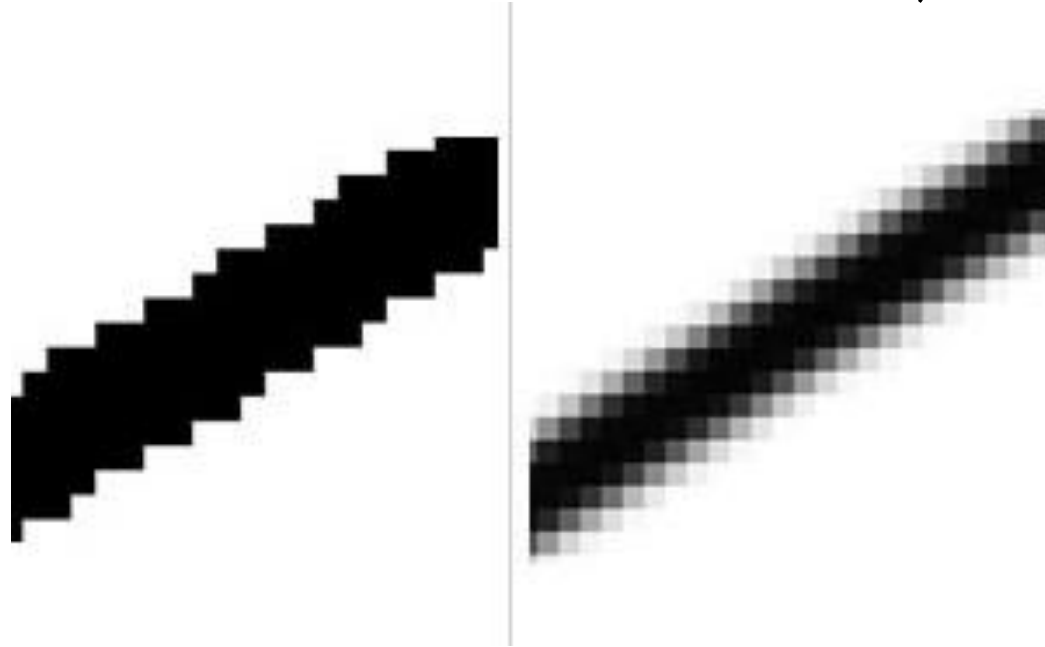


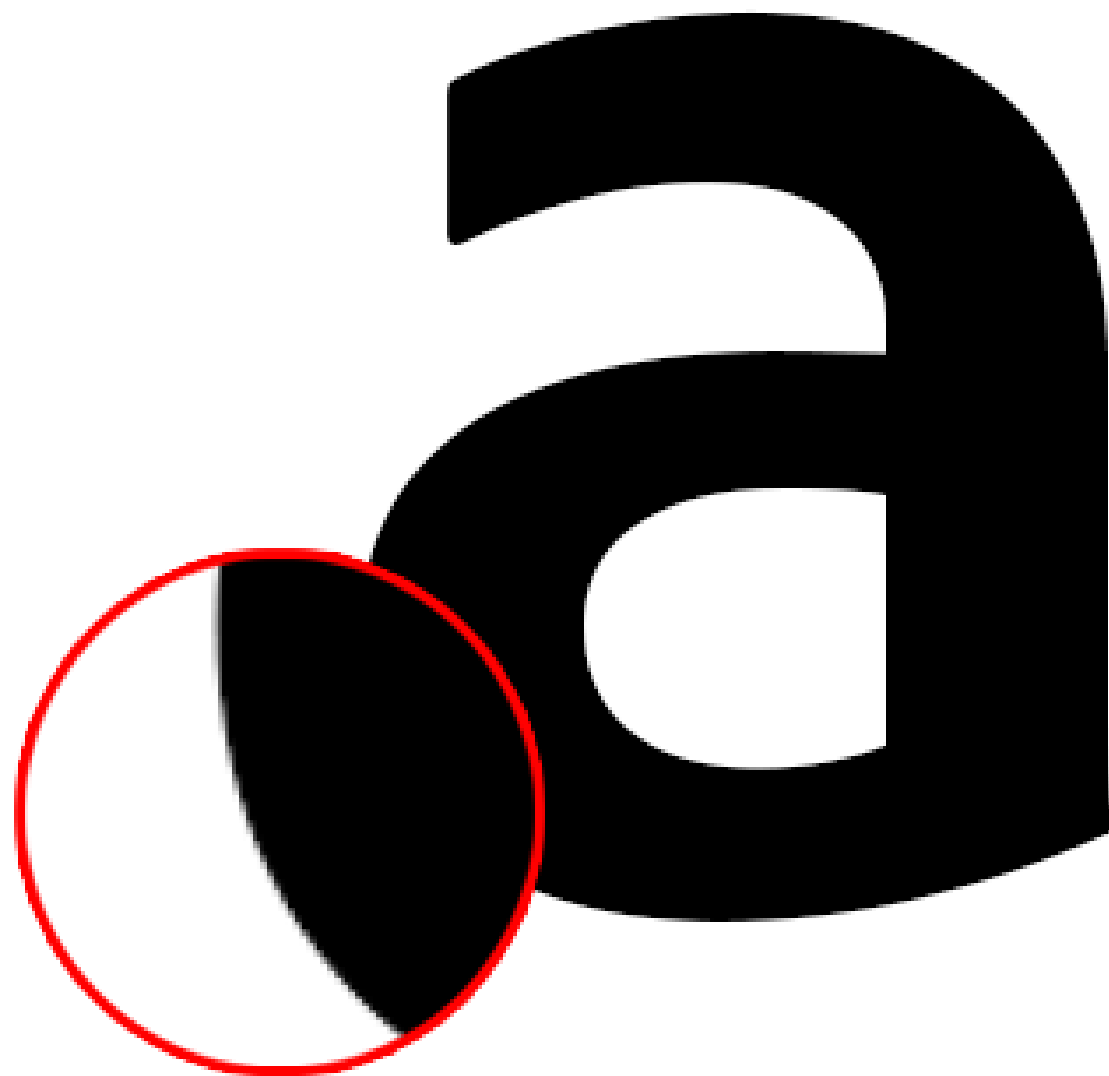
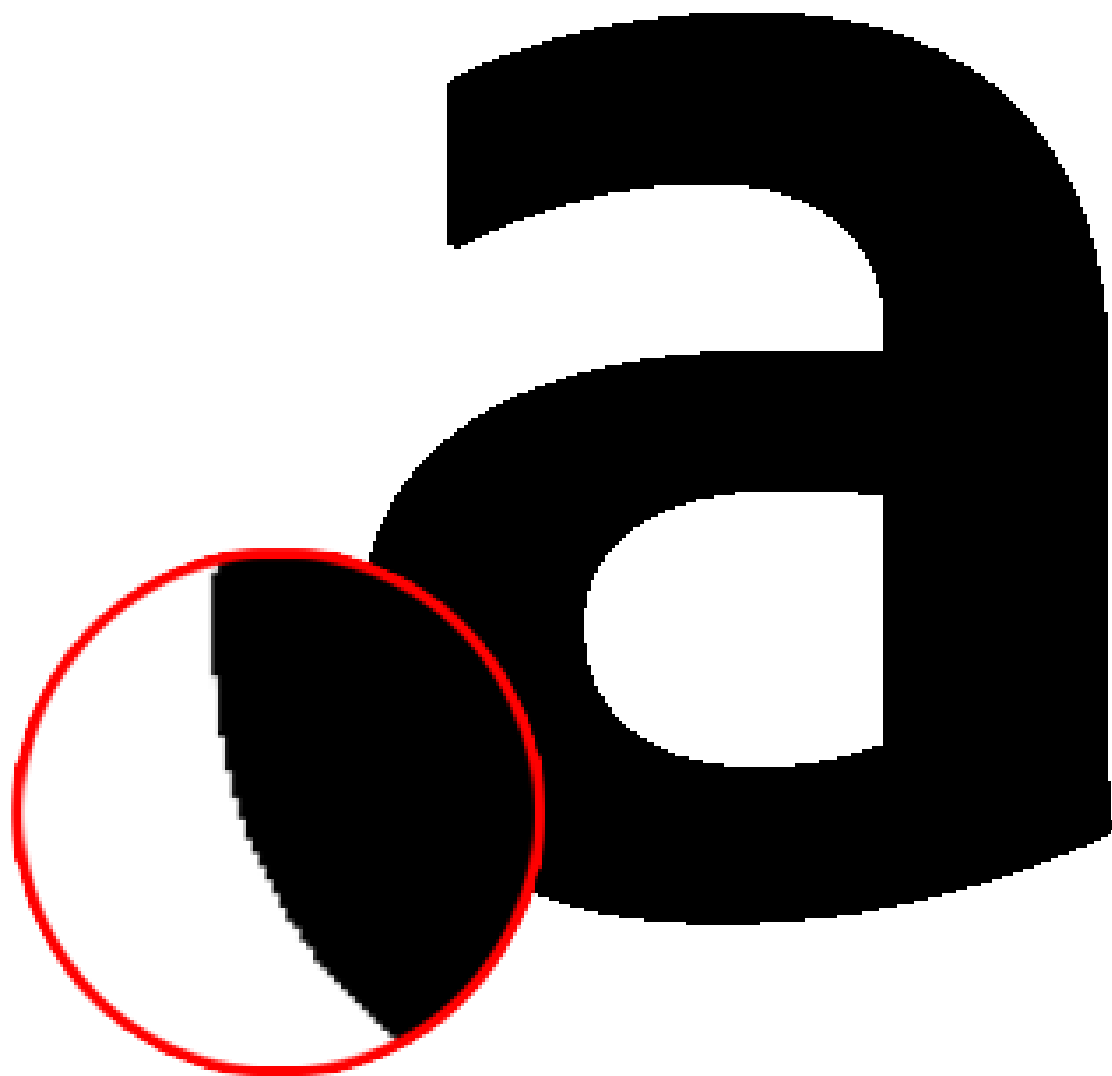
IFT 3355: INFOGRAPHIE ALIASAGE

Livre de référence: G: 16, S: 4

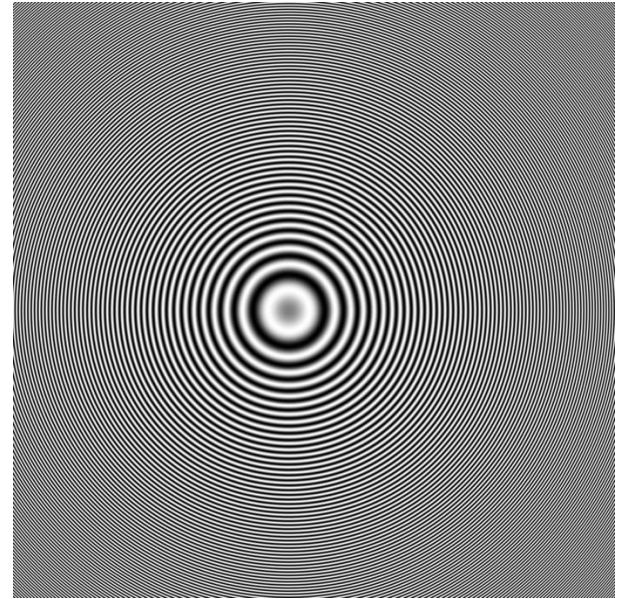
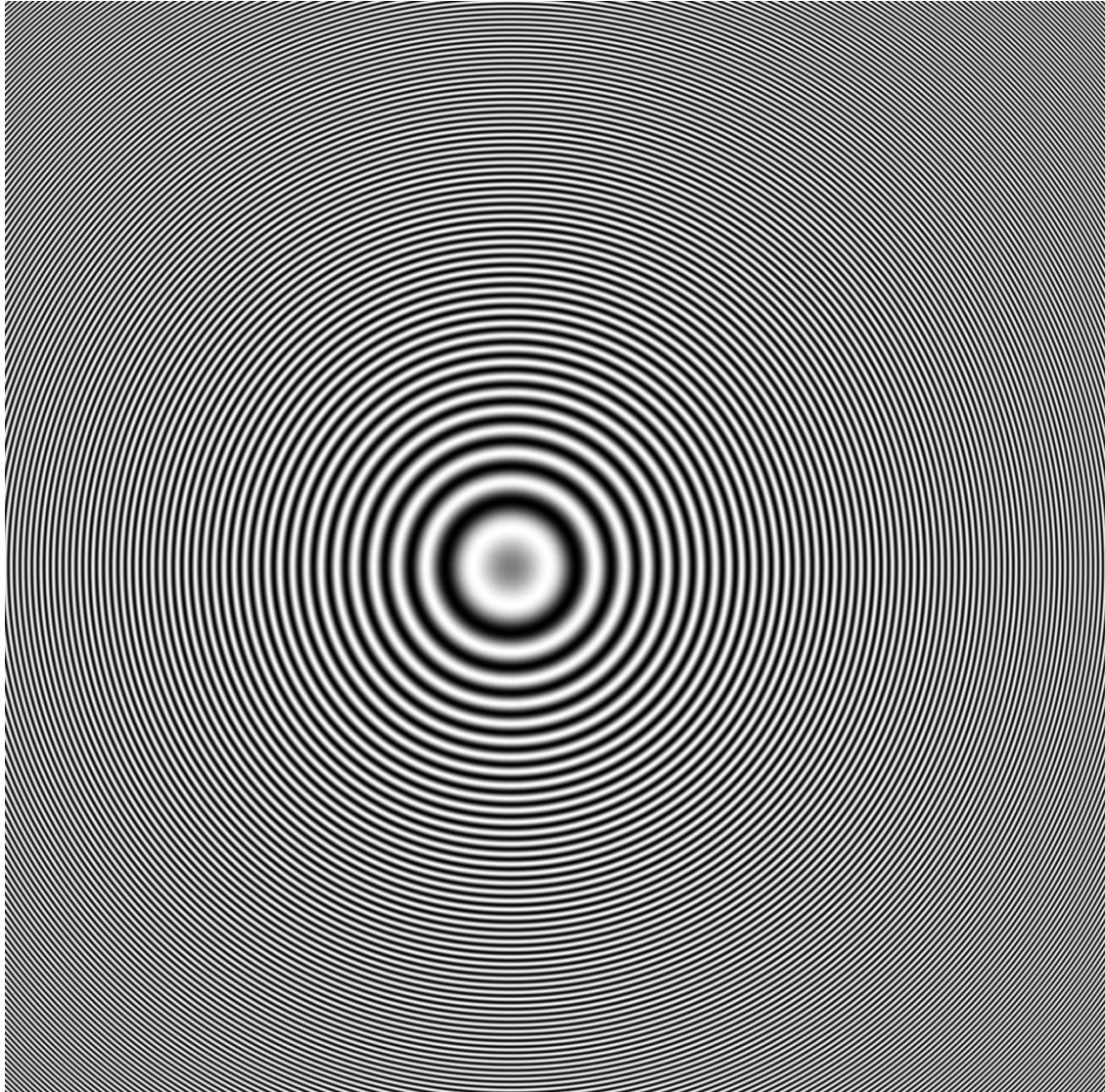


<http://tiny.cc/ift3355>
Mikhail Bessmeltsev

ALIASING & ANTI-ALIASING



ALIASSEGE: MOIRÉS

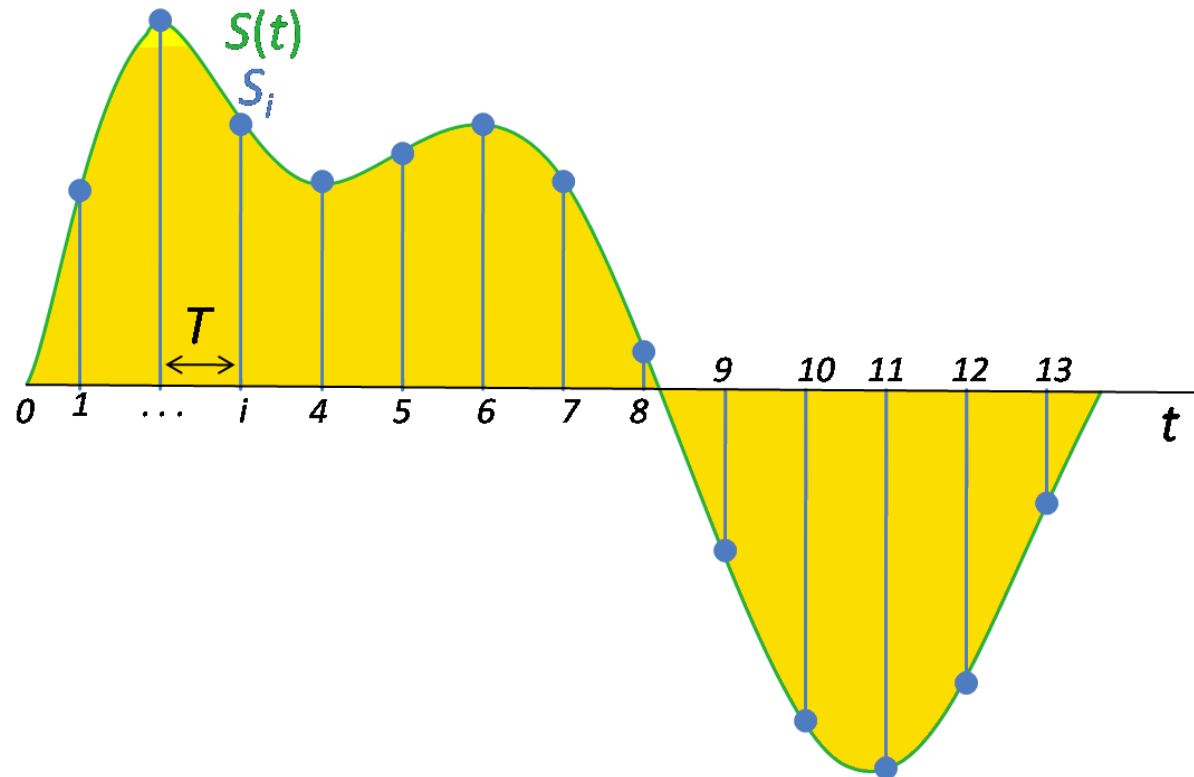


DEUX INTERPRÉTATIONS D'IMAGES

- Une *image continue* $I(x, y)$ est une fonction de deux variables
 - L'image de la fonction est dans un espace linéaire de couleurs
- Une *image discrète* $I[i][j]$ est un tableau de valeurs
- On associe chaque paire de i, j aux coordonnées continues dans image ($x = i, y = j$)

CONTINU VS DISCRET

- Continu $\rightarrow$ Discret: *Échantillonnage*
- Discret $\rightarrow$ Continu: *Reconstruction / Interpolation*



ÉCHANTILLONNAGE

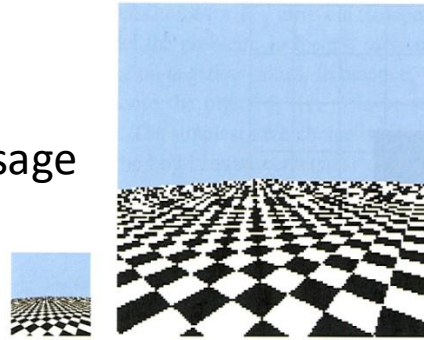
- La méthode la plus simple est l'échantillonnage ponctuel
- Pour obtenir la valeur i, j , on échantillonne l'image à un seul endroit avec les coordonnées entières

$$I[i][j] \leftarrow I(i, j)$$

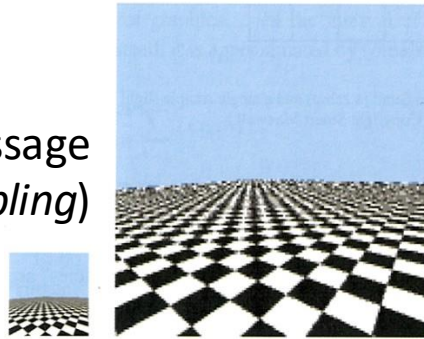
- Ça peut entraîner des artefacts

ALIASAGE ET ANTI-ALIASAGE

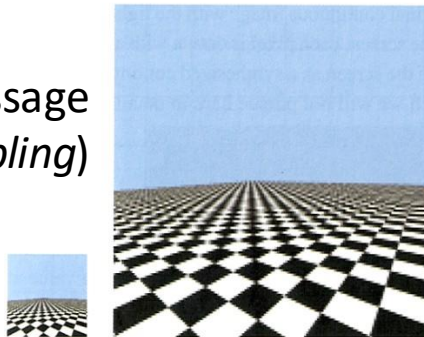
Aliassage



Anti-aliasage
(*multi-sampling*)

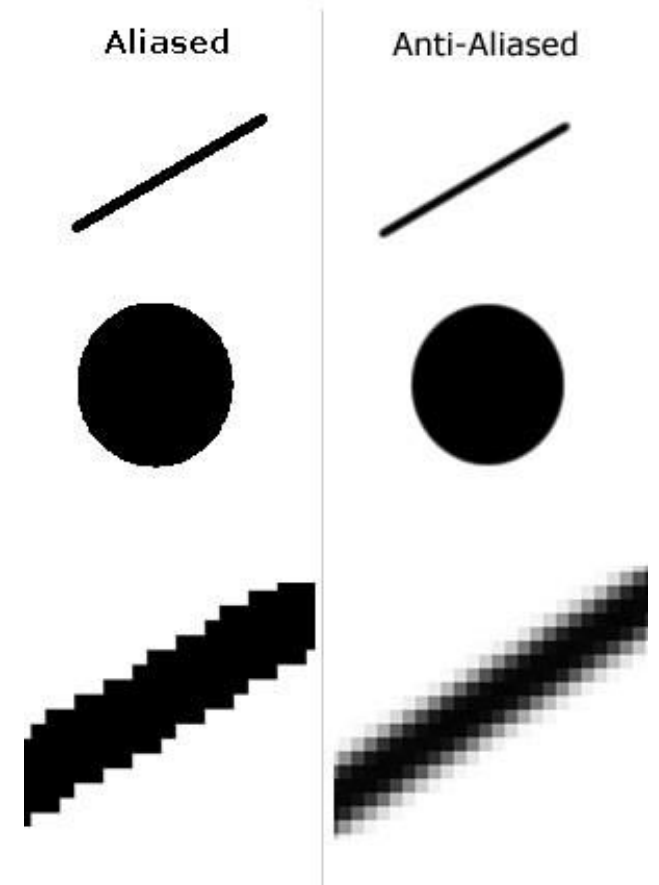


Anti-aliasage
(*super-sampling*)



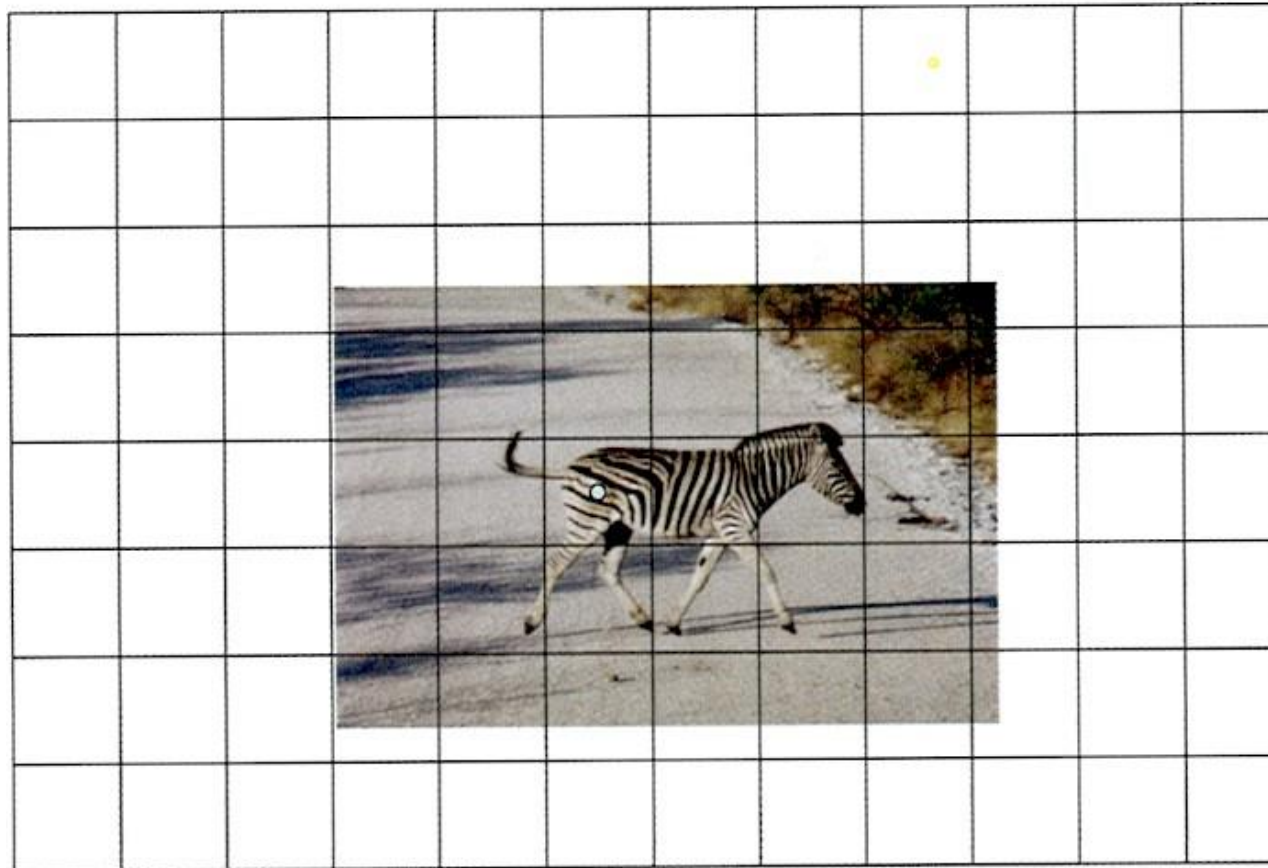
ALIASAGE

- Pour les objets à contrastes élevés: *jaggies* sur la frontière
 - Ils vont glisser pendant l'animation
- Si les triangles sont très petits, on obtient des pixels clignotants apparemment aléatoires
 - Encore, pire pour l'animation



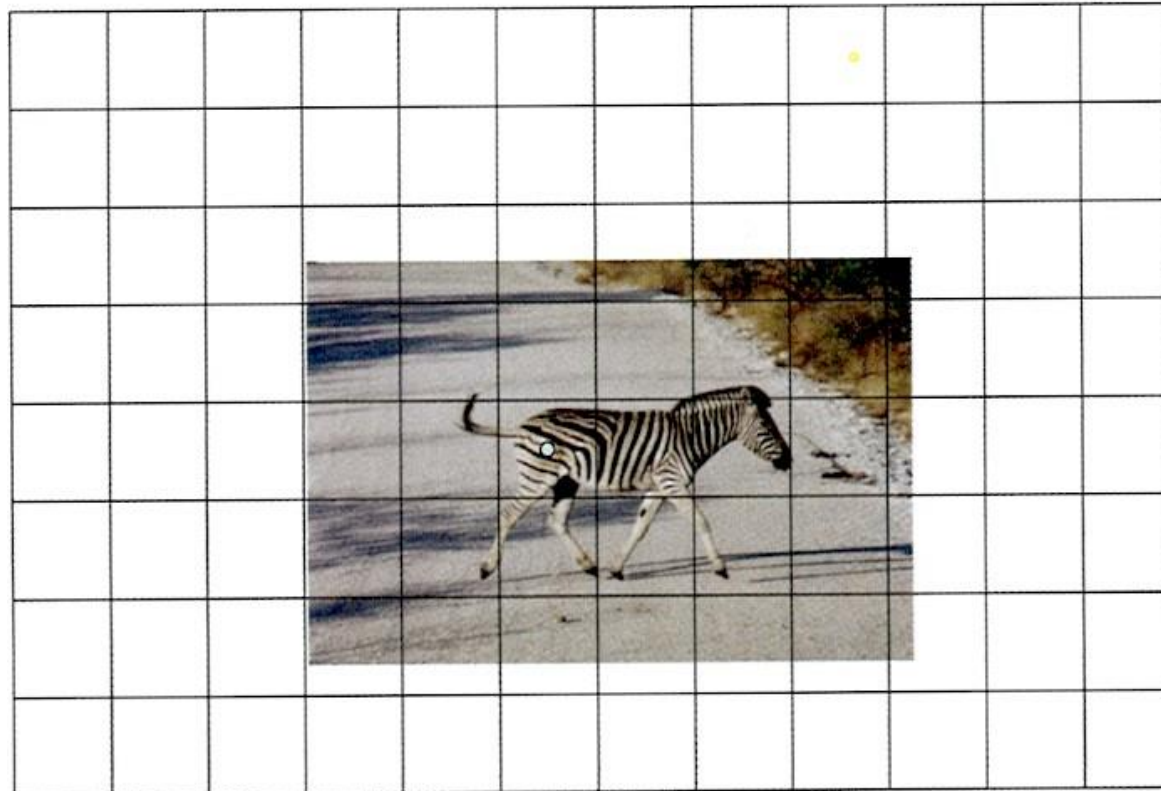
ALIASAGE

- Le cœur du problème: un signal de haute résolution affiché sur un dispositif de basse résolution



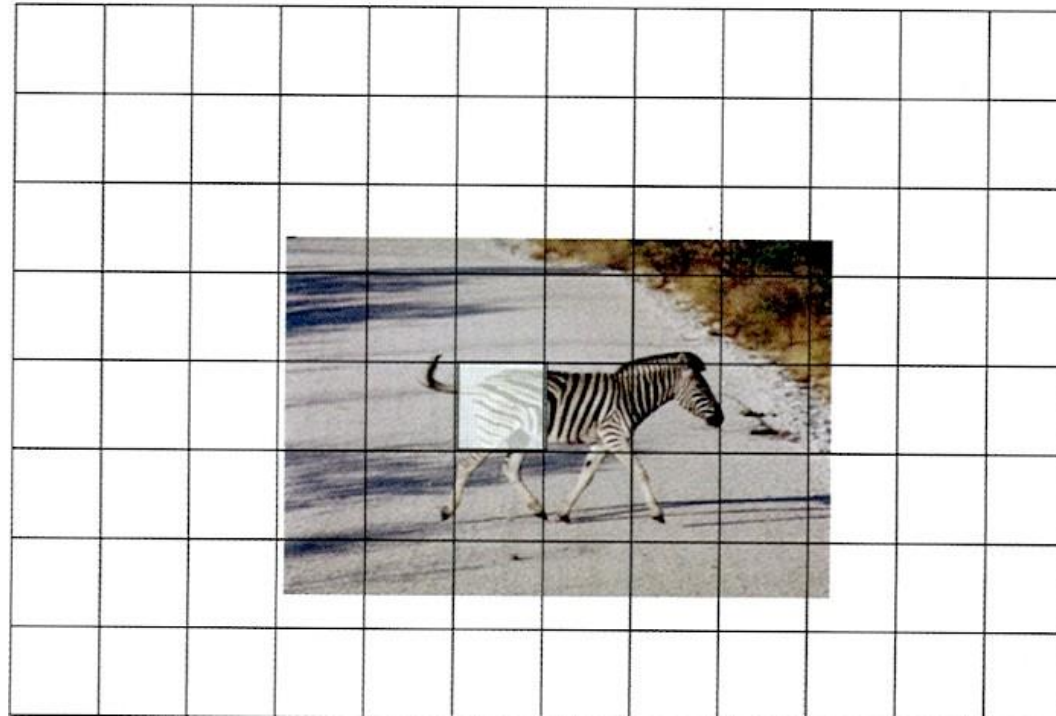
ALIASAGE

- Le cœur du problème: un signal de haute résolution affiché sur un dispositif de basse résolution
- Trop information pour un seul pixel!



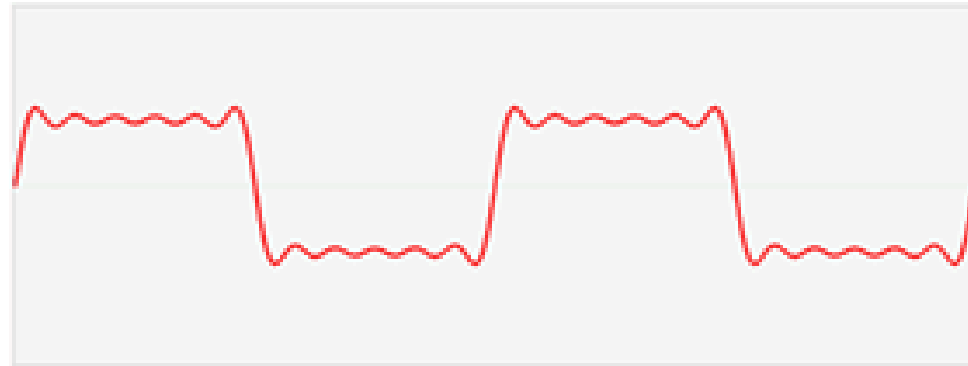
ANTI-ALIASING

- Une intuition:
 - La valeur d'un seul point ne représente pas toutes les valeurs dans le pixel
 - On doit calculer une certaine valeur générale, par exemple, moyenne
- Pour l'exemple ci-dessous, peut-être gris?



ANTI-ALIASSAGE

- Mathématiquement, on peut le modéliser en utilisant *l'analyse de Fourier*
 - Ordonner les données par fréquence
- Puis, décider que faire avec les hautes fréquences non représentables

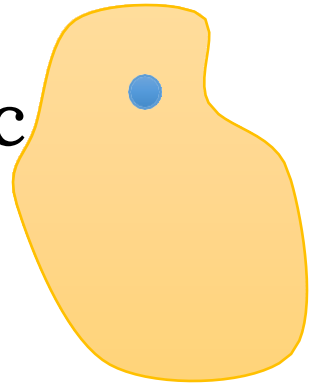


ANTI-ALIASAGE

- On peut le modéliser comme un problème d'optimisation
- Ces approches conduisent à:

$$\iint_{\blacksquare} I(x, y) F_{i,j}(x, y) dx dy$$

Où $F_{i,j}(x, y)$ est une fonction qui mesure l'ampleur avec laquelle la valeur de l'image continue à $[x, y]$ devrait influencer la valeur à (i, j)

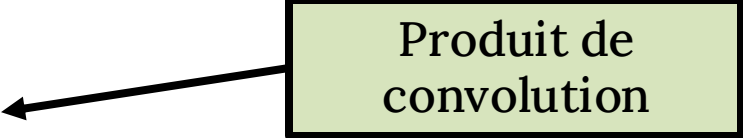


ANTI-ALIASSAGE

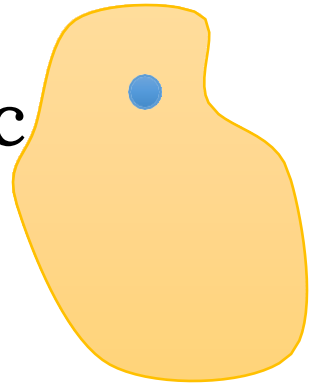
- On peut le modéliser comme un problème d'optimisation
- Ces approches conduisent à:

$$\iint_{\blacksquare} I(x, y) F_{i,j}(x, y) dx dy$$

Produit de
convolution



Où $F_{i,j}(x, y)$ est une fonction qui mesure l'ampleur avec laquelle la valeur de l'image continue à $[x, y]$ devrait influencer la valeur à (i, j)



ANTI-ALIASSAGE

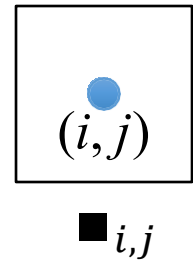
- Ici, la fonction $F_{i,j}(x, y)$ s'appelle *un filtre*
 - En d'autres termes, la meilleure valeur du pixel est déterminée par certaine moyenne pondérée continue près du pixel
 - En effet, cela brouille l'image avant de l'échantillonner

UN FILTRE BOITE

- On choisit souvent un filtre $F_{i,j}(x, y)$ non optimal, mais qui permet les calculs rapides
- Le choix le plus simple est un *filtre boite* où $F_{i,j}(x, y) = 0$ partout sauf dans un carré 1 par 1 centré en $x = i, y = j$

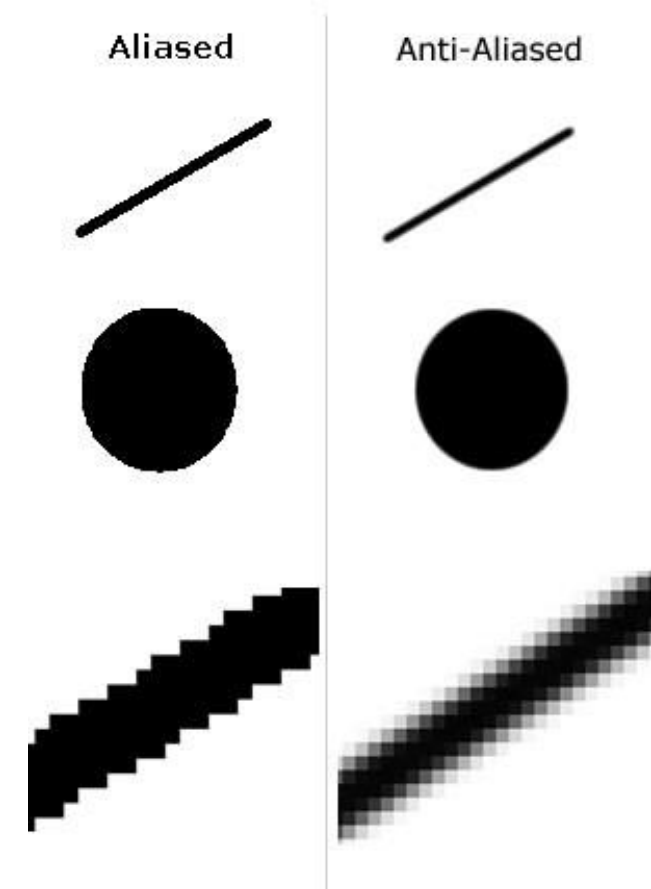
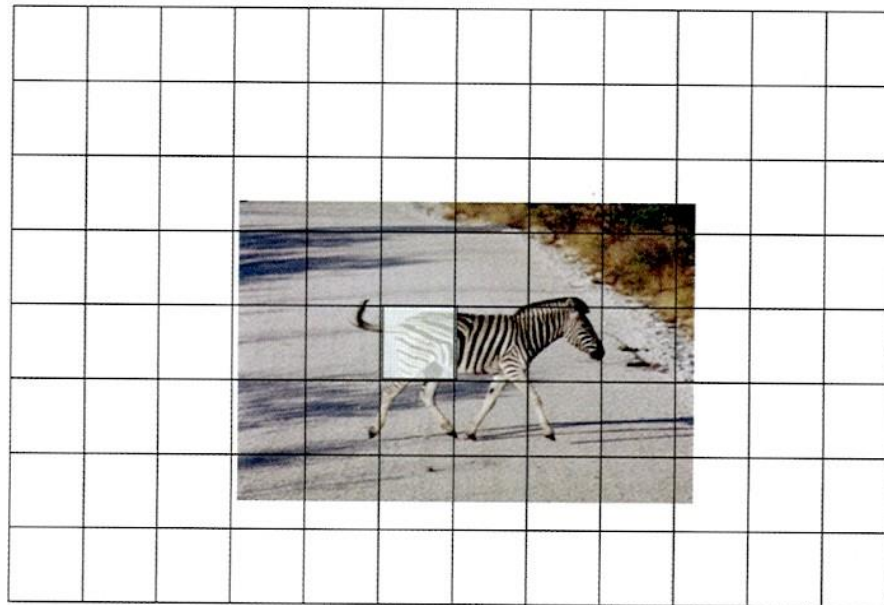
Soit $\blacksquare_{i,j}$ ce carré, puis:

$$I[i, j] = \iint_{\blacksquare_{i,j}} I(x, y) dx dy$$



UN FILTRE BOITE

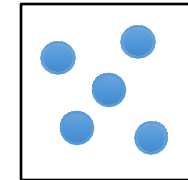
Dans ce cas, la valeur est simplement la moyenne de l'image continue sur le domaine carré du pixel



OVER-SAMPLING

- Même cette intégrale n'est pas facile à calculer
- Au lieu de cela, elle est approximée par une somme:

$$I[i, j] \leftarrow \frac{1}{n} \sum_{k=1}^n I(x_k, y_k)$$



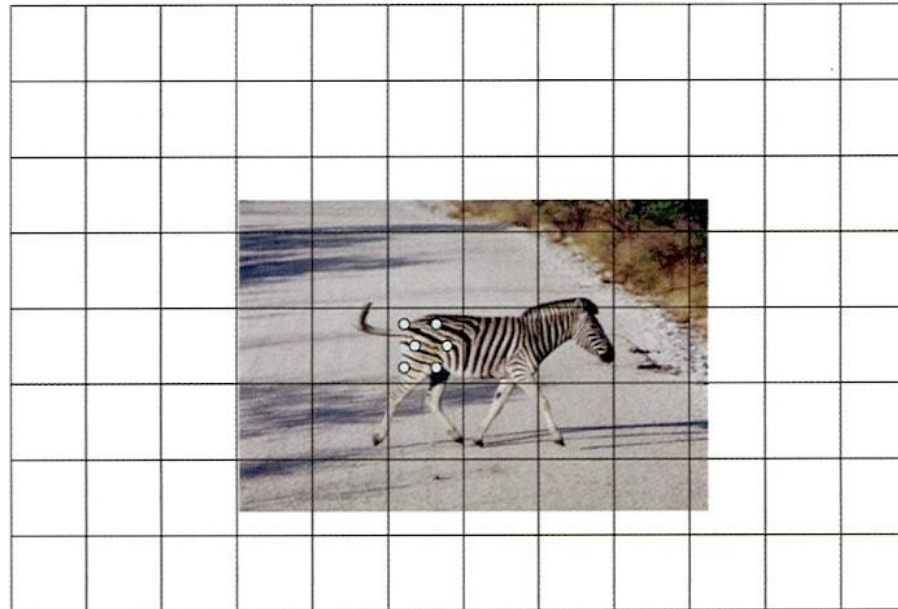
Où k indexe un ensemble de points (x_k, y_k) appelés les positions d'échantillons

Il faut rendre la version de haute résolution pour la couleur est la profondeur

- On utilise le mot «échantillon» pour les pixels des images de hautes résolutions

OVER-SAMPLING

- On calcule la moyenne de ces pixels de haute résolution



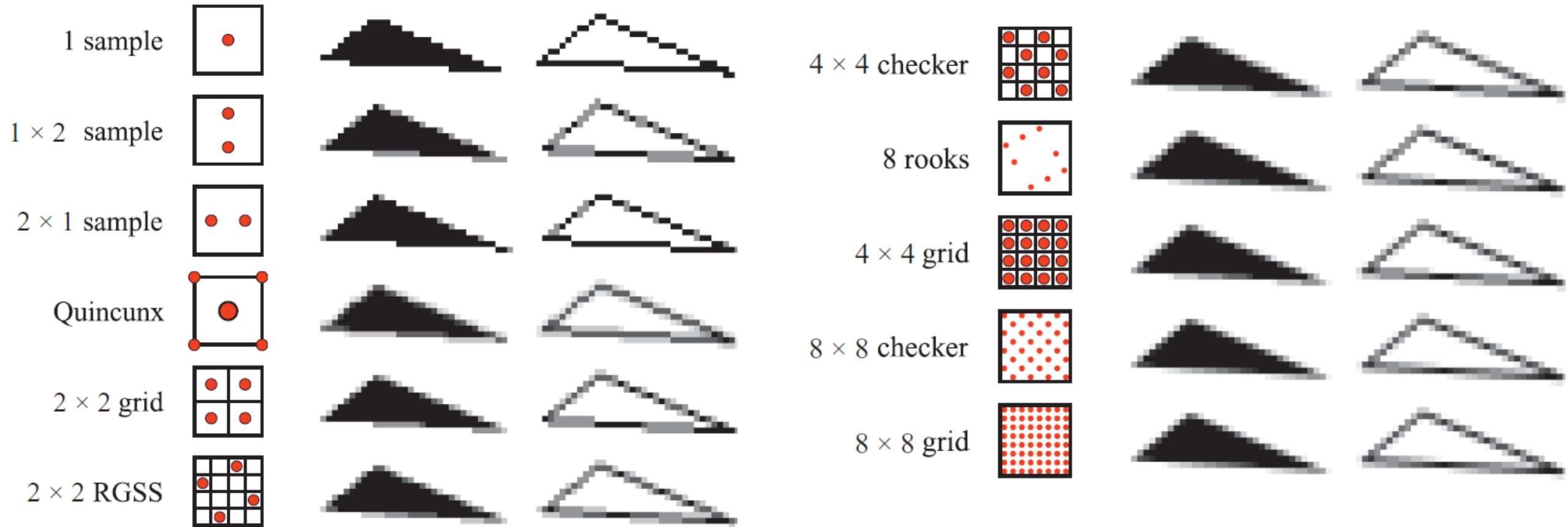
SUPERSAMPLING (SSAA)

- Si les positions d'échantillonnage sont sur grille régulière, c'est *super-sampling*
- = Rendre la scène complète à la résolution haute, puis réduire sa taille
- On peut utiliser les autres modèles des positions
 - Les modèles moins réguliers peuvent éviter les erreurs systématiques (biais)

SUPERSAMPLING (SSAA)

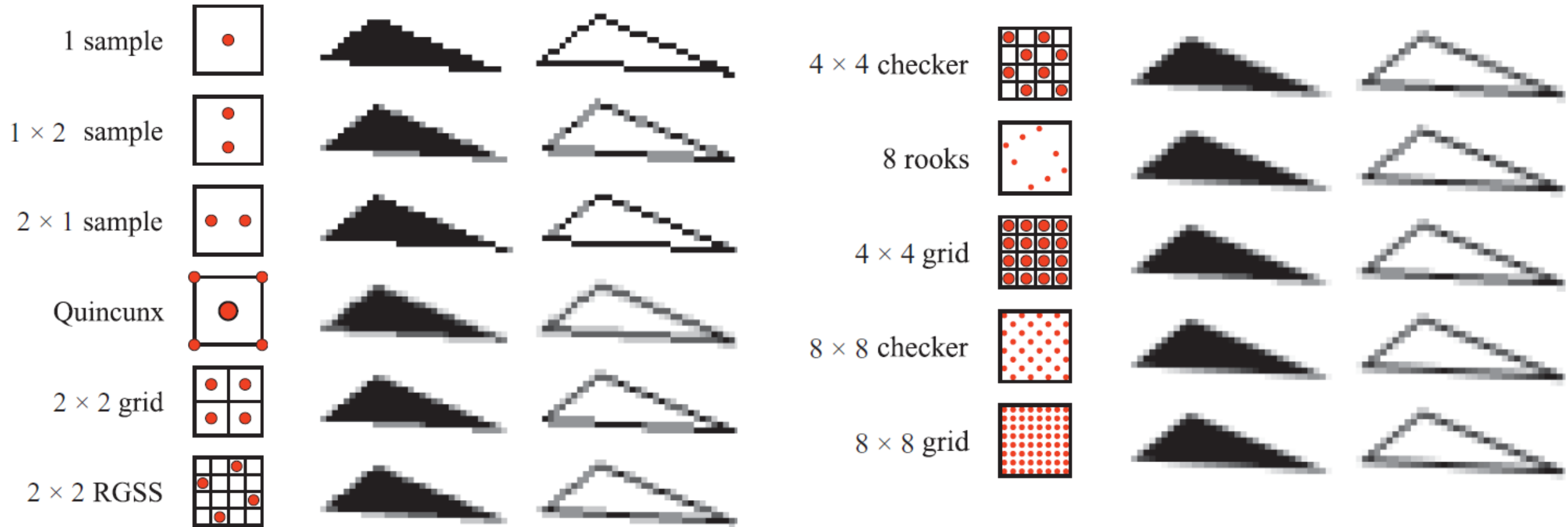
- Si les positions d'échantillonnage sont sur grille régulière, c'est *super-sampling*
- = Rendre la scène complète à la résolution haute, puis réduire sa taille
- Tellement cher: pipeline entier pour chaque pixel en haute résolution

SUPERSAMPLING (SSAA)



Real-Time Rendering, 3rd Edition, A K Peters 2008

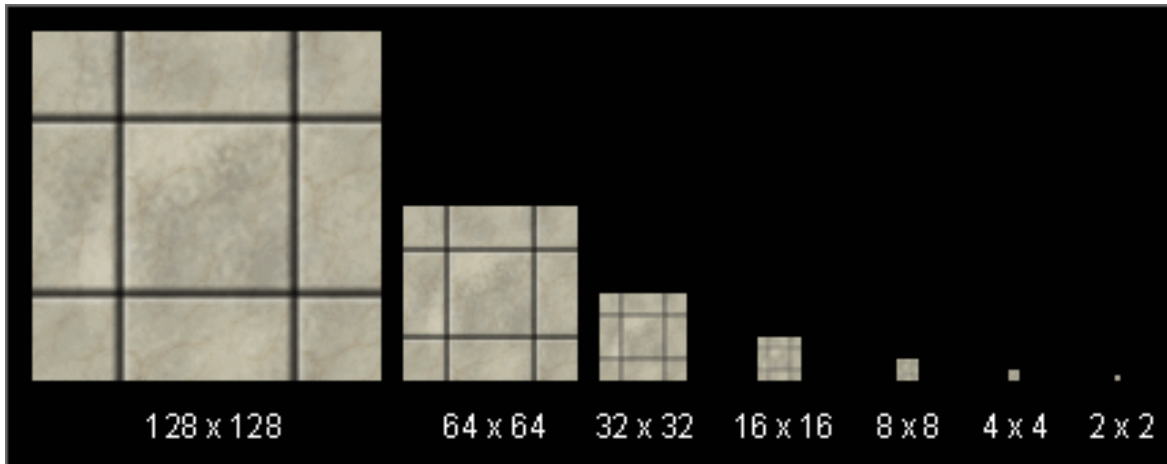
SUPERSAMPLING (SSAA)



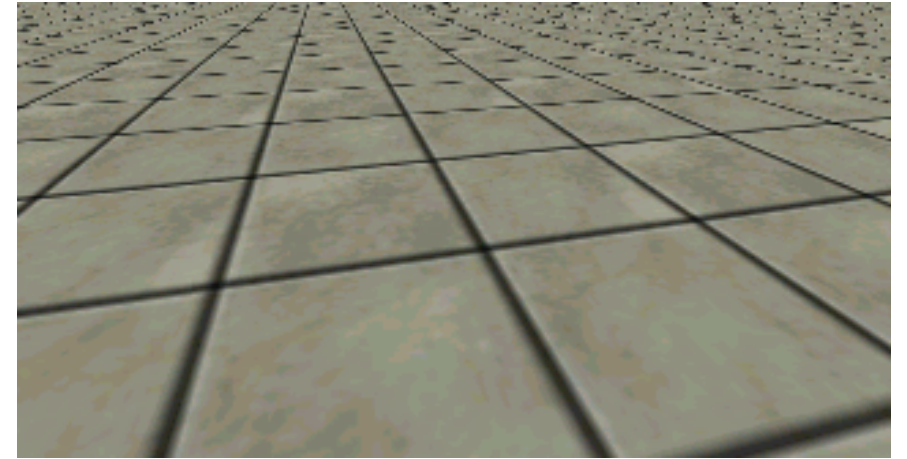
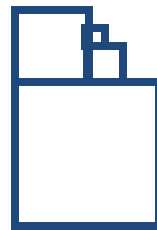
Real-Time Rendering, 3rd Edition, A K Peters 2008

MIPMAPPING

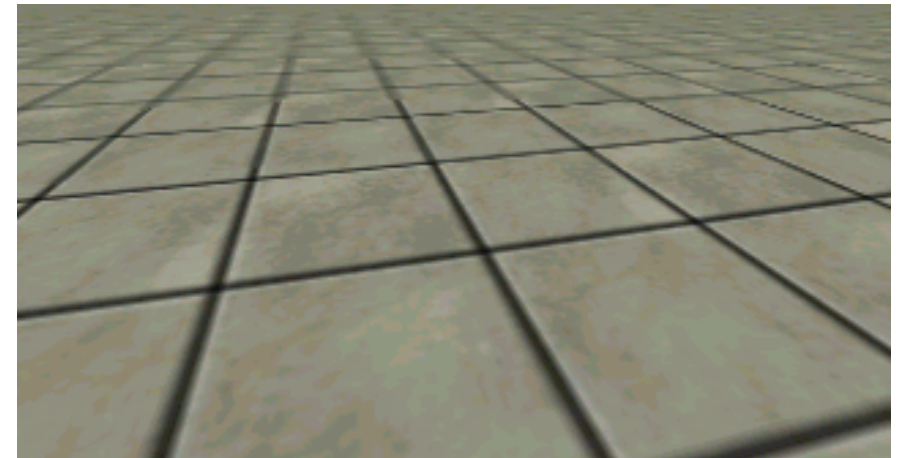
Utiliser “une pyramide d’images” pour
précalculer les versions moyennes d’une image



Stocker la pyramide
entière comme un seul
bloc de mémoire



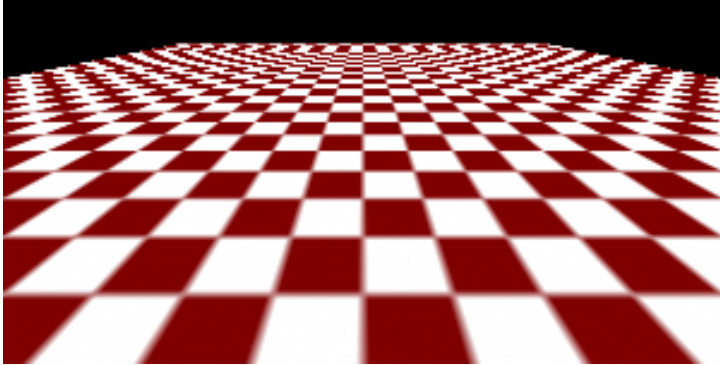
Sans MIP-mapping



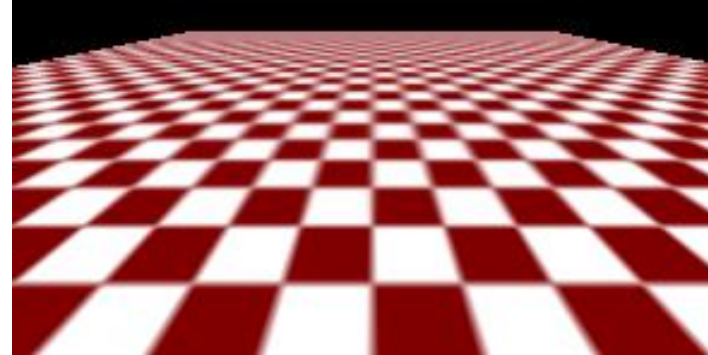
Avec MIP-mapping

MIPMAPPING

sans

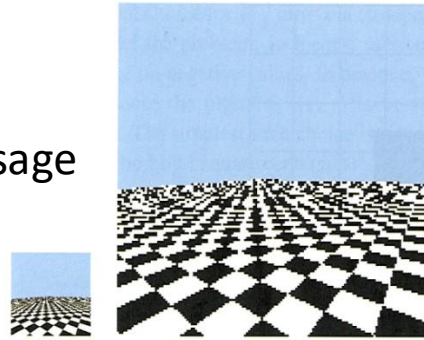


avec

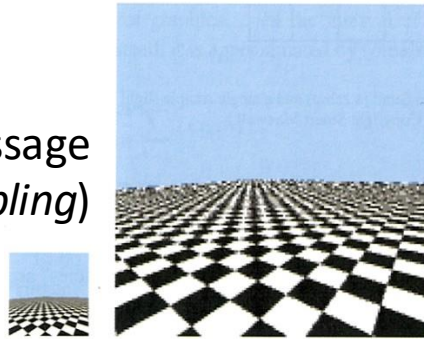


ALIASAGE ET ANTI-ALIASAGE

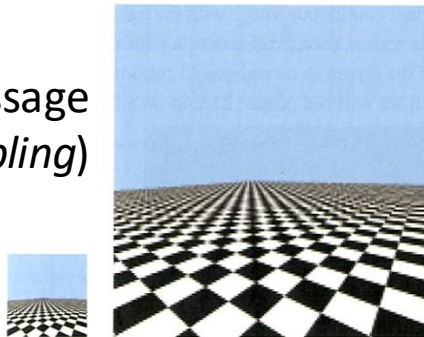
Aliassage



Anti-aliasage
(*multi-sampling*)



Anti-aliasage
(*super-sampling*)

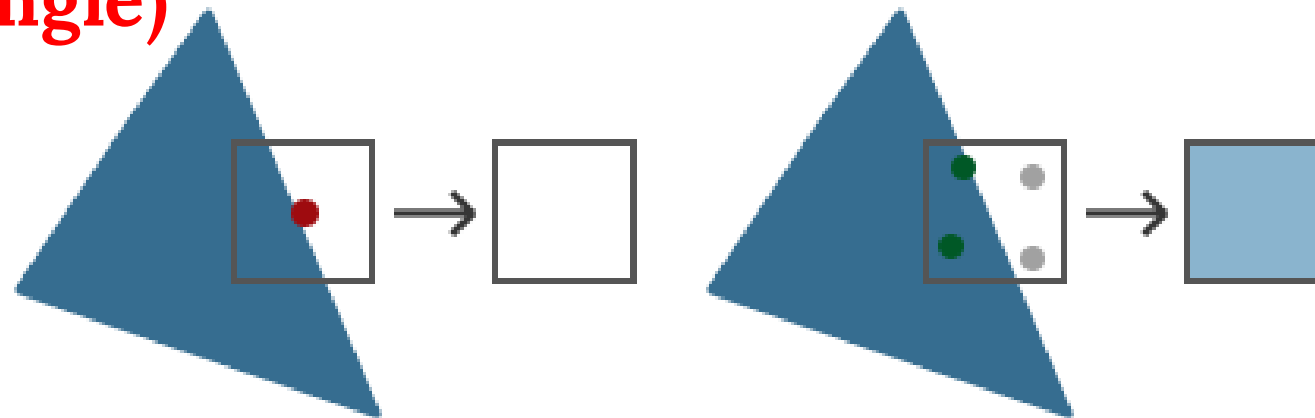


SUPERSAMPLING (SSAA)

- Comment accélérer?
- Observation:
 - Les couleurs dans chaque triangle sont déjà filtrées (mipmapping)
 - Aliassage est seulement sur les bords des triangles
- Idée:
 - Calculer « couverture » pour chaque échantillon (i.e., est-ce que cet échantillon est couvert par le triangle)
 - Faire le teste de profondeur par échantillon
 - MAIS exécuter le *fragment shader* 1 fois par fragment final

MULTI-SAMPLING (MSAA)

- Plus rapide que *supersampling*
- Faire le rendu en haute résolution (la couverture + profondeur)
- Pendant la rasterisation de chaque triangle, “la couverture” et les valeurs z sont calculées à ce niveau
- (efficacité) Mais le *fragment shader* est exécuté **une fois par pixel final (par triangle)**



MULTI-SAMPLING

- Faire le rendu en haute résolution (la couverture + profondeur)
- Pendant la rasterisation de chaque triangle, “la couverture” et les valeurs z sont calculées à ce niveau
- (efficacité) Mais le *fragment shader* est exécuté **une fois par pixel final (par triangle)**
 - Ces données de couleurs sont partagées entre tous les échantillons touchés par le triangle dans un seul pixel (résolution finale)
- Après la rasterisation, les groupes de ces échantillons à haute résolution sont moyennés ensemble

MULTI-SAMPLING

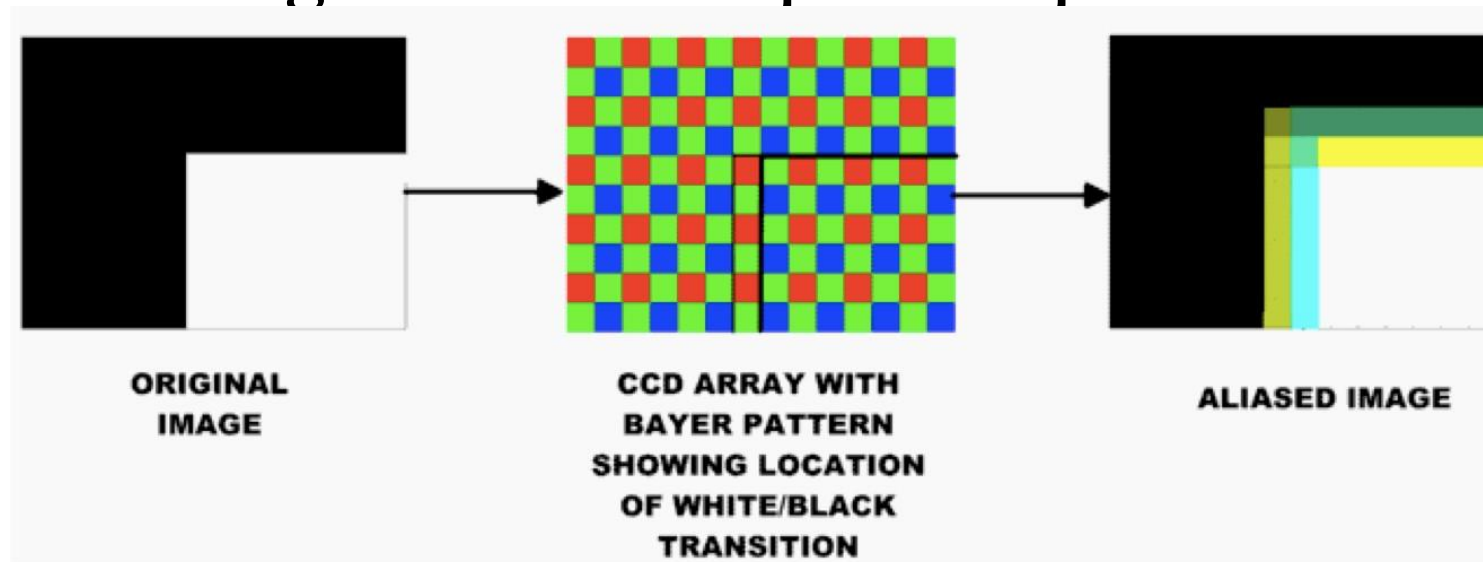
- Pourquoi c'est efficace?
- Les couleurs souvent changent lentement à l'intérieur de chaque triangle
 - ⇒ on ne doit pas calculer en haute résolution
 - Pas correct pour les textures
- Pour les textures: prétraiter la texture elle-même
 - Mipmap!

DANS LES APPAREILS PHOTO NUMÉRIQUES

- Anti-aliasage est:
 - L'intégration sur l'aire de chaque pixel
 - Un flou optique (en raison de la lentille)
- Parfois il y a des éléments optiques additionnels pour flouer l'image continue avant l'échantillonnage par les senseurs

CAMÉRA: LE PROBLÈME DE DÉMATRIÇAGE (*DEMOSAICING*)

- Imaginer un coin noir et blanc
- Il faut interpoler les valeurs de R,V,B!
- Un aliassage de couleur peut se produire



LA CAMÉRA: LE SENSEUR *FOVEON*

- Les couches transparentes RVB
- Pas de dématricage → pas de Moirés

