

TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN



PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

Phát triển ứng dụng game, dùng cho nhiều người chơi

Ping Pong Game trên PYTHON

GVHD: Từ Lăng Phiêu
SV: Võ Thanh Trọng - 3121410531
Nguyễn Thanh Phong - 3121410383
Nguyễn Đăng Khôi - 3121410279

TP. HỒ CHÍ MINH, THÁNG 5/2024

Mục lục

1	Phần giới thiệu	2
1.1	Giới thiệu về Pygame	2
1.1.1	Pygame là gì ?	2
1.1.2	Lịch sử hình hành	2
1.1.3	Một số tính năng của Pygame	2
1.2	Giới thiệu về Socket	3
1.2.1	Socket là gì ?	3
1.2.2	Lịch sử hình hành	3
1.2.3	Một số ứng dụng của Socket	4
1.2.4	Ứng dụng Socket vào trong Pygame	4
2	Giới thiệu về Game Ping Pong	5
2.1	Lý do	5
2.2	Mục tiêu	5
3	Tiến hành xây dựng game	6
3.1	Cài đặt Pygame	6
3.2	Thiết kế các thành phần của game	6
3.2.1	Class Player	6
3.2.2	Class Puck	7
3.2.3	Class Game	7
3.3	Thiết kế Network	10
3.4	Thiết kế Server Socket	11
3.5	Xây dựng Client	14
3.5.1	Thiết kế giao diện người dùng	14
3.5.2	Thiết kế chức năng chơi game	16
3.6	Kết quả thực thi chương trình	17

1 Phần giới thiệu

1.1 Giới thiệu về Pygame

1.1.1 Pygame là gì ?

Pygame là một thư viện mã nguồn mở cho Python, được sử dụng để phát triển trò chơi và ứng dụng đa phương tiện. Nó cung cấp các công cụ cần thiết để xây dựng các trò chơi 2D, bao gồm quản lý cửa sổ, đồ họa, âm thanh, và xử lý sự kiện.

Pygame được xây dựng trên thư viện SDL (Simple DirectMedia Layer), cho phép truy cập đến các chức năng đồ họa và âm thanh trên nhiều nền tảng, bao gồm cả Windows, macOS và Linux. Nó cung cấp một giao diện dễ sử dụng để tạo ra các trò chơi đơn giản hoặc phức tạp, và là một công cụ phổ biến trong cộng đồng phát triển trò chơi Python.



Hình 1: Logo của Pygame

1.1.2 Lịch sử hình thành

Pygame bắt đầu phát triển vào năm 1999 bởi Pete Shinnars như một dự án cá nhân. Ông đã tạo ra Pygame bằng cách sử dụng thư viện Simple DirectMedia Layer (SDL) để tạo ra một giao diện lập trình ứng dụng (API) dễ sử dụng cho việc phát triển trò chơi bằng Python.

Ban đầu, Pygame chỉ là một dự án nhỏ và không có sự phát triển lớn cho đến khi Pete Shinnars công bố mã nguồn Pygame và phát hành phiên bản 1.0 vào năm 2000. Sau khi công bố, Pygame nhận được sự quan tâm từ cộng đồng phát triển Python và trở thành một thư viện phổ biến cho việc phát triển trò chơi 2D và ứng dụng đa phương tiện.

Với sự phát triển và đóng góp từ cộng đồng, Pygame đã trở thành một công cụ mạnh mẽ cho việc phát triển trò chơi và ứng dụng đa phương tiện trên Python. Các phiên bản mới của Pygame đã được phát hành với nhiều tính năng và cải tiến, bao gồm hỗ trợ âm thanh, đồ họa, xử lý sự kiện, và các tính năng khác giúp người phát triển tạo ra những trò chơi và ứng dụng đa phương tiện phong phú.

1.1.3 Một số tính năng của Pygame

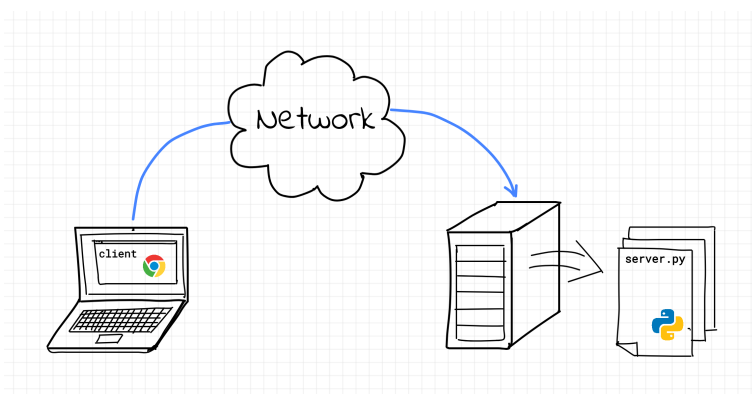
- **Đồ họa 2D:** Pygame cung cấp các công cụ để vẽ đồ họa 2D, bao gồm các hình ảnh, văn bản, và hình vẽ hình học.
- **Âm thanh:** Pygame hỗ trợ phát âm thanh và nhạc nền trong trò chơi và ứng dụng đa phương tiện.
- **Xử lý sự kiện:** Pygame cho phép xử lý sự kiện như nhấn phím, di chuyển chuột, và nhấn nút trong trò chơi.

- **Quản lý cửa sổ:** Pygame cung cấp các công cụ để tạo và quản lý cửa sổ trò chơi, bao gồm cài đặt kích thước, tiêu đề, và chế độ toàn màn hình.
- **Hỗ trợ nền tảng:** Pygame hỗ trợ nhiều nền tảng, bao gồm Windows, macOS, và Linux.

1.2 Giới thiệu về Socket

1.2.1 Socket là gì ?

Socket là một giao diện lập trình ứng dụng (API) cho việc truyền thông giữa các ứng dụng trên mạng. Nó cung cấp các công cụ cần thiết để tạo ra các kết nối mạng, truyền dữ liệu giữa các máy tính, và xử lý các giao thức mạng như TCP và UDP.



Hình 2: Mô hình Socket

Trong ngữ cảnh của lập trình mạng, một socket được coi là một đầu cuối của một kênh truyền thông giữa hai thiết bị. Một socket có thể hoạt động dựa trên nhiều giao thức truyền thông như TCP (Transmission Control Protocol) hoặc UDP (User Datagram Protocol).

1.2.2 Lịch sử hình thành

Socket là một khái niệm và công nghệ quan trọng trong lĩnh vực lập trình mạng. Được phát triển từ những năm 1970, socket đã trở thành một phần không thể thiếu trong việc xây dựng ứng dụng mạng hiệu quả.

Ban đầu, socket được phát triển bởi Ủy ban Nghiên cứu Mạng xã hội (ARPA) của Bộ Quốc phòng Hoa Kỳ (Hiện nay là Cục Nghiên cứu Dự án Tiên tiến - DARPA) như một phần của Dự án ARPANET, một mạng máy tính tiền thân của Internet. Socket được sử dụng để thiết lập kết nối và truyền thông giữa các máy tính trên mạng. Ban đầu, socket được viết bằng ngôn ngữ lập trình C.

Sau đó, socket được tiêu chuẩn hóa trong các tài liệu RFC (Request for Comments) của IETF (Internet Engineering Task Force), đặc biệt là RFC 793 "Transmission Control Protocol" và RFC 768 "User Datagram Protocol". Các RFC này định nghĩa giao thức TCP và UDP, và cung cấp cách sử dụng socket để truyền dữ liệu qua mạng bằng hai giao thức này.



1.2.3 Một số ứng dụng của Socket

- **Truyền dữ liệu:** Socket được sử dụng để truyền dữ liệu giữa các máy tính trên mạng, bao gồm truyền tệp, tin nhắn, và dữ liệu khác.
- **Thiết lập kết nối:** Socket được sử dụng để thiết lập kết nối giữa các máy tính trên mạng, bao gồm kết nối TCP và UDP.
- **Xử lý giao thức:** Socket được sử dụng để xử lý các giao thức truyền thông như TCP và UDP, bao gồm việc gửi và nhận dữ liệu, xác định địa chỉ IP và cổng, và xử lý lỗi truyền thông.
- **Phát triển ứng dụng mạng:** Socket được sử dụng để phát triển ứng dụng mạng như trò chơi trực tuyến, ứng dụng trò chuyện, và ứng dụng chia sẻ tệp.
- **Kiểm tra mạng:** Socket được sử dụng để kiểm tra mạng bằng cách gửi và nhận dữ liệu giữa các máy tính trên mạng.
- **Bảo mật mạng:** Socket được sử dụng để bảo mật mạng bằng cách mã hóa dữ liệu trước khi truyền và giải mã dữ liệu sau khi nhận.

1.2.4 Ứng dụng Socket vào trong Pygame

Ứng dụng của socket trong Pygame là tạo ra các trò chơi đa người chơi thông qua mạng. Bằng cách sử dụng socket, ta có thể thiết lập kết nối mạng giữa các máy tính và truyền dữ liệu giữa chúng để tạo trải nghiệm chơi game đa người chơi. Dưới đây là một số ứng dụng của socket trong Pygame:

- **Trò chơi đa người chơi:** Sử dụng socket để tạo ra các trò chơi đa người chơi trên mạng, bao gồm trò chơi đối kháng, trò chơi hợp tác, và trò chơi trực tuyến.
- **Trò chơi trực tuyến:** Sử dụng socket để tạo ra các trò chơi trực tuyến giữa các máy tính trên mạng, cho phép người chơi chơi cùng nhau từ xa.
- **Trò chơi địa phương:** Sử dụng socket để tạo ra các trò chơi địa phương giữa các máy tính trên cùng một mạng LAN, cho phép người chơi chơi cùng nhau trên cùng một mạng.
- **Trò chơi chia sẻ màn hình:** Sử dụng socket để chia sẻ màn hình giữa các máy tính trên mạng, cho phép người chơi xem và điều khiển trò chơi từ xa.

2 Giới thiệu về Game Ping Pong

2.1 Lý do

- **Đơn giản và dễ hiểu:** Ping pong là một trò chơi đơn giản với nguyên tắc chơi dễ hiểu. Nó chỉ yêu cầu hai người chơi sử dụng thanh điều khiển để đẩy một quả bóng qua lại trên một bàn. Điều này giúp dễ dàng hiểu và triển khai các quy tắc và tính năng của trò chơi.
- **Trực quan và thú vị:** Game ping pong mang đến một trải nghiệm trực quan và thú vị. Người chơi có thể tận hưởng sự thách thức và cảm giác phấn khích khi cố gắng đánh bại đối thủ bằng cách sử dụng kỹ năng và phản xạ nhanh nhạy. Việc theo dõi quả bóng di chuyển nhanh và phản ứng kịp thời để đẩy nó trở lại tạo ra một trạng thái tập trung cao đồng thời mang lại niềm vui và hứng thú.
- **Dễ dàng tùy chỉnh và mở rộng:** Game ping pong có thể dễ dàng tùy chỉnh và mở rộng để thêm các tính năng và chế độ chơi mới. Bằng cách sử dụng Pygame và socket, ta có thể tạo ra các phiên bản mới của trò chơi với các tính năng độc đáo và chế độ chơi đa dạng, từ trò chơi đơn giản giữa hai người chơi đến trò chơi trực tuyến giữa nhiều người chơi.
- **Thích hợp cho mọi lứa tuổi:** Game ping pong là một trò chơi phổ biến và thích hợp cho mọi lứa tuổi. Người chơi có thể tham gia trò chơi mà không cần kiến thức hoặc kỹ năng đặc biệt, chỉ cần sự tập trung và phản xạ nhanh nhạy. Điều này giúp trò chơi trở thành một lựa chọn tuyệt vời cho cả trẻ em và người lớn.
- **Thúc đẩy sự cạnh tranh và hợp tác:** Game ping pong thúc đẩy sự cạnh tranh và hợp tác giữa người chơi. Người chơi cần cạnh tranh với đối thủ để giành chiến thắng, nhưng cũng cần hợp tác để duy trì trò chơi và tạo ra những pha đánh bóng đẹp mắt. Điều này giúp tạo ra một môi trường chơi game tích cực và hấp dẫn.
- **Giúp cải thiện kỹ năng và phản xạ:** Game ping pong giúp cải thiện kỹ năng và phản xạ của người chơi. Việc theo dõi và phản ứng nhanh chóng để đẩy bóng trở lại yêu cầu sự tập trung và kỹ năng điều khiển chính xác. Điều này giúp người chơi phát triển kỹ năng và phản xạ của mình trong khi tham gia trò chơi.

2.2 Mục tiêu

- **Phát triển trò chơi ping pong trên Python:** Mục tiêu chính của dự án là phát triển trò chơi ping pong trên Python bằng cách sử dụng thư viện Pygame. Trò chơi sẽ bao gồm các tính năng cơ bản như di chuyển thanh điều khiển, đẩy bóng qua lại, và tính điểm.
- **Hỗ trợ chế độ chơi đơn và đa người:** Trò chơi sẽ hỗ trợ cả chế độ chơi đơn và chế độ chơi đa người. Chế độ chơi đơn sẽ cho phép người chơi chơi với máy tính, trong khi chế độ chơi đa người sẽ cho phép hai người chơi chơi với nhau trên cùng một máy tính hoặc qua mạng.
- **Tích hợp socket để tạo trò chơi đa người:** Trò chơi sẽ sử dụng socket để tạo kết nối mạng giữa hai máy tính và truyền dữ liệu giữa chúng để tạo trải nghiệm chơi game đa người. Người chơi sẽ có thể chơi trò chơi với nhau từ xa thông qua mạng.
- **Tùy chỉnh và mở rộng trò chơi:** Trò chơi sẽ được thiết kế để dễ dàng tùy chỉnh và mở rộng. Bằng cách sử dụng Pygame, ta có thể thêm các tính năng mới, chế độ chơi, và cấu hình trò chơi để tạo ra các phiên bản



- **Thiết kế giao diện người dùng thân thiện:** Trò chơi sẽ có giao diện người dùng thân thiện và dễ sử dụng. Người chơi sẽ có thể dễ dàng điều khiển thanh điều khiển, đẩy bóng qua lại, và theo dõi điểm số trong trò chơi.
- **Tạo trải nghiệm chơi game thú vị:** Mục tiêu cuối cùng của dự án là tạo ra một trải nghiệm chơi game thú vị và hấp dẫn cho người chơi. Trò chơi sẽ mang lại niềm vui, hứng thú, và thách thức cho người chơi mỗi khi tham gia.

3 Tiến hành xây dựng game

3.1 Cài đặt Pygame

Để bắt đầu phát triển trò chơi ping pong trên Python, ta cần cài đặt thư viện Pygame trước. Pygame là một thư viện mã nguồn mở cho Python, được sử dụng để phát triển trò chơi và ứng dụng đa phương tiện. Nó cung cấp các công cụ cần thiết để xây dựng các trò chơi 2D, bao gồm quản lý cửa sổ, đồ họa, âm thanh, và xử lý sự kiện.

Để cài đặt Pygame, ta có thể sử dụng trình quản lý gói pip của Python. Mở cửa sổ dòng lệnh và chạy lệnh sau để cài đặt Pygame:

```
1 pip install pygame
```

Sau khi cài đặt xong, ta có thể bắt đầu phát triển trò chơi ping pong trên Python bằng cách sử dụng thư viện Pygame.

3.2 Thiết kế các thành phần của game

3.2.1 Class Player

```
1 class Player:
2     def __init__(self, x, y, color):
3         self.pos = [x, y]
4         self.color = color
5         self.radius = 34
6         self.velocity = PLAYER_V
7         self.is_collided = False
8
9     def draw(self, win):
10         pygame.draw.circle(win, self.color, (self.pos[0], self.pos[1]), self.radius)
```

- **pos:** Vị trí của người chơi trên màn hình.
- **color:** Màu sắc của người chơi.
- **radius:** Bán kính của người chơi.
- **velocity:** Vận tốc di chuyển của người chơi.
- **is_collided:** Biến kiểm tra xem người chơi đã va chạm với bóng hay chưa.
- **draw():** Phương thức vẽ người chơi lên màn hình.



3.2.2 Class Puck

```
1 class Puck:
2     def __init__(self, x, y, color):
3         self.pos = [x, y]
4         self.color = color
5         self.radius = 13
6         self.velocity = PUCK_V
7         self.vx, self.vy = random.choice([(-1, -1), (-1, 1), (1, -1), (1, 1)])
8
9     def draw(self, win):
10        pygame.draw.circle(win, self.color, (self.pos[0], self.pos[1]), self.radius)
```

- **pos**: Vị trí của bóng trên màn hình.
- **color**: Màu sắc của bóng.
- **radius**: Bán kính của bóng.
- **velocity**: Vận tốc di chuyển của bóng.
- **vx, vy**: Hướng di chuyển của bóng.
- **draw()**: Phương thức vẽ bóng lên màn hình.

3.2.3 Class Game

```
1 class Game:
2     def __init__(self, id):
3         self.id = id
4         self.players = [Player(120, 120, YELLOW), Player(480, 120, GREEN),
5                          Player(120, 480, BLUE), Player(480, 480, PINK)]
6         self.ready = False
7         self.puck = Puck(300, 300, (221, 221, 221))
8         self.winner = None
9
10
11
12     def end_game(self, winner):
13         self.winner = winner
14
15     def play(self, p, moves):
16         moves = moves.split('&&')
17         if 'left' in moves:
18             if 70 < self.players[p].pos[0] - self.players[p].velocity -
19                 self.players[p].radius:
20                 self.players[p].pos[0] -= self.players[p].velocity
21         if 'right' in moves:
22             if self.players[p].pos[0] + self.players[p].velocity +
23                 self.players[p].radius < 530:
24                 self.players[p].pos[0] += self.players[p].velocity
25         if 'up' in moves:
```

```
24         if 70 < self.players[p].pos[1] - self.players[p].velocity -
25             self.players[p].radius:
26             self.players[p].pos[1] -= self.players[p].velocity
27         if 'down' in moves:
28             if self.players[p].pos[1] + self.players[p].velocity +
29                 self.players[p].radius < 530:
30                 self.players[p].pos[1] += self.players[p].velocity
31
32     def update(self):
33         self.update_puck()
34
35     def update_puck(self):
36         for pl in self.players:
37             x = pl.pos[0] - self.puck.pos[0]
38             y = pl.pos[1] - self.puck.pos[1]
39             dis = math.sqrt(x * x + y * y)
40             if dis <= pl.radius + self.puck.radius:
41                 if not pl.is_collided:
42                     if self.puck.pos[0] > pl.pos[0]:
43                         self.puck.vx = 1
44                     else:
45                         self.puck.vx = -1
46                     if self.puck.pos[1] > pl.pos[1]:
47                         self.puck.vy = 1
48                     else:
49                         self.puck.vy = -1
50                 pl.is_collided = True
51             else:
52                 pl.is_collided = False
53         if (self.puck.pos[0] - self.puck.radius <= 70 and self.puck.pos[1] <= 300) or \
54             (self.puck.pos[1] - self.puck.radius <= 70 and self.puck.pos[0] <= 300):
55             self.end_game('Yellow')
56         if (self.puck.pos[0] + self.puck.radius >= 530 and self.puck.pos[1] <= 300) or \
57             (self.puck.pos[1] - self.puck.radius <= 70 and self.puck.pos[0] > 300):
58             self.end_game('Green')
59         if (self.puck.pos[0] - self.puck.radius <= 70 and self.puck.pos[1] > 300) or \
60             (self.puck.pos[1] + self.puck.radius >= 530 and self.puck.pos[0] <= 300):
61             self.end_game('Blue')
62         if (self.puck.pos[0] + self.puck.radius >= 530 and self.puck.pos[1] > 300) or \
63             (self.puck.pos[1] + self.puck.radius >= 530 and self.puck.pos[0] > 300):
64             self.end_game('Pink')
65         self.puck.pos[0] += self.puck.vx * self.puck.velocity
66         self.puck.pos[1] += self.puck.vy * self.puck.velocity
67
68     def get_pos(self, p):
69         return self.players[p]
70
71     def connected(self):
72         return self.ready
73
74     def get_winner(self):
```

```
75     return self.winner
76
77     def resetWent(self):
78         self.players = [Player(120, 120, YELLOW), Player(480, 120, GREEN),
79                         Player(120, 480, BLUE), Player(480, 480, PINK)]
80         self.puck = Puck(300, 300, (221, 221, 221))
81         self.winner = None
```

- **players**: Khởi tạo danh sách các người chơi trong trò chơi.
- **ready**: Biến kiểm tra xem trò chơi đã sẵn sàng chưa.
- **puck**: Khởi tạo đối tượng bóng trong trò chơi.
- **winner**: Người chơi chiến thắng trong trò chơi.

Các phương thức chính:

- **end_game()**: Kết thúc trò chơi và xác định người chơi chiến thắng.
- **play()**: Xử lý các thao tác di chuyển của người chơi.
 - Đầu tiên, chuỗi moves được chia thành danh sách các hành động riêng biệt bằng cách sử dụng phương thức split('&&').
 - * Nếu 'left' có trong moves, và người chơi p di chuyển sang trái mà không vượt quá giới hạn bên trái của cửa sổ chơi, thì vị trí x của người chơi p được giảm đi một lượng bằng vận tốc (self.players[p].velocity).
 - * Tương tự, nếu 'right' có trong moves, và người chơi p di chuyển sang phải mà không vượt quá giới hạn bên phải của cửa sổ chơi, thì vị trí x của người chơi p được tăng lên một lượng bằng vận tốc.
 - * Tương tự, nếu 'up' có trong moves, và người chơi p di chuyển lên trên mà không vượt quá giới hạn phía trên của cửa sổ chơi, thì vị trí y của người chơi p được giảm đi một lượng bằng vận tốc.
 - * Tương tự, nếu 'down' có trong moves, và người chơi p di chuyển xuống dưới mà không vượt quá giới hạn phía dưới của cửa sổ chơi, thì vị trí y của người chơi p được tăng lên một lượng bằng vận tốc.
- **update()**: Cập nhật trạng thái của trò chơi.
- **update_puck()**: Cập nhật trạng thái của bóng trong trò chơi.
 - Đầu tiên, vòng lặp for được sử dụng để kiểm tra xem bóng có va chạm với người chơi nào không.
 - Đối với mỗi người chơi pl, mã tính toán khoảng cách giữa vị trí của người chơi và vị trí của puck bằng cách sử dụng công thức Euclid: $dis = \sqrt{x^2 + y^2}$, trong đó x là hiệu của tọa độ x giữa người chơi và puck, và y là hiệu của tọa độ y giữa người chơi và puck.
 - Nếu khoảng cách dis nhỏ hơn hoặc bằng tổng bán kính của người chơi và bán kính của puck (pl.radius + self.puck.radius), tức là người chơi và puck va chạm, thì xác định hướng di chuyển của puck dựa trên vị trí của puck và người chơi:

- * Nếu vị trí x của puck lớn hơn vị trí x của người chơi, thì hướng di chuyển theo chiều x của puck là 1, ngược lại là -1.
 - * Tương tự, nếu vị trí y của puck lớn hơn vị trí y của người chơi, thì hướng di chuyển theo chiều y của puck là 1 (di chuyển sang phải), ngược lại là -1.
 - * Đặt biến `is_collided` của người chơi pl thành True để xác định rằng người chơi và puck đã va chạm.
 - Nếu puck không va chạm với bất kỳ người chơi nào, biến `pl.is_collided` của tất cả các người chơi được đặt thành False.
 - Code kiểm tra vị trí của puck để xác định xem puck có đi qua các vùng giới hạn của trò chơi hay không. Nếu puck đi qua một trong các vùng giới hạn dưới đây, trò chơi kết thúc và người chơi tương ứng được xác định là người chiến thắng:
 - * Nếu tọa độ x của puck trừ bán kính của puck nhỏ hơn hoặc bằng 70 và tọa độ y của puck nhỏ hơn hoặc bằng 300, người chơi màu vàng ('Yellow') chiến thắng.
 - * Nếu tọa độ y của puck trừ bán kính của puck nhỏ hơn hoặc bằng 70 và tọa độ x của puck nhỏ hơn hoặc bằng 300, người chơi màu xanh lá cây ('Green') chiến thắng.
 - * Nếu tọa độ x của puck trừ bán kính của puck nhỏ hơn hoặc bằng 70 và tọa độ y của puck lớn hơn 300, người chơi màu xanh dương ('Blue') chiến thắng.
 - * Nếu tọa độ x của puck cộng bán kính của puck lớn hơn hoặc bằng 530 và tọa độ y của puck lớn hơn 300, người chơi màu hồng ('Pink') chiến thắng.
 - Cuối cùng, vị trí của puck được cập nhật bằng cách thay đổi các tọa độ x và y của puck dựa trên vận tốc và hướng di chuyển của puck (`self.puck.vx` và `self.puck.vy`). Cụ thể, các tọa độ x và y của puck được tăng lên với giá trị `self.puck.vx * self.puck.velocity` và `self.puck.vy * self.puck.velocity` tương ứng.
- **get_pos()**: Lấy vị trí của một người chơi.
 - **connected()**: Kiểm tra xem trò chơi đã sẵn sàng chưa.
 - **get_winner()**: Lấy thông tin về người chơi chiến thắng.
 - **resetWent()**: Thiết lập lại trạng thái của trò chơi.

3.3 Thiết kế Network

Trong mô hình socket, lớp Network được sử dụng để tạo và quản lý kết nối mạng giữa client và server. Lớp này có nhiệm vụ thiết lập kết nối TCP/IP giữa client và server thông qua giao thức socket, gửi và nhận dữ liệu qua kết nối này.

- Lớp Network được sử dụng trong phần client (`client.py`) để thiết lập kết nối với server thông qua socket, gửi và nhận dữ liệu với server.
- Trong `__init__`, nó khởi tạo một socket và cố gắng kết nối đến địa chỉ và cổng đã chỉ định.
- Phương thức `send()` được sử dụng để gửi dữ liệu từ client đến server.
- Phương thức `connect()` được sử dụng để thiết lập kết nối với server.
- Phương thức `getP()` được sử dụng để nhận thông tin về vai trò của client trong trò chơi (ví dụ: người chơi 1 hoặc người chơi 2).



- Trong phần server (`server.py`), lớp `Network` cũng được sử dụng để nhận dữ liệu từ client và gửi dữ liệu trả về cho client.

Tóm lại, lớp `Network` là một cơ chế trung gian cho việc giao tiếp mạng giữa client và server trong ứng dụng, giúp trừu tượng hóa việc quản lý kết nối mạng và truyền dữ liệu.

```
1 import socket
2 import pickle
3
4 class Network:
5     def __init__(self):
6         self.client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7         self.server = "localhost"
8         self.port = 5555
9         self.addr = (self.server, self.port)
10        self.p = self.connect()
11
12    def getP(self):
13        return self.p
14
15    def connect(self):
16        try:
17            self.client.connect(self.addr)
18            return self.client.recv(2048).decode()
19        except:
20            pass
21
22    def send(self, data):
23        try:
24            self.client.send(str.encode(data))
25            return pickle.loads(self.client.recv(2048*2))
26        except socket.error as e:
27            print(e)
```

3.4 Thiết kế Server Socket

Mục đích: Lớp `server.py` là một phần của hệ thống server trong trò chơi ping pong, điều khiển việc kết nối và giao tiếp giữa client và server.

Chức năng chính:

- **Khởi tạo và Quản lý Kết nối:**

- Lớp này tạo ra một máy chủ socket và lắng nghe các kết nối đến từ client.
- Khi một client kết nối, nó bắt đầu một luồng riêng biệt để xử lý các yêu cầu từ client đó.

- **Xử lý Dữ liệu:**

- Khi kết nối được thiết lập, nó nhận dữ liệu từ client và xử lý theo yêu cầu của client, chẳng hạn như di chuyển của người chơi.
- Sau đó, nó gửi lại dữ liệu mới nhất về trạng thái của trò chơi cho client.



- **Quản lý Trò chơi:**

- Lớp này quản lý các trò chơi ping pong đang diễn ra, theo dõi trạng thái của từng trò chơi và cập nhật thông tin của chúng.
- Khi một trò chơi kết thúc, nó thông báo cho tất cả các client liên quan và chuẩn bị cho trò chơi mới.

- **Điều khiển Luồng:**

- Lớp này sử dụng các luồng riêng biệt để xử lý đồng thời các kết nối từ nhiều client, đảm bảo rằng máy chủ có thể phục vụ nhiều người chơi cùng một lúc.

```
1 server = "localhost"
2 port = 5555
3 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4
5 try:
6     s.bind((server, port))
7 except socket.error as e:
8     str(e)
9
10 s.listen(2)
11 print("Waiting for a connection, Server Started")
```

Chúng ta sử dụng giao thức TCP để chấp nhận kết nối từ các client và xử lý trò chơi đa người chơi.

Tạo socket với hàm `socket()` chúng ta có thể truyền vào tham số hoặc để trống, ở đây ta truyền vào 2 tham số là hằng `socket.AF_INET` (tham số truyền vào phiên bản IP chúng ta sẽ sử dụng ở đây là phiên bản 4), tiếp theo là hằng số `socket.SOCK_STREAM` chỉ loại kết nối TCP IP hoặc UDP,..

Máy chủ được liên kết với địa chỉ IP "localhost" và cổng "5555" bằng cách sử dụng `s.bind((server, port))`. Nếu liên kết không thành công, mã sẽ in ra lỗi.

Máy chủ lắng nghe kết nối đến với số lượng tối đa là 2 kết nối bằng cách sử dụng `s.listen(2)`.

Máy chủ sẽ in ra thông báo "Waiting for a connection, Server Started" để cho biết máy chủ đã bắt đầu chạy và đang chờ kết nối từ client.

```
1 connected = set()
2 games = {}
3 idCount = 0
```

Một số biến được khởi tạo:

- **connected:** một tập hợp (set) để lưu trữ các kết nối đã được thiết lập.
- **games:** một dict (dictionary) để lưu trữ các trò chơi đang diễn ra, với khóa là gameId và giá trị là đối tượng Game tương ứng.
- **idCount:** một biến đếm để đếm số client đã kết nối và phân biệt các client.

```
1 def threaded_client(conn, p, gameId):
2     global idCount
3     conn.send(str.encode(str(p)))
```



```
4 while True:
5     try:
6         data = conn.recv(4096).decode()
7         if gameId in games:
8             game = games[gameId]
9             game.update()
10            if not data:
11                break
12            else:
13                if data == "reset":
14                    game.resetWent()
15                elif data != "get":
16                    game.play(p, data)
17                conn.sendall(pickle.dumps(game))
18            else:
19                break
20        except:
21            break
22    print("Lost connection")
23    try:
24        del games[gameId]
25        print("Closing Game", gameId)
26    except:
27        pass
28    idCount -= 1
29    conn.close()
```

Hàm `threaded_client()` được sử dụng để xử lý kết nối từ client. Hàm này nhận các tham số là `conn` (kết nối socket), `p` (vị trí của người chơi trong trò chơi), và `gameId` (id của trò chơi).

Trong vòng lặp vô hạn `while True`, máy chủ nhận dữ liệu từ client bằng cách sử dụng `data = conn.recv(4096).decode()`. Nếu `gameId` tồn tại trong `games`, máy chủ sẽ cập nhật trạng thái của trò chơi và xử lý dữ liệu nhận được từ client.

Nếu không có dữ liệu nhận được từ client, vòng lặp dừng lại và kết thúc luồng xử lý cho client đó.

Nếu dữ liệu nhận được từ client là "reset", máy chủ đặt lại trạng thái của trò chơi bằng cách sử dụng `game.resetWent()`. Nếu dữ liệu nhận được từ client không phải là "get", máy chủ chơi lượt cho người chơi `p` với dữ liệu nhận được từ client bằng cách sử dụng `game.play(p, data)`.

Máy chủ gửi lại trạng thái của trò chơi cho client bằng cách sử dụng `conn.sendall(pickle.dumps(game))`.

Nếu `gameId` không tồn tại trong `games`, vòng lặp dừng lại và kết thúc luồng xử lý cho client đó. Máy chủ in ra thông báo "Lost connection" để cho biết kết nối đã bị mất và xóa trò chơi có `gameId` tương ứng trong `games` bằng cách sử dụng `del games[gameId]`. Biến `idCount` được giảm đi 1 sau đó kết nối với client được đóng bằng cách sử dụng `conn.close()`.

```
1 while True:
2     conn, addr = s.accept()
3     print("Connected to:", addr)
4     idCount += 1
5     gameId = (idCount - 1) // 4
6     if idCount % 4 == 1:
7         games[gameId] = Game(gameId)
8         print("Creating a new game...")
9     elif idCount % 4 == 0:
```

```
10     games[gameId].ready = True
11     p = (idCount - 1) % 4
12     start_new_thread(threaded_client, (conn, p, gameId))
```

Trong vòng lặp vô hạn `while True`, máy chủ chấp nhận kết nối từ client bằng cách sử dụng `conn, addr = s.accept()`. Khi một kết nối được thiết lập thành công, máy chủ in ra thông báo "Connected to:" và địa chỉ của client. Biến `idCount` được tăng lên 1.

`gameId` được tính toán dựa trên giá trị của `idCount`. Nếu `idCount` chia hết cho 4 dư 1, tức là một trò chơi mới được tạo ra. Máy chủ tạo một đối tượng `Game` mới với `gameId` tương ứng và lưu trữ nó trong `games`. Máy chủ in ra thông báo "Creating a new game...".

Nếu `idCount` chia hết cho 4 dư 0, tức là đã có đủ 4 người chơi trong một trò chơi. Máy chủ đánh dấu trạng thái "ready" của trò chơi đó bằng cách sử dụng `games[gameId].ready = True`.

Biến `p` được tính toán dựa trên giá trị của `idCount` để xác định người chơi thứ mấy trong trò chơi.

Phương thức `threaded_client` được gọi trong một luồng mới với các tham số tương ứng.

3.5 Xây dựng Client

3.5.1 Thiết kế giao diện người dùng

```
1  pygame.font.init()
2  width = 600
3  height = 600
4  logo = pygame.image.load('Assets/logo.png')
5  win = pygame.display.set_mode((width, height))
6  pygame.display.set_icon(logo)
7  pygame.display.set_caption("Nhóm 4 - Ping Pong")
```

Đầu tiên, chúng ta khởi tạo môi trường Pygame bằng cách sử dụng `pygame.font.init()`.

Sau đó, chúng ta khởi tạo cửa sổ chơi với kích thước 600x600 bằng cách sử dụng `pygame.display.set_mode((width, height))`.

Hình ảnh logo của trò chơi được tải lên từ tệp "logo.png" và được sử dụng làm biểu tượng cho cửa sổ chơi bằng cách sử dụng `pygame.display.set_icon(logo)`.

Tiêu đề của cửa sổ chơi được đặt là "Nhóm 4 - Ping Pong" bằng cách sử dụng `pygame.display.set_caption("Nhóm 4 - Ping Pong")`.

```
1  def redrawWindow(win, game):
2      if not (game.connected()):
3          pygame.draw.rect(win, PINK, (0, 0, 150, 600))
4          pygame.draw.rect(win, BLUE, (150, 0, 150, 600))
5          pygame.draw.rect(win, GREEN, (300, 0, 150, 600))
6          pygame.draw.rect(win, YELLOW, (450, 0, 150, 600))
7          font = pygame.font.SysFont("SF Pro Display", 130, True)
8          font2 = pygame.font.SysFont("SF Pro Text", 30)
9          t1 = font.render("P", True, (255, 255, 255))
10         t2 = font.render("O", True, (255, 255, 255))
11         t3 = font.render("N", True, (255, 255, 255))
12         t4 = font.render("G", True, (255, 255, 255))
13         t5 = font2.render("Waiting for Players!!", True, (255, 255, 255))
14         win.blit(t1, (40, 250))
15         win.blit(t2, (190, 250))
```

```
16     win.blit(t3, (340, 250))
17     win.blit(t4, (490, 250))
18     win.blit(t5, (200, 500))
19     else:
20         win.fill((37, 37, 37))
21         pygame.draw.rect(win, PINK, (300, 300, 235, 235))
22         pygame.draw.rect(win, BLUE, (65, 300, 235, 235))
23         pygame.draw.rect(win, GREEN, (300, 65, 235, 235))
24         pygame.draw.rect(win, YELLOW, (65, 65, 235, 235))
25         pygame.draw.rect(win, (16, 16, 16), (70, 70, 460, 460))
26         game.puck.draw(win)
27         for pl in game.players:
28             pl.draw(win)
29
30     pygame.display.update()
```

Hàm `redrawWindow()` được sử dụng để vẽ lại cửa sổ chơi với trạng thái mới nhất của trò chơi.

Nếu trò chơi chưa kết nối (`game.connected()` trả về `False`), hàm sẽ vẽ một cửa sổ chờ với các hình chữ nhật màu sắc khác nhau và văn bản "Waiting for Players!!" để hiển thị thông báo đang chờ người chơi.

Nếu trò chơi đã kết nối, hàm sẽ vẽ một cửa sổ chơi với màu nền đen và các hình chữ nhật màu sắc khác nhau để đại diện cho các người chơi và bóng. Hàm cũng vẽ người chơi và bóng lên cửa sổ chơi bằng cách sử dụng phương thức `draw()` của các đối tượng `Player` và `Puck`.

Cuối cùng, hàm cập nhật cửa sổ chơi bằng cách sử dụng `pygame.display.update()`.

```
1  def menu_screen():
2      waiting = True
3      clock = pygame.time.Clock()
4      while waiting:
5          clock.tick(60)
6          win.fill((232, 63, 111))
7          font = pygame.font.SysFont("SF Pro Text", 30)
8          text = font.render("Click, if you're ready", True, (255, 255, 255))
9          win.blit(text, (190, 300))
10         pygame.display.update()
11         for event in pygame.event.get():
12             if event.type == pygame.QUIT:
13                 pygame.quit()
14                 waiting = False
15             if event.type == pygame.MOUSEBUTTONDOWN:
16                 waiting = False
17     main()
```

Hàm `menu_screen()` được sử dụng để hiển thị màn hình menu trước khi bắt đầu trò chơi.

Trong vòng lặp `while waiting`, hàm vẽ màn hình menu với màu nền hồng và văn bản "Click, if you're ready" để hướng dẫn người chơi.

Hàm kiểm tra các sự kiện của người chơi bằng cách sử dụng `pygame.event.get()`. Nếu người chơi nhấn nút thoát, trò chơi sẽ kết thúc bằng cách sử dụng `pygame.quit()` và biến `waiting` được đặt thành `False`. Nếu người chơi nhấn chuột, biến `waiting` cũng được đặt thành `False`.

Cuối cùng, hàm chuyển sang chế độ chơi game bằng cách gọi hàm `main()`.

3.5.2 Thiết kế chức năng chơi game

```
1 def main():
2     run = True
3     clock = pygame.time.Clock()
4     n = Network()
5     player = int(n.getP())
6     print("You are player", player)
7     while run:
8         try:
9             game = n.send("get")
10        except:
11            run = False
12            print("Couldn't get game")
13            break
14        if game.get_winner() is not None:
15            font = pygame.font.SysFont("SF Pro Text", 30)
16            text = font.render("Player " + game.get_winner() + " Lose ", True, (255,
17                                255, 255))
18            win.blit(text, (215, 300))
19            pygame.display.update()
20            time.sleep(3)
21            n.send("reset")
22
23        keys = pygame.key.get_pressed()
24        if game.connected():
25            moves = []
26            if keys[pygame.K_LEFT]:
27                moves.append('left')
28            if keys[pygame.K_RIGHT]:
29                moves.append('right')
30            if keys[pygame.K_UP]:
31                moves.append('up')
32            if keys[pygame.K_DOWN]:
33                moves.append('down')
34            if len(moves) > 0:
35                n.send('&&'.join(moves))
36        for event in pygame.event.get():
37            if event.type == pygame.QUIT:
38                run = False
39                pygame.quit()
40        redrawWindow(win, game)
41        clock.tick(60)
```

Hàm `main()` chứa các chức năng chính của trò chơi ping pong.

Trước hết, biến `run` được khởi tạo là `True` và biến `clock` được khởi tạo để theo dõi tốc độ khung hình.

Một kết nối mạng `n` được tạo ra bằng cách sử dụng lớp `Network`. Người chơi được xác định bằng cách gọi phương thức `n.getP()` và lưu vào biến `player`.

Trong vòng lặp `while run`, trò chơi được lấy từ server bằng cách sử dụng `n.send("get")`. Nếu không thể lấy được trò chơi, biến `run` được đặt thành `False` và vòng lặp dừng lại.

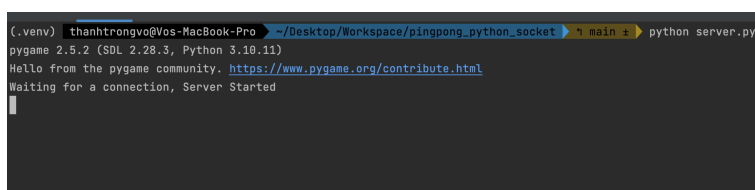
Nếu người chơi chiến thắng được xác định, một thông báo với văn bản "Player [player]"

Lose" sẽ được hiển thị trên cửa sổ chơi. Sau đó, trò chơi sẽ được đặt lại bằng cách sử dụng `n.send("reset")`.

Phím được nhấn bởi người chơi được xác định bằng cách sử dụng `pygame.key.get_pressed()`. Nếu trò chơi đã kết nối, các hành động di chuyển của người chơi được xác định bằng cách kiểm tra các phím mũi tên được nhấn và gửi đến server bằng cách sử dụng `n.send('&&'.join(moves))`.

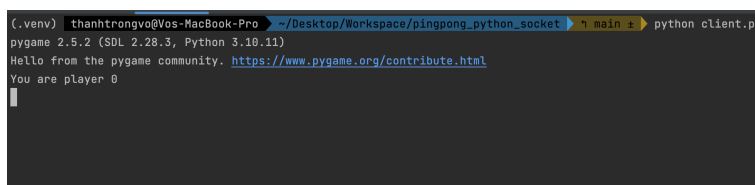
Cuối cùng, vòng lặp xử lý các sự kiện của người chơi và vẽ lại cửa sổ chơi bằng cách sử dụng hàm `redrawWindow()` và cập nhật cửa sổ chơi với tốc độ 60 khung hình mỗi giây bằng cách sử dụng `clock.tick(60)`.

3.6 Kết quả thực thi chương trình



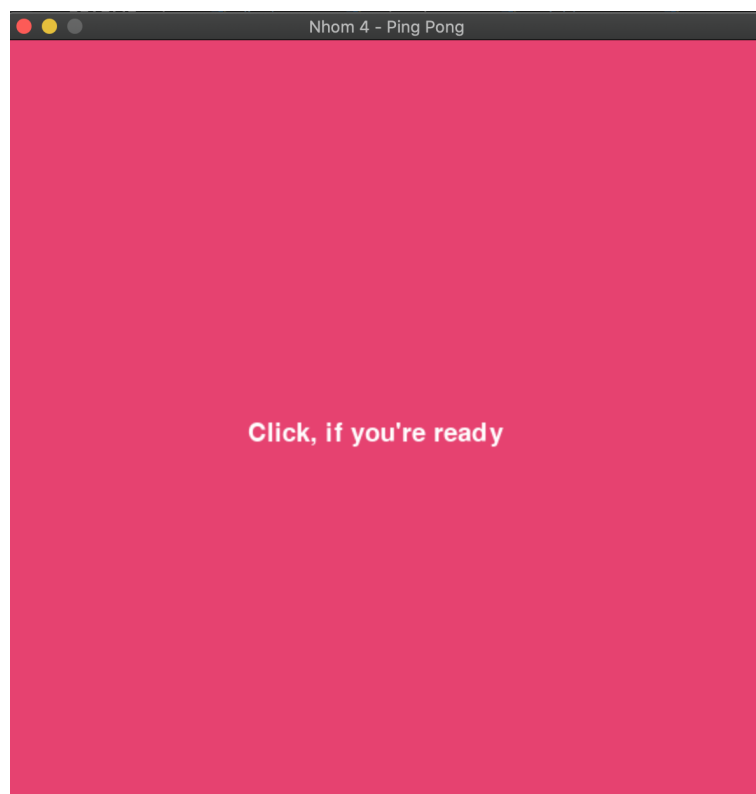
```
(.venv) thanhtrongvo@Vos-MacBook-Pro: ~/Desktop/Workspace/pingpong_python_socket [main] python server.py
pygame 2.5.2 (SDL 2.28.3, Python 3.10.11)
Hello from the pygame community. https://www.pygame.org/contribute.html
Waiting for a connection, Server Started
```

Hình 3: Kết quả khi thực thi server.py

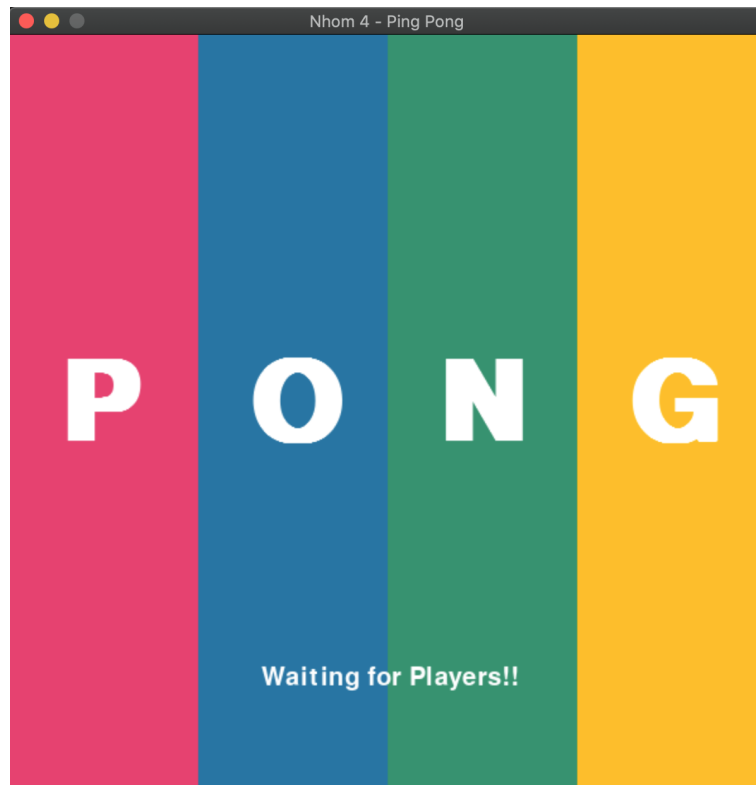


```
(.venv) thanhtrongvo@Vos-MacBook-Pro: ~/Desktop/Workspace/pingpong_python_socket [main] python client.py
pygame 2.5.2 (SDL 2.28.3, Python 3.10.11)
Hello from the pygame community. https://www.pygame.org/contribute.html
You are player 0
```

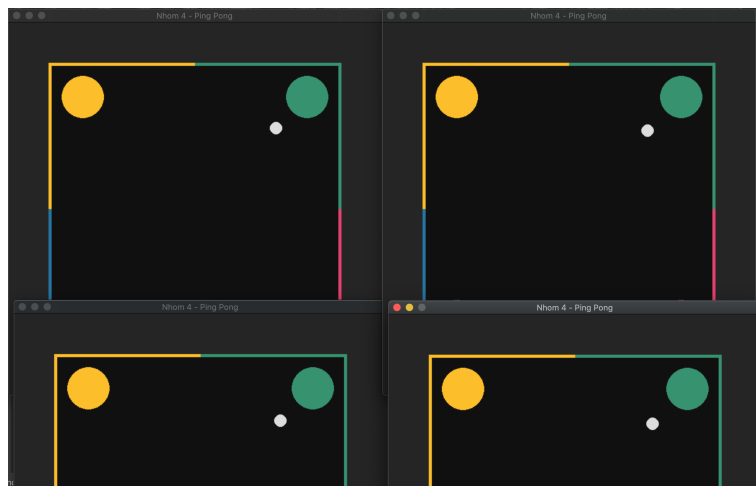
Hình 4: Kết quả khi thực thi client.py



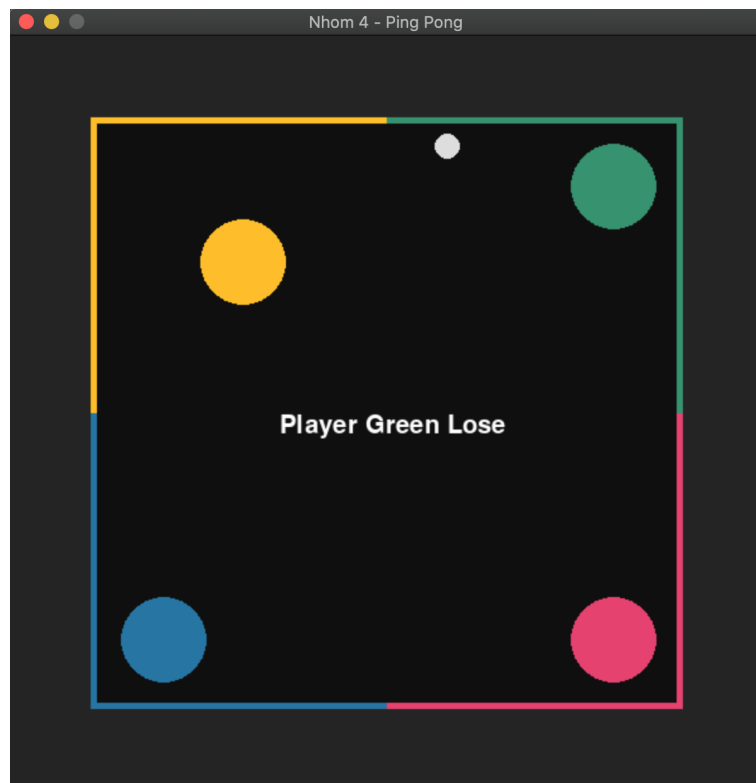
Hình 5: Cửa sổ chờ người chơi ready



Hình 6: Cửa sổ chờ người chơi khác



Hình 7: Cửa sổ chơi game



Hình 8: Thông báo người kết thúc trò chơi