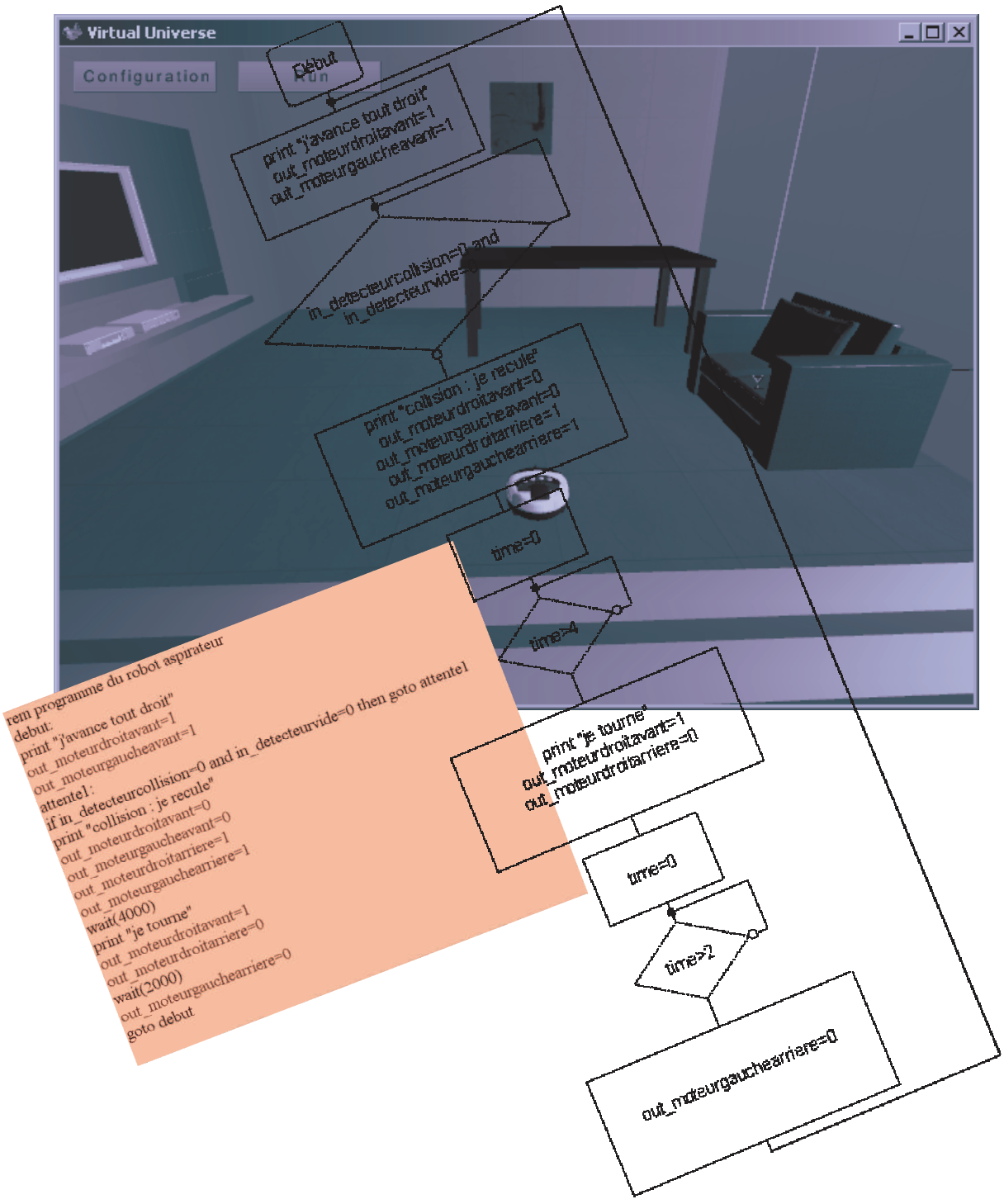
Virtual Universe

Pilotage des modèles 3D en langage Basic ou organigramme



Sommaire

Généralités

Ce manuel à pour but d'expliquer l'utilisation de l'éditeur de scripts et l'éditeur d'organigrammes intégrés à Virtual Universe. Ces éditeurs permettent la création de programmes en langage Basic ou sous forme de représentation organigramme dans le but de piloter les modèles numériques de Virtual Universe.

Configuration de Virtual Universe

Pour que l'éditeur de script puisse dialoguer avec Virtual Universe, il faut modifier la propriété suivante dans Virtual Universe :

- Univers : « serveur TCP » : choisir « True ».

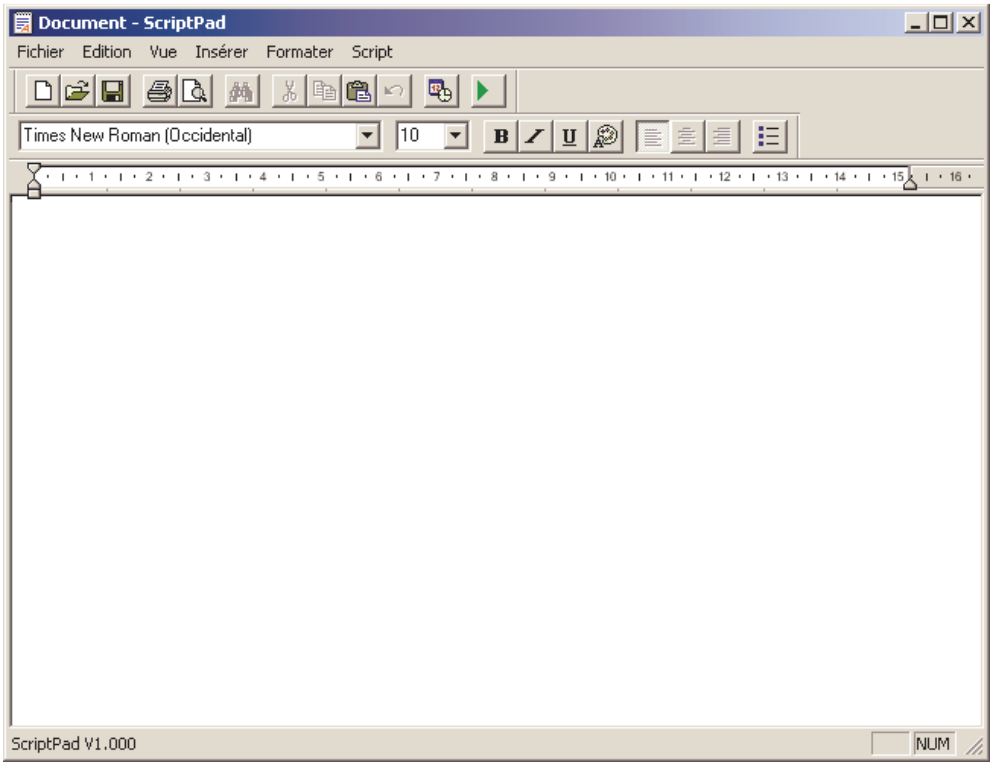
Editeur de script

L'éditeur de script permet de décrire le fonctionnement d'un modèle simulé en langage Basic.

Lancement de l'éditeur de script

L'éditeur de script se trouve dans le répertoire d'installation de Virtual Universe et se nomme « ScriptPad.exe ». Un raccourci de lancement est également créé dans le menu Démarrer/programme de Windows à l'installation de Virtual Universe.

L'éditeur se présente sous la forme d'un simple éditeur de texte :



Langage

Le langage utilisé est un langage Basic « minimaliste ». Le fichier d'aide nommé basic_api.chm qui se trouve dans le répertoire d'installation de Virtual Universe donne l'ensemble des éléments pour l'utilisation de ce langage.

Complément de langage

En complément des instructions du langage Basic, deux instructions ont été ajoutées pour gérer les évènements liés au temps (attente, etc.) :

Wait(durée en millisecondes) : provoque une attente de la durée spécifiée en milliseconde.

Time() : retourne la durée écoulée depuis le lancement de l'exécution du script en millisecondes.

Accès aux entrées / sorties de Virtual Universe

Si les noms des variables utilisées dans le script commencent par la suite de caractères « in_ » ou « out_ », alors les variables concernées sont considérées comme des variables externes appartenant à Virtual Universe.

Plus précisément, les variables commençant par « in_ » seront lues depuis Virtual Universe et les variables commençant par « out_ » seront écrites vers Virtual Universe.

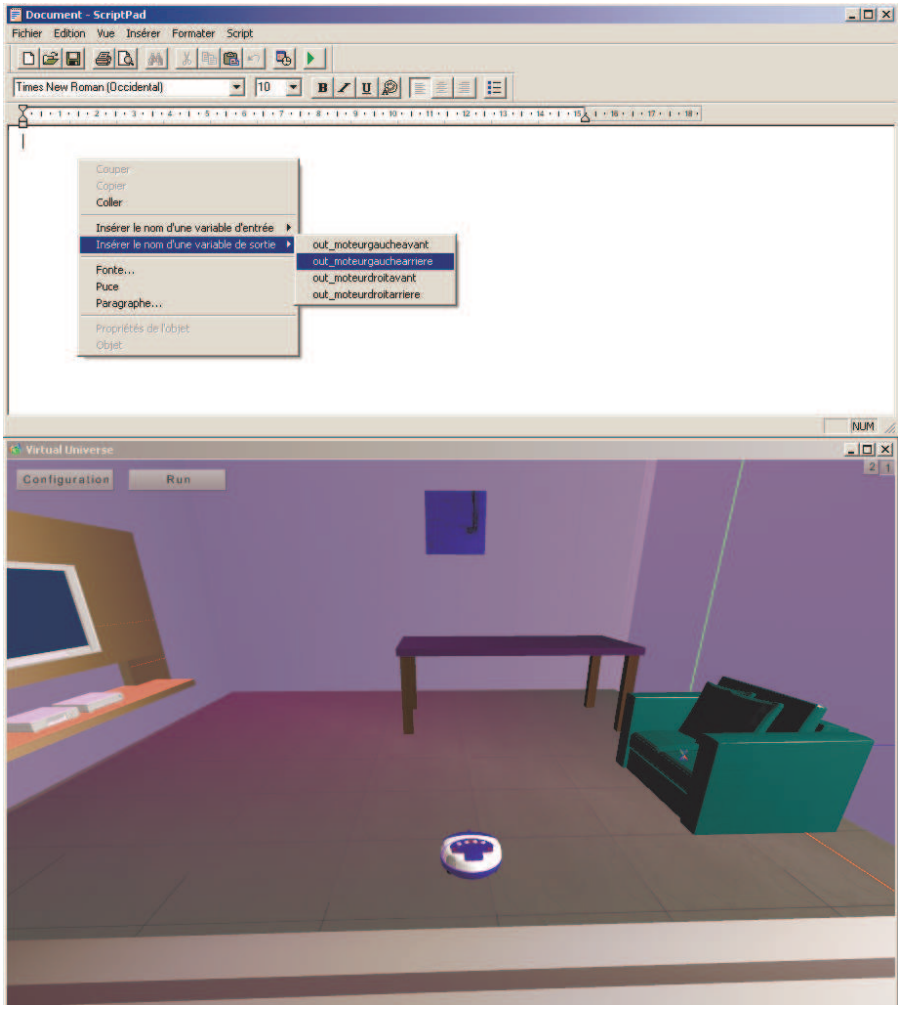
Les caractères suivants « in_ » ou « out_ » correspondent à un nom de comportement ou un alias de nom de comportement de Virtual Universe.

Par exemple, « in_capteur » sera une variable lue depuis l'état du comportement nommé « capteur » de Virtual Universe. « out_moteur » sera une variable écrite vers l'état du comportement nommé « moteur » dans Virtual Universe.

Il est possible d'insérer directement un nom d'entrées sorties disponible par rapport au fichier chargé dans Virtual Universe. Pour ceci, cliquez avec le bouton droit de la souris sur la page dans l'éditeur de script et sélectionnez « Insérer le nom d'une variable d'entrée » ou « Insérer le nom d'une variable de sortie ».

Pour être accessibles sous ces rubriques, les noms de variables doivent être renseignés sous la rubrique « alias » dans les propriétés des comportements de Virtual Universe.

Exemple :



Exécution

Pour lancer l'exécution du script, cliquez sur le bouton « PLAY » (triangle vert) dans la barre d'outils ou sélectionnez « Lancer » dans le menu « Script ». Virtual Universe doit être lancé et le projet Virtual Universe doit être réglé comme serveur TCP (paramètre dans les propriétés de l'univers). Si ces conditions ne sont pas satisfaites, un message d'erreur de connexion à Virtual Universe est affiché au lancement du script.

Structure du programme

Le programme s'exécute jusqu'à ce que la dernière ligne ait été atteinte. Il convient donc de créer une boucle pour que le programme ne se termine pas immédiatement. Cette boucle peut être réalisée avec un saut ou une structure de répétition.

Exemple 1 :

```
maboucle:

rem écrire ici le corps du programme

goto maboucle
```

Exemple 2 :

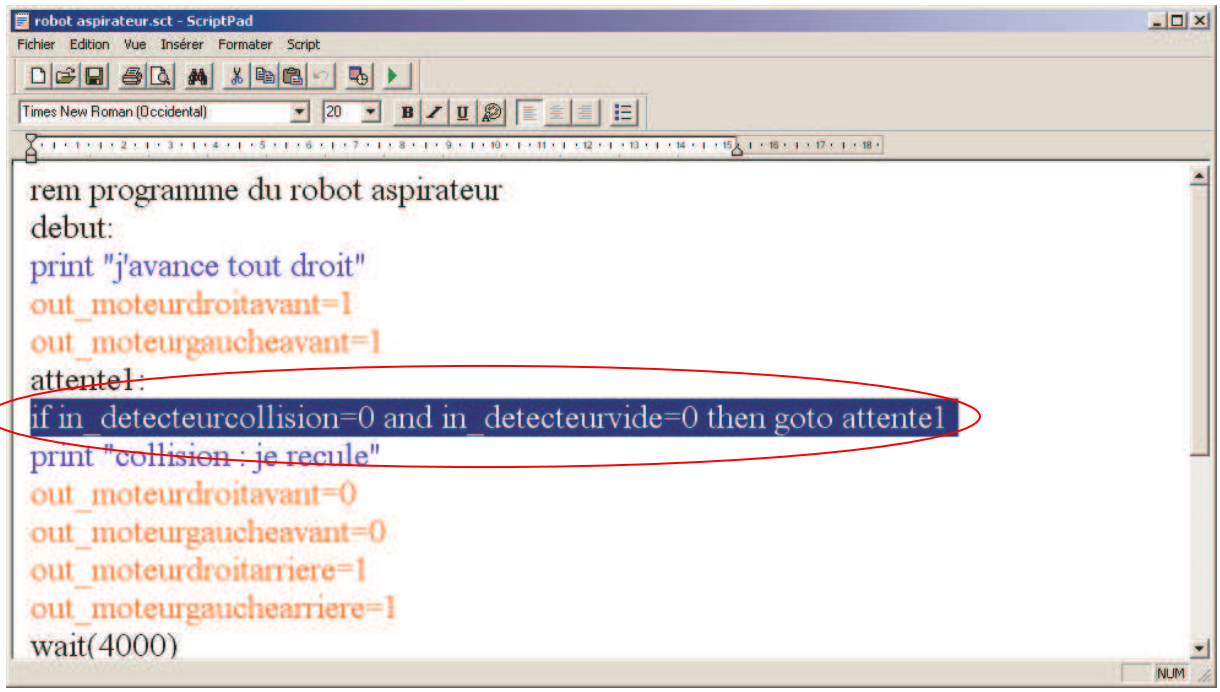
```
do

rem écrire ici le corps du programme

loop
```

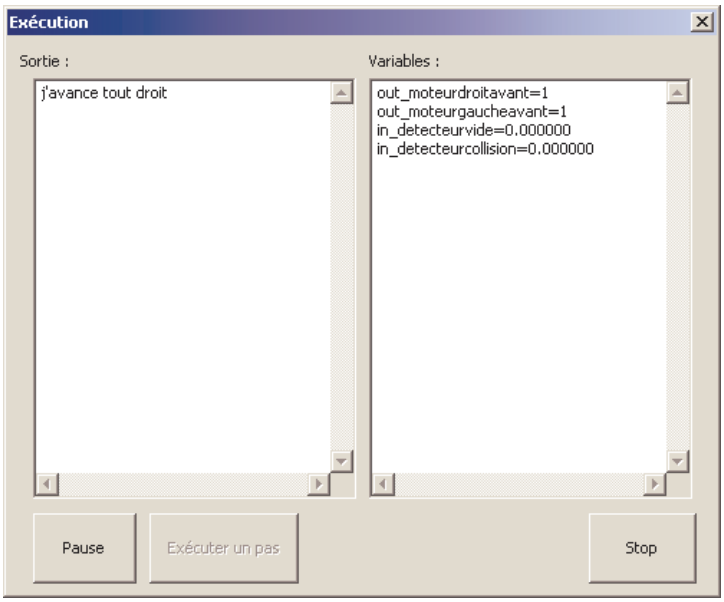
Etat du programme pendant l'exécution

Pendant l'exécution, la ligne courante (le pointeur de programme) est affichée en dynamique dans l'éditeur. Exemple :



Fenêtre de mise au point

Au lancement du script, la fenêtre suivante s'ouvre :



L'élément « Sortie » affiche les sorties provoquées par l'instruction « print » du langage Basic. L'élément « Variables » liste l'ensemble des variables utilisées par le script ainsi que l'état pour chacune de celles-ci.

Les boutons « Pause », « Exécuter un pas » et « Stop » permettent respectivement de mettre le programme en pause ou de le relancer, d'avancer d'un pas ou de mettre fin à son exécution.

Exécution sur des PCs différents

Par défaut, l'éditeur de script se connecte à Virtual Universe lancé sur le même PC. Il est possible de se connecter à un ordinateur distant en ajoutant la directive #SERVERADDR= sur une ligne du script.

Par exemple :

#SERVERADDR=192.168.1.2

Forcera une connexion à Virtual Universe lancé sur le PC d'adresse IP 192.168.1.2

Le paramètre se trouvant « = » peut être une adresse IP ou un nom réseau.

Il est également possible de définir le port à utiliser (par défaut 6000) avec la directive #SERVERPORT= (veiller, si vous modifiez cette valeur à la modifier également dans le paramétrage de Virtual Universe : propriété « Port serveur TCP » de l'univers).

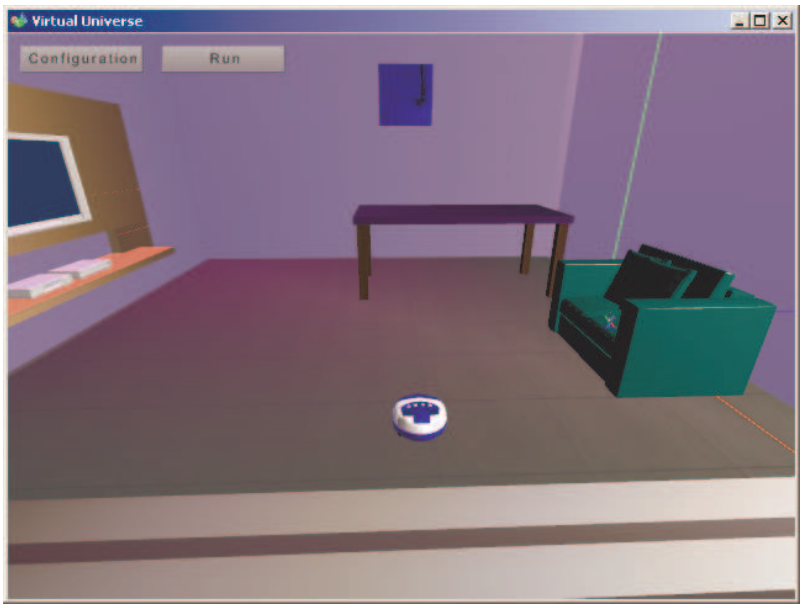
Exécution de plusieurs instances

Il est possible de lancer plusieurs instances de l'éditeur de script et d'activer simultanément l'exécution de ces différentes instances. L'exemple Virtual Universe « 2 robots nxt pilotés en basic » et les 2 fichiers scripts « nxt1 » et « nxt2 » associés illustrent le pilotage simultané de 2 robots NTX à partir de 2 scripts indépendants.

Exemples de scripts

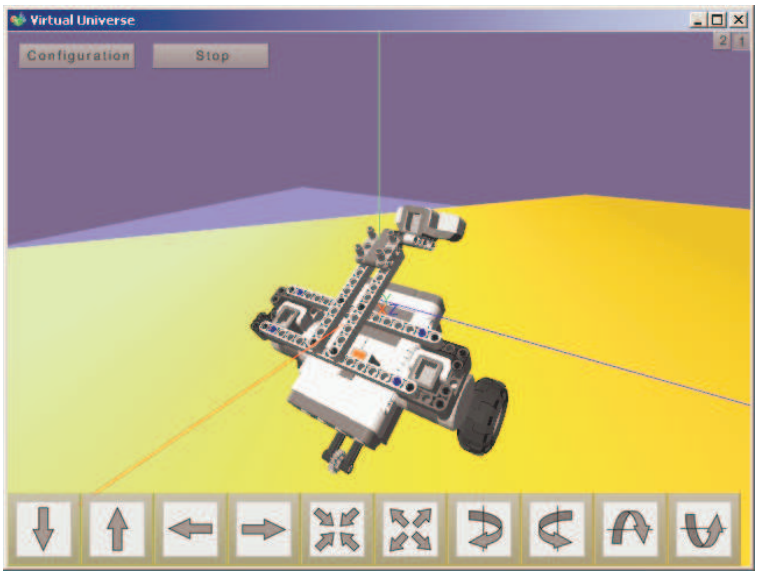
Le sous-répertoire « Exemples » du répertoire d'installation de Virtual Universe contient des exemples Virtual Universe pilotables par script ainsi que les scripts correspondants.

Robot aspirateur




```
rem programme du robot aspirateur
debut:
print "j'avance tout droit"
out_moteurdroitavant=1
out_moteurgaucheavant=1
attente1:
if in_detecteurcollision=0 and in_detecteurvide=0 then goto attente1
print "collision : je recule"
out_moteurdroitavant=0
out_moteurgaucheavant=0
out_moteurdroitarriere=1
out_moteurgauchearriere=1
wait(4000)
print "je tourne"
out_moteurdroitavant=1
out_moteurdroitarriere=0
wait(2000)
out_moteurgauchearriere=0
goto debut
```

Robot NXT




```
REM Exemple de programme NXT
do
print "tout droit"
out_puissancemoteur1=50
out_puissancemoteur2=50
out_moteur1avant=1
out_moteur2avant=1
do until in_capteurir>100
loop
print "tourner"
out_puissancemoteur1=70
out_puissancemoteur2=70
out_moteur1avant=0
out_moteur1arriere=1
do until in_capteurir<100
loop
out_moteur1arriere=0
loop
```

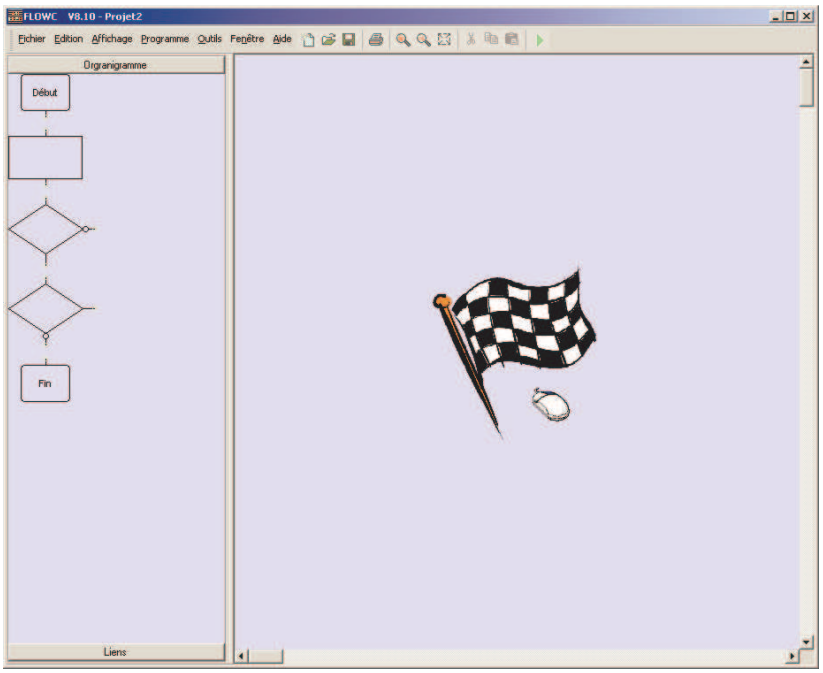
Editeur d'organigrammes

L'éditeur d'organigrammes permet de décrire le fonctionnement d'un modèle simulé sous la forme d'un organigramme.

Lancement de l'éditeur d'organigramme

L'éditeur d'organigramme se trouve dans le sous-répertoire « ae » du répertoire d'installation de Virtual Universe et se nomme « Flowc.exe ». Un raccourci de lancement est également créé dans le menu Démarrer/programme de Windows à l'installation de Virtual Universe.

L'éditeur se présente sous la forme suivante :



Création des programmes.

Saisissez les blocs sur la partie gauche de l'écran et déplacez-les sur la page.

Vous pouvez également ouvrir un assistant en cliquant avec le bouton droit de la souris sur la page et en choisissant « Ajouter un objet ».

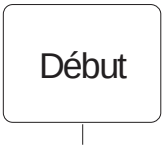
Pour documenter les blocs, double cliquez dessus, la boîte de dialogue des propriétés de l'objet s'ouvre alors.

Les blocs peuvent être déplacés et redimensionnés (saisir un des carrés affichés autour des blocs lorsqu'ils sont sélectionnés). Des pastilles jaunes montrent les connexions des blocs. Les blocs peuvent être placés pour que les pastilles coïncident, des liaisons peuvent également être créées en cliquant sur les pastilles (le curseur affiche une forme de cible) puis en cliquant sur le chemin où doit être tracé la liaison et en finissant par un clic sur la pastille de destination.

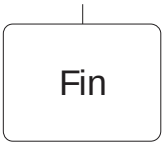
Langage

La représentation utilisée est l'organigramme. Le fonctionnement est un compromis entre un langage séquentiel multitâches (type Grafcet) et un langage informatique. En pratique, ceci signifie que plusieurs branches d'organigrammes pourront s'exécuter simultanément.

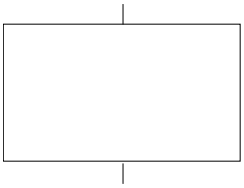
Les blocs



Ce bloc marque le début du programme. Il devrait y avoir au moins un bloc « Début » dans un programme. Ce bloc est actif au démarrage du programme, il active le bloc connecté à sa sortie dés le lancement du programme.

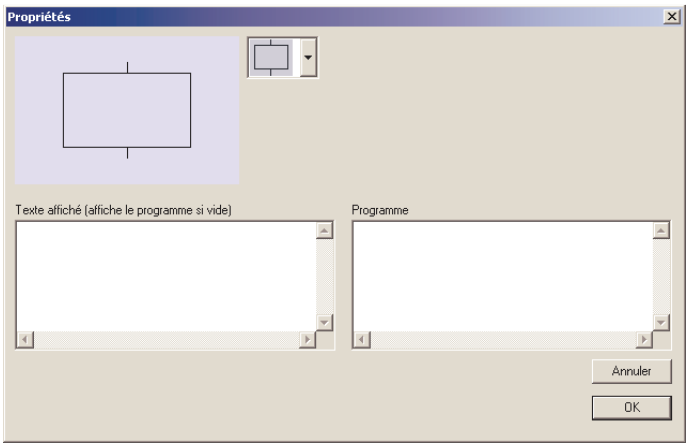


Ce bloc marque la fin du programme. Ce bloc peut ne pas être utilisé si le programme ne doit jamais se terminer.



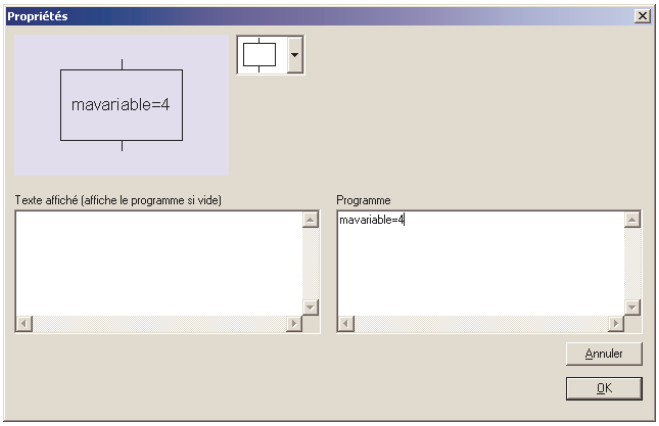
Ce bloc est un bloc d'action. Les instructions écrites dans les propriétés de ce bloc seront exécutées une fois lorsque ce bloc sera activé, le bloc est ensuite désactivé puis le bloc connecté à la sortie (connexion du bas) est activé.

Voici la boîte de dialogue des propriétés du bloc d'action :



La zone programme doit contenir le code à exécuter. La syntaxe à utiliser est celle du langage Basic décrite dans la première partie de ce manuel.

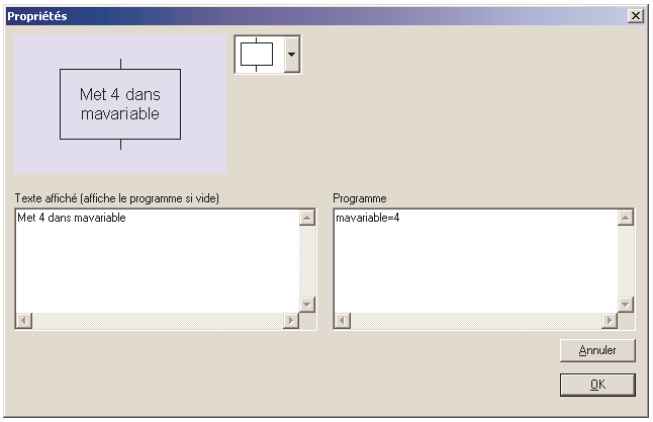
Par exemple :



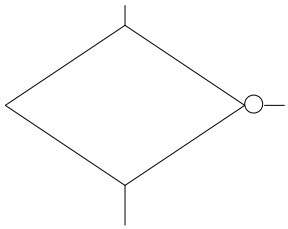
Placera la valeur 4 dans la variable nommée « mavariable ».

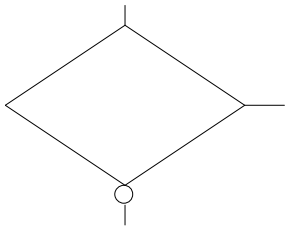
La zone « Texte affiché » peut contenir un texte qui sera affiché à la place du programme sur la page.

Par exemple :

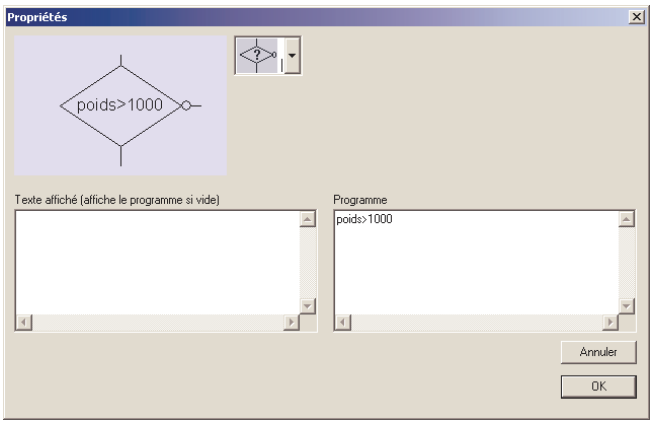


Ceci n'a un impacte que sur l'affichage, le code exécuté restera (pour cet exemple) « mavariable=4 ».

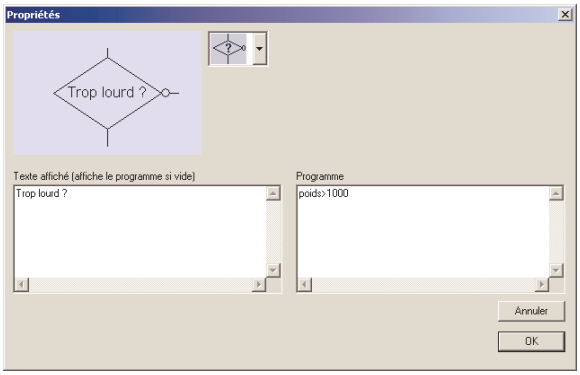




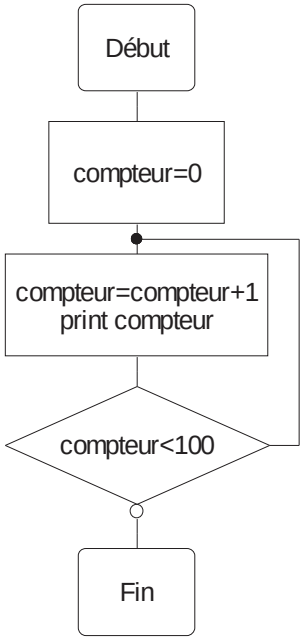
Ces blocs sont des tests. L'exécution du programme continuera sur une des deux branches de sorties : la branche marquée d'un cercle (à droite ou en bas selon le bloc) si le test est faux ou l'autre branche (en bas ou à droite selon le bloc) si le test est vrai. La syntaxe à utiliser est celle d'un test du langage Basic : une expression telle qu'on peut l'écrire derrière une instruction IF du langage Basic. Par exemple :



Ici aussi la zone texte permet de substituer un texte pour l'affichage sur la page. Par exemple :

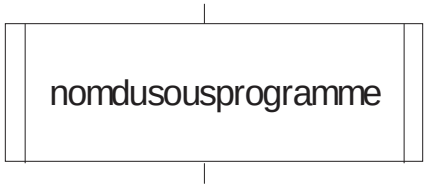


Voici un exemple de programme réalisant une boucle qui affiche les nombres de 1 à 100 :

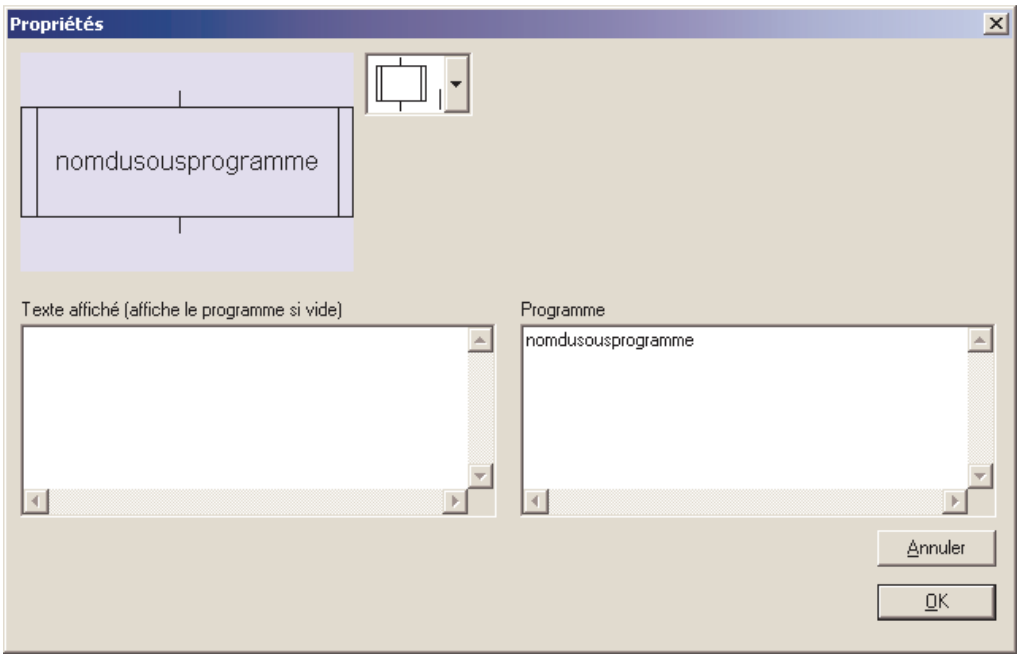


Sous-programmes

Le bloc d'appel de sous-programme est le suivant :



Le nom du sous-programme doit être renseigné sous la rubrique « Programme » des propriétés de ce bloc :



Le sous-programme doit débuter par un bloc « Début de sous-programme » :

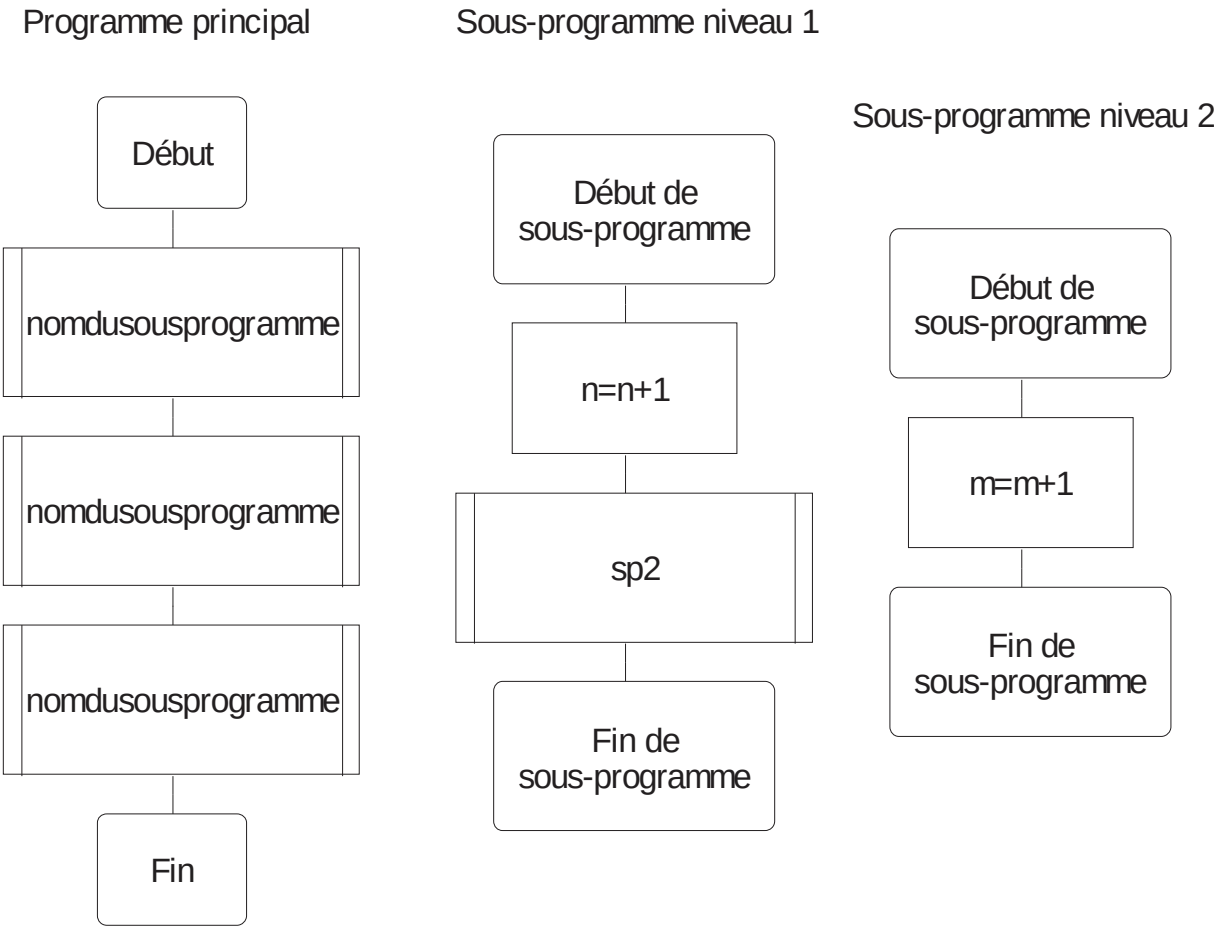
Et se terminer par un bloc « Fin de sous-programme » :

Pour ces 2 blocs, le nom du sous-programme doit aussi être documenté sous la rubrique « Programme ».

Règles concernant les sous-programmes :

- Des appels imbriqués sont utilisables,
- Les sous-programmes ne sont pas réentrants.

Exemple de sous-programme



Accès aux entrées / sorties de Virtual Universe

Si les noms des variables utilisées dans le script commencent par la suite de caractères « in_ » ou « out_ », alors les variables concernées sont considérées comme des variables externes appartenant à Virtual Universe.

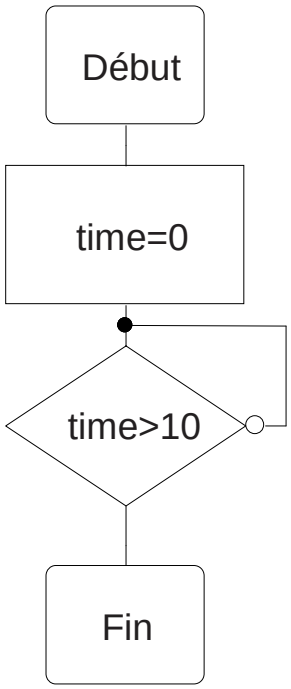
Plus précisément, les variables commençant par « in_ » seront lues depuis Virtual Universe et les variables commençant par « out_ » seront écrites vers Virtual Universe.

Les caractères suivants « in_ » ou « out_ » correspondent à un nom de comportement ou un alias de nom de comportement de Virtual Universe.

Par exemple, « in_capteur » sera une variable lue depuis l'état du comportement nommé « capteur » de Virtual Universe. « out_moteur » sera une variable écrite vers l'état du comportement nommé « moteur » dans Virtual Universe.

Gestion du temps

La structure suivante permet de créer une attente dans un programme, voici par exemple une attente de 10 secondes :



Les variables commençant par « time » sont des variables spéciales incrémentées au rythme des secondes. Si plusieurs branches d'organigramme doivent pouvoir évoluer indépendamment, il faut utiliser des variables « time » différentes. Par exemple : « time1 », « time2 », etc.

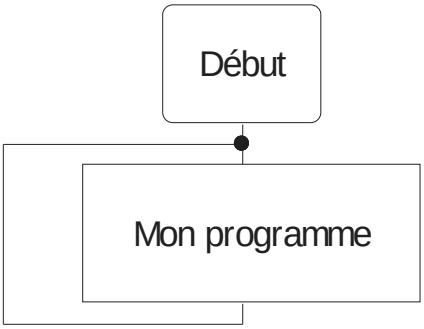
Exécution

Pour lancer l'exécution du programme, cliquez sur le bouton « PLAY » (triangle vert) dans la barre d'outils ou sélectionnez « GO » dans le menu « Programme ». Virtual Universe doit être lancé et le projet Virtual Universe doit être réglé comme serveur TCP (paramètre dans les propriétés de l'univers). Pour mettre fin au programme, cliquez sur ce même bouton (il prend l'aspect d'un bouton stop quand le programme est lancé).

Structure du programme

Le programme s'exécute jusqu'à ce qu'un bloc fin ait été atteint. Il convient donc de créer une boucle pour que le programme ne se termine pas immédiatement. Cette boucle peut être réalisée en traçant une liaison qui reboucle le programme.

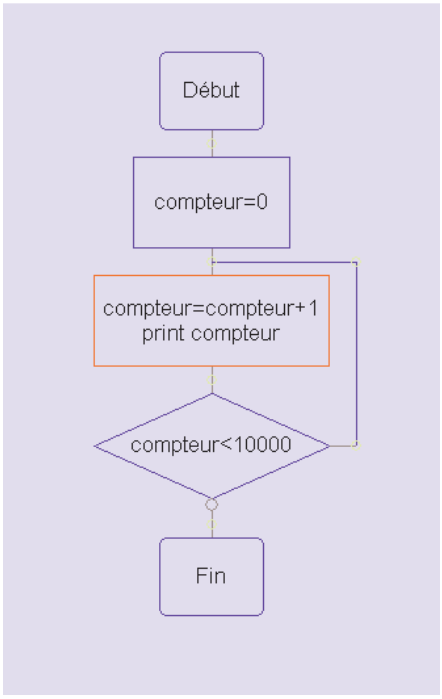
Exemple :



Etat du programme pendant l'exécution

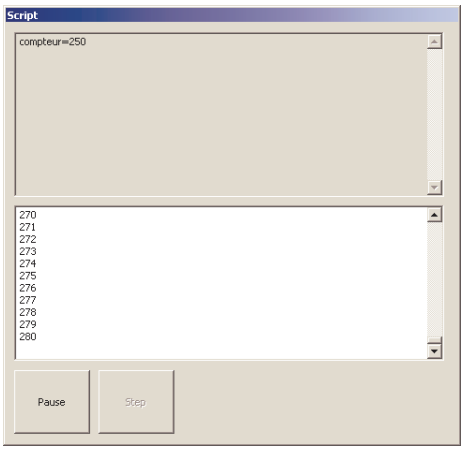
Pendant l'exécution, le pointeur de programme est affiché (bloc en rouge) en dynamique dans l'éditeur.

Exemple :



Fenêtre de mise au point

Au lancement de l'exécution du programme, la fenêtre suivante s'ouvre :

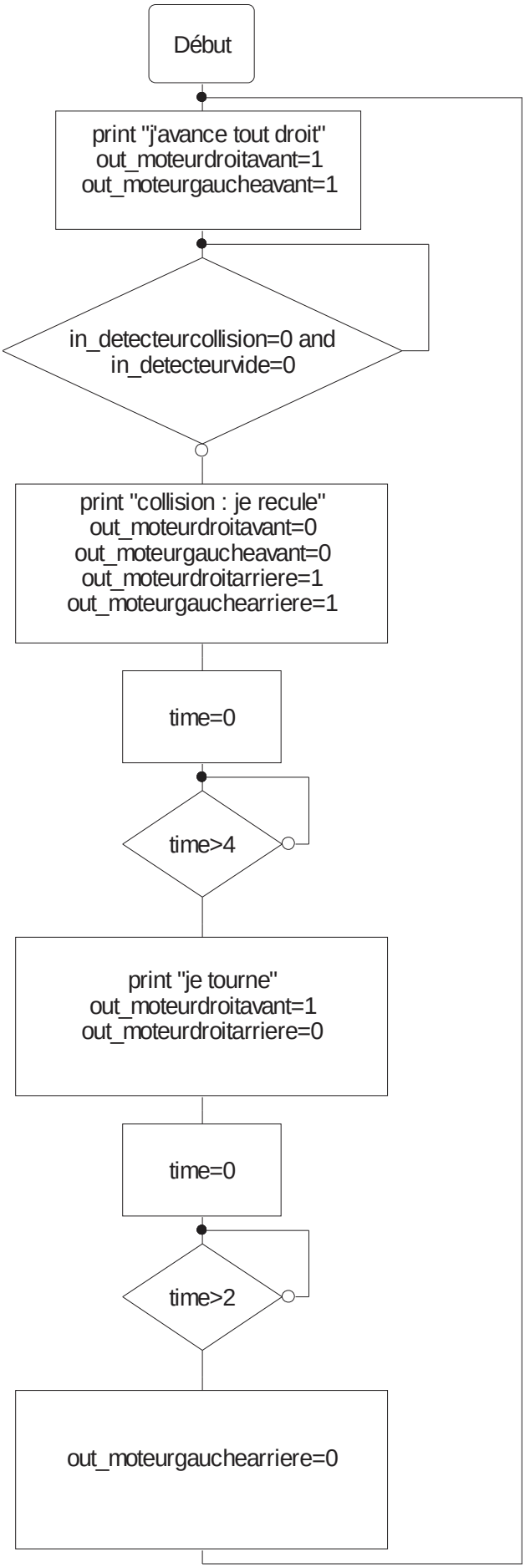


L'état des variables ainsi qu'une fenêtre affichant les sorties des commandes « print » utilisées dans le programme sont affichées. Le bouton « Pause » permet de mettre le programme en pause ou de reprendre son exécution. Le bouton « Step » permet de faire avancer le programme d'un pas.

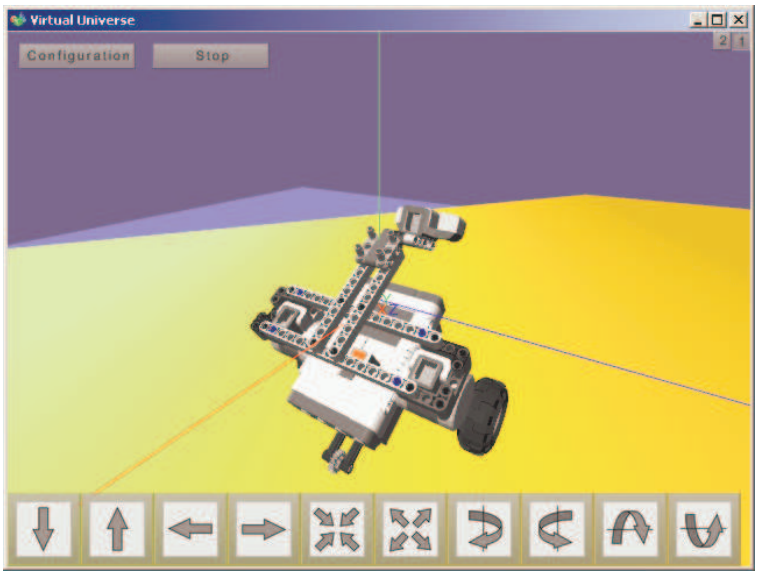
Exemples d'organigrammes

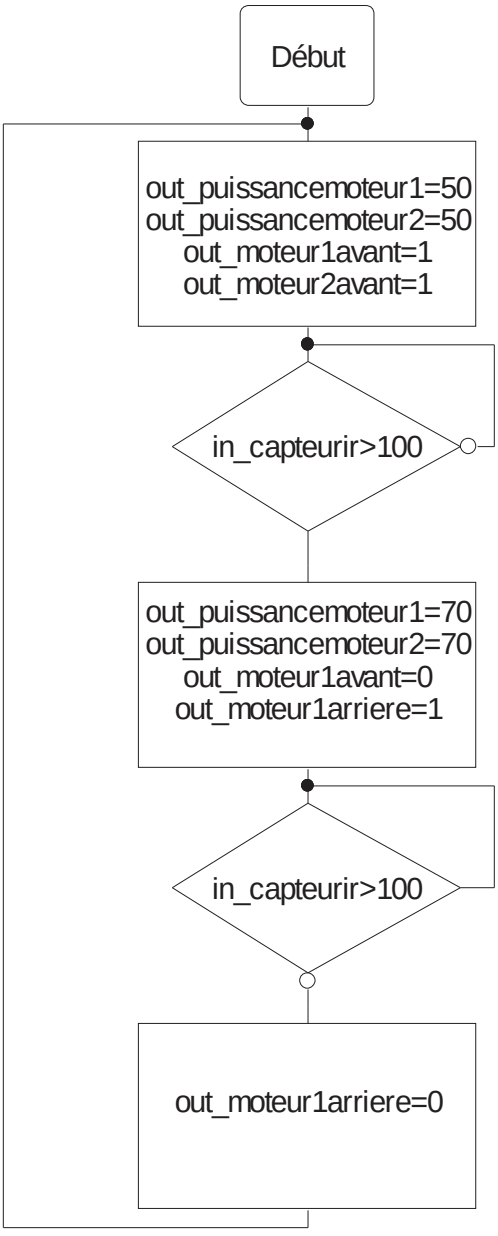
Le sous-répertoire « Exemples » du répertoire d'installation de Virtual Universe contient des exemples Virtual Universe pilotables par orgranigramme ainsi que les organigrammes correspondants (fichiers .FLC).

Robot aspirateur

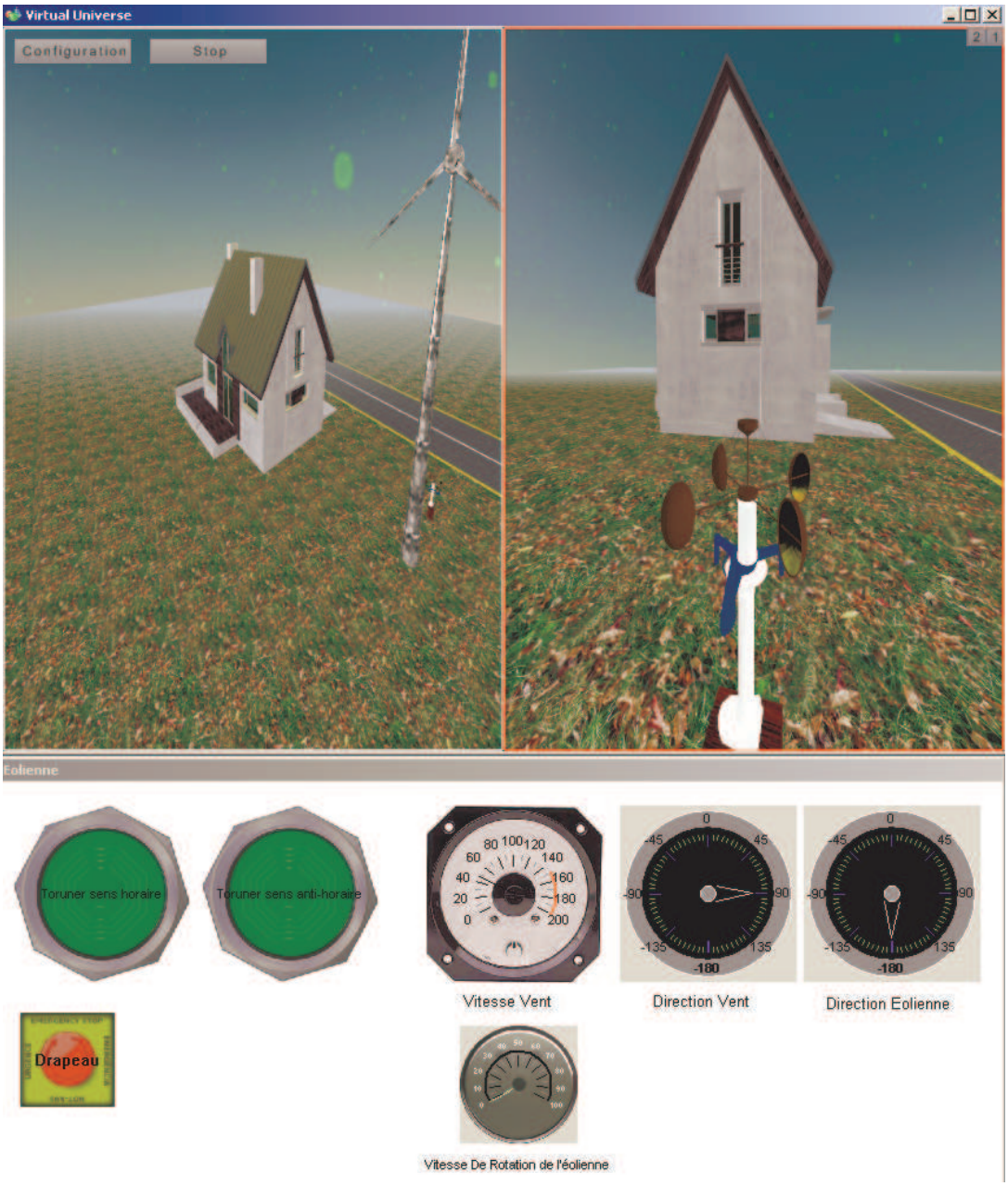

Début
print "j'avance tout droit"
out_moteurdroitavant=1
out_moteurgaucheavant=1
in_detecteurcollision=0 and
in_detecteurvide=0
print "collision : je recule"
out_moteurdroitavant=0
out_moteurgaucheavant=0
out_moteurdroitarriere=1
out_moteurgauchearriere=1
time=0
time>4
print "je tourne"
out_moteurdroitavant=1
out_moteurdroitarriere=0
time=0
time>2
out_moteurgauchearriere=0

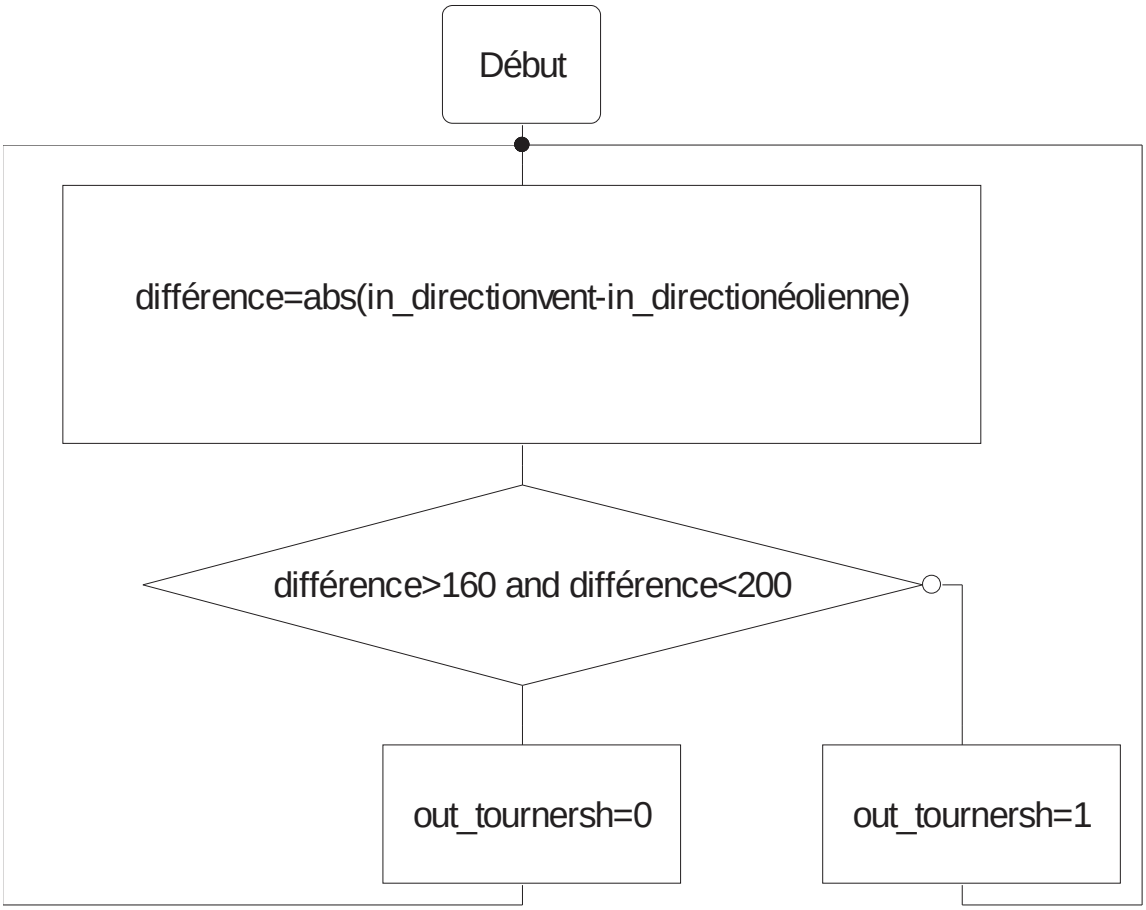
Robot NXT

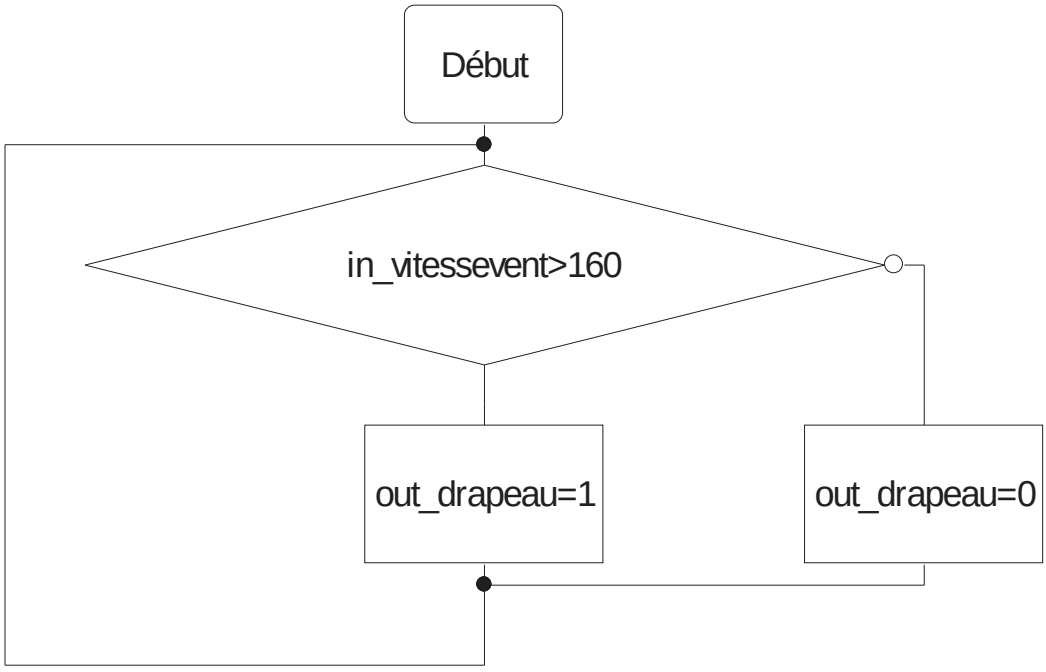

Virtual Universe
Configuration
Stop


Début
out_puissancemoteur1=50
out_puissancemoteur2=50
out_moteur1avant=1
out_moteur2avant=1
in_capteurir>100
out_puissancemoteur1=70
out_puissancemoteur2=70
out_moteur1avant=0
out_moteur1arriere=1
in_capteurir>100
out_moteur1arriere=0

Eolienne


Virtual Universe
Configuration
Stop
2 | 1
Eolienne
Toruner sens horaire
Toruner sens anti-horaire
Vitesse Vent
Direction Vent
Direction Eolienne
Drapeau
Vitesse De Rotation de l'éolienne


Début
différence=abs(in_directionvent-in_directionéolienne)
différence>160 and différence<200
out_tournersh=0
out_tournersh=1


Début
in_vitessevent>160
out_drapeau=1
out_drapeau=0