WeThinkCode_ Project Report

RobotWorlds 2

Table of Contents

RobotWorlds Report

This is a report containing a brief overview of the RobotWorlds project.

1. Introduction

In this project, we progressively worked towards building a project that consists of a protocol and two standalone Java programs namely the server program and the client program and that communicates with each other.

- The server receives instructions from a robot, updates the world, and sends a response back to the robot client.
- The client connects over the network to the robot world.
- The protocol allows client robot programs to join any other team's world server programs that are running on the same network. Thus, the client sends request messages to the server and receives response messages from the server. Every request must be replied to by a response.

2.1 Project Objectives - The Server

Overview

Build a server that listens on a network port for robots that will inhabit the world. Once a connection is received, it is responsible for receiving instructions from each robot and updating the world based on the behaviour of each robot.

Server requirements:

- World Grid: A world in the form of a grid.
    - The world has a world coordinate system.
    - The world uses a standard compass with (directions) north, south, east and west.
    - Robots can be launched in the world.
    - The world controls the range of visibility of all robots.
    - The world has obstacles and bottomless pits - it has at least one of each or more the two.

- Obstacles
    - Obstacles are rectangular or square in shape.
    - Obstacles are positioned in the world - in random or specified positions within the bounds of the world grid.
    - Obstacles block the robot's path therefore, a robot cannot move past an obstacle.
    - Obstacles cannot overlap.

- Pits
    - Pits are rectangular or square in shape.
    - Pits are positioned in the world - in random or specified positions within the bounds of the world grid.
    - If a robot enters a bottomless pit, it dies.
    - Pits can overlap.

2.2 Project Objectives - The Client

Overview

The client program must connect to a world using the world's IP address and port. Once connected, it must launch a robot into the world. As a user, you must be able to send the robot commands over the network to the world and the world will update where the robot is, whether it was blocked, etc.

Client requirements:

- Movement and direction
    - The robot can move forward or backward in the axis (vertical/horizontal) it is on and it can also turn left or right.

- Looking around
    - The robot can look around for the maximum distance specified by the world and the world will respond with information within its visible range.

- Robot abilities:
    - Defend - robot has a shield to withstand hits to a certain extent.
    - Repair - robot has ability to repair itself after taking on damage.
    - Active attack - robot has the ability (a gun) to attack enemies (other robots).
        - Reload - robot has to reload after emptying its rounds.
    - Passive attack - robots can place a mine in an unoccupied coordinate in the world.

2.3 Project Objectives - The Protocol

The client sends request messages to the server and receives response messages from the server in the form of a JSON string. This JSON string contains the relevant information (data) about the world and/or robot.

3. Research

Client/Server Communication

The Client/Server communication involves two components, namely a client and a server. The clients send requests to the server and the server responds to the client requests. Basically, the server acts as the producer and the client acts as a consumer.

The three main methods to client/server communication are:

- **Sockets**

  Sockets facilitate communication between two processes on the same machine or different machines. They are used in a client/server framework and consist of the IP address and port number. Many application protocols use sockets for data connection and data transfer between a client and a server.

- **Protocol (RPC)**

  The Remote Procedure Call (RPC) was designed as a way to abstract the procedure call mechanism for use between systems with network connection. It is a software communication protocol that one program can use to request a service from a program located in another computer on a network without having to understand the network's details.

- **Pipes**

  A pipe acts as a connection which allows two processes to communicate, they are interprocess communication methods that contain two end points. Data is entered from one end of the pipe by a process and consumed from the other end by the other process. They typically provide one of the simpler ways for processes to communicate with one another.

4. Planning and Communication

Stand-ups/Scrums

- Weekdays from 09:15 am to 09:30 am were reserved for stand ups.
    - We would use these sessions for progress updates.
- Weekdays from 13:00 pm to 14:00 pm, were reserved for scrum meetings.
    - Which were used for support and planning.

Retrospectives

- Held at the end of every week to reflect on the group performance and individual participation.

Other

- Communication platforms such as slack and google workspaces aided in brief and short communication, in particular for reminders, check ups and to do's.

5.1 Development Tools

Development Tools

- Git
  - Git: Source files version control.
  - Gitlab: Store source files for remote access.
- Maven
  - Dependency and plugin management.
  - Compiling, testing and packaging.
- Docker
  - Developing, shipping, and running applications.
  - Manage project infrastructure.
- IntelliJ IDEA
  - Integrated Development Environment (IDE) for JVM languages designed to maximize developer productivity.
  - Best IDE for java development.
- CodeScene
  - A behavioural code analysis tool that provides visualisations based on version-control data and machine learning algorithms that identify social patterns and hidden risks in code.
  - Detects hotspots in complex code that an organization has to work with frequently and prioritises technical debt based on how the developers work with the code.

5.2 Development Technologies

Development Technologies

- Java
  - Programming language used.
  - Object Oriented Programming was preferred over Functional Programming.
- JDBC (Java Database Connectivity)
  - Database implementation.
  - Java-based data access technology used for Java database connectivity.
- Javelin
  - A lightweight web framework for Kotlin and Java designed to be simple and blocking by default, supporting WebSockets, HTTP2, and async requests.
- Thymeleaf
  - Java template engine for processing and creating HTML, CSS , and JS.
- HTML (HyperText Markup Language)
  - Code used to structure a web page and its contents.
- CSS (Cascading Style Sheets)
  - Used to style and layout web pages.
- JS (JavaScript)
  - Text-based programming language used to make web pages interactive, it is used on both the client-side and server-side.

6. Conclusion

The team put its best efforts into learning the new project relevent concepts , working together to come up with a minimum viable product.

When it came to the planning and communication, some things could have been planned better, given that we did not achieve all the milestones the team had set for itself.

Conclusively, I believe we had a steep learning curve when it came to this project, however, we got to learn much and get a viable product in the end.