A METHOD DETAILS

AniME decomposes the story-to-video task into hierarchical stages coordinated by a centralized **Director Agent** and executed by a set of **Specialized Agents**. Each agent A_i has a well-defined input type $\mathcal{I}_i$, output type O_i, and a local MCP toolbox $\mathcal{T}_i$. Communication between agents is performed via structured JSON messages. In this section, we detail the overall framework, the role of the Director, asset memory management, inter-agent communication, and the specialized agents that together enable long-form animation production.

A.1 Director Agent

The Director Agent serves as the central controller of AniME. It manages the global workflow, decomposes the input story into subtasks, allocates them to specialized agents, evaluates quality, and maintains the global Asset Memory Bank. Unlike traditional pipeline orchestration, the Director employs hierarchical planning with explicit dependency tracking to preserve both narrative coherence and visual consistency.

A.1.1 Workflow of the Director. Given a story script $\mathcal{R}$, the Director first segments it into scenes and shots. The segmentation uses semantic similarity across narrative units, emotional state transitions, and temporal markers (e.g., dialogue breaks, action beats). Each scene inherits a global style vector for visuals $\mathbf{s}_v$ and acoustics $\mathbf{s}_a$, determined through a style analysis model.

The Director then generates a task list T and constructs a workflow graph $\mathcal{W} = (N, E)$, where nodes represent subtasks (e.g. "render background for shot 5") and edges represent dependencies (e.g., character design precedes animation). Tasks are executed in topological order, with failed or low-quality outputs triggering localized revisions without re-running the full pipeline.

A.1.2 Asset Memory Management. The Asset Memory Bank stores canonical creative assets, ensuring consistency and reusability across the pipeline. Each table is queryable, versioned, and embedding-augmented, enabling retrieval by semantic similarity. For example, the Character table stores not only reference images but also identity embeddings, ensuring that the Animator preserves visual fidelity. Table 2 lists representative tables used in the memory.

To prevent drift, the Director enforces single-writer semantics for canonical fields (e.g., only the Character Designer can update identity vectors), while enabling multi-reader access for dependent agents. Version control is integrated, allowing rollback or branching when user-guided revisions are made.

A.2 Inter-Agent Communication Protocol

All agents communicate using structured JSON messages. Each request message contains task identifiers, task description, assets, and metadata. A typical task request message is as follows.

```
{
"id": "shot_03_v1",
"type": "storyboard",
"task": {
    "prompt": "split the given shot into segments,
    plan camera angles, and generate the keyframes. ",
"requirement": "style: should be 3d animation"},
"assets": { "style": [...], "identity": [...] },
"meta": { "version": 1, "producer": "StoryboardAgent" }
}
```

This structure supports message versioning, multimodal embedding sharing, and agent provenance tracking. Agents can query the Asset Memory with message IDs or embedding similarity, enabling flexible yet consistent coordination.

The response messages contain the generated outputs, including asset IDs, visual aids, and metadata. For example:

```
{
"id": "shot_03_v1",
"type": "storyboard",
"status": "success",
"outputs": {
    "keyframes": ["kf1.png", "kf2.png"],
    "camera_plan": {
        "angles": [30, 45],
        "transitions": ["fade", "cut"]
    }
},
"meta": { "version": 1, "producer": "StoryboardAgent" }
}
```

This response structure allows agents to validate outputs, track dependencies, and maintain a coherent workflow.

A.3 Specialized Agents and MCP Toolsets

AniME employs a set of specialized agents, each aligned to a creative stage. Each agent leverages a dedicated MCP toolset, allowing adaptive model selection based on task requirements. Below, we expand on their detailed designs.

A.3.1 Script and Storyboard Agent. The Script and Storyboard Agent transforms narrative text into a temporal sequence of shots. It performs three key steps: (1) *segmentation* of text into scenes and shots using discourse parsing and emotion tagging; (2) *camera planning*, including shot type, trajectory, and transitions; and (3) *reference retrieval*, where keyframes are generated using retrieval-augmented text-to-image models. The outputs are structured JSON shot descriptors with the corresponding visual aids.

Tool selection. This agent adaptively selects among multiple tools to generate storyboards: *text-to-image* for empty establishing shots (simple, fast, but limited layout control), *reference image generation* for maintaining character identity consistency (strong on character fidelity but cannot generate empty scenes), and *layout-guided generation* for precise multi-character or object-dense panels (accurate composition but higher computational cost). LLM-based segmentation, camera planning, and layout planning modules orchestrate the shot-level structure.

A.3.2 Character Designer. The Character Designer creates canonical character assets. Starting from textual descriptions and style vectors, it generates multi-view portraits via diffusion-based text-to-image models. Identity consistency is enforced by CLIP-based scoring against stored embeddings. Multi-view synthesis further ensures that side and back profiles match the canonical frontal view. Failure detection is applied to reject outputs with drifted identity or inconsistent clothing details.

Tool Type	Functionality	Pros	Cons
Text-to-Image	Generate images directly from text prompts, useful for establishing shots or background scenes.	• Simple interface • Good for empty-scene (establishing) shots	• Hard to control layout • Limited consistency across shots
Reference Image Generation	Generate images conditioned on reference characters to ensure identity consistency across shots.	• Ensures character consistency • Good for dialogue-focused panels	• Cannot generate empty establishing shots
Layout-Guided Image Generation	Generate images following a predefined layout or sketch, suitable for precise camera and composition control.	• Strong spatial control • Useful for complex multi-character panels	• Higher complexity • Requires layout design overhead

Table 3: Storyboard-related MCP tools example. Different generation types (text-to-image, reference image generation, layout-guided image generation) are adaptively select by the Script and Storyboard Agent according to the generation task requirements.

Tool selection. The agent mainly leverages *reference image generation* to ensure consistent character identity. *Text-to-image* is used for initial exploration or design iteration, while *multi-view synthesis* and refinement modules enforce multi-angle consistency and correct visual drift.

A.3.3 Scene Designer. The Scene Designer generates environments and backgrounds. It uses depth-guided diffusion for spatial coherence, layout-guided generation for furniture and object placement, and relighting models for temporal consistency across shots. Outputs include layered assets that can be reused across multiple scenes.

Tool selection. Scene Designer prioritizes *layout-guided generation* for precise object placement, *depth-guided image generation* for spatially coherent scenes, and relighting models to maintain temporal lighting consistency. *Text-to-image* can optionally generate broad establishing shots but provides less control over composition.

A.3.4 Animator. The Animator synthesizes motion sequences from keyframes, poses, and camera trajectories. It integrates keyframe-conditioned video diffusion with audio-driven lip synchronization. To maintain temporal coherence, optical flow guidance and motion interpolation are applied. The Animator also aligns phoneme embeddings from dialogue audio with mouth movements, ensuring expressive yet identity-preserving animation.

Tool selection. Animator mainly uses keyframe/audio/pose/camera-conditioned video generation models. In revision cycles, it may request additional layout-guided or reference image outputs from upstream agents to repair missing or inconsistent frames, but it generally does not directly invoke text-to-image tools.

A.3.5 Audio Production Agent. The Audio Production Agent manages dialogue, sound effects, and music. It employs speaker-conditioned TTS to generate character-specific voices, enriched with emotion embeddings. Background music is composed via text-to-music generation and synchronized with scene dynamics. Foley and ambient effects are retrieved from external databases or synthesized directly. An audio mixer then balances speech, music, and effects.

Tool selection. This agent selects among *speaker-conditioned TTS*, *text-to-music generation*, and *audio mixer programs*. Each tool complements others: TTS ensures identity-preserving dialogue, text-to-music adapts to scene dynamics, and the mixer balances all audio tracks for coherent output.

A.3.6 Video Editor Agent. The Video Editor integrates all assets into a coherent final video. Automated editing tools suggest cuts and transitions, while a color pipeline ensures visual consistency. Multi-pass encoding via FFmpeg produces the final distribution-ready video. Human-in-the-loop revision is supported at this stage for fine-grained adjustments.

Tool selection. Video Editor primarily relies on *transition effects*, *color pipelines*, and *FFmpeg multi-pass encoding*. While it does not generate new content, it consumes outputs from other agents to produce a professional-quality video and apply global visual adjustments.

A.3.7 Quality Evaluator Agent. The Quality Evaluator performs multimodal validation. It checks text-to-video similarity to ensure semantic alignment, identity verification to prevent drift, and audio-visual synchronization. Additionally, a vision-language model (VLM) is used for narrative consistency evaluation, ensuring that the generated sequence adheres to the intended storyline. When failures are detected, targeted revision requests are sent back to the responsible agent.

Tool selection. The evaluator uses *text-to-video similarity scoring*, *identity verification*, *AV sync checks*, and *VLM-based narrative evaluation*. Based on detected errors, it may recommend specific tool changes, such as switching storyboard generation from text-to-image to layout-guided for better spatial composition.

A.3.8 MCP Tool Selection. In AniME, the Model Context Protocol (MCP) enables specialized agents to autonomously select and invoke the most appropriate tools for their tasks based on the context provided by the Director. As an example, consider a scene description sent from the Director to the Storyboard Agent:

> *"In Ye's training room, Ye raises a blue-and-white porcelain cup with both hands, tilting his head to bring it to his mouth, while the system AI angrily try to stop him, telling him to stop."*

Upon receiving this shot description, the Script & Storyboard Agent first generate the following storyboard descriptions and choose the image generation tools. For each shot, the Storyboard Agent produces a structured JSON output, including the selected tool, prompts, references, and notes. Example:

```
{
```

(a) sceneYX01_shot_01 (b) sceneYX01_shot_02 (c) sceneYX01_shot_03

Figure 2: Storyboard keyframes for scene `sceneYX01`. Each shot shows the corresponding keyframe generated by the Storyboard Agent using selected tools (text-to-image, reference image, layout-guided generation).


```
"scene_id": "scene_YX01",
"storyboard_shots": [
  {
    "shot_id": "scene_YX01_shot_01",
    "tool": "reference_image_generation",
  "prompt": "Ye holding a blue-and-white porcelain cup,
     tilting head to drink",
   "reference_images": ["assets/char_YX_front.png"],
    "notes": "Preserves character identity"
  },
  {
    "shot_id": "scene_YX01_shot_02",
    "tool": "layout_guided_generation",
    "prompt": "System AI angrily stopping Ye",
    "layout_bboxes": [
          {
          "object": "Ye Xuan",
          "bbox": [100, 300, 400, 900],
      "notes": "Left side of the frame; full body visible"
          },
          {
          "object": "System AI",
          "bbox": [600, 350, 900, 850],
         "notes": "Right side of the frame; mid-body
           visible, pointing gesture"
          }
      ],
    "notes": "Precise spatial relationship between characters"
    },
    {
      "shot_id": "scene_YX01_shot_03",
      "tool": "layout_guided_generation",
    "prompt": "Close-up of Ye's facial expression reacting to AI",
    "layout": "assets/layouts/scene_YX01_shot04.json",
     "notes": "Detailed facial expression and posture"
    }
  ]
}
```


Through MCP, the Storyboard Agent adapts its tool selection based on the task requirements and scene context, ensuring efficient and high-quality storyboard generation without manual intervention. The Director's role remains providing the high-level scene context, while MCP governs how the specialized agent internally orchestrates its tools.