COMS 30115

Carl Henrik Ek - carlhenrik.ek@bristol.ac.uk

February 9th, 2018

http://www.carlhenrik.com

Last time

- BRDFs
    - a more realistic way to parametrise ray-surface interaction
- Approximations
    - Bump and Environment mapping

Today

- Shadows
  - Last element of raytracer lab
- Cameras
- Optimisations
  - ideas
- Summarise raytracing

The Book

- Shadows URL
- Cameras URL
- Optimisations
  - Jacco Bikker Flipcode
  - Gavan Woolery, Gamasutra, Why I still Think Raytracing is the Future
  - Accelerated Structures, Scratchapixel
  - Robin Marcus, Realtime rendering blog - 4 part article about realtime raytracing

Shadows

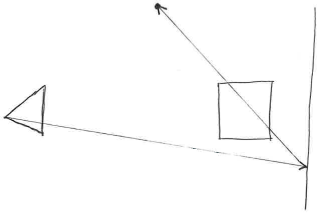
Shadows

1

[1] X-Files Universe

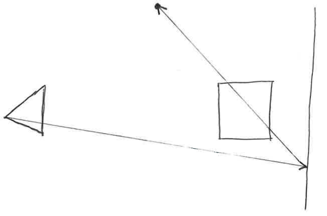
Direct Shadows

- Find intersection camera ray and surface



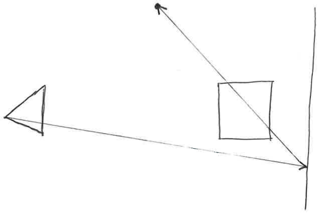
Direct Shadows

- Find intersection camera ray and surface
- Shoot ray from surface towards light



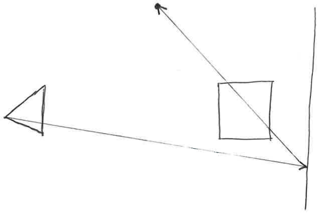
Direct Shadows

- Find intersection camera ray and surface
- Shoot ray from surface towards light
- called a shadow ray



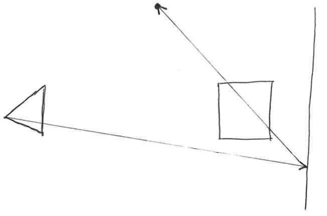
Direct Shadows

- Find intersection camera ray and surface
- Shoot ray from surface towards light
- called a shadow ray
- If ray intersect other surface before light then surface is in shadow by this light



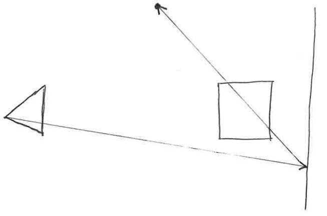
Direct Shadows

- Find intersection camera ray and surface
- Shoot ray from surface towards light
- called a shadow ray
- If ray intersect other surface before light then surface is in shadow by this light
- Shade accordingly



Direct Shadows

- Find intersection camera ray and surface
- Shoot ray from surface towards light
- called a shadow ray
- If ray intersect other surface before light then surface is in shadow by this light
- Shade accordingly
- *Remember that ordering is not important*

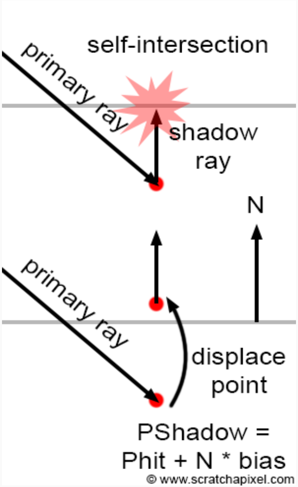


Shadow Acne

- Self-intersection
- If our primitive is a triangle we do not have self-shadows
- Numerical error
- *you will see this in the lab*
- shadow bias

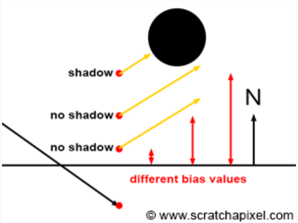
Shadow Acne

- Self-intersection
- If our primitive is a triangle we do not have self-shadows
- Numerical error
- *you will see this in the lab*
- shadow bias

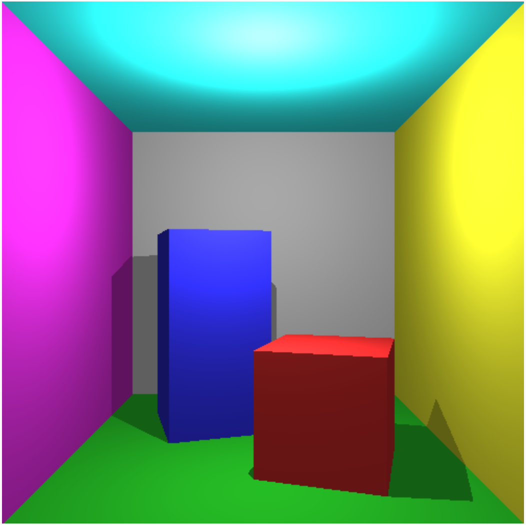


Shadow Acne

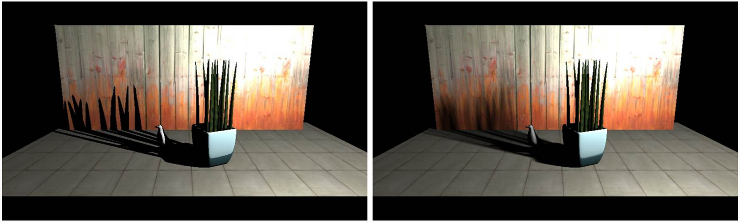
- Self-intersection
- If our primitive is a triangle we do not have self-shadows
- Numerical error
- *you will see this in the lab*
- shadow bias



Lab 1



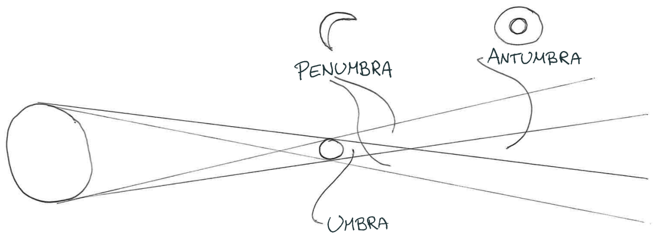
Soft Shadows



[2]

[2] Image URL

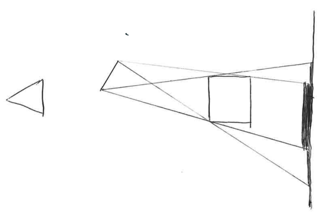
Soft Shadows



[2] Image URL

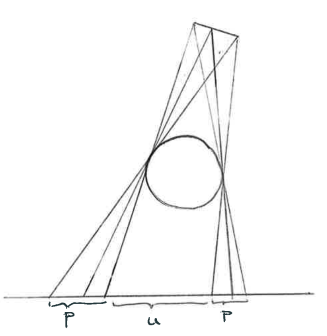
Soft Shadows

- Zero dimensional things aren't really that common
- Most light-sources have spatial extent
- umbra - hard shadow
- penumbra - partial shadow
- antumbra - cross over



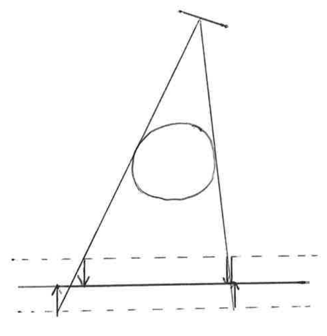
Soft Shadows

- Zero dimensional things aren't really that common
- Most light-sources have spatial extent
- umbra - hard shadow
- penumbra - partial shadow
- antumbra - cross over
- Apprixomate using several point/spot lights



Soft Shadows

- Zero dimensional things aren't really that common
- Most light-sources have spatial extent
- umbra - hard shadow
- penumbra - partial shadow
- antumbra - cross over
- Apprixomate using several point/spot lights
- Approximate with several intersection planes



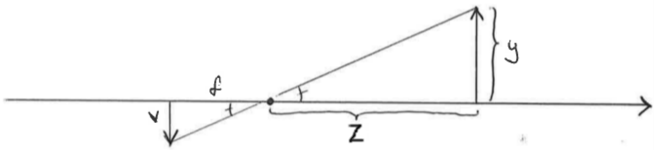
Summary

- Shadows adds a lot of realism
- Expensive to compute as you will need more rays
- If you use hard shadows compensate with a lot of ambient light

Cameras

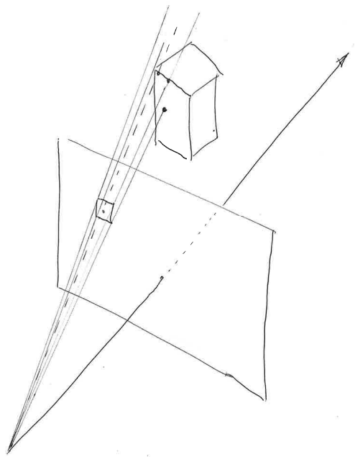
Image

Pinhole Cameras



- Raytraced images often looks too "clean"
- infinite depth-of-field

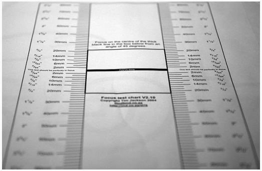
Anti-Aliasing



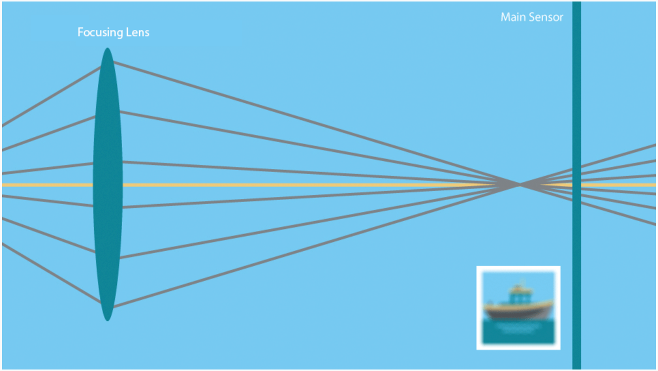
- shoot several rays per pixel and blend colours together

Depth-of-Field

- Film needs to be bombarded with sufficient energy to generate a colour
- Opening for rays bigger than a one photon wide pinhole: Aperture
- Same as glossy reflection, each ray hitting the eye comes from different parts of the world

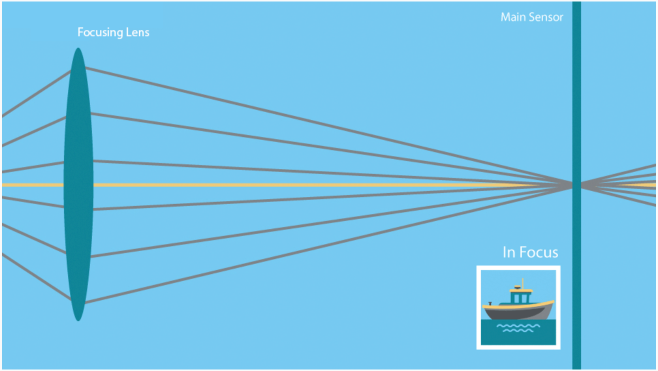


Depth-of-Field[3]



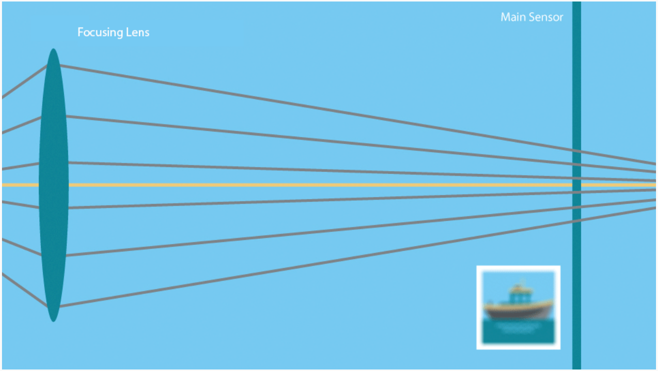
[3] https://www.bhphotovideo.com/explora/photography/tips-and-solutions/how-focus-works

Depth-of-Field[3]



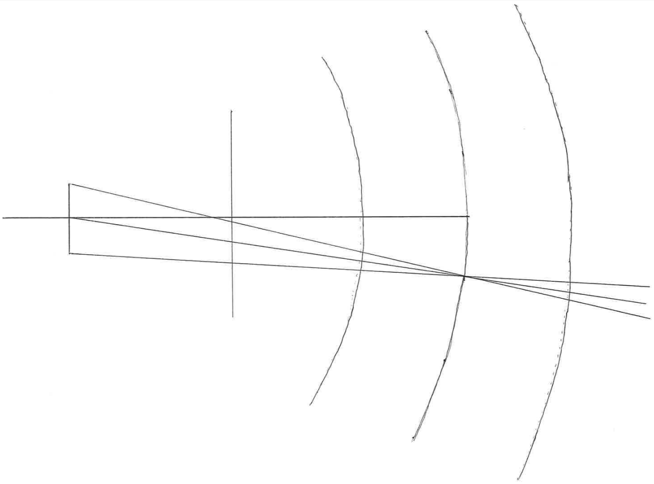
[3] https://www.bhphotovideo.com/explora/photography/tips-and-solutions/how-focus-works

Depth-of-Field[3]



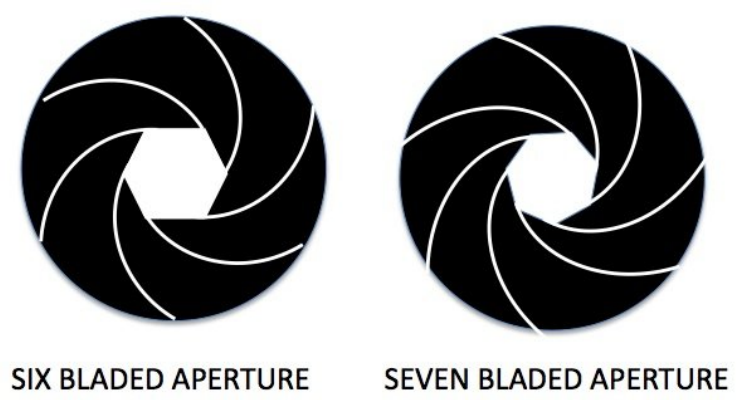
[3] https://www.bhphotovideo.com/explora/photography/tips-and-solutions/how-focus-works

Depth-of-Field

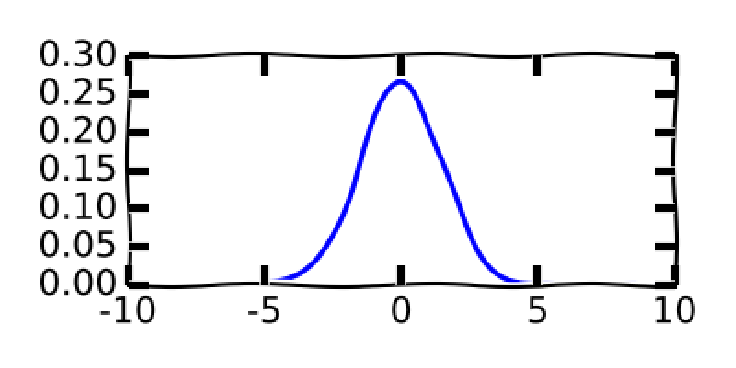


- Which rays to trace, how many?
- Different geometry aperture creates different images

Aperture



Random is your friend[4]



- *We (humans) are really good at picking up regularities, avoid them for increased fidelity*

[4] http://www-alg.ist.hokudai.ac.jp/~jan/randsphere.pdf

Lens Flare[5]

- A real camera has a lens
- Transition between two media causes refraction
- Light bounces inside lens
- Refraction calculations inside lens

[5] URL

Summary

- We have seen quite a few things now
  - reflection, refraction
  - shadows
  - depth-of-field
  - reflection models
  - material properties
  - etc.

Summary

- We have seen quite a few things now
  - reflection, refraction
  - shadows
  - depth-of-field
  - reflection models
  - material properties
  - etc.
- All follows the same principle
  - *basic physics*
  - *If light can be assumed to be a ray, then we just have to follow it along its path!*

Optimisation

So far

Ek In the first part of the course we will look at raytracing

So far

Ek In the first part of the course we will look at raytracing

UoB Damn this is slow, I can't even move the camera, or wait, did it move?

So far

Ek In the first part of the course we will look at raytracing

UoB Damn this is slow, I can't even move the camera, or wait, did it move?

Ek Well thats ray-tracing, try to reduce the screen-size to 10×10 so that we can see something

So far

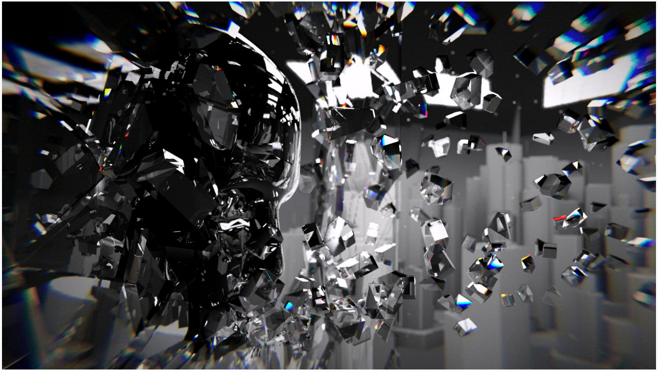
Ek In the first part of the course we will look at raytracing

UoB Damn this is slow, I can't even move the camera, or wait, did it move?

Ek Well thats ray-tracing, try to reduce the screen-size to 10×10 so that we can see something

UoB Really useful this unit is, the graphics is both ugly and slow

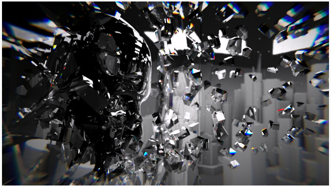
Smash of Fairlight



WTF

5 Faces

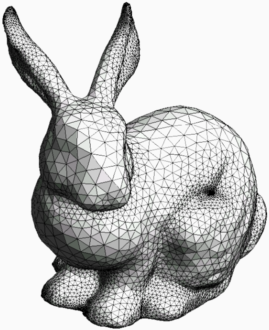
- Refraction
- Secondary rays in several levels (lots)
- Depth of Field
- Implicit surfaces
- Real-Time

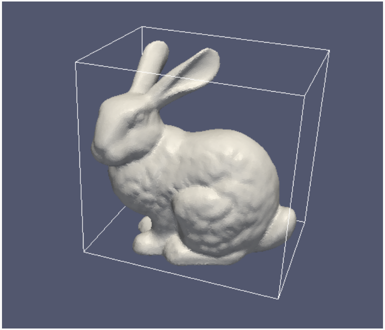


Optimisation

- Structure our data to do less calculations
  - Bounding Boxes, Tree's, etc.
- Optimise your code
- Reduce visual quality/correctness for speed

Data Structures

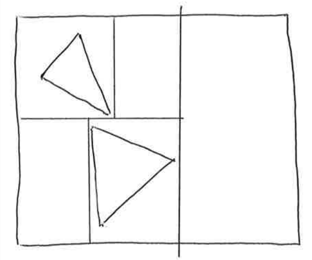




Bounding Boxes

- Intersections are expensive to compute
- Create simpler geometry that surrounds object
- The tighter the bounding volume the better but that usually requires more vertices

KD-Trees



- Do we need to check all object everywhere in space?
- Create a KD-Tree that splits up space
- Find large areas with single object

Cramer's Rule

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

$$= \left\{\det(\mathbf{a}, \mathbf{b}, \mathbf{c}) = -(\mathbf{a} \times \mathbf{c})^{\mathrm{T}}\mathbf{b} = -(\mathbf{c} \times \mathbf{b})^{\mathrm{T}}\mathbf{a}\right\}$$

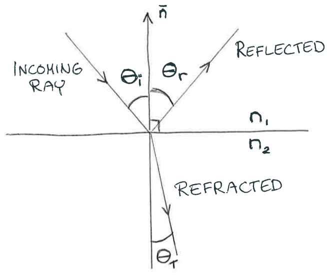
$$= \frac{1}{\mathbf{p}^{\mathrm{T}}\mathbf{e}_1} = \begin{pmatrix} \mathbf{q}^{\mathrm{T}}\mathbf{e}_2 \\ \mathbf{p}^{\mathrm{T}}\mathbf{s} \\ \mathbf{q}^{\mathrm{T}}\mathbf{d} \end{pmatrix}$$

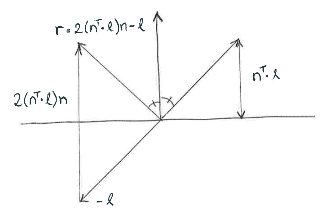
$$\mathbf{p} = \mathbf{d} \times \mathbf{e}_2$$

$$\mathbf{q} = \mathbf{s} \times \mathbf{e}_1$$

- Computing an inverse is more work $\mathbf{A}^{\mathrm{T}}\mathbf{A} = \mathbf{I}$

Refraction Rays





Refraction

$$\mathbf{t} = \left(\frac{n_i}{n_t} \cos(\theta_i) - \sqrt{1 - \left(\frac{n_i}{n_t}\right)^2 cos^2(\theta_i)} \right) \mathbf{n} - \frac{n_i}{n_t} \mathbf{i}$$

Refraction Rays

```
float b = Ni/Nt; float b2 = b*b;
float ni =N.x*I.x+N.y*I.y+N.z*I.z; //3m+2a
float ni2 = ni*ni;                 //1m
float D2 = 1.0f - b2*ni2;          //1m+1a
a = b*ni-sqrtf(D2);                //1m+1a+1sqrt
T.x = a*N.x-b*I.x;
T.y = a*N.y-b*I.y;
T.z = a*N.z-b*I.z;                 //6m+3a
```

Total: 12 multiplications, 6 additions, 1 square-root

[6] http://hugi.scene.org/online/hugi23/torefrac.htm Code on the course website

Refraction Rays

Code Optimisation

- What can we pre-compute?
- Something that does not have a massive range
- Something that has few indexing variables

[6] http://hugi.scene.org/online/hugi23/torefrac.htm Code on the course website

Refraction Rays

a is just a function of **ni** it can be pre-computed and tabled (**ni** is bounded)

```
float scalefac = 16384;
float *aLUT;

float ni=N.x*I.x+N.y*I.y+N.z*I.z; //3m+2a
float a=aLUT[(int)(ni*scalefac)]; //1m+1lu

T.x = a*N.x-b*I.x;
T.y = a*N.y-b*I.y;
T.z = a*N.z-b*I.z;                  //6m+3a
```

Total: 10 multiplications, 5 additions, 1 look-up

[6] http://hugi.scene.org/online/hugi23/torefrac.htm Code on the course website

Refraction Rays

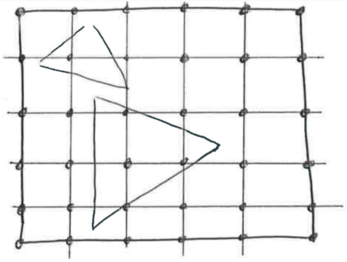
Do we need the vector to be normalised, if not we can table $\mathbf{g} = \frac{\mathbf{a}}{\mathbf{b}}$

```
float scalefac = 16384;
float *gLUT; //(a/b)

float ni=N.x*I.x+N.y*I.y+N.z*I.z; //3m+2a
float g=gLUT[(int)(ni*scalefac)]; //1m+1lu

T.x = g*N.x-I.x;
T.y = g*N.y-I.y;
T.z = g*N.z-I.z;                  //3m+3a
```

- Total: 7 multiplications, 5 additions, 1 look-up
- *This version is about twice as fast as the first*[6]

[6] http://hugi.scene.org/online/hugi23/torefrac.htm Code on the course website

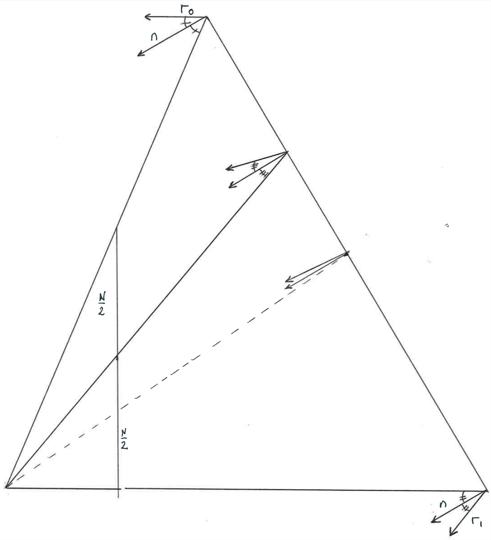
Reduce Visual Quality



Interpolation

- Light calculations are expensive
- Interpolation is cheap
- Compute lighting in few places and interpolate

Reduce Visual Quality



Summary

Summary

- Calculate number of operations
  - is there a less general way to do this
  - are there special cases, then code all of them
- Use heuristics
  - what is the most general case I'll see
- What shows and what doesnt?
- Profile your code

Raytracing

- Know your two weapons
  - Inner product

$$\mathbf{x}^{\mathrm{T}}\mathbf{y} = ||\mathbf{x}||||\mathbf{y}||\cos(\theta)$$

  - Outer product

$$\begin{aligned} \mathbf{x} \times \mathbf{y} =& \mathbf{z} \\ & \mathbf{z} \perp \mathbf{x} \\ & \mathbf{z} \perp \mathbf{y} \end{aligned}$$

  - Think how things work physically
    - how can we "mimick" this behaviour?
  - *Thats rendering for you*

Next Time

Next Time

Lecture Monday 13th of February

- We start with Real-time Graphics
- Rasterisation
- Will start with material for Lab 2

Lab Continue with Lab 1 and try to finish up to 50%

END

eof