🎓 LESSON 05 Ball Creation

Start

LESSON GOALS

Goal Summary:

- Create interactive, networked balls
- Implement pickup/drop mechanics
- Add realistic throw physics
- Sync ball movement through network impulses
- Add respawn/despawn logic
- Build BallController to manage multiple balls

Video Link

Next

Learning Objectives + Deliverables

Learning Objectives

- Build an interactable GameBall prefab
- Use UdonSynced variables (ownerPlayerId, impulseVector, isAlive)
- Detect input for pickup/throw
- Implement respawn/despawn behavior
- Build BallController to coordinate ball logic across the game

Deliverables

- ✓ A ball that can be picked up in VR & Desktop
- ✓ A ball that can be thrown with correct physics
- ✓ Movement synchronized across clients
- ✓ Respawn logic that only Master can execute
- ✓ BallController that spawns balls at game start

GENERAL DESCRIPTION OF THE LESSON

High-Level Description

🎯 **Goal:** We will spawn a number of balls that is half of the players that are in the game when it starts. They will be able to pick and throw the balls.

🔘 **LEVEL 1: Advanced Challenge**

In this lesson we will:

- Import a soccer ball and attach Rigidbody component
- Create a GameBall script controlling:
- Use a UdonSynced variable to apply an impulse when ball is released by the player.
- Add respawn logic to the balls to appear close to the center of the field when the game changes to the state GAME
- Build BallController:
- In **GameController** script type this code before starting the lesson:

🧠AI Lesson Prompt: Ball Creation

Next

- Ownership: Claim the ownership when a player picks up the ball
- Pickup & Release: Keep pressed the trigger to hold the ball.
- Throw impulse: Release the trigger to throw the ball.
- Update local position of the ball while held: Updated by the position of the player (VR and Desktop)

- Keeps track of all balls
- Assigns respawn positions
- Detects who is carrying a ball
- Spawns correct number of balls for players

```
public bool IsVR  {
    get { return _isVR; }
    set { _isVR = value; }
  }

  void Start()  {
    _isVR = Networking.LocalPlayer.IsUserInVR();
    ...

 public bool CheckFireTriggeredDown()  {
   if (_isVR)  {
     // Primary hand trigger check (left or right)
     return Input.GetAxisRaw("Oculus_CrossPlatform_PrimaryIndexTrigger") > 0.75f
       || Input.GetAxisRaw("Oculus_CrossPlatform_SecondaryIndexTrigger") > 0.75f;
   }
   else  {
     return Input.GetButtonDown("Fire1");
   }
 }

 public bool CheckFireTriggeredUp() {
    if (_isVR)  {
      // Primary hand trigger check (left or right)
      return Input.GetAxisRaw("Oculus_CrossPlatform_PrimaryIndexTrigger") < 0.1f
        && Input.GetAxisRaw("Oculus_CrossPlatform_SecondaryIndexTrigger") < 0.1f;
    }
    else  {
      return Input.GetButtonUp("Fire1");
    }
 }
```

Exercise 1: Create the GameBall object + components.

- Actions:

  1) Import a soccer ball from the Unity asset store

  2) Create an empty container in the scene named “Balls”

  3) Add the soccer ball there and create a new prefab (/Game/Resources/GameBall)

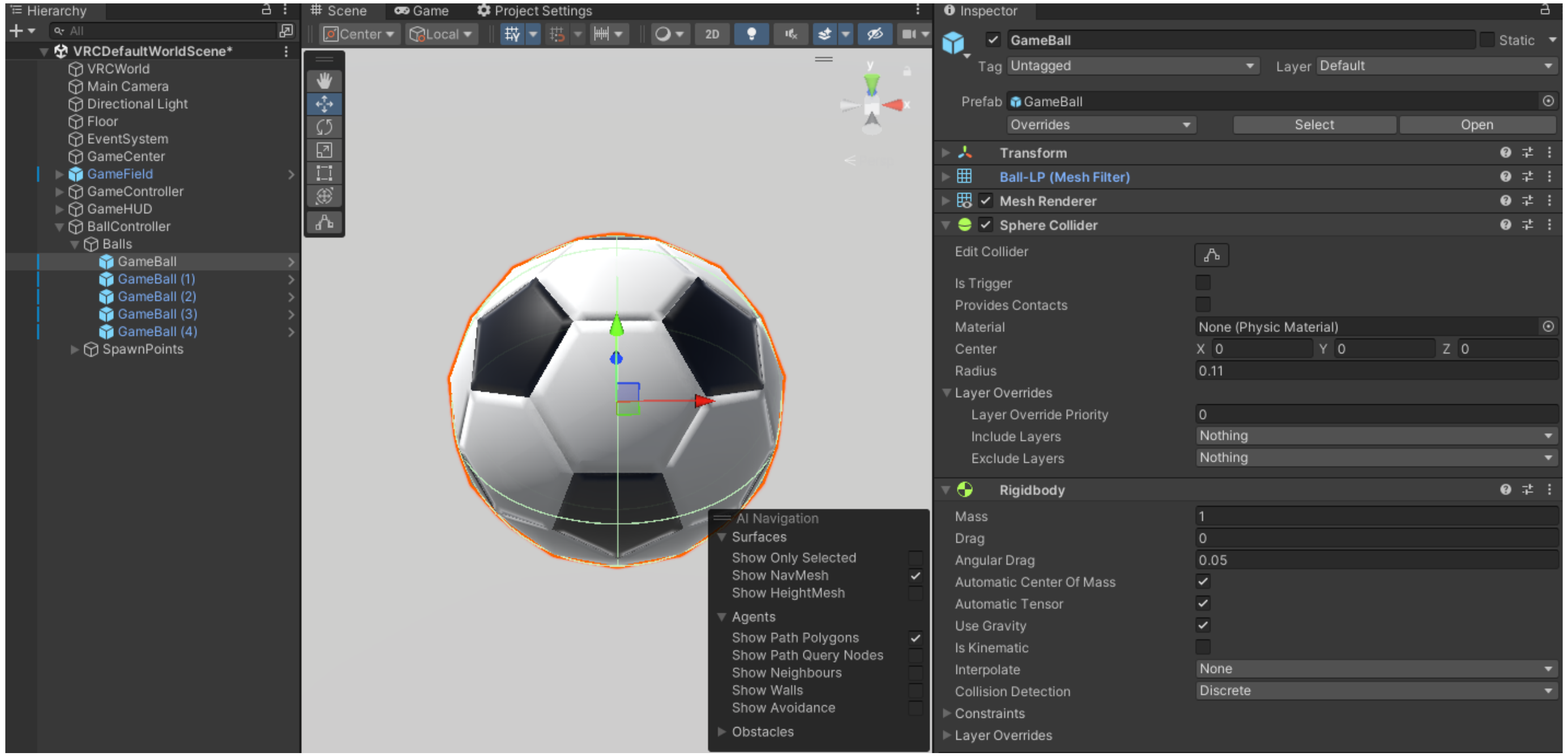
  4) Add a Rigidbody

  5) Set Sphere’s Collider to isTrigger = false

  6) Add UdonBehaviour with empty script (class GameBall)

GameObject

Script



```
VRCDefaultWorldScene*
  VRCWorld
  Main Camera
  Directional Light
  Floor
  EventSystem
  GameCenter
  GameField
  GameController
  GameHUD
  BallController
    Balls
      GameBall
      GameBall (1)
      GameBall (2)
      GameBall (3)
      GameBall (4)
    SpawnPoints
```

GameBall — Static
Tag: Untagged — Layer: Default
Prefab: GameBall
Overrides — Select — Open

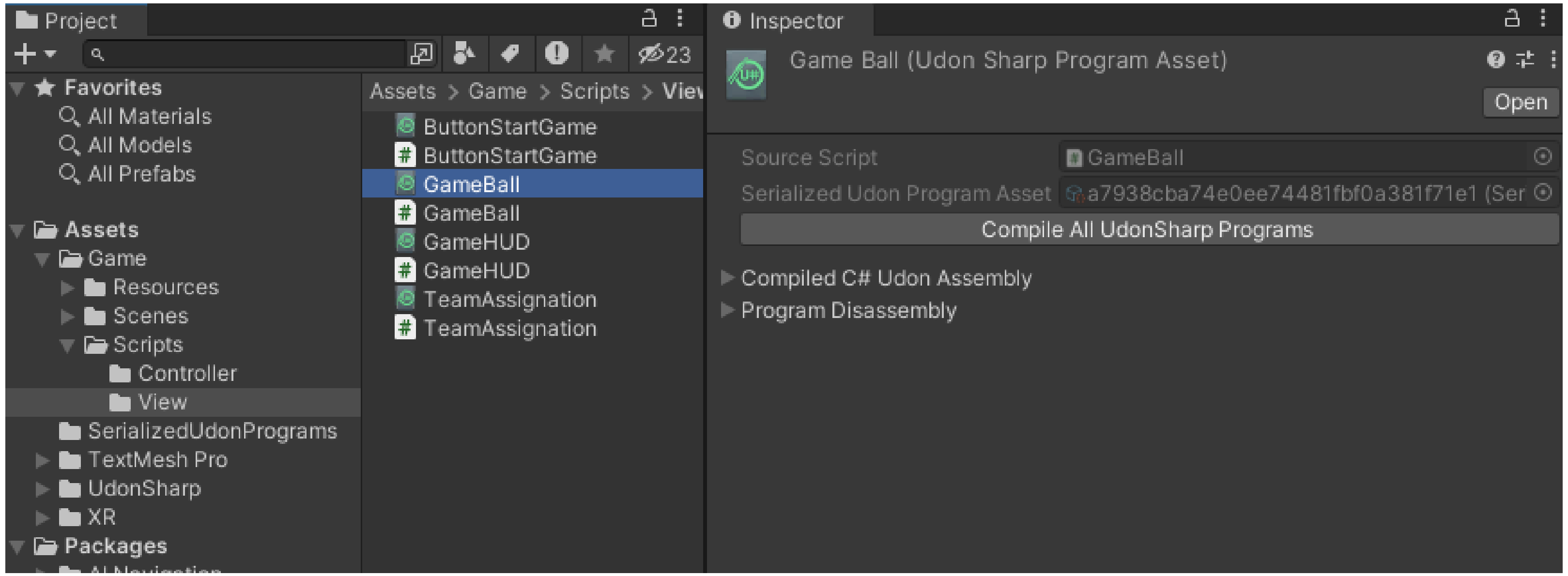
Transform
Ball-LP (Mesh Filter)
Mesh Renderer ✓

Sphere Collider ✓

Edit Collider	
Is Trigger	☐
Provides Contacts	☐
Material	None (Physic Material)
Center	X 0 Y 0 Z 0
Radius	0.11
Layer Overrides	
Layer Override Priority	0
Include Layers	Nothing
Exclude Layers	Nothing

Rigidbody

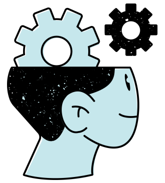
Mass	1
Drag	0
Angular Drag	0.05
Automatic Center Of Mass	✓
Automatic Tensor	✓
Use Gravity	✓
Is Kinematic	☐
Interpolate	None
Collision Detection	Discrete
Constraints	
Layer Overrides	



Game Ball (Udon Sharp Program Asset)

Source Script: GameBall

Serialized Udon Program Asset: a7938cba74e0ee74481fbf0a381f71e1 (Ser



Exercise 2: Implement **GameBall** behaviour to pick and release the ball between all the different players in game.

- Actions in **GameBall** script (Part 1/2):

1) Create the next member variables:

2) Implement methods:

void ActivatePhysics(bool isActive) // Will activate/deactivate the physics of the ball

bool IsOwned() // Will check if the ball is already owned by one player

void ClearOwner() // Will set (ownerPlayerId=-1), activate ball physics and RequestSerialitzation to update the state in the network.

override void OnDeserialization() // Checks if who is the owner by checking the network variable (ownerPlayerId). If there is an owner deactivate physics, if there is no owner activate physics.

override void Interact() // Before grabbing the ball check if it's already assigned to an existing player, if not then proceed to update (ownerPlayerId) and RequestSerialization().

override void OnPlayerLeft(VRCPlayerApi player) // if the ball was owned by the leaving player then clear the ownership.

void Update() // Checks if there is an owner for the ball and update the position of the ball considering if we are on VR or on desktop. On desktop mode click once to pick, click again to release, but in VR keep pressed trigger to grab and release trigger to release. Beware that in VR you should use _ignoreTime to avoid picking up again the ball after releasing. This method is more difficult than usual, so feel free to ask AI for help.

```
public void ClearOwner()
{
    ownerPlayerId = -1;
    _ignoreTime = 0;
    _currentOwner = null;
    ActivatePhysics(true);
    RequestSerialization();
    OnDeserialization();
}
```

```
public void ActivatePhysics(bool isActive)
{
    _body.useGravity = isActive;
    _body.isKinematic = !isActive;
}
```

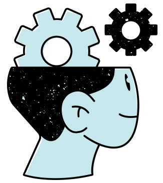
```
public override void OnDeserialization()
{
   if (ownerPlayerId == -1)
   {
      _currentOwner = null;
      ActivatePhysics(true);
   }
   else
   {
     _currentOwner = VRCPlayerApi.GetPlayerById(ownerPlayerId);
     ActivatePhysics(false);
   }
}
```

```
public override void OnPlayerLeft(VRCPlayerApi player)
{
  if (player == _currentOwner)
  {
     ClearOwner();
     RequestSerialization();
  }
}
```

```
public override void Interact()
{
  var player = Networking.LocalPlayer;
  if (!IsOwned() && (_ignoreTime <= 0))
  {
     if (!Networking.IsOwner(gameObject))
            Networking.SetOwner(player, gameObject);

     _ignoreTime = 1;
     ownerPlayerId = player.playerId;
     RequestSerialization();
     OnDeserialization();
     return;
   }
}
```

```
[RequireComponent(typeof(Rigidbody))]
public class GameBall : UdonSharpBehaviour
{
    [Header("Desktop/Editor Placement (Head View)")]
    public Vector3 headPositionOffset = new Vector3(0.3f, -0.2f, 0.6f);
    public Vector3 headRotationOffset = Vector3.zero;

    [Header("VR Placement (Right Hand Grip)")]
    public Vector3 handPositionOffset = new Vector3(0.08f, -0.05f, 0.15f);
    public Vector3 handRotationOffset = Vector3.zero;

    [UdonSynced]
    public int ownerPlayerId = -1; // Id player who owns the ball

    [SerializeField] private GameController gameController;

    private VRCPlayerApi _currentOwner;
    private Rigidbody _body;
    private float _ignoreTime = 0;
private void Update() {
  if (_currentOwner ≠ null) {
   if (_currentOwner == Networking.LocalPlayer) {
      if (_ignoreTime > 0) {
            _ignoreTime -= Time.deltaTime;
      }
      if (_ignoreTime ⩽ 0) {
         if (gameController.IsVR) {
            if (gameController.CheckFireTriggeredUp()) {
                 ClearOwner();
            }
         }
         else {
           if (gameController.CheckFireTriggeredDown()) {
              ClearOwner();
           }
         }
      }
   }
   if (_currentOwner ≠ null) {
      if (gameController.IsVR) {
          VRCPlayerApi.TrackingData handData = _currentOwner.GetTrackingData(VRCPlayerApi.TrackingDataType.RightHand);
          transform.position = handData.position + handData.rotation * handPositionOffset;
      }
      else {
          VRCPlayerApi.TrackingData headData = _currentOwner.GetTrackingData(VRCPlayerApi.TrackingDataType.Head);
          transform.position = headData.position + headData.rotation * headPositionOffset;
      }
   }
  }
  else {
     if (_ignoreTime > 0) {
        _ignoreTime -= Time.deltaTime;
     }
  }
}
```

```
public bool IsOwned()
{
  return ownerPlayerId ≠ -1;
}
```



Exercise 2: Implement **GameBall** behaviour to pick and and release the ball between all the different players in game.

- Actions in **GameBall** script (Part 2/2):

3) Test in Unity Editor pick up/release

4) Test with multiple VRChat instances that all the players can pick up/release the ball

5) When releasing the ball, instead of doing (**ClearOwner()**) implement a method (**ThrowBall()**) that will throw the ball with an impulse. Create this network variable and the method:

6) You will apply the impulseVector on the (**OnDeserialization()**) when you are releasing the ball.

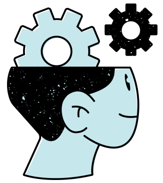
7) Test it in the Unity Editor

8) Test it with multiple VRChat instances

Code Checkpoint: Single Ball

```
public const float TotalForceBallImpulse = 30;

[UdonSynced] public Vector3 impulseVector;

public void ThrowBall()
{
    if (gameController.IsVR)
    {
       VRCPlayerApi.TrackingData handData = _currentOwner.GetTrackingData(VRCPlayerApi.TrackingDataType.RightHand);
       impulseVector = (handData.rotation * Quaternion.Euler(new Vector3(0, 55.3f, 0))) * Vector3.forward;
    }
    else
    {
       VRCPlayerApi.TrackingData headData = _currentOwner.GetTrackingData(VRCPlayerApi.TrackingDataType.Head);
       impulseVector = headData.rotation * Vector3.forward;
    }
    impulseVector.Normalize();
    ClearOwner();
    _ignoreTime = 1;
}
```

```
public override void OnDeserialization()
 {
     if (ownerPlayerId == -1)
     {
         _currentOwner = null;
         ActivatePhysics(true);
         if (impulseVector != Vector3.zero)
         {
            if (_body.velocity.magnitude == 0)
            {
               _body.AddForce(impulseVector * TotalForceBallImpulse, ForceMode.Impulse);
            }
         }
     }
     else
     {
         _currentOwner = VRCPlayerApi.GetPlayerById(ownerPlayerId);
         ActivatePhysics(false);
     }
 }
```



Exercise 3: The ball should only appear when the game changes to the GAME state. We should respawn the ball when the game starts at the center of the field.

- Actions (Part 1/2):

1) In (**GameBall**) script, create these variable members:

2) Implement the methods:

RespawnBall(Vector3 position) // Respawn a ball setting it alive

void DespawnBall() // Despawn a ball setting it not alive

void Start() // Set a ball not alive, deactivate physics and set the position to the _resetPosition

3) In (**GameController**) create a member variable and initialize it:

4) Still in (**GameController**), we are going to call (**RespawnBall(Vector3 position)**) method with the position on the center of the field when we change to the GAME state.

5) Test in the Unity Editor

6) Test it in multiple VRChat instance and observe the ball doesn't appear to all the clients at the transition to GAME state.

```
public void DespawnBall()
{
    if (isAlive)
    {
       if (Networking.IsOwner(gameObject))
       {
           isAlive = false;
           ownerPlayerId = -1;
           this.transform.position = _resetPosition;
           RequestSerialization();
           OnDeserialization();
        }
    }
}
```

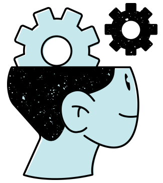
```
public void RespawnBall(Vector3 position)
{
    if (Networking.IsMaster)
    {
       if (!Networking.IsOwner(gameObject))
              Networking.SetOwner(Networking.LocalPlayer, gameObject);

       isAlive = true;
       ownerPlayerId = -1;
       transform.position = position;
       ActivatePhysics(true);
       RequestSerialization();
       OnDeserialization();
    }
}
```

```
private void StateChanged()
{
    switch (_currentState)
    {

       case GameState.GAME:
         // OTHER CODE...

         gameBall.RespawnBall(centerField.transform.position);
         break;
```

```
[UdonSynced] public bool isAlive; // Reports if the ball is being used

private Vector3 _resetPosition; // An out of the field position when the ball is not alive
```

```
[SerializeField] private GameBall gameBall;
```

```
private void Start()
{
    isAlive = false;
    _body = GetComponent<Rigidbody>();
    _resetPosition = new Vector3(1000, 1000, 1000);
    this.transform.position = _resetPosition;
    ActivatePhysics(false);
}
```



Exercise 3: The ball should only appear when the game changes to the GAME state. We should respawn the ball when the game starts at the center of the field.

- Actions (Part 2/2):

7) Still in (**GameBall**), to fix the previous we need to do a patch in the system. We are going to create variables that will keep the position of the collision of the ball with the floor:

8) The calculation of this collision (**OnCollisionEnter(Collision collision)**) should only be done once just after the ball has been respawned, create a boolean variable that allow you to control that.

9) This initialPosition will be used in the **Update()** of the GameBall in order to set the ball position when there is no owner.

10) Reset the (initialPosition) to zero when the player (**void Interact()**) with it.

11) If you succeed the ball will appear in the non-master clients at the collision point with the floor.

12) Test it with multiple VRChat instances

Code Checkpoint: Ball Respawn

```
void Update()
{
    if (isAlive)
    {
        if (_currentOwner ≠ null)
        {
            // OTHER CODE ...
        }
        else
        {
            if (_ignoreTime > 0)
            {
                _ignoreTime -= Time.deltaTime;
            }
            if (initialPosition ≠ Vector3.zero)
            {
                transform.position = initialPosition;
            }
        }
    }
    else
    {
        transform.position = _resetPosition;
    }
```

```
public void OnCollisionEnter(Collision collision)
{
   if (_hasBeenRespawned)
   {
      if (collision.gameObject == gameController.Floor)
      {
         _hasBeenRespawned = false;
         initialPosition = this.transform.position;
         RequestSerialization();
       }
    }
}
```

```
public override void Interact()
{
    var player = Networking.LocalPlayer;
    if (!IsOwned() && (_ignoreTime <= 0))
    {
        if (!Networking.IsOwner(gameObject))
            Networking.SetOwner(player, gameObject);
        _ignoreTime = 1;
        ownerPlayerId = player.playerId;
        initialPosition = Vector3.zero;
        RequestSerialization();
        OnDeserialization();
        return;
    }
}
```

```
[UdonSynced] public Vector3 initialPosition; // First floor collision

private bool _hasBeenRespawned; // Set to true when the ball has been respawned


void Start()
{
   isAlive = false;
   _body = GetComponent<Rigidbody>();
   _resetPosition = new Vector3(1000, 1000, 1000);
   this.transform.position = _resetPosition;
   ActivatePhysics(false);
   _hasBeenRespawned = true;  // Initialize to true
}
```

Exercise 4: Since we are going to work with multiple balls (up to 5) we need to create a Udon Script **BallController** that will handle the balls respawning and despawning.

- Actions (Part 1/2):

1) Create the script (BallController) in **/Game/Scripts/Controller** folder.

2) Create an empty GameObject in the scene, name it BallController and add the script.

3) We are going to define these members:

4) We are going to create 5 balls in the scene and we are going to create a collection of invisible gameObjects that will define the respawn position that the ball can appear. Initialize the previous variable member with all these GameObjects.

5) We are going to implement these functions:

int GetTotalFreeBalls() // Total balls free (meaning !isAlive)

GameBall GetNextFreeBall() // Next free ball (meaning one with !isAlive)

bool LocalPlayerCarryingABall() // If the local player owns a ball

Vector3 GetRandomRespawnPosition() // Get a random respawn position

void RespawnBall() // Respawn the next free ball

GameBall GetBallOwnedByLocalPlayer() // Get the ball owned by local player

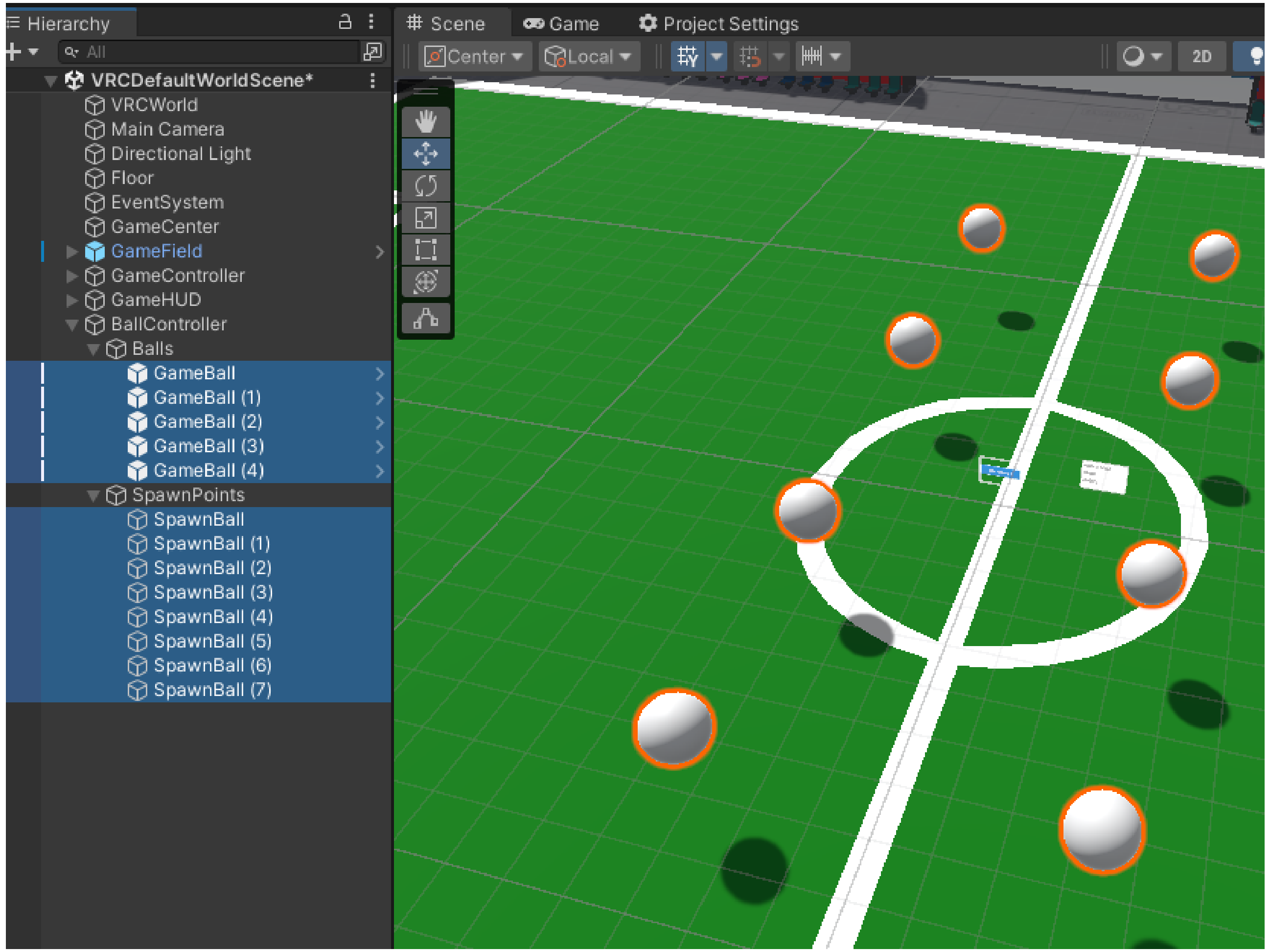
```
[SerializeField] private GameController gameController;

[SerializeField] private GameBall[] balls; // References to the GameBalls

[SerializeField] private Transform[] spawnPoints; // Ball respawn points
```

```
public int GetTotalFreeBalls()
{
    int counter = 0;
    foreach (GameBall ball in balls)
    {
        if (!ball.isAlive)
        {
            counter++;
        }
    }
    return counter;
}
```

```
public void RespawnBall()
{
    if (!Networking.IsMaster) return;
    if (GetTotalFreeBalls() == 0) return;
    GameBall ball = GetNextFreeBall();

    ball.RespawnBall(GetRandomRespawnPosition());
    ball.ClearOwner();
    RequestSerialization();
}
```

```
public GameBall GetBallOwnedByLocalPlayer()
{
    int idLocalPlayer = Networking.LocalPlayer.playerId;
    foreach (GameBall ball in balls)
    {
      if (ball.ownerPlayerId == idLocalPlayer)
      {
         return ball;
      }
    }
    return null;
}
```

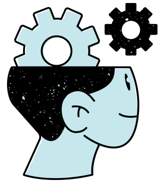
```
public GameBall GetNextFreeBall()
{
    foreach (GameBall ball in balls)
    {
        if (!ball.isAlive)
        {
            return ball;
        }
    }
    return null;
}
```



```
VRCDefaultWorldScene*
  VRCWorld
  Main Camera
  Directional Light
  Floor
  EventSystem
  GameCenter
  GameField
  GameController
  GameHUD
  BallController
    Balls
      GameBall
      GameBall (1)
      GameBall (2)
      GameBall (3)
      GameBall (4)
    SpawnPoints
      SpawnBall
      SpawnBall (1)
      SpawnBall (2)
      SpawnBall (3)
      SpawnBall (4)
      SpawnBall (5)
      SpawnBall (6)
      SpawnBall (7)
```

```
public bool LocalPlayerCarryingABall()
{
   int idLocalPlayer = Networking.LocalPlayer.playerId;
   foreach (GameBall ball in balls)
   {
      if (ball.isAlive && ball.ownerPlayerId == idLocalPlayer)
      {
           return true;
      }
    }
    return false;
}
```

```
private Vector3 GetRandomRespawnPosition()
{
   int index = Random.Range(0, spawnPoints.Length);
   return spawnPoints[index].position;
}
```



Exercise 4: Since we are going to work with multiple balls (up to 5) we need to create a Udon Script **BallController** that will handle the balls respawning and despawning.

- Actions (Part 2/2):

6) Back in (**GameController**), remove the reference in the (**private GameBall gameBall**) it was a temporal solution to test the respawning functionality. Now we are going to have a reference (**private BallController ballController**)

7) Still in (**GameController**) when changing to the GAME state, we are going to respawn a number of balls that is half the total number of players.

8) Test it in the Unity Editor. Different executions should spawn the ball in different positions.

9) Test with multiple VRChat instances.

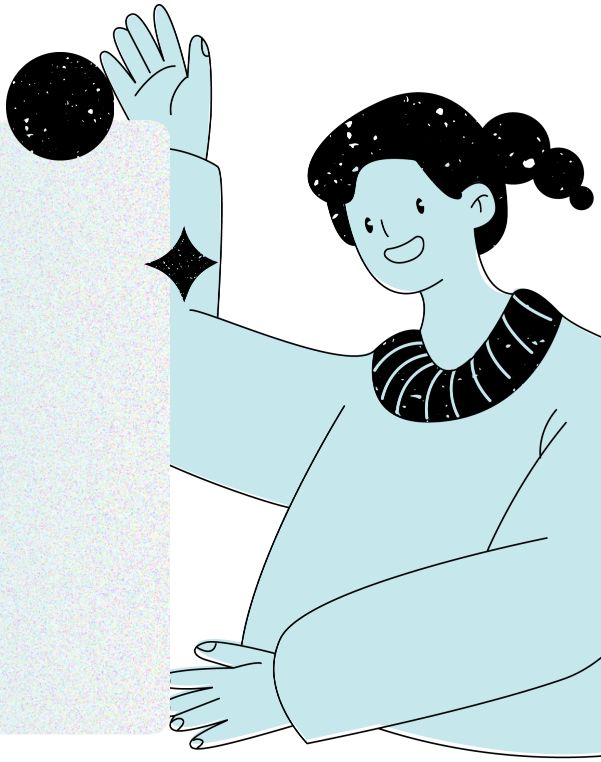
Code Checkpoint: Ball Controller

```
private void StateChanged()
{
        switch (_currentState)
        {
            case GameState.GAME:
                 // OTHER CODE  ...

             #if UNITY_EDITOR || TEST_SINGLEPLAYER_VR
                 _totalPlayersInGame = 2;
             #endif

                 int playersForTeam = (int)(_totalPlayersInGame / 2);
                 for (int i = 0; i < playersForTeam; i++)
                 {
                     ballController.RespawnBall();
                 }
                 break;
```

LESSON 05 COMPLETED

You now have:

- Interactive, throwable balls
- VR/Desktop input support
- Real physics behavior
- Network-synchronized impulses
- Respawn logic
- A full multi-ball manager



Self-Evaluation

It's time to put what we've learned into practice! Here are 4 questions to check by yourself what you have learnt in this lesson.

Question 1

Question 2

Question 3

Question 4

Why does the ball need an “owner” when a player picks it up?

- To ensure that only one player controls the ball’s position and actions
- To improve physics performance
- To change the color of the ball based on the player

Why is physics disabled (set to kinematic) while a player is holding the ball?

To reduce network synchronization

To prevent the ball from being visible

To prevent physics forces from interfering while the ball follows the player

How is the throwing action synchronized across all players?

By syncing the throw impulse so all clients apply the same physics force	By making every player calculate the throw locally	By continuously syncing the ball’s position every frame

Why is ball respawn logic usually restricted to the Master player?

To avoid multiple players respawning the same ball at the same time	Because non-Master players cannot use physics	Because only the Master can see the ball

Help us to improve

Did you understand how ownership transfer works in VRChat?

Write your answer here.

Send

Was the throwing logic (forward vector, physics) clear?

Write your answer here.

Send

Which part was hardest: Pickup, Throw, Physics, or Networking?

Write your answer here.

Send

What example or visual aid would help you understand ball physics better?

Write your answer here.

Send