Project Overview

- **Goal**
  - Take items to get score.
  - Disturb the opponent to prevent getting scores.
  - Accumulate more scores than the opponent to win.
- **How to take item?**
  - Items with random scores are randomly spawned to the intersections of the map.
  - You can take an item by simply reaching an intersection.
  - A server lets you know where items are located.
- **How to sabotage the opponent?**
  - You can set a trap when you reach an intersection.
  - A trap reduces the score when a player (Include you!!) reaches the intersection.
  - A server lets you know which intersection has a trap.
  - A server lets you know where your opponent is located.
- **How can I control my robot?**
  - You will set the algorithm to achieve the goals.
  - Your Raspbot automatically moves along with lines based on your algorithm.
  - You cannot manually manipulate your Raspbot.

Game Environment

- **Play time**
  - About 2 minutes. (Might be changed)
- **Map**
  - 4X4 grid with 4 rows, 4 columns and 25 intersections.
  - $(x, y)\ where,\ 0 \leq x, y \leq 4$
  - See figure 1.
- **QR code**
  - QRcode blocks will be placed on every single intersection.
  - A QRcode gives you the location of the corresponding intersection.
  - You should detect and decode QRcode to reach the intersection.
  - A QRcode gives you "xy".
    - For example, if you reach the intersection (1,4), you will get 14.
    - $(0,0) \Rightarrow 00\ (3,2) \Rightarrow 32$
  - You have to convert "xy" to two integer values.
    - For example, when you get 14, convert it to int row =1, int col =4
    - You're going to use these integers to communicate with a server.
  - In short, read the QR code and send the locational information to server. Then, you "reach" the intersection.
- **Progress**
  - An intersection can have three statuses.
    - No item: Nothing will happen.
    - Item: Get score

        - Trap: Loss score
    - When you reach an intersection, the following processes will happen
        - The server might change your score depending on the status.
        - The status of the intersection will be turned into “No item”.
        - You can set bomb or just move on to another intersection.

- **Initial state**
    - Player 1 starts at (0,0) and Player 2 starts at (4,4).
    - Each player has 4 traps.
    - 10 items are placed on a map.

Server

- **How to connect with a server?**
    - You can communicate with a server by using a socket.
    - A port number and address should be variable.
    - You can test your own raspbot code by connecting via local host (127.0.0.1).
- **Server ⇒ You**
    - Server gives you whole map information whenever
        - A new item is spawned.
        - The location of your opponent is changed.
    - A server gives you a struct DGIST.
        - DGIST.players contain information about you and your opponent.
        - DGIST.map contains information about whole intersections.
        - You should unpack the structure to utilize the information.
- **You ⇒ Server**
    - You must let the server know your location and your action whenever you reach an intersection.
    - You give the server a ClientAction structure.
        - ClientAction.row is your x value of (x,y)
        - ClientAction.col is your y value of (x,y)
        - ClientAction.action
            - 0, if you don’t want to set a trap.
            - 1, if you want to set a trap.
- **See “server.h” in our git respiratory. It might be helpful.**

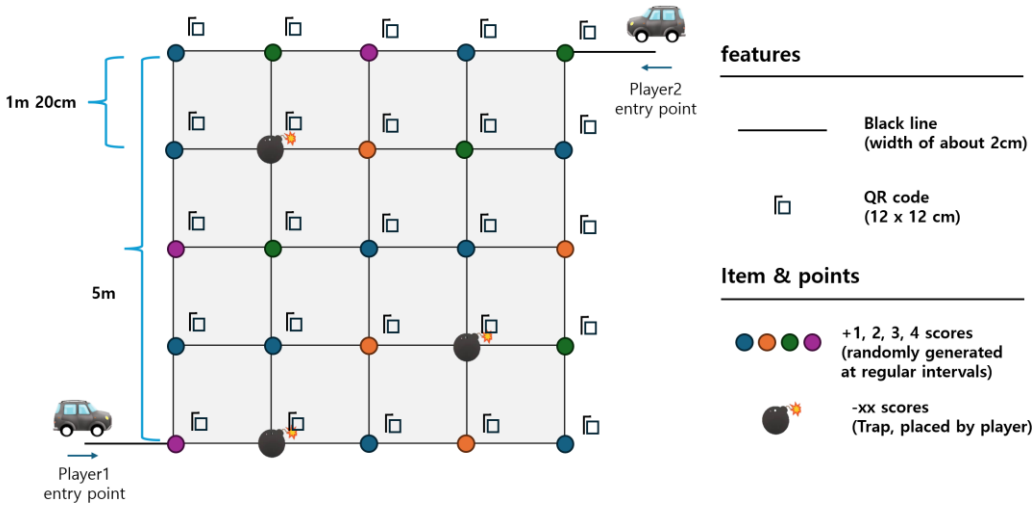

Figure1. Overview

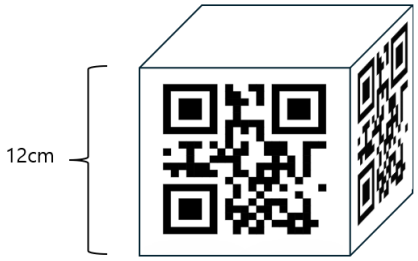

Figure2. QRcode block
(about 7cm away from the ground)

Due & Grading

- **Important Note**:
  - Bring your team raspbot to the class on time at June 4, Tuesday.
  - If you failed to recognize the QRcode, you cannot get items and scores.
  - We uploaded server code on a public git repository. (https://github.com/CELL-DGIST/2024_SystemProgramming_Server.git).
  - The code might be modified. So, you should update the git respiratory to keep the latest status by using “git pull”.
  - Rankings will be made in a double elimination way and the ranking will be reflected in the grade.
- **Submission Requirements:**
  - **GitHub Repository:** Commit your source code until the deadline. The summited code will be used for grading.
  - **Report:** Submit a report of no more than one page. Explain your team’s strategy.
  - **Peer Review**: Each individual should do a mutual evaluation. Write the uploaded form and submit it individually. Please rename the file to “StudentID_Name”.
- **Deadline:**
  - Submit all required files via the Learning Management System (LMS) by June 3, Monday, at 11:59:59 PM.

Getting Help

If you need any help, send us an email.
When you send an email to TAs, you should send mail to both TAs by using CC.

- 이호연: lhyzone@dgist.ac.kr
- 이준영: lolcy3205@dgist.ac.kr

If you need to discuss the project details, you can also email to the instructor.

- 김예성: yeseongkim@dgist.ac.kr