**Guide d’utilisation de Krop**

Krop est un projet C# permettant d’aborder les notions de bases de la programmation en écrivant un programme simple dans un langage de programmation simplifié qui dicte le comportement d’une fourmi.

# Prérequis

Pour pouvoir utiliser Krop, il vous faut :

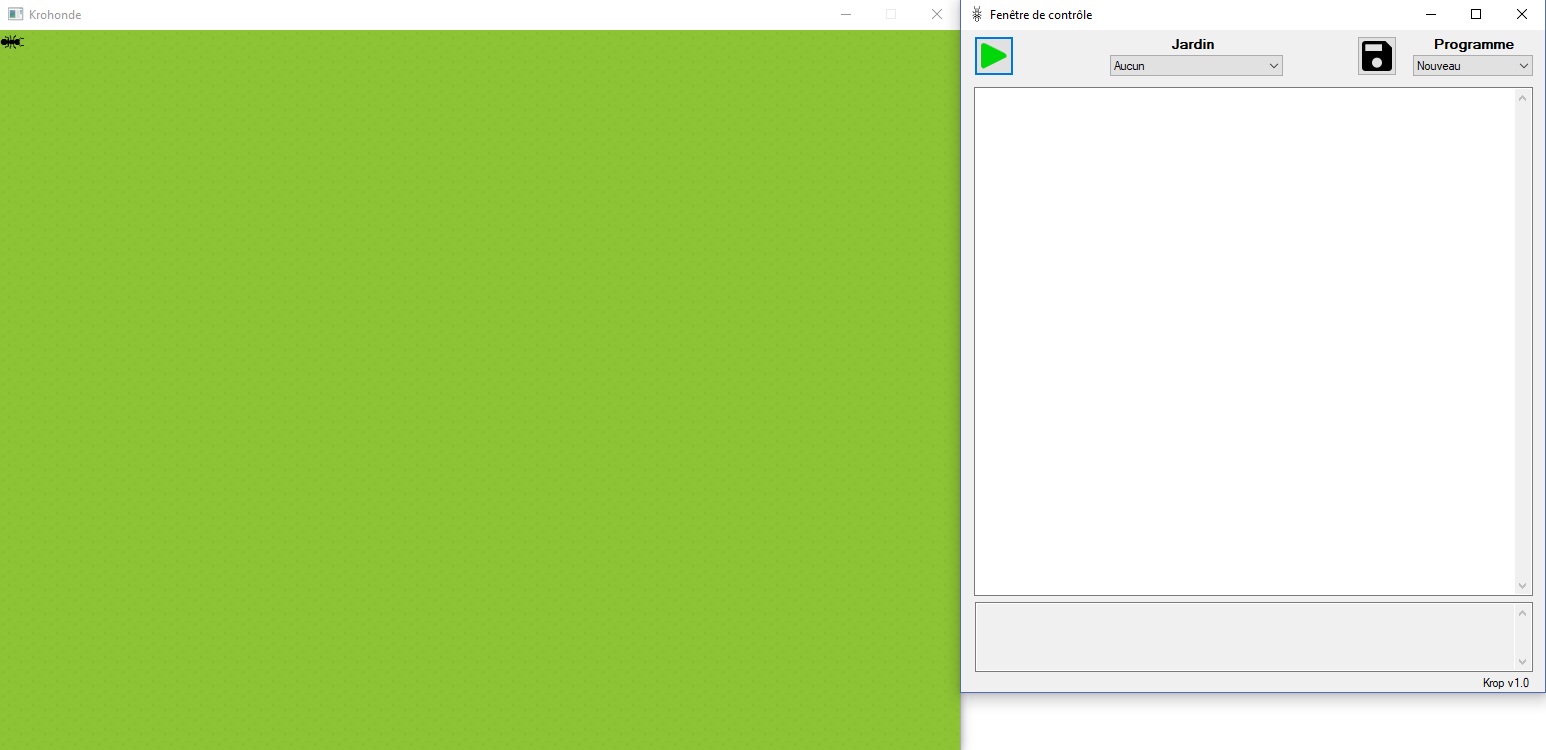
* Utiliser Windows 7 ou 10
* Faire une copie locale du dossier « Krop »

# Démarrage

Pour démarrer Krop, il faut procéder comme suit :

* Ouvrez le dossier « Krop » (la copie locale)
* Double-cliquez sur le fichier « Krop.exe »

Ensuite l’application Krop s’ouvre :



L’application est composée des deux fenêtres suivantes :

* La fenêtre Krohonde affiche les résultats de l’exécution du programme
* La fenêtre de contrôle permet de créer, modifier ou supprimer un programme. Elle permet aussi de démarrer, mettre en pause ou stopper un programme. Le terminal en bas de la fenêtre affiche les messages d’erreur et les paroles de la fourmi.

# La fenêtre de contrôle

|  |  |  |
| --- | --- | --- |
| Créer un nouveau programme | | |
| 1 | Sélectionnez « Nouveau » dans la liste des programmes |  |
| 2 | Écrivez votre code dans le champ texte |  |
| 3 | Cliquez sur le bouton « sauvegarder » |  |
| 4 | Écrivez le nom de votre nouveau programme  Attention si vous réutilisez le même nom qu’un programme déjà existant, ce dernier sera remplacé par le nouveau |  |
| 5 | Cliquez sur le bouton « Valider » |  |

|  |  |  |
| --- | --- | --- |
| Sauvegarder un programme | | |
| 1 | Sélectionnez votre programme dans la liste des programmes |  |
| 2 | Modifiez votre code |  |
| 3 | Cliquez sur le bouton « Sauvegarder » |  |

|  |  |  |
| --- | --- | --- |
| Supprimer un programme | | |
| 1 | Sélectionnez votre programme dans la liste des programmes |  |
| 2 | Cliquez sur le bouton « Supprimer » |  |
| 3 | Cliquez sur le bouton « Yes » |  |

|  |  |  |
| --- | --- | --- |
| Changer de jardin | | |
| 1 | Sélectionnez votre jardin dans la liste des jardins |  |

|  |  |  |
| --- | --- | --- |
| Démarrer un programme | | |
| 1 | Cliquez sur le bouton « Démarrer » |  |

|  |  |  |
| --- | --- | --- |
| Mettre en pause un programme | | |
| 1 | Cliquez sur le bouton « Pause » |  |

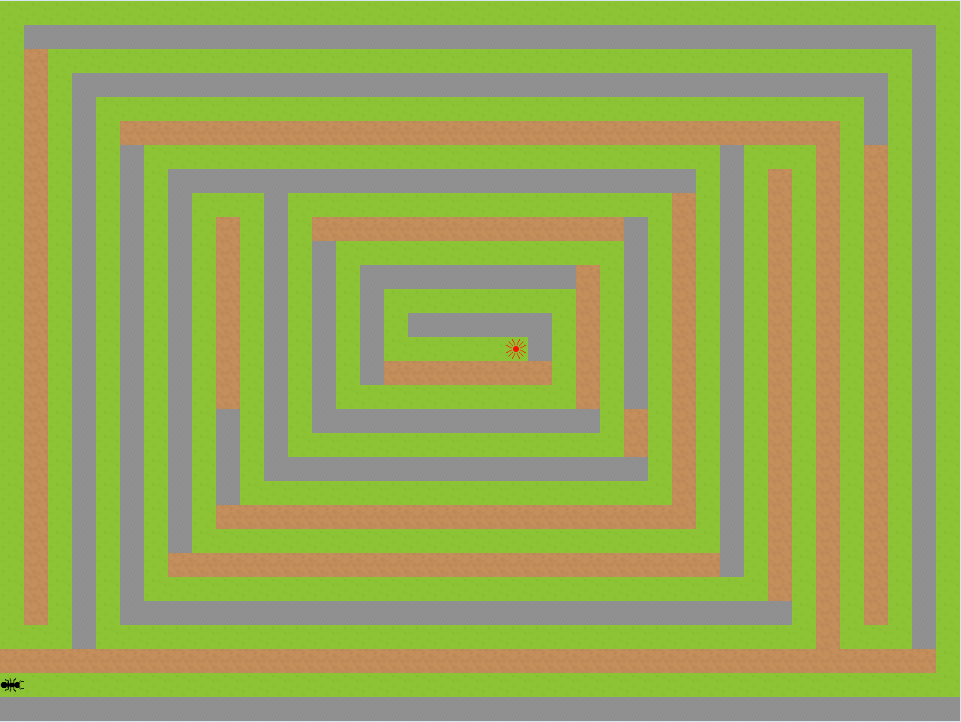
|  |  |  |
| --- | --- | --- |
| Stopper un programme | | |
| 1 | Cliquez sur le bouton « Stop » |  |

# Les jardins

Un jardin est un fichier **.txt** se trouvant dans le dossier « Garden » à la racine du dossier « Krop ». Le fichier doit contenir 30 lignes contenant chacune 40 caractères autorisés.

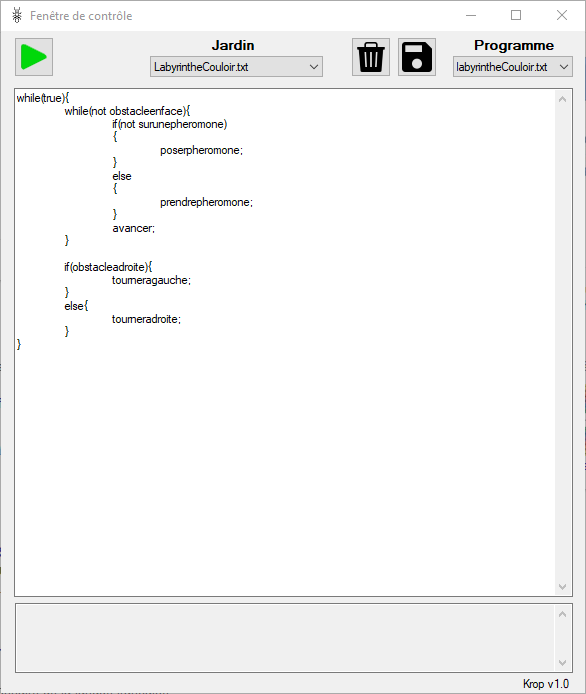
|  |  |  |  |
| --- | --- | --- | --- |
| Labyrinthe.txt | Caractères autorisés | | |
| . | Herbe |  |
| A | Fourmilière (Obstacle) |  |
| R | Roche (Obstacle) |  |
| P | Phéromone |  |
| N | Fourmi en direction du nord |  |
| E | Fourmi en direction de l’est |  |
| S | Fourmi en direction du sud |  |
| W | Fourmi en direction de l’ouest |  |

Le résultat du fichier labyrinthe.txt :

**

# Les programmes

Un programme est fichier **.txt** se trouvant dans le dossier « Code » à la racine du dossier « Krop ». Il peut être composé de tous les éléments se trouvant dans la partie « Syntaxe de Krop ». Un programme permettant de résoudre le labyrinthe du jardin « Labyrinthe.txt » peut par exemple ressembler à ceci :



# Syntaxe de Krop

|  |  |
| --- | --- |
| Glossaire | |
| **Condition** | Ce terme doit être remplacé par une instruction du tableau « Les conditions » |
| *Déclaration* | Ce terme doit être remplacé par une instruction du tableau « Les déclarations de variables » |
| Instruction | Ce terme doit être remplacé par une instruction du tableau « Les instructions » |
| ***NomVariable*** | Ce terme doit être remplacé par le nom de la variable qui doit être uniquement composé de lettre |
| *Expression* | Ce terme désigne une expression algorithmique pouvant contenir des chiffres et des variables Int |
| **NOT** | Inverse le résultat de la condition |
| **\*** | Le symbole « \* » signifie que le terme à gauche peut apparaître de 0 à plusieurs fois |
| **+** | Le symbole « + » signifie que le terme à gauche peut apparaître de 1 à plusieurs fois |
| **?** | Le symbole « ? » signifie que le terme à gauche peut apparaître de 0 à 1 fois |

|  |  |  |
| --- | --- | --- |
| Les instructions de base | | |
| **IF** | If(**NOT?** **Condition**)  {  *Déclaration***\***  Instruction**+**  }  Else  {  *Déclaration***\***  Instruction**+**  } | L’instruction If n’est pas obligée d’avoir une branche « Else » et le NOT n’est pas obligatoire |
| **WHILE** | While(**NOT?** **Condition**)  {  *Déclaration***\***  Instruction**+**  } | Le NOT n’est pas obligatoire |

|  |  |  |
| --- | --- | --- |
| Les déclarations de variables | | |
| **INT** | Int ***NomVariable*** = *Expression* ; | Attention une déclaration de variable ne peut qu’être déclaré en début de programme, While, If et Else |

|  |  |
| --- | --- |
| Les conditions | |
| **True** | Retourne **True** |
| **False** | Retourne **False** |
| **ObstacleEnFace** | Retourne **True** s’il y a un obstacle en face de la fourmi |
| **ObstacleADroite** | Retourne **True** s’il y a un obstacle à droite de la fourmi |
| **ObstacleAGauche** | Retourne **True** s’il y a un obstacle à gauche de la fourmi |
| **SurUneGraine** | Retourne **True** s’il y a une phéromone en dessous de la fourmi |
| *Expression* **=** *Expression* | Retourne **True** si les deux expressions algorithmiques sont égales |
| *Expression* **>** *Expression* | Retourne **True** si la première expression est plus grande que la seconde |
| *Expression***<** *Expression* | Retourne **True** si la première expression est plus petite que la seconde |

|  |  |
| --- | --- |
| Les instructions | |
| **Avancer**; | La fourmi avance d’une case devant elle |
| **TournerADroite**; | La fourmi pivote de 90° vers la droite |
| **TournerAGauche**; | La fourmi pivote de 90° vers la gauche |
| **PoserPheromone**; | La fourmi pose une phéromone sous elle |
| **PrendrePheromone** ; | La fourmi prend une phéromone sous elle |
| **Dire *NomVariable*** ; | Affiche dans le terminal la valeur de la variable |
| **Dire** 1234 ; | Affiche dans le terminal le nombre écrit après l’instruction Dire |
| **Dire ‘**Hello World**’ ;** | Affiche dans le terminal le texte écrit entre les « ‘ »  Attention uniquement des lettres, chiffres et espaces |
| ***NomVariable*** = *Expression* ; | Change la valeur de la variable avec le résultat de l’expression |