BỘ THÔNG TIN VÀ TRUYỀN THÔNG

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

VIỆN KHOA HỌC KỸ THUẬT BƯU ĐIỆN

------□------

BÁO CÁO

ĐỀ TÀI THI CUỐI KỲ

Đề tài :Dự án sản phẩm game

Giảng viên: Phan Lý Huỳnh

Tên nhóm : Lttile Stork

Lớp : D23CQCC01- B

Sinh viên thực hiện:

1. Nguyễn Ngọc Duy - B23DCCC049

2. Nguyễn Tiến Tuấn - B23DCCC173

3. Nguyễn Thị Hà - B23DCCC057

4. Trần Tuấn Minh - B23DCCC113

5. Hoàng Thị Hòa - B23DCCC069

Hà Nội, ngày 01/01/2024

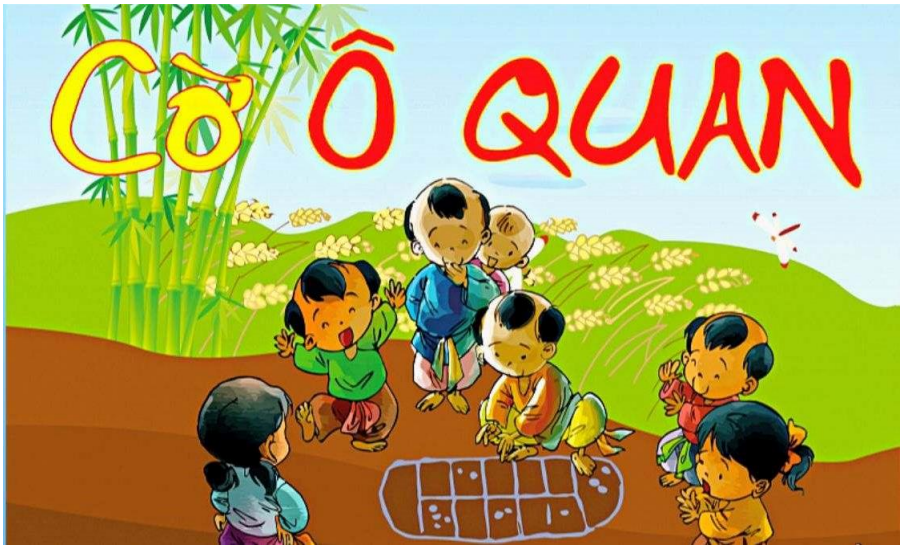
Mục lục

Bảng phần chia công việc và đánh giá thành viên

Công việc thực hiện	Họ và tên	Mức độ hoàn thành (thang điểm 10)	NHẬN XÉT
1. Code thuật toán.	1. Nguyễn Tiến Tuấn 2. Nguyễn Thị Hà 3. Trần Tuấn Minh 4. Hoàng Thị Hòa 5. Nguyễn Ngọc Duy	**10.0** **8.5** **9.5** **9.5**	
2. Code giao diện.	1. Nguyễn Tiến Tuấn 2. Nguyễn Thị Hà 3. Trần Tuấn Minh 4. Hoàng Thị Hòa 5. Nguyễn Ngọc Duy	**10.0** **10.0** **9.5** **8**	
3. Slide	1. Nguyễn Tiến Tuấn 2. Nguyễn Thị Hà 3. Trần Tuấn Minh 4. Hoàng Thị Hòa 5. Nguyễn Ngọc Duy	**?** **?** **?** **9.0**	
4. Báo cáo	1. Nguyễn Tiến Tuấn 2. Nguyễn Thị Hà 3. Trần Tuấn Minh 4. Hoàng Thị Hòa 5. Nguyễn Ngọc Duy	**9.5** **10.0** **9.5**	
5. Đồ họa	1. Nguyễn Tiến Tuấn 2. Nguyễn Thị Hà 3. Trần Tuấn Minh 4. Hoàng Thị Hòa 5. Nguyễn Ngọc Duy	**10.0**	

Phần 1: Giới thiệu bài toán

I. NGUỒN GỐC TRÒ CHƠI



Ô ăn quan, hay còn gọi tắt là ăn quan hoặc ô quan có nguồn gốc từ miền lục địa châu Phi, đặc biệt là vùng đông và nam châu Phi. Tại đây, trò chơi được gọi là Awale, có nghĩa là "túi hạt". Từ Awale, tên gọi ô ăn quan đã được hình thành. Đây là trò chơi có tính chất chiến thuật thường dành cho hai hoặc ba người chơi và có thể sử dụng.

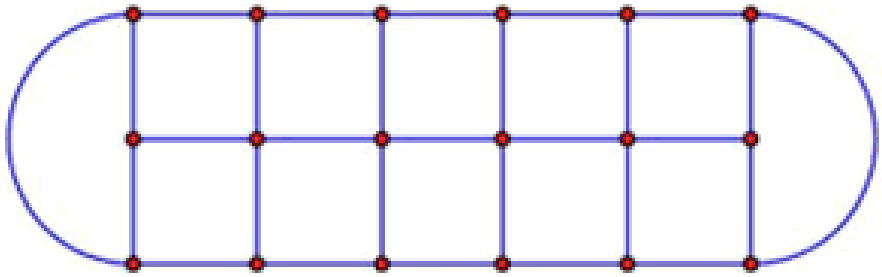
Về việc trò chơi ô ăn quan xuất hiện tại Việt Nam từ thời điểm nào và nguồn gốc ra sao vẫn chưa được rõ ràng. Tuy nhiên, có thể nhận thấy rằng trò chơi này đã có mặt trên đất nước từ rất lâu đời, Có thể ô ăn quan được lấy cảm hứng từ cánh đồng lúa và khát vọng của người nông dân mong muốn sự sung túc, giàu có trong cuộc sống. Những câu truyện lưu truyền về Mạc Hiển Tích (chưa rõ năm sinh, năm mất), đỗ Trạng nguyên năm 1086 nói rằng ông đã có một tác phẩm bàn về các phép tính trong trò chơi Ô ăn quan và đề cập đến số ẩn (số âm) của ô trống xuất hiện trong khi chơi. Ô ăn quan đã từng phổ biến ở khắp ba miền Bắc, Trung, Nam của Việt Nam nhưng những năm gần đây chỉ còn được rất ít trẻ em chơi. Bảo tàng Dân tộc học Việt Nam có trưng bày, giới thiệu và hướng dẫn trò chơi này.

Theo các nhà nghiên cứu, ô ăn quan thuộc họ trò chơi mancala, tiếng Ả Rập là manqala hoặc minqala (khi phát âm, trọng âm rơi vào âm tiết đầu ở Syria và âm tiết thứ hai ở (Ai Cập) có nguồn gốc từ động từ naqala có nghĩa là di chuyển. Bàn chơi mancala đã hiện diện ở Ai Cập từ thời kỳ Đế chế (khoảng 1580 - 1150 TCN). Tuy nhiên còn một khoảng trống giữa lần xuất hiện này với sự tồn tại của mancala ở

Ceylon(Srilanka) những năm đầu Công nguyên và ở Ả Rập trước thời Muhammad.(2)

1.2. MÔ TẢ TRÒ CHƠI

Chuẩn bị:



Bàn chơi: bàn chơi Ô ăn quan kẻ trên một mặt bằng tương đối phẳng có kích thước linh hoạt miễn là có thể chia ra đủ số ô cần thiết để chứa quân đồng thời không quá lớn để thuận tiện cho việc di chuyển quân, vì thế có thể được tạo ra trên nền đất, vỉa hè, trên miếng gỗ phẳng .... Bàn chơi được kẻ thành một hình chữ nhật rồi chia hình chữ nhật đó thành mười ô vuông, mỗi bên có năm ô đối xứng nhau. Ở hai cạnh ngắn hơn của hình chữ nhật, kẻ hai ô hình bán nguyệt hoặc hình vòng cung hướng ra phía ngoài. Các ô hình vuông gọi là ô dâcòn hai ô hình bán nguyệt hoặc vòng cung gọi là ô quan.

Quân chơi: gồm hai loại quan và dân, được làm hoặc thu thập từ nhiều chất liệu có hình thể ổn định, kích thước vừa phải để người chơi có thể cầm, nắm nhiều quân bằng một bàn tay khi chơi. Quan có kích thước lớn hơn dân đáng kể cho dễ phân biệt với nhau. Quân chơi có thể là những viên sỏi, gạch, đá, hạt của một số loại quả... hoặc được sản xuất công nghiệp từ vật liệu cứng mà phổ biến là nhựa. Số lượng quan luôn là 2 còn dân có số lượng tùy theo luật chơi nhưng phổ biến nhất là 50.

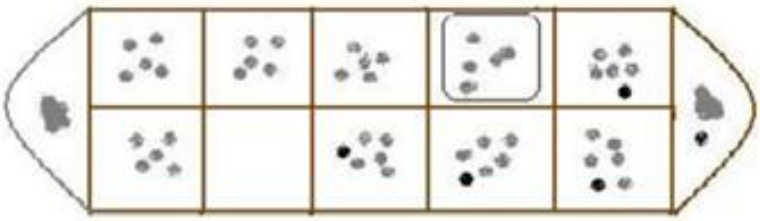
Bố trí quân chơi: quan được đặt trong hai ô hình bán nguyệt hoặc cánh cung, mỗi ô một quân, dân được bố trí vào các ô vuông với số quân đều nhau, mỗi ô 5 dân. Trường hợp không muốn hoặc không thể tìm kiếm được quan phù hợp thì có thể thay quan bằng cách đặt số lượng dân quy đổi vào ô quan. Người chơi: thường gồm hai

người chơi, mỗi người ở phía ngoài cạnh dài hơn của hình chữ nhật và những ô vuông bên nào thuộc quyền kiểm soát của người chơi bên đó.

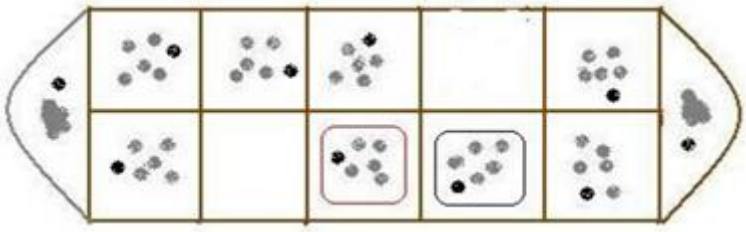
Luật chơi: Người thắng cuộc trong trò chơi này là người mà khi cuộc chơi kết thúc có tổng số dân nhiều hơn. Tùy theo luật chơi từng địa phương hoặc thỏa thuận giữa hai người chơi nhưng phổ biến là 1 quan được quy đổi bằng 10 dân hoặc 5 dân.

Di chuyển quân: từng người chơi khi đến lượt của mình sẽ di chuyển dân theo phương án để có thể ăn được càng nhiều dân và quan hơn đối phương càng tốt. Người thực hiện lượt đi đầu tiên thường được xác định bằng cách oẳn tù tì hay thỏa thuận.

Khi đến lượt, người chơi sẽ dùng tất cả số quân trong một ô có quân bất kỳ do người đó chọn trong số 5 ô vuông thuộc quyền kiểm soát của mình để lần lượt rải vào các ô, mỗi ô 1 quân, bắt đầu từ ô gần nhất và có thể rải ngược hay xuôi chiều kim đồng hồ tùy ý. Khi rải hết quân cuối cùng, tùy tình huống mà người chơi sẽ phải xử lý tiếp như sau:



Nếu liền sau đó là một ô vuông có chứa quân thì tiếp tục dùng tất cả số quân đó để rải tiếp theo chiều đã chọn.



Nếu liền sau đó là một ô trống (không phân biệt ô quan hay ô dân) rồi đến một ô có chứa quân thì người chơi sẽ được ăn tất cả số quân trong ô đó. Số quân bị ăn sẽ được loại ra khỏi bàn chơi để người chơi tính điểm khi kết thúc.

Nếu liền sau ô có quân đã bị ăn lại là một ô trống rồi đến một ô có quân nữa thì người chơi có quyền ăn tiếp cả quân ở ô này... Do đó trong cuộc chơi có thể có phương án rải quân làm cho người chơi ăn hết toàn bộ số quân trên bàn chơi chỉ trong một lượt đi của mình. Một ô có nhiều dân thường được trẻ em gọi là ô nhà giàu, rất nhiều dân thì gọi là giàu sụ. Người chơi có thể bằng kinh nghiệm hoặc tính toán phương án nhằm nuôi ô nhà giàu rồi mới ăn để được nhiều điểm.

Nếu liền sau đó là ô quan có chứa quân hoặc 2 ô trống trở lên hoặc sau khi vừa ăn thì người chơi bị mất lượt và quyền đi tiếp thuộc về đối phương. Trường hợp đến lượt đi nhưng cả 5 ô vuông thuộc quyền kiểm soát của người chơi đều không có dân thì người đó sẽ phải dùng 5 dân đã ăn được của mình để đặt vào mỗi ô 1 dân để có thể thực hiện việc di chuyển quân.

Nếu người chơi không đủ 5 dân thì phải vay của đối phương và trả lại khi tính điểm.

Cuộc chơi sẽ kết thúc khi toàn bộ dân và quan ở hai ô quan đã bị ăn hết. Trường hợp hai ô quan đã bị ăn hết nhưng vẫn còn dân thì quân trong những hình vuông phía bên nào coi như thuộc về người chơi bên ấy; tình huống này được gọi là hết quan, tàn dân, thu quân, kéo về hay hết quan, tàn dân, thu quân, bán ruộng. Ô quan có ít dân (có số dân nhỏ hơn 5 phổ biến được coi là ít) gọi là quan non và để cuộc chơi không bị kết thúc sớm cho tăng phần thú vị, luật chơi có thể quy định không được ăn quan non, nếu rơi vào tình huống đó sẽ bị mất lượt.

Phần 2: Cơ sở lý thuyết

I. Đặt vấn đề:

Trong cờ Ô ăn quan, mỗi cách bố trí quân cờ trên bàn cờ là một vị trí. Vị trí ban đầu là sự sắp xếp các quân cờ lúc đầu cuộc chơi. Mỗi bước đi hợp lệ là một phép biến đổi vị trí, nó biến đổi vị trí trên bàn cờ thành một vị trí khác.

Như vậy, ta xác định được các yếu tố:

- Trạng thái bắt đầu (TTBĐ): là trạng thái trước khi một sự kiện trên bàn cờ xảy ra.

- Trạng thái kết thúc (TTKT): là trạng thái khi sự kiện trên bàn cờ gặp điều kiện dừng và kết thúc thì các trạng thái đã được biến đổi và được di chuyển xung quanh bàn cờ theo nguyên tắc (luật chơi) tạo thành 1 không gian bao gồm các sự kiện.

- Sự kiện: là các bước bao gồm di chuyển dân, ăn quân, ăn dân, gặp điều kiện dừng di chuyển dân,…

Đặc điểm của các trò chơi thể loại này:

- Có hai đấu thủ, mỗi người chỉ được đi một nước khi tới lượt (lượt đấu xoay vòng).

- Các đấu thủ đều biết mọi thông tin về tình trạng trận đấu.

- Trận đấu không kéo dài vô tận, phải diễn ra hòa hoặc một bên thắng hoặc thua.

II. Phương án.

Việc cần thực hiện là tìm kiếm cách để quy các sự kiện, trạng thái có thể diễn ra thành 1 giới hạn nhất định trong từng vị trí ô bàn cờ đi từ đầu đến đích. Khi ta cần biết đây là sự kiện, trạng thái nào thì nó đã được xác định tại vị trí ô đó.

Như chúng ta đã biết, trò chơi Ô ăn quan cũng là một trò chơi đối kháng. Cụ thể trò chơi này thuộc dạng trò chơi có điểm tổng ban đầu bằng 0, gồm có 2 người chơi. Người chơi phải cần tính toán để có thể ăn được nhiều quân hay quan nhất; tích được nhiều điểm nhất.

Không thể có trường hợp cả hai bên đều thắng hoặc đều thua. Nếu một bên thắng thì bên kia nhất định thua và ngược lại.

Nếu viết chương trình để người chơi với người thì cần xây dựng các tập luật chơi để người chơi không phạm quy và kết thúc khi gặp điều kiện dùng trò chơi.

III. Môi trường và công cụ.

1. Ngôn ngữ và công cụ lập trình:

- Để xây dựng game Ô ăn quan, nhóm sử dụng công cụ lập trình:

+ Ngôn ngữ Python.

Vì sao chọn dùng ngôn ngữ Python

a. Định nghĩa:

- Python là một ngôn ngữ lập trình bậc cao cho các mục đích lập trình đa năng.
- Python được thiết kế với ưu điểm mạnh là dễ đọc, dễ học và dễ nhớ.
- Python là ngôn ngữ có hình thức rất sáng sủa, cấu trúc rõ ràng, thuận tiện cho người mới học lập trình và là ngôn ngữ lập trình dễ học; được dùng rộng rãi trong phát triển trí tuệ nhân tạo.

b. Ý nghĩa:

- Python được thiết kế để có cú pháp đơn giản và dễ đọc, giúp người lập trình tập trung vào logic của mã nguồn hơn là cú pháp phức tạp.
- Python là ngôn ngữ đa mục đích, có thể được sử dụng như phát triển web, trí tuệ nhân tạo, khoa học dữ liệu, máy móc, và nhiều lĩnh vực khác.
- Python thường được chọn làm ngôn ngữ đầu tiên cho người mới học lập trình vì tính dễ học và đơn giản
- Python chạy được nhiều ngôn ngữ khác nhau như Windows, maOS và Linux

Các thư viện sử dụng:

- pygame: là 1 thư viện chuyên dụng cho phát triển trò chơi và ứng dụng đồ họa sử dụng ngôn ngữ lập trình Python. Nó cung cấp các chức năng cho việc xử lý đồ họa, sự kiện, âm thanh,... để tạo ra trải nghiệm đồ họa đa dạng.

- Sys: là 1 module thuộc thư viện chuẩn của python nó cung cấp các chức năng và biến liên quan đến hệ thống. Trong khi thực hiện chơi game " Ô Ăn Quan" ta sử dụng nó để thoát khỏi trò chơi (chương trình) khi người chơi đóng của sổ hoặc khi kết thúc ứng dụng.

- Radom: là thư viện cung cấp các công cụ để thực hiện các phương pháp ngẫu nhiên như chọn 1 phần tử ngẫu nhiên từ 1 danh sách,..... các hoạt động liên quan đến ngẫu nhiên.

Công cụ:

- Visual Studio, pycharm để thiết kế giao diện**,** thuật toán.
- Adobe Photoshop để vẽ hình ảnh, thiết kế đồ họa trò chơi.
- Các công cụ phục vụ nghiên cứu.
- …

Phần 3: Giải thuật

A. Thuật toán.

I. Phương án giải quyết

1. Ý tưởng.

Xây dựng các thành hàm phần: Xét nước đi hợp lệ hay không, xét đến lượt của ai đi, xét thắng thua, hàm di chuyển, xét trạng thái kết thúc game, thay đổi các hiển thị quân trong ô cờ,...

Ví dụ:

- Ta dùng hàm <*tạo bảng*> để khởi tạo ra không gian trò chơi (các ô trên bàn cờ) của trò chơi, bao gồm bảng, người chơi hiện tại, và cách tính điểm của hai người chơi.

- Ta tiếp tục dùng hàm <*chơi game*> để chơi game và dùng để lặp lại các bước cho đến khi trò chơi kết thúc.

- Quan trọng nhất là sử dụng hàm <*phân phối*> để thực hiện phân phối đá về phía bên phải hay bên trái dựa trên lựa chọn chiều của người chơi.

- Sau đó ta kiểm tra điều kiện kết thúc và cập nhập trạng thái của game, dùng hàm <*rải thêm*> để kiểm tra điều kiện kết thúc trò chơi và cập nhập điểm cho người chơi.

…

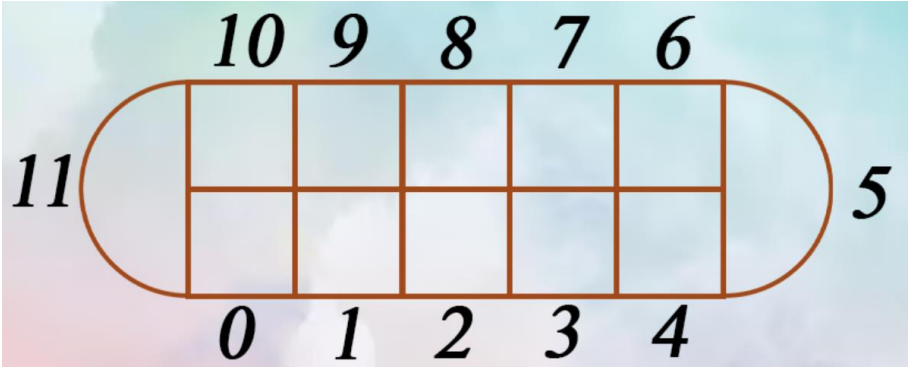
- Cuối cùng là lặp lại quá trình và chuyển đổi lượt chơi giữa hai người chơi.

Chú thích: kí hiệu cho một hàm <*tên hàm*>.

II. Áp dụng cơ sở lý thuyết

Như đã đề cập ở phần 2, về phương án giải quyết để quy các sự kiện, trạng thái có thể diễn ra thành 1 giới hạn nhất định trong từng vị trí ô bàn cờ đi từ đầu đến đích. Khi ta cần biết đây là sự kiện, trạng thái nào thì nó đã được xác định tại vị trí ô đó.

Trước đó cần tạo một không gian bàn cờ để mọi thứ diễn ra trên đó. Để xác định được các ô và xử lý các dữ liệu (đặc điểm) của ô bao gồm số quân, vị trí của ô,… ta gán vị trí (tọa độ) của ô trong không gian bàn cờ:



- Các ô trên bàn cờ bao gồm 12 ô được gán tọa độ là các số từ 0 – 11.
- Ta sẽ xem đây là vị trí các ô và sẽ thực hiện các phép biến đổi của sự kiện dựa trên việc đánh giá, kiểm tra hay tính toán trên tọa độ này.

Trong trạng thái bắt đầu (trước khi sự kiện diễn ra), số quân và dân trên bàn cờ đang ở yên vị trí, số lượng và không có tương tác hay di chuyển.

Khi sự kiện diễn ra sẽ bao gồm loạt các sự kiện và phải kiểm tra điều kiện để diễn ra nó:

- Sự kiện di chuyển dân, lần lượt trên bàn cờ theo chiều ngược hay xuôi chiều kim đồng hồ đều tuân thủ quy tắc di chuyển 1 quân vào ô kế tiếp và tiếp tục cho đến khi hết dân sau đó sẽ kiểm tra điều kiện để tiếp tục hoặc dừng lại.

- Sự kiện tiếp tục di chuyển dân được diễn ra khi kiểm tra nếu như khi di chuyển dân trước đó được dừng lại và số dân của ô tiếp theo lớn hơn 0.

- Sự kiện ăn dân, quan được diễn ra khi kiểm tra nếu như số dân ô tiếp theo của ô cuối cùng được di chuyển dân đến đó bằng 0 mà ô tiếp theo nữa có số dân > 0 thì sẽ được dân ở ô này và thực kiển kiểm tra ăn dân vẫn được tiếp tục diễn ra tương tự cho đến khi gặp hai ô trống hoặc hai ô có dân liền kề.

=> Số dân, quan ăn được sẽ được quy điểm và cộng vào điểm của người chơi.

- Sự kiện đổi người chơi được thay đổi luân phiên khi tất cả các sự kiện trên được dừng.

Trạng thái kết thúc của loạt sự kiện trên cũng chính là trạng thái bắt đầu của người chơi tiếp theo trong vòng luân phiên. Nếu trạng thái kết thúc của loạt sự kiện trùng với một trong các trạng thái cuối (dừng lại) của trò chơi sẽ có các trường hợp kéo theo các sự kiện có thể xảy ra:

- Nếu khi 2 quan đồng thời bị ăn hết không còn dân nào trong ô quan:

Đối với trường hợp này số dân còn lại trong bàn cờ thuộc 2 bên người chơi sẽ được cộng vào điểm của từng người và tiến hành so sánh số điểm để kết luận thắng thua.

- Khi các ô thuộc một bên nắm giữ hết dân mà trong kho (điểm) của mình không đủ 5 dân (5 điểm) để rải đều cho 5 ô của mình: Đối với trường hợp này các quân dân còn lại sẽ được rải vào từng ô một theo chiều từ trái sang phải cho đến khi hết dân và tiếp tục chơi.

- Khi các ô thuộc một bên mà mình nắm giữ hết dân mà trong kho (điểm) của mình có nhiều hơn 5 dân (điểm) thì tiến hành rải đều cho 5 ô của mình mỗi ô một dân.

** Các trường hợp thuộc TTKT nêu trên có thể diễn ra nhiều hơn 1 lần thì các sự kiện tiếp theo vẫn sẽ lặp lại tương tự.

III. Phát biểu bài toán một cách hình thức:

- Không gian trạng thái (KGTT): Toàn bộ trạng thái có thể có của bàn cờ.

- Trạng thái đầu: Bàn cờ có 10 ô quân và 2 ô quan, mỗi ô quân có 5 quân cờ, ô quan có 1 quan và không có dân (quân) nào rải vào.

- Trạng thái cuối: Bàn cờ không còn 2 ô quan hoặc một trong 2 người chơi không còn dân (quân) để rải.

- Thuật toán: Cách xác định ô, kiểm tra các điều kiện và rẽ nhánh ra các điều kiện để bốc ô quân, chiều rải quân, … trong bàn cờ một cách hợp lý, chính xác. Những phép tính toán nằm trong vòng lặp có điều kiện dừng và tiếp tục rẽ nhánh khi có điều kiện khác thỏa mãn với vòng lặp.

IV. Cấu trúc dữ liệu và cách biểu diễn các trạng thái của bài toán

1. Biểu diễn trạng thái:

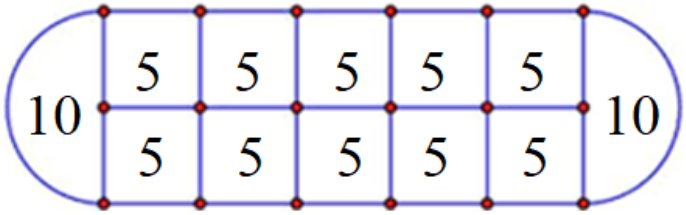
Mỗi cách bố trí quân cờ trên bàn cờ là một trạng thái. Sử dụng mảng để tạo ra một mảng (list) có 12 phần tử, mỗi phần tử có giá trị mặc định là 5. Mảng này đại diện cho bảng Ô Ăn Quan, trong đó có 12 ô cụ thể là:

- Các ô từ: 0 đến 4 là phần của người chơi 1.
- Ô số 5 là ô quan bên phải.
- Các ô từ 6 đến 10 là phần của người chơi thứ 2.
- Ô số 11 là ô quan bên trái.

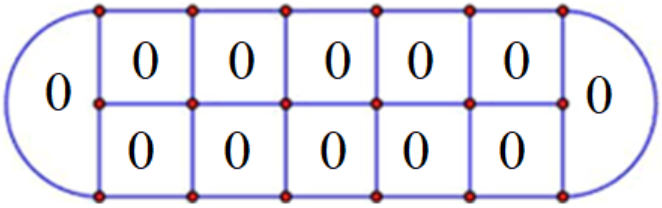
- Mỗi bước đi hợp lệ là 1 toán tử chuyển trạng thái.

2. Không gian trạng thái:

- *Trạng thái đầu*: Gồm 1 bàn cờ. Trong bàn cờ gồm có 2 quan và mỗi ô dân thì có 5 quân.



- *Trạng thái cuối*: Khi 2 quan đồng thời bị ăn hết không còn dân nào trong ô quan.



Nói cách khác: Trạng thái cuối là khi không còn đối thủ nào có thể thực hiện được nước đi hợp lệ. Các phép tính toán (kiểm tra ô tiếp theo, ăn dân, rải tiếp,…) đều phụ thuộc vào chọn ô để rải theo chiều trái hay phải (-1 hay 1).

3. Cấu trúc dữ liệu

Danh sách biến chính cần dùng:

(người chơi): Biến này đánh dấu người chơi hiện tại.

(điểm 1): Biến này để lưu trữ điểm số người chơi thứ 1.

(điểm 2): Biến này lưu trữ điểm số người chơi thứ 2.

(bảng[i]): Biến này đại diện cho bảng của trò chơi Ô Ăn Quan, và nó được sử dụng để in ra giá trị của các ô trên bảng.

(số đá): Biến này lưu trữ số hạt đá được lấy từ ô của người chơi hiện tại để sau đó phân phối ra các ô tiếp theo.

(i=tọa độ ô): Biến này được sử dụng để duyệt qua các ô trên bàn cờ khi thực hiện phân phối hạt đá.

(chiều): Là một biến được sử dụng để xác định hướng phân phối của hạt đá.

(điểm cộng): Đây là biến được sử dụng để tính điểm và hiển thị thông tin về số điểm nhận được sau khi ăn quân.

(Đá mới): Đây là biến được sử dụng để lưu trữ số quân khi 5 ô của người chơi đó hết sẽ lấy 5 quân (số quân tích lũy >= 5) hoặc lấy số quân tích lũy(< 5)

Chú thích: kí hiệu cho các biến *(tên biến)*

<u>Danh sách các hàm:</u>

<tạo bảng>: Đây là hàm khởi tạo bảng.

<hiển thị> dùng để hiển thị trạng thái hiện tại của bảng ô ăn quan.

<chơi game>: dùng để quản lý quá trình chơi của trò chơi ô ăn quan.

*<phân phối>:*hàm thực hiện quá trình phân phối đá từ một ô đã chọn ra các ô tiếp theo theo chiều trái hoặc phải.

<rải thêm>: hàm thực hiện việc rải đá khi có người chơi hoặc cả hai người chơi đã hết đá trong các lỗ của mình.

B. Giao Diện

I. Mục tiêu:

- Tạo giao diện menu: hình ảnh, văn bản, hình nền cho menu , các chế độ.

- Các phần tử bao gồm: các lựa chọn chế độ chơi (2 người chơi hoặc chơi với máy) khung lựa chon chế độ, tiêu đề " Ô Ăn Quan" và văn bản cho các lựa chọn

- Tạo một cửa sổ và hiển thị nền (background), biểu tượng cho cửa sổ cho trò chơi "Ô Ăn Quan".

- Cài đặt font, điều chỉnh kích thước cho phù hợp.

- Tạo ra bàn cờ gồm 12 ô trong đó có 10 ô quân (hình vuông) và 2 ô quan (2 nửa ô hình elip).

- Khởi tạo biến để lưu trữ thông tin về trạng thái của ô chứa dân (số lượng dân trong các ô, số lượng quân trong ô quan ở mỗi bên, ...).

- Xử lý điều khiển di chuyển và lựa chọn ô trên bàn cờ.

- Cách tạo tọa độ các ô để gán giá trị tọa độ của ô nhằm xác định chính xác vị trí từng ô cụ thể.

- Hiển thị số lượng hạt của từng ô lên màn hình.

- Hiển thị lựa chọn, bàn cờ, số lượng đá ra màn hình

- Xử lý lựa chọn từ bàn phím để thực hiện các câu lệnh.

- Cập nhật màn hình liên tục để hiện thị trạng thái mới nhất.

II. Phương án:

1. Giao diện chọn chế độ chơi:

Hiển thị từ hàm <*menu*> , hàm menu sẽ chạy trước để hiển thị màn hình game.

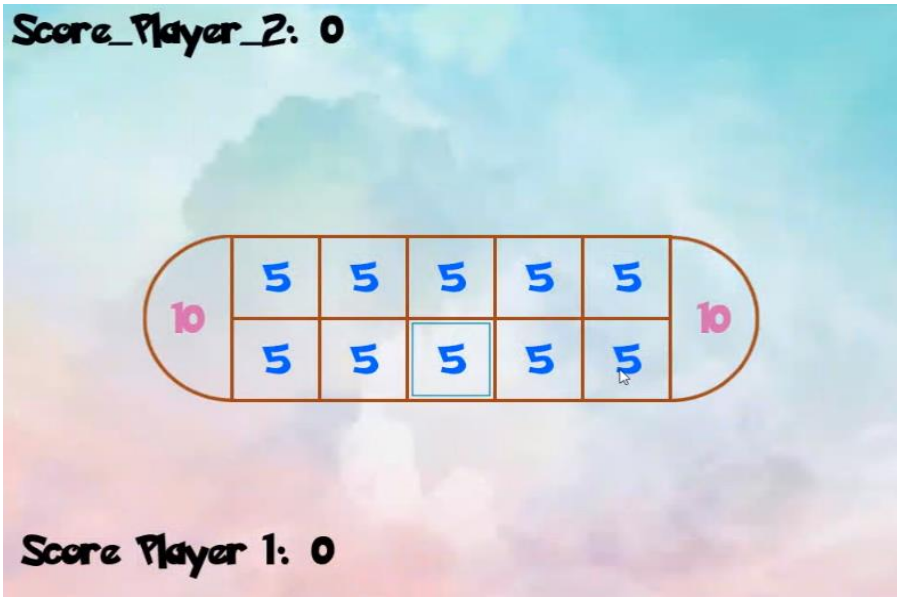
- Ô chọn hiển thị trạng thái lựa chọn nếu như người chơi nhấn vào:

  + Chế độ "2 người chơi", từ hàm <*menu*> sẽ khởi chạy hàm <*chedo1*> để người chơi bước vào màn hình game và chơi game tại đó.

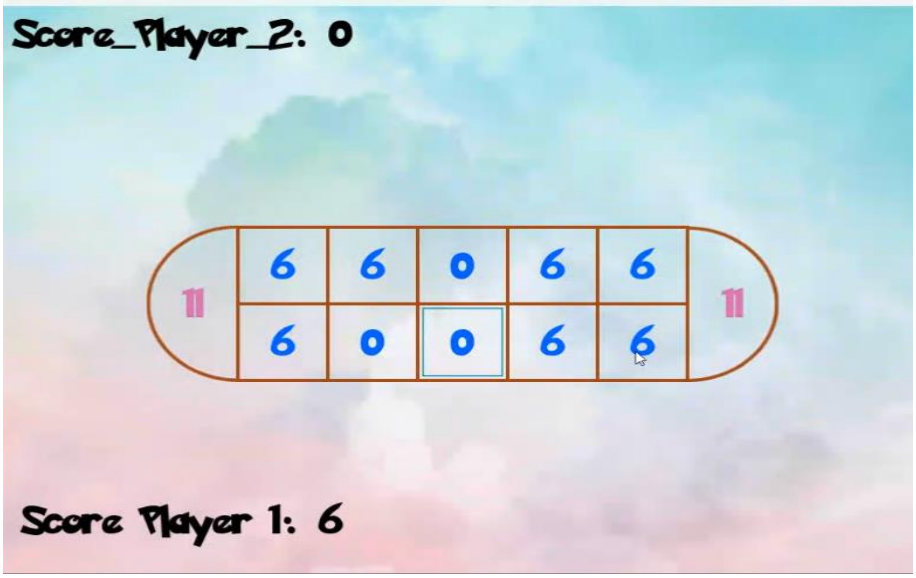
  + Chế độ "Chơi với máy", từ hàm <*menu*> sẽ khởi chạy hàm <*chedo2*> để người chơi bước vào màn hình game và chơi game tại đó.

2. Giao diện màn hình chơi game.

- Màn hình trước khi bắt đầu chơi:

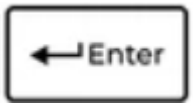


- Thực hiện rải quân khi chọn ô thứ 2, di chuyển sang bên phải và thực hiện ăn quân.



b. Các nút chức năng cơ bản trong game.

: Di chuyển ô chọn.



: Xác nhận vào game.

: Chọn chiều trái phải, chọn ô.

Phần 4: : Phân tích bài toán theo tư duy lập trình Thiết kế thuật toán trò chơi dựa trên ngôn ngữ lập trình Python

I. Khởi tạo biến cơ bản:

- Ban đầu khởi tạo được bàn chơi có 12 ô và mỗi ô có 5 viên đá.
- Sau đó gán giá trị cho 2 ô quan là ô thứ 11 và ô thứ 5 là 10 viên đá.
- Khởi tạo người chơi ban đầu (người chơi thứ 1) = 0 và điểm số mỗi người chơi ban đầu là 0.

```
def taobang():
    global board, so1, so2, so3, so4, so5, so_quanR,
so6, so7, so8, so9, so10, so_quanL, \
        current_player, scored_01, scored_02
    board = [5] * 12
    board[5] = board[11] = 10
    current_player = 0
    scored_01 = 0
    scored_02 = 0
```

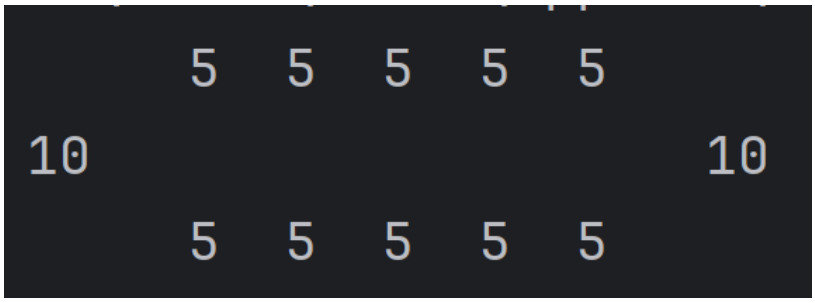
II. Hiển thị bàn cờ trên console:

- Dùng **print(“ ”)** để in ra bàn cờ theo định dạng.
- Dùng cấu trúc for để k chạy trên các ô trong khoảng cố định từ (0,5) theo thứ tự tăng dần và (10,5,-1) theo thứ tự giảm dần.
- In giá trị của **board[k]** với khoảng trống để tạo ra khoảng cách cho mỗi ô trên bàn cờ.
- In ra một dòng trống để tạo ra khoảng cách giữa các lần in bàn cờ.

```
def print_board():
    print("     ", end="")
    for k in range(10, 5, -1):
        print(board[k], " ", end="")
    print()
    print(board[11], "                ", board[5])
    print("     ", end="")
    for k in range(5):
```

```
        print(board[k], " ", end="")
    print("\n")
```

OUTPUT:



III. Điều kiện chơi:

- Phần ô chơi của người thứ 1 nằm trong khoảng từ ô thứ 0 đến ô thứ 5 tương ứng là từ ô thứ 0 đến ô thứ 4 trên bàn cờ và người chơi thứ 2 là từ ô số 6 đến ô số 11 tương ứng từ ô thứ 6 đến ô thứ 10 trên bàn cờ.
- Đặt người chơi thứ 1 là 0 và người chơi thứ 2 là 1 nếu người chơi tiếp theo bằng 2 thì ta sẽ gán cho người chơi đó trở về người chơi thứ 1.
- Trường hợp trò chơi kết thúc hay vòng lặp kết thúc khi và chỉ khi 2 ô quan không còn đá. Khi đó số đá thuộc phần ô của ai thì người đó sẽ thu lại và đếm số lượng đá và so sánh số đá để phân loại thắng thua hòa và sau đó in ra màn hình kết quả của trò chơi
- Với **board[0:5]** là đá từ ô 0 đến 4 và **sum(board[0:5])** là tổng cộng số đá trong 5 ô.

```
-
    if current_player == 2:
        current_player = 0
    if board[5] == 0 and board[11] == 0:
        scored_01 += sum(board[0:5])
        scored_02 += sum(board[6:11])
        if scored_01 < scored_02:
-
            print("Player 2: Win!")
        elif scored_02 < scored_01:
            print("Player 1: Win!")
        else:
            print("Hòa")
        break
```

```
-
```

IV. CHỌN Ô

- Người chơi thứ 1 được chọn ô bất kì từ 0 đến 4 tương ứng với ô từ 1 đến 5 trên bàn cờ.
- Người chơi thứ 2 được chọn ô bất kì từ 6 đến 10 tương ứng với ô từ 7 đến ô thứ 11 trên bàn cờ (hay chính là ô từ 1 đến 5 dưới góc nhìn của người thứ 2).
- Trong hình ảnh bên dưới là người chơi thứ 2:
  + Thực hiện chọn ô.

  + Kiểm tra điều kiện: Nếu chọn vào lỗ thứ 5 hoặc ô thứ 11 hoặc ô được chọn không có đá thì sẽ thực hiện chọn lại ô.

  + Nếu là người chơi thứ 1 thì thực hiện tương tự.

```
if current_player == 0:
    print("Player 1")
    hole = int(input("Chọn lỗ (0-4) <=> (1-5): "))
    count += 1
    if hole == 5 or hole == 11 or board[hole] == 0:
        print(" lựa chọn không hợp lệ. vui lòng chọn lại !")
        continue
if current_player == 1:
    print("Player 2")
    hole = int(input("Chọn lỗ (6-11) <=> (1-5): "))
    if hole == 5 or hole == 11 or board[hole] == 0:
        print(" lựa chọn không hợp lệ. vui lòng chọn lại !")
         continue
```

- Gán stones = số đá trong ô đã chọn và trả về ô chọn giá trị bằng 0.

```
stones = board[hole]
board[hole] = 0
```

V. CHỌN CHIỀU

- Nhập từ bàn phím giá trị trái hoặc phải.
  + Nếu chiều là “ t ”(trái) thì giá trị i đc gán bằng tọa độ ô chọn trừ đi 1 ô.

  + Nếu chiều là “ p ”(phải) thì giá trị i đc gán bằng tọa độ ô chọn cộng thêm 1 ô.

```
# rẽ nhánh, chọn chiều rải đá
chieu = input("Chieu p/t: ")
if chieu == "t":
    i = hole - 1
elif chieu == "p":
    i = hole + 1
```

VI. THỰC HIỆN RẢI ĐÁ

	CHIỀU BÊN TRÁI	CHIỀU BÊN PHẢI

SỐ ĐÁ SAU KHI BỐC Ô	+ Sử dụng vòng lặp while để thực hiện vòng lặp rải đá cho đến khi số đá = 0.
	
	+ Giá trị i thay đổi, di chuyển sang bên trái sau mỗi lần rải đá và từng ô tương ứng cộng đá.
	
	+ Vì sao ta có điều kiện nếu i < 0 thì i = 11?
	
	Vì bàn cờ có 12 ô đá nên i chạy từ 0 đến 11. Nếu trường hợp số đá lớn hơn 12 và mỗi lần lặp biến i sẽ trừ đi 1 giá trị nên sẽ âm nên khi i < 0 thì ta sẽ gán i = 11.
	
	`while stones > 0:`
	`  if i < 0:`
	`  i = 11`
	`  board[i] += 1`
	`  stones -= 1`
	`  i = i - 1`
	
	Output:
	
	6 6 6 6 5
	11 10
	0 5 5 5 5
		+ Sử dụng vòng lặp while để thực hiện vòng lặp rải đá cho đến khi số đá = 0.
		
		+ Giá trị i thay đổi, di chuyển sang bên phải sau mỗi lần rải đá và từng ô tương ứng cộng đá.
		
		+ Vì sao ta có điều kiện nếu i > 11 thì i = 0?
		
		Vì bàn cờ có 12 ô đá nên i chạy từ 0 đến 11. Nếu trường hợp số đá lớn hơn 12 và mỗi lần lặp biến i sẽ cộng thêm 1 giá trị nên sẽ vượt quá 12 ô(không có đá ở ô thứ 12) do đó khi mà i > 11 thì ta sẽ gán i = 0.
		
		`while stones > 0:`
		`  if i > 11:`
		`  i = 0`
		`  board[i] += 1`
		`  stones -= 1`
		`  i = i + 1`
		
		Output:
		
		5 5 5 5 6
		10 11
		5 0 6 6 6

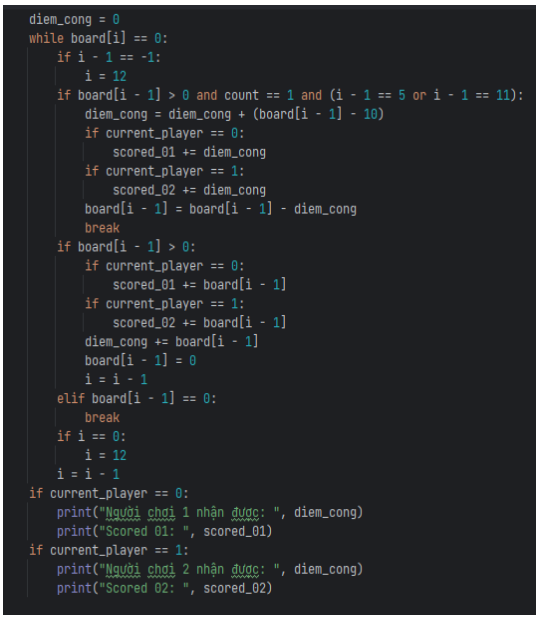
Thực hiện rải tiếp quân	+ i là giá trị thay đổi liên tục nhưng nằm trong giới hạn từ 0 đến 11 nên ta phải đặt điều kiện để tạo sự chặt chẽ.
	
	+ Nếu ô cuối cùng để rải đá là ô thứ 0 thì ô tiếp theo là ô thứ -1 sẽ bị cộng lung tung vì không có ô thứ -1 thay vào đó ta sẽ chặn giá trị nhỏ hơn 0 và gán nó = 11.
	
	+ Sử dụng vòng lặp while để thực hiện vòng lặp kiểm tra ô tiếp theo và thực hiện các câu lệnh cho đến khi ô tiếp theo = 0 thì kết thúc.
	
	+ Nếu khi rải xong đá và ô tiếp theo lớn hơn 0 nhưng là ô quan thì sẽ mất lượt.
	
	`if i < 0:`
	`  i = 11`
	`while self.board[i] > 0:`
	
	`  if i == 5 or i == 11:`
	`  break`
	`  stones2 = self.board[i]`
	`  self.board[i] = 0`
	`  i = i - 1`
	`  while stones2 > 0:`
	`  if i < 0:`
	`  i = 11`
	`  self.board[i] += 1`
	`  stones2 -= 1`
	`  i = i - 1`
	`  if stones2 == 0 and i == -1:`
	`  i = 0`
	
	Output:
	
	`  6 6 6 6 0`
	`11 11`
	`  0 6 6 6 6`
		+ i là giá trị thay đổi liên tục nhưng nằm trong giới hạn từ 0 đến 11 nên ta phải đặt điều kiện để tạo sự chặt chẽ.
		
		+ Nếu ô cuối cùng để rải đá là ô thứ 11 thì ô tiếp theo là ô thứ 12 sẽ báo lỗi vì không có ô thứ 12. Thay vào đó ta sẽ chặn giá trị lớn hơn 12 và gán nó = 0.
		
		+ Sử dụng vòng lặp while để thực hiện vòng lặp kiểm tra ô tiếp theo và thực hiện các câu lệnh cho đến khi ô tiếp theo = 0 thì kết thúc.
		
		+ i là giá trị thay đổi liên tục nhưng nằm trong giới hạn từ 0 đến 11 nên ta phải đặt điều kiện để tạo sự chặt chẽ.
		
		+ Nếu khi rải xong quân và ô tiếp theo lớn hơn 0 nhưng là ô quan thì sẽ mất lượt.
		
		`if i > 11:`
		`  i = 0`
		`while self.board[i] > 0:`
		`  if i == 5 or i == 11:`
		`  break`
		`  stones2 = self.board[i]`
		`  self.board[i] = 0`
		`  i = i + 1`
		`  while stones2 > 0:`
		`  if i > 11:`
		`  i = 0`
		`  self.board[i] += 1`
		`  stones2 -= 1`
		`  i = i + 1`
		`  if stones2 == 0 and i == 12:`
		`  i = 0`
		
		Output:
		
		`  6 6 6 0 6`
		`11 11`
		`  6 0 6 6 6`

<table>
<tr>
<td>Ăn quân</td>
<td>+ Nếu ô tiếp theo bằng 0 ta sẽ kiểm tra ô thứ i - 1 nếu lớn hơn 0 thì ta sẽ đc lấy số đá đó cho vào số điểm của mình và gán số đá ô đó trở về 0 và gán i bằng ô vừa ăn và thực hiện lại vòng lặp ăn đá.

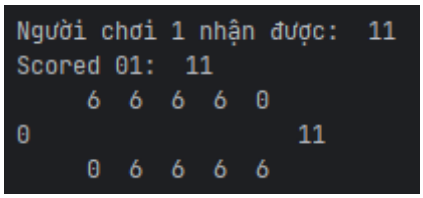
+ Nếu ô tiếp theo = 0 và ô thứ i - 1 = 0 thì kết thúc.
Trong trường hợp ô tiếp theo = 0 ta sẽ gán về 12 để thực hiện tiếp vòng lặp.

+ điểm cộng là tổng số đá người chơi có được sau mỗi lượt chơi và khi đến lượt chơi tiếp theo biến bày sẽ trở về giá trị 0.

+ điểm self,scored_02 là tổng số đá người chơi thứ 2 tích lũy sau các ván chơi.

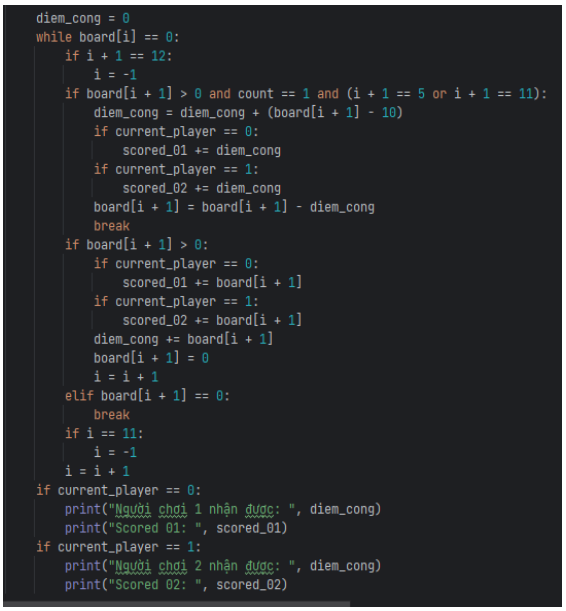
Output:

 
</td>
<td>+ Nếu ô tiếp theo bằng 0 ta sẽ kiểm tra ô thứ i + 1. Nếu lớn hơn 0 thì ta sẽ đc lấy số đá đó cho vào số điểm của mình và gán số đá của ô đó trở về 0 và gán i bằng ô vừa ăn và thực hiện lại vòng lặp ăn đá.

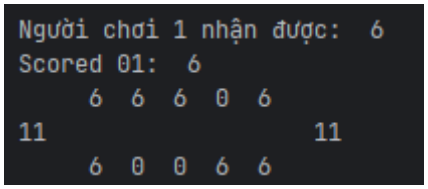
+ Nếu ô tiếp theo = 0 và ô thứ i + 1 = 0 thì kết thúc.
Trong trường hợp ô tiếp theo = 11 ta sẽ gán về 0 để thực hiện tiếp vòng lặp.

+ điểm cộng là tổng số đá người chơi có được sau mỗi lượt chơi và khi đến lượt chơi tiếp theo biến bày sẽ trở về giá trị 0.

+ điểm self.scored_01 là tổng số đá người chơi thứ 1 tích lũy sau các ván chơi.

Output:

 
</td>
</tr>
</table>

VII. KHI CẢ 5 Ô ĐỀU HẾT ĐÁ MÀ TRÒ CHƠI CHƯA KẾT THÚC

+ Nếu có số đá > 5 thì sẽ rải 5 viên đá vào 5 ô với mỗi ô 1 viên và số điểm sẽ trừ đi 5 sau đó tiếp tục quay lại thực hiện chọn ô để rải đá.
+ Nếu 0 < số đá < 5 thì có bao nhiêu đá sẽ rải vào từng ô theo thứ tự từ ô 0-4 hoặc từ 6-10 và số điểm sẽ trở về 0 sau đó tiếp tục quay lại thực hiện chọn ô để rải đá.
+ Code dưới là người chơi thứ 1 thì ta tính tổng đá trong khoảng từ (0,5) và cho các ô trong khoảng đó cộng bằng 1, thực hiện bằng điểm số của người 1.
- Nếu là người chơi thứ 2 thì tính tổng đá trong khoảng (6,11) và cho các ô trong khoảng đó cộng bằng 1, thực hiện bằng điểm số của người 2.

```
ef raithhem():
    global current_player, scored_01, scored_02
    if current_player == 0 or current_player == 2:
        if sum(board[0:5]) == 0:

            if 0 <= scored_01 < 5:
                new_stones = scored_01
                scored_01 = 0
                for z in range(0, new_stones):
                    board[z] += 1

                new_stones = 0
            if scored_01 >= 5:
                scored_01 -= 5
                board[0] = board[1] = board[2] = board[3] = board[4] = 1

    if current_player == 1:
        if sum(board[6:11]) == 0:

            if 0 <= scored_02 < 5:
                new_stones = scored_02
                scored_02 = 0
                for z in range(6, 6 + new_stones):
                    board[z] += 1

                new_stones = 0
            if scored_02 >= 5:
                scored_02 -= 5
                board[6] = board[7] = board[8] = board[9] = board[10] = 1
```

VIII: Điều kiện kết thúc trò chơi

1. Khi đá ở 2 ô quan đều bằng 0.

```
if board[5] == 0 and board[11] == 0:
    scored_01 += sum(board[0:5])
    scored_02 += sum(board[6:11])
    board[0] = board[1] =board[2] =board[3] =board[4] =board[5] = 0
    board[6] = board[7] =board[8] =board[9] =board[10] =board[11] = 0
    if scored_01 < scored_02:
        print("scored 01: ", scored_01)
        print("scored 02: ", scored_02)
        print("Player 2: Win!")
    elif scored_02 < scored_01:
        print("scored 01: ", scored_01)
        print("scored 02: ", scored_02)
        print("Player 1: Win!")
    else:
        print("scored 01: ", scored_01)
        print("scored 02: ", scored_02)
        print("Hòa")
```

2. Khi 1 người không còn đá trong 5 ô và tổng số đá tích lũy trong cả bàn chơi bằng 0 như ở phần VII đã thể hiện ở hình ảnh.

=>> Khi đó ta sẽ thu hết đá thuộc ô của mình và thực hiện so sánh để phân loại thắng thua.

- Nếu số đá người thứ 1 lớn hơn người thứ 2 thì người thứ 1 THẮNG và ngược lại.
- Nếu số đá người thứ 1 lớn hơn người thứ 2 thì HÒA.

IX. Chế độ chơi với máy.

Sử dụng <random> để chọn ngẫu nhiên ô có đá và rải về chiều ngẫu nhiên

Phần 5: Thiết kế giao diện trò chơi trên ngôn ngữ lập trình Python.

A. Đồ họa trò chơi.

I. Phương án.

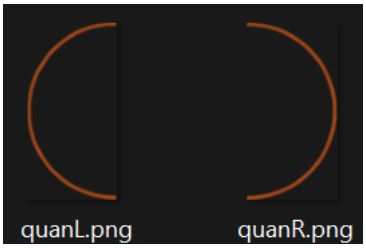
Để đơn giản việc tạo hình ảnh cho trò chơi => thiết kế hình ảnh và đưa vào trong trò chơi các hình ảnh đó. Đặt chúng đúng vị trí mong muốn hiển thị trên màn hình phù hợp với các chức năng của trò chơi.

II. Thực hiện.

1. Thiết kế các hình ảnh phù hợp với trò chơi:

a) Ô vuông (ô quan) trong trò chơi:

b) Hai ô quan:



c) Ô con trỏ (chọn vị trí, hiển thị vị trí ô hiện tại nếu như người chơi chọn vào).

d) Một số hình ảnh khác (hình nền, chọn chế độ…..)

e) Hiển thị thông số (điểm, điểm được cộng…)
Lấy giá trị của biến quy định các giá trị biến thiên cần cho hiển thị và hiển thị lên trên màn hình.

f) Phông chữ:
Lựa chọn phông chữ phù hợp. (Pokemon-Solid.ttf'…)

h) …

2. Xây dựng giao diện trên Python với Pygame.

- Thêm thư viện pygame để thực hiện các chức năng hiển thị trên màn hình trò chơi, tương tác sẽ các sự kiện, đối tượng,…

```
import pygame, sys,random
```

- Thêm từ thư mục chưa file đã chuẩn bị các file cần thiết để hiển thị hình ảnh trong các hàm:

```
BG = pygame.image.load("assets/Background.png")
pygame.display.set_caption("Ô Ăn Quan")
icon = pygame.image.load(r'image\icon.png')
```

```
pygame.font.Font("assets/font.ttf", size)
```

```
game_font = pygame.sysfont.Font(r'font\Pokemon-Solid.ttf', 40)
```

```
background = pygame.image.load(r'image\background0.png')
```

- Tạo ô dân, quan:

```
# Ô Vuông
ovuong8 = pygame.image.load(r'image\ovuong.png')
ovuong8_rect = ovuong8.get_rect(center=(500, 325))
ovuong7 = pygame.image.load(r'image\ovuong.png')
ovuong7_rect = ovuong7.get_rect(center=(596, 325))
ovuong6 = pygame.image.load(r'image\ovuong.png')
ovuong6_rect = ovuong8.get_rect(center=(692, 325))
ovuong5 = pygame.image.load(r'image\ovuong.png')
ovuong5_rect = ovuong8.get_rect(center=(692, 421))
ovuong4 = pygame.image.load(r'image\ovuong.png')
ovuong4_rect = ovuong8.get_rect(center=(596, 421))
ovuong3 = pygame.image.load(r'image\ovuong.png')
ovuong3_rect = ovuong8.get_rect(center=(500, 421))
```

```
ovuong2 = pygame.image.load(r'image\ovuong.png')
ovuong2_rect = ovuong8.get_rect(center=(404, 421))
ovuong1 = pygame.image.load(r'image\ovuong.png')
ovuong1_rect = ovuong8.get_rect(center=(308, 421))
ovuong9 = pygame.image.load(r'image\ovuong.png')
ovuong9_rect = ovuong8.get_rect(center=(404, 325))
ovuong10 = pygame.image.load(r'image\ovuong.png')
ovuong10_rect = ovuong8.get_rect(center=(308, 325))
# Ô quan
oquanR = pygame.image.load(r'image\quanR.png')
oquanR_rect = oquanR.get_rect(center=(788, 373))
oquanL = pygame.image.load(r'image\quanL.png')
oquanL_rect = oquanL.get_rect(center=(212, 373))
```

- Hiển thị các số tương ứng lên màn hình trò chơi:

```
so1_surface = game_font.render(str(so1), True, (0, 100, 255))
so2_surface = game_font.render(str(so2), True, (0, 100, 255))
so3_surface = game_font.render(str(so3), True, (0, 100, 255))
so4_surface = game_font.render(str(so4), True, (0, 100, 255))
so5_surface = game_font.render(str(so5), True, (0, 100, 255))
so6_surface = game_font.render(str(so6), True, (0, 100, 255))
so7_surface = game_font.render(str(so7), True, (0, 100, 255))
so8_surface = game_font.render(str(so8), True, (0, 100, 255))
so9_surface = game_font.render(str(so9), True, (0, 100, 255))
so10_surface = game_font.render(str(so10), True, (0, 100, 255))
```

```
so_rect10 = so10_surface.get_rect(center=(308, 330))
so_rect9 = so9_surface.get_rect(center=(404, 330))
so_rect8 = so8_surface.get_rect(center=(500, 330))
so_rect7 = so7_surface.get_rect(center=(596, 330))
so_rect6 = so6_surface.get_rect(center=(692, 330))
so_rect5 = so5_surface.get_rect(center=(692, 426))
so_rect4 = so4_surface.get_rect(center=(596, 426))
so_rect3 = so3_surface.get_rect(center=(500, 426))
so_rect2 = so2_surface.get_rect(center=(404, 426))
so_rect1 = so1_surface.get_rect(center=(308, 426))
```

```
so_quanR_surface = game_font.render(str(so_quanR), True, (213, 108, 177))
so_quanL_surface = game_font.render(str(so_quanL), True, (213, 108, 177))

quanR_rect = so_quanR_surface.get_rect(center=(788, 378))
quanL_rect = so_quanL_surface.get_rect(center=(212, 378))
```

#Hiển thị điểm, tên, điểm được cộng.

```
def hien_thi_ten():
    global Player_surface_1, Player_surface_2, Player_rect_1, Player_rect_2
    Player_surface_1 = game_font.render('Player 1', True, (0, 0, 0))
    Player_surface_2 = game_font.render('Player 2', True, (0, 0, 0))
    Player_rect_2 = Player_surface_2.get_rect(center = (170, 30))
    Player_rect_1 = Player_surface_1.get_rect(center = (170, 630))
hien_thi_ten()
def hien_thi_diem():
    global score_surface_n1, score_surface_n2, score_rect_n1, score_rect_n2
```

```
    score_surface_n1 = game_font.render(f'Score 1: {scored_01}', True, (0, 0,
0))
    score_surface_n2 = game_font.render(f'Score 2: {scored_02}', True, (0, 0,
0))
    score_rect_n2 = score_surface_n2.get_rect(center=(200, 70))
    score_rect_n1 = score_surface_n1.get_rect(center=(200, 670))
hien_thi_diem()

def hien_thi_diem_cong():
    global diem_cong_n1, diem_cong_n2, diem_cong_rect_1, diem_cong_rect_2
    if current_player == 0 or current_player == 2:
        diem_cong_n1 = game_font.render(f'Plus point: {diem_cong}', True, (0, 0,
0))
        diem_cong_rect_1 = diem_cong_n1.get_rect(center=(200, 110))
    if current_player == 1:
        diem_cong_n2 = game_font.render(f'Plus point: {diem_cong}', True, (0, 0,
0))
        diem_cong_rect_2 = diem_cong_n2.get_rect(center = (200, 710))
hien_thi_diem_cong()
```

```
window.blit(background, (0, 0))

window.blit(so1_surface, so_rect1)
window.blit(so2_surface, so_rect2)
window.blit(so3_surface, so_rect3)
window.blit(so4_surface, so_rect4)
window.blit(so5_surface, so_rect5)
window.blit(so6_surface, so_rect6)
window.blit(so7_surface, so_rect7)
window.blit(so8_surface, so_rect8)
window.blit(so9_surface, so_rect9)
window.blit(so10_surface, so_rect10)

window.blit(so_quanR_surface, quanR_rect)
window.blit(so_quanL_surface, quanL_rect)
# Ovuong
window.blit(ovuong1, ovuong1_rect)
window.blit(ovuong2, ovuong2_rect)
window.blit(ovuong3, ovuong3_rect)
window.blit(ovuong4, ovuong4_rect)
window.blit(ovuong5, ovuong5_rect)
window.blit(ovuong6, ovuong6_rect)
window.blit(ovuong7, ovuong7_rect)
window.blit(ovuong8, ovuong8_rect)
window.blit(ovuong9, ovuong9_rect)
window.blit(ovuong10, ovuong10_rect)

window.blit(ochon, (ochon_dichuyen_ngang, ochon_dichuyen_doc))

# Oquan
window.blit(oquanR, oquanR_rect)
window.blit(oquanL, oquanL_rect)

window.blit(Player_surface_1, Player_rect_1)
window.blit(Player_surface_2, Player_rect_2)
window.blit(score_surface_n1, score_rect_n1)
window.blit(score_surface_n2, score_rect_n2)
```

.....

Cập nhật màn hình khi có sự kiện diễn ra.

```
# Cập nhật màn hình
pygame.display.flip()
```

Phần 6: Kết quả chạy chương trình

- Giao diện đẹp, kết cấu và màu sắc phù hợp với đề tài.

- Các ô đều được hiển thị số lượng quân lên màn hình. Điều đó khiến cho việc quan sát tính toán điểm trở nên dễ dàng, dễ nắm bắt.

- Việc ăn quân và rải quân được diễn ra thuận lợi.

- Thao tác thực hiện rải quân nhanh chóng.

- Các trường hợp đặc biệt như rải thêm quân, ăn quân liên tiếp đã được thực hiện.

- Các điều kiện chọn ô để rải quân được lặp lại khi không thỏa mãn.

- Điểm số của mỗi người chơi được hiển thị ra màn hình sau mỗi lượt chơi.

- Khi 2 ô quan đều bằng giá trị 0 thì thực hiện thu quân và phân loại thắng thua.

Phần 7 : Kết luận

I. KẾT QUẢ ĐẠT ĐƯỢC

1. Game đã chạy đúng luật của trò chơi.
2. Đã hoàn thành các chức năng cơ bản trong game.
3. Có hiệu ứng âm thanh.
4. Giao diện dễ dùng có các chức năng cơ bản.
5. Nhóm khá ưng ý với giao diện nhưng cần hoàn thiện thêm.
6. Tìm hiểu và hiểu được cách máy tính lựa chọn nước đi, tính toán trước nhiều bước giống như suy nghĩ của con người
7. Có các chế độ chơi cho người chơi lựa chọn.

II. HẠN CHẾ

a. Cách chọn cấp độ.
b. Chưa đưa thời gian vào trong game.
c. Chưa cài đặt được danh sách lưu điểm người chơi.
d. Chưa có form thông báo kết quả riêng và hiện lên màn hình.

III. HƯỚNG PHÁT TRIỂN

1. Khắc phục các mặt hạn chế, hoàn thiện các chức năng cơ bản trong game.
2. Tối ưu hóa các vấn đề trong game: Tăng sự thông minh cho máy mà tránh được việc tràn bộ nhớ.
3. Hoàn thiện giao diện hiện tại.
4. Chương trình xử lý việc phân phối đá một cách đầy đủ và chính xác.
5. Điều kiện kết thúc trò chơi được kiểm tra một cách đúng đắn.

IV. TỔNG KẾT

- Chương trình Ô Ăn Quan đã đạt được mục tiêu cơ bản là thực hiện hóa một phiên bản đơn giản của trò chơi truyền thống. Đây có thể là một bước khởi đầu tốt để phát triển và mở rộng chương trình trong tương lai, với sự thêm vào các tính năng mới và cải thiện người dùng.