Introducción

Implementación en Python de C-Robots (https://tpoindex.github.io/crobots/).

PyRobots es un juego para programadores. A diferencia de los juegos de tipo arcade, los que requieren el input humano para controlar algún objeto, toda la estrategia de PyRobots debe ser completada antes de que el juego comience. La estrategia del juego se condensa en un programa en Python que debes diseñar y escribir. Tu programa controla un robot cuya misión es buscar, seguir y destruir otros robots, cada uno ejecutando diferentes programas. Cada robot está igualmente equipado, y pueden competir hasta 4 robots al mismo tiempo.

Cada robot dispone de funciones para scanear enemigos, iniciar y detener el motor, disparar cañones, etc. Luego de que se inician los robots, cada uno ejecutando un programa, se puede observar la batalla. Robots moviéndose, misiles volando y explotando, y cierta información de status en tiempo real.

Usuarios

PyRobots sólo puede jugarse registrándose primero al sistema. El juego permite el registro de nuevos usuarios utilizando un email, nombre de usuario e imagen de avatar. Cada usuario puede crear tantos robots como desee para utilizar en el juego.

Tipos de juego

PyRobots se puede jugar de dos maneras: simple, en el que se puede observar el juego en pantalla completa, o múltiple, en el que se corren una cantidad de partidos y se muestra únicamente la cantidad de partidos que ganó cada jugador. Por defecto es juego simple. Juego múltiple está pensado para ver cómo performa cada programa en promedio.

El juego

Campo de batalla

El campo de batalla es un cuadrado de 1000 x 1000 mts. El campo está rodeado por una pared en todo su perímetro, así que cuando un robot choca contra la pared, incurre en daños.

Las coordenadas (0, 0) corresponden a la esquina inferior izquierda. Las coordenadas (999, 999) corresponden a la esquina superior derecha.

Las direcciones son las siguientes:

- Derecha (este): 0 grados
- Arriba (norte): 90 grados
- Izquierda (oeste): 180 grados
- Abajo (sur): 270 grados

Ofensiva

Las principales armas ofensivas son el cañón y el escáner. El cañón tiene un rango de 700 metros. Se puede disparar un número ilimitado de misiles, pero luego de cada disparo, el cañón necesita un tiempo para recargarse.El cañón está montado en una torre independiente, y por lo tanto puede disparar en cualquier dirección (0 a 359 grados), más allá de la dirección de vuelo del robot.

El escáner es un dispositivo óptico que puede escanear en cualquier dirección (0 a 359 grados), con una resolución de 0 a ±10 grados.

Defensiva

La única defensa disponible son el motor y el status. El motor se puede activar en cualquier dirección (0 a 359 grados), en velocidades desde 0% a 100%. Una velocidad de 0% detiene el motor. Hay factores de aceleración y desaceleración. Se puede girar a velocidades iguales o menores a 50%, hacia cualquier dirección.

El programa puede acceder a cierta información de status en todo momento: porcentaje de daño, posición y velocidad actual.

Eliminando enemigos

Un robot se considera muerto cuando el porcentaje de daño llega al 100%. El daño se infringe de las siguientes maneras:

- Colisión con otro robot: ambos reciben un daño de 2%
- Colisión con la pared: 2%
- Explosión de misil a menos de 40 metros: 3%
- Explosión de misil a menos de 20 metros: 5%
- Explosión de misil a menos de 5 metros: 10%

El daño es acumulativo y no se puede reparar. Sin embargo, un robot no pierde movilidad ni potencia de disparo por tener daños elevado. En otras palabras, un robot con 99% de daño performa exactamente igual que un robot sin daño.

Rondas

Un juego consiste en a lo sumo 10000 rondas. En una ronda, todos los robots realizan simultáneamente ciertas acciones (leer el status, disparar el cañón, apuntar el escáner, accionar el motor).

El programa decide qué acciones realizar, y al final de la ronda se ejecutan todas las acciones. Por ejemplo, un robot puede decidir disparar el cañón a 58% a una distancia de 534 mts, y acelerar el motor a 60% de velocidad. Pero recién al final de la ronda, se van a ejecutar estas acciones.

Crear un robot

Un robot es un archivo que contiene una clase que hereda de la clase **Robot**. La clase y el archivo deben tener el mismo nombre. Por ejemplo, si la clase se llama SuperMegaRobot, el archivo deberá ser llamado super_mega_robot.py. Todos los robots deberán ser guardados en el subdirectorio **robots**.

La clase debe implementar al menos 2 métodos: **initialize** y **response**:

```
class SuperMegaRobot(Robot):

  def initialize(self):
      ...

   def respond(self):
      ...
```

initialize se ejecuta exactamente una vez cuando el robot es creado. Se puede usar para iniciar variables por ejemplo.

respond se ejecuta exactamente una vez por robot en cada ronda. En este método se puede usar cualquiera de los métodos públicos de la clase Robot, que se describen a continuación. Usar un método interno (los que comienzan con _) está prohibido.

Cañón:

is_cannon_ready(): Cuando se dispara un misil, el cañón requiere un tiempo para recargarse. Se puede usar esta función para chequear si el cañón está completamente recargado.

cannon(degree, distance): Cuando se llama a este método, se prepara el cañón para disparar. Si se llama a este método dos veces seguidas, sólo la última tiene efecto. Recién se ejecuta el disparo al finalizar la ronda.

Escáner:

point_scanner(direction, resolution_in_degrees): con este método se puede apuntar el escáner en cualquier dirección. Pero el resultado del escaneo estará disponible en la siguiente ronda, a través del siguiente método.

scanned(): devuelve el resultado del escaneo de la ronda previo. Devuelve la distancia al robot más cercano en la dirección apuntada.

Motor:

drive(direction, velocity): establece una nueva dirección y velocidad para el robot. Al igual que **cannon**, si se llama dos veces seguidas, sólo tiene efecto la última, al finalizar la ronda.

Status:

Estos métodos se pueden llamar en cualquier momento:

get_direction()

get_velocity()

get_position()

get_damage()