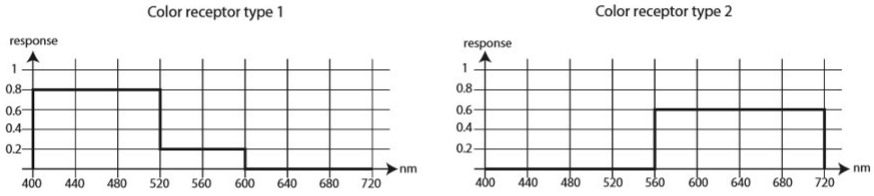
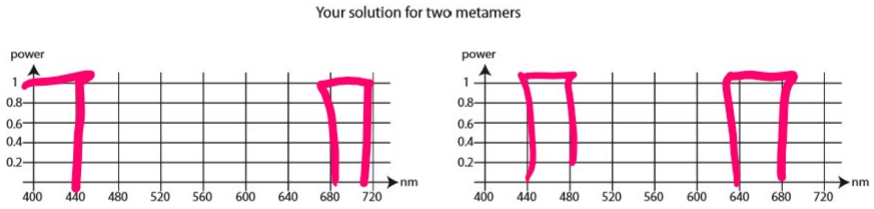
1 Colors

(a) **(2 points)** Explain the term *metamer*.

(b) **(2 points)** Assume there is an exotic animal species that has two types of color receptors with response curves as shown below. Sketch two spectral distributions that are metamers for the color perception of this species. Use the empty graphs in the sketch and give a short explanation for your solution. Note that there are many possible solutions.





(a) Many power spectra lead to the same perceived color

2 Raytracing

a) ray: $o+td = \begin{pmatrix} o_x \\ o_y \\ o_z \end{pmatrix} + t\begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix} = \begin{pmatrix} o_x + t d_x \\ o_y + t d_y \\ o_z + t d_z \end{pmatrix}$

paraboloid: $z = x^2 + y^2$

$$\Rightarrow \quad o_z + t d_z = (o_x + t d_x)^2 + (o_y + t d_y)^2$$

$$0 = \underbrace{o_x^2 + o_y^2 - o_z}_{=A} + \underbrace{(2(o_x d_x + o_y d_y) - d_z)t}_{=B} + \underbrace{(d_x^2 + d_y^2)t^2}_{=C}$$

$$t = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A} \qquad (*)$$

b) intersection point $P_0(x_0, y_0, z_0)$

Idea: Find two tangential vectors t_1, t_2 (1) $\rightarrow n = t_1 \times t_2$ (2)

(1): Das Paraboloid wird beschrieben durch $\vec{r}(x,y) = [x, y, x^2+y^2]$

$\frac{\partial \vec{r}}{\partial x}(x_0, y_0) = [1, 0, 2x_0] = t_1$ ("y festhalten")

"Steigung in x Richtung"

$\frac{\partial \vec{r}}{\partial y}(x_0, y_0) = [0, 1, 2y_0] = t_2$ ("x festhalten")

"Steigung in y Richtung"

$$\vec{n} = \begin{pmatrix} 1 \\ 0 \\ 2x_0 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ 2y_0 \end{pmatrix} = \begin{pmatrix} -2x_0 \\ -2y_0 \\ 1 \end{pmatrix}$$

c) Wenn der Strahl parallel zur z-Achse ist, hat er als direction $\begin{pmatrix} 0 \\ 0 \\ d_z \end{pmatrix}$

$\rightarrow$ nur 1 intersection point $\rightarrow$ bei (*) kommt nur 1 Lösung heraus

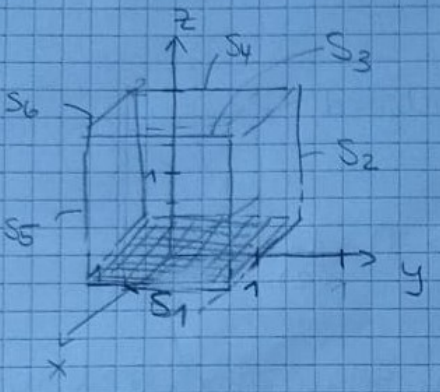
$\rightarrow B^2 - 4AC = 0$

Problem: Computer können nicht jede Zahl exakt als floating point Zahl darstellen; d.h. $B^2 - 4AC$ wird (evtl) nicht exakt Null geben (obwohl es das theoretisch sollte). Somit würden wir zwei Lösungen (nahe beieinander) erhalten, obwohl es nur eine gäbe.

(Lösung)
Im Falle von einem Strahl parallel zur z-Achse vereinfacht sich die (quadratische) intersection Gleichung zu einer linearen Gleichung und wir können t direkt berechnen mit

$$t = \frac{o_x^2 + o_y^2 - o_z}{d_z}$$

d) → bounding box
zuerst berechnen, ob es eine intersection gibt mit der Box um das Paraboloid:



$-1 \leq x, y \leq 1 \rightarrow 0 \leq z \leq 2$

easier to calculate than actual intersection with paraboloid (!)

Check if there's an intersection with S_1. If so, calculate actual intersection with Paraboloid, if not, check if there's intersection with S_2 and so on. If there's no intersection with S_1, S_2, S_3, S_4, S_5 then there's no intersection with the paraboloid.

E.g. intersection with S_1 if

$$o_z + t d_z = 0$$

$$t_0 = -\frac{o_z}{d_z}$$

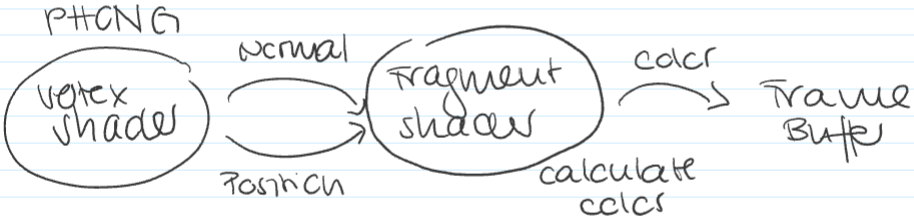
and $-1 \leq o_y + t_0 d_y \leq 1 \wedge -1 \leq o_x + t_0 d_x \leq 1$

(for this t) $= t_0$

(a) Gouraud is a per-vertex color computation. Phong is a per-fragment color-computation.

The vertex shader provides the normal and position data as out variables to the fragment shader. The fragment shader then interpolates these variables and computes the color.

In Gouraud shading, the color for the fragment is computed in the vertex shader, whereas in Phong shading, the color for the fragment is computed in the fragment shader.



(b) The Gouraud shading is computationally less expensive, only requiring the evaluation of the intensity equation at the vertices and bilinear interpolation of these values for each pixel. For phong shading more calculation is required, including the interpolation of the surface normal and the evaluation of the intensity function for each pixel.

4 Triangle Mesh

a) Face set = 9 floats per triangle
Indexed face set: 3 floats per vertex and
3 integers per triangle

b) 

Face set = 9 floats → 36 bytes (9 · 4)
Indexed face set: 9 floats + 3 integers → 48 bytes

5 Transformations and Projections

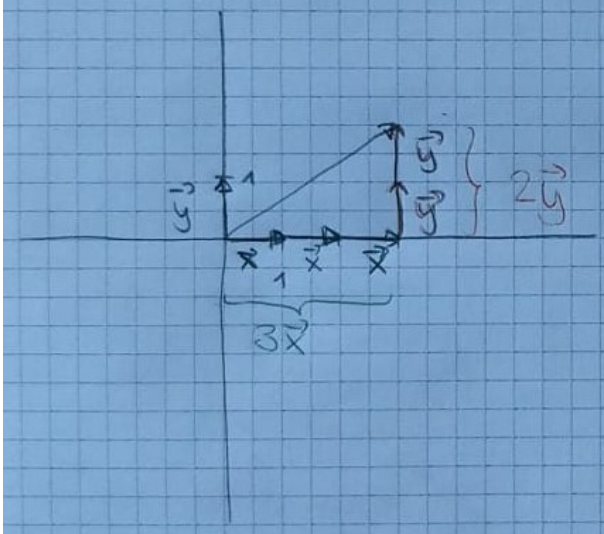
5.1. Transformations

a) coordinate (c.s) system for 2-d plane: 2 vectors
" " " 3-d space: 3 vectors

With one vector we can only describe a one dimensional line, to define a two dim cs we need two vectors.
Analogously with two vectors we can only describe some two dimensional plane and not a coordinate system for a (infinite) space.
(? oder, was steckt mehr "hinter der Frage?)

b) $\vec{v} = a\vec{x} + b\vec{y}$

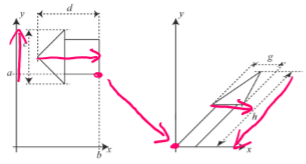
Example: $\vec{v} = \begin{pmatrix} 3 \\ 2 \end{pmatrix} = 3 \cdot \begin{pmatrix} 1 \\ c \end{pmatrix} + 3 \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} (= 3 \cdot \vec{x} + 3\vec{y})$



c) i) „zuerst Rotation mit R, dann Translation mit T" $TR = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_{14} \\ r_{21} & r_{22} & r_{23} & t_{24} \\ r_{31} & r_{32} & r_{33} & t_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}$

ii) $R^{-1} = R^T = \begin{pmatrix} r_{11} & r_{21} & r_{31} & 0 \\ r_{12} & r_{22} & r_{32} & 0 \\ r_{13} & r_{23} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

d)



$\begin{pmatrix} b \\ 0 \\ 1 \end{pmatrix} \to \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$

$\begin{pmatrix} 0 \\ c \\ d \end{pmatrix} \to \begin{pmatrix} g \\ 0 \\ 0 \end{pmatrix}$

$\begin{pmatrix} d \\ 0 \\ 0 \end{pmatrix} \to \begin{pmatrix} -h \\ -h \\ 0 \end{pmatrix}$

$\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \to \begin{pmatrix} g/c \\ 0 \\ 0 \end{pmatrix}$

$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \to \begin{pmatrix} -h/d \\ -h/d \\ 0 \end{pmatrix}$

$\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} -h/d & g/c & t_x \\ -h/d & 0 & t_y \\ 0 & 0 & 1 \end{pmatrix}}_{A} \cdot \begin{pmatrix} b \\ a \\ 1 \end{pmatrix} \Rightarrow$

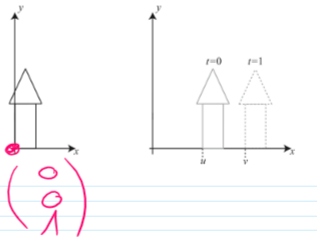
$0 = -hb/d + bg/c + t_x$

$0 = -hb/d + t_y$

$t_x = bg/c + hb/d$

$t_y = bh/c$

e)



$\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$

$A(0) \cdot \begin{pmatrix} 0 \\ y \\ 0 \end{pmatrix} = \begin{pmatrix} u \\ y \\ 1 \end{pmatrix}$

$A(1) \cdot \begin{pmatrix} 0 \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} v \\ y \\ 1 \end{pmatrix}$

$A(t) \cdot \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 0 \end{pmatrix}$

$A(t) := \begin{pmatrix} 1 & 0 & (1-t)u + tv \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

5.2

(a) strahlensatz

$$\frac{y_s}{d} = \frac{y_w}{z_w}$$

$$\tan\theta = \frac{y_s}{d} = \frac{y_w}{z_w} \quad \Rightarrow y_s = \frac{y_w \cdot d}{z_w}$$

(c) NO, because division is impossible

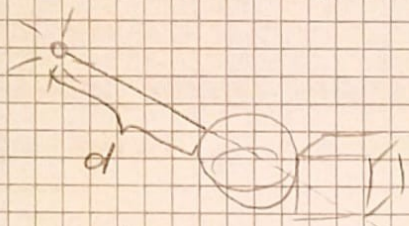
(d) $x_s = \frac{x_w \cdot d}{z_w}$

3D coordinate in
2D homogenous coordinate out

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1/d \end{pmatrix} \cdot \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} = \begin{pmatrix} x_w \\ y_w \\ z_w/d \end{pmatrix} \Rightarrow \begin{bmatrix} x_w \cdot d/z \\ y_w \cdot d/z_w \end{bmatrix}$$

6 Textures & Shadows

(a) Distance of the closest intersection point a ray has with an object when origin at the light source.



d is in the shadow map

to positional lighting

(b) Difference: there's no light source to calculate the distance from it to a point and no frustum
We cannot just render the scene from the light source in the same way we render the scene.
Because with a light source the spotlight behaves like a frustum perspective
and we can use the same perspective projection matrix as the camera in order to render into the shadow map.
For directional light we need an orthographic projection matrix, so the direction of the light is uniform across the map (so as to approximate a very far away light source (sun)). The frustum in this case is built using:

- a view matrix, where the direction is parallel to the light source;
- an orthographic projection (3D→2D) matrix

To get the distance for the shadow map, we take an arbitrary plane - orthogonal to the light and save the distance from a point to this plane, respectively the closest one.
(neg values ⇒ closer than positives)

(c) omnidirectional light: light from every side ⇒ „shadow map for each side"

(e) Generate for each light source a shadow map and then OpenGL will render the scene correctly, according to different lights.
the

7 Fractals & L-Systems

7.1. L-System grammar Expansion

a) pro R jeweils 3 symbole und pro L ebenfalls → exponentielles Wachstum der Länge

exponential $O(k^n)$ →

b) $T_5 = 422$
$X_5 = 243$

$T_6 = 2 \cdot T_5 + 2 \cdot X_5$ — Jedes X wird mit einem neuen Term mit 2T ersetzt
/ Jedes T wird mit 2T ersetzt
/ $(X \rightarrow \underline{T}[+X]\underline{T}[-X]+X)$

$T_7 = 2 \cdot (2T_5 + 2X_5) + \underbrace{2 \cdot X_6}_{=2 \cdot 3 X_5} = 2^2 T_5 + (2^2 + 2 \cdot 3) X_5 = \underline{4T_5 + 10X_5}$

$X_6 = 3X_5$
$X_7 = 3 \cdot X_6 = \underline{3^2 X_5} = 9X_5$

c) System (A) contains an error: you pop more than you push and you cannot pop from an empty stack

7.2 Fractal dimension

a) (It should hold: $d < 2$) ← less than a plane

$\frac{\log(8)}{\log(4)} = \frac{\log(2^3)}{\log(2^2)} = \underline{\frac{3}{2}}$

Seite wird gevierteilt

8 Würfel werden gebraucht

b) (should hold $d > 1$) ← more than a line

$\frac{\log(16)}{\log(4)} = \frac{\log(2^4)}{\log(2^2)} = \frac{4}{2} = \underline{2}$

5 Procedural Modeling

(a) We define frequency according to grid size: grid size limited → frequency limited (=band-limited)

(b) The gradients are not actually random. Just pseudo random.
They are looked up in a hash-map (→ same grid point, same hash value)

(c) double freq → double number of grid points → $f(x,y,z) :=$ perlin_noise$(2x, 2y, 2z)$

(d) Turbulence takes absolute values of noise function.

9 Bézier Curves

The figure below shows the control points of a Bézier curve.

(a) **(1 point)** What is the degree of the curve? 4 (5 points → degree 4)

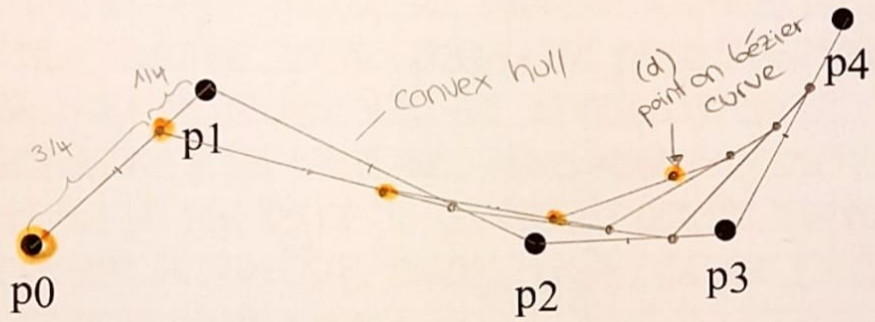
(b) **(2 points)** Sketch the convex hull of the original control points. Explain the convex hull property.

(c) **(1 point)** Explain the symmetry property.

(d) **(1 point)** Sketch the calculation of a point on the curve for the parameter value $t = 0.75$ (approximately) using the Casteljau algorithm.

(e) **(1 point)** The point at $t = 0.75$ splits the curve into two segments. Indicate the new control points of the first segment that goes from $t = 0$ to $t = 0.75$.

(f) **(2 points)** Give pseudo-code for an algorithm to draw the curve as a sequence of line segments using adaptive subdivision. The algorithm should be based on recursive curve splitting.



(b) points p_0, p_1, p_2, p_3, p_4 will span the convex hull of the bezier curve, which means the bezier curve will be „more or less closly below" them. As such, it lends a measure of predictability to the curve

(c) The same bézier curve shape is obtained if the control points are specified in the opposite order. The only difference will be parametric direction of the curve.