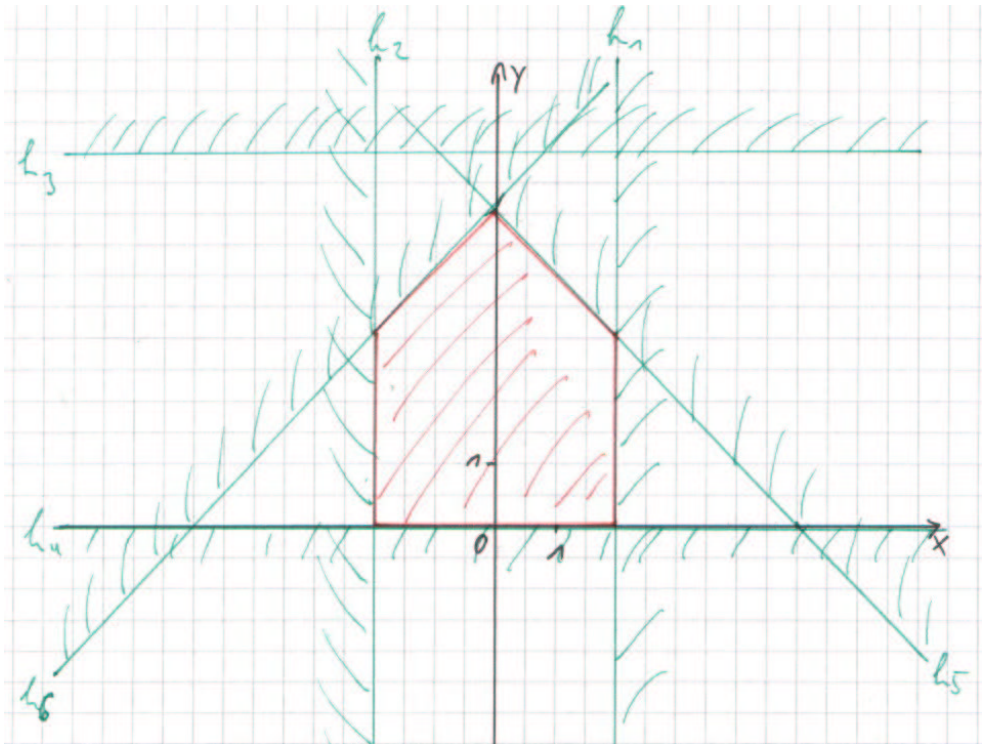
Aufgabe 1: Komposition von Halbebenen

Die Halbebenen h_1 bis h_6 sind in ihren impliziten Darstellungen gegeben:

$$h_1(x,y) = -x + 2 \geq 0$$
$$h_2(x,y) = x + 2 \geq 0$$
$$h_3(x,y) = -y + 6 \geq 0$$
$$h_4(x,y) = y \geq 0$$
$$h_5(x,y) = -x - y + 5 \geq 0$$
$$h_6(x,y) = x - y + 5 \geq 0$$

a) Ich habe die Halbebenen auf einem Blatt Papier gezeichnet und anschließend eingescannt:



Die Halbebenen sind in grün gehalten und verlaufen auf der Seite der jeweiligen Begrenzungslinien, die *nicht* grün schraffiert ist. Zur Verdeutlichung: h_1 umfasst alle Punkte, die links von der Linie liegen (und direkt auf der Linie), die die x-Achse bei $x = 2$ schneidet.

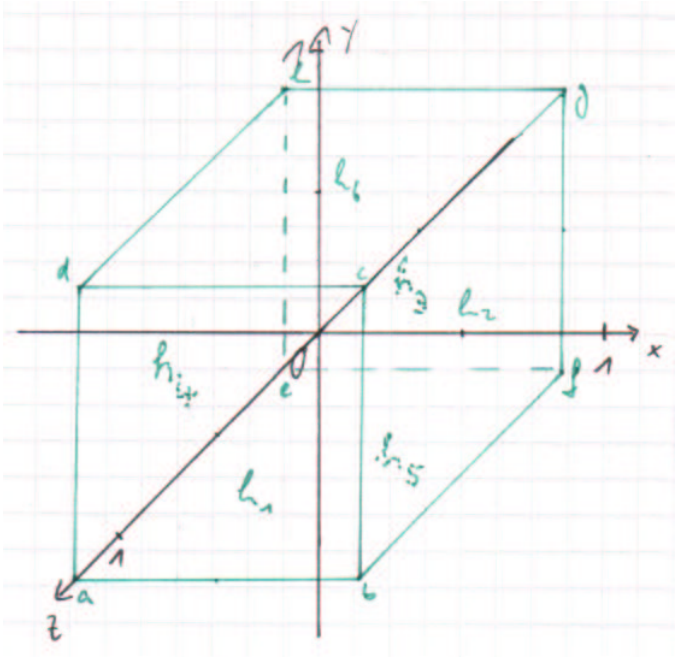
Als Schnittfläche aller Halbebenen entsteht ein rot schraffiertes Fünfeck, dessen spezielle Form als “Haus des Nikolaus” bekannt ist.

b) Die Kombination der Halbebenen ist *nicht* optimal. Im Scan erkennt man das daran, dass h_3 die Form des Polygons in keiner Weise verändert. Mathematisch kann eine Kombination von Halbebenen nur dann als optimal bezeichnet werden, wenn man jede Halbebene ein $\exists p \notin h_i$ findet, für das $\forall h_j, j \neq i : p \in h_j$ gilt, d.h. es für jede Halbebene min. 1 Punkt gibt, der nicht in ihr, aber in allen anderen Halbebenen liegt.
Die Figur lässt sich demzufolge auch unter Beschränkung auf die Halbebenen h_1, h_2, h_4, h_5 und h_6 darstellen.

Aufgabe 2: Konstruktion dreidimensionaler Objekte durch Halbräume

a) Zuerst zeichne ich die Objekte auf, danach bestimme ich die Halbräume.

Einheitswürfel:



Die Ebenen h_1 bis h_6 werden durch die Lage der Punkte *a* bis *h* definiert:

$a,b,c,d \in h_1;\ b,f,g,c \in h_2;\ e,f,g,h \in h_3;\ a,e,h,d \in h_4;\ a,b,e,f \in h_5;\ d,c,g,h \in h_6$

Die entsprechenden impliziten Darstellungen sind mit der Hesseschen Normalform der Ebenengleichung sehr schnell zu ermitteln, da die Normalenvektoren aller Ebenen jeweils entlang einer Achse verlaufen:

$$h_1(x,y,z) = -z + \frac{1}{2} \geq 0$$

$$h_2(x,y,z) = -x + \frac{1}{2} \geq 0$$

$$h_3(x,y,z) = z + \frac{1}{2} \geq 0$$

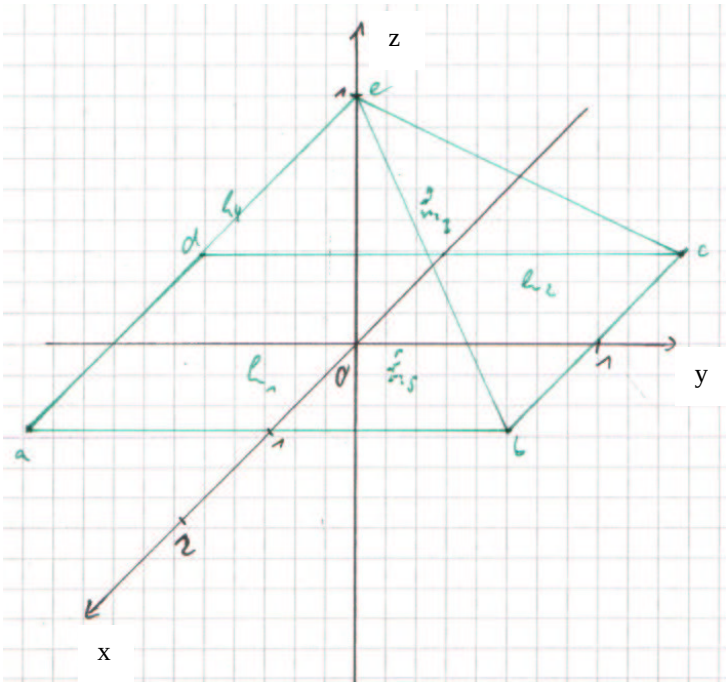
$$h_4(x,y,z) = x + \frac{1}{2} \geq 0$$

$$h_5(x,y,z) = y + \frac{1}{2} \geq 0$$

$$h_6(x,y,z) = -y + \frac{1}{2} \geq 0$$

Pyramide:

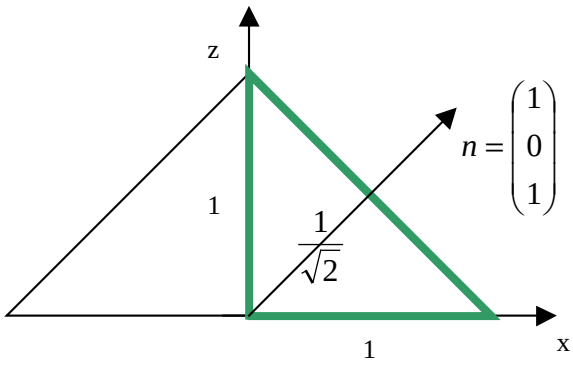
Leider habe ich mich bei der Erstellung der Skizze etwas mit der Achsenbezeichnung vertan, so dass ich per Computer nachhelfen musste:



Auch hier bestimmen die Punkte *a* bis *e* die Halbebenen h_1 bis h_5:

$$a,b,e \in h_1;\ b,c,e \in h_2;\ c,d,e \in h_3;\ a,d,e \in h_4;\ a,b,c,d \in h_5$$

Die Umschreibung mit Hilfe impliziter Funktionen ergibt sich erneut aus der Hesseschen Normalform. Diesmal sind aber nicht alle Normalen achsenparallel. Eine kleine Skizze veranschaulicht den Sachverhalt für h_1. Der Normalenvektor hat (nicht normiert) die Länge $\sqrt{2}$. Da sich in Kombination mit dem Abstand der Hypotenuse im Dreieck (entspricht einer Seite der Pyramide) von $\frac{1}{\sqrt{2}}$ sich aber unpraktische Werte ergeben, habe ich beide Seiten der impliziten Gleichungen jeweils mit $\sqrt{2}$ multipliziert.

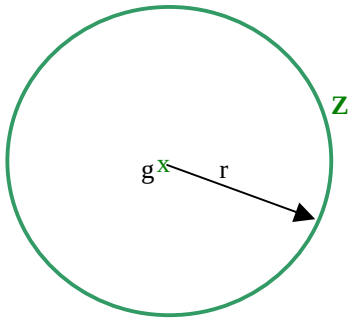


$$h_1(x,y,z) = -x - z + 1 \geq 0$$
$$h_2(x,y,z) = -y - z + 1 \geq 0$$
$$h_3(x,y,z) = x - z + 1 \geq 0$$
$$h_4(x,y,z) = y - z + 1 \geq 0$$
$$h_5(x,y,z) = z \geq 0$$

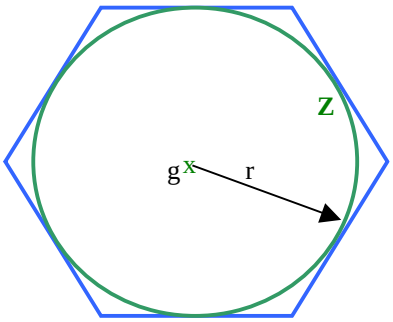
b) Die Hülle eines Zylinders *Z* besteht aus der Menge von Punkten *p*, die zu der Geraden *g*, die durch die Mitte des Zylinders verläuft, einen Abstand haben, der dem Radius *r* entspricht:

$$g(s) = m + sn,\ m,n \in R^3, s \in R$$
$$Z = \{p \in R^3 \mid d(p,g) = r\}$$

Um diesen Zylinder durch Halbebenen zu approximieren, nutze ich die vorherige Überlegung, indem ich eine hinreichend große Anzahl von Halbebenen konstruiere, die alle den Abstand *r* von *g* haben. Idealerweise sind diese Ebenen regelmäßig angeordnet, um eine gute Approximation zu erreichen. Blickt man entlang *g*, so erhält man folgendes Bild:



Mein Verfahren zur Annäherung durch (bspw. 6) Halbebenen ergibt bei gleichem Sichtwinkel:



Es fällt auf, dass der Zylinder stets kleiner als die approximierende blaue Hülle ist. Man könnte den Abstand der Ebenen zu *g* verringern, um so eine bessere Annäherung zu erzielen, allerdings erfordert dies mehr Rechenaufwand, so dass ich darauf vorerst verzichte.
Je mehr Halbebenen man verwendet, desto genauer nähert sich die Hülle dem Zylinder an. Hier ist eine Abwägung zwischen Aufwand und Nutzen zu treffen, da die Darstellung i.d.R. auf einem diskreten Raster erfolgt und Subpixelpräzision nicht unbedingt notwendig ist.

Ich habe bisher offen gelassen, wie man die erwähnten Ebenen bestimmt. Zunächst ermittelt man den Normalenvektor, der senkrecht auf dem Richtungsvektor von *g* steht:

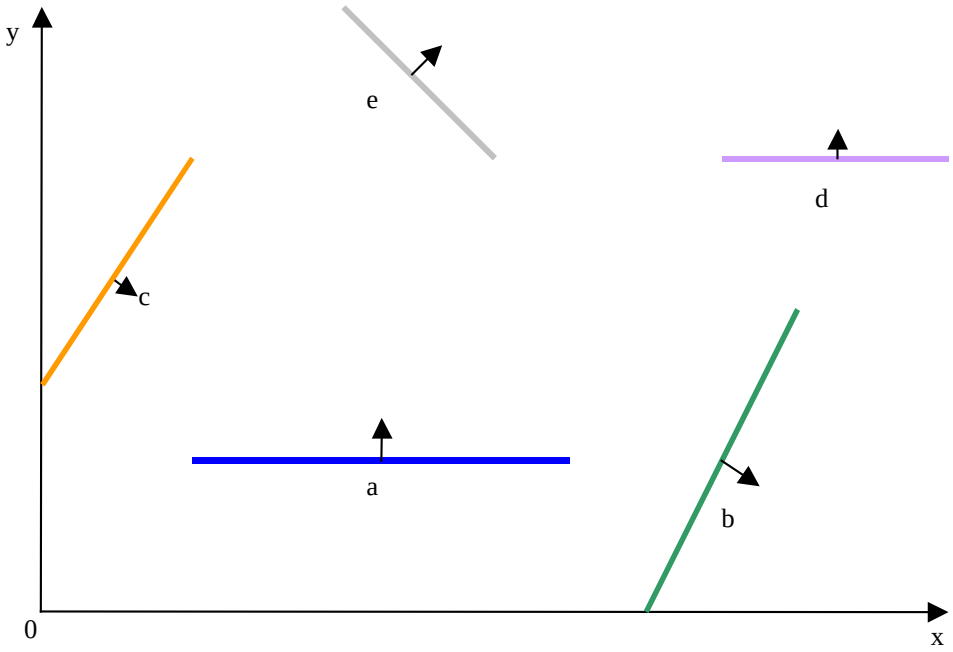
$$n_{Ebene} \bullet n_{Gerade} = 0$$

Je nachdem, wie gut die Näherung sein soll, kann man entweder n_{Ebene} eher zufällig oder systematisch bestimmen. Letzteres erfordert, dass man von einem ersten n_{Ebene} ausgehend alle in einer Ebene liegenden weiteren n_{Ebene} durch einen Winkel α ermittelt, der von 0 bis 2π läuft.

Wenn die Normale n_{Ebene} gefunden wurde, muss man noch einen Punkt in der Ebene bestimmen. Der ergibt sich, wenn man zu einem Punkt der Gerade *g* noch $n_{Ebene} \cdot$ Radius addiert.

Anmerkung: Ich gehe stets davon aus, dass der Zylinder unendlich lang ist. Sollte dies nicht der Fall sein, d.h. Deckel- und Grundfläche existieren, so sind entsprechend noch zwei Halbebenen hinzuzufügen, die diese beiden Begrenzungen herstellen.

Aufgabe 3: Konstruktion eines BSP-Trees

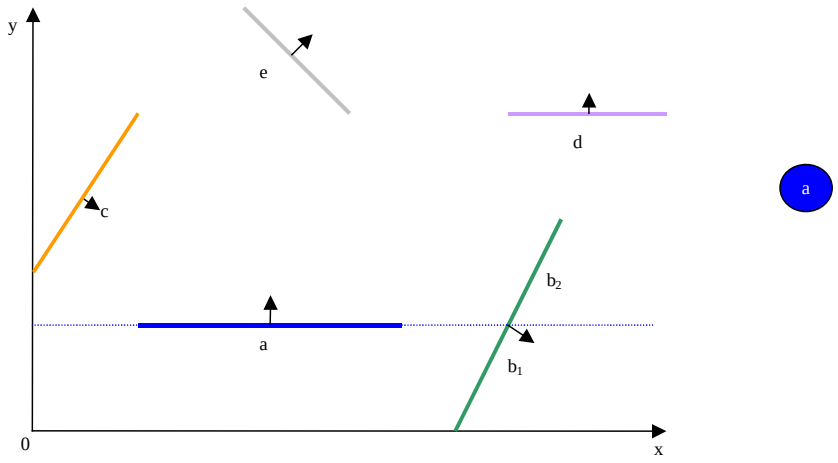


Zuerst habe ich alle Objekte im zweidimensionalen Raum aufgezeichnet:

a: (2, 2) – (7, 2)
b: (8, 0) – (10, 4)
c: (0, 3) – (2, 6)
d: (9, 6) – (12, 6)
e: (4, 8) – (6, 6)

Die Vorderseite der Linie wurde durch kleine Pfeile symbolisiert.
Wenn man BSP-Trees interaktiv ausprobieren möchte, so empfehle ich, sich ein Java-Applet im Internet anzuschauen: http://symbolcraft.com/pjl/graphics/bsp/bsptreedemo_german.html

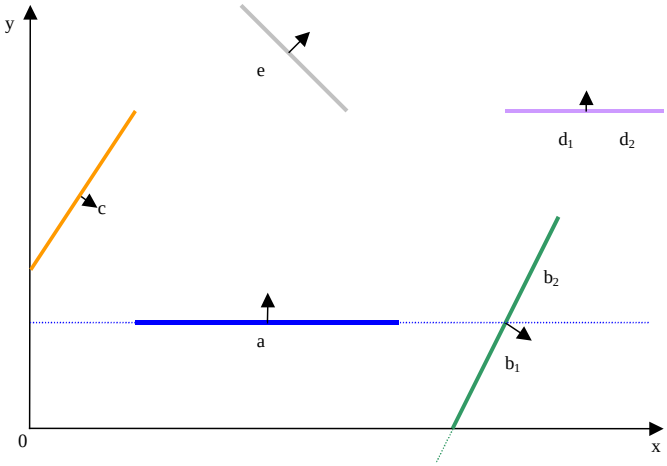
a) Wenn man die Linien alphabetisch durchwandert, um so einen BSP-Tree zu erstellen, dann entstehen die folgenden Schritte:

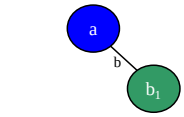


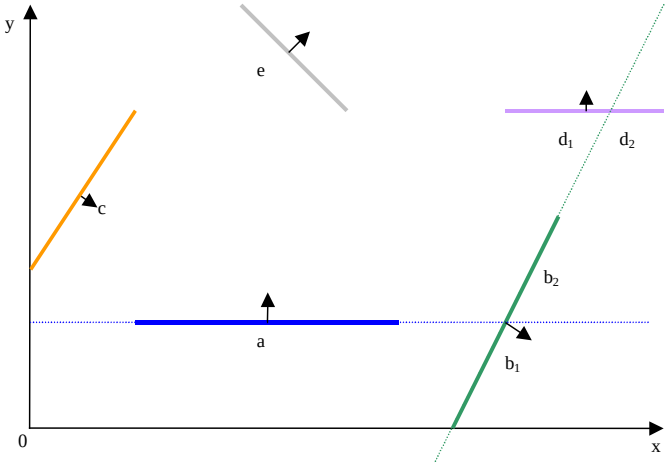
Schon gleich zu Beginn wird *b* durch die durch *a* verlaufende Halbebene geschnitten, so dass *b* durch b_1 und b_2 ersetzt wird.

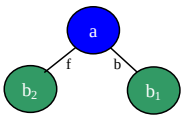
Anmerkung: Im BSP-Baum wird links von einem Knoten die Vorderseite (f für front) und rechts die Rückseite (b für back) eingetragen.

Durch den Schnitt von *b* mit *a* entstanden aus einer Linie zwei neue. Zunächst wird b_1 betrachtet, danach b_2:

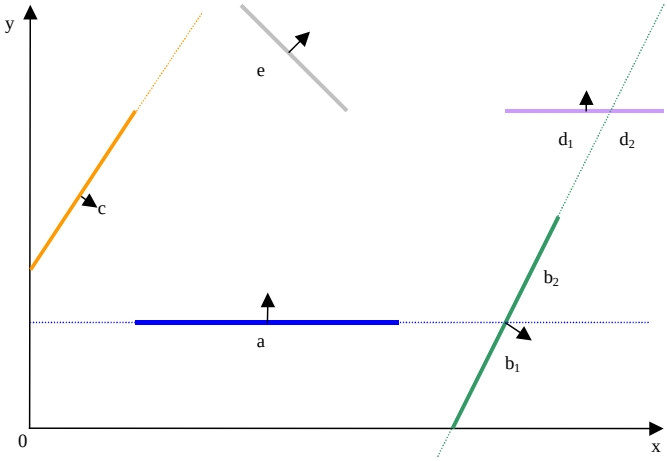


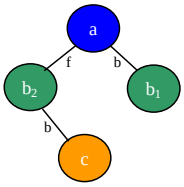




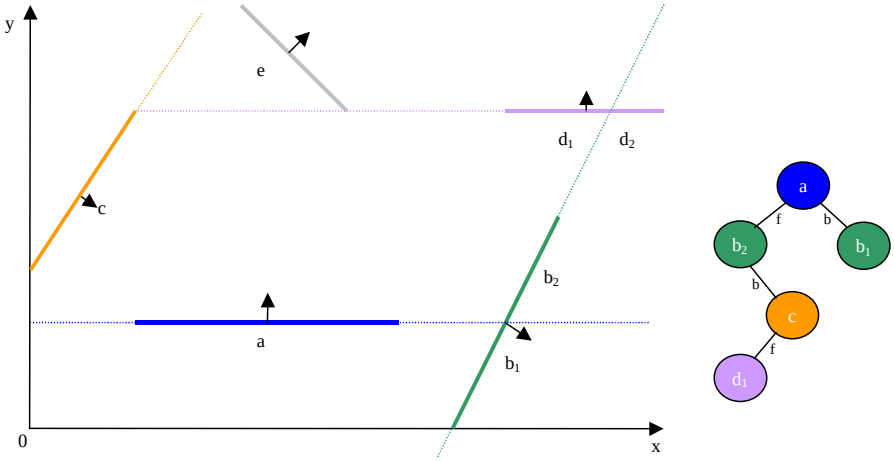


Die Linie *c* fügt sich unproblematisch ein:

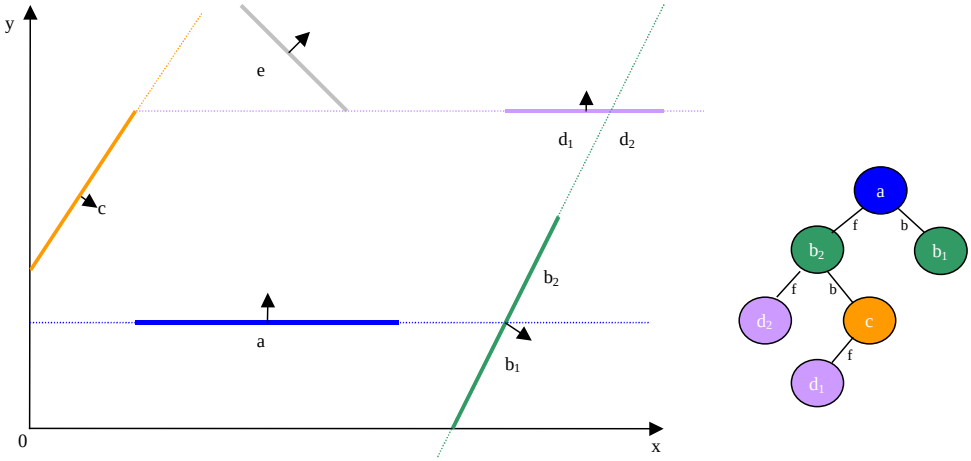




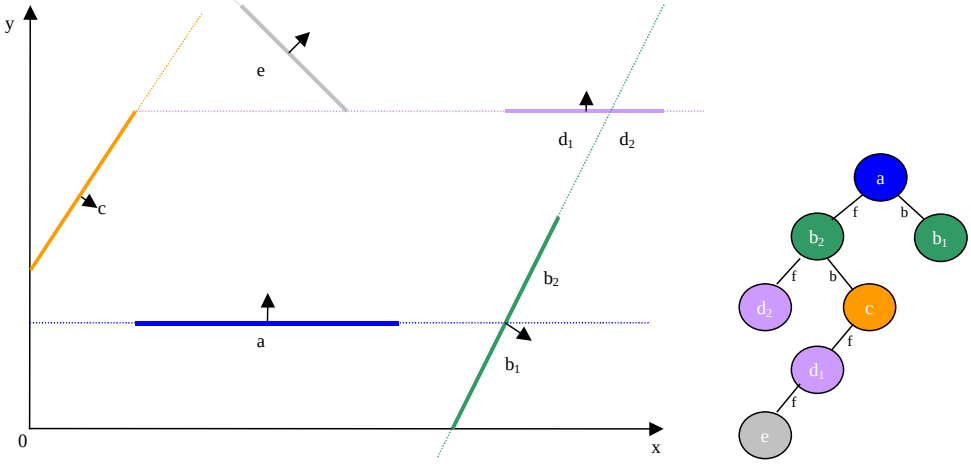
Bei *d* ist wieder zu beachten, dass durch b_2 eine Partitionierung in d_1 und d_2 entstand:



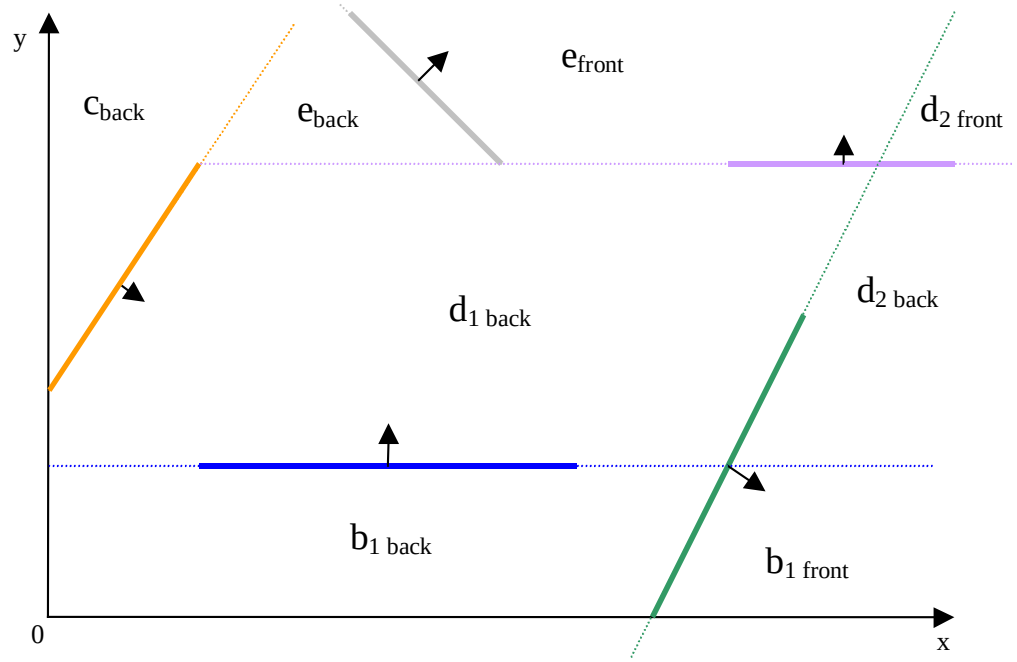
d_1 verläuft zwar durch die Endpunkte von *c* und *e*, allerdings wird dadurch keine Unterteilung dieser Linien notwendig. Die den Raum teilende gestrichelte Linie von d_1 ist ferner bis *c* begrenzt und ist nicht unendlich lang, da d_1 nur den Bereich zwischen der Vorderseite von *c* und der Rückseite von b_2 partitionieren darf. Ein weitere Feinheit ist die Einordnung von d_2 in den Baum: Man muss dieses Liniensegment unterhalb b_2 und nicht bei *c* einordnen.



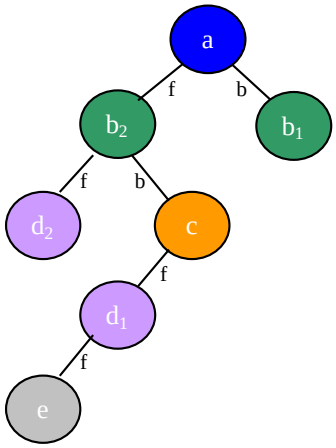
Ohne weitere Überlegung ist auch *e* sofort einordbar:



Abschließend möchte ich noch einmal die entstandene Unterteilung des Raumes verdeutlichen, indem ich den einzelnen Teilflächen die entsprechende Position im BSP-Baum zuordne:

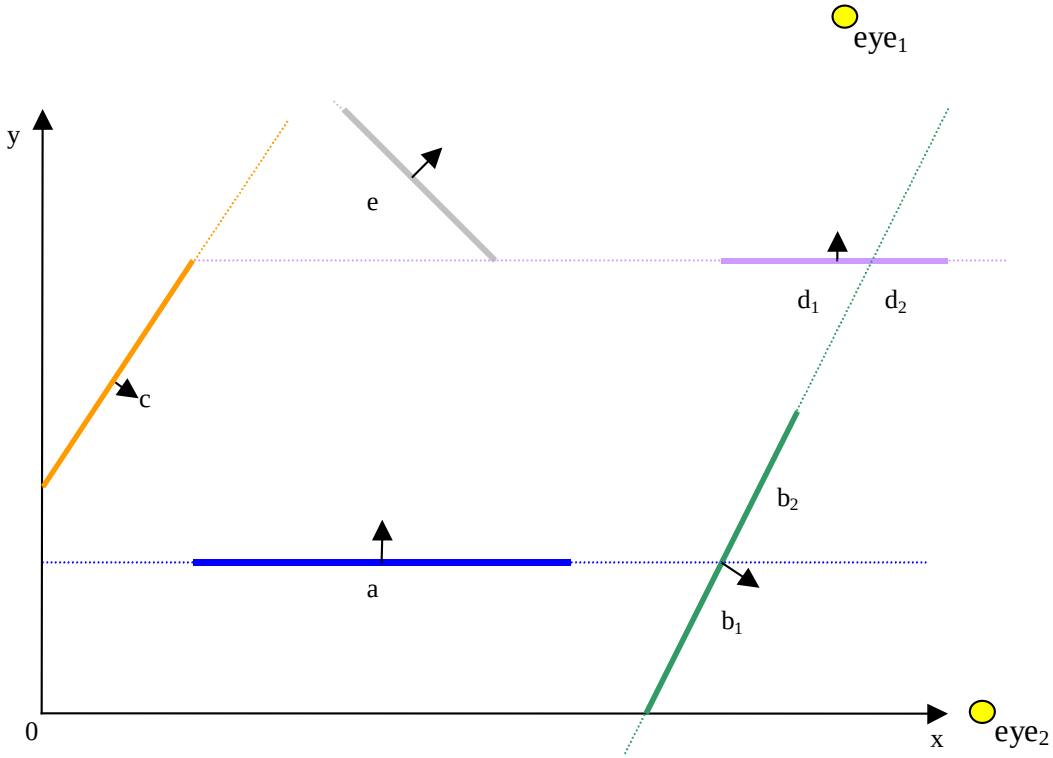


Für die Einordnung ist entscheidend, welche Linie den jeweiligen Teilraum zuletzt partitioniert hat. Zum Vergleich wiederhole ich den BSP-Tree:



Zur Überprüfung kann man untersuchen, ob jedesmal, wenn im Baum ein Knoten weniger als zwei Unterknoten hat, der fehlende Teil als Teilraum in der obigen Grafik auftaucht. Z.B. hat *c* nur einen Unterknoten (namens d_1), der auf der Vorderseite von *c* liegt. Die Rückseite von *c* steht als c_{back} in der Grafik.

b) Die beiden Kameras eye_1 und eye_2 füge ich zunächst als gelbe Punkte in die Grafik ein:
eye_1: (10, 10)
eye_2: (12, 0)



Durch Einordnung in die jeweiligen Teilräume findet man sehr schnell die notwendige Traversierungsreihenfolge des BSP-Trees:

$eye_1 \in e_{front}$: $b_1 - a - d_2 - b_2 - c - d_1 - e$
$eye_2 \in b_{1\,front}$: $c - e - d_1 - d_2 - b_2 - a - b_1$