Real time realistic rendering

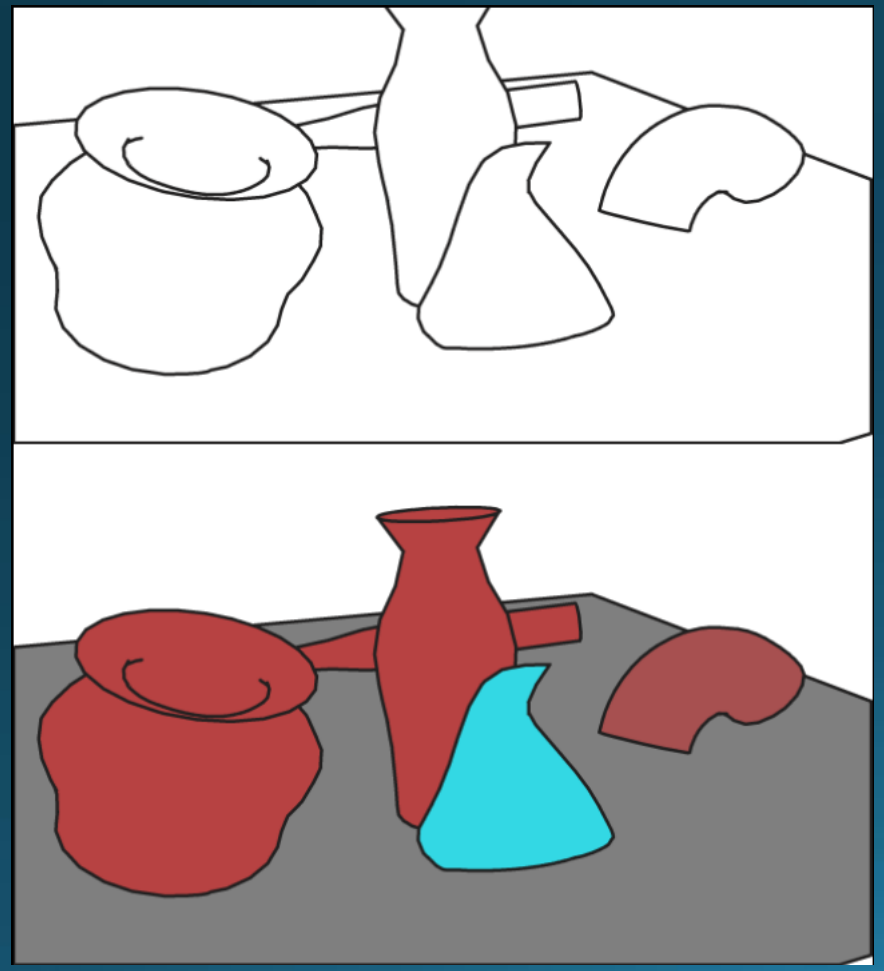
Yevhen Tyshchenko

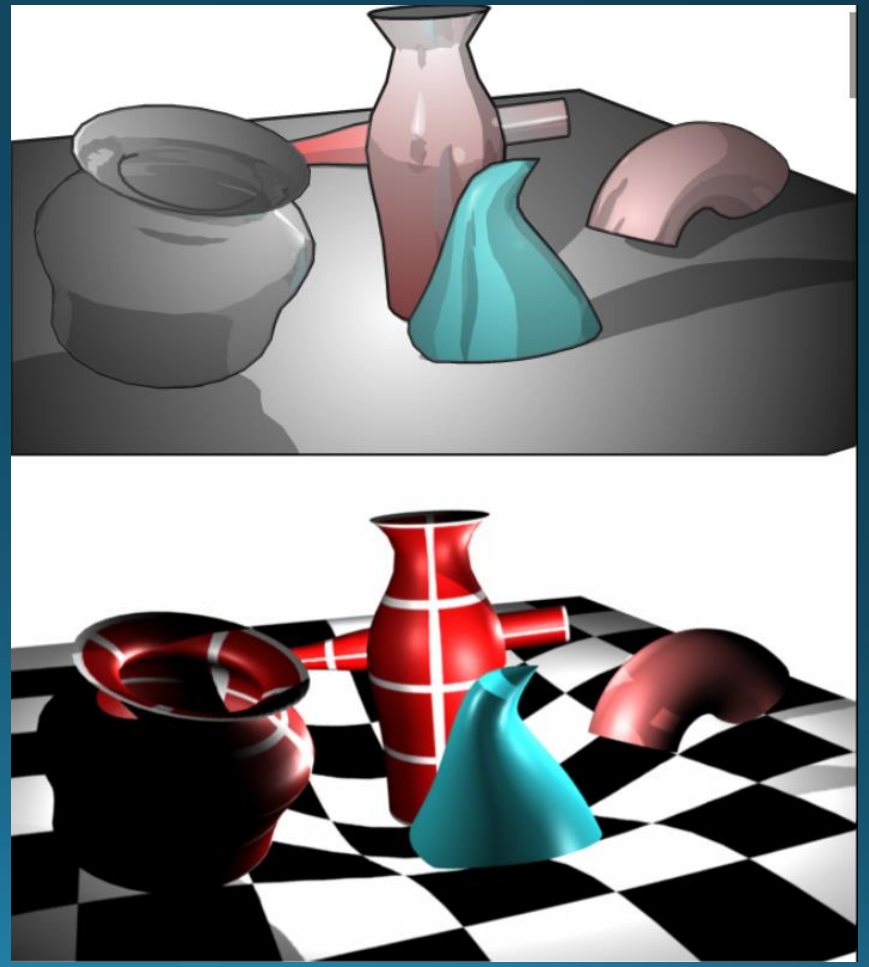
Plan

1) What is rendering?
   - Goals
   - Features
2) Rendering techniques
   - Computer graphics pipeline architecture
   - Rasterizer
   - Ray tracing & ray casting
   - Radiosity
3) Unreal Engine 4 example

Rendering is

the process of generating an image from the 2D or 3D model.





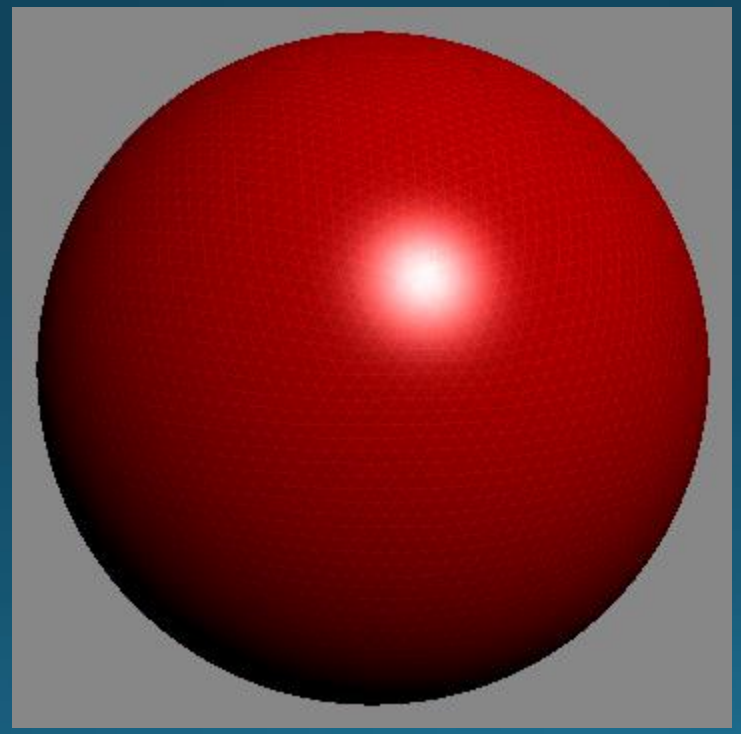
Goals

- ***Interactivity***: impossible to predict how a player will interact with the game environment.

- ***Speed:*** fluid motion – minimum of 18-20 fps

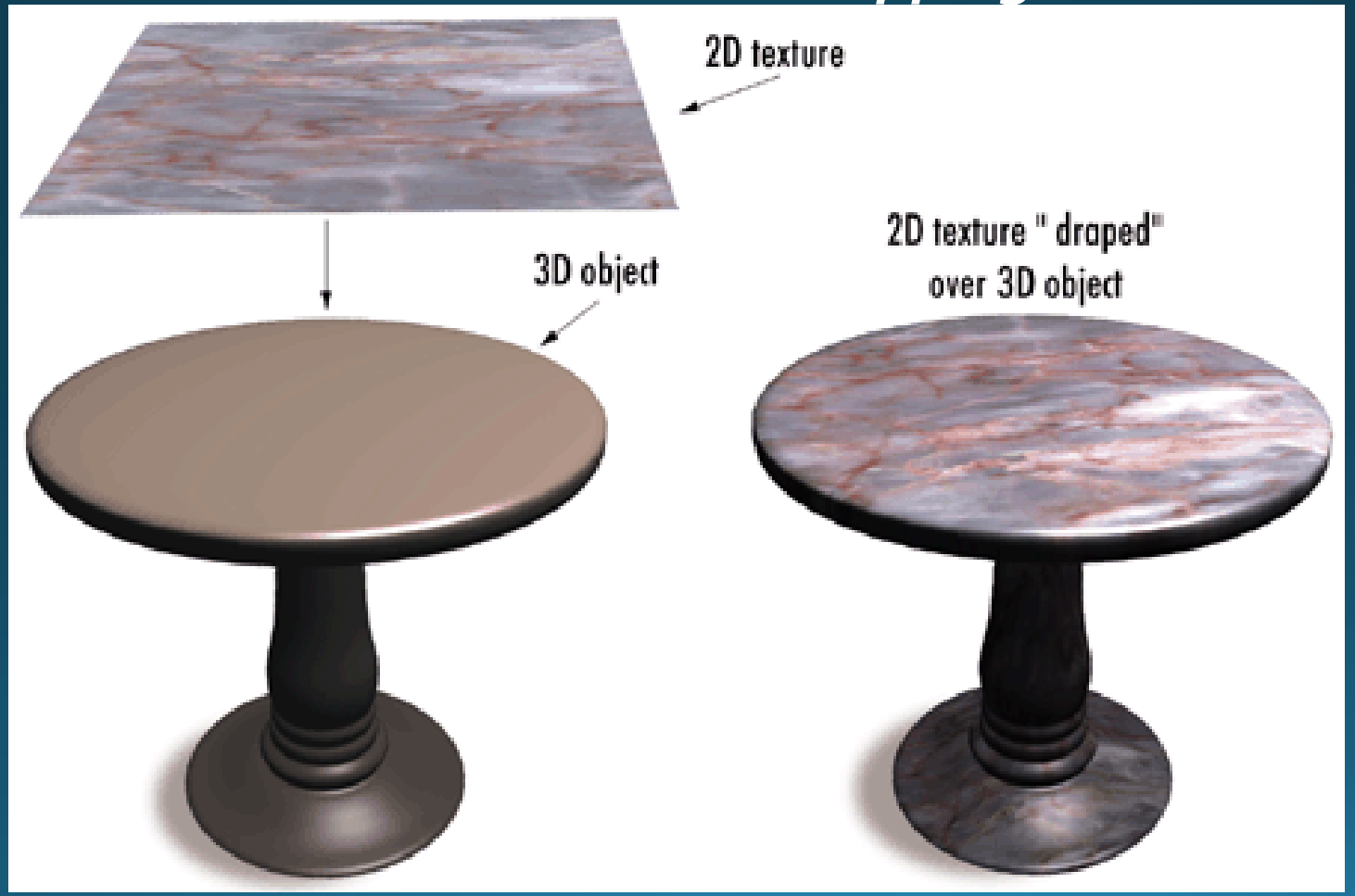
Rendering

- Consists of visible features: ***shading***



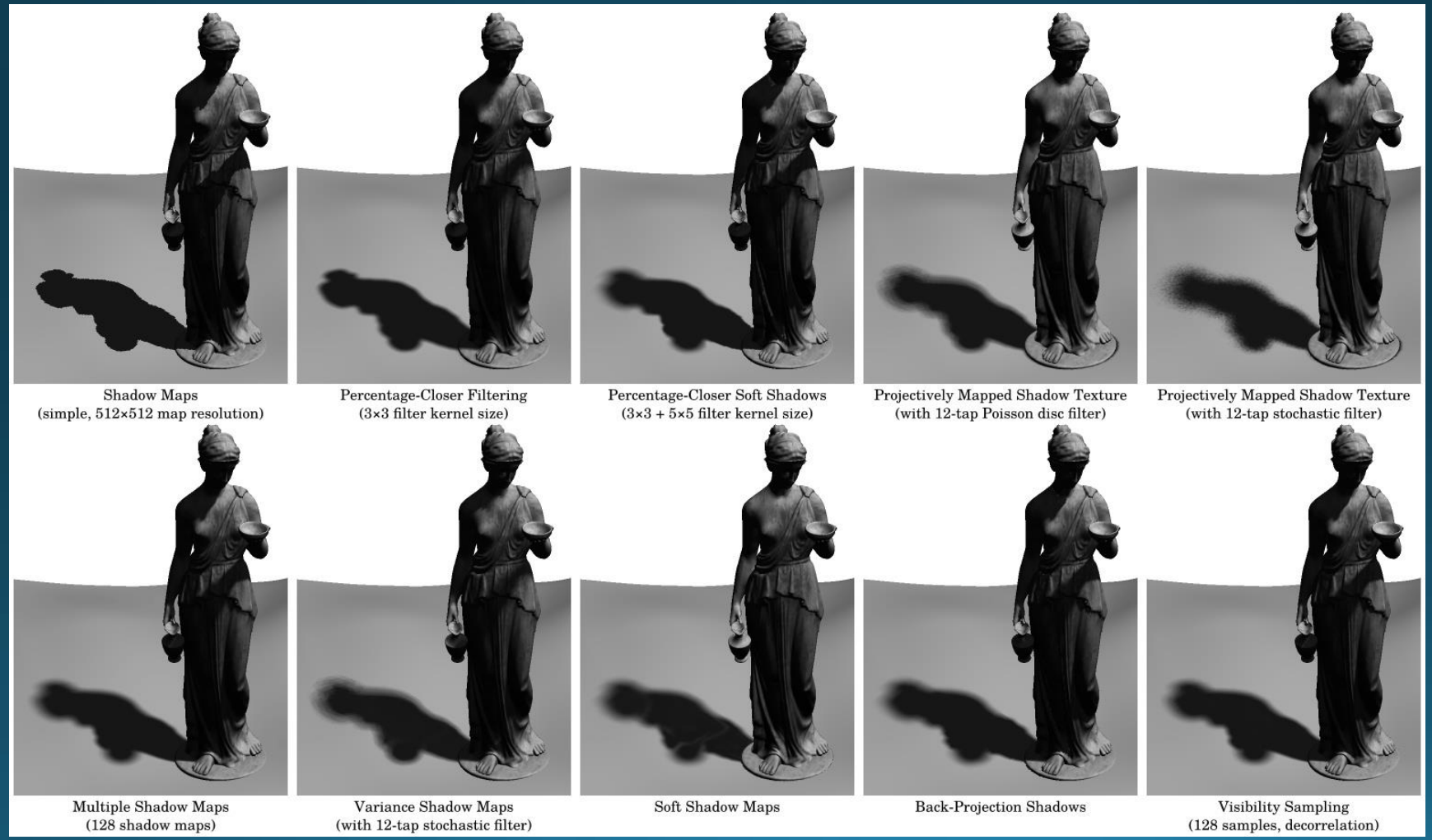
Rendering

- Consists of visible features: ***texture mapping***



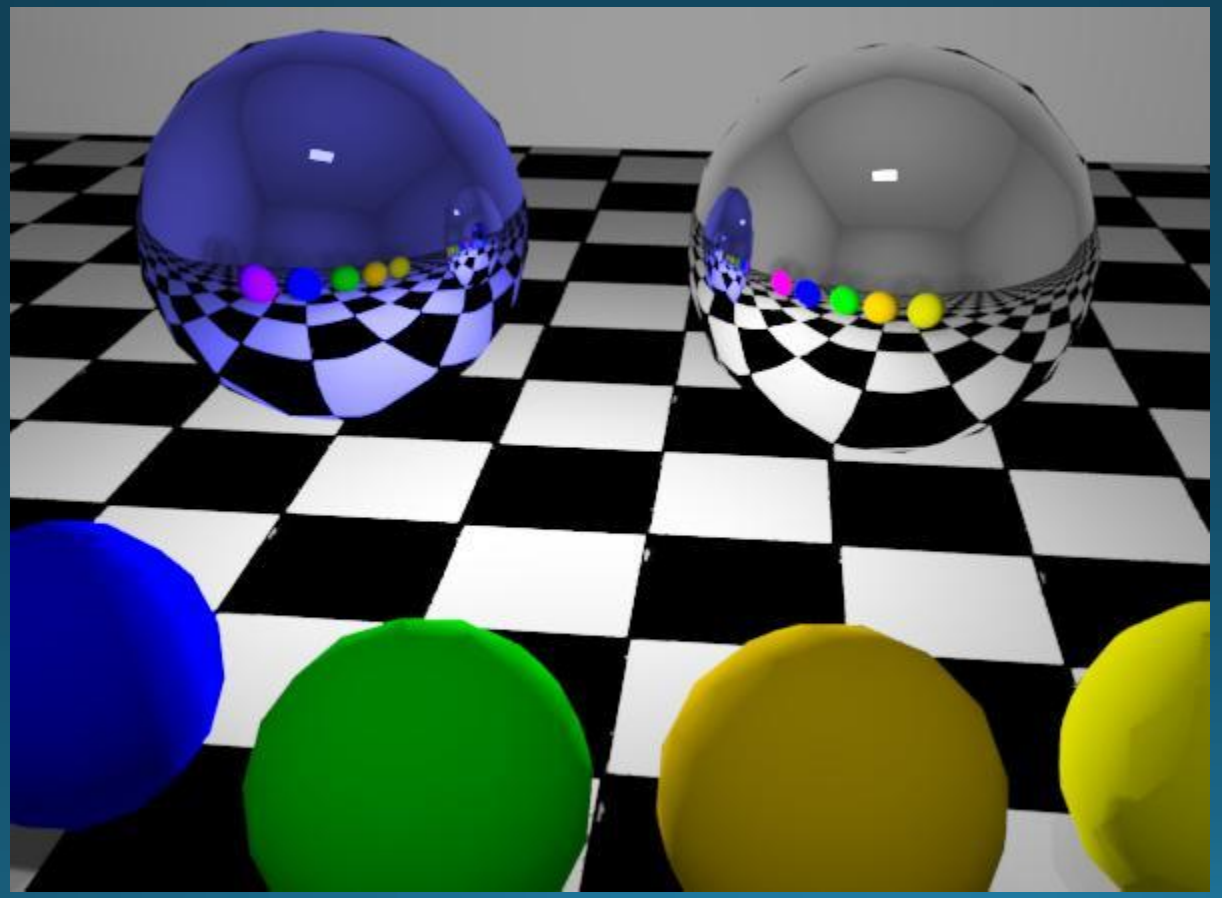
Rendering

- Consists of visible features: ***shadows***



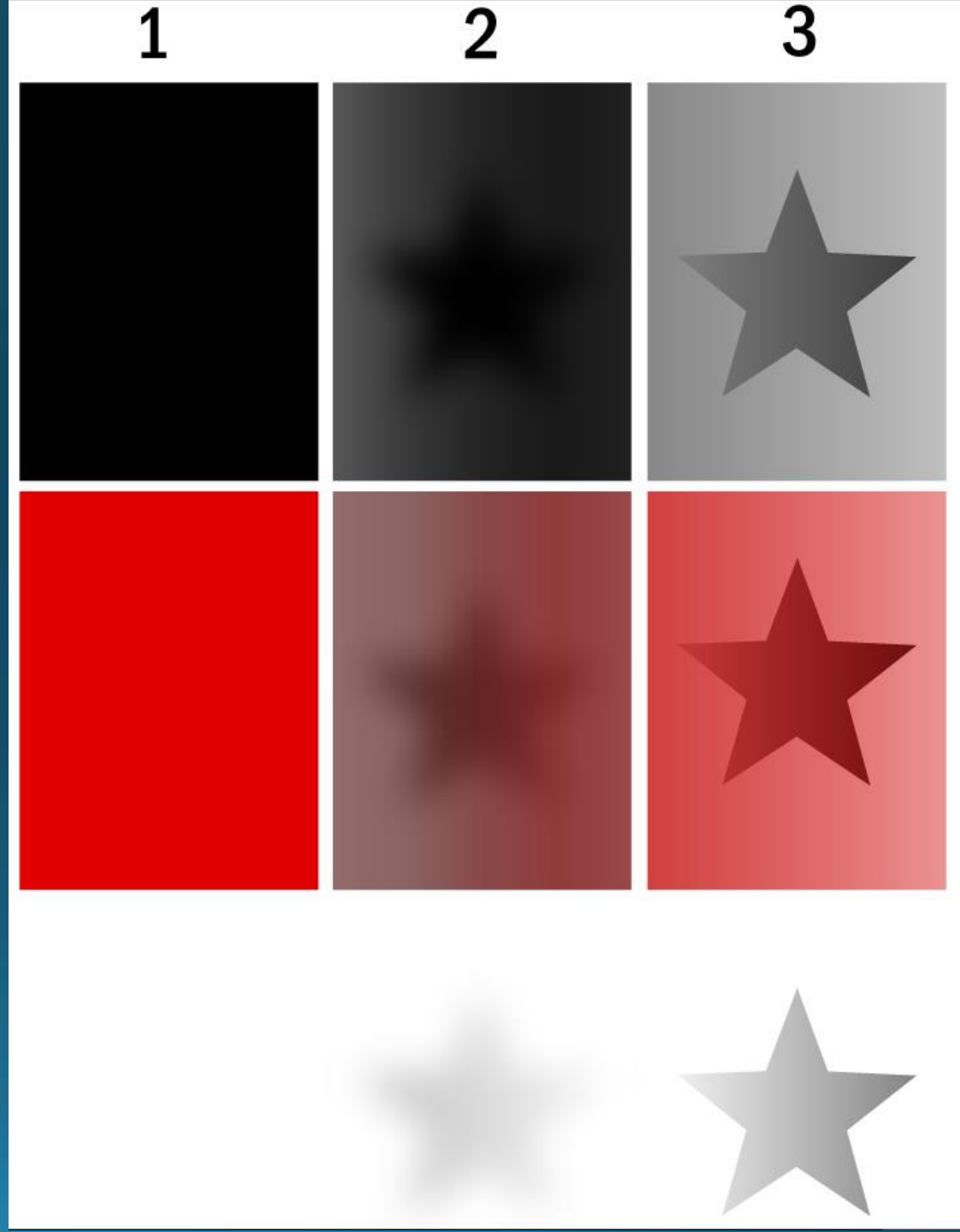
Rendering

- Consists of visible features: ***reflections***



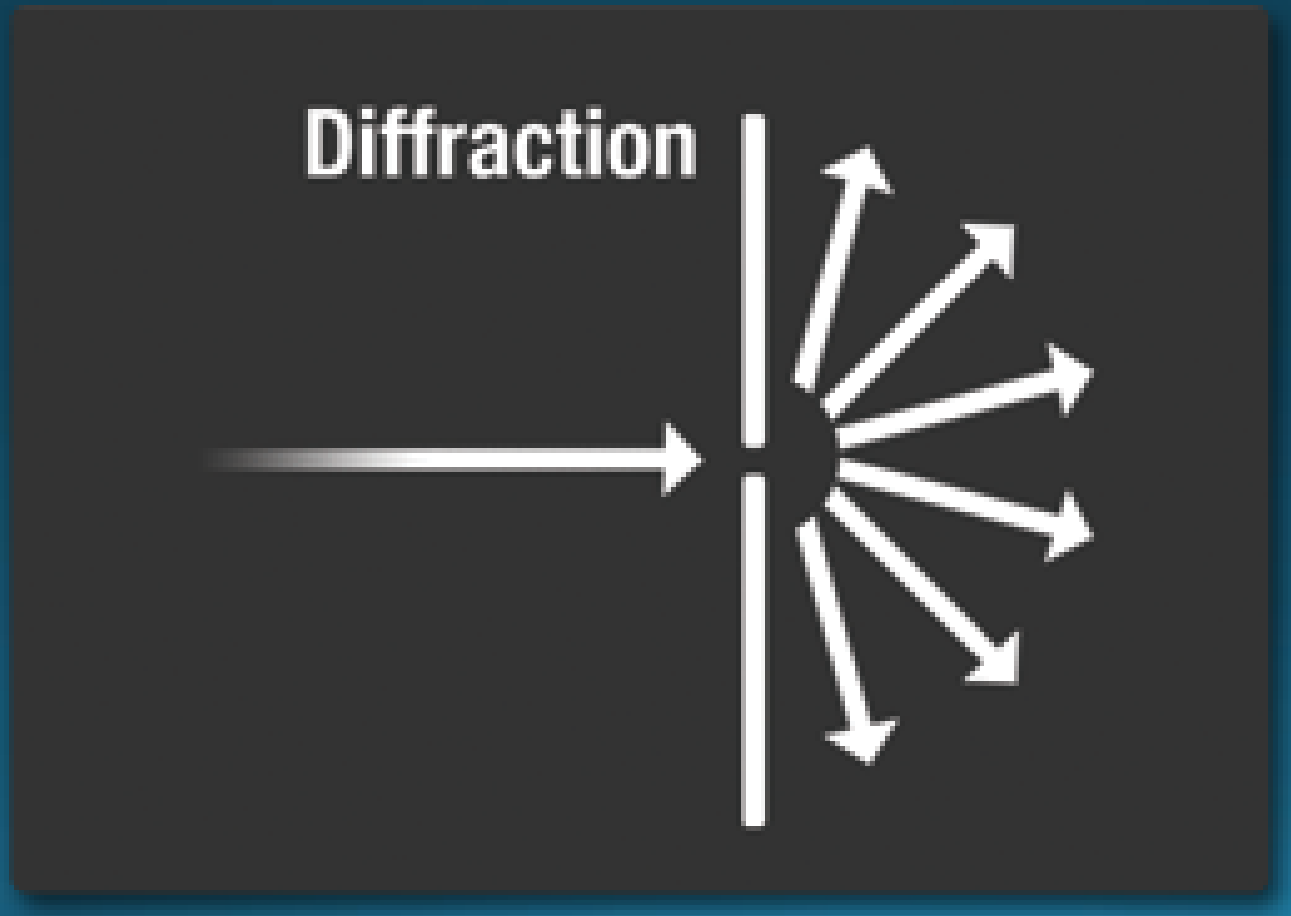
Rendering

- Consists of visible features: ***opacity (1), translucency (2), transparency (3)***



Rendering

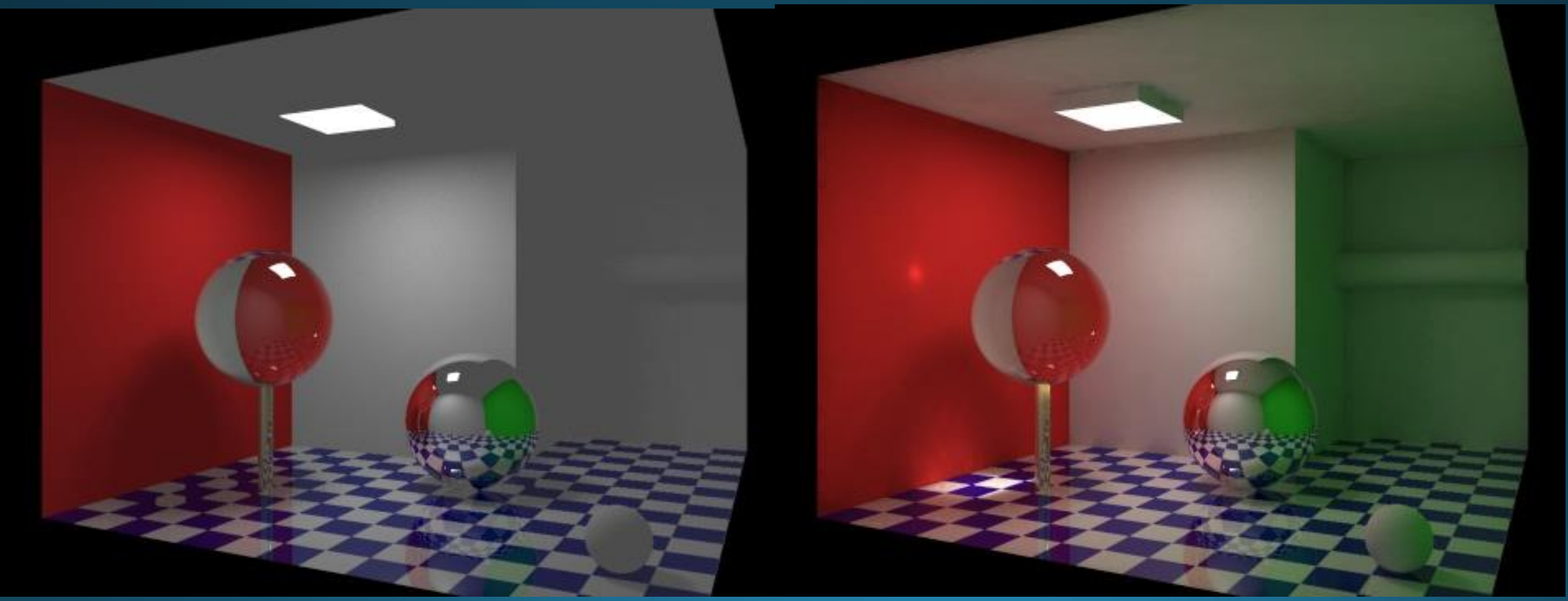
- Consists of visible features: ***diffraction***



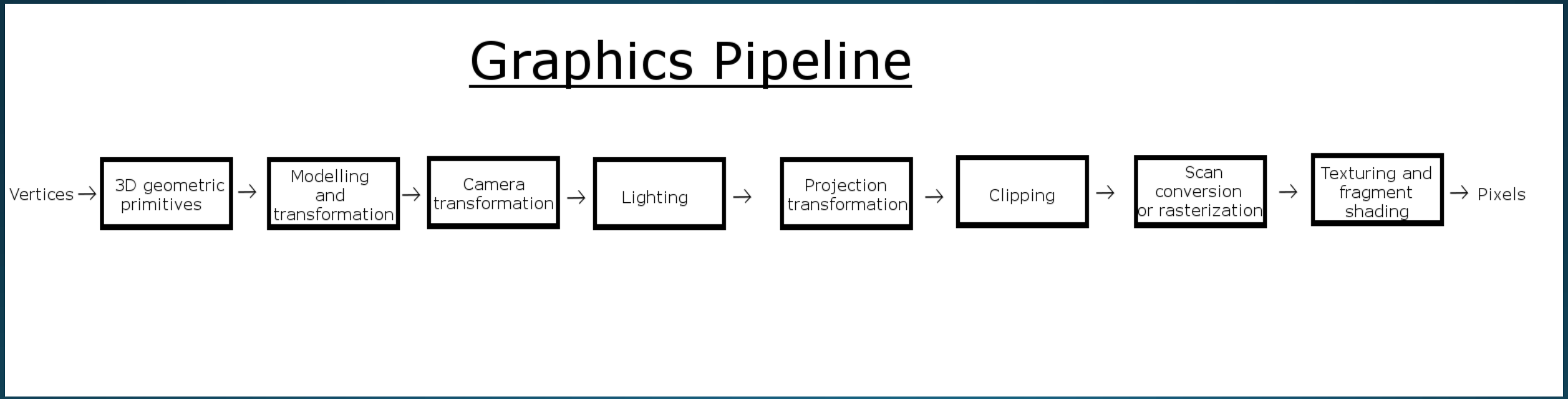
Rendering

- Consists of visible features: ***global illumination***

And so on…



Computer graphics pipeline

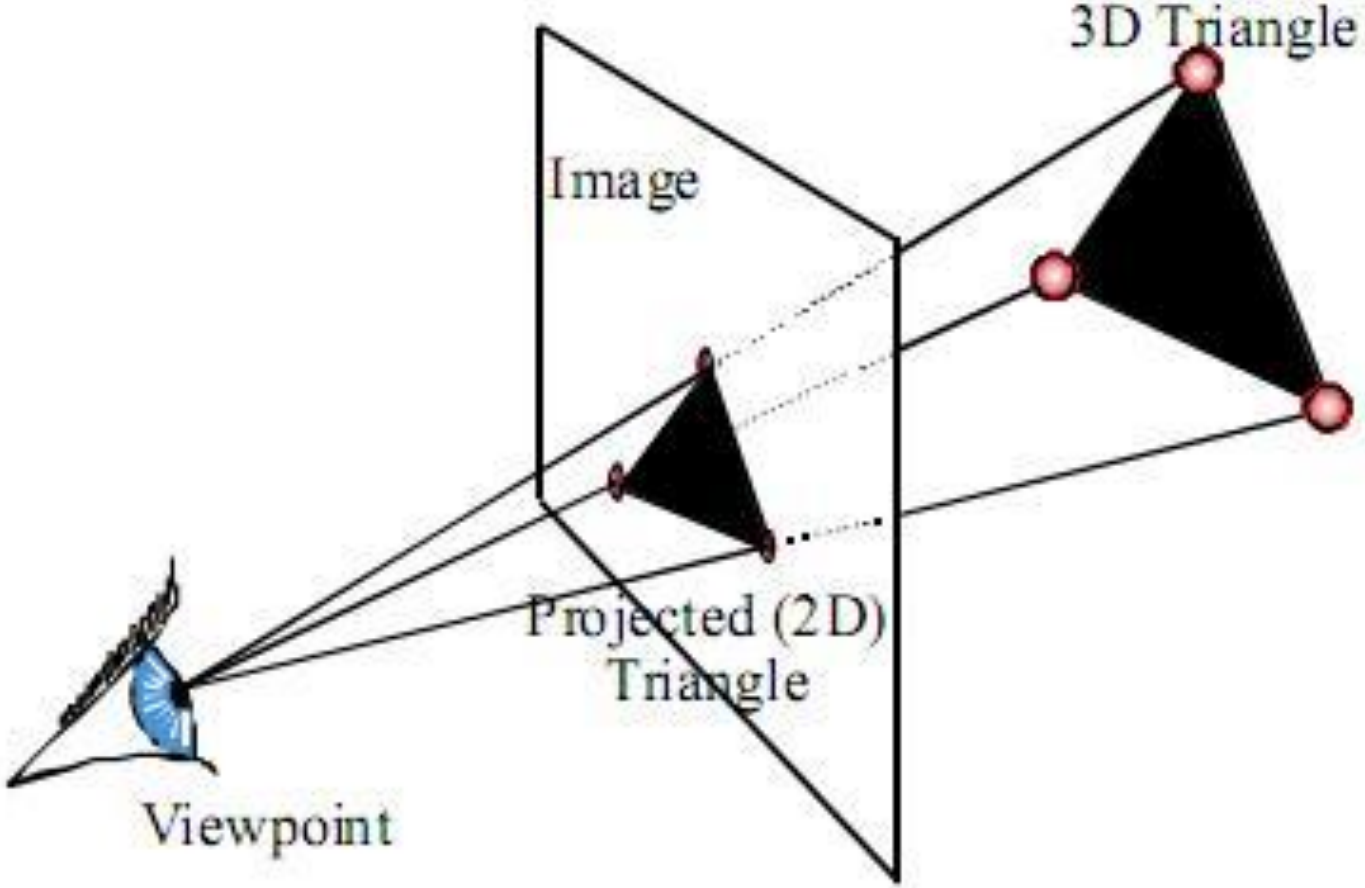


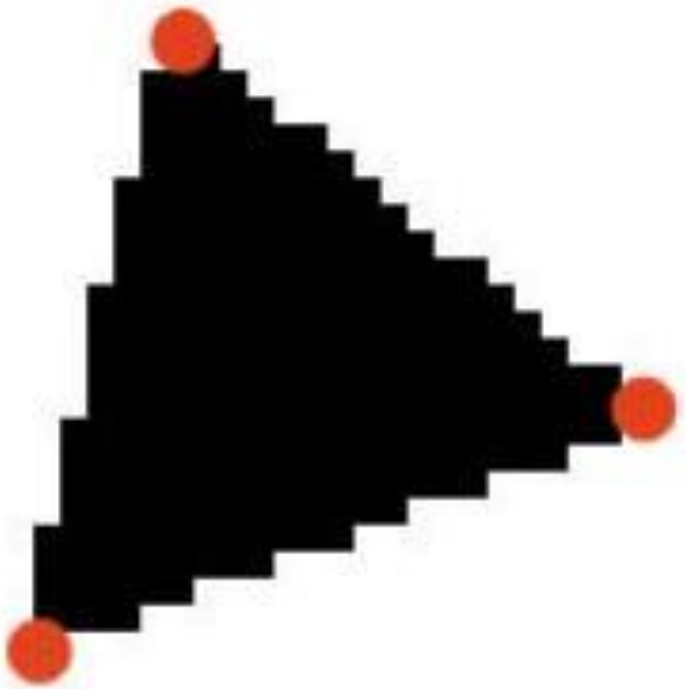
Rendering techniques

Rasterization (scanline rendering)

- input: vector graphics image representation (primitives)
- output: raster image

Rasterization





Transformations

- Stream of vertices goes to rasterizer
- They are transformed into 2D surface by means of matrix multiplications

A perspective projection transformation can be represented by the following matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & (N+F)/N & -F \\ 0 & 0 & 1/N & 0 \end{bmatrix}$$

F and N here are the distances of the far and near viewing planes, respectively.

Rotation about the X-axis:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation about the Y-axis:

$$\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation about the Z-axis:

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformations

Scaling

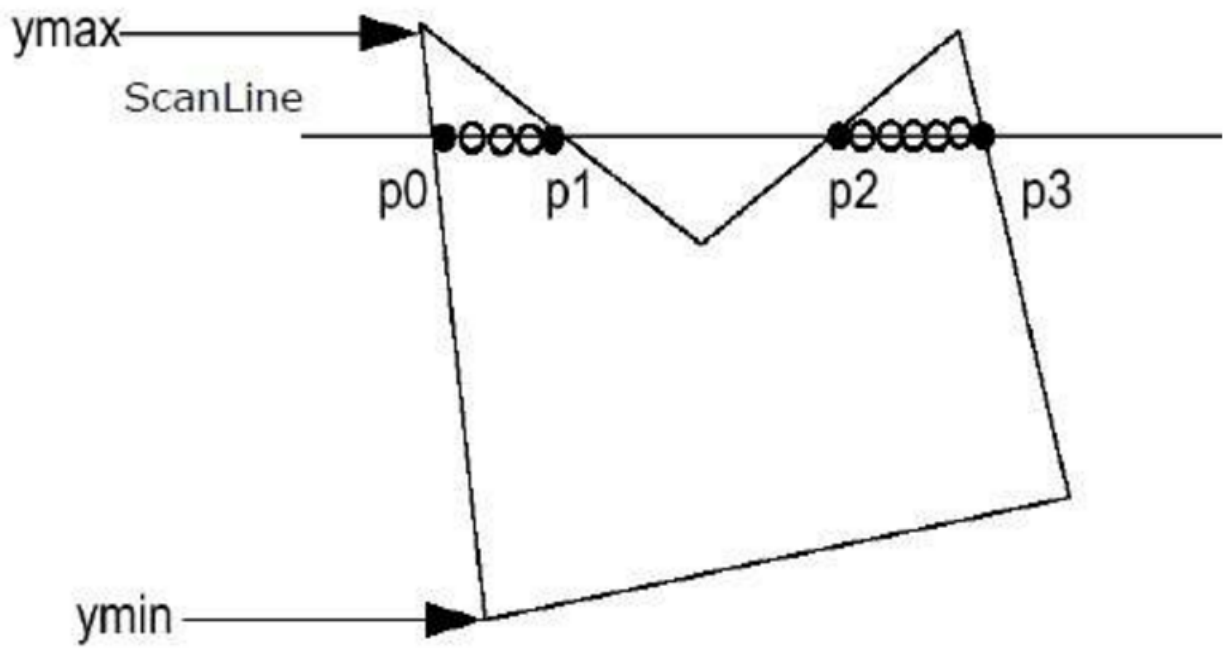
$$\begin{bmatrix} X & 0 & 0 & 0 \\ 0 & Y & 0 & 0 \\ 0 & 0 & Z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translation

$$\begin{bmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scanline rendering

Step 1 – Find out the Ymin and Ymax from the given polygon.



Step 2 – ScanLine intersects with each edge of the polygon from Ymin to Ymax. Name each intersection point of the polygon. As per the figure shown above, they are named as p0, p1, p2, p3.

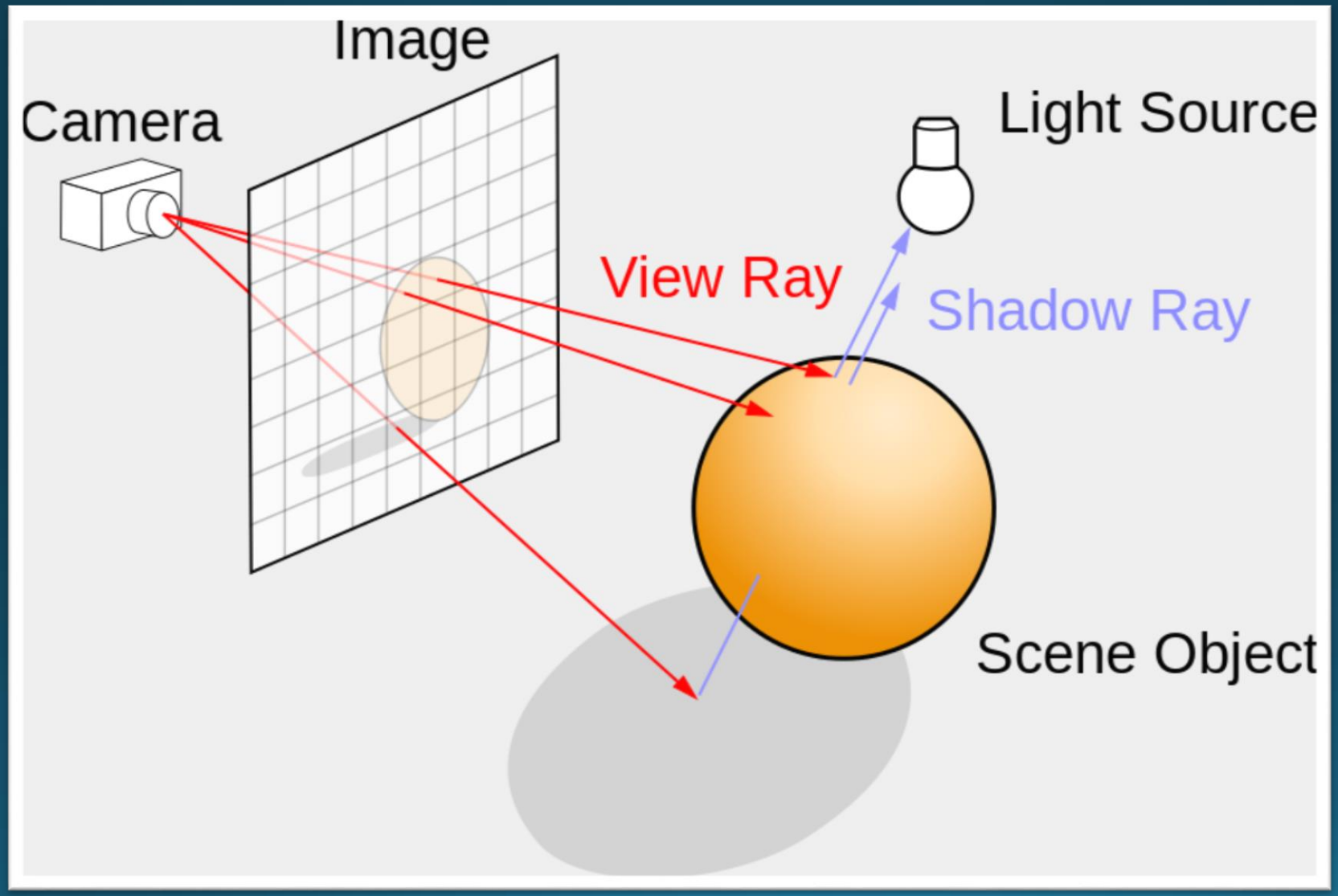
Step 3 – Sort the intersection point in the increasing order of X coordinate i.e. (p0, p1), (p1, p2), and (p2, p3).

Step 4 – Fill all those pair of coordinates that are inside polygons and ignore the alternate pairs.

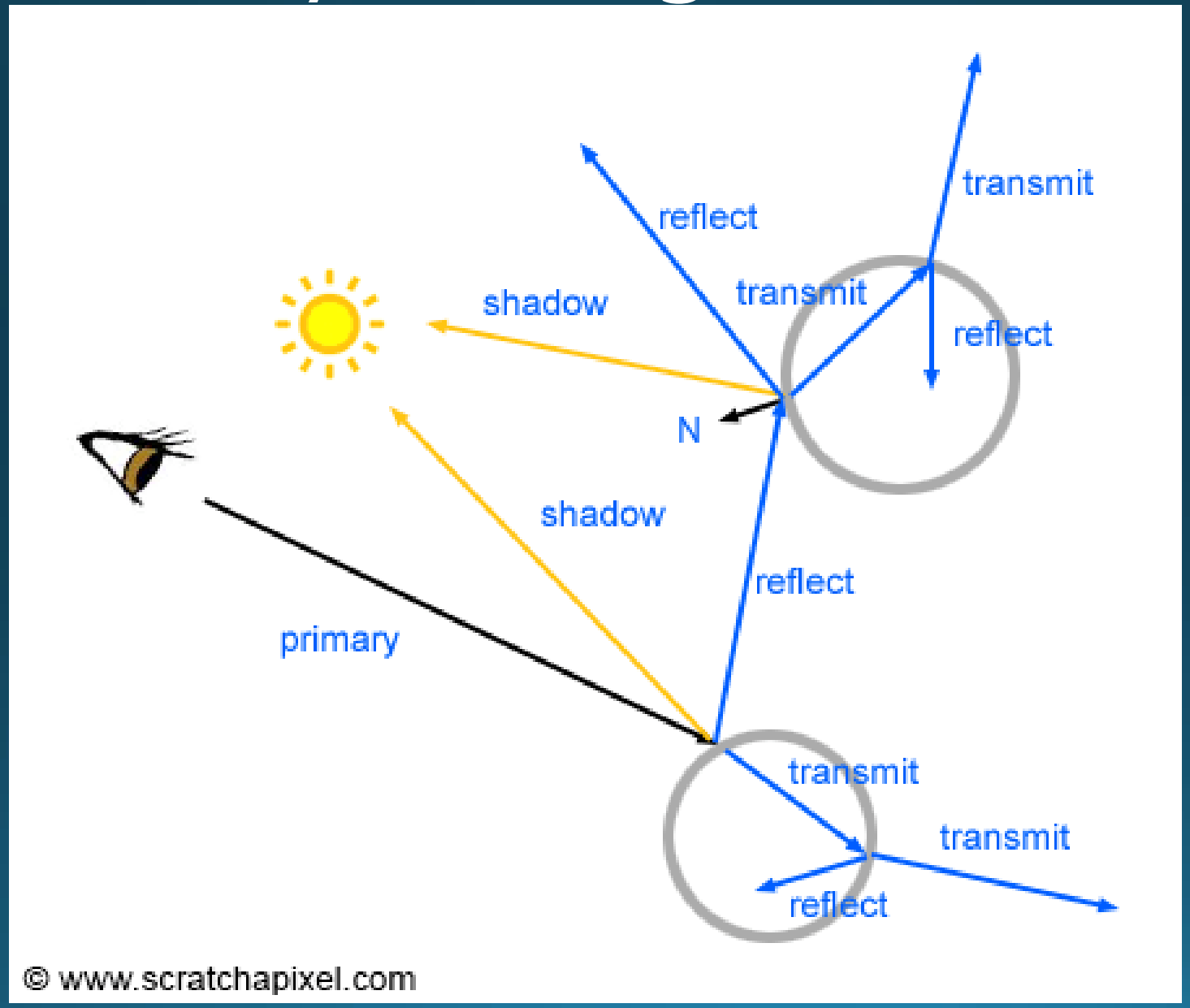
Rendering techniques

- Ray tracing & Ray casting
  - technique for generating an image by tracing the path of light through pixels in an image plane and simulating the effects of its encounters with virtual objects.

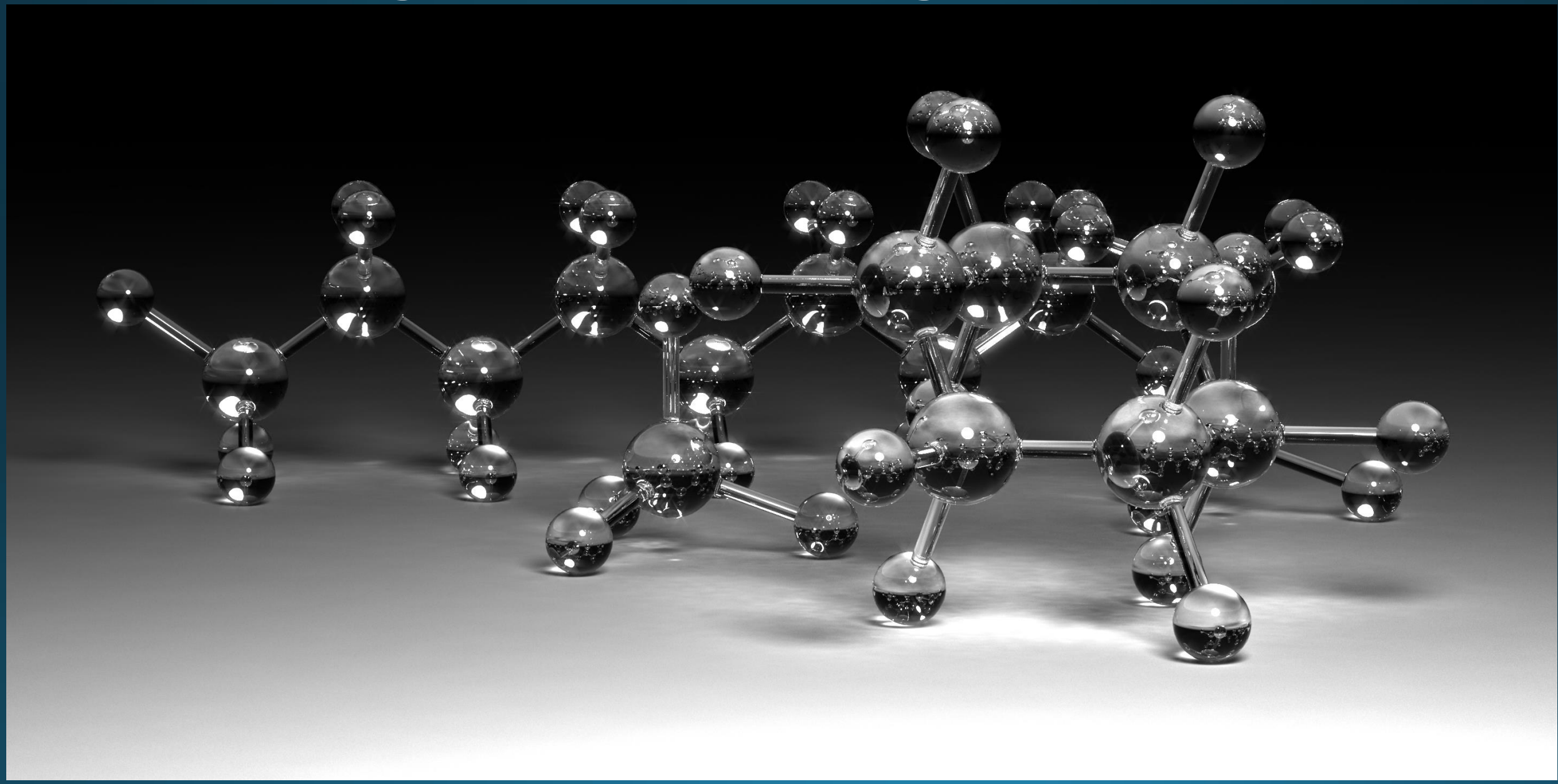
Ray tracing



Recursive ray tracing



Ray tracing & Ray casting



Ray casting vs ray tracing

- Difference:

  Ray casting is a rendering algorithm that never recursively traces secondary rays, whereas other ray tracing-based rendering algorithms may do so.

Problem with ray tracing/casting

Computationally expensive:

- Good for non-realtime rendering because of high quality of rendered image **BUT**

- To slow for realtime rendering

Alternative: z-buffer triangle rasterization

Z-buffer triangle rasterization

Z-buffer triangle rasterization

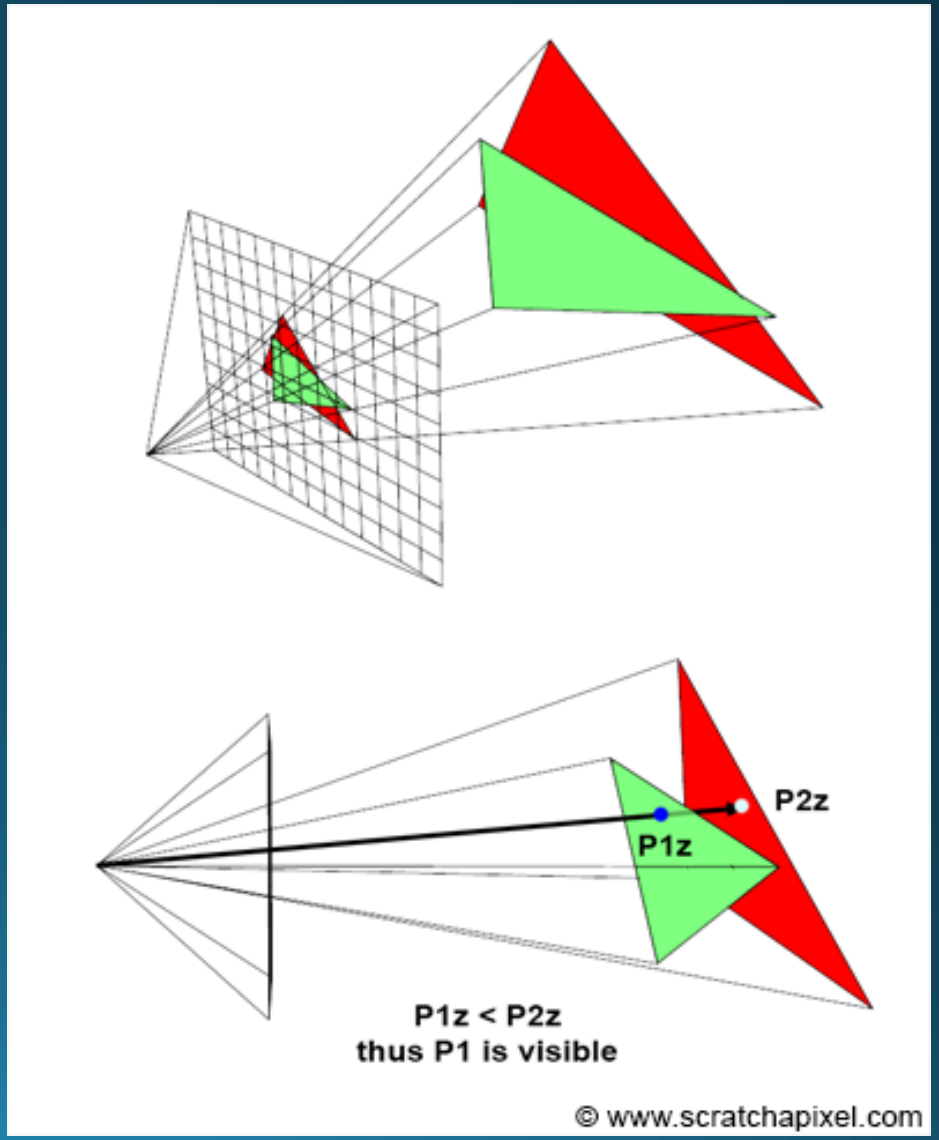
- algorithm for visible surface determination in 3D computer graphics

Z-buffer is a two-dimensional array

Z-buffer triangle rasterization

1) Compute the z-coordinate or depth of the point on the triangle that the pixel overlaps.

2) Compare that current triangle depth with the value stored in the depth buffer for that pixel.



Radiosity

- Radiosity is a global illumination algorithm which consider illumination arriving on a surface comes not just directly from the light sources, but also from other surfaces reflecting light.

- Visual effects:
  - Soft shadows
  - Color bleeding

Radiosity

Direct lighting only

+ indirect lighting

Example: Unreal Engine 4

- https://docs.unrealengine.com/latest/INT/Resources/Showcases/RealisticRendering/index.html