Ant Colony: Ant WebXR First-Person Survival Shooter in Three.js

Eileen Rashduni Siddhartha Tamma
Department of Electrical and Computer Engineering
Rutgers, The State University of New Jersey
ECE 376/571 - Spring 2026 er816@scarletmail.rutgers.edu st1223@scarletmail.rutgers.edu

***Abstract*—We present *Ant Colony: Descent*, a browser-based WebXR first-person survival shooter built with Three.js r128. The player navigates a procedurally generated underground ant colony, defeats AI-driven worker ants, and destroys a Queen Ant boss. The experience runs from a single HTML file in both desktop and immersive-VR modes with no build step. Advanced features include a full WebXR session with dolly rig, dual-hand controller input, a six-weapon inventory with 3D hand models (two of which use Blender-authored `.glb` assets), a boss enemy with a canvas-texture health bar, and a particle-based atmosphere system.**

I. Project Overview

Ant Colony: Descent runs from a single `index3.html` file using Three.js r128 and the WebXR Device API, no build step or server-side code required. The player is shrunk to ant-scale and must descend through a branching underground colony, eliminating worker ants and defeating a **Queen Ant** boss to win. The subterranean horror theme is realized through earthy vertex-colored tunnel geometry, bioluminescent mushrooms, ambient dust particles, exponential fog, and a pulsing red light from the Queen's Chamber. Both a **desktop mode** (mouse-look + WASD) and an **immersive VR mode** for WebXR headsets run from the same codebase.

Advanced features implemented:

- Full WebXR immersive-VR session with dolly rig and 6DOF tracking
- Dual-hand VR controller input (locomotion, snap-turn, attack, pickup, eat)
- Six-weapon inventory with unique 3D hand models and muzzle flash
- Boss enemy (Queen Ant) with canvas-texture health-bar sprite
- Particle system: 120 dust motes, fog, vignette, CRT scan lines

II. Scene and Interaction Description

A. Virtual Environment

The world is a single descending tunnel (≈75 world units) built from a `CatmullRomCurve3` spine with 11 control points, extruded into an organic tube with procedural vertex-color variation darkening from warm brown to near-black with depth. Four branching side tunnels extend from the main path. Decorative elements include 10 bioluminescent mushrooms (each with a point light), hanging root tendrils from sub-splines, three egg clusters near the Queen, and 60 scattered rock debris meshes. Twelve point lights in an orange-to-red gradient plus a pulsing queen-light illuminate the space; `FogExp2` (density 0.055) limits draw distance and reinforces claustrophobia. Depth zones, *Surface*, *Upper Tunnels*, *Mid Tunnels*, *Deep Colony*, and *Queen's Chamber*, are announced by a zone-flash HUD notification at each threshold.

B. Navigation

Desktop: Click to lock the pointer; WASD to translate, mouse to look, Space to jump. Gravity is applied each frame; camera Y is clamped to the procedural floor via a 50-sample spline lookup.

VR: Left thumbstick drives smooth locomotion relative to the camera forward direction; dolly Y is floor-clamped every frame. Right thumbstick snaps orientation in 30° increments to reduce motion sickness.

C. Interactions and Pickups

Eight glowing weapon orbs (spanning all six weapon types) and seven food orbs float and rotate along the tunnel. Desktop `[E]` or VR right-grip picks up the nearest orb within 2.8 units into the first free inventory slot. Desktop left-click or VR right-trigger fires the active weapon via raycasting. Desktop `[F]` or VR left-trigger consumes the carried food item (Crumb +20 HP, Berry +35 HP, Mushroom +50 HP, Honey Drop +70 HP).

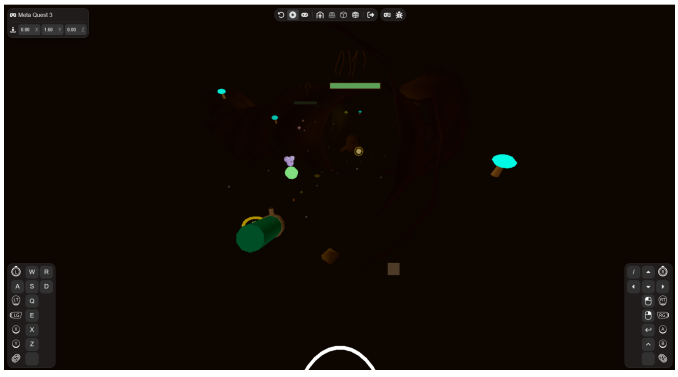

Fig. 1. Mid-colony combat showing weapon and food orbs, damage-flash overlay, and weapon hotbar (Interactions and Pickups, §II-C).

D. Enemy AI

Eight worker ants are spawned at game start, all sharing one `InstancedMesh` (pool of 40), and autonomously pursue and attack the player (6 dmg/hit, 1.2 s cooldown). Each worker has 15 HP and its own canvas-texture HP-bar sprite. Each ant's movement speed is randomised at spawn between 1.8 and 2.8 world-units per second. The population respawns toward 6 when it falls below that threshold. The Queen Ant (300 HP, 15 dmg/hit, 1.5 world-units/s, 0.7 s cooldown) activates in her chamber. All ants orient toward the player and bob sinusoidally each frame.

E. HUD and UI

A persistent overlay provides: a colour-coded HP bar, kill counter, depth label, 5-slot weapon hotbar with per-slot ammo counts, a separate food slot, a damage-flash overlay, kill-feed strip, zone-flash notifications, and a proximity pickup hint.

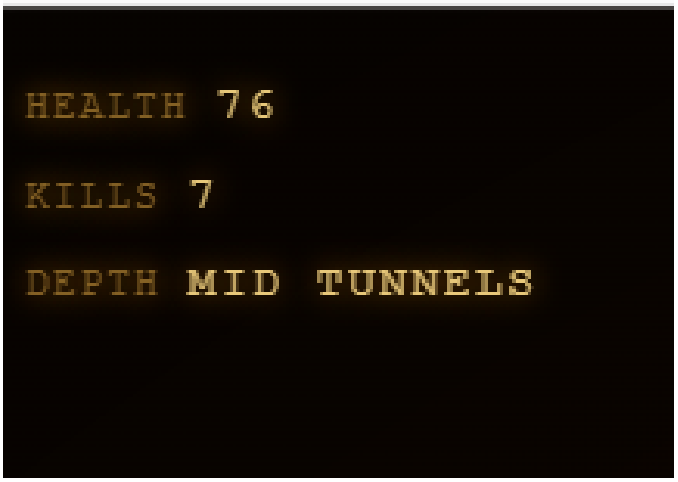

Fig. 2. HUD overlay showing health, kill count, and current depth zone (§II-E).

Screen Recording: https://drive.google.com/file/d/13-8SXgCDCzQ2$_l$eImzKM$_H$o0sd6W6DX6/view?usp = sharing

III. Features Table

Table I lists every base and advanced requirement (★ = advanced), how each is satisfied in the project, and the section of this report where it is discussed.

IV. Advanced Features Detail

A. WebXR Immersive-VR Session

An `immersive-vr` session is requested via `navigator.xr` with the `local-floor` required feature. The camera is nested inside an `xrDolly` Group so the headset drives camera pose (6DOF tracking) while the dolly handles world-space locomotion. An *Enter VR* button appears when WebXR is supported; a descriptive fallback message is shown otherwise.

TABLE I
Feature Requirements and Implementation

Feature	How the Requirement Is Met	Sec.
3D Environment	Procedural `CatmullRom` tunnel + 4 branches, mushrooms, roots, rock debris, egg clusters	II-A
1st-Person Nav	WASD + mouse-look (desktop); VR left-thumbstick smooth locomotion with floor clamping	II-B
Collectibles	8 weapon orbs (6 weapon types) + 7 food orbs; proximity pickup via `[E]` key or VR grip button	II-C
Enemy AI	8 initial worker ants (`InstancedMesh`, pool of 40) + Queen boss; chase, attack, per-ant HP bar, auto-respawn below 6	II-D
HUD / UI	HP bar, 5-slot hotbar + food slot, kill feed, zone flash, damage overlay	II-E
WebXR VR ★	`immersive-vr` session, `xrDolly` rig, 6DOF tracking, Enter VR button with graceful fallback	IV
Controllers ★	Trigger = shoot, grip = pickup, L-trigger = eat, thumbstick = move / snap-turn	IV
Inventory ★	6 weapons, 5 inventory slots, per-weapon ammo, 3D hand model (2 `.glb`, 4 procedural) + muzzle-flash effect	IV
Boss Enemy ★	Queen: 300 HP, canvas HP-bar sprite, pulsing point light, procedural model, egg chamber	IV
Particles ★	120 dust particles, `FogExp2`, CSS vignette, CRT scan lines, breathing gradient	IV

B. Dual-Hand VR Controller Input

Both controllers are attached to the dolly rig. `selectstart` (trigger) inspects `event.data.handedness`: the right hand fires the weapon (ray cast from the controller's world-space direction) and the left hand eats food. `squeezestart` (grip) on the right triggers item pickup; on the left it also eats food. Thumbstick axes drive smooth locomotion (left) and 30° snap-turning (right).

C. Six-Weapon Inventory System

A 5-slot inventory stores weapon objects with individual ammo counts. The weapons are: Magnifying Glass (∞, 18 dmg), Bug Spray (35 shots, 9 dmg), Steel Boot (∞, 32 dmg), Flamethrower (25 shots, 14 dmg), Tweezers (∞, 48 dmg), and Ant Acid (18 shots, 22 dmg). Two weapons—the Magnifying Glass and the Flamethrower,use Blender-authored `.glb` models loaded at runtime via `GLTFLoader`; the remaining four use hand models built from Three.js primitives. All weapons feature recoil animation and a muzzle-flash sphere, parented to the camera (desktop) or shooting controller (VR).

D. Boss Enemy with Health Bar

The Queen has 300 HP, a fixed movement speed of 1.5 world-units/s, and 15 dmg/hit at a 0.7 s cooldown. Her procedural 3D model features a segmented body, gold crown, translucent wings, and six legs. A canvas-texture `Sprite` renders her

health bar in world space above her head, updated on every hit. A pulsing `PointLight` (intensity oscillating between 1.3 and 3.7 at 3.5 Hz) throbs beneath her. Victory and defeat overlays appear in both desktop and VR modes.

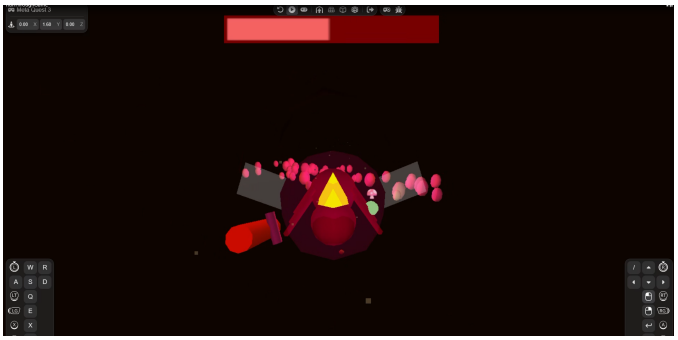

Fig. 3. Queen Ant boss encounter: procedural model, canvas health bar sprite, and pulsing point light (§IV).

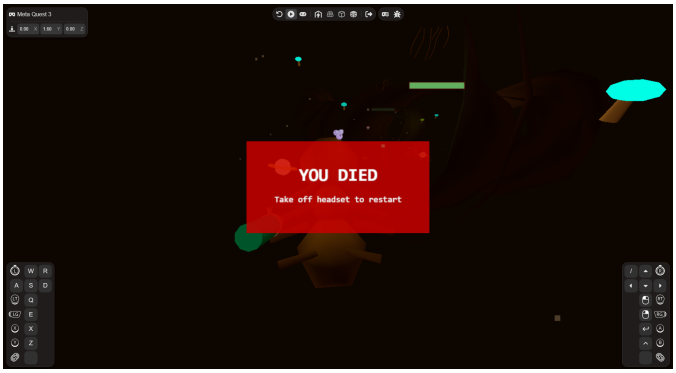

Fig. 4. Queen's Chamber — defeat overlay displayed in WebXR emulator (§IV).

E. Particle and Atmosphere System

One hundred and twenty dust particles carry randomised downward velocities and are repositioned to a random tunnel-spine point when they fall below the floor, creating a continuous falling-debris effect. A `PointsMaterial` renders them at 0.03 world-unit size. Atmosphere layers include: radial CSS vignette, 4 s sinusoidal breathing-gradient animation, repeating-linear-gradient CRT scan lines, and Three.js `FogExp2` (density 0.055).

V. Challenges and Solutions

VR Controller Handedness. The Chrome WebXR emulator occasionally dispatches `selectstart` without `event.data`, making handedness detection unreliable. A safe fallback (right = shoot, left = eat) was implemented and the limitation documented. On real hardware `event.data.handedness` is always populated.

Floor Clamping on Curved Geometry. The spline descends in 3D, so a constant floor Y is invalid. `getFloorY(x, z)` samples 50 curve points, returning the spline Y at the closest parameter. Cost is <0.2 ms/frame, keeping the player and all ants visually grounded regardless of tunnel curvature.

Instanced Mesh Raycasting. Three.js does not natively support per-instance raycasting. A parallel array of ant world-space positions is maintained; the attack ray is tested against a `THREE.Sphere` (r=1.8) around each ant, a fast, sufficiently accurate approximation for this gameplay context.

VR Weapon Re-Parenting. In desktop mode `handGroup` is a child of the camera; in VR it must re-parent to the shooting controller to track hand motion. A 150 ms `setTimeout` inside the `sessionstart` listener avoids the race condition in which the controller world-transform is not yet initialised.

TABLE II
Individual Contributions by Component

Component	Lead	Split (A / B)
Tunnel geometry & décor	Eileen	70% / 30%
Lighting & atmosphere	Siddartha	40% / 60%
Enemy AI (workers + Queen)	Eileen	80% / 20%
Weapon & inventory system	Siddartha	30% / 70%
WebXR / VR controllers	Eileen	75% / 25%
HUD / UI overlays	Eileen	35% / 65%
Food system & pickups	Siddartha	40% / 60%
Integration & testing	Both	50% / 50%

VI. Discussion and Future Work

What Works Well. The shared game loop means desktop and VR are behaviourally identical, requiring no mode-specific game logic. Instanced rendering caps draw calls at $\approx$3 for all ants combined, yielding stable 72+ fps on a Meta Quest via Air Link. Procedural geometry produces a cohesive visual aesthetic with minimal external asset downloads (only two `.glb` weapon models).

Limitations. The absence of spatial audio noticeably reduces immersion. Floor clamping can fail at sharp Y-bends in the spline. Weapon balance favours desktop play, the Tweezers' 2-unit range is nearly unusable in VR. There is no save state; death requires a full page refresh.

Future Work. Planned additions include: spatial audio (footsteps, weapon fire, Queen screech); physics-based tunnel collision; WebXR Hand Input API support for controller-free interaction; a mini-map rendered to an off-screen canvas; and a second distinct boss encounter (a ceiling-dwelling spider) to diversify gameplay pacing.

VII. Individual Contribution Table

Table II lists each project component, the team member who led it, and the approximate contribution split. Both members collaborated on all areas, but leads are identified as required.

VIII. Asset Credits

Most geometry,tunnels, ants, Queen, weapons, mushrooms, eggs,is generated procedurally at runtime from Three.js

primitives. Two weapon models (Magnifying Glass and Flamethrower) were authored in Blender and loaded as `.glb` files. No external textures, images, or audio files are used beyond the libraries listed below.

- **Three.js r128** (MIT License): https://cdnjs.cloudflare.com/ajax/libs/three.js/r128/three.min.js
- **Three.js GLTFLoader** (MIT License): https://cdn.jsdelivr.net/npm/three@0.128.0/examples/js/loaders/GLTFLoader.js
- **WebXR Device API**: W3C Specification, implemented by browser vendors.

IX. Code and Execution Instructions

A. Desktop Mode

1) Place `index3.html`, `magnifying_glass.glb`, and `flamethrower.glb` in the same local folder.
2) Start a local HTTP server: use the VS Code *Live Server* extension or run `python3 -m http.server 8080`.
3) Navigate to `http://localhost:`*`PORT`*`/index3.html`.
4) Click the canvas to acquire pointer lock.
   Controls: `WASD` move | mouse look | click attack | `[E]` pickup | `[F]` eat | Space jump | `[1–5]` switch weapon.

B. VR Mode (Meta Quest / Any WebXR Headset)

1) Ensure the local server is reachable on your LAN; note the host IP.
2) On the headset, open Meta Browser and go to `http://`*`HOST_IP`*`:`*`PORT`*`/index3.html`.
3) Tap **ENTER VR** (lower-right corner).
   Controls: left thumbstick move | right thumbstick snap-turn | right trigger attack | right grip pickup | left trigger/grip eat.

C. Browser Emulator (No Headset)

Install the *WebXR API Emulator* Chrome extension, open DevTools → WebXR tab, choose a headset preset, and click **ENTER VR**. Note: controller handedness detection may be unreliable in the emulator; validate full VR behaviour on real hardware.