# Génération de Maillage Hexaédrique Par Dual Contouring

Mustafa Senol

Septembre 2021 - Janvier 2022

# Contents

# 1 Abstract

Dual Contouring est un algorithme, similaire au Marching Cubes, qui crée des objets en 3D à partir d'un champ scalaire. Il permet de résoudre quelques limitations rencontrées en Marching Cubes. Dans ce rapport, on va d'abord faire une introduction au Dual Contouring, on va parler un peu sur l'implémentation, montrez-vous quelques résultats et finalement on va parler sur des améliorations possibles.

# 2 Introduction

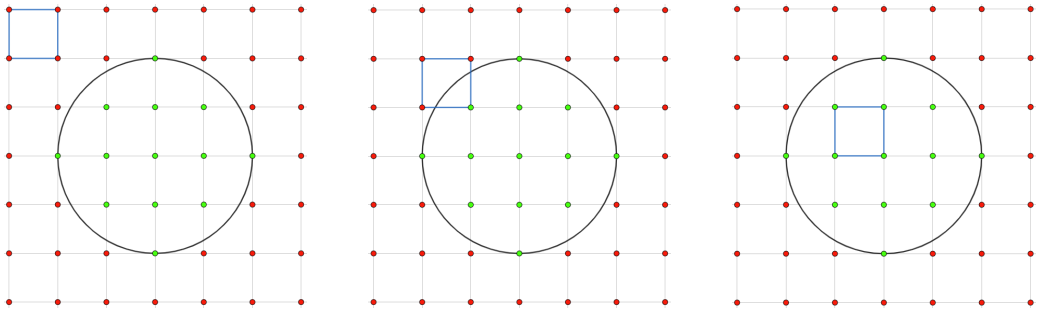
Les maillages, et donc maillages hexaèdrique, sont vastement utilisée dans notre vie. Dans le secteur de transport, architecture, médecine, pour le design et prototypages on utilise les maillages et on doit les créer en quelques sorts. Algorithme de Marching Cubes est l'une de méthodes pour créer ces maillages. Cet Algorithme prend un champ scalaire et retourne un objet en 3D, mais il a quelques limitations. Par exemple avec Marching Cubes, on ne peut pas mettre les sommets n'importe où. Pour cette raison, on ne peut pas représenter des coins des objets avec précision. Un autre problème avec Marching Cubes c'est il y a beaucoup de cas différent, en plus il y a des cas ambigieux qu'on doit gérer. Avec Dual Contouring, on va essayer de résoudre ces problèmes et créer des maillages dans le but de tester la qualité de ces maillages générés.

L'idée principale est au lieu de positionner les sommets sur les arrêts des cubes unitaires comme Marching Cube, on va prendre la liberté de les positionner n'importe où dans le cube unitaire. On va calculer la position optimal pour les sommets dans la section 3.1.1.

# 3 Mise en Œuvre

## 3.1 Trouver des Sommets

On commence par trouver des sommets de nos objets. Pour trouver des sommets, on va regarder à chaque cube unitaire dans la grille (La taille de grille est déterminée par utilisateur.) et on va calculer lesquelles devraient avoir un sommet dedans. Ce calcul est simple, car on doit seulement regarder s'il y a un changement de signe aux coins du cube unitaire ou pas (Le changement de signe veut dire qu'on est entré ou sortie de l'objet.). S'il y a, ça veut dire qu'on doit mettre un sommet dans ce cube unitaire (Car on veut des résultats volumiques, on va aussi mettre un sommet dans le cube unitaire si tous ses coins sont dans notre objet.). On peut simplement mettre ce point au milieu du cube, mais pour des résultats plus précis, il faut qu'on calcule où on doit mettre le point.



(a) (b) (c)

Figure 1: Dans ce projet on a travaillé en 3D mais car il est plus facile de visualiser en 2D on va faire des illustrations en 2D. Dans les figures a, b et c on voit un cercle de rayon 2. Sur cette cercle la fonction f qu'on veut représenter vaut 0. Les points rouges represente les points sur le grille qui est en dehors de circle. Sur des points rouge f est plus grande que 0. Les points verts sont des points sur la grille qui est dans ou sur la cercle. Sur des points verts f vaut plus petit que 0. Le carrée bleu est le carrée unitaire et il est équivalente au cube unitaire de la version 3D. Dans la figure a on peut voir que tous les coins du carrée unitaire sont en dehors du cercle. Donc on ne va pas mettre un sommet dans cette cellule. Par contre dans la figure b on peut voir que le carré unitaire a 3 sommets en dehors de cercle, et 1 sommet dans la circle. Donc on peut conclure qu'on doit mettre un sommet dans cette cellule. Finalement dans la figure c on peut voir que tous les coins sont dans la cercle donc on va aussi mettre un sommet dans cette cellule.

### 3.1.1 Positionner des Sommets

Pour positionner des sommets, on va d'abord trouver les positions ou notre fonction coupe le cube unitaire. Après, il faut qu'on calcule le gradient de la fonction sur ces points d'intersection. À partir des positionnes des intersections et des gradients, on peut calculer la position optimale pour le sommet.

Pour trouver la position optimale le sommet, il faut qu'on minimise la valeur de qef $= sum_i((p_i - s).n_i)^2$ où des $p_i$ sont des positionnes des intersections avec l'objet et grille, des $n_i$ sont des gradients et s est les coordonnées qu'on cherche.

$$
\begin{aligned}
\text{qef} &= \textstyle\sum_i((p_i - s).n_i)^2 \\
&= \textstyle\sum_i(((p_{ix}, p_{iy}, p_{iz}) - (x, y, z)).(n_{ix}, n_{iy}, n_{iz}))^2 \\
&= \textstyle\sum_i(((p_{ix} - x, p_{iy} - y, p_{iz} - z)).(n_{ix}, n_{iy}, n_{iz}))^2 \\
&= \textstyle\sum_i(((p_{ix} - x) * n_{ix} + (p_{iy} - y) * n_{iy} + (p_{iz} - z) * n_{iz}))^2 \\
&= \textstyle\sum_i((-n_{ix} * x) + (-n_{iy} * y) + (-n_{iz} * z) + (n_{ix} * p_{ix} + n_{iy} * p_{iy} + n_{iz} * p_{iz}))^2 \\
&= \textstyle\sum_i((-n_{ix} * x) + (-n_{iy} * y) + (-n_{iz} * z) + (n_{ix} * p_{ix} + n_{iy} * p_{iy} + n_{iz} * p_{iz}))^2
\end{aligned}
$$

$= (\sum_i (-n_{ix})^2) * x^2 + (\sum_i 2 * (-n_{ix}) * (-n_{iy})) * xy$
$+ (\sum_i 2 * (-n_{ix}) * (-n_{iz})) * xz$
$+ (\sum_i 2 * (-n_{ix}) * (n_{ix} * p_{ix} + n_{iy} * p_{iy} + n_{iz} * p_{iz})) * x + (\sum_i (-n_{iy})^2) * y^2$
$+ ((\sum_i 2 * (-n_{iy}) * (-n_{iz}))) * yz$
$+ (\sum_i 2 * (-n_{iy}) * (n_{ix} * p_{ix} + n_{iy} * p_{iy} + n_{iz} * p_{iz})) * y$
$+ (\sum_i (-n_{iz})^2) * z^2 + (\sum_i (-n_{iz}) * (n_{ix} * p_{ix} + n_{iy} * p_{iy} + n_{iz} * p_{iz})) * z$
$+ \sum_i (n_{ix} * p_{ix} + n_{iy} * p_{iy} + n_{iz} * p_{iz})^2$

On peut simplifier l'équation avec un changement variable. On peut dite que:

a = $(\sum_i (-n_{ix})^2)$

b = $(\sum_i 2 * (-n_{ix}) * (-n_{iy}))$

c = $(\sum_i 2 * (-n_{ix}) * (-n_{iz}))$

d = $(\sum_i 2 * (-n_{ix}) * (n_{ix} * p_{ix} + n_{iy} * p_{iy} + n_{iz} * p_{iz}))$

e = $(\sum_i (-n_{iy})^2)$

f = $((\sum_i 2 * (-n_{iy}) * (-n_{iz})))$

g = $(\sum_i 2 * (-n_{iy}) * (n_{ix} * p_{ix} + n_{iy} * p_{iy} + n_{iz} * p_{iz}))$

h = $(\sum_i (-n_{iz})^2)$

i = $(\sum_i (-n_{iz}) * (n_{ix} * p_{ix} + n_{iy} * p_{iy} + n_{iz} * p_{iz}))$

j = $\sum_i (n_{ix} * p_{ix} + n_{iy} * p_{iy} + n_{iz} * p_{iz})^2$

qef = a$x^2$ + bxy + cxz + dx + e$y^2$ + fyz + gy + h$z^2$ + iz + j

Pour trouver le minimum on va calculer les dérivées partielles $\frac{\partial qef}{\partial x}$, $\frac{\partial qef}{\partial y}$, $\frac{\partial qef}{\partial z}$.

$\frac{\partial qef}{\partial x}$ = 2ax+by+cz+d

$\frac{\partial qef}{\partial y}$ = 2ey+fz+bx+g

$\frac{\partial qef}{\partial z}$ = 2hz+cx+fy+i

Pour trouver le minimum on va trouver des valeurs x, y, z telle que ces trois dérivées vont être 0. Quand on résout ces trois équations on trouve:

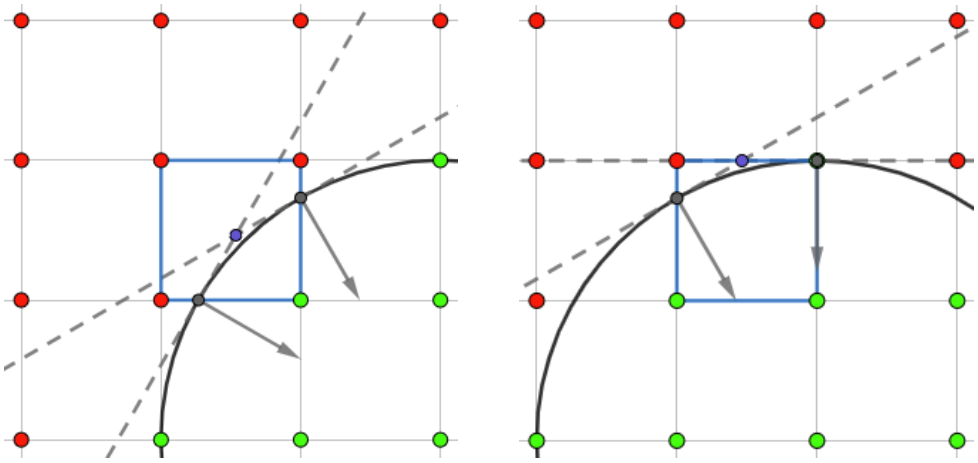
x = (bfk-2bgh-2cek+cfg+4deh-df$^2$)/(-8aeh+2af$^2$+2b$^2$h-2bcf+2c$^2$e)

y = (-2afk+4agh+bck-2bdh-gc$^2$+cdf)/(-8aeh+2af$^2$+2b$^2$h-2bcf+2c$^2$e)

z = (4aek-2afg-kb$^2$+bcg+bdf-2cde)/(-8aeh+2af$^2$+2b$^2$h-2bcf+2c$^2$e)

Sous conditions de (-8aeh+2af$^2$+2b$^2$h-2bcf+2c$^2$e)$\neq$0, a$\neq$0, e$\neq$0, h$\neq$0. Donc ça veut dire que si un minimum existe il est au point (x, y, z).

Parfois, on peut trouver que la meilleure position pour le sommet est en dehors de cube unitaire. Pour éviter cela faut qu'on ajoute un biais vers le milieu du cube unitaire. Si le sommet est toujours en dehors on va choisir un point dans le cube unitaire. Pour simplicité, on a simplement choisi le milieu des points intersections.



(a) (b)

Figure 2: La représentation 2D de trouver le sommet optimal. Les vecteurs représentent le gradient de la fonction, le point bleu c'est le sommet optimal.

## 3.2 Trouver des hexaèdres

On sait qu'il y a une bijection entre points de grille qui est dans la fonction et les hexaèdres de notre maillage. Ça veut dire qu'on peut itérer sur notre grille et chaque fois on trouve un point que la fonction vaut moins que 0, on peut regarder aux cellules incidentes et trouve l'hexaèdre contienne ce point. Si on fait ça pour tous les points, on peut trouver tous les hexaèdres.

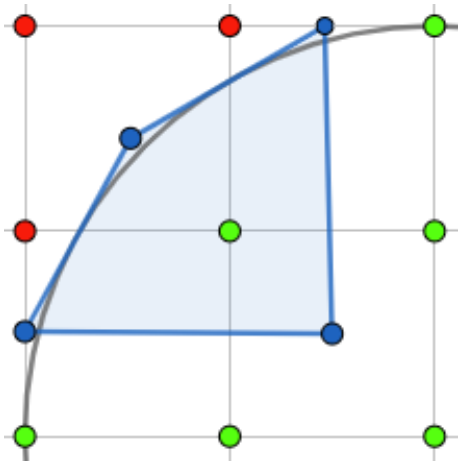


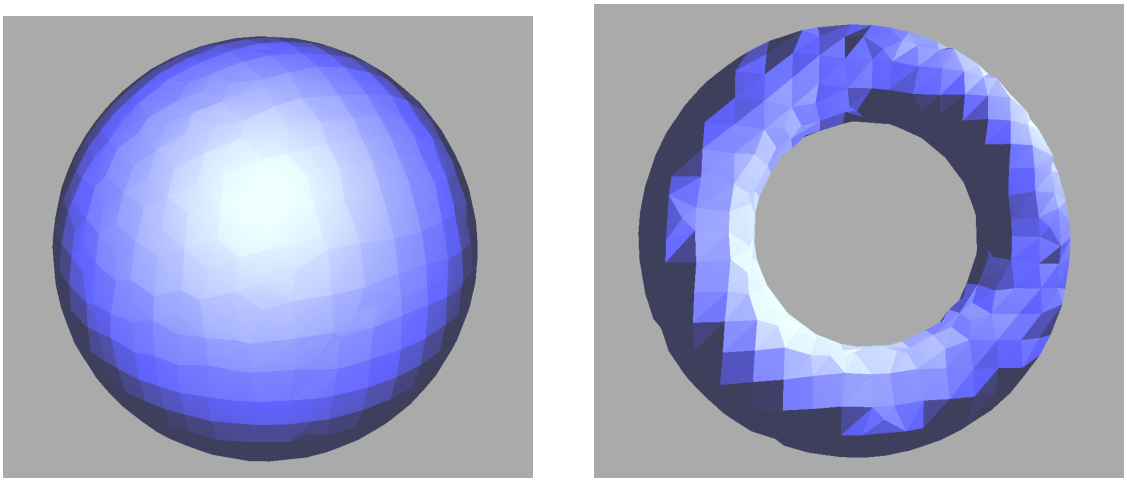
Figure 3: La représentation 2D de trouver les hexaèdres. La face dans l'image est équivalente à un hexaèdre en version 3D. On peut voir que chaque hexaèdre va contenir un point vert.

### 3.3 Construction de l'Objet

La construction de l'objet a été effectuée en utilisant la bibliothèque CMapJS. C'est une bibliothèque similaire au CGoGN mais c'est en JavaScript. Pour la construction on a utilisé les hexaèdres et les positionnes les sommets qu'on a calculés dans les étapes précédentes.

## 4 Résultats

On peut voir qu'on obtienne des résultats assez proches à l'objet original. Les résultats sont plutôt lisse, mais on peut occasionnellement voir des parties rugueux.



(a) (b)

Figure 4

## 5 Optimisations Possibles

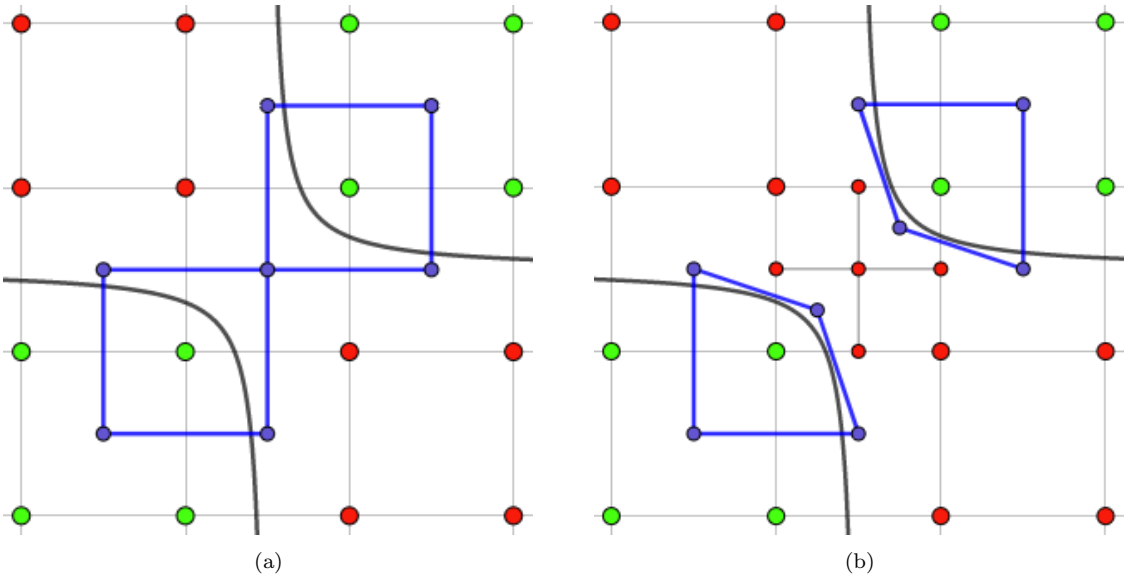
La première optimisation possible c'est d'utiliser plusieurs threads pour accélérer les calculs de positions des sommets et hexaèdres car la positionne d'un sommet ne dépend pas aux autres. C'est pareil avec des hexaèdres.

Une autre optimisation possible c'est de minimiser les calculs redondants. En ce moment, on n'a pas stocké des valeurs que la fonction retourne sur des points d'intérêts et on le calcule chaque fois à la volée. Pour un point dans l'objet, en moyenne on calcule sa valeur neuf fois. On peut utiliser un tableau des entiers et éviter de faire le calcule chaque fois on en a besoin. Noter que cette optimisation peut accélérer nos calculs mais nos programmes va aussi utiliser plus de mémoire.

# 6 Améliorations Possibles

Évidemment, on peut implémenter les optimisations qu'on a indiquées dans la section 5. À part ça, on peut aussi améliorer la fonction qui calcule la position optimale pour un sommet (section 3.1.1). En ce moment, quand on rencontre un problème avec cette fonction (par exemple quand le point optimal est en dehors du cube même après le biais vers le centre), au lieu de trouver la position optimale pour le sommet, on a calculé le milieu des points d'intersections. Les résultats sont pas mal, mais on peut l'améliorer pour qu'il donne des résultats plus lisse. Pour cela, il faut qu'on trouve le point qui minimise le qef (section 3.1.1) dans le cube unitaire.

De plus, parfois on peut voir des singularités dans les résultats. Pour éviter ça, on peut détecter des singularités et essayer de les faire disparaître en augmentant la résolution localement. Donc, si on modifie l'algorithme tel qu'on peut changer la taille du cube unitaire, on peut aussi potentiellement résoudre ce problème.



(a) (b)

Figure 5: Dans la figure a on voit un exemple singularité en 2D. Dans la figure b on voit le résultat obtenu en augmentant la résolution localement.

# 7 Références

[1] Boris: "Dual Contouring Tutorial"
https://www.boristhebrave.com/2018/04/15/dual-contouring-tutorial/
[2] François Protais: "Marching hex: Voxelization++"
https://gtmg2021.sciencesconf.org/350686/document
https://www.youtube.com/watch?v=6igfpWlXhBAab_channel=DmitrySokolov
[3] P. Viville, P. Kraemer, D. Bechmann: "Hexahedral Mesh Generation For Tubular Shapes Using Skeletons and Connection Surfaces"
[4] Scott Schaefer, Tao Ju, Joe Warren: "Manifold Dual Contouring"