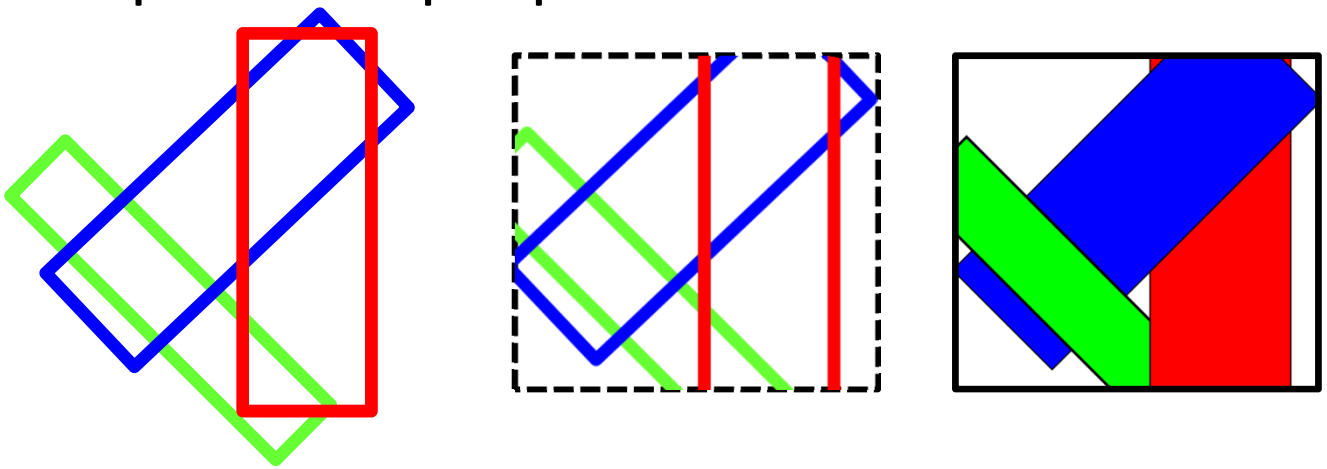
IFT3355: Infographie
Sujet 4: Visibilité 1 (z-buffer)

Derek Nowrouzezahrai
Département d'informatique et de recherche opérationelle
Université de Montréal

Université de Montréal

Visibilité

1. Pour créer une image, on doit premièrement effectuer la projection et le clippage
2. Pour une scène constituée d'objets **opaques** et ***non-réfléchissants***, on doit ensuite déterminer ce qui est à l'avant pour chaque pixel



3. Finalement, on doit afficher l'élément de l'image avec la bonne couleur

Visibilité

- Déterminer les surfaces visibles ou éliminer les surfaces cachées

- Il existe deux classes d'algorithmes pour résoudre le problème de la visibilité:
  - Algorithmes en espace image
  - Algorithmes en espace objet

Algorithmes en espace image

- Détermine pour chaque élément d'image (de *p* pixels), l'objet (de *n* objets) qui est le plus près du centre de projection tout en étant dans le volume de vision
- O(*n p*)

- Ex: 100 x (640x480) = 30,720,000 opérations

Université de Montréal

Algorithmes en espace image

- Précision dépend de la résolution de l'image
- La visibilité doit être recalculée à chaque changement de résolution et/ou de positionnement de la fenêtre

+ Les algorithmes et leurs structures sont habituellement plus simples

Ex: tampon de profondeur, balayage de lignes, subdivision, lancer de rayons

Algorithmes en espace objet

- Détermine entre chaque objet lequel est devant l'autre étant donnée la direction de projection
- O(n^2)

- Ex: 100 x 100 = 10,000 opérations

Université de Montréal

Algorithmes en espace objet

+ Précision dépend de la résolution des objets
+ La visibilité est indépendante de la résolution de l'image et peut même être indépendante de la position de la fenêtre
- Les algorithmes et leurs structures sont habituellement assez complexes

• Ex: algorithme du peintre, octree, arbres BSP

Université de Montréal

Cohérences

- Hiérarchies en 3D
- Face polygonale (si aucune inter-pénétration)
- Changements seulement aux segments
- Faces orientées vers l'arrière *(backface culling)*
- Similarités entre lignes de balayage adjacentes
- Similarités entre régions adjacentes de l'espace
- Similarités de profondeur pour des parties d'un même objet
- Similarités dans une séquence d'images

Tampon de profondeur *(z-buffer)*

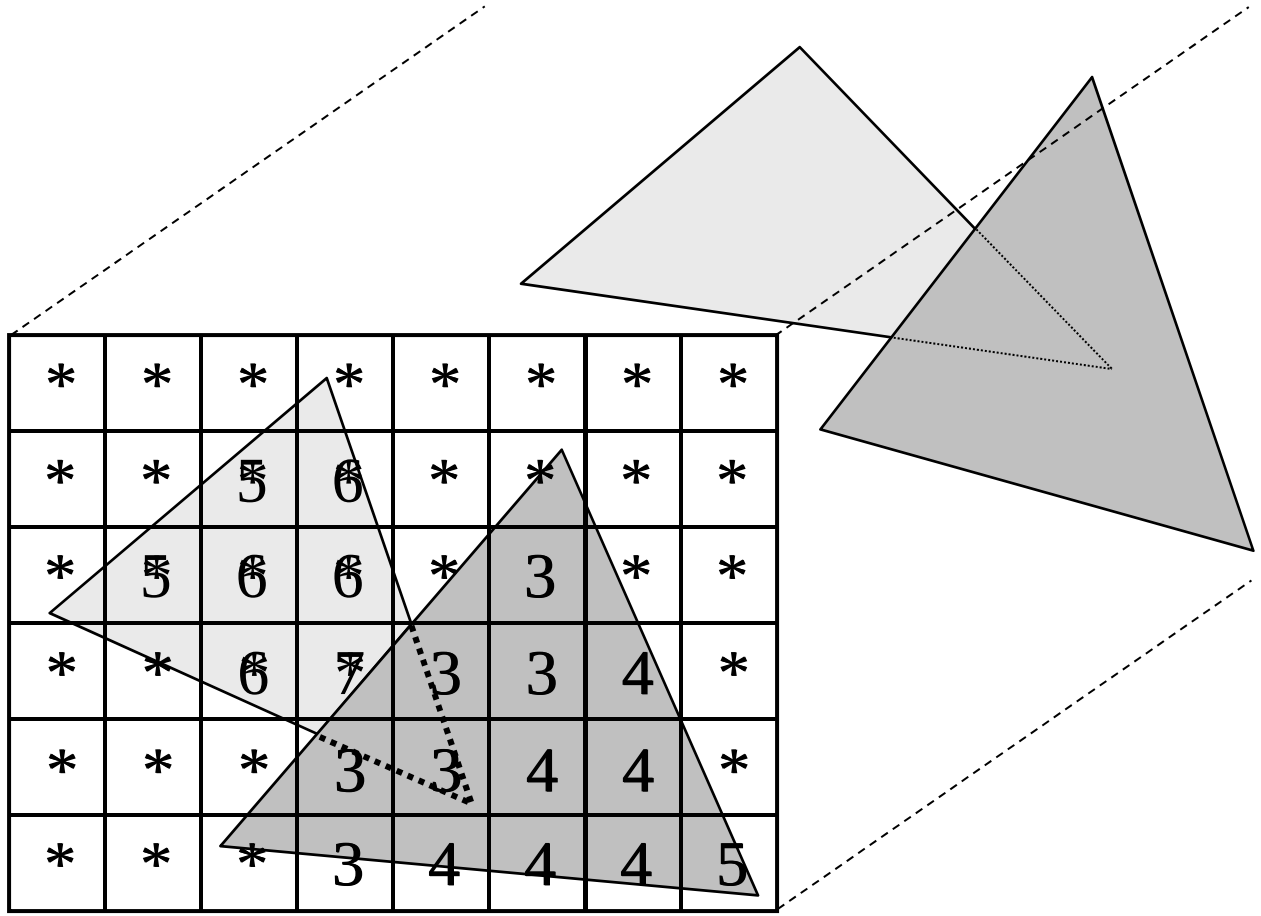
- Initialise tous les pixels

```
rgb(x,y) = couleur arrière-plan
z(x,y) = distance arrière-plan
```

- Pour chaque objet, scan-conversion dans le tampon

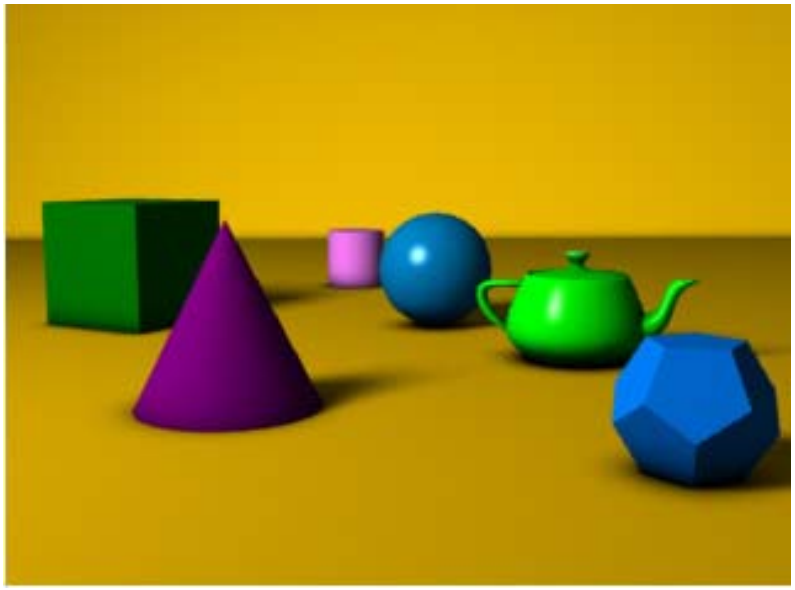
```
Pour chaque pixel (x,y) couvert par l'objet
   Si distance-objet (x,y) < z(x,y)
      z(x,y) = distance-objet (x,y)
      rgb(x,y) = couleur objet
```

Tampon de profondeur *(z-buffer)*



Université de Montréal

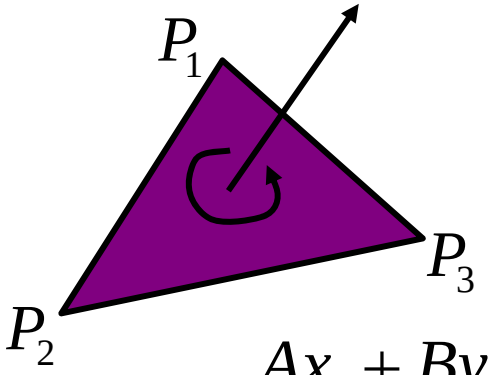
Tampon de profondeur *(z-buffer)*



Université de Montréal

Tampon de profondeur *(z-buffer)*

- *Scan-conversion*: remplissage par balayage
- Profondeur:



$$\vec{N} = \frac{(P_2 - P_1) \times (P_3 - P_1)}{\left\| (P_2 - P_1) \times (P_3 - P_1) \right\|} = (A, B, C)$$

$$Ax_1 + By_1 + Cz_1 + D = 0 \quad \Rightarrow \quad D = -Ax_1 - By_1 - Cz_1$$

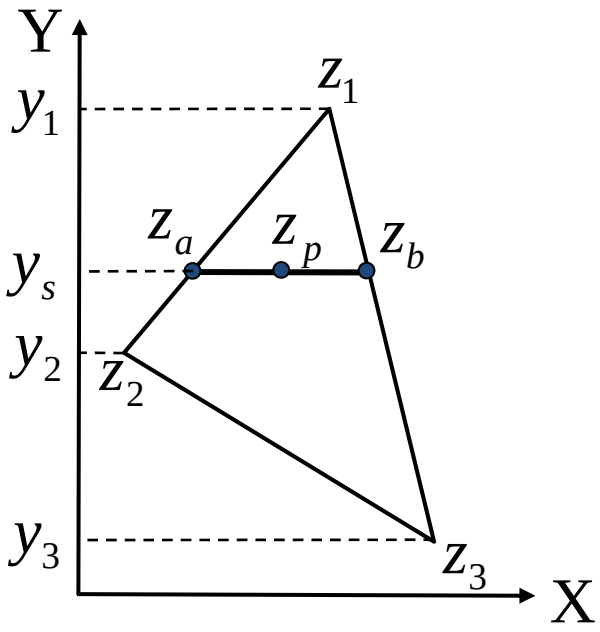
$$z(x, y) = \frac{-D - Ax - By}{C}$$

$$z(x + \Delta x, y) = \frac{-D - A(x + \Delta x) - By}{C} = z(x, y) - \frac{A\Delta x}{C} \left.\begin{array}{l} \Delta x = 1 \\ A/C = const \end{array}\right\}$$

$$z(x, y + \Delta y) = \frac{-D - Ax - B(y + \Delta y)}{C} = z(x, y) - \frac{B\Delta y}{C} \left.\begin{array}{l} \Delta y = 1 \\ B/C = const \end{array}\right\}$$

Université de Montréal

Interpolation bilinéaire (alternative)



$$y_s = y_1 + t(y_2 - y_1)$$

$$z_a = z_1 + t(z_2 - z_1)$$

$$\frac{z_a - z_1}{z_2 - z_1} = \frac{y_s - y_1}{y_2 - y_1}$$

$$z_a = z_1 + (z_2 - z_1)\frac{(y_s - y_1)}{(y_2 - y_1)}$$

$$z_b = z_1 + (z_3 - z_1)\frac{(y_s - y_1)}{(y_3 - y_1)}$$

Interpole aussi:

couleur, texture, normale, etc.

$$z_p = z_a + (z_b - z_a)\frac{(x_p - x_a)}{(x_b - x_a)}$$

Université de Montréal

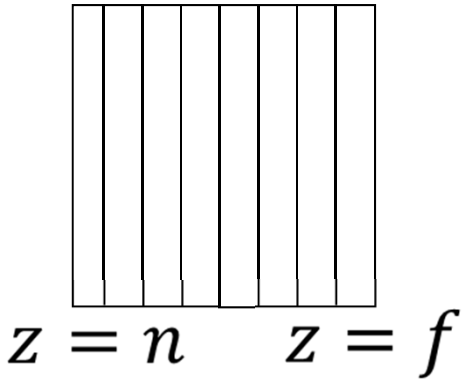
Tampon de profondeur *(z-buffer)*

Avantages/désavantages:

+ simplicité, généralité
+ hardware, parallélisme
- mémoire additionnelle pour les tampons
+ aucune mémoire pour conserver la scène
- aliassage (en XY et en Z)

Z-fighting

$\Delta z = (f - n)/2^b$ pour b bits



$$\begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -fn \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

$$z = n + f - \frac{fn}{z_w}$$

$$\Delta z \approx \frac{fn\Delta z_w}{z_w^2}$$

$$\Delta z_w \approx \frac{z_w^2 \Delta z}{fn}$$ ← taille des bins en espace monde

$$\Delta z_w^{\max} \approx \frac{f\Delta z}{n} \text{ lorsque } z_w = f$$

Donc rendre $\Delta z_w^{\max}$ aussi petit que possible correspond à minimiser f et maximiser n

Université de Montréal