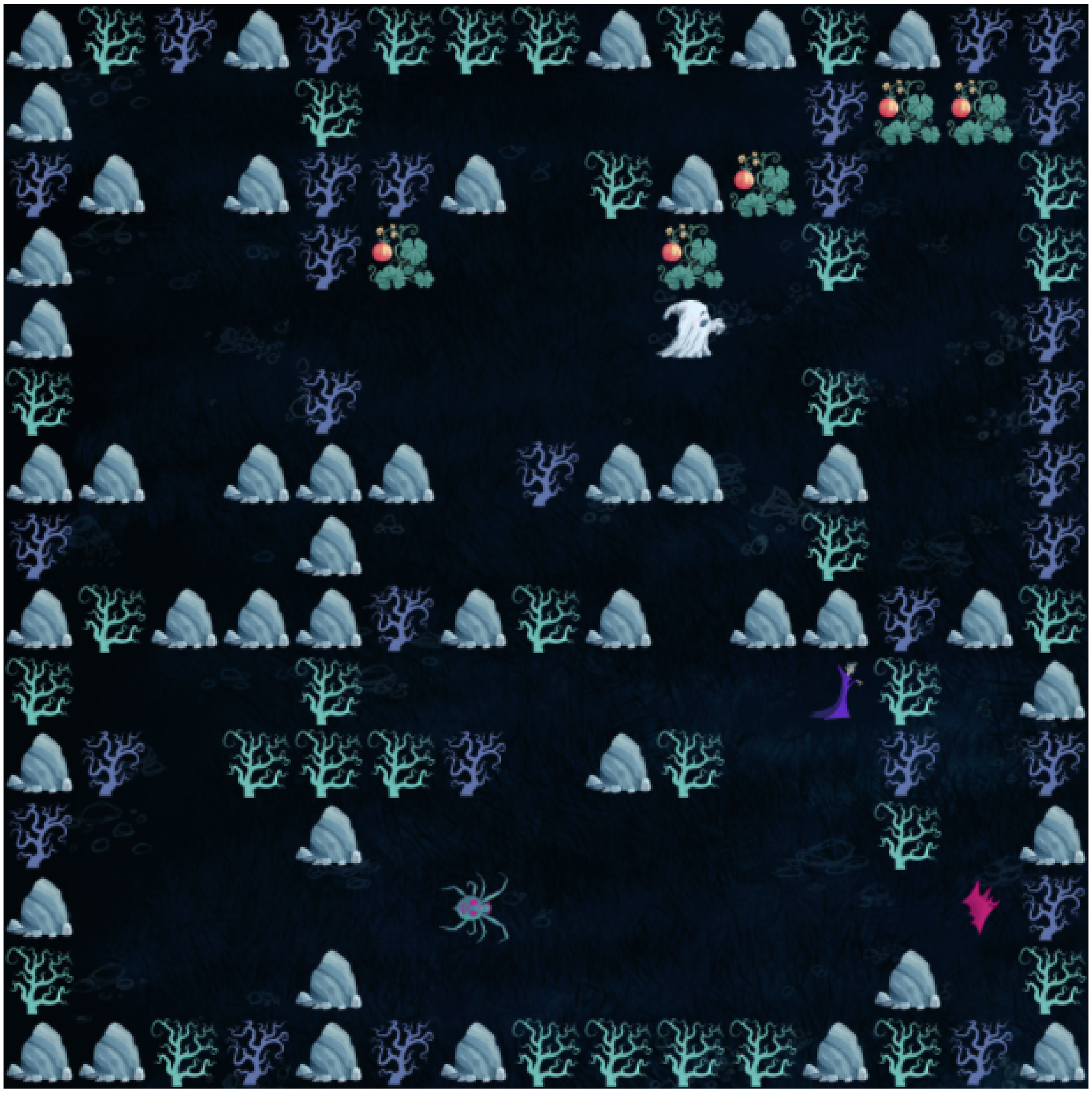
Rules Code Challenge 2020

The idea is quite simple: You have to **create a REST API** to **move a player** across a random generated board. In this board there will also be a number of **enemies** and the other players. You will be able to **move** or **fire** at any direction (up, down, left or right), but you will have limited shots.

You will know the size of the board, the position of your character and some other data, but you won't see the entire board. The game engine will call your API, sending the current conditions, and your API will have to respond with your next move: `up`, `down`, `left`, `right`, `fire-up`, `fire-down`, `fire-left` or `fire-right`.

The game

The game board is a matrix of `N x M` cells containing a randomly generated maze. The width and the height are configurable and can vary from one game to another. Width: `10 ≤ N ≤ 50`, height: `10 ≤ M ≤ 25`. Each cell of the game can contain only one item at the same time: a wall, a player or an enemy.



The **player's character** is moved by the player's REST API. The game engine can manage up to 9 different players concurrently on the same board.

The **enemies** will kill you if they catch you. They move randomly across the board. The number of enemies in the game and the spawn frequency can be configured and it changes from game to game.

When a new enemy is born, he is **neutral** for some movements. A neutral enemy cannot kill a player, but players can kill neutral enemies by touching them. After a few movements a neutral enemy becomes a regular enemy.

When a player is killed, it remains **dead** for some movements. A killed player will respawn in the same place where he was killed.

Players can **shoot** to the enemies or to the other players. f a shot hunts a player or an enemy, it kills him. Shots have a limited range.

The shots are instantaneous, but limited to the **visible area**. If two players shoot each other, both are killed. After shooting, a player must reload and he cannot shoot again until a few movements.

If two or more players try to occupy the **same cell** at the same time, they are **all killed**. If a player occupies the same cell as an enemy, one of them dies. If the enemy is neutral, he dies. Otherwise, the player is the one who dies.

The game engine processes all the **movements** from the players and then the **shots**. So a player can dodge a shot. When all the movements and the shots are processed the game engine moves the **enemies**. So the enemies cannot dodge a shot.

The length of the game is defined by the number of movements which is a configurable parameter of the game (`50` to `5000`). In every movement, a call is made to each of the players' APIs and all responses are processed. The players' APIs have a timeout of `3` seconds. If an API does not respond before the timeout, its movement is considered null. Since all calls are made in parallel, each movement takes a maximum of 3 seconds. So a game of 50 movements will take 2.5 seconds maximum.

< Tech/>

Rules

Basics

- The **Code Challenge 2020** is a programming competition.
- It's open to **all Veepee employees**, particularly to **vpTech** ones.
- The required **programming level** to participate is **high**.
- To join the challenge you'll need to **upload your API** to an Internet server.
- Your API has to be **accessible** from the game server through an URL or IP address.

Restrictions

- No human interaction is allowed within the players' APIs. That is, the APIs must decide their next move without human intervention.

Competition format

- The Competition format will depend on **the number of participants**.
- The idea is to do some **qualifying rounds** to discard non-optimized APIs and then a **big final**.

> ### Example
> 50 participants:
>
> - **1st round**: 50 players, 6 groups (4 of 8 players + 2 of 9 players) → 18 players qualified for the next round.
> - **2nd round**: 18 players, 2 groups of 9 players → 6 players qualified for the final.
> - **Final**: 6 players, 1 group of 9 players, → 3 top players (1 winner).

- The number of **matches per group** and the number of **movements per match** will change from round to round (as well as the width and height of the board and the number of enemies).
- The duration of the challenge will depend on the **number of participants**, the **matches per group** per each round and the **movements per match** per each round.

> ### Example
> 50 participants, 2 rounds and a big final:
>
> - **1st round**: 6 groups, 2 matches per group, 50 movements per match (~2.5 minutes) → 12 matches, ~30 minutes.
> - **2nd round**: 2 groups, 3 matches per group, 100 movements per match (~5 minutes) → 6 matches, ~30 minutes.
> - **Final**: 1 group, 4 matches, 150 movements (~7.5 minutes) → 4 matches, ~30 minutes.
>
> Total 1.5 hours.

API Documentation

Players are simple REST APIs with **two endpoints**:

- `/name` : should provide basic information on a player. See **Name Endpoint** for more information.
- `/move` : will receive the game status and will respond with the next move for the player. See the **Move Endpoint** for more information.

Name Endpoint

Your `/name` URL API will receive a **POST** request without body. Reply with JSON format indicating the **name** of the player or team and the contact **email**.

Request format

```
POST /name HTTP/1.1
Host: your-api.com
```

Response format

```
HTTP/1.1 200 OK

{
    "name": string,        // The player or team name
    "email": string        // The contact email
}
```

Move Endpoint

Your `/move` URL API will receive a POST request with the information about the visible part of the map in JSON format, and you need to reply with the next movement also in JSON format. See visible area section for more information.

Request format

```
POST /move HTTP/1.1
Host: your-api.com

{
    "game": {                   // Object - game data
        "id": uuid              // Unified unique ID of the game
    },
    "player": {                 // Object - player data
        "id": uuid,             // Unified unique ID of the player
        "name": string,         // The name of the player
        "position": {           // Object - current position of the player
            "y": int,
            "x": int
        },
        "previous": {           // Object - previous positions of the player
            "y": int,
            "x": int
        },
        "area": {               // Object - visible area for the player
            "y1": int,
            "x1": int,
            "y2": int,
            "x2": int
        },
        "fire ": "bool"         // If the player can fire this round
    },
    "board": {                  // Object - board data
        "size": {               // Object - Total size of the maze
            "height": int,
            "width": int
        },
        "walls": [              // Array - visible walls
            {                   // Object - wall position
                "y": int,
                "x": int
            },
        ]
    },
    "players": [                // Array - other players positions
        {                       // Object - other players position
            "y": int,
            "x": int
        }
    ],
    "enemies": [               // Array - enemies positions
        {                       // Object - enemy position
            "y": int,
            "x": int,
            "neutral": "bool"   // If the enemy can be killed by touching
        }
    ]
}
```

Response format

```
HTTP/1.1 200 OK

{
    "move": string         // up, down, left or right, fire-up, fire-down, fire-right or fire-left
}
```

Visible area

The board is a 0-based matrix, where [0, 0] is the upper left corner. The height and width are sent in the `board.size.height` and `board.size.width` vars of the request body in the Move endpoint.

Each player has its own **visible area** based on its current position (see Figure 1). The visible area is sent in the `player.area` var which is an object with four vars `y1` , `x1` , `y2` and `x2` .

The information sent in `board.walls` , `enemies` and `players` vars depends on the **visible area**.

The **fire range** also depends on the **visible area**. You can shoot in any straight direction (up, down, left or right) and the fire range is limited to what you can see.

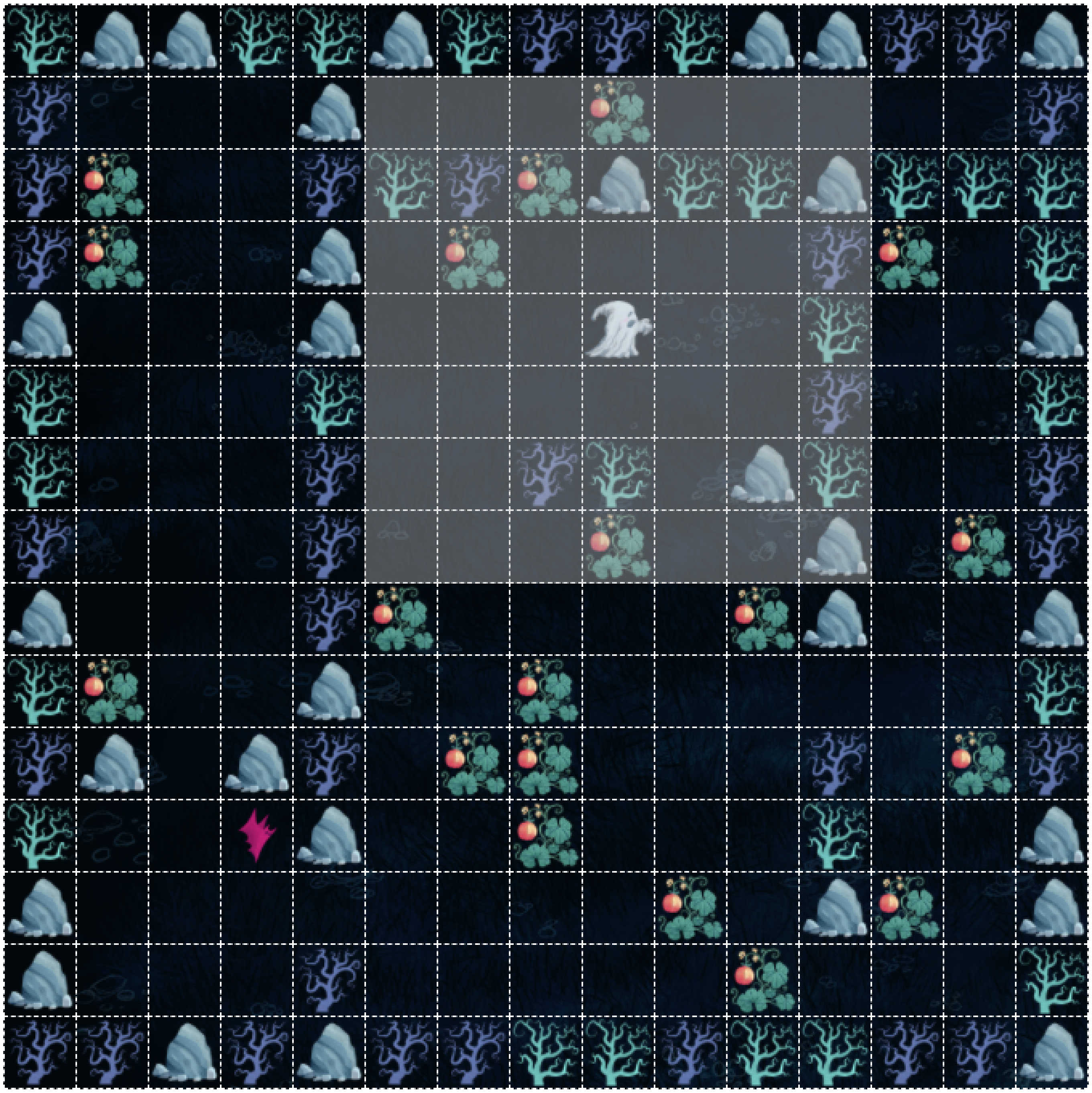


Figure 1: Visible area

Example of request

```
POST /move HTTP/1.1
Host: your-api.com

{
    "game": {
        "id": "78df6fe1-4ba4-408a-ab99-b7122967214f"
    },
    "player": {
        "id": "78df6fe1-4ba4-408a-ab99-b7122967214f",
        "name": "Test player",
        "position": {
            "y": 3,
            "x": 4
        },
        "previous": {
            "y": 2,
            "x": 4
        },
        "area": {
            "y1": 0,
            "x1": 0,
            "y2": 8,
            "x2": 9
        },
        "fire": false
    },
    "board": {
        "size": {
            "height": 15,
            "width": 15
        },
        "walls": [ {
            "y": 0,
            "x": 1
        }, {
            "y": 0,
            "x": 2
        }, {
            "y": 0,
            "x": 3
        }, {
            "y": 0,
            "x": 4
        }, {
            "y": 0,
            "x": 5
        }, {
            "y": 0,
            "x": 6
        }, {
            "y": 0,
            "x": 7
        }, {
            "y": 0,
            "x": 8
        }, {
            "y": 0,
            "x": 9
        }, {
            "y": 1,
            "x": 0
        }, {
            "y": 2,
            "x": 0
        }, {
            "y": 3,
            "x": 0
        }, {
            "y": 4,
            "x": 0
        }, {
            "y": 5,
            "x": 0
        }, {
            "y": 5,
            "x": 1
        }, {
            "y": 5,
            "x": 2
        }, {
            "y": 5,
            "x": 3
        }, {
            "y": 5,
            "x": 4
        }, {
            "y": 5,
            "x": 5
        }, {
            "y": 5,
            "x": 6
        }, {
            "y": 5,
            "x": 7
        }, {
            "y": 5,
            "x": 9
        }, {
            "y": 6,
            "x": 0
        }, {
            "y": 6,
            "x": 4
        }, {
            "y": 7,
            "x": 0
        }, {
            "y": 8,
            "x": 0
        }, {
            "y": 8,
            "x": 4
        } ]
    },
    "players": [ {
        "y": 1,
        "x": 1
    } ],
    "enemies": [ {
        "y": 2,
        "x": 2,
        "neutral": false
    }, {
        "y": 8,
        "x": 3,
        "neutral": true
    } ]
}
```

Example of response

```
HTTP/1.1 200 OK

{
    "move": "fire-up"
}
```