Calibrado : sincronización entre el sistema de coordenadas de Blender y de RoboMap

Autor:
Alberto Ramos Sánchez

Agosto 2020

Índice

1. Calibrado

El calibrado permite sincronizar el sistema de coordenadas de Blender con el propio de Robomap. Dentro del editor, el calibrado tiene la función de crear beacons estáticos en las posiciones que sean indicadas por el usuario. Este, manualmente, deberá situar los beacons que dibuja el calibrado en el escenario 3D que se haya creado.

En la figura 1 vemos que se han creado dos funciones asociadas al calibrado. El botón *calibrate* se encarga de comunicarse con el servidor para recibir la posición de *beacons* estáticos, y la función *drop all static beacons* elimina todos los *beacons* estáticos de la escena.

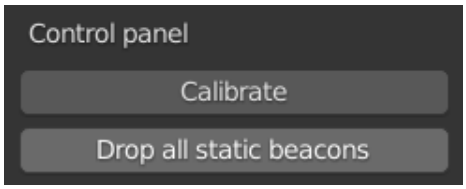

Figura 1: Funciones de calibrado.

1.1. Implementación

Como se explica en la memoria del TFT, el usuario puede añadir *beacons* en la posición que desee. Para incluir los *beacons* estáticos se han agregado las categorías `STATIC_ULTRASOUND_BEACON` y `STATIC_BLUETOOTH_BEACON` en la propiedad *object_type*, en el fichero `utilities/object_properties.py` (Ver apartado 9.3, Categorización de objetos en Desarrollo de módulo de comunicación en Blender para la interoperación con plataforma robótica, TFT, Julio 2020).

Estos beacons son creados del mismo modo pero se ha bloqueado la posibilidad de que sean eliminados o movidos por el usuario, para evitar que sea modificado el calibrado accidentalmente durante la creación de escenarios.

El calibrado se implementa en `CalibrateOperator` (`wm.calibrate`). Este se apoya en la función `receive_calibration_data` de `ConnectionHandler`↩ , que realiza todas las operaciones de comunicación con el servidor y lectura de la clase `Buffer`.

El eliminado de *beacons* estáticos se realiza en el operador `DropAllStaticBeacons`↩ , que busca cualquier tipo de *beacon* estático en la escena y lo borra (`STATIC_ULTRASOUND_BEACON`↩ y `STATIC_BLUETOOTH_BEACON`). Al inicio del calibrado también se eliminan todos los *beacons* estáticos existentes.

2. Otros cambios realizados

Corrección del cerrado del hilo

En el último párrafo de la página 65, dentro del apartado 9.7.2 (Esquema de comunicación), hay que incluir que se espera a cerrar el hilo con un *join* antes de cerrar y eliminar el objeto *socket*, para evitar que la operación *recvfrom* del módulo *socket* de Python (que queda bloqueada durante un tiempo establecido), ejecute operaciones sobre un *socket* cerrado. Con esto se corrige un error que se producía al salir del modo *Robomap*.

Reconocimiento de paquetes

Para los paquetes que requieren reconocimiento, ahora es el hilo encargado de recibir paquetes el que envía el paquete de reconocimiento —y no el hilo de *Blender* al recoger los paquetes del *Buffer*—, para así asegurar que se realiza lo más rápido posible.