- Projet de 2^{nde} année -
Compte rendu de la réunion du 08/02

Ce document récapitule l'ensemble des informations qui ont été dites à l'issue des 2 dernières réunions, et l'évolution du projet.

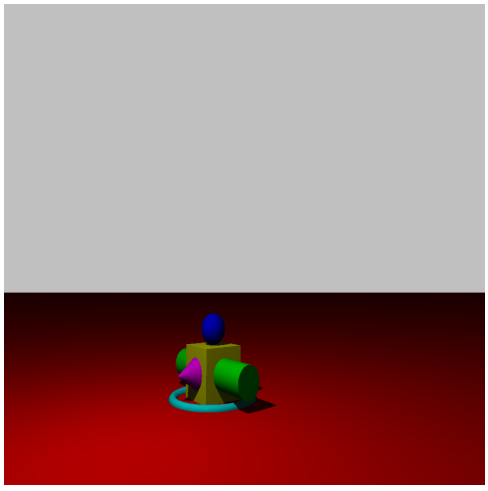
1 Récapitulatif de la réunion

- La réunion a commencé avec un récapitulatif des erreurs résolues :

  ↪ le problèle de *parsing* qui résultait à l'exécution a été réglé (on lance l'éxécutable depuis le dossier unix de povray-master). Cependant, le problème de parsing venait d'un oubli malencontreux de slash dans la redirection,

  ↪ le problème d'enregistrement de l'image a également été résolu : quand bien même elle ne reste affichée que quelques dizaines de millisecondes à l'éxécution (c'est normal à priori), l'image s'enregistre dans le dossier unix de povray-master,

  ↪ enfin, quelques modifications dans le fichier demo.pov ont permis une visualisation correcte de la scène 3D : ce sera notre image de référence (ensemble de formes géométriques imbriquées posé sur un plan rouge sur fond gris).



- Florent a ensuite exposé la stratégie et les calculs que l'on va adopter concernant la matrice de lentilles minces (modèles sténopés). Il faut prendre en compte la potentielle rotation de la matrice de lentilles. Nous fixons pour l'instant arbitrairement l'angle d'ouverture de chaque lentille en dur dans le code (une fois que les paramètres seront fixés dans le fichier de config, il n'y aura plus besoin de recompiler le code).

- Depuis la semaine dernière, les conseils de M. TILMANT et les vôtres nous ont poussé à nettoyer le code, ie à repartir sur des bases saines. Tout le code qui avait été ajouté a été mis en commentaire. Le modèle de caméra à une lentille convergente a été implémneté une nouvelle fois. Nous avons malheureusement cru que **ray.Origin** était le vecteur contenant les coordonnées du pixel de départ sur l'écran : ce n'est pas le cas sur ce modèle (marchait avec le sténopé au vu de son fonctionnement). Il suffira donc de changer l'initialisation du **ray.Origin**. Nous avons retesté et des résultats s'avèrent plus cohérents que ceux obtenus avant mais restent faux (la modification de la distance focale n'a que trop peu d'influence sur l'image résultat).

  Image avec une focale de 50 unités à gauche et 5 unités à droite (l'unité de f pour POVRAY reste à trouver) :

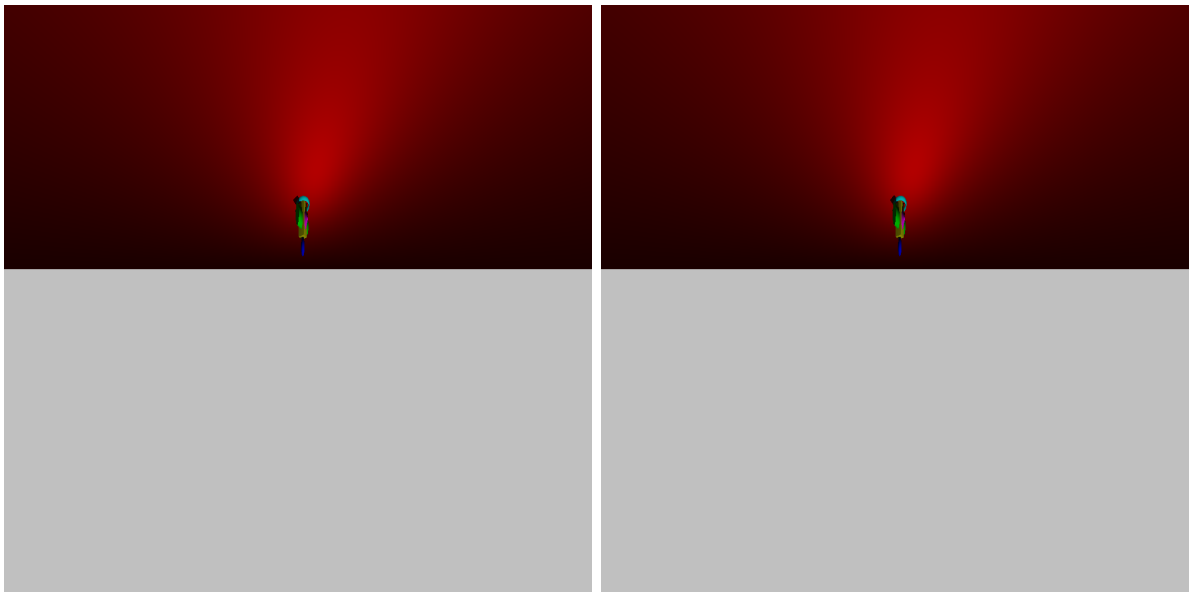
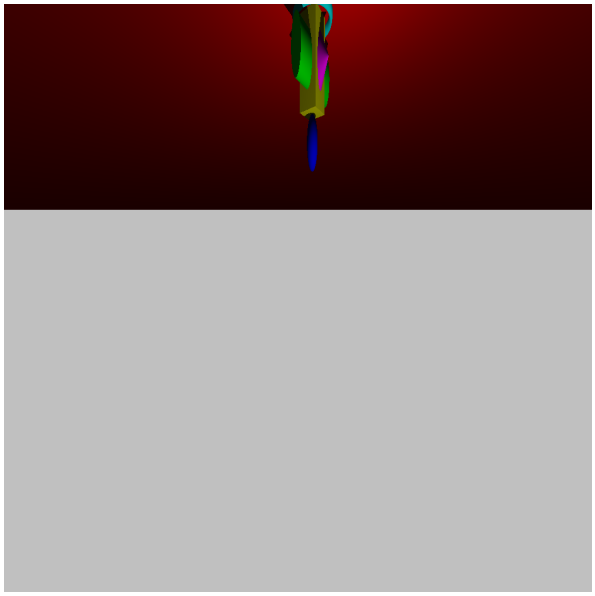


  Image avec une focale de 0.005 unités :

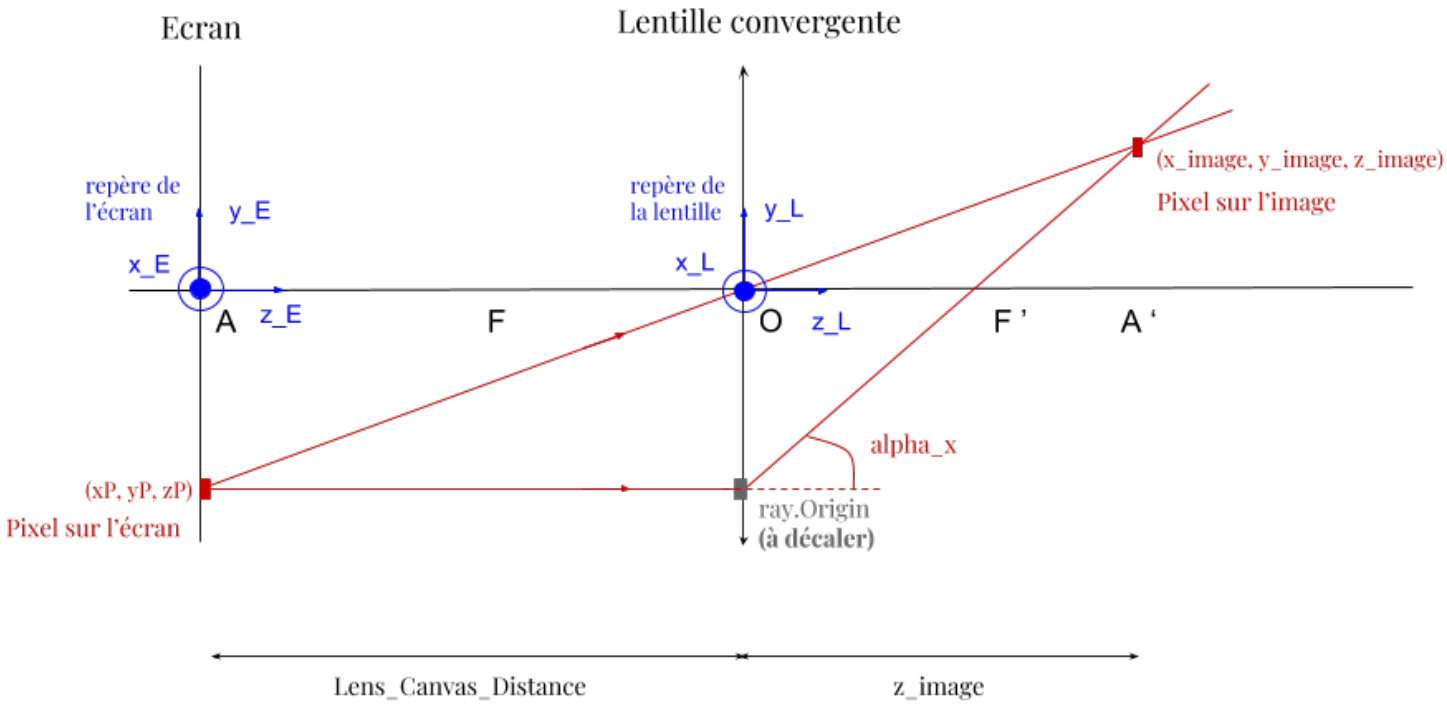


De plus, l'image semble un peu trop aplatie en largeur mais on peut constater qu'elle est retournée (cohérent au vu du modèle implémenté), ce qui n'était pas le cas avant

2 Mise en équation des rayons et stratégies

2.1 Modèle à une lentille convergente

Rappel du modèle d'une lentille convergente et calculs effectués ensemble la semaine dernière :



On cherche les coordonnées du pixel visualisé sur l'image à partir des coordonnées du pixel sur l'écran et de la distance focale.

On a, d'après la relation de conjugaison : $\dfrac{1}{\overline{OA'}} - \dfrac{1}{\overline{OA}} = \dfrac{1}{f} \Leftrightarrow \overline{OA'} = \dfrac{f \times \overline{OA}}{f + \overline{OA}}$

(on raisonne avec les grandeurs algébriques).

D'où, avec les notations du code : $z_{image} = \dfrac{f \times y_P}{f + y_P}$

où y_P reste à déterminer en fonction de **ray.Origin**

On peut maintenant déterminer x_{image} et y_{image} en fonction de z_{image} :

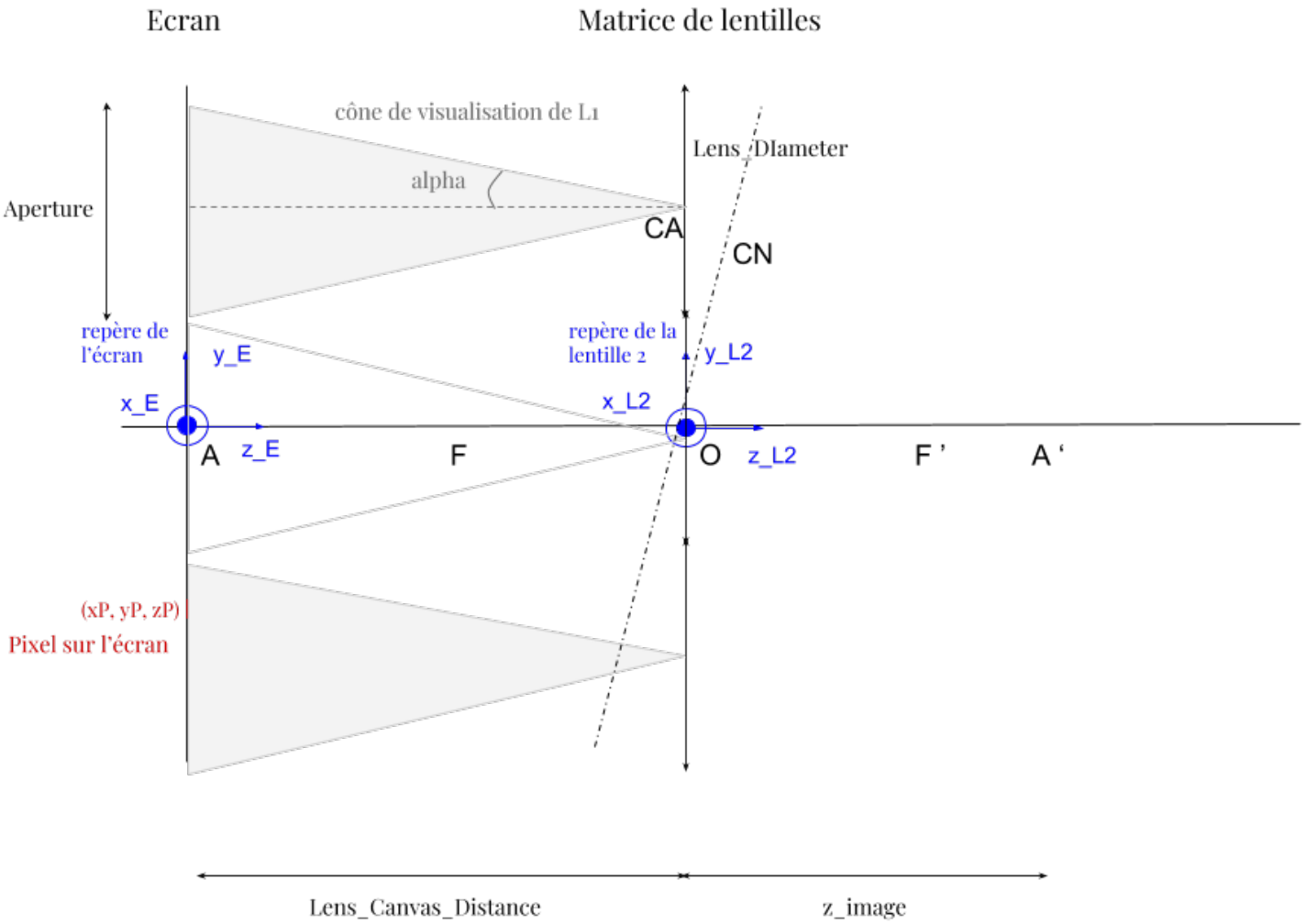
$$\tan(\alpha_x) = \frac{y_{image} + y_p}{z_{image}} \Leftrightarrow y_{image} = \tan(\alpha_x) \times z_{image} - y_p$$

On détermine les angles α_x et α_y de la sorte :

$$\alpha_x = \arctan(\frac{y_P}{f}) \text{ et } \alpha_y = \arctan(\frac{x_P}{f})$$

2.2 Modèle à une matrice de lentilles

Modèle adopté pour le calcul de la direction d'un rayon dans une matrice de lentilles (chacune sténopé) :



Etape 1 : calcul de la position du centre de la lentille

On prendra $lens\_diameter = d$ et φ correspond à l'angle que sépare le plan droit des matrices et le plan incliné. D'où : $y_{CA} = d + \frac{1}{2} \times d$

Il vient : $\sin(90 - \varphi) = \dfrac{y_{CN}}{y_{CA}} \Leftrightarrow y_{CN} = \sin(90 - \varphi) \times y_{CA}$

De manière analogue : $\cos(90 - \varphi) = \dfrac{z_{CN}}{y_{CA}} \Leftrightarrow z_{CN} = \cos(90 - \varphi) \times y_{CA}$

Etape 2 : calcul de l'ouverture de la lentille

h_1 et h_2 sont les hauteurs en face des angles respectifs $\alpha + \varphi$ et $\alpha - \varphi$

On calcule la limite haute de position : $\tan(\varphi + \alpha) = \dfrac{h_1}{z_P + c_N}$ et $\tan(\alpha - \varphi) = \dfrac{h_2}{z_P + c_N}$

D'où : $aperture = h_1 + h_2$, $y_{debut\ ouverture} = y_{CN} - h_1$
et $y_{debut\ ouverture} + ou - aperture =$ position du pixel sur l'écran

Etape 3 : algorithme pour la matrice de lentilles

```
Fonction position_centre_lentille(arguments nécessaires) :
    • calculer cône de visualisation 1ere lentille ;
    Tant que (coor_pixel ∉ cône de visualisation) faire
        • passage à lentille suivante ;
        • calculer cône de visualisation lentille courante ;
    Fait
    c = calculer le centre de la lentille
    Retourner c ;
Fin
```

Algorithme 1: *La fonction* POSITION_CENTRE_LENTILLE

3 Brouillon de plan pour le rapport

cf document 2 en pièce jointe du mail

4 Objectifs pour les prochains jours

- Vérifier les unités de toutes les variables intervenant dans les calculs,
- Etudier dans quels repères sont exprimées certains paramètres,
- Avoir le code du modèle de lentille convergente fonctionnel après avoir vérifié les 2 points ci-dessus et implémenter le modèle de matrice de lentilles.

Prochaine réunion : à confirmer.