

TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN



PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

Xây dựng và Phát triển

Ứng dụng game Bomber trên PYTHON

GVHD: Từ Lãng Phiêu

SV: Phan Văn Hoàn - 3120410178

Dặng Thị Mỹ Ly - 3120410313

Trần Thị Hạ Long - 3120410289

Đoàn Thành Lợi - 3120410300

Mục lục

Lời nói đầu	2
1 Giới thiệu	3
1.1 Sơ lược game Boomer	3
1.2 Luật chơi	3
2 Môi trường cài đặt	4
2.1 Ngôn ngữ lập trình	4
2.2 Công cụ phát triển	4
2.3 Thư viện	4
2.4 Hệ điều hành	5
3 Thiết kế game Boom	5
3.1 Tổng quan	5
3.2 Hàm xử lý các chức năng	5
3.2.1 Quản lý vòng lặp và sự kiện	5
3.2.2 Giao tiếp với máy chủ	8
3.2.3 Hiển thị giao diện trò chơi	9
3.2.4 Xử lý trò chơi	10
3.3 Kết quả demo	13
4 Hướng dẫn cài đặt ở máy ảo	17
Kết luận	19



Lời nói đầu

Trong thời đại công nghệ ngày nay, việc phát triển phần mềm mã nguồn mở không chỉ là một xu hướng mà còn là một cách tiếp cận phổ biến trong ngành công nghiệp công nghệ thông tin. Việc sử dụng mã nguồn mở mang lại nhiều lợi ích, từ việc tiết kiệm chi phí phát triển đến việc tạo ra các sản phẩm chất lượng cao và dễ dàng tùy chỉnh.

Trong báo cáo này, chúng tôi xin giới thiệu về quá trình xây dựng và phát triển ứng dụng game Boom trên ngôn ngữ lập trình Python, một ví dụ cụ thể về việc áp dụng phương pháp phát triển phần mềm mã nguồn mở vào thực tế.

Báo cáo này sẽ bao gồm mô tả về luật chơi của game Boom, môi trường cài đặt cần thiết, quy trình thiết kế của ứng dụng, cùng với hướng dẫn cài đặt trên máy ảo. Ngoài ra, chúng tôi cũng sẽ chia sẻ kết quả demo của ứng dụng game để minh họa cho quá trình phát triển.

Cuối cùng, chúng tôi xin chân thành cảm ơn giáo viên hướng dẫn và các thành viên trong nhóm dự án đã đóng góp và hỗ trợ tích cực trong quá trình phát triển ứng dụng này.



1 Giới thiệu

1.1 Sơ lược game Boomer

BOOMBER GAME là một trò chơi đầy hấp dẫn và thách thức, được thiết kế đặc biệt cho hai người chơi. Trong trò chơi "Boomer Game", mỗi người chơi sẽ đảm nhận vai trò của một nhân vật phiêu lưu sẵn sàng đối mặt với những thách thức nguy hiểm.

Mỗi trường chơi của BOOMBER GAME là một mê cung lớn, nơi mà người chơi phải tìm cách vượt qua và đánh bại các kẻ thù để tiến đến cấp độ tiếp theo. Trò chơi này yêu cầu sự khéo léo, chiến thuật và phản xạ nhanh nhạy từ người chơi để có thể vượt qua mỗi thử thách.

Mỗi người chơi sẽ được trải nghiệm cảm giác hồi hộp và kịch tính khi phải đối mặt với các vụ nổ và bẫy trong mê cung. Sự cạnh tranh giữa hai người chơi càng làm cho trò chơi trở nên căng thẳng và hấp dẫn hơn.

BOOMING GAME EMPIRE không chỉ mang lại trải nghiệm giải trí mà còn thú vị và đầy thách thức cho cả hai người chơi, và là lựa chọn hoàn hảo cho một cuộc phiêu lưu đầy kịch tính và hấp dẫn.

1.2 Luật chơi

Trong BOOMBER GAME, các người chơi sẽ phải tuân theo các luật chơi sau:

- Mục tiêu:** Mỗi người chơi sẽ cố gắng tiêu diệt kẻ thù và tiến đến đích đầu tiên để chiến thắng trận đấu.
- Di chuyển:** Người chơi có thể di chuyển nhân vật của mình qua các ô trên bản đồ bằng các phím điều khiển được chỉ định.
- Đặt bom:** Người chơi có thể đặt bom bằng cách nhấn một phím đặt bom được chỉ định. Mỗi lần đặt bom, người chơi sẽ phải chờ một thời gian nhất định trước khi có thể đặt bom tiếp theo.
- Tránh bom:** Người chơi phải cẩn thận tránh bom của chính mình và của đối thủ. Nếu người chơi bị bom nổ, họ sẽ mất một mạng và cần phải bắt đầu lại từ vị trí xuất phát.
- Thu thập vật phẩm:** Trên bản đồ có thể xuất hiện các vật phẩm hỗ trợ như tăng sức mạnh hoặc tăng số lượng bom. Người chơi có thể thu thập các vật phẩm này để tăng khả năng chiến đấu.



- **Chiến thắng:** Người chơi nào đạt được mục tiêu đầu tiên và tiến đến đích đầu tiên sẽ chiến thắng trận đấu.

2 Môi trường cài đặt

2.1 Ngôn ngữ lập trình

Python: là một ngôn ngữ lập trình đa nền tảng, cung cấp một loạt các thư viện và framework hỗ trợ lập trình game, cho phép xử lý đồ họa, âm thanh, điều khiển gameplay và xây dựng các tính năng khác nhau của trò chơi.

2.2 Công cụ phát triển

PyCharm: là một nền tảng hybrid được JetBrains phát triển như một IDE cho Python. Nó thường được sử dụng để phát triển ứng dụng Python. PyCharm có thể tích hợp với các công cụ và framework game như Pygame và nhiều công nghệ khác. Điều này giúp giảm thời gian cài đặt và cấu hình môi trường phát triển cho game.

2.3 Thư viện

Pygame: là một thư viện của ngôn ngữ lập trình Python và là một tập hợp các mô-đun Python được thiết kế riêng để lập trình trò chơi. Có thể xây dựng các trò chơi trên nhiều nền tảng khác nhau như Windows, Mac, Linux.

Có thể quản lý tất cả các yếu tố trong quá trình phát triển trò chơi. Đó có thể là các chức năng như xuất đồ họa, xử lý sự kiện, hoạt ảnh, hiệu ứng âm thanh và phát lại nhạc.

Socket: được coi là một cổng giao tiếp giữa các ứng dụng trên mạng. Nó cho phép các thiết bị trao đổi thông tin với nhau thông qua một kết nối định sẵn.

Dự án này tập trung vào việc phát triển một ứng dụng game sử dụng stream socket. Việc sử dụng stream socket giúp xây dựng ứng dụng với khả năng truyền dữ liệu trực tiếp và liên tục, đồng thời đảm bảo tính ổn định và hiệu suất cao trong quá trình truyền thông. Stream Socket chỉ có thể hoạt động nếu như client và server đã kết nối cùng với nhau.

Tkinter: là một thư viện trong ngôn ngữ lập trình Python được sử dụng để tạo giao diện đồ họa người dùng (GUI). Tkinter có khả năng tạo các thành phần giao diện như cửa sổ, nút, ô văn



bản,... để tương tác với người dùng. Tkinter cung cấp cả các sự kiện và phương thức để xử lý tương tác người dùng và thay đổi trạng thái của ứng dụng.

2.4 Hệ điều hành

Các hệ điều hành mà ứng dụng game Bomber có thể chạy trên: Windows, macOS, Linux. Nó đều cung cấp một loạt các công cụ và tính năng hỗ trợ cho việc phát triển game, bao gồm cả IDE PyCharm, thư viện và framework phát triển game như Pygame, Tkinter,..., cũng như các công cụ debugging và profiling.

3 Thiết kế game Boom

3.1 Tổng quan

Pygame là một thư viện ngôn ngữ lập trình Python miễn phí và mã nguồn mở được sử dụng để tạo ra các ứng dụng đa phương tiện như trò chơi. Với Pygame, bạn có thể phát triển các trò chơi động, hấp dẫn với đồ họa và âm thanh sống động.

Trò chơi "Bomber Game" là một ví dụ điển hình về việc sử dụng Pygame để tạo ra trải nghiệm trò chơi độc đáo. Trong trò chơi này, người chơi đóng vai một nhân vật mạo hiểm, phải vượt qua các thách thức nguy hiểm trong một mê cung lớn. Pygame cung cấp các công cụ và tính năng mạnh mẽ để xây dựng các môi trường trò chơi phức tạp như mê cung và kiểm soát các nhân vật.

Ngoài ra, bằng cách sử dụng Socket và giao diện lập trình ứng dụng (API) socket, Pygame cũng cho phép trò chơi kết nối và giao tiếp với nhau qua mạng. Điều này mở ra cơ hội để tạo ra các trò chơi đa người chơi trực tuyến, nơi mà người chơi có thể tương tác và cạnh tranh với nhau từ xa.

3.2 Hàm xử lý các chức năng

3.2.1 Quản lý vòng lặp và sự kiện

```
1     def run(self):
2         response_thread = threading.Thread(target=self.receive_response_from_server)
3         response_thread.start()
4         while self.running:
```



```
5         for event in pygame.event.get():
6             if event.type == QUIT:
7                 self.running = False
8             elif event.type == KEYDOWN:
9                 if event.key == K_BACKSPACE:
10                     self.input_text = self.input_text[:-1]
11                 elif event.key == K_RETURN: # Set event when press enter
12                     if self.input_text.strip() == '':
13                         self.mess = self.font.render("Name is not empty", True,
14                                         (255, 0, 0))
15                         self.dialog_text = self.mess
16                         self.show_dialog = True
17                         pygame.time.set_timer(USEREVENT, 2000)
18                     else:
19                         client_socket.send(self.input_text.encode())
20                         self.name = self.input_text
21                 else:
22                     self.input_text += event.unicode
23             elif event.type == MOUSEMOTION: # Handle mouse motion event
24                 if self.start_button.collidepoint(event.pos): # If the mouse is
25                     over the button
26                     self.button_state = 'hover'
27                 else:
28                     self.button_state = 'normal'
29             elif event.type == MOUSEBUTTONDOWN:
30                 if self.start_button.collidepoint(event.pos):
31                     if self.input_text.strip() == '':
32                         self.mess = self.font.render("Name is not empty", True,
33                                         (255, 0, 0))
34                         self.dialog_text = self.mess
35                         self.show_dialog = True
36                         pygame.time.set_timer(USEREVENT, 2000)
37                     else:
38                         client_socket.send(self.input_text.encode())
39                         self.name = self.input_text
40
41             # Draw everything
42             self.view.blit(self.icon_surface, (0, 0)) # Draw the main_Image
43             self.view.blit(self.main_Image, (0, 0)) # Draw the main_Image
44             self.view.blit(self.button_start_image, self.start_button)
45
46             # Draw input box
47             pygame.draw.rect(self.view, (255, 255, 255), self.input_rect)
48             pygame.draw.rect(self.view, (0, 0, 0), self.input_rect, 2)
49             text_surface = self.font.render(self.input_text, True, (0, 0, 0))
50             self.view.blit(text_surface, (self.input_rect.x + 5, self.input_rect.y +
51                                         10))
52
53             if self.button_state == 'normal':
```



```
53         self.view.blit(self.button_start_image,
54                         self.start_button) # Draw the normal state button image
55                         on the screen
55     elif self.button_state == 'hover':
56         self.view.blit(self.button_start_hover,
57                         self.start_button) # Draw the hover state button image
57                         on the screen
58
59     # Draw dialog if needed
60     if self.show_dialog:
61         pygame.draw.rect(self.view, (255, 255, 255), self.dialog_rect) #
61             Draw white background for dialog
62         pygame.draw.rect(self.view, (0, 0, 0), self.dialog_rect, 2) # Draw
62             black border for dialog
63         self.view.blit(self.dialog_text, (self.dialog_rect.x + 25,
63                         self.dialog_rect.y + 75))
64
64     if self.server_full:
65         self.running = False
66         waiting_room = WaitingRoom(self.view, self.name, client_socket)
66         waiting_room.run()
67
68     pygame.display.update()
69     self.clock.tick(60)
70
71
72     pygame.quit()
```

```
1     def run(self):
2         response_thread = threading.Thread(target=self.receive_response_from_server)
3         response_thread.start()
4         while self.running:
5             for event in pygame.event.get():
6                 if event.type == pygame.QUIT:
7                     self.running = False
8                 if event.type == pygame.KEYDOWN:
9                     if event.key == K_a:
10                         self.press_a = True
11                     elif event.key == K_s:
12                         self.press_s = True
13                     elif event.key == K_d:
14                         self.press_d = True
15                     elif event.key == K_w:
16                         self.press_w = True
17                     elif event.key == K_SPACE:
18                         pass
19                 if event.type == pygame.KEYUP:
20                     if event.key == K_a:
21                         self.press_a = False
22                     elif event.key == K_s:
23                         self.press_s = False
```



```
24         elif event.key == K_d:
25             self.press_d = False
26         elif event.key == K_w:
27             self.press_w = False
28         elif event.key == K_SPACE:
29             if self.mBomber.status == Actor.ALIVE:
30                 self.innitBomb()
31                 self.mBomber.run_bomb = Actor.ALLOW_RUN
32
33         self.draw()
34         self.mBomber.draw_actor(self.screen)
35         if self.other is not None:
36             self.other.draw_actor(self.screen)
37
38         self.move()
39         self.set_run_bomber()
40         self.check_impact_item()
41         self.dead_line_all_bomb()
42         self.check_dead()
43
44         pygame.display.flip()
45         self.clock.tick(60)
46
47     pygame.quit()
```

3.2.2 Giao tiếp với máy chủ

Gửi tin nhắn hoặc yêu cầu từ người chơi đến máy chủ

```
1     def send_message(self, message):
2         text = self.name + ": " + message
3         data = {
4             "send_message": text
5         }
6         self.client_socket.send(json.dumps(data).encode())
```

Lắng nghe và xử lý các phản hồi từ máy chủ,

```
1     def receive_response_from_server(self):
2         while True:
3             try:
4                 response = self.client_socket.recv(1024).decode()
5                 data = json.loads(response)
6                 if "END_GAME" in data:
7                     with self.lock:
8                         self.dialog_text = self.font.render("You Win", True, (255, 0,
9                                         0))
10                        self.showDialog = True
```



```
11         if "send_data" in data:
12             with self.lock:
13                 self.other = Actor(data["x"], data["y"], data["type"],
14                                     data["orient"], data["speed"],
15                                     data["sizeBomb"],
16                                     data["quantity_bomb"], data["img"],
17                                     data["name"], data["heart"])
18             if "send_data_item" in data:
19                 with self.lock:
20                     for item in self.arrItem:
21                         if item.x == data["x"] and item.y == data["y"]:
22                             self.arrItem.remove(item)
23                             break
24             if "send_bomb" in data:
25                 with self.lock:
26                     bomb = Bomb(data["x"], data["y"], data["size"],
27                               data["timeline"])
28                     self.arrBomb.append(bomb)
29
30     except json.JSONDecodeError as e:
31         print("Error decoding JSON:", str(e))
```

Gửi tín hiệu cho máy chủ để bắt đầu trò chơi

```
1     def start_game(self):
2         self.client_socket.send("start_game".encode())
```

3.2.3 Hiển thị giao diện trò chơi

```
1     def draw(self):
2         self.screen.fill((255, 255, 255))
3
4         pygame.draw.rect(self.screen, (200, 200, 200), self.input_rect)
5         pygame.draw.rect(self.screen, (0, 0, 0), self.send_button_rect)
6
7         if len(self.client_sockets) == 2 and self.name == self.client_sockets[0]:
8             pygame.draw.rect(self.screen, (0, 0, 0), self.start_button_rect)
9
10        input_text_surface = self.font.render(self.input_text, True, (0, 0, 0))
11        self.screen.blit(input_text_surface, (self.input_rect.x + 5,
12                                         self.input_rect.y + 5))
13
14        send_button_text = self.font.render("Send", True, (255, 255, 255))
15        self.screen.blit(send_button_text, (self.send_button_rect.x + 10,
16                                         self.send_button_rect.y + 8))
17
18        start_button_text = self.font.render("Start", True, (255, 255, 255))
19        self.screen.blit(start_button_text, (self.start_button_rect.x + 10,
```



```
        self.start_button_rect.y + 8))

18
19     # Dialog
20     pygame.draw.rect(self.screen, (255, 255, 255), self.dialog_rect) # Draw
21         white background for dialog
22     pygame.draw.rect(self.screen, (0, 0, 0), self.dialog_rect, 2) # Draw black
23         border for dialog
24     self.screen.blit(self.dialog_text, (self.dialog_rect.x + 15,
25             self.dialog_rect.y + 75))

26
27     messages_rect = pygame.Rect(20, 20, self.WIDTH - 340, self.HEIGHT - 150)
28     pygame.draw.rect(self.screen, (220, 220, 220), messages_rect)

29
30     message_y = messages_rect.y + 5
31     for message in self.messages:
32         message_surface = self.font.render(message, True, (0, 0, 0))
33         self.screen.blit(message_surface, (messages_rect.x + 5, message_y))
34         message_y += 25

35
36     if len(self.client_sockets) > 0:
37         text_name = f"User 1: {self.client_sockets[0]} and User 2:
38             {self.client_sockets[1]}"
39         self.text = self.font.render(text_name, True, (255, 0, 0))
40         self.screen.blit(self.text, (self.dialog_rect.x + 15, self.dialog_rect.y
41             + 75))
42         text_name = "Me: " + self.name
43         self.text = self.font.render(text_name, True, (255, 0, 0))
44         self.screen.blit(self.text, (self.dialog_rect.x + 15, self.dialog_rect.y
45             + 105))

46
47     if len(self.messages) < 2:
48         self.messages.append(self.client_sockets[0] + " da tham gia")
49         self.messages.append(self.client_sockets[1] + " da tham gia")
```

3.2.4 Xử lý trò chơi

Hàm xử lý di chuyển:

```
1     def move(self):
2         if self.typeActor == 1:
3             img = "bebong"
4         else:
5             img = "khokho"
6
7         if not self.showDialog:
8             if self.press_a:
9                 self.mBomber.changeOrient(Actor.LEFT, img + "_left", self.screen)
10                self.mBomber.move(self.arrBomb, self.arrBox)
11            elif self.press_d:
12                self.mBomber.changeOrient(Actor.RIGHT, img + "_right", self.screen)
```



```
13         self.mBomber.move(self.arrBomb, self.arrBox)
14     elif self.press_s:
15         self.mBomber.changeOrient(Actor.DOWN, img + "_down", self.screen)
16         self.mBomber.move(self.arrBomb, self.arrBox)
17     elif self.press_w:
18         self.mBomber.changeOrient(Actor.UP, img + "_up", self.screen)
19         self.mBomber.move(self.arrBomb, self.arrBox)
20     self.send_data()
```

Hàm xử lý đặt bom :

```
1     def innitBomb(self):
2         if self.mBomber.status == Actor.DEAD:
3             return
4
5         x = self.mBomber.x + self.mBomber.width // 2
6         y = self.mBomber.y + self.mBomber.height // 2
7
8         for bomb in self.arrBomb:
9             if bomb.isImpact(x, y):
10                 return
11
12         if len(self.arrBomb) >= self.mBomber.quantity_bomb:
13             return
14
15         mBomb = Bomb(x, y, self.mBomber.sizeBomb, 2.5)
16         data = {
17             "send_bomb": "send_bomb",
18             "x": mBomb.x,
19             "y": mBomb.y,
20             "size": mBomb.size,
21             "timeline": 2.5
22         }
23
24         self.client_socket.send(json.dumps(data).encode())
25         self.arrBomb.append(mBomb)
```

Hàm kiểm tra va chạm:

```
1     def check_impact_item(self):
2         for index, item in enumerate(self.arrItem):
3             if item.isImpactItemVsBomber(self.mBomber):
4                 if not self.is_sending_data:
5                     self.is_sending_data = True
6
7                 data = {
8                     "send_data_item": "send_data_item",
9                     "position": index,
10                    "x": item.x,
```



```
11         "y": item.y
12     }
13
14     self.client_socket.send(json.dumps(data).encode())
15
16     self.is_sending_data = False
17     if item.type == Item.Item_Bomb:
18         self.mBomber.quantity_bomb += 1
19         self.arrItem.remove(item)
20         break
21     elif item.type == Item.Item_BombSize:
22         self.mBomber.sizeBomb += 1
23         self.arrItem.remove(item)
24         break
25     elif item.type == Item.Item_Shoe:
26         self.mBomber.speed += 1
27         self.arrItem.remove(item)
28         break
```

Hàm xử lý hoạt động của bom

```
1     def set_run_bomber(self):
2         if len(self.arrBomb) > 0:
3             if not self.arrBomb[-1].setRun(self.mBomber):
4                 self.mBomber.set_run_bomb(Actor.DISALLOW_RUN)
```

Hàm kết thúc trò chơi :

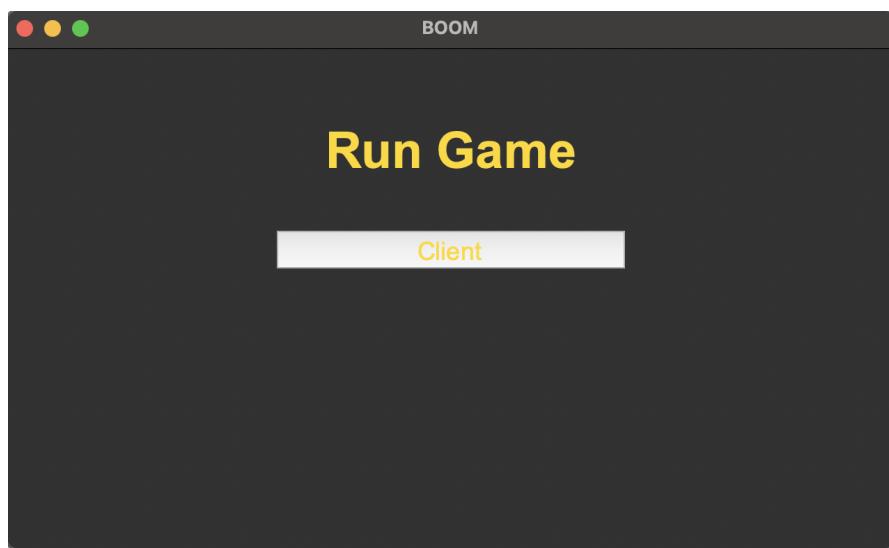
```
1     def dead_line_all_bomb(self):
2         for i in range(len(self.arrBomb)):
3             self.arrBomb[i].drawActor(self.screen)
4
5         for i in range(len(self.arrBombBang)):
6             self.arrBombBang[i].drawBongBang(self.screen)
7
8         for i in range(len(self.arrBomb)):
9             self.arrBomb[i].deadlineBomb()
10            if self.arrBomb[i].timeline == 0:
11                bom_bang = BombBang(self.arrBomb[i].x, self.arrBomb[i].y,
12                                    self.arrBomb[i].size, self.arrBox)
13                bom_bang.drawBongBang(self.screen)
14                self.arrBombBang.append(bom_bang)
15                self.arrBomb.pop(i)
16                break
17
18         for i in range(len(self.arrBombBang)):
19             for j in range(len(self.arrBomb)):
20                 if self.arrBombBang[i].isImpactBombBangvsBomb(self.arrBomb[j]):
21                     bom_bang = BombBang(self.arrBomb[j].x, self.arrBomb[j].y,
```



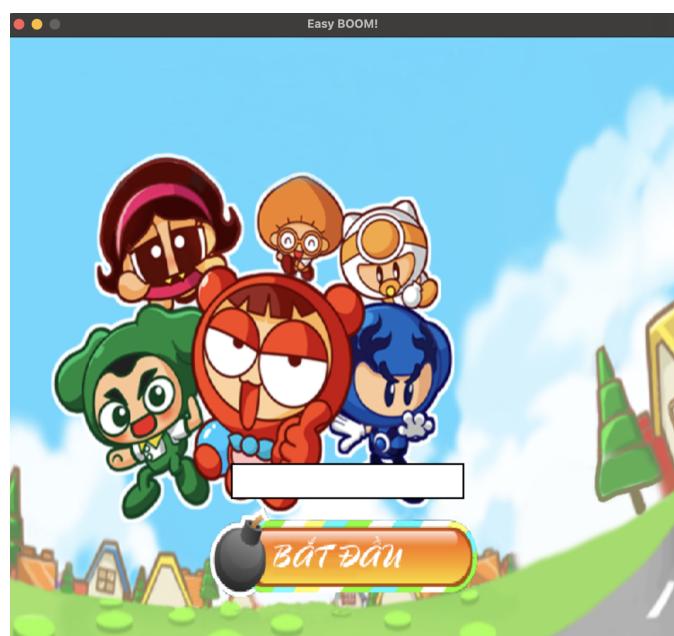
```
        self.arrBomb[j].size,
21            self.arrBox)
22        self.arrBombBang.append(bom_bang)
23        self.arrBomb.pop(j)
24        break
25
26    for k in range(len(self.arrBombBang)):
27        self.arrBombBang[k].deadlineBomb()
28
29    if self.arrBombBang[k].timeLine == 0:
30        for i in range(len(self.arrBombBang)):
31            if self.arrBombBang[i].isImpactBombBangVsActor(self.mBomber):
32                if self.mBomber.heart > 0:
33                    self.mBomber.heart -= 1
34
35            self.arrBombBang.pop(k)
36            break
37
38    for i in range(len(self.arrBombBang)):
39        for j in range(len(self.arrBox)):
40            if self.arrBombBang[i].isImpactBombBangvsBox(self.arrBox[j]):
41                self.arrBox.pop(j)
42                break
43
44    for i in range(len(self.arrBombBang)):
45        for j in range(len(self.arrItem)):
46            if self.arrBombBang[i].isImpactBombBangvsItem(self.arrItem[j]):
47                self.arrItem.pop(j)
48                break
```

```
1     def check_dead(self):
2         if self.mBomber.heart == 0:
3             if self.imgName == "bebong_down":
4                 self.mBomber.img = "bebong_dead"
5             else:
6                 self.mBomber.img = "khokho_dead"
7
8         self.mBomber.status = Actor.DEAD
9
10        self.client_socket.send("END_GAME".encode())
11        self.send_data()
12        self.dialog_text = self.font.render("You Lose", True, (255, 0, 0))
13        self.showDialog = True
```

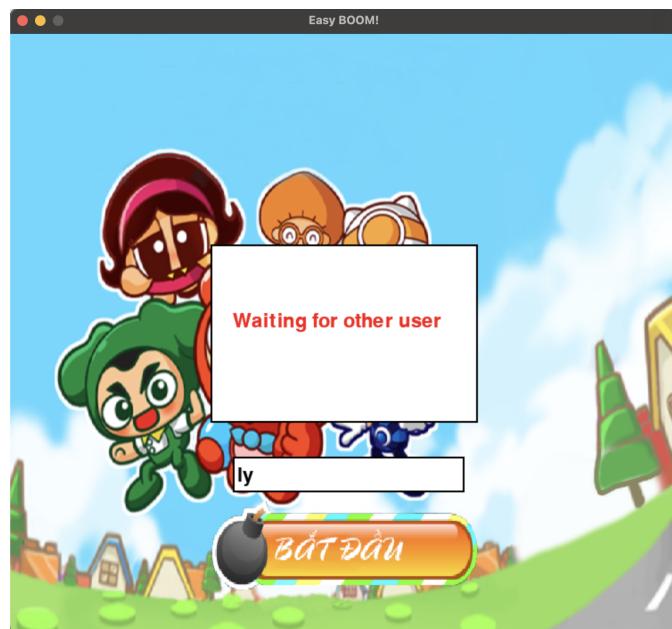
3.3 Kết quả demo



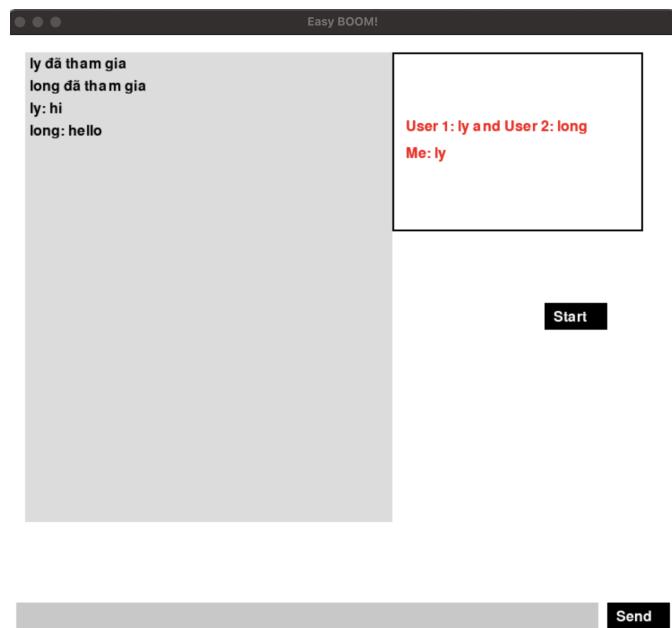
Hình 1. Giao diện tham gia vào trò chơi



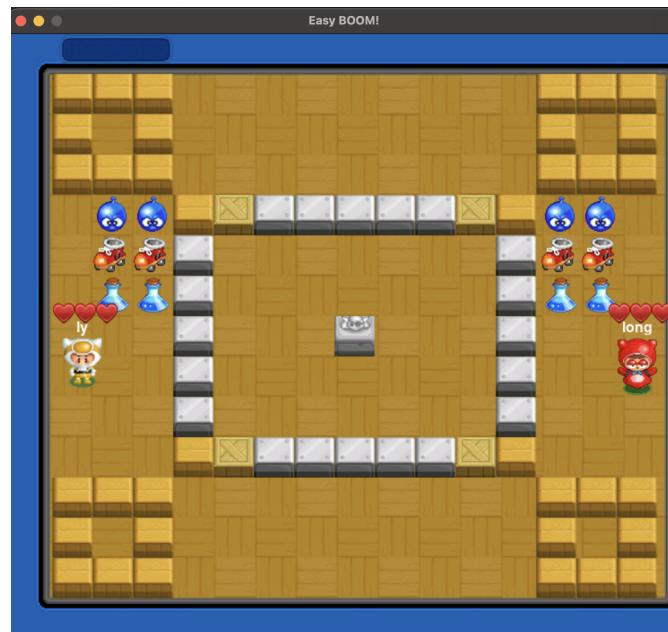
Hình 2. Giao diện người dùng nhập tên trước khi bắt đầu trò chơi



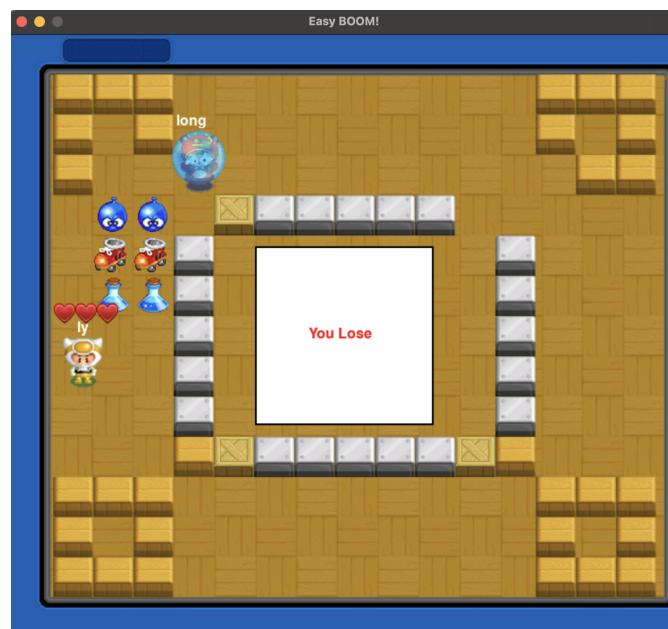
Hình 3. Giao diện chờ người chơi khác tham gia



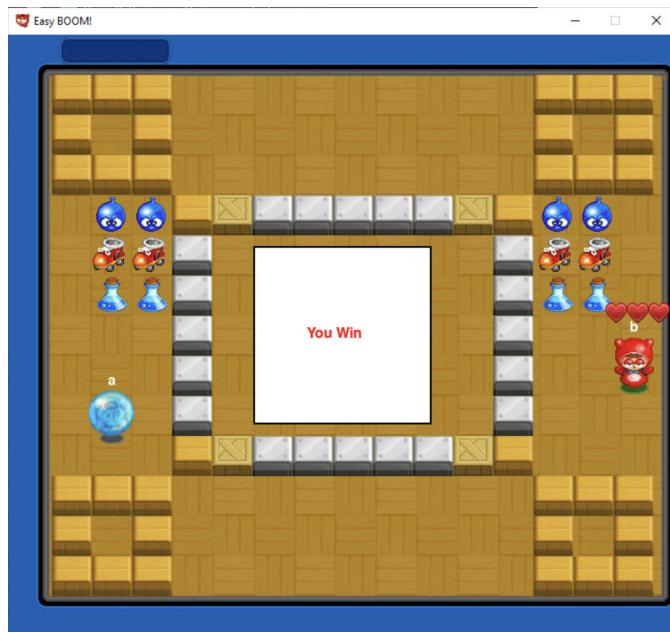
Hình 4. Giao diện chat



Hình 5. Giao diện trò chơi



Hình 6. Giao diện khi người chơi thua



Hình 7. Giao diện khi người chơi thắng

4 Hướng dẫn cài đặt ở máy ảo

Dưới đây là các bước chi tiết để cài đặt môi trường phát triển Python và PyCharm trên một máy ảo Linux.

Bước 1: Cài đặt Python 3 và pip 3

Terminal:

```
sudo apt update
sudo apt install python3
sudo apt install python3-pip
```

Bước 2: Cài đặt thư viện để chạy game

Terminal:

```
sudo apt-get install python3-pygame
sudo apt-get install python3-tk
```

Bước 3: Cài đặt PyCharm và thêm vào ứng dụng của máy ảo

3.1. Cài đặt:

Truy cập vào trang web chính thức của PyCharm (<https://www.jetbrains.com/pycharm/>) và tải xuống phiên bản Community hoặc Professional của PyCharm dưới dạng tệp cài đặt .tar.gz.



Mở terminal và di chuyển đến thư mục chứa tệp cài đặt PyCharm. Giải nén tệp cài đặt bằng lệnh sau:

```
tar -xzf pycharm-*.tar.gz
```

Lưu ý: Thay "pycharm-*.tar.gz" bằng tên thực tế của tệp bạn đã tải xuống.

3.2. Thêm vào phần ứng dụng:

Tạo một tập tin mới với phần mở rộng .desktop bằng trình soạn thảo văn bản nano hoặc gedit.

Trong tập tin này, thêm các dòng sau:

[Desktop Entry]

Version=1.0

Type=Application

Name=PyCharm

Exec=/đường/dẫn/đến/pycharm-*/bin/pycharm.sh

Icon=/đường/dẫn/đến/pycharm-*/bin/pycharm.svg

Comment=Integrated Development Environment

Categories=Development;IDE;

Terminal=false

Lưu tập tin và di chuyển nó vào thư mục /usr/share/applications để nó có thể được truy cập từ menu ứng dụng của hệ thống.

Bước 4: Tải project từ github

Cài đặt git trong linux:

```
sudo apt update  
sudo apt install git
```

Clone dự án về máy:

```
git clone <URL_dự_án_GitHub>
```



Kết luận

Trong quá trình phát triển trò chơi Boom sử dụng Pygame, chúng tôi đã trải qua một chặng đường học tập và khám phá đầy ý nghĩa về khả năng của thư viện này trong việc tạo ra trải nghiệm trò chơi độc đáo và hấp dẫn.

Pygame không chỉ đơn thuần là một công cụ để phát triển trò chơi đa phương tiện, mà còn là một nền tảng linh hoạt cho sự sáng tạo của chúng tôi. Khả năng kiểm soát đồ họa, âm thanh và tương tác người dùng của Pygame đã giúp chúng tôi xây dựng một môi trường trò chơi đa dạng và phong phú.

Kết hợp Pygame với Socket API cũng mở ra một cánh cửa mới cho trò chơi trực tuyến. Điều này không chỉ giúp chúng tôi kết nối và giao tiếp với các người chơi khác qua mạng, mà còn tạo ra một môi trường tương tác độc đáo và thú vị.

Tóm lại, việc phát triển trò chơi Boom đã mang lại cho chúng tôi nhiều kiến thức và kinh nghiệm quý báu không chỉ về lập trình và thiết kế trò chơi mà còn về sự sáng tạo và khám phá trong lĩnh vực công nghệ trò chơi.