Đồ án tốt nghiệp

# GIỚI THIỆU CHUNG

Battleships là một game được thiết kế để chơi giữa hai người, trong đó hai người chơi kiểm soát một số con tàu. Hai người chơi lần lượt bắn các con tàu của người chơi còn lại, nhằm đạt được mục đích đánh chìm toàn bộ các con tàu và giành chiến thắng. Battleships được chọn vì game có thể sử dụng các hình vẽ đồ họa đơn giản và được chơi một cách đồng bộ (nói rõ hơn ở phần sau).

## Game nhiều người chơi online

Với sự ra đời và phát triển của Internet, việc kết nối nhiều người trở nên dễ dàng hơn. Đồng thời các game cũng phát triển mạnh mẽ, phục vụ cho nhu cầu giải trí của con người. Theo đó, em quyết định phát triển game Battleships, mong muốn mọi người có thể dễ dàng kết nối và có khoảng thời gian thư giãn, vui vẻ.

Game online có thể được chia ra thành nhiều thể loại: RPG (Role Playing Game), FPS (First Person Shooter), vân vân. Nói đến công nghệ thực hiện game online, một số khía cạnh phát triển game cần được xác định rõ:

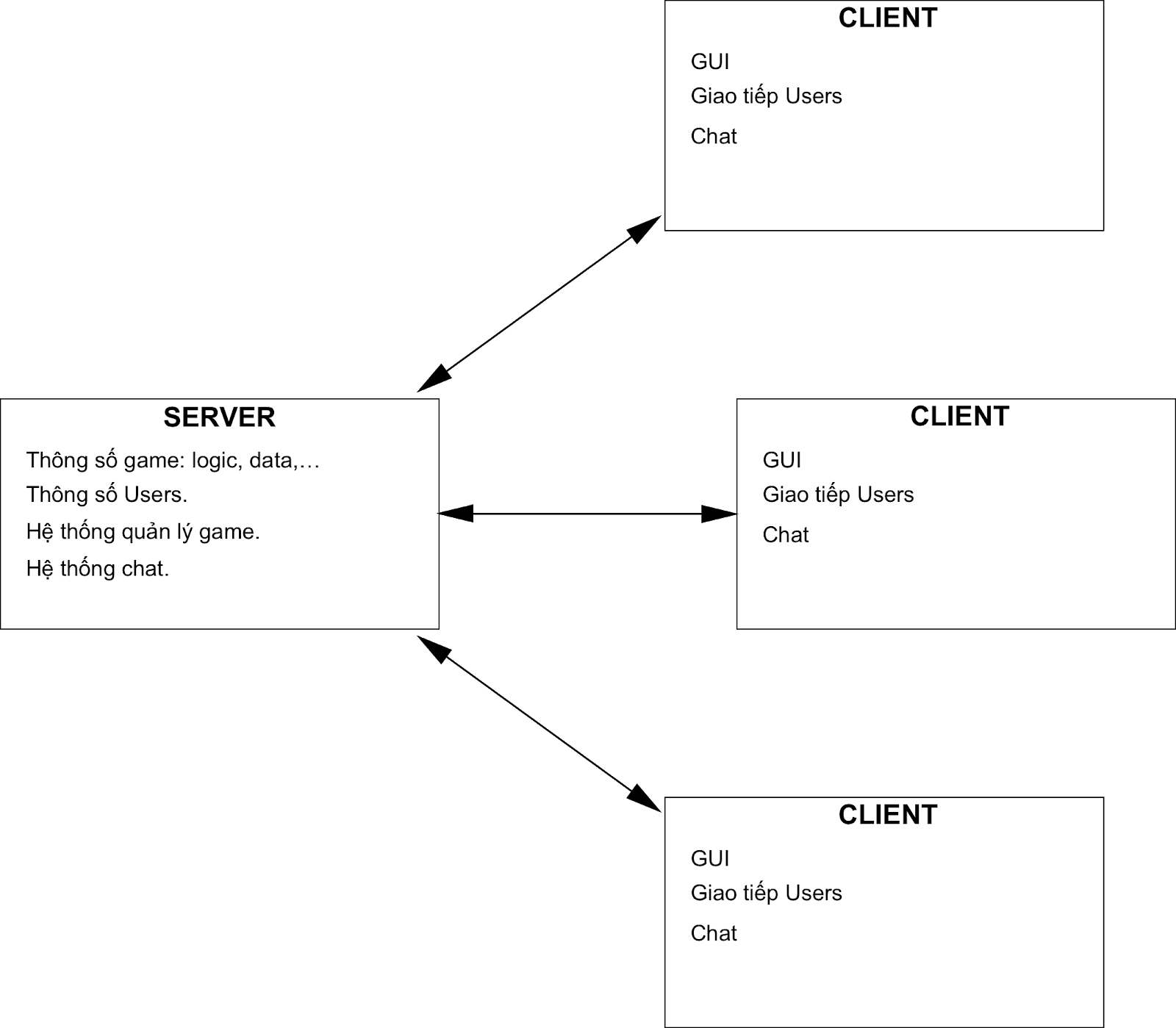
* Mối quan hệ giữa các người chơi trong game (hay nói cách khác là mô hình truyền dữ liệu được sử dụng: mô hình Client – Server, Peer-hai-Peer, vân vân).
* Sự đồng bộ trong game: Game được coi là đồng bộ khi các người chơi online đồng thời đang tham gia game, không đồng bộ khi các người chơi online có thể không đồng thời tham gia game.
* Các khía cạnh liên quan đến game client: xử lý đồ họa, âm thanh, giao tiếp với người dung và với Server, vân vân.

Như vậy, có thể thấy việc phát triển game phụ thuộc khá nhiều vào công nghệ được sử dụng và yêu cầu của game.

## Mô hình Client – Server

Với đề tài này em chọn sử dụng mô hình Client – Server để thực hiện việc giao tiếp giữa các người chơi. Mô hình được thể hiện ở *Hình 1-1*. Lựa chọn này nhằm tách riêng việc xử lý logic của game và các công việc xử lý đồ họa:

* Server: Hệ thống quản lý game (lưu trữ các thông số của game, thông số của người dùng), Xử lý logic của game.
* Client: Xử lý các công việc liên quan đến giao tiếp người dùng (đồ họa, âm thanh, thao tác người dùng), Giao tiếp với Server.



Hình 1‑1: Mô hình Client – Server

## Bố cục báo cáo

Em trình bày báo cáo theo bố cục như sau:

**Chương 2: Tổng quan về đề tài**

Chương này trình bày về game Battleships nhằm giúp người đọc có hiểu biết nhất định về mục tiêu sản phẩm hướng tới. Đồng thời chương đưa ra những yêu cầu công nghệ cho sản phẩm cuối cùng.

**Chương 3: Phát triển Server**

**Chương 4: Phát triển Client**

**Chương 5: Giao tiếp giữa Client – Server**

**Chương 6: Kết quả**

**Chương 7: Công cụ sử dụng**

**Chương 8: Phản hồi và cải thiên sản phẩm trong tương lai**

**Chương 9: Kết luận**

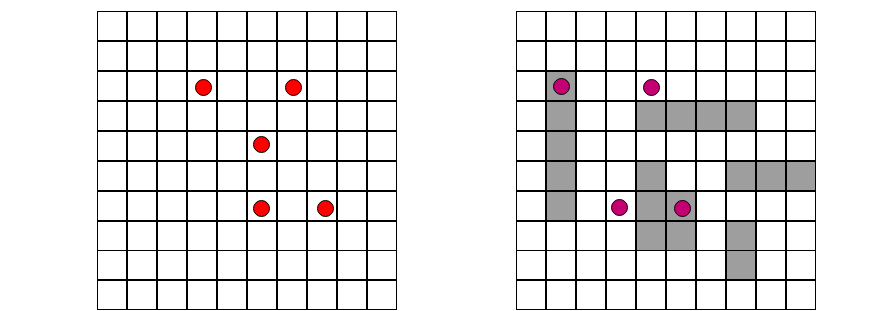
# TỔNG QUAN VỀ ĐỀ TÀI

## Game Battleships

Battleships là game pencil-and-paper (tạm dịch: giấy và bút) khá phổ biến, từ Thế Chiến thứ I. Do ngày nay công nghệ phát triển, game có thể được chơi trên các nền tảng khác.

Luật chơi khá đơn giản. Số lượng người chơi cần thiết là hai người:

* Mỗi người chơi sẽ dùng hai ma trận vuông, đề tài này sử dụng ma trận 10x10 (*Hình 2-1*).
* Một ma trận A dùng để sắp xếp các con tàu và ghi lại vị trí bị bắn.
* Một ma trận B dùng để ghi lại vị trí bắn.



Hình 2‑1: Minh hoạ hai ma trận dùng trong game.

Bên trái: Ma trận B. Bên phải: Ma trận A.

* Trước khi bắt đầu game, mỗi người chơi cần phải sắp xếp các con tàu của mình trên ma trận A và không để cho người còn lại biết vị trí các con tàu của mình:
  + Mỗi con tàu sẽ chiếm một khoảng các ô nằm cạnh nhau trên ma trận. Điều này có nghĩa là một con tàu chỉ được nằm ngang hoặc nằm dọc và không bị đứt quãng.
  + Các con tàu không nằm đè lên nhau
  + Các loại tàu là như nhau với cả hai người chơi. Đề tài này sử dụng các loại tàu sau:

|  |  |  |
| --- | --- | --- |
| **Loại tàu** | **Độ dài** | **Số lượng** |
| Tàu 5 | 5 | 1 |
| Tàu 4 | 4 | 1 |
| Tàu 3 | 3 | 2 |
| Tàu 2 | 2 | 2 |

Bảng 2‑1: Các loại tàu trong game

* Sau khi đã sắp xếp tàu xong, hai người chơi lần lượt “bắn” (Người chơi 1, người đầu tiên kết nối với server, được bắn trước). Mục đích là để đánh chìm tàu đối phương, khi một người không còn con tàu nào thì người còn lại thắng cuộc.
  + Mỗi người chơi lần lượt bắn một lần. Mỗi lần được bắn một ô trên ma trận.
  + Người bị bắn phải thông báo cho người bắn biết là họ có bắn trúng tàu hay không
* Game kết thúc khi tất cả các con tàu của một người chơi bị đánh chìm.

## Thiết kế hệ thống

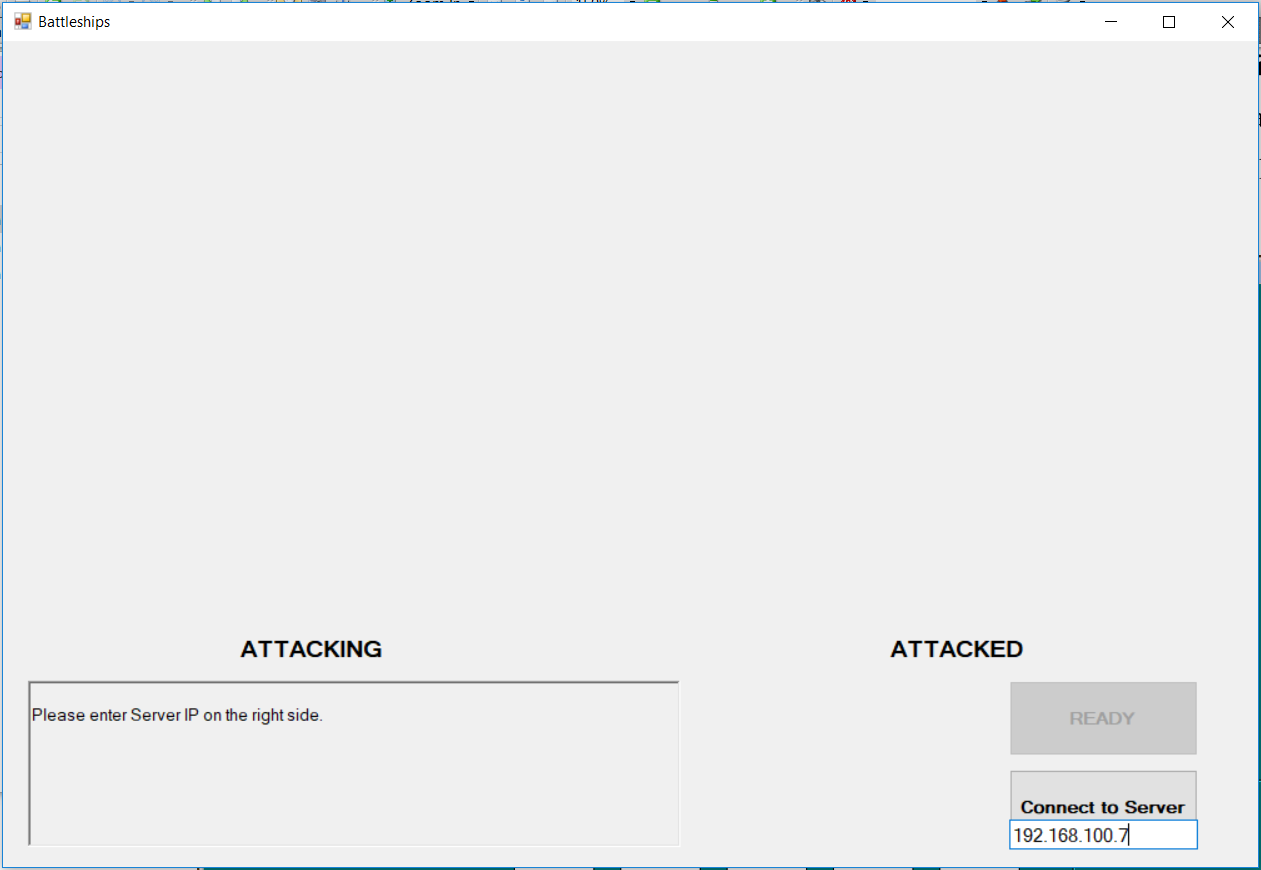
Hệ thống game được xây dựng trên mô hình Client – Server, tại đó nhiều client có thể cùng kết nối với nhau thông qua một server trung tâm. Hầu hết những logic của game diễn ra tại server, còn client tập trung xử lý các công việc liên quan đến giao tiếp người dùng. Ngôn ngữ em chọn sử dụng là C#, server chạy trên Console, client chạy trên Windows Form. Cả client và server đều chạy trên nền tảng Windows.

### Thiết kế Server

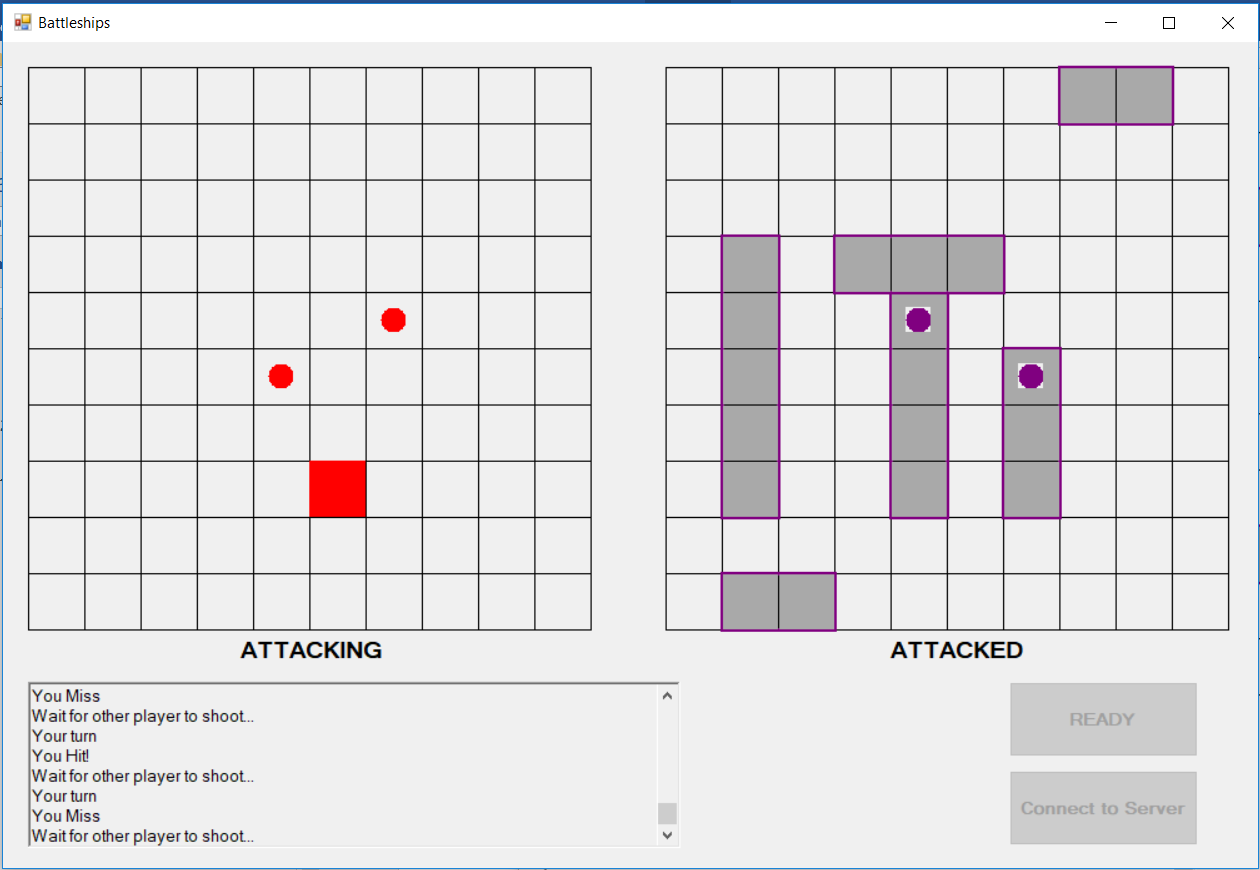
Server thực hiện 4 nhiệm vụ chính: kết nối các client, khởi tạo thông số game, xử lý logic game và kết thúc game. Việc kết nối các client có thể thực hiện bằng cách sử dụng socket. Sau khi đã kết nối với đầy đủ client (ở đây là hai client, tương ứng với hai người chơi), server khởi tạo các thông số của game, gửi đến các client để thực hiện các công việc giao tiếp với người dùng ở bên client. Hiện tại, server chỉ có thể hỗ trợ hai người chơi cùng một lúc, tức là tại mỗi thời điểm chỉ có thể có một ván Battleships diễn ra. Sau đó game tiếp tục diễn ra với mỗi người chơi lần lượt bắn tàu, trao đổi thông tin qua lại giữa client và server. Server xác định điều kiện kết thúc game và thông báo lại cho client.

### Thiết kế Client

Client thực hiện 2 nhiệm vụ chính: kết nối với server, giao tiếp với người dùng (xử lý đồ họa, tương tác của người dùng). Trước tiên, người dùng cần phải kết nối với server bằng cách gõ địa chỉ IP của server (*Hình 2-2*). Khi server nhận được thông báo đã có đủ client kết nối (hai client), các client lần lượt nhận được các thông số của game như kích thước của ma trận, các con tàu được sử dụng. Dựa vào đó, client vẽ trên Form thể hiện các ma trận. Các con tàu được đặt vào ma trận nhờ kích hoạt Event MouseDown. Sau khi đặt các con tàu xong, người dùng thông báo cho server biết là mình đã sẵn sàng bằng nút “READY”. Tiếp đó, các người dùng lần lượt bắn tàu, vị trí bắn được hiển thị trên các ma trận. Hình ảnh về client khi được khởi tạo đầy đủ được đưa ra ở *Hình 2-3*.



Hình 2‑2: Client, kết nối với Server



Hình 2‑3: Client, trong game

# PHÁT TRIỂN SERVER

Khi server khởi động, nó sẽ bắt đầu thiết lập socket để kết nối với các client. Khi đã kết nối với đầy đủ hai client, server sẽ khởi tạo class Battleships, class này sẽ xử lý toàn bộ logic và dữ liệu liên quan đến game.

## Socket

Để có thể giao tiếp giữa các máy trong cùng một mạng LAN ta cần phải có phương thức để đại diện cho mạng. Ta cũng cần truyền dữ liệu đến đúng địa điểm và không bị lỗi. Socket là một lựa chọn hợp lý. Ở đề tài này em sử dụng hai loại socket, một cho server và loại còn lại dùng cho client. Socket sử dụng ở server để “lắng nghe” các client.

Để có thể sử dụng được socket ở server, ta cần phải tạo ra socket và các thông số cho nó. Trước hết, ta cần lấy địa chỉ IP của máy mà server chạy trên đó và xác định số port để client có thể gửi dữ liệu đến đúng chương trình. Do ta cần dữ liệu được truyền đến địa điểm nhận một cách tin cậy và đúng thứ tự nên phương thức kết nối sẽ là TCP. Tiếp theo, ta “lắng nghe” và chấp nhận tín hiệu yêu cầu kết nối từ các client.

// Listening at SERVER\_IP and PORT\_NO

SERVER\_IP = GetLocalIP();

serverSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

serverSocket.Bind(new IPEndPoint(IPAddress.Parse(SERVER\_IP), PORT\_NO));

serverSocket.Listen(NUM\_OF\_CLIENTS);

serverSocket.BeginAccept(new AsyncCallback(acceptCallback), null);

Socket tại server sẽ kết nối và nhận dữ liệu không đồng bộ, tức là quá trình kết nối và nhận dữ liệu sẽ không ngăn cản server chạy những quá trình khác. Server có thể “lắng nghe” cùng lúc hai client (const int NUM\_OF\_CLIENTS = 2). Ta cần phải kết thúc quá trình chấp nhận client bằng cách gọi EndAccept của socket tại hàm acceptCallback. Khi đó ta nhận được dữ liệu trả về là socket đại diện cho client đã kết nối. Từ đây ta có thể nhận và gửi dữ liệu với client này. Ta cũng cần có một List (clientSockets) các client đã kết nối với server.

static void acceptCallback(IAsyncResult result) {

Socket socket = null;

try {

socket = serverSocket.EndAccept(result);

clientSockets.Add(socket);

// Start receiving data

socket.BeginReceive(buffer, 0, buffer.Length, SocketFlags.None, new AsyncCallback(receiveCallback), socket);

// Accepting new client

serverSocket.BeginAccept(new AsyncCallback(acceptCallback), null);

}

catch (Exception e) {

Console.WriteLine(e.ToString());

}

Do hiện tại server chỉ có thể hỗ trợ một ván game duy nhất, tức là chỉ có hai người chơi kết nối với server cùng một lúc, ta cần chặn yêu cầu kết nối từ client thứ 3 sau khi gọi EndAccept.

// Block third connection attempts

if (clientSockets.Count == NUM\_OF\_CLIENTS) {

Console.WriteLine("Maximum clients reached");

return;

}

Từ đây client và server liên tục trao đổi dữ liệu. Vấn đề này sẽ được trình bày rõ hơn ở **Chương 5**.

Ngoài ra, do socket hoạt động không đồng bộ khi kết nối và nhận dữ liệu, sẽ có thời điểm chương trình cần phải chờ tín hiệu từ một quá trình khác để tiếp tục chạy mà không xảy ra lỗi. Ví dụ, server cần phải thiết lập được kết nối với đầy đủ hai client mới được phép khởi tạo class Battleships. Để làm được điều này, em cần phải tạm dừng thread khởi tạo Battleships, chờ tín hiệu từ thread kết nối với client. Một cách đơn giản là sử dụng class WaitHandle đã đươc định nghĩa sẵn.

// Begin accepting clients, Asyncly

serverSocket.BeginAccept(new AsyncCallback(acceptCallback), null);

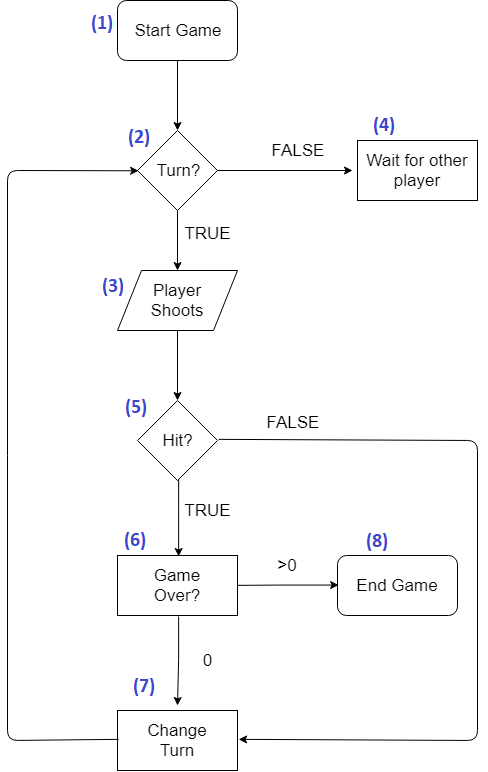
WaitHandle.WaitAll(new WaitHandle[] { eventConnect1, eventConnect2 });

battleships = new Battleships();

eventConnect1 và eventConnect2 sẽ gọi hàm Set với mỗi client đã kết nối với server. Sau đó tiến hành khởi tạo một object của class Battleships.

## Class Battleships

Do logic của game khá đơn giản nên có thể chỉ cần sử dụng một class để xử lý toàn bộ thông số của game, cùng với các kiểu dữ liệu cơ bản (int, bool). *Hình 3-1* thể hiện vòng lặp chính của game đối với mỗi người chơi.



Hình 3‑1: Core loop

Do đề tài sử dụng mô hình client – server để giao tiếp giữa các người dùng và giữa client và server nên sẽ có những quá trình chỉ xảy ra ở client hoặc chỉ xảy ra ở server. Những quá trình (1), (2), (5), (6), (7) và (8) sẽ chỉ xảy ra ở server và sau các quá trình này những dữ liệu cần thiết sẽ được gửi đến client. Những quá trình (3) và (4) sẽ chỉ xảy ra ở client, do đây là tương tác của người dùng. Sơ đồ khối cho việc giao tiếp giữa client và server sẽ được trình bày rõ hơn ở **Chương 5**.

### Thông số của game

Ta cần hai ma trận để lưu giữ vị trí các con tàu của hai người chơi và một biến Turn1 để xác định lượt của người chơi nào (true nếu là lượt của người chơi 1). Do game chỉ hỗ trợ hai người chơi nên việc chọn kiểu bool cho biến Turn1 là hợp lý. Đồng thời ta cũng cần xác định rõ loại con tàu và số lượng của mỗi loại.

public int dimension;

public int[,] matrix1, matrix2;

public bool Turn1;

public int[] NumShips;

public readonly int[] ship5 = new int[5] { 1, 1, 1, 1, 1 };

public readonly int[] ship4 = new int[4] { 1, 1, 1, 1 };

public readonly int[] ship3 = new int[3] { 1, 1, 1 };

public readonly int[] ship2 = new int[2] { 1, 1 };

public int currShipLength;

public int GameOver;

public int hitPoints1, hitPoints2;

Để hỗ trợ phía client trong việc sắp xếp tàu, server cũng định nghĩa con tàu đầu tiên cần được sắp xếp là loại tàu nào (biến currShipLength). Biến GameOver để lưu giữ trạng thái hiện tại của game:

|  |  |
| --- | --- |
| **Giá trị biến GameOver** | **Trạng thái** |
| 0 | Game chưa kết thúc |
| 1 | Người chơi 1 thắng |
| 2 | Người chơi 2 thắng |

Bảng 3‑1: Biến GameOver và trạng thái của game

Để thuận tiện cho việc kiểm tra điều kiện game đã kết thúc chưa, em sử dụng một biến hitPoints cho mỗi người chơi. Ban đầu sẽ khởi tạo bằng không, và sau khi người chơi đã sắp xếp đầy đủ các con tàu thì biến này sẽ là tổng độ dài của tất cả các con tàu.

Sau đó bắt đầu quá trình (1) ở *Hình 3-1*, khởi tạo các giá trị cho các thông số này cho game (như đã định nghĩa ở mục 2.1.

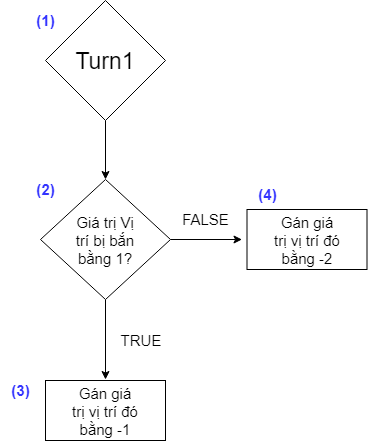
### Kiểm tra việc bắn tàu

Để kiếm tra việc bắn tàu của người chơi có trúng con tàu nào không, ta cần dựa vào các ma trận đại diện cho mỗi người chơi. Các giá trị của mỗi thành phần trong ma trận được định nghĩa như sau:

|  |  |
| --- | --- |
| **Giá trị** | **Định nghĩa** |
| 0 | Vị trí này không bị chiếm bởi con tàu nào và chưa bị bắn. |
| -2 | Vị trí này không bị chiếm bởi một con tàu và đã bị bắn. |
| 1 | Vị trí này bị chiếm bởi một con tàu và chưa bị bắn. |
| -1 | Vị trí này bị chiếm bởi một con tàu và đã bị bắn. |

Bảng 3‑2: Giá trị của các thành phần trong ma trận và định nghĩa

Sơ đồ khối để kiểm tra và cập nhật giá trị vị trí bị bắn được trình bày ở *Hình 3-2*. Quá trình (1) cần thiết để xác định lượt bắn là của người chơi nào và cập nhật đúng vị trí bắn ở ma trận của người chơi bị bắn. Có thể thấy ở sơ đồ khối bên dưới chỉ kiểm tra vị trí bị bắn có bằng 1 hay không. Ý tưởng ở đây là người chơi chỉ được phép bắn vào những vị trí chưa bị bắn, tức là các giá trị -1 và -2 sẽ không cần kiểm tra. Và việc hạn chế này được thực hiện ở phía client, tức client sẽ phản ứng ngay lập tức, không phải gửi dữ liệu lên server và chờ server gửi phản hồi về. Quá trình sẽ được trình bày rõ hơn ở **Chương 4**.



Hình 3‑2: Kiểm tra và cập nhật vị trí bị bắn