Sampling and Object intersection

Michael Doggett
Department of Computer Science
Lund university

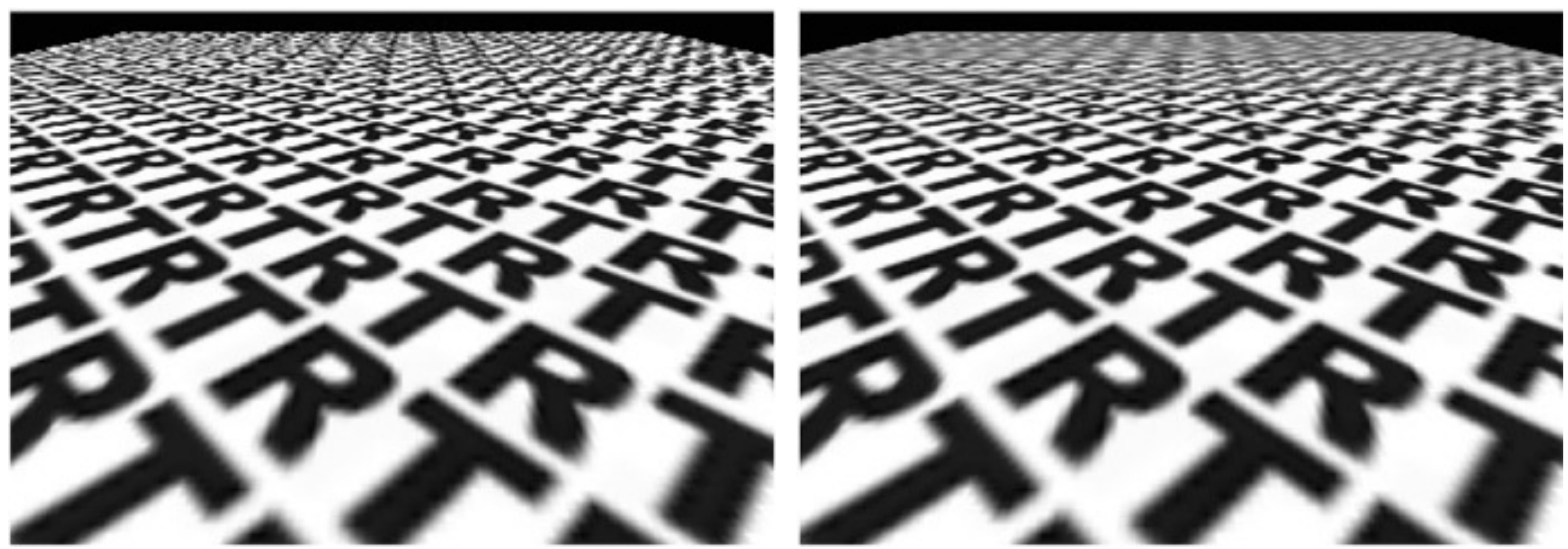
Today

- Sampling
- Object intersection
  - Spheres
  - Triangles
  - ...

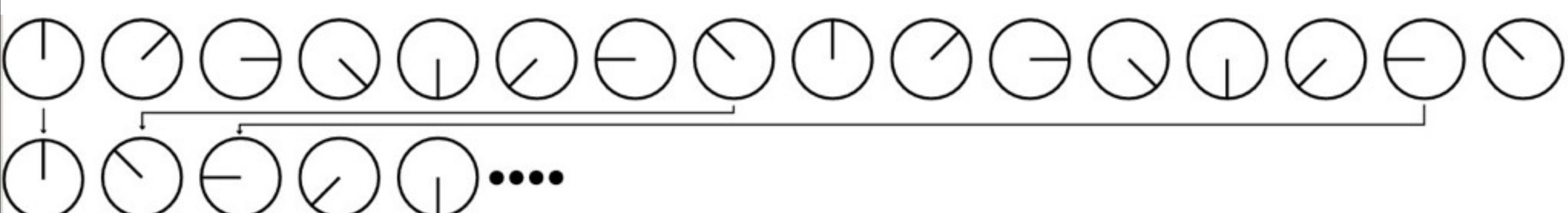
CG is a sampling and filtering process

- Pixels

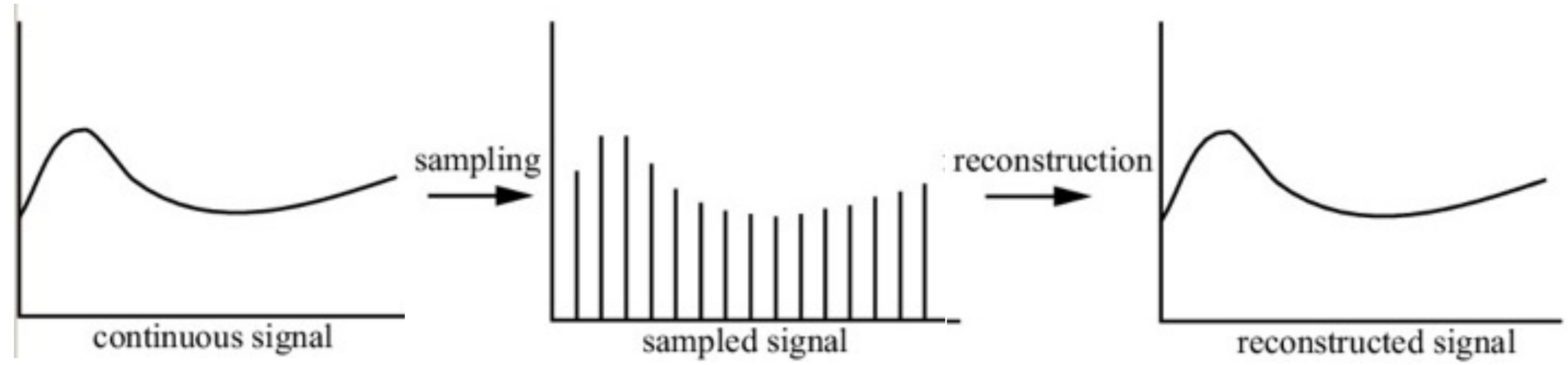
- Texture



- Time



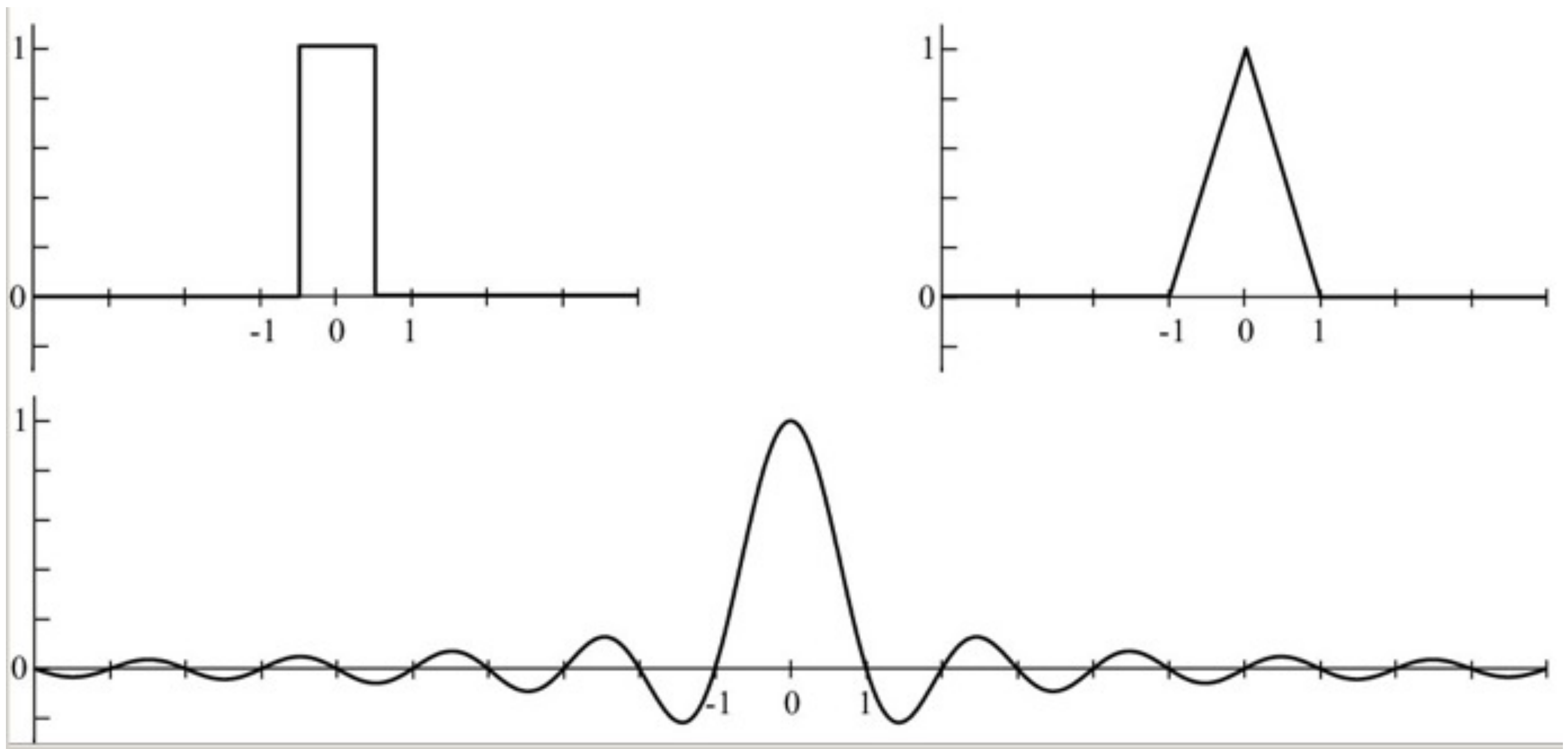
Sampling and reconstruction



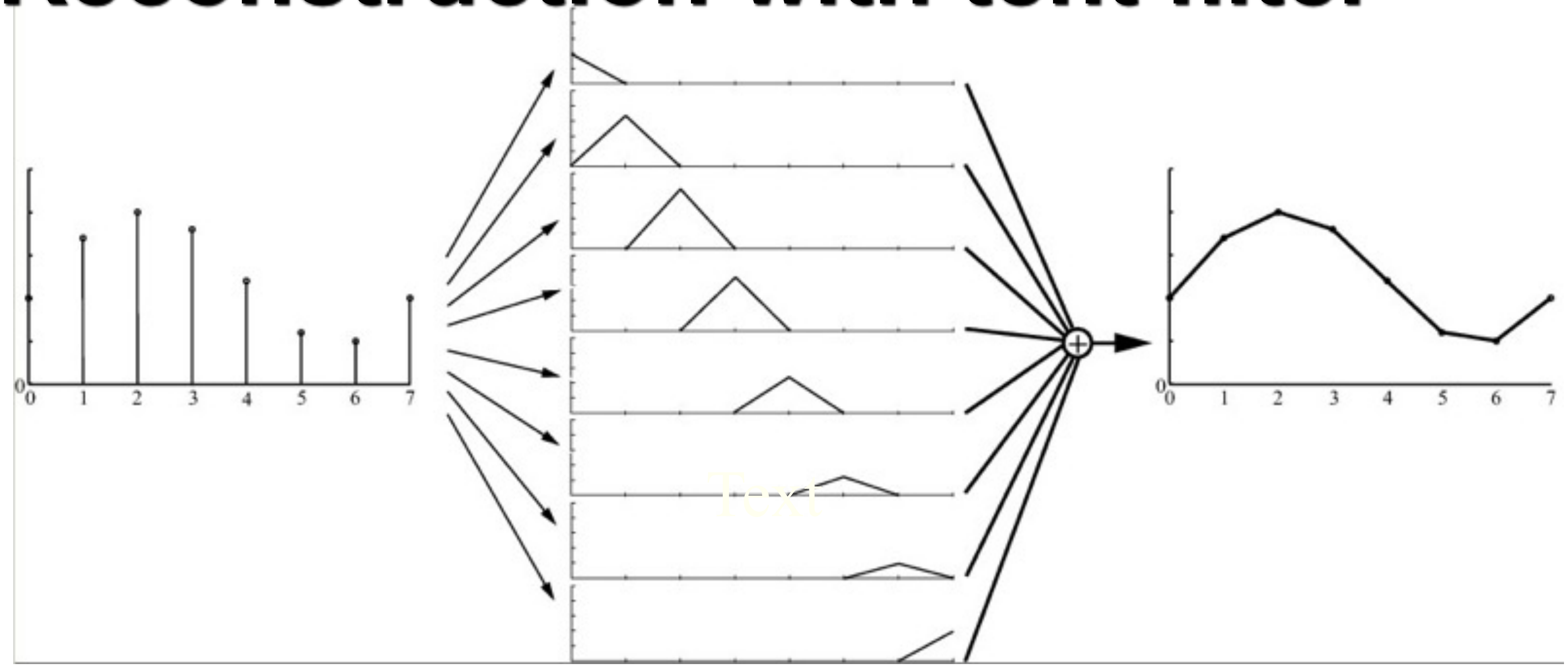
- Sampling: from continuous signal to discrete
- Reconstruction recovers the original signal
- Care must be taken to avoid aliasing

Sampling is simple: now turn to reconstruction

- Assume we have a bandlimited signal (e.g., a texture)
- Use filters for reconstruction



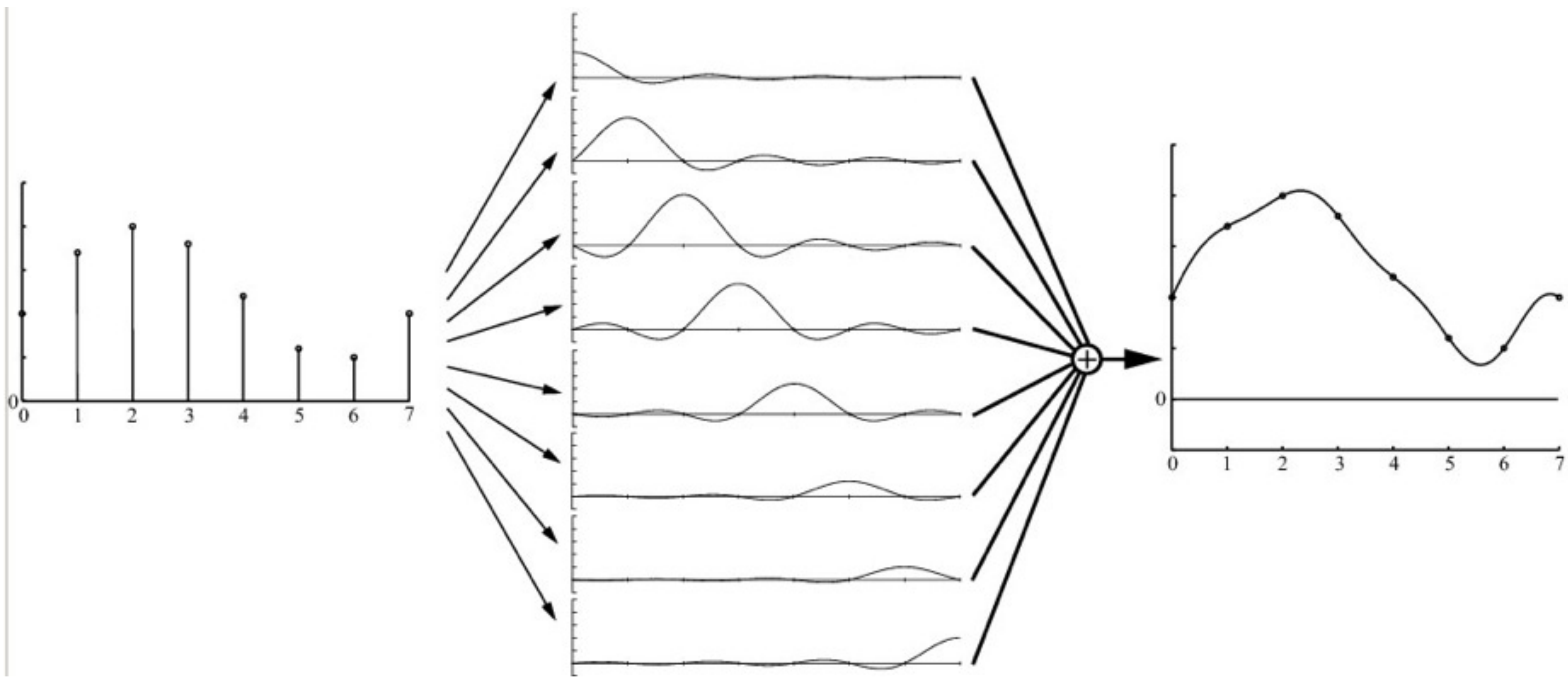
Reconstruction with tent filter



Nearest neighbor

Linear

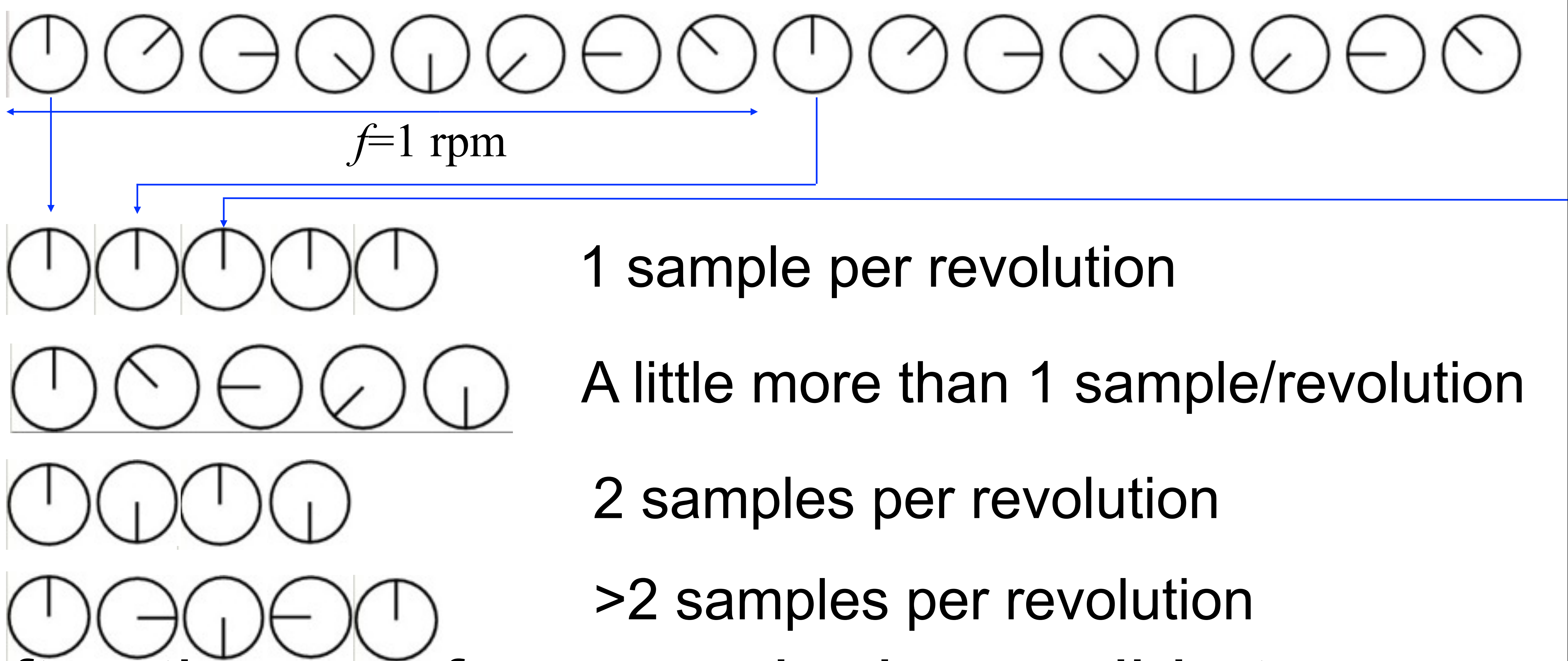
Reconstruction with sinc-filter



- In theory, the ideal filter
- Not practical (infinite extension, negative)
- Practical: Gaussian filter, symmetric piecewise cubic filter (Mitchell & Netravali)

Sampling theorem

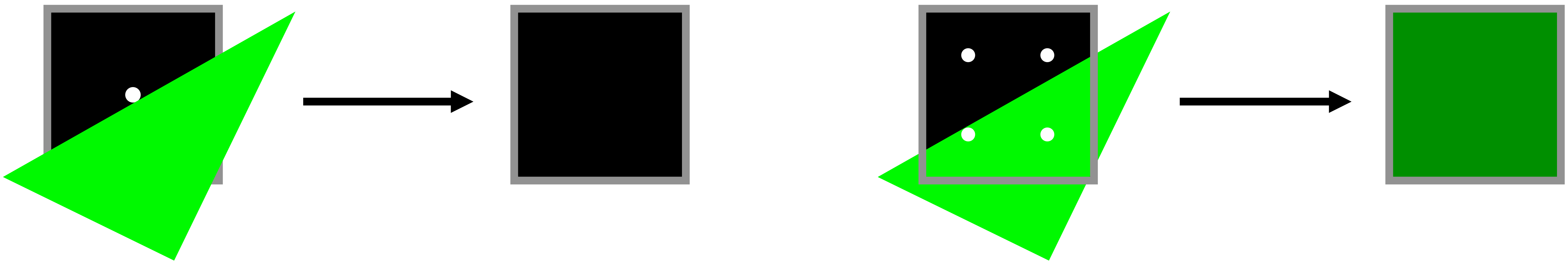
- Nyquist theorem: *the sampling frequency should be at least twice the max frequency in the signal*



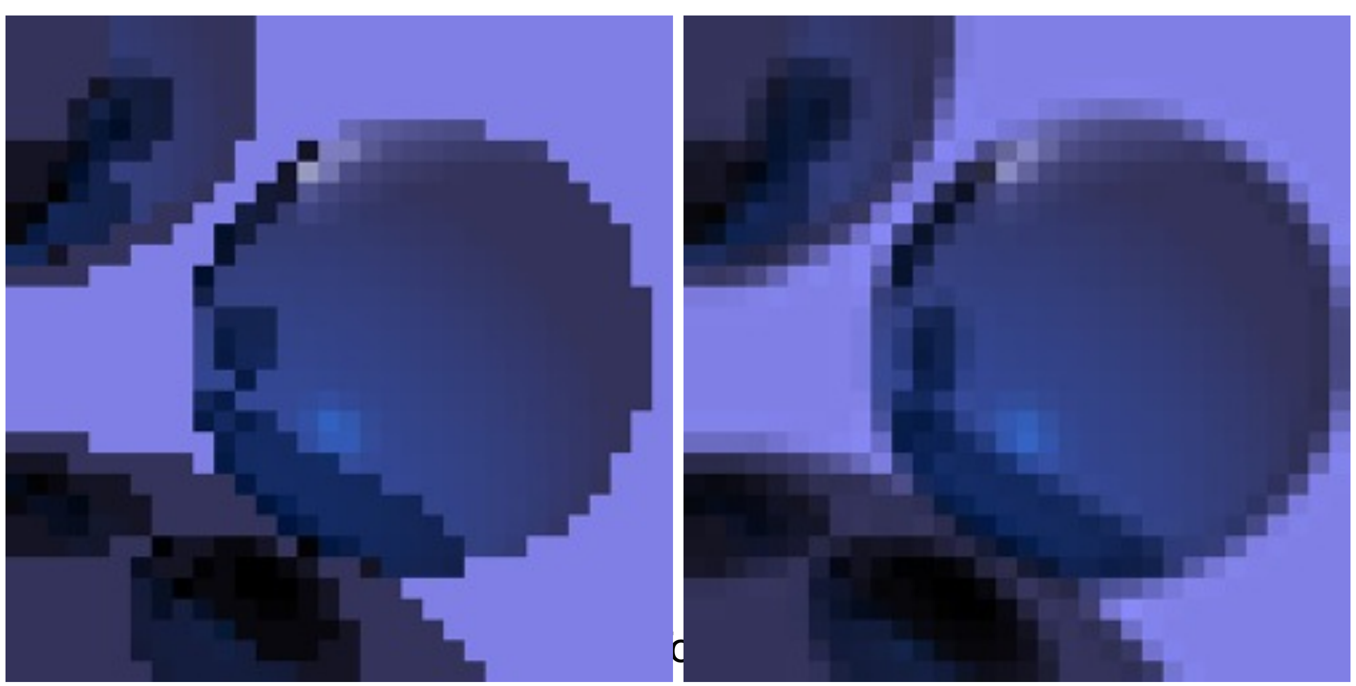
- Often the max freq. may be impossible to know, or even be infinite. What to do??

Screen-based antialiasing

- Hard case: edge has infinite frequency
- Supersampling: use more than one sample per pixel



1 sample per pixel



4x4 samples per pixel

How do we get rid of aliasing?

- What we really want: integrate the ”color” (radiance) over the entire pixel
  - In fact, over a region slightly larger than a pixel... Depends on filter!
- Randomize sample positions → replaces aliasing with noise!
- Better for HVS (human visual system)
- Spend next minutes of lecture on:
  - Monte Carlo integration
  - Jittering

Monte Carlo integration

- Rendering equation=integral. Many integrals in CG...
- Hard to evaluate
- MC can estimate integrals: $I = \int_a^b f(x)dx$
- Assume we can compute the mean of $f(x)$ over the interval $[a,b]$
  - Then the integral is mean*(b-a)
- Thus, focus on estimating mean of $f(x)$
- Idea: sample f at n uniformly distributed random locations, x_i:

$$I_{MC} = (b-a)\frac{1}{n}\sum_{i=1}^{n} f(x_i) \qquad \text{Monte Carlo estimate}$$

- When $n \rightarrow \mathit{infinity}$, $I_{MC} \rightarrow I$
- Standard deviation convergence is slow: $\sigma \propto \frac{1}{\sqrt{n}}$
- Thus, to halve error, must use 4x number of samples!!

More MC integration

X is stochastic random variable, drawn from PDF $p(x)$

$$E[X] = \int x p(x) dx$$

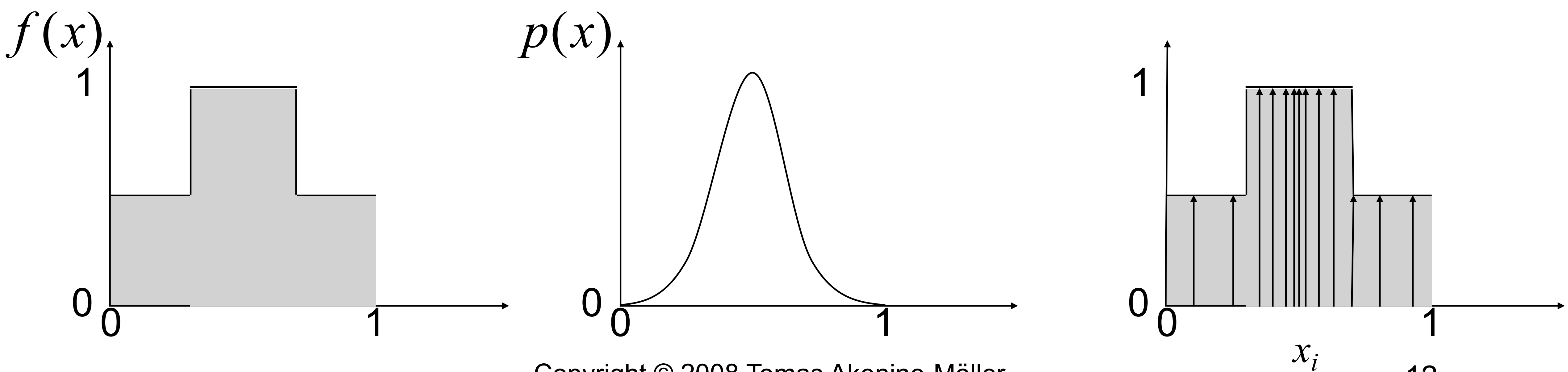
$$E[f(X)] = \int f(x) p(x) dx$$

How to evaluate the integral of $f(x)$ when x is drawn from PDF $p(x)$?

$$\int f(x)dx = \int \frac{f(x)}{p(x)} p(x) dx = \int g(x) p(x) dx = E[g(X)] = E[f(X)/p(X)]$$

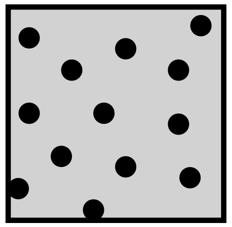
$$E[f(X)/p(X)] \approx \sum f(x_i)/p(x_i)$$ x_i drawn from PDF

Simple example

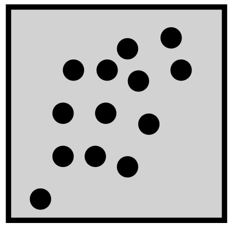


Back to sampling...

- So, randomize sample position within pixel:

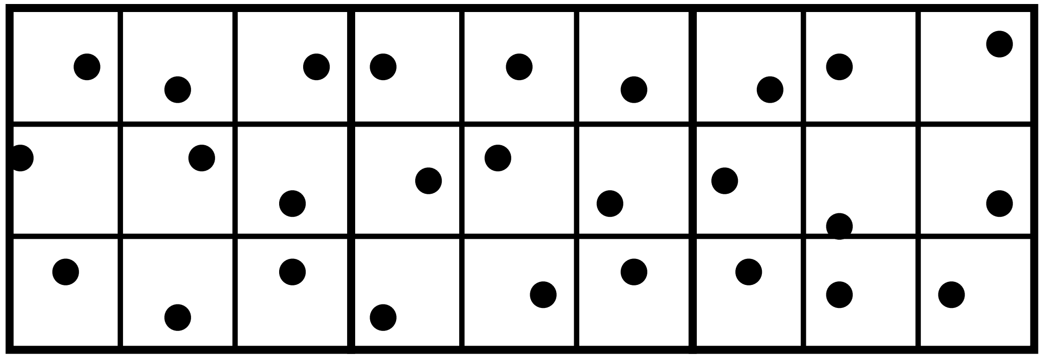


Works well
many times



But not always!
How to avoid clumping?

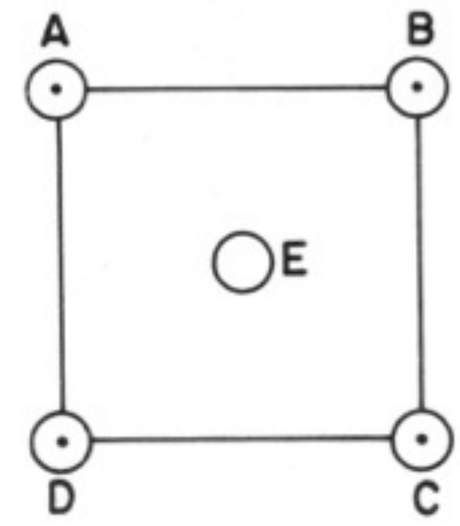
- Instead use jittering (stratified sampling):



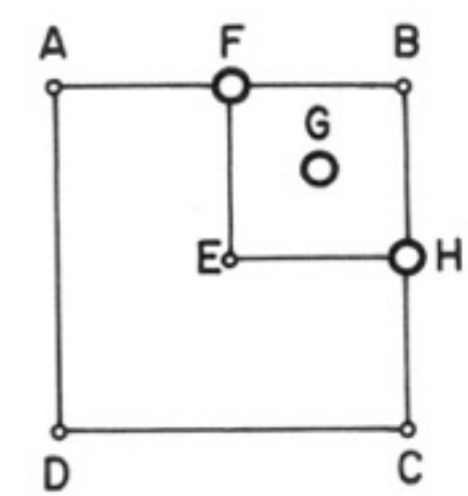
  – Divide pixel into nxn subpixels
  – Randomly position a sample in each subpixel
  – Variance always smaller than pure MC

Adaptive supersampling (1)

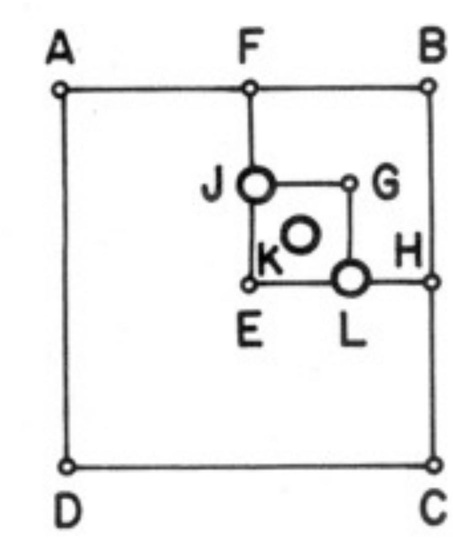
- Quincunx sampling pattern to start with
  - –2 samples per pixel, 1 in center, 1 in upper-left
  - –Note: adaptive sampling is not feasible in graphics hardware, but simple in a ray tracer



- Colors of AE, DE are quite similar, so don't waste more time on those.
- The colors of B & E are different, so add more samples there with the same sampling pattern

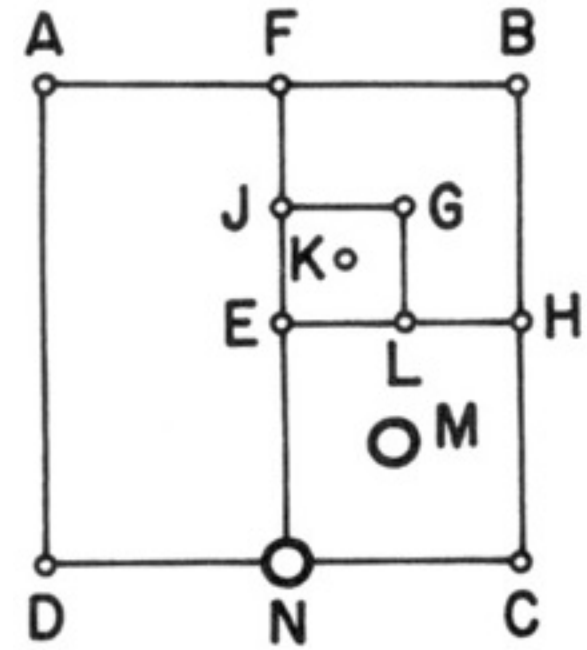


- Same thing again, check FG, BG, HG, EG: only EG needs more sampling
- So, add rays for J, K, and L

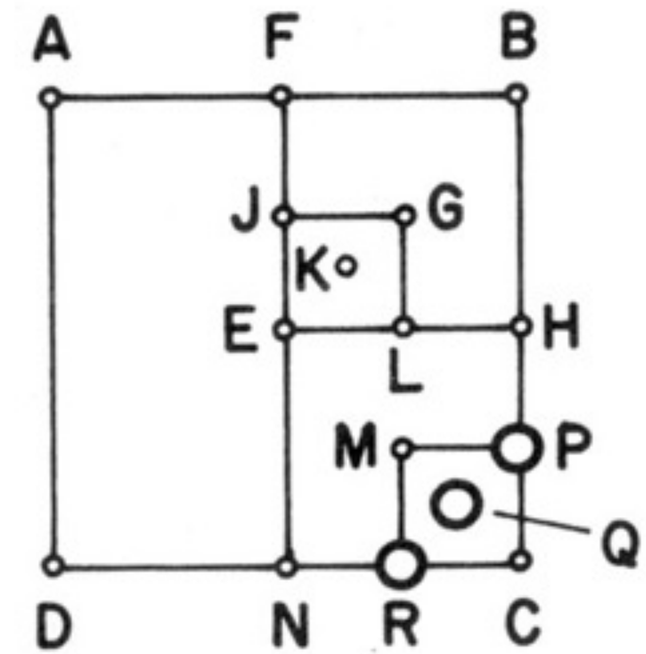


Adaptive supersampling (2)

- C & E were different too
- Add N & M
- Compare EM, HM, CM, NM



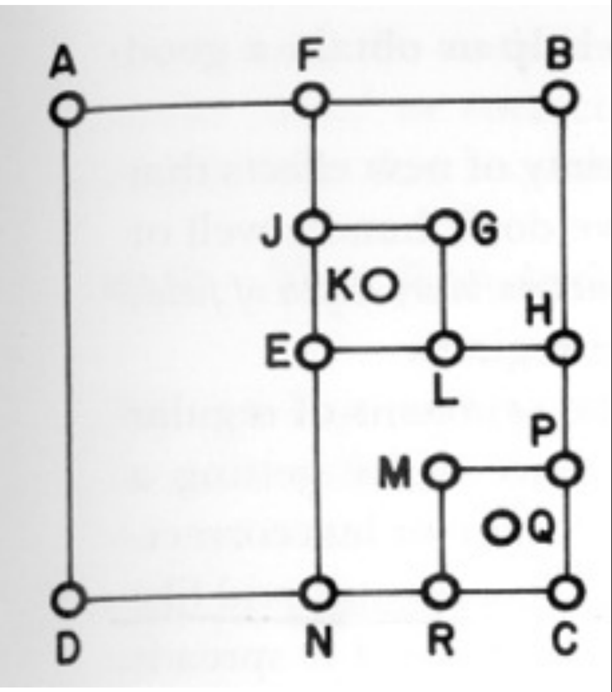
- C & M are too different
- So add rays at P, Q, and R



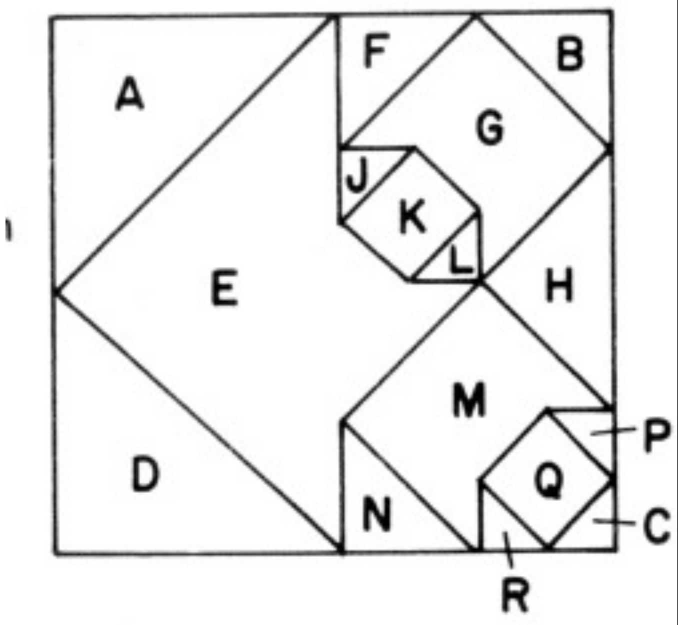
- At this point, we consider the entire pixel to be sufficiently sampled
- Time to weigh (filter) the colors of all rays

Adaptive supersampling (3)

- Final sample pattern for pixel:



- How to filter the colors of the rays?
- Think of the pattern differently:



- And use the area of each ray sample as its weight:

$$\frac{1}{4}\left(\frac{A+E}{2}+\frac{D+E}{2}+\frac{1}{4}\left[\frac{F+G}{2}+\frac{B+G}{2}+\frac{H+G}{2}+\frac{1}{4}\left\{\frac{J+K}{2}+\frac{G+K}{2}+\frac{L+K}{2}+\frac{E+K}{2}\right\}\right]\right.$$

$$\left.+\frac{1}{4}\left[\frac{E+M}{2}+\frac{H+M}{2}+\frac{N+M}{2}+\frac{1}{4}\left\{\frac{M+Q}{2}+\frac{P+Q}{2}+\frac{C+Q}{2}+\frac{R+Q}{2}\right\}\right]\right).$$

Caveats with adaptive supersampling (4)

- May miss really small objects anyway
- It’s still supersampling, but smart supersampling
  - Cannot fool Nyquist!
  - Only reduce aliasing – does not eliminate it

Sampling conclusion

- Be careful when sampling
- Do it well → good quality
- Do it in a clever way → reasonably fast

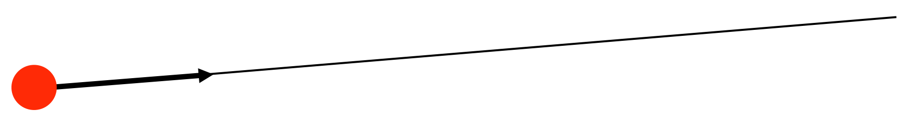
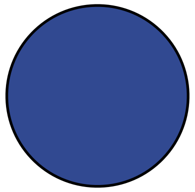
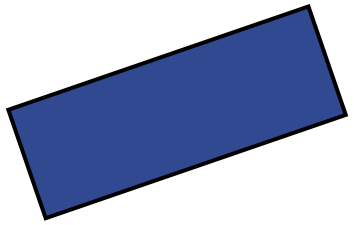
Object intersection

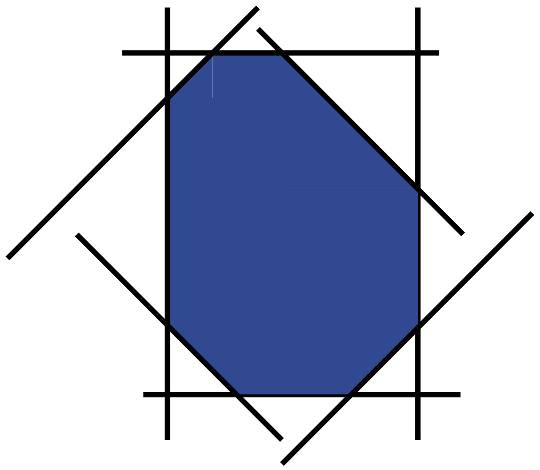
What for?

- A tool needed for graphics people all the time…
- Very important components:
  - Need to make them fast!

- Finding if (and where) a ray hits an object
  - Picking
  - Ray tracing and global illumination
- For speed-up techniques
- Collision detection

Some basic geometrical primitives

- Ray: 
- Sphere: 
- Box
  - Axis-aligned (AABB) 
  - Oriented (OBB) 
- *k*-DOP
  - convex polyhedron

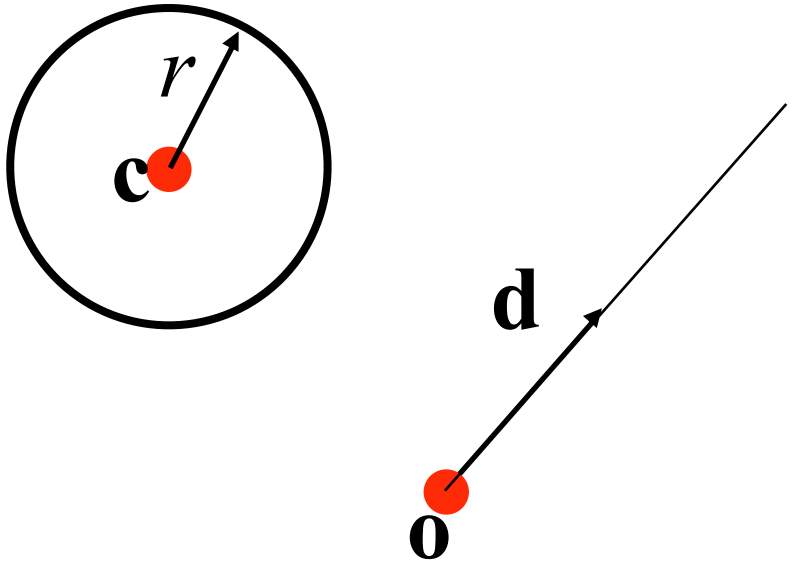


Four different techniques

- Analytical
- Geometrical
- Separating axis theorem (SAT)
- Dynamic tests

- Given these, one can derive many tests quite easily
  - However, often tricks are needed to make them fast

Analytical: Ray/sphere test



- Sphere center: $\mathbf{c}$, and radius r
- Ray: $\mathbf{r}(t)=\mathbf{o}+t\mathbf{d}$
- Sphere formula: $\|\mathbf{p}-\mathbf{c}\|=r$
- Replace $\mathbf{p}$ by $\mathbf{r}(t)$, and square it:

$$(\mathbf{r}(t)-\mathbf{c})\cdot(\mathbf{r}(t)-\mathbf{c})-r^2=0$$

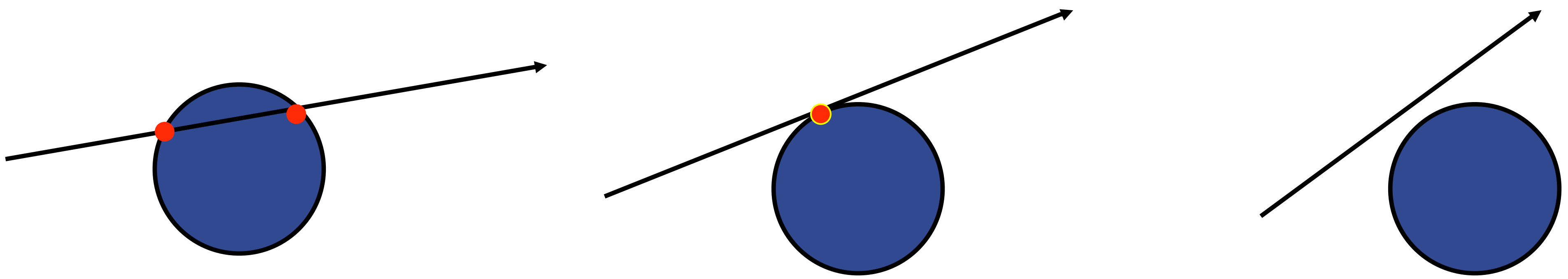
$$(\mathbf{o}+t\mathbf{d}-\mathbf{c})\cdot(\mathbf{o}+t\mathbf{d}-\mathbf{c})-r^2=0$$

$$(\mathbf{d}\cdot\mathbf{d})t^2+2((\mathbf{o}-\mathbf{c})\cdot\mathbf{d})t+(\mathbf{o}-\mathbf{c})\cdot(\mathbf{o}-\mathbf{c})-r^2=0$$

$$t^2+2((\mathbf{o}-\mathbf{c})\cdot\mathbf{d})t+(\mathbf{o}-\mathbf{c})\cdot(\mathbf{o}-\mathbf{c})-r^2=0 \qquad \|\mathbf{d}\|=1$$

Analytical, continued

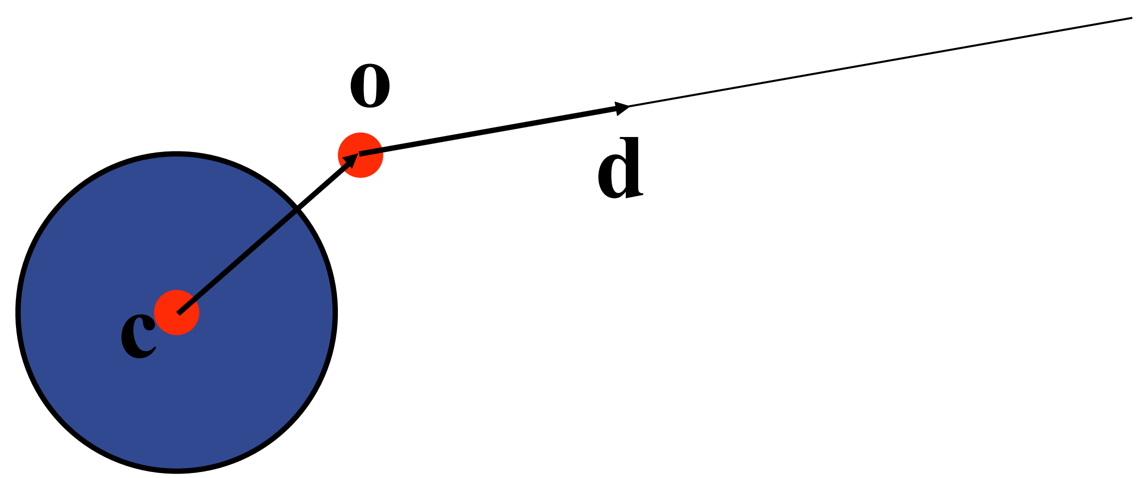
$$t^2 + 2((\mathbf{o}-\mathbf{c})\cdot\mathbf{d})t + (\mathbf{o}-\mathbf{c})\cdot(\mathbf{o}-\mathbf{c}) - r^2 = 0$$



- Be a little smart…

$$(\mathbf{o}-\mathbf{c})\cdot\mathbf{d} > 0?$$

$$(\mathbf{o}-\mathbf{c})\cdot(\mathbf{o}-\mathbf{c}) - r^2 < 0?$$



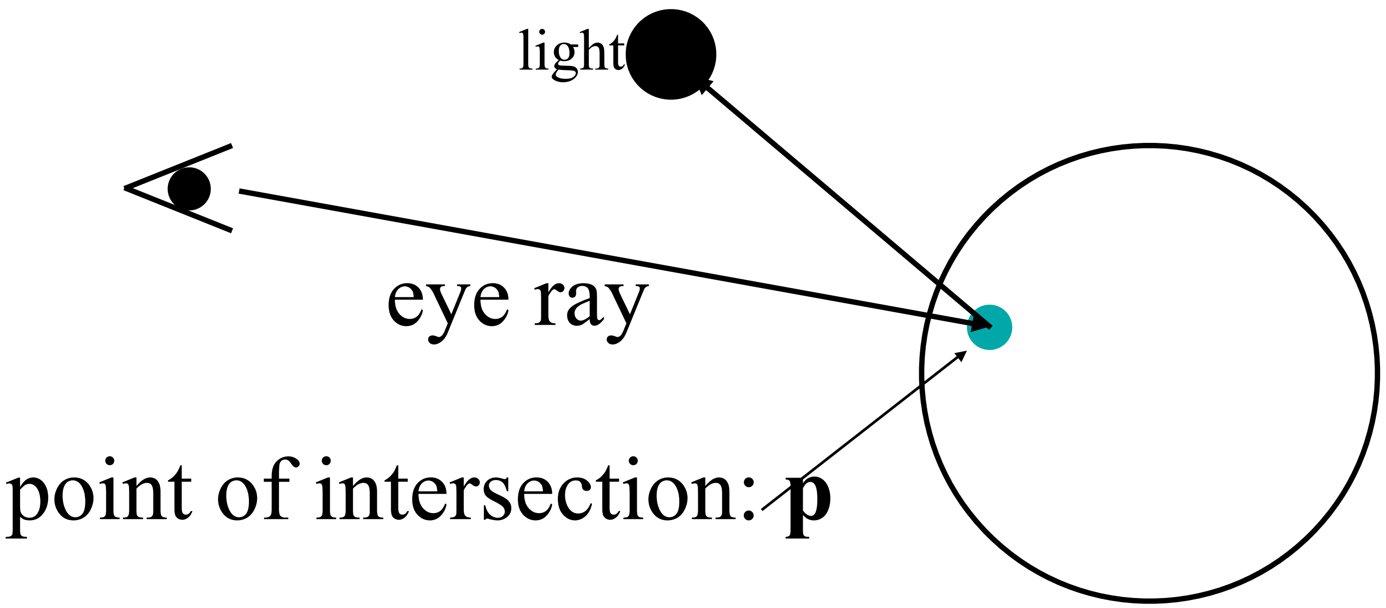
- Such tests are called ”rejection tests”
- Other shapes: $p_x^2 + p_y^2 = r^2$

$$(p_x / a)^2 + (p_y / b)^2 + (p_z / c)^2 = 1$$

$$(p_x / a)^2 + (p_y / b)^2 - p_z = 0$$

Precision problems in intersection testing

- Exaggerated:

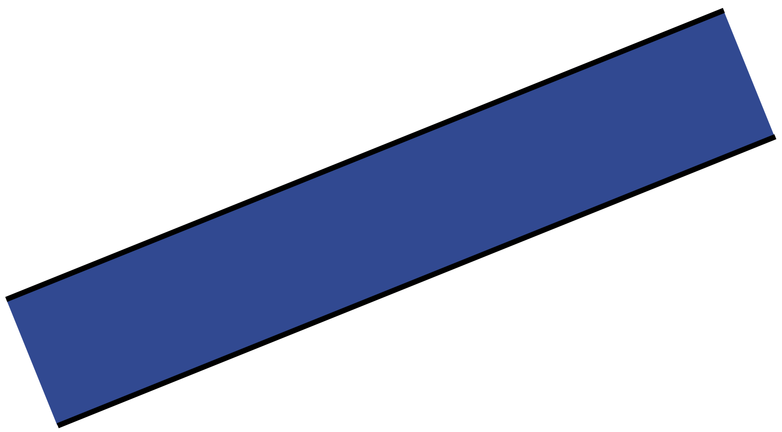


The point, **p**, will be incorrectly self-shadowed, due to imprecision

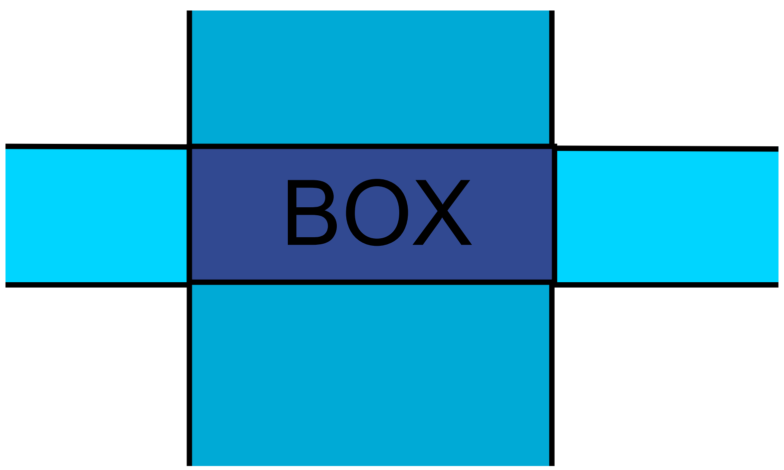
Solution: after **p** has been computed, update as: $\mathbf{p}' = \mathbf{p} + \varepsilon \mathbf{n}$
($\mathbf{n}$ is normal at $\mathbf{p}$, ε is small number >0)

Geometrical: Ray/Box Intersection

- Boxes and spheres often used as bounding volumes
- A slab is the volume between two parallel planes:

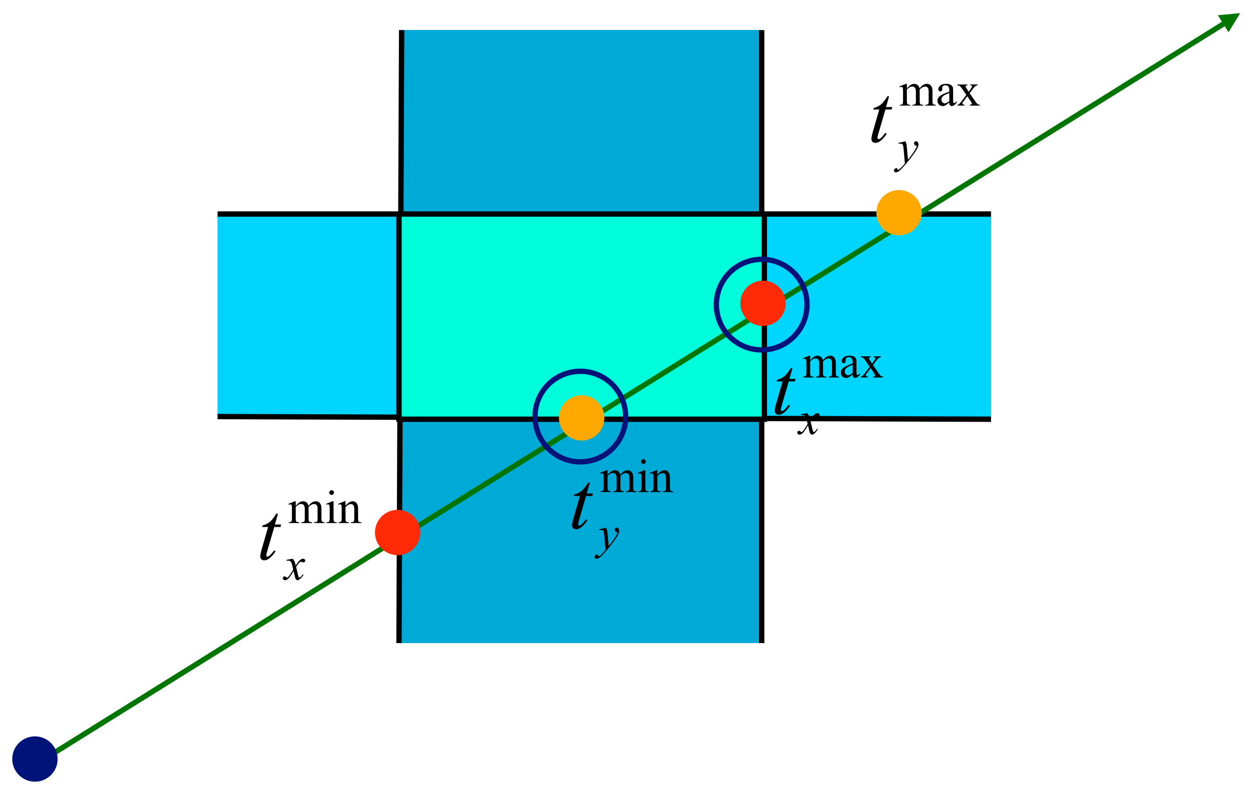


- A box is the logical intersection of three slabs (2 in 2D):



Geometrical: Ray/Box Intersection (2)

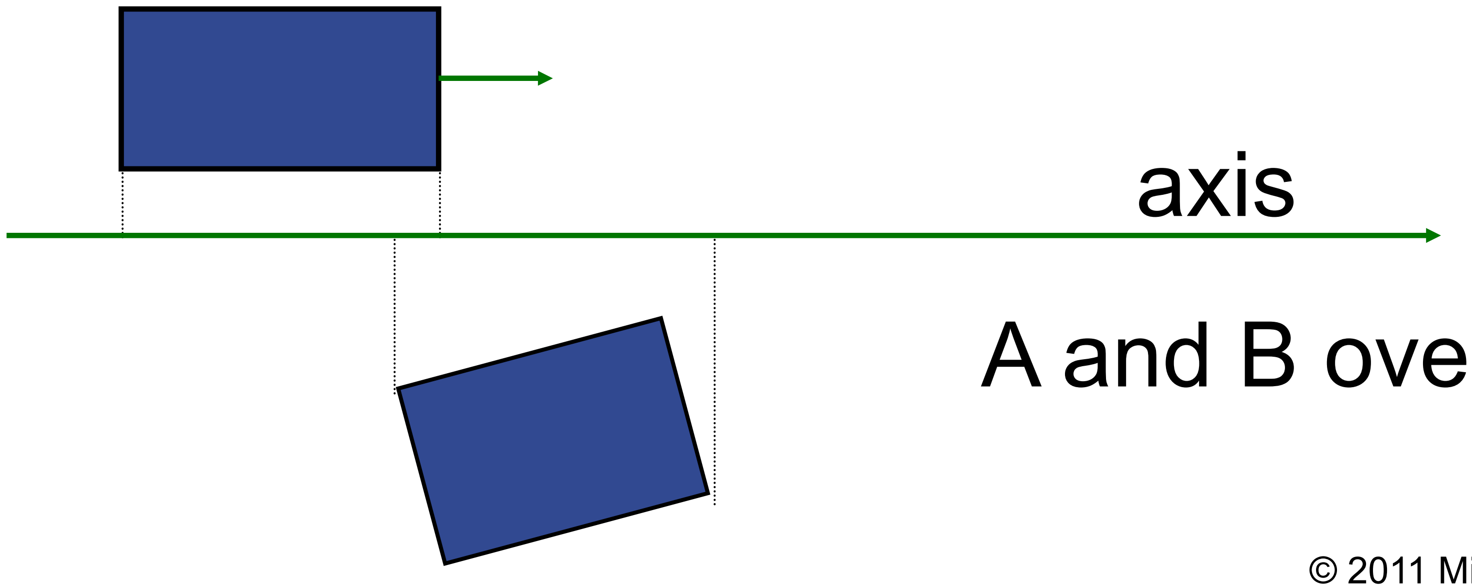
- Intersect the 2 planes of each slab with the ray



- Keep max of t^{min} and min of t^{max}
- If $t^{min} < t^{max}$ then we get an intersection
- Special case when ray parallel to slab

Separating Axis Theorem (SAT)

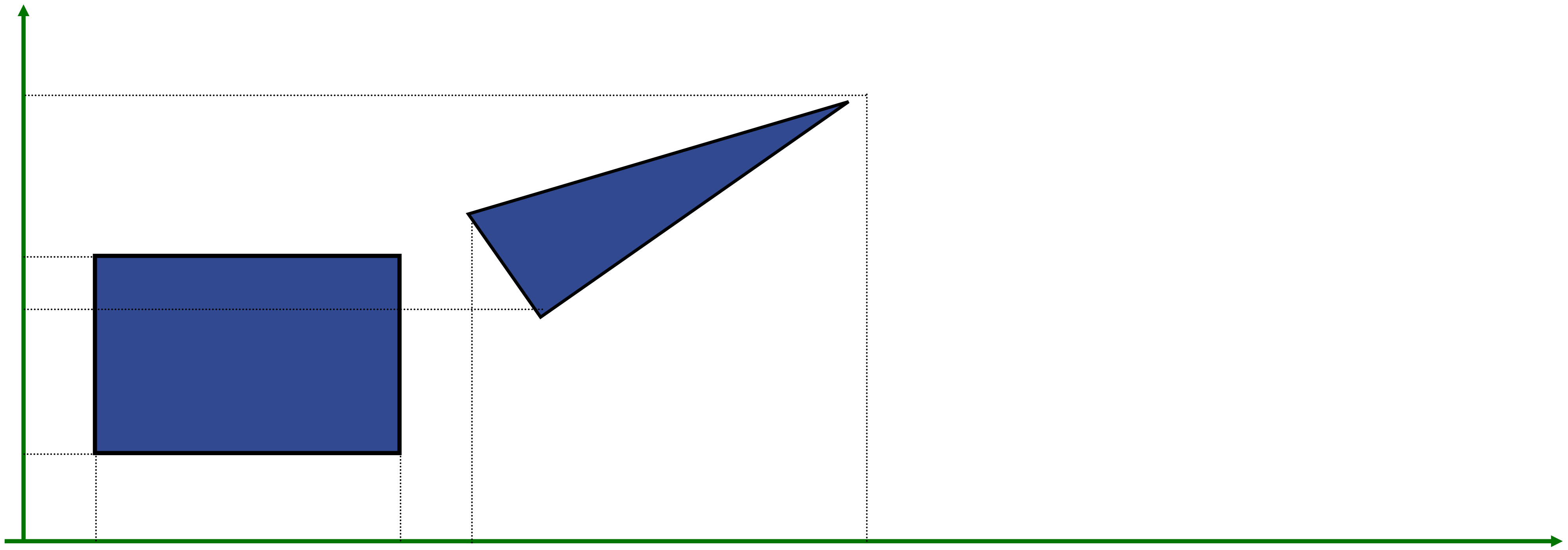
- Two convex polyhedrons, A and B, are disjoint if any of the following axes separate the objects:
  - An axis orthogonal to a face of A
  - An axis orthogonal to a face of B
  - An axis formed from the cross product of one edge from each of A and B



A and B overlaps on this axis

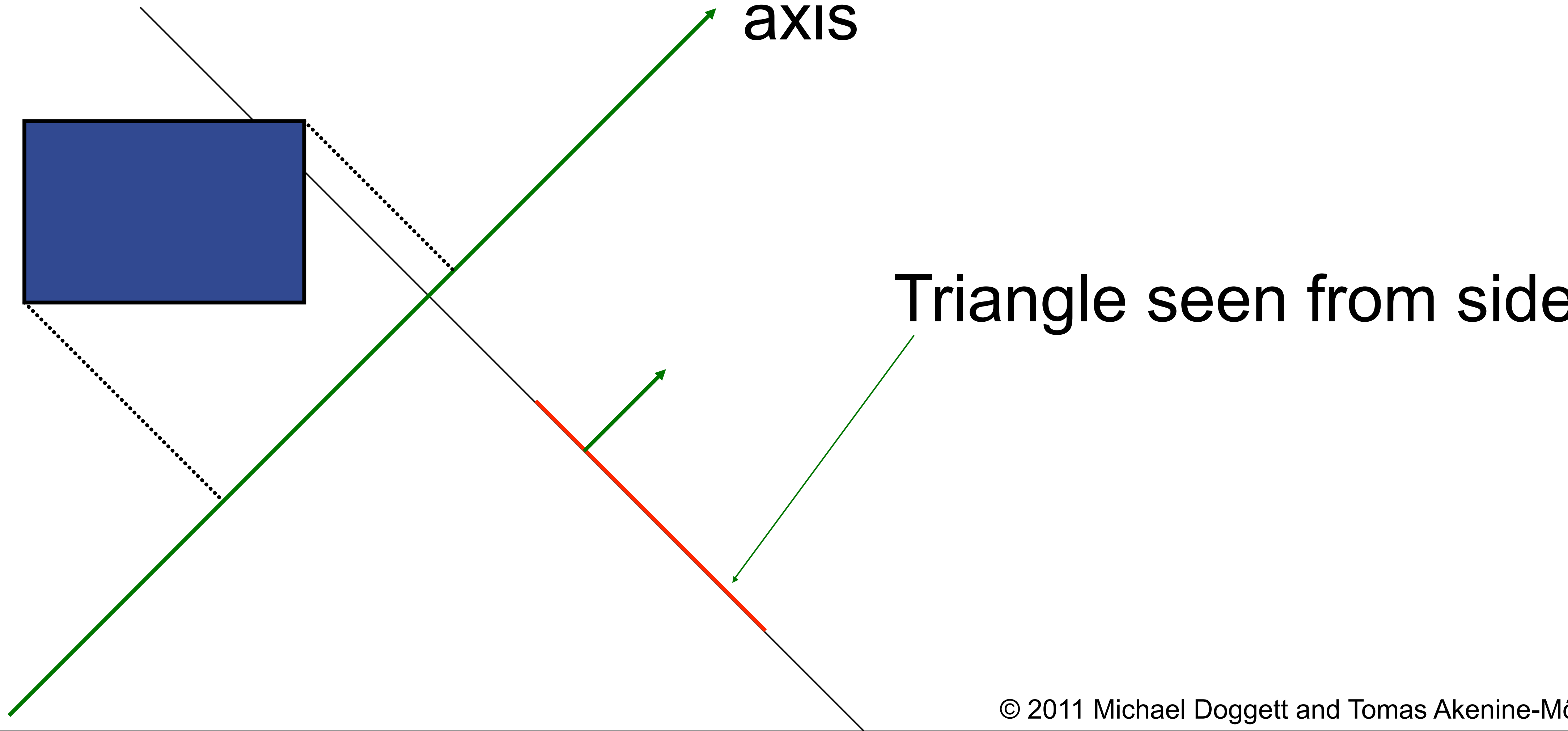
SAT example: Triangle/Box

- Box is axis-aligned
- **1)** test the axes that are orthogonal to the faces of the box
- That is, x,y, and z



Triangle/Box with SAT (2)

- Assume that they overlapped on x,y,z
- Must continue testing
- **2)** Axis orthogonal to face of triangle



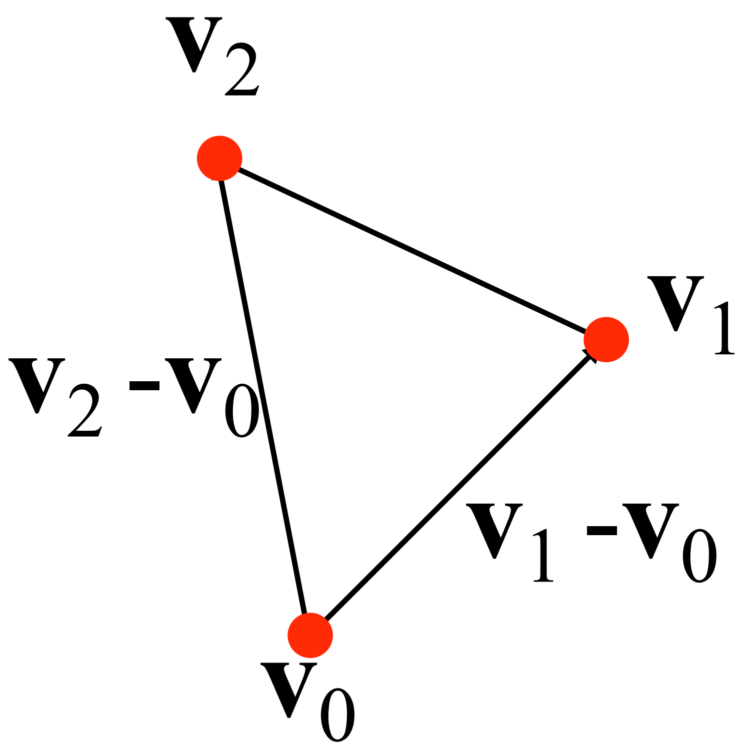
Triangle/Box with SAT (3)

- If still no separating axis has been found…
- **3)** Test axis: $\mathbf{t}=\mathbf{e}_{\text{box}} \times \mathbf{e}_{\text{triangle}}$
- Example:
  - x-axis from box: $\mathbf{e}_{\text{box}}=(1,0,0)$
  - $\mathbf{e}_{\text{triangle}}=\mathbf{v}_1-\mathbf{v}_0$
- Test all such combinations
- If there is at least one separating axis, then the objects do not collide
- Else they do overlap

Rules of Thumb for Intersection Testing

- Acceptance and rejection test
  - Try them early on to make a fast exit
- Postpone expensive calculations if possible
- Use dimension reduction
  - E.g. 3 one-dimensional tests instead of one complex 3D test, or 2D instead of 3D
- Share computations between objects if possible
- Timing!!!

Another analytical example: Ray/Triangle in detail



- Ray: $\mathbf{r}(t)=\mathbf{o}+t\mathbf{d}$
- Triangle vertices: $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2$
- A point in the triangle:
- $\mathbf{t}(u,v)=\mathbf{v}_0+u(\mathbf{v}_1 - \mathbf{v}_0) + v(\mathbf{v}_2 - \mathbf{v}_0)$
  $=(1-u-v)\mathbf{v}_0+u\mathbf{v}_1+v\mathbf{v}_2 \quad [u,v>=0,\ u+v<=1]$
- Set $\mathbf{t}(u,v)=\mathbf{r}(t)$, and solve!

$$\begin{pmatrix} | & | & | \\ -\mathbf{d} & \mathbf{v}_1-\mathbf{v}_0 & \mathbf{v}_2-\mathbf{v}_0 \\ | & | & | \end{pmatrix}\begin{pmatrix} t \\ u \\ v \end{pmatrix}=\begin{pmatrix} | \\ \mathbf{o}-\mathbf{v}_0 \\ | \end{pmatrix}$$

Ray/Triangle (2)

$$\begin{pmatrix} | & | & | \\ -\mathbf{d} & \mathbf{v}_1 - \mathbf{v}_0 & \mathbf{v}_2 - \mathbf{v}_0 \\ | & | & | \end{pmatrix} \begin{pmatrix} t \\ u \\ v \end{pmatrix} = \begin{pmatrix} | \\ \mathbf{o} - \mathbf{v}_0 \\ | \end{pmatrix}$$

$$\mathbf{e}_1 = \mathbf{v}_1 - \mathbf{v}_0 \qquad \mathbf{e}_2 = \mathbf{v}_2 - \mathbf{v}_0 \qquad \mathbf{s} = \mathbf{o} - \mathbf{v}_0$$

- Solve with Cramer's rule:

$$\begin{pmatrix} t \\ u \\ v \end{pmatrix} = \frac{1}{\det(-\mathbf{d}, \mathbf{e}_1, \mathbf{e}_2)} \begin{pmatrix} \det(\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2) \\ \det(-\mathbf{d}, \mathbf{s}, \mathbf{e}_2) \\ \det(-\mathbf{d}, \mathbf{e}_1, \mathbf{s}) \end{pmatrix}$$

Use this fact : $\det(\mathbf{a}, \mathbf{b}, \mathbf{c}) = (\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c} = -(\mathbf{a} \times \mathbf{c}) \cdot \mathbf{b}$

$$\begin{pmatrix} t \\ u \\ v \end{pmatrix} = \frac{1}{(\mathbf{d} \times \mathbf{e}_2) \cdot \mathbf{e}_1} \begin{pmatrix} (\mathbf{s} \times \mathbf{e}_1) \cdot \mathbf{e}_2 \\ (\mathbf{d} \times \mathbf{e}_2) \cdot \mathbf{s} \\ (\mathbf{s} \times \mathbf{e}_1) \cdot \mathbf{d} \end{pmatrix}$$

- Share factors to speed up computations

Ray/Triangle (3) Implementation

$$\begin{pmatrix} t \\ u \\ v \end{pmatrix} = \frac{1}{(\mathbf{d}\times\mathbf{e}_2)\cdot\mathbf{e}_1}\begin{pmatrix} (\mathbf{s}\times\mathbf{e}_1)\cdot\mathbf{e}_2 \\ (\mathbf{d}\times\mathbf{e}_2)\cdot\mathbf{s} \\ (\mathbf{s}\times\mathbf{e}_1)\cdot\mathbf{d} \end{pmatrix}$$

- Be smart!
  - Compute as little as possible then test
- Examples:

$$\mathbf{p} = \mathbf{d}\times\mathbf{e}_2$$
$$a = \mathbf{p}\cdot\mathbf{e}_1$$
$$f = 1/a$$

- Compute $u = f(\mathbf{p}\cdot\mathbf{s})$
- Then test valid bounds
- `if (u<0 or u>1) exit;`

Plane: $\pi : \mathbf{n}\cdot\mathbf{p} + d = 0$

Point/Plane

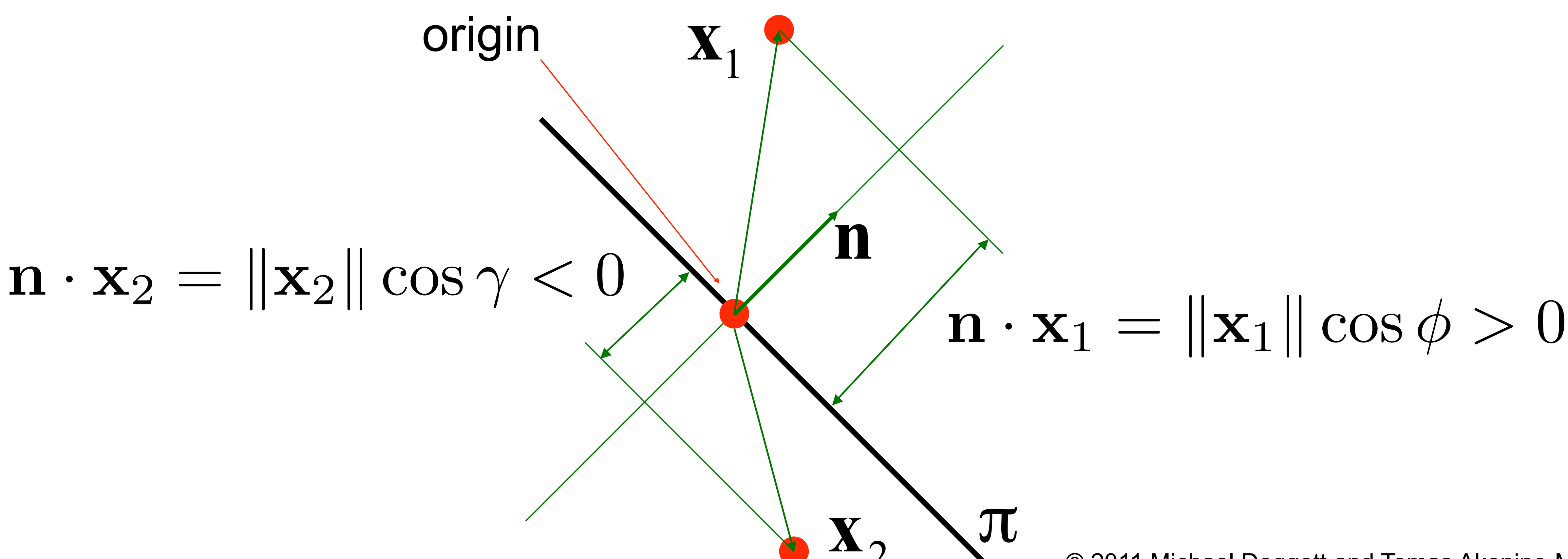
- Insert a point $\mathbf{x}$ into plane equation:

$$f(\mathbf{x}) = \mathbf{n}\cdot\mathbf{x} + d = ?$$

$f(\mathbf{x}) = \mathbf{n}\cdot\mathbf{x} + d = 0$ for **x**'s on the plane

Negative half space: $f(\mathbf{x}) = \mathbf{n}\cdot\mathbf{x} + d < 0$ for **x**'s on one side of the plane

Positive half space: $f(\mathbf{x}) = \mathbf{n}\cdot\mathbf{x} + d > 0$ for **x**'s on the other side



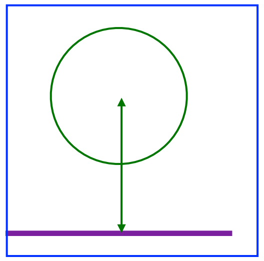
Sphere/Plane
AABB/Plane

Plane: $\pi : \mathbf{n} \cdot \mathbf{p} + d = 0$

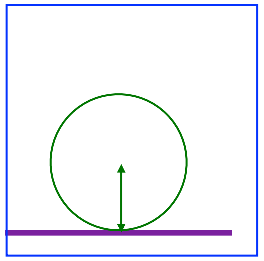
Sphere: $\mathbf{c}$ r

Box: $\mathbf{b}^{\min}$ $\mathbf{b}^{\max}$

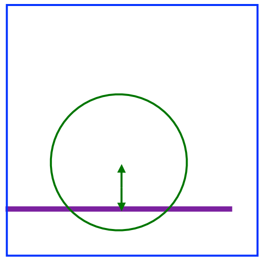
- Sphere: compute $f(\mathbf{c}) = \mathbf{n} \cdot \mathbf{c} + d$
- $f(\mathbf{c})$ is the signed distance ($\mathbf{n}$ normalized)



- $\mathrm{abs}(f(\mathbf{c})) > \mathrm{r}$ no collision



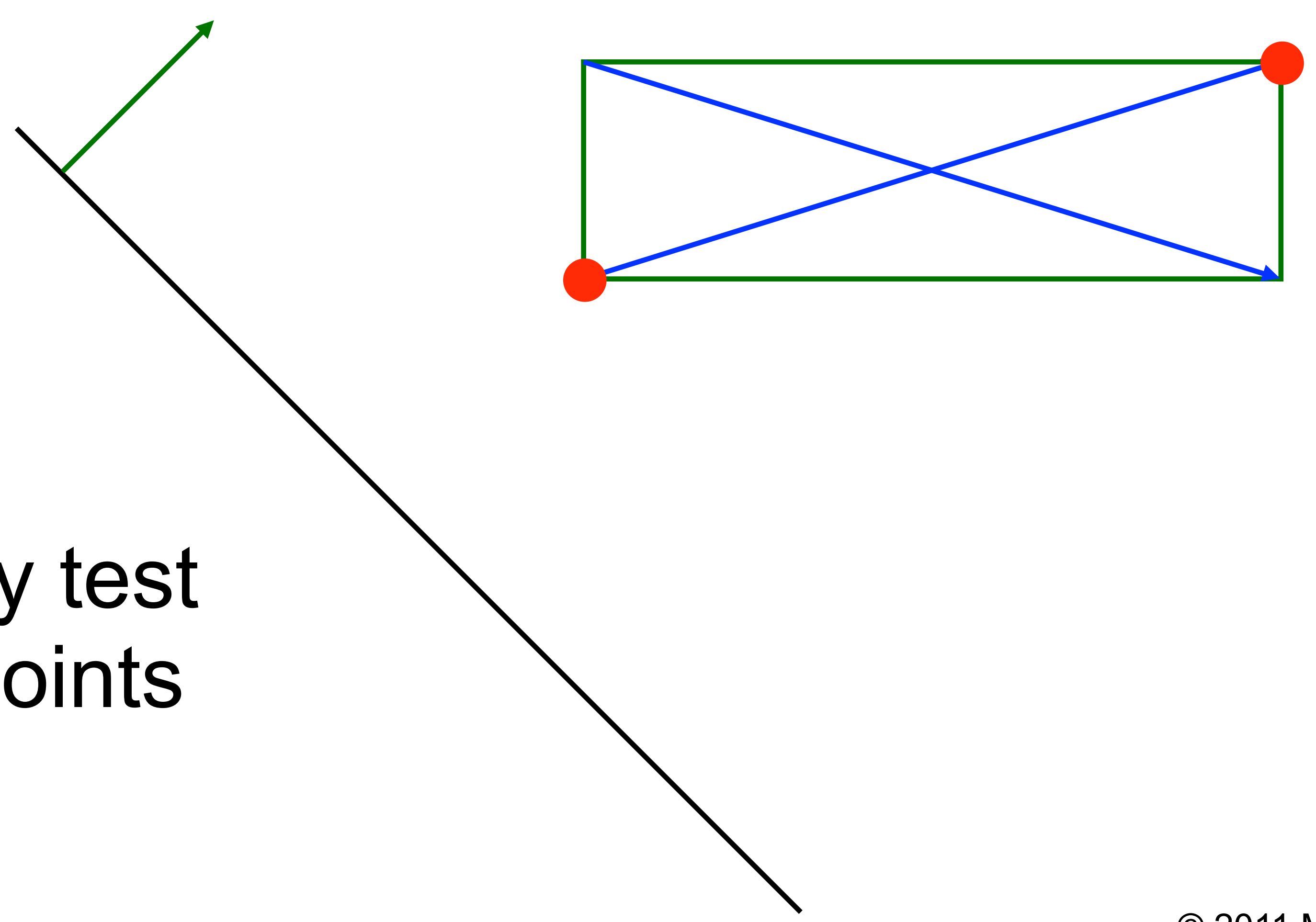
- $\mathrm{abs}(f(\mathbf{c})) = \mathrm{r}$ sphere touches the plane



- $\mathrm{abs}(f(\mathbf{c})) < \mathrm{r}$ sphere intersects plane
- Box: insert all 8 corners
- If all f's have the same sign, then all points are on the same side, and no collision

AABB/Plane

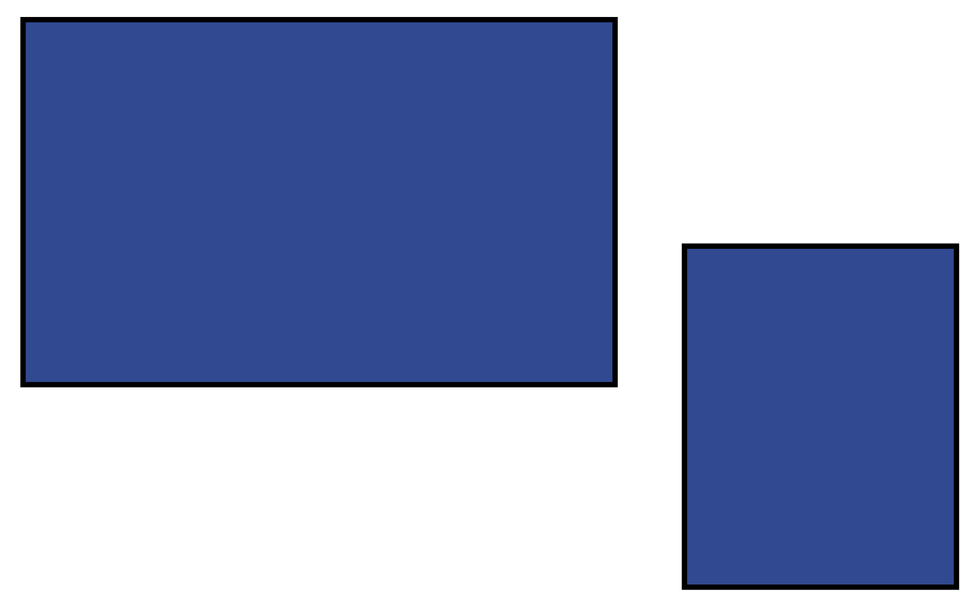
- The smart way (shown in 2D)
- Find diagonal that is most closely aligned with plane normal



Need only test the red points

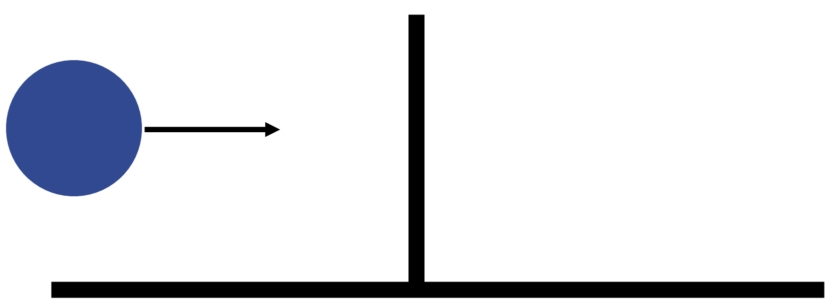
Volume/Volume tests

- Used in collision detection
- Sphere/sphere
  - Compute squared distance between sphere centers, and compare to $(r_1+r_2)^2$
- Axis-Aligned Bounding Box/AABB
  - Test in 1D for x,y, and z
  - If all 1D intersect then they intersect

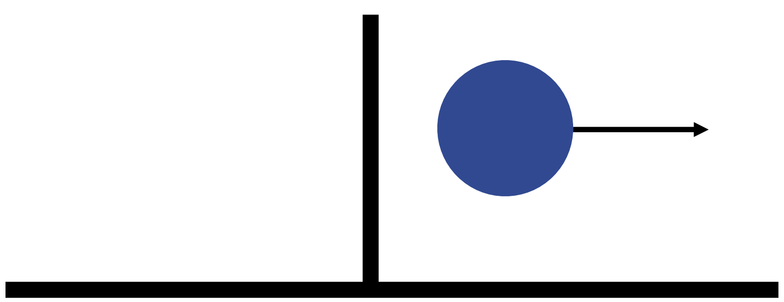


Dynamic Intersection Testing

- Testing is often done every rendered frame, i.e., at discrete time intervals
- Therefore, you can get ”quantum effects”



Frame *n*



Frame *n+1*

- Dynamic testing deals with this
- Is more expensive
- Deals with a time interval: time between two frames

Summary

- Sampling
  - Important for image quality and speed

- Object intersection
  - Range of primitives
  - Basic operation, so speed is important
  - Find more in "Real-Time Rendering" book

Next

- C++
- Monday - Seminar
  - prTracer code overview (code is available now on webpage)
  - Assignment 1 Seminar : Ray Tracing (Magnus)
- Tuesday - Lecture
  - Acceleration data structures