ViMax 架构详解

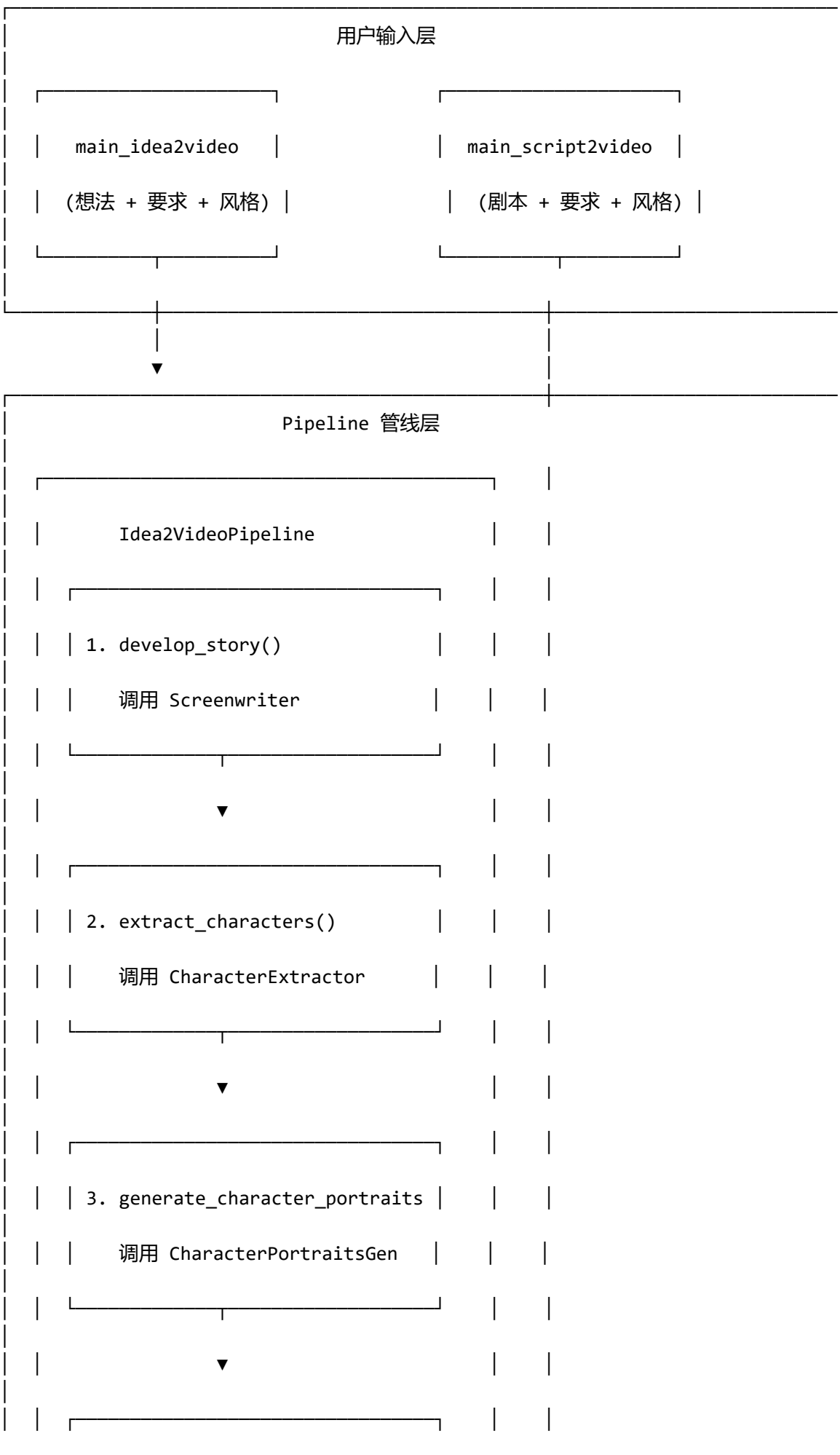
ViMax 架构详解

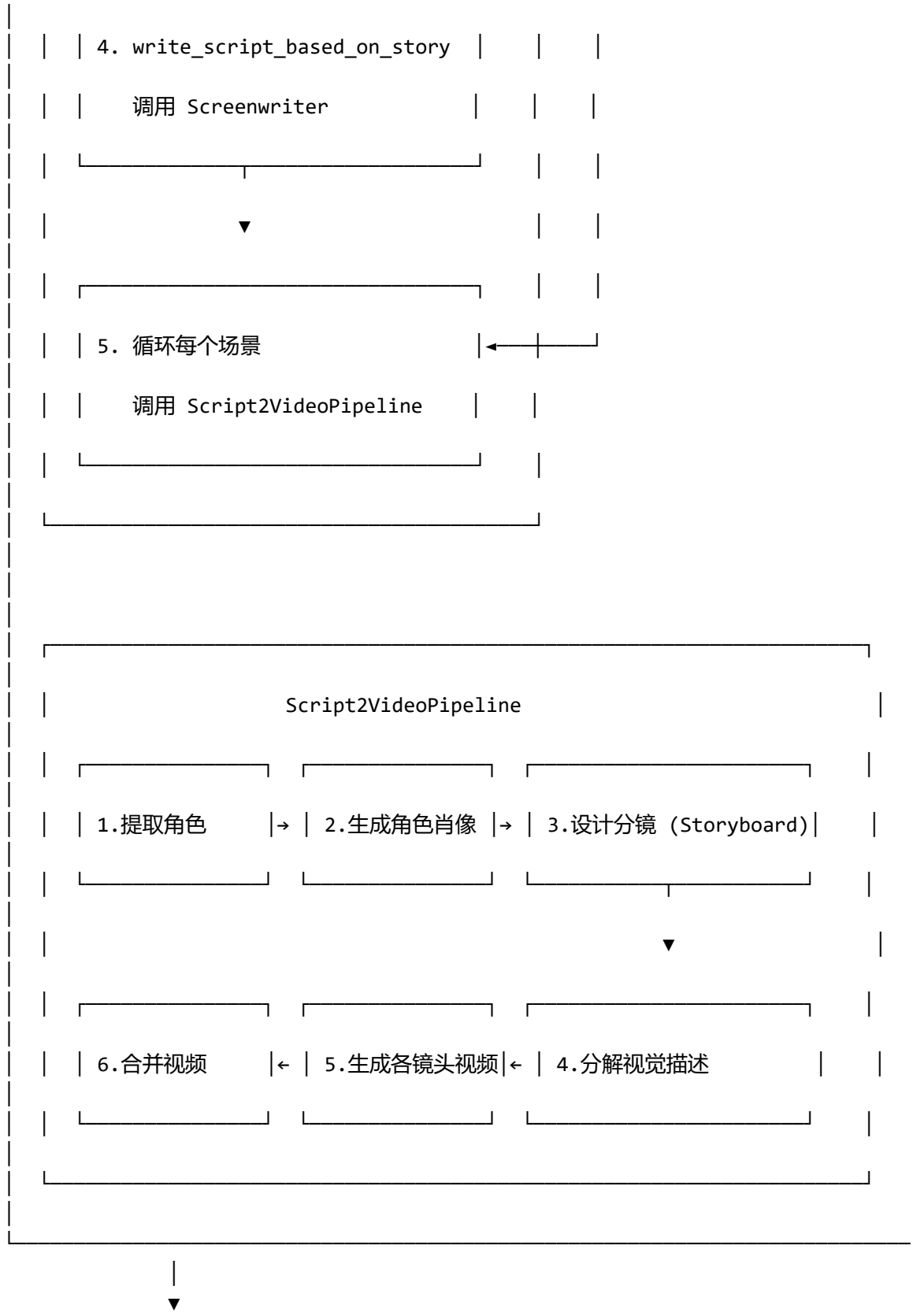
> 本文档详细介绍 ViMax 的代码架构、模块功能和调用流程。

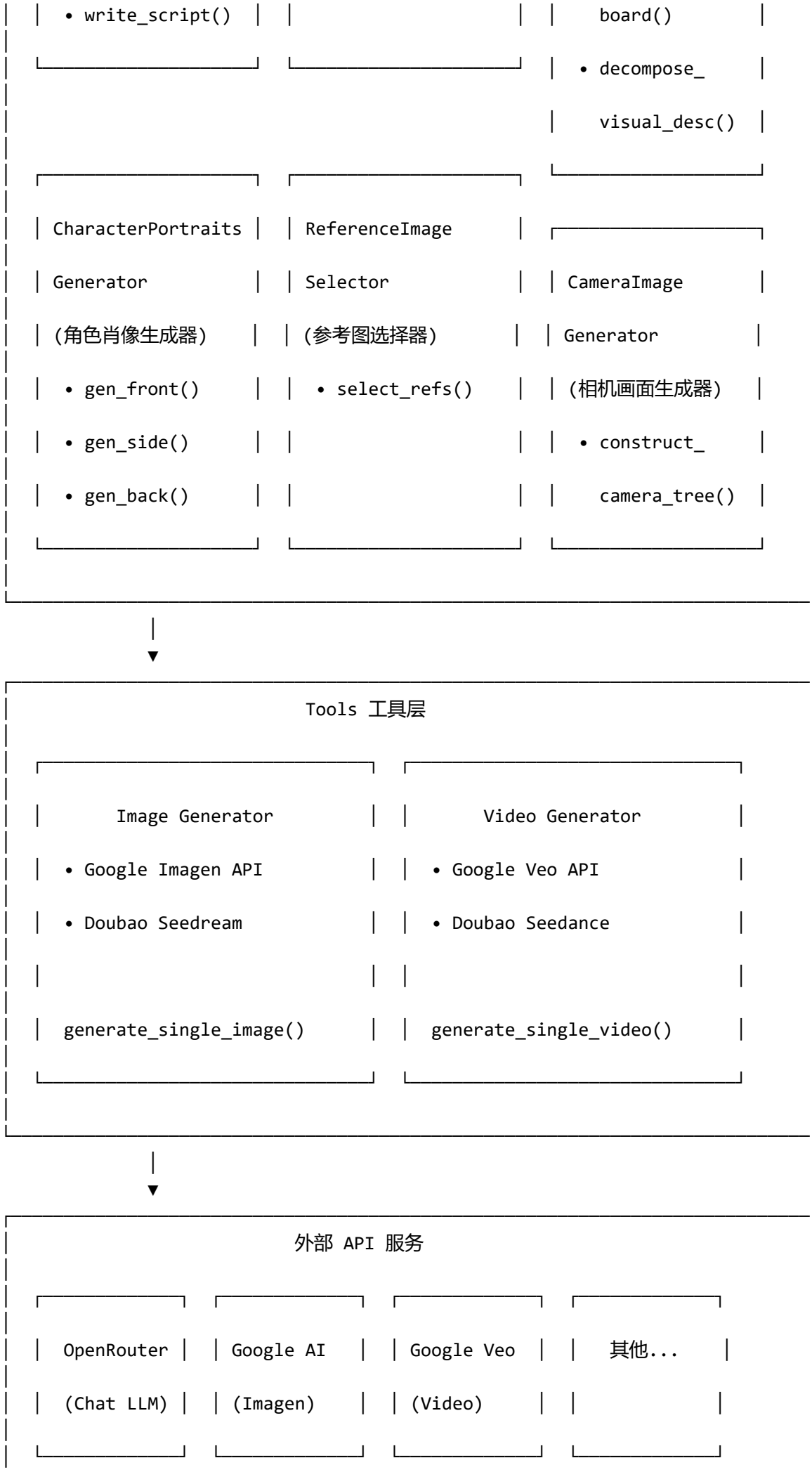
📁 项目结构

```
ViMax/
├── main_idea2video.py      # 入口：想法 → 视频
├── main_script2video.py    # 入口：剧本 → 视频
├── configs/                # 配置文件
│   ├── idea2video.yaml
│   └── script2video.yaml
├── pipelines/              # 主流程管线
│   ├── idea2video_pipeline.py
│   └── script2video_pipeline.py
├── agents/                 # AI 智能体（核心逻辑）
│   ├── screenwriter.py           # 编剧
│   ├── character_extractor.py    # 角色提取器
│   ├── character_portraits_generator.py  # 角色肖像生成器
│   ├── storyboard_artist.py      # 分镜师
│   ├── camera_image_generator.py # 相机画面生成器
│   └── reference_image_selector.py # 参考图像选择器
├── tools/                  # 外部工具封装
│   ├── image_generator_*.py      # 图像生成器
│   └── video_generator_*.py      # 视频生成器
├── interfaces/             # 数据结构定义
│   ├── character.py
│   ├── scene.py
│   ├── shot_description.py
│   └── camera.py
└── utils/                  # 工具函数
    ├── rate_limiter.py
    ├── retry.py
    └── video.py
```

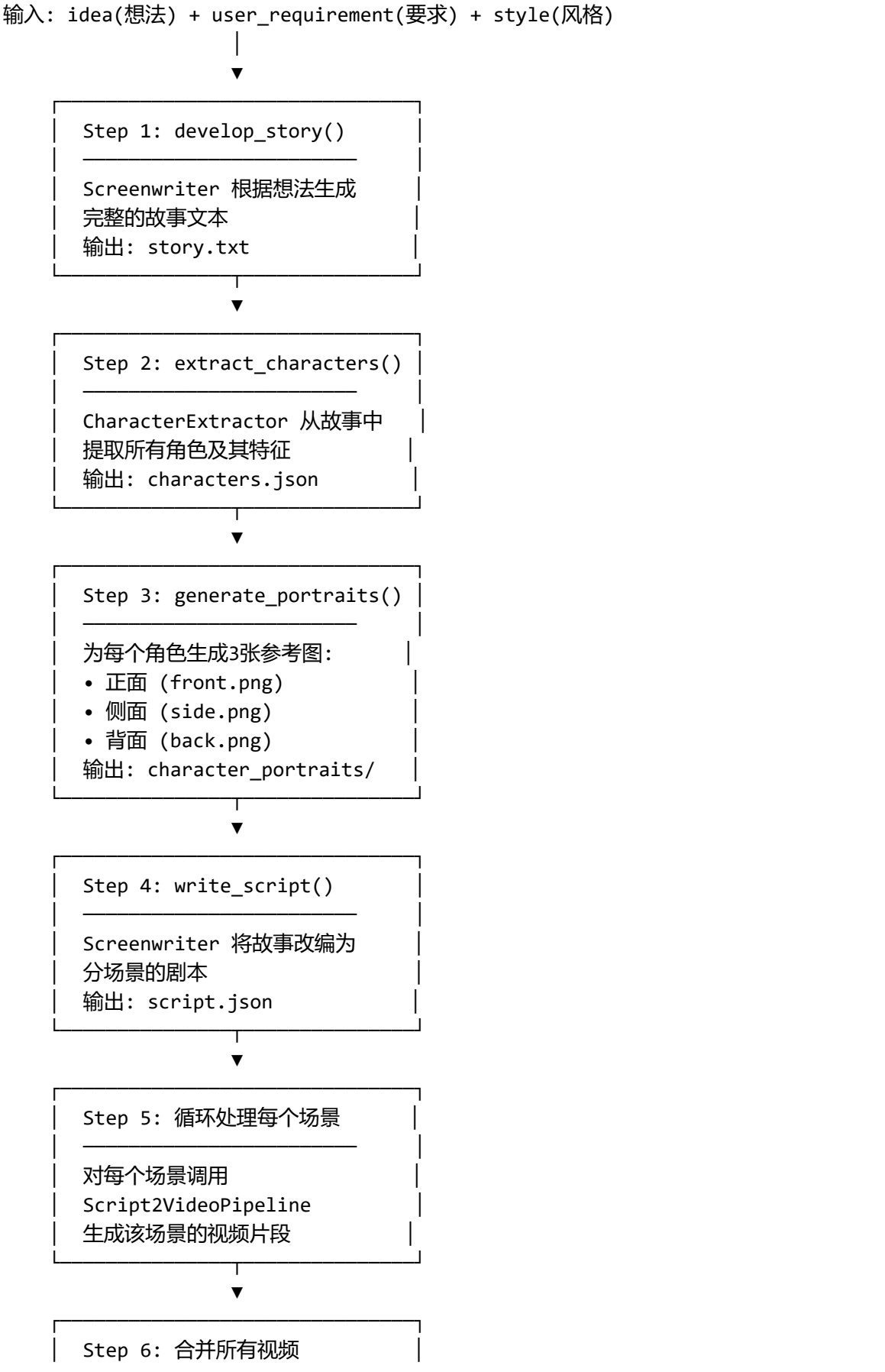
整体架构图


用户输入层
main_idea2video
(想法 + 要求 + 风格)
main_script2video
(剧本 + 要求 + 风格)
Pipeline 管线层
Idea2VideoPipeline
1. develop_story()
调用 Screenwriter
2. extract_characters()
调用 CharacterExtractor
3. generate_character_portraits
调用 CharacterPortraitsGen


4. write_script_based_on_story
调用 Screenwriter
5. 循环每个场景
调用 Script2VideoPipeline
Script2VideoPipeline
1.提取角色
2.生成角色肖像
3.设计分镜 (Storyboard)
4.分解视觉描述
5.生成各镜头视频
6.合并视频
Agents 智能体层
Screenwriter
(编剧)
• develop_story()
CharacterExtractor
(角色提取器)
• extract_chars()
StoryboardArtist
(分镜师)
• design_story


• write_script()
board()
• decompose_
visual_desc()
CharacterPortraits
Generator
(角色肖像生成器)
• gen_front()
• gen_side()
• gen_back()
ReferenceImage
Selector
(参考图选择器)
• select_refs()
CameraImage
Generator
(相机画面生成器)
• construct_
camera_tree()
Tools 工具层
Image Generator
• Google Imagen API
• Doubao Seedream
generate_single_image()
Video Generator
• Google Veo API
• Doubao Seedance
generate_single_video()
外部 API 服务
OpenRouter
(Chat LLM)
Google AI
(Imagen)
Google Veo
(Video)
其他...

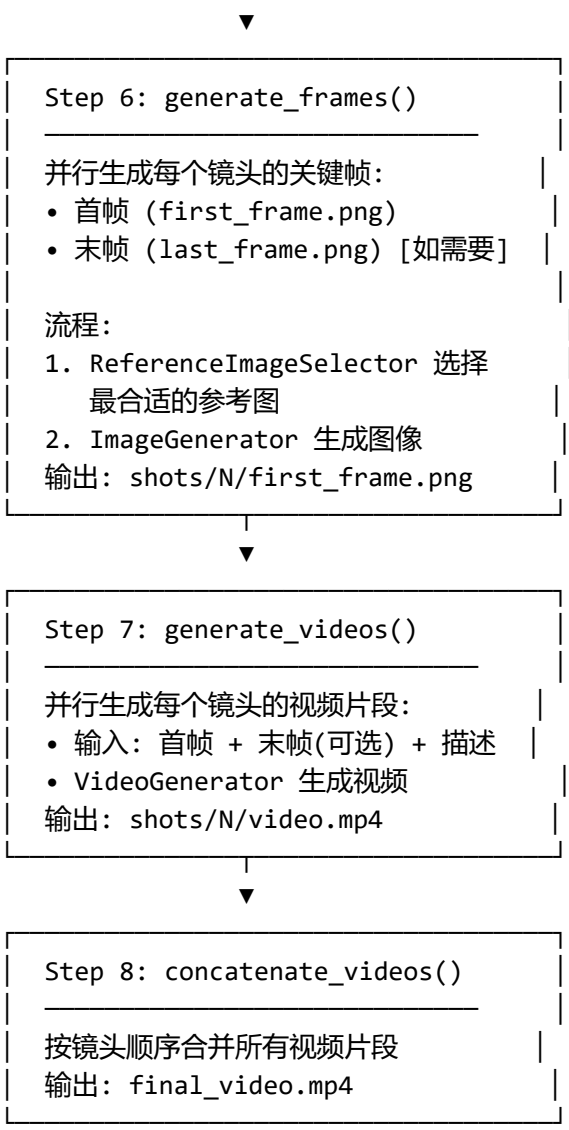
Idea2Video 完整流程



```
┌──────────────────────────────────┐
│  ──────────────────────          │
│  使用 moviepy 将所有场景          │
│  视频拼接成最终视频               │
│  输出: final_video.mp4           │
└──────────────────────────────────┘
```

🔄 Script2Video 完整流程

```
输入: script(剧本) + user_requirement(要求) + style(风格)
                    │
                    ▼
┌──────────────────────────────────────┐
│  Step 1: extract_characters()        │
│  从剧本中提取角色信息                 │
└──────────────────┬───────────────────┘
                   ▼
┌──────────────────────────────────────┐
│  Step 2: generate_character_         │
│          portraits()                 │
│  为每个角色生成正面/侧面/背面肖像      │
└──────────────────┬───────────────────┘
                   ▼
┌──────────────────────────────────────┐
│  Step 3: design_storyboard()         │
│  ──────────────────────────          │
│  StoryboardArtist 设计分镜:          │
│  • 镜头编号 (shot_idx)               │
│  • 相机编号 (cam_idx)                │
│  • 视觉描述 (visual_desc)            │
│  • 音频描述 (audio_desc)             │
│  输出: storyboard.json               │
└──────────────────┬───────────────────┘
                   ▼
┌──────────────────────────────────────┐
│  Step 4: decompose_visual_           │
│          descriptions()              │
│  ──────────────────────────          │
│  将每个镜头的视觉描述分解为:          │
│  • 首帧描述 (ff_desc)                │
│  • 末帧描述 (lf_desc)                │
│  • 运动描述 (motion_desc)            │
│  • 变化类型 (small/medium/large)     │
│  输出: shots/N/shot_description      │
└──────────────────┬───────────────────┘
                   ▼
┌──────────────────────────────────────┐
│  Step 5: construct_camera_tree()     │
│  ──────────────────────────          │
│  构建相机依赖树:                      │
│  确定哪些镜头共用同一相机位置,        │
│  哪些镜头之间需要过渡                 │
│  输出: camera_tree.json              │
└──────────────────┬───────────────────┘
                   ▼
```



Agents 详细说明

1. Screenwriter (编剧)

文件: `agents/screenwriter.py`

功能: - 将简单想法扩展为完整故事 - 将故事改编为分场景剧本

方法:

方法	输入	输出	说明
`develop_story()`	idea, user_requirement	str (故事文本)	根据想法生成完整故事
`write_script_based_on_story()`	story, user_requirement	List[str] (场景剧本列表)	将故事改编为剧本

---

2. CharacterExtractor (角色提取器)

文件: `agents/character_extractor.py`

功能: 从剧本/故事中识别和提取角色信息

方法:

方法	输入	输出	说明
`extract_characters()`	script/story	List[CharacterInScene]	提取角色名称、特征、描述

输出数据结构 `CharacterInScene`:

```
{
    "idx": 0,                              # 角色索引
    "identifier_in_scene": "Alice",        # 角色名称
    "static_features": "短发，绿色裙子",  # 静态特征（外貌/服装）
    "dynamic_features": "活泼，爱笑"       # 动态特征（性格/动作习惯）
}
```

---

3. CharacterPortraitsGenerator (角色肖像生成器)

文件: `agents/character_portraits_generator.py`

功能: 为每个角色生成一致性参考图像

方法:

方法	输入	输出	说明
`generate_front_portrait()`	character, style	ImageOutput	生成正面肖像
`generate_side_portrait()`	character, front_path	ImageOutput	基于正面生成侧面
`generate_back_portrait()`	character, front_path	ImageOutput	基于正面生成背面

---

4. StoryboardArtist (分镜师)

文件: `agents/storyboard_artist.py`

功能: - 将剧本转换为专业分镜 - 分解视觉描述为可执行的帧描述

方法:

方法	输入	输出	说明
`design_storyboard()`	script, characters, requirement	List[ShotBriefDescription]	设计完整分镜
`decompose_visual_description()`	shot_brief_desc, characters	ShotDescription	分解为首帧/末帧/运动描述

输出数据结构 `ShotDescription`:

```
{
    "idx": 0,                    # 镜头索引
    "cam_idx": 0,                # 相机索引（同一相机可拍多个镜头）
    "visual_desc": "...",        # 完整视觉描述
    "ff_desc": "...",            # 首帧描述
    "ff_vis_char_idxs": [0, 1],  # 首帧中可见的角色索引
    "lf_desc": "...",            # 末帧描述
    "lf_vis_char_idxs": [0],     # 末帧中可见的角色索引
    "motion_desc": "...",        # 运动描述（相机移动+角色动作）
    "audio_desc": "...",         # 音频描述（对话/音效）
    "variation_type": "small"    # 变化程度: small/medium/large
}
```

---

5. CameraImageGenerator (相机画面生成器)

文件: `agents/camera_image_generator.py`

功能: - 构建相机依赖树 - 生成相机过渡视频

方法:

方法	输入	输出	说明
`construct_camera_tree()`	cameras, shot_descs	List[Camera]	构建相机依赖关系

方法	输入	输出	说明
`generate_transition_video()`	shot1_desc, shot2_desc, shot1_ff	VideoOutput	生成过渡视频

6. ReferenceImageSelector (参考图选择器)

文件: `agents/reference_image_selector.py`

功能: 智能选择最合适的参考图像用于生成新帧

方法:

方法	输入	输出	说明
`select_reference_images_and_generate_prompt()`	available_pairs, frame_desc	dict	选择参考图 + 生成提示词

🔧 Tools 工具说明

Image Generator (图像生成器)

类名	API	说明
`ImageGeneratorNanobananaGoogleAPI`	Google AI Studio	使用 Imagen 生成图像
`ImageGeneratorDoubaoSeedreamYunwuAPI`	字节云雾	使用豆包生成图像

通用方法:

```python
async def generate_single_image(
    prompt: str,                    # 生成提示词
    reference_image_paths: List[str],  # 参考图路径列表
    size: str = "1600x900"          # 输出尺寸
) -> ImageOutput
```

Video Generator (视频生成器)

类名	API	说明
`VideoGeneratorVeoGoogleAPI`	Google Veo	使用 Veo 生成视频
`VideoGeneratorDoubaoSeedanceYunwuAPI`	字节云雾	使用豆包生成视频

通用方法:

```python
async def generate_single_video(
    prompt: str,                    # 生成提示词（动作+音频描述）
    reference_image_paths: List[str]   # 参考帧路径（首帧/末帧）
) -> VideoOutput
```

📁 输出目录结构

```
.working_dir/
├── idea2video/                     # Idea2Video 输出
│   ├── story.txt                   # 生成的故事
│   ├── characters.json             # 提取的角色
│   ├── character_portraits/        # 角色肖像
│   │   ├── 0_Alice/
│   │   │   ├── front.png
│   │   │   ├── side.png
│   │   │   └── back.png
│   │   └── 1_Bob/
│   │       └── ...
│   ├── character_portraits_registry.json
│   ├── script.json                 # 分场景剧本
│   ├── scene_0/                    # 第一个场景
│   │   ├── storyboard.json         # 分镜
│   │   ├── camera_tree.json        # 相机树
│   │   ├── shots/
│   │   │   ├── 0/
│   │   │   │   ├── shot_description.json
│   │   │   │   ├── first_frame.png
│   │   │   │   ├── last_frame.png (如需要)
│   │   │   │   └── video.mp4
│   │   │   ├── 1/
│   │   │   └── ...
│   │   └── final_video.mp4         # 场景视频
│   ├── scene_1/
│   └── final_video.mp4             # 最终完整视频
│
└── script2video/                   # Script2Video 输出
    └── (结构同上)
```

🚀 快速使用

1. 安装依赖

```
cd C:\Users\yanie\Desktop\ViMax
uv sync
```

2. 配置 API Key

编辑 `configs/idea2video.yaml`:

```yaml
chat_model:
  init_args:
    model: google/gemini-2.5-flash-lite-preview-09-2025
    api_key: <你的 OpenRouter API Key>
    base_url: https://openrouter.ai/api/v1

image_generator:
  class_path: tools.ImageGeneratorNanobananaGoogleAPI
  init_args:
    api_key: <你的 Google AI Studio API Key>

video_generator:
  class_path: tools.VideoGeneratorVeoGoogleAPI
  init_args:
    api_key: <你的 Google AI Studio API Key>
```

3. 运行

```bash
# 想法转视频
uv run python main_idea2video.py

# 剧本转视频
uv run python main_script2video.py
```

📝 关键概念

相机 (Camera) vs 镜头 (Shot)

- **相机 (cam_idx)**: 一个固定的拍摄位置/角度
- **镜头 (shot_idx)**: 一段连续的视频片段

一个相机可以拍摄多个镜头（不移动相机的情况下）。

变化类型 (variation_type)

类型	说明	需要末帧?
`small`	微小变化（表情、小动作）	❌
`medium`	中等变化（新角色出现、转身）	✅
`large`	大变化（相机大幅移动、场景切换）	✅

异步并行处理

ViMax 使用 `asyncio` 实现并行处理： - 多个角色的肖像可以并行生成 - 多个镜头的首帧可以并行生成 - 视频生成等待对应的帧生成完成后开始

最后更新: 2026-02-03