TIEA311
Tietokonegrafiikan perusteet
kevät 2019

(“Principles of Computer Graphics” – Spring 2019)

TIEA311 Tietokonegrafiikan perusteet – kevät 2019 (“Principles of Computer Graphics” – Spring 2019)

Adapted from: *Wojciech Matusik*, and *Frédo Durand*: 6.837 Computer Graphics. Fall 2012. Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu/.

Frontpage of the local course version, held during Spring 2019 at the Faculty of Information technology, University of Jyväskylä:
`http://users.jyu.fi/~nieminen/tgp19/`

TIEA311 - Today in Jyväskylä

Plan for today:

- (Possible announcements or food-for-thought)
- Usual warm-up and group discussion
- Try to address the most urgent issues
- Break – reset the brain.
- Then continue with the theory.

TIEA311 - Today in Jyväskylä

We start by discussion, reflection and **questions**!

Work in groups of 3 students if possible:

- Fast warm-up: 90 seconds evenly split between group members (30s each in groups of 3), no interruptions from others: Foremost feelings right now?
- Reflection: Silent work, solo, 1 minute, **list words on paper**: What have you learned during the last week? Or since the course started?
- Interaction: 1.5 minutes group discussion: Compare if you learned the same or different things? Do those things feel useful? Why or why not?
  $\rightarrow$ Sum it up classwide.
- Interaction: Group work, 1.5 minutes or less if talk ends: At the moment, what would be the most helpful thing to help you (or others!)?
  $\rightarrow$ Sum it up classwide, and try to address the findings.

TIEA311 - Today in Jyväskylä

What were the findings in group discussion?

What were found to be the most important issues to address right now?

$\rightarrow$ Classwide discussion is found on the lecture video.

NOTE: Even if you watch at home, please think about the same things and try to be in ”virtual dialogue” with those in classroom. Use pen and paper! I believe, more and more every day, that doing so will make your brain perform activities that help **your own learning**.

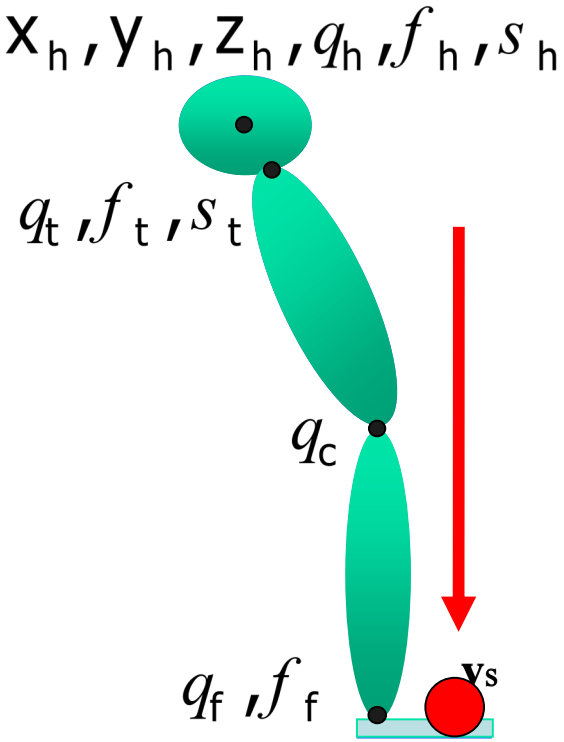
NOTE: Contemplate if you could watch the lecture videos with some friends who would also like to learn computer graphics? Get some pizza and coke if it helps you get to the mood(?).

TIEA311 - Today in Jyväskylä

Plan for today:

- (Possible announcements or food-for-thought)
- Usual warm-up and group discussion
- Try to address the most urgent issues
- Break – reset the brain.
- Then continue with the theory.

Forward Kinematics



Transformation matrix **S** for a point **v**s is a matrix composition of all joint transformations between the point and the root of the hierarchy. **S** is a function of all the joint angles between here and root.

Note that the angles have a non-linear effect.

This product is **S**

$$\mathbf{v}_w = \mathbf{T}(x_h, y_h, z_h)\mathbf{R}(q_h, f_h, s_h)\ \mathbf{TR}(q_t, f_t, s_t)\,\mathbf{TR}(q_c)\,\mathbf{TR}(q_f, f_f)\ \mathbf{v}_s$$

$$v_w = S\left(\underbrace{x_h, y_h, z_h, \theta_h, \phi_h, \sigma_h, \theta_t, \phi_t, \sigma_t, \theta_c, \theta_f, \phi_f}_{p}\right) v_s = S(p)\, v_s$$

parameter vector **p**

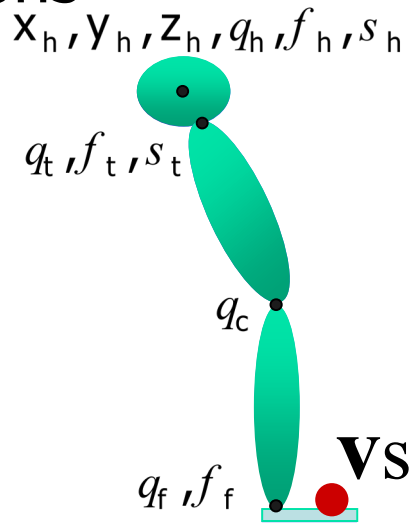
TIEA311 - “Fast-forward” just a bit

Worry not, if you get an “yli hilseen” feeling about the following few things. On this course, we leave them on a “nice to know” level. That is, while learning basics, it is nice to know where the rabbit hole could lead to!

Note to self: apply great speed with the next slide button on lecture.

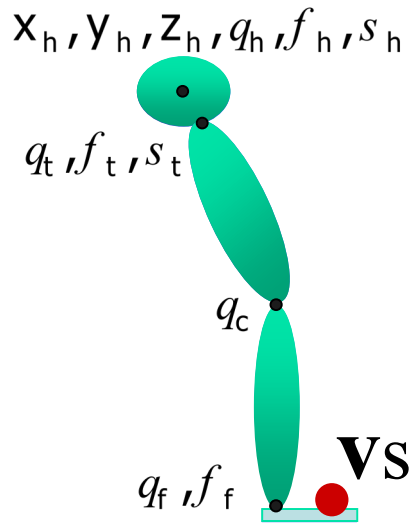
Inverse Kinematics

- Context: an animator wants to "pose" a character
- Specifying every single angle is tedious and not intuitive
- Simpler interface:
  directly manipulate position of e.g. hands and feet
- That is, specify vw, infer joint transformations



Inverse Kinematics

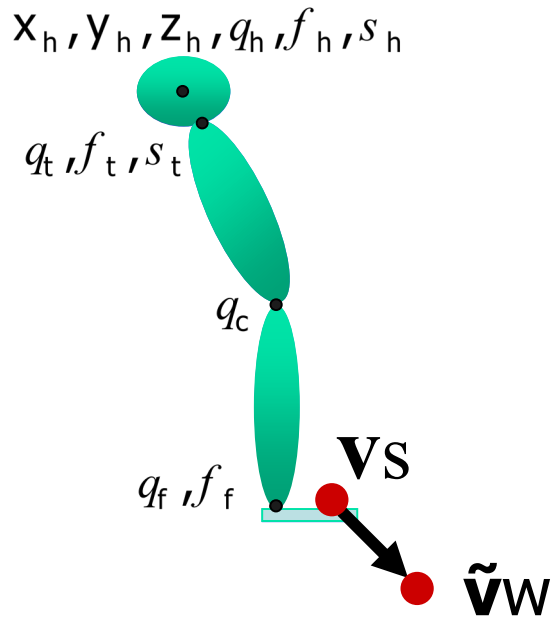
- Forward Kinematics
  - Given the skeleton parameters **p** (position of the root and the joint angles) and the position of the point in local coordinates **v**s, what is the position of the point in the world coordinates **v**w?
  - Not too hard, just apply transform accumulated from the root.



Inverse Kinematics

- Forward Kinematics
  - Given the skeleton parameters **p** (position of the root and the joint angles) and the position of the point in local coordinates **v**s, what is the position of the point in the world coordinates **v**w?
  - Not too hard, just apply transform accumulated from the root.

- Inverse Kinematics
  - Given the current position of the point and the desired new position $\tilde{\mathbf{v}}_w$ in world coordinates, what are the skeleton parameters **p** that take the point to the desired position?



Inverse Kinematics

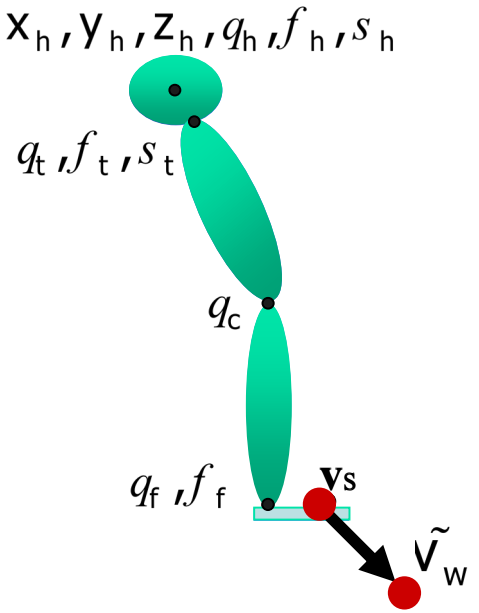
- Given the position of the point in local coordinates **v**s and the desired position $\tilde{\mathbf{v}}_w$ in world coordinates, what are the skeleton parameters **p**?

$$\tilde{v}w_i = S\left(\underbrace{x_h, y_h, z_h, \theta_h, \phi_h, \sigma_h, \theta_t, \phi_t, \sigma_t, \theta_c, \theta_f, \phi_f}_{p}\right) v_s = S(p)\, v_s$$

skeleton parameter vector **p**

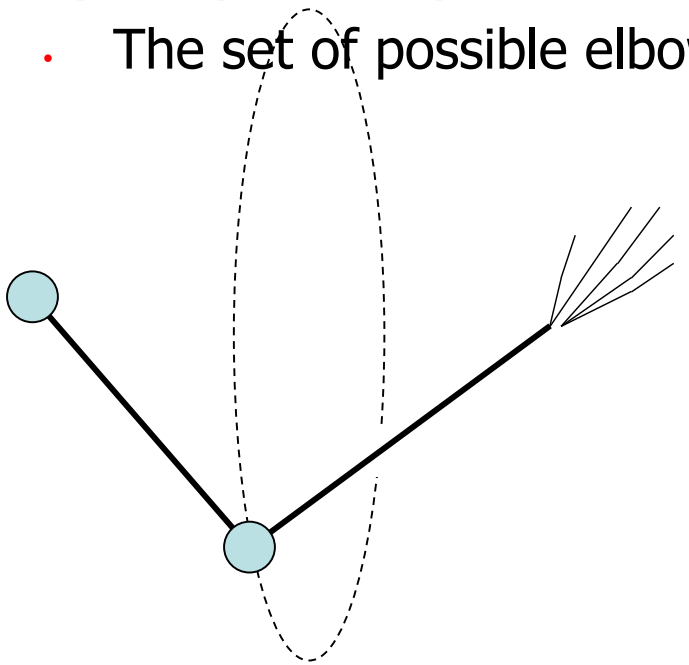
- Requires solving for **p**, given **v**s and $\tilde{\mathbf{v}}_w$
  - Non-linear and …

It's Underconstrained



- Count degrees of freedom:
  - We specify one 3D point (3 equations)
  - We usually need more than 3 angles
  - **p** usually has tens of dimensions

- Simple geometric example (in 3D): specify hand position, need elbow & shoulder
  - The set of possible elbow location is a circle in 3D



How to tackle these problems?

$$\boldsymbol{v}_{\mathrm{WS}} = \boldsymbol{S}(\boldsymbol{p})\,\boldsymbol{v}_{\mathrm{S}}$$

- Deal with non-linearity:
  Iterative solution (steepest descent)
  - Compute Jacobian matrix of world position w.r.t. angles
    - Jacobian: "If the parameters **p** change by tiny amounts, what is the resulting change in the world position **v**WS?"
  - Then invert Jacobian.
    - This says "if **v**WS changes by a tiny amount, what is the change in the parameters **p**?"
  - But wait! The Jacobian is non-invertible (3xN)
  - Deal with ill-posedness: Pseudo-inverse
    - Solution that displaces things the least
    - See http://en.wikipedia.org/wiki/Moore-Penrose_pseudoinverse
  - Deal with ill-posedness: Prior on "good pose" (more advanced)
- Additional potential issues: bounds on joint angles, etc.
  - Do not want elbows to bend past 90 degrees, etc.

$$\left[\frac{\partial(\boldsymbol{v}_{\mathrm{WS}})_i}{\partial p_j}\right]$$

Example: Style-Based IK

- Video
- Prior on “good pose”
- Link to paper: Grochow, Martin, Hertzmann, Popovic: Style-Based Inverse Kinematics, ACM SIGGRAPH 2004

Mesh-Based Inverse Kinematics

- Video

- Doesn’t even need a hierarchy or skeleton: Figure proper transformations out based on a few example deformations!

- Link to paper:
  Sumner, Zwicker, Gotsman, Popovic: Mesh-Based Inverse Kinematics, ACM SIGGRAPH 2005

TIEA311 - Was that cool or was that cool?

Hyperlinks were broken in PDF translation but these were easily found in YouTube based on the titles:
https://www.youtube.com/watch?v=X5Z7ZJ39zAA
(original style IK)
https://www.youtube.com/watch?v=QqJjr4eFAnc
(later research from the same group, project page: http://www.kevinwampler.com/research/2016_blendwarp/)

Important note: You must also learn to find (and, the difficult part: understand) full text research articles, available from within the university IP address space, because the **institute is paying a lot of tax payers' money to let YOU do that.** Would be a shame not to start using this privilege today!

TIEA311 - Was that cool or was that cool?

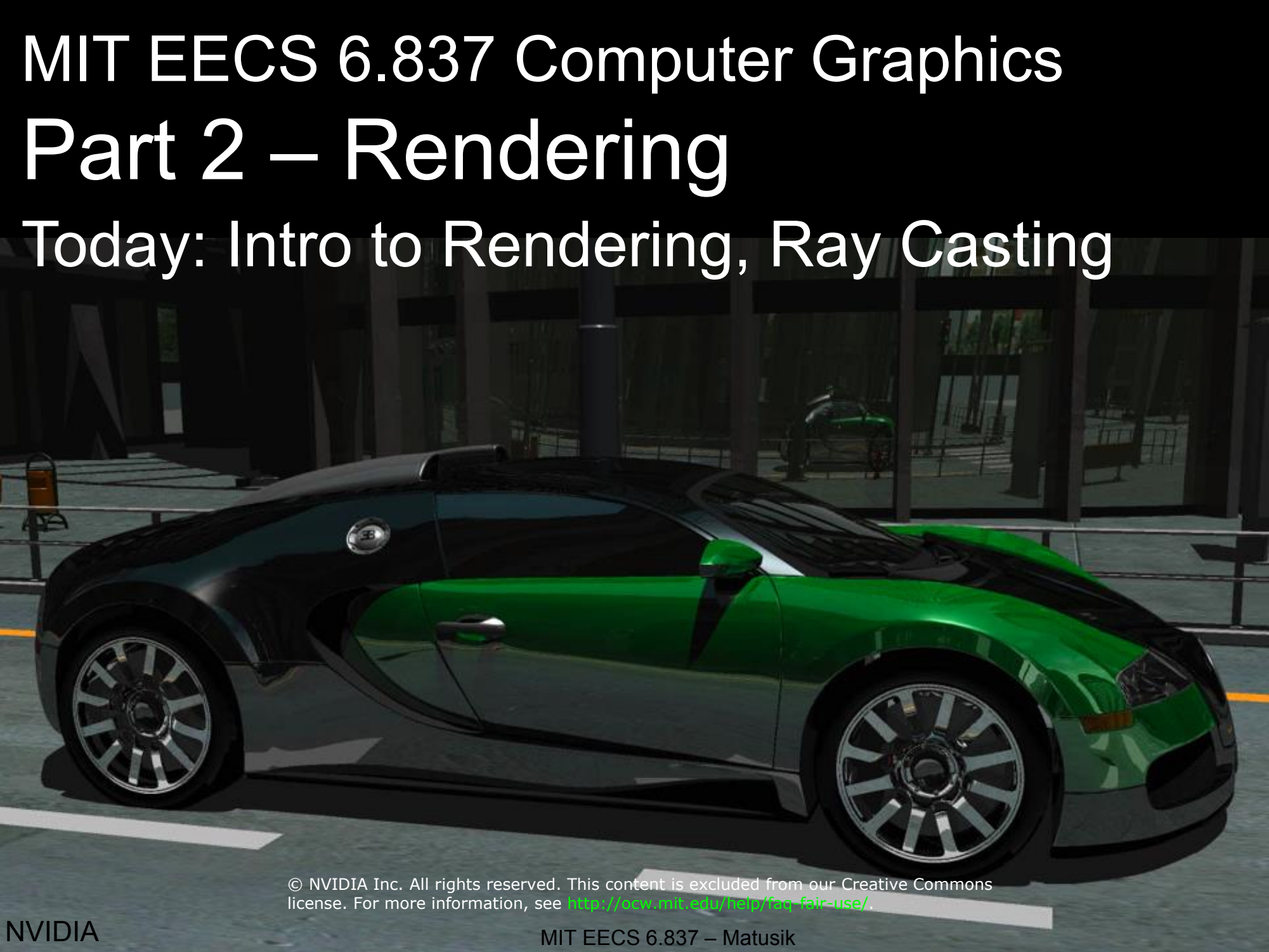
Interested in this stuff? Words of advice:

- Learn about mathematics, statistics, optimization, and numerics. (programming “comes for free” at IT Faculty)
- Just a few keywords from the video narrative: “local optimum”, “global optimum”, “probability density function”, “[machine] learning”
- Cool stuff happens at the bleeding edge of science. . .
- Your first step **can not be a sudden appearance at the top** of this mountain (or any mountain)!
- At the Faculty, **we support every step you choose to make** towards being a graphics algorithm pro (e.g., BSc, MSc theses).
- Today, I offer the first piece of support by repeating the main words of advice: Get serious about your math, and keep applying it to graphics, if that makes you tick!

MIT EECS 6.837 Computer Graphics

Part 2 – Rendering

Today: Intro to Rendering, Ray Casting



NVIDIA

The Story So Far

- Modeling
  - splines, hierarchies, transformations, meshes, etc.
- Animation
  - skinning, ODEs, masses and springs
- **Now we'll to see how to generate an image given a scene description!**

TIEA311 - Rest of the Course in Jyväskylä

The next slide is verbatim from the MIT OCW course. Here is the current plan for the remaining part of our adapted short version:

- Ray Casting: cover fully. Ray Tracing: cover the basic idea, skip details $\rightarrow$ possible to continue as a “hobby project”; NOTE: teachers of “TIEA306 Ohjelmointityö” may be contacted regarding the possibility of receiving credit for hobby projects.
- Shading, texture mapping: Cover the principles up to Phong model and texture coordinates.
- Rasterization, z-buffering: cover basic ideas
- Global Illumination, Monte Carlo techniques: skip this $\rightarrow$ could be suitable for bachelor thesis (introductory topics) or even master thesis (current state-of-the-art techniques)
- Image-based rendering: skip $\rightarrow$ defer to advanced courses (e.g. “TIES411 Konenäkö ja kuva-analyysi”)
- Sampling and antialiasing: mostly skip $\rightarrow$ defer theory to “TIES324 Signaalinkäsittely” and techniques (possibly) to “TIES471 Reaaliaikainen renderöinti”.
- Shadow techniques, Graphics Hardware $\rightarrow$ defer to TIES471.

The Remainder of the Term

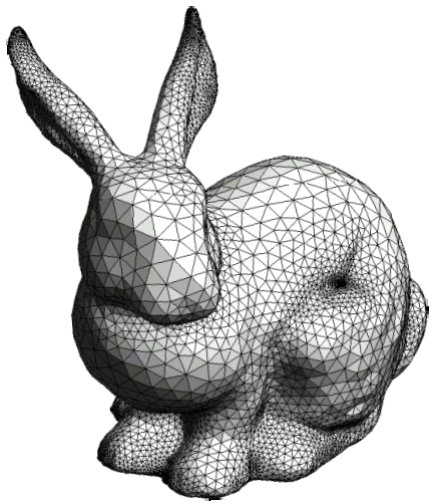
- Ray Casting and Ray Tracing
- Intro to Global Illumination
  - Monte Carlo techniques, photon mapping, etc.
- Shading, texture mapping
  - What makes materials look like they do?
- Image-based Rendering
- Sampling and antialiasing
- Rasterization, z-buffering
- Shadow techniques
- Graphics Hardware

[Lehtinen et al. 2008]

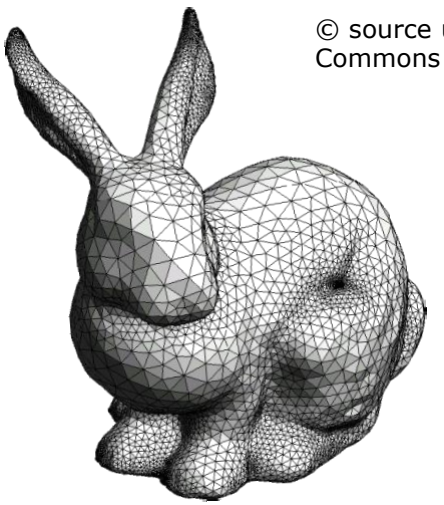
Today

- What does *rendering* mean?

- Basics of ray casting



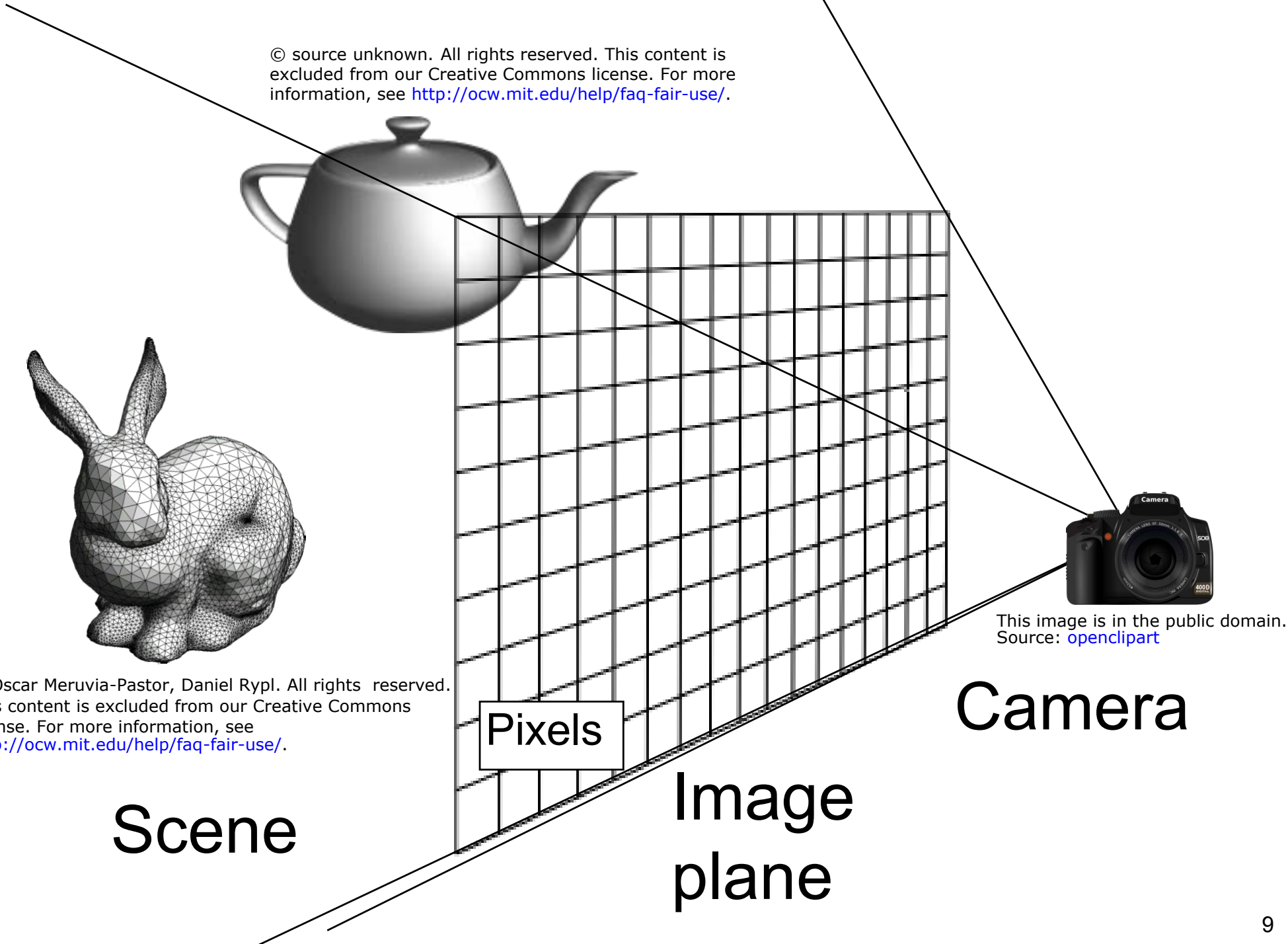
Scene



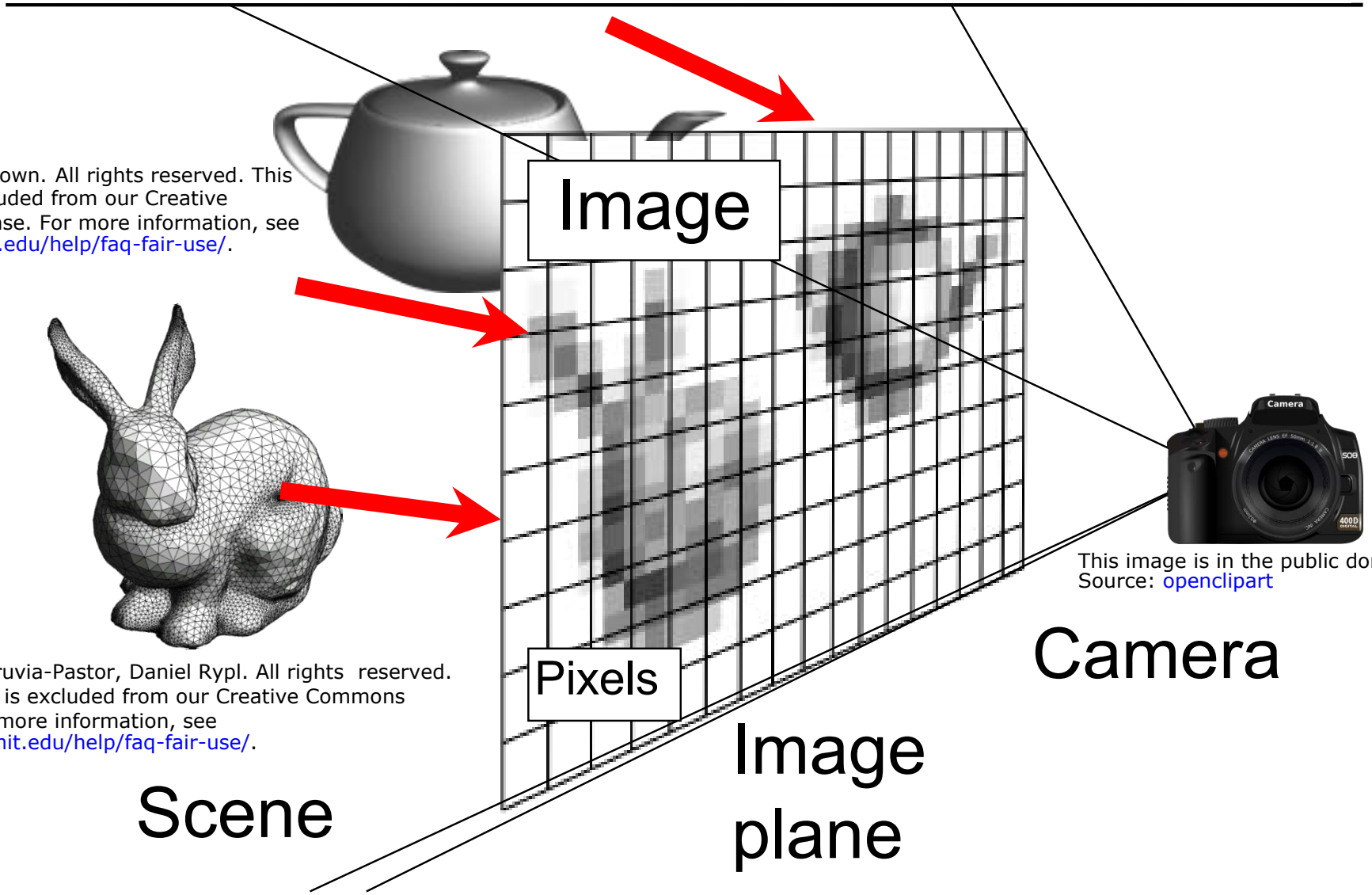
Scene

This image is in the public domain.
Source: openclipart

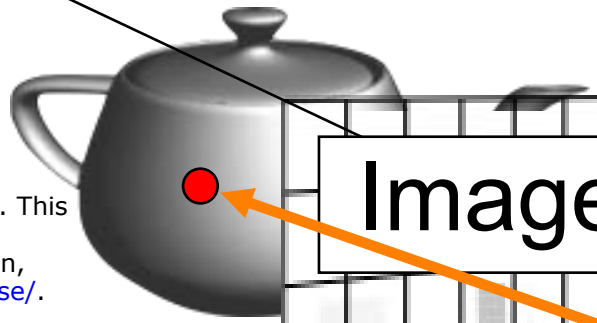
Camera


© source unknown. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.
Camera
This image is in the public domain.
Source: openclipart
© Oscar Meruvia-Pastor, Daniel Rypl. All rights reserved. This content is excluded from our Creative Commons license. For more information, see http://ocw.mit.edu/help/faq-fair-use/.
Pixels
Camera
Scene
Image plane

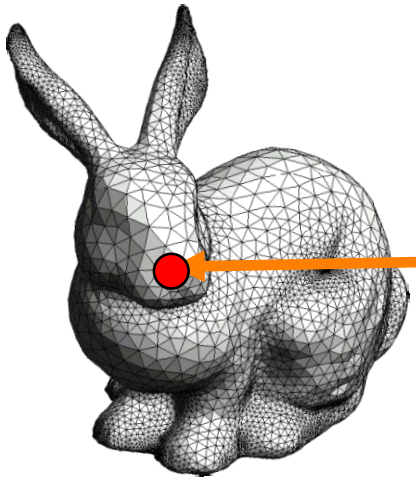
Rendering = Scene to Image



Rendering – Pinhole Camera



Image



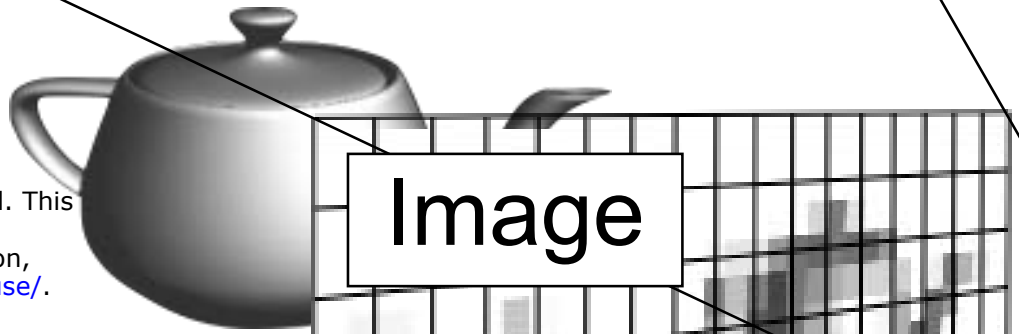
This image is in the public domain.
Source: openclipart

Pixels

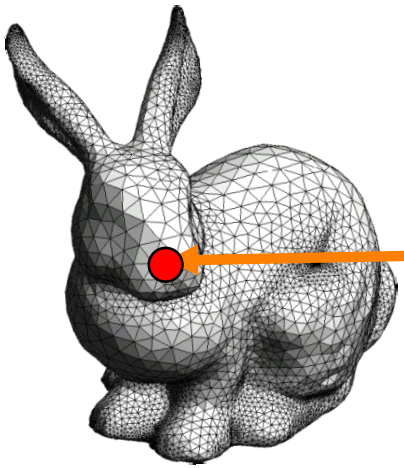
Scene

Each pixel corresponds to one ray. We need to figure out which scene point each one hits.

Rendering



Image



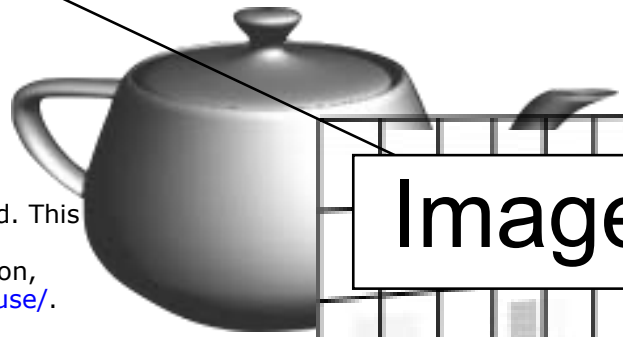
This image is in the public domain.
Source: openclipart

Pixels

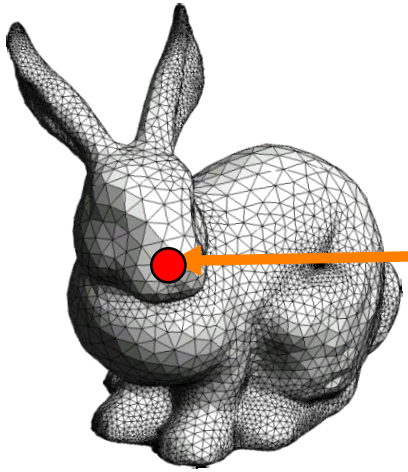
Scene

What's the color you put in each pixel?

Rendering



Image



This image is in the public domain. Source: openclipart

Pixels

Pixel Color Determined by *Lighting/Shading*

Scene

Rendering

- "Rendering" refers to the entire process that produces color values for pixels, given a 3D representation of the scene
- Pixels correspond to rays; need to figure out the **visible** scene point along each ray
  - Called "hidden surface problem" in older texts
  - "Visibility" is a more modern term
  - Also, we assume (for now) a single ray per pixel

Rendering

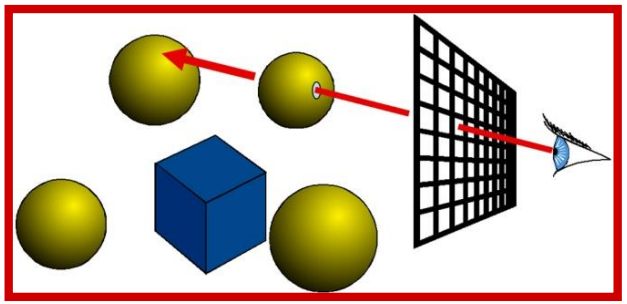
- "Rendering" refers to the entire process that produces color values for pixels
- Pixels correspond to rays; need to figure out the **visible** scene point along each ray
  - Called "hidden surface problem" in older texts
  - "Visibility" is a more modern term
  - Also, we assume (for now) a single ray per pixel
- Major algorithms: **Ray casting and rasterization**

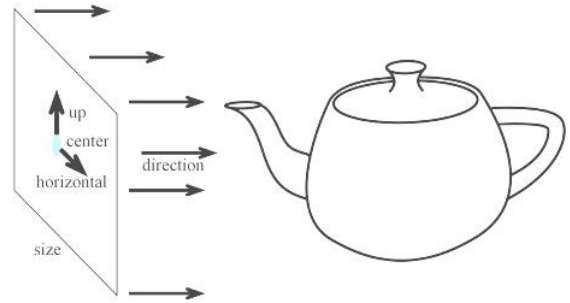
- Note: We are assuming a pinhole camera (for now)

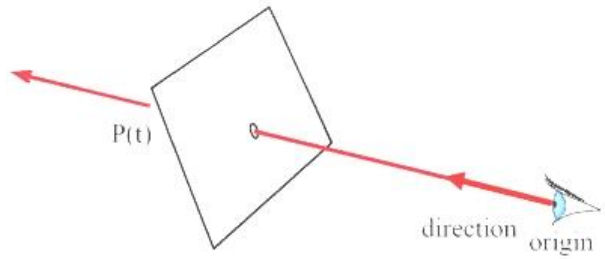
Questions?

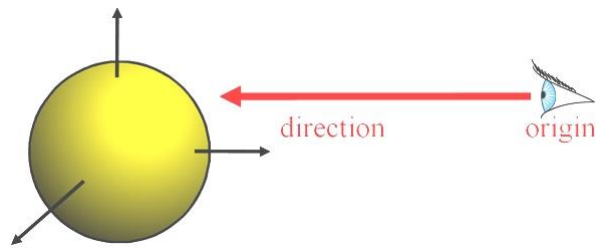
Ray Casting

- Ray Casting Basics
- Camera and Ray Generation
- Ray-Plane Intersection
- Ray-Sphere Intersection



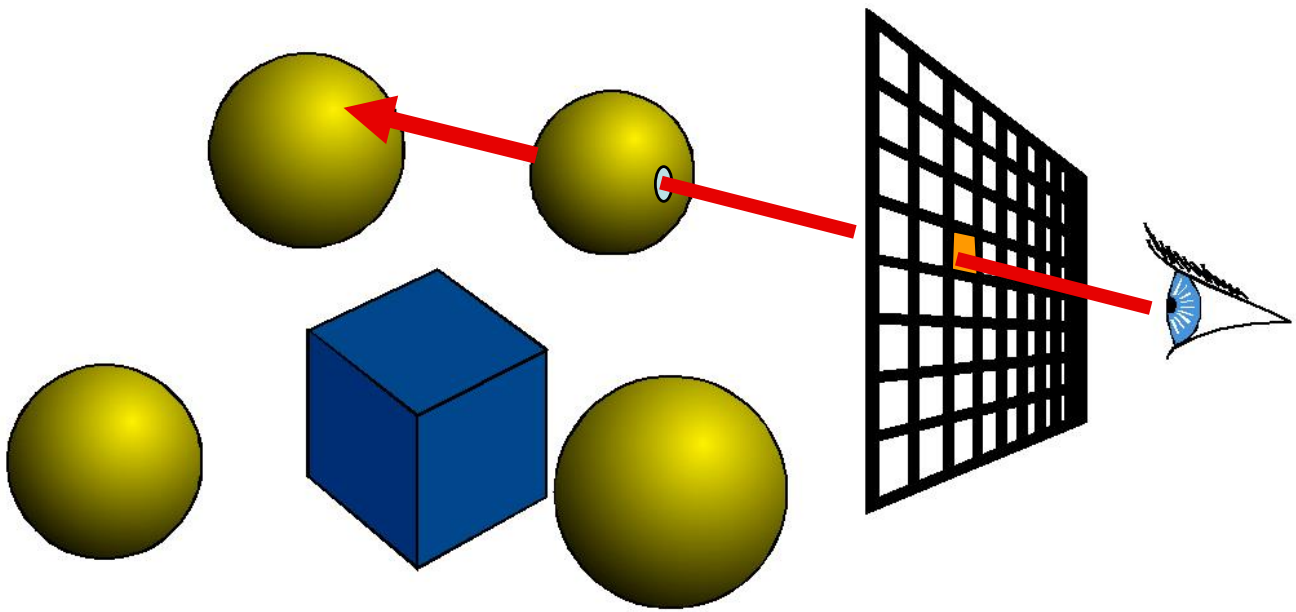






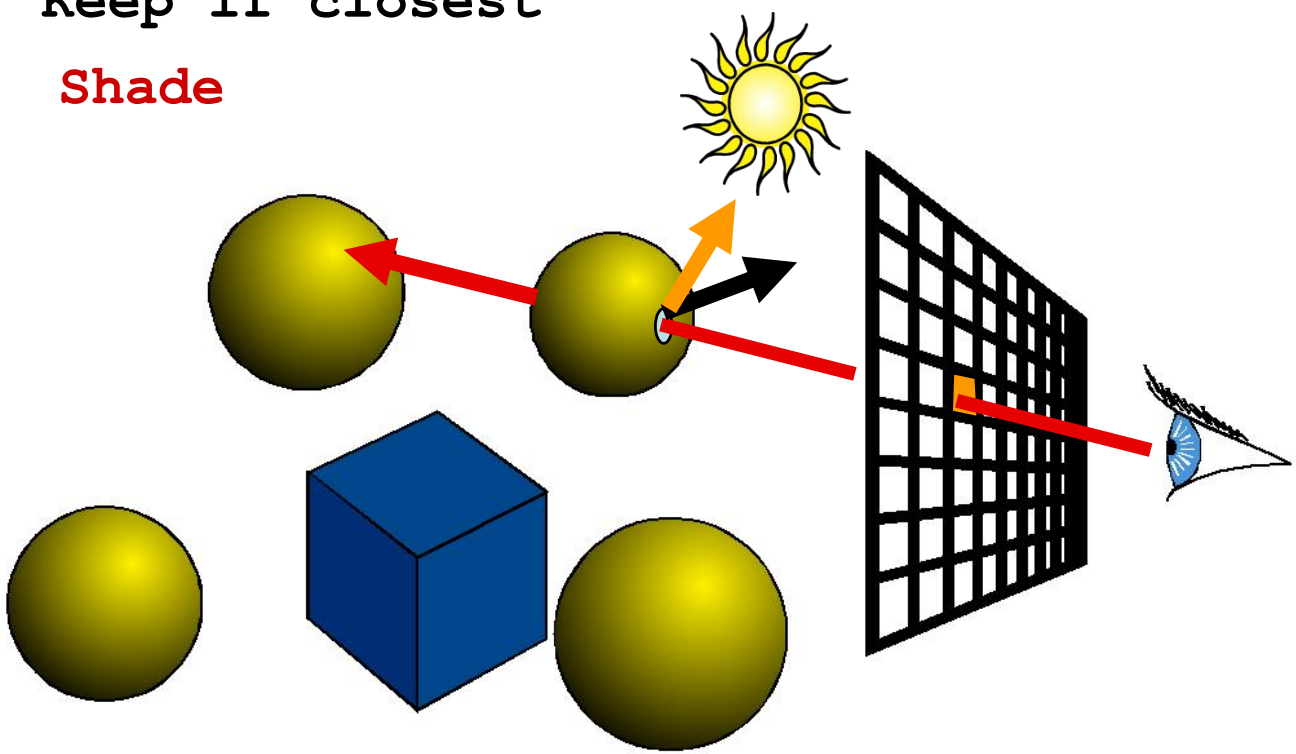
Ray Casting

```
For every pixel
   Construct a ray from the eye
   For every object in the scene
       Find intersection with the ray
       Keep if closest
```



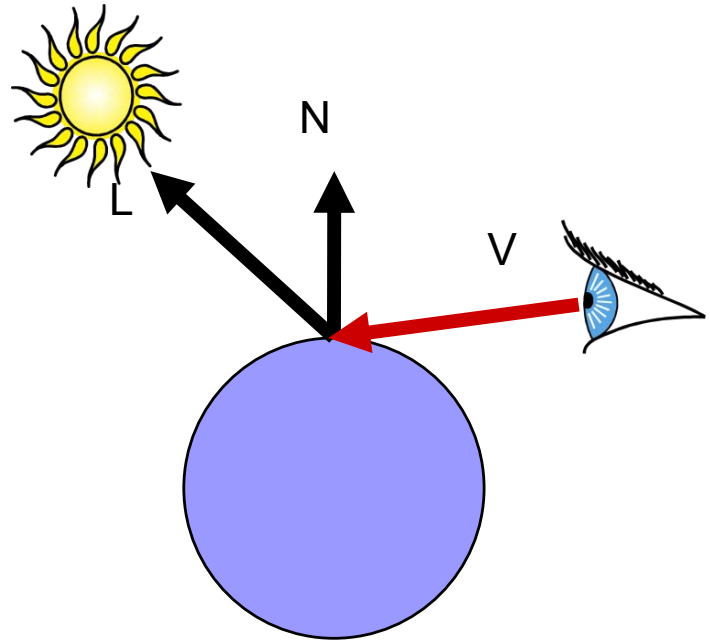
Shading

```
For every pixel
   Construct a ray from the eye
   For every object in the scene
       Find intersection with the ray
       Keep if closest
        Shade
```

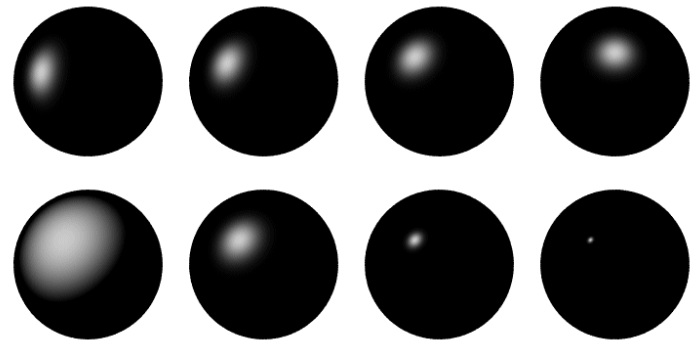


Shading = What Surfaces Look Like

- Surface/Scene Properties
  - surface normal
  - direction to light
  - viewpoint
- Material Properties
  - Diffuse (matte)
  - Specular (shiny)
  - …
- Light properties
  - Position
  - Intensity, ...
- Much more!



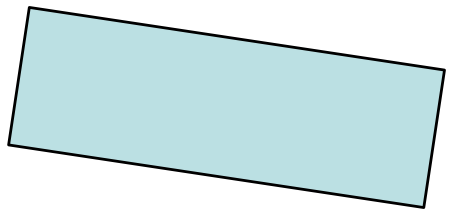
Diffuse sphere

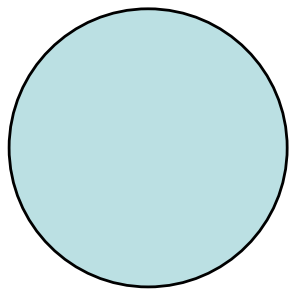


Specular spheres

Ray Casting vs. Ray Tracing

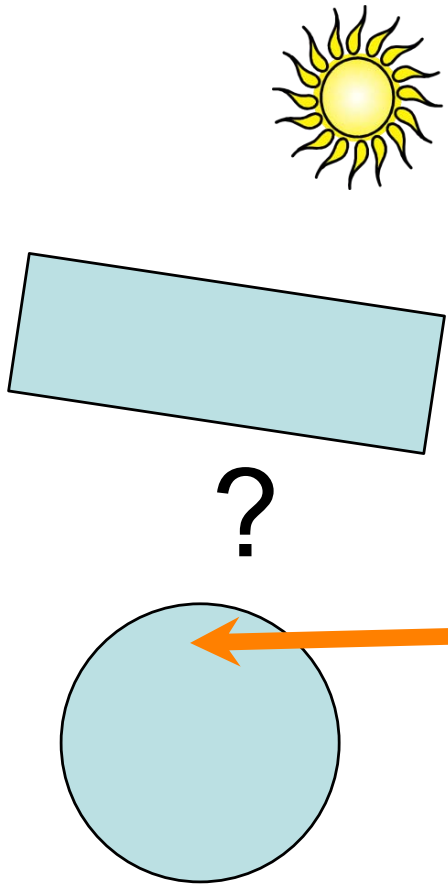
- Let's think about shadows...





This image is in the public domain.
Source: openclipart

Ray Casting vs. Ray Tracing

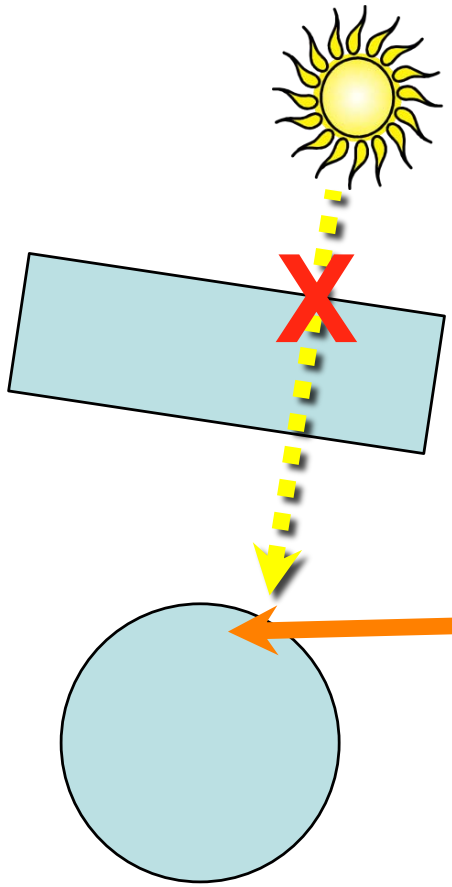


?

This image is in the public domain.
Source: openclipart

"camera rays" are rays from the camera to the scene

Ray Casting vs. Ray Tracing

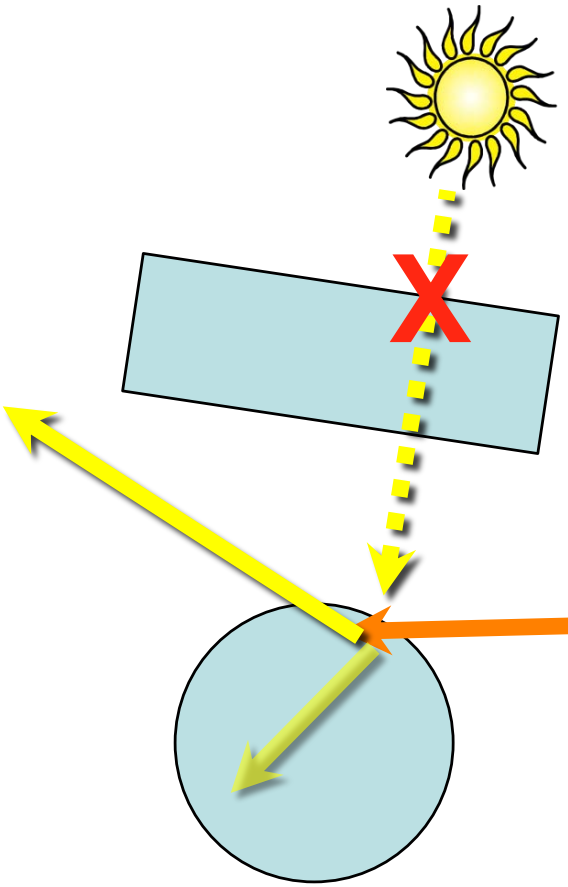


ray from light to hit point is blocked, i.e., **point is in shadow**

This image is in the public domain.
Source: openclipart

Ray Casting vs. Ray Tracing

- Ray casting = eye rays only, tracing = also secondary

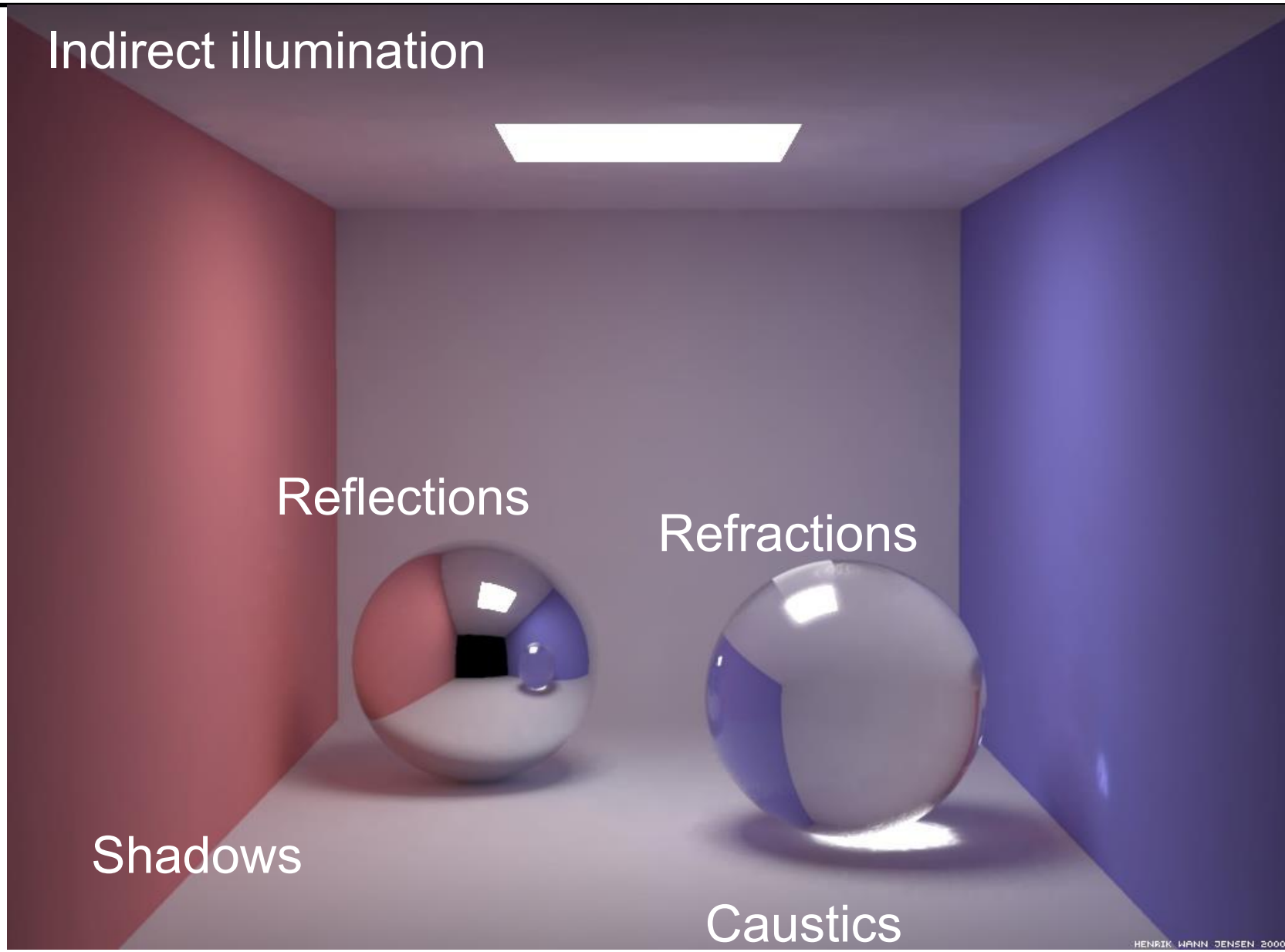


Secondary rays are used for testing shadows, doing reflections, etc.

This image is in the public domain.
Source: openclipart

We'll do all this a little later!

Secondary Rays

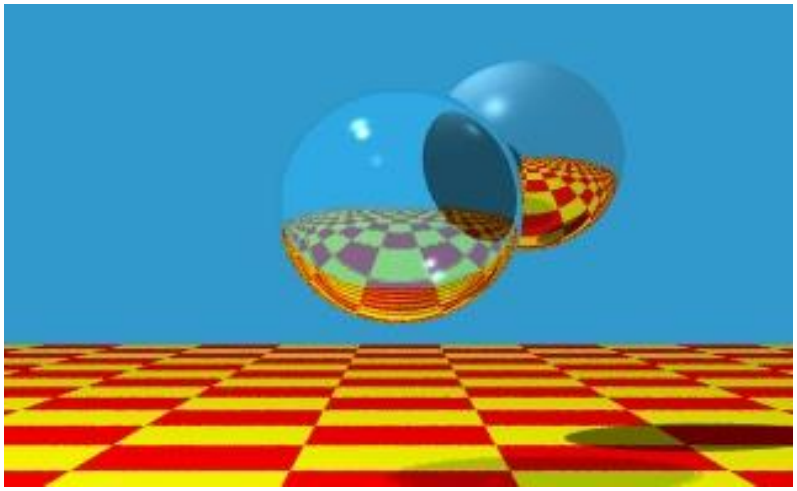


Courtesy of Henrik Wann Jensen. Used with permission.

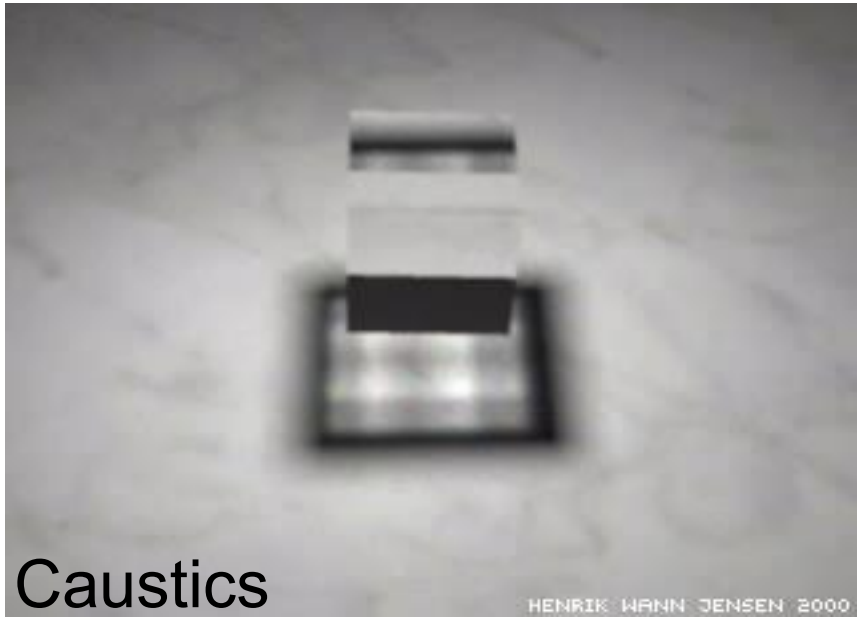
Henrik Wann Jensen

Ray Tracing

Courtesy of Henrik Wann Jensen. Used with permission.



Reflections, refractions



Questions?

TIEA311 - Today in Jyväskylä

The time allotted for this lecture is now over.

Now: Break until tomorrow morning. Sleep if you have time.

But also **try to wake up and come to the lecture**!

We will pick up our thoughts soon enough!