TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

LÊ MINH QUÂN – 20120356

BÁO CÁO ĐỒ ÁN ZACE GAME

MÔN HỌC CHUYÊN ĐỀ HỆ THỐNG PHÂN TÁN

CHƯƠNG TRÌNH CHÍNH QUY

GIÁO VIÊN LÝ THUYẾT

TS. TRẦN TRUNG DŨNG

Tp. Hồ Chí Minh, tháng 01/2024

Thông tin cơ bản

Đồ án sử dụng ngôn ngữ lập trình là Java với build tool là Gradle.

Main Flow

Giao tiếp giữa các tiến trình

Phần này em chưa hoàn thiện, nhưng em dự định sẽ làm tương tự với đồ án SES trước đó trong việc giao tiếp giữa các tiến trình.

Mỗi chương trình khi khởi chạy cần truyền vào một đối số dòng lệnh cho biết port của tiến trình. Cụ thể, câu lệnh sẽ có dạng như sau:

```
./gradlew run -Pport=8080
```

Các giá trị port có thể sử dụng được quy định trước. Em dự định sẽ viết một bash script để kiểm tra xem port có khả dụng hay không, nếu có thì gọi đến câu lệnh trên và truyền vào port khả dụng.

Sau khi khởi chạy thành công thì chương trình sẽ tạo ra một instance của lớp **Process** chịu trách nhiệm lắng nghe các kết nối và gửi message cho các tiến trình khác.

Cụ thể, constructor của **Process** sẽ tạo một instance của lớp **ServerSocket**. Nếu tạo thành công thì sẽ tiếp tục tạo lớp **Receiver** dùng để tạo một client socket tương ứng với mỗi yêu cầu kết nối gửi đến. Quá trình lắng nghe và tạo socket cho các kết nối gửi đến này của process sẽ diễn tra ở trong một thread riêng biệt để tránh blocking.

Sau đó, process sẽ cố gắng mở kết nối socket đến các port đã được quy định trước ở trong class **Configuration** và đồng thời tạo một instance của class **Sender** tương ứng với mỗi port được mở. Nếu có một port nào không thể mở kết nối thì process sẽ thử lại sau 10 giây. Quá trình này cũng diễn ra ở trong một luồng riêng biệt.

Cuối cùng, đối với việc gửi tin nhắn, mỗi khi cần gửi tin nhắn, chương trình sẽ dùng instance của lớp **Sender** để gửi tin. Lớp này có chứa hai instance của hai lớp: **BufferedReader** và **BufferedWriter** cho phép đọc và ghi dữ liệu được gửi thông qua socket. Việc gửi tin cần phải được thực hiện ở trong một luồng riêng để tránh blocking.

Cấu trúc message bao gồm:

- Port bên gửi
- Port bên nhận
- Loại message
- Thông điệp của message
- Các thông tin về thời gian (timestamp, vector clock)

Message sẽ được chuyển thành dạng JSON trước khi gửi đi.

Đồng bộ hóa

Khi một người chơi khởi chạy chương trình, mê cung sẽ được random ngẫu nhiên bởi lớp **Maze**. Thời gian sinh mê cung sẽ được lưu lại. Nếu các người chơi khác khởi chạy sau thì sẽ phải gửi message đến các process để nhận phản hồi về process có thời gian sinh mê cung sớm nhất và chọn process đó để đồng bộ mê cung.

Quá trình đồng bộ này có thể được loại bỏ nếu sử dụng một cấu hình mê cung cố định.

Đối với các thông số khác, chẳng hạn như thông tin về người chơi (lưu ở trong lớp **Player**), process khi khởi chạy sẽ tự động random các thông tin này và gửi cho các process khác để đồng bộ.

Design Flow

Khi người chơi nhấn Enter, trò chơi sẽ bắt đầu, mê cung sẽ được khởi tạo. Vị trí và hướng đi sẽ là random. Tốc độ của nhân vật là cố định.

Khi người chơi nhấn A, D, S, W thì nhân vật sẽ di chuyển sang trái, phải, xuống và lên.

Khi người chơi nhấn Space thì sẽ bắn đạn, có thể có nhiều viên đạn của người chơi cùng tồn tại một lúc và sẽ được đánh số từ 0.

Khi nhấn P thì trò chơi sẽ tạm dừng và nhấn Q thì trò chơi sẽ kết thúc. Khi nhấn hai phím này thì chương trình sẽ tiến hành ghi log ra file.

Tất cả các chức năng liên quan đến chế độ chơi 1 người từ 1 đến 7 đều đã hoàn thành.

Đảm bảo thứ tự

Em dự định sẽ đảm bảo thứ tự bằng cách sử dụng thuật toán SES để so sánh vector clock và timestamp có trên từng messasge.

Tham khảo

- https://www.youtube.com/watch?v=ATz7bIqOjiA
- https://youtu.be/Wcvqnx14cZA?si=7QupELX660v4dJ7
- https://pygamezero-pacman.readthedocs.io/en/latest/index.html