MMDAgent 開発者向けリファレンス

Ver. 1.02

2017 年 1 月 24 日

名古屋工業大学

目次

1. 本文書の位置付け

本仕様書は MMDAgent を拡張する開発者に向けた仕様書である。
この仕様書では、Windows 上の開発環境および実行環境の構築方法、新規プラグインの開発手順、Android 版の開発環境および実行環境の構築方法を説明する。
本仕様書が対象とする MMDAgent のバージョンを以下に記す。

▼対象とするバージョン

項目	バージョン
MMDAgent.exe 本体	1.7
MMDAgent_Example	1.7

2. 開発環境/実行環境の構築（Windows 版）

概要

公開されている MMDAgent のソースコードを、Windows 上で開発および実行できるようにするための手順と、新規プラグインの開発手順をまとめた項目である。

本項目は以下の環境を基に記述するが、Windows 8 等でも開発を行うことは可能である。そうした場合、この仕様書に記述されている内容を、自身の環境に合わせて読み替える必要がある。

項目	対象
開発環境 OS	Windows 7 64bit
開発用ソフトウェア	Visual Studio Community 2013

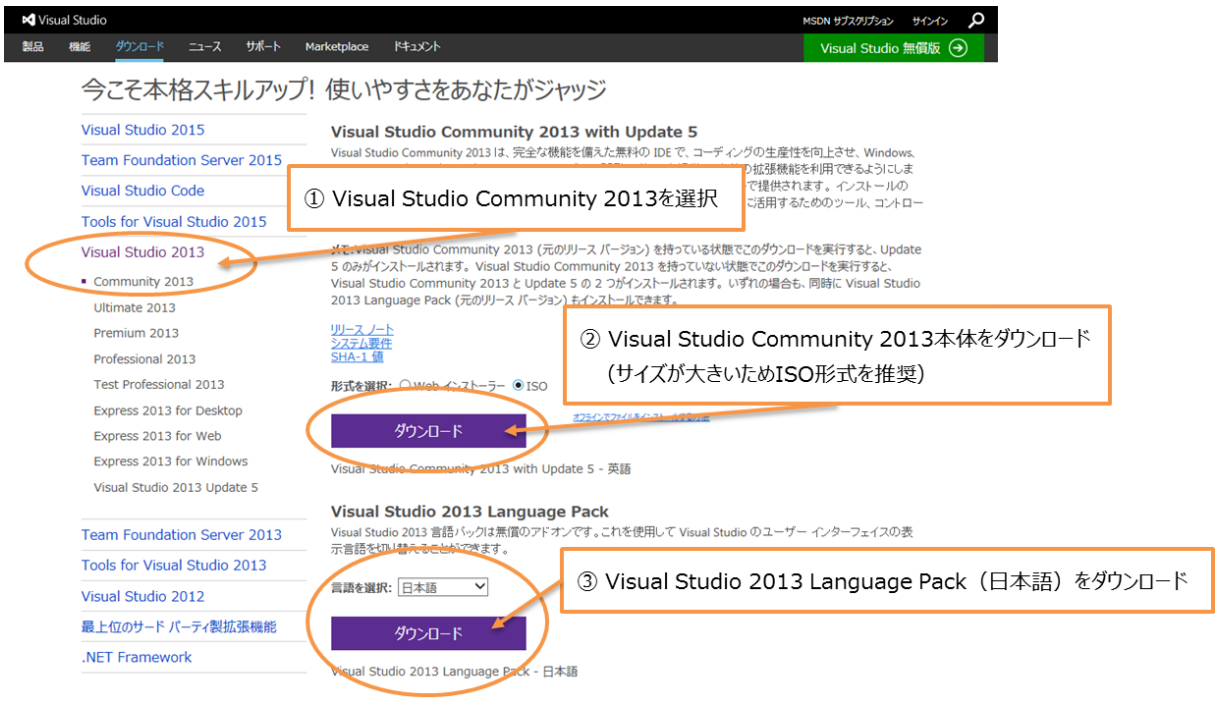
[備考]
公開されている MMDAgent のソリューションファイル（.sln）およびプロジェクトファイル（.vcxproj）は Visual Studio 2010 向けに作成されているが、2010 以降のバージョンであれば、プロジェクトをアップグレード（ソリューションを初めて開くと自動的にアップグレードウィンドウが出現）することで開発できるようになる。

開発環境の構築

Visual Studio Community 2013 のダウンロード

公式サイトから Visual Studio Community 2013 と日本語 Language Pack をダウンロードする。
https://www.visualstudio.com/downloads/download-visual-studio-vs

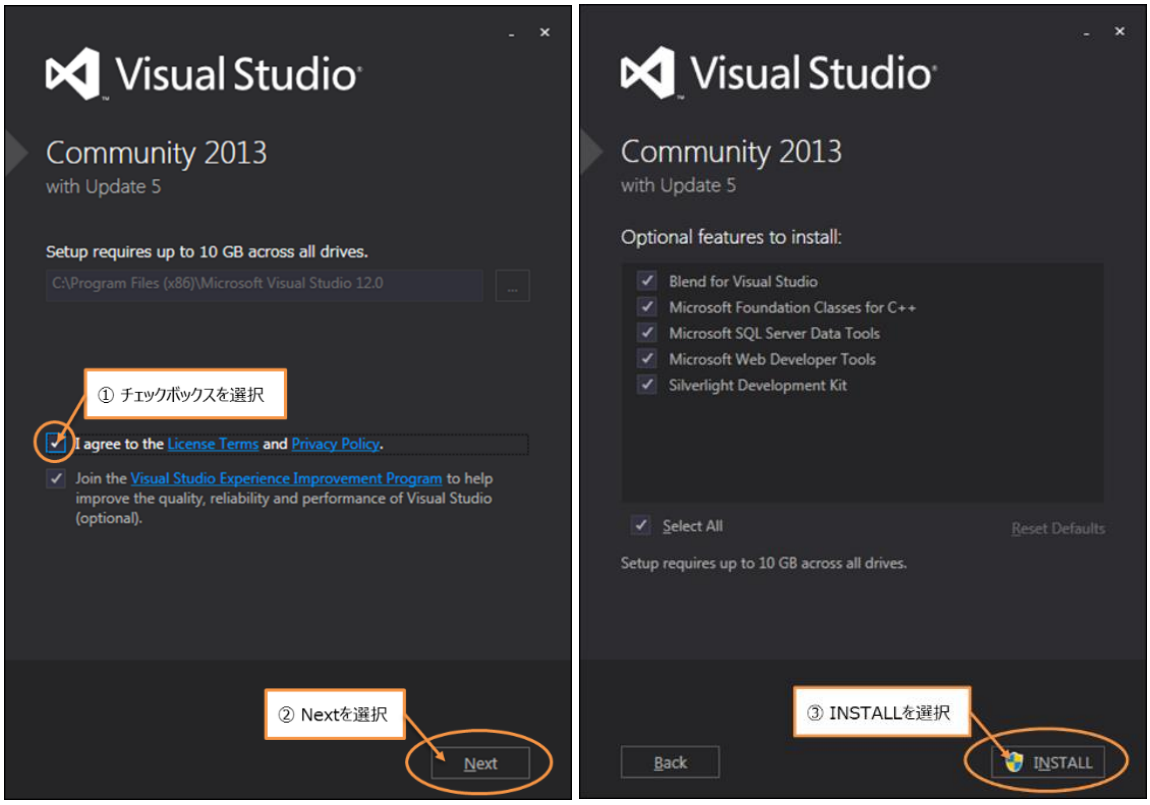
▼手順



Visual Studio Community 2013 のインストール

ダウンロードした本体ファイル（ISO または Web インストーラー）を実行して、
Visual Studio Community 2013 をインストールする。

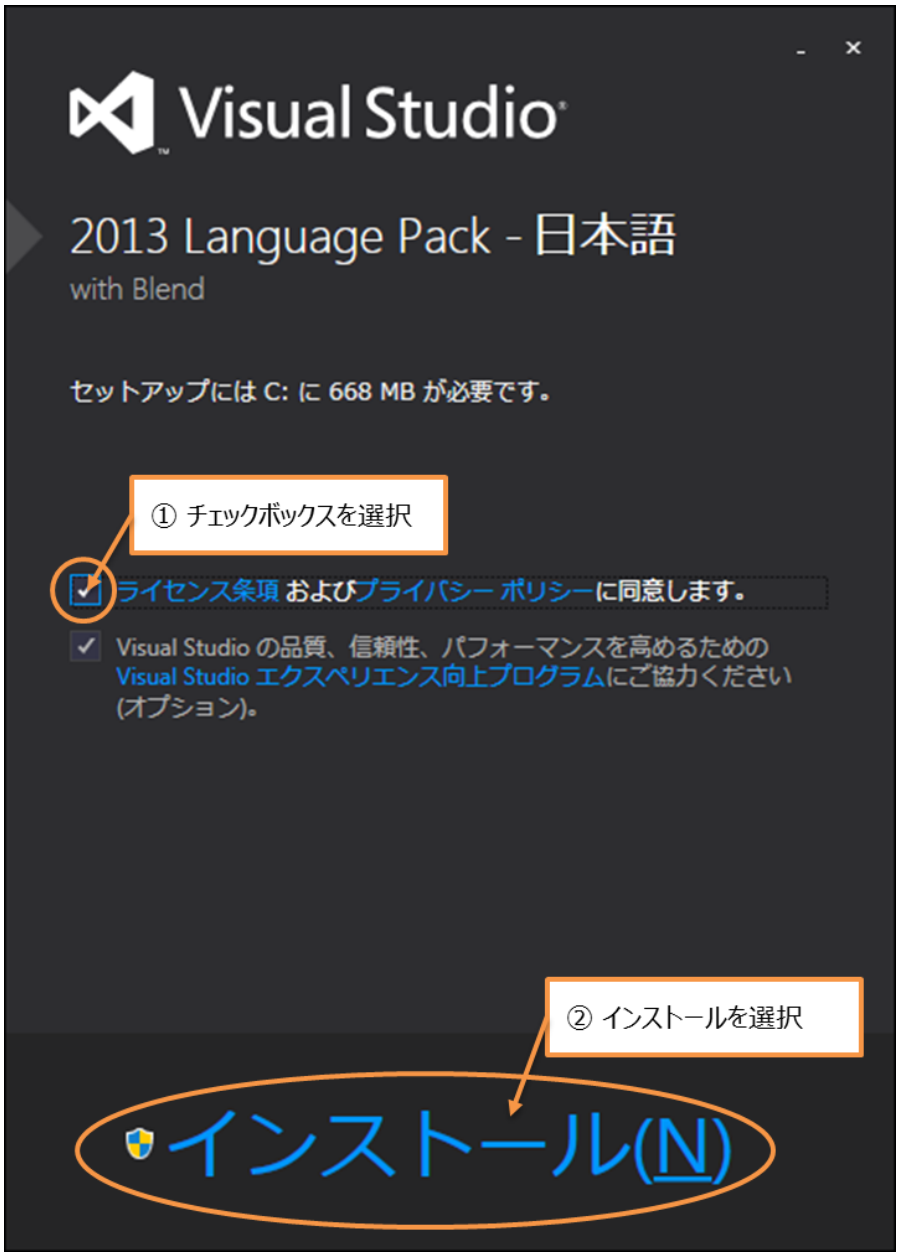
▼手順



Visual Studio 2013 Language Pack のインストール

ダウンロードした Language Pack のインストーラーを実行して、Visual Studio 2013 Language Pack（日本語）をインストールする。なお、実際に日本語化するには Visual Studio の設定を変更する必要があり、変更方法は後述にて記載する。

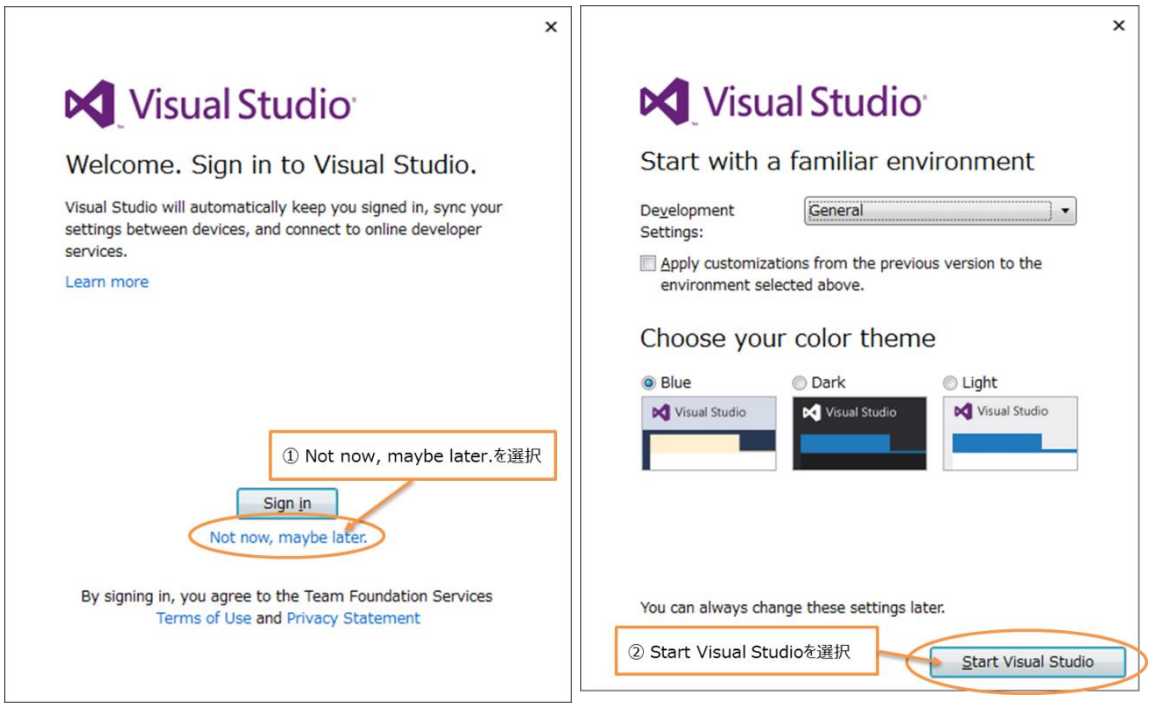
▼手順



Visual Studio Community 2013 の初期設定

インストールした Visual Studio Community 2013 を起動して初期設定を行う。

▼手順（設定ウィンドウは初回起動時に自動的に表示される。）



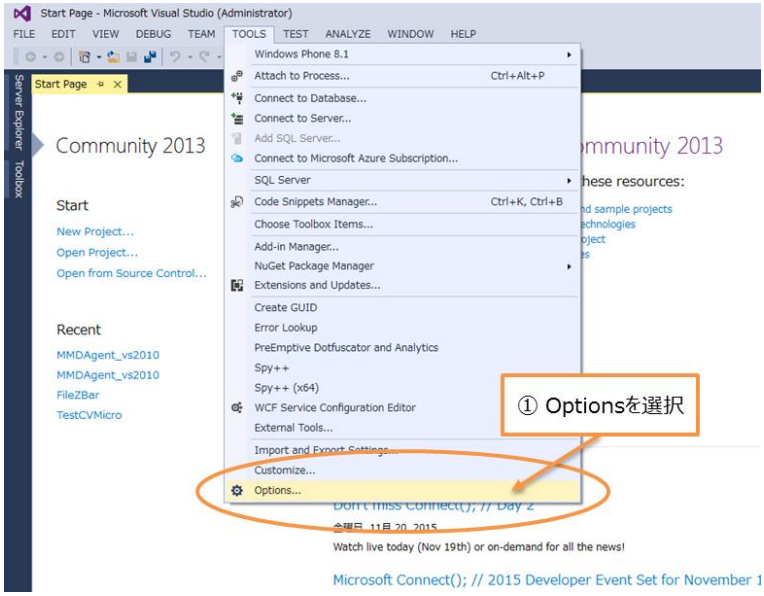
[備考]

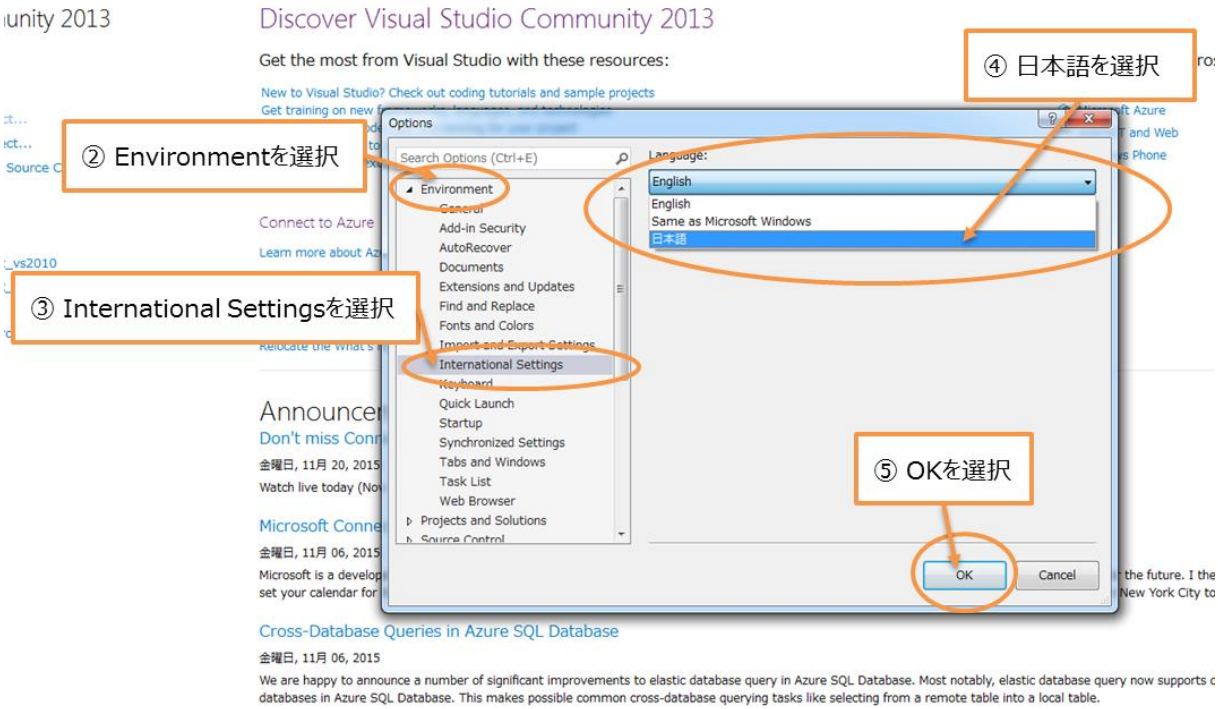
評価期間（30 日）以降も継続利用するには、Microsoft アカウントでサインインする必要がある。

Visual Studio Community 2013 の日本語化

オプション設定を変更して Visual Studio Community 2013 を日本語化する。

▼手順（変更は Visual Studio 再起動後に適用される。）





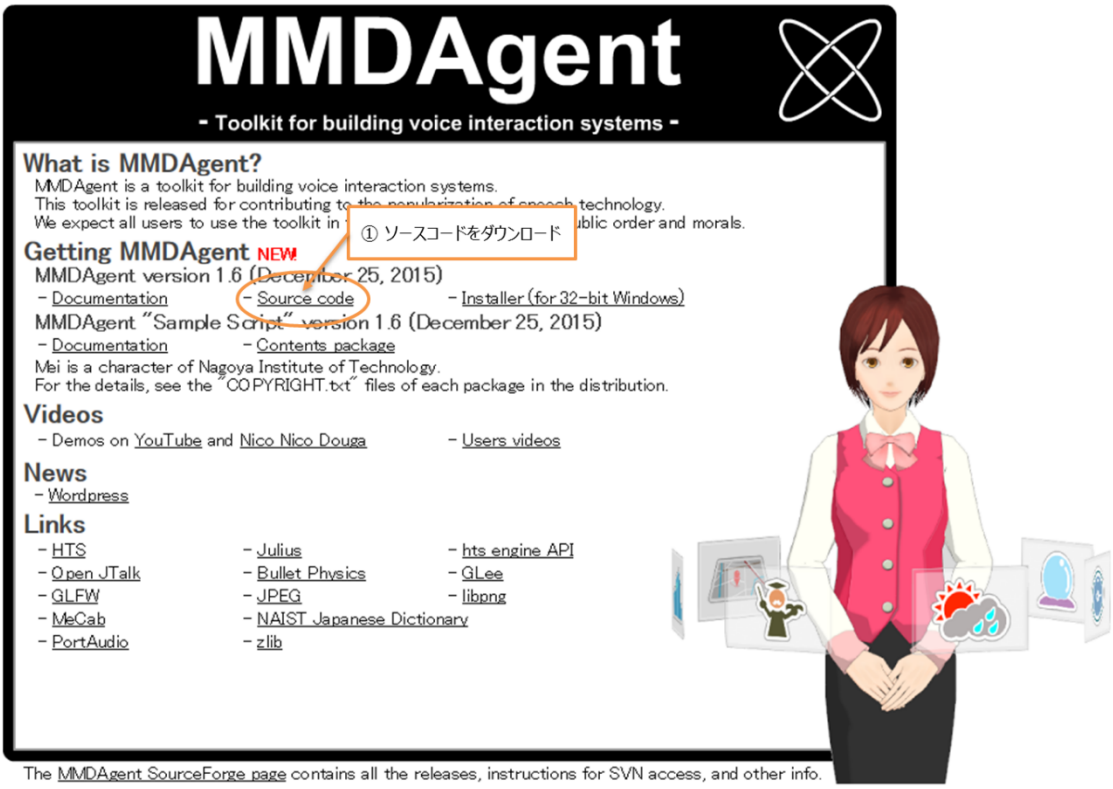
実行環境の構築

ソースコードの取得

公式サイトから MMDAgent のソースコードをダウンロードし、PC 内の任意の場所に保存する。
なお、ソースコードは Zip 圧縮されているため、以降の手順を進めるためには解凍が必要となる。

▼MMDAgent 公式サイト
http://www.mmdagent.jp/

▼手順



ソースコードのビルド/実行

ダウンロードしたソースコードを Visual Studio Community 2013 で実行できる状態にする。

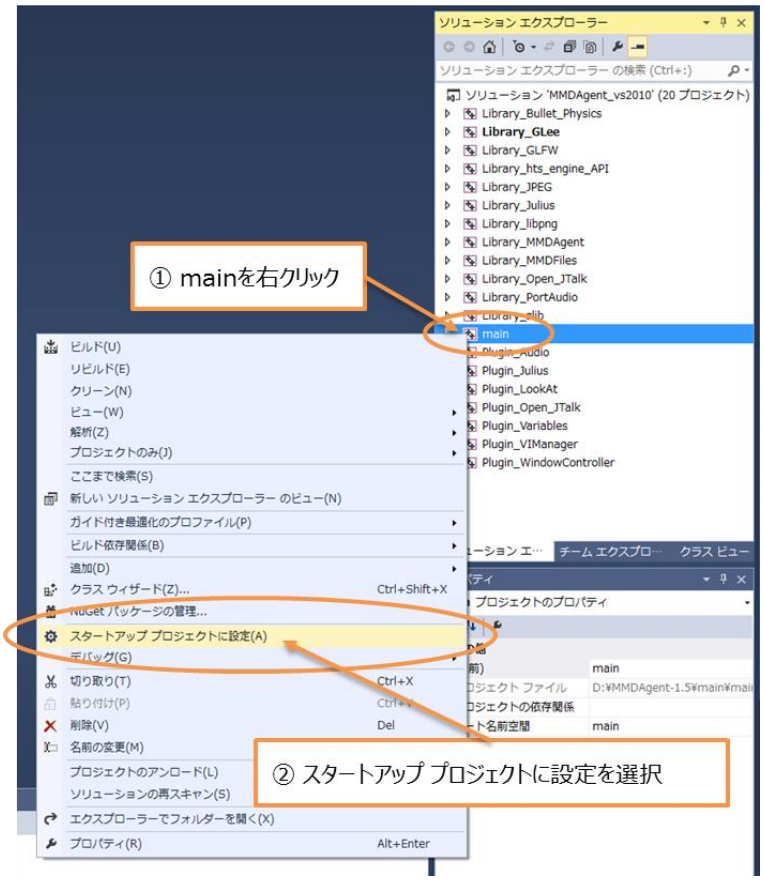
▼手順

1. ソリューションを開く

   解凍したフォルダに存在する MMDAgent_vs2010.sln を Visual Studio Community 2013 で開く。なお、ソリューションを開くとプロジェクトファイルを Visual Studio 2013 用にアップグレードするウィンドウが出現するので、「OK」を選択し全てのプロジェクトをアップグレードする。

2. スタートアップ プロジェクトの設定

main プロジェクトをスタートアップ プロジェクトに設定する。

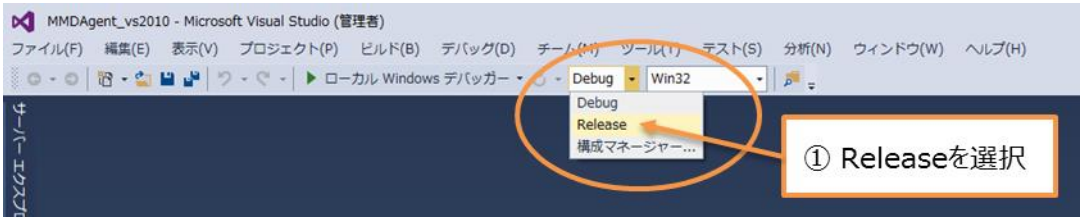


[備考]

スタートアップ プロジェクトに設定されているプロジェクト名は太字で表示される。

3. ソリューション構成の変更

ソリューション構成を Release に変更する。

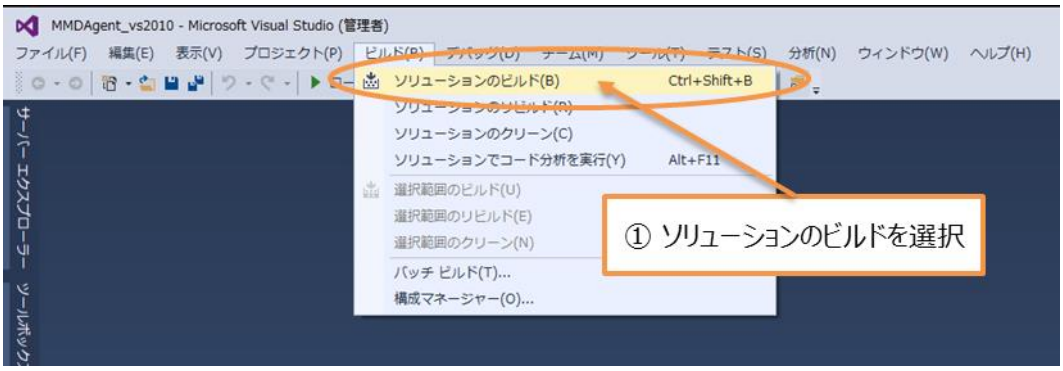


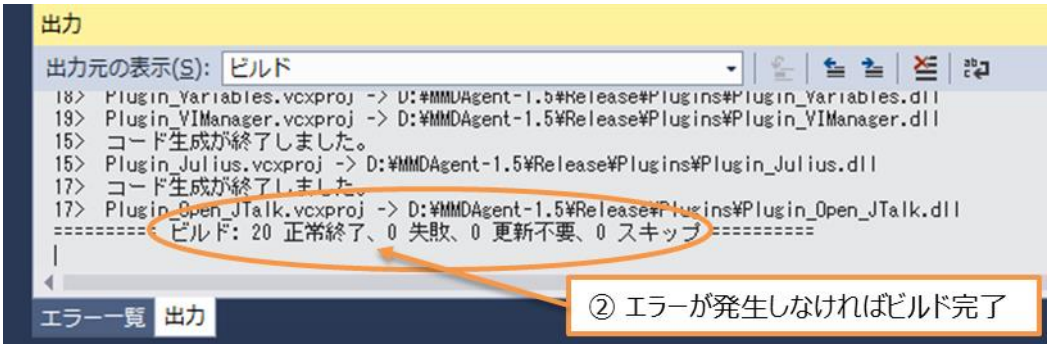
[備考]

Debug 構成で実行したい場合、Release フォルダに存在する AppData と MMDAgent.mdf を Debug フォルダにコピーする。

4. ビルド

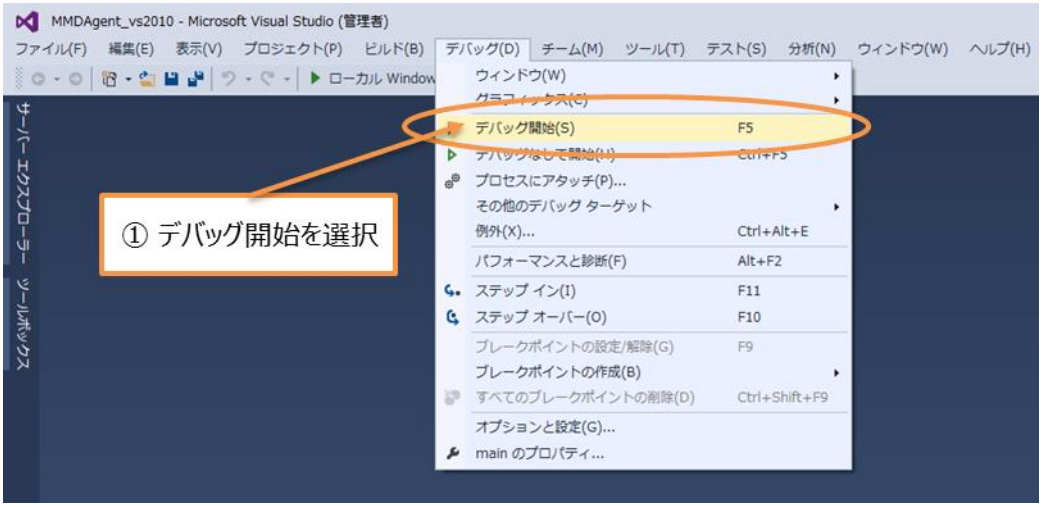
ソリューションをビルドする。

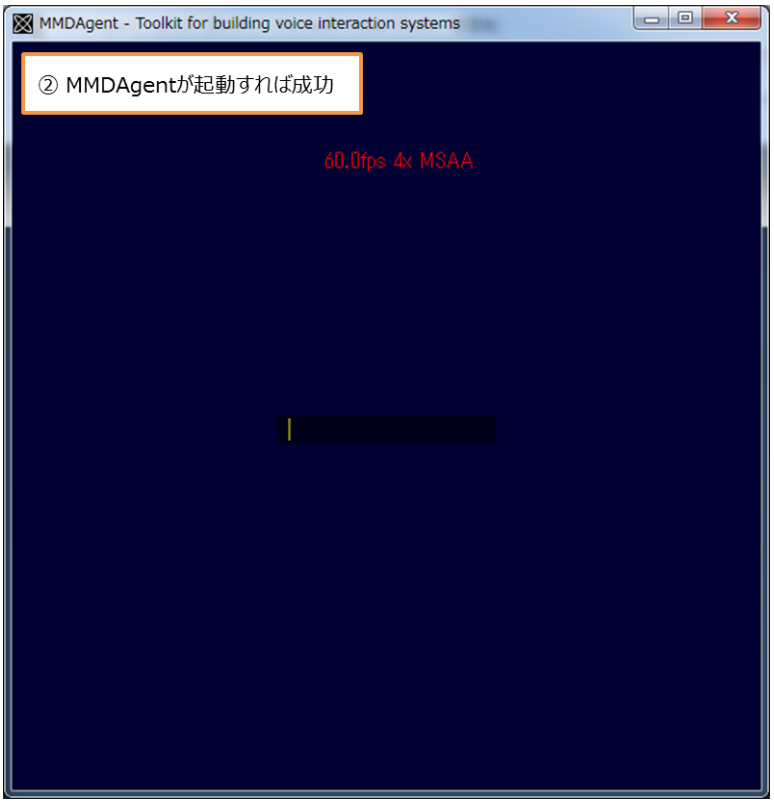




5. 実行

ビルドして生成した exe ファイルを実行する。

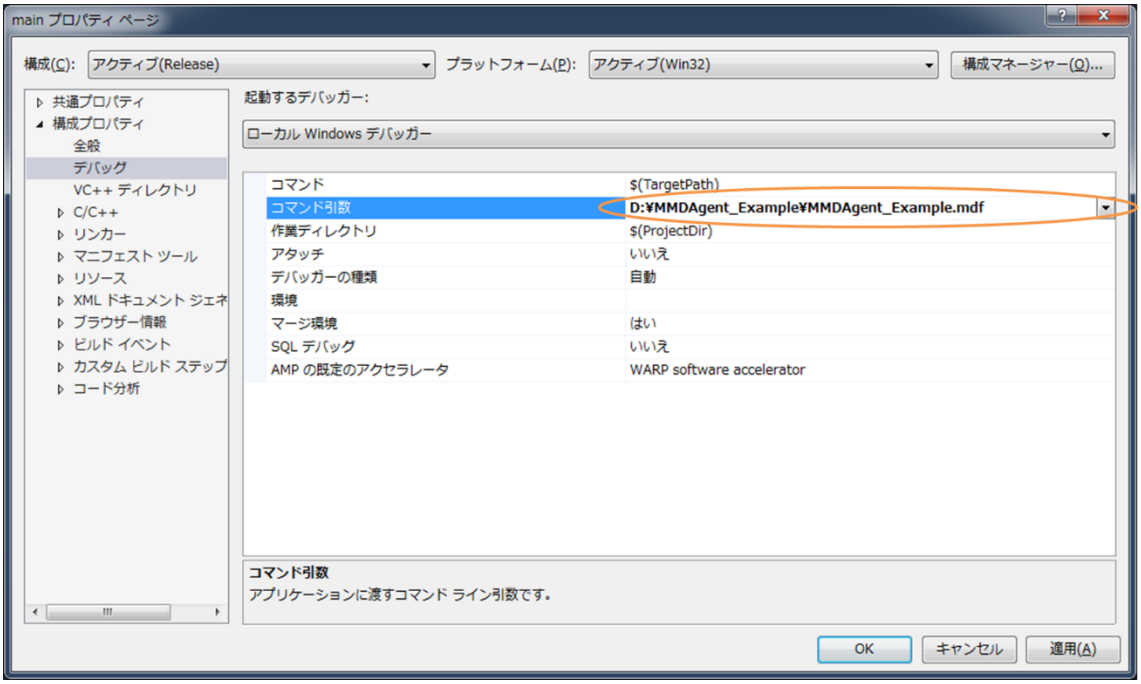




[備考]
公式サイトで配布されている Contents Package を解凍したフォルダに存在する、MMDAgent_Example.mdf のパスを実行時の引数として与えることで、メイちゃんを表示した状態で開発を行うことができる。

▼設定手順
main プロジェクトのプロパティを開き、[構成プロパティ]->[デバッグ] に存在する [コマンド引数] に対し、MMDAgent_Example.mdf のパスを設定する。

▼参考画像



新規プラグインの開発

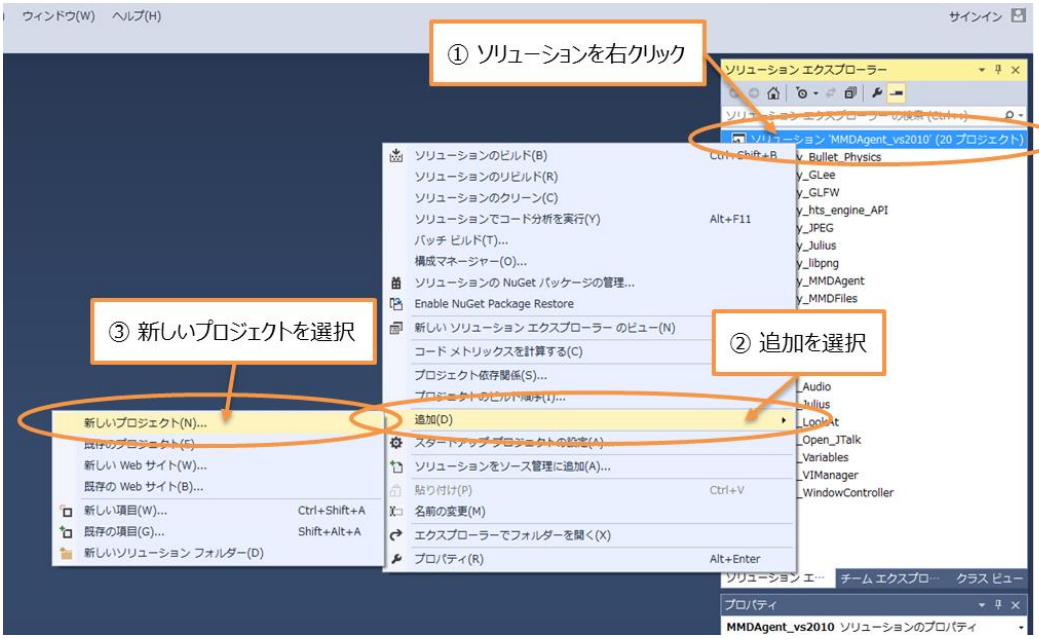
プロジェクトの追加/設定

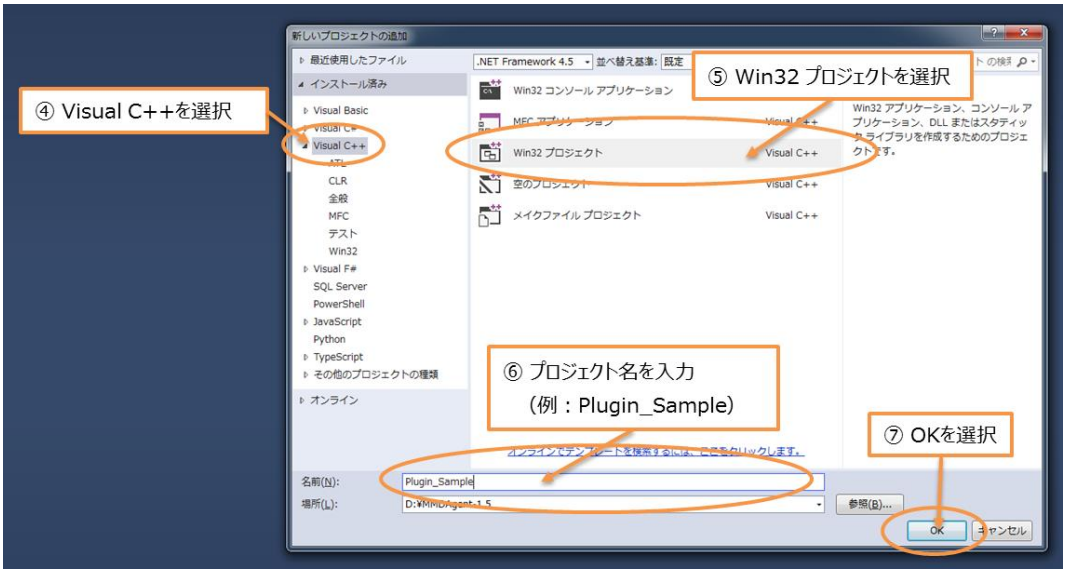
開発用のプロジェクトを追加して新規プラグインを開発できる状態にする。

▼手順

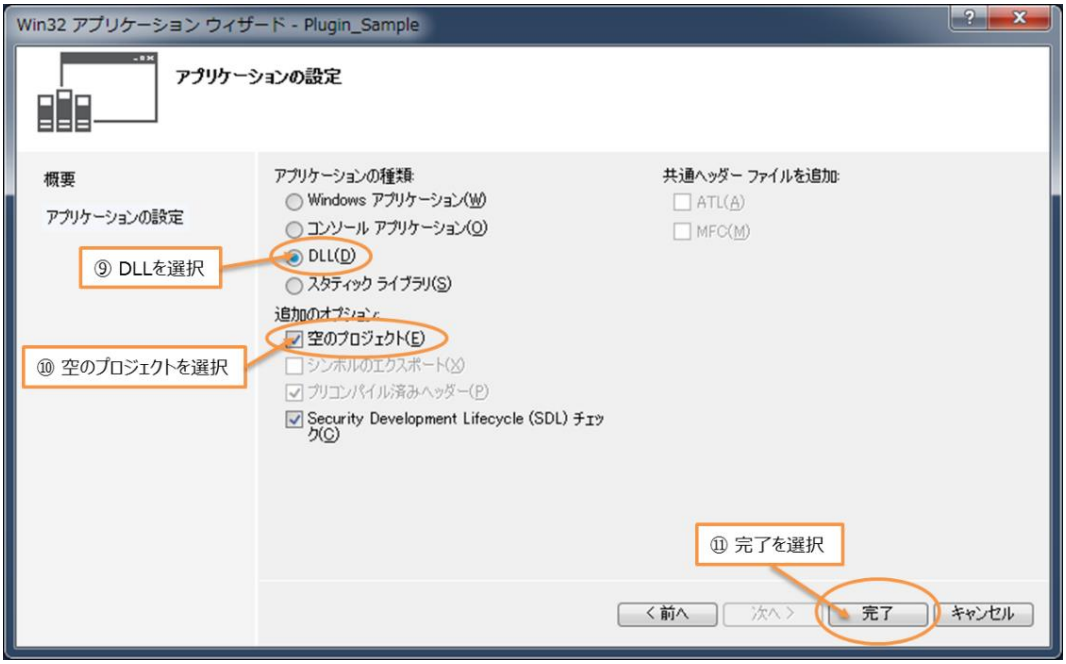
1. プロジェクトの追加

   ソリューションにプロジェクトを追加する。ソリューションエクスプローラーから新しいプロジェクトを追加するとプロジェクトの追加ウィンドウが出現するので、以下の画面を参考に初期設定を行う。



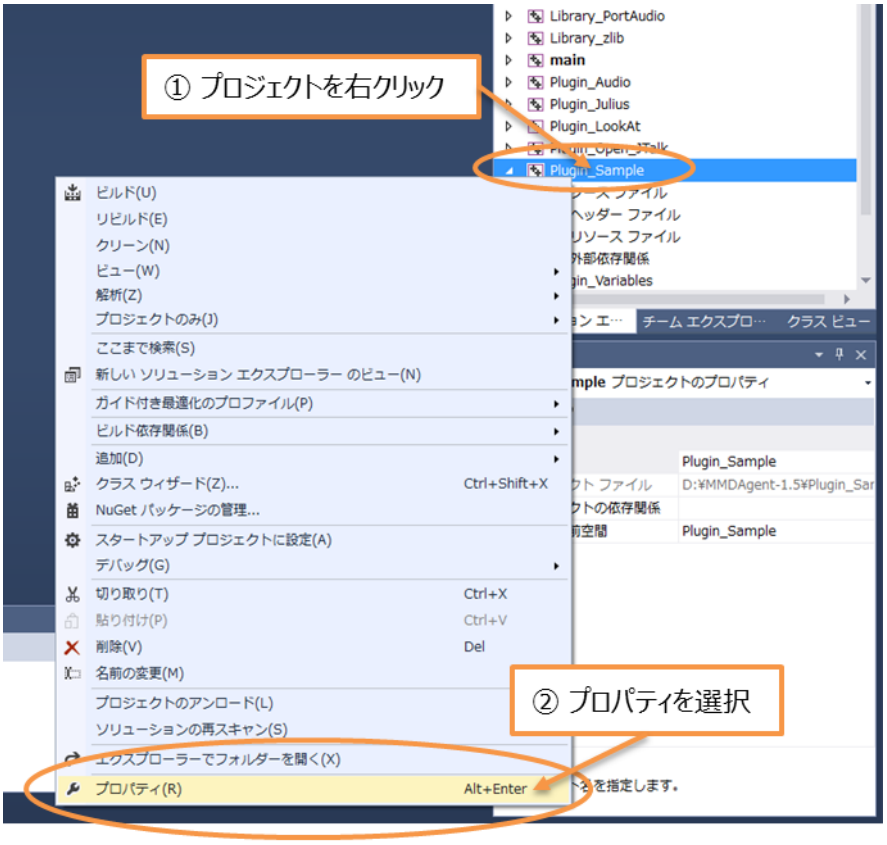



Win32 アプリケーション ウィザード - Plugin_Sample
Win32 アプリケーション ウィザードへようこそ
概要
アプリケーションの設定
現在のプロジェクト設定:
• Windows アプリケーション
現在の設定を有効にするには、完了をクリックしてください。
プロジェクトの作成後、プロジェクトの readme.txt ファイルでプロジェクトの機能と生成ファイルに関する情報を参照してください。
⑧ 次へを選択
< 前へ
次へ >
完了
キャンセル


Win32 アプリケーション ウィザード - Plugin_Sample
アプリケーションの設定
概要
アプリケーションの設定
⑨ DLLを選択
⑩ 空のプロジェクトを選択
アプリケーションの種類:
Windows アプリケーション(W)
コンソール アプリケーション(O)
DLL(D)
スタティック ライブラリ(S)
追加のオプション:
空のプロジェクト(E)
シンボルのエクスポート(X)
プリコンパイル済みヘッダー(P)
Security Development Lifecycle (SDL) チェック(C)
共通ヘッダー ファイルを追加:
ATL(A)
MFC(M)
⑪ 完了を選択
< 前へ
次へ >
完了
キャンセル

2. プロジェクトの設定変更

追加したプロジェクトの設定を変更する。プロパティ ページのウィンドウが出現した後、
後述するⅠ～Ⅴの設定を順番に行う。



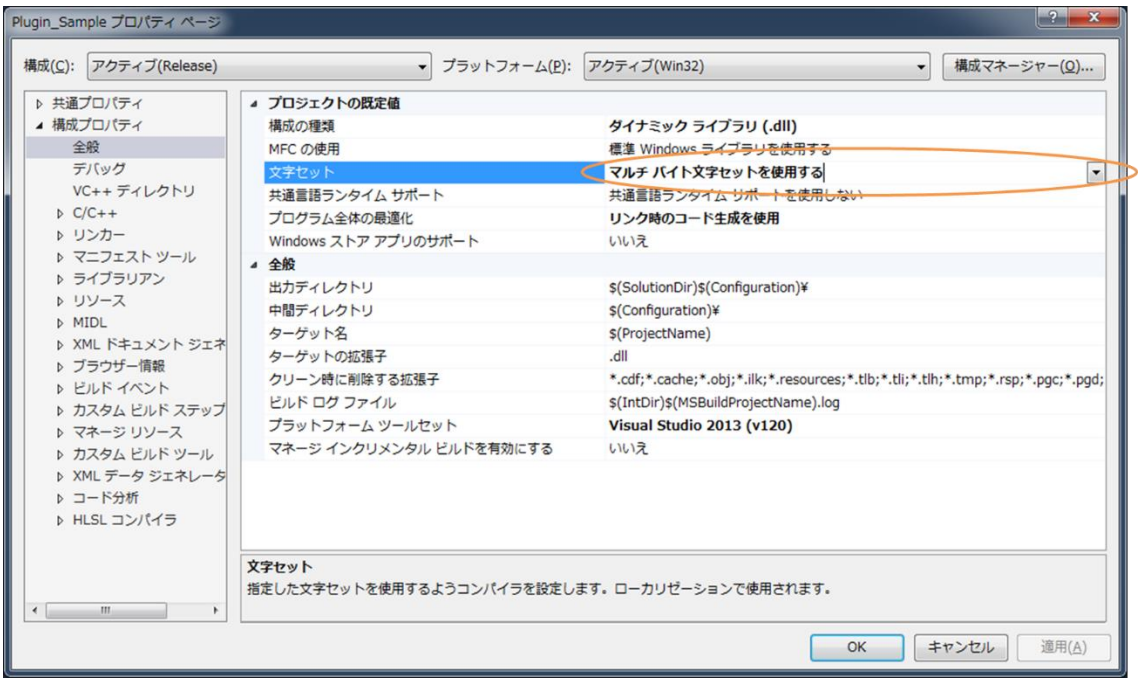
[備考]

Debug 構成で実行したい場合、ソリューション構成を Debug に変更した状態で設定を行う。

I. 文字セットの設定

[構成プロパティ]->[全般] に存在する [文字セット] の設定を、
[マルチバイト文字セットを使用する] に変更する。

▼参考画面



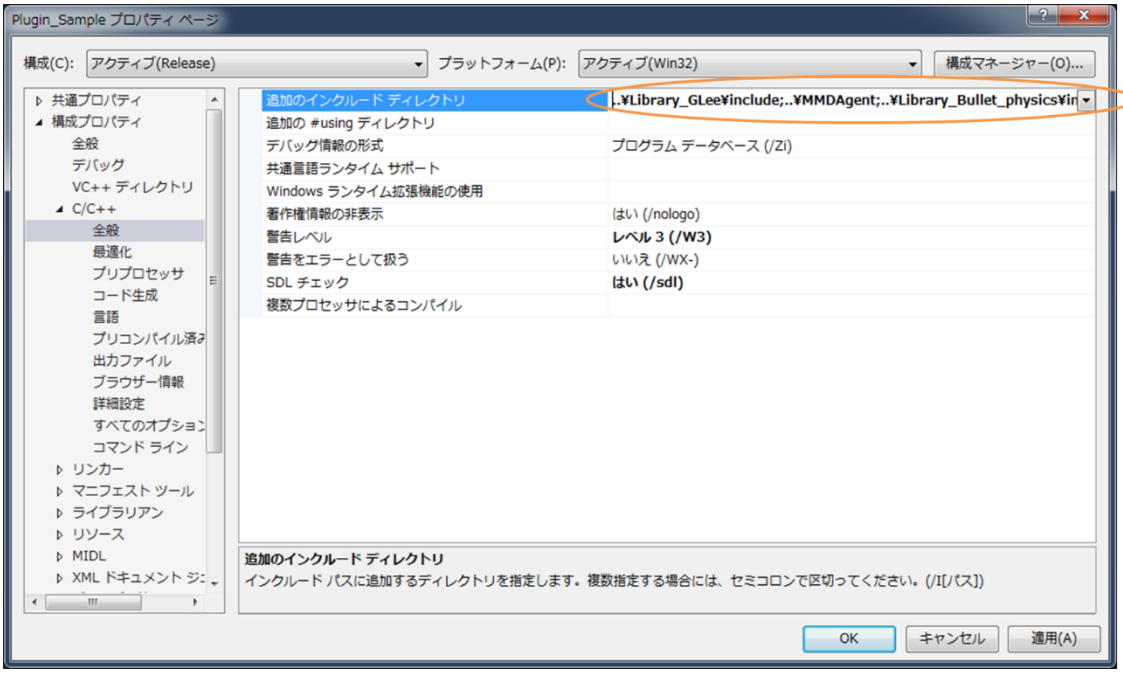
II. 追加のインクルード ディレクトリの設定

[構成プロパティ]->[C/C++]->[全般] に存在する [追加のインクルード ディレクトリ] に対し、以下の設定値を入力する。

▼設定値

```
..¥Library_GLee¥include;..¥Library_Bullet_physics¥include;..¥Library_MMDFiles¥include;..¥Library_Julius¥include;..¥Library_MMDAgent¥include;..¥Library_GLFW¥include;
```

▼参考画面



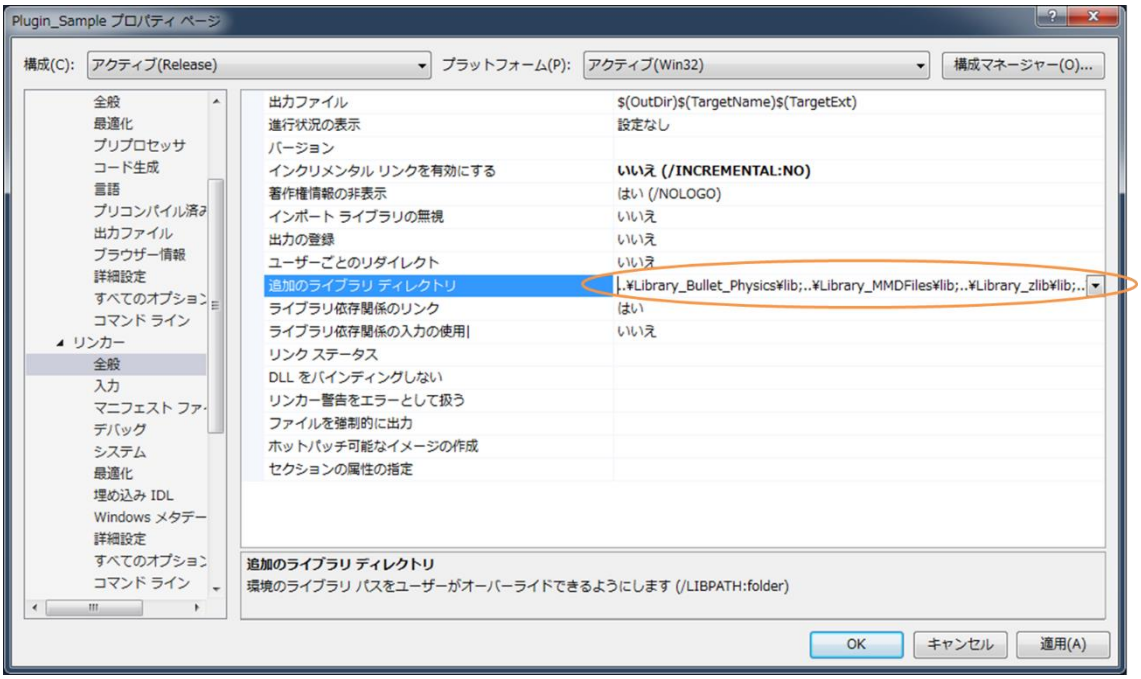
III. 追加のライブラリ ディレクトリの設定

[構成プロパティ]->[リンカー]->[全般] に存在する [追加のライブラリ ディレクトリ] に対し、以下の設定値を入力する。

▼設定値

```
..¥Library_Bullet_Physics¥lib;..¥Library_MMDFiles¥lib;..¥Library_zlib¥lib;..¥Library_libpng¥lib;..¥Library_GLee¥lib;..¥Library_MMDAgent¥lib;..¥Library_JPEG¥lib;..¥Library_GLFW¥lib;..¥Library_FreeType¥lib
```

▼参考画面



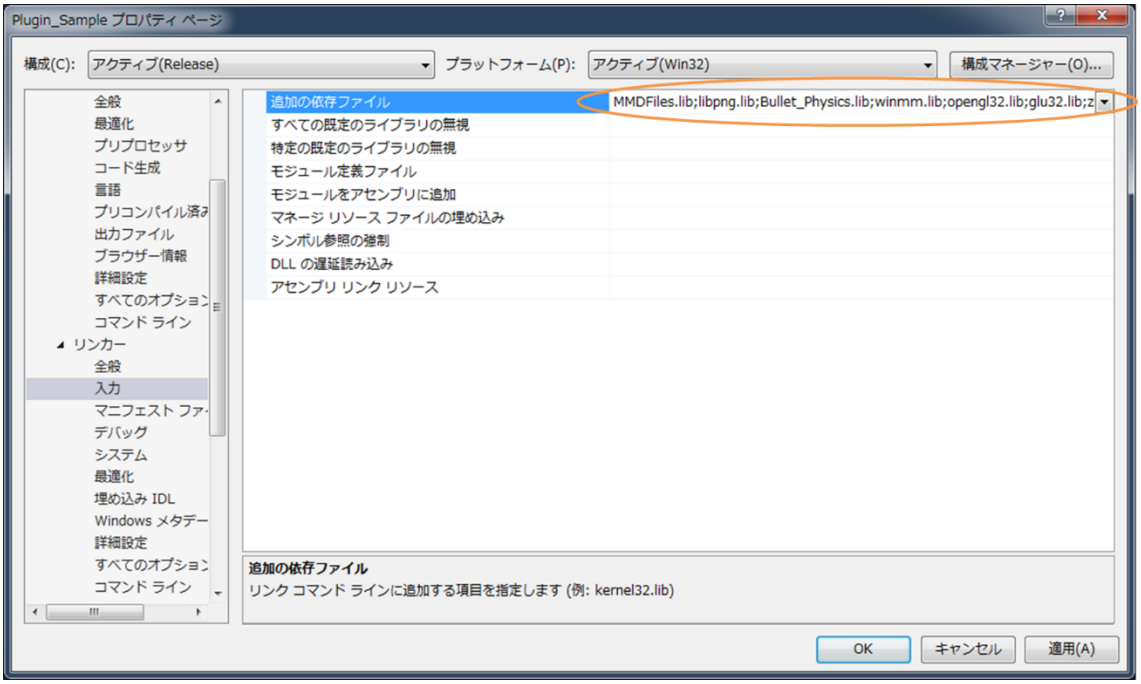
IV. 追加の依存ファイルの設定

[構成プロパティ]->[リンカー]->[入力] に存在する [追加の依存ファイル] に対し、
以下の設定値を入力する。なお、最初から入力されている設定値は上書きして良い。

▼設定値

```
MMDFiles.lib;libpng.lib;Bullet_Physics.lib;winmm.lib;opengl32.lib;
glu32.lib;zlib.lib;GLee.lib;MMDAgent.lib;JPEG.lib;GLFW.lib;FreeTyp
e.lib;%(AdditionalDependencies)
```

▼参考画面



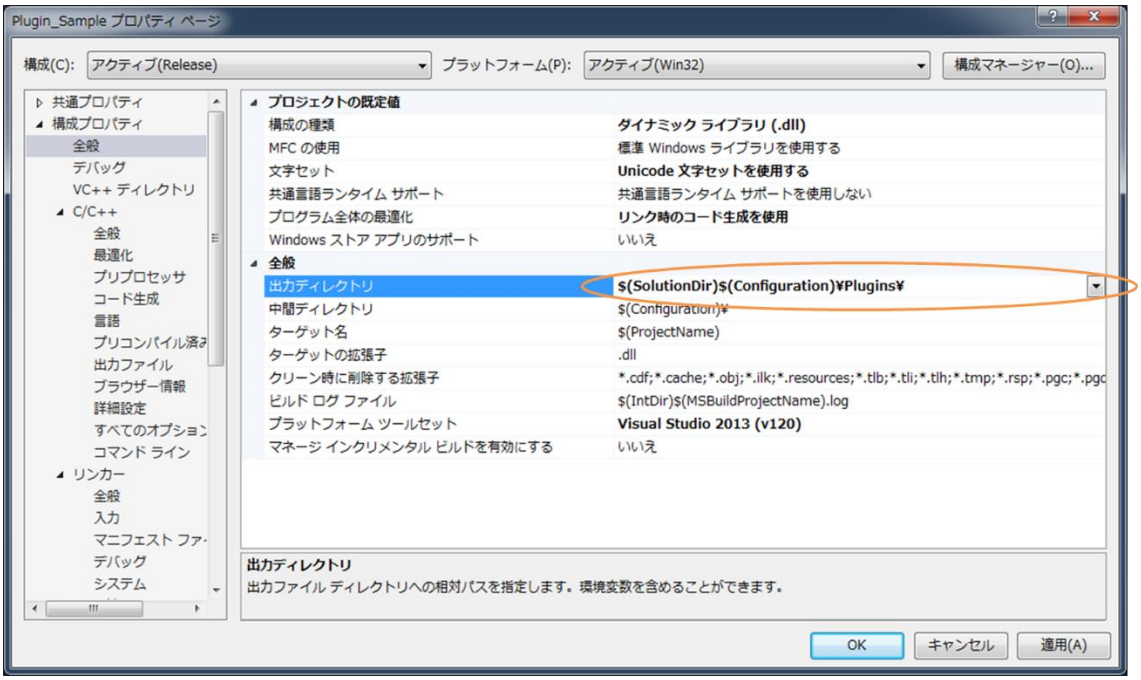
V. 出力ディレクトリの設定

[構成プロパティ]->[全般] に存在する [出力ディレクトリ] に対し、
以下の設定値を入力する。

▼設定値

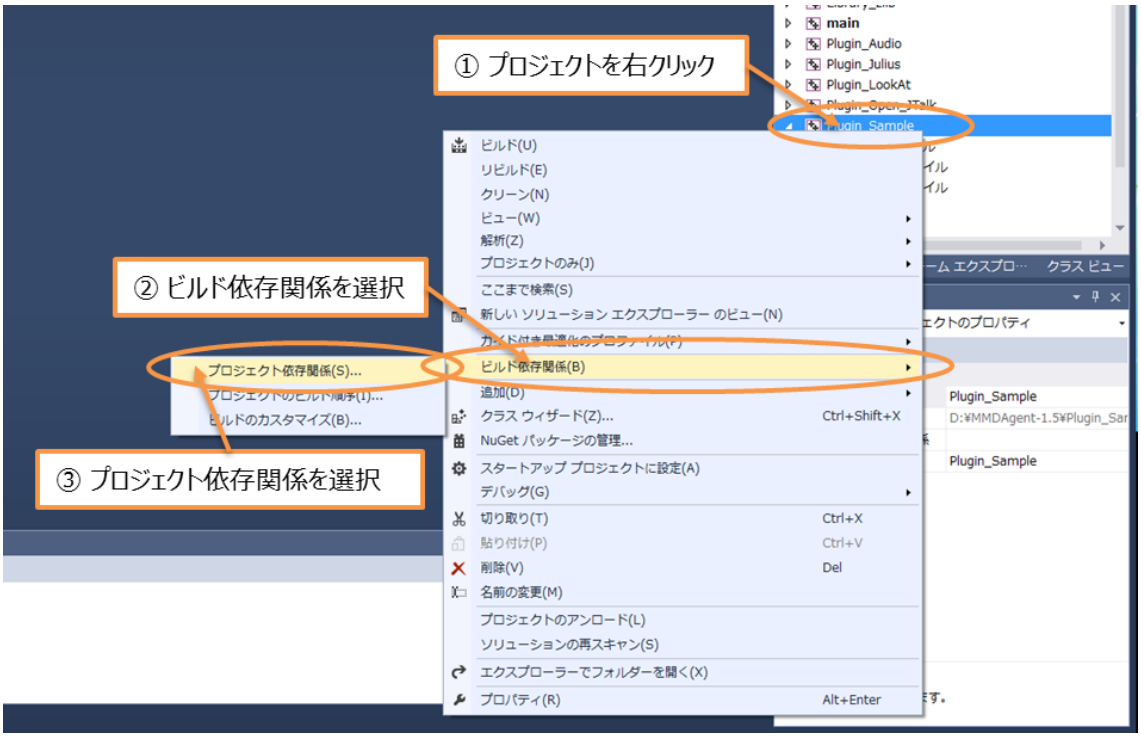
```
$(SolutionDir)$(Configuration)¥Plugins¥
```

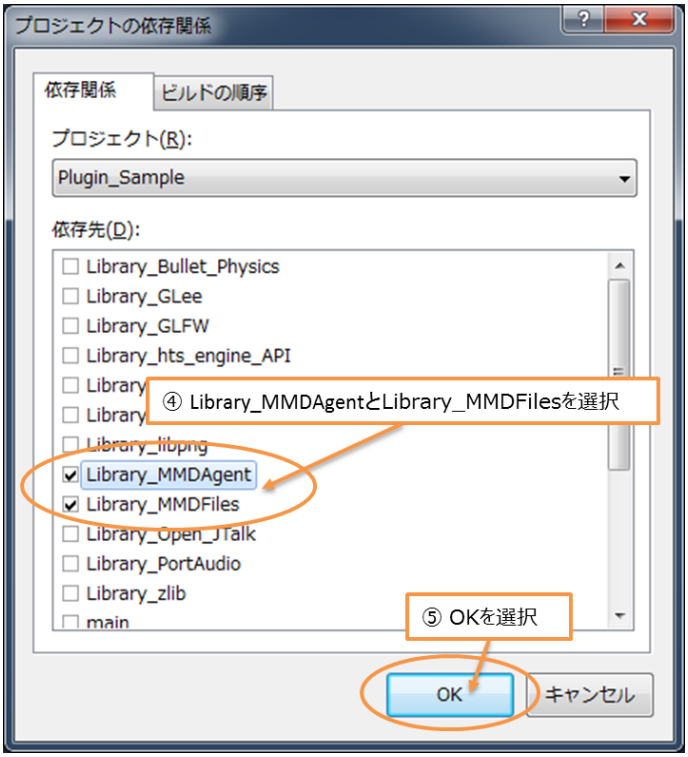
▼参考画面



3. プロジェクトの依存関係を設定

追加したプロジェクトの依存関係を設定する。





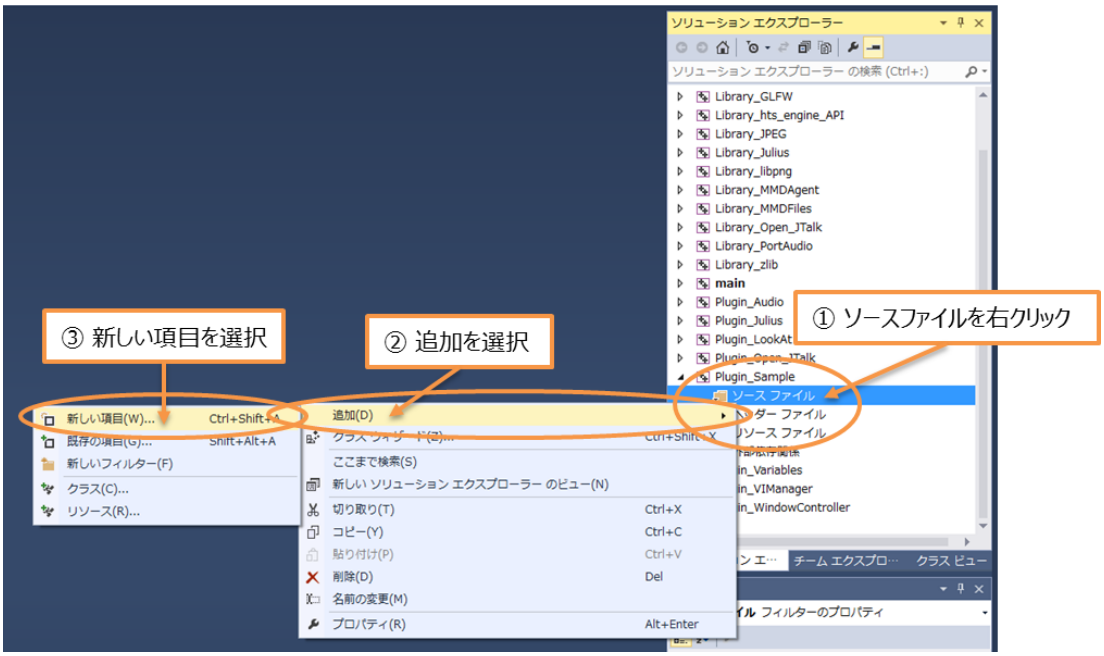
ファイルの作成/ビルド/実行

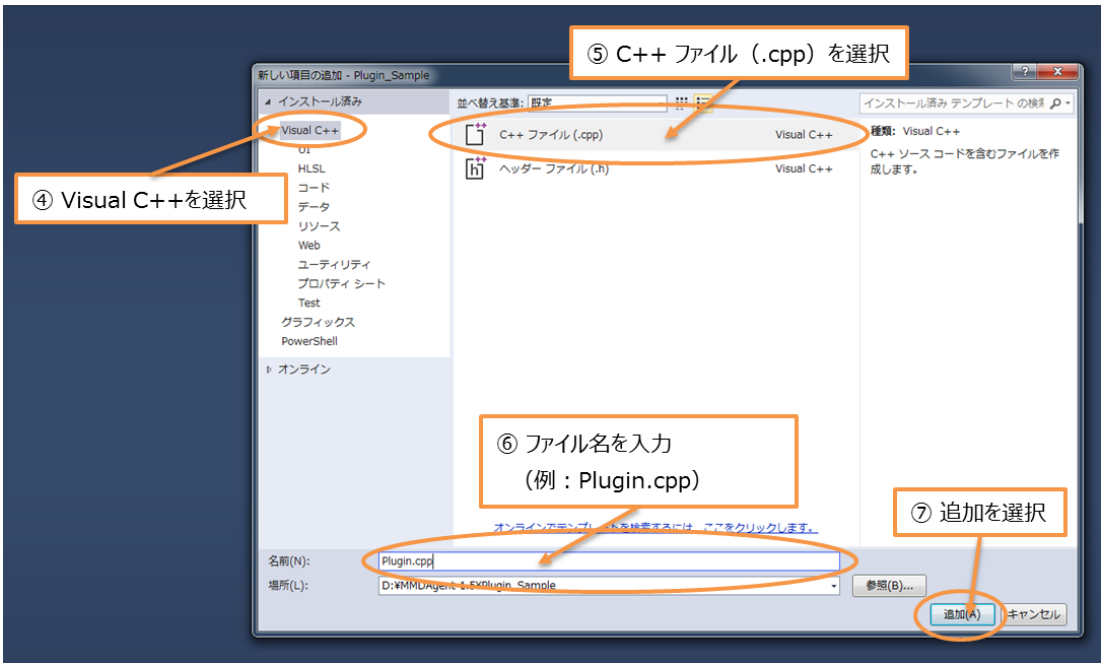
開発用のソースファイルを追加してテスト実行する。

▼手順

1. ソースファイルの追加

   プロジェクトにソースファイルを追加する。





2. テストコードの記述

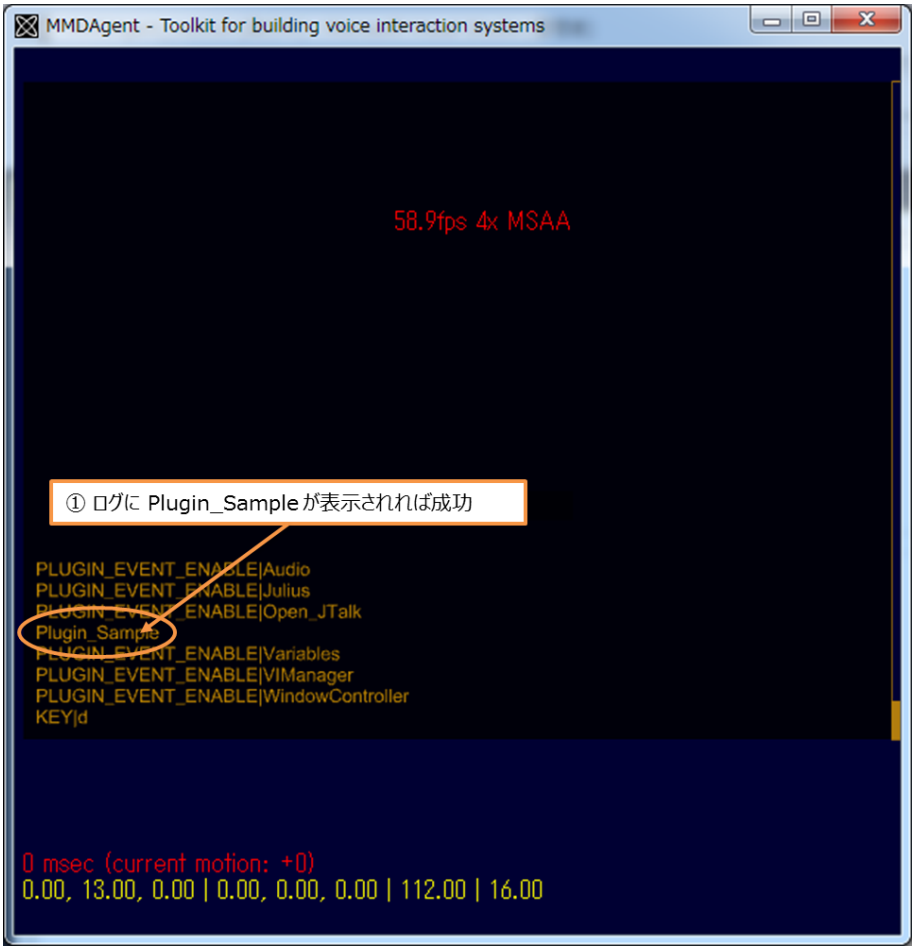
追加したソースファイルに以下のテスト用コードを記述する。

```
1  #ifdef _WIN32
2  #define EXPORT extern "C" __declspec(dllexport)
3  #else
4  #define EXPORT extern "C"
5  #endif
6
7  #include "MMDAgent.h"
8
9  EXPORT void extAppStart(MMDAgent *mmdagent)
10 {
11     // ログ出力：Plugin_Sample
12     mmdagent->sendMessage("Plugin_Sample", "");
13 }
```

3. ビルド/実行

実行環境の構築と同様の手順でビルドと実行を行ない、ログに Plugin_Sample が出力されることを確認する。

▼参考画面



[備考]

プラグイン本体（.dll）は MMDAgent_vs2010.sln と同階層に存在する Release¥Plugins フォルダに生成される。

実装可能な関数群

MMDAgent のプラグインとして実装可能な関数を記す。なお、後述する関数群を実装する場合、ソースファイルの先頭などで、以下の EXPORT 定義および MMDAgent.h の呼び出しが必要となる。

▼必要な記述

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

#include "MMDAgent.h"
```

▼MMDAgent クラスについて

プラグインとして実装可能な関数には、MMDAgent クラスにアクセスするためのポインタが引数として渡される。この引数を利用することで、MMDAgent で内部メッセージを発行するなど、MMDAgent クラスが公開している機能を使用することができる。

以下は内部メッセージを発行するメソッド（sendMessage）の構文と記述例である。その他の公開された機能については、Library_MMDAgent¥include¥MMDAgent.h に記述されている。

・構文

```
void sendMessage(const char *type, const char *format, ...);
```

・説明

内部メッセージを発行するメソッド。

引数

・*type*

内部メッセージの種類。

・*format*（可変長引数）

フォーマット指定子。C 言語標準ライブラリの printf と同様。

戻り値

無し

・記述例

```
mmdagent->sendMessage("Plugin_Sample", "%s", "Arg");
```

extAppStart 関数

▼説明

MMDAgent 起動時に 1 回だけ呼ばれる関数。
プラグインの初期化処理を記述するために使用される。

▼構文

```
EXPORT void extAppStart(MMDAgent *mmdagent) {}
```

[引数]

・*mmdagent*
　MMDAgent の機能を使用するための参照値。

[戻り値]

・*void*
　無し

extAppEnd 関数

▼説明

MMDAgent 終了時（ウィンドウが閉じる時）に 1 回だけ呼ばれる関数。
プラグインの終了処理を記述するために使用される。

▼構文

```
EXPORT void extAppEnd(MMDAgent *mmdagent) {}
```

[引数]

・*mmdagent*
　MMDAgent の機能を使用するための参照値。

[戻り値]

・*void*
　無し

extProcMessage 関数

▼説明

MMDAgent で内部メッセージ（EventMessage または CommandMessage）が発行されると呼び出される関数。発行されたメッセージに対する処理を記述するために使用される。

▼構文

```
EXPORT void extProcMessage(MMDAgent *mmdagent, const char *type, const char *args) {}
```

[引数]

・*mmdagent*

MMDAgent の機能を使用するための参照値。

・*type*

発行された内部メッセージの種類。

・*args*

発行された内部メッセージの内容。

[戻り値]

・*void*

無し

extUpdate 関数

▼説明

MMDAgent が更新処理を行う度に呼び出される関数。
シーンの時間進行に合わせた更新処理を記述されるために使用される。

▼構文

```
EXPORT void extUpdate(MMDAgent *mmdagent, double deltaFrame) {}
```

[引数]

・*mmdagent*
　MMDAgent の機能を使用するための参照値。

・*deltaFrame*
　前回の処理から経過した差分フレーム。単位は 1/30 秒。

[戻り値]

・*void*
　無し

extRender 関数

▼説明

MMDAgent が描画処理を行う度に呼び出される関数。
垂直同期ごとの描画処理を記述するために使用される。

▼構文

```
EXPORT void extRender(MMDAgent *mmdagent) {}
```

[引数]

・*mmdagent*
　MMDAgent の機能を使用するための参照値。

[戻り値]

・*void*
　無し

簡易実装例

プラグインの簡易実装例として、MMDAgent のメッセージをファイルに出力するプラグインを以下に記す。

▼Plugin_LogMessage.cpp

```
/* 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;
```

```
33
34      /* ログファイルを開く（追記） */
35      ofs.open(fileName, std::ios::out | std::ios::app);
36      if (!ofs) { /* もしファイルを開くのに失敗していたら */
37          /* ファイルを開くのに失敗したメッセージを発行する */
38          mmdagent->sendMessage(MMDAGENT_EVENT_FILEOPEN, "%s can not be opened!", fileName);
39      }
40      else { /* ファイルを開くのに成功していたら */
41          /* ファイルを開くのに成功したメッセージを発行する */
42          mmdagent->sendMessage(MMDAGENT_EVENT_FILEOPEN, "%s can be opened", fileName);
43
44          /* 現在時刻を取得 */
45          t = time(0);
46          char buf[32];
47          ctime_s(buf, sizeof(buf), &t);
48
49          /* 時刻を書き込む */
50          ofs << buf;
51          ofs << "[[Start]]" << std::endl;
52      }
53  }
54
55  /* extProcMessage: process message */
56  EXPORT void extProcMessage(MMDAgent *mmdagent, const char *type, const char *args)
57  {
58      if (enable == true) {
59          /* 出力ストリームに出力する（1行） */
60          /* 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） */
61          /* の時のみログファイルに書き込むようになる */
62          // if (MMDAgent_strequal(type, "RECOG_EVENT_STOP"))
63          {
64              ofs << type << "|" << args << std::endl;
65          }
66      }
67  }
68
```

```
69  /* extAppEnd: stop and free thread */
70  EXPORT void extAppEnd(MMDAgent *mmdagent)
71  {
72      /* MMDAgent のログの区切りを示す */
73      ofs << "[[End]]" << std::endl;
74      ofs << std::endl;
75
76      /* MMDAgent 終了時にログファイルを閉じる */
77      ofs.close();
78      mmdagent->sendMessage(MMDAGENT_EVENT_FILECLOSE, "%s was closed", LOGFILENAME);
79  }
80
81  /* execUpdate: run when motion is updated */
82  EXPORT void extUpdate(MMDAgent *mmdagent, double deltaFrame)
83  {
84  }
85
86  /* execRender: run when scene is rendered */
87  EXPORT void extRender(MMDAgent *mmdagent)
88  {
89  }
90
```

▼出力されるログファイルについて
プラグインが正常動作すると、MessageLog.txt がカレントディレクトリ（通常は MMDAgent.exe と同じフォルダ）に出力される。

テンプレート（雛形）

プラグインが使用できる関数群を空実装した雛形を以下に記す。

▼Plugin_Template.cpp

```cpp
#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) {}

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

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

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

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

```

3. 開発環境/実行環境の構築（Android 版）

概要

公開されている MMDAgent のソースコードを、Android 上で実行できるようにするための手順をまとめた項目である。なお、開発マシンの OS は Windows を対象としている。

本項目は以下の環境を基に記述するが、他の環境（Windows 8 以前や、別のバージョンの Android Studio 等）でも開発を行うことは可能である。その場合、この仕様書に記述されている内容を、環境に合わせて読み替える必要がある。
また、必要となる JDK のバージョンは、使用する Android Studio により変化するため、Android 開発者サイト（https://developer.android.com）の「Android Studio と SDK Tools のダウンロード」ページのシステム要件を確認すること。

項目	対象
開発環境 OS	Windows 10 64bit
実行環境 OS	Android 5.0.2（4.0 以上を推奨）
開発用ソフトウェア	Android Studio 2.2.3
JDK のバージョン	Java SE Development Kit 8u111
Android SDK のバージョン	25.0.2
Android NDK のバージョン	13.1.3345770

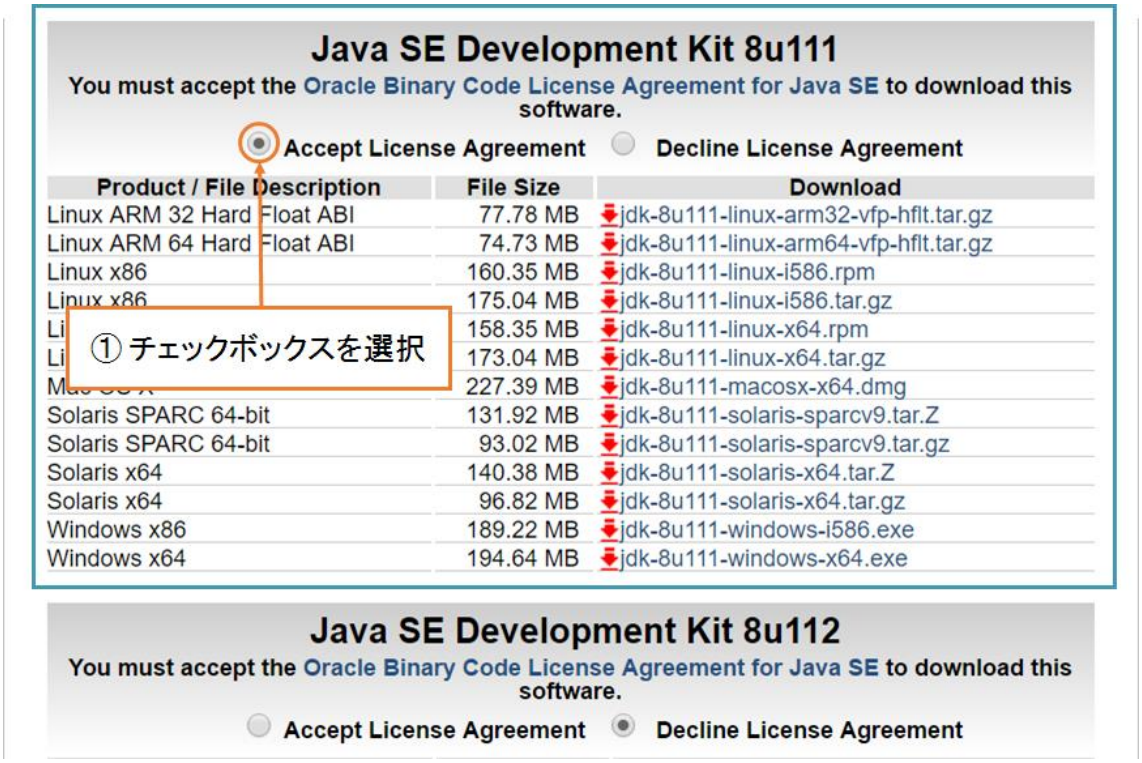
[備考]
実行環境として使用する Android 端末は、開発者向けオプションを有効化する必要がある。

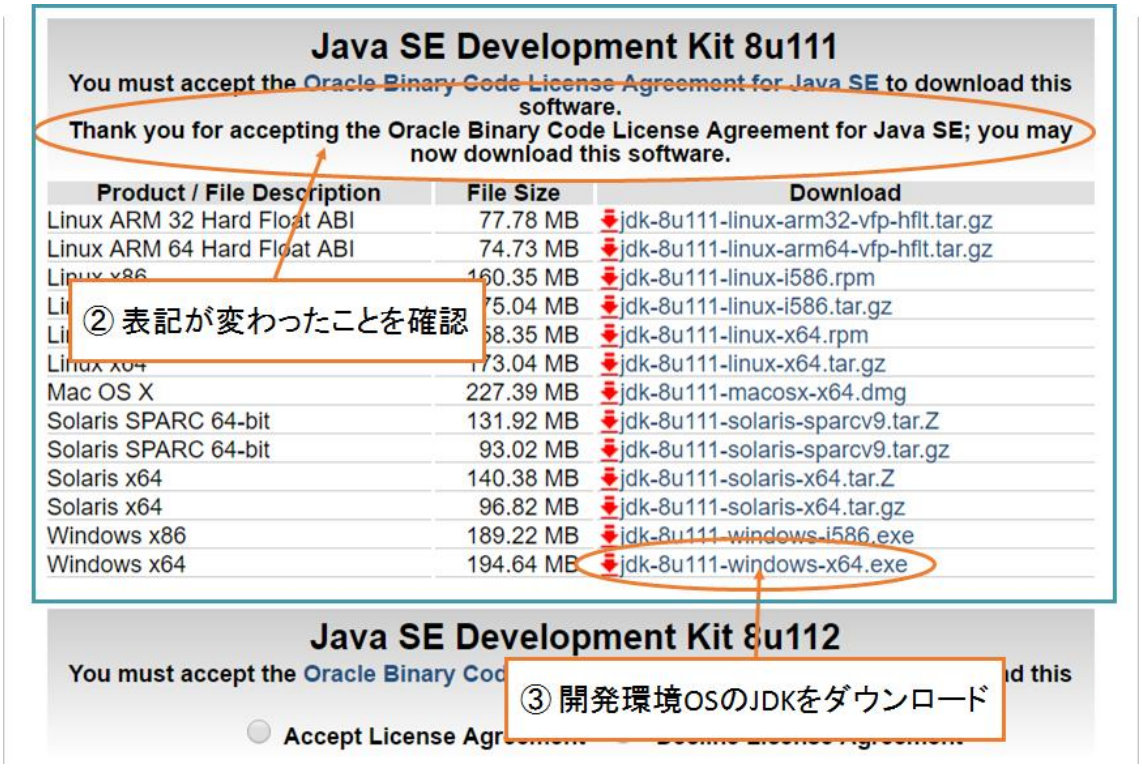
開発環境の構築

Java SE Development Kit 8 のダウンロード

公式サイトから Java SE Development Kit 8 のインストーラーをダウンロードする。
http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

▼手順





[備考]

Java SE Development Kit は JDK、Java Runtime Environment は JRE と呼ばれる。

Java SE Development Kit 8 のインストール

ダウンロードしたインストーラーを実行して Java SE Development Kit 8 をインストールする。また、インストールの途中で JRE のインストール画面が出現するので、画面の指示に従い JRE もインストールする。

▼手順（JDK インストール）

※インストール先のパスを任意に変更することができるが、後述する設定に影響するため推奨しない。

▼手順（JRE インストール、自動的にウィンドウが出現）

環境変数（JAVA_HOME）の設定

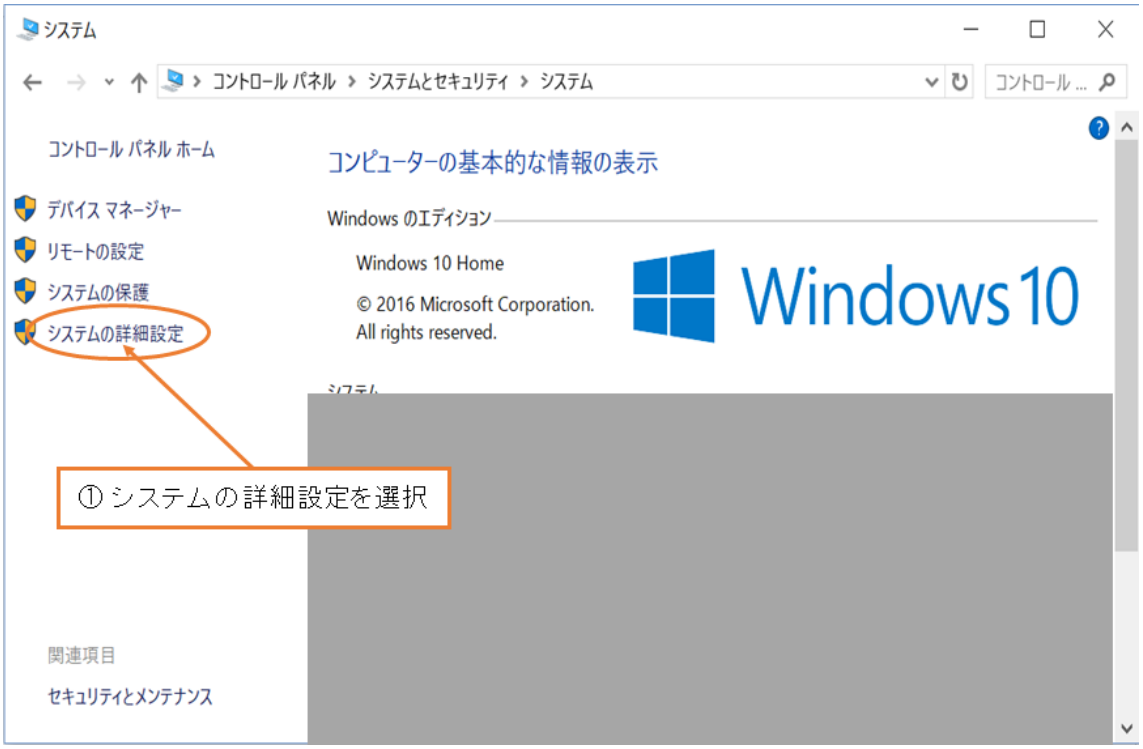
インストールした JDK のパスをシステム環境変数に登録する。

なお、登録する環境変数は以下の通りである。

▼登録する環境変数

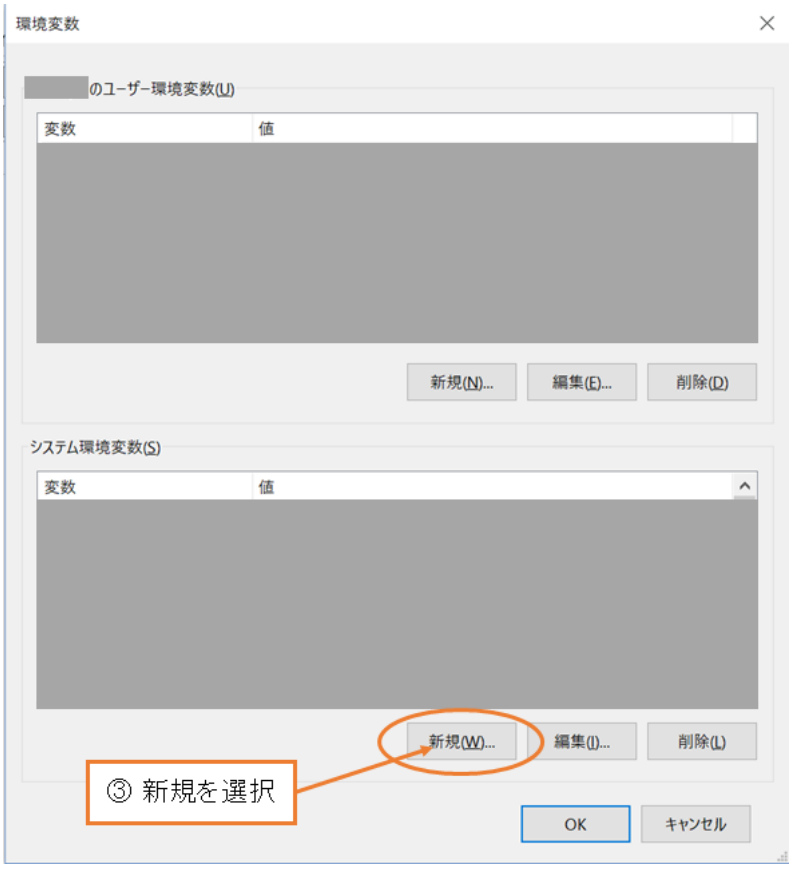
項目	対象
変数名	JAVA_HOME
変数値	C:¥Program Files¥Java¥jdk1.8.0_111

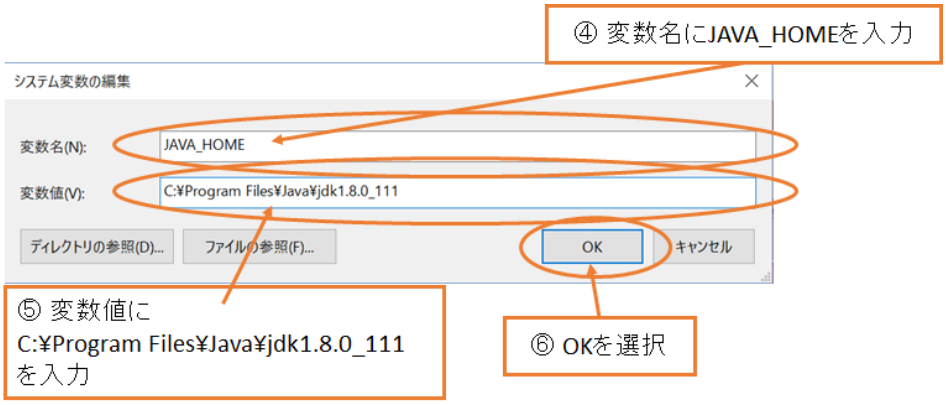
▼手順

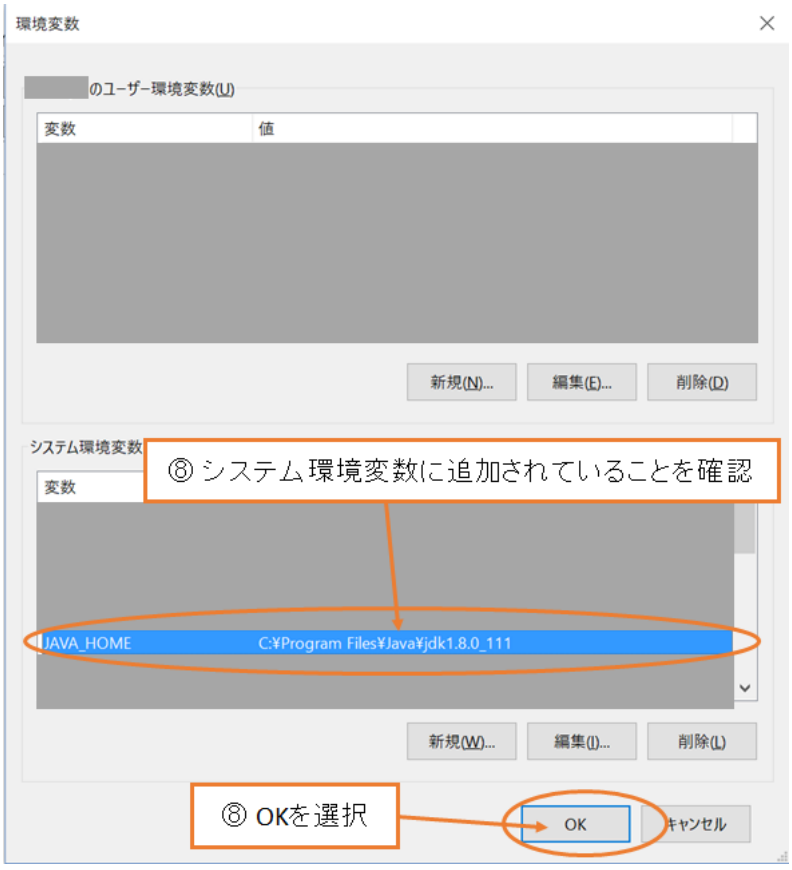


※上記画面は [コントロールパネル]->[システムとセキュリティ]->[システム] で表示することができる。


システムのプロパティ
コンピューター名
ハードウェア
詳細設定
システムの保護
リモート
Administrator としてログオンしない場合は、これらのほとんどは変更できません。
パフォーマンス
視覚効果、プロセッサのスケジュール、メモリ使用、および仮想メモリ
設定(S)...
ユーザー プロファイル
サインインに関連したデスクトップ設定
設定(E)...
② 環境変数を選択
起動と回復
システム起動、システム障害、およびデバッグ情報
設定(T)
環境変数(N)...
OK
キャンセル
適用(A)


環境変数
のユーザー環境変数(U)
変数
値
新規(N)...
編集(E)...
削除(D)
システム環境変数(S)
変数
値
新規(W)...
編集(I)...
削除(L)
③ 新規を選択
OK
キャンセル


④ 変数名にJAVA_HOMEを入力
システム変数の編集
変数名(N):
JAVA_HOME
変数値(V):
C:¥Program Files¥Java¥jdk1.8.0_111
ディレクトリの参照(D)...
ファイルの参照(F)...
OK
キャンセル
⑤ 変数値に
C:¥Program Files¥Java¥jdk1.8.0_111
を入力
⑥ OKを選択


環境変数
のユーザー環境変数(U)
変数
値
新規(N)...
編集(E)...
削除(D)
システム環境変数
⑧ システム環境変数に追加されていることを確認
変数
JAVA_HOME
C:¥Program Files¥Java¥jdk1.8.0_111
新規(W)...
編集(I)...
削除(L)
⑧ OKを選択
OK
キャンセル

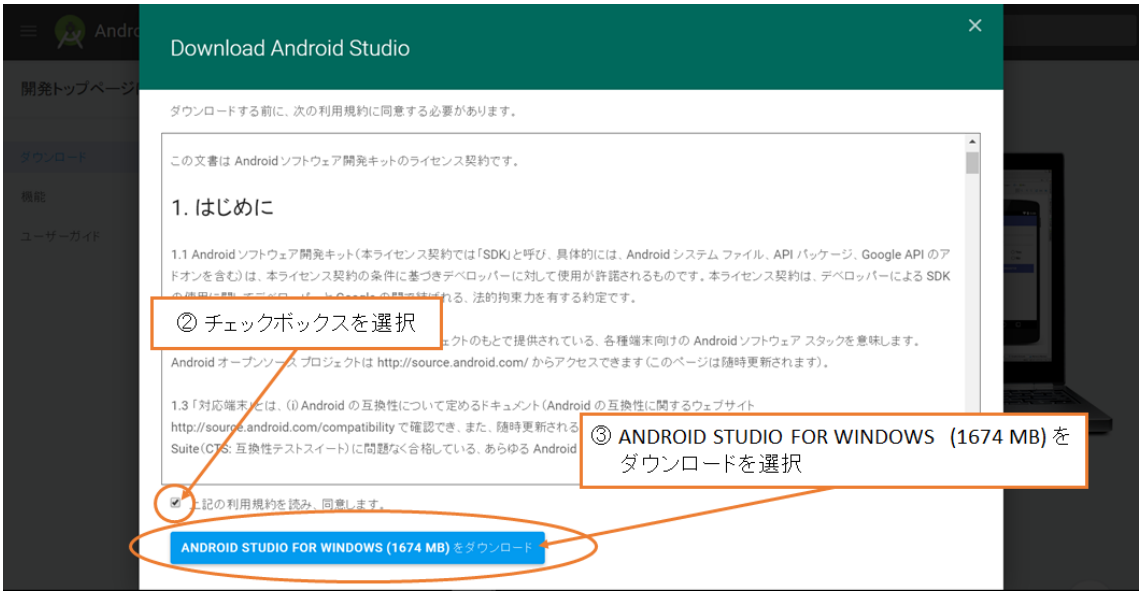

システムのプロパティ
コンピューター名
ハードウェア
詳細設定
システムの保護
リモート
Administrator としてログオンしない場合は、これらのほとんどは変更できません。
パフォーマンス
視覚効果、プロセッサのスケジュール、メモリ使用、および仮想メモリ
設定(S)...
ユーザー プロファイル
サインインに関連したデスクトップ設定
設定(E)...
起動と回復
システム起動、システム障害、およびデバッグ情報
設定(T)
⑨ OKを選択
環境変数(N)...
OK
キャンセル
適用(A)

Android Studio のダウンロード

公式サイトから Android Studio をダウンロードする。

https://developer.android.com/studio/index.html

▼手順



[備考]

公式サイトで公開されている Android Studio は常に最新版となっているが、

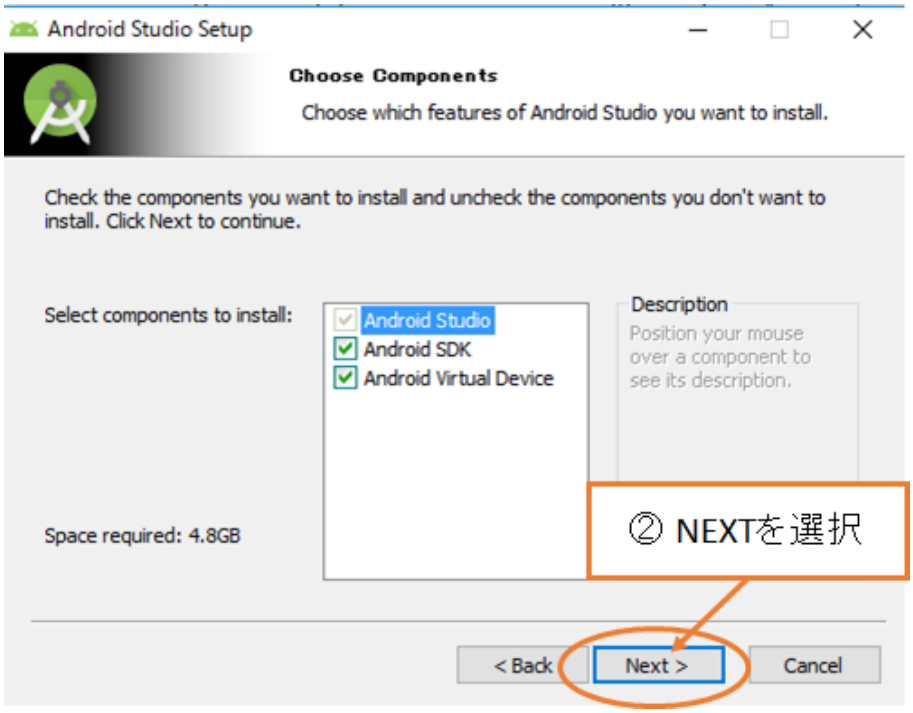
以下のサイトから過去バージョンを入手することができる。

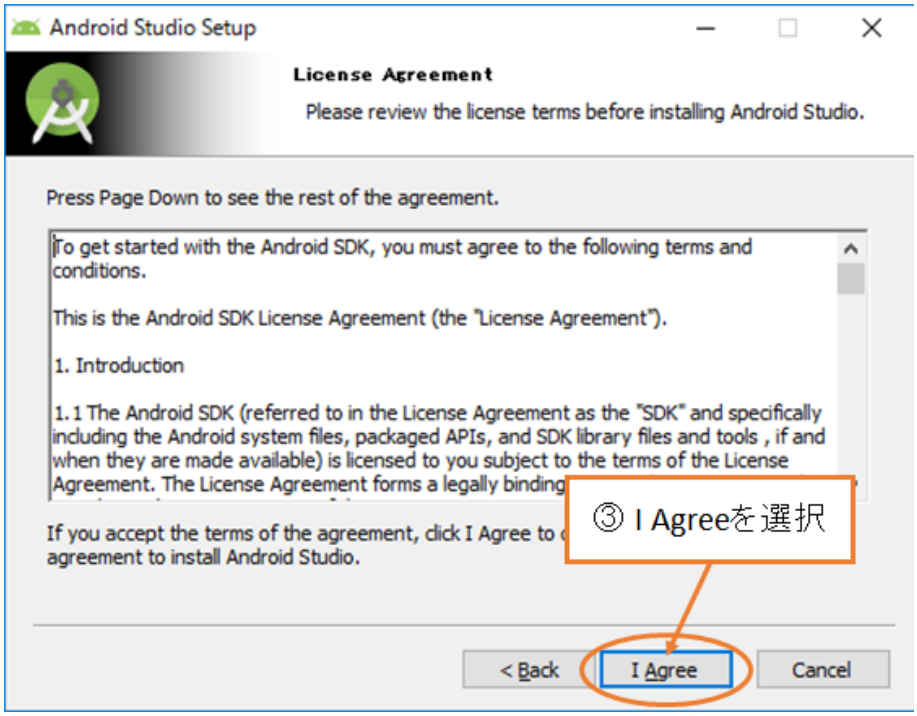
http://tools.android.com/download/studio/canary

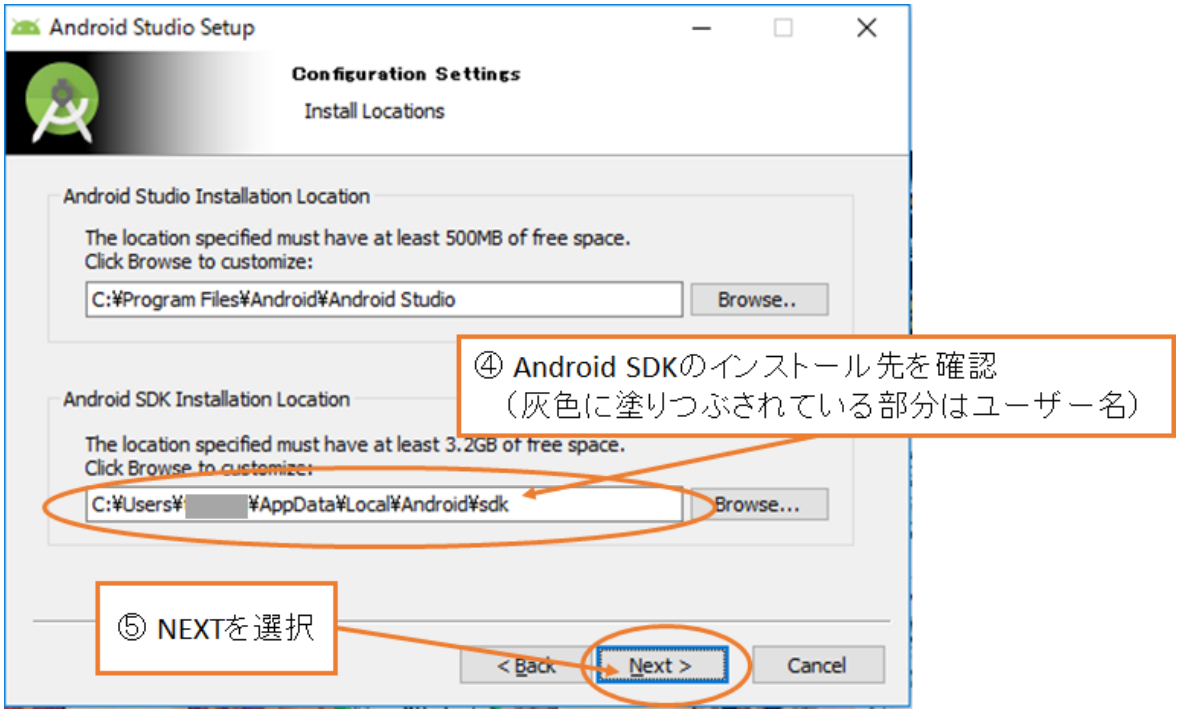
Android Studio のインストール

ダウンロードしたインストーラーを実行して Android Studio をインストールする。

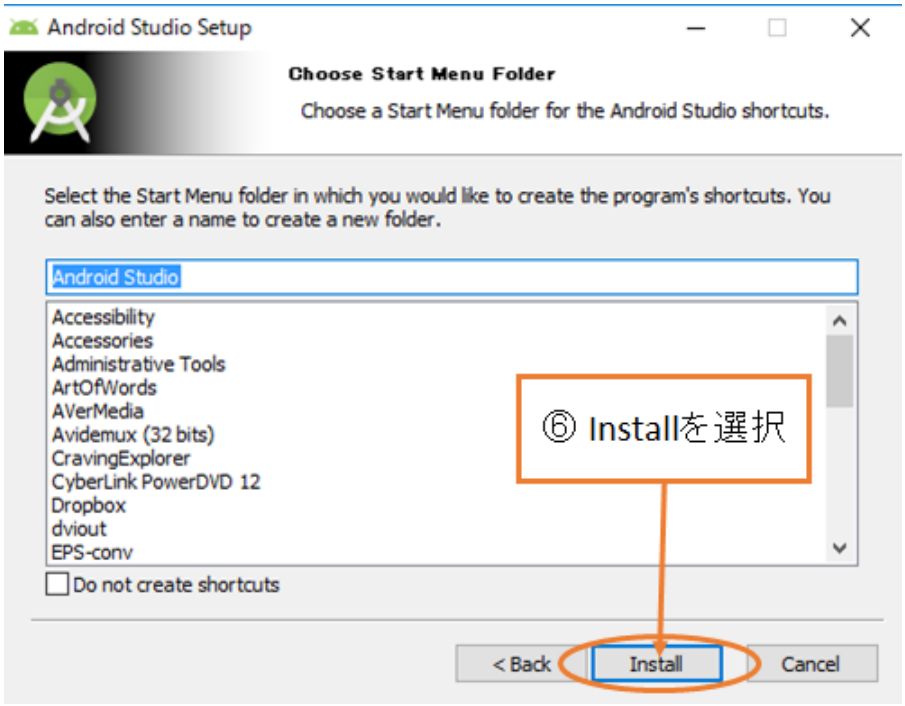
▼手順

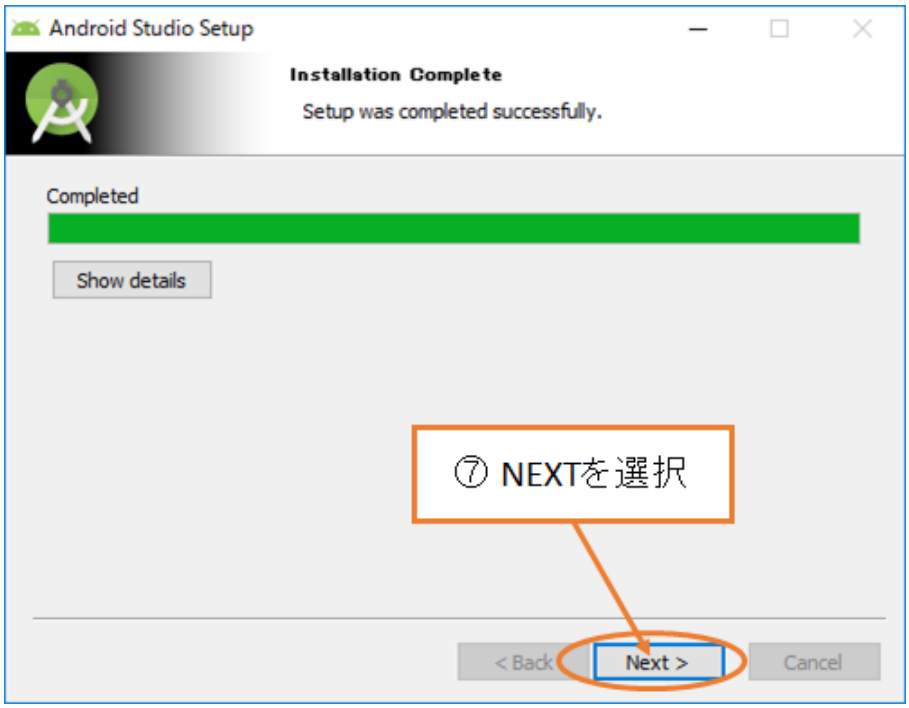




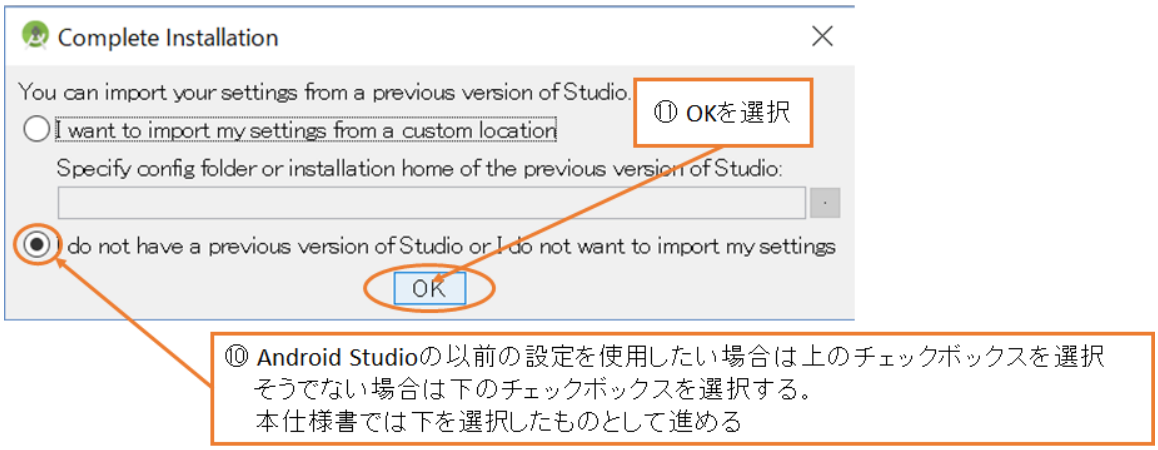


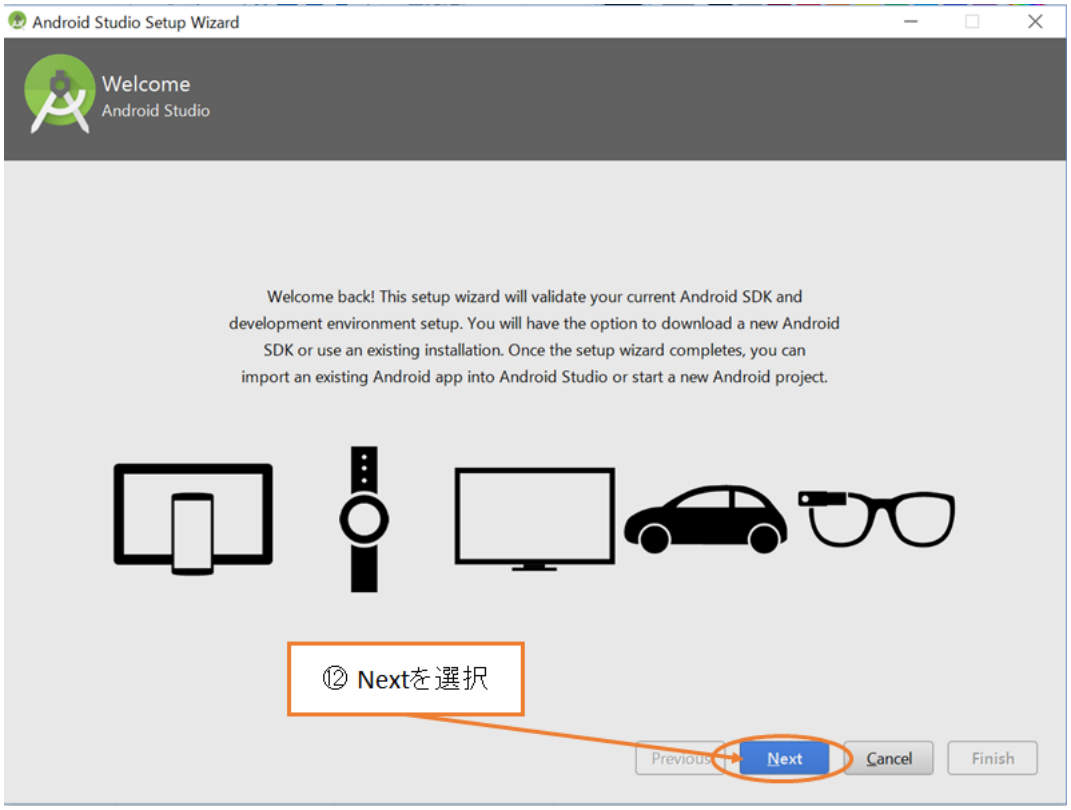
※インストール先のパスを任意に変更することができるが、後述する設定に影響するため推奨しない。

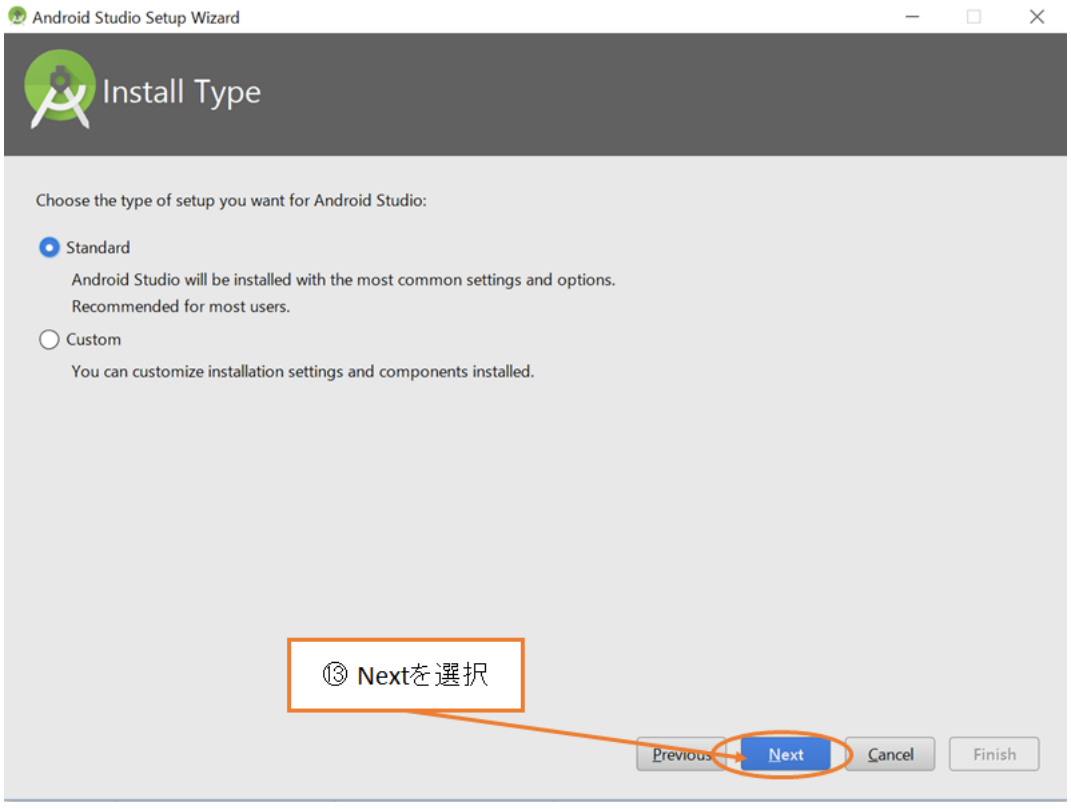

Android Studio Setup
Choose Start Menu Folder
Choose a Start Menu folder for the Android Studio shortcuts.
Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.
Android Studio
Accessibility
Accessories
Administrative Tools
ArtOfWords
AVerMedia
Avidemux (32 bits)
CravingExplorer
CyberLink PowerDVD 12
Dropbox
dviout
EPS-conv
Do not create shortcuts
⑥ Installを選択
< Back
Install
Cancel

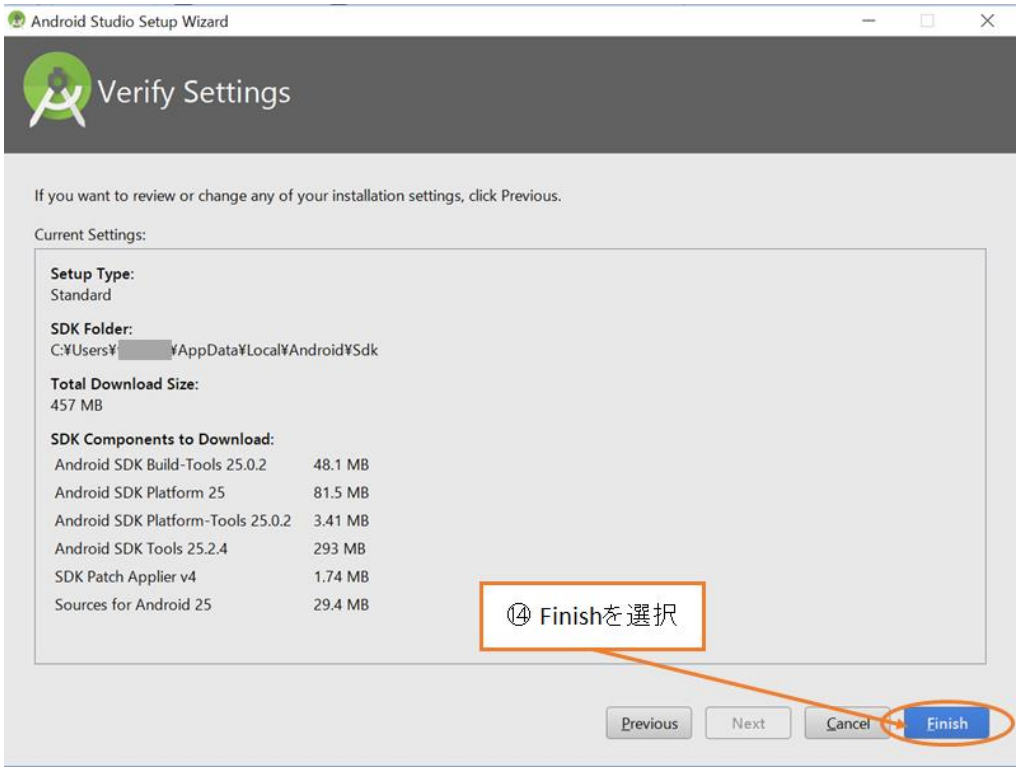

Android Studio Setup
Installation Complete
Setup was completed successfully.
Completed
Show details
⑦ NEXTを選択
< Back
Next >
Cancel

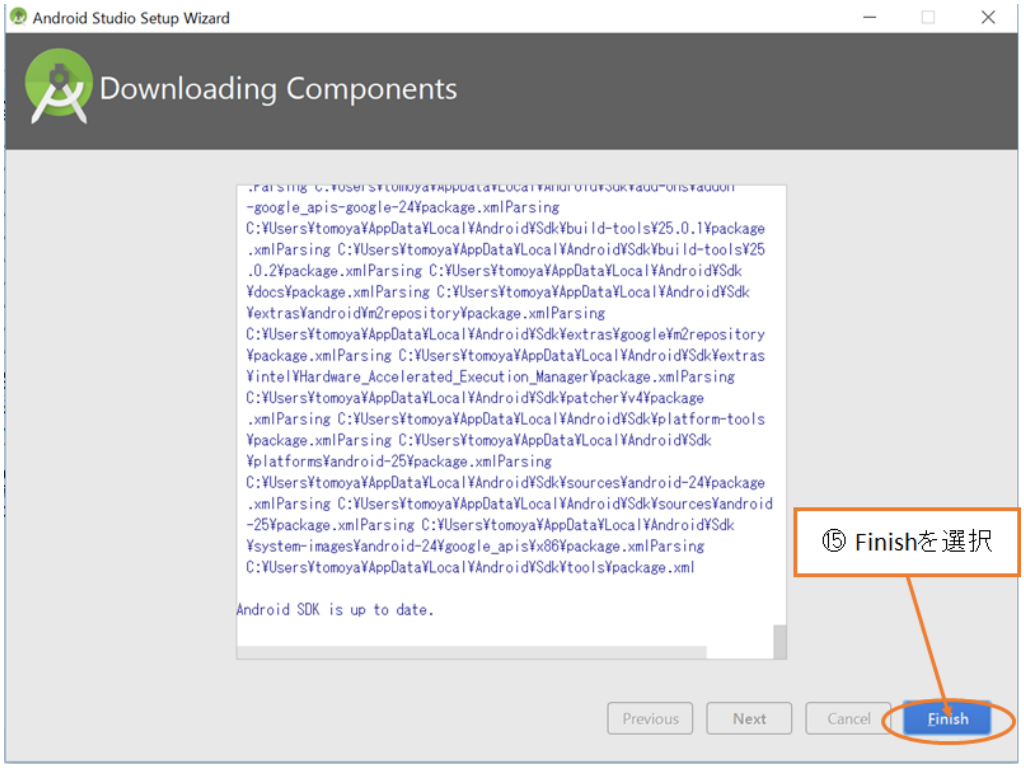

Android Studio Setup
Completing Android Studio Setup
Android Studio has been installed on your computer.
Click Finish to close Setup.
Start Android Studio
⑧ チェックボックスを選択
Android Studio
⑨ Finishを選択
< Back
Finish
Cancel


Complete Installation
You can import your settings from a previous version of Studio.
I want to import my settings from a custom location
Specify config folder or installation home of the previous version of Studio:
I do not have a previous version of Studio or I do not want to import my settings
OK
① OKを選択
⑩ Android Studioの以前の設定を使用したい場合は上のチェックボックスを選択
そうでない場合は下のチェックボックスを選択する。
本仕様書では下を選択したものとして進める


Android Studio Setup Wizard
Welcome
Android Studio
Welcome back! This setup wizard will validate your current Android SDK and development environment setup. You will have the option to download a new Android SDK or use an existing installation. Once the setup wizard completes, you can import an existing Android app into Android Studio or start a new Android project.
⑫ Nextを選択
Previous
Next
Cancel
Finish


Android Studio Setup Wizard
Install Type
Choose the type of setup you want for Android Studio:
Standard
Android Studio will be installed with the most common settings and options.
Recommended for most users.
Custom
You can customize installation settings and components installed.
⑬ Nextを選択
Previous
Next
Cancel
Finish





[備考]

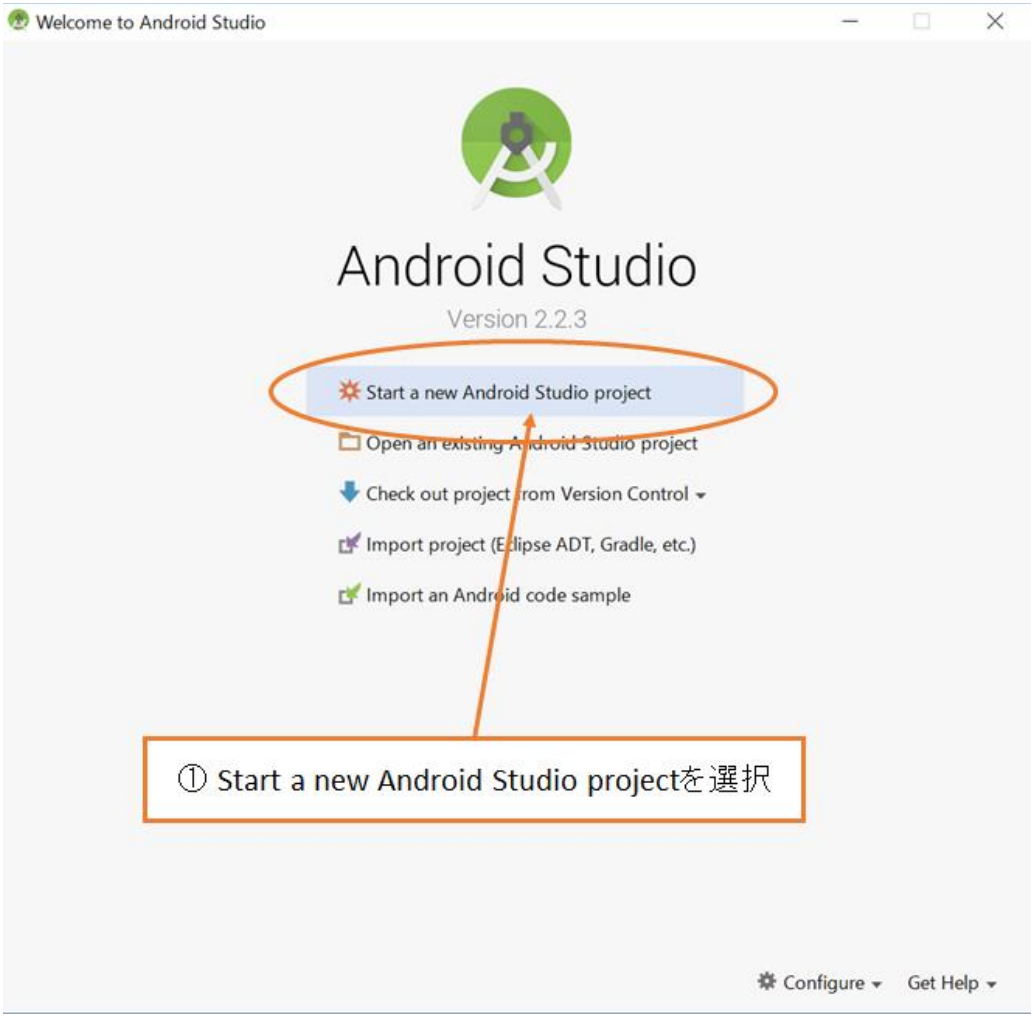
Android の開発には Android SDK も必要となるが、フル版の Android Studio であれば、インストーラーに Android SDK が内包されているため、個別にインストールする必要は無い。

実行環境の構築

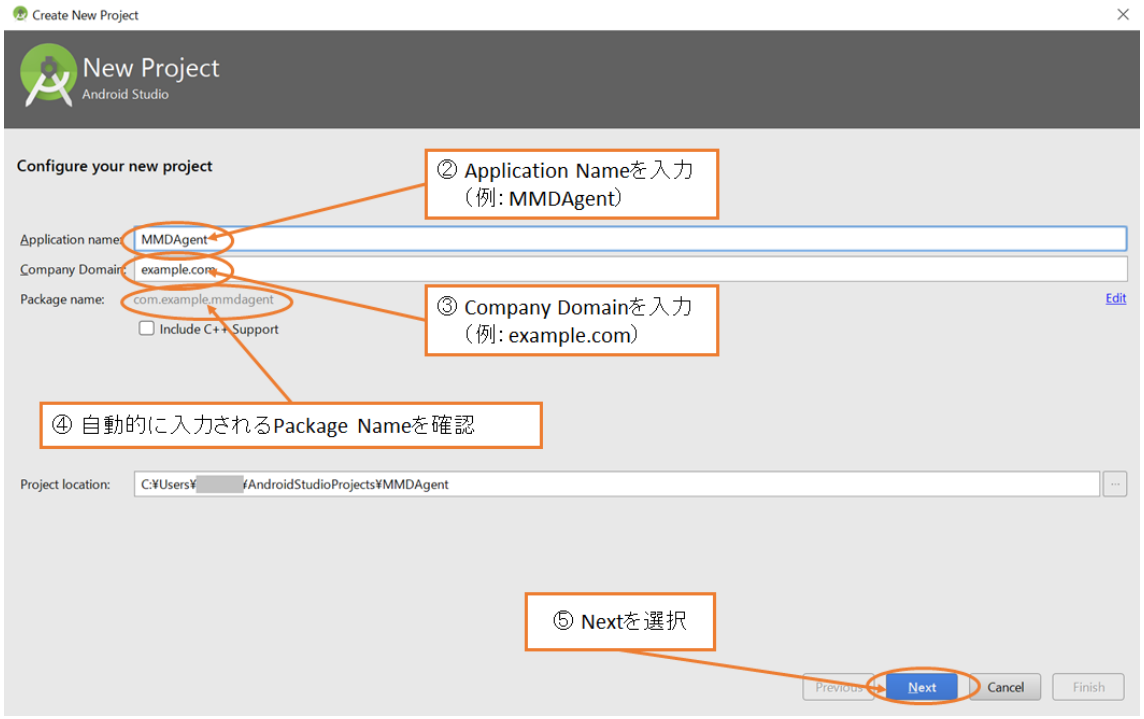
プロジェクトの新規作成

開発用のプロジェクトを新規作成する。

▼手順

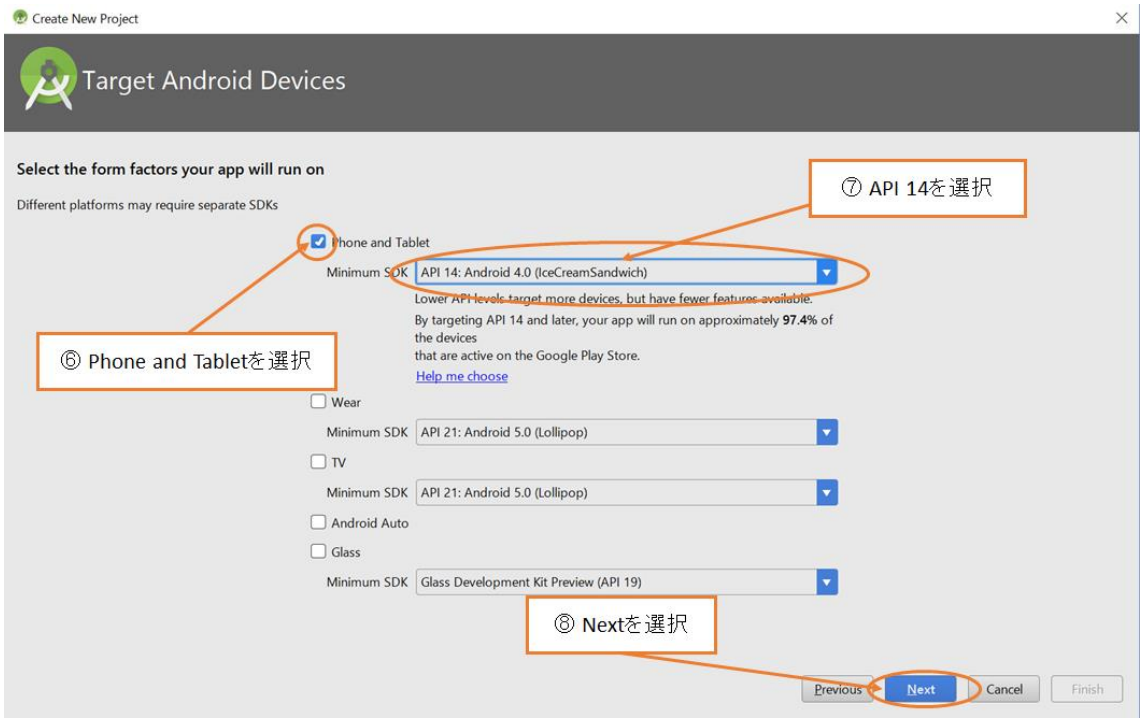


※上記画面は Android Studio の起動時に自動的に表示される。

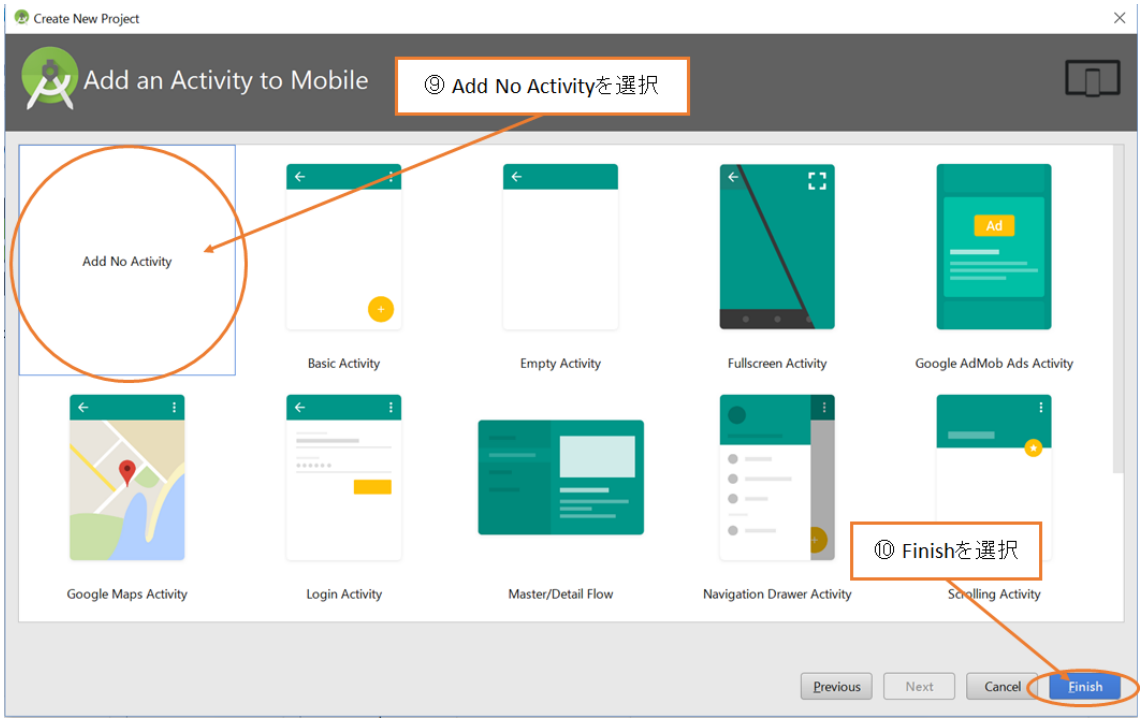


※Package name はアプリの識別 ID となるため、既存アプリと被らない名前を設定する。

※後述する設定で自動入力された Package name を使用する。

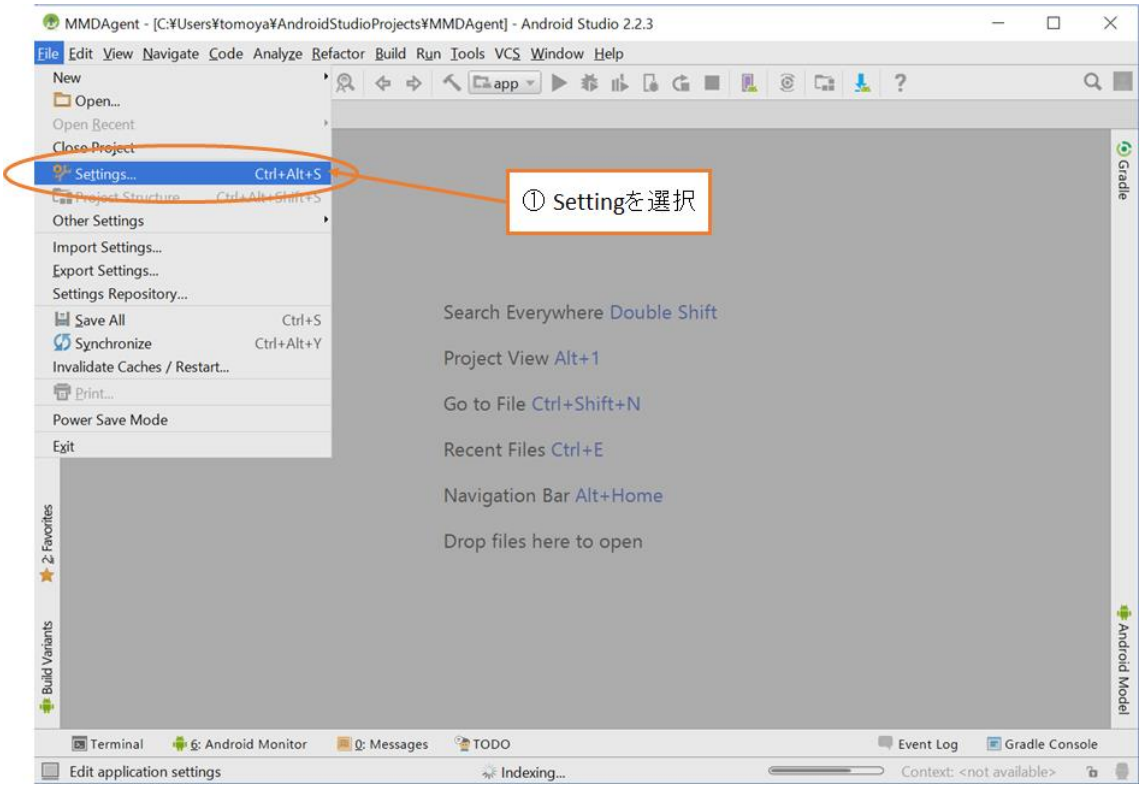


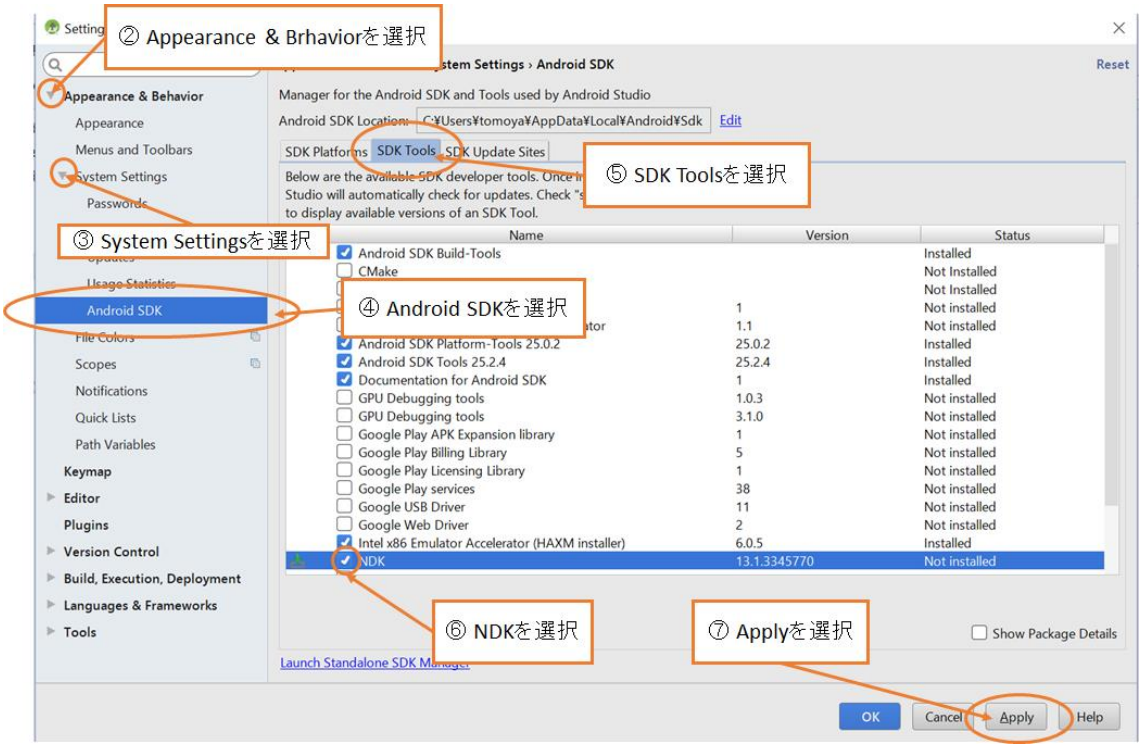
※API 14 未満のコンパイルおよび動作は保証されていない。

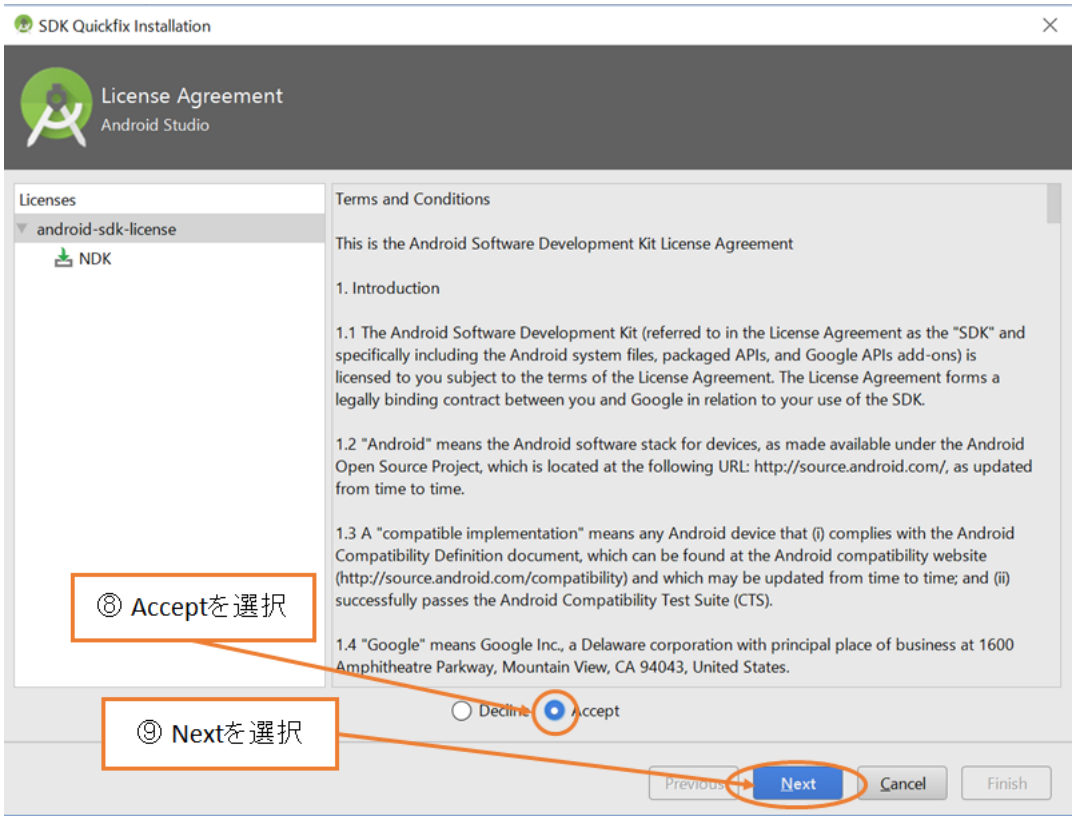

Create New Project
Add an Activity to Mobile
⑨ Add No Activityを選択
Add No Activity
Basic Activity
Empty Activity
Fullscreen Activity
Google AdMob Ads Activity
Google Maps Activity
Login Activity
Master/Detail Flow
Navigation Drawer Activity
Scrolling Activity
⑩ Finishを選択
Previous
Next
Cancel
Finish

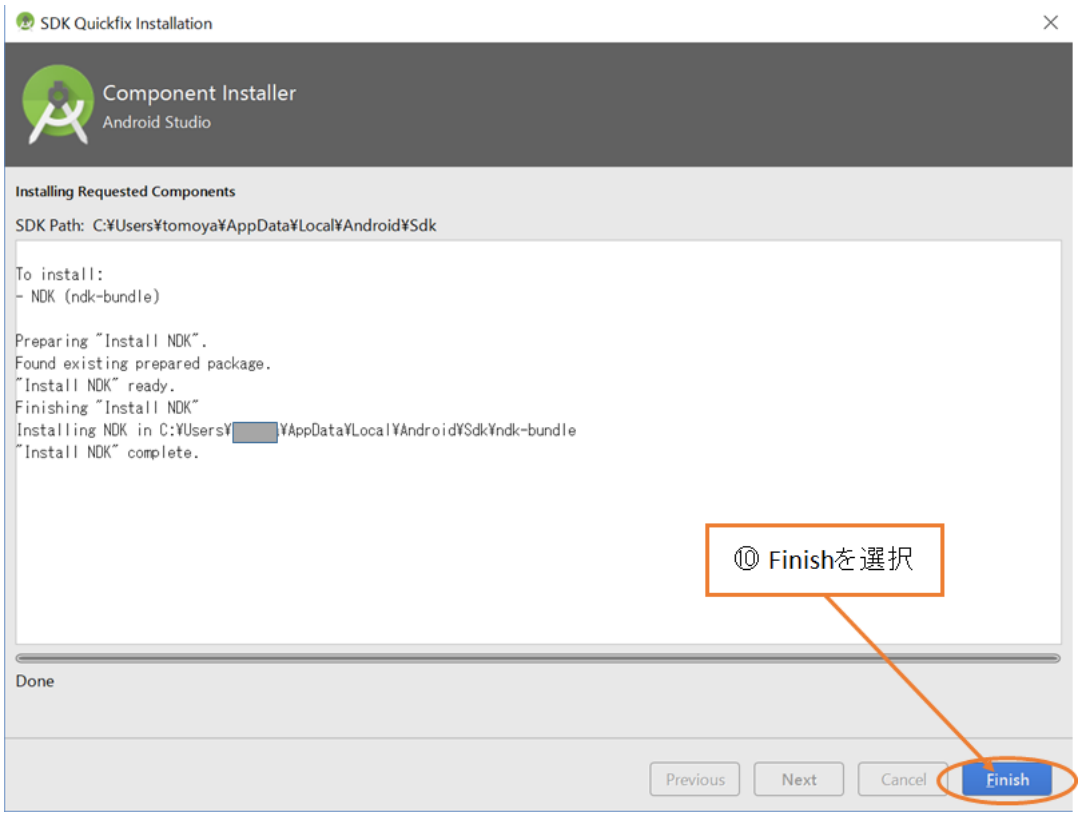
Android NDK のインストール

Android Studio から Android NDK をインストールする。






SDK Quickfix Installation
License Agreement
Android Studio
Licenses
android-sdk-license
NDK
Terms and Conditions
This is the Android Software Development Kit License Agreement
1. Introduction
1.1 The Android Software Development Kit (referred to in the License Agreement as the "SDK" and specifically including the Android system files, packaged APIs, and Google APIs add-ons) is licensed to you subject to the terms of the License Agreement. The License Agreement forms a legally binding contract between you and Google in relation to your use of the SDK.
1.2 "Android" means the Android software stack for devices, as made available under the Android Open Source Project, which is located at the following URL: http://source.android.com/, as updated from time to time.
1.3 A "compatible implementation" means any Android device that (i) complies with the Android Compatibility Definition document, which can be found at the Android compatibility website (http://source.android.com/compatibility) and which may be updated from time to time; and (ii) successfully passes the Android Compatibility Test Suite (CTS).
1.4 "Google" means Google Inc., a Delaware corporation with principal place of business at 1600 Amphitheatre Parkway, Mountain View, CA 94043, United States.
⑧ Acceptを選択
Decline
Accept
⑨ Nextを選択
Previous
Next
Cancel
Finish


SDK Quickfix Installation
Component Installer
Android Studio
Installing Requested Components
SDK Path: C:¥Users¥tomoya¥AppData¥Local¥Android¥Sdk
To install:
- NDK (ndk-bundle)
Preparing "Install NDK".
Found existing prepared package.
"Install NDK" ready.
Finishing "Install NDK"
Installing NDK in C:¥Users¥[redacted]¥AppData¥Local¥Android¥Sdk¥ndk-bundle
"Install NDK" complete.
⑩ Finishを選択
Done
Previous
Next
Cancel
Finish

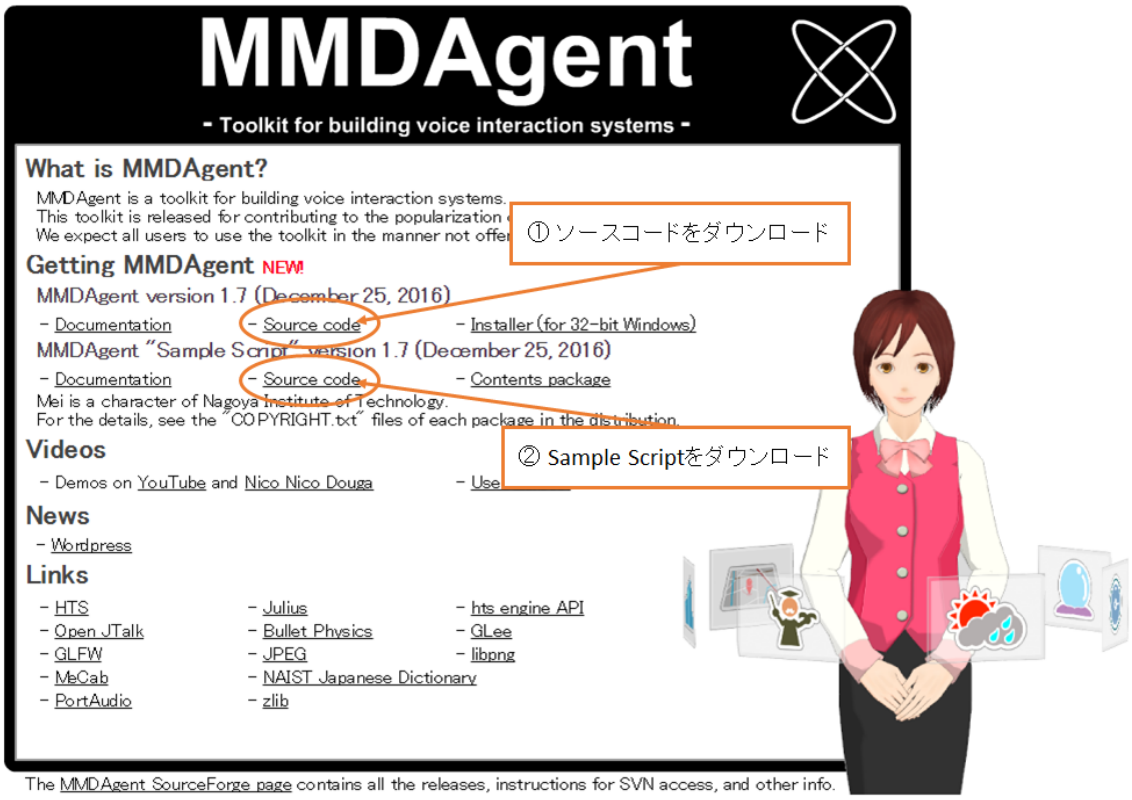
ソースコードと Sample Script の取得

公式サイトから MMDAgent のソースコードと Sample Script をダウンロードし、PC 内の任意の場所に保存する。なお、ダウンロードしたファイルは Zip 圧縮されているため、以降の手順を進めるためには解凍が必要となる。

▼MMDAgent 公式サイト

http://www.mmdagent.jp/

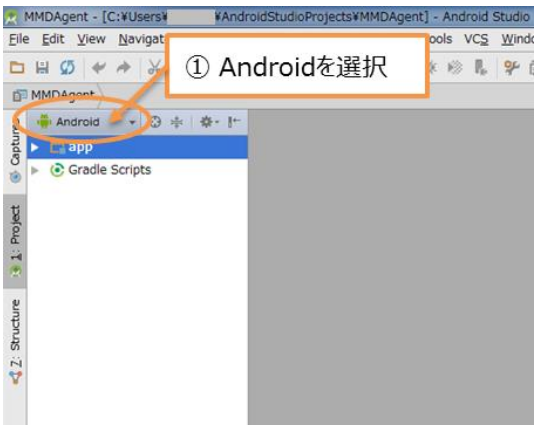
▼手順

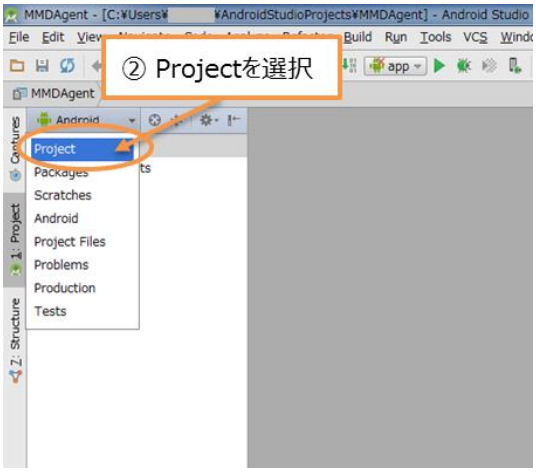


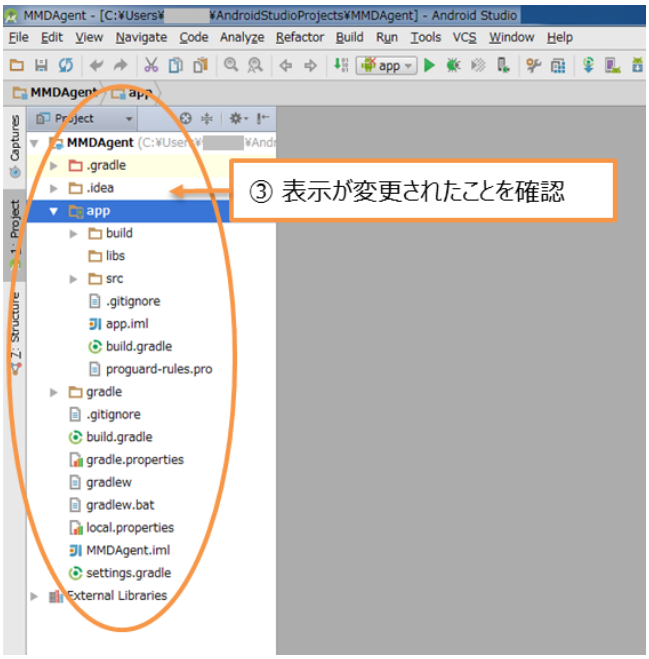
プロジェクトの表示方法を変更

プロジェクトが実際のフォルダ構造と同じ見た目になるよう表示方法を変更する。

▼手順



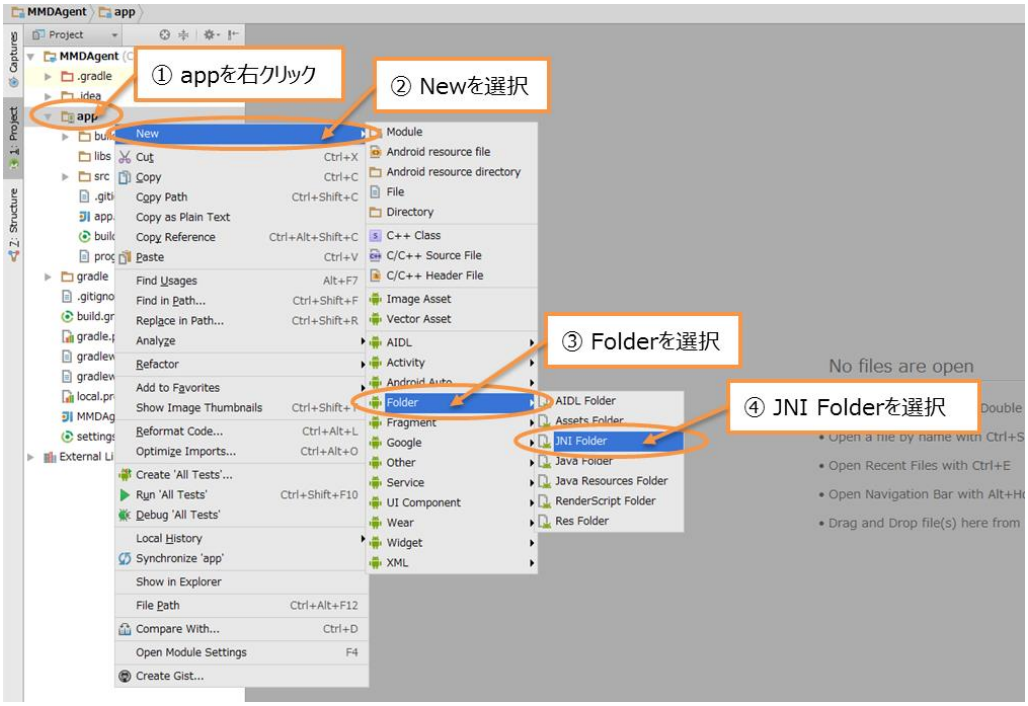


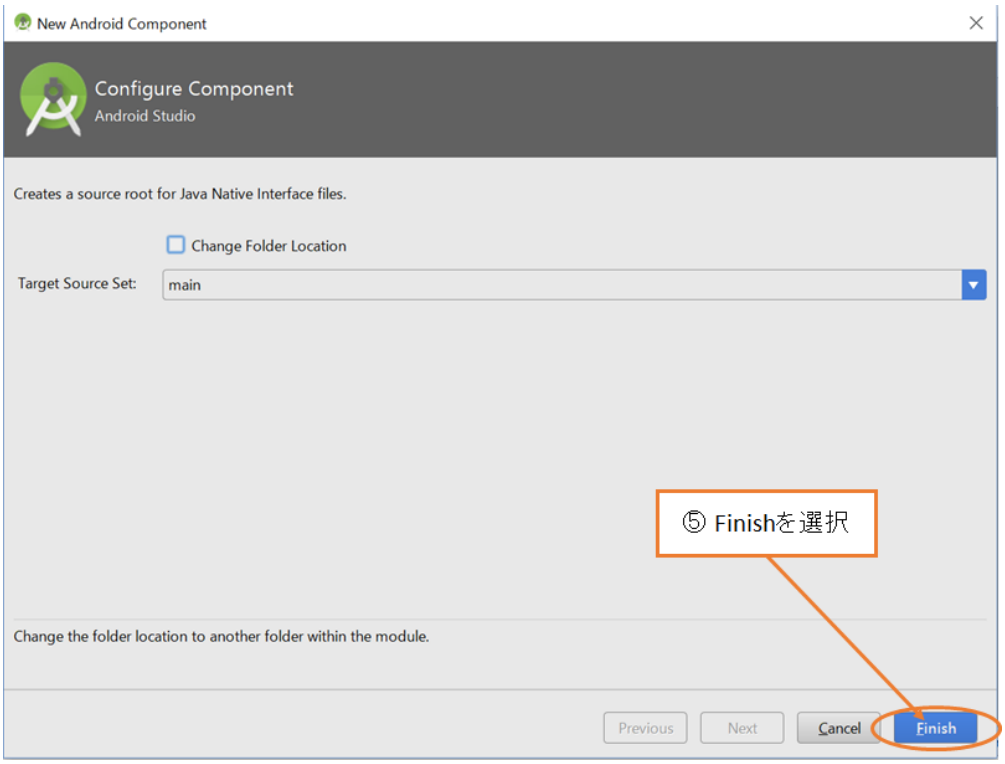


JNI フォルダの作成

MMDAgent のソースコードを格納するための jni フォルダを作成する。

▼手順



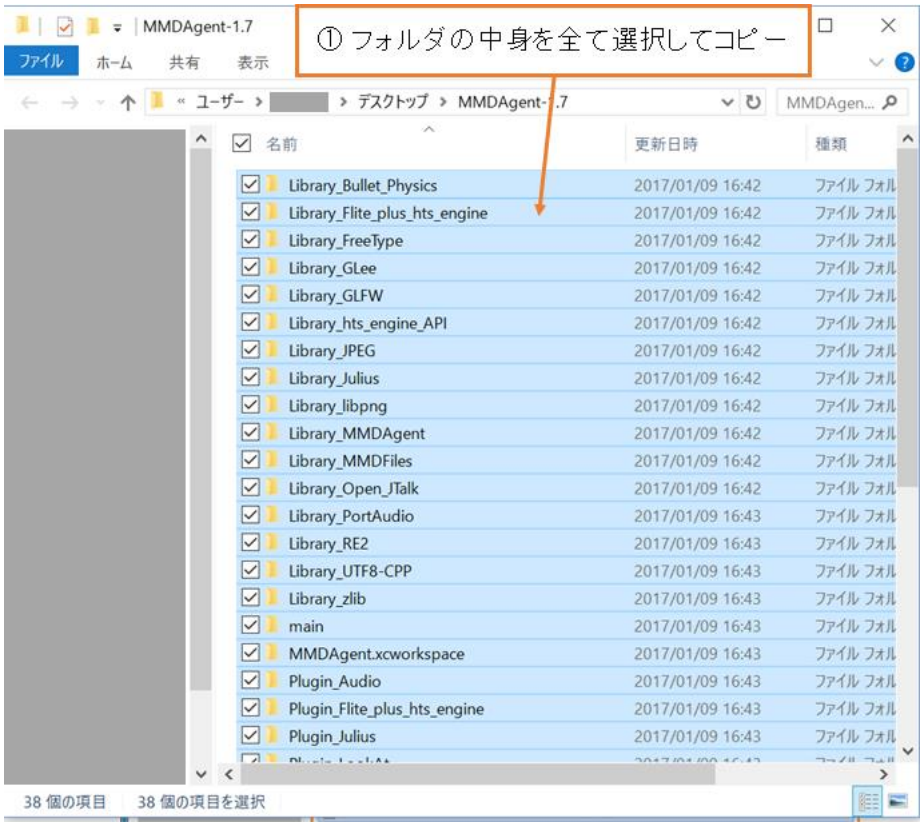


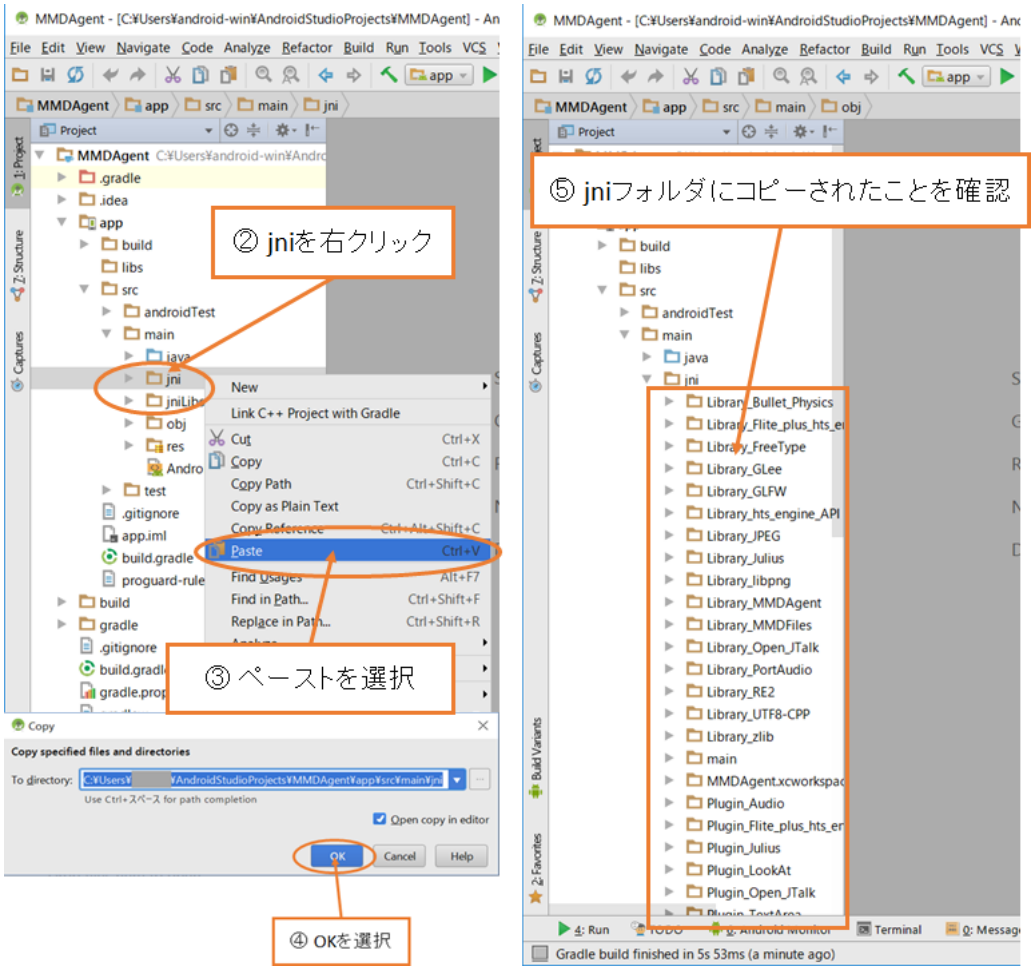
※JNI フォルダは MMDAgent¥app¥src¥main フォルダに作成される。

ソースコードのインポート

ダウンロードしたソースコードの中身を jni フォルダにコピーする。

▼手順（標準エクスプローラー使用時）





▼手順（Android Studio）

[備考]

Android 版では使用されないファイルが含まれているが、開発に影響しないため削除しない。

Android 端末にシステムディレクトリとコンテンツディレクトリを作成

テストにあたっては実行環境となる Android 端末に、MMDAgent のシステムディレクトリとコンテンツディレクトリの作成が必要となる。
（配布するアプリとして完成させる場合は、アプリ側でデータをサーバーからダウンロードして格納するなどする。）

システムディレクトリの作成

Android 端末に AppData ディレクトリを格納したシステムディレクトリを作成する。

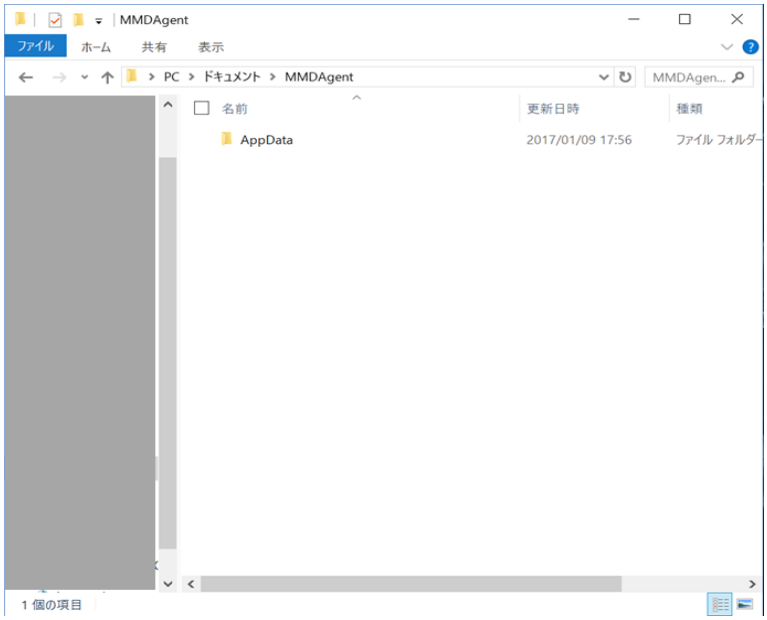
▼手順

1. 転送元フォルダの作成
   PC 内の任意の場所に、転送元となるフォルダをフォルダ名 MMDAgent として作成する。

2. AppData を転送元フォルダにコピー
   ダウンロードしたソースコード内の Release¥AppData フォルダを転送元フォルダにコピーする。

3. 転送元フォルダを実行環境の端末にコピー
   PC 内に作成した転送元フォルダを実行環境の端末にコピーする。
   コピー先のパスには任意の位置を指定できるが、ここでは内蔵ストレージの「*/sdcard*」フォルダを対象に作業する。
   ※端末によっては */sdcard* が内蔵ストレージとは限らない。

▼転送元フォルダの参考画像



コンテンツディレクトリの作成

Android 端末にサンプルスクリプトを格納したコンテンツディレクトリを作成する。

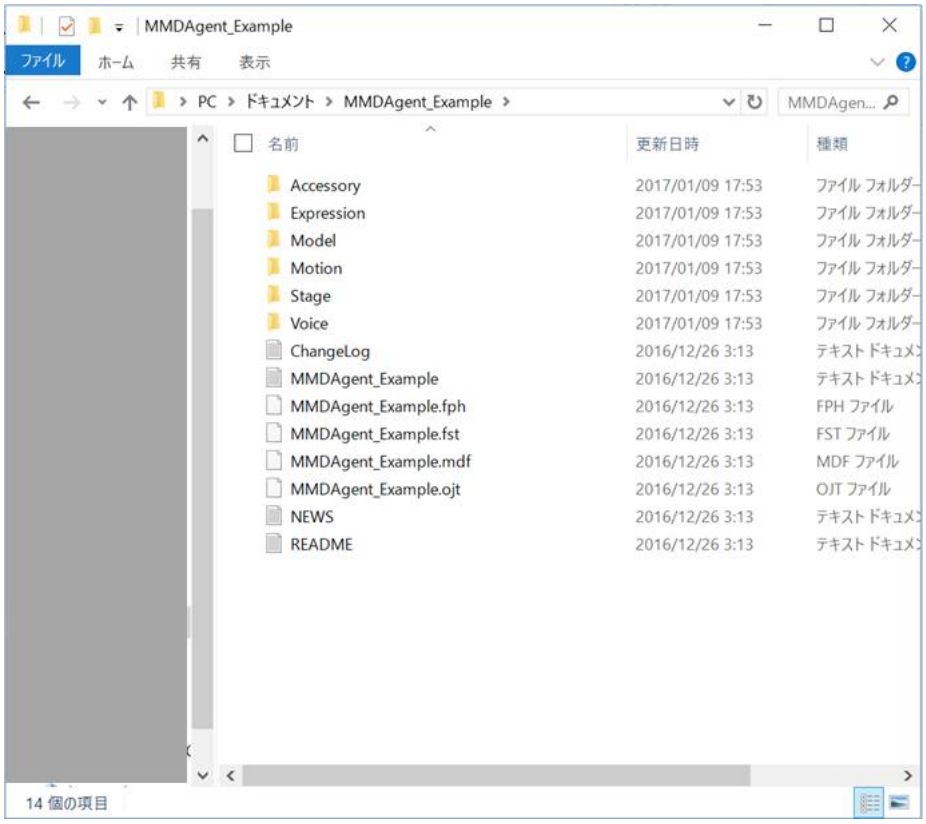
▼手順

1. 転送元フォルダの作成
   PC 内の任意の場所に、転送元となるフォルダをフォルダ名 MMDAgent_Example として作成する。

2. Sample Script を転送元フォルダにコピー
   ダウンロードした Sample Script の中身を転送元フォルダにコピーする。

3. 転送元フォルダを実行環境の端末にコピー
   PC 内に作成した転送元フォルダを実行環境の端末にコピーする。
   コピー先のパスには任意の位置を指定できるが、ここでは内蔵ストレージの「*/sdcard*」フォルダを対象に作業する。
   ※端末によっては */sdcard* が内蔵ストレージとは限らない。

▼転送元フォルダの参考画像



ファイルの修正

Android 版 MMDAgent をビルドするためには、一部のファイルに修正が必要となる。
以下の対象ファイルについて、修正箇所と修正後のファイル内容を参考に修正を行う。

AndroidManifest.xml の修正

▼ファイルのパス
MMDAgent¥app¥src¥main¥AndroidManifest.xml
※同名ファイルが複数存在するため、修正対象を間違えないよう注意する。

▼修正箇所

1. Package name を変更（1 行目）
   Package 属性の値を、プロジェクト作成時に自動的に入力される値に変更する。

2. 外部ストレージの読み込み権限と書き込み権限を追加（4～5 行目）
   外部ストレージに対する読み込み権限と書き込み権限を追加する。
   ※MMDAgent 1.6 以降は書き込み権限の許可が必須。

3. マイクの使用権限を追加（6 行目）
   音声認識でマイクを使用するための権限を追加する。

4. activity タグの追加（15～21 行目）
   NativeActivity が読み込む共有ライブラリの名前を meta-data タグで記述し、
   アクティビティのメイン設定と Android ランチャーへの登録設定を intent-filter タグで記述する。

▼修正後のファイル内容

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.mmdagent">

    <!-- add permission -->
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.RECORD_AUDIO" />

    <application
        android:allowBackup="true"
        android:label="@string/app_name"
        android:icon="@mipmap/ic_launcher"
        android:theme="@style/AppTheme">

        <!-- add NativeActivity -->
        <activity android:name="android.app.NativeActivity" android:label="@string/app_name">
            <meta-data android:name="android.app.lib_name" android:value="main" />
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
</manifest>
```

[備考]
Android 版 MMDAgent を拡張して Java コードを含める場合、application タグに
android:hasCode 属性を追加する必要がある。

・android:hasCode 属性の値について

値	概要
true（既定値）	Java コードを含める。
false	Java コードを含めない。

local.properties の修正

▼ファイルのパス

MMDAgent¥local.properties

▼修正箇所

1. Android NDK のパスを追加（11 行目）
   Android NDK をインストールしたフォルダのパスを追加する。

▼修正後のファイル内容

```
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=C¥:¥¥Users¥¥<ユーザー名>¥¥AppData¥¥Local¥¥Android¥¥Sdk
11 ndk.dir=C¥:¥¥Users¥¥<ユーザー名>¥¥AppData¥¥Local¥¥Android¥¥Sdk¥¥ndk-bundle
```

※<ユーザー名>の部分は自身の環境に合わせて記述する。

※フォルダの区切り文字は [¥¥] と記述する。

build.gradle の修正

▼ファイルのパス

MMDAgent¥app¥build.gradle

※同名ファイルが複数存在するため、修正対象を間違えないよう注意する。

▼修正箇所

1. Package name を変更（9 行目）
   applicationId をプロジェクト作成時に自動的に入力される値に変更する。

2. Android Studio のビルドを無効化（16 行目）
   ndk-build コマンドを使用するため、Android Studio のビルドを無効化する。

3. コンパイルを実行するタスクを追加（31~40 行目）
   ndk-build コマンドを実行するためのタスクを追加する。
   ※33 行目の ndkDir は自身の環境に合わせて記述する。

4. タスクの依存関係を追加（42~44 行目）
   このファイルに記述したタスク間の依存関係を設定する。

5. ndk-build 関連のファイルを削除するタスクを追加（47~55 行目）
   ndk-build コマンドによって生成されたファイルを削除するタスクを追加する。
   ※49 行目の ndkDir は自身の環境に合わせて記述する。

▼修正後のファイル内容

```
1  apply plugin: 'com.android.application'
2  import org.apache.tools.ant.taskdefs.condition.Os
3
4  android {
5      compileSdkVersion 25
6      buildToolsVersion "25.0.2"
7
8      defaultConfig {
9          applicationId "com.example.mmdagent"
10         minSdkVersion 14
11         targetSdkVersion 25
12         versionCode 1
13         versionName "1.0"
14     }
15
16     sourceSets.main.jni.srcDirs = [] // avoid using NdkCompile task
17
18     buildTypes {
19         release {
20             minifyEnabled false
21             proguardFiles(getDefaultProguardFile('proguard-android.txt'), 'proguard-
   rules.pro');
22         }
23     }
24 }
25
26 dependencies {
27     compile fileTree(dir: 'libs', include: ['*.jar'])
28     compile 'com.android.support:appcompat-v7:25.0.1'
29 }
30
31 task buildNative(type:Exec) {
32     //def ndkDir = project.plugins.findPlugin('com.android.application').getNdkFolder()
33     def ndkDir = "C:¥¥Users¥¥<ユーザー名>¥¥AppData¥¥Local¥¥Android¥¥Sdk¥¥ndk-bundle"
34     def jOption = '-j'+Runtime.runtime.availableProcessors()
```

```
35      if(Os.isFamily(Os.FAMILY_WINDOWS)){
36          commandLine("$ndkDir/ndk-build.cmd", jOption, '-C', file('src/main').absolutePath,
    'NDK_APP_LIBS_OUT=jniLibs');
37      }else{
38          commandLine("$ndkDir/ndk-build", jOption, '-C', file('src/main').absolutePath,
    'NDK_APP_LIBS_OUT=jniLibs');
39      }
40  }
41
42  tasks.withType(JavaCompile) {
43      compileTask -> compileTask.dependsOn 'buildNative'
44  }
45
46
47  task cleanNative(type:Exec){
48      //def ndkDir = project.plugins.findPlugin('com.android.application').getNdkFolder()
49      def ndkDir = "C:¥¥Users¥¥<ユーザー名>¥¥AppData¥¥Local¥¥Android¥¥Sdk¥¥ndk-bundle"
50      if(Os.isFamily(Os.FAMILY_WINDOWS)){
51          commandLine("$ndkDir/ndk-build.cmd", 'clean', '-C', file('src/main').absolutePath,
    "NDK_APP_LIBS_OUT=jnilibs");
52      }else{
53          commandLine("$ndkDir/ndk-build", 'clean', '-C', file('src/main').absolutePath,
    "NDK_APP_LIBS_OUT=jnilibs");
54      }
55  }
56
57  cleanNative.dependsOn 'cleanNative'
58  clean.dependsOn 'cleanNative'
```

Android.mk の修正

▼ファイルのパス

MMDAgent¥app¥src¥main¥jni¥Library_MMDAgent¥Android.mk

※同名ファイルが複数存在するため、修正対象を間違えないよう注意する。

▼修正箇所

1. DMMDAGENT_OVERWRITEEXEFILE のパスを変更（38 行目）
   記述されているパスをコンテンツディレクトリ内の fst ファイルのパスに変更する。
   なお、exe ファイルのパスとして参照されるため、拡張子部分は exe として記述する。

```
¥"/sdcard/MMDAgent_Example/MMDAgent_Example.exe¥"
```

   ※本書の指示通りに作業している場合は上記パスを入力する。

2. DMMDAGENT_OVERWRITECONFIGFILE のパスを変更（39 行目）
   記述されているパスをコンテンツディレクトリ内の mdf ファイルのパスに変更する。

```
¥"/sdcard/MMDAgent_Example/MMDAgent_Example.mdf¥"
```

   ※本書の指示通りに作業している場合は上記パスを入力する。

3. DMMDAGENT_OVERWRITESYSDATADIR のパスを変更（40 行目）
   記述されているパスをシステムディレクトリ内の AppData フォルダのパスに変更する。

```
¥"/sdcard/MMDAgent/AppData¥"
```

   ※本書の指示通りに作業している場合は上記パスを入力する。

4. DMMDAGENT_OVERWRITEPLUGINDIR のパスを変更（41 行目）
   記述されているパスを作成したプロジェクトの Package name に合わせて変更する。

```
¥"/data/data/<packageName>/lib¥"
```

   ※<packageName>の部分に、プロジェクト作成時に自動入力された値を入力する。

▼修正後のファイル内容

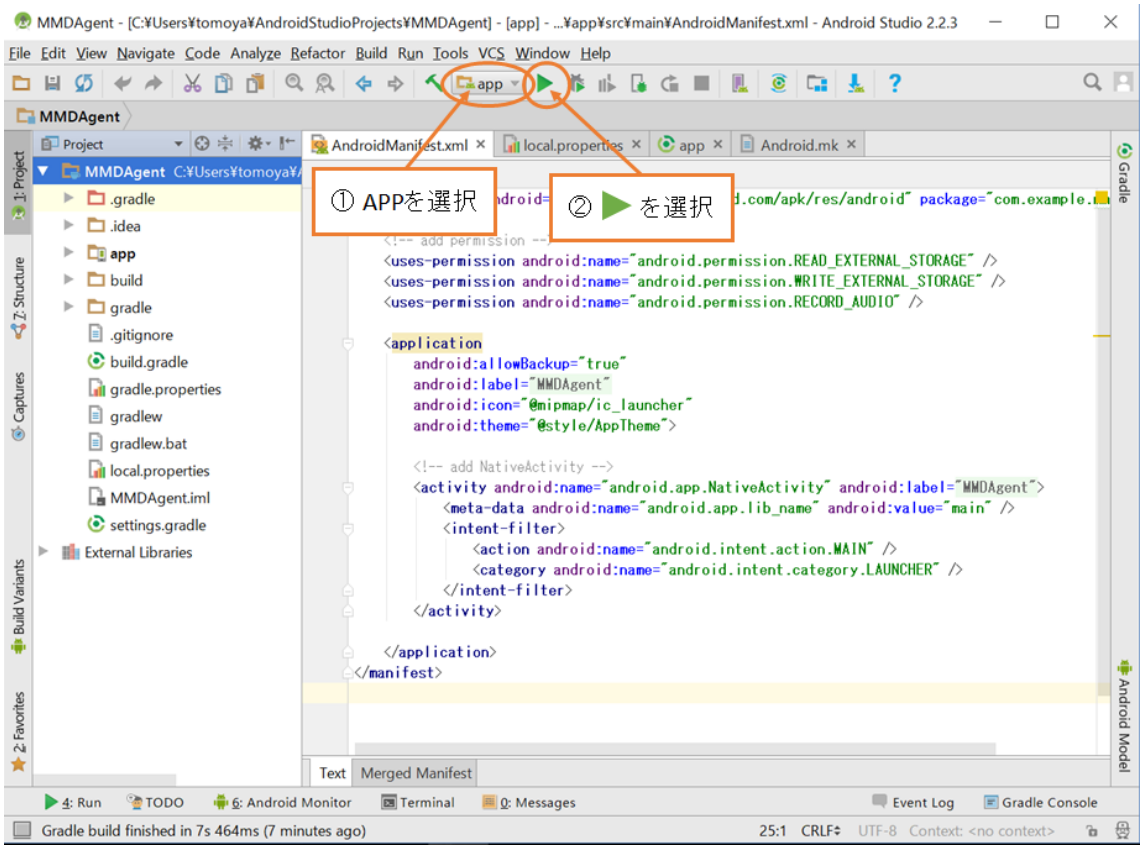
```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

LOCAL_MODULE     := MMDAgent
LOCAL_SRC_FILES  := src/lib/BoneController.cpp ¥
                    src/lib/LipSync.cpp ¥
                    src/lib/LogText.cpp ¥
                    src/lib/Message.cpp ¥
                    src/lib/MMDAgent.cpp ¥
                    src/lib/MMDAgent_utils.cpp ¥
                    src/lib/MotionStocker.cpp ¥
                    src/lib/Option.cpp ¥
                    src/lib/PMDObject.cpp ¥
                    src/lib/Plugin.cpp ¥
                    src/lib/Render.cpp ¥
                    src/lib/ScreenWindow.cpp ¥
                    src/lib/Stage.cpp ¥
                    src/lib/FreeTypeGL.cpp ¥
                    src/lib/TileTexture.cpp ¥
                    src/lib/Timer.cpp
LOCAL_C_INCLUDES := $(LOCAL_PATH)/include ¥
                    $(LOCAL_PATH)/../Library_JPEG/include ¥
                    $(LOCAL_PATH)/../Library_Bullet_Physics/include ¥
                    $(LOCAL_PATH)/../Library_GLee/include ¥
                    $(LOCAL_PATH)/../Library_libpng/include ¥
                    $(LOCAL_PATH)/../Library_zlib/include ¥
                    $(LOCAL_PATH)/../Library_MMDFiles/include ¥
                    $(LOCAL_PATH)/../Library_GLFW/include ¥
                    $(LOCAL_PATH)/../Library_FreeType/include ¥
                    $(LOCAL_PATH)/../Library_UTF8-CPP/include
LOCAL_CFLAGS     += -DMMDAGENT_DONTRENDERDEBUG ¥
                    -DMMDAGENT_DONTUSESHADOWMAP ¥
                    -DMMDAGENT_DONTPICKMODEL ¥
                    -DMMDAGENT_DONTUSEMOUSE ¥
```

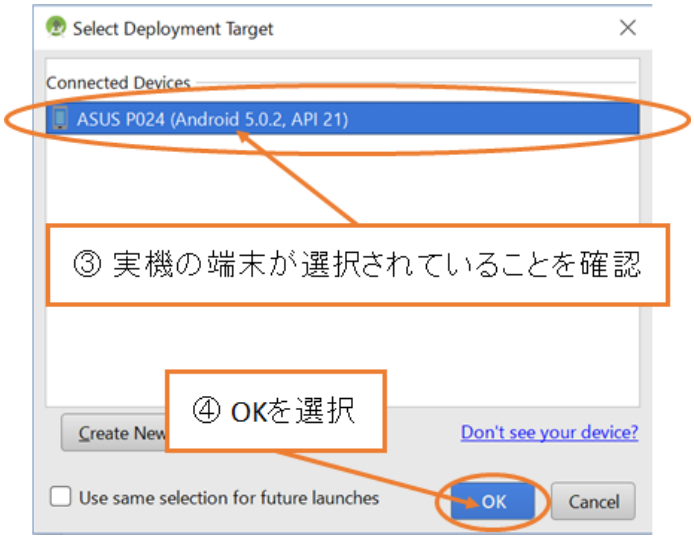
```
36                      -DMMDAGENT_DONTUSEWINDOW ¥
37                      -DMMDAGENT ¥
38                      -DMMDAGENT_OVERWRITEEXEFILE="¥"/sdcard/MMDAgent_Example/MMDAgent_Example.exe¥"" ¥
39                      -DMMDAGENT_OVERWRITECONFIGFILE="¥"/sdcard/MMDAgent_Example/MMDAgent_Example.mdf¥"" ¥
40                      -DMMDAGENT_OVERWRITESYSDATADIR="¥"/sdcard/MMDAgent/AppData¥"" ¥
41                      -DMMDAGENT_OVERWRITEPLUGINDIR="¥"/data/data/com.example.mmdagent/lib¥""
42
43  include $(BUILD_STATIC_LIBRARY)
44
```

ソースコードのビルド/実行

ソースコードをビルドし、Android 端末で実行する。

▶手順





※ビルド時にエラーが起きなければ、自動的に上記ウィンドウが出現する。


LTE
100
16:57
⑥ Android端末でMMDAgentが表示されれば成功