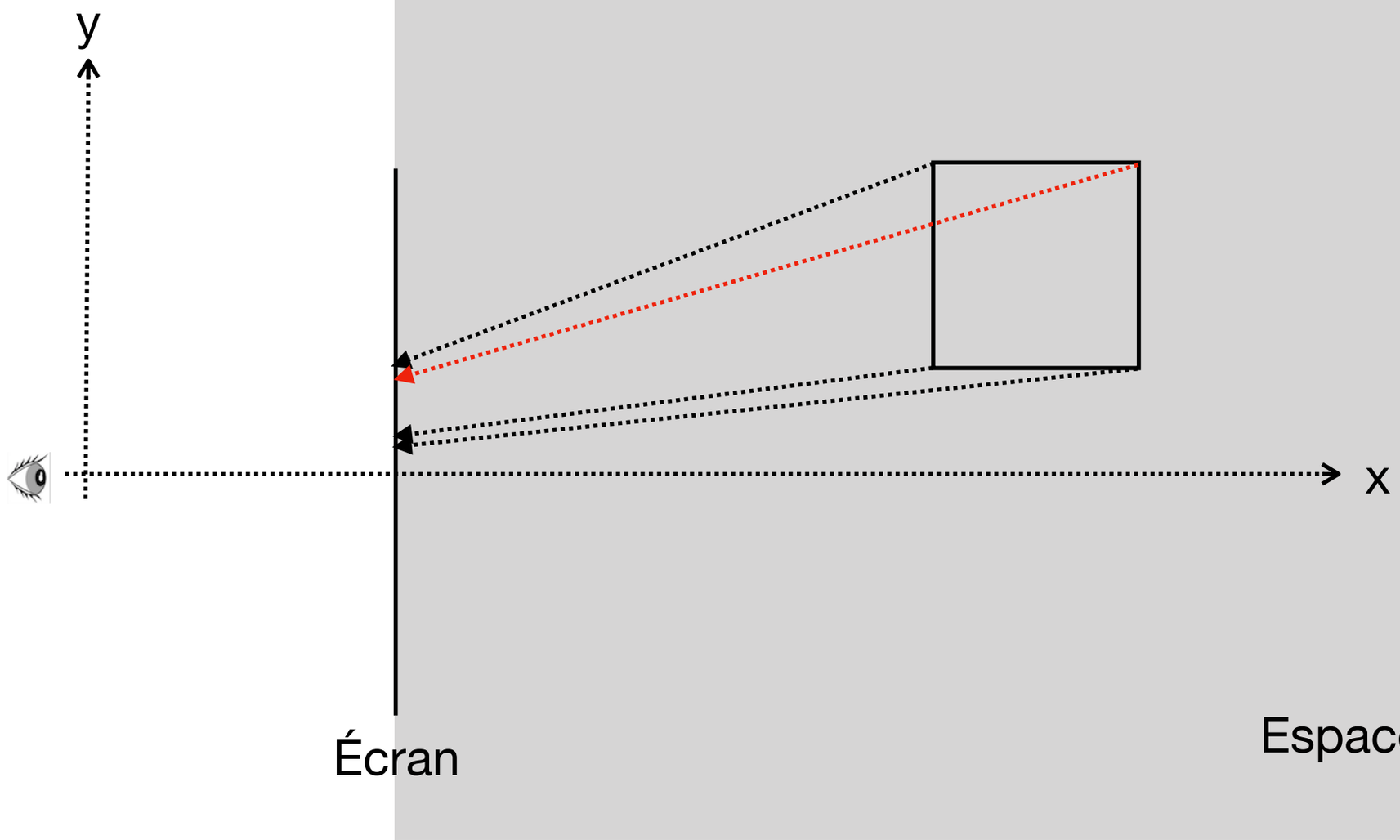
le Lancer de rayon
Ray casting

Jean-Baptiste.Yunes@univ-paris-diderot.fr
10/2018

- Ray casting vs ray tracing
    - Ray casting : lancer de rayon
        - fait référence à la technique géométrique permettant de déterminer ce qu'un œil, dans une direction donnée, voit en calculant l'intersection d'une demi-droite issue de l'œil avec les objets de la scène
    - Ray tracing : lancer de rayon
        - fait référence à un algorithme de vision «réaliste» utilisant le ray casting pour déterminer les caractéristiques lumineuses de ce qui est «vu»

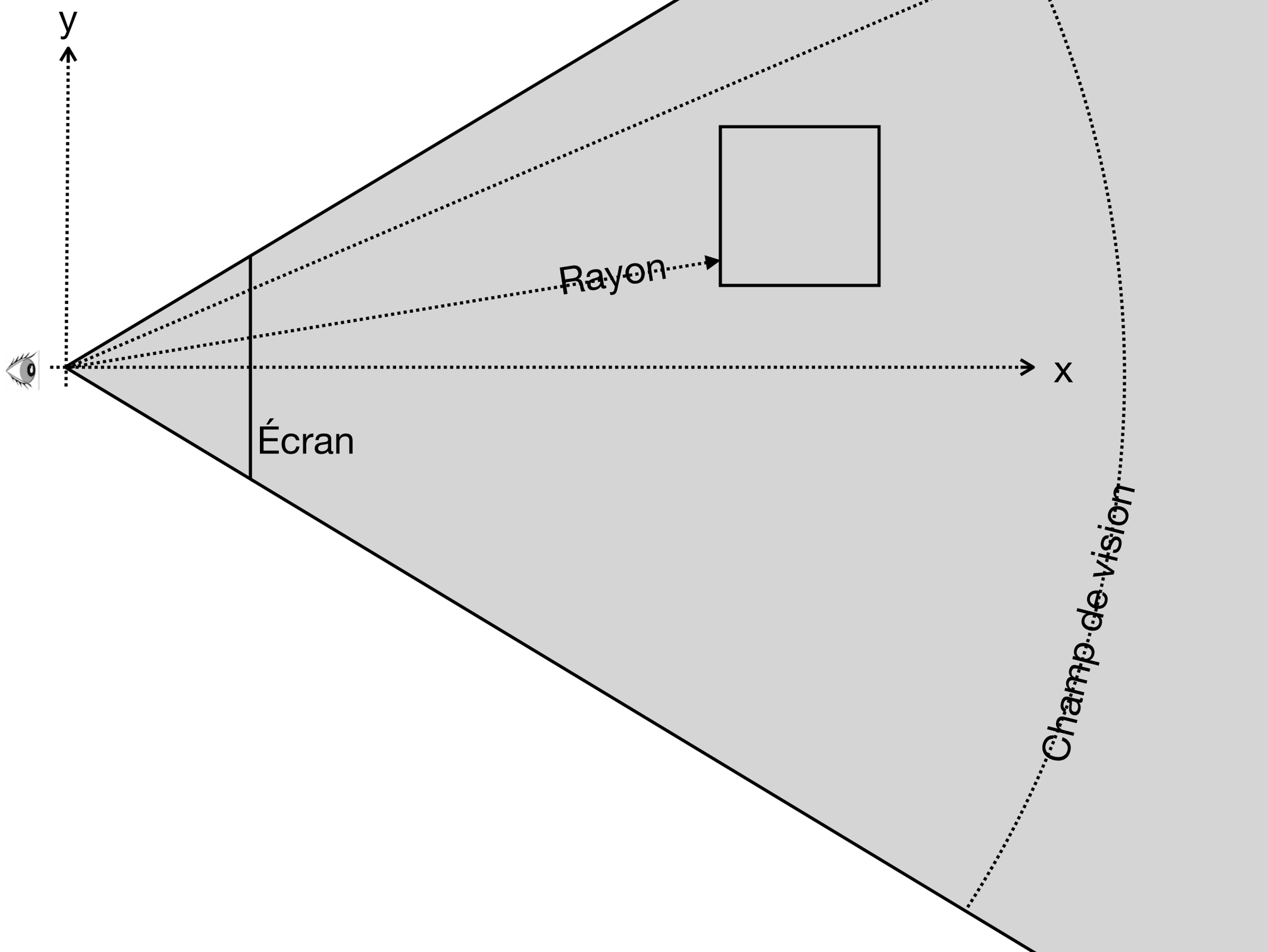
- Le ray casting est donc une technique d’élimination de faces cachées
- c’est une technique inverse des projections de scène
    - on procède en partant de l’œil plutôt qu’à partir des objets

- Projection

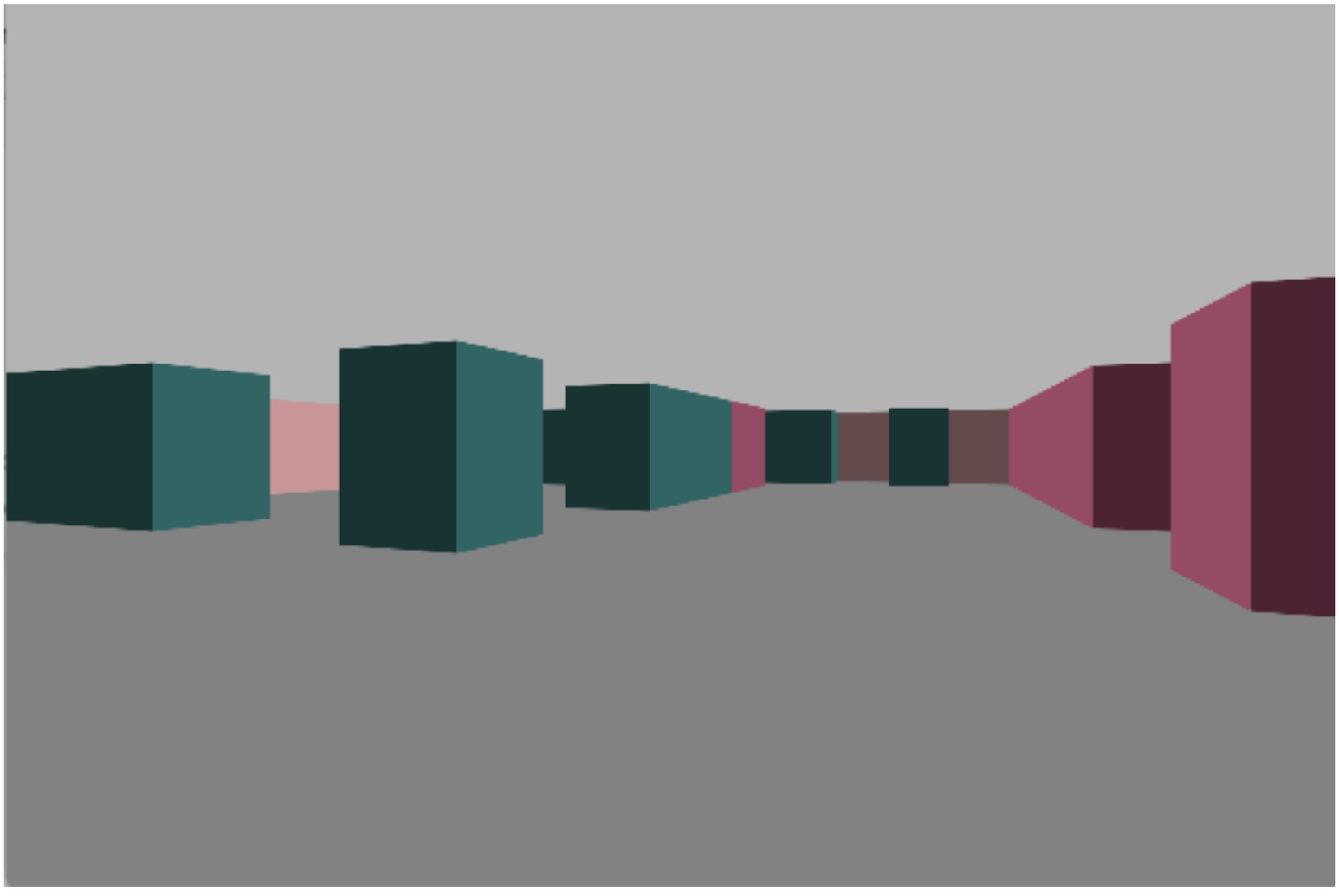


- on prend ce qui peut être vu et on le projette (sous condition éventuelle de visibilité - face cachées)

- Ray casting

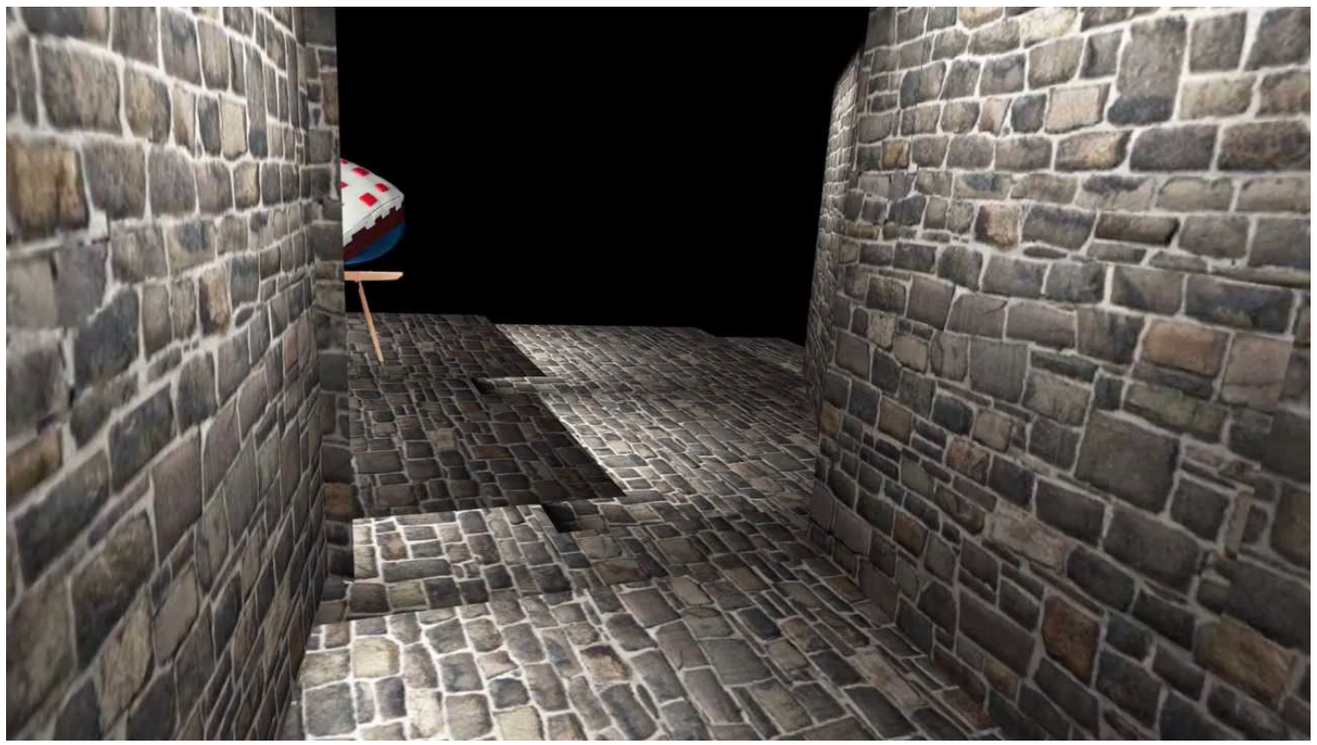


- Application du lancer de rayon
  - Obtention d’une vision 3D d’un modèle 2D

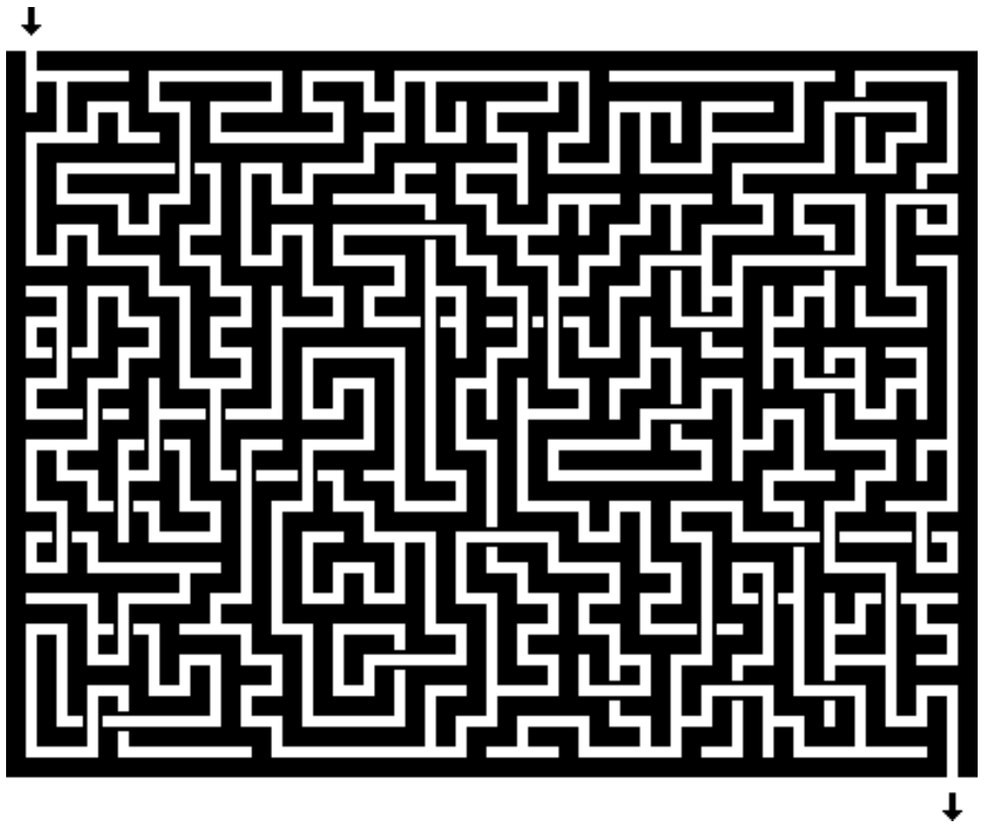


- Technique dite «2,5D» ou «pseudo 3D»
  - utilisée dans de nombreux contextes consistant essentiellement à se déplacer dans un plan mais à en avoir une vision tridimensionnelle
    - c’est ce que l’on «vit» lorsqu’on se déplace dans un bâtiment (par exemple)
      - c’est pourquoi le réalisme obtenu est suffisant

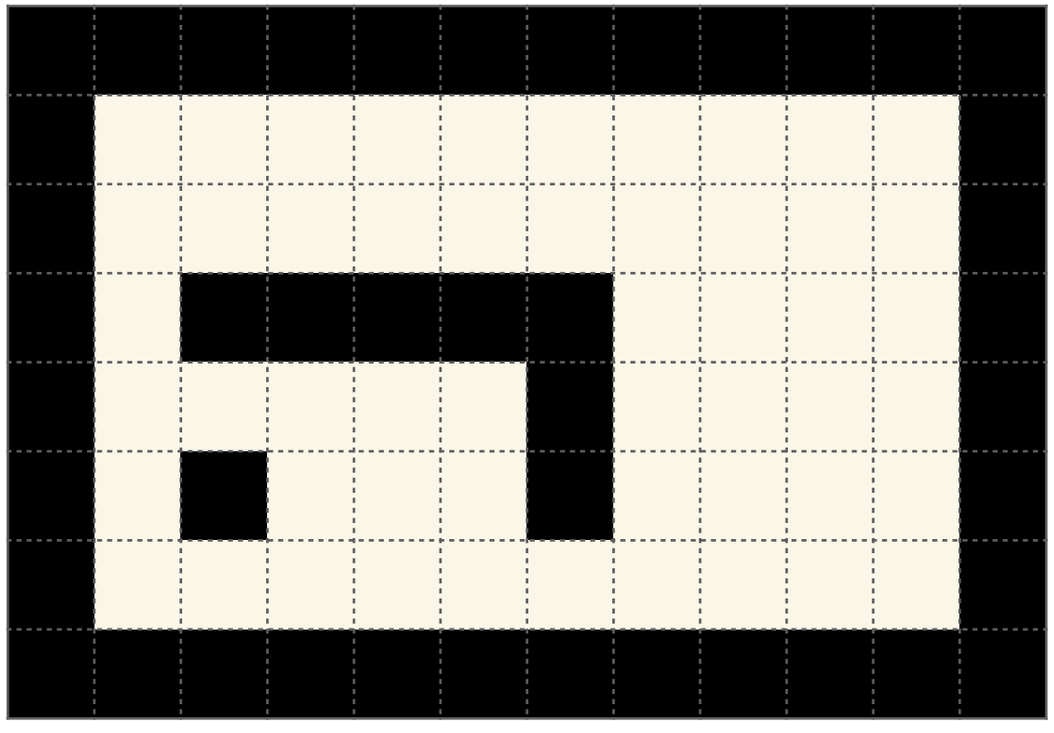
La représentation tridimensionnelle



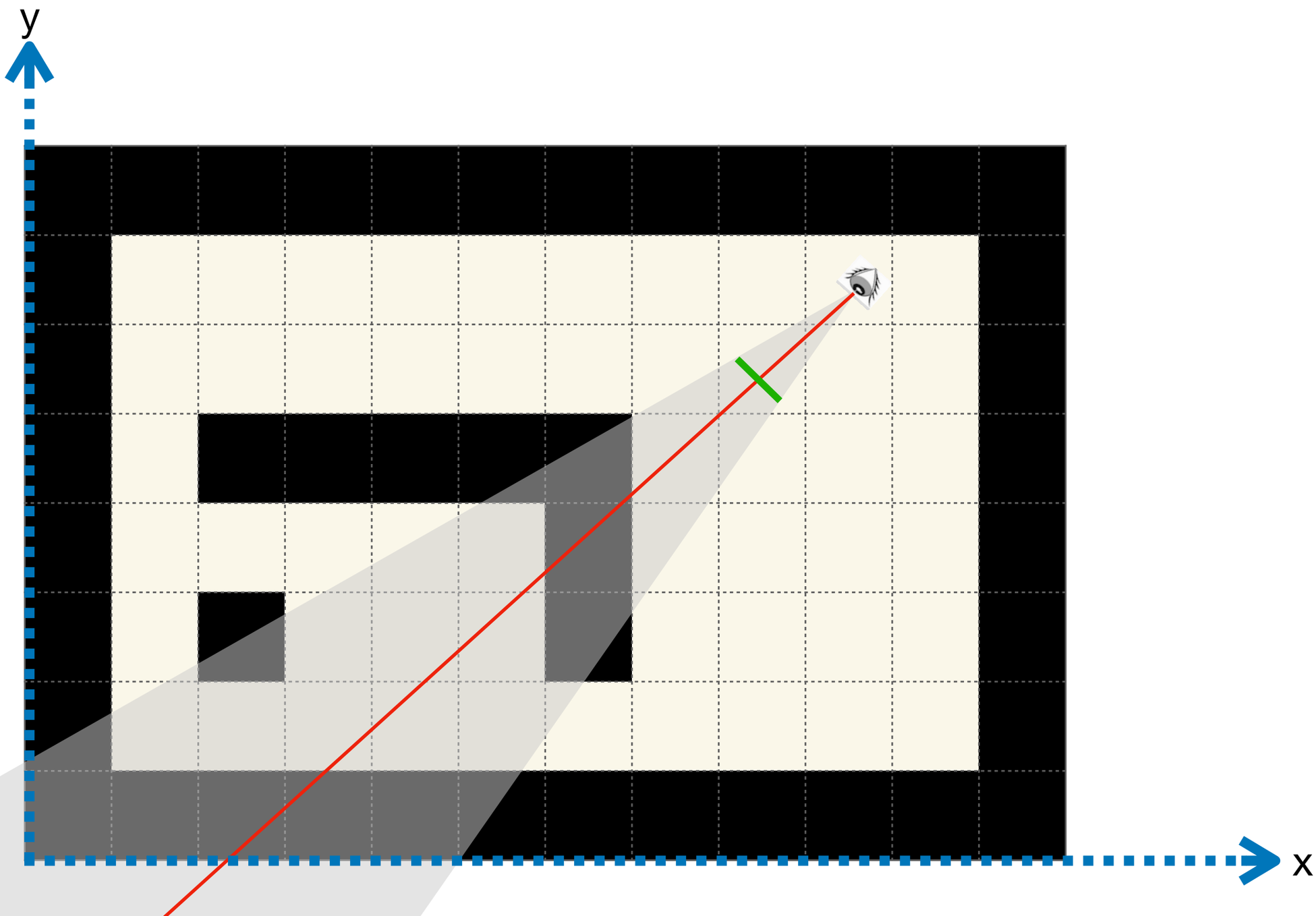
Le modèle plan



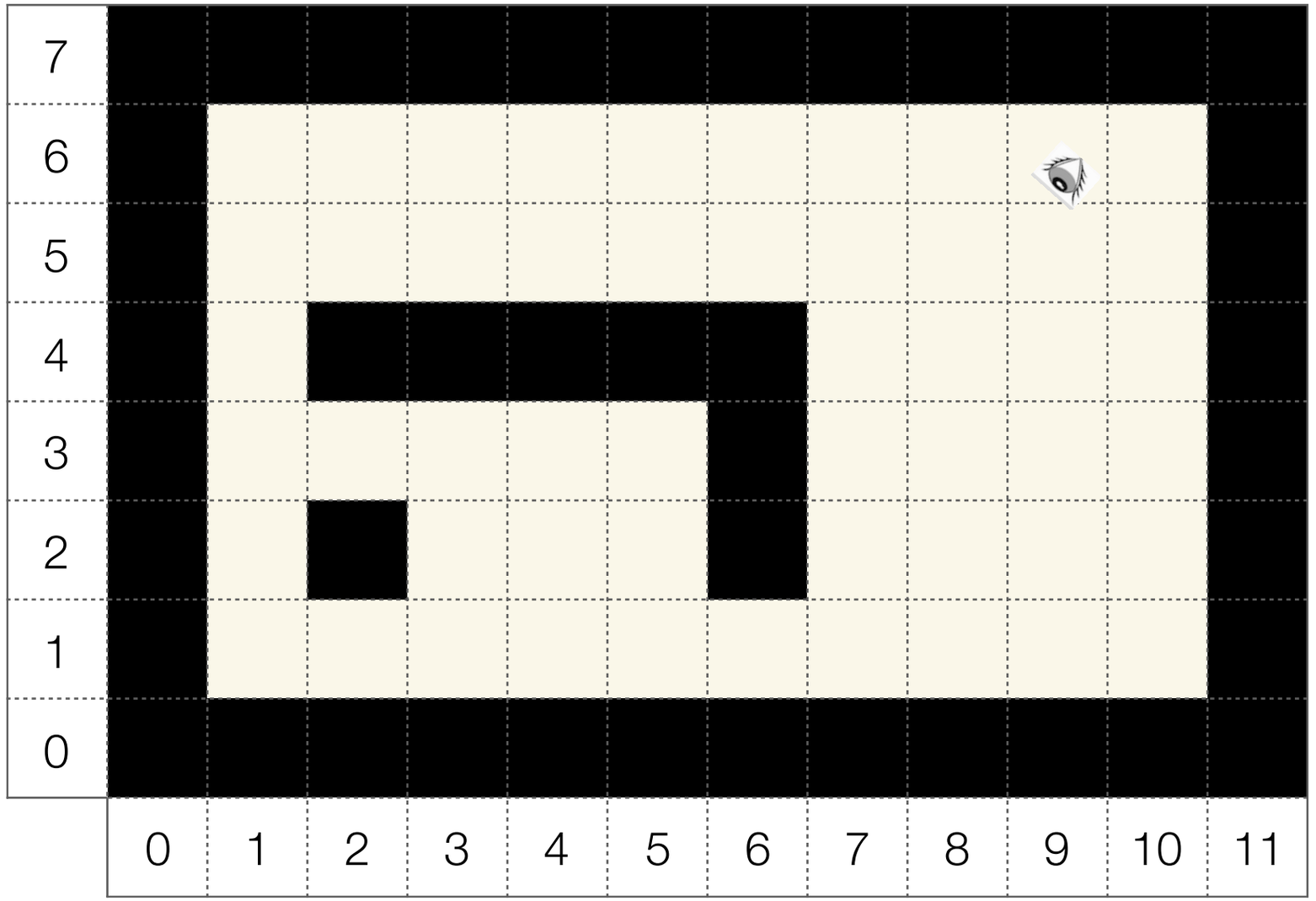
- Le modèle (exemple)
  - le monde sera décrit par une grille sur laquelle seront placés des «obstacles pleins» (des briques) toutes de même hauteur
    - le modèle est bien 3D mais la description de la troisième dimension est réduite à la constante «hauteur des briques», «hauteur sous plafond»



- Le personnage (l'œil) y sera **positionné** avec toutes ses caractéristiques associées : **écran**, **champ visuel**, **direction de vision**

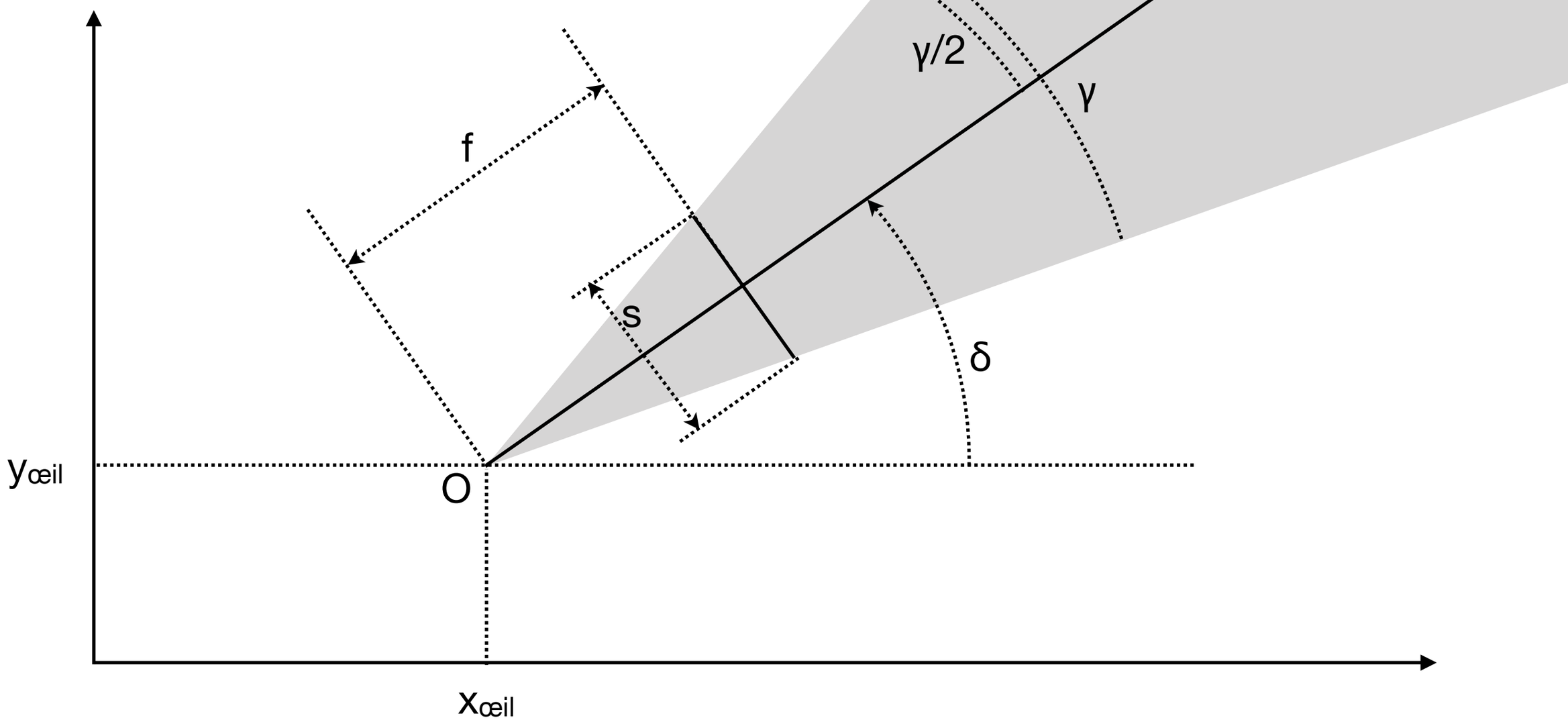


- Pour plus de finesse dans les déplacements l'œil pourra être placé en des points quelconques de l'espace continu sous-jacent
  - *ex.* : position = (8,5 ; 5,3)



- Domaine des variables
  - position de l'œil O $\in \mathbb{R}_+^2$
  - écran virtuel e $\in \mathbb{R}_+$
  - direction $\delta \in [0,2\pi[$
  - champ de vision $\gamma \in [0,2\pi[$
  - modèle m: $\mathbb{N}^2 \longmapsto \mathbb{B}$
- Écran, champ de vision et focale (si on préfère la focale) sont liés $f = e / \tan(\boldsymbol{\gamma}/2)$

- Conventions

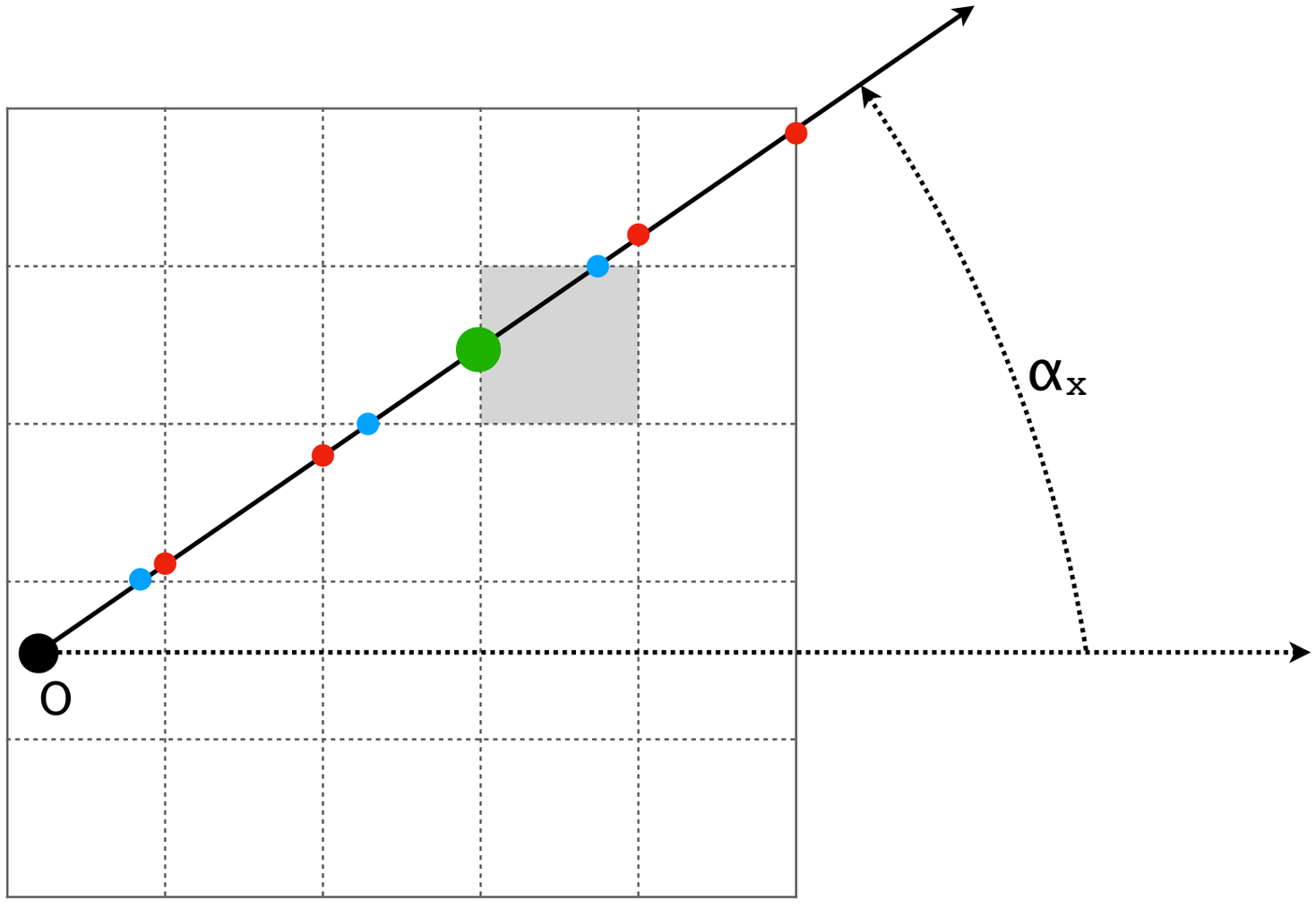


- Quels rayons lancer ?
    - première remarque : on doit lancer un rayon par pixel de l’écran
    - seconde remarque : nous n’avons qu’à lancer des rayons dans le plan de vision (*i.e.* dans le modèle plan)
    - l’effet 3D sera obtenu par la suite en résolvant la question de la hauteur apparente

- Pour un pixel de l'écran il faut déterminer l'angle du rayon correspondant, noté α_x
  - rappel : on ne considère que le plan du modèle, on travaille en 2D
- L'écran physique est de largeur $\texttt{l} \in \mathbb{N}_+$
  - le pixel milieu $(\texttt{l}/2)$ correspond à la direction de vision
- L'écran virtuel est de largeur $\texttt{s} \in \mathbb{R}_+$
- On a donc comme pseudo-code :

```
lancerLesRayons:
  pour x de 0 à l-1
    αx = δ + atan((s/2-x*s/(l-1))/f)
    lancer(αx)
```

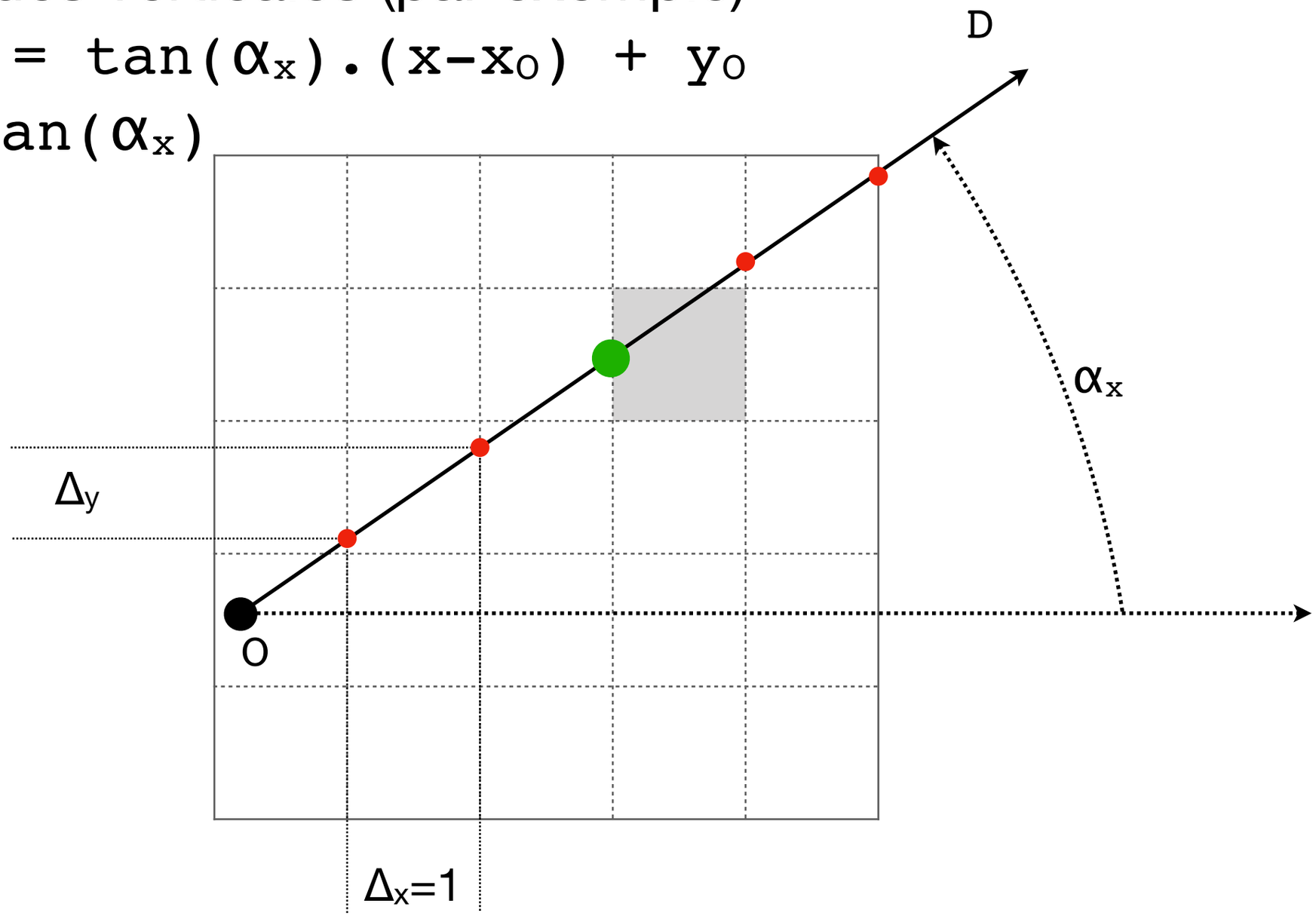
- Dans notre modèle déterminer le premier point d'intersection avec une brique consiste à
  - déterminer le premier point d'intersection avec une verticale ou une horizontale et qui correspond à une brique



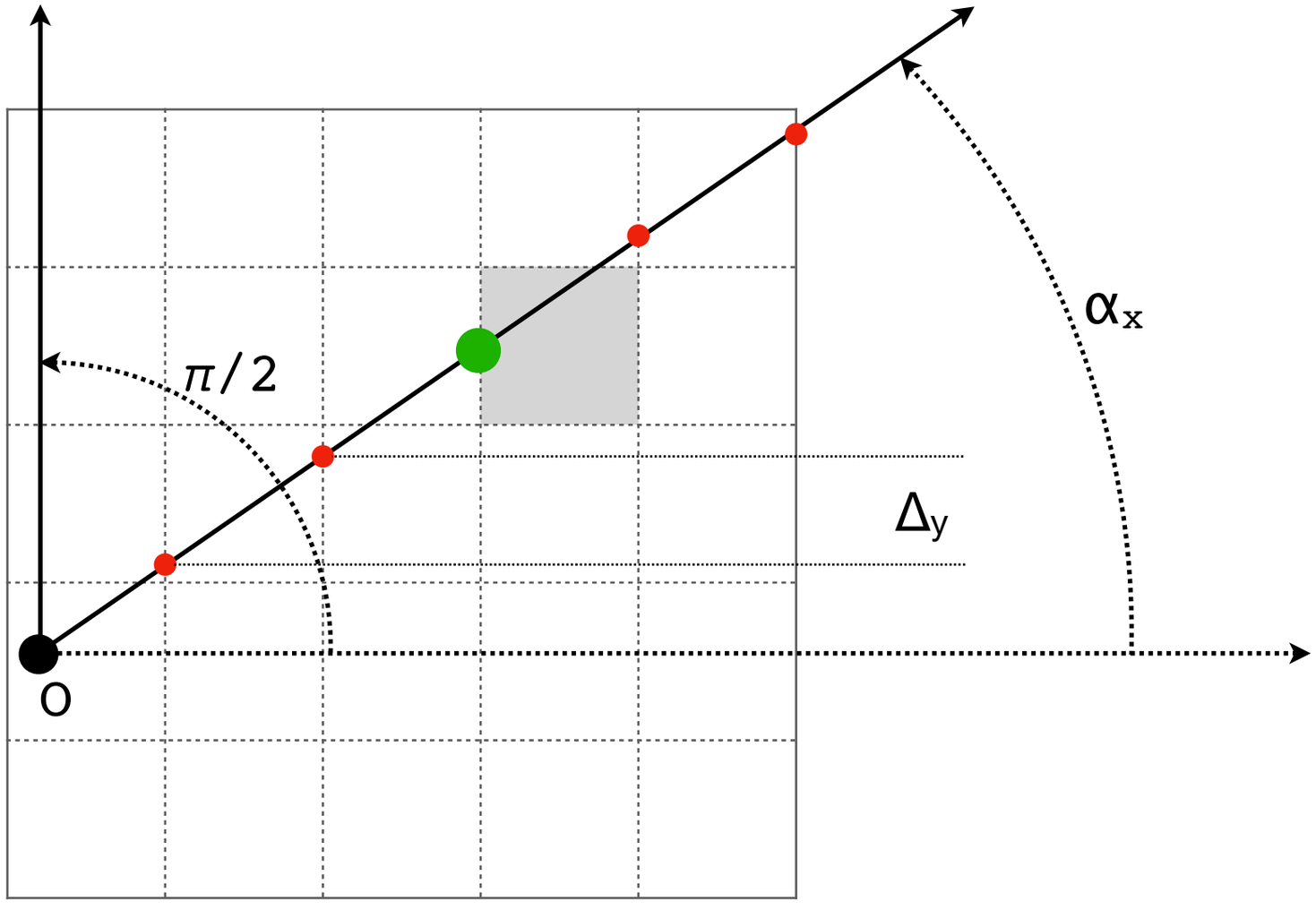
- On peut résoudre ce problème on peut utiliser effectuer une analyse différentielle puisque l'intervalle entre deux verticales ou horizontales est une constante
- Le cas des verticales (par exemple)

$D : y = \tan(\alpha_x).(x-x_0) + y_0$

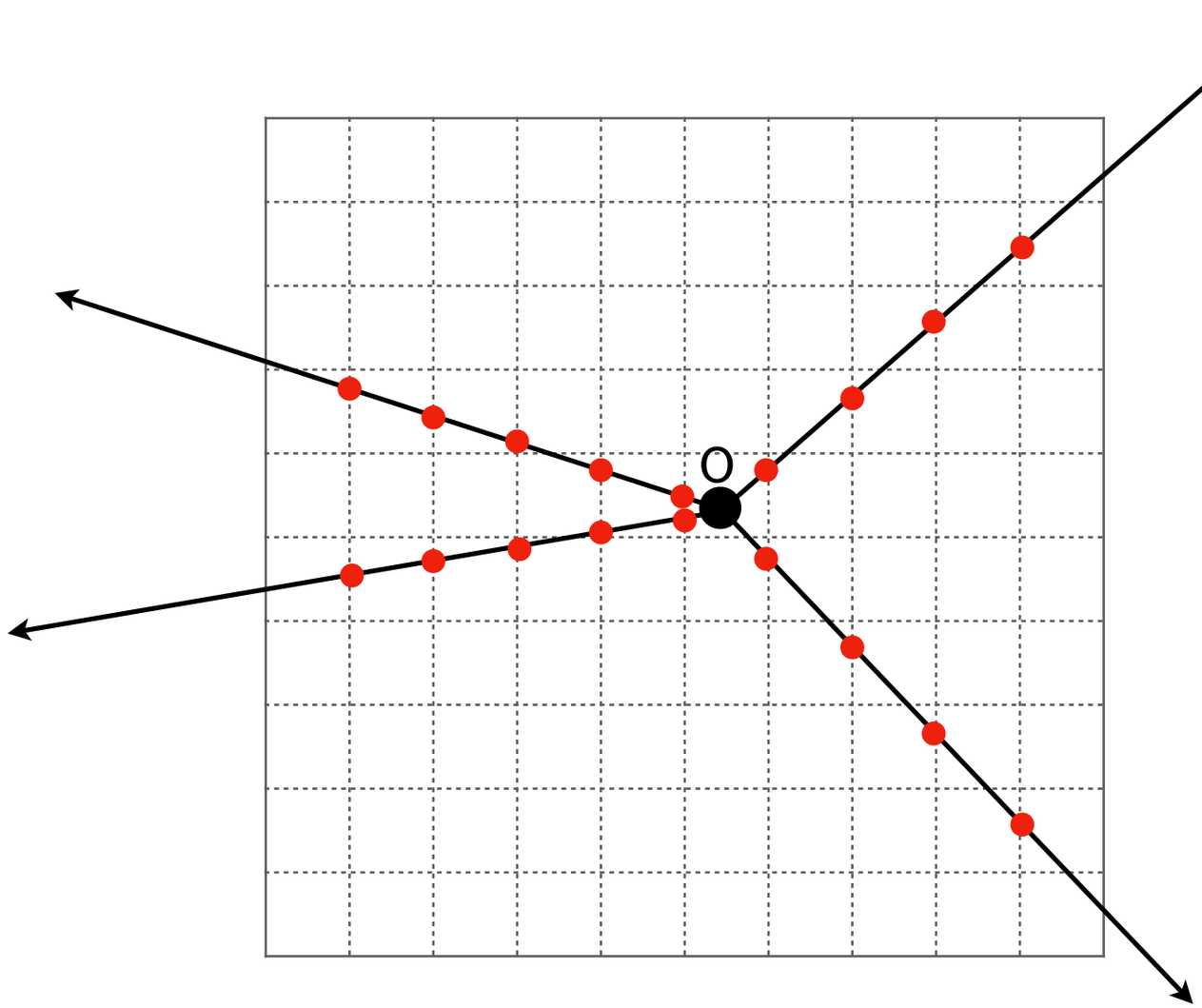
$\Delta_y = \tan(\alpha_x)$



- Le cas des verticales, $\Delta_y = \tan(\alpha_x)$
- Attention au cas particulier $\alpha_x = \pm\pi/2$
  - dans ce cas il n'y a pas de collision avec une verticale donc la distance de l'œil à la collision est ∞



- Quelles verticales traiter ?
  - Deux cas
    - $\alpha_x \in]-\pi/2,\pi/2[\Leftrightarrow \cos(\alpha_x)>0$
    - $\alpha_x \in]\pi/2,3\pi/2[\Leftrightarrow \cos(\alpha_x)<0$



- Pseudo-code

```
lancerCollisionEnX:
  cas cos(αx)>0
    pour vx de ⌊Ox⌋+1 à l-2
      pc = (vx , tan(αx)(vx-Ox)+Oy)
      si m(pc) est une brique
        return |pcp|
    return ∞
  cas cos(αx)<0
    pour vx de ⌊Ox⌋ à 1
      pc = (vx , tan(αx)(vx-Ox)+Oy)
      si m(gauche(pc)) est une brique
        return |pcp|
    return ∞
  cas cos(αx)=0
    return ∞
```

- Qu’est ce que gauche ???
  - m(p) désigne la case du modèle dont les coordonnées sont obtenues à l’aide de la partie entière des coordonnées de p
  - le problème est que notre modèle décrit la brique de coordonnées x,y $\in \mathbb{N}$ comme l’ensemble de la surface **ouverte** des points P / $\forall \varepsilon_x, \varepsilon_y \in [0,1[$, $P_x = (x+\varepsilon_x, y+\varepsilon_y)$
    - quand on arrive par la gauche pas de souci
    - quand on arrive par la droite, il faut donc considérer la brique à gauche
      - la fonction gauche(p) permet d’obtenir le point situé immédiatement à gauche du point p

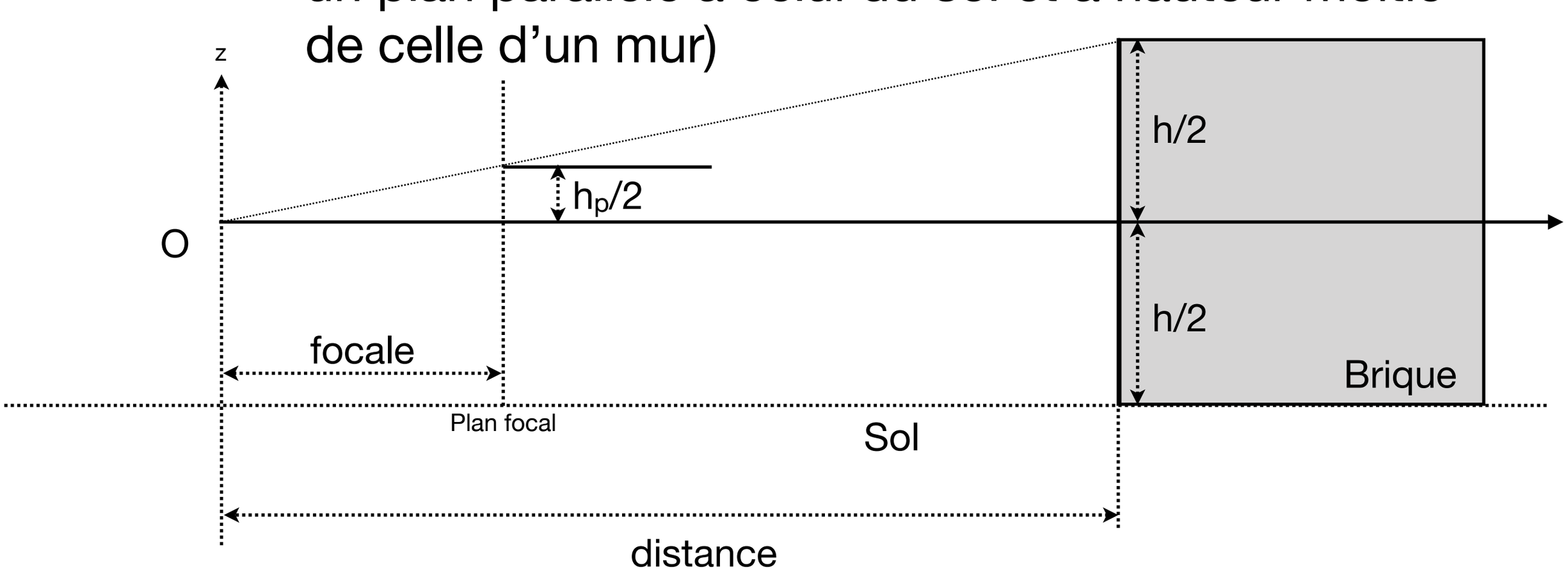
- Le cas des horizontales est traité à l’identique et ne pose pas de problème

- On a donc à notre disposition la distance de l'œil au point qui correspond au mur vertical le plus proche : $distance_{vx}$

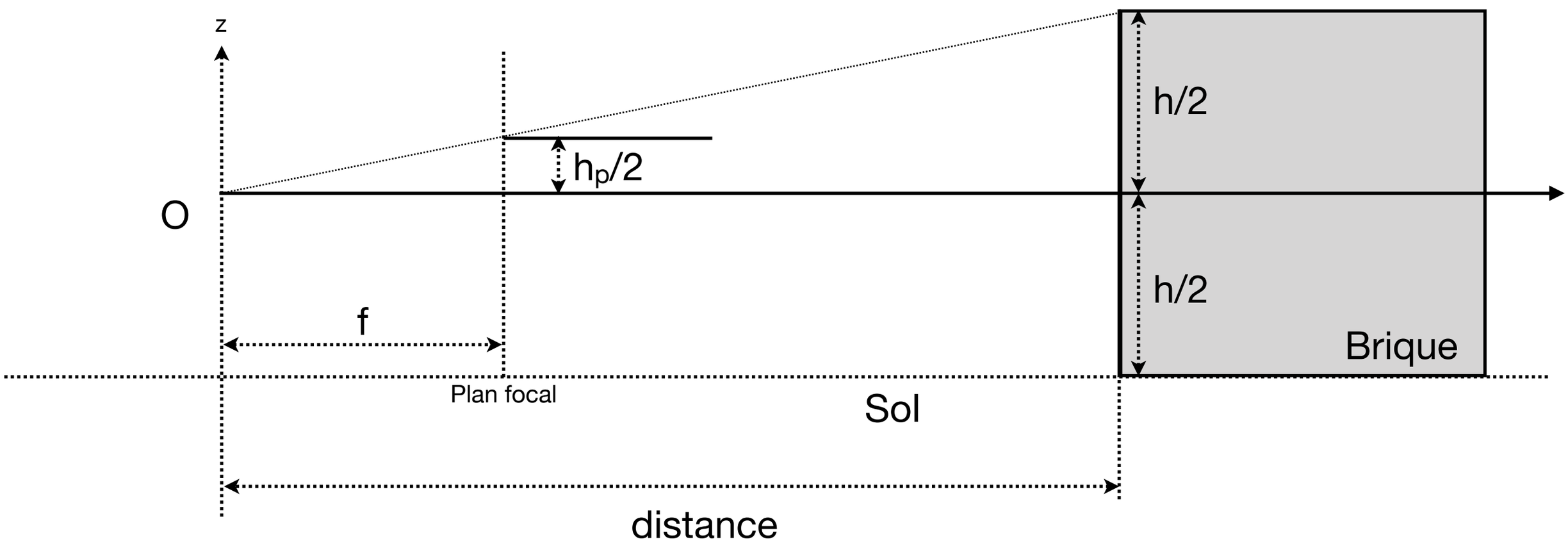
- La distance de l'œil au point qui correspond au mur horizontal le plus proche : $distance_{vy}$

- Le point le plus proche est donc (trivialement) celui de distance = min($distance_{vx}$,$distance_{vy}$)

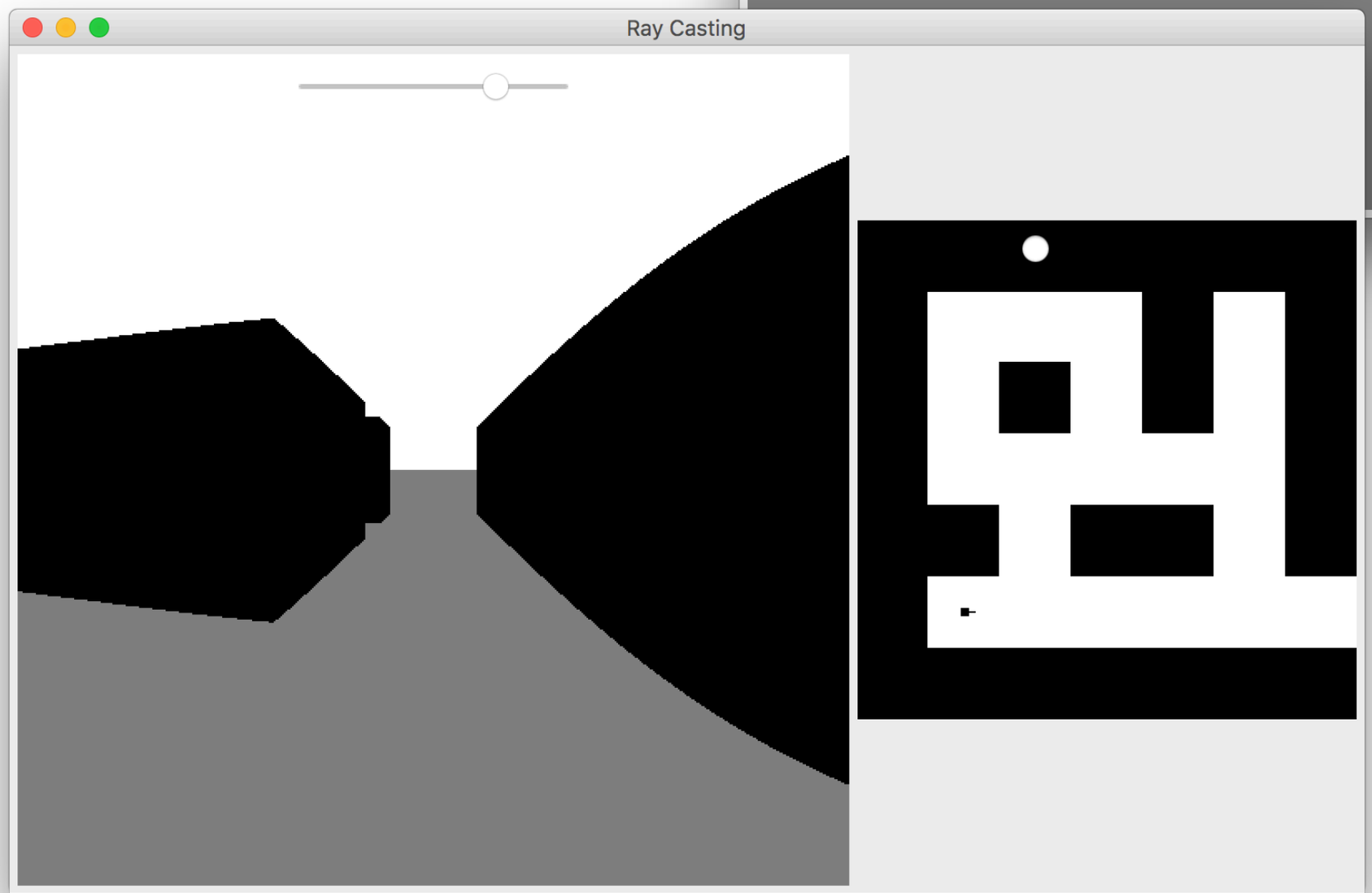
- La troisième dimension
- Calcul de la hauteur projetée h_p du mur visible (hauteur h)
  - Un simple calcul géométrique dans le plan vertical qui contient le rayon suffit
    - Il faut juste définir le regard
      - ici on considère que la direction de vision est dans un plan parallèle à celui du sol et à hauteur moitié de celle d'un mur)

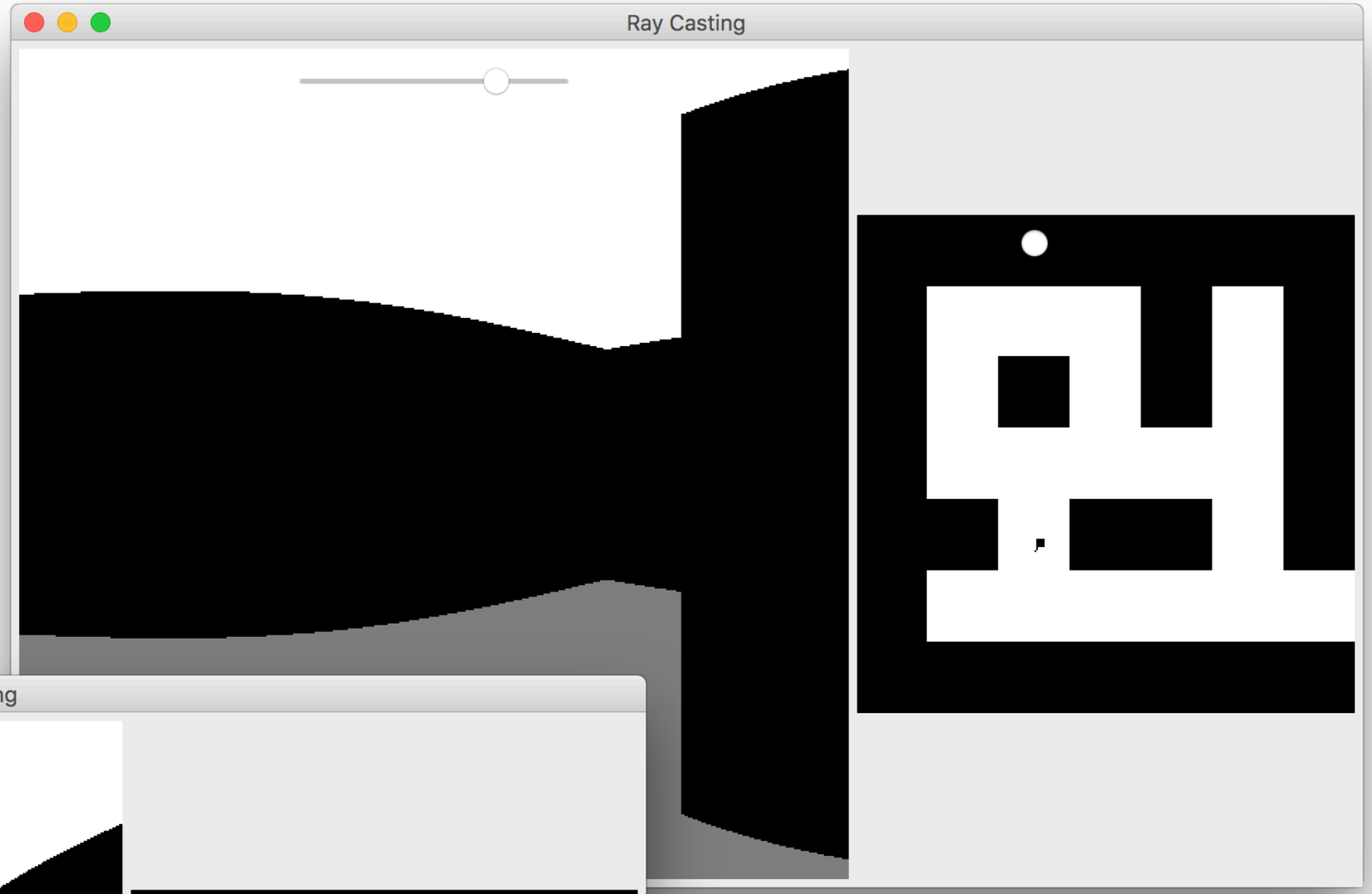


- On sait que f/distance = h_p/h
  - donc h_p = f.h/distance

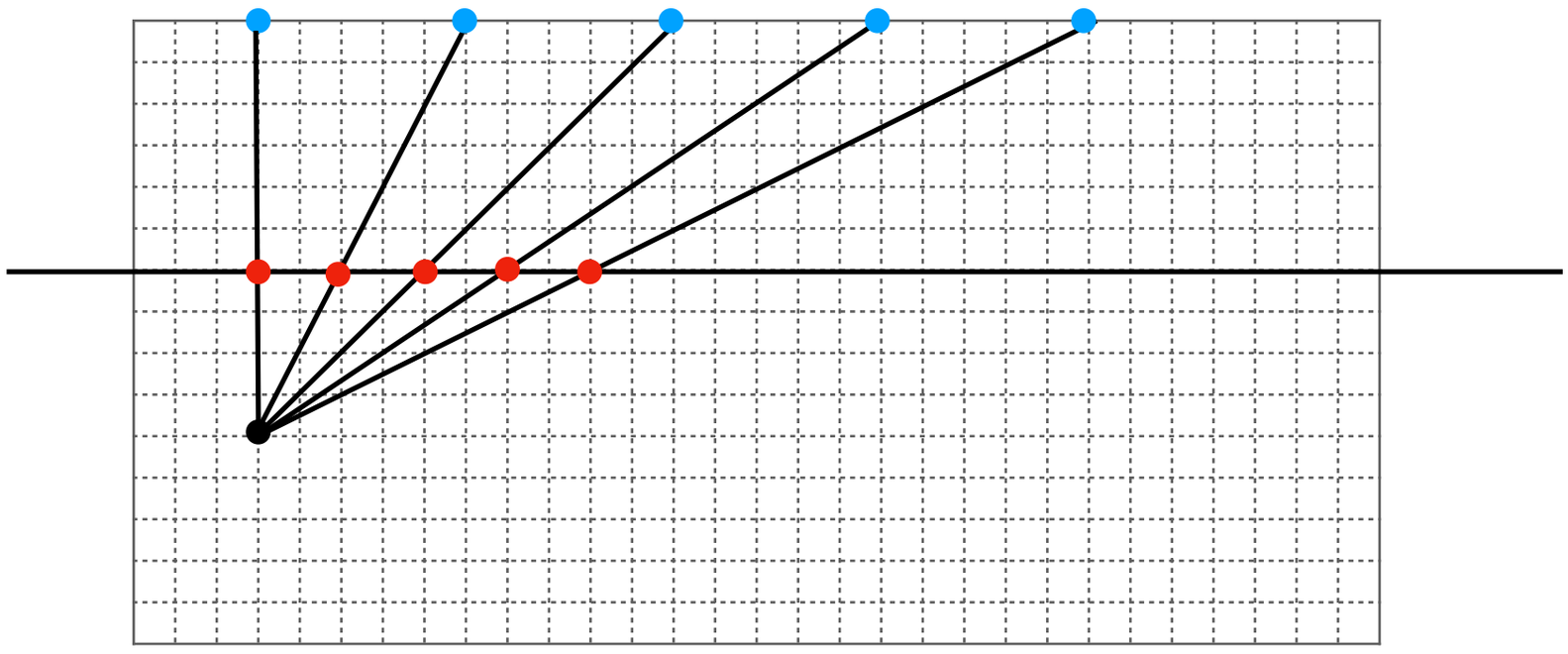


- Problème



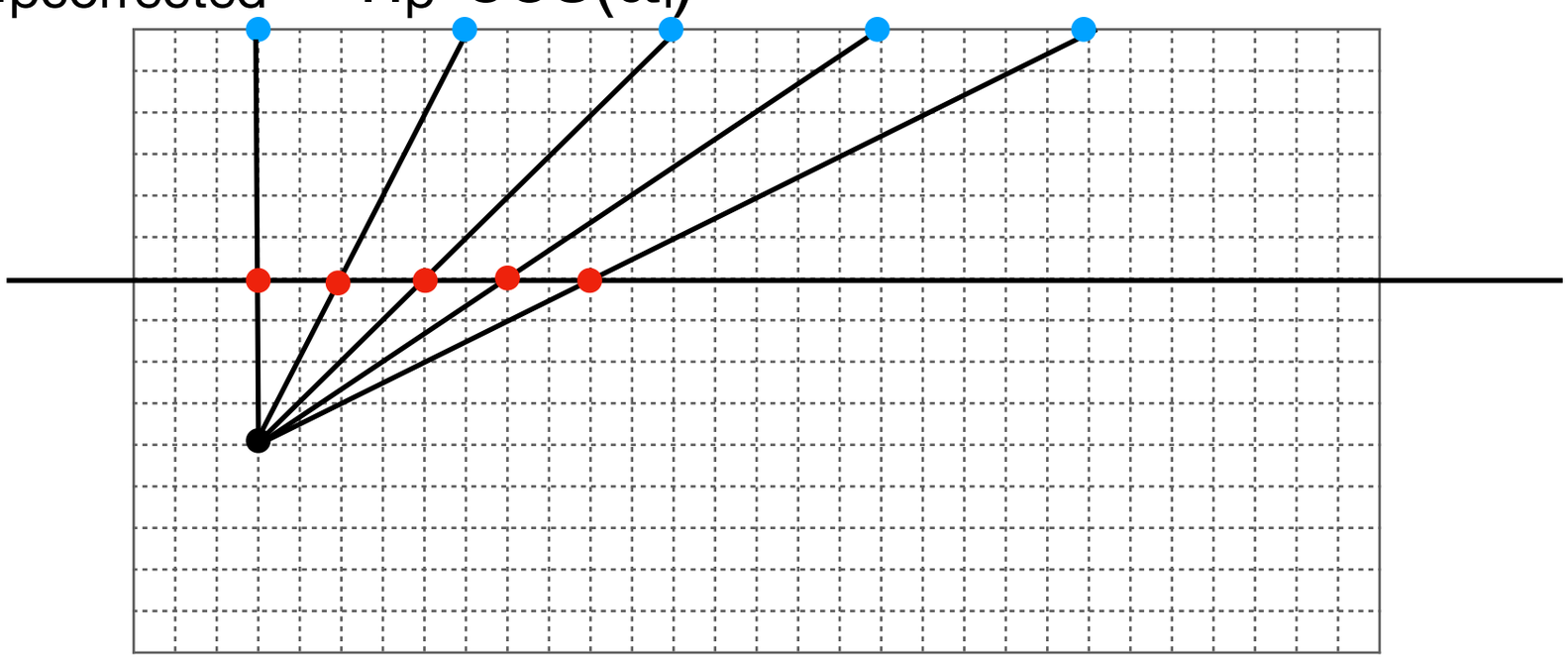


- Pourquoi les droites sont-elles courbes ?
  - Ce n’est pas le cas des verticales, c’est donc un problème de projection

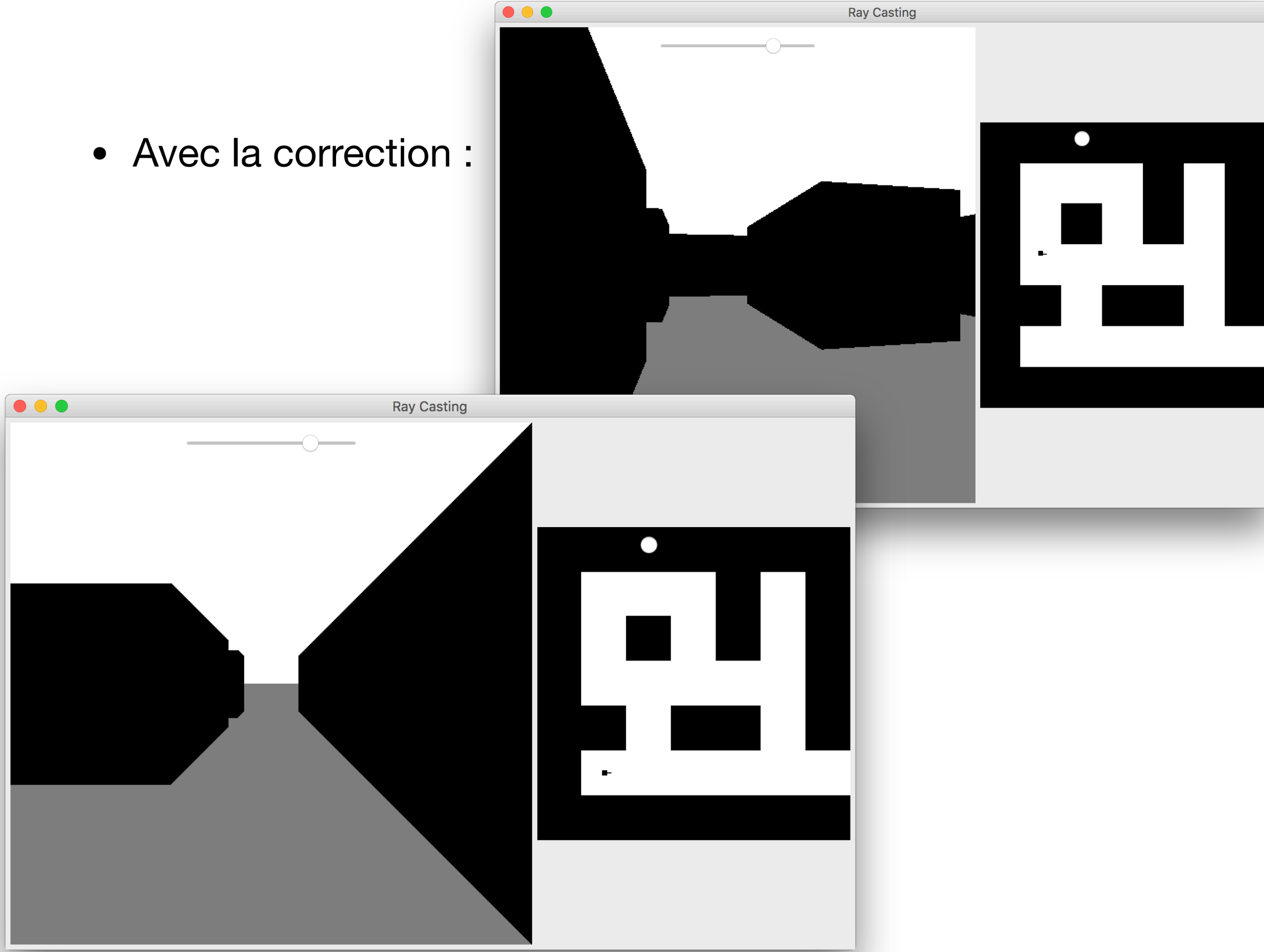


- Au fur et à mesure que l’on s’éloigne de la direction de vision l’incrément angulaire diminue (puisque les points eux sont également séparés
- La distance augmente et la hauteur projetée aussi.

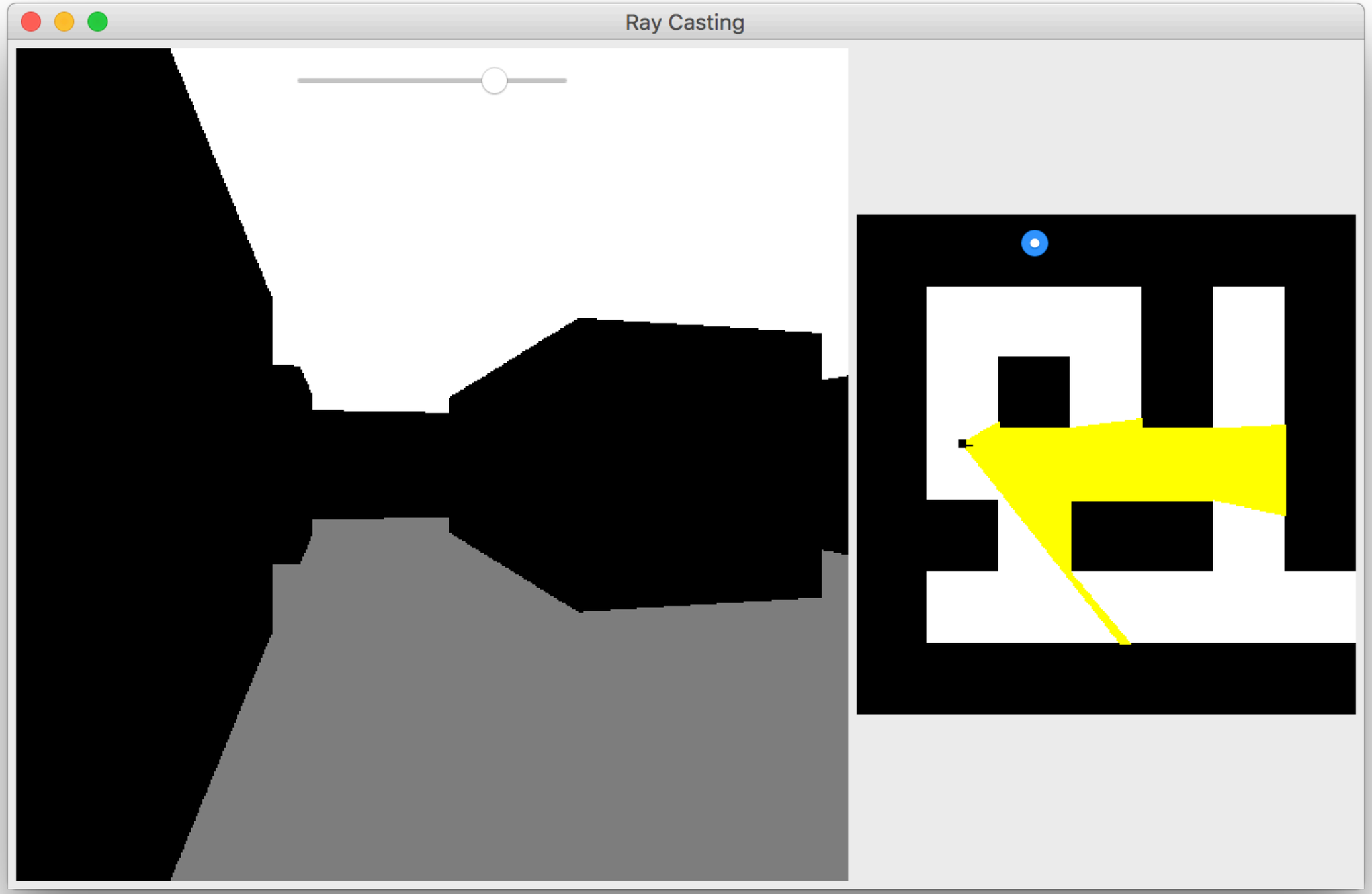
- Pourquoi les droites sont-elles courbes ?
  - Pour une droite perpendiculaire à la direction de vision, on obtient :
    - $d_i \cos(\boldsymbol{\alpha}_i) = d_0$
    - hors la hauteur projetée est calculée par
      $h_p = f.h/d_i$
      donc $h_p = f.h.d_0/\cos(\boldsymbol{\alpha}_i)$
  - Il faut donc opérer la correction optique correspondante
    $h_{pcorrected} = h_p * \cos(\boldsymbol{\alpha}_i)$



- Avec la correction :



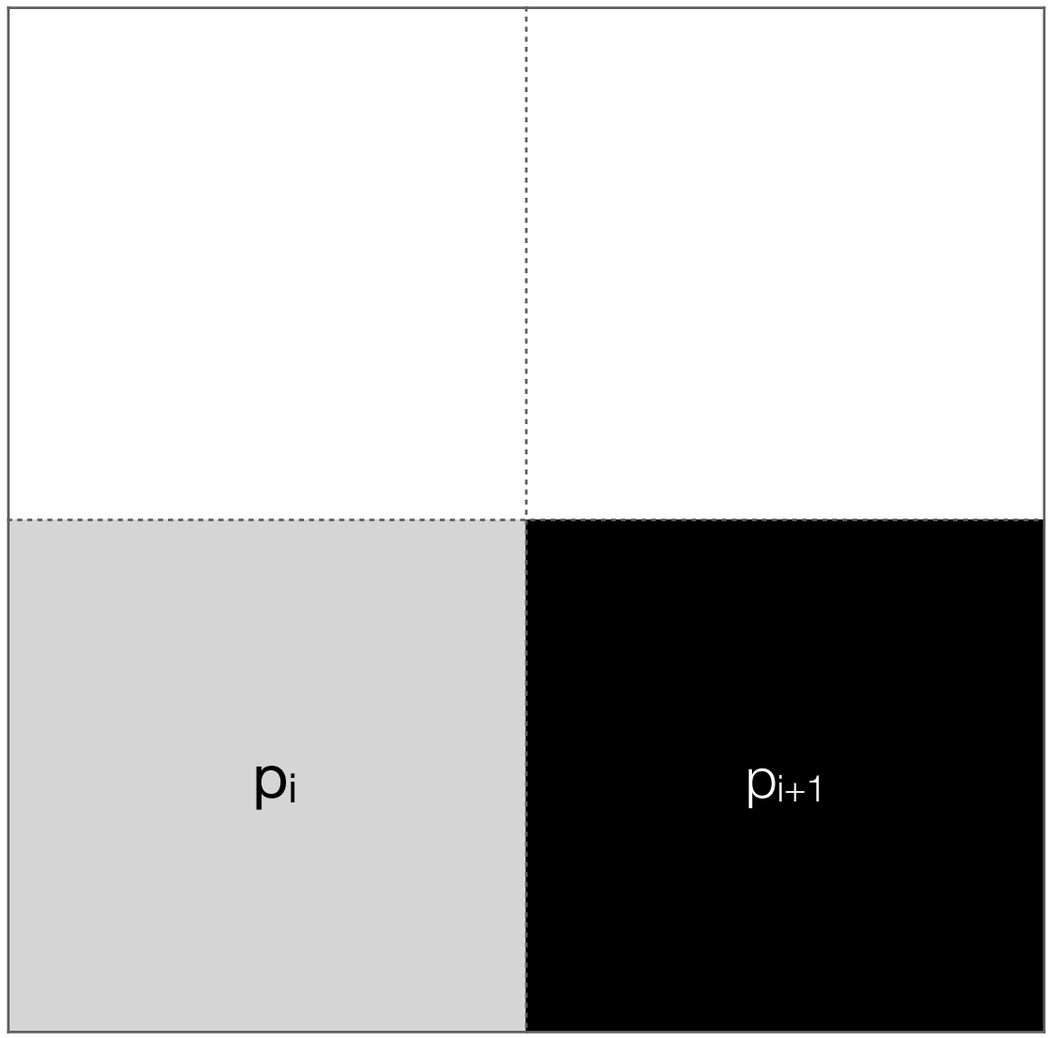
- On a suffisamment de données pour visualiser les parcours des rayons lancés

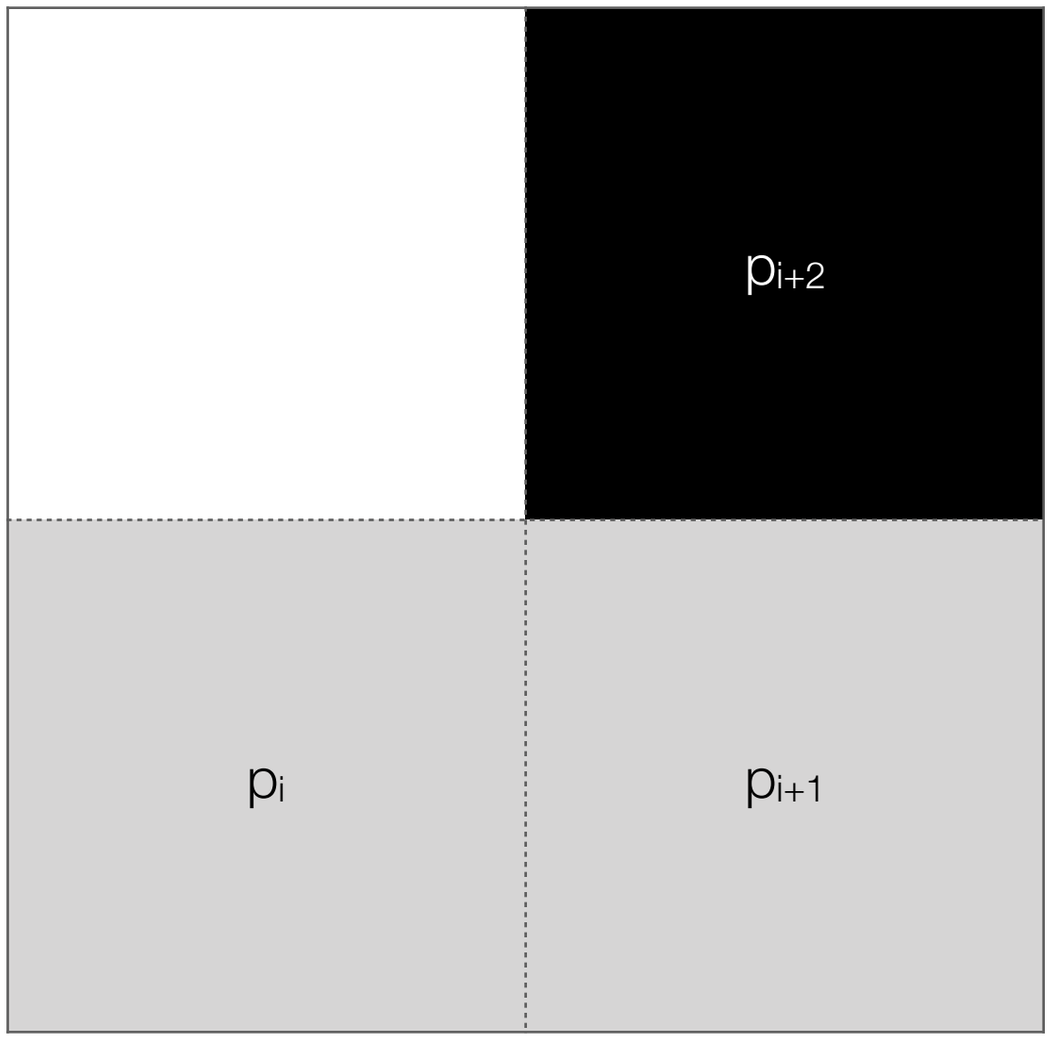


- On peut rajouter
  - des murs colorés
  - des effets lumineux, avec la normale à la surface si l’on souhaite (diffusion, spéculaire)
  - etc.

- La dernière amélioration possible serait d’utiliser Brensenham
    - puisque Bresenham permet d’obtenir le tracé d’un point à un autre et utilise essentiellement
        - un point de départ
        - une pente
- Une petite analyse permet de comprendre comment utiliser Brensenham pour déterminer quelle surface des briques sont atteintes lors du lancer

- Deux cas





- donc en conservant le point précédent, lorsqu'on atteint une brique on sait si l'on entre horizontalement ou verticalement.

- on a donc toutes les informations nécessaires