# Computer Graphics

Sam Sartor

February 2, 2017

Mines Linux Users Group

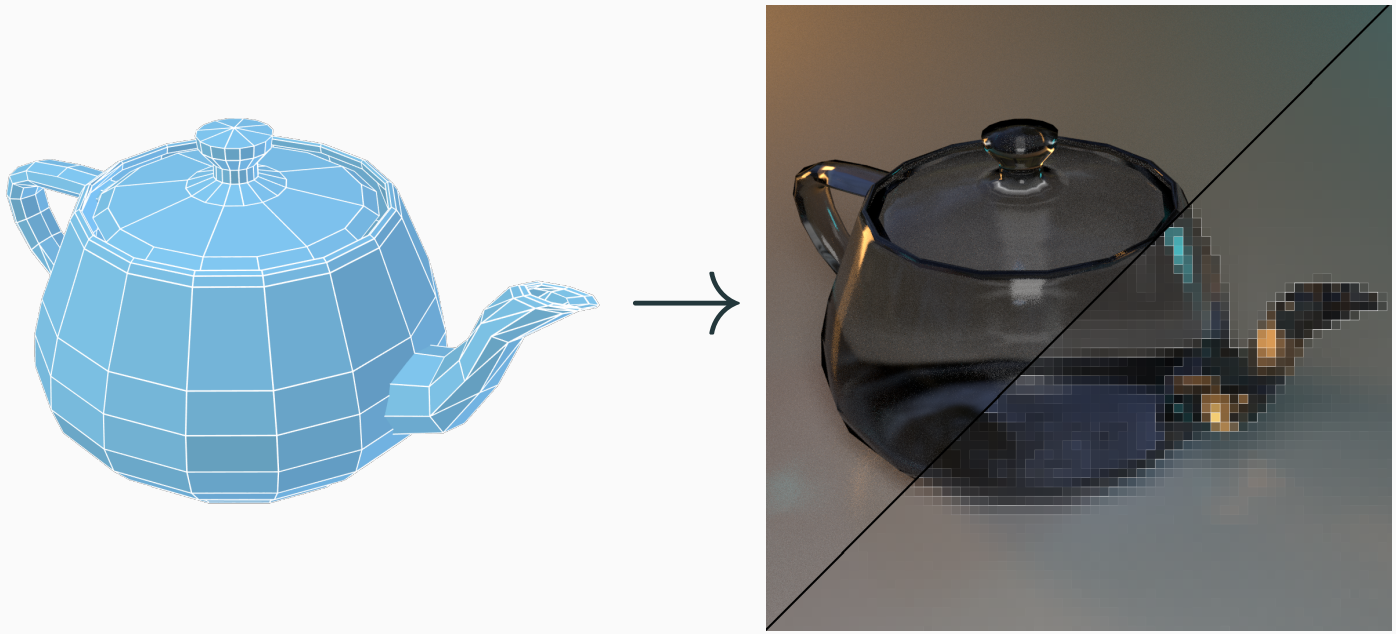
# Introduction

# 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



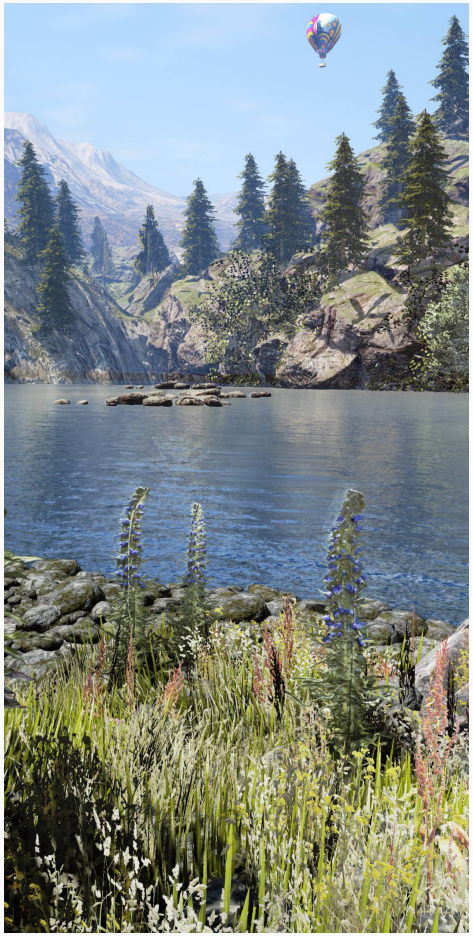
Computer graphics is the science of turning *shapes* into *pixels*. [1]

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

# Behind the Scenes

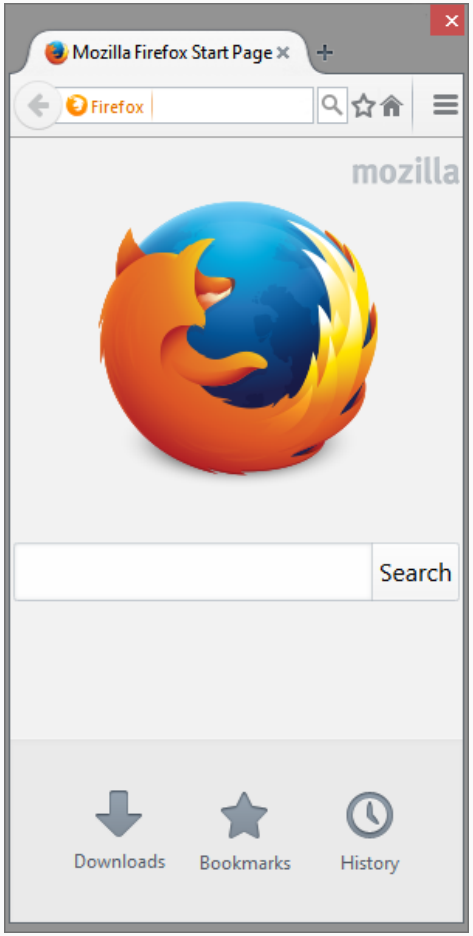
# Realtime

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.

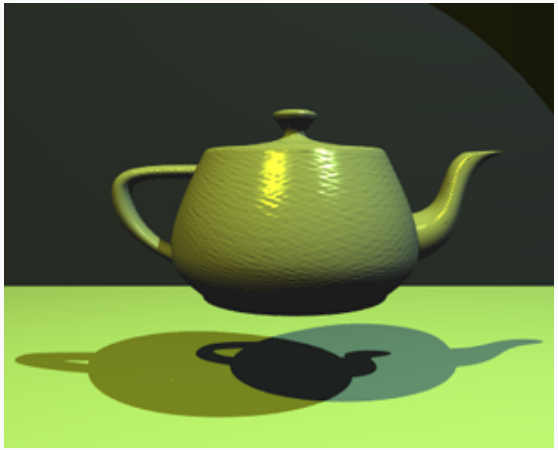
# History

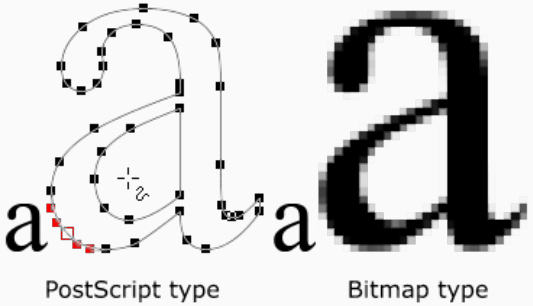
# 1950s & 1960s

- 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



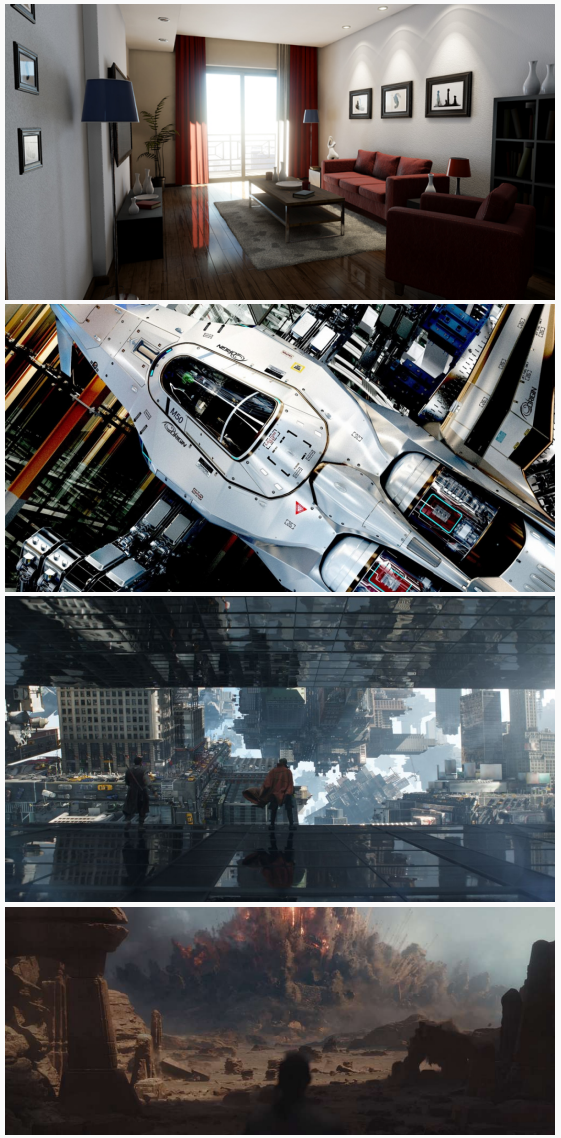


# 1990s & 2000s

- 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

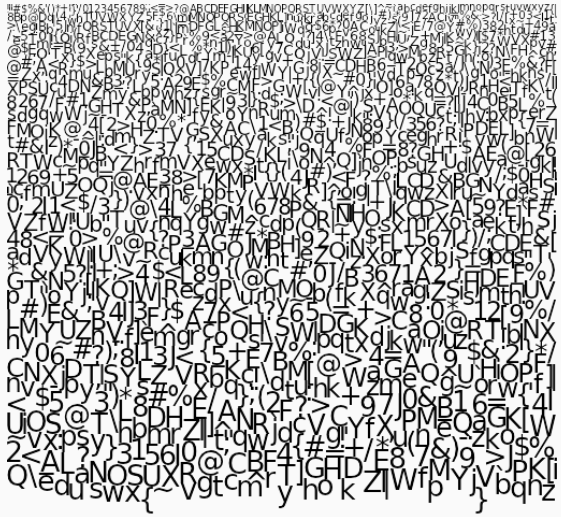
- Given enough time, budget and expertise, offline graphics are photorealistic
- Particle and fluid simulations are extremely fast and accurate
- Realtime graphics make extensive use of shaders and PBR techniques
- UIs and offline graphics are increasingly GPU accelerated
- Linux and Mac have improved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

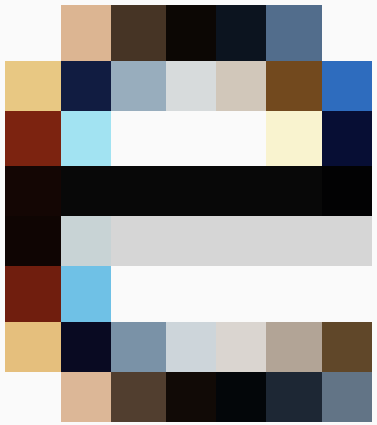


# Easy on the Eyes

# Text

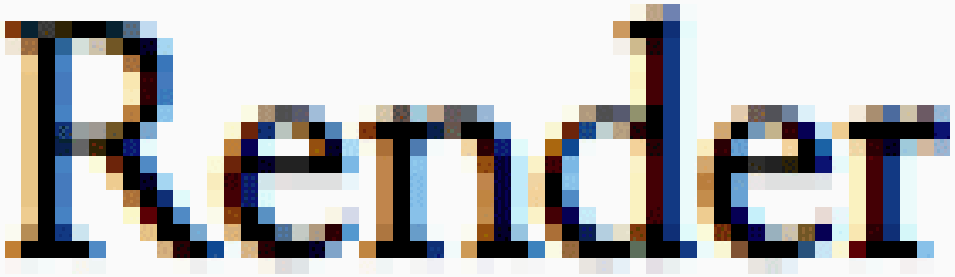
The most common thing shown on screens is probably text. It is no wonder then that displaying text is one of the most sphisticated areas of computer graphics.





# Which is better?

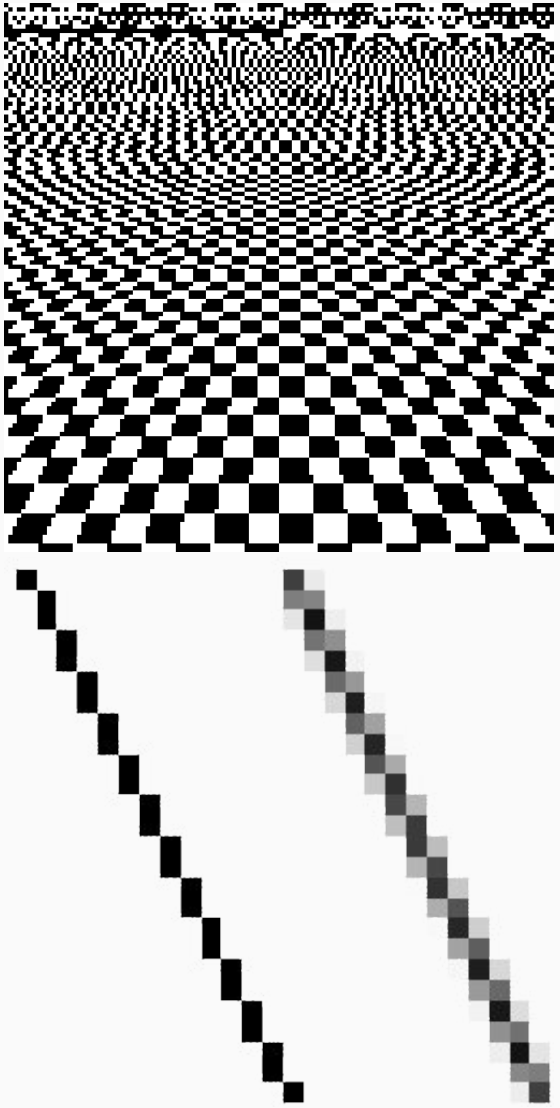
Standard Rendering 


Subpixel Rendering 

# Anti-aliasing

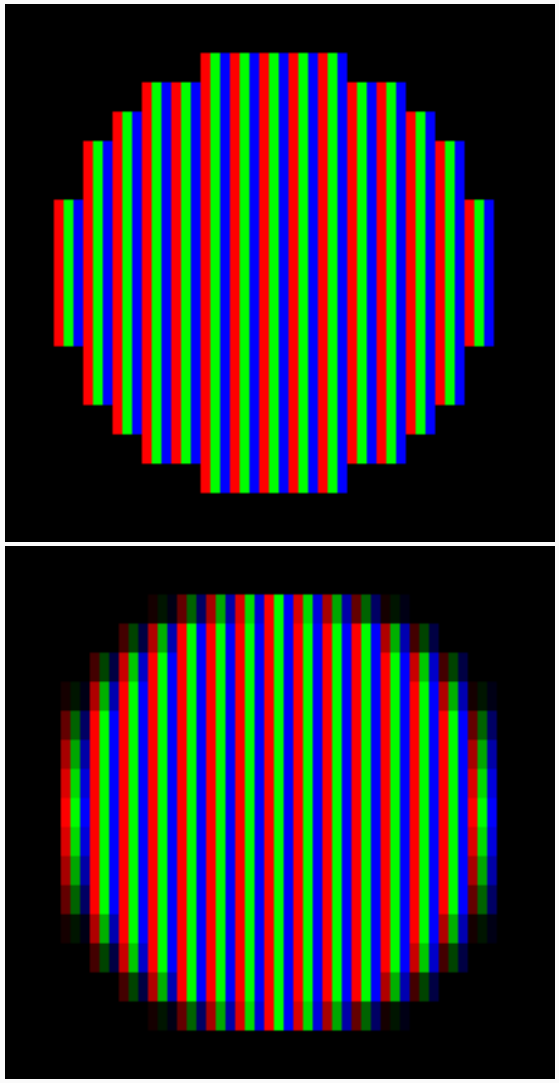
When small shapes (such as letters) are rasterized, details can find themselves squished into single pixels. In these cases, the color of the pixel is chosen arbitrarily depending on how the rasterization is performed. This creates all sorts of strange effects and hard edges, called "aliases".

The quality of the image can be greatly improved in some cases by using "anti-aliasing" algorithms and techniques.



# Sub-pixels

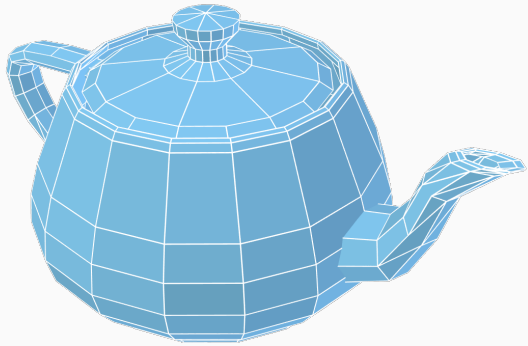
Anti-aliasing can be taken further. After all, each pixel is really 3 small red, green, and blue rectangles joined together. Clever people have figured out how to use these extra rectangles to further smooth the edges of letters and lines. This is why blown-up text (in an image editor) ends up being slightly colored on the edges.



# 3D Shapes & Geometry

# Meshes

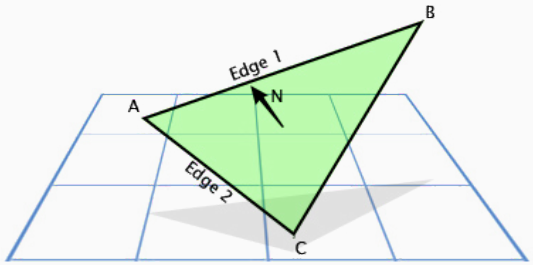
3D shapes are often stored as a jumble of points (called verticies), scattered in space. These points are linked together into polygons, which form the surface of the model. Other methods of storing 3D shapes exist, such as the parametric forms used by Solid Works, but "meshes" are by far the most common. 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:

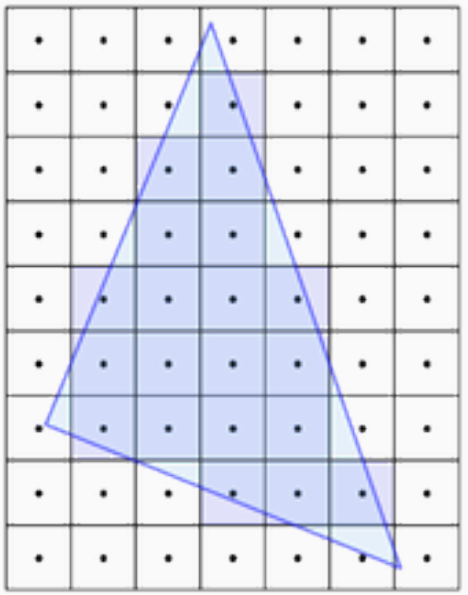
- It is very easy to test if a point is within a triangle (barycentric coordinates!)
- Any 3 points can be haphazardly connected together and still form a flat triangle
- It is easy to calculate the direction a triangle is facing

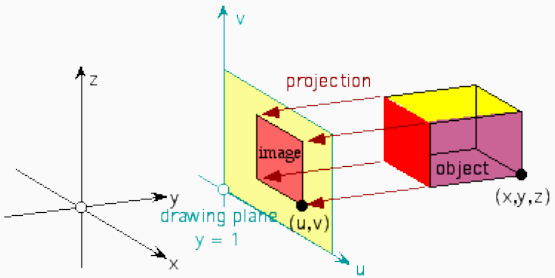


# Projection & Rasterization

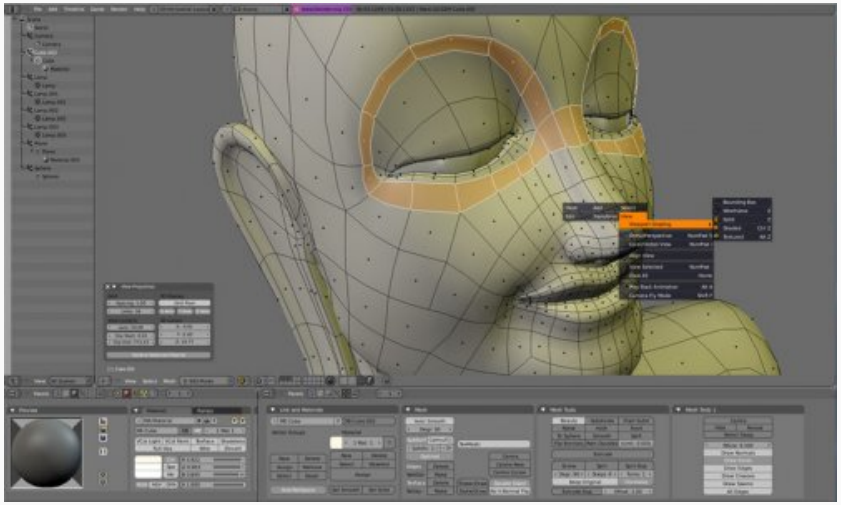
Meshes are 3D geometry. Screens are 2D pixels. Of the ways to convert between them, projection and rasterization is by far the simplest and most common. It works like so:

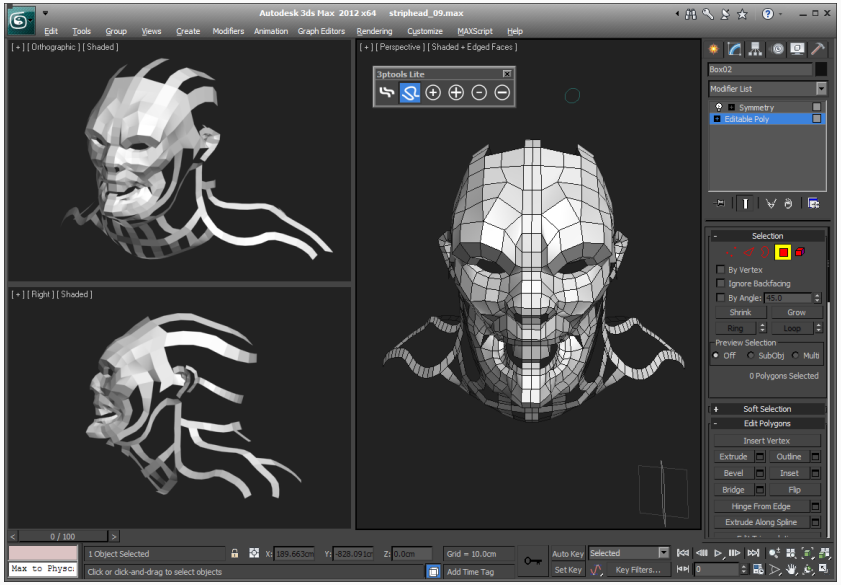
1. Rotate the scene such that $x$ is horizontal, $y$ is vertical, and $z$ is into the screen
2. Do math for perspective
3. Remove the $z$ value to "project" everything onto the $xy$ plane
4. Iterate over the pixels the triangle covers





# 3D Modeling



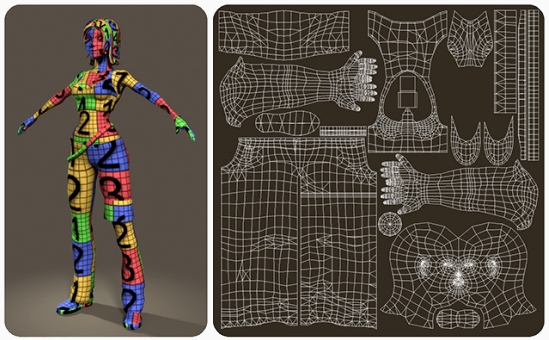

Autodesk 3ds Max 2012 x64 striphead_09.max
Edit Tools Group Views Create Modifiers Animation Graph Editors Rendering Customize MAXScript Help
[ + ] [ Orthographic ] [ Shaded ]
[ + ] [ Perspective ] [ Shaded + Edged Faces ]
[ + ] [ Right ] [ Shaded ]
3ptools Lite
Box02
Modifier List
Symmetry
Editable Poly
Selection
By Vertex
Ignore Backfacing
By Angle:
Shrink
Grow
Ring
Loop
Preview Selection
Off SubObj Multi
0 Polygons Selected
Soft Selection
Edit Polygons
Insert Vertex
Extrude
Outline
Bevel
Inset
Bridge
Flip
Hinge From Edge
Extrude Along Spline
0 / 100
1 Object Selected
Click or click-and-drag to select objects
Add Time Tag
Grid = 10.0cm
Auto Key
Selected
Set Key
Key Filters...

# Making it Pretty

# 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.

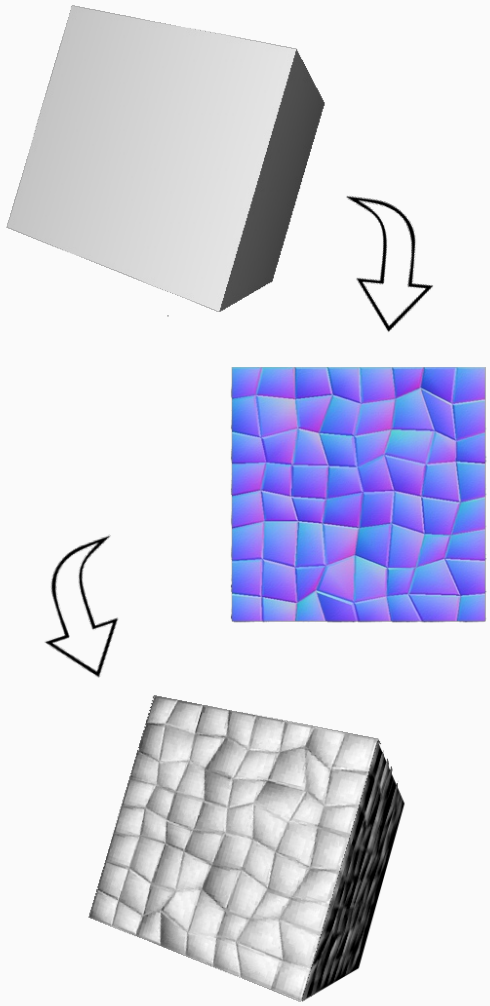
This second set of positions is called the "UV map".



# 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.

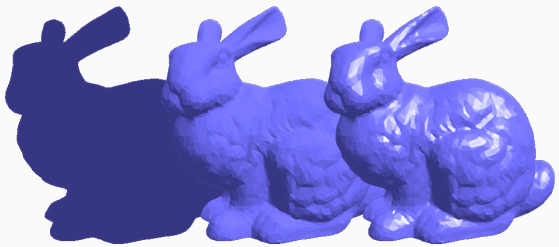
More advanced techniques (such as bump, normal, and parallax mapping) can be used to fake bumps and other minute, shaded details on technically flat surfaces. Games use this extensively to make low-resolution objects appear highly detailed (think screws, buckles, bricks, tree bark, ground texture, etc.)



# 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.

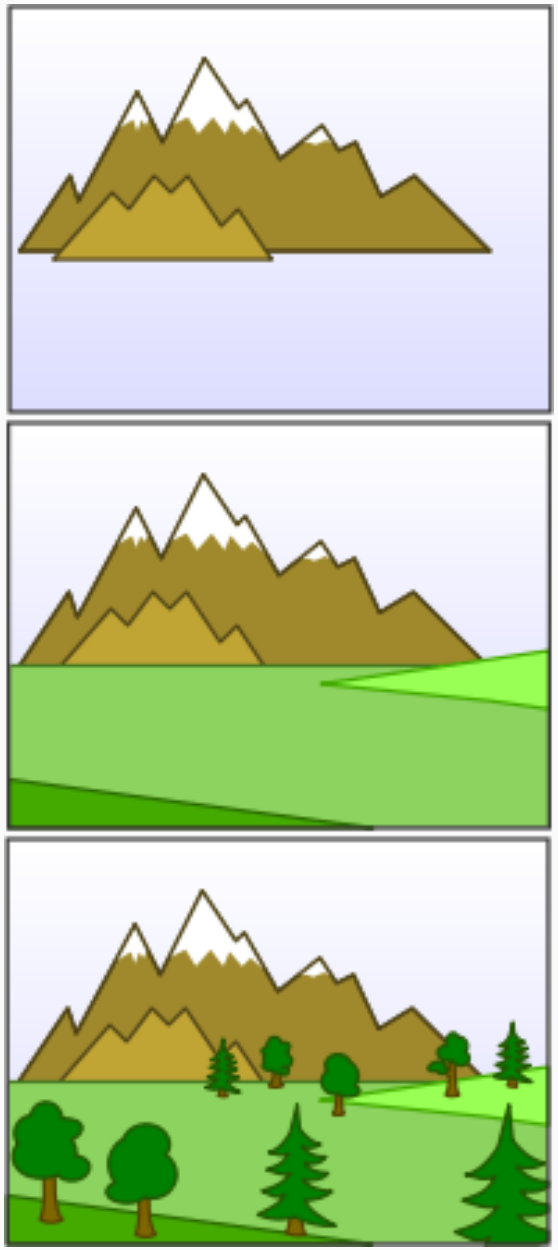


# Painter's Algorithm

It goes without saying that things that are farther away should be hidden by things that are closer. The simplest way to do this in a computer is to draw the farther stuff first, so that when nearer stuff is drawn, it covers the previous stuff up.

However, this requires all the objects in the scene to be constantly resorted. It also can't deal with objects that mutually cover each other (like the Olympic rings).

That being said, there are still cases where the Painter's Algorithm is the only viable method.



# Depth

A more common way to hide objects behind each other is to use a depth buffer. A depth buffer is a sort of distance image. In addition to keeping track of the color of each pixel, the distance from the camera to that pixel is also tracked. Then, whenever a new pixel is calculated, the computer checks whether is in front of or behind what is already there.

# Effects

# Transparency

Simplicity, fidelity, speed. Pick one. Once transparency gets involved, the color of a given pixel is suddenly determined by more than one surface. This problem of mixing colors is common to pretty much all difficult-to-do things in computer graphics.

Most of the time transparency is handled by resorting back to the Painter's Algorithm. The rest of the scene can still use depth-buffers, but it no longer makes sense to assign a single depth to those particular pixels.

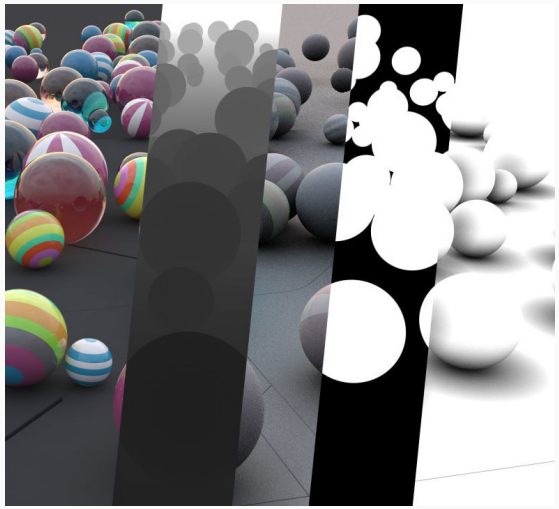
# Post-Processing

Texturing and shading is not always enough to make an image look good. Any photoshop guru can tell you how much a little tweaking can improve things. These "post-processing effects" arn't just used to make realtime graphics nicer, they are used to style-up photographs and videos as well.

# Multipass Rendering

Post-processing can be taken even further. Instead of lighting and textured being performed as the shapes are rasterized, data from along the way can be saved in buffers and stapled together after the fact.

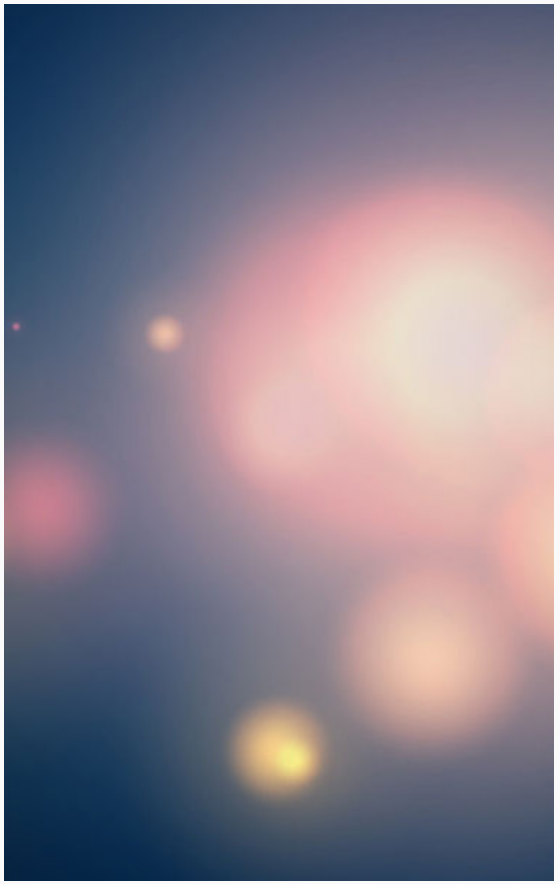
This has numerous benefits depending on how it is used. It is useful in realtime graphics because data from a single pass can be recycled to create multiple effects. Artists working on offline graphics like it because it lets them tweak shading and lighting without re-rendering the whole scene.



# Bluring

There are all sorts of algorithms to blur things, but the general case is pretty simple. The pixels around a point are weighted and averaged together.

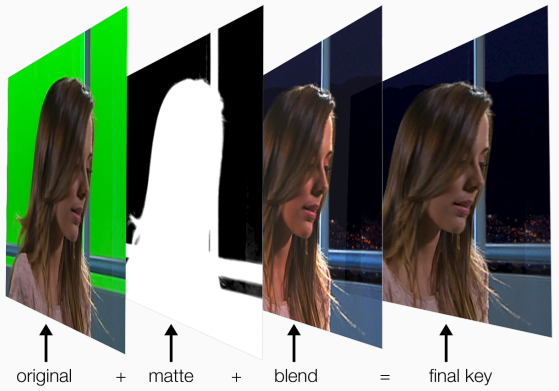
Blurring is generally a post processing effect, but it can get more complicated when involving out-of-focus (field of view) effects.

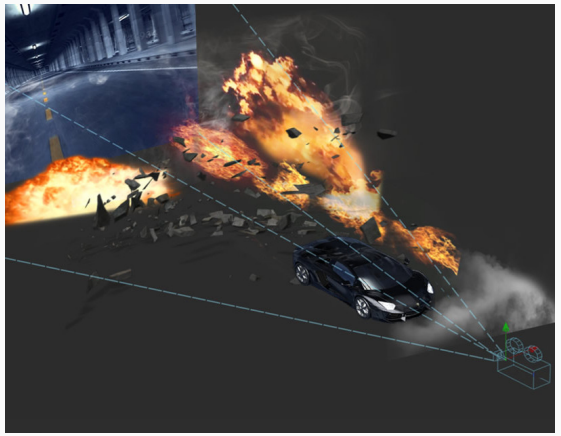


# Compositing

Compositing is mostly associated with movies and TV but it can apply to realtime visualizations and user interfaces as well. It is the process of combining graphics from multiple sources into a single image.

Green screening, color grading, atmospheric effects, etc. often fall under the umbrella of compositing.



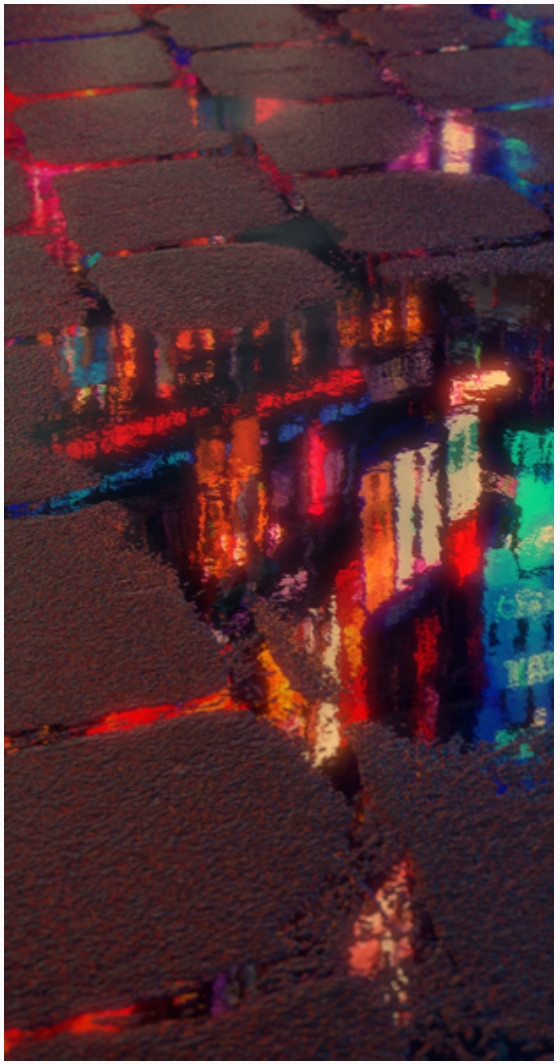



# Realism

# Reflection

We have all seen images reflected in water, mirrors, shiny things, clean tables, computer screens, the list goes on. Achieving it in a computer is actually quite difficult. Often (especially for realtime applications) reflections are just cleverly distorted panoramas slapped onto the object of question. This breaks down on any complicated surface or scene however.

There is a REAL way of doing it, we will get to that in a few slides.



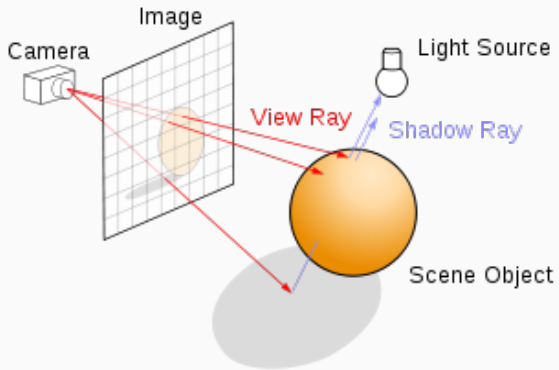
# Refraction

Refraction is even freakier and fakeir. You can *kindof* do it in real time with fancy panoramas as well, but you don't want to know how. Just trust me.

# Raytracing

So far we have been (mosty) talking about rasterization, just iterating through pixels. That is the only way to do things realtime, but if you remember physics 101 (or have even played with a flash light), you know that light doesn't work that way. It travels around in beams (or rays).

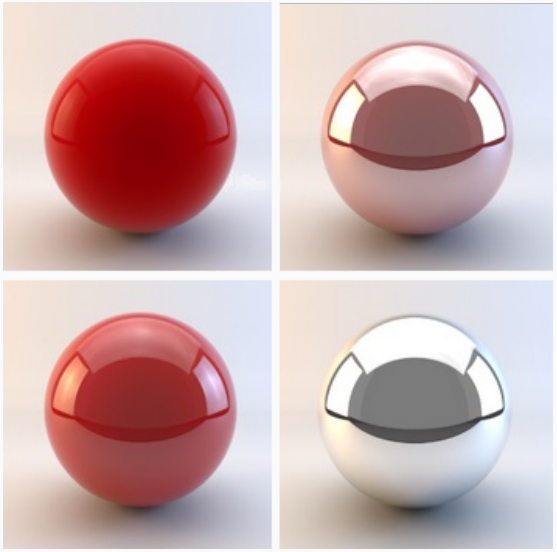
Raytracing is a technique where "rays" of light are followed backward, out of the camera and into the scene. They bounce off mirrors, refract through glass, and pick up color along the way.



# Fresnel & PBR

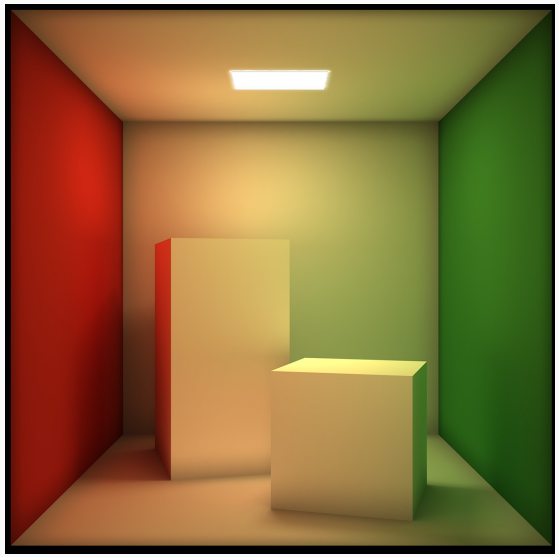
Our Phong shading model is also hopelessly unrealistic. For a start, the reflectivity of an object actually changes depending on the angle you look at it from (a property called "fresnel"). How materials scatter light is really quite complicated in general. Taking that complexity into account is called Physically Based Rendering or PBR.

It is hot stuff these days.



# Global Illumination

We have also been assuming that the photons emitted from lights are somehow magical. Really everything is a light. A brightly illuminated red wall will reflect red light onto the rest of the scene. A corner is darker than the rest of the room because it is more shielded from rouge light bouncing around. A mirror isn't perfectly reflective, but actually blurs and scatters the image slightly. Shadows aren't on-or-off, they blur towards the edges.



All of this is collectively called "global illumination" because light is a global property, it bounces and is emitted from everything, not just a few key sources.

# Pathtracing

Which brings us to pathtracing. Pathtracing is the penultimate technique. Faster and more advanced methods exist in dissertations and prototypes, but pathtracing is the only thing in this presentation that can pass as photo-realistic. It is more or less the method used today for CGI in movies, TV, advertisements, and art.

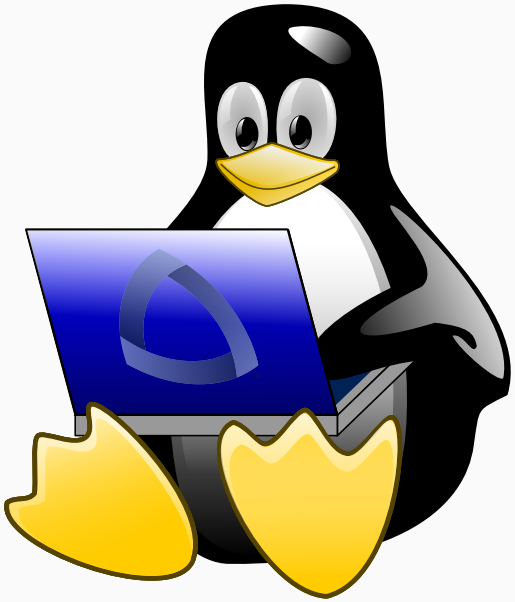
It is similar to raytracing, but instead of light bouncing directly, billions of theoretical and random paths are launched through the scene. Probibilities are calculated for each, and then after some some fancy statistics is applied, you get Iron Man.

# Wrapping it Up

# 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