**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA ĐA PHƯƠNG TIỆN**

**------------------**

**BÁO CÁO BÀI TẬP LỚN**

**MÔN: THỰC HÀNH CHUYÊN SÂU**

**ĐỀ TÀI: “PHÁT TRIỂN GAME WORM ONLINE CHO PC BẰNG JAVASCRIPT”**

**Giảng viên hướng dẫn :**

**Sinh viên thực hiện : Nguyễn Thăng Long**

**Mã sinh viên : B16DCPT095**

**Lớp chuyên ngành : D16PTDPT**

**Khóa : 2016-2020**

**Hệ : ĐẠI HỌC CHÍNH QUY**

**HÀ NỘI 2020**

# LỜI CẢM ƠN

# MỤC LỤC

[LỜI CẢM ƠN 1](#_Toc87817526)

[MỤC LỤC 2](#_Toc87817527)

[DANH MỤC HÌNH VẼ 4](#_Toc87817528)

[LỜI MỞ ĐẦU 5](#_Toc87817529)

[CHƯƠNG 1: TỔNG QUAN VỀ GAME “REAL WORM” 6](#_Toc87817530)

[1.1 Ý tưởng 6](#_Toc87817531)

[1.2 Concept 6](#_Toc87817532)

[1.1.1 Character 6](#_Toc87817533)

[1.1.2 Control 6](#_Toc87817534)

[1.1.3 Camera 6](#_Toc87817535)

[1.1.4 Chalenge 6](#_Toc87817536)

[1.2 Pitch 6](#_Toc87817537)

[1.3 Game Design Document 7](#_Toc87817538)

[1.3.1 Character 7](#_Toc87817539)

[1.3.2 Storyline 7](#_Toc87817540)

[1.3.3 Gameplay 7](#_Toc87817541)

[1.3.4 Storyboard 10](#_Toc87817542)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 11](#_Toc87817543)

[2.1 Tổng quan về Physic Engine 11](#_Toc87817544)

[2.2 Tổng quan về trò chơi đa người chơi 13](#_Toc87817545)

[2.2.1 Khái niệm về trò chơi đa người chơi 13](#_Toc87817546)

[2.2.2 Lịch sử phát triển và phân loại trò chơi đa người chơi 13](#_Toc87817547)

[2.2.3 Cấu trúc mạng trong trò chơi đa người chơi 17](#_Toc87817548)

[2.2.4 Các yếu tố ảnh hưởng đến trò chơi đa người chơi 20](#_Toc87817549)

[CHƯƠNG 3: NGHIÊN CỨU SỬ DỤNG SOCKET 26](#_Toc87817550)

[3.1 Socket 26](#_Toc87817551)

[3.1.1 Khái niệm 26](#_Toc87817552)

[3.1.2 Phân loại 26](#_Toc87817553)

[3.1.3 Hướng sử dụng các loại Socket 28](#_Toc87817554)

[3.2 Sử dụng socket với javascript 28](#_Toc87817555)

[3.3 Cơ chế xử lý cho game Worm 29](#_Toc87817556)

[CHƯƠNG 4: PHÂN TÍCH THIẾT KẾ 29](#_Toc87817557)

[4.1 Thông số trò chơi 29](#_Toc87817558)

[4.2 Sơ đồ Use case 29](#_Toc87817559)

[4.3 Đặc tả Use case 30](#_Toc87817560)

[4.4 Kiến trúc hệ thống 33](#_Toc87817561)

[4.5 Giải quyết bài toán 34](#_Toc87817562)

[4.5.1 Cơ chế đồng bộ game state 34](#_Toc87817563)

[4.5.2 Kiến tạo bản đồ 36](#_Toc87817564)

[4.5.3 Xử lý vật lý 38](#_Toc87817565)

[4.5.4 Truyền tải bản đồ 39](#_Toc87817566)

[4.6 Biểu đồ lớp thiết kế 40](#_Toc87817567)

[4.6.1 Biểu đồ lớp truyền tài: 40](#_Toc87817568)

[PHỤ LỤC: CÀI ĐẶT VÀ TRIỂN KHAI 42](#_Toc87817569)

[Cài đặt server: 42](#_Toc87817570)

[Cài đặt client: 42](#_Toc87817571)

[KẾT LUẬN 43](#_Toc87817572)

[TÀI LIỆU THAM KHẢO 44](#_Toc87817573)

# DANH MỤC HÌNH VẼ

[Hình 2–1: Tennis for two được phát triển năm 1958 15](#_Toc87816614)

[Hình 2–2: Game FableTop dựa trên Multi-User Dungeons 16](#_Toc87816615)

[Hình 2–3: CSGO - một tựa game nổi tiếng có thể chơi thông qua mạng LAN 17](#_Toc87816616)

[Hình 2–4: World of Warcraft đã phát triển trong 14 năm qua với nhiều bước đột phá 18](#_Toc87816617)

[Hình 2–5: Các tầng trong mô hình TCP/IP 20](#_Toc87816618)

[Hình 2–6: Mô hình Client-Server 22](#_Toc87816619)

[Hình 2–7: Mô hình Peer-to-Peer 22](#_Toc87816620)

[Hình 2–8: Biểu đồ thời gian trễ 24](#_Toc87816621)

[Hình 2–9: Jitter gây ra sự sai lệch thứ tự các gói tin được gửi 25](#_Toc87816622)

[Hình 3–1: Mô hình kết nối TCP Socket 28](file:////Users/macbookpro/Downloads/GitHub/netgame/docs/Nguyễn%20Thăng%20Long%20-%20B16DCPT095%20-%20Báo%20cáo%20thực%20hành%20chuyên%20sâu.docx#_Toc87816623)

[Hình 3–2: Mô hình kết nối UDP Socket 29](file:////Users/macbookpro/Downloads/GitHub/netgame/docs/Nguyễn%20Thăng%20Long%20-%20B16DCPT095%20-%20Báo%20cáo%20thực%20hành%20chuyên%20sâu.docx#_Toc87816624)

[Hình 4–1: Sơ đồ use case 32](#_Toc87816625)

[Hình 4–2: Data lưu ở client 38](#_Toc87816626)

[Hình 4–3: Data lưu ở server 38](#_Toc87816627)

[Hình 4–4: Biểu diễn mảng bản đồ với hàm sinh số ngẫu nhiên 39](#_Toc87816628)

[Hình 4–5: Biểu diễn hình ảnh của bản đồ tạo bởi hàm sinh số ngẫu nhiên 40](#_Toc87816629)

[Hình 4–6: Biểu diễn mảng bản đồ với hàm sinh nhiễu Perlin 1D 40](#_Toc87816630)

[Hình 4–7: Biểu diễn bản đồ được sinh bởi hàm sinh nhiễu Perlin 41](#_Toc87816631)

[Hình 4–8: Trạng thái va chạm giữa đối tượng và bản đồ 42](#_Toc87816632)

[Hình 4–9: Biểu đồ lớp truyền tải 1 43](#_Toc87816633)

[Hình 4–10: Biểu đồ lớp truyền tải 2 44](#_Toc87816634)

# LỜI MỞ ĐẦU

Nội dung khoá luận được chia thành 4 chương:

CHƯƠNG 1: TỔNG QUAN VỀ GAME “REAL WORM”

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

CHƯƠNG 3: NGHIÊN CỨU VỀ SOCKET

CHƯƠNG 1: PHÂN TÍCH THIẾT KẾ

# TỔNG QUAN VỀ GAME “REAL WORM”

## Ý tưởng

Game được lấy ý tưởng từ tựa game worm trên điện thoại nổi tiếng. Người chơi có thể chơi qua mạng với nhau. Người chơi có thể dùng nhiều vũ khí và kỹ năng để giành chiến thắng.

Trò chơi hứa hẹn sẽ mang đến cho người chơi các trải nghiệm thú vị và hấp dẫn khi vượt qua được các thử thách và trùm cuối.

## Concept

### Character

Nhân vật có khả năng di chuyển bằng cách nhảy sang hai bên.

Nhân vật khả năng sử dụng vũ khí như súng, lựu đạn.

Nhân vật có thể dùng kỹ năng đặc biệt.

### Control

* Sử dụng bàn phím để thực hiện các thao tác di chuyển, ngắm bắn.
* Sử dụng chuột để thao tác

### Camera

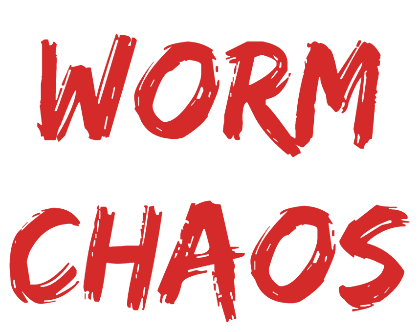
* Góc nhìn từ bên hông nhân vật giống như game platform.
* Camera tập trung vào người chơi hiện tại.

### Chalenge

* Người chơi sẽ cố gắng tiêu diệt tất cả các đối thủ khác trong bàn chơi để chiến thắng.

## Pitch

* Tên game: Worm Chaos
* Độ tuổi người chơi hướng tới: 10+
* Hệ thống thực thi trò chơi: Nền tảng Desktop
* Genre: 2D platform
* Logo trò chơi:



## Game Design Document

### Character

Nhân vật chính là các chú sâu vui vẻ. Họ bị cuốn vào một cuộc chiến sống còn. Họ phải sử dụng các vũ khí và kỹ năng của mình để giành chiến thắng.

### Storyline

### Gameplay

#### Mechanic

Các con số trong phiên bản này có thể sẽ thay đổi trong các phiên bản sau.

Trận đấu sẽ diễn ra theo lượt. Mỗi lượt có thời gian tối đa. Khi hết lượt đội này thì sẽ chuyển sang lượt của đội đối thủ.

Thứ tự chơi của mỗi người trong 1 đội là ngẫu nhiên. Nhưng vẫn đảm bảo khi tất cả người chơi trong đội chơi hết 1 lượt thì sẽ lặp lại thứ tự đó từ người đầu tiên.

Người chơi đã bị tiêu diệt sẽ được theo dõi trò chơi.

Mỗi lượt chơi người chơi sẽ được tăng 5 năng lượng.

Khi người chơi nhận sát thương thì sẽ được hồi lại một lượng năng lượng tỉ lệ với lượng sát thương nhận vào.

* Tỉ lệ hồi năng lượng: 1 năng lượng / 1 sát thương

Trong lượt người chơi có thể sử dụng năng lượng để thực hiện 1 hoặc nhiều hành động sau:

* Di chuyển: 5 px / năng lượng
* Sử dụng kỹ năng: tốn năng lượng tuỳ vào kỹ năng
* Bắn (sử dụng vũ khí): không tốn năng lượng

#### Goals

Người chơi sẽ cố gắng sử dụng vũ khí và kỹ năng để tiêu diệt toàn bộ đối thủ.

#### Unit

Trò chơi sẽ có một loại nhân vật để chơi.

* Sâu vui vẻ:
  + Máu: 100
  + Năng lượng: 100
  + Tốc độ: 20 px/s

#### Item

Mỗi người chơi trước khi vào bàn chơi sẽ được chọn 1 vũ khí (trừ máy bay giấy) để dùng cho game đấu đó, trong game sẽ không được đổi vũ khí.

* Súng phóng lựu:
  + Nổ khi va chạm
  + Bán kính nổ: 1
  + Sát thương: 10 ± 3
  + Lực bắn: 1
* Lựu đạn
  + Nổ sau 5s
  + Bán kính nổ 1
  + Sát thương: 13 ± 4
  + Lực bắn: 0.5
* Tên lửa:
  + Bán kính nổ 2.5
  + Nổ khi va chạm
  + Sát thương: 20 ± 7
  + Lực bắn: 5
* Máy bay giấy:
  + Lực bắn: 1

#### Skill

Mỗi khi vào game người chơi sẽ có thể chọn 2 skill trong danh sách skill để sử dụng trong bàn chơi.

* Mưa bom:
  + Mana: 60.
  + Chọn vị trí trên bản đồ.
  + Thả mưa bom xuống một vị trí trên bản đồ. Gây sát thương theo lượt cho người chơi trong đó.
  + Sát thương: 4/lượt
* Bất tử:
  + Mana: 60.
  + Chọn người chơi.
  + Giúp cho một người chơi không chịu sát thương từ vũ khí hoặc kỹ năng.
* Chảy máu:
  + Mana: 90.
  + Chọn người chơi.
  + Khiến cho 1 người chơi bị mất máu khi di chuyển. Di chuyển càng xa mất càng nhiều máu.
  + Sát thương: 0.2 máu/1px
* Hồi máu:
  + Năng lượng: 50
  + Chọn người chơi.
  + Hồi máu ngay lập tức cho một người chơi. Đồng thời giảm sát thương người chơi đó nhận phải trong 1 lượt.
  + Hồi máu: 30 máu.
  + Giảm sát thương: 50%.
* Dịch chuyển:
  + Năng lượng: 20
  + Được sử dụng vũ khí máy bay trong 1 lượt.
  + Dịch chuyển đến địa điểm được bắn tới.

### Storyboard

Loading

Main menu

Chơi ngay

Tìm game

Chọn phòng

Danh sách phòng

Main game

Vào phòng

Tự vào phòng

Rời phòng

# CƠ SỞ LÝ THUYẾT

## Tổng quan về Physic Engine

Nếu nói đến game không thể không nói đến tương tác vật lý tồn tại tại đa số game, tương tác vật lý áp dụng các luật vật lý khiến game có độ thực tế cao hơn. Mục đích của tương tác vật lý là sử dụng thiết bị vi tính để tái hiện lại những hiện tượng vật lý qua các công thức, tại thời điểm hiện tại máy tính cũng như đa số thiết bị điện tử đều có phần cứng khá tốt để tạo các tương tác gần giống tương đương với thế giới thực.

Với game hiện tại, đa số game studio sử dụng những engine vật lý từ phía thứ ba để phát triển game của mình, những engine phổ biến nhất hay được dung như Box2D (tồn tại trên nhiều ngôn ngữ và hệ máy, miễn phí, Angry Birds dung engine này), PhysX (engine của Nvidia, dung bởi nhiều game hành động như Batman, Assassin’s Creed), Havok (engine vật lý rất mạnh dung trong rất nhiều game cần tính toán độ va chạm chân thực nhất như Soul Calibur, Mortal Kombat, Tekken hay Skyrim) . Bài viết này nhằm mục đích giải thích và nêu ra những điều cần biết về tương tác vật lý trong game mà không đi quá sâu vào kỹ thuật, bài viết sẽ dung engine game Unity 3D để đưa ra những ví dụ cụ thể hơn, Unity 3D có engine vật lý là PhysX của NVidia(Unity 5 sử dụng PhysX ver.3).

Về cơ bản thì tương tác vật lý chia ra làm hai phần, mô phỏng rigid-body (vật cứng) và soft-body (vật mềm), tại đây chúng ta sẽ tập trung vào rigidbody. Trong mô phỏng rigid body các object được nhóm ra thành các loại khác nhau dựa trên chúng phải tương tác với nhau thế nào, thường thì loại mô phỏng này tốn ít tài nguyên hơn.

Rigid Body là hiệu ứng được mô phỏng nhiều nhất trong game, vì đa số vật mà chúng ta tương tác ngoài đời cũng tương đối cứng, và mô phỏng chúng tương đối đơn giản. Thông thường các tính toán vật lý được thực hiện bởi các engine vật lý, nó sẽ nhận được thông số tại mỗi thời điểm tương tác cộng thêm những thông số được người phát triển đưa vào trước, sau đó sự mô phỏng sẽ được thực hiện theo các bước, mỗi bước đẩy tiến độ của sự mô phỏng, chỉ vài milli giây, và kết quả sẽ được thể hiện trên màn hình. Dĩ nhiên, công việc của engine chỉ là thực hiện các phép tính, còn các bước thế nào thường dựa vào sự cần thiết của từng game.

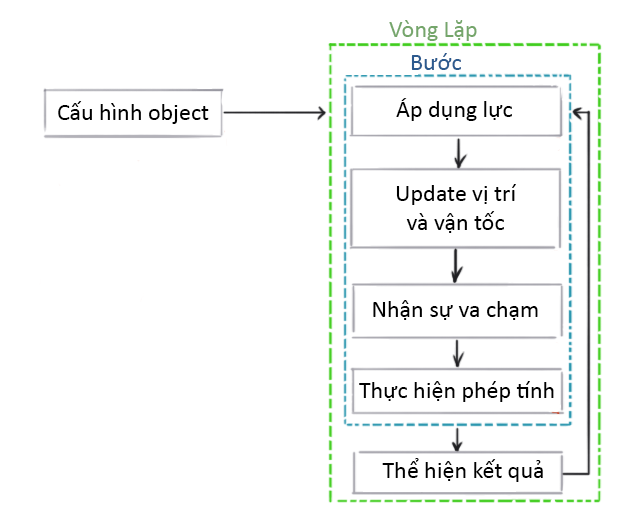
Sự chuyển động của rigid body có thể được thể hiện bởi cơ học Newton, chủ yếu được dựa trên 3 điều luật cơ học nổi tiếng của Isaac Newton:

Quán tính (Inertia) – Nếu không có lực (force) tác động lên vật, vận tốc (velocity) sẽ không thay đổi.

Lực, khối lượng và gia tốc – Lực được thực hiện trên vận tương đương với khối lượng (mass) nhân với gia tốc (acceleration). Với biểu thức là F = m\*a.

Hành động và phản ứng – Với mỗi hành động sẽ có phản ứng tương tự, nói cách khác, mỗi khi một vật tạo ra lực trên vật khác, vật này sẽ chịu lực tương tự với chiều hướng ngược lại so với lực vật kia nhận phải. Với ba định luật đó, chúng ta có thể tạo một tương tác vật lý đơn giản mà chúng ta thường biêt ngoài đời.

Vòng lặp có thể được thực hiện như sau:



Một thành phần tối quan trọng trong mỗi engine vật lý là việc phát hiện và xử lý các va chạm.

## Tổng quan về trò chơi đa người chơi

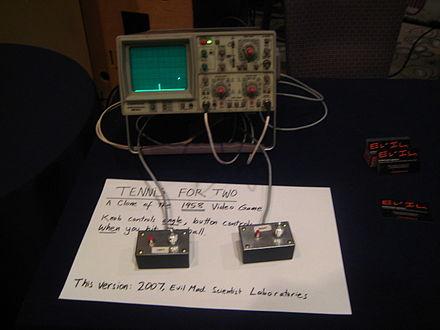
### Khái niệm về trò chơi đa người chơi

Trò chơi đa người chơi là một trò chơi trong đó nhiều người có thể chơi trong cùng một môi trường trò chơi cùng một lúc, thông qua mạng cục bộ hoặc trực tuyến. Trò chơi đa người chơi thường yêu cầu người chơi chia sẻ tài nguyên của một hệ thống trò chơi hoặc sử dụng công nghệ mạng để chơi với nhau trên một khoảng cách lớn hơn. Người chơi có thể cạnh tranh với một hoặc nhiều người chơi khác, hoặc cùng hợp tác để đạt được mục tiêu chung.

### Lịch sử phát triển và phân loại trò chơi đa người chơi

#### Local Multiplayer Games

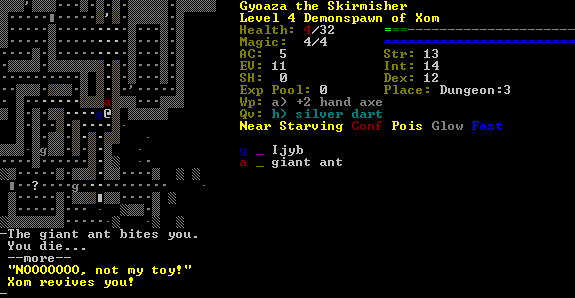
Local Multiplayer Games là những game được thiết kế cho hai hoặc nhiều người chơi trên cùng một máy tính. Đối với hầu hết các phần, local multiplayer games có thể được lập trình theo cách tương tự như các trò chơi một người chơi. Sự khác biệt duy nhất thường là các viewpoints và / hoặc hỗ trợ nhiều thiết bị đầu vào.



Hình 2–1: Tennis for two được phát triển năm 1958

#### Multi-User Dungeons

Multi-user dungeon hoặc MUD là một trò chơi nhiều người chơi (thường dựa trên văn bản), trong đó một số người chơi được kết nối với cùng một thế giới ảo cùng một lúc. Loại trò chơi này lần đầu tiên trở nên phổ biến tại các trường đại học lớn và thuật ngữ này bắt nguồn từ trò chơi MUD (1978), được tạo ra bởi Rob Trushaw tại Đại học Essex. Theo một số cách, MUD có thể được coi là phiên bản máy tính đầu tiên của trò chơi nhập vai Dungeons và Dragons, mặc dù không phải tất cả các MUD đều là game nhập vai.



Hình 2–2: Game FableTop dựa trên Multi-User Dungeons

Khi máy tính cá nhân trở nên mạnh hơn, các nhà sản xuất phần cứng bắt đầu cung cấp các modem cho phép hai máy tính giao tiếp với nhau qua các đường dây điện thoại tiêu chuẩn. Mặc dù tốc độ truyền tải cực kỳ chậm theo tiêu chuẩn hiện đại, điều này cho phép MUD được phát bên ngoài trường đại học. Một số trò chơi MUD đã chạy trên hệ thống bảng thông báo (bulletin board system), cho phép nhiều người dùng kết nối qua modem với một hệ thống có thể chạy nhiều thứ, kể cả game

#### Local Area Network Games

Mạng cục bộ hoặc mạng LAN là một thuật ngữ được sử dụng để mô tả một số máy tính được kết nối với nhau trong một khu vực tương đối nhỏ.



Hình 2–3: CSGO - một tựa game nổi tiếng có thể chơi thông qua mạng LAN

Nhiều trò chơi hỗ trợ nhiều người chơi được nối mạng qua mạng LAN cũng hỗ trợ nhiều người chơi được nối mạng theo những cách khác, cho dù bằng kết nối modem hoặc mạng trực tuyến. Trong nhiều năm, phần lớn các trò chơi nối mạng cũng hỗ trợ chơi trò chơi trên mạng LAN. Điều này dẫn đến sự gia tăng của các bên LAN nơi mọi người sẽ gặp nhau tại một địa điểm và kết nối máy tính của họ để chơi các trò chơi nối mạng. Mặc dù một số trò chơi nhiều người chơi được nối mạng vẫn được phát hành với chơi LAN, nhưng xu hướng trong những năm gần đây dường như có các nhà phát triển từ bỏ việc chơi LAN cho nhiều người chơi trực tuyến độc quyền.

#### Online Games

Khi Internet bắt đầu bùng nổ vào cuối những năm 1990, các trò chơi trực tuyến đã diễn ra cùng với nó. Một số trò chơi phổ biến trong những năm đầu bao gồm Software Quake (1996) và Epic Game Rush Unreal (1998).

Trong một trò chơi trực tuyến, người chơi kết nối với nhau qua một số mạng lớn với các máy tính ở xa về mặt địa lý. Mặc dù một trò chơi trực tuyến được triển khai theo cách tương tự như trò chơi LAN, một điều cần cân nhắc chính là độ trễ hoặc thời gian để dữ liệu truyền qua mạng.

#### Massively Multiplayer Online Games

Ngày nay, hầu hết các trò chơi nhiều người chơi trực tuyến chỉ giới hạn ở một số lượng nhỏ người chơi trên mỗi phiên trò chơi, ở đâu đó từ 4 đến 32 thường là số lượng người chơi được hỗ trợ. Tuy nhiên, trong Trò chơi trực tuyến nhiều người chơi (MMO), hàng trăm nếu không phải hàng ngàn người chơi có thể tham gia vào một phiên trò chơi duy nhất. Hầu hết các game MMO là game nhập vai và do đó được gọi là MMORPG. Tuy nhiên, chắc chắn có những phong cách khác của game MMO như game bắn súng góc nhìn thứ nhất (MMOFPS).

Một tựa game MMORPG vô cùng thành công và đã gây bão trên toàn thế giới đó là World of Warcraft ra mắt năm 2004. Tại một thời điểm, MMORPG Blizzard đã có hơn 12 triệu người đăng ký hoạt động trên toàn thế giới và trò chơi đã trở thành một phần lớn của văn hóa phổ biến đến mức nó được xuất hiện trong một tập phim hoạt hình South Park năm 2006.



Hình 2–4: World of Warcraft đã phát triển trong 14 năm qua với nhiều bước đột phá

#### Mobile Networked Games

Trò chơi di động là trò chơi được chơi trên điện thoại di động. Nhiều trò chơi di động hỗ trợ nhiều người chơi thông qua qua mạng cục bộ hoặc thông qua Wi-Fi, Bluetooth hoặc công nghệ tương tự.

Hai cơ chế chính của trò chơi nhiều người chơi qua di động là cơ chế bất đồng bộ: các trò chơi theo lượt không yêu cầu truyền dữ liệu theo thời gian thực và cơ chế đồng bộ thời gian thực: người chơi ngẫu nhiên từ khắp nơi chơi với nhau.

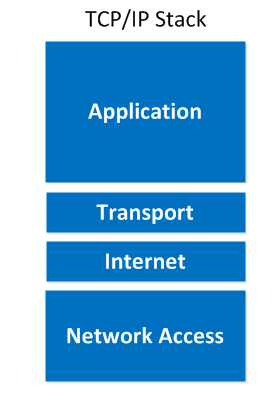
Từ quan điểm kỹ thuật, một trò chơi nối mạng không đồng bộ sẽ dễ thực hiện hơn so với một thời gian thực một. Điều này đặc biệt đúng trên các nền tảng di động. Tuy nhiên, với sự phát triển của các thiết bị di động, mạng internet các trò chơi kết nối mạng thời gian thực xuất hiện trên các thiết bị này.

### Cấu trúc mạng trong trò chơi đa người chơi

#### Mô hình TCP/IP

TCP/IP (Transmission Control Protocol/ Internet Protocol - Giao thức điều khiển truyền nhận/ Giao thức liên mạng), là một bộ giao thức trao đổi thông tin được sử dụng để truyền tải và kết nối các thiết bị trong mạng Internet. TCP/IP được phát triển để mạng được tin cậy hơn cùng với khả năng phục hồi tự động.

Đối với một lập trình viên trò chơi nhiều người chơi, hiểu và tận dụng được mô hình TCP/IP sẽ giúp cho trò chơi hoạt động hiệu quả hơn. Thông thường, các lập trình viên sẽ chỉ làm việc với lớp cao nhất của mô hình (tầng ứng dụng), nhưng để làm điều đó một cách hiệu quả, sẽ rất hữu ích khi hiểu các lớp bên dưới và cách chúng ảnh hưởng đến các lớp bên trên chúng.

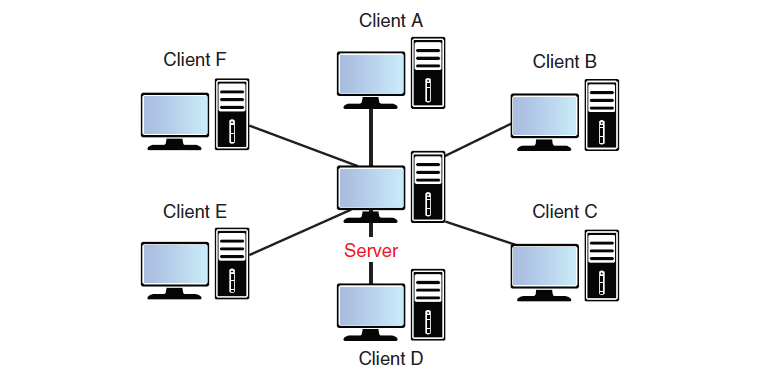


Hình 2–5: Các tầng trong mô hình TCP/IP

* *Tầng vật lý (Network Access):* 
  + Tầng vật lý là sự kết hợp của tầng Data Link và Physical trong mô hình [OSI](https://en.wikipedia.org/wiki/OSI_model), là tầng thấp nhất trong mô hình TCP/IP, chịu trách nhiệm truyền dữ liệu giữa các thiết bị trong cùng một mạng. Tại đây, các gói dữ liệu được đóng vào khung (Frame) và được định tuyến đi đến đích được chỉ định ban đầu.
* *Tầng mạng (Internet)*
  + Gần giống như tầng mạng của mô hình OSI. Tại đây, nó cũng được định nghĩa là một giao thức chịu trách nhiệm truyền tải dữ liệu một cách logic trong mạng. Các phân đoạn dữ liệu sẽ được đóng gói (Packets) với kích thước mỗi gói phù hợp với mạng chuyển mạch mà nó dùng để truyền dữ liệu. Lúc này, các gói tin được chèn thêm phần Header chứa thông tin của tầng mạng và tiếp tục được chuyển đến tầng tiếp theo.
* *Tầng giao vận (Transport)*
  + Chức năng chính của tầng giao vận là xử lý vấn đề giao tiếp giữa các máy chủ trong cùng một mạng hoặc khác mạng được kết nối với nhau thông qua bộ định tuyến. Tại đây dữ liệu sẽ được phân đoạn, mỗi đoạn sẽ không bằng nhau nhưng kích thước phải nhỏ hơn 64KB. Cấu trúc đầy đủ của một Segment lúc này là Header chứa thông tin điều khiển và sau đó là dữ liệu.
  + Trong tầng này còn bao gồm 2 giao thức cốt lõi là TCP và UDP. TCP và UDP là hai giao thức được sử dụng rộng rãi nhất trong các trò chơi nối mạng nhiều người chơi. Cả hai giao thức đều cung cấp dịch vụ liên lạc giữa các máy khách trong một hệ thống mạng. Nói một cách đơn giản, chúng là các giao thức cho phép chúng ta gửi và nhận các gói dữ liệu theo cách mà dữ liệu có thể được xác định và diễn giải theo cách có thể dự đoán được. Ưu nhược điểm cả 2 giao thức sẽ được nói rõ trong phần 1.6.3
* *Tầng ứng dụng (Application)*
  + Ở trên cùng của lớp bánh TCP/IP là tầng ứng dụng và đây là nơi mã trò chơi nhiều người chơi tồn tại. Tầng ứng dụng cũng là nơi có nhiều giao thức cơ bản của Internet. Dữ liệu khi đến đây sẽ được định dạng theo kiểu Byte nối Byte, cùng với đó là các thông tin định tuyến giúp xác định đường đi đúng của một gói tin.

#### Mô hình Client-Server

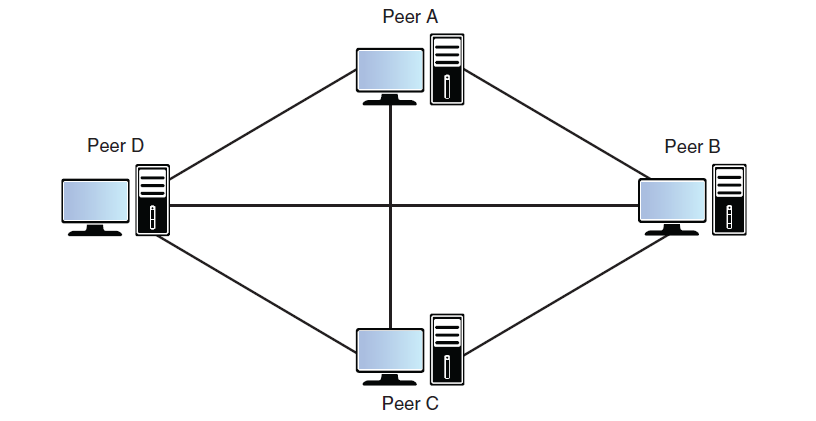
Client/server network là một loại mô hình mạng máy tính bao gồm 02 thành phần là máy chủ và máy khách. Mỗi máy khách chỉ giao tiếp với máy chủ, trong khi máy chủ chịu trách nhiệm liên lạc với tất cả các máy khách.



Hình 2–6: Mô hình Client-Server

#### Mô hình Peer-to-Peer

Trong mô hình liên kết ngang hàng, mỗi cá nhân tham gia được kết nối với mọi người tham gia khác, tất cả đều tiêu thụ cùng dữ liệu và chia sẻ bất kỳ dữ liệu nào họ tạo ra để những người khác có thể duy trì đồng bộ hóa.



Hình 2–7: Mô hình Peer-to-Peer

### Các yếu tố ảnh hưởng đến trò chơi đa người chơi

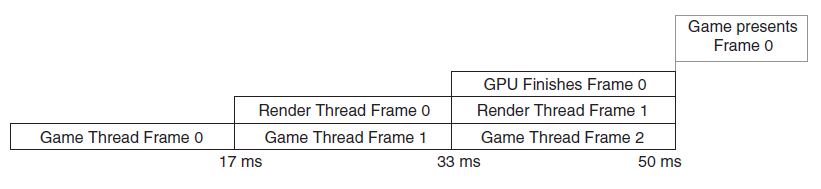
Các trò chơi đa ngườ chơi kết nối mạng thường phải đối mặt với rất nhiều vấn đề như: cạnh tranh băng thông trên các mạng cũ, gửi các gói đến máy chủ và máy khách rải rác trên khắp thế giới. Điều này dẫn đến mất dữ liệu và sự chậm trễ thường sẽ không sảy ra trong quá trình phát triển trò chơi trên một mạng cục bộ.

#### Độ trễ (Latency)

Yếu tố đầu tiên trong số này là độ trễ. Độ trễ từ có ý nghĩa khác nhau trong các tình huống khác nhau. Trong ngữ cảnh của trò chơi đa người chơi, độ trễ đề cập đến lượng thời. cần thiết để một gói thông tin truyền từ máy tính này sang máy tính khác. Mặc dù độ trễ không quan trọng đối với các trò chơi theo lượt, nhưng các trò chơi dựa trên hành động, thời gian thực luôn đòi hỏi phải có độ chính xác cao.

Một quan niệm sai lầm thường thấy rằng sự chậm trễ mạng là nguồn chính cho độ trễ trong trò chơi. Mặc dù trao đổi gói qua mạng là một nguồn đáng kể cho độ trễ, nhưng nó chắc chắn không phải là nguồn duy nhất. Có vài nguồn độ trễ khác, một số nguồn không thuộc quyền kiểm soát của bạn:

* Độ trễ từ thiết bị đầu vào: Khoảng thời gian giữa khi người dùng nhấn nút và khi trò chơi phát hiện ra rằng nhấn nút có thể là đáng kể.
* Độ trễ trong quá trình chiết xuất: GPU không thực hiện các lệnh vẽ trong thời điểm CPU xử lý các lệnh đó. Thay vào đó, trình điều khiển(driver) chèn các lệnh vào bộ đệm lệnh và GPU sẽ thực thi các lệnh đó tại một thời điểm nào đó trong tương lai. Nếu có rất nhiều kết xuất phải làm, GPU có thể lag toàn bộ khung phía sau CPU trước khi nó hiển thị hình ảnh được hiển thị cho người dùng.
* Độ trễ trong quá trình chiết xuất đa luồng: Các trò chơi đa luồng có độ trễ thậm chí nhiều hơn trong quá trình kết xuất. Thông thường, một hoặc nhiều luồng chạy mô phỏng trò chơi, cập nhật một thế giới mà họ chuyển đến một hoặc nhiều luồng kết xuất. Các luồng kết xuất này sau đó xử lý các lệnh GPU trong khi các luồng mô phỏng đã mô phỏng khung tiếp theo. Hình 1.8 trình bày cách kết xuất đa luồng có thể thêm một khung độ trễ khác cho trải nghiệm người dùng.



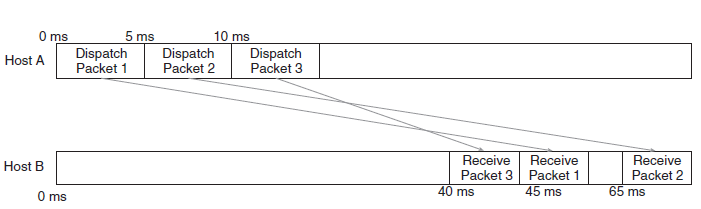
Hình 2–8: Biểu đồ thời gian trễ

* Mặc dù có nhiều nguồn độ trễ, độ trễ của gói tin khi nó di chuyển từ máy chủ nguồn đến đích thường là nguyên nhân quan trọng nhất gây ra độ trễ trong chơi game nhiều người chơi. Có bốn độ trễ chính mà một trải nghiệm gói trong suốt vòng đời của nó:
* Xử lý chậm trễ (Processing delay): hãy nhớ rằng bộ định tuyến mạng hoạt động bằng cách đọc các gói từ giao diện mạng, kiểm tra địa chỉ IP đích, tìm ra máy tiếp theo sẽ nhận gói và sau đó chuyển tiếp ra khỏi giao diện thích hợp. Thời gian dành cho việc kiểm tra địa chỉ nguồn và xác định tuyến thích hợp được gọi là độ trễ xử lý. Xử lý độ trễ cũng có thể bao gồm bất kỳ chức năng bổ sung nào mà bộ định tuyến cung cấp, chẳng hạn như NAT hoặc mã hóa.
* Độ trễ truyền (Transmission delay): Để bộ định tuyến chuyển tiếp gói, nó phải có giao diện lớp liên kết cho phép nó chuyển tiếp gói dọc theo một số phương tiện vật lý. Giao thức lớp liên kết kiểm soát tốc độ trung bình mà tại đó các bit có thể được ghi vào phương tiện. Chẳng hạn, 1 MB Kết nối Ethernet cho phép ghi khoảng 1 triệu bit vào cáp Ethernet mỗi giây. Do đó, phải mất khoảng một phần triệu giây (1 μs) để ghi một chút vào cáp Ethernet 1 MB, và do đó 12,5 ms để ghi toàn bộ gói 1500 byte. Thời gian này dành cho việc ghi các bit vào phương tiện vật lý được gọi là độ trễ truyền
* Độ trễ hàng đợi (Queuing delay): Một bộ định tuyến chỉ có thể xử lý một số lượng hạn chế các gói tại một thời điểm. Nếu các gói đến với tốc độ nhanh hơn bộ định tuyến có thể xử lý chúng, chúng sẽ đi vào hàng đợi nhận, chờ xử lý. Tương tự, giao diện mạng chỉ có thể xuất một gói một lần, vì vậy sau khi gói được xử lý, nếu giao diện mạng thích hợp bị bận, nó sẽ đi vào hàng đợi truyền. Thời gian dành cho các hàng đợi này được gọi là độ trễ hàng đợi.
* Độ trễ lan truyền (Propagation delay): Đối với hầu hết các phần, bất kể môi trường vật lý, thông tin không thể truyền nhanh hơn tốc độ ánh sáng. Do đó, độ trễ khi gửi gói ít nhất là 0,3 lần so với số mét mà gói phải đi. Điều này có nghĩa, ngay cả trong điều kiện lý tưởng, phải mất ít nhất 12 ms cho một gói để đi qua nước Mỹ. Thời gian này được gọi là sự chậm trễ lan truyền.

#### Jitter

Trong trò chơi trực tuyến, jitter được định nghĩa là tốc độ ping thay đổi trong một thời gian nhất định. Một kết nối có tỷ lệ jitter cao có thể gây ra sự khó khăn trong kết nối và gameplay. Thông thường, các game thủ sẽ tìm kiếm một kết nối có ping ổn định và vững chắc, điều này đặc biệt quan trọng trong các trò chơi có yếu tố co giật cao, chẳng hạn game PlayerUnknown's Battlegrounds hoặc trong các trận đấu đồng đội trong Liên minh huyền thoại. Giá trị jitter cho kết nối càng thấp thì càng tốt.

Jitter cũng gây ra sự sai lệch thứ tự các gói tin được gửi. Hình 1.9 minh họa làm thế nào điều này có thể xảy ra. Máy chủ A gửi Gói 1, Gói 2 và Gói 3, theo thứ tự, cách nhau 5 ms, bị ràng buộc với Máy chủ từ xa B. Gói 1 mất 45 ms để đến Máy chủ B, nhưng do lưu lượng truy cập đột ngột trên tuyến, Gói 2 mất 60 ms đến được Host B. Ngay sau khi có lưu lượng truy cập, các bộ định tuyến sẽ tự động điều chỉnh tuyến đường khiến Gói 3 chỉ mất 30 ms để đến Máy chủ B. Điều này dẫn đến Máy chủ B nhận Gói 3 trước, sau đó đến Gói 1, rồi Gói 2.



Hình 2–9: Jitter gây ra sự sai lệch thứ tự các gói tin được gửi

#### Độ tin cậy (Reliability)

Do nhu cầu về một số mức độ tin cậy của mạng trong hầu hết mọi trò chơi nhiều người chơi, một quyết định quan trọng cần đưa ra sớm trong quá trình phát triển là giữa TCP và UDP. Trò chơi của bạn nên dựa vào hệ thống độ tin cậy hiện có được tích hợp trong TCP hay bạn nên cố gắng phát triển hệ thống độ tin cậy tùy chỉnh của riêng bạn trên UDP? Để trả lời câu hỏi này, cần xem xét lợi ích và chi phí của mỗi giao thức vận chuyển.

Ưu điểm chính của TCP là nó cung cấp độ tin cậy được kiểm tra theo thời gian, mạnh mẽ và ổn định. Không cần nỗ lực kỹ thuật thêm, nó đảm bảo tất cả dữ liệu sẽ không chỉ được gửi mà còn được gửi theo thứ tự. Ngoài ra, nó cung cấp các tính năng kiểm soát tắc nghẽn phức tạp nhằm hạn chế mất gói bằng cách gửi dữ liệu ở tốc độ không áp đảo các bộ định tuyến trung gian.

Hạn chế lớn của TCP là mọi thứ nó gửi phải được gửi một cách đáng tin cậy và được xử lý theo thứ tự. Trong một trò chơi nhiều người chơi với trạng thái thay đổi nhanh chóng, có ba kịch bản khác nhau trong đó việc truyền tải đáng tin cậy, phổ biến bắt buộc này có thể gây ra sự cố:

* Mất dữ liệu ưu tiên thấp gây cản trở việc nhận dữ liệu ưu tiên cao: Ví dụ có 2 gói tin được gửi theo thứ tự cho người chơi A. Gói thứ nhất có độ ưu tiên thấp, gói thứ 2 có độ ưu tiên cao. Do lưu lượng truy cập mạng bị biến động, gói đầu tiên bị hủy, nhưng gói thứ hai thì không. Tuy nhiên, vì TCP xử lý tất cả các gói theo thứ tự, mô-đun TCP sẽ không chuyển gói chuyển động đến trò chơi khi nhận được. Thay vào đó, nó sẽ đợi cho đến khi máy chủ truyền lại gói bị rơi có mức độ ưu tiên thấp hơn trước khi cho phép ứng dụng xử lý gói chuyển động có mức độ ưu tiên cao. Điều này gây ra những trải nghiệm không tốt cho người chơi
* Hai luồng dữ liệu được sắp xếp đáng tin cậy giao thoa với nhau: Ngay cả trong một trò chơi không có dữ liệu ưu tiên thấp, trong đó tất cả dữ liệu phải được truyền một cách đáng tin cậy, hệ thống thứ tự của TCP vẫn có thể gây ra sự cố.
* Truyền lại trạng thái trò chơi cũ: Hãy tưởng tượng Người chơi B chạy khắp bản đồ để bắn Người chơi A. B bắt đầu ở vị trí x = 0 và trong vòng 5 giây, chạy đến vị trí x = 100. Máy chủ gửi các gói đến Người chơi A năm lần mỗi giây chứa tọa độ x mới nhất của vị trí Người chơi B. Nếu máy chủ phát hiện ra rằng bất kỳ hoặc tất cả các gói đó bị hủy, nó sẽ gửi lại chúng. Điều này có nghĩa là trong khi Người chơi B đang tiếp cận vị trí cuối cùng của mình là x = 100, thì máy chủ có thể đang truyền lại dữ liệu trạng thái cũ có Người chơi B gần hơn với x = 0. Điều này dẫn đến Người chơi A đang xem vị trí rất cũ của Người chơi B và nhận được bắn trước khi nhận được bất kỳ thông tin nào mà Người chơi B ở gần đó. Đây không phải là một trải nghiệm chấp nhận được cho Người chơi A.

Cuối cùng, TCP phân bổ rất nhiều tài nguyên để quản lý các kết nối và theo dõi tất cả dữ liệu có thể phải được gửi lại. Các phân bổ này thường được quản lý bởi hệ điều hành và có thể khó theo dõi hoặc định tuyến thông qua trình quản lý bộ nhớ tùy chỉnh nếu muốn.

Mặt khác, UDP không cung cấp độ tin cậy và kiểm soát luồng tích hợp mà TCP cung cấp. Tuy nhiên, nó hiện một khung vẽ trống mà bạn có thể vẽ bất kỳ loại hệ thống độ tin cậy tùy chỉnh nào mà trò chơi của bạn yêu cầu. Bạn có thể cho phép gửi dữ liệu đáng tin cậy và không đáng tin cậy hoặc đan xen các luồng dữ liệu đáng tin cậy được sắp xếp riêng biệt. Bạn cũng có thể xây dựng một hệ thống chỉ gửi thông tin mới nhất khi thay thế các gói bị mất, thay vì truyền lại dữ liệu chính xác đã bị mất. Bạn có thể tự quản lý bộ nhớ của mình và kiểm soát chi tiết về cách dữ liệu được nhóm vào các gói lớp mạng.

Trong hầu hết các trường hợp, sự lựa chọn sử dụng giao thức vận chuyển nào sẽ thuộc về câu hỏi này: Có phải mọi phần dữ liệu mà trò chơi gửi đi đều cần được nhận và có cần xử lý theo cách hoàn toàn không? Nếu câu trả lời là có, bạn nên cân nhắc sử dụng TCP. Điều này thường đúng với các trò chơi theo lượt. Mỗi phần đầu vào phải được nhận bởi mọi máy chủ và được xử lý theo cùng một thứ tự, vì vậy TCP là sự phù hợp hoàn hảo. Nếu TCP không phải là sự phù hợp hoàn hảo tuyệt đối cho trò chơi của bạn và đối với hầu hết các trò chơi, bạn nên sử dụng UDP với một hệ thống độ tin trên nền UPD. Điều này có nghĩa là sử dụng một bên thứ ba giải pháp trung gian hoặc xây dựng một hệ thống tùy chỉnh.

# NGHIÊN CỨU SỬ DỤNG SOCKET

## Socket

### Khái niệm

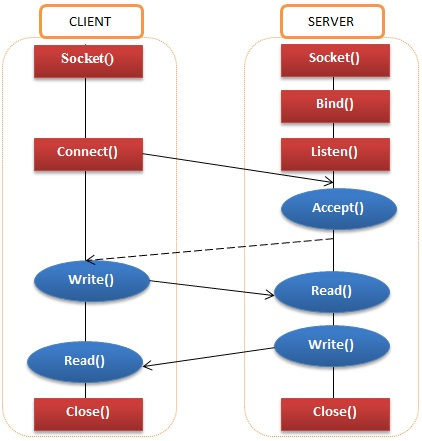
- Socket là một điểm cuối (endpoint) của liên kết giao tiếp hai chiều (two-way communication) giữa hai chương trình chạy trên mạng. Nghĩa là một socket được sử dụng để cho phép 1 process nói chuyện với 1 process khác.

- Socket trong mạng là 1 định danh để 2 máy tính có thể kết nối với nhau.

- Socket trong TCP/IP biểu diễn bởi 1 cặp (pair) bao gồm host và port. 2 socket khác nhau sẽ có host và port khác nhau.

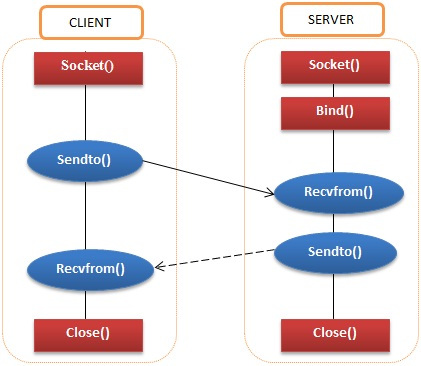
### Phân loại

* Stream socket: Hay còn gọi là TCP Socket.
  + Dựa trên giao thức TCP (Transmission Control Protocol), stream socket thiết lập giao tiếp 2 chiều theo mô hình client và server. Được gọi là socket hướng kết nối.



Hình 3–1: Mô hình kết nối TCP Socket

* + Các gói tin gửi đi sẽ có xác nhận khi bên nhận nhận được gói tin. Các gói tin không có xác nhận sẽ được gửi lại.
  + TCP socket đảm bảo dữ liệu được truyền đến nơi nhận một cách đáng tin cậy: Mọi gói tin đều được kiểm tra tính đúng đắn, nếu gói tin bị sai lệch thì sẽ được gửi lại. Do đó mọi gói tin khi truyền tới nơi, khi được xử lý bởi tầng Application đều đảm bảo chính xác.
  + TCP Socket đảm bảo các gói tin được truyền đến đúng tuần tự nhờ vào cơ chế quản lý luồng lưu thông trên mạng và cơ chế chống tắc nghẽn. Do đó khi có 2 gói tin thì gói tin gửi trước sẽ đảm bảo được nhận trước, và gói tin gửi sau sẽ chắc chắn được nhận sau.
  + TCP Socket sử dụng giao thức phức tạp, do đó gói tin TCP sẽ lớn (header của mỗi gói tin TCP Socket là 20 Byte).
* Datagram socket: Hay còn gọi là UDP Socket
  + Dựa trên giao thức UDP (User Datagram Protocol) việc truyền dữ liệu không yêu cầu có sự thiết lập kết nối giữa 2 process. Tức là nó cung cấp điểm cuối không kết nối cho việc gửi và nhận packets. Gọi là socket không hướng kết nối.



Hình 3–2: Mô hình kết nối UDP Socket

* + Các gói tin gửi đi sẽ không có xác nhận từ phía nhận. Do đó bên gửi sẽ không biết liệu bên nhận có nhận được gói tin hay không.
  + UDP socket đảm bảo dữ liệu được truyền đến nơi nhận (nếu gói tin có thể truyền đến nơi) một cách đáng tin cậy. Mọi gói tin đều được kiểm tra tính đúng đắn, nếu gói tin bị sai lệch thì sẽ bị bỏ qua (drop). Do đó mọi gói tin khi truyền tới nơi, khi được xử lý bởi tầng Application đều đảm bảo chính xác.
  + Thứ tự các gói tin khi truyền đến nơi không đảm bảo giống với thứ tự các gói tin được gửi đi. Ví dụ 2 gói tin được gửi đi thì gói tin được gửi sau có thể đến nơi trước gói tin gửi trước.
  + UDP Socket sử dụng giao thức đơn giản, do đó gói tin UDP thường khá nhỏ (header của mỗi gói tin UDP Socket là 8 Byte).
* Websocket:
  + Dựa trên giao thức TCP, hỗ trợ việc lập trình bằng javascript và có thể chạy trên trình duyệt.
  + Kế thừa các tính năng của TCP Socket.
  + Có API theo hướng sự kiện.

### Hướng sử dụng các loại Socket

* TCP Socket có ưu điểm về độ tin cậy của gói tin, nhưng tốc độ truyền tải sẽ chậm hơn UDP Socket và độ trễ sẽ lớn hơn. Do đó TCP Socket sẽ phù hợp với những nghiệp vụ không yêu cầu tốc độ truyền tải cao, ví dụ như việc cập nhật thông tin người chơi.
* UDP Socket có ưu điểm tốc độ cao, độ trễ thấp. Do đó sẽ phù hợp với các nghiệp vụ yêu cầu tốc độ truyền tải cao, độ trễ thấp, như cập nhật game state.

## Sử dụng socket với javascript

* Javascript được dùng ở 2 môi trường là trình duyệt và nodejs.
  + Môi trường trình duyệt không hỗ trợ làm việc trực tiếp với TCP Socket và UDP Socket mà chỉ hỗ trợ Websocket.
  + Môi trường nodejs hỗ trợ tất cả các loại Socket, với TCP Socket nằm ở module Net, UDP Socket nằm ở module Dgram và Websocket được nhiều thư viện implement.

## Cơ chế xử lý cho game Worm

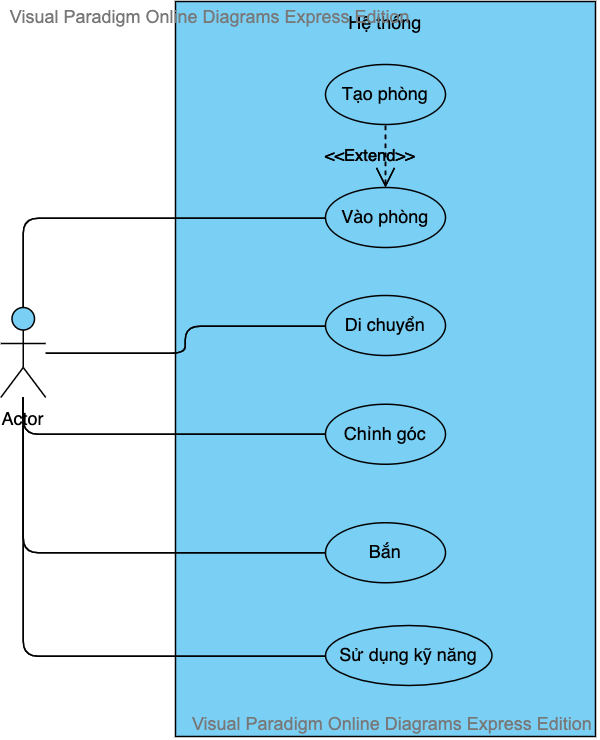
* Game sẽ sử dụng kiến trúc client-server để kết nối.
* Game sẽ chia ra 2 giai đoạn:
  + Giai đoạn 1: Ở ngoài lobby, gửi nhận các thông số về người chơi, phòng chơi, game sử dụng TCP Socket.
  + Giai đoạn 2: Ở trong game chính, gửi nhận game state, game sử dụng UDP Socket. Game sẽ huỷ kết nối TCP Socket ở giai đoạn này, do không sử dụng đến.

# PHÂN TÍCH THIẾT KẾ

## Thông số trò chơi

* Tên game: Worm chaos
* Độ tuổi: Mọi người
* Thể loại: Casual Game
* Platform: PC.
* Tham số ESRB: E (Everyone).

## Sơ đồ Use case



Hình 4–1: Sơ đồ use case

## Đặc tả Use case

|  |  |
| --- | --- |
| Tên | Di chuyển |
| Tác nhân | Người chơi |
| Tiền điều kiện | Đã đến lượt người chơi, người chơi vẫn còn thể lực. |
| Đảm bảo tối thiểu | Nhân vật thực hiện hành động di chuyển. |
| Đảm bảo thành công | Tất cả người chơi trong phòng đều thấy được hành động di chuyển của nhân vật tương ứng. |
| Kích hoạt | Người chơi nhấn phím di chuyển. |
| Chuỗi sự kiện:   1. Người chơi nhấn phím di chuyển trái hoặc phải. 2. Hệ thống di chuyển người chơi sang hướng tương ứng. Đồng thời truyền thông tin đến các người chơi khác trong cùng phòng. | |
| Ngoại lệ | |

|  |  |
| --- | --- |
| Tên | Chỉnh góc |
| Tác nhân | Người chơi |
| Tiền điều kiện | Đã đến lượt người chơi. |
| Đảm bảo tối thiểu | Góc bắn của nhân vật được thay đổi. |
| Đảm bảo thành công | Tất cả người chơi trong phòng đều thấy góc bắn của nhân vật được thay đổi. |
| Kích hoạt | Người chơi nhấn phím chỉnh góc |
| Chuỗi sự kiện:   1. Người chơi nhấn phím chỉnh góc (phím mũi tên lên hoặc xuống). 2. Hệ thống chỉnh góc tăng, giảm tương ứng. Đồng thời truyền thông tin đến các người chơi khác trong cùng phòng. | |
| Ngoại lệ | |

|  |  |
| --- | --- |
| Tên | Bắn |
| Tác nhân | Người chơi |
| Tiền điều kiện | Đã đến lượt người chơi. |
| Đảm bảo tối thiểu | Nhân vật bắn ra đạn tương ứng với vũ khí. |
| Đảm bảo thành công | Tất cả người chơi trong phòng đều thấy nhân vật bắn. |
| Kích hoạt | Người chơi nhấn phím bắn |
| Chuỗi sự kiện:   1. Người chơi nhấn và giữ phím bắn (space bar) để tăng lực bắn, sau đó thả phím bắn. 2. Hệ thống thực hiện bắn ra đạn. Đồng thời truyền thông tin đến các người chơi khác trong cùng phòng. | |
| Ngoại lệ | |

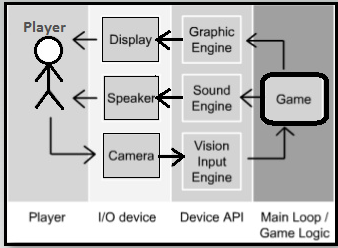
|  |  |
| --- | --- |
| Tên | Sử dụng kỹ năng |
| Tác nhân | Người chơi |
| Tiền điều kiện | Đã đến lượt người chơi, người chơi vẫn còn thể lực. |
| Đảm bảo tối thiểu | Kỹ năng tương ứng được thực hiện. |
| Đảm bảo thành công | Tất cả người chơi trong phòng đều thấy được kỹ năng tương ứng được thực hiện. |
| Kích hoạt | Người chơi nhấn phím tương ứng với kỹ năng hoặc click vào icon kỹ năng |
| Chuỗi sự kiện:   1. Người chơi nhấn phím tương ứng với kỹ năng, hoặc sử dụng chuột để click vào icon kỹ năng. Sau đó chọn mục tiêu cho kỹ năng bằng chuột. 2. Hệ thống thực hiện kỹ năng tương ứng. | |
| Ngoại lệ | |

|  |  |
| --- | --- |
| Tên | Vào phòng |
| Tác nhân | Người chơi |
| Tiền điều kiện | Người chơi muốn vào chơi và có phòng chơi không đầy đang chờ. |
| Đảm bảo tối thiểu | Người chơi vào được phòng chơi |
| Đảm bảo thành công | Tạo được phòng mới |
| Kích hoạt | Có người chơi muốn vào chơi nhưng tất cả các phòng đều đã đầy hoặc đang chơi. |
| Chuỗi sự kiện:   1. Người chơi nhấn vào nút “vào chơi” 2. Hệ thống tạo phòng chơi mới. | |
| Ngoại lệ | |

|  |  |
| --- | --- |
| Tên | Tạo phòng |
| Tác nhân | Hệ thống |
| Tiền điều kiện | Có người chơi muốn vào chơi nhưng tất cả các phòng đều đã đầy hoặc đang chơi. |
| Đảm bảo tối thiểu | Tạo được phòng mới |
| Đảm bảo thành công | Tạo được phòng mới |
| Kích hoạt | Có người chơi muốn vào chơi nhưng tất cả các phòng đều đã đầy hoặc đang chơi. |
| Chuỗi sự kiện:   1. Người chơi nhấn vào nút “vào chơi” 2. Hệ thống tạo phòng chơi mới. | |
| Ngoại lệ | |

## Kiến trúc hệ thống

Game sử dụng lập trình theo hướng đối tượng, kiến trúc game, thiết kế theo các module 3 lớp và kiến trúc client-server với fat client.



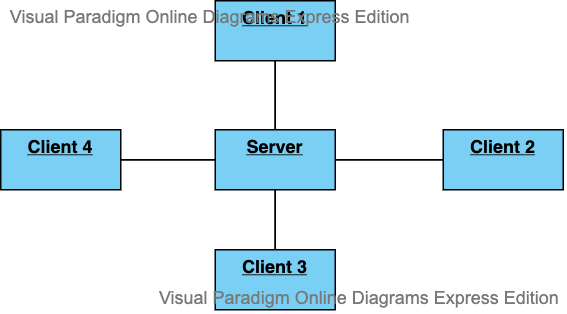
*Hình 4-12: Kiến trúc game*

Cocos2d-x đã xử lý phần I/O device và Device API cho chúng ta có thể tập trung vào phần logic của game.

Logic của game sẽ được thiết kế theo các module. Mỗi module sẽ được chia thành 2 thành phần: logic và hiển thị.

Với kiến trúc client-server với fat client, client sẽ xử lý hầu hết logic của game.

Server được sử dụng như một trung tâm để phát sóng lại các hành động của client. Do đó client sẽ chỉ cần kết nối đến một server và hoạt động như thể các client khác cũng đang được xử lý trên cùng một thiết bị.



*Hình 4-13: Mô hình hoạt động của game*

Khi một client có hành động như di chuyển, bắn, sử dụng kỹ năng, hành động đó sẽ được gửi lên server và server sẽ gửi lại toàn bộ tin nhắn đó cho các client khác đang trong cùng một phiên trò chơi.

## Giải quyết bài toán

### Cơ chế đồng bộ game state

Game sử dụng cơ chế đồng bộ Deterministic Lockstep, cơ chế này hoạt động như sau:

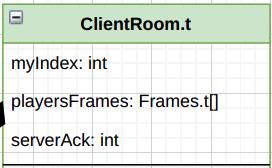
Mỗi client và server sẽ có một engine (trong trường hợp này là engine vật lý matter.js). Trong mỗi frame của engine, input (phím đang ấn, vị trí chuột, hướng tay cầm hoặc hướng điện thoại) của người chơi sẽ được gửi cho toàn bộ các người chơi khác và server. Sau đó engine của các máy sẽ cập nhật game state dựa vào hành động đó của người chơi. Do tất cả các máy sẽ sử dụng engine giống nhau, nhận vào input giống nhau trong từng frame, nên kết quả là state của engine sẽ giống nhau với tất cả các máy.

Nếu sử dụng TCP Socket để cập nhật input thì game sẽ bị ảnh hưởng rất nhiều bởi jitter. Vì TCP Socket sẽ luôn đợi gói tin trước gửi tới nơi để nhận gói tin sau. Do đó trong trường hợp 1 gói tin bị thất lạc, các gói tin đến sau sẽ phải đợi trong hàng đợi, khi nào gói tin bị thất lạc về tới nơi thì mới xử lý các gói tin đến sau. Khi đó game sẽ không thể cập nhật engine, dẫn đến việc đóng băng màn hình. Do đó không thể sử dụng TCP Socket cho việc cập nhật game state.

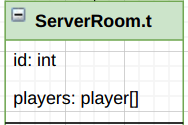
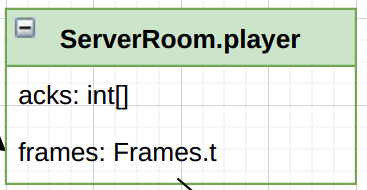
Nếu sử dụng UDP Socket thì ta cần phải xử lý việc gói tin bị thất lạc. Nếu chọn cách gửi lại gói tin bị mất thì ta sẽ gặp phải tình trạng giống như khi sử dụng TCP Socket.

Ta có thể gửi toàn bộ input phía bên kia chưa nhận được trong mỗi gói tin. Vì bên gửi cần biết bên nhận đã nhận được bao nhiêu input nên bên nhận sẽ gửi về số input đã nhận được, để bên gửi không gửi lại những input đó.

Thiết kế data phía client và server như sau:



Hình 4–2: Data lưu ở client



Hình 4–3: Data lưu ở server

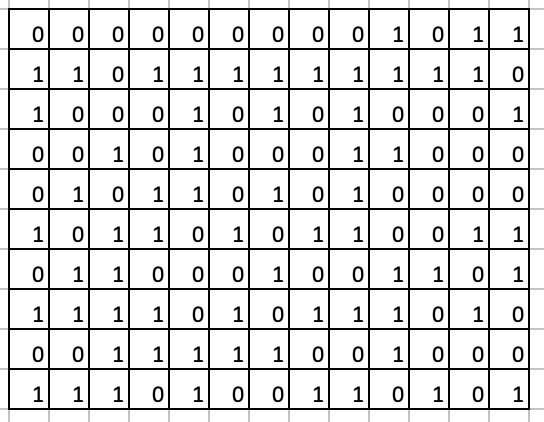
Ý nghĩa các trường:

* ClientRoom.t.myIndex: id của client, để phân biệt các client với nhau.
* ClientRoom.t.playersFrames: các input của mọi người chơi trong bàn chơi. Client dùng các input này để cập nhật engine.
* ClientRoom.t.serverAck: Số input server đã nhận được. Khi nhận được input từ server, client sẽ cập nhật trường này để không gửi lại các gói tin server đã nhận được.
* ServerRoom.player.acks: Các input client đã nhận được (input của các người chơi khác). Server khi nhận input từ client sẽ cập nhật trường này để không gửi lại những input mà client đã nhận được.
* ServerRoom.player.frames: Các input của client đã gửi lên. Server dùng trường này để cập nhật engine.
* ServerRoom.t.id: id của phòng chơi. Client sẽ gửi kèm id của phòng chơi lên cho server. Server sẽ bỏ qua các gói tin không có id phòng chơi tương ứng.

### Kiến tạo bản đồ

Mỗi bàn chơi mới được tạo ra sẽ có một bản đồ riêng. Do đó cần sử dụng các hàm tạo số ngẫu nhiên để tạo ra bản đồ.

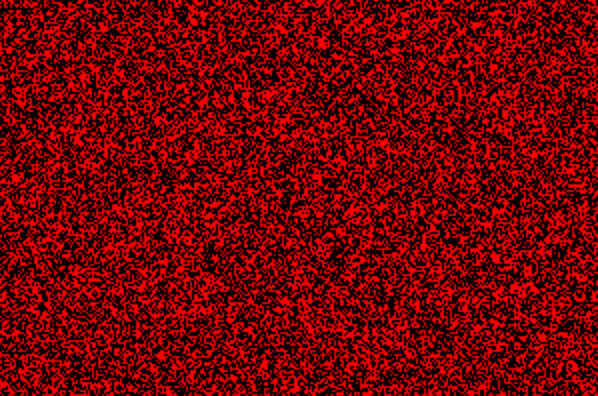
Bản đồ sẽ được biểu diễn bởi một mảng 2 chiều các số nguyên.



Hình 4–4: Biểu diễn mảng bản đồ với hàm sinh số ngẫu nhiên

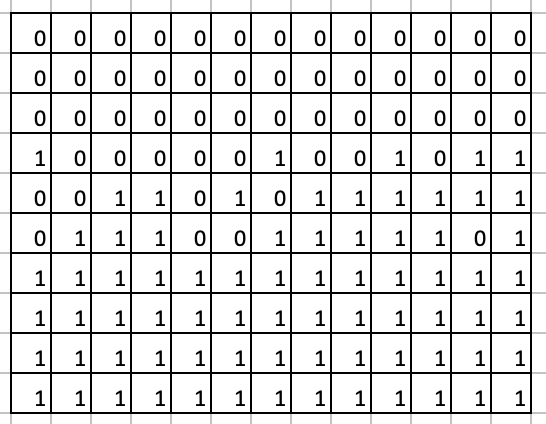
Các số 1 sẽ tương ứng với vùng đất, các vật thể không thể đi xuyên qua. Các số 0 sẽ tương ứng với không khí, các vật thể có thể đi xuyên qua.

Nếu sử dụng hàm sinh số ngẫu nhiên cho tất cả các ô thì sẽ chỉ nhận được một bản đồ với các ô đất rời rạc. Do đó không thể tạo nên địa hình.



Hình 4–5: Biểu diễn hình ảnh của bản đồ tạo bởi hàm sinh số ngẫu nhiên

Sử dụng hàm sinh nhiễu theo vùng (Gradient noise) sẽ cho ra một bản đồ liên kết hơn.



Hình 4–6: Biểu diễn mảng bản đồ với hàm sinh nhiễu Perlin 1D



Hình 4–7: Biểu diễn bản đồ được sinh bởi hàm sinh nhiễu Perlin

### Xử lý vật lý

#### Xử lý vị trí các đối tượng

Để xử lý di chuyển, mỗi đối tượng đều có các thông tin sau:

* Vector vị trí p(px, py)
* Vector vận tốc v(vx, vy)
* Vector gia tốc a(ax, ay)

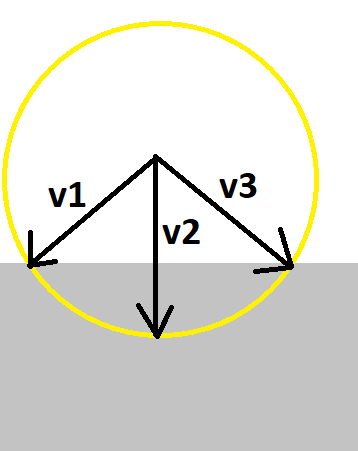
Cùng gia tốc trọng trường (0, gy), mỗi khung hình (trong thời gian t giây) sẽ xử lý vật lý một lần.

* Tính vận tốc mới:
* Tính vị trí mới:
* Xử lý va chạm với bản đồ
* Vẽ lại đối tượng ở vị trí mới.

#### Xử lý va chạm với bản đồ

Coi mỗi đối tượng là một hình tròn với tâm là vị trí và bán kính tương ứng với kích thước của đối tượng. Từ đó có thể kiểm tra va chạm với bản đồ một cách đơn giản hơn so với tính toán chính xác từng pixel của đối tượng.

Ví dụ với trạng thái sau:



Hình 4–8: Trạng thái va chạm giữa đối tượng và bản đồ

Tính tổng 3 vector v1, v2, v3 để ước lượng vector va chạm. Từ đó tìm ra vector pháp tuyến ) để bật ngược đối tượng trở lại theo pháp tuyến đó.

Khi đó vận tốc mới của đối tượng sẽ có hướng của vector v.

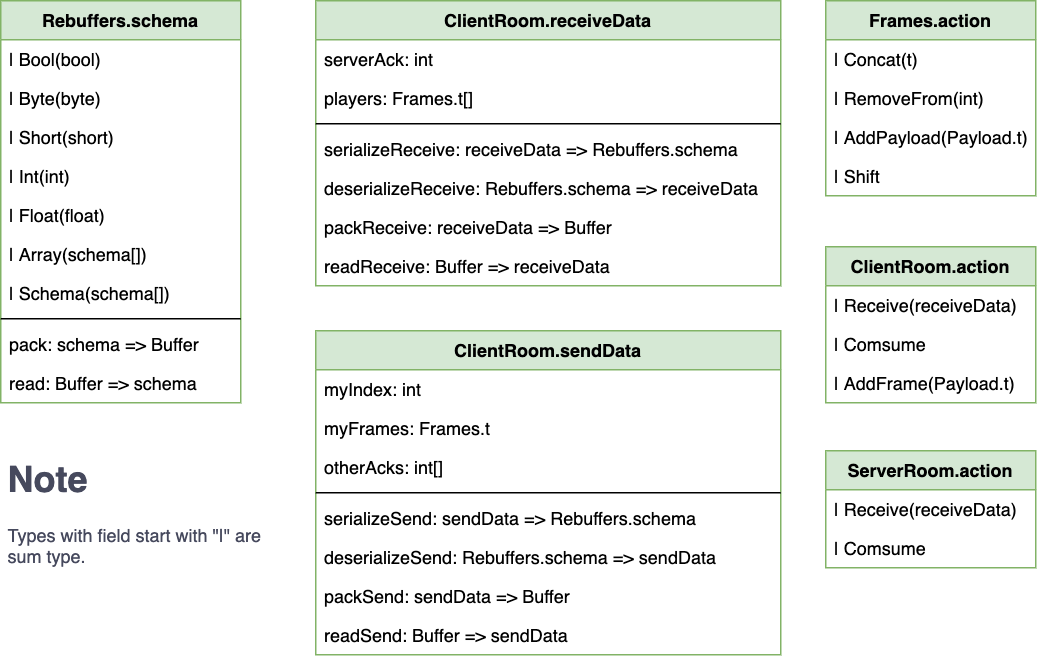
### Truyền tải bản đồ

Bản đồ được biểu diễn bằng một mảng 2 chiều có kích thước lớn (có thể lên đến hàng MB) và có thể thay đổi. Do đó việc truyền trực tiếp mảng bản đồ là không hiệu quả.

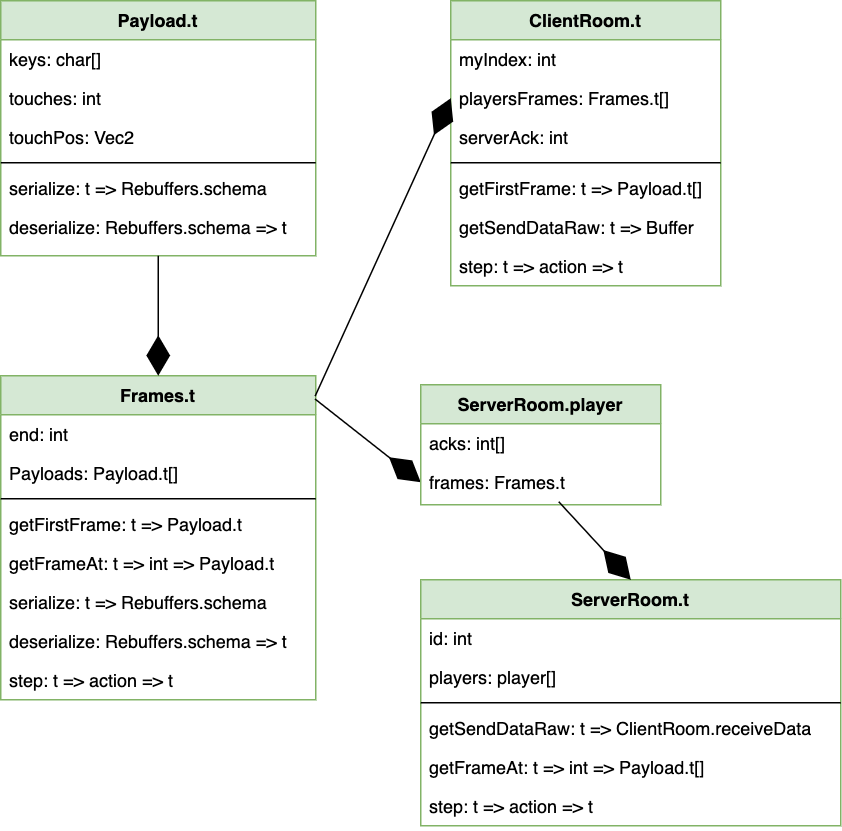
Tuy nhiên do bản đồ được tạo từ hàm sinh nhiễu Perlin cho nên có thể tận dụng hàm sinh số giả ngẫu nhiên với cùng một nguồn (seed) thì sẽ luôn luôn cho ra một kết quả giống nhau, do hàm sinh số ngẫu nhiên sẽ luôn sinh ra một dãy số giống hệt nhau khi có seed giống nhau. Do đó chỉ cần truyền seed này đến các máy khách và các máy này có thể tự sinh ra bản đồ giống nhau, khi các máy khách đều sử dụng chung một hàm sinh bản đồ giống nhau.

## Biểu đồ lớp thiết kế

### Biểu đồ lớp truyền tài:



Hình 4–9: Biểu đồ lớp truyền tải 1



Hình 4–10: Biểu đồ lớp truyền tải 2

# PHỤ LỤC: CÀI ĐẶT VÀ TRIỂN KHAI

## Cài đặt server:

Yêu cầu:

* Git
* NodeJS 12.0.0 trở lên

Tải dự án về máy:

Cài đặt:

npm install

Chạy server:

npm run start

## Cài đặt client:

Yêu cầu:

* Git
* NodeJS 12.0.0 trở lên

Tải dự án về máy

Cài đặt:

# KẾT LUẬN

Những vấn đề đã đạt được

* Hoàn thành Demo sản phẩm với các chức năng quan trọng:
  + Truyền gửi dữ liệu đầy đủ.
  + Hệ thống vật lý hoạt động tương đối chính xác và hiệu năng cao.
  + Môi trường có thể được phá huỷ khi tương tác với đạn đạo.
  + Hệ thống ghép cặp hoàn chỉnh.

Những vấn đề chưa đạt được

* Chưa hoàn thành đầy đủ chức sản phẩm như thiết kế:
  + Chức năng sử dụng kỹ năng và điểm năng lượng.
* Đồ hoạ và giao diện chưa đẹp.

Kết luận và định hướng phát triển:

* Cải thiện thêm phần giao diện người dùng.

# TÀI LIỆU THAM KHẢO

Multiplayer Game Programming: Architecting Networked Games (Game Design) - Joshua Glazer và Sanjay Madhav.

"What is a Game Engine?”:

<https://www.gamecareerguide.com/features/529/what_is_a_game_.php>

O'Neill, John (January 15, 2008). "My Turn: The Real Cost of Middleware".

<http://www.gamedaily.com/articles/features/my-turn-the-real-cost-of-middleware/71334/?biz=1>