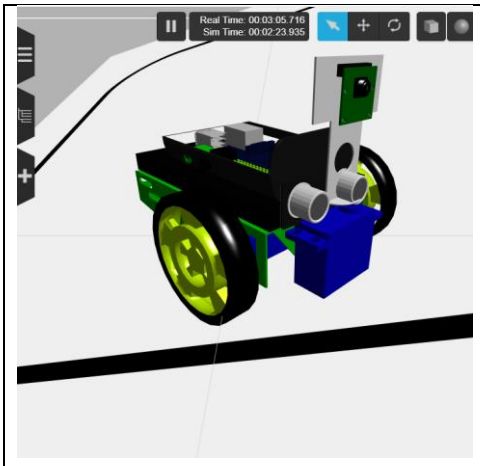
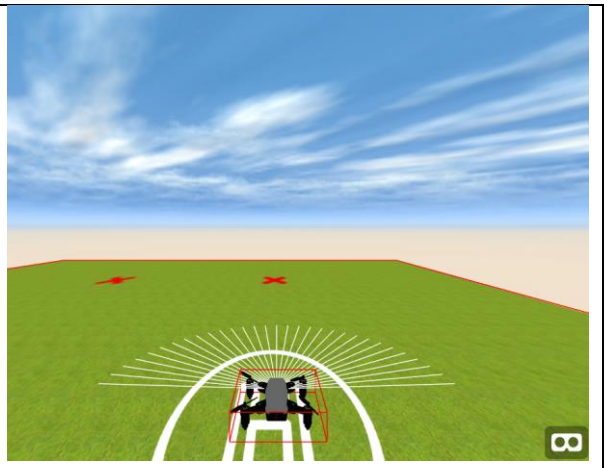
Diseño Software – Grado en Ingeniería de Robótica Software

Proyecto: Programación orientada a objetos (II)

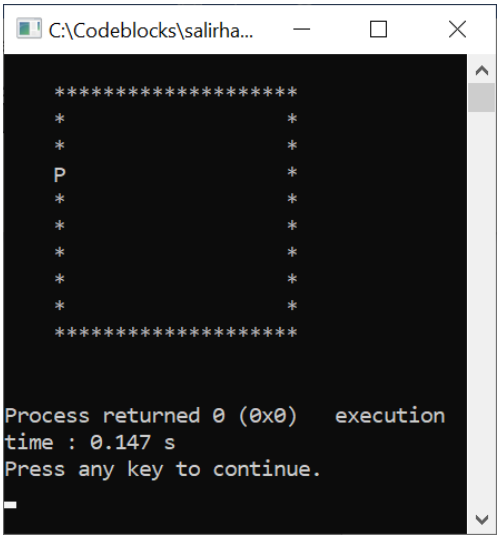
La empresa Io-Rovout quiere desarrollar un programa para comprobar mediante simulaciones el comportamiento de sus dos robots educativos PiBot y Tello.

	
PiBot	**Tello**

En concreto, el programa debe simular las acciones realizadas por cada robot para conseguir salir de una habitación, utilizando sus sensores de ultrasonidos y su cámara incorporada, enfocada al lateral izquierdo del robot.

El programa debe realizar las siguientes acciones:

1. Imprimirá en la pantalla un rectángulo de dimensiones determinadas que representa una habitación vista desde arriba (tamaño máximo de 20 filas por 60 columnas, interpretada como una matriz de caracteres). Las paredes de la habitación se representarán empleando caracteres '*'. A continuación dibujará un carácter 'P' para representar una puerta. La situación de la puerta se calculará de forma aleatoria, siendo posible cualquier punto de las 4 paredes de la habitación, excepto las esquinas. Aquí vemos un ejemplo de habitación 10 filas por 20 columnas con la puerta en la pared de la izquierda:



2. Imprimirá en pantalla, en una posición aleatoria siempre dentro de la habitación, uno de los 2 robots citados (el tipo de robot será seleccionable por el usuario del programa):

    a. El robot PiBot se representará empleando los caracteres '^', 'v', '>', o '<', según se encuentre mirando hacia el norte, sur, este u oeste. La orientación inicial del robot será aleatoria.

    b. El drone Tello se representará con el carácter 'x' si se encuentra en el suelo y con el carácter 'X' cuando esté volando. Inicialmente estará en el suelo y su orientación inicial será aleatoria.

3. El programa irá simulando los movimientos del robot con el objetivo de encontrar la puerta de la habitación, a razón de un movimiento por segundo, con las siguientes normas:

    a. En cualquier instante, el robot habrá encontrado la puerta si ésta se encuentra adyacente justo a su izquierda (posiciones relativas siempre desde el punto de vista del robot). El robot debe comprobarlo antes de realizar cada movimiento utilizando su cámara incorporada en el lateral.

    b. Cuando la orientación del robot sea Norte, el robot avanzará en línea recta en dirección Norte hasta llegar a la pared superior (o encontrar la puerta situada a su izquierda, en cuyo caso se detendrá junto a ella). Irá comprobando a cada paso si ha llegado a la pared con su sensor (frontal) de ultrasonidos. Al llegar a dicha pared, girará a la derecha, con lo que su orientación pasará a ser Este

    c. Cuando la orientación del robot sea Este, el robot avanzará en dirección Este hasta llegar a la pared de la derecha (o encontrar la puerta situada a su izquierda, en cuyo caso se detendrá junto a ella). Irá comprobando a cada paso si ha llegado a la pared con su sensor (frontal) de ultrasonidos. Al llegar a dicha pared, girará la derecha, con lo que su orientación pasará a ser Sur

d. Cuando la orientación del robot sea Sur, el robot avanzará en dirección Sur hasta llegar a la pared inferior (o encontrar la puerta situada a su izquierda, en cuyo caso se detendrá junto a ella). Irá comprobando a cada paso si ha llegado a la pared con su sensor (frontal) de ultrasonidos. Al llegar a dicha pared, girará la derecha, con lo que su orientación pasará a ser Oeste

e. Cuando la orientación del robot sea Oeste, el robot avanzará en dirección Oeste hasta llegar a la pared de la izquierda (o encontrar la puerta situada a su izquierda, en cuyo caso se detendrá junto a ella). Irá comprobando a cada paso si ha llegado a la pared con su sensor (frontal) de ultrasonidos. Al llegar a dicha pared, girará la derecha, con lo que su orientación pasará a ser Norte

Las principales diferencias entre los 2 robots son:

1. PiBot avanza a una posición por segundo, mientras que Tello puede avanzar a 1 o a 2 posiciones por segundo, dependiendo de su nivel de batería (avanzará a 2 posiciones por segundo en los 25 primeros movimientos, después pasará a avanzar a 1 posición por segundo).

2. La cámara del Tello tiene mayor resolución que la del PiBot, por tanto es capaz de detectar la puerta si ésta se encuentra a 2 posiciones a su izquierda (el PiBot sólo la detecta si se encuentra a 1 posición a su izquierda)

3. El drone Tello puede volar, el PiBot no

Aquí se muestra un ejemplo de simulación finalizada con el PiBot:

Aquí se muestra un ejemplo de simulación finalizada con el Tello:

La empresa Io-Rovout contactó con Ayam Destruct para pedirle ayuda en el desarrollo, pero se había prejubilado y estaba ilocalizable en algún lugar de Panamá, disfrutando de los ingentes retornos económicos que le proporcionaron sus flamantes desarrollos con programación modular. No obstante, contactaron con su primo segundo Ayam Deobyet, mucho más joven que Destruct, y con otro enfoque diferente a la hora de afrontar el desarrollo de aplicaciones.

Deobyet proporcionó varias clases como ayuda en el desarrollo, a condición de que le prometieran que, utilizasen o no su código, la aplicación final contendría **clases abstractas**, **herencia** y **poliformismo** (Deobyet es un fanático de estos 3 conceptos). También les proporcionó un pseudocódigo indicando la secuencia de acciones que debería realizar un robot para encontrar la puerta de la habitación, a cambio de un billete de avión a Panamá para visitar a su primo Destruct (al parecer estaba ilocalizable, pero no para su primo).

En este proyecto debemos completar el desarrollo de la aplicación descrita anteriormente. El código de partida son las clases proporcionadas por el susodicho Deobyet. Este código puede ser modificado/ampliado/no utilizado en absoluto a voluntad.

Además de funcionar correctamente, el programa debe cumplir las siguientes normas de calidad con el objetivo de que el código final sea lo más modular/extensible/mantenible/reutilizable posible:

1. Dentro de la función “main()” sólo puede haber declaraciones de y asignaciones a variables locales y llamadas a otras funciones

2. El fichero “main.cpp” no puede contener ninguna otra función aparte de la función “main()”

3. Las distintas funciones/métodos a desarrollar deben contener un máximo de 20 líneas de código (sin contar los comentarios)

4. Las variables globales están prohibidas

5. Las estructuras de datos como tales están prohibidas, el programa debe ser diseñado usando exclusivamente orientación a objetos

6. El idioma a utilizar tanto en los mensajes de salida hacia el usuario como en los comentarios del código debe ser único

7. Se prohíbe el uso de las sentencias “continue” y “break” en los bucles del programa

8. Cada función debe tener un único punto de retorno (una sola aparición de la sentencia “return”)

9. Se deben definir las constantes necesarias para aumentar la legibilidad del código y reducir el tiempo de desarrollo ante posibles cambios de valor de las mismas