

**ỦY BAN NHÂN DÂN TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN**

—o0o—



**BÁO CÁO
PHÁT TRIỂN HỆ ĐIỀU HÀNH MÃ NGUỒN MỞ
ĐỀ TÀI:
XÂY DỰNG GAME BOMBERMAN**

Giảng viên hướng dẫn: **ThS.Từ Lãng Phiêu**

Nhóm: **17**

Sinh viên: **3120410321 - Nguyễn Hoàng Minh**
3120410137 - Nguyễn Đào Linh Đan
3120410326 - Võ Thị Diễm My

Lớp: **DCT1214**

TP.Hồ Chí Minh, 5/2024

MỤC LỤC

LỜI NÓI ĐẦU

LỜI CẢM ƠN

CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI	1
1.1 Lịch sử phát triển	1
1.1.1 Bomberman - Huyền thoại game đặt bom với hành trình bùng nổ .	1
1.1.2 1983 - Khởi đầu cho huyền thoại:	1
1.1.3 1985 - 1990 - Bùng nổ và lan tỏa:	1
1.1.4 1997 - Bomberman 64 - Bước tiến đột phá:	1
1.1.5 1999 - Bomberman Online - Kỷ nguyên kết nối:	1
1.2 Mục tiêu	2
1.3 Phạm vi đề tài	2
1.4 Kết hoạch triển khai	2
 CHƯƠNG 2. GIỚI THIỆU VỀ PYGAME, SOCKET VÀ CÁC ỨNG DỤNG, CÔNG NGHỆ	 4
2.1 Giới thiệu về Pygame	4
2.1.1 Khái niệm về Pygame	4
2.1.2 Lịch sử hình thành và phát triển pygame	5
2.1.3 Ưu – nhược điểm của pygame	6
2.1.4 Ưu điểm	6
2.1.5 Nhược điểm	6
2.2 Giới thiệu về Socket	7
2.2.1 Khái niệm về Socket	7
2.2.2 Vai trò của Socket	7

2.2.3	Cách thức hoạt động của Socket	8
2.2.4	Lịch sử phát triển của Socket	8
2.2.5	Ưu - nhược điểm của Socket	9
2.2.6	Ưu điểm	9
2.2.7	Nhược điểm	9
2.3	Các ứng dụng, công nghệ và công cụ thực hiện	10
2.3.1	Các ứng dụng, công nghệ	10
2.3.2	Công cụ thực hiện	11
CHƯƠNG 3. PHÂN TÍCH THIẾT KẾ		12
3.1	Sơ đồ class Bomber	12
CHƯƠNG 4. CÀI ĐẶT MÔI TRƯỜNG		13
4.1	Cài đặt Pycharm JetBrain sử dụng Toolbox App	13
4.2	Install package trong Pycharm	18
CHƯƠNG 5. XÂY DỰNG VÀ PHÁT TRIỂN TRÒ CHƠI BOOMBERMAN		21
5.1	Xây dựng trò chơi Bomberman	21
5.1.1	Khởi tạo đối tượng Player	21
5.1.2	Các phương thức vẽ và cập nhật của Player	22
5.1.3	Phương thức di chuyển của Player	23
5.1.4	Kiểm tra hướng di chuyển hợp lệ	24
5.1.5	Kiểm tra va chạm và lấy thông tin vị trí và hình ảnh của player . . .	25
5.1.6	Đặt bom và lấy danh sách bom	26
5.1.7	Xử lý thời gian đếm ngược và nổ bom của người chơi	26
5.1.8	Xử lý kết thúc và chờ đợi trong trò chơi	27
5.1.9	Khôi phục và kiểm tra kết thúc game cho người chơi	28
5.1.10	Truy xuất thuộc tính người chơi	29
5.1.11	Phương thức truy xuất thông tin người chơi	29

5.1.12	Cập nhật thông tin người chơi từ DTO	30
5.1.13	Khởi tạo hiệu ứng nổ bom	32
5.1.14	Vẽ hiệu ứng nổ bom	32
5.1.15	Thiết lập tọa độ cho hiệu ứng nổ bom	33
5.1.16	Quản lý hiệu ứng nổ bom	33
5.1.17	Kiểm tra va chạm với người chơi	34
5.1.18	Khởi tạo trường đánh và các đối tượng trong trò chơi	35
5.1.19	Vẽ trường đánh	36
5.1.20	Xóa ô hộp khỏi ma trận và màn hình	37
5.1.21	Vẽ các phần tử trong trò chơi	38
5.1.22	Chỉ số vị trí của người chơi trong ma trận	39
5.1.23	Tính vị trí đối tượng trong ma trận	40
5.1.24	Debugging và In thông tin ma trận	40
5.1.25	Animate Bom	41
5.1.26	Vẽ các đối tượng lên màn hình	42
5.1.27	Các phương thức đặt bom	43
5.1.28	Kích hoạt vụ nổ của bomb	44
5.1.29	Ẩn bomb sau khi nổ	45
5.1.30	Tạo hiệu ứng nổ của bom	46
5.1.31	Khởi tạo kết nối và luồng xử lý cho client	47
5.1.32	Giao tiếp Client-Server cho Game nhiều người chơi	48
5.1.33	Khởi tạo Client Socket	49
5.1.34	Kết nối đến server và nhận dữ liệu từ server	50
5.1.35	Phương thức gửi dữ liệu và nhận phản hồi	51

CHƯƠNG 6. GIAO DIỆN GAME 53

6.1	Giao diện Game Bomberman	53
6.1.1	Giao diện khi bắt đầu vào Game	53

6.1.2	Giao diện đặt Boom	54
6.1.3	Giao diện game 2 người chơi	54
6.1.4	Giao diện khi Boom nổ và người chơi chết	55

CHƯƠNG 7. TÀI LIỆU THAM KHẢO	56
-------------------------------------	-----------

LỜI NÓI ĐẦU

Trong thời đại công nghệ phát triển như hiện nay, các trò chơi trực tuyến đang ngày càng trở thành một phần không thể thiếu trong cuộc sống hàng ngày của chúng ta. Trong số đó, trò chơi Poker là một trong những trò chơi phổ biến nhất và được yêu thích trên toàn thế giới.

Việc xây dựng một game Bomberman không chỉ đem lại sự giải trí mà còn là một thách thức đối với các nhà phát triển. Trò chơi này đòi hỏi một hệ thống phức tạp để quản lý luật chơi, tương tác giữa người chơi và máy chủ, cũng như các yếu tố đồ họa và âm thanh để tạo ra trải nghiệm chân thực nhất cho người chơi.

Trong khóa luận này, chúng tôi tập trung vào việc xây dựng một game Bomberman nhiều người chơi sử dụng ngôn ngữ lập trình Python và các thư viện. Chúng tôi sẽ trình bày cách thiết kế và triển khai hệ thống, cùng với các chi tiết về cách thức hoạt động của game và các tính năng mà chúng tôi đã thực hiện.

Hy vọng rằng khóa luận này sẽ cung cấp cho bạn cái nhìn tổng quan về quy trình phát triển một game Bomberman, cũng như cung cấp ý tưởng và kiến thức để bạn có thể tiếp tục nghiên cứu và phát triển sản phẩm của riêng mình trong tương lai.

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn chân thành đến với giảng viên môn Phát triển phần mềm mã nguồn mở – ThS.Từ Lăng Phiêu, người đã nhiệt tình giảng dạy và hướng dẫn chúng em trong suốt quá trình học tập cũng như hoàn thiện bài báo cáo đề tài môn học này. Trong quá trình nghiên cứu và làm báo cáo đề tài, vì trình độ lý luận cũng như kinh nghiệm chuyên môn còn nhiều hạn chế nên báo cáo tiểu luận sẽ có sai sót, chúng em mong là sẽ nhận được ý kiến đóng góp, đánh giá từ thầy để có thể rút kinh nghiệm, học hỏi thêm để những đề tài tiếp theo được hoàn thiện hơn.

Chúng em xin chân thành cảm ơn thầy!

CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI

1.1 Lịch sử phát triển

1.1.1 *Bomberman - Huyền thoại game đặt bom với hành trình bùng nổ*

Bomberman, tựa game đặt bom huyền thoại, đã đồng hành cùng tuổi thơ của biết bao thế hệ game thủ. Khởi nguồn từ xứ sở hoa anh đào, Bomberman đã trải qua hành trình phát triển đầy ấn tượng, ghi dấu ấn sâu đậm trong lòng người hâm mộ.

1.1.2 *1983 - Khởi đầu cho huyền thoại:*

Năm 1983, Bomberman lần đầu tiên được Hudson Soft ra mắt trên hệ máy MSX. Trò chơi nhanh chóng thu hút sự chú ý bởi lối chơi đơn giản nhưng đầy hấp dẫn: điều khiển nhân vật đặt bom, phá hủy chương ngại vật và tiêu diệt kẻ thù.

1.1.3 *1985 - 1990 - Bùng nổ và lan tỏa:*

Bomberman tiếp tục phát triển mạnh mẽ, trở thành "bom tấn" trên các hệ máy NES, SNES, Sega Genesis,... Phiên bản này ghi dấu ấn với nhiều chế độ chơi đa dạng, kho vũ khí phong phú và hệ thống màn chơi đầy thử thách.

1.1.4 *1997 - Bomberman 64 - Bước tiến đột phá:*

Năm 1997, Bomberman 64 đánh dấu bước ngoặt lịch sử khi chuyển sang góc nhìn 3D đầy ấn tượng. Bomberman 64 mở ra kỷ nguyên mới cho series, với đồ họa đẹp mắt, lối chơi hấp dẫn và thu hút đông đảo người chơi.

1.1.5 *1999 - Bomberman Online - Kỷ nguyên kết nối:*

Bomberman Online ra mắt trên hệ máy Sega Dreamcast, mang đến trải nghiệm chơi game trực tuyến đầy mới mẻ. Người chơi có thể so tài cùng bạn bè, thi đấu trên nhiều bản đồ khác nhau, tạo nên bầu không khí sôi động và náo nhiệt.

- 2000 - nay - Bomberman đa nền tảng:
- Trò chơi tiếp tục phát triển và mở rộng trên nhiều nền tảng, bao gồm PlayStation, Xbox, PC, và các thiết bị di động.
- Bomberman Live, Bomberman Ultra, và nhiều phiên bản khác ra mắt, mang lại trải nghiệm đa dạng và phong phú.
- Bomberman R, phiên bản mới nhất, được phát hành trên Nintendo Switch, PS4, Xbox

One, và PC, tiếp tục thu hút lượng lớn người hâm mộ.

1.2 Mục tiêu

Phát triển kỹ năng lập trình Python thông qua việc xây dựng game Bomberman đơn giản nhưng hấp dẫn.

Tiếp cận và hiểu cơ bản về lập trình game, bao gồm xử lý sự kiện, đồ họa, và tương tác người chơi. Khuyến khích sự sáng tạo thông qua việc tự do thiết kế và mở rộng trò chơi dựa trên ý tưởng cá nhân.

Áp dụng kiến thức đã học trong việc giải quyết vấn đề thực tế, từ việc xử lý vật lý đến tạo ra trải nghiệm người chơi tốt.

1.3 Phạm vi đề tài

- Nghiên cứu về cơ chế gameplay trong trò chơi đặt bom:
 - Cách di chuyển
 - Cách đặt bom
 - Chiến lược
- Triển khai, thiết kế và phát triển giao diện người dùng cho trò chơi:
 - Trộn địa game
 - Các nút điều khiển
 - Hiển thị thông tin về người chơi
 - Hiển thị các đối tượng, vật thể
- Phát triển trò chơi đặt bom trên nhiều nền tảng:
 - Ứng dụng di động (Android, iOS)
 - Ứng dụng web
 - Các nền tảng khác nếu có yêu cầu

1.4 Kết hoạch triển khai

- Sử dụng nền tảng công nghệ: Python3.
- Môi trường phát triển: PyCharm.
- Công cụ quản lý mã nguồn như Git.

- Sử dụng một số Framework như: Pygame (logic game), Socket (kết nối các người chơi với nhau),...

- Phát triển các tính năng chính của game Poker:

+ Tạo phòng (tạo nhiều phòng)

+ Vào phòng

+ Chế độ chơi nhiều người

+ Chức năng phân lượt

- Phát triển các tính năng chính của game đặt Bomb:

+ Bắt đầu game

+ Đợi người chơi

+ 2 người chơi

+ Chức năng đặt bomb

+ Phá vật thể

+ Phân thắng thua

- Tối ưu hóa giao diện người dùng, đảm bảo trải nghiệm về game cho người chơi.

CHƯƠNG 2. GIỚI THIỆU VỀ PYGAME, SOCKET VÀ CÁC ỨNG DỤNG, CÔNG NGHỆ

2.1 Giới thiệu về Pygame

2.1.1 *Khái niệm về Pygame*

- Pygame là một thư viện Python để phát triển trò chơi và ứng dụng đa phương tiện, dựa trên thư viện SDL (Simple DirectMedia Layer).
- Pygame cung cấp các công cụ và chức năng cần thiết cho việc tạo ứng dụng đồ họa từ đơn giản đến phức tạp.
- Đồ họa và âm thanh:
 - Cung cấp các chức năng mạnh mẽ để vẽ đồ họa và xử lý âm thanh.
 - Tạo cửa sổ đồ họa, vẽ hình ảnh, quản lý sprite và phát lại âm thanh dễ dàng.
- Xử lý sự kiện:
 - Quản lý và xử lý sự kiện người dùng như nhấn phím, di chuyển chuột và các sự kiện khác.
 - Quan trọng cho việc xây dựng trò chơi tương tác.
- Đối tượng Sprite:
 - Hỗ trợ mô hình đối tượng Sprite để tổ chức và quản lý các đối tượng trong trò chơi.
- Dễ học và sử dụng:
 - Được thiết kế để dễ học và sử dụng, đặc biệt cho người mới bắt đầu lập trình trò chơi.
- Di động và nền tảng đa dạng:
 - Khả năng chạy trên nhiều nền tảng, bao gồm Windows, macOS và Linux.
 - Phát triển ứng dụng một cách linh hoạt.
- Các hàm hỗ trợ khác giúp giảm độ phức tạp khi phát triển trò chơi.
- Hiểu cách Pygame hoạt động giúp bạn phát triển các trò chơi với cấu trúc và logic riêng.

2.1.2 Lịch sử hình thành và phát triển pygame

- Pygame là một thư viện Python để phát triển trò chơi và ứng dụng đa phương tiện, dựa trên thư viện SDL (Simple DirectMedia Layer).
- Pygame cung cấp các công cụ và chức năng cần thiết cho việc tạo ứng dụng đồ họa từ đơn giản đến phức tạp.
- Đồ họa và âm thanh:
 - Cung cấp các chức năng mạnh mẽ để vẽ đồ họa và xử lý âm thanh.
 - Tạo cửa sổ đồ họa, vẽ hình ảnh, quản lý sprite và phát lại âm thanh dễ dàng.
- Xử lý sự kiện:
 - Quản lý và xử lý sự kiện người dùng như nhấn phím, di chuyển chuột và các sự kiện khác.
 - Quan trọng cho việc xây dựng trò chơi tương tác.
- Đối tượng Sprite:
 - Hỗ trợ mô hình đối tượng Sprite để tổ chức và quản lý các đối tượng trong trò chơi.
- Dễ học và sử dụng:
 - Được thiết kế để dễ học và sử dụng, đặc biệt cho người mới bắt đầu lập trình trò chơi.
- Di động và nền tảng đa dạng:
 - Khả năng chạy trên nhiều nền tảng, bao gồm Windows, macOS và Linux.
 - Phát triển ứng dụng một cách linh hoạt.
- Các hàm hỗ trợ khác giúp giảm độ phức tạp khi phát triển trò chơi.
- Hiểu cách Pygame hoạt động giúp bạn phát triển các trò chơi với cấu trúc và logic riêng.

2.1.3 Ưu – nhược điểm của pygame

2.1.4 Ưu điểm

- **Dễ học và sử dụng:**
 - Thân thiện với người mới bắt đầu với lập trình game hoặc Python.
 - Nhiều tài nguyên học trực tuyến.
- **Cộng đồng lớn và hỗ trợ:**
 - Cộng đồng người dùng lớn.
 - Nhiều tài nguyên hỗ trợ trực tuyến và ví dụ mã nguồn.
- **Mã nguồn mở và linh hoạt:**
 - Cho phép sửa đổi và tùy chỉnh mã nguồn.
- **Tương thích đa nền tảng:**
 - Hỗ trợ Windows, macOS và Linux.
- **Sử dụng cùng Python:**
 - Tận dụng các ưu điểm của Python như đơn giản, dễ đọc, và nhanh chóng.

2.1.5 Nhược điểm

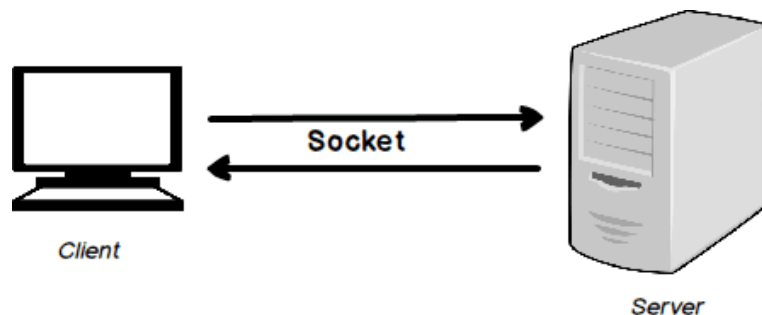
- **Hiệu suất không cao:**
 - So với một số thư viện và ngôn ngữ lập trình game khác, Pygame có thể có hiệu suất không cao đối với các trò chơi phức tạp và đòi hỏi nhiều tài nguyên.
- **Thiếu các tính năng nâng cao:**
 - Pygame không cung cấp nhiều tính năng nâng cao mà một số dự án lớn và phức tạp có thể yêu cầu.
 - Điều này có thể làm giảm khả năng mở rộng của nó trong một số trường hợp.
- **Khả năng tương thích hạn chế:**
 - Pygame có thể không phải là lựa chọn tốt nhất cho các dự án yêu cầu tích hợp sâu với các công nghệ và thư viện khác ngoài Python.
- **Đang trong quá trình phát triển (nếu có):**

- Mặc dù Pygame có một cộng đồng đông đảo, nhưng đôi khi nó vẫn đang trong quá trình phát triển.
- Điều này có thể tạo ra sự không ổn định trong các phiên bản cụ thể.

2.2 Giới thiệu về Socket

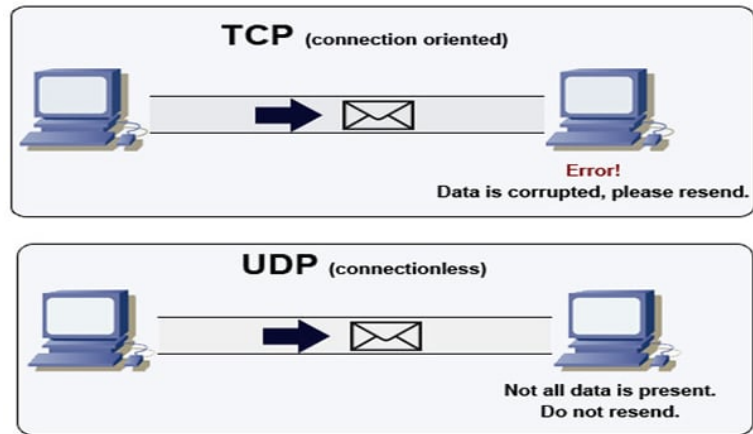
2.2.1 Khái niệm về Socket

- Socket là điểm cuối (end-point) trong liên kết truyền thông hai chiều (two-way communication) giữa máy khách và máy chủ (Client – Server).
- Các lớp Socket được ràng buộc với một cổng (port) cụ thể để các tầng TCP (TCP Layer) định danh ứng dụng mà dữ liệu sẽ được gửi tới.
- Socket là giao diện lập trình ứng dụng mạng dùng để truyền và nhận dữ liệu trên internet.
- Giữa hai chương trình chạy trên mạng cần có một liên kết giao tiếp hai chiều (two-way communication) để kết nối hai process trò chuyện với nhau.
- Điểm cuối (endpoint) của liên kết giữa hai process này được gọi là Socket.



2.2.2 Vai trò của Socket

Nhiều socket được sử dụng liên tục để tiết kiệm thời gian và nâng cao hiệu suất làm việc. Ưu điểm lớn nhất của socket là hỗ trợ hầu hết các hệ điều hành bao gồm MS Windows, Linux, và cũng được sử dụng với nhiều ngôn ngữ lập trình như C, C++, Java, Visual Basic hay Visual C++. Do đó, nó tương thích với hầu hết mọi đối tượng người dùng với những cấu hình máy khác nhau.



2.2.3 Cách thức hoạt động của Socket

- Chức năng của socket là kết nối giữa máy khách và máy chủ thông qua TCP/IP và UDP để truyền và nhận dữ liệu qua Internet.
- Giao diện lập trình ứng dụng mạng này chỉ hoạt động khi biết thông số IP và số hiệu cổng của 2 ứng dụng cần trao đổi dữ liệu cho nhau.
- Điều kiện cần giữa 2 ứng dụng truyền thông tin để socket hoạt động:
 - 2 ứng dụng nằm cùng trên một máy hoặc 2 máy khác nhau.
 - Trường hợp 2 ứng dụng nằm trên cùng một máy, số hiệu cổng không trùng nhau.
- Socket được chia làm 4 loại khác nhau:
 - Stream Socket
 - Datagram Socket
 - Websocket
 - Unix socket

2.2.4 Lịch sử phát triển của Socket

- 1970s: ARPANET phát triển giao thức NCP, và sau đó TCP và IP trở thành tiêu chuẩn.
- 1970-1980: TCP/IP được phát triển và giới thiệu khái niệm "cổng".
- 1980s: BSD Unix bổ sung tính năng cho TCP/IP stack, đưa ra chuẩn de facto cho socket API.

- 1990s: Internet phát triển mạnh mẽ, nhu cầu cho socket tăng lên. Thư viện socket xuất hiện trong các ngôn ngữ như C và C++.
- 1990s-2000s: Java tích hợp thư viện socket trong API, giúp phổ biến hóa socket.
- Hiện đại: Socket vẫn quan trọng trong lập trình mạng, được hỗ trợ trong các ngôn ngữ như Python, JavaScript và Ruby.

2.2.5 Ưu - nhược điểm của Socket

2.2.6 Ưu điểm

- Giao diện lập trình đơn giản: Sockets cung cấp một giao diện lập trình đơn giản cho truyền thông giữa các máy tính trên mạng, cho phép ứng dụng gửi và nhận dữ liệu thông qua các kết nối socket.
- Hiệu suất cao: Sockets cho phép truyền dữ liệu trực tiếp giữa các máy tính mà không cần qua các tầng giao thức trung gian, giúp giảm thiểu độ trễ và tăng hiệu suất.
- Hỗ trợ nhiều giao thức: Sockets không chỉ hỗ trợ TCP mà còn hỗ trợ UDP và các giao thức khác, cho phép phát triển ứng dụng với nhiều mô hình truyền thông khác nhau.
- Cross-platform: Sockets được hỗ trợ trên nhiều hệ điều hành và nền tảng phần cứng, từ máy tính cá nhân đến thiết bị nhúng.
- Tích hợp trong các ngôn ngữ lập trình: Các ngôn ngữ lập trình phổ biến như C/C++, Java, Python, JavaScript đều cung cấp các API hoặc thư viện cho việc sử dụng socket.

2.2.7 Nhược điểm

- Khó sử dụng: Socket có thể phức tạp và khó hiểu đối với người mới bắt đầu, đặc biệt là trong việc xử lý lỗi và quản lý kết nối.
- Tự xử lý nhiều vấn đề: Sử dụng socket đòi hỏi người phát triển phải tự xử lý các vấn đề như quản lý kết nối, bảo mật và đồng bộ hóa dữ liệu.
- Thiếu tính linh hoạt: Socket không cung cấp một số tính năng cao cấp như đa luồng (multithreading) hoặc quản lý trạng thái kết nối, điều này có thể gây hạn chế trong việc xây dựng các ứng dụng phức tạp.
- Khả năng xảy ra lỗi: Trong môi trường mạng không ổn định, việc sử dụng socket có thể dẫn đến các vấn đề như mất kết nối hoặc dữ liệu bị mất, cần phải xử lý cẩn thận để đảm bảo tính ổn định và đáng tin cậy của ứng dụng.

2.3 Các ứng dụng, công nghệ và công cụ thực hiện

2.3.1 Các ứng dụng, công nghệ

- Pygame: Là một thư viện mã nguồn mở trong Python được thiết kế để phát triển game 2D. Nó cung cấp chức năng cho việc vẽ đồ họa, xử lý sự kiện và âm thanh, giúp tập trung vào logic game.
- Panda: Là một engine game mã nguồn mở cung cấp nền tảng cho phát triển game. Nó hỗ trợ Python làm ngôn ngữ kịch bản và tính năng như xử lý đồ họa, âm thanh và vật lý.
- IDE (Integrated Development Environment): Sử dụng một IDE như Pycharm.
- Pygame Mixer, PIL (Python Imaging Library), Pillow: Thư viện xử lý âm thanh và hình ảnh
- Git và GitHub: Sử dụng Git để theo dõi phiên bản và GitHub để lưu trữ mã nguồn của dự án.

2.3.2 Công cụ thực hiện

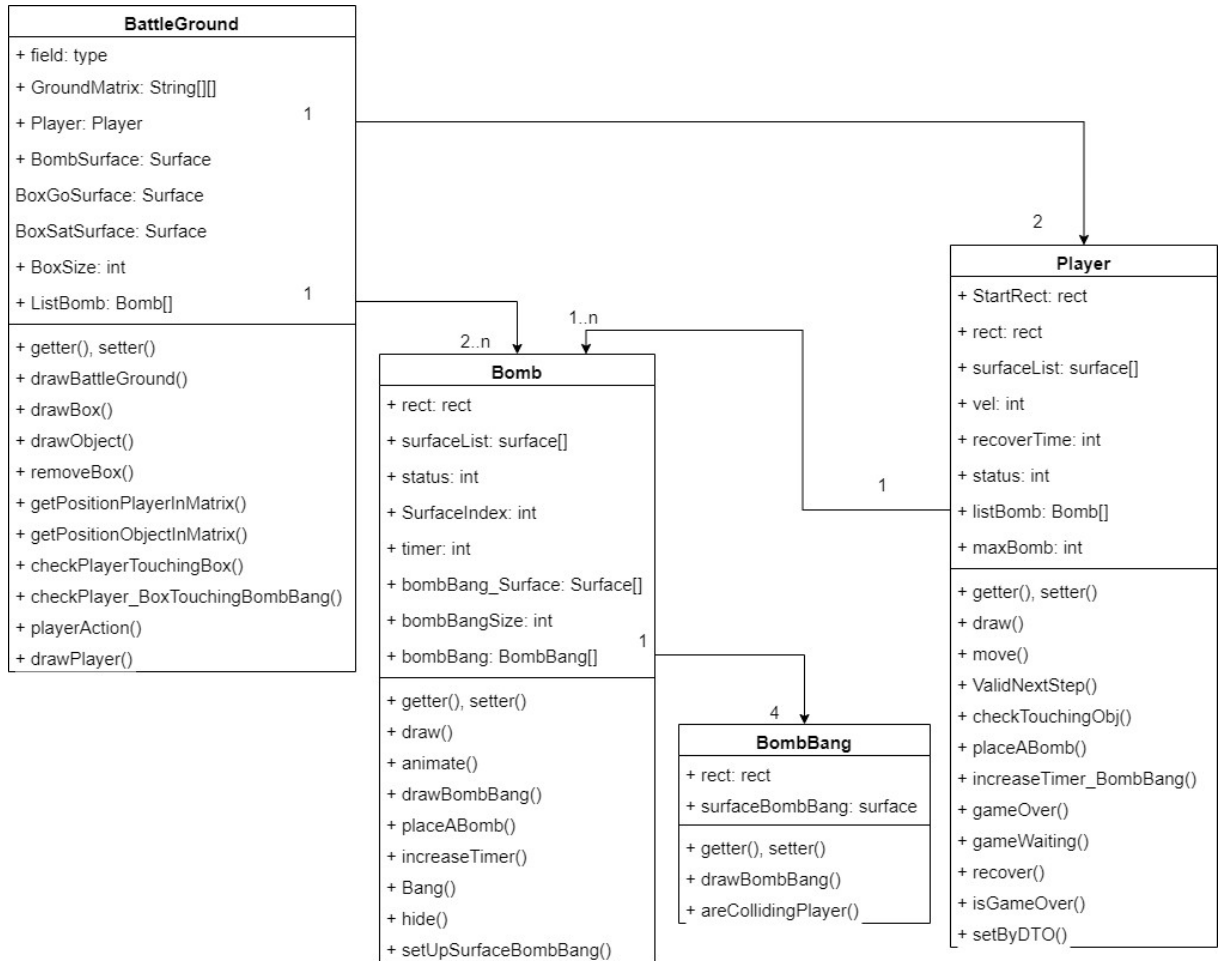
Sử dụng Pycharm JetBrains để thực hiện đề tài “Xây dựng trò chơi Bomberman”.

Một số đặc điểm của Pycharm bao gồm:

- Hỗ trợ Python: PyCharm là một IDE chuyên dụng cho Python, cung cấp nhiều tính năng và công cụ hỗ trợ phát triển ứng dụng Python, bao gồm cả game.
- Giao diện sử dụng thân thiện: PyCharm có một giao diện sử dụng thân thiện và dễ sử dụng, giúp cho việc phát triển game trở nên dễ dàng và hiệu quả.
- Gỡ lỗi và phân tích mã nguồn: PyCharm cung cấp khả năng gỡ lỗi tích hợp và phân tích mã nguồn, giúp bạn dễ dàng theo dõi và sửa lỗi trong mã nguồn của trò chơi Bomberman.
- Tích hợp với các công cụ phát triển game: PyCharm có thể tích hợp với các công cụ phát triển game như Pygame, một thư viện phổ biến cho việc phát triển game 2D với Python, giúp bạn dễ dàng xây dựng trò chơi Bomberman.
- Hỗ trợ quản lý dự án: PyCharm cung cấp các tính năng quản lý dự án như Git Integration để theo dõi thay đổi mã nguồn và quản lý phiên bản của trò chơi Bomberman.
- Tính di động và linh hoạt: PyCharm có thể sử dụng trên nhiều hệ điều hành khác nhau như Windows, macOS và Linux, đảm bảo tính di động và linh hoạt trong việc phát triển trò chơi Bomberman trên các nền tảng khác nhau.

CHƯƠNG 3. PHÂN TÍCH THIẾT KẾ

3.1 Sơ đồ class Bomber

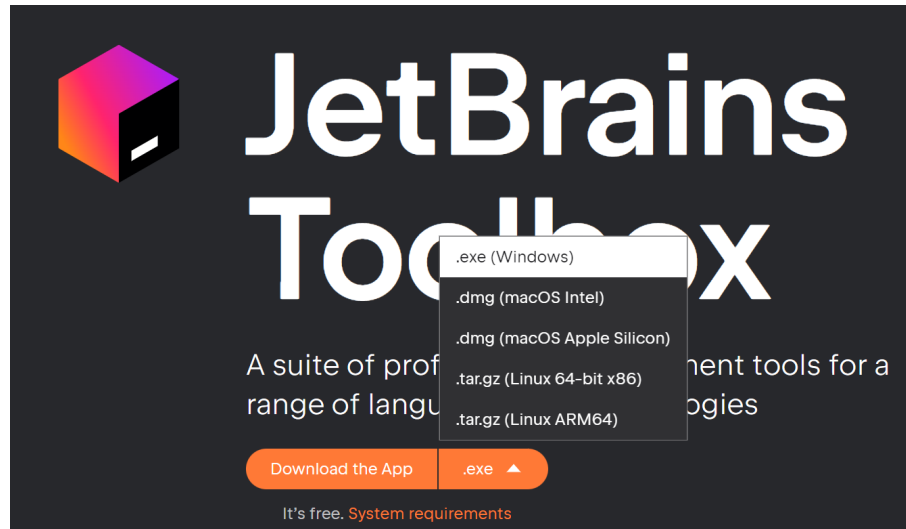


CHƯƠNG 4. CÀI ĐẶT MÔI TRƯỜNG

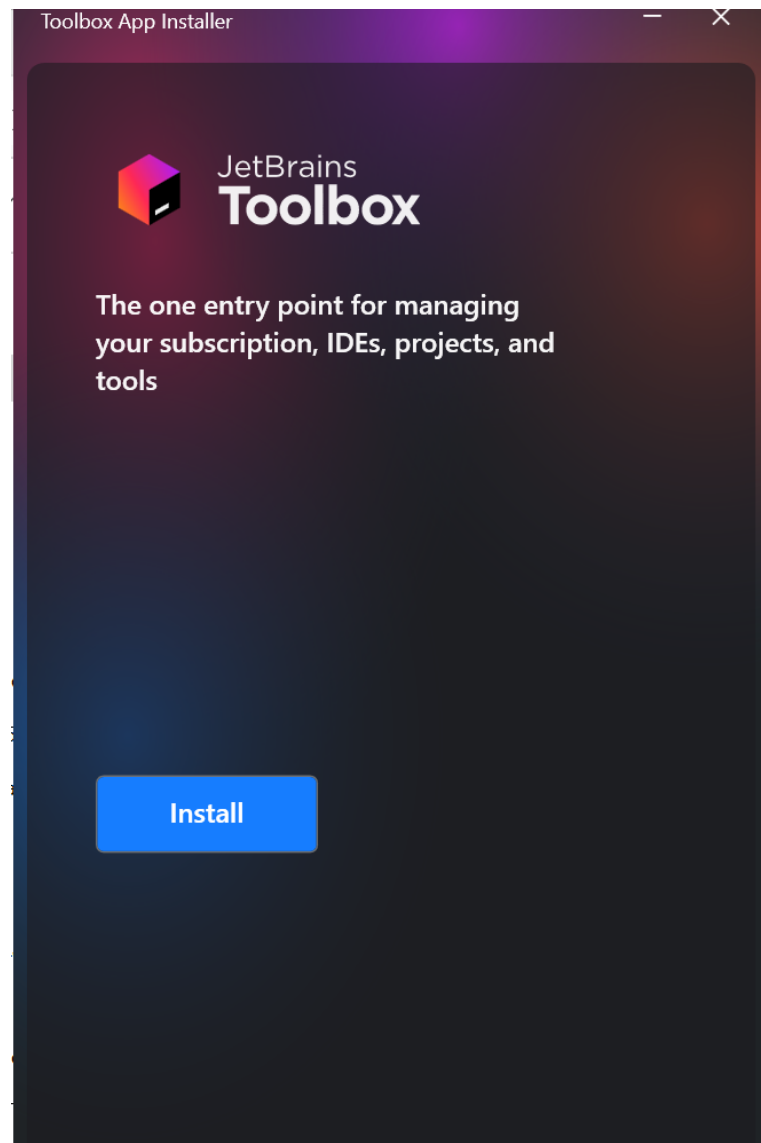
4.1 Cài đặt Pycharm JetBrain sử dụng Toolbox App

PyCharm là một IDE đa nền tảng cung cấp trải nghiệm nhất quán trên các hệ điều hành Windows, macOS và Linux.

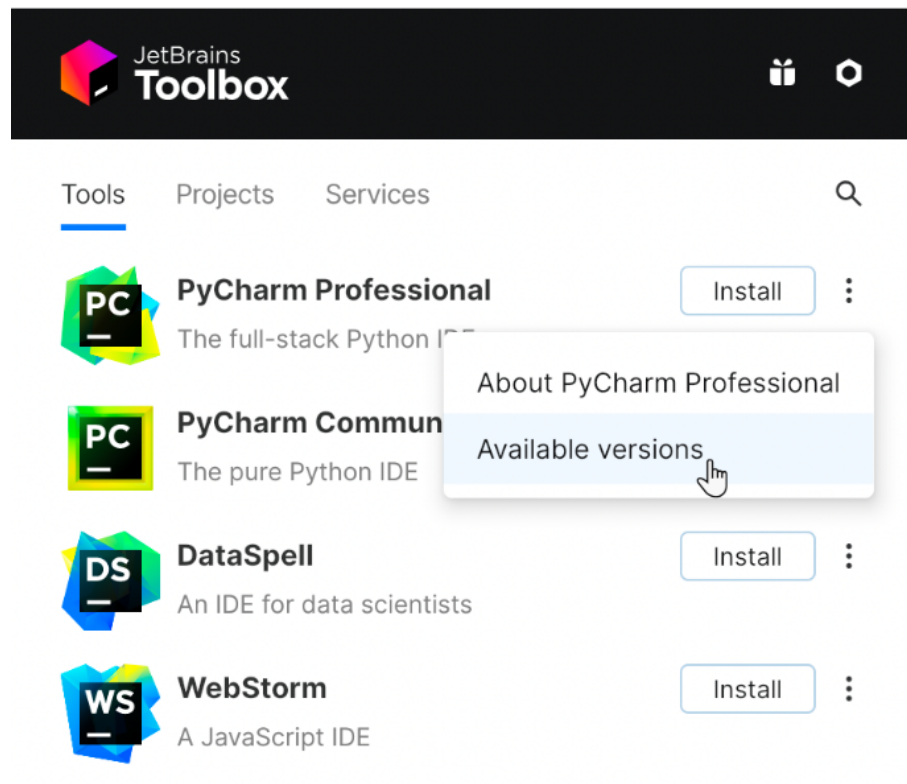
Bước 1: Download ToolBox App



Bước 2: Install Toolbox app



Bước 3: Chọn phiên bản Pycharm (Community hoặc Professional), chọn version, ấn install




Bước 4: Sau khi cài đặt Pycharm thành công

Tools²ProjectsServices


🔍

Installed


Update all




IntelliJ IDEA Ultimate
2024.1
[What's new in 2024.1.1](#)



Android Studio by Google
Jellyfish 2023.3.1 RC 2
[What's new in Koala 2024.1.1 B](#)



PhpStorm
2024.1.1



PyCharm Professional
2024.1.1

Update

⋮

Settings

What's new

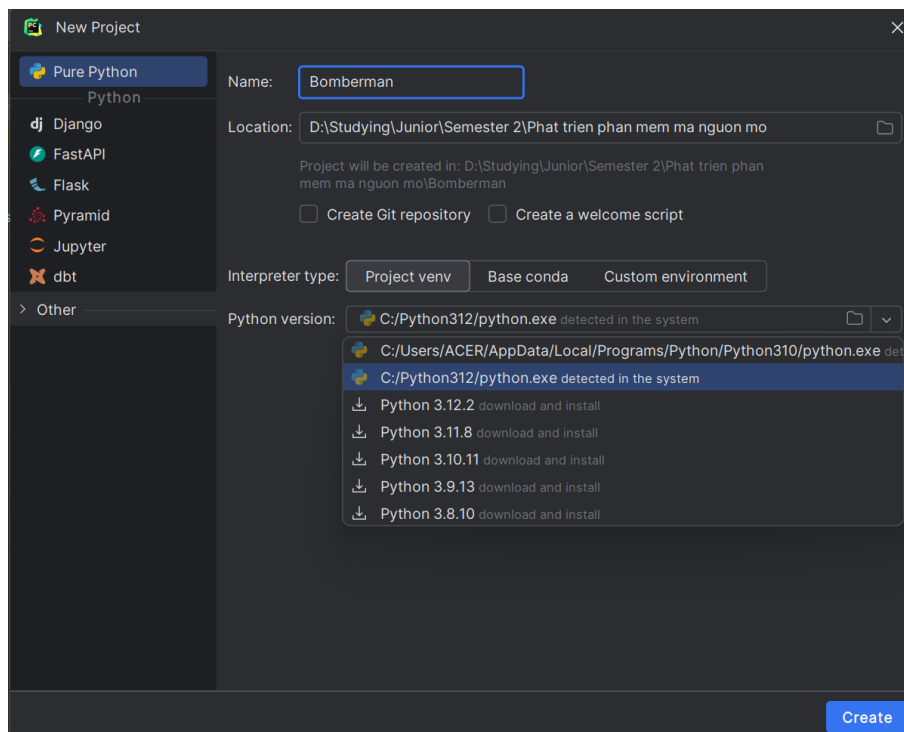
Other versions

Run as administrator

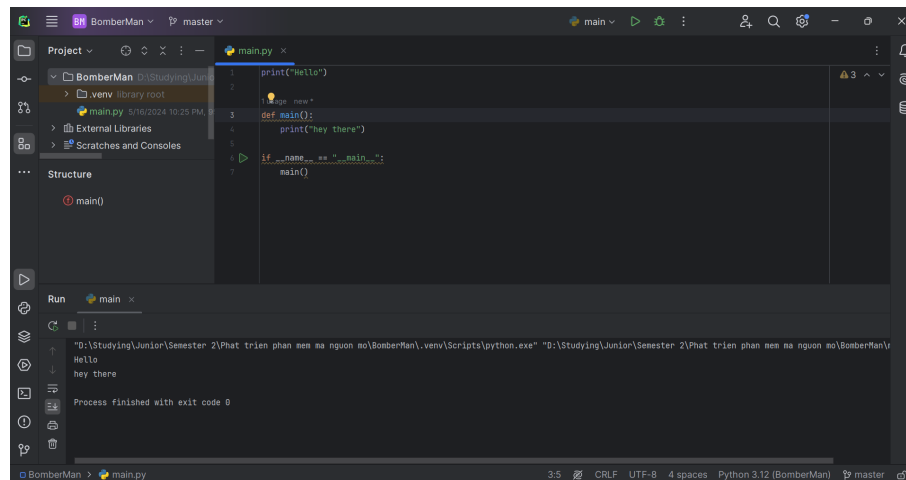
Uninstall...

⋮

Bước 5: Tạo project đầu tiên, chọn phiên bản Python đã có hoặc download và install phiên bản khác

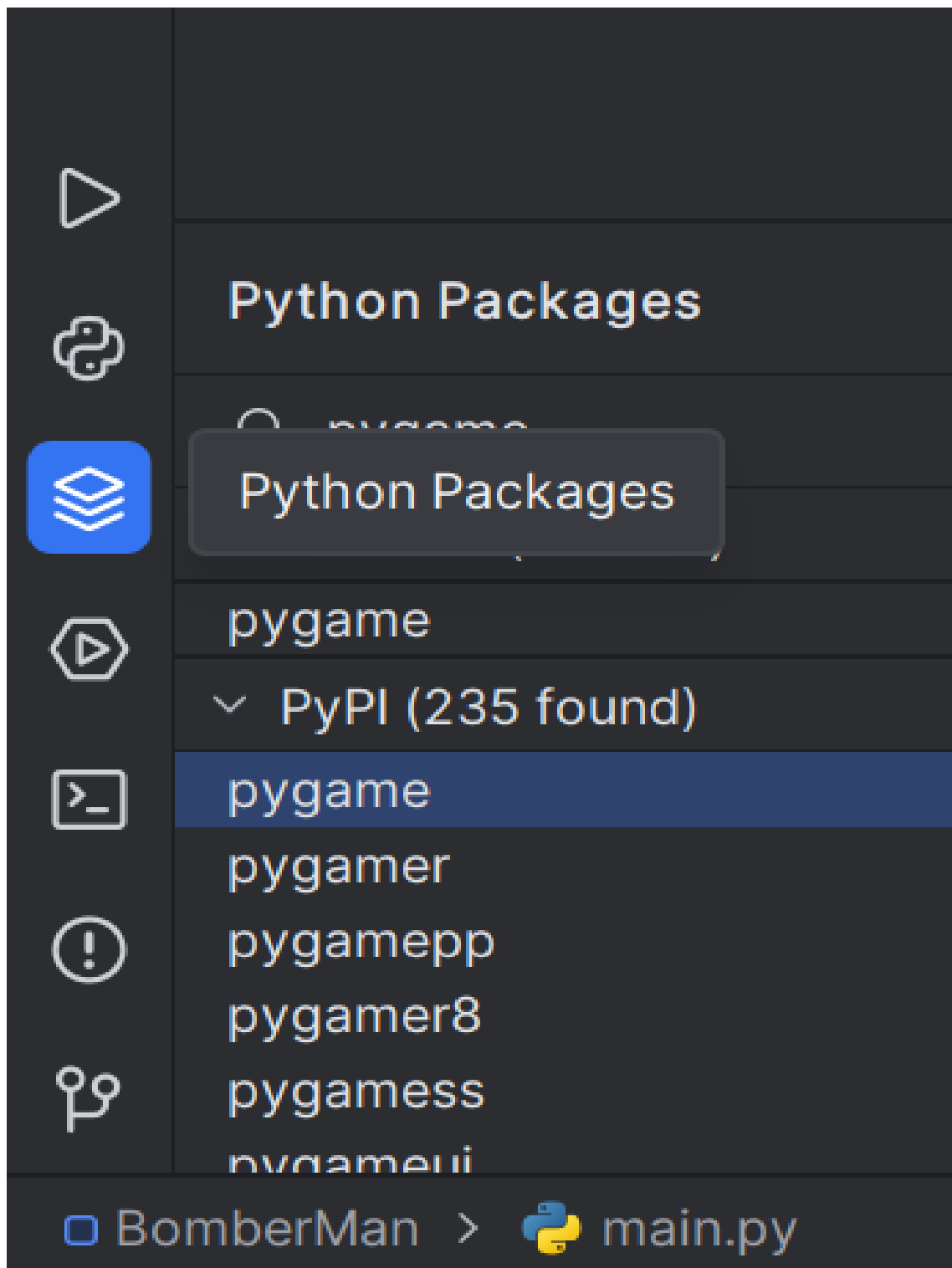


Bước 6: Chạy main.py

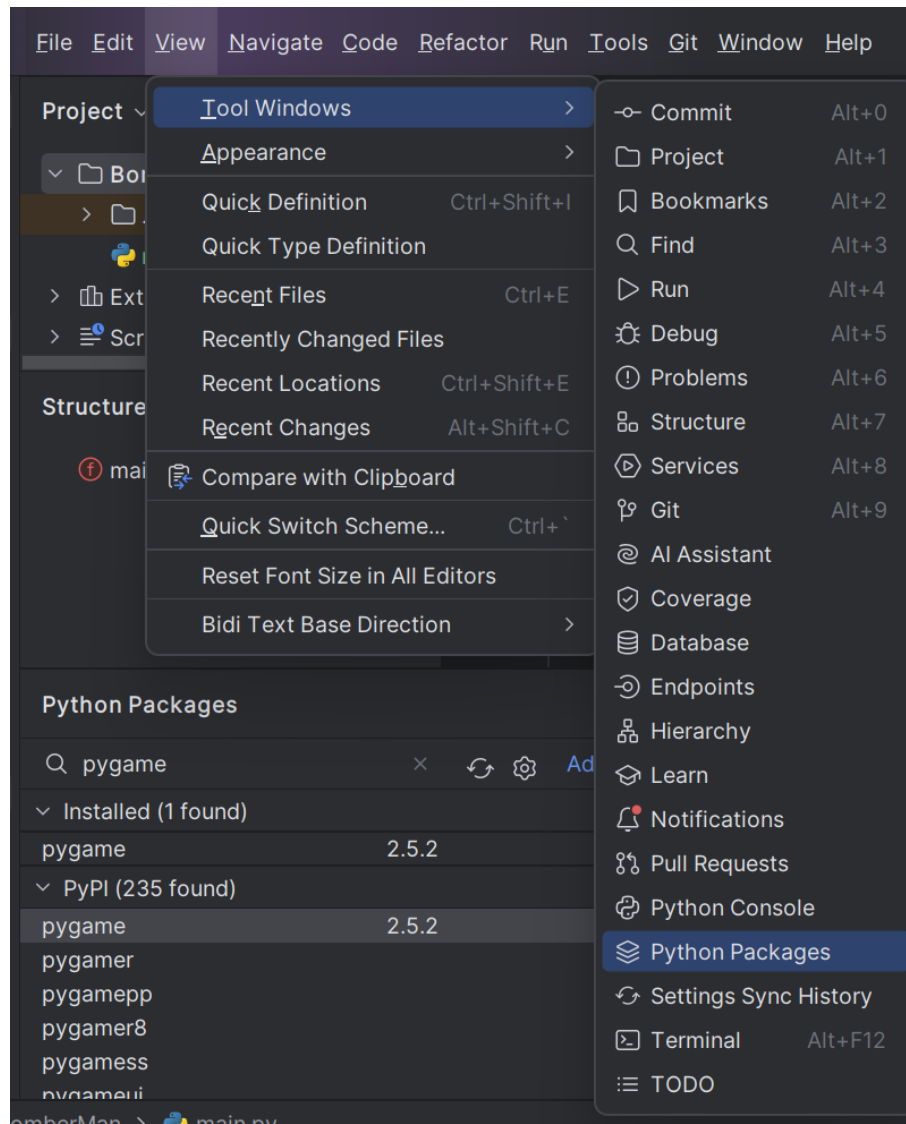


4.2 Install package trong Pycharm

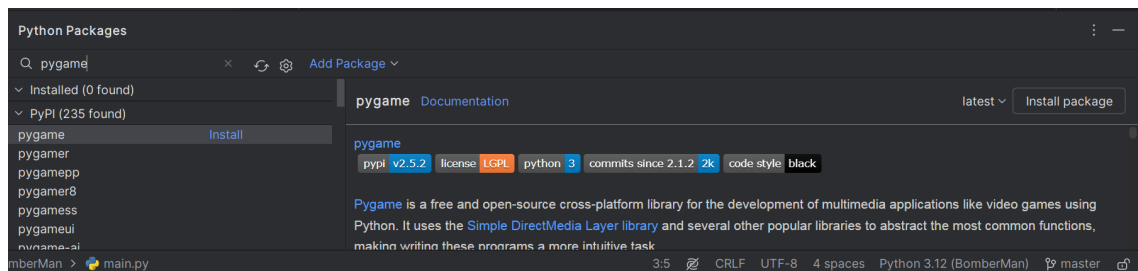
Bước 1: Mở Python Package trên Nabar



Hoặc: Chọn "View" -> "Tool Window" -> "Python Package"



Bước 2: Tìm kiếm và cài đặt thư viện



CHƯƠNG 5. XÂY DỰNG VÀ PHÁT TRIỂN TRÒ CHƠI BOMBERMAN

5.1 Xây dựng trò chơi Bomberman

5.1.1 Khởi tạo đối tượng *Player*

```
4 usages (4 dynamic)  LinhDancute
def draw(self, screen):
    screen.blit(self.surface, self.rect)

LinhDancute
def update(self, playerRect):
    self.rect = playerRect

LinhDancute
def setSurface_Rect(self, playerRect, indexSurface):
    self.rect = playerRect
    self.surface = self.surfaceList[indexSurface]
```

Định nghĩa lớp Player trong game Bomberman SocketPython.

- rect: Đây là hình chữ nhật mô tả vị trí và kích thước của player trên màn hình.
- surfaceList: Là danh sách các hình ảnh của player tương ứng với các hướng di chuyển khác nhau.
- bomb: Đối tượng Bomb của player.
- startRect: Lưu trữ vị trí ban đầu của player.
- vel: Tốc độ di chuyển của player.
- surface: Hình ảnh của player hiện tại.

- maxBomb: Số lượng bom tối đa mà player có thể đặt.
- listBomb: Danh sách các đối tượng bom của player.
- status: Trạng thái của player, có thể là 1 (còn sống) hoặc 0 (chết).
- recoverTime: Thời gian hồi phục sau khi chết của player.

5.1.2 Các phương thức vẽ và cập nhật của Player

```

LinhDancute
def __init__(self, rect, surfaceList, bomb):
    self.startRect = [rect.centerx, rect.centery]
    self.rect = rect
    self.vel = 4
    self.surfaceList = surfaceList
    self.surface = surfaceList[0]
    self.maxBomb = 1
    self.listBomb = bomb
    self.status = 1
    self.recoverTime = 0
    self.flag_StartGame = 0

```

- draw(self, screen): Vẽ hình ảnh của người chơi lên màn hình.
- update(self, plapyerRect): Cập nhật vị trí của người chơi dựa trên hộp giới hạn mới.
- surfaceList: Là danh sách các hình ảnh của player tương ứng với các hướng di chuyển khác nhau.
- setSurface_Rect(self, playerRect, indexSurface): Cập nhật hình ảnh và hộp giới hạn mới cho người chơi.

5.1.3 Phương thức di chuyển của Player

```
8 usages (8 dynamic)  👤 LinhDancute
def move(self, direction):
    if self.status == 1:
        if direction == "left":
            if self.ValidNextStep("left"):
                self.rect.centerx -= self.vel
        if direction == "right":
            if self.ValidNextStep("right"):
                self.rect.centerx += self.vel
        if direction == "up":
            if self.ValidNextStep("up"):
                self.rect.centery -= self.vel
        if direction == "down":
            if self.ValidNextStep("down"):
                self.rect.centery += self.vel
```

Phương thức move trong đoạn mã này cho phép di chuyển Player theo hướng chỉ định. Nó kiểm tra trạng thái của Player (self.status) để đảm bảo rằng Player có thể di chuyển trước khi thực hiện hành động. Sau đó, nó kiểm tra hướng di chuyển được chỉ định và kiểm tra xem bước tiếp theo có hợp lệ không thông qua phương thức ValidNextStep.

- Nếu hướng là "left" (trái), nó sẽ giảm giá trị centerx của rect đi self.vel nếu bước tiếp theo là hợp lệ.
- Nếu hướng là "right" (phải), nó sẽ tăng centerx.
- Nếu hướng là "up" (lên), nó sẽ giảm giá trị centery của rect đi self.vel.
- Nếu hướng là "down" (xuống), nó sẽ tăng centery.

Trong mỗi trường hợp, nó kiểm tra xem bước tiếp theo có hợp lệ không trước khi di chuyển.

5.1.4 Kiểm tra hướng di chuyển hợp lệ

```
4 usages  LinhDancute
def ValidNextStep(self, direction):
    if direction == "left":
        if self.rect.centerx - self.vel <= 50:
            return False
    if direction == "right":
        if self.rect.centerx + self.vel >= 800-50:
            return False
    if direction == "up":
        if self.rect.centery - self.vel <= 0+50:
            return False
    if direction == "down":
        if self.rect.centery + self.vel >= 800-50:
            return False
    return True
```

- Phương thức này kiểm tra xem người chơi có thể di chuyển vào hướng tiếp theo được không.

- Đối với mỗi hướng di chuyển (trái, phải, lên, xuống), nó kiểm tra xem việc di chuyển sẽ đưa ra khỏi ranh giới của màn hình hoặc vào trong vùng cấm (ví dụ như khu vực hiển thị trò chơi).

- Nếu điều kiện nào đó không thỏa mãn, nó sẽ trả về False, ngược lại sẽ trả về True, cho phép người chơi di chuyển.

5.1.5 Kiểm tra va chạm và lấy thông tin vị trí và hình ảnh của player

```
10 usages (10 dynamic)  LinhDancute
def checkTouchingObj(self, Obj):
    if Obj == 0:
        return False
    if self.rect.colliderect(Obj):
        return True
    return False

2 usages (2 dynamic)  LinhDancute
def getRect(self):
    return self.rect

LinhDancute
def getSurface(self):
    return self.surface
```

- checkTouchingObj(self, obj): kiểm tra xem player có va chạm với một object nào đó không. Nếu player va chạm với object, hàm trả về True, ngược lại trả về False.
- getRect(self): trả về hình chữ nhật đại diện cho vị trí của player.
- getSurface(self): trả về hình ảnh hiển thị của player.


```

1 usage (1 dynamic)  LinhDancute
def placeABomb(self, bombRectx, bombRecty, sound_PlaceBomb):
    for i in range(len(self.listBomb)):
        if self.listBomb[i].getStatus() == 0:
            self.listBomb[i].setStatus(1)
            self.listBomb[i].setRectWithPosition(bombRectx, bombRecty)
            sound_PlaceBomb.play()
            break

```

5.1.6 Đặt bom và lấy danh sách bom

Định nghĩa phương thức placeABomb() để đặt một quả bom tại vị trí xác định trên màn hình. Phương thức này nhận vào hai đối số bombRectx và bombRecty, là tọa độ x và y của vị trí muốn đặt bom, cũng như một âm thanh sound_PlaceBomb được phát khi đặt bom.

- Trong phương thức placeABomb(), vòng lặp duyệt qua danh sách bom của người chơi để tìm một quả bom chưa được sử dụng.
- Nếu tìm thấy một quả bom có trạng thái là 0 (chưa được sử dụng), phương thức đặt trạng thái của nó thành 1 (đã được đặt).
- Cập nhật vị trí của quả bom dựa trên bombRectx và bombRecty.
- Phát âm thanh sound_PlaceBomb.
- Kết thúc vòng lặp và thoát khỏi phương thức.
- Phương thức getListBomb() trả về danh sách các quả bom của người chơi.

5.1.7 Xử lý thời gian đếm ngược và nổ bom của người chơi

- increaseTimer_BombBang(self, value, sound_BombBang): Phương thức này được sử dụng để tăng thời gian đếm ngược cho các quả bom của người chơi và phát âm thanh khi quả bom nổ.
- surface: value: Đại diện cho giá trị thời gian được tăng lên cho mỗi vòng lặp.
- sound_BombBang: Âm thanh được phát khi quả bom nổ.

```

1 usage  LinhDancute
def increaseTimer_BombBang(self, value, sound_BombBang):
    for bomb in self.listBomb:
        if bomb.getStatus()==1 or bomb.getStatus()==2:
            bomb.increaseTimer(value)
            if bomb.getTimer() == 2000:
                bomb.Bang()
                sound_BombBang.play()
            if bomb.getTimer() == 2500:
                bomb.hide()

```

Trong hàm này:

- Duyệt qua danh sách các quả bom trong trò chơi.
- Nếu trạng thái của quả bom là 1 hoặc 2 (tức là quả bom đang chờ nổ hoặc đang trong quá trình nổ):
 - Tăng thời gian của quả bom lên value.
 - Nếu thời gian của quả bom đạt đến 2000 (2 giây):
 - * Gọi hàm Bang() của quả bom để nổ.
 - * Phát âm thanh của quả bom nổ.
 - Nếu thời gian của quả bom đạt đến 2500 (2.5 giây): Ẩn quả bom.

5.1.8 Xử lý kết thúc và chờ đợi trong trò chơi

gameOver(self) và gameWaiting(self): được sử dụng để thay đổi trạng thái của người chơi khi trò chơi kết thúc hoặc khi đang chờ đợi.

- Trong hàm gameOver, trạng thái của người chơi được đặt thành 0, và bề mặt (surface) của người chơi được thay đổi thành một hình ảnh khác (thường là hình ảnh của người chơi bị thua cuộc).
- Trong hàm gameWaiting, trạng thái của người chơi được đặt thành 2, và bề mặt của người chơi cũng được thay đổi thành một hình ảnh khác (thường là hình ảnh của người chơi đang chờ đợi).

```

2 usages (2 dynamic)  LinhDancute
def gameOver(self):
    self.status = 0
    self.surface = self.surfaceList[1]

2 usages  LinhDancute
def gameWaiting(self):
    self.status = 2
    self.surface = self.surfaceList[1]

```

5.1.9 *Khôi phục và kiểm tra kết thúc game cho người chơi*

```

2 usages  LinhDancute
def recover(self):
    self.surface = self.surfaceList[0]
    self.rect.centerx = self.startRect[0]
    self.rect.centery = self.startRect[1]
    self.status = 1

2 usages  LinhDancute
def isGameOver(self):
    if self.status == 0:
        return True
    return False

```

- Phương thức recover: Được sử dụng để khôi phục trạng thái của người chơi sau khi kết thúc một ván chơi. Khi gọi phương thức này, người chơi sẽ được di chuyển về vị trí ban

đầu của mình, hình ảnh của người chơi cũng sẽ được chuyển về hình ảnh bình thường, và trạng thái của người chơi sẽ được đặt lại thành 1 (đang hoạt động).

- Phương thức `isGameOver`: Được sử dụng để kiểm tra xem trạng thái của người chơi có phải là kết thúc game hay không. Nếu trạng thái của người chơi là 0 (kết thúc game), phương thức trả về `True`, ngược lại trả về `False`.

5.1.10 Truy xuất thuộc tính người chơi

```
1 usage (1 dynamic)  LinhDancute
def getCenterx(self):
    return self.rect.centerx

1 usage (1 dynamic)  LinhDancute
def getCentery(self):
    return self.rect.centery

1 usage (1 dynamic)  LinhDancute
def getVel(self):
    return self.vel
```

Định nghĩa các phương thức để truy xuất các thuộc tính của đối tượng người chơi.

- Phương thức `getCenterx`: Trả về tọa độ x của trung tâm của hình chữ nhật đại diện cho người chơi.
- Phương thức `getCentery`: Trả về tọa độ y của trung tâm của hình chữ nhật đại diện cho người chơi.
- Phương thức `getVel`: Trả về vận tốc di chuyển của người chơi.

5.1.11 Phương thức truy xuất thông tin người chơi

* `getMaxBomb(self)`: Trả về số lượng bom tối đa mà người chơi có thể giữ được.

```
1 usage (1 dynamic)  LinhDancute
def getMaxBomb(self):
    return self.maxBomb

8 usages (8 dynamic)  LinhDancute
def getStatus(self):
    return self.status

1 usage (1 dynamic)  LinhDancute
def getRecoverTime(self):
    return self.recoverTime
```

Biến trả về: maxBomb - Số lượng bom tối đa.

getStatus(self): Trả về trạng thái hiện tại của người chơi (0 - Game over, 1 - Đang hoạt động, 2 - Đang chờ).

Biến trả về: status - Trạng thái của người chơi.

getRecoverTime(self): Trả về thời gian hồi phục của người chơi sau khi bị loại khỏi trò chơi (đối với trạng thái chờ).

Biến trả về: recoverTime - Thời gian hồi phục của người chơi.

5.1.12 Cập nhật thông tin người chơi từ DTO

2 usages LinhDancute

```
def setByDTO(self, playerDTO):  
    self.rect.centerx = playerDTO.get_centerx()  
    self.rect.centery = playerDTO.get_centery()  
    self.status = playerDTO.get_status()  
    self.vel = playerDTO.get_vel()  
    self.maxBomb = playerDTO.get_maxBomb()  
    self.recoverTime = playerDTO.get_recoverTime()  
    self.listBomb = []  
    for bombDTO in playerDTO.get_ListBomb():  
        self.listBomb.append(bombConvert().toBomb(bombDTO))
```

Được sử dụng để cập nhật thông tin của đối tượng Player từ một đối tượng DTO (Data Transfer Object).

- playerDTO: Là một đối tượng chứa thông tin của người chơi, bao gồm vị trí, trạng thái, tốc độ, số lượng bom, thời gian hồi phục và danh sách bom.
- self.rect.centerx: Cập nhật vị trí trục x của người chơi từ playerDTO.
- self.rect.centery: Cập nhật vị trí trục y của người chơi từ playerDTO.
- self.status: Cập nhật trạng thái của người chơi từ playerDTO.
- self.vel: Cập nhật tốc độ của người chơi từ playerDTO.
- self.maxBomb: Cập nhật số lượng bom tối đa của người chơi từ playerDTO.
- self.recoverTime: Cập nhật thời gian hồi phục của người chơi từ playerDTO.
- self.listBomb: Khởi tạo danh sách bom của người chơi từ playerDTO, bằng cách chuyển đổi từng đối tượng bombDTO trong danh sách bom của playerDTO sang đối tượng bom và thêm vào danh sách listBomb của người chơi.

```

LinhDancute
def __init__(self, surfaceBombBang):
    self.rect = 1000
    self.surfaceBombBang = surfaceBombBang
    self.bombExplodeSize = 1

```

5.1.13 Khởi tạo hiệu ứng nổ bom

Khởi tạo của một lớp bombBang trong game.

- surfaceBombBang: là hình ảnh hoặc bề mặt đại diện cho hiệu ứng nổ bom.
- rect: là vị trí ban đầu của hiệu ứng nổ bom trên màn hình. Trong đoạn code này, nó được đặt là 1000, có thể là một giá trị không hợp lệ.
- bombExplodeSize: là kích thước của hiệu ứng nổ bom, thường được sử dụng để xác định phạm vi tác động của hiệu ứng nổ. Trong đoạn code này, giá trị ban đầu là 1.

5.1.14 Vẽ hiệu ứng nổ bom

```

LinhDancute
def setRect(self, rect):
    self.rect = rect

2 usages (2 dynamic)  LinhDancute
def getRect(self):
    return self.rect

1 usage (1 dynamic)  LinhDancute
def hide(self):
    self.rect = self.surfaceBombBang.get_rect(center=(-1000, 0))

```

Hàm drawBombBang dùng để vẽ hiệu ứng nổ bom lên màn hình.

- screen: Màn hình pygame.

- self.surfaceBombBang: Hình ảnh của hiệu ứng nổ bom.
- self.rect: Vị trí của hiệu ứng nổ bom trên màn hình.

5.1.15 Thiết lập tọa độ cho hiệu ứng nổ bom

```
3 usages (2 dynamic)  LinhDancute
def drawBombBang(self, screen):
    screen.blit(self.surfaceBombBang, self.rect)
```

-Phương thức setRectCenterx(self, value) được sử dụng để đặt tọa độ x của tâm của hình chữ nhật.

-Phương thức setRectCentery(self, value) được sử dụng để đặt tọa độ y của tâm của hình chữ nhật.

-Phương thức setRectCenterX_Y(self, centerx, centery) đặt tọa độ của tâm của hình chữ nhật theo hai giá trị centerx và centery cho trước.

5.1.16 Quản lý hiệu ứng nổ bom

```
4 usages (4 dynamic)  LinhDancute
def setRectCenterx(self, value):
    self.rect.centerx=value

LinhDancute
def setRectCentery(self, value):
    self.rect.centery=value

4 usages  LinhDancute
def setRectCenterX_Y(self, centerx, centery):
    self.rect = self.surfaceBombBang.get_rect(center=(centerx, centery))
```

-Hàm setRect: Thiết lập tọa độ và kích thước cho hiệu ứng nổ bom.

-Hàm getRect: Trả về hình chữ nhật đại diện cho hiệu ứng nổ bom.

-Hàm hide: Ẩn hiệu ứng nổ bom bằng cách đặt tọa độ ngoài màn hình.

5.1.17 Kiểm tra va chạm với người chơi

```

2 usages (2 dynamic)  LinhDancute
def areCollidingPlayer(self, obj):
    if self.rect.colliderect(obj):
        return True
    return False

```

-Hàm areCollidingPlayer kiểm tra xem 2 đối tượng này có va chạm với nhau không. Đối tượng được truyền vào qua tham số obj.

-Nếu hai đối tượng va chạm, tức là phần diện tích của chúng giao nhau, hàm trả về True, ngược lại trả về False.

5.1.18 Khởi tạo trường đánh và các đối tượng trong trò chơi

```

def __init__(self, groundMatrix, player, player2, bombSurface, boxGoSurface,
              boxSatSurface, itemBombSurface, itemBombSizeSurface):
    self.groundMatrix = groundMatrix
    self.player = player
    self.player2 = player2
    self.bombSurface = bombSurface
    self.boxGoSurface = boxGoSurface
    self.boxSatSurface = boxSatSurface
    self.itemBombSurface = itemBombSurface
    self.itemBombSizeSurface = itemBombSizeSurface
    self.boxSize = 50
    self.groundRect = [[0 for _ in range(17)] for _ in range(17)]
    self.listBomb = []
    self.listBomb.append(player.getListBomb())
    self.listBomb.append(player2.getListBomb())
    self.rectTemp = self.boxGoSurface.get_rect(center=(-1000, 0))

```

Phương thức khởi tạo (__init__):

- groundMatrix: Là ma trận đại diện cho mặt đất trong trò chơi. Mỗi phần tử của ma trận này đại diện cho một ô trên mặt đất, và có thể là các giá trị như 'g' (mặt đất), 's' (hộp không di chuyển được), '-' (ô trống) và các giá trị khác tùy thuộc vào quy ước của trò chơi.

- `player`: Đối tượng của người chơi 1.
- `player2`: Đối tượng của người chơi 2.
- `bombSurface`: Là bề mặt của quả bom.
- `boxGoSurface`: Là bề mặt của hộp di chuyển được.
- `boxSatSurface`: Là bề mặt của hộp không di chuyển được.
- `boxSize`: Kích thước của mỗi ô trên mặt đất, cũng là kích thước của hộp.
- `groundRect`: Một mảng hai chiều, chứa các hình chữ nhật đại diện cho từng ô trên mặt đất. Đối tượng này được sử dụng để vẽ hình chữ nhật đại diện cho mặt đất trên màn hình.
- `listBomb`: Danh sách chứa tất cả các quả bom trong trò chơi, bao gồm cả quả bom của cả hai người chơi.
- `rectTemp`: Là hình chữ nhật tạm thời được sử dụng trong quá trình thực hiện các thao tác về hộp.

5.1.19 Vẽ trường đánh

-Hàm `drawBattleGround` nhận một màn hình (`screen`) làm đối số và vẽ trường đánh trên màn hình đó.

-Hai vòng lặp được sử dụng để duyệt qua từng ô của ma trận `groundMatrix`, biểu diễn các ô trên trường đánh.

- Vòng lặp bên ngoài duyệt qua các hàng của ma trận.
- Vòng lặp bên trong duyệt qua các cột của hàng hiện tại.

-Trong mỗi vòng lặp, giá trị của từng ô trong ma trận được kiểm tra:

- Nếu giá trị là 's' (đại diện cho ô chứa hộp sắt), hàm `drawBox` được gọi để vẽ hộp sắt tại vị trí tương ứng trên màn hình.
- Nếu giá trị là 'g' (đại diện cho ô trống), hàm `drawBox` được gọi để vẽ ô trống tại vị trí tương ứng.

```

1 usage (1 dynamic)  LinhDancute
def drawBattleGround(self, screen):
    for i in range(len(self.groundMatrix)): # Duyệt qua các hàng
        for j in range(len(self.groundMatrix)): # Duyệt qua các cột của hàng đó
            value = self.groundMatrix[i][j]
            if value == 's':
                self.groundRect[i][j] = self.boxSatSurface.get_rect(center=((j+1)*self.boxSize, (i+1)*self.boxSize))
                self.drawBox(self.boxSatSurface, self.groundRect[i][j], screen)
            if value == 'g':
                self.groundRect[i][j] = self.boxGoSurface.get_rect(center=((j+1)*self.boxSize, (i+1)*self.boxSize))
                self.drawBox(self.boxGoSurface, self.groundRect[i][j], screen)
            if value == 'i_b':
                self.groundRect[i][j] = self.itemBombSurface.get_rect(center=((j+1)*self.boxSize, (i+1)*self.boxSize))
                self.drawBox(self.itemBombSurface, self.groundRect[i][j], screen)
            if value == 'i_bs':
                self.groundRect[i][j] = self.itemBombSizeSurface.get_rect(center=((j+1)*self.boxSize, (i+1)*self.boxSize))
                self.drawBox(self.itemBombSizeSurface, self.groundRect[i][j], screen)
            if value == '-':
                if self.groundRect[i][j] != 0:
                    self.groundRect[i][j].centerx = -1000

            # Lấy giá trị và chỉ mục của phần tử tại hàng i, cột j

            index = (i, j)
self.drawObject(screen)

```

- Nếu giá trị là '-' (đại diện cho ô không có gì), vị trí của ô trên màn hình được đặt xa ra (x = -1000) để ẩn nó đi.

-Sau khi duyệt qua toàn bộ ma trận, hàm drawObject được gọi để vẽ các đối tượng (người chơi và bom) trên trường đánh.

5.1.20 Xóa ô hộp khỏi ma trận và màn hình

-Xử lý việc loại bỏ ô hộp (box) khỏi ma trận đất (ground matrix) và cập nhật trạng thái của ma trận và hiển thị trên màn hình.

-removeBox(self, i, j, screen): Phương thức này nhận các tham số i và j, đại diện cho vị trí của ô trong ma trận đất, và screen, đại diện cho màn hình trò chơi.

-Nếu giá trị của ô trong ma trận là 'g' (ô chứa hộp):

- self.groundRect[i][j] = self.rectTemp: Cập nhật hình dạng của ô chứa hộp về giá trị mặc định (self.rectTemp).
- screen.blit(self.boxGoSurface, self.groundRect[i][j]): Vẽ hộp trên màn hình với hình dạng mới.

4 usages LinhDancute

```
def drawBox(self, boxSurface, boxRect, screen):  
    # Vẽ Box  
    screen.blit(boxSurface, boxRect)
```

1 usage LinhDancute

```
def drawObject(self, screen):  
    # Vẽ Bomb  
    for bomb in self.player.getListBomb():  
        if bomb.getStatus() <= 1:  
            bomb.draw(screen)  
        else:  
            bomb.drawBombBang(screen)  
    for bomb in self.player2.getListBomb():  
        if bomb.getStatus() <= 1:  
            bomb.draw(screen)  
        else:  
            bomb.drawBombBang(screen)  
  
    # Vẽ Player  
    self.drawPlayer(screen)
```

- `self.groundMatrix[i][j] = '-'`: Cập nhật giá trị của ô trong ma trận về '-' để chỉ ra rằng ô này không còn chứa hộp nữa.

5.1.21 Vẽ các phần tử trong trò chơi

```

4 usages  LinhDancute
def removeBox(self, i, j, screen):
    if self.groundMatrix[i][j] == 'g':
        # print(self.groundRect[i][j].centerx, '-', self.groundRect[i][j].centery)
        randomItem = random.randint(a=1, b=11)
        if randomItem <= 3:
            self.groundMatrix[i][j] = 'i_b'
            screen.blit(self.itemBombSurface, self.groundRect[i][j])
        elif randomItem >= 7:
            self.groundMatrix[i][j] = 'i_bs'
            screen.blit(self.itemBombSizeSurface, self.groundRect[i][j])
        else:
            self.groundRect[i][j].centerx = -1000
            # screen.blit(self.boxGoSurface, self.rectTemp)
            self.groundMatrix[i][j] = '-'

```

Tất cả các hình vẽ trong trò chơi, bao gồm cả các ô đất, quả bom và người chơi, được vẽ ra trên màn hình trong hàm drawObject.

- Trong hàm drawBox, mỗi ô đất hoặc ô box được vẽ ra trên màn hình bằng cách sử dụng phương thức blit của pygame.
- Hàm drawObject sau đó gọi draw của mỗi quả bom trong danh sách bom của cả hai người chơi. Nếu trạng thái của quả bom là nhỏ hơn hoặc bằng 1, nghĩa là nó đang chờ hoặc đang đếm ngược để nổ, thì quả bom được vẽ ra bằng hàm draw của quả bom. Ngược lại, nếu trạng thái là lớn hơn 1, nghĩa là quả bom đã nổ hoặc đang nổ, thì hàm drawBombBang được gọi để vẽ hiệu ứng nổ.
- Cuối cùng, hàm drawObject cũng vẽ ra người chơi bằng cách gọi hàm drawPlayer.

5.1.22 Chỉ số vị trí của người chơi trong ma trận

```

3 usages  LinhDancute
def getPositionPlayerInMatrix(self):
    rectPlayer = self.player.getRect()
    indexI = round(rectPlayer.centery/50)
    indexJ = round(rectPlayer.centerx/50)
    return indexI, indexJ

```

-Hàm getPositionPlayerInMatrix trả về vị trí của người chơi trong ma trận của trò chơi.

- Đầu tiên, nó lấy hình chữ nhật biểu diễn người chơi bằng cách gọi phương thức `getRect()` từ đối tượng `player`.
- Tiếp theo, nó tính chỉ số hàng `indexI` bằng cách chia tọa độ y của trung tâm của hình chữ nhật cho kích thước ô (50 pixel).
- Tương tự, chỉ số cột `indexJ` được tính bằng cách chia tọa độ x của trung tâm của hình chữ nhật cho kích thước ô.
- Cuối cùng, hàm trả về cặp chỉ số hàng và cột của người chơi trong ma trận.

5.1.23 Tính vị trí đối tượng trong ma trận

```
1 usage  LinhDancute
def getPositionObjectInMatrix(self, ObjectRect):
    indexI = round(ObjectRect.centery/50)
    indexJ = round(ObjectRect.centerx/50)
    return indexI, indexJ
```

-Hàm này nhận vào một hình chữ nhật đại diện cho một đối tượng và trả về chỉ số hàng và cột tương ứng của nó trong ma trận game. Đối tượng này có thể là một ô vuông trên màn hình hoặc một đối tượng khác như bom hoặc người chơi.

-Cách tính chỉ số hàng và cột:

- Đối với chỉ số hàng (`indexI`): Lấy tọa độ y của tâm của hình chữ nhật và chia cho kích thước của ô vuông.
- Đối với chỉ số cột (`indexJ`): Lấy tọa độ x của tâm của hình chữ nhật và chia cho kích thước của ô vuông.

-Sau đó, hàm trả về cặp chỉ số hàng và cột của đối tượng trong ma trận game.

5.1.24 Debugging và In thông tin ma trận

-In ra ma trận và các hình chữ nhật tương ứng trong ma trận.

```

LinhDancute
def printTest(self):
    print("GroundMatrix: \n")
    for i in self.groundMatrix:
        for j in i:
            print(j, end=",")
        print()
    print("GroundRect:")
    for i in self.groundRect:
        for j in i:
            print(j.centerX, end=",")
        print()

```

-Vòng lặp đầu tiên (dòng 2-7) in ra các giá trị của ma trận groundMatrix. Mỗi hàng của ma trận được in trên một dòng và các giá trị của mỗi hàng được phân tách bằng dấu phẩy.

-Vòng lặp thứ hai (dòng 9-14) in ra tọa độ trung tâm của các hình chữ nhật trong groundRect. Mỗi hàng của groundRect tương ứng với một hàng trong ma trận, và các tọa độ trung tâm của các hình chữ nhật trong hàng đó được in trên cùng một dòng và phân tách bằng dấu phẩy.

-Đây là một hàm debug, giúp theo dõi và kiểm tra dữ liệu trong quá trình phát triển ứng dụng.

5.1.25 *Animate Bom*


```
def animate(self):
    if self.surfaceIndex == 0:
        self.rect.centery -= 3
        self.surfaceIndex = 1
    else:
        self.rect.centery += 3
        self.surfaceIndex = 0
    self.surface = self.surfaceList[self.surfaceIndex]
```

-Tạo hiệu ứng chuyển động cho bom trước khi nổ.

-Phương thức animate thực hiện việc thay đổi vị trí và hình ảnh của bom.

- Nếu surfaceIndex là 0, tức là đang sử dụng hình ảnh đầu tiên trong danh sách surfaceList, bom sẽ di chuyển lên trên (self.rect.centery -= 3) và chuyển sang hình ảnh thứ hai trong danh sách (self.surfaceIndex = 1).
- Nếu surfaceIndex là 1, tức là đang sử dụng hình ảnh thứ hai trong danh sách surfaceList, bom sẽ di chuyển xuống dưới (self.rect.centery += 3) và chuyển sang hình ảnh đầu tiên trong danh sách (self.surfaceIndex = 0).

-Cuối cùng, cập nhật surface của bom thành hình ảnh mới tương ứng với surfaceIndex.

5.1.26 Vẽ các đối tượng lên màn hình

```
def draw(self, screen):
    screen.blit(self.surface, self.rect)
```

- Phương thức draw: Dùng để vẽ bom lên màn hình. Nó nhận đối số screen, là đối tượng màn hình pygame, và sử dụng phương thức blit để vẽ hình ảnh của bom lên vị trí được chỉ định bởi thuộc tính rect.

```
def drawBombBang(self, screen):
    for BombBang in self.bombBang:
        BombBang.drawBombBang(screen)
```

- Phương thức drawBombBang: Dùng để vẽ các hiệu ứng nổ bom lên màn hình. Nó nhận đối số screen và sử dụng một vòng lặp để vẽ mỗi hiệu ứng nổ bom trong danh sách bombBang. Mỗi hiệu ứng nổ bom được vẽ bằng cách gọi phương thức drawBombBang của đối tượng hiệu ứng nổ bom đó.

5.1.27 Các phương thức đặt bom

```
def placeABomb(self, playerRect):
    self.status = 1
    self.rect = playerRect
```

```
def setSurface_Rect(self, playerRect, indexSurface):
    self.rect = playerRect
    self.surface = self.surfaceList[indexSurface]
```

```
def setStatus(self, status):
    self.status = status
    if status == 1:
        self.timer=0
```

-Định nghĩa một số phương thức trong một lớp (hoặc một đối tượng) để thực hiện các hành động liên quan đến việc đặt bom trong trò chơi.

- `placeABomb`: Phương thức này được sử dụng để đặt bom vào vị trí của người chơi. Khi bom được đặt, trạng thái của bom sẽ chuyển sang 1, đồng thời cập nhật vị trí của bom bằng vị trí của người chơi.
- `setSurface_Rect`: Phương thức này cập nhật vị trí và hình ảnh của bom. Nó nhận vào vị trí của người chơi và chỉ số của hình ảnh bom trong danh sách hình ảnh của bom.
- `setStatus`: Phương thức này được sử dụng để đặt trạng thái của bom. Khi trạng thái được đặt thành 1 (đã đặt bom), đồng hồ đếm thời gian của bom được đặt lại.

5.1.28 *Kích hoạt vụ nổ của bomb*

```

def Bang(self):
    self.status = 2
    i=0
    for bang in self.bombBang:
        if i==0:
            bang.setRectCenterX_Y(self.getRect().centerx,self.getRect().centery-25*self.bombBangSize)
        elif i==1:
            bang.setRectCenterX_Y(self.getRect().centerx,self.getRect().centery+25*self.bombBangSize)
        elif i==2:
            bang.setRectCenterX_Y(self.getRect().centerx-25*self.bombBangSize,self.getRect().centery)
        elif i==3:
            bang.setRectCenterX_Y(self.getRect().centerx+25*self.bombBangSize,self.getRect().centery)
        i+=1

```

-Phương thức Bang trong đoạn mã này được sử dụng để kích hoạt bomb, khi nó đã đạt đến thời gian nổ. Cách nó hoạt động:

-Thay đổi trạng thái của bomb: Đặt trạng thái của bomb thành 2 để chỉ ra rằng nó đã nổ.

-Tạo và đặt vị trí cho các vụ nổ của bomb: Duyệt qua danh sách các vụ nổ của bomb và đặt vị trí cho mỗi vụ nổ dựa trên vị trí hiện tại của bomb và kích thước vụ nổ.

- Nếu $i == 0$, vụ nổ đặt ở phía trên bomb.
- Nếu $i == 1$, vụ nổ đặt ở phía dưới bomb.
- Nếu $i == 2$, vụ nổ đặt ở bên trái của bomb.
- Nếu $i == 3$, vụ nổ đặt ở bên phải của bomb.

-Kết quả là, khi bomb nổ, nó sẽ tạo ra các vụ nổ ở các hướng khác nhau xung quanh vị trí của nó.

5.1.29 Ấn bomb sau khi nổ

```
def hide(self):
    self.rect = self.surface.get_rect(center=(-1000, 200))
    self.bombBang = [BombBang(self.bombBang_Surface[0][self.bombBangSize-1]),
                     BombBang(self.bombBang_Surface[1][self.bombBangSize-1]),
                     BombBang(self.bombBang_Surface[2][self.bombBangSize-1]),
                     BombBang(self.bombBang_Surface[3][self.bombBangSize-1])]
    self.Bang()
    self.status = 0
```

Hàm này được sử dụng để ẩn bomb khỏi màn hình khi nó đã nổ.

- Đầu tiên, nó thiết lập lại vị trí của bomb ra xa màn hình bằng cách đặt centerx của rect của nó thành -1000. Điều này khiến cho bomb không còn được vẽ trên màn hình nữa.
- Tiếp theo, nó gọi hàm hide() trên các vụ nổ của bomb (nếu có), để cũng ẩn chúng khỏi màn hình.
- Cuối cùng, nó đặt trạng thái của bomb thành 0 để chỉ đánh dấu rằng bomb đã bị ẩn và không còn hoạt động nữa.

5.1.30 Tạo hiệu ứng nổ của bom

-Hàm setUpSurfaceBombBang tạo ra các hình ảnh để vẽ hiệu ứng nổ của bom. Mỗi hiệu ứng nổ sẽ có một hướng cụ thể (lên, xuống, trái, phải) và sẽ bao gồm một chuỗi các frame hoạt hình để tạo ra hiệu ứng chuyển động.

-Đầu tiên, vòng lặp for chạy từ 1 đến 4 để tạo ra hiệu ứng nổ cho mỗi hướng (lên, xuống, trái, phải). Trong mỗi vòng lặp, biến direction được gán giá trị tương ứng với hướng nổ: 'up', 'down', 'left', 'right'.

-Tiếp theo, vòng lặp for tiếp theo chạy từ 1 đến 10 để tạo ra 10 frame hoạt hình cho mỗi hướng.

-Mỗi frame hoạt hình được tạo bằng cách tải hình ảnh từ file có định dạng 'bomb-bang_directionX.png', trong đó X là chỉ số của frame.

-Kích thước của các hình ảnh được điều chỉnh tùy thuộc vào hướng nổ: các frame ở hướng dọc (lên, xuống) có chiều cao tăng dần, còn các frame ở hướng ngang (trái, phải) có chiều rộng tăng dần.

-Mỗi list các frame hoạt hình tương ứng với một hướng nổ sẽ được lưu vào list self.bombBang_Surface để sử dụng sau này khi vẽ hiệu ứng nổ của bom.

```

def setUpSurfaceBombBang(self):
    for i in range(1, 6):
        arrayTemp = []
        if i==1:
            direction = 'up'
        if i==2:
            direction = 'down'
        if i==3:
            direction = 'left'
        if i==4:
            direction = 'right'
        for j in range(1, 11):
            skin = pygame.image.load('./img/Bomb/bombbang_'+str(direction)+''+str(j)+'.png')
            if i==1 or i==2:
                arrayTemp.append(pygame.transform.scale(skin, size: (45,45*(j+1))))
            else:
                arrayTemp.append(pygame.transform.scale(skin, size: (45*(j+1),45)))
        skin = pygame.image.load('./img/Bomb/bombbang_'+str(direction)+''+str(1)+'.png')
        if i==1 or i==2:
            arrayTemp.append(pygame.transform.scale(skin, size: (45,1)))
        else:
            arrayTemp.append(pygame.transform.scale(skin, size: (1,45)))
        self.bombBang_Surface.append(arrayTemp)

```

5.1.31 Khởi tạo kết nối và luồng xử lý cho client

```

currentPlayer = 0
while True:
    conn, addr = s.accept()
    print("Connected to:", addr)
    start_new_thread(threaded_client, args: (conn, currentPlayer))
    currentPlayer += 1

```

-Thực hiện việc chấp nhận kết nối từ client và khởi tạo một luồng xử lý cho mỗi kết nối. Biến currentPlayer được sử dụng để gán một chỉ mục duy nhất cho mỗi client kết nối, để mỗi client có thể tương tác với trận đấu của họ một cách độc lập.

- s.accept(): Chờ đợi và chấp nhận một kết nối từ client. Trả về một cặp giá trị conn (một đối tượng socket mới để giao tiếp với client) và addr (địa chỉ của client).
- print("Connected to:", addr): In ra địa chỉ của client đã kết nối.

- `start_new_thread(threaded_client, (conn, currentPlayer))`: Khởi tạo một luồng mới để xử lý client vừa kết nối. Tham số đầu tiên là hàm `threaded_client`, tham số thứ hai là một tuple chứa `conn` (đối tượng socket) và `currentPlayer` (chỉ mục của người chơi hiện tại).
- `currentPlayer += 1`: Tăng chỉ mục của người chơi hiện tại để sử dụng cho kết nối tiếp theo.

5.1.32 Giao tiếp Client-Server cho Game nhiều người chơi

```
def threaded_client(conn, battleGround_Index):
    if battleGround_Index == 0 | battleGround_Index % 2 == 0:
        battleGround_Index = 0
    else:
        battleGround_Index = 1
    conn.send(pickle.dumps(battleGrounds[battleGround_Index]))
    reply = ""
    while True:
        try:
            data = pickle.loads(conn.recv(2048))
            battleGrounds[battleGround_Index] = data

            if not data:
                print("Disconnected")
                break
            else:
                if battleGrounds[0].get_groundMatrix() != battleGrounds[1].get_groundMatrix():
                    if battleGrounds[0].get_groundMatrix() != groundMatrix:
                        groundMatrix = battleGrounds[0].get_groundMatrix()
                        battleGrounds[1].set_groundMatrix(groundMatrix)
                    elif battleGrounds[1].get_groundMatrix() != groundMatrix:
                        groundMatrix = battleGrounds[1].get_groundMatrix()
                        battleGrounds[0].set_groundMatrix(groundMatrix)
                else:
                    groundMatrix = battleGrounds[0].get_groundMatrix()

                if battleGround_Index == 1:
                    reply = battleGrounds[0]
                else:
                    reply = battleGrounds[1]

            conn.sendall(pickle.dumps(reply))
        except:
            break
```

-Hàm này quản lý giao tiếp giữa server và client để cập nhật và đồng bộ hóa trạng thái của battleground trong một trò chơi nhiều người chơi

-Hàm này nhận hai đối số: conn, là socket kết nối, và battleGround_Idex, một chỉ mục chỉ định battleground nào được xử lý.

-Nó gửi trạng thái hiện tại của battleground được chỉ định bởi battleGround_Idex tới client. Hàm pickle.dumps() được sử dụng để serialize dữ liệu trước khi gửi đi.

-Nó bắt đầu một vòng lặp trong đó liên tục lắng nghe dữ liệu từ client.

-Khi nhận được dữ liệu từ client, nó deserialize dữ liệu đó bằng cách sử dụng pickle.loads() và cập nhật trạng thái battleground tương ứng.

- Nếu dữ liệu nhận được từ client là trống, nó chỉ ra rằng client đã ngắt kết nối, và vòng lặp sẽ kết thúc.
- Nếu client không ngắt kết nối, nó chuẩn bị một phản hồi dựa trên trạng thái hiện tại của battleground khác. Nếu battleGround_Idex là 1, nó sẽ trả lời với trạng thái của battleground 0, và ngược lại.

-Phản hồi được gửi trở lại cho client bằng cách sử dụng conn.sendall(pickle.dumps(reply)).

- Nếu xảy ra một ngoại lệ trong quá trình giao tiếp với client, nó sẽ thoát khỏi vòng lặp và đóng kết nối.

-Cuối cùng, nó in một thông báo chỉ ra rằng kết nối đã bị mất và đóng kết nối lại.

5.1.33 Khởi tạo Client Socket

```
def __init__(self):  
    self.client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
    self.server = SystemVariable.serverIP  
    self.port = SystemVariable.port  
    self.addr = (self.server, self.port)  
    self.battleGround = self.connect()
```

Hàm này khởi tạo một client socket để kết nối đến một server socket với địa chỉ và cổng nhất định.

- Tạo một đối tượng socket cho client với kiểu AF_INET (đại diện cho IPv4) và SOCK_STREAM (đại diện cho TCP socket).
- Lấy địa chỉ IP của server từ biến serverIP trong một module hoặc class khác.
- Lấy số cổng của server từ biến port trong cùng module hoặc class đó.
- Tạo tuple chứa địa chỉ IP và số cổng của server.

-Gọi phương thức connect() để thiết lập kết nối tới server và nhận dữ liệu về trạng thái của battleground.

5.1.34 Kết nối đến server và nhận dữ liệu từ server

```
def connect(self):
    try:
        self.client.connect(self.addr)
        return pickle.loads(self.client.recv(2048))
    except:
        pass
```

Phương thức connect trong lớp này được sử dụng để kết nối với server.

-Đầu tiên, nó cố gắng kết nối đến địa chỉ được chỉ định (self.addr) bằng cách sử dụng phương thức connect của socket.

- Nếu kết nối thành công, nó nhận dữ liệu từ server bằng cách gọi self.client.recv(2048). Dữ liệu nhận được được giải pickled (deserialize) bằng cách sử dụng pickle.loads.
- Nếu có bất kỳ lỗi nào xảy ra trong quá trình này (ví dụ: kết nối thất bại hoặc không thể nhận dữ liệu), nó chỉ đơn giản là bỏ qua và tiếp tục.

5.1.35 Phương thức gửi dữ liệu và nhận phản hồi

```
def send(self, data):
    try:
        self.client.send(pickle.dumps(data))
        return pickle.loads(self.client.recv(2048))
    except socket.error as e:
        print(e)
```

Phương thức send dùng để gửi dữ liệu lên máy chủ và nhận phản hồi từ máy chủ. Nó nhận một đối số là dữ liệu cần gửi, sau đó nó sẽ thực hiện các bước sau:

- Gửi dữ liệu đã được đóng gói bằng pickle lên máy chủ.

- Nhận phản hồi từ máy chủ, cũng đã được đóng gói bằng pickle.
- Trả về dữ liệu đã được giải nén từ máy chủ.
- Nếu có lỗi xảy ra trong quá trình gửi hoặc nhận, nó sẽ in ra lỗi đó.

CHƯƠNG 6. GIAO DIỆN GAME

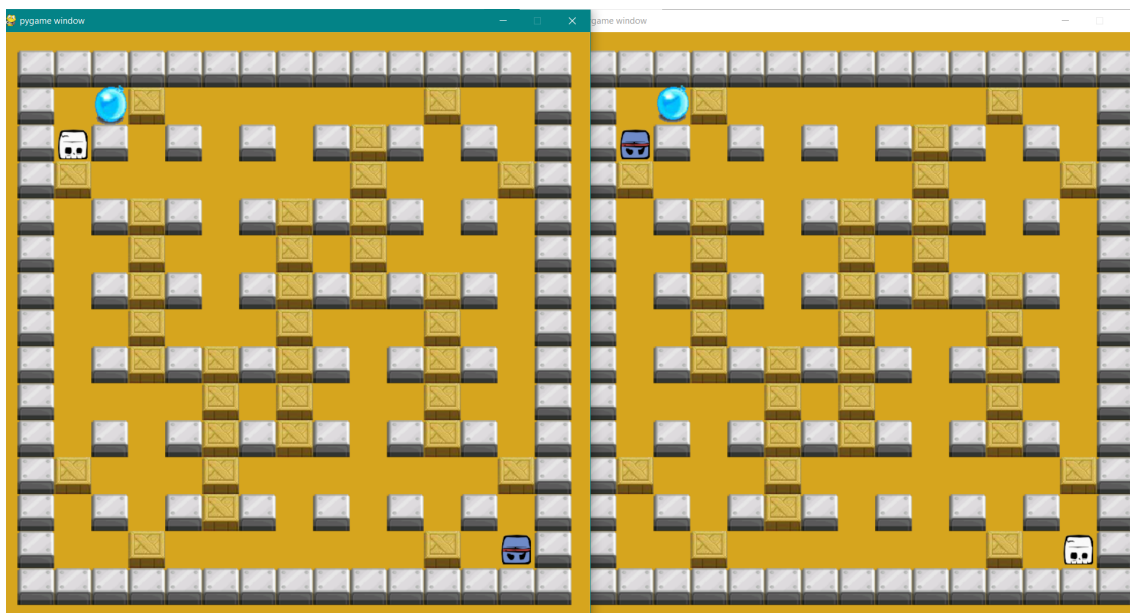
6.1 Giao diện Game Bomberman

6.1.1 *Giao diện khi bắt đầu vào Game*



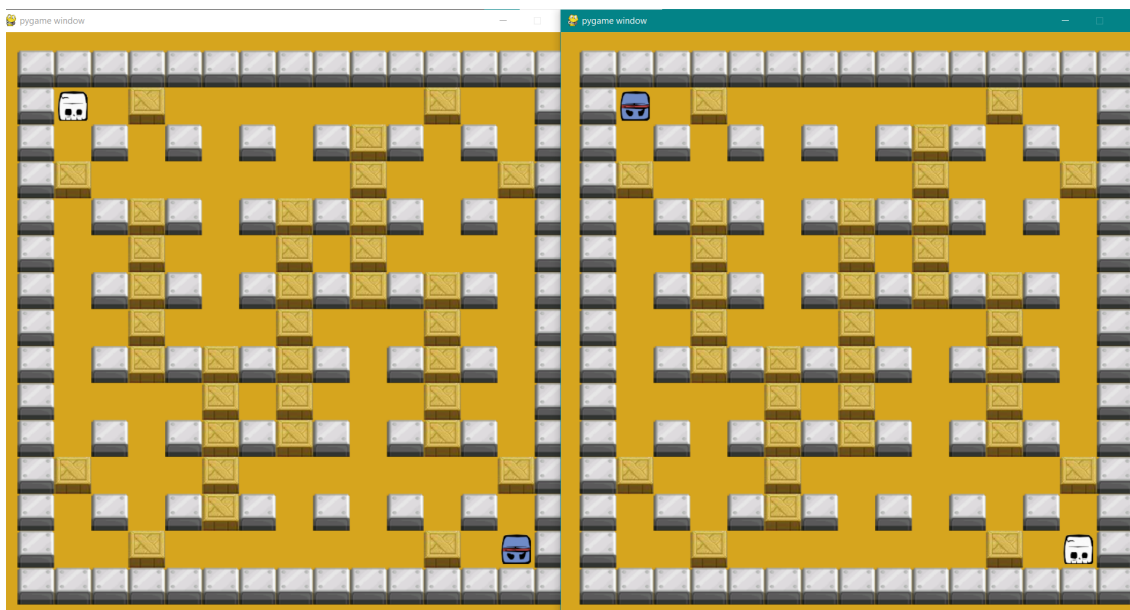
Đây là giao diện khi bắt đầu game.

6.1.2 *Giao diện đặt Boom*



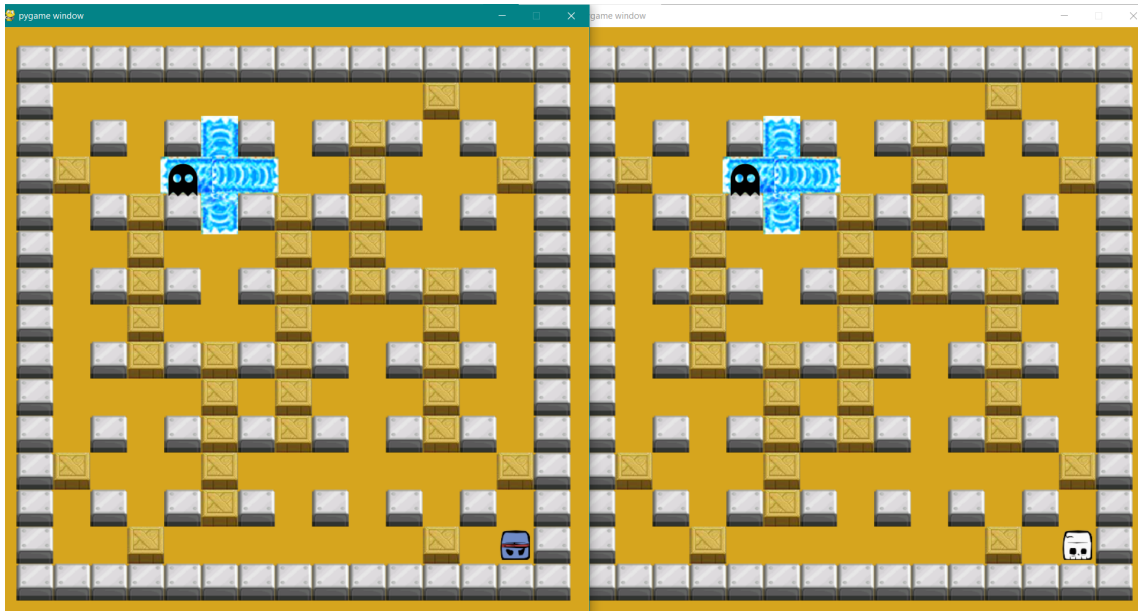
Đây là phần của trò chơi nơi người chơi có thể đặt bom tại vị trí hiện tại của họ trên bản đồ.

6.1.3 *Giao diện game 2 người chơi*



Đây là chế độ chơi của trò chơi mà hai người chơi có thể tham gia cùng một lúc và cạnh tranh với nhau.

6.1.4 *Giao diện khi Boom nổ và người chơi chết*



Khi bom nổ, nó có thể gây ra sự kiện người chơi chết nếu họ ở trong phạm vi tác động của nó.

CHƯƠNG 7. TÀI LIỆU THAM KHẢO

1. <https://www.pygame.org/wiki/Contribute>
2. <https://vi.wikipedia.org/wiki/Pygame>
3. <https://hostingviet.vn/socket-la-gi>