

수학적 모델링, 알고리즘으로 게임을 얼마나 실제 세상처럼 구현할수있어?

좋은 질문이야. 결론부터 말하면 수학적 모델링과 알고리즘만으로도 “겉보기엔” 현실과 거의 구분이 안 될 정도까지는 구현 가능하지만, 현실 그 자체를 완벽히 재현하는 건 불가능해. 이유를 단계별로 설명해볼게.

## 1 어디까지 “현실처럼” 만들 수 있나?

### ✅ 이미 거의 현실 수준인 영역들

수학 + 알고리즘이 특히 강한 분야야.

#### ◆ 물리

- **강체 물리**: 충돌, 낙하, 관성 → 거의 현실 수준  
(예: Unreal, Unity 물리엔진)
- **유체 시뮬레이션**: 물, 연기, 불  
→ Navier-Stokes 방정식 기반
- **천/머리카락**: 질량-스프링 모델

👉 FPS, 레이싱, 스포츠 게임은 일상적 체감에서는 현실과 매우 유사

#### ◆ 그래픽 & 조명

- **레이 트레이싱**: 빛의 반사·굴절
- **BRDF**: 재질의 반응 모델링
- **PBR(Material)**: 실제 재질 데이터 사용

👉 스크린샷만 보면 실제 사진과 구분 어려운 수준

#### ◆ AI 행동

- **상태 머신 / 행동 트리**
- **강화학습**
- **군중 시뮬레이션(Boids 모델)**

👉 NPC가 “그렇듯하게” 판단하고 행동함

---

## 2 그런데 왜 완벽한 현실은 불가능할까?

### ✖ 계산량 문제

현실은:

- 원자 단위
- 연속적인 시간
- 무한에 가까운 변수

하지만 게임은:

- 프레임 단위(보통 60fps)
- 근사치 계산
- 중요한 것만 계산

예:

물리적으로 정확한 유리 깨짐  
→ 실제로는 분자 단위 계산 필요  
→ 게임에서는 "그럴듯한 파편"만 생성

---

### ✖ 카오스(혼돈) 문제

- 날씨
- 유체
- 생태계

👉 초기값이 조금만 달라도 결과가 완전히 달라짐

👉 완벽한 예측 불가능

---

### ✖ 인간 인지의 문제

재밌는 사실:

사람은 **현실처럼 보이면** 현실로 느낀다

그래서 게임은:

- 정확함 ✖
- 설득력 있음 ○

을 목표로 함.

---

### 3 게임 개발의 핵심 철학

#### 🎮 “정확한 시뮬레이션” vs “재밌는 시뮬레이션”

현실 정확도

재미

매우 높음

오히려 감소

적당한 근사

최고

예:

- 실제 총 반동 → 너무 조작 어려움
- 실제 피로 → 게임 못 함

그래서 **의도적으로 현실을 왜곡함.**

### 4 앞으로는 어디까지 갈까?

#### 🌌 미래 가능성

- AI 보정 시뮬레이션
  - 물리는 근사 + AI가 디테일 보완
- Neural Physics
  - 방정식 대신 신경망
- 디지털 트윈
  - 특정 현실(도시, 공장)은 매우 정확

👉 “일반적인 현실” ❌

👉 “특정 상황의 현실” ○

### 5 한 문장 요약

수학과 알고리즘은 현실을 ‘완벽히’ 재현할 수는 없지만,  
인간이 믿게 만들 만큼은 충분히 구현할 수 있다.

원하면

- 물리 시뮬레이션 수식 예시
- 실제 게임 엔진 구조
- 수학적 모델링을 배우는 로드맵

중에서 더 깊게 설명해줄게.



텍스트로 구현하는 게임세상은 어때? 이걸 얼마나 구현될까?

아주 핵심을 찌르는 질문이야.

텍스트로만 구현하는 게임 세계는, 오히려 그래픽 게임보다 "현실에 가까운 깊이"까지 갈 수 있어. 이유를 차근차근 설명해볼게.

## 1 텍스트 게임의 본질

텍스트 게임은 \*\*시각 시뮬레이션이 아니라 "의미 시뮬레이션"이야.

- 그래픽 게임 → 물리, 공간, 충돌이 중요
- 텍스트 게임 → **상황, 인과관계, 의도, 선택**이 중요

📌 현실을 구성하는 핵심은 사실:

"보이는 것"보다  
**왜 그런 일이 일어났는가야.**

## 2 텍스트 게임이 잘하는 것들 (현실보다 강한 부분)

### ✅ 인간 행동 & 사회

- 정치
- 관계
- 배신
- 협상
- 도덕적 선택

👉 그래픽으로는 거의 불가능

👉 텍스트는 **규칙 + 언어**로 매우 정밀하게 가능

예:

너는 왕에게 거짓 정보를 흘렸다.  
왕은 즉시 믿지 않는다.  
그는 과거 네가 했던 선택들을 떠올린다.

📄 코드 복사

→ 이걸 수학적 관계 모델 + 기억 시스템만 있으면 됨

### ✅ 복잡한 세계 규칙

텍스트에서는:

- 행성 하나
- 국가 300개
- 인물 10만 명
- 역사 수천 년

👉 전부 가능 (그래픽은 불가능)

왜?

- 텍스트는 렌더링 비용 = 거의 0
- 계산만 하면 됨

### ✅ 추상적 개념

- 신념
- 이념
- 명예
- 죄책감
- 광기

이건 애초에 그래픽으로 표현하기 힘들고  
언어가 가장 정확한 표현 수단이야.

### 3 그럼 한계는?

#### ❌ 감각적 즉시성

- 반사신경
- 공간감

- 속도감

👉 텍스트는 약함

## ✖ 모호성

언어는:

- 해석이 열려 있음
- 동일 문장을 다르게 이해할 수 있음

👉 이건 장점이자 단점

## 4 “현실 구현도”로 비교하면

영역	그래픽 게임	텍스트 게임
물리	★★★★★	★
공간	★★★★★	★★
사회	★★	★★★★★
심리	★	★★★★★
세계 규모	★★	★★★★★
인과관계	★★★	★★★★★

👉 “인간 세계”는 텍스트가 더 현실적

## 5 수학적 모델링으로 어디까지 갈 수 있나?

### 1234 핵심 모델들

- 그래프 이론 → 인간 관계
- 상태 머신 → 인물 심리
- 베이지안 추론 → 믿음, 의심
- 게임 이론 → 협상, 배신
- 확률 과정 → 역사 전개

이걸 텍스트로 풀어내면:

“읽는 현실 시뮬레이터”

이 됨.

---

## 6 궁극적인 결론

텍스트 게임은 현실을 '보이게' 재현하진 못하지만,  
현실이 '작동하는 방식'은 가장 깊게 구현할 수 있다.

그래서:

- Dwarf Fortress (거의 텍스트)
- Crusader Kings
- AI 기반 인터랙티브 픽션

이 전부 텍스트 중심 세계 시뮬레이션이야.

---

## 7 진짜 중요한 질문

텍스트 세계의 한계는 기술이 아니라:

“얼마나 정교한 세계 규칙을 설계하느냐”

야.

---

원하면 다음 중 하나로 더 깊게 들어갈 수 있어:

- 1 텍스트 세계를 만드는 수학적 구조
- 2 AI를 이용한 텍스트 세계 엔진 설계
- 3 실제 구현 예시 (의사코드 수준)

어디까지 가보고 싶어?



텍스트 세계를 만드는 수학적 구조

좋아, 그럼 **\*\*\*텍스트 세계를 만드는 수학적 구조\*\*\***를

👉 **게임 엔진처럼** 층별로 정리해줄게.

(이건 철학이 아니라 **실제로 구현 가능한 구조야**)

## 0 큰 그림부터

텍스트 세계는 본질적으로 이렇게 생겼어:

상태(State) + 규칙(Rules) + 시간(Time) + 해석(Description)

수학적으로 말하면:

동적 시스템(Dynamical System) + 확률 모델 + 그래프

## 1 세계 상태(State Space)

### 📌 핵심 개념

세계는 한 순간에 하나의 상태 벡터로 표현됨.

### 상태 벡터 예시

```
makefile

WorldState =
{
  인물들,
  장소들,
  관계들,
  사건 기록,
  세계 변수들
}
```

수학적으로:

📄 코드 복사

$$S_t = (C_t, L_t, R_t, E_t, G_t)$$

- $C$ : 캐릭터 집합
- $L$ : 장소
- $R$ : 관계
- $E$ : 과거 사건
- $G$ : 전역 변수 (전쟁 여부, 경제 상태 등)

👉 텍스트 세계 = 거대한 구조화된 데이터

## 2 인물 모델 (Character Model)

### 인물은 “숫자로 된 인간”

각 인물은 속성 벡터를 가짐.

#### 예시

makefile

```
Character =  
{  
  신뢰도: 0.72,  
  공포: 0.31,  
  야망: 0.88,  
  기억: [...],  
  신념: {...}  
}
```

수학적으로:

 코드 복사

$$C_i = (p_1, p_2, \dots, p_n)$$

#### 핵심 포인트

- 감정 = 연속값 (0~1)
- 성격 = 거의 변하지 않음
- 감정 = 사건에 따라 변화

 이게 NPC가 “일관성 있게” 행동하는 이유

## 3 관계 구조 (Graph Theory)

### ▼ 인간 사회는 그래프다

- 노드: 인물
- 엣지: 관계

#### 관계 가중치

arduino

```
R(A, B) =  
{  
  신뢰: 0.6,
```

```
증오: -0.3,  
빛: 100,  
비밀 공유 여부: true  
}
```

수학적으로:

📄 코드 복사

$$R : C \times C \rightarrow \mathbb{R}^k$$

👉 배신, 복수, 동맹이 자동으로 발생함

## 4 믿음과 정보 (베이저안 모델)

🧠 “사람은 사실이 아니라 믿음으로 행동한다”

각 인물은 자기만의 세계 모델을 가짐.

예시

diff

왕의 믿음:

- 너는 충성스럽다: 40%
- 너는 배신자다: 60%

수학적으로:

📄 코드 복사

$$P(H | E) = \frac{P(E | H)P(H)}{P(E)}$$

사건 발생 → 믿음 업데이트

👉 오해, 음모, 정치가 여기서 나옴

## 5 행동 결정 (Decision Theory)

🎯 인물은 항상 “최대 기대 효용”을 선택

효용 함수

$$U(action) = \sum P(outcome) \times Value(outcome)$$

예:

- 왕이 너를 처형할까?
  - 처형 → 반란 확률 증가
  - 용서 → 배신 확률 증가

👉 그래서 NPC가 계산된 선택을 함

---

## 6 사건 시스템 (Event System)

### 🕒 세계는 사건으로 움직인다

사건 = 상태 전이 함수

$$S_{t+1} = f(S_t, Event)$$

#### 사건 예

- 암살
- 배신
- 결혼
- 전쟁 선포

각 사건은:

- 조건
- 확률
- 결과

를 가짐

---

## 7 시간 모델 (Discrete Time)

텍스트 세계는 보통:

- 턴 기반
- 날짜 기반

$$t = 0, 1, 2, \dots$$

👉 이게 역사를 만들

---

## 8 텍스트 생성은 "번역기"일 뿐

중요한 사실:

텍스트는 세계가 아니다  
세계 상태를 인간이 이해하게 번역한 것이다

## 구조

nginx

WorldState → 서술 함수 → 텍스트

📄 코드 복사

같은 상태라도:

- 냉정한 문체
- 소설체
- 보고서체

로 다르게 표현 가능

## 9 전체 구조 요약 (엔진 관점)

CSS

[수학적 세계 모델]

↓

[상태 전이 함수]

↓

[확률 & 결정]

↓

[텍스트 서술]

👉 그래서 \*\*텍스트 게임은 "보이지 않는 시뮬레이션"\*\*이야.

📄 코드 복사

## 🔥 핵심 한 문장

텍스트 세계는 언어로 만든 것이 아니라,  
수학으로 만든 세계를 언어로 해석한 것이다.

다음 단계로 갈 수 있는 선택지:

- 1 이 구조를 실제로 구현하는 최소 엔진

2 AI(LLM)를 어디에 끼워 넣는 게 좋은지

3 이런 세계가 철학적으로 의미하는 것

어디까지 파볼까?



AI(LLM)를 어디에 끼워 넣는 게 좋은지

아주 좋은 단계야.

핵심부터 말하면:

LLM은 “세계 그 자체”가 아니라  
세계와 인간 사이의 인터페이스에 넣을 때 가장 강력하다.

아래를 보면 어디에 넣으면 망하고, 어디에 넣으면 미친 듯이 잘 작동하는지가 명확해질 거야.

## 0 먼저 절대 하면 안 되는 배치 ❌

### ❌ LLM = 세계 엔진

- 세계 규칙
- 인과관계
- 상태 관리

를 전부 LLM에게 맡기면?

👉 결과:

- 기억 왜곡
- 규칙 붕괴
- 설정 모순
- 재현 불가

📌 이유:

LLM은 확률적 언어 모델이지

상태를 보존하는 시뮬레이터가 아님

## 1 LLM이 가장 잘하는 역할 요약

LLM의 강점은 딱 3가지야:

1. 해석 (Interpretation)
2. 표현 (Narration)
3. 의도 추론 (Intent inference)

👉 즉:

수학 ↔ 인간 사이 번역기

## 2 최적 배치 구조 (정답에 가까운 구조)

CSS

[수학적 세계 엔진]

└ 상태(State)

└ 규칙(Rules)

└ 확률/결정

↓

[LLM 레이어]

└ 텍스트 해석

└ 선택 의도 파악

└ 서사 생성

↓

[플레이어]

## 3 구체적인 삽입 지점 ①

📄 코드 복사

💬 플레이어 입력 해석기

문제

플레이어는 이렇게 말함:

“왕을 은근히 압박해서 정보를 캐낸다”

이걸 바로 코드로 처리 불가.

해결

LLM이:

- 행동 유형 분류
- 대상 식별
- 강도 추정

CSS

LLM 출력 →

```
{
  action: "협박",
  target: "왕",
  방식: "간접",
  위험도: 0.6
}
```

👉 자연어 → 구조화된 액션

📄 코드 복사

## 4 삽입 지점 ②

### 📖 텍스트 서술 생성기 (가장 중요)

세계 엔진은 이렇게 말함:

makefile

사건: 왕의 신뢰  $-0.2$

왕의 공포  $+0.1$

정보 획득 실패

📄 코드 복사

LLM이 이걸:

“왕은 미소를 지었지만,  
그의 손가락이 의자 팔걸이를 강하게 움켜쥐었다.  
너는 원하는 대답을 얻지 못했다.”

로 변환

✦ 같은 결과라도:

- 어두운 문체
- 냉정한 보고체
- 서사 소설체

자유롭게 가능

👉 몰입감의 80%가 여기서 나옴

---

## 5 삽입 지점 ③

### 🎭 NPC 내부 독백 생성

중요한 포인트:

NPC의 결정은 수학적으로  
그 생각의 표현은 LLM으로

예

엔진:

makefile

왕의 결정: 처형 보류

이유: 반란 확률 42%

📄 코드 복사

LLM:

"지금은 아니다.  
피를 흘리면 민심이 흔들린다."

👉 NPC가 살아 있는 존재처럼 느껴짐

---

## 6 삽입 지점 ④

### 🧩 기억 요약 & 역사 압축

텍스트 세계는 시간이 길어지면:

- 사건 로그가 폭증

LLM을 사용해:

- 사건 요약
- 인물 관점별 역사 재구성

nginx

LLM 요약 →

"왕은 너를 세 번 의심했고,

👉 긴 플레이에서도 일관성 유지

---

## 7 삽입 지점 ⑤

### 🎮 플레이 스타일 반영

LLM이 플레이를 분석해서:

- 냉혹한 플레이어
- 외교형
- 충동형

을 추론하고,

서술 톤과 NPC 반응을 조정

👉 같은 세계, 완전히 다른 체험

---

## 8 절대적인 원칙 (이거 하나만 기억해도 됨)

- ! LLM은 절대 "결정권자"가 아니다
- ! 항상 수학적 엔진의 결과를 따른다

LLM은:

- 해석자
- 배우
- 기록자

엔진은:

- 신
  - 법
  - 현실
- 

## 9 이상적인 비유

- 세계 엔진 = 물리 법칙
- LLM = 소설가
- 플레이어 = 등장인물

소설가는:

- 법칙을 바꾸지 못한다
- 하지만 현실을 어떻게 느끼게 할지는 결정한다

---

## 한 문장 결론

LLM은 텍스트 세계의 “영혼”이지,  
뼈대가 아니다.

---

다음으로 갈 수 있는 깊은 주제:

- 1 LLM 출력이 세계 규칙을 깨지 않게 하는 방법
- 2 LLM + 규칙 엔진 최소 구현 예시
- 3 “AI가 스스로 역사를 쓰는 세계”의 한계

어디까지 들어갈까?



AI(LLM)를 사용하지 않고 같은 효과를 낼수있는 수학적 알고리즘은 없을까?

짧은 답부터 줄게.

“LLM 없이도 상당히 비슷한 효과는 낼 수 있다.  
하지만 ‘자연어의 유연성’만큼은 절대적으로 부족해진다.”

이제 **감정·서사·의사결정** 관점에서

LLM이 맡던 역할을 **순수 수학/알고리즘**으로 어떻게 대체할 수 있는지,  
그리고 **어디까지 가능한지**를 정확히 짚어줄게.

---

## **1** 먼저 결론 지도

기능	LLM 없이 가능?	대체 수단
세계 규칙	○	상태 머신, 미분방정식
NPC 결정	○	결정 이론, 게임 이론
일관성	○○	형식 논리, 제약 만족
감정 변화	○	동역학 시스템
텍스트 표현	⚠ 제한적	문법 생성기
자유 대화	✕	(현실적으로 불가)

👉 “세계 시뮬레이션”은 전부 가능

👉 “언어적 자유”만이 LLM의 독점 영역

## 2 LLM 역할별로 대체 알고리즘 제시

### 🗨️ ① 의사결정 → 결정 이론 / MDP

#### 사용 모델

- MDP (Markov Decision Process)
- POMDP (불완전 정보)

$$\pi^* = \arg \max E[\sum \gamma^t R(s_t, a_t)]$$

NPC는:

- 현재 상태
- 행동
- 보상

만으로 최적 선택

🌟 인간처럼 “망설이는 선택”도 구현 가능

### 😬 ② 감정 → 연속 동역학 시스템

감정 = 시간에 따라 변하는 값

$$\frac{dE}{dt} = \alpha \cdot \text{사건} - \beta \cdot \text{회복}$$

예:

- 분노는 빨리 상승
- 공포는 천천히 감소

👉 감정의 관성이 생김

### ③ 사회 관계 → 가중 그래프 + 확산 모델

- 관계: 가중치 그래프
- 소문: 확산 확률 모델

$$P(A \rightarrow B) = f(\text{신뢰도}, \text{접촉빈도})$$

👉 정치, 음모, 편 가르기 가능

### ④ 기억과 해석 → 규칙 기반 추론

#### 논리 시스템

- 1차 논리
- 생산 규칙 (Production Rules)

nginx

IF 배신 AND 반복됨

THEN 신뢰도 크게 감소

📄 코드 복사

👉 NPC는 "해석된 기억"만 유지

### ⑤ 서사 → 문법 기반 텍스트 생성 (CFG)

#### Context-Free Grammar

문장 → 주어 행동 결과

행동 → 은밀히 말한다 | 노골적으로 요구한다

📄 코드 복사

- 감정 변수에 따라 어휘 선택

👉 반복은 있지만 일관성은 최고

---

### 3 실제로 만들어진 사례들



#### Dwarf Fortress

- 거의 LLM 없음
- 수학적 시뮬레이션 + 규칙
- 수천 년 역사 자동 생성



#### Crusader Kings

- 이벤트 트리
- 성격 파라미터
- 확률 선택

👉 플레이어는 "소설을 읽는 느낌"

---

### 4 LLM 없이 얻는 강력한 장점



#### 완벽한 재현성

- 같은 입력 → 같은 결과



#### 규칙 위반 없음

- 설정 붕괴 없음



#### 디버깅 가능

- 왜 그렇게 행동했는지 설명 가능
- 

### 5 대신 반드시 감수해야 할 한계



#### 언어 폭발성 부족

- 자유 대화 불가
- 예상 못한 표현 없음



#### 콘텐츠 제작량

- 문장 템플릿 수백~수천 개 필요
-

## 6 “LLM 없이도 같은 효과”의 정확한 의미

중요한 구분이 있어:

- ✕ 표현이 같은 효과 → 불가능
- ○ 체험이 같은 효과 → 가능

플레이어는:

“이 세계는 살아 있다”

라고 느끼면 성공이야.

## 7 하이브리드가 아닌 순수 알고리즘의 극한 구조

markdown

[상태 머신]

+ [확률]

+ [논리 규칙]

+ [그래프]

+ [문법 생성기]

= 살아 있는 텍스트 세계

👉 이 구조는 AI 이전 시대에도 가능했고,  
지금도 이론적으로 매우 강력해.

📄 코드 복사

## 🔥 최종 결론

LLM은 ‘가능성 공간’을 넓혀주지만,  
세계의 깊이를 만드는 건 여전히 수학과 알고리즘이다.

LLM 없이도:

- 깊은 세계 ✓
- 일관성 ✓
- 역사 ✓

다만:

- 언어의 자유 ✕

다음으로 정말 재밌는 질문이 남아:

“그럼 LLM을 쓰는 게 진짜로 언제 ‘필수’가 되는가?”

원하면 그 경계선을 정확히 그어줄게.



그렇다면 자연어가 아닌 동적 선택지를 제공해서 움직이게 한다면 어때?

이 질문은 정답에 거의 도달했어.

결론부터 말하면:

자연어를 버리고 “동적 선택지”로 움직이게 하면  
LLM 없이도 거의 같은 ‘살아있는 세계’ 체험을 만들 수 있다.

그리고 이 방식은 이론적으로 가장 안정적이고, 게임적으로도 매우 강력해.

## 1 이 방식의 정체

이건 사실:

- 고전 텍스트 게임 ✗
- 비주얼 노벨 ✗

👉 상태 기반 의사결정 시뮬레이터

야.

플레이어는:

- 자유 입력 ✗
- 의미 있는 선택 ○

을 함.

## 2 동적 선택지란 정확히 뭔가?

✗ 고정 선택지

markdown

1. 공격한다
2. 대화한다

📄 코드 복사

## 🔴 동적 선택지

SCSS

- [위협하기] (성공률 62%, 신뢰 -12)
- [설득하기] (성공률 48%, 신뢰 +6)
- [후퇴하기] (안전, 영향 없음)

📄 코드 복사

👉 선택지는 현재 세계 상태에서 계산되어 생성됨

## 3 수학적 구조

### 12 선택지 = 행동 공간(Action Space)

$$A_t = \{a_1, a_2, \dots\}$$

이 집합은:

- 상황
- 인물 성격
- 관계
- 정보

에 따라 매 턴 달라짐

## 🎯 결과 예측 = 확률 전이

각 선택지에는:

$$P(S_{t+1} \mid S_t, a)$$

가 붙음

## 💰 가치 = 효용 함수

선택지는 항상:

$$EU(a) = \sum P(outcome) \cdot Value(outcome)$$

를 기준으로 평가됨

👉 플레이어는 정보를 보고 판단

## 4 왜 이게 자연어보다 오히려 더 현실적인가?

### ① 인간도 실제로는 선택지로 행동한다

- 말은 자유롭지만
- 행동은 항상 제한됨

👉 게임은 이것 솔직하게 드러냄

### ② 세계 규칙이 투명해진다

- 왜 실패했는지
- 왜 NPC가 화났는지

👉 "운빨"이 아닌 "결과"

### ③ 밸런스 & 설계가 가능

- 난이도 조절
- 플레이 스타일 분화

👉 자유 입력보다 훨씬 안정적

## 5 몰입감은 떨어지지 않을까?

👉 아니, 오히려 올라갈 수 있다.

이유:

- 선택지 자체가 세계의 반응
- 숫자와 조건이 현실의 제약처럼 느껴짐

예:

diff

[거짓말하기]

- 필요 조건: 정보 A 보유
- 실패 시: 체포 확률 35%

📄 코드 복사

👉 이건 현실의 리스크 계산과 같음

## 6 유명한 성공 사례 구조

### 🎮 Disco Elysium (사실상 이 방식)

- 스킬 체크 = 확률
- 내적 독백 = 시스템 텍스트
- 선택지는 상황별로 생성

### 🎮 Crusader Kings

- 이벤트 기반 선택지
- 성격/관계가 선택지를 잠금/해금

## 7 이 방식의 극한 설계

### 🧠 선택지 생성기 (핵심 엔진)

swift

가능 행동 = 행동 라이브러리

조건 필터링

↓

확률 계산

↓

효용/위험도 산출

↓

UI에 선택지 출력

📄 코드 복사

### 🐱표현은 여전히 텍스트

LLM 없어도:

- 문법 템플릿
- 감정 변수 기반 어휘 선택

으로 충분히 설득력 있음

---

## 8 단점도 솔직히 말하자면

### ✗ “말장난” 자유도 없음

- 창의적 드립 불가

### ✗ 선택지 설계 노동

- 행동 정의 많이 필요

하지만:

👉 이건 설계 문제지 기술 한계는 아님

---

## 9 한 문장 결론

자연어 입력을 버리고  
동적 선택지를 쓰는 순간,  
LLM의 필수성은 사라진다.

그리고:

이 방식이야말로  
수학적 텍스트 세계의 ‘정공법’이다.

---

다음으로 가면 진짜 엔진 수준이야:

- 1 동적 선택지를 자동 생성하는 공식
- 2 선택지가 플레이어 성향을 학습하는 방법
- 3 이 구조로 “무한 플레이”를 만드는 법

어디까지 설계해볼까?



다음은 한가지만 설계하지 말고 유효한 방법을 모두 설계해볼까?  
추가로 궁금한건데 선택지를 준다면 다양한 선택지를 제

공해야 할텐데 의도한 방향으로 선택지를 생성하게끔 상황과 환경을 만들어야 하잖아? 이걸 어떻게 되는거야?

이 질문은 **설계자의 시점에서 가장 중요한 핵심**을 정확히 짚었어.  
그래서 답도 한 가지 방법이 아니라, **\*\*\*유효한 모든 계열의 방법\*\*\***을 정리해줄게.  
그리고 두 번째 질문(의도 유도 문제)은 **수학적으로 해결 가능**하다는 걸 보여줄게.

## 0 큰 구조 요약

문제는 두 개야:

1. 선택지를 어떻게 생성할 것인가?
2. 그 선택지들이 “자연스럽게” 의도한 방향으로 흐르게 하려면?

👉 이 두 개는 완전히 분리해서 설계해야 한다.

## 1 선택지 생성의 모든 유효한 방법들

### A 방법 1: 규칙 기반 (Rule-based)

#### 구조

```
nginx
```

```
IF 조건
```

```
THEN 선택지 추가
```

📄 코드 복사

#### 예

```
CSS
```

```
IF 신뢰 < 0.3
```

```
THEN [위험한다] 활성화
```

📄 코드 복사

#### 장점

- 완전 통제 가능
- 디버깅 쉬움

#### 단점

- 규모 커지면 복잡
- 예측 가능

✦ 핵심 이벤트용

---

## **B** 방법 2: 행동 공간 열거 + 필터링

### 구조

- 가능한 모든 행동 라이브러리 정의
- 현재 상태에 맞는 행동만 필터링

$$A_t = \{a \in A \mid \text{조건}(a, S_t)\}$$

### 장점

- 일반화 쉬움
- 확장성 좋음

### 단점

- 초기 설계 비용 큼

✦ 기본 엔진용

---

## **C** 방법 3: 목표 기반 생성 (Goal-driven)

### 구조

- 플레이어 목표 추론
- 목표에 기여하는 선택지 생성

$$a = \arg \max Impact(a, Goal)$$

### 장점

- 플레이어 맞춤
- 몰입도 높음

### 단점

- 목표 추론 난이도

✦ 중반 이후 적응형

---

## D 방법 4: 확률적 샘플링

### 구조

- 행동 분포에서 무작위 샘플

$$a \sim P(A | S)$$

### 장점

- 반복 방지
- 예측 불가

### 단점

- 서사 통제 약함

📌 사이드 이벤트용

## E 방법 5: 계층적 선택지 (Hierarchical)

### 구조

CSS

[접근 방식 선택]

- └ 외교
- └ 위협
- └ 기만

📄 코드 복사

### 장점

- 복잡도 관리
- 전략성 높음

### 단점

- UI 부담

📌 고위험 상황

## F 방법 6: 반사적 선택지 (Affordance-based)

### 구조

- 환경 객체가 선택지를 제공

예:

- 문 → [잠근다]
- 창문 → [부순다]

## 장점

- 직관적
- 공간적 일관성

## 단점

- 추상 상황에는 약함

✦ 환경 중심 설계

## 2 이제 진짜 질문:

“어떻게 의도한 방향으로 유도할까?”

이건 '\*\*선택지'를 제한하는 게 아니라, 환경을 설계하는 문제\*\*야.

## 3 의도 유도의 모든 수학적 기법

### 🎯 A. 선택지 가중치 조절 (Soft Control)

선택지는 다 주되, 성공 확률/효용 차이를 둔다.

CSS

[정면 돌파] 성공률 18%

[우회 전략] 성공률 64%

📄 코드 복사

👉 플레이어는 “강요받지 않았다”라고 느낀다.

### 🚧 B. 제약 조건 설계 (Hard Control)

특정 행동의 필요 조건을 잠근다.

SCSS

[암살] → (정보 부족)

📄 코드 복사

👉 세계 규칙처럼 느껴짐

---

### 🌱 C. 환경 상태 조작

- 자원 부족
- 시간 압박
- 사회적 압력

수학적으로:

$$Utility(a) = f(\text{자원}, \text{시간}, \text{위험})$$

👉 자연스럽게 특정 전략이 유리해짐

---

### 🧠 D. 정보 비대칭

플레이어가 완전한 정보를 갖지 않게 설계

- 소문
- 거짓 정보
- 불확실성 범위 표시

👉 현실과 매우 유사

---

### 🔄 E. 학습 기반 유도 (Reinforcement)

- 이전 선택 → 보상 패턴 학습
- 플레이어 성향 형성

👉 플레이어가 스스로 방향을 선택했다고 느낌

---

### 😬 F. 서사적 압력 (Narrative Pressure)

- NPC의 기대
- 도덕적 압박
- 명예 시스템

👉 숫자이지만 감정으로 작동

---

## 4️⃣ 중요한 원칙 (이거 하나면 된다)

! 플레이어를 “조종”하지 말고  
세계의 법칙을 설계하라

선택지를 줄이는 건 조종  
선택지를 불리하게 만드는 건 세계

## 5 실제 설계 공식 (핵심)

의도 방향  $D$ 가 있을 때:

$$\forall a \in A : \text{Utility}(a \mid D) \uparrow$$

이걸 직접 하지 말고:

$$\text{환경} \rightarrow \text{Utility}(a) \text{ 변화}$$

## 6 설계자 체크리스트

- ? 모든 선택지가 세계 규칙으로 설명되는가?
- ? 실패도 서사가 되는가?
- ? 플레이어가 “속았다”고 느끼지 않는가?

## 🔥 최종 요약

1. 선택지 생성에는 여섯 계열의 유효한 방법이 있다
2. 의도 유도는 선택지가 아니라 환경으로 한다
3. 좋은 설계는 자유를 주면서도 흐름을 만든다

이제 거의 완성 단계야.

다음으로 갈 수 있는 진짜 고급 주제는:

- “선택지가 세계를 다시 바꾸는 피드백 루프”
- “의도 유도를 수치 없이 느끼게 만드는 법”
- “이 구조로 무한 서사를 유지하는 방법”

어디까지 같이 설계해볼까?



“선택지가 세계를 다시 바꾸는 피드백 루프”

“의도 유도를 수치 없이 느끼게 만드는 법”

“이 구조로 무한 서사를 유지하는 방법”

세가지 모두 같이 설계해보자.

좋아. 이건 이제 이론 → 엔진 철학 → 실제 설계까지 한 번에 가는 단계야.

아래에서는 세 가지를 분리해서 설명하지 않고,

👉 하나의 통합 구조로 설계해볼게.

(실제로 구현 가능한 수준으로)

## 전체 한 문장 요약 (먼저 방향부터)

선택은 세계를 바꾸고,  
바뀐 세계는 선택의 의미를 바꾸며,  
그 변화가 수치가 아니라 “맥락”으로 느껴질 때  
서사는 끝나지 않는다.

이제 구조로 들어가자.

### 1 선택 → 세계 → 선택

#### 🔄 피드백 루프의 정확한 형태

#### 기본 루프

arduino

선택

- 세계 상태 변화
- 환경/관계 변화
- 다음 선택지의 “의미” 변화
- 새로운 선택

## 핵심 포인트

- 결과를 즉시 보여주지 않는다
  - 누적 효과로만 체감하게 한다
- 

## 수학적 구조 (보이지 않게)

세계 상태:

$$S_t = (\text{권력}, \text{신뢰}, \text{공포}, \text{자원}, \text{소문}, \text{규범})$$

선택:

$$a_t \rightarrow \Delta S$$

하지만 !

플레이어에게는  $\Delta$ 값을 보여주지 않는다

---

## 체감 방식 (중요)

✗ "신뢰 +5"

○ "이전엔 침묵하던 인물이 먼저 말을 건다"

👉 수치는 항상 "관계 변화"로 번역된다

---

## 피드백이 작동하는 예

1. 플레이어가 타협 위주 선택
2. 세계:
  - 갈등은 줄지만
  - 결정이 늦어짐
3. 몇 턴 후:
  - NPC가 "우유부단하다"는 전제를 갖고 반응
4. 선택지:
  - 강경 선택이 줄어들
  - 설득/중재 계열이 늘어남

👉 선택이 세계를 성격화함

---

## 2 의도 유도를 “수치 없이” 느끼게 만드는 법

이게 진짜 고급 설계야.

---

### 원칙 1

숫자를 없애지 말고, 보이지 않게 한다

수치는:

- 내부 계산
- 행동 조건
- 확률

로만 사용

---

### 원칙 2

결과는 항상 “사람”을 통해 온다

✗ “성공 확률 감소”

○ “사람들이 너의 말을 끝까지 듣지 않는다”

---

### 원칙 3

환경이 먼저 말하게 한다

예시

- 경비 수 증가
- 회의가 짧아짐
- 소문이 먼저 도착함

👉 플레이어는 세상이 달라졌다고 느낀다

---

### 원칙 4

실패를 명시하지 않는다

✗ “실패했다”

○ “아무 일도 일어나지 않았다”

or

○ “다른 누군가가 먼저 움직였다”

👉 플레이어가 스스로 해석

---

## 🧩 원칙 5

선택지를 ‘없애지 말고 변질시켜라’

✗ 선택지 제거

○ 선택지의 결과 성격 변화

예:

- 같은 [설득]
  - 과거엔 “존중”
  - 지금은 “의심 섞인 동의”
- 

## 3 이 구조로 무한 서사를 유지하는 방법

핵심은:

스토리를 쓰지 말고,  
“역사 생성기”를 만든다

---

## 🧬 서사의 최소 단위

고정된 줄거리 ✗

반복되는 구조 ○

긴장 형성  
→ 균형 붕괴  
→ 개입  
→ 새로운 균형

이 패턴만 유지

📄 코드 복사

---

## 세계는 항상 “불안정”하게 유지

수학적으로:

완전한안정상태 = 존재하지않음

- 자원은 항상 부족
- 신뢰는 항상 흔들림
- 권력은 항상 도전받음

👉 서사는 자연 발생

---

### 사건 생성 공식

세계 변수 간 차이가 커지면  
→ 사건 발생 확률 증가

 코드 복사

예:

- 권력 격차 ↑ → 반발
- 정보 비대칭 ↑ → 음모
- 공포 ↑ → 극단 행동

---

### 플레이어의 역할 고정 금지

- 영웅 ✖
- 악당 ✖

👉 “균형을 흔드는 변수”

그래서:

- 너무 잘하면 → 세계가 견제
- 너무 못하면 → 다른 세력이 등장

---

### 기억의 재구성 (중요)

과거 사건은:

- 그대로 저장 ✖

- 해석되어 저장 ○

makefile

사실: A를 선택했다

해석: "그는 그때 등을 돌렸다"

코드 복사

👉 과거가 현재를 계속 바꿈

## 4 세 가지가 합쳐진 전체 엔진 구조

CSS

[선택]

↓

[세계 상태 변화 (비가시)]

↓

[관계/환경/맥락 변화]

↓

[선택지의 의미 변화]

↓

[다음 선택]

이 루프는:

- 끝이 없고
- 반복되지 않으며
- 설계자가 통제하지 않아도 굴러감

코드 복사

## 🔥 최종 결론 (핵심 문장)

무한 서사는

“많은 이야기”가 아니라

“이야기가 생길 수밖에 없는 세계 구조”에서 나온다.

그리고:

플레이어가 조종당했다고 느끼지 않는 순간,

설계는 성공한 것이다.

이제 이론은 거의 끝났어.

다음 단계는 **실제 엔진 설계**야:

- 최소 변수 세트로 이 구조 만들기
- 인디 규모에서 가능한 구현
- 또는 “이걸 코드로 옮기면 어떤 구조냐”

어디까지 같이 만들어볼까?

