Optitrack

lundi 4 janvier 2021

11:35

**Trame principale**

Le protocole Natnet communique sur la base de la trame suivante  
*trame* = [*packetType*, *packetSize,* **paquet***, ‘\0*’] avec

|  |  |  |  |
| --- | --- | --- | --- |
| **Désignation** | **Taille (bytes)** | **Type** | **Contenu** |
| *packetType* | 2 | int | Type du paquet |
| *packetSize* | 2 | int | Longueur totale du paquet incluant le byte de séparation |
| **paquet** |  | String | Paquet contenant un message dont la teneur et la longueur dépendent de l'identifiant (voir paragraphe suivant) |
| *'\0'* | 1 | String | Byte de séparation, termine un packet |

On recense dix types de paquets identifiés selon la codification :

    # Client/server message ids

    NAT\_PING                  = 0

    NAT\_PINGRESPONSE          = 1

    NAT\_REQUEST               = 2

    NAT\_RESPONSE              = 3

    NAT\_REQUEST\_MODELDEF      = 4

    NAT\_MODELDEF              = 5

    NAT\_REQUEST\_FRAMEOFDATA   = 6

    NAT\_FRAMEOFDATA           = 7

    NAT\_MESSAGESTRING         = 8

    NAT\_DISCONNECT            = 9

    NAT\_UNRECOGNIZED\_REQUEST  = 100

Ces paquets peuvent être reçus ou émis par le client. Dans le cadre de notre application, nous allons nous focaliser sur deux types : **NAT\_REQUEST\_MODELDEF** et **NAT\_FRAMEOFDATA.** Ces informations sont en partie extraites du wiki d'optitrack (<https://v22.wiki.optitrack.com/index.php?title=NatNet:_Data_Types#Frame_of_Mocap_Data>), du code *PythonSample.py* et *NatNetClient.py*

**NAT \_MODELDEF**

Ce paquet est identifié par l'entier 5. Il encapsule la description des trois entités pouvant être publiées par Motive:

* *Les groupes de marqueurs* (marker set) : ce sont des listes de marqueurs n'étant pas apparentés à un corps rigide ou à un squelette
* Les *corps rigides* (rigid body) : ce sont les modèles des objets suivis par le logiciel. Ils se composent de marqueurs dont les étiquettes sont automatiquement définis par le logiciel. Par définition, les marqueurs d'un corps rigide auront des inter-distances invariant dans le temps.
* Les *squelettes* (Skeletons) : ce sont les modèles des objets articulés suivis par le logiciel, comme par exemple un bras robotisé ou un humain. Ils se composent de corps rigides hiérarchiquement organisés selon une relation parent/enfant.

Notons que ces descriptions sont regroupées par collection (set) mais ne sont pas nécessairement ordonnées.

Soit un paquet **descriptPacket** de type NAT \_MODELDEF, celui-ci se lit comme :

**descriptPacket** *=* [*setNb,* {**markerSetDescript**}, {**rigidBodyDescript**}, {**SkeletonDescript**}]

Où *setNb* est un entier sur 4 bytes donnant le nombre total de collections contenu dans le paquet. Les symboles {} désignent les collections d'entité qu'il faut décapsuler à l'aide par exemple d'une boucle (fonction *for*).

* Pour un groupe de marqueurs, la description contient :   
  **markerSetDescript**= [*setType*, *setName, markerCount,* {***markerName***}]

Où les variables en italique sont définies comme :

|  |  |  |  |
| --- | --- | --- | --- |
| **Désignation** | **Taille (bytes)** | **Type** | **Contenu** |
| *setType* | 4 | Int | Type de l'entité:  0 - groupe de marqueurs 1 - corps rigide 2 - squelette |
| *setName* |  | String | Nom du groupe de marqueurs terminé par un byte de séparation '\0' |
| *markerCount* | 4 | Int | Nombre de marqueurs contenus dans le groupe |

Ce paquet contient une collection :

* {***markerName***} : une liste d'étiquette au format utf8 et séparés par le byte '\0', qui dénomment les marqueurs du groupe

* Pour un corps rigide, le format de la description dépend de la version du protocole NatNet. Dans le cas où la version utilisée est supérieure ou égale à 3.0, alors   
  **rigidBodyDescript** *=* [*setType, setName, setId, parentID, offParent, mkrCount,* {**mkrOffset**}, {**activeLabel**}]

avec

|  |  |  |  |
| --- | --- | --- | --- |
| **Désignation** | **Taille (bytes)** | **Type** | **Contenu** |
| *setType* | 4 | Int | Type de l'entité:  0 - groupe de marqueurs 1 - corps rigide 2 - squelette |
| *setName* |  | String | Nom du corps terminé par un byte de séparation '\0' |
| *setId* | 4 | int | Identifiant du corps rigide |
| *parentID* | 4 | int | Identifiant du corps rigide parent (utilisé lorsque que le corps appartient à un squelette) |
| *offParent* | 12 | Float[3] | Coordonnée sur xyz du décalage avec le corps parent |
| *mkrCount* | 4 | Int | Nombre de marqueurs contenu dans le corps |

Ce paquet contient deux collections :

* {**mkrOffset**} : une liste des coordonnées des marqueurs dans le repère du corps ayant pour origine le point « pivot ». Le format d'une coordonnée est un tableau de trois float, chacun étant encodé sur 4 bytes
* {**activeLabel**} : une liste d’identifiants sur 4 bytes pour les marqueurs utilisés en mode actif. En mode inactif, la valeur des identifiants est 0 pour tous les marqueurs.
* Pour un squelette, on a   
  **SkeletonDescript** *=* [*setType, setName, setId, rbCount,* {**rigidBodyDescript**}]

Avec

|  |  |  |  |
| --- | --- | --- | --- |
| **Désignation** | **Taille (bytes)** | **Type** | **Contenu** |
| *setType* | 4 | Int | Type de l'entité:  0 - groupe de marqueurs 1 - corps rigide 2 - squelette |
| *setName* |  | String | Nom du squelette terminé par un byte de séparation '\0' |
| *setId* | 4 | int | Identifiant unique du squelette |
| *rbCount* | 4 | int | Nombre de corps rigides contenu dans le squelette |

Ce paquet encapsule une collection de corps rigide {**rigidBodyDescript**}.

**NAT\_FRAMEOFDATA**

Ce paquet est identifié par l'entier 7. Il encapsule les informations contenues dans une *frame*, notamment la localisation des marqueurs, des corps rigides et des squelettes. La taille de ce paquet dépend des options sélectionnées dans Motive. Néanmoins, sa structure est invariante et s'organise ainsi :

|  |  |  |  |
| --- | --- | --- | --- |
| **Désignation** | **Taille (bytes)** | **Type** | **Contenu** |
| *iFrame* | 4 | Int | Numéro de la frame défini par l'hôte (serveur) |
| *nMarkers* | 4 | Int | Nombre total de marqueurs étiquetés dans la frame |
| {**LabeledMarkers}** |  |  | Collection de données pour les marqueurs étiquetés |
| *nOtherMarkers* | 4 | Int | Nombre de marqueurs non-étiquetés |
| {**OtherMarkers**} |  |  | Collection de données pour les marqueurs non-étiquetés |
| *nMarkersSets* | 4 | Int | Nombre de groupes de marqueurs |
| {**MarkerSetsData**} |  |  | Collection de données pour les groupes de marqueurs |
| *nRigidBodies* | 4 | Int | Nombre de corps rigides |
| {**RigidBodiesData**} |  |  | Collection de données pour les corps rigides |
| *nSkeletons* | 4 | Int | Nombre de squelettes |
| {**SkeletonData**} |  |  | Collection de données pour les squelettes |
| *nForcePlates* | 4 | Int | Nombre de capteurs d'effort |
| {**ForcesPlates**} |  |  | Collection de données pour les capteurs d'effort |
| *nDevices* | 4 | Int | Nombre total d'instrument analogue (e.g. NI-DAQ) |
| {**Devices**} |  |  | Collection de données pour les instruments analogiques |
| **TimeInfos** |  |  | Structure de données relative au temps (voir détails plus bas) |

Dans notre cas applicatif, nous utiliserons principalement les corps rigides et les données temporelles. Pour cette raison, nous détaillons uniquement les structures de données liées à ces entités.

* Pour un corps rigide, on lira la structure de données correspondante comme :

**RigidBodiesData** = [*id, pos, rot*]

Avec

|  |  |  |  |
| --- | --- | --- | --- |
| **Désignation** | **Taille (bytes)** | **Type** | **Contenu** |
| *id* | 4 | Int | Identifiant unique défini comme une propriété du corps sous Motive. Si le corps appartient à un squelette, alors cette identifiant concatène l'identifiant du squelette et celui du corps rigide (os). |
| *pos* | 12 | Float[3] | Coordonnées du centre de masse du corps dans le repère xyz |
| *rot* | 16 | Float[4] | Orientation du corps sous forme de quaternion |

* Pour les informations temporelles, on lira :

**TimeInfos** = [*timeCode, timeCodeSub, timeStamp, stpCamExpo, stpDRcv, stpDTr, param*]

|  |  |  |  |
| --- | --- | --- | --- |
| **Désignation** | **Taille (bytes)** | **Type** | **Contenu** |
| *timeCode* | 4 | Int | Temps de référence pour la frame |
| *timeCodeSub* | 4 | Int | Sous-division du temps de référence, utilisé lorsque la période d'échantillonnage vidéo est supérieur à celle de l'horloge |
| *timeStamp* | 8 | Double | Temps de référence depuis le démarrage du logiciel Motive |
| *stpCamExpo* | 8 | Int | Indice de temps correspondant au centre de la fenêtre d'exposition des caméra (utile pour le calcul de latence) |
| *stpDRcv* | 8 | Int | Indice de temps auquel le logiciel a recu les informations des caméras (utile pour le calcul de latence) |
| *stpDTr* | 8 | Int | Indice de temps auquel le logiciel a traité les données de caméra et est prêt à transmettre ces informations (utile pour le calcul de latence) |
| *param* | 2 | Hex | Paramètres de contrôle. En partant du bit de poids faible : bit 0 - état de l'enregistrement : activé (1), désactivé (0) bit 1 - état du mode de suivi : activé (1), désactivé (0) |