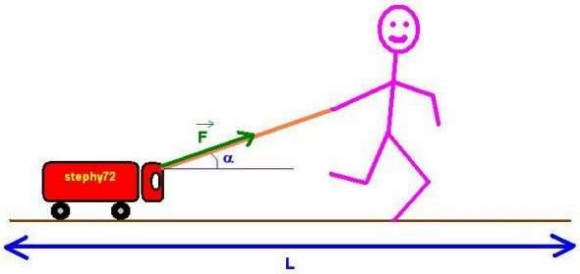
M1 Animation
Système de particules / Masse-Ressort

Objectifs : Notion de mécanique du point
Seconde loi de Newton et intégration

Avec le principe fondamental de la dynamique, ça bouge !

1. Rappelez la deuxième loi de Newton.



Jusqu'à présent (Terminale) vous avez toujours supposé que les forces étaient constantes, par exemple uniquement la gravité. Mais il y a de nombreuses situations où les forces varient en fonction du temps. Imaginez par exemple une caisse poussée par 3 personnes en même temps, ces 3 personnes ne poussent pas de manière continue. Parfois l'une se repose un peu, une autre pousse dans une direction un peu différente, etc. La trajectoire de la caisse ne peut donc pas se déduire de manière analytique en faisant l'intégrale analytiquement. Il faut faire une intégration discrète, c'est-à-dire en découpant le calcul par petits intervalles de temps. On dira aussi « pas de temps ».

Nous allons noter p_t la position au temps t, v_t la vitesse au temps t et at l'accélération au temps t. L'accélération est la variation de la vitesse. Soit dt un intervalle de temps, petit, par exemple 10^{-3}seconde (0.001 seconde). L'accélération est la variation de la vitesse :

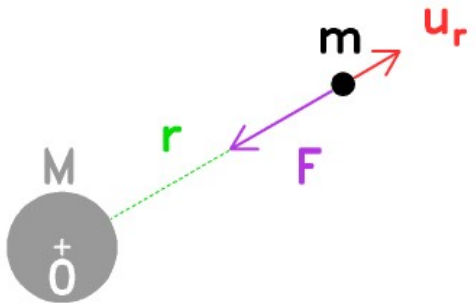
$$\frac{dv_t}{dt} = \frac{v_t - v_{t-dt}}{dt} = a_t$$

Nous pouvons faire de même pour la vitesse qui est la variation de position :

$$\frac{p_t - p_{t-dt}}{dt} = v_t$$

2. Calculez la nouvelle position en fonction de la vitesse et de la position au pas de temps précédent.
3. Calculez la mise à jour de la vitesse en fonction de la vitesse au pas de temps précédent et de l'accélération.
4. A partir de ces deux expressions et du principe fondamental de la dynamique, calculez la position d'une particule en fonction des forces subits au temps t.

Et si on codait ça ?



La position d'une particule se trouvant au point P peut s'écrire comme étant le vecteur OP, un vecteur à 2 dimensions dans le cas du plan que nous allons traiter ici. La vitesse de la particule est également un vecteur.

On supposera que la classe *Vector* qui stocke un vecteur à 3 dimensions munis des opérator+,-*,/

5. Ecrivez en C++ la classe Particle qui stocke les informations nécessaires à une simulation par le second principe de la dynamique (2e loi de Newton). Une particule est représentée par une position, une vitesse, une force et une masse. Si la particule subit plusieurs forces, les vecteurs forces sont sommés.

6. Ecrivez la procédure *void addForce(const Vector& force)* qui ajoute une force à la particules.

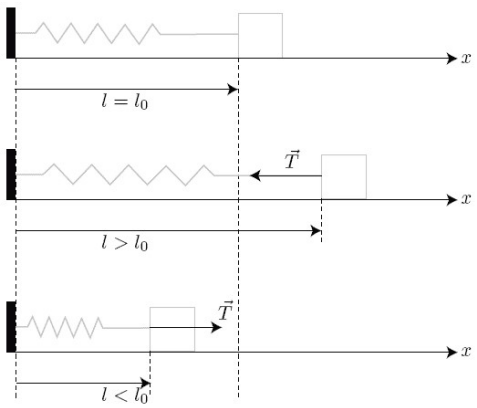
7. Ecrivez la procédure *void groundCollision()* qui résoud une collision avec le sol (y=0)

8. Ecrivez la procédure *void update* qui met à jour la vitesse et la position d'une particule.

9. Ecrivez la classe *World* qui contient un tableau (std ::vector) de particules. Ecrivez sa fonction update qui calcule le déplacement des particules en leur faisant subir la gravité terrestre.

Masses-Ressorts

Nous allons allez un peu plus loin avec la manipulation de particules en mouvements en connectant deux particules ensembles avec un ressort.

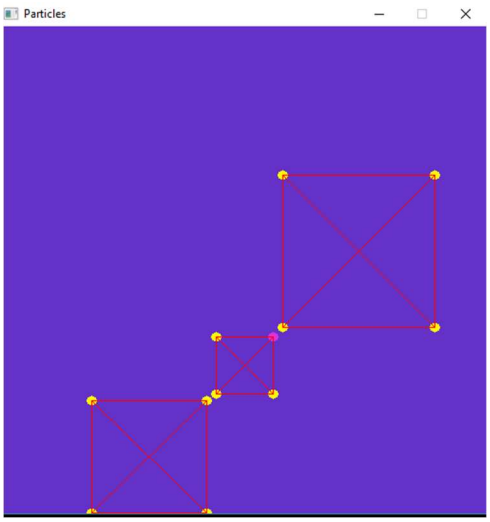


La force de rappel élastique exercée par le ressort sur la masse m est

$$T = -k \cdot \Delta l \cdot e_r$$

où:

- e_r est un vecteur unitaire dirigé suivant l'axe du ressort, orienté vers l'extérieur.
- $\Delta l = l - l_0$ est l'allongement du ressort en notant l la longueur du ressort, et l_0 sa longueur à vide.
- k est la **raideur du ressort**, intrinsèque au ressort considéré, elle caractérise la capacité du raideur à résister au mouvement de la masse m, et par conséquent à revenir à sa position d'équilibre.



14. Définissez une classe *Spring* (ressort) s'adossant à la structure *PhysicalWorld* contenant l'ensemble des particules du monde. Vous pourrez utiliser l'indice d'une particule dans le tableau comme identifiant. Un Spring contient les deux indices des particules, la longueur au report et la raideur du ressort.

15. Ajoutez un tableau (std ::vector) de ressorts à la structure du monde *PhysicalWorld*

16. Ajoutez à la classe PhysicalWorld les procédures addLine, addSquare, addCube, etc. qui ajoute N particules et R ressorts en fonction de la forme demandée. Nous vous conseillons d'écrire une structure *addParticle* qui renvoie l'indice de la particule ajoutée.

```
int addParticle(const Vector& p) ;
void addLine(const Vector& p1, const Vector& p2);
void addCube(const Vector& center, float l);
```

17. Modifiez la procédure *draw* pour qu'elle affiche les ressorts représentés par des lignes.

18. Ecrivez la procédure *computeParticleForceSpring* qui calcule les forces qu'exercent les ressorts sur toutes les particules du monde.

Loi universelle de la gravitation (Pour aller plus loin)

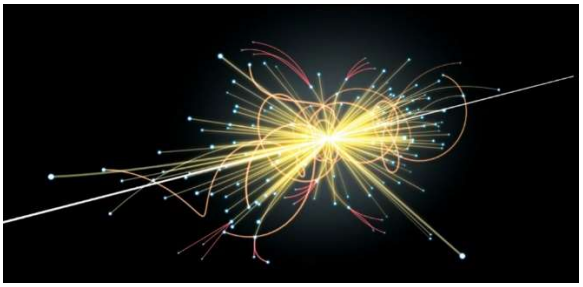
1. Deux particules interagissent par la force de gravité. Notons M et m leur deux masses respectives (en kg), r leur distance (en m), u_r étant le vecteur unitaire entre les deux particules. La force F que subit chaque particule est

$$\mathbf{F} = -\frac{\mathcal{G}Mm}{r^2}\,\mathbf{u}_\mathrm{r}$$

   Modifiez votre programme pour que les particules ne subissent plus la gravité terrestre mais plutôt pour que chaque particule se comportent comme un astre subissant la gravité de tous les autres astres et provoquant une force de gravité vers tous les autres astres.

2. En premier test vous simulerez uniquement 2 particules : un soleil au centre avec une masse assez élevée et une particule positionnée à une distance de 100 pixels.
   Quelle trajectoire allez-vous observer ?
   Modifiez votre programme pour que la petite particule tourne autour de la grande.

3. Testez plusieurs configurations de particules avec différentes masses. Essayez de retrouver quelque chose qui ressemble au système solaire.
   Masse_lune = 0,0123 * Masse_terre
   Masse_soleil = 330000 * Masse_terre



Collision entre un tissu et un objet quelconque (Pour aller plus loin++)

gKit2light vous permet de charger des objets au format obj. Modifiez votre programme pour qu'il détecte et réagisse aux collisions entre un tissu et un objet statique dont la forme est un maillage quelconque.