GAME DESIGN DOCUMENT

Relic Runner

Purpose

This document explains the design intent behind Relic Runner. It justifies why specific mechanics, systems, and constraints exist, and demonstrates that design decisions were made deliberately to support a clear player experience-not by accident or convenience.

High Concept

Relic Runner is a **2D top-down action–puzzle game** where the player explores dangerous ruins, collects ancient relics, avoids hazards and enemies, and escapes each level alive.

Targe Player Experience

The player should feel:

- **Alert** – constantly reading space and threats
- **In control** – success comes from movement decisions, not luck
- **Challenged but treated fairly** – failure is understandable and instructional
- **Satisfied** – short levels encourage “one more try” improvement

---

Core Design Pillars

Note to students:

These pillars guide all decisions.
If a feature violates a pillar, it must be changed or removed.

1. Movement-First Gameplay

Movement is the primary verb in Relic Runner. Every challenge—enemies, hazards, collectibles, and level layout—exists to test how well the player moves through space. There are no complex combo systems or deep inventories; instead, mastery comes from positioning, timing, and route planning. This keeps the game readable, focused, and mechanically cohesive.

2. Readable Danger and Fair Failure

All threats in the game are designed to be **visible, consistent, and predictable**. Enemies telegraph their behavior, hazards are visually distinct, and damage is never hidden or random. When the player fails, they should understand *why* they failed and feel that a better decision could have prevented it. This supports learning and reduces frustration.

3: Simple Systems, Emergent Challenge

Relic Runner uses a small number of simple mechanics that interact in meaningful ways. Difficulty emerges from **how elements are combined and placed**, not from increasing numerical complexity. This allows the game to scale in challenge while remaining approachable and easy to understand.

4: Short, Focused Levels

Levels are intentionally short (30–90 seconds) and built around a single idea or challenge. This supports rapid iteration, replayability, and flow. Short levels also make failure less punishing and encourage players to experiment, learn, and improve rather than play cautiously.

Target Platform & Audience

- **Platform:** PC (Windows primary)
- **Engine:** Godot 4.x
- **Audience:** Casual to mid-core players
- **Skill Level:** Beginner-friendly, mastery-rewarding

Core Gameplay Loop

Enter Level → Observe Space → Move & Avoid → Collect Relics → Reach Exit → Score & Progress

Players enter a level, quickly assess the layout and threats, move carefully through the space while avoiding hazards and enemies, collect relics to increase score, and reach the exit. Performance is evaluated through scoring and completion, encouraging replay and mastery rather than one-time completion.

Note to students: This loop must remain intact throughout development.

Player Mechanics

Base Abilities

Note to students: These are required.

- Four-directional movement
- Collision-based damage
- Relic collection
- Level exit interaction

Advanced Abilities

Note to students: At least two advanced abilities of your choice are required for the final project. These are to be added/implemented later on but you may decide/choose now. Examples:

- Dash (short burst movement)
- Temporary invulnerability
- Ranged interaction (projectile or trigger)

Player Stats System

- Health
- Max health
- Score
- Relics collected
- Lives (optional)

Enemy Mechanics

Note to students: List down types of enemies (zombies, monsters, wild animals) and their behavior (ex: patrol, chase, ranged).

Enemies must:

- Follow predictable rules
- Use simple state machines
- Escalate difficulty through placement, not stats

Enemies exist to **control space**, not overwhelm the player.

Hazards

Note to students: List down all hazards in your game. Examples:

- Static hazards (spikes, damage zones)
- Moving hazards (patrolling traps, timed obstacles)

Hazards are consistent and visually readable, reinforcing the “fair failure” pillar.

Relics & Collectibles

- Increase score
- Optional: unlock doors or paths
- Provide audio/visual feedback when collected

*Note to students: Relics must always be **visible and intentional**.*

Level Design

Teaching Mechanics

Levels introduce mechanics safely:

1. A low-risk space demonstrates the mechanic
2. The player applies it in a simple challenge
3. The mechanic is combined with another element
4. The level ends with a clear success test

Difficulty Progression

Difficulty increases through:

- Tighter spaces

- Faster or overlapping threats

- Combined enemy and hazard placement

What a Good Level Looks Like

A good Relic Runner level:

- Has a clear visual goal (exit)

- Is readable at a glance

- Can be completed quickly once mastered

- Teaches or reinforces exactly one main idea

UI & Feedback

Note to students: choose what UI and feedback elements you plan to implement.

- Health bar
- Score display
- Pause menu
- Main menu
- Game over / victory screen
- Animation feedback
- Audio feedback

UI must be:

- *readable at all resolutions*
- *responsive to game events*
- *animated subtly (no static dead UI)*

Audio & Game Feel

Note to students: choose what sound/music and effects that you plan to use to define and enhance the game feel.

- Background music

- Jump/interaction sounds
- Damage sound
- Collect sound

Enhancements:

- Screen shake on damage or major events
- Hit stop on impactful actions
- Particle effects for movement, pickups, and enemy defeat

These techniques reinforce player actions and improve clarity without adding mechanical complexity.

Save & Progression

- Save unlocked levels
- Save best score/time
- Save settings (audio, accessibility)

Accessibility Requirements

At least **two** must be implemented:

- Difficulty options (Easy / Normal / Hard)
- Reduced screen shake option
- Colorblind-friendly hazard colors
- Control remapping
- Configure audio levels

Note to students: Accessibility features are treated as part of design quality, not optional extras. This will be graded.

Art & Visual Style

Note to students: explain your game's art and visual style. Examples:

- *Clean, readable sprites*

- *High contrast between player, enemies, hazards*
- *Consistent tile size*
- *Minimal visual noise*

*Assets may be sourced from free libraries, but must be **credited**.*

Cut or Changed Features

Note to students: This is to be filled up later in the sem and only if you have really removed or changed some features that you initially planned on implementing.

Example:

Cut Feature: Complex Inventory System

An early idea included multiple relic types with unique effects. This was removed because it distracted from the movement-first focus and increased cognitive load without improving the core experience. A simpler scoring-based relic system better supports clarity and replayability.

Changed Feature: Enemy Scaling

Originally, enemies were planned to scale in health and damage. This was changed to layout-based difficulty after testing showed that higher stats felt unfair and reduced readability. Difficulty is now driven by placement and timing, aligning better with the design pillars.

Final Design Note

Relic Runner is intentionally modest in scope.

Its goal is not to impress through quantity of features, but through **clarity, consistency, and polish**. Every system exists to serve movement, readability, and player learning. Anything that did not support those goals was changed or removed.

Additional Notes to Students (not part of the Game Design Document)

Technical Constraints

- Godot 4.x only
- GDScript only
- Organized project structure
- Typed variables encouraged

Performance target:

- 60 FPS with 10+ enemies on screen

Success Criteria (What "Done" Means)

A successful **Relic Runner** build:

- Is fully playable from menu to ending
- Has clear win/lose conditions
- Feels responsive and fair
- Demonstrates all required systems
- Includes documentation and credits
- Reflects iteration based on playtesting

Non-Goals (Important!)

Relic Runner is **not**:

- A massive open world
- A story-heavy RPG
- A multiplayer game
- A graphics showcase

Creativity, imagination, and the desire to build better games are admirable characteristics for game designers and developers to have. But, whether as students of professional game/designers and developers, it is important to consider all project constraints (such as time, and technology) and make sure that your design is feasible.

Student Responsibility Statement

Students are responsible for:

- Maintaining a working build each week
- Documenting changes
- Iterating based on feedback
- Asking for help early

Broken builds cost more than missing features.

Final Note to Students

You are not building a “perfect” game.
You are building a **finished** one.
A small, polished game beats an ambitious, broken one—every time.