ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

MÔN HỌC: KIẾN TRÚC MÁY TÍNH

BÁO CÁO BÀI TẬP LỚN

BATTLESHIP

Giảng viên hướng dẫn: PGS.TS Phạm Quốc Cường

Lớp: TN01

Sinh viên thực hiện	Mã số sinh viên
Nguyễn Trương Thái Bảo	2210251

Thành phố Hồ Chí Minh, tháng 11 năm 2023

MỤC LỤC

1. GIỚI THIỆU

1.1 Về game Battleship

Battleship (hay còn gọi là Sea Battle hoặc Game of Battleships) là một trò chơi chiến thuật trên bảng, trong đó hai người chơi cố gắng tìm và tiêu diệt tàu chiến của đối phương. Luật chơi được tóm tắt như sau:

- Trò chơi sử dụng một bảng vuông được chia thành các ô, mỗi ô đại diện cho một vị trí trên "biển" và mỗi người chơi sẽ đặt các tàu chiến của mình lên bảng này.

- Các tàu chiến có kích thước khác nhau, ví dụ như 4x1, 3x1, 2x1. Mục tiêu đặt các tàu chiến vào bảng sao cho đối phương không thể dễ dàng tìm ra vị trí của chúng.

- Sau khi kết thúc giai đoạn chuẩn bị, hai người chơi lần lượt bắn phá vào bảng của đối phương bằng cách nói ra tọa độ (hàng, cột) của ô mà họ muốn bắn, nếu bắn trúng tàu, người chơi sẽ đánh dấu ô đó là "trúng".

- Trò chơi sẽ kết thúc khi có người chơi tiêu diệt hết tất cả các tàu chiến của đối phương (tàu chiến bị coi là bị phá hủy hoàn toàn khi tất cả các ô mà nó chiếm đóng đều bị bắn trúng).

1.2 Mục tiêu của bài tập lớn

Bài tập lớn yêu cầu sinh viên hiện thực một phiên bản đơn giản của Battleship bằng cấu trúc tập lệnh MIPS, qua đó giúp sinh viên nắm rõ và sử dụng thành thục được:

- MARS MIPS Simulator.
- Các lệnh toán học và truyền dữ liệu.
- Các lệnh rẽ nhánh có điều kiện và lệnh nhảy không điều kiện.
- Các thủ tục của hợp ngữ.

2. CÁC Ý TƯỞNG HIỆN THỰC TRÒ CHƠI

Để trò chơi được hoạt động trơn tru và hấp dẫn người chơi, cần phải thực hiện tốt được 2 phần cơ bản là Logic trò chơi và giao diện người dùng.

2.1 Logic trò chơi

Xây dựng logic trò chơi bao gồm các phần chính như sau: xử lí dữ liệu đầu vào (tọa độ tàu, tọa độ bắn), lưu trữ các dữ liệu vừa nhập và xử lí việc bắn phá.

- Trước khi dữ liệu được lưu vào bộ nhớ và tiến hành xử lí ở bước tiếp theo, chương trình phải luôn kiểm tra được tính hợp lệ của những gì người chơi nhập vào. Các đầu vào không hợp lệ sẽ dẫn đến chương trình xảy ra lỗi không mong muốn. Những yếu tố cần kiểm tra trong đầu vào:

+ Đối với tọa độ tàu chiến, đầu vào phải có dạng "a b c d" với a,b,c,d là các số tự nhiên trong đoạn từ 0 đến 6; hơn nữa, các số nhập vào phải biểu diễn được đúng kích thước của tàu chiến đang cần đặt vào "biển"; cuối cùng phải đảm bảo được rằng các tàu chiến không được đặt trùng vào nhau.

+ Đối với tọa độ bắn phá thì đơn giản hơn, chương trình chỉ cần kiểm tra rằng đầu vào phải có dạng "a b" với a,b là các số tự nhiên trong đoạn từ 0 đến 6.

- Tiếp theo, khi người chơi đã nhập vào thông tin hợp lệ, ta cần phải có một cấu trúc dữ liệu (hoặc vùng nhớ) để lưu trữ thông tin đó. Cụ thể, khi người chơi nhập vào tọa độ tàu chiến, ta phải đánh dấu được rằng tàu đang nằm ở vị trí nào bằng cách tạo ra 2 mảng (cho 2 người chơi) với kích thước 7x7 và biểu thị sự hiện diện của tàu chiến bằng giá trị của mỗi ô trên bảng (0 cho vùng "biển" trống, 1 cho tàu chiến).

- Cuối cùng, khi giai đoạn chuẩn bị ở trên đã hoàn tất, trò chơi bắt đầu. Các người chơi sẽ lần lượt nhập tọa độ mà mình muốn bắn phá, lúc này chương trình sẽ phải dùng tọa độ đó truy xuất vào mảng của người chơi còn lại để kiểm tra xem ô đó có chứa tàu hay không, nếu trúng tàu thì phải đánh dấu ô đó đã bị đánh

trúng. Chương trình cần lặp đi lặp lại bước này cho đến khi có một bên bị tiêu diệt hết tàu.

2.2 Giao diện người dùng

Đối với yêu cầu thiết kế một giao diện người dùng thân thiện để người chơi có thể chơi một cách dễ dàng mà không gây ra bất cứ nhầm lẫn nào, ta phải xây dựng được một hệ thống các lời chỉ dẫn, thông báo và hình ảnh hóa các dữ liệu được người chơi nhập vào.

Để hướng dẫn người chơi trong lúc đang thực hiện các thao tác của trò chơi, chương trình phải được thiết kế để in ra những dòng yêu cầu người chơi phải làm gì, làm như thế nào để chương trình chạy đúng theo ý muốn. Cụ thể, trong giai đoạn chuẩn bị của trò chơi, các câu lệnh yêu cầu người chơi nhập tọa độ của các loại thuyền có kích thước khác nhau cũng như các yêu cầu để nhập được những tọa độ hợp lệ như đã trình bày ở phần trên được in ra để người chơi biết mình đang đặt loại thuyền nào xuống "biển" và đặt như thế nào cho đúng.

Bên cạnh đó, nếu có xuất hiện lỗi trong đầu vào thì chương trình cũng phải in ra thông báo để người chơi biết đầu vào đã bị lỗi và cho biết đó là lỗi gì để người chơi khắc phục vào lần nhập lại ngay sau đó.

Tiếp theo, để hình ảnh hóa vị trí tàu chiến sau khi người chơi nhập tọa độ vào, ta dùng công cụ bitmap display được tích hợp sẵn trong MARS MIPS simulator để vẽ nên một bảng có kích thước 7x7 để khi chương trình nhận được tọa độ đầu tàu và đuôi tàu, các ô trong bảng sẽ được tô màu tại những vị trí mà tàu đang chiếm đóng để người chơi có thể dễ dàng tính toán được chiến thuật từ đó sắp xếp tiếp vị trí các tàu chiến tiếp theo.

3. HIỆN THỰC TRÒ CHƠI

3.1 Hiện thực logic trò chơi

3.1.1 Nhận, kiểm tra đầu vào

- Các đầu vào trong chương trình đều sẽ được nhận vào theo kiểu chuỗi (string) để tối ưu với việc đọc và trích xuất dữ liệu. Đối với input là tọa độ đầu và đuôi thuyền, ta chỉ cần khai báo số kí tự tối đa để đọc là 8, vì input có dạng "a b c d" cộng thêm kí tự kết thúc "\0" ở cuối tổng là 8 kí tự, điều này giúp ngăn người chơi có thể nhập nhiều hơn tạo thành những input lỗi, bên cạnh đó còn tiết kiệm được vùng nhớ. Còn với input là tọa độ để bắn phá, số kí tự tối đa để đọc là 4 cũng vì lí do tương tự.

- Để kiểm tra được tính hợp lệ của input là tọa độ đầu tàu và mũi tàu, ta dùng vòng lặp lặp qua từng kí tự từ đó đánh giá tính đúng đắn:

+ Trước tiên, ta cần khởi tạo một biến đếm để đếm số lần xuất hiện của các kí tự số xuất hiện trong input và mở stack với 4 vị trí (4 byte) để lưu lại 4 chữ số đầu tiên tìm được trong input.

+ Nếu các kí tự đang xét là chữ số từ 0 đến 6, ta lưu nó vào stack theo thứ tự từ xuất hiện trước đến sau và tăng biến đếm lên 1 đơn vị.

+ Nếu kí tự đang xét là dấu cách, ta chuyển sang lần lặp tiếp theo.

+ Nếu xuất hiện các kí tự khác 2 loại vừa nêu trên (trừ kí tự kết thúc "\0"), có nghĩa là input đã không hợp lệ, ta in ra thông báo lỗi về định dạng của input và quay lại phần yêu cầu người dùng nhập input.

+ Ta kết thúc vòng lặp khi gặp kí tự "\0". Khi này, nếu biến đếm có giá trị khác 4 cũng có nghĩa là input cũng đã vi phạm lỗi định dạng và chương trình sẽ xử lí như trên.

- Để kiểm tra được tính hợp lệ của input là tọa độ bắn phá, ta cũng làm tương tự chỉ khác ở chỗ stack chỉ cần mở 2 byte và biến đếm khi kết thúc cũng chỉ cần có giá trị bằng 2.

3.1.2 Lưu trữ dữ liệu nhận được từ đầu vào

- Để biểu diễn vị trí của các tàu chiến đang chiếm đóng trong bộ nhớ, ta cần khai báo 2 vùng nhớ có độ lớn 49 byte (= 7 byte x 7 byte, lưu ý rằng ta chỉ cần 1 byte để lưu trữ trạng thái của vùng biển là trống hay có tàu đang chiếm) lần lượt gọi là arr1 và arr2. Do các vùng nhớ trên MIPS không thể biểu thị dạng mảng 2 chiều mà ta chỉ có thể dùng mảng 1 chiều tuyến tính, cho nên việc truy xuất phần tử trong mảng tại vị trí (tọa độ) x,y được tính như sau: arr[x][y] ⇔ arr[x * 7 + y] (7 là tổng số byte trên một hàng hoặc cột).
- Từ đó, với mỗi input nhập vào sau khi trải qua hết các giai đoạn ở phần 3.1.1, nếu input vẫn hợp lệ thì ta đi xử lí lại dữ liệu được đọc tại stack. Nhắc lại rằng input đầu tàu và mũi tàu có dạng "a b c d", ta chia ra thành 2 trường hợp là a = c (tàu nằm dài theo chiều trục x) và b = d (tàu nằm dài theo chiều trục y) vì 2 trường hợp này có cách truy xuất phần tử trong mảng là khác nhau, nếu input không nằm trong 2 trường hợp này cũng bị coi là không hợp lệ. Gọi s0 là biến chứa yếu tố bằng nhau, s1 là biến chứa giá trị nhỏ hơn trong 2 giá trị còn lại và s2 là biến chứa giá trị lớn hơn s1, ta đi tiếp tục kiểm tra tính hợp lệ của input này khi nó phải thỏa mãn biểu thức: s2 – s1 = (chiều dài của tàu – 1). Nếu không thỏa có nghĩa là người chơi đã nhập nhầm kích thước của tàu.
- Tiếp theo, tùy theo input ở trường hợp a = c hoặc b = d mà ta tùy chỉnh công thức truy xuất phần tử cho phù hợp để truy xuất lần lượt các phần tử trong arr1 hoặc arr2 từ arr[a][c] đến arr[a][d] hoặc từ arr[c][a] đến arr[d][a] để kiểm tra xem nếu tất cả các phần tử đó đều có giá trị bằng 0 hay không ở "chế độ đọc" của vòng lặp, nếu đúng là vậy thì có nghĩa là ta có thể đặt được tàu ở đây, khi đó chuyển vòng lặp sang "chế độ ghi" và chạy lại nó một lần nữa nhưng lần này ta ghi giá trị "1" xuống vùng nhớ để đánh dấu rằng tại đây có tàu chiếm đóng. Ngược lại, nếu có bất cứ giá trị nào mà "chế độ đọc" đọc được bằng 1 thì có nghĩa là người chơi đã đặt tàu vào nơi đã có tàu chiếm vào trước đó, lúc này input cũng bị coi là không hợp lệ.

- Tiếp tục quá trình cho đến khi nào thông tin tất cả các thuyền chiến của người chơi được ghi thành công vào mảng của họ.

3.1.3 Xử lí việc bắn phá

- Đây là công việc khá đơn giản khi chương trình chỉ cần truy xuất phần tử tại mảng của người chơi còn lại bằng tọa độ được nhập vào của người chơi này. Nếu giá trị truy xuất được bằng 1 thì in ra thông báo "HIT!" và ghi đè lên vùng nhớ với giá trị 0, ngược lại thì chỉ in ra thông báo "NOT HIT!".
- Để kiểm tra tính kết thúc của trò chơi, ta cần khởi tạo 2 biến đếm lùi với giá trị ban đầu bằng 16 (16 = 1*4 + 2*3 + 3*2 là tổng số ô mà các tàu chiến chiếm đóng). Biến này của từng người chơi sẽ giảm đi 1 đơn vị nếu người chơi còn lại bắn trúng tàu của mình và ai làm biến đếm của đối thủ về 0 trước thì người đó là người chiến thắng.

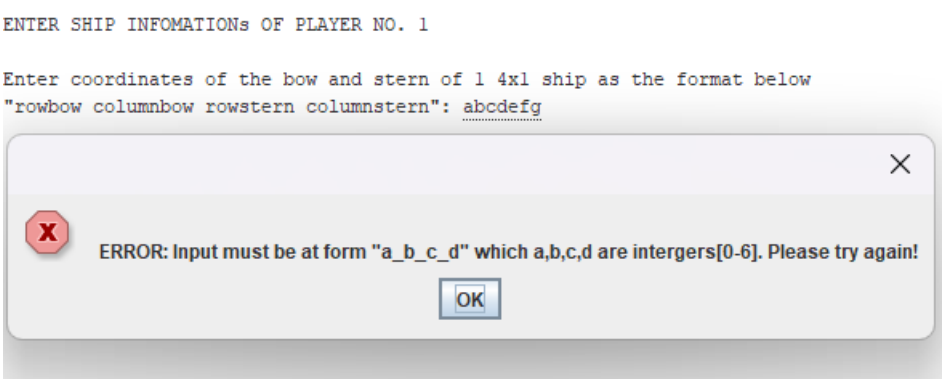
3.2 Hiện thực giao diện người dùng

Như trên đã đề cập rằng quy ước định dạng dữ liệu nhập vào là do chương trình quy định. Do đó, chương trình phải hướng dẫn người chơi thao tác nhằm tránh việc gây nhầm lẫn cho người chơi. Cụ thể, khi yêu cầu người chơi nhập bất cứ dữ liệu nào, chương trình đều in ra những dòng thông báo cộng với hướng dẫn, ví dụ:

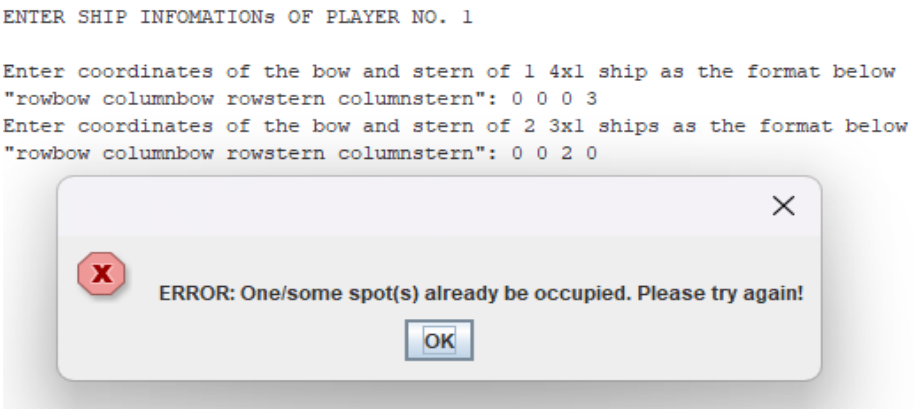
```
ENTER SHIP INFOMATIONs OF PLAYER NO. 1

Enter coordinates of the bow and stern of 1 4x1 ship as the format below
"rowbow columnbow rowstern columnstern":
```

Hình 1: Thông báo yêu cầu người chơi 1 nhập vào tọa độ của tàu có kích thước 4x1 theo mẫu

Bên cạnh đó, mỗi khi dữ liệu nhập vào bị lỗi, chương trình cũng sẽ thông báo cho người chơi rằng đầu vào là không hợp lệ và cho người chơi biết đó là lỗi gì:



Hình 2: Thông báo đầu vào bị lỗi định dạng



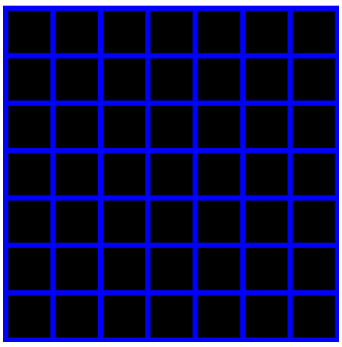
Hình 3: Thông báo đầu vào bị lỗi xếp chồng do ô (0,0) đã có tàu chiếm đóng trước

Làm được như vậy là do chương trình đã được xây dựng nhiều hàm con dùng để báo lỗi, những hàm này được gọi khi bộ phận xử lí đầu vào bắt được lỗi và gọi hàm theo đúng lỗi phát hiện được.

Tiếp theo, để hình ảnh hóa được vị trí các tàu chiến của người chơi, chương trình tiến hành vẽ một bảng có kích thước 7x7 với công cụ bitmap display. Bảng này được vẽ bởi 2 thành phần là 8 hàng ngang và 8 cột dọc. Do bảng sẽ có hình vuông, cho nên người lập trình đã chọn kích thước của bitmap là 512x512 pixel, từ đó chiều rộng của các đường thẳng được vẽ sẽ là 8 pixel và các ô vuông ở trong sẽ có kích thước 64x64 pixel (8x8 + 7x64 = 512).

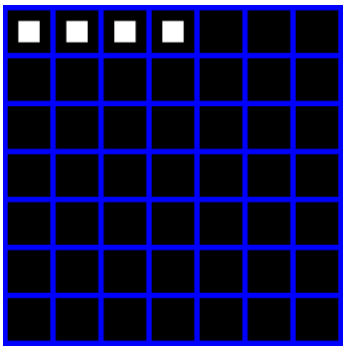
Các hàng/cột sẽ được vẽ bằng cách "lấp đầy" các byte dữ liệu bằng dữ liệu màu mà ta muốn vẽ. Cụ thể, ta dùng 3 vòng lặp lồng nhau: vòng lặp đầu tiền điều khiển số lượng đường thẳng được vẽ (8 đường), vòng lặp giữa điều khiển số dòng (chiều rộng) của một đường thẳng được vẽ (8 dòng hoặc 8 pixel) và vòng lặp cuối cùng điều khiển việc "lấp đầy" 512 pixel với màu xanh (0x000000ff). Với mỗi lần kết thúc vòng lặp thứ 3, ta sẽ tăng giá trị địa chỉ gốc

của bitmap cho số byte mà ta cần bỏ qua để được vị trí bắt đầu của đường thẳng tiếp theo (73728_{10} đối với hàng và 144_{10} đối với cột). Sau khi tiến hành 2 lần lặp để vẽ các hàng và các cột, ta được như hình sau:



Hình 4: Bảng có kích thước 7x7 hiển thị tại bitmap display.

Tiếp theo, khi người chơi nhập tọa độ của thuyền và tọa độ này đã được kiểm tra là hợp lệ, chương trình tiến hành lặp lại qua tọa độ các điểm từ đầu tàu đến đuôi tàu để với từng tọa độ đó, vẽ một ô vuông vào trong bảng trên để thể hiện rằng tại tọa độ có đang có tàu chiến chiếm đóng.



Hình 5: Bảng được đánh dấu có tàu chiến khi người chơi nhập tọa độ tàu kích thước 4x1 là "0 0 0 3".

Để xây dựng được hàm vẽ hình vuông, ta cần có tham số là tọa độ của ô cần vẽ và cần tính được đúng byte bắt đầu để vẽ. Ở đây ta chọn kích thước hình vuông là 32x32 pixel và hình vuông trắng nằm đúng chính giữa ô vuông đen, cho nên khoảng cách từ mọi cạnh của ô vuông đến cạnh của hình vuông là 16 pixel. Từ đó, gọi (x,y) là tọa độ của ô vuông cần vẽ hình vuông vào, ta có tọa độ (x', y') biểu diễn pixel bắt đầu để vẽ được tính như sau: x' = 8(x+1) + 64x + 16 = 72x + 24 (pixel), trong đó 8(x+1) là số pixel bỏ qua của các cột (hàng) ,64x là số pixel bỏ qua của các ô vuông trước đó, 16 là khoảng cách vừa nêu trên. Tương tự cũng tính được y' = 72y + 24 (pixel), từ đó dùng 2 vòng lặp lồng nhau vẽ được hình vuông và đến đây ta đã hoàn thành việc hiện thực trò chơi.

4. KẾT LUẬN

- Bài báo cáo đã trình bày được ý tưởng cũng như cách hiện thực cho từng phần của trò chơi, từ đó xây dựng được một phiên bản đơn giản của Battleship với đầu đủ các tính năng cơ bản và một giao diện người dùng thân thiện.
- Qua bài tập lớn, sinh viên đã nắm rõ và thành thục về:

+ MARS MIPS Simulator.

+ Các lệnh tính toán, di chuyển dữ liệu.

+ Các lệnh rẽ nhánh có điều kiện, lệnh nhảy không điều kiện.

+ Các thủ tục trong MIPS.

+ …