TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN

PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

Xây dựng game

Online trên PYTHON

GVHD: Từ Lãng Phiêu
SV: SV1 - MSSV
Trần Giang Nam - 3120410336
SV2 - MSSV
Nguyễn Quốc Duệ - 3120560009
SV3 - MSSV
Nguyễn Việt Hoàng - 3120410182
Liên hệ: giangnam.050402@gmail.com

TP. HỒ CHÍ MINH, THÁNG 2/2024

Mục lục

1 Giới thiệu chương trình **2**

2 Cơ sở lý thuyết **3**
2.1 Tổng quan về Python . . . 3
2.1.1 Đặc điểm nổi bật của Python: . . . 3
2.1.2 Ứng dụng của Python: . . . 3
2.2 Pygame . . . 3
2.2.1 Đặc điểm nổi bật của lập trình Pygame . . . 4
2.2.2 Cấu trúc một chương trình Pygame . . . 4
2.2.3 Cách cài thư viện Pygame trong Python . . . 5
2.2.4 Các thành phần cấu tạo nên Pygame . . . 5
2.3 Đa luồng(Multithreading) . . . 6
2.3.1 Luồng (thread) là gì ? Sự khác nhau giữa thread và process . . . 6
2.3.2 Đa luồng (Multithreading) . . . 7
2.3.3 Đồng bộ hóa các Thread trong Python . . . 8
2.4 Socket . . . 8
2.4.1 Tạo server . . . 9
2.4.2 Tạo client . . . 10

3 Thiết kế ứng dụng **11**
3.1 Mô tả thiết kế của chương trình . . . 11
3.2 Cấu trúc mã nguồn . . . 11
3.3 Mô hình ứng dụng . . . 17
3.4 Flowchart . . . 18

4 Hiện thực ứng dụng **20**
4.1 Giao diện Menu . . . 20
4.2 Chế độ Offline . . . 21
4.3 Chế độ Online . . . 22

5 Cách thức cài đặt ứng dụng **24**
5.1 Cài đặt python . . . 24
5.2 Cài đặt pip và pygame . . . 24
5.3 Cài đặt Game . . . 24
5.4 Chạy File Game . . . 25

6 Nhiệm vụ và vai trò của từng thành viên **26**

1 Giới thiệu chương trình

Đề tài của chúng em là xây dựng game cờ Caro online có thể chạy được trên linux. Đây là chương trình được xây dựng bằng ngôn ngữ Python. Có kết hợp với đa luông và socket.

Cờ Caro, còn được gọi là X và O, là một trò chơi trí tuệ đơn giản nhưng hấp dẫn dành cho hai người chơi. Mỗi người chơi lần lượt đánh dấu X hoặc O vào các ô vuông trên bàn cờ , với mục tiêu tạo thành một hàng, cột hoặc đường chéo gồm 5 ký hiệu giống nhau để giành chiến thắng. Trò chơi này đòi hỏi tư duy logic, chiến lược và khả năng dự đoán nước đi của đối thủ.

Việc phát triển game Cờ Caro 2 người chơi bằng Python mang đến nhiều lợi ích:

- Tính giải trí: Cung cấp một trò chơi trí tuệ thú vị để giải trí và rèn luyện tư duy.
- Giáo dục: Game Cờ Caro có thể được sử dụng như một công cụ giáo dục để dạy trẻ em về các khái niệm logic, chiến lược và tư duy phản biện. Việc chơi game giúp trẻ rèn luyện khả năng tập trung, kiên nhẫn và đưa ra quyết định.
- Học lập trình Python: Giúp người chơi học lập trình Python thông qua việc thực hành xây dựng một ứng dụng đơn giản nhưng đầy đủ chức năng.
- Tính di động: Chạy trên hệ điều hành Linux, mang lại sự linh hoạt và khả năng tiếp cận cho người chơi.
- Cộng đồng hỗ trợ: Lợi ích từ cộng đồng lập trình Python rộng lớn để được hỗ trợ và giải đáp thắc mắc.

Với những ưu điểm trên, game Cờ Caro 2 người chơi bằng Python trên Linux là một dự án lập trình thú vị và bổ ích cho cả người mới bắt đầu và người lập trình có kinh nghiệm.

2 Cơ sở lý thuyết

2.1 Tổng quan về Python

Python là một ngôn ngữ lập trình đa năng, cấp cao, được sử dụng rộng rãi trong nhiều lĩnh vực như khoa học dữ liệu, trí tuệ nhân tạo, phát triển web, tự động hóa và ứng dụng phần mềm. Nó được đánh giá cao bởi tính dễ học, dễ đọc, dễ sử dụng và có cộng đồng người dùng lớn, năng động.

2.1.1 Đặc điểm nổi bật của Python:

- Dễ học, dễ đọc: Python có cú pháp đơn giản, gần gũi với ngôn ngữ tự nhiên, giúp người mới bắt đầu dễ dàng tiếp cận và học nhanh chóng.
- Mã nguồn ngắn gọn: So với các ngôn ngữ lập trình khác, Python thường viết code ngắn gọn hơn, tiết kiệm thời gian và công sức cho lập trình viên.
- Giải thích: Python hỗ trợ tính năng giải thích (interpreter) giúp thực thi code trực tiếp mà không cần biên dịch, cho phép kiểm tra và sửa lỗi nhanh chóng.
- Đa dạng thư viện: Python sở hữu hệ sinh thái thư viện phong phú, cung cấp sẵn các công cụ cho nhiều lĩnh vực khác nhau, từ khoa học dữ liệu, học máy đến phát triển web, tự động hóa.
- Cộng đồng lớn: Python có cộng đồng người dùng rộng rãi, năng động, sẵn sàng hỗ trợ và chia sẻ kiến thức, giúp lập trình viên dễ dàng tìm kiếm giải pháp cho vấn đề gặp phải.

2.1.2 Ứng dụng của Python:

- Khoa học dữ liệu: Python được sử dụng phổ biến trong phân tích dữ liệu, học máy, trí tuệ nhân tạo nhờ các thư viện mạnh mẽ như NumPy, Pandas, scikit-learn, TensorFlow.
- Phát triển web: Python được ứng dụng rộng rãi trong xây dựng web backend với các framework như Django, Flask.
- Tự động hóa: Python được sử dụng để tự động hóa các tác vụ lặp đi lặp lại, tiết kiệm thời gian và công sức cho con người.
- Phát triển ứng dụng: Python được sử dụng để phát triển nhiều loại ứng dụng khác nhau như ứng dụng desktop, ứng dụng mobile, game.

2.2 Pygame

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. Pygame được viết bởi Pete Shinners thay thế cho chương trình PySDL sau khi quá trình phát triển dự án này bị đình trệ. Chính thức phát hành từ năm 2000, Pygame được phát hành theo phần mềm miễn phí GNU Lesser General Public License.

Pygame có thể chạy trên nhiều nền tảng và hệ điều hành khác nhau. Với thư viện pygame trong Python, các nhà phát triển có thể sử dụng công cụ và chức năng mở rộng để tạo ra các trò chơi nhập vai ấn tượng. Bởi vậy, Pygame đang ngày càng phổ biến với nhà phát triển vì tính đơn giản, linh hoạt, dễ sử dụng.

2.2.1 Đặc điểm nổi bật của lập trình Pygame

- Pygame sử dụng Simple DirectMedia Layer (SDL), một thư viện phát triển đa nền tảng cho phép các nhà phát triển có thể truy cập vào phần cứng máy tính như đồ họa, âm thanh và thiết bị đầu vào.
- Xây dựng các trò chơi trên nhiều nền tảng khác nhau như Windows, Mac, Linux thậm chí là cả các thiết bị di động
- Nhà phát triển 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
- Cung cấp nhiều chức năng mở rộng hỗ trợ nhà phát triển tập trung phát triển trò chơi
- API trực quan và dễ hiểu, hỗ trợ người mới sử dụng hay cả những nhà phát triển có kinh nghiệm đều có thể truy cập được
- Nguồn tài nguyên và tài liệu phong phú, các nhà phát triển có thể sử dụng các mã nguồn mở miễn phí để phát triển dự án của mình
- Tính đa phương tiện giúp nhà phát triển có thể ứng dụng để xử lí hình ảnh hay video, mô phỏng, công cụ giáo dục. . . .

2.2.2 Cấu trúc một chương trình Pygame

Một chương trình Pygame sẽ bao gồm các phần sau:

- Import pygame, sys: Gọi thư viện pygame và thư viện sys
- Pygame.init: Khởi tạo một chương trình pygame
- Screen = pygame.display.set_mode ((400,300)): Khởi tạo kích thước màn hình game có ngang là 400, chiều dọc là 300
- pygame.display.set_caption('Hello World!'): Tiêu đề của chương trình Pygame
- while True: Vòng lặp vô hạn
- for event in pygame.event.get(): Vòng lặp bắt các sự kiện của chương trình pygame
- if event.type == pygame.QUIT: sự kiện ấn thoát game
- pygame.quit(): Thoát khỏi chương trình pygame
- sys.exit(): Thoát khỏi hệ thống

Hệ tọa độ trong Pygame bao gồm:

- O: Gốc tọa độ nằm ở góc trên bên trái của màn hình game
- X: Trục X là trục ngang. Hướng từ gốc O sang phải có tọa độ dương, hướng từ gốc O sang trái tọa độ âm
- Y: Trục Y là trục dọc. Hướng từ gốc O trên xuống dưới có tọa độ dương, hướng từ gốc O lên trên có tọa độ âm

2.2.3 Cách cài thư viện Pygame trong Python

Để cài đặt thư viện Pygame trong Python bạn thực hiện theo các bước dưới đây:

- Bước 1: Trước khi cài đặt thư viện Pygame trong Python, hãy đảm bảo rằng bạn đã cài Python trên máy tính của mình. Sau đó truy cập Python.org và tải xuống phiên bản tương thích với hệ điều hành của mình.
- Bước 2: Chạy file "Python-version" trong terminal
- Bước 3: Để cài đặt lập trình Pygame, bạn cần sử dụng trình quản lý gói pip (thường đi xem với các bản cài đặt Python). Chạy thiết bị đầu cuối và thực hiện lệnh "pip install pygame". Lệnh sẽ tải xuống và cài đặt phiên bản Pygame mới nhất từ Python Package Index (PyPI)
- Bước 4: Tạo một file mới bằng Python và nhập mô-đun Pygame bằng cách thêm dòng "Nhập pygame", lưu tập có đuôi ".py" và bắt đầu trải nghiệm thư viện Pygame

2.2.4 Các thành phần cấu tạo nên Pygame

Việc nắm được các thành phần chính của Pygame sẽ giúp bạn tận dụng và lập trình game với Pygame dễ dàng hơn. Bên cạnh đó, làm quen với mô-đun của Pygame, hoạ tiết, bề mặt, xử lí sự kiện, âm thanh, âm nhạc sẽ giúp bạn nắm được các công cụ cần thiết để lập trình game bằng Pygame Python và tạo ra các trò chơi hấp dẫn.

Dưới đây là các thành phần cấu tạo nên Pygame:

- Pygame Module (Mô-đun Pygame): Đóng vai trò là khối xây dựng để giúp phát triển từng bước tạo ra trò chơi. Mô-đun hỗ trợ nhà phát triển các tác vụ như quản lý cửa sổ trò chơi, xử lý sự kiện và hiển thị đồ họa màn hình. Bằng cách kết hợp mô-đun Pygame vào code của mình, nhà phát triển sẽ tận dụng được tối đa tính năng này
- Display Surface (Bề mặt hiển thị): Đóng vai trò rất lớn trong Pygame vì đây là nơi trò chơi của nhà phát triển hiển thị. Màn hình sẽ hiển thị lần lượt các đối tượng, hình ảnh, hoạt ảnh cũng như môi trường đồ hoạ trong trò chơi. Với chức năng này, nhà phát triển cũng có thể điều chỉnh kích thước, tiêu đề và các thuộc tính theo ý muốn của mình.
- Sprites (Nhân vật): Là một trong những yếu tố thiết yếu để phát triển trò chơi. Nhân vật sẽ đại diện cho thực thể trong thế giới trò chơi. Thư viện Pygame đơn giản hoá việc quản lý các nhân vật bằng cách cung cấp các nhân vật có sẵn được hiển thị trên màn hình. Pygame cũng cung cấp chức năng để nhà phát triển phát hiện được những xung đột giữa tương tác nhân vật và các hoạ tiết được thêm vào
- Surface (Bề mặt): Các bề mặt trong Pygame là bơi hiển thị đồ hoạ như hình ảnh, văn bản... Với thư viện Pygame, nhà phát triển có thể tùy ý tải lên các hình ảnh khác nhau
- Event Handling (Xử lí sự kiện): Thao tác này được Pygame đơn giản hóa bằng quá trình phát hiện và phản hồi các tương tác của người dùng. Bằng việc sử dụng mô-đun sự kiện do Pygame cung cấp, nhà phát triển có thể nắm bắt các sự kiện như nhấn phím, nhấp chuột và thay đổi kích thước cửa sổ. Chức năng này cho phép nhà phát triển tạo các trò chơi tương tác phản hồi linh hoạt, tạo điều kiện thuận lợi cho việc triển khai trò chơi và giao diện người dùng

- Sound and Music (Âm thanh và Âm nhạc): Đây là hai thành tố không thể thiếu trong trò chơi, Pygame cung cấp các chức năng để nhà điều khiển có thể tự mình tạo ra các âm thanh cho trò chơi của mình. Bên cạnh đó, mô-đun trộn trong Pygame còn cho phép nhà phát triển tải và phát hiệu ứng âm thanh, nhạc nền, điều chỉnh âm lượng... để giúp trò chơi sống động hơn
- Collision Detection (Phát hiện va chạm): Đây là một trong những khía cạnh quan trọng trong quá trình phát triển trò chơi. Pygame cung cấp cơ chế tích hợp để phát hiện va chạm giữa các họa tiết và các đối tượng bằng việc sử dụng tính năng phát hiện va chạm, nhà điều khiển có thể phát hiện và phản hồi các va chạm, triển khai cơ chế chơi trò chơi như tính điểm và tạo tương tác thực tế giữa với các nhân vật trong thế giới của mình

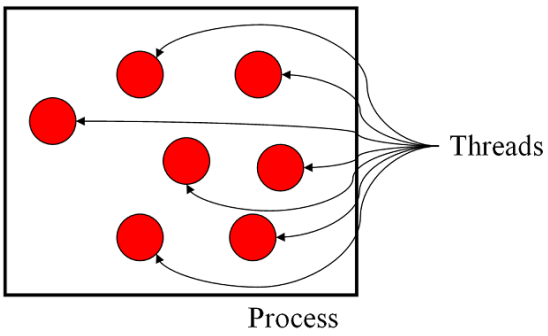
Việc liên tục tìm hiểu và ứng dụng các thành phần trong Pygame giúp nhà phát triển tối ưu được toàn bộ tính năng trong thư viện Pygame và ứng dụng biến những ý tưởng trò chơi thành hiện thực.

2.3 Đa luồng(Multithreading)

Bạn dùng máy tính hàng ngày, mở hàng chục trang web khác nhau, cùng một cơ số đếm không xuể các ứng dụng nghe nhạc, xem phim, game ở ngoài, bạn có tự hỏi vì sao máy tính có thể cân hết chừng đấy việc một lúc không? Dường như các chương trình đều phản ứng chỉ trong tích tắc, và đang chạy đồng thời cùng nhau. Nhưng thực tế ảo diệu hơn thế nhiều, hóa ra, trong một đơn vị thời gian (nanosecond), chỉ có một chương trình (process) được chạy. Và trong chương trình đó, lại chia ra thành nhiều luồn (thread) con, thực thi cùng một lúc (multithread, ít nhất là trong thời điểm hiện tại), tạo cho người dùng cảm giác chương trình đang chạy nhanh hơn. Nhờ khả năng xử lí các task có thể coi như đồng thời (concurrency), chương trình có thể đáp ứng tốt với người dùng trong khi đang bận làm việc khác. Và đó là chính ý tưởng cơ bản của multithread.

2.3.1 Luồng (thread) là gì ? Sự khác nhau giữa thread và process

- Nói về cấu trúc máy tính : Thread là một đơn vị cơ bản trong CPU. Một luồng sẽ chia sẻ với các luồng khác trong cùng process về thông tin data, các dữ liệu của mình. Việc tạo ra thread giúp cho các chương trình có thể chạy được nhiều công việc cùng một lúc
- Process là quá trình hoạt động của một ứng dụng. Tiến trình (process)chứa đựng thông tin tài nguyên, trạng thái thực hiện của chương trình
- Thread là một bước điều hành bên trong một process. Luồng (thread) là một khối các câu lệnh (instructions) độc lập trong một tiến trình và có thể được lập lịch bởi hệ điều hành. Hay nói một cách đơn giản, Thread là các hàm hay thủ tục chạy độc lập đối với chương trình chính. Một process dĩ nhiên có thể chứa nhiều thread bên trong nó. Điểm quan trọng nhất cần chú ý là một thread có thể làm bất cứ nhiệm vụ gì một process có thể làm.
- Một điểm khác biệt nữa đó là nhiều thread nằm trong cùng một process dùng một không gian bộ nhớ giống nhau, trong khi process thì không. Điều này cho phép các thread đọc và viết cùng một kiểu cấu trúc và dữ liệu, giao tiếp dễ dàng giữa các thread với nhau. Giao thức giữa các process, hay còn gọi là IPC (inter-process communication) thì tương đối phức tạp bởi các dữ liệu có tính tập trung sâu hơn. Ngoài các tài nguyên riêng của mình (các biến cục bộ trong hàm), các luồng chia sẻ tài nguyên chung của tiến trình. Việc thay đổi tài nguyên chung (ví dụ, đóng file, gán giá trị mới cho biến) từ một thread sẽ được nhìn



Hình 1: Minh họa sự khác nhau giữa luồng và tiến trình

thấy bởi tất cả các thread khác. Vì vậy, lập trình viên cần phải thực hiện đồng bộ việc truy cập tài nguyên chung giữa các luồng.

- Đây là nhưng kiến thức trung xuất phát từ máy tính nói chung, đến các ngôn ngữ lập trình nói riêng thì những khái niệm này cũng tương tự như vậy

2.3.2 Đa luồng (Multithreading)

Một chương trình đa luồng chứa hai hoặc nhiều phần mà có thể chạy đồng thời và mỗi phần có thể xử lý tác vụ khác nhau tại cùng một thời điểm, để sử dụng tốt nhất các nguồn có sẵn, đặc biệt khi máy tính của bạn có nhiều CPU.

Python cung cấp thread Module và threading Module để bạn có thể bắt đầu một thread mới cũng như một số tác vụ khác trong khi lập trình đa luồng. Mỗi một Thread đều có vòng đời chung là bắt đầu, chạy và kết thúc. Một Thread có thể bị ngắt (interrupt), hoặc tạm thời bị dừng (sleeping) trong khi các Thread khác đang chạy được gọi là yielding.

Mặc dù thread Module rất hiệu quả với đa luồng tầm thấp nhưng khi so sánh với threading Module thì nó có nhiều điểm hạn chế. Phần tiếp theo giới thiệu về threading Module.

Module Threading cung cấp nhiều hỗ trợ mạnh mẽ và cấp độ cao hơn cho các Thread trong khi so sánh với thread Module ở trên. Ngoài các phương thức có trong thread Module, thì threading Module còn bổ sung thêm một số phương thức khác, đó là:

- threading.activeCount(): Trả về số đối tượng thread mà là active.
- threading.currentThread(): Trả về số đối tượng thread trong Thread control của Caller.
- threading.enumerate(): Trả về một danh sách tất cả đối tượng thread mà hiện tại là active.

Bên cạnh đó, threading Module có lớp Thread để triển khai đa luồng. Lớp này có các phương thức sau:

- run(): Là entry point cho một Thread.
- start(): Bắt đầu một thread bởi gọi phương thức run().
- join([time]): Đợi cho các thread kết thúc.
- isAlive(): Kiểm tra xem một thread có đang thực thi hay không.

- getName(): Trả về tên của một thread.
- setName(): Thiết lập tên của một thread.

2.3.3 Đồng bộ hóa các Thread trong Python

Trong lập trình đa luồng, các threads chia sẻ chung tài nguyên của tiến trình, vì vậy có những thời điểm nhiều luồng sẽ đồng thời thay đổi dữ liệu chung. Do đó, ta cần những cơ chế để đảm bảo rằng, tại một thời điểm chỉ có duy nhất một luồng được phép truy cập vào dữ liệu chung, nếu các luồng khác muốn truy cập vào đoạn dữ liệu này thì cần phải đợi cho thread trước đó hoàn thành công việc của mình.

Python cung cấp threading Module, mà bao gồm một kỹ thuật locking cho phép bạn đồng bộ hóa các Thread một cách dễ dàng. Một lock mới được tạo bởi gọi phương thức Lock().

Phương thức acquire(blocking) của đối tượng lock mới này được sử dụng để ép các Thread chạy một cách đồng bộ. Tham số blocking tùy ý cho bạn khả năng điều khiển để xem một Thread có cần đợi để đạt được lock hay không.

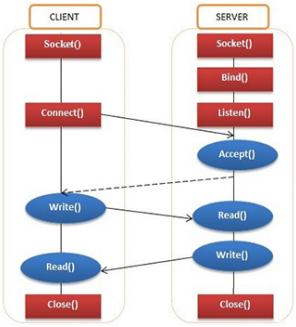
Nếu tham số blocking được thiết lập là 0, tức là Thread ngay lập tức trả về một giá trị 0 nếu không thu được lock và trả về giá trị 1 nếu thu được lock. Nếu blocking được thiết lập là 1, thì Thread cần đợi cho đến khi lock được giải phóng.

Phương thức release() của đối tượng lock được sử dụng để giải phóng lock khi nó không cần nữa.

2.4 Socket

Socket là giao diện lập trình ứng dụng mạng được 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 2 process trò chuyện với nhau. Điểm cuối (endpoint) của liên kết này được gọi là socket.

Một chức năng khác của socket là giúp các tầng TCP hoặc TCP Layer định danh ứng dụng mà dữ liệu sẽ được gửi tới thông qua sự ràng buộc với một cổng port (thể hiện là một con số cụ thể), từ đó tiến hành kết nối giữa client và server.



Hình 2: Minh họa cơ chế của socket

Mô tả mô hình

- Trước tiên chúng ta sẽ tạo ra một máy chủ bằng cách mở một socket - Socket()
- Sau đó chúng ta sẽ liên kết nó với một host hoặc một máy và một port - Bind()

- Tiếp theo server sẽ bắt đầu lắng nghe trên port đó - Listen()
- Yêu cầu kết nối từ client được gửi tới server - Connect()
- Server sẽ accept yêu cầu từ client và sau đó kết nối được thiết lập - Accept()
- Bây giờ cả hai đều có thể gửi và nhận tin tại thời điểm đó - Read() / Write()
- Và cuối cùng khi hoàn thành chúng có thể đóng kết nối - Close()

2.4.1 Tạo server

Chúng ta sẽ tạo một DCP IP server nhằm nhiệm vụ lắng nghe trên một cổng cho client gửi request đến.

Các bước để tạo lên 1 chương trình phía server:

- Tạo socket với hàm socket () chúng ta có thể truyền vào tham số hoặc để trống, ở đây mình 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,.. (ở đây mình dùng TCP)
- Gán địa chỉ cho socket bind(host, port) với các tham số là máy chủ và cổng
- Chỉ định socket lắng nghe kết nối listen () có thể điền số lượng tối đa các kết nối đang chờ
- Chờ/chấp nhận kết nốiaccept (). Hàm này trả về hai giá tric: kết nối; addr: địa chỉ của request hoặc địa chỉ của client
- Sau đó server có thể gửi dữ liệu qua hàm send().
- Sau khi thực hiện xong thì đóng kết nối lại close().

Ví dụ:

```
#server.py

import socket

host = 'localhost'
port = 4000
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((host, port))

s.listen(1)
print("Server listening on port", port)

c, addr = s.accept()
print("Connect from ", str(addr))

c.send(b"Hello, how are you")
c.send("Bye", encode())
c.close()
```

2.4.2 Tạo client

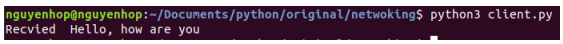
Chúng ta sẽ tạo một DCP IP client để kết nối với server ở cổng 4000.

- Tạo socket với hàm socket()
- Tạo kết nối với server với hàm connect(("localhost", 4000))
- Đọc sữ liệu được server gửi tới recv()
- Đóng socket close()

Ví dụ:

```
#client .py

import socket

s = socket.socket()
s.connect(("localhost", 4000))

msg = s.recv(1024)

while msg:
  print("Recvied ", msg.decode())
  msg = s.recv(1024)

s.close()
```

Sau khi chỉ chạy server.py rồi chạy client.py thì ta thấy client đã nhận được dữ liệu do server gửi tới.



Hình 3: Minh họa client đã nhận được dữ liệu do server gửi tới

3 Thiết kế ứng dụng

3.1 Mô tả thiết kế của chương trình

Chương trình này là một trò chơi caro trực tuyến đơn giản với hai phần chính: máy chủ (server) và máy khách (client), sử dụng socket để giao tiếp giữa các máy tính. Nó cũng sử dụng Pygame để tạo giao diện người dùng cho trò chơi. Khi người chơi mở ứng dụng, nếu chưa có người chơi nó sẽ đợi cho có người chơi khác vào máy chủ, nếu đã đủ hai người thì ván cờ sẽ bắt đầu. Ván cờ sẽ tiếp diễn cho đến khi có người ăn được hàng 5, có người đầu hàng và toàn bộ bàn cờ được lấp đầy(ván cờ sẽ hòa). khi kết thúc ván cờ, người chơi có thể yêu cầu tái đấu và bắt đầu lại ván cờ.

3.2 Cấu trúc mã nguồn

Mã nguồn của chương trình được chia thành các phần chính sau:

- Phần Máy Chủ (Server): Xử lý kết nối và quản lý trò chơi.
- Phần Máy Khách (Client): Kết nối với máy chủ, gửi và nhận dữ liệu, cho người dùng
- Giao diện người dùng (Main): Hiển thị giao diện người dùng, hiển thị bảng caro và các nút chức năng sử dụng Pygame.

Phần code R

Chi tiết mã nguồn máy chủ:

```
import socket
from _thread import *
import pickle
import time

server = "192.168.1.112"
port = 8080
current_player = 0

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

try:
    s.bind((server, port))
except socket.error as e:
    str(e)

s.listen(2)
print("waiting for a connection, Server Started")

# Thong tin bang va trang thai tro choi
client = [[1, 0], [2, 0]]
board = [[0]*11 for _ in range(11)]
game_ready = [0, 0]
game_state = "ready to play"
send_start = 0

```

```
def threaded_client(conn, player):
    global current_player, board, game_state, game_ready, send_start
    rematch_offer = False
    reply = ["Send client state", player, current_player]
    conn.send(pickle.dumps(reply))
    while True:
        try:
            reply = []
            data = conn.recv(4096)
            if not data:
                print("Disconnected")
                game_state = "ready to play"
                game_ready = [0, 0]
                client[player - 1][1] = 0
                current_player -= 1
                break

            data_receive = pickle.loads(data)
            if data_receive[0] == "waiting for opponent":
                reply.append("send client num")
                reply.append(current_player)
            elif data_receive[0] == "start game":
                reply.append("board data")
                reply.append(current_player)
                reply.append(board)
            elif data_receive[0] == "maintain connection":
                if current_player < 2:
                    board = [[0]*11 for _ in range(11)]
                    reply.append("player has disconnected")
                    reply.append(current_player)
                elif current_player >= 2 and game_state == "ready to play":
                    reply.append("board data")
                    reply.append(current_player)
                    reply.append(board)
                elif game_state != "ready to play" and game_ready != [1, 1] and not
                     rematch_offer:
                    reply.append("your opponent has resigned")
                    reply.append(game_ready)
                elif game_state != "ready to play" and game_ready == [1, 1]:
                    board = [[0]*11 for _ in range(11)]
                    reply.append("start again")
                    reply.append(-1)
                    time.sleep(0.1)
                    game_state = "ready to play"
                    rematch_offer = False
            elif data_receive[0] == "play pos data":
                board[data_receive[1][0]][data_receive[1][1]] = data_receive[1][2]
                reply.append("board data")
                reply.append(current_player)
                reply.append(board)
            elif data_receive[0] == "Resign":
                game_state = "not ready"
```

```
                game_ready = [0, 0]
                reply.append("you have resigned")
                reply.append(-1)
            elif data_receive[0] == "Rematch":
                rematch_offer = True
                game_ready[player - 1] = 1
                reply.append("you have offered a rematch")
                reply.append(-1)

            conn.sendall(pickle.dumps(reply))
        except:
            print("Connection has died")
            game_state = "ready to play"
            game_ready = [0, 0]
            client[player - 1][1] = 0
            current_player -= 1
            break

while True:
    conn, addr = s.accept()
    print("connected to:", addr)
    for i in range(2):
        if client[i][1] == 0:
            client[i][1] = 1
            start_new_thread(threaded_client, (conn, client[i][0]))
            current_player += 1
            break
```

Chi tiết mã nguồn máy khác:

```
import socket
import pickle

class Network:
    def __init__(self):
        self.client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.server = "192.168.1.112"
        self.port = 8080
        self.addr = (self.server, self.port)
        self.client_state = self.connect()

    def get_client_state(self):
        return self.client_state

    def connect(self):
        try:
            self.client.connect(self.addr)
            return pickle.loads(self.client.recv(4096))
        except:
            pass

```

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

    def receive(self):
        return pickle.loads(self.client.recv(4096))
```

Chi tiết mã nguồn giao diện:

```
import pygame
import sys
from network import Network

# Thiet lap man hinh
screen_width = 750
screen_height = 550
screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption("Caro")

# Kich thuoc bang va o
board_size = 11
tile_size = 50

# Tai va thay doi kich thuoc hinh anh
img_empty = pygame.image.load('Assets/caro_asset-01.png')
img_empty = pygame.transform.scale(img_empty, (tile_size, tile_size))
img_x = pygame.image.load('Assets/caro_asset-02.png')
img_x = pygame.transform.scale(img_x, (tile_size, tile_size))
img_o = pygame.image.load('Assets/caro_asset-03.png')
img_o = pygame.transform.scale(img_o, (tile_size, tile_size))
rect = img_empty.get_rect()

# Vi tri cua cac nut
retreat_rect = pygame.Rect(600, 140, 100, 50)
rematch_rect = pygame.Rect(600, 200, 100, 50)

# Khoi tao phong chu
pygame.font.init()
font_text_info = pygame.font.Font(None, 50)
font_state_info = pygame.font.Font(None, 30)
font_button_info = pygame.font.Font(None, 30)

def make_empty_board(sz):
    return [[0]*sz for _ in range(sz)]

def draw_screen(img_empty, img_x, img_o, rect, board):
    for x in range(11):
        rect.center = (50*x + 25, -25)
        for y in range(11):
```

```
            rect.centery += 50
            if board[x][y] == 0:
                screen.blit(img_empty, rect)
            elif board[x][y] == 1:
                screen.blit(img_x, rect)
            else:
                screen.blit(img_o, rect)

def draw_info_panel(screen, play_as, state):
    panel_width = screen_width - (board_size * tile_size)
    panel_color = (255, 155, 155, 150)
    info_panel = pygame.Surface((panel_width, screen_height), pygame.SRCALPHA)
    info_panel.fill(panel_color)

    font_color = (255, 255, 255)
    text_info = font_text_info.render("Player " + str(play_as), True, font_color)
    text_rect = text_info.get_rect()
    text_rect.center = (screen_width - 150, 50)
    info_panel.blit(text_info, text_rect)

    if state == 0:
        text_state = font_state_info.render("Your Turn", True, font_color)
    elif state == 1:
        text_state = font_state_info.render("Wait", True, font_color)
    else:
        text_state = font_state_info.render("Invalid", True, font_color)

    state_rect = text_state.get_rect()
    state_rect.center = (screen_width - 150, 100)
    info_panel.blit(text_state, state_rect)

    draw_button(info_panel, retreat_rect, "Resign")
    draw_button(info_panel, rematch_rect, "Rematch")

    screen.blit(info_panel, (screen_width - panel_width, 0))

def draw_button(panel, rect, caption):
    font_color = (255, 255, 255)
    pygame.draw.rect(panel, (255, 0, 0), rect)
    text_button = font_button_info.render(caption, True, font_color)
    text_rect = text_button.get_rect()
    text_rect.center = rect.center
    panel.blit(text_button, text_rect)

def get_board_pos(pos):
    return (pos[0]//50, pos[1]//50)

def main():
    clock = pygame.time.Clock()
    network = Network()
    client_state = network.get_client_state()
    player = client_state[1]
```

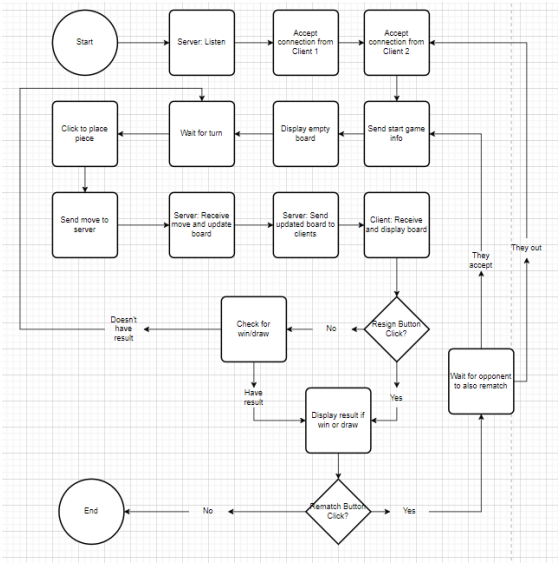
```
    turn = client_state[2]
    board = make_empty_board(board_size)

    if turn == player:
        state = 0
    else:
        state = 1

    while True:
        screen.fill((0, 0, 0))
        draw_screen(img_empty, img_x, img_o, rect, board)
        draw_info_panel(screen, player, state)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
                pos = pygame.mouse.get_pos()
                board_pos = get_board_pos(pos)
                if pos[0] < 550 and turn == player:
                    if board[board_pos[0]][board_pos[1]] == 0:
                        if player == 1:
                            board[board_pos[0]][board_pos[1]] = 1
                        else:
                            board[board_pos[0]][board_pos[1]] = 2
                        network.send(["play pos data", (board_pos[0], board_pos[1],
                            player)])
                        state = 1
                        turn = 3 - player
                elif retreat_rect.collidepoint(pos):
                    network.send(["Resign", player])
                elif rematch_rect.collidepoint(pos):
                    network.send(["Rematch", player])

        reply = network.receive()
        if reply[0] == "board data":
            turn = reply[1]
            board = reply[2]
            if turn == player:
                state = 0
            else:
                state = 1
        elif reply[0] == "start again":
            turn = player
            board = make_empty_board(board_size)
            state = 0
        elif reply[0] == "player has disconnected":
            turn = player
            board = make_empty_board(board_size)
            state = 2

```

```
144         pygame.display.flip()
145         clock.tick(60)
146 
147 if __name__ == "__main__":
148     main()
```

3.3 Mô hình ứng dụng

Ứng dụng này sử dụng mô hình Client-Server, trong đó:

- Máy chủ (Server):
  - Đóng vai trò trung tâm để quản lý trạng thái trò chơi và giao tiếp giữa các máy khách.
  - Máy chủ chờ kết nối từ các máy khách, sau đó xử lý các yêu cầu từ mỗi máy khách trong các luồng riêng biệt.
- Máy khách (Client):
  - Kết nối đến máy chủ để tham gia trò chơi.
  - Gửi các hành động của người chơi lên máy chủ và nhận lại trạng thái trò chơi cập nhật.
  - Hiển thị giao diện trò chơi cho người chơi và xử lý các tương tác người dùng.
- Kết nối mạng:
  - Máy khách kết nối với máy chủ thông qua địa chỉ IP và cổng được chỉ định.
  - Máy chủ chấp nhận kết nối từ hai máy khách.
- Quản lý trò chơi:
  - Máy chủ duy trì trạng thái trò chơi, bao gồm thông tin người chơi và bảng caro.
  - Quản lý lượt chơi của người chơi.
  - Xử lý yêu cầu bắt đầu trò chơi, giữ kết nối, chơi lại, và đầu hàng từ máy khách.
- Giao diện người dùng:
  - Sử dụng Pygame để hiển thị bảng caro và các nút chức năng (Đầu hàng và Chơi lại).
  - Hiển thị trạng thái trò chơi và thông báo cho người chơi về lượt đi của mình hoặc đối thủ.
  - Vẽ các quân cờ X và O lên bảng caro dựa trên hành động của người chơi.



Hình 4: Sơ đồ flowchart của chương trình TicTacToe

3.4 Flowchart

Dưới đây là quy trình hoạt động (flowchart) của ứng dụng Caro trực tuyến dựa trên mã nguồn đã cung cấp. Quy trình này bao gồm sự tương tác giữa máy chủ (server), máy khách (client) và giao diện người dùng (UI) được xây dựng bằng Pygame.

- **Khởi động máy chủ (Server)**
  - **Máy chủ được khởi động, lắng nghe các kết nối từ máy khách trên địa chỉ IP và cổng được chỉ định.**
  - **Máy chủ chấp nhận kết nối từ hai máy khách. Mỗi kết nối từ máy khách được xử lý trong một luồng riêng biệt (threaded-client).**
- **Kết nối từ máy khách (Client)**
  - **Máy khách khởi động và kết nối với máy chủ thông qua địa chỉ IP và cổng của máy chủ.**
  - **Sau khi kết nối thành công, máy khách nhận thông tin về trạng thái của mình và lượt chơi từ máy chủ.**
- **Bắt đầu trò chơi**
  - **Khi hai máy khách đã kết nối, máy chủ gửi thông tin bắt đầu trò chơi và bảng caro ban đầu đến cả hai máy khách.**

  - Giao diện người dùng hiển thị bảng caro trống và các thông tin liên quan (như lượt chơi) cho người chơi.

- Vòng lặp trò chơi
  - Mỗi máy khách chờ lượt chơi của mình. Khi đến lượt, người chơi sẽ nhấp chuột để đặt quân cờ (X hoặc O) lên bảng caro.
  - Máy khách gửi dữ liệu về vị trí và loại quân cờ lên máy chủ.
  - Máy chủ nhận dữ liệu, cập nhật trạng thái bảng caro và gửi lại thông tin bảng caro mới cho cả hai máy khách.
  - Máy khách nhận dữ liệu cập nhật từ máy chủ và hiển thị lại bảng caro mới. Vòng lặp tiếp tục cho đến khi trò chơi kết thúc hoặc có yêu cầu từ người chơi.
- Đầu hàng (Resign)
  - Khi người chơi nhấp vào nút "Resign", máy khách gửi yêu cầu đầu hàng lên máy chủ.
  - Máy chủ nhận yêu cầu, cập nhật trạng thái trò chơi và gửi thông báo đầu hàng đến cả hai máy khách.
  - Máy khách hiển thị thông báo và kết thúc trò chơi.
- Chơi lại (Rematch)
  - Khi người chơi nhấp vào nút "Rematch", máy khách gửi yêu cầu chơi lại lên máy chủ.
  - Máy chủ nhận yêu cầu, nếu cả hai người chơi đều đồng ý, máy chủ khởi tạo lại bảng caro và gửi thông tin bắt đầu trò chơi mới cho cả hai máy khách.
  - Máy khách nhận thông tin và hiển thị lại bảng caro trống, bắt đầu một vòng lặp trò chơi mới.

4 Hiện thực ứng dụng

Trong trò chơi ca-rô này sẽ bao gồm 2 chế độ chơi chính bao gồm:

- Chế độ offline: Chế độ này cho phép hai người chơi sẽ chơi trên cùng một máy, và lần lược mỗi người sẽ đánh những quân cờ của mình. Sau khi có người chiến thắng, trò chơi sẽ đưa ra thông báo lên màn hình. Người chơi có thể thực hiện làm mới lại bàn cờ và thực hiện một ván đấu mới
- Chế độ online: Chế độ này cho phép hai người chơi sẽ chơi trên hai thiết bị khác nhau có kết nói cùng một mạng với nhau. Tương tự như chế độ offline, hai người chơi lần lược đặt các quân cờ của mình. Người chơi có thể thực hiện đầu hàng hoặc đấu lại khi trò chơi kết thúc. Hệ thống sẽ đưa ra thông báo khi màn chơi hoàn thành hoặc có người thoát kết nối

4.1 Giao diện Menu

Trong gao diện menu, hệ thông sẽ có hai phím bấm cho người chơi bao gồm:

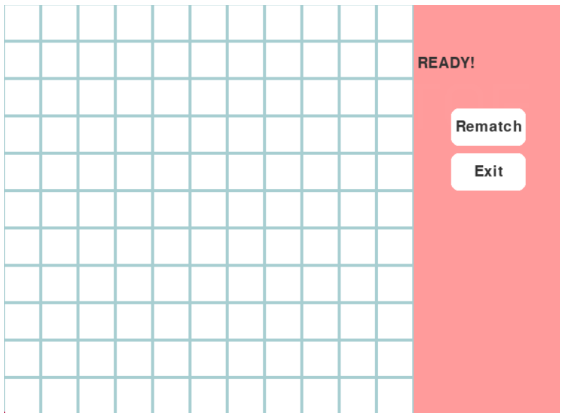
- Nút Online: Khi bấm vào nút này hệ thống sẽ đưa người chơi vào chế độ chơi Online.
- Chế độ offline: Hệ thống đưa người chơi vào chế độ offline

Hình 5: Màn hình menu của trò chơi

4.2 Chế độ Offline

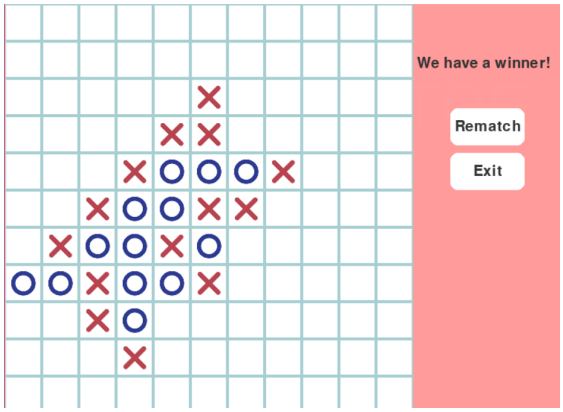
Sau khi ấn vào phím Offline trong màn hình chính. Người chơi sẽ được đưa đến giao diện của chế độ Offline. Trong đây bao gồm các thành phần:

- **Bàn cờ: Nơi cả hai người chơi sẽ đặt các quân cờ xuống**
- **Nút Rematch: Làm mới lại bàn cờ để chơi lại một màn chơi mới**
- **Nút Exit: Trở về màn hình chính**



Hình 6: Màn hình màn chơi offline

Sau khi hoàn thành một màn chơi, hệ thống sẽ thông báo đã có người chiến thắng để người chơi biết được. Đồng thời hệ thống sẽ không cho phép đặt bất kỳ quân cờ nào.



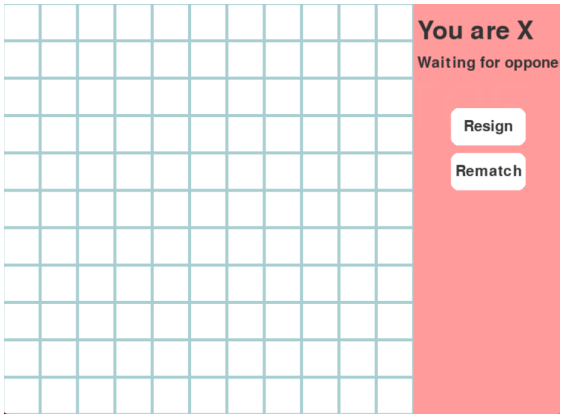
Hình 7: Màn hình khi hoàn thành trò chơi trong chế độ offline

4.3 Chế độ Online

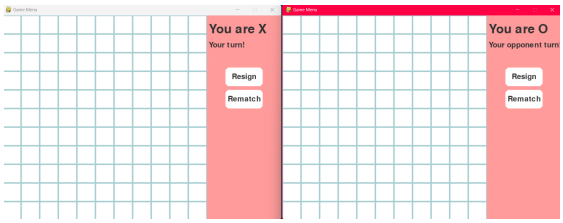
Sau khi ấn vào phím Online trong màn hình chính, hệ thống sẽ đưa người chơi đến giao diện của chế độ Online. Giao diện này bao gồm:

- Bàn cờ: Nơi cả hai người chơi sẽ đặt các quân cờ xuống
- Nút Rematch: Gửi yêu cầu đến người chơi thứ hai để xác nhận đấu lại. Nếu người chơi thứ hai cũng bấm thì sẽ làm mới lại màn chơi.
- Nút Resign: Dùng để kết thúc sớm màn chơi.

Sau khi vào giao diện chính, hệ thống sẽ đưa ra thông báo đợi đối thủ kết nói và vô hiệu hóa bàn cờ. Nếu có người chơi thứ hai thì sẽ bắt đầu trò chơi.

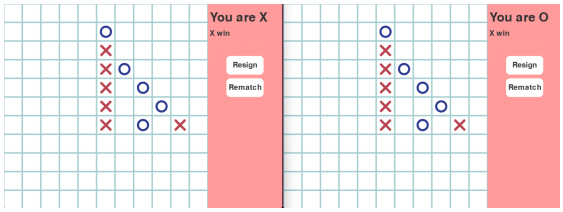


Hình 8: Màn hình chế độ Online



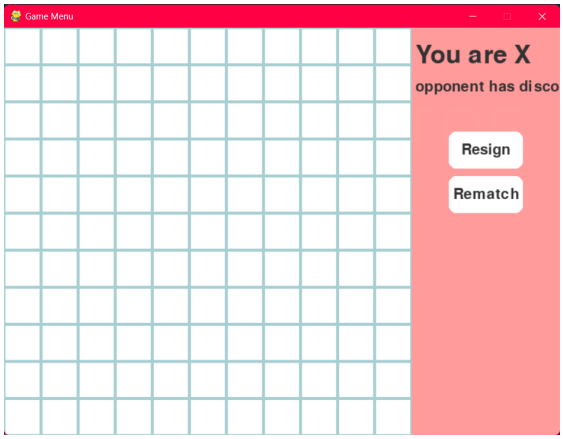
Hình 9: Hai người chơi đã kết nối

Sau khi kết nối thành công, hai người chơi sẽ lần lược thả quân cờ cho đến khi có người chơi chiến thắng. Khi có người chơi chiến thắng thì sẽ dừng game lại và thông báo cho người chơi biết người chiến thắng.



Hình 10: Giao diện sau khi hoàn thành trò chơi

Trong quá trình chơi, nếu có một người thoát ra hoặc mất kết nối đến máy chủ. Hệ thống sẽ thông báo đến người chơi và cho dừng màn chơi ngay lập tức. Nếu có người chơi khác kết nối thì màn chơi sẽ được bắt đầu lại.



Hình 11: Minh họa cơ chế của socket

5 Cách thức cài đặt ứng dụng

5.1 Cài đặt python

Trước tiên, chúng ta cần thực hiện cài đặt python cho linux bằng các câu lệnh như sau:

```
sudo apt update
sudo apt install python3
```

Sau khi cài đặt xong, ta sẽ kiểm tra lại phiên bản python thông qua lệnh sau:

```
python3 --version
```

5.2 Cài đặt pip và pygame

Ta cần cài đặt trình quản lý gói của Python (pip). Để cài đặt, ta sữ dụng câu lệnh sau trên màn hình Terminal để tiến hành cài đặt pip.

```
sudo apt install python3-pip
pip3 --version
```

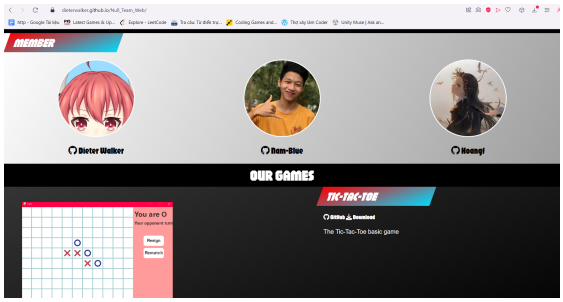
Tiếp theo, ta cần cài đặt thư viện pygame của python để có thể chạy các tệp tin trong trò chơi. Cách để cài đặt thư viện Pygame thì ta sẽ sử dụng câu lệnh sau trong màn hình Terminal.

```
sudo apt install python3-pygame
```

5.3 Cài đặt Game

WEBSITE:
Ta có thể truy cập vào trang web sau để thiến hành tải trò chơi về:

```
https://dieterwalker.github.io/Null_Team_Web/
```



Hình 12: Màn hình website tải game

GIT:

Trước tiên, ta cần cài đặt git thông qua các câu lệnh sau đây trên Terminal:

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

Sau khi cài đặt thành công, chúng ta thực hiện clone thư mục của trò chơi trên github về:

```
git clone https://github.com/nam050402/MNM.git
```

5.4 Chạy File Game

Sau cùng, ta thực hiện chạy chương trình và server thông qua câu lệnh sau:

```
gedit TicTacToe.py
```

Thực hiện chạy server:

```
gedit server.py
```

6 Nhiệm vụ và vai trò của từng thành viên

Trần Giang Nam

- **Thực hiện tính năng online cho trò chơi**
- **Triển khai socket**
- **Viết báo cáo latex**

Nguyễn Quốc Duệ

- **Thực hiện chức năng offline cho trò chơi**
- **Thực hiện tạo webside để tải và cài đặt trò chơi**
- **Thực hiện giao diện trong trò chơi**

Nguyễn Việt Hoàng

- **Viết báo cáo latex**
- **Thực hiện giao diện cho trò chơi**

Tài liệu

[1] **CVX Introduction "link: http://cvxr.com/cvx/doc/intro.html/", *What is CVX*, lần truy cập cuối: 15/04/2017.**