ViMax集成计划

ViMax → video-agent-skill 集成计划

概述

本文档详细说明如何将 ViMax 的独特功能整合到 video-agent-skill 项目中，包括需要复制的文件、需要添加/修改的函数，以及集成步骤。

一、ViMax 独特功能清单

1. 角色一致性系统

功能	描述	video-agent-skill 现状
CharacterExtractor	从剧本自动提取角色信息	❌ 无
CharacterPortraitsGenerator	生成角色多角度肖像(正面/侧面/背面)	❌ 无
ReferenceImageSelector	智能选择最佳参考图像	❌ 无

2. 相机系统

功能	描述	video-agent-skill 现状
Camera 接口	层级相机树结构	❌ 无

功能	描述	video-agent-skill 现状
CameraImageGenerator	生成相机过渡视频	❌ 无
场景检测帧提取	使用 scenedetect 提取关键帧	❌ 无

3. 多 Agent 协作管道

功能	描述	video-agent-skill 现状
Screenwriter	创意→剧本生成	⚠️ 有基础文本生成
StoryboardArtist	剧本→分镜头脚本	❌ 无
Pipeline 编排	YAML 配置的 Agent 管道	❌ 无

4. 结构化数据接口

接口	用途
CharacterInScene	场景中的角色信息
CharacterInEvent	事件中的角色信息
CharacterInNovel	小说中的角色信息
ShotDescription	完整镜头描述
ShotBriefDescription	简化镜头描述
Camera	相机配置与层级
ImageOutput	图像输出封装
VideoOutput	视频输出封装

二、需要复制的文件

2.1 核心 Agent 文件

```
ViMax/agents/
├── character_extractor.py       → video-agent-skill/agents/
├── character_portraits_generator.py → video-agent-skill/agents/
├── reference_image_selector.py  → video-agent-skill/agents/
├── camera_image_generator.py    → video-agent-skill/agents/
├── screenwriter.py              → video-agent-skill/agents/
├── storyboard_artist.py         → video-agent-skill/agents/
└── __init__.py                  → 合并到现有 __init__.py
```

2.2 接口定义文件

```
ViMax/interfaces/
├── character.py                → video-agent-skill/interfaces/
├── shot_description.py         → video-agent-skill/interfaces/
├── camera.py                   → video-agent-skill/interfaces/
├── image_output.py             → video-agent-skill/interfaces/
└── video_output.py             → video-agent-skill/interfaces/
```

2.3 工具函数文件

```
ViMax/utils/
└── image.py                    → video-agent-skill/utils/
```

2.4 管道文件

```
ViMax/pipelines/
├── idea2video_pipeline.py      → video-agent-skill/pipelines/
└── script2video_pipeline.py    → video-agent-skill/pipelines/
```

2.5 配置文件

```
ViMax/configs/
├── idea2video.yaml             → video-agent-skill/configs/
└── script2video.yaml           → video-agent-skill/configs/
```

2.6 Prompt 模板

```
ViMax/prompts/
└── *.txt                       → video-agent-skill/prompts/
```

三、需要创建的目录结构

在 video-agent-skill 中创建以下目录：

```bash
mkdir -p video-agent-skill/agents
mkdir -p video-agent-skill/interfaces
mkdir -p video-agent-skill/pipelines
mkdir -p video-agent-skill/configs
mkdir -p video-agent-skill/prompts
```

四、需要添加的依赖

在 `video-agent-skill/requirements.txt` 或 `pyproject.toml` 中添加：

```
pydantic>=2.0
scenedetect[opencv]>=0.6
opencv-python>=4.8
pillow>=10.0
google-genai>=1.0
litellm>=1.0
pyyaml>=6.0
```

五、需要修改的函数/代码

5.1 适配 image_generator

ViMax 使用 Google Imagen，video-agent-skill 有多个图像生成器。需要创建适配层：

```python
# video-agent-skill/adapters/image_generator_adapter.py

class ImageGeneratorAdapter:
    """适配 ViMax agents 到 video-agent-skill 的图像生成器"""

    def __init__(self, generator_type="flux"):
        self.generator = self._init_generator(generator_type)

    async def generate(self, prompt: str, **kwargs) -> str:
        """返回生成图像的 base64 或路径"""
        # 调用 video-agent-skill 的图像生成
        pass
```

5.2 适配 video_generator

```python
# video-agent-skill/adapters/video_generator_adapter.py

class VideoGeneratorAdapter:
    """适配 ViMax agents 到 video-agent-skill 的视频生成器"""

    def __init__(self, generator_type="kling"):
        self.generator = self._init_generator(generator_type)

    async def generate(self, image: str, prompt: str, **kwargs) -> str:
        """返回生成视频的路径"""
        pass
```

5.3 适配 LLM 调用

ViMax 使用 `litellm`，需要确保与 video-agent-skill 的 LLM 调用兼容：

```python
# video-agent-skill/adapters/llm_adapter.py

from litellm import completion

class LLMAdapter:
    """统一 LLM 调用接口"""

    def __init__(self, model="openrouter/anthropic/claude-3.5-sonnet"):
        self.model = model

    async def chat(self, messages: list, **kwargs):
        response = await completion(
            model=self.model,
            messages=messages,
            **kwargs
        )
        return response
```

六、集成步骤

第一阶段：基础设施 (1-2天)

1. **创建目录结构**

   ```
   cd video-agent-skill
   mkdir -p agents interfaces pipelines configs prompts adapters
   ```

2. **复制接口文件**

   ```
   cp ../interfaces/*.py interfaces/
   ```

3. **复制工具函数**

   ```
   cp ../utils/image.py utils/
   ```

4. **安装依赖**

   ```
   pip install pydantic scenedetect opencv-python pillow google-genai
            litellm pyyaml
   ```

第二阶段：适配层 (2-3天)

1. **创建 ImageGeneratorAdapter**
   - 映射 ViMax 的 `image_generator` 到 video-agent-skill 的 Flux/SDXL

2. **创建 VideoGeneratorAdapter**
   - 映射 ViMax 的 `video_generator` 到 video-agent-skill 的 Kling/Runway
3. **创建 LLMAdapter**
   - 统一 LLM 调用接口

第三阶段：Agent 移植 (3-5天)

1. **移植 CharacterExtractor**
   - 修改 LLM 调用为适配层
   - 测试角色提取功能
2. **移植 CharacterPortraitsGenerator**
   - 修改图像生成调用为适配层
   - 测试多角度肖像生成
3. **移植 ReferenceImageSelector**
   - 确保多模态 LLM 调用正常
   - 测试参考图选择
4. **移植 CameraImageGenerator**
   - 确保 scenedetect 正常工作
   - 测试相机视频生成
5. **移植 Screenwriter 和 StoryboardArtist**
   - 修改 prompt 路径
   - 测试剧本和分镜生成

第四阶段：管道集成 (2-3天)

1. **移植 Pipeline 类**
   - 适配配置加载
   - 适配 Agent 调用
2. **创建统一入口**

```python
# video-agent-skill/run_vimax.py
from pipelines import Idea2VideoPipeline, Script2VideoPipeline
```

3. **测试端到端流程**

七、文件修改详情

7.1 character_extractor.py 修改

```python
# 原始 (ViMax)
from litellm import completion

# 修改为
```

```python
from adapters.llm_adapter import LLMAdapter

# 原始
response = completion(model=self.model, messages=messages)

# 修改为
llm = LLMAdapter(model=self.model)
response = await llm.chat(messages)
```

7.2 character_portraits_generator.py 修改

```python
# 原始 (ViMax)
image_result = self.image_generator.generate(prompt)

# 修改为
from adapters.image_generator_adapter import ImageGeneratorAdapter
image_gen = ImageGeneratorAdapter(generator_type="flux")
image_result = await image_gen.generate(prompt)
```

7.3 camera_image_generator.py 修改

```python
# 原始 (ViMax)
video_result = self.video_generator.generate(image, prompt)

# 修改为
from adapters.video_generator_adapter import VideoGeneratorAdapter
video_gen = VideoGeneratorAdapter(generator_type="kling")
video_result = await video_gen.generate(image, prompt)
```

八、测试计划

单元测试

```python
# tests/test_character_extractor.py
def test_extract_characters():
    extractor = CharacterExtractor(model="gpt-4")
    script = "小明和小红在公园里散步..."
    characters = extractor.extract(script)
    assert len(characters) >= 2

# tests/test_portrait_generator.py
def test_generate_portraits():
    generator = CharacterPortraitsGenerator()
    character = CharacterInScene(name="小明", description="20岁男性")
    portraits = generator.generate(character)
    assert "front" in portraits
```

```
    assert "side" in portraits
    assert "back" in portraits
```

集成测试

```
# tests/test_pipeline.py
def test_idea2video_pipeline():
    pipeline = Idea2VideoPipeline(config_path="configs/idea2video.yaml")
    result = pipeline.run(idea="一个关于友谊的短片")
    assert result.video_path.exists()
```

九、预估时间

阶段	时间	备注
基础设施	1-2天	创建目录、复制文件、安装依赖
适配层	2-3天	创建三个适配器
Agent 移植	3-5天	移植6个 Agent
管道集成	2-3天	移植 Pipeline、创建入口
测试调试	2-3天	单元测试、集成测试
总计	**10-16天**	

十、风险与注意事项

10.1 API 兼容性

- ViMax 使用 Google Imagen/Veo，video-agent-skill 使用其他模型
- 需要确保 prompt 格式兼容

10.2 异步/同步

- ViMax 部分代码是同步的
- video-agent-skill 可能需要异步
- 需要统一处理

10.3 依赖冲突

- 检查 pydantic 版本兼容性
- 检查 opencv 和 scenedetect 版本

10.4 文件路径

- ViMax 使用相对路径
- 需要适配 video-agent-skill 的路径结构

十一、快速开始命令

```
# 1. 进入 video-agent-skill 目录
cd C:\Users\yanie\Desktop\ViMax\video-agent-skill

# 2. 创建目录
mkdir agents interfaces pipelines configs prompts adapters

# 3. 复制文件
copy ..\interfaces\*.py interfaces\
copy ..\utils\image.py utils\
copy ..\agents\*.py agents\
copy ..\pipelines\*.py pipelines\
copy ..\configs\*.yaml configs\
xcopy ..\prompts\* prompts\ /E

# 4. 安装依赖
pip install pydantic scenedetect opencv-python pillow google-genai litellm
         pyyaml

# 5. 创建适配层（手动编写）
# adapters/image_generator_adapter.py
# adapters/video_generator_adapter.py
# adapters/llm_adapter.py
```

附录：文件复制清单

必须复制

- [x] interfaces/character.py
- [x] interfaces/shot_description.py
- [x] interfaces/camera.py
- [x] interfaces/image_output.py
- [x] interfaces/video_output.py
- [x] utils/image.py
- [x] agents/character_extractor.py
- [x] agents/character_portraits_generator.py
- [x] agents/reference_image_selector.py
- [x] agents/camera_image_generator.py

可选复制

- [ ] `agents/screenwriter.py` (如需完整管道)
- [ ] `agents/storyboard_artist.py` (如需完整管道)
- [ ] `pipelines/*.py` (如需管道编排)
- [ ] `configs/*.yaml` (如需配置管理)
- [ ] `prompts/*.txt` (如需 prompt 模板)

文档版本: 1.0 创建日期: 2026-02-03 作者: AI Assistant