V-RADS

Design Document

Concept

The Virtual Radiation Awareness & Detection System (V-RADS) is a VR simulation that places players in the role of a safety technician traversing a hazardous environment. The objective is to identify active radiation sources using a virtual Geiger counter. This tool is designed to train nuclear facility employees in radiation emergency protocols within a safe, controlled setting.

Project Justification

Virtual Reality addresses the limitations of traditional safety training, which often fails to capture the physical reality of invisible hazards. V-RADS solves three critical training gaps:

- **Spatial Perception:** By placing the user inside a 1:1 scale environment, the simulation reinforces the Inverse Square Law, demonstrating how physical distance significantly reduces exposure. This practical application cements the core safety principle of ALARA (As Low As Reasonably Achievable).
- **Zero-Risk Environment:** Users experience high-stress hazard scenarios and practice response protocols with zero real-world physical risk.
- **Tool Familiarity:** The simulation builds muscle memory for equipment handling, specifically the operation of a Geiger counter, which cannot be effectively replicated using standard mouse and keyboard inputs.

Development Environment

- **Engine:** Unity 6.3
- **Language:** C#
- **Toolkit:** XR Interaction Toolkit for VR mechanics
- **Version Control:** Git/Github
- **IDE:** Visual Studio 2022
- **Tested On:** Meta Quest 2

Gameplay

The goal of the game is to locate all active radiation hazards. You use the audio and visual feedback from the geiger counter to find the general direction of radiation, then locate it. Interact with the hazard to fix it and get visual confirmation that was an active hazard with text and a safe green material.

Mechanics & Controls

- **Movement:** The simulation supports physical movement, however, continuous joystick locomotion is implemented to allow navigation beyond the physical boundaries of the room the user is in.
    - Left Joystick - Continuous Movement
    - Right Joystick - Snap Turn
- **Geiger Counter:** The detector is an XR Grab Interactable that provides audio-visual feedback.
    - **Visuals:** A custom algorithm approximates the micro-Sievert per hour (uSv/h) rate based on the total radiation intensity. While this is a simplified calculation for real-time performance, it effectively demonstrates the relationship between proximity and dosage.
- **Hazards:** The system utilizes the Fisher-Yates Shuffle algorithm $(O(n))$ to efficiently randomize active hazard locations at the start of each round. This ensures a strictly unbiased distribution of the 7 active hazards among the 18 potential spawn points.
- **Inverse Square Law:** Radiation intensity is calculated using the formula $Intensity = \frac{1}{d^2}$, where *d* is the distance to the source. This ensures radiation drop-off behaves realistically, similar to the dissipation of light from a source.

Audio

- **Locomotion Audio:** To maintain player comfort and prevent motion sickness, camera shake effects were omitted. Instead, footstep sound cues are utilized to provide movement feedback without disrupting the user.
- **Geiger Logic:** The click frequency is calculated by dividing the detector sensitivity by the current radiation intensity. This determines the delay interval between audio clips, with random pitch changes added for variety.

```
// Determine target delay based on intensity and sensitivity
float targetDelay = Mathf.Clamp(sensitivity / totalIntensity, minClickDelay, maxClickDelay);

if (Time.time >= nextClickTime)
{
    // Play click sound and haptics
    PlayClick(totalIntensity);

    float randomFactor = Random.Range(0f, targetDelay * 0.2f);
    nextClickTime = Time.time + targetDelay + randomFactor;
}
```

```
void PlayClick(float intensity)
{
    // Don't play if barely any radiation (prevents single random clicks far away)
    if (intensity <= 0.1f) return;

    // Randomize pitch slightly for variety
    audioSource.pitch = Random.Range(0.95f, 1.05f);
    audioSource.PlayOneShot(audioSource.clip);
```

- **Background audio:** To immerse the user even more I added a subtle background audio to enhance the feeling of being in a nuclear warehouse.

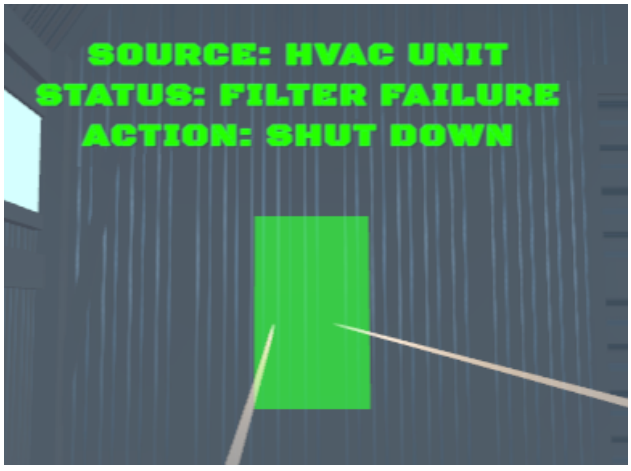
User Interface

- **Title Screen:** Floating Menu that has the game cover and buttons to start the simulation, also a controls panel

- **In-Game HUD:** Slider showing how much dosage you are currently at

- **Geiger Counter Text:** Created a system to convert radiation intensity to Microsievert per hour and displayed it on the geiger

- **Successful Found Hazards:** Floating Success text appears above a hazard when fixed, it explains the hazard and how to deal with it



- **Endgame:** Success or Gameover canvas based on if the user finds all the radiation hazards or goes over the max dose, both have restart and menu buttons

Demo Video & Download

https://drive.google.com/drive/folders/1sVl2wLMsn7llDb0dOHRZL96gycf9kD1Q?usp=sharing

Github Repository

https://github.com/Marsh14/V-RADS

Credits

Source: https://www.nrc.gov/docs/ML1121/ML11210B521.pdf (For Inverse Square law)
ALARA Source: https://www.cdc.gov/radiation-health/safety/index.html
uSv/h Source: https://www.unscear.org/unscear/en/areas-of-work/radiation-faq.html#:~:text=The%20rate%20of%20accumulation%20is,hour%20(%C2%B5Sv%2Fh).
Geiger Counter: https://sketchfab.com/3d-models/geiger-counter-25b8e5c2de1148c0b991bb239efa1808
Hazard Assets: https://assetstore.unity.com/packages/3d/environments/industrial/rpg-fps-game-assets-for-pc-mobile-industrial-set-v3-0-101429
Interior: https://sketchfab.com/3d-models/warehouse-fbx-model-free-daa7fd3ff88945298d00045ca40a4c03
Font: https://fonts.google.com/specimen/Black+Ops+One (Black Ops One)
Hazard Signs: https://assetstore.unity.com/packages/3d/props/exterior/three-signs-iso-7010-188492
Caution tapes: https://sketchfab.com/3d-models/caution-tapes-a082d307df38402d8eaed443c2e42501