



HAPTION
Atelier relais
ZA – Route de Laval
53210 Soulgé sur Ouette

API VIRTUOSE V3.80

DOCUMENTATION



	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 2

TABLE DES MATIÈRES

1	INTRODUCTION	3
1.1	Qu'est-ce que l'API VIRTUOSE ?	3
1.2	Composition du document	3
2	MANUEL D'INSTALLATION	4
3	MANUEL UTILISATEUR	4
3.1	Contenu de l'API VIRTUOSE	4
3.2	Conventions	5
3.2.1	Programmation	5
3.2.2	Modes d'interaction	5
3.2.3	Définition des repères	5
3.2.4	Unités physiques utilisées	7
3.2.5	Expression des grandeurs géométriques	7
3.3	Mise en œuvre de l'API VIRTUOSE	8
3.3.1	Initialisation	8
3.3.2	Utilisation comme pointeur	10
3.3.3	Couplage à un objet	11
3.3.4	Temporisation	12
3.3.5	Evolution logicielle de la version 3.80	13
3.4	Résolution de problèmes	14
3.4.1	Compatibilité avec les versions 1.x	14
3.4.2	Fonctions obsolètes	14
3.4.3	Instabilité	14
3.4.4	Collage intempestif	14
3.4.5	Raideur trop faible	15
3.4.6	Retour d'effort trop faible	15
4	CHANGEMENT DE CONVENTIONS	16
4.1	Exemple de conversion	16
5	MANUEL DE REFERENCE	18
6	GLOSSAIRE	129

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 3

1 INTRODUCTION

1.1 *Qu'est-ce que l'API VIRTUOSE ?*

L'API VIRTUOSE est une bibliothèque de fonctions C permettant de s'interfacer avec un système à retour d'effort de la gamme VIRTUOSE.

L'API VIRTUOSE supporte les produits suivants :

- VIRTUOSE 6D35-45
- VIRTUOSE 3D35-45
- VIRTUOSE 3D10-20
- VIRTUOSE DESKTOP
- VIRTUOSE INCA

Elle permet de piloter simultanément plusieurs produits de la gamme, ainsi que le simulateur VIRTUOSE.

Elle est disponible pour différentes architectures et systèmes d'exploitation :


- Architecture PC sous Microsoft Windows 98/NT/2000/XP/7
- Architecture PC sous Linux (noyau 2.4 et noyau 2.6)

Selon les cas, elle se présente sous forme d'une librairie statique et/ou dynamique (dll sous Windows).

1.2 *Composition du document*

Ce document regroupe les chapitres suivants :

1. Introduction (ce chapitre)
2. Manuel d'Installation
3. Manuel Utilisateur
4. Manuel de Référence
5. Glossaire

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 4

2 MANUEL D'INSTALLATION

Comme il l'a été précédemment expliqué, l'API Virtuose est une librairie C. Ainsi son installation consiste simplement à inclure les fichiers qui la constituent au sein de l'environnement de développement. Ces fichiers (header et binaires) diffèrent en fonction du système d'exploitation utilisé.

Ainsi, les fichiers à inclure à l'environnement de développement sont :


- Sous Microsoft Windows 98/NT/2000/XP/7 :
 - virtuoseAPI.h
 - virtuoseAPI.lib
 - virtuoseAPI.dll
- Sous Linux 2.4 ou 2.6 :
 - virtuoseAPI.h
 - virtuoseAPI.so
 - libvirtuose.a

3 MANUEL UTILISATEUR

3.1 Contenu de l'API VIRTUOSE

L'API VIRTUOSE se compose des éléments suivants :

- Un fichier d'en-tête C et C++ à inclure dans votre projet : virtuoseAPI.h
- Une librairie statique au format Windows, Linux (ELF) : virtuoseAPI.lib
- Une librairie dynamique au format Windows (DLL)

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 5

3.2 Conventions

3.2.1 Programmation

Les fonctions de l'API VIRTUOSE respectent les conventions suivantes :

- Les noms de fonctions sont en langue anglaise, ils commencent toujours par « *virt* » et ne comportent aucun caractère « souligné » (*underscore*), la séparation entre mots étant matérialisée par des majuscules (ex : *virtGetPosition*).
- À l'exception de la fonction « *virtOpen* », toutes les fonctions prennent comme premier argument un objet de type « *VirtContext* » ; cet objet est créé par la fonction « *virtOpen* » et détruit par « *virtClose* ».
- À l'exception des fonctions « *virtOpen* », « *virtGetErrorCode* » et « *virtGetErrorMessage* », toutes les fonctions de l'API renvoie 0 en cas de succès et -1 en cas d'échec.
- Les fonctions de l'API ont toutes un prototype unique en C ; lorsqu'elles sont utilisées dans une application C++, leur prototype est spécifié comme « *extern "C"* » de façon à éviter toute surcharge.

3.2.2 Modes d'interaction

L'API VIRTUOSE comporte plusieurs modes d'interaction avec le système à retour d'effort. Selon le type d'application envisagé, le mode choisi sera différent :

1. Couplage force/position : dans ce mode très basique, l'application transmet au système les valeurs d'effort à appliquer au repère courant, et reçoit en retour la position et la vitesse du repère courant
2. Couplage position/force : ce mode évolué permet le couplage direct avec un objet virtuel ; dans ce cas, l'application transmet au système la position et la vitesse du centre de l'objet, et reçoit en retour le torseur d'efforts à appliquer au centre de l'objet lors de l'intégration de la dynamique. Les gains optimaux d'asservissement sont calculés par le contrôleur embarqué, à partir de la masse et du torseur d'inertie de l'objet couplé.
3. Couplage position/force avec guidage virtuel : ce mode ajoute au mode précédent un guidage virtuel de différent type (translation, rotation, ...).


3.2.3 Définition des repères

Les conventions d'orientation sont matérialisées par la notion de « repère ». Toute grandeur géométrique (position, vitesse, effort, inertie, etc.) est définie par rapport à un repère particulier.

Par définition, tous les repères utilisés dans l'API VIRTUOSE sont directs. Cela constitue une différence fondamentale par rapport aux conventions traditionnellement utilisées en informatique graphique. Nous proposons plus loin un schéma de conversion permettant de faire cohabiter les deux types de conventions.

Par défaut et à l'initialisation de l'API, tous les repères sont confondus et définis comme suit :

- Origine au centre du système à retour d'effort, c'est-à-dire au point d'intersection entre l'axe 1 (rotation de la base) et l'axe 2 (premier mouvement vertical).

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 6

- Axe X horizontal, aligné selon la position médiane de l'axe 1 (rotation de la base) et orienté vers l'utilisateur.
- Axe Y horizontal, perpendiculaire à l'axe X et orienté vers la droite de l'utilisateur.
- Axe Z vertical et orienté vers le haut

L'API définit les repères suivants :

1. Le repère d'environnement, qui correspond à l'origine de la scène ; il est fixé par l'application indépendamment de l'API.
2. Le repère d'observation, qui représente en général le repère écran ; il est positionné par rapport au repère d'environnement.
3. Le repère de base, qui représente le centre du système à retour d'effort ; il est positionné par rapport au repère d'observation.
4. Le repère outil, qui correspond à la base de l'outil fixé à l'extrémité du Virtuose ; il est positionné par rapport au repère d'environnement.
5. Le repère avatar, qui correspond à la position de la main de l'utilisateur par rapport au système haptique, et qui permet de prendre en compte la géométrie de l'outil fixé à son extrémité ; il est positionné par rapport au repère outil.

La position des repères suivants peut être modifiée à l'initialisation par l'API :


- Repère d'observation (par rapport au repère d'environnement)
- Repère de base (par rapport au repère d'observation)
- Repère avatar (par rapport au repère outil)

La position du repère suivant peut être modifiée à tout moment par l'API :

- Repère d'observation (par rapport au repère d'environnement)

La position du repère suivant ne peut jamais être modifiée :

- Repère outil (par rapport au repère d'environnement)

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 7

3.2.4 Unités physiques utilisées

Toutes les valeurs utilisées dans l'API sont exprimées en unités physiques et en convention métrique, à savoir :

- Durées en secondes (s)
- Dimensions en mètres (m)
- Angles en radians (rad)
- Vitesses linéaires en mètres par seconde (m.s^{-1})
- Vitesses angulaires en radians par seconde (rad.s^{-1})
- Efforts en Newtons (N)
- Couples en Newtons-mètres (N.m)
- Masses en kilogrammes (kg)
- Inerties en kg.m^2

3.2.5 Expression des grandeurs géométriques

L'API utilise différentes grandeurs géométriques, qui sont exprimées de la façon suivante :

- Les positions sont exprimées sous la forme de vecteur déplacement, qui regroupe sept composantes, à savoir un terme de translation (x, y, z) suivi d'un terme de rotation sous forme de quaternion normalisé (qx, qy, qz, qw) (cf. glossaire).
- Les vitesses sont exprimées sous forme de torseur cinématique (vx, vy, vz, wx, wy, wz) (cf. glossaire)
- Les efforts sont exprimés sous forme de torseur (fx, fy, fz, cx, cy, cz) (cf. glossaire)

3.3 Mise en œuvre de l'API VIRTUOSE

3.3.1 Initialisation

La première étape dans la mise en œuvre de l'API VIRTUOSE consiste à établir la communication avec le périphérique. La fonction `virtOpen` réalise cette opération, et elle crée en même temps toutes les variables internes nécessaires au fonctionnement de l'API, en renvoyant un pointeur de type `VirtContext`. Ce pointeur doit ensuite être utilisé pour chaque appel de fonction de l'API : il sert d'une part à accéder à l'état courant de la communication, et d'autre part à identifier le VIRTUOSE auquel est adressée la requête dans le cas d'une application à plusieurs interfaces haptiques.

Il est indispensable de tester la valeur de retour de la fonction `virtOpen`, car le fonctionnement de toute l'API est subordonné au succès de la connexion :

```
VirtContext VC;
VC = virtOpen("192.168.1.1");
if (VC == NULL)
{
    fprintf(stderr, "Erreur dans virtOpen: %s\n",
        virtGetErrorMessage(virtGetErrorCode(NULL));
    return -1;
}
```

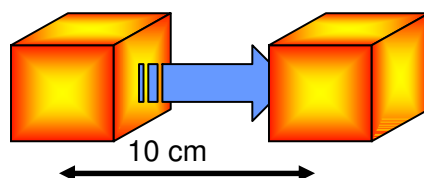
Ensuite, il est conseillé de réaliser explicitement la configuration du couplage, dans la mesure où la configuration par défaut n'est pas forcément conforme à l'utilisation qu'on veut faire du bras :

1. Le facteur d'échelle en déplacement (fonction `virtSetSpeedFactor`)

L'intérêt d'un facteur d'homothétie en vitesse est l'augmentation de l'amplitude des translations effectuées durant les simulations. Cette fonctionnalité est très appréciable dans le cadre des simulations nécessitant la manipulation d'objets dans un scène de grandes dimensions.

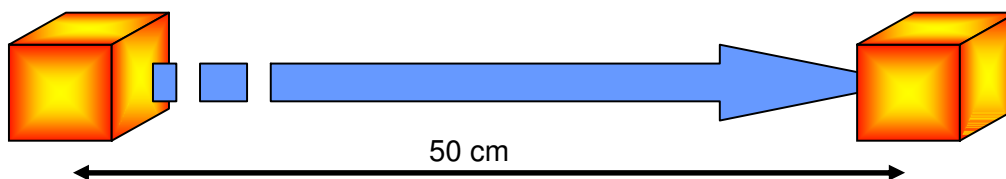
Le schéma suivant illustre l'effet d'un facteur d'homothétie en vitesse sur les déplacements en translation de l'objet virtuel manipulé :

Cas 1 : Pas d'homothétie en vitesse



Pour un déplacement horizontal de 10 cm de l'interface haptique, le déplacement du cube virtuel est équivalente.

Cas 2 : Homothétie de 5 en vitesse



Pour un déplacement horizontal de 10 cm de l'interface haptique, le déplacement du cube virtuel est 5 fois plus grand au sein du monde virtuel.

2. Le facteur d'échelle en efforts (fonction `virtSetForceFactor`)

L'intérêt d'un facteur d'homothétie en effort est la possibilité de jouer sur l'intensité des efforts à fournir par l'opérateur durant les manipulations grâce à une diminution des efforts restitués par l'interface haptique. Cette fonctionnalité est très appréciable dans le cadre des simulations nécessitant la manipulation d'objets de grande masse.

3. Le mode d'indexation (fonction `virtSetIndexingMode`)

Lorsque l'opérateur arrive proche d'une butée, il a la possibilité de remettre dans une position confortable en agissant sur le bouton de décalage. Il existe trois types de décalage :

- le type `INDEXING_ALL`, autorisant un décalage sur les translations et sur les rotations,
- le type `INDEXING_TRANS`, autorisant un décalage uniquement sur les translations,
- le type `INDEXING_NONE`, n'autorisant aucun décalage; l'action sur le bouton de décalage n'a aucun effet.

4. Le pas d'intégration du simulateur (fonction `virtSetTimeStep`)

5. La position du repère de base par rapport au repère d'observation (fonction `virtSetBaseFrame`)

6. La position du repère d'observation par rapport au repère d'environnement (fonction `virtSetObservationFrame`)

7. La vitesse du repère d'observation par rapport au repère d'environnement (fonction `virtSetObservationFrameSpeed`)

8. Le type de couplage (fonction `virtSetCommandType`)

Le code suivant correspond à une configuration d'usage fréquent :

```
float identity[7] = {0.0f,0.0f,0.0f,0.0f,0.0f,0.0f,1.0f};
virtSetIndexingMode(VC, INDEXING_ALL);
virtSetForceFactor(VC, 1.0f);
virtSetSpeedFactor(VC, 1.0f);
virtSetTimeStep(VC, 0.003f);
virtSetBaseFrame(VC, identity);
virtSetObservationFrame(VC, identity);
```

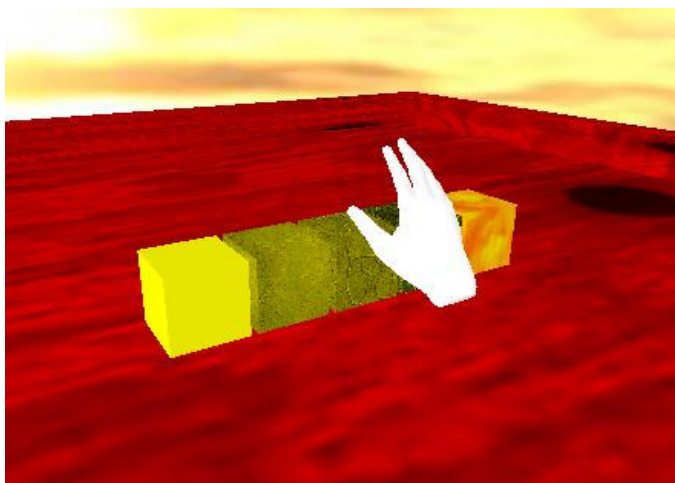
```
virtSetObservationFrameSpeed(VC, identity)
virtSetCommandType(VC, COMMAND_TYPE_VIRTMECH);
```

Enfin, l'initialisation doit se terminer par l'autorisation de mise sous puissance (fonction `virtSetPowerOn`). Cette fonction active le bouton-poussoir « MARCHE » du VIRTUOSE, qui permet à l'utilisateur de mettre les moteurs sous tension :

```
virtSetPowerOn(VC, 1);
```

3.3.2 Utilisation comme pointeur


Dans de nombreuses applications, il est nécessaire de disposer d'un mode « pointeur », dans lequel la position du VIRTUOSE est représentée à l'écran par un objet purement graphique (l'avatar), sur lequel ne s'exerce aucun effort. Ce mode est utile pour désigner un objet sur lequel on veut se coupler, ou encore pour appeler une fonction à l'aide d'un menu contextuel.



Afin d'implémenter ce mode de fonctionnement, on définit une fonction appelée périodiquement, qui va servir à mettre à jour les consignes du périphérique haptique et à récupérer sa position. Cette fonction est appelée par l'application, par exemple sous la forme d'un callback.

En mode simple `COMMAND_TYPE_IMPEDANCE` cette fonction doit forcer à 0 la consigne d'effort, et renvoyer la position du bras. Elle peut prendre la forme suivante :

```
int update_avatar(VirtContext VC, float *position)
{
    float null [6] = {0.0f,0.0f,0.0f,0.0f,0.0f,0.0f};
    virtSetForce(VC, null);
    virtGetPosition(VC, position);
    return 0;
}
```

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 11

En mode couplé `COMMAND_TYPE_VIRTMECH` (admittance), la fonction lit la position et la vitesse du bras, la renvoie à celui-ci comme consigne (ainsi, l'erreur de position et de vitesse est nulle, et donc aucun effort n'est généré) :

```
int update_avatar(VirtContext VC, float *position)
{
    float speed[6];
    float position[7];
    virtGetPosition(VC, position);
    virtGetSpeed(VC, speed);
    virtSetPosition(VC, position);
    virtSetSpeed(VC, speed);
    return 0;
}
```

Dans les deux cas, la fonction renvoie dans la variable `position` la position courante de l'avatar, en translation et en rotation. Cette position est affectée par le choix des repères de base et d'observation, ainsi que par le bouton de décalage (ou bouton d'indexation) : lorsque ce dernier est enfoncé, l'avatar ne suit plus les mouvements du bras.

3.3.3 Couplage à un objet

En mode couplé `COMMAND_TYPE_VIRTMECH`, on peut attacher le VIRTUOSE directement à un objet, et le bras devient alors une contrainte dynamique à intégrer dans la simulation de la scène.

Le couplage à un objet est effectué selon la procédure suivante :

1. L'appel de la fonction `virtAttachVO` réalise le couplage, en calculant les paramètres optimaux et en garantissant la stabilité ; pour cela, elle a besoin de connaître le pas d'intégration du simulateur, ainsi que la masse et le tenseur d'inertie de l'objet
2. La position et la vitesse du point de référence de l'objet sont envoyés comme consignes (fonction `virtSetPosition` et `virtSetSpeed`)

Le code suivant réalise le couplage avec un objet de masse m , d'inertie m_{xymz} , et dont le centre d'inertie est défini par la position P et la vitesse V :

```
virtAttachVO(VC, m, mxmymz);
virtSetPosition(VC, P);
virtSetSpeed(VC, S);
```

Afin de faire évoluer l'objet, on définit une fonction de mise à jour appelée périodiquement par l'application, par exemple sous forme de callback :

```
int update_object(VC, float *P, float *S, float *torseur_effort)
{
    virtSetPosition(VC, P);
    virtSetSpeed(VC, S);
    virtGetForce(VC, torseur_effort);
    return 0;
}
```

3.3.4 Temporisatation

Certaines temporisations logicielles sont nécessaires entre le positionnement d'un paramètre et sa lecture, le temps au contrôleur de prendre en compte ce nouveau paramétrage. Cela concernent :

- le repère de base du mécanisme virtuel,
- les enchaînements de mécanismes virtuels.

Exemples:


Mécanisme déplacement cartésien de la position courante de l'effecteur sur 20 cm le long de l'axe x de l'effecteur suivi d'un guide de type rotation d'axe x de l'effecteur.

```
float virtVmBaseFrame[7];
// demande au contrôleur de prendre la position courante de l'effecteur
// comme base du mécanisme virtuel
virtVmSetType(VC, VM_TYPE_CartMotion);
virtVmSetBaseFrameToCurrentFrame(VC);
Sleep(10);

// nouvelle modification du repère de base
virtVmGetBaseFrame(VC, virtVmBaseFrame);
virtVmBaseFrame[0] += 0.2;
virtVmSetBaseFrame(VC, virtVmBaseFrame);


// activation du guide
virtVmActivate(VC);
// attente d'arrivée en butée
virtVmWaitUpperBound(VC);
virtVmDeactivate(VC);
Sleep(10);

// mécanisme rotation suivant le même axe
virtVmSetType(VC, VM_TYPE_Rx);
virtVmActivate(VC);
```

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 13

3.3.5 Evolution logicielle de la version 3.80

1. Il existe maintenant un guide virtuel plan défini par les axes Ox et Oy du repère de base du mécanisme virtuel. L'opérateur et l'objet virtuel sont contraints à rester sur ce plan après activation du mécanisme.
2. Deuxième guide virtuel, le mécanisme manivelle. Le bras de levier est déterminé au moment de l'activation par la position du bras par rapport à l'axe Ox du repère de base du mécanisme virtuel.
3. En cours de simulation, il est possible de modifier dynamiquement le coefficient de déplacement (ou facteur d'échelle) et de travailler à différentes échelles dans une scène virtuelle. On peut imaginer d'effectuer en même temps un zoom de la caméra.
4. Le repère d'observation est lui aussi modifiable dynamiquement. Il se peut, après couplage, qu'il existe un petit effet d'inertie si le repère d'observation subit un fort changement d'orientation.
5. Un objet virtuel peut maintenant être saisi non plus au centre de gravité mais à une position précisée par la simulation: par exemple, la saisie d'un marteau par son manche. On aura donc un effet de levier du marteau sur les rotations.
6. Auparavant, l'opération ne commandait un objet virtuel qu'en position. Actuellement, l'opérateur peut commander l'objet en vitesse. L'opérateur doit définir le rayon d'une sphère dont le centre est le centre de l'espace de travail du périphérique. Lorsque la poignée est à l'intérieure, la commande reste en position. Quand elle est à l'extérieure, l'opération contrôle la vitesse de l'objet virtuel (norme et direction).
7. Il existe un autre type d'indexation, INDEXING_ALL_FORCE_FEEDBACK_INHIBITION, permettant d'inhiber le retour d'effort lorsque vous effectuez un décalage (appui sur le bouton de décalage). Si l'objet couplé est en collision avec une autre pièce de la scène à ce moment, le retour d'effort disparaît en peu à peu. Il réapparaît aussi peu à peu lorsque l'opération de décalage est terminé.
8. Le contrôle de la connexion est activé une fois la boucle haptique lancé. Ainsi, la boucle haptique peut ne pas suivre immédiatement la phase d'initialisation du virtuose.
9. Un mode de commande permet de contrôler le curseur de la souris sans se déplacer du périphérique haptique. Il s'active et se désactive sur double-clic d'un bouton du Virtuose. Dans ce mode, l'effecteur est contraint sur un plan vertical représentant l'écran, l'objet virtuel éventuellement attaché ne se déplace pas. Deux boutons situés sur la poignée simulent le bouton droit et gauche de la souris. Cette dernière reste toujours opérationnelle.

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 14

3.4 Résolution de problèmes

Nous citons ci-dessous quelques problèmes classiques qui peuvent survenir lors des premiers couplages avec un objet virtuel. Pour chacun d'entre eux, nous donnons des règles simples à adopter :

3.4.1 Compatibilité avec les versions 1.x

L'API VIRTUOSE est modifié sur les points suivants :

1. Les types de commande `COMMAND_TYPE_FORCE` et `COMMAND_TYPE_POSITION` sont remplacés par `COMMAND_TYPE_IMPEDANCE` et `COMMAND_TYPE_VIRT_MECH`.
2. La lecture de l'état des boutons se réalise par l'intermédiaire d'une seule fonction.
3. Certains paramètres, notamment la position du vecteur de base et d'observation, ne peuvent être positionné qu'avant la sélection du type de commande.

3.4.2 Fonctions obsolètes

Les fonctions `virtSetLimitTorque` et `virtGetLimitTorque` sont obsolètes. La saturation des efforts est maintenant implémentée dans la fonction `virtSaturateTorque`; elle a été nettement améliorée.

3.4.3 Instabilité

L'API VIRTUOSE calcule les paramètres de couplage en respectant deux critères :

1. Stabilité du système couplé (simulateur – objet virtuel – bras)
2. Optimalité (raideur maximale)

Si une instabilité survient néanmoins lors d'un couplage, il est bon de faire certaines vérifications :


- Le pas d'intégration donnée à l'API par la fonction `virtSetTimeStep` est-il vraiment identique à celui qui est utilisé par le simulateur ?
- Le simulateur ne prend-il pas du retard par rapport à ce pas d'intégration, c'est-à-dire le temps réel est-il vraiment synchrone avec le temps simulé ?
- Les paramètres de masse et d'inertie utilisés par le simulateur pour son intégration sont-ils les mêmes que ceux transmis au `Virtuose` ? Les unités sont-elles respectées ?

Par ailleurs, certains moteurs de détection de collision peuvent renvoyer des informations très bruitées, avec des normales de contact mal définies, qui créent des réactions à haute fréquence, que l'on peut interpréter comme une instabilité.

3.4.4 Collage intempestif

Ce phénomène est caractéristique d'une perte de performances de la liaison avec le Virtuose. C'est particulièrement le cas lorsque le bras est branché sur une branche réseau partagée avec de nombreuses stations de travail, ou encore lorsque le bras et le simulateur ne se trouvent pas sur la même branche réseau.

Dans ce cas, il est nécessaire de modifier la topologie du réseau, la meilleure solution étant de consacrer une carte réseau à la seule communication avec le VIRTUOSE.

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 15

3.4.5 Raideur trop faible

Ce phénomène apparaît lorsqu'on manipule des objets de grande taille, ou bien lorsque le facteur d'échelle en déplacement est inadapté. Il se traduit par une difficulté à maîtriser le déplacement de l'objet couplé, qui a trop d'inertie par rapport aux efforts exercés par le VIRTUOSE.

Dans ce cas, il est bon de diminuer l'inertie et la masse de l'objet couplé, soit en jouant sur sa taille (facteur d'échelle global pour toute la scène), soit en diminuant sa densité. On peut également, dans une certaine mesure, réduire le facteur d'échelle en effort, ce qui a pour effet de décupler les efforts exercés par le VIRTUOSE (voir ci-dessous).

3.4.6 Retour d'effort trop faible

Ce phénomène est observé lorsque le facteur d'échelle en effort est trop faible. Il peut aussi se produire avec certaines combinaisons de masse, d'inertie, et de facteur d'échelle en déplacement.

Dans tous les cas, il est souhaitable de remonter la valeur du facteur d'échelle en effort, ou à défaut de se rapprocher d'un cas nominal (objet de 10 cm et 300 g, échelle 1).

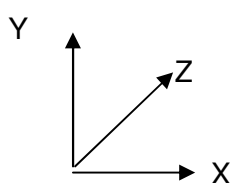
4 CHANGEMENT DE CONVENTIONS

Un problème classique est le passage entre une convention graphique (repères indirects, axe Z vers le fond de l'écran) et une convention robotique (repères directs, axe Z vertical).

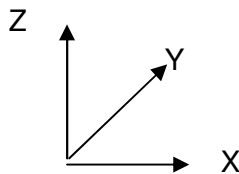
Nous proposons ci-dessous un schéma simple de conversion entre ces deux conventions, illustré par l'exemple du moteur physique Vortex™ distribué par la société Critical Mass Labs.

4.1 Exemple de conversion

Nous définissons les fonctions suivantes, qui ont pour effet de permuter les axes Y et Z, de façon à passer d'un repère graphique à un repère robotique et inversement :



Convention Vortex




Convention Virtuoso

- Les déplacements Vortex sont définies sous la forme de position + quaternion.
- Les vitesses Vortex sont définies sous la forme de vitesse linéaire + vitesse angulaire.

```
void transformVortexPosQuatToVirtPosEnv(float *vortexPos,
                                         float *vortexQuat,
                                         float *virtPosEnv)
{
    virtPosEnv[0] = vortexPos[0];
    virtPosEnv[1] = vortexPos[2];
    virtPosEnv[2] = vortexPos[1];
    virtPosEnv[3] = -vortexQuat[1];
    virtPosEnv[4] = -vortexQuat[3];
    virtPosEnv[5] = -vortexQuat[2];
    virtPosEnv[6] = vortexQuat[0];
}

void transformVirtPosEnvToVortexPosQuat(float *virtPosEnv,
                                         float *vortexPos,
                                         float *vortexQuat)
{
    vortexPos[0] = virtPosEnv[0];
    vortexPos[1] = virtPosEnv[2];
    vortexPos[2] = virtPosEnv[1];
    vortexQuat[0] = virtPosEnv[6];
    vortexQuat[1] = -virtPosEnv[3];
    vortexQuat[2] = -virtPosEnv[5];
    vortexQuat[3] = -virtPosEnv[4];
}
```


	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 17

```

void transformVortexLinAngVelToVirtSpeedEnv(float *vortexLinVel,
                                           float *vortexAngVel,
                                           float *virtSpeedEnv)
{
    virtSpeedEnv[0] = vortexLinVel[0];
    virtSpeedEnv[1] = vortexLinVel[2];
    virtSpeedEnv[2] = vortexLinVel[1];
    virtSpeedEnv[3] = -vortexAngVel[0];

    virtSpeedEnv[4] = -vortexAngVel[2];
    virtSpeedEnv[5] = -vortexAngVel[1];
}

void transformVirtTorsToVortexForceTorque(float *virtTors,
                                          float *vortexForce,
                                          float *vortexTorque)
{
    vortexForce[0] = virtTors[0];
    vortexForce[1] = virtTors[2];
    vortexForce[2] = virtTors[1];
    vortexTorque[0] = -virtTors[3];
    vortexTorque[1] = -virtTors[5];
    vortexTorque[2] = -virtTors[4];
}

void transformVortexInertiaTensorToVirtInertiaTensor(float *vortexTensor,
                                                      float *virtTensor)
{
    virtTensor[0] = vortexTensor[0];
    virtTensor[1] = vortexTensor[2];
    virtTensor[2] = vortexTensor[1];
    virtTensor[3] = vortexTensor[6];
    virtTensor[4] = vortexTensor[8];
    virtTensor[5] = vortexTensor[7];
    virtTensor[6] = vortexTensor[3];
    virtTensor[7] = vortexTensor[5];
    virtTensor[8] = vortexTensor[4];
}

```


5 MANUEL DE REFERENCE

Connexion...

virtOpen	page 22
virtClose	page 24

Initialisation...

virtAPIVersion	page 25
virtGetControllerVersion	page 26
virtSetOutputFile	page 27
virtDisableControlConnexion	page 28
virtSetPeriodicFunction	page 29
virtStartLoop	page 30
virtStopLoop	page 31
virtSetSpeedFactor	page 32
virtGetSpeedFactor	page 33
virtSetForceFactor	page 34
virtGetForceFactor	page 35
virtSetTimeoutValue	page 36
virtGetTimeoutValue	page 37
virtSetBaseFrame	page 38
virtGetBaseFrame	page 39
virtSetIndexingMode	page 40
virtGetIndexingMode	page 41
virtSetPowerOn	page 42
virtSetCommandType	page 43
virtGetCommandType	page 44
virtSetDebugFlags	page 45
virtGetDeviceID	page 46
virtSetGripperCommandType	page 47
virtSetTimeStep	page 48
virtGetTimeStep	page 49

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 19

Liaison à un objet...

virtAttachVO	page 50
virtDetachVO	page 51
virtAttachVOAvatar	page 52
virtDetachVOAvatar	page 53
virtSetCatchFrame	page 54
virtGetCatchFrame	page 55

Etat du bras...

virtGetButton	page 56
virtGetDeadMan	page 57
virtGetEmergencyStop	page 58
virtGetError	page 59
virtGetErrorCode	page 60
virtGetErrorMessage	page 62
virtGetPowerOn	page 63
virtIsInBounds	page 64
virtGetAlarm	page 65
virtIsInShiftPosition	page 66
virtGetMouseState	page 67
virtGetTrackball	page 68
virtGetTrackballButton	page 69
virtGetADC	page 70
virtWaitPressButton	page 71
virtForceShiftButton	page 72

Commande du bras...

virtSetPosition	page 73
virtGetPosition	page 74
virtSetSpeed	page 75
virtGetSpeed	page 76
virtSetForce	page 77
virtGetForce	page 78
virtSetObservationFrame	page 79
virtGetObservationFrame	page 80
virtGetAvatarPosition	page 81


virtGetPhysicalPosition	page 82
virtGetPhysicalSpeed	page 83
virtAddForce	page 84
virtGetArticularPositionOfAdditionalAxe	page 85
virtSetArticularPositionOfAdditionalAxe	page 86
virtGetArticularSpeedOfAdditionalAxe	page 87
virtSetArticularSpeedOfAdditionalAxe	page 88
virtSetArticularForceOfAdditionalAxe	page 89
virtGetArticularPosition	page 90
virtSetArticularPosition	page 91
virtGetArticularSpeed	page 92
virtSetArticularSpeed	page 93
virtSetArticularForce	page 94
virtEnableForceFeedback	page 95
virtSaturateTorque	page 96

Commande en vitesse...

virtActiveSpeedControl	page 97
virtDeactiveSpeedControl	page 98
virtActiveRotationSpeedControl	page 99
virtDeactiveRotationSpeedControl	page 100
virtIsInSpeedControl	page 101
virtSetForceInSpeedControl	page 102
virtSetTorqueInSpeedControl	page 103
virtGetCenterSphere	page 104
virtGetAxisOfRotation	page 105

Mécanisme virtuel...

virtVmSetType	page 106
virtVmActivate	page 107
virtVmDeactivate	page 108
virtTrajSetSamplingTimeStep	page 109
virtTrajRecordStart	page 110
virtTrajRecordStop	page 111
virtVmStartTrajSampling	page 112
virtVmGetTrajSamples	page 113

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 21


virtVmSetDefaultToCartesianMode	page 114
virtVmSetDefaultToTransparentPosition	page 115
virtVmSetBaseFrame	page 116
virtVmGetBaseFrame	page 117
virtVmWaitUpperBound	page 118
virtVmSetRobotMode	page 119
virtVmSaveCurrentSpline	page 120
virtVmLoadSpline	page 121
virtVmDeleteSpline	page 122
virtVmSetBaseFrameToCurrentFrame	page 123

Texture...

virtSetTexture	page 124
virtSetTextureForce	page 125
virtConvertRGBToGrayscale	page 126

Utilitaires...

virtConvertDeplToHomogeneMatrix	page 127
virtConvertHomogeneMatrixToDepl	page 128

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 22

NOM

virtOpen – connexion de l'API au contrôleur du Virtuose

SYNOPTIQUE

```
#include "virtuoseAPI.h"
VirtContext virtOpen(const char *host)
```

DESCRIPTION

La fonction `virtOpen` réalise la connexion de l'API au contrôleur du Virtuose.

PARAMETRES

Le paramètre `host` contient l'URL (*Uniform Ressource Locator*) du contrôleur VIRTUOSE auquel l'API doit se connecter.

Dans la version courante de l'API, un seul type de protocole est supporté, l'URL est donc de la forme suivante :

```
"udpxdr://identification:numero_de_port+interface"
```

`udpxdr` désigne le protocole à utiliser.

`identification` contient le nom du contrôleur VIRTUOSE s'il peut être résolu par un serveur DNS, ou à défaut son adresse IP sous forme de quatre entiers décimaux séparés par des point (ex. "192.168.0.1").

`numero_de_port` est un entier décimal spécifiant le port à utiliser. Par défaut, il vaut 0, et dans ce cas l'API cherche un port libre en partant de 3131.

`interface` correspond au lien physique à utiliser (ignoré dans le cas du protocole `udpxdr`).

Dans le cas où seul `nom_ou_adresse_ip` est renseigné, l'URL est complété de la façon suivante :

```
"udpxdr://nom_ou_adresse_ip:0"
```

Attention : cette complétion automatique est limitée au seul protocole `udpxdr`.
Le pré-préfixe standard "url:" défini par la norme est supporté mais ignoré.

VALEUR DE RETOUR

En cas de succès, la fonction `virtOpen` renvoie un pointeur de type `VirtContext` (pointeur sur une structure de données masquée). Sinon, elle renvoie `NULL` et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS


`VIRT_E_SYNTAX_ERROR`

Le paramètre `host` ne correspond pas à un URL valide.

`VIRT_E_COMMUNICATION_FAILURE`

La fonction `virtOpen` n'a pas réussi à communiquer avec le contrôleur du Virtuose

`VIRT_E_OUT_OF_MEMORY`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 23


La fonction `virtOpen` n'a pas réussi à allouer la mémoire nécessaire pour l'objet `VirtContext`.

`VIRT_E_INCOMPATIBLE_VERSION`

La version de l'API et celle du contrôleur sont incompatibles.

VOIR AUSSI

`virtClose`, `virtGetErrorCode`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 24

NOM

`virtClose` – fermeture de la connexion au contrôleur du Virtuose

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtClose(VirtContext VC);
```

DESCRIPTION

La fonction `virtClose` ferme la connexion au contrôleur du Virtuose.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

VALEUR DE RETOUR

En cas de succès, la fonction `virtClose` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtOpen`, `virtGetErrorCode`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 25

NOM

virtAPIVersion – lecture de la version de la librairie Virtuose

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtAPIVersion(VirtContext VC, int * major, int * minor);
```

DESCRIPTION

La fonction permet de lire le numéro de la version de la librairie virtuose.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `major` est un pointeur sur un entier. Il correspond à l'indice majeur de la version logicielle.

Le paramètre `minor` est un pointeur sur un entier. Il correspond à l'indice mineur de la version logicielle.


VALEUR DE RETOUR

En cas de succès, la fonction `virtAPIVersion` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 26

NOM

virtGetControlerVersion – lecture de la version logicielle du contrôleur

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetControlerVersion(VirtContext VC,
                           int* major, int* minor);
```

DESCRIPTION

La fonction permet de lire la version logicielle du contrôleur embarqué.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `major` est un pointeur sur un entier. Il correspond à l'indice majeur de la version logicielle.

Le paramètre `minor` est un pointeur sur un entier. Il correspond à l'indice mineur de la version logicielle.


VALEUR DE RETOUR

En cas de succès, la fonction `virtGetControlerVersion` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 27

NOM

virtSetOutputFile – définit le fichier de sortie

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetOutputFile(VirtContext VC, char * name);
```

DESCRIPTION

La fonction `virtSetOutputFile` permet de créer un fichier texte qui sera utilisé lors de messages pendant l'utilisation du périphérique. S'il existe déjà, le fichier sera écrasé et le contenu perdu.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `name` est une chaîne de caractère. Il correspond au nom du fichier créé.


VALEUR DE RETOUR

En cas de succès, la fonction `virtSetOutputFile` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 28

NOM

virtDisableControlConnexion – désactivation du contrôle de connexion entre la virtuelleAPI et le contrôleur

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtDisableControlConnexion (VirtContext VC, int disable)
```

DESCRIPTION

La fonction `VirtDisableControlConnexion` désactive ou réactive le mécanisme de surveillance de la connexion entre la librairie virtuelleAPI et le logiciel intégré dans le contrôleur.

Ce contrôle a été mis en place pour palier à un plantage de la simulation et remettre le contrôleur dans un état déterminé. Il considère la simulation plantée lorsque celle-ci n'envoie pas d'information au bout de 2 secondes.

Cette fonction permet ainsi le debugage de la simulation et de positionner des points d'arrêt sans interrompre le contrôleur.

Par contre, après l'utilisation de cette fonction puis plantage de la simulation, nous ne garantissons pas l'état du contrôleur. Il est aussi impératif d'utiliser la fonction `virtClose` à la fin de la simulation.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `disable` désactive le mécanisme s'il est positionné à 1, et permet de le réactiver s'il est positionné à 0.

VALEUR DE RETOUR

En cas de succès, la fonction `virtDisableControlConnexion` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtClose`

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 29

NOM

virtSetPeriodicFunction – initialisation du mécanisme d'appel de fonction périodique

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetPeriodicFunction (VirtContext VC,
                             void (*fn) (VirtContext, void*),
                             float* period,
                             void* arg);
```

DESCRIPTION

La fonction `VirtSetPeriodicFunction` permet de fournir un pointeur sur la fonction qui sera appelé cycliquement, à la période fournit en paramètre. L'appel de fonction est synchronisé sur la réception de trame en provenance du controleur, ce mécanisme fournit de bonne performance temps-réel.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `fn` est un pointeur de fonction.

Le paramètre `period` correspond à la période d'appel de la fonction, exprimé en secondes. Celle-ci doit correspondre à la période fournie lors de l'appel à la fonction `virtSetTimeStep`.

Le paramètre `arg` correspond à l'argument passé lors de l'appel.

VALEUR DE RETOUR

En cas de succès, la fonction `VirtSetPeriodicFunction` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtStartLoop`
`virtStopLoop`
`virtSetTimeStep`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 30

NOM

virtStartLoop – démarrage du mécanisme d'appel de fonction périodique

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtStartLoop (VirtContext VC);
```

DESCRIPTION

La fonction `VirtStartLoop` permet de lancer le mécanisme d'appel de fonction périodique défini précédemment lors de l'appel à la fonction `VirtSetPeriodicFunction`.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

VALEUR DE RETOUR

En cas de succès, la fonction `VirtStartLoop` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtSetPeriodicFunction`
`virtStopLoop`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 31

NOM

virtStopLoop – arrêt du mécanisme d'appel de fonction périodique

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtStopLoop (VirtContext VC);
```

DESCRIPTION

La fonction `VirtStopLoop` permet d'arrêter le mécanisme d'appel de fonction périodique défini précédemment lors de l'appel à la fonction `VirtSetPeriodicFunction`.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

VALEUR DE RETOUR

En cas de succès, la fonction `virtStopLoop` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtSetPeriodicFunction`
`virtStartLoop`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 32

NOM

`virtSetSpeedFactor` – fixe le facteur d'homothétie en déplacement

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetSpeedFactor(VirtContext VC, float factor)
```

DESCRIPTION

La fonction `virtSetSpeedFactor` permet de changer le facteur d'homothétie en déplacement, qui correspond à un changement d'échelle entre les déplacements du VIRTUOSE et ceux de la simulation.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `factor` contient le facteur d'homothétie à appliquer. Un facteur plus grand que l'unité correspond à une dilatation de l'espace de travail du Virtuose.

VALEUR DE RETOUR

En cas de succès, la fonction `virtSetSpeedFactor` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.


`VIRT_E_INCORRECT_VALUE`

La valeur du paramètre `factor` est incorrecte. Cette erreur est générée dans les cas suivants :

- facteur d'homothétie négatif ou nul
- facteur d'homothétie trop grand ou trop faible pour garantir la stabilité du système

VOIR AUSSI

`virtGetSpeedFactor`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 33

NOM

`virtGetSpeedFactor` – renvoie le facteur d'homothétie en déplacement

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetSpeedFactor(VirtContext VC, float *factor)
```

DESCRIPTION

La fonction `virtGetSpeedFactor` permet de lire la valeur du facteur d'homothétie en déplacement, qui correspond à un changement d'échelle entre les déplacements du VIRTUOSE et ceux de la simulation.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `factor` est renseigné par la fonction `virtGetSpeedFactor`. Un facteur plus grand que l'unité correspond à une dilatation de l'espace de travail du Virtuose.

VALEUR DE RETOUR

En cas de succès, la fonction `virtGetSpeedFactor` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtSetSpeedFactor`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 34

NOM

`virtSetForceFactor` – fixe le facteur d'homothétie en effort

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetForceFactor(VirtContext VC, float factor)
```

DESCRIPTION

La fonction `virtSetForceFactor` permet de changer le facteur d'homothétie en effort, qui correspond à un changement d'échelle entre les efforts exercés au niveau du VIRTUOSE et ceux de la simulation.

La fonction doit être appelée avant la sélection du type de commande.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `factor` contient le facteur d'homothétie à appliquer. Un facteur plus petit que l'unité correspond à une amplification des efforts depuis le VIRTUOSE vers la simulation.

La fonction doit être appelée avant la sélection du type de commande.

VALEUR DE RETOUR

En cas de succès, la fonction `virtSetForceFactor` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_INCORRECT_VALUE`

La valeur du paramètre `factor` est incorrecte. Cette erreur est générée dans les cas suivants :


- facteur d'homothétie négatif ou nul
- facteur d'homothétie trop grand ou trop faible pour garantir la stabilité du système

`VIRT_E_CALL_TIME`

La fonction est appelée après la sélection du type de commande.

VOIR AUSSI

`virtGetForceFactor`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 35

NOM

virtGetForceFactor – renvoie le facteur d'homothétie en effort

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetForceFactor(VirtContext VC, float *factor)
```

DESCRIPTION

La fonction `virtGetForceFactor` permet de lire le facteur d'homothétie en effort, qui correspond à un changement d'échelle entre les efforts exercés au niveau du VIRTUOSE et ceux de la simulation.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `factor` est rempli par la fonction `virtGetForceFactor`. Un facteur plus petit que l'unité correspond à une amplification des efforts depuis le VIRTUOSE vers la simulation.

VALEUR DE RETOUR

En cas de succès, la fonction `virtGetForceFactor` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtSetForceFactor`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 36

NOM

`virtSetTimeoutValue` – modification de la valeur du délai d'attente

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetTimeoutValue(VirtContext VC, float tout)
```

DESCRIPTION

La fonction `virtSetTimeoutValue` permet de modifier la valeur du délai d'attente avant abandon utilisé dans la communication avec le contrôleur embarqué.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `tout` contient la valeur du nouveau délai d'attente à utiliser.

VALEUR DE RETOUR

En cas de succès, la fonction `virtSetTimeoutValue` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`


Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_INCORRECT_VALUE`

La valeur du paramètre `tout` est incorrecte.

VOIR AUSSI

`virtGetTimeoutValue`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 37

NOM

`virtGetTimeoutValue` – accès à la valeur du délai d'attente

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetTimeoutValue(VirtContext VC, float *tout)
```

DESCRIPTION

La fonction `virtGetTimeoutValue` renvoie la valeur courante du délai d'attente avant abandon utilisé pour la communication avec le contrôleur embarqué.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `tout` est rempli par la fonction avec la valeur en secondes du délai d'attente.

VALEUR DE RETOUR

En cas de succès, la fonction `virtGetTimeoutValue` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtSetTimeoutValue`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 38

NOM

`virtSetBaseFrame` – positionner le repère de base du bras dans le repère d'observation

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetBaseFrame(VirtContext VC, float* base)
```

DESCRIPTION

La fonction `virtSetBaseFrame` permet positionner le repère du bras dans le repère d'observation.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `base` est un tableau de 7 flottants. Les trois premières valeurs correspondent aux translations, les quatres suivantes correspondent aux rotations sous la forme d'un quaternion.

VALEUR DE RETOUR

En cas de succès, la fonction `virtGetAxisOfRotation` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtGetBaseFrame`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 39

NOM

`virtGetBaseFrame` – position courante du repère de base

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetBaseFrame(VirtContext VC, float *pos)
```

DESCRIPTION

La fonction `virtGetBaseFrame` renvoie la position courante du repère de base par rapport au repère d'observation.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `pos` correspond au déplacement du repère de base par rapport au repère d'observation.

VALEUR DE RETOUR

En cas de succès, la fonction `virtGetBaseFrame` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtSetBaseFrame`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 40

NOM

`virtSetIndexingMode` – modification du mode d'indexation

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetIndexingMode(VirtContext VC, VirtIndexingType mode)
```

DESCRIPTION

La fonction `virtSetIndexingMode` permet de modifier le mode d'indexation (aussi appelé mode de décalage).

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `mode` correspond au mode d'indexation souhaité. Les valeurs possibles sont :

`INDEXING_ALL`

L'indexation agit sur les translations et les rotations. Le décalage s'effectue lors de l'appui du bouton de décalage ou lorsqu'il n'y a pas de puissance (bouton de facade ou relâchement du bouton « homme mort »).

`INDEXING_TRANS`

L'indexation agit sur les translations seulement. A la remise sous puissance, le bras est contraint sur un segment en rotation pour revenir à l'orientation au moment de l'arrêt de la puissance.

`INDEXING_NONE`

L'indexation est inopérante, même hors puissance. A la remise sous puissance, le bras est contraint sur un segment pour revenir à la position au moment de l'arrêt de la puissance.

Les autres modes correspondent aux modes ci-dessus à la différence que le retour de d'effort est inhibé pendant le décalage.

VALEUR DE RETOUR

En cas de succès, la fonction `virtSetIndexingMode` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`


Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_INCORRECT_VALUE`

La valeur du paramètre `mode` ne correspond pas à un mode valide.

VOIR AUSSI

`virtGetIndexingMode`

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 41

NOM

`virtGetIndexingMode` – accès au mode courant d'indexation

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virt(VirtContext VC, VirtIndexingMode *mode)
```

DESCRIPTION

La fonction `virtGetIndexingMode` permet d'accéder au mode d'indexation (aussi appelé mode de décalage).

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `mode` est rempli par la fonction. Les valeurs possibles sont :

`INDEXING_ALL`

L'indexation agit sur les translations et les rotations.

`INDEXING_TRANS`

L'indexation agit sur les translations seulement.

`INDEXING_NONE`

L'indexation est inopérante.

VALEUR DE RETOUR

En cas de succès, la fonction `virtGetIndexingMode` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtSetIndexingMode`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 42

NOM

virtSetPowerOn – Autorisation logicielle de la mise sous puissance

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetPowerOn(VirtContext VC, int power);
```

DESCRIPTION

La fonction `virtSetPowerOn` permet l'autorisation logicielle de la mise sous puissance des moteurs.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `power` correspond à l'autorisation logicielle. La valeur 1 signifie une autorisation de mise sous puissance des moteurs, la valeur 0 signifie une interdiction de la mise sous puissance des moteurs.

VALEUR DE RETOUR

En cas de succès, la fonction `virtGetPowerOn` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtGetPowerOn`

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 43

NOM

virtSetCommandType – changement du mode de couplage

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetCommandType(VirtContext VC, VirtCommandType type)
```

DESCRIPTION

La fonction `virtSetCommandType` permet de modifier le mode de couplage.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `type` correspond au mode de couplage souhaité. Les valeurs possibles sont :

`COMMAND_TYPE_NONE`

Aucun mouvement possible

`COMMAND_TYPE_IMPEDANCE`

Couplage force/position

`COMMAND_TYPE_VIRTMECH`

Couplage position/force avec mécanisme virtuel

VALEUR DE RETOUR

En cas de succès, la fonction `virtSetCommandType` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`


Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_INCORRECT_VALUE`

La valeur du paramètre `type` ne correspond pas à un mode valide.

VOIR AUSSI

`virtGetCommandType`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 44

NOM

virtGetCommandType – état du mode de couplage

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetCommandType(VirtContext VC, VirtCommandType *type)
```

DESCRIPTION

La fonction `virtGetCommandType` renvoie le mode de couplage actif.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `type` est rempli par la fonction. Les valeurs possibles sont :

`COMMAND_TYPE_IMPEDANCE`

Couplage force/position

`COMMAND_TYPE_VIRTMECH`

Couplage position/force avec mécanisme virtuel

VALEUR DE RETOUR

En cas de succès, la fonction `virtGetCommandType` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtSetCommandType`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 45

NOM

virtSetDebugFlags – activation d'un ou plusieurs niveaux de diagnostic

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetDebugFlags(VirtContext VC, unsigned short flags)
```

DESCRIPTION

La fonction `virtSetDebugFlags` permet d'activer les différents niveaux de diagnostic disponibles dans l'API VIRTUOSE.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `flags` est un champ de bits correspondant aux différents niveaux de diagnostic à activer :

- `DEBUG_SERVO` : informations relatives à la stabilité du système
- `DEBUG_LOOP` : informations relatives à l'exécution

VALEUR DE RETOUR

En cas de succès, la fonction `virtSetDebugFlags` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

NOM

virtGetDeviceID – lecture du type et du numéro de série du périphérique

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetDeviceID(VirtContext VC,
                   int * device_type
                   int * serial_number);
```

DESCRIPTION

La fonction permet de lire le type du périphérique et le numéro de série contenu sur la carte variateur du bras.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `device_type` correspond au type du périphérique rencontré dans la gamme des produits HAPTION.

DEVICE_VIRTUOSE_3D	1
DEVICE_VIRTUOSE_3D_DESKTOP	2
DEVICE_VIRTUOSE_6D	3
DEVICE_VIRTUOSE_6D_DESKTOP	4
DEVICE_VIRTUOSE_7D	5
DEVICE_MAT6D	6
DEVICE_MAT7D	7
DEVICE_INCA_6D	8
DEVICE_INCA_3D	9
DEVICE_ORTHESE	10
DEVICE_SCALE1	11
DEVICE_1AXE	12
DEVICE_OTHER	13

Le paramètre `serial_number` correspond au numéro de série du périphérique.

VALEUR DE RETOUR

En cas de succès, la fonction `virtGetDeviceID` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_INCOMPATIBLE_VERSION`

La version du contrôleur embarqué est inférieure à 5.50.

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 47

NOM

virtSetGripperCommandType – choix du type de commande de la pince

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetGripperCommandType(VirtContext VC,
                              VirtGripperCommandType type);
```

DESCRIPTION

La fonction permet de positionner le type de commande de la pince lorsque celle-ci est connectée au 7^e axe de la carte variateur.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `type` correspond au type de commande

GRIPPER_COMMAND_TYPE_NONE : non utilisation de la pince

GRIPPER_COMMAND_TYPE_POSITION : utilisation de la pince en mode admittance

GRIPPER_COMMAND_TYPE_IMPEDANCE : utilisation de la pince en mode impédance


VALEUR DE RETOUR

En cas de succès, la fonction `virtGetDeviceID` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

VIRT_E_INVALID_CONTEXT

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 48

NOM

`virtSetTimeStep` – fixe la valeur du pas d'échantillonnage de la simulation

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetTimeStep(VirtContext VC, float timeStep)
```

DESCRIPTION

La fonction `virtSetTimeStep` permet de communiquer au contrôleur embarqué la valeur du pas d'échantillonnage de la simulation. Cette valeur est utilisée par celui-ci afin de garantir la stabilité du système couplé.

La fonction doit être appelée avant la sélection du type de commande.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `timeStep` correspond au pas d'échantillonnage de la simulation, exprimé en secondes.

VALEUR DE RETOUR

En cas de succès, la fonction `virtSetTimeStep` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_INCORRECT_VALUE`

La valeur du paramètre `timeStep` est incorrecte. Cette erreur est générée dans les cas suivants :


- valeur négative ou nulle
- valeur inférieure au pas d'échantillonnage utilisé par le contrôleur embarqué
- valeur trop grande pour assurer la stabilité du système

`VIRT_E_CALL_TIME`

La fonction est appelée après la sélection du type de commande.

VOIR AUSSI

`virtGetTimeStep`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 49

NOM

`virtGetTimeStep` – lecture du pas d'échantillonnage de la simulation

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetTimeStep(VirtContext VC, float* step)
```

DESCRIPTION

La fonction `virtGetTimeStep` permet de lire la valeur du pas d'échantillonnage de la simulation préalablement fixée par la fonction `virtGetTimeStep`. Elle retourne zéro si celle-ci n'est pas définie.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `step` correspond au pas d'échantillonnage de la simulation, exprimé en secondes.

VALEUR DE RETOUR

La fonction `virtGetTimeStep` renvoie 0 dans tous les cas.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtSetTimeStep`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 50

NOM

virtAttachVO – couple un objet avec le VIRTUOSE

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtAttachVO(VirtContext VC, float mass, float *inertia)
```

DESCRIPTION

La fonction `virtAttachVO` réalise le couplage de l'objet désigné avec le VIRTUOSE, et calcule des gains d'asservissement optimaux en garantissant la stabilité globale du système.

L'objet est défini par la position de son centre (point de réduction du torseur d'efforts), qui doit impérativement être envoyée à l'aide de la fonction `virtSetPosition` avant l'appel de `virtAttachVO`.

Cette fonction n'est accessible que dans les modes de commande `COMMAND_TYPE_VIRTMECH` et `COMMAND_TYPE_IMPEDANCE`.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `mass` correspond à la masse de l'objet, exprimée en kg.

Le paramètre `inertia` correspond à la matrice d'inertie de l'objet, stockée en lignes sous forme d'un vecteur à 9 composantes.

VALEUR DE RETOUR

En cas de succès, la fonction `virtAttachVO` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtDetachVO`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 51

NOM

virtDetachVO – découple un objet de la liaison avec le VIRTUOSE

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtDetachVO(VirtContext VC)
```

DESCRIPTION

La fonction `virtDetachVO` annule le couplage en cours d'un objet avec le VIRTUOSE, et remet à zéro les gains d'asservissement.

Cette fonction n'est accessible que dans les modes de commande `COMMAND_TYPE_VIRTMECH` et `COMMAND_TYPE_IMPEDANCE`.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

VALEUR DE RETOUR

En cas de succès, la fonction `virtDetachVO` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtAttachVO`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 52

NOM

virtAttachVOAvatar – couple un objet avec le VIRTUOSE

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtAttachVOAvatar(VirtContext VC, float mass, float
*inertia)
```

DESCRIPTION

La fonction `virtAttachVOAvatar` réalise le couplage de l'objet désigné avec le VIRTUOSE, et calcule des gains d'asservissement optimaux en garantissant la stabilité globale du système.

L'objet est défini par la position de son centre (point de réduction du torseur d'efforts), qui doit impérativement être envoyée à l'aide de la fonction `virtSetPosition` avant l'appel de `virtAttachVOAvatar`.

La différence avec la fonction `virtAttachVO` consiste en la saisie de l'objet en son centre de gravité mais au repère de l'avatar. Il existe donc un bras de levier.

Cette fonction n'est accessible que dans les modes de commande `COMMAND_TYPE_VIRTMECH` et `COMMAND_TYPE_IMPEDANCE`.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `mass` correspond à la masse de l'objet, exprimée en kg.

Le paramètre `inertia` correspond à la matrice d'inertie de l'objet, stockée en lignes sous forme d'un vecteur à 9 composantes.

VALEUR DE RETOUR

En cas de succès, la fonction `virtAttachVOAvatar` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtDetachVOAvatar`
`virtSetCatchFrame`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 53

NOM

virtDetachVOAvatar – découple un objet de la liaison avec le VIRTUOSE

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtDetachVOAvatar(VirtContext VC)
```

DESCRIPTION

La fonction `virtDetachVOAvatar` annule le couplage en cours d'un objet avec le VIRTUOSE, et remet à zéro les gains d'asservissement.

Cette fonction n'est accessible que dans les modes de commande `COMMAND_TYPE_VIRTMECH` et `COMMAND_TYPE_IMPEDANCE`.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

VALEUR DE RETOUR

En cas de succès, la fonction `virtDetachVOAvatar` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtAttachVOAvatar`

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 54

NOM

virtSetCatchFrame – position du repère de prise d'un objet virtuel

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetCatchFrame (VirtContext VC, float* frame)
```

DESCRIPTION

La fonction `virtSetCatchFrame` permet de position le repère de prise par rapport au repère du centre de l'objet.

Ce décalage par rapport au centre doit être fixé avant la demande de couplage effectuée par la fonction `virtAttachVO`. Il est remis à zéro à chaque appel de la fonction `virtDettachVO`.

La partie orientation du repère de prise n'a aucune influence.

La fonction n'est disponible que dans le mode `COMMAND_TYPE_VIRTMECH`.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `frame` correspond au repère de prise de l'objet exprimé dans le repère du centre de gravité de l'objet virtuel.

VALEUR DE RETOUR

En cas de succès, la fonction `virtSetCatchFrame` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`


Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_INCORRECT_VALUE`

Les valeurs du paramètre `frame` est incorrect (quaternion non normé)

VOIR AUSSI

`virtAttachVO`
`virtDettachVO`
`virtGetCatchFrame`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 55

NOM

virtGetCatchFrame – lecture du repère de prise d'un objet virtuel

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetCatchFrame (VirtContext VC, float* frame)
```

DESCRIPTION

La fonction `virtGetCatchFrame` permet de récupérer le repère de prise par rapport au repère du centre de l'objet, préalablement fixé par la fonction `virtSetCatchFrame`.

La fonction n'est disponible que dans le mode `COMMAND_TYPE_VIRTMECH`.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `frame` correspond au repère de prise de l'objet exprimé dans le repère du centre de gravité de l'objet virtuel.

VALEUR DE RETOUR

En cas de succès, la fonction `virtGetCatchFrame` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtAttachVO`
`virtDettachVO`
`virtSetCatchFrame`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 56

NOM

virtGetButton – état d'un bouton

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetButton(VirtContext VC, int button, int *state)
```

DESCRIPTION

La fonction `virtGetButton` renvoie l'état d'un des boutons placés sur l'outil monté à l'extrémité du Virtuose.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `button` correspond au numéro de bouton.

Le paramètre `state` est renseigné par la fonction (0 : bouton relâché, 1 : bouton enfoncé).

VALEUR DE RETOUR

En cas de succès, la fonction `virtGetButton` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`VirtGetDeadMan`, `virtGetEmergencyStop`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 57

NOM

virtGetDeadMan – état du capteur de sécurité

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetDeadMan(VirtContext VC, int *deadman)
```

DESCRIPTION

La fonction `virtGetDeadMan` renvoie l'état du capteur de sécurité placé à l'extrémité du bras (aussi appelé « capteur d'homme mort »).

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `deadman` est rempli par la fonction. La valeur 1 signifie que le capteur de sécurité est enclenché (utilisateur présent), la valeur 0 que le capteur est déclenché (utilisateur absent).

VALEUR DE RETOUR

En cas de succès, la fonction `virtGetDeadMan` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtGetEmergencyStop`, `virtGetPowerOn`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 58

NOM

virtGetEmergencyStop – état de la chaîne d'arrêt d'urgence

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetEmergencyStop(VirtContext VC, int *stop)
```

DESCRIPTION

La fonction `virtGetEmergencyStop` renvoie l'état de la chaîne d'arrêt d'urgence.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `stop` est rempli par la fonction. La valeur 1 signifie que la chaîne est fermée (le système est opérationnel), la valeur 0 qu'elle est ouverte (le système est arrêté).

VALEUR DE RETOUR

En cas de succès, la fonction `virtGetEmergencyStop` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtGetDeadMan`, `virtGetPowerOn`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 59

NOM

virtGetError – accès à l'état de fonctionnement du virtuose

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetError (VirtContext VC, int* error)
```

DESCRIPTION

La fonction `virtGetError` permet de tester l'état de fonctionnement du virtuose, que ce soit en termes de pannes ou de simples erreurs au niveau de chacun des composants matériels. Les composants testés sont les variateurs, les capteurs de position et le capteur d'effort.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `error` est rempli par la fonction avec la valeur d'état de fonctionnement du Virtuose. La valeur 0 signifie que le virtuose est en bon état de fonctionnement. La valeur 1 signifie une panne ou une erreur de fonctionnement.

VALEUR DE RETOUR

En cas de succès, la fonction `virtGetError` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_HARDWARE_ERROR`

Ce message d'erreur signifie la présence d'une ou plusieurs pannes ou erreurs au niveau de l'interface haptique.

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 60

NOM

virtGetErrorCode – accès au code de la dernière erreur signalée

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetErrorCode(VirtContext VC);
```

DESCRIPTION

La fonction `virtGetErrorCode` renvoie le code de la dernière erreur signalée sur la liaison avec le contrôleur Virtuose correspondant au paramètre `VC`.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`, ou à `NULL` dans le cas d'un échec de la fonction `virtOpen`.

VALEUR DE RETOUR

Par définition, la fonction `virtGetErrorCode` renvoie toujours un code d'erreur (voir ci-dessous).

ERREURS

Les erreurs ci-dessous ont une signification générique. Dans certains cas, le diagnostic dépend de la fonction qui a signalé l'erreur. Se reporter alors à la description de la fonction en cause.

`VIRT_E_NO_ERROR`

Aucune erreur signalée.

`VIRT_E_OUT_OF_MEMORY`

Erreur d'allocation mémoire.

`VIRT_E_INVALID_CONTEXT`

Un parameter de type `VirtContext` erroné a été transmis à l'API.

`VIRT_E_COMMUNICATION_FAILURE`

Une erreur a été signalée sur la communication avec le contrôleur du Virtuose.

`VIRT_E_FILE_NOT_FOUND`

Le nom donné en paramètre ne correspond à aucun fichier.

`VIRT_E_WRONG_FORMAT`

Une erreur a été détectée dans un format de données.

`VIRT_E_TIME_OUT`

Le délai maximum d'attente de réponse a été dépassé.


`VIRT_E_NOT_IMPLEMENTED`

La fonction n'est pas implémentée dans cette version de l'API.

`VIRT_E_VARIABLE_NOT_AVAILABLE`

La variable demandée n'est pas disponible sur ce modèle de Virtuose.

`VIRT_E_INCORRECT_VALUE`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 61

Une valeur transmise à l'API a une valeur incorrecte ou en-dehors des limites autorisées.

VIRT_E_INVALID_CONTEXT


Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VIRT_E_HARDWARE_ERROR

Problème de fonctionnement d'un des composants matériels du virtuelle.

VOIR AUSSI

`virtOpen`, `virtGetErrorMsg`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 62

NOM

virtGetErrorMessage – message explicatif d'un code d'erreur

SYNOPTIQUE

```
#include "virtuoseAPI.h"
const char *virtGetErrorMessage(int code)
```

DESCRIPTION

La fonction `virtGetErrorMessage` renvoie un message explicatif du code d'erreur donné en paramètre.

PARAMETRES

Le paramètre `code` contient le code d'erreur dont on veut obtenir une explication.

VALEUR DE RETOUR


Dans tous les cas, la fonction `virtGetErrorMessage` renvoie un message d'erreur.

ERREURS

Aucune.

VOIR AUSSI

`virtGetErrorCode`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 63

NOM

virtGetPowerOn – état de l'alimentation des moteurs

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetPowerOn(VirtContext VC, int *poweron)
```

DESCRIPTION

La fonction `virtGetPowerOn` renvoie l'état de l'alimentation des moteurs.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `poweron` est rempli par la fonction. La valeur 1 signifie que les moteurs sont alimentés, la valeur 0 qu'ils ne le sont pas.

VALEUR DE RETOUR

En cas de succès, la fonction `virtGetPowerOn` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtSetPowerOn`, `virtGetEmergencyStop`, `virtGetDeadMan`

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 64

NOM

`virtIsInBounds` – indique si l'opérateur se situe en butée sur le bras VIRTUOSE

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtIsInBounds (VirtContext VC, unsigned int *bounds);
```

DESCRIPTION

La fonction `virtIsInBounds` permet de savoir si l'opérateur se situe en butée sur le bras VIRTUOSE.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `bounds` est rempli par la fonction. Il représente un ensemble de bit dont voici la signification :

- le bit `VIRT_BOUND_LEFT_AXE_1` correspond à la butée gauche sur l'axe 1,
- le bit `VIRT_BOUND_RIGHT_AXE_1` correspond à la butée droite sur l'axe 1,
- le bit `VIRT_BOUND_SUP_AXE_2` correspond à la butée supérieure sur l'axe 2,
- le bit `VIRT_BOUND_INF_AXE_2` correspond à la butée inférieure sur l'axe 2,
- le bit `VIRT_BOUND_SUP_AXE_3` correspond à la butée supérieure sur l'axe 3,
- le bit `VIRT_BOUND_INF_AXE_3` correspond à la butée inférieure sur l'axe 3,
- le bit `VIRT_BOUND_LEFT_AXE_4` correspond à la butée gauche sur l'axe 4,
- le bit `VIRT_BOUND_RIGHT_AXE_4` correspond à la butée droite sur l'axe 4,
- le bit `VIRT_BOUND_SUP_AXE_5` correspond à la butée supérieure sur l'axe 5,
- le bit `VIRT_BOUND_INF_AXE_5` correspond à la butée inférieure sur l'axe 5,
- le bit `VIRT_BOUND_LEFT_AXE_6` correspond à la butée gauche sur l'axe 6,
- le bit `VIRT_BOUND_RIGHT_AXE_6` correspond à la butée droite sur l'axe 6.


VALEUR DE RETOUR

En cas de succès, la fonction `virtStopLoop` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 65

NOM

virtGetAlarm – récupère les alarmes en cours

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetAlarm (VirtContext VC, unsigned int *alarm);
```

DESCRIPTION

La fonction `VirtGetAlarm` permet de récupérer les alarmes en cours.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `alarm` est rempli par la fonction. Il représente un ensemble de bit dont voici la signification :

- le bit `VIRT_ALARM_OVERHEAT` indique qu'un des moteurs est en chauffe. Le retour d'effort est alors réduit pour maintenir une température acceptable.
- le bit `VIRT_ALARM_SATURATE` indique une saturation des moteurs; par défaut, le retour d'effort est au maximum de 35 Newton en force instantanée et de 3.1 Nm en couple (sur les VIRTUOSE 6D).
- le bit `VIRT_ALARM_CALLBACK_OVERRUN` indique que le temps d'exécution de la fonction périodique définie lors de l'appel de `virtSetPeriodicFunction` dépasse le temps fourni en paramètre. Aucun traitement n'est réalisé en cas de dépassement. Il est indispensable que le temps d'exécution de la callback soit inférieure au temps précisé.

VALEUR DE RETOUR

En cas de succès, la fonction `virtStopLoop` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtSaturateTorque`
`virtSetPeriodicFunction`
`virtSetTimeStep`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 66

NOM

`virtIsInShiftPosition` – indique l'état de décalage en translation

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtIsInShiftPosition
    (VirtContext VC, int* decalage)
```

DESCRIPTION

La fonction `virtIsInShiftPosition` indique l'état du décalage.

Il retourne

- 1, lorsque le périphérique est en décalage,
- 0, sinon.

Le périphérique est en décalage s'il n'est pas sous puissance ou si le bouton de décalage est enfoncé, autorisé ou pas suivant le type d'indexation.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `decalage` correspond à l'état de décalage en translation.


VALEUR DE RETOUR

En cas de succès, la fonction `virtIsInShiftPosition` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 67

NOM

`virtGetMouseState` – indique l'état des boutons en mode souris

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetMouseState (VirtContext VC,
                      int* actif,
                      int* clic_gauche,
                      int* clic_droit)
```

DESCRIPTION

La fonction `virtGetMouseState` indique si le mode souris est activé, l'état du bouton droite et gauche situés sur la poignet du périphérique.
 Cette fonction est utile si la machine graphique est différente de la machine de gestion de la physique et du périphérique haptique.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `actif` correspond à l'état du mode souris.

Le paramètre `clic_gauche` correspond l'état du bouton gauche.

Le paramètre `clic_droit` correspond l'état du bouton droit.


VALEUR DE RETOUR

En cas de succès, la fonction `virtIsInShiftPosition` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 68

NOM

virtGetTrackball – lecture de la trackball

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetTrackball(VirtContext VC, int* x_move, int* y_move)
```

DESCRIPTION

La fonction `virtGetTrackball` permet la lecture des mouvements x et y de la trackball. Les deux positions sont filtrées dans le contrôleur puis remontées à travers la librairie.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `x_move` correspond à la position sur l'axe x.

Le paramètre `y_move` correspond à la position sur l'axe y.


VALEUR DE RETOUR

En cas de succès, la fonction `virtGetTrackball` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 69

NOM

virtGetTrackballButton – lecture des boutons de la trackball

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetTrackballButton(VirtContext VC,
                           int* actif,
                           int* btn_gauche,
                           int* btn_milieu,
                           int* btn_droit)
```

DESCRIPTION

Il existe deux modes de fonctionnement de la poignet : le mode Virtuose et le souris. Le mode est sélectionnable par un switch situé sur la poignet. En mode souris, la fonction `virtGetTrackballButton` permet la lecture des boutons gauche, milieu et droit de la poignet. En mode Virtuose, tous les boutons de la souris sont considérés comme inactifs.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `actif` correspond à la position du switch. 0 signifie que on est en mode Virtuose, 1 signifie que on est en mode souris.

Le paramètre `btn_gauche` correspond à l'état du bouton gauche.

Le paramètre `btn_milieu` correspond à l'état du bouton du milieu.

Le paramètre `btn_droit` correspond à l'état du bouton droit.


VALEUR DE RETOUR

En cas de succès, la fonction `virtGetTrackballButton` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 70

NOM

virtGetADC – lecture d'une entrée analogiques

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetADC(VirtContext VC, int line, float* adc);
```

DESCRIPTION

La fonction `virtGetADC` permet de lire une des trois premières entrées analogiques après application d'un gain et d'un offset puis d'un filtrage.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `line` est un entier. Elle doit prendre la valeur 1, 2 ou 3. Les entrées supplémentaires ne sont pas remontées.

Le paramètre `adc` est un pointeur sur un flottant. Il est rempli par la fonction. Pour une pince, la valeur retournée est comprise entre 0 et 1 correspondant à un pourcentage de fermeture.


VALEUR DE RETOUR

En cas de succès, la fonction `virtGetADC` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 71

NOM

`virtWaitPressButton` – attente de l'appui d'un bouton

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtWaitPressButton(VirtContext VC,
                        int button_number);
```

DESCRIPTION

La fonction `virtWaitPressButton` permet d'attendre l'appui d'un bouton. Cette fonction est bloquante : elle s'endort jusqu'à l'arrivée de l'appui du bouton dont le numéro est passé en paramètre.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `button_number` est un entier. Il correspond au numéro de bouton.


VALEUR DE RETOUR

En cas de succès, la fonction `virtWaitPressButton` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 72

NOM

virtForceShiftButton – Force le décalage

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtForceShiftButton(VirtContext VC, int forceShiftButton);
```

DESCRIPTION

La fonction `virtForceShiftButton` permet de forcer de manière logicielle le décalage comme avec l'appui du bouton de décalage. Le décalage est effectif si celui-ci est autorisé.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `forceShiftButton` est un entier. La valeur 1 correspond à forcer le décalage, la valeur 0 permet de désactiver le décalage forcé.


VALEUR DE RETOUR

En cas de succès, la fonction `virtForceShiftButton` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 73

NOM

`virtSetPosition` – envoie au contrôleur la position de l'objet couplé

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetPosition(VirtContext VC, float *pos_p_env)
```

DESCRIPTION

La fonction `virtSetPosition` envoie au contrôleur la position de l'objet auquel le VIRTUOSE est couplé.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `pos_p_env` est un vecteur déplacement à 7 composantes (cf. glossaire).

VALEUR DE RETOUR

En cas de succès, la fonction `virtSetPosition` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.


`VIRT_E_INCORRECT_VALUE`

La valeur du paramètre `pos_p_env` est incorrecte. Cette erreur est générée dans les cas suivants :

- quaternion non normé

VOIR AUSSI

`VirtAttachVO`, `virtSetSpeed`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 74

NOM

`virtGetPosition` – renvoie la position courante de l'avatar ou de l'objet couplé au VIRTUOSE

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetPosition(VirtContext VC, float *pos_p_env)
```

DESCRIPTION

La fonction `virtGetPosition` renvoie la position courante de l'avatar, ou de l'objet couplé au VIRTUOSE si un couplage est en cours.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `pos_p_env` est renseigné par la fonction `virtGetPosition`. Elle est exprimée sous forme d'un vecteur déplacement à 7 composantes, projeté dans le repère d'environnement.

VALEUR DE RETOUR

En cas de succès, la fonction `virtGetPosition` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtSetPosition`, `virtAttachVO`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 75

NOM

`virtSetSpeed` – envoie au contrôleur embarqué la vitesse de l'objet couplé

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetSpeed(VirtContext VC, float *speed_p_env_r_ps)
```

DESCRIPTION

La fonction `virtSetSpeed` envoie au contrôleur du VIRTUOSE la vitesse de l'objet auquel il est couplé.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `speed_p_env_r_ps` correspond à la vitesse du centre de l'objet, exprimée sous forme d'un torseur 6 composantes projeté dans le repère d'environnement et réduit au centre de l'objet.

VALEUR DE RETOUR

En cas de succès, la fonction `virtSetSpeed` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtAttachVO`, `virtSetPosition`

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 76

NOM

`virtGetSpeed` – renvoie la vitesse de l'avatar, ou de l'objet couplé au VIRTUOSE

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetSpeed(VirtContext VC, float *speed_p_env_r_ps)
```

DESCRIPTION

La fonction `virtGetSpeed` renvoie la vitesse de l'avatar, ou bien de l'objet couplé au VIRTUOSE si un couplage est actif.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `speed_p_env_r_ps` est renseigné par la fonction `virtGetSpeed`. Elle est exprimée sous la forme d'un torseur à 6 composantes, projeté dans le repère d'environnement et réduit au centre de l'avatar ou de l'objet couplé au VIRTUOSE, selon le cas.

VALEUR DE RETOUR

En cas de succès, la fonction `virtGetSpeed` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtSetSpeed`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 77

NOM

virtSetForce – envoie l'effort à appliquer au VIRTUOSE

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetForce(VirtContext VC, float *force_p_env_r_ps)
```

DESCRIPTION

La fonction `virtSetForce` envoie la force à appliquer au VIRTUOSE, dans le cas d'un couplage en effort.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `force_p_env_r_ps` contient la force à appliquer, sous la forme d'un torseur à 6 composantes, projeté dans le repère d'environnement et réduit au centre du repère de l'objet virtuel.

Cette fonction n'est accessible qu'en mode de commande `COMMAND_TYPE_IMPEDANCE`.

VALEUR DE RETOUR

En cas de succès, la fonction `virtSetForce` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtSetCommandType`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 78

NOM

`virtGetForce` – renvoie le torseur d'efforts à appliquer à l'objet couplé

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetForce(VirtContext VC, float *force_p_env_r_ps)
```

DESCRIPTION

La fonction `virtGetForce` renvoie le torseur d'efforts à appliquer à l'objet auquel est couplé le VIRTUOSE, permettant la simulation dynamique de la scène.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `force_p_env_r_ps` est renseigné par la fonction `virtGetForce`. Il correspond aux efforts à appliquer à l'objet, sous forme d'un torseur 6 composantes projeté dans le repère d'environnement et réduit au centre de l'objet.

VALEUR DE RETOUR

En cas de succès, la fonction `virtGetForce` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtSetPosition`, `virtSetSpeed`

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 79

NOM

virtSetObservationFrame – modification du repère d'observation

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetObservationFrame(VirtContext VC, float *pos)
```

DESCRIPTION

La fonction `virtSetObservationFrame` permet de modifier la position du repère d'observation par rapport au repère d'environnement.
Elle doit être appelée avant la sélection du type de commande.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `pos` correspond au déplacement du repère d'observation par rapport au repère d'environnement.

VALEUR DE RETOUR

En cas de succès, la fonction `virtSetObservationFrame` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_INCORRECT_VALUE`


Le paramètre `pos` n'est pas un déplacement valide.

`VIRT_E_CALL_TIME`

La fonction est appelée après la sélection du type de commande.

VOIR AUSSI

`VirtGetObservationFrame`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 80

NOM

`virtGetObservationFrame` – position courante du repère d'observation

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetObservationFrame(VirtContext VC, float *pos)
```

DESCRIPTION

La fonction `virtGetObservationFrame` renvoie la position courante du repère d'observation par rapport au repère d'environnement.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `pos` correspond au déplacement du repère d'observation par rapport au repère d'environnement.

VALEUR DE RETOUR

En cas de succès, la fonction `virtGetObservationFrame` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtSetObservationFrame`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 81

NOM

`virtGetAvatarPosition` – récupère la position du bras Virtuose avec décalage

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetAvatarPosition (VirtContext VC, float* pos)
```

DESCRIPTION

La fonction `VirtGetAvatarPosition` permet de récupérer la position du poignet du Virtuose par rapport au repère d'environnement. Cette position tient compte du coefficient de déplacement.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `pos` correspond à la position. Elle est exprimée par rapport au repère d'environnement.

VALEUR DE RETOUR

En cas de succès, la fonction `VirtGetPhysicalPosition` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtGetAvatarSpeed`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 82

NOM

virtGetPhysicalPosition – récupère la position physique du bras Virtuose

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetPhysicalPosition (VirtContext VC, float* pos)
```

DESCRIPTION

La fonction `VirtGetPhysicalPosition` permet de récupérer la position de la poignet du Virtuose par rapport à sa base.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `pos` est un tableau de 7 flottants et correspond à la position. Elle est exprimée par rapport au repère de base du Virtuose.

VALEUR DE RETOUR

En cas de succès, la fonction `VirtGetPhysicalPosition` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtSetBaseFrame`
`virtGetPhysicalSpeed`

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 83

NOM

virtGetPhysicalSpeed – lecture de la vitesse de la poignet

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetPhysicalSpeed(VirtContext VC,
                        float* speed);
```

DESCRIPTION

La fonction `virtGetPhysicalSpeed` permet de lire la vitesse de la poignet dans le repère du bras réduit dans la poignet.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `speed` est un tableau de 6 flottants. Les 3 premières valeurs correspondent aux translations `vx`, `vy`, `vz` et les 3 suivantes correspondent aux rotations.


VALEUR DE RETOUR

En cas de succès, la fonction `virtGetPhysicalSpeed` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 84

NOM

virtAddForce – Ajout d'un effort en impédance

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtAddForce(VirtContext VC, float* force_p_bm_r_pm);
```

DESCRIPTION

La fonction permet d'ajouter un effort en impédance quelque soit le mode de commande.

PARAMETRES

Le paramètre VC correspond à l'objet VirtContext renvoyé par la fonction virtOpen.

Le paramètre force_p_bm_r_pm contient la force à appliquer, sous la forme d'un torseur à 6 composantes, projeté dans le repère de base du virtuose et réduit au centre du repère effecteur.


VALEUR DE RETOUR

En cas de succès, la fonction virtAddForce renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction virtGetErrorCode permet d'accéder au code d'erreur.

ERREURS

VIRT_E_INVALID_CONTEXT

Le paramètre VC ne correspond pas à un objet VirtContext valide.

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 85

NOM

virtGetArticularPositionOfAdditionalAxe – lecture de la position articulaire de la pince

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int  virtGetArticularPositionOfAdditionalAxe(VirtContext VC,
float* pos);
```

DESCRIPTION

La fonction permet la lecture de la position articulaire d'un moteur supplémentaire ou de la pince.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `pos` est un pointeur sur un flottant.


VALEUR DE RETOUR

En cas de succès, la fonction `virtGetArticularPositionOfAdditionalAxe` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 86

NOM

`virtSetArticularPositionOfAdditionalAxe` – envoie d'une consigne de position articulaire

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int  virtSetArticularPositionOfAdditionalAxe(VirtContext VC,
float* pos);
```

DESCRIPTION

La fonction permet l'envoi d'une consigne de position articulaire d'un moteur supplémentaire ou de la pince.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `pos` est un pointeur sur un flottant.


VALEUR DE RETOUR

En cas de succès, la fonction `virtSetArticularPositionOfAdditionalAxe` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 87

NOM

virtGetArticularSpeedOfAdditionalAxe – lecture de la vitesse articulaire de la pince

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int      virtGetArticularSpeedOfAdditionalAxe(VirtContext      VC,
float* speed);
```

DESCRIPTION

La fonction permet la lecture de la vitesse articulaire d'un moteur supplémentaire ou de la pince.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `speed` est un pointeur sur un flottant.


VALEUR DE RETOUR

En cas de succès, la fonction `virtGetArticularSpeedOfAdditionalAxe` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 88

NOM

virtSetArticularSpeedOfAdditionalAxe – envoi d'une consigne de vitesse articulaire

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int      virtSetArticularSpeedOfAdditionalAxe(VirtContext    VC,
float*   speed);
```

DESCRIPTION

La fonction permet l'envoi d'une consigne de vitesse articulaire d'un moteur supplémentaire ou de la pince.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `speed` est un pointeur sur un flottant.


VALEUR DE RETOUR

En cas de succès, la fonction `virtSetArticularSpeedOfAdditionalAxe` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 89

NOM

virtSetArticularForceOfAdditionalAxe – envoie d'une consigne d'effort

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int      virtSetArticularSpeedOfAdditionalAxe(VirtContext    VC,
float* effort);
```

DESCRIPTION

La fonction permet l'envoi d'une consigne d'effort articulaire d'un moteur supplémentaire ou de la pince. Il est nécessaire d'être dans le mode impédance pour être en impédance sur la pince.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `effort` est un pointeur sur un flottant.


VALEUR DE RETOUR

En cas de succès, la fonction `virtSetArticularForceOfAdditionalAxe` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 90

NOM

virtGetArticularPosition – lecture de la position articulaire du périphérique

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetArticularPosition(VirtContext VC, float* pos);
```

DESCRIPTION

La fonction permet la lecture de la position articulaire du périphérique haptique, sans aucun décalage.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `pos` est un tableau de flottant, de taille le nombre d'axe du bras (pince non comprise).

VALEUR DE RETOUR

En cas de succès, la fonction `virtGetArticularPosition` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_INCOMPATIBLE_VERSION`

La version du contrôleur embarqué est inférieure à 3.60

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 91

NOM

virtSetArticularPosition – envoie d'une consigne de position articulaire

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetArticularPosition(VirtContext VC, float* pos);
```

DESCRIPTION

La fonction permet l'envoi d'une consigne de position articulaire du périphérique haptique.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `pos` est un tableau de flottant, de taille le nombre d'axe du bras (pince non comprise).

VALEUR DE RETOUR

En cas de succès, la fonction `virtSetArticularPosition` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_INCOMPATIBLE_VERSION`

La version du contrôleur embarqué est inférieure à 3.60

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 92

NOM

virtGetArticularSpeed – lecture de la vitesse articulaire du périphérique

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetArticularSpeed(VirtContext VC, float* speed);
```

DESCRIPTION

La fonction permet la lecture de la vitesse articulaire du périphérique haptique.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `speed` est un tableau de flottant, de taille le nombre d'axe du bras (pince non comprise).

VALEUR DE RETOUR

En cas de succès, la fonction `virtGetArticularSpeed` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_INCOMPATIBLE_VERSION`

La version du contrôleur embarqué est inférieure à 3.60

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 93

NOM

virtSetArticularSpeed – envoie d'une consigne de vitesse articulaire

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetArticularSpeed(VirtContext VC, float* speed);
```

DESCRIPTION

La fonction permet l'envoi d'une consigne de vitesse articulaire du périphérique haptique.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `speed` est un tableau de flottant, de taille le nombre d'axe du bras (pince non comprise).

VALEUR DE RETOUR

En cas de succès, la fonction `virtSetArticularSpeed` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_INCOMPATIBLE_VERSION`

La version du contrôleur embarqué est inférieure à 3.60

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 94

NOM

`virtSetArticularForce` – envoie d'une consigne d'effort articulaire

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetArticularForce(VirtContext VC, float* force);
```

DESCRIPTION

La fonction permet l'envoi d'une consigne d'effort articulaire du périphérique haptique. Cette fonction n'est utilisable qu'en mode `COMMAND_TYPE_ARTICULAR_IMPEDANCE`.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `force` est un tableau de flottant, de taille le nombre d'axe du bras (pince non comprise).

VALEUR DE RETOUR

En cas de succès, la fonction `virtSetArticularForce` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_INCOMPATIBLE_VERSION`

La version du contrôleur embarqué est inférieure à 3.60

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 95

NOM

virtEnableForceFeedback – active et désactive le retour d'effort

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtEnableForceFeedback(VirtContext VC, int enable);
```

DESCRIPTION

La fonction `virtEnableForceFeedback` permet d'activer et de désactiver le retour d'effort du périphérique. Par défaut, le système autorise le retour d'effort. Cette fonction permet de désactiver temporairement le retour d'effort c'est-à-dire d'avoir le bras toujours sous puissance mais transparent. Ne pas activer le retour d'effort si l'objet manipulé est en inter pénétration (en collision).

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `enable` est un entier. La valeur 1 signifie une demande d'activation du retour d'effort, la valeur 0 signifie une désactivation du retour d'effort.


VALEUR DE RETOUR

En cas de succès, la fonction `virtEnableForceFeedback` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 96

NOM

virtSaturateTorque – saturation des efforts

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSaturateTorque(VirtContext VC,
                      float forceThreshold,
                      float momentThreshold);
```

DESCRIPTION

La fonction `virtSaturateTorque` permet de saturer en translation et en rotation l'effort cartésien appliqué au périphérique.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `forceThreshold` est un flottant. Elle correspond à l'effort cartésien en translation du périphérique. Elle ne doit pas être supérieure à l'effort initial fournie par HAPTION pour le périphérique. Par exemple, 35 N pour un virtuose 6D.

Le paramètre `momentThreshold` est un flottant. Elle correspond au couple appliqué sur les rotations. Par exemple, 3.3 Nm pour un virtuose 6D. Pour un système 3D, ce paramètre doit rester à 0.


VALEUR DE RETOUR

En cas de succès, la fonction `virtGetADC` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 97

NOM

`virtActiveSpeedControl` – active la commande en vitesse

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtActiveSpeedControl (VirtContext VC,
                           float radius,
                           float speedFactor);
```

DESCRIPTION

La fonction `virtActiveSpeedControl` permet d'activer la commande en vitesse. On définit une sphère dont le centre correspond au centre de l'espace de travail du périphérique haptique et de rayon, celui fournit en paramètre. A l'intérieur de la sphère, l'opérateur contrôle l'objet en position. A l'extérieur de la sphère, l'opérateur contrôle l'objet en vitesse. Plus la position du poignée s'écarte de la sphère, plus la vitesse de l'objet augmente. La valeur du rayon pouvant être utilisée pour un virtuose 3D 15-25 est de 0.1. Pour un virtuose 6D 35-45 est de 0.2. La valeur du coefficient dépend essentiellement des dimensions de la scène virtuelle (1.0 par exemple).

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.
 Le paramètre `radius` correspond au rayon de la sphère précédemment décrite.
 Le paramètre `speedFactor` est un coefficient multiplicatif sur la vitesse de l'objet virtuel.

VALEUR DE RETOUR

En cas de succès, la fonction `virtActiveSpeedControl` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`
 Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_INCORRECT_VALUE`
 Le rayon ou le coefficient multiplicatif ne sont pas positifs.

VOIR AUSSI

`virtDeactiveSpeedControl`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 98

NOM

`virtDeactiveSpeedControl` – désactive la commande en vitesse

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtDeactiveSpeedControl (VirtContext VC)
```

DESCRIPTION

La fonction `virtDeactiveSpeedControl` permet de désactiver la commande en vitesse. L'opérateur contrôle l'objet en position sur tout l'espace de travail.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.


VALEUR DE RETOUR

En cas de succès, la fonction `virtDeactiveSpeedControl` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 99

NOM

virtActiveRotationSpeedControl – activation du mode vitesse en rotation

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtActiveRotationSpeedControl(VirtContext VC,
                                   float angle,
                                   float speedFactor);
```

DESCRIPTION

La fonction permet d'activer et de paramétrer le mode vitesse en rotation. On définit un cône dont l'angle est défini en paramètre. A l'intérieur de ce cône, l'opérateur contrôle l'orientation de l'objet virtuel en position. A l'extérieur du cône, l'opérateur fournit une consigne de vitesse de rotation.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `angle` correspond à l'angle du cône.

Le paramètre `speedFactor` correspond à un coefficient multiplicatif sur la vitesse de rotation.

VALEUR DE RETOUR

En cas de succès, la fonction `virtActiveRotationSpeedControl` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`


Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_INCOMPATIBLE_VERSION`

La version du contrôleur embarqué est inférieure à 3.55

`VIRT_E_INCORRECT_VALUE`

Un des paramètres est négatif.

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 100

NOM

virtDeactiveRotationSpeedControl – désactivation du mode vitesse en rotation

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtDeactiveRotationSpeedControl(VirtContext VC);
```

DESCRIPTION

La fonction permet de désactiver le mode vitesse en rotation.

PARAMETRES

Le paramètre VC correspond à l'objet VirtContext renvoyé par la fonction virtOpen.

VALEUR DE RETOUR

En cas de succès, la fonction virtDeactiveRotationSpeedControl renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction virtGetErrorCode permet d'accéder au code d'erreur.


ERREURS

VIRT_E_INVALID_CONTEXT

Le paramètre VC ne correspond pas à un objet VirtContext valide.

VIRT_E_INCOMPATIBLE_VERSION

La version du contrôleur embarqué est inférieure à 3.55

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 101

NOM

`virtIsInSpeedControl` – indique si l'opération est en mode vitesse

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtIsInSpeedControl(VirtContext VC,
                        int* translation, int* rotation);
```

DESCRIPTION

La fonction indique si l'opérateur est en mode vitesse en translation et/ou en rotation.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `translation` est un pointeur sur un entier. Une valeur 0 correspond au mode position (position de l'effecteur à l'intérieur de la sphère). Une valeur 1 correspond au mode vitesse (position de l'effecteur à l'extérieur de la sphère).

Le paramètre `rotation` est un pointeur sur un entier. Une valeur 0 correspond au mode position (orientation de l'effecteur à l'intérieur du cône). Une valeur 1 correspond au mode vitesse (orientation de l'effecteur à l'extérieur du cône).

VALEUR DE RETOUR

En cas de succès, la fonction `virtIsInSpeedControl` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_INCOMPATIBLE_VERSION`

La version du contrôleur embarqué est inférieure à 3.60.

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 102

NOM

virtSetForceInSpeedControl – paramétrage de l'effort de rappel en mode vitesse

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetForceInSpeedControl(VirtContext VC,
                               float force);
```

DESCRIPTION

La fonction paramètre l'effort de rappel en translation lorsque l'opérateur est en mode vitesse.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `force` correspond à l'effort de rappel en newton lorsque l'opération commande un objet virtuel en mode vitesse, c'est-à-dire lorsque la position de l'effecteur est en dehors du cercle. Cet effort n'est pas saturé, cependant il est conseillé de ne pas dépasser l'effort nominal du périphérique haptique.

VALEUR DE RETOUR

En cas de succès, la fonction `virtSetForceInSpeedControl` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_INCOMPATIBLE_VERSION`

La version du contrôleur embarqué est inférieure à 3.60.

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 103

NOM

virtSetTorqueInSpeedControl – paramétrage du moment de rappel en mode vitesse

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetTorqueInSpeedControl(VirtContext VC,
                                float torque);
```

DESCRIPTION

La fonction paramètre le moment de rappel lorsque l'opérateur est en mode vitesse en rotation.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `torque` correspond au moment de rappel en newton-mètre lorsque l'opération commande un objet virtuel en mode vitesse en rotation, c'est-à-dire lorsque l'orientation de l'effecteur est en dehors du cône. Ce moment n'est pas saturé.

VALEUR DE RETOUR

En cas de succès, la fonction `virtSetTorqueInSpeedControl` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_INCOMPATIBLE_VERSION`

La version du contrôleur embarqué est inférieure à 3.60.

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 104

NOM

virtGetCenterSphere – lecture de la position du centre de la sphère

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetCentreSphere(VirtContext VC, float* pos)
```

DESCRIPTION

La fonction `virtGetCentreSphere` permet de lire la position du centre de la sphère dans le cas où l'on utilise la commande en vitesse. Ce centre se déplace lorsque le bras est en décalage, immobile dans le cas contraire.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `pos` est un tableau de 7 flottants.

VALEUR DE RETOUR

En cas de succès, la fonction `virtGetCenterSphere` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtGetAxisOfRotation`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 105

NOM

virtGetAxisOfRotation – lecture de l'axe de rotation de la poignet

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtGetAxisOfRotation(VirtContext VC, float* axe)
```

DESCRIPTION

La fonction `virtGetAxisOfRotation` permet de lire l'axe de rotation de la poignet dans le repère environnement. Dans le cas de la commande en vitesse sur les rotations, cet axe permet de visualiser quand l'opération sort du cône et passe en mode vitesse. La norme du vecteur rotation à la limite correspond au rayon de la sphère défini en paramètre de la fonction d'activation du mode en vitesse.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `axe` est un tableau de 3 flottants. Il correspond au vecteur rotation dans le repère environnement.

VALEUR DE RETOUR

En cas de succès, la fonction `virtGetAxisOfRotation` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

- `virtActiveSpeedControl`
- `virtActiveRotationSpeedControl`
- `virtGetCenterSphere`
- `virtDeactiveSpeedControl`
- `virtDeactiveRotationSpeedControl`

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 106

NOM

virtVmSetType— sélection du type de mécanisme virtuel

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtVmSetType(VirtContext VC, virtVmType type)
```

DESCRIPTION

La fonction virtVmSetType permet de sélectionner le type de mécanisme virtuel.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `type` correspond au type de mécanisme virtuel. Les types disponibles sont :

`VM_TYPE_CartMotion` : guide virtuel à zéro degré de liberté correspondant à un point, défini par l'origine du repère de base du mécanisme virtuel,

`VM_TYPE_SPLINE` : guide virtuel à un degré de liberté basé sur une courbe spline,

`VM_TYPE_Rx` : guide virtuel à un degré de liberté correspondant à une liaison de type pivot d'axe `Ox` du repère de base du mécanisme virtuel,

`VM_TYPE_Tx` : guide virtuel à un degré de liberté correspondant à une liaison de type glissière d'axe `Ox` du repère de base du mécanisme virtuel,

`VM_TYPE_TxTy` : guide virtuel à deux degrés de liberté correspondant à deux liaisons de type glissière d'axe `Ox` et `Oy` du repère de base du mécanisme virtuel,

`VM_TYPE_TxRx` : guide virtuel à deux degrés de liberté correspondant à une liaison de type pivot-glissant d'axe `Ox` du repère de base du mécanisme virtuel.

`VM_TYPE_RxRyRz` : guide virtuel à trois degrés de liberté correspondant à une liaison de type rotule par rapport au repère de base du mécanisme virtuel.

`VM_TYPE_Crank` : guide virtuel à deux degrés de liberté correspond à un mouvement d'une manivelle autour de l'axe `Ox` du repère de base du mécanisme virtuel.

VALEUR DE RETOUR

En cas de succès, la fonction `virtVmSetType` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`


Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_INCORRECT_VALUE`

La valeur du paramètre `type` est incorrecte.

VOIR AUSSI

`virtVmSetBaseFrame`

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 107

NOM

virtVmActivate– activation du mécanisme virtuel

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtVmActivate(VirtContext VC)
```

DESCRIPTION

La fonction `virtVmActivate` permet d'activer le mécanisme virtuel une fois le VIRTUOSE positionné.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

VALEUR DE RETOUR

En cas de succès, la fonction `virtVmActivate` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtVmDeactivate`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 108

NOM

virtVmDeactivate – désactivation du mécanisme virtuel

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtVmDeactivate(VirtContext VC)
```

DESCRIPTION

La fonction `virtVmDeactivate` permet de désactiver le mécanisme virtuel précédemment activé.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

VALEUR DE RETOUR

En cas de succès, la fonction `virtVmDeactivate` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtVmActivate`

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 109

NOM

`virtTrajSetSamplingTimeStep` – fixe la valeur du pas d'échantillonnage de l'enregistrement

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtTrajSetSamplingTimeStep(VirtContext VC,
                                float timeStep,
                                unsigned int* recordTime)
```

DESCRIPTION

La fonction `VirtTrajSetSamplingTimeStep` permet de fixer le pas d'échantillonnage de l'enregistrement de la trajectoire.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `timeStep` correspond au pas d'échantillonnage exprimé en secondes.

Le paramètre `recordTime` correspond au temps maximum d'enregistrement exprimé en secondes.

VALEUR DE RETOUR

En cas de succès, la fonction `VirtTrajSetSamplingTimeStep` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.


`VIRT_E_INCORRECT_VALUE`

Le paramètre du paramètre `timeStep` est incorrect. Cette erreur est générée dans les cas suivants :

- valeur négative ou nulle
- valeur inférieure au pas d'échantillonnage utilisé par le contrôleur embarqué

VOIR AUSSI

`VirtTrajRecordStart`, `virtTrajRecordStop`

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 110

NOM

`virtTrajRecordStart` – démarre l'enregistrement

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtTrajRecordStart (VirtContext VC)
```

DESCRIPTION

La fonction `VirtTrajRecordStart` permet de lancer l'enregistrement de la trajectoire.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

VALEUR DE RETOUR

En cas de succès, la fonction `VirtTrajRecordStart` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtTrajRecordStop`

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 111

NOM

virtTrajRecordStop – arrête l'enregistrement

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtTrajRecordStop (VirtContext VC)
```

DESCRIPTION

La fonction `VirtTrajRecordStop` permet d'arrêter l'enregistrement de la trajectoire.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

VALEUR DE RETOUR

En cas de succès, la fonction `VirtTrajRecordStop` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtTrajRecordStart`

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 112

NOM

virtVmStartTrajSampling – effectue une demande de lecture de la trajectoire

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtVmStartTrajSampling (VirtContext VC, unsigned int
nbSamples)
```

DESCRIPTION

La fonction `virtVmStartTrajSampling` permet de lancer une lecture d'une partie des points de trajectoire précédemment enregistrés.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `nbSamples` correspond au nombre de points.

VALEUR DE RETOUR

En cas de succès, la fonction `virtVmStartTrajSampling` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`


Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_INCORRECT_VALUE`

Le paramètre `nbSamples` est négatif ou supérieur à 3000 points.

VOIR AUSSI

`virtVmGetTrajSamples`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 113

NOM

virtVmGetTrajSamples – récupère les points de trajectoire

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtVmGetTrajSamples (VirtContext VC, float* samples)
```

DESCRIPTION

La fonction `VirtVmGetTrajSamples` permet de récupérer les points de trajectoire précédemment enregistrés. Cette fonction est bloquante jusqu'à la fin de la réception des points depuis le contrôleur.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `samples` correspond à un tableau de points.

VALEUR DE RETOUR

En cas de succès, la fonction `VirtVmGetTrajSamples` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtVmStartTrajSampling`

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 114

NOM

`virtVmSetDefaultToCartesianPosition` – bloque le bras avant l'activation du guide virtuel

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtVmDefaultToCartesianPosition (VirtContext VC)
```

DESCRIPTION

La fonction `VirtVmDefaultToCartesianPosition` permet de bloquer le bras du virtuose avant activation du mécanisme virtuel. Le bras sera de nouveau bloqué après désactivation.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

VALEUR DE RETOUR

En cas de succès, la fonction `VirtVmDefaultToCartesianPosition` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtSetDefaultToTransparentMode`
`virtVmActivate`
`virtVmDeactivate`

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 115

NOM

`virtVmSetDefaultToTransparentMode` – libère le bras avant l'activation du guide virtuel

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtVmDefaultToTransparentMode (VirtContext VC)
```

DESCRIPTION

La fonction `VirtVmDefaultToCartesienPosition` permet de mettre le bras du virtuose en mode transparent, avant activation du mécanisme virtuel. Le bras sera de nouveau dans ce mode après désactivation.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

VALEUR DE RETOUR

En cas de succès, la fonction `VirtVmDefaultToTransparentMode` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtSetDefaultToCartesianMode`
`virtVmActivate`
`virtVmDeactivate`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 116

NOM

virtVmSetBaseFrame – positionnement de la base du mécanisme virtuel

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtVmSetBaseFrame (VirtContext VC, float *base)
```

DESCRIPTION

La fonction `VirtVmSetBaseFrame` permet de positionner le repère de base du mécanisme virtuel.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `base` correspond au repère de base. Il est exprimé par rapport au repère d'environnement.

VALEUR DE RETOUR

En cas de succès, la fonction `VirtVmSetBaseFrame` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtVmType`
`virtVmGetBaseFrame`

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 117

NOM

virtVmGetBaseFrame – lecture de la base du mécanisme virtuel

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtVmGetBaseFrame (VirtContext VC, float *base)
```

DESCRIPTION

La fonction `VirtVmGetBaseFrame` permet de lire le repère de base du mécanisme virtuel.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `base` correspond au repère de base. Il est exprimé par rapport au repère d'environnement.

VALEUR DE RETOUR

En cas de succès, la fonction `VirtVmGetBaseFrame` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtVmSetBaseFrame`

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 118

NOM

virtVmWaitUpperBound – attente d'une butée supérieure d'un mécanisme virtuel

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtVmWaitUpperBound (VirtContext VC)
```

DESCRIPTION

La fonction `VirtVmWaitUpperBound` n'est disponible qu'en mode `COMMAND_TYPE_VIRTMECH` et pour les mécanismes déplacement cartésien et splines.

Avant activation, la fonction est non bloquante et retourne un code d'erreur.

Après activation, la fonction est bloquante. En mécanisme déplacement cartésien, elle est débloquée lorsque la butée supérieure du segment rejoignant le point final est atteint. En mécanisme Spline, elle est débloquée lorsque la butée supérieure de la spline est atteinte.

Dans tous les cas, la fonction est débloquée sur désactivation du mécanisme.

Cette fonction est souvent utilisée en mode robot et permet d'enchaîner les mécanismes virtuels.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

VALEUR DE RETOUR

En cas de succès, la fonction `VirtVmWaitUpperBound` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_INCORRECT_VM_TYPE`

Le type de mécanisme est incorrecte.

`VIRT_E_WRONG_MODE`


Le type de commande est incorrecte.

`VIRT_E_CALL_TIME`

L'appel de la fonction est effectuée avant activation du mécanisme.

VOIR AUSSI

virtVmSetType
virtVmActivate
virtVmDeactivate

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 119

NOM

virtVmSetRobotMode – Active et désactive le mode robot

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtVmSetRobotMode(VirtContext VC, int OnOff);
```

DESCRIPTION

La fonction `virtVmSetRobotMode` permet d'activer ou de désactiver le déplacement automatique du périphérique pour les mécanisme virtuels Segment et Splines.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `OnOff` est un entier. La valeur 1 correspond à l'activation du mode robot, la valeur 0 correspond à la désactivation du mode. Par défaut, le mode robot est désactivé.

VALEUR DE RETOUR

En cas de succès, la fonction `virtVmSetRobotMode` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.


ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_WRONG_MODE`

Le type de commande est différent de `COMMAND_TYPE_VIRTMECH`.

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 120

NOM

virtVmSaveCurrentSpline – sauvegarde la trajectoire en cours

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtVmSaveCurrentSpline(VirtContext VC, char * name);
```

DESCRIPTION

La fonction `virtVmSaveCurrentSpline` permet de sauvegarder une trajectoire sous la forme d'un fichier texte.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `name` est une chaîne de caractère. Il correspond au nom du fichier créé.

VALEUR DE RETOUR

En cas de succès, la fonction `virtVmSaveCurrentSpline` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`


Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_WRONG_MODE`

Le type de commande est différent de `COMMAND_TYPE_VIRTMECH`.

`VIRT_E_INCORRECT_VM_TYPE`

Le type de mécanisme virtuel est différent de `VM_TYPE_Spline`.

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 121

NOM

virtVmLoadSpline – chargement d'une trajectoire

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtVmLoadSpline(VirtContext VC, char * file_name);
```

DESCRIPTION

La fonction `virtVmLoadSpline` permet de charger une trajectoire précédemment sauvegardée.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `file_name` est une chaîne de caractère. Il correspond au nom du fichier précédemment sauvegardé.

VALEUR DE RETOUR

En cas de succès, la fonction `virtVmLoadSpline` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`


Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_WRONG_MODE`

Le type de commande est différent de `COMMAND_TYPE_VIRTMECH`.

`VIRT_E_INCORRECT_VM_TYPE`

Le type de mécanisme virtuel est différent de `VM_TYPE_Spline`.

	API VIRTUOSE V3.80		HAP/02/RT.006
	DOCUMENTATION		Rév. 2 page 122

NOM

virtVmDeleteSpline – suppression d'une trajectoire

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtVmDeleteSpline(VirtContext VC, char * file_name);
```

DESCRIPTION

La fonction `virtVmDeleteSpline` permet de charger une trajectoire précédemment sauvegardée.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `file_name` est une chaîne de caractère. Il correspond au nom du fichier à supprimer.

VALEUR DE RETOUR

En cas de succès, la fonction `virtVmDeleteSpline` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`


Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_WRONG_MODE`

Le type de commande est différent de `COMMAND_TYPE_VIRTMECH`.

`VIRT_E_INCORRECT_VM_TYPE`

Le type de mécanisme virtuel est différent de `VM_TYPE_Spline`.

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 123

NOM

`virtVmSetBaseFrameToCurrentFrame` – positionner le repère de base du mécanisme virtuel

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtVmSetBaseFrameToCurrentFrame(VirtContext VC)
```

DESCRIPTION

La fonction `virtVmSetBaseFrameToCurrentFrame` permet de positionner le repère de base du mécanisme virtuel au repère avatar, les deux repères étant exprimés dans le repère environnement.

Cette fonction n'est accessible que dans le mode de commande `COMMAND_TYPE_VIRTMECH`.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

VALEUR DE RETOUR

En cas de succès, la fonction `virtVmSetBaseFrameToCurrentFrame` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`


Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_WRONG_MODE`

Le type de commande est différent de `COMMAND_TYPE_VIRTMECH`.

VOIR AUSSI

`virtVmSetBaseFrame`

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 124

NOM

virtSetTexture – application d'une texture en un point de contact

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetTexture (VirtContext VC, float *H_texture_p_env,
float* intensite, int reinit)
```

DESCRIPTION

La fonction `VirtSetTexture` permet d'appliquer une texture en un point de contact suivant le niveau de gris ou plutôt de la variation des niveaux de gris en différents points. Elle permet de créer des sensations de reliefs, les bosses sont associées au blanc et les creux associés au noir.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `H_texture_p_env` est un vecteur déplacement à 7 composants correspondant à la position du point de contact.

Le paramètre `intensite` est un flottant correspondant à un niveau de gris entre 0 et 1.

Le paramètre `reinit` réinitialise le module de génération de texture au niveau du contrôleur. Il doit être positionner à 1 lors d'une nouvelle collision à 0 pour la prise en compte de texture.

VALEUR DE RETOUR

Dans tous les cas, la fonction `VirtSetTexture` renvoie 0.

ERREURS

`VIRT_E_INVALID_CONTEXT`


Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

`VIRT_E_INCORRECT_VALUE`

Le paramètre `intensité` n'est pas compris entre 0 et 1.

VOIR AUSSI

`virtConvertRGBToGrayscale`

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 125

NOM

virtSetTextureForce – application d'une force liée à une texture

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtSetTextureForce (VirtContext VC,
                        float *W_texture_p_env_r_ps)
```

DESCRIPTION

La fonction `virtSetTextureForce` permet d'appliquer directement une force.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `W_texture_p_env_r_ps` correspond à la force à appliquer, sous la forme d'un torseur à 6 composantes, projeté dans le repère d'environnement et réduit au centre du repère outil.


VALEUR DE RETOUR

En cas de succès, la fonction `virtSetTextureForce` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006	
	DOCUMENTATION	Rév. 2	page 126

NOM

virtConvertRGBToGrayscale – conversion RGB en niveau de gris

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtConvertRGBToGrayscale (VirtContext VC, float *rgb,
float* gray)
```

DESCRIPTION

La fonction `VirtConvertRGBToGrayscale` permet de convertir une couleur définie en RGB en niveau de gris. Les proportions réalisées sont :

29.9% de rouge,
58.7% de vert,
11.4% de bleu.

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `rgb` est un tableau de trois composants dont le premier correspond à la couleur rouge, le second à la couleur verte et le troisième à la couleur bleu.

VALEUR DE RETOUR

Dans tous les cas, la fonction `VirtConvertRGBToGrayscale` renvoie 0.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

VOIR AUSSI

`virtVmSetTexture`

NOM

virtConvertDeplToHomogeneMatrix – conversion d'un déplacement en matrice homogène

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtConvertDeplToHomogeneMatrix (VirtContext VC,
                                     float* d,
                                     float* m)
```

DESCRIPTION

La fonction `virtConvertDeplToHomogeneMatrix` permet la conversion d'une position représentée par un tableau en matrice homogène.

La position est définie par les 3 translations et le quaternion (la partie scalaire en dernier). La matrice homogène est représentée par un tableau de 16 flottants ou matrice 4x4 rangée en colonne:

a0	a4	a8	a12
a1	a5	a9	a13
a2	a6	a10	a14
a3	a7	a11	a15

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `d` correspond à une position en translation et rotation.

Le paramètre `m` correspond à une matrice.


VALEUR DE RETOUR

En cas de succès, la fonction `virtConvertDeplToHomogeneMatrix` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 128

NOM

virtConvertHomogeneMatrixToDepl – conversion d'une matrice homogène en déplacement

SYNOPTIQUE

```
#include "virtuoseAPI.h"
int virtConvertHomogeneMatrixToDepl (VirtContext VC,
                                     float* d,
                                     float* m)
```

DESCRIPTION

La fonction `virtConvertHomogeneMatrixToDepl` permet la conversion d'une matrice homogène en position.

La position est définie par les 3 translations et le quaternion (la partie scalaire en dernier). La matrice homogène est représentée par un tableau de 16 flottants ou matrice 4x4 rangée en colonne:

a0	a4	a8	a12
a1	a5	a9	a13
a2	a6	a10	a14
a3	a7	a11	a15

PARAMETRES

Le paramètre `VC` correspond à l'objet `VirtContext` renvoyé par la fonction `virtOpen`.

Le paramètre `d` correspond à une position en translation et rotation.

Le paramètre `m` correspond à une matrice.


VALEUR DE RETOUR

En cas de succès, la fonction `virtConvertHomogeneMatrixToDepl` renvoie 0. Dans le cas contraire, elle renvoie -1 et la fonction `virtGetErrorCode` permet d'accéder au code d'erreur.

ERREURS

`VIRT_E_INVALID_CONTEXT`

Le paramètre `VC` ne correspond pas à un objet `VirtContext` valide.

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 129

6 GLOSSAIRE

Quaternion

On peut passer d'une rotation angulaire d'angle θ autour d'un vecteur (a, b, c) à un quaternion par la formule suivante :

$$\begin{cases} qx = a \sin\left(\frac{\theta}{2}\right) \\ qy = b \sin\left(\frac{\theta}{2}\right) \\ qz = c \sin\left(\frac{\theta}{2}\right) \\ qw = \cos\left(\frac{\theta}{2}\right) \end{cases}$$

On peut ensuite normaliser le quaternion en divisant chacune de ses quatre composantes par sa norme calculée ainsi :

$$n = \sqrt{qx^2 + qy^2 + qz^2 + qw^2}$$

Torseur cinématique

Soit un solide (S) attaché à un repère (R1)={O1,x1,y1,z1} en mouvement dans un repère (R0)={O,x0,y0,z0}.

La vitesse d'un point M quelconque de (S) par rapport à (R0) est notée :


$$\vec{V}(M \in S / R_0) = \left(\frac{dOM}{dt} \right)_{R_0} = \left(\frac{dOO_1}{dt} \right)_{R_0} + x \left(\frac{dx_1}{dt} \right)_{R_0} + y \left(\frac{dy_1}{dt} \right)_{R_0} + z \left(\frac{dz_1}{dt} \right)_{R_0}$$

L'indice R0 de la dérivation / temps signifie que, dans le calcul de la vitesse, on suppose les

vecteurs de base de (R0) constants. On note : $\left(\frac{dOO_1}{dt} \right)_{R_0} = v_x \vec{x}_0 + v_y \vec{y}_0 + v_z \vec{z}_0$

Le vecteur rotation de S dans R0 est le vecteur Ω tel que :

$$\begin{cases} \left(\frac{dx_1}{dt} \right)_{R_0} = \vec{\Omega} \wedge \vec{x}_1 \\ \left(\frac{dy_1}{dt} \right)_{R_0} = \vec{\Omega} \wedge \vec{y}_1 \\ \left(\frac{dz_1}{dt} \right)_{R_0} = \vec{\Omega} \wedge \vec{z}_1 \end{cases}$$

	API VIRTUOSE V3.80	HAP/02/RT.006
	DOCUMENTATION	Rév. 2 page 130

Il est indépendant du choix du repère R1 lié à S. On note :

On peut en déduire la relation suivante :

$$V(M \in S / R_0) = V(O_1 / R_0) + \Omega \wedge O_1 M$$

Le champ des vitesses de S est celui du champ de moment d'un torseur appelé torseur cinématique dont les éléments de réduction sont :

$$(VS / R_0)_{O_1} = \left\{ \begin{array}{l} V_{\vec{\omega}}(O_1 \in S / R_0) \\ \Omega(S / R_0) \end{array} \right\} = (v_x, v_y, v_z, \omega_x, \omega_y, \omega_z)$$

Dans ce document, on parlera du « torseur cinématique de S réduit en O1 et projeté dans R0 ».