cs155 – z sweedyk

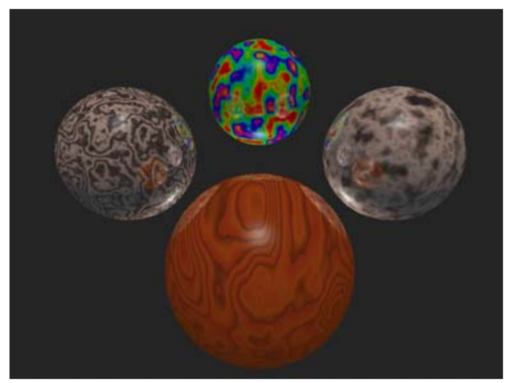
ray tracing

cheap tricks

- texture mapping
- bump mapping
- procedural texture mapping

procedural texture mapping

- procedure returns a texture color for any point in 3d space (note this is not an image stack)
- sample to find texture for surface



adrian mettler, spring 2003

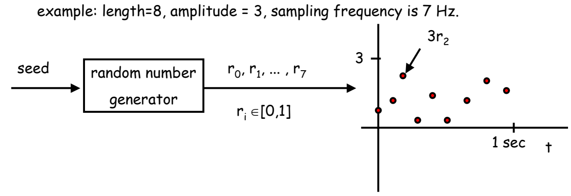
procedural textures

- advantages
  - don't need to find a mapping from a (complex) 3d surface to a 2d texture image
  - concise representation of texture
- disadvantages
  - ad hoc techniques cannot duplicate photographs

perlin noise – 1D example

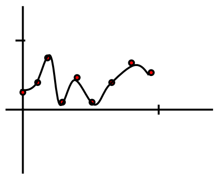
step 1: generate discrete noise function with specified length, amplitude, sampling frequency

example: length=8, amplitude = 3, sampling frequency is 7 Hz.



perlin noise – 1D example

step 2: interpolate with smoothing

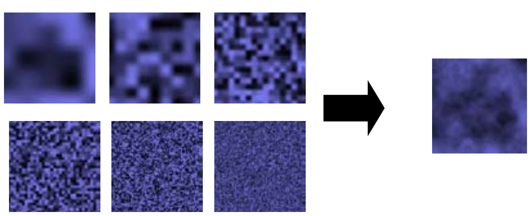


perlin noise – 1D example

step 3: repeat with various amplitudes/frequencies

step 4: add together

perlin noise – 2D example



for more info see Perlin Noise link on proj2 web site

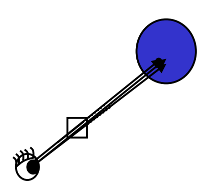
cheap tricks

- texture mapping
- bump mapping
- procedural texture mapping
- jittering

jittering: antialiasing technique

Run rt on Solaris for example

jittering: anti-aliasing technique



for pixel i,j:

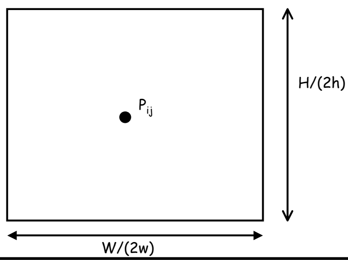
- cast several random rays through pixel i,j region into scene
- for each:
  - find intersection point (if any) that is closest to eye
  - compute color at intersection
- compute average of all samples as color for pixel i,j

random ray

shoot ray i,j through P_{ij} + (dx,dy) where

dx is chosen uniformly at random over [-W/(2w), W/(2w)]

dy is chosen uniformly at random over [-H/(2h),H/(2h)]



cheap tricks

- texture mapping
- bump mapping
- procedural texture mapping
- jittering
- soft shadows

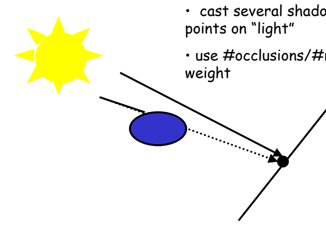
soft shadows

run rt on solaris for examples

soft shadows

In reality, we don't have point lights (or point spot lights)!!

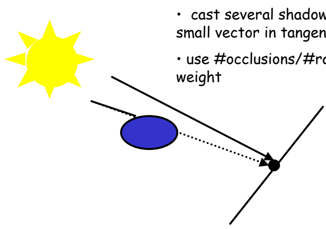
- cast several shadow rays at random points on "light"
- use #occlusions/#rays as shadow weight



soft shadows

For diretional lights

- cast several shadow rays perturbed by small vector in tangent plane
- use #occlusions/#rays as shadow weight



ray tracing

- simple ray casting
- recursive ray tracing
- modeling transforms
- cheap tricks
- **optimizations**

ray tracing complexity

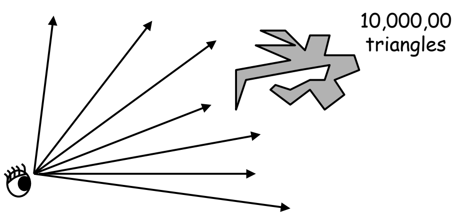
O(# of intersection tests)
= O(#pixels x # objects)

Can we reduce the number of intersection tests?

optimization

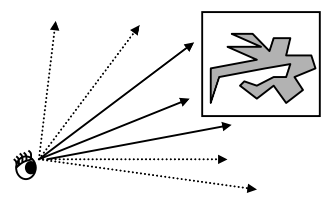
- bounding boxes
- oct-trees
- BSP-trees

bounding boxes: intuition



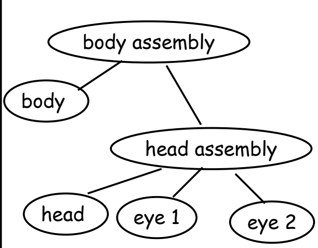
most rays miss the object

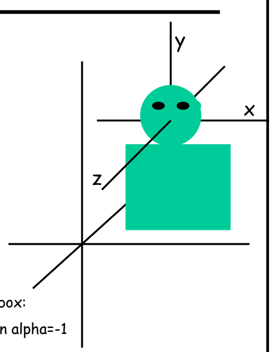
bounding boxes



1. rule out rays by simple intersection test with bounding box
2. perform exhaustive test on remaining rays

scene graph



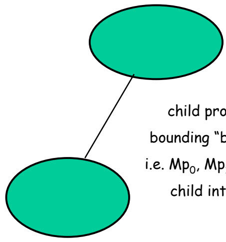


each node has an axis-aligned bounding box:
test ray against box – if no intersection return alpha=-1
otherwise proceed as usual

constructing the bounding boxes

- Leaf of scene graph
  - Sphere: radius r, center $c=(c_x,c_y,c_z)$
    Corners of bounding box are_________
  - Triangle: vertices (x_0,y_0,z_0), (x_1,y_1,z_1), (x_2,y_2,z_2)
    Corners of bounding box are__________

constructing bounding box

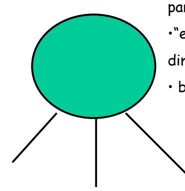


child provides parent the corners of its bounding "box" in parent's coordinate system i.e. $Mp_0, Mp_1, \dots, Mp_7$ where M transforms the child into its parent coordinate system

bounding box with corners

$p_0, p_1, \dots, p_7$

constructing bounding box



parent computes

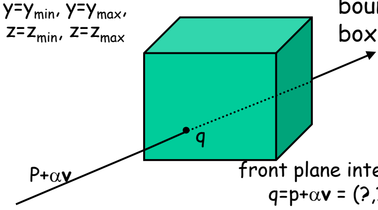
- "extrema" of coordinates in each dimension: $x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}$
- bounding corners

$p=(x,y,z)$ where $x \varepsilon \{x_{min},x_{max}\}$, $y \varepsilon \{y_{min},y_{max}\}$, $z \varepsilon \{z_{min},z_{max}\}$

bounding box intersection

1. find intersection of ray with each bounding plane of box

$x=x_{min}$, $x=x_{max}$,
$y=y_{min}$, $y=y_{max}$,
$z=z_{min}$, $z=z_{max}$



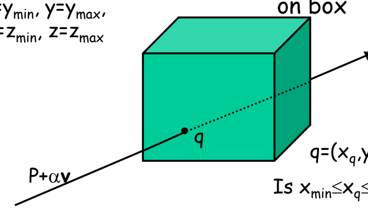
front plane intersection:
$q=p+\alpha\mathbf{v} = (?,?,z_{max})$

$q=(p_x+\alpha\mathbf{v}_x, p_y+\alpha\mathbf{v}_y, p_z+\alpha\mathbf{v}_z)$
where $\alpha = (z_{max}-p_z)/v_z$

bounding box intersection

2. determine if any intersection point is on box

$x=x_{min}$, $x=x_{max}$,
$y=y_{min}$, $y=y_{max}$,
$z=z_{min}$, $z=z_{max}$



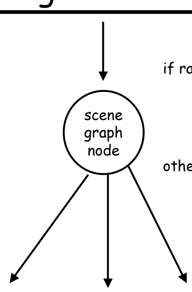
$q=(x_q,y_q,z_q)$

Is $x_{min} \le x_q \le x_{max}$ and

$y_{min} \le y_q \le y_{max}$ and

$z_{min} \le z_q \le z_{max}$?

bounding box intersection test



if ray intersects with node's bounding

check for intersection with local object and

pass ray to children

otherwise return no intersection

REMEMBER: bounding box encloses objects at node and all bounding boxes of children

optimization

- bounding boxes
- oct-trees
- BSP-trees

We won't cover these now. Look on the web for details.