Macro and Micro Manual

> Sioux Weekend of Code 2017

Macro

> Control your UFOs to conquer planets and gain credits.

Starting a macro game

You can start the macro game from the command line w ith the follow ing arguments:

```
MacroEngine.exe playername /path/to/macro/script/ /path/to/micro/script ...
```

The first argument is the player name, the second argument is the directory w hich contains your Macro script and a runCommand.txt file, and the third argument is the directory w hich contains your Micro script and a runCommand.txt file. Macro w ill read the command that has to be executed from the runCommand.txt file.

Example command line:

```
MacroEngine.exe jesper C:/scripts/macro/ C:/scripts/micro ferdi C:/scripts/macro/ C:/scripts/micro
```

Example runCommand.txt

```
python C:/scripts/macro/script.py
```

Conquering Planets

The goal of Macro is to command your UFOs to move to planets and conquering them.
You can conquer planets that have already been conquered by other players.
If you try to conquer a planet w hile one or more enemy UFOs are in nearby the planet, you w ill have to battle (Micro) for the right to conquer the planet.
The w inner of the battle w ill conquer the planet and in case of a draw, no one w ill conquer the planet.

See commands: conquer, moveToPlanet, and moveToCoord.

Credits

When a planet is conquered you gain credits. These are used to buy more UFOs. The player w ith the most credits at the end of the game w ill w in so be mindful of how you are spending your credits.

If you have at least one planet conquered, your base income is a **1500** credits per second. On top of that you get an additional amount credits for

the total number of planet you have conquered per second.
Your total income per second is calculated w ith the follow ing formula:

```
Bi + ((Bp - P * S) * S)
where
    Bi = base income (1500 credits)
    Bp = base income per planet (225 credits)
    P = penalty per planet (3 credits)
    S = number of conquered planets (up to a maximum of 50) + 3
```

See commands: buy

Receiving Game State from Macro

On each tick of the game, Macro w ill send the **game state** to all Macro scripts. Macro w ill w ait for **250 milliseconds** during each tick.

The game state consists of the follow ing:

- List of solar systems, each containing a list of planets
- List of players, each containing a list of ufos

Example game state:

```
{
    "id": 1,
    "name": "Match1",
    "tick": 0,
    "solarSystems": [
        {
            "id": 1,
            "name": "S1",
            "coords": {
                "x": 50,
                "y": 15
            },
            "planets": [
                {
                    "id": 1,
                    "name": "S1P1",
                    "orbitDistance": 10,
                    "orbitRotation": 10,
                    "ownedBy": 1
                }
            ]
        }
    ],
    "players": [
        {
            "id": 1,
            "name": "Player1",
            "credits": 9001,
            "ufos": [
                {
                    "id": 1,
                    "inFight": true,
                    "coord": {
                        "x": 25,
                        "y": 75
                    }
                }
            ]
        }
    ]
}
```

Sending Commands to Macro

You can send commands to Macro by writting them to `stdout`.
There are 5 types of Macro commands:

- conquer
- fight
- moveToPlanet
- moveToCoord
- buy

Macro will wait for **250 milliseconds** (a tick) before processing all commands. Within that time you can send as many commands as you like and they will be queued in Macro for processing. At the end of the tick, Macro will process all the commands

conquer

In order to conquer a planet you have to have at least one UFO that is within a radius of **256** "space meters" of the planet that you want to conquer, and you have to send the `conquer` command with the ID of that planet.

If there are other enemy UFOs in the radius of the planet, a Micro battle will start with all the UFOs in the radius of the planet.
UFOs that are already in a battle will not be included.

Example: lets say you have UFO at (45,75) and you want to conquer planet 42 which is located at (50,60). You can calculate the distance with the formula sqrt((50-45)(50-45)+(60-75)(60-75)) = 15 which is within the radius of **256**.

Sending the following `conquer` command will let you conquer planet 42.

```
{
    "command": "conquer",
    "planetId": 42,
}
```

[Pro-tip] *Planet Sniping*: If you try to conquer a planet where there are other UFOs in its radius that are already in a fight for that planet, you can take that planet without starting a battle and you can keep it until the other UFOs are done fighting (the winner will take the planet).
If the battle is a draw, you get to keep the planet.

fight

In order to fight bots in space, you can send the `fight` command with the ID of the UFO you want to fight.

```
{
    "command": "fight",
    "ufoId": 1,
}
```

moveToPlanet

To move your UFO(s) tow ards a planet you can send the `moveToPlanet` command w ith the list of UFOs that you w ant to move and the ID of the planet you w ant to move to.

Sending the follow ing `moveToPlanet` command w ill start moving UFOs 1, 2, and 3 tow ards planet 42.

```
{
    "command": "moveToPlanet",
    "ufos": [1, 2, 3],
    "planetId": 42
}
```

m oveToCoord

To move your UFO(s) tow ards a specific coordinate you can send the `moveToCoord` command w ith the list of UFOs that you w ant to move and the coordinate you w ant to move to.

Sending the follow ing `moveToCoord` command w ill start moving UFOs 1, 2, and 3 tow ards the coordinate (75,25).

```
    {
        "command": "moveToCoord",
        "ufos": [1, 2, 3],
        "coord": {
            "X": 75,
            "Y": 25
        }
    }
```

If you send multiple `moveToCoord` commands during one tick

buy

To buy additional UFOs you can send the `buy` command w ith the amount of UFOs that you w ant to buy and the ID of the planet w here you w ant your new UFOs to spaw n. The price of a UFO is **50000** credits.

There are some constraints:

1. The buy command w ill be ignored if you try to buy more UFOs that you can afford.
2. The buy command w ill be ignored if you try to buy UFOs on a planet that you have not conquered.

If you have no conquered planets, you can still buy UFOs using **-1** as the planet ID. This w ill spaw n the UFOs at random coordinates in the universe.

Sending the follow ing `buy` command w ill spaw n 3 UFOs at planet 42.

```
    {
        "command": "buy",
        "amount": 3,
        "planet": 42
    }
```

[Pro-tip] *Second Life*: If all your UFOs have been destroyed and you do not have enough credits to buy more UFOs or any conquered planets, you are allowed to buy **one** UFO and go to negative credits.

Macro Commands Reference

conquer

Properties

- `command`: Type of command (conquer)
- `planetId`: ID of planet you want to conquer

```
{
    "command": "string",
    "planetId": "int",
}
```

fight

Properties

- `command`: Type of command (fight)
- `ufoId`: ID of UFO you want to fight

```
{
    "command": "fight",
    "ufoId": 1,
}
```

moveToPlanet

Properties

- `command`: Type of command (moveToPlanet)
- `ufos`: List of UFOs you want to move
- `planetId`: ID of planet you want to move to

```
{
    "command": "string",
    "ufos": "[int]",
    "planetId": "int"
}
```

moveToCoord

Properties

- `command`: Type of command (moveToCoord)
- `ufos`: List of UFOs you want to move
- `coord`: (X,Y) coordinate you want to move to

```
{
    "command": "string",
    "ufos": "[int]",
    "coord": {
        "x": "int",
        "y": "int"
    }
}
```

buy

Properties

- `command` : Type of command (buy)
- `amount` : Quantity of UFOs that you w ant to buy
- `planetId` : ID of planet w here your UFOs w ill spaw n

```
    {
        "command": "string",
        "amount": "int",
        "planet": "int"
    }
```

Micro

> Control your UFOs and battle against enemy UFOs.

Game Mechanics

Players control UFOs using 2 commands: `move` and `shoot` .

Moving

The `move` command moves the UFO in the specified `direction` at the specified `speed` .
The movement is based on the polar coordinate system, `direction` being the *angle* and `speed` being the *radius*.

Parameters:

- `direction` : Value betw een **0** and **360**.
- `speed` : Value betw een **0** and **10**.

Shooting

The `shoot` command shoots a laser from the origin of the UFO in the specified `direction` .
Unlike moving, shooting has a fixed projectile speed.
The movement is based on the polar coordinate system, `direction` being the *angle* and the projectile speed w hich is alw ays **15** (fixed speed).
Shooting also has a *cooldown* of **1** seconds meaning that you can only shoot once per second.
Each shot does **25** points of damage. There is no friendly fire.

Parameters:

- `direction` : Value betw een: **0** and **360**.

Hitpoints

A UFO starts w ith **100** `hitpoints` and is destroyed w hen it's `hitpoints` reaches `0` .

Position

All UFOs have a `position` "**x,y**". This represents the UFO's current position on the arena.
The arena has a specified `width` and `height`. The top left corner of the arena represents "**0,0**" and the bottom right corner of the arena represents "**width,height**".
Your UFO can only move within the bounds of the arena. There is no collision between UFOs.

The Laser Fence

The arena is surrounded by a laser fence. If any UFO touches this fence, they will immediately be destroyed (**0** `hitpoints`).
In order to keep the games short, the arena will start shrinking after a fixed amount of time and will keep shrinking until the battle is over.

Winning Condition

Last man standing: if all your UFOs are destroyed you lose.

Scripting

Game Loop

The game loop of any basic micro UFO is as follows:

1. Read `JSON` formatted game state (input) from micro using `stdin`
2. Script logic (where the magic happens)
3. Write `JSON` formatted commands (output) to micro using `stdout`

Note: The input and output are in JSON format

Python example:

```
while True:
    input = input()
    # Script logic
    print(output)
```

Reading input from Micro

The input received from micro has the following format:

```
{
    "tick": "integer",
    "arena": {
        "height": "integer",
        "width": "integer",
        "shrinkRate": "integer",
        "shrinkThreshold": "integer"
    },
    "player": "string",
    "players": [
        {
            "id": "integer",
            "name": "string",
            "ufos": [
                {
                    "id": "integer",
                    "name": "string",
                    "hitpoints": "float",
                    "position": {
                        "x": "float",
                        "y": "float"
                    }
                }
            ]
        }
    ],
    "projectiles": [
        {
            "position": {
                "x": "float",
                "y": "float"
            },
            "direction": "float",
        },
    ]
}
```

Input example:

```
{
    "tick": 10,
    "arena": {
        "height": 1000,
        "width": 1000,
        "shrinkRate": 10,
        "shrinkThreshold": 2
    },
    "playerId": 0,
    "playerName": "player0",
    "players": [
        {
            "id": 0,
            "name": "player0",
            "ufos": [
                {
                    "id": 0,
                    "name": "ufo0",
                    "hitpoints": 20,
                    "position": {
                        "x": 25,
                        "y": 25
                    }
                },
                {
                    "id": 1,
                    "name": "ufo1",
                    "hitpoints": 10,
                    "position": {
                        "x": 25,
                        "y": 25
                    }
                }
            ]
        },
        {
            "id": 1,
            "name": "player1",
            "ufos": [
                {
                    "id": 0,
                    "name": "ufo0",
                    "hitpoints": 15,
                    "position": {
                        "x": 75,
                        "y": 75
                    }
                },
                {
                    "id": 1,
                    "name": "ufo1",
                    "hitpoints": 15,
                    "position": {
                        "x": 50,
                        "y": 75
                    }
                }
            ]
        }
    ],
    "projectiles": [
        {
            "position": "23,34",
            "direction": "32",
        },
        {
            "position": {
                "x": 25,
                "y": 25
            },
            "direction": 32
        }
    ]
}
```

Writing output to Micro

The output sent to micro has the following format:

```
{
    "commands": [
        {
            "id": "integer",
            "move": {
                "direction": "float",
                "speed": "float"
            },
            "moveTo": {
                "x": "float",
                "y": "float",
                "speed": "float"
            },
            "shoot": {
                "direction": "float"
            },
            "shootAt": {
                "x": "float",
                "y": "float"
            }
        },
    ]
}
```

Output example:

```
{
    "commands": [
        {
            "id": 0,
        },
        {
            "id": 1,
            "move": {
                "direction": 0,
                "speed": 10
            }
        },
        {
            "id": 2,
            "shoot": {
                "direction": 90
            }
        },
        {
            "id": 3,
            "move": {
                "direction": 180,
                "speed": 10
            },
            "shoot": {
                "direction": 270
            }
        },
    ]
}
```