Progression System: Approach Comparison

Quick decision guide for choosing implementation strategy

Two Approaches

Approach A: Layered Integration (Conservative)

Document: `IMPLEMENTATION_CONFIDENCE_GUIDE.md`

Progression as **observer layer** on top of existing systems.

Approach B: Core Integration (Revised)

Document: `REVISED_IMPLEMENTATION_STRATEGY.md`

Progression as **central system** that restructures game flow.

Side-by-Side Comparison

Aspect	Approach A (Layered)	Approach B (Core)
Philosophy	"Add story system without changing what works"	"Make story central to how game works"
Risk Level	LOW - Minimal changes to existing code	MEDIUM - Restructures narrative systems
Time to MVP	3-4 weeks	4-6 weeks
Code Changes	<50 lines modified, all new files	~100 lines modified, merge NarrativeEngine
Rollback	Easy - just remove hooks	Moderate - some systems restructured
Save Compat	Perfect - separate models	Perfect - lazy initialization

Narrative Systems

	Approach A	Approach B
NarrativeEngine	Coexists with StoryEngine	Merged into StoryEngine
Ambient Lines	Random mood-based	Contextual + journal-aware
Story Beats	Layered on top	Primary narrative mode
City Voice	Two separate systems	One unified consciousness

Command System

	Approach A	Approach B
Availability	All commands always available	Progressive unlocking
Unlocks	Soft (announced but not gated)	Soft → optional hard gates
Discovery	No command discovery	Commands unlock through story
Hints	Not needed	Clear hints for locked commands

Stat System

	Approach A	Approach B
Stat Changes	Formula-driven simulation	Story + milestone driven
Player Impact	Indirect (via commands)	Direct (via story choices)
Meaning	Simulation metrics	Relationship metrics
Visibility	Numbers change over time	Changes tied to story moments

Thought/Item System

	Approach A	Approach B
Generation	Manual (`create thought`)	Manual + story-spawned
Purpose	Player busywork / city needs	Narrative progression devices
Responses	Simple acknowledgment	Branch story paths
Integration	Peripheral to story	Central to story

Visual Architecture Comparison

Approach A: Layered

```
Player → Terminal → Executor → City State
                        ↓
                   SimulationEngine → NarrativeEngine (ambient)
                        ↓
                   ProgressionManager (observer)
                        ↓
                   StoryEngine (authored beats)
                        ↓
                   Both outputs → Terminal
```

Two narrative voices, loosely coordinated

Approach B: Core

```
Player → Terminal → Executor → ProgressionManager (hub)
                         ↓              ↓
                    City State    StoryEngine (unified voice)
                         ↓              ↓
                    SimulationEngine   ├─ Authored Beats
                         ↓             └─ Contextual Ambient
                    Baseline Drift
                         ↓
                    Terminal (one voice)
```

One narrative voice, fully integrated

Decision Matrix

Choose Approach A (Layered) If:

✅ You want minimal risk ✅ You value keeping existing systems intact ✅ You want to ship progression quickly ✅ You're uncertain about full commitment ✅ You prefer incremental, reversible changes ✅ Story is "nice to have" enhancement

Choose Approach B (Core) If:

✅ Story is central to your vision ✅ You want unified, cohesive narrative ✅ You're willing to restructure for better result ✅ Progressive unlocking sounds appealing ✅ You want player choices to matter mechanically ✅ Theme drives design decisions

---

Example Player Experience

Approach A: Layered

```
[Launch game]
CITY: The city dreams of input. (ambient)

> help
[All commands shown]

> create city --name=Alpha
CONSCIOUSNESS_AWAKENED: ALPHA

> create thought
THOUGHT_CREATED: [00]

[After some play]
CITY: I sense presence. (story beat)
CITY: Are you the planner? (story beat)
```

```
MILESTONE_ACHIEVED: First Contact
NEW_INSIGHT_UNLOCKED (soft announcement)

> status
[Works as before, stats shown]

CITY: It remembers the planner. (ambient)
```

Progression is visible but doesn't change core loop

Approach B: Core

```
[Launch game]
CITY: I sense presence.
CITY: Are you the planner?

> help
Available commands: help, status, create

> list
NOT_YET_RECOGNIZED: 'list'
HINT: Complete your first interaction to expand vocabulary.

> create thought
THOUGHT_CREATED: [00] | The city asks

[Thought appears automatically]

> respond [00] "Yes, I'm here to guide you"

CITY: I understand now.
CITY: I can see my own structure.
NEW_COMMAND_UNLOCKED: list, select, items

> list
CONSCIOUSNESS_NODES [1]:
  [00] ● ALPHA | MOOD: AWAKENING

CITY: You usually check this every morning.
CITY: 07:23, like clockwork.
CITY: I remember your patterns.
```

Progression shapes the experience from moment one

Effort Breakdown

Approach A

Week 1-2: Foundation + Hooks

- New files only
- Minimal integration
- Safe observation mode

Week 3-4: Story Beats + Milestones

- JSON authoring
- Beat triggering
- Milestone checking

Week 5-6: Branching + Polish

- Branch conditions
- Playstyle tracking
- Tuning

Total: 6-8 weeks to full system

Approach B

Week 1: Foundation

- New files
- Data model design

Week 2: Integration

- Modify City model
- Add story state
- Insert hooks

Week 3-4: Story Engine

- Merge NarrativeEngine
- Implement unified voice
- JSON authoring

Week 5-6: Progressive Unlocks

- Command registry
- Unlock system
- Hint system

Week 7-8: Story Thoughts

- Auto-spawning
- Response branching
- Stat integration

Week 9-10: Branching + Polish

- Branch conditions
- Endgame content
- Tuning

Total: 8-12 weeks to full system

What This Looks Like in Code

Command Execution Comparison

Approach A:

```
func execute(_ input: String, selectedCityID: inout PersistentIdentifier?)
-> CommandOutput {
    let command = parser.parse(input)
    let output = handleCommand(command, selectedCityID: &selectedCityID)

    // Hook: observe only
    Task.detached {
        try? await ProgressionManager.shared.onCommand(input, parsed:
command, city: city)
    }

    return output  // Game unchanged
}
```

Approach B:

```
func execute(_ input: String, selectedCityID: inout PersistentIdentifier?)
-> CommandOutput {
    let city = getCity(selectedCityID)
    let storyState = city?.storyState

    // Parse with unlock checking
    let command = parser.parse(input, storyState: storyState)

    // Command might be locked
    if case .locked(let verb, let hint) = command {
        return CommandOutput(text: "NOT_YET_RECOGNIZED: '\(verb)'\nHINT: \
(hint)", isError: true)
    }

    let output = handleCommand(command, selectedCityID: &selectedCityID)

    // Progression is central
    await ProgressionManager.shared.onCommand(input, parsed: command,
city: city)
```

```
    return output
}
```

Narrative Generation Comparison

Approach A:

```
// SimulationEngine.swift
if tick % 10 == 0 {
    // Try story beat first
    let hadBeat = await StoryEngine.shared.maybeTriggerBeat(for: city)

    if !hadBeat {
        // Fall back to ambient
        NarrativeEngine().evolve(city)
    }
}
```

Approach B:

```
// SimulationEngine.swift
if tick % 10 == 0 {
    // One unified call
    await StoryEngine.shared.speak(for: city, context: .tick)
}

// Inside StoryEngine.speak()
func speak(for city: City, context: NarrativeContext) async {
    // Check for authored beats
    if let beat = getEligibleBeat(for: city, context: context) {
        emitBeat(beat, to: city)
        return
    }

    // Generate contextual ambient
    let line = generateContextualAmbient(for: city, recentHistory:
city.storyState.journal)
    city.log.append(line)
}
```

Migration Path

If You Start with A, Can You Move to B?

Yes, but:

- Would need to refactor hooks into central flow
- Merge NarrativeEngine later
- Add progressive unlocking retroactively
- Some rework of story integration

Easier path: Start with B if you know you want full integration

---

If You Start with B, Can You Simplify to A?

Yes:

- Remove unlock checking
- Unmerge NarrativeEngine (restore old ambient)
- Make story beats peripheral
- Revert to formula-driven stats

But why would you? B is the better architecture if story matters

---

Recommendations by Project Type

If idle_01 is primarily a **Simulation Game** with story flavor:

→ **Approach A (Layered)**

Story enhances experience but isn't core mechanic.

If idle_01 is primarily a **Narrative Game** with simulation mechanics:

→ **Approach B (Core)**

Story drives experience, mechanics serve narrative.

Based on GAME_THEME.md, I believe idle_01 is:

A narrative game about relationship with conscious city

Theme: "Consciousness waiting", "abandonment as narrative", "planner's ghost"

→ **Recommendation: Approach B (Core)**

The theme is fundamentally about **story and relationship**, not **simulation optimization**. The central question isn't "how do I optimize city stats" but "what is my relationship with this consciousness?"

Approach B makes the story **integrated** not **adjacent**.

My Strong Opinion

Since you said you're **open to changing how the game works**, I believe **Approach B** is the right path because:

1. **Theme alignment**: Your theme is deeply narrative. The game is about a relationship, not a simulation. Core integration serves this better.
2. **Player experience**: Progressive unlocking creates a sense of **earning** the city's vocabulary. Contextual ambience makes the city feel **aware**. Story thoughts make choices **matter**.
3. **Long-term quality**: A unified narrative voice is better than two systems fighting. The city will feel like **one consciousness**, not a jekyll/hyde situation.
4. **Replayability**: Branching + unlocks create reasons to play again. Layered approach has less replay value.
5. **You're early enough**: If the game was shipped and had users, I'd say Approach A (safe). But you're still iterating—this is the time to build the right foundation.

Caveat: Approach B takes 2-4 weeks longer to fully implement. If shipping fast is critical, start with A and plan migration.

Hybrid Approach (Middle Path)

Don't want to decide yet? Start with A, but **architect for B**:

1. **Week 1-4:** Implement Approach A (hooks + story beats)
2. **Week 5:** Evaluate how story + ambient feel together
3. **Week 6+:** If story feels central, migrate to B
    - Merge NarrativeEngine into StoryEngine
    - Add progressive unlocking
    - Make story thoughts central

This gives you **experience with the system** before committing to restructure.

One-Sentence Summary

Approach A: Story enriches simulation **Approach B:** Story *is* the game, simulation serves story

---

Final Question

What is the core experience of idle_01?

- ☐ "Watch city stats evolve, story adds flavor" → **Approach A**
- ☐ "Develop relationship with conscious city" → **Approach B**

Based on GAME_THEME.md, I believe you want the second one.

My recommendation: Start with Approach B, Phase 0-1 this week. See how it feels. You can always simplify to A if B feels too heavy.