Acid Rain

Team Members:
Franz Mischke: Developer, Team Lead
Peter Hilbert: Developer
Manjot Kaur: Developer

This game is based off of an apocalyptic real world situation. The user has to guide their player through an apocalyptic town and Seek Shelter From the acid rain which will occur every 10 seconds and last about 5 seconds. The user will have to navigate their player through obstacles to get to shelter, which will cover them from the acid rain. The acid rain will ultimately kill them if the player is under the acid rain for more than 3 seconds. Once the player is inside a shelter it is not safe for them to stay there still and wait out the rain. They will have to kill enemies which are also inside the shelters. Enemies will not be able to leave the shelters. When the rain stops the player can leave the shelters and not worry about being harmed by an enemy. This game can be completed by reaching the end of this apocalyptic town where it will stay sunny and they will not be any enemies

The creation of this game will pose some key technical challenges:

Module for rain “weather”
Our team will create a custom module for rain that will stay on for 5 seconds and occur every 10 seconds in the game.

Custom fluid dynamics
We will implement a custom module for fluid dynamics so the rain will pull up in areas with lower elevation. This will give the user a wet environment feel, but does not harm them when it is sunny outside.

Specialized pathfinding for enemies
Our team will create a custom module so enemies will attack the player in a certain radius and if they are inside the shelter that the enemy is in.

Health bar
We will incorporate a health bar so the years and knows how much health the player has.

Module Specification

Overview and Purpose

The AcidRainModule is a custom C++ Godot module that implements a fully deterministic environmental hazard system for 2D games. Its primary purpose is to control global acid-rain cycles, track player exposure, and provide a clean, event-driven API for game logic and UI feedback. The module centralizes all timing, state changes, and exposure logic so that gameplay code stays simple and focused on reactions rather than managing time manually.

In our prototype (Acid Rain), this module drives the core tension loop: 10 seconds clear, 5 seconds acid rain, with player death occurring after 3 seconds of exposure. The system exposes this logic through a C#-accessible API, allowing the game to read cycle state, subscribe to events, and query player safety conditions.

This module addresses a clear gap in the base engine: Godot has no built-in concept of global environmental hazards, persistent timers, synchronized world effects, or exposure-tracking utilities. Implementing this as a module (instead of GDScript/C# scripts) guarantees consistency, low-level control, and engine-level performance.

Architecture

1. AcidRainManager

- Handles a global state, either RAINING or CLEAR.
- Handles timers for both phases.
- Emits signals/events when the cycle changes.
- Stores global configuration values (cycle lengths, damage thresholds).
- Exposed to C# via GDCLASS bindings.

2. ExposureTracker

- Tracks exposure time for any entity/node registered with it (primarily the player)
- Provides an interface to check exposure for an entity using raycasting.

This design follows several common design patterns, including:

- Singleton - AcidRainController is meant to be globally accessible and unique
- Observer - The controller emits signals that can be used by the game to track cycle transitions

API Documentation

Class: AcidRainManager

Methods

- `void start_cycle()` - Begins the repeating rain/clear loop.
- `bool is_raining()` - Returns true during the rain phase.
- `float get_time_until_rain()`
- `float get_time_remaining_in_phase()`
- `void set_cycle_durations(float clear_seconds, float rain_seconds)`
- `void set_warning_time(float warning_time)` - Set the time which players will be warned (e.g. 2 seconds before rain). Must be smaller than `clear_seconds`.

Signals

- `rain_started()`
- `rain_stopped()`
- `pre_rain_warning()`
- `exposure_tick(Node2D entity, float exposureTime)` - emits every game tick whenever an entity is determined to be exposed, allowing that particular entity to handle its own logic pertaining to exposure time.

Class: ExposureTracker

Methods

- `void register_entity(Node2D entity)`
- `bool is_entity_exposed(Node2D entity)`

Godot Integration Approach

The acid rain system will be implemented as a custom C++ module for Godot. We will be extending Godot version 4.4 using the process outlined here.

Comparison with Existing Solutions

Godot includes general-purpose timers and signals, but **no existing system** provides:

- Persistent environmental cycle logic
- Exposure-tracking utilities
- Global hazard state management
- Engine-level C++ implementation for performance and determinism

Most Godot projects implement weather/hazards in GDScript scenes, which leads to:

- Inconsistent timing
- Difficult cross-scene communication
- Repetitive logic
- No engine awareness of global hazard states

Our module centralizes all of that under a **reusable, general-purpose system** that can be dropped into any 2D game, not just ours.

Game Design

Game Concept and Genre

Acid rain takes place in an apocalyptic town ruined by waves of corrosive acid rain. The player controls a lone survivor trying to cross the ruined City to reach the final destination, a safe haven where the sun is always out and there is no danger. The player will have to undergo environmental factors such as acid rain that will fall every 10 seconds and last about 5 Seconds forcing the player to consistently move between shelters while dealing with enemies to occupy those safe spaces. If the player isn't fast enough and is under the acid rain for more than 3 seconds the acid rain will then corrode their body and the player will die.

The twist to this game is that safety always comes with the price as a player deals with the acid rain and seeks shelter from it. The shelters will house a number of enemies that the player will then enter and have to deal with. The player will have to navigate through these buildings dealing with enemies to reach the exit of the shelter to move onto the next shelter. Once the player leaves the shelter the enemy will not be focused on them anymore. When the player is outside the shelter they must find the next shelter to hide from the rain once they reach the end of the city then they will be met with a clear sky with a non-looking apocalyptic City where they will be safe from acid rain and enemies.

This game is a survival game that will have the user focused at every given moment during the game. This fast-paced game will include player side to side movement, jumping actions, and fighting actions. As this game is a single player game it will focus all aspects on a solo player creating an immersive game.

Core Gameplay Mechanics

Movement and Traversal

- Player Movement:
    - Standard side to side gameplay where the user will use left and right arrows to move side to side, spacebar to jump, left Mouse click to attack.
- Level Layout:
    - Objects to get into above ground shelters giving the game a parkour aspect.
    - Environmental hazards such as areas a player cannot jump out of, encouraging the player to think about their next move.

Environmental Factor

- Acid Rain:
    - Rain phase: Approximately 5 Seconds of intense acid rain
    - Clear/Sunny phase: Approximately 10 seconds of clear Sunny environment for the player to move around to the next shelter.
- Exposure:
    - A custom module will allow the game to track how long the player has been under the rain for.
    - If the player is under the rain for more than 3 seconds the player will corrode from the acid rain and die.
    - The rain exposure is based off of time rather than amount of rain hitting the player
- Environmental Feedback:
    - The player will know the rain is about to start as the sky starts to get darker.
    - There will be an audio effect of thunder approximately 2 seconds before the rain starts

Combat and Enemies:

- Enemy Behavior:
    - Enemies will only exist within buildings and are free to move around each level of the building.
    - Enemies will not be able to exit the buildings.
    - Enemies will be able to only see you once you are inside the building,Once you have exited the building the enemy will stop following you.
- Combat:
    - The player can attack with a simple weapon such as a sword.
    - killing all the enemies in the shelter makes it safer for future rain cycles if the player needs to turn around and go to the previous shelter

Health, Death and Progression:

- Health system:
    - If the player gets hit three times by an enemy the player will die.
    - The player will not gain any health from Clearing a shelter and moving onto the next one

    - For every second under the rain your health will go down by 1/3 percent. An example would be if a player is under the rain for 2 seconds and manages to get inside a shelter on the third second the player will have only one third of the health they started at.
- Death Conditions:
    - If the player stays under the rain for more than 3 seconds the player will die.
    - If the player gets hit by an enemy three times the player will die.
- Win Conditions:
    - The player will reach a final safe Zone at the end of the level which will indicate a win condition.
    - There will be no enemies and there will be no rain.
    - The player will be prompted if they want to play again.

Target Audience and Platform

Target Audience

- Age range:
    - Users should be approximately 13 years of age to play this game, as this game will have some violent aspects.
- Player Experience:
    - The player should expect to have an immersive survival side to side game .
    - The player should enjoy overcoming challenging time based obstacles.

Platforms

- Systems:
    - This game can be played in either Windows or IOS systems
- Controls:
    - Space Bar = Jump action
    - Left Arrow Key = Left movement
    - Right Arrow Key = Right movement
    - Left Mouse Button = Attack
    - These controls should remain the same for either system.

Visual Style and Aesthetic Direction

Overall Look

- Color palette:
    - Clear Phase: The clear phase in the apocalyptic town will have darker colors, Distinguishable from the winning condition clear phase.
    - Rain Phase: The rain phase will include darker colors with yellow and green colors of rain indicating poison.
- World Tone:
    - Broken pathways leading to buildings, abandoned cars in the background, houses with little to no light, and overgrowth of shrubs.
- Rain Visuals
    - Rain will be thin sharp streaks falling from top down.
    - Puddles will form on the ground for visual aspects.

Technical implementation Plan

Development Timeline With Milestones

- Phase 1 – Engine Setup & Module Framework (Oct 30 - Nov 5)
    - Install and configure Godot 4.4 and C++ module build environment.
    - Create initial AcidRainModule structure and GDCLASS bindings.
    - Implement module registration.
- Phase 2 – Core Acid Rain Cycle System (Nov 6- Nov 12)
    - Implement 10s clear / 5s rain cycle.
      Add configurable parameters and signals.
      Add global phase timers.
- Phase 3 – ExposureTracking & Player Integration (Nov 13 - Nov 19)
    - Implement ExposureTracker for entity exposure.
    - Integrate exposure logic with player health.
    - Implement a health bar UI.
- Phase 4 – Enemy AI and Pathfinding (Nov 20 - Nov 26)
    - Implement shelter-only enemy AI and detection.
    - Add combat detection and movement logic.
    - Integrate enemies with level layout.
- Phase 5 – Environmental Effects & Visuals (Nov 27 - Dec 1)
    - Implement rain shader, thunder FX, and puddle effects.
    - Add environmental audio warning cues.
- Phase 6 – Level Design & Final Integration (Dec 2 - Dec 3)
    - Build a complete city level.
    - Add shelters, enemies, and traversal obstacles.
    - Implement win-state logic.
- Phase 7 – Testing, Bug Fixing & Optimization (Dec 4)
    - Conduct QA tests for modules, AI, and exposure logic.
    - Optimize shaders and performance.

Technology Stack and Dependencies

- Game Engine: Godot Engine 4.4
- Languages: C++ for engine modules, C# for gameplay logic
- Dependencies:
    - SCons build system
    - MSVC or Clang/GCC compilers
    - GitHub for version control
    - Aseprite/Photoshop for art, Audacity for audio
- Custom Engine Components:
    - AcidRainManager – global hazard controller
    - ExposureTracker – entity exposure system

    - Fluid Dynamics – puddle simulation
    - Custom Pathfinding – shelter-restricted AI behavior

Testing Strategy and Success Metrics

- Testing Strategy:
    - Unit Testing – cycle timing, exposure thresholds, signals.
    - Integration Testing – player + rain module, enemies + shelters.
    - Playtesting – level completion, warnings, combat feedback.
    - Performance Testing – rain rendering, puddle simulation, AI load.
- Success Metrics:
    - Rain/clear cycle accurate within ±0.1 seconds.
    - Exposure detection matches shelter coverage 100%.
    - The player dies exactly at 3 seconds exposure.
    - The game maintains 60 FPS during heavy rain.
    - The level is complete without softlocks.

Risk Assessment and Mitigation Strategies

- C++ Module Instability
    - Risk: Crashes due to memory mismanagement.
    - Mitigation: RAII, smart pointers, incremental build testing.
- Exposure Tracking Errors
    - Risk: Incorrect exposure resets or false positives.
    - Mitigation: Debug overlays, exhaustive movement tests.
- Enemy AI Failures
    - Risk: Enemies escaping shelters, collisions, stuck movement.
    - Mitigation: Tilemap navigation, collision tuning, AI debug paths.
- Visual Performance Drops
    - Risk: Rain shaders or puddles lowering FPS.
    - Mitigation: GPU-based effects, scalable quality settings.
- Timing Desync Between Effects and Logic
    - Risk: Rain visuals not matching hazard timer.
    - Mitigation: All timing controlled through one authoritative C++ timer
- Player Confusion With Hazard Warnings
    - Risk: Players do not understand exposure rules.
    - Mitigation: UI alerts, thunder audio, tutorial introduction.

Team Collaboration Plan

Team Roles

Team Lead/Developer - Franz Mischke:

- Manages project, directing team into the right direction
- Delegates task to team members
- Primary developer for core systems
    - AcidRainManager module
    - ExposureTracker
    - Rain shader and hazard effect
- Coordinates team meeting
- Ensures technical integrity across modules

Developer - Peter Hilbert

- Focuses on enemy AI design and pathfinding. Along will enemy behavior in shelters.
- Implements combat interactions with enemies, and detection radius for enemies.
- Assist with level layout environment.

Developer - Manjot kaur

- Focuses on player health systems and ui implementation of environmental factors.
- Design the level environments, such as structures and obstacles.
- Implements audio que for environmental factors.
- Assist with bug detection and problem solving with other team members.

Communication standards

Discord:

- Discord will be the main source of communication with the team.
- We will be holding a meeting regarding the project on discord.
- Weekly meetings will be announced in discord general chat so team members are aware.

Github

- Github will be where we will upload code so the team can access new and update code.
    - Commits - team member can commit to a separate branch
    - Branches - Team members will upload commits to a branch with a description of what was added or changed.

- Pull request - The team will create a pull request which another team member will look over and merge.

Citations

- Anon. Custom modules in C++. Retrieved November 18, 2025 from https://docs.godotengine.org/en/stable/engine_details/architecture/custom_modules_in_cpp.html