

Game design document for Orbital Compliance

Summary (generated)

Game Title: Orbital Compliance (working title)

Logline: Manage the chaotic integration of incompatible Cold War-era technology onto a vital orbital mining rig in this strategic, text-based simulation played entirely within the terminal.

Concept Overview: Set in a near-future cyberpunk Poland where geopolitical tensions mirror a new Cold War, players take charge of systems integration aboard a crucial orbital platform. This rig processes asteroid ore, but its continuous expansion relies on components sourced from rival blocs (New World, Old World, European Bloc) who refuse common standards. Think NATO vs. Warsaw Pact tech incompatibility, but for space station parts – different power requirements, data protocols, physical connectors, and even material tolerances create constant challenges. The aesthetic and gameplay are designed for a rich game.

Player Role & Objective: The player acts as the Integration Compliance Officer. Their core responsibility is *not* direct mining, but managing the lifecycle of components: receiving shipments, meticulously verifying manifests and specifications against rig requirements and known compatibility issues, approving parts for integration, diagnosing system failures often caused by mismatched technology, and managing the fallout. The overarching goal could be meeting production quotas, surviving a contract term, or navigating specific geopolitical crises reflected in supply chain disruptions and directives from Earth.

Core Gameplay Loop: The game operates in discrete cycles ("Shifts"). Each cycle involves: 1. Receiving a status report on rig performance and alerts from the previous operational period. 2. Processing newly arrived shipments: Inspecting manifests, cross-referencing data, running simulated checks, and deciding whether to Accept & Queue for Integration, Quarantine, or Reject parts. 3. Managing currently installed systems: Scheduling maintenance, running diagnostics. 4. Strategic Requisition: Using a budget ("Logistics Points") to order necessary parts, adapters, or tools for future cycles, balancing cost, reliability, compatibility risks, and delivery times from various suppliers. 5. Ending the shift ("Coffee Break"), triggering a simulated Operational Phase where queued integrations happen and the consequences of player decisions (and random events) unfold, generating the report for the next cycle.

Key Features: Deep simulation of technical incompatibility as the central mechanic; strategic, turn-based gameplay focused on inspection, decision-making, and resource management; emergent narrative driven by system interactions, failures, and external events; minimalist yet informative. Inspired by the focused bureaucracy of *Papers, Please* and the systems management of simulation games.

Prompt 1

I like the ideas of Border Protocol and Deep Supervision the most. I think about combining those two together. In a cold war setting, operating a space station is probably going to be challenging. Was thinking about something like this:

You are in charge of a mining rig that processes ore from asteroids, which have been brought into the Earth's orbit. The rig is a big machine, so naturally it is a continuously growing machine and needs new parts. Those parts can come from many different industry partners. Because it is a cold war, the Old World and New World industrial powers have all different standards of equipment, as they refuse to yield to international standards, just like how it was in the real Cold War. NATO had its 5.56mm rifle round, while the Soviets had 5.45mm, and so on. In the game, this could mean something like Chinese drill heads not being designed to operate at the torque that the rig produces.

What do you think? it does sound like it should be easy to make a minimum viable product, but from which you can expand easily to be more complex. Bouncing some ideas would be welcome.

Answer 1

That's a *fantastic* synthesis! Combining the bureaucratic scrutiny of "Border Protocol" with the systems management of "Deep Supervision" in the context of an orbital mining rig plagued by Cold War incompatible standards creates a really strong, focused concept. Let's call it **"Orbital Compliance: Rig Integration"** (or something similar).

This hits all the right notes:

1. **Clear Role:** You are the **Integration Compliance Officer** (or similar title) for the orbital rig. Your primary job isn't mining itself, but ensuring the *parts* and *systems* needed for mining (and station survival) are correctly sourced, verified, integrated, and monitored, despite the geopolitical mess.
2. **Core Conflict:** The incompatible standards (New World vs. Old World vs. European Bloc vs. maybe Non-Aligned tech) are the cen-

tral mechanical and narrative challenge. This isn't just flavor; it directly impacts gameplay.

3. **Combines Elements:**

- **Border Protocol:** You receive manifests for incoming shipments (parts, software patches, maybe even specialist technicians). You check origins, specs, certifications against rig requirements and a database of known compatibilities/issues/regulations. You approve, quarantine, or reject shipments.
- **Deep Supervision:** You monitor the rig's overall health (power grid, structural integrity, processing efficiency, network stability) *specifically looking for anomalies caused by newly integrated or existing mismatched parts*. You diagnose failures potentially linked back to specific components or integration procedures.

4. **Scalability:**

- **MVP:** Start with a few key rig systems (e.g., Power, Life Support, Ore Processing Unit 1). Handle simple part deliveries (e.g., power conduits, filters, basic actuators) from maybe two blocs (New World vs. Old World). Basic checks: Does the voltage match? Does the physical connector type require an adapter (which adds a failure point)?
- **Expansion:**
 - **More Systems:** Add complexity – environmental controls, communication arrays, docking clamps, defense systems (this is a Cold War), advanced sensor suites, research labs.
 - **More Part Types:** Introduce complex components like CPU modules, AI subroutines, advanced drill heads, reactor components, software patches.
 - **More Factions/Sources:** Add European Bloc tech (maybe focuses on efficiency but is expensive?), Non-Aligned Movement (cheaper but less reliable?), Lunar/Martian suppliers with their own quirks.
 - **Deeper Compatibility Issues:** Move beyond simple physical/power mismatch. Introduce data protocol conflicts (requiring software shims/translators that eat CPU cycles or introduce bugs), firmware incompatibilities, differing maintenance requirements/schedules, thermal expansion mismatches causing stress, subtle material impurities leading to premature failure.
 - **Subterfuge:** Introduce counterfeit parts, deliberately mislabeled shipments, parts with hidden spyware/sabotage functions that you need to detect through deeper analysis or by observing anomalous system behavior *after* integration.

- **Integration Process:** Instead of instant integration, maybe parts go into a queue, and the *process* of fitting them takes time and resources, and can fail, requiring your intervention. You might need to schedule downtime for critical system upgrades.
- **Resource Management:** Perhaps you have a budget for adapters, diagnostic tools, or express shipping for compatible parts. Maybe rejecting too many shipments from one bloc causes political heat or supply shortages.

5. TUI Implementation:

- **Bubbletea:** Main screen could have panes for: Incoming Shipments Queue, Selected Shipment Details (manifest view), Rig Systems Status Overview, Alert Log, Command Input/Action Menu.
- **Lipgloss:** Crucial for styling. Color-code parts/shipments by origin (e.g., NW=Blue, OW=Red, EB=Green). Highlight discrepancies in manifests vs. requirements. Use status indicators (blinking red for critical system alert, yellow for warnings/compatibility issues). Style the system overview to clearly show stats like power draw, heat, stress, efficiency.
- **Glamour:** Render detailed part specifications from a database, display official regulations regarding component sourcing, show news feeds about geopolitical events affecting supply chains, format incident reports you might need to file.

Bouncing Specific Ideas/Mechanics:

- **Adapters:** Need to fit a New World high-torque drill motor onto an Old World mounting? You need the “NW-OW Heavy Torque Mount Adapter MkII”. Ordering/managing adapter inventory could be a sub-system. Adapters themselves could have failure rates or efficiency losses.
- **Software Layer:** Integrating a New World sensor package might require a “Protocol Translation Daemon” that consumes processing power from the rig’s main computer. Running too many translators slows everything down or causes crashes. Patching software becomes risky – will an Old World OS update break compatibility with New World hardware drivers?
- **Diagnostic Mini-Games:** When a system fails, you might get raw sensor logs. You’d need to run simple “commands” (`analyze power_fluctuations`, `check network_packets [source_component]`, `cross_reference failure_log [part_database]`) to pinpoint the cause, which might trace back to a specific incompatible part.
- **Technician Verification:** Sometimes shipments include specialist technicians for installation. You’d need to vet *their* credentials too (Border Protocol style). Are they certified for *this* rig’s config-

uration? Are their security clearances valid?

- **“Known Issues” Database:** As you encounter problems, you (or the system) build a database. “Warning: OW Model 7 Thrusters exhibit harmonic resonance when paired with NW Power Grid regulators above 80% capacity.” This database becomes a key tool for future decisions.
- **Emergent Narrative:** A cascade failure during a critical ore processing run because you approved a slightly-out-of-spec (but cheaper) Old World coolant pump tells a story through mechanics. A sudden directive from Earth demanding you prioritize parts from one bloc despite the technical risks creates tension.

Why this combined idea is strong:

- **Focus:** It has a very clear player role and objective.
- **Tension:** The core mechanic (incompatibility) generates constant problems and interesting decisions.
- **Worldbuilding:** It perfectly reflects the Cold War theme in a practical, gameplay-relevant way.
- **Scalability:** The complexity can grow organically by adding more systems, parts, and incompatibility rules.
- **TUI Suitability:** It's heavily information-driven and procedural, ideal for a Bubbletea/Lipgloss interface.

This feels like a really solid direction. The potential for complex interactions and emergent problems from simple rules is high, fitting the *Papers, Please* inspiration well.

Prompt 2

This is the idea I will go with in that case. However, there is still one important detail left to be addressed, which is that of how the game will progress. Making it a real-time game in the terminal would probably be awkward, and difficult not to make stressful or a chore. An idea I have in my head is making the game strictly based on cycles. Perhaps it could work like this:

1. The player receives shipments of components for the rig.
2. The player has all the time they need to do whatever is needed to inspect the equipment, and implement it or report any discrepancies to the supervising body, rejecting them.
3. The player requests further equipment for the next cycle. Perhaps they would have an allowance each cycle, effectively adding a deckbuilding dimension to the game.
4. The player then takes his “first coffee break,” concluding the first cycle. Results of what the player has done would then become

evident in the next cycle (rig reports errors resulting from faulty equipment implemented, etc.).

5. This repeats until the player's last "coffee break," before the match concludes. Not sure how the match would conclude, what that would entail, but it is probably something that can be refined over time.

Answer 2

Yes, that cycle-based approach is an excellent way to manage complexity and avoid the pitfalls of real-time stress in a TUI environment. It turns the game into a more thoughtful, strategic puzzle rather than a frantic reaction test. Let's refine this cycle structure for **"Orbital Compliance: Rig Integration"**:

The Core Cycle: "Shift Operations"

Think of each cycle as a work shift or a specific operational period (e.g., 8 hours, 1 day).

1. Shift Start / Status Report:

- The cycle begins. You receive a concise report (nicely formatted with Glamour) summarizing the rig's performance during the *previous* Operational Phase (the time between your last "coffee break" and now).
- **Content:** Key Performance Indicators (KPIs) like power stability, ore processed vs. quota, structural integrity readings, system uptime percentages.
- **Alerts:** Any new warnings or critical errors generated during operation (e.g., "Warning: Power fluctuation detected in Sector Gamma - linked to newly installed OW Regulator #78B", "Critical: Coolant leak detected near Processing Unit 2 - potential link to EB Pump Seal #42").
- **Incoming Shipments:** A list of new component/personnel shipments that have arrived at the rig's docking bay.
- **Directives:** Any new orders or policy changes from Earth HQ (e.g., "Prioritize installation of New World sensor package," "Reduce reliance on Old World processors due to suspected vulnerabilities," "Budgetary cut: Logistics Point allowance reduced by 10%").

2. Inspection & Integration Phase (Your Active Turn):

- **Process Shipments:** Select incoming shipments one by one.
 - Review the manifest (origin, specs, certifications). Use Lipgloss to highlight key data points.
 - Cross-reference against rig requirements and your "Known Issues/Compatibility Database" (a key tool

you build or is provided). Run simulated diagnostics (check_spec part_id, verify_cert part_id, query_compat part_id rig_slot).

– **Decision:**

- * **Accept & Queue for Integration:** The part seems okay (or you accept the risk). It's added to the integration queue for the *next* Operational Phase. Maybe specify *where* it should be installed if there are options.
- * **Quarantine:** Needs further analysis or requires a specific adapter/software patch you don't have yet. Takes up valuable storage space.
- * **Reject:** File a report (maybe choose from template reasons) back to supplier/HQ. This might have political/supply chain consequences later.

• **System Management:** Review the status of currently installed components, especially those flagged in alerts.

- Schedule maintenance (uses resources/time in the next Operational Phase).
- Run deeper diagnostics on suspect systems (could cost time/resources, yield more detailed info).
- Maybe initiate emergency shutdowns or reconfigurations if things look dire.

• **Requisition & Logistics Phase (The “Deckbuilding” Aspect):**

- Review rig status and upcoming needs (e.g., known wear on parts, planned expansions from HQ directives).
- You have a budget per cycle (e.g., “Logistics Points,” “Compliance Credits”).
- Browse available parts/suppliers (different suppliers offer parts from different blocs with varying costs, reliability estimates, compatibility risks, and delivery times).
- Place orders for the *next* cycle's deliveries. This is where you make strategic choices:
 - * Do you order the cheap, possibly incompatible Old World part that arrives next cycle?
 - * Or the expensive but reliable European Bloc part that takes two cycles?
 - * Or the standard New World part that uses more power but is readily available?
 - * Do you spend points on faster shipping or a better diagnostic tool?
 - * Maybe order necessary adapters?
- This isn't literal deckbuilding, but strategic resource allocation and planning for future “hands” (shipments).

3. **End Shift / “First Coffee Break” (Concluding the Cycle):**

- Finalize your decisions for this cycle (queued integrations, rejections, requisitions).
 - Confirm “End Shift.”
 - A brief summary screen might appear, perhaps with a snippet of flavor text or a news headline (“Tensions flare in Lunar negotiations,” “New cybernetic regulations proposed in European Bloc”).
4. **Operational Phase (Simulated Time between Cycles):**
- The game simulation runs based on the rig’s state *after* your decisions.
 - Queued parts are integrated (this might have an immediate success/failure chance, or just start influencing system performance).
 - The rig operates: processes ore, consumes power, components undergo stress.
 - **This is where the consequences manifest:**
 - Incompatible parts cause errors, inefficiencies, or cascade failures.
 - Well-managed systems meet quotas.
 - Scheduled maintenance is performed.
 - New wear and tear accumulates.
 - Random events might occur (minor meteoroid impact, solar flare interference, unexpected resource pocket causing surge).
 - The simulation generates the Status Report, Alerts, and potentially new Directives for the start of your *next* shift.
5. **Repeat** until a conclusion condition is met.

Concluding the Match:

- **Objective-Based:** HQ gives you a major goal (e.g., “Prepare Rig for Phase 2 Expansion within 50 Cycles,” “Achieve stable 95% operational efficiency for 10 consecutive cycles,” “Survive the Trade Embargo Period - 30 Cycles”). Succeeding wins, failing loses.
- **Score Attack:** Run for a fixed number of cycles (e.g., 100 cycles = one contract term). Your final score is based on cumulative ore processed, rig uptime, budget management, number of major incidents handled vs. caused, maybe political standing. Compete against your own high scores.
- **Survival:** The rig faces increasingly difficult challenges (more complex parts, harsher failure rates, more aggressive political interference). How many cycles can you last before catastrophic failure or being fired?
- **Narrative Endpoint:** A specific story arc concludes (e.g., you uncover a major sabotage plot, the Cold War heats up dramatically changing your supply lines, the rig is sold to another faction). Your ending might vary based on choices made.

This cycle-based structure provides a strong foundation. It allows for deep strategic thinking during the Planning Phase, while the delayed consequences in the Operational Phase create suspense and make your decisions meaningful. The “requisition” phase adds that resource management layer you wanted, forcing trade-offs that fit the Cold War theme perfectly.