DemoAsteroids - Game - Astroid.cs

```
using UnityEngine;

using Random = UnityEngine.Random;
using Photon.Pun.UtilityScripts;

namespace Photon.Pun.Demo.Asteroids
{
    public class Asteroid : MonoBehaviour
    {
        public bool isLargeAsteroid;
        private bool isDestroyed;
        private PhotonView photonView;
        private new Rigidbody rigidbody;

        #region UNITY
        public void Awake()
        {
            photonView = GetComponent<PhotonView>();

            rigidbody = GetComponent<Rigidbody>();

            if (photonView.InstantiationData != null)
            {
                rigidbody.AddForce((Vector3) photonView.InstantiationData[0]);
                rigidbody.AddTorque((Vector3) photonView.InstantiationData[1]);

                isLargeAsteroid = (bool) photonView.InstantiationData[2];
            }
        }

        public void Update()
        {
            if (!photonView.IsMine)
            {
                return;
            }

            if (Mathf.Abs(transform.position.x) > Camera.main.orthographicSize * Camera.main.aspect || Mathf.Abs(transform.position.z) >
            Camera.main.orthographicSize)
            {
                // Out of the screen
                PhotonNetwork.Destroy(gameObject);
            }
        }

        public void OnCollisionEnter(Collision collision)
        {
            if (isDestroyed)
            {
                return;
            }

            if (collision.gameObject.CompareTag("Bullet"))
            {
                if (photonView.IsMine)
                {
                    Bullet bullet = collision.gameObject.GetComponent<Bullet>();
                    bullet.Owner.AddScore(isLargeAsteroid ? 2 : 1);

                    DestroyAsteroidGlobally();
                }
```

```
                else
                {
                    DestroyAsteroidLocally();
                }
            }
            else if (collision.gameObject.CompareTag("Player"))
            {
                if (photonView.IsMine)
                {
                    collision.gameObject.GetComponent<PhotonView>().RPC("DestroySpaceship", RpcTarget.All);
                    DestroyAsteroidGlobally();
                }
            }
        }
        #endregion

        private void DestroyAsteroidGlobally()
        {
            isDestroyed = true;

            if (isLargeAsteroid)
            {
                int numberToSpawn = Random.Range(3, 6);

                for (int counter = 0; counter < numberToSpawn; ++counter)
                {
                   Vector3 force = Quaternion.Euler(0, counter * 360.0f / numberToSpawn, 0)
                                * Vector3.forward * Random.Range(0.5f, 1.5f) * 300.0f;
                    Vector3 torque = Random.insideUnitSphere * Random.Range(500.0f, 1500.0f);
                    object[] instantiationData = {force, torque, false, PhotonNetwork.Time};

                        PhotonNetwork.InstantiateSceneObject("SmallAsteroid", transform.position + force.normalized * 10.0f, Quaternion.Euler(0,
                        Random.value * 180.0f, 0), 0, instantiationData);
                }
            }

            PhotonNetwork.Destroy(gameObject);
        }

        private void DestroyAsteroidLocally()
        {
            isDestroyed = true;
            GetComponent<Renderer>().enabled = false;
        }
    }
}
```

DemoAsteroids - Game - AstroidsGameManager.cs

```
using System.Collections;

using UnityEngine;
using UnityEngine.UI;

using Photon.Realtime;
using Photon.Pun.UtilityScripts;
using Hashtable = ExitGames.Client.Photon.Hashtable;

namespace Photon.Pun.Demo.Asteroids
{
    public class AsteroidsGameManager : MonoBehaviourPunCallbacks
    {
        public static AsteroidsGameManager Instance = null;
```

```
        public Text InfoText;
        public GameObject[] AsteroidPrefabs;

        #region UNITY
        public void Awake()
        {
            Instance = this;
        }

        public override void OnEnable()
        {
            base.OnEnable();
            CountdownTimer.OnCountdownTimerHasExpired += OnCountdownTimerIsExpired;
        }

        public void Start()
        {
            InfoText.text = "Waiting for other players...";

            Hashtable props = new Hashtable
            {
                {AsteroidsGame.PLAYER_LOADED_LEVEL, true}
            };
            PhotonNetwork.LocalPlayer.SetCustomProperties(props);
        }

        public override void OnDisable()
        {
            base.OnDisable();

            CountdownTimer.OnCountdownTimerHasExpired -= OnCountdownTimerIsExpired;
        }

        #endregion

        #region COROUTINES
        private IEnumerator SpawnAsteroid()
        {
            while (true)
            {
                yield return new WaitForSeconds(Random.Range(AsteroidsGame.ASTEROIDS_MIN_SPAWN_TIME,
AsteroidsGame.ASTEROIDS_MAX_SPAWN_TIME));

                Vector2 direction = Random.insideUnitCircle;
                Vector3 position = Vector3.zero;

                if (Mathf.Abs(direction.x) > Mathf.Abs(direction.y))
                {
                    // Make it appear on the left/right side
                    position = new Vector3(Mathf.Sign(direction.x) * Camera.main.orthographicSize * Camera.main.aspect, 0, direction.y *
Camera.main.orthographicSize);
                }
                else
                {
                    // Make it appear on the top/bottom
                    position = new Vector3(direction.x * Camera.main.orthographicSize * Camera.main.aspect, 0, Mathf.Sign(direction.y) *
Camera.main.orthographicSize);
                }

                // Offset slightly so we are not out of screen at creation time (as it would destroy the asteroid right away)
                position -= position.normalized * 0.1f;
```

```
                Vector3 force = -position.normalized * 1000.0f;
                Vector3 torque = Random.insideUnitSphere * Random.Range(500.0f, 1500.0f);
                object[] instantiationData = {force, torque, true};

                PhotonNetwork.InstantiateSceneObject("BigAsteroid", position,
                                Quaternion.Euler(Random.value * 360.0f, Random.value * 360.0f, Random.value * 360.0f), 0, instantiationData);
            }
        }

        private IEnumerator EndOfGame(string winner, int score)
        {
            float timer = 5.0f;

            while (timer > 0.0f)
            {
                InfoText.text = string.Format("Player {0} won with {1} points.\n\n\nReturning to login screen in {2} seconds.", winner, score,
timer.ToString("n2"));

                yield return new WaitForEndOfFrame();

                timer -= Time.deltaTime;
            }

            PhotonNetwork.LeaveRoom();
        }

        #endregion

        #region PUN CALLBACKS

        public override void OnDisconnected(DisconnectCause cause)
        {
            UnityEngine.SceneManagement.SceneManager.LoadScene("DemoAsteroids-LobbyScene");
        }

        public override void OnLeftRoom()
        {
            PhotonNetwork.Disconnect();
        }

        public override void OnMasterClientSwitched(Player newMasterClient)
        {
            if (PhotonNetwork.LocalPlayer.ActorNumber == newMasterClient.ActorNumber)
            {
                StartCoroutine(SpawnAsteroid());
            }
        }

        public override void OnPlayerLeftRoom(Player otherPlayer)
        {
            CheckEndOfGame();
        }

        public override void OnPlayerPropertiesUpdate(Player targetPlayer, Hashtable changedProps)
        {
            if (changedProps.ContainsKey(AsteroidsGame.PLAYER_LIVES))
            {
                CheckEndOfGame();
                return;
            }

            if (!PhotonNetwork.IsMasterClient)
            {
```

```
            return;
        }

        if (changedProps.ContainsKey(AsteroidsGame.PLAYER_LOADED_LEVEL))
        {
            if (CheckAllPlayerLoadedLevel())
            {
                Hashtable props = new Hashtable
                {
                    {CountdownTimer.CountdownStartTime, (float) PhotonNetwork.Time}
                };
                PhotonNetwork.CurrentRoom.SetCustomProperties(props);
            }
        }
    }

    #endregion

    private void StartGame()
    {
        float angularStart = (360.0f / PhotonNetwork.CurrentRoom.PlayerCount) * PhotonNetwork.LocalPlayer.GetPlayerNumber();
        float x = 20.0f * Mathf.Sin(angularStart * Mathf.Deg2Rad);
        float z = 20.0f * Mathf.Cos(angularStart * Mathf.Deg2Rad);
        Vector3 position = new Vector3(x, 0.0f, z);
        Quaternion rotation = Quaternion.Euler(0.0f, angularStart, 0.0f);

        PhotonNetwork.Instantiate("Spaceship", position, rotation, 0);

        if (PhotonNetwork.IsMasterClient)
        {
            StartCoroutine(SpawnAsteroid());
        }
    }

    private bool CheckAllPlayerLoadedLevel()
    {
        foreach (Player p in PhotonNetwork.PlayerList)
        {
            object playerLoadedLevel;

            if (p.CustomProperties.TryGetValue(AsteroidsGame.PLAYER_LOADED_LEVEL, out playerLoadedLevel))
            {
                if ((bool) playerLoadedLevel)
                {
                    continue;
                }
            }

            return false;
        }

        return true;
    }

    private void CheckEndOfGame()
    {
        bool allDestroyed = true;

        foreach (Player p in PhotonNetwork.PlayerList)
        {
            object lives;
            if (p.CustomProperties.TryGetValue(AsteroidsGame.PLAYER_LIVES, out lives))
            {
```

```
                    if ((int) lives > 0)
                    {
                        allDestroyed = false;
                        break;
                    }
                }
            }

            if (allDestroyed)
            {
                if (PhotonNetwork.IsMasterClient)
                {
                    StopAllCoroutines();
                }

                string winner = "";
                int score = -1;

                foreach (Player p in PhotonNetwork.PlayerList)
                {
                    if (p.GetScore() > score)
                    {
                        winner = p.NickName;
                        score = p.GetScore();
                    }
                }

                StartCoroutine(EndOfGame(winner, score));
            }
        }

        private void OnCountdownTimerIsExpired()
        {
            StartGame();
        }
    }
}
```

DemoAsteroids - Game - Bullet.cs

```
using Photon.Realtime;
using UnityEngine;

namespace Photon.Pun.Demo.Asteroids
{
    public class Bullet : MonoBehaviour
    {
        public Player Owner { get; private set; }

        public void Start()
        {
            Destroy(gameObject, 3.0f);
        }

        public void OnCollisionEnter(Collision collision)
        {
            Destroy(gameObject);
        }

        public void InitializeBullet(Player owner, Vector3 originalDirection, float lag)
        {
            Owner = owner;
```

```
            transform.forward = originalDirection;

            Rigidbody rigidbody = GetComponent<Rigidbody>();
            rigidbody.velocity = originalDirection * 200.0f;
            rigidbody.position += rigidbody.velocity * lag;
        }
    }
}
```

DemoAsteroids – Game – PlayerOverviewPanel .cs

```
using System.Collections.Generic;

using UnityEngine;
using UnityEngine.UI;

using ExitGames.Client.Photon;
using Photon.Realtime;
using Photon.Pun.UtilityScripts;

namespace Photon.Pun.Demo.Asteroids
{
    public class PlayerOverviewPanel : MonoBehaviourPunCallbacks
    {
        public GameObject PlayerOverviewEntryPrefab;

        private Dictionary<int, GameObject> playerListEntries;

        #region UNITY

        public void Awake()
        {
            playerListEntries = new Dictionary<int, GameObject>();

            foreach (Player p in PhotonNetwork.PlayerList)
            {
                GameObject entry = Instantiate(PlayerOverviewEntryPrefab);
                entry.transform.SetParent(gameObject.transform);
                entry.transform.localScale = Vector3.one;
                entry.GetComponent<Text>().color = AsteroidsGame.GetColor(p.GetPlayerNumber());
                entry.GetComponent<Text>().text = string.Format("{0}\nScore: {1}\nLives: {2}", p.NickName, p.GetScore(),
AsteroidsGame.PLAYER_MAX_LIVES);

                playerListEntries.Add(p.ActorNumber, entry);
            }
        }
        #endregion

        #region PUN CALLBACKS

        public override void OnPlayerLeftRoom(Player otherPlayer)
        {
            Destroy(playerListEntries[otherPlayer.ActorNumber].gameObject);
            playerListEntries.Remove(otherPlayer.ActorNumber);
        }

        public override void OnPlayerPropertiesUpdate(Player targetPlayer, Hashtable changedProps)
        {
            GameObject entry;
            if (playerListEntries.TryGetValue(targetPlayer.ActorNumber, out entry))
            {
                entry.GetComponent<Text>().text = string.Format("{0}\nScore: {1}\nLives: {2}", targetPlayer.NickName, targetPlayer.GetScore(),
targetPlayer.CustomProperties[AsteroidsGame.PLAYER_LIVES]);
```

```
            }
        }
        #endregion
    }
}
```

DemoAsteroids - Game - Spaceship.cs

```
using System.Collections;

using UnityEngine;

using Photon.Pun.UtilityScripts;
using Hashtable = ExitGames.Client.Photon.Hashtable;

namespace Photon.Pun.Demo.Asteroids
{
    public class Spaceship : MonoBehaviour
    {
        public float RotationSpeed = 90.0f;
        public float MovementSpeed = 2.0f;
        public float MaxSpeed = 0.2f;

        public ParticleSystem Destruction;
        public GameObject EngineTrail;
        public GameObject BulletPrefab;

        private PhotonView photonView;

        private new Rigidbody rigidbody;
        private new Collider collider;
        private new Renderer renderer;

        private float rotation = 0.0f;
        private float acceleration = 0.0f;
        private float shootingTimer = 0.0f;

        private bool controllable = true;

        #region UNITY
        public void Awake()
        {
            photonView = GetComponent<PhotonView>();

            rigidbody = GetComponent<Rigidbody>();
            collider = GetComponent<Collider>();
            renderer = GetComponent<Renderer>();
        }

        public void Start()
        {
            foreach (Renderer r in GetComponentsInChildren<Renderer>())
            {
                r.material.color = AsteroidsGame.GetColor(photonView.Owner.GetPlayerNumber());
            }
        }
        public void Update()
        {
            if (!photonView.IsMine || !controllable)
            {
                return;
            }
```

```
        rotation = Input.GetAxis("Horizontal");
        acceleration = Input.GetAxis("Vertical");

        if (Input.GetButton("Jump") && shootingTimer <= 0.0)
        {
            shootingTimer = 0.2f;

            photonView.RPC("Fire", RpcTarget.AllViaServer, rigidbody.position, rigidbody.rotation);
        }

        if (shootingTimer > 0.0f)
        {
            shootingTimer -= Time.deltaTime;
        }
    }

    public void FixedUpdate()
    {
        if (!photonView.IsMine)
        {
            return;
        }

        if (!controllable)
        {
            return;
        }

        Quaternion rot = rigidbody.rotation * Quaternion.Euler(0, rotation * RotationSpeed * Time.fixedDeltaTime, 0);
        rigidbody.MoveRotation(rot);

        Vector3 force = (rot * Vector3.forward) * acceleration * 1000.0f * MovementSpeed * Time.fixedDeltaTime;
        rigidbody.AddForce(force);

        if (rigidbody.velocity.magnitude > (MaxSpeed * 1000.0f))
        {
            rigidbody.velocity = rigidbody.velocity.normalized * MaxSpeed * 1000.0f;
        }

        CheckExitScreen();
    }
    #endregion

    #region COROUTINES
    private IEnumerator WaitForRespawn()
    {
        yield return new WaitForSeconds(AsteroidsGame.PLAYER_RESPAWN_TIME);

        photonView.RPC("RespawnSpaceship", RpcTarget.AllViaServer);
    }

    #endregion

    #region PUN CALLBACKS
    [PunRPC]
    public void DestroySpaceship()
    {
        rigidbody.velocity = Vector3.zero;
        rigidbody.angularVelocity = Vector3.zero;

        collider.enabled = false;
        renderer.enabled = false;
```

```
            controllable = false;

            EngineTrail.SetActive(false);
            Destruction.Play();

            if (photonView.IsMine)
            {
                object lives;
                if (PhotonNetwork.LocalPlayer.CustomProperties.TryGetValue(AsteroidsGame.PLAYER_LIVES, out lives))
                {
                    PhotonNetwork.LocalPlayer.SetCustomProperties(new Hashtable {{AsteroidsGame.PLAYER_LIVES, ((int) lives <= 1) ? 0 : ((int) lives -
1)}});

                    if (((int) lives) > 1)
                    {
                        StartCoroutine("WaitForRespawn");
                    }
                }
            }
        }

        [PunRPC]
        public void Fire(Vector3 position, Quaternion rotation, PhotonMessageInfo info)
        {
            float lag = (float) (PhotonNetwork.Time - info.SentServerTime);
            GameObject bullet;

            /** Use this if you want to fire one bullet at a time **/
            bullet = Instantiate(BulletPrefab, rigidbody.position, Quaternion.identity) as GameObject;
            bullet.GetComponent<Bullet>().InitializeBullet(photonView.Owner, (rotation * Vector3.forward), Mathf.Abs(lag));
        }

        [PunRPC]
        public void RespawnSpaceship()
        {
            collider.enabled = true;
            renderer.enabled = true;

            controllable = true;

            EngineTrail.SetActive(true);
            Destruction.Stop();
        }

        #endregion

        private void CheckExitScreen()
        {
            if (Camera.main == null)
            {
                return;
            }

            if (Mathf.Abs(rigidbody.position.x) > (Camera.main.orthographicSize * Camera.main.aspect))
            {
                rigidbody.position = new Vector3(-Mathf.Sign(rigidbody.position.x) * Camera.main.orthographicSize * Camera.main.aspect, 0,
rigidbody.position.z);
                rigidbody.position -= rigidbody.position.normalized * 0.1f; // offset a little bit to avoid looping back & forth between the 2 edges
            }

            if (Mathf.Abs(rigidbody.position.z) > Camera.main.orthographicSize)
            {
                rigidbody.position = new Vector3(rigidbody.position.x, rigidbody.position.y, -Mathf.Sign(rigidbody.position.z) *
```

```
Camera.main.orthographicSize);
                    rigidbody.position -= rigidbody.position.normalized * 0.1f; // offset a little bit to avoid looping back & forth between the 2 edges
                }
            }
        }
    }
```

DemoAsteroids – Lobby – LobbyMainPanel.cs

```
using ExitGames.Client.Photon;
using Photon.Realtime;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

namespace Photon.Pun.Demo.Asteroids
{
    public class LobbyMainPanel : MonoBehaviourPunCallbacks
    {
        [Header("Login Panel")]
        public GameObject LoginPanel;

        public InputField PlayerNameInput;

        [Header("Selection Panel")]
        public GameObject SelectionPanel;

        [Header("Create Room Panel")]
        public GameObject CreateRoomPanel;

        public InputField RoomNameInputField;
        public InputField MaxPlayersInputField;

        [Header("Join Random Room Panel")]
        public GameObject JoinRandomRoomPanel;

        [Header("Room List Panel")]
        public GameObject RoomListPanel;

        public GameObject RoomListContent;
        public GameObject RoomListEntryPrefab;

        [Header("Inside Room Panel")]
        public GameObject InsideRoomPanel;

        public Button StartGameButton;
        public GameObject PlayerListEntryPrefab;

        private Dictionary<string, RoomInfo> cachedRoomList;
        private Dictionary<string, GameObject> roomListEntries;
        private Dictionary<int, GameObject> playerListEntries;

        #region UNITY
        public void Awake()
        {
            PhotonNetwork.AutomaticallySyncScene = true;

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

            PlayerNameInput.text = "Player " + Random.Range(1000, 10000);
        }

        #endregion
```

```
#region PUN CALLBACKS

public override void OnConnectedToMaster()
{
    this.SetActivePanel(SelectionPanel.name);
}

public override void OnRoomListUpdate(List<RoomInfo> roomList)
{
    ClearRoomListView();

    UpdateCachedRoomList(roomList);
    UpdateRoomListView();
}

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

    ClearRoomListView();
}

public override void OnCreateRoomFailed(short returnCode, string message)
{
    SetActivePanel(SelectionPanel.name);
}

public override void OnJoinRoomFailed(short returnCode, string message)
{
    SetActivePanel(SelectionPanel.name);
}

public override void OnJoinRandomFailed(short returnCode, string message)
{
    string roomName = "Room " + Random.Range(1000, 10000);

    RoomOptions options = new RoomOptions {MaxPlayers = 8};

    PhotonNetwork.CreateRoom(roomName, options, null);
}

public override void OnJoinedRoom()
{
    SetActivePanel(InsideRoomPanel.name);

    if (playerListEntries == null)
    {
        playerListEntries = new Dictionary<int, GameObject>();
    }

    foreach (Player p in PhotonNetwork.PlayerList)
    {
        GameObject entry = Instantiate(PlayerListEntryPrefab);
        entry.transform.SetParent(InsideRoomPanel.transform);
        entry.transform.localScale = Vector3.one;
        entry.GetComponent<PlayerListEntry>().Initialize(p.ActorNumber, p.NickName);

        object isPlayerReady;
        if (p.CustomProperties.TryGetValue(AsteroidsGame.PLAYER_READY, out isPlayerReady))
        {
            entry.GetComponent<PlayerListEntry>().SetPlayerReady((bool) isPlayerReady);
        }
```

```
            playerListEntries.Add(p.ActorNumber, entry);
        }

        StartGameButton.gameObject.SetActive(CheckPlayersReady());

        Hashtable props = new Hashtable
        {
            {AsteroidsGame.PLAYER_LOADED_LEVEL, false}
        };
        PhotonNetwork.LocalPlayer.SetCustomProperties(props);
    }

    public override void OnLeftRoom()
    {
        SetActivePanel(SelectionPanel.name);

        foreach (GameObject entry in playerListEntries.Values)
        {
            Destroy(entry.gameObject);
        }

        playerListEntries.Clear();
        playerListEntries = null;
    }

    public override void OnPlayerEnteredRoom(Player newPlayer)
    {
        GameObject entry = Instantiate(PlayerListEntryPrefab);
        entry.transform.SetParent(InsideRoomPanel.transform);
        entry.transform.localScale = Vector3.one;
        entry.GetComponent<PlayerListEntry>().Initialize(newPlayer.ActorNumber, newPlayer.NickName);

        playerListEntries.Add(newPlayer.ActorNumber, entry);

        StartGameButton.gameObject.SetActive(CheckPlayersReady());
    }

    public override void OnPlayerLeftRoom(Player otherPlayer)
    {
        Destroy(playerListEntries[otherPlayer.ActorNumber].gameObject);
        playerListEntries.Remove(otherPlayer.ActorNumber);

        StartGameButton.gameObject.SetActive(CheckPlayersReady());
    }

    public override void OnMasterClientSwitched(Player newMasterClient)
    {
        if (PhotonNetwork.LocalPlayer.ActorNumber == newMasterClient.ActorNumber)
        {
            StartGameButton.gameObject.SetActive(CheckPlayersReady());
        }
    }

    public override void OnPlayerPropertiesUpdate(Player targetPlayer, Hashtable changedProps)
    {
        if (playerListEntries == null)
        {
            playerListEntries = new Dictionary<int, GameObject>();
        }

        GameObject entry;
        if (playerListEntries.TryGetValue(targetPlayer.ActorNumber, out entry))
```

```
        {
            object isPlayerReady;
            if (changedProps.TryGetValue(AsteroidsGame.PLAYER_READY, out isPlayerReady))
            {
                entry.GetComponent<PlayerListEntry>().SetPlayerReady((bool) isPlayerReady);
            }
        }

        StartGameButton.gameObject.SetActive(CheckPlayersReady());
    }

    #endregion
    #region UI CALLBACKS
    public void OnBackButtonClicked()
    {
        if (PhotonNetwork.InLobby)
        {
            PhotonNetwork.LeaveLobby();
        }

        SetActivePanel(SelectionPanel.name);
    }

    public void OnCreateRoomButtonClicked()
    {
        string roomName = RoomNameInputField.text;
        roomName = (roomName.Equals(string.Empty)) ? "Room " + Random.Range(1000, 10000) : roomName;

        byte maxPlayers;
        byte.TryParse(MaxPlayersInputField.text, out maxPlayers);
        maxPlayers = (byte) Mathf.Clamp(maxPlayers, 2, 8);

        RoomOptions options = new RoomOptions {MaxPlayers = maxPlayers};

        PhotonNetwork.CreateRoom(roomName, options, null);
    }

    public void OnJoinRandomRoomButtonClicked()
    {
        SetActivePanel(JoinRandomRoomPanel.name);

        PhotonNetwork.JoinRandomRoom();
    }

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

    public void OnLoginButtonClicked()
    {
        string playerName = PlayerNameInput.text;

        if (!playerName.Equals(""))
        {
            PhotonNetwork.LocalPlayer.NickName = playerName;
            PhotonNetwork.ConnectUsingSettings();
        }
        else
        {
            Debug.LogError("Player Name is invalid.");
        }
    }
```

```
public void OnRoomListButtonClicked()
{
    if (!PhotonNetwork.InLobby)
    {
        PhotonNetwork.JoinLobby();
    }

    SetActivePanel(RoomListPanel.name);
}

public void OnStartGameButtonClicked()
{
    PhotonNetwork.CurrentRoom.IsOpen = false;
    PhotonNetwork.CurrentRoom.IsVisible = false;

    PhotonNetwork.LoadLevel("DemoAsteroids-GameScene");
}
#endregion

private bool CheckPlayersReady()
{
    if (!PhotonNetwork.IsMasterClient)
    {
        return false;
    }

    foreach (Player p in PhotonNetwork.PlayerList)
    {
        object isPlayerReady;
        if (p.CustomProperties.TryGetValue(AsteroidsGame.PLAYER_READY, out isPlayerReady))
        {
            if (!(bool) isPlayerReady)
            {
                return false;
            }
        }
        else
        {
            return false;
        }
    }

    return true;
}

private void ClearRoomListView()
{
    foreach (GameObject entry in roomListEntries.Values)
    {
        Destroy(entry.gameObject);
    }

    roomListEntries.Clear();
}

public void LocalPlayerPropertiesUpdated()
{
    StartGameButton.gameObject.SetActive(CheckPlayersReady());
}

private void SetActivePanel(string activePanel)
{
```

```
            LoginPanel.SetActive(activePanel.Equals(LoginPanel.name));
            SelectionPanel.SetActive(activePanel.Equals(SelectionPanel.name));
            CreateRoomPanel.SetActive(activePanel.Equals(CreateRoomPanel.name));
            JoinRandomRoomPanel.SetActive(activePanel.Equals(JoinRandomRoomPanel.name));
            RoomListPanel.SetActive(activePanel.Equals(RoomListPanel.name));       // UI should call OnRoomListButtonClicked() to activate this
            InsideRoomPanel.SetActive(activePanel.Equals(InsideRoomPanel.name));
        }

        private void UpdateCachedRoomList(List<RoomInfo> roomList)
        {
            foreach (RoomInfo info in roomList)
            {
                // Remove room from cached room list if it got closed, became invisible or was marked as removed
                if (!info.IsOpen || !info.IsVisible || info.RemovedFromList)
                {
                    if (cachedRoomList.ContainsKey(info.Name))
                    {
                        cachedRoomList.Remove(info.Name);
                    }

                    continue;
                }

                // Update cached room info
                if (cachedRoomList.ContainsKey(info.Name))
                {
                    cachedRoomList[info.Name] = info;
                }
                // Add new room info to cache
                else
                {
                    cachedRoomList.Add(info.Name, info);
                }
            }
        }

        private void UpdateRoomListView()
        {
            foreach (RoomInfo info in cachedRoomList.Values)
            {
                GameObject entry = Instantiate(RoomListEntryPrefab);
                entry.transform.SetParent(RoomListContent.transform);
                entry.transform.localScale = Vector3.one;
                entry.GetComponent<RoomListEntry>().Initialize(info.Name, (byte)info.PlayerCount, info.MaxPlayers);

                roomListEntries.Add(info.Name, entry);
            }
        }
    }
}
```

DemoAsteroids – Lobby – LobbyTopPanel.cs

```
using UnityEngine;
using UnityEngine.UI;

namespace Photon.Pun.Demo.Asteroids
{
    public class LobbyTopPanel : MonoBehaviour
    {
        private readonly string connectionStatusMessage = "    Connection Status: ";

        [Header("UI References")]
        public Text ConnectionStatusText;
```

```
        #region UNITY

        public void Update()
        {
            ConnectionStatusText.text = connectionStatusMessage + PhotonNetwork.NetworkClientState;
        }

        #endregion
    }
}
```

DemoAsteroids - Lobby - PlayerListEntry.cs

```
using UnityEngine;
using UnityEngine.UI;

using ExitGames.Client.Photon;
using Photon.Realtime;
using Photon.Pun.UtilityScripts;

namespace Photon.Pun.Demo.Asteroids
{
    public class PlayerListEntry : MonoBehaviour
    {
        [Header("UI References")]
        public Text PlayerNameText;

        public Image PlayerColorImage;
        public Button PlayerReadyButton;
        public Image PlayerReadyImage;

        private int ownerId;
        private bool isPlayerReady;

        #region UNITY

        public void OnEnable()
        {
            PlayerNumbering.OnPlayerNumberingChanged += OnPlayerNumberingChanged;
        }

        public void Start()
        {
            if (PhotonNetwork.LocalPlayer.ActorNumber != ownerId)
            {
                PlayerReadyButton.gameObject.SetActive(false);
            }
            else
            {
                Hashtable initialProps = new Hashtable() {{AsteroidsGame.PLAYER_READY, isPlayerReady}, {AsteroidsGame.PLAYER_LIVES,
AsteroidsGame.PLAYER_MAX_LIVES}};
                PhotonNetwork.LocalPlayer.SetCustomProperties(initialProps);
                PhotonNetwork.LocalPlayer.SetScore(0);

                PlayerReadyButton.onClick.AddListener(() =>
                {
                    isPlayerReady = !isPlayerReady;
                    SetPlayerReady(isPlayerReady);

                    Hashtable props = new Hashtable() {{AsteroidsGame.PLAYER_READY, isPlayerReady}};
                    PhotonNetwork.LocalPlayer.SetCustomProperties(props);

                    if (PhotonNetwork.IsMasterClient)
```

```
                {
                    FindObjectOfType<LobbyMainPanel>().LocalPlayerPropertiesUpdated();
                }
            });
        }
    }

    public void OnDisable()
    {
        PlayerNumbering.OnPlayerNumberingChanged -= OnPlayerNumberingChanged;
    }

    #endregion

    public void Initialize(int playerId, string playerName)
    {
        ownerId = playerId;
        PlayerNameText.text = playerName;
    }

    private void OnPlayerNumberingChanged()
    {
        foreach (Player p in PhotonNetwork.PlayerList)
        {
            if (p.ActorNumber == ownerId)
            {
                PlayerColorImage.color = AsteroidsGame.GetColor(p.GetPlayerNumber());
            }
        }
    }

    public void SetPlayerReady(bool playerReady)
    {
        PlayerReadyButton.GetComponentInChildren<Text>().text = playerReady ? "Ready!" : "Ready?";
        PlayerReadyImage.enabled = playerReady;
    }
}
}
```

DemoAsteroids - Lobby - RoomListEntry.cs

```
using UnityEngine;
using UnityEngine.UI;

namespace Photon.Pun.Demo.Asteroids
{
    public class RoomListEntry : MonoBehaviour
    {
        public Text RoomNameText;
        public Text RoomPlayersText;
        public Button JoinRoomButton;

        private string roomName;

        public void Start()
        {
            JoinRoomButton.onClick.AddListener(() =>
            {
                if (PhotonNetwork.InLobby)
                {
                    PhotonNetwork.LeaveLobby();
                }

                PhotonNetwork.JoinRoom(roomName);
```

```
            });
        }

        public void Initialize(string name, byte currentPlayers, byte maxPlayers)
        {
            roomName = name;

            RoomNameText.text = name;
            RoomPlayersText.text = currentPlayers + " / " + maxPlayers;
        }
    }
}
```

DemoAsteroids - AsteroidGame.cs

```
using UnityEngine;
namespace Photon.Pun.Demo.Asteroids
{
    public class AsteroidsGame
    {
        public const float ASTEROIDS_MIN_SPAWN_TIME = 5.0f;
        public const float ASTEROIDS_MAX_SPAWN_TIME = 10.0f;
        public const float PLAYER_RESPAWN_TIME = 4.0f;
        public const int PLAYER_MAX_LIVES = 3;
        public const string PLAYER_LIVES = "PlayerLives";
        public const string PLAYER_READY = "IsPlayerReady";
        public const string PLAYER_LOADED_LEVEL = "PlayerLoadedLevel";
        public static Color GetColor(int colorChoice)
        {
            switch (colorChoice)
            {
                case 0: return Color.red;
                case 1: return Color.green;
                case 2: return Color.blue;
                case 3: return Color.yellow;
                case 4: return Color.cyan;
                case 5: return Color.grey;
                case 6: return Color.magenta;
                case 7: return Color.white;
            }
            return Color.black;
        }
    }
}
```