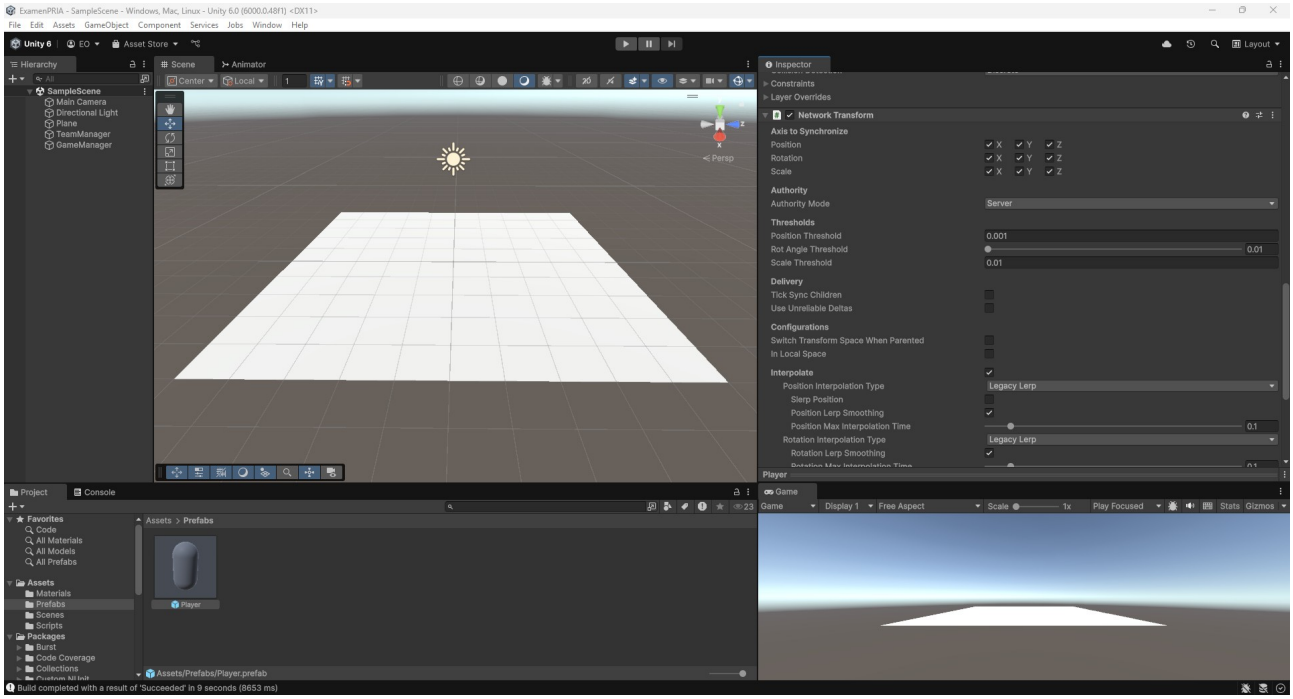
(1 punto) Uso axeitado de variables, métodos, orde, eficiencia e calidade do código. Realización dun pequeno informe con capturas dos pasos e brevísimas explicacións para cada apartado.
(2 puntos) Crea desde cero un xogo 3D semellante ao getting started do manual de netcode. Isto é, un xogo no que hai un plano e sobre él os playeres, que serán unha cápsula que se move nas catro direccións sobre o plano utilizando teclas (frechas, ASDW ou ambas). Debes usar Network transform.

Prefab de la Capsula usando network transform



Parte del script PlayerController.cs que maneja el movimiento con WASD (los ejes verticales/horizontales)

```
// Movimento con ASDW ou flechas
1 reference
private void HandleMovementInput()
{
    float h = Input.GetAxis("Horizontal");
    float v = Input.GetAxis("Vertical");
    Vector3 dir = new Vector3(h, 0f, v);

    if (dir.magnitude > 0.1f)
    {
        Vector3 move = dir.normalized * speed * Time.deltaTime;
        transform.Translate(move, Space.World);
    }
}
```

(1 punto) Fai que cando se spawnea apareza nun punto aleatorio da parte central do plano (a que está etiquetada como "sen equipo" no esquema de abaixo) e de cor branca. Crea un botón chamado "mover a inicio" que leve ao player á parte central do plano, fai que a tecla "m" teña a mesma acción. Se estamos executando en servidor esa acción debe ser feita por todos os players.

Tengo 3 scripts:

GameManager que se encarga de gestionar Clientes y Hosts que va en un Game Object vacio junto el componente Network Manager y el Unity Transport.

TeamManager que se encarga de gestionar los equipos, cada jugador en cada equipo, los colores y las posiciones (de aquí es de donde voy sacar la zona de spawneo central). Este script va en un Game Object vacio que se llama TeamManager

PlayerController que va en el prefab del jugador, sincronizando su movimiento, color y equipo dependiendo si en Owner. Junto los componentes: Network Object, Network Transform y Rigidbody. Hay problemas de sincronizacion con los clientes en el proyecto final que no pude resolver.

Para el spawn en la parte central del plano, en TeamManager creamos el metodo

```
/// <summary>
/// Devolve unha posición aleatoria no cadrado central (X e Z en [-2,2], Y=1).
/// </summary>
2 references
public Vector3 GetRandomCenterPosition()
{
    float x = Random.Range(-2f, 2f);
    float z = Random.Range(-2f, 2f);
    return new Vector3(x, 1f, z);
}
```

Lo llamamos en el spawn en el GameManager

```
/// <summary>
/// Executado no servidor cando un cliente conéctase correctamente.
/// Teleporta ao player á zona central e léo ao equipo 0 (sen equipo, cor branca).
/// </summary>
1 reference
private void OnClientConnected(ulong clientId)
{
    if (!NetworkManager.Singleton.IsServer) return;
    if (_teamManager == null) return;

    var playerObj = _netManager.SpawnManager.GetPlayerNetworkObject(clientId);
    if (playerObj == null) return;

    // Spawn aleatorio na zona central
    Vector3 centerPos = _teamManager.GetRandomCenterPosition();
    playerObj.transform.position = centerPos;

    // Engadir ao equipo 0 (sen equipo)
    _teamManager.AddPlayerToTeam(clientId, 0);

    // Poñer cor branca no servidor e notificar a todos
    var pc = playerObj.GetComponent<PlayerController>();
    if (pc != null)
        pc.SetInitialTeamOnServer(0);
}
```

Y cuando un cliente pide que el servidor lo mueva con el metodo RequestTeleportServerRcp, de paso muestro el metodo que usare mas tarde para gestionar los cambios de posición

```
// ============================================================
// == 1) TELEPORT: Cliente → Servidor
// ============================================================
[ServerRpc(RequireOwnership = false)]
4 references
public void RequestTeleportServerRpc(ServerRpcParams rpcParams = default)
{
    // Só o servidor pode reposicionar de verdade: calculamos nova posición no centro
    Vector3 centerPos = TeamManager.Instance.GetRandomCenterPosition();
    transform.position = centerPos;
}
```

en GameManager tengo el metodo OnGUI para mostrar botones, dependiendo si somos cliente o host, nos mueve solo a nosotros o nos mueve a todos llamando al metodo RequestTeleportServerRcp

```
// 3) Botón "Mover a inicio":
//    - se somos servidor puro, teleporta a todos;
//    - se somos cliente/host, só a nós mesmos
if (GUILayout.Button("Mover a inicio"))
{
    if (_netManager.IsServer && !_netManager.IsClient)
    {
        // Servidor puro: teleporta a todos os clients
        foreach (ulong clientId in _netManager.ConnectedClientsIds)
        {
            var netObj = _netManager.SpawnManager.GetPlayerNetworkObject(clientId);
            if (netObj == null) continue;
            var pc = netObj.GetComponent<PlayerController>();
            if (pc != null)
                pc.RequestTeleportServerRpc();
        }
    }
    else
    {
        // Cliente ou Host: só teleporta o local
        var localObj = _netManager.SpawnManager.GetLocalPlayerObject();
        if (localObj != null)
        {
            var pc = localObj.GetComponent<PlayerController>();
            if (pc != null)
                pc.RequestTeleportServerRpc();
        }
    }
}
```

y en el PlayerController le asigno que con la tecla M haga lo mismo

```
// Tecla "M" para teletransportarse á zona central
1 reference
private void HandleTeleportInput()
{
    if (Input.GetKeyDown(KeyCode.M))
    {
        RequestTeleportServerRpc();
    }
}
```

(1 punto) Os players sen equipo (os que están na parte central do plano) teñen cor branca. Se o player se move para a zona do Equipo 1 (ver esquema abaixo) poñerase de cor vermello e os players que se poñan no Equipo 2 collerán cor azul. Se volven á parte central volverán a estar sen equipo e por tanto volverán ser de cor branca.

Todo esto lo maneja el TeamManager, en la siguiente captura muestro:

Las listas de colores de los equipos 1 y 2, _team1Colors y _team2Colors

El diccionario que guarda el ID de los equipos y los clientes que estan suscritos a cada uno, _teamMembership

Y el diccionario que guarda los colores asignados a cada cliente, _assignedColors

```
11 references
public static TeamManager Instance { get; private set; }

// Límite de players por equipo (definido polo servidor vía GameManager)
4 references
private int _maxPerTeam = 2;

// Dicionario: clave = ID de equipo (0=sen equipo, 1=Equipo 1, 2=Equipo 2), valor = lista de clientIds
10 references
private readonly Dictionary<int, List<ulong>> _teamMembership = new Dictionary<int, List<ulong>>();

// Paletas de cores por equipo
1 reference
private readonly List<Color> _team1Colors = new List<Color>
{
    Color.red,
    new Color(1f, 0.5f, 0f), // laranxa
    new Color(1f, 0.4f, 0.7f) // rosa
};
1 reference
private readonly List<Color> _team2Colors = new List<Color>
{
    new Color(0f, 0f, 0.5f),  // azul escuro
    new Color(0.5f, 0f, 0.5f),// violeta
    new Color(0f, 0.5f, 1f)   // azul claro
};

// Copias dispoñibles de cores (pool) para cada equipo
3 references
private List<Color> _team1Available;
3 references
private List<Color> _team2Available;

// Para gardar que cor foi asignada a cada clientId
8 references
private readonly Dictionary<ulong, Color> _assignedColors = new Dictionary<ulong, Color>();
```

Determinamos equipo del cliente por su posicion

```
/// <summary>
/// Dada unha posición, devolve:
/// 0 = sen equipo (central),
/// 1 = Equipo 1 (X < -2),
/// 2 = Equipo 2 (X > 2).
/// </summary>
1 reference
public int DetermineTeamByPosition(Vector3 pos)
{
    if (pos.x < -2f && pos.z >= -3f && pos.z <= 3f) return 1;
    if (pos.x > 2f && pos.z >= -3f && pos.z <= 3f) return 2;
    return 0;
}
```

Esto lo llama PlayerController para sincronizar

```
// Comproba cada frame en que zona estamos e, se mudou, pídelle ao servidor cambiar de equipo
1 reference
private void HandleTeamCheck()
{
    int newTeam = TeamManager.Instance.DetermineTeamByPosition(transform.position);
    int oldTeam = _currentTeam.Value;

    if (newTeam != oldTeam)
    {
        RequestChangeTeamServerRpc(newTeam);
    }
}
```

en todo momento en el update

```
0 references
private void Update()
{
    if (!IsOwner) return;

    // Gardar posición anterior en todo momento
    _lastPosition = transform.position;

    HandleMovementInput();
    HandleTeleportInput();
    HandleTeamCheck();
}
```

En el spawn se añade al player automaticamente al equipo 0 osea color blanca, con AddPlayerToTeam(clientId, 0) en el GameManager

```
private void OnClientConnected(ulong clientId)
{
    if (!NetworkManager.Singleton.IsServer) return;
    if (_teamManager == null) return;

    var playerObj = _netManager.SpawnManager.GetPlayerNetworkObject(clientId);
    if (playerObj == null) return;

    // Spawn aleatorio na zona central
    Vector3 centerPos = _teamManager.GetRandomCenterPosition();
    playerObj.transform.position = centerPos;

    // Engadir ao equipo 0 (sen equipo)
    _teamManager.AddPlayerToTeam(clientId, 0);

    // Poñer cor branca no servidor e notificar a todos
    var pc = playerObj.GetComponent<PlayerController>();
    if (pc != null)
        pc.SetInitialTeamOnServer(0);
}
```

Con el AddPlayerToTeam en TeamMananer se añade el ID del cliente al equipo

```
/// <summary>
/// Engade o clientId ao novo equipo (se cabe).
/// Elimina de calquera equipo anterior.
/// </summary>
2 references
public void AddPlayerToTeam(ulong clientId, int newTeam)
{
    // Quitar de equipo anterior
    foreach (var kvp in _teamMembership)
    {
        if (kvp.Value.Contains(clientId))
        {
            kvp.Value.Remove(clientId);
            break;
        }
    }
    // Se é equipo 1 ou 2, comprobamos límite
    if (newTeam != 0 && _teamMembership[newTeam].Count >= _maxPerTeam)
        return;

    _teamMembership[newTeam].Add(clientId);
}
```

comprobando antes si se puede entrar

```
/// <summary>
/// Devolve true se aínda cabe outro player no equipo indicado.
/// O equipo 0 (sen equipo) non ten límite.
/// </summary>
1 reference
public bool CanJoinTeam(int teamId)
{
    if (teamId == 0) return true;
    return _teamMembership[teamId].Count < _maxPerTeam;
}
```

Ahora para aplicar colores, el metodo que selecciona un color aleatorio dependiendo del equipo

```
/// <summary>
/// Cando un player entra nun equipo, devolve unha cor aleatoria non usada dese equipo.
/// Se o pool está baleiro, devolve branco.
/// </summary>
1 reference
public Color GetRandomColorForTeam(int teamId)
{
    List<Color> pool = teamId == 1 ? _team1Available : _team2Available;
    if (pool == null || pool.Count == 0)
        return Color.white;

    int idx = Random.Range(0, pool.Count);
    Color c = pool[idx];
    pool.RemoveAt(idx);
    return c;
}
```

y en el propio PlayerController llamamos a los metodos anteriormente mencionados del TeamManager para aplicar a ese player en especifico su equipo y color por ID enviandoselo al Servidor

```
public void RequestChangeTeamServerRpc(int desiredTeam, ServerRpcParams rpcParams = default)
{
    ulong clientId = rpcParams.Receive.SenderClientId;
    int oldTeam = _currentTeam.Value;

    // Se cabe no equipo desexado:
    if (TeamManager.Instance.CanJoinTeam(desiredTeam))
    {
        // 1) Se estaba noutro equipo, liberamos a cor dese equipo
        if (oldTeam == 1 || oldTeam == 2)
        {
            TeamManager.Instance.ReleaseColorFromTeam(oldTeam, clientId);
        }

        // 2) Engadimos ao novo equipo (0, 1 ou 2)
        TeamManager.Instance.AddPlayerToTeam(clientId, desiredTeam);
        _currentTeam.Value = desiredTeam;

        // 3) Xestionar cor segundo equipo
        if (desiredTeam == 1 || desiredTeam == 2)
        {
            Color newCol = TeamManager.Instance.GetRandomColorForTeam(desiredTeam);
            _assignedColor = newCol;
            TeamManager.Instance.RecordAssignedColor(clientId, newCol);
        }
        else
        {
            // Se volve ao centro, cor branca
            _assignedColor = Color.white;
        }

        // 4) Aplicar cor localmente en servidor (propagaráse aos clientes vía NetworkVariable
        ApplyColor(_assignedColor);
    }
    else
    {
        // Equipo cheo: devolvemos ao client á posición anterior e avisámolo
        transform.position = _lastPosition;
        TeamManager.Instance.NotifyClientTeamFull(clientId);
    }
}
```

en esta parte del PlayerController es donde se gestiona la aplicación correcta de colores en el propio GameObject del player

```
[ClientRpc]
1 reference
public void TeamFullClientRpc(ClientRpcParams rpcParams = default)
{
    if (!IsOwner) return;
    Debug.Log("Equipo cheo! Non podes unirte a ese equipo de momento.");
    // Podes amosar aquí un aviso en pantalla (GUI, etc.)
}

// ============================================================
// == 3) ONVALUECHANGED para _currentTeam: aplica cor
// ============================================================
2 references
private void OnTeamChanged(int oldTeam, int newTeam)
{
    // El _assignedColor xa foi calculado no ServerRpc
    Color col = (newTeam == 1 || newTeam == 2) ? _assignedColor : Color.white;
    ApplyColor(col);
}

3 references
private void ApplyColor(Color col)
{
    var renderer = GetComponent<Renderer>();
    if (renderer == null) return;
    renderer.material = new Material(renderer.sharedMaterial);
    renderer.material.color = col;
}

/// <summary>
/// Chamado por GameManager cando se fai spawn inicial para poñer no servidor equipo 0 e cor branca.
/// </summary>
1 reference
public void SetInitialTeamOnServer(int teamId)
{
    _currentTeam.Value = teamId;
    _assignedColor = Color.white;
    ApplyColor(Color.white);
}
```

(2 puntos) Fai que o equipo 1 teña as cores vermella, laranxa e rosa, e o equipo 2 teñas as cores azul escuro, violeta e azul claro. Cando un xogador entre nun equipo collerá unha das cores aleatorias que non estén collidas.

Ya explique el funcionamiento de esto en el punto 2 pero lo vuelvo a mostrar, las colores a elegir estan definidas en estas listas

```
// Paletas de cores por equipo
1 reference
private readonly List<Color> _team1Colors = new List<Color>
{
    Color.red,
    new Color(1f, 0.5f, 0f), // laranxa
    new Color(1f, 0.4f, 0.7f) // rosa
};
1 reference
private readonly List<Color> _team2Colors = new List<Color>
{
    new Color(0f, 0f, 0.5f),  // azul escuro
    new Color(0.5f, 0f, 0.5f),// violeta
    new Color(0f, 0.5f, 1f)   // azul claro
};
```

y en este metodo se escoge una de forma aleatoria que este libre

```
/// <summary>
/// Cando un player entra nun equipo, devolve unha cor aleatoria non usada dese equipo.
/// Se o pool está baleiro, devolve branco.
/// </summary>
1 reference
public Color GetRandomColorForTeam(int teamId)
{
    List<Color> pool = teamId == 1 ? _team1Available : _team2Available;
    if (pool == null || pool.Count == 0)
        return Color.white;

    int idx = Random.Range(0, pool.Count);
    Color c = pool[idx];
    pool.RemoveAt(idx);
    return c;
}
```

(2 puntos) Fai que en cada equipo só poida haber como máximo dous players (os sen equipo non teñen limitación). E en caso de que un equipo esté cheo, só se poderán mover os players dese equipo e os outros so poderán moverse de novo cando o equipo non esté cheo (utiliza ClientRPC para avisalos). Prográmao de tal xeito que poidas mudar facilmente o número máximo de players nun equipo no futuro.

(1 punto) Fai que no servidor apareza unha interface para seleccionar o número de players máximos de cada equipo.

Variables en el GameManager para el string del GUI y el int que marca el limite de jugadores por equipo

```
// Límite de players por equipo en string para o TextField
3 references
private string _maxPlayersInput = "2";
5 references
private int _maxPlayersPerTeam = 2;
```

en el metodo que gestiona los GUI añadimos el boton que nos permite elegir el limite

```
    // 4) Se somos servidor (ou Host), amosamos campo para configurar límite por equipo
    if (_netManager.IsServer)
    {
        GUILayout.Label("Límite max por equipo:");
        _maxPlayersInput = GUILayout.TextField(_maxPlayersInput, GUILayout.Width(50));
        if (int.TryParse(_maxPlayersInput, out int parsed))
            _maxPlayersPerTeam = Mathf.Max(1, parsed);

        if (GUILayout.Button("Aplicar límite"))
        {
            if (_teamManager != null)
                _teamManager.SetMaxPerTeam(_maxPlayersPerTeam);
        }
    }
}
```

y en el metodo ApproveOrReject del GameManager tenemos en cuenta la variable _maxPlayerPerTeam haciendo recuento de jugadores por equipo con GetCurrentCount y aprueba o rechaza la conexion

```
/// <summary>
/// Antes de aprobar a conexión, rexeita se os dous equipos xa están cheos.
/// Cada equipo admite _maxPlayersPerTeam. Os sen equipo non teñen límite.
/// </summary>
1 reference
private void ApproveOrReject(
    NetworkManager.ConnectionApprovalRequest req,
    NetworkManager.ConnectionApprovalResponse res)
{
    if (_teamManager == null)
    {
        // Se non hai TeamManager, permitimos a conexión
        res.Approved = true;
        res.CreatePlayerObject = true;
        res.PlayerPrefabHash = null;
        res.Position = Vector3.zero;
        res.Rotation = Quaternion.identity;
        return;
    }

    int countTeam1 = _teamManager.GetCurrentCount(1);
    int countTeam2 = _teamManager.GetCurrentCount(2);
    if (countTeam1 >= _maxPlayersPerTeam && countTeam2 >= _maxPlayersPerTeam)
    {
        res.Approved = false;
        res.Reason = "Lobby cheo: equipos en capacidade.";
        return;
    }

    res.Approved = true;
    res.CreatePlayerObject = true;
    res.PlayerPrefabHash = null;
    res.Position = Vector3.zero;
    res.Rotation = Quaternion.identity;
}
```

GetCurrentCount es un metodo simple del TeamManager que cuenta los jugadores por equipo

```
/// <summary>
/// Devolve cantos players hai agora no equipo indicado.
/// </summary>
2 references
public int GetCurrentCount(int teamId)
{
    return _teamMembership[teamId].Count;
}
```