



말보다모달

**GIFPT**

김예나 왕대현 우태호 조예린

00

# What is

G I F P T ?

문제 정의

유사 서비스

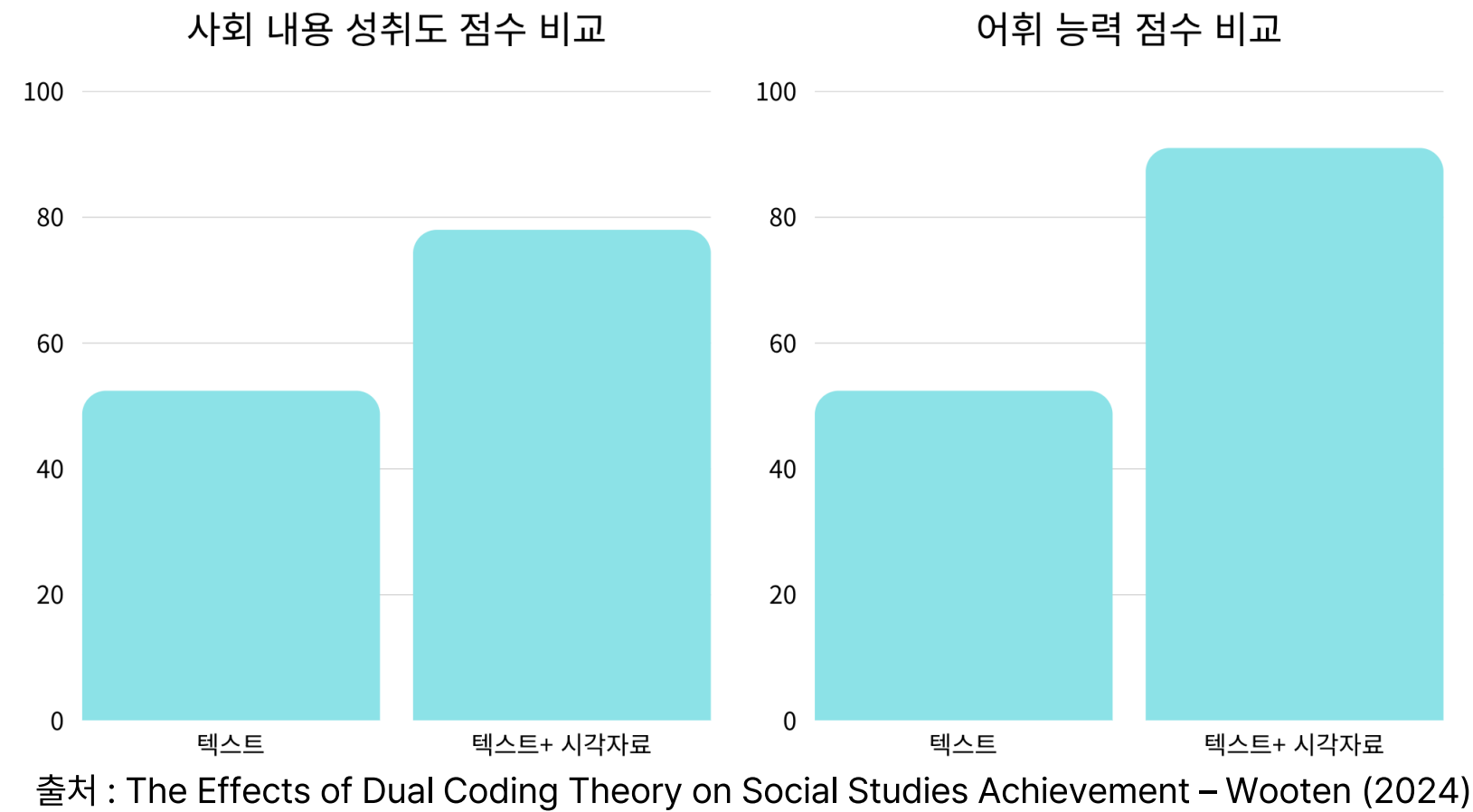
차별성

GIFPT  
Pipeline



# 01 What is G I F T ?

## 문제 정의



- 방대한 텍스트 속에서 핵심적인 내용을 빠르게 추출하고 정리하기 어려움
- 학습 자료에 포함된 도표와 수식은 설명이 부족하여 쉽게 소화하지 못함
- 개념을 단순히 암기하는 데 그치며, 학습 경험이 비효율적이고 흥미를 잃게 되는 문제 발생

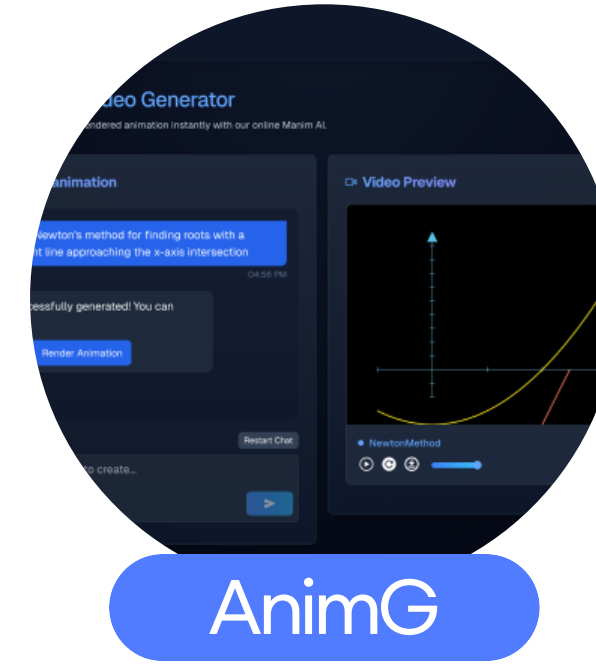
# 02 Visual Learning Research & Existing Tools



미리 정의된 알고리즘,  
자료구조에 한정된  
시각화



코드 기반 도구,  
학습자가 알고리즘을  
구현해야 함



자연어 프롬프트 기반  
단일 Manim 장면  
생성에 특화



문서 요약·설명에 강점

강의·논문 내용을 자유  
롭게 애니메이션화하  
기 어려움

GIFPT처럼  
‘원본 자료 → 자동 시각화’  
파이프라인은 제공 X

문서 전체를 다장면  
학습 영상으로 구현에  
한계

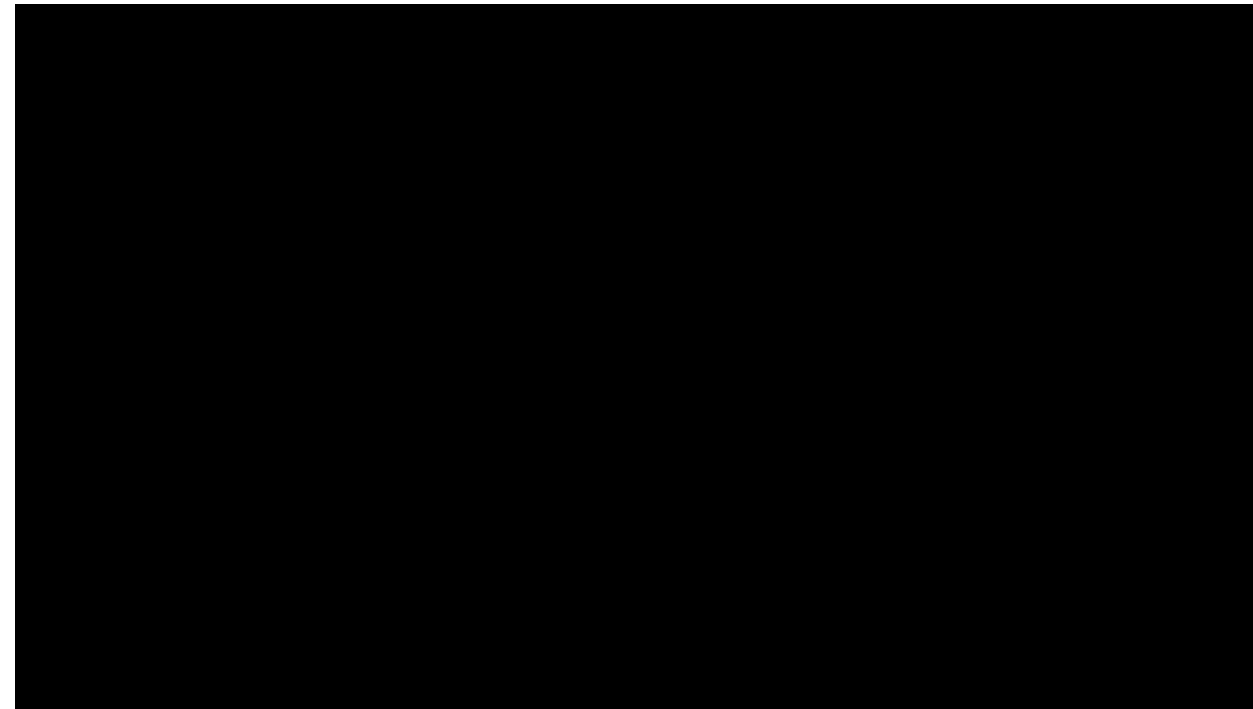
수식·절차·알고리즘을  
GIF/동영상으로  
자동 변환 기능 X

2

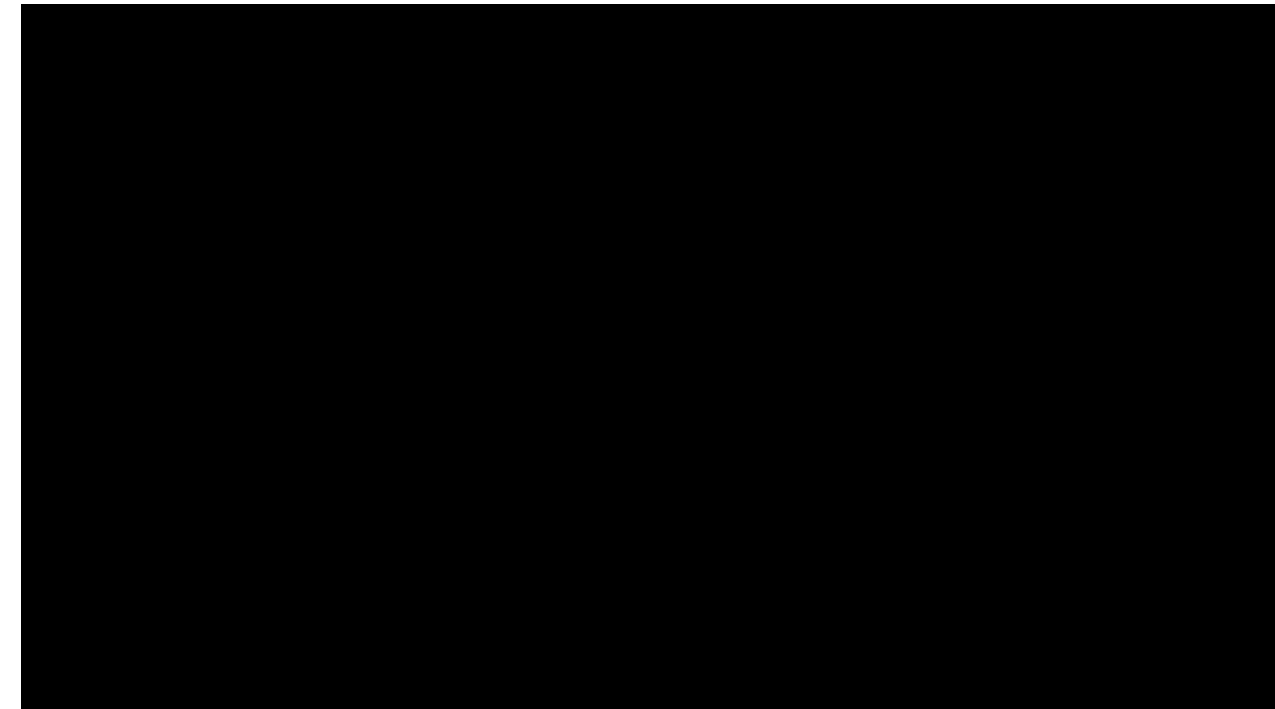
기존 도구들의 알고리즘 제한·정확도 한계를 넘어  
Manim + 생성형 AI 기반 ‘정확한 구조적 애니메이션’

03

# What makes GIFPT different?



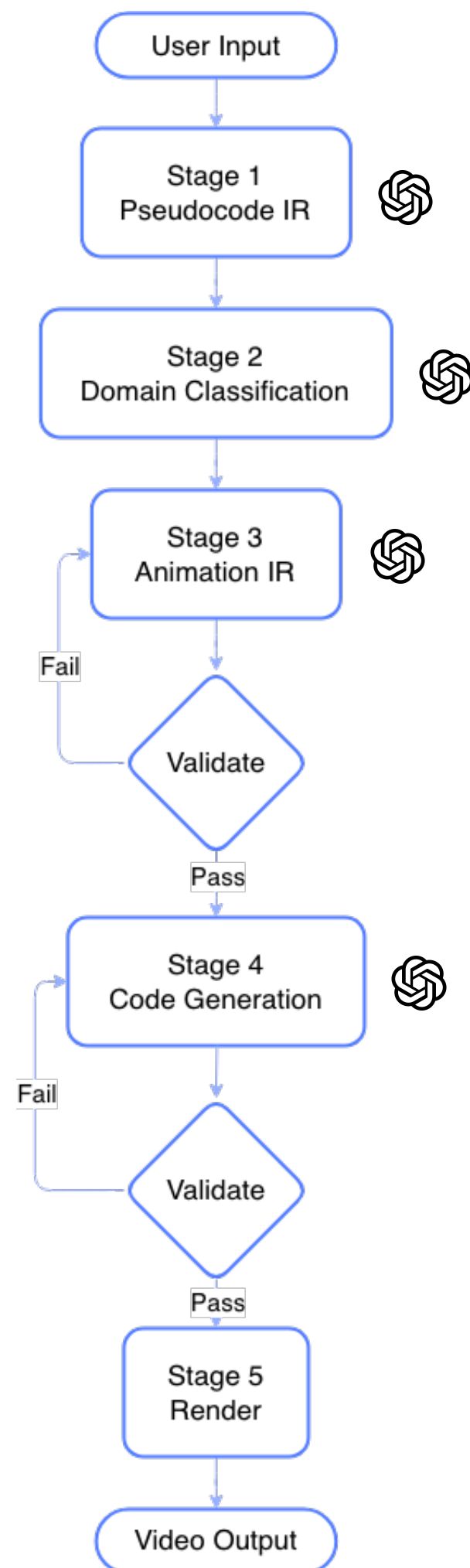
단일 AI + Manim



GIFPT

- Document-native input  
예제나 코드 조각, 단일 프롬프트가 아닌 실제 강의 슬라이드와 논문을 그대로 입력으로 사용
- Structured understanding  
단순 요약이 아니라, 시간 순서까지 담은 JSON 기반 IR 시퀀스로 구조화
- Multi-scene animations  
단발성 데모 장면이 아닌, 실제 알고리즘의 흐름을 따라가는 일관된 MP4 시퀀스 생성

# 04 How does G I F T work?



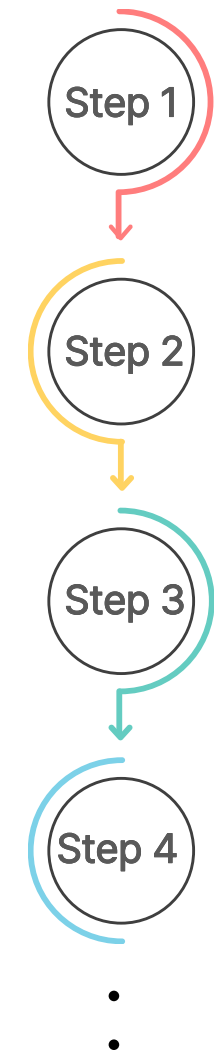
## Pipeline - 5단계의 순차적 처리

- Stage 1 (Pseudocode IR):  
사용자 설명을 구조화된 중간 표현으로 변환
- Stage 2 (Domain&Pattern Classification):  
알고리즘의 도메인과 패턴 분류
- Stage 3 (Animation IR):  
시각화를 위한 레이아웃과 애니메이션 IR 생성
- Stage 4 (Code Generation):  
실행 가능한 Manim 코드 생성
- Stage 5 (Render):  
최종 영상 렌더링

# 04 How does G I F T work?

## Stage 1: Pseudocode IR

```
**예시 (Bubble Sort)**:  
``json  
{  
  "metadata": {"title": "Bubble Sort"},  
  "entities": [  
    {"id": "array", "type": "array", "attributes": {"size": 5}}  
  ],  
  "operations": [  
    {"step": 1, "subject": "array", "action": "create",  
     "description": "Initialize array [5,2,8,1,9]},  
    {"step": 2, "subject": "array", "action": "compare",  
     "target": "array[0]", "description": "Compare 5 and 2"},  
    {"step": 3, "subject": "array", "action": "swap", "target":  
     "array[0]", "description": "Swap elements"}  
  ]  
}
```

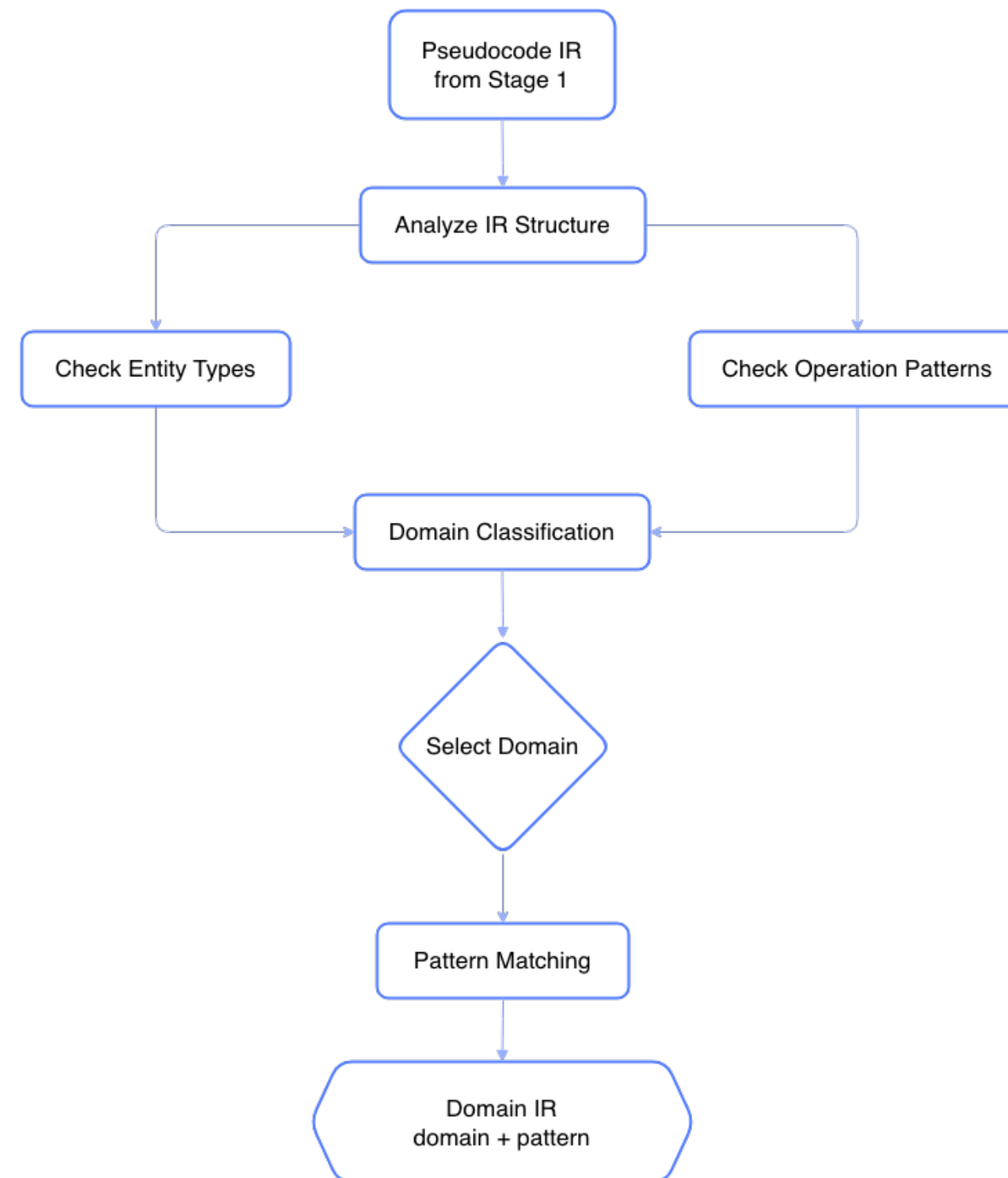


- 사용자의 input을 받아 알고리즘 정보를 추출하고 핵심 구성 요소 식별
- 단계별 동작을 step-by-step으로 정리
- 시각화할 객체들을 정의하는 Entities(id, type, attributes 형식) 생성
- 최종 출력은 순수 JSON (코드 생성 파이프라인에서 이 JSON을 받아 Manim 코드로 변환)

# 04 How does G I F T work?



## Stage 2: Domain Classification



- LLM이 1단계에서 생성된 Pseudocode IR 분석  
--> 알고리즘이 속한 도메인&시각화 패턴 분류
- 도메인은 '무엇을 그릴지', 패턴은 '어떻게 배치할지'  
--> 분류 재사용성·확장성·검증 가능성 크게 향상
- 이후 Animation IR 생성 단계에서  
적절한 시각적 구조와 애니메이션 스타일을 결정하는  
핵심 정보로 활용

# 04

## How does G I F P T work?

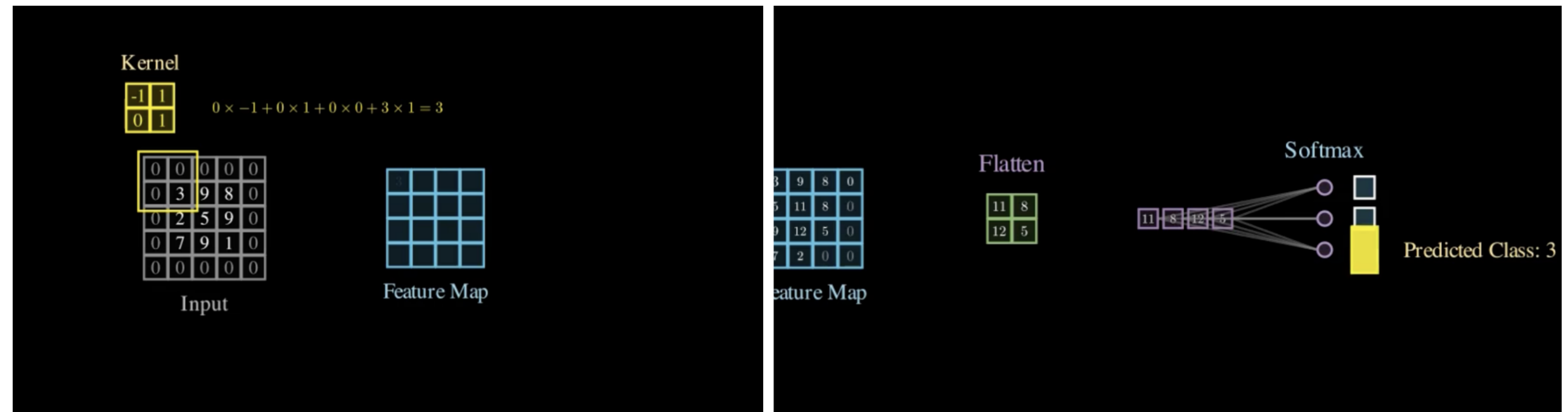
### Stage 3: Animation IR

```
```json
{
  "metadata": {"domain": "string", "title": "string"},
  "layout": [
    {"id": "string", "shape": "matrix | array | rectangle | circle", "position": [x, y],
    "color": "string", "label": "string", "data": "array | matrix | string",
    "dimensions": "string"}
  ],
  "actions": [
    {"step": "number", "target": "string", "animation": "fade_in | fade_out | move | highlight | swap", "description": "string"}
  ]
}
```
```

- Pseudocode IR를 기반으로 시각화에 필요한 객체(layout)와 동작(actions) 정의
- Domain 정보(metadata.domain)를 활용해 형태/데이터 디테일 풍부화  
(예: sorting 배열 값, cnn kernel 크기, attention Q/K/V 등)
- 화면상의 위치, 모양, 크기 등의 시각적 속성 포함
- Pseudocode IR의 operations를 Animation actions로 변환  
--> 각 step마다 어떤 객체(target)가 어떤 애니메이션을 수행할지 정의

# 04 How does G I F T work?

## Stages 4, 5: Code Generation and Rendering



- 4단계: 검증 완료된 Animation IR을 기반으로 실제 실행 가능한 Manim Python 코드 생성
- 5단계: 렌더링으로 진입하여, 생성된 Python 파일을 임시 디렉토리에 저장  
--> Manim 렌더러 실행하여 영상 생성

05

# How does G I F T validate?

## Validation

How do we ensure the best quality?

Animation IR  
구조 검증

필수 필드 Animation IR 생성 이후...

- JSON 파싱 확인
- 필수 필드 존재 여부
- step 단조 증가
- layout.id 중복 확인
- data-shape 기반 정합성 점검

오류 발생 시

Domain/Pattern

분류 결과 유지

--> Animation IR만 재생성

Manim Code  
검증

생성된 Manim 코드에 대해...

- 색상 팔레트 일치 여부
- 기본 구문 결함 검사
- 실행 가능성 확인

오류 발생 시

동일한 Animation IR 유지

--> 최대 3회까지 코드 재생성

RAG

- ChromaDB에 저장된 성공적인 과거 예제 라이브러리 참조
- 현재 생성된 코드가 유사한 패턴의 검증된 사례와 구조적으로 일치하는지 비교 검증
- 생성된 코드와 레이아웃을 렌더링 전에 RAG로 검색된 참조 예제와 대조하여 잠재적 오류를 사전에 감지

05

# How does G I F P T validate?

## Limitations & Future work



### Limitations

- Domain / Pattern 범위의 제한
- Semantic Validation 미흡
- 레이아웃 안정성 부족
- 높은 토큰 사용량과 지연 시간

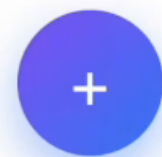


### Future work

- Domain & Pattern 확장  
그래프·DP·수학 개념 등 도메인 확대 및 패턴 조합 템플릿 지원
- Semantic-level Validator 도입  
엔티티 참조 무결성, 연산 순서 규칙 등 의미적 검증 기능 강화
- 레이아웃 최적화 엔진 개발  
자동 충돌 탐지, 좌표 보정, 과밀도 기반 썬 분할 기능 추가
- 파이프라인 비용·지연 최적화  
경량 모델 활용, 프롬프트 압축으로 실행 효율 개선

# My Projects

Create new projects or continue working on existing ones



Create New Project

Upload a PDF and generate a new video

Sequence  
1 2 3 4 5 6 7

1764523708637\_12-...

Status: SUCCESS

Initial Matrix  

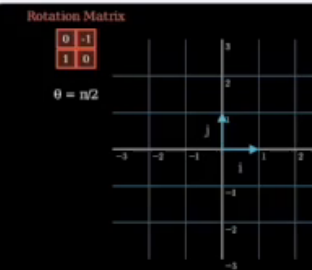
|   |   |   |
|---|---|---|
| a | b | c |
| d | e | f |
| g | h | i |

Rotated Matrix  

|  |  |  |
|--|--|--|
|  |  |  |
|  |  |  |
|  |  |  |

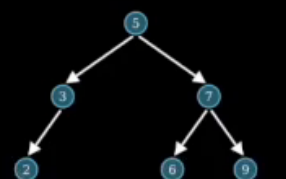
1764522646603\_RotationMatr

Status: SUCCESS



1764522453969\_RotationMatri

Status: SUCCESS



1764519053809\_search

Status: SUCCESS



**Thank You**