AI V-Tuber システム開発のための反復型開発計画書に関する技術的評価報告書

I. 序論

A. AI V-Tuber システムプロジェクト概要

本報告書は、自律的または半自律的なパフォーマンスが可能な AI V-Tuber システムの開発を目指すプロジェクト計画を技術的な観点から評価するものである。提示された計画は、反復型開発アプローチを採用しており、これは本プロジェクトのような複雑性とリスクを伴うシステム開発において、適切な方法論であると評価できる [1]。このアプローチは、迅速なフィードバックループの確立、変化への適応性の向上といった利点を提供する [1]。

計画されているシステムは、主要コンポーネントとして、フロントエンドのレンダリングを担当する vtube-stage、バックエンドの制御ハブとなる stage-director、AI ロジックを担う vtuber-behavior-engine、そして OBS Studio との連携機能から構成される。

B. 主要技術スタック

本プロジェクトで採用が計画されている主要技術は以下の通りである。

- **フロントエンド:** React/Vite/Three.js
- **VRM 処理:** @pixiv/three-vrm
- **バックエンド:** FastAPI
- **通信:** WebSockets
- **AI エージェント:** Google Agent Development Kit (ADK)
- **ツール連携プロトコル:** Model Context Protocol (MCP)
- **OBS 連携:** obs-websocket

これらの技術選定は、現代的で高性能なフレームワーク（例: FastAPI [5]）と、特定のドメインに特化したライブラリ（例: @pixiv/three-vrm [8], ADK [9]）を活用するという点で、プロジェクトの目標達成に向けた合理的な判断であると言える。各技術は、提案されたアーキテクチャ内で明確な役割を担うことが期待される。

C. 本報告書の目的と構成

本報告書の目的は、提示された反復型開発計画を専門的な技術的見地からレビューし、関連する最新の研究成果やベストプラクティスを踏まえた上で、計画の強化に資する洞察と提言を提供することにある [1]。報告書の構成は、ユーザーが提示したイテレーション計画に沿って進められる。

II. イテレーション 1: コア舞台の構築 (vtube-stage)

A. 目標

このイテレーションの目標は、システムの視覚的基盤を確立することである。具体的には、Web ブラウザ上に VRM モデルをレンダリングし、基本的な手動制御（表情、ポーズ）を可能にすることを目指す。

B. 技術的実装 **(vtube-stage)**

1. プロジェクトセットアップ **(React/Vite/Three.js):**
    - React と Vite の組み合わせは、モダンでパフォーマンスの高いフロントエンド開発環境を提供する。Vite は高速なビルドと HMR (Hot Module Replacement) を実現し、開発効率を高める。
    - Three.js は、Web ブラウザ上で 3D グラフィックスを描画するための標準的なライブラリであり、React アプリケーション内での統合が必要となる。react-three-fiber [24] のようなライブラリ（計画には明記されていないが、一般的な実践方法）を使用すると、Three.js のシーン管理を宣言的に記述でき、React との親和性が高まる。
2. **VRM** の読み込みとレンダリング **(@pixiv/three-vrm):**
    - VRM は glTF 形式をベースとした 3D アバター用のフォーマットであり [27]、その仕様を扱うためには @pixiv/three-vrm ライブラリが不可欠である。これは Three.js の GLTFLoader のプラグインとして機能する [8]。
    - 基本的な読み込み手順は、GLTFLoader インスタンスを作成し、VRMLoaderPlugin を登録 (loader.register) し、loader.load() を呼び出して VRM ファイルの URL を渡すことである [8]。ロード完了後、gltf.userData.vrm を介して VRM インスタンスにアクセスし、シーンに追加する [8]。
    - このライブラリには、使用方法を示すサンプルコード ([8]) やドキュメント ([8]) が存在するが、詳細な API ドキュメントが不足している可能性も指摘されている [8]。
    - WebGPU レンダラーとの互換性も提供されているが、特定の Material タイプ (MToonNodeMaterial) や新しい Three.js バージョンが必要となる場合がある点に留意が必要である。
3. 手動による表情とポーズの制御**:**
    - VRM 仕様は、表情 (BlendShape) とポーズ (Humanoid ボーン) の標準化された制御方法を定義している [27]。
    - @pixiv/three-vrm は、これらの制御インターフェースを提供する。表情制御には vrm.expressionManager (または旧バージョン/別文脈では vrm.blendShapeProxy [41])、ポーズ制御には vrm.humanoid を使用する [30]。
    - 具体的な制御例としては、vrm.expressionManager.setValue(expressionName, value) [43] や vrm.humanoid.getNormalizedBoneNode(boneName) [43] を介したボーン操作が考えられる。関連するサンプルコードも存在する [32]。
    - 制御対象となる VRM モデル固有の BlendShape プリセット名（例: happy, angry, sad, relaxed, surprised, aa, ih, ou, ee, oh, blink [37]）やボーン名を正確に把握する

ことが重要である [41]。

4. **OBS ブラウザソースキャプチャ:**
    - このステップは比較的単純で、OBS Studio のブラウザソース機能を使用して vtube-stage アプリケーションがレンダリングされているウィンドウをキャプチャするように設定する。
    - 必要に応じて背景透過設定や、キャプチャによるパフォーマンスへの影響を考慮する必要がある。

C. 完了条件

- Web ブラウザ上に VRM モデルが表示されること。
- 手動 UI を介して VRM モデルの表情とポーズが変更できること。
- OBS Studio でブラウザソースとしてキャプチャできること。

D. 技術的考察と推奨事項

@pixiv/three-vrm ライブラリは強力である一方、ドキュメントが断片的である可能性があり [8]、成功のためにはサンプルコードの探索 [24] や、場合によってはソースコードやコミュニティでの議論 [33] の確認が不可欠となる。ユーザー計画は @pixiv/three-vrm を指定しているが、[8] によれば主要な README には表情/ポーズ/視線制御 API の詳細がなく、サンプルやガイドを参照するよう指示されている。他の情報源もライブラリの使用 [8] やサンプルの存在 [24] を裏付けている。したがって、このイテレーションにおける主要な課題と推奨事項は、主要ドキュメントを超えた積極的な情報収集である。

また、VRM 仕様自体も進化しており（例: VRM 0.x と 1.0 での座標系、T-Pose 要件、表情名の変更 [27]）、使用する VRM モデルとライブラリのバージョン間の互換性を確認し、仕様バージョンによる違いを理解することが重要である。ライブラリ自体もバージョン管理されており ([31] は v0.6.8 を示しているが、現在は v3+ [8])、モデルバージョンとライブラリバージョンの互換性問題や、仕様バージョンに基づく前提の違いは潜在的なリスクとなる。実装時にはこれらに明示的に注意を払う必要がある。

推奨事項: 対象となる VRM モデルの BlendShape 名とボーン名を正確に理解することを優先する。可能であれば、Blender と VRM アドオン [45] などのツールを使用して事前にモデルを検査する。モデルの読み込みと機能アクセスに関する堅牢なエラーハンドリングを実装する。

III. イテレーション 2: 通信経路の確立 (stage-director)

A. 目標

バックエンド (stage-director) とフロントエンド (vtube-stage) 間で WebSocket 通信を確立し、制御コマンドを送受信できるようにする。

B. 技術的実装

1. **stage-director** セットアップ **(FastAPI):**
   - FastAPI の選択は、その高いパフォーマンス、WebSocket に不可欠な非同期サポート、自動生成される API ドキュメント、Pydantic によるデータ検証といった利点から正当化される [5]。
   - FastAPI は ASGI (Asynchronous Server Gateway Interface) フレームワークであり、Uvicorn のような ASGI サーバーと組み合わせて使用される [6]。
2. **WebSocket** 実装 **(FastAPI):**
   - @app.websocket("/ws/{client_id}") デコレータを使用して WebSocket エンドポイントを定義する [46]。
   - FastAPI/Starlette が提供する WebSocket オブジェクトを使用する [46]。
   - 接続ライフサイクル管理: await websocket.accept() で接続を受け入れる [46]。
   - メッセージ受信: await websocket.receive_text(), receive_bytes(), receive_json() [46]。
   - メッセージ送信: await websocket.send_text(), send_bytes(), send_json() [46]。
   - 切断処理: WebSocketDisconnect 例外を捕捉する [46]。
   - 将来的な認証や依存性注入のために、WebSocket ルートで Depends を使用することも可能である [46]。
3. **API** 定義 **(**制御コマンド**):**
   - 明確な JSON ベースのコマンド構造を定義する（例: {"command": "setCharacterExpression", "payload": {"name": "happy", "value": 0.8}}）。
   - stage-director の WebSocket ハンドラ内にコマンド解析ロジックを実装する。
4. **vtube-stage** クライアント実装**:**
   - ネイティブのブラウザ WebSocket API [46] またはライブラリを使用する。この段階で必要なシンプルさを考えると、ネイティブ API で十分である可能性が高い。
   - 接続ロジック (new WebSocket("ws://...")）、onopen, onmessage, onclose, onerror ハンドラを実装する [46]。
   - stage-director から受信したコマンドを解析し、対応する VRM 更新（イテレーション 1 のロジックを使用）をトリガーする。
5. 手動 **UI** のリファクタリング**:**
   - イテレーション 1 で作成した UI を変更し、VRM を直接操作する代わりに WebSocket を介して stage-director にコマンドを送信するようにする。stage-director はそのコマンドを vtube-stage にブロードキャスト/送信する（初期段階でクライアントが 1 つと仮定する場合は直接送信も可能）。

C. 完了条件

- stage-director から送信されたコマンド（初期はテストエンドポイントや修正された UI からトリガー）に基づいて、vtube-stage 上の VRM の状態が正常に変更されること。

D. 技術的考察と推奨事項

FastAPI は WebSocket のセットアップを簡素化するが [46]、複数のクライアント、状態、および潜在的なブロードキャストロジックの管理は、慎重なアーキテクチャ設計を必要とする。複雑さが増すにつれて（例: 複数のステージ、ディレクター UI）、ConnectionManager パターン [46] や encode/broadcaster [46] のような外部ライブラリの使用が重要になる。ユーザー計画は基本的な通信から始まるが、[46] は複数のクライアントとブロードキャストのために ConnectionManager を明確に推奨している。イテレーション 2 では vtube-stage クライアントが 1 つだけかもしれないが、アーキテクチャは将来のニーズ（例: 複数のキャラクター、ディレクターインターフェース）を予測すべきである。したがって、初期には簡略化されていても、堅牢な接続管理戦略を早期に採用することが、スケーラビリティのための重要な考慮事項となる。

FastAPI が使用するネイティブ WebSocket は、低遅延と効率性を提供する [5]。Socket.IO のようなライブラリは、ルーム、自動再接続、フォールバックなどの機能を追加するが、独自のプロトコルとオーバーヘッドを導入する [51]。この特定のサーバーからブラウザへの制御シナリオでは、両端が制御下にあるため、FastAPI を介したネイティブ WebSocket の選択は適切であり、不要な依存関係を回避できる [52]。ユーザー計画は FastAPI での直接的な WebSocket 使用を示唆しており、調査結果は WebSocket と Socket.IO を比較している [51]。Socket.IO の利点（フォールバック、容易なブロードキャスト）は、FastAPI の直接サポート [46] を考慮すると、ネイティブ WebSocket が提供する低遅延と標準プロトコル遵守よりも重要度が低い。これは計画されたアプローチを検証するが、トレードオフを浮き彫りにする。

推奨事項: 早期に堅牢でバージョン管理されたコマンドプロトコルを定義する。クライアントとサーバーの両方で、WebSocket イベント（接続、切断、エラー）に対する基本的なエラーハンドリングとロギングを実装する。最初から ConnectionManager パターン [46] の導入を検討する。

IV. イテレーション 3: 単一 AI の接続 (vtuber-behavior-engine)

A. 目標

単一の AI エージェント（ADK で構築）を接続し、stage-director を介して VRM の基本的な表情・ポーズを制御できるようにする。

B. 技術的実装

1. **vtuber-behavior-engine セットアップ (ADK):**
    - ADK を、エージェント開発のためのコードファーストフレームワークとして導入する [9]。Google エコシステムとの強力な統合が特徴である [9]。
    - プロジェクト設定の詳細: Python 環境、pip install google-adk [66]、API キー用の環境変数 (.env) [66]。
    - 選択したモデル（例: Gemini [9]）を使用して単純な対話生成が可能な基本的な

LlmAgent [9] を実装する。その instruction と name を定義する。

2. **ADK-stage-director 通信 (MCPToolset):**
    - Model Context Protocol (MCP) を、AI アプリケーションがツールやデータと対話するための標準プロトコルとして説明する [69]。
    - ADK の MCPToolset を、ADK エージェントが MCP クライアントとして機能するためのメカニズムとして導入する [9]。
    - 技術的判断点: **MCP** サーバーの配置場所
        - オプション *1 (MCP* サーバーを *stage-director* 内に配置*):* stage-director が MCP サーバー（mcp-server-python ライブラリを使用 [74]）を実装し、setCharacterExpression、setCharacterPose などのツールを公開する。ADK エージェントは SseServerParams [9] を使用して MCPToolset.from_server でこのサーバーに接続する。これにより、ステージ制御は stage-director に一元化される。
        - オプション *2 (ADK* 内の直接 *WebSocket* ツール*):* vtuber-behavior-engine 内にカスタム ADK FunctionTool [10] を作成し、これが stage-director の WebSocket エンドポイント（イテレーション 2 で作成）に直接接続してコマンドを送信する。これにより、初期段階で別の MCP サーバーを実装する手間は省けるが、後々の標準化や柔軟性の面で劣る可能性がある。
        - 推奨: オプション 1 は ADK/MCP エコシステム [9] との整合性が高く、他のツールが予想される場合にはスケーラブルである可能性が高い。[74] はローカルサーバー用の StdioServerParameters の例を提供しているが、stage-director へのリモート接続には SseServerParams が必要である。[9] (特に adk_mcp_server.py の例) は ADK ツールを MCP サーバーとしてラップする方法を示しており、これは逆方向だがサーバー側ライブラリの使用法を示している。[74] は MCP サーバー スクリプトの作成を指している。
    - ADK エージェント内で MCPToolset.from_server() を呼び出し、選択した MCP サーバー（おそらく stage-director 内）に接続する [9]。適切な接続クリーンアップのために exit_stack を処理する [9]。
    - 取得したツールを LlmAgent の tools パラメータに渡す [9]。
3. 感情分析と **BlendShape** マッピング**:**
    - ADK エージェント内（または専用のコールバック/ツール内）で、LLM の応答テキストを分析して感情的な内容を抽出するロジックを実装する。
        - 感情分析には Python ライブラリを活用する（例: Hugging Face Transformers [89], NLTK [90], TextBlob [91], VADER [91], spaCy [92]）。関連チュートリアルも存在する [90]。
        - 利用可能/適用可能であれば、EmoLLMs のようなファインチューニングされたモデルを検討する [95]。研究によれば、LLM はこの種のタスクに効果的である [95]。
        - 検出された感情（例: [37] に基づく "happy", "sad", "angry"）から VRM BlendShape 名と目標ウェイト（0.0 から 1.0）へのマッピングを定義する。マッピ

ング技術に関する研究も存在する [41]。
    - エージェントは MCP ツール（例: setCharacterExpression）を使用して、マッピングされた BlendShape 値を stage-director に送信する。

C. 完了条件

- AI エージェントが対話を生成し、その感情を分析し、対応する VRM の表情変化を stage-director と WebSocket 通信を介してトリガーすること。

D. 技術的考察と推奨事項

ADK を stage-director のような外部システムと MCP 経由で統合するには、MCP クライアント (MCPToolset in ADK [9]) と MCP サーバー (likely in stage-director [9]) の両方を慎重に設定する必要がある。MCP の概念（リソース、ツール、プロンプト [69]）と接続方法（Stdio vs. SSE [9]）を理解することが不可欠である。ユーザー計画は ADK MCPToolset の使用を指定している。[74] と [9] はその使用法を詳述し、StdioServerParameters または SseServerParams を介して MCP サーバーに接続する。[69] は MCP 自体を説明している。これらを統合するには、MCP プロトコルのクライアント側 (ADK) とサーバー側 (stage-director) の両方を実装する必要があり、これは直接的な WebSocket 呼び出しと比較して複雑さを増すが、標準化の利点を提供する [70]。この複雑さは重要な考慮事項であり、潜在的な障害となる。

テキストの感情を離散的な BlendShape にマッピングすることは自明ではない。単純なキーワードマッチングや基本的な感情極性分析では不十分な場合がある。より洗練された NLP モデル [89] が、ニュアンスのある表現のためには必要となる。マッピング自体（どの感情がどの BlendShape に、どの程度のウェイトで対応するか）は主観的であり、V-Tuber の認識されるパーソナリティにとって極めて重要である [38]。ユーザー計画は感情分析とマッピングを要求している。研究は様々な NLP 技術 [90] や LLM ベースのアプローチ [95] を強調している。また、顔追跡 [98] やテキスト/画像 [100] からのブレンドシェイプへのマッピングについても議論されており、入力分析と出力表現の間の関連性を強調している [38]。課題は翻訳層にあり、テキストから導き出された潜在的に複雑な感情状態を、限られた VRM ブレンドシェイプの重みセット [41] に変換する必要がある。このマッピングには慎重な設計と調整が必要である。

推奨事項: シンプルな感情分類モデル（例: Hugging Face Transformers [89] を使用）と基本的なマッピング（例: 検出された "joy" -> "happy" BlendShape を 0.8 に設定）から始める。視覚的な結果に基づいて、モデルとマッピングを反復的に改良する。ADK/MCP の原則 [9] とのアーキテクチャ整合性を高めるために、MCP サーバーを stage-director 内に実装する。MCP 通信のための堅牢なエラーハンドリングを確保する。

提案表: 感情-VRM BlendShape マッピング戦略 (セクション IV.B)

アプローチ	利点	欠点	ライブラリ/モデル例 ()	VRM ターゲット例 ()
キーワードマッチング	実装が容易	文脈を無視、ニュアンス欠如	(カスタム実装)	プリセット表情
感情極性分析 (Positive/Negative/Neutral)	ライブラリ豊富 (TextBlob, VADER)	感情の種類が限定的	TextBlob, VADER	happy/sad/neutral プリセット
多クラス感情分類 (Joy, Sadness, Anger 等)	より詳細な感情表現が可能	モデル精度、データセット依存性	Hugging Face Transformers (j-hartmann/emotion-english-distilroberta-base [93]), EmoLLMs [95]	各感情プリセット
次元感情モデル (Valence-Arousal 等)	連続的な感情状態を表現可能	マッピングが複雑、対応するVRM表現の設計難易度高	(研究レベル)	複数BlendShapeの組み合わせ

理由: この表は、このイテレーションの中核タスクである感情マッピング機能の実装に関する技術的な選択肢をユーザーが理解するのに役立つ。異なるアプローチの複雑さと潜在的な効果を比較し、このイテレーションの主要な課題に直接対処する。感情分析と VRM 表現に関連する複数の情報源 ([98]–[96]) からの情報を統合している。

V. イテレーション 4: マルチエージェント協調

A. 目標

舞台制御エージェントと複数のキャラクターエージェントを ADK を使用して実装し、協調して対話と舞台演出（画像表示など）を行えるようにする。

B. 技術的実装

1. マルチエージェントアーキテクチャ設計 **(ADK):**
    - ADK が持つマルチエージェントシステムへのネイティブサポートを活用する [9]。
    - 階層構造を設計する [9]: ルートとなる「舞台制御エージェント」が複数の「キャラクターエージェント」をオーケストレーションする。
    - 舞台制御エージェントの実装: 動的な制御が必要な場合は LlmAgent、相互作用パターンが予測可能な場合（例: ラウンドロビン方式のターン）は SequentialAgent や LoopAgent [9] を検討する。役割には、話題管理、ターン制御、舞台効果のトリガーな

どが含まれる。
    - キャラクターエージェントの実装: 複数の LlmAgent インスタンスを作成し、それぞれに固有の name、description、およびペルソナを定義する instruction を設定する [9]。
2. エージェント協調メカニズム **(ADK):**
    - ADK が提供するエージェント間通信および制御メカニズムを利用する [9]:
        - 共有セッション状態 **(session.state):** 同じ呼び出しコンテキスト内で動作するエージェント間で受動的にデータを共有する（例: 現在の話題、キャラクターの気分）[9]。
        - **LLM** による委譲 **(transfer_to_agent):** 舞台制御エージェント（LlmAgent の場合）が、コンテキストに基づいて特定のキャラクターエージェントにタスクやターンを動的にルーティングできるようにする [9]。LLM が情報に基づいた決定を下すためには、明確なエージェントの description が必要となる [9]。
        - 明示的な呼び出し **(AgentTool):** キャラクターエージェントを AgentTool でラップし、舞台制御エージェントが呼び出し可能なツールとして扱えるようにする [9]。同期的で制御された相互作用を提供する [9]。
    - 選択した協調ロジックを舞台制御エージェント内に実装する。
3. マルチエージェント対話生成**:**
    - マルチエージェント対話における課題に対処する: コンテキスト維持、ペルソナの一貫性、一貫したターンテーキング [110]。
    - エージェントのプロンプト (instruction) を改良し、他のエージェントの存在、対話履歴（ADK の Session と InvocationContext [9] によって管理）、および全体的な目標に対する認識を含める。
    - 必要に応じて、長いコンテキストを処理するために要約技術を検討する [114]。
4. 舞台効果コマンド**:**
    - stage-director API（およびイテレーション 3 のオプション 1 を使用している場合は対応する MCP ツール）に、舞台効果（例: displayImage, showWebpage）のための新しいコマンドを定義する。
    - vtube-stage にこれらの要素を表示するためのハンドラ（例: 画像や iframe のオーバーレイ表示）を実装する。
    - エージェント（おそらく舞台制御エージェント）が会話の流れに基づいてこれらのコマンドをトリガーできるようにする。

C. 完了条件

- 複数の AI キャラクターが、制御エージェントによって調整され、それぞれの役割に応じたペルソナで一貫した対話を行うこと。
- 会話の流れに合わせて、関連する舞台効果（画像表示など）が自動的にトリガーされること。

D. 技術的考察と推奨事項

ADK はマルチエージェントシステム向けに明示的に設計されており [9]、特定の協調プリミティブを提供している [9]。LangGraph のようなフレームワークはグラフベースの状態管理に焦点を当て [111]、CrewAI はよりシンプルな抽象化でロールベースのコラボレーションを強調している [65]。ADK のアプローチは、計画されている階層的な制御構造に適しているように思われるが、代替案を理解することは ADK の強み（例: Google Cloud との緊密な統合 [65]）と潜在的な弱み（例: 開発者体験の複雑さ [76]）を文脈化するのに役立つ。ユーザー計画はマルチエージェント領域に進む。ADK が選択されたツールである。調査は ADK、LangGraph、CrewAI を比較している [65]。これらの比較を理解することで、ADK の設計哲学（コードファースト、階層的、Google 中心）と、それが LangGraph のグラフ中心や CrewAI のロールベースのシンプルさとどのように対照的であるかが明らかになる。これは、この潜在的に複雑な階層システムの選択肢として ADK を検証するのに役立つ一方で、他のフレームワークが異なる利点を提供する可能性がある領域（例: CrewAI のプロトタイピングの容易さ [111]）を認識させる。

一貫性があり、魅力的なマルチエージェント会話を生成することは困難である [110]。成功は、プロンプトエンジニアリング [110]、効果的なコンテキスト管理（ADK の Session 状態 [9]）、および潜在的に洗練された協調ロジック（例: ターン順序の管理、話題の逸脱防止 [114]）に大きく依存する。エージェントの数やコミュニケーションパラダイムもパフォーマンスに影響を与える [114]。目標はマルチエージェント対話である。研究 () はその複雑さを詳述している：一貫性、ペルソナ、コンテキストの維持、そして問題の逸脱のような問題の回避 [114]。ADK はセッション状態やエージェント転送のようなツールを提供する [9] が、中核的な課題はインタラクションロジックとプロンプトを効果的に設計することにある。これは、単に複数のエージェントをインスタンス化するだけでは不十分であり、それらのインタラクションプロトコルの慎重な設計が最も重要であることを示唆している。

推奨事項**:** シンプルな協調メカニズム（例: session.state を使用して舞台制御エージェントが管理するラウンドロビン方式のターン）から始める。各キャラクターエージェントに対して、そのペルソナと相互作用ルールを明確に定義する堅牢なプロンプトを作成することに集中する。舞台効果のトリガーのような重要で同期的なアクションには AgentTool [9] を使用する。会話の質をテストするために、ADK の評価機能 [9] を早期に活用する。

提案表**: ADK** マルチエージェント協調メカニズム (セクション V.B)

メカニズム	説明 **(**に基づく**)**	通信タイプ	ユースケース例	利点	欠点
session.state	同じ呼び出しコンテキスト内	受動的/非同期	話題共有、エージェント状	シンプル、パイプラインや	同期や直接的な制御には不

	のエージェント間で辞書ベースの状態を共有		態（気分など）の伝達	ループでのデータ渡しに便利	向き
transfer_to_agent	LLM が判断し、階層内の別のエージェントに実行を委譲（LlmAgent の機能）	動的/LLM駆動	タスクの委譲、会話のルーティング	柔軟性が高い、LLM の判断に基づいた動的なフローが可能	LLM の解釈に依存、ターゲットエージェントの description が重要
AgentTool	あるエージェントを別の LlmAgent から呼び出し可能なツールとしてラップ	明示的/同期的	ユーティリティエージェントの呼び出し、制御された連携	同期的な制御が可能、他のツールと同様に扱える	呼び出し側のフローをブロックする、ラップする手間

理由: この表は、イテレーション 4 の中核タスクであるマルチエージェント協調の実装に直接対応する。ADK が提供する特定のツール [9] を明確に概説し、ユーザーが階層構造内で異なる相互作用ニーズに適したメカニズムを選択するのに役立つ。

VI. イテレーション 5: OBS 連携

A. 目標

AI システム（stage-director または vtuber-behavior-engine）が obs-websocket を介して OBS Studio のシーン切り替えやソース表示/非表示を制御できるようにする。

B. 技術的実装

1. **obs-websocket 連携:**
   - OBS Studio で obs-websocket プラグインが有効になっていることを確認する（通常、OBS 28 以降ではデフォルト [123]）。ポート番号とパスワードを設定する [123]。
   - Python クライアントライブラリを選択する: obsws-python [123] または simpleobsws (非同期) [123]。ユーザー計画では非同期要件が明記されていないため、制御ロジックが同期的であれば obsws-python の方がシンプルかもしれない。
   - 選択したライブラリをインストールする (pip install obsws-python [124])。
2. 制御ロジック実装:
   - 技術的判断点: OBS 制御ロジックの配置場所
     - オプション *1 (stage-director):* stage-director が OBS に接続する。ADK エージェントは高レベルのコマンド（例: "switch_to_gaming_scene"）を

stage-director に送信し（MCP ツールまたは WebSocket 経由）、stage-director がこれを特定の obs-websocket リクエストに変換する。OBS との対話を AI コアロジックから分離できる。
    - オプション *2 (vtuber-behavior-engine):* ADK エージェント（おそらく舞台制御エージェント）が直接 obs-websocket クライアントライブラリを使用する（潜在的に ADK FunctionTool でラップ）。より緊密な統合だが、関心事が混在する。
    - 推奨: オプション 1 (stage-director) は、関心事の分離が優れている。stage-director は、すべての外部ステージ/ストリーミングハードウェア制御の中心ハブとして機能する。
  - 選択したコンポーネント（おそらく stage-director）に、クライアントライブラリを使用して接続ロジックを実装する（例: obsws-python の場合 obs.ReqClient(host, port, password) [124]）。
  - 特定のリクエストを送信する関数を実装する:
    - シーン切り替え: SetCurrentProgramScene リクエストを使用する [123]。obsws-python での例: client.set_current_program_scene(scene_name) [124]。obs-cli (内部で obsws-python を使用) での例: obs-cli scene switch "SceneName" [126]。
    - ソース表示/非表示: SetSceneItemEnabled リクエストを使用する [123]。sceneName、sceneItemId (または潜在的に sceneItemName)、および sceneItemEnabled (boolean) が必要 [123]。obs-cli での例: obs-cli item show --scene "SceneName" "ItemName" [126]。直接的な obsws-python の使用には、対応するメソッド（例: set_scene_item_enabled(...)）の呼び出しが含まれる。
3. **AI 駆動トリガー:**
  - AI エージェント内で OBS アクションをトリガーする条件（例: 会話のトピック、検出された感情、特定のキーワード、明示的なコマンドに基づく）を定義する。
  - エージェントは高レベルのコマンド（例: "show_alert_overlay"）を OBS 接続を管理するコンポーネント（おそらく stage-director）に送信する。

C. 完了条件

- AI V-Tuber のアクションや対話コンテキストに応じて、OBS のシーンやソースの表示/非表示が自動的に変化すること。

D. 技術的考察と推奨事項

obs-websocket は OBS に対する詳細な制御を提供する [123]。主な課題は、高レベルの AI の意図（例:「興奮を示す」）を特定の OBS アクション（例:「興奮シーン」に切り替え、「キラキラオーバーレイ」ソースを有効にする）にマッピングすることである。これには、AI と OBS 制御ロジックの間に明確に定義された抽象化レイヤーが必要となる。ユーザー計画は AI の決定に基づいて OBS を制御することである。[123] は obs-websocket プロトコルを詳述し、[123] はクライ

アントライブラリを詳述している。プロトコルは低レベル（特定のシーン、アイテム、プロパティ）で動作する。AI はより高い抽象レベル（トピック、感情、意図）で動作する。このギャップを埋めるには、中間的なコマンドレイヤー（例: AI が "show_gameplay" を要求し、stage-director がこれを SetCurrentProgramScene("Gameplay") と SetSceneItemEnabled("Webcam Overlay", True) に変換する）を設計する必要がある。この抽象化は保守性のために不可欠である。

OBS のような外部ハードウェアの制御は、潜在的な障害（OBS が実行されていない、不正なシーン/ソース名、ネットワークの問題）を引き起こす可能性がある。obs-websocket クライアント呼び出しにおける堅牢なエラーハンドリングが不可欠である。さらに、AI の OBS 状態の理解が実際の状態と一致することを保証することは困難な場合があり、必要に応じて OBS 状態を照会するメカニズム（例: GetSceneList [130], GetSceneItemList）を検討する必要がある。イテレーション 5 では外部ハードウェア制御が追加される。これは本質的に、純粋なソフトウェアインタラクションには存在しない新しい障害点を導入する。obs-websocket ライブラリ ([124]) はエラーを報告する方法を持っている可能性が高いが、アプリケーションロジックはそれらを適切に処理する必要がある（例: エラーをログに記録し、ユーザー/AI に通知し、再試行を試みる）。さらに、AI が想定される OBS 状態に基づいて決定を下す場合（例: 現在シーン A にいると仮定して「シーン B に切り替える」）、不一致は予期しない動作を引き起こす可能性がある。これは、慎重な状態管理または obs-websocket API 呼び出しによる定期的な状態クエリの必要性を示唆している。

推奨事項**:** OBS 制御ロジックを stage-director 内に実装する。AI が要求できる明確な高レベル OBS コマンドセットを定義する。stage-director アーキテクチャで非同期が厳密に要求されない限り、同期的なシンプルさのために obsws-python を使用する。すべての obs-websocket 呼び出しを try-except ブロックでラップする。シンプルなトリガー（例: 対話中の特定のコマンドフレーズ）から始め、徐々によりコンテキスト認識型のトリガーを追加する。

提案表**: V-Tuber** 制御のための主要 **obs-websocket** リクエスト (セクション VI.B)

アクション	**obs-websocket** リクエスト名 **()**	主要パラメータ **()**	**obsws-python** 呼び出し例 **()**	ユースケース例
シーン切り替え	SetCurrentProgramScene	sceneName (string, required)	client.set_current_program_scene(scene_name)	会話トピックに応じてゲーム画面や雑談画面に切り替え
ソース表示	SetSceneItemEn	sceneName,	client.set_scene	特定の感情表現

	abled	sceneItemId or sceneItemName, sceneItemEnabled=true	_item_enabled(...)	時にエフェクトオーバーレイを表示
ソース非表示	SetSceneItemEnabled	sceneName, sceneItemId or sceneItemName, sceneItemEnabled=false	client.set_scene_item_enabled(...)	画面共有終了時に共有ウィンドウソースを非表示
録画開始/停止	StartRecord, StopRecord	(なし)	client.start_record(), client.stop_record()	特定のセグメントやイベントの自動録画
配信開始/停止	StartStream, StopStream	(なし)	client.start_stream(), client.stop_stream()	スケジュールに基づく自動配信開始/終了

理由: この表は、イテレーション 5 の中核機能実装に必要な特定の API 呼び出しのクイックリファレンスを提供する。V-Tuber アクションを基盤となるプロトコル [123] および可能性の高い Python ライブラリ [124] に直接マッピングする。

VII. イテレーション 6 以降: 改良と高度化

A. 目標

システム全体の品質、パフォーマンス、および機能セットを向上させる。

B. 主要な改善領域

1. リアルタイム性能と同期精度:
   - 潜在的なボトルネック(ネットワーク遅延、AI 推論時間、レンダリング)を分析する。
   - WebSocket 通信を最適化する(例: 該当する場合はバイナリ形式、メッセージバッチ処理)。
   - AI エージェントの応答時間を改善する(モデル選択、プロンプト最適化)。
   - フロントエンドのレンダリングパフォーマンスを向上させる(Three.js の最適化)。
2. エラーハンドリング:
   - すべてのコンポーネント(フロントエンド、バックエンド、AI、OBS 制御)にわたる包括的なエラーハンドリングを実装する。

    - 堅牢なロギングとモニタリングを追加する。
    - コンポーネント障害時のフォールバック動作を定義する。
3. **AI** の高度化**:**
    - 対話能力の強化: コンテキスト追跡の改善、長期記憶の実装（ADK はメモリモジュールを提供 [10]）、より高度な対話管理戦略の探求 [110]。
    - 感情分析とマッピングを改良し、よりニュアンスのある表現を実現する [38]。
    - エージェントのプランニングとツール使用を改善する [110]。
4. **VRM** アニメーション**:**
    - AI によってトリガーされる既成の VRM アニメーション（.vrma ファイルなど）を統合する [27]。サンプルでは Mixamo アニメーションの読み込みが示されている [32]。
    - AI の状態に基づいたプロシージャルアニメーション技術を探求する（例: 微妙なアイドル動作、ジェスチャー生成）。
5. ユーザー設定**:**
    - エージェントのペルソナ、API キー、OBS 接続詳細、VRM モデル選択などを設定するための UI を開発する。
6. テスト**:**
    - 統合テスト**:** コンポーネント間の相互作用（vtube-stage, stage-director, vtuber-behavior-engine, OBS）を検証する。
    - パフォーマンス テスト**:** 負荷状況下での応答時間、フレームレート、リソース使用量を測定する。
    - ユーザビリティ テスト**:** V-Tuber のパフォーマンスやユーザー設定インターフェースに関するフィードバックを収集する。
    - エージェント評価 **(ADK):** ADK の評価スイート [9] を使用して、テストケースに対するエージェントの応答品質とタスク完了を体系的に評価する。
7. デプロイ準備**:**
    - アプリケーションをコンテナ化する (Docker) [9]。
    - デプロイターゲットを選択する（例: ローカルマシン、Web コンポーネント用の Cloud Run [9]、ADK エージェント用の Vertex AI Agent Engine [9]）。
    - 環境変数とシークレット管理を設定する。

C. 完了条件

- 定義された改善目標が達成されること。
- システムの安定性とパフォーマンスが向上すること。

D. 技術的考察と推奨事項

WebSocket、AI モデル、ハードウェア制御を含むマルチコンポーネントシステムのデプロイは、慎重な計画を必要とする。コンテナ化 [9] は依存関係管理を簡素化するが、オーケストレーション、ネットワーキング（特に WebSocket と OBS 制御）、およびスケーリングを考慮する必要がある。Cloud Run や Agent Engine のような Google Cloud オプションはマネージドソ

リューションを提供するが、その詳細を理解する必要がある。ユーザー計画は洗練とデプロイで最高潮に達する。[9] は、コンテナ化と Cloud Run / Agent Engine との統合による ADK のデプロイ準備を明示的に言及している。これは、デプロイメントが後付けではなく、選択されたフレームワーク (ADK) に組み込まれた機能であることを強調している。しかし、システム全体（React フロントエンド、FastAPI バックエンド、ADK エンジン、OBS 接続）をデプロイするには、これらの潜在的にコンテナ化された部分をオーケストレーションし、それらのネットワーク通信を管理し、スケーラビリティを確保する必要があり、これは運用上の複雑さを大幅に増加させる。

反復的な性質は初期開発後に停止すべきではない。特に AI コンポーネント（ADK の評価ツールを使用 [9]）と全体的なシステムパフォーマンスに対する継続的な評価は、長期的な品質と適応にとって不可欠である。ユーザーフィードバックも引き続き重要である [2]。イテレーション 6+ は改善に焦点を当てている。反復開発方法論自体が継続的な改善を強調している [1]。ADK は特に評価機能を含んでいる [9]。これらを組み合わせると、「イテレーション 6+」フェーズは単なるバグ修正ではなく、特に適応的な AI コンポーネントについて、監視、評価（自動化とユーザーベースの両方）、およびシステムの改善のための継続的なプロセスを確立することを示唆している。

推奨事項: ボトルネックを特定するために、イテレーション 6 の早い段階でパフォーマンスプロファイリングを優先する。すべてのコンポーネントに構造化されたロギングを実装する。エージェントの動作に対するリグレッションテストを作成するために ADK の評価フレームワークを活用する。コンポーネント間の通信ニーズ（例: WebSocket の永続性、ADK から Director への通信、Director から OBS への接続）を考慮してデプロイ戦略を計画する。

VIII. 反復型開発プロセスの統合

A. 反復原則の適用

- 中核となるサイクル（計画 -> 設計/分析 -> 実装 -> テスト -> 評価 [1]）を繰り返す。
- フィードバック機会を最大化するために、短いイテレーション期間（計画通り 1〜2 週間）を重視する [11]。
- ステークホルダーからのフィードバックを得るために、各イテレーションの終わりにレビュー/デモを実施することの重要性を強調する [1]。
- フィードバックや技術的な課題に基づいて、柔軟性と適応性が必要であることを強調する [1]。

B. テスト戦略

- テストを終了時だけでなく、各イテレーション全体に統合する [1]。
- 単体テスト: React、FastAPI、ADK エージェント内の個々の関数/コンポーネントに対して実施する。
- 統合テスト: コンポーネント間のインターフェース（Stage <-> Director, Director <->

Engine, Engine <-> OBS）に焦点を当てる。イテレーション 2 以降、これらのテストを追加し始める。
- エンドツーエンドテスト: ユーザーインタラクションや AI 駆動シナリオをシミュレートし、完全なワークフローを検証する。後のイテレーションで重要となる。
- ユーザビリティテスト: V-Tuber の表現力、応答性、およびユーザー向け設定ツールに関するフィードバックを収集する [19]。
- エージェント評価 **(ADK):** AI の動作を体系的にテストするために ADK のツールを使用する [9]。

C. リスク管理

- 反復的なリスク管理アプローチを採用する [14]。
- 各イテレーションの開始時に潜在的なリスク（技術的、スケジュール、要件）を特定する。例: VRM 互換性の問題（イテレーション 1）、WebSocket スケーリング（イテレーション 2）、MCP 統合の複雑さ（イテレーション 3）、マルチエージェント協調のバグ（イテレーション 4）、OBS 接続障害（イテレーション 5）、パフォーマンスボトルネック（イテレーション 6）。
- リスクを分析し、優先順位を付ける [14]。
- 緩和戦略を計画する（例: 技術的スパイク、代替設計、バッファ時間の追加）。
- イテレーション中にリスクを監視する（例: デイリースタンドアップ中 [17]）。
- イテレーションレビュー中にリスクステータスを確認する。

D. 技術的考察と推奨事項

このような反復計画を成功裏に実行するには、プロセスステップ（計画、デモ、レビュー、フィードバックの組み込み）を遵守する規律が必要である [1]。レビューを省略したり、フィードバックの統合を遅らせたりすると、アプローチの利点が損なわれる。ユーザーは反復計画を提供した。調査結果 ([1]–[13]) は、サイクル、フィードバック、適応を強調する反復開発のベストプラクティスを説明している。ここでの重要な点は、プロセス自体が技術的な実装と同じくらい重要であるということである。反復的な規律（例: 定期的なデモ、フィードバックの組み込み）に従わない場合、アプローチの利点が無効になる。

リスク管理を別のフェーズとして扱うのではなく、各イテレーションに統合すること [15] が、この AI V-Tuber システムのような複雑な研究開発プロジェクト固有の課題に積極的に対処するために不可欠である。反復開発はリスクを低減することを目的としている [2]。特定の反復的リスク管理手法が存在する [14]。重要な点は、リスク管理は受動的ではなく、各イテレーションの計画とレビューの積極的な一部であるべきであり、チームがそのイテレーションの目標に固有の問題（例: イテレーション 3 における MCP の技術的実現可能性）を予測し、軽減できるようにすることである。

推奨事項: イテレーション終了時のレビュープロセスを形式化し、技術デモ、フィードバック収集、リスクと計画調整に関する明確な議論を確実に行う。各イテレーションで更新される生きた

リスク登録簿を維持する。システムの複雑さに合わせてテスト戦略を進化させる。

IX. 結論と戦略的推奨事項

A. 調査結果の概要

- 提案された反復計画の強みを要約する: 論理的な進行、コア技術への焦点、複雑さの段階的な構築。
- 各イテレーションで特定された主要な技術的課題と洞察を要約する（VRM のニュアンス、WebSocket 管理、ADK/MCP 統合、感情マッピングの複雑さ、マルチエージェント協調、OBS 制御の抽象化、デプロイメント）。

B. 包括的な戦略的推奨事項

1. ペルソナと感情マッピングの優先順位付け: V-Tuber の表現力は鍵となる。感情分析と BlendShape マッピングの改良に十分な時間を割り当てる [38]。これはイテレーション 3 を超えても、かなりの反復が必要になる可能性が高い。
2. 通信バックボーンの確立: WebSocket 通信 [46] と ADK-Director インターフェース（おそらく MCP [9]）が、後続のすべての AI インタラクションの基盤となるため、早期に堅牢で明確に定義されていることを確認する。
3. モジュール性の活用 (ADK & コンポーネント): ADK のマルチエージェント機能 [9] を最大限に活用し、vtube-stage、stage-director、vtuber-behavior-engine 間の関心事の明確な分離を維持する。これはテスト、保守、将来の拡張に役立つ。
4. テストと評価への投資: 単体テスト、統合テスト、エンドツーエンドテストを継続的に適用する。AI の品質を確保するために、ADK の評価ツール [9] を厳密に利用する。
5. 早期のデプロイ計画: 特にネットワーキングとリソース要件に関して、開発プロセス全体を通じてターゲットのデプロイ環境とその制約を考慮する [9]。

C. 最終的な考察

提案された計画は、堅固な基盤を提供する。本報告書で概説された技術的な洞察と推奨事項、特に VRM 制御のニュアンス、ADK/MCP 統合、マルチエージェントダイナミクス、および反復プロセスの規律に関する点を組み込むことにより、プロジェクトは成功する強い可能性を秘めている。

引用文献

1. Understanding the Iterative Process (with Examples) [2025] - Asana, 4月 13, 2025にアクセス、https://asana.com/resources/iterative-process
2. A Comprehensive Guide to Iterative Development For Developers - Relia Software, 4月 13, 2025にアクセス、https://reliasoftware.com/blog/iterative-development
3. The Benefits of Iterative Development in Software Engineering | DailyBot Insights,

4月 13, 2025にアクセス、https://www.dailybot.com/insights/the-benefits-of-iterative-development-in-software-engineering
4. What is an iterative approach and what are its benefits? - Atlassian Community, 4月 13, 2025にアクセス、https://community.atlassian.com/t5/Agile/What-is-an-iterative-approach-and-what-are-its-benefits/ba-p/2144733
5. Fast API WebSockets: A Comprehensive Guide - Orchestra, 4月 12, 2025にアクセス、https://www.getorchestra.io/guides/fast-api-websockets-a-comprehensive-guide
6. Comparing Python's Quart vs FastAPI: Which Async Framework Is Right for You? | slaptijack, 4月 12, 2025にアクセス、https://slaptijack.com/programming/python-quart-vs-fastapi.html
7. FastAPI vs Flask: A Comprehensive Python Web Framework Comparison | Startup House, 4月 12, 2025にアクセス、https://startup-house.com/blog/fastapi-vs-flask-comparison
8. pixiv/three-vrm: Use VRM on Three.js - GitHub, 4月 13, 2025にアクセス、https://github.com/pixiv/three-vrm
9. Agent Development Kit - Google, 4月 12, 2025にアクセス、https://google.github.io/adk-docs/
10. google/adk-python: An open-source, code-first Python toolkit for building, evaluating, and deploying sophisticated AI agents with flexibility and control. - GitHub, 4月 13, 2025にアクセス、https://github.com/google/adk-python
11. Chapter 11: Iterative Development - Agile Business Consortium, 4月 13, 2025にアクセス、https://www.agilebusiness.org/dsdm-project-framework/iterative-development.html
12. Is Iterative Development the New Black of Software Development? - UPTech Team, 4月 13, 2025にアクセス、https://www.uptech.team/blog/iterative-development
13. 7 Software Development Best Practices in 2024 - Zuci Systems, 4月 13, 2025にアクセス、https://www.zucisystems.com/blog/top-7-software-development-best-practices/
14. The 5 Steps of Iterative Risk Management - Cognition Blog, 4月 13, 2025にアクセス、https://blog.cognition.us/the-5-steps-of-iterative-risk-management
15. Iterative Risk Management test - Agile Risk Management - TrustEd Institute, 4月 13, 2025にアクセス、https://trustedinstitute.com/concept/agile-project-management/agile-risk-management/iterative-risk-management/
16. Guide to the Spiral Model: Risk-Driven Software Development - Devzery, 4月 13, 2025にアクセス、https://www.devzery.com/post/guide-to-the-spiral-model-risk-driven-software-development

17. Integrating Risk Management into Agile Development Processes - SoftwareSeni, 4月 13, 2025にアクセス、https://www.softwareseni.com/integrating-risk-management-into-agile-development-processes/
18. Agile Risk Management: Lessons from SpaceX's Iterative Rocket Development Process, 4月 13, 2025にアクセス、https://www.readinow.com/blog/agile-risk-management-lessons-from-spacexs-iterative-rocket-dev-process
19. What Is Iterative Testing - Easy Definition and FAQ - Airfocus, 4月 13, 2025にアクセス、https://airfocus.com/glossary/what-is-iterative-testing/
20. Iterative Testing in Product Design: Best Practices, 4月 13, 2025にアクセス、https://productschool.com/blog/user-experience/iterative-testing
21. What is Iterative Testing? | Definition and Overview - ProductPlan, 4月 13, 2025にアクセス、https://www.productplan.com/glossary/iterative-testing/
22. Iterative Testing: How to Fail Fast & Succeed - Qohash, 4月 13, 2025にアクセス、https://qohash.com/iterative-testing/
23. How Does Iterative Testing Help Build Better Products [+Examples] - Userpilot, 4月 13, 2025にアクセス、https://userpilot.com/blog/iterative-testing/
24. three-vrm.md - GitHub Gist, 4月 12, 2025にアクセス、https://gist.github.com/ahuglajbclajep/6ea07f6feb250aa776afa141a35e725b
25. @pixiv/three-vrm examples - CodeSandbox, 4月 12, 2025にアクセス、https://codesandbox.io/examples/package/@pixiv/three-vrm
26. Lerping camera and lookAt together with dynamic taret (React three fiber) - Questions, 4月 12, 2025にアクセス、https://discourse.threejs.org/t/lerping-camera-and-lookat-together-with-dynamic-taret-react-three-fiber/28896
27. Features and contents of VRM, 4月 12, 2025にアクセス、https://vrm.dev/en/vrm/vrm_features/
28. vrm-specification/specification/VRMC_vrm-1.0/README.md at master · vrm-c/vrm ... - GitHub, 4月 12, 2025にアクセス、https://github.com/vrm-c/vrm-specification/blob/master/specification/VRMC_vrm-1.0/README.md
29. Implementation of VRM on Three.js - W3C, 4月 12, 2025にアクセス、https://www.w3.org/2019/09/Meetup/YutakaObuchi_VRM.pdf
30. types-vrm-0.0 | @pixiv/three-vrm - GitHub Pages, 4月 13, 2025にアクセス、https://pixiv.github.io/three-vrm/docs/modules/types-vrm-0.0.html
31. pixiv/three-vrm - UNPKG, 4月 13, 2025にアクセス、https://unpkg.com/browse/@pixiv/three-vrm@0.6.8/README.md
32. three-vrm example, 4月 12, 2025にアクセス、https://pixiv.github.io/three-vrm/packages/three-vrm/examples/
33. pixiv three-vrm · Discussions - GitHub, 4月 13, 2025にアクセス、https://github.com/pixiv/three-vrm/discussions
34. pixiv/three-vrm-materials-mtoon - GitHub Pages, 4月 13, 2025にアクセス、https://pixiv.github.io/three-vrm/packages/three-vrm-materials-mtoon/docs/
35. Issues · pixiv/three-vrm - GitHub, 4月 13, 2025にアクセス、

https://github.com/pixiv/three-vrm/issues
36. 1月 1, 1970にアクセス、https://pixiv.github.io/three-vrm/docs/modules/three-vrm
37. VRMC_vrm: expression | VRM, 4月 12, 2025にアクセス、https://vrm.dev/vrm1/expression/
38. Getting Google MediaPipe to control VRM Characters - Extra Ordinary, the Series, 4月 12, 2025にアクセス、https://extra-ordinary.tv/2024/01/07/getting-google-mediapipe-to-control-vrm-characters/
39. BlendShape Setting - VRM, 4月 12, 2025にアクセス、https://vrm.dev/en/univrm/blendshape/univrm_blendshape/
40. BlendShape Setup (v0.45) - VRM, 4月 12, 2025にアクセス、https://vrm.dev/en/univrm/blendshape/blendshape_setup/
41. Configure blend shape on VRM [VirtualCast], 4月 12, 2025にアクセス、https://wiki.virtualcast.jp/wiki/en/vrm/setting/blendshap
42. Emotional Features of Short Conversations between a Generative AI-Controlled Virtual Communication Trainer and Four Children - Stephy Publishers, 4月 12, 2025にアクセス、https://www.stephypublishers.com/sojei/pdf/SOJEI.MS.ID.000504.pdf
43. LookAt blendshapes · pixiv three-vrm · Discussion #1303 - GitHub, 4月 12, 2025にアクセス、https://github.com/pixiv/three-vrm/discussions/1303
44. Getting older VRM character to look at the camera · Issue #1173 · pixiv/three-vrm - GitHub, 4月 12, 2025にアクセス、https://github.com/pixiv/three-vrm/issues/1173
45. Trying to get VRM blendshape to make mouth work, every sliders simply do that. Why does it happen and how can I fix it? : r/vtubertech - Reddit, 4月 12, 2025にアクセス、https://www.reddit.com/r/vtubertech/comments/1aorbuo/trying_to_get_vrm_blendshape_to_make_mouth_work/
46. WebSockets - FastAPI, 4月 13, 2025にアクセス、https://fastapi.tiangolo.com/advanced/websockets/
47. FastAPI and WebSockets: A Comprehensive Guide - Orchestra, 4月 13, 2025にアクセス、https://www.getorchestra.io/guides/fastapi-and-websockets-a-comprehensive-guide
48. Mastering FastAPI WebSockets: A Beginner's Guide - Apidog, 4月 13, 2025にアクセス、https://apidog.com/blog/fastapi-websockets/
49. Build dynamic, secure APIs with FastAPI: Features DB integration, real-time WebSocket, streaming, and efficient request handling with middleware, powered by Starlette and Pydantic. - GitHub, 4月 13, 2025にアクセス、https://github.com/zhiyuan8/FastAPI-websocket-tutorial
50. WebSockets - FastAPI, 4月 13, 2025にアクセス、https://fastapi.tiangolo.com/reference/websockets/
51. WebSockets, Socket.IO, and Real-Time Communication with Node.js, 4月 12, 2025にアクセス、https://dev.to/imsushant12/websockets-socketio-and-real-time-communication-with-nodejs-4ea0

52. WebSocket vs Ws vs Socket.io ? : r/node - Reddit, 4月 12, 2025にアクセス、https://www.reddit.com/r/node/comments/18vx421/websocket_vs_ws_vs_socketio/
53. Socket.IO vs. WebSocket: Key differences and which one to use in 2024 - Ably, 4月 12, 2025にアクセス、https://ably.com/topic/socketio-vs-websocket
54. SSE vs WebSockets: Comparing Real-Time Communication Protocols - SoftwareMill, 4月 12, 2025にアクセス、https://softwaremill.com/sse-vs-websockets-comparing-real-time-communication-protocols/
55. Tutorial - Overview of the API - Socket.IO, 4月 12, 2025にアクセス、https://socket.io/docs/v4/tutorial/api-overview
56. Introduction | Socket.IO, 4月 12, 2025にアクセス、https://socket.io/docs/v4/
57. 8 best WebSocket libraries for Node - Ably, 4月 12, 2025にアクセス、https://ably.com/blog/websocket-libraries-for-node
58. Get started - Socket.IO, 4月 12, 2025にアクセス、https://socket.io/get-started/chat
59. Differences between socket.io and websockets - Stack Overflow, 4月 12, 2025にアクセス、https://stackoverflow.com/questions/10112178/differences-between-socket-io-and-websockets
60. Socket. IO vs. WebSocket: Keys Differences - Apidog, 4月 12, 2025にアクセス、https://apidog.com/articles/socket-io-vs-websocket/
61. Socket.IO vs WebSocket: Comprehensive Comparison and Implementation Guide, 4月 12, 2025にアクセス、https://www.videosdk.live/developer-hub/websocket/socketio-vs-websocket
62. Tutorial - Introduction - Socket.IO, 4月 12, 2025にアクセス、https://socket.io/docs/v4/tutorial/introduction
63. Get started - Socket.IO, 4月 12, 2025にアクセス、https://socket.io/get-started/
64. google/adk-docs: An open-source, code-first Python toolkit for building, evaluating, and deploying sophisticated AI agents with flexibility and control. - GitHub, 4月 12, 2025にアクセス、https://github.com/google/adk-docs
65. Agent Development Kit: Making it easy to build multi-agent applications, 4月 12, 2025にアクセス、https://developers.googleblog.com/en/agent-development-kit-easy-to-build-multi-agent-applications/
66. Quickstart: Build an agent with the Agent Development Kit | Generative AI on Vertex AI, 4月 12, 2025にアクセス、https://cloud.google.com/vertex-ai/generative-ai/docs/agent-development-kit/quickstart
67. Quickstart - Agent Development Kit - Google, 4月 13, 2025にアクセス、https://google.github.io/adk-docs/get-started/quickstart/
68. Agents - Agent Development Kit - Google, 4月 12, 2025にアクセス、https://google.github.io/adk-docs/agents/
69. Specification - Model Context Protocol, 4月 12, 2025にアクセス、https://modelcontextprotocol.io/specification/2025-03-26

70. Model Context Protocol: Introduction, 4月 12, 2025にアクセス、https://modelcontextprotocol.io/introduction
71. Model Context Protocol (MCP) an overview - Philschmid, 4月 12, 2025にアクセス、https://www.philschmid.de/mcp-introduction
72. Model Context Protocol (MCP): A Guide With Demo Project - DataCamp, 4月 12, 2025にアクセス、https://www.datacamp.com/tutorial/mcp-model-context-protocol
73. MCP Server in Python — Everything I Wish I'd Known on Day One | DigitalOcean, 4月 12, 2025にアクセス、https://www.digitalocean.com/community/tutorials/mcp-server-python
74. MCP tools - Agent Development Kit - Google, 4月 13, 2025にアクセス、https://google.github.io/adk-docs/tools/mcp-tools/
75. Build and manage multi-system agents with Vertex AI | Google Cloud Blog, 4月 12, 2025にアクセス、https://cloud.google.com/blog/products/ai-machine-learning/build-and-manage-multi-system-agents-with-vertex-ai
76. Just did a deep dive into Google's Agent Development Kit (ADK). Here are some thoughts, nitpicks, and things I loved (unbiased) - Reddit, 4月 12, 2025にアクセス、https://www.reddit.com/r/AI_Agents/comments/1jvsu4l/just_did_a_deep_dive_into_googles_agent/
77. Model context protocol (MCP) - OpenAI Agents SDK, 4月 12, 2025にアクセス、https://openai.github.io/openai-agents-python/mcp/
78. Just did a deep dive into Google's Agent Development Kit (ADK). Here are some thoughts, nitpicks, and things I loved (unbiased) - Reddit, 4月 12, 2025にアクセス、https://www.reddit.com/r/LocalLLaMA/comments/1jvsvzj/just_did_a_deep_dive_into_googles_agent/
79. Thinking if MCP server could work with anki connect? thx - Add-ons, 4月 12, 2025にアクセス、https://forums.ankiweb.net/t/thinking-if-mcp-server-could-work-with-anki-connect-thx/57436
80. Google CAN'T STOP COOKING: FREE AI AGENTS SDK JUST DROPPED - YouTube, 4月 12, 2025にアクセス、https://www.youtube.com/watch?v=tldS6e5rD8A
81. Introducing Agent Development Kit - YouTube, 4月 12, 2025にアクセス、https://www.youtube.com/watch?v=zgrOwow_uTQ
82. The Model Context Protocol (MCP) Explained (and one cool code example.) - YouTube, 4月 12, 2025にアクセス、https://www.youtube.com/watch?v=5ZWeCKY5WZE
83. The MCP Integration EVERYONE is Sleeping On (MCP + Custom AI Agents) - YouTube, 4月 12, 2025にアクセス、https://www.youtube.com/watch?v=soC4n-nKWF8
84. Building AI Agents with Model Context Protocol: From Specification to Implementation, 4月 12, 2025にアクセス、https://www.youtube.com/watch?v=oSGVQIZxi7s
85. pyproject.toml - google/adk-python - GitHub, 4月 12, 2025にアクセス、https://github.com/google/adk-python/blob/main/pyproject.toml

86. ADK で作った agent を mcp server で公開する - Zenn, 4月 13, 2025にアクセス、https://zenn.dev/satohjohn/articles/48a82ff7de531b
87. Model Context Protocol · GitHub, 4月 12, 2025にアクセス、https://github.com/modelcontextprotocol
88. Build Your First Intelligent Agent Team: A Progressive Weather Bot with ADK - Google, 4月 12, 2025にアクセス、https://google.github.io/adk-docs/get-started/tutorial/
89. Python Code For Facial Expression Analysis - Restack, 4月 12, 2025にアクセス、https://www.restack.io/p/ai-for-emotion-recognition-answer-python-code-facial-expression-analysis-cat-ai
90. Sentiment Analysis: First Steps With Python's NLTK Library - Real Python, 4月 12, 2025にアクセス、https://realpython.com/python-nltk-sentiment-analysis/
91. Sentiment Analysis Using Python - Analytics Vidhya, 4月 12, 2025にアクセス、https://www.analyticsvidhya.com/blog/2022/07/sentiment-analysis-using-python/
92. 6 Must-Know Python Sentiment Analysis Libraries - Netguru, 4月 12, 2025にアクセス、https://www.netguru.com/blog/python-sentiment-analysis-libraries
93. 2.8 Project: Detecting Emotions from Text — Practical NLP with Python - NLPlanet, 4月 12, 2025にアクセス、https://www.nlplanet.org/course-practical-nlp/02-practical-nlp-first-tasks/08-emotion-classification
94. Text Analysis in Python: Techniques and Libraries Explained - Airbyte, 4月 12, 2025にアクセス、https://airbyte.com/data-engineering-resources/text-analysis-in-python
95. EmoLLMs: A Series of Emotional Large Language Models and Annotation Tools for Comprehensive Affective Analysis - arXiv, 4月 12, 2025にアクセス、https://arxiv.org/html/2401.08508v2
96. UCREL/Session-6-LLM-Based-Emotion-Analysis - GitHub, 4月 12, 2025にアクセス、https://github.com/UCREL/Session-6-LLM-Based-Emotion-Analysis
97. Emotion Analysis AI Model for Sensing Architecture Using EEG - MDPI, 4月 12, 2025にアクセス、https://www.mdpi.com/2076-3417/15/5/2742
98. Emotion-Preserving Blendshape Update With Real-Time Face Tracking - Feng Xu, 4月 12, 2025にアクセス、http://xufeng.site/publications/2020/wzb_Emotion-preserving%20Blendshape%20Update%20with%20Real-time%20Face%20Tracking.pdf
99. Research on map emotional semantics using deep learning approach - ResearchGate, 4月 12, 2025にアクセス、https://www.researchgate.net/publication/368696901_Research_on_map_emotional_semantics_using_deep_learning_approach
100. Emo3D: Metric and Benchmarking Dataset for 3D Facial Expression Generation from Emotion Description - arXiv, 4月 12, 2025にアクセス、https://arxiv.org/html/2410.02049v1
101. Group Emotion Detection Based on Social Robot Perception - MDPI, 4月 12, 2025にアクセス、https://www.mdpi.com/1424-8220/22/10/3749
102. Multimodal Feature Learning for Video Captioning - ResearchGate, 4月 12, 2025にアクセス、

https://www.researchgate.net/publication/323285196_Multimodal_Feature_Learning_for_Video_Captioning
103. SURP Archives - - Cal Poly College of Engineering, 4月 12, 2025にアクセス、https://ceng.calpoly.edu/surp-archives/
104. Py-Feat: Python Facial Expression Analysis Toolbox - PMC, 4月 12, 2025にアクセス、https://pmc.ncbi.nlm.nih.gov/articles/PMC10751270/
105. Text Emotion Detection using NLP | Python | Streamlit Web Application - YouTube, 4月 12, 2025にアクセス、https://m.youtube.com/watch?v=J87jV9OGodw
106. Emotion detection with Python and OpenCV | Computer vision tutorial - YouTube, 4月 12, 2025にアクセス、https://www.youtube.com/watch?v=Vq_01gFG2vk
107. How to Swap VRoid VRM textures with blendshapes in Unity - YouTube, 4月 12, 2025にアクセス、https://www.youtube.com/watch?v=MlzJtdvgdb0
108. Emotion Detection of Text Using Machine Learning and Python - YouTube, 4月 12, 2025にアクセス、https://m.youtube.com/watch?v=t1TkAcSDsI8&pp=ygUVI25vb2llbW90aW9uZGV0ZWN0aW9u
109. TensorFlow.js in the browser for expression prediction - Extra Ordinary, the Series, 4月 12, 2025にアクセス、https://extra-ordinary.tv/2024/01/11/tensorflow-js-in-the-browser-for-expression-prediction/
110. LLM-based Multi-Agent Systems: Techniques and Business Perspectives - arXiv, 4月 12, 2025にアクセス、https://arxiv.org/html/2411.14033v2
111. Top 3 Trending Agentic AI Frameworks: LangGraph vs AutoGen vs Crew AI - Datagrom, 4月 12, 2025にアクセス、https://www.datagrom.com/data-science-machine-learning-ai-blog/langgraph-vs-autogen-vs-crewai-comparison-agentic-ai-frameworks
112. A Tour of Popular Open Source Frameworks for LLM-Powered Agents - Dataiku blog, 4月 12, 2025にアクセス、https://blog.dataiku.com/open-source-frameworks-for-llm-powered-agents
113. Comparing AI agent frameworks: CrewAI, LangGraph, and BeeAI - IBM Developer, 4月 12, 2025にアクセス、https://developer.ibm.com/articles/awb-comparing-ai-agent-frameworks-crewai-langgraph-and-beeai
114. Multi-Agent Large Language Models for Conversational Task-Solving - arXiv, 4月 12, 2025にアクセス、https://arxiv.org/html/2410.22932v1
115. Beyond Self-Talk: A Communication-Centric Survey of LLM-Based Multi-Agent Systems, 4月 12, 2025にアクセス、https://arxiv.org/html/2502.14321v1
116. Multi-Agent Collaboration Mechanisms: A Survey of LLMs - arXiv, 4月 12, 2025にアクセス、https://arxiv.org/html/2501.06322v1
117. Exploring and Controlling Diversity in LLM-Agent Conversation - arXiv, 4月 12, 2025にアクセス、https://arxiv.org/html/2412.21102v1
118. Evaluating LLM-based Agents for Multi-Turn Conversations: A Survey - arXiv, 4月 12, 2025にアクセス、https://arxiv.org/html/2503.22458v1
119. A Survey on LLM-based Multi-Agent System: Recent Advances and New

Frontiers in Application - arXiv, 4月 12, 2025にアクセス、https://arxiv.org/html/2412.17481v2
120. Modeling Response Consistency in Multi-Agent LLM Systems: A Comparative Analysis of Shared and Separate Context Approaches - arXiv, 4月 12, 2025にアクセス、https://arxiv.org/html/2504.07303v1
121. Choosing the Right AI Agent Framework: LangGraph vs CrewAI vs OpenAI Swarm, 4月 12, 2025にアクセス、https://www.relari.ai/blog/ai-agent-framework-comparison-langgraph-crewai-openai-swarm
122. Comparing Open-Source AI Agent Frameworks - Langfuse Blog, 4月 12, 2025にアクセス、https://langfuse.com/blog/2025-03-19-ai-agent-comparison
123. obsproject/obs-websocket: Remote-control of OBS Studio ... - GitHub, 4月 12, 2025にアクセス、https://github.com/obsproject/obs-websocket
124. obsws-python - PyPI, 4月 13, 2025にアクセス、https://pypi.org/project/obsws-python/1.1.0/
125. How to Setup and Use OBS WebSocket? - Video SDK, 4月 12, 2025にアクセス、https://www.videosdk.live/developer-hub/websocket/obs-websocket
126. pschmitt/obs-cli: CLI for controlling OBS Studio - GitHub, 4月 13, 2025にアクセス、https://github.com/pschmitt/obs-cli
127. Issues · aatikturk/obsws-python - GitHub, 4月 12, 2025にアクセス、https://github.com/aatikturk/obsws-python/issues
128. aatikturk/obsws-python: A Python SDK for OBS Studio WebSocket v5.0 - GitHub, 4月 13, 2025にアクセス、https://github.com/aatikturk/obsws-python
129. ObsWs is a API library for accessing OBS WebSocket API through Godot - GitHub, 4月 13, 2025にアクセス、https://github.com/eumario/ObsWs
130. Python Script to connect to OBS Websocket Server Help, 4月 12, 2025にアクセス、https://obsproject.com/forum/threads/python-script-to-connect-to-obs-websocket-server-help.173395/
131. Python OBS-WebSocket-Py Set Source Visibility - Stack Overflow, 4月 12, 2025にアクセス、https://stackoverflow.com/questions/78210213/python-obs-websocket-py-set-source-visibility
132. Elektordi/obs-websocket-py - GitHub, 4月 13, 2025にアクセス、https://github.com/Elektordi/obs-websocket-py
133. GoogleCloudPlatform/agent-starter-pack: A collection of production-ready Generative AI Agent templates built for Google Cloud. It accelerates development by providing a holistic, production-ready solution, addressing common challenges (Deployment & Operations, Evaluation, Customization, Observability) in building and deploying GenAI agents. - GitHub, 4月 12, 2025にアクセス、https://github.com/GoogleCloudPlatform/agent-starter-pack