# Welcome!

**Graphics 2008/2009, period 1**

Wolfgang Hürst

*Information and Computing Sciences*
*Game, Media and Agent Technology*
*Multimedia and Geometry Research Group*

# Outline

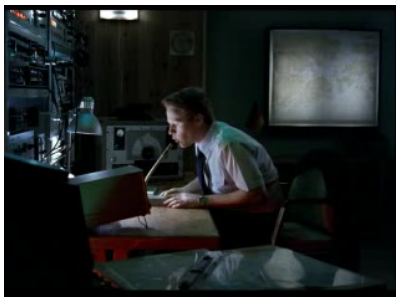
# Introduction

**Wolfgang Hürst**
Email: huerst@cs.uu.nl
Office: Centrumgebouw Noord, A205
Phone: +31 (30) 253 1806
Research group: Multimedia and geometry
(part of game, media and agent technology)

# Language

Language of the course:

- Lecture, tutorials, and programming labs: English (But TAs speak Dutch)
- Exams: English, but answers in Dutch are ok



Video "German coast guard", from YouTube at http://www.youtube.com/watch?v=rD4roXEY8hk

# Lectures (college)

- Tue 11:00–12:45
  Thu 15:15–17:00
- Textbook: *Fundamentals of Computer Graphics*, Peter Shirley (2nd edition). Available from A-Eskwadraat (www.a-eskwadraat.nl), and from Selexyz Broese (the store next to the Basket bar on campus).
- Other resources might be posted on the web
- 12 lectures, cf. schedule on the web site

# Lectures (college)

- Lectures will be recorded
  (audio and synchronized screen capture)
- Recordings will be available on the web for download
  (usually within a day, except for today's lecture)
- Very important to keep in mind:
  - Technology can fail (and most likely will)
  - We might stop this without further notice

  So, don't rely on this!
- Think carefully of how you want to take advantage of this offer!

# Tutorials (werkcollege)

- TA's: Rene Kersten (and Dirk-Jan Hoppenbrouwer, Jalmer van de Berg)
- Tue 09:00–10:45 (groups 1 & 2)
  Thu 13:15–15:00 (groups 3 & 4)
- 6 exercises, cf. schedule on the web site

# Programming labs (practicum)

- TA's: Corijn Kamerling, Dirk-Jan Hoppenbrouwer, Jalmer van de Berg
- Thu 13:15–15:00 (groups 1 & 2)
  Tue 09:00–10:45 (groups 3 & 4)
- Two assignments P1 and P2
- Deadlines: TBA
- Grade: P = ( P1 + P2 ) / 2
- Deadlines are strict (submission details: see website)

# Programming labs (practicum)

Rules and exceptions:

- Assignments must be done in teams of two students
- If you have trouble finding a partner: contact your TA
- No reusage of last year's code
  (be warned: we use software to verify that!)
- Exceptions: students who did this course last year
  - Grades from last year don't count!
  - You are allowed to work alone or with a partner
  - If you work allone or with a partner who also did this course last year: you can reuse your code from last year (but same grade not guaranteed)
  - Otherwise: new code required

# Exams

- Exams (tentative!):

  T1 Thu, Oct 2, 2008
  T2 Thu, Nov 6, 2008

- Grade: T = ( T1 + T2 ) / 2
- Grades of last year are not valid.
- If you can't make it to the exams for very good reasons (skiing holidays don't count!), make sure to notify me a.s.a.p., and be prepared to backup your claims with evidence.
- Coverage: Will be announced on website (most likely T1 = lecture 1-6, T2 = lecture 7-12, tentative!)

# Grading

Final grade = ( 2T + P ) / 3

- Minimum grades needed to pass: T and P at least 5.0, final grade at least 6.0 (note: for the final grade, 5.5 is rounded to 6.0)
- Make-up assignment P3 may replace either P1 or P2, but only if you handed in both assignments, took both exams, and got a total grade of at least 4.0.
- Make-up exams Tm1 and Tm2 may replace either T1 and T2, respectively, but only if you handed in both assignments, took both exams, and got a total grade of at least 4.0.
- Date for make-up exams: ~~Thu~~, Dec 23, 2008 (Tuesday)
- Refer to the web site for further info

# Outline

2. What is computer graphics, and why do we study it?
   - Topics
   - Applications
3. Graphics in the ICS curriculum
4. Basic methods for image generation
   - Projection vs. ray tracing
   - Projective methods
   - Ray tracing / ray casting
   - Basic mathematics for ray tracing

# What is computer graphics?

**Computer Graphics** describes any use of computers to create or manipulate images *(definition from the textbook)*.

Keyphrase here: "create images", i.e. represent an image of a virtual world of graphics objects from a specific view

Related to, but different from

- Image processing
- Computer vision

Other closely related fields:

- Interaction/HCI, VR, visualization, ...

# So, what exactly is computer graphics now?

Computer graphics: create an image virtual world of graphical objects from a specific view on a graphics device (usually: raster display)

*(movie source: http://www.bigbuckbunny.org)*

Major areas of computer graphics

- Modeling (not much in this course)
  *mathematical specification of shape and appearance*
- Rendering
  *creation of shaded images from 3D computer models*
- Animation (not covered in this course)
  *create an illusion of motion through a sequence of images*

# What will we do here?

Two approaches for teaching graphics:

- Basics and backgrounds (this course)
- vs. applications and libraries (not here)

Other things we will miss:

- Graphics hardware
- and many other things (animation, interfaces, . . . )

# Why do we study computer graphics?

Applications in:

- Virtual reality (e.g., remote access to museum collection)
- Movie industry (special effects, 3D animations, cartoons)
- Games
- Information visualization
- Simulation (flight & car simulators, fire-fighting, . . . )
- CAD/CAM, architectural design (walk-through)
- Medical Imaging

# Related courses

- WIS, IMP, DS, MSO
- DDM, OPT
- Master GMT: GA, DDM, AGRA, VW
- Information Science: A3DM (user perspective)
- many more
- Experimentation project/ Thesis project

# Basic methods for image generation

- Modelling (creating 3d virtual worlds)
- Rendering (creating 2d images from 3d models)

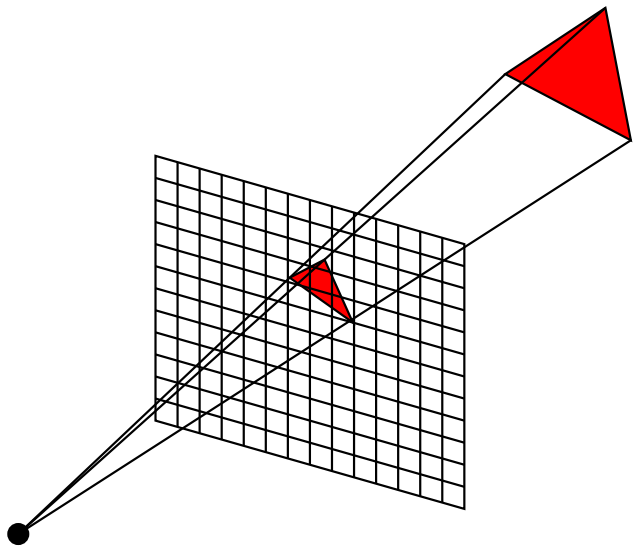
Two basic methods:

- Projective methods
- Ray tracing

# Projective methods

A popular method for generating images from a 3D-model is projection:

- Used by OpenGL and DirectX
- Real-time applications (games)
- Very fast: hardware supported

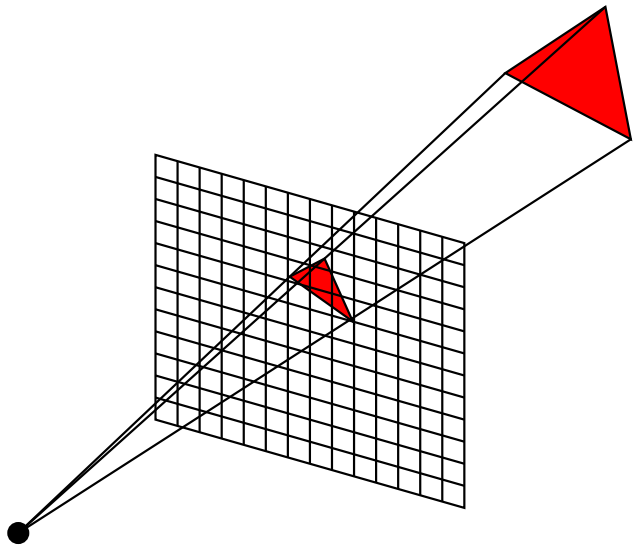


# Projective methods

Idea:

- 3D triangles project to 2D triangles
- Project vertices
- Fill/shade 2D triangle

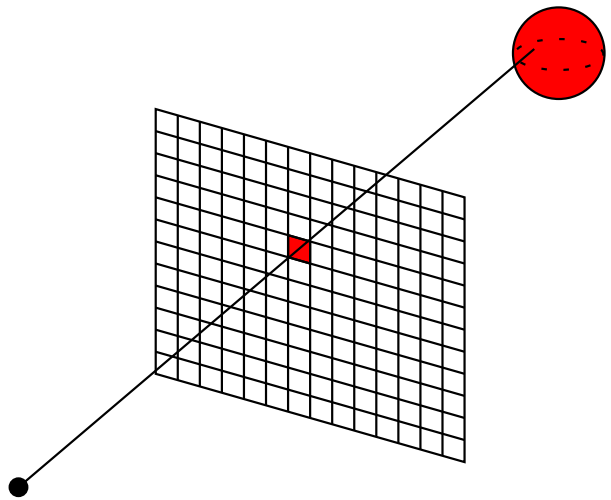
Q: what if two or more triangles project onto the same pixels?



# Ray tracing / ray casting

For photo-realistic rendering of 3D-models, usually ray tracing algorithms are used:

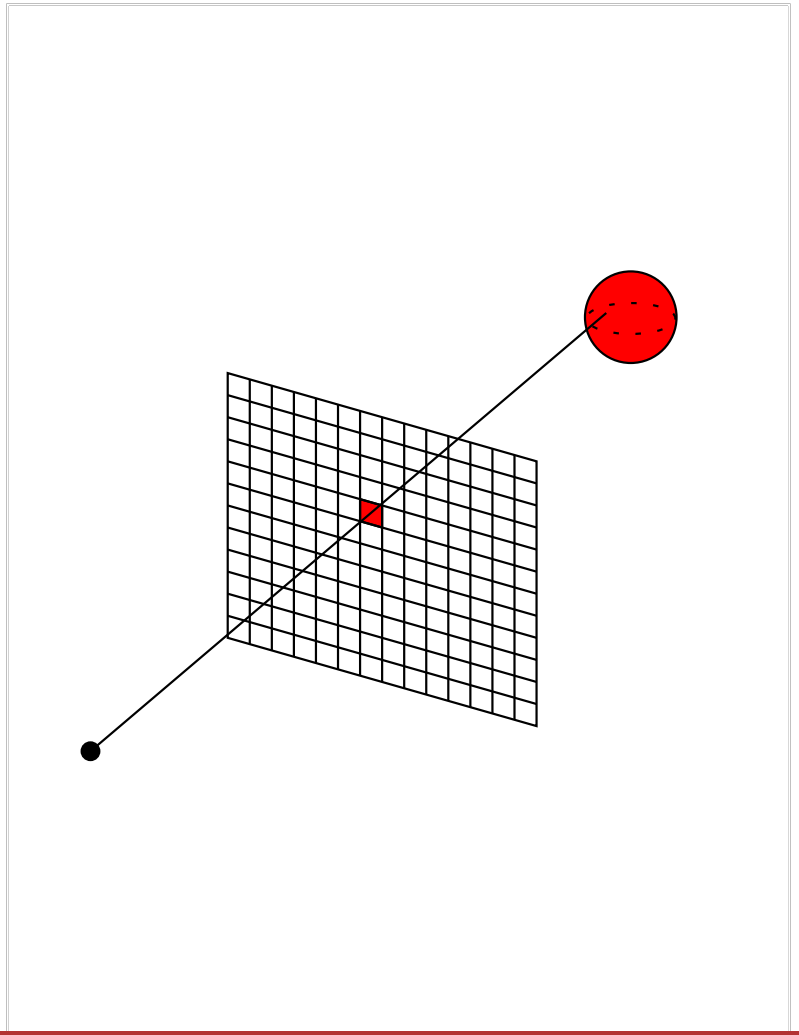
- Global Illumination
- Traditionally (very) slow
- Recent developments: real-time ray tracing



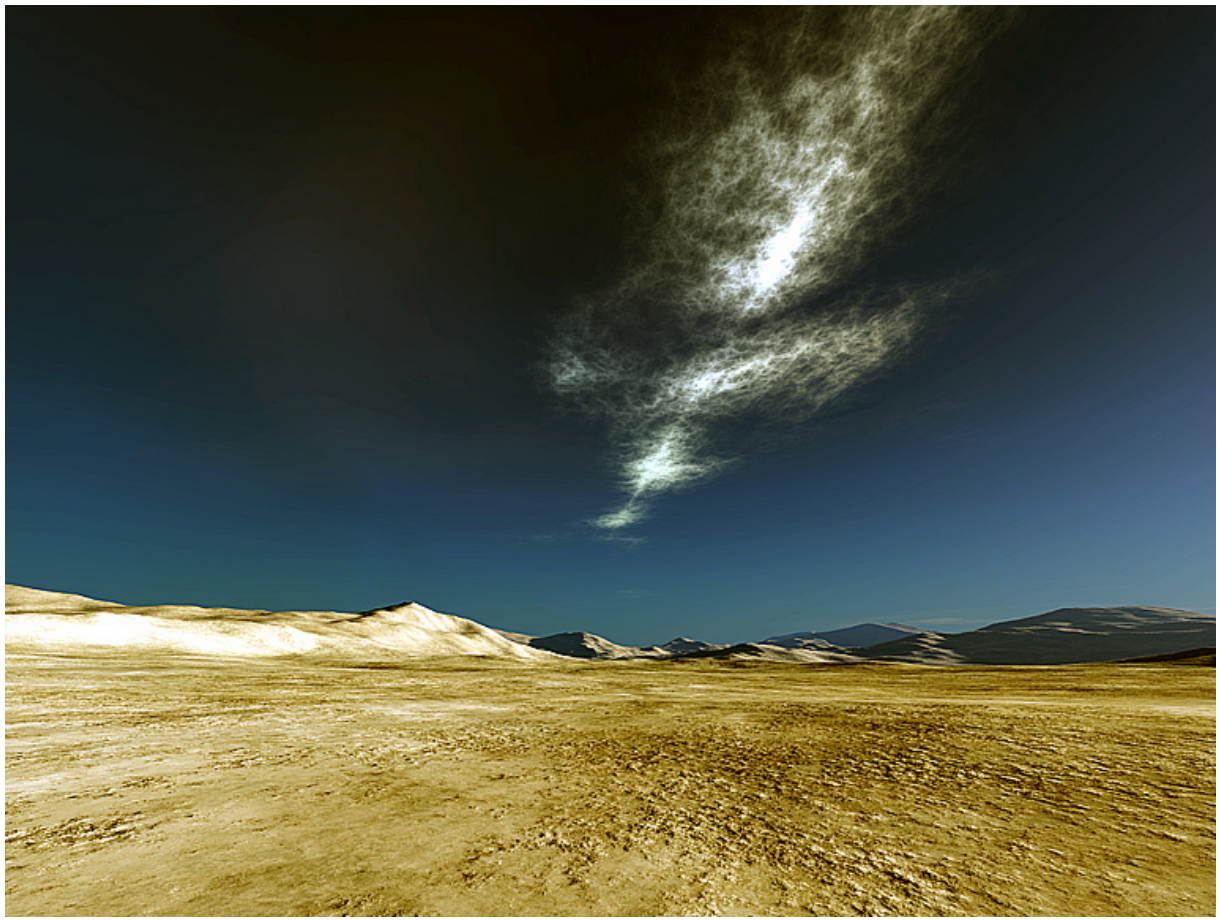
# Ray tracing / ray casting

Idea: for every pixel

- Compute ray from viewpoint through pixel center
- Determine first object hit by ray (including intersection point)
- Calculate shading for the pixel (possibly with recursion)

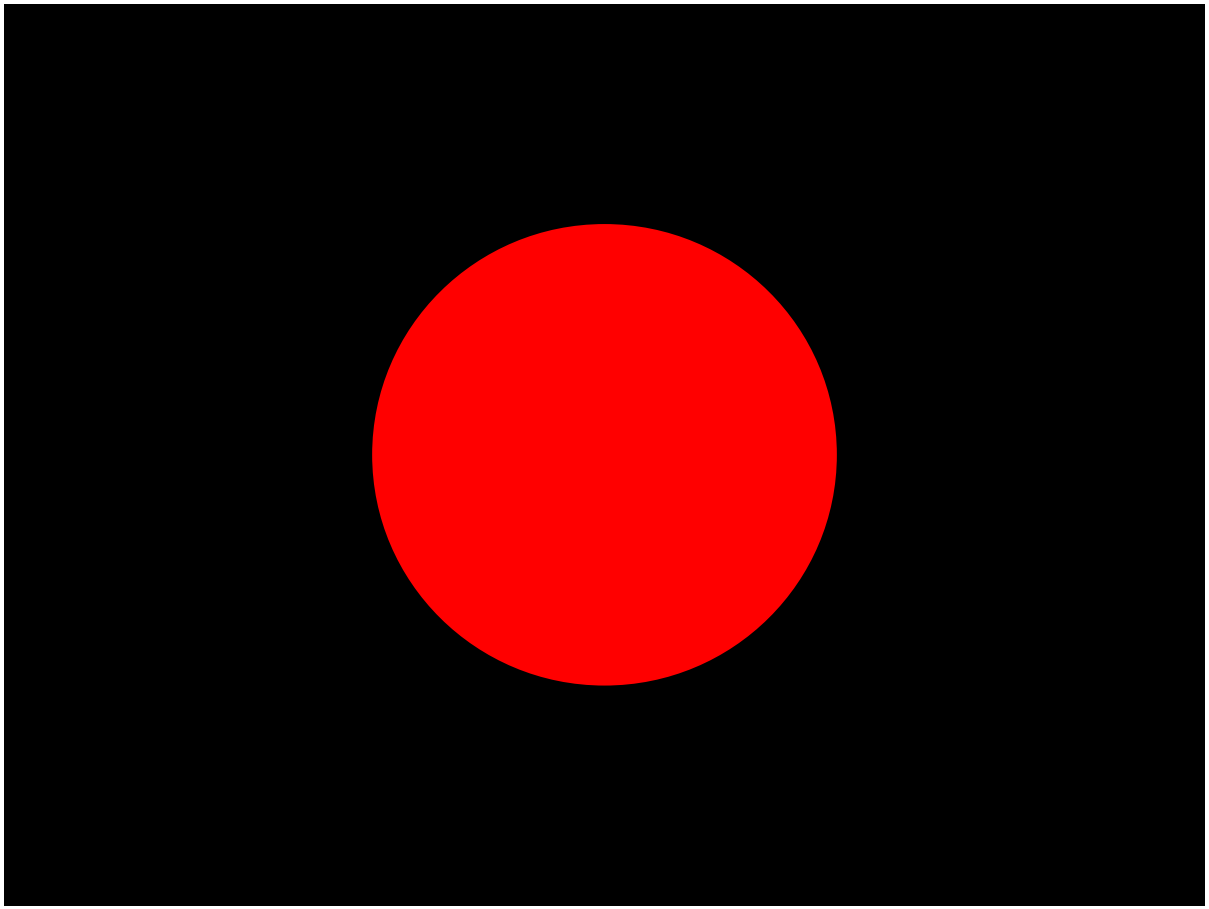


# Examples: landscapes (www.terragen.org)

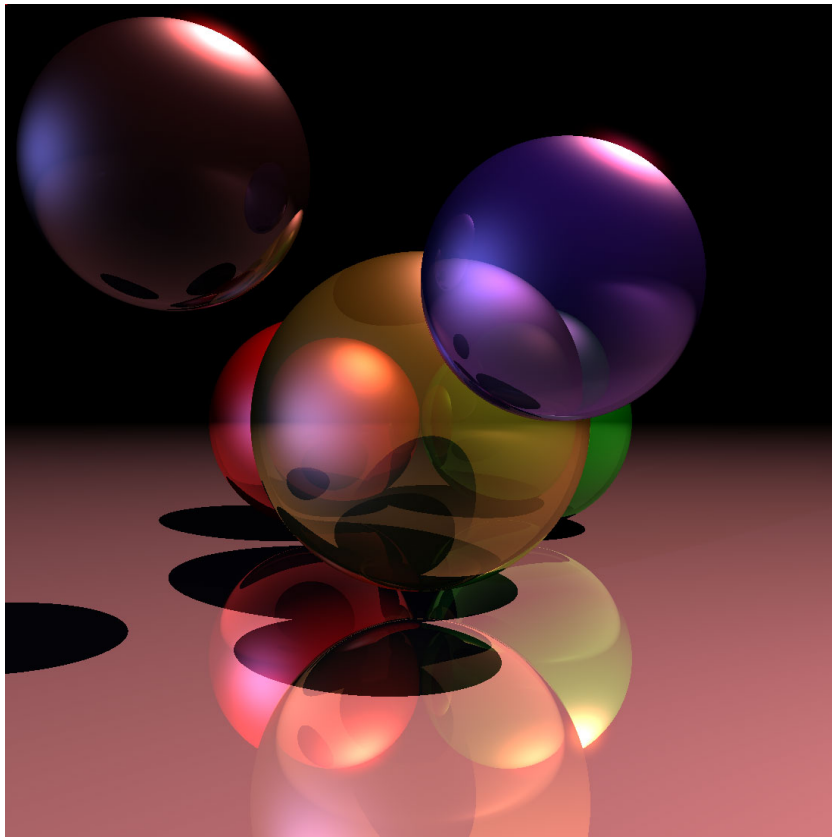


# Examples: office (www.povray.org)

# Examples: spheres (you!)



# Examples: spheres (you!)

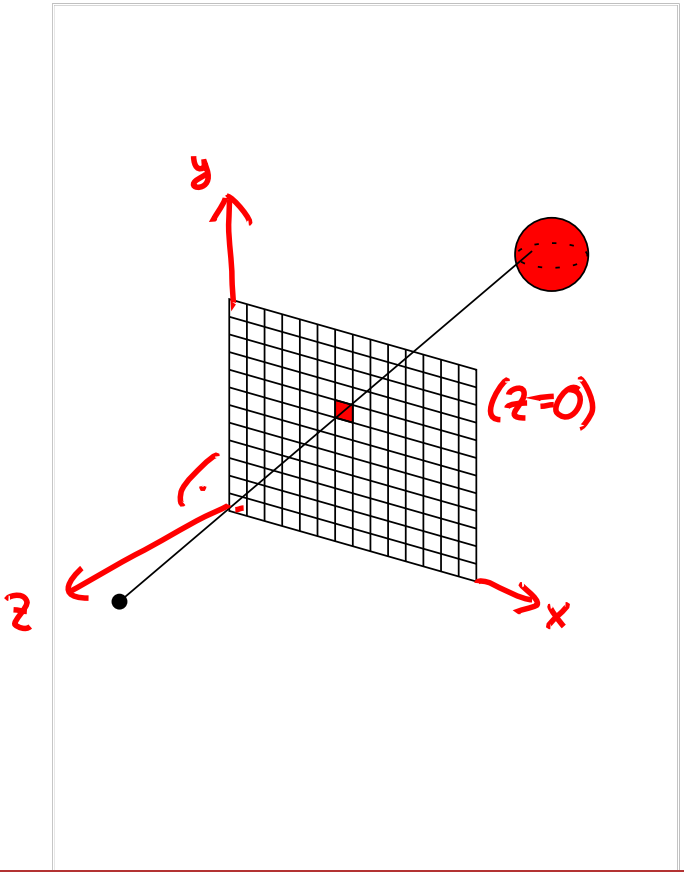


# Basic mathematics for ray tracing

Initial assumption: simple camera, projection rectangle parallel to the XY-plane, centered around the Z-axis. Objects are spheres.

We need the following:

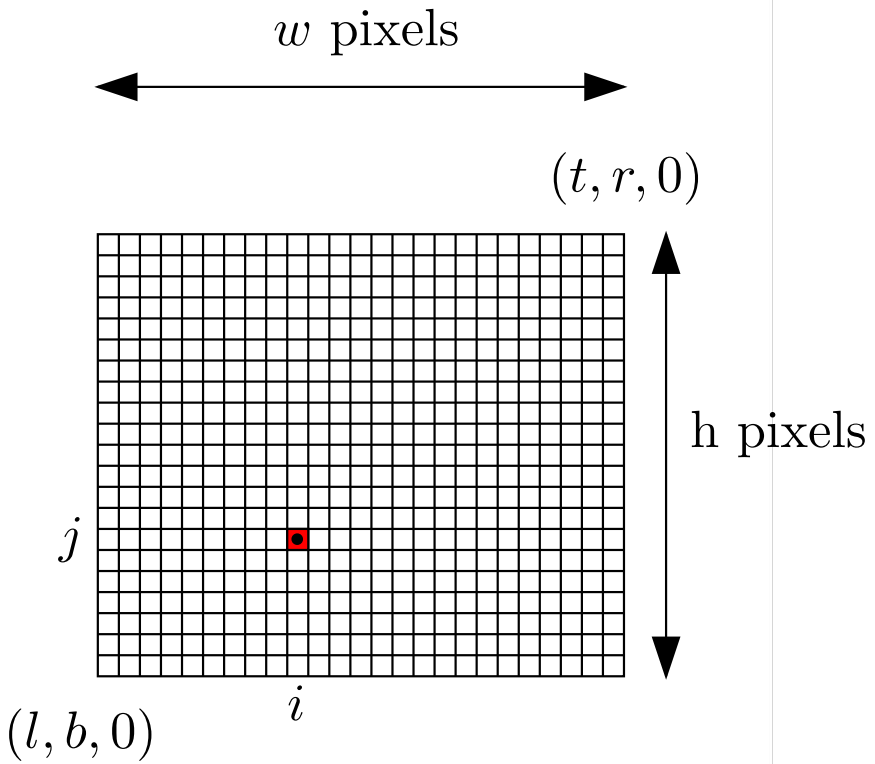
- Viewpoint coordinates: origin
- Pixel center coordinates
- Ray equation
- (First) intersection point of ray and obstacle



# Pixel center coordinates

To calculate the coordinates of the center of $\mathrm{pixel}[i, j]$ we need to know

- The $x$- and $y$-coordinates of the corners of the projection rectangle
- The resolution of the image
- The order of the pixel rows (up or down)

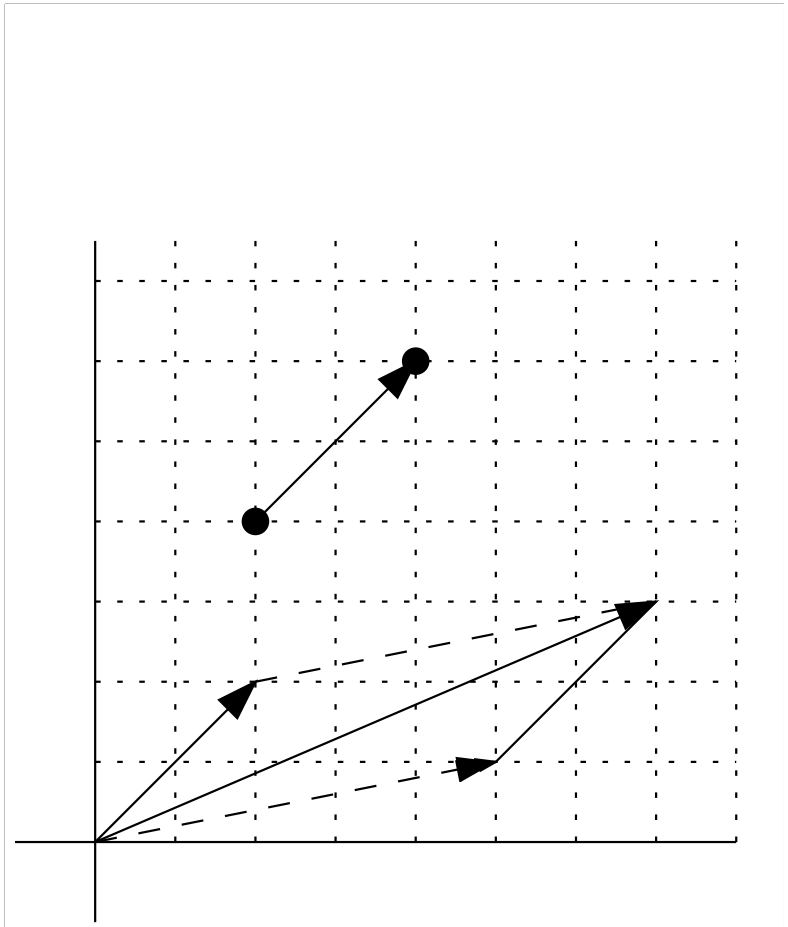


# Points and vectors

3D vectors and points can both be represented with 3-tuples $(x, y, z)$ but they are not the same!

- Vectors can be added, points can't
- Vectors can be added to points

*Note: Vectors will be discussed in detail in the next lecture*



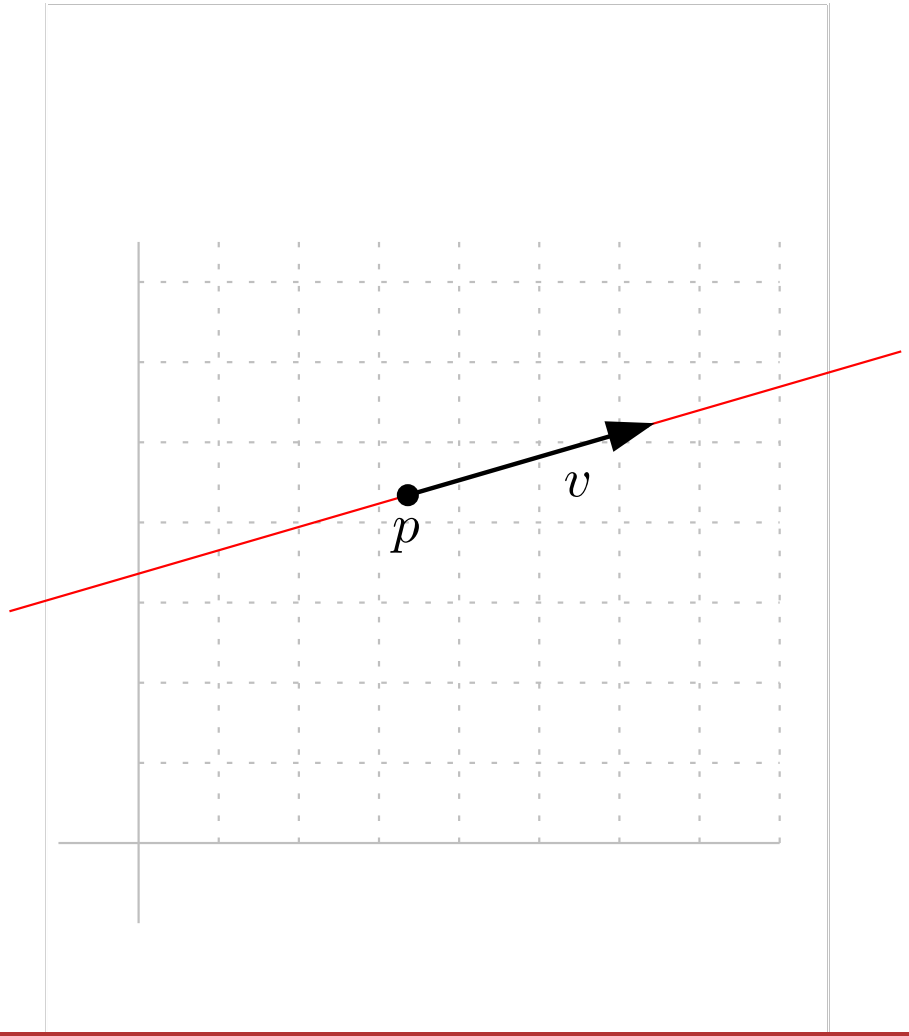
# Lines and rays

Parametric equation of a line:

- $p + t \cdot v$
- $(p_x, p_y, p_z) + t(v_x, v_y, v_z)$
- $(p_x + t \cdot v_x, p_y + t \cdot v_y, p_z + t \cdot v_z)$

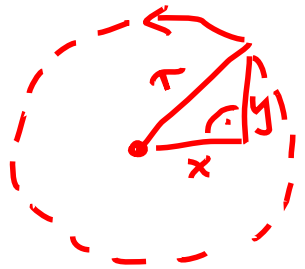
(Where $p$ is a point, and $v$ is a (direction) vector).

For a ray, $t$ is bounded (usually from below by 0).



# Spheres

2D:

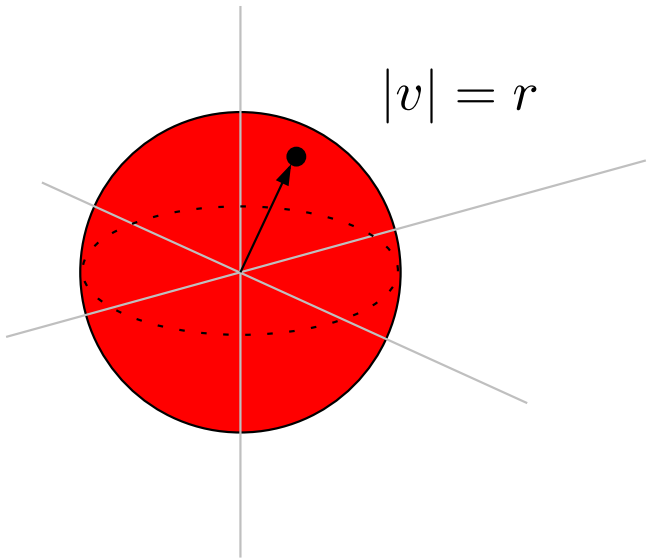



$x^2 + y^2 = r^2$

For points $(x, y, z)$ on a sphere centered at the origin and with radius $r$, we have

$$x^2 + y^2 + z^2 = r^2$$

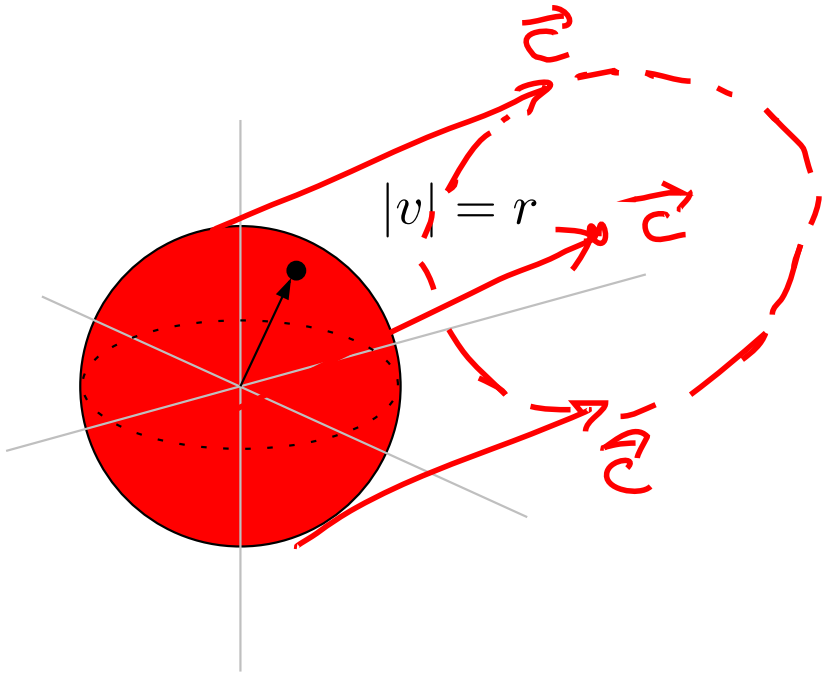
(This is basically Pythagoras' Theorem)



# Spheres

If the sphere is centered at the point $c = (c_x, c_y, c_z)$, the equation becomes

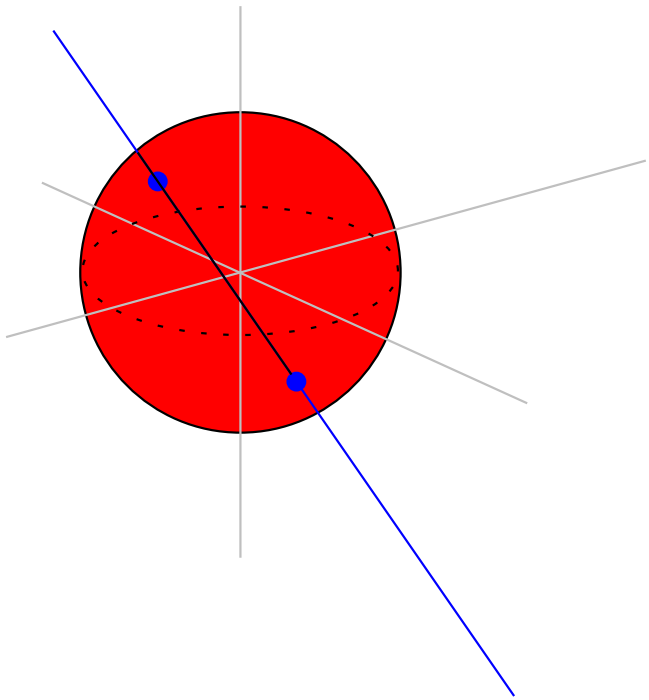
$$(x-c_x)^2+(y-c_y)^2+(z-c_z)^2 = r^2$$



# Intersections between rays and spheres

Given a sphere and a ray, with respective equations
$(x-c_x)^2+(y-c_y)^2+(z-c_z)^2 = r^2$
and
$p + t \cdot v, \quad t \geq 0$
the intersection point(s) satisfy both equations:

$$(p_x + t \cdot v_x - c_x)^2 +$$
$$(p_y + t \cdot v_y - c_y)^2 +$$
$$(p_z + t \cdot v_z - c_z)^2 = r^2$$

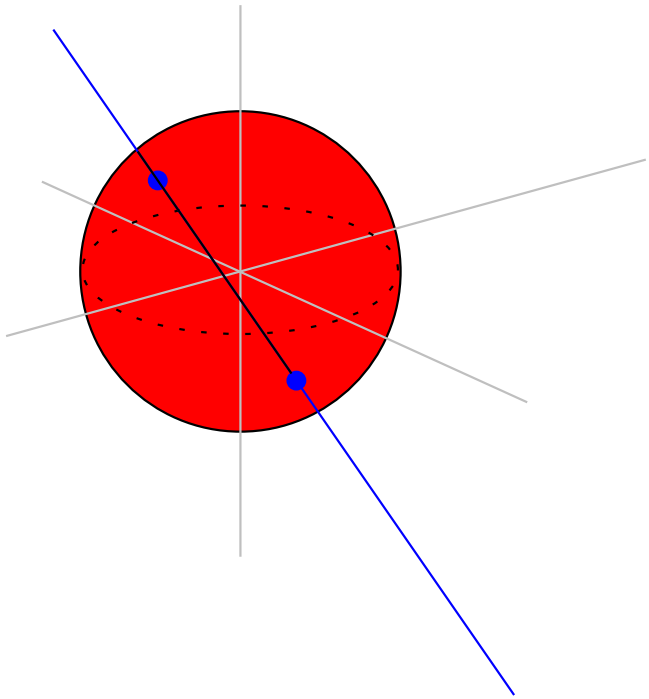


# Intersections between rays and spheres

$At^2 + Bt + C = 0 \quad \curvearrowright \quad t = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$

$$
\begin{aligned}
&(p_x + t \cdot v_x - c_x)^2 + \\
&(p_y + t \cdot v_y - c_y)^2 + \\
&(p_z + t \cdot v_z - c_z)^2 = r^2
\end{aligned}
$$

This is a quadratic equation in $t$, with 0, 1, or 2 solutions.



# Intersections between rays and spheres

Q: what is the geometric interpretation of the number of solutions?

Q: what if one of the solutions is less than zero?