Visualisation de Graphe en 2D

Torko Hervé , Abak-kali Nizar

17/01/2014

Contents

Chapter 1

Introduction

1.1 Objectif du projet

On souhaite visualiser Un graphe en 2D

- Pour cela, on doit placer les noeuds de maniere optimal selon un certain nombre de critère:
  [-] éviter les croisements d'arcs
  [-] les noeuds reliés entre eux doivent êtres proches
- On pourra utiliser un modèle physique :
  [-] elastique : les arcs se comporte comme des elastiques
  [-] gravitation les noeuds ont des champs de gravitation qui attire ou repousse les autres noeuds selon si il y a liason ou non
  [-] les noeuds non liés se repoussent
- Il faura determiner la qualités d"un graphe selon certain critère
- Une partie graphique permettra de visualiser l'evolution du graphe

1.2 Notre Projet

1.2.1 Presentation du Projet

Outils Utilise

Langage utilisé Lors de notre projet nous avons programmé en C++ ,pour deux raisons :

-> Le langage est plus facile de prise en main et contient beaucoup plus outils que le C .Deplus, le C++ contient beaucoup de structure prefaite très utile (ex: list , vector , map,etc..)

-> Pour ce projet nous allons utiliser un outils qui nous a été très utile pour travailler et manipuler des graphes: LEMON [1]. Or LEMON est present uniquement en C++ .

Bibliothèques SDL Pour repondre au besoins d'une partie graphique nous avons opté pour SDL . Il est simple d'utilisation , nous savons l'utiliser, et on disposait deja d'une petite bibliothèque de fonctions graphique.

[1] Library for Efficient Modeling and Optimization in Networks

Chapter 2

Explication du Code

2.1 auxi.cpp

Il s'agit du fichier qui contient les fonctions de dessin tel que circle pour des cercles , line pour dessiner des ligne, arrow pour des lignes flechées. Ces fonctions ont été trouvées sur le site de Mr Audibert .

2.1.1 Structure utilisé

Aucune structure particulière dans ce fichier n'est utilisé.

2.1.2 Fonctions utilisé

putpixel()

```
 void putpixel(SDL_Surface *ecran,int xe, int ye, Uint32 c) {
  Uint32 * numerocase;
  numerocase = (Uint32*)(ecran->pixels)+xe+ye*ecran->w;
  *numerocase = c;
}
```

Il s'agit de la base des fonctions de dessin. Elle permet de poser une couleur à une position donné.

getpixel()

```
Uint32 getpixel(SDL_Surface* ecran, int xe , int ye){
  Uint32 * numerocase;
  numerocase= (Uint32 *)(ecran->pixels)+xe+ye*ecran->w;
  return (*numerocase);

}
```

La encore il s'agit d'une fonction de base qui consiste a recuperer la couleur d'un pixel dont on donne la position et à la retourné.

line()

```
void line( SDL_Surface *ecran,int x0,int y0, int x1,int y1, Uint32 c) {
  int dx,dy,x,y,residu,absdx,absdy,stepx,stepy,i;
```

```
  dx=x1-x0; dy=y1-y0; residu=0;
  x=x0;y=y0; putpixel(ecran,x,y,c);
  if (dx>0)
    stepx=1;
  else stepx=-1;
  if (dy>0)
    stepy=1;
  else stepy=-1;
  absdx=abs(dx);
  absdy=abs(dy);
  if (dx==0)
    for(i=0;i<absdy;i++) {
      y+=stepy;
      putpixel(ecran,x,y,c);
    }
  else if(dy==0)
    for(i=0;i<absdx;i++){
      x+=stepx;
      putpixel(ecran,x,y,c);
    }
  else if (absdx==absdy)
    for(i=0;i<absdx;i++) {
      x+=stepx; y+=stepy;
      putpixel(ecran,x,y,c);
    }
  else if (absdx>absdy)
    for(i=0;i<absdx;i++)
      { x+=stepx;
residu+=absdy;
if(residu >= absdx)
  {residu -=absdx; y+=stepy;}
putpixel(ecran,x,y,c);
      }
  else for(i=0;i<absdy;i++)
 {
   y+=stepy; residu +=absdx;
   if (residu>=absdy)
     {residu -= absdy;x +=stepx;}
   putpixel(ecran,x,y,c);
 }
}
```

il s'agit de la fonction qui permet de dessiner une ligne entre deux couples de coordonnées .

arrow

```
void arrow(SDL_Surface *screen,int x1, int y1, int x2, int y2, Uint32 c)
{
  int dx,dy;
  float xf1,yf1,xf2,yf2,d,dx1,dy1,ndx1,ndy1,ndx2,ndy2,angle=M_PI/6.;
  line(screen,x1,y1,x2,y2,c);
  dx=x2-x1; dy=y2-y1; d=sqrt(dx*dx+dy*dy);
  if (d!=0.)
    { dx1=6.*(float)dx/d; dy1=6.*(float)dy/d;
      ndx1=dx1*cos(angle)-dy1*sin(angle);
      ndy1=dx1*sin(angle)+dy1*cos(angle);
      xf1=0.3*x1+0.7*x2; yf1=0.3*y1+0.7*y2; xf2=xf1-ndx1; yf2=yf1-ndy1;
      line(screen,xf1,yf1,xf2,yf2,c);
      ndx2=dx1*cos(-angle)-dy1*sin(-angle);
      ndy2=dx1*sin(-angle)+dy1*cos(-angle);
      xf2=xf1-ndx2; yf2=yf1-ndy2; line(screen,xf1,yf1,xf2,yf2,c);
    }
  else
    {cercle(screen,x1+10,y1,10,c); line(screen,x1+20,y1,x1+23,y1-6,c);
      line(screen,x1+20,y1,x1+15,y1-5,c);
    }
}
```

Cette fonction dessine un vecteur d'un couple de coordonnées A{xa,ya} B{xb,yb} A->B .

2.2 phisic.cpp

Ce fichier contient toutes les fonctions physique du programme tel que le rapprochement ou l'éloignement de noeuds .

2.2.1 Structure utilisé

droite$_t$ Cette structure est censé représenter une droite . Or une droite a pour formule $y = (a * x) + b$. Donc ,notre structure contient deux doubles , un double qui represente le coefficient directeur et un autre qui correspond a l'intersection avec l'axe des ordonnées.

2.2.2 Fonctions

```
    struct droite_t{
   double coeff;
  double inter;
  };
typedef struct droite_t drt;
```

Cette structure sert entre autre lors des rapprochements ou des eloignements pour pouvoir garder les deux noeuds sur un même axe .

2.2.3 Explication des fonctions

Les fonctions de calcul de distance

- distancepointcentre() : distance entre les coordonnées d'un noeud et le centre de la fenêtre
- distance2points(): distance entre deux noeuds

La fonction d'initialisation de la droite

```
drt initdroite(ListDigraph::Node noeud1, ListDigraph::Node noeud2, ListDigraph::NodeMap<int>& xe ,ListDigraph::NodeMap<int>& ye){
  drt f;
  //initialisation du coefficient directeur
  f.coeff = (ye[noeud1]-ye[noeud2])/(xe[noeud1]-xe[noeud2]);

//cacul intersection avec l'axe des ordonnees
   f.inter=-(f.coeff*xe[noeud1])+ye[noeud1];
  return f;
}
```

cette fonction prends comme argument deux noeuds ainsi que les deux nodemaps de noeuds contenant les coordonnées de tout les noeuds . cette fonction initialise la droite a partir des coordonnées des deux noeuds.

2.3 visu.cpp

2.3.1 Structure utilisé

2.3.2 Fonctions

Chapter 3

Fonctionnement

3.1 Mais comment que sa marche

3.2 Quelques screenshots

Chapter 4

Conclusion

4.1 Ce qui a été fait

4.2 Probleme rencontré

4.3 Amelioration possible