Kinao

Generated by Doxygen 1.7.6.1

Sat Jul 12 2014 14:26:09

Contents

Chapter 1

Class Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

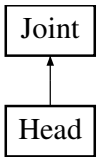
Class Documentation

3.1 Head Class Reference

Aqui se describe la clase derivada "Head".

`#include <joint.h>`

Inheritance diagram for Head:



Public Member Functions

- Head ()

  Declaracion de la clase derivada "Head" y su funcionalidad.
- virtual ~Head (void)

  Constructor del Objeto.
- virtual float getHeadPitch (XnUserID, xn::UserGenerator)

  Obtiene el angulo que se determina a partir del kinect.
- virtual float setHeadPitch (float, float &, float &)

  Genera el angulo necesario para mover el NAO.

3.1.1 Detailed Description

Aqui se describe la clase derivada "Head".

3.1.2 Member Function Documentation

3.1.2.1 float Head::getHeadPitch (XnUserID *user*, xn::UserGenerator *guser*) [virtual]

Obtiene el angulo que se determina a partir del kinect.

Obtiene el angulo real dado por el kinect del movimiento frontal de la cabeza.

Destructor del Objeto

The documentation for this class was generated from the following files:

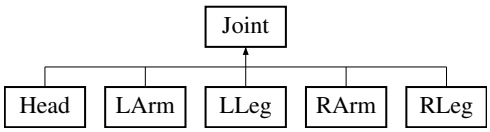
- joint.h
- head.cpp

3.2 Joint Class Reference

Aqui se describe la clase base "Joint".

`#include <joint.h>`

Inheritance diagram for Joint:



Public Member Functions

- Joint ()

  Esta es la decralarion de la funcionalidad de la clase base "Joint", de aqui se deriban las demas.
- virtual ~Joint (void)

  Constructor del Objeto.
- virtual float length (float[3], float[3])

  Destructor del Objeto.

- virtual float angle (float, float, float)
- virtual float getAngle (XnVector3D, XnVector3D, XnVector3D)

  L1,L2,L3(L3 es el opuesto del angulo a determinar)
- virtual XnVector3D getNormalVector (XnVector3D, XnVector3D, XnVector3D)

  Genera el vector resultante de realizar el producto cruz entre los 2 vectores coplanares.
- virtual XnVector3D getProyectionVector (XnVector3D, XnVector3D)

  Obtiene el vector resultante de proyectar un vector sobre otro.
- virtual XnFloat getProyection (XnVector3D Vector1, XnVector3D Vector2)

  Obtiene la constante de la proyeccion.
- virtual XnReferenceAxis generateReference (XnVector3D, XnVector3D, XnVector3D)

  Construye un nuevo eje de coordenadas a partir de 3 vectores otrtogonales.

3.2.1 Detailed Description

Aqui se describe la clase base "Joint".

3.2.2 Member Function Documentation

3.2.2.1 float Joint::angle (float *RL1,* float *RL2,* float *IML3*) [virtual]

P1[3],P2[3] Determina el angulo entre 3 puntos dadas las 3 lineas descritas por estos puntos El angulo a determinar viene dado por la linea imaginaria que esta opuesta a este angulo IML3 es el segmento contrario al angulo por determinar, el cual no existe, es IMaginario

3.2.2.2 XnReferenceAxis Joint::generateReference (XnVector3D *J1,* XnVector3D *J2,* XnVector3D *J3*) [virtual]

Construye un nuevo eje de coordenadas a partir de 3 vectores otrtogonales.

Este método genera un nuevo eje de referencia centrado en un Joint. PointNormal1 es un nuevo Joint para calcular una de las normales que será parte del marco de referencia

Utilizamos el struct XnReferenceAxis definido previamente para almacenar el nuevo marco de referencia

3.2.2.3 float Joint::getAngle (XnVector3D *J1,* XnVector3D *J2,* XnVector3D *J3*) [virtual]

L1,L2,L3(L3 es el opuesto del angulo a determinar)

J2 es la articulacion a la que se va a sacar el angulo.

Aunque se podria, no se crea una funcion getAngle aqui porque se necesitan pasar los parametros que usa el kinect para cada una de las posiciones, lo que se quiere es simplemente decir head.getAngle y que se obtenga el de la cabeza. Se pordria hacer, pero inicializando todos los joints y que se esten actualizando en tiempo real en la clasd Joint, para luego simplemente llamarlos desde las respectivas subclases igualando el valor actual del Joint en x,y,z a mi variable dentro de la subclase, luego como el objeto es de tipo Head, entonces el automaticamente sabe que las variables que debe pasar son las corerspondientes a los puntos que describen el angulo del cuello. Por ahora queda como mejora Determina directamente el angulo que hay en el segundo vector Se toman los tres vectores y se determina la distancia entre ellos, donde Imag3 es la distancia opuesta el J2

3.2.2.4 XnVector3D Joint::getNormalVector (XnVector3D *J1,* XnVector3D *J2,* XnVector3D *J3*) [virtual]

Genera el vector resultante de realizar el producto cruz entre los 2 vectores coplanares.

J2 es el punto de unión de los vectores J2->J1 y J2->J3, a los cuales se les sacará el vector normal. Generamos 2 vectores a partir de los cuales calcularemos el producto cruz, Vector1 X Vector2 = VectorNormal

Para Vector1 = J2->J1

Para Vector2 = J2->J3

Generamos el vector ortogonal VectorNormal

3.2.2.5 XnFloat Joint::getProyection (XnVector3D *Vector1,* XnVector3D *Vector2*) [virtual]

Obtiene la constante de la proyeccion.

Calculamos la proyección del Vector1 sobre el Vector2. Generamos el vector en el cual se va a guardar la proyección resultante

Calculamos el producto punto entre Vector1 y Vector2

Calculamos la norma del Vector2

Ahora calculamos la constante que multiplica al vector sobre el cual estamos proyectando para generar el nuevo vector proyección

3.2.2.6 XnVector3D Joint::getProyectionVector (XnVector3D *Vector1,* XnVector3D *Vector2*) [virtual]

Obtiene el vector resultante de proyectar un vector sobre otro.

Calculamos la proyección del Vector1 sobre el Vector2. Generamos el vector en el cual se va a guardar la proyección resultante

Calculamos el producto punto entre Vector1 y Vector2

Calculamos la norma del Vector2

3.2.2.7 float Joint::length (float *p1[3],* float *p2[3]*) [virtual]

Destructor del Objeto.

Determina el vector que va de P1 a P2 Determina el vector que va de P1 a P2

The documentation for this class was generated from the following files:

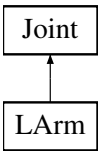
- joint.h
- joint.cpp

3.3 LArm Class Reference

Esta es la clase que describe el comportamiento para los brazos.

`#include <joint.h>`

Inheritance diagram for LArm:



Public Member Functions

- virtual float getElbowRoll (XnUserID, xn::UserGenerator)
  Obtiene el angulo que se determina a partir del kinect del codo.
- virtual float setElbowRoll (float, float &, float &)
  Genera el angulo necesario para mover el NAO.
- virtual float getShoulderRoll (XnUserID, xn::UserGenerator)
  Obtiene el angulo que se determina a partir del kinect del hombro.
- virtual float setShoulderRoll (float, float &, float &)
  Genera el angulo necesario para mover el NAO.
- virtual float getElbowYaw (XnUserID, xn::UserGenerator)
- virtual float **setElbowYaw** (float, float &, float &)

3.3.1 Detailed Description

Esta es la clase que describe el comportamiento para los brazos.

3.3.2 Member Function Documentation

3.3.2.1 float LArm::getElbowRoll (XnUserID *user*, xn::UserGenerator *guser*) [virtual]

Obtiene el angulo que se determina a partir del kinect del codo.

Obtiene el angulo real dado por el kinect del movimiento del codo.

3.3.2.2 float LArm::getElbowYaw (XnUserID *user*, xn::UserGenerator *guser*) [virtual]

Marco de referencia en un momento anterior con punto central sobre el codo

Obtenemos el vector elToHand (codo -> mano)

Obtenemos la proyeccion del vector elToHand sobre el eje Z de la referencia en el codo

Obtenemos el punto imaginario

Obtenemos la proyeccion sobre el plano XY, para ello obtenemos se forma entre la proyeccion y el vector elToHand

Obtenemos primer punto que es la proyeccion de la mano sobre el plano XY

Obtenemos el segundo punto sobre nuestro eje Y

Obtenemos el angulo entre la proyeccion entre el plano XY y el eje Y

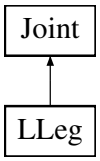
Determinamos el signo del angulo para saber si rotamos el codo en sentido horario o antihorario Obtenemos la proyeccion del vector proyVectorPlane sobre el eje X de la referencia en el codo

The documentation for this class was generated from the following files:

- joint.h
- larm.cpp

3.4 LLeg Class Reference

Inheritance diagram for LLeg:



Public Member Functions

- virtual float getHipRoll (XnUserID, xn::UserGenerator)

  Obtiene el angulo que se determina a partir del kinect del hombro.
- virtual float setHipRoll (float, float &, float &)

  Genera el angulo necesario para mover el NAO.
- virtual float getKneePitch (XnUserID, xn::UserGenerator)

  Obtiene el angulo que se determina a partir del kinect.
- virtual float setKneePitch (float, float &, float &)

  Genera el angulo necesario para mover el NAO.
- virtual float getAnklePitch (XnUserID, xn::UserGenerator)

  Obtiene el angulo que se determina a partir del kinect.
- virtual float setAnklePitch (float, float &, float &)

  Genera angulo para mover NAO.

3.4.1 Member Function Documentation

3.4.1.1 float LLeg::getHipRoll (XnUserID *user*, xn::UserGenerator *guser*) [virtual]

Obtiene el angulo que se determina a partir del kinect del hombro.

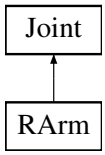
Obtiene el angulo real dado por el kinect del movimiento del codo.

The documentation for this class was generated from the following files:

- joint.h
- lleg.cpp

3.5 RArm Class Reference

Inheritance diagram for RArm:



Public Member Functions

- virtual float getElbowRoll (XnUserID, xn::UserGenerator)

  Obtiene el angulo que se determina a partir del kinect del codo.
- virtual float setElbowRoll (float, float &, float &)

  Genera el angulo necesario para mover el NAO.
- virtual float getShoulderRoll (XnUserID, xn::UserGenerator)

  Obtiene el angulo que se determina a partir del kinect del hombro.
- virtual float setShoulderRoll (float, float &, float &)

  Genera el angulo nececsario para mover el NAO.
- virtual float **getElbowYaw** (XnUserID, xn::UserGenerator)
- virtual float **setElbowYaw** (float, float &, float &)

3.5.1 Member Function Documentation

3.5.1.1 float RArm::getElbowRoll (XnUserID *user*, xn::UserGenerator *guser*) [virtual]

Obtiene el angulo que se determina a partir del kinect del codo.

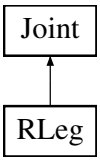
Obtiene el angulo real dado por el kinect del movimiento del codo.

The documentation for this class was generated from the following files:

- joint.h
- rarm.cpp

3.6 RLeg Class Reference

Inheritance diagram for RLeg:



Public Member Functions

- virtual float getHipRoll (XnUserID, xn::UserGenerator)

  Obtiene el angulo que se determina a partir del kinect del hombro.
- virtual float setHipRoll (float, float &, float &)

  Genera el angulo necesario para mover el NAO.
- virtual float getKneePitch (XnUserID, xn::UserGenerator)

  Obtiene el angulo que se determina a partir del kinect.
- virtual float setKneePitch (float, float &, float &)

  Genera el angulo necesario para mover el NAO.
- virtual float getAnklePitch (XnUserID, xn::UserGenerator)

  Obtiene el angulo que se determina a partir del kinect.
- virtual float setAnklePitch (float, float &, float &)

  Genera angulo para mover NAO.

3.6.1 Member Function Documentation

3.6.1.1 float RLeg::getHipRoll (XnUserID *user*, xn::UserGenerator *guser*) [virtual]

Obtiene el angulo que se determina a partir del kinect del hombro.

Obtiene el angulo real dado por el kinect del movimiento del codo.

The documentation for this class was generated from the following files:

- joint.h
- rleg.cpp

3.7 XnReferenceAxis Struct Reference

Public Attributes

- XnVector3D **NewX**
- XnVector3D NewY

Es un vector de coordenadas del nuevo eje en X.

- XnVector3D NewZ

  Es un vector de coordenadas del nuevo eje en Y.

The documentation for this struct was generated from the following file:

- definition.h