
ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

---

# BÁO CÁO ĐỒ ÁN 3

**Đề tài: MAZE BATTLE**

---

**Môn học: Chuyên đề Hệ thống phân tán**

*Sinh viên thực hiện:*

Võ Hữu Tuấn - 22127439

*Giáo viên hướng dẫn:*

Thầy Trần Trung Dũng

# Mục lục

# 1 Giới thiệu

## 1.1 Mô tả bài toán

Đồ án này xây dựng một ứng dụng trò chơi phân tán tên "Maze War" (Cuộc chiến mê cung). Trò chơi cho phép nhiều người chơi (Multiplayer) tham gia cùng lúc, mỗi người điều khiển một chiến binh (xe tank) trong một mê cung có kích thước cố định $32 \times 16$ ô

Mục tiêu của người chơi là ghi điểm bằng cách bắn hạ đối thủ, đồng thời tránh né làn đạn. Hệ thống cần đảm bảo tính nhất quán về trạng thái game (vị trí, điểm số, đạn bay) giữa tất cả các người chơi (Clients).

## 1.2 Yêu cầu chức năng

1. **Góc nhìn:** Từ trên cao xuống (Top-down view), toàn bộ bản đồ hiển thị giống nhau cho mọi người chơi.
2. **Cơ chế bắn:** Tốc độ đạn nhanh gấp 4 lần tốc độ di chuyển của xe ($v_{bullet} = 4 \times v_{tank}$).
3. **Hồi sinh (Respawn):** Khi bị trúng đạn, người chơi bị trừ 5 điểm, bị loại khỏi màn chơi tạm thời và hồi sinh tại vị trí ngẫu nhiên an toàn sau một khoảng thời gian chờ.
4. **Tính điểm:** Bắn trúng địch (+11 điểm), Bắn đạn (-1 điểm/viên).
5. **Đồng bộ:** Đảm bảo "nguyên lý một viên đạn chỉ trúng một đích" và hai người chơi không thể trùng vị trí.

# 2 Phân tích & Thiết kế

## 2.1 Kiến trúc Hệ thống

Hệ thống được thiết kế theo mô hình **Client-Server Authoritative** (Máy chủ quyết định).

- **Server (Máy chủ):** Đóng vai trò là "Source of Truth". Server chịu trách nhiệm nhận Input từ Client, tính toán logic vật lý (di chuyển, va chạm), cập nhật trạng thái game và gửi (Broadcast) trạng thái mới xuống cho toàn bộ Client.

- **Client (Người chơi):** Đóng vai trò là "Dumb Terminal". Client chỉ chịu trách nhiệm thu thập sự kiện phím bấm (thao tác của người chơi), gửi yêu cầu về Server và vẽ (Render) hình ảnh dựa trên dữ liệu Server trả về. Client không tự ý thay đổi tọa độ cục bộ để tránh gian lận.

## 2.2 Giao thức Truyền tin (Protocol)

Để giải quyết vấn đề phân mảnh gói tin (Fragmentation) trong giao thức TCP, nhóm sử dụng kỹ thuật **Length-Prefixed Framing** (Đóng gói theo độ dài). Cấu trúc gói tin được định nghĩa trong module `common/protocol.py`:

$$Packet = \underbrace{[Header]}_{4 \text{ bytes}} + \underbrace{[Payload]}_{N \text{ bytes}} \tag{1}$$

- **Header:** Một số nguyên 4-byte (Unsigned Int, Big-Endian) biểu diễn độ dài $N$ của phần dữ liệu.
- **Payload:** Dữ liệu game được tuần tự hóa (Serialize) dưới dạng chuỗi JSON và mã hóa UTF-8.

## 2.3 Luồng xử lý sự kiện (Design Flow)

Quy trình xử lý khi một người chơi thực hiện hành động (Ví dụ: Di chuyển) như sau:

1. **Bước 1 (Client Input):** Người chơi nhấn phím. Client đóng gói yêu cầu JSON: `{"action": "MOVE", "dir": "UP"}` và gửi qua TCP Socket.
2. **Bước 2 (Server Queue):** Server nhận gói tin tại luồng riêng biệt (`ClientHandler`). Yêu cầu được đưa vào hàng đợi xử lý hoặc xử lý tức thời để cập nhật vector hướng của nhân vật.
3. **Bước 3 (Game Loop):** Tại chu kỳ cập nhật tiếp theo (Server Tick  60Hz):
   - Server tính toán tọa độ mới: $P_{new} = P_{old} + v \times \Delta t$.
   - Kiểm tra va chạm với tường (Map) và các xe khác.
   - Nếu hợp lệ, cập nhật vị trí vào bộ nhớ.
4. **Bước 4 (Broadcast):** Server gửi gói tin `GAME_UPDATE` chứa toàn bộ danh sách tọa độ người chơi và đạn cho tất cả Client đang kết nối.

5. **Bước 5 (Render):** Client nhận gói tin, xóa màn hình và vẽ lại toàn bộ đối tượng tại tọa độ mới.

### 2.4 Giải thuật Đồng bộ Xử lý va chạm

**Xử lý va chạm (Collision Detection):** Sử dụng phương pháp AABB (Axis-Aligned Bounding Box) đơn giản hóa trên lưới (Grid-based). Bản đồ được chia thành ma trận $32 \times 16$. Một vị trí $(x, y)$ được coi là va chạm nếu ô lưới tương ứng $[row][col]$ được đánh dấu là tường.

**Thuật toán Hồi sinh an toàn (Safe Respawn):** Khi người chơi bị hạ gục, Server sử dụng thuật toán sau để tìm vị trí hồi sinh:

```
def find_safe_spawn():
    While True:
        pos = random_cell()
        if is_wall(pos): continue
        if is_player_at(pos): continue
        if not is_line_of_fire_blocked(pos): continue
        return pos
```

Listing 1: Thuật toán tìm vị trí hồi sinh

## 3 Tổ chức hệ thống

### 3.1 Cấu trúc Thư mục

Dự án được tổ chức theo cấu trúc module hóa, tách biệt Client và Server:

- **client/**: Mã nguồn phía người chơi.
  - `client.py`: Điểm khởi chạy chính, chứa vòng lặp Pygame.
  - `game_client.py`: Class xử lý logic mạng, nhận gửi gói tin.
  - `ui_manager.py`: Quản lý việc vẽ bản đồ và sprite.
- **server/**: Mã nguồn phía máy chủ.
  - `server.py`: Chứa vòng lặp game chính (Game Loop) và quản lý kết nối.

  - `map_manager.py`: Load bản đồ và kiểm tra va chạm.
  - `user_manager.py`: Quản lý đăng nhập và lưu điểm số vào `users.json`.

- **common/**: Các module dùng chung (Constants, Protocol, Logger).

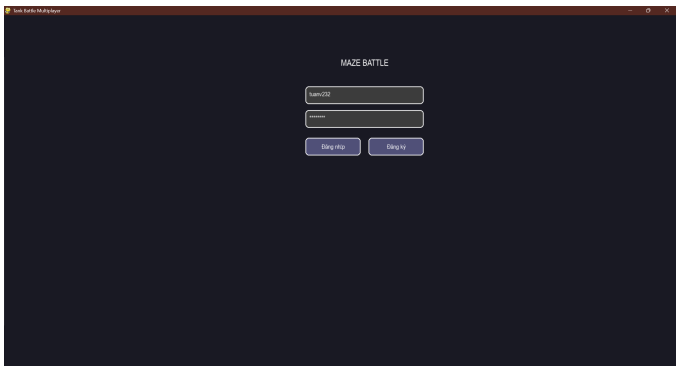
### 3.2 Công nghệ sử dụng

- **Ngôn ngữ:** Python 3.8+.
- **Thư viện đồ họa:** Pygame
- **Giao thức mạng:** Python `socket` (TCP/IP), `threading` (Đa luồng).
- **Lưu trữ:** JSON (Lưu thông tin tài khoản như username và mật khẩu cùng điểm số tích lũy).

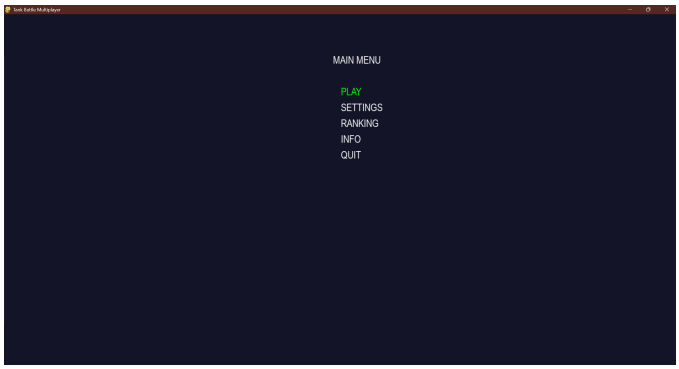
## 4 Kết quả đạt được

### 4.1 Giao diện Đăng nhập & Menu

Hệ thống hỗ trợ đăng ký và đăng nhập. Dữ liệu người dùng được lưu trữ bền vững (Persistent) trong file `data/users.json`.



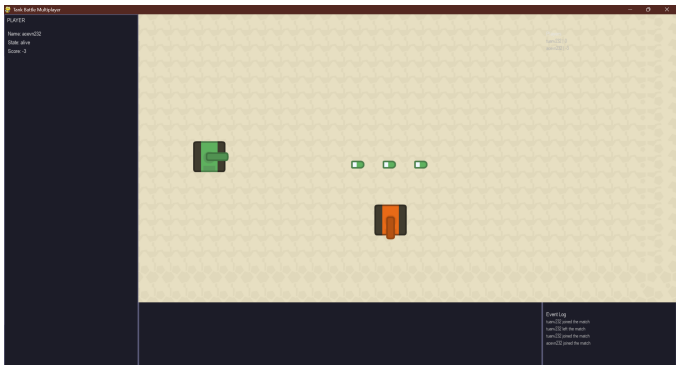

Hình 1: Giao diện đăng nhập




Hình 2: Giao diện Menu chính

### 4.2 Màn hình Gameplay

Trò chơi hoạt động mượt mà với nhiều Client kết nối cùng lúc. Các tính năng di chuyển, bắn đạn, tính điểm và hồi sinh hoạt động đúng theo yêu cầu thiết kế.



Hình 3: Giao diện game play

## 5 Tổng kết và Hướng phát triển

### 5.1 Kết luận

Đồ án đã xây dựng thành công một hệ thống Game Multiplayer phân tán cơ bản. Nhóm đã áp dụng thành công kiến thức về Lập trình mạng (Socket), xử lý đồng bộ (Synchronization) và kiến trúc Client-Server. Hệ thống đảm bảo tính đúng đắn của dữ liệu và trải nghiệm thời gian thực.

### 5.2 Hướng phát triển

- **Bảo mật:** Mã hóa mật khẩu người dùng (Hashing) thay vì lưu Plain-text.

- **Tối ưu mạng:** Cải thiện UI/UX đẹp hơn để nâng cấp trải nghiệm người dùng
- **Gameplay:** Bổ sung các vật phẩm hỗ trợ (Buff) và bản đồ đa dạng hơn.

# PHỤ LỤC

## A. Liên kết mã nguồn

- Link Project 03 | GitHub Repository
- Link Logs (demo) | Google Drive

## B. Video demo

- Link Video demo | Google Drive

# TÀI LIỆU THAM KHẢO

1. Tài liệu bài giảng Hệ thống phân tán, **Day 2 CDIO** - Khoa Công nghệ Thông tin, Trường Đại học Khoa học Tự nhiên - ĐHQGHCM
2. TOP DOWN TANKS - KENNEY