Photon Unity Network Lobby

Arthur Pai

https://github.com/ispan-course/umvr08-tanks

大廳及房間匹配

- 加入大廳及房間的方式
  - 隨機匹配
  - 指定房間名稱
  - 透過列表選擇
- 房間過濾
  - 可以透過等級、權限或是擁有的技能等條件

注意事項

- 注意事項
  - 確認使用相同的Photon AppId
  - 確認客戶是否連接到同一地區(Region)，只有連接到同一地區的玩家才能一起遊戲
  - 確認使用相同的AppVersion
  - 確認玩家是否有不同的UserID，相同UserID的玩家不能加入同一個房間
  - 加入一個指定名字的房間之前，確保這個房間已經被創建，或者可以使用JoinOrCreateRoom去加入
  - 如果要加入一個隨機房間時，確保選擇Lobby是指定相同的大廳名稱和類型
  - 如果要使用房間屬性作為過濾器進行隨機匹配，請確保在創建房間時將這些屬性設為在大廳中是可見的
  - 如果要使用SQL過濾器進行隨機匹配，請確保將保留的過濾屬性鍵設為在大廳中是可見的。
    - 在每次隨機匹配嘗試中放鬆過濾條件
    - 或者使用鍊式過濾器(chained filters)
    - 或者在多次嘗試失敗後在某個時間點創建新房間。

快速匹配

- 現在的大多數玩家只想馬上進入遊戲，所以Photon提供了隨機進入房間的機制
- JoinRandomOrCreateRoom
- JoinRandomRoom / CreateRoom
  - 失敗原因：現有的房間 closed、invisible、full

Custom Room Properties

- 自定義房間屬性

```
public const string MAP_PROP_KEY = "map";
public const string GAME_MODE_PROP_KEY = "gm";
public const string AI_PROP_KEY = "ai";
```

```
RoomOptions roomOptions = new RoomOptions();
roomOptions.CustomRoomPropertiesForLobby
   = new [] { MAP_PROP_KEY, GAME_MODE_PROP_KEY, AI_PROP_KEY };
roomOptions.CustomRoomProperties
   = new ExitGames.Client.Photon.Hashtable {
        { MAP_PROP_KEY, 1 },
        { GAME_MODE_PROP_KEY, 0 }
     };
PhotonNetwork.CreateRoom(null, roomOptions, null);
```

- 隨機加入符合特定屬性的房間

```
byte expectedMaxPlayers = 4;
var expectedCustomRoomProperties
      = new ExitGames.Client.Photon.Hashtable { { MAP_PROP_KEY, 1 } };
PhotonNetwork.JoinRandomRoom(expectedCustomRoomProperties, expectedMaxPlayers);
```

Custom Room Properties Example

Custom Room Properties Example

- GameManager
  - 將 PhotonNetwork.ConnectUsingSettings 提出到另一個 function

```
void Start()
{
  SceneManager.sceneLoaded += OnSceneLoaded;

  PhotonNetwork.GameVersion = gameVersion;
}

public bool ConnectToServer(string account)
{
  PhotonNetwork.NickName = account;
  return PhotonNetwork.ConnectUsingSettings();
}
```

Custom Room Properties Example

```
namespace Tanks
{
  public class MainMenu : MonoBehaviourPunCallbacks
  {
    public static MainMenu instance;          // 改成 Public
    private GameObject m_ui;
    private TMP_InputField  m_accountInput; // 新增 輸入匡
    private Button m_loginButton;             // 新增 登入按鈕
    private Button m_joinGameButton;

    void Awake()
    {
      if (instance != null)
      {
        DestroyImmediate(gameObject);
        return;
      }

      instance = this;

      m_ui = transform.FindAnyChild<Transform>("UI").gameObject;
      m_accountInput = transform.FindAnyChild<TMP_InputField>(“AccountInput"); // 抓取輸入匡元件
      m_loginButton = transform.FindAnyChild<Button>(“LoginButton");            // 抓取登入按鈕元件
      m_joinGameButton = transform.FindAnyChild<Button>("JoinGameButton");

      ResetUI(); // 抽出 UI 初始化
    }
```

Custom Room Properties Example

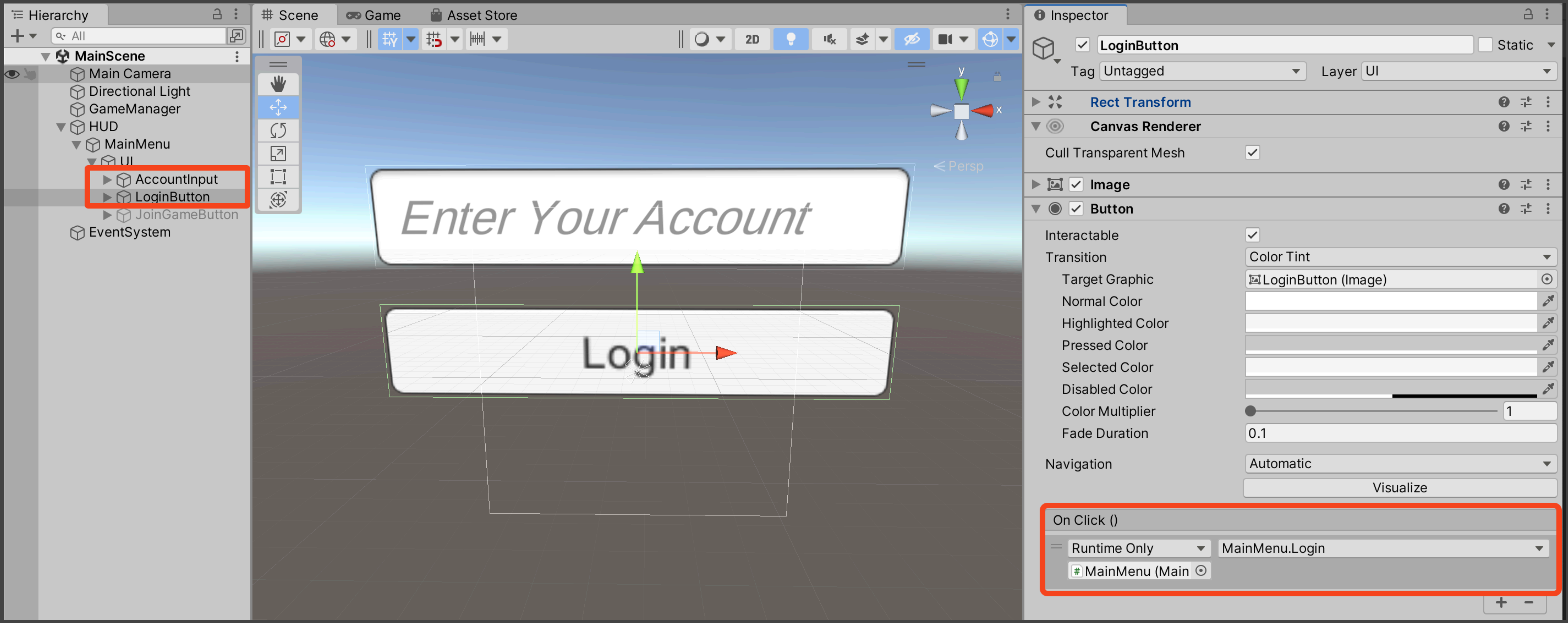
```
private void ResetUI() // 重置 UI
{
  m_ui.SetActive(true);
  m_accountInput.gameObject.SetActive(true);
  m_loginButton.gameObject.SetActive(true);
  m_joinGameButton.gameObject.SetActive(false);

  m_accountInput.interactable = true;
  m_loginButton.interactable = true;
  m_joinGameButton.interactable = true;
}

public override void OnEnable()
{
  // Always call the base to add callbacks
  base.OnEnable();

  SceneManager.sceneLoaded += OnSceneLoaded;
}

public override void OnDisable()
{
  // Always call the base to remove callbacks
  base.OnDisable();

  SceneManager.sceneLoaded -= OnSceneLoaded;
}
```

Custom Room Properties Example

```
    public void Login() // 處理 登入伺服器流程
    {
      if (string.IsNullOrEmpty(m_accountInput.text))
      {
        Debug.Log("Please input your account!!");
        return;
      }

      m_accountInput.interactable = false;
      m_loginButton.interactable = false;

      if (!GameManager.instance.ConnectToServer(m_accountInput.text))
      {
        Debug.Log("Connect to PUN Failed!!");
      }
    }

    private void OnSceneLoaded(Scene scene, LoadSceneMode mode)
    {
      m_ui.SetActive(!PhotonNetwork.InRoom);
    }

    public override void OnConnectedToMaster() // 處理連線後 UI 變化
    {
      m_accountInput.gameObject.SetActive(false);
      m_loginButton.gameObject.SetActive(false);
      m_joinGameButton.gameObject.SetActive(true);
    }
  }
}
```

Custom Room Properties Example

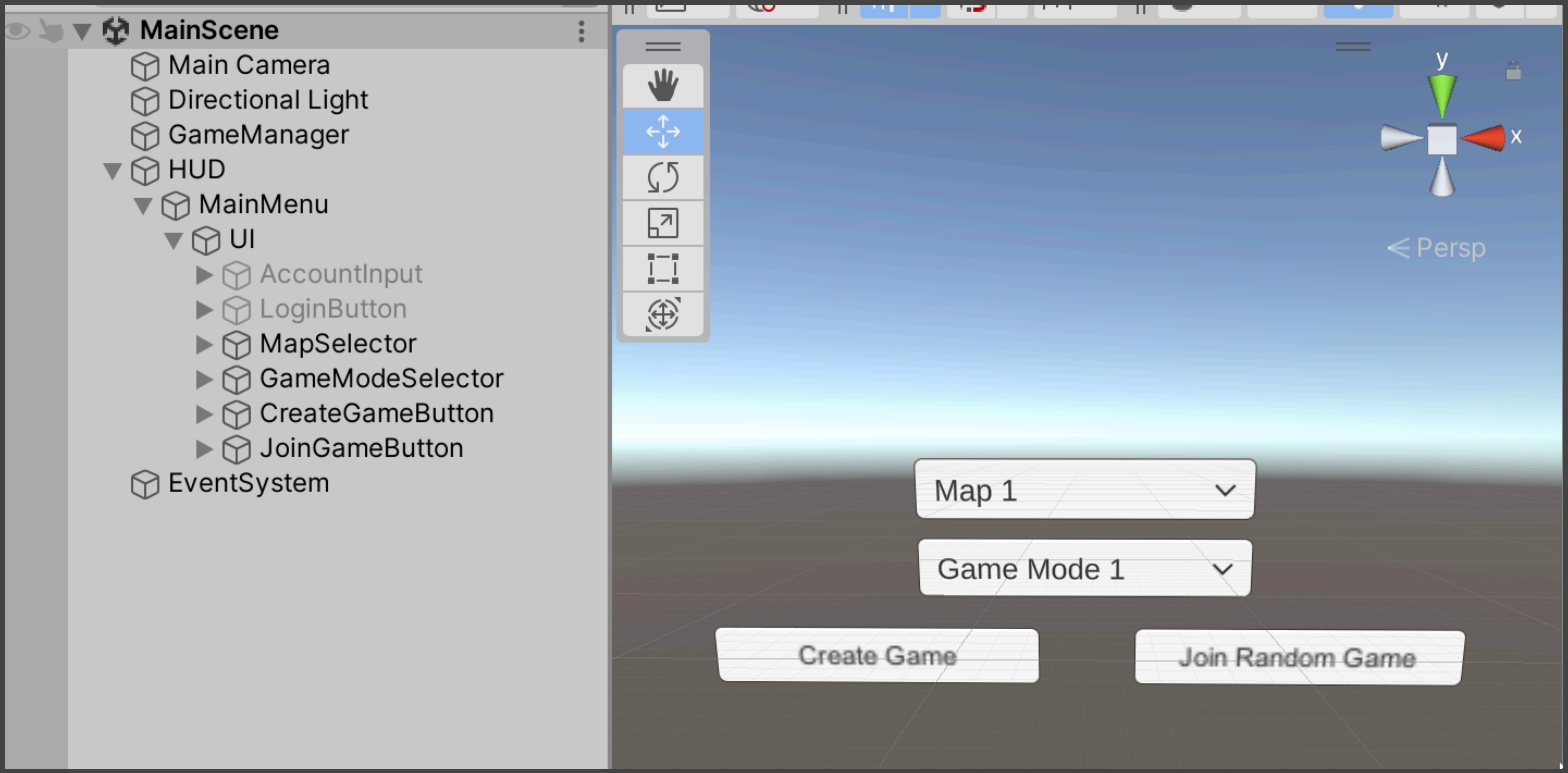
- 新增兩個UI
  - AccountInput
  - LoginButton



LoginButton Static
Tag Untagged Layer UI
Rect Transform
Canvas Renderer
Cull Transparent Mesh ✓
Image ✓
Button ✓
Interactable ✓
Transition Color Tint
Target Graphic LoginButton (Image)
Normal Color
Highlighted Color
Pressed Color
Selected Color
Disabled Color
Color Multiplier 1
Fade Duration 0.1
Navigation Automatic
Visualize
On Click ()
Runtime Only MainMenu.Login
MainMenu (Main

練習

Custom Room Properties Example



Custom Room Properties Example

```
public class GameManager : MonoBehaviourPunCallbacks
{
  public static GameManager instance;
  public static GameObject localPlayer;
  private GameObject defaultSpawnPoint;

  private const string MAP_PROP_KEY = "map";
  private const string GAME_MODE_PROP_KEY = "gm";
  private const string AI_PROP_KEY = "ai";

  string gameVersion = “1";
  …
```

Custom Room Properties Example

```
public void CreateGame(int map, int gameMode)
{
  var roomOptions = new RoomOptions();
  roomOptions.CustomRoomPropertiesForLobby = new[] { MAP_PROP_KEY, GAME_MODE_PROP_KEY, AI_PROP_KEY };
  roomOptions.CustomRoomProperties = new ExitGames.Client.Photon.Hashtable
    {
      { MAP_PROP_KEY, map },
      { GAME_MODE_PROP_KEY, gameMode }
    };
  roomOptions.MaxPlayers = 4;

  PhotonNetwork.CreateRoom(null, roomOptions, null);
}

public void JoinRandomGame(int map, int gameMode)
{
  byte expectedMaxPlayers = 0;
  var expectedCustomRoomProperties = new ExitGames.Client.Photon.Hashtable
    {
      { MAP_PROP_KEY, map },
      { GAME_MODE_PROP_KEY, gameMode }
    };
  PhotonNetwork.JoinRandomRoom(expectedCustomRoomProperties, expectedMaxPlayers);
}
```

Custom Room Properties Example

```
public override void OnJoinedRoom()
{
  Debug.Log($"Joined room: {PhotonNetwork.CurrentRoom.Name}
{PhotonNetwork.CurrentRoom.CustomProperties}");

  if (PhotonNetwork.IsMasterClient)
  {
    PhotonNetwork.LoadLevel("GameScene");
  }
}

public override void OnJoinRandomFailed(short returnCode, string
message)
{
  Debug.Log($"Join Random Room Failed: ({returnCode}) {message}");
}
```

Custom Room Properties Example

```
public class MainMenu : MonoBehaviourPunCallbacks
{
  public static MainMenu instance;
  private GameObject m_ui;

  private TMP_InputField  m_accountInput;
  private Button m_loginButton;

  private TMP_Dropdown m_mapSelector;
  private TMP_Dropdown m_gameModeSelector;
  private Button m_createGameButton;
  private Button m_joinGameButton;
```

Custom Room Properties Example

```
void Awake()
{
  if (instance != null)
  {
    DestroyImmediate(gameObject);
    return;
  }

  instance = this;

  m_ui = transform.FindAnyChild<Transform>("UI").gameObject;

  m_accountInput = transform.FindAnyChild<TMP_InputField>("AccountInput");
  m_loginButton = transform.FindAnyChild<Button>("LoginButton");

  m_mapSelector = transform.FindAnyChild<TMP_Dropdown>("MapSelector");
  m_gameModeSelector = transform.FindAnyChild<TMP_Dropdown>("GameModeSelector");
  m_createGameButton = transform.FindAnyChild<Button>("CreateGameButton");
  m_joinGameButton = transform.FindAnyChild<Button>("JoinGameButton");

  ResetUI();
}
```

Custom Room Properties Example

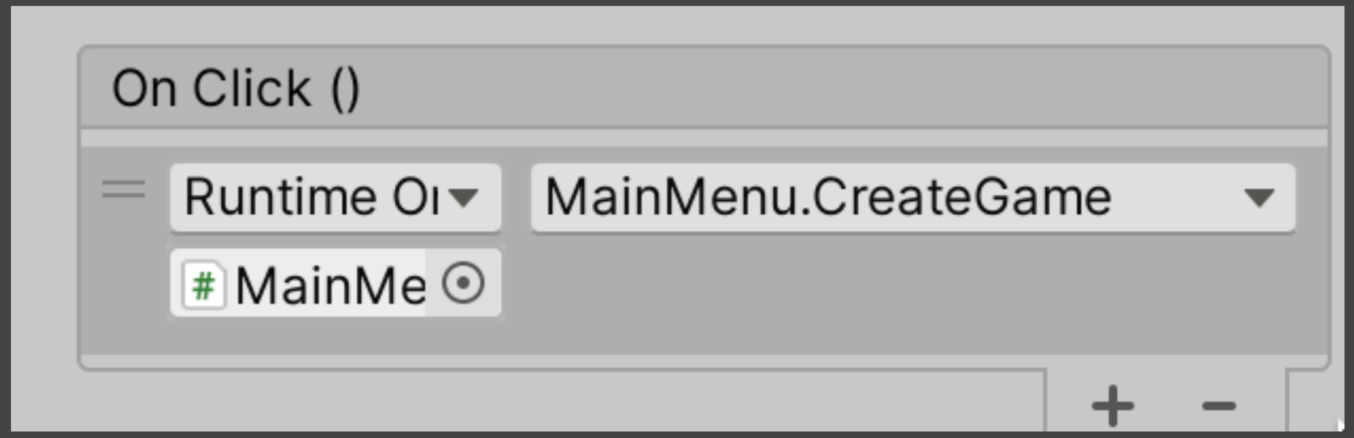
```
private void ResetUI()
{
  m_ui.SetActive(true);

  m_accountInput.gameObject.SetActive(true);
  m_loginButton.gameObject.SetActive(true);

  m_mapSelector.gameObject.SetActive(false);
  m_gameModeSelector.gameObject.SetActive(false);
  m_createGameButton.gameObject.SetActive(false);
  m_joinGameButton.gameObject.SetActive(false);

  m_accountInput.interactable = true;
  m_loginButton.interactable = true;
  m_mapSelector.interactable = true;
  m_gameModeSelector.interactable = true;
  m_createGameButton.interactable = true;
  m_joinGameButton.interactable = true;
}
```

Custom Room Properties Example

```
public void CreateGame()
{
  GameManager.instance.CreateGame(m_mapSelector.value + 1, m_gameModeSelector.value + 1);
}

public void JoinRandomGame()
{
  GameManager.instance.JoinRandomGame(m_mapSelector.value + 1, m_gameModeSelector.value + 1);
}

public override void OnConnectedToMaster()
{
  m_accountInput.gameObject.SetActive(false);
  m_loginButton.gameObject.SetActive(false);

  m_mapSelector.gameObject.SetActive(true);
  m_gameModeSelector.gameObject.SetActive(true);
  m_createGameButton.gameObject.SetActive(true);
  m_joinGameButton.gameObject.SetActive(true);
}
```

Custom Room Properties Example

- CreateGameButton

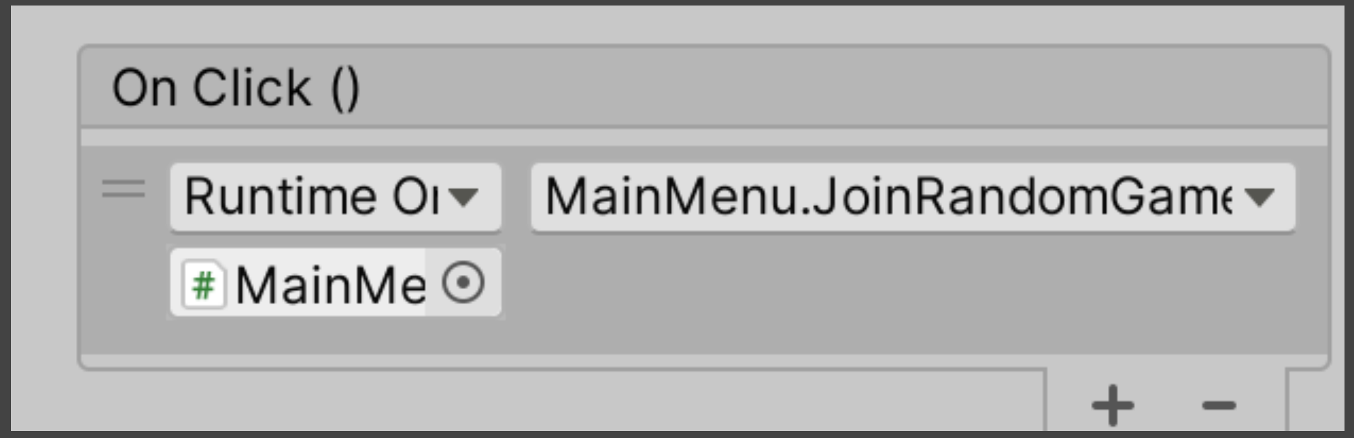


On Click ()

Runtime Or MainMenu.CreateGame

MainMe

- JoinGameButton



On Click ()

Runtime Or MainMenu.JoinRandomGame

MainMe

練習

Application Statistics

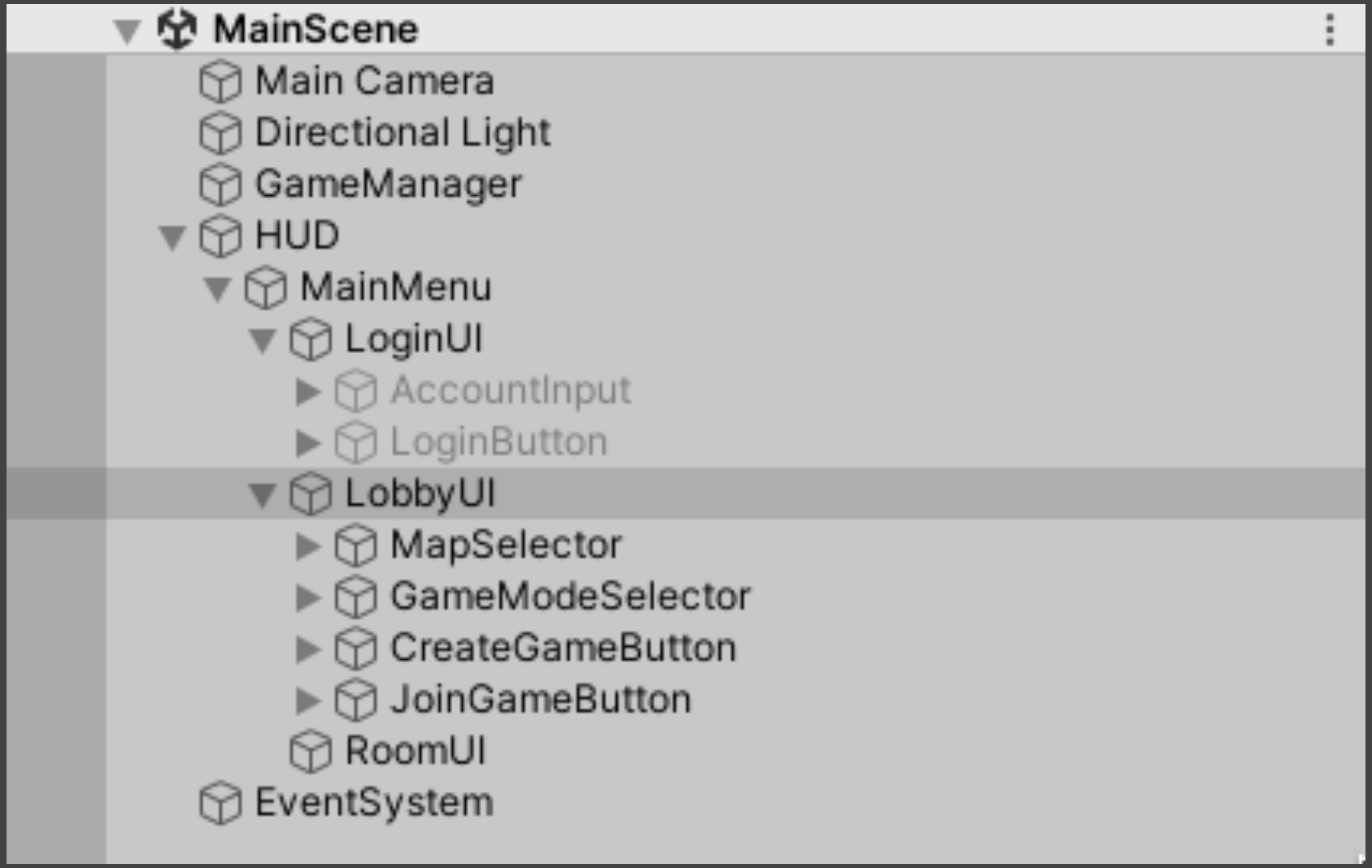
- 當連接到 Photon Master Server 時，Photon Client 就會收到 Application Statistics
- 不管 Client 是否加入 Lobby，都會收到 AppStats Event
- AppStats Event 每五秒更新一次

    - PhotonNetwork.CountOfPlayers：線上玩家總數
    - PhotonNetwork.CountOfRooms：目前房間數量
    - PhotonNetwork.CountOfPlayersInRooms：房間裡的玩家數量
    - PhotonNetwork.CountOfPlayersOnMaster：未加入房間的玩家數量

Application Statistics Example

Application Statistics Example

- 整理 UI 結構



```
▼ MainScene
    Main Camera
    Directional Light
    GameManager
  ▼ HUD
    ▼ MainMenu
      ▼ LoginUI
        ▶ AccountInput
        ▶ LoginButton
      ▼ LobbyUI
        ▶ MapSelector
        ▶ GameModeSelector
        ▶ CreateGameButton
        ▶ JoinGameButton
        RoomUI
    EventSystem
```

Application Statistics Example

```
public class MainMenu : MonoBehaviourPunCallbacks
{
  public static MainMenu instance;

  private GameObject m_loginUI;
  private TMP_InputField  m_accountInput;
  private Button m_loginButton;

  private GameObject m_lobbyUI;
  private TMP_Dropdown m_mapSelector;
  private TMP_Dropdown m_gameModeSelector;
  private Button m_createGameButton;
  private Button m_joinGameButton;

  private GameObject m_roomUI;
}
```

Application Statistics Example

```
void Awake()
{
  if (instance != null)
  {
    DestroyImmediate(gameObject);
    return;
  }

  instance = this;

  m_loginUI = transform.FindAnyChild<Transform>("LoginUI").gameObject;
  m_accountInput = transform.FindAnyChild<TMP_InputField>("AccountInput");
  m_loginButton = transform.FindAnyChild<Button>("LoginButton");

  m_lobbyUI = transform.FindAnyChild<Transform>("LobbyUI").gameObject;
  m_mapSelector = transform.FindAnyChild<TMP_Dropdown>("MapSelector");
  m_gameModeSelector = transform.FindAnyChild<TMP_Dropdown>("GameModeSelector");
  m_createGameButton = transform.FindAnyChild<Button>("CreateGameButton");
  m_joinGameButton = transform.FindAnyChild<Button>("JoinGameButton");

  m_roomUI = transform.FindAnyChild<Transform>("RoomUI").gameObject;

  ResetUI();
}
```

Application Statistics Example

```
private void ResetUI()
{
  m_loginUI.SetActive(true);
  m_accountInput.interactable = true;
  m_loginButton.interactable = true;

  m_lobbyUI.SetActive(false);
  m_mapSelector.interactable = true;
  m_gameModeSelector.interactable = true;
  m_createGameButton.interactable = true;
  m_joinGameButton.interactable = true;

  m_roomUI.SetActive(false);
}
```

Application Statistics Example

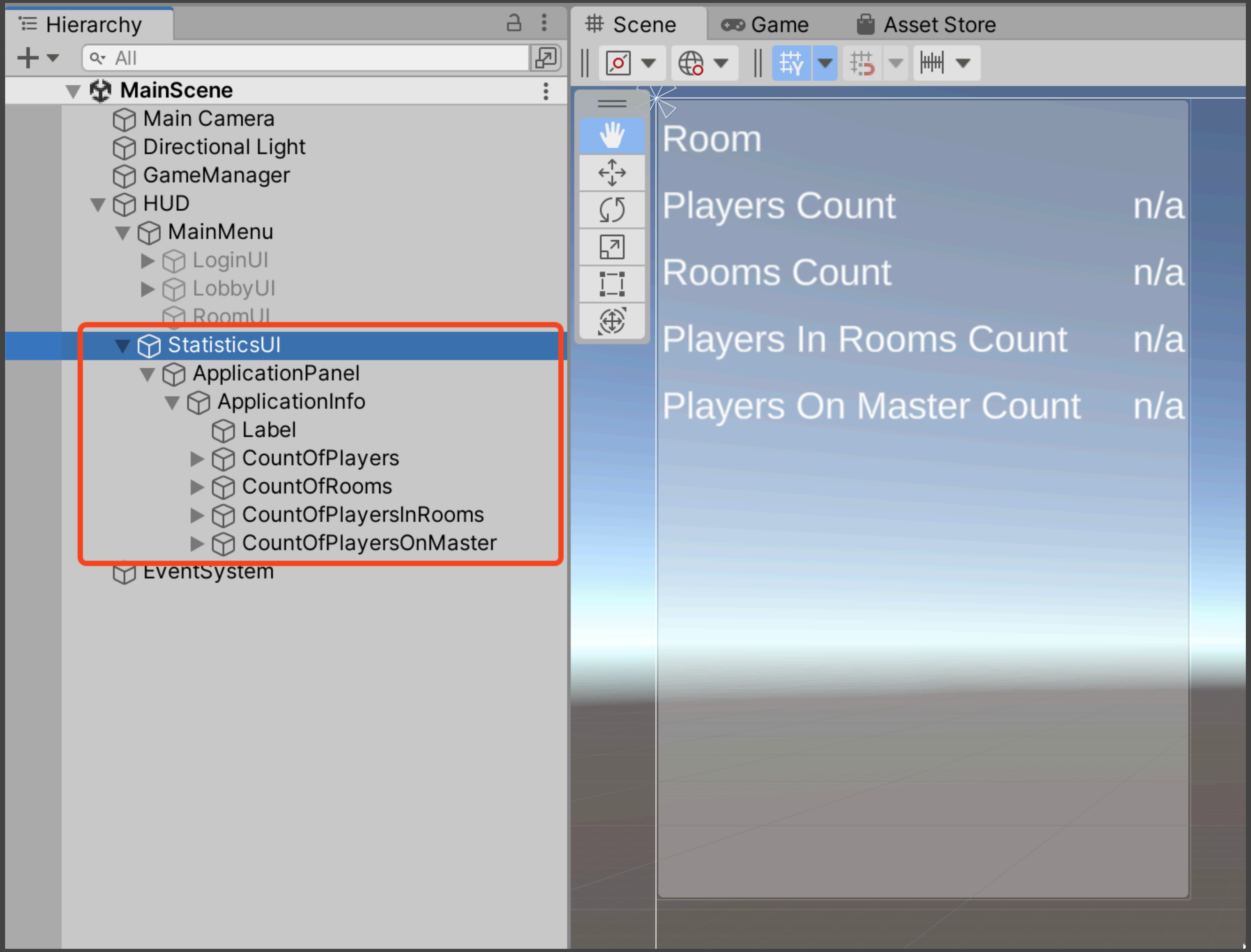
```
public void Login()
{
  if (string.IsNullOrEmpty(m_accountInput.text))
  {
    Debug.Log("Please input your account!!");
    return;
  }

  m_accountInput.interactable = false;
  m_loginButton.interactable = false;

  if (!GameManager.instance.ConnectToServer(m_accountInput.text))
  {
    Debug.Log("Connect to PUN Failed!!");
    m_accountInput.interactable = true;
    m_loginButton.interactable = true;
  }
}
```

Application Statistics Example

```
  public override void OnConnectedToMaster()
  {
    m_loginUI.SetActive(false);
    m_lobbyUI.SetActive(true);
  }

  public void CreateGame()
  {
    GameManager.instance.CreateGame(m_mapSelector.value + 1, m_gameModeSelector.value + 1);
  }

  public void JoinRandomGame()
  {
    GameManager.instance.JoinRandomGame(m_mapSelector.value + 1, m_gameModeSelector.value + 1);
  }

  private void OnSceneLoaded(Scene scene, LoadSceneMode mode)
  {
    if (!PhotonNetwork.InRoom)
    {
      ResetUI();
    }
    else
    {
      m_lobbyUI.SetActive(false);
      m_roomUI.SetActive(false);
    }
  }
}
```

練習

Application Statistics Example



MainScene
- Main Camera
- Directional Light
- GameManager
- HUD
  - MainMenu
    - LoginUI
    - LobbyUI
    - RoomUI
  - StatisticsUI
    - ApplicationPanel
      - ApplicationInfo
        - Label
        - CountOfPlayers
        - CountOfRooms
        - CountOfPlayersInRooms
        - CountOfPlayersOnMaster
- EventSystem

Room	
Players Count	n/a
Rooms Count	n/a
Players In Rooms Count	n/a
Players On Master Count	n/a

Application Statistics Example

```
public class StatisticsUI : MonoBehaviourPunCallbacks
{
  private int CountOfPlayers = -1;
  public TMP_Text CountOfPlayersText;

  private int CountOfRooms = -1;
  public TMP_Text CountOfRoomsText;

  private int CountOfPlayersInRooms = -1;
  public TMP_Text CountOfPlayersInRoomsText;

  private int CountOfPlayersOnMaster = -1;
  public TMP_Text CountOfPlayersOnMasterText;

  // Update is called once per frame
  void Update()
  {
    UpdateApplicationInfo();
  }
```

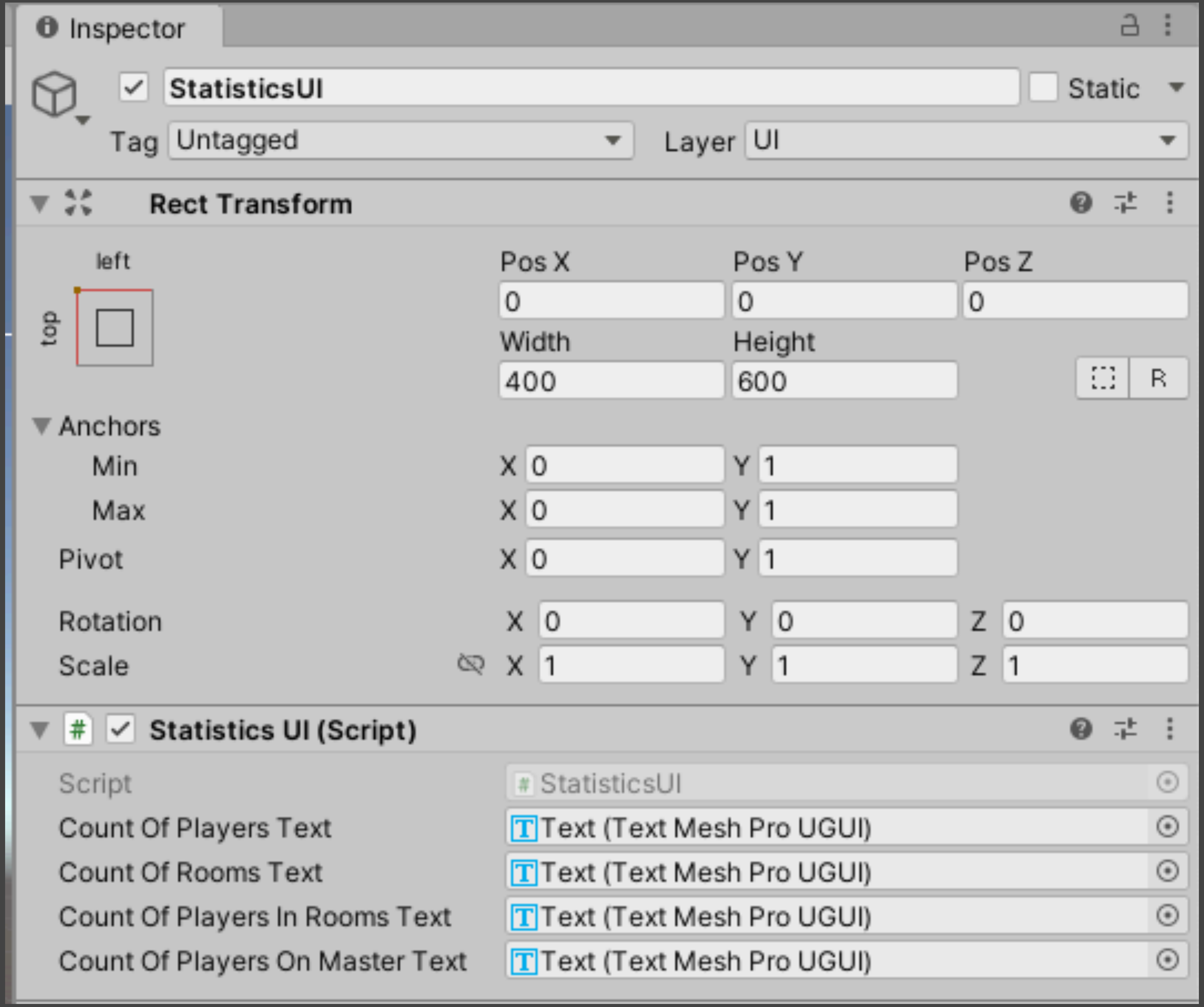
Application Statistics Example

```
private void UpdateApplicationInfo()
{
  if (PhotonNetwork.NetworkingClient.Server == ServerConnection.MasterServer)
  {
    RefreshApplicationInfo();
  }
  else
  {
    ResetApplicationInfo();
  }
}

private void ResetApplicationInfo()
{
  if (CountOfPlayers != -1)
  {
    CountOfPlayers = -1;
    CountOfPlayersText.text = "n/a";
  }
  if (CountOfRooms != -1)
  {
    CountOfRooms = -1;
    CountOfRoomsText.text = "n/a";
  }
  if (CountOfPlayersInRooms != -1)
  {
    CountOfPlayersInRooms = -1;
    CountOfPlayersInRoomsText.text = "n/a";
  }
  if (CountOfPlayersOnMaster != -1)
  {
    CountOfPlayersOnMaster = -1;
    CountOfPlayersOnMasterText.text = "n/a";
  }
}
```

Application Statistics Example

```
  private void RefreshApplicationInfo()
  {
    if (!PhotonNetwork.IsConnected)
    {
      ResetApplicationInfo();
      return;
    }

    if (CountOfPlayers != PhotonNetwork.CountOfPlayers)
    {
      CountOfPlayers = PhotonNetwork.CountOfPlayers;
      CountOfPlayersText.text = CountOfPlayers.ToString();
    }

    if (CountOfRooms != PhotonNetwork.CountOfRooms)
    {
      CountOfRooms = PhotonNetwork.CountOfRooms;
      CountOfRoomsText.text = CountOfRooms.ToString();
    }

    if (CountOfPlayersInRooms != PhotonNetwork.CountOfPlayersInRooms)
    {
      CountOfPlayersInRooms = PhotonNetwork.CountOfPlayersInRooms;
      CountOfPlayersInRoomsText.text = CountOfPlayersInRooms.ToString();
    }

    if (CountOfPlayersOnMaster != PhotonNetwork.CountOfPlayersOnMaster)
    {
      CountOfPlayersOnMaster = PhotonNetwork.CountOfPlayersOnMaster;
      CountOfPlayersOnMasterText.text = CountOfPlayersOnMaster.ToString();
    }
  }
}
```

Application Statistics Example



Inspector

StatisticsUI — Static

Tag: Untagged — Layer: UI

Rect Transform

left / top

	Pos X	Pos Y	Pos Z
	0	0	0
	Width	Height	
	400	600	
Anchors			
Min	X 0	Y 1	
Max	X 0	Y 1	
Pivot	X 0	Y 1	
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Statistics UI (Script)

Script	StatisticsUI
Count Of Players Text	Text (Text Mesh Pro UGUI)
Count Of Rooms Text	Text (Text Mesh Pro UGUI)
Count Of Players In Rooms Text	Text (Text Mesh Pro UGUI)
Count Of Players On Master Text	Text (Text Mesh Pro UGUI)

練習

Lobby 大廳

- Photon的房間是使用「大廳(Lobby) 」來做群組
- 大廳是通過 Name 和 Type 來識別的
    - 名稱：可以是任何字串
    - 類型：Default、SQL 以及 Asynchronous Type
- 所有的Photon App都有一個預設的大廳(Default Lobby)
- 也可以創立其他大廳
    - 如果在呼叫下面函示時，指定一個新的 lobby definition 時，大廳就會自動建立
    - JoinLobby、CreateRoom 或 JoinOrCreateRoom

Lobby 大廳

- 像房間一樣，可以加入大廳(JoinLobby)、離開大廳(LeaveLobby)

```
PhotonNetwork.JoinLobby();
PhotonNetwork.LeaveLobby();

public override void OnJoinedLobby() {}
public override void OnLeftLobby() {}
```

- 是否在大廳中

```
bool inLobby = PhotonNetwork.InLobby;
```

- 目前的大廳

```
TypedLobby lobby = PhotonNetwork.CurrentLobby;
```

- 大廳只能用來列出房間列表，沒有其他功能
  - 在大廳裡沒有辦法與其他人交談
- 如果要切換大廳的話，直接呼叫 JoinLobby 加入另一個大廳就好，不用先離開目前的大廳

Default Lobby Type

- 最適合同步隨機匹配的類型
- 如果加入到 Default 類型的大廳時，客戶端就會定期收到房間列表的更新 (OnRoomListUpdate Event)
- 收到的房間列表會用兩個標準來進行排序：
    - open or closed, full or not
    - 所以房間會分成這三組順序
        1. open and not full (joinable)
        2. full but not closed (not joinable)
        3. closed (not joinable, could be full or not)
    - 在每組中，沒有任何特定的順序（隨機）

Default Lobby

- Photon 預設有一個 null name 的大廳，類型為 Default Lobby Type
- 在C# SDKs中，定義為
  ```
  TypedLobby.Default
  ```
- Default Lobby 的名字是保留的
  - 只有 Default Lobby 可以為空的名字
  - 所有其他的大廳都需要指定名字。
  - 如果使用一個 null 或 空字串 作為大廳的名字，不管指定的類型，都是指 Default Lobby
- 加入 Default Lobby 的方式：不傳參數
  ```
  PhotonNetwork.JoinLobby();
  ```

大廳房間規則(指定房間)

```
public static bool JoinOrCreateRoom(string roomName, RoomOptions roomOptions,
  TypedLobby typedLobby, string[] expectedUsers = null)
```

- 假設客戶已經在某一個大廳
  - 呼叫 JoinOrCreate 來加入房間時，沒有指定大廳的參數
    - 該房間會被加入到當前的大廳
  - 呼叫 JoinOrCreate 來加入房間時，有明確指定大廳的參數
    - 沒指定大廳名稱：該房間會被加入到當前的大廳
    - 有指定大廳名稱：該房間會被加入到指定的大廳
- 客戶沒有在任一個大廳時
  - 呼叫 JoinOrCreate 來加入房間時，沒有指定大廳的參數
    - 該房間會被加入到 Default Lobby
  - 呼叫 JoinOrCreate 來加入房間時，有明確指定大廳的參數
    - 沒指定大廳名稱：該房間會被加入到 Default Lobby
    - 有指定大廳名稱：該房間會被加入到指定的大廳

```
public class TypedLobby
{
  public string Name;
  public LobbyType Type;
  // Default, SqlLobby, AsyncRandomLobby
}
```

大廳房間規則(隨機)

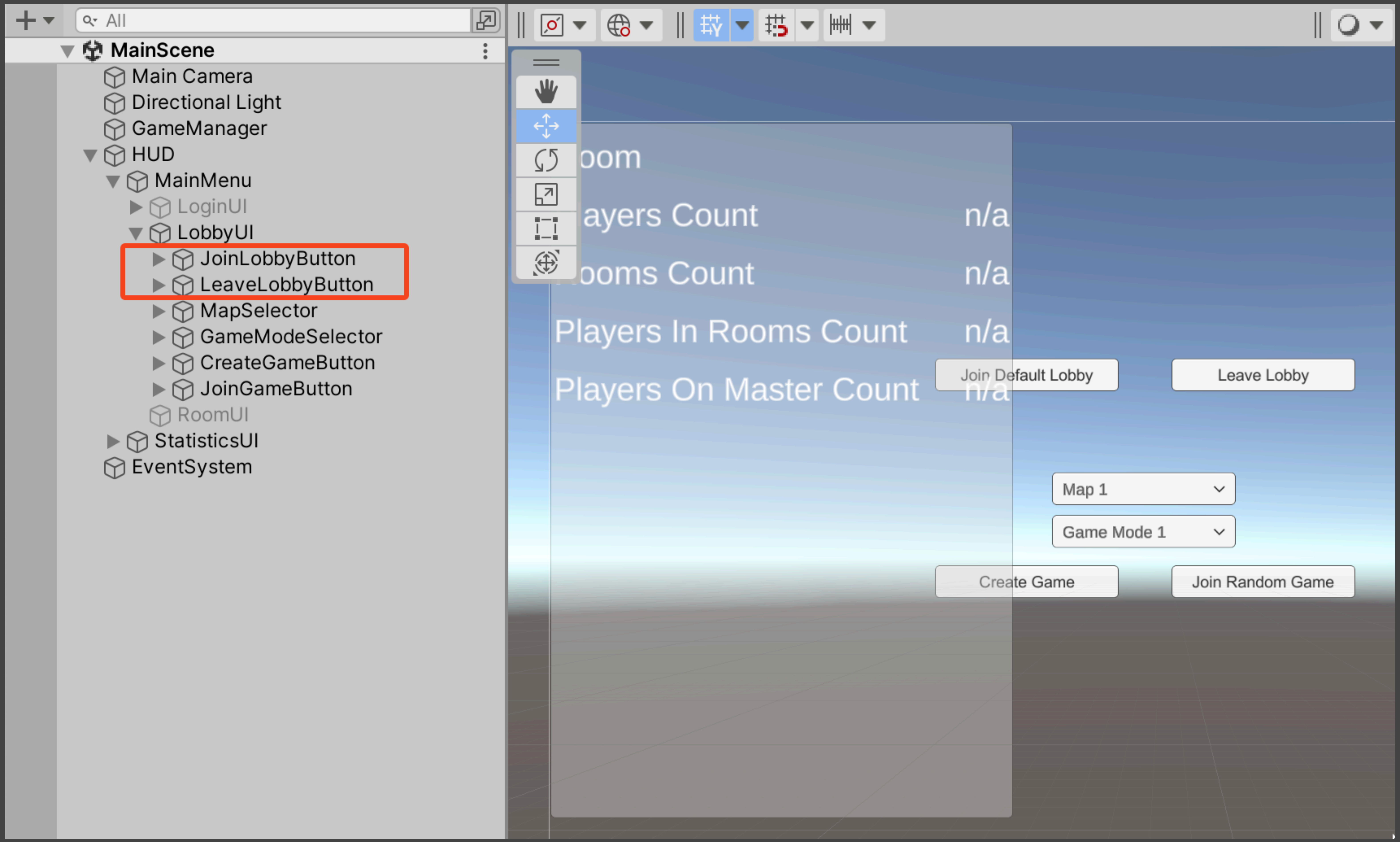
```
public static bool JoinOrCreateRoom(string roomName, RoomOptions roomOptions,
  TypedLobby typedLobby, string[] expectedUsers = null)
```

- 假設客戶已經在某一個大廳
  - 要加入一個隨機的房間，沒有指定大廳的參數
    - 會在目前加入的大廳中尋找房間
  - 要加入一個隨機的房間，有明確指定大廳的參數
    - 沒指定大廳名稱：會在目前加入的大廳中尋找房間
    - 有指定大廳名稱：會在所指定的大廳中尋找房間
- 客戶沒有在任一個大廳時
  - 要加入一個隨機的房間，沒有指定大廳的參數
    - 會在 Default Lobby 中尋找房間
  - 要加入一個隨機的房間，有明確指定大廳的參數
    - 沒指定大廳名稱：會在 Default Lobby 中尋找房間
    - 有指定大廳名稱：會在所指定的大廳中尋找房間

```
public class TypedLobby
{
  public string Name;
  public LobbyType Type;
  // Default, SqlLobby, AsyncRandomLobby
}
```

Default Lobby Example

Default Lobby Example



Default Lobby Example

```
public class MainMenu : MonoBehaviourPunCallbacks
{
  …

  private GameObject m_lobbyUI;
  private Button m_joinLobbyButton;
  private Button m_leaveLobbyButton;
  …

  void Awake()
  {
    …

    m_lobbyUI = transform.FindAnyChild<Transform>("LobbyUI").gameObject;
    m_joinLobbyButton = transform.FindAnyChild<Button>("JoinLobbyButton");
    m_leaveLobbyButton = transform.FindAnyChild<Button>("LeaveLobbyButton");
    …
  }

  private void ResetUI()
  {
    …

    m_lobbyUI.SetActive(false);
    m_joinLobbyButton.interactable = true;
    m_leaveLobbyButton.interactable = false;

    …
  }
```

Default Lobby Example

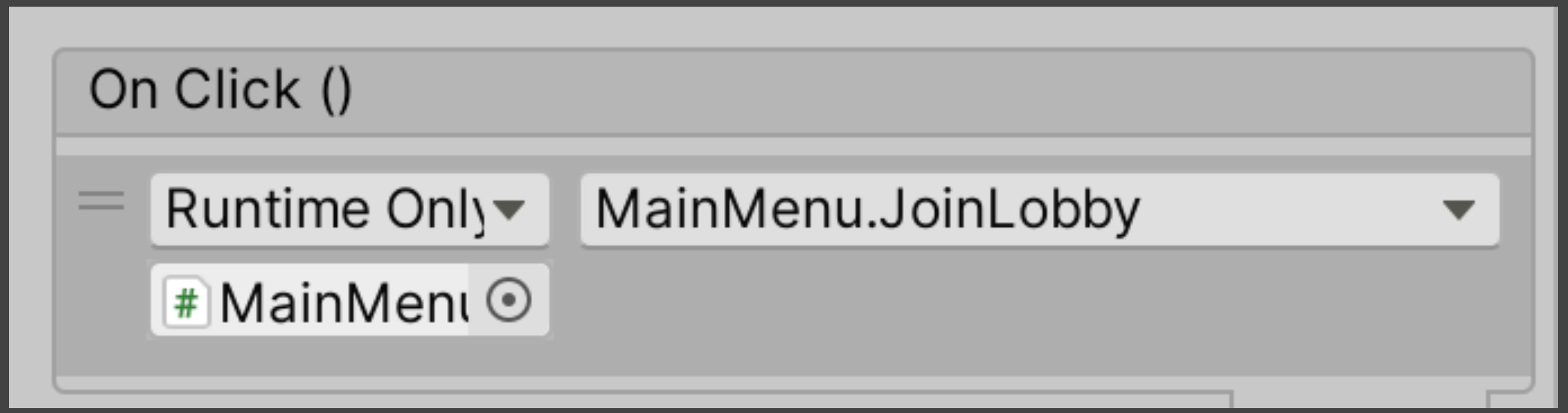
```
public void JoinLobby()
{
  PhotonNetwork.JoinLobby();
}

public void LeaveLobby()
{
  PhotonNetwork.LeaveLobby();
}

public override void OnJoinedLobby()
{
  Debug.Log($"Joined Lobby: {PhotonNetwork.CurrentLobby.Name} {PhotonNetwork.CurrentLobby.Type}");
  m_joinLobbyButton.interactable = false;
  m_leaveLobbyButton.interactable = true;
}

public override void OnLeftLobby()
{
  // 離開 Lobby 的時候，會加回 Default Lobby
  Debug.Log($"Left Lobby: {PhotonNetwork.CurrentLobby.Name} {PhotonNetwork.CurrentLobby.Type}");
  m_joinLobbyButton.interactable = true;
  m_leaveLobbyButton.interactable = false;
}

…
```

Default Lobby Example

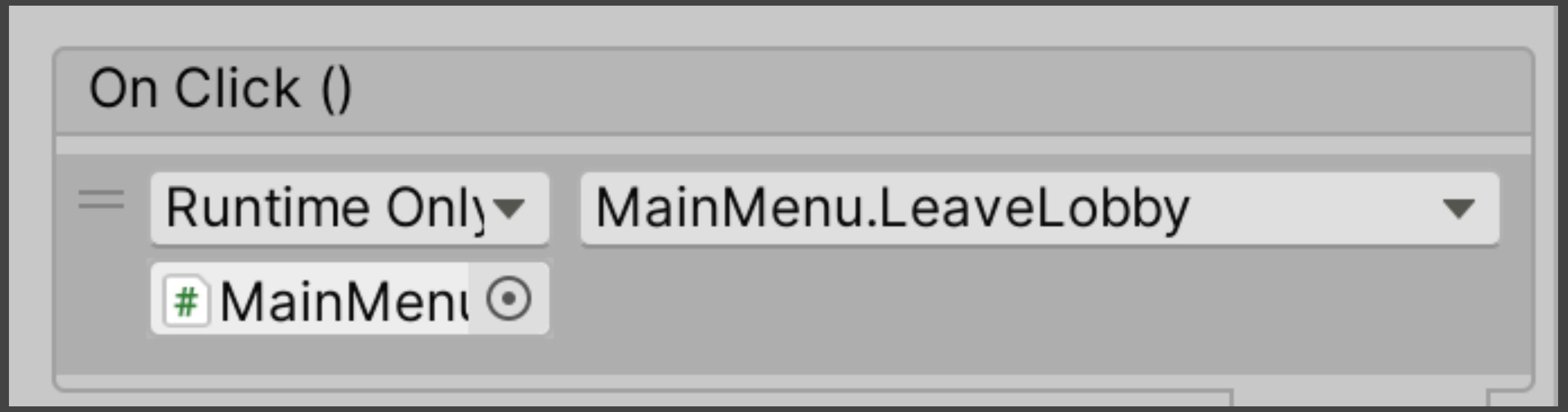
```
    …

    public override void OnRoomListUpdate(List<RoomInfo> roomList)
    {
      var message = $"Room List: {roomList.Count} rooms\n";
      foreach (var roomInfo in roomList)
      {
        message += $"  {roomInfo.Name}, {roomInfo.IsOpen}, {roomInfo.PlayerCount}/{roomInfo.MaxPlayers}";
      }

      Debug.Log(message);
    }

    …
  }
```

Default Lobby Example

- JoinLobbyButton


On Click ()
Runtime Only MainMenu.JoinLobby
MainMenu

- LeaveLobbyButton

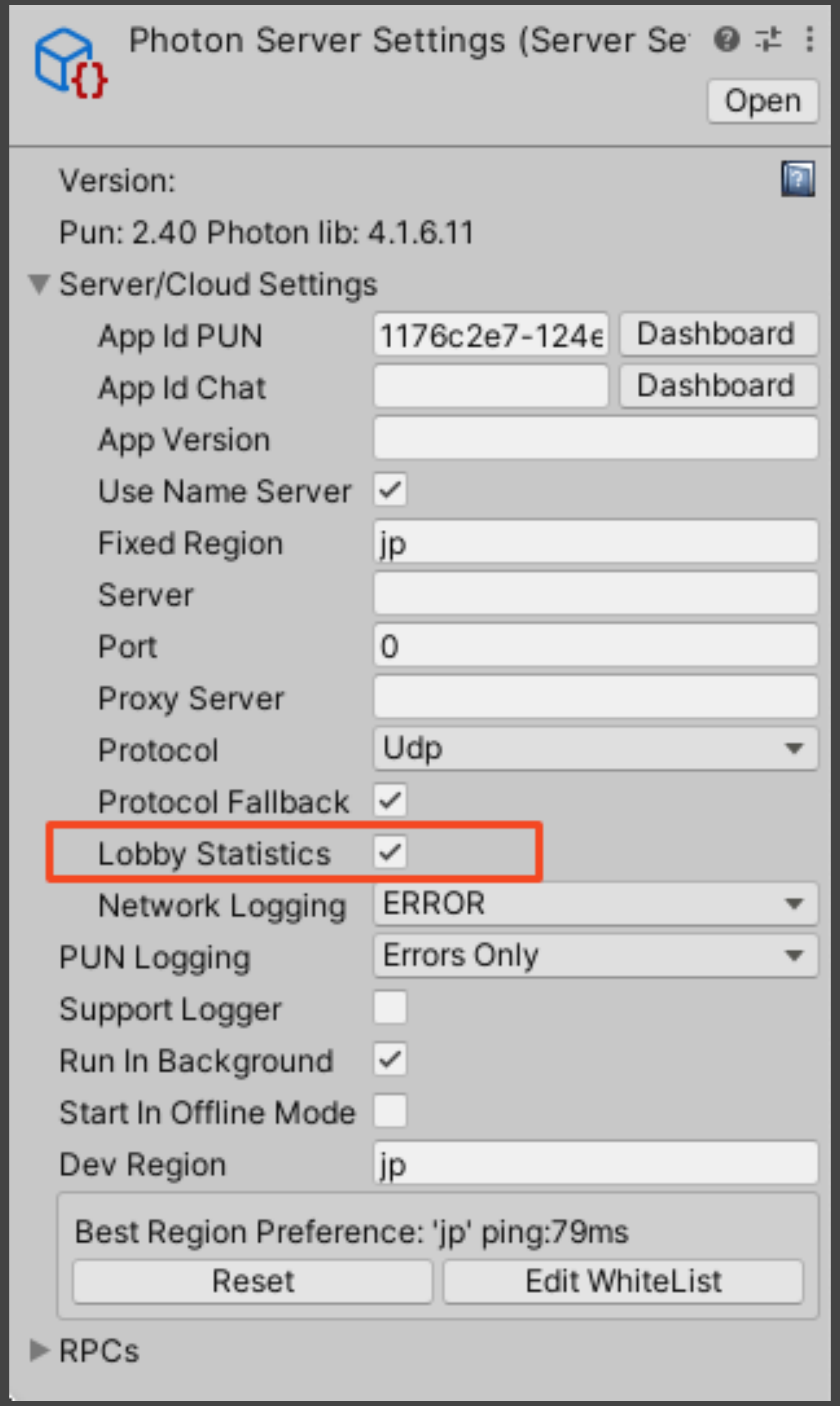

On Click ()
Runtime Only MainMenu.LeaveLobby
MainMenu

練習

Lobby Statistics

- 如果遊戲建立了很多大廳，可以使用 Lobby Statistics 來顯示大廳的情況
- 每個 Region 的大廳是分開統計
- 必須開啟 Photon Server Settings 的 Lobby Statistics 設定
- 可以透過下面事件取得所有大廳狀態

```
void OnLobbyStatisticsUpdate(List<TypedLobbyInfo> lobbyStatistics);

public class TypedLobbyInfo : TypedLobby
{
  public int PlayerCount; // 玩家數量
  public int RoomCount;   // 房間數量
}

public class TypedLobby
{
  public string Name;
  public LobbyType Type; // Default, SqlLobby, AsyncRandomLobby
}
```



Version:
Pun: 2.40 Photon lib: 4.1.6.11

▼ Server/Cloud Settings

App Id PUN	1176c2e7-124e
App Id Chat	
App Version	
Use Name Server	✓
Fixed Region	jp
Server	
Port	0
Proxy Server	
Protocol	Udp
Protocol Fallback	✓
Lobby Statistics	✓
Network Logging	ERROR
PUN Logging	Errors Only
Support Logger	
Run In Background	✓
Start In Offline Mode	
Dev Region	jp

Best Region Preference: 'jp' ping:79ms

Lobby Statistics Example

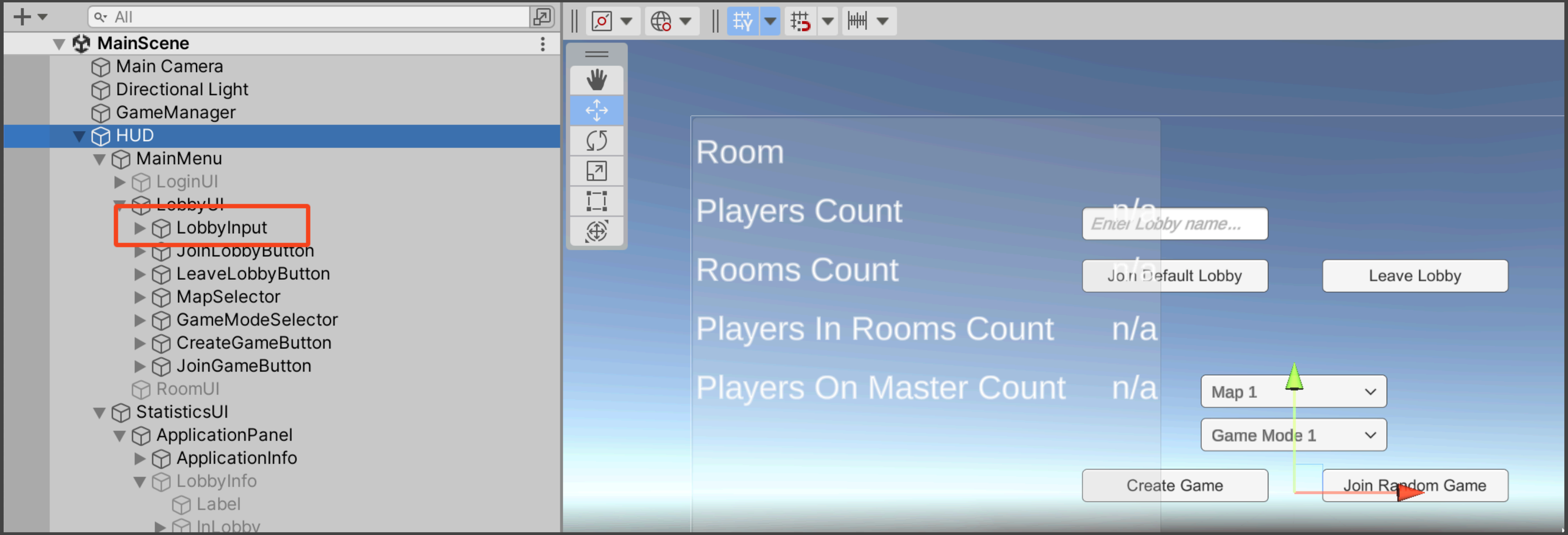
Lobby Statistics Example

```
public override void OnLobbyStatisticsUpdate(List<TypedLobbyInfo> lobbyStatistics)
{
  var message = $"Lobby List: {lobbyStatistics.Count} lobbies\n";
  foreach (var lobbyInfo in lobbyStatistics)
  {
    message += $"  {lobbyInfo.Name}, {lobbyInfo.Type}, {lobbyInfo.RoomCount} rooms, " +
               $"{lobbyInfo.PlayerCount} players\n";
  }

  Debug.Log(message);
}
```

練習

Named Default Type Lobby Example

Named Default Type Lobby Example



Named Default Type Lobby Example

```
public class MainMenu : MonoBehaviourPunCallbacks
{
  …

  private GameObject m_lobbyUI;
  private TMP_InputField m_lobbyInput;
  private Button m_joinLobbyButton;
  private Button m_leaveLobbyButton;
  …

  private Dictionary<string, RoomInfo> cachedRoomList = new Dictionary<string, RoomInfo>();

  void Awake()
  {
    …

    m_lobbyUI = transform.FindAnyChild<Transform>("LobbyUI").gameObject;
    m_lobbyInput = transform.FindAnyChild<TMP_InputField>("LobbyInput");
    m_joinLobbyButton = transform.FindAnyChild<Button>("JoinLobbyButton");
    m_leaveLobbyButton = transform.FindAnyChild<Button>("LeaveLobbyButton");
    …
  }

  private void ResetUI()
  {
    …

    m_lobbyUI.SetActive(false);
    m_lobbyInput.interactable = true;
    m_joinLobbyButton.interactable = true;
    m_leaveLobbyButton.interactable = false;
    …
    cachedRoomList.Clear();
  }
```

Named Default Type Lobby Example

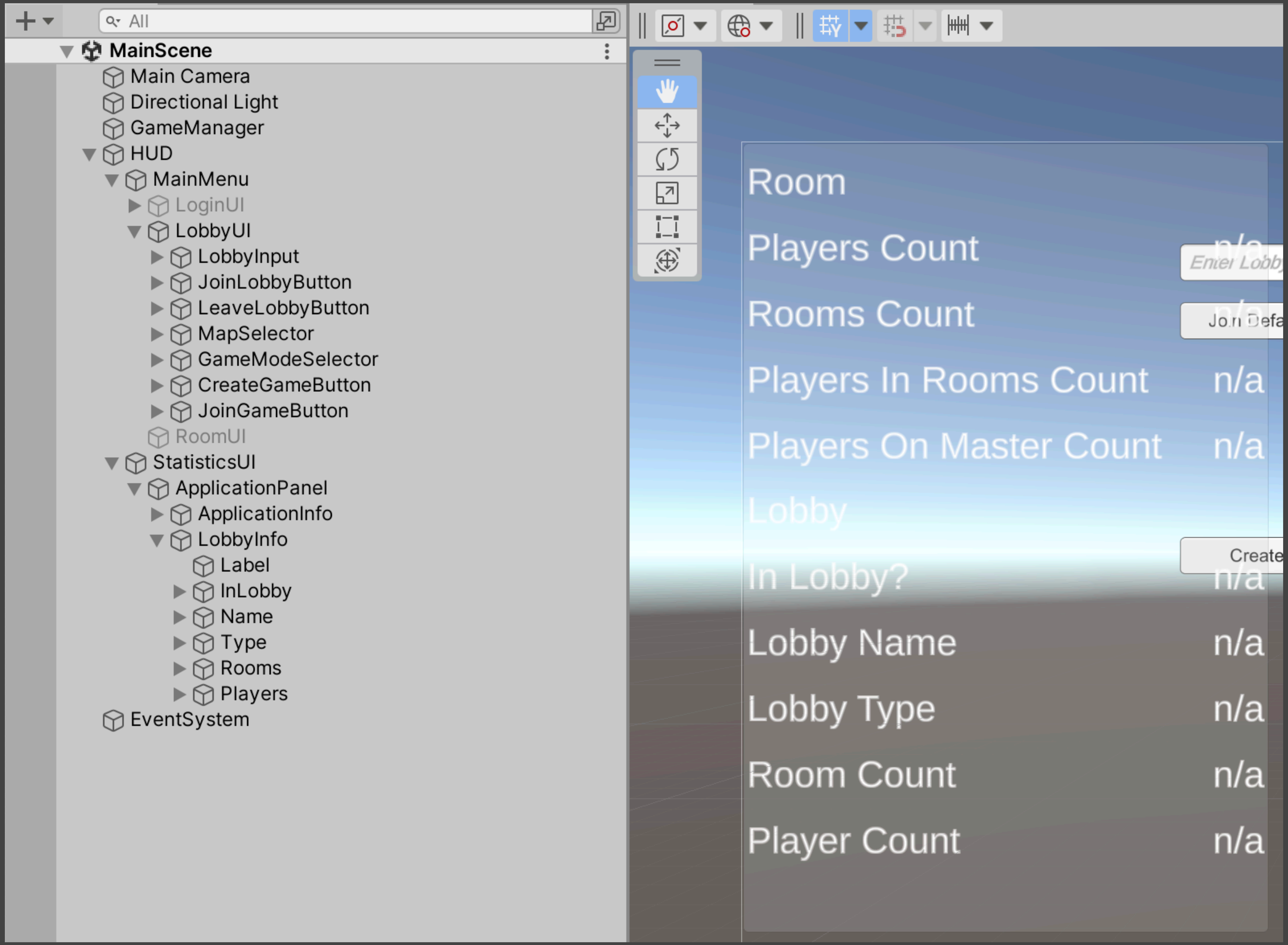
```
    …
    public void JoinLobby()
    {
      cachedRoomList.Clear();

      var typedLobby = new TypedLobby(m_lobbyInput.text, LobbyType.Default);
      PhotonNetwork.JoinLobby(typedLobby);
    }

    public void LeaveLobby()
    {
      PhotonNetwork.LeaveLobby();
    }

    public override void OnJoinedLobby()
    {
      Debug.Log($"Joined Lobby: {PhotonNetwork.CurrentLobby.Name} {PhotonNetwork.CurrentLobby.Type}");
      m_leaveLobbyButton.interactable = true;
      cachedRoomList.Clear();
    }

    public override void OnLeftLobby()
    {
      // 離開 Lobby 的時候，會加回 Default Lobby
      Debug.Log($"Left Lobby: {PhotonNetwork.CurrentLobby.Name} {PhotonNetwork.CurrentLobby.Type}");
      m_leaveLobbyButton.interactable = false;
      cachedRoomList.Clear();
    }
    …
  }
```

Named Default Type Lobby Example

```
  …
  public override void OnRoomListUpdate(List<RoomInfo> roomList)
  {
    UpdateCachedRoomList(roomList);
    printRoomList();
  }

  private void UpdateCachedRoomList(List<RoomInfo> roomList)
  {
    for(int i=0; i < roomList.Count; i++)
    {
      RoomInfo info = roomList[i];
      if (info.RemovedFromList) // 不紀錄已關閉、滿了、或是隱藏的房間
        cachedRoomList.Remove(info.Name);
      else
        cachedRoomList[info.Name] = info;
    }
  }

  private void printRoomList()
  {
    var message = $"Room List: {cachedRoomList.Count} rooms\n";
    foreach (var roomInfo in cachedRoomList)
    {
      message += $"  {roomInfo.Key}, {roomInfo.Value.IsOpen}, " +
                 $"{roomInfo.Value.PlayerCount}/{roomInfo.Value.MaxPlayers}\n";
    }

    Debug.Log(message);
  }
  …
}
```

練習

Named Default Type Lobby Example



```
▼ MainScene
    Main Camera
    Directional Light
    GameManager
  ▼ HUD
    ▼ MainMenu
      ▶ LoginUI
      ▼ LobbyUI
        ▶ LobbyInput
        ▶ JoinLobbyButton
        ▶ LeaveLobbyButton
        ▶ MapSelector
        ▶ GameModeSelector
        ▶ CreateGameButton
        ▶ JoinGameButton
        RoomUI
    ▼ StatisticsUI
      ▼ ApplicationPanel
        ▶ ApplicationInfo
        ▼ LobbyInfo
            Label
          ▶ InLobby
          ▶ Name
          ▶ Type
          ▶ Rooms
          ▶ Players
    EventSystem
```

```
Room
Players Count                n/a
Rooms Count                  n/a
Players In Rooms Count       n/a
Players On Master Count      n/a
Lobby
In Lobby?                    n/a
Lobby Name                   n/a
Lobby Type                   n/a
Room Count                   n/a
Player Count                 n/a
```

Named Default Type Lobby Example

```
public class StatisticsUI : MonoBehaviourPunCallbacks
{
  …
  private bool InLobby = false;
  public TMP_Text InLobbyText;

  private string LobbyName;
  public TMP_Text LobbyNameText;

  private LobbyType? LobbyType = null;
  public TMP_Text LobbyTypeText;

  private int CountOfRoomOnLobby = -1;
  public TMP_Text CountOfRoomOnLobbyText;

  private int CountOfPlayerOnLobby = -1;
  public TMP_Text CountOfPlayerOnLobbyText;

  private Dictionary<string, RoomInfo> cachedRoomList = new Dictionary<string, RoomInfo>();

  // Start is called before the first frame update
  void Start()
  {
    cachedRoomList.Clear();
  }
```

Named Default Type Lobby Example

```
public override void OnJoinedLobby()
{
  cachedRoomList.Clear();
}

public override void OnLeftLobby()
{
  cachedRoomList.Clear();
}

public override void OnRoomListUpdate(List<RoomInfo> roomList)
{
  UpdateCachedRoomList(roomList);
  printRoomList();
}

// Update is called once per frame
void Update()
{
  UpdateApplicationInfo();
  UpdateLobbyInfo();
}
```

Named Default Type Lobby Example

```
…
private void UpdateCachedRoomList(List<RoomInfo> roomList)
{
  for(int i=0; i < roomList.Count; i++)
  {
    RoomInfo info = roomList[i];
    if (info.RemovedFromList) // 不紀錄已關閉、滿了、或是隱藏的房間
      cachedRoomList.Remove(info.Name);
    else
      cachedRoomList[info.Name] = info;
  }
}

private void printRoomList()
{
  var message = $"Room List: {cachedRoomList.Count} rooms\n";
  foreach (var roomInfo in cachedRoomList)
  {
    message += $"  {roomInfo.Key}, {roomInfo.Value.IsOpen}, " +
               $"{roomInfo.Value.PlayerCount}/{roomInfo.Value.MaxPlayers}\n";
  }

  Debug.Log(message);
}
```

Named Default Type Lobby Example

```
private void UpdateLobbyInfo()
{
  if (InLobby != PhotonNetwork.InLobby)
  {
    InLobby = PhotonNetwork.InLobby;
    InLobbyText.text = InLobby ? "true" : "false";
  }

  if (!InLobby || PhotonNetwork.CurrentLobby == null)
  {
    ResetLobbyInfo();
  }
  else
  {
    if (LobbyName != PhotonNetwork.CurrentLobby.Name)
    {
      LobbyName = PhotonNetwork.CurrentLobby.Name;
      LobbyNameText.text = LobbyName;
    }

    if (LobbyType != PhotonNetwork.CurrentLobby.Type)
    {
      LobbyType = PhotonNetwork.CurrentLobby.Type;
      LobbyTypeText.text = LobbyType.ToString();
    }

    if (CountOfRoomOnLobby != cachedRoomList.Count)
    {
      CountOfRoomOnLobby = cachedRoomList.Count;
      CountOfRoomOnLobbyText.text = CountOfRoomOnLobby.ToString();
    }

    var count = cachedRoomList.Sum(keyValuePair => keyValuePair.Value.PlayerCount);
    if (CountOfPlayerOnLobby != count)
    {
      CountOfPlayerOnLobby = count;
      CountOfPlayerOnLobbyText.text = count.ToString();
    }
  }
}
```

Named Default Type Lobby Example

```
  private void ResetLobbyInfo()
  {
    if (!string.IsNullOrEmpty(LobbyName))
    {
      LobbyName = null;
      LobbyNameText.text = "n/a";
    }

    if (LobbyType != null)
    {
      LobbyType = null;
      LobbyNameText.text = "n/a";
    }

    if (CountOfRoomOnLobby != -1)
    {
      CountOfRoomOnLobby = -1;
      CountOfRoomOnLobbyText.text = "n/a";
    }

    if (CountOfPlayerOnLobby != -1)
    {
      CountOfPlayerOnLobby = -1;
      CountOfPlayerOnLobbyText.text = "n/a";
    }
  }
}
```

練習

SQL Lobby Type

- 在 SQL Lobby Type 中，JoinRandomRoom中改成用「SQL 字串」來過濾(Filters)
- 並且無法使用定期更新房間列表(OnRoomListUpdate)的機制，而是要用 Custom Room Listing 的方式去取得列表(使用 GetCustomRoomList 函式)
- SQL Lobby Type 可以使用複雜的匹配來過濾房間，例如根據等級、技能
- SQL Lobby 將房間保存在一個 SQLite Table 中，該表最多有10個特殊的「SQL過濾屬性」
    - 這些SQL屬性的命名被固定為 "C0"、"C1 "到 “C9"，只允許整數類型和字符串類型的值
    - 儘管有屬性名稱是寫死的，但可以自己定義哪些是大廳中需要的，以及其含義
- 在建立或加入房間後，仍然可以使用SQL屬性以外的自定義房間屬性(Custom Room Properties) 以及 visible屬性，但是就不能用來做配對條件。

使用 SQL 語句過濾房間

```
public const string ELO_PROP_KEY = “C0"; // 難度
public const string MAP_PROP_KEY = “C3"; // 地圖類型
private TypedLobby sqlLobby = new TypedLobby("race", LobbyType.SqlLobby);

private void CreateRoom()
{
  RoomOptions roomOptions = new RoomOptions();
  roomOptions.CustomRoomProperties
     = new ExitGames.Client.Photon.Hashtable { { ELO_PROP_KEY, 400 }, { MAP_PROP_KEY, "Map3" } };
  // makes "C0" and "C3" available in the lobby
  roomOptions.CustomRoomPropertiesForLobby = new [] { ELO_PROP_KEY, MAP_PROP_KEY };

  PhotonNetwork.CreateRoom(null, roomOptions, sqlLobby);
}

private void JoinRandomRoom()
{
  string sqlLobbyFilter = "C0 BETWEEN 345 AND 475 AND C3 = 'Map2'";
  //string sqlLobbyFilter = "C0 > 345 AND C0 < 475 AND (C3 = 'Map2' OR C3 = \"Map3\")";
  //string sqlLobbyFilter = "C0 >= 345 AND C0 <= 475 AND C3 IN ('Map1', 'Map2', 'Map3')";
  PhotonNetwork.JoinRandomRoom(null, 0, MatchmakingMode.FillRoom, sqlLobby, sqlLobbyFilter);
}
```

使用 SQL 語句過濾房間

```
public override void OnJoinRandomFailed(short returnCode, string
message)
{
   CreateRoom();
}

public override void OnJoinedRoom()
{
   // joined a room successfully,
   // both JoinRandomRoom or CreateRoom lead here on success
}
```

Chained Filters

- Examples:
  - C0 BETWEEN 345 AND 475
  - C0 BETWEEN 345 AND 475;C0 BETWEEN 475 AND 575
  - C0 BETWEEN 345 AND 475;C0 BETWEEN 475 AND 575;C0 >= 575

Custom Room Listing

```
private TypedLobby sqlLobby = new TypedLobby("race", LobbyType.SqlLobby);

private void GetCustomRoomList(string sqlLobbyFilter)
{
  // sqlLobbyFilter = "C0 BETWEEN 345 AND 475"
  // sqlLobbyFilter = "C0 BETWEEN 345 AND 475;C0 BETWEEN 475 AND 575"
  // sqlLobbyFilter = "C0 BETWEEN 345 AND 475;C0 BETWEEN 475 AND 575;C0 >= 575"
  PhotonNetwork.GetCustomRoomList(sqlLobby, sqlLobbyFilter);
}

public override void OnRoomListUpdate(List<RoomInfo> roomList)
{
  // here you get the response, empty list if no rooms found
}
```

Skill-based Matchmaking

- 你可以使用SQL類型的大廳來實現基於技能的配對
- 首先，每個房間都有一個指定的技能，玩家應該有這個技能才能加入它
  - 這個值不應該改變，否則會使房間裡的玩家之前做的任何配對都無效
- 玩家應該通過 JoinRandomRoom 來加入房間。過濾器應該是基於玩家的技能。客戶端過濾出需要某些 “技能” 的房間
- JoinRandomRoom 如果沒有找到適合的房間，客戶端應該等待幾秒鐘，然後再試一次
- 客戶端可以做很多次或很少次的請求。也可以使用鍊式過濾器。但是最重要的是。客戶端必須隨著時間的推移開始放鬆過濾器的條件
- 放鬆過濾器條件是很重要的。房間裡有一個技能不太適合的玩家加入，總比沒人可以一起玩要好
- 也可以定義時間，超過這個時間，一直沒找到房間的話，這個客戶端必須用本身玩家的技能開一個新房間。然後等待其他人就加入
- 有時候，當房間很少的時候，這個工作流程可能需要一些時間。可以通過檢查 "application stats” 來拯救你的玩家
- 透過 application stats 可以知道有多少房間是可用的、有多少玩家在線上、有多少玩家沒有在房間內

Excluded SQL Keywords

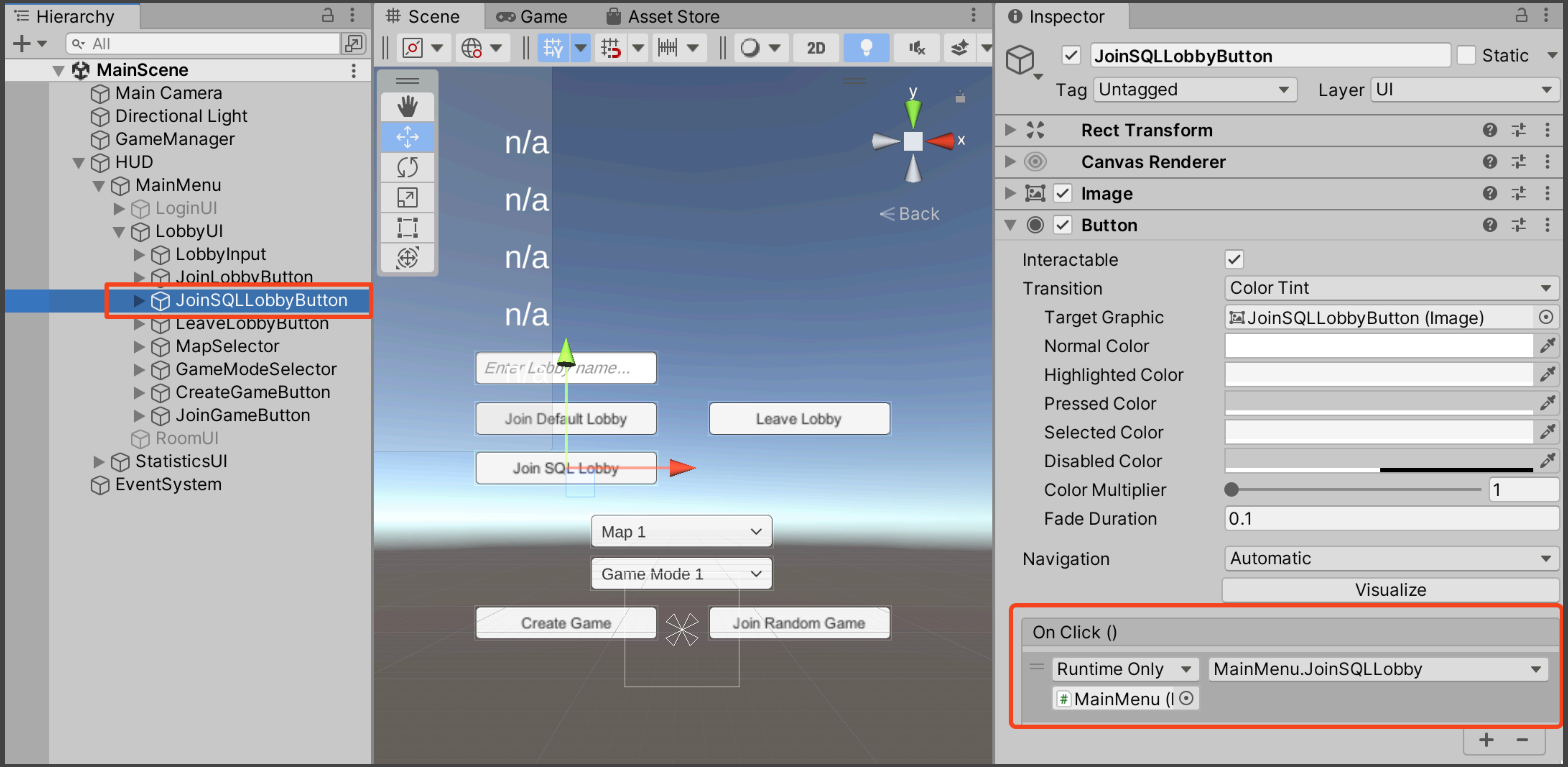
- ALTER
- CREATE
- DELETE
- DROP
- EXEC
- EXECUTE
- INSERT
- INSERT INTO
- MERGE
- SELECT
- UPDATE
- UNION
- UNION ALL

SQL Lobby Example

SQL Lobby Example

```
public class MainMenu : MonoBehaviourPunCallbacks
{
  …
  public void JoinLobby()
  {
    var typedLobby = new TypedLobby(m_lobbyInput.text, LobbyType.Default);
    PhotonNetwork.JoinLobby(typedLobby);
  }

  public void JoinSQLLobby()
  {
    var typedLobby = new TypedLobby(m_lobbyInput.text, LobbyType.SqlLobby);
    PhotonNetwork.JoinLobby(typedLobby);
  }
  …
}
```

SQL Lobby Example



JoinSQLLobbyButton   Static
Tag Untagged   Layer UI

Rect Transform
Canvas Renderer
Image
Button

Interactable ✓
Transition: Color Tint
Target Graphic: JoinSQLLobbyButton (Image)
Normal Color
Highlighted Color
Pressed Color
Selected Color
Disabled Color
Color Multiplier: 1
Fade Duration: 0.1
Navigation: Automatic
Visualize

On Click ()
Runtime Only   MainMenu.JoinSQLLobby
MainMenu (I

練習

SQL Lobby Example

```
public class StatisticsUI : MonoBehaviourPunCallbacks
{
  public static StatisticsUI instance;

  …
  private void Awake()
  {
    if (instance != null)
    {
      DestroyImmediate(gameObject);
      return;
    }

    instance = this;
  }

  …

  public void ClearRoomList()
  {
    cachedRoomList.Clear();
  }

  private void UpdateCachedRoomList(List<RoomInfo> roomList)
  {
    …
  }

  …
}
```

SQL Lobby Example

```
public class GameManager : MonoBehaviourPunCallbacks
{
  …
  private const string MAP_PROP_KEY = "C0";
  private const string GAME_MODE_PROP_KEY = "C1";
  private const string AI_PROP_KEY = "C2";

  string gameVersion = "1";

  …

  public void CreateGame(int map, int gameMode, TypedLobby type)
  {
    var roomOptions = new RoomOptions();
    roomOptions.CustomRoomPropertiesForLobby
      = new[] { MAP_PROP_KEY, GAME_MODE_PROP_KEY, AI_PROP_KEY };
    roomOptions.CustomRoomProperties
      = new ExitGames.Client.Photon.Hashtable
      {
        { MAP_PROP_KEY, map },
        { GAME_MODE_PROP_KEY, gameMode }
      };
    roomOptions.MaxPlayers = 4;

    PhotonNetwork.CreateRoom(null, roomOptions, type);
  }
```

SQL Lobby Example

```
  public void JoinRandomGame(int map, int gameMode, TypedLobby type, string sqlFilter)
  {
    byte expectedMaxPlayers = 0;
    ExitGames.Client.Photon.Hashtable expectedCustomRoomProperties = null;

    if (type.Type == LobbyType.Default)
    {
      expectedCustomRoomProperties = new ExitGames.Client.Photon.Hashtable
      {
        { MAP_PROP_KEY, map },
        { GAME_MODE_PROP_KEY, gameMode }
      };
    }

    PhotonNetwork.JoinRandomRoom(expectedCustomRoomProperties, expectedMaxPlayers,
MatchmakingMode.FillRoom, type, sqlFilter);
  }

  …
}
```

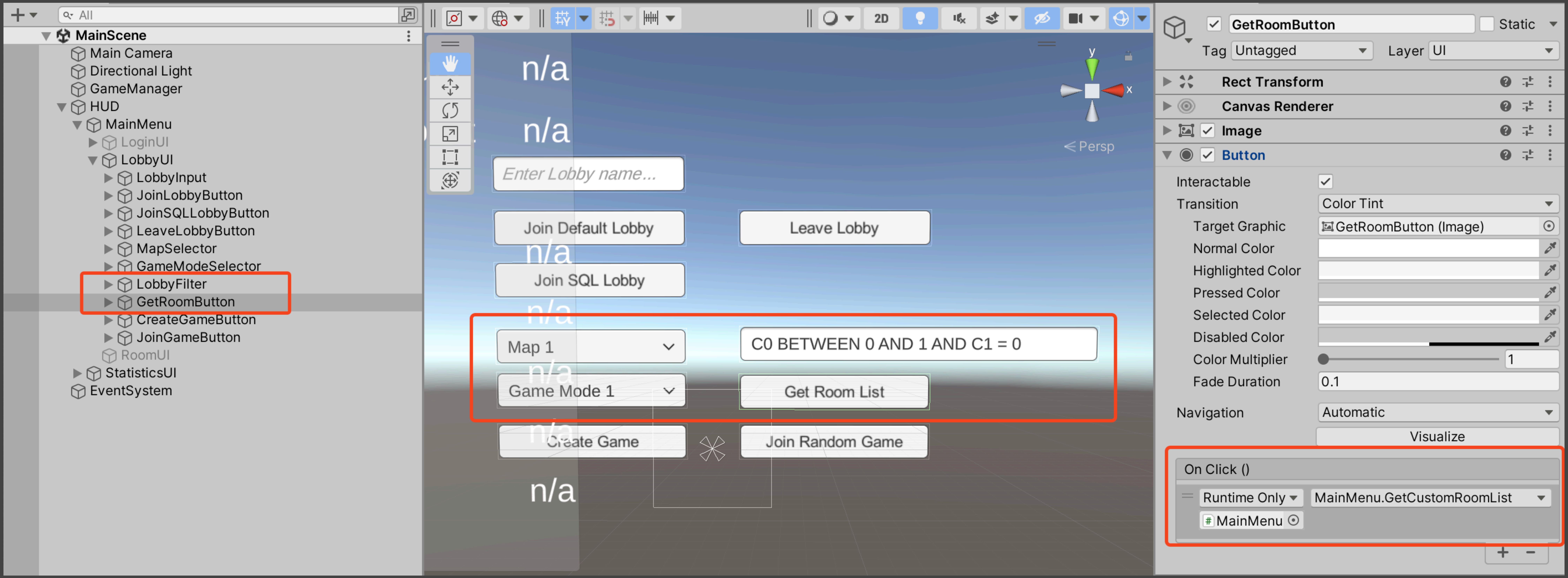
SQL Lobby Example

```
public class MainMenu : MonoBehaviourPunCallbacks
{
  …
  private TMP_InputField m_lobbyFilter;
  private Button m_createGameButton;
  private Button m_joinGameButton;

  private GameObject m_roomUI;

  void Awake()
  {
    …
    m_lobbyFilter = transform.FindAnyChild<TMP_InputField>("LobbyFilter");
    m_gameModeSelector = transform.FindAnyChild<TMP_Dropdown>("GameModeSelector");
    m_createGameButton = transform.FindAnyChild<Button>("CreateGameButton");
    m_joinGameButton = transform.FindAnyChild<Button>("JoinGameButton");

    …
  }

  …
  public void GetCustomRoomList()
  {
    var sqlLobby = new TypedLobby(m_lobbyInput.text, LobbyType.SqlLobby);
    var sqlLobbyFilter = m_lobbyFilter.text;

    StatisticsUI.instance.ClearRoomList();

    // C0 BETWEEN 0 AND 1 AND C1 = 0
    PhotonNetwork.GetCustomRoomList(sqlLobby, sqlLobbyFilter);
  }
```

SQL Lobby Example

```
  public void CreateGame()
  {
    var sqlLobby = new TypedLobby(m_lobbyInput.text, LobbyType.SqlLobby);
    GameManager.instance.CreateGame(m_mapSelector.value + 1, m_gameModeSelector.value + 1, sqlLobby);
  }

  public void JoinRandomGame()
  {
    var sqlLobby = new TypedLobby(m_lobbyInput.text, LobbyType.SqlLobby);
    var sqlLobbyFilter = m_lobbyFilter.text;
    GameManager.instance.JoinRandomGame(
      m_mapSelector.value + 1, m_gameModeSelector.value + 1, sqlLobby, sqlLobbyFilter);
  }

  public override void OnJoinRandomFailed(short returnCode, string message)
  {
    Debug.Log($"Join Random Failed: ({returnCode}) {message}");
  }
  …
}
```

SQL Lobby Example



MainScene
- Main Camera
- Directional Light
- GameManager
- HUD
  - MainMenu
    - LoginUI
    - LobbyUI
      - LobbyInput
      - JoinLobbyButton
      - JoinSQLLobbyButton
      - LeaveLobbyButton
      - MapSelector
      - GameModeSelector
      - LobbyFilter
      - GetRoomButton
      - CreateGameButton
      - JoinGameButton
    - RoomUI
  - StatisticsUI
- EventSystem

n/a
n/a
Enter Lobby name...
Join Default Lobby
Leave Lobby
Join SQL Lobby
Map 1
C0 BETWEEN 0 AND 1 AND C1 = 0
Game Mode 1
Get Room List
Create Game
Join Random Game
n/a

GetRoomButton
Static
Tag Untagged
Layer UI
Rect Transform
Canvas Renderer
Image
Button
Interactable ✓
Transition: Color Tint
Target Graphic: GetRoomButton (Image)
Normal Color
Highlighted Color
Pressed Color
Selected Color
Disabled Color
Color Multiplier: 1
Fade Duration: 0.1
Navigation: Automatic
Visualize
On Click ()
Runtime Only — MainMenu.GetCustomRoomList
MainMenu

練習

Asynchronous Random Lobby Type

- 這種大廳與 Default Lobby Type 類似，但有兩個主要區別：
  - 房間從遊戲服務器中刪除後，會在大廳列表中停留一小時（可用於匹配）。房間需要是可見的和開放的，以便在非同步配對中被考慮。
  - 房間列表不會被發送到客戶端，所以必須事先要知道房間名稱

- 這種類型的大廳應該與 webhooks 或任何其他持久化房間狀態的手段相結合，以便讓非同步(重新)加入完全發揮作用

Room Player List Example

Room Player List Example

```
public class GameManager : MonoBehaviourPunCallbacks
{
  …
  public override void OnJoinedRoom()
  {
    Debug.Log($"Joined room: {PhotonNetwork.CurrentRoom.Name} " +
          $"{PhotonNetwork.CurrentRoom.CustomProperties}");
  }

  public void EnterGame()
  {
    if (PhotonNetwork.IsMasterClient)
    {
      PhotonNetwork.LoadLevel("GameScene");
    }
  }
  …
}
```

Room Player List Example

```
public class MainMenu : MonoBehaviourPunCallbacks
{
  …
  private GameObject m_roomUI;
  private List<TMP_Text> m_playerNameTexts = new List<TMP_Text>();
  private Button m_enterGameButton;

  void Awake()
  {
    …
    m_roomUI = transform.FindAnyChild<Transform>("RoomUI").gameObject;
    m_playerNameTexts.Add(transform.FindAnyChild<TMP_Text>("PlayerName01"));
    m_playerNameTexts.Add(transform.FindAnyChild<TMP_Text>("PlayerName02"));
    m_playerNameTexts.Add(transform.FindAnyChild<TMP_Text>("PlayerName03"));
    m_playerNameTexts.Add(transform.FindAnyChild<TMP_Text>("PlayerName04"));
    m_enterGameButton = transform.FindAnyChild<Button>("EnterGameButton");

    ResetUI();
  }

  private void ResetUI()
  {
    …

    m_roomUI.SetActive(false);
    foreach (var mPlayerNameText in m_playerNameTexts)
    {
      mPlayerNameText.text = "n/a";
    }
    m_enterGameButton.interactable = true;
  }
```

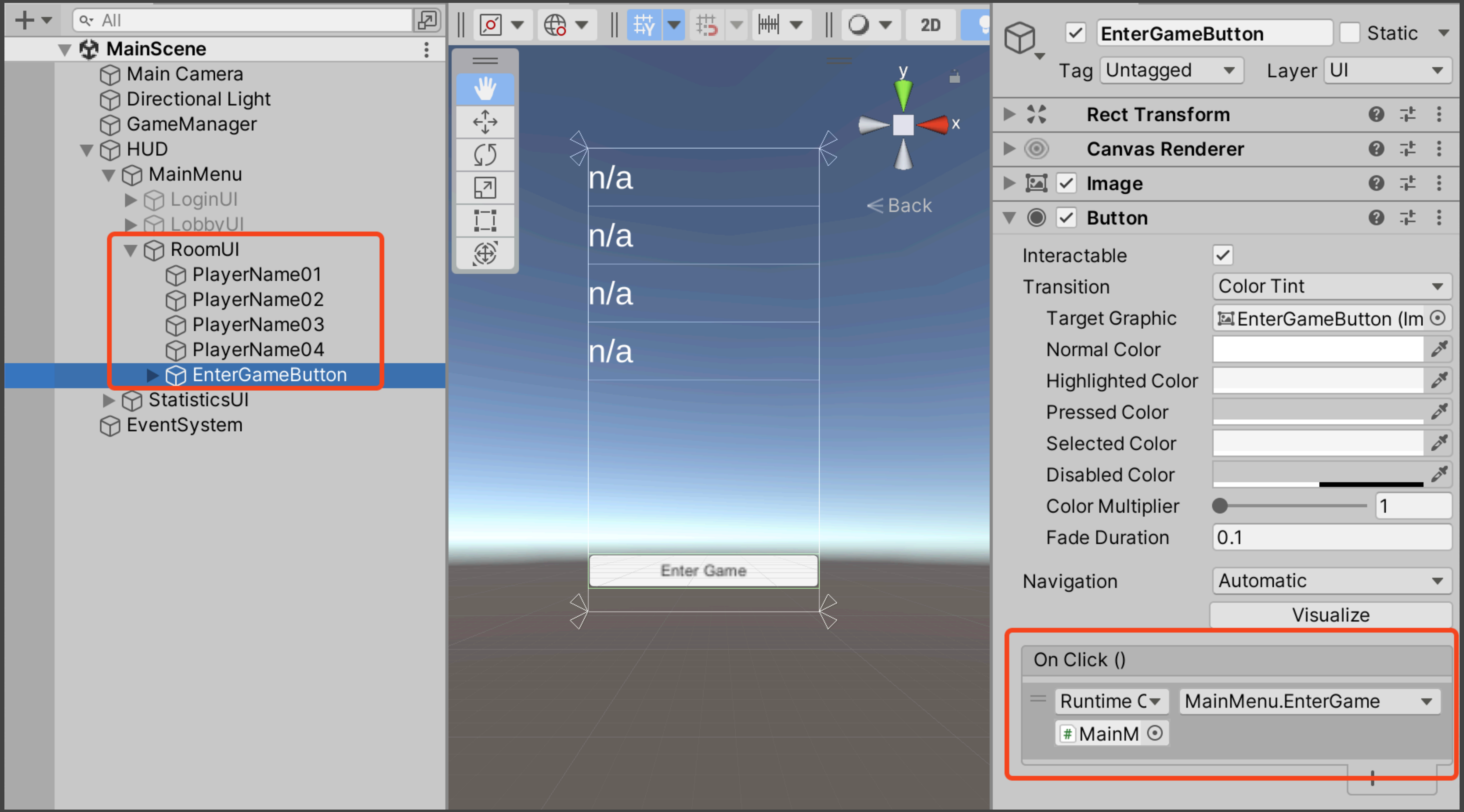
Room Player List Example

```
…
public override void OnJoinedRoom()
{
  m_lobbyUI.SetActive(false);
  m_roomUI.SetActive(true);

  // m_enterGameButton.gameObject.SetActive(PhotonNetwork.IsMasterClient);

  refreshPlayerList();
}

public override void OnPlayerEnteredRoom(Player newPlayer)
{
  refreshPlayerList();
}

public override void OnPlayerLeftRoom(Player newPlayer)
{
  refreshPlayerList();
}
```

Room Player List Example

```
  private void refreshPlayerList()
  {
    // 可以試試看，把這行搬到 OnJoinedRoom event 裏面，會有什麼現象
    m_enterGameButton.gameObject.SetActive(PhotonNetwork.IsMasterClient);

    var i = 0;
    for (i = 0; i < PhotonNetwork.PlayerList.Length; i++)
    {
      m_playerNameTexts[i].text = PhotonNetwork.PlayerList[i].NickName;
    }
    for (; i < 4; i++)
    {
      m_playerNameTexts[i].text = "n/a";
    }
  }

  public void EnterGame()
  {
    GameManager.instance.EnterGame();
  }

  …
}
```

Room Player List Example



EnterGameButton Static
Tag Untagged Layer UI
Rect Transform
Canvas Renderer
Image
Button
Interactable ✓
Transition Color Tint
Target Graphic EnterGameButton (Im
Normal Color
Highlighted Color
Pressed Color
Selected Color
Disabled Color
Color Multiplier 1
Fade Duration 0.1
Navigation Automatic
Visualize
On Click ()
Runtime C MainMenu.EnterGame
MainM

練習

Back Main Scene Example

Back Main Scene Example

```
public class GameManager : MonoBehaviourPunCallbacks
{
  …
  public void EnterGame()
  {
    if (PhotonNetwork.IsMasterClient)
    {
      PhotonNetwork.CurrentRoom.IsOpen = false;
      PhotonNetwork.LoadLevel("GameScene");
    }
  }

  …

  public void LeaveGame()
  {
    PhotonNetwork.LeaveRoom();
  }

  public override void OnLeftRoom()
  {
    PhotonNetwork.LoadLevel("MainScene");
  }
```

Back Main Scene Example

```
public class MainMenu : MonoBehaviourPunCallbacks
{
  public static MainMenu instance;

  private bool IsInitialed = false;

  …
  private GameObject m_gameUI;
  private Button m_leaveGameButton;

  void Awake()
  {
    …
    m_gameUI = transform.FindAnyChild<Transform>("GameUI").gameObject;
    m_leaveGameButton = transform.FindAnyChild<Button>("LeaveGameButton");

    ResetUI();
  }

  private void ResetUI()
  {
    …
    m_gameUI.SetActive(false);
    m_leaveGameButton.interactable = true;
  }
```

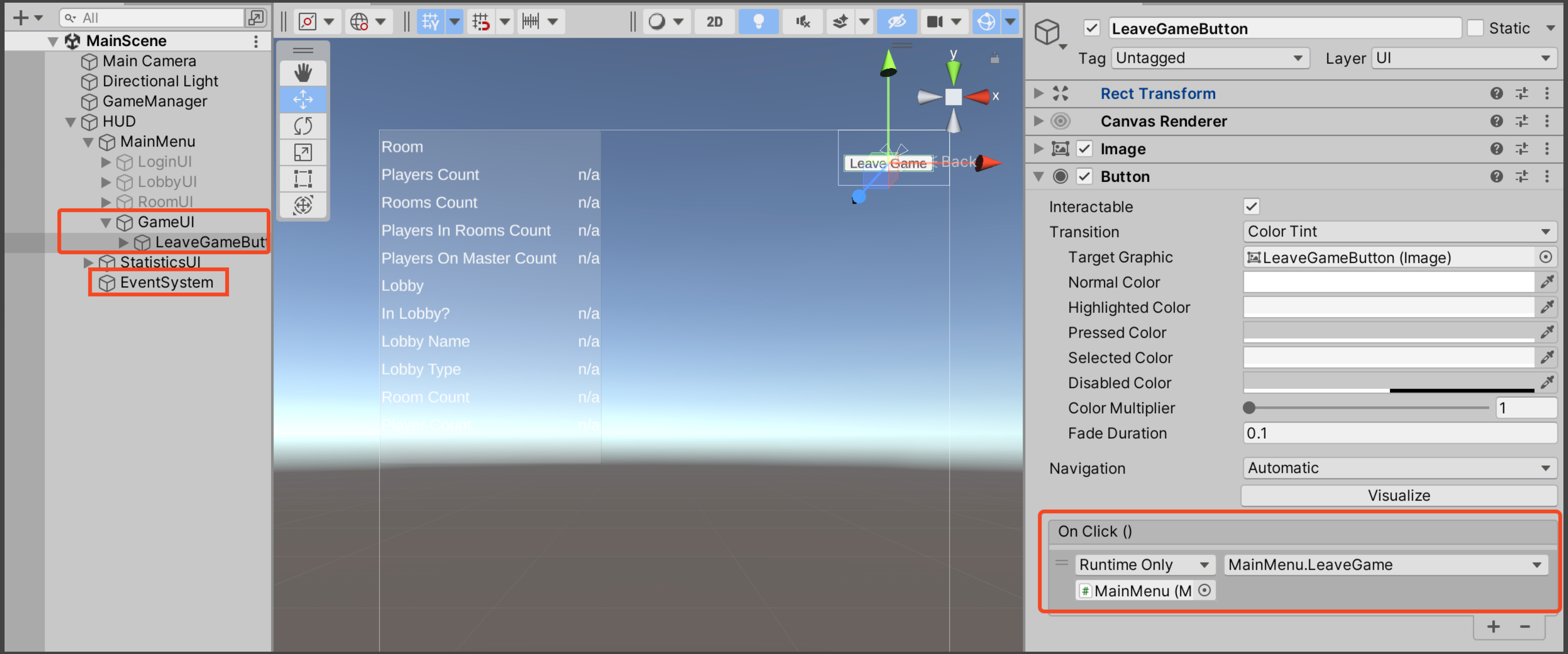
Back Main Scene Example

```
public void BackLobby()
{
  m_loginUI.SetActive(false);

  m_lobbyUI.SetActive(false);
  m_lobbyInput.interactable = true;
  m_joinLobbyButton.interactable = true;
  m_leaveLobbyButton.interactable = false;
  m_mapSelector.interactable = true;
  m_gameModeSelector.interactable = true;
  m_createGameButton.interactable = true;
  m_joinGameButton.interactable = true;

  m_roomUI.SetActive(false);
  m_gameUI.SetActive(false);
}

…
public void EnterGame()
{
  GameManager.instance.EnterGame();
}

public void LeaveGame()
{
  Debug.Log("LeaveGame");
  GameManager.instance.LeaveGame();
}
```

Back Main Scene Example

```
    private void OnSceneLoaded(Scene scene, LoadSceneMode mode)
    {
      Debug.Log($"Scene Loaded: {scene.name}");
      if (!PhotonNetwork.InRoom)
      {
        if (!IsInitialed)
        {
          IsInitialed = true;
          ResetUI();
        }
        else
        {
          BackLobby();
        }
      }
      else
      {
        m_lobbyUI.SetActive(false);
        m_roomUI.SetActive(false);
        m_gameUI.SetActive(true);
      }
    }
  }
```

Back Main Scene Example



```
Room
Players Count                n/a
Rooms Count                  n/a
Players In Rooms Count       n/a
Players On Master Count      n/a
Lobby
In Lobby?                    n/a
Lobby Name                   n/a
Lobby Type                   n/a
Room Count                   n/a
Player Count                 n/a
```

LeaveGameButton — Static
Tag: Untagged — Layer: UI

Rect Transform
Canvas Renderer
Image
Button

Interactable	✓
Transition	Color Tint
Target Graphic	LeaveGameButton (Image)
Normal Color	
Highlighted Color	
Pressed Color	
Selected Color	
Disabled Color	
Color Multiplier	1
Fade Duration	0.1
Navigation	Automatic

Visualize

On Click ()
Runtime Only — MainMenu.LeaveGame
MainMenu (M

練習