ISART DIGITAL

PARIS

Rasterization

et plus encore

summary

- Introduction
- La rasterization c’est quoi?
- Le pipeline graphique
- Dessin de lignes
- Edge function
- Coordonnées barycentriques
- Règles de rastérisation
- Anti-aliasing
- Clipping
- Perspective Correct Interpolation

Introduction

Objectifs

- Le but de ce module est de mieux comprendre ce qui se passe dans les moteurs de rendu et comment sont transformés les positions monde en pixel sur l'écran

La rasterization, c’est quoi?

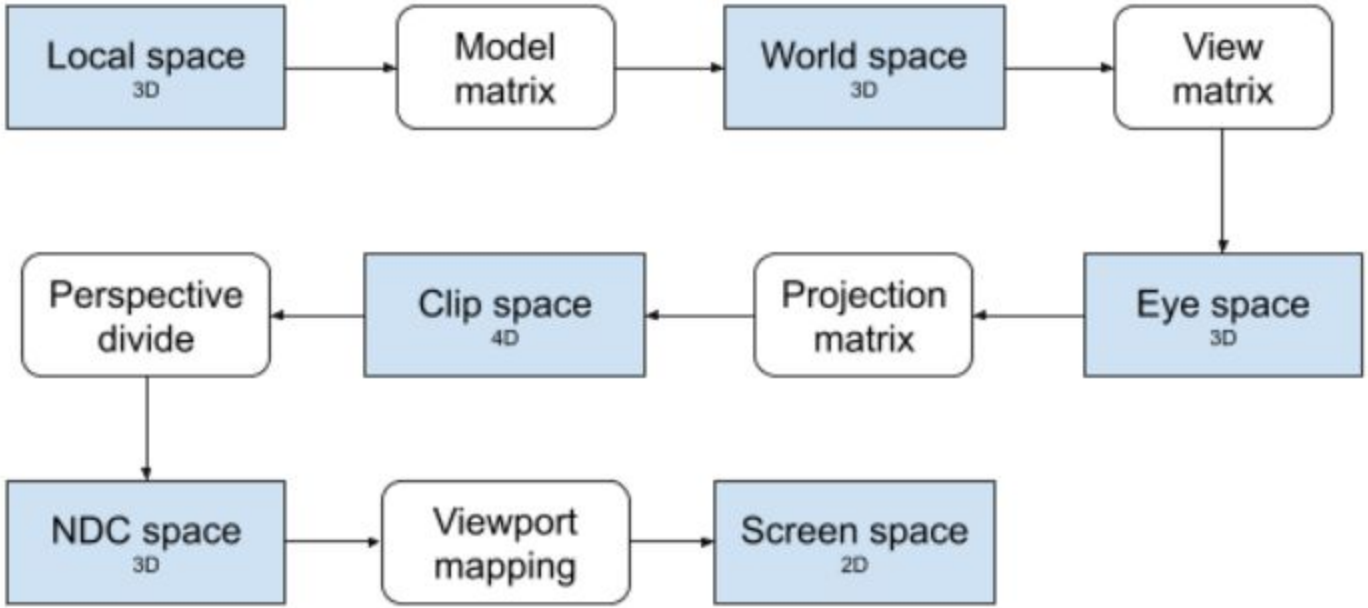
Wikipédia

“La rastérisation, ou matricialisation, est un procédé qui consiste à convertir une image vectorielle en une image matricielle destinée à être affichée sur un écran ou imprimée par un matériel d'impression.”

Rastérisation - Wikipédia

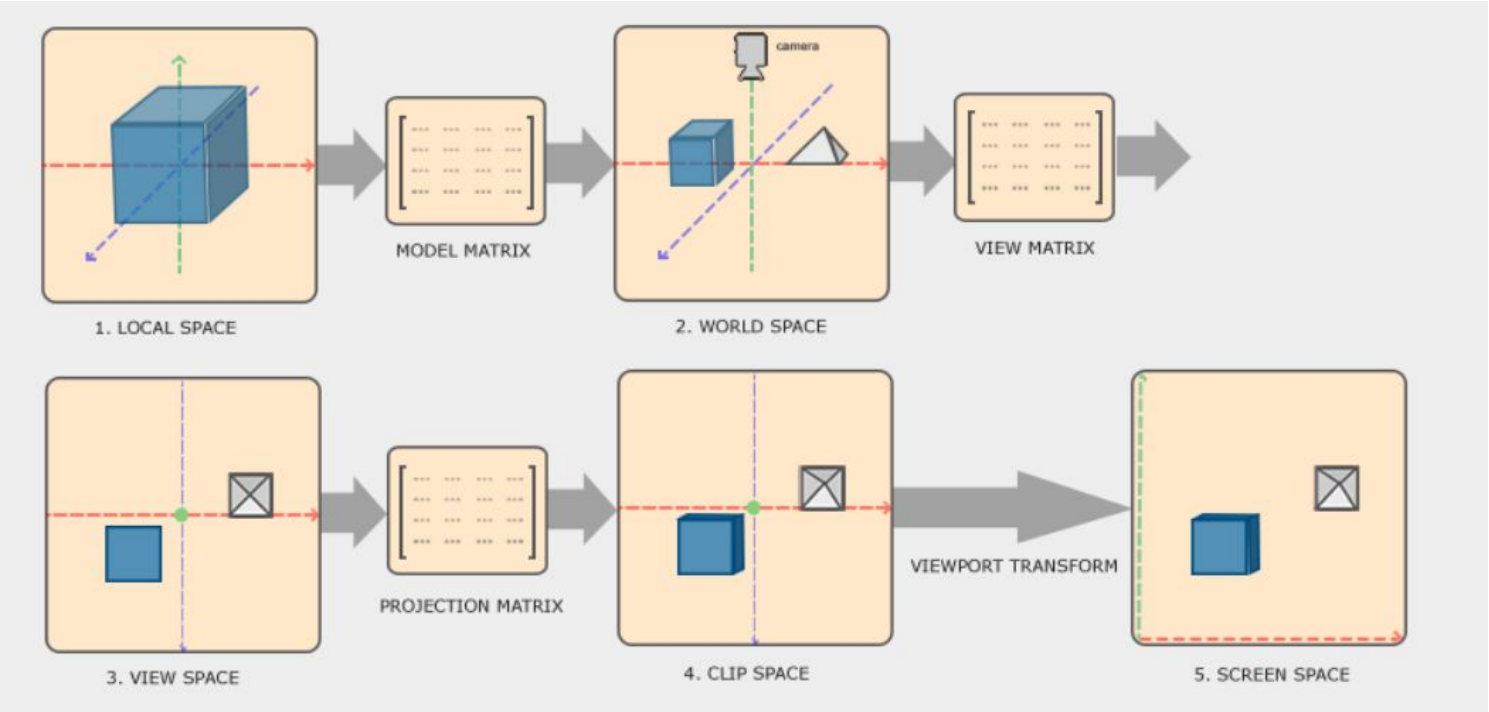
La rasterization, c’est quoi?

Transformation d’un vertex



La rasterization, c’est quoi?

Transformation d’un vertex



https://learnopengl.com/Getting-started/Coordinate-Systems

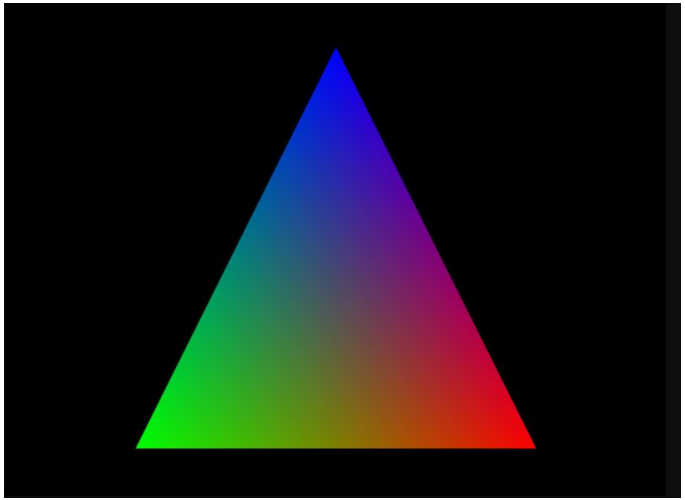
La rasterization, c’est quoi?

Voilà ce qu’il faut faire

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 255 255 255 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 255 255 255 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 255 255 255 0 0 0 255 255 255 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 255 255 255 0 0 0 255 255 255 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 255 255 255 0 0 0 255 255 255 0 0 0 0 0 0
0 0 0 0 0 0 255 255 255 0 0 0 0 0 0 0 0 0 255 255 255 0 0 0
0 0 0 0 0 0 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

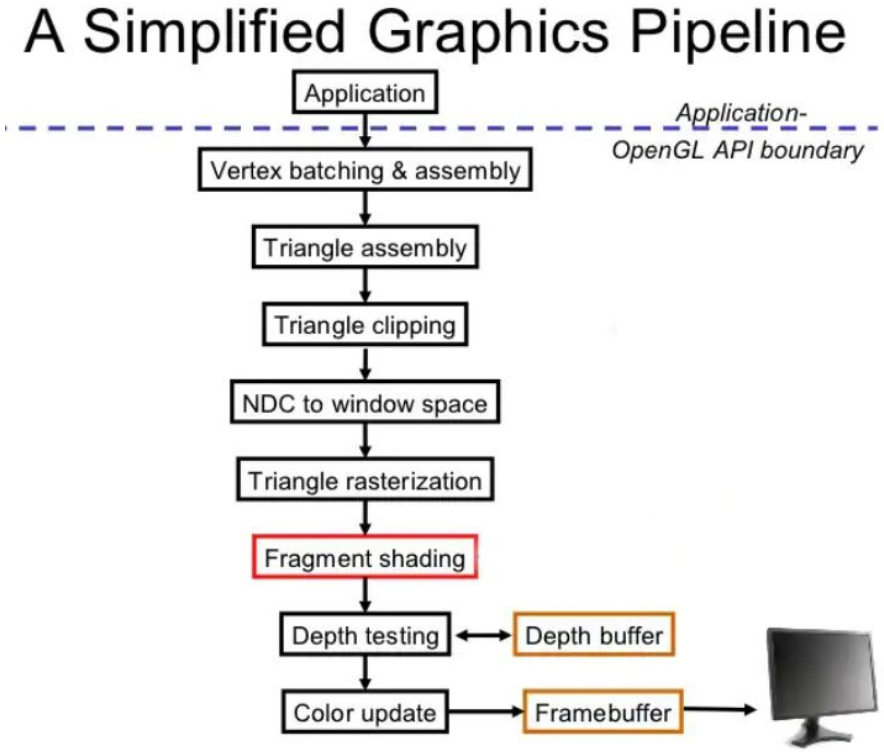
C’est un module qui va nécessiter pas mal de recherches de votre part, bon courage!

La suite sert à vous aider dans ces recherches.



Le pipeline graphique

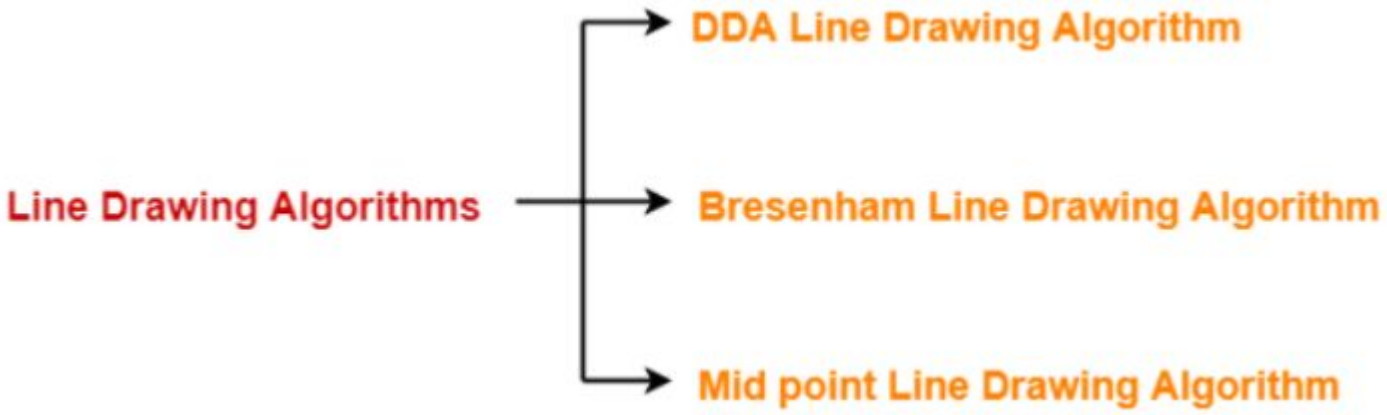
quand intervient la rasterization



Dessin de lignes

quand intervient la rasterization

Les trois principaux algorithmes de dessin de lignes sont le DDA, Midpoint et Bresenham.



Dessin de lignes

quand intervient la rasterization

Bresenham est une évolution du DDA, et également une modification du Midpoint algorithm. Midpoint est très utilisé pour dessiner des cercles.

La principale différence est que le Bresenham “classique” ne travaille qu’avec des entiers.

La logique de Bresenham dépend de l’ ”octant” de l’espace 2D dans lequel la droite se déplace :

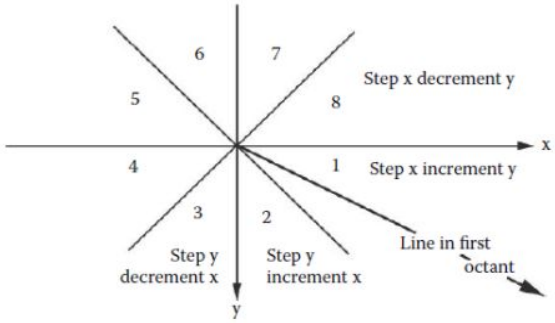


Figure 4.42: The Cartesian octants. Note that the positive y coordinate is downward. When drawing lines the center of the octants is taken to lie at the same place as the first point on the line. The illustrated line lies in the first octant.

Dessin de lignes

Quelques liens

Bresenham's Line Generation Algorithm

What's the difference between midpoint and Bresenham's line drawing algorithms?

Which is faster? Which looks better (without antialiasing)?

Polygon Rasterization

The Beauty of Bresenham's Algorithm

Edge function

Une edge function est une fonction qui permet de déterminer si un point est d'un côté ou de l'autre d'un segment.

En général, elles servent pour savoir si un point est à l'intérieur d'un triangle ou non, en comparant ce point à une droite sur le plan du triangle (en général un côté du triangle).

Si E est notre edge function, P le point à tester, et AB le segment comparé à P, alors :

- $E_{AB}(P) > 0$ si P est "à droite" du segment
- $E_{AB}(P) = 0$ si P est sur le segment
- $E_{AB}(P) < 0$ si P est "à gauche" du segment

En utilisant les côtés du triangle comme segments, on est donc capable de déterminer si un point est à l'intérieur du triangle ou pas : il suffit que le résultat de la edge function ait le même signe pour les trois côtés :

- négatif en sens antihoraire
- positif en sens horaire.

Edge function

La technique de Pineda utilise lourdement ce concept

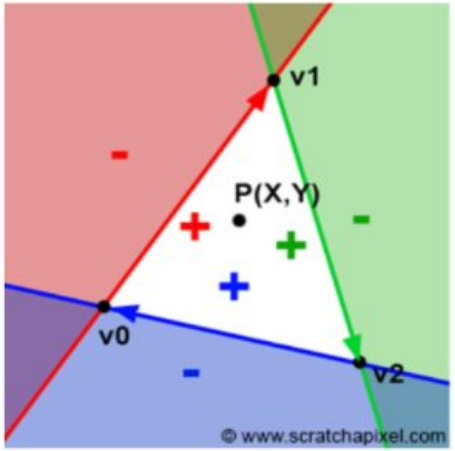


Figure 3: points contained within the white area are all located to the right of all three edges of the triangle.

$$E_{01}(P) = (P.x - v0.x) * (V1.y - V0.y) - (P.y - V0.y) * (V1.x - V0.x).$$

Attention car la fonction peut changer légèrement selon que les calculs se font sur des triangles dans le sens horaire ou antihoraire. Il faut donc bien vérifier quelle est la convention utilisée par la formule qu'on choisit

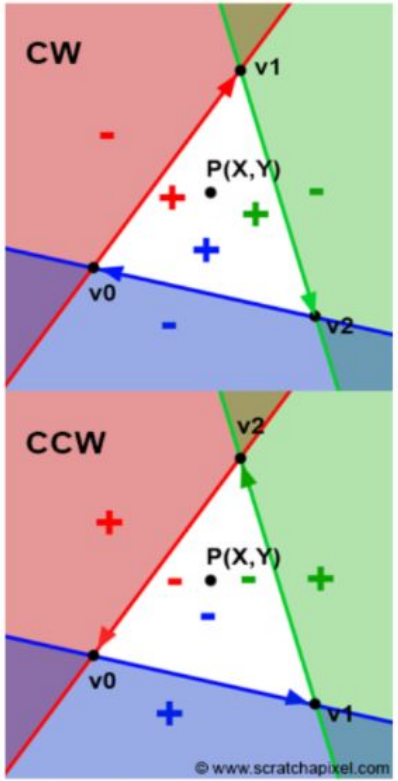


Figure 10: the ordering defines if points inside the triangle are positive or negative.

Coordonnées barycentriques

Connaître sa couleur

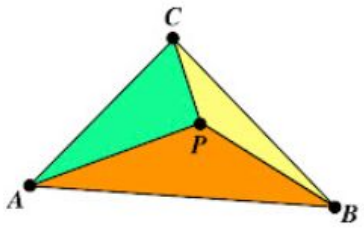
Les coordonnées barycentriques sont très utiles pour interpoler les attributs de vertex au cours de la rastérisation de triangle.

Elles consistent basiquement à établir “où on se situe” sur le triangle au moyen de trois coefficients relatifs à chacun des points des triangles. Ces coefficients (float) sont tous compris entre 0 et 1 et leur somme doit toujours être égale à 1.

Elles permettent également de déterminer si un pixel est à l’intérieur du triangle, via une **edge function** comme celle de **Juan Pineda** par exemple.

Barycentric Coordinates

- A coordinate system
  in which the location of a point of a simplex



$$P = w_A \times A + w_B \times B + w_C \times C$$

$$w_A = \frac{\Delta PBC}{\Delta ABC} = \frac{\text{▲}}{\text{▲}}$$

$$w_B = \frac{\Delta PCA}{\Delta ABC} = \frac{\text{▲}}{\text{▲}}$$

$$w_C = \frac{\Delta PAB}{\Delta ABC} = \frac{\text{▲}}{\text{▲}}$$

inside condition

$$0 \le w_A, w_B, w_C \le 1 \qquad w_A + w_B + w_C = 1$$

Coordonnées barycentriques

Connaître sa couleur

À noter qu'au cours de la rastérisation, il est possible d'optimiser le calcul des coefficients barycentriques en les incrémentant d'un step fixe à chaque pixel.

Cela évite d'avoir à recalculer la edge function à chaque pixel, ce qui est assez coûteux (deux multiplications et cinq soustractions).

[The barycentric conspiracy | The ryg blog]
[Rasterization: a Practical Implementation (The Rasterization Stage)]
[Optimizing the basic rasterizer | The ryg blog]

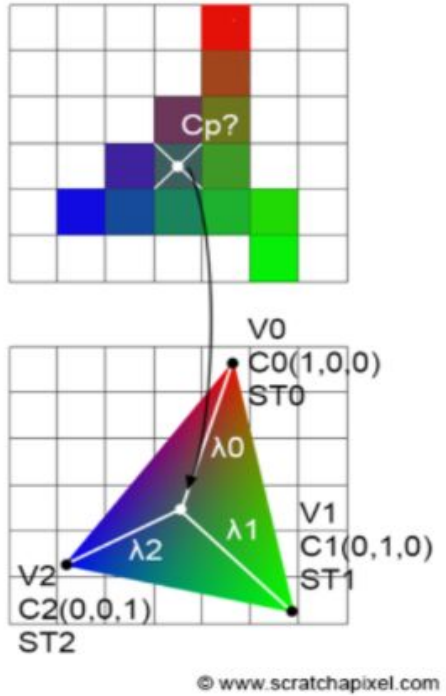


Figure 1: Finding the value of Cp requires to interpolate the colors defined at the triangle vertices using P's barycentric coordinates.

Règles de rasterization

Afin d'optimiser la rastérisation et de donner un comportement prévisible en toute circonstance, les rasterizers ont adopté deux règles plus ou moins standard :

- **La top-left rule**
- **La diamond-exit rule.**

La règle top-left résout un problème qui se pose couramment : lorsque deux triangles partagent un côté sur exactement les mêmes pixels, comment déterminer lequel des deux doit être rastérisé ?

On détermine deux cas particuliers de côtés de triangle :

- Les **top edges** : un côté parfaitement horizontal et au-dessus des autres côtés
- Les **left edges** : un côté qui n'est pas parfaitement horizontal et sur la gauche du triangle. Un triangle peut avoir 1 ou 2 left edges

La **règle top-left** peut se résumer à :

Un pixel ne prend la couleur d'un triangle que s'il est à l'intérieur du triangle, ou si le centre du pixel se situe sur une top edge ou une left edge.
Sinon, ce pixel n'est pas rastérisé pour ce triangle

Règles de rasterization

Afin d'optimiser la rastérisation et de donner un comportement prévisible en toute circonstance, les rasterizers ont adopté deux règles plus ou moins standard :

- **La top-left rule**
- **La diamond-exit rule.**

La règle top-left résout un problème qui se pose couramment : lorsque deux triangles partagent un côté sur exactement les mêmes pixels, comment déterminer lequel des deux doit être rastérisé ?

On détermine deux cas particuliers de côtés de triangle :

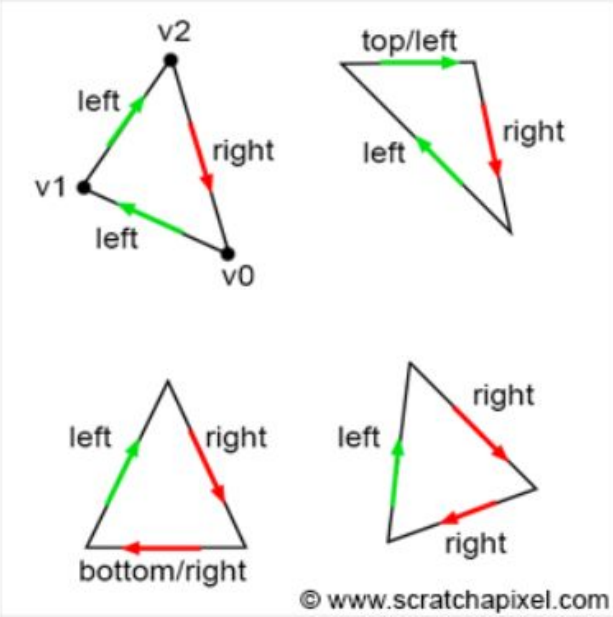
- Les **top edges** : un côté parfaitement horizontal et au-dessus des autres côtés
- Les **left edges** : un côté qui n'est pas parfaitement horizontal et sur la gauche du triangle. Un triangle peut avoir 1 ou 2 left edges

La **règle top-left** peut se résumer à :

Un pixel ne prend la couleur d'un triangle que s'il est à l'intérieur du triangle, ou si le centre du pixel se situe sur une top edge ou une left edge.
Sinon, ce pixel n'est pas rastérisé pour ce triangle

Règles de rasterization

La diamond-exit rule est une règle servant pour la rastérisation de lignes. Elle permet de standardiser le comportement du rastériseur lorsque deux lignes partagent un même point de début ou de fin.

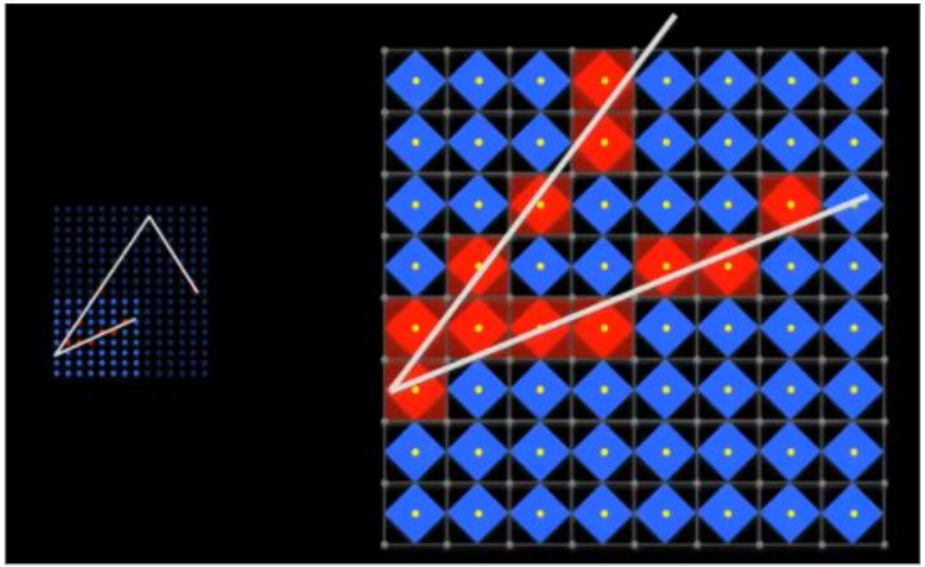


Règles de rasterization

On peut la résumer par :

Imaginons un losange (ou diamant/diamond) équilatéral au centre d'un pixel. **Lorsqu'une ligne traverse le pixel, elle n'est rastérisée que si la ligne sort du losange. Sinon, elle ne sera pas rastérisée.**

La bonne nouvelle, c'est que normalement, un algorithme de Bresenham bien implémenté applique automatiquement la diamond-exit rule, il n'y a donc pas besoin d'une implémentation particulière pour cette règle.



On dessine de gauche à droite. Rouge = dessiné

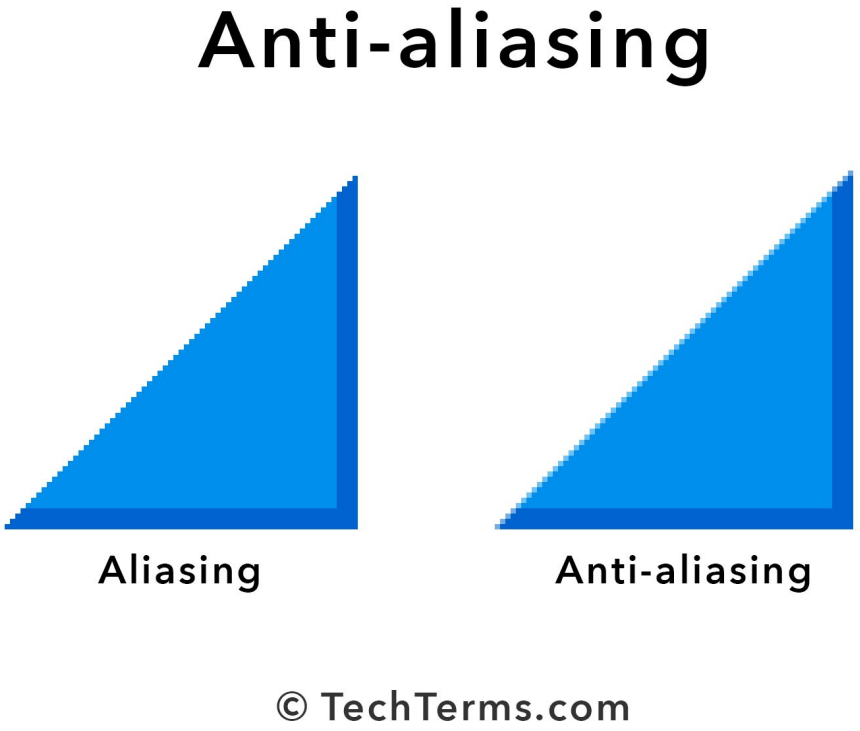
[(Unit 3) Drawing Primitives 3: Diamond Exit Rule]

Anti-aliasing

Les escaliers c’est pas si beau

Le but de l’anti-aliasing est de réduire le coté escalier en traitant les pixels autour de l’edge.

Rasterization: a Practical Implementation (Rasterization: a Practical Implementation)



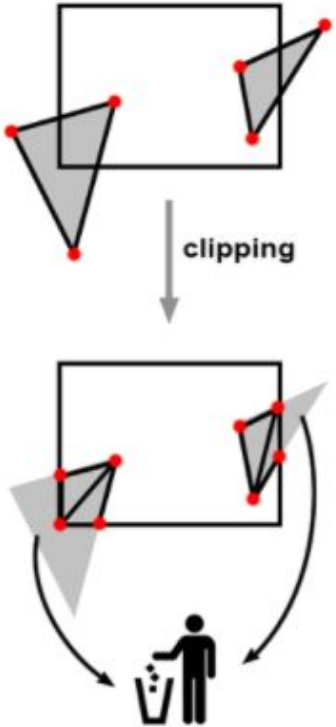
Clipping

Je n’affiche que ce que je vois

Le clipping consiste à éliminer toute la géométrie qui serait hors de l’écran (et donc invisible) avant la phase de rastérisation.

Pour cela, on va faire des tests (sur la primitive entière ou sur ses points) avec les plans du frustum de caméra pour savoir si ils sont à l’intérieur ou à l’extérieur.

Lorsqu’une primitive entière est à l’extérieur du volume de projection, elle est éliminée du pipeline. Si une partie de la primitive est visible, on va la subdiviser en plus petites primitives qui correspondent à la partie affichée à l’écran :



Clipping

Je n’affiche que ce que je vois

Le but du clipping est double :

- pour la performance
  - on évite des calculs sur de la géométrie qui ne sera de toute façon pas affichée
- pour corriger les artefacts de perspective
  - un problème des formules de perspective classiques est que les points derrière la caméra, une fois projetés, réapparaissent “en miroir” sur le volume de projection. C’est un “bug mathématique” qu’on souhaite éviter et pour cela, on “clip” la géométrie qui est en dehors du frustum.

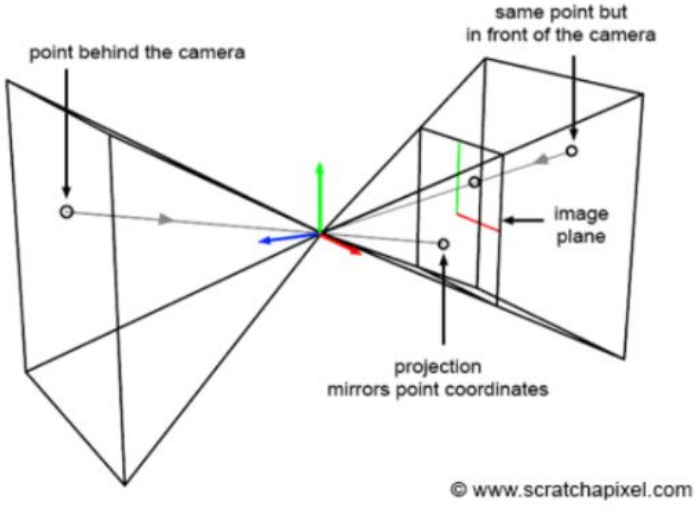


Figure 4: a point located behind the camera will be projected as are points in front, but its projected coordinates will mirrored in both directions.

Clipping

Je n’affiche que ce que je vois

En 3D, il faut clipper les lignes et les polygones.

Les algorithmes de clipping les plus populaires sont :

- pour les lignes :
  - Sutherland-Cohen, “le classique”
  - Liang-Barsky et Cyrus-Beck, des approches différentes mais plus optimisées.
  - Nicholl-Lee-Nicholl, réputé encore plus optimisé que Liang-Barsky
- pour les polygones :
  - Sutherland-Hodgman, fonctionne pour tout polygone convexe.

The Perspective and Orthographic Projection Matrix (Projection Matrices: What You Need to Know First)
https://simonstechblog.blogspot.com/2012/04/software-rasterizer-part-1.html

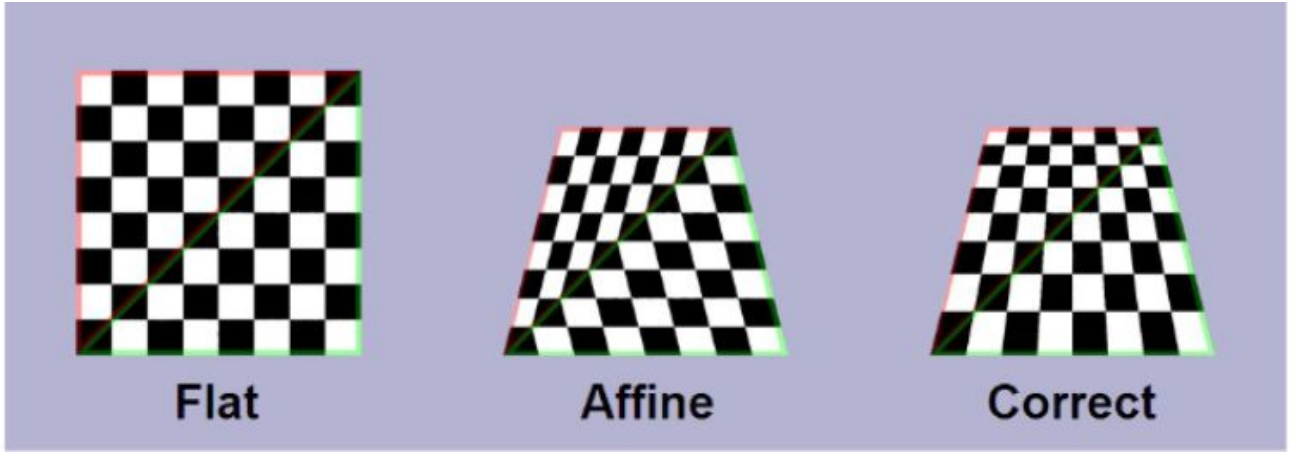
Perspective Correct Interpolation

Il faut que ce soit plus naturel

Un problème connu lorsque vous allez commencer à interpoler vos attributs :

La transformation de perspective n'étant pas une transformation affine, à cause de la division par le Z, les attributs interpolés de façon linéaire vont être soumis à une "distorsion" qui va être accrue en fonction de la taille du triangle.

Ceci est rendu évident avec les textures par exemple :



Perspective Correct Interpolation

Il faut que ce soit plus naturel

De face, le problème est presque imperceptible.

Dès que la surface est regardée sous un certain angle (lorsque l'objet tourne par exemple), il devient évident que les coordonnées de texture ne sont pas correctement interpolées. Cela provoque des artefacts comme sur la figure du milieu.

Le résultat correct est à droite.

https://simonstechblog.blogspot.com/2012/04/software-rasterizer-part-2.html

La suite?

On verra un plus tard les lights et les modèles d’éclairages ainsi que les normales avec les matériaux.

Bon courage