Unité 6 : Coordonnées — **Application : Marche aléatoire**

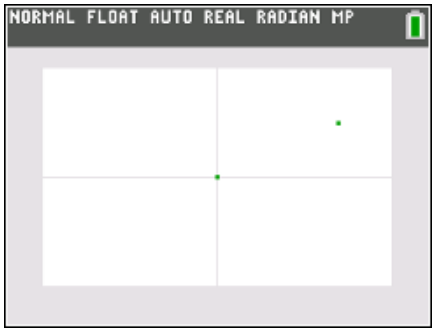
Vue d'ensemble : Les explorations avec des nombres aléatoires peuvent conduire à des observations fascinantes. Cette application vous donne l'occasion d'explorer les probabilités et de programmer ainsi le déplacement du Rover sur une grille.

Objectifs :

- Utiliser les coordonnées pour simuler une « marche aléatoire »
- Utiliser les compteurs dans un programme
- Utiliser les instructions conditionnelles **non(** ; **et**

Une « *marche aléatoire* » est une expérience programmée sur un ordinateur. Cette activité relie différentes compétences en programmation.

Le Problème :



Supposons que les rues de votre ville soient disposées sur un quadrillage, et que votre école se trouve au point de coordonnées (0,0) de votre repère. Votre maison est située au point de coordonnées (7,3), ce qui représente 7 blocs à l'Est de l'école et 3 blocs au Nord. On a représenté sur le graphique ci-contre une image de la situation.

En partant de l'école, vous marchez de façon aléatoire dans une direction (Nord, Sud, Est ou Ouest) en parcourant à chaque fois un bloc de bâtiment, A chaque intersection, vous avancez d'un bloc à chaque fois. Pourrez-vous rentrer ainsi chez vous ? Combien de blocs seront nécessaires ?

La planification est une partie importante du codage. Pensez à ce que le Rover peut faire et à ce que votre langage de programmation peut également faire. Conservez à l'esprit que lorsque vous travaillez avec des nombres aléatoires, vous êtes à la merci de la machine. Cela peut prendre beaucoup de temps avant que le Rover ne rentre « à la maison ». Nous préciserons donc dans le programme un nombre limité de blocs.

Conseil à l'enseignant : Préparer les étudiants afin de permettre au programme de saisir l'emplacement de la maison **Input** aux coordonnées (7,3). Utiliser des variables pour cette position au début du programme. En fait il est préférable d'utiliser des variables aussi souvent que possible (plutôt que des nombres). L'utilisation de variables permet facilement d'ajuster un programme. Pour initialiser un programme, on préfère l'instruction **Input** au lieu de la commande **Sto**.

1. Commencer le programme avec l'instruction CONNECT RV. Fixer la taille de la grille du Rover à 5 cm en utilisant **Send("SET RV.GRID.M/UNIT .05")**.

   Souvenez-vous que cette instruction se trouve dans le menu **prgm > Hub > Rover (RV)… > RV Setup…**et change l'unité de mouvement du Rover (en utilisant **FORWARD 1**) depuis 10 cm jusqu'à 5 cm, en autorisant plus de points sur la grille dans un espace plus petit.

```
NORMAL FIXE2 AUTO a+bi RAD MP
ÉDIT MENU: [alpha] [f5]
PROGRAM:ROVER6A
:Send("CONNECT RV")
:Send("SET RV.GRID.M/UNIT
0.05")
:Pause "ENTRER POUR COMMEN
CER"
:7→A
:3→B
:0→W
:10→E
```

Initialiser les variables

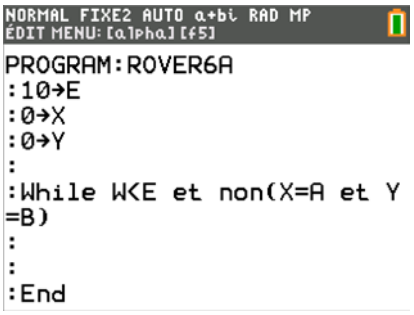
2. Sauvegarder les coordonnées de la maison (7,3) et le nombre de blocs parcourus (0) respectivement dans les variables ***A***, ***B*** et ***W***. La variable ***W*** sera utilisée pour suivre le nombre de blocs parcourus par le Rover et également pour le stopper s'il va trop loin. Le programme s'arrête au bout d'un certain nombre de blocs parcourus.

   Utilisons une valeur de 10 blocs avant que ne cesse le mouvement et

stockons cette valeur dans la variable **E**.

3. Le Rover arrive à la maison lorsque sa position que nous appelons (X,Y) est (7,3). Les variables X et Y sont initialisées à 0.

La boucle principale



4. La boucle principale du programme consiste à réaliser une boucle **While** en adressant deux conditions. Les conditions correspondent au retour du Rover « à la maison » (lorsque **X=A** et **Y=B**) ou en quittant de manière exhaustive (lorsque **W** = **E**). Le Rover stoppe son mouvement lorsque la variable **W** (le nombre de blocs marchés) est égal à la valeur sauvegardée dans la variable **E**.

   La boucle **While** continue tant que ces deux conditions sont fausses, Nous avons donc mis en place les conditions opposées.

   La boucle **While** sera :
   While W<E et non(X=A and Y=B)

   Souvenez-vous que les valeurs relatives au problème sont stockées dans **E**, **A**, et **B**.

   non( ) est utilisé afin de s'assurer que le programme continue tant que le Rover n'est pas rentré « à la maison ». En Logique, **et non(** est « l'opposé » de **ou**.

5. N'oubliez pas d'inclure une instruction **End** à la fin de la boucle.

A l'intérieur de la boucle

6. Incrémenter **W** (le nombre de blocs marchés) : **W+1→W**.

7. Choisir une direction aléatoire (nord, sud, est, ou ouest) :
   - La commande **TO ANGLE** du Rover tourne celui-ci dans une direction absolue. La direction : 0 est l'Est, 90 est le Nord, 180 est l'Ouest, et 270 est le Sud
   - **nbrAléatEnt(0,3)** donne un entier aléatoire qui peut être 0, 1, 2, and 3
   - Multiplier ces valeurs par 90 pour avoir 0, 90, 180, ou 270
   - L'instruction pour avoir une direction aléatoire **D** est donc :

     90nbrAléatEnt(0,3)→D

8. Faire tourner le Rover d'un angle **D**: **Send("RV TO ANGLE eval(D)")**.

9. Faire avancer le Rover de 1 unité (1 bloc dans notre simulation) : **Send("RV FORWARD 1")**.

10. Adapter la position du Rover dans le programme :
    - Si le Rover va au Nord, alors incrémenter **Y** de 1
    - Si le Rover va à l'Est, alors incrémenter **X** de 1
    - Si le Rover va au Sud, alors incrémenter **Y** de 1
    - Si le Rover va à l'Ouest, alors incrémenter **X** de 1

Inclure quelques instructions **Wait** afin de conserver le déroulement du programme synchronisé avec les mouvements du Rover. Tourner prend du temps et avancer également. Le délai **Wait** dépend donc de l'angle de rotation effectué et de la distance parcourue. Aussi vous devez réaliser quelques expérimentations afin de fixer précisément les délais à fixer à l'instruction.

Après la boucle

La fin de la boucle s'effectue selon deux possibilités :

- Si **W**=**E**, alors le Rover quitte la marche de manière impromptue (objectif non atteint). Joue un son de « tristesse », affiche une couleur rouge sur la DEL du Rover, et affiche « ROVER QUITTE » sur l'écran de la calculatrice.
- Si **X**=**A** et **Y**=**B**, le Rover est « rentré à la maison ». Joue un son de victoire, affiche une couleur verte sur le DEL, et affiche un message « ROVER EST RENTRE » sur l'écran de la calculatrice.

Pouvez-vous faire effectuer au Rover une danse joyeuse lorsqu'il a atteint son objectif ?

Conseil à l'enseignant :

Exemple de solution : Les étudiants peuvent tester leur code en l'absence du Rover. L'instruction **Send** ne causera aucun dommage. La première instruction **Disp** dans le code ci-dessous donne la position du Rover sur la grille même si le robot n'est pas disponible. Les temps d'attente (**Wait)** peuvent nécessiter des ajustements et peuvent être supprimés si on souhaite exécuter le programme plus rapidement en l'absence du Rover.

```
ClrHome
Disp "ROVER UNIT6 APP"
Send("CONNECT RV")
Send("SET RV.GRID.M/UNIT .05")
Pause "PRESS ENTER TO START"
7→A
3→B
0→W
10→E
0→X
0→Y
While W<E and not(X=A and Y=B)
W+1→W
90randInt(0,3)→D
Send("RV TO ANGLE eval(D)")
"Wait 1
Send("RV FORWARD 1")
"Wait .5
If D=0:X+1→X
If D=90:Y+1→Y
```

```
If D=180:X-1→X
If D=270:Y-1→Y
Disp {X,Y}
End
If W=E:Disp "ROVER QUIT"
If X=A and Y=B:Disp "ROVER MADE IT HOME"
```

Prolongements possibles :

- Allumer la diode de couleur COLOR LED d'une couleur différente lors de chaque changement de direction (et seulement lors d'un déplacement sans changement de direction). Le temps peut être critique.
- Ajouter un SON pour des effets spéciaux
- Ajouter un test pour voir si le Rover est de retour à l'école après avoir débuté son mouvement. Cela pourrait être une autre condition pour terminer le programme et qui pourrait être ajouté à l'instruction **While** (mais pas au début, sinon la boucle ne serait jamais démarrée).
- Ajouter du code pour suivre et afficher le nombre total de blocs parcourus par le Rover et afficher la distance parcourue (ou bien la distance à vol d'oiseau depuis le point de départ)
- Envisager de limiter le Rover à se déplacer uniquement vers l'Est ou le Nord. Comment la logique du programme est-elle affectée ? (Indice : voir la condition de la boucle **While**.)

Bonus 1 : A quelle distance de l'école à vol d'oiseau se trouve le Rover à la fin du programme ?

Bonus 2 : Quelle était la distance maximale depuis la maison, depuis l'école ?