Game Design Document: Shonen Showdown (v0.2)

Version: 0.2 (Alpha Architecture)
Engine: Unity 6 (URP)
Network: Photon Fusion 2 (Shared Mode) + Photon Voice

---

1. Executive Summary

Shonen Showdown is a multiplayer First-Person Trading Card Game (TCG) that bridges the gap between physical card games and anime battles. Unlike 2D simulators, this project simulates the immersive experience of sitting in a "Share House" (anime apartment), holding physical cards, and witnessing monsters spawn as 3D holograms on the coffee table.

- **Primary Platform:** PC (Keyboard/Mouse) $\to$ Transition to VR.
- **Core Loop:** Collect cards, build decks, and compete in social, voice-enabled lobbies.
- **Community Integration:** Exclusive lobbies and cosmetics for YouTube Channel Members.

---

2. Team & Responsibilities

Strict adherence to these roles prevents merge conflicts.

Member	Role	Branch Prefix	Responsibilities
Georgi	Lead Developer	feature/	**Owner of:** GameScene.unity (The Battle logic). • Photon Fusion Networking implementation. • Core Mechanics (Turns, Phases, Stacks). • VR Interaction

			handling.
Ricardo	Art Lead	art/	**Owner of:** MainMenu.unity. • 3D Character Models & Animations. • Environment Design (The "Share House"). • UI/UX Design (Card layouts, HUD). **• Card Art Import (Art/CardIllustrations).**
Sam	Architect (Data/Audio)	data/	**Owner of:** Scriptable Objects (Cards/ and Keywords/). **• Managing Keyword Assets (e.g., "Piercing").** • Game Balance (Stats, Effects). • Sound Design (SFX/BGM selection). • Quality Assurance (QA) & Playtesting.

3. Project Architecture (The "Code Bible")

3.1 Folder Structure

We adhere to the "Underscore Rule". All team-generated assets reside in _Shonen to separate them from third-party plugins.

Plaintext

```
Assets/
├── _Shonen/              <-- TEAM WORKSPACE
│   ├── Documents/        <-- Store this GDD & Playtest Logs here
│   ├── _Scenes/          <-- Only Georgi & Ricardo edit these
│   ├── Art/              <-- Ricardo's Domain
│   │   ├── Characters/     <-- 3D FBX Models
│   │   ├── CardIllustrations/ <-- 2D Sprites for Card Faces (NEW)
│   │   ├── Environment/
│   │   ├── UI/
│   │   └── Materials/
│   ├── Audio/            <-- Sam's Domain (SFX, BGM)
│   ├── Data/             <-- Sam's Domain
│   │   ├── Cards/        <-- Card Data Scriptable Objects
│   │   ├── Keywords/      <-- Keyword Assets (NEW)
│   │   └── GameSettings/
│   ├── Prefabs/          <-- Shared Assets
│   │   ├── Monsters/      <-- 3D Gameplay Models (Linked to Cards)
│   │   ├── Cards/        <-- Hand/Deck Templates
│   │   └── Effects/      <-- VFX/Particles
│   └── Scripts/          <-- Georgi's Domain (C# Code)
│
├── Plugins/              <-- Third-party tools (DO NOT TOUCH)
├── Photon/               <-- Networking Engine
└── TextMesh Pro/         <-- UI Text Engine
```

3.2 Network Logic (Photon Fusion)

- **Topology:** Shared Mode.
- **Why:** Allows physics interactions (throwing cards) to feel responsive without complex server-side prediction.
- **State Authority:** The Master Client (Host) controls the Turn Logic.

- **Voice:** Proximity Chat enabled. Players hear opponents louder if they lean in (VR). Global "Shout" button available.

3.3 Asset Standards

- **3D Models:**
  - **Scale:** 1 Unity Unit = 1 Meter.
  - **Pivot Point:** Must be at the FEET (Bottom Center, Y=0).
  - **Poly Count:** Optimized for VR (Low poly + Baked Normal Maps).
- **Audio:**
  - **No Copyright Rips:** Do not use direct audio from anime (Naruto/DBZ). Use royalty-free "sound-alikes."
- **Textures:**
  - **Max Resolution:** 2048x2048 (Bosses), 1024x1024 (Standard).

---

4. Gameplay Mechanics

4.1 Turn Structure

The game follows a strict phase system enforced by the TurnManager script.

1. **Draw Phase:** Active player draws 1 card.
2. **Standby Phase:** Upkeep effects (e.g., "Take 500 damage per turn").
3. **Main Phase 1:** Normal Summon (1 per turn), Set/Activate Spells & Traps.
4. **Battle Phase:** Declare attacks, Damage Calculation.
5. **Main Phase 2:** Final setups before ending turn.
6. **End Phase:** Pass turn to the next player (Clockwise).

4.2 Summoning Rules

- **Level 1-4:** Normal Summon (No cost).
- **Level 5-6:** Tribute Summon (Send 1 monster from field to Graveyard).
- **Level 7+ (Bosses):** Tribute Summon (Send 2 monsters).
- **Visual Effect:** Triggers 3D Model instantiation. The card remains on the table, but a 3D avatar stands above it.

4.3 The "Stack" (Chain System)

- **Definition:** "Stacking a card on top of another for added effect."
- **Logic:** LIFO (Last-In, First-Out).
- **Example:** Player A (Attack) $\to$ Player B (Trap) $\to$ Player A (Quick-Spell).
- **Resolution:** Quick-Spell resolves $\to$ Trap resolves $\to$ Attack resolves.

4.4 Unique Systems

- **Showdown Phase:** If two Level 7+ monsters battle, the game enters a cinematic state. The table dims, and a "Clash" animation plays out.
- **Resource Caps:** Max Deck Size: 60. Max LP: 50,000.
- **Conditional Limit:** Max 3 conditional effects per card.

---

5. Game Modes (Public Names)

Standardized nomenclature for internal and public use.

Mode	Format	Description
Standard Duel	1v1	Standard TCG rules. 8000 LP. Opponents sit across from each other.
Tag Team	2v2	16,000 Shared LP. Turn Order: A1 $\to$ B1 $\to$ A2 $\to$ B2. Voice chat open to all.
Raid Boss	2v1	Asymmetrical. 1 "Boss" Player (Double LP, Extra Cards, Boss Skill) vs 2 Challengers.
Battle Royale	FFA	4-Player Free-For-All. Last man standing. Diplomacy and betrayal encouraged.

---

6. Development Roadmap

Phase 1: The Greybox (Current)

- **Goal:** Functional Multiplayer Loop.
- **Visuals:** Grey Cubes and capsule avatars.
- **Deliverables:**
  - Connect to Photon.
  - Spawn a Deck.
  - Draw/Play cards (Networked).

    - Verify "Stack" logic.

Phase 2: The Art Injection

- **Goal:** Visual Identity.
- **Visuals:** Ricardo's initial models and the Room Environment.
- **Deliverables:**
    - Replace capsules with 3D Anime Characters.
    - Replace UI with styled assets.
    - Sam completes data entry for the "Starter Deck" (40 cards).

Phase 3: VR Integration & Polish

- **Goal:** Immersion.
- **Visuals:** Final lighting and particle effects.
- **Deliverables:**
    - Oculus/OpenXR Hand Tracking.
    - "Physical" Card throwing/interactions.
    - YouTube Member integration.

Sample Card Entry (For Sam)

Card Name: Red Ribbon Fleet

- **Frame:** Effect (Orange)
- **Attribute:** WIND (Select from Dropdown)
- **Race:** Machine (Select from Dropdown)
- **Keywords:** DirectAttack (Drag Keyword Asset here)
- **Description:** "If there are other Wind monsters, this card cannot be attacked."

---

7. Version Control Rules (The "Anti-Disaster" List)

1. **NEVER** move files using Windows Explorer. Use Unity Project Window only.
2. **ALWAYS** pull from dev before starting work.
3. **NEVER** touch the main branch directly.
4. **IF** you see a .meta file conflict, stop and call Georgi.
5. **NAMING CONVENTION:** All Card Data files must be named Card_[Name] (e.g., Card_GedoStatue). This prevents confusing them with the 3D prefabs.

⚙️ Automation: The Folder Script

To create the project structure automatically:

1. Create a new text file inside your project Assets folder named SetupFolders.bat.
2. Paste the code below inside.

3. Double-click it.

DOS

```
@echo off
mkdir _Shonen
cd _Shonen
mkdir Documents
mkdir _Scenes
mkdir Art
mkdir Audio
mkdir Data
mkdir Prefabs
mkdir Scripts
cd Art
mkdir Characters Environment UI Materials Animations CardIllustrations
cd ..
cd Audio
mkdir BGM SFX
cd ..
cd Data
mkdir Cards GameSettings Keywords
cd ..
cd Scripts
mkdir Core Network UI Cards
cd ..
cd Prefabs
mkdir Monsters Cards Effects
echo Folders Created Successfully!
pause
```

---

8. Technical Implementation (Alpha Status)

This section documents the systems currently live in the dev branch.

8.1 Networking Architecture (Photon Fusion)

We utilize **Shared Mode** to offload physics calculations to clients while keeping state authority on the Host for game logic.

A. The Connection Flow (FusionLauncher.cs)

- **Role:** Acts as the "Lobby" manager.
- **Behavior:**
    1. On Start, checks if we are in GameScene. If not, creates a NetworkRunner.
    2. Automatically connects to the dev room (Shared Mode).
    3. If GameScene is loaded, it waits 0.5s for network stability, then triggers the Spawner.
- **Logic:** Persistent Singleton (DontDestroyOnLoad).

B. Deterministic Spawning (FusionGameSpawner.cs)

- **Problem:** In Shared Mode, anyone can spawn anywhere.
- **Solution:** We enforce seating based on SharedModeMasterClient status.
    - **Host (Master Client):** Always spawns at **Seat A** (Transform: SeatA_Spawn).
    - **Client (Player 2):** Always spawns at **Seat B** (Transform: SeatB_Spawn).
- **Prevention:** The script checks runner.GetPlayerObject(runner.LocalPlayer) to prevent double-spawning.

8.2 Player Rig & Authority (PlayerSetup.cs)

Since all players spawn the same Prefab, we must strip control from non-local instances to prevent "Ghost inputs" (moving the other player's character).

- **Local Player (Me):**
    - Camera: **Enabled**
    - Audio Listener: **Enabled**
    - Movement/Raycaster: **Enabled**
- **Remote Player (You):**
    - Spawned() detects !HasStateAuthority.
    - **Disables:** Camera, AudioListener, Controller Scripts, and CardRaycaster.
    - *Result:* I see my view; I see your avatar (but cannot control it).

8.3 Data Architecture

A. Card Data (CardData.cs)

- **Type:** ScriptableObject
- **Key Feature:** Uses **Enums** for Rarity/Attribute/Race to prevent string errors (e.g., typing "Feind" instead of "Fiend").
- **Architecture:** Separation of Concerns. This file holds *Data* only. Logic is handled by external controllers.

B. Keyword System (CardKeyword.cs)

- **Type:** ScriptableObject

- **Purpose:** Reusable effect tags (e.g., PIERCING, DOUBLE_ATTACK).
- **Usage:** Drag-and-drop into the CardData inspector. Allows Sam to create new mechanics without writing code.

C. Interaction (CardRaycaster.cs)

- **Status:** *Debug Phase*
- **Function:** Casts a ray from the center of the PlayerCamera.
- **Visual:** Draws a Cyan debug line in the Scene view.
- **Future:** Will detect Card layer objects to handle "Hover" and "Click" events.