Unity Assignment

Objective:

Create a **simple hyper-casual game** where the **screen is divided into two halves**:

- **Right Side** → The player controls their character.
- **Left Side** → The ghost player with a **real-time synced version** of the player's actions, simulating network syncing **locally**.

This assignment tests **real-time state synchronization**, **shader & particle effects**, and **performance optimization** without an actual multiplayer server.

Assignment Title: "Sync Dash"

Concept:

A **glowing cube** moves forward automatically. The **player taps to jump** and avoid obstacles while collecting glowing orbs. The **left side mirrors the player's movements in real-time**, simulating a networked opponent.

Requirements:

1. Core Gameplay (Strong Logic)

1. The **player-controlled cube** moves forward automatically on the **right side**.
2. **Tap to jump** and avoid obstacles.
3. **Collect glowing orbs** for points.
4. The **left side of the screen should mirror the player's movements in real-time**.
    a. **The player must send data to the left-side character in the same way it would be sent to a networked multiplayer opponent.**
5. **Game speed increases** over time.
   Implement a **score system** (points for distance and collectibles).

---

2. Real-Time State Syncing (Simulating Multiplayer Locally)

✅ The **left side** should mimic the **right side's player actions in real time** (jump, movement, orb collection, obstacle collision).

✅ The left side should **introduce a slight lag (optional, configurable delay)** to make it feel like a real network sync.
✅ Ensure **smoothing interpolation** so that the left side's movement isn't jittery.
✅ Use **local data structures (e.g., ring buffer or queue)** to sync player actions in real time.

---

3. UI & Game Flow

✅ A **main menu** with "Start" and "Exit" buttons.
✅ A **game over screen** with "Restart" and "Main Menu" buttons.
✅ Display the **current score** at the top.
✅ A **subtle motion blur or shader effect** as the speed increases.

---

4. Performance Optimization

✅ **Use Object Pooling** for obstacles and collectibles.
✅ Optimize the **syncing mechanism** to minimize frame drops.
✅ Keep the build size **under 50MB**.

Bonus Tasks (Optional but Encouraged)

Shaders & Visual Effects (Mandatory)

✅ The **player cube should have a glowing shader**.
✅ Obstacles should **use a dissolve shader** when hit.
✅ Collecting an orb should **trigger a particle burst effect**.
✅ Upon crashing, show a **screen distortion effect (chromatic aberration, screen shake, ripple effect)**.

Submission Guidelines

1. Provide a **GitHub repository link** with a **README** explaining the game concept and mechanics.
2. Include a **short gameplay video or GIF** demonstrating the mechanics.
3. Ensure the project is compatible with **Unity 2021 or later**.
4. Share a **build (.apk)**

Evaluation Criteria

- **Logic & State Syncing** – Is the real-time sync smooth and accurate?
- **Shader & Particle Effects** – Are the effects visually engaging? **- BONUS**
- **Optimization** – Does the game run smoothly on mobile?
- **Creativity** – Does the left side feel like a real multiplayer opponent?
- **Polish & UX** – Is the UI clean and intuitive?
