University of BRISTOL

# COMS 30115

## Shading

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

February 23th, 2018

http://www.carlhenrik.com

# Introduction

# Last time

- Lines
- Triangles
  - spantables
  - barycentric coordinates
- Perpective correct interpolation

# Today

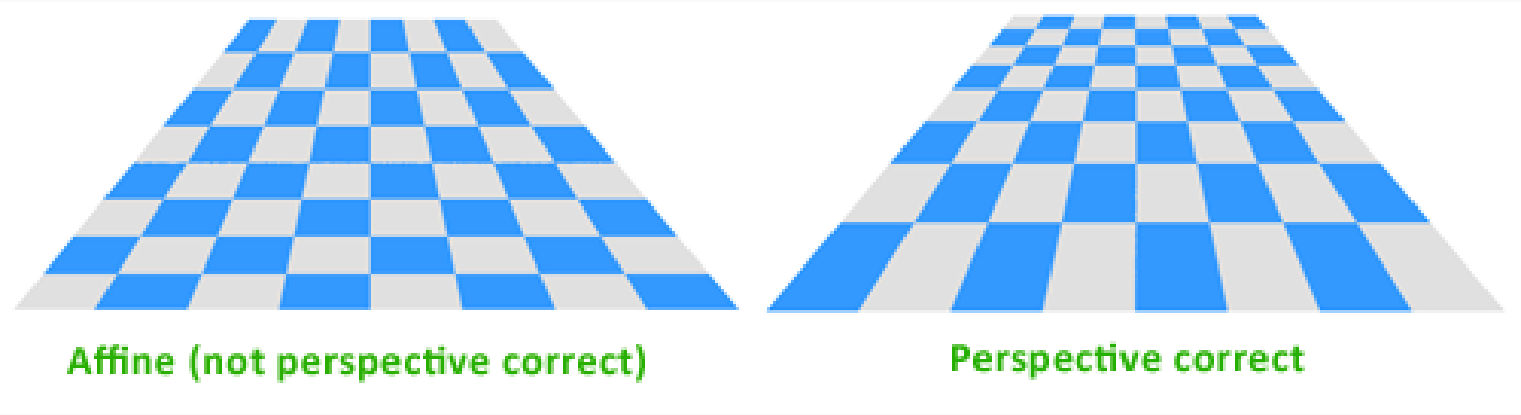
- Perspective correct interpolation of quantities
- Short re-cap on light
- Shading
    - what we want to interpolate

# The Book

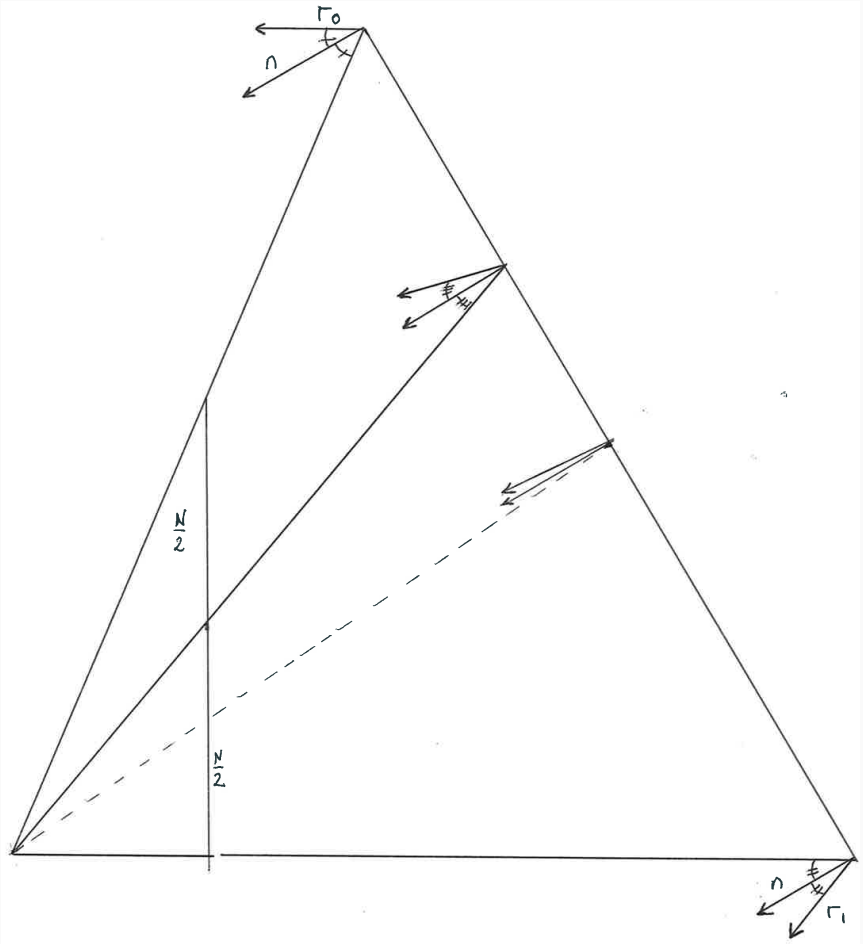
- Perspective Correct Interpolation and Vertex Attributes URL

# Perspective Correct

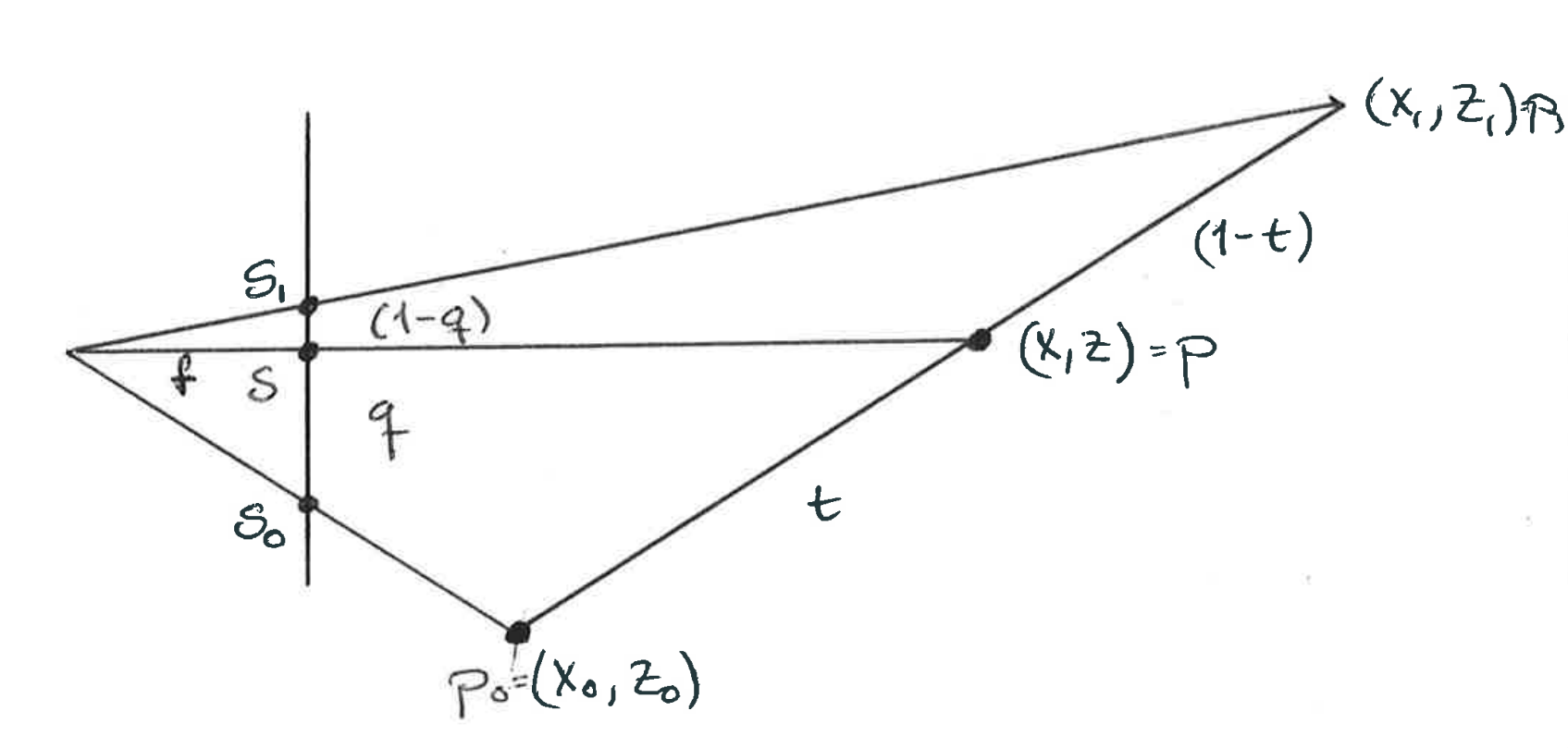
# Perspective Correct



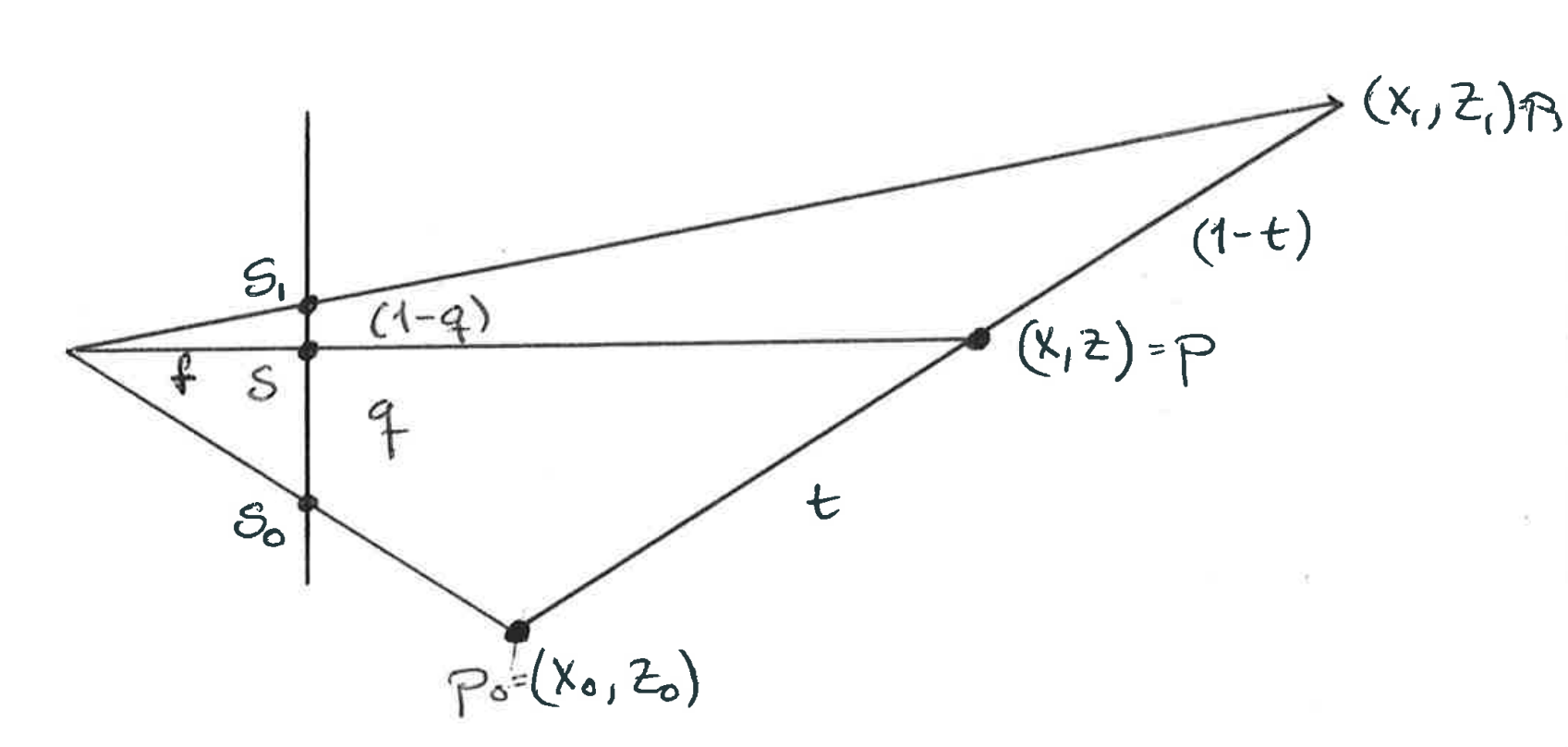
# Perspective Correct



# Perspective Correct



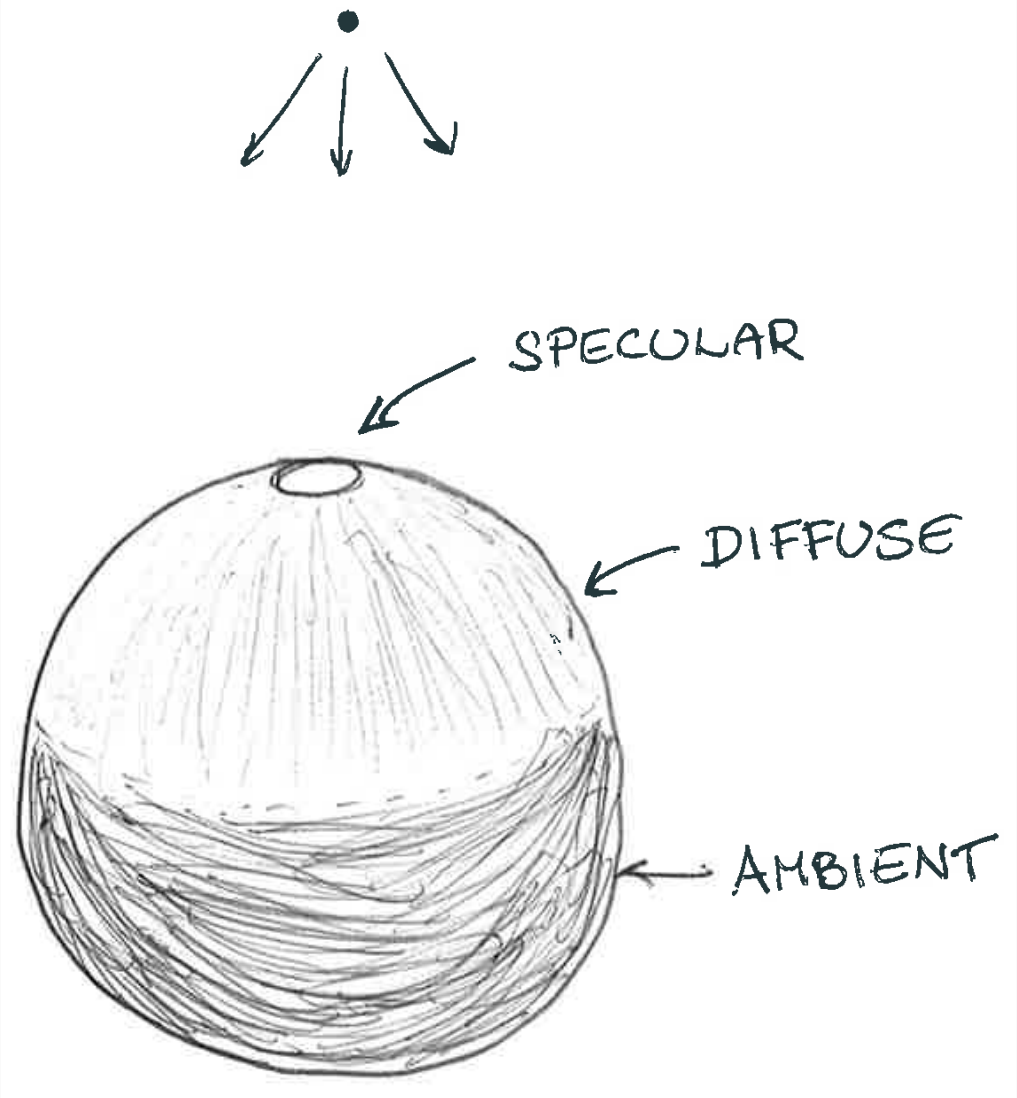
# Perspective Correct




$$\frac{1}{z} = \frac{1}{z_0}(1-q) + \frac{1}{z_1}q$$

# Shading

# Lights Ball

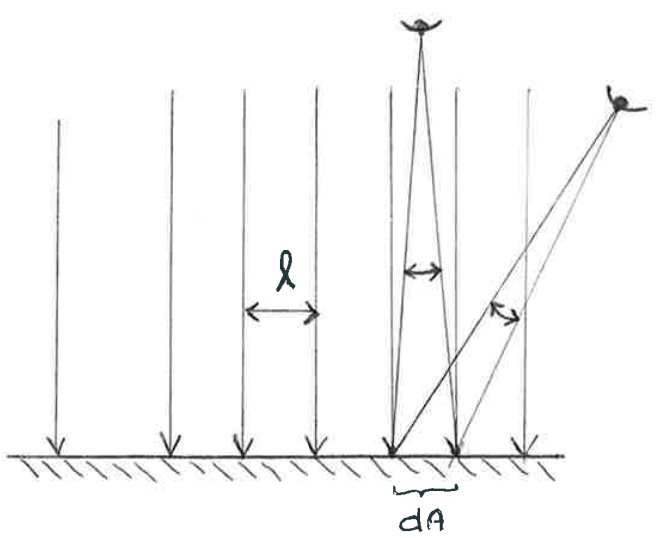


# Light Factorisation

$$\mathbf{i}_{tot} = f(\mathbf{i}_{amb}, \mathbf{i}_{diff}, \mathbf{i}_{spec})$$

- There is only one type of light
- Approximation: Factorise into Ambient, Diffuse and Specular

# Diffuse/Lambertian




- View independent
- Lambertian Surface (Looks the same from all directions)

$$i_{diff} = \mathbf{n}^{\mathrm{T}}\mathbf{l} = \cos\theta$$

$$\mathbf{i}_{diff} = \max\left((0, \mathbf{n}^{\mathrm{T}}\mathbf{l})\right) \mathbf{m}_{diff} \circ \mathbf{s}_{diff}$$

# Specular

- View dependent
- Non-linear "highlights"
  $\mathbf{r} = 2(\mathbf{n}^{\mathrm{T}}\mathbf{l})\mathbf{n} - \mathbf{l}$

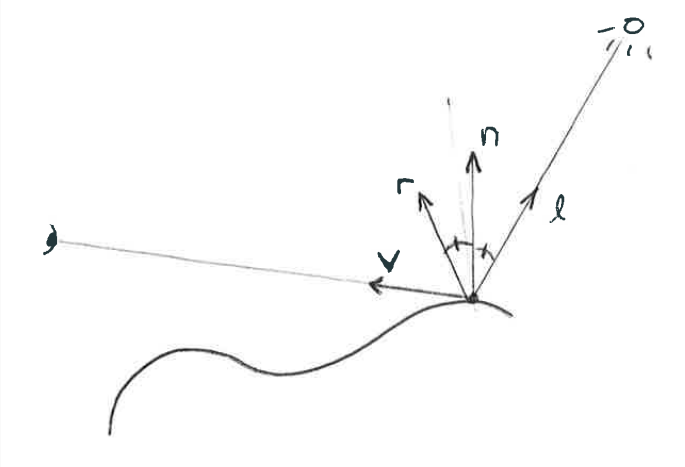
Phong $i_{spec} = (\mathbf{r}^{\mathrm{T}}\mathbf{v})^{m_{shi}}$

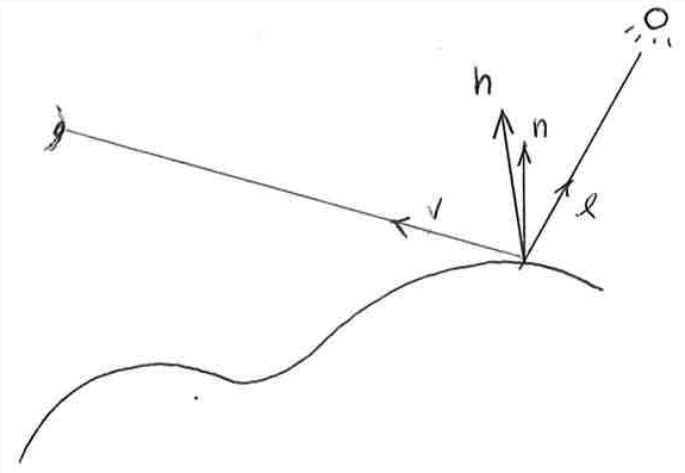
Blinn

$$i_{spec} = (\mathbf{n}^{\mathrm{T}}\mathbf{h})^{m_{shi}}$$

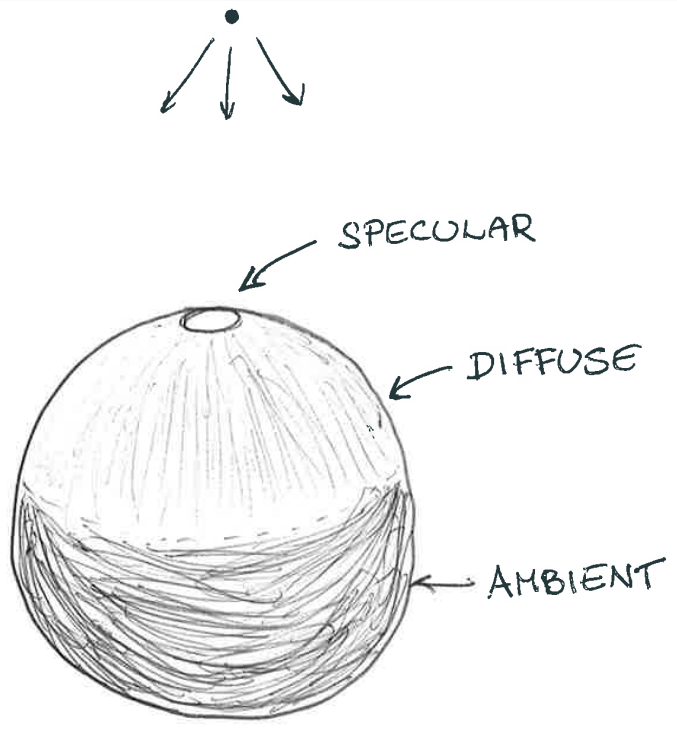
$$\mathbf{h} = \frac{\mathbf{l} + \mathbf{v}}{((\mathbf{l} + \mathbf{v})^{\mathrm{T}}(\mathbf{l} + \mathbf{v}))^{\frac{1}{2}}}$$

- Specular colour $\mathbf{i}_{spec} = \max((0, i_{spec}))\, \mathbf{m}_{spec} \circ \mathbf{s}_{spec}$



# Ambient




- Accounts for indirect light
- Not particularly realistic

$$\mathbf{i}_{amb} = \mathbf{m}_{amb} \circ \mathbf{s}_{amb}$$

# Distance Attenuation

$$\mathbf{i}_{tot} = f(\mathbf{i}_{amb}, \mathbf{i}_{diff}, \mathbf{i}_{spec})$$

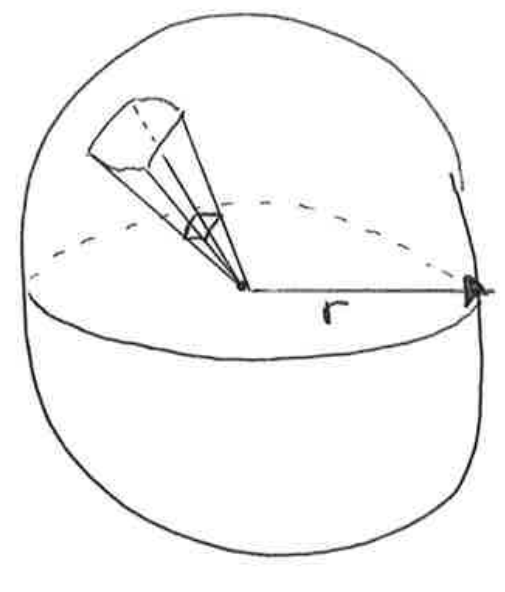
- Law of conservation of matter: $A = 4\pi r^2$
- Distance attenuation

$$d = \left(s_c + s_l \cdot r + s_q \cdot r^2\right)^{-1}$$

$$r = \left((\mathbf{s}_{pos} - \mathbf{p})^{\mathrm{T}}(\mathbf{s}_{pos} - \mathbf{p})\right)^{\frac{1}{2}}$$

- Surface emission: $\mathbf{m}_{emi}$

# Distance Attenuation



# Distance Attenuation

$$\mathbf{i}_{tot} = f(\mathbf{i}_{amb}, \mathbf{i}_{diff}, \mathbf{i}_{spec})$$

- Law of conservation of matter: $A = 4\pi r^2$
- Distance attenuation

$$d = \left(s_c + s_l \cdot r + s_q \cdot r^2\right)^{-1}$$

$$r = \left((\mathbf{s}_{pos} - \mathbf{p})^{\mathrm{T}}(\mathbf{s}_{pos} - \mathbf{p})\right)^{\frac{1}{2}}$$

- Surface emission: $\mathbf{m}_{emi}$

# All Together

$$
\begin{aligned}
\mathbf{i}_{tot} &= f(\mathbf{i}_{amb}, \mathbf{i}_{diff}, \mathbf{i}_{spec}) \\
&= \mathbf{m}_{emi} + \sum_{i=0}^{N-1} \Big( \mathbf{m}_{amb} \circ \mathbf{s}^i_{amb} \\
&\quad + \frac{\max((\mathbf{n}^{\mathrm{T}}\mathbf{l}^i), 0)\mathbf{m}_{diff} \circ \mathbf{s}^i_{diff} + \max(((\mathbf{r}^i)^{\mathrm{T}}\mathbf{v}), 0)^{m_{shi}} \mathbf{m}_{spec} \circ \mathbf{s}^i_{spec}}{s^i_c + s^i_l \left((\mathbf{s}_{pos} - \mathbf{p})^{\mathrm{T}}(\mathbf{s}_{pos} - \mathbf{p})\right)^{\frac{1}{2}} + s^i_q \left((\mathbf{s}^i_{pos} - \mathbf{p})^{\mathrm{T}}(\mathbf{s}^i_{pos} - \mathbf{p})\right)^{\frac{1}{2}}} \Big)
\end{aligned}
$$

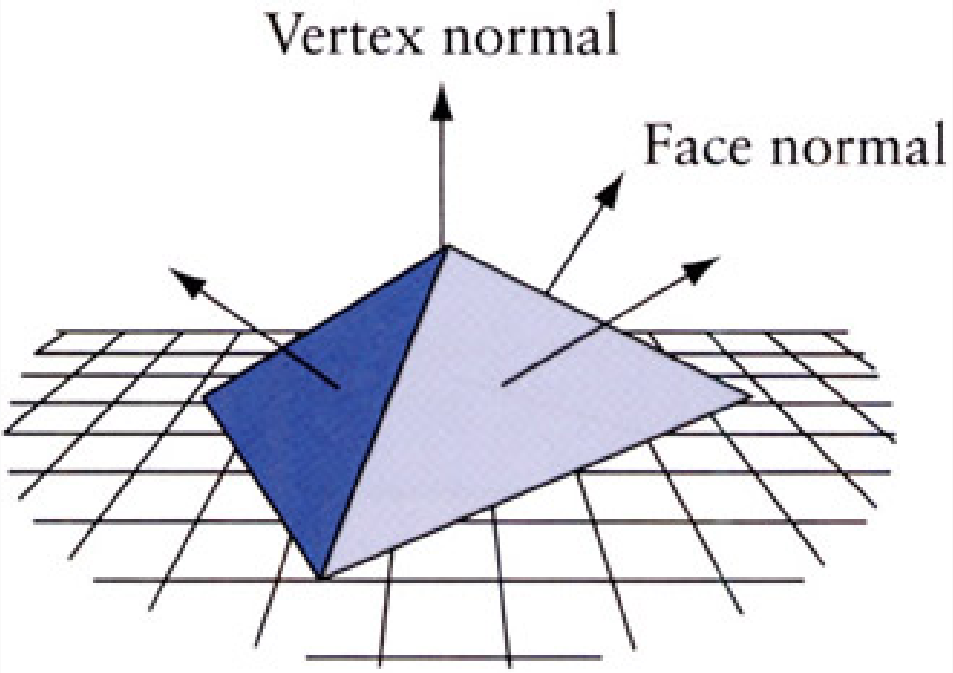
# Vertex Shading

- In the raytracer we did this computation per/pixel
- In the rasteriser we want to do this per/vertex and then interpolate the light
    - make few expensive computations

# Flat Shading

- shade for one point (center) of polygon and fill with that colour

# Vertex Normal



# Gouraud Shading

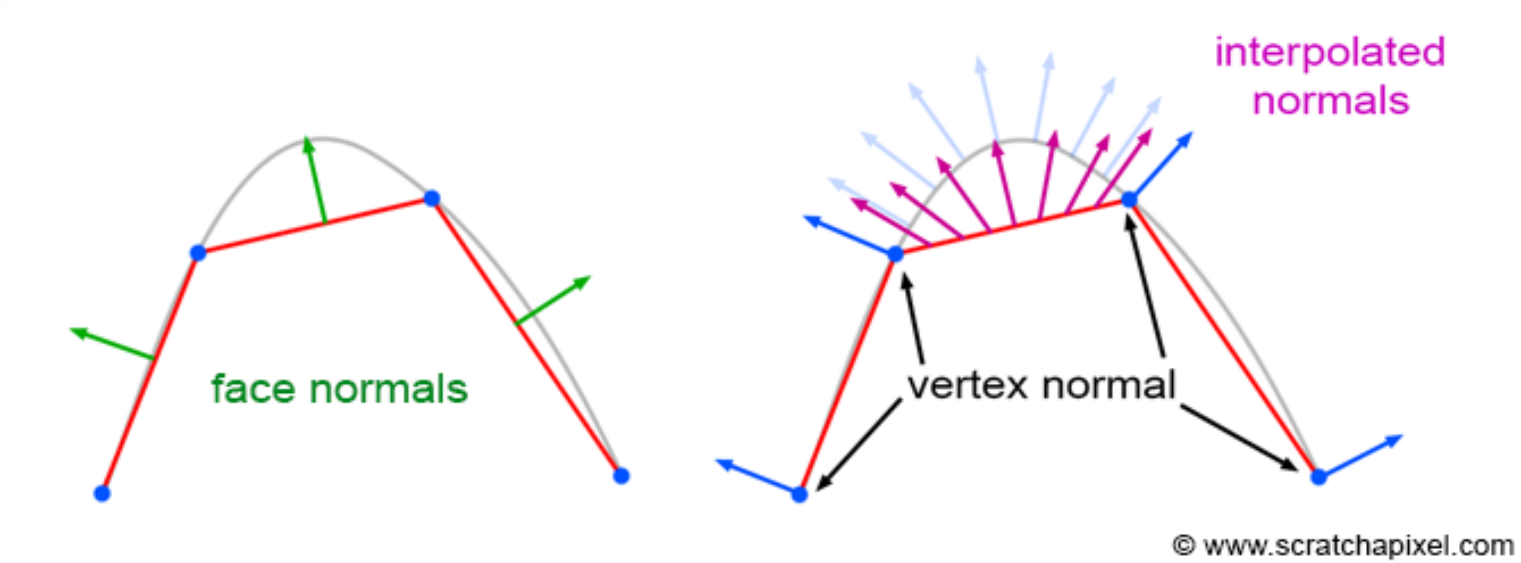
- Compute shading at each vertex

$$i_{a,b,c} = \max(0, \mathbf{l}_{a,b,c}^{\mathrm{T}} \mathbf{n}_{a,b,c})$$

$$i(u, v) = ?$$

- Interpolate shading across polygon
- Hard to get specular highlights correct

# Interpolated Normals



# Pixel Shader

- Interpolate vertex normals across polygon

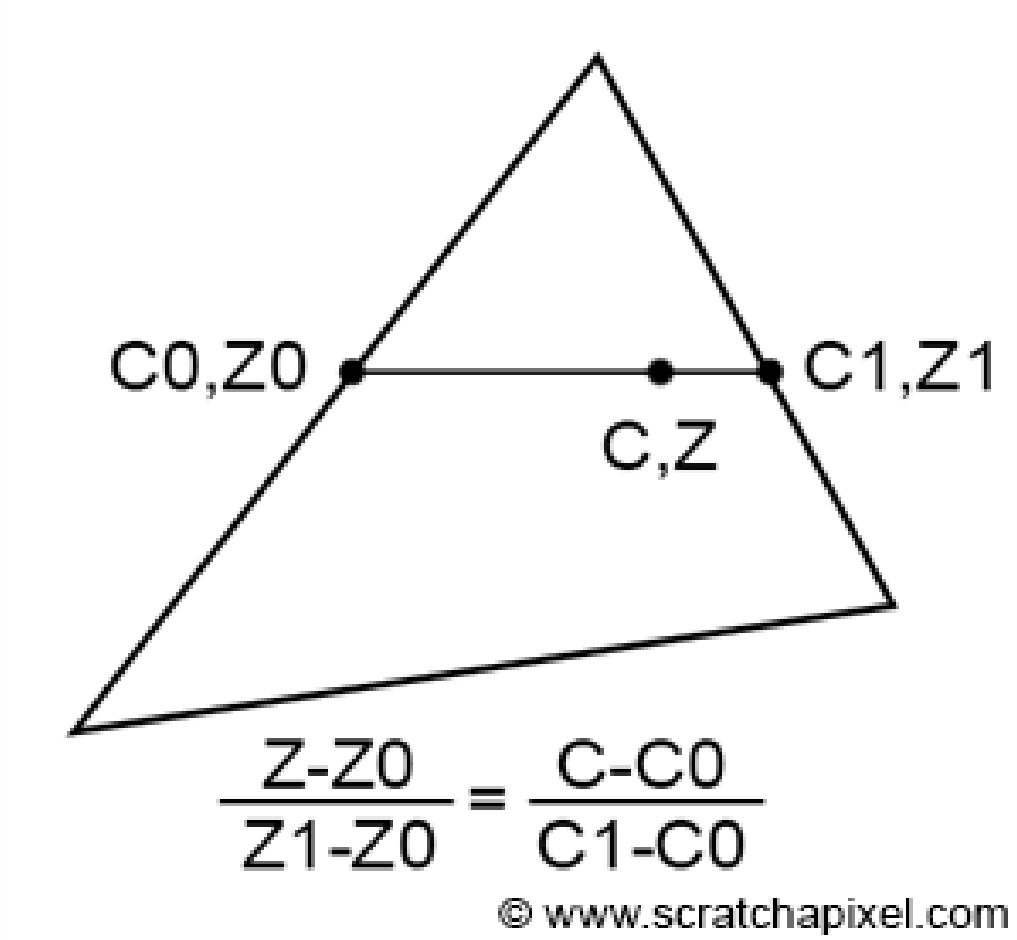
$$\mathbf{n}(u, v) = f(u, v, \mathbf{n}_{left}, \mathbf{n}_{right})$$

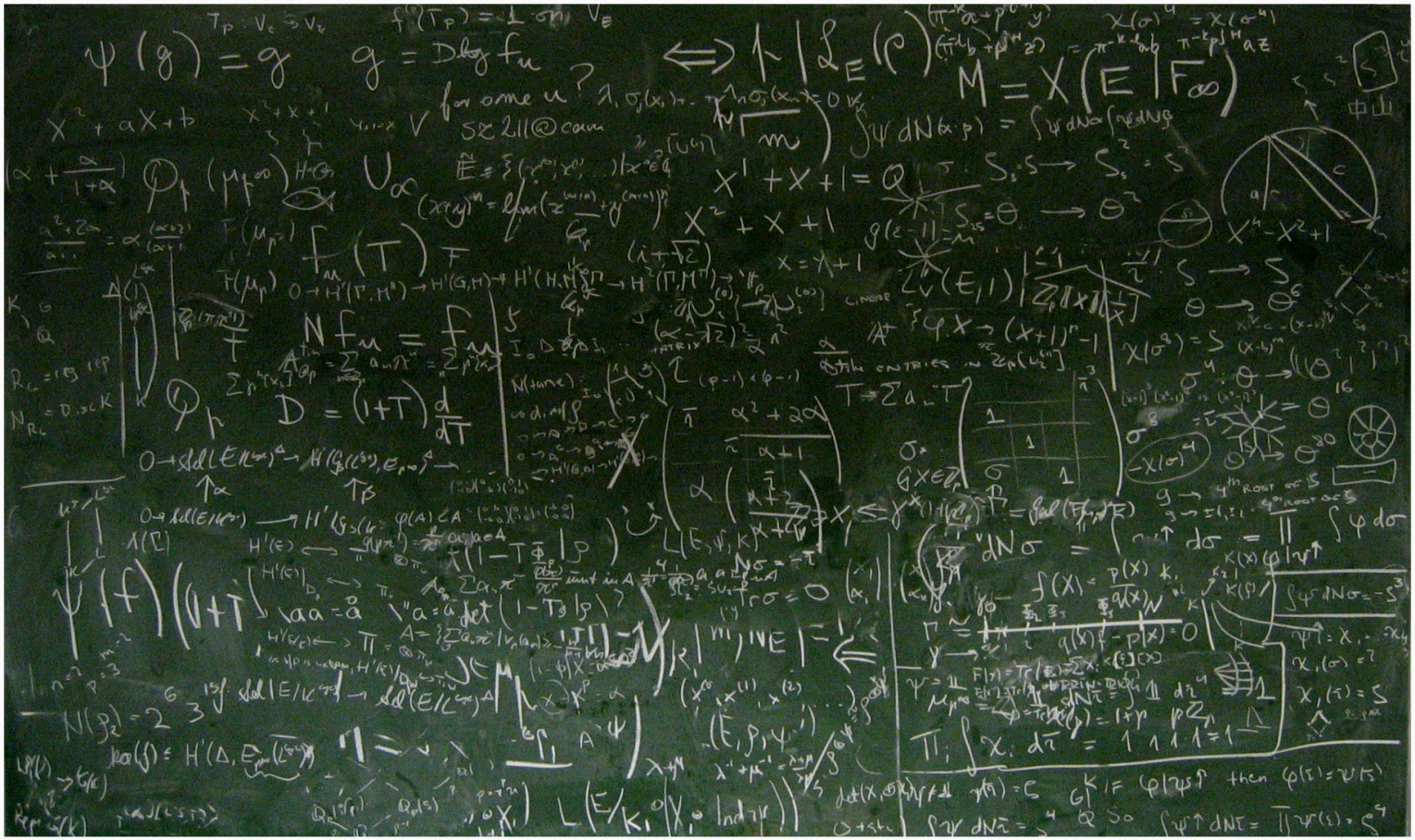
- Compute shading for each pixel

$$i(u, v) = \max(0, \mathbf{l}(u, v)^{\mathrm{T}} \mathbf{n}(u, v))$$

# Interpolation of Atributes

# Interpolation of Atributes





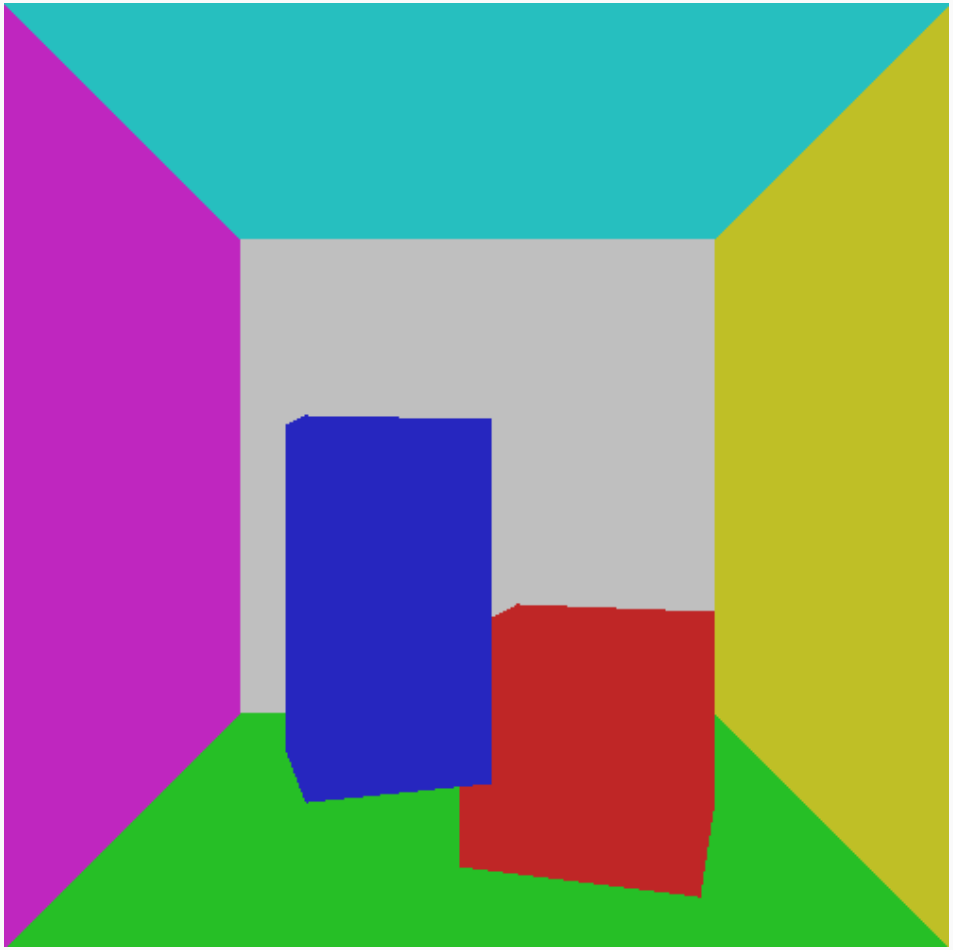
# Perspective Correct Interpolation

$$c(q) = z\left(\frac{c_0}{z_0}(1-q) + \frac{c_1}{z_1}q\right)$$
$$z(q) = \frac{1}{\frac{1}{z_0}(1-q) + \frac{1}{z_1}q}$$

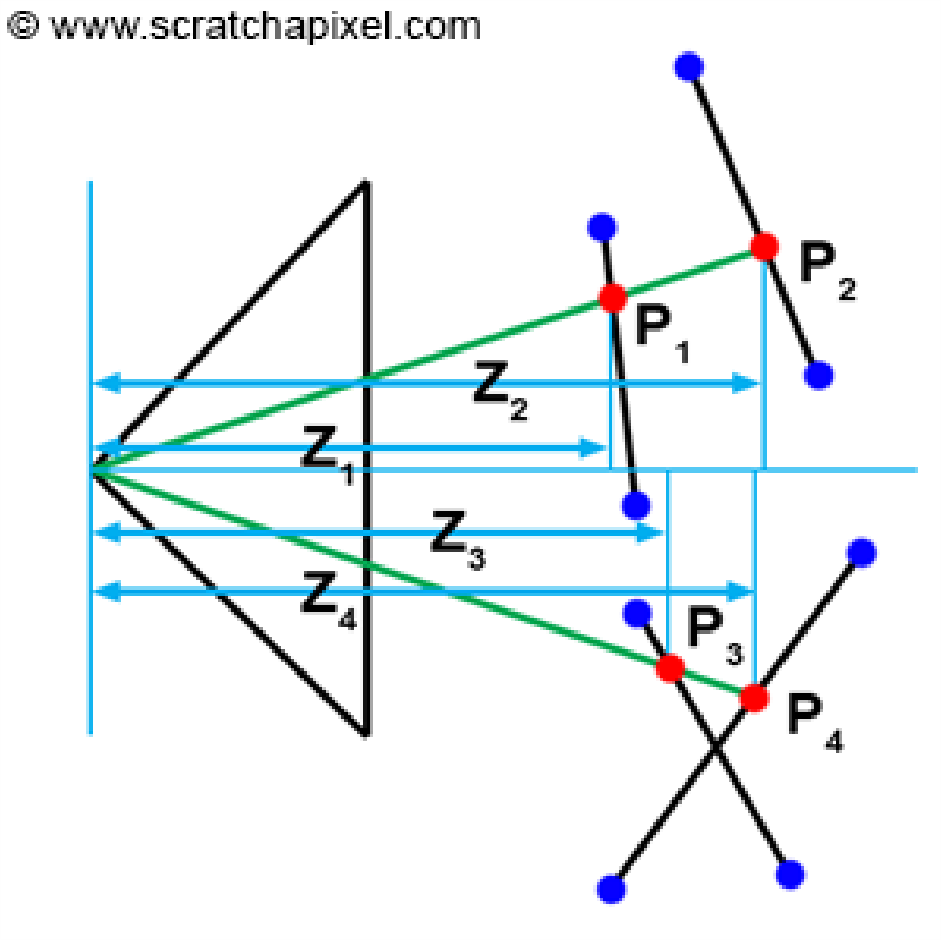
- Most likely the most important equations in a rasteriser
- Write a really flexible function that can interpolate perspective correct

# Depth Buffer

# Visibility



# Visibility



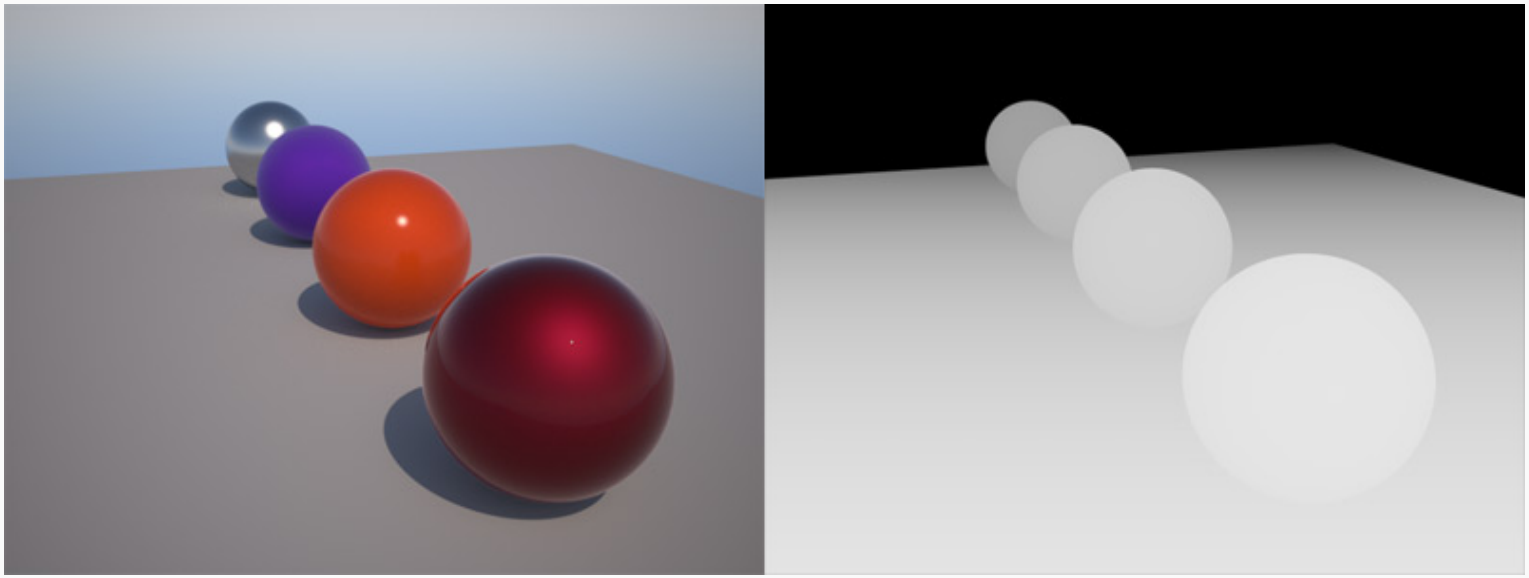
# Depth Buffer

1. Create buffer same size as screen
2. Initialise buffer with `max` type value
3. Loop through all primitives
4. If current `z` is smaller than depth value
    - $\Rightarrow$ draw pixel
    - Update depthbuffer with new `z`

# Visibility

```
uint32_t* depthbuffer = (uint32_t) malloc(sizeof(unint32_t)
                                          *width*height);
memset(depthbuffer, std::numerical_limits<uint32_t>::max(),
       sizeof(unint32_t)*width*height);
for(unit32_t i=0;i<nr_triangles;i++)
  {
    /* filling as normal */
    if(depthbuffer[v*width+u]>z)
      {
        depthbuffer[v*width+u] = z;
        /*
          draw pixel
         */
      }
  }
```

# Depth Buffer



# Fog

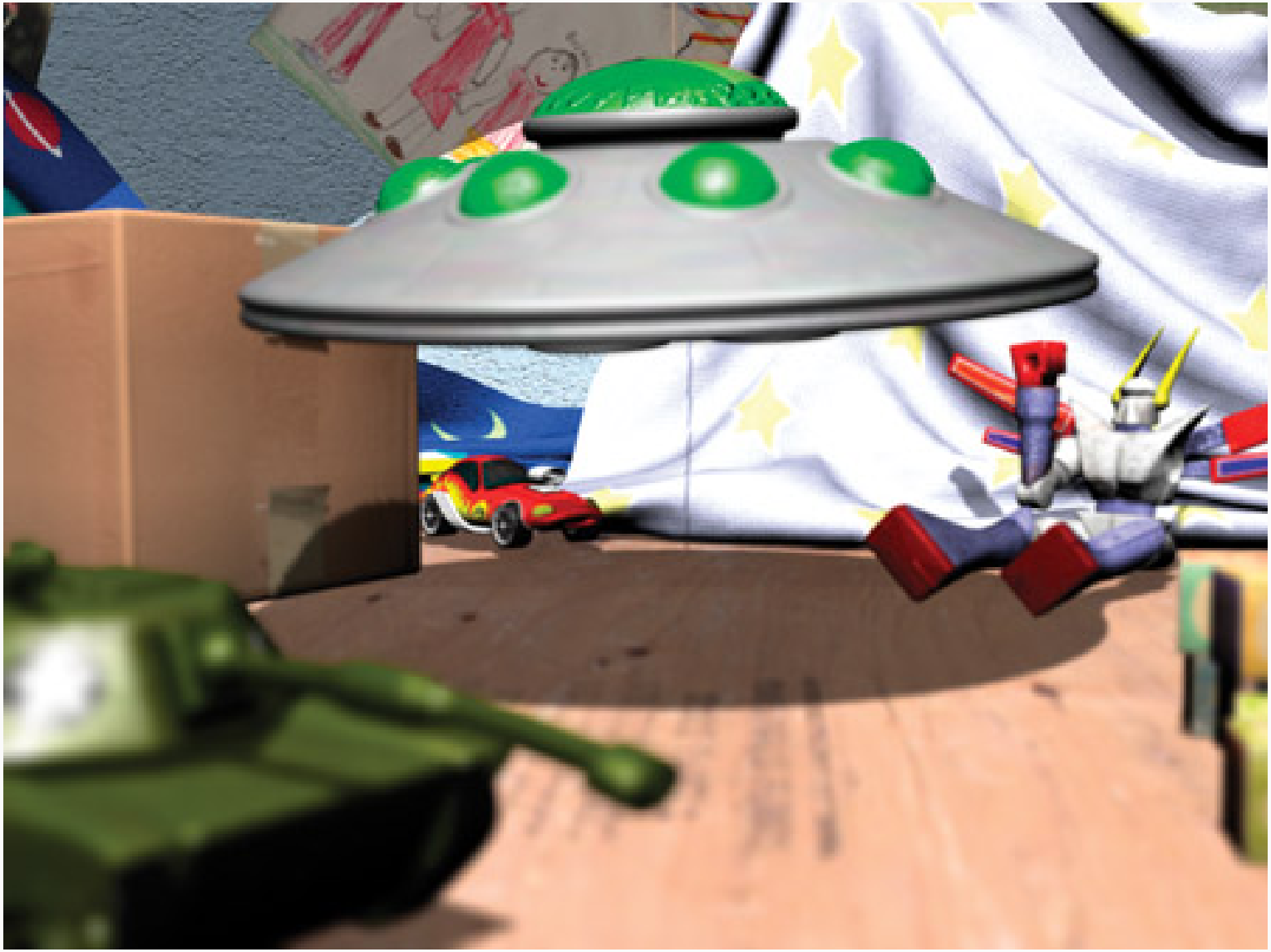
# Fog

$$c(u,v) = (1 - \lambda(u,v)) \cdot c^{\text{shading}}(u,v) + \lambda(u,v) c^{\text{fog}}(u,v)$$

$$\lambda_a = \max\left(\frac{r_a - r_0^{\text{fog}}}{r_1^{\text{fog}} - r_0^{\text{fog}}}, 0\right)$$

- We have already generated a depth buffer
- Blend pixels with a "fog" based on depth
- Compute blending at each vertex and interpolate out

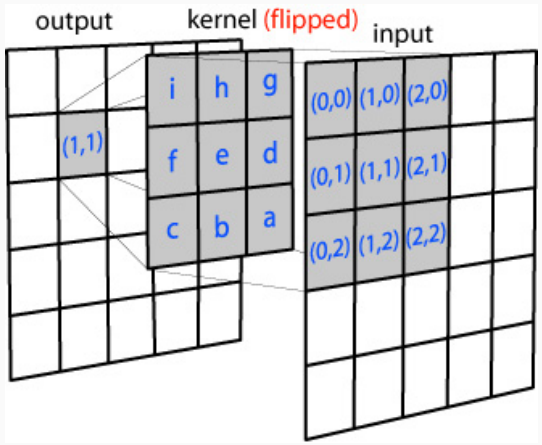
# Depth of Field



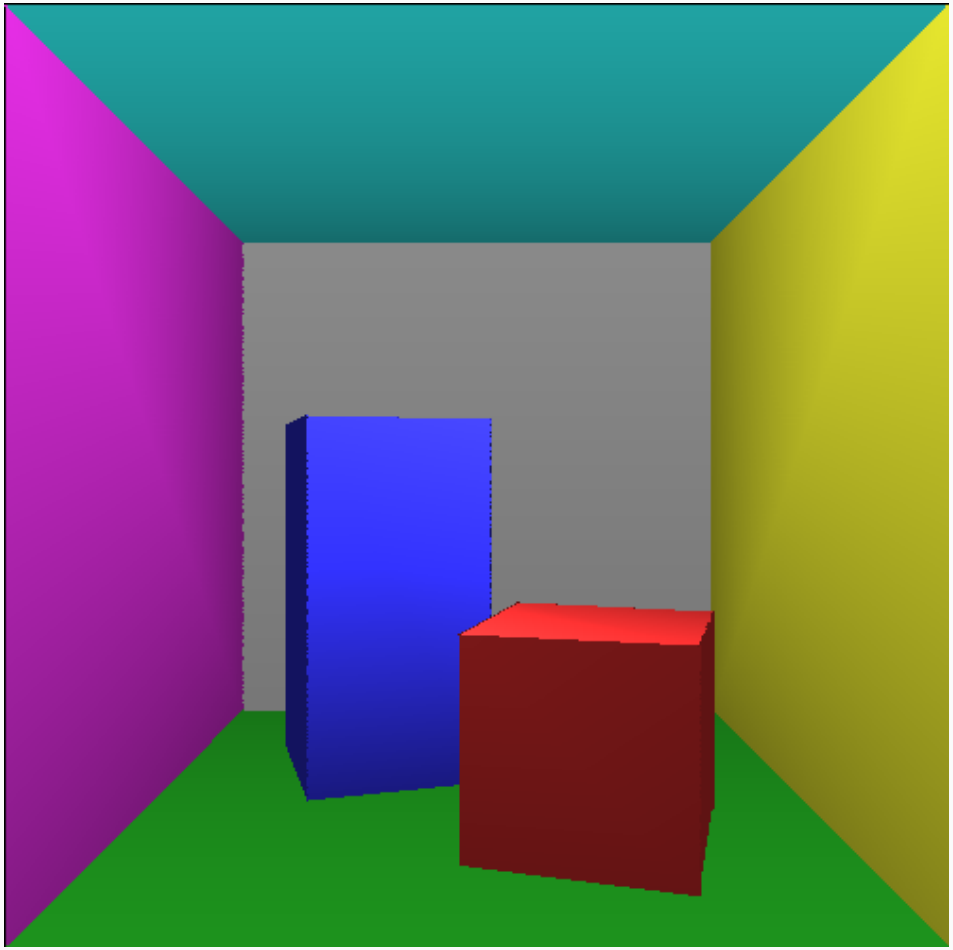
# Depth of Field

$$c(u,v) = \sum_{-1}^{1}\sum_{-1}^{1} k^z(x,y) \cdot c(u+x, v+y)$$

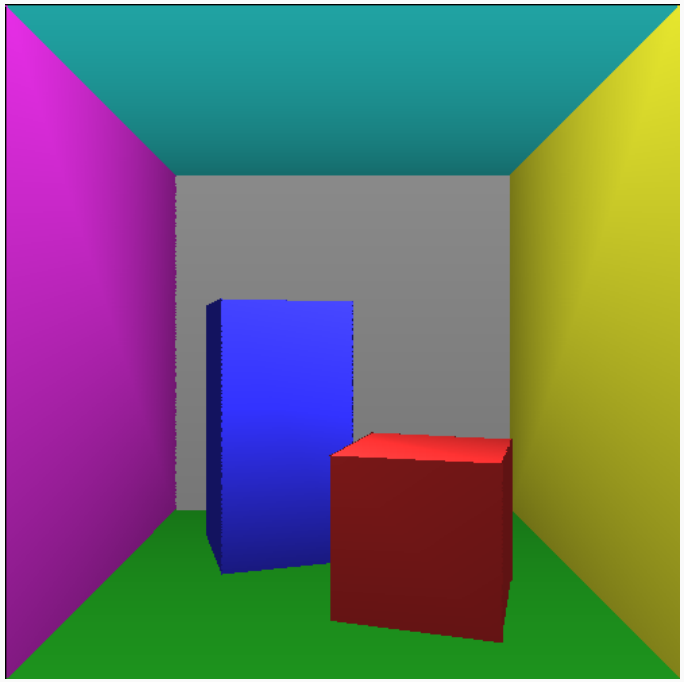
- Blur image based on depth
- Make the kernel dependent on depth



# Aliasing



# Aliasing



# Summary

# Summary

- Interpolation is the key to speed-up rendering
- Vertex shading vs. Pixel shading
- We have left the realms of physics, interpolate stuff and fake effects
  - fog, depth-of-field, aa, etc. etc.
  - come up with new things, play around, innovate

# Next Time

**Lecture** textures

- texture mapping

**Lab** continue with Lab 2

Labs will run during explore week

Lecture on the 5th of March cancelled

eof