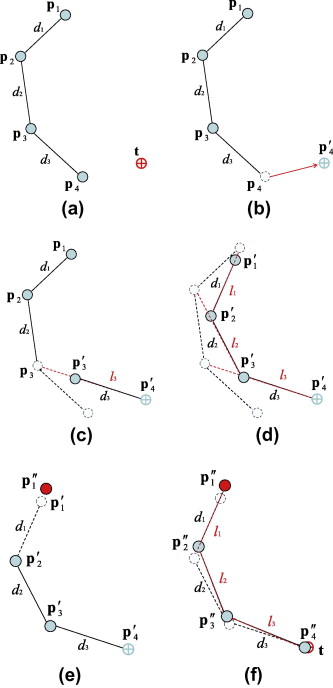
MOULARD Valentin

Simulation Physique.

Pour la simulation de squelette, j’ai créé en Unity, un objet nœud qui sera lié à un autre nœud donné en paramètre. Celui-ci devra respecter une contrainte de longueur et une contrainte d’angle. Les nœuds ne pourront pas s’espacer de plus d’une certaine longueur et ne pourront pas pivoter au-delà d’un certain angle.

Ensuite, il a fallu gérer les déplacements de tous les nœuds tout en respectant les contraintes définies. Voici un schéma illustrant comment les nœuds sont déplacés.

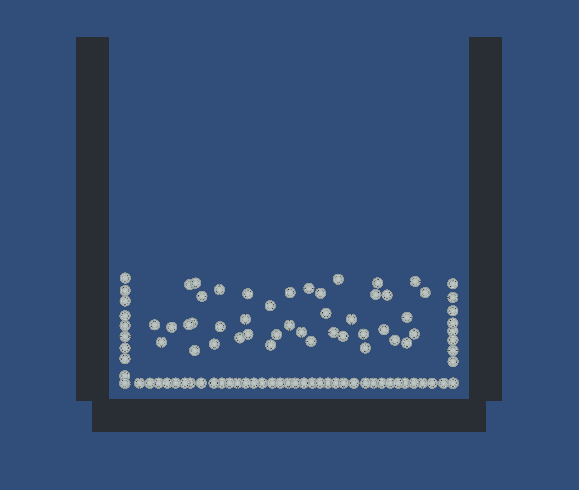


Quand on déplace un nœud p4 vers une destination *t*, on rapproche le nœud p3 qui lui est attaché pour respecter les contraintes. On fait de même pour les points p2 et P1. Cependant le point p1 ne se trouve pas au bon endroit. Donc on réitère l’opération mais dans le sens inverse. Si on suit cette méthode, on n’arrivera jamais au déplacement ‘parfait’ car à chaque fois qu’on répète cette opération de déplacement des nœuds, soit p1 ou p4 ne seront à l’endroit parfait. On intègre donc une certaine marge. On limite donc le processus de déplacement des nœuds à 4 itérations pour avoir un résultat ‘acceptable’.

* Simulation de particules
* Interactions entre particules et bords
* Structure accélératrice en en créant une liste de voisins selon un rayon donné pour chaque particule.

Pour la simulation de fluide, j’ai créé une box avec 3 bords (bas, gauche et droite) dans laquelle seront mes particules. J’ai créé un objet particule qui sera instanciée dans la box à plusieurs reprises. Celle-ci interagit avec les autres particules pour simuler un comportement de fluide. En se basant sur l’algorithme du document ‘Particle-based Viscoelastic Fluid Simulation’, j’ai implémenté la gravité, la sauvegarde de la position précédente (pour le calcul de la vitesse : v = pos en t – pos en t-1), la gestion des collisions avec les bords, le calcul de la densité désirée et la mise à jour de la vitesse de la particule.

Le calcul de la densité se fait en prenant en compte le nombre de particule voisin d’une particule dans un certain rayon. Pour optimiser les calculs et éviter de vérifier toutes les particules pour calculer la densité, je crée une liste de voisin pour chacune des particules. Cette liste est créée en ajoutant les particules voisines d’une particule comprise dans un certain rayon. Au lieu de vérifier toutes les particules, on vérifie que les particules voisines, ce qui réduit le nombre de vérification à faire.



Unity commençait à ramer à partir de 200 particules environ. Avec mon optimisation, Unity commençait à ramer à partir de 1100 particules.