Requirements Specification Document

Akemi Isles Video Game

Table of Contents

1 Scope

1.1 Identification
This document defines the requirements for Akemi Isles, a 2D narrative-driven puzzle adventure game inspired by the game Poptropica and mythological stories.

1.2 System Overview
Akemi Isles is a side-scrolling adventure game where the player explores narrative themed islands, solves puzzles, interacts with NPCs, and collects items to progress through the story. The game emphasizes storytelling, exploration, and multimedia integration (art and interactive systems). For the semester, we will be focused on creating one coherent island that is simple, yet enjoyable to play.

1.3 Document Overview
This document specifies the functional and non-functional requirements of Akemi Isles, including gameplay, interface, art, and computing resources.

2 Applicable Documents

2.1 Video Preface
This section will be updated over time as more online tutorials are used for the project.

2.2 Player Character Animation
Idle, Jump, and Run Animations - Unity 2D (YouTube: Game Code Library)
This tutorial will guide player movement and animation implementation.

2.3 Inventory System
Let’s Make an Inventory System in Unity (YouTube playlist: Night Run Studio)
This tutorial will guide the creation of Akemi Isles’ inventory system.

3 Functional Requirements

3.1 Subsystem Divisions

- Game Manager CSC – Oversees game state, transitions, saving/loading.
- Player CSC – Manages character movement, interactions, collision, and respawning.
- Inventory Manager CSC – Handles item collection, storage, persistence, and UI.

- Narrative & Dialogue CSC – Controls branching storylines and NPC interactions.
- Puzzle CSC – Implements puzzle mechanics and validation.
- Art & UI CSC – Provides consistent visuals, HUD, scaling, and menus.

3.2 Game Manager CSC

3.2.1 The Game Manager shall initialize and manage core game states (e.g., start menu, active gameplay, pause, and game over).

3.2.2 The Game Manager shall save and load player progress, including completed levels, collected items, and story checkpoints.

3.2.3 The Game Manager shall reset all relevant data when a new game is started

3.2.4 The Game Manager shall handle transitions between scenes, levels, or narrative segments

3.2.5 The Game Manager shall track overall game completion percentage and display it to the player.

3.3 Player CSC

3.3.1 The Player subsystem shall allow the player character to move, jump, and interact with objects in the game world.

3.3.2 The Player subsystem shall allow the player to collect items and interact with NPCs.

3.3.3 The Player subsystem shall detect collisions between the player and world objects, enemies, or hazards.

3.3.4 The Player subsystem shall respawn the player at a designated checkpoint when transitioning between scenes and defeat.

3.4 Inventory Manager CSC

3.4.1 The Inventory Manager subsystem shall allow the player to add and view items in their inventory for each individual island.

3.4.2 The Inventory Manager subsystem shall support stackable items (e.g., multiple coins or keys).

3.4.3 The Inventory Manager subsystem shall display item details, including name, description, and quantity.

3.4.4 The Inventory Manager system shall allow the player to use or equip items where applicable.

3.4.5 The Inventory Manager subsystem shall persist inventory contents across scenes and sessions.

3.5 Narrative and Dialogue CSC

3.5.1 The Narrative and Dialogue subsystem shall display branching dialogue with NPCs based on character choices.

3.5.2 The Narrative and Dialogue subsystem shall trigger dialogue sequences when the player interacts with specific characters or objects.

3.5.3 The Narrative and Dialogue subsystem shall allow store player dialogue choices and reflect them in future narrative events.

3.5.4 The Narrative and Dialogue subsystem shall synchronize dialogue with visual cues (e.g., animations and cutscenes).

3.6 Puzzle CSC

3.6.1 The Puzzle subsystem shall provide interactive puzzles that the player must solve to progress through the game.

3.6.2 The Puzzle subsystem shall validate puzzle solutions and unlock rewards, story events, or new areas upon success.

3.6.3 The Puzzle subsystem shall allow resetting puzzles to their initial state if the player fails.

3.6.4 The Puzzle subsystem shall increase puzzle complexity as the game progresses.

3.6.5 The Puzzle system shall store puzzle completion state so solved puzzles are not repeated unnecessarily.

3.7 Art and UI CSC

3.7.1 The Art and UI subsystem shall present consistent visual style across menus, dialogue boxes, and in-game environments.

3.7.2 The Art and UI subsystem shall provide a HUD displaying essential player information (settings, inventory shortcuts, islands to choose from).

3.7.3 The Art and UI subsystem shall scale UI elements to fit different screen resolutions.

3.7.4 The Art and UI subsystem shall support tooltips and contextual icons for intractable objects.

4 Performance Requirements

4.1 Game Initialization Time

The system shall load the start menu and first playable scene within **10 seconds** on a standard laptop (8 GB RAM, integrated graphics). This ensures the player can quickly begin gameplay without excessive delays.

4.2 Frame Rate Consistency

The system shall maintain a minimum frame rate of **30 FPS** during active gameplay on standard hardware. This requirement ensures smooth player movement, puzzle interaction, and dialogue navigation.

4.3 Scene Transition Responsiveness

The system shall complete transitions between scenes (e.g., moving between areas of an island or starting a puzzle) in **under 5 seconds**. This prevents interruptions to narrative flow and exploration.

4.4 Inventory and UI Responsiveness

The system shall update inventory displays, dialogue boxes, and other UI interactions in under 1 second after player input. This ensures immediate feedback when selecting items, interacting with NPCs, or navigating menus.

4.5 Memory Usage

The system shall not exceed 1 GB of RAM usage during normal gameplay. This requirement ensures that the game runs reliably on standard laptops without resource overload.

4.6 Save/Load Performance

The system shall save player progress in under 3 seconds and load saved data in under 5 seconds. This ensures efficient resumption of gameplay and reduces disruption to the user experience.

5 Environment Requirements

The following are the hardware and software requirements for Akemi Isles.

5.1 Hardware Requirements

Category	Requirement
Processor	Intel Core i5-10300H or similar (minimum) / Intel Core i9-12900HK (development machine)
RAM	8 GB (minimum), 32 GB (development machine)
Hard Drive Space	2 GB free space for installation and assets
Display	1920×1080 resolution, integrated or discrete GPU support
Graphics	Intel Iris Xe Graphics or similar (minimum), NVIDIA GeForce RTX 3050 Ti 4 GB (development GPU)
Input Devices	Standard keyboard, mouse, optional gamepad support

The development machine used is an XPS 15 9520 (Intel i9-12900HK, RTX 3050 Ti 4 GB, 32 GB RAM). Minimum specifications ensure broad accessibility for standard laptops/PCs.

5.2 Software Requirements

Category	Requirement
Operating System	Windows 10/11 (64-bit) or macOS Monterey (or later)
Game Engine	Unity 2D (latest LTS release)
Programming	C# (within Unity scripting environment)
Version Control	GitHub for collaboration and source control as well as BOX to save weekly progress as insurance
Art Tools	Piskel (sprite creation), Adobe Photoshop or GIMP (optional for assets)
IDE/Editor	Visual Studio Code / Visual Studio Community Edition

Unity 2D with C# will be the primary development environment. GitHub ensures collaborative version control, with BOX as additional insurance.