Retro Game Design Document (RGDD)

1. Introduction

This document serves as a **guideline** for designing retro-inspired games that adhere to **technical limitations** from past hardware generations while maintaining creative flexibility. The goal is to achieve **authenticity in spirit** without being bound by strict constraints.

---

2. Defining the Era & Hardware Inspiration

Select Console Era:

- **8-bit (NES, Master System, Game Boy)**
- **16-bit (SNES, Genesis, PC-Engine)**
- **32-bit (PS1, Saturn, N64 low-poly style)**
- **Handheld (GBA, DS, PSP)**
- **Early 3D (GameCube, PS2, Dreamcast, Xbox Classic)**

Core Hardware Constraints as Guidelines:

- **Graphics:** Limited palettes, resolution, and tilemaps
- **Audio:** Limited sound channels and chiptune synthesis
- **Memory Management:** Classic RAM-based loading methods
- **Processing Power:** Frame-based optimizations, fixed-time step logic
- **Game Length & Scope:** Shorter, replayable levels over massive open-worlds

---

3. Visual & Art Guidelines

Pixel Art & Sprites (2D Games)

✅ Keep sprite sizes era-authentic (e.g., NES: 8x8, 16x16, SNES: 32x32 max) ✅ Use a **limited color palette** (NES: 4 colors per tile, SNES: 15+1 transparency) ✅ Avoid excessive transparency effects unless mimicking known techniques ✅ Use **limited animation frames** to maintain classic movement fidelity

Low-Poly Models (3D Games)

✅ Keep **polygon count low**, similar to PS1/N64/GameCube assets ✅ Use **vertex shading and low-resolution textures** instead of modern shaders ✅ Mimic **texture warping and affine texture mapping** for authenticity ✅ Frame-based character animations (instead of interpolated motion capture)

---

4. Audio & Sound Guidelines

✅ **Limited audio channels** (NES: 5 channels, SNES: 8 channels, GBA: 6 channels) ✅ **Mono or stereo sound** (based on era—PS1 introduced full stereo mixes) ✅ **Chiptune-style compositions** instead of full orchestrations for retro styles ✅ Reverb and echo effects **limited to mimic console audio chips**

---

5. Gameplay Mechanics & AI

✅ **Simple AI patterns** (predictable enemy movement, scripted difficulty scaling) ✅ **Input delay considerations** (replicate hardware-based controller polling) ✅ **Physics system limits** (2D games: no floating point physics, 3D: vertex-based collision) ✅ **Level design based on tiles, screens, or rooms instead of procedural generation** ✅ **Platformers:** Pixel-perfect jump physics, acceleration curves similar to original games ✅ **RPGs:** Limited overworld size, text-based dialogue systems, turn-based or fixed-action combat

---

6. Game Length & Structure

✅ **Shorter but highly replayable levels** (mimic classic stage-based design) ✅ **Password or limited save systems** (if replicating pre-memory card era) ✅ **Single-player or local multiplayer focus** (if mimicking cartridge-based games) ✅ **Intentional pacing inspired by original era loading speeds & memory limits**

---

7. Rendering & Performance Guidelines

✅ **Fixed frame rate based on console specs** (NES: 60 FPS, SNES: 60 FPS, PS1/N64: 30 FPS) ✅ **Limited screen resolution & scaling options** (Match console's native output but allow modern scaling) ✅ **Use sprite batching techniques for optimization (if applicable to era)** ✅ **No real-time dynamic lighting (unless mimicking tricks like pre-rendered lightmaps)** ✅ **No physics engines unless specifically designed to feel era-authentic**

---

8. Modern Adaptations While Maintaining Authenticity

🚀 **Allowed Modern Enhancements:**

- Quality-of-life improvements (save states, accessibility options, widescreen support)
- Improved controls (smoother input, customizable buttons)
- Online leaderboards or multiplayer (if applicable without breaking retro feel)
- Modern backend (built in Unity, Unreal, Godot but following era-authentic constraints)

🚫 **Avoid Breaking Authenticity:**

- Overuse of high-resolution assets with retro gameplay (breaks immersion)
- Excessively smooth animations that don't match original era techniques
- Open-world structures in games meant to feel like classic level-based gameplay
- Overcomplicating UI with modern elements (classic games had minimal UI)

---

9. Conclusion

This **Retro Game Design Document (RGDD)** is meant to **guide development** towards an experience that feels **authentic** to classic gaming eras **without strictly enforcing outdated technical limitations**.

By following this document, you can ensure your game is a **spiritual successor to retro titles**, staying true to their **style, constraints, and gameplay philosophies** while still leveraging modern tools where necessary.

---

10. Next Steps: Customizing This Document

🔹 Choose an era-specific focus and set guidelines accordingly 🔹 Define the game's resolution, art style, and sound limitations 🔹 Outline gameplay mechanics and AI that match classic systems 🔹 List which modern features will be allowed vs. avoided for authenticity