Project Senior

VISUALISATION DE LA CROISSANCE DE MATIERE

Encadrement Institut Image :

Ruding LOU

Encadrement LaBoMaP (ENSAM,Cluny) :

Aurélien BESNARD

ZHU Yuanju

Expertise MTI3D

06/01/2020

Arts et Métiers Sciences et Technologies

SOMMAIRE

Introduction

Contexte de la technologie PVD

La méthode Dépôt sous Vide en phase Vapeur (Physical Vapor Deposition) est un ensemble de plusieurs procédés pour produire un film sur des pièces. La méthode de pulvérisation peut être utilisée pour préparer une variété de matériaux tels que des métaux, des semi-conducteurs, des isolants, etc., ses avantages sont: équipement simple, contrôle facile, grande surface de revêtement et forte adhésion. Et la pulvérisation cathodique magnétron (appelé « magnetron sputtering » en anglais, inventée en 1970) plus d'avantages: une vitesse élevée, une température basse et des dégâts minimes.

Le processus de PVD

Le principe de fonctionnement de la pulvérisation magnétron est que:

Les électrons entrent en collision avec des atomes d'argon au cours du processus de vol jusqu'au substrat sous l'action d'un champ électrique, ce qui provoque leur ionisation pour produire des ions positifs Ar^+ et des électrons secondaires (la génération de plasma).

Les électrons volent vers le substrat, au contraire, les ions Ar^+ accélèrent sous l'action du champ électrique jusqu'à la cible de la cathode et bombardent la surface de la cible avec une énergie élevée, provoquant une pulvérisation de la cible. Dans les particules pulvérisées, des atomes(ou molécules) neutres sont déposés sur le substrat pour former un film mince.

Les électrons secondaires sont affectés par les champs électriques et magnétiques, donc ils dérivent. S'il s'agit d'un champ magnétique toroïdal, les électrons sont contraints de faire un mouvement circulaire dans la zone du plasma près de la surface cible et donc une grande quantité d'Ar est ionisée dans cette zone pour bombarder le matériau cible, ce qui augmente la vitesse de dépôt.

Au fur et à mesure que le nombre de collisions augmente, l'énergie des électrons secondaires s'épuise, et ils s'éloignent progressivement de la surface de la cible et se déposent enfin sur le substrat sous l'action du champ électrique. L'énergie de l'électron étant très faible, l'énergie transférée sur le substrat est très faible, ce qui entraîne une élévation température modérée du substrat.

Le procédé qui nous concerne est la pulvérisation cathodique magnétron(voir le schéma 1 au dessous).

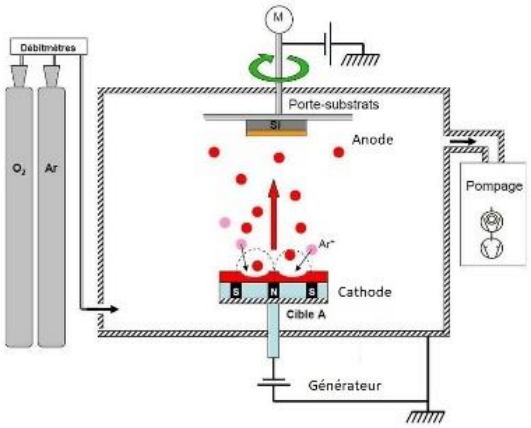


Figure 1 Schéma de la pulvérisation cathodique magnétron
(source : https://images.app.goo.gl/XEz1XaEF2exVFbir9)

La pulvérisation cathodique magnétron est une technique dominante pour la croissance de films minces, car une grande quantité de films minces peut être préparée avec une pureté relativement élevée et un coût faible.

État de l'art

Simulation du processus PVD

Actuellement, de nombreux chercheurs ont effectué des simulations informatiques sur le processus de PVD, mais leur travail consiste souvent à visualiser la formation de films minces avec différents temps de dépôt en mode bidimensionnel. Par exemple

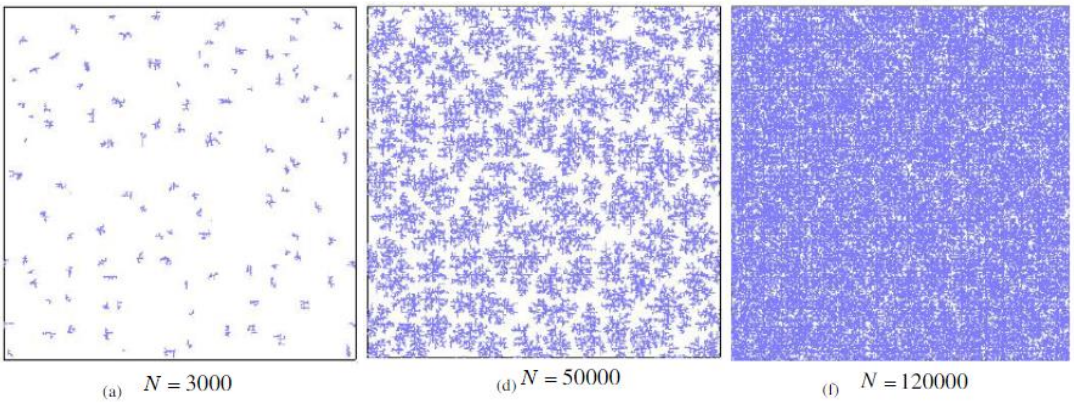


Figure 2 Des films former par PVD(N décrit le temps de dépôt)

Bien que le degré de réduction de ces simulations soit relativement élevé, nous ne pouvons pas visualiser dynamiquement l'ensemble du processus en trois dimensions, ce qui n'est pas le résultat optimal pour nous. Mais leur travail fournit également des moyens et des méthodes pertinents pour nous de simuler le processus PVD en 3D en utilisant Unity.

Extraction de contour de nuage de points

Les chercheurs ont effectué des recherches connexes. Dans CHENG Xiaojun et al., Un algorithme rapide pour générer des courbes de niveau de contour basé sur des données de nuage de points découpées a été proposé. Sa méthode comprend les étapes suivantes: 1. Génération de tranches de nuages de points; 2. Extraction de points; 3. Génération de lignes d'entités de contour.

Figure 5 (a) la sculpture; (b) extraction de points; (c) lignes de contour

Cahier des charges

Nous devons réaliser la visualisation du PVD sur le processus de croissance du revêtement de surface d'une pièces sur Unity. Les objectifs spécifiques sont les suivants:

- Visualisation interactive en 3D des atomes composants le film (Fig. 4.a).
- Identification des regroupements des atomes pour les différentes colonnes (Fig. 4.b)
- Génération d'une surface à partir d'une colonne (Fig. 4.c)
- Génération d'un volume de matière avec les cavités intérieures (Fig. 4.d)

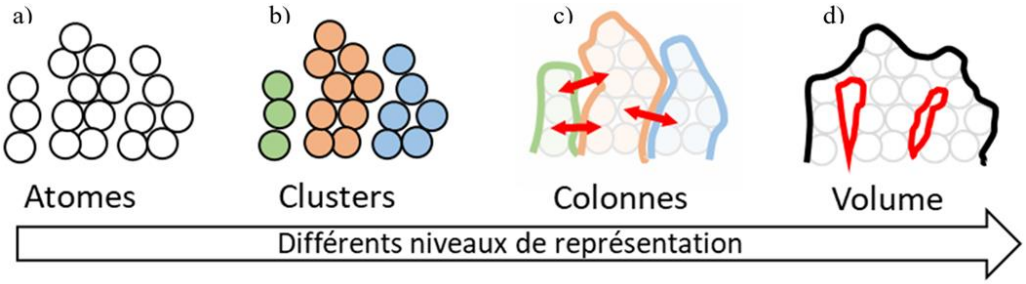


Figure 4 Exemple de la visualisation et représentation de la croissance de matière

Planning

Nous n'avons fait que la planification suivante en fonction du calendrier du projet:

Tableau1 planning du projet

N	Tâche	Qualification	Duration	Date	Commentaire
1	Compréhension de la technologie PVD	mieux si y'en a	1 jour	02.décembre	Visite de campus Cluny (discussion avec les profs; visite de la machine)
2	Discussion avec prof.	obligé	3 mois	Tout au long du projet	Se faire compréhension de la problématique; le choix de solution et algorithme; avancement deprogrammation.
3	Problématique, état de l'art	obligé	5 jours	01.novembre-5.novembre	
4	Études de cahier des charges et planning	obligé	5 jours	5.novembre-10.novembre	
5	Cherche de la méthode pour une solution	obligé	15 jours	10.novembre-25.novembre	
6	Développement d'algorithme et logiciel	obligé	25 jours	25.novembre-20.décembre	discussion avec le prof.; cherche dans les forum de technologie
7	Rédaction de rapport	obligé	17 jours	20.décembre-6.javier	faire référence à "PS_architecure_rapport"
8	Préparation de la soutenance	obligé	7 jours	6.javier-13.javier	Rédaction d'un PPT et répétition

Travail effectué

Après avoir importé les données dans Unity, nous devons d'abord regrouper les données selon différentes colonnes. Ensuite puisque nous voulons voir la surface extérieure de chaque volume et ne nous soucions pas de la disposition des atomes à l'intérieur du colonne, afin d'économiser les ressources informatiques, nous avons l'intention de supprimer les points de chaque groupe qui ne sont pas à la position limite.

1.Regroupement des atomes

Méthode

Tout d'abord, nous allons regrouper tous les atomes, afin que nous puissions colorer les atomes plus tard pour faciliter l'observation de la distribution des atomes dans le revêtement.

Nous avons l'intention de sélectionner au hasard un point sur une tranche de colonne, de trouver ses voisins à travers ce point, puis d'utiliser les voisins pour trouver leurs nouveaux voisins pour regrouper les données par colonne. Lorsqu'aucun nouveau voisin ne peut être trouvé, nous pensons que cette colonne contient toutes les données que nous recherchons et que toutes les autres données appartiennent à d'autres colonnes. Par conséquent, afin de trouver les colonnes restantes, nous effectuerons une nouvelle recherche au hasard Répétez le travail ci-dessus jusqu'à ce que toutes les données des colonnes soient trouvées.

L'algorithme

L'initiation est de laisser l'utilisateur entrer une valeur de hauteur($Z=Z_0$), définissez la distance maximale entre les atomes dans un même groupe($\varepsilon=\varepsilon_0$). Supposons que tous les points du plan Z_0 constituent un tableau P avec les coordonnées tridimensionnelles des points comme ses éléments. Si vous regardez la **Figure 5** au dessous, toutes les étoiles dans le plan Z_0 constituent le tableau P.

L'idée est de rechercher un ensemble de points P1 avec une distance inférieure à ε autour de lui par le point p_0 arbitrairement sélectionné, puis de trouver un ensemble de points P2 avec une distance inférieure à ε autour de lui par P1 (P2 ne contient pas p_0), jusqu'à ce que nous trouvions Il n'y a pas de points avec une distance inférieure à K. À ce moment, nous avons trouvé le premier groupe de colonnes. Ensuite, nous recherchons d'autres colonnes de la même manière, jusqu'à ce que tous les points soient triés dans chaque colonne.

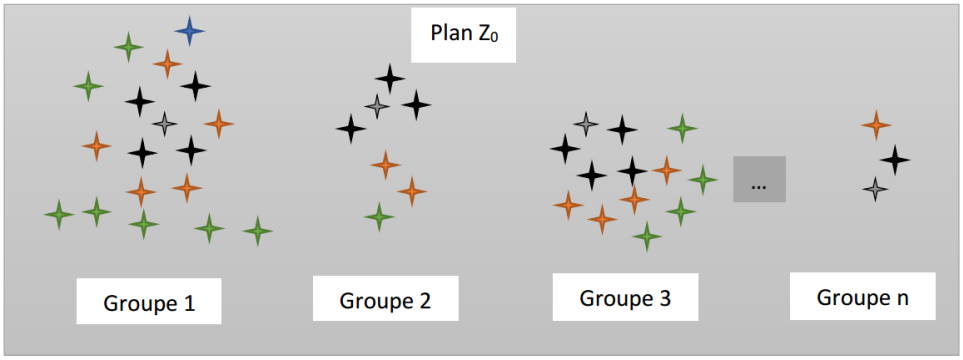


Figure 5 Schéma de l'algorithme

L'organigramme du programme :

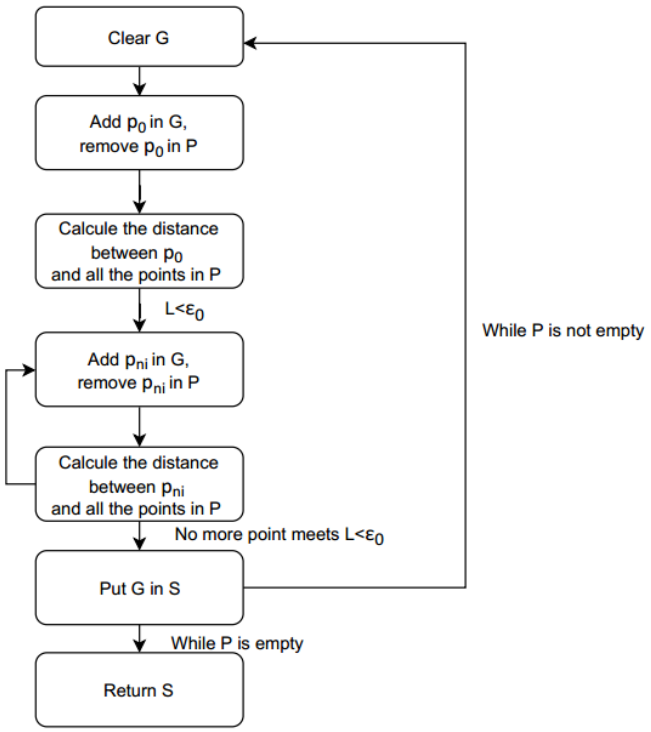


Figure 6 L'organigramme du programme pour regrouper les atomes

Ensuite, nous trouverons les points de données pour le contour de surface de chaque colonne.

2.Trouver le contour d'une colonne

Méthode

Afin de trouver le contour d'une colonne, nous avons d'abord pensé à scanner les données dans les directions X et Y respectivement, puis à trouver le contour.

Mais nous avons constaté que cette méthode ne peut être appliquée qu'à certains contours avec des formes relativement simples, ensuite, nous avons pensé à citer un rayon pour scanner

à partir du centre de masse de la colonne. Cependant, cette méthode présente des inconvénients similaires à l'ancienne.

Nous avons vu qu'il existe un moyen qui utilise un point de départ pour trouver le contour , Mais cette méthode est relativement compliquée.

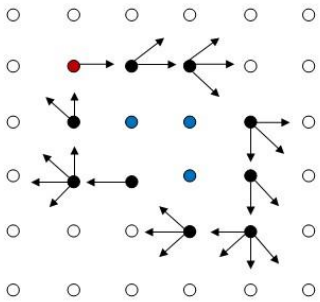


Figure 7 La cherche de contour avec le « seed » point rouge

Nous avons donc utilisé la méthode de "densité" que nous allons parler suivante pour résoudre notre problème.

Sur la base des atomes dans le colonne sont répartis « uniformément ». Notre idée est de déterminer si nous sommes à la frontière en calculant la « densité » du point où se trouve chaque atome.

Voici comment nous définissons la «densité»: Dans le plan $Z = Z_0$, la densité du point d'un atome est égale au nombre d'atomes les plus proches autour d'elle (contrairement au calcul du nombre d'atomes dans la cellule unitaire en chimie).

Comme au-dessous: supposons que la matrice suivante soit tous les atomes d'une colonne dans le plan $Z = Z_0$ que nous avons trouvé de la manière précédente, chaque petit carré noir est un atome. Donc La densité de l'emplacement A, B, C et D est: 3, 2, 4, 1 respectivement.

De plus, nous pouvons voir que, tant que la densité d'un certain point est inférieure à 4, alors ce point est à la position de la frontière, et nous stockons ce pour tracer le contour après.

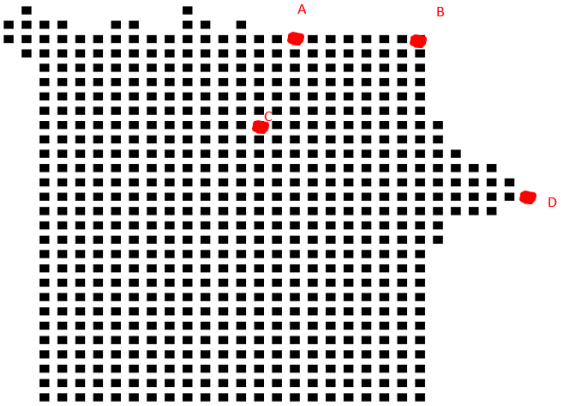


Figure 8 La cherche de contour avec la « densité » méthode

L'alogorithmes

Nous lisons d'abord tous les points de coordonnées des colonnes et les mettons dans le tableau un T. Après l'achèvement, nous prenons un point p(du premier au dernier) dans chacun colonne. Initialisons « densité = 0 » et trouvons combien de points dans T sont inférieurs ou égaux à la distance minimale que nous avons définie, Augmentez la valeur de « densité » de 1 chaque fois qu'un point est satisfait. Enfin, nous comparerons la valeur de la densité avec la valeur de « 5 ». Tant que la densité est inférieure à 5, nous considérons ce point p comme un point sur le contour, enregistrons-le dans un tableau « contourPoints ». Une fois que tous les ContourPoints ont été trouvés, nous plaçons ContourPoints dans le tableau contourGroups pour faciliter nos appels ultérieurs.

3.Trouver le centre de masse d'une colonne

Méthode

Afin de trouver l'axe d'une colonne, nous devons d'abord trouver le centre de masse des sections de la colonne. Ensuite, utiliser une certaine méthode pour ajuster une ligne de plusieurs centres de masse, ainsi, l'inclinaison de la colonne peut être calculée et utilisée pour guider la génération de l'extrusion du volume.

L'alogorithmes

Pour trouver le centre de masse, dans le plan Z_0, nous prenons toutes les coordonnées X dans les données d'une colonne pour calculer le moyen d'X: Xbar, puis nous prenons toutes les coordonnées Y dans les données d'une colonne pour calculer le moyen d'Y: Ybar. Alors, (Xbar, Ybar, Z_0) est le centre de masse que nous recherchons.

Résultats expérimentaux

Configuration

Supprimez d'abord tous les informations à l'exception des coordonnées tridimensionnelles des points, puis nous plaçons le fichier TXT sous Assets / StreamingAssets / , vous pouvez saisir le nom du fichier à traiter dans nom du fichier.

puis saisir une valeur de hauteur à traiter. Nous obtiendrons le regroupement des données de coordonnées de chaque colonne dans cette tranche de hauteur, et le regroupement des données de coordonnées du contour de chaque colonne.

"Upsilon" est un paramètre que nous pouvons ajouter en douceur pour le regroupement. Sa signification est l'intervalle minimum admissible entre deux colonnes. Vous pouvez ajuster le regroupement des données en fonction de cette différence de valeur (bien sûr, le meilleure intervalle de valeurs pour nos données existantes est (1, √2)).

Distance Minimale est un paramètre que nous introduisons pour trouver chaque contour, ce paramètre est très important, il est déterminé par nos données en TXT. Pour toutes nos données existantes, sa valeur doit être 1.

To Draw: Entrez "la coupe" pour obtenir la coupe à la hauteur Z; entrez "les contours" pour obtenir le contour de la coupe ; "all" pour obtenir la coupe et les contours.

L'interface

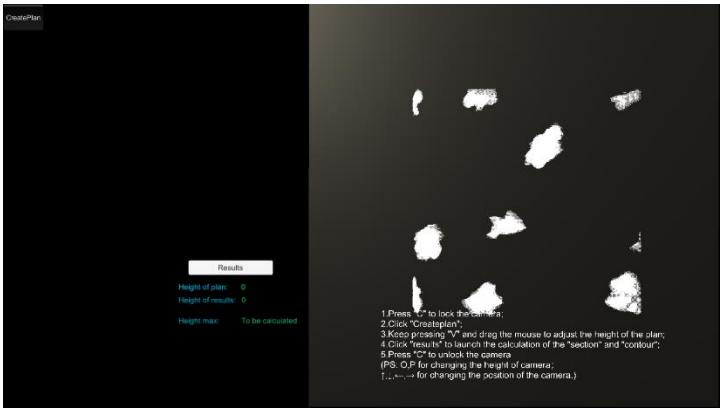


Figure 9 Les colonnes de 04_famille200*200 avant le regroupement ((a)vue de dessus, (b) vue perspective)

Dans le build, vous pouvez établir un plan en cliquant "Createplan" puis modifier sa hauteur par appuyer sur "V" et déplacer la souris en même temps. Cliquez sur le bouton "resultats", vous obtevez la coupe et les contours de la coupe de la hauteur arrondie du plan. La haueur maximum du nuage de points est affichée par l'ogiciel aussi. Pour balader dans les atomes, cliquez sur " ↑ , ↓ ,←,→, O et P".

Vous pouvez aussi calculer des autre fichiers TXT en remplaçant le fichier existant "04_famille200*200" sous le directoire "PVD_deposition_simulation_Data\StreamingAssets" avec votre fichier mais avec le même nom "04_famille200*200".

Résultats

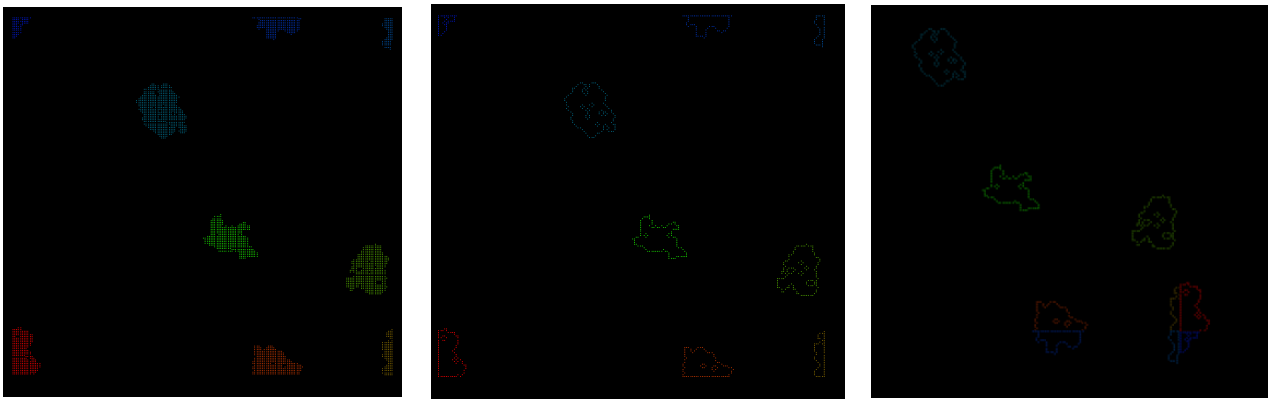
Avant terminer le regroupement, nous pouvons utiliser mesh pour faire les colonnes dans Unity. Voici les colonnes que nous avons obtenu:

(a) (b)

Figure 10 Les colonnes de 04_famille200*200 avant le regroupement ((a)vue de dessus, (b) vue perspective)

Comme la figure 2 dans l'annexe, nous pouvons d'abord obtenir le nombre de colonne(qui est 9 dans notre cas) lorsque la hauteur est 0. Ensuite, nous affichons les points de coordonnées dans chaque colonne(dans la figure 2 sont les coordonnées de la première colonne).

Après avoir obtenu le regroupement des colonnes de coup(Z=0), nous pouvons afficher le regroupement dans unity(la figure 11(a) au dessous). Nous affichons ensuite les coordonnées du centre de masse de chaque tranche de colonne (la figure dans l'annexe affiche les coordonnées du centre de masse de la neuvième colonne). Le centre de masse est très importante pour nous de trouver l'axe à l'avenir.



(a) (b) (c)

Figure 11 (a)La coupe de Z=0 après le regroupement ; (b)Les contours des 9 colonnes; (c) Les contours des 5 colonnes(après le déplacement);

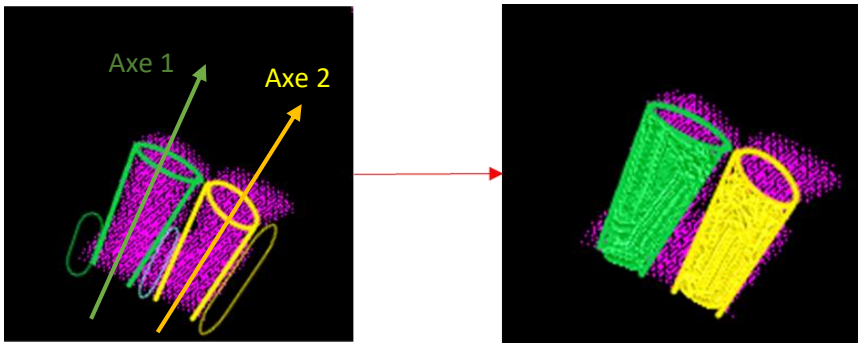
Ensuite, nous sortons les coordonnées de chaque contour (les coordonnées des huitième et neuvième contour du contour dans la figure 4 dans 1'annexe). Et nous pouvons tracer les contour dans la figure (b).

Nous pouvons voir que ces neuf colonnes ne sont en fait pas de véritables colonnes, certains d'entre eux sont séparés à la frontière, et ces neuf colones peuvent être réorganisés en cinq colonies par translation(voir la figure (c)).

Travaux futurs

Pour ce projet, nous avons encore pas mal de travail à faire :

1.Nous devons trouver l'axe de chaque colonne afin de pouvoir passer à l'extrusion des colonnes. Nous devons aussi trouver une solution pour absorber les atomes qui appartiennent à une même colonne.



2.Une fois l'extrusion est terminée, nous devons utiliser le composant « cloth » pour ajouter leur surface à chaque colonne.

3.L'optimisation des algos pour que logiciel puisse calculer plus vite.

4.L'ajout des fonctionnalités pour que l'interaction soit plus facile, par exemple: la facilité pour changer le ficher à calculer.

Conclusion

Projet Senior est le plus long projet de ce semestre, c'est aussi la première fois que je fais un projet de développement relativement indépendant. Je remercie beaucoup mes deux encadrement: Ruding et Aurélien. Tout d'abord, ils m'ont aidé à comprendre la PVD, un processus technique compliqué, en m'emmenant au laboratoire et en discutant entre eux. M. BESNARD a également fourni de nombreux documents connexes. Je voudrais tout particulièrement remercier M.LOU pour son aide importante dans la mise en œuvre du projet et des algorithmes utilisés en programmation.

Grâce à ce projet, j'ai appris beaucoup de nouvelles connaissances:

Par exemple, l'algorithme de classification d'un tas de données par programme; comment trouver un contour d'un point de nuage par programmation; Trouver l'axe d'une colonne généré par les coordonnées des points et extruder le contour en colonne; utiliser le component « cloth » pour ajouter une surface en colonne générée; l'ajout d'une capsule de perspective à la première personne…

Ces nouvelles connaissances acquises par ce projet m'aideront grandement dans mes études et mon travail futurs.

Bibliographie:

[1] https://www.sypopo.com/post/nNrPVWRYo7/

[2] Xiaojun Cheng, D. Jia, Y. Liu A Fast Contour Generation Algorithm of Massive Point Cloud Data [J]. Journal of Tongji University(Natural Science), 2012, 40(10): 1559-1563.

Annexe

Les figures :

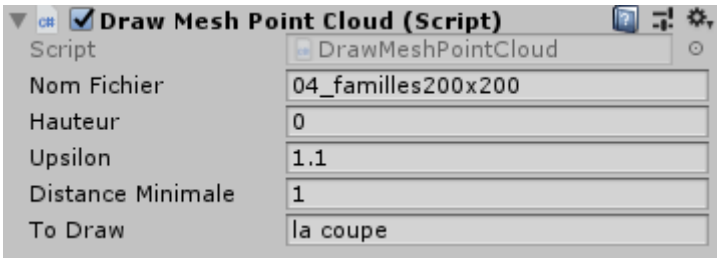


Figure 1 Les paramètres à saisir

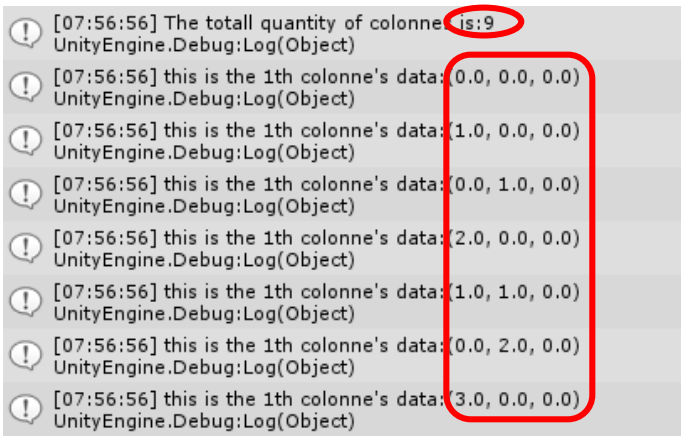


Figure 2 Le nombre de colonne et les points dans les colonnes

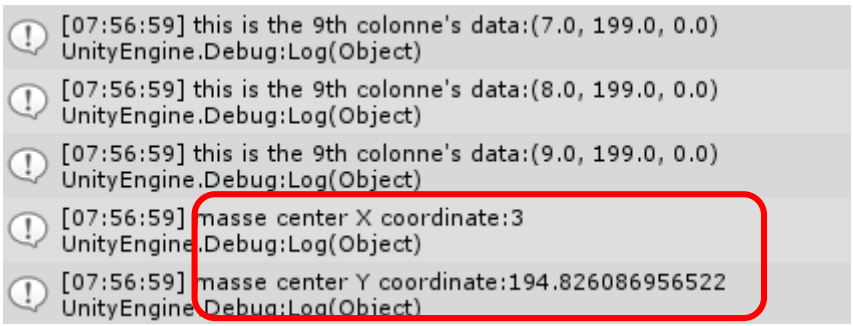


Figure 3 Le centre de masse dans la colonne 9

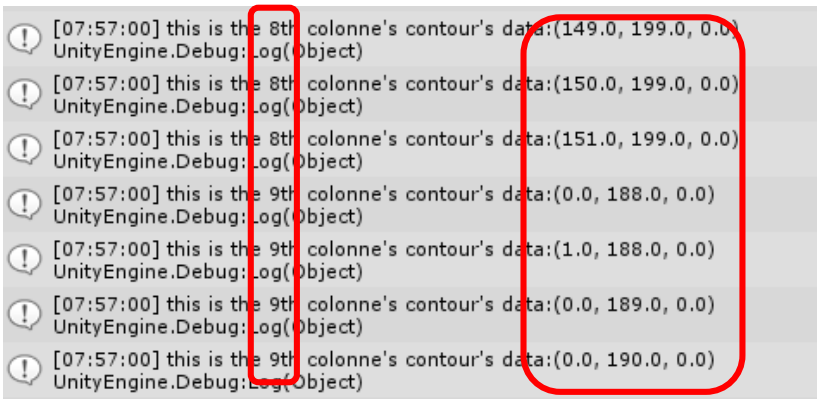


Figure 4 Les points de countour dans la colonne 9

Les scripts :

```
1.  using UnityEngine;
2.  using System.Collections;
3.  using System.IO;
4.  using System;
5.  using UnityEngine.UI;
6.
7.  public class DrawMeshPointCloud : MonoBehaviour
8.  {
9.      public string nomFichier;        //user input the name of a file
10.     public double hauteur;            //user defined the hight of a colonne
11.     public double upsilon;           //user defined the maximum distance between two
     atoms,to regroup the atoms in different colonnes
12.     public double distanceMinimale;  //user defined the minimum distance for finding
     the contour of a colonne(depend on the data in the file)
13.     public string toDraw;
14.     static public double hateurMax;
15.     ArrayList PointsRead = new ArrayList();
16.     ArrayList P = new ArrayList();   //The points at Z height
17.     ArrayList pointsGroups = new ArrayList();
18.
19.     ArrayList S = new ArrayList();//the array who stores the colonnes
20.     int sNB = 0;//colonnes number
21.     ArrayList pointsGroupsDansS = new ArrayList();//colonne in S
22.     ArrayList contourPointsGroups = new ArrayList();
23.
24.     public CreatePrimitive createPrimitive;
25.
26.     double XBar;
27.     double YBar;
28.
29.     public Transform parentContour;
30.     public Transform parentCoupe;
31.     public Transform parentAllPoints;
32.
33.
34.     int goCounter;
35.     public Text hauteurResults;
36.     public Text hauteurMaxi;
37.
38.     //GameObject pointObjAll;
39.     //GameObject pointObjContour;
40.     //public GameObject pointObjCoupe = new GameObject();
41.
42.     void Start()
43.     {
44.         // 1. Read file
45.         PointsRead = ReadFile();
46.         //To get the points at height Z
47.         ////P = ExtrairePoints();
48.         //Regroupe points
49.         /////pointsGroups = RegrouperPoints();
50.         //Find masse center
51.         FindMasseCenter();
52.         //Find the contour
53.         ///////contourPointsGroups = findcontour();
54.         CreateMesh();
55.         //if (toDraw == "tous les atomes")
56.         //CreateMesh();
57.         //////if (toDraw == "la coupe")
58.         //////    CreateCoupe();
```

```
59.         //////if (toDraw == "les contours")
60.         //////    CreateContour();
61.         ////if (toDraw == "all")
62.         ////{
63.         ////CreateMesh();
64.         ////CreateContour();
65.         ////CreateCoupe();
66.         ////}
67.     }
68.
69.     private void Update()
70.     {
71.
72.         ////if (toDraw == "all" && hauteur!= createPrimitive.Z)
73.         ////{
74.         ////    //P=ExtrairePoints();
75.         ////    //pointsGroups = RegrouperPoints();
76.         ////    //contourPointsGroups = findcontour();
77.         ////    //CreateContour();
78.         ////    //CreateCoupe();
79.         ////    renew();
80.         ////}
81.         ////hauteur = createPrimitive.Z;
82.         ////Debug.Log("这是 Z 的值" + createPrimitive.Z);
83.
84.     }
85.
86.     public void renew()
87.     {
88.         ////foreach (Transform child in parentGO)
89.         ////{
90.         ////    Destroy(child.gameObject);
91.         ////}
92.
93.         //hauteur += 1;
94.         if (toDraw == "all" && hauteur != Math.Round(createPrimitive.Z, 0))
95.         {
96.
97.
98.             foreach (Transform child in parentContour)
99.             {
100.                Debug.Log("ailzugeailzueg");
101.                Destroy(child.gameObject);
102.            }
103.
104.            foreach (Transform child in parentCoupe)
105.            {
106.                Debug.Log("ailzugeailzueg");
107.                Destroy(child.gameObject);
108.            }
109.
110.            Debug.Log("这是 Z 的值" + createPrimitive.Z);
111.            hauteur = Math.Round(createPrimitive.Z, 0);
112.            Debug.Log("这是 hauteur 的值" + createPrimitive.Z);
113.            hauteurResults.text = hauteur.ToString();
114.
115.            P.Clear();
116.            pointsGroups.Clear();
117.            contourPointsGroups.Clear();
118.
119.            P = ExtrairePoints();
120.            pointsGroups = RegrouperPoints();
121.            contourPointsGroups = findcontour();
122.            //Destroy(pointObjContour);
123.            //Destroy(pointObjCoupe);
```

```
                    //Destroy(pointCloudCoupe);
                    CreateContour();
                    CreateCoupe();
                    //hauteur = createPrimitive.Z;
                    Debug.Log("height max=" + hateurMax);
                    hauteurMaxi.text = hateurMax.ToString();
                    //Debug.Log("height in unity" + parentAllPoints.position.y) ;

                }

            }



        ArrayList ReadFile()
            {
                // save the points as TXT format file and put the file at Asset/StreamingAssets folder.
                string path = (Application.streamingAssetsPath + "/" + nomFichier+".txt");
                FileInfo fInfo = new FileInfo(path);
                float x, y, z;
                string s = "";
                StreamReader r;
                ArrayList pointsRead = new ArrayList();

                if (fInfo.Exists)
                {
                    r = new StreamReader(path);
                }
                else
                {
                    Debug.Log("File does not exist");
                    return null;
                }
                while ((s = r.ReadLine()) != null)
                {
                    string[] words = s.Split();
                    x = float.Parse(words[0], System.Globalization.CultureInfo.InvariantCulture.NumberFormat);
                    y = float.Parse(words[1], System.Globalization.CultureInfo.InvariantCulture.NumberFormat);
                    z = float.Parse(words[2], System.Globalization.CultureInfo.InvariantCulture.NumberFormat);
                    Vector3 xyz = new Vector3(x, y, z);

                    pointsRead.Add(xyz);
                }
                return pointsRead;
            }
            //  update*****************************************************************************
            ArrayList ExtrairePoints()
            {
                Vector3 ZPoints;
                ArrayList pointsHauteurZ = new ArrayList();
                bool ifhauteurTrue = false;
                for (int i = 0; i < PointsRead.Count; i++) //Read all points
                {
                    ZPoints = (Vector3)PointsRead[i];
                    if (ZPoints[2] == hauteur)
                    { pointsHauteurZ.Add(ZPoints); ifhauteurTrue = true;     }

                    if (ZPoints[2] > hateurMax)
```

```
184.                     {
185.                         hateurMax = ZPoints[2];
186.                     }
187.
188.                 }
189.                 if (ifhauteurTrue == false)
190.                 {
191.                     Debug.Log("Height bad input, please relaunch the application");
192.                 }
193.                 return pointsHauteurZ;
194.
195.
196.             }
197.             //  update*************************************************************
    **************
198.             ArrayList RegrouperPoints()
199.             {
200.
201.                 Vector3 p1 = new Vector3();
202.                 while (P.Count != 0)
203.                 {
204.                     ArrayList G = new ArrayList();  //initialize G everytime finish a
    group
205.                     p1 = (Vector3)P[0];//the elements in arraylist are object type, n
    eed to transfer to vector3
206.                     G.Add(p1);
207.                     P.Remove(p1);
208.                     for (int i = 0; i < G.Count; i++)
209.                     {
210.                         Vector3 m = (Vector3)G[i];
211.                         ArrayList mNeighborPoints = new ArrayList();  //find m'neighb
    or
212.
213.                         for (int j = 0; j < P.Count; j++)
214.                         {
215.                             Vector3 n = (Vector3)P[j];
216.                             double distance = Math.Sqrt((m[0] - n[0]) * (m[0] - n[0])
    + (m[1] - n[1]) * (m[1] - n[1]) + (m[2] - n[2]) * (m[2] - n[2]));
217.                             bool exist = G.Contains(n);
218.                             if (distance < upsilon && !exist)
219.                             {
220.                                 G.Add(n);
221.                                 mNeighborPoints.Add(n);
222.                             }
223.                         }
224.
225.                         for (int j = 0; j < mNeighborPoints.Count; j++)
226.                             P.Remove(mNeighborPoints[j]);
227.                     }
228.
229.
230.                     S.Add(G);
231.                     sNB++;
232.                 }
233.                 Debug.Log("The totall quantity of colonnes is:" + sNB);
234.                 return S;
235.             }
236.             //  update*************************************************************
    **************
237.             void FindMasseCenter()
238.             {
239.                 double XSomme;
240.                 double YSomme;
241.                 double XCoodinateDansUnGroupe;
242.                 double YCoodinateDansUnGroupe;
243.                 Vector3 pointCoordinate;
```

```
244.                //find the masse center
245.                for (int i = 0; i < S.Count; i++)
246.                {
247.                    int q = i + 1;
248.                    pointsGroupsDansS = (ArrayList)S[i];
249.                    XSomme = 0;
250.                    YSomme = 0;
251.                    for (int j = 0; j < pointsGroupsDansS.Count; j++)
252.                    {
253.                        Debug.Log("this is the " + q + "th " + "colonne's data:" + po
    intsGroupsDansS[j]);
254.                        pointCoordinate = (Vector3)pointsGroupsDansS[j];
255.                        XCoodinateDansUnGroupe = pointCoordinate[0];
256.                        YCoodinateDansUnGroupe = pointCoordinate[1];
257.                        XSomme += XCoodinateDansUnGroupe;
258.                        YSomme += YCoodinateDansUnGroupe;
259.                    }
260.                    XBar = XSomme / pointsGroupsDansS.Count;
261.                    YBar = YSomme / pointsGroupsDansS.Count;
262.                    Debug.Log("masse center X coordinate:" + XBar);
263.                    Debug.Log("masse center Y coordinate:" + YBar);
264.                }
265.            }
266.            //  update*************************************************************
    ***************
267.            ArrayList findcontour()
268.            {
269.                Vector3 points=new Vector3();
270.                double distance;
271.                ArrayList contourGroups = new ArrayList();
272.                for (int i = 0; i < pointsGroups.Count; i++)
273.                {
274.                    ArrayList contourPoints = new ArrayList();
275.                    ArrayList T = (ArrayList)pointsGroups[i];
276.                    for (int k = 0; k < T.Count; k++)
277.                    {
278.                        Vector3 p = (Vector3)T[k];
279.                        int density = 0; //to observe how many atoms here are
280.                        for (int j = 0; j < T.Count; j++)
281.                        {
282.                            points = (Vector3)T[j];
283.                            distance = Math.Sqrt((points[0] - p[0]) * (points[0] - p[
    0]) + (points[1] - p[1]) * (points[1] - p[1]));
284.                            if (distance <= distanceMinimale)
285.                            { density++; }
286.                        }
287.                        if (density < 5)
288.                        {
289.                            contourPoints.Add(p);
290.                        }
291.                    }
292.                    contourGroups.Add(contourPoints);
293.                }
294.
295.                for (int i = 0; i < contourGroups.Count; i++)
296.                {
297.                    int q = i + 1;
298.                    ArrayList KKK = (ArrayList)contourGroups[i];
299.                    for (int j = 0; j < KKK.Count; j++)
300.                    { }   //Debug.Log("this is the " + q + "th " + "colonne's contour
    's data:" + KKK[j]);
301.                }
302.                return (contourGroups);
303.            }
304.
305.            void CreateContour()
```

```
306.           {
307.               Material mat = Resources.Load("MaterialPoint", typeof(Material)) as M
    aterial;
308.               for (int j = 0; j < contourPointsGroups.Count; ++j)
309.               {
310.                   ArrayList list = (ArrayList)contourPointsGroups[j];
311.                   int num = list.Count;
312.                   GameObject pointObj = new GameObject();
313.                   pointObj.name = "point cloud Contour " + goCounter.ToString();
314.                   goCounter++;
315.                   pointObj.transform.parent = parentContour;
316.                   pointObj.transform.position = new Vector3(105, 100 , -300);
317.                   pointObj.transform.rotation = Quaternion.Euler(new Vector3(0, 0,
    0));
318.                   pointObj.AddComponent<MeshFilter>();
319.                   pointObj.AddComponent<MeshRenderer>();
320.                   pointObj.GetComponent<MeshRenderer>().material = mat;
321.                   Mesh meshNeed = new Mesh();
322.                   pointObj.GetComponent<MeshFilter>().mesh = meshNeed;
323.                   Vector3[] points = new Vector3[num];
324.                   Color[] colors = new Color[num];
325.                   int[] indecies = new int[num];
326.                   for (int i = 0; i < num; ++i)
327.                   {
328.                       points[i] = (Vector3)list[i];
329.                       indecies[i] = i;
330.                       if (j < (float)pointsGroups.Count / 2)
331.                           colors[i] = Color.Lerp(Color.red, Color.green, 2 * (float
    )j / (float)pointsGroups.Count);
332.                       else
333.                           colors[i] = Color.Lerp(Color.green, Color.blue, (float)j
    / (float)pointsGroups.Count);
334.                   }
335.                   meshNeed.vertices = points;
336.                   meshNeed.colors = colors;
337.                   meshNeed.SetIndices(indecies, MeshTopology.Points, 0);
338.               }
339.           }
340.           void CreateCoupe()
341.           {
342.               Material mat = Resources.Load("MaterialPoint", typeof(Material))as Ma
    terial;
343.               for (int j = 0; j < pointsGroups.Count; ++j)
344.               {
345.                   ArrayList list = (ArrayList)pointsGroups[j];
346.                   int num = list.Count;
347.
348.                   GameObject pointObj = new GameObject();
349.                   pointObj.name = "point cloud Coupe";
350.                   pointObj.transform.parent = parentCoupe;
351.                   pointObj.transform.position = new Vector3(-105, 100, -300);
352.                   pointObj.transform.rotation = Quaternion.Euler(new Vector3(0, 0,
    0));
353.                   pointObj.AddComponent<MeshFilter>();
354.                   pointObj.AddComponent<MeshRenderer>();
355.                   pointObj.GetComponent<MeshRenderer>().material = mat;
356.                   Mesh meshNeed = new Mesh();
357.                   pointObj.GetComponent<MeshFilter>().mesh = meshNeed;
358.                   Vector3[] points = new Vector3[num];
359.                   Color[] colors = new Color[num];
360.                   int[] indecies = new int[num];
361.                   for (int i = 0; i < num; ++i)
362.                   {
363.                       points[i] = (Vector3)list[i];
364.                       indecies[i] = i;
365.                       if (j < (float)pointsGroups.Count / 2)
```

```
366.                             colors[i] = Color.Lerp(Color.red, Color.green, 2 * (float
    )j / (float)pointsGroups.Count);
367.                         else
368.                             colors[i] = Color.Lerp(Color.green, Color.blue, (float)j
    / (float)pointsGroups.Count);
369.
370.                     }
371.                     meshNeed.vertices = points;
372.                     meshNeed.colors = colors;
373.                     meshNeed.SetIndices(indecies, MeshTopology.Points, 0);
374.                 }
375.             }
376.
377.             void CreateMesh()
378.             {
379.                 Material mat = Resources.Load("MaterialAllPoint", typeof(Material)) a
    s Material;
380.                 int num = PointsRead.Count;
381.                 GameObject pointObj = new GameObject();
382.                 pointObj.name = "point cloud all atoms";
383.                 pointObj.transform.parent = parentAllPoints;
384.                 pointObj.transform.rotation = Quaternion.Euler(new Vector3(270, 0, 0)
    );
385.                 pointObj.AddComponent<MeshFilter>();
386.                 pointObj.AddComponent<MeshRenderer>();
387.                 pointObj.GetComponent<MeshRenderer>().material = mat;
388.                 Mesh meshNeed = new Mesh();
389.                 pointObj.GetComponent<MeshFilter>().mesh = meshNeed;
390.                 Vector3[] points = new Vector3[num];
391.                 Color[] colors = new Color[num];
392.                 int[] indecies = new int[num];
393.                 for (int i = 0; i < num; ++i)
394.                 {
395.                     points[i] = (Vector3)PointsRead[i];
396.                     indecies[i] = i;
397.                     colors[i] = Color.Lerp(Color.white, Color.gray, 0);
398.                 }
399.                 meshNeed.vertices = points;
400.                 meshNeed.colors = colors;
401.                 meshNeed.SetIndices(indecies, MeshTopology.Points, 0);
402.             }
403.         }
```

Arts et Métiers Sciences et Technologies

Contact

Mail

Téléphone

Adresse