ControlBuild

... by TH-Software

Global Automation Power

Fonctions de manipulation de produits

Généralités

Editeurs terminaux

Développement d'applications

Outils de tests et d'animation

Documentation

Eléments de langage

Modules annexes

A.1

Sommaire détaillé

1 Description

1.1 La référence à un modèle

Pour obtenir le modèle d'un comportement de nom ***comp*** se trouvant dans l'application ***applis***, il faut inclure dans l'onglet d'*entête* :

- Pour le code C :
  #include ‘’applis/exec/comp.h’’
- Pour le langage ST:
  include ‘applis/exec/box.h’ ;

et utiliser l'identificateur **applis__comp** dans l’onglet *code*.

1.2 La fonction *prodNew*

La fonction ***prodNew*** crée une nouvelle instance de produit ayant pour modèle model, pour positions **apx**, **apy** et **apz** et pour dimensions **adx**, **ady** et **adz**.

- La fonction ***prodNew*** retourne un entier positif ou nul si la création a réussi : cet entier est un identifiant de l'instance de produit créé.
- La fonction ***prodNew*** retourne -1 en cas d'échec.

Cette fonction peut être utilisée dans le code des produits ou des acteurs.

Exemple d'utilisation :

Pour créer, dans l’application «toto» un produit «box» de dimension 1*1*1, positionné à l’origine du repère absolu :

prodNew(toto__box, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0);

Remarque :

- **apx**, **apy**, et **apz** représentent les plus petites coordonnées de l'objet.
- **adx**, **ady** et **adz** ne doivent pas être négatives.

1.3 La fonction *prodKill*

La fonction ***prodKill*** détruit une instance de produit identifiée **i**.
La fonction ***prodKill*** retourne :

- la valeur de **numeroProduit** si la destruction est réussie,
- -1 en cas d'échec.

Cette fonction peut être utilisée dans le code des produits ou des acteurs.

1.4 La fonction *prodGetStatus*

La fonction ***prodGetStatus*** retourne l'état du produit ou de l’acteur identifié par i.

Cette fonction peut être utilisée dans le code des produits ou des acteurs.

Les valeurs possibles de l'état sont :

- -1 -> i est un identifiant incorrect,
- 0 -> i est un identifiant d'acteur,
- 1 -> i est un identifiant de produit créé/actif,
- 2 -> i est un identifiant de produit détruit/inactif

1.5 Le mot clé *myself*

Le mot clé ***myself*** contient un identificateur de l'instance de produit courante.

Ce mot clé peut être utilisé, comme argument de fonction, dans le code des produits ou des acteurs.

1.6 Le mot clé *other*

Le mot clé ***other*** contient un identificateur de l'instance d'acteur ou de produit en interaction avec le produit courant. La variable ***other*** n'a de signification ni dans le prélude et ni dans le postlude de l'éditeur de produits.

Ce mot clé peut être utilisé seulement en argument des fonctions d'interaction des produits.

1.7 La fonction *otherVar*

otherVar est une fonction d'accès aux variables de ***other***.
otherVar(nom) désigne la variable ***nom*** de l'instance de comportement en cours d'interaction avec le produit. Cette variable peut être consultée ou affectée.
ov(nom) est un synonyme abrégé de ***otherVar(nom)***.

Cette fonction peut être utilisée seulement dans le code d'interaction des produits.

Attention :

Les variables d'entrée et de sortie du produit ou acteur en interaction ne sont pas accessibles avec cette fonction.

Exemple :

```
otherVar(vitesse) = otherVar(vitesse) + 1;
```

1.8 La fonction *var*

var est une fonction d'accès aux variables de l'acteur ou du produit identifié par **i**.
var(i, applis__comp__nom) désigne la variable ***nom*** de l'instance de comportement ***comp*** se trouvant dans l'application ***applis*** et identifié par **i**. Cette variable peut être consultée ou affectée.

Cette fonction peut être utilisée dans le code des produits et des acteurs.

Pour utiliser cette fonction, il faut inclure dans l'onglet d'*entête*:

- Pour le code C,:
  #include ‘’applis/exec/comp.h’’
- Pour le langage ST, dans le code :
  include ‘applis/exec/box.h’ ;

et utiliser l'identificateur **applis__comp** dans l'onglet *code*.

La fonction ***otherVar(nom)*** est équivalente à ***var(other, applis__comp__nom)*** mais ***var*** est utilisable dans le code des acteurs et des produits à la différence de ***otherVar*** qui n'est valable que dans le code produits.

Attention :

Les variables d'entrée et de sortie du produit ou acteur en interaction ne sont pas accessibles avec cette fonction.

Exemple:

```
var(i, applis__box__vx);
```

1.9 Autres variables

NbActeurs contient le nombre total d'acteurs.
NbProduits contient le nombre total de produits.
NbActeursEtProduits contient ***NbActeurs + NbProduits***.

Ces variables ne doivent pas être réaffectées.
Les identificateurs valides d'acteurs sont compris dans l'intervalle : [0, NbActeurs[.
Les identificateurs valides de produits sont compris dans l'intervalle : [NbActeurs, NbActeursEtProduits[.

Remarque :
Ces variables ne sont accessibles que dans l'éditeur de langage C.

2 Fonctions de translation

2.1 La fonction *moveTo*

La fonction ***moveTo*** déplace, dans le repère absolu de l'application, l'instance (acteur ou produit) identifiée par **i** vers un point de coordonnées : (**px**, **py**, **pz**).

Cette fonction peut être utilisée dans le code des produits et des acteurs.

La fonction ***moveTo*** retourne :

- la valeur de **numeroProduit** si le déplacement a réussi.
- -1 en cas d'échec du déplacement.

Exemple d'utilisation :

Si on veut placer une instance id, dans le plan xy à la cote 2 suivant x et -4 suivant y :

moveTo(id, 2.0, -4.0, 0.0) ;

id peut être : myself, l'identifiant d'un comportement quelconque déclaré acteur ou produit, ou other si cette instruction est utilisée dans le code d'un produit.

Remarque :

Les coordonnées **px**, **py** et **pz** représentent la position du coin de plus petites coordonnées du solide selon respectivement les axes x, y et z.

2.2 Les fonctions *moveXTo*, *moveYTo*, *moveZTo*

La fonction ***moveXTo*** (respectivement ***moveYTo*** et ***moveZTo***) correspond à la fonction moveTo mais avec un déplacement uniquement suivant l'axe x (respectivement y et z) vers un point de coordonnée : **px** (respectivement **py** et **pz**).

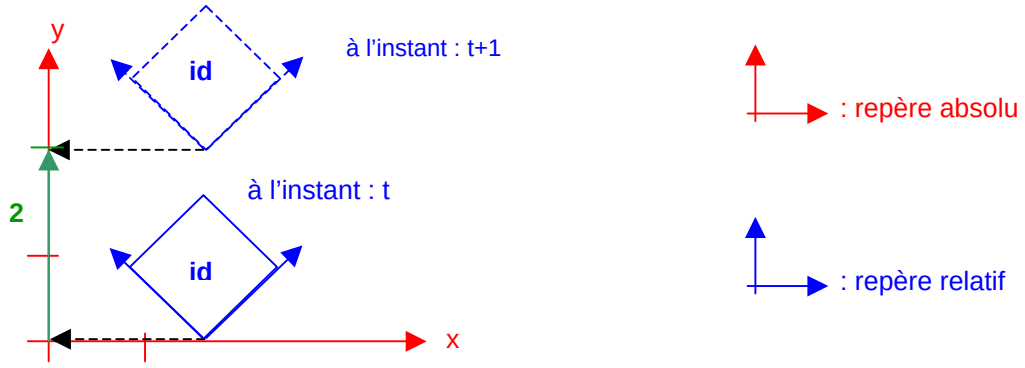
Ces fonctions peuvent être utilisées dans le code des produits et des acteurs.

Exemple d'utilisation :

Pour déplacer une instance id, vers une position y = 2 suivant l'axe y.

moveYTo(id, 2.0) ;

Schéma :



2.3 La fonction *moveBy*

La fonction ***moveBy*** déplace, dans le repère absolu de l'application, l'instance (acteur ou produit) identifiée par **i** d'un vecteur incrément : (**dx**, **dy**, **dz**).
Cette fonction peut être utilisée dans le code des produits et des acteurs.

La fonction ***moveBy*** retourne :

- la valeur de **numeroProduit** si le déplacement a réussi.
- -1 en cas d'échec du déplacement.

Exemple d'utilisation :

Pour déplacer une instance id, dans le plan xy de 2 unités de longueur suivant x et de -4 suivant y.

moveBy(id, 2.0, -4.0, 0.0) ;

id peut être : *myself*, l'identifiant d'un comportement déclaré acteur ou produit, ou *other* si cette instruction est utilisée dans le code d'un produit.

2.4 Les fonctions *moveXBy*, *moveYBy*, *moveZBy*

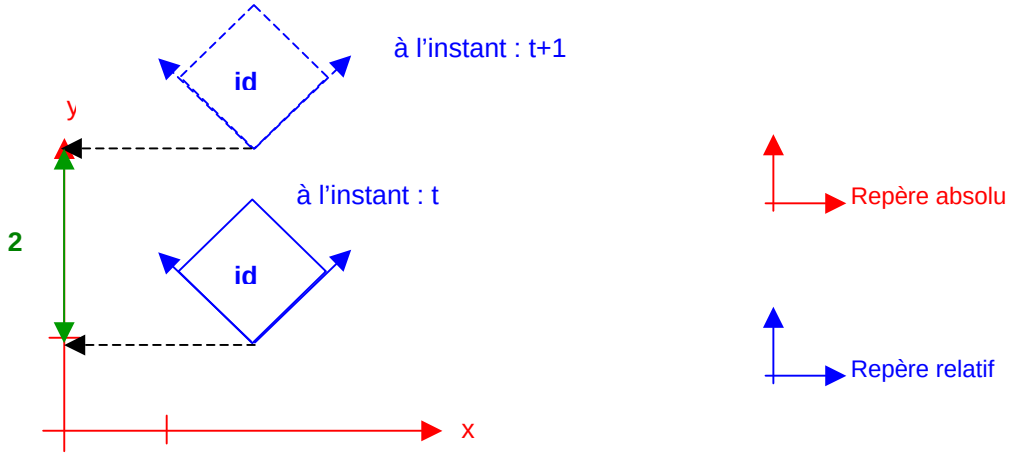
La fonction ***moveXBy*** (respectivement ***moveYBy*** et ***moveZBy***) correspond à la fonction moveBy mais avec un déplacement uniquement suivant l'axe x (respectivement y et z) d'un incrément : **dx** (respectivement **dy** et **dz**).
Ces fonctions peuvent être utilisées dans le code des produits et des acteurs.

Exemple d'utilisation :

Pour déplacer une instance id, d'un incrément dy = 2.

moveYBy(id, 2.0) ;

Schéma :



2.5 La fonction *moveRelativeBy*

La fonction ***moveRelativeBy*** déplace l'instance (acteur ou produit) identifiée par **i** dans son repère relatif, d'un vecteur incrément : (**dx**, **dy**, **dz**).
Cette fonction peut être utilisée dans le code des produits et des acteurs.

La fonction ***moveRelativeBy*** retourne :

- la valeur de **numeroProduit** si le déplacement a réussi.
- -1 en cas d'échec du déplacement.

Exemple d'utilisation :

Pour déplacer une instance id, d'un vecteur : (2.5, -4.3, 10.0) mais dans son repère relatif

moveRelativeBy(id, 2.5, -4.3, 10.0) ;

id peut être : *myself*, l'identifiant d'un comportement déclaré acteur ou produit, ou *other* si cette instruction est utilisée dans le code d'un produit.

2.6 Les fonctions *moveRelativeXBy*, *moveRelativeYBy*, *moveRelativeZBy*

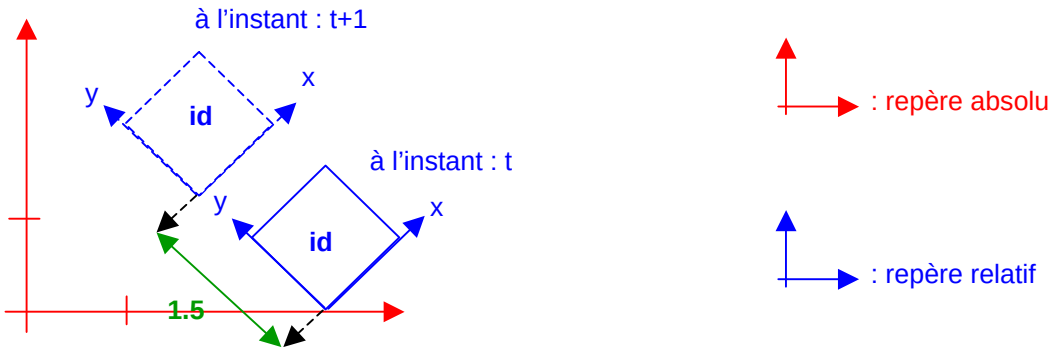
La fonction ***moveRelativeXBy*** (respectivement ***moveRelativeYBy*** et ***moveRelativeZBy***) correspond à la fonction moveRelativeBy mais avec un déplacement uniquement suivant l'axe x (respectivement y et z) d'un incrément : **dx** (respectivement **dy** et **dz**).
Ces fonctions peuvent être utilisées dans le code des produits et des acteurs.

Exemple d'utilisation :

Pour déplacer une instance id, relativement à son repère relatif, d'un incrément dy = 1.5.

moveRelativeYBy(id, 1.5) ;

Schéma :



2.7 La fonction *moveRelativeAnotherBy*

La fonction ***moveRelativeAnotherBy*** déplace, d'un vecteur incrément : (**dx**, **dy**, **dz**), l'instance (acteur ou produit) identifiée par **i** relativement au repère de l'instance identifiée par **j**.
Cette fonction peut être utilisée dans le code des produits et des acteurs.

La fonction ***moveRelativeAnotherBy*** retourne :

- la valeur de **numeroProduit** de l'instance déplacée si le déplacement a réussi.
- -1 en cas d'échec du déplacement.

Exemple d'utilisation :

Pour déplacer une instance «caisse» d'un vecteur : (2.5, 0.0, 0.0) mais relativement au repère d'un autre modèle «convoyeur» afin de modéliser son convoyage

moveRelativeAnotherBy(caisse, convoyeur, 2.5, 0.0, 0.0) ;

caisse et *convoyeur* peuvent également être : *myself*, ou *other* si cette instruction est utilisée dans le code d'un produit.

2.8 Les fonctions *moveRelativeAnotherXBy*, *moveRelativeAnotherYBy*, *moveRelativeAnotherZBy*

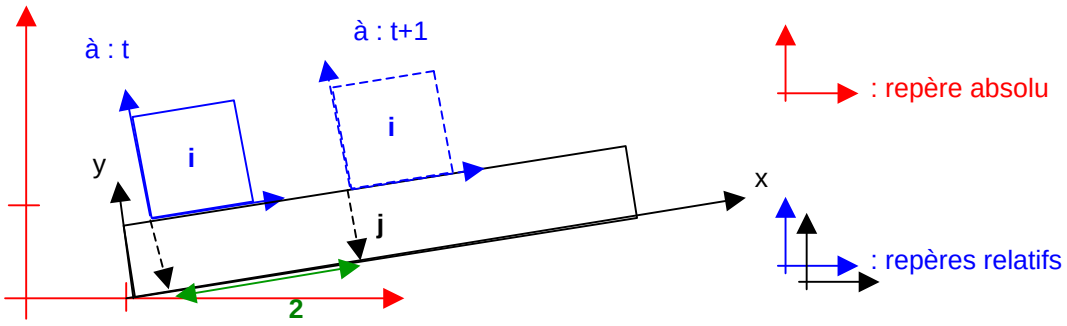
La fonction ***moveRelativeAnotherXBy*** (respectivement ***moveRelativeYBy*** et ***moveRelativeZBy***) correspond à la fonction moveRelativeAnotherBy mais avec un déplacement uniquement suivant l'axe x (respectivement y et z) d'un incrément : **dx** (respectivement **dy** et **dz**).
Ces fonctions peuvent être utilisées dans le code des produits et des acteurs.

Exemple d'utilisation :

Pour déplacer une instance i, relativement au repère d'une instance j, d'un incrément dx = 2.

moveRelativeAnotherXBy(i, j, 2) ;

Schéma :



3 Fonctions de rotation

3.1 La fonction *rotateBy*

La fonction ***rotateBy*** fait pivoter, par rapport au repère absolu de l'application, l'instance (acteur ou produit) identifiée par **i,** d'un incrément d'angle : **rx** autour de l'axe x, **ry** autour de l'axe y, **rz** autour de l'axe z et par rapport à un centre de rotation de coordonnées : (**cx**, **cy**, **cz**).
Cette fonction peut être utilisée dans le code des produits et des acteurs.

La fonction ***rotateBy*** retourne :

- la valeur de **numeroProduit** si la rotation a réussi.
- -1 en cas d'échec de la rotation.

Exemple d'utilisation :

Pour faire pivoter une instance id, d'un angle π autour de x, π/4 autour d'y et 0 autour de z, et par rapport à un centre de rotation de coordonnées : (1.05 ; 4.32, -20.5)

rotateBy(id, 3.14, 0.78, 0.0, 1.05, 4.32, -20.5) ;

id peut être : myself, l'identifiant d'un comportement déclaré acteur ou produit, ou other si cette instruction est utilisée dans le code d'un produit.

Remarque :

Les valeurs des angles de rotation **rx**, **ry** et **rz** doivent être saisies en radian.

3.2 Les fonctions *rotateXBy*, *rotateYBy*, *rotateZBy*

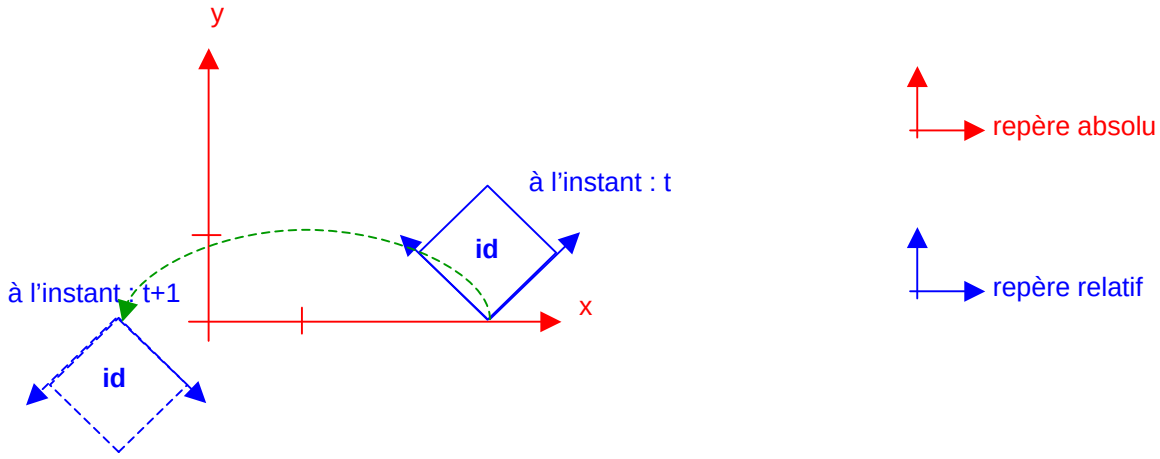
La fonction ***rotateXBy*** (respectivement ***rotateYBy*** et ***rotateZBy***) correspond à la fonction rotateBy mais avec une rotation uniquement autour de l'axe x (respectivement y et z) d'un incrément d'angle : **rx** (respectivement **ry** et **rz**) et par rapport à un centre de rotation de coordonnées : (**cx**, **cy**, **cz**).
Ces fonctions peuvent être utilisées dans le code des produits et des acteurs.

Exemple d'utilisation :

Pour faire pivoter une instance id, d'un angle π autour de z par rapport à un centre de coordonnées : (1.0 ; 0.0, -20.5)

rotateZBy(id, 180, 1.0, 0.0, -20.5) ;

Schéma :



3.3 La fonction *rotateCenterBy*

La fonction ***rotateCenterBy*** fait pivoter, par rapport au repère absolu de l'application, l'instance (acteur ou produit) identifiée par **i,** d'un incrément d'angle : **rx** autour de l'axe x, **ry** autour de l'axe y, **rz** autour de l'axe z et par rapport à son centre.
Cette fonction peut être utilisée dans le code des produits et des acteurs.

La fonction ***rotateCenterBy*** retourne :

- la valeur de **numeroProduit** si la rotation a réussi.
- -1 en cas d'échec de la rotation.

Exemple d'utilisation :

Pour faire pivoter une instance id, par rapport à son centre, d'un angle π autour de x, π/4 autour d'y et 0 autour de z.

rotateCenterBy(id, 3.14, 0.78, 0.0) ;

id peut être : myself, l'identifiant d'un comportement quelconque déclaré acteur ou produit ou other si cette instruction est utilisée dans le code d'un produit.

Remarque :

Les valeurs des angles de rotation **rx**, **ry** et **rz** doivent être saisies en radian.

3.4 Les fonctions *rotateCenterXBy*, *rotateCenterYBy*, *rotateCenterZBy*

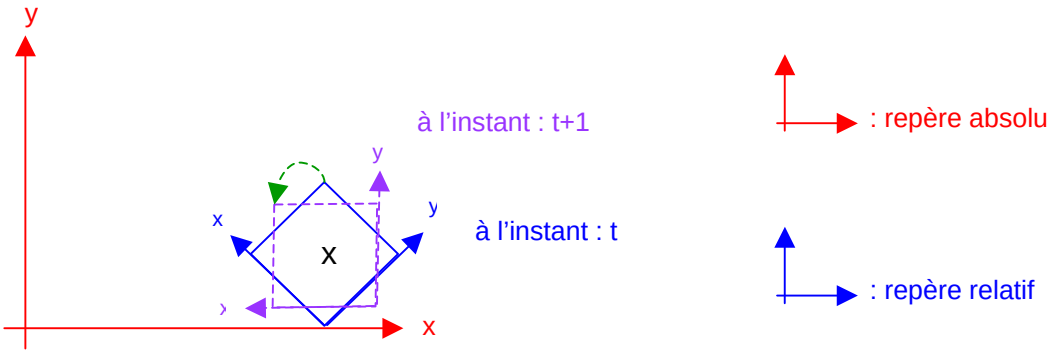
La fonction ***rotateCenterXBy*** (respectivement ***rotateCenterYBy*** et ***rotateCenterZBy***) correspond à la fonction rotateCenterBy mais avec une rotation uniquement autour de l'axe x (respectivement y et z) d'un incrément d'angle : **rx** (respectivement **ry** et **rz)**.
Ces fonctions peuvent être utilisées dans le code des produits et des acteurs.

Exemple d'utilisation :

Pour faire pivoter une instance id, par rapport à son centre d'un angle π autour de z

rotateCenterZBy(id, 180);

Schéma :



3.5 La fonction *rotateTo*

La fonction ***rotateTo*** fait pivoter, par rapport au repère absolu de l'application, l'instance (acteur ou produit) identifiée par **i,** vers un angle : **rx** autour de l'axe x, **ry** autour de l'axe y, **rz** autour de l'axe z et par rapport à un centre de rotation de coordonnées : (**cx**, **cy**, **cz**).
Cette fonction peut être utilisée dans le code des produits et des acteurs.
La fonction ***rotateTo*** retourne :

- la valeur de **numeroProduit** si la rotation a réussi.
- -1 en cas d'échec de la rotation.

Exemple d'utilisation :

Pour faire pivoter une instance id, vers un angle π autour de x, π/4 autour d'y et 0 autour de z, par rapport à un centre de coordonnées : (1.05 ; 4.32, -20.5)

rotateTo(id, 3.14, 0.78, 0.0, 1.05, 4.32, -20.5) ;

id peut être : myself, l'identifiant d'un comportement quelconque déclaré acteur ou produit ou other si cette instruction est utilisée dans le code d'un produit.

Remarque :

Les valeurs des angles de rotation **rx**, **ry** et **rz** doivent être saisies en radian.

3.6 Les fonctions `rotateXTo, rotateYTo, rotateZTo`

La fonction ***rotateXTo*** (respectivement ***rotateYTo*** et ***rotateZTo***) correspond à la fonction ***rotateTo*** mais avec une rotation uniquement autour de l'axe X (respectivement Y et Z) d'un d'angle : **rx** (respectivement **ry** et **rz)**.
Ces fonctions peuvent être utilisées dans le code des produits et des acteurs.

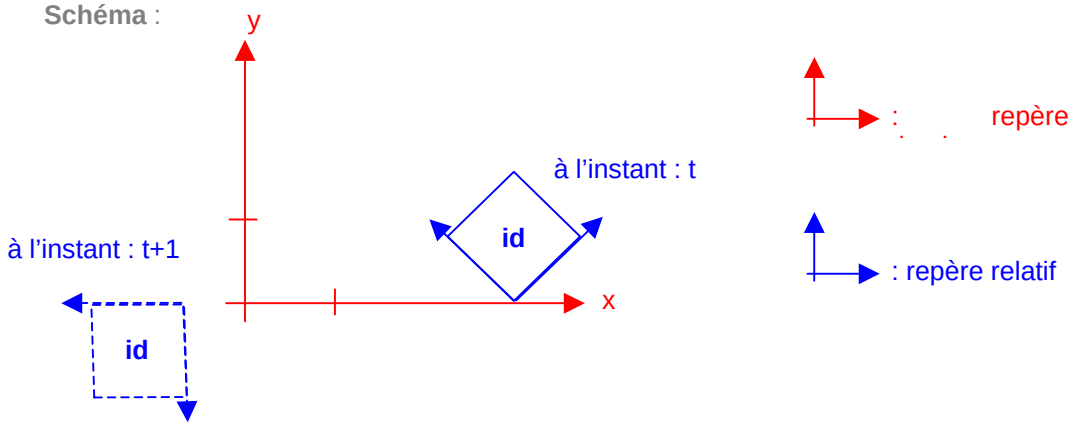
Exemple:

Pour faire pivoter une instance id vers un angle π autour de z par rapport à un centre de coordonnées (1.0, 0.0, -20.5) :

rotateZTo(id, 180.0, 1.0, 0.0, -20.5) ;

id peut être : myself, l'identifiant d'un comportement quelconque déclaré acteur ou produit ou other si cette instruction est utilisée dans le code d'un produit.

Schéma :



3.7 La fonction *rotateCenterTo*

La fonction ***rotateCenterTo*** correspond à la fonction rotateTo mais avec une rotation par rapport au centre de l'instance **i**, vers un angle : **rx** autour de l'axe x, **ry** autour de l'axe y, **rz** autour de l'axe z.
Cette fonction peut être utilisée dans le code des produits et des acteurs.

La fonction ***rotateCenterTo*** retourne :

- la valeur de **numeroProduit** si la rotation a réussi.
- -1 en cas d'échec de la rotation.

Exemple d'utilisation :

Pour faire pivoter une instance id, par rapport à son centre vers un angle π autour de x, π/4 autour d'y et 0 autour de z

rotateCenterTo(id, 3.14, 0.78, 0.0) ;

3.8 Les fonctions *rotateCenterXTo*, *rotateCenterYTo*, *rotateCenterZTo*

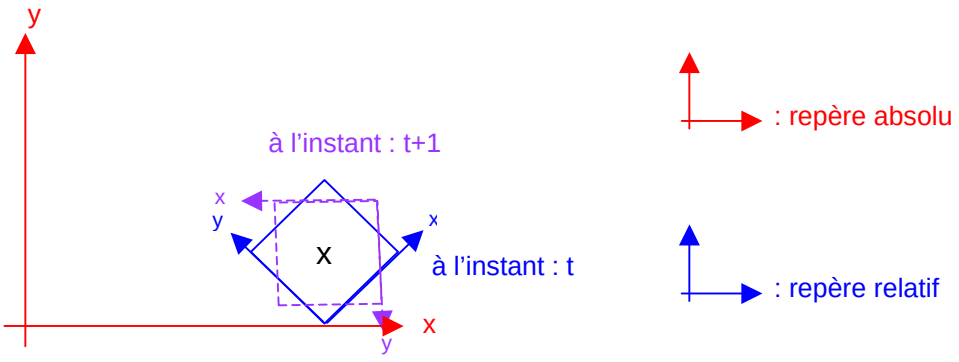
La fonction ***rotateCenterXTo*** (respectivement ***rotateCenterYTo*** et ***rotateCenterZTo***) correspond à la fonction ***rotateCenterTo*** mais avec une rotation uniquement autour de l'axe X (respectivement Y et Z) d'un d'angle : **rx** (respectivement **ry** et **rz)**.
Ces fonctions peuvent être utilisées dans le code des produits et des acteurs.

Exemple:

Pour faire pivoter une instance id vers un angle π autour de z par rapport à son centre :

rotateZTo(id, 180.0) ;

Schéma :



4 Fonction de redimentionnement

4.1 La fonction *resizeTo*

La fonction ***resizeTo*** permet de redimensionner l'instance (acteur ou produit) identifiée par **i,** afin d'obtenir un objet de nouvelle taille : **adx*ady*adz**.
Cette fonction peut être utilisée dans le code des produits et des acteurs.

La fonction ***resizeTo*** retourne :

- la valeur de **numeroProduit** si le redimensionnement a réussi.
- -1 en cas d'échec du redimensionnement.

Exemple d'utilisation :

Pour redimensionner une instance id, de façon à obtenir un objet de taille : 10.5 * 55.25 * 20.0

resizeTo(id, 10.5, 55.25, 20.0) ;

id peut être : myself, l'identifiant d'un comportement quelconque déclaré acteur ou produit, ou other si cette instruction est utilisée dans le code d'un produit.

4.2 Les fonctions *resizeXTo*, *resizeYTo*, *resizeZTo*

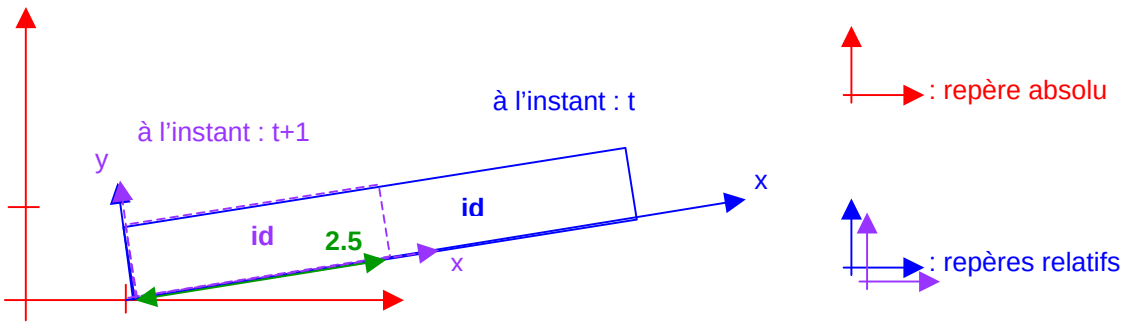
La fonction ***resizeXTo*** (respectivement ***resizeYTo*** et ***resizeZTo***) correspond à la fonction resizeTo mais avec un redimensionnement uniquement suivant la dimension x (respectivement y et z) vers une valeur : **adx** (respectivement **ady** et **adz)**.
Ces fonctions peuvent être utilisées dans le code des produits et des acteurs.

Exemple d'utilisation :

Pour obtenir une instance id, de longueur 2.5 suivant sa dimension x à la place de sa valeur actuelle

resizeXTo(id, 2.5) ;

Schéma :



4.3 La fonction *resizeBy*

La fonction ***resizeBy*** permet d'augmenter ou de diminuer les dimensions de l'instance (acteur ou produit) identifiée par **i,** d'une quantité **adx** suivant x, **ady** suivant y, **adz** suivant z.
Cette fonction peut être utilisée dans le code des produits et des acteurs.

La fonction ***resizeBy*** retourne :

- la valeur de **numeroProduit** si le redimensionnement a réussi.
- -1 en cas d'échec du redimensionnement.

Exemple d'utilisation :

Pour redimensionner une instance id, de façon à obtenir un objet plus grand de 2 unités suivant x, et plus petit de 3.5 unités suivant y

resizeBy(id, 2.0, -3.5, 0.0) ;

id peut être : myself, l'identifiant d'un comportement quelconque déclaré acteur ou produit ou other si cette instruction est utilisée dans le code d'un produit.

4.4 Les fonctions *resizeXBy*, *resizeYBy*, *resizeZBy*

La fonction ***resizeXBy*** (respectivement ***resizeYBy*** et ***resizeZBy***) correspond à la fonction resizeBy mais avec un redimensionnement uniquement suivant la dimension x (respectivement y et z) d'une quantité : **adx** (respectivement **ady** et **adz)**.
Ces fonctions peuvent être utilisées dans le code des produits et des acteurs.

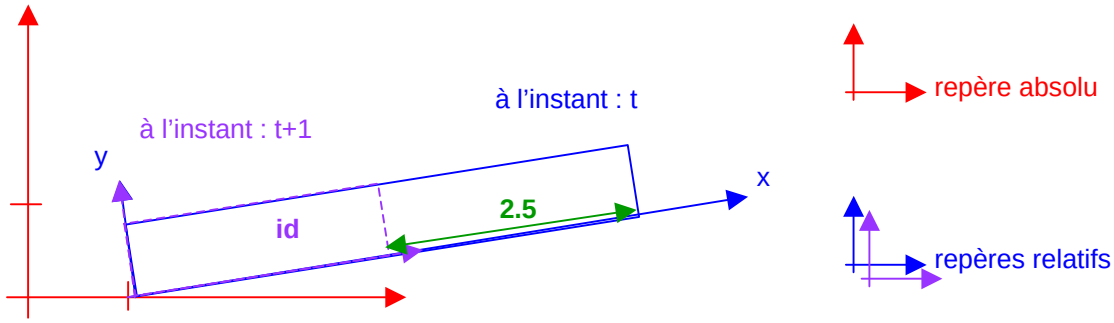
Exemple d'utilisation :

Pour réduire la longueur suivant x d'une instance id, de 2.5 par rapport à sa dimension actuelle

resizeXBy(id, -2.5) ;

id peut être : myself, l'identifiant d'un comportement quelconque déclaré acteur ou produit ou other si cette instruction est utilisée dans le code d'un produit.

Schéma :



5 Fonction de renseignement

5.1 Les fonctions *getPositionX*, *getPositionY*, *getPositionZ*

La fonction ***getPositionX*** (respectivement ***getPositionY***, ***getPositionZ***) renvoie, sous forme d'une valeur de type REAL (type double), la position dans le repère absolu de l'application, suivant l'axe x (respectivement y et z) de l'instance (acteur ou produit) identifiée par **i**.
Ces fonctions peuvent être utilisées dans le code des produits et des acteurs.

Exemple d'utilisation :

Pour connaître la position suivant z de l'instance id

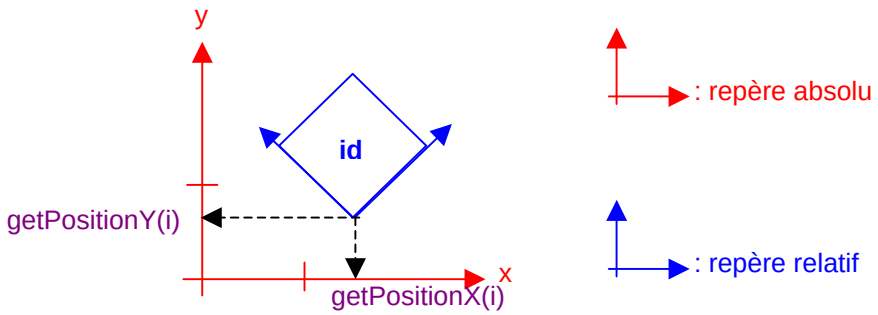
getPositionZ(id) ;

id peut être : myself, l'identifiant d'un comportement déclaré acteur ou produit, ou other si cette instruction est utilisée dans le code d'un produit.

Remarque :

La coordonnée renvoyée par la fonction représente la position du coin de plus petites coordonnées du solide.

Schéma :



5.1.1 Les fonctions *getPositionXInAnother, getPositionYInAnother, getPositionZInAnother*

La fonction ***getPositionXInAnother*** (respectivement ***getPositionYInAnother***, ***getPositionZInAnother***) renvoie, sous forme d'un REAL (type double), la position de l'instance (acteur ou produit) identifiée par **i** suivant l'axe x (respectivement y et z) du repère relatif de l'instance (acteur ou produit) identifiée par **j**.
Ces fonctions peuvent être utilisées dans le code des produits et des acteurs.

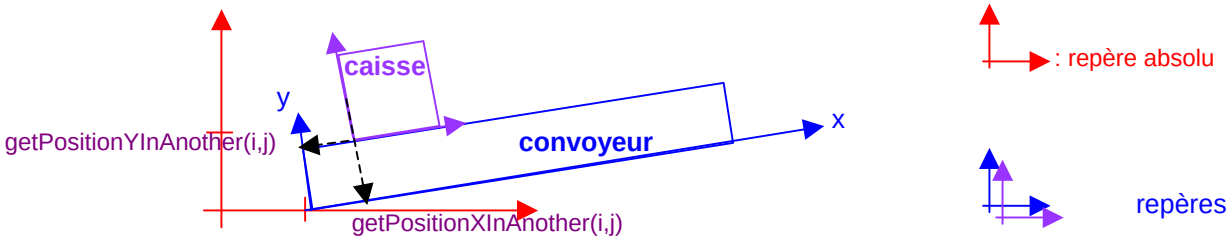
Exemple d'utilisation :

Pour connaître la coordonnée suivant z de l'instance *caisse* dans le repère de l'instance *convoyeur* :
getPositionZInAnother(caisse, convoyeur) ;
caisse et ***convoyeur*** peuvent également être : *myself*, ou l'identifiant renvoyé par la fonction *prodNew* lors de la création d'un produit, ou *other* si cette instruction est utilisée dans le code d'interaction d'un produit.

Remarque :

La coordonnée renvoyée par la fonction représente la position du coin de plus petites coordonnées du solide.

Schéma :



5.2 Les fonctions *getProjectionX*, *getProjectionY*, *getProjectionZ*

La fonction ***getProjectionX*** (respectivement ***getProjectionY***, ***getProjectionZ***) renvoie, sous forme d'une valeur de type REAL (type double), la projection sur l'axe x (respectivement y et z) du repère absolu de l'application, de l'arête relative à ce même axe mais dans le repère relatif de l'instance (acteur ou produit) identifiée par **i**.
Ces fonctions peuvent être utilisées dans le code des produits et des acteurs.

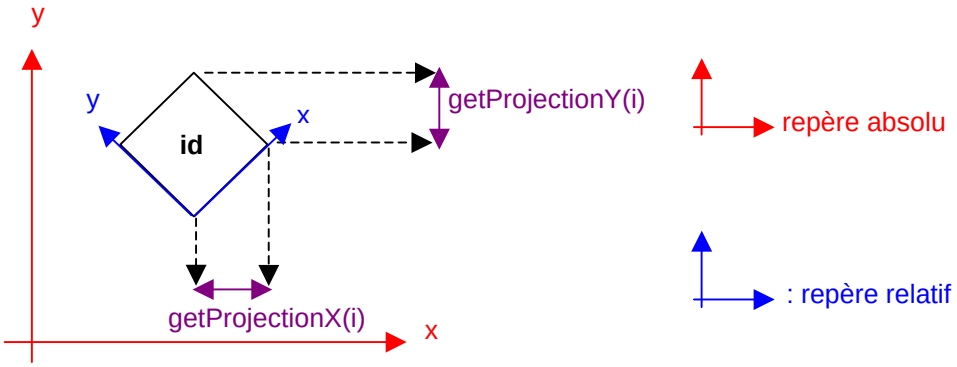
Exemple d'utilisation :

Pour connaître la projection sur z de la hauteur (dz) de l'instance id

getProjectionZ(id) ;

id peut être : myself, l'identifiant d'un comportement quelconque déclaré acteur ou produit, ou other si cette instruction est utilisée dans le code d'un produit.

Schéma :



5.3 Les fonctions *getAngleX*, *getAngleY*, *getAngleZ*

La fonction ***getAngleX*** (respectivement ***getAngleY***, ***getAngleZ***) renvoie, sous forme d'une valeur DINT (entier long), la position angulaire par rapport l'axe x (respectivement y et z), dans le repère absolu de l'application, de l'instance (acteur ou produit) identifiée par **i**.
Ces fonctions peuvent être utilisées dans le code des produits et des acteurs.

Exemple d'utilisation :

Pour connaître l'orientation par rapport à l'axe z de l'instance id

getAngleZ(id) ;

id peut être : myself, l'identifiant d'un comportement quelconque déclaré acteur ou produit, ou other si cette instruction est utilisée dans le code d'un produit.

Remarque :

La valeur d'angle renvoyée est en radians.

5.4 Les fonctions *getCenterX*, *getCenterY*, *getCenterZ*

La fonction ***getCenterX*** (respectivement ***getCenterY***, ***getCenterZ***) renvoie, sous forme d'un double, la position sur l'axe x (respectivement y et z), dans le repère absolu de l'application, du centre de l'instance (acteur ou produit) identifiée par **i**.
Ces fonctions peuvent être utilisées dans le code des produits et des acteurs.

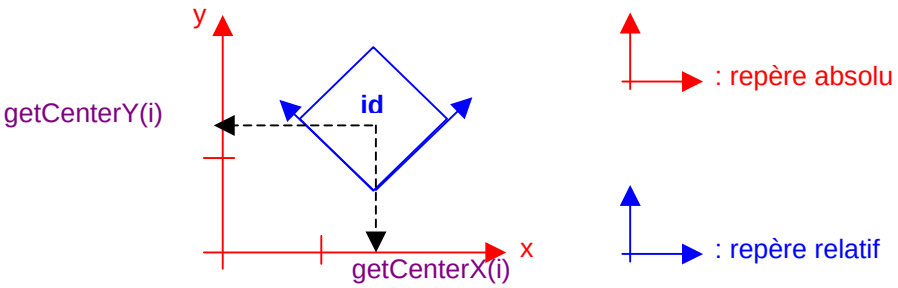
Exemple d'utilisation :

Pour connaître la coordonnée suivant z du centre de l'instance id

getCenterZ(id) ;

id peut être : myself, l'identifiant d'un comportement quelconque déclaré acteur ou produit, ou other si cette instruction est utilisée dans le code d'un produit.

Schéma :



5.5 Les fonctions *getCenterXInAnother*, *getCenterYInAnother*, *getCenterZInAnother*

La fonction ***getCenterXInAnother*** (respectivement ***getCenterYInAnother***, ***getCenterZInAnother***) renvoie la position du centre de l'instance (acteur ou produit) identifiée par **i** suivant l'axe x (respectivement y et z) du repère relatif de l'instance (acteur ou produit) identifiée par **j**. La valeur est renvoyée sous forme d'un double (variable de type *REAL* dans ControlBuild).
Ces fonctions peuvent être utilisées dans le code des produits et des acteurs.

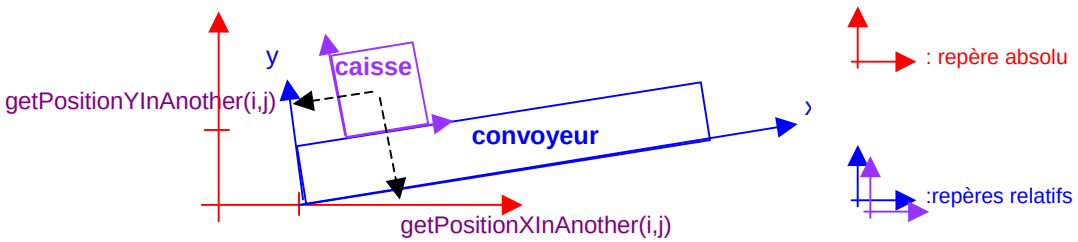
Exemple d'utilisation :

Pour connaître la coordonnée suivant z du centre de l'instance *caisse* dans le repère de l'instance *convoyeur* :

getCenterZInAnother(caisse, convoyeur) ;

caisse et ***convoyeur*** peuvent également être : *myself*, ou l'identifiant renvoyé par la fonction *prodNew* lors de la création d'un produit, ou *other* si cette instruction est utilisée dans le code d'interaction d'un produit.

Schéma :



5.6 Les fonctions *getSizeX*, *getSizeY*, *getSizeZ*

La fonction ***getSizeX*** (respectivement ***getSizeY***, ***getSizeZ***) renvoie, sous forme d'un double, la dimension suivant l'axe des x (respectivement y et z) de l'instance (acteur ou produit) identifiée par **i**.
Ces fonctions peuvent être utilisées dans le code des produits et des acteurs.

Exemple d'utilisation :

Si on veut connaître la hauteur de l'instance id

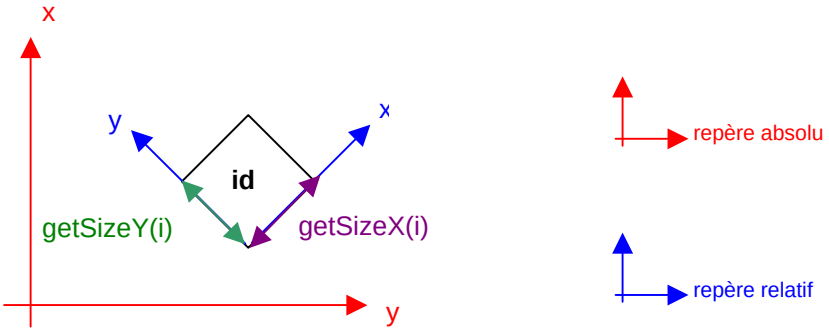
getSizeZ(id) ;

id peut être : myself, l'identifiant d'un comportement quelconque déclaré acteur ou produit, ou other si cette instruction est utilisée dans le code d'un produit.

Remarque :

Ces fonctions correspondent aux instructions getDX, getDY, getDZ de la version 4.50 de Spex. Ces dernières sont cependant toujours reconnues dans le code des acteurs et des produits.

Schéma :



5.7 Les fonctions *isMyCenterIncludedInXYOfOther*, *isMyCenterIncludedInYZOfOther*, *isMyCenterIncludedInZXOfOther*

La fonction ***isMyCenterIncludedInXYOfOther*** (respectivement ***isMyCenterIncludedInYZOfOther***, ***isMyCenterIncludedInXZOfOther***) renvoie un booléen égal à :

- 1 si le centre de l'instance (acteur ou produit) identifiée par i est inclus dans le plan XY (respectivement YZ et XZ) de l'instance (acteur ou produit) identifiée par j ;
- 0 sinon.

Ces fonctions peuvent être utilisées dans le code des produits et des acteurs.

Exemple d'utilisation :

Pour savoir si le centre de l'instance *caisse* est inclus dans le plan XY de *convoyeur* donc au dessus du convoyeur :

isMyCenterIncludedInXYOfOther (caisse, convoyeur) ;

caisse et ***convoyeur*** peuvent également être : *myself*, ou l'identifiant renvoyé par la fonction *prodNew* lors de la création d'un produit, ou *other* si cette instruction est utilisée dans le code d'interaction d'un produit.

6 Fonctions d'alignement

6.1 Les fonctions : alignTwoObjectsRelativeThirdX, alignTwoObjectsRelativeThirdY, alignTwoObjectsRelativeThirdZ

La fonction ***alignTwoObjectsRelativeThirdX*** (respectivement ***alignTwoObjectsRelativeThirdY***, ***alignTwoObjectsRelativeThirdZ***) permet d'aligner perpendiculairement à l'axe x relatif à une instance (acteur ou produit) identifiées par **k,** les faces les plus proches de deux instances (acteurs ou produits) identifiées par **i** et **j**.
Ces fonctions peuvent être utilisées dans le code des produits et des acteurs.

La fonction ***alignTwoObjectsRelativeThirdX*** retourne :

- la valeur de **numeroProduit** si l'alignement a réussi.
- -1 en cas d'échec de l'alignement.

Exemple d'utilisation :

Sur un convoyeur avec butée, les produits s'entassent de telle façon que les faces en contact (donc les plus proches) des instances «caisse1» et «caisse2» s'alignent. Pour indiquer de quelles faces il s'agit, il suffit de spécifier que ce sont celles perpendiculaires à la direction de convoyage. Dans cet exemple, on prendra l'axe x de l'instance «convoyeur».

alignTwoObjectsRelativeThirdX (caisse1, caisse2, convoyeur) ;

caisse1, *caisse2* et convoyeur peuvent être : *myself*, l'identifiant d'un comportement quelconque déclaré acteur ou produit, ou *other* si cette instruction est utilisée dans le code d'un produit.

Remarque :

Pour que l'alignement se fasse correctement, il faut que les faces des deux instances à aligner soient parallèles entre elles.
La troisième instance **k**, celle indiquant la direction d'alignement des faces, peut être égale à une des instances à aligner **i** ou **j**.