PROJET DE PROGRAMMATION : SIMULATION DE FOULE

CE PROJET EST A RENDRE POUR .....

Vous devez le réaliser en binôme. Il sera évalué selon les critères suivants:

- *Qualité de votre code (commentaires, indentation)*
- *Qualité de votre algorithme (score, explications écrites et orales, efficacité)*
- *Répartition du travail dans le binôme*
- *Capacité à expliquer votre programme lors de la soutenance*
- *Améliorations diverses*

L'objet de ce projet est de modéliser les trajectoires de personnes dans un plan comportant des obstacles, entre un point de départ et un point d'arrivée donnés.

Les personnes sont modélisées par des carrés se déplaçant dans un plan de 80 cases sur 60. La représentation graphique de cette simulation sera réalisée dans une fenêtre de 800x600 où chaque case aura donc une dimension 10x10.

Le programme prendra en entrée un plan, fourni (voir `data/plan`) sous la forme d'un fichier texte selon le format suivant:

- sur la 1ère ligne, deux entiers indiquant les dimensions du plan (80 60 pour le moment);
- sur les 80 lignes suivantes, 60 caractères '_' ou '*', le premier figurant un espace vide, le second un mur.

Nous vous fournissons des fichiers `fxfoule.c/.h` contenant une fonction

```
void charge_plan(char *nom_fichier, int plan[80][60])
```

qui prend en paramètre un fichier et un tableau 2D d'entiers représentant le plan, et qui remplit le second avec le premier. Les '_' et '*' du fichier `plan` sont codés respectivement 0 et 1 dans le tableau d'entiers. Modifiez à votre guise les types et les codes proposés.

L'écriture de l'algorithme de parcours du joueur entre son point de départ et son point d'arrivée constitue le coeur du projet. Vous devez en effet inventer par vous-même l'algorithme de votre choix, qui soit efficace dans des configurations de plan simple (sans obstacles trop complexes). Pour ce faire, vous pourrez par exemple user de méthodes:

- déterministes: où chaque mouvement est précisément déterminé par la configuration rencontrée;
- partiellement aléatoires: utilisant des tirages aléatoires dans certaines situations de blocage (voir les fonctions `rand()` et `srand()`);
- ...ou tout autre méthode pertinente de votre choix.

OBJECTIF DE BASE

Vous devez réaliser un programme qui charge un plan, l'affiche, ainsi qu'un joueur positionné à un endroit de votre choix, et sa destination. Puis votre programme doit faire parcourir la trajectoire qui sépare le joueur de sa position d'arrivée. **Les personnes ont uniquement connaissance de leur destination et des cases qui les entourent, pas de connaissance préalable des obstacles qui ne seront découverts que lors de leur rencontre (donc pas d'algorithme de graphes complexes copiés sur le net et que vous ne sauriez expliquer lors de la soutenance).**

Il est très fortement conseillé d'utiliser une structure adaptée pour stocker les coordonnées de la personne, qui contiendrait en l'occurrence deux champs x et y.

La qualité de votre algorithme sera mesurée par un score final indiquant le nombre de déplacements effectués par le joueur pour atteindre son but.

OBJECTIF FOULE

Vous devez désormais gérer N personnes, et charger leurs positions de départ et d'arrivée depuis un fichier texte (voir le fichier `data/joueurs` fourni) utilisant le format:

- 1ère ligne: nombre N de personnes;
- sur les N lignes suivantes, 4 entiers indiquant les coordonnées de départ et d'arrivée de chacun des N joueurs.

Une fois ces N personnes chargées dans le plan, votre algorithme doit les déplacer à tour de rôle, sans qu'elles ne se chevauchent, selon les règles déterminées par votre algorithme.

Le seul moment où des personnes peuvent se chevaucher est le démarrage du programme. On pourra donc positionner plusieurs joueurs sur la même case au commencement.

OBJECTIF INTERFACE

Vous pourrez améliorer l'interface de votre programme à votre guise. Quelques propositions:

- donner des couleurs différentes aux personnes partant de points de départ différents;
- permettre de dessiner et sauvegarder un plan à l'aide de la souris;
- modifier la vitesse de déplacement en temps réel;
- proposer une interface graphique de sélection de plans et de positionnement des joueurs;
- les joueurs laissent des traces derrière eux, dont la couleur s'intensifie selon le nombre de passages ;
- ...
- Toute amélioration pertinente est la bienvenue.

OBJECTIF STRATEGIE

Votre algorithme doit maintenant être amélioré de façon à pouvoir gérer les situations de blocage, et à s'en extraire. Vous pourrez par exemple programmer plusieurs stratégies différentes, utilisées par certaines personnes ou bien à tour de rôle selon la situation.

Exemples:

- En cas de blocage avéré, un joueur pourra changer de stratégie afin de s'extraire;
- ...ou bien décider de ne pas bouger pour attendre (dans le cas de blocage par d'autres personnes).
- Connaissance de l'environnement proche (par exemple à 2 ou 3 cases de distance) afin de motiver les décisions.
- Pré-connaissance du terrain complet et utilisation d'algorithmes de graphes complexes (que vous saurez expliquer au moins dans les grandes lignes).