IUT de Bordeaux
Semestre 2

Année 2016 - 2017

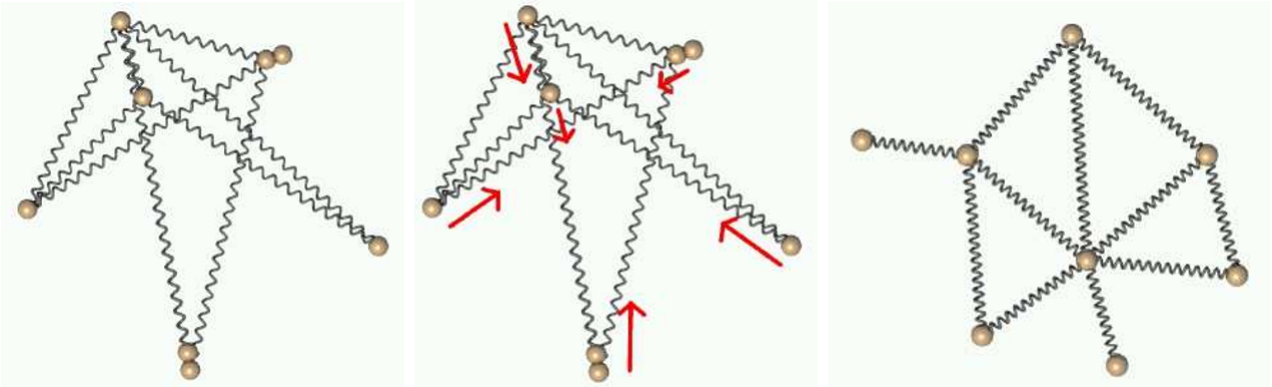
Projet de M2201
- Dessin de graphe -

Objectif : Nous allons lors de ce projet mettre en place un algorithme de dessin de graphe par modèle de forces.

Fichiers fournis : Vous trouverez dans ~/Bibliotheque/M2201/Projet un répertoire contenant un fichier projet Qt (Graphe.pro).

Commencez par identifier les classes qui vous sont fournies. A priori, il ne sera pas nécessaire de modifier ces classes et tout le code que vous produirez pourra être ajouté dans le fichier main.cpp.

Principe des algorithmes par Modèle de Forces



Le principe des algorithmes par modèle de forces est de simuler un modèle physique masse-ressort (voir fr.wikipedia.org/wiki/Système_masse-ressort). Dans cette simulation, les sommets du graphe sont considérés comme des objets physiques et deux objets physiques sont reliés par un ressort si les sommets du graphe correspondants sont reliés par une arête.
Ces algorithmes suivent le schéma général suivant :

```
Initialiser la position des sommets
Répéter n fois :
    Pour chaque sommet v :
            Calculer les forces d'attractions appliquées sur le sommet v par ses voisins
        Calculer les forces de répulsions appliquées sur le sommet v par tous les autres sommets
        Déplacer le sommet v en fonction de ces forces
```

Partie I

Suivant l'algorithme utilisé, le calcul des forces n'est pas le même (utilisation de différents modèles physiques). Dans ce projet, nous utiliserons le modèle suivant :

- Calcul de la force d'attraction exercée sur le sommet *v* par ses voisins :

$$f_{attr}(v)=\sum_{u\in Neigh(v)}\frac{dist_R(u,v)^2}{length^2}*p(uv)$$

où *Neigh(v)* est l'ensemble des voisins de *v*, *length* un paramètre représentant la longueur d'arête idéale (généralement une constante prédéfinie) et *p(uv)* le vecteur unitaire de *u* à *v*.

- Calcul de la force de répulsion exercée sur le sommet *v* par tous les autres sommets :

$$f_{rep}(v)=\sum_{u\in V,u\neq v}\frac{length}{dist_R(u,v)^2}*p(vu)$$

- Calcul de la force totale exercée sur sommet *v* :

$$f_{totale}(v)=f_{attr}(v)+f_{rep}(v)$$

Quelques rappels :

Soit $u=(x_u, y_u)$ et $v=(x_v, y_v)$ deux coordonnées dans un espace 2D, le vecteur *uv* est défini par :

$$uv=(x_v-x_u, y_v-y_u)$$

On appelle vecteur unitaire tout vecteur de norme égale à 1. Et, le vecteur unitaire *de u à v*, noté *p(uv)* dans ce document, est défini par :

$$p(uv)=uv/norme(uv)$$

où la norme du vecteur *uv* est défini par :

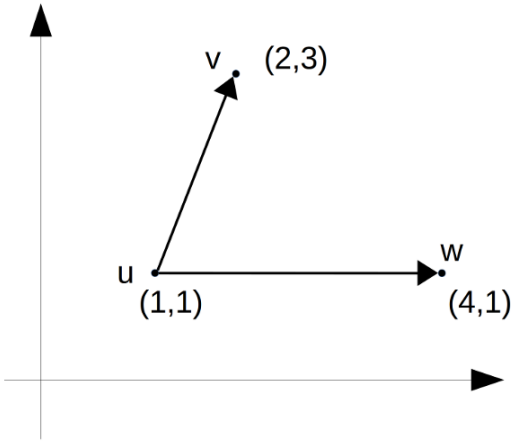
$$norme(uv)=\sqrt{(x_v-x_u)^2+(y_v-y_u)^2}$$

Dans l'exemple ci-contre,
uv=(1,2) et *vu=(-1,-2)*
uw=(3,0) et *wu=(-3,0)*

Et la norme de uw $norme(uw)=\sqrt{3^2+0^2}=3$

Enfin, le vecteur unitaire de u à *w* est :

$$p(uw)=uw/3=(1,0)$$



Note : Dans le code, la classe Coord permet de représenter les coordonnées des sommets mais peut aussi être utilisée pour représenter des vecteurs.

Question 1 : Initialisation aléatoire des positions
Implémentez la fonction void initialiserDessin(Graphe & g, unsigned int largeur, unsigned int hauteur) qui initialise aléatoirement toutes les positions des sommets du graphe g entre (0,0) et (largeur, hauteur).

Question 2 : Calcul des forces et déplacement
Implémentez les fonctions :

1. Coord calculerAttractions(const Graphe &g, Sommet v) qui retourne la force d'attraction exercée sur *v* par ses voisins.
2. Coord calculerRepulsions(const Graphe & g, Sommet v) qui retourne la force de répulsion exercée sur *v* par tous les autres sommets.
3. Coord calculerForces(const Graphe &g, Sommet v) qui retourne la force totale exercée sur *v*.
4. void deplacer(Graphe &g, Sommet v, Coord deplacement) qui effectue une translation de vecteur deplacement du sommet *v*.

Question 3 : Ajout d'une force de gravité
Pour éviter que le dessin de soit trop étalé, nous pouvons ajouter au modèle de forces une force de gravité qui attire tout sommet vers le barycentre sur dessin. La force de gravité exercée sur un sommet v est alors :

$$f_{gravity}(v) = grav * (p(bary\, v))$$

où *grav* est une constante strictement plus petite que 1 (vous pouvez utiliser la constante GRAVITE donnée dans le fichier), et *bary* est le barycentre actuel du dessin. La force totale exercée sur le sommet v devient alors :

$$f_{totale}(v) = f_{attr}(v) + f_{rep}(v) + f_{gravity}(v)$$

1. Implémentez la fonction calculerBarycentre qui calcule le barycentre des positions des sommets du graphe.
2. Implémentez la fontion calculerForceGravite qui calcule la force de gravité exercée sur un sommet donné.
3. Modifiez votre algorithme pour qu'il prenne en compte cette force de gravité.

Question 4 : Limitation du déplacement
Une manière de réduire les « sauts » dans le dessin est de limiter les déplacements des sommets. Vous pouvez pour cela utiliser les constantes MAX_ATTRACTIVE et MAX_REPULSIVE qui représente les normes maximales des forces d'attraction et de répulsion appliquées sur chaque sommet. Il est aussi possible de borner la norme du vecteur de déplacement, pour cela utilisez la constante MAX_DEPLACEMENT. Modifiez le code en conséquence.

Question 5 : Vers une meilleure initialisation
L'un des problèmes des algorithmes par modèle de forces est que la qualité du résultat dépend clairement du placement initial. Pour l'instant vous utilisez comme placement initial des positions tirées aléatoirement. Pour améliorer ce placement initial, implémentez la fonction void initialiserIntelligementDessin(Graphe & g, unsigned int largeur, unsigned int hauteur)
qui positionne aléatoirement les sommets du graphe puis, pour chaque sommet, le déplace au barycentre des positions de ces voisins.

Question 6 : « Ralentir » les déplacements
Afin d'améliorer le dessin, le déplacement maximum de chaque sommet diminue au cours du temps, c'est-à-dire au cours des itérations. Modifiez votre code afin de le prendre en compte.

Partie II

Dans cette partie, vous devrez créer un deuxième projet à nous remettre (vous pouvez utiliser le projet précédent comme point de départ). Il s'agit ici d'implémenter un autre algorithme par modèle de forces utilisant non plus uniquement les distances dans le plan et une longueur idéale (*length*) mais aussi des distances dans le graphe (nombre d'arêtes d'un plus court chemin). Le calcul de la force totale exercée sur v devient alors :

$$f_{totale}(v) = \sum_{u \in V, u \neq v} \frac{dist_R(u,v)}{length^2 * dist_G(u,v)} * p(vu)$$

Question 7 : Calcul de distances
Implémenter les fonctions :

1. `void calculerDistances(const Graphe &g, Sommet v, map<Sommet, int> & distances)` qui calcule et stocke dans *distances* les distances dans le graphe du sommet *v* aux autres sommets.
2. `void calculerDistances(const Graphe &g, map<Sommet, map<Sommet, int> > & distances)` qui calcule et stocke dans *distances* les distances dans le graphe entre toute paire de sommets.

Question 8 :
Modifiez le code afin de prendre en compte ce nouveau modèle de forces utilisant les distances dans le graphe.