Vision d'ensemble du projet : il s'agit de modéliser de manière simple le déplacement d'une foule. Nous sommes dans le cadre d'un cours programmation concurrente et donc ce qui est important ce n'est ni l'IHM, ni l'évolutivité du code mais bien l'identification des contraintes de synchronisation et y apporter une réponse adéquate.
Vous pouvez programmer ce projet dans les langages suivants : Python, GO, C, C++ mais pas Java. Si vous choisissez C/C++ alors vous devrez utiliser la bibliothèque Posix pour gérer les threads et la synchronisation.

Spécification de la mise en œuvre
Le terrain sur lequel se déplace la foule fait 512 pixels x 128 pixels et sera représenté par une matrice de taille identique ayant 512 colonnes (numérotées de 0 à 511) et de 128 lignes (numérotée de 0 à 127). Des obstacles de taille quelconque seront présent sur le terrain. En aucun cas ces obstacles touchent le bord du terrain. La sortie sera le coin supérieur gauche et fera 2 pixels * 2 pixels. La figure 1 ci-dessous représente le terrain.

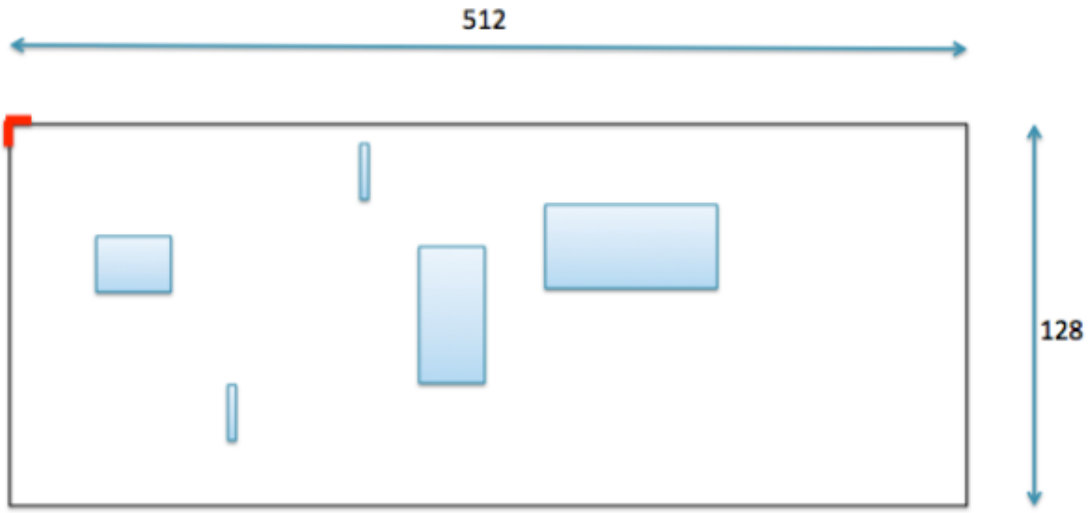

Fig. 1 – forme du terrain

Une personne fait 1*1 pixel et se déplace d'un seul pixel selon les directions Nord, Sud, Est et Ouest ainsi que les 4 diagonale Nord-Sud, Nord-Ouest, Sud-Est et Sud-Ouest.
Après une phase d'initialisation qui met en place le terrain et qui distribue aléatoirement 2^P personnes, le simulateur fait avancer chacune des personnes vers la sortie. A chaque déplacement la personne doit se rapprocher de la sortie. Pour simplifier le déplacement des personnes, un point azimuth sera fixé (par exemple l'angle en haut à gauche) et le déplacement choisi sera le déplacement **possible** qui minimise la distance avec le point (cf. figure 2). Bien évidemment une personne ne peut pas occuper une place occupée.

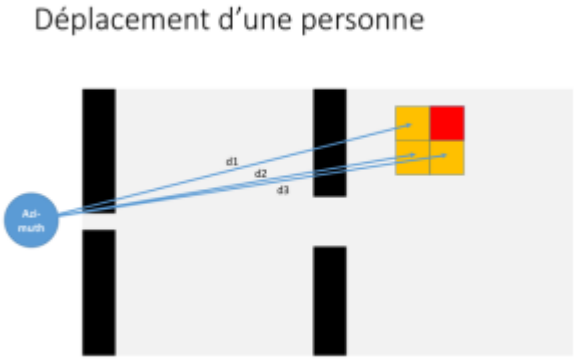

Fig. 2 – point azimuth

Afin de comparer les différentes mises en œuvre, il est souhaitable qu'à chaque exécution, les personnes partent de la même place.
Mises-en œuvre : vous allez programmer un simulateur ultra-simpliste du déplacement d'une foule. Toutes les versions seront intégrées dans le même code et des options du programme permettront de choisir la version à exécuter :

-p [0123456789] : nombre de personnes présentent sur le terrain
∗ p varie de 0 à 9 et le nombre de personnes vaut 2^p (i.e. varie de 1 à 512)

-t [01] : scénario de créations des threads
∗ -t0 : un thread est associé à chacune des personnes créées et chaque thread doit faire avancer la personne qu'elle gère ;
∗ -t1 : le terrain est partagé en 4 partie égale. Un thread est associé à chaque partie du terrain et chaque thread doit faire avancer successivement chacune des personnes présentes sur la portion de terrain que le thread gère.

-m : mesure du temps d'exécution
∗ mesure la consommation du CPU et le temps de réponse du programme
∗ lorsque des mesures sont effectuées : la phase d'initialisation du programme n'est pas prise en compte et l'affichage n'est pas actif
∗ pour effectuer les mesures, l'application est lancée 5 fois et la mesure est la moyenne des 3 valeurs intermédiaires

Pour simplifier la portabilité des programmes entre votre machine et la mienne, il doit être possible de compiler votre programme sans inclure la partie graphique qui doit visualiser le déplacement de la foule sur le terrain.