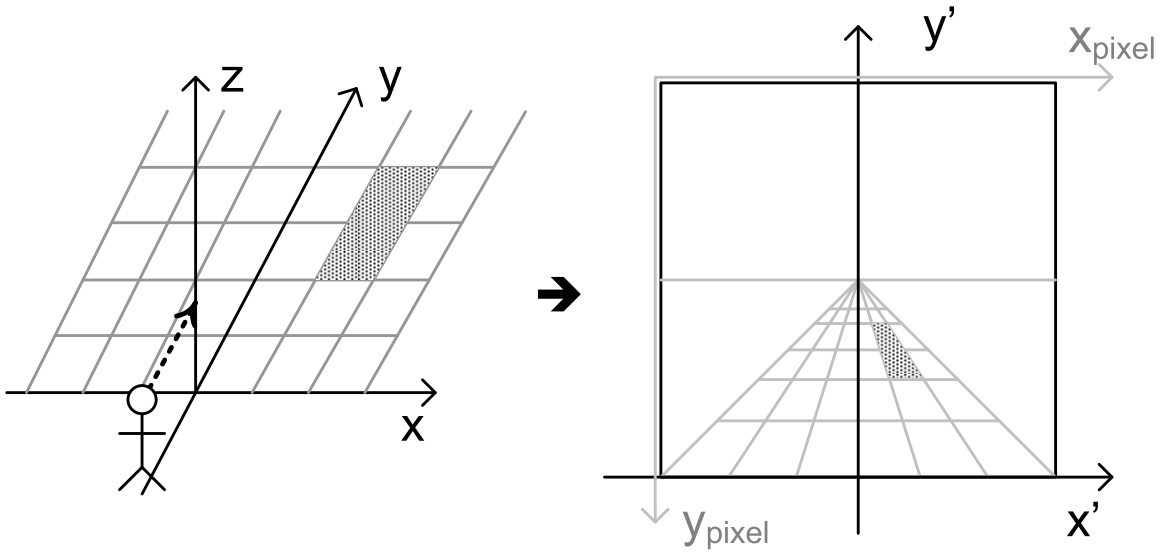
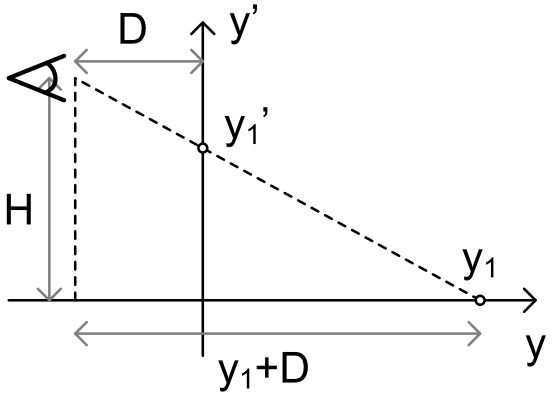
Volgende figuur legt het principe van een perspectieftransformatie uit:



Het lijkt misschien vrij ingewikkeld, maar we zullen helpen met de transformaties. In volgende formules onderstellen we dat we naar het scherm kijken vanop een afstand D en een hoogte H. We zullen eerst de transformatie (x,y,z) → (x',y') berekenen, waarna we dan simpel (x',y') → (x_{pixel}, y_{pixel}) berekenen.

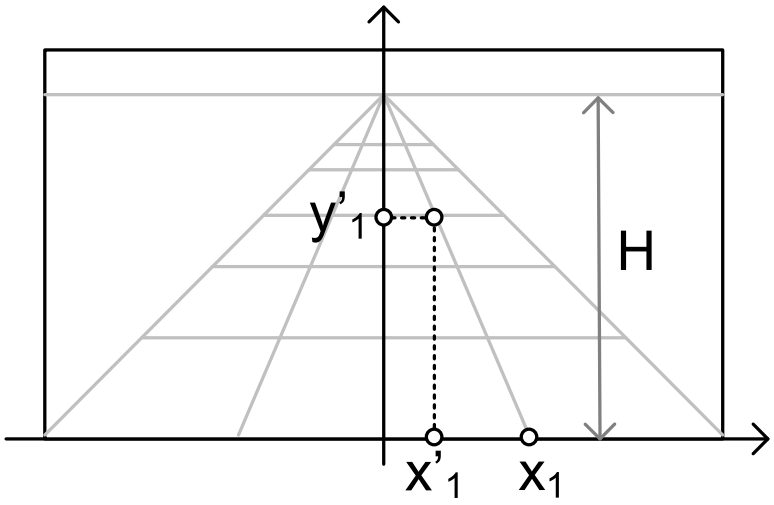
Ten eerste de y-as (stel even dat z = 0):



De y'-as is hier het scherm, en het oog is "ons" oog waarmee we naar het scherm kijken. Met de hoogte en afstand van het oog gekend, zien we dat er twee gelijkvormige driehoeken onstaan. De verhouding van y_1 tot y'_1 zal dus gelijk zijn aan de verhouding van y_1+D tot H. Met andere woorden:

$$\frac{y'_1}{y_1} = \frac{H}{y_1 + D}$$

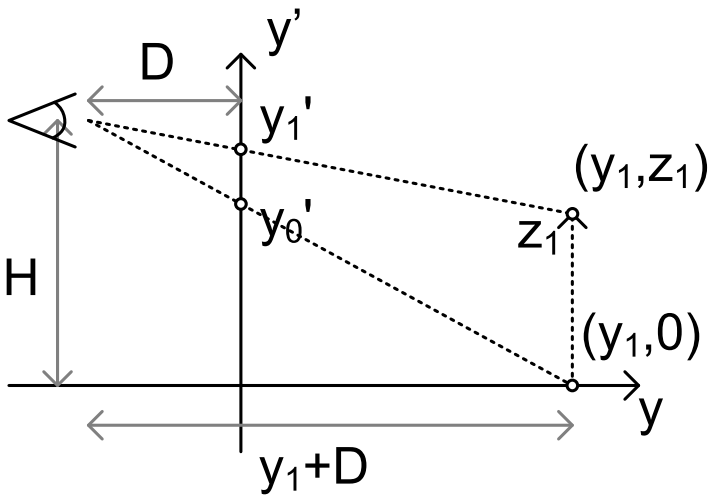
Voor de x-coordinaten kunnen we een gelijkaardige redenering opstellen:



We zien dat de x-as onderaan beeld hetzelfde ligt als de x'-as, maar dat de x-richting steeds smaller wordt hoe hoger we gaan. Uit de vorige stap kennen we al de y' coordinaat van het punt en weten we dat de horizon op een hoogte H ligt. Ook hier krijgen we twee gelijkvormige driehoeken. Er zal dus gelden dat:

$$\frac{x_1}{H} = \frac{x_1 - x'_1}{y'_1}$$

Met deze redenering kunnen we dus (x',y') berekenen gegeven een (x,y,z) met z = 0. Maar wat als z nu verschillend is van nul? Laten we eens kijken naar wat er dan met onze eerste figuur gebeurt:



We zien dat het originele punt (x_1,y_1,0) omhoog wordt geschoven met een waarde z_1, en dat hierdoor ook de waarde y'_1 verschuift. We zien weer een evenredigheid in de figuur als volgt:

$$\frac{y'_1 - y'_0}{z_1} = \frac{D}{y_1 + D}$$

Uit deze vergelijking kunnen we dan y'_1 halen in functie van y_1, z_1 en y'_0.
Er zal dus voor (x,y,z) → (x',y') volgen dat:

$$y'_0 = \frac{y}{D+y} \cdot H$$

$$\begin{cases} x' = \dfrac{H - y'_0}{H} \cdot x \\ y' = y'_0 + \dfrac{D}{D+y} \cdot z \end{cases}$$

Hier is y'_0 dus een hulpcoordinaat die het makkelijker maakt x' en y' te berekenen. Voor z = 0 geldt trouwens dat y' = y'_0. Deze x' en y' kunnen we nu simpel aanpassen naar het coordinatensysteem van het scherm:

$$\begin{cases} x_{pixel} = \dfrac{SCHERMBREEDTE}{2} + x' \\ y_{pixel} = SCHERMHOOGTE - y' \end{cases}$$

Tip: het is misschien handig een procedure te maken als volgt:

```
TYPE Pixel = RECORD
             (* represents a point on the screen *)
              x,y : CARDINAL;
             END;
     Point3D = RECORD
              (* represents a point in the game space *)
              x,y,z : ... ;
             END;

PROCEDURE Point3DToPixel(p:Point3D):Pixel;
...
```

Zelf te berekenen: wat doe je als de originele (x,y,z)-ruimte ook nog eens kan draaien?