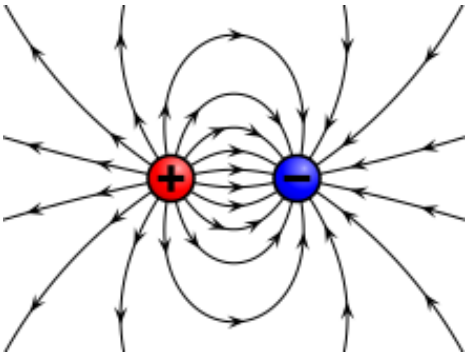
Arthur Cayuso

Physique appliquée
Cours de Orestis
28.06.2021
ISC 1ère - Soir

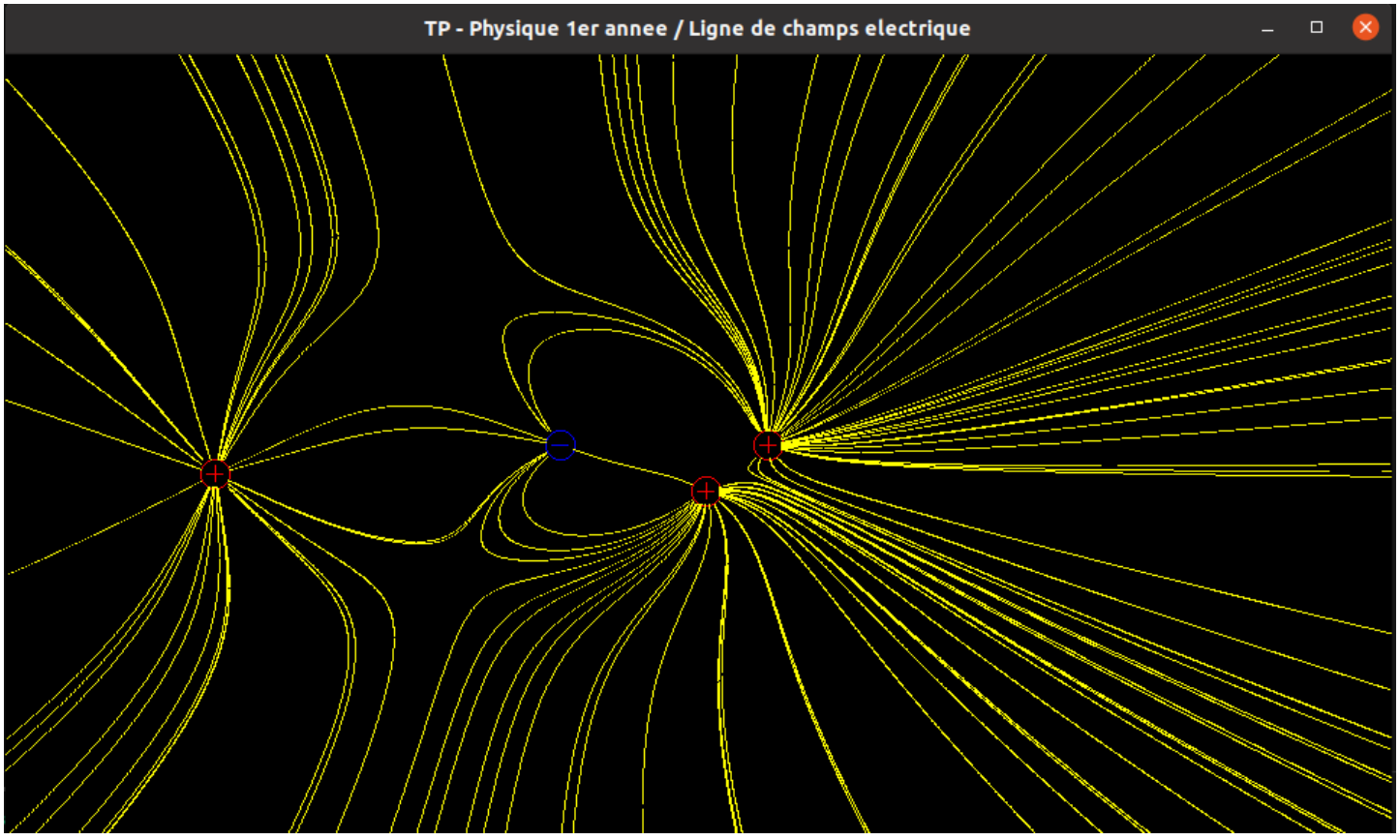
TP physique
Ligne de champs électrique



Contexte :

Créer une simulation d'un champ électrique avec une charge positive et négative dans le langage C.
Nous savons que chaque charge Q induit un champ électrique E qui modifie l'espace autour d'elle. Donc le but sera de visualiser nos lignes de champs électriques générées par N particules.

Rendu final :



Synthèse :

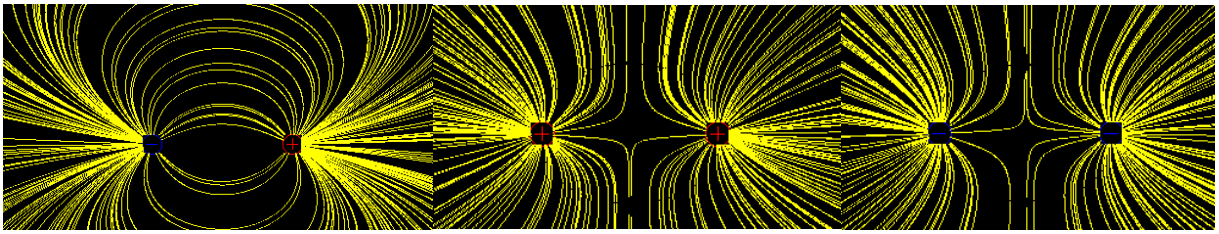
Matériels utilisés

Machine virtuelle : VMWare Workstation 16 avec Ubuntu 20.04
Langage utilisé : langage C

Le champ d'électrique

Nous avons vu dans le cours qu'une charge chargée électriquement positive ou négative peut exercer une force d'attraction ou de répulsion sur l'autre charge.

Dans notre cas nous allons utiliser l'exemple de deux charges où l'on va changer son potentiel.
Étude simulée dans mon programme



Le champ électrique s'attire dans ce cas, puis s'oppose, car il y a deux charges positives et réciproquement pour deux charges négatives.

Approche mathématique et physique pour le rendu

J'ai utilisé les formules fournis tels que :

$$E = k\frac{Q}{r^2} \; avec \; k = \frac{1}{4\pi\epsilon_0}$$

Pour créer une charge, nous les avons ajoutés dans une liste (tableau). Chaque charge possède les propriétés suivantes:

- **Capacité de la charge**
- **Position x et Position y**

```
typedef struct{
   double q ;
   vec pos ;
} charge_t ;
```

Notre cas, j'ai décidé la position de nos deux charges et de leur fixer leur potentiel.

```
charges[0].q = 1/ +EC; // Plus la charge est grande, plus d'influance
charges[0].pos.x = 0.4;
charges[0].pos.y = 0.5;

charges[1].q = 1/ -EC;
charges[1].pos.x = 0.55;
charges[1].pos.y = 0.5;
```

Pour dessiner les lignes de champs électriques, j'ai créé un point aléatoire dans notre univers discret. C'est-à-dire que l'univers est connu. Donc dans le cadre de ce travail, il a été fixé d'avoir un rectangle de [0,1] par [0,1].

Par conséquent lorsque l'on créer un point dans notre univers et que l'on travaille qu'avec deux charges qui sont attirées.

CHARGE (+) --------°-------->CHARGE (-)

Nous devons relier le point à la charge positive et rebrousser le chemin pour relier le point a la charge négative.
Pour cela, on calcule la valeur du champ électrique de ce point en prenant en compte la globalité de toutes les charges de notre univers.

Donc une fois l'étape de la position du point, comme demandé on assigne P = P0 et on calcule le Psuivant.
Le Psuivant est calculé grâce aux formules fournies dans le TP qui sont :

$$P_{suivant} = P + \Delta x \frac{E}{||E||}$$

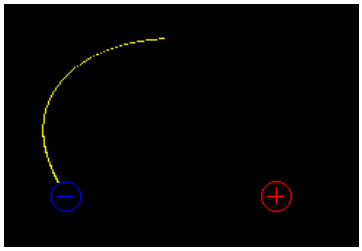
$$\Delta x = \frac{1}{\sqrt{largeur^2 + hauteur^2}}$$

Les formules utilisent des vecteurs, c'est donc là que la librairie que l'on a programmée en début d'année est réutilisée.

```
vec vec_create(double x, double y);
void vec_print(vec v, char *name);
vec vec_add(vec lhs, vec rhs);
vec vec_sub(vec lhs, vec rhs);
vec vec_mul(vec v, double alpha);
vec vec_div(vec lhs, double alpha);
double vec_scalar_product(vec lhs, vec rhs);
double vec_norm_sqr(vec v);
double vec_norm(vec v);
double vec_distance(vec v1, vec v2);
double sqr(double a);
```

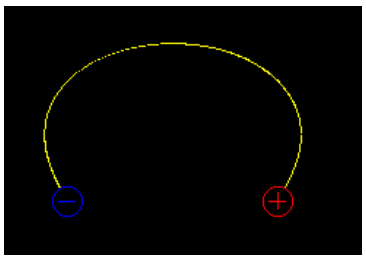
Une fois le sens en direction de la charge négative tracée, il faut tracer dans la direction de la charge positive.

CHARGE (+) --------°--------->CHARGE (-)



Donc nous revenons à l'étape 3.

CHARGE (+) --------°--------->CHARGE (-)



Vérification de proximités x,y

Lorsque l'on laisse apparaître des points aléatoires, il y a toujours un risque que les points aient dans une zone interdite. Donc un travail de post calcule a été effectué. Par exemple, si nous souhaitons vouloir voir le potentiel des charges (positif ou négatif), il ne fallait pas tirer de trait à l'intérieur de charges donc une constante dans le main.c permet de contrôler la règle de proximité qui a été fixée à 10, car j'ai décidé que 10 pixels était la taille d'une charge dans ma simulation.

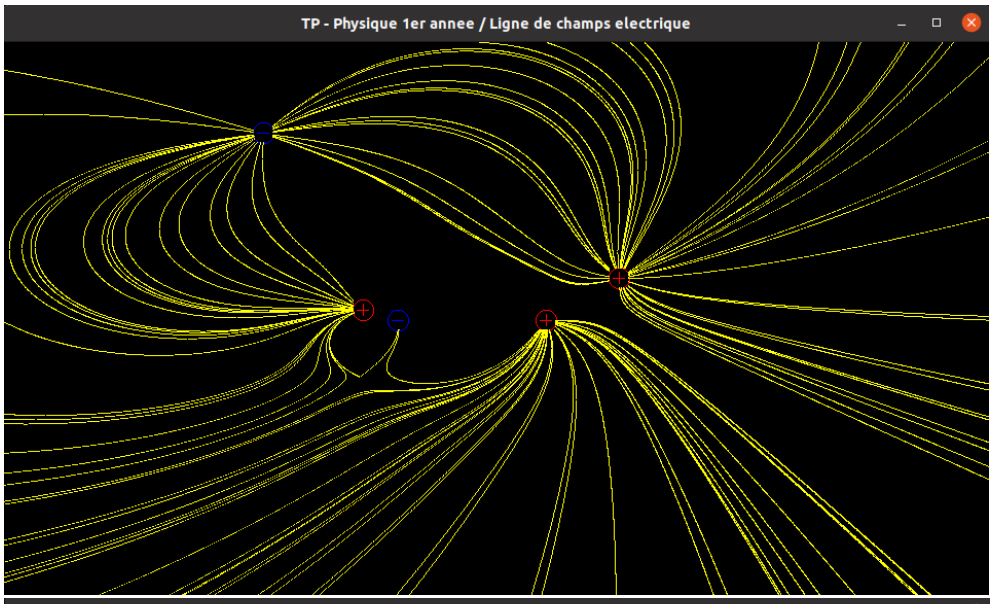
Ensuite dans l'étape 4 de l'énoncé, il était demandé de ne pas tracer des traits trop proches. Que le traçage devait être supérieur à un seuil choisi et que les points tracés devaient toujours être dans notre univers.

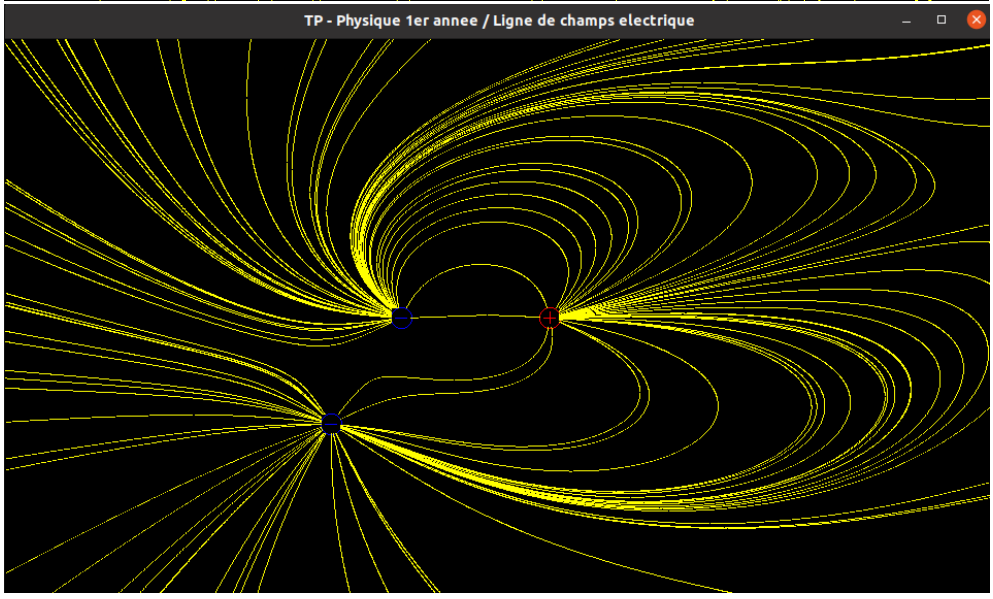
Première zone de contrôle : p est dans notre univers de [0,1] par [0,1] ?
Deuxième zone de contrôle : pSuivant est dans notre univers de [0,1] par [0,1] ?
Troisième zone de contrôle : p est à une distance acceptable de pSuivant pour tracer ?

- Distance acceptable définit par une constante dans particule.h

Quatrième zone de contrôle (je vous promets c'est la dernière xD) : p n'est pas trop proche de la charge ?

Images aléatoires des réactions des lignes de champs électriques :





Conclusion :

Dans l’ensemble, j’ai trouvé très formateur de réaliser ce travail pratique en C. Le fait de pouvoir allié physique et rendu visuel m’aide beaucoup pour la compréhension. Grâce à cela, on peut savoir comment réagissent les différentes charges, à différentes distances avec différents paramètres.

Comme le premier TP sur le troue noir, l’idée de ce travail est très sympa.

Cependant, il est bien de parler aussi de chose qui peut aider à améliorer ce travail pratique. Le TP est pour ma part trop axé programmation, je trouve que l’on perd en réflexion physique, la partie que j’apprécie le plus. Il serait peut-être plus sympa d’avoir un cadre comme vous l’avez écrit dans l’énoncé, mais avec plusieurs fonctions déjà réaliser pour gagner du temps et permettre de se focaliser sur la partie physique de ce travail.

Dans tous les cas, réussir à sortir une simulation des lignes de champs électriques et pouvoir la modeler comme on le souhaite a été très gratifiant.