Ocean Atlas

Current Status and Architecture Overview

A working Unity maritime simulation prototype moving toward a persistent geospatial world platform for offshore wind operations, maritime training, planning, infrastructure protection, and simulation-driven review.

Prepared as companion document for the prototype walkthrough video.

Status	Working prototype / technical concept demonstrator. Not a finished commercial product.
Current focus	Offshore Wind Operations Vertical Slice.
Engine	Unity 6 HDRP with Ocean Atlas runtime systems.
World model	SQLite-backed persistent geospatial world prototype; local Unity projection bubble.
Document purpose	Give MARIN, Aydin Uluc, Brainport, Infotron, and related contacts a concise view of what exists now and what is being built next.

1. Executive Summary

Ocean Atlas is a working maritime simulation prototype built in Unity. It already demonstrates a broad maritime scene with multiple vessels, environmental systems, world/map interaction, database-backed object spawning, and the beginnings of operational use cases around offshore wind, training, planning, and maritime security simulation.

The important current transition is architectural: Ocean Atlas is moving from scene-level simulation toward a persistent geospatial world. The world exists as database records with stable identity, latitude/longitude, prefab keys, layers, source metadata, and mutable state. Unity renders only the local active bubble around the player or selected location, and changes are saved back to the database.

This document is deliberately not a glossy brochure. It explains what works now, what is rough, what can be demonstrated, and what the architecture is becoming.

Immediate positioning

- Ocean Atlas is currently best described as a **working prototype and platform foundation**, not a polished product.
- The strongest near-term vertical slice is **offshore wind operations**: navigation to wind farm areas, turbines, substations, vessels, environment, and mission/inspection/training scenarios.
- The architecture also supports later extensions: defense/coastguard rehearsal, maritime education, port planning, proposal comparison, AIS/fleet monitoring concepts, and historical/story layers.

Document sections

2. Current Prototype Status

The current state is a functioning Unity prototype with many systems already present. The main issue is not lack of capability; it is integration polish. Most components work, but the user experience still has jagged edges: switching between vessels is not yet clean enough, some vehicle presentation details need polish, and several systems still feel like technical tools rather than a guided demo.

Currently working or substantially present

Area	Current status
World and scene	Latest scene contains the collected player ships and provides a base for switching or demonstrating multiple vessel roles.
Vessels	Ship, frigate, submarine/Nautilus direction, helicopter, ROV/diver concepts, and other player vehicle assets exist in the project.
Environment	Ocean, sky, weather/sea-state direction, and visual maritime environment exist. Live/synchronized weather path exists and scenario overrides are planned/used as needed.
Database world	SQLite OceanAtlasWorld.db is active as a local prototype. Database-backed markers, turbines, offshore substations, and persistent state changes have been proven.
Local bubble	Unity projection bubble loads nearby database objects around selected/global lat-lon positions rather than relying only on static scene placement.
World map / navigation	Location selection and marker travel are working; wind farm navigation exists but needs better targeting and closer operational anchors.
AIS/fleet direction	AISNearbyScan/AISFleetStream concepts exist for nearby vessel detection/spawning. Curated/synthetic fallback remains important when external AIS streams are unreliable.
Defense/training experiments	Frigate, USV, swarm/FAB, targeting, damage, and OPUS observation concepts exist as experimental modules.

3. What Ocean Atlas Can Already Show

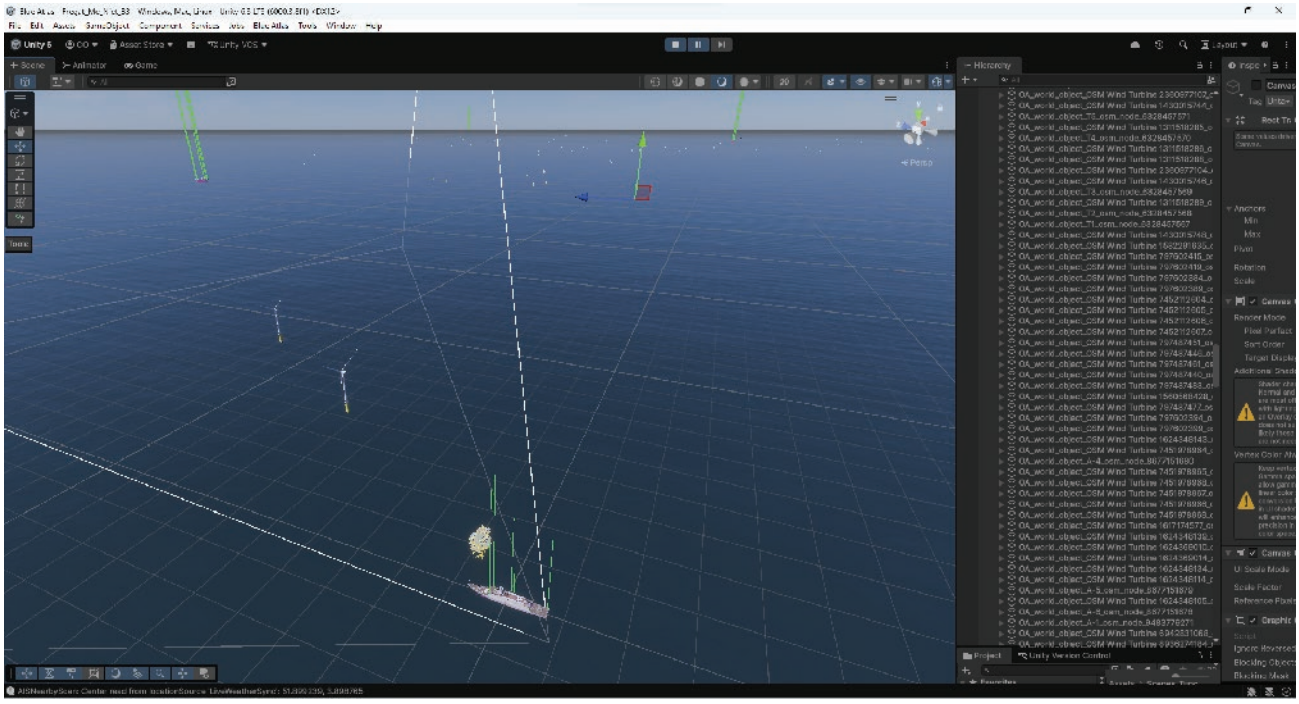
The current walkthrough can show the breadth of the platform: world-scale maritime context, vessels, environmental presentation, and database-backed maritime assets. It should be described as a prototype walkthrough because the interface and transitions are still being hardened.

Demonstrable capabilities

- **World-scale maritime context:** the project is organized around real-world-style latitude/longitude locations rather than a single isolated game level.
- **Multiple vessel roles:** different player ships and vehicles can be placed in the operational scene and used as the basis for role-specific demonstrations.
- **Offshore wind context:** wind farm groups, turbine records, and offshore substation records can exist as database-backed world objects.
- **Persistent objects:** object identity and state are saved in the database; a changed marker or asset can persist after leaving and returning to the area.
- **Environmental simulation direction:** sea, sky, weather, and operational conditions are part of the platform rather than decorative background only.
- **Training and scenario potential:** the same architecture can support inspection, service, patrol, response, defense rehearsal, and after-action review.

Prototype caveat

The current build is not yet a smooth finished product. Some transitions, UI elements, vehicle switching, vehicle motion cues, and demo flows need hardening. This is normal for the stage: the value is that the underlying world and simulation spine now exists.



4. Architecture Overview

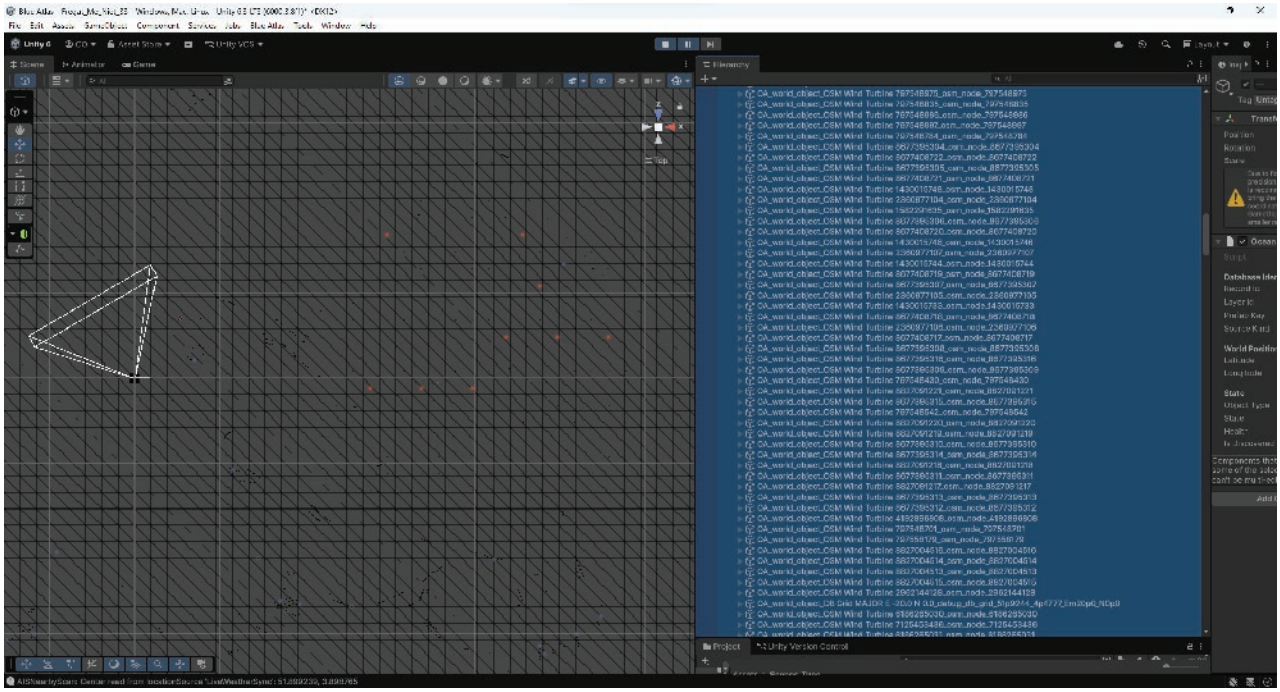
The architectural direction is to separate global truth from local presentation. The database stores what exists in the maritime world. Unity asks what should be visible near the current bubble center, spawns the appropriate prefab representations, and writes important state changes back to the database.

Core doctrine

Database truth -> local Unity projection -> simulated interaction -> persistent state update.

Main architectural components

Component	Role
OceanAtlasWorldDatabaseService	SQLite service layer. Stores world objects, prefab catalog, layers, placed instances, import cache, and mutable object state. Provides nearby-object queries.
OceanAtlasLocalBubbleController	Defines the current global center point and active/unload radius. Converts global lat-lon positions into local Unity offsets.
OceanAtlasSpawnProjectionService	Turns database records into Unity prefabs using prefab_key mapping. It should prefer Inspector-assigned catalog overrides over fragile Resources paths.
World map / location picker	Lets the user choose or travel to global positions. Should drive the bubble center and reload nearby database objects.
AIS/fleet systems	Represent live, cached, or synthetic vessel contacts as world/fleet entities that can be rendered inside the active bubble.
Scenario and mission layer	Adds operational tasks, training events, inspection missions, defense scenarios, or educational/historical layers over the base world.
OPUS observation loop	Observed Pattern Upgrade System: runtime behavior is observed, named, measured, and converted into better doctrine, debrief metrics, and future scenario logic.



5. Persistent Database and Local Bubble

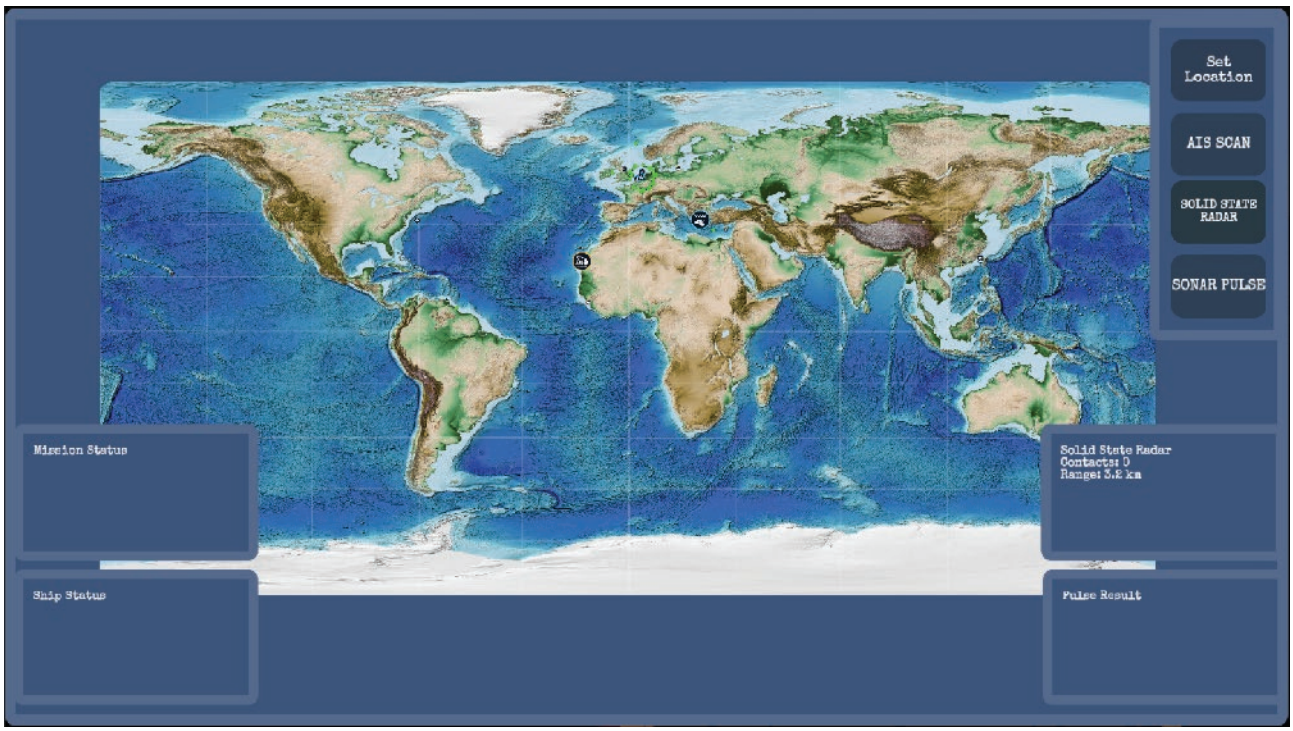
The current database prototype is one of the most important platform advances. It proves Ocean Atlas can become a persistent geospatial world rather than a collection of independent Unity scenes.

Current proven database facts

- SQLite database workflow is active through **OceanAtlasWorld.db**.
- Editor workflow can use the StreamingAssets template database directly, while runtime/build can copy to persistentDataPath as needed.
- World object records and prefab catalog records are being used to spawn scene objects from database data.
- Database-backed marker state has been proven: a marker can change prefab/state, unload, reload, and remain changed after Unity restart.
- A broad debug/global marker grid has been seeded for world-scale test coverage.
- Offshore wind and substation records can be inserted and spawned through the database pipeline.

Important table families

Table / record family	Purpose
world_layers	Separate base world, client layers, proposal layers, training layers, historical layers, or private project layers.
prefab_catalog	Maps stable prefab_key values to human-readable descriptions and spawnable asset types.
world_objects	Stable global records: object_id, lat/lon, object_type, prefab_key, group/source metadata.
world_object_state	Mutable state: health, changed prefab_key, discovered flag, state_json, inspection/mission state.
placed_prefab_instances	Explicitly placed objects or saved placements that should persist across sessions.
regional_import_cache	Cache for imported wind farms, OSM/Overpass data, or other region-based external source data.

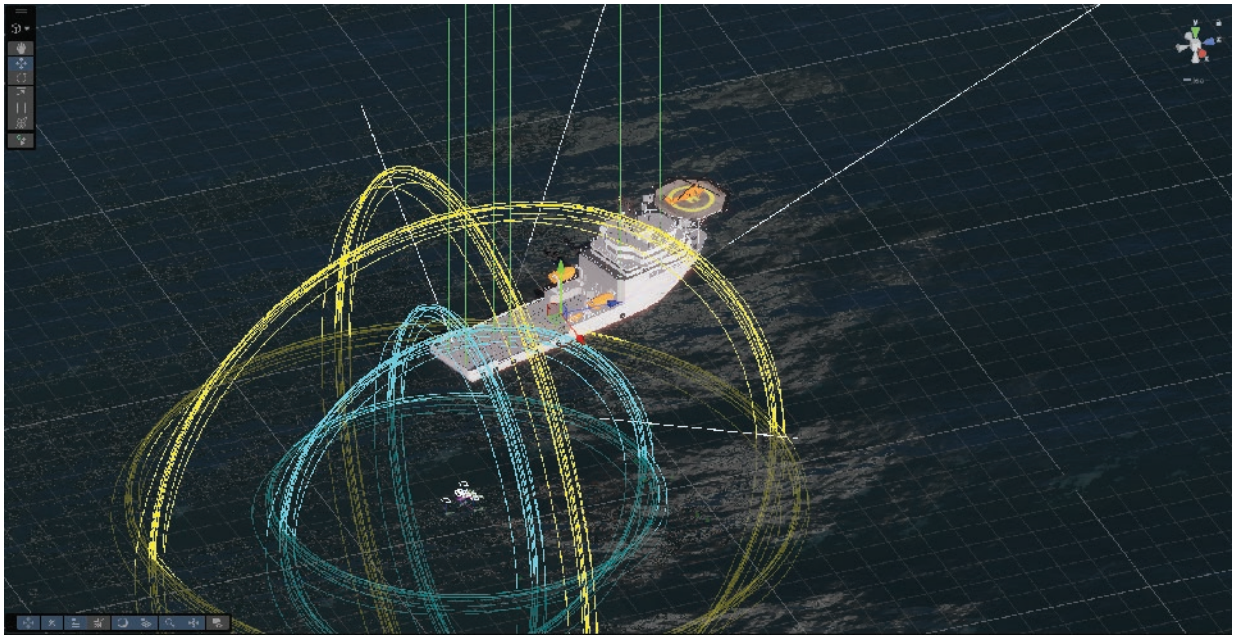


Local bubble behavior

The local bubble is the bridge between the global database and the Unity scene. Unity cannot render the whole world at once, so it renders only the active area near the player, selected location, or demo focus. When the center moves, distant objects unload and nearby database objects spawn.

Recommended bubble rules

- Use a clear active radius, for example 30 km, and a larger unload radius to avoid visible pop-in and repeated reloads.
- Keep ship/world movement separate from world truth. The ship may move locally, but global position should be derived and saved through the bubble projection service.
- Never treat Unity transform as the authoritative global state. Store lat/lon, object_id, prefab_key, and mutable state in the database.
- For future multiplayer, use the same database identity model but move the authoritative world to a server/database stack such as PostgreSQL/PostGIS behind an API.



6. AIS, Fleet, and Traffic Systems

AIS and fleet systems are a major part of the intended Ocean Atlas value: they allow the world to contain vessel traffic, contacts, patrol targets, training objects, and scenario participants. The current state should be framed carefully: AIS concepts and scripts exist, but external AIS streams can be unreliable, so the platform should support live feeds, cached data, and synthetic scenario traffic.

Current AIS/fleet direction

- **AISNearbyScan:** scans for vessels around the current area and can support nearby contact spawning or tactical overlays.
- **AISFleetStream:** intended path for streamed vessel/contact data; external stream availability should not be allowed to break demo stability.
- **Curated/synthetic fleet fallback:** essential for demos and training scenarios. When live AIS is unavailable, known vessels or generated contacts should still populate the world.
- **Radar/tactical overlays:** AIS contacts, heading, vessel identity, and range can feed radar-like UI and scenario logic.
- **Database integration:** relevant contacts should be representable as world/fleet records so they can participate in local bubble rendering and scenario replay.

AIS architecture

Layer	Role
Input adapters	Live AIS provider, cached AIS files, manual/curated vessels, synthetic scenario generator.
Normalization layer	Convert each source into common vessel/contact records: MMSI/name/type/position/heading/speed/timestamp/confidence/source.
World/contact service	Stores or serves contacts by region/time, resolves stale data, and decides what appears inside the local bubble.
Unity projection	Spawns/updates visual contacts, radar blips, labels, and scenario participants.
Scenario logic	Uses contacts for training, patrol, risk visualization, defense rehearsal, or traffic analysis.

7. Offshore Wind Operations Vertical Slice

The strongest immediate business/demo focus is offshore wind operations. This keeps Ocean Atlas grounded in a clear maritime use case while still showing the broader platform.

Why this is the right current focus

- Offshore wind requires geospatial awareness, vessels, weather, sea state, assets, inspection tasks, planning, and operational training.
- It gives the prototype a practical center: turbines, substations, service vessels, transfer operations, weather windows, inspection/repair missions, and scenario review.
- It is visually understandable for non-technical recipients: a ship goes to a wind farm, identifies assets, and performs an operational task.
- It connects naturally to ports, training institutes, engineering companies, insurers, authorities, and defense/coastguard-adjacent infrastructure protection discussions.

Current vertical slice ingredients

Ingredient	Status / next use
Wind farm groups	Group/centroid direction exists. Navigation needs better anchoring so markers take the user close to turbines or substations.
Turbine objects	Database-backed turbine spawning has been demonstrated.
Offshore substation	Database-seeded substation prefab has been proven. It should become the preferred navigation/mission anchor when present.
Service vessel / player ship	Multiple vessels are now collected in the latest scene; next step is clean switching and role-specific UI.
Weather/sea state	Can demonstrate operational conditions and later weather windows.
Mission module	Next step: simple service/inspection mission at a turbine/substation with clear start, route, objective, completion, and debrief.

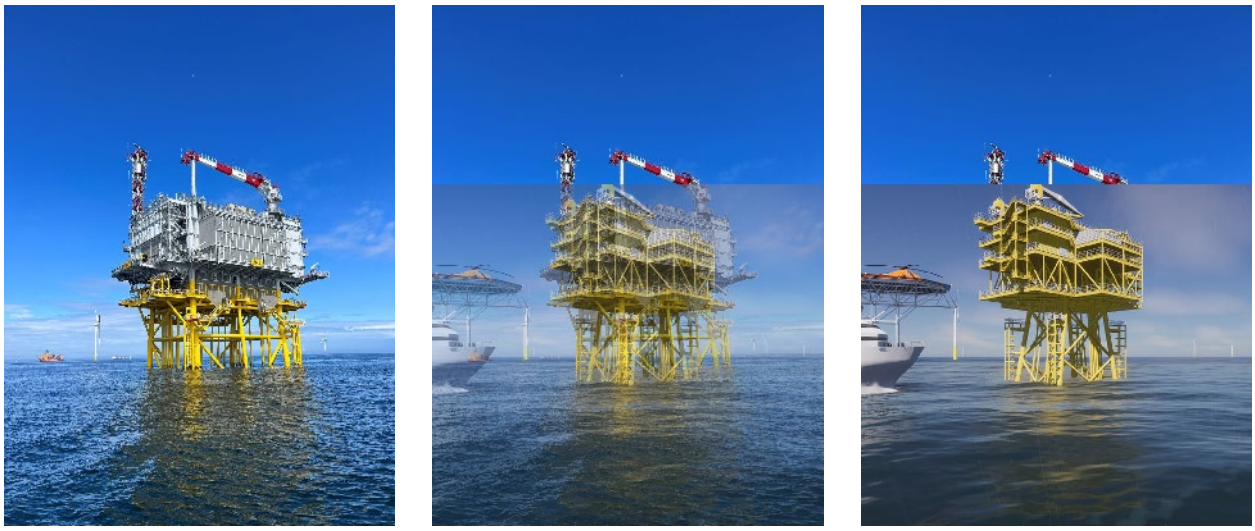


Image: Hollandse Kust Noord offshore platform.
Source: offshoreWIND.biz, “TenneT Takes Control of Hollandse Kust Noord Platform,” 1 June 2023. Image credit: Equans.

8. Limitations and Presentation Hardening

The prototype is already broad, but broad prototypes often feel rough before they feel professional. This section should be read positively: these are known hardening tasks, not unknown blockers.

Known rough edges

Rough edge	Hardening response
Ship switching is not yet clean enough	Build a central vessel selection/switching system with one active player vessel, camera handoff, input handoff, and parked/idle old vessel state.
Camera transitions can feel abrupt	Add a camera transition manager for vessel switch, map view, external view, cockpit view, and demo route movements.
Vehicle presentation details	Fix rotor/propeller animations, idle cues, audio cues, and simple movement indicators. Helicopter main rotor is a visible polish item.
Mixed UI styles	Move toward one Ocean Atlas visual language. Hide debug and old Nautilus/frigate UI when sending external demos unless deliberately explained.
Wind farm navigation not precise enough	Prefer offshore_substation or nearest turbine anchor for marker travel; load around operational targets instead of only group centroids.
External AIS uncertainty	Use curated/synthetic fallback for demos; treat live AIS as optional enhancement.
Too many beautiful directions	Keep immediate work on the Offshore Wind Operations Vertical Slice until it becomes demo-ready.

Important framing

Ocean Atlas should be presented as a working platform foundation. The purpose of the current video and PDF is to earn a serious follow-up conversation, not to claim product completion.

9. Near-Term Roadmap

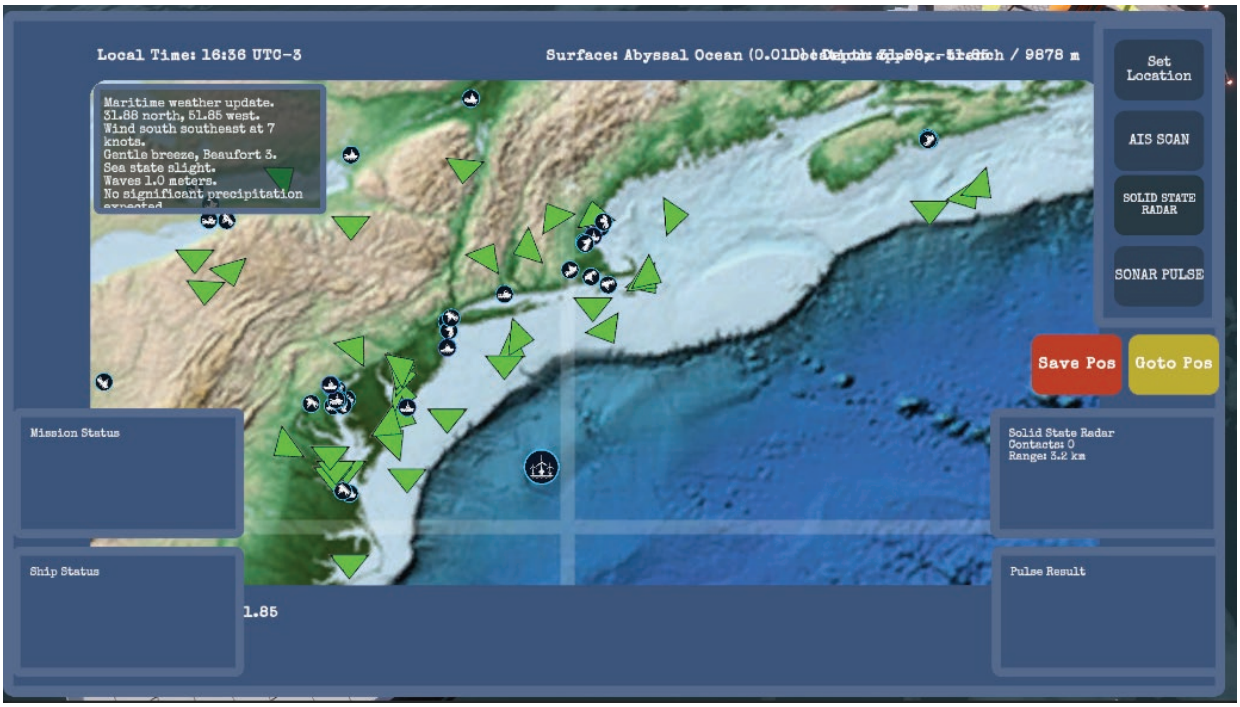
The next work should convert the current broad prototype into a reliable, repeatable, outreach-ready vertical slice.

Immediate work order

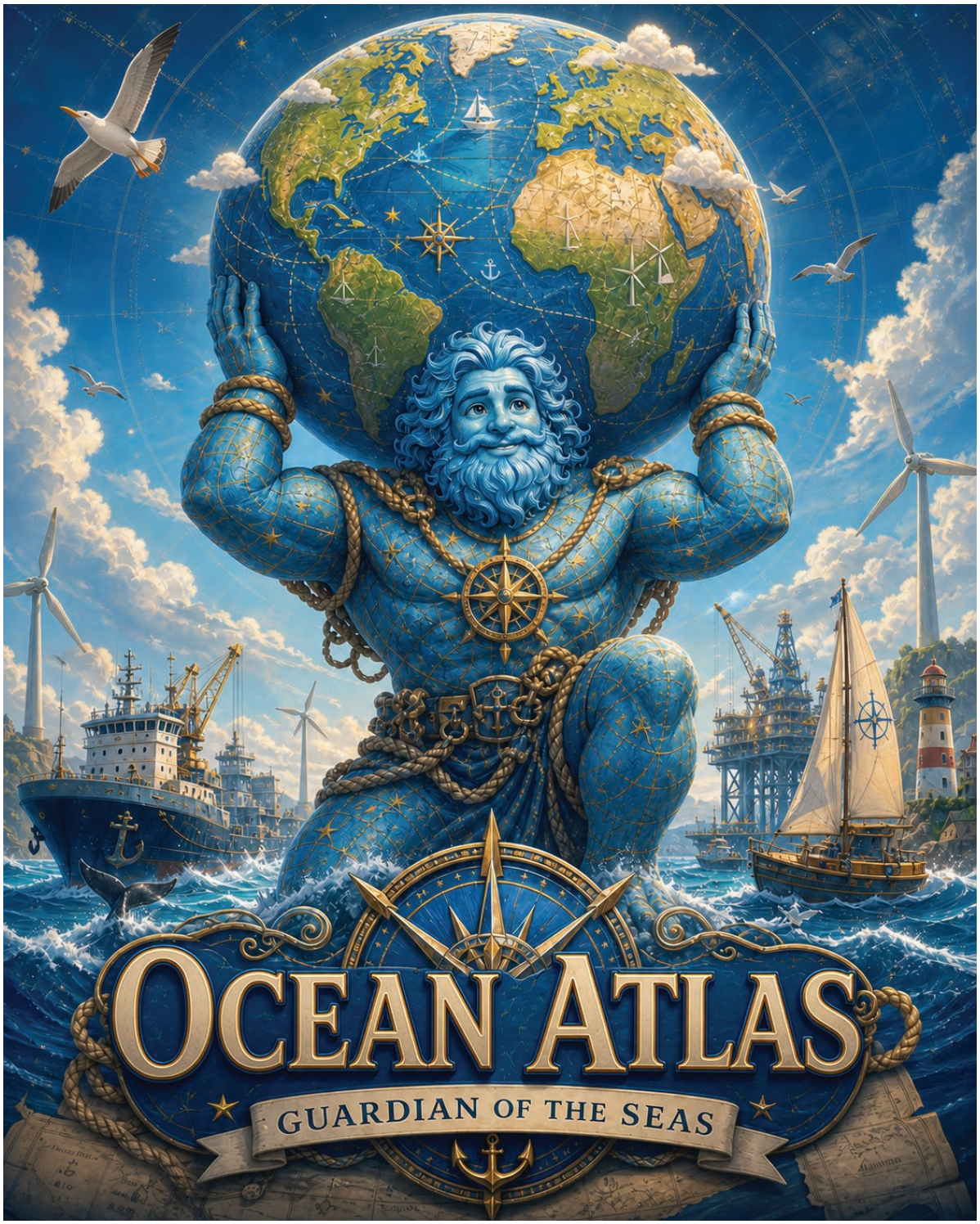
Priority	Task
1	Design and build a transparent and efficient data structure in a database able to include world objects.
2	Build/clean the ship switching system: one active vessel, camera/input handoff, clean UI, parked vessel state.
3	Improve offshore wind navigation: wind farm marker should prefer substation anchor or nearest operational asset.
4	Create guided demo route: World -> Location -> Wind Farm -> Substation -> Vessel Operation -> Weather/Sea State -> Use Cases.
5	Fix obvious presentation defects: helicopter rotor, prop/rotor cues, old UI clutter, confusing debug objects, naming inconsistencies.
6	Add first offshore wind mission module: inspect/check/repair one asset and produce a simple completion/debrief result.
7	Stabilize AIS/fleet fallback: curated contacts should always work even when external streams fail.

Next architecture phase

- Continue SQLite local prototype for fast iteration.
- Keep database schema stable around object_id, lat/lon, object_type, prefab_key, layer_id, source metadata, and mutable state_json.
- Prepare a later server path: PostgreSQL/PostGIS plus API for multiplayer, client layers, permissions, and shared-world operations.
- Design client/proposal layers so ports, offshore operators, contractors, or educators can work in the same baseline world without corrupting base data.
- Turn OPUS observations into metrics and debrief outputs for training and operational review.



Ocean Atlas is intended to help maritime stakeholders rehearse, inspect, explain, and review operations around real-world offshore assets. The platform connects geospatial data, vessels, weather, infrastructure, scenarios, and persistent state into one interactive operational environment.



Ocean Atlas is being developed by a small design team. Tunc Olcer is the project founder, concept engineer, Unity developer, maritime simulator designer, and visual/operational director of the platform.
Sol is an engineering and systems design collaborator, supporting architecture, code structure, technical planning, documentation, and decision discipline.
Mira is a critical and aesthetic design collaborator, focused on clarity, presentation force, elegance, audience perception, and strategic sharpness.
Together, the team combines practical simulation development, maritime systems thinking, software architecture, design judgment, and continuous critical review to turn Ocean Atlas from a working prototype into a focused offshore wind, maritime training, and infrastructure-protection platform.