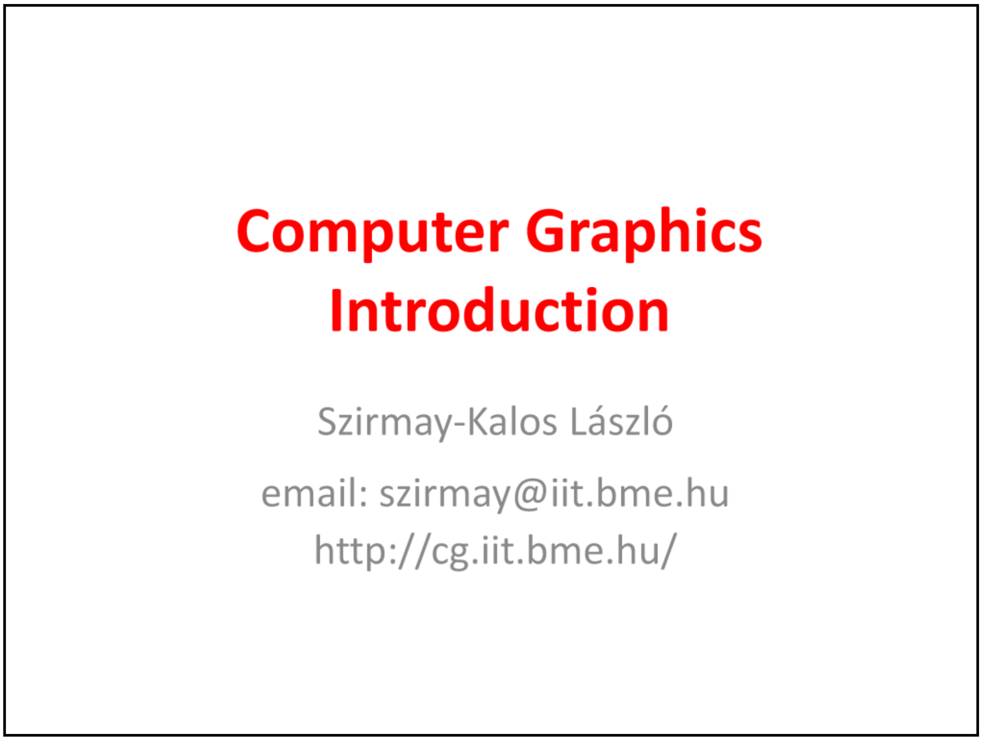
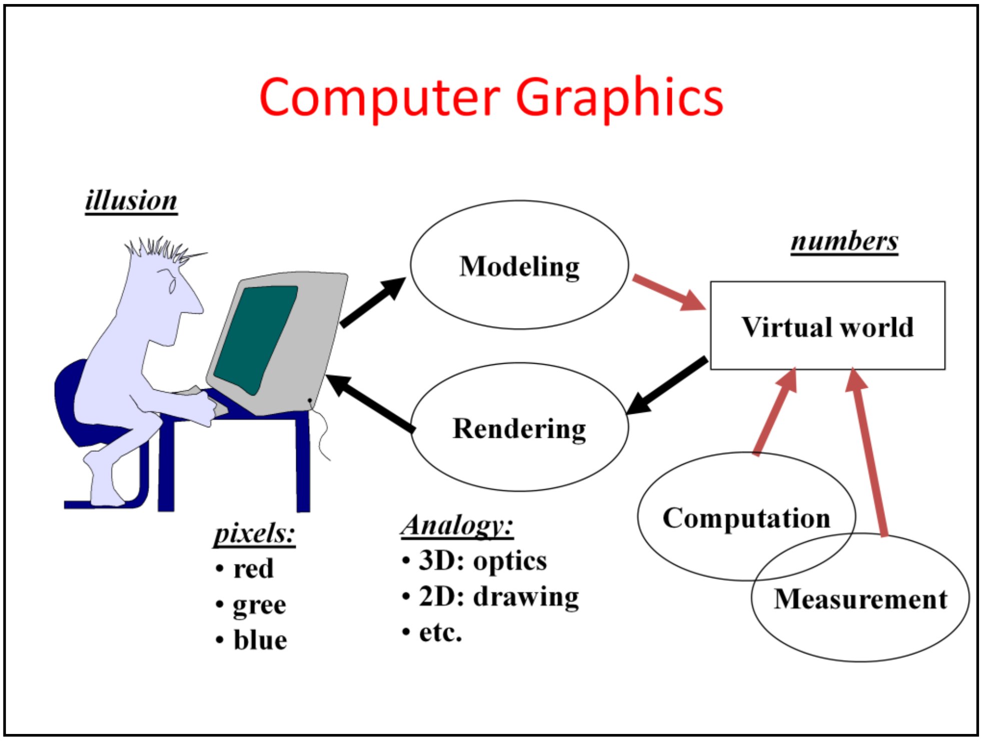


Computer Graphics Introduction

Szirmay-Kalos László

email: szirmay@iit.bme.hu
http://cg.iit.bme.hu/



Computer Graphics

illusion

Modeling

numbers

Virtual world

Rendering

Computation

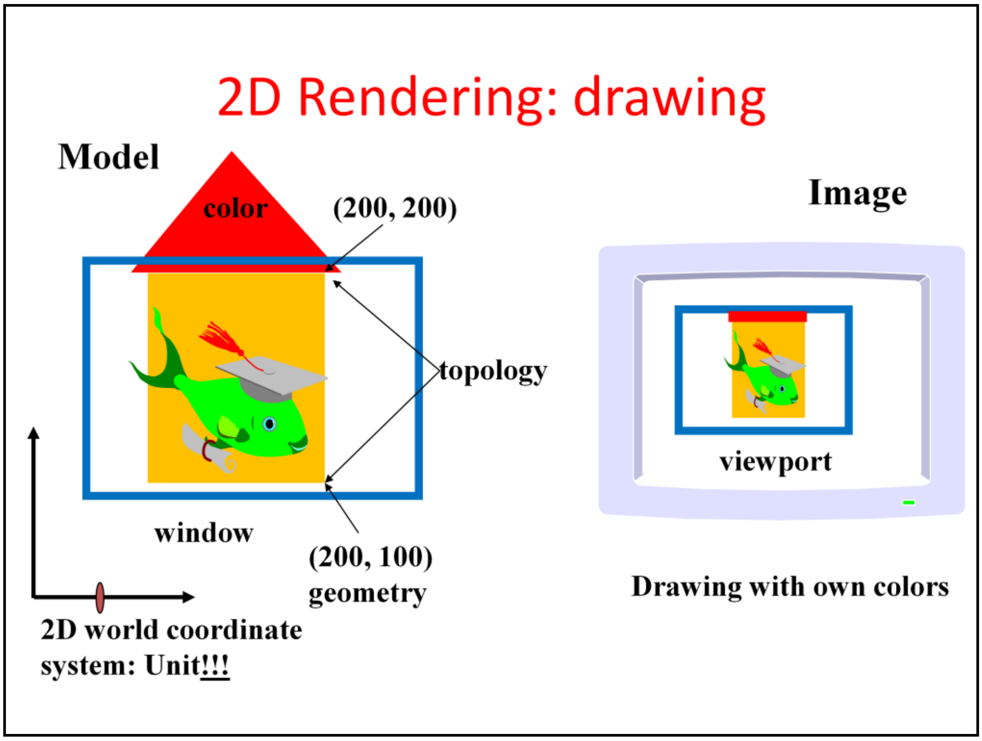
Measurement

pixels:
- red
- gree
- blue

Analogy:
- 3D: optics
- 2D: drawing
- etc.

In computer graphics we render virtual worlds by taking a photo of them and presenting their image to the user. The virtual world is stored in the computer memory. The virtual world model can be the result of an interactive modeling process, simulation, measurement, etc.

Rendering can be regarded as an abstract mapping from the virtual world model to the intensity and color values of the computer screen. There are infinite number of possibilities to define this mapping. If we wish to have images that look like real images, we should simulate the image creation process or the real world. For example, we can simulate light transport, i.e. optics, or manual drawing.



2D Rendering: drawing

Model

color

(200, 200)

topology

window

(200, 100)
geometry

2D world coordinate
system: Unit!!!

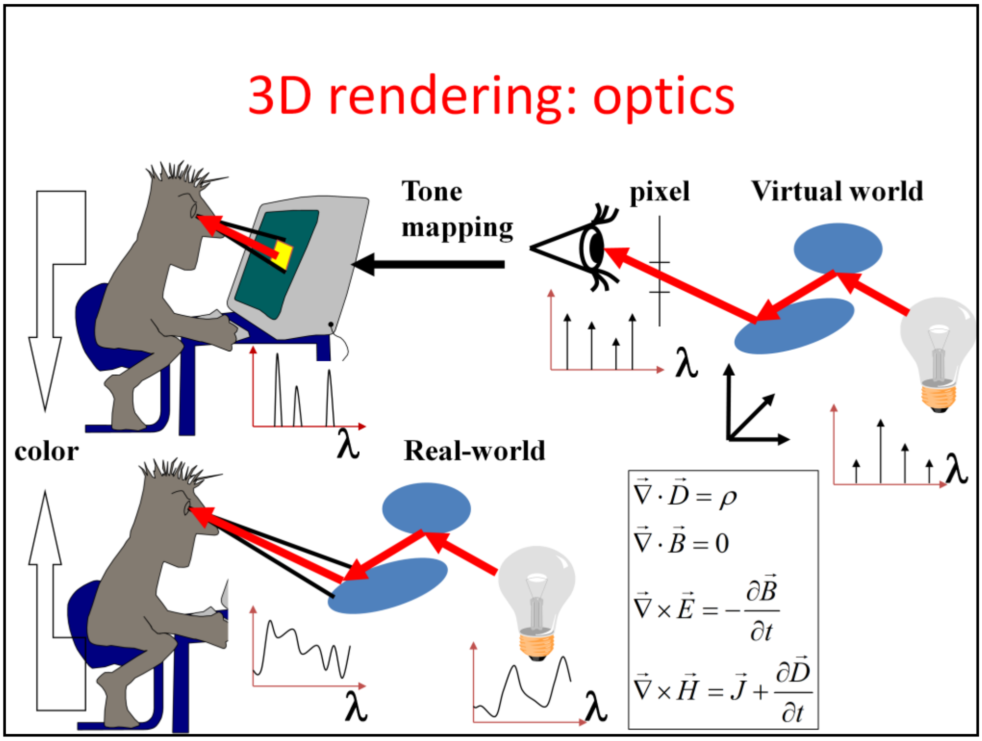
Image

viewport

Drawing with own colors

Let us look at the details when the virtual world is two dimensional, so objects are in a plane. A convenient reference system is a Cartesian coordinate system with an origin, two axes and also a unit. Using these, every point of the plane can be specified by two numbers defining the distance traveled along the two axes and measured with respect to the unit.

With pairs of numbers, points can be defined, which can form primitives by adding topology information. For example, we can say that these three points define a triangle. Primitives are given material properties, which usually include the color.



3D rendering: optics

$$\vec{\nabla} \cdot \vec{D} = \rho$$
$$\vec{\nabla} \cdot \vec{B} = 0$$
$$\vec{\nabla} \times \vec{E} = -\frac{\partial \vec{B}}{\partial t}$$
$$\vec{\nabla} \times \vec{H} = \vec{J} + \frac{\partial \vec{D}}{\partial t}$$

If we want to create photo like images, we should simulate the light transport and provide the user with the illusion that he watches the real world and not a computer screen.

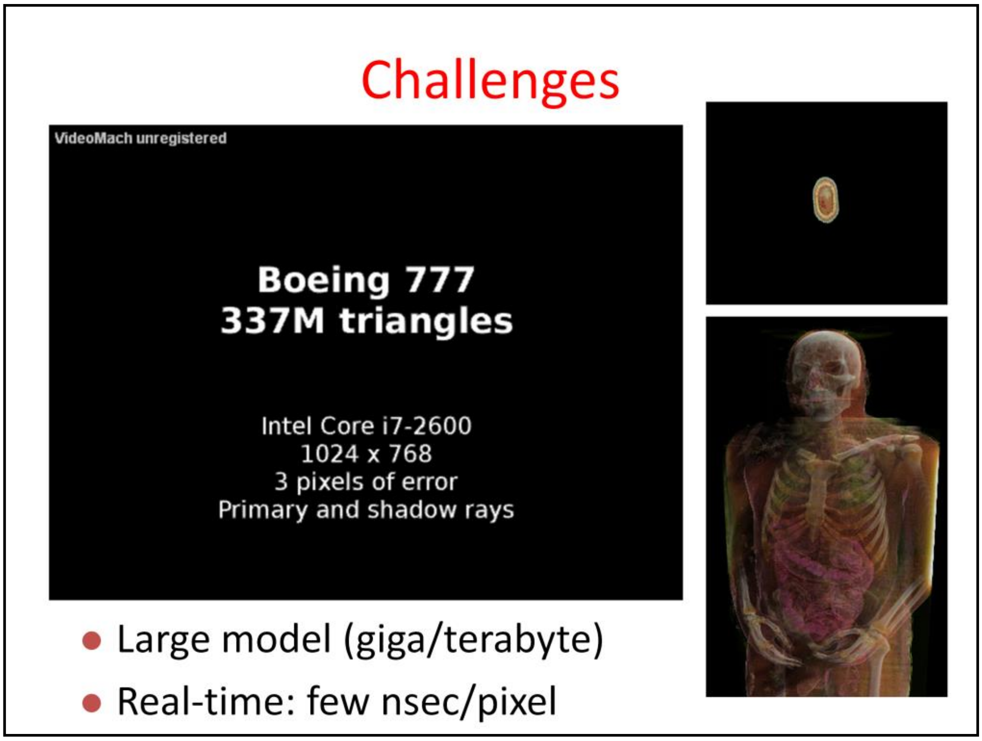
If we could guarantee that the human eye gets the same photons (i.e. the same number and of the same frequency) from the solid angle subtended by a pixel as the eye got from the real world, then it would not be possible for the user to distinguish between the computer monitor and the real world since the same photons result in similar color impressions. So in computer graphics, we should compute the number and frequency of photons, i.e. the power spectrum of the light that would enter the eye from the solid angle of each pixel. Then the display should be controlled to emit similar photons. Fortunately, we do not have to emit exactly the same spectrum since the human eye is very bad in measuring a spectrum. In fact, the illusion of most of the spectra can be provided by carefully selected red, green and blue intensities. So having calculated the spectrum, we convert it to an equivalent red/green/blue intensity triplet and get the monitor to emit it.

The calculation of the light spectrum requires the solution of the photon transfer or the transfer of electromagnetic waves. The equations describing these phenomena are the Maxwell equations, so in fact, graphics should solve these fundamental equations to obtain the image.

Photorealistic rendering

LuxRender
BME-IIT
20

The results of the simulation of optics laws or Maxwell equations are indeed like real photos.



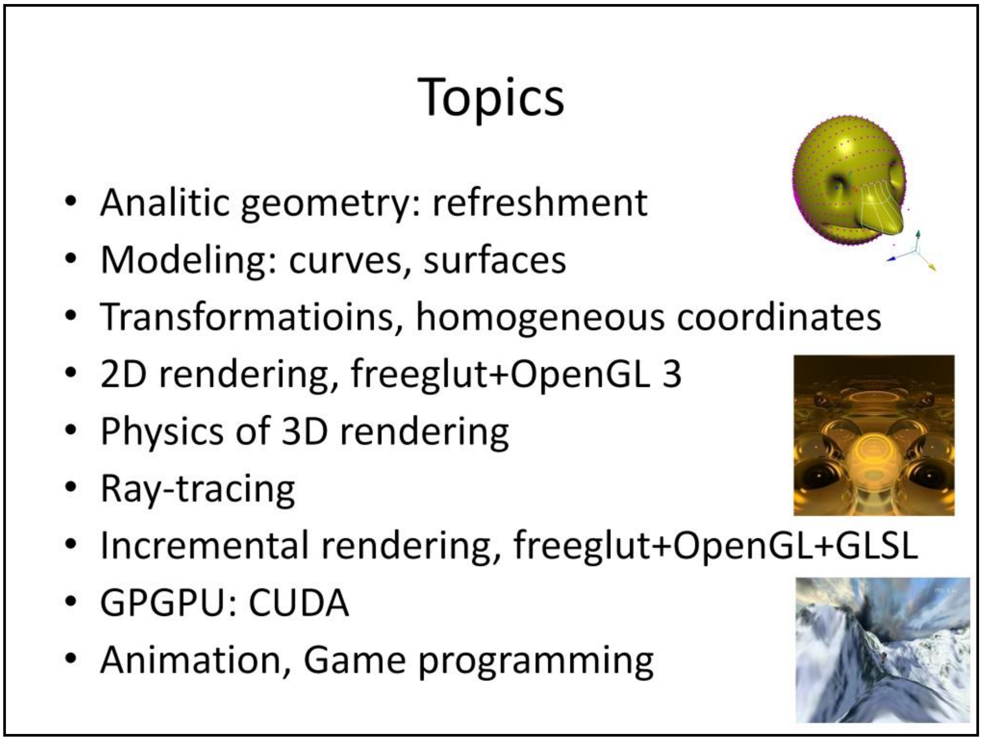
Challenges

Boeing 777
337M triangles

Intel Core i7-2600
1024 x 768
3 pixels of error
Primary and shadow rays

- Large model (giga/terabyte)
- Real-time: few nsec/pixel

The main challenge is that we have very little time for a pixel when real-time animation is needed.



Topics

- Analitic geometry: refreshment
- Modeling: curves, surfaces
- Transformatioins, homogeneous coordinates
- 2D rendering, freeglut+OpenGL 3
- Physics of 3D rendering
- Ray-tracing
- Incremental rendering, freeglut+OpenGL+GLSL
- GPGPU: CUDA
- Animation, Game programming

This is the menu of the course, including the main topics.