THIẾT KẾ VÀ PHÁT TRIỂN GAME

Bài 7: Unity networking

Nội dung

1. Multiplayer game
2. Multiplayer flow
3. Networking
4. Multiplayer game trong unity
5. Game Pong, phiên bản mạng

Phần 1

Network multiplayer game

Multiplayer game

- “A multiplayer game is a game played by multiple people”
- Nhiều người cùng chơi một game?
  - Local (single device)
    - Badlands
    - King of the Opera
    - FIFA
  - Network
    - World of Warcraft
    - Clash of Clans
    - Hearthstone

Local multiplayer game (one device)

- Rất thú vị
- Phong phú
- Phổ biến
- Nhưng bản chất kĩ thuật thì không khác gì game cho một người chơi
  - Có thể phức tạp hơn đôi chút khi phải xử lý yêu cầu từ nhiều thiết bị, nhưng về cơ bản thì không khác gì nhiều so với game người chơi với máy

Biến cố xảy ra trong game

- Biến cố tương đương với việc phải xử lý sự kiện, vì thế biến cố trong game hầu như tương đương với khối lượng lập trình và xử lý gameloop
- Theo lượt (turn base):
  - Chess
  - Heroes of Might and Magic
- Thời gian thực (real time):
  - World of Warcraft
  - Quake
- Clash of Clans???

Game thực sự sẽ thực thi ở đâu?

- "Who runs the world?"
- On clients
  - P2P
  - Light server – heavy client
- On server
  - Runs the game
  - Client works as terminal
- Câu hỏi này quan trọng vì liên quan đến kiến trúc của các framework hỗ trợ xử lý qua mạng

Kiến trúc server thắng thế?

- Chi phí cho server đang giảm dần
- Ít gặp vấn đề khi kết nối (firewall, port forwards,…)
- Dễ ngăn chặn cheater, auto tools,…
- Game server ≠ server
  - Với những game nhỏ thì thiết bị của một người chơi nào đó có thể đóng vai trò server
  - Điều khác biệt ở đây là “mọi thứ” của trò chơi sẽ diễn ra trên server, còn các máy khác chỉ thực hiện nhiệm vụ đồng bộ trò chơi với server mà thôi
  - Chơi trên máy server sẽ ít bị lag hơn? Không hẳn, nhưng chắc chắn không thể lag nhiều hơn máy client

Phần 2

Multiplayer flow

Multiplayer flow

L
- Lobby
- Hoạt động trước khi chơi

M
- Match
- Hoạt động trải nghiệm chính của game

C
- Conclusion
- Động lực khiến player bỏ tiền ra

Lobby

- Các thiết bị chưa biết thông tin về trận đấu (cần có cơ chế phù hợp để tìm kiếm trận đấu)
- Xây dựng giao thức tìm kiếm hợp lý để có thể tìm được server (host)
- Trung tâm kết nối các người chơi (server lớn)
- Tách nhóm
- Phân hạng
- Các hoạt động tổ/đội
- Khá thú vị nhưng không phải chủ đề của bài này

Match

- Server đóng vai trò khởi tạo và xử lý PGS (persistent game state)
- Client gửi các action đến server (dựa trên action của người chơi tại client đó)
- Server thực hiện việc kiểm tra, xác nhận, cập nhật trạng thái và thông báo lại cho tất cả các client
  - Việc thông báo này thường là một chiều
- Với những game nhỏ hoặc có thời gian ngắn, một client có thể đóng vài trò server

Match

- PGS (persistent game state) là gì? Tập hợp những biến (variable) để mô tả lên toàn bộ trạng thái của game (MVC của game)
  - PGS của Chess? Game board
  - PGS của FPS? Vị trí của các player
  - Bản đồ của match, cách chơi của bot,…?
- Các client dựa trên PGS và các local option để xây dựng lại game tại client
- Nguyên tắc: một biến chỉ có một đối tượng chịu trách nhiệm về nó, các đối tượng khác chỉ tham chiếu đến

Actions vs Changes

- Các client phải thông tin đến server mỗi khi người chơi phát sinh action, trong trường hợp này client có nhiều lựa chọn về cách thông tin đến server
  - Cập nhật trạng thái (state based): máu của người chơi A là 300
  - Cập nhật thay đổi (delta based): máu của người chơi A tăng thêm 100
  - Cập nhật hành động (action based): người chơi A uống 100 máu
- Tương tự như vậy, server cần phải chọn phương án thông tin cho client một cách phù hợp

Actions vs Changes

- Chú ý: client chỉ cần thông báo với server về sự kiện do người chơi local tạo ra, nhưng server cần báo cho mọi client tất cả các cập nhật do cả những client khác tạo ra
- Thiết lập protocol giao tiếp giữa client và server rất tùy thuộc vào thể loại trò chơi
- State based:
  - Đơn giản, dễ hiểu
  - Tốn băng thông
  - Kém an toàn

Actions vs Changes

- Delta based:
  - An toàn hơn state based
  - Ít tốn băng thông nhất
  - Ít tốn công suất xử lý nhất cho client
  - Server xử lý nặng hơn một chút so với state based
- Action based:
  - An toàn nhất
  - Client và server đều tốn công suất xử lý
  - Ít gặp phải vấn đề về lag, đồng bộ
  - Dễ ghi biên bản trận đấu

Conclusion

- Hệ thống phần thưởng là quan trọng
- Cập nhật metagame của người chơi
  - High score
  - Rank
  - XP
- Replay
- Cập nhật thống kê

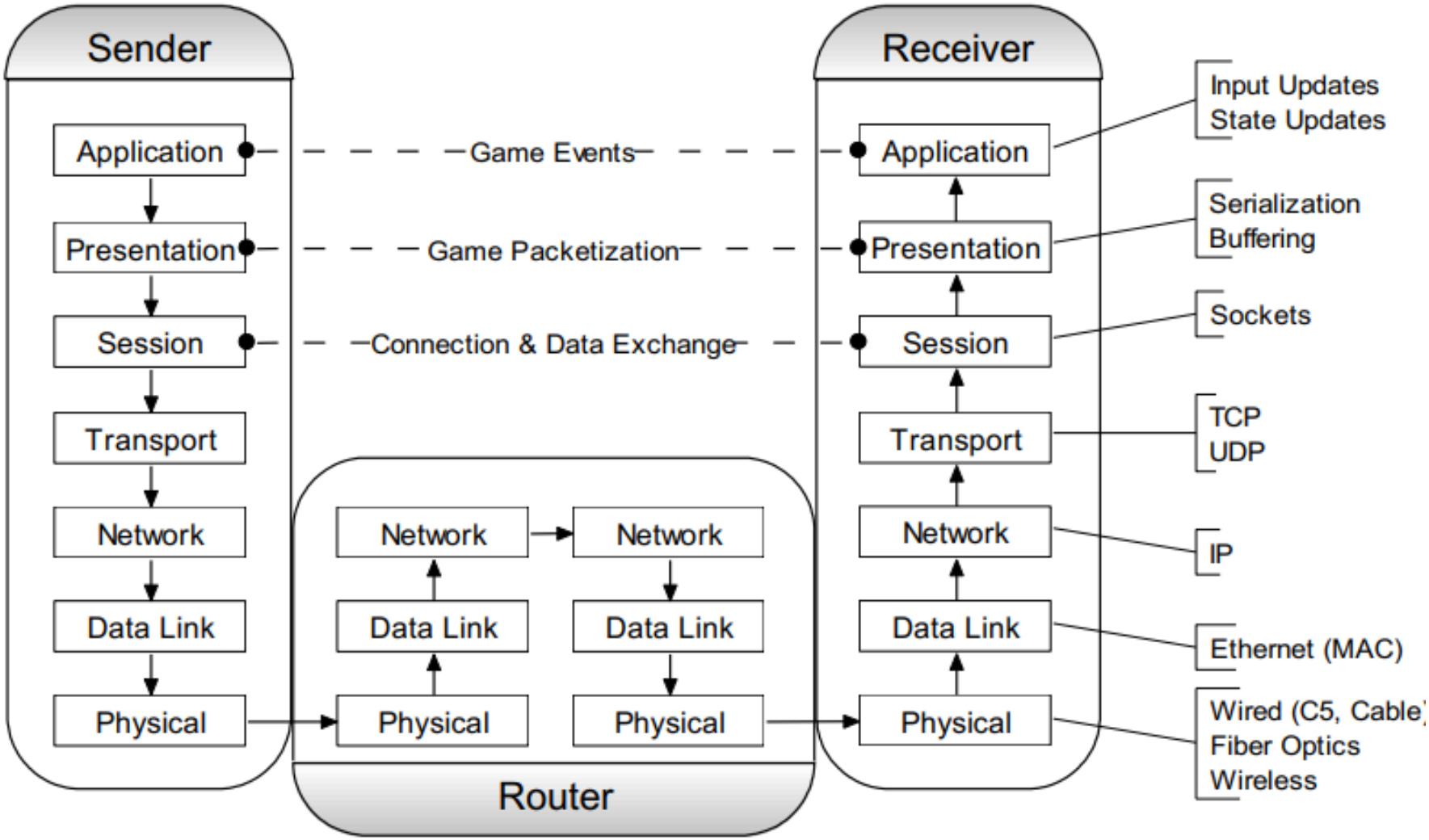
Phần 3

Networking

Networking

- Networking = kết nối giữa các máy tính
  - Vấn đề truyền dẫn dữ liệu
  - Thiết kế hệ thống thế nào để có thể làm việc tốt
  - Độ dài gói dữ liệu cho mỗi lượt truyền
  - Phương pháp nhận
  - Kiểm tra lỗi, sửa lỗi
  - Nén dữ liệu
  - Mã hóa dữ liệu
  - Điều khiển gói tin

Layers


Sender
Receiver
Application
Presentation
Session
Transport
Network
Data Link
Physical
Game Events
Game Packetization
Connection & Data Exchange
Network
Network
Data Link
Data Link
Physical
Physical
Router
Application
Presentation
Session
Transport
Network
Data Link
Physical
Input Updates
State Updates
Serialization
Buffering
Sockets
TCP
UDP
IP
Ethernet (MAC)
Wired (C5, Cable)
Fiber Optics
Wireless

Tầng vật lý

- Băng thông
  - Độ lớn của đường truyền
  - Đo bằng bps = bits per second
- Độ trễ
  - Thời gian gói tin đi từ điểm A đến điểm B
  - Đo bằng milliseconds
- Thiết bị: cáp quang, cáp đồng, hồng ngoại, không dây,…

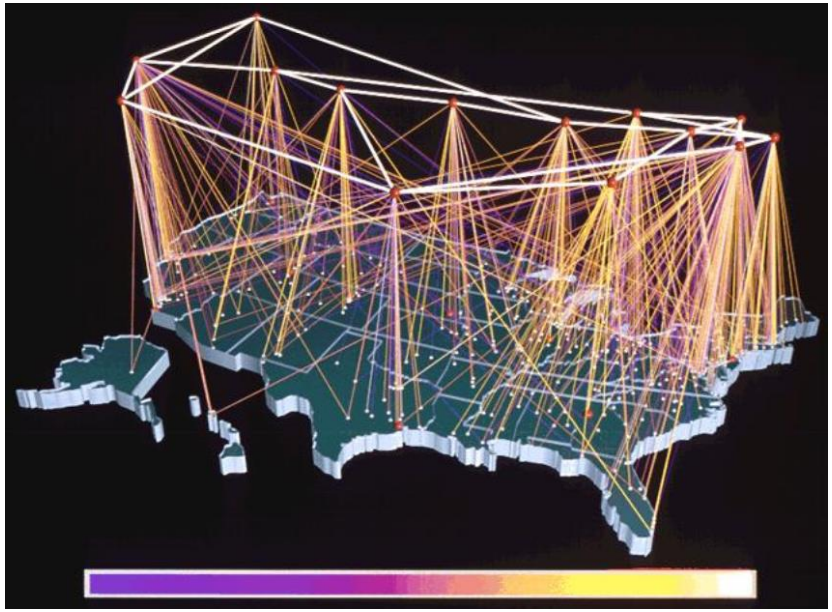
	Serial	USB 1&2	ISDN	DSL	Cable	LAN 10/100/1G BaseT	Wireless 802.11 a/b/g	Power Line	T1
Speed (bps)	20K	12M 480M	128k	1.5M down 896K up	3M down 256K up	10M 100M 1G	b=11M a,g=54M	14M	1.5M

Tầng liên kết dữ liệu (data link)

- Serialize dữ liệu đến/từ tầng vật lý
- Thiết bị mạng và giao tiếp
  - Ethernet
  - Địa chỉ MAC

Tầng mạng (network)

- Packet Routing
  - Hops
    - No connection
    - Guarantees sending
    - Doesn’t guarantee receiving
    - Non-deterministic path
  - Routers, Hubs, Switches



- Internet Protocol (IP)
  - Contains Source & Destination IP Address
  - IPv4 vs IPv6
- Unicast, Broadcast, Loop back

Tầng mạng: domain name service

- Dịch vụ tên miền
  - Chuyển đổi từ tên sang địa chỉ IP
  - Phải kết nối đến một (hoặc vài) DNS server để phân giải tên miền
  - Có thể lưu trữ cache ở máy local
- Tips
  - Dịch vụ phân giải tên miền chạy khá chậm và có thể trả về lỗi thay vì kết quả, vì thế nên thực hiện phân giải tên miền lần đầu, sau đó lưu trữ và sử dụng trong toàn phiên kết nối
  - Trường hợp phiên dài, có thể check lại dns sau vài giờ

Tầng giao vận (transport)

- Quản lý luồng dữ liệu giữa các điểm cuối
  - Sửa lỗi
  - Data flow
- TCP and UDP used with IP
  - Contains Source and Destination Port
- Port + IP = Net Address
  - Port Range = 0-64k
  - Well known Ports 0-1k
  - http, ftp, ssh, …

Tầng giao vận: TCP

- Connection based
  - Keep Alive
  - Handles breaking up data into correct size
  - Packet window
  - Packet Coalescense
- Guaranteed, in order delivery
  - ack, nack, resend
- Flow Control
- Easy to use
  - Reading and writing, just like a file
- Requires more header data

Tầng giao vận: UDP

- No connection
- No guarantees
  - May not arrive
    - TTL (time to live) – hop count limit
  - May not arrive in order
  - May arrive multiple times
  - Source not verified
- Datagram
  - Sent in packets exactly as user sends them
- Capable of broadcasting

Tầng giao vận: TCP vs UDP

- Khi nào dùng cái nào?
  - Tùy vào từng loại game
  - Hoặc sử dụng kết hợp cả 2
- TCP
  - Turn based games, leader boards
- UDP
  - More common, especially for time sensitive games
  - Add TCP features as needed
  - Unity uses UDP, with features for reliable, in order transmission

Tầng phiên (session)

- Manages Connections between Apps
  - Connect
  - Terminate
  - Data Exchange
- Socket API live at this layer
  - Cross platform
  - Cross language

Tầng phiên: sockets

- Based on File I/O
  - File Descriptors
  - Open/Close
  - Read/Write
- Modes
  - Blocking
    - Sử dụng thread riêng cho từng socket
  - Non-blocking
    - Thăm dò socket định kỳ

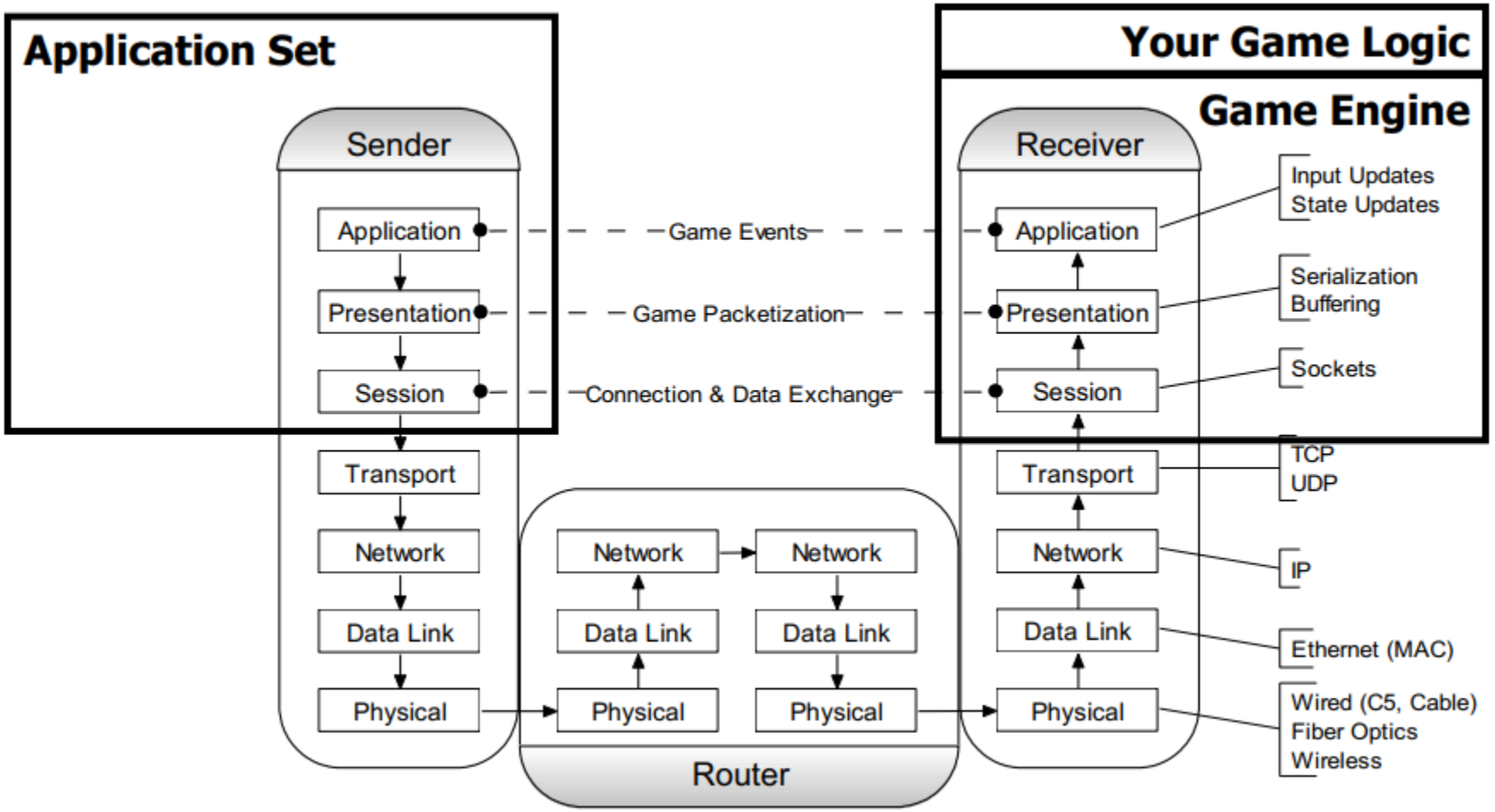
Tầng presentation

- Chuẩn bị dữ liệu cho việc giao vận
  - Nén
  - Mã hóa
  - Endian Order
    - 0b1000 vs 0b0001
  - Serialize
  - Buffering
    - Hợp nhất packet
    - Increased Latency
    - Store local data and wait

Tầng application

- Giao tiếp với người dùng
- Xử lý logic của trò chơi
- Chuyển vận / đồng bộ dữ liệu giữa các hệ thống

Protocol stack



Phần 4

Multiplayer game trong unity

Multiplayer game trong unity

- Multiplayer game trong unity mới được chuẩn hóa sau một thời gian dài "vật vã" giữa các lựa chọn khác nhau
- Phiên bản mới dựa trên Raknet tương đối ổn định, cách tiếp cận hợp lý
- Có nhiều cách triển khai multiplayer trên unity
  - HTTP requests
  - Plugin (GPGS, Photon, Smartfox,…)
  - Unity networking
  - UNet (Unity 5.1 trở đi)

HTTP requests

- Phù hợp với những game đơn giản (server web) hoặc xử lý một số tình huống trong game (chẳng hạn: upload ảnh lên web)
- Sử dụng UnityWebRequest, chú ý xử lý đồng bộ

```
IEnumerator GetRequest(string uri) {
    UnityWebRequest uwr = UnityWebRequest.Get(uri);
    yield return uwr.SendWebRequest();

    // xử lý lỗi
    if (uwr.isNetworkError)
        …
}
```

Plugins

- Các plugin thường là các giải pháp của những công ty cung cấp dịch vụ server
- Có thể tải xuống từ Asset Store
- Một số plugin nên thử:
  - Bluetooth LE for iOS and Android
  - Google Play Game Services
  - PUN (Photon Unity Network)
  - SmartFoxServer2X

Thử một ví dụ với UNet

1. Tạo một Scene
2. Tạo một đối tượng quản lý kết nối: NetworkManager (empty game object)
    1. Thêm component NetworkManager
    2. Thêm component NetworkManagerHUD
3. Tạo đối tượng player:
    1. Một GameObject bất kì
    2. Thêm component NetworkIdentity
    3. Đưa đối tượng vào Prefabs
4. Đăng ký player với NetworkManager (Spawn Info)

Thử một ví dụ với UNet

- Viết mã di chuyển player

```
public class Player : MonoBehaviour
{
    void Update()
    {
        var x = Input.GetAxis("Horizontal") * 0.1;
        var y = Input.GetAxis("Vertical") * 0.1;
        transform.Translate(x, y, 0);
    }
}
```

- Cách test: build một bản chạy trên PC, chạy 2 bản song song, 1 server, 1 client; kết nối và cùng chạy

Thử một ví dụ với UNet

- Test 1: các vấn đề
  - Chưa có đồng bộ trạng thái trên hai thiết bị
  - Khi điều khiển thì sẽ điều khiển đồng thời cả 2 đối tượng ➔ Chưa phân biệt được các player của các máy khác nhau
- Cập nhật 1:
  - Thêm NetworkTransform vào player
  - Chuyển class thành NetworkBehaviour
  - Viết mã chỉ điều khiển local player mà thôi

Thử một ví dụ với UNet

- Mã:

```
using UnityEngine.Networking;

public class PlayerMove : NetworkBehaviour
{
    void Update()
    {
        if (!isLocalPlayer) return;

        var x = Input.GetAxis("Horizontal") * 0.1;
        var y = Input.GetAxis("Vertical") * 0.1;
        transform.Translate(x, y, 0);
    }
}
```

Thử một ví dụ với UNet

- Test 2 ➔ đã phân biệt được player, nhưng chưa đồng bộ
- Update 2: phần NetworkIdentity chọn Local Player Authority
- Test 3:
  - Đã đồng bộ (hơi giật, làm thế nào để đỡ giật?)
  - Chưa phân biệt được các player (vì giống hệt nhau)
- Update 3: đổi màu người chơi local

Thử một ví dụ với UNet

- Mã:

```
public override void OnStartLocalPlayer()
{
  GetComponentInChildren<SpriteRenderer>().color =
Color.yellow;
}
```

Netcode

- Cho đến nay (2023), Unity vẫn chưa có một giải pháp trọn vẹn cho game nhiều người chơi
- UNet trở thành một phần của Unity MLAPI
- Nhưng MLAPI cũng đã bị dẹp qua và chuyển sang Netcode
- Hầu như giữ nguyên cách làm của UNet, nhưng khác biệt trong những tình huống RPC
- Công nghệ chưa trưởng thành, nên cân nhắc các giải pháp khác (SmartFox, Photon Engine,...)

Phần 5

Game Pong, phiên bản mạng