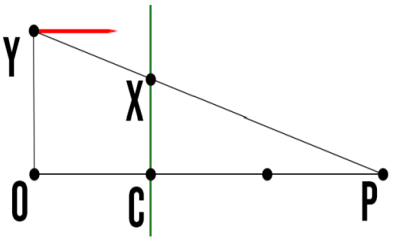
Comment calculer une grille en 3 dimensions ou, comment calculer la perspective centrale informatiquement

Durant le XV^{e} siècle (Renaissance italienne) les peintres italiens développèrent des méthodes, des techniques afin de transposer un point d'un espace en 3 dimensions (tel qu'une cour, un bâtiment, un jardin) et ces coordonnées 3D (vector3 float) en coordonnées 2D (vector2 float). Pour cela, ils s'imaginèrent (leurs yeux) dans l'espace 3D pointant une direction. Mais également une toile de peintre (canvas), plan traversé par les segments reliant le point et les yeux du peintre.

On considère **O** comme l'origine, **Y** comme les yeux du peintre pointant en (1, 0, 0), P comme le point visé, C en tant que canvas, et, **X** comme le point symbolisé par l'intersection du segment **YP** et du plan (canvas). **C** est le point correspondant à la position du plan (canvas).



On en déduit une situation de Thalès. Par conséquent :

$$\frac{YO}{XC}=\frac{OP}{CP}=\frac{PX}{PY}$$ (théorème de Thalès)

Donc :

$$XC=\frac{YO\times CP}{OP} \quad = \quad Y_{onScreen}=\frac{Y_{camera}\times X_{canvas}}{X_{point}}$$ (règle de trois)

ATTENTION : LES NOMS

Les yeux sont ici remplacés par l'objet **Camera**, la toile par le **Canvas,** le point par **Point** et enfin le point d'intersection du canvas et de YP par **onScreen**

XC correspondant à la position **Y** (ordonnée) du point transposé sur notre écran. Nous obtenons donc la position en Y de notre point transposé à l'écran. En procédant de la même manière pour les coordonnées Y d'un point, on obtient :

$$Y_{onScreen}=\frac{Z_{point}\times X_{canvas}}{X_{canvas}}$$

Cependant un problème survient. Les résultats des opérations sont bien trop petits pour pouvoir être utilisé sur un ordinateur, en effet, la plupart des résultats ne seront guère supérieurs à 1, il est donc préférable / nécessaire de le multiplier par 100 ou tout autre multiplicateur correspondant à l'usage. En plus de cela, les points représentés seront inversé, effectivement en informatique le comptage des coordonnées ce fait de haut en bas, c'est pour cela que nous soustrairont à la taille de notre fenêtre (W_{frame} pour la largeur de notre fenêtre : de l'anglais Width et, H_{frame} pour la hauteur de notre fenêtre : de l'anglais Height) le résultat précédant (multiplié par 100 ou tout autre multiplicateur correspondant à l'usage). Enfin, nous ajouterons à X_{camera} X_{canvas} effectivement, le canvas se veut parenté/attaché aux yeux du peintre : ici notre camera.

Donc :

$$X_{onScreen}=W_{frame}-\frac{Z_{point}\times(X_{camera}+X_{canvas})}{X_{camera}+X_{canvas}}\times 100$$

$$Y_{onScreen}=H_{frame}-\frac{Y_{camera}\times(X_{camera}+X_{canvas})}{X_{point}}\times 100$$

Exemple avec les grandeurs du document 1 et une fenêtre de 500x500 :

$$Y_{onScreen}=500-\frac{1\times(0+1)}{3}\times 100=66{,}6666666\ldots \qquad X_{onScreen}=500-\frac{1\times(0+1)}{0+1}\times 100=400$$

DONC : onScreen = [400, 66.66666]

Exemple d'application de la méthode par un processus informatique avec Python 3.6.4

```
import tkinter as tk

fWidth, fHeight = 500, 500

lignCountH = 25
lignCountV = 100

camPos = [0, 1, 0]

canvasPosX = 2

lastLignPosY = 0

frame = tk.Tk()
frame.title("3D engine")

c = tk.Canvas(frame, width = fWidth, height = fHeight, background = "black")

c.delete("all")
for h in range(0, lignCountH) : # Drawing horizontal ligns
        pointOnScreenY = fHeight - int(((camPos[1] * ((h + 1) - canvasPosX)) / (h + 1)) * 100)

        if h == 0 :
                c.create_line(0, pointOnScreenY, fWidth, pointOnScreenY, fill = "blue")
        else :
                c.create_line(0, pointOnScreenY, fWidth, pointOnScreenY, fill = "white")

        if h == (lignCountH - 1) :
                lastLignPosY = pointOnScreenY
                print("")
                print("[DEBUG] : last lign Y =", lastLignPosY)

print("")

for v in range(0, int(lignCountV / 2)) : # Drawing 'vertical' ligns (Right)
        x1 = int(((canvasPosX * (v + 1)) / lignCountV) * 100)
        x2 = int(((canvasPosX * (v + 1)) / 1) * 100)

        x1 += ( fHeight / 2)
        x2 += ( fHeight / 2)

        c.create_line(x1, lastLignPosY, x2, fHeight, fill = "white")

for v in range(0, int(lignCountV / 2)) :  # Drawing 'vertical' ligns (Left)
        x1 = int(((canvasPosX * (v + 1)) / lignCountV) * 100)
        x2 = int(((canvasPosX * (v + 1)) / 1) * 100)

        x1 = -x1 + ( fHeight / 2)
        x2 = -x2 + ( fHeight / 2)

        c.create_line(x1, lastLignPosY, x2, fHeight, fill = "white")

c.create_line(fWidth / 2, lastLignPosY, fWidth / 2, fHeight, fill = "red")

frame.mainloop()
```