音声インタラクションシステム構築ツールキット

MMDAgent

平成27年3月25日版

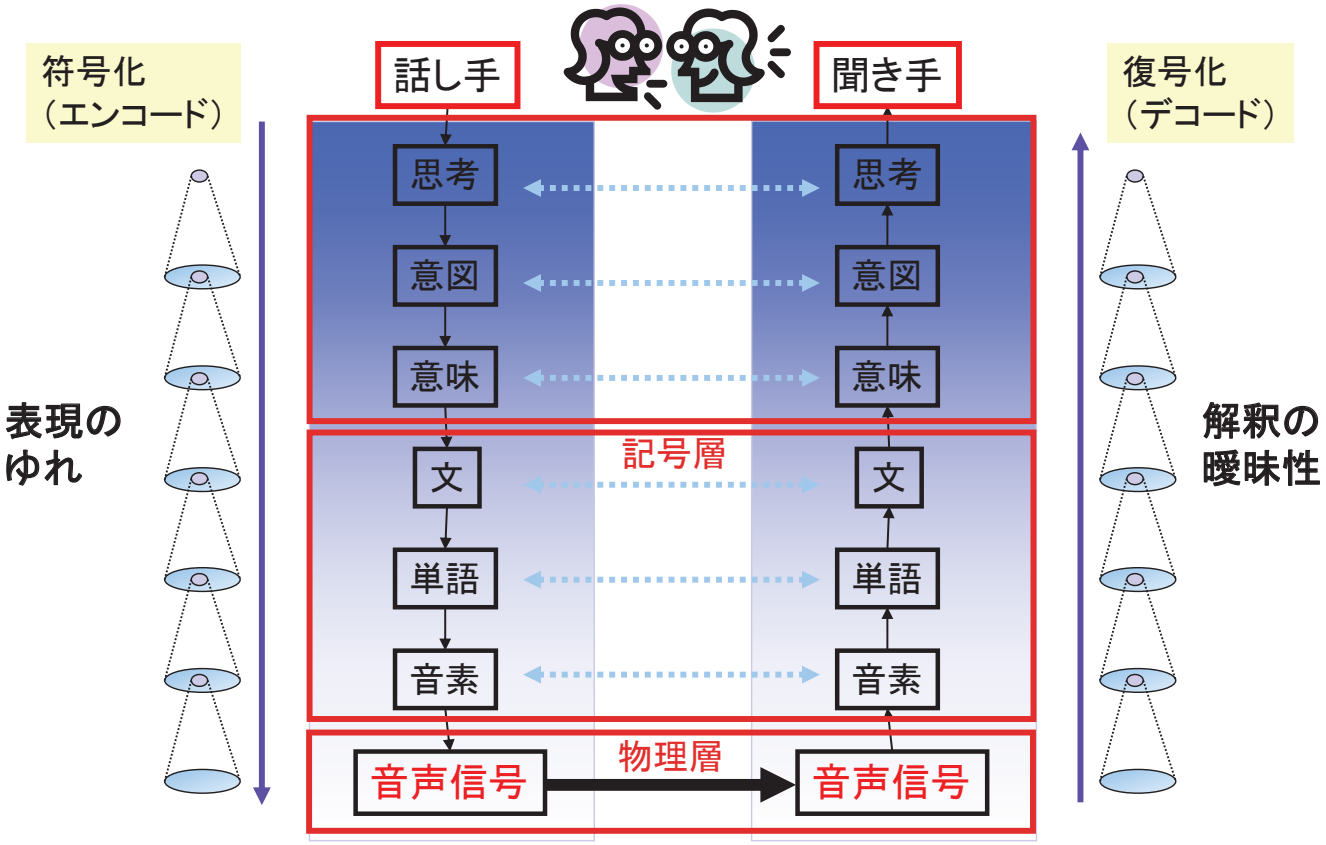
JST CREST uDialogue Project
http://www.udialogue.org/

音声対話システム

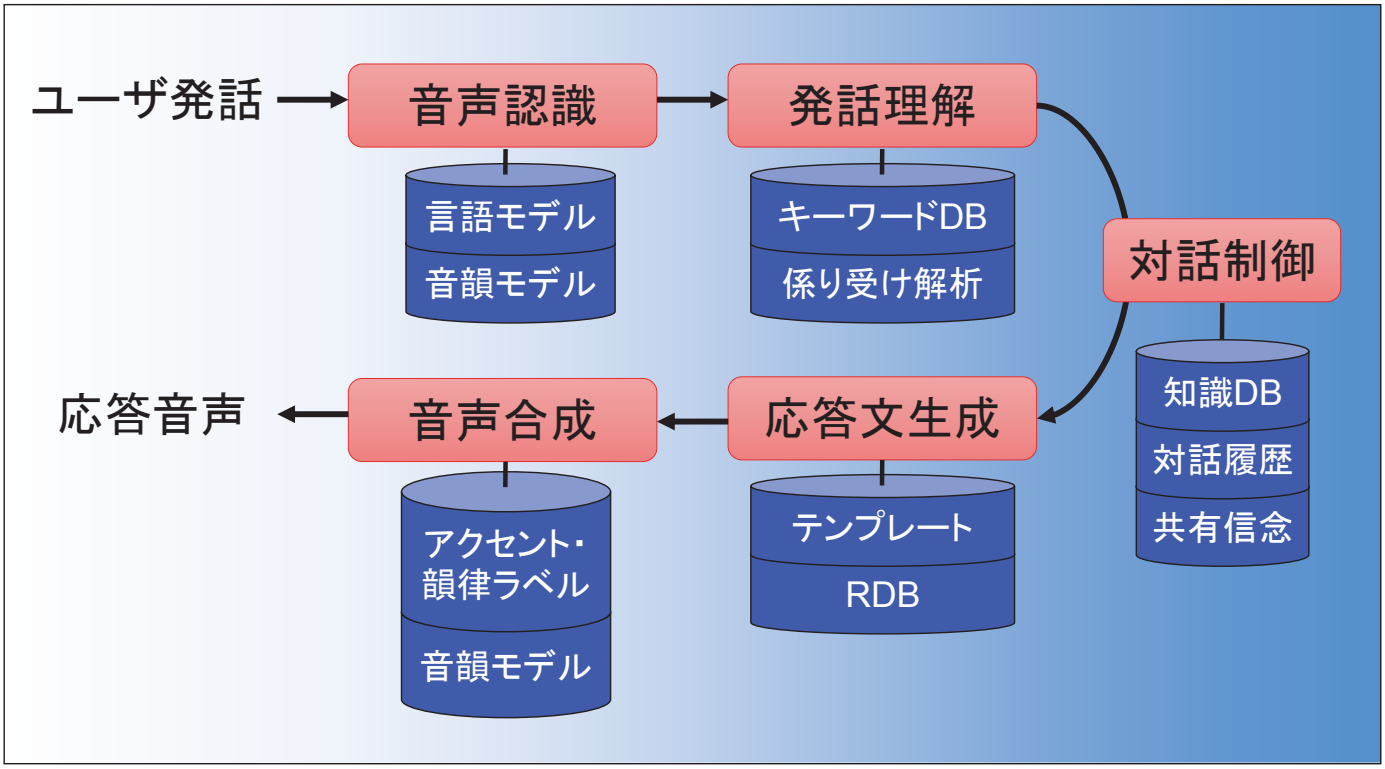
音声対話システム

- 音声言語を用いた自然なインタフェース
- 「話す機械」の実現
- 近年の音声認識・音声合成の発達が後押し

音声コミュニケーションの階層モデル



音声対話システムの基本構成



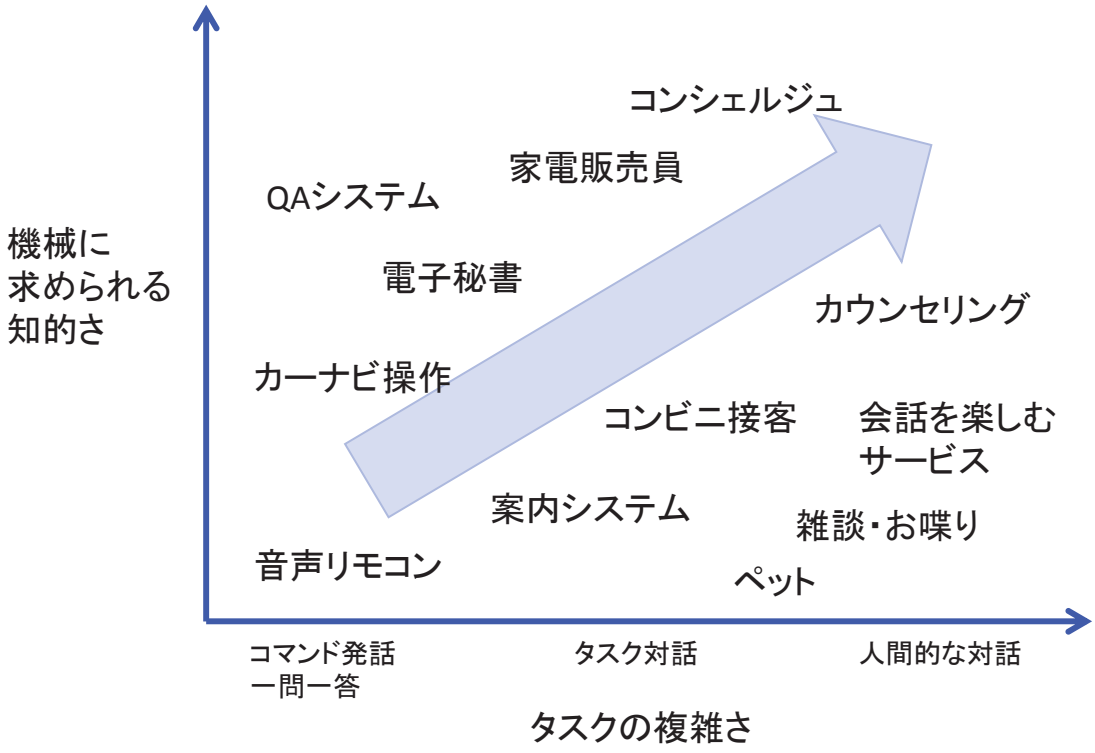
これまでの音声対話サービス

- 情報サービスの自動化
  - フライト情報，天気予報，スポーツの試合結果，株価
  - Web検索，路線検索
- トランザクションの実行
  - ホテル予約，レンタカー予約
  - カーナビ
- 問題解決
  - サポート窓口の自動応答

関連する研究項目

- 知識表現・問題解決
- 対話戦略・イニシアチブ
- エラー検出・予測・回復
- 協調的やりとり
- 適応
- ユーザモデル
- マルチモーダル
- 会話調インタフェース
- 擬人化エージェント
- 感情状態の認識・感情の表出
- ノンバーバルコミュニケーション（視線，ジェスチャ，うなずき）

音声対話システムの未来の応用



機械と…

- 話したいと思いますか？

- 何を話せばいいか分かりますか？

- 気負わず話せますか？

ユーザから見た音声インタフェース

認識率100%の前提でも・・・

- 「面白い！でも要らない」
- 「タッチやキーボードで十分」
- 「面倒くさい」
- 「機械と話すのが恥ずかしい」
- 「何をしゃべっていいか分からない」
- 「なんでわざわざ機械と喋らなければいけない？」

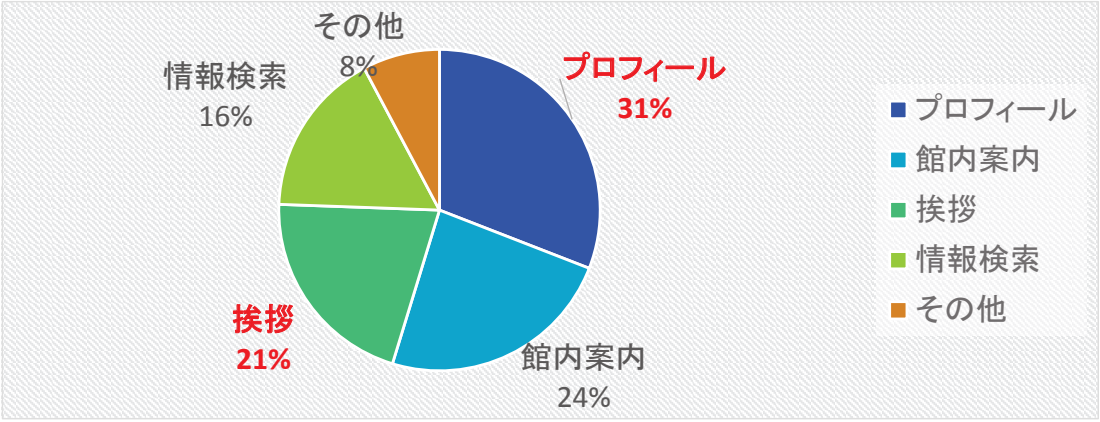
恥ずかしい？

面倒？

分かりにくい？

資料：公共情報案内システムへの自由発話の内訳

- 「たけまるくん」
  - 生駒市公共施設ロビーに常設
  - 施設案内／あいさつ／天気等
  - 2002年11月より稼動



何を期待され，何が足りないのか

- もっと知的であれば話せるのか
- 見た目が素敵であれば話せるのか
- 機能が分かりやすければよいのか
- 受付単語がリストで与えられれば発話できるのか

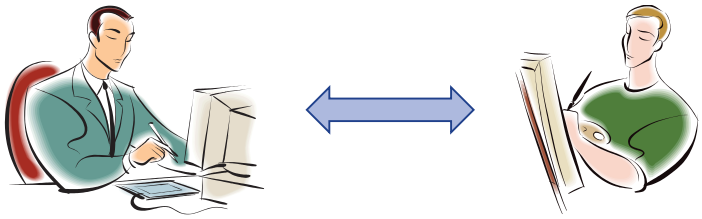
- 機械に喋る行為への慣れの問題なのか
- 文化的な問題なのか
- 人が「話そう」と思えるモチベーションとは

Natural Interaction

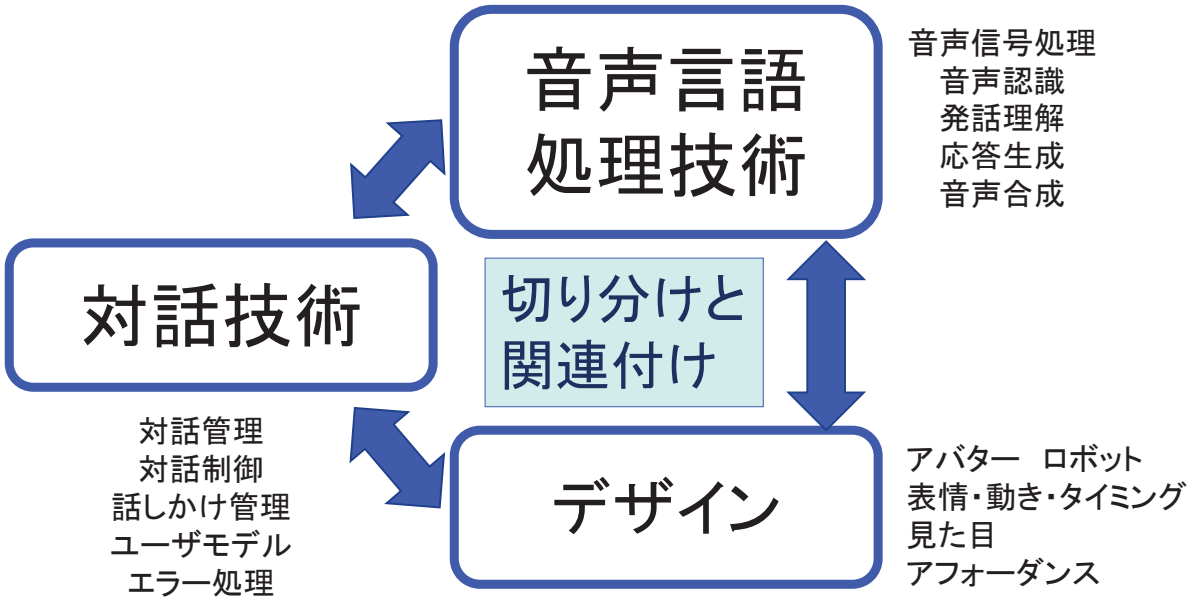
- ノンバーバル・マルチモーダル音声対話
  - より**直感的**で **affordable** （受け入れられる）な音声対話システムへ
- Embodied Conversational Agents [Cassel et al.,2003]
  - 人間同士の対話の分析とモデル化
    - 視線，ジェスチャ，社会的関係性，etc…
  - キャラクターと人間のインタラクションの頑健化へ
  - 「気が合う（調和する）感覚が会話を起こさせる」
- “Affective Computing”
  - 「ユーザの心的傾向や状態に合わせた社交的で共感的な表出は，ユーザに好まれ信頼される」

技術とデザインの接点の設計

- 音声対話のためのデザインとは？
  - システムの能力を生かしたデザイン
  - ユーザ要求から乖離しないシステム
- どちらの世界からも歩み寄るには
  - 技術者から見たデザインへの要求
  - デザイナーが活かし切れる技術の提供
- クリアなインタフェースが必要

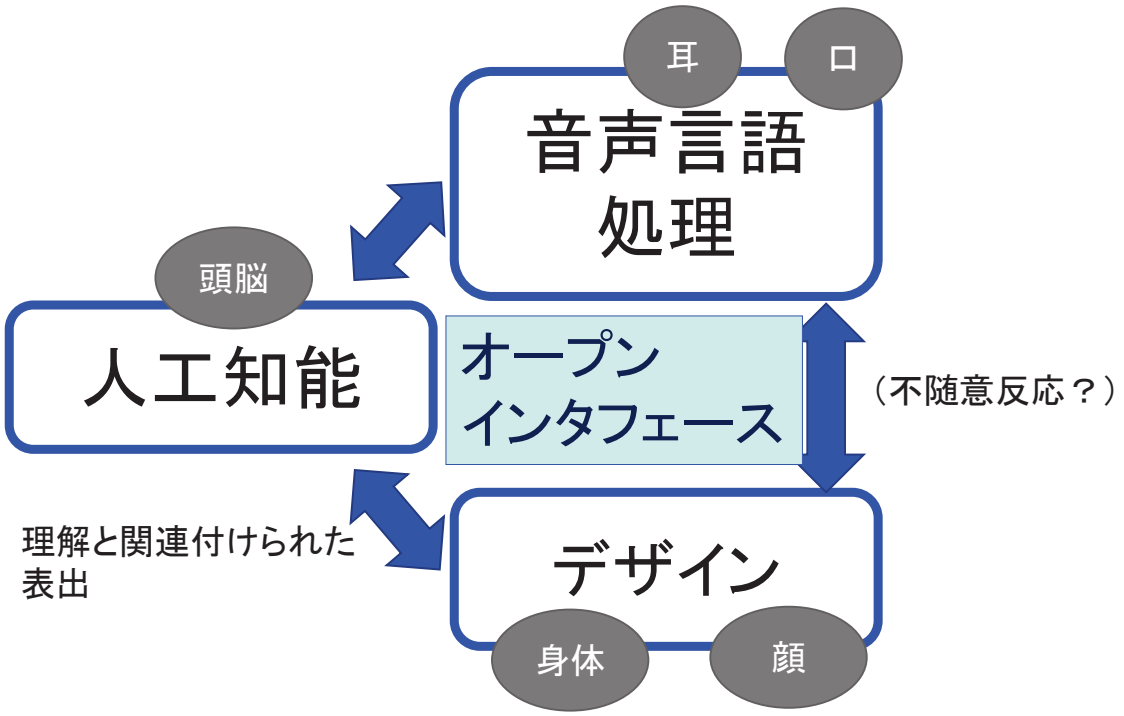


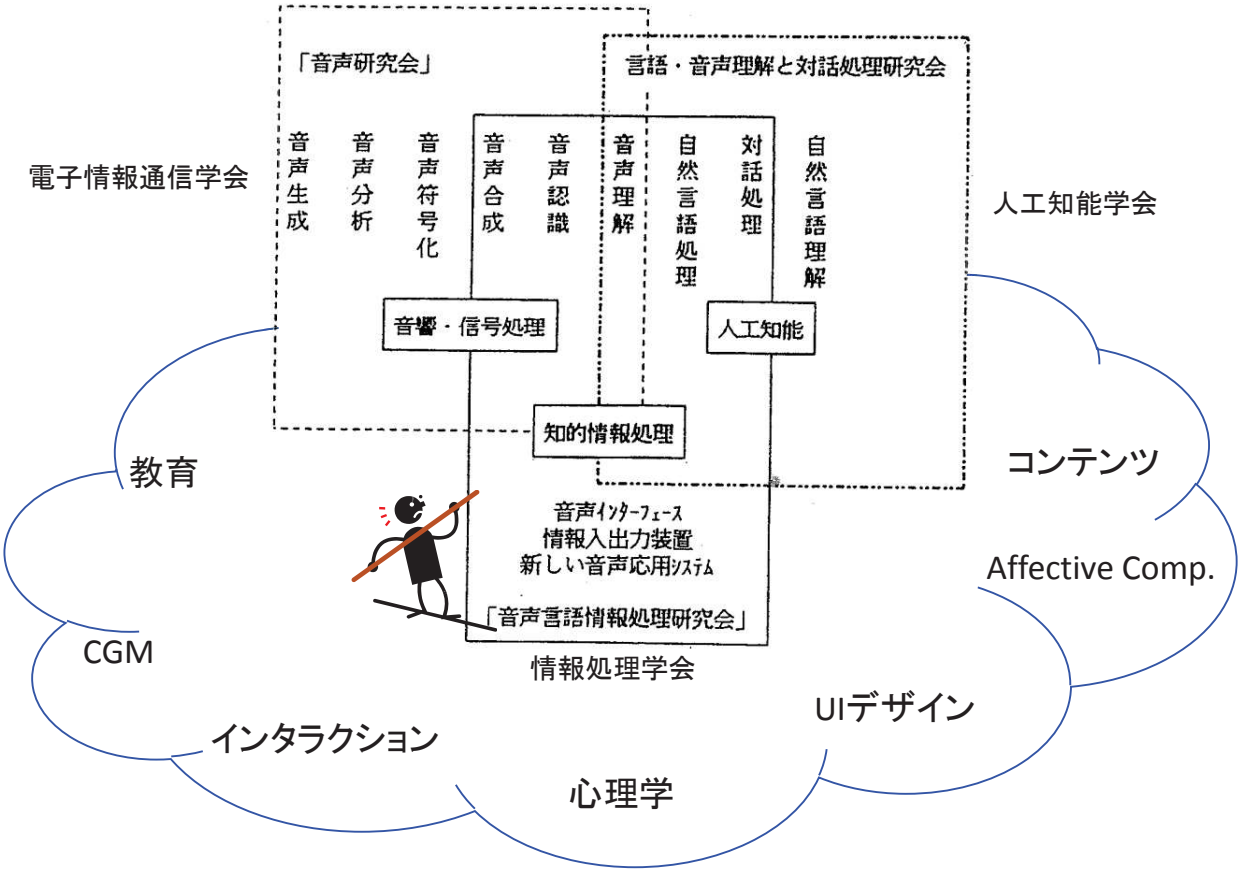
音声対話における役割の分離

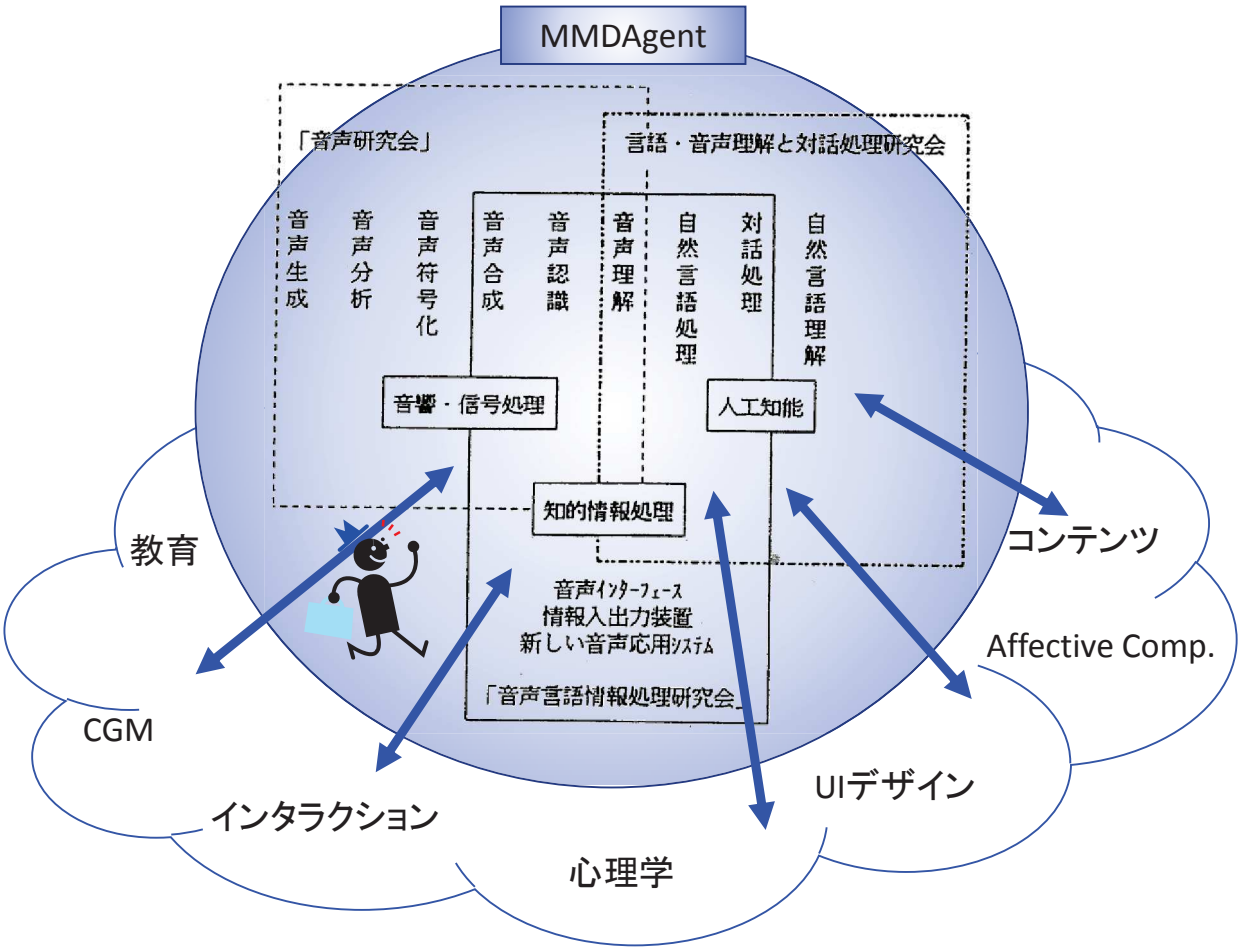


目

理解のための入出力


耳
口
音声言語処理
頭脳
人工知能
オープンインタフェース
（不随意反応？）
理解と関連付けられた表出
デザイン
身体
顔


「音声研究会」
言語・音声理解と対話処理研究会
電子情報通信学会
音声生成
音声分析
音声符号化
音声合成
音声認識
音声理解
自然言語処理
対話処理
自然言語理解
人工知能学会
音響・信号処理
人工知能
知的情報処理
教育
コンテンツ
音声ｲﾝﾀｰﾌｪｰｽ
情報入出力装置
新しい音声応用ｼｽﾃﾑ
Affective Comp.
「音声言語情報処理研究会」
CGM
情報処理学会
UIデザイン
インタラクション
心理学



MMD = MikuMikuDance

- 三次元オブジェクトを操作する
  アニメーション CG 作成ソフトウェア
  - VOCALOID 「初音ミク」 用PV作成ツール
- 3Dモデルとモーションが部品化
  - ある程度共通のフォーマット
  - 盛んに配布と共有
- CGM(Consumer Generated Media)シーンを形成
  - 多くのアマチュア・プロクリエイター
- 対話システムに活かせる高い表現力
  - 演算による表現付与（物理演算、IK）

デザイン側のプラットフォームとして有望

音声認識＋音声合成＋MMD＝MMDAgent

- 研究：音声対話研究のプラットフォーム
  - 耳、口、骨、皮、空間を提供
  - デフォルトの対話管理は単純なFSTのみ
  - → 「魂（脳みそ）」は未実装
    - 人工知能技術？
    - ユーザの作り込み？
- 応用：音声対話システム・音声インタフェースを自由自在に構築できるツールキット
  - オープンソース，オープンフォーマット

MMDAgent

- 世界最先端の音声モジュール
- 高度な表現が可能な3-Dモデル表示
- 時間粒度の高い音声対話・インタラクション制御
- 関連ツールとの互換性
- オープンソース

- 多様な音声対話・音声インタラクションを自在に実現

名工大の正門にて実証実験中

音声認識

- 大語彙連続音声認識エンジン Julius
  - オープンソース
  - 高速，低遅延，高効率な動作
  - http://julius.sourceforge.jp/
- MMDAgent のプラグインとして組み込み
  - 全ての機能が利用可能
    - GMMベースの入力棄却，マルチモデル認識，・・・
  - タスク依存の追加辞書機能
    - 対話シナリオごとに認識単語を個別に追加可能
- 日本語の音声認識用モデルを同梱
  - 60,000語の辞書，Webテキストで学習
  - 不特定話者音響モデル

音声→テキスト

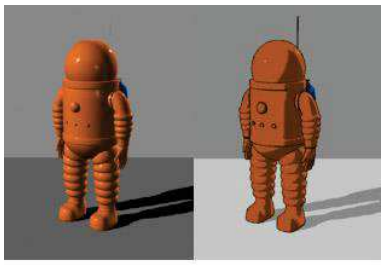
音声合成

- 音声対話システムのための音声合成
  - 多様な発話スタイル
  - シナリオに合わせた制御
- HMM-based TTS
  - 統計的パラメトリック手法
    - 小サイズ（ボイスデータは統計的パラメータで保存）
    - 素片接続に比べて歪の小さい合成音
    - パラメータ変換により出力音声の細かいコントロールが可能
- ボイスモデル（HMM）学習ツール
  - HTS: HMM-based speech synthesis system
    http://hts.sp.nitech.ac.jp/
- MMDAgent 用音声合成プラグイン
  - Open JTalk: a Japanese TTS system
    http://open-jtalk.sourceforge.net/

テキスト→音声

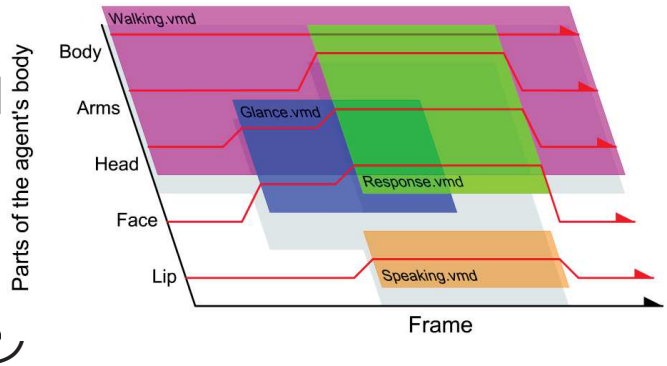
3-D エージェント・オブジェクト

- OpenGLベース
  - トゥーンシェーダ搭載
  - 任意オブジェクトを複数表示・操作
- オブジェクト定義
  - 階層ボーン構造
  - 頂点モーフィング
  - 物理剛体＋ジョイント
  - 物理演算
- オブジェクト表現
  - リアルタイムトゥーンレンダリング
  - テクスチャ・スフィアマップ・etc…



動作・モーション

- モーション
  - ボーン動作・表情モーフィング
  - 全体モーション
  - 部分モーション
  - 音声認識等の任意イベント
    に対応付け
- 複数モーションの合成
  - 非同期並列動作
  - 開始・終了のスムージング
  - 重ね合わせ（上書き・加算）



対話記述例

入力シンボル：イベント、認識結果等のメッセージ
出力シンボル：命令、動作、様々なコマンドメッセージ

```
#### format: [from_state] [to_state] [event] [command]
#Initialization: add agent and set waiting motion as base motion
0   1   <eps>                          MODEL_ADD|mei|mei.pmd
1   2   <eps>                          MOTION_ADD|mei|base|wait.vmd|FULL|LOOP
#Sample pattern 1: Listening
2   2   RECOG_EVENT_START              MOTION_ADD|mei|listen|listen.vmd|PART|ONCE
#Sample pattern 2: Greeting
2   3   RECOG_EVENT_STOP|こんにちは     MOTION_ADD|mei|greet|greet.vmd|PART|ONCE
3   4   <eps>                          MOTION_ADD|mei|smile|smile.vmd|PART|LOOP
4   5   <eps>                          SYNTH_START|mei|normal|こんにちは
5   2   SYNTH_EVENT_STOP|mei|normal    MOTION_DELETE|mei|smile
```

オープンフォーマット

- 音響・言語モデルから3-Dモデル，モーションまで自作可能

Contents	Format	Related tools
3-D Object	.pmd	MikuMikuDance* Blender, 3ds Max, etc.
Motion	.vmd	MikuMikuDance*
Acoustic model for ASR	HTK ASCII / Julius	HTK
Language model for ASR	ARPA standard format / Julius	SRILM etc.
Dialog Scenario	FST	OpenFST
Acoustic model for SS	HTS	HTS

* 3-Dオブジェクト定義（.pmd）は Blender や 3ds Max 等の3DCG作成ツールから変換可能。

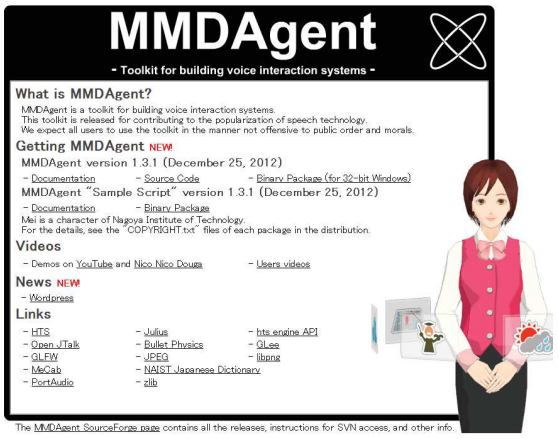
オープンソース

Module	Component	License type
3-D rendering	BulletPhysics	zlib license
	GLee	BSD license
	GLFW	zlib license
	JPEG	BSD-style license
	libpng	zlib license
	zlib	zlib license
Speech recognition	CSRC Dictionary	BSD-style license
	Julius	BSD-style license
	PortAudio	BSD-style license
Speech synthesis	hts_engine API	BSD license
	MeCab	BSD license
	Naist Japanese Dictionary	BSD license
	Open JTalk	BSD license

公開

- 最新版：1.5（2014/12/25公開）
- オール・イン・ワン パッケージ
  - Windows, Mac, Linux, Android
  - 日本語モデル（音声認識・合成）
- サンプルスクリプト・モデル
- ソースコード規模
  - 約2万行（MMDAgent固有部分）
  - 約9万行（ASR, TTS を含む全体）

http://mmdagent.jp/

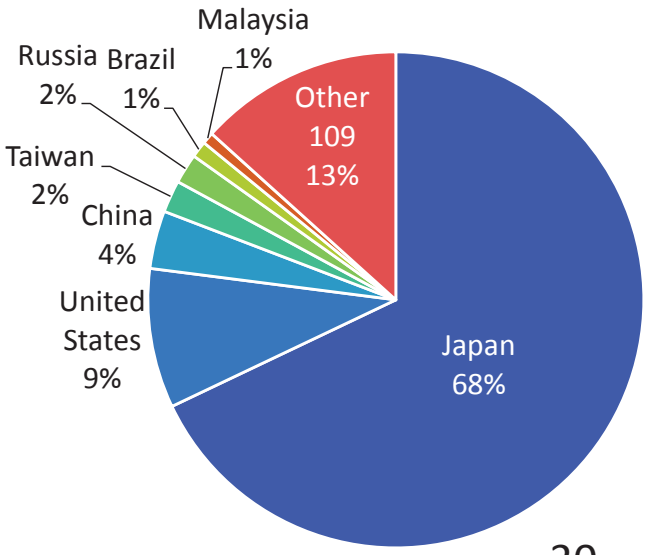


開発履歴・統計

2009年 開発開始

- 2011年5月 初版リリース

- 総計ダウンロード数
- 65,279 downloads



まとめ

- 音声対話研究の導入
- MMDAgent の開発同期とねらい
- MMDAgent の紹介

MMDAgentの実用例紹介（山本大介先生）

MMDAgentの基本的操作

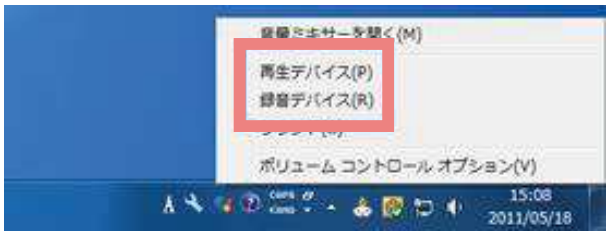
マシン設定（音声環境）

マシン設定

- イヤホン接続
  - マイク（ピンク）
  - イヤホン（緑）

音声設定方法

- サウンド機能の設定
  - 「サウンド」ウィンドウ

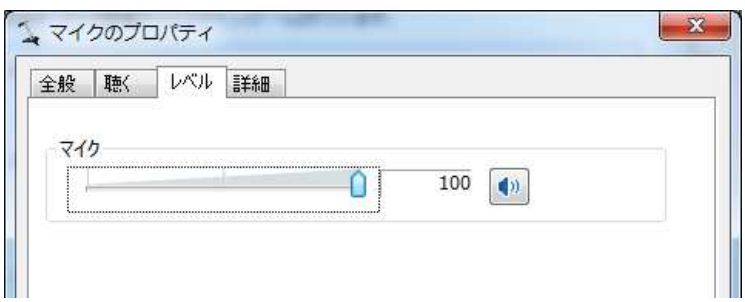


スピーカ（再生デバイス）

- 既定のデバイスに設定する
  - マイクを右クリック→規定のデバイス

マイク（録音デバイス）

- 既定のデバイスに設定する
- 音量調整
  - 右クリック→プロパティ
  - 「レベル」タブ



音量チェック

- Windows付属のサウンドレコーダー
  - スタート → すべてのプログラム → アクセサリ　内

もしくは

  - ＋R　→　「soundrecorder」と入力

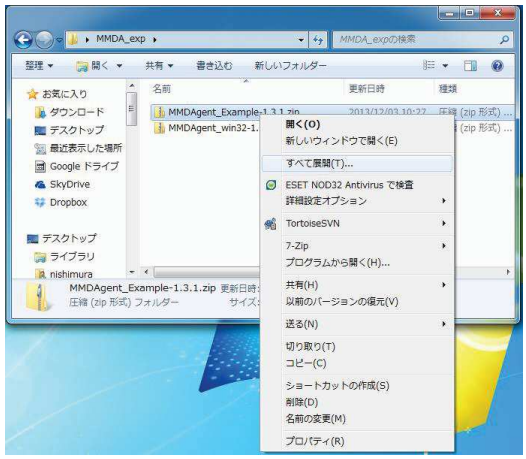
ソフトウェア(MMDAgent)ダウンロード

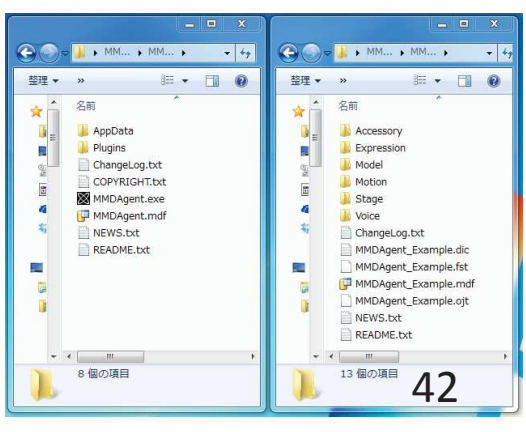
ファイルのダウンロード

- Webページにアクセス
  - http://mmdagent.jp/
- 本体・データの
  ダウンロード
  - MMDAgent_win32-1.5.zip
  - MMDAgent_Example-1.4.zip

ファイルの解凍

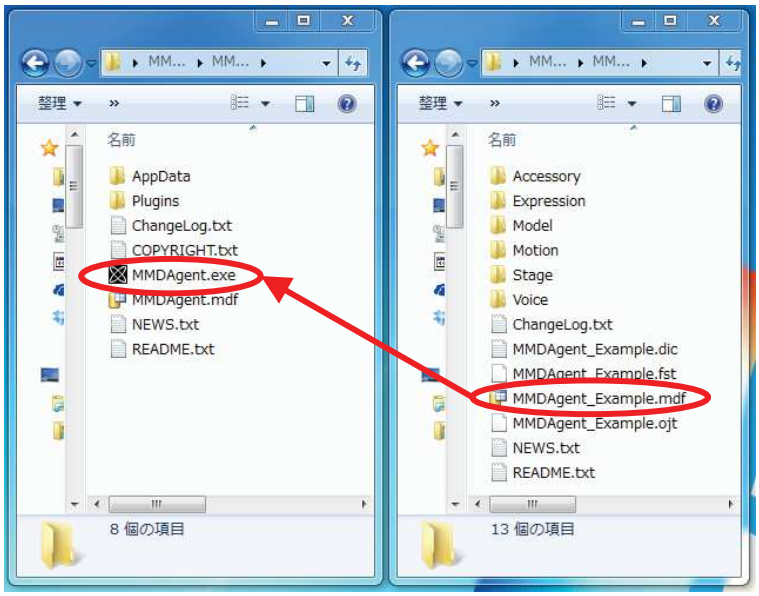
- ダウンロードしたファイルを
  解凍する
  - 右クリック→すべて展開
  - 画面の指示に従い解凍
    - 解凍後，フォルダが二重に
      なる場合があるので注意
  - 解凍後のフォルダの中身（右図）
    - 本体ファイル（左側）
    - 各種モデル，対話サンプル（右側）





実行（ドラッグ＆ドロップ法）

- .mdfファイルを本体（.exe)にD&D
  - サンプルの.mdfを
  - MMDAgent.exeに

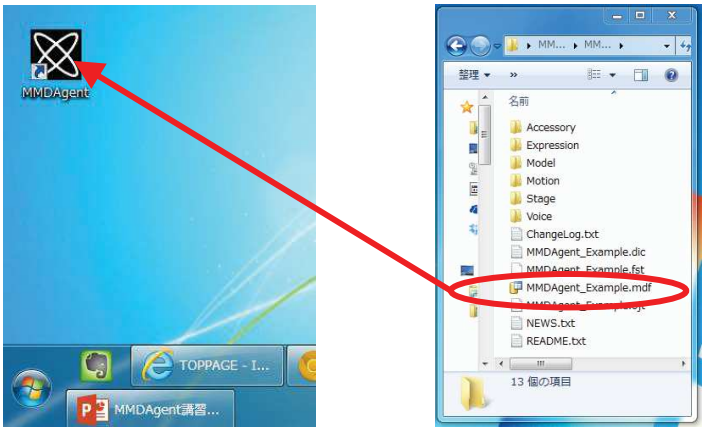


実行結果

- 実行に成功すると図のようになる
  - ３Dキャラクタ（メイちゃん）の表示
  - 背景（壁・床）の表示
  - 回転パネルの表示
  - 音量バーの表示

実行（デスクトップ上にMMDAgentショートカット作成法）

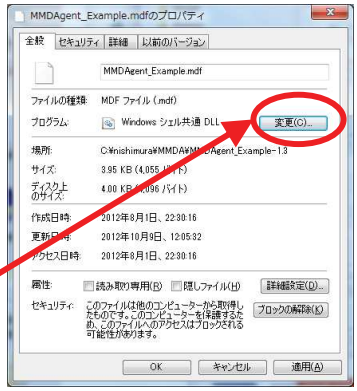
- MMDAgent.exeへのショートカットを作成
  - MMDAgent.exeを右ドラッグ→デスクトップでドロップ（離す）
  - 「ショートカットをここに作成」を選択
- 作成したMMDAgentショートカットに対してD&D
  - .mdfファイルを，MMDAgentのショートカットにドロップする

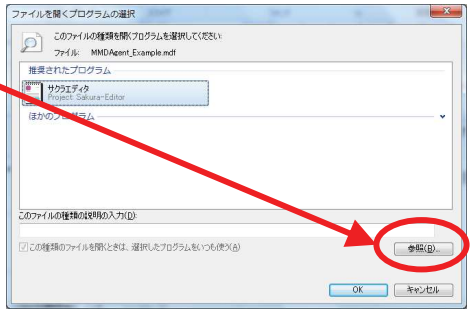


mdfファイルを
ショートカットの上に
ドラッグ &ドロップ

.mdfファイルをMMDAgentに関連付け

- .mdfファイルを右クリック
  - 「プロパティ」を選択
- プロダクラム：「変更」を選択
- 「参照」を選択
  - MMDAgent.exeを指定





以降は，.mdfをダブルクリックすることで
MMDAgentを起動できる

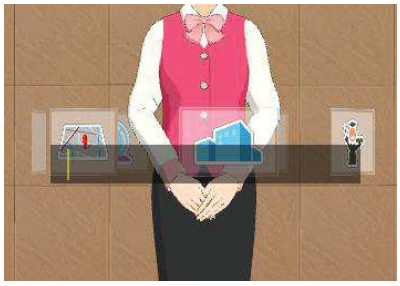
おためし対話（設定調整）

- マイクに向かって発話してみる
  - 音量バーが黄色線を超えると認識開始
  - 音量オーバーだと赤くなる（ダメ）
- 対話できる内容（サンプル）
  - こんにちは
  - あなたは誰？
  - 図書館はどこ？
  - バイバイ
  - ありがとう

音声認識のコツ

- 音声認識のしゃべり方のコツ
  - ゆっくり，はっきり，丁寧に
    - 外国人にしゃべっているような感じで
  - アナウンサーになった気分で
  - ゆっくり過ぎてはダメ
    - 「こーんーにーちーはー」はNG
  - 1単語よりも，文で話すほうが良い

マイク音量の調整



- MMDAgentのレベルメータで、
  黄色の線を超えると音声認識を開始
  - 話しかけると黄色い線を超えるように調整
- 設定方法
  - 推奨：マイク→プロパティのレベルを調整
  - 推奨：音量調整機能付きマイクアンプ*を利用
  - Jconf.txtの「-lv」の値を調整(1～32767)
    （黄色い線の位置を変更する）

* たとえば、audio-technicaのAT-MA2など

マイク音量の調整のTIPS

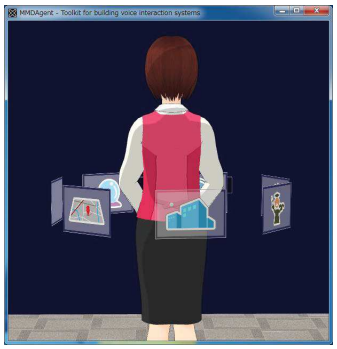
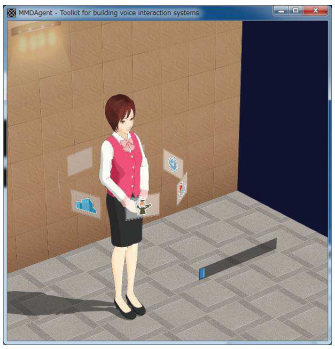
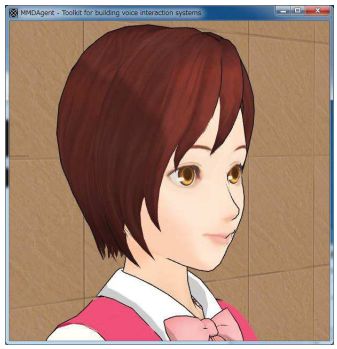
- マイクブースト*は避ける（録音音質の悪化）
  - マイクの音量が足りない場合は、マイクの変更、マイクアンプの利用、ＵＳＢオーディオの利用などを検討
- 騒音下では指向性マイクの利用も検討
  - 指向性マイク ECM-VG1
    - 要ファンタム給電。TASCAM US-366*などを利用
- マイクとスピーカの位置を調整
  - マイクと人の口の距離を可能な限り近づけ、
    マイクとスピーカの距離を可能な限り離す

* マイクデバイスによってはマイクブーストの設定項目は無い

* 要ファームウェア更新。バージョン1.0はバグ有

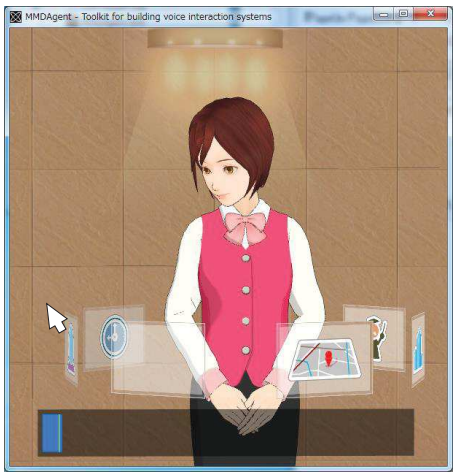
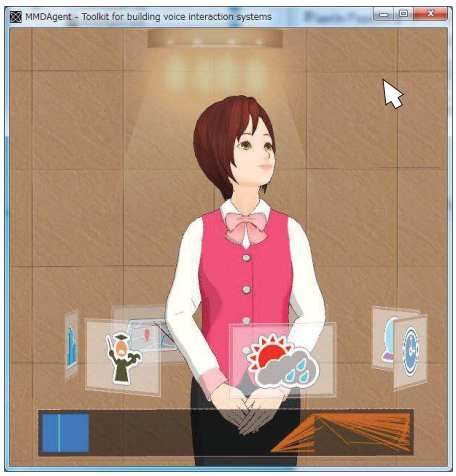
マウスで動かしてみよう

- マウスを使って画面を動かすことができる
  - ドラッグ　　：視点回転
  - Shift＋ドラッグ　　：視点移動
  - ホイール　　：ズーム

マウスカーソル視線追従をしてみよう

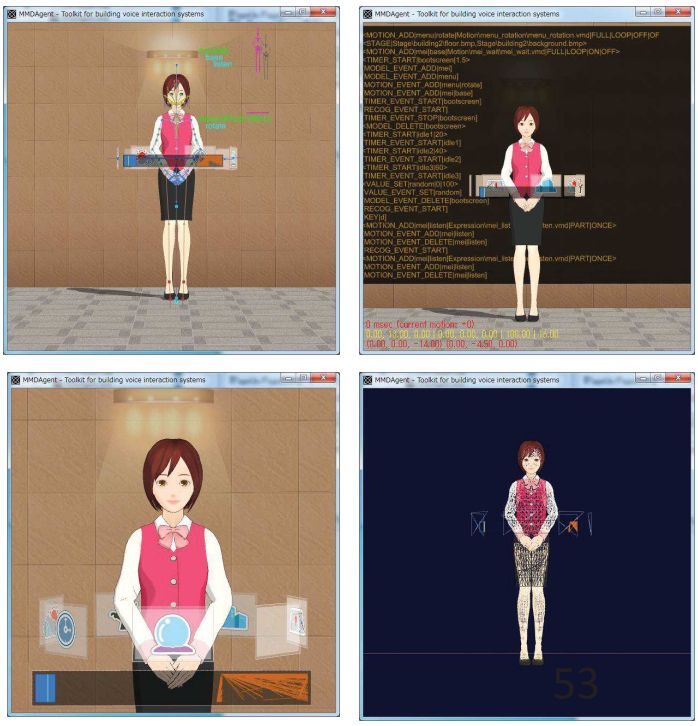
- メイちゃんがマウスカーソルをみる
  - マウスカーソル追従
    - Lキーを押す：マウスカーソルの方を向く
    - Lキーをもう一度押す：解除

キーボードを押してみよう

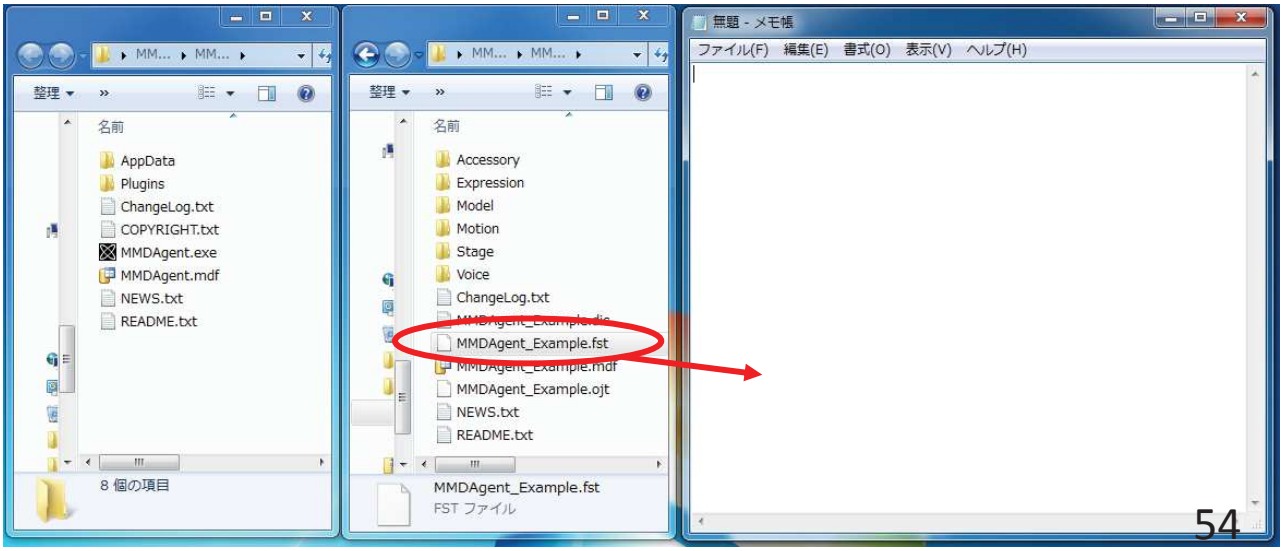
- 各キーに色々な機能が割り当てられている
  - カーソルキー：視点回転
  - shift＋カーソル：視点移動
  - プラス/マイナス：ズーム
  - Esc：プログラム終了
  - D：ログ表示
  - H：モーション停止
  - Shift＋J：音量バー表示
  - B：ボーン表示
  - F：フルスクリーン表示

  （などなど！他にもあります）



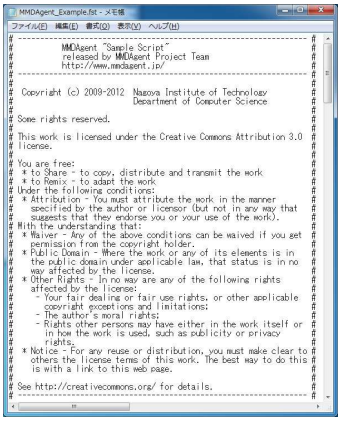
対話のスクリプトファイルを開こう

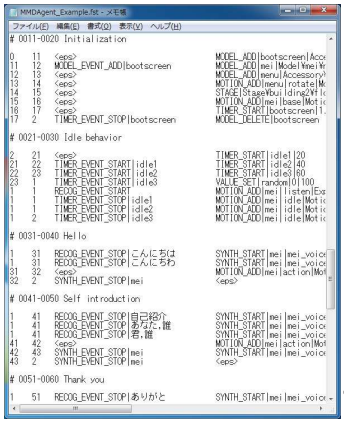
- 対話の内容は，.fstファイルに書かれている
  - エディタ（メモ帳など）で開いてみよう
  - メモ帳を開いてドラッグ＆ドロップ



FSTファイルの構造

- FSTファイルの内容
  - コピーライト
  - スクリプトの命令の説明
  - スクリプト（対話内容）

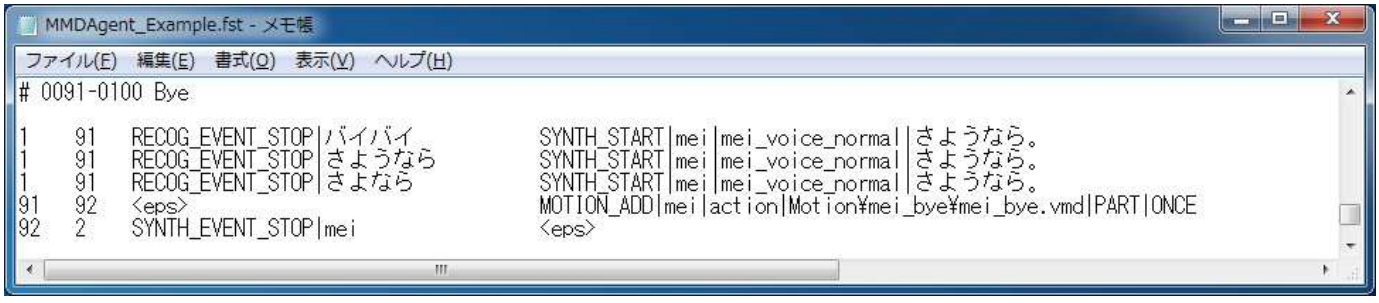




55

スクリプトを見てみよう

- 「バイバイ」の例



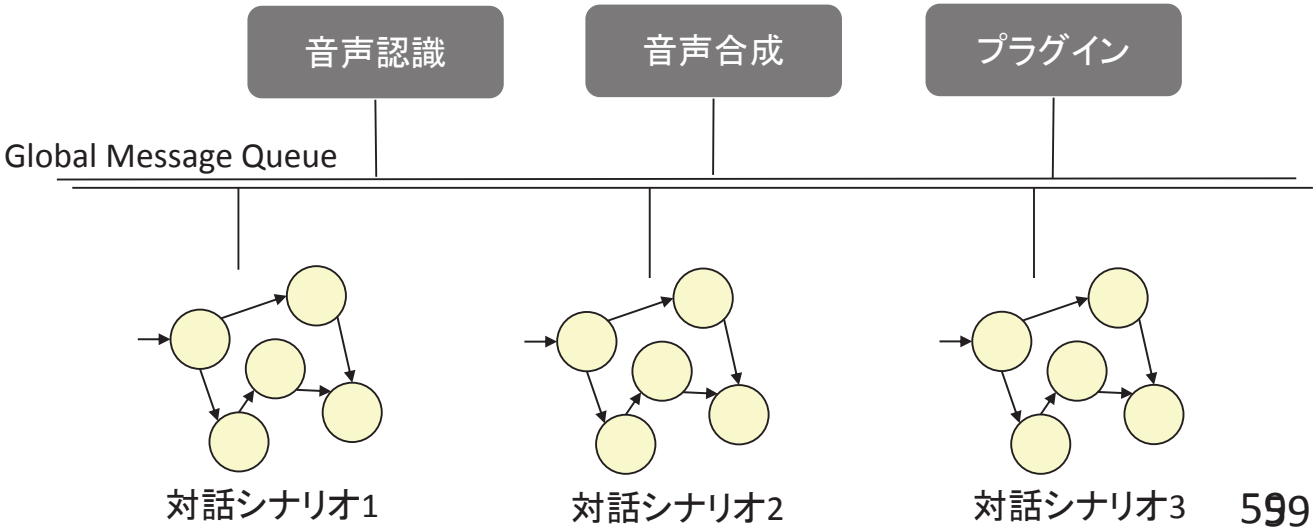
  - 音声出力の内容を変更してみよう
  - 音声合成出力（SYNTH_START）
    - 「さようなら」と音声合成している
    - ここを書き換えることで，音声合成出力の内容を変更できる
    - 注意：スクリプト書き換え後は，MMDAgentを再起動させる

MMDAgent を用いた音声対話システム構築

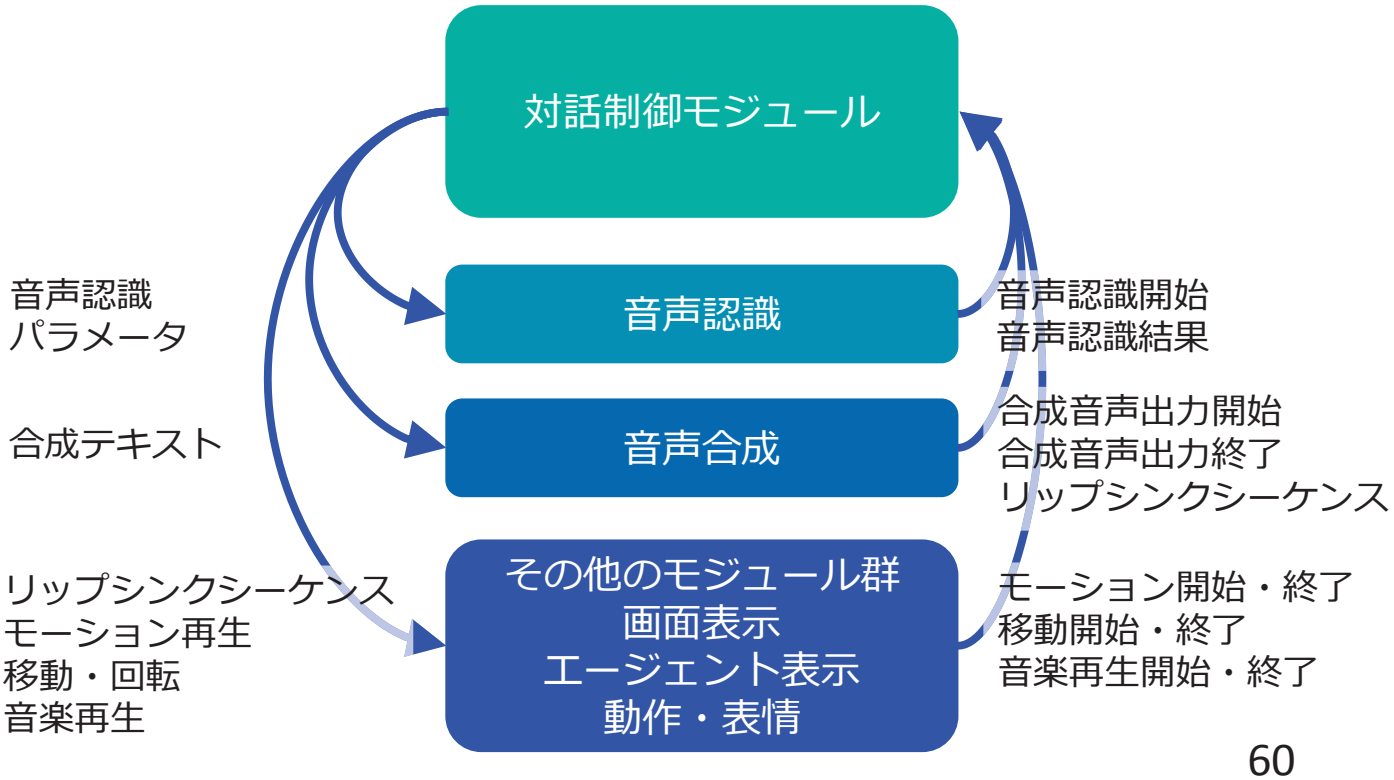
1．しくみの講義

MMDAgent内の情報のやりとり

- 対話シナリオとモジュール（プラグイン）間の連携
  - それぞれが並列かつ独立で動作
  - 「メッセージ」を通じて連携

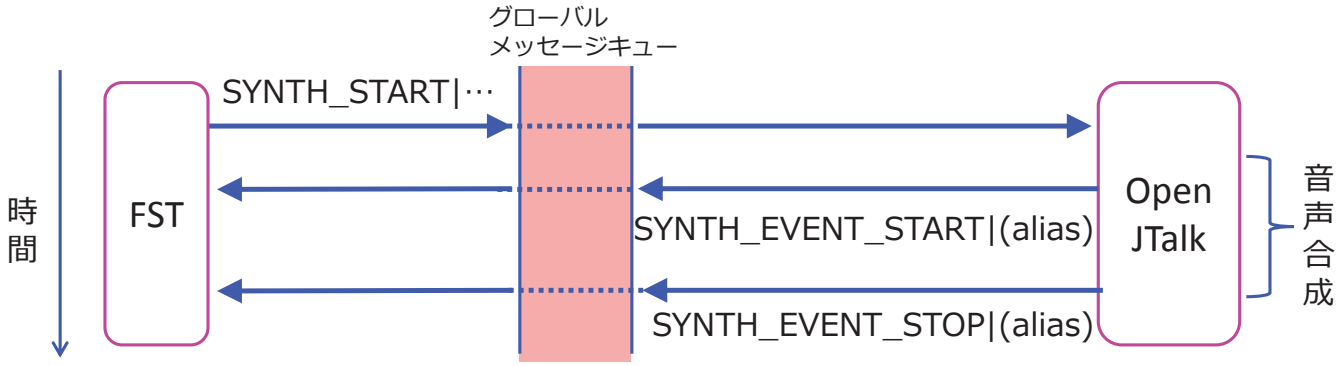


対話制御モジュール



入力・出力メッセージ

- XXX_YYY|alias|…
  - XXX：モジュール名 (SYNTH, MOTIONなど)
  - YYY：コマンド (START, STOPなど)
  - …：パラメータ(|で区切る)
  - aliasはインスタンス名（モデル名や任意の文字列）
  - モジュール開始時と終了時などにEVENT



音声合成(SYNTH)の例

メッセージの詳細

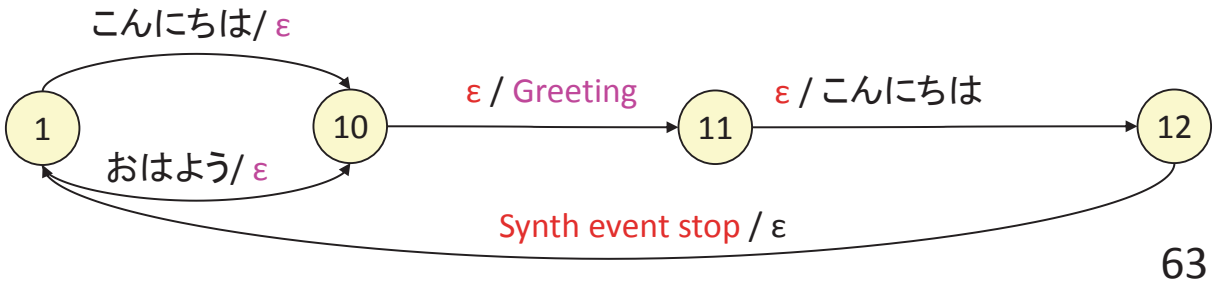
- サンプルスクリプト
  - MODEL_xxx　モデルの追加・削除など
  - MOTION_xxx　動きの追加・削除など
  - SYNTH_xxx　音声合成など
  - RECOG_EVENT_xxx　音声認識のイベントなど
  - VALUE_xxx　（簡易）変数など
  - 他多数

- 詳細はMMDAgent_Example.fstを参照のこと

対話スクリプト(FST)の記述

- 対話スクリプトは有限状態遷移 (FST)で記述
  - 「状態番号、次状態番号、入力条件、出力」のリスト
  - 入力条件を**イベントメッセージ,** 出力を**コマンドメッセージ**

状態番号	次状態番号	入力条件	出力
1	10	RECOG_EVENT_STOP|こんにちは	<eps>
1	10	RECOG_EVENT_STOP|おはよう	<eps>
10	11	<eps>	MOTION_ADD|mei|greet|greet.vmd
11	12	<eps>	SYNTH_START|mei|normal|こんにちは
12	1	SYNTH_EVENT_STOP|mei	<eps>



対話スクリプト

- 一問一答（動作なく答える）
  - 口の動き（リップシンク）は自動で行われる
  - ユーザ　：「あなたは誰ですか？」
  - システム：「メイと言います。」

```
 1  41  RECOG_EVENT_STOP|あなた,誰     SYNTH_START|mei|mei_voice_normal|メイと言います。
41   2  SYNTH_EVENT_STOP|mei           <eps>
```

一問一答

- 動作を加える
  - ユーザ　：「あなたは誰ですか？」
  - システム：「メイと言います。（動作）
    よろしくお願いします。」

```
 1  41  RECOG_EVENT_STOP|あなた,誰   SYNTH_START|mei|mei_voice_normal|メイと言います。
41  42  <eps>            MOTION_ADD|mei|action|Motion¥mei_self_introduction.vmd|PART|ONCE
42  43  SYNTH_EVENT_STOP|mei          SYNTH_START|mei|mei_voice_normal|よろしくお願いします。
43   2  SYNTH_EVENT_STOP|mei          <eps>
```


長い対話のやりとり

- ユーザ　：「暑いね。」
- システム：「夏と冬のどっちが好きですか？」
- ユーザ　：「冬かなー。」
- システム：「クリスマスがありますね．」

```
 1  41  RECOG_EVENT_STOP|暑い
                          SYNTH_START|mei|mei_voice_normal|夏と冬のどっちが好きですか？
41  42  SYNTH_EVENT_STOP|mei    <eps>

42  43  RECOG_EVENT_STOP|夏     SYNTH_START|mei|mei_voice_normal|海に行きたくなりますね。
43  44
 ⋮   ⋮

42  51  RECOG_EVENT_STOP|冬     SYNTH_START|mei|mei_voice_normal|クリスマスがありますね。
51  52
 ⋮   ⋮
```

細やかなインタラクション

- バージイン
  - システムの応答途中で，ユーザが割り込む
- パネル表示

2．ExampleのFST改変

Exampleの改変演習

- 「バイバイ」の例

MMDAgent_Example.fst - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

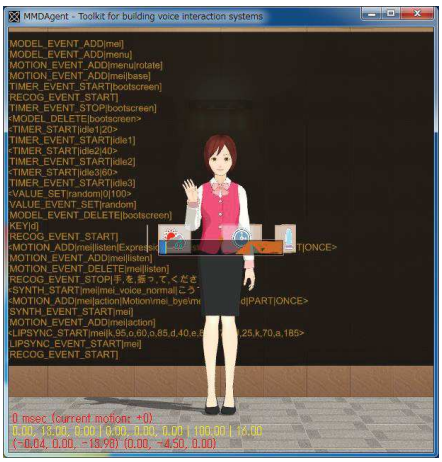
```
# 0091-0100 Bye

1    91    RECOG_EVENT_STOP|バイバイ          SYNTH_START|mei|mei_voice_normal|さようなら。
1    91    RECOG_EVENT_STOP|さようなら        SYNTH_START|mei|mei_voice_normal|さようなら。
1    91    RECOG_EVENT_STOP|さよなら          SYNTH_START|mei|mei_voice_normal|さようなら。
91   92    <eps>                              MOTION_ADD|mei|action|Motion¥mei_bye¥mei_bye.vmd|PART|ONCE
92   2     SYNTH_EVENT_STOP|mei               <eps>
```

- この例では
  - 音声認識終了（RECOG_EVENT_STOP）
    - 「バイバイ」「さようなら」「さよなら」との認識
  - 音声合成出力（SYNTH_START）
    - 「さようなら」と音声合成
  - 手を振るモーション（MOTION_ADD）
  - 音声合成終了待ち（SYNTH_EVENT_STOP）

受け付ける認識結果の書き方

- システムが受ける入力は，単語単位で書く
  - ユーザ入力：「あなたは，誰ですか？」

    →あなた,誰　　と書く
    - 認識単語を区切る**「,」は半角**　　→全角にすると動作しない
    - **全角スペースは使わない！**　　→含まれると動作しない
    - ファイルの保存を忘れずに



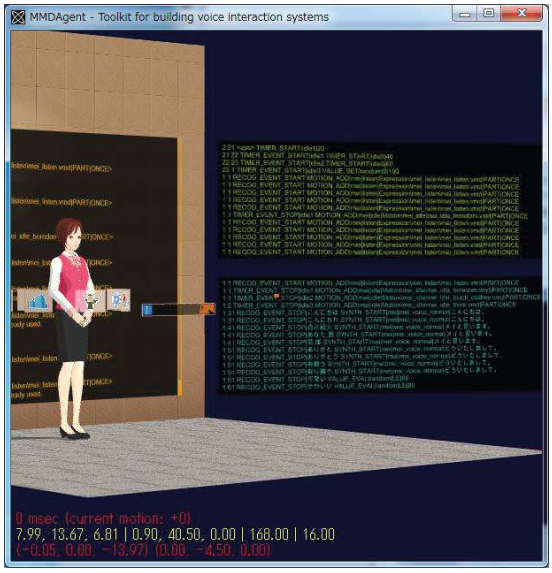
- デバッグ表示を使う
  - 「D」キーを押してデバッグ表示
  - 認識結果が表示される
  - 認識できる単語の確認

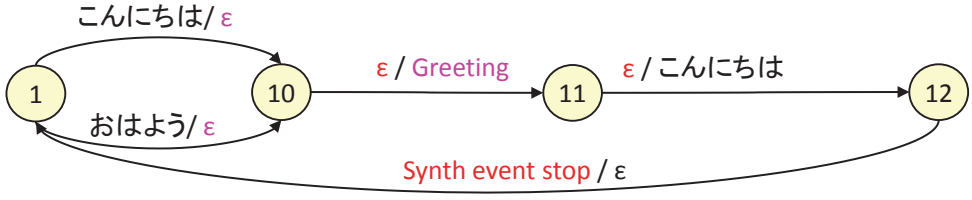
会話して確かめてみよう

- MMDA再起動（新しいFSTを読み込む）
- 入力したことをしゃべって確かめる
  - 「手を振ってください」「手を振って」など
  - 確認時にもデバッグモードにすると良い
    - 「D」キーを押す
    - 音声認識結果が確認できる

FSTログ表示

- FSTのログが表示できる
  - 「Shift」＋「F」キーを押す
  - メイちゃんの右側に表示される
    - 画面をスクロールさせないと見えない
  - 2つの画面が表示される
    - 上画面：これまでの状態遷移の履歴
    - 下画面：次に実行される遷移





音声合成テキストの書き方

- システムが読み間違える場合
  - 句読点で区切る

応答を追加してみよう

- スクリプトを追記
  - 「手を振ってください」というと「こうですか？」と応答
  - 注意点
    - 認識単語は，「手」「振っ」　→「振って」という単語は無い
    - 認識単語を区切る**「,」は半角**　→全角にすると動作しない
    - **全角スペースは使わない！**　→含まれると動作しない
    - ファイルの保存を忘れずに
    - 「# 0091-0100 Bye」をコピーして使うと楽かも

演習課題

演習用のファイルの用意の仕方

- 演習課題毎に新しいファイルを作る
  - .mdf　.fst　を同じファイル名で作成
  - （注意：拡張子を表示させる）
  - .fstは，１から作って頂きたい
  - メッセージの一覧は，.fstの上部にある.

課題（情報提供系）

演習課題 1

- 駅の情報案内システムを作る
- 例）
  - ユーザ　：「新幹線の改札はどこ？」
  - システム：「新幹線の改札は中央を真っ直ぐ進み、突き当りを右に曲がったところにあります」

演習課題２

- 天気予報をしてくれるシステムを作る
- 例）
  - ユーザ　：「明日の東京の天気は？」
  - システム：「明日の東京の天気は晴れです。」

演習課題３

- ホテルのコンシェルジュを作る

例）

- ユーザ　：「田町駅付近でのおいしいレストランを教えてください」
- システム：「はい。それは○○レストランです。」

- ユーザ　：「帰りの新幹線で、指定席の手配をお願いします」
- システム：「はい。○時○分発の○○席をご用意します。」

演習課題４

- 自分（会社社長）の秘書を作る
- 例）
  - ユーザ　：「今日のスケジュールは？」
  - システム：「今日は午前中に会議、午後からは○○社との会食があります。」
  - ユーザ　：「現在の○○社の株価は？」
  - システム：「株価は○○円で、前日比○○%です。」

演習課題５

- ツアーコンダクターを作る
- 例）
  - ユーザ　：「旅館に到着する時刻はいつごろですか？」
  - システム：「はい。○時○分ごろです。」
  - ユーザ　：「東京タワー付近の観光名所は何ですか？」
  - システム：「はい。付近に増上寺という菩提寺がございます。」

演習課題６

- お料理ナビシステムを作る
- 例）
  - ユーザ　：「カルボナーラソースの作り方は？」
  - システム：「卵2つと粉チーズ大さじ1杯をかき混ぜるとできます。」

  - ユーザ　：「ピーマンを使ったおいしい料理を教えて」
  - システム：「ピーマンの肉詰めはどうでしょうか。」

演習課題７

- 占いを行うシステムを作る
- 例）
  - ユーザ　：「今日の獅子座の運勢は？」
  - システム：「今日の獅子座の運勢は最悪です。」

演習課題８

- 気分に合わせて音楽紹介をするシステムを作る
- 例）
  - ユーザ　：「楽しいときに聞きたい音楽を教えてください」
  - システム：「ショパンの『子犬のワルツ』はどうですか。」

演習課題９

- パソコン関係の専門用語を説明するシステムを作る
- 例）
  - ユーザ　：「CPUとは何ですか？」
  - システム：「データの処理や演算を行うコンピュータの
    中枢を担う装置のことです。」

演習課題１０

- 家電製品などの使用状況を伝えるシステムを作る
- 例）
  - ユーザ　：「石油ストーブにある灯油の量を教えてください」
  - システム：「現在、全体の7%です。灯油を補充してください。」

  - ユーザ　：「エアコンの消費電力はどのくらいか教えてください」
  - システム：「○○kWhです。使いすぎていませんか？」

課題（知識埋め系）

演習課題 1

- 歴史上の人物について何でも知っている
  エージェントを作る
- 例）
  - ユーザ　：「織田信長ってどんな人？」
  - システム：「天下統一を目指した戦国武将です。
    よく奇妙な行動を取っていたので
    尾張の大うつけと呼ばれていました。」

演習課題 2

- ワールドカップについて何でも知っている
  エージェントを作る
- 例）
  - ユーザ　：「日本代表選手のキャプテンは誰？」
  - システム：「長谷部誠です。ドイツの
    ブンデスリーガに所属しています。」
  - ユーザ　：「どのチームが勝つと思いますか？」
  - システム：「ブラジルです。ネイマール選手が
    活躍すると思います。」

演習課題３

- アイドルについて何でも知っている
  エージェントを作る
- 例）
  - ユーザ　：「東京のご当地アイドルは何かありますか？」
  - システム：「はい。AKB48です。」

  - ユーザ　：「あなたが好きなアイドルは何ですか？」
  - システム：「○○です。理由は・・・」

演習課題４

- 東京について何でも知っている
  エージェントを作る
- 例）
  - ユーザ　：「東京スカイツリーの高さは？」
  - システム：「634mです。」

  - ユーザ　：「東京都の花と言えば何ですか？」
  - システム：「ソメイヨシノです。」

演習課題５

- あるアニメのキャラクターについて
  何でも知っているエージェントを作る
- 例）
  - ユーザ　：「身長は？」
  - システム：「129.3cmです。」

  - ユーザ　：「好きな食べ物は？」
  - システム：「ドラ焼きです。」

演習課題６

- 自分のカバンの中にあるものについて
  何でも知っているエージェントを作る
- 例）
  - ユーザ　：「この青いケースは何ですか？」
  - システム：「眼鏡ケースです。家に後3つほどあります。」

  - ユーザ　：「この本は何ですか？」
  - システム：「推理小説です。電車の中でいつも読んでいます。」

演習課題 7

- 映画について何でも知っている
  エージェントを作る
- 例）
  - ユーザ　：「今月公開の映画で
    おすすめなものはありますか？」
  - システム：「アンジェリーナ・ジョリー主演の
    『マレフィセント』です。」

演習課題 8

- 食べ物（グルメ）について何でも知っている
  エージェントを作る
- 例）
  - ユーザ　：「東京のラーメンの特徴を教えてください」
  - システム：「醤油ベースで和風だしの効いたラーメンが特徴です。」

  - ユーザ　：「東京付近で最近の話題の食べ物はありますか？」
  - システム：「渋谷の『アンドザフリット』はどうでしょう。」

演習課題９

- 音楽について何でも知っている
  エージェントを作る
- 例）
  - ユーザ　：「最近の発売された曲で良い曲はありますか？」
  - システム：「はい。○○です。」

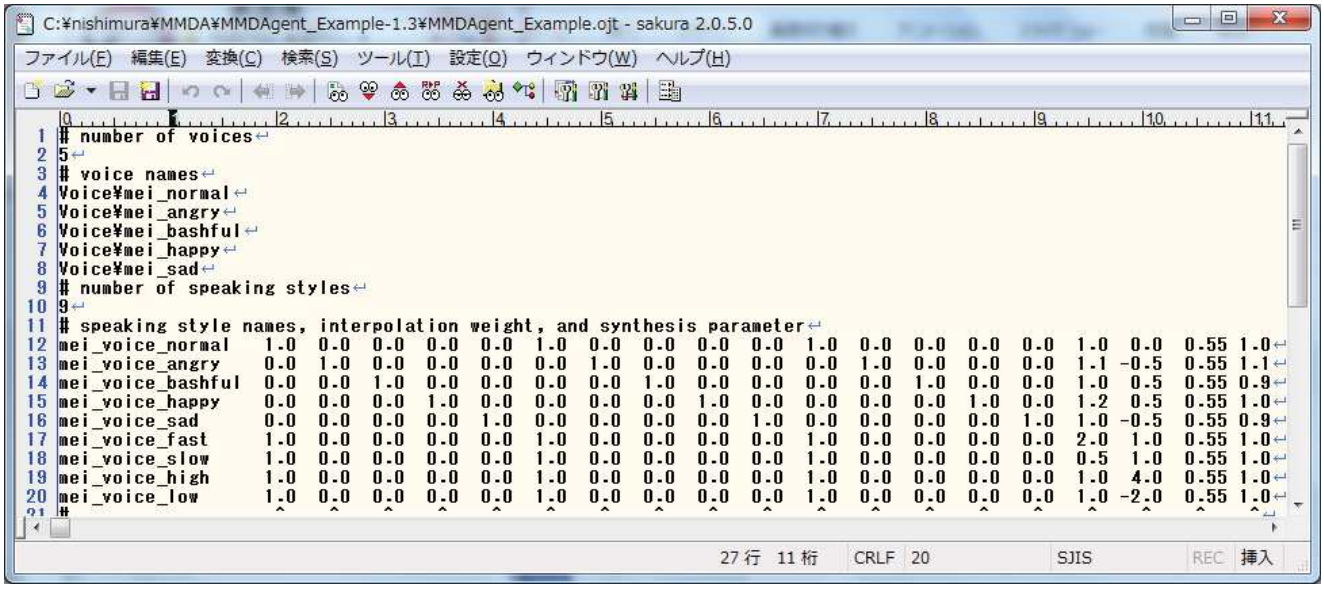
演習課題１０

- オリンピックについて何でも知っている
  エージェントを作る
- 例）
  - ユーザ　：「次の夏季オリンピックの開催地はどこですか？」
  - システム：「リオデジャネイロです。」

課題（その他）

補足課題

- 音声合成のパラメータ変更
  - 声質を変更　（.ojtファイル）



最後に

さいごに

- 編集したファイルの提出のお願い（任意）
  - 音声対話システム構築に関する研究に用いる
  - 同意して頂ける方は，今回作成したものを提出してください
    - 提出するもの：.fstファイル，及び関連ファイル
    - 提出方法：メール（or USBメモリ）
- 講習会後にアンケートのお願いがあります
  - ぜひご回答頂きたいと思います

より表現豊かな音声対話システム

マウス操作

- ドラッグ　視点（カメラ）の回転
- Shift+ドラッグ　視点（カメラ）の並進
- マウスホイール　視点（カメラ）の前進後退
- CTRL+Shift+ドラッグ　光源方向の回転

- モデル上で：
- CTRL+ドラッグ　XY平面移動
- CTRL+Shift+ドラッグ　XZ平面移動
- ダブルクリック　モデルを選択

キー操作（１）

- 矢印キー　　視点の回転
- Shift＋矢印キー　　視点の並進
- “+” “-”　　視点の前進・後退

- モデル選択してDEL　　モデル削除

- PAGE UP/DOWN　　ログのスクロール

キー操作（２）

- D　　ログ表示
- F　　フルスクリーン
- S　　FPS表示
- X　　シャドウマップ

- W　　Wire表示
- Shift+W
  物理剛体表示
- B　　ボーン表示

- E, Shift+E　エッジの太さ
- P　　物理演算ON/OFF
- H　　ホールド
- V　　垂直同期ON/OFF

演習

- キー操作をいろいろ試してみましょう.

ファイル構成

MMDAgent ディレクトリ

- <Sysdir>
- |- MMDAgent.exe　MMDAgent本体
- |- MMDAgent.mdf　システム共通設定
- |- AppData/　システムデータ
- |- Plugins/　プラグイン
- （以下は .exe のみ起動した時に読み込まれる）
- |- MMDAgent.fst
- |- MMDAgent.dic
- |- MMDAgent.ojt

個別のFSTのディレクトリ

- <ContentsDir>
- |- [パス名].mdf　個別設定
- |- [パス名].fst　対話スクリプト
- |- [パス名].dic　認識ユーザ辞書
- |- [パス名].ojt　合成ボイス設定

読み込まれるファイル

- <Sysdir>
- |- MMDAgent.exe　MMDAgent本体
- |- MMDAgent.mdf　システム共通設定
- |- AppData/　システムデータ
- |- Plugins/　プラグイン

（以下は .exe のみ起動した時に読み込まれる）

- |- MMDAgent.fst
- |- MMDAgent.dic
- |- MMDAgent.ojt

- <ContentsDir>
- |- [パス名].mdf　個別設定
- |- [パス名].fst　対話スクリプト
- |- [パス名].dic　認識ユーザ辞書
- |- [パス名].ojt　合成ボイス設定

.exe のみで起動

.mdf を指定して起動
（関連付け・D&D等）

システムデータ

- <SysDir>/AppData
- |- lip.txt　リップシンク用音素・モーフ対応定義
- |- OpenJTalk/　音声合成（OpenJTalk）ボイス定義
- |- Julius/　音声認識（Julius）用ファイル
  - |- lang_m/　言語モデル
  - |- phone_m/　音響モデル
  - |- jconf.txt　Jconf設定ファイル

.mdf ファイル

- 1) システム共通設定
  - <SysDir>/MMDAgent.mdf
- 2) FSTごとの設定
  - <ContentsDir>/[パス名].mdf

1) -> 2) の順に上書きで読み込まれる

以下，主要な設定の説明（値はデフォルト値）

設定：システム

- # Window
- window_size=600,600　　初期ウィンドウサイズ
- full_screen=false　　フルスクリーンで起動
- # OpenGL
- max_multi_sampling=4　　マルチサンプリング数
- # Misc
- max_num_model=10　　最大表示モデル数
- display_comment_time=5.0　　モデルコメント表示時間(秒)

設定：セルシェーディング

- # Cartoon rendering
- use_cartoon_rendering=true　　ON/OFF
- use_mmd_like_cartoon=true　　MMD互換性
- cartoon_edge_width=0.7　　エッジ幅
- cartoon_edge_step=1.2　　エッジ調整幅
- cartoon_edge_selected_color=1.0,0.0,0.0,1.0
  選択時エッジ色

設定：ステージ

- # Stage
- stage_size=25.0,25.0,40.0　　ステージ WxDxH
- # Campus color
- campus_color=0.0,0.0,0.2　　背景空間色
- # Light color
- light_direction=0.5,1.0,0.5,0.0　光源方向
- light_intensity=0.6　　光の強さ
- light_color=1.0,1.0,1.0　　光の色

設定：表示

- # Fps
- show_fps=true　FPS表示ON/OFF
- fps_position=-2.5,22.0,3.0　FPS座標
- # Log
- log_size=80,30　ログ画面WxH
- log_position=-17.5,3.0,-15.0　ログ座標
- log_scale=1.0　ログスケール

設定：物理演算

- # Bullet Physics
- bullet_fps=120　演算フレーム/秒
- gravity_factor=2.0　重力スケール係数

設定：モーション

- #motion
- motion_adjust_time=0.0

  モーション開始タイミング微修正（秒）
- lipsync_priority=100.0

  リップシンクのモーション優先度

演習

- ・MDFファイルをいじってみましょう
  - max_multi_sampling=1
  - display_comment_time=0.0
  - use_cartoon_rendering=false
  - 表示関連，等

認識用辞書の変更・語彙追加

- 資料
- １） Tips_Reference.pdf 内 ４ページ目
- 1.5節「認識単語を辞書に新規追加する」
- ２）MMDAgent_Example.dic

- 演習：辞書にない単語を追加して使ってみましょう
- （スポーツ選手の名前，海外の土地名，商品名等）
- ヒント： 単語の区切れ目は実際の認識結果を見て！

モデルの追加・変更

- **方法１）**PMDファイルをモデル上にドロップする
  →そのモデルが切り替わる

- **方法２）**PMDファイルをCTRLを押しながらドロップ
  →新たに追加される

- **方法３）**MODEL_ADD, MODEL_CHANGEメッセージ

MODEL_ADD

モーションの変更

- **方法１）**.vmdファイルをモデルへドロップ
  →”base” という名前のモーションを入れ替え
- **方法２）**.vmdファイルをShift＋モデルへドロップ
  → 部分モーションとして重ね実行
- **方法３）**CTRLを押しながら１）あるいは２）
  →全てのモデルへ上記を適用
- **方法４）**MOTION_ADD メッセージ

MOTION_ADD

演習

- PMD, VMDファイルは Example の中にあります
- １）エージェントを増やしてみましょう
  - ドロップで
  - FSTで
- ２）モーションを選んで実行してみましょう
  ドロップで
  FSTで
- （暇な方）エージェントどうしで会話するFSTを考えてみましょう

MMDAgent_Example.fst 解説

- ・パネル表示
- ・バージイン
- Tips_Reference.pdf の 1.6 節も参考に

声質を変える

- .ojt ファイルを編集する
- Tips_Reference.pdf の2.2節を参考に

モーション・Modelの作成の紹介

- MikuMikuDance
- PMDEditor

システム拡張

- プラグインによる拡張
- 規定名の関数を定義・EXPORTしたDLLを作成する

関数名	呼び出されるタイミング
extAppStart	起動時
extAppEnd	終了時
extProcMessage	メッセージ受信時
extUpdate	画面更新時（毎フレーム）
extRender	レンダリング時（毎フレーム）

デフォルトプラグイン

- Plugin_Julius　音声認識
- Plugin_Open_Jtalk　音声合成
- Plugin_Audio　オーディオ再生
- Plugin_LookAt　マウス視線追従
- Plugin_WindowController　ファイル実行等
- Plugin_Variable　変数
- Plugin_VIManager　対話管理（FST）

MMDAgent Tips

目 次

1　音声認識

1.1　音声入力レベルの調整

MMDAgent で音声がうまく認識されないときは，まず音声入力がうまく検出されているかを以下の方法で確認しましょう．

認識モジュールの状態が，下図のようなインジケータに表示されます．青いバーが現在の音声入力レベルを表します．音声認識部が一定以上の音の大きさを検出すると認識が開始されます．認識処理中は，右側からオレンジ色の線が表示されていきます．入力が終了すると同時にオレンジ色の線が消え，認識結果が出力されます．

話していない状態でバーが青，しゃべっている間だけバーがオレンジ色になるか確認してみてください．

- 喋ってもバーが動かない場合
  マイクがきちんと接続されているか，規定の録音デバイスが正しく選択されているか，デバイスがミュートされていないか，確認してください．
- 喋ったときのバーの動きが小さく，音声認識が開始しない場合
  録音デバイスの入力レベルが小さすぎます．入力レベルを大きくしてみてください．
- 喋ったときバーが右端まで達してしまったり，ときおり赤くなる場合
  オーバーフローしています．録音デバイスの入力レベルを小さくしてみてください．
- 喋っていないときもずっと音声認識中になってしまうとき
  周囲の雑音を誤検知しつづけている状態です．録音デバイスの入力レベルを小さく調整してみてください．また周囲の雑音を抑えてください．

なお，音声認識エンジン Julius の設定やメッセージでマイク感度の調整を行うことも可能です．

1.2 音声認識の精度が低い場合のコツ

音声認識の動作は周囲の雑音，マイクの特性，発声様式等に大きく左右されます．音声認識率が低いときは，以下のような点に気をつけると改善されることがありますので意識してみましょう．

- 口を大きく動かしてはっきりと発音する．
- 適度な速度で話す．
- 静かな場所で使い，周りの雑音が入るのを抑える．
- 外付けの性能の良いマイクを使う．

「D」キーを押すと出てくるログに認識結果が表示されます．それを見ながら話し方を調節するとよいでしょう．

なお，上記は一般的な音声認識のアドバイスであり，個人差により効果が少ない場合があります．

1.3 音声認識のパラメータ調整

MMDAgent では音声認識に，大語彙連続音声認識エンジン Julius を使っています．設定ファイルは `AppData\Julius\jconf.txt` で，これを編集することで音声認識の設定やパフォーマンスの調整を行うことができます．ここでは設定できる項目のうち，代表的なものを説明します．なお，それぞれ記されている数値はデフォルトの値です．

- -lv 1500
  設定値より大きい入力音声の振幅が一定時間続けば音声認識開始とみなされます．設定値を大きくすると，感度が鈍くなります．なお，下図のキャラクターの前に表示されている青いバーが入力音声の振幅であり，黄色い線がこの設定値です．

- -b 800
  探索幅です．設定値が大きいと認識中にたくさんの候補を扱うようになり，計算量はかかるようになりますが精度は安定します．逆に小さくするほど候補を絞るので認識処理が高速になりますが，精度が悪くなる可能性が高くなります．

- -rejectshort 700
  検出した音の長さ（ミリ秒）がこの設定値未満の時，認識を行わずに入力を無視します．鋭い物音等で音声認識が意図せず開始されるのを防ぐ役割を持っています．

- -tailmargin 240
  音の長さの終了部のマージン（ミリ秒）です．設定値を小さくすると，発話終了の検出が早くなりますが，発話終了の検出が不安定になる可能性があります．大きくすることで安定しますが，認識結果が出力されるのが遅くなります．

上記を含む，設定可能な全てのパラメータについては，Julius の Web ページにある “The Julius-book” の julius の節にあります．参考にしてください．

1.4 音声認識結果で複数の単語を用いる方法

MMDAgent では音声認識の結果をカンマ区切りの単語列として取得しています.「D」キーを押すと，ログを見ながら自分の発話がどのように認識されているかを確認できます.

認識結果に反応する動作を音声対話スクリプト (.fst) に書く際は,「今日」「天気」のように認識結果をキーワードで指定することでそのキーワードを含む発話の認識結果を捉えることができます．例えば，ある単独のキーワードに反応させるには，以下のように記述します.

```
1 11 RECOG_EVENT_STOP|こんにちは SYNTH_START|mei|mei_voice_normal|
こんにちは
```

また複数のキーワードを指定することが可能です．AND 条件，すなわち複数のキーワードが全て含まれる時に反応させたいときは，下記のように半角コンマ (,) で区切って複数のキーワードを記述します．この部分は，書かれた全てのキーワードが，1 発話の中で登場した時にだけ反応します．なお，半角コンマの前後にはスペースを入れないでください.

```
1 11 RECOG_EVENT_STOP|名古屋,天気 SYNTH_START|mei|mei_voice_normal|晴
れです.
```

OR 条件，すなわち複数のキーワードのどれかが発話に含まれる時に反応させたい場合は，下記のようにそれぞれのキーワードに対する反応を並列に記述します．なお，音声対話スクリプト (.fst) では，同じ状態から複数の遷移が定義されている場合，.fst 内での記述順に上から適用されます.

```
1 11 RECOG_EVENT_STOP|昼ごはん SYNTH_START|mei|mei_voice_normal|A
ランチがおすすめです.
1 11 RECOG_EVENT_STOP|ランチ SYNTH_START|mei|mei_voice_normal|A
ランチがおすすめです.
1 11 RECOG_EVENT_STOP|昼食 SYNTH_START|mei|mei_voice_normal|A
ランチがおすすめです.
```

1.5 認識単語を辞書に新規追加する（ユーザ辞書ファイル）

ユーザ辞書ファイルを作成することで，辞書にない単語を登録して認識させることができます.

ユーザ辞書ファイルの場所は，起動の方法によって異なります．設定ファイル (.mdf) からダブルクリック，あるいはドラッグアンドドロップで MMDAgent を起動した時は，その設定ファイル (.mdf) と同じフォルダの，拡張子を「.dic」に換えた辞書ファイル (.dic) を読み込みます．また，実行ファイル (.exe) を直接起動した時は，その実行ファイル (.exe) と同じパスにある辞書ファイル (.dic) を読み込みます.

辞書ファイル (.dic) はテキスト形式です．以下のように，追加する単語を記述します.

```
東京:トーキョー:東京:515 @1.0 鶴舞:ツルマイ:鶴舞:515 [鶴舞] ts u r u m a i
花子:ハナコ:花子:513 @2.0 メイ:メイ:メイ:513 [メイ] m e i
花子:ハナコ:花子:513 @2.0 メイ:メー:メイ:513 [メイ] m e:
<unk> @0.0 <unk> [ヘロー] h e r o:
```

第 1 フィールドは単語の出現を表す文字列を指定します．システム辞書内にある単語の中で似た単語の第 1 フィールドを指定します．音声認識時には，ここで指定された単語の出現パターンがそのままこの新単語に適用されます．上の最初の例では「鶴舞」は地名であり，システム辞書内にある「東京：トーキョー：東京：515」を指定することで出現を定義しています．分からない場合は「<unk>」と記述して，下記の第 2 フィールドの補正値を変えることで手動で調整することもできます．

第 2 フィールドの@の後の数字は出現確率（対数確率）の補正値で，デフォルトは 0 です．音声認識中には，単語の出現確率（対数）にこの値が加算されます．単語が認識されにくいときは，この値を大きく設定することで，認識結果に出現しやすく調整することができます．ただし，大きくし過ぎると別の発話でもその単語が誤って出現する確率が高まってしまうので注意が必要です．

第 3 フィールドにはその単語の登録情報を，表記:発音:基本形の順で記述します．数字は第 1 フィールドと同じ物を使用してください．また第 1 フィールド同様，分からない場合は「<unk>」で構いません．

第 4 フィールドの [] 内に書かれたものは，認識結果として出てくる単語となります．

第 5 フィールドは読みを表す音素列を表します．カタカナ読みから音素列への変換は表 1 の音素対応表に従ってください．なお，促音・長音については表 2 のルールに従います．

MMDAgent の AppData フォルダ内にシステム辞書が存在します．システム辞書も上記のユーザ辞書と同じフォーマットで書かれていますので，こちらを調整することも可能です．例えば，誤認識結果としてよく現れてしまう単語については，システム辞書上で削る，等の対処を行うことができます．

ただし，上記はあくまで学習された既存の言語モデルに対する微調整です．数ユーザ辞書の大きさが単語〜数十単語であれば問題ありませんが，大きくなると調整が難しくなります．より認識精度を高めたい場合は，音声認識用の言語モデル（単語 N-gram）および辞書を自分で構築する，あるいは Julius の孤立単語認識や文法ベース認識を使うなど，音声認識システム全体を用途に合わせてカスタマイズする方法があります．

表 1: 音素対応表

ア	イ	ウ	エ	オ
a	i	u	e	o
カ	キ	ク	ケ	コ
k a	k i	k u	k e	k o
サ	シ	ス	セ	ソ
s a	sh i	s u	s e	s o
タ	チ	ツ	テ	ト
t a	ch i	ts u	t e	t o
ナ	ニ	ヌ	ネ	ノ
n a	n i	n u	n e	n o
ハ	ヒ	フ	ヘ	ホ
h a	h i	f u	h e	h o
マ	ミ	ム	メ	モ
m a	m i	m u	m e	m o
ヤ		ユ		ヨ
y a		y u		y o
ラ	リ	ル	レ	ロ
r a	r i	r u	r e	r o
ワ		ヲ		ン
w a		o		N
ガ	ギ	グ	ゲ	ゴ
g a	g i	g u	g e	g o
ザ	ジ	ズ	ゼ	ゾ
z a	z i	z u	z e	z o
ダ	ヂ	ヅ	デ	ド
d a	z i	z u	d e	d o
バ	ビ	ブ	ベ	ボ
b a	b i	b u	b e	b o
パ	ピ	プ	ペ	ポ
p a	p i	p u	p e	p o

キャ		キュ		キョ
ky a		ky u		ky o
シャ		シュ	シェ	ショ
sh a		sh u	sh e	sh o
チャ		チュ	チェ	チョ
ch a		ch u	ch e	ch o
ニャ		ニュ		ニョ
ny a		ny u		ny o
ヒャ		ヒュ		ヒョ
hy a		hy u		hy o
ミャ		ミュ		ミョ
my a		my u		my o
リャ		リュ		リョ
ry a		ry u		ry o
ギャ		ギュ		ギョ
gy a		gy u		gy o
ジャ		ジュ	ジェ	ジョ
j a		j u	j e	j o
ビャ		ビュ		ビョ
by a		by u		by o
ピャ		ピュ		ピョ
py a		py u		py o
ファ	フィ	フュ	フェ	フォ
f a	f i	hy u	f e	f o
	ウィ		ウェ	ウォ
	w i		w e	w o
ヴァ	ヴィ		ヴェ	ヴォ
b a	b i		b e	b o
ツァ	ツィ		ツェ	ツォ
ts a	ts i		ts e	ts o
	ティ	テュ		
	t i	t u		
	ディ	デュ		
	d i	d u		
		ドゥ		
		d u		

1.6 バージインの記述例

音声インタフェースではシステムの発話中にユーザがその発話を遮って音声入力を行うことがあります．これをバージインといいます．

例えば，以下のように音声対話スクリプト (.fst) を記述することでバージインをある程度扱うことができます．このスクリプトは，エージェントの発話中にユーザが話しかけるとその時点で音声合成を中断するだけですが，最終行からさらに動作を追加すれば，より人間に近いバージインの対処が記述できるでしょう．

```
1 10 RECOG_EVENT_STOP|こんにちは SYNTH_START|mei|mei_voice_normal|こんにちは。

10 20 SYNTH_EVENT_STOP|mei SYNTH_START|mei|mei_voice_normal|私の名前はメイと言います。
```

表 2: 促音・長音の表記

促音（ッ）	q	（例）ラッコ	r a q k o
長音（ー）	a: i: u: e: o:	（例）チーター	ch i: t a:

20 30 SYNTH_EVENT_STOP|mei SYNTH_START|mei|mei_voice_normal|私は情報案内ができます。

30 50 SYNTH_EVENT_STOP|mei SYNTH_START|mei|mei_voice_normal|ご用件をおっしゃってください。

```
10 40 RECOG_EVENT_START SYNTH_STOP|mei
20 40 RECOG_EVENT_START SYNTH_STOP|mei
30 40 RECOG_EVENT_START SYNTH_STOP|mei
40 50 SYNTH_EVENT_STOP|mei SYNTH_START|mei|mei_voice_normal|はい。
50  2 SYNTH_EVENT_STOP|mei <eps>
```

2 音声合成

2.1 音声合成関連のメッセージ

SYNTH_START コマンドの第 1 引数にリップシンクを行うモデルのエイリアス名，第 2 引数に「Definition of speaking style」のように定義されている発話スタイル名，第 3 引数に喋らせたい文を記述します．

発話を途中で終了させるには SYNTH_STOP コマンドを用います．第 1 引数にはモデルのエイリアス名を記述します．

発話終了時には SYNTH_EVENT_STOP イベントが発行されます．

2.2 発話スタイル定義ファイル (.ojt)

MMDAgent の音声合成で指定できる発話スタイルは，発話スタイル設定ファイル (.ojt) で定義しています．この発話スタイル設定ファイル (.ojt) では，発話スタイルごとにボイスの混合重み，持続時間，音高などの音声合成用パラメータを指定します．

MMDAgent の Example には 4 種類の異なる感情のボイスが含まれています．.ojt ファイルを変更することで，これらを任意の比率で混ぜ合わせたり，発話速度や声の高さ等を自由に変更したりできます．

以下は，normal, angry, bashful の 3 種類のボイスがフォルダ Voice 以下に存在するときのいくつかの発話スタイルの定義例です．なお，#以降はコメントです．

```
# number of voices
3
# voice names
Voice\normal
Voice\angry
Voice\bashful
```

```
# number of speaking styles
7

# speaking style names, interpolation weight, and synthesis parameter
normal   1.0  0.0  0.0  1.0  0.0  0.0  1.0  0.0  0.0  1.0  0.0  0.55 1.0
angry    0.0  1.0  0.0  0.0  1.0  0.0  0.0  1.0  0.0  1.1 -0.5  0.55 1.1
bashful  0.0  0.0  1.0  0.0  0.0  1.0  0.0  0.0  1.0  1.0  0.5  0.55 0.9
fast     1.0  0.0  0.0  1.0  0.0  0.0  1.0  0.0  0.0  2.0  1.0  0.55 1.0
slow     1.0  0.0  0.0  1.0  0.0  0.0  1.0  0.0  0.0  0.5  1.0  0.55 1.0
high     1.0  0.0  0.0  1.0  0.0  0.0  1.0  0.0  0.0  1.0  4.0  0.55 1.0
low      1.0  0.0  0.0  1.0  0.0  0.0  1.0  0.0  0.0  1.0 -2.0  0.55 1.0
```

最初の部分で使用するボイスを列挙し，次に発話スタイルを 1 行につき 1 つずつ定義します．それぞれ，各ボイスが持つ声質・音の高さ・話速の補間重み（0.0〜1.0）をそれぞれボイスごとに指定したあと，全体の話速（0.1〜10.0，標準 1.0）・音高（−12.0〜12.0，標準 0.0）・ジェンダー（−0.9〜0.9，標準 0.55）・音量（0.1〜10.0，標準 1.0）の 4 つの調整パラメータを指定します．

3　モーション

3.1　物理演算に基づく動作および MMD との互換性

MMDAgent では自然な動きをシミュレートするために物理演算ライブラリを用いています．MikuMikuDance と同じライブラリを使用しており，物理演算用の定義を含むモデルが MMDAgent でも利用可能です．MMD との主要な違いは以下のとおりです．

- 重力係数の倍率が異なります．MMDAgent のデフォルトは 2.0 で，MikuMikuDance に比べ浮いているように感じるかもしれません．10.0 とすると MikuMikuDance に近い挙動になります．設定するには設定ファイル（.mdf）の中に「gravity_factor=2.0」のように値を指定します．
- 床（XZ 平面）には剛体が入っていません．

「Shift+W」キーで物理剛体の表示を切り替えられます．表示は物理剛体のボーン関連タイプによって色分けされています．

また,「P」キーで物理演算を一時停止・再開することができます．停止前後で各剛体の挙動は維持されます．

3.2　MOTION_ADD のオプション詳細

モーションを実行する MOTION_ADD コマンドでは，第 4 引数でモーションを定義ボーン全てに適用するか一部に適用するか変えることができます．この機能はモーションで一部のボーンのみ動かしたいときに使います．

FULL オプションを指定した場合，モーションファイル (.vmd) 内で定義されているボーン・表情のキーフレーム全てが適用されます．PART オプションを指定した場合，0 フレーム目にあるキーフレームがスキップされます．記事「Motion superimposition」で紹介した，モーションファイル

(.vmd) を「Shift」キーを押しながら D&D したときの動作は，このオプションと同じ動作になっています．

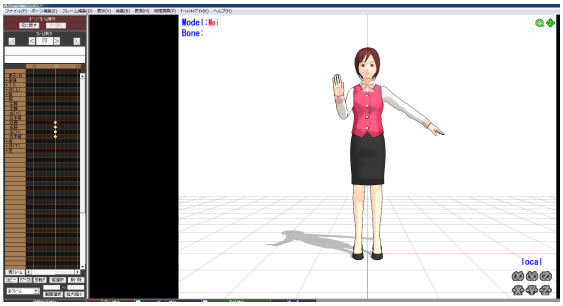
また，第 5 引数はモーションをループ再生するかどうかを指定できます．値は ONCE あるいは LOOP を指定します．

ONCE オプションを指定した場合，モーションが最終フレームまで再生された後，自動的に終了します．終了時には MOTION_EVENT_DELETE イベントが発行されます．LOOP オプションを指定した場合，モーションが最終フレームまで再生され，開始フレームへ戻り，ループ再生されます．モーションを終了させるには MOTION_DELETE コマンドを使います．

3.3 モーションを重ね合わせる

MMDAgent では，複数のモーションを任意のタイミングで重ねあわせて再生できます．これにより，まばたきや腕の動きなどの部分的なモーションをイベントに対応付けて実行することができます．

重ねあわせ再生を試すには,「Shift」キーを押しながらモーションファイル (.vmd) を D&D するか，あるいは MOTION_ADD コマンド の PART オプションを用います．例えば，右手を挙げるモーションを重ねたい場合，まず以下の図のように右手のみを挙げるモーションを作成します．

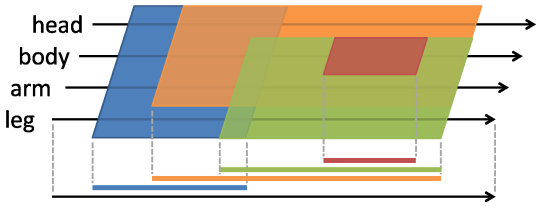


次に MMDAgent を起動してキャラクターを待機させておき,「Shift」 キーを押しながら先ほど作成したモーションファイル (.vmd) をキャラクターへ D&D すると，右手以外は待機のモーションを保ったまま右手を挙げるモーションが重ねて実行されます．この重ねあわせ再生では 0 フレーム目が無視されるため，0 フレーム目以外にキーフレームがあるボーン（この場合は右手）が実行されます．

スクリプト・メッセージで重ね合わせを行うには,，あるモーションを実行中のモデルに MOTION_ADD コマンドで PART オプションをつけて他のモーション（右手を上げる等）を実行させます．

モーションを重ねあわせたときの制御のイメージは以下のようになります．下図のように「head」「body」「arm」「leg」ボーンを持つモデルを想定します．図のように青，オレンジ，緑，赤色のモーションを実行した場合を表します．また，横軸は時間です．このように，ボーン集合ごとのモーションを重ね合わせながら実行できます．また，重ね合わせの開始・終了はスムージングが行われます．

ボーンが重なった場合の優先順位については，デフォルトでは後で実行されたほうが優先されます．例えば「body」ボーンにおいては，青→オレンジ→緑→赤→緑の順で各モーションで指示された動きが実行されます．この優先度は MOTION_ADD コマンドの PRIORITY オプションで個別に設定することもできます．



「B」キーを押すと，実行中の全モーションのエイリアス名とそれぞれが制御しているボーンの略図を表示させることができます．

4　モデル

4.1　オブジェクトを他のオブジェクト・エージェントに載せる

オブジェクトを他の表示中のオブジェクトの任意のボーン上に載せることが可能です．これを行うには，載せる対象のモデルを MODEL_ADD コマンドの 7 番目のオプションで指定します．

たとえば，モデルエイリアス名 mei のモデルに回転メニューを追従して表示させる場合は，以下のように記述します．これは，MMDAgent 本体と同時に配布されている「Sample　Script」でも使われています．

```
0 1 <eps> MODEL_ADD|menu|Accessory\menu\menu.pmd|0.0,-4.5,0.0|0.0,0.0,0.0|ON|mei
```

また，ボーン名を追加で記述することで特定のボーン上に載せることもできます．以下の例では，mei の右手に回転メニューを載せています．

```
0 1 <eps> MODEL_ADD|menu|Accessory\menu\menu.pmd|7,-1.5,0|0,90,90|ON|mei|右手首
```

5　カメラ

5.1　カメラと照明の MMD との互換性・相違点

MikuMikuDance とのカメラ及び照明に関する違いは以下の通りです.

- MMDAgent を起動した時のカメラ距離，視野角はそれぞれ 100, 16 です．MikuMikuDance ではそれぞれ 35, 45 です.
- カメラの Y 軸を中心とした回転の向きが MikuMikuDance と逆です．カメラモーションを読み込む際は，内部で自動的に変換します.
- 遠近法（パースペクティブ）は常に有効です.
- 照明のモーションはサポートしていません.

5.2 カメラ位置の設定および動作

カメラ視点を複数の方法で制御することができます.「D」キーを押すことで，カメラの焦点がどこにあるかをシーン内にオレンジの立方体として表示させることができます.

1. マウスを使って手動でカメラ視点を変更できます.
   - ドラッグで回転
   - 「Shift」キー＋ドラッグで平行移動
   - マウスホイールで焦点からの距離を変更（ズームイン・アウト）
   - 「Ctrl ＋ Shift」キー＋マウスホイールで画角の調整
2. 音声対話スクリプト (.fst) 上で CAMERA コマンドでカメラのパラメータを指定できます.「Viewing Log Information」で説明したログの左下の数字を以下のようにオプションとして与えます.

   `CAMERA|x 座標,y 座標,z 座標|回転の x 成分,y 成分,z 成分|距離|視野角|所要時間`
3. 最後の所要時間は変更にかける長さ（秒）です．0 で即座に変更します．-1 を与えるとでスムーズにパラメータが変化します．MikuMikuDance で作成したカメラモーションを CAMERA コマンドで与えることでカメラを動かすことができます.

   `CAMERA|カメラモーション.vmd`

6 変数

6.1 変数

音声対話スクリプト (.fst) では数値を格納する変数を扱うことができます．変数名のアルファベットの小文字と大文字は区別されます.

変数への代入は VALUE_SET コマンドで行います．宣言は必要ありません．新規名の場合は新規に定義され，すでにある名前の場合は上書きされます．代入終了時には VALUE_EVENT_SET イベントが発行されます．

変数の値の参照は VALUE_GET コマンドで行います．指定された変数名とその値を含む VALUE_EVENT_GET イベントが発行されます．

変数の値を固定値と比較するには VALUE_EVAL コマンドで行います．変数名，比較値および以下の比較演算子の 1 つを指定します．

EQ (EQual)	$=$
NE (Not Equal)	$\neq$
LE (Less or Equal)	$\leq$
LT (Less Than)	$<$
GE (Greater or Equal)	$\geq$
GT (Greater Than)	$>$

比較終了時には比較結果の TRUE または FALSE を含む VALUE_EVENT_EVAL イベントが発行されます．

変数を解放するには VALUE_UNSET コマンドを用います．変数解放時には VALUE_EVENT_UNSET イベントが発行されます．

6.2 ランダム変数

音声対話スクリプト (.fst) では変数に乱数を代入することができます．乱数の利用の際には，VALUE_SET コマンドで最大値と最小値を指定します．なお，乱数の値は値を代入した際に固定され，参照・評価には同じ値が使われます．以下にじゃんけんを行う例を示します．

```
1  11   RECOG_EVENT_STOP|じゃんけん     VALUE_SET|x|0|3
11 12   VALUE_EVENT_SET|x                VALUE_EVAL|x|LE|1
12 14   VALUE_EVENT_EVAL|x|LE|1|TRUE    SYNTH_START|mei|mei_voice_normal|グー
12 13   VALUE_EVENT_EVAL|x|LE|1|FALSE   VALUE_EVAL|x|LE|2
13 14   VALUE_EVENT_EVAL|x|LE|2|TRUE    SYNTH_START|mei|mei_voice_normal|チョキ

13 14   VALUE_EVENT_EVAL|x|LE|2|FALSE   SYNTH_START|mei|mei_voice_normal|パー
14  2   SYNTH_EVENT_STOP|mei             <eps>
```

この例では，まず変数 x に 0 以上 3 以下の乱数を代入しています．そして，VALUE_EVAL コマンドにより変数に代入された値の評価を行うことで，x の値によってじゃんけんの手を出力します．

6.3 タイマー変数

音声対話スクリプト (.fst) では，通常の変数の他に，時間経過でカウントダウンされる「タイマー変数」を利用できます．TIMER_START コマンドによりタイマー変数に値をセットすると，その代入した秒数から 0 へ向かって 0.1 秒単位でカウントダウンが開始されます．値が 0 に達する

と TIMER_EVENT_STOP イベントが発行されます．以下は 3 分待ってから「三分経ちました.」と喋る例です．

```
1  11  RECOG_EVENT_STOP|三分  TIMER_START|timer|180.0
11 12  TIMER_EVENT_STOP|timer  SYNTH_START|mei|mei_voice_normal|三分経ちました.
12  2  SYNTH_EVENT_STOP|mei  <eps>
```

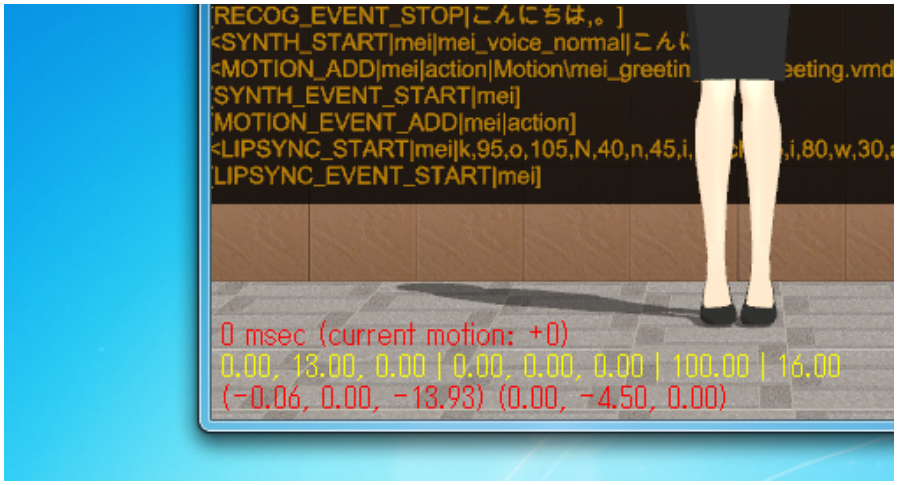
7　システム

7.1　ログに表示される情報

MMDAgent では「D」キーを押すことで各種ログを表示できます．

まず，背景に表示されるのはメッセージログです．下図のように，<>で囲まれたメッセージは各モジュールへの命令となるメッセージで，[] で囲まれたメッセージは各モジュールから出力されたメッセージです．

画面左下に表示される数値はシステムの状態を表しています．1 行目はモーションの時間ずれの補正量（ミリ秒）です．2 行目はカメラの視点パラメータで「x 座標, y 座標, z 座標 — 回転の x 成分, y 成分, z 成分 — 距離 —視野角」となっています．3 行目は表示されているモデルごとの原点座標の位置です．

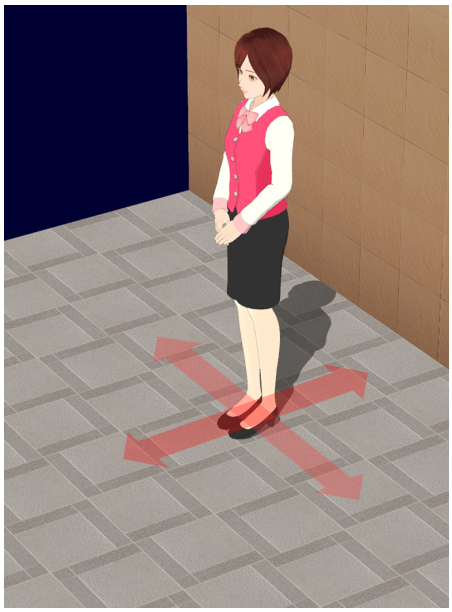


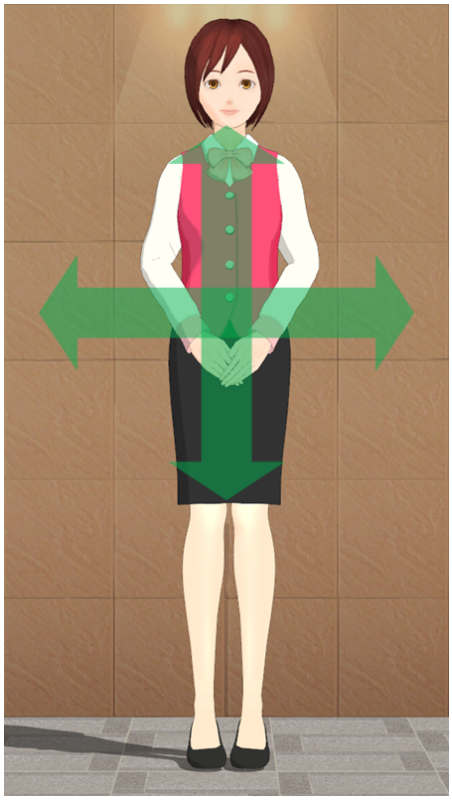
7.2　マウスでエージェント・オブジェクトを動かす

「Ctrl」キーを押しながらモデルをドラッグすることで，前後左右にモデルを移動できます．また,「Ctrl」＋「Shift」キーを押しながらドラッグすることで上下左右に移動できます．

7.3　シーン全体を停止する（HOLD 機能）

「H」キーを押すと，MMDAgent の全てのモーション実行を一時停止することができます．キーを押した瞬間に全てのモーションが停止し，その中で視点を自由に変更することができるので，対





話シナリオのある瞬間におけるシーン全体を確認することができます.


<<HOLD>>

プラグインの作成法

プラグインの作成と実装

- プラグインの新規作成準備
- プラグインで使用可能な関数
- プラグインのひな形
- プラグインの簡単な実装例
- その他作成上の注意点

プラグインの新規作成準備
開発環境

- 開発環境はVisual Studio
- Visual Studio 2013 community が無料で利用できます
  - 継続利用するためにはユーザ登録が必要
  - 日本語の言語パックで日本語化できます
  - http://www.visualstudio.com/ からダウンロードできます

プラグインの新規作成準備
ソリューションのダウンロード

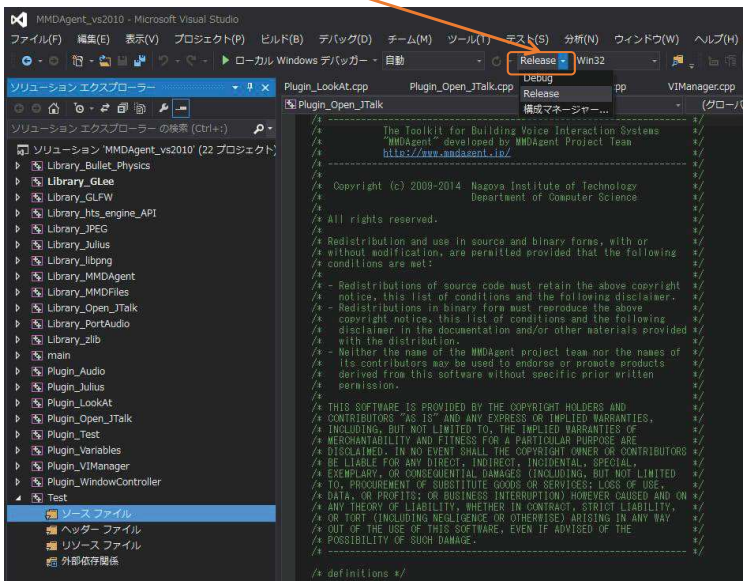
- MMDAgentの公式サイトからMMDAgentのソースコードをダウンロード
  - http://www.mmdagent.jp/
- 適当なフォルダに展開する
  - C:¥Users¥ユーザ名¥Documents¥Visual Studio 2013¥Projects 等に展開する

プラグインの新規作成準備
ソリューションのインポート

- 左上のファイルをクリック → “開く”
  → “プロジェクト/ソリューション” を選択
- ダウンロードし，展開されたMMDAgentの
  ソース中の“MMDAgent_vs2010.sln”を選択
- これだけでインポート完了

プラグインの新規作成準備
ソリューション構成の変更

- ソリューション構成を”Release”に変更

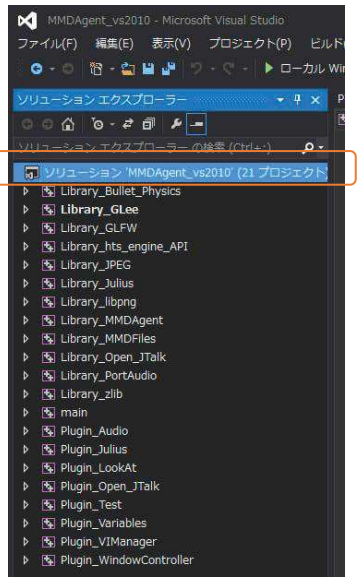


プラグインの新規作成準備
ビルド

- ためしにビルドしてみましょう
- ソリューション構成が”Release”になっていることを確認し，上のメニューバーの“ビルド”から“ソリューションのビルド”を選択
- もしくはF7を押す
- ソリューションフォルダ（先ほど展開させたフォルダ）中のReleaseフォルダにMMDAgent.exeが生成される
- ビルド済みMMDAgentと同じように使用できる

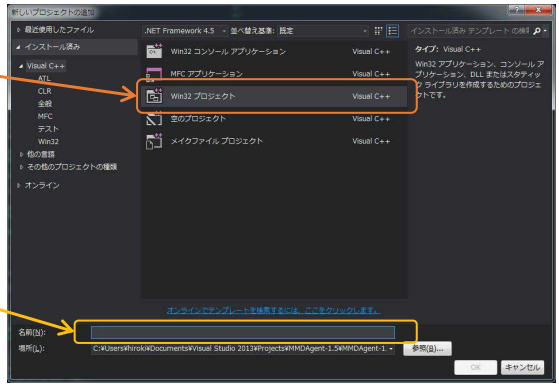
プラグインの新規作成準備
新規プロジェクトの追加

- MMDAgentのソリューションを右クリック
- “追加”をクリック
- “新しいプロジェクト”をクリック
- 新しいプロジェクトの追加ウィンドウが表示される



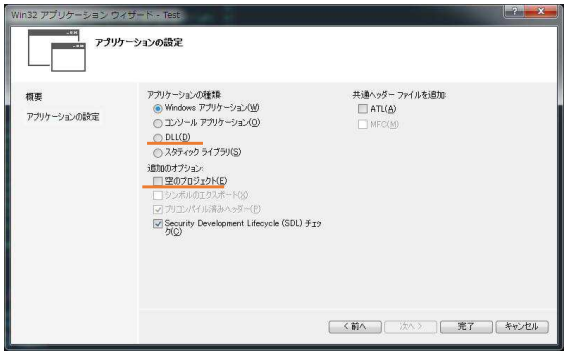
プラグインの新規作成準備
新規プロジェクトの追加

- Win32プロジェクトを選択する
- プロジェクトの名前を入力する
- “OK”をクリック
- Win32アプリケーションウィザードのウィンドウが開かれる



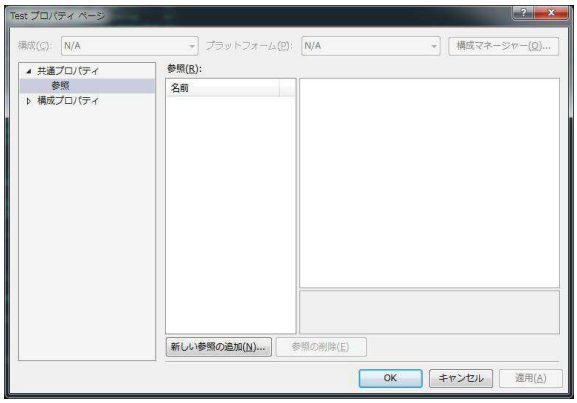
プラグインの新規作成準備
新規プロジェクトの追加

- “次へ”をクリック
- “アプリケーションの種類”の“DLL”をチェック
- “追加のオプション”の“空のプロジェクト”をチェック
- “完了”をクリック
- 新しいプロジェクトがMMDAgentのソリューションに追加される



プラグインの新規作成準備
プロジェクトの設定変更

- 新規作成したプロジェクトを右クリック
- “プロパティ”をクリック
- プロパティページが表示される

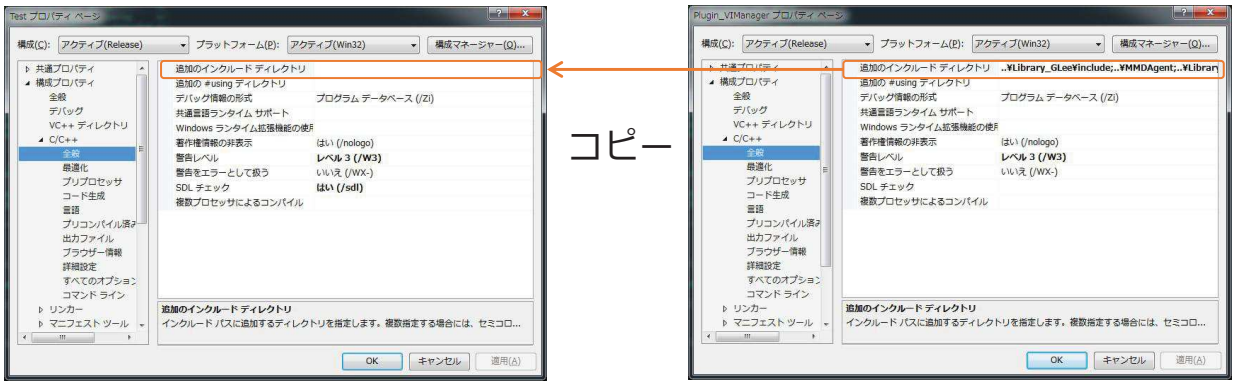


プラグインの新規作成準備
プロジェクトの設定変更

- 構成を“アクティブ(Release)”にする
- “構成プロパティ”を開く
- プロパティを変更する
  - “C/C++” → “全般”の
    “追加のインクルードディレクトリ”に記述を追加
  - “リンカー” → “全般”の
    “追加のライブラリディレクトリ”に記述を追加
  - “リンカー” → “入力”の
    “追加の依存ファイル”に記述を追加
  - プロジェクトの出力先ディレクトリを変更する
- 詳しくは次のページから説明する

プラグインの新規作成準備
プロジェクトの設定変更

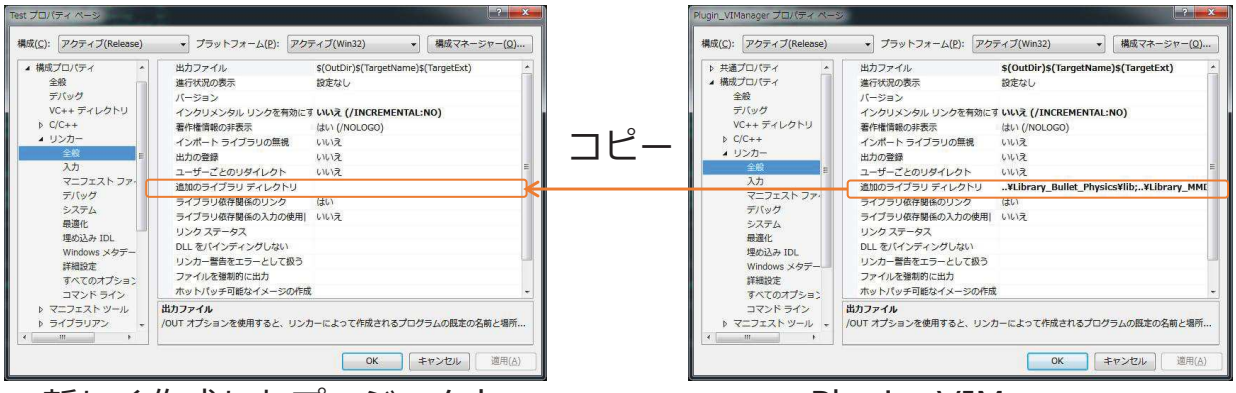
- “C/C++” → “全般”の
  “追加のインクルードディレクトリ”に
  記述を追加
  - Plugin_VIManagerの同じ部分からコピー&ペースト



新しく作成したプロジェクト　　Plugin_VIManager

プラグインの新規作成準備
プロジェクトの設定変更

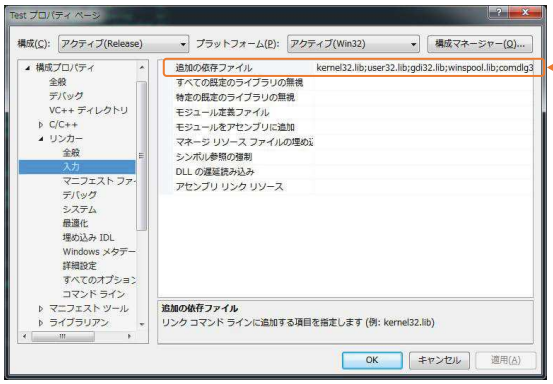
- “リンカー” → “全般”の
  “追加のライブラリディレクトリ”に
  記述を追加
  - Plugin_VIManagerの同じ部分からコピー&ペースト



新しく作成したプロジェクト　　Plugin_VIManager

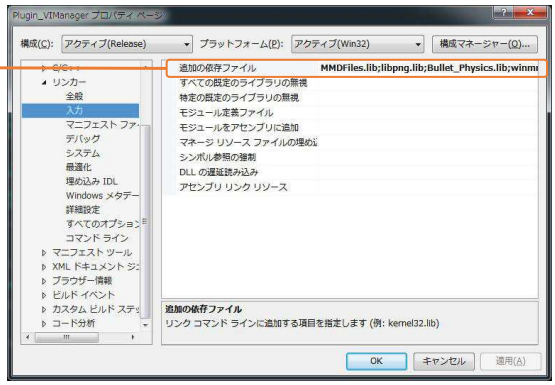
プラグインの新規作成準備
プロジェクトの設定変更

- “リンカー” → “入力”の
  “追加の依存ファイル”に記述を追加
  - Plugin_VIManagerの同じ部分からコピー&ペースト
  - 元々記述されていたものは消してしまってもよい



新しく作成したプロジェクト

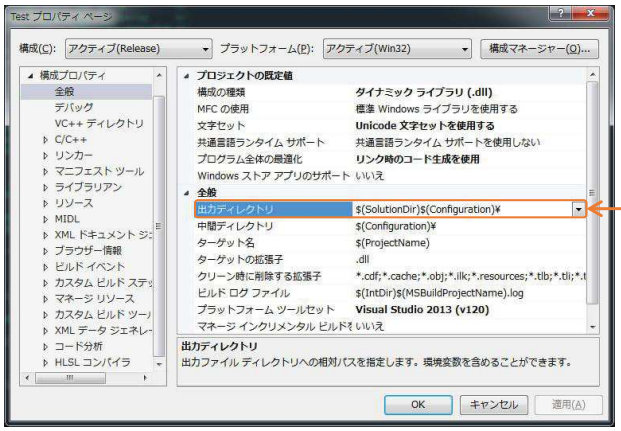
コピー



Plugin_VIManager

プラグインの新規作成準備
プロジェクトの設定変更

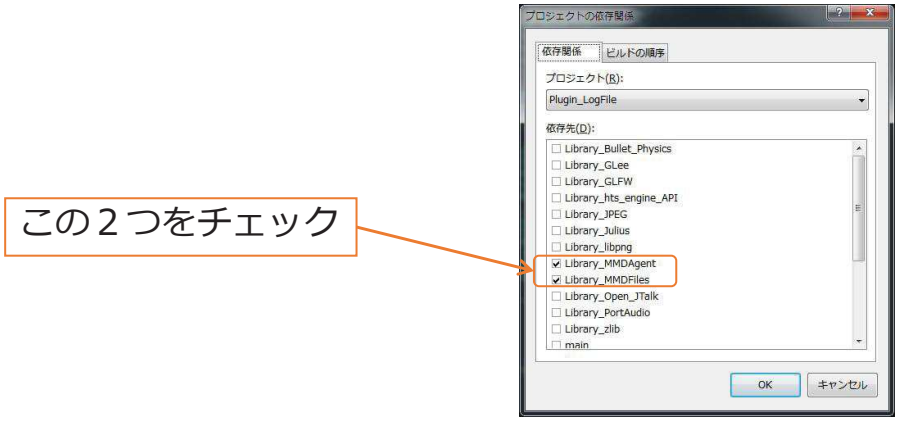
- プロジェクトの出力先ディレクトリを変更する
  - “構成プロパティ” → “全般” の
    “出力ディレクトリ”を変更する
  - $(SolutionDir)$(Configuration)¥Plugins¥と設定



この部分を変更する

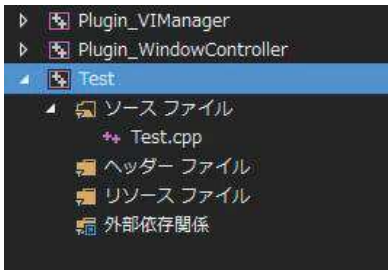
プラグインの新規作成準備
プロジェクトの依存関係の設定

- プロジェクトを右クリック →
  “ビルド依存関係” → “プロジェクト依存関係”
- “Library_MMDAgent”と
  “Library_MMDFiles”をチェック



プラグインの新規作成準備
ファイルの新規作成

- 作成したプロジェクトの“ソースファイル”
  もしくは“ヘッダーファイル”を右クリック
- “追加” → “新しい項目”を選び適時追加する
- プロジェクト内にファイルが作成されれば
  いいので、フォルダの使用はお好みで



デバッグモードで起動したい場合

- ソリューション構成をDebugにして
  同様に設定する
- DebugフォルダにReleaseフォルダから
  AppDataと.mdfファイルをコピーする
  - ダウンロードしてきたソリューションにはDebug
    フォルダがないので自分で作成するか、
    一度ビルドを実行すれば自動で生成してくれる

プラグインで使用可能な関数 extAppStart

- MMDAgent起動時に 1 回だけ呼ばれる
- プラグインの初期化処理を記述する
- 引数
  - mmdagent：MMDAgentの機能を利用できる
    詳しくは後述

```
/* extAppStart: load models and start thread */
EXPORT void extAppStart(MMDAgent *mmdagent)
{
/* プラグインの初期化処理はここに書く */
}
```

プラグインで使用可能な関数 extAppEnd

- MMDAgentが終了する時（ウィンドウが閉じられる時）に１回だけ呼ばれる
- プラグインの終了処理を記述する
- 引数
  - mmdagent：MMDAgentの機能を利用できる
    詳しくは後述

```
/* extAppEnd: stop and free thread */
EXPORT void extAppEnd(MMDAgent *mmdagent)
{
/* プラグインの終了処理を書く */
}
```

プラグインで使用可能な関数 extProcMessage

- MMDAgentの内部メッセージ (EventMessageとCommandMessage) が発行された時に呼び出される
- 内部メッセージによって処理を変える
- 引数
  - mmdagent：MMDAgentの機能を利用できる
    詳しくは後述
  - type：内部メッセージの種類 (MMDAgent.hを参照)
  - args：内部メッセージの内容

```
/* extProcMessage: process message */
EXPORT void extProcMessage(MMDAgent *mmdagent,
                           const char *type,
                           const char *args)
{
/* メッセージの種類と内容によって処理を変える */
}
```

プラグインで使用可能な関数
引数 mmdagent

- MMDAgentの様々なシステム機能を利用可能
- MMDAgent.h (Library_MMDAgent内)
  を参照する

例：sendMessage　内部メッセージを発行させる関数

```
/* sendMessage: send message
    to global message queue */
void sendMessage(const char *type,
                 const char *format, ...);
```

```
mmdagent->sendMessage(MMDAGENT_EVENT_PLUGINENABLE,
                      "%s", "MyPlugin");
```

プラグインで使用可能な関数
その他

- extUpdate
  - 処理の毎フレームに呼ばれる
  - モーションを変化させたい時などに使用する
- extRender
  - 処理の毎フレームに呼ばれる
  - 画面上に新たにグラフィック描写を追加する時
    などに使用する

プラグインのひな形

- プラグインのひな形となるソースコードを添付資料に記載します
  - Plugin_Template.pdf
  - Plugin_Template.cpp

プラグインの実装例

- ログを作成するプラグイン
  - extAppStartでファイルストリームを開く時刻を取得し，ファイルに書き込む
  - extProcMessageで内部メッセージを取得，ファイルに内部メッセージを書き込む
  - extAppEndでファイルストリームを閉じる
  - 起動に用いた.mdfファイルと同じ階層にMessageLog.txtが生成されます
- ソースコードを添付資料に記載します
  - Plugin_LogMessage.pdf
  - Plugin_LogMessage.cpp

プラグインのビルドと実行

- ソリューションのビルドを実行すればよい
  - ソリューション構成がReleaseになっていれば，MMDAgentのソリューションフォルダ内のReleaseフォルダに実行ファイルが生成される
  - Releaseフォルダ内のPluginsフォルダにプラグイン(.dll)が生成される
- 公式サイトからダウンロードしたMMDAgent_Example内の.mdfファイルを生成されたMMDAgent.exeにドラッグ＆ドロップ
- 生成されたプラグイン(.dll)をコピーして利用も可能
  - MMDAgent.exe直下のPluginsフォルダ内のプラグイン(.dll)は自動的に全て読み込まれる

作成上の注意点

- 過去のバージョンのプラグインの書き方は使えない
- c++標準ライブラリの<ctime>のlocaltime関数を使おうとすると実行時にエラーとなる

```
/* definitions */
#ifdef _WIN32
#define EXPORT extern "C" __declspec(dllexport)
#else
#define EXPORT extern "C"
#endif /* _WIN32 */

#define LOGFILENAME  "MessageLog.txt"  /* ログファイルのファイル名 */
#define PLUGINLOGMESSAGE_NAME  "LogMessage"   /* プラグインの名前 */
/* ログファイル関係のメッセージタイプ
（自由に内部メッセージタイプを設定できる） */
#define MMDAGENT_EVENT_FILEOPEN  "LOGMESSAGE_EVENT_FILEOPEN"
#define MMDAGENT_EVENT_FILECLOSE "LOGMESSAGE_EVEMT_FILECLOSE"

/* headers */
#include "MMDAgent.h"
#include <fstream>
#include <ctime>

/* variables */
static bool enable;
static std::ofstream ofs;
static time_t t;
static tm *x;

/* extAppStart: load models and start thread */
EXPORT void extAppStart(MMDAgent *mmdagent)
{
        enable = true;
       mmdagent->sendMessage(MMDAGENT_EVENT_PLUGINENABLE, "%s",
       PLUGINLOGMESSAGE_NAME);

        /* ログ作成に使用するファイル名 */
        const char *fileName = LOGFILENAME;

        /* ログファイルを開く（追記） */
        ofs.open(fileName, std::ios::out | std::ios::app);
```

```
	if (!ofs) {  /* もしファイルを開くのに失敗していたら */
		/* ファイルを開くのに失敗したことメッセージを発行する */
		mmdagent->sendMessage(MMDAGENT_EVENT_FILEOPEN,
		"%s can not be opened!", fileName);
	}
	else {  /* ファイルを開くのに成功していたら */
		/*ファイルを開くのに成功したことメッセージを発行する*/
		mmdagent->sendMessage(MMDAGENT_EVENT_FILEOPEN,
		"%s can be opened", fileName);

		/* 現在時刻を取得 */
		t = time(0);
		char buf[32];
		ctime_s(buf, sizeof(buf), &t);
		/* 時刻を書き込む */
		ofs << buf;
		ofs << "[[Start]]" << std::endl;
	}
}

/* extProcMessage: process message */
EXPORT void extProcMessage(MMDAgent *mmdagent, const char *type, const char *args)
{
	if (enable == true) {
		/* 出力ストリームに出力する（１行） */
		/* 以下のコメントアウトを外すと特定のメッセージタイプ（音声入力する）の
時のみログファイルに書き込むようになる */
//		if (MMDAgent_strequal(type, "RECOG_EVENT_STOP"))
		{
			ofs << type << "|" << args << std::endl;
		}
	}
}




/* extAppEnd: stop and free thread */
```

```
EXPORT void extAppEnd(MMDAgent *mmdagent)
{
	/* MMDAgent のログの区切りを示す */
	ofs << "[[End]]" << std::endl;
	ofs << std::endl;

	/* MMDAgent 終了時にログファイルを閉じる */
	ofs.close();
	mmdagent->sendMessage(MMDAGENT_EVENT_FILECLOSE,
	"%s was closed", LOGFILENAME);
}

/* execUpdate: run when motion is updated */
EXPORT void extUpdate(MMDAgent *mmdagent, double deltaFrame) {

}

/* execRender: run when scene is rendered */
EXPORT void extRender(MMDAgent *mmdagent) {

}
```

```
#ifdef _WIN32
#define EXPORT extern "C" __declspec(dllexport)
#else
#define EXPORT extern "C"
#endif /* _WIN32 */

/* definitions */

/* headers */
#include "MMDAgent.h"

/* variables */

/* extAppStart: load models and start thread */
EXPORT void extAppStart(MMDAgent *mmdagent) {

}

/* extProcMessage: process message */
EXPORT void extProcMessage(MMDAgent *mmdagent, const char *type,
                           const char *args) {

}

/* extAppEnd: stop and free thread */
EXPORT void extAppEnd(MMDAgent *mmdagent) {

}

/* execUpdate: run when motion is updated */
EXPORT void extUpdate(MMDAgent *mmdagent, double deltaFrame) {

}

/* execRender: run when scene is rendered */
EXPORT void extRender(MMDAgent *mmdagent) {

}
```

Android 版 の利用方法

はじめに

- 教科書に参考資料がある
- ソースコードは参考資料に載っている
- USBメモリのMMDAgent_Androidフォルダにソースコードの一部を添付
- このスライドでは簡単な説明にとどめる

Android 版を利用するまでの手順

1. Android アプリの開発準備
2. MMDAgent のソースコードのダウンロード
3. Android プロジェクトの作成
4. ソースコードのインポート
5. ソースコードの編集
6. ビルド

Android アプリ開発の準備（１）

- Android 端末を用意
- その端末でアプリ開発を行う準備
  - 開発者オプションの設定項目を出す
  - 開発用パソコンに，用意した端末の
    ドライバをインストール

詳しい方法はインターネットを参考に

Android アプリ開発の準備（2）

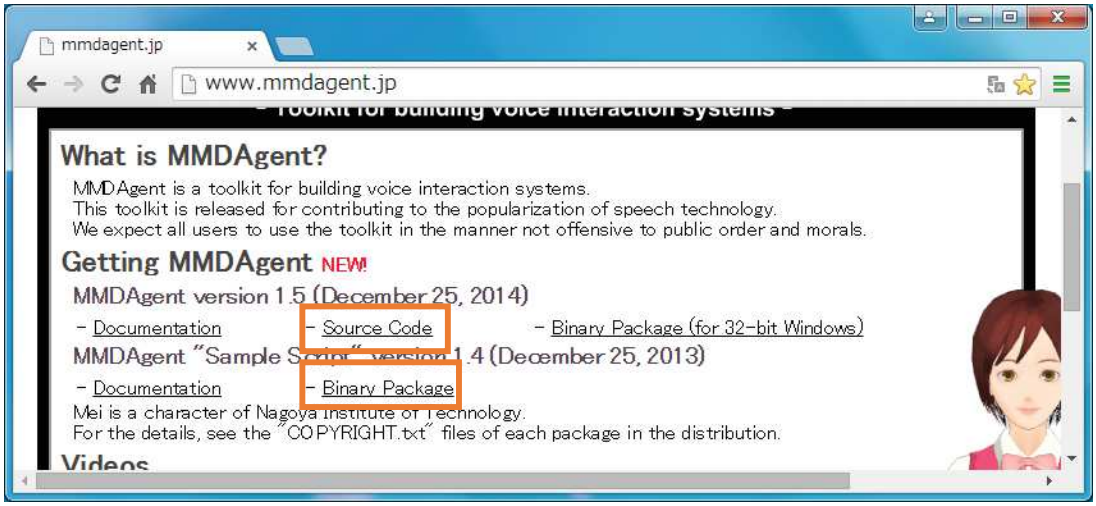
- Android Studio での開発環境を整える
- 必要なソフトウェア
  - JDK：Java の開発キット
  - Android Studio：統合開発環境
  - Android NDK：C/C++ による開発ツールキット

詳しいインストール方法は公式サイトを参考に

本スライドでは Android Studio 1.0.2 を基に説明

MMDAgent のダウンロード

- www.mmdagent.jp からダウンロード
  - “MMDAgent” の “Source Code”
  - “Sample Script” の “Binary Package”



プロジェクトの作成（１）

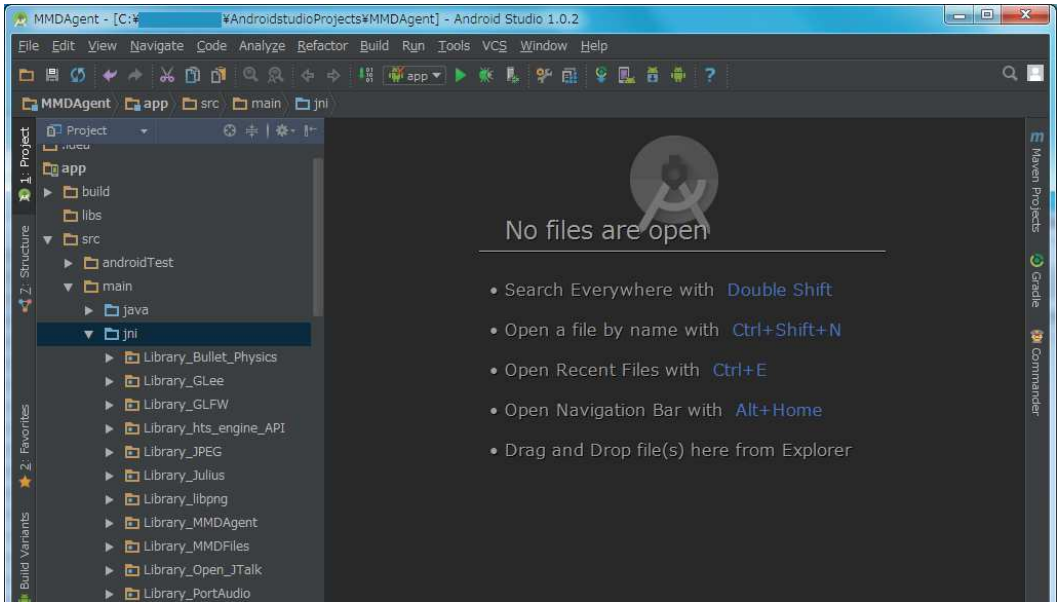
- 新規プロジェクトを作成
- アプリ名はご自由に
- パッケージ名もご自由に
  - パッケージ名はアプリの識別IDである
  - なるべく既存アプリと被らないように
  - パッケージ名を覚えておく

プロジェクトの作成（２）

- アプリの動作対象の設定
  - 使用する Android 端末に合わせる
  - 基本は “Phone and Tablet“ 最低 SDK “Android 4.0”
- アクティビティーは作成しない
  - アプリの画面みたいなもの
  - MMDAgent のソースコードに含まれている

ソースコードのインポート

- MMDAgent のソースコードを
  “app/main/jni/” フォルダ以下にコピー



マニフェストファイルの編集

“app/src/main/AndroidManifest.xml” を編集する

1. パーミッションの追加
   - 外部ストレージの読み込み権限
   - 音声録音権限
2. NativeActivity の追加

local.properties の編集

- NDK のパスを記述する
  - “ndk.dir=<Android NDK の保存先>”
  - Windows の場合，フォルダ区切り文字 “¥” を “¥¥” と入力すること
    - “¥” はエスケープ文字であるため

build.gredle の編集

- app/build.gredle を編集
  - 同名のファイルが他フォルダにもあるため注意
- MMDAgent のソースコード中の Makefile と同じようなことを記述する
  - Config.h を生成
  - ソースコードを NDK でコンパイル

MMDAgent のスクリプトの設置

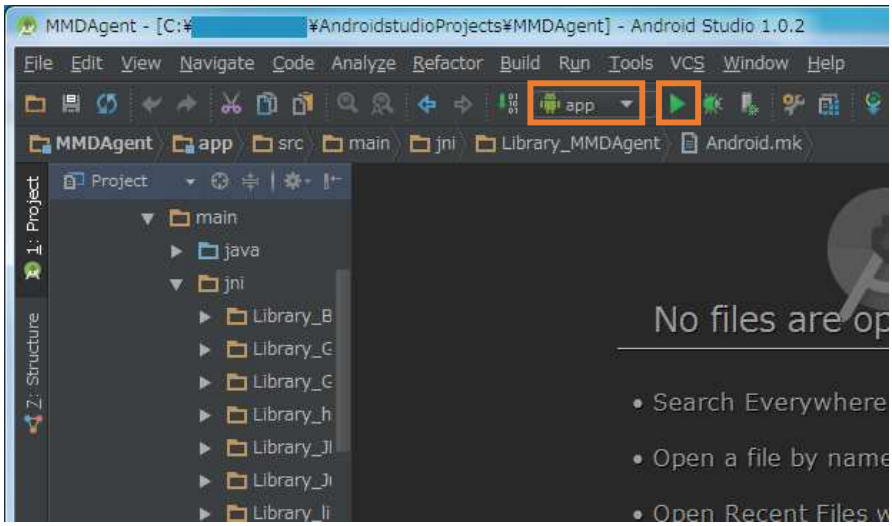
- Android のストレージに MMDAgent のスクリプトをコピー
- ソースコードの中の App ディレクトリもコピー
- パスを覚えておく

Android.mk の編集

- app/src/main/jni/Library_MMDAgent/Android.mk を編集
  - Android.mk というファイルは他のフォルダにもあるため注意
- -DMMDAGENT…=“…” となっている部分を編集
  - Android にコピーしたスクリプトのパス
  - パッケージ名

プロジェクトのビルドおよび実行

- 実行ボタンを押すだけ
  - app になってることを確認



動かないときは？（１）

- ビルドできない
  - NDKのパスはあってますか？
  - build.gradleは正しいですか？
  - Android.mk は正しいですか？
- アプリが起動できない
  - マニフェストファイルは正しいですか？
  - Android.mkで正しいパッケージ名を指定しましたか？

動かないときは？（２）

- 画面が黒いまま（Open GL さえ動いていない）
  - Android.mkで正しいパッケージ名を指定しましたか？
- 画面が青いまま（Open GL は動いている）
  - Android.mkで（略）
  - Android に AppData フォルダをコピーしましたか？
  - FST は正しいですか？

動かないときは？（３）

- 音声を認識しない（音声バーが表示されない）
  - Android にマイクはありますか？
  - 他のアプリがマイクを使用中ではありませんか？
  - Android に AppData フォルダをコピーしましたか？
- メイちゃんが反応しない
  - FST は正しいですか？
  - マイクのレベル設定はちょうどいいですか？

補足

- Eclipse でも開発可能
  - マニフェストの編集，Android.mk の編集は共通
  - 他の部分は Eclipse に合わせること
- JNI を用いて Java との連携も可能

MMDAgent を Android で利用するための手順

1. はじめに

この資料は，人工知能処理学会主催の「第 7 回 AI ツール入門講座」の MMDAgent 講習会に参加する方々へ配布する教科書の，参考資料として作られたものである．この資料では MMDAgent 1.5 を Android 端末で利用するための手順を説明する．想定する開発環境は Windows 7 である．他の OS を利用する場合は各 OS に合わせて読みかえること．（例えばフォルダのセパレータ記号）

2. Android アプリ開発環境の準備

まず，Android 端末の用意をする．

次に，以下のソフトウェアをインストールする．

- Java Development Kit (JDK) : Java の開発ツールキット
- Android Studio : Android アプリの統合開発環境
- Android SDK : Android アプリの開発キット（Android Studio に同梱されている）
- Android NDK : C/C++による Android アプリ開発ツールキット

各ツールのインストール方法は以降で説明するが，Web で検索したものも参考にすると詳しく最新の情報が得られるため良いだろう．今回は統合開発環境として Android Studio を用いるが，Eclipse でも Android 版 MMDAgent を開発することができる．ただし今回の資料で説明する手順と大きく変わってしまうため注意すること．

2.1 Android 端末の用意

まず，Android 端末を用意する．Android のバージョンは 4.0 以上推奨．それ以下のバージョンでの動作は未確認である．エミュレータでも開発できないこともないが，MMDAgent のスクリプトの配置やマイクの設定，動作が遅いという点で苦労する．

次に，用意した Android 端末で開発者オプションを使えるようにする．端末によって設定方法が違う場合があり，各自で調べて欲しいが一応説明する．まず，Android で設定アプリを起動する．そうすると図 1 のような画面が開くはずだ．

「端末情報」や「タブレット情報」というような名前の項目があるはずなので，それタップして開くと図 2 のような画面が表示される．

図 1: 設定アプリのトップ画面

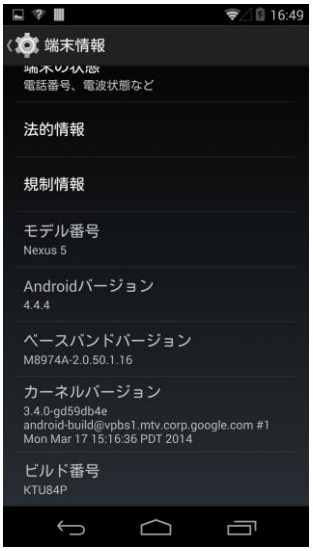


図 2: 端末情報の画面

この画面では「ビルド番号」という項目があるはずなので，その項目を連打する．

「あなたは開発者になりました」というようなポップアップ表示が出たら連打をやめる．途中で「あと 5 回で開発者になります」というような表示が出るかもしれない．

戻るボタンで設定アプリのトップ（図 1 の画面）に戻る．先ほどは表示されていなかった「開発者向けオプション」の項目が表示されたはずである．

「開発者向けオプション」の項目をタップして開くと図 4 のような画面が表示される．「USB デバッグ」という項目にチェックを入れたら完了である．

図 3：開発者向けオプションが表示された

図 4：開発者向けオプションの画面

これで Android 端末の用意は完了である．パソコンと Android 端末の接続のために，場合によっては開発用パソコンに用意した Android 端末のドライバをインストールする必要がある．

2.2 JDK のインストール

次に Java Development Kit (JDK) のインストール方法の説明をする．

まず JDK をダウンロードする．JDK のエディションは Java SE 7 である．資料作成時点での最新版は Java SE 8 であるが，Android Studio 1.0.2 で公式にサポートされているのは Java SE 7 である．Java SE 8 でも Android Studio の設定を編集すれば動作するらしい．JDK 7 以上での Java 開発環境が整っていればこの作業は必要ない．

JDK のダウンロードが済んだらウィザードに従ってインストールする．その際，JDK のインストール先を覚えておくこと．

インストールが済んだら環境変数「JAVA_HOME」を設定する．Mac や Linux 等の場合は「~/.bashrc」ファイルに記述すればよい．Windows の場合は，コントロールパネルを開き「システムとセキュリティ」>「システム」と選択すると図 5 のような画面が表示される．

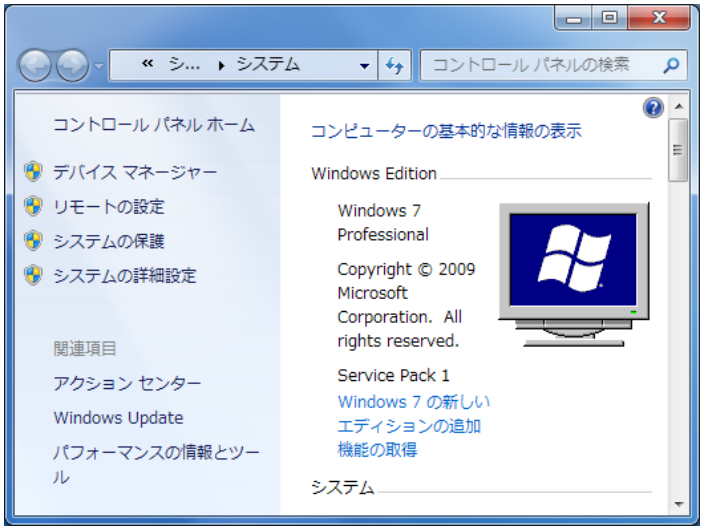


図 5：コントロールパネルの「システム」の画面

この画面で，システムの詳細設定をクリックすると次の図 6 のような新しいウィンドウが開く．

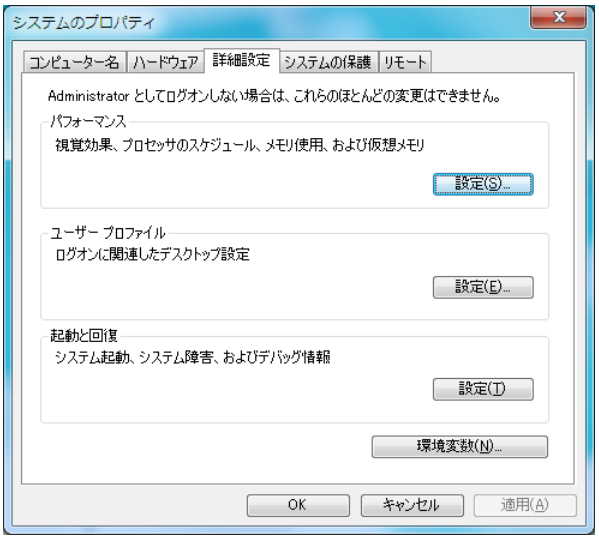


図 6：システムの詳細設定画面

この画面で「環境変数」ボタンをクリックすると次の図 7 のような画面が開く．

図 7：環境変数設定画面

システム環境変数に「JAVA_HOME」があるなら「編集」ボタンをクリックして「JAVA_HOME」の値を先ほど JDK をインストールしたフォルダに書き換える．「JAVA_HOME」がない場合は「新規」ボタンをクリックして，「変数名」に「JAVA_HOME」，「変数値」に JDK をインストールしたフォルダを書いて，「OK」ボタンを押す．

2.3 Android Studio のインストール

まず，Android Developers から Android Studio をダウンロードする．ダウンロード画面にインストール方法が説明されているはずである．それにしたがってインストールすること．インストール方法は OS によって違う．Windows ならダウンロードした .exe ファイルを開くと，セットアップウィザードが起動し，画面に従えばインストールできるはずだ．Mac ならダウンロードした .dmg ファイルを開き，Android Studio をアプリケーションフォルダにコピーする．その後 Android Studio を開くとセットアップウィザードが起動するので，画面に従えばインストールできるはずだ．

2.4 Android SDK のインストール

Android SDK は Android Studio に同梱されているため，ダウンロードやインストールをする必要はない．ただ，Eclipse などのほかの統合開発環境で開発したい場合などでは必要となる．Android SDK が同梱されていない Android Studio も入手できるため，

そちらをダウンロードした方も Android SDK を別途ダウンロード・インストールする必要がある．

2.5 Android NDK のインストール

Android NDK は C/C++ で Android アプリを開発できるツールキットである．Android NDK は Android Studio に同梱されていないため，ダウンロード・インストールが必要である．

まず，Android Developers から Android NDK をダウンロードする．ダウンロードされたファイルは自己解凍方式の zip ファイルである．任意の場所に解凍することでインストールする．後の設定で使うため，インストールした場所をメモしておくこと．

3. MMDAgent のダウンロード

“www.mmdagent.jp” から MMDAgent のソースコードとスクリプトをダウンロードする．（講習会で配布した USB メモリにもファイル一式が含まれている．）

ダウンロードしたファイルは好きなフォルダで解凍する．“MMDAgent-1.5”, “MMDAgent_Example-1.4” というフォルダが作成される．“MMDAgent-1.5” は MMDAgent のソースコードの一式である．“MMDAgent_Example-1.4” は MMDAgent の対話スクリプトや 3D モデルファイル，モーションファイル，設定ファイルが含まれている．

4. プロジェクトの作成

Android Studio を起動すると次の図 8 のようなメニューが表示される．

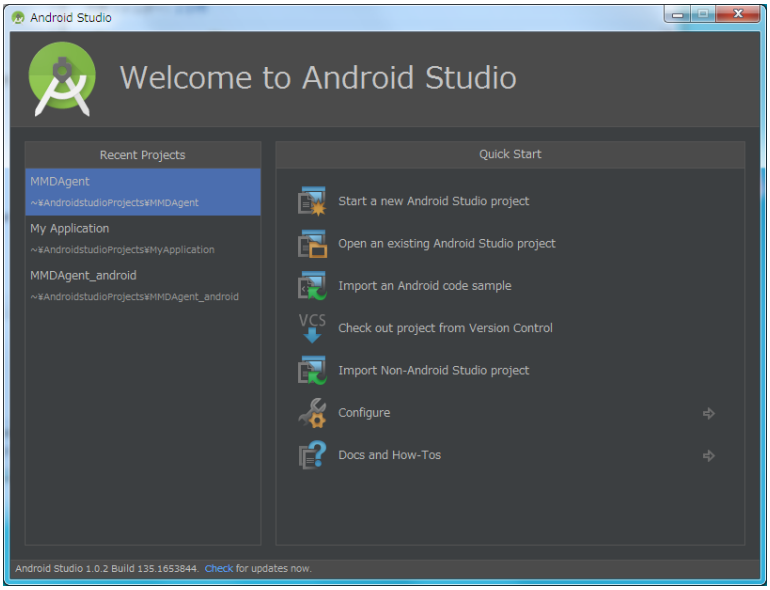


図 8 Android Studio の起動画面

この画面で「Start a new Android Studio project」を選択し，プロジェクトの新規作成をする．すると図 9 のような画面が表示される．

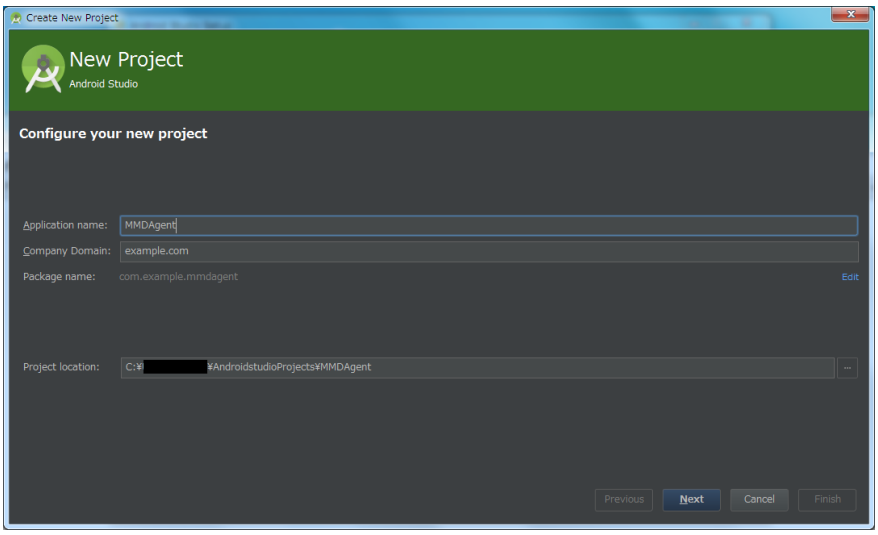


図 9：新規プロジェクトのアプリ名・パッケージ名の設定画面

ここでは，作成するアプリの基本情報を入力する．「Application name」の欄にアプリ名として好きな名前を入力する（例えば 「MMDAgent」）．次に「Company Domain」に使用するドメインを入力する（例えば「example.com」）．すると「Package name」の欄が自動的に入力される（例えば 「com.example.mmdagent」）．

Package name は Android アプリの識別 ID として扱われるため，なるべく既存のアプリと被らないようにすること．自動で入力されたものから変更したい場合は，右側の「Edit」を押すと手動で入力することができる．ソースコードを編集するときに使うため，パッケージ名は覚えておくこと．入力を終えたら「Next」ボタンで次へ進む．

「Next」ボタンを押すと図 10 のような画面が表示される．

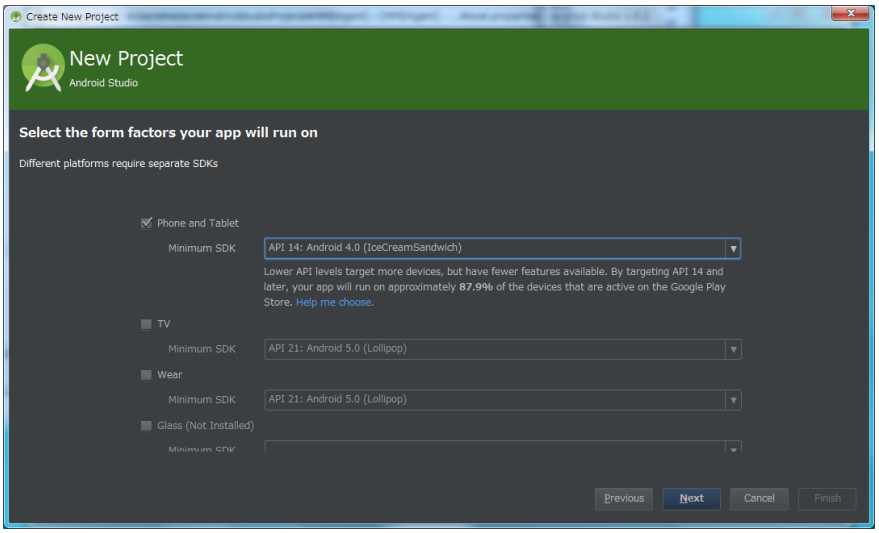


図 10：アプリの動作対象デバイスの設定画面

ここでは作成するアプリの対象デバイスについての設定を行う．まずチェックボックスでデバイスの種類を選択する．対象がスマートフォン・タブレットである場合は「Phone and Tablet」を選択する．

次に選択したデバイスの「Minimum SDK」の欄でこのアプリがサポートする Android OS の最低バージョンを指定する．基本的には「API 14」を選択すればよいが，使用する端末の Android OS のバージョンが低い場合は端末に合わせて API 14 より下げること．ただし API 14 以下での動作確認はしていないため，コンパイルができなかったりアプリが実行できなかったりする可能性があるので注意すること．ここまで入力を終えたら「Next」ボタンで次へ進む．

「Next」ボタンを押すと図 11 のような画面が表示される．

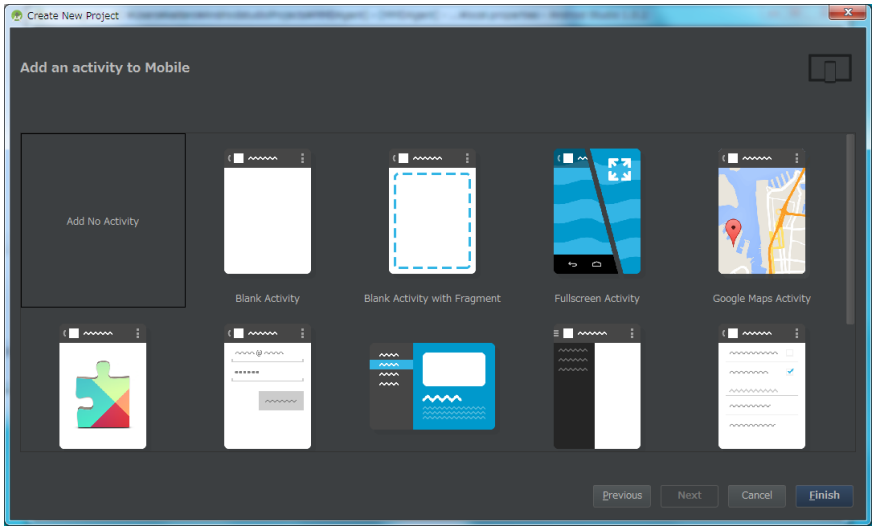


図 11：アクティビティ追加画面

この画面はアクティビティー（Android におけるアプリの画面に関するクラス）の作成についての設定画面である．ここではアクティビティーを追加しないので「Add No Activity」を選択して「Finish」ボタンを押す．しばらく待つとプロジェクトの作成が行われて次のような画面が表示される．これでプロジェクトの作成作業は終了である．

5. ソースコードのインポート

5.1 事前準備

ソースコードのインポートの前に少し準備をする．ここまでの作業を終えた段階では Android Studio は図 12 のような画面になっているはずである．

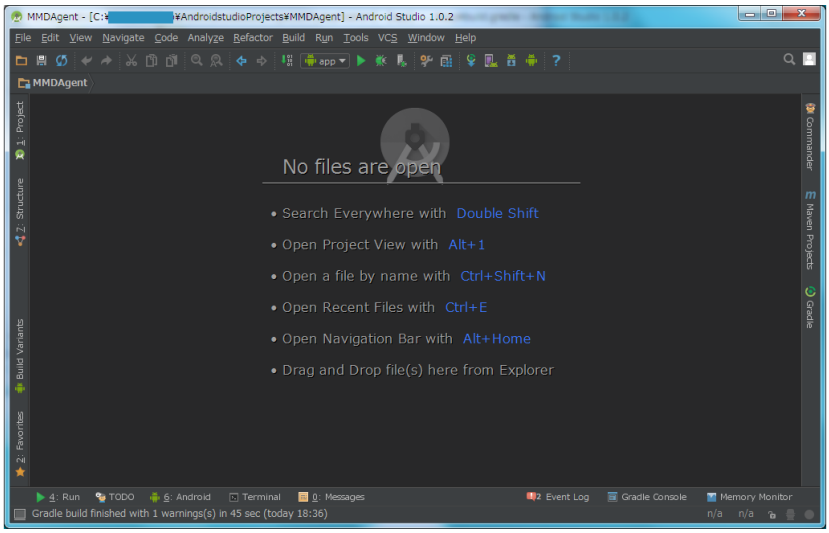


図 12 : Android Studio の画面

ここで左上の「Project」をクリックすると図 13 のような画面になる．

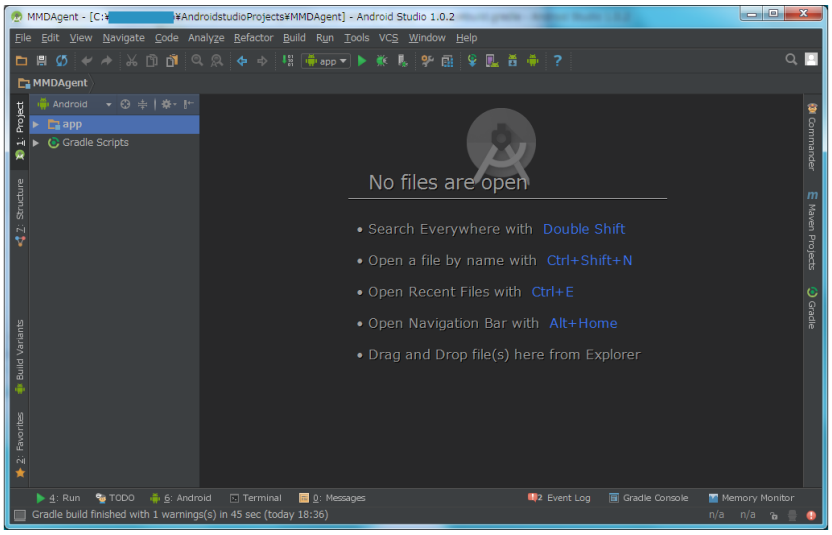


図 13 : Project View を開いた画面

新しく表示された画面はプロジェクトファイルを表示する画面である．左上に「Android」と表示されているはずである．この状態のファイル表示は実際のフォルダ構造とは違いソースコードの種類ごとに分けられた仮想的なフォルダ構造となっている．このままでは説明がしづらいので実際のフォルダ構造と同じ表示形式に変更する．

「Android」の部分をクリックすると図 14 のようなメニューが表示される．

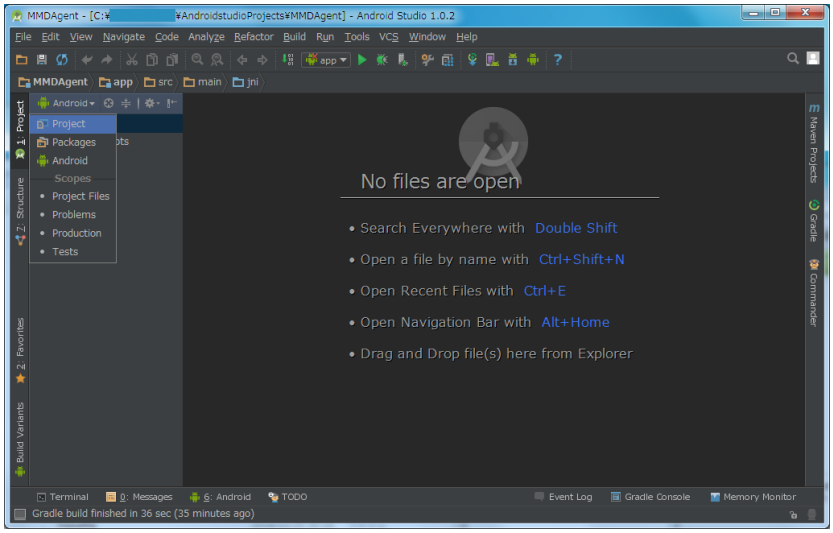


図 14 : Project View で「Android」をクリックした画面

ここで「Project」を選択するとフォルダの表示が実際のフォルダ構造と同じになる．

5.2 JNI フォルダの作成

フォルダ表示を「Project」にすると，アプリ名とほぼ同じ名前のフォルダ（例えば「MMDAgent」）と「External Libraries」という名前のフォルダが表示されているはずである．以降，アプリ名とほぼ同じ名前のフォルダの名前は「MMDAgent」であると仮定して説明する．ここで「MMDAgent」フォルダの三角ボタンをクリックするとフォルダが開く．その中の「app」フォルダを右クリックするとメニューが表示され，「New」>「Folder」と選択していくと図 15 のような画面が表示される．

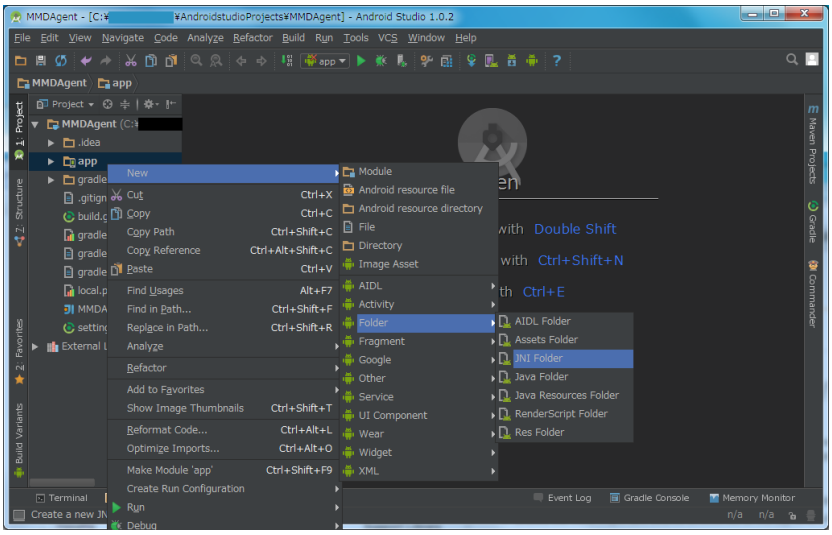


図 15 : JNI フォルダを追加するメニュー

ここで「Folder」メニューの中の「JNI Folder」をクリックすると，図 16 のような画面が表示される．

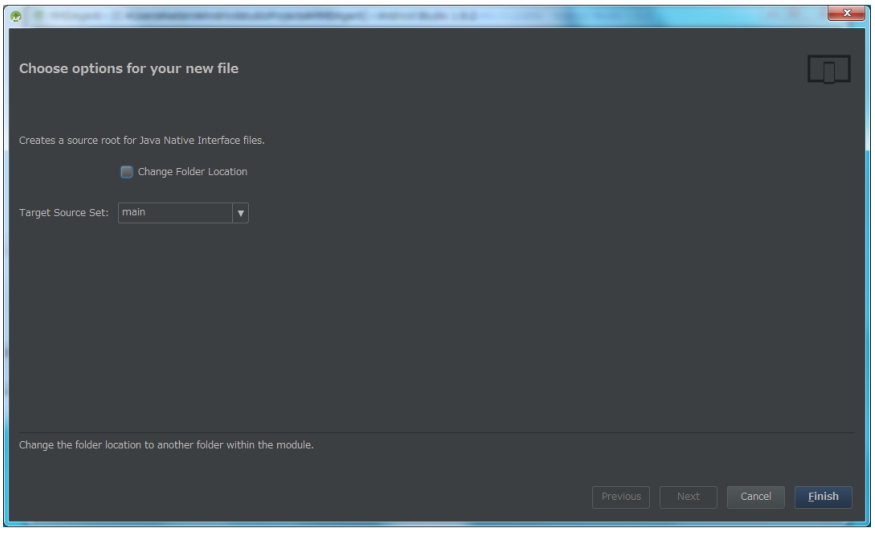


図 16 : JNI フォルダ作成画面

何も設定は変える必要はないため，そのまま「Finish」ボタンを押す．そうすると MMDAgent¥app¥src¥main¥ フォルダの中に jni フォルダが作成される．jni フォルダは C/C++ のソースコードを置くためのフォルダである．

5.3 ソースコードのインポート

次に jni フォルダの中に，あらかじめダウンロードしておいた MMDAgent のソースコード一式をコピーする．エクスプローラ（もしくは好きなファイラ）でファイルを選択してコピーし，Android Studio のプロジェクトビューで jni フォルダにペーストする．コピーさえできればよいため，これ以外でも好きな方法を用いて構わない．

MMDAgent-1.5 フォルダをそのまま jni フォルダの中に入れるのではなく，MMDAgent-1.5 フォルダの中身を jni フォルダの中に入れることに注意する．コピーした状態は図 17 の画面のようになっているはずである．

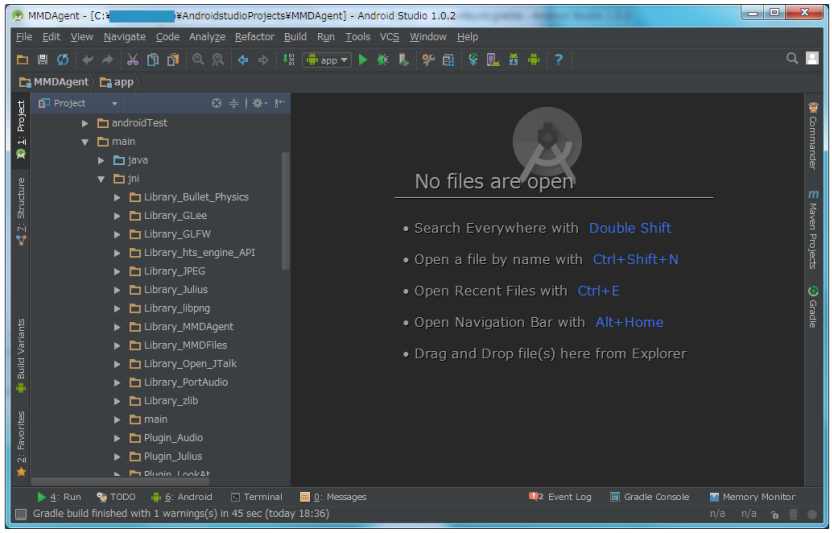


図 17 : jni フォルダにソースコードをコピーした様子

MMDAgent-1.5 フォルダの中には Android 版では使用しないファイルも含まれているが，気にならなければそのままでよい．もし気になるようであれば，消しても構わない．

6. マニフェストファイルの編集

6.1 簡単な説明

次にアプリのマニフェストファイルを編集する．マニフェストファイルとは，このアプリはどんな画面があり，どんな機能が必要で，どんなことをするかを示すファイルである．

マニフェストファイルは MMDAgent¥app¥main¥AndroidManifest.xml である．このファイルを開き編集していく．編集前の内容はソースコード 1 のようになっているはずだ．

ソースコード 1：編集前の AndroidManifest.xml

```
1  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2      package="com.example.myapplication">
3
4      <application
5          android:allowBackup="true"
6          android:label="@string/app_name"
7          android:icon="@drawable/ic_launcher"
8          android:theme="@style/AppTheme">
9
10     </application>
11
12 </manifest>
```

編集後の内容はソースコード 2 のとおりである．

ソースコード 2：編集後の AndroidManifest.xml

```
1    <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2        package="com.example.mmdagent">
3
4        <!-- add permission -->
5        <uses-permission
6            android:name="android.permission.READ_EXTERNAL_STORAGE" />
7        <uses-permission
8            android:name="android.permission.RECORD_AUDIO" />
9
10         <application android:allowBackup="true"
11           android:label="@string/app_name"
```

```
12          android:icon="@drawable/ic_launcher"
13          android:theme="@style/AppTheme">
14
15          <!-- add NativeActivity -->
16          <activity android:name="android.app.NativeActivity"
17              android:label="@string/app_name">
18              <meta-data android:name="android.app.lib_name"
19                  android:value="main" />
20              <intent-filter>
21                  <action android:name="android.intent.action.MAIN" />
22                  <category android:name="android.intent.category.LAUNCHER" />
23              </intent-filter>
24          </activity>
25
26      </application>
27  </manifest>
```

基本的には，編集後のソースコードをそのままコピーすればよいが，要素 manifest の属性 package の値はアプリのパッケージ名に変更すること．

6.2 変更点の詳細

変更部分は，パーミッションの追加と activity 要素の追加である．以降で変更部分の説明をするが，Android アプリの開発経験がない方や，とりあえず MMDAgent が動けば良いという方は次の章まで進んでも構わない．

まず，パーミッションの追加について説明する．編集後のソースコードの 5,6 行目で外部ストレージの読み込み権限を追加した．これは，外部ストレージに保存してある MMDAgent のスクリプトを読み込むためである．7,8 行目でマイクの使用権限を追加した．これは，MMDAgent で音声認識をするためである．

次に要素 activity の追加について説明する．16 行目から 24 行目までが新たに追加した部分である．要素 application の子要素として要素 activity を追加する．要素 activity の属性 android:label の値は変えてもよいし，変えなくてもよい．

18, 19 行目で要素 activity に要素 meta-data が追加している．これは NativeActivity が読み込む共有ライブラリの名前を指定している部分である．共有ライブラリは C/C++ で書かれたプログラムを後でコンパイルして作成することになる．

次に，20 行目から 23 行目で要素 activity に要素 intent-filter を追加している．21 行目が，このアクティビティがこのアプリのメインであるということを示す記述で，22 行目が Android のランチャーに登録させることを示す記述である．

（補足）
『NativeActivity を使用する場合は，application 要素に属性「android:hasCode=”false”」を追加する.』と書かれた Web サイトが多く見つかる．Android 版 MMDAgent では NativeActivity を使用しているが，筆者の環境では android:hasCode=”false” を記述しなくても動作することを確認した．属性 android:hasCode はアプリに Java のコードが含まれるかどうかを意味する．この記述をすることで，Android のシステムはアプリの起動時に Java コードをロードしようとしなくなる．アプリの起動を速くしたい場合はこの属性を指定すると良いだろう．ここで注意しなければいけないことがある．Android 版 MMDAgent を拡張し，**アプリに Java コードを含めた場合は，android:hasCode 属性の値を「true」に替える必要がある**．さらに補足しておくと，android:hasCode 属性が記述されていない場合は，「true」として扱われる．

7. local.properties の編集

次に local.properties の編集する．MMDAgent¥local.properties というファイルを開くと図 18 のような画面が表示される．

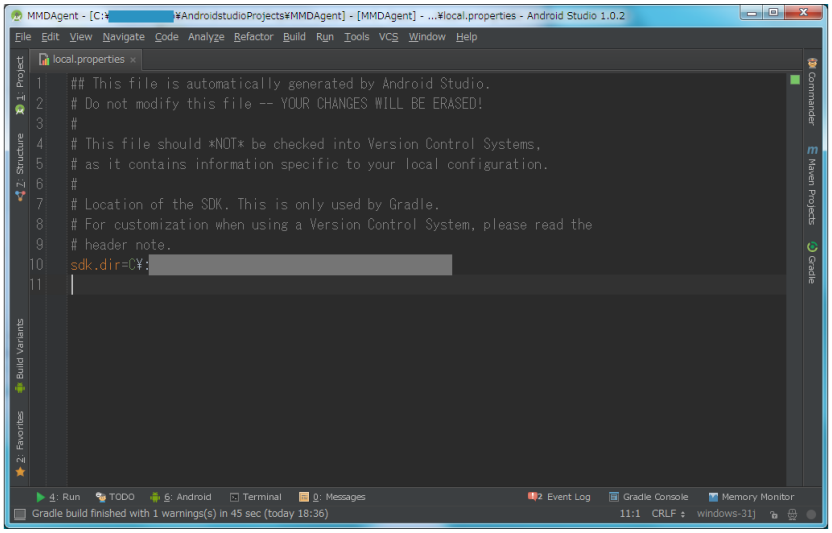


図 18 : MMDAgent¥local.properties を開いた様子

「sdk.dir=<Android SDK のパス>」と書かれた行の下に，「ndk.dir=<Android NDK のパス>」と記述する．「<Android SDK のパス>」，「<Android NDK のパス>」と書かれた部分は開発環境に合わせて変えること．また，Windows のフォルダ区切り記号としての円マーク「¥」（環境によってはバックスラッシュ「¥」）がエスケープ文字として認

識されてしまうため，「¥¥」（「¥¥」）と入力すること．編集後の内容はソースコード 3 のようになっているはずだ．

ソースコード 3：編集後の local.properties

```
1    ## This file is automatically generated by Android Studio.
2    # Do not modify this file -- YOUR CHANGES WILL BE ERASED!
3    #
4    # This file should *NOT* be checked into Version Control Systems,
5    # as it contains information specific to your local configuration.
6    #
7    # Location of the SDK. This is only used by Gradle.
8    # For customization when using a Version Control System, please read the
9    # header note.
10   sdk.dir=<android SDK のパス>
11   ndk.dir=<android NDK のパス>
```

8. build.gredle の編集

8.1 簡単な説明

次に build.gredle を編集する．編集するファイルは MMDAgent¥app¥build.gredle である．MMDAgent¥build.gredle ではないので注意すること．MMDAgent¥app¥build.gredle を開こう．

Android Studio では，ビルドツールとして Gradle が使われている．build.gredle は Gradle プロジェクトをどのようにビルドするかを規定するためのファイルである．GNU make における Makefile, Apache Ant における build.xml にあたるものが build.gredle である．

編集前の内容はソースコード 4 のようになっているはずだ．

ソースコード 4：編集前の app¥build.gradle

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 21
    buildToolsVersion "21.1.2"

    defaultConfig {
        applicationId "com.example.mmdagent"
        minSdkVersion 15
        targetSdkVersion 21
        versionCode 1
        versionName "1.0"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles(
                   getDefaultProguardFile('proguard-android.txt'),
                   'proguard-rules.pro');
        }
    }
}
```

編集後の内容をソースコード 5 に示す．Gradle のバージョンアップなどで変わらない限り，編集後のソースコードをそのままコピーすればよい．以降では編集内容について詳しく説明していく．とりあえず MMDAgent が動けばよいという方は次の章まで進んでも構わない．

ソースコード 5：編集後の app¥build.gradle

```
apply plugin: 'com.android.application'
import org.apache.tools.ant.taskdefs.condition.Os

android {
    compileSdkVersion 21
    buildToolsVersion "21.1.2"

    defaultConfig {
        applicationId "com.example.mmdagent"
        minSdkVersion 15
        targetSdkVersion 21
        versionCode 1
        versionName "1.0"
    }

    sourceSets.main.jni.srcDirs = []    // avoid using NdkCompile task

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles(
                    getDefaultProguardFile('proguard-android.txt'),
                    'proguard-rules.pro');
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:21.0.3'
}

task makeConfigFile << {
    def configFile1 = file(
            "src/main/jni/Library_Julius/include/julius/config.h");
    configFile1.createNewFile()
    configFile1.write('#define JULIUS_PRODUCTNAME ""¥n')
    configFile1.append('#define JULIUS_VERSION "4.3"¥n')
```

```
39      configFile1.append('#define JULIUS_SETUP "fast"¥n')
40      configFile1.append('#define JULIUS_HOSTINFO ""¥n')
41      configFile1.append('#define RETSIGTYPE void¥n')
42      configFile1.append('#define STDC_HEADERS 1¥n')
43      configFile1.append('#define UNIGRAM_FACTORING 1¥n')
44      configFile1.append('#define LOWMEM2 1¥n')
45      configFile1.append('#define PASS1_IWCD 1¥n')
46      configFile1.append('#define SCAN_BEAM 1¥n')
47      configFile1.append('#define GPRUNE_DEFAULT_BEAM 1¥n')
48      configFile1.append('#define CONFIDENCE_MEASURE 1¥n')
49      configFile1.append('#define LM_FIX_DOUBLE_SCORING 1¥n')
50      configFile1.append('#define GRAPHOUT_DYNAMIC 1¥n')
51      configFile1.append('#define GRAPHOUT_SEARCH 1¥n')
52      configFile1.append('#define HAVE_STRCASECMP 1¥n')
53
54      def configFile2 = file(
55              "src/main/jni/Library_Julius/include/sent/config.h");
56      configFile2.createNewFile()
57      configFile2.write('#define LIBSENT_VERSION "4.3"¥n')
58      configFile2.append('#define AUDIO_API_NAME ""¥n')
59      configFile2.append('#define AUDIO_API_DESC ""¥n')
60      configFile2.append('#define AUDIO_FORMAT_DESC ""¥n')
61      configFile2.append('#define GZIP_READING_DESC ""¥n')
62      configFile2.append('#define STDC_HEADERS 1¥n')
63      configFile2.append('#define USE_MIC 1¥n')
64      configFile2.append('#define USE_ADDLOG_ARRAY 1¥n')
65      configFile2.append('#define HAVE_SOCKLEN_T 1¥n')
66      configFile2.append('#define HAVE_UNISTD_H 1¥n')
67      configFile2.append('#define HAVE_ZLIB 1¥n')
68      configFile2.append('#define HAVE_STRCASECMP 1¥n')
69      configFile2.append('#define HAVE_SLEEP 1¥n')
70      configFile2.append('#define CLASS_NGRAM 1¥n')
71      configFile2.append('#define MFCC_SINCOS_TABLE 1¥n')
72  }
73
74  task buildNative(type:Exec) {
75      def ndkDir =
76          project.plugins.findPlugin('com.android.application').getNdkFolder()
77      def jOption = '-j'+Runtime.runtime.availableProcessors()
78      if(Os.isFamily(Os.FAMILY_WINDOWS)){
79          commandLine("$ndkDir/ndk-build.cmd", jOption,
80                  '-C', file('src/main').absolutePath,
81                  'NDK_APP_LIBS_OUT=jniLibs');
82      }else{
83          commandLine("$ndkDir/ndk-build", jOption,
84                  '-C', file('src/main').absolutePath,
```

```
                'NDK_APP_LIBS_OUT=jniLibs');
    }
}

buildNative.dependsOn 'makeConfigFile'
tasks.withType(JavaCompile) {
    compileTask -> compileTask.dependsOn 'buildNative'
}

task cleanConfigFile << {
    def configFile1 = file(
            'src/main/jni/Library_Julius/include/julius/config.h');
    configFile1.delete();
    def configFile2 = file(
            'src/main/jni/Library_Julius/include/sent/config.h');
    configFile2.delete();
}

task cleanNative(type:Exec){
    def ndkDir =
        project.plugins.findPlugin('com.android.application').getNdkFolder()
    if(Os.isFamily(Os.FAMILY_WINDOWS)){
        commandLine("$ndkDir/ndk-build.cmd", 'clean',
                '-C', file('src/main').absolutePath,
                "NDK_APP_LIBS_OUT=jnilibs");
    }else{
        commandLine(
                "$ndkDir/ndk-build", 'clean',
                '-C', file('src/main').absolutePath,
                "NDK_APP_LIBS_OUT=jnilibs");
    }
}

cleanNative.dependsOn 'cleanNative'
clean.dependsOn 'cleanNative'
```

8.2 変更点の詳細

MMDAgent のソースコードは make でのビルドが想定されていて，ソースコードの中には Makefile が含まれている．しかし，Android Studio では make でのビルドはできない．そこで，Makefile の内容を build.gradle で再現するという作業を行う．

33 行目から 72 行目までは makeConfigFile というタスクを定義している部分である．これは，MMDAgent で使われている Julius という音声認識モジュールの設定ファイルを生成するというタスクである．このタスクを実行すると MMDAgent¥app¥src¥main¥jni¥Library_Julius¥include¥julius¥config.h と MMDAgent¥app¥src¥main¥jni¥Library_Julius¥include¥sent¥config.h というファイルが生成される．config.h の内容については Julius のドキュメントを参考にすること．

16 行目に「sourceSets.main.jni.srcDirs = []」と書かれている．Android Studio では sourceSets.main.jni.srcDirs で指定されたフォルダを C/C++ のコードが含まれるフォルダとしてビルドしようとする．（指定がない場合は jni フォルダが設定される．）しかし，Android Studio による C/C++ のビルドは Android.mk というファイルが読み込まれない．今回は C/C++ のビルドに Android NDK に含まれている ndk-build コマンドを用いるため，Android Studio によるビルドが行われないように srcDirs を空にする．ちなみに，Android.mk は ndk-build でのビルド方法について記述したファイルであり，make における Makefile のようなファイルである．

次に，74 行目から 87 行目では，buildNative というタスクを定義している．このタスクは ndk-build コマンドを実行して C/C++ のコードをコンパイルするタスクである．

75,76 行目では，Android NDK のフォルダのパスを取得し変数 ndkDir に代入している．また，77 行目では ndk-build の –j*n* というオプションを指定している．-j*n* オプションは *n* の値だけジョブを発行し，並列ビルドするというオプションである．*n* の部分には，使用中のコンピュータの CPU のコア数が入るように記述してある．

そして 78 行目の記述は使用中の OS を判断して分岐するというものである．分岐した先ではどちらも ndk-build を実行する．commandLine は以降の文字列をコマンドプロンプトやターミナルで実行することを表し，$ndkDir/ndk-build(.cmd) は ndk-build コマンドを表す．jOption は先ほど説明した通りで，’-C’, file(src/main).absolutepath で ndk-build を実行するディレクトリを src/main にするということを表し，’NDK_APP_LIBS_OUT = jniLibs” でコンパイルして生成したファイルを jniLibs フォルダに入れるということを表している．

続いて，今までで作成したタスクの依存関係を設定し，ビルドするときに作成したタスクが実行されるようにする．89 行目から 92 行目までが依存関係を設定している部分である．89 行目で buildNative タスクが makeConfigFile タスクに依存していることを示し，90 行目から 92 行目で JavaCompile タスクが buildNative タスクに依存していることを示す記述である．JavaCompile タスクは Android アプリのビルドをするときに必ず実行されるタスクである．

最後に 94 行目から 116 行目までは clean に関する記述である．cleanConfigFile タスクは config.h を削除するタスクで，cleanNative タスクは ndk-build で生成したファイルを削除するタスクである．118 行目と 119 行目は依存関係の設定に関する部分である．

9. Android 端末にスクリプト等のファイルを設置

MMDAgent では，設定ファイルや対話シナリオ，3D モデル，モーションファイルを Android 端末から読み込む．この章では，スクリプト等のファイルを設置する方法について説明する．この章の作業では Android Studio は使わない．

あらかじめダウンロードしておいた MMDAgent のサンプルスクリプト MMDAgent_Example-1.4 を Android 端末にコピーする．コピーする先は Android 端末の内部ストレージでも SD カードでもなんでも良い．どのフォルダにコピーしても良いが，コピーした先のパスをメモしておくこと．コピーした先のパスは Android 端末のファイラアプリで確認できる．ファイラアプリは端末によってはインストールされてない場合があるため，Google Play 等からインストールしなければならない場合がある．内部ストレージをもつ Android 端末でも内部ストレージのパスは /sdcard である可能性が高いが，端末によってパスが変わるので確認すること．

次にソースコードのフォルダ MMDAgent-1.5 の中に Release¥AppData というフォルダがある．このフォルダも Android 端末にコピーする．このフォルダもどのフォルダにコピーしても良いが，MMDAgent_Example-1.4 の中に入れると良いだろう．AppData フォルダのパスもメモしておくこと．

10. Android.mk の編集

アプリのパッケージ名やスクリプトの設置場所を指定するため，Android.mk を編集する必要がある．再び Android Studio での作業に戻る．

Android Studio で MMDAgent¥app¥src¥main¥jni¥Library_MMDAgent¥Android.mk を開く．**Android.mk という名前のファイルは別のフォルダにもあるため，間違えないように注意すること．**Library_MMDAgent¥Android.mk の内容は次のようになっている．

ソースコード 6 : jni¥Library_MMDAgent¥Android.mk

```
1  LOCAL_PATH := $(call my-dir)
2
3  include $(CLEAR_VARS)
4
5  LOCAL_MODULE     := MMDAgent
6  LOCAL_SRC_FILES  := src/lib/BoneController.cpp ¥
7                      src/lib/LipSync.cpp ¥
8                      src/lib/LogText.cpp ¥
9                      src/lib/Message.cpp ¥
10                     src/lib/MMDAgent.cpp ¥
11                     src/lib/MMDAgent_utils.cpp ¥
12                     src/lib/MotionStocker.cpp ¥
13                     src/lib/Option.cpp ¥
14                     src/lib/PMDObject.cpp ¥
15                     src/lib/Plugin.cpp ¥
16                     src/lib/Render.cpp ¥
17                     src/lib/ScreenWindow.cpp ¥
18                     src/lib/Stage.cpp ¥
19                     src/lib/TextRenderer.cpp ¥
20                     src/lib/TileTexture.cpp ¥
21                     src/lib/Timer.cpp
22 LOCAL_C_INCLUDES := $(LOCAL_PATH)/include ¥
23                     $(LOCAL_PATH)/../Library_JPEG/include ¥
24                     $(LOCAL_PATH)/../Library_Bullet_Physics/include ¥
25                     $(LOCAL_PATH)/../Library_GLee/include ¥
26                     $(LOCAL_PATH)/../Library_libpng/include ¥
27                     $(LOCAL_PATH)/../Library_zlib/include ¥
28                     $(LOCAL_PATH)/../Library_MMDFiles/include ¥
29                     $(LOCAL_PATH)/../Library_GLFW/include
30 LOCAL_CFLAGS     += -DMMDAGENT_DONTRENDERDEBUG ¥
31                     -DMMDAGENT_DONTUSESHADOWMAP ¥
32                     -DMMDAGENT_DONTPICKMODEL ¥
33                     -DMMDAGENT_DONTUSEMOUSE ¥
34                     -DMMDAGENT_DONTUSEWINDOW ¥
```

```
35                      -DMMDAGENT ¥
36      -DMMDAGENT_OVERWRITEEXEFILE="¥"/sdcard/MMDAgent/MMDAgent.exe¥"" ¥
37      -DMMDAGENT_OVERWRITECONFIGFILE="¥"/sdcard/MMDAgent/MMDAgent.mdf¥"" ¥
38      -DMMDAGENT_OVERWRITESYSDATADIR="¥"/sdcard/MMDAgent/AppData¥"" ¥
39      -DMMDAGENT_OVERWRITEPLUGINDIR="¥"/data/data/com.example.mmdagent/lib¥""
40
41  include $(BUILD_STATIC_LIBRARY)
```

36 行目から 38 行目までが書き換える部分である．C/C++ の経験がある方はわかるかもしれないが，これらの部分は C/C++ のマクロを設定する部分である．コンパイラのマクロ設定のオプションの書き方とほぼ同じである．

後でそれぞれの行ごとにも説明するが,「-DMMDAgent…=”¥”<path>¥”” ¥」の <path> の部分を，コピーしたスクリプト等のパスに書き換える．「”¥”…¥””」と一見おかしな表現があるが消さないように注意して欲しい．外側の「”」で囲まれた部分がマクロの値で,「¥”」は「”」を表すエスケープシーケンスで, マクロを展開後の値は「”<path>”」であることを示している．

まず，37 行目を先に説明する．37 行目は .mdf という拡張子を持つ設定ファイルの場所を指定する部分である. 先ほど Android 端末にコピーしたスクリプトの中に .mdf という拡張子を持つファイルが存在する．そのファイルのパスに書き換える

次に，36 行目の説明をする．36 行目は .exe という拡張子を持つファイルの場所を指定する部分である. しかし, 先ほど Android 端末にコピーしたスクリプトの中に .exe という拡張子はない．代わりに .fst という拡張子を持つファイルがあるので，そのファイルの .fst の部分を .exe に書き換えたものを指定する．Windows 版では .exe という拡張子の実行ファイルを生成し, そのファイルの拡張子を .fst などに書き換えたスクリプトを自動で読み込むという動作があるためこのようにする．

38 行目は第 9. 章でコピーした AppData フォルダのパスに書き換える．

39 行目は今までの書き換え方と少し変わるため注意する. この部分は共有ライブラリのインストール先を設定する部分である．共有ライブラリのインストール先はアプリのパッケージ名に依存する．ここでは「/data/data/<packageName>/lib」の <packageName> の部分を第 4. 章で設定したパッケージ名に書き換える．

11. プロジェクトのビルドおよび実行

ここまでの作業が終われば, あとはビルドして Android 端末で実行するだけである. まず，Android 端末とパソコンを USB ケーブルで接続する．Android Studio の画面を見よう.

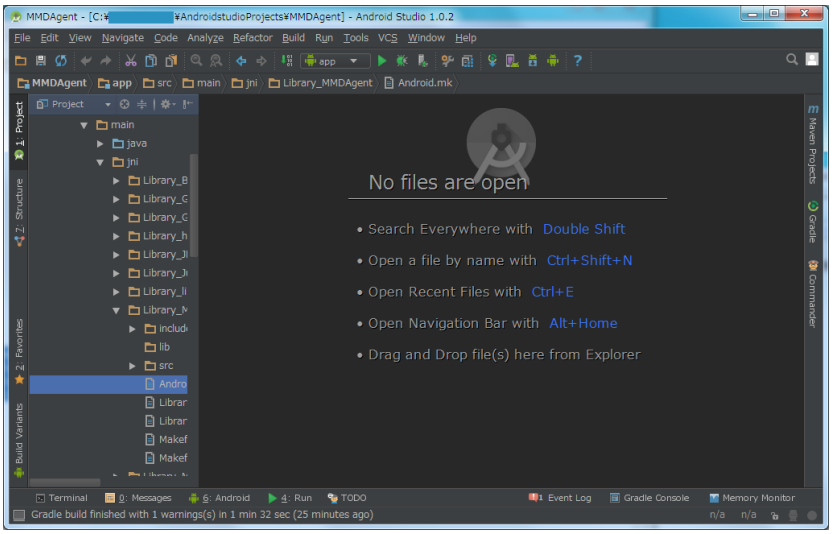


図 19 : Android Studio の画面

緑色の三角形のボタンがある．その左側が App になっていることを確認する．もしここで，App になっていなかったらその部分をクリックして App に変えること．その後，緑の三角形のボタンを押すとビルドが開始される．C/C++ のコンパイルに時間がかかるためしばらく待つ．フリーズしたか心配になったら下の方の「Gradle: Executing tasks」の左でグルグルしているか確認する.

ビルドが終わると図 20 のような画面が表示される.

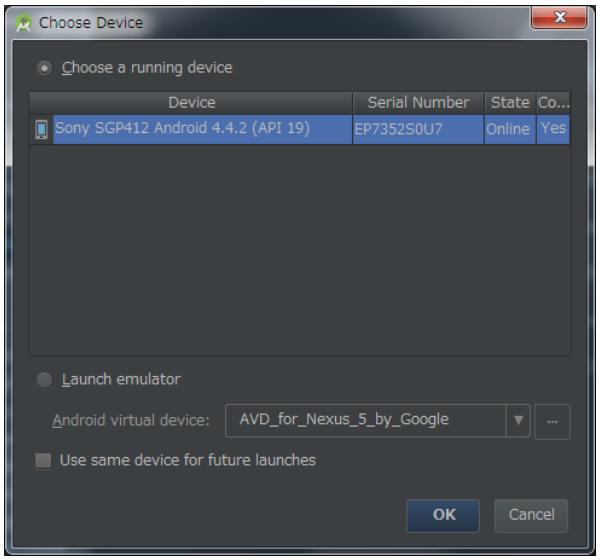


図 20 : 実行デバイス選択画面

ここで「Choose a running device」のラジオボックスにチェックを入れ，実行する端末を選択し，OK ボタンを押す．もし，端末を選択できない場合は，一度 Android 端末の画面を見る．「USB デバッグを許可しますか?」といった表示が出ている場合は，OK を選択する．そうすることでこの端末を選択できるようになるはずである．それでもダメな場合は，USB ケーブルをつなぎなおしたり，開発者オプションを表示できるようにしたかどうか，ドライバが必要かどうかを確認する．

無事実行できた場合は図 21 のような画面が Android 端末に表示されるはずである．

図 21 : Android で MMDAgent が起動した様子

12. 動かない時は？

12.1 ビルドできない

まず，どんなエラーが出ているのかコンソールで確認する．ビルドできない原因としては次のものが考えられる．

- build.gradle の記述に誤りがある
- local.properties で NDK のパスに誤りがある
- Android.mk の記述に誤りがある

この段階での問題は，エラーがしっかりでるためエラーの内容を調べれば解決できるはずだ．

12.2 アプリが起動できない

「問題が発生したためアプリを終了します」というようなメッセージが出てアプリが起動できない場合は次の原因が考えられる．

- AndroidManifest.xml の記述に誤りがある
- Library_MMDAgent¥Android.mk の記述に誤りがある

AndroidManifest.xml については，パーミッションの設定が間違っていないか，activity の設定が間違っていないかを確認する．

Android.mk については，第 10. 章での編集内容が間違っていないか確認する．パッケージ名が間違っている場合は，共有ライブラリが読み込めずにアプリが起動しない場合がある．スクリプトファイルのパスを間違えていた場合もアプリが起動しない場合がある．

12.3 起動したが画面が黒いまま

黒い画面のままであるということは，Activity は起動できたがプログラムは動いていないという状態である．この場合に原因として考えられるものは，Library_MMDAgent¥Android.mk の記述に誤りがあることだ．第 10. 章での編集内容が間違っていないか確認する．パッケージ名が間違っている可能性が高い．

12.4 起動したが画面が青いまま

青い画面が起動しているということは，OpenGL は動作しているが MMDAgent が正常に動作していないということである．この場合に原因として考えられるものは，次の通りである．

- Library_MMDAgent¥Android.mk の記述に誤りがある
- Android 端末に AppData フォルダをコピーし忘れた
- 対話スクリプト (FST) が正しくない

12.5 音声を認識しない（音声レベルバーが表示されない）

この状態の場合に考えられる原因は次のとおりである．

- 使用中の Android 端末にマイクがない
- 他のアプリがマイクを占有している
- Android 端末に AppData フォルダをコピーし忘れた
- Julius の設定に誤りがある

音声レベルバーが表示されない場合は，音声認識エンジン Julius が正しく動作していない．Julius が正しく動作しない理由としては，何らかの理由でマイクが利用できない，Julius の設定が間違っているなどが考えられる．

12.6 メイちゃんが反応しない

この場合の原因としては次のことが考えられる．

- 意図したとおりに音声が認識されていない
- マイクの感度が適正値でない
- まわりの雑音を拾っている
- 対話スクリプト（FST）に誤りがある
- 辞書ファイルに単語が登録されていない

MMDAgent が起動し，メイちゃんが画面に表示されている場合は，プログラム的には問題なく実行できているはずだ．PC 版の MMDAgent と同じように，マイク，シナリオ，設定ファイルが正しいか確認すること．マイクの感度が適正値ではないとき，AppData¥julius¥jconf.txt を編集すると良い．

―メイちゃんについて―

- 商用利用は出来ません。
- ご利用の際はコピーライト表示をお願いしています。
- 著しく“メイちゃん”のイメージを損なうご使用はお控えください。

―コピーライト表示方法―

- イラストを使用する際は，以下に示すようにコピーライトを表示してください．
- CC ライセンスロゴマークも同様に添付してください。

※クリエイティブ・コモンズ（CC）ライセンスについて（詳しくは　http://creativecommons.jp/licenses/）