Tarea 1

Programación Orientada a Objetos

1 Problema

En esta tarea deberán desarrollar un juego simulador de batallas entre robots, tanto batallas individuales como batallas por equipo. Su juego deberá permitir simular distintos tipos de competiciones, como ligas (todos contra todos) o playoff (eliminación directa). Habrán robots de distintos tipos los cuáles se diferenciaran por los ataques que poseen, como también por sus habilidades.

Adicionalmente, su programa deberá permitir que el usuario escoja un robot o un equipo para que sea el usuario el que lo controle escogiendo los ataques cuando corresponda. En esta modalidad, su programa deberá permitir guardar una competición para luego recomenzarla en el mismo estado en que fue guardada.

2 Detalles

2.1 Robot

Un robot tiene un nombre, una energía máxima, y un conjunto de ataques y habilidades. Los ataques tienen un nombre y las siguientes características:

- *type* → a distancia (*long*), ataque directo con arma (*sword*), ataque cuerpo a cuerpo (*hand*).
- *objective* → define si el ataque es directo a un robot (*robot*), o se realiza a todo el equipo rival (*team*).
- *damage* → indica la cantidad de daño que genera el ataque. El daño se representa en resta de energía en el objetivo y siempre será un número entero mayor que 0.
- *precision* → es un porcentaje que indica la cantidad de veces que el ataque acierta su objetivo. Se representa como número entero entre 10 y 100.
- *recharge* → indica cuántos turnos deben esperarse antes de poder volver a usar este ataque. Por ejemplo, si el valor es 0 significa que no hay espera y el ataque puede usarse en cualquier turno, mientras que si el valor es 2, entonces significa que luego de usar este ataque hay que esperar 2 turnos de ataque antes de poder volver a usarlo.

Por su parte, las habilidades son condiciones especiales que pueden activarse durante la batalla. Las habilidades, además de tener un nombre, poseen las siguientes características:

- *trigger* → indica que debe ocurrir durante la batalla para que la habilidad se active. En esta línea, existirán distintos mecanismos de activación:
  - *energy* → activación cuando la energía del robot baja de un cierto umbral.
  - *attack_type* → activación al recibir un tipo de ataque determinado.
  - *attack_team* → activación ante ataques cuyo objetivo es el equipo.
  - *turns* → activación luego de una cierta cantidad de turnos de ataque.

- *duration* → se refiere a la cantidad de rondas en que la habilidad estará activa.
- *objective* → indica si la habilidad se aplica solo al robot que la posee (*robot*), o a todo el equipo (*team*).
- *effect* → los efectos de la habilidad pueden ser tanto en ataque como en defensa:
  - *shield_type* → reducción del daño de un tipo de ataque en un porcentaje definido.
  - *steroids* → aumento en un porcentaje definido del daño de un tipo de ataque definido.
  - *hawk* → aumento en un porcentaje de la precisión de un tipo de ataque.
  - *shifting* → reducción de la precisión de un tipo de ataque recibido en un porcentaje definido.
  - *fast_recharge* → eliminar los turnos de recarga.
  - *magic* → aumento de una cierta cantidad de energía.

2.2 Competición

Su programa debe permitir realizar distintos tipos de competiciones:

- Liga → todos batallan contra todos. Se genera una tabla de posiciones en base a la cantidad de victorias, en caso de empate en victorias, quedará arriba aquel que haya ganado la batalla entre los robots/equipos empatados.
- Playoff → batallas de eliminación directa, el que gana la batalla continua en competición avanzando a la siguiente ronda. Así hasta que queden dos, los cuáles disputan la final, siendo campeón quien gana esa batalla.
- Torneo → se hacen grupos en los cuáles batallan todos contra todos, clasificando los dos mejores a unos playoff. El tamaño de los grupos es configurable entre 3 y 8.

Todos los tipos de competiciones pueden ser entre robots o entre equipos de robots.

2.3 Batalla

Las batallas son en modo de turnos, donde uno de los contrincantes parte realizando su primer ataque, y luego le corresponde al otro. El ataque a realizar debe ser escogido aleatoriamente entre los ataques disponibles del robot/equipo. En el caso de batallas por equipo, aplica la restricción de que no se puede utilizar en dos turnos seguidos ataques del mismo robot.

Una vez se escoge el ataque este debe ser aplicado al robot o equipo contrincante. Ahora bien, antes de aplicar el daño se debe revisar si se activa alguna habilidad producto de este ataque, así como revisar si las habilidades activas modifican los daños del ataque recibido. De esta forma, es importante que las habilidades se activan previo a calcular el daño, y que ese turno se considera como el primer turno donde la habilidad se encuentra activa.

Cuando un robot se queda sin energía, ese robot no puede seguir participando en la batalla, incluidas sus habilidades. La batalla la gana quien logre sacar de la partida a todos los robots contrincantes.

Antes de comenzar una batalla, todos los robots participantes vuelven a su nivel máximo de energía.

2.4 Modos de juego

Deberán haber dos modos de juego posibles:

- Automático → se realiza la competición de forma automática, se muestra en la consola cada partida con sus resultados, siguiendo una batalla después de otra hasta finalizar.
- Participativo → el usuario escoge un robot/equipo al cual controlar, de tal forma que será el usuario quien escoja los ataques de su competidor. En las batallas donde participe el robot/equipo del usuario, se deberá mostrar en consola el estado de todos los robots participantes, indicando la energía que les queda, y las habilidades que se encuentran activadas. Cuando se utiliza este modo, el usuario podrá guardar la partida para continuarla después (esto implica que puede terminar el programa y continuarla una próxima vez que lo ejecute, sin perder información).

2.5 Reportes

El usuario podrá definir previo a ejecutar una competición, los reportes que querá visualizar al finalizar la competición. Estos reportes podrán ser gráficos (guardados como imagen) o archivos en formato CSV que representen tablas. Es así comod deberán estar disponibles los siguientes reportes:

- Tabla de victorias y derrotas por robot/equipo, incluyendo la cantidad de turnos totales disputados.
- Para cada robot/equipo, un gráfico de barras con la cantidad de veces que se utilizó cada ataque durante la competición.
- Para cada robot/equipo, un gráfico de barras con la cantidad de veces que se activo cada habilidad.
- Tabla con cada batalla disputada, indicando el ganador, la cantidad de turnos, y la energía total con la que quedñi el ganador de la batalla.

3 Programa

En base a los detalles anteriores su programa deberá permitir crear una competición, para lo cuál deberá pedir al usuario que escoja el formato de competición deseado, junto a los parámetros que esta requiera. A su vez, deberá pedir un archivo en formato JSON desde el cuál se deberán cargar todos los robot/equipos conteniendo toda su información. Un robot se representará con una estructura como la siguiente:

```
{
  "robots": [
    {
      "name": "fuego",
      "energy": 75,
      "attacks": [
        {
```

```
            "name": "arco",
            "type": "long",
            "objective": "team",
            "damage": 20,
            "precision": 80,
            "recharge": 1
          },
          {
            "name": "cabezazo",
            "type": "hand",
            "objective": "robot",
            "damage": 15,
            "precision": 95,
            "recharge": 0
          }
        ],
        "skills": [
          {
            "name": "posi n",
            "trigger": "energy",
            "trigger_value": 30,
            "duration": 2,
            "objective": "team",
            "effect": "magic",
            "effect_value": 10
          },
          {
            "name": "fuerza lenta",
            "trigger": "turns",
            "trigger_value": 3,
            "duration": 1,
            "objective": "robot",
            "effect": "steroids",
            "effect_value": 30
          }
        ]
      }, {...}
    ]
}
```

Cuando el archivo represente una lista de equipos, entonces la estructura del archivo JSON será la siguiente:

```
{
  "teams": [
    {
```

```
      "name": "teamA",
      "robots": [
        {...}, {...},
      ]
    },
    {
      "name": "teamB",
      "robots": [
        {...}, {...},
      ]
    }
  ]
}
```

Una vez cargados los competidores de la competición, el programa deberá preguntarle al usuario la modalidad de juego y los reportes que quiere activar. En caso de que la modalidad de juego escogida sea *Participativo*, el programa debe pedirle al usuario que seleccione su robot/equipo mostrandole todos los disponibles en conjunto con sus principales valores.

Finalmente en su programa deberá incorporar un nuevo gatillador y un nuevo efecto para las habilidades, además de dos nuevos reportes.

4 Entregas

La tarea se dividirá en 2 entregas:

- Entrega Parcial 1 (EP1) [1.5 ptos] → jueves 29 de agosto a las 15:30, mostrar en horario de laboratorio.
  - Competición solo liga.
  - Robot sin habilidades.
  - Solo batallas individuales (no por equipos).
  - Sólo modo automático.
  - Reporte de tabla de victorias y derrotas, y de gráfico con la cantidad de usos de cada ataque.
- Entrega Final [5.5 ptos] → miércoles 11 de septiembre a las 23:00 por github.
  - Todas las funcionalidades.
  - Deberán incorporar a su repositorio un documento con instrucciones de ejecución y uso, así como también donde se explique los componentes definidos por ustedes para su juego (el nuevo gatillador, efecto y reportes).
  - También el modelo de clases debe estar en el repositorio, representando fielmente el código realizado.

5 Consideraciones

Las siguientes son consideraciones y restricciones para el desarrollo de su tarea. El no cumplimiento de una o más de estas implicará que su tarea NO sea revisada y se evalúe con nota 1.0. Estas consideraciones se deben aplicar para TODAS las entregas, a menos que se indique lo contrario.

- La tarea debe estar completamente definida en base al paradigma orientado a objetos.
- Deberán utilizar el sistema de tipos de Python.
- Las competiciones, reportes y habilidades deben cargarse al momento de ejecutar el programa (*reflection*, solo aplicable en entrega final).
  - No puede haber un `if-else` sobre tipos.
  - Cada competición, reporte y habilidad debe ser una clase diferente y estar en un archivo diferente.
  - Deben tener una carpeta independiente con los archivos de las clases de cada uno.
- Todo el trabajo de su tarea deberán hacerlo utilizando un repositorio en github según las instrucciones que les entregará el profesor. La información del repositorio se utilizará tanto para las revisiones, como para evaluar si todos los integrantes del grupo trabajaron adecuadamente.