Université Gustave Eiffel — INSTITUT D'ÉLECTRONIQUE ET D'INFORMATIQUE GASPARD-MONGE

Master Informatique 1

Correction TD Infographie n°3

RayTracer 2.0 - Caméra : construction et positionnement

Dans cette séance, l'idée est de construire une pseudo caméra 2D, à l'image de ce que sera la caméra 3D finale et de mettre en place le tout premier niveau de l'algorithme de Lancer de Rayon.

"pseudo" Caméra dans $\mathbb{R}^2$

Une caméra est définie par un *écran de pixels* (ici, une simple ligne de n pixels), un point de visée (oeil) qui est le point de convergence des rayons traversant l'écran (comme un appareil photo), un vecteur de visée, issu de l'oeil, dirigé vers un point de la scène (qui sera au centre de l'écran) et une distance focale définissant la position de l'écran sur le vecteur de visée. Cette distance focale fixe l'angle d'ouverture de la caméra.

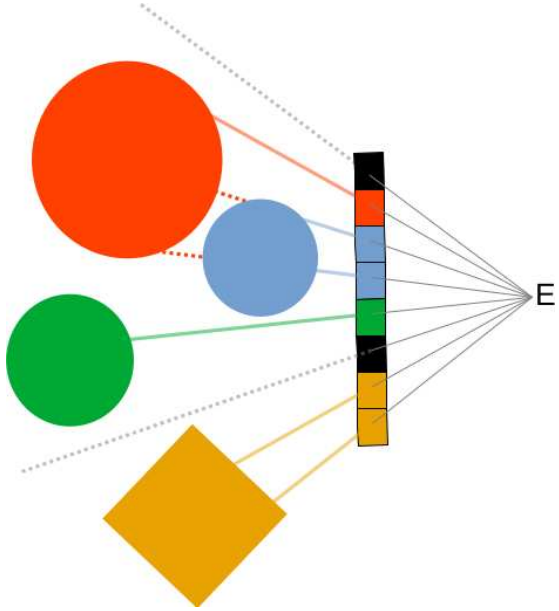
Le principe général du Lancer de Rayon est le suivant :

- Pour chaque pixel P_i de l'écran, on lance un rayon $[P_i, \overrightarrow{u_i})$ de direction $\overrightarrow{EP_i}$ (E étant la position de l'oeil). Ce rayon va se propager dans la scène et peut-être intersecter des objets.
  L'algorithme va donc devoir tester *tous* les objets et garder en mémoire le plus proche.

- Pour chaque objet rencontré, le rayon détermine la distance de P_i au point d'intersection trouvé. Si cette distance est inférieure à celle stockée dans le rayon, celui-ci enregistre l'*id* de l'objet et la nouvelle distance.

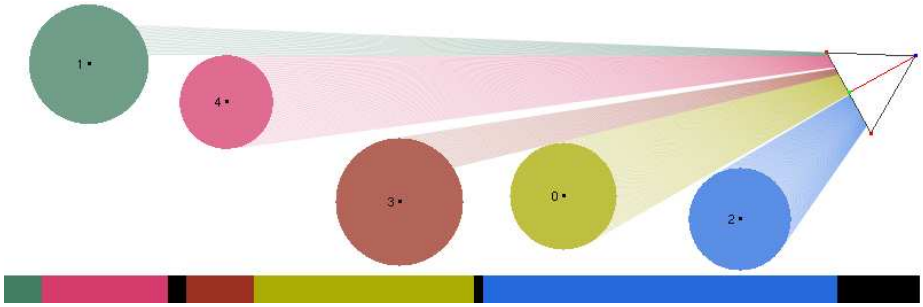
- Lorsque tous les objets ont été testés, le rayon sait lequel est le plus proche. Le pixel P_i prend alors la couleur de cet objet.

- Si le rayon ne rencontre aucun objet, le pixel reste noir.



A ce stade, l'image finale ne représente que l'*empreinte* des objets, avec leur couleur. Aucune information de forme n'est encore disponible.

Pour cette première ébauche de caméra, on partira d'une caméra canonique de référence, que l'on pourra positionner et orienter à l'aide de transformations judicieusement construites.



▸ Exercice 1. (une camera 2D)

① Caméra canonique : définir un objet Camera constitué d'une matrice de transformation M_d, d'une "taille" d'image (nombre de pixels utiles) n et d'un tableau de couleurs (qui sera ajusté à la taille utile n dynamiquement) qui constituera l'image finale.

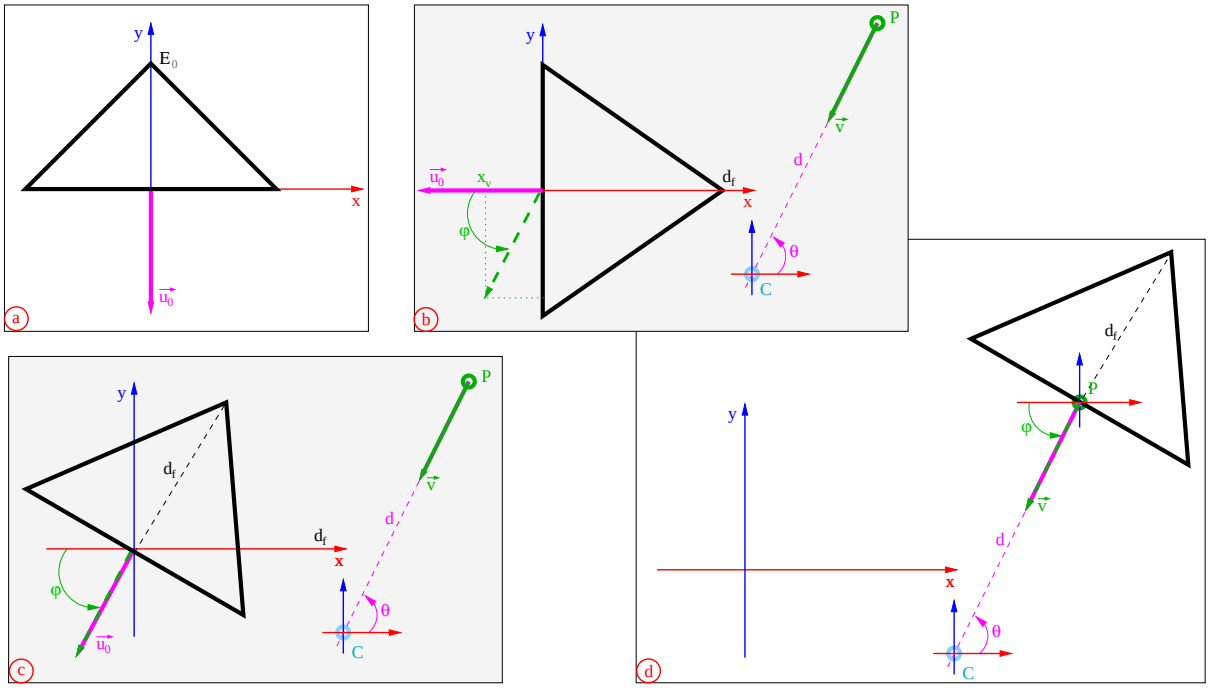
- La caméra canonique sera crée (initialisée) avec une taille de pixel maximale (N) et le programme permettra de contrôler sa "taille utile" n via un réel res $\in]0,1]$ (résolution n = res $*$ N).
- Par défaut, le centre de l'écran sera le point $\Omega(0.,0.)$, l'oeil sera en $E_0(0.,1)$ (et donc le vecteur de visée sera $\overrightarrow{u_0}(0,-1)$) et la "largeur" de l'écran sera de 2 (segment $[(-1.0),(+1,0)]$, sur lequel s'étalonneront les n "pixels" utiles).

② Orientation et positionnement : c'est la partie un peu subtile si on veut pouvoir contrôler la caméra

La méthode la plus efficace consiste à simuler ce que l'on fait dans la réalité : on choisit une position P et un point de visée (ou cible $C(x_c,y_c)$), puis on règle la distance focale d_f pour ajuster le champs de vision. Choisir une position peut se faire en choisissant directement un point de l'espace $P(x_p,y_p)$, mais il est plus pratique de choisir une distance à la cible d et un angle θ (coordonnées sphériques) qui deviennent alors des paramètres contrôlables de la caméra, que l'on peut ajuster facilement sans changer le point de visée (la caméra tourne autour, s'en approche ou s'en éloigne).

La mise à jour de la caméra consiste essentiellement à calculer sa matrice de transformation M_d :

- M_d est initialisée à l'Identité – cadre ⓐ
- On lui applique l'homothétie de rapports $(1.,d_f)$, où d_f, distance focale, est un paramètre contrôlable.
- On applique une première rotation d'angle $\phi_0 = -\frac{\pi}{2}$ pour aligner correctement la caméra.
- On détermine la position du centre de l'écran, par rapport à la cible, d'après ses coordonnées sphériques (d,θ) : $P\begin{vmatrix} x_c + d\cos(\theta) \\ y_c + d\sin(\theta)\end{vmatrix}$, et on construit le vecteur de visée $\overrightarrow{v} = \overrightarrow{PC}/\|\overrightarrow{PC}\|$
- A ce stade, la configuration serait telle qu'au cadre ⓑ
  Il reste à déterminer la rotation d'angle ϕ nécessaire pour aligner le vecteur de visée $\overrightarrow{u_0}$ sur le vecteur souhaité $\overrightarrow{v}$ – cadre ⓒ
  Si le vecteur $\overrightarrow{v}$ à bien été normé, cet angle ϕ_v est donné par $\phi_v = \begin{vmatrix} +\arccos(x_v) & \text{si} & y_v>0 \\ -\arccos(x_v) & \text{si} & y_u<0 \end{vmatrix}$
  (c'est la seule et unique fois où l'on fera appel à une telle fonction !).
- Il ne reste plus qu'à appliquer une translation de vecteur $\overrightarrow{\Omega P}$ pour positionner la camera dans sa configuration finale – cadre ⓓ



Au total, la matrice est donnée par $\mathsf{M_d} = \mathsf{T}_{\overrightarrow{\Omega P}} \times \mathsf{R}_{(\phi_v - \frac{\pi}{2})} \times \mathsf{H}_{(1.,d_f)}$

En 3D le principe sera le même sauf qu'il y aura 2 angles (ϕ_v, ψ_v) à déterminer (rotations autours de (z) et de (y).

③ Dimensionnement : cela consiste simplement à ajuster la taille utile n en fonction de la résolution res souhaitée. Cet ajustement aura bien sûr un impact direct sur le positionnement des "pixels" lors de la création des rayons primaires.

▸ Exercice 2. (lancer de rayons primaires)

Les rayons primaires *canoniques* sont ceux issus de la caméra canonique : en fonction du paramètre de résolution, on définit les centres de chacun des n pixels canoniques utiles sur l'écran canonique $\mathsf{P}^0_k(-1.+\frac{2.k}{n},\ 0.)$

- on construit le rayon $\mathsf{R}^0_k[\mathsf{P}^0_k, \overrightarrow{u^0_k})$ (où $\overrightarrow{u^0_k} = \overrightarrow{E^0 P^0_k}$).
- on "envoie" ce rayon dans le repère général : $\mathsf{R}_k[\mathsf{P}_k, \overrightarrow{u_k})$ où $\mathsf{P}_k = \mathsf{M_d} \times \mathsf{P}^0_k$ et $\overrightarrow{u_k} = \mathsf{M_d} \times \overrightarrow{u^0_k}$
  ☞il faut penser à renormer $\overrightarrow{u_k}$ – la matrice contient une homothétie !
- on "lance" ce rayon $\mathsf{R}_k[\mathsf{P}_k, \overrightarrow{u_k})$ à travers la scène (intersection avec les objets) et on "remonte" la couleur de l'objet le plus proche.
- la couleur ainsi récupérée est placée dans le tableau de couleur de la caméra, au bon endroit (indice k, comme le pixel).

A ce stade, il n'y a pas encore de récursivité. Seule l'"empreinte" des objets apparait, sans effets de relief, ni rélections.

☞ Mettre en place un programme qui permet de piloter une caméra et d'afficher (par exemple sous la forme d'une simple barre de couleurs) la trace d'objets placés dans la scène. Les objets pourront, pour l'instant, être de simples cercles (donnés par leur centre, leur rayon et leur couleur).

Pour ce premier niveau on se focalisera sur le paramétrage et le pilotage de la caméra via ses paramètre $(\mathsf{d_f}, \mathsf{d}, \theta, \mathsf{res})$ accessibles par des ScrollBar ou tout autre moyen jugé utile.

▸ **Exercice 3. (Orientation d'un écran en 3D)**

La contruction et la mise en place d'une caméra en 3D suit exactement les mêmes principes qu'en 2D. On supposera, à partir de maintenant, que la Caméra Canonique 3D est définie par :

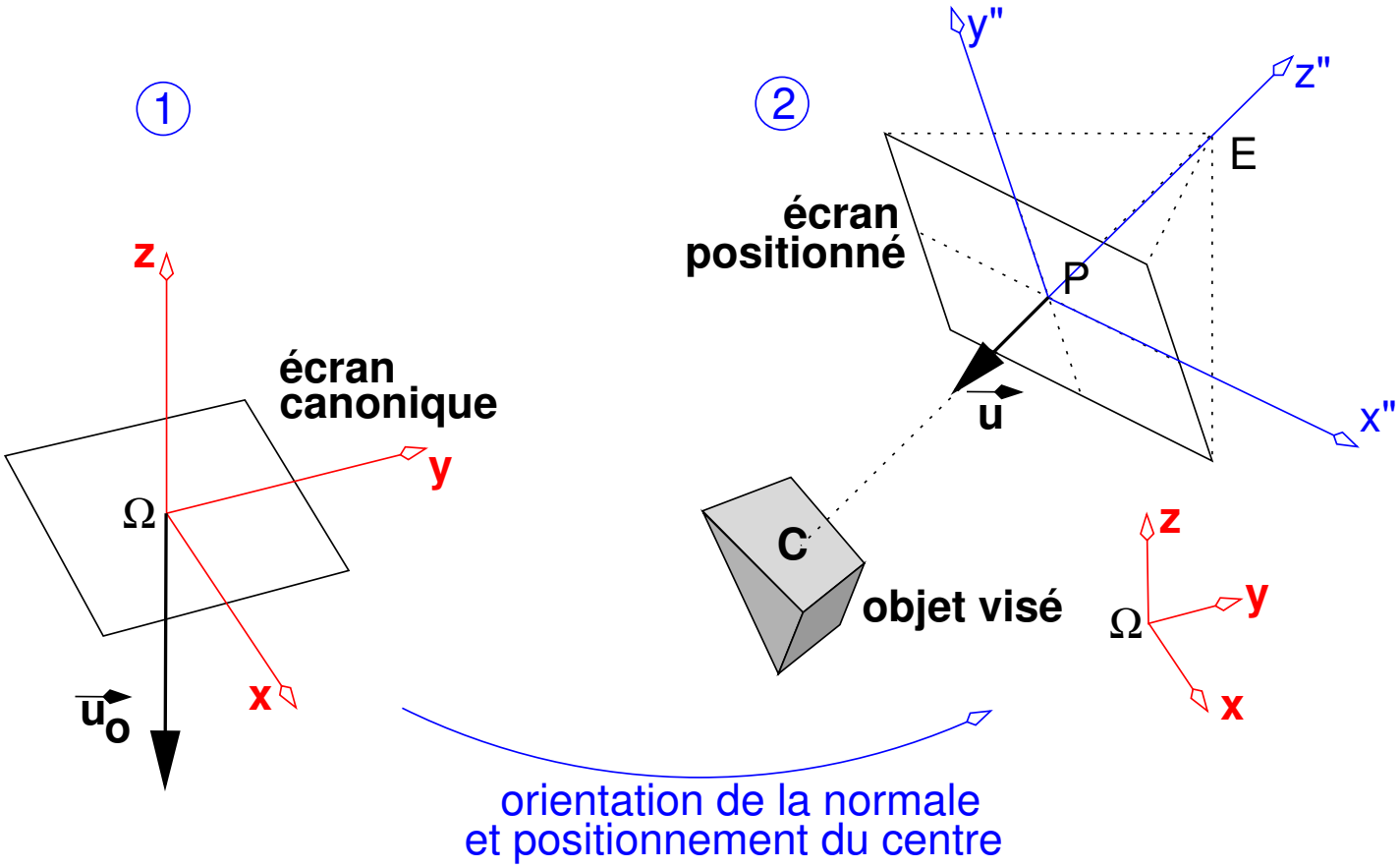
- une (ou deux) variables de résolution (selon qu'on souhaite une image carrée ou non) dont découlent les tailles utiles nbl et nbc (nombres de lignes et de colonnes).
- une matrice de transformation directe $\mathsf{M_d}$
- une colmap (carte des couleurs) de taille utiles nbl $*$ nbc.
- son écran est le carré canonique $\{(-1,-1,0) \leftrightarrow (+1,+1,0)\}$
- son point focal est $\mathsf{E_0} = (0,0,+1)$ et donc son vecteur de visée est $\overrightarrow{u_0} = (0,0,-1)$

Et comme en 2D, les caractéristiques géométriques (positionnement, orientation, dimensions, ouverture) sont toutes gérées directement par la matrice $\mathsf{M_d}$.

Pour ce qui est du positionnement et de l'orientation, il faut définir ces caractéristiques en terme de *rotations autour des axes.*

Mais orienter un écran dans une scène lorsque celui-ci est caractérisé simplement par ses trois angles de rotation autour des axes, est difficile car peu naturel. En effet, il est beaucoup plus naturel de régler l'orientation en précisant, par exemple, quel objet on souhaite voir au centre de la scène. Pour celà il suffit de fixer un point C de la scène, et d'orienter l'écran de sorte que son vecteur normal $\overrightarrow{u}$ soit colinéaire au vecteur $\overrightarrow{v} = \overrightarrow{PC}$ où P est le centre (ou position) de l'écran.

Les données du problème sont alors : la taille de l'écran (et sa résolution courante), la distance focale $\mathsf{d_f}$, la position du point de visée C, une distance au point de visée d et deux angles d'orientation (θ, ϕ)).
Déterminer la suite d'opérations à effectuer pour positionner et orienter correctement l'écran à partir de ces données.



Comme d'habitude, les opérations se font dans l'ordre inverse :

1. mise à l'échelle : on commence par le réglage de la distance focale $\mathsf{d_f}$ par une homotétie de rapports $(1., 1., \mathsf{d_f})$.
2. orientation par rapport à la normale : la normale $\overrightarrow{\mathsf{u_0}}$ à l'écran canonique n'est autre que le vecteur $\overrightarrow{\mathsf{Oz}}$. La première opération est donc une rotation d'angle $\phi_0 = -\frac{\pi}{2}$ autour de l'axe des z.
3. orientation de la normale : c'est l'étape la plus difficile. Il faut traduire en terme de rotation autour des axes le passage du vecteur $\overrightarrow{\mathsf{u_0}}$ au vecteur $\overrightarrow{\mathsf{v}}$ final, colinéaire à $\overrightarrow{\mathsf{PC}}$. On procèdera en plusieurs étapes en s'aidant de l'exercice précédent.
   - Passer $\overrightarrow{\mathsf{v}}$ en coordonnées sphériques :

$$\overrightarrow{\mathsf{v}} = \left| \begin{array}{l} x_v \\ y_v \\ z_v \end{array} \right. = \frac{\overrightarrow{\mathsf{PC}}}{\|\overrightarrow{\mathsf{PC}}\|} \quad et \quad \overrightarrow{\mathsf{v}} = \left| \begin{array}{l} \cos(\theta)\sin(\phi) \\ \sin(\theta)\sin(\phi) \\ \cos(\phi) \end{array} \right.$$

$$\Longrightarrow \text{ en coord. sphériques } : \overrightarrow{\mathsf{v}} = \left| \begin{array}{l} 1 \\ \phi = \arccos(z_v) \ \in [0, \pi[\\ \theta = \pm \arccos(\frac{x_v}{\sin(\phi)}) \ \in [0, 2\pi[\ \ si\ \phi \neq 0 \ \ et\ selon\ signe\ de\ y_v \end{array} \right.$$

   - Rotation d'angle ϕ autour de l'axe des y, pour placer $\overrightarrow{\mathsf{u_0}}$ dans le plan $(\mathsf{y} = \mathsf{0})$
   - Rotation d'angle θ autour de l'axe des z, pour rendre $\overrightarrow{\mathsf{u_0}}$ colinéaire à $\overrightarrow{\mathsf{v}}$
4. positionnement : on finit par la translation de vecteur $\overrightarrow{\Omega\mathsf{P}}$