

Orbital Guardian

A zero-gravity exploration and station-defense prototype

Matt Topham

Demo Day: September 8, 2026

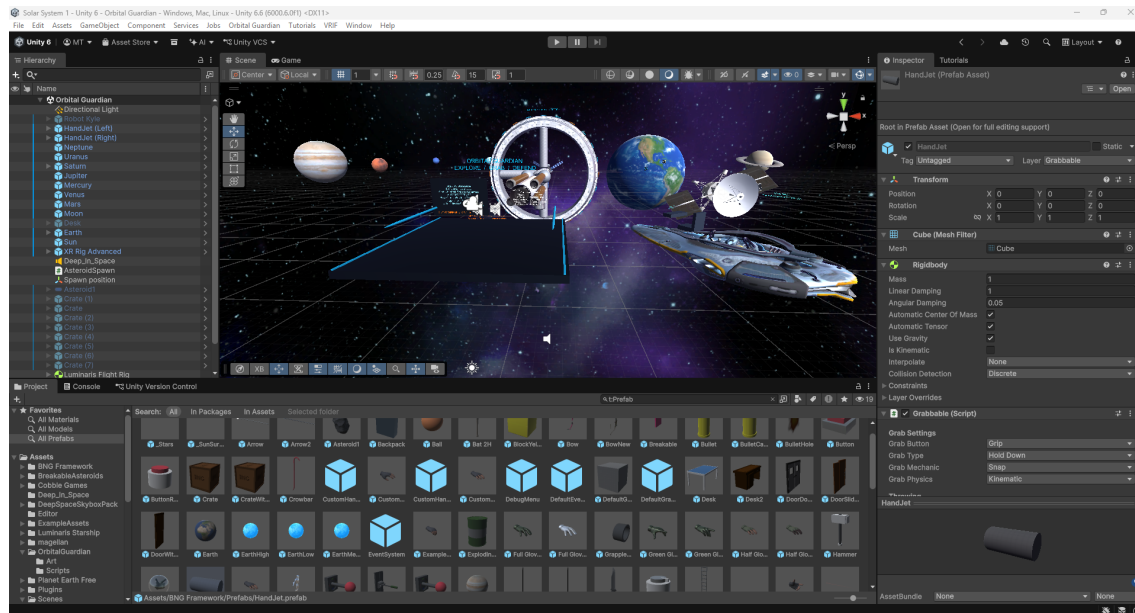


Figure 1. The integrated Orbital Guardian scene in Unity 6: the station, planetary environment, launch dock, hand tools and spacecraft. Screenshot supplied after the Demo Day walkthrough.

Project overview

Orbital Guardian is an interactive Windows / PC VR prototype built in Unity 6000.6.0f1. The player leaves a launch dock, moves using momentum and handjets, grabs the rotating station, defends it against asteroids, and approaches planets to discover small colony landmarks.

I began these space and VR experiments around 2020-2021, first with a station prefab and a simple rotation script. I later assembled a larger planetary scene with Bearded Ninja Games (BNG) VR Interaction Framework, hand tools and destructible objects. For this assignment, I returned to those older projects and used Codex to repair and combine their existing pieces into a station-defense game.

Author: Matt Topham. Presented September 8, 2026 through Chrome Remote Desktop: earlier scenes and code, their evolution, then a recorded demo with live explanation. This revised report and a reconstructed walkthrough PDF document that presentation. The accompanying MP4 is the original 6:12 recording.

Motivation and development history

From a rotating prefab to an interactive place

My starting point was curiosity about building a space environment in Unity. A station prefab and a few simple scripts made the scene move. Adding a VR rig, more planets and interactive prefabs gave me a way to explore the scene and experiment with objects using my hands.

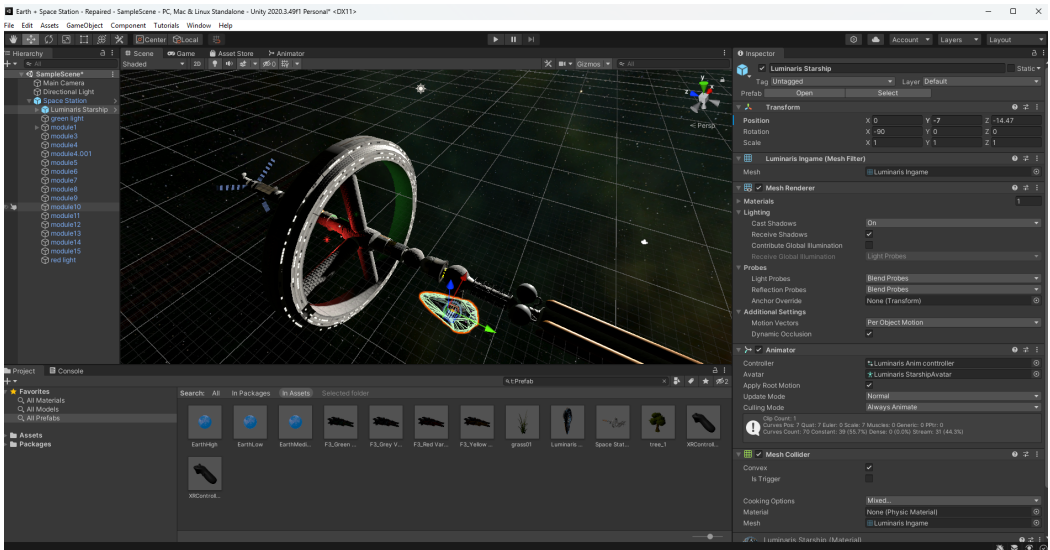


Figure 2. The earlier Earth + Space Station project, shown in Unity 2020.3.49f1. The station and spacecraft assets existed before the combined defense game.

Project chronology

Period / stage	What I built or changed
Around 2020-2021	Started with a space-station prefab, a simple scene, skybox and rotation script.
Later experiments	Added BNG VR Interaction Framework, its player rig and hand tools, more planets, and additional prefabs in the solar-system project.
Destructible objects	Experimented with breakable boxes, then searched the Unity Asset Store for breakable asteroids. Added impact-driven fracture and respawn, though the setup remained clunky.
Time away	Put the project on the back burner while busy with other work.
This assignment, 2026	Asked Codex to repair the broken project files, then combine the older projects and interactions into a station-defense game.

The approximate start date and sequence above come from my presentation account. The supplied screenshots show the older projects opened for review; their capture dates do not establish when those projects were first created.

Earlier code and interaction experiments

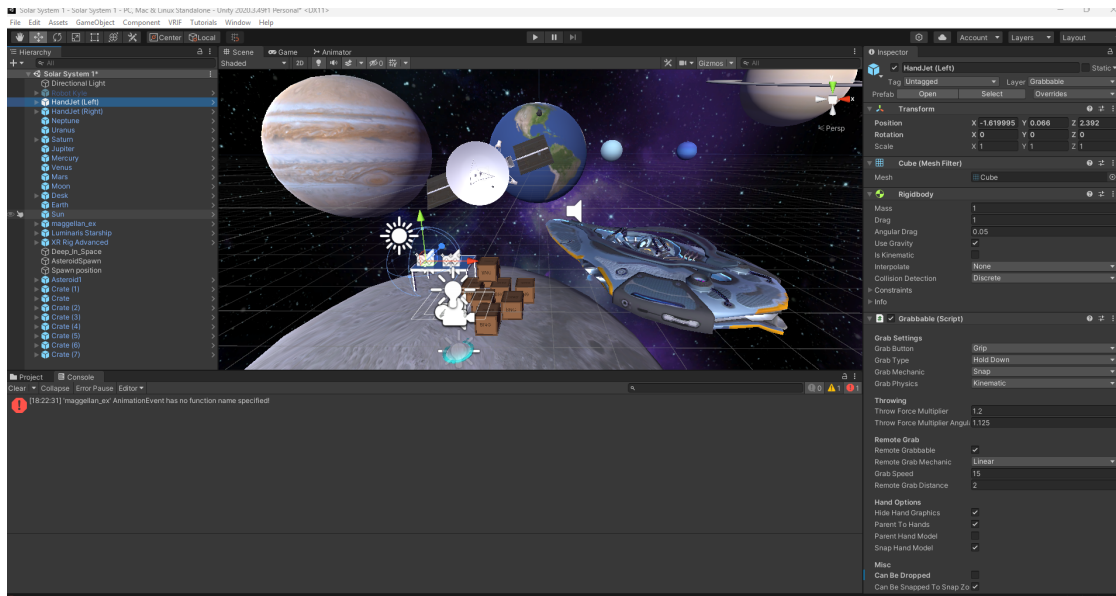


Figure 3. The older Solar System 1 scene already contained planets, BNG handjets and the XR rig, spacecraft, crates, and asteroid/spawn objects. The Console also shows an animation-event problem later addressed during repair.

A small script established motion

The original Rotate.cs shown during the walkthrough applies a local Z rotation each frame. Multiplying by Time.deltaTime makes the intended rotation rate 10 degrees per second. The current station preserves that visual rate through a kinematic Rigidbody so station motion can cooperate with grabbing.

```
void Update() {
    transform.Rotate(
        new Vector3(0f, 0f, 10f) * Time.deltaTime);
}
```

Code transcribed from the supplied Rotate.cs screenshot; the complete screenshot is included in the walkthrough PDF.

Shooting, fracture and respawn were already present

I had already been experimenting with breakable boxes when I looked for breakable asteroids on the Unity Asset Store. I added those assets to the scene and got them to break apart on raycast impact and respawn. Those interactions worked in pieces, but the scene did not yet provide a coherent station-defense objective.

BNG supplied the hand/rig, grabbing and weapon foundations. The code screenshot shown during the presentation is Grabbable.cs: it exposes held-object state, including BeingHeld and BeingHeldWithTwoHands. A RaycastWeapon.cs tab is visible, but that screenshot displays the grabbing code rather than the weapon implementation.

Repairing and combining the projects

The assignment gave the older experiments a goal

When this assignment arrived, I asked Codex whether it could repair the broken files in the older Unity projects. After the repairs, I used it to help combine the station project with the solar-system and VR interaction project. Much of the environment and many interactions were already there; the new work connected them into an encounter with a beginning, an objective and an outcome.

The resulting loop is to equip handjets, move around the rotating station, intercept incoming asteroids, survive the timed defense, and continue exploring. The combined scene also makes an existing spacecraft pilotable and supplies recovery controls for returning to the dock.

Contribution and dependency boundaries

Source of work	Role in the final project
My earlier project work	Selected and assembled assets; developed the station/planet scenes; added rotation, VR interaction and destructible-object experiments; tested shooting, fracture and respawn.
Existing asset packages	Provided the station, spacecraft, planets and asteroid art, plus BNG grabbing, hand/rig and weapon foundations. These are credited dependencies.
My assignment direction	Revived the project, described the repair and gameplay goals, chose and reviewed changes, tested the experience, and presented the development story and demo.
Codex-assisted repair and integration	Repaired and combined the projects; added the defense objective, HUD, timer, shield and scoring; coordinated EVA, station grabs, tools and ship piloting; added recovery and planet-approach behavior; helped investigate regressions.

Concrete integration problems

Movement ownership: free flight, a moving station handhold and a pilot seat must not independently move the same player. The integration suspends and restores the appropriate movement systems when the mode changes.

Weapon ownership: a projectile or ejected case could collide with its firing rig or held tool. Owner-pair filtering and inherited velocity repaired the reproduced flight jolt while retaining target collisions.

Animation and scene repair: preserve planet and station rotation; remove the empty Magellan animation event; and offset its original local animation using a parent so the route begins at the selected placement.

The implementation sections that follow describe the final integrated system. They should not be read as a claim that every asset or interaction was created from scratch during this assignment.

Inspiration and related experiences

Lone Echo and Lone Echo II are direct inspirations for this project. My development foundations were Unity Asset Store examples and packages, especially BNG VR Interaction Framework and the breakable-box and asteroid assets. Other space experiences provide useful comparisons.

Lone Echo and Lone Echo II [8, 9]

Ready At Dawn's Lone Echo places the player in a detailed space-station environment near Saturn, using hands-on interaction with tools and equipment. Lone Echo II continues the experience with zero-gravity controls and additional tools and abilities. They are relevant inspirations for moving through space with the hands, interacting with station surfaces, and making a virtual environment feel usable rather than simply decorative.

Orbital Guardian explores that interest through handjets, grabbable station surfaces and a compact defense encounter. Its asteroid objective, artistic planetary layout and recovery controls serve this smaller prototype. The Lone Echo games are design references; their assets and code are not dependencies of this project.

BNG VR Interaction Framework [4]

BNG is a direct dependency already integrated into my older project. Its rig, Grabber, Grabbable, Climbable and weapon components supplied the interaction foundation. The assignment work coordinates those systems and adds mission rules, equipment recovery, scaling and shot ownership handling.

NASA Eyes on the Solar System [1]

NASA Eyes provides information-rich 3D exploration of planets, moons and spacecraft. It is a useful comparison for a future educational layer. Orbital Guardian currently uses artistic sizes and distances, without ephemeris data.

Titans of Space PLUS [2]

DrashVR uses a narrated VR tour and celestial-size comparisons. It offers a comparison for communicating scale. Orbital Guardian instead changes the complete player rig during approach, as an exploratory gameplay mechanic.

Mission:ISS [3]

Magnopus presents zero-gravity navigation, docking and spacewalking in an ISS recreation informed by NASA models and astronaut accounts. It is a relevant station-interaction comparison; Orbital Guardian uses a fictional rotating station and asteroid encounter.

Contribution of this prototype

The contribution is bringing earlier scene and interaction experiments together into a repeatable game loop. The technical work coordinates free EVA motion, moving handholds, recoverable tools, ship travel, a bounded mission and scale transitions. Individual mechanics have precedents; their integration is the focus.

Reproduction: environment and scene

1. Open the existing project

Use Unity **6000.6.0f1** on Windows with Windows Build Support. Open **Solar System 1 - Unity 6**, let asset import finish, then select **Orbital Guardian > Open Demo**. The entry scene is **Assets/Scenes/Orbital Guardian.unity**. It is the enabled build scene. The original **Assets/Scenes/Solar System 1.unity** remains available.

Retain the project's Built-in render pipeline, package manifest, both input backends, and existing Oculus/OpenXR provider configuration. Use the installed BNG asset version and its metadata. Do not substitute URP materials or recreate .meta files when copying assets. A licensed copy of the original third-party assets is required to rebuild from a clean project.

2. Assemble and connect the main objects



Figure 4. Runtime ownership and data flow. Input feeds one EVA controller; interaction and mode rules gate movement and scaling. Ship boarding temporarily transfers movement ownership.

Object / component	Required configuration
XR Rig Advanced	Keep the outer rig, sibling body/inventory/head colliders, and inner PlayerController. Retain BNG player, CharacterController, hand Grabbers, PlayerClimbing, and snap turning.
PlayerController additions	Assign ZeroGPlayerLocomotion to the player, gravity, climbing, locomotion, capsule and PlanetApproachScale. Assign scaling RigRoot to outer XR Rig Advanced and Head to CenterEyeAnchor.
Station and dock	Place the station at (0, 10, 50) and dock at (0, 10, 0). Add a kinematic Rigidbody, StationRotation and 40 local Climbable surfaces / rails. Turn off remote grabbing for these surfaces.
Tools and console	Retain RaycastWeapon; add HandJetShotSafety to the shared HandJet prefab. Configure FaceButtonRelease and EquipmentDock on both scene instances. Assign rack sockets and the console's player, mission, arrival and tools.

The main code is in Assets/OrbitalGuardian/Scripts. Existing HandJet.cs, RaycastWeapon.cs and SpaceshipPilot.cs contain integration points. The demo does not require the temporary validation or scene-generation helpers used during development.

Reproduction: movement and tools

3. Make movement ownership explicit

ZeroGPlayerLocomotion acquires EVA movement, suspending BNG smooth locomotion, gravity, teleportation and mode switching while retaining snap turning. Stick or handjet acceleration changes a stored velocity; the CharacterController moves with that velocity each frame. With no input, momentum persists. Braking reduces it toward zero. Station climbing is applied first, and its displacement becomes bounded release momentum.

Parameter at normal scale	Value	Purpose
Maximum EVA speed	6 m/s	Limits free-flight velocity.
Handjet / stick acceleration	3 / 1.5 m/s ²	Allows deliberate buildup of motion.
Brake acceleration	8 m/s ²	Provides a quick recovery action.
Maximum push-off speed	3 m/s	Bounds release from a moving handhold.
Equipment recall delay	8 seconds	Returns an unheld tool to its dock socket.
Dock rescue hold	1.5 seconds	Both stick clicks release tools and stop drift.

4. Separate thrust, fire, release and recovery

Grip equips a handjet and fires its cannon; trigger applies thrust. Y releases the left tool and B releases the right. FaceButtonRelease defers detachment until BNG has finished iterating input callbacks, so releasing does not invalidate that iteration. EquipmentDock recalls only unheld tools. Rescue also cancels remote pulls, prevents immediate regrip, and brakes the player.

5. Preserve shot motion without self-collision

Physical projectiles and ejected cases previously contacted the player or held tool during flight. The shared HandJet prefab now opts into a WeaponShotSafety companion. At firing time, it captures the owning rig, held tool colliders, and world velocity. Each shot/case ignores those owner pairs while keeping world and target collisions. Initial world velocity equals **owner velocity + weapon-relative shot velocity**.

The delayed casing coroutine consumes the firing-time ownership snapshot even when a tool is released before ejection. Optional ray queries skip owner colliders and select the nearest valid world hit. This repair preserves the authored physical-projectile mode and the two scene shot speeds of 100 and 1,000 m/s.

SpaceshipPilot suspends EVA ownership before boarding and restores it on exit. Scaling is also blocked while piloting. These transitions are necessary: independently enabled movement components can otherwise move the same rig in incompatible ways.

Reproduction: mission and scale

6. Configure the defense loop

The new `OrbitalDefenseMission` wraps the earlier asteroid experiment with a bounded objective. It references the station anchor, existing breakable asteroid prefabs, and terminal text. Disable the old automatic spawner only in the demo scene. Mission instances preserve native damage / fracture behavior while disabling automatic respawn. `StationDefenseState` owns the timer, shield and score; each `DefenseAsteroid` can finish only once, preventing duplicate impact or intercept rewards.

Defense setting	Configured value
Round and waves	120 seconds; 3 waves
Shield / station impact / intercept	100 shield; 10 damage; 100 score
Spawn shell / station impact radius	85 m / 18 m
Initial incoming speed / active cap	3.2 m/s / 8 active asteroids
Spawn distribution	Golden-angle bearings, 20 m minimum separation; no catch-up burst

The console is usable within 5 m: A starts defense and X returns to exploration and recalls unheld equipment. A head-relative HUD shows shield, time, score, and nearest-threat direction. States are Exploration, Defending, Victory and Failed; retry and cleanup are transitions.

7. Configure planetary approach

For each planet, record its center, collision radius, start altitude, full-shrink altitude and minimum scale. Surface altitude is distance from the head to the center minus the radius. Blend toward the selected scale with logarithmic interpolation, then translate the outer rig to preserve the head's world position. Earth uses radius 37.5 m, start altitude 45 m, full-shrink altitude 1.2 m and minimum scale 0.06.

`PlanetApproachScale` runs after climbing and EVA movement. It scales the complete outer rig, adapts movement and camera near clipping, and pauses while holding, remote grabbing, climbing or piloting. After interaction it waits 0.35 seconds. Capsule clearance checks prevent growth through solid geometry. The logarithmic transition rate is 0.7 per second.

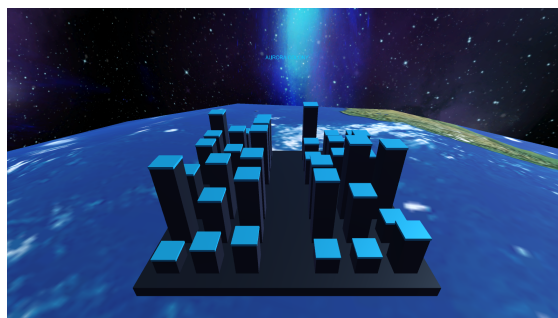


Figure 5. Separate development render of a colony scale cue. Two colony patches follow Earth's preserved rotation; the submitted gameplay recording does not visit them.

Reproduction: layout and controls

The planets are separated for visibility and manageable travel, while their distance from the unchanged Sun increases in solar-system order. The minimum gap between approach envelopes is 65.9 m. Sizes and distances are artistic; this is not an astronomy simulation.

Body	World position (x, y, z), metres
Sun	(-38, 0, -7367)
Mercury / Venus	(-1700, 0, -1900) / (800, 0, -800)
Earth / Moon	(90, 0, 260) / (245, 8, 185)
Mars / Jupiter	(-480, 0, 680) / (-1450, 0, 1550)
Saturn / Uranus / Neptune	(1650, 0, 2250) / (-850, 0, 3200) / (-1900, 0, 4000)

Keep all nine existing planet/Moon Animators enabled. Their rotation-only clips preserve the layout. StationRotation turns local Z at 10 degrees/second through a kinematic Rigidbody. For Magellan, move the **Magellan Flight Path** parent, not the animated child: the original 22.05-second clip then begins at the selected world position (18.39, 10.56, 12.49). The parent offset is (18.8177, 8.678472, -2.510184).

Control reference

Action	Tracked controllers	Desktop fallback
EVA translation	Left stick; right stick up/down	WASD; Space / Ctrl
Brake / dock rescue	Either stick click / both held 1.5 s	Shift / Home
Tool thrust / cannon	Trigger / grip	Full tool use requires tracked or simulated hands
Detach tool	Y left; B right	Requires tracked or simulated hands
Station handhold	Empty hand + grip; release to drift	Full hand interaction requires tracked or simulated hands
Defense / exploration	A / X near terminal	Enter / Backspace
Ship	A board; X center seat; B exit when stopped	Use configured BNG input / tracked controls

Build and run

In Unity Build Profiles select Windows x86-64 and include the Orbital Guardian scene. Build into a new folder. Keep the executable beside its Data folder and DLLs; launching the .exe alone after moving it will not work. The delivered build is PC VR / Windows, not a standalone Quest APK. Start from the dock and use rescue between demo segments.

Evidence, validation and limits

Validation ran in an isolated copy using Unity 6000.6.0f1 on September 7, 2026. Tested assets were copied back with matching hashes. The latest Windows build completed with **0 errors and 49 existing warnings**; warning count includes repeated vendor diagnostics. These automated checks used simulated actions; the supplied gameplay recording is separate session evidence.

Check	Observed result
Scene integrity and motion	No missing scripts; 40 station grab surfaces, 2 tools, 9 scale regions. Planet and station motion ran naturally; Magellan preserved its selected start after save/reload.
EVA, grabs and recovery	Actual CharacterController acceleration, coast, brake, no gravity, two-hand grab, rotating-station following, release momentum, tool release/recall and dock rescue passed.
Firing while moving	Baseline owner contacts reduced 6 m/s flight to 4.8422 m/s with 36.1928° deflection. Repaired Editor and standalone tests retained 6 m/s, zero deflection and zero owner contacts.
Defense and target damage	Real moving-asteroid projectile hit, one-time score/damage, native fracture, spawn separation, active cap, win/fail/retry and cleanup passed.
Ship and scale transitions	Board/exit restored movement ownership. Reversible 1x to 0.06x scaling, head anchoring, whole-rig scaling, near clip and held-object freeze passed.
Executable startup	The Windows player loaded without gameplay exceptions or invalid animation events in a headless startup check.

What the results establish

The checks demonstrate the exercised logic and Unity physics paths under simulated actions. The firing reproduction establishes unwanted owner collision as a cause of drift loss in that test; it does not establish that Unity 6 itself caused the problem, or that every possible headset hand-bounce issue is solved.

What remains to be measured

Verify physical controller mapping, tracking behavior, comfort during station rotation and gradual scaling, readability in a headset, and sustained frame rate. No user study, learning-gain measurement or performance benchmark was performed. Existing shader/API warnings and a rock convex-hull warning remain documented; they did not prevent the tested build and collision behavior.

During Demo Day I showed the project evolution through Chrome Remote Desktop and explained the demo while it played. The supplied 6:12 recording shows VICTORY, a Luminaris piloting HUD, EVA and return to the dock. Its audio track is silent; the in-class explanation was live.

Future work, credits and references

Next development steps

Comfort and onboarding. Run documented target-headset usability and frame-time tests. Add an in-world control walkthrough, adjustable acceleration and rotation comfort options, and clearer recovery feedback.

More meaningful exploration. Add planet information, selectable science-scale comparisons, destinations and objectives at the colonies, and an optional guided route. Separate artistic play space from any future educational scale mode.

Deeper station defense. Add asteroid telegraphing, difficulty selection, audio and haptic feedback, mission variation, and score persistence. Multiplayer would require authoritative mission state and synchronized ownership, so it is a larger extension.

Attribution and project scope

Project author: **Matt Topham**. Existing assets: BNG Framework / VR Interaction Framework, Zoe solar-system content, Planet Earth Free, Cobble Games Spaceship / Space Station, Luminaris Starship, BreakableAsteroids, and the project's skybox/audio packages. Preserve the licenses bundled with those packages. Added work includes colony geometry, layout, gameplay coordination, mission logic and HUD. I assembled the earlier projects and directed the final experience. Codex assisted with the 2026 repair, coding, integration, debugging and documentation; the original asset and framework authors retain credit for their work.

Demo and references

Online demo: [CSCI 4616 Unity Demo - thetopham](#). The accompanying MP4 is the same recording. The walkthrough PDF reconstructs the in-class screen-sharing sequence from the supplied screenshots; it was prepared afterward.

[8] Ready At Dawn / Meta. Lone Echo.

<https://www.meta.com/experiences/pcvr/lone-echo/1368187813209608/>

[9] Ready At Dawn / Meta. Lone Echo II.

<https://www.meta.com/experiences/pcvr/lone-echo-ii/1711938725528735/>

[1] NASA Science. Eyes: Eyes on the Solar System.

<https://science.nasa.gov/eyes/>

[2] DrashVR LLC. Titans of Space PLUS: VR Tour of the Solar System.

<https://drashvr.com/titans-of-space/>

[3] Magnopus. Mission: ISS: Immersive VR Experience. Product release: March 2017.

<https://www.magnopus.com/projects/mission-iss>

[4] Bearded Ninja Games. Grabbable and Grabber, VRIF Wiki.

<https://wiki.beardedninjagames.com/en/Core/Grabbable>

[4, companion] Bearded Ninja Games. Grabber, VRIF Wiki.

<https://wiki.beardedninjagames.com/en/Core/Grabber>

[5] Ryo Suzuki. Lecture 5: AR Research Showcase 3. Course slides.

<https://docs.google.com/presentation/d/1EriSc4jXkasXIZym-sEYIY5pez6GelDhNd4HLLQYmK/edit>

[6] ACM UIST 2024. Video Guide.

<https://uist.acm.org/2024/video-guide/>

[7] ACM UIST 2023. Video Guide.

<https://uist.acm.org/2023/video-guide/>

References checked September 7-8, 2026. Project history and inspirations supplied by the author.