

# Unit 5: Graphics

CS 3570

Shang-Hong Lai



# Graphics — Outline

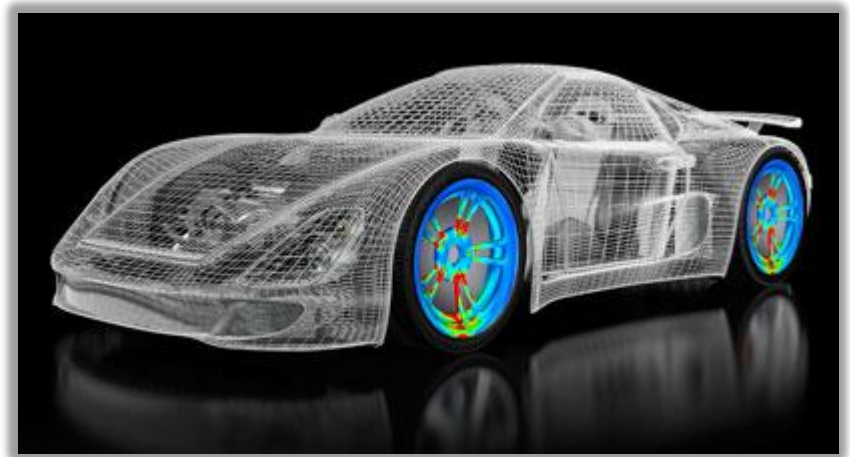
- **Vector graphics**
  - Difference between bitmap and vector graphics
  - To describe a curve — Bézier curve
- **3D Graphics**
  - 3D model
  - Lighting
- **Applications**

# Vector Graphics

- **Vector graphics** is an image file format suitable for pictures with simple components, such as
  - areas of solid
  - clearly separated colors

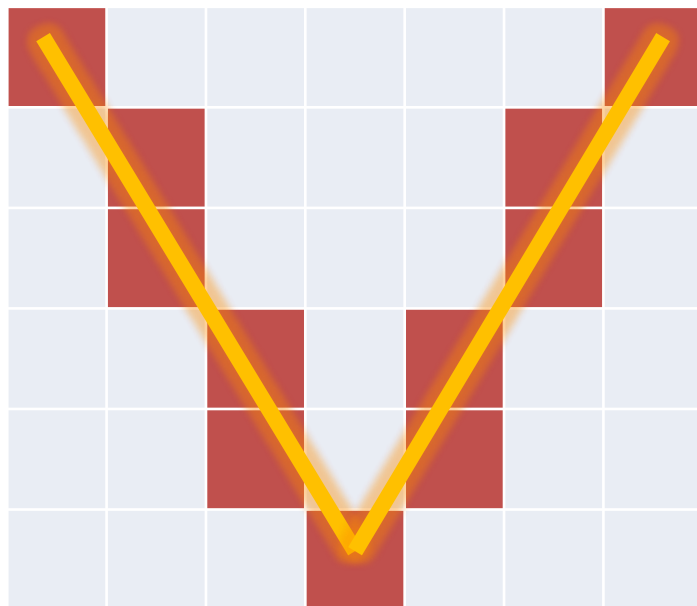
It is widely used in

- animation and games
- font design
- architectural design
- industrial design

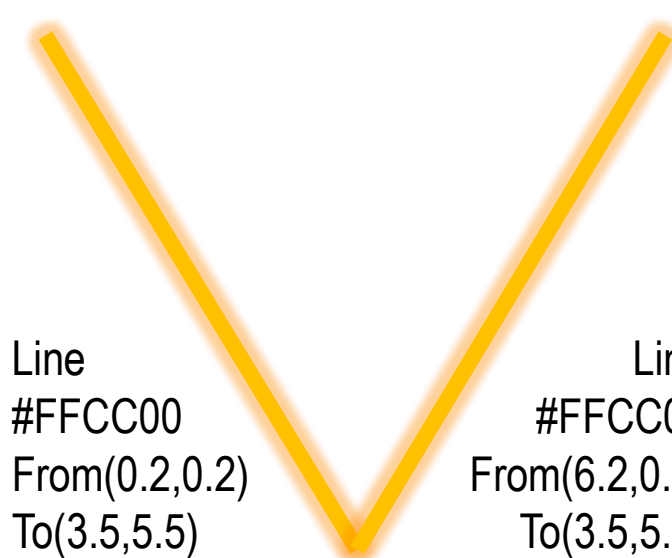


# Vector Graphics

- A bitmap saves an image **pixel by pixel**, while a vector graphic describes an object in terms of combination of **geometric shapes**.



representing a "V" using bitmap



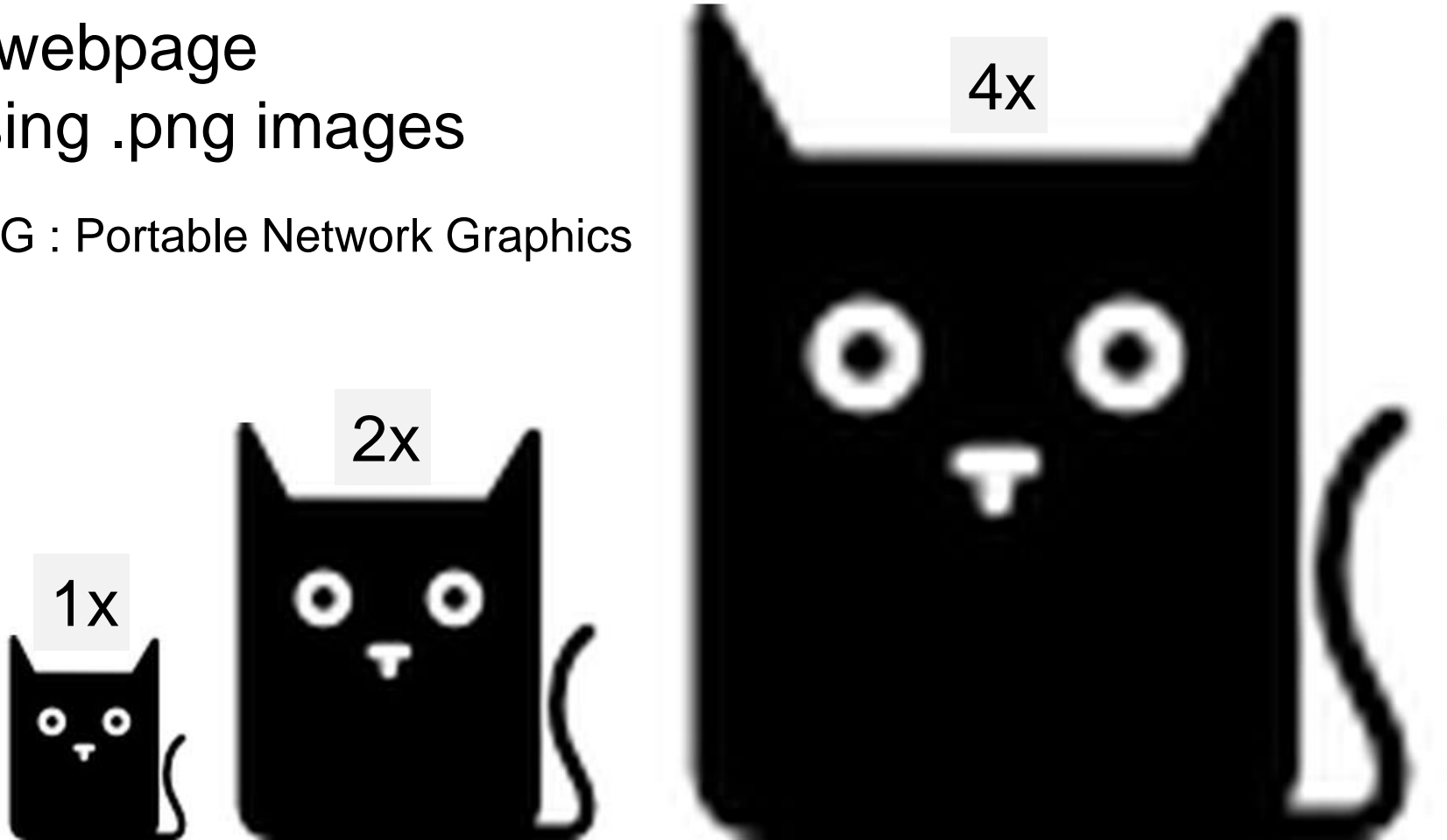
representing a "V" using vector graphic



# Vector Graphics

A webpage  
using .png images

PNG : Portable Network Graphics



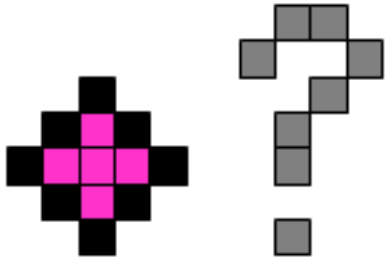
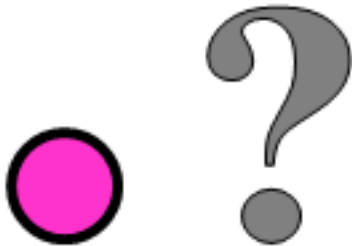
# Vector Graphics

A webpage  
using .svg images

SVG : Scalable Vector Graphics



# Bitmaps vs. Vector graphics

	Pixel	Vector
Example		
Grow and shrink	Bad	Good
Speed	Fast to create. Very hard to edit!	Slow to create. Much faster to edit.
Applications	Painter Photoshop	Power Point Illustrator



# Specifying curves

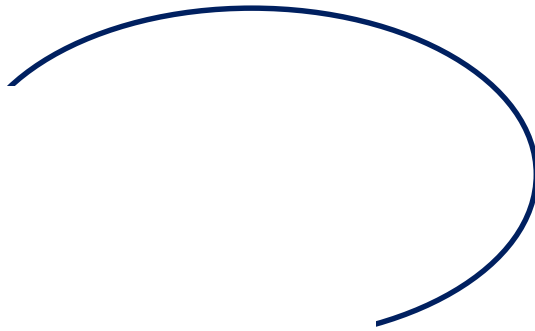
- **Explicit form of a function**  $y = f(x)$   
provide a prescription for determining the *output* value of the function  $y$  in terms of the *input* value  $x$ 
  - E.g.  $y = \sqrt{r^2 - x^2}$
- **Implicit form of the function**  $f(x, y) = 0$   
the value of  $y$  is obtained from  $x$  by *solving* an equation of the form
  - E.g.  $x^2 + y^2 - r^2 = 0$



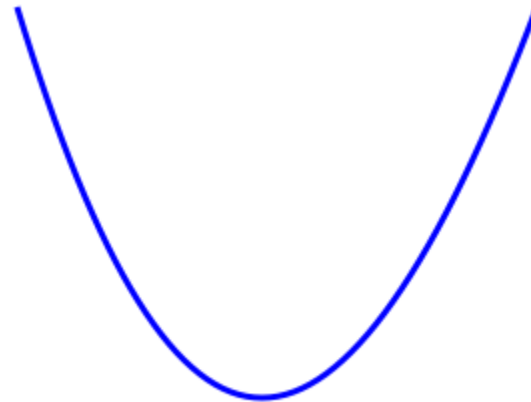


# Specifying curves

- **Straight line** is easy to describe, yet how about an curve ?



Elliptical Arc



Parabola

- More complex curve ?



# Parametric Curve

- Parametric representation of a function  
 $p(t) = (x(t), y(t))$

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x$$

$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y$$

$$P(t) = [x(t) \ y(t)] = \underbrace{[t^3 \ t^2 \ t \ 1]}_T \underbrace{\begin{bmatrix} a_x & a_y \\ b_x & b_y \\ c_x & c_y \\ d_x & d_y \end{bmatrix}}_C \quad 0 \leq t \leq 1$$

$$P = T C$$

- Curve-generating algorithms can be divided into 2 main categories: interpolation algorithms and approximation algorithms.

# Bézier Curves

- A Bézier curve is a parametric curve described by polynomials based on a sequence of **control points**
  - Degree of polynomials = number of points - 1
  - A Bézier curve passes through its first and last control points, but, in general, no others.

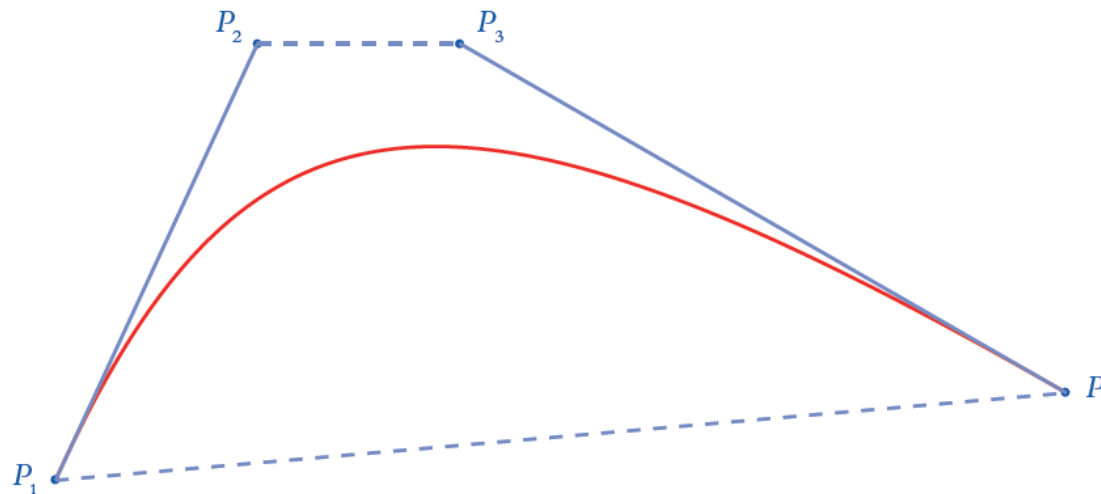
# Bézier Curves

- *The entire curve will lie within the polygon constructed by joining the control points with straight lines. This polygon is called the curve's **convex hull**.*
- The line between the starting end point and the next control point is tangent to the curve, and so is the line between the final end point and the preceding control point.

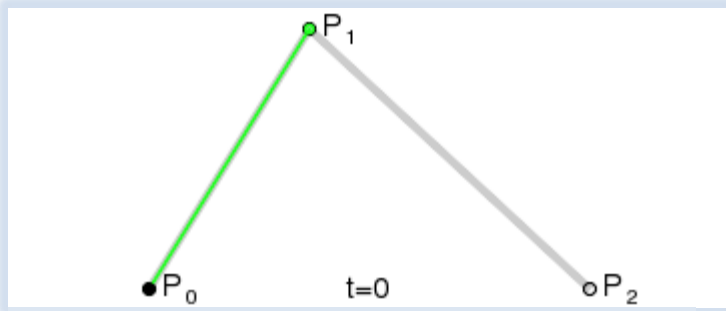


# Cubic Bézier Curves

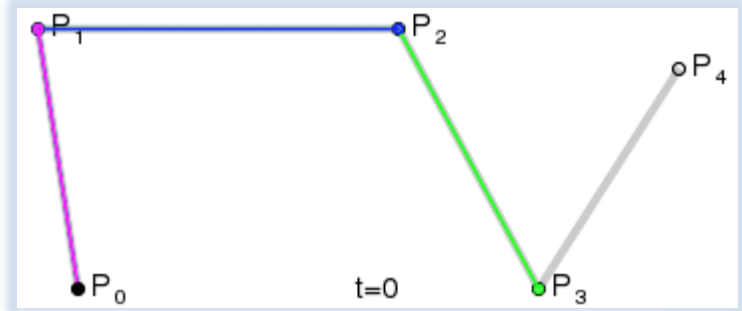
- Bézier curves of degree 3, commonly called “cubic Bézier curves”, are most commonly used in vector graphics.
- They have just four control points: two are the **end points**, and the other two are called ***direction points***.



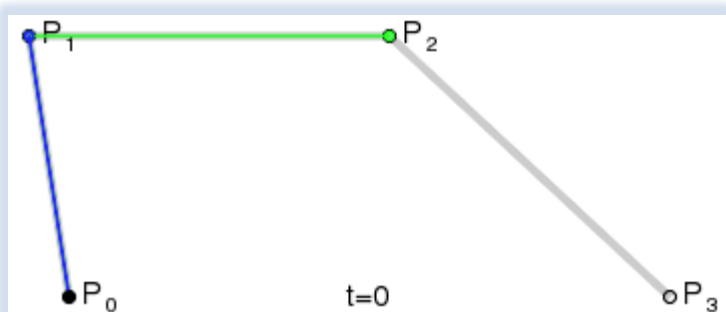
# Bézier curves - Examples



2<sup>nd</sup> order (quadratic) Bézier curve

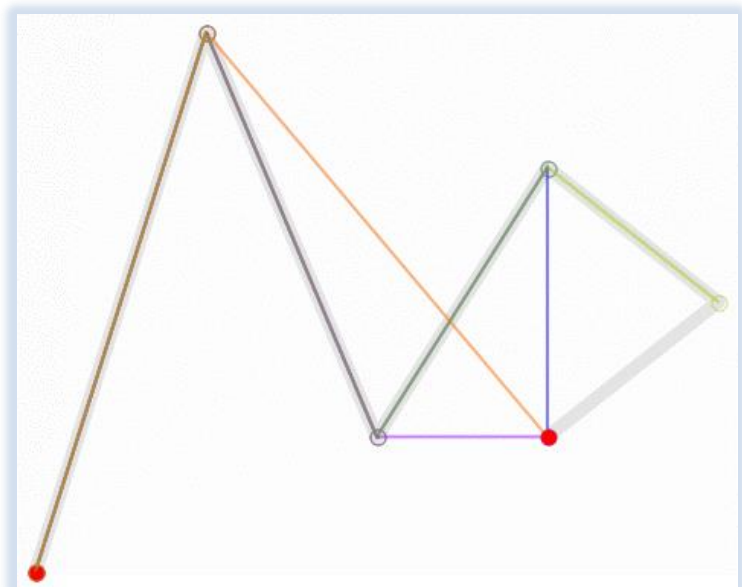


4<sup>th</sup> order (quartic) Bézier curve



3<sup>rd</sup> order (cubic) Bézier curve

[http://en.wikipedia.org/wiki/B%C3%A9zier\\_curve](http://en.wikipedia.org/wiki/B%C3%A9zier_curve)



# Bézier curve

- A Bézier curve is defined by a cubic polynomial equation
- $\mathbf{P}(t) = \mathbf{T} * \mathbf{M} * \mathbf{G}$
- $\mathbf{T} = [t^3 \ t^2 \ t \ 1]$ ,  $\mathbf{M}$  is called the **basis matrix**.  $\mathbf{G}$  is called the **geometry matrix**.

$$\mathbf{M} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{G} = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

- The blending functions are given by  $\mathbf{T} * \mathbf{M}$ ,  
 $\mathbf{P}(t) = (\mathbf{T} * \mathbf{M}) * \mathbf{G} = (1-t)^3 p_0 + 3t(1-t)^2 p_1 + 3t^2(1-t)p_2 + t^3 p_3$

# Blending Function

- Bézier curves can be described in terms of the blending functions

$$P(t) = \sum_{k=0}^n p_k \text{blending}_{k,n}(t) \quad \text{for } 0 \leq t \leq 1$$

- In the blending functions  $\text{blending}_{k,n}(t)$ ,  $n$  is the degree of the polynomial and  $k$  refers to the “weight” for the  $k$ th term in the polynomial.

$$\text{blending}_{k,n}(t) = C(n, k)t^k(1 - t)^{n-k}$$

$$C(n, k) = \frac{n!}{k!(n - k)!}$$

- For cubic polynomials,

$$\text{blending}_{0,3} = (1 - t)^3, \text{blending}_{1,3} = 3t(1 - t)^2$$

$$\text{blending}_{2,3} = 3t^2(1 - t), \text{blending}_{3,3} = t^3$$



# Bézier curves – advantage & disadvantages

- Advantage
  - Very simple
- Disadvantages
  - Expensive to evaluate the curve at many points.
  - No easy way of knowing how fine to sample points, so maybe sampling rate must vary along a curve.
  - No easy way to adapt.  
In particular, it is hard to measure the deviation of a line segment from the exact curve.

# Upscaling

- Even with some modern algorithms, upscaling a bitmap may cause some artifacts



Original  
Image

Upscaling using  
Bilinear Interpolation

Upscaling using  
S-Spline Interpolation

Ground Truth



# Curves Fitting

- If we can convert bitmap into vector graphics, we can improve the display quality of object resizing.



Input Image

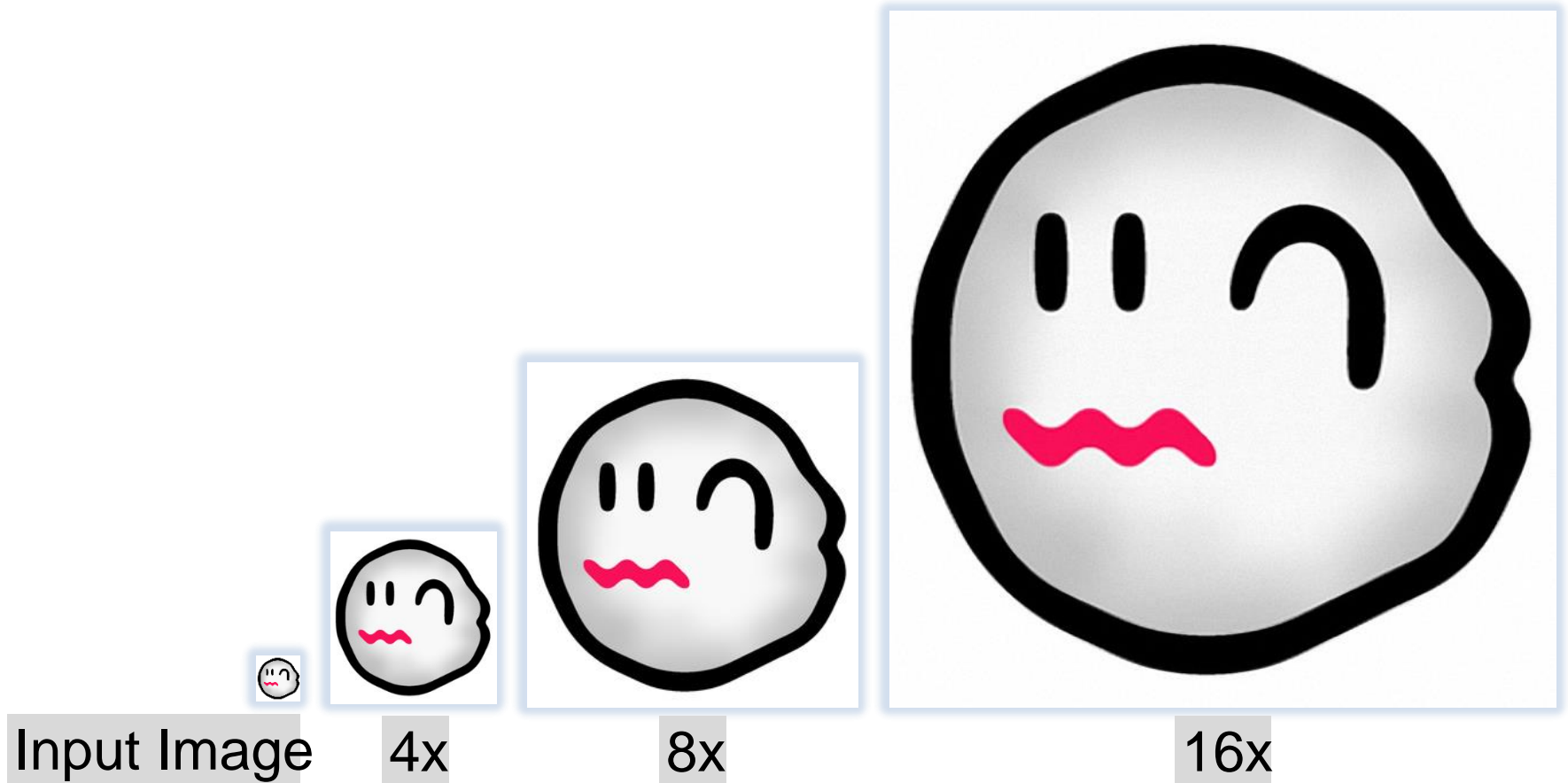


Depixelized

Johannes Kopf, Microsoft Research and Dani Lischinski, The Hebrew University. “Depixelizing Pixel Art”, SIGGRAPH 2011



# Upscaling a depixelized bitmap



# 3D Graphics

- 3D Graphics Pipeline

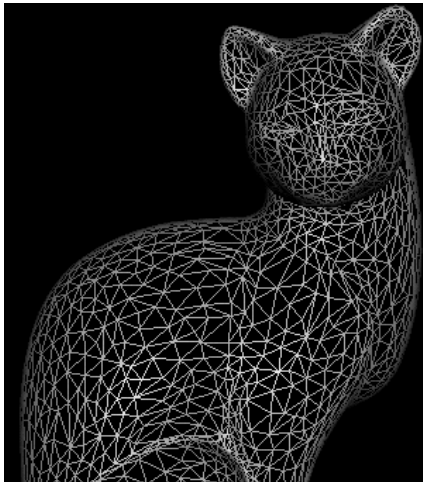
## Modeling

(Creating 3D Geometry)



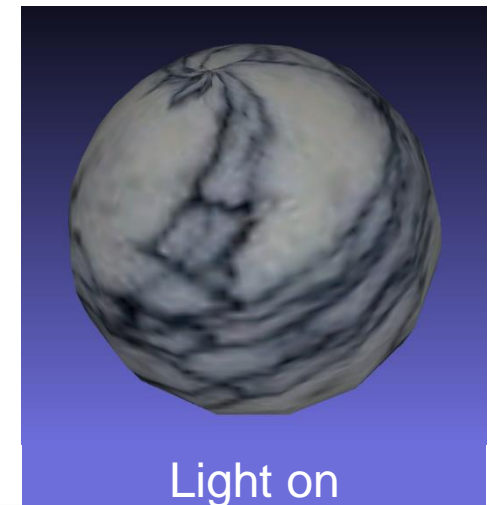
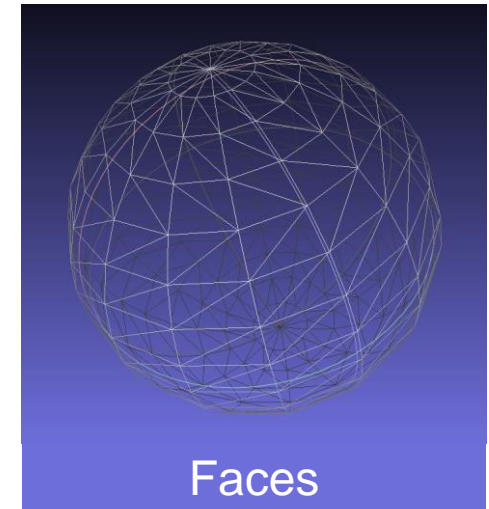
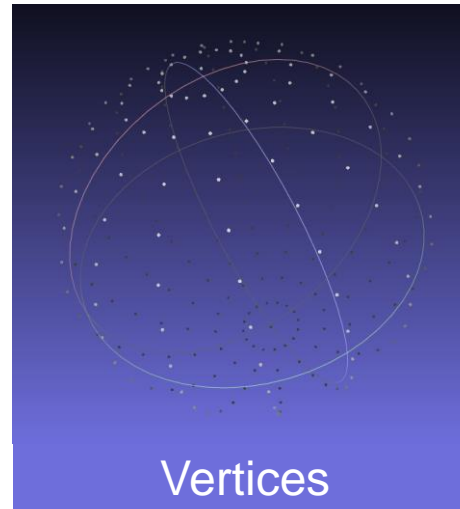
## Rendering

(Creating, shading images from geometry, lighting, materials)





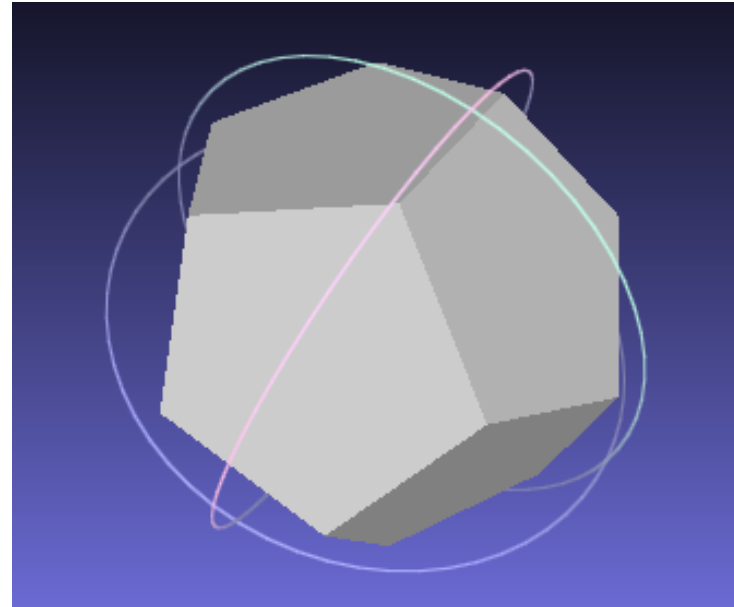
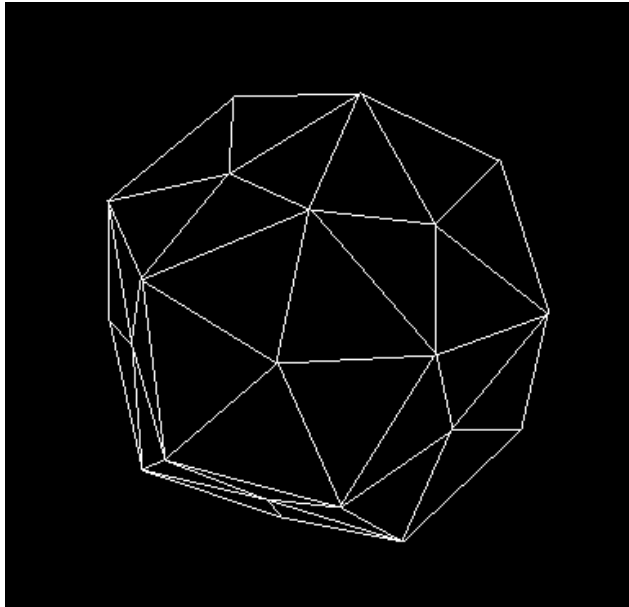
# 3D Graphics – 3D Model

- Structure
  - Vertices
  - Faces
- Transformation & Projection
- Texture
- Lighting
  - Normal Vectors
  - Material



# 3D Graphics – 3D Model

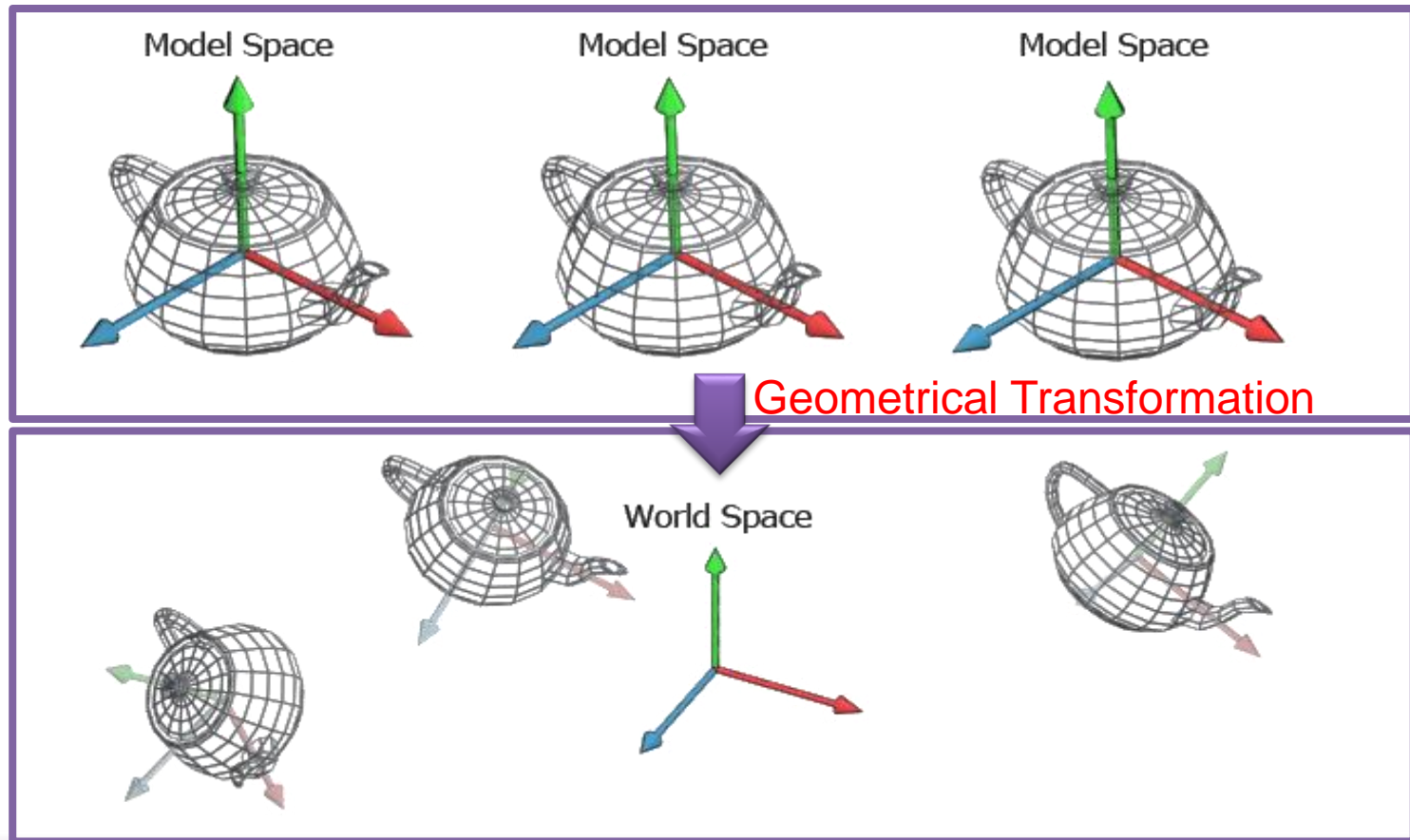
- A triangular face consists of 3 vertices 
  - While a convex quadrilateral face needs 4 
- A mesh model is composed of many faces





# 3D Graphics – Transformation

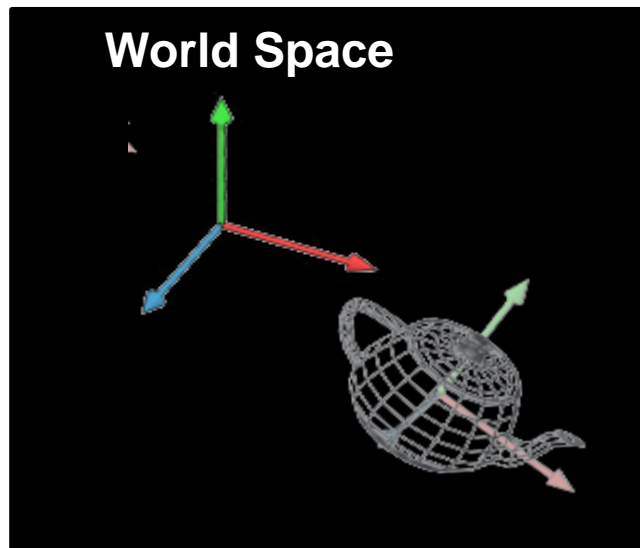
- Geometrical Transformation
  - From Model Space to World Space



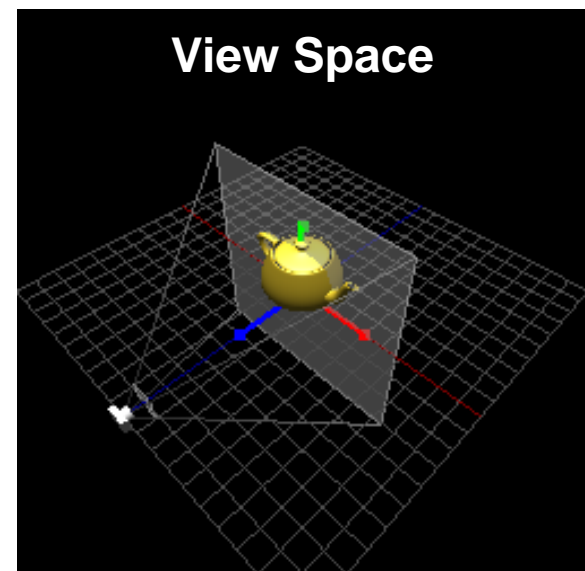


# 3D Graphics – Transformation

- Viewing Transformation
  - Form World Space to View Space (Eye Space or Camera Space)

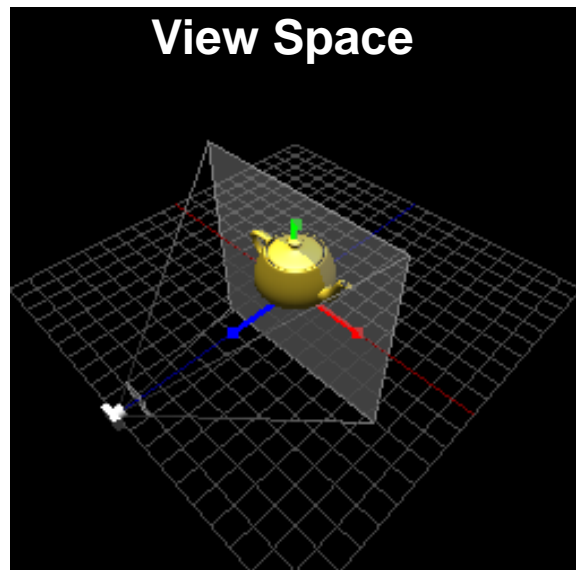


Viewing  
Transformation

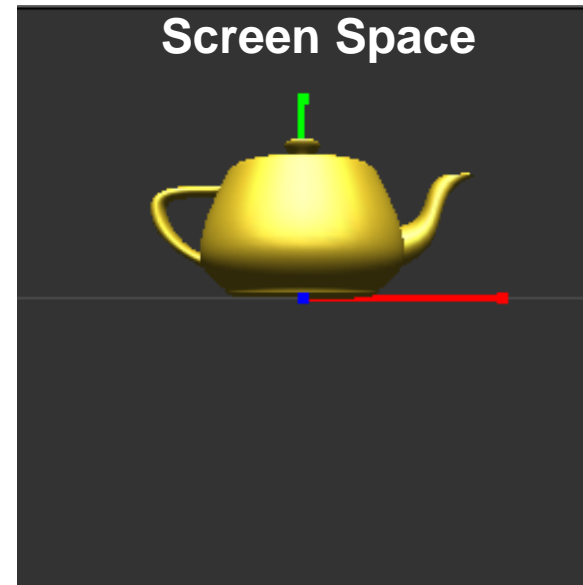


# 3D Graphics – Transformation

- Projection and Viewport Transformation
  - From View Space to Screen Space



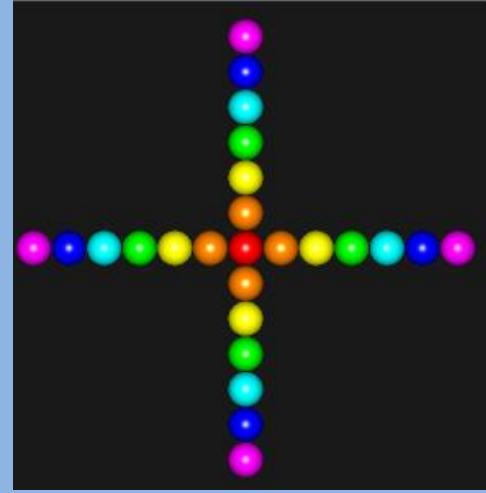
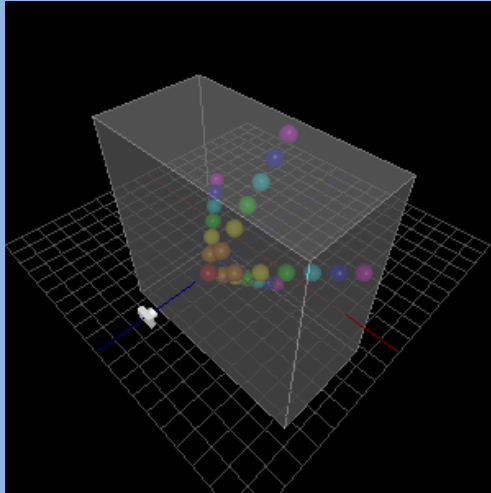
Projection



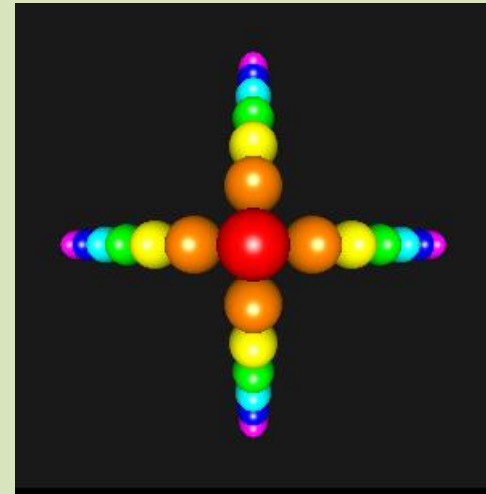
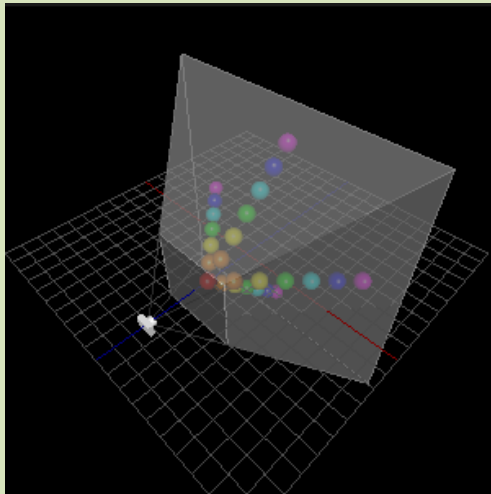
# 3D Graphics – Projection

- Parallel projection vs. perspective projection

Parallel  
projection



Perspective  
projection



# 3D Graphics – 3D Model

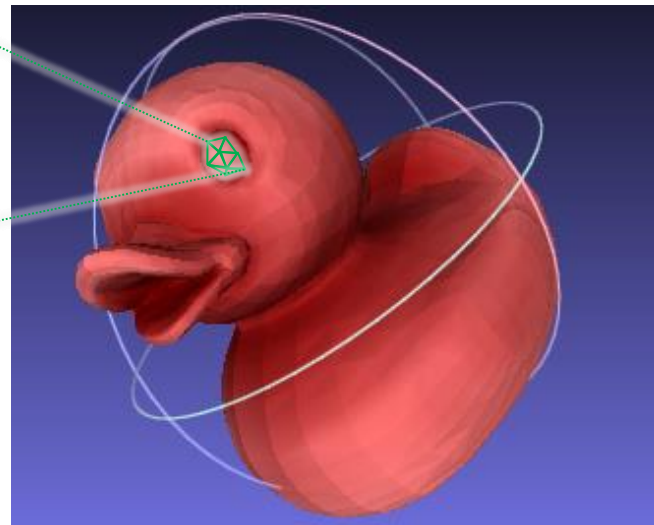
- Texture mapping

Each face of the 3-D model is mapped to corresponding triangular piece of the 2-D image.



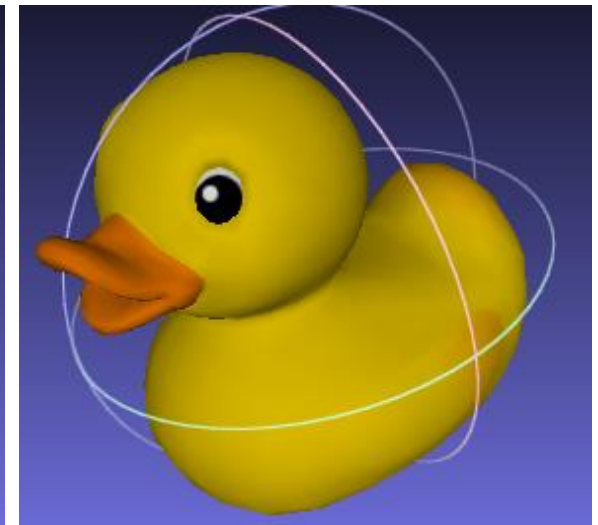
Texture image

+



Textureless model

→



Model with texture

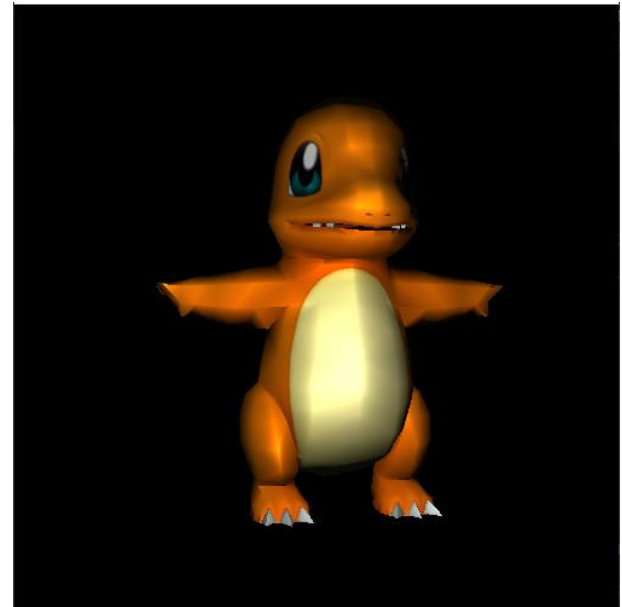
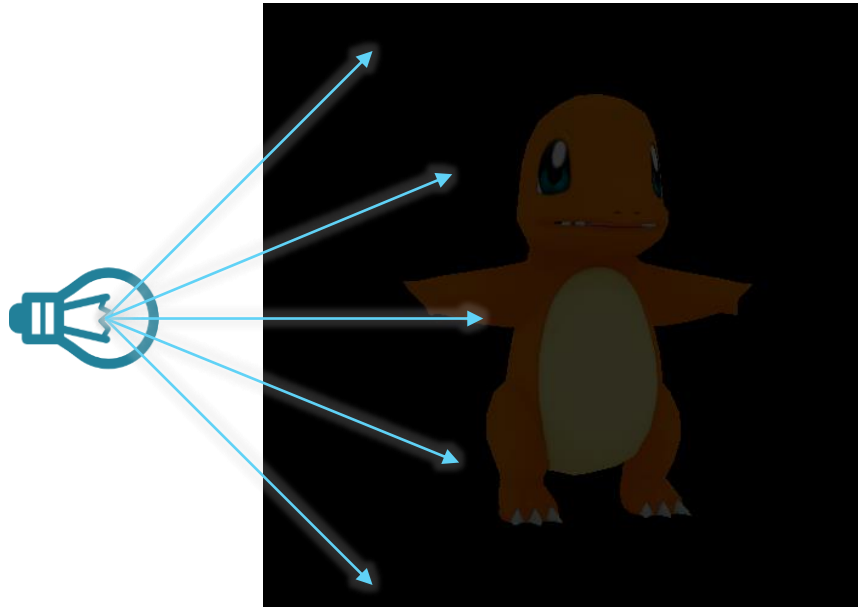


# How do we do this?



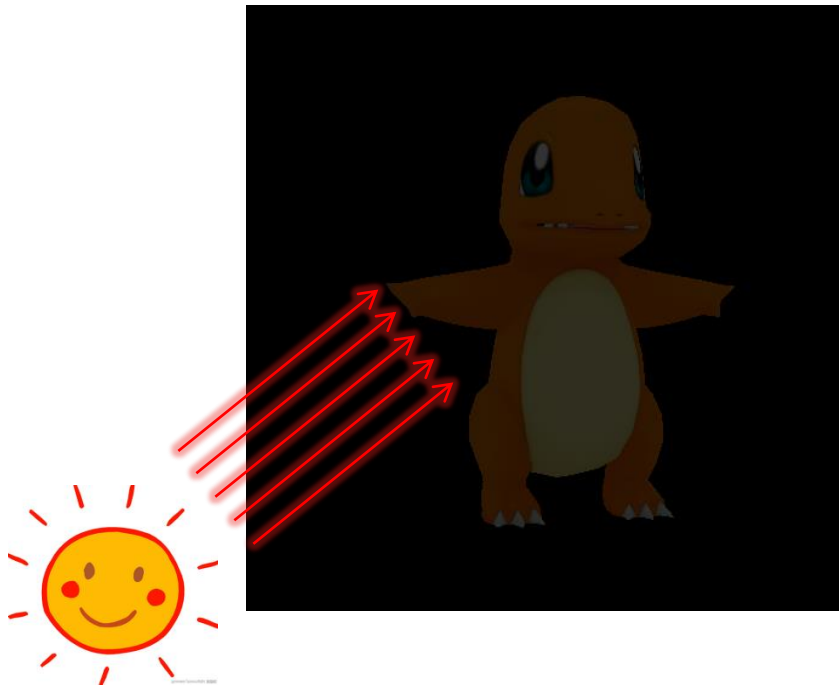
# 3D Graphics – Light Sources

- Positional light(Point Light)
  - Light source located at a specific position



# 3D Graphics – Light Sources

- Directional light (positional light at infinity)
  - Light source located at infinite far away. Such as sun.





# 3D Graphics – Light Sources

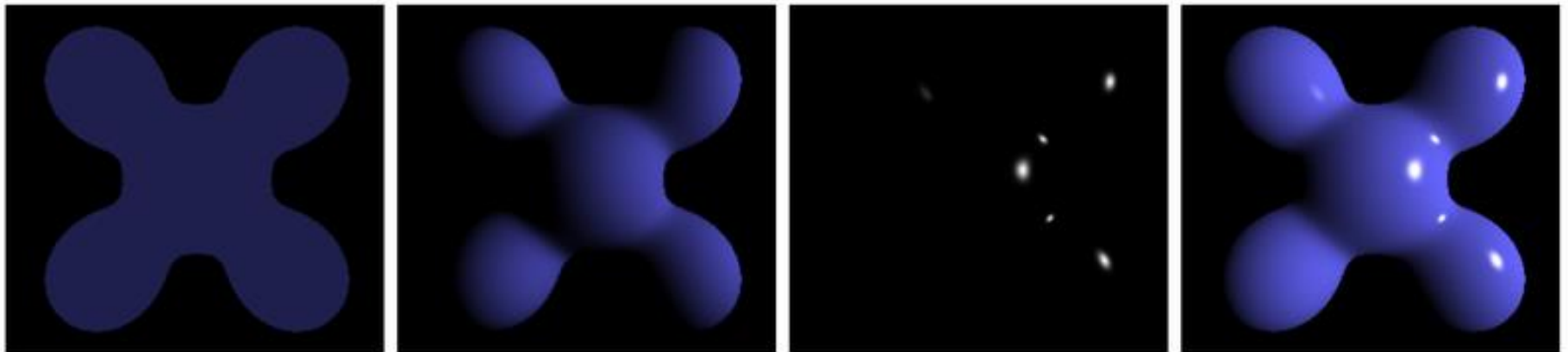
- Spot light (positional light with angle limitation)
  - Light source located at a specific position with certain cutoff range.





# 3D Graphics – Lighting Equation

- **Intensity = Ambient + Diffuse + Specular**



Ambient + Diffuse + Specular = Phong Reflection



# 3D Graphics – Ambient

- Ambient
  - Illumination surrounding a scene without providing any specific light source.

$$I = I_a \times k_a$$

$I$  : resulting intensity

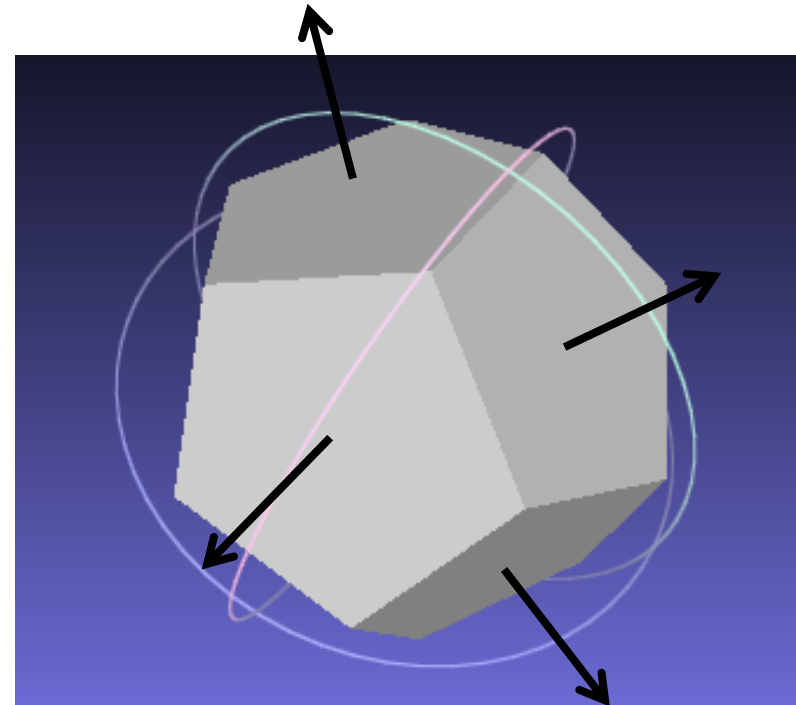
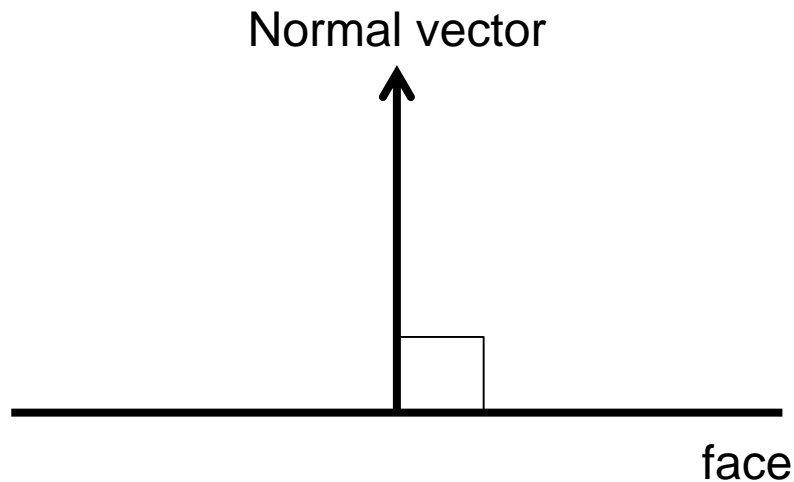
$I_a$  : ambient light intensity

$k_a$  : ambient reflection coefficient



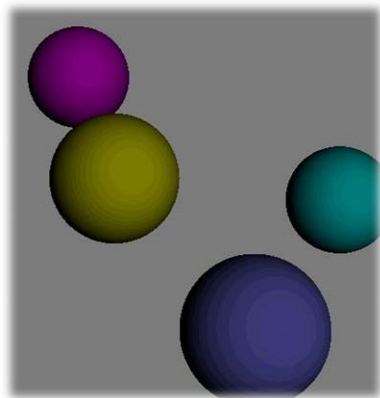
# 3D Graphics – Surface Normal

- Each face has a normal vector.

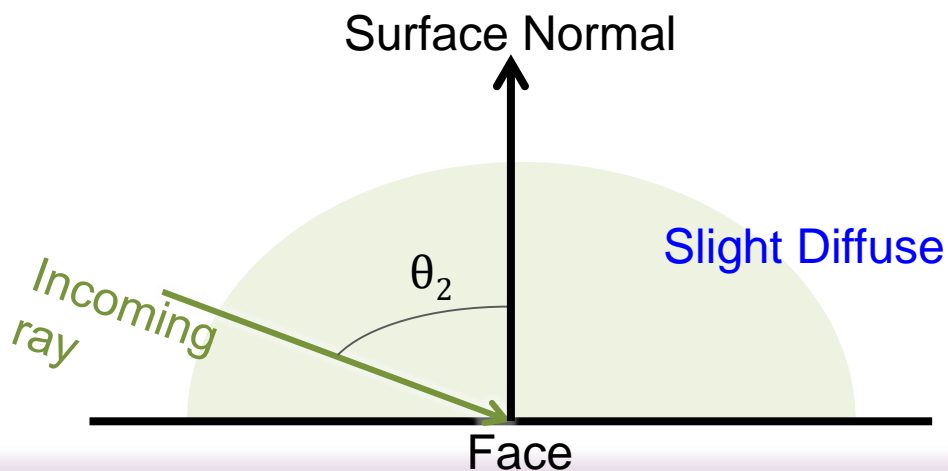
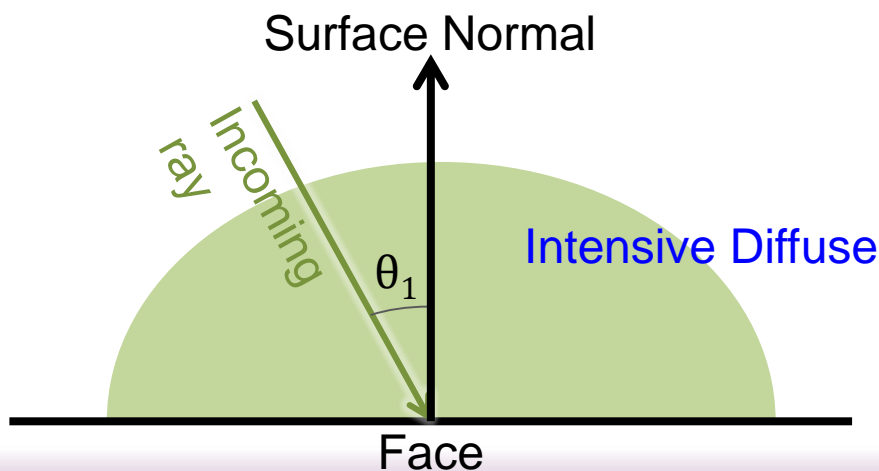


# 3D Graphics – Diffuse

- Diffuse reflection intensity :  $I_p \times k_d \times \cos^+ \theta$

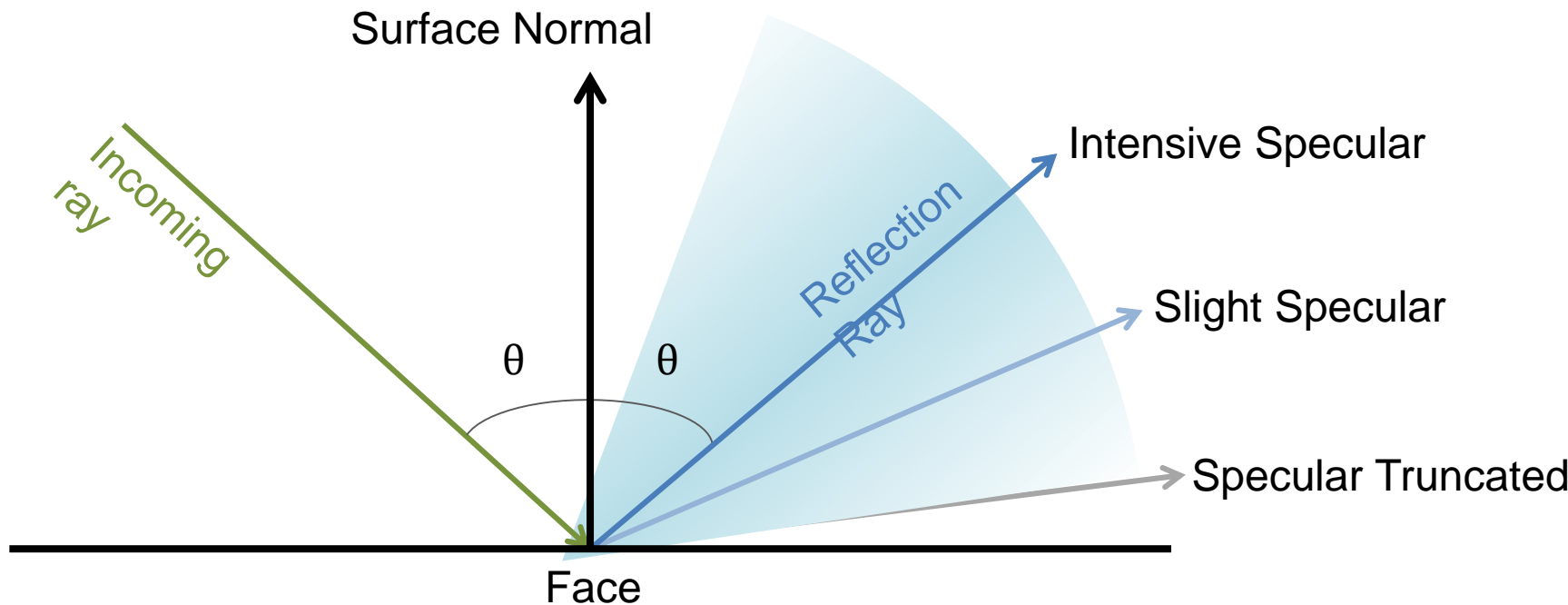


$I_p$  : point light source intensity  
 $k_d$  : diffuse reflection coefficient  
 $\cos^+ \theta$  :  $\max(0, \cos \theta)$



# 3D Graphics – Specular

- Specular reflection



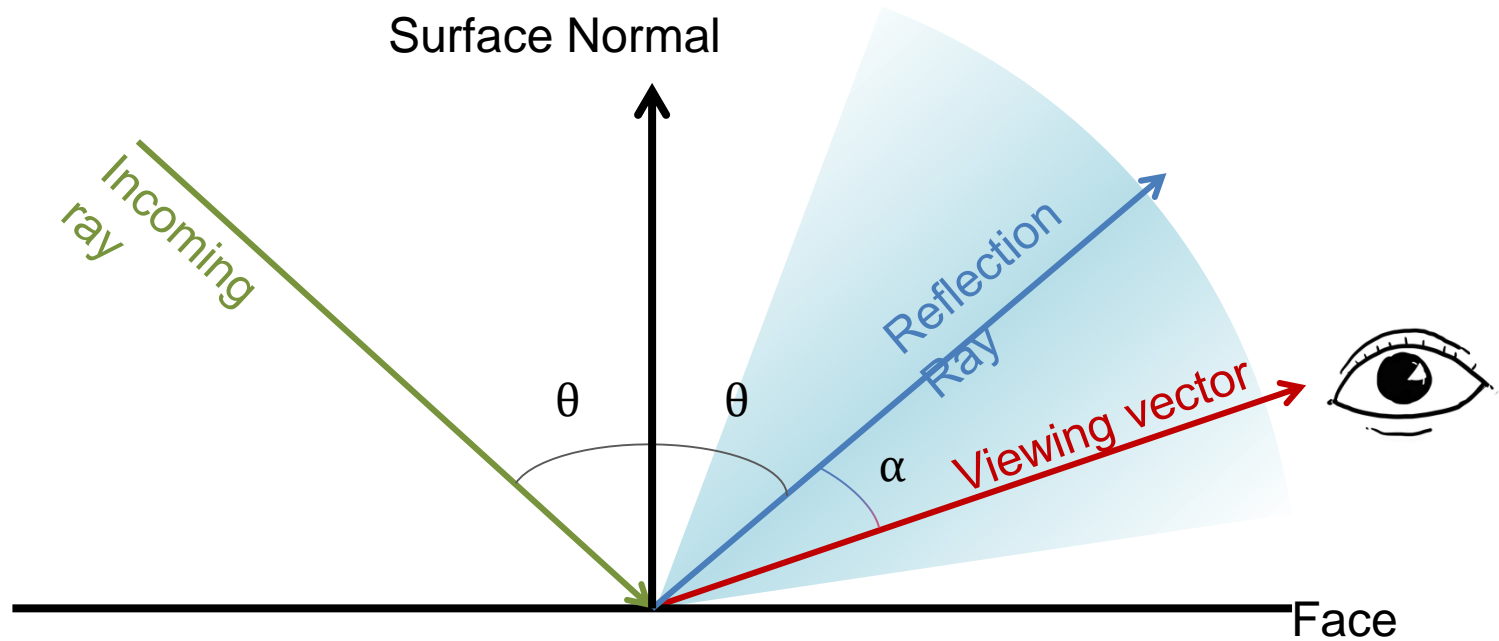
# 3D Graphics – Specular

You will see the lighting intensity :  $I_p \times k_s \times (\cos \alpha)^n$

$I_p$  : Light source specular intensity

$k_s$  : specular reflection coefficient

$n$  : a positive parameter



# 3D Graphics – Specular

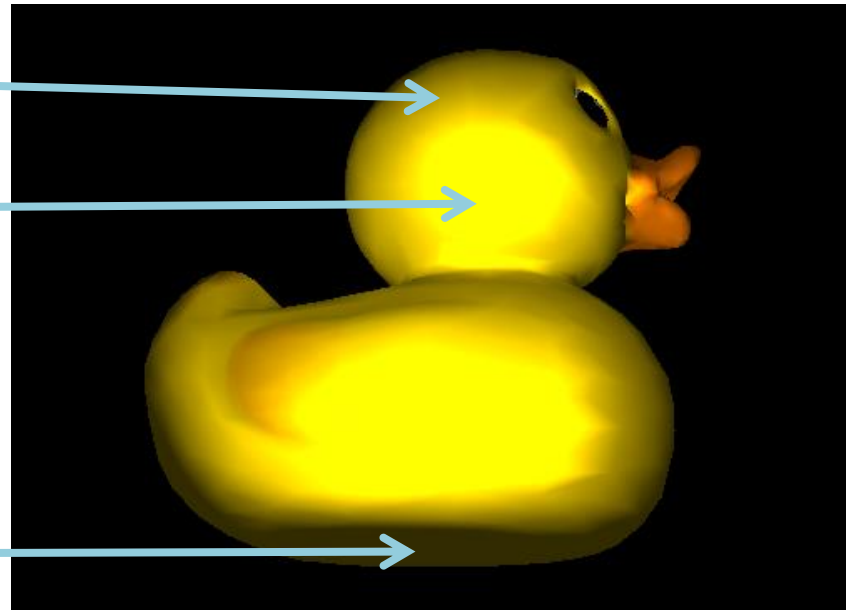
You will see the lighting intensity :  $I_p \times k_s \times (\cos \alpha)^n$

The bigger  $|\alpha|$  , the smaller  $I_p \times k_s \times (\cos \alpha)^n$

$$|\alpha| > 0$$

$$|\alpha| \doteq 0$$

$$|\alpha| \gg 0$$



# 3D Graphics – Phong model

- Phong model

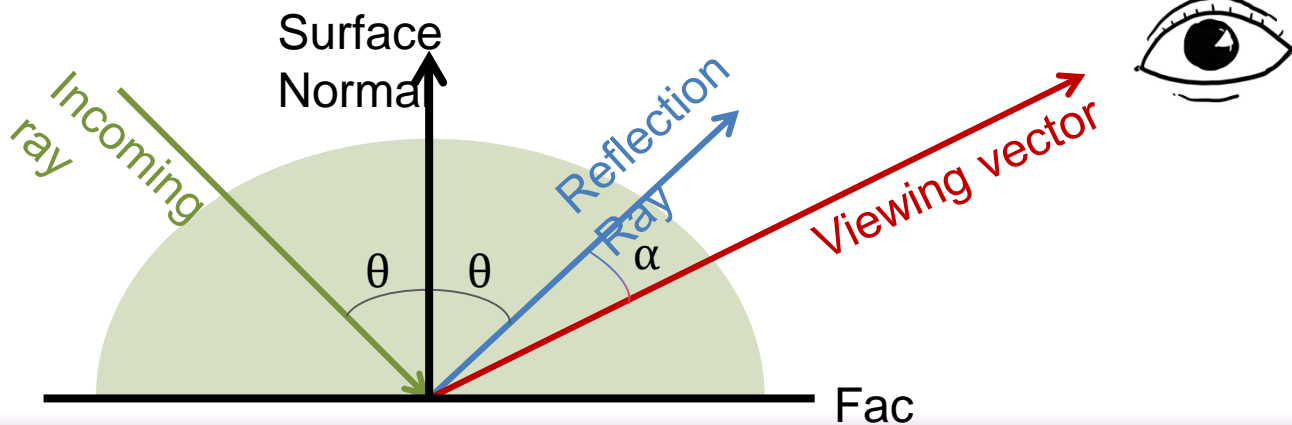
- Intensity = Ambient + Diffuse + Specular

- $\Rightarrow (I_a \times k_a) + (I_p \times k_d \times \cos \theta) + (I_p \times k_s \times (\cos \alpha)^n)$

Ambient

Diffuse

Specular

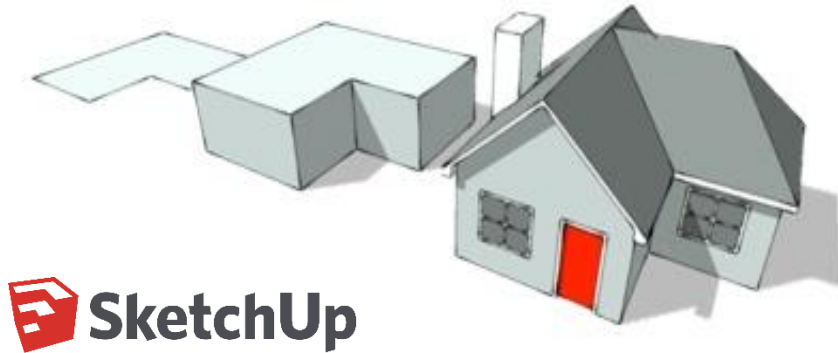




# 3D Face Modeling Example



# CG Tools



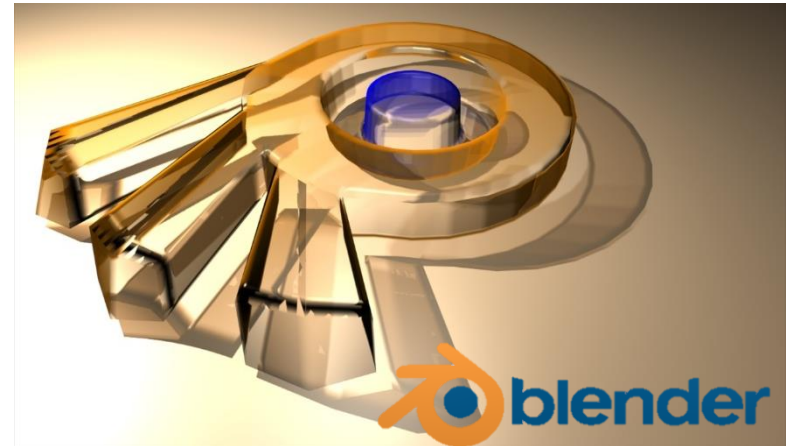
(Free trial)



**MAYA**



**WINGS 3D**



# Input Devices

- Locator Devices
- Keyboard
- Scanner
  - Images
  - Laser
- Cameras (research)



# Locator Devices

When queried, locator devices return a position and/or orientation.

- Mouse (2D and 3D)
- Trackball
- Joystick (2D and 3D)



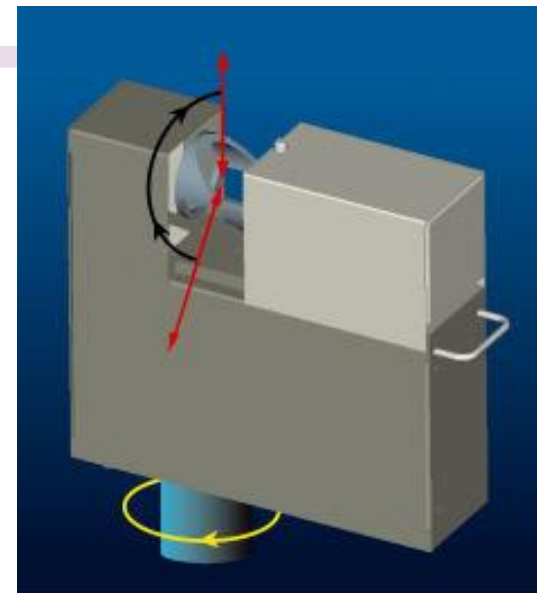
# Locator Devices

- When queried, locator devices return a position and/or orientation.
- Tablet
- Virtual Reality Trackers
- Data Gloves
- Digitizers



# Scanners

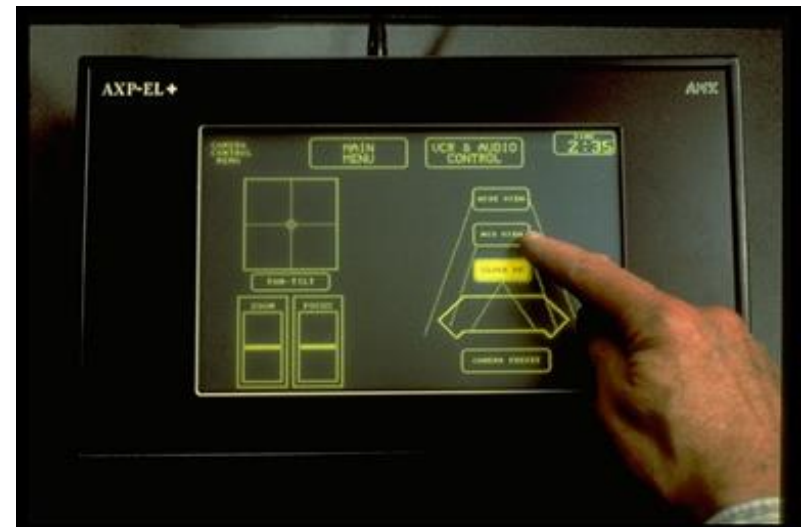
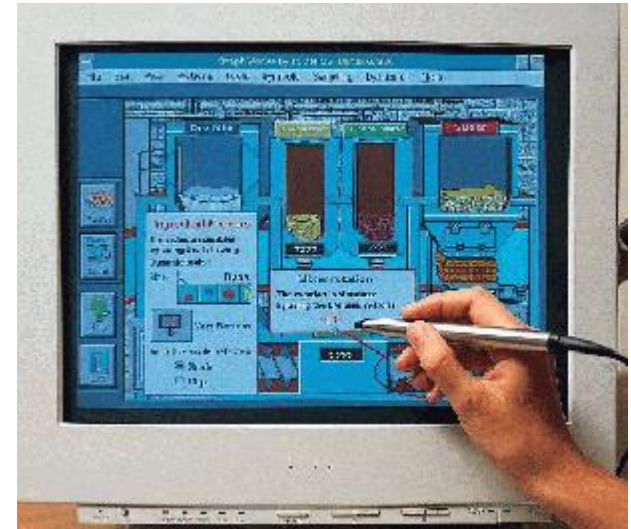
- Image Scanners - Flatbed, etc.
  - What type of data is returned?  
Bitmap
- Laser Scanners - Deltasphere
  - Emits a laser and does time of flight. Returns 3D point
- Camera (image) based
  - Examine camera image(s) and try to figure out vertices from them.





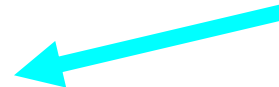
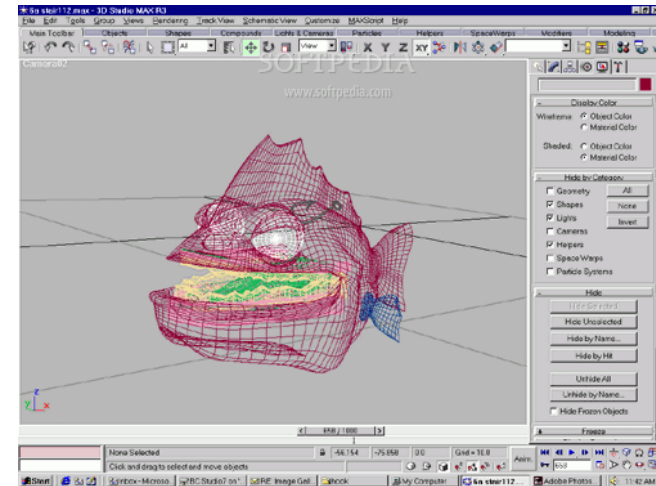
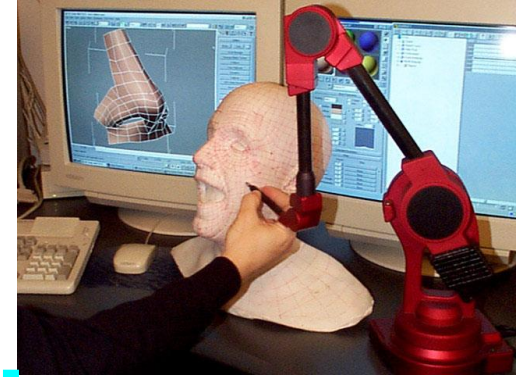
# Many others

- Light Pens
- Voice Systems
- Touch Panels
- Camera/Vision Based
- Which is best?



# Common Modeling Approach

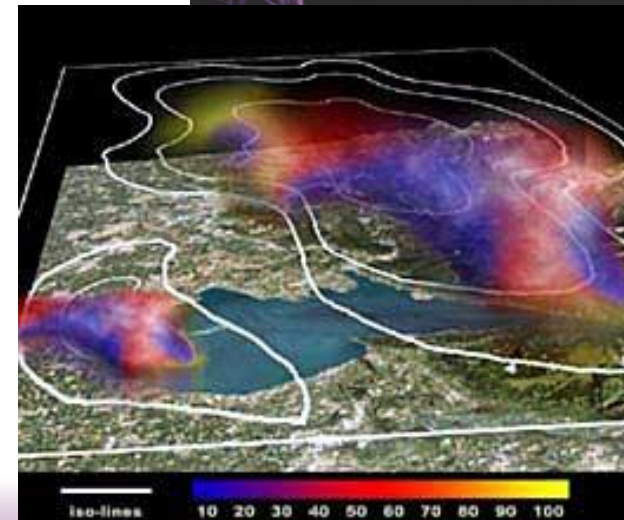
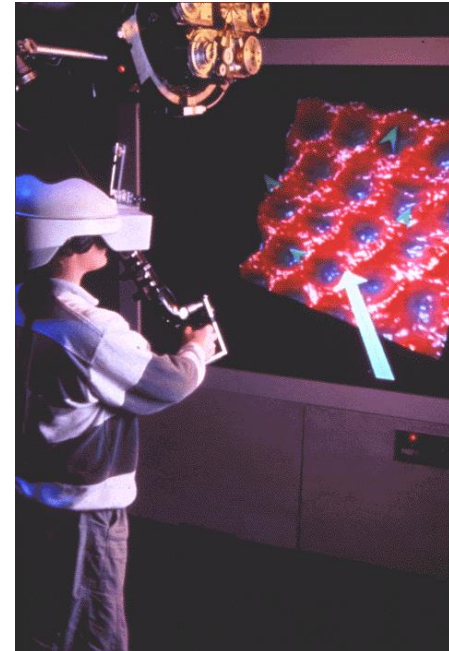
- Hybrid
- Animator jobs





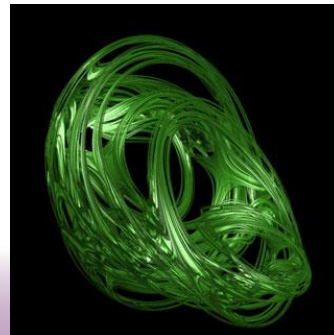
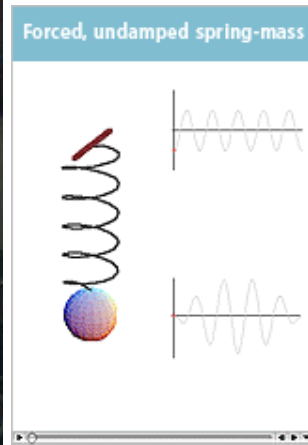
# Computer Graphics Applications

- Virtual Reality
  - VR: User interacts and views with a 3D world using “more natural” means
- Data Visualization
  - Scientific, Engineering, Medical data
  - Visualizing millions to billions of data points
  - See trends
  - Different schemes



# Computer Graphics Applications

- Education and Training
  - Models of physical, financial, social systems
  - Comprehension of complex systems
- Computer Art
  - Fine and commercial art
  - Performance Art
  - Aesthetic Computing
  - SIGGRAPH



# Computer Graphics Applications

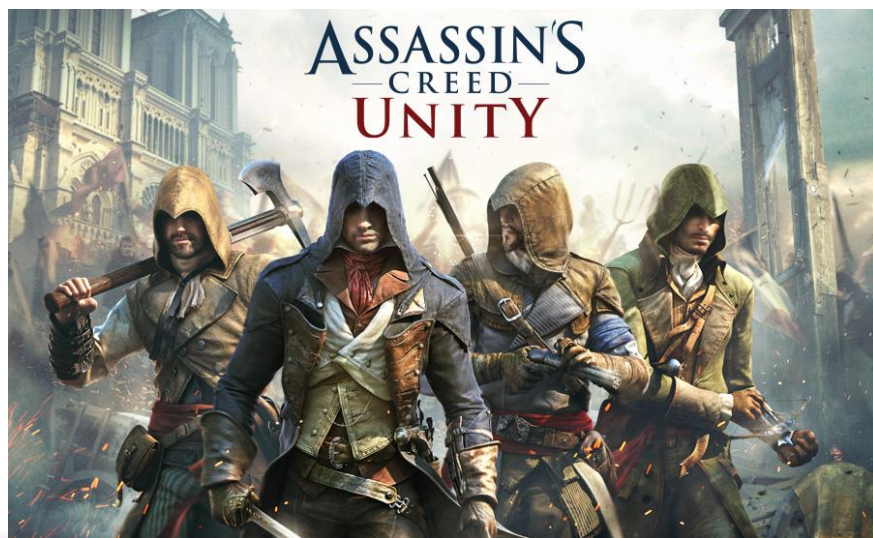
- Games/Movies





# Augmented Reality

- It's computationally expensive to render a 3-D scene in real-time
- To render these vivid 3D scenes, a high-end gaming PC is necessary



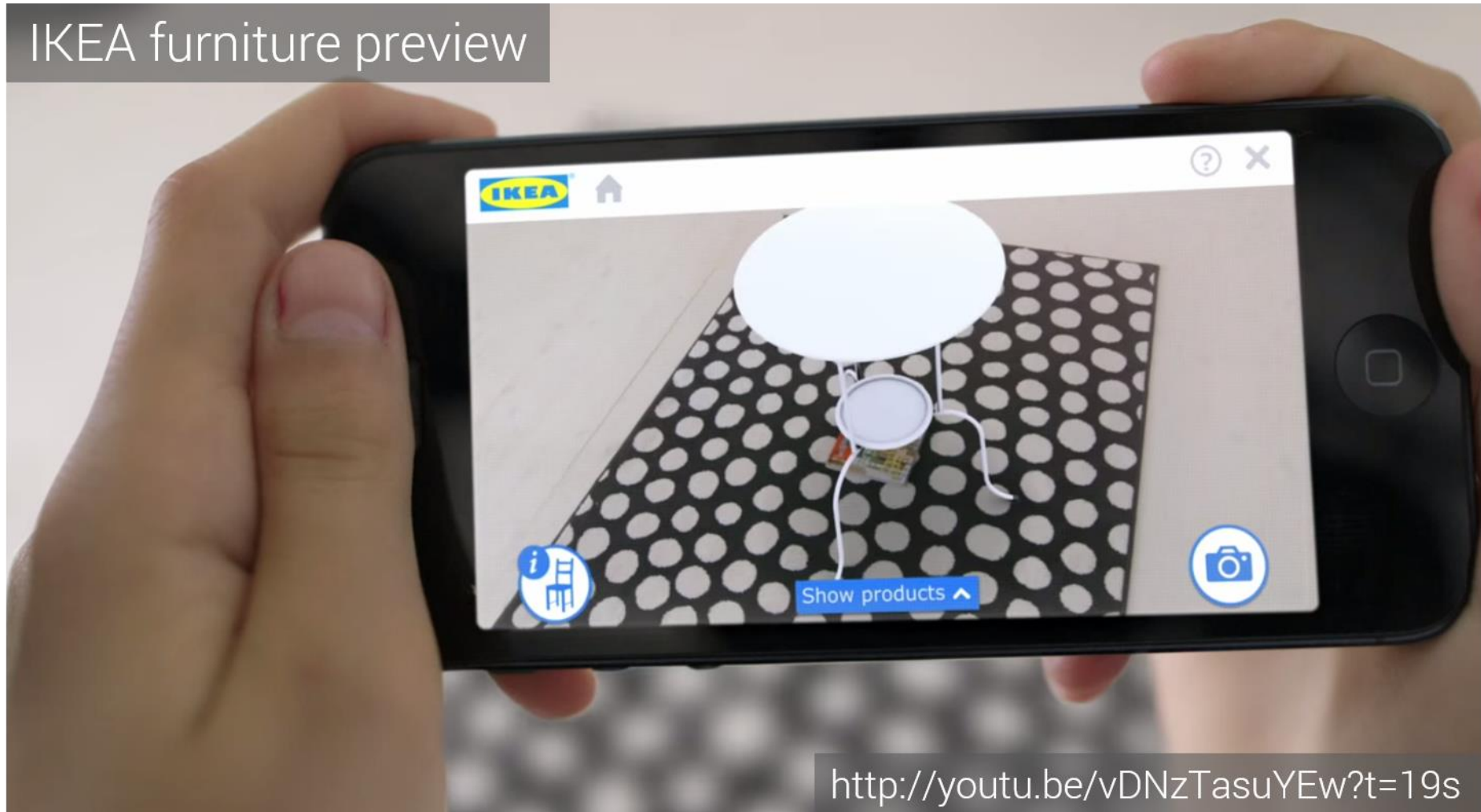
# Augmented Reality

- It's computationally expensive to render a 3-D scene in real-time
- To render these vivid 3D scenes, a high-end gaming PC is necessary
- Even if a 3-D scene is rendered, it's difficult to interact with objects inside
- What if we render some virtual objects in real world ?



# Augmented Reality

IKEA furniture preview



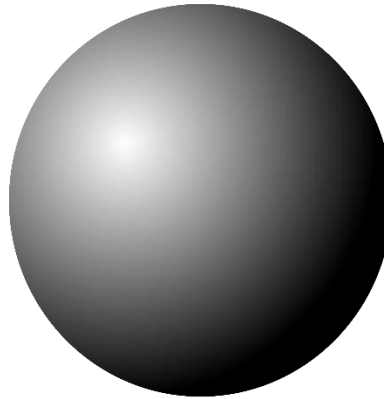
<http://youtu.be/vDNzTasuYEw?t=19s>



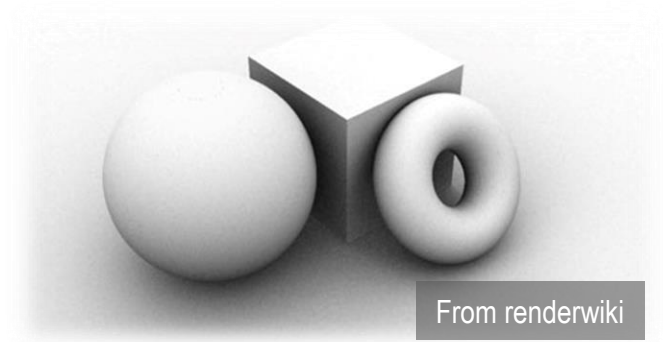
# 3D perception — Depth



Size and  
Vanishing point



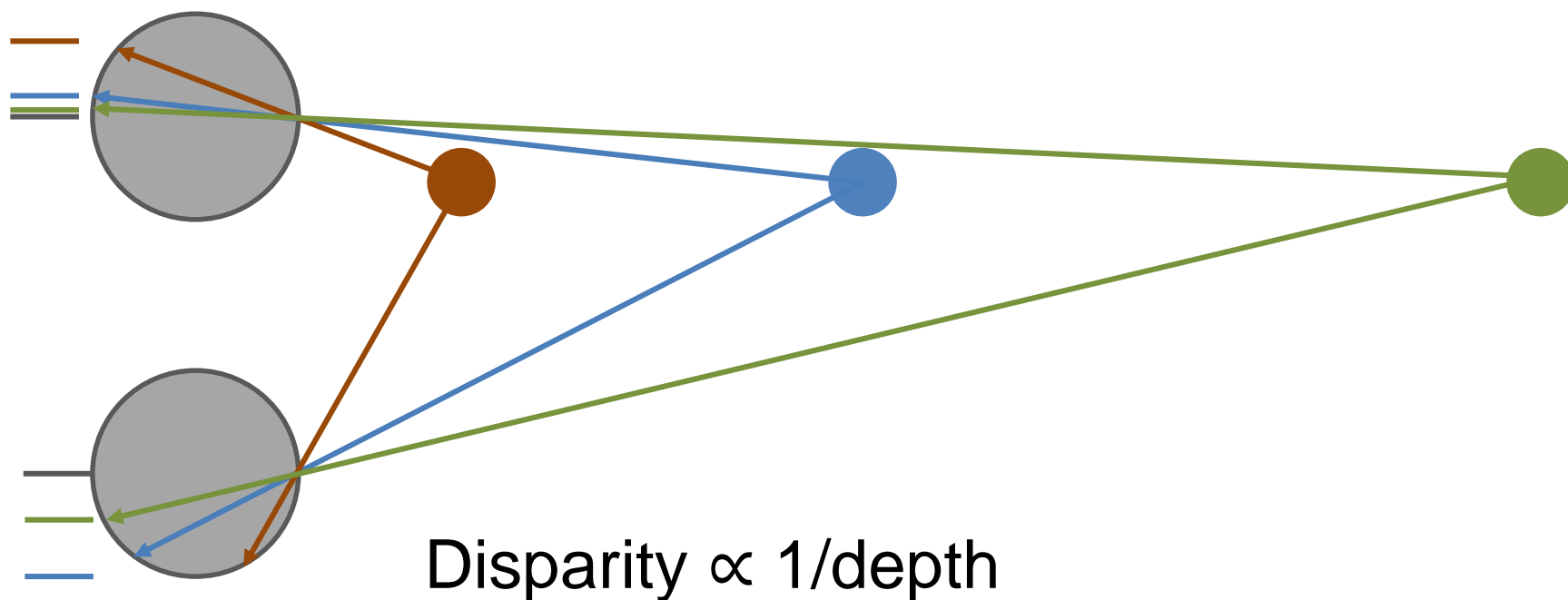
Light and Shadow



Occlusion



# Stereo 3D perception



The moon seems to follow me, but the street lamp doesn't.

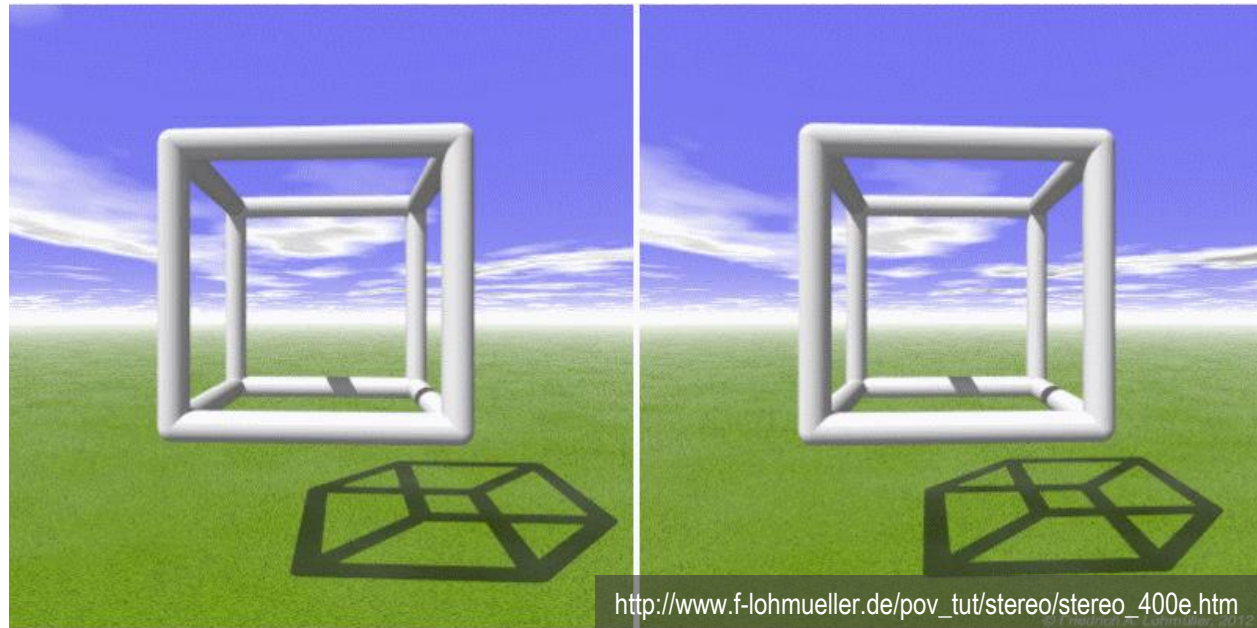




# Stereo Camera and Stereo Images



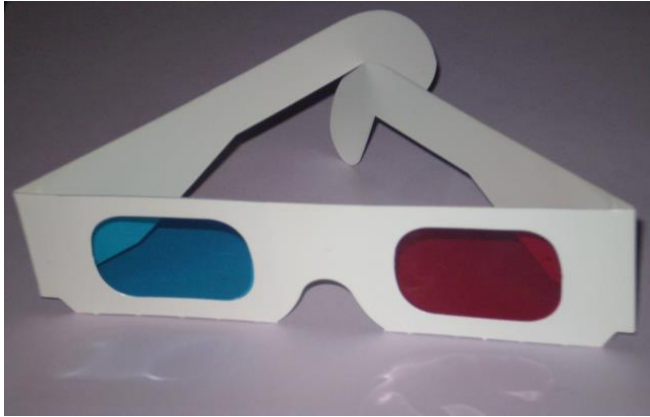
Stereo Camera



Stereo Images



# 3D movies & 3D TV



Anaglyph 3D glasses



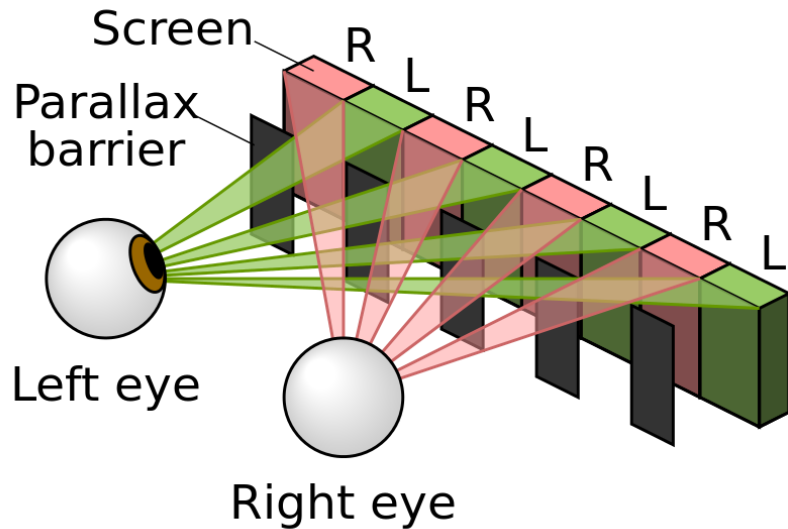
Polarized glasses



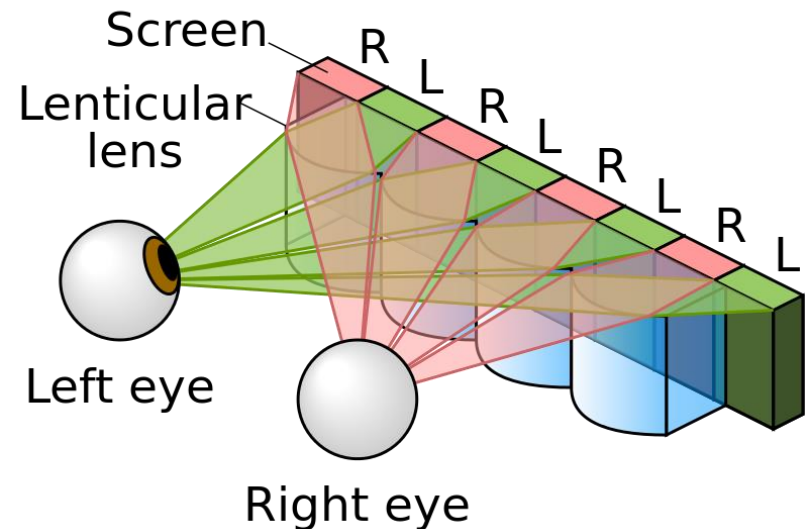
Shutter glasses



# Autostereoscopy (Glass-less)



Parallax barrier



Lenticular screen

# Summary

---

- 2D Vector Graphics
- 3D Graphics
  - 3D modeling
  - Image rendering
- Applications

