

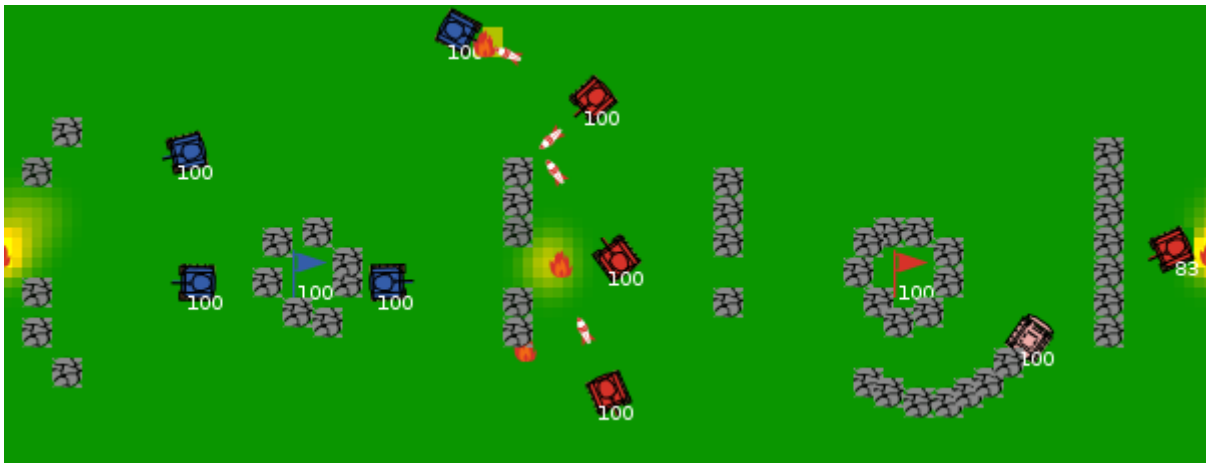
Préambule.....	2
Installation du langage Netlogo	3
Installation du jeu.....	4
Introduction au langage Netlogo.....	6
Piloter les unités.....	6
Piloter les tanks	6
Bibliothèque de codes	9
Attaque.....	9
Défense.....	10
Programmation.....	11
Stratégie Tanks	11
ATTAQUE.....	11
DEFENSE	13
Stratégie Bouclier.....	14
Stratégie Bulldozer : Soyons plus « constructif ».....	15
« Inspection des troupes ».....	15
Bibliothèque de fonctions utiles.....	16
Conclusion	18

Préambule

L'objectif du jeu est *simple* : protéger son propre drapeau et détruire celui de l'équipe adverse.

Pas besoin de savoir programmer pour gagner !

Vous trouverez une courte vidéo d'explication de l'interface à cette adresse :
<http://www.youtube.com/watch?v=03FFtxrAdJs>



Afin de remplir cet objectif vous disposez de 5 unités mobiles à choisir parmi deux types :

- les tanks : ils peuvent tirer mais se déplacent lentement.
- Les bulldozers : ils ne peuvent pas tirer mais peuvent construire des murs de protection, appelés *bunker*. Ils se déplacent plus vite que les tanks et ne sont pas bloqués par les *bunkers*.

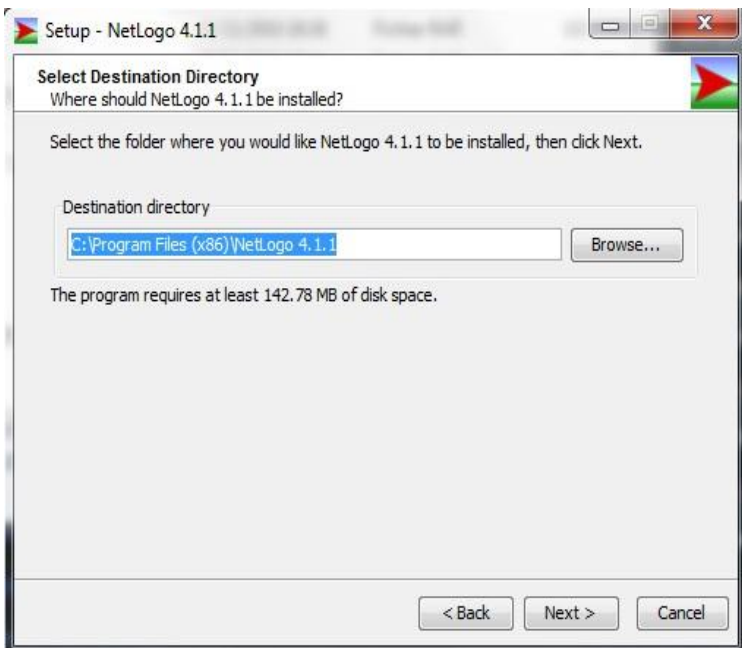
Pilotage

Le pilotage des unités mobiles se fait par l'intermédiaire de 5 fonctions, c'est-à-dire 5 blocs de textes écrits dans le langage *Netlogo*. De nombreuses fonctions et codes d'exemples vous sont fournis pour vous simplifier la tâche.

C'est facile et à portée de tous.
Suivez le guide...

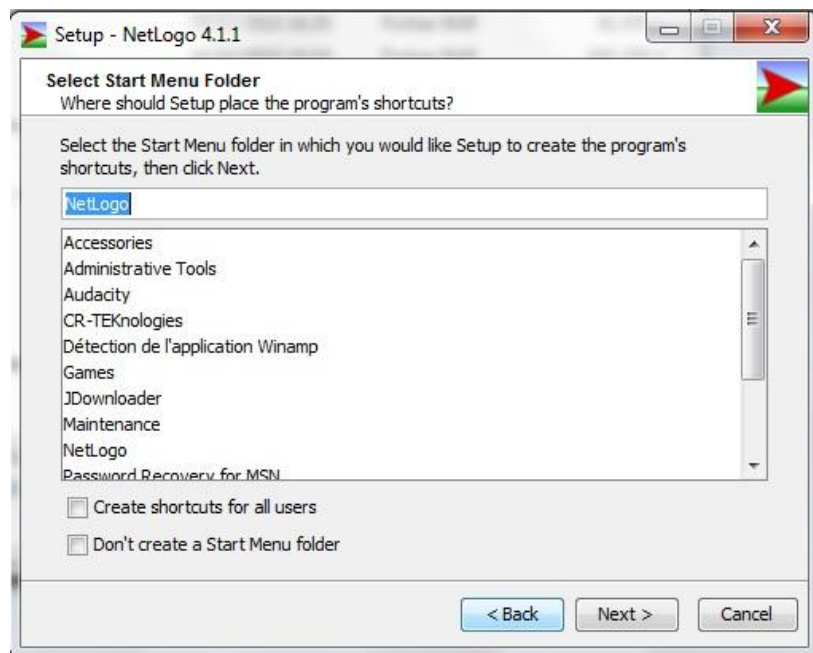
Installation du langage Netlogo

Téléchargez Netlogo 4.1



← Lancez l'installation et suivez les instructions...
(cliquez sur « next »)

Puis : (Cochez l'option « Create shortcuts for all users » afin d'avoir un raccourci sur le bureau.) →



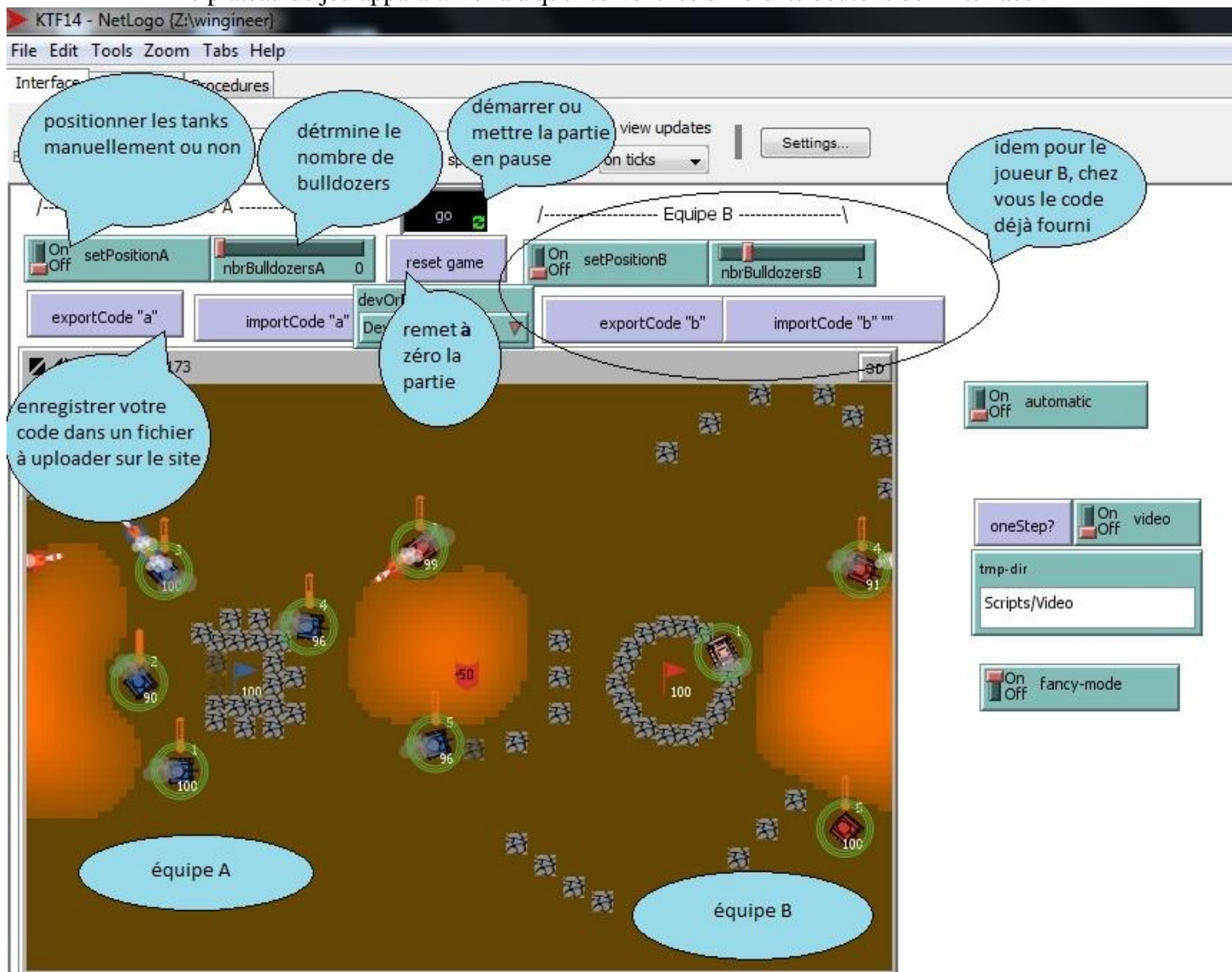
Netlogo est maintenant installé. Pour lancer l'application *Netlogo* et si vous n'avez pas créé de raccourci sur le bureau lors de l'installation, vous devez aller rechercher le fichier « .exe » dans le répertoire où vous avez installé le logiciel, et simplement double-cliquer dessus.

Prochaine étape : ***Installer le jeu !***

Installation du jeu

La plateforme *Netlogo* est installée. Il reste à charger le jeu *WinGineer 3 ICI*.

- Ouvrir le fichier du jeu dans la plateforme *Netlogo* depuis la barre des menus. Cliquez ensuite sur « file », ensuite « open », ce qui ouvrira un sélecteur de fichiers. Choisissez le fichier du jeu que vous venez de télécharger (extension « .nlogo »).
- Le plateau de jeu apparaît. Voilà à quoi servent les différents boutons de l'interface :

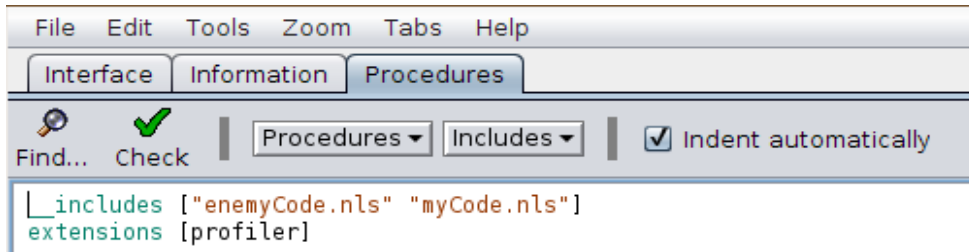


Ça y est vous êtes prêt à programmer en Netlogo !

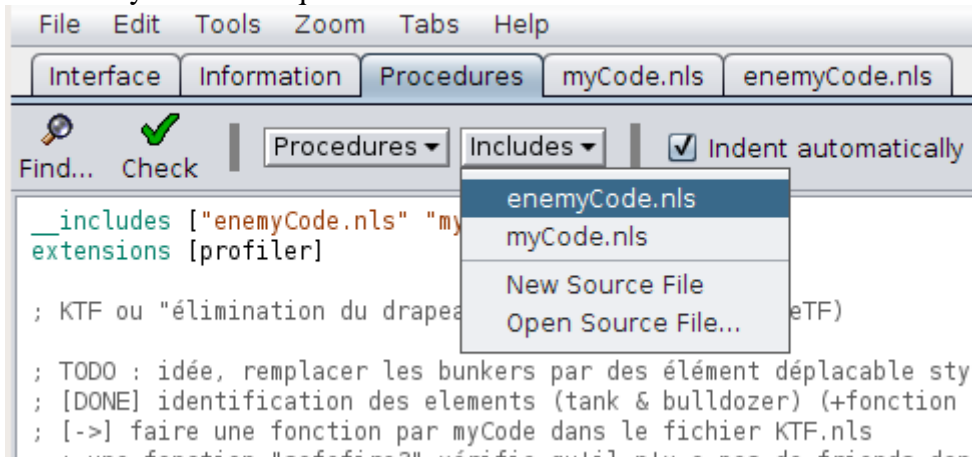
Cette arène utilise les mêmes règles de fonctionnement que l'arène utilisée pour le concours, mais est faite uniquement pour pouvoir tester des robots.

- Rouge et Bleu sont en compétition. Rouge possède déjà un comportement de base pour vous entraîner.
- Vous codez directement des fonctions de la simulation (le comportement de vos robots). Cela vous permettra de profiter de tous les outils de développement du langage Netlogo. Les exemples donnés dans ce document doivent être insérés dans n'importe laquelle des 5 fonctions de chaque candidat. MyCode[1...5] (pour l'équipe « A ») et enemyCode[1...5] (pour l'équipe « B »)

Pour accéder au code cliquez sur l'onglet « Procédure ».



Le texte que vous voyez alors est le programme qui gère l'ensemble du jeu. Il ne nous est pas utile. Le code concernant le comportement des robots est écrit dans 2 fichiers séparés : *myCode.nls*, *enemyCode.nls*. Pour y accéder cliquez sur le menu « includes ».



Ces 2 fichiers sont alors ouverts dans deux onglets supplémentaires...

Rappel et remarques : Vous n'avez besoin de programmer que le comportement de vos robots, tout le reste est déjà fait.

Notez l'utilisation du caractère « ; » permet d'écrire des **commentaires** : des lignes qui ne seront pas interprétées par Netlogo, mais qui permettent de faire des remarques pour vous faciliter la compréhension du code.

Allez dans l'onglet « *interface* » et cliquez sur le bouton « *go* », pour voir votre programme s'exécuter..

Résultat : on voit que l'unité No 1 avance, elle reste vite bloquée par des obstacles, c'est normal, notre programme est vraiment trop simple, nous allons le faire évoluer...

Cliquez à nouveau sur « *go* » pour arrêter le programme, puis sur l'onglet procédure pour retourner au code pour aller le modifier.

```
move ; avancer
right 2 ; tourner à droite de 2°
```

Ce nouveau code demande au robot de tourner légèrement sur sa droite après avoir avancé. Résultat le robot tourne en rond. Changez l'argument (=l'angle) de l'instruction *right* pour voir ce que cela donne. Le tank tourne bien plus que de 2 degrés. Normal, le code des unités est exécuté environ 25 fois par seconde, donc ici, 50°/s.

Note 1 : Il est inutile de mettre plusieurs fois la commande *move*. Seul la première exécution aura un effet, toutes les autres seront ignorées. Ce n'est donc pas comme cela que vous avancerez plus vite!

Note 2 : Idem pour *fire* (qui permet de tirer!). Attention au temps incompressible entre deux tirs (une 1/2 seconde) pour recharger! Vous risquez aussi un blocage dû à la surchauffe si vous tirer avec une trop grande cadence trop longtemps.

Note 3 : si vous désirez que plusieurs unités aient le même comportement, programmez seulement votre unité puis utilisez la commande *same-as* suivie du numéro de l'unité qui contient le code à suivre. C'est **la seule commande** qui marchera sur le jeu en réseau.

Dans l'exemple juste à coté, les unités 3, 4, 5 font la même chose que l'unité 1.

Maintenant que vous avez compris les bases de « commandement » des unités, nous pouvons aller un peu plus loin dans la programmation.

Vous avez maintenant le choix pour aborder ce concours :

- **Soit vous n'avez jamais programmé**, dans ce cas continuez ce tuto pas-à-pas en commençant par la « **bibliothèque de codes** » où vous pourrez piocher des comportements d'unité « standard ».

Composez votre équipe : tanks et bulldozers. Choisissez leurs positions initiales ainsi que les bunkers de base. Une fois que vous serez familiarisé avec ces quelques codes, et que vous aurez testé les comportements, vous identifierez vous-même les modifications à faire pour les adapter à votre stratégie.

Vous pourrez alors lire la partie « programmation » de ce tutoriel, qui vous aidera à créer vous-même vos codes ou



```
File Edit Tools Zoom Tabs Help
Interface Information Procedures myCode.n
Find... Check Close Procedures
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; code candidat : équipe A (ble
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

to myCode1
  move
  right 2
end ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

to myCode2
  move
  fire
end ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

to myCode3
  same-as 1
end ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

to myCode4
  same-as 1
end ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

to myCode5
  same-as 1
end ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

simplement modifier ceux de la « bibliothèque de codes ». Le langage *Netlogo* étant à la fois simple et puissant, les débutants arrivent très rapidement au niveau des candidats confirmés.

- **Soit vous avez déjà des notions de programmation** (quelque soit le langage, java, Javascript, php, C, etc...). Dans ce cas vous pouvez directement passer à la section « Programmation » et les suivantes. Attention cependant, l'avantage que vous pensez avoir sur ceux qui n'ont jamais programmé est assez faible. La difficulté de ce concours est surtout d'ordre stratégique. Le langage *Netlogo* étant à la fois simple et puissant, les débutants arrivent très rapidement au niveau des candidats confirmés.

Pensez aussi que vous n'est pas seul ! Lors de votre inscription, un accès au forum vous a été ouvert. Vous pouvez donc poster vos questions. Pour cela une équipe de 10 étudiants de l'ESIEA vous aideront. Les candidats sont encouragés à s'entraider. Le concours est fait pour que vous ne risquiez rien à aider un concurrent. Il suffit d'avoir réussi à bien se classer une seule fois pour obtenir un ticket pour la finale. Ce ticket vous est acquis quelle que soit l'évolution de votre classement. Vous serez tenu au courant de l'obtention de ce ticket au moment de votre login sur le site web du concours.

<http://www.wingineer.fr>

Le forum:

[http://www.wingineer.fr/forum-\(1421\).cml?FZ=1&g=forum](http://www.wingineer.fr/forum-(1421).cml?FZ=1&g=forum)

Bibliothèque de codes

Nous proposons ici des codes simples (défensif, agressif ou neutre) qui vous permettent de vous focaliser sur les problématiques stratégiques : choix des unités (tank ou bulldozer), comportements, positions initiales, positions des premiers bunkers.

En copiant-collant ces codes, vous pouvez vous lancer dans la compétition.

Attaque

Description du comportement	code
<p>« tank risque tout » : Faire face au drapeau ennemi, tirer (fréquence basse).</p> <p>Si on peut avancer sans risque alors avancer, sinon faire face à l'endroit le moins risqué et avancer.</p> <p>Note : <i>risque</i> signifie la présence d'une explosion devant l'unité en question.</p>	<pre>face enemy-flag fire-one ifelse safe-move? [move] [face safer-place move]</pre> <p>Voir une vidéo d'exemple utilisant ce code : http://www.youtube.com/watch?v=mubxHEWhY6E</p>
<p>« tank nettoyeur de bunkers » : Faire face au drapeau ennemi. Identifier tous les bunkers qui sont dans un cône de longueur 20 cases et d'un angle de 30 degrés (globalement devant). S'il y en a, lui faire face et s'il n'y a pas de rocket sur le chemin(c'est-à-dire si on a pas déjà tiré sur ce bunker), alors tirer. Ensuite, Si on peut avancer sans danger, on avance, sinon on cherche une zone sans danger.</p> <p>Note : ce tank avancera globalement moins vite que le « risque tout » car il fera « le ménage » devant lui.</p>	<pre>face enemy-flag let b bunkers in-cone 20 30 ifelse any? b [face one-of b fire 1] [ifelse safe-move? [move] [face safer-place move]]</pre> <p>Voir une vidéo d'exemple utilisant ce code : http://www.youtube.com/watch?v=CDkJF0T5RIU</p>
<p>« tank prudent » : Ce tank fuit les ennemis qui sont devant lui (cône de 25 cases de rayon et 45 degrés d'angle), non sans leur avoir envoyé une rocket avant de changer de direction. Par défaut il tire devant lui un missile (pour s'ouvrir la voie).</p> <p>Note : ce tank est assez lent (car il fait un détour dès qu'il croise un ennemi), et n'ouvre pas vraiment de voies pour les autres, mais, sur la durée , il a plus de chances d'arriver au but.</p>	<pre>face enemy-flag if safe-fire? [fire 1] let danger enemies in-cone 25 45 if any? danger [face one-of danger if safe-fire? [fire 2] right 45] ifelse safe-move? [move] [face safer-place move]</pre> <p>Voir une vidéo d'exemple utilisant ce code : http://www.youtube.com/watch?v=EWpPS7QdBgI</p>

Défense

Description du comportement	code
<p>« bulldozer constructeur de remparts » : Il construit un rempart autour du drapeau, c'est le choix du positionnement initial qui décidera du rayon du rempart.</p>	<pre>face friend-flag right 90 build move</pre> <p>Voir une vidéo d'exemple utilisant ce code : http://www.youtube.com/watch?v=9kDSYM0TfZY</p>
<p>« tank de défense fixe » : Ce tank, fixe, doit être positionné assez près du drapeau. Dès qu'il voit un danger devant lui et sur les cotés (200 degrés), que ce soit un ennemi ou une rocket (ennemie), il tire dessus.</p> <p>Note: Le fait qu'il ne bouge pas et ne s'occupe que de ce qui est devant lui et sur les côtés lui évite de tirer sur son propre drapeau. Par contre, son coté statique le rend assez vulnérable.</p>	<pre>face friend-flag right 180 let d enemies in-cone 20 200 let r rockets in-cone 20 200 with [enemy?] if safe-fire? [if any? r [face one-of r fire-all] if any? d [face one-of d fire-all]]</pre> <p>Voir une vidéo d'exemple utilisant ce code : http://www.youtube.com/watch?v=H0FYhCmR5XQ</p>
<p>« tank de défense mobile » : Ce tank surveille autour de lui dans un rayon de 40 cases. Il identifie l'ennemi le plus proche et se rapproche de lui jusqu'à une distance de 15 cases. Il s'arrête alors et tire sur l'adversaire.</p> <p>Note: Attention le coté à la fois mobile et acharné de ce tank le pousse parfois à se s'éloigner un peu trop du drapeau qu'il est censé défendre.</p>	<pre>let danger enemies in-radius 40 if any? danger [let proche min-one-of danger [distance myself] face proche let d distance proche ifelse d > 15 [move] [if safe-fire? [fire 1]]]</pre> <p>Voir une vidéo d'exemple utilisant ce code : http://www.youtube.com/watch?v=A8vmYhH5NGI</p>

Programmation

A partir de cette section nous allons voir pas à pas tout ce que vous avez besoin de comprendre pour écrire vous-même vos stratégies.

Pour s'orienter dans l'arène de jeu, il existe un certain nombre de points stratégiques que l'on peut nommer.

friend-flag : le drapeau de son équipe

enemy-flag : le drapeau de l'équipe adverse

friend suivit du numéro d'identification (1-> 5) : une unité de son équipe

enemy suivit du numéro d'identification (1-> 5) : une unité adverse.

etc...

Ainsi pour se déplacer vers le drapeau adverse cela s'écrit :

```
face enemy-flag  
move
```

Pour suivre l'unité No 2 de son équipe, cela s'écrit :

```
face friend 2  
move
```

Stratégie Tanks

ATTAQUE

On se rend rapidement compte que les bunkers gênent la progression, il n'y a que deux façons de faire pour résoudre le problème :

- **Solution 1 : contourner le bunker**

```
face enemy-flag  
ifelse move? [ move ] [left 90 move]
```

Cela peut se traduire par : « faire face au drapeau adverse. Si je peux avancer j'avance, sinon je tourne à gauche de 90 degrés et j'avance ».

Note 1 : l'instruction *ifelse* permet de choisir l'action à mener en fonction de tests. Le test, dans cet exemple, est de savoir si on peut avancer. Si le test est vrai, alors le premier bloc de code est exécuté, s'il est faux c'est le deuxième qui sera exécuté.

Note 2 : On peut regrouper des instructions dans un « bloc » grâce aux crochets. Dans le cas où on ne peut pas avancer, le programme effectue deux opérations : « tourner à gauche » et « avancer ».

Vous pouvez enchaîner autant d'instructions que vous le désirez.

Remarque : rien ne nous assure que l'instruction « move » (après avoir tourné de 90 degrés) sera possible, cela veut dire que cette technique d'évitement est perfectible, mais dans des cas simples, c'est fonctionnel.

Pour plus de clarté , il est conseillé de mettre en forme ce code comme ceci :

```
face enemy-flag  
ifelse move? [ move ]  
[ left 90  
move]
```

On identifie alors mieux le fait qu'il y a plusieurs instructions dans le deuxième bloc de code.

- **Solution 2 : exploser le bunker qui nous bloque.**

```
face enemy-flag
ifelse move? [ move ] [ fire 1]
```

C'est maladroit car les explosions détruisent autant notre tank que le bunker. Pour limiter ce genre de problèmes, la fonction *safe-fire?* permet de savoir si le tir est sûr ou pas.

```
face enemy-flag
ifelse move? [ move ]
[ if safe-fire? [fire 1] ]
```

L'instruction *if* permet de conditionner un seul bloc d'action à un test.
Remarque : on reste bloqué car tirer dans cette situation est dangereux.

Il faudrait être capable de prévoir les éventuels blocages. Pour cela des instructions très pratiques permettent de connaître ce qu'il y a dans l'environnement d'une unité.
Par exemple si on veut savoir s'il y a un bunker devant notre unité à une distance donnée, cela s'écrit :

```
face enemy-flag
ifelse any? bunkers in-cone 15 30 [ fire 1] [move]
```



Ce qu'on peut traduire par : « s'il y a un bunker dans un cône de rayon de 15 et d'angle 30 degrés, alors tirer, sinon avancer. » Le cône est visualisé en rouge sur cette illustration.

La suite d'instructions « *bunkers in-cone 15 30* » désigne un ensemble d'agents, c'est-à-dire tous les éléments qui se trouvent là.

L'instruction « *any?* » qui la précède transforme alors toute l'expression en quelque chose de vrai ou faux : « Y a-t-il quelque chose dans le cône de 15 case et 30 degrés devant moi ? » On peut alors utiliser l'instruction *if* (ou *ifelse*) qui exige d'être suivie par quelque chose qui est soit vrai soit faux (« *true* » ou « *false* »).

Tir et surchauffe :

A chaque tir, le tank chauffe (voir le thermomètre orange à côté du tank). A surveiller de très près. En cas de surchauffe (*heat 50*), vous ne pouvez plus tirer.

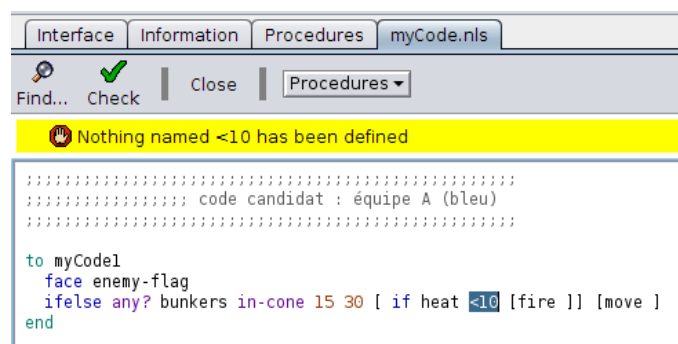
Exemple:

```
if heat < 20 [ fire 3]
```

Si la température est inférieure à 20, alors tirer 3 rockets.

Important Syntaxe : si vous voulez que vos instructions soient bien prises en compte, mettez un espace entre elles.

Exemple: le simple oubli d'un espace entre le symbole « < » et « 10 » rend le programme incompréhensible par Netlogo, et amène alors à un message d'erreur.



DEFENSE

Si un ennemi est trop proche de moi, lui faire face et tirer:

```
if any? enemies in-radius 20 [  
  face one-of enemies in-radius 20  
  fire 1  
]
```

enemies désigne l'ensemble des unités ennemies

in-radius permet de filtrer uniquement les unités qui sont positionnées à une distance inférieure à un seuil donné (ici 20, la portée des rockets).

one-of permet de ne sélectionner qu'une seule unité.

enemies in-radius 40 peut désigner plusieurs unités différentes dans diverses directions.

Si vous oubliez le **one-of** vous aurez un message d'erreur.

Note : **one-of** choisit aléatoirement une unité dans la liste. Si on veut faire une sélection sur un critère donné il faut le spécifier.

Exemple de code pour sélectionner l'unité ennemie la plus proche:

```
face min-one-of enemies [ distance myself]  
myself désigne l'unité que vous commandez.
```

Pour sélectionner l'unité la plus faible, repérez la valeur du bouclier **shield**. Quand elle est nulle, l'unité disparaît.

```
face min-one-of enemies [ shield ]
```

La présence de crochets indique ici non pas un bloc de code mais l'accès à une information provenant d'une autre unité. De même, la variable **shield** dans ce cas désigne la valeur d'état du bouclier d'une unité ennemie. Donc si vous voulez sélectionner l'unité ennemie la plus proche de votre drapeau, cela s'écrit :

```
face min-one-of x [ distance enemy-flag ]
```

Pour faciliter la lecture et aussi la vitesse d'exécution de votre programme, il est préférable d'écrire une seule fois la même chose. Le code précédent s'écrit plus élégamment comme ceci :

```
let x enemies in-radius 40  
if any? x [  
  face one-of x  
  fire 1  
]
```

La première ligne déclare une variable qui stocke la liste de tous les ennemis dans un rayon de 40. L'instruction **let** permet de déclarer une variable (et de lui donner une valeur), l'instruction **set** permet d'en modifier sa valeur. La suite du code est la même que dans l'exemple précédent, mais on réutilise **x** pour nommer cette liste. Il est possible de donner une nouvelle valeur à la variable **x** comme ceci :

```
set x rockets in-radius 20
```

- Stratégie pour tirer sur une unité ennemie qui a un indice de chauffe élevé: (vous devinez pourquoi)

```
let p enemies with [heat > 40 ]  
if any? p [  
  face one-of p  
  fire-one]
```

Remarque : il est possible de combiner une sélection en fonction d'une position et en fonction de variables internes. Par exemple il est inutile de sélectionner les ennemis qui sont au-delà de 20 cases de distance car les rockets sont hors de portée.

Cela s'écrit alors :

```
let p enemies in-radius 20 with [heat > 40 ]
if any? p [
  face one-of p
  fire-one]
```

Stratégie Bouclier

Il faut souvent plusieurs rockets pour détruire un objectif, car chaque élément du jeu (unités mobiles, drapeaux, bunkers) a un bouclier (**shield**). Tant que cette valeur est positive ou nulle l'élément résiste.

Donc surveiller son propre bouclier, et celui des ennemis.

- Changer d'objectif si mon bouclier est inférieur à un seuil donné.

```
ifelse shield > 50 [face enemy-flag] [ face friend-flag]
move
```

- Sélectionner l'ennemi le plus fragile pour lui tirer dessus.

```
let p enemies in-radius 10
if any? p [
  face min-one-of p [shield]
  fire 1
]
```

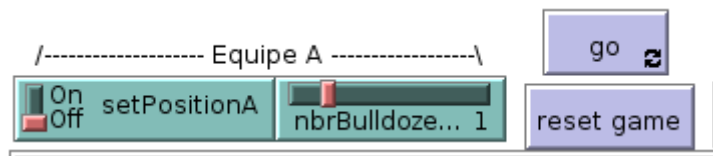
L'instruction **min-one-of** permet de sélectionner un seul élément. Elle choisit le plus petit élément selon le critère défini entre crochets. Notez que l'instruction **max-one-of** existe aussi.

Le bloc **[shield]** n'est pas à comprendre comme un bloc de code conditionné à l'instruction précédente, mais comme l'extraction d'une variable hors de son contexte. Dans cet exemple, on sélectionne dans la liste *p* l'élément qui a le bouclier le plus faible. La paire de crochets sert à signaler que la variable **shield** n'est pas le bouclier de votre unité mais de chacune des unités contenues dans la variable *p*.

Stratégie Bulldozer : Soyons plus « constructif »...

A tout moment, vous pouvez constituer votre équipe de tanks et de bulldozers. Notez seulement que les bulldozers **ne peuvent pas tirer**.

Pour créer des bulldozers, modifiez « *nbrBulldozers* ». Mettre sur « ON » l'interrupteur « *setPosition* » et cliquez ensuite sur le bouton « *setup* ».



Vous pouvez alors positionner vos unités ainsi que quelques bunkers. Remettez ensuite l'interrupteur « *setPosition* » sur OFF.

Note : identifiez bien quelles unités sont les bulldozer et quels unités sont les tanks, car l'instruction « *fire* » n'aura aucune action dans le code d'un bulldozer.

Les bulldozers ne peuvent pas tirer mais ils construisent des bunkers pour protéger votre drapeau ou pour ralentir la progression de l'ennemi.

Voici un code simple permettant de faire construire un rempart autour de votre drapeau :

```
face friend-flag  
right 90  
build  
move
```

Ce qui peut se traduire par : « faire face à notre drapeau, tourner de 90 degrés, construire un bunker, avancer. »

Notez qu'il est possible de conserver des informations avec la variable *memory*.

« Inspection des troupes »

À tout moment dans le jeu, après arrêt de la simulation (bouton « **Go** ») vous avez accès aux informations de chacune des unités par un *clik droit*.

Sélectionnez l'unité qui vous intéresse et choisissez l'option « *inspect* »...

S'ouvre alors une fenêtre vous donnant accès à 3 choses très pratiques pour vérifier des comportements et



téster des bouts de code :

1. un zoom sur l'unité en question. Vous pouvez ajuster ce zoom à votre convenance en vous servant du curseur qui est juste sous l'image.
2. Une vue sur toutes les variables de cette unité (dont les niveaux de bouclier ou de chauffe)
3. Tout en bas, une zone permettant de tester des bouts de code.

Bibliothèque de fonctions utiles

- **fire n** : tire n rockets (il n'y a pas surchauffe). **n** est le nombre de rockets que le tank essayera de tirer. Attention, les rockets décomptées sont « en l'air ». **fire-one** : Cette fonction permet de tirer au débit le plus bas et donc de limiter la chauffe.
- **fire-one** : attendre que la température ait baissé avant de tirer à nouveau.
- **fire-all** : Comme **fire**, tire jusqu'à atteindre la surchauffe, mais « ventile » les tirs sur des angles différents pour éviter que la première explosion ne déclenche toutes les autres, ce qui finirait par infliger des dégâts au tireur.

- **enemy n** : identifie l'unité ennemie numéro n

Pour suivre l'unité ennemie 3, il suffit d'écrire :

```
face enemy 3
move
```

- **enemies** : désigne tous les ennemis (tanks et bulldozers)

Exemple pour choisir un ennemi au hasard et lui tirer dessus:

```
face one-of enemies
fire 1
```

- **safe-fire?** Répond par *true* ou *false* si le tir qu'on s'apprête à faire risque de nuire à l'équipe.

Exemple pour limiter les tirs dangereux :

```
if safe-fire? [fire 1]
```

Avec cette fonction, il se peut que, dans certains cas, quelques unités amies soient touchées quand même. Mais il pourrait être stratégiquement de sacrifier une unité.

- **safe-fire?** Répond par *true* ou *false* si le tir qu'on s'apprête à faire risque de nuire à l'équipe.

Exemple pour limiter les tirs dangereux :

```
if safe-fire? [fire 1]
```

Avec cette fonction, il se peut que, dans certains cas, quelques unités amies soient touchées quand même. Mais il pourrait être stratégiquement de sacrifier une unité.

- **friend-flag, enemy-flag** : désigne le drapeau ami ou ennemi

- **friend n** : identifie l'unité amie numéro n

Pour suivre l'unité 3 amie, il suffit d'écrire :

```
face friend 3
move
```

(Évidemment il ne faut pas mettre ce code dans la fonction gérant l'unité No 3...)

- **friend? x** : recherche si l'unité x est une unité amie ?

Exemple : s'éloigner des autres tanks amis (pour ne pas offrir une cible facile) :

```
let x one-of other tanks in-radius 10 ; soit x un des tanks inférieur à une distance de 10
if friend? x [ ; si c'est un ami
face x ; lui faire face
right 180 ; faire demi tour
move ; bouger
```

]

- **enemy? x** : recherche si l'unité x est une unité ennemie ?

Exemple : voir l'exemple précédent.

```
let x one-of other tanks in-radius 10 ; soit x un des tanks inférieur à une distance de 10
if enemy? x [ ; si c'est un ennemi
face x ; lui faire face
fire 1 ; tirer...
]
```

- **safe-move?** : cette fonction répond à la question : « puis-je avancer sans danger », pour éviter d'avancer dans le souffle d'une explosion.

Exemple :

```
if safe-move ? [move]
```

- **safe-here?** : cette fonction répond à la question « suis-je en danger? ».

Exemple : « si on est en danger se déplacer ».

```
if not safe-here? [ move ]
```

L'instruction **not** permet d'inverser le sens logique dans un test. Ici on ne décide de se déplacer que si le lieu actuel n'est **pas** sûr.

- **safer-place** : cette fonction donne l'orientation présentant le moins de danger.

Exemple : faire une unité « lâche », qui ne fait que fuir le danger

```
face safer-place
move
```

- **facing-me** : fonction à utiliser avec l'instruction *with*.

Elle permet de sélectionner, les dangers les plus menaçants en mesurant l'angle qui nous sépare du danger, comme l'angle de tir d'une rocket par exemple. Plus l'angle est important, mais le danger est grand.

Voici un exemple d'utilisation :

```
let r rockets in-radius 15 with [facing-me < 10]
if any? r [ face one-of r
left 90
move
]
```

Qui peut se traduire en : « sélectionner les rockets dans un rayon de 15 cases (et) qui sont face à moi (à 10 degrés près..,S'il y en a, en choisir une, lui faire face, tourner à gauche (pour l'esquiver) puis avancer. »

Conclusion

Vous avez là déjà quelques bases pour élaborer des stratégies intéressantes, à vous de jouer ;-).

Vous trouverez encore bien d'autres aides et conseils sur le **forum** du site *Wingineer*.

Vous pourrez trouver un **dictionnaire des fonctions disponibles en Netlogo** en explorant le menu « help » de l'interface ou à cette adresse : <http://ccl.northwestern.edu/netlogo/docs/primindex.html>.

Attention pour que votre code soit valide, vous devez respecter **la règle du jeu** :

- Vous êtes autorisés à lire toute information sur le jeu.
- Vous êtes autorisés à agir sur le jeu mais uniquement avec les instructions : *move, fire, right, left, face, build* appliquées à vos unités.
- Bien sûr vous avez le droit d'utiliser toutes les instructions de *Netlogo* vous aidant à produire les décisions de votre robot.
- Vous ne devez par contre **en aucun cas agir directement** sur le jeu (en essayant de modifier directement son score, par exemple). Cela serait considéré comme de la triche, votre code supprimé et votre score mis à 0 jusqu'à soumission d'un nouveau code respectant les règles.

Bonne chance à tous !