Deux projets au choix

Lundi 23 Octobre 2017

Michael FRANÇOIS

francois@esiea.fr

https://francois.esiea.fr/

LABYRINTHE 2D

Présentation générale

• L'objectif de ce projet est de modéliser en C, un labyrinthe en 2D. Ensuite faire une recherche de chemin quelconque entre l'entrée et la sortie.

• Le labyrinthe est considéré comme un tableau bidimensionnel d'entiers, de taille donnée.

• On doit pouvoir charger le labyrinthe depuis un fichier texte, qui contient toutes les caractéristiques nécessaires.

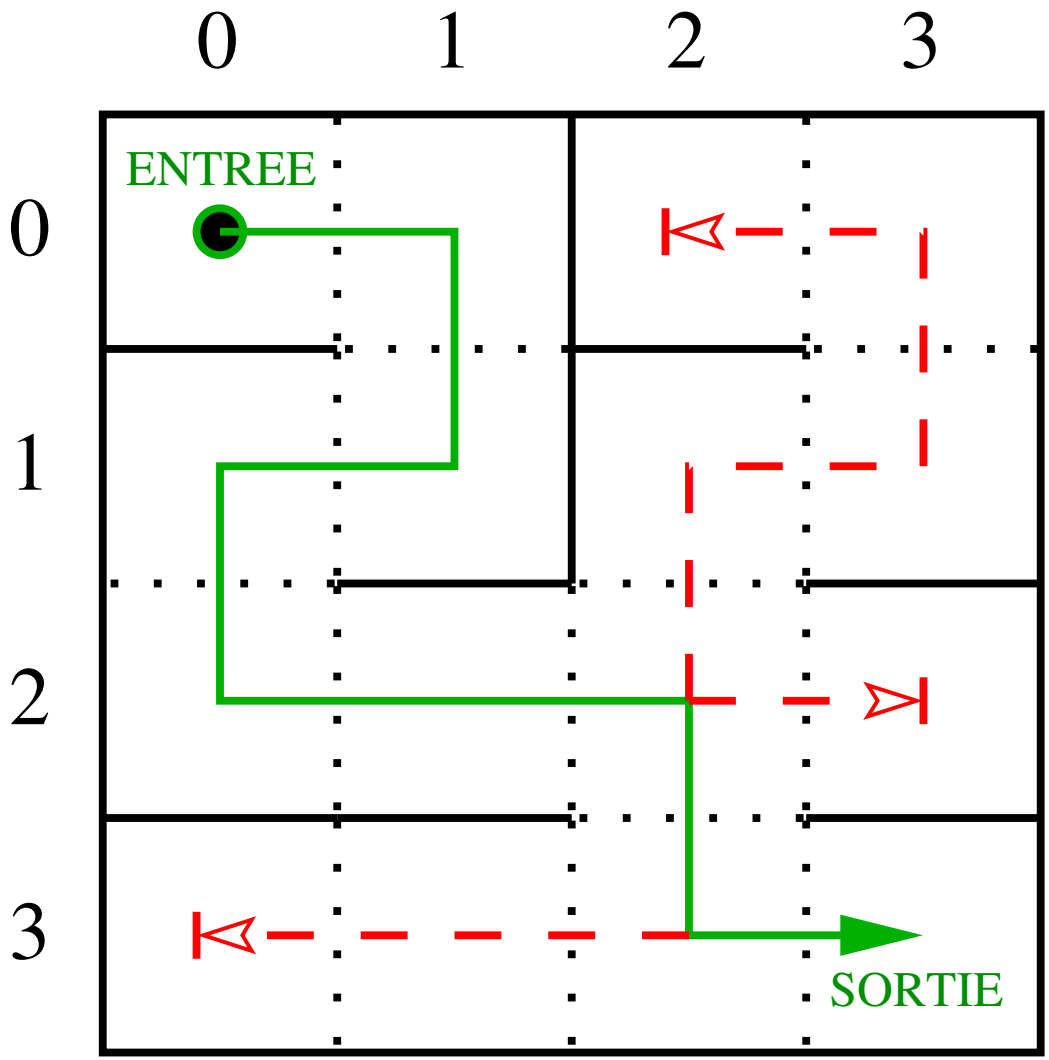
• On doit aussi pouvoir générer aléatoirement un labyrinthe avec toutes ses caractéristiques de base.

Création du labyritnhe

Le labyrinthe sera représenté par une structure, contenant les éléments suivants :

- le tableau 2D d'entiers
- le nombre de lignes et de colonnes du labyrinthe,
- la position en x et y de l'entrée/sortie du labyrinthe,
- ainsi que la position en x et y du chercheur de chemin.

Exemple de labyrinthe 4×4, avec une entrée (resp. sortie) en (0,0) (resp. (3,3)) :



Représentation et configuration d'une cellule

Une cellule du tableau peut être représentée par un entier court de type `unsigned short`
(*i.e.* sur 16 bits : $b_{15}b_{14}b_{13}b_{12}b_{11}b_{10}b_9b_8b_7b_6b_5b_4b_3b_2b_1b_0$) :

- Les quatre bits de poids faible ($b_3b_2b_1b_0$) permettront de stocker la configuration initiale de la cellule, c'est-à-dire les 4 murs autour de la cellule.
- Les quatre bits de poids suivants permettront de stocker l'évolution de la configuration de la cellule lors de la recherche d'un chemin.
- Les huit bits restants peuvent également servir à enregistrer des infos, lors de la recherche d'un chemin quelconque.

• La configuration d'une cellule est représentée sur les quatre bits de poids faible ($b_3 b_2 b_1 b_0$), de la façon suivante :
Le bit b_3 sera égal à 1 si la cellule contient un mur infranchissable vers le haut. Les bits b_2 , b_1 , et b_0 indiqueront la présence d'un mur à droite, en bas, et à gauche respectivement. Par exemple, la cellule $(0, 0)$ du labyrinthe précédent, sera configurée initialement à l'aide de 1011, alors que la cellule $(2, 3)$ sera représentée par 1110.

• Pendant la génération du labyrinthe, il faudra faire attention que les cellules soient cohérentes entre elles. Par exemple, lorsqu'une cellule est initialisée sans mur vers le bas, il faudra s'assurer que celle en dessous dans le labyrinthe n'ait pas de mur vers le haut.

Exemple de fichier texte contenant une configuration de labyrinthe de taille 10×10 ayant comme entrée la position $(0, 0)$ et en sortie la position $(9, 9)$:

```
10 10 0 0 9 9
9 12 13 11 8 8 12 13 9 12
3 2 2 8 0 2 2 0 6 7
9 14 9 0 0 8 10 4 11 14
1 12 3 6 1 4 9 0 14 13
1 4 9 8 0 2 0 4 15 5
5 1 0 4 1 10 2 4 9 6
1 4 5 1 0 14 13 5 5 13
7 5 5 1 6 15 5 1 0 6
9 0 2 4 9 12 7 1 4 13
7 3 10 2 6 7 15 7 3 6
```

Recherche d'un chemin

La recherche d'un chemin quelconque, sera accessible à travers le menu principal de votre programme. Un parcours de l'espace de recherche en profondeur permettra de trouver un chemin possible dans le labyrinthe.

Le projet est à réaliser seul sous environnement GNU/Linux, et doit être envoyé au plus tard le 05/11/2017 à 23h59 (à confirmer), sous la forme d'une archive `.tar.gz` à vos noms et prénoms (*i.e.* `NOM_prenom.tar.gz`), contenant tous les fichiers sources du projet.

Lors de la soutenance (en anglais), l'évaluation sera globalement scindée en 3 parties :

- présentation orale ;
- démonstration sur un fichier contenant une configuration de labyrinthe. Le fichier sera donné le jour de la soutenance ;
- phase de questions/réponses.

Démonstration

SIMULATEUR DE CIRCULATION URBAINE

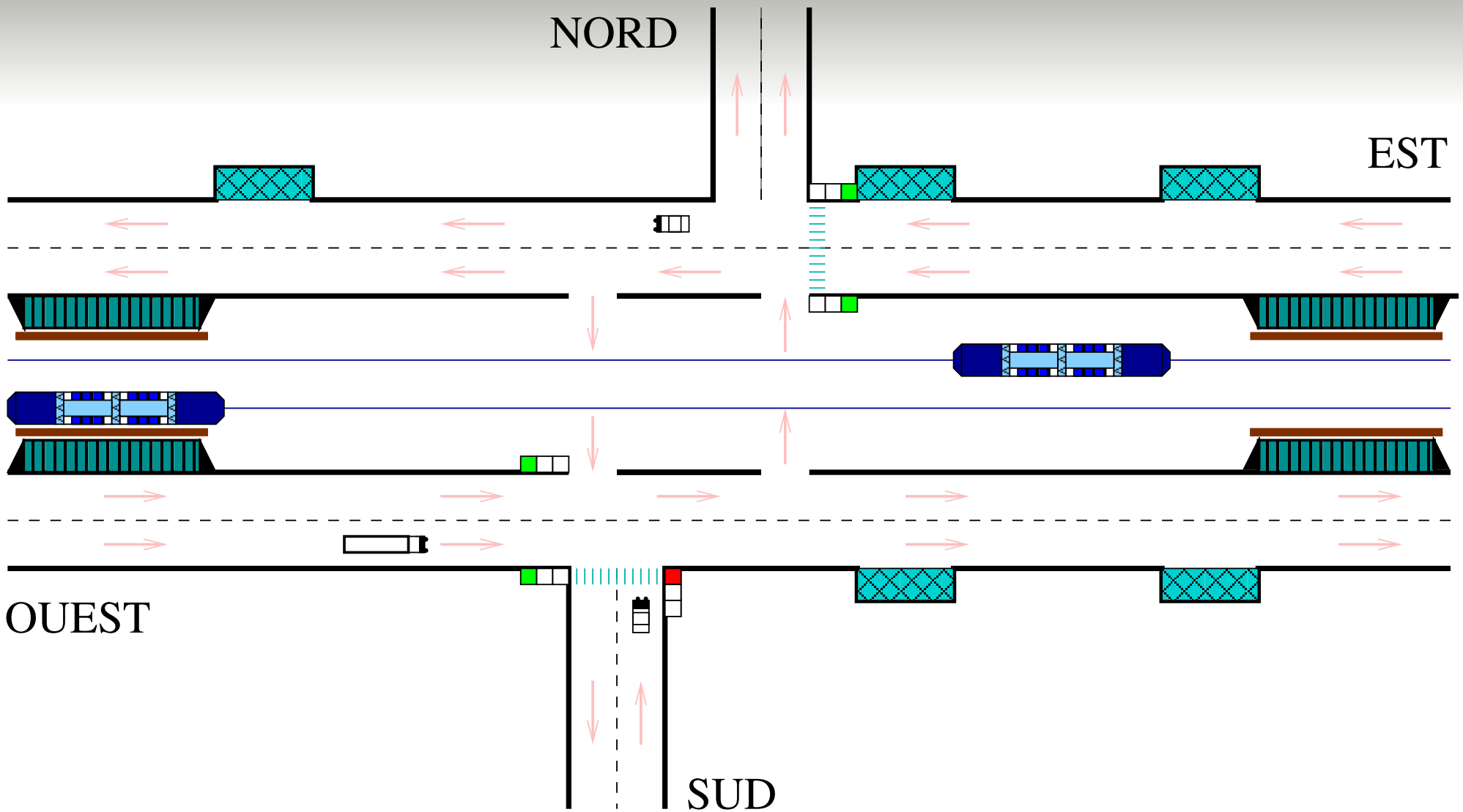
Présentation générale

• L'objectif de ce projet est de programmer en C, un simulateur de trafic automobile sur un plan de circulation donné.

• La simulation doit se faire comme dans la vraie vie (véhicules lents, rapides, dangereux, accidents, pannes, etc.).

• L'affichage doit se faire uniquement sur console (pas besoin d'utiliser des librairies avec un graphisme évolué comme par exemple la SDL).

Plan de circulation à utiliser



Exemple de structure d'une voiture

```
\begin{verbatim}
-----------------------------------------------------------
typedef struct voiture VOITURE;
struct voiture
{
  char direction ;  /*N => Nord, S => Sud, E => EST, O => OUEST*/
  int posx;         /*Position courante x de la voiture*/
  int posy;         /*Position courante y de la voiture*/
  int vitesse;      /*Vitesse du véhicule*/
  char alignement;  /*'g'=>gauche ou 'd'=>droite*/
  char type;        /*'v'=>voiture, 'c'=>camion, etc.*/
  char custom[30];  /*Contient le véhicule customisé*/
  char ravitaillement ; /*1=>ravitaillement 2=>carburant et 0=>sinon*/
  char etat;        /*État du véhicule => actif ou inactif*/
  struct voiture * NXT; /*Pointeur vers une prochaine voiture,
                          necessaire pour la liste chainée*/
  /*Vous pouvez rajouter d'autres variables si nécessaire */
};
-----------------------------------------------------------
```

Liens pratiques

- Pour avoir un affichage plus joli, les caractères de l'ASCII étendu peuvent être utilisés :
`http://www.theasciicode.com.ar/extended-ascii-code/copyright-symbol-ascii-code-184.html`

- Émoticônes pour un meilleur rendu de votre jeu à ce lien :
`https://fr.piliapp.com/twitter-symbols/`

Il suffit de cliquer sur le symbole souhaité, le copier puis le coller simplement dans votre programme. Dans le cas où le symbole ne s'affiche pas correctement sur le terminal, vous devez installer le package "ttf-ancient-fonts" via la commande :

```
sudo apt-get install ttf-ancient-fonts
```

Évaluation

Lors de la soutenance (en anglais), l'évaluation sera globalement scindée en 3 parties :

- présentation orale ;
- démonstration du jeu en mode fluide (si possible panne et accident en plus) ;
- phase de questions/réponses.

Dates importantes

- Date de soutenance : Lundi 06 Novembre à partir de 9h00.
- Code source à rendre par email (`francois@esiea.fr`) en format `.tar.gz` ou `.zip` le dimanche 5 Novembre au plus tard à 23h59.

Démonstration