Khan* and 1985's *Young Sherlock Holmes*
- Modern GUIs are developed
- High-quality digital typesetting becomes commonplace



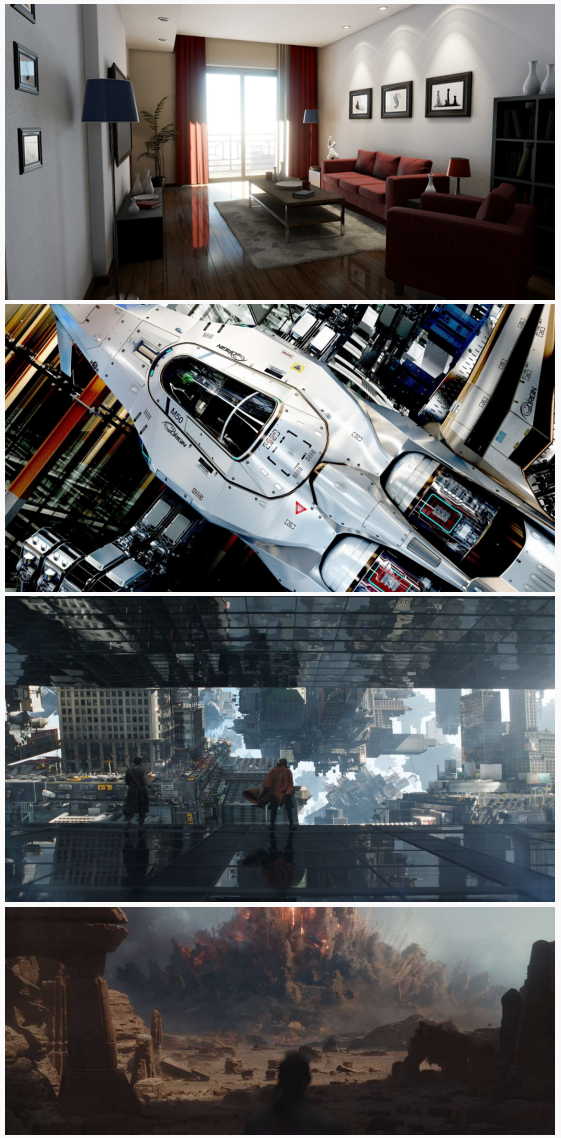


# 1990s & 2000s

- Fidelity and performance are immensely increased
- Personal computers, 3D video games, and GUIs become ubiquitous
- OpenGL and Direct3D standardize hardware graphics support
- CGI becomes commonplace in Movies, advertisements, and TV
- Global illumination and physically based rendering (PBR) techniques developed

# Today

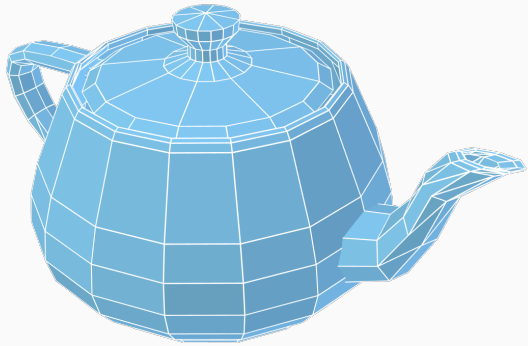
- Given enough time, budget and expertise, offline graphics are photorealistic
- Particle and fluid simulations are extremely fast and accurate
- Realtime graphics make extensive use of shaders and PBR techniques
- UIs and offline graphics are increasingly GPU accelerated
- Linux and Mac have improved support for games and graphical software



# Fancy-pants-ery

# Meshes

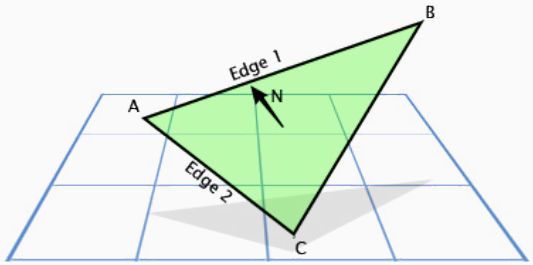
3D shapes are often stored as a jumble of points (called verticies), scattered in space. These points are linked together into polygons, which form the surface of the model. Other methods of storing 3D shapes exist, such as the parametric forms used by Solid Works, but "meshes" are by far the most common. In fact, even when a different form is used, the model is almost always converted into a mesh for rendering anyway.



# Triangles

While any polygon could be used in a mesh, modern computer graphics deal almost exclusively in triangles. This is because triangles have some nice geometric properties that other shapes don't:

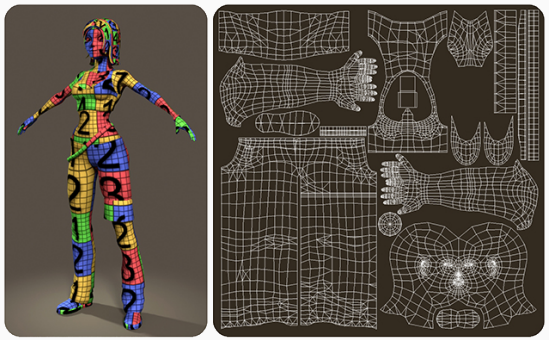
- It is very easy to test if a point is within a triangle (barycentric coordinates!)
- Any 3 points can be haphazardly connected together and still form a flat triangle
- It is easy to calculate the direction a triangle is facing



# Textures

Textures are images (though they can contain more than colors) that are wrapped around 3D models. The exact method for this varies, but most of the time the wrapping is defined by a second set of coordinates attached to each vertex/point. This specifies where on the texture that point (and by extension, attached triangles) should lie.

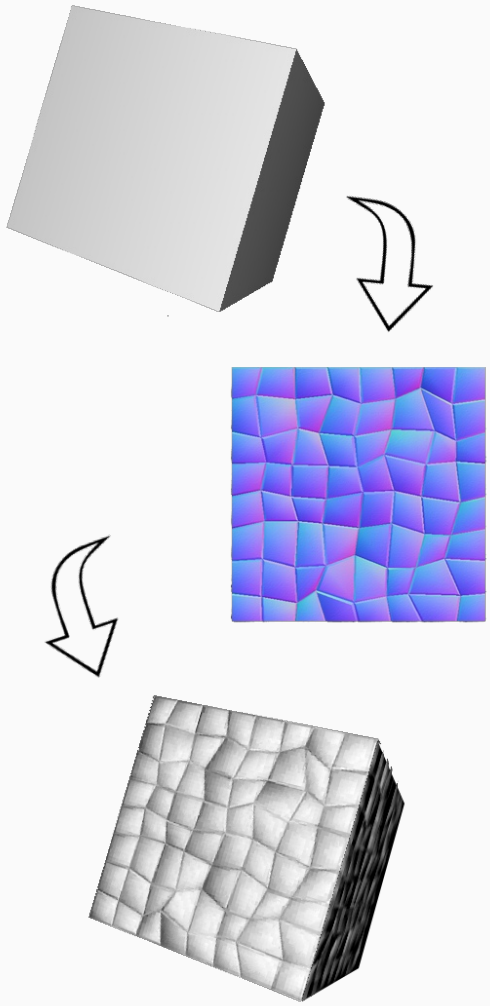
This second set of positions is called the "UV map".



# Mapping

Textures can be used to cover objects in all sorts of information. Not just color, but also material properties such as shininess and even precaculated lighting information.

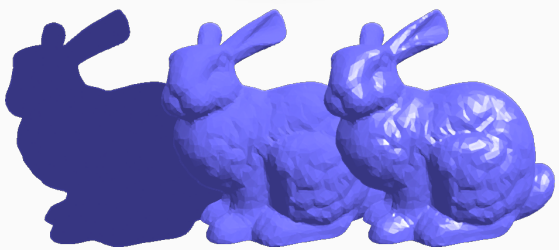
More advanced techniques (such as bump, normal, and parallax mapping) can be used to fake bumps and other minute, shaded details on technically flat surfaces. Games use this extensively to make low-resolution objects appear highly detailed (think screws, buckles, bricks, tree bark, ground texture, etc.)



# Phong

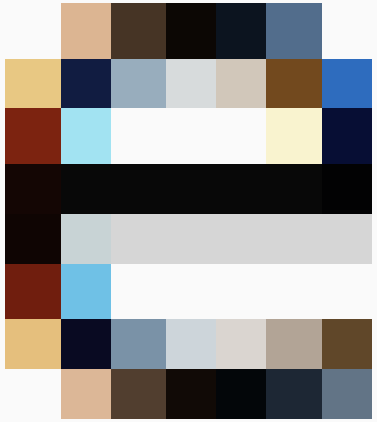
Phong is the simplest common shading model. It mixes diffuse shading (areas facing a light are brighter) and specular shading (areas that reflect light towards the camera are brighter) together. Think of how you would shade something with a pencil.

While this kind of works, it massively simplifies how real-world materials and lights interact. As a result, it looks downright fake.



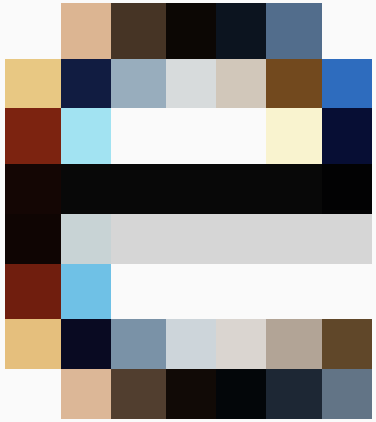
# Painter's Algorithm

- ...



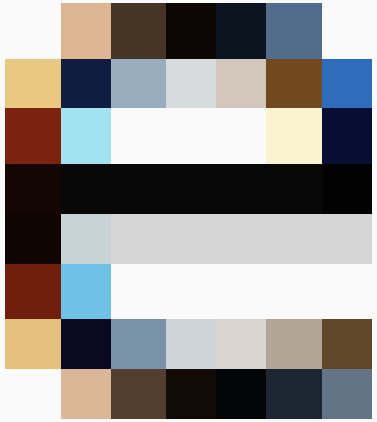
# Depth

- ...



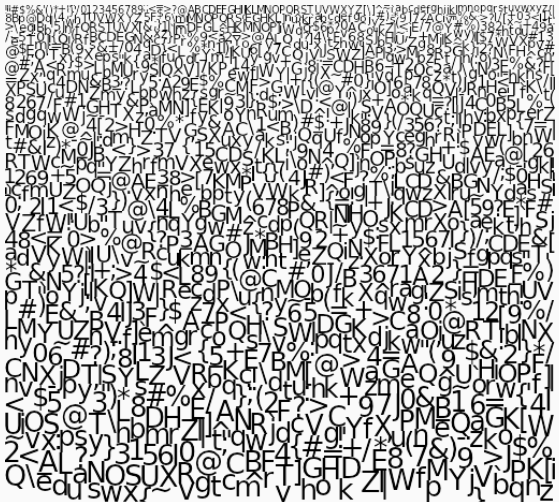
# 3D Modeling

- ...



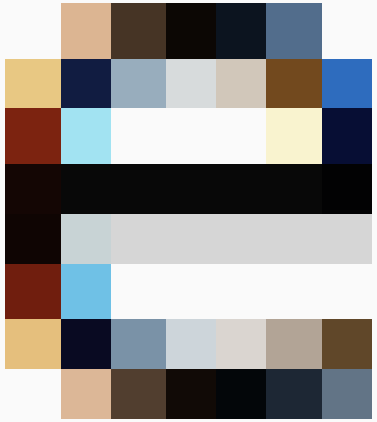
# Text

- ...



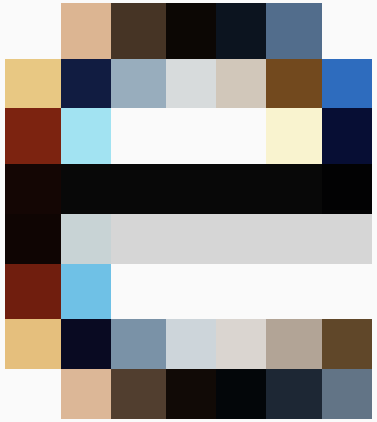
# Sub-pixels

- ...



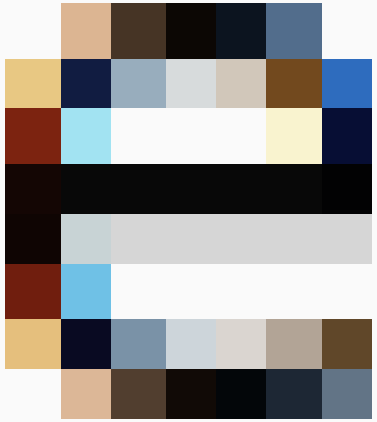
# Multipass Rendering

- ...



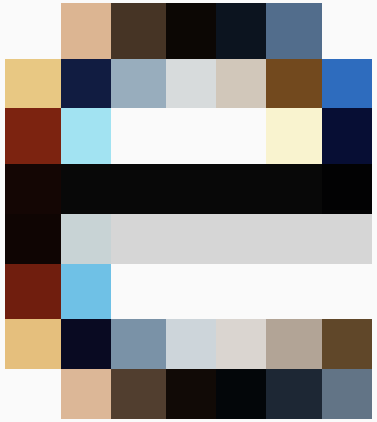
# Raytracing

- ...



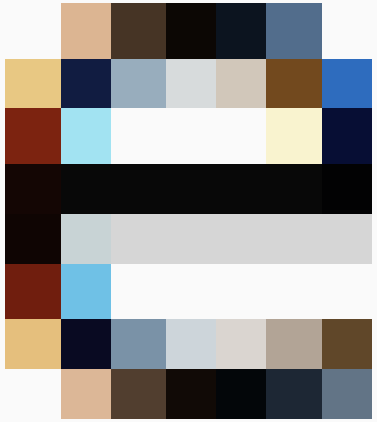
# Fresnel & PBR

- ...



# Pathtracing

- ...

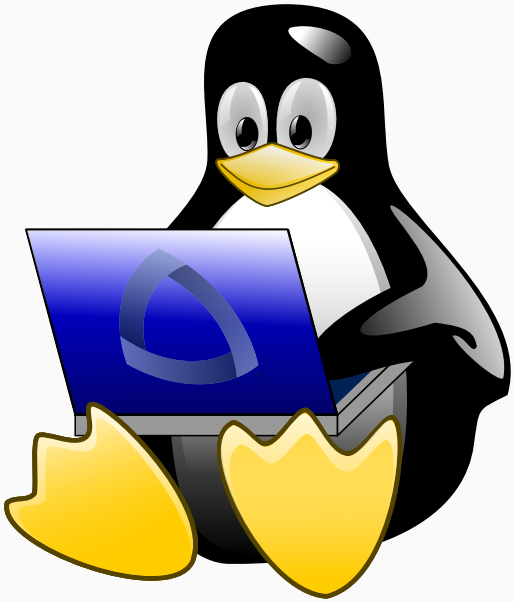


# Cool Stuff!

# Copyright Notice

This presentation was from the **Mines Linux Users Group**. A mostly-complete archive of our presentations can be found online at `https://lug.mines.edu`.

Individual authors may have certain copyright or licensing restrictions on their presentations. Please be certain to contact the original author to obtain permission to reuse or distribute these slides.



**Colorado School of Mines**
Linux Users Group